



Panduan Developer

# AWS IoT Events



# AWS IoT Events: Panduan Developer

Copyright © 2026 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 di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

---

# Table of Contents

.....	viii
Apa itu AWS IoT Events? .....	1
Manfaat dan fitur .....	1
Kasus penggunaan .....	2
Memantau dan memelihara perangkat jarak jauh .....	3
Kelola robot industri .....	3
Lacak sistem otomatisasi bangunan .....	3
AWS IoT Events Akhir dari Dukungan .....	4
Pertimbangan saat bermigrasi menjauh dari AWS IoT Events .....	4
Model detektor .....	5
Membandingkan arsitektur .....	5
Langkah 1: Ekspor (Opsional) AWS IoT Events konfigurasi model detektor .....	7
Langkah 2: Buat peran IAM .....	8
Langkah 3: Buat Amazon Kinesis Data Streams .....	10
Langkah 4: Buat atau perbarui aturan perutean pesan MQTT .....	11
Langkah 5: Dapatkan titik akhir untuk topik MQTT tujuan .....	12
Langkah 6: Buat tabel Amazon DynamoDB .....	13
Langkah 7: Buat AWS Lambda fungsi (konsol) .....	13
Langkah 8: Tambahkan pemicu Amazon Kinesis Data Streams .....	22
Langkah 9: Uji konsumsi data dan fungsionalitas keluaran (AWS CLI) .....	23
Alarm .....	23
Membandingkan arsitektur .....	23
Langkah 1: Aktifkan pemberitahuan MQTT pada properti aset .....	24
Langkah 2: Buat AWS Lambda Fungsi .....	25
Langkah 3: Buat AWS IoT Core aturan perutean pesan .....	27
Langkah 4: Lihat CloudWatch metrik .....	27
Langkah 5: Buat CloudWatch alarm .....	28
Langkah 6: (Opsional) impor CloudWatch alarm ke AWS IoT SiteWise .....	28
Penyiapan .....	29
Menyiapkan sebuah Akun AWS .....	29
Mendaftar untuk Akun AWS .....	29
Menyiapkan izin untuk AWS IoT Events .....	29
Izin tindakan .....	30
Mengamankan data masukan .....	32

Kebijakan peran CloudWatch pencatatan Amazon .....	33
Kebijakan peran pesan Amazon SNS .....	35
Memulai .....	37
Prasyarat .....	39
Buat masukan .....	40
Buat file masukan JSON .....	40
Buat dan konfigurasi masukan .....	40
Buat masukan dalam Model Detektor .....	41
Buat model detektor .....	42
Uji model detektor .....	49
Praktik terbaik .....	53
Aktifkan CloudWatch pencatatan Amazon saat mengembangkan model AWS IoT Events detektor .....	53
Publikasikan secara teratur untuk menyimpan model detektor Anda saat bekerja di AWS IoT Events konsol .....	54
Tutorial .....	55
Menggunakan AWS IoT Events untuk memantau perangkat IoT Anda .....	55
Bagaimana Anda tahu status mana yang Anda butuhkan dalam model detektor? .....	57
Bagaimana Anda tahu jika Anda memerlukan satu contoh detektor atau beberapa? .....	59
step-by-stepContoh sederhana .....	59
Buat input untuk menangkap data perangkat .....	62
Buat model detektor untuk mewakili status perangkat .....	63
Kirim pesan sebagai input ke detektor .....	66
Pembatasan dan batasan model detektor .....	70
Contoh komentar: Kontrol suhu HVAC .....	73
Definisi input untuk model detektor .....	74
Buat definisi model detektor .....	78
Gunakan BatchUpdateDetector .....	98
Gunakan BatchPutMessage untuk input .....	100
Menelan pesan MQTT .....	102
Hasilkan pesan Amazon SNS .....	104
Konfigurasi DescribeDetector API .....	105
Gunakan mesin AWS IoT Core aturan .....	107
Tindakan yang didukung .....	111
Gunakan tindakan bawaan .....	112
Atur tindakan pengatur waktu .....	112

Atur ulang tindakan pengatur waktu .....	113
Hapus tindakan pengatur waktu .....	113
Tetapkan tindakan variabel .....	113
Bekerja dengan AWS layanan lain .....	114
AWS IoT Core .....	115
AWS IoT Events .....	116
AWS IoT SiteWise .....	117
Amazon DynamoDB .....	120
Amazon DynamoDB (v2) .....	122
Amazon Data Firehose .....	123
AWS Lambda .....	124
Layanan Notifikasi Sederhana Amazon .....	125
Amazon Simple Queue Service .....	126
Ekspresi .....	128
Sintaks untuk memfilter data perangkat .....	128
Literal .....	128
Operator .....	128
Fungsi untuk ekspresi .....	130
Referensi untuk input dan variabel dalam ekspresi .....	135
Templat substitusi .....	137
Penggunaan .....	138
Menulis AWS IoT Events ekspresi .....	138
Contoh model detektor .....	141
Kontrol suhu HVAC .....	141
Cerita latar belakang .....	141
Definisi masukan .....	142
Definisi model detektor .....	144
BatchPutMessage contoh .....	162
BatchUpdateDetector contoh .....	168
AWS IoT Core aturan mesin .....	170
Crane .....	173
Kirim perintah .....	174
Model detektor .....	175
Masukan .....	182
Pesan .....	183
Contoh: Deteksi peristiwa dengan sensor .....	184

Perangkat HeartBeat .....	187
Alarm ISA .....	189
Alarm sederhana .....	199
Pemantauan dengan alarm .....	204
Bekerja dengan AWS IoT SiteWise .....	204
Akui aliran .....	204
Membuat model alarm .....	205
Persyaratan .....	205
Membuat model alarm (konsol) .....	206
Menanggapi alarm .....	209
Mengelola pemberitahuan alarm .....	210
Membuat fungsi Lambda .....	211
Menggunakan fungsi Lambda .....	220
Mengelola penerima alarm .....	221
Keamanan .....	223
Manajemen identitas dan akses .....	224
Audiens .....	224
Mengautentikasi dengan identitas .....	224
Mengelola akses menggunakan kebijakan .....	226
Lebih lanjut tentang identitas dan manajemen akses .....	227
Bagaimana AWS IoT Events bekerja dengan IAM .....	227
Contoh kebijakan berbasis identitas .....	231
Pencegahan deputi kebingungan lintas layanan untuk AWS IoT Events .....	238
Pemecahan masalah .....	242
Memantau .....	244
Alat yang tersedia untuk memantau AWS IoT Events .....	245
Pemantauan AWS IoT Events dengan Amazon CloudWatch .....	247
Pencatatan panggilan AWS IoT Events API dengan AWS CloudTrail .....	248
Validasi kepatuhan .....	268
Ketahanan .....	268
Keamanan infrastruktur .....	268
Kuota .....	270
Penandaan .....	271
Dasar-dasar tag .....	271
Pembatasan dan batasan tanda .....	272
Menggunakan tanda dengan kebijakan IAM .....	272

Pemecahan Masalah .....	276
Biasa AWS IoT Events masalah dan solusi .....	276
Kesalahan pembuatan model detektor .....	277
Pembaruan dari model detektor yang dihapus .....	277
Kegagalan pemicu tindakan (saat memenuhi suatu kondisi) .....	277
Kegagalan pemicu tindakan (saat melewati ambang batas) .....	278
Penggunaan status salah .....	278
Pesan koneksi .....	278
InvalidRequestException pesan .....	278
Kesalahan Amazon CloudWatch Logs <code>Action.setTimer</code> .....	279
Kesalahan CloudWatch payload Amazon .....	280
Tipe data yang tidak kompatibel .....	281
Gagal mengirim pesan ke AWS IoT Events .....	282
Memecahkan masalah model detektor .....	283
Informasi diagnostik .....	283
Analisis model detektor (Konsol) .....	297
Menganalisis model detektor (AWS CLI) .....	298
Commands .....	304
AWS IoT Events tindakan .....	304
AWS IoT Events data .....	304
Riwayat dokumen .....	305
Pembaruan lebih awal .....	306

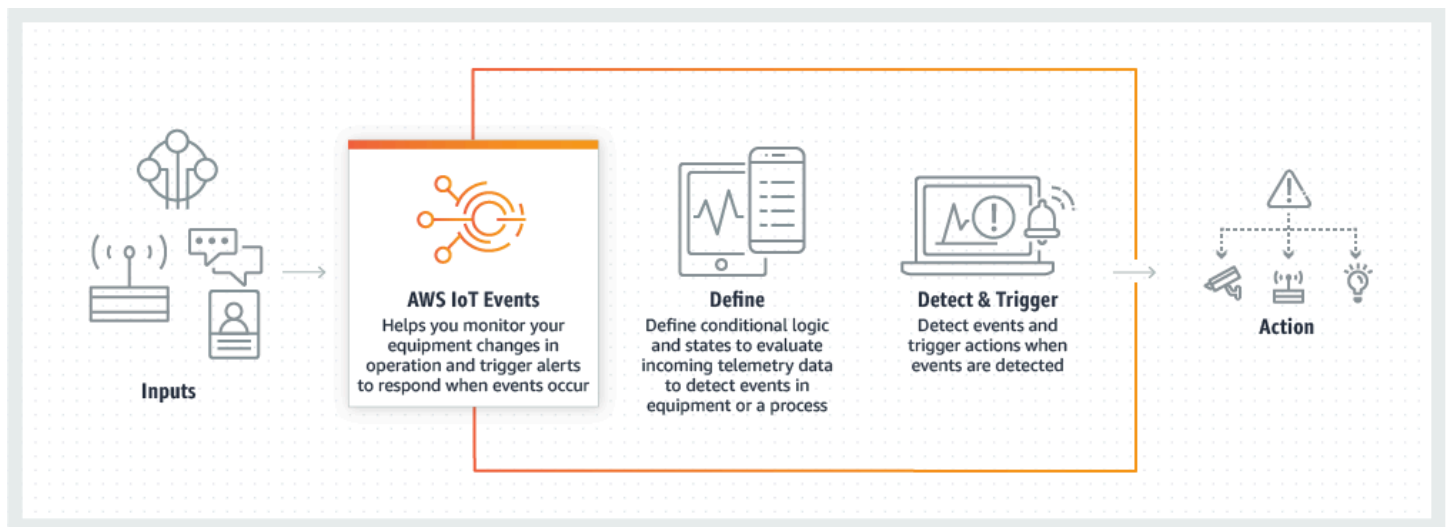
Pemberitahuan akhir dukungan: Pada 20 Mei 2026, AWS akan mengakhiri dukungan untuk AWS IoT Events. Setelah 20 Mei 2026, Anda tidak akan lagi dapat mengakses AWS IoT Events konsol atau AWS IoT Events sumber daya. Untuk informasi selengkapnya, lihat [AWS IoT Events akhir dukungan](#).

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.

# Apa itu AWS IoT Events?

AWS IoT Events memungkinkan Anda untuk memantau peralatan atau armada perangkat Anda untuk kegagalan atau perubahan dalam operasi, dan untuk memicu tindakan ketika peristiwa tersebut terjadi. AWS IoT Events terus memantau data sensor IoT dari perangkat, proses, aplikasi, dan AWS layanan lain untuk mengidentifikasi peristiwa penting sehingga Anda dapat mengambil tindakan.

Gunakan AWS IoT Events untuk membangun aplikasi pemantauan peristiwa kompleks di AWS Cloud yang dapat Anda akses melalui AWS IoT Events konsol atau API.



## Topik

- [Manfaat dan fitur](#)
- [Kasus penggunaan](#)

## Manfaat dan fitur

Terima masukan dari berbagai sumber

AWS IoT Events menerima input dari banyak sumber data telemetri IoT. Ini termasuk perangkat sensor, aplikasi manajemen, dan AWS IoT layanan lainnya, seperti AWS IoT Core dan AWS IoT Analytics. Anda dapat mendorong input data telemetri apa pun AWS IoT Events dengan menggunakan antarmuka API standar (BatchPutMessageAPI) atau konsol. AWS IoT Events

Untuk informasi lebih lanjut tentang memulai AWS IoT Events, lihat [Memulai dengan AWS IoT Events konsol](#).

## Gunakan ekspresi logis sederhana untuk mengenali pola peristiwa yang kompleks

AWS IoT Events dapat mengenali pola peristiwa yang melibatkan beberapa input dari satu perangkat atau aplikasi IoT, atau dari beragam peralatan dan banyak sensor independen. Ini sangat berguna karena setiap sensor dan aplikasi memberikan informasi penting. Tetapi hanya dengan menggabungkan beragam sensor dan data aplikasi Anda dapat memperoleh gambaran lengkap tentang kinerja dan kualitas operasi. Anda dapat mengonfigurasi AWS IoT Events detektor untuk mengenali peristiwa ini menggunakan ekspresi logis sederhana alih-alih kode kompleks.

Untuk informasi lebih lanjut tentang ekspresi logis, lihat [Ekspresi untuk memfilter, mengubah, dan memproses data peristiwa](#).

## Memicu tindakan berdasarkan peristiwa

AWS IoT Events memungkinkan Anda untuk langsung memicu tindakan di Amazon Simple Notification Service (Amazon SNS),, Lambda, Amazon SQS AWS IoT Core, dan Amazon Kinesis Firehose. Anda juga dapat memicu AWS Lambda fungsi menggunakan mesin AWS IoT aturan yang memungkinkan untuk mengambil tindakan menggunakan layanan lain, seperti Connect Customer, atau aplikasi perencanaan sumber daya perusahaan (ERP) Anda sendiri.

AWS IoT Events menyertakan pustaka tindakan bawaan yang dapat Anda ambil, dan juga memungkinkan Anda untuk menentukan sendiri.

Untuk mempelajari selengkapnya tentang memicu tindakan berdasarkan peristiwa, lihat [Tindakan yang didukung untuk menerima data dan memicu tindakan di AWS IoT Events](#).

## Skala otomatis untuk memenuhi permintaan armada Anda

AWS IoT Events skala secara otomatis saat Anda menghubungkan perangkat homogen. Anda dapat menentukan detektor satu kali untuk jenis perangkat tertentu, dan layanan akan secara otomatis menskalakan dan mengelola semua instance perangkat yang terhubung AWS IoT Events.

Untuk mengeksplorasi contoh model detektor, lihat [AWS IoT Events contoh model detektor](#).

## Kasus penggunaan

AWS IoT Events memiliki banyak kegunaan. Berikut adalah beberapa contoh kasus penggunaan.

## Memantau dan memelihara perangkat jarak jauh

Memantau armada mesin yang dikerahkan dari jarak jauh dapat menjadi tantangan, terutama ketika kerusakan terjadi tanpa konteks yang jelas. Jika satu mesin berhenti berfungsi, ini mungkin berarti mengganti seluruh unit pemrosesan atau mesin. Tapi ini tidak berkelanjutan. Dengan AWS IoT Events Anda dapat menerima pesan dari beberapa sensor pada setiap mesin untuk membantu Anda mendiagnosis masalah tertentu dari waktu ke waktu. Alih-alih mengganti seluruh unit, Anda sekarang memiliki informasi yang diperlukan untuk mengirim teknisi dengan bagian yang tepat yang perlu diganti. Dengan jutaan mesin, penghematan dapat menambah hingga jutaan dolar, menurunkan total biaya Anda untuk memiliki atau memelihara setiap mesin.

## Kelola robot industri

Menyebarkan robot di fasilitas Anda untuk mengotomatiskan pergerakan paket dapat sangat meningkatkan efisiensi. Untuk meminimalkan biaya, robot dapat dilengkapi dengan sensor sederhana dan murah yang melaporkan data ke cloud. Namun, dengan lusinan sensor dan ratusan mode operasi, mendeteksi masalah secara real time dapat menjadi tantangan. Dengan menggunakan AWS IoT Events, Anda dapat membangun sistem ahli yang memproses data sensor ini di cloud, membuat peringatan untuk memberi tahu staf teknis secara otomatis jika kegagalan sudah dekat.

## Lacak sistem otomatisasi bangunan

Di pusat data, pemantauan suhu tinggi dan kelembaban rendah membantu mencegah kegagalan peralatan. Sensor sering dibeli dari banyak produsen dan masing-masing jenis dilengkapi dengan perangkat lunak manajemennya sendiri. Namun, perangkat lunak manajemen dari vendor yang berbeda terkadang tidak kompatibel, sehingga sulit untuk mendeteksi masalah. Dengan menggunakan AWS IoT Events, Anda dapat mengatur peringatan untuk memberi tahu analis operasi Anda tentang masalah dengan sistem pemanas dan pendingin Anda jauh sebelum kegagalan. Dengan cara ini, Anda dapat mencegah penutupan pusat data yang tidak terjadwal yang akan menelan biaya ribuan dolar dalam penggantian peralatan dan potensi pendapatan yang hilang.

# AWS IoT Events Akhir dari Dukungan

Setelah mempertimbangkan dengan cermat, kami memutuskan untuk mengakhiri dukungan untuk AWS IoT Events layanan ini, efektif 20 Mei 2026. AWS IoT Events tidak akan lagi menerima pelanggan baru mulai 20 Mei 2025. Sebagai pelanggan lama dengan akun yang mendaftar untuk layanan sebelum 20 Mei 2025, Anda dapat terus menggunakan AWS IoT Events fitur. Setelah 20 Mei 2026, Anda tidak akan dapat lagi menggunakannya AWS IoT Events.

Halaman ini memberikan instruksi dan pertimbangan bagi AWS IoT Events pelanggan untuk beralih ke solusi alternatif untuk memenuhi kebutuhan bisnis Anda.

## Note

Solusi yang disajikan dalam panduan ini dimaksudkan untuk berfungsi sebagai contoh ilustratif, bukan sebagai pengganti fungsionalitas yang siap produksi. AWS IoT Events Sesuaikan kode, alur kerja, dan AWS sumber daya terkait dengan kebutuhan bisnis Anda.

## Topik

- [Pertimbangan saat bermigrasi menjauh dari AWS IoT Events](#)
- [Prosedur migrasi untuk model detektor di AWS IoT Events](#)
- [Prosedur migrasi untuk AWS IoT SiteWise alarm di AWS IoT Events](#)

## Pertimbangan saat bermigrasi menjauh dari AWS IoT Events

- Menerapkan praktik terbaik keamanan, termasuk menggunakan peran IAM dengan hak istimewa paling sedikit untuk setiap komponen dan mengenkripsi data saat istirahat dan dalam perjalanan. Untuk informasi selengkapnya tentang administrator, lihat [Praktik terbaik keamanan di IAM](#) di dalam Panduan Pengguna IAM.
- Pertimbangkan jumlah pecahan untuk aliran Kinesis berdasarkan persyaratan konsumsi data Anda. Untuk informasi selengkapnya tentang pecahan Kinesis, lihat [terminologi dan konsep Amazon Kinesis Data Streams](#) di Panduan Pengembang Amazon Kinesis Data Streams.
- Siapkan pemantauan dan debugging CloudWatch komprehensif menggunakan metrik dan log. Untuk informasi lebih lanjut, lihat [Apa itu CloudWatch?](#) di Panduan CloudWatch Pengguna Amazon.

- Pertimbangkan struktur penanganan kesalahan Anda, termasuk cara mengelola pesan yang gagal diproses berulang kali, menerapkan kebijakan coba ulang, dan menyiapkan proses untuk mengisolasi dan menganalisis pesan yang bermasalah.
- Gunakan [Kalkulator AWS Harga](#) untuk memperkirakan biaya untuk kasus penggunaan spesifik Anda.

## Prosedur migrasi untuk model detektor di AWS IoT Events

Bagian ini menjelaskan solusi alternatif yang menghadirkan fungsionalitas model detektor serupa saat Anda AWS IoT Events bermigrasi.

Anda dapat memigrasikan konsumsi data melalui AWS IoT Core aturan ke kombinasi layanan lain. AWS Alih-alih menelan data melalui [BatchPutMessage](#) API, data dapat dirutekan ke topik MQTT. AWS IoT Core

Pendekatan migrasi ini memanfaatkan topik AWS IoT Core MQTT sebagai titik masuk untuk data IoT Anda, menggantikan input langsung ke AWS IoT Events Topik MQTT dipilih karena beberapa alasan utama. Mereka menawarkan kompatibilitas yang luas dengan perangkat IoT karena penggunaan MQTT yang meluas di industri. Topik-topik ini dapat menangani volume pesan yang tinggi dari berbagai perangkat, memastikan skalabilitas. Mereka juga memberikan fleksibilitas dalam perutean dan pemfilteran pesan berdasarkan konten atau jenis perangkat. Selain itu, topik AWS IoT Core MQTT terintegrasi secara mulus dengan AWS layanan lain, memfasilitasi proses migrasi.

Data mengalir dari topik MQTT ke dalam arsitektur yang menggabungkan Amazon Kinesis Data Streams, AWS Lambda fungsi, tabel Amazon DynamoDB, dan jadwal Amazon. EventBridge Kombinasi layanan ini mereplikasi dan meningkatkan fungsionalitas yang sebelumnya disediakan oleh AWS IoT Events, menawarkan Anda lebih banyak fleksibilitas dan kontrol atas jalur pemrosesan data IoT Anda.

## Membandingkan arsitektur

AWS IoT Events Arsitektur saat ini menyerap data melalui AWS IoT Core aturan dan BatchPutMessage API. Arsitektur ini digunakan AWS IoT Core untuk konsumsi data dan penerbitan acara, dengan pesan yang dirutekan melalui AWS IoT Events input ke model detektor yang menentukan logika status. Peran IAM mengelola izin yang diperlukan.

Solusi baru dipertahankan AWS IoT Core untuk konsumsi data (sekarang dengan topik MQTT input dan output khusus). Ini memperkenalkan Kinesis Data Streams untuk partisi data dan fungsi Lambda

evaluator untuk logika keadaan. Status perangkat sekarang disimpan dalam tabel DynamoDB, dan peran IAM yang disempurnakan mengelola izin di seluruh layanan ini.

Tujuan	Solusi	Perbedaan
Penyerapan data - Menerima data dari perangkat IoT	AWS IoT Core	Sekarang membutuhkan dua topik MQTT yang berbeda: satu untuk menelan data perangkat dan satu lagi untuk menerbitkan acara keluaran
Arah pesan - Merutekan pesan masuk ke layanan yang sesuai	AWS IoT Core aturan perutean pesan	Mempertahankan fungsionalitas perutean yang sama tetapi sekarang mengarahkan pesan ke Kinesis Data Streams, bukan AWS IoT Events
Pemrosesan data - Menangani dan mengatur aliran data yang masuk	Kinesis Data Streams	Menggantikan fungsionalitas AWS IoT Events input, menyediakan konsumsi data dengan partisi ID perangkat untuk pemrosesan pesan
Evaluasi logika — Memproses perubahan status dan memicu tindakan	Penilai Lambda	Mengganti model AWS IoT Events detektor, menyediakan evaluasi logika status yang dapat disesuaikan melalui kode alih-alih alur kerja visual
Manajemen negara - Mempertahankan status perangkat	Tabel DynamoDB	Komponen baru yang menyediakan penyimpanan status perangkat yang persisten, menggantikan manajemen AWS IoT Events status internal
Keamanan - Mengelola izin layanan	IAM Role	Izin yang diperbarui sekarang mencakup akses ke Kinesis Data Streams, DynamoDB, dan selain izin yang ada EventBridge AWS IoT Core

## Langkah 1: Ekspor (Opsional) AWS IoT Events konfigurasi model detektor

Sebelum membuat sumber daya baru, ekspor definisi model AWS IoT Events detektor Anda. Ini berisi logika pemrosesan acara Anda dan dapat berfungsi sebagai referensi historis untuk menerapkan solusi baru Anda.

### Console

Dengan menggunakan AWS IoT Events Konsol Manajemen AWS, lakukan langkah-langkah berikut untuk mengekspor konfigurasi model detektor Anda:

Untuk mengekspor model detektor menggunakan Konsol Manajemen AWS

1. Masuk ke [Konsol AWS IoT Events](#).
2. Di panel navigasi kiri, pilih Model detektor.
3. Pilih model detektor untuk diekspor.
4. Pilih Ekspor. Baca pesan informasi mengenai output dan kemudian pilih Ekspor lagi.
5. Ulangi proses untuk setiap model detektor yang ingin Anda ekspor.

File yang berisi output JSON dari model detektor Anda ditambahkan ke folder unduhan browser Anda. Anda dapat menyimpan setiap konfigurasi model detektor secara opsional untuk menyimpan data historis.

### AWS CLI

Menggunakan AWS CLI, jalankan perintah berikut untuk mengekspor konfigurasi model detektor Anda:

Untuk mengekspor model detektor menggunakan AWS CLI

1. Daftar semua model detektor di akun Anda:

```
aws iotevents list-detector-models
```

2. Untuk setiap model detektor, ekspor konfigurasinya dengan menjalankan:

```
aws iotevents describe-detector-model \  
  --detector-model-name your-detector-model-name
```

3. Simpan output untuk setiap model detektor.

## Langkah 2: Buat peran IAM

Buat peran IAM untuk memberikan izin untuk mereplikasi fungsionalitas. AWS IoT Events Peran dalam contoh ini memberikan akses ke DynamoDB untuk manajemen status, untuk penjadwalan EventBridge, Kinesis Data Streams untuk konsumsi data, untuk memublikasikan pesan, dan untuk pencatatan. AWS IoT Core CloudWatch Bersama-sama, layanan ini berfungsi sebagai pengganti AWS IoT Events.

1. Buat peran IAM dengan izin sebagai berikut. Untuk petunjuk selengkapnya tentang cara membuat peran IAM, lihat [Membuat peran untuk mendelegasikan izin ke AWS layanan di Panduan Pengguna IAM](#).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DynamoDBAccess",
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb>DeleteItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/EventsStateTable"
    },
    {
      "Sid": "SchedulerAccess",
      "Effect": "Allow",
      "Action": [
        "scheduler:CreateSchedule",
        "scheduler>DeleteSchedule"
      ],
      "Resource": "arn:aws:scheduler:us-east-1:123456789012:schedule/*"
    }
  ]
}
```

```

        "Sid": "KinesisAccess",
        "Effect": "Allow",
        "Action": [
            "kinesis:GetRecords",
            "kinesis:GetShardIterator",
            "kinesis:DescribeStream",
            "kinesis:ListStreams"
        ],
        "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/*"
    },
    {
        "Sid": "IoTPublishAccess",
        "Effect": "Allow",
        "Action": "iot:Publish",
        "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
    },
    {
        "Effect": "Allow",
        "Action": "logs:CreateLogGroup",
        "Resource": "arn:aws:logs:us-east-1:123456789012:*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogStream",
            "logs:PutLogEvents"
        ],
        "Resource": [
            "arn:aws:logs:us-east-1:123456789012:log-group:/aws/lambda/your-
Lambda:*"
        ]
    }
]
}

```

2. Tambahkan kebijakan kepercayaan peran IAM berikut. Kebijakan kepercayaan memungkinkan AWS layanan yang ditentukan untuk mengambil peran IAM sehingga mereka dapat melakukan tindakan yang diperlukan. Untuk petunjuk selengkapnya tentang cara membuat kebijakan kepercayaan IAM, lihat [Membuat peran menggunakan kebijakan kepercayaan khusus](#) di Panduan Pengguna IAM.

## JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "scheduler.amazonaws.com",
          "lambda.amazonaws.com",
          "iot.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

### Langkah 3: Buat Amazon Kinesis Data Streams

Buat Amazon Kinesis Data Streams Konsol Manajemen AWS menggunakan atau. AWS CLI

#### Console

Untuk membuat aliran data Kinesis menggunakan Konsol Manajemen AWS, ikuti prosedur yang ditemukan di halaman [Buat aliran data](#) di Panduan Pengembang Amazon Kinesis Data Streams.

Sesuaikan jumlah pecahan berdasarkan jumlah perangkat dan ukuran payload pesan.

#### AWS CLI

Menggunakan AWS CLI, buat Amazon Kinesis Data Streams untuk menyerap dan mempartisi data dari perangkat Anda.

Kinesis Data Streams digunakan dalam migrasi ini untuk menggantikan fungsionalitas konsumsi data. AWS IoT Events Ini menyediakan cara yang terukur dan efisien untuk mengumpulkan, memproses, dan menganalisis data streaming waktu nyata dari perangkat IoT Anda, sambil menyediakan penanganan dan integrasi data yang fleksibel dengan AWS layanan lain.

```
aws kinesis create-stream --stream-name your-kinesis-stream-name --shard-count 4 --  
region your-region
```

Sesuaikan jumlah pecahan berdasarkan jumlah perangkat dan ukuran payload pesan.

## Langkah 4: Buat atau perbarui aturan perutean pesan MQTT

Anda dapat membuat aturan perutean pesan MQTT baru atau memperbarui aturan yang ada.

### Console

1. Tentukan apakah Anda memerlukan aturan perutean pesan MQTT baru atau apakah Anda dapat memperbarui aturan yang ada.
2. Buka [konsol AWS IoT Core](#).
3. Di panel navigasi, pilih Perutean Pesan, lalu pilih Aturan.
4. Di bagian Kelola, pilih Perutean pesan, lalu Aturan.
5. Pilih Buat aturan.
6. Pada halaman Tentukan properti aturan, masukkan nama AWS IoT Core aturan untuk nama Aturan. Untuk Deskripsi Aturan - opsional, masukkan deskripsi untuk mengidentifikasi bahwa Anda sedang memproses peristiwa dan meneruskannya ke Kinesis Data Streams.
7. Pada halaman pernyataan Configure SQL, masukkan yang berikut untuk pernyataan SQL: **SELECT \* FROM 'your-database'**, lalu pilih Berikutnya.
8. Pada halaman Lampirkan tindakan aturan, dan di bawah Tindakan aturan, pilih kinesis.
9. Pilih aliran Kinesis Anda untuk aliran. Untuk kunci partisi, masukkan **your-instance-id**. Pilih peran yang sesuai untuk peran IAM, lalu pilih Tambahkan tindakan aturan.

Untuk informasi selengkapnya, lihat [Membuat aturan AWS IoT untuk merutekan data perangkat ke layanan lain](#).

### AWS CLI

1. Buat file JSON dengan konten berikut. File konfigurasi JSON ini mendefinisikan AWS IoT Core aturan yang memilih semua pesan dari topik dan meneruskannya ke aliran Kinesis tertentu, menggunakan ID instance sebagai kunci partisi.

```
{
  "sql": "SELECT * FROM 'your-config-file'",
  "description": "Rule to process events and forward to Kinesis Data Streams",
  "actions": [
    {
      "kinesis": {
        "streamName": "your-kinesis-stream-name",
        "roleArn": "arn:aws:iam::your-account-id:role/service-role/your-iam-role",
        "partitionKey": "${your-instance-id}"
      }
    }
  ],
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23"
}
```

2. Buat aturan topik MQTT menggunakan. AWS CLI Langkah ini menggunakan aturan AWS CLI untuk membuat AWS IoT Core topik menggunakan konfigurasi yang ditentukan dalam `events_rule.json` file.

```
aws iot create-topic-rule \
  --rule-name "your-iot-core-rule" \
  --topic-rule-payload file://your-file-name.json
```

## Langkah 5: Dapatkan titik akhir untuk topik MQTT tujuan

Gunakan topik MQTT tujuan untuk mengonfigurasi tempat topik Anda mempublikasikan pesan keluar, menggantikan fungsionalitas yang sebelumnya ditangani oleh. AWS IoT Events Titik akhir unik untuk AWS akun dan wilayah Anda.

### Console

1. Buka [konsol AWS IoT Core](#).
2. Di bagian Connect di panel navigasi kiri, pilih Konfigurasi domain.
3. Pilih konfigurasi Data-ATS domain iot: untuk membuka halaman detail konfigurasi.
4. Salin nilai nama Domain. Nilai ini adalah titik akhir. Simpan nilai endpoint karena Anda akan membutuhkannya di langkah selanjutnya.

## AWS CLI

Jalankan perintah berikut untuk mendapatkan AWS IoT Core titik akhir untuk menerbitkan pesan keluar untuk akun Anda.

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

## Langkah 6: Buat tabel Amazon DynamoDB

Tabel Amazon DynamoDB menggantikan fungsionalitas AWS IoT Events manajemen status, menyediakan cara yang skalabel dan fleksibel untuk mempertahankan dan mengelola status perangkat Anda dan logika model detektor dalam arsitektur solusi baru Anda.

### Console

Buat tabel Amazon DynamoDB untuk mempertahankan status model detektor. Untuk informasi selengkapnya, lihat [Membuat tabel di DynamoDB di](#) Panduan Pengembang Amazon DynamoDB.

Gunakan yang berikut ini untuk detail tabel:

- Untuk nama Tabel, masukkan nama tabel pilihan Anda.
- Untuk kunci Partition, masukkan ID instans Anda sendiri.
- Anda dapat menggunakan pengaturan Default untuk pengaturan Tabel

## AWS CLI

Jalankan perintah berikut untuk membuat tabel DynamoDB.

```
aws dynamodb create-table \  
    --table-name your-table-name \  
    --attribute-definitions AttributeName=your-instance-  
id,AttributeType=S \  
    --key-schema AttributeName=your-instance-id,KeyType=HASH \  
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=10
```

## Langkah 7: Buat AWS Lambda fungsi (konsol)

Fungsi Lambda berfungsi sebagai mesin pemrosesan inti, menggantikan logika evaluasi model detektor. AWS IoT Events Dalam contoh, kami mengintegrasikan dengan AWS layanan lain untuk

menangani data yang masuk, mengelola status, dan memicu tindakan berdasarkan aturan yang Anda tetapkan.

Buat fungsi Lambda dengan NodeJS runtime. Gunakan cuplikan kode berikut, ganti konstanta hard-code:

1. Buka [AWS Lambda console](#).
2. Pilih Buat fungsi.
3. Masukkan nama untuk nama Fungsi.
4. Pilih NodeJS 22.x sebagai Runtime.
5. Di dropdown Ubah peran eksekusi default, pilih Gunakan peran yang ada, lalu pilih peran IAM yang Anda buat di langkah sebelumnya.
6. Pilih Buat fungsi.
7. Tempel cuplikan kode berikut setelah mengganti konstanta kode keras.
8. Setelah fungsi Anda dibuat, di bawah tab Kode, tempel contoh kode berikut, ganti **your-destination-endpoint** titik akhir dengan milik Anda sendiri.

```
import { DynamoDBClient, GetItemCommand } from '@aws-sdk/client-dynamodb';
import { PutItemCommand } from '@aws-sdk/client-dynamodb';
import { IoTDataPlaneClient, PublishCommand } from "@aws-sdk/client-iot-data-plane";
import { SchedulerClient, CreateScheduleCommand, DeleteScheduleCommand } from "@aws-
sdk/client-scheduler"; // ES Modules import

///// External Clients and Constants
const scheduler = new SchedulerClient({});
const iot = new IoTDataPlaneClient({
  endpoint: 'https://your-destination-endpoint-ats.iot.your-region.amazonaws.com/'
});
const ddb = new DynamoDBClient({});

///// Lambda Handler function
export const handler = async (event) => {
  console.log('Incoming event:', JSON.stringify(event, null, 2));

  if (!event.Records) {
    throw new Error('No records found in event');
```

```
}

const processedRecords = [];

for (const record of event.Records) {
  try {
    if (record.eventSource !== 'aws:kinesis') {
      console.log(`Skipping non-Kinesis record from ${record.eventSource}`);
      continue;
    }

    // Assumes that we are processing records from Kinesis
    const payload = record.kinesis.data;
    const decodedData = Buffer.from(payload, 'base64').toString();
    console.log("decoded payload is ", decodedData);

    const output = await handleDecodedData(decodedData);

    // Add additional processing logic here
    const processedData = {
      output,
      sequenceNumber: record.kinesis.sequenceNumber,
      partitionKey: record.kinesis.partitionKey,
      timestamp: record.kinesis.approximateArrivalTimestamp
    };

    processedRecords.push(processedData);

  } catch (error) {
    console.error('Error processing record:', error);
    console.error('Failed record:', record);
    // Decide whether to throw error or continue processing other records
    // throw error; // Uncomment to stop processing on first error
  }
}

return {
  statusCode: 200,
  body: JSON.stringify({
    message: 'Processing complete',
    processedCount: processedRecords.length,
    records: processedRecords
  })
};
```

```
};

// Helper function to handle decoded data
async function handleDecodedData(payload) {
  try {
    // Parse the decoded data
    const parsedData = JSON.parse(payload);

    // Extract instanceId
    const instanceId = parsedData.instanceId;
    // Parse the input field
    const inputData = JSON.parse(parsedData.payload);
    const temperature = inputData.temperature;
    console.log('For InstanceId: ', instanceId, ' the temperature is:',
temperature);

    await iotEvents.process(instanceId, inputData)

    return {
      instanceId,
      temperature,
      // Add any other fields you want to return
      rawInput: inputData
    };
  } catch (error) {
    console.error('Error handling decoded data:', error);
    throw error;
  }
}

///// Classes for declaring/defining the state machine
class CurrentState {
  constructor(instanceId, stateName, variables, inputs) {
    this.stateName = stateName;
    this.variables = variables;
    this.inputs = inputs;
    this.instanceId = instanceId
  }

  static async load(instanceId) {
    console.log(`Loading state for id ${instanceId}`);
    try {
```

```
        const { Item: { state: { S: stateContent } } } = await ddb.send(new
getItemCommand({
    TableName: 'EventsStateTable',
    Key: {
        'InstanceId': { S: `${instanceId}` }
    }
}));

    const { stateName, variables, inputs } = JSON.parse(stateContent);

    return new CurrentState(instanceId, stateName, variables, inputs);
} catch (e) {
    console.log(`No state for id ${instanceId}: ${e}`);
    return undefined;
}
}

static async save(instanceId, state) {
    console.log(`Saving state for id ${instanceId}`);
    await ddb.send(new PutItemCommand({
        TableName: 'your-events-state-table-name',
        Item: {
            'InstanceId': { S: `${instanceId}` },
            'state': { S: state }
        }
    }));
}

setVariable(name, value) {
    this.variables[name] = value;
}

changeState(stateName) {
    console.log(`Changing state from ${this.stateName} to ${stateName}`);
    this.stateName = stateName;
}

async setTimer(instanceId, frequencyInMinutes, payload) {
    console.log(`Setting timer ${instanceId} for frequency of ${frequencyInMinutes}
minutes`);

    const base64Payload = Buffer.from(JSON.stringify(payload)).toString();
    console.log(base64Payload);
}
```

```
const scheduleName = `your-schedule-name-${instanceId}-schedule`;
const scheduleParams = {
  Name: scheduleName,
  FlexibleTimeWindow: {
    Mode: 'OFF'
  },
  ScheduleExpression: `rate(${frequencyInMinutes} minutes)`,
  Target: {
    Arn: "arn:aws::kinesis:your-region:your-account-id:stream/your-kinesis-
stream-name",
    RoleArn: "arn:aws::iam:your-account-id:role/service-role/your-iam-
role",
    Input: base64Payload,
    KinesisParameters: {
      PartitionKey: instanceId,
    },
    RetryPolicy: {
      MaximumRetryAttempts: 3
    }
  },
};

const command = new CreateScheduleCommand(scheduleParams);
console.log(`Sending command to set timer ${JSON.stringify(command)}`);
await scheduler.send(command);
}

async clearTimer(instanceId) {
  console.log(`Cleaning timer ${instanceId}`);

  const scheduleName = `your-schedule-name-${instanceId}-schedule`;
  const command = new DeleteScheduleCommand({
    Name: scheduleName
  });
  await scheduler.send(command);
}

async executeAction(actionType, actionPayload) {
  console.log(`Will execute the ${actionType} with payload ${actionPayload}`);
  await iot.send(new PublishCommand({
    topic: `${this.instanceId}`,
    payload: actionPayload,
    qos: 0
  }));
}
```

```
    }));
  }

  setInput(value) {
    this.inputs = { ...this.inputs, ...value };
  }

  input(name) {
    return this.inputs[name];
  }
}

class IoTEvents {

  constructor(initialState) {
    this.initialState = initialState;
    this.states = {};
  }

  state(name) {
    const state = new IoTEventsState();
    this.states[name] = state;
    return state;
  }

  async process(instanceId, input) {
    let currentState = await CurrentState.load(instanceId) || new
    CurrentState(instanceId, this.initialState, {}, {});
    currentState.setInput(input);

    console.log(`With inputs as: ${JSON.stringify(currentState)}`);
    const state = this.states[currentState.stateName];

    currentState = await state.evaluate(currentState);
    console.log(`With output as: ${JSON.stringify(currentState)}`);

    await CurrentState.save(instanceId, JSON.stringify(currentState));
  }
}

class Event {
  constructor(condition, action) {
    this.condition = condition;
  }
}
```

```
        this.action = action;
    }
}

class IoTEventsState {
    constructor() {
        this.eventsList = []
    }

    events(eventListArg) {
        this.eventsList.push(...eventListArg);
        return this;
    }

    async evaluate(currentState) {
        for (const e of this.eventsList) {
            console.log(`Evaluating event ${e.condition}`);
            if (e.condition(currentState)) {
                console.log(`Event condition met`);
                // Execute any action as defined in iotEvents DM Definition
                await e.action(currentState);
            }
        }

        return currentState;
    }
}

///// DetectorModel Definitions - replace with your own defintions
let processAlarmStateEvent = new Event(
    (currentState) => {
        const source = currentState.input('source');
        return (
            currentState.input('temperature') < 70
        );
    },
    async (currentState) => {
        currentState.changeState('normal');
        await currentState.clearTimer(currentState.instanceId)
        await currentState.executeAction('MQTT', `{"state": "alarm cleared, timer
deleted" }`);
    }
);
```

```
let processTimerEvent = new Event(
  (currentState) => {
    const source = currentState.input('source');
    console.log(`Evaluating timer event with source ${source}`);
    const booleanOutput = (source !== undefined && source !== null &&
      typeof source === 'string' &&
      source.toLowerCase() === 'timer' &&
      // check if the currentState == state from the timer payload
      currentState.input('currentState') !== undefined &&
      currentState.input('currentState') !== null &&
      currentState.input('currentState').toLowerCase !== 'normal');
    console.log(`Timer event evaluated as ${booleanOutput}`);
    return booleanOutput;
  },
  async (currentState) => {
    await currentState.executeAction('MQTT', `{"state": "timer timed out in Alarming state"}`);
  }
);

let processNormalEvent = new Event(
  (currentState) => currentState.input('temperature') > 70,
  async (currentState) => {
    currentState.changeState('alarm');
    await currentState.executeAction('MQTT', `{"state": "alarm detected, timer started"}`);
    await currentState.setTimer(currentState.instanceId, 5, {
      "instanceId": currentState.instanceId,
      "payload": `{"currentState\\": \\\"alarm\\", \\\"source\\": \\\"timer\\\"}"`
    });
  }
);

const iotEvents = new IoTEvents('normal');
iotEvents
  .state('normal')
  .events(
    [
      processNormalEvent
    ]
  );
iotEvents
  .state('alarm')
  .events([
    processAlarmStateEvent,
    processTimerEvent
  ]
);
```

```
]
);
```

## Langkah 8: Tambahkan pemicu Amazon Kinesis Data Streams

Tambahkan pemicu Kinesis Data Streams ke fungsi Lambda menggunakan atau. Konsol Manajemen AWS AWS CLI

Menambahkan pemicu Kinesis Data Streams ke fungsi Lambda Anda akan membuat koneksi antara pipeline konsumsi data dan logika pemrosesan Anda, memungkinkannya secara otomatis mengevaluasi aliran data IoT yang masuk dan bereaksi terhadap peristiwa secara real-time, mirip dengan cara memproses input. AWS IoT Events

### Console

Untuk informasi selengkapnya, lihat [Membuat pemetaan sumber peristiwa untuk menjalankan fungsi Lambda](#) di Panduan Pengembang.AWS Lambda

Gunakan yang berikut ini untuk detail pemetaan sumber acara:

- Untuk nama Fungsi, masukkan nama lambda yang digunakan di [Langkah 7: Buat AWS Lambda fungsi \(konsol\)](#).
- Untuk Konsumen - opsional, masukkan ARN untuk aliran Kinesis Anda.
- Untuk Ukuran batch, masukkan **10**.

### AWS CLI

Jalankan perintah berikut untuk membuat pemicu fungsi Lambda.

```
aws lambda create-event-source-mapping \  
  --function-name your-lambda-name \  
  --event-source arn:aws:kinesis:your-region:your-account-id:stream/your-kinesis-  
stream-name \  
  --batch-size 10 \  
  --starting-position LATEST \  
  --region your-region
```

## Langkah 9: Uji konsumsi data dan fungsionalitas keluaran (AWS CLI)

Publikasikan payload ke topik MQTT berdasarkan apa yang Anda tentukan dalam model detektor. Berikut ini adalah contoh payload ke topik MQTT `your-topic-name` untuk menguji implementasi.

```
{
  "instanceId": "your-instance-id",
  "payload": "{\"temperature\":78}"
}
```

Anda akan melihat pesan MQTT yang dipublikasikan ke topik dengan konten berikut (atau serupa):

```
{
  "state": "alarm detected, timer started"
}
```

## Prosedur migrasi untuk AWS IoT SiteWise alarm di AWS IoT Events

Bagian ini menjelaskan solusi alternatif yang memberikan fungsionalitas alarm serupa saat Anda bermigrasi. AWS IoT Events

Untuk AWS IoT SiteWise properti yang menggunakan AWS IoT Events alarm, Anda dapat bermigrasi ke solusi menggunakan CloudWatch alarm. Pendekatan ini memberikan kemampuan pemantauan yang kuat dengan SLA yang sudah mapan dan fitur tambahan seperti deteksi anomali dan alarm yang dikelompokkan.

### Membandingkan arsitektur

Konfigurasi AWS IoT Events alarm saat ini untuk AWS IoT SiteWise properti memerlukan pembuatan `AssetModelCompositeModels` dalam model aset, seperti yang dijelaskan dalam [Tentukan alarm eksternal AWS IoT SiteWise di Panduan AWS IoT SiteWise Pengguna](#). Modifikasi pada solusi baru biasanya dikelola melalui AWS IoT Events konsol.

Solusi baru ini menyediakan manajemen alarm dengan memanfaatkan CloudWatch alarm. Pendekatan ini menggunakan AWS IoT SiteWise notifikasi untuk mempublikasikan titik data properti ke topik AWS IoT Core MQTT, yang kemudian diproses oleh fungsi Lambda. Fungsi ini mengubah

notifikasi ini menjadi CloudWatch metrik, memungkinkan pemantauan alarm melalui kerangka kerja yang mengkhawatirkan CloudWatch.

Tujuan	Solusi	Perbedaan
Sumber data — Data properti dari AWS IoT SiteWise	AWS IoT SiteWise Pemberitahuan MQTT	Menggantikan integrasi IoT Events langsung dengan notifikasi MQTT dari properti AWS IoT SiteWise
Pemrosesan data - Mengubah data properti	Fungsi Lambda	Memproses pemberitahuan AWS IoT SiteWise properti dan mengubahnya menjadi metrik CloudWatch
Evaluasi alarm - Memantau metrik dan memicu alarm	CloudWatch Alarm Amazon	Mengganti AWS IoT Events alarm dengan CloudWatch alarm, menawarkan fitur tambahan seperti deteksi anomali
Integrasi - Koneksi dengan AWS IoT SiteWise	AWS IoT SiteWise alarm eksternal	Kemampuan opsional untuk mengimpor CloudWatch alarm kembali AWS IoT SiteWise sebagai alarm eksternal

## Langkah 1: Aktifkan pemberitahuan MQTT pada properti aset

Jika Anda menggunakan AWS IoT Events integrasi untuk AWS IoT SiteWise alarm, Anda dapat mengaktifkan notifikasi MQTT untuk setiap properti untuk dipantau.

- Ikuti [Konfigurasi alarm pada aset dalam AWS IoT SiteWise](#) prosedur hingga Anda setiap langkah untuk Mengedit properti model aset.
- Untuk setiap properti yang akan dimigrasi, ubah status Pemberitahuan MQTT menjadi AKTIF.

The screenshot shows the 'Properties' section of the AWS IoT SiteWise console. On the left, there is a sidebar with 'Property Type' and a list of categories: 'Attributes (1)', 'Measurements (2)', 'Transforms (0)', and 'Metrics (2)'. The 'Attributes (1)' category is selected. In the main area, under 'Attributes', there is a field for 'Alarm-Recipient' with a text input box and a note 'Must be less than 2048 characters.' To the right, there is a section titled 'MQTT Notification status' with a dropdown menu set to 'ACTIVE'. Below this, there is a text area showing the notification topic: 'Notification will be published to topic \$aws/sitewise/asset-models/[...]/assets/[...]/properties/[...]'.

- Perhatikan jalur topik yang dipublikasikan alarm untuk setiap atribut alarm yang dimodifikasi.

Untuk informasi selengkapnya, lihat sumber dokumentasi berikut:

- [Memahami properti aset dalam topik MQTT di Panduan Pengguna.AWS IoT SiteWise](#)
- [Topik MQTT di Panduan Pengembang.AWS IoT](#)

## Langkah 2: Buat AWS Lambda Fungsi

Buat fungsi Lambda untuk membaca array TQV yang diterbitkan oleh topik MQTT dan mempublikasikan nilai individual ke. CloudWatch Kita akan menggunakan fungsi Lambda ini sebagai tindakan tujuan untuk memicu dalam Aturan AWS IoT Core Pesan.

- Buka [AWS Lambda console](#).
- Pilih Buat fungsi.
- Masukkan nama untuk nama Fungsi.
- Pilih NodeJS 22.x sebagai Runtime.
- Di dropdown Ubah peran eksekusi default, pilih Gunakan peran yang ada, lalu pilih peran IAM yang Anda buat di langkah sebelumnya.

### Note

Prosedur ini mengasumsikan bahwa Anda telah memigrasikan model detektor Anda. Jika Anda tidak memiliki peran IAM, lihat [Langkah 2: Buat peran IAM](#).

- Pilih Buat fungsi.
- Tempel cuplikan kode berikut setelah mengganti konstanta kode keras.

```
import json
import boto3
from datetime import datetime

# Initialize CloudWatch client
cloudwatch = boto3.client('cloudwatch')

def lambda_handler(message, context):
    try:
        # Parse the incoming IoT message
```

```
# Extract relevant information
asset_id = message['payload']['assetId']
property_id = message['payload']['propertyId']

# Process each value in the values array
for value in message['payload']['values']:
    # Extract timestamp and value
    timestamp = datetime.fromtimestamp(value['timestamp']['timeInSeconds'])
    metric_value = value['value']['doubleValue']
    quality = value.get('quality', 'UNKNOWN')

# Publish to CloudWatch
response = cloudwatch.put_metric_data(
    Namespace='IoTSiteWise/AssetMetrics',
    MetricData=[
        {
            'MetricName': f'Property_{your-property-id}',
            'Value': metric_value,
            'Timestamp': timestamp,
            'Dimensions': [
                {
                    'Name': 'AssetId',
                    'Value': 'your-asset-id'
                },
                {
                    'Name': 'Quality',
                    'Value': quality
                }
            ]
        }
    ]
)

return {
    'statusCode': 200,
    'body': json.dumps('Successfully published metrics to CloudWatch')
}

except Exception as e:
    print(f'Error processing message: {str(e)}')
    return {
        'statusCode': 500,
        'body': json.dumps(f'Error: {str(e)}')
```

}

## Langkah 3: Buat AWS IoT Core aturan perutean pesan

- Ikuti [Tutorial: Menerbitkan ulang prosedur pesan MQTT](#) memasukkan informasi berikut saat diminta:
  - Aturan SiteWiseToCloudwatchAlarms perutean pesan nama.
  - Untuk kueri, Anda dapat menggunakan yang berikut ini:

```
SELECT * FROM '$aws/sitewise/asset-models/your-asset-model-id/assets/your-asset-id/properties/your-property-id'
```

- Dalam tindakan Aturan, pilih tindakan Lambda untuk mengirim data yang dihasilkan dari AWS IoT SiteWise ke. CloudWatch Contoh:

**Rule actions** Info  
Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. You can add up to 10 actions.

**Action 1**

▼ **Lambda** Info Remove  
Send a message to a Lambda function

**Lambda function** Info  
ListenForSiteWiseUpdates View Create a Lambda function

**Lambda function version**  
\$LATEST Refresh

Add rule action

## Langkah 4: Lihat CloudWatch metrik

Saat Anda menyerap data AWS IoT SiteWise, properti yang dipilih sebelumnya [Langkah 1: Aktifkan pemberitahuan MQTT pada properti aset](#), data rute ke fungsi Lambda yang kami buat. [Langkah 2: Buat AWS Lambda Fungsi](#) Pada langkah ini, Anda dapat memeriksa untuk melihat Lambda mengirimkan metrik Anda. CloudWatch

- Buka [CloudWatch Konsol Manajemen AWS](#).
- Di navigasi kiri, pilih Metrik, lalu Semua metrik.
- Pilih URL alarm untuk membukanya.
- Di bawah tab Sumber, CloudWatch output terlihat mirip dengan contoh ini. Informasi sumber ini menegaskan bahwa data metrik dimasukkan ke dalam CloudWatch.

```
{
  "view": "timeSeries",
  "stacked": false,
  "metrics": [
    [ "IoTSiteWise/AssetMetrics", "Property_your-property-id-hash", "Quality",
    "GOOD", "AssetId", "your-asset-id-hash", { "id": "m1" } ]
  ],
  "region": "your-region"
}
```

## Langkah 5: Buat CloudWatch alarm

Ikuti [Membuat CloudWatch alarm berdasarkan prosedur ambang statis](#) di Panduan CloudWatch Pengguna Amazon untuk membuat alarm untuk setiap metrik yang relevan.

### Note

Ada banyak opsi untuk konfigurasi alarm di Amazon CloudWatch Untuk informasi selengkapnya tentang CloudWatch alarm, lihat [Menggunakan CloudWatch alarm Amazon](#) di CloudWatch Panduan Pengguna Amazon.

## Langkah 6: (Opsional) impor CloudWatch alarm ke AWS IoT SiteWise

Anda dapat mengonfigurasi CloudWatch alarm untuk mengirim data kembali AWS IoT SiteWise menggunakan tindakan CloudWatch alarm dan Lambda. Integrasi ini memungkinkan Anda untuk melihat status alarm dan properti di portal SiteWise Monitor.

1. Konfigurasi alarm eksternal sebagai properti dalam model aset. Untuk informasi selengkapnya, lihat [Mendefinisikan alarm eksternal AWS IoT SiteWise di](#) Panduan AWS IoT SiteWise Pengguna.
2. Buat fungsi Lambda yang menggunakan [BatchPutAssetPropertyValue](#) API yang terdapat di Panduan AWS IoT SiteWise Pengguna untuk mengirim data alarm ke AWS IoT SiteWise
3. Siapkan tindakan CloudWatch alarm untuk menjalankan fungsi Lambda Anda saat status alarm berubah. Untuk informasi selengkapnya, lihat bagian [Tindakan alarm](#) di Panduan CloudWatch Pengguna Amazon.

# Menyiapkan AWS IoT Events

Bagian ini menyediakan panduan untuk menyiapkan AWS IoT Events, termasuk membuat AWS akun, mengonfigurasi izin yang diperlukan, dan menetapkan peran untuk mengelola akses ke sumber daya.

Topik

- [Menyiapkan sebuah Akun AWS](#)
- [Menyiapkan izin untuk AWS IoT Events](#)

## Menyiapkan sebuah Akun AWS

### Mendaftar untuk Akun AWS

Untuk memulai AWS, Anda membutuhkan Akun AWS. Untuk informasi tentang membuat Akun AWS, lihat [Memulai dengan Akun AWS](#) di Panduan AWS Account Management Referensi.

## Menyiapkan izin untuk AWS IoT Events

Menerapkan izin yang tepat penting untuk penggunaan yang aman dan efektif. AWS IoT Events Bagian ini menjelaskan izin yang diperlukan untuk menggunakan beberapa fitur. AWS IoT Events Anda dapat menggunakan AWS CLI perintah atau konsol AWS Identity and Access Management (IAM) untuk membuat peran dan kebijakan izin terkait untuk mengakses sumber daya atau menjalankan fungsi tertentu. AWS IoT Events

[Panduan Pengguna IAM](#) memiliki informasi lebih rinci tentang mengendalikan izin secara aman untuk mengakses sumber daya. AWS Untuk informasi khusus AWS IoT Events, lihat [Tindakan, sumber daya, dan kunci kondisi untuk AWS IoT Events](#).

Untuk menggunakan konsol IAM untuk membuat dan mengelola peran dan izin, lihat [Tutorial IAM: Mendelegasikan akses di seluruh AWS akun](#) menggunakan peran IAM.

#### Note

Tombol dapat 1-128 karakter dan dapat mencakup:

- huruf besar atau kecil a-z
- angka 0-9
- karakter khusus -, \_, atau:.

## Izin tindakan untuk AWS IoT Events

AWS IoT Events memungkinkan Anda untuk memicu tindakan yang menggunakan AWS layanan lain. Untuk melakukannya, Anda harus memberikan AWS IoT Events izin untuk melakukan tindakan ini atas nama Anda. Bagian ini berisi daftar tindakan dan contoh kebijakan yang memberikan izin untuk melakukan semua tindakan ini pada sumber daya Anda. Ubah *region* dan *account-id* referensi sesuai kebutuhan. Jika memungkinkan, Anda juga harus mengubah wildcard (\*) untuk merujuk ke sumber daya tertentu yang akan diakses. Anda dapat menggunakan konsol IAM untuk memberikan izin AWS IoT Events untuk mengirim peringatan Amazon SNS yang telah Anda tetapkan.

AWS IoT Events mendukung tindakan berikut yang memungkinkan Anda menggunakan timer atau mengatur variabel:

- [setTimer](#) untuk membuat timer.
- [resetTimer](#) untuk mengatur ulang timer.
- [clearTimer](#) untuk menghapus timer.
- [setVariable](#) untuk membuat variabel.

AWS IoT Events mendukung tindakan berikut yang memungkinkan Anda bekerja dengan AWS layanan:

- [iotTopicPublish](#) untuk mempublikasikan pesan tentang topik MQTT.
- [iotEvents](#) untuk mengirim data ke AWS IoT Events sebagai nilai input.
- [iotSiteWise](#) untuk mengirim data ke properti aset di AWS IoT SiteWise.
- [dynamoDB](#) untuk mengirim data ke tabel Amazon DynamoDB.
- [dynamoDBv2](#) untuk mengirim data ke tabel Amazon DynamoDB.
- [firehose](#) untuk mengirim data ke aliran Amazon Data Firehose.
- [lambda](#) untuk memanggil AWS Lambda fungsi.

- [sns](#) untuk mengirim data sebagai pemberitahuan push.
- [sqs](#) untuk mengirim data ke antrian Amazon SQS.

## Example Kebijakan

### JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
    },
    {
      "Effect": "Allow",
      "Action": "iotevents:BatchPutMessage",
      "Resource": "arn:aws:iotevents:us-east-1:123456789012:input/*"
    },
    {
      "Effect": "Allow",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "dynamodb:PutItem",
      "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ],
      "Resource": "arn:aws:firehose:us-east-1:123456789012:deliverystream/*"
    },
    {
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
```

```

    "Resource": "arn:aws:lambda:us-east-1:123456789012:function:*"
  },
  {
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-1:123456789012:*"
  },
  {
    "Effect": "Allow",
    "Action": "sqs:SendMessage",
    "Resource": "arn:aws:sqs:us-east-1:123456789012:*"
  }
]
}

```

## Mengamankan data input di AWS IoT Events

Penting untuk mempertimbangkan siapa yang dapat memberikan akses ke data input untuk digunakan dalam model detektor. Jika Anda memiliki pengguna atau entitas yang izin keseluruhannya ingin Anda batasi, tetapi diizinkan untuk membuat atau memperbarui model detektor, Anda juga harus memberikan izin kepada pengguna atau entitas tersebut untuk memperbarui perutean input. Ini berarti bahwa selain memberikan izin untuk `iotevents:CreateDetectorModel` dan `iotevents:UpdateDetectorModel`, Anda juga harus memberikan izin untuk `iotevents:UpdateInputRouting`.

### Example

Kebijakan berikut menambahkan izin untuk `iotevents:UpdateInputRouting`.

### JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "updateRoutingPolicy",
      "Effect": "Allow",
      "Action": [
        "iotevents:UpdateInputRouting"
      ]
    }
  ]
}

```

```

    "Resource": "*"
  }
]
}

```

Anda dapat menentukan daftar masukan Nama Sumber Daya Amazon (ARN) alih-alih wildcard "" untuk \* "Resource" untuk membatasi izin ini ke input tertentu. Ini memungkinkan Anda untuk membatasi akses ke data input yang dikonsumsi oleh model detektor yang dibuat atau diperbarui oleh pengguna atau entitas.

## Kebijakan peran CloudWatch pencatatan Amazon untuk AWS IoT Events

Dokumen kebijakan berikut menyediakan kebijakan peran dan kebijakan kepercayaan yang memungkinkan AWS IoT Events untuk mengirimkan log atas nama Anda. CloudWatch

Kebijakan peran:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:PutMetricFilter",
        "logs:PutRetentionPolicy",
        "logs:GetLogEvents",
        "logs>DeleteLogStream"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ]
    }
  ]
}

```

## Kebijakan kepercayaan:

### JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Anda juga memerlukan kebijakan izin IAM yang dilampirkan ke pengguna yang memungkinkan pengguna untuk meneruskan peran, sebagai berikut. Untuk informasi selengkapnya, lihat [Memberikan izin pengguna untuk meneruskan peran ke AWS layanan](#) di Panduan Pengguna IAM.

### JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/Role_To_Pass"
    }
  ]
}
```

Anda dapat menggunakan perintah berikut untuk menempatkan kebijakan sumber daya untuk CloudWatch log. Hal ini memungkinkan AWS IoT Events untuk menempatkan peristiwa log ke dalam CloudWatch aliran.

```
aws logs put-resource-policy --policy-name ioteventsLoggingPolicy --policy-document "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Sid\": \"IoTEventsToCloudWatchLogs\", \"Effect\": \"Allow\", \"Principal\": { \"Service\": [ \"iotevents.amazonaws.com\" ] }, \"Action\": \"logs:PutLogEvents\", \"Resource\": \"*\" } ] }"
```

Gunakan perintah berikut untuk menempatkan opsi logging. Ganti `roleArn` dengan peran logging yang Anda buat.

```
aws iotevents put-logging-options --cli-input-json "{ \"loggingOptions\": { \"roleArn\": \"arn:aws:iam::123456789012:role/testLoggingRole\", \"level\": \"INFO\", \"enabled\": true } }"
```

## Kebijakan peran pesan Amazon SNS untuk AWS IoT Events

Mengintegrasikan AWS IoT Events dengan Amazon SNS memerlukan manajemen izin yang cermat untuk pengiriman notifikasi yang aman dan efisien. Panduan ini memandu Anda melalui proses mengonfigurasi peran dan kebijakan IAM untuk memungkinkan mempublikasikan pesan AWS IoT Events ke topik Amazon SNS.

Dokumen kebijakan berikut menyediakan kebijakan peran dan kebijakan kepercayaan yang memungkinkan AWS IoT Events untuk mengirim pesan SNS.

Kebijakan peran:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "sns:*"
      ],

```

```
        "Effect": "Allow",
        "Resource": "arn:aws:sns:us-east-1:123456789012:testAction"
    }
]
}
```

Kebijakan kepercayaan:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

# Memulai dengan AWS IoT Events konsol

Bagian ini menunjukkan cara membuat input dan model detektor menggunakan [AWS IoT Events konsol](#). Anda memodelkan dua keadaan mesin: keadaan normal dan kondisi tekanan berlebih. Ketika tekanan yang diukur dalam mesin melebihi ambang batas tertentu, model bertransisi dari keadaan normal ke keadaan tekanan berlebih. Kemudian mengirimkan pesan Amazon SNS untuk memberi tahu teknisi tentang kondisinya. Ketika tekanan kembali turun di bawah ambang batas untuk tiga pembacaan tekanan berturut-turut, model kembali ke keadaan normal dan mengirimkan pesan Amazon SNS lainnya sebagai konfirmasi.

Kami memeriksa tiga pembacaan berturut-turut di bawah ambang tekanan untuk menghilangkan kemungkinan gagap tekanan berlebih atau pesan normal, dalam kasus fase pemulihan nonlinier atau pembacaan tekanan anomali.

Di konsol, Anda juga dapat menemukan beberapa templat model detektor yang sudah jadi yang dapat Anda sesuaikan. Anda juga dapat menggunakan konsol untuk mengimpor model detektor yang telah ditulis orang lain atau mengekspor model detektor Anda dan menggunakannya di AWS Wilayah yang berbeda. Jika Anda mengimpor model detektor, pastikan Anda membuat input yang diperlukan atau membuatnya ulang untuk Wilayah baru, dan memperbarui peran ARNs apa pun yang digunakan.

Gunakan AWS IoT Events konsol untuk mempelajari hal-hal berikut.

## Tentukan input

Untuk memantau perangkat dan proses Anda, mereka harus memiliki cara untuk mendapatkan data telemetri AWS IoT Events. Ini dilakukan dengan mengirim pesan sebagai input ke AWS IoT Events. Anda dapat melakukannya dengan dua cara:

- Gunakan [BatchPutMessage](#) operasi.
- Di AWS IoT Core, tulis aturan [AWS IoT Events tindakan](#) untuk mesin AWS IoT aturan yang meneruskan data pesan Anda. AWS IoT Events Anda harus mengidentifikasi input dengan nama.
- Di AWS IoT Analytics, gunakan [CreateDataset](#) operasi untuk membuat kumpulan data dengan `contentDeliveryRules`. Aturan-aturan ini menentukan AWS IoT Events input di mana konten kumpulan data dikirim secara otomatis.

Sebelum perangkat Anda dapat mengirim data dengan cara ini, Anda harus menentukan satu atau lebih input. Untuk melakukannya, berikan setiap input nama dan tentukan bidang mana dalam data pesan masuk yang dimonitor input.

## Buat model detektor

Tentukan model detektor (model peralatan atau proses Anda) menggunakan status. Untuk setiap status, tentukan logika kondisional (Boolean) yang mengevaluasi input yang masuk untuk mendeteksi peristiwa penting. Ketika model detektor mendeteksi suatu peristiwa, ia dapat mengubah status atau memulai tindakan yang dibuat khusus atau yang telah ditentukan sebelumnya menggunakan layanan lain. AWS Anda dapat menentukan peristiwa tambahan yang memulai tindakan saat memasuki atau keluar dari status dan, secara opsional, ketika suatu kondisi terpenuhi.

Dalam tutorial ini, Anda mengirim pesan Amazon SNS sebagai tindakan ketika model memasuki atau keluar dari status tertentu.

## Memantau perangkat atau proses

Jika Anda memantau beberapa perangkat atau proses, tentukan bidang di setiap input yang mengidentifikasi perangkat atau proses tertentu dari mana input berasal. Lihat key bidang `diCreateDetectorModel`. Ketika bidang input diidentifikasi oleh key mengenali nilai baru, perangkat baru diidentifikasi dan detektor dibuat. Setiap detektor adalah contoh dari model detektor. Detektor baru terus merespons input yang berasal dari perangkat itu hingga model detektornya diperbarui atau dihapus.

Jika Anda memantau satu proses (meskipun beberapa perangkat atau subproses mengirimkan input), Anda tidak menentukan bidang identifikasi key unik. Dalam hal ini, model membuat detektor tunggal (instance) ketika input pertama tiba.

## Kirim pesan sebagai input ke model detektor Anda

Ada beberapa cara untuk mengirim pesan dari perangkat atau memproses sebagai input ke AWS IoT Events detektor yang tidak mengharuskan Anda melakukan pemformatan tambahan pada pesan. Dalam tutorial ini, Anda menggunakan AWS IoT konsol untuk menulis aturan [AWS IoT Events tindakan](#) untuk mesin AWS IoT aturan yang meneruskan data pesan Anda. AWS IoT Events

Untuk melakukan ini, identifikasi input berdasarkan nama dan terus gunakan AWS IoT konsol untuk menghasilkan pesan yang diteruskan sebagai input ke. AWS IoT Events

**Note**

Tutorial ini menggunakan konsol untuk membuat yang sama input dan detector model ditunjukkan pada contoh di [Tutorial untuk kasus AWS IoT Events penggunaan](#). Anda dapat menggunakan contoh JSON ini untuk membantu Anda mengikuti tutorial.

## Topik

- [Prasyarat untuk memulai AWS IoT Events](#)
- [Buat masukan untuk model di AWS IoT Events](#)
- [Buat model detektor di AWS IoT Events](#)
- [Kirim input untuk menguji model detektor di AWS IoT Events](#)

## Prasyarat untuk memulai AWS IoT Events

Jika Anda tidak memiliki AWS akun, buat satu.

1. Ikuti langkah [Menyiapkan AWS IoT Events](#) untuk memastikan pengaturan dan izin akun yang tepat.
2. Buat dua topik Amazon Simple Notification Service (Amazon SNS).

Tutorial ini (dan contoh yang sesuai) berasumsi bahwa Anda membuat dua topik Amazon SNS. Topik-topik ini ditampilkan sebagai: `arn:aws:sns:us-east-1:123456789012:underPressureAction` dan `arn:aws:sns:us-east-1:123456789012:pressureClearedAction`. ARNs Ganti nilai ini dengan ARNs topik Amazon SNS yang Anda buat. Untuk informasi lebih lanjut, lihat [Panduan Developer Layanan Notifikasi Sederhana Amazon](#).

Sebagai alternatif untuk menerbitkan peringatan ke topik Amazon SNS, Anda dapat meminta detektor mengirim pesan MQTT dengan topik yang Anda tentukan. Dengan opsi ini, Anda dapat memverifikasi bahwa model detektor Anda membuat instance dan instans tersebut mengirimkan peringatan dengan menggunakan konsol AWS IoT Core untuk berlangganan dan memantau pesan yang dikirim ke topik MQTT tersebut. Anda juga dapat menentukan nama topik MQTT secara dinamis saat runtime dengan menggunakan input atau variabel yang dibuat dalam model detektor.

3. Pilih Wilayah AWS yang mendukung AWS IoT Events. Untuk informasi selengkapnya, lihat [AWS IoT Events](#) di Referensi Umum AWS. Untuk bantuan, lihat [Memulai dengan layanan Konsol Manajemen AWS di Memulai dengan Konsol Manajemen AWS](#).

## Buat masukan untuk model di AWS IoT Events

Saat Anda membuat input untuk model Anda, sebaiknya kumpulkan file yang berisi contoh muatan pesan yang dikirim perangkat atau proses Anda untuk melaporkan status kesehatannya. Memiliki file-file ini membantu Anda menentukan input yang diperlukan.

Anda dapat membuat input melalui beberapa metode yang dijelaskan di bagian ini.

### Buat file masukan JSON

1. Untuk memulai, buat file bernama `input.json` pada sistem file lokal Anda dengan konten berikut:

```
{
  "motorid": "Fulton-A32",
  "sensorData": {
    "pressure": 23,
    "temperature": 47
  }
}
```

2. Sekarang Anda memiliki `input.json` file starter ini, Anda dapat membuat input. Ada dua cara untuk membuat input. Anda dapat membuat input dengan menggunakan panel navigasi di [AWS IoT Events konsol](#). Atau, Anda dapat membuat input dalam model detektor setelah dibuat.

### Buat dan konfigurasi masukan

Pelajari cara membuat input, untuk model alarm atau model detektor.

1. Masuk ke [AWS IoT Events konsol](#) atau pilih opsi untuk Buat AWS IoT Events akun baru.
2. Di AWS IoT Events konsol, di sudut kiri atas, pilih dan perluas panel navigasi.
3. Di panel navigasi kiri, pilih Input.
4. Di sudut kanan konsol, pilih Buat input.

5. Berikan yang unik InputName.
6. Opsional — masukkan Deskripsi untuk masukan Anda.
7. Untuk Mengunggah file JSON, pilih `input.json` file yang Anda buat dalam ikhtisar. [Buat file masukan JSON](#) Pilih atribut input muncul dengan daftar atribut yang Anda masukkan.
8. Untuk Pilih atribut input, pilih atribut yang akan digunakan, dan pilih Buat. Dalam contoh ini, kita memilih `motorid` dan `sensordata.pressure`.
9. Opsional - Tambahkan Tag yang relevan ke input.

#### Note

Anda juga dapat membuat input tambahan dalam model detektor di [AWS IoT Events konsol](#). Untuk informasi selengkapnya, lihat [Buat masukan dalam Model Detektor di AWS IoT Events](#).

## Buat masukan dalam Model Detektor di AWS IoT Events

Input detektor AWS IoT Events berfungsi sebagai jembatan antara sumber data Anda dan model detektor. Input detektor menyediakan data mentah yang mendukung deteksi peristiwa dan kemampuan otomatisasi. AWS IoT Events Pelajari cara mengonfigurasi input detektor untuk membantu model Anda merespons secara akurat peristiwa dan kondisi dunia nyata di ekosistem IoT Anda.

Bagian ini menunjukkan cara menentukan input untuk model detektor untuk menerima data telemetri, atau pesan.

Untuk menentukan input untuk model detektor

1. Buka [konsol AWS IoT Events](#).
2. Di AWS IoT Events konsol, pilih Buat model detektor.
3. Pilih Buat baru.
4. Pilih Buat masukan.
5. Untuk masukan, masukkan InputName, Deskripsi opsional, dan pilih Unggah file. Di kotak dialog yang ditampilkan, pilih `input.json` file yang Anda buat dalam ikhtisar [Buat file masukan JSON](#).
6. Untuk Pilih atribut input, pilih atribut yang akan digunakan, dan pilih Buat. Dalam contoh ini, kami memilih `MotorID` dan `SensorData.pressure`.

## Buat model detektor di AWS IoT Events

Dalam topik ini, Anda mendefinisikan model detektor (model peralatan atau proses Anda) menggunakan status.

Untuk setiap status, Anda mendefinisikan logika kondisional (Boolean) yang mengevaluasi input yang masuk untuk mendeteksi peristiwa penting. Ketika suatu peristiwa terdeteksi, itu mengubah status dan dapat memulai tindakan tambahan. Peristiwa ini dikenal sebagai peristiwa transisi.

Di status Anda, Anda juga menentukan peristiwa yang dapat menjalankan tindakan setiap kali detektor masuk atau keluar dari status tersebut atau ketika input diterima (ini dikenal sebagai `OnEnter`, `OnExit` dan `OnInput` peristiwa). Tindakan berjalan hanya jika logika kondisional acara mengevaluasi `true`.

Untuk membuat model detektor

1. Status detektor pertama telah dibuat untuk Anda. Untuk memodifikasinya, pilih lingkaran dengan label `State_1` di ruang pengeditan utama.
2. Di panel Negara, masukkan nama Negara dan `OnEnter`, pilih Tambah acara.
3. Pada halaman Tambah `OnEnter` acara, masukkan nama Acara dan kondisi Acara. Dalam contoh ini, enter `true` untuk menunjukkan acara selalu dimulai ketika status dimasukkan.
4. Di bawah Tindakan acara, pilih Tambah tindakan.
5. Di bawah tindakan Acara, lakukan hal berikut:
  - a. Pilih Tetapkan variabel
  - b. Untuk operasi Variabel, pilih Tetapkan nilai.
  - c. Untuk nama Variabel, masukkan nama variabel yang akan ditetapkan.
  - d. Untuk nilai Variabel, masukkan nilai `0` (nol).
6. Pilih Simpan.

Variabel, seperti yang Anda tentukan, dapat diatur (diberi nilai) dalam peristiwa apa pun dalam model detektor. Nilai variabel hanya dapat direferensikan (misalnya, dalam logika kondisional suatu peristiwa) setelah detektor mencapai status dan menjalankan tindakan di mana ia didefinisikan atau ditetapkan.

7. Di panel State, pilih X di sebelah State untuk kembali ke palet model Detector.
8. Untuk membuat status detektor kedua, dalam palet model Detektor, pilih Status dan seret ke ruang pengeditan utama. Ini menciptakan negara berjudul `untitled_state_1`.

9. Jeda pada keadaan pertama (Normal). Panah muncul di lingkaran negara.
10. Klik dan seret panah dari status pertama ke status kedua. Garis terarah dari keadaan pertama ke keadaan kedua (berlabel Untitled) muncul.
11. Pilih baris Untitled. Di panel peristiwa Transisi, masukkan nama Acara dan logika pemicu peristiwa.
12. Di panel acara Transisi, pilih Tambah tindakan.
13. Pada panel Tambahkan tindakan peristiwa transisi, pilih Tambah tindakan.
14. Untuk Pilih tindakan, pilih Tetapkan variabel.
  - a. Untuk operasi Variabel, pilih Tetapkan nilai.
  - b. Untuk nama Variabel, masukkan nama variabel.
  - c. Untuk Menetapkan nilai, masukkan nilai seperti:  
`$variable.pressureThresholdBreached + 3`
  - d. Pilih Simpan.
15. Pilih status kedua `untitled_state_1`.
16. Di panel Negara, masukkan nama Negara dan untuk On Enter, pilih Tambah acara.
17. Pada halaman Tambah OnEnter acara, masukkan nama Acara dan kondisi Acara. Pilih Tambahkan tindakan.
18. Untuk Pilih tindakan, pilih Kirim pesan SNS.
  - a. Untuk topik SNS, masukkan ARN target topik Amazon SNS Anda.
  - b. Pilih Simpan.
19. Lanjutkan untuk menambahkan acara dalam contoh.
  - a. Untuk OnInput, pilih Tambahkan acara, lalu masukkan dan simpan informasi acara berikut.

```
Event name: Overpressurized
Event condition: $input.PressureInput.sensorData.pressure > 70
Event actions:
  Set variable:
    Variable operation: Assign value
    Variable name: pressureThresholdBreached
    Assign value: 3
```

- b. Untuk OnInput, pilih Tambahkan acara, lalu masukkan dan simpan informasi acara berikut.

```
Event name: Pressure Okay
Event condition: $input.PressureInput.sensorData.pressure <= 70
Event actions:
  Set variable:
    Variable operation: Decrement
    Variable name: pressureThresholdBreached
```

- c. Untuk OnExit, pilih Tambahkan acara, lalu masukkan dan simpan informasi acara berikut menggunakan ARN dari topik Amazon SNS yang Anda buat.

```
Event name: Normal Pressure Restored
Event condition: true
Event actions:
  Send SNS message:
    Target arn: arn:aws:sns:us-east-1:123456789012:pressureClearedAction
```

20. Jeda pada keadaan kedua (Berbahaya). Panah muncul di lingkaran negara
21. Klik dan seret panah dari status kedua ke status pertama. Baris terarah dengan label Untitled muncul.
22. Pilih baris Tanpa Judul dan di panel peristiwa Transisi, masukkan nama Acara dan logika pemicu peristiwa menggunakan informasi berikut.

```
{
  Event name: BackToNormal
  Event trigger logic: $input.PressureInput.sensorData.pressure <= 70 &&
  $variable.pressureThresholdBreached <= 0
}
```

Untuk informasi lebih lanjut tentang mengapa kami menguji `$input` nilai dan `$variable` nilai dalam logika pemicu, lihat entri untuk ketersediaan nilai variabel di [AWS IoT Events pembatasan dan batasan model detektor](#).

23. Pilih status Mulai. Secara default, status ini dibuat saat Anda membuat model detektor). Di panel Mulai, pilih status Tujuan (misalnya, Normal).
24. Selanjutnya, konfigurasi model detektor Anda untuk mendengarkan input. Di sudut kanan atas, pilih Publish.
25. Pada halaman model Publish detector, lakukan hal berikut.

- a. Masukkan nama model Detektor, Deskripsi, dan nama Peran. Peran ini dibuat untuk Anda.
  - b. Pilih Buat detektor untuk setiap nilai kunci unik. Untuk membuat dan menggunakan Peran Anda sendiri, ikuti langkah-langkahnya [Menyiapkan izin untuk AWS IoT Events](#) dan masukkan sebagai Peran di sini.
26. Untuk kunci pembuatan Detektor, pilih nama salah satu atribut input yang Anda tentukan sebelumnya. Atribut yang Anda pilih sebagai kunci pembuatan detektor harus ada di setiap input pesan, dan harus unik untuk setiap perangkat yang mengirim pesan. Contoh ini menggunakan atribut motorid.
27. Pilih Simpan dan terbitkan.

### Note

Jumlah detektor unik yang dibuat untuk model detektor tertentu didasarkan pada pesan input yang dikirim. Ketika model detektor dibuat, kunci dipilih dari atribut input. Kunci ini menentukan instance detektor mana yang akan digunakan. Jika kunci belum pernah terlihat sebelumnya (untuk model detektor ini), instance detektor baru dibuat. Jika kunci telah terlihat sebelumnya, kami menggunakan instance detektor yang ada sesuai dengan nilai kunci ini.

Anda dapat membuat salinan cadangan definisi model detektor Anda (dalam JSON) membuat ulang atau memperbaiki model detektor atau digunakan sebagai templat untuk membuat model detektor lain.

Anda dapat melakukan ini dari konsol atau dengan menggunakan perintah CLI berikut. Jika perlu, ubah nama model detektor agar sesuai dengan yang Anda gunakan saat Anda menerbitkannya di langkah sebelumnya.

```
aws iotevents describe-detector-model --detector-model-name motorDetectorModel >
motorDetectorModel.json
```

Ini membuat file (`motorDetectorModel.json`) yang memiliki konten yang mirip dengan berikut ini.

```
{
  "detectorModel": {
    "detectorModelConfiguration": {
      "status": "ACTIVE",
      "lastUpdateTime": 1552072424.212,
```

```

    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "creationTime": 1552072424.212,
    "detectorModelArn": "arn:aws:iotevents:us-
west-2:123456789012:detectorModel/motorDetectorModel",
    "key": "motorid",
    "detectorModelName": "motorDetectorModel",
    "detectorModelVersion": "1"
  },
  "detectorModelDefinition": {
    "states": [
      {
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "Overpressurized",
              "actions": [
                {
                  "setVariable": {
                    "variableName":
"pressureThresholdBreached",
                    "value":
"$variable.pressureThresholdBreached + 3"
                  }
                }
              ],
              "condition": "$input.PressureInput.sensorData.pressure
> 70",
              "nextState": "Dangerous"
            }
          ],
          "events": []
        },
        "stateName": "Normal",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "actions": [
                {
                  "setVariable": {
                    "variableName":
"pressureThresholdBreached",
                    "value": "0"
                  }
                }
              ]
            }
          ]
        }
      }
    ]
  }
}

```

```

        }
      ],
      "condition": "true"
    }
  ]
},
"onExit": {
  "events": []
}
},
{
  "onInput": {
    "transitionEvents": [
      {
        "eventName": "Back to Normal",
        "actions": [],
        "condition": "$variable.pressureThresholdBreached <= 1
&& $input.PressureInput.sensorData.pressure <= 70",
        "nextState": "Normal"
      }
    ],
    "events": [
      {
        "eventName": "Overpressurized",
        "actions": [
          {
            "setVariable": {
              "variableName":
"pressureThresholdBreached",
              "value": "3"
            }
          }
        ],
        "condition": "$input.PressureInput.sensorData.pressure
> 70"
      }
    ],
    {
      "eventName": "Pressure Okay",
      "actions": [
        {
          "setVariable": {
            "variableName":
"pressureThresholdBreached",

```

```

        "value":
"$variable.pressureThresholdBreached - 1"
        }
    }
    ],
    "condition": "$input.PressureInput.sensorData.pressure
<= 70"
    }
]
},
"stateName": "Dangerous",
"onEnter": {
    "events": [
        {
            "eventName": "Pressure Threshold Breached",
            "actions": [
                {
                    "sns": {
                        "targetArn": "arn:aws:sns:us-
west-2:123456789012:MyIoTButtonSNSTopic"
                    }
                }
            ],
            "condition": "$variable.pressureThresholdBreached > 1"
        }
    ]
},
"onExit": {
    "events": [
        {
            "eventName": "Normal Pressure Restored",
            "actions": [
                {
                    "sns": {
                        "targetArn": "arn:aws:sns:us-
west-2:123456789012:IoTVirtualButtonTopic"
                    }
                }
            ],
            "condition": "true"
        }
    ]
}
}
}

```

```
    ],
    "initialStateName": "Normal"
  }
}
```

## Kirim input untuk menguji model detektor di AWS IoT Events

Ada beberapa cara untuk menerima data telemetri di AWS IoT Events (lihat [Tindakan yang didukung untuk menerima data dan memicu tindakan di AWS IoT Events](#)). Topik ini menunjukkan cara membuat AWS IoT aturan di AWS IoT konsol yang meneruskan pesan sebagai input ke detektor Anda AWS IoT Events . Anda dapat menggunakan klien MQTT AWS IoT konsol untuk mengirim pesan pengujian. Anda dapat menggunakan metode ini untuk memasukkan data telemetri AWS IoT Events saat perangkat Anda dapat mengirim pesan MQTT menggunakan broker pesan. AWS IoT

Untuk mengirim input untuk menguji model detektor

1. Buka [konsol AWS IoT Core](#). Di panel navigasi kiri, di bawah Kelola, pilih Perutean pesan, lalu pilih Aturan.
2. Pilih Buat aturan di kanan atas.
3. Pada halaman Buat aturan, selesaikan langkah-langkah berikut:

1. Langkah 1. Tentukan properti aturan. Lengkapi bidang-bidang berikut:

- Nama aturan. Masukkan nama untuk aturan Anda, seperti `MyIoTEventsRule`.

### Note

Jangan gunakan spasi.

- Deskripsi aturan. Ini opsional.
- Pilih Berikutnya.

2. Langkah 2. Konfigurasi pernyataan SQL. Lengkapi bidang-bidang berikut:

- Versi SQL. Pilih opsi yang sesuai dari daftar.
- Pernyataan SQL. Masukkan **`SELECT *, topic(2) as motorid FROM 'motors/+/' status'`**.

Pilih Berikutnya.

### 3. Langkah 3. Lampirkan tindakan aturan. Di bagian Tindakan aturan, lengkapi yang berikut ini:

- Tindakan 1. Pilih IoT Events. Bidang berikut muncul:
  - a. Nama masukan. Pilih opsi yang sesuai dari daftar. Jika input Anda tidak muncul, pilih Refresh.

Untuk membuat input baru, pilih input Create IoT Events. Lengkapi bidang-bidang berikut:

- Nama masukan. Masukkan `PressureInput`.
- Deskripsi. Ini opsional.
- Unggah file JSON. Unggah salinan file JSON Anda. Ada tautan ke file sampel di layar ini, jika Anda tidak memiliki file. Kode tersebut meliputi:

```
{
  "motorid": "Fulton-A32",
  "sensorData": {
    "pressure": 23,
    "temperature": 47
  }
}
```

- Pilih atribut input. Pilih opsi yang sesuai.
- Tanda. Ini opsional.

Pilih Buat.

Kembali ke Buat aturan layar dan segarkan bidang Nama input. Pilih input yang baru saja Anda buat.

- b. Mode Batch. Ini opsional. Jika payload adalah array pesan, pilih opsi ini.
- c. ID pesan. Ini memang opsional, tetapi direkomendasikan.
- d. Peran IAM. Pilih peran yang sesuai dari daftar. Jika peran tidak terdaftar, pilih Buat peran baru.

Ketik nama Peran dan pilih Buat.

Untuk menambahkan aturan lain, pilih Add rule action

- Tindakan kesalahan. Bagian ini opsional. Untuk menambahkan tindakan, pilih Tambahkan tindakan kesalahan dan pilih tindakan yang sesuai dari daftar.

Lengkapi bidang yang muncul.

- Pilih Berikutnya.


4. Langkah 4. Tinjau dan buat. Tinjau informasi di layar dan pilih Buat.

4. Di panel navigasi kiri, di bawah Uji, pilih klien pengujian MQTT.

5. Pilih Publikasikan ke topik. Lengkapi bidang-bidang berikut:

- Nama topik. Masukkan nama untuk mengidentifikasi pesan, seperti `motors/Fulton-A32/status`.
- Muatan pesan. Masukkan yang berikut ini:

```
{
  "messageId": 100,
  "sensorData": {
    "pressure": 39
  }
}
```

 Note

Ubah `messageId` setiap kali Anda mempublikasikan pesan baru.

6. Untuk Publish, pertahankan topik tetap sama, tetapi ubah payload ke nilai yang lebih besar dari nilai ambang batas yang Anda tentukan dalam model detektor (seperti **85**). `"pressure"`

7. Pilih Terbitkan.

Instans detektor yang Anda buat menghasilkan dan mengirimi Anda pesan Amazon SNS. Lanjutkan mengirim pesan dengan pembacaan tekanan di atas atau di bawah ambang tekanan (70 untuk contoh ini) untuk melihat detektor beroperasi.

Dalam contoh ini, Anda harus mengirim tiga pesan dengan pembacaan tekanan di bawah ambang batas untuk beralih kembali ke keadaan Normal dan menerima pesan Amazon SNS yang menunjukkan kondisi tekanan berlebih telah dihapus. Setelah kembali dalam keadaan Normal, satu pesan dengan pembacaan tekanan di atas batas menyebabkan detektor memasuki keadaan Berbahaya dan mengirim pesan Amazon SNS yang menunjukkan kondisi itu.

Sekarang setelah Anda membuat model input dan detektor sederhana, coba yang berikut ini.

- Lihat lebih banyak contoh model detektor (templat) di konsol.
- Ikuti langkah-langkah [Buat AWS IoT Events detektor untuk dua status menggunakan CLI](#) untuk membuat model input dan detektor menggunakan AWS CLI
- Pelajari detail yang [Ekspresi untuk memfilter, mengubah, dan memproses data peristiwa](#) digunakan dalam acara.
- Pelajari tentang [Tindakan yang didukung untuk menerima data dan memicu tindakan di AWS IoT Events](#).
- Jika ada sesuatu yang tidak bekerja, lihat [Pemecahan masalah AWS IoT Events](#).

# Praktik terbaik untuk AWS IoT Events

Ikuti praktik terbaik ini untuk mendapatkan manfaat maksimal AWS IoT Events.

Topik

- [Aktifkan CloudWatch pencatatan Amazon saat mengembangkan model AWS IoT Events detektor](#)
- [Publikasikan secara teratur untuk menyimpan model detektor Anda saat bekerja di AWS IoT Events konsol](#)

## Aktifkan CloudWatch pencatatan Amazon saat mengembangkan model AWS IoT Events detektor

Amazon CloudWatch memantau AWS sumber daya Anda dan aplikasi yang Anda jalankan AWS secara real time. Dengan CloudWatch, Anda mendapatkan visibilitas seluruh sistem ke dalam penggunaan sumber daya, kinerja aplikasi, dan kesehatan operasional. Saat Anda mengembangkan atau men-debug model AWS IoT Events detektor, CloudWatch membantu Anda mengetahui apa yang AWS IoT Events sedang dilakukan, dan kesalahan apa pun yang ditemuinya.

Untuk mengaktifkan CloudWatch

1. Jika Anda belum melakukannya, ikuti langkah-langkah [Menyiapkan izin untuk AWS IoT Events](#) untuk membuat peran dengan kebijakan terlampir yang memberikan izin untuk membuat dan mengelola CloudWatch log. AWS IoT Events
2. Pergi ke [AWS IoT Events konsol](#).
3. Pada panel navigasi, silakan pilih Pengaturan.
4. Pada halaman Pengaturan, pilih Edit.
5. Pada halaman Edit opsi pencatatan, di bagian Opsi pencatatan, lakukan hal berikut:
  - a. Untuk Tingkat verbositas, pilih opsi.
  - b. Untuk peran Pilih, pilih peran dengan izin yang cukup untuk melakukan tindakan logging yang Anda pilih.
  - c. (Opsional) Jika Anda memilih Debug untuk Level verbositas, Anda dapat menambahkan target Debug dengan melakukan hal berikut:
    - i. Di bawah target Debug, pilih Tambahkan Opsi Model.

- ii. Masukkan Nama Model Detektor dan (opsional) Key/Value untuk menentukan model detektor dan detektor tertentu (instance) untuk dicatat.

## 6. Pilih Perbarui.

Opsi pencatatan Anda berhasil diperbarui.

## Publikasikan secara teratur untuk menyimpan model detektor Anda saat bekerja di AWS IoT Events konsol

Saat Anda menggunakan AWS IoT Events konsol, pekerjaan Anda yang sedang berlangsung disimpan secara lokal di browser Anda. Namun, Anda harus memilih Publish untuk menyimpan model detektor Anda AWS IoT Events. Setelah Anda mempublikasikan model detektor, karya Anda yang diterbitkan akan tersedia di browser apa pun yang Anda gunakan untuk mengakses akun Anda.

### Note

Jika Anda tidak mempublikasikan karya Anda, itu tidak akan disimpan. Setelah mempublikasikan model detektor, Anda tidak dapat mengubah namanya. Namun, Anda dapat terus memodifikasi definisinya.

# Tutorial untuk kasus AWS IoT Events penggunaan

AWS IoT Events tutorial menyediakan kumpulan prosedur yang mencakup berbagai aspek AWS IoT Events, dari pengaturan dasar hingga kasus penggunaan yang lebih spesifik. Setiap tutorial menunjukkan contoh skenario praktis, membantu Anda membangun keterampilan dunia nyata dalam membuat model detektor, mengonfigurasi input, menyiapkan tindakan, dan mengintegrasikan dengan AWS layanan lain untuk menciptakan solusi IoT yang kuat.

Bab ini menunjukkan kepada Anda bagaimana untuk:

- Dapatkan bantuan untuk memutuskan status mana yang akan disertakan dalam model detektor Anda, dan tentukan apakah Anda memerlukan satu instance detektor atau beberapa.
- Ikuti contoh yang menggunakan AWS CLI.
- Buat input untuk menerima data telemetri dari perangkat dan model detektor untuk memantau dan melaporkan status perangkat yang mengirimkan data tersebut.
- Tinjau batasan dan batasan input, model detektor, dan AWS IoT Events layanan.
- Lihat contoh model detektor yang lebih kompleks, dengan komentar disertakan.

Topik

- [Menggunakan AWS IoT Events untuk memantau perangkat IoT Anda](#)
- [Buat AWS IoT Events detektor untuk dua status menggunakan CLI](#)
- [AWS IoT Events pembatasan dan batasan model detektor](#)
- [Contoh komentar: Kontrol suhu HVAC dengan AWS IoT Events](#)

## Menggunakan AWS IoT Events untuk memantau perangkat IoT Anda

Anda dapat menggunakannya AWS IoT Events untuk memantau perangkat atau proses Anda, dan mengambil tindakan berdasarkan peristiwa penting. Untuk melakukannya, ikuti langkah-langkah dasar ini:

## Buat masukan

Anda harus memiliki cara agar perangkat dan proses Anda memasukkan data telemetri. AWS IoT Events Anda melakukan ini dengan mengirim pesan sebagai input ke AWS IoT Events. Anda dapat mengirim pesan sebagai input dengan beberapa cara:

- Gunakan [BatchPutMessage](#) operasi.
- [Tentukan iotEvents aturan-tindakan untuk mesin aturan.AWS IoT Core](#) Aturan-tindakan meneruskan data pesan dari masukan Anda ke. AWS IoT Events
- Di AWS IoT Analytics, gunakan [CreateDataset](#) operasi untuk membuat kumpulan data dengan `contentDeliveryRules`. Aturan-aturan ini menentukan AWS IoT Events input di mana konten kumpulan data dikirim secara otomatis.
- Tentukan [tindakan iotEvents](#) dalam model AWS IoT Events `detectorOnInput`, `onExit` atau peristiwa. `transitionEvents` Informasi tentang contoh model detektor dan peristiwa yang memulai tindakan dimasukkan kembali ke sistem sebagai input dengan nama yang Anda tentukan.

Sebelum perangkat Anda mulai mengirim data dengan cara ini, Anda harus menentukan satu atau lebih input. Untuk melakukannya, berikan setiap input nama dan tentukan bidang mana dalam data pesan masuk yang dimonitor input. AWS IoT Events menerima masukannya, dalam bentuk payload JSON, dari banyak sumber. Setiap input dapat ditindaklanjuti dengan sendirinya, atau dikombinasikan dengan input lain untuk mendeteksi peristiwa yang lebih kompleks.

## Buat model detektor

Tentukan model detektor (model peralatan atau proses Anda) menggunakan status. Untuk setiap status, Anda menentukan logika bersyarat (Boolean) yang mengevaluasi input masuk untuk mendeteksi kejadian penting. Ketika suatu peristiwa terdeteksi, peristiwa dapat mengubah status atau memulai tindakan yang dibuat khusus atau yang telah ditentukan sebelumnya menggunakan layanan lain. AWS Anda dapat menentukan peristiwa tambahan yang memulai tindakan saat memasuki atau keluar dari status dan, secara opsional, ketika suatu kondisi terpenuhi.

Dalam tutorial ini, Anda mengirim pesan Amazon SNS sebagai tindakan ketika model memasuki atau keluar dari status tertentu.

## Memantau perangkat atau proses

Jika Anda memantau beberapa perangkat atau proses, Anda menentukan bidang di setiap input yang mengidentifikasi perangkat tertentu atau memproses input berasal. (Lihat key bidang `diCreateDetectorModel`.) Ketika perangkat baru diidentifikasi (nilai baru terlihat di bidang input

yang diidentifikasi oleh key), detektor dibuat. (Setiap detektor adalah contoh dari model detektor.) Kemudian detektor baru terus merespons input yang berasal dari perangkat itu hingga model detektornya diperbarui atau dihapus.

Jika Anda memantau satu proses (meskipun beberapa perangkat atau subproses mengirim input), Anda tidak menentukan bidang identifikasi key unik. Dalam hal ini, detektor tunggal (instance) dibuat ketika input pertama tiba.

Kirim pesan sebagai input ke model detektor Anda

Ada beberapa cara untuk mengirim pesan dari perangkat atau memproses sebagai input ke AWS IoT Events detektor yang tidak mengharuskan Anda melakukan pemformatan tambahan pada pesan. Dalam tutorial ini, Anda menggunakan AWS IoT konsol untuk menulis aturan [AWS IoT Events tindakan](#) untuk mesin AWS IoT Core aturan yang meneruskan data pesan Anda. AWS IoT Events Untuk melakukan ini, Anda mengidentifikasi input dengan nama. Kemudian Anda terus menggunakan AWS IoT konsol untuk menghasilkan beberapa pesan yang diteruskan sebagai input ke. AWS IoT Events

## Bagaimana Anda tahu status mana yang Anda butuhkan dalam model detektor?

Untuk menentukan status apa yang harus dimiliki model detektor Anda, pertama-tama putuskan tindakan apa yang dapat Anda ambil. Misalnya, jika mobil Anda menggunakan bensin, Anda melihat pengukur bahan bakar ketika Anda memulai perjalanan untuk melihat apakah Anda perlu mengisi bahan bakar. Di sini Anda memiliki satu tindakan: beri tahu pengemudi untuk “pergi mendapatkan bensin”. Model detektor Anda membutuhkan dua status: “mobil tidak membutuhkan bahan bakar”, dan “mobil memang membutuhkan bahan bakar”. Secara umum, Anda ingin menentukan satu status untuk setiap tindakan yang mungkin, ditambah satu lagi untuk saat tidak ada tindakan yang diperlukan. Ini berfungsi bahkan jika tindakan itu sendiri lebih rumit. Misalnya, Anda mungkin ingin mencari dan memasukkan informasi tentang di mana menemukan pompa bensin terdekat, atau harga termurah, tetapi Anda melakukan ini ketika Anda mengirim pesan untuk “pergi mendapatkan bensin”.

Untuk memutuskan status mana yang akan dimasukkan selanjutnya, Anda melihat input. Input berisi informasi yang Anda butuhkan untuk memutuskan negara bagian mana Anda seharusnya berada. Untuk membuat input, Anda memilih satu atau beberapa bidang dalam pesan yang dikirim oleh perangkat atau proses yang membantu Anda memutuskan. Dalam contoh ini, Anda memerlukan satu input yang memberi tahu Anda tingkat bahan bakar saat ini (“persen penuh”).

Mungkin mobil Anda mengirim Anda beberapa pesan berbeda, masing-masing dengan beberapa bidang berbeda. Untuk membuat input ini, Anda harus memilih pesan dan bidang yang melaporkan tingkat pengukur gas saat ini. Panjang perjalanan yang akan Anda ambil (“jarak ke tujuan”) dapat di-hardcode untuk menjaga hal-hal sederhana; Anda dapat menggunakan panjang perjalanan rata-rata Anda. Anda akan melakukan beberapa perhitungan berdasarkan input (berapa galon yang diterjemahkan sepenuhnya oleh persen itu? adalah panjang perjalanan rata-rata lebih besar dari mil yang dapat Anda tempuh, mengingat galon yang Anda miliki dan rata-rata “mil per galon” Anda). Anda melakukan perhitungan ini dan mengirim pesan dalam acara.

Sejauh ini Anda memiliki dua status dan satu input. Anda memerlukan acara dalam keadaan pertama yang melakukan perhitungan berdasarkan input dan memutuskan apakah akan pergi ke keadaan kedua. Itu adalah peristiwa transisi. (`transitionEvents` berada dalam daftar `onInput` acara negara bagian. Saat menerima masukan dalam keadaan pertama ini, acara melakukan transisi ke keadaan kedua, jika acara `condition` terpenuhi.) Ketika Anda mencapai status kedua, Anda mengirim pesan segera setelah Anda memasuki negara bagian. (Anda menggunakan `onEnter` acara. Saat memasuki keadaan kedua, acara ini mengirimkan pesan. Tidak perlu menunggu masukan lain tiba.) Ada jenis acara lain, tetapi hanya itu yang Anda butuhkan untuk contoh sederhana.

Jenis acara lainnya adalah `onExit` dan `onInput`. Segera setelah input diterima, dan kondisi terpenuhi, suatu `onInput` peristiwa melakukan tindakan yang ditentukan. Ketika operasi keluar dari keadaan saat ini, dan kondisi terpenuhi, `onExit` acara melakukan tindakan yang ditentukan.

Apa kau melewatkan sesuatu? Ya, bagaimana Anda kembali ke keadaan “mobil tidak perlu bahan bakar” pertama? Setelah Anda mengisi tangki bensin Anda, input menunjukkan tangki penuh. Dalam keadaan kedua Anda, Anda memerlukan peristiwa transisi kembali ke status pertama yang terjadi ketika input diterima (dalam `onInput`: peristiwa status kedua). Ini harus beralih kembali ke keadaan pertama jika perhitungannya menunjukkan bahwa Anda sekarang memiliki cukup gas untuk membawa Anda ke tempat yang Anda inginkan.

Itulah dasar-dasarnya. Beberapa model detektor menjadi lebih kompleks dengan menambahkan status yang mencerminkan input penting, bukan hanya tindakan yang mungkin. Misalnya, Anda mungkin memiliki tiga status dalam model detektor yang melacak suhu: keadaan “normal”, keadaan “terlalu panas”, dan status “masalah potensial”. Anda beralih ke keadaan masalah potensial ketika suhu naik di atas tingkat tertentu, tetapi belum menjadi terlalu panas. Anda tidak ingin mengirim alarm kecuali tetap pada suhu ini selama lebih dari 15 menit. Jika suhu kembali normal sebelum itu, detektor bertransisi kembali ke keadaan normal. Jika timer kedaluwarsa, detektor bertransisi ke keadaan terlalu panas dan mengirimkan alarm, hanya untuk berhati-hati. Anda dapat melakukan

hal yang sama menggunakan variabel dan serangkaian kondisi acara yang lebih kompleks. Tetapi seringkali lebih mudah untuk menggunakan negara lain untuk, pada dasarnya, menyimpan hasil perhitungan Anda.

## Bagaimana Anda tahu jika Anda memerlukan satu contoh detektor atau beberapa?

Untuk memutuskan berapa banyak contoh yang Anda butuhkan, tanyakan pada diri sendiri “Apa yang ingin Anda ketahui?” Katakanlah Anda ingin tahu seperti apa cuaca hari ini. Apakah hujan (negara bagian)? Apakah Anda perlu mengambil payung (tindakan)? Anda dapat memiliki sensor yang melaporkan suhu, sensor lain yang melaporkan kelembaban, dan sensor lain yang melaporkan tekanan barometrik, kecepatan dan arah angin, dan curah hujan. Tetapi Anda harus memantau semua sensor ini bersama-sama untuk menentukan keadaan cuaca (hujan, salju, mendung, cerah) dan tindakan yang tepat untuk diambil (ambil payung atau gunakan tabir surya). Terlepas dari jumlah sensor, Anda memerlukan satu contoh detektor untuk memantau keadaan cuaca dan memberi tahu Anda tindakan mana yang harus diambil.

Tetapi jika Anda adalah peramal cuaca untuk wilayah Anda, Anda mungkin memiliki beberapa contoh array sensor tersebut, yang terletak di lokasi yang berbeda di seluruh wilayah. Orang-orang di setiap lokasi perlu tahu seperti apa cuaca di lokasi itu. Dalam hal ini, Anda memerlukan beberapa contoh detektor Anda. Data yang dilaporkan oleh setiap sensor di setiap lokasi harus menyertakan bidang yang telah Anda tetapkan sebagai key bidang. Bidang ini memungkinkan AWS IoT Events untuk membuat instance detektor untuk area tersebut, dan kemudian melanjutkan untuk merutekan informasi ini ke instance detektor itu saat terus berdatangan. Tidak ada lagi rambut yang rusak atau hidung yang terbakar sinar matahari!

Pada dasarnya, Anda memerlukan satu instance detektor jika Anda memiliki satu situasi (satu proses atau satu lokasi) untuk dipantau. Jika Anda memiliki banyak situasi (lokasi, proses) untuk dipantau, Anda memerlukan beberapa instance detektor.

## Buat AWS IoT Events detektor untuk dua status menggunakan CLI

Dalam contoh ini, kita menyebut AWS CLI perintah AWS IoT Events APIs using untuk membuat detektor yang memodelkan dua keadaan mesin: keadaan normal dan kondisi tekanan berlebih.

Ketika tekanan yang diukur di mesin melebihi ambang batas tertentu, model beralih ke keadaan tekanan berlebih dan mengirimkan pesan Amazon Simple Notification Service (Amazon SNS) untuk

mengingatkan teknisi tentang kondisi tersebut. Ketika tekanan turun di bawah ambang batas untuk tiga pembacaan tekanan berturut-turut, model kembali ke keadaan normal dan mengirimkan pesan Amazon SNS lainnya sebagai konfirmasi bahwa kondisi telah dihapus. Kami memerlukan tiga pembacaan berturut-turut di bawah ambang tekanan untuk menghilangkan kemungkinan kegagalan pesan berlebihan/normal jika terjadi fase pemulihan nonlinier atau pembacaan pemulihan anomali satu kali.

Berikut ini adalah ikhtisar langkah-langkah untuk membuat detektor.

Buat input.

Untuk memantau perangkat dan proses Anda, mereka harus memiliki cara untuk mendapatkan data telemetri AWS IoT Events. Ini dilakukan dengan mengirim pesan sebagai input ke AWS IoT Events. Anda dapat melakukannya dengan dua cara:

- Gunakan [BatchPutMessage](#) operasi. Metode ini mudah tetapi mengharuskan perangkat atau proses Anda dapat mengakses AWS IoT Events API melalui SDK atau file. AWS CLI
- Di AWS IoT Core, tulis aturan [AWS IoT Events tindakan](#) untuk mesin AWS IoT Core aturan yang meneruskan data pesan Anda ke dalam AWS IoT Events. Ini mengidentifikasi input dengan nama. Gunakan metode ini jika perangkat atau proses Anda dapat, atau sudah, mengirim pesan melalui AWS IoT Core. Metode ini umumnya membutuhkan daya komputasi yang lebih sedikit dari perangkat.
- Dalam AWS IoT Analytics, gunakan [CreateDataset](#) operasi untuk membuat kumpulan data dengan `contentDeliveryRules` yang menentukan AWS IoT Events input, di mana konten kumpulan data dikirim secara otomatis. Gunakan metode ini jika Anda ingin mengontrol perangkat atau proses berdasarkan data yang dikumpulkan atau dianalisis. AWS IoT Analytics

Sebelum perangkat Anda dapat mengirim data dengan cara ini, Anda harus menentukan satu atau lebih input. Untuk melakukannya, berikan setiap input nama dan tentukan bidang mana dalam data pesan masuk yang dipantau input.

Buat model detektor

Buat model detektor (model peralatan atau proses Anda) menggunakan status. Untuk setiap status, tentukan logika kondisional (Boolean) yang mengevaluasi input yang masuk untuk mendeteksi peristiwa penting. Ketika suatu peristiwa terdeteksi, peristiwa dapat mengubah status atau memulai tindakan yang dibuat khusus atau yang telah ditentukan sebelumnya menggunakan layanan lain. AWS Anda dapat menentukan peristiwa tambahan yang memulai tindakan saat memasuki atau keluar dari status dan, secara opsional, ketika suatu kondisi terpenuhi.

## Memantau beberapa perangkat atau proses

Jika Anda memantau beberapa perangkat atau proses dan ingin melacak masing-masing perangkat secara terpisah, tentukan bidang di setiap input yang mengidentifikasi perangkat tertentu atau memproses asal input. Lihat key bidang di `CreateDetectorModel`. Ketika perangkat baru diidentifikasi (nilai baru terlihat di bidang input yang diidentifikasi oleh key), instance detektor dibuat. Instance detektor baru terus merespons input yang berasal dari perangkat tertentu hingga model detektornya diperbarui atau dihapus. Anda memiliki banyak detektor unik (instance) karena ada nilai unik di bidang input key.

## Memantau satu perangkat atau proses

Jika Anda memantau satu proses (meskipun beberapa perangkat atau subproses mengirim input), Anda tidak menentukan bidang identifikasi key unik. Dalam hal ini, detektor tunggal (instance) dibuat ketika input pertama tiba. Misalnya, Anda mungkin memiliki sensor suhu di setiap ruangan rumah, tetapi hanya satu unit HVAC untuk memanaskan atau mendinginkan seluruh rumah. Jadi Anda hanya dapat mengontrol ini sebagai satu proses, bahkan jika setiap penghuni kamar ingin suara (input) mereka menang.

## Kirim pesan dari perangkat atau proses Anda sebagai input ke model detektor Anda

Kami menjelaskan beberapa cara untuk mengirim pesan dari perangkat atau proses sebagai input ke AWS IoT Events detektor dalam input. Setelah Anda membuat input dan membangun model detektor, Anda siap untuk mulai mengirim data.

### Note

Saat Anda membuat model detektor, atau memperbarui model yang sudah ada, dibutuhkan beberapa menit sebelum model detektor baru atau yang diperbarui mulai menerima pesan dan membuat detektor (instance). Jika model detektor diperbarui, selama waktu ini Anda mungkin terus melihat perilaku berdasarkan versi sebelumnya.

## Topik

- [Buat AWS IoT Events input untuk menangkap data perangkat](#)
- [Buat model detektor untuk mewakili status perangkat di AWS IoT Events](#)
- [Kirim pesan sebagai input ke detektor di AWS IoT Events](#)

## Buat AWS IoT Events input untuk menangkap data perangkat

Saat menyiapkan input AWS IoT Events, Anda dapat memanfaatkan untuk menentukan cara perangkat Anda mengkomunikasikan data sensor. AWS CLI Misalnya, jika perangkat Anda mengirim pesan berformat JSON dengan pengenalan motor dan pembacaan sensor, Anda dapat menangkap data ini dengan membuat input yang memetakan atribut tertentu dari pesan, seperti tekanan dan ID motor. Proses dimulai dengan mendefinisikan input dalam file JSON, menentukan titik data yang relevan, dan menggunakan AWS CLI untuk mendaftarkan input untuk. AWS IoT Events Hal ini memungkinkan AWS IoT untuk memantau dan merespons kondisi kritis berdasarkan data sensor real-time.

Sebagai contoh, misalkan perangkat Anda mengirim pesan dengan format berikut.

```
{
  "motorid": "Fulton-A32",
  "sensorData": {
    "pressure": 23,
    "temperature": 47
  }
}
```

Anda dapat membuat input untuk menangkap `pressure` data dan `motorid` (yang mengidentifikasi perangkat tertentu yang mengirim pesan) menggunakan AWS CLI perintah berikut.

```
aws iotevents create-input --cli-input-json file://pressureInput.json
```

File `pressureInput.json` berisi yang berikut ini.

```
{
  "inputName": "PressureInput",
  "inputDescription": "Pressure readings from a motor",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorData.pressure" },
      { "jsonPath": "motorid" }
    ]
  }
}
```

Saat Anda membuat input sendiri, ingatlah untuk terlebih dahulu mengumpulkan pesan contoh sebagai file JSON dari perangkat atau proses Anda. Anda dapat menggunakannya untuk membuat input dari konsol atau CLI.

## Buat model detektor untuk mewakili status perangkat di AWS IoT Events

Di [Buat AWS IoT Events input untuk menangkap data perangkat](#), Anda membuat input berdasarkan pesan yang melaporkan data tekanan dari motor. Untuk melanjutkan contoh, berikut adalah model detektor yang merespons peristiwa tekanan berlebih pada motor.

Anda membuat dua status: "Normal", dan "Dangerous". Setiap detektor (instance) memasuki status Normal "" saat dibuat. Instance dibuat ketika input dengan nilai unik untuk key "motorid" tiba.

Jika instance detektor menerima pembacaan tekanan 70 atau lebih besar, ia memasuki status Dangerous "" dan mengirim pesan Amazon SNS sebagai peringatan. Jika pembacaan tekanan kembali normal (kurang dari 70) untuk tiga input berturut-turut, detektor kembali ke status "Normal" dan mengirim pesan Amazon SNS lainnya sebagai semua jelas.

Model detektor contoh ini mengasumsikan Anda telah membuat dua topik Amazon SNS yang Amazon Resource Names ARNs () ditampilkan dalam definisi "targetArn": "arn:aws:sns:us-east-1:123456789012:underPressureAction" sebagai dan. "targetArn": "arn:aws:sns:us-east-1:123456789012:pressureClearedAction"

Untuk informasi selengkapnya, lihat [Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon](#) dan, lebih khusus lagi, dokumentasi [CreateTopic](#) operasi di Referensi API Layanan Pemberitahuan Sederhana Amazon.

Contoh ini juga mengasumsikan Anda telah membuat peran AWS Identity and Access Management (IAM) dengan izin yang sesuai. ARN dari peran ini ditunjukkan dalam definisi model detektor sebagai. "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole" Ikuti langkah-langkah [Menyiapkan izin untuk AWS IoT Events](#) untuk membuat peran ini dan salin ARN peran di tempat yang sesuai dalam definisi model detektor.

Anda dapat membuat model detektor menggunakan AWS CLI perintah berikut.

```
aws iotevents create-detector-model --cli-input-json file://motorDetectorModel.json
```

File "motorDetectorModel.json" berisi yang berikut ini.

```
{
  "detectorModelName": "motorDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Normal",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "pressureThresholdBreach",
                    "value": "0"
                  }
                }
              ]
            }
          ]
        },
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "Overpressurized",
              "condition": "$input.PressureInput.sensorData.pressure > 70",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "pressureThresholdBreach",
                    "value": "$variable.pressureThresholdBreach + 3"
                  }
                }
              ],
              "nextState": "Dangerous"
            }
          ]
        }
      },
      {
        "stateName": "Dangerous",
        "onEnter": {
```

```

    "events": [
      {
        "eventName": "Pressure Threshold Breached",
        "condition": "$variable.pressureThresholdBreached > 1",
        "actions": [
          {
            "sns": {
              "targetArn": "arn:aws:sns:us-
east-1:123456789012:underPressureAction"
            }
          }
        ]
      }
    ],
    "onInput": {
      "events": [
        {
          "eventName": "Overpressurized",
          "condition": "$input.PressureInput.sensorData.pressure > 70",
          "actions": [
            {
              "setVariable": {
                "variableName": "pressureThresholdBreached",
                "value": "3"
              }
            }
          ]
        },
        {
          "eventName": "Pressure Okay",
          "condition": "$input.PressureInput.sensorData.pressure <= 70",
          "actions": [
            {
              "setVariable": {
                "variableName": "pressureThresholdBreached",
                "value": "$variable.pressureThresholdBreached - 1"
              }
            }
          ]
        }
      ]
    },
    "transitionEvents": [
      {

```

```
        "eventName": "BackToNormal",
        "condition": "$input.PressureInput.sensorData.pressure <= 70 &&
$variable.pressureThresholdBreached <= 1",
        "nextState": "Normal"
    }
]
},
"onExit": {
    "events": [
        {
            "eventName": "Normal Pressure Restored",
            "condition": "true",
            "actions": [
                {
                    "sns": {
                        "targetArn": "arn:aws:sns:us-
east-1:123456789012:pressureClearedAction"
                    }
                }
            ]
        }
    ]
}
},
"initialStateName": "Normal"
},
"key" : "motorid",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}
```

## Kirim pesan sebagai input ke detektor di AWS IoT Events

Anda sekarang telah menentukan input yang mengidentifikasi bidang penting dalam pesan yang dikirim dari perangkat (lihat [Buat AWS IoT Events input untuk menangkap data perangkat](#)). Di bagian sebelumnya, Anda membuat sebuah `detector model` yang merespons peristiwa tekanan berlebih di motor (lihat [Buat model detektor untuk mewakili status perangkat di AWS IoT Events](#)).

Untuk melengkapi contoh, kirim pesan dari perangkat (dalam hal ini komputer dengan yang AWS CLI diinstal) sebagai input ke detektor.

**Note**

Saat Anda membuat model detektor atau memperbarui yang sudah ada, dibutuhkan beberapa menit sebelum model detektor baru atau yang diperbarui mulai menerima pesan dan membuat detektor (instance). Jika Anda memperbarui model detektor, selama waktu ini Anda mungkin terus melihat perilaku berdasarkan versi sebelumnya.

Gunakan AWS CLI perintah berikut untuk mengirim pesan dengan data yang melanggar ambang batas.

```
aws iotevents-data batch-put-message --cli-input-json file://highPressureMessage.json --cli-binary-format raw-in-base64-out
```

File "highPressureMessage.json" berisi yang berikut ini.

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "PressureInput",
      "payload": "{\"motorid\": \"Fulton-A32\", \"sensorData\": {\"pressure\": 80, \"temperature\": 39} }"
    }
  ]
}
```

Anda harus mengubah messageId setiap pesan yang dikirim. Jika Anda tidak mengubahnya, AWS IoT Events sistem menghapus duplikasi pesan. AWS IoT Events mengabaikan pesan jika memiliki pesan yang messageID sama dengan pesan lain yang dikirim dalam lima menit terakhir.

Pada titik ini, detektor (instance) dibuat untuk memantau peristiwa untuk motor "Fulton-A32". Detektor ini memasuki "Normal" status saat dibuat. Tetapi karena kami mengirim nilai tekanan di atas ambang batas, itu segera beralih ke "Dangerous" negara. Saat melakukannya, detektor mengirim pesan ke titik akhir Amazon SNS yang ARN-nya. `arn:aws:sns:us-east-1:123456789012:underPressureAction`

Jalankan AWS CLI perintah berikut untuk mengirim pesan dengan data yang berada di bawah ambang tekanan.

```
aws iotevents-data batch-put-message --cli-input-json file://normalPressureMessage.json
--cli-binary-format raw-in-base64-out
```

File `normalPressureMessage.json` berisi yang berikut ini.

```
{
  "messages": [
    {
      "messageId": "00002",
      "inputName": "PressureInput",
      "payload": "{\"motorid\": \"Fulton-A32\", \"sensorData\": {\"pressure\": 60,
        \"temperature\": 29} }"
    }
  ]
}
```

Anda harus mengubah file `messageId` dalam setiap kali Anda memanggil `BatchPutMessage` perintah dalam jangka waktu lima menit. Kirim pesan dua kali lagi. Setelah pesan dikirim tiga kali, detektor (contoh) untuk motor "Fulton-A32" mengirim pesan ke titik akhir Amazon SNS `"arn:aws:sns:us-east-1:123456789012:pressureClearedAction"` dan memasuki kembali status. `"Normal"`

#### Note

Anda dapat mengirim beberapa pesan sekaligus dengan `BatchPutMessage`. Namun, urutan pemrosesan pesan ini tidak dijamin. Untuk menjamin pesan (input) diproses secara berurutan, kirimkan satu per satu dan tunggu respons yang berhasil setiap kali API dipanggil.

Berikut ini adalah contoh muatan pesan SNS yang dibuat oleh contoh model detektor yang dijelaskan di bagian ini.

pada acara "Ambang Tekanan Dilanggar"

```
IoT> {
  "eventTime":1558129816420,
  "payload":{
    "actionExecutionId":"5d7444df-a655-3587-a609-dbd7a0f55267",
    "detector":{
```

```

    "detectorModelName":"motorDetectorModel",
    "keyValue":"Fulton-A32",
    "detectorModelVersion":"1"
  },
  "eventTriggerDetails":{
    "inputName":"PressureInput",
    "messageId":"00001",
    "triggerType":"Message"
  },
  "state":{
    "stateName":"Dangerous",
    "variables":{
      "pressureThresholdBreach":3
    },
    "timers":{}
  }
},
"eventName":"Pressure Threshold Breached"
}

```

pada acara “Tekanan Normal Dipulihkan”

```

IoT> {
  "eventTime":1558129925568,
  "payload":{
    "actionExecutionId":"7e25fd38-2533-303d-899f-c979792a12cb",
    "detector":{
      "detectorModelName":"motorDetectorModel",
      "keyValue":"Fulton-A32",
      "detectorModelVersion":"1"
    },
    "eventTriggerDetails":{
      "inputName":"PressureInput",
      "messageId":"00004",
      "triggerType":"Message"
    },
    "state":{
      "stateName":"Dangerous",
      "variables":{
        "pressureThresholdBreach":0
      },
      "timers":{}
    }
  }
}

```

```
},  
  "eventName": "Normal Pressure Restored"  
}
```

Jika Anda telah menentukan timer apa pun, statusnya saat ini juga ditampilkan dalam muatan pesan SNS.

Muatan pesan berisi informasi tentang status detektor (instance) pada saat pesan dikirim (yaitu, pada saat tindakan SNS dijalankan). Anda dapat menggunakan [https://docs.aws.amazon.com/iotevents/latest/apireference/API\\_iotevents-data\\_DescribeDetector.html](https://docs.aws.amazon.com/iotevents/latest/apireference/API_iotevents-data_DescribeDetector.html) operasi untuk mendapatkan informasi serupa tentang keadaan detektor.

## AWS IoT Events pembatasan dan batasan model detektor

Hal-hal berikut ini penting untuk dipertimbangkan saat membuat model detektor.

### Cara menggunakan **actions** bidang

`actionsBidang` adalah daftar objek. Anda dapat memiliki lebih dari satu objek, tetapi hanya satu tindakan yang diizinkan di setiap objek.

#### Example

```
"actions": [  
  {  
    "setVariable": {  
      "variableName": "pressureThresholdBreached",  
      "value": "$variable.pressureThresholdBreached - 1"  
    }  
  }  
  {  
    "setVariable": {  
      "variableName": "temperatureIsTooHigh",  
      "value": "$variable.temperatureIsTooHigh - 1"  
    }  
  }  
]
```

### Cara menggunakan **condition** bidang

`condition` diperlukan untuk `transitionEvents` dan opsional dalam kasus lain.

Jika `condition` bidang tidak ada, itu setara dengan `"condition": true`.

Hasil evaluasi ekspresi kondisi harus berupa nilai Boolean. Jika hasilnya bukan nilai Boolean, itu setara dengan `false` dan tidak akan memulai `actions` atau transisi ke yang `nextState` ditentukan dalam acara tersebut.

#### Ketersediaan nilai variabel

Secara default, jika nilai variabel ditetapkan dalam suatu peristiwa, nilai barunya tidak tersedia atau digunakan untuk mengevaluasi kondisi dalam peristiwa lain dalam grup yang sama. Nilai baru tidak tersedia atau digunakan dalam kondisi peristiwa di `onExit` bidang yang sama `onInput`, `onEnter` atau.

Atur `evaluationMethod` parameter dalam definisi model detektor untuk mengubah perilaku ini. Ketika `evaluationMethod` diatur ke `SERIAL`, variabel diperbarui dan kondisi peristiwa dievaluasi dalam urutan bahwa peristiwa didefinisikan. Jika tidak, ketika `evaluationMethod` disetel ke `BATCH` atau defaultnya, variabel dalam keadaan diperbarui dan peristiwa dalam keadaan dilakukan hanya setelah semua kondisi peristiwa dievaluasi.

Di "Dangerous" negara bagian, di `onInput` lapangan, `"$variable.pressureThresholdBreach"` dikurangi oleh satu "Pressure Okay" jika kondisi terpenuhi (ketika input saat ini memiliki tekanan kurang dari atau sama dengan 70).

```
{
  "eventName": "Pressure Okay",
  "condition": "$input.PressureInput.sensorData.pressure <= 70",
  "actions": [
    {
      "setVariable": {
        "variableName": "pressureThresholdBreach",
        "value": "$variable.pressureThresholdBreach - 1"
      }
    }
  ]
}
```

Detektor harus bertransisi kembali ke "Normal" keadaan ketika `"$variable.pressureThresholdBreach"` mencapai 0 (yaitu, ketika detektor telah menerima tiga pembacaan tekanan yang berdekatan kurang dari atau sama dengan 70). "BackToNormal" Peristiwa di `transitionEvents` harus

menguji "\$variable.pressureThresholdBreached" yang kurang dari atau sama dengan 1 (bukan 0), dan juga memverifikasi lagi bahwa nilai saat ini yang "\$input.PressureInput.sensorData.pressure" diberikan oleh kurang dari atau sama dengan 70.

```
"transitionEvents": [  
  {  
    "eventName": "BackToNormal",  
    "condition": "$input.PressureInput.sensorData.pressure <= 70 &&  
$variable.pressureThresholdBreached <= 1",  
    "nextState": "Normal"  
  }  
]
```

Jika tidak, jika kondisi hanya menguji nilai variabel, dua pembacaan normal diikuti dengan pembacaan tekanan berlebih akan memenuhi kondisi dan transisi kembali ke "Normal" keadaan. Kondisi ini melihat nilai yang "\$variable.pressureThresholdBreached" diberikan selama waktu sebelumnya input diproses. Nilai variabel diatur ulang ke 3 dalam "Overpressurized" acara tersebut, tetapi ingat bahwa nilai baru ini belum tersedia untuk siapa pun condition.

Secara default, setiap kali kontrol memasuki onInput bidang, a hanya condition dapat melihat nilai variabel seperti pada awal pemrosesan input, sebelum diubah oleh tindakan apa pun yang ditentukan dalam onInput. Hal yang sama berlaku untuk onEnter dan onExit. Setiap perubahan yang dibuat ke variabel ketika kita masuk atau keluar dari status tidak tersedia untuk kondisi lain yang ditentukan dalam onExit bidang yang sama onEnter atau.

### Latensi saat memperbarui model detektor

Jika Anda memperbarui, menghapus, dan membuat ulang model detektor (lihat [UpdateDetectorModel](#)), ada beberapa penundaan sebelum semua detektor yang muncul (instance) dihapus dan model baru digunakan untuk membuat ulang detektor. Mereka dibuat ulang setelah model detektor baru mulai berlaku dan input baru tiba. Selama waktu ini input mungkin terus diproses oleh detektor yang dihasilkan oleh versi sebelumnya dari model detektor. Selama periode ini, Anda mungkin terus menerima peringatan yang ditentukan oleh model detektor sebelumnya.

## Spasi di tombol input

Spasi diperbolehkan dalam kunci input, tetapi referensi ke kunci harus diapit dalam backticks, baik dalam definisi atribut input dan ketika nilai kunci direferensikan dalam ekspresi. Misalnya, diberikan payload pesan seperti berikut:

```
{
  "motor id": "A32",
  "sensorData" {
    "motor pressure": 56,
    "motor temperature": 39
  }
}
```

Gunakan yang berikut ini untuk menentukan input.

```
{
  "inputName": "PressureInput",
  "inputDescription": "Pressure readings from a motor",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorData.`motor pressure`" },
      { "jsonPath": "`motor id`" }
    ]
  }
}
```

Dalam ekspresi kondisional, Anda harus merujuk ke nilai kunci tersebut menggunakan backticks juga.

```
$input.PressureInput.sensorData.`motor pressure`
```

## Contoh komentar: Kontrol suhu HVAC dengan AWS IoT Events

Beberapa contoh file JSON berikut memiliki komentar sebaris, yang membuatnya JSON tidak valid. Versi lengkap dari contoh-contoh ini, tanpa komentar, tersedia di [Contoh: Menggunakan kontrol suhu HVAC dengan AWS IoT Events](#).

Contoh ini mengimplementasikan model kontrol termostat yang memberi Anda kemampuan untuk melakukan hal berikut.

- Tentukan hanya satu model detektor yang dapat digunakan untuk memantau dan mengontrol beberapa area. Sebuah instance detektor dibuat untuk setiap area.
- Menelan data suhu dari beberapa sensor di setiap area kontrol.
- Ubah titik setel suhu untuk suatu area.
- Tetapkan parameter operasional untuk setiap area dan atur ulang parameter ini saat instance sedang digunakan.
- Menambah atau menghapus sensor secara dinamis dari suatu area.
- Tentukan runtime minimum untuk melindungi unit pemanas dan pendingin.
- Tolak pembacaan sensor anomali.
- Tentukan titik setel darurat yang segera melibatkan pemanasan atau pendinginan jika ada satu sensor yang melaporkan suhu di atas atau di bawah ambang batas tertentu.
- Laporkan pembacaan anomali dan lonjakan suhu.

## Topik

- [Definisi input untuk model detektor di AWS IoT Events](#)
- [Buat definisi model AWS IoT Events detektor](#)
- [Gunakan BatchUpdateDetector untuk memperbarui model AWS IoT Events detektor](#)
- [Gunakan BatchPutMessage untuk input di AWS IoT Events](#)
- [Menelan pesan MQTT di AWS IoT Events](#)
- [Hasilkan pesan Amazon SNS di AWS IoT Events](#)
- [Konfigurasi DescribeDetector API di AWS IoT Events](#)
- [Gunakan mesin AWS IoT Core aturan untuk AWS IoT Events](#)

## Definisi input untuk model detektor di AWS IoT Events

Kami ingin membuat satu model detektor yang dapat kami gunakan untuk memantau dan mengontrol suhu di beberapa area berbeda. Setiap area dapat memiliki beberapa sensor yang melaporkan suhu. Kami berasumsi setiap area dilayani oleh satu unit pemanas dan satu unit pendingin yang dapat dinyalakan atau dimatikan untuk mengontrol suhu di area tersebut. Setiap area dikendalikan oleh satu instance detektor.

Karena area berbeda yang kami pantau dan kontrol mungkin memiliki karakteristik berbeda yang menuntut parameter kontrol yang berbeda, kami mendefinisikan 'seedTemperatureInput' untuk

menyediakan parameter tersebut untuk setiap area. Ketika kami mengirim salah satu pesan input ini AWS IoT Events, contoh model detektor baru dibuat yang memiliki parameter yang ingin kami gunakan di area itu. Berikut adalah definisi dari masukan tersebut.

Perintah CLI:

```
aws iotevents create-input --cli-input-json file://seedInput.json
```

Berkas: seedInput.json

```
{
  "inputName": "seedTemperatureInput",
  "inputDescription": "Temperature seed values.",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "areaId" },
      { "jsonPath": "desiredTemperature" },
      { "jsonPath": "allowedError" },
      { "jsonPath": "rangeHigh" },
      { "jsonPath": "rangeLow" },
      { "jsonPath": "anomalousHigh" },
      { "jsonPath": "anomalousLow" },
      { "jsonPath": "sensorCount" },
      { "jsonPath": "noDelay" }
    ]
  }
}
```

Respons:

```
{
  "inputConfiguration": {
    "status": "ACTIVE",
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/seedTemperatureInput",
    "lastUpdateTime": 1557519620.736,
    "creationTime": 1557519620.736,
    "inputName": "seedTemperatureInput",
    "inputDescription": "Temperature seed values."
  }
}
```

## Catatan

- Sebuah instance detektor baru dibuat untuk setiap unik yang 'areaId' diterima dalam pesan apa pun. Lihat 'key' bidang dalam 'areaDetectorModel' definisi.
- Suhu rata-rata dapat bervariasi dari 'desiredTemperature' 'allowedError' sebelum unit pemanas atau pendingin diaktifkan untuk area tersebut.
- Jika ada sensor yang melaporkan suhu di atas 'rangeHigh', detektor melaporkan lonjakan dan segera memulai unit pendingin.
- Jika ada sensor yang melaporkan suhu di bawah 'rangeLow', detektor melaporkan lonjakan dan segera memulai unit pemanas.
- Jika ada sensor yang melaporkan suhu di atas 'anomalousHigh' atau di bawah 'anomalousLow', detektor melaporkan pembacaan sensor anomali, tetapi mengabaikan pembacaan suhu yang dilaporkan.
- Ini 'sensorCount' memberi tahu detektor berapa banyak sensor yang melaporkan untuk area tersebut. Detektor menghitung suhu rata-rata di area tersebut dengan memberikan faktor berat yang sesuai untuk setiap pembacaan suhu yang diterimanya. Karena itu, detektor tidak perlu melacak apa yang dilaporkan setiap sensor, dan jumlah sensor dapat diubah secara dinamis, sesuai kebutuhan. Namun, jika sensor individu offline, detektor tidak akan mengetahui hal ini atau memberikan kelonggaran untuk itu. Kami menyarankan Anda membuat model detektor lain khusus untuk memantau status koneksi setiap sensor. Memiliki dua model detektor komplementer menyederhanakan desain keduanya.
- 'noDelay' Nilainya bisa true atau false. Setelah unit pemanas atau pendingin dihidupkan, unit harus tetap menyala selama waktu minimum tertentu untuk melindungi integritas unit dan memperpanjang masa operasinya. Jika 'noDelay' disetel ke false, instance detektor memberlakukan penundaan sebelum mematikan unit pendingin dan pemanas, untuk memastikan bahwa mereka dijalankan untuk waktu minimum. Jumlah detik penundaan telah di-hardcode dalam definisi model detektor karena kami tidak dapat menggunakan nilai variabel untuk mengatur timer.

'temperatureInput' Ini digunakan untuk mengirimkan data sensor ke instance detektor.

## Perintah CLI:

```
aws iotevents create-input --cli-input-json file://temperatureInput.json
```

Berkas: temperatureInput.json

```
{
  "inputName": "temperatureInput",
  "inputDescription": "Temperature sensor unit data.",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorId" },
      { "jsonPath": "areaId" },
      { "jsonPath": "sensorData.temperature" }
    ]
  }
}
```

Respons:

```
{
  "inputConfiguration": {
    "status": "ACTIVE",
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/temperatureInput",
    "lastUpdateTime": 1557519707.399,
    "creationTime": 1557519707.399,
    "inputName": "temperatureInput",
    "inputDescription": "Temperature sensor unit data."
  }
}
```

## Catatan

- 'sensorId' Ini tidak digunakan oleh instance detektor contoh untuk mengontrol atau memantau sensor secara langsung. Ini secara otomatis diteruskan ke notifikasi yang dikirim oleh instance detektor. Dari sana, dapat digunakan untuk mengidentifikasi sensor yang gagal (misalnya, sensor yang secara teratur mengirimkan pembacaan anomali mungkin akan gagal), atau yang telah offline (ketika digunakan sebagai input ke model detektor tambahan yang memantau detak jantung perangkat). Ini juga 'sensorId' dapat membantu mengidentifikasi zona hangat atau dingin di suatu daerah jika pembacaannya secara teratur berbeda dari rata-rata.
- 'areaId' Ini digunakan untuk merutekan data sensor ke instance detektor yang sesuai. Sebuah instance detektor dibuat untuk setiap unik yang 'areaId' diterima dalam pesan apa pun. Lihat 'key' bidang dalam 'areaDetectorModel' definisi.

## Buat definisi model AWS IoT Events detektor

'areaDetectorModel' Contohnya memiliki komentar sebaris.

Perintah CLI:

```
aws iotevents create-detector-model --cli-input-json file://areaDetectorModel.json
```

Berkas: areaDetectorModel.json

```
{
  "detectorModelName": "areaDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "start",
        // In the 'start' state we set up the operation parameters of the new detector
        // instance.
        // We get here when the first input message arrives. If that is a
        // 'seedTemperatureInput'
        // message, we save the operation parameters, then transition to the 'idle'
        // state. If
        // the first message is a 'temperatureInput', we wait here until we get a
        // 'seedTemperatureInput' input to ensure our operation parameters are set.
        // We can
        // also reenter this state using the 'BatchUpdateDetector' API. This enables
        // us to
        // reset the operation parameters without needing to delete the detector
        // instance.
        "onEnter": {
          "events": [
            {
              "eventName": "prepare",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    // initialize 'sensorId' to an invalid value (0) until an actual
                    // sensor reading
                    // arrives
                    "variableName": "sensorId",
                    "value": "0"
                  }
                }
              ]
            }
          ]
        }
      }
    ]
  }
}
```

```

    }
  },
  {
    "setVariable": {
      // initialize 'reportedTemperature' to an invalid value (0.1) until
an actual
      // sensor reading arrives
      "variableName": "reportedTemperature",
      "value": "0.1"
    }
  },
  {
    "setVariable": {
      // When using 'BatchUpdateDetector' to re-enter this state, this
variable should
      // be set to true.
      "variableName": "resetMe",
      "value": "false"
    }
  }
]
}
],
},
"onInput": {
  "transitionEvents": [
    {
      "eventName": "initialize",
      "condition": "$input.seedTemperatureInput.sensorCount > 0",
      // When a 'seedTemperatureInput' message with a valid 'sensorCount' is
received,
      // we use it to set the operational parameters for the area to be
monitored.
      "actions": [
        {
          "setVariable": {
            "variableName": "rangeHigh",
            "value": "$input.seedTemperatureInput.rangeHigh"
          }
        },
        {
          "setVariable": {
            "variableName": "rangeLow",
            "value": "$input.seedTemperatureInput.rangeLow"

```

```
    }
  },
  {
    "setVariable": {
      "variableName": "desiredTemperature",
      "value": "$input.seedTemperatureInput.desiredTemperature"
    }
  },
  {
    "setVariable": {
      // Assume we're at the desired temperature when we start.
      "variableName": "averageTemperature",
      "value": "$input.seedTemperatureInput.desiredTemperature"
    }
  },
  {
    "setVariable": {
      "variableName": "allowedError",
      "value": "$input.seedTemperatureInput.allowedError"
    }
  },
  {
    "setVariable": {
      "variableName": "anomalousHigh",
      "value": "$input.seedTemperatureInput.anomalousHigh"
    }
  },
  {
    "setVariable": {
      "variableName": "anomalousLow",
      "value": "$input.seedTemperatureInput.anomalousLow"
    }
  },
  {
    "setVariable": {
      "variableName": "sensorCount",
      "value": "$input.seedTemperatureInput.sensorCount"
    }
  },
  {
    "setVariable": {
      "variableName": "noDelay",
      "value": "$input.seedTemperatureInput.noDelay == true"
    }
  }
}
```

```

    }
  ],
  "nextState": "idle"
},
{
  "eventName": "reset",
  "condition": "($variable.resetMe == true) &&
($input.temperatureInput.sensorData.temperature < $variable.anomalousHigh &&
$input.temperatureInput.sensorData.temperature > $variable.anomalousLow)",
  // This event is triggered if we have reentered the 'start' state using
the
  // 'BatchUpdateDetector' API with 'resetMe' set to true. When we
reenter using
  // 'BatchUpdateDetector' we do not automatically continue to the 'idle'
state, but
  // wait in 'start' until the next input message arrives. This event
enables us to
  // transition to 'idle' on the next valid 'temperatureInput' message
that arrives.
  "actions": [
    {
      "setVariable": {
        "variableName": "averageTemperature",
        "value": "(((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
      }
    }
  ],
  "nextState": "idle"
}
]
},
"onExit": {
  "events": [
    {
      "eventName": "resetHeatCool",
      "condition": "true",
      // Make sure the heating and cooling units are off before entering
'idle'.
      "actions": [
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
          }
        }
      ]
    }
  ]
}
}
}

```

```

    },
    {
      "sns": {
        "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0ff"
      }
    },
    {
      "iotTopicPublish": {
        "mqttTopic": "hvac/Heating/Off"
      }
    },
    {
      "iotTopicPublish": {
        "mqttTopic": "hvac/Cooling/Off"
      }
    }
  ]
}
],
},
},

{
  "stateName": "idle",
  "onInput": {
    "events": [
      {
        "eventName": "whatWasInput",
        "condition": "true",
        // By storing the 'sensorId' and the 'temperature' in variables, we make
them
        // available in any messages we send out to report anomalies, spikes,
or just
        // if needed for debugging.
        "actions": [
          {
            "setVariable": {
              "variableName": "sensorId",
              "value": "$input.temperatureInput.sensorId"
            }
          },
          {
            "setVariable": {

```

```

        "variableName": "reportedTemperature",
        "value": "$input.temperatureInput.sensorData.temperature"
    }
}
],
},
{
    "eventName": "changeDesired",
    "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
    // This event enables us to change the desired temperature at any time by
sending a
    // 'seedTemperatureInput' message. But note that other operational
parameters are not
    // read or changed.
    "actions": [
        {
            "setVariable": {
                "variableName": "desiredTemperature",
                "value": "$input.seedTemperatureInput.desiredTemperature"
            }
        }
    ]
},
{
    "eventName": "calculateAverage",
    "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
    // If a valid temperature reading arrives, we use it to update the
average temperature.
    // For simplicity, we assume our sensors will be sending updates at
about the same rate,
    // so we can calculate an approximate average by giving equal weight to
each reading we receive.
    "actions": [
        {
            "setVariable": {
                "variableName": "averageTemperature",
                "value": "((( $variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
            }
        }
    ]
}
]

```

```

    }
  ],
  "transitionEvents": [
    {
      "eventName": "anomalousInputArrived",
      "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
      // When an anomalous reading arrives, send an MQTT message, but stay in
the current state.
      "actions": [
        {
          "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/anomaly"
          }
        }
      ],
      "nextState": "idle"
    },
    {
      "eventName": "highTemperatureSpike",
      "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
      // When even a single temperature reading arrives that is above the
'rangeHigh', take
      // emergency action to begin cooling, and report a high temperature
spike.
      "actions": [
        {
          "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/spike"
          }
        },
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0n"
          }
        },
        {
          "iotTopicPublish": {
            "mqttTopic": "hvac/Cooling/On"
          }
        }
      ],
    },
  ],

```

```

        {
            "setVariable": {
                // This is necessary because we want to set a timer to delay the
shutoff
                //   of a cooling/heating unit, but we only want to set the timer
when we
                //   enter that new state initially.
                "variableName": "enteringNewState",
                "value": "true"
            }
        }
    ],
    "nextState": "cooling"
},

{
    "eventName": "lowTemperatureSpike",
    "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
    // When even a single temperature reading arrives that is below the
'rangeLow', take
    //   emergency action to begin heating, and report a low-temperature
spike.
    "actions": [
        {
            "iotTopicPublish": {
                "mqttTopic": "temperatureSensor/spike"
            }
        },
        {
            "sns": {
                "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
            }
        },
        {
            "iotTopicPublish": {
                "mqttTopic": "hvac/Heating/On"
            }
        },
        {
            "setVariable": {
                "variableName": "enteringNewState",
                "value": "true"
            }
        }
    ]
}

```

```

    }
  ],
  "nextState": "heating"
},

{
  "eventName": "highTemperatureThreshold",
  "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) >
($variable.desiredTemperature + $variable.allowedError))",
  // When the average temperature is above the desired temperature plus the
allowed error factor,
  // it is time to start cooling. Note that we calculate the average
temperature here again
  // because the value stored in the 'averageTemperature' variable is not
yet available for use
  // in our condition.
  "actions": [
    {
      "sns": {
        "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
      }
    },
    {
      "iotTopicPublish": {
        "mqttTopic": "hvac/Cooling/On"
      }
    },
    {
      "setVariable": {
        "variableName": "enteringNewState",
        "value": "true"
      }
    }
  ],
  "nextState": "cooling"
},

{
  "eventName": "lowTemperatureThreshold",
  "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) <
($variable.desiredTemperature - $variable.allowedError))",

```

```

        // When the average temperature is below the desired temperature minus
the allowed error factor,
        // it is time to start heating. Note that we calculate the average
temperature here again
        // because the value stored in the 'averageTemperature' variable is not
yet available for use
        // in our condition.
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
                }
            },
            {
                "iotTopicPublish": {
                    "mqttTopic": "hvac/Heating/On"
                }
            },
            {
                "setVariable": {
                    "variableName": "enteringNewState",
                    "value": "true"
                }
            }
        ],
        "nextState": "heating"
    }
]
}
},

{
    "stateName": "cooling",
    "onEnter": {
        "events": [
            {
                "eventName": "delay",
                "condition": "!$variable.noDelay && $variable.enteringNewState",
                // If the operational parameters specify that there should be a minimum
time that the
                // heating and cooling units should be run before being shut off again,
we set
                // a timer to ensure the proper operation here.
            }
        ]
    }
}

```

```

    "actions": [
      {
        "setTimer": {
          "timerName": "coolingTimer",
          "seconds": 180
        }
      },
      {
        "setVariable": {
          // We use this 'goodToGo' variable to store the status of the timer
expiration
          // for use in conditions that also use input variable values. If
lost.
          // 'timeout()' is used in such mixed conditionals, its value is

          "variableName": "goodToGo",
          "value": "false"
        }
      }
    ],
    {
      "eventName": "dontDelay",
      "condition": "$variable.noDelay == true",
      // If the heating/cooling unit shutoff delay is not used, no need to
wait.

      "actions": [
        {
          "setVariable": {
            "variableName": "goodToGo",
            "value": "true"
          }
        }
      ]
    },
    {
      "eventName": "beenHere",
      "condition": "true",
      "actions": [
        {
          "setVariable": {
            "variableName": "enteringNewState",
            "value": "false"
          }
        }
      ]
    }
  ]
}

```

```

    ]
  }
]
},

"onInput": {
  "events": [
    // These are events that occur when an input is received (if the condition
is
    // satisfied), but don't cause a transition to another state.
    {
      "eventName": "whatWasInput",
      "condition": "true",
      "actions": [
        {
          "setVariable": {
            "variableName": "sensorId",
            "value": "$input.temperatureInput.sensorId"
          }
        },
        {
          "setVariable": {
            "variableName": "reportedTemperature",
            "value": "$input.temperatureInput.sensorData.temperature"
          }
        }
      ]
    },
    {
      "eventName": "changeDesired",
      "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
      "actions": [
        {
          "setVariable": {
            "variableName": "desiredTemperature",
            "value": "$input.seedTemperatureInput.desiredTemperature"
          }
        }
      ]
    },
    {
      "eventName": "calculateAverage",

```

```

        "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
        "actions": [
            {
                "setVariable": {
                    "variableName": "averageTemperature",
                    "value": "(((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
                }
            }
        ],
        {
            "eventName": "areWeThereYet",
            "condition": "(timeout(\"coolingTimer\"))",
            "actions": [
                {
                    "setVariable": {
                        "variableName": "goodToGo",
                        "value": "true"
                    }
                }
            ]
        }
    ],
    "transitionEvents": [
        // Note that some tests of temperature values (for example, the test for an
anomalous value)
        // must be placed here in the 'transitionEvents' because they work
together with the tests
        // in the other conditions to ensure that we implement the proper
"if..elseif..else" logic.
        // But each transition event must have a destination state ('nextState'),
and even if that
        // is actually the current state, the "onEnter" events for this state
will be executed again.
        // This is the reason for the 'enteringNewState' variable and related.
        {
            "eventName": "anomalousInputArrived",
            "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
            "actions": [

```

```
        {
          "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/anomaly"
          }
        }
      ],
      "nextState": "cooling"
    },

    {
      "eventName": "highTemperatureSpike",
      "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
      "actions": [
        {
          "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/spike"
          }
        }
      ],
      "nextState": "cooling"
    },

    {
      "eventName": "lowTemperatureSpike",
      "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
      "actions": [
        {
          "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/spike"
          }
        },
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0ff"
          }
        },
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:heat0n"
          }
        }
      ],
    }
  ]
}
```

```

        "iotTopicPublish": {
            "mqttTopic": "hvac/Cooling/Off"
        }
    },
    {
        "iotTopicPublish": {
            "mqttTopic": "hvac/Heating/On"
        }
    },
    {
        "setVariable": {
            "variableName": "enteringNewState",
            "value": "true"
        }
    }
],
"nextState": "heating"
},
{
    "eventName": "desiredTemperature",
    "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) <=
($variable.desiredTemperature - $variable.allowedError)) && $variable.goodToGo ==
true",
    "actions": [
        {
            "sns": {
                "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0ff"
            }
        },
        {
            "iotTopicPublish": {
                "mqttTopic": "hvac/Cooling/Off"
            }
        }
    ],
    "nextState": "idle"
}
]
}
},

```

```
{
  "stateName": "heating",
  "onEnter": {
    "events": [
      {
        "eventName": "delay",
        "condition": "!$variable.noDelay && $variable.enteringNewState",
        "actions": [
          {
            "setTimer": {
              "timerName": "heatingTimer",
              "seconds": 120
            }
          },
          {
            "setVariable": {
              "variableName": "goodToGo",
              "value": "false"
            }
          }
        ]
      },
      {
        "eventName": "dontDelay",
        "condition": "$variable.noDelay == true",
        "actions": [
          {
            "setVariable": {
              "variableName": "goodToGo",
              "value": "true"
            }
          }
        ]
      },
      {
        "eventName": "beenHere",
        "condition": "true",
        "actions": [
          {
            "setVariable": {
              "variableName": "enteringNewState",
              "value": "false"
            }
          }
        ]
      }
    ]
  }
}
```

```

    ]
  }
]
},

"onInput": {
  "events": [
    {
      "eventName": "whatWasInput",
      "condition": "true",
      "actions": [
        {
          "setVariable": {
            "variableName": "sensorId",
            "value": "$input.temperatureInput.sensorId"
          }
        },
        {
          "setVariable": {
            "variableName": "reportedTemperature",
            "value": "$input.temperatureInput.sensorData.temperature"
          }
        }
      ]
    },
    {
      "eventName": "changeDesired",
      "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
      "actions": [
        {
          "setVariable": {
            "variableName": "desiredTemperature",
            "value": "$input.seedTemperatureInput.desiredTemperature"
          }
        }
      ]
    },
    {
      "eventName": "calculateAverage",
      "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
      "actions": [

```

```

        {
            "setVariable": {
                "variableName": "averageTemperature",
                "value": "(((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
            }
        }
    ],
    },
    {
        "eventName": "areWeThereYet",
        "condition": "(timeout(\"heatingTimer\"))",
        "actions": [
            {
                "setVariable": {
                    "variableName": "goodToGo",
                    "value": "true"
                }
            }
        ]
    }
},
"transitionEvents": [
    {
        "eventName": "anomalousInputArrived",
        "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
        "actions": [
            {
                "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/anomaly"
                }
            }
        ]
    },
    "nextState": "heating"
},
    {
        "eventName": "highTemperatureSpike",
        "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
        "actions": [
            {

```

```

        "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/spike"
        }
    },
    {
        "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
        }
    },
    {
        "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
        }
    },
    {
        "iotTopicPublish": {
            "mqttTopic": "hvac/Heating/Off"
        }
    },
    {
        "iotTopicPublish": {
            "mqttTopic": "hvac/Cooling/On"
        }
    },
    {
        "setVariable": {
            "variableName": "enteringNewState",
            "value": "true"
        }
    }
],
"nextState": "cooling"
},
{
    "eventName": "lowTemperatureSpike",
    "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
    "actions": [
        {
            "iotTopicPublish": {
                "mqttTopic": "temperatureSensor/spike"
            }
        }
    ]
}

```

```

    ],
    "nextState": "heating"
  },
  {
    "eventName": "desiredTemperature",
    "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) >=
($variable.desiredTemperature + $variable.allowedError)) && $variable.goodToGo ==
true",
    "actions": [
      {
        "sns": {
          "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
        }
      },
      {
        "iotTopicPublish": {
          "mqttTopic": "hvac/Heating/Off"
        }
      }
    ],
    "nextState": "idle"
  }
]
}
}
],
"initialStateName": "start"
},
"key": "areaId",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}

```

Respons:

```

{
  "detectorModelConfiguration": {
    "status": "ACTIVATING",
    "lastUpdateTime": 1557523491.168,
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",

```

```
    "creationTime": 1557523491.168,  
    "detectorModelArn": "arn:aws:iotevents:us-west-2:123456789012:detectorModel/  
areaDetectorModel",  
    "key": "areaId",  
    "detectorModelName": "areaDetectorModel",  
    "detectorModelVersion": "1"  
  }  
}
```

## Gunakan BatchUpdateDetector untuk memperbarui model AWS IoT Events detektor

Anda dapat menggunakan BatchUpdateDetector operasi untuk menempatkan instance detektor ke dalam keadaan yang diketahui, termasuk timer dan nilai variabel. Dalam contoh berikut, BatchUpdateDetector operasi mengatur ulang parameter operasional untuk area yang berada di bawah pemantauan dan kontrol suhu. Operasi ini memungkinkan Anda untuk melakukan ini tanpa harus menghapus, dan membuat ulang, atau memperbarui model detektor.

Perintah CLI:

```
aws iotevents-data batch-update-detector --cli-input-json file://areaDM.BUD.json
```

Berkas: areaDM.BUD.json

```
{  
  "detectors": [  
    {  
      "messageId": "0001",  
      "detectorModelName": "areaDetectorModel",  
      "keyValue": "Area51",  
      "state": {  
        "stateName": "start",  
        "variables": [  
          {  
            "name": "desiredTemperature",  
            "value": "22"  
          },  
          {  
            "name": "averageTemperature",  
            "value": "22"  
          }  
        ]  
      }  
    }  
  ]  
}
```

```
    },
    {
      "name": "allowedError",
      "value": "1.0"
    },
    {
      "name": "rangeHigh",
      "value": "30.0"
    },
    {
      "name": "rangeLow",
      "value": "15.0"
    },
    {
      "name": "anomalousHigh",
      "value": "60.0"
    },
    {
      "name": "anomalousLow",
      "value": "0.0"
    },
    {
      "name": "sensorCount",
      "value": "12"
    },
    {
      "name": "noDelay",
      "value": "true"
    },
    {
      "name": "goodToGo",
      "value": "true"
    },
    {
      "name": "sensorId",
      "value": "0"
    },
    {
      "name": "reportedTemperature",
      "value": "0.1"
    },
    {
      "name": "resetMe",
```

```

        // When 'resetMe' is true, our detector model knows that we have reentered
the 'start' state
        // to reset operational parameters, and will allow the next valid
temperature sensor
        // reading to cause the transition to the 'idle' state.
        "value": "true"
    }
    ],
    "timers": [
    ]
}
]
}

```

Respons:

```

{
  "batchUpdateDetectorErrorEntries": []
}

```

## Gunakan BatchPutMessage untuk input di AWS IoT Events

### Example 1

Gunakan BatchPutMessage operasi untuk mengirim "seedTemperatureInput" pesan yang menetapkan parameter operasional untuk area tertentu di bawah kontrol suhu dan pemantauan. Setiap pesan yang diterima oleh AWS IoT Events yang memiliki hal baru "areaId" menyebabkan instance detektor baru dibuat. Tetapi instance detektor baru tidak akan mengubah keadaan menjadi "idle" dan mulai memantau suhu dan mengendalikan unit pemanas atau pendingin sampai "seedTemperatureInput" pesan diterima untuk area baru.

Perintah CLI:

```

aws iotevents-data batch-put-message --cli-input-json file://seedExample.json --cli-
binary-format raw-in-base64-out

```

Berkas: seedExample.json

```

{

```

```
"messages": [  
  {  
    "messageId": "00001",  
    "inputName": "seedTemperatureInput",  
    "payload": "{\"areaId\": \"Area51\", \"desiredTemperature\": 20.0, \"allowedError  
\": 0.7, \"rangeHigh\": 30.0, \"rangeLow\": 15.0, \"anomalousHigh\": 60.0,  
\"anomalousLow\": 0.0, \"sensorCount\": 10, \"noDelay\": false}"  
  }  
]
```

Respons:

```
{  
  "BatchPutMessageErrorEntries": []  
}
```

Example

2

Gunakan BatchPutMessage operasi untuk mengirim "temperatureInput" pesan untuk melaporkan data sensor suhu untuk sensor di area kontrol dan pemantauan tertentu.

Perintah CLI:

```
aws iotevents-data batch-put-message --cli-input-json file://temperatureExample.json --  
cli-binary-format raw-in-base64-out
```

Berkas: temperatureExample.json

```
{  
  "messages": [  
    {  
      "messageId": "00005",  
      "inputName": "temperatureInput",  
      "payload": "{\"sensorId\": \"05\", \"areaId\": \"Area51\", \"sensorData\":  
{\"temperature\": 23.12} }"  
    }  
  ]  
}
```

**Respons:**

```
{
  "BatchPutMessageErrorEntries": []
}
```

**Example 3**

Gunakan BatchPutMessage operasi untuk mengirim "seedTemperatureInput" pesan untuk mengubah nilai suhu yang diinginkan untuk area tertentu.

**Perintah CLI:**

```
aws iotevents-data batch-put-message --cli-input-json file://seedSetDesiredTemp.json --cli-binary-format raw-in-base64-out
```

**Berkas: seedSetDesiredTemp.json**

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "seedTemperatureInput",
      "payload": "{\"areaId\": \"Area51\", \"desiredTemperature\": 23.0}"
    }
  ]
}
```

**Respons:**

```
{
  "BatchPutMessageErrorEntries": []
}
```

## Menelan pesan MQTT di AWS IoT Events

Jika sumber daya komputasi sensor Anda tidak dapat menggunakan "BatchPutMessage" API, tetapi dapat mengirim datanya ke broker AWS IoT Core pesan menggunakan klien MQTT ringan, Anda dapat membuat aturan AWS IoT Core topik untuk mengarahkan data pesan ke input. AWS IoT Events Berikut ini adalah definisi aturan AWS IoT Events topik yang

mengambil "areaId" dan "sensorId" memasukkan bidang dari topik MQTT, dan bidang dari "sensorData.temperature" bidang payload "temp" pesan, dan memasukkan data ini ke dalam kami. AWS IoT Events "temperatureInput"

Perintah CLI:

```
aws iot create-topic-rule --cli-input-json file://temperatureTopicRule.json
```

Berkas: seedSetDesiredTemp.json

```
{
  "ruleName": "temperatureTopicRule",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as areaId, topic(4) as sensorId, temp as
sensorData.temperature FROM 'update/temperature/#'",
    "description": "Ingest temperature sensor messages into IoT Events",
    "actions": [
      {
        "iotEvents": {
          "inputName": "temperatureInput",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/anotherRole"
        }
      }
    ],
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23"
  }
}
```

Tanggapan: [tidak ada]

Jika sensor mengirim pesan pada topik "update/temperature/Area51/03" dengan payload berikut.

```
{ "temp": 24.5 }
```

Ini menghasilkan data yang dicerna AWS IoT Events seolah-olah panggilan "BatchPutMessage" API berikut telah dilakukan.

```
aws iotevents-data batch-put-message --cli-input-json file://spooferExample.json --cli-binary-format raw-in-base64-out
```

Berkas: spooferExample.json

```
{
  "messages": [
    {
      "messageId": "54321",
      "inputName": "temperatureInput",
      "payload": "{\"sensorId\": \"03\", \"areaId\": \"Area51\", \"sensorData\": {\"temperature\": 24.5} }"
    }
  ]
}
```

## Hasilkan pesan Amazon SNS di AWS IoT Events

Berikut ini adalah contoh pesan SNS yang dihasilkan oleh instance "Area51" detektor.

AWS IoT Events dapat berintegrasi dengan Amazon SNS untuk menghasilkan dan mempublikasikan notifikasi berdasarkan peristiwa yang terdeteksi. Bagian ini menunjukkan bagaimana instance AWS IoT Events detektor, khususnya detektor "Area51", menghasilkan pesan Amazon SNS. Contoh-contoh ini menampilkan struktur dan konten notifikasi Amazon SNS yang dipicu oleh berbagai status dan peristiwa AWS IoT Events di dalam detektor, menggambarkan kekuatan penggabungan dengan AWS IoT Events Amazon SNS untuk peringatan dan komunikasi waktu nyata.

```
Heating system off command> {
  "eventTime":1557520274729,
  "payload":{
    "actionExecutionId":"f3159081-bac3-38a4-96f7-74af0940d0a4",
    "detector":{

      "detectorModelName":"areaDetectorModel","keyValue":"Area51","detectorModelVersion":"1"}, "event
{"inputName":"seedTemperatureInput","messageId":"00001","triggerType":"Message"},"state":
{"stateName":"start","variables":
{"sensorCount":10,"rangeHigh":30.0,"resetMe":false,"enteringNewState":true,"averageTemperature"
{}}},"eventName":"resetHeatCool"}
```

```
Cooling system off command> {"eventTime":1557520274729,"payload":
{"actionExecutionId":"98f6a1b5-8f40-3cdb-9256-93afd4d66192","detector":
{"detectorModelName":"areaDetectorModel","keyValue":"Area51","detectorModelVersion":"1"},"event
{"inputName":"seedTemperatureInput","messageId":"00001","triggerType":"Message"},"state":
{"stateName":"start","variables":
{"sensorCount":10,"rangeHigh":30.0,"resetMe":false,"enteringNewState":true,"averageTemperature"
{}}},"eventName":"resetHeatCool"}
```

## Konfigurasi DescribeDetector API di AWS IoT Events

`DescribeDetectorAPI` di dalamnya AWS IoT Events memungkinkan Anda untuk mengambil informasi rinci tentang instance detektor tertentu. Operasi ini memberikan wawasan tentang keadaan saat ini, nilai variabel, dan pengatur waktu aktif detektor. Dengan menggunakan API ini, Anda dapat memantau status real-time AWS IoT Events detektor Anda, memfasilitasi debugging, analisis, dan pengelolaan alur kerja pemrosesan peristiwa IoT Anda.

Perintah CLI:

```
aws iotevents-data describe-detector --detector-model-name areaDetectorModel --key-
value Area51
```

Respons:

```
{
  "detector": {
    "lastUpdateTime": 1557521572.216,
    "creationTime": 1557520274.405,
    "state": {
      "variables": [
        {
          "name": "resetMe",
          "value": "false"
        },
        {
          "name": "rangeLow",
          "value": "15.0"
        },
        {
          "name": "noDelay",
          "value": "false"
        }
      ]
    }
  }
}
```

```
{
  "name": "desiredTemperature",
  "value": "20.0"
},
{
  "name": "anomalousLow",
  "value": "0.0"
},
{
  "name": "sensorId",
  "value": "\"01\""
},
{
  "name": "sensorCount",
  "value": "10"
},
{
  "name": "rangeHigh",
  "value": "30.0"
},
{
  "name": "enteringNewState",
  "value": "false"
},
{
  "name": "averageTemperature",
  "value": "19.572"
},
{
  "name": "allowedError",
  "value": "0.7"
},
{
  "name": "anomalousHigh",
  "value": "60.0"
},
{
  "name": "reportedTemperature",
  "value": "15.72"
},
{
  "name": "goodToGo",
  "value": "false"
}
}
```

```

    ],
    "stateName": "idle",
    "timers": [
      {
        "timestamp": 1557520454.0,
        "name": "idleTimer"
      }
    ]
  },
  "keyValue": "Area51",
  "detectorModelName": "areaDetectorModel",
  "detectorModelVersion": "1"
}
}

```

## Gunakan mesin AWS IoT Core aturan untuk AWS IoT Events

Aturan berikut menerbitkan ulang pesan AWS IoT Core MQTT sebagai pesan permintaan pembaruan bayangan. Kami berasumsi bahwa AWS IoT Core hal-hal didefinisikan untuk unit pemanas dan unit pendingin untuk setiap area yang dikendalikan oleh model detektor. Dalam contoh ini, kita telah mendefinisikan hal-hal bernama "Area51HeatingUnit" dan "Area51CoolingUnit".

Perintah CLI:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowCoolOffRule.json
```

Berkas: ADMShadowCoolOffRule.json

```

{
  "ruleName": "ADMShadowCoolOff",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Cooling/Off'",
    "description": "areaDetectorModel mqtt topic publish to cooling unit shadow request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republiish": {
          "topic": "$$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/update",

```

```
        "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMSHadowRole"
    }
}
]
```

Tanggapan: [kosong]

Perintah CLI:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowCoolOnRule.json
```

Berkas: ADMSHadowCoolOnRule.json

```
{
  "ruleName": "ADMSHadowCoolOn",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Cooling/On'",
    "description": "areaDetectorModel mqtt topic publish to cooling unit shadow
request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMSHadowRole"
        }
      }
    ]
  }
}
```

Tanggapan: [kosong]

Perintah CLI:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowHeatOffRule.json
```

**Berkas: ADMSHadowHeatOffRule.json**

```
{
  "ruleName": "ADMSHadowHeatOff",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Heating/Off'",
    "description": "areaDetectorModel mqtt topic publish to heating unit shadow
request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republsh": {
          "topic": "$$aws/things/${payload.detector.keyValue}HeatingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMSHadowRole"
        }
      }
    ]
  }
}
```

Tanggapan: [kosong]

Perintah CLI:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowHeatOnRule.json
```

**Berkas: ADMSHadowHeatOnRule.json**

```
{
  "ruleName": "ADMSHadowHeatOn",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Heating/On'",
    "description": "areaDetectorModel mqtt topic publish to heating unit shadow
request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republsh": {
```

```
        "topic": "$$aws/things/${payload.detector.keyValue}HeatingUnit/shadow/  
update",  
        "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"  
    }  
  }  
]  
}  
}
```

Tanggapan: [kosong]

# Tindakan yang didukung untuk menerima data dan memicu tindakan di AWS IoT Events

AWS IoT Events dapat memicu tindakan ketika mendeteksi peristiwa tertentu atau peristiwa transisi. Anda dapat menentukan tindakan bawaan untuk menggunakan timer atau menyetel variabel, atau mengirim data ke AWS sumber daya lain. Pelajari cara mengonfigurasi dan menyesuaikan tindakan ini untuk membuat respons otomatis terhadap berbagai peristiwa IoT Anda.

## Note

Saat Anda menentukan tindakan dalam model detektor, Anda dapat menggunakan ekspresi untuk parameter yang merupakan tipe data string. Untuk informasi selengkapnya, lihat [Ekspresi](#).

AWS IoT Events mendukung tindakan berikut yang memungkinkan Anda menggunakan timer atau mengatur variabel:

- [setTimeout](#) untuk membuat timer.
- [resetTimer](#) untuk mengatur ulang timer.
- [clearTimer](#) untuk menghapus timer.
- [setVariable](#) untuk membuat variabel.

AWS IoT Events mendukung tindakan berikut yang memungkinkan Anda bekerja dengan AWS layanan:

- [iotTopicPublish](#) untuk mempublikasikan pesan tentang topik MQTT.
- [iotEvents](#) untuk mengirim data ke AWS IoT Events sebagai nilai input.
- [iotSiteWise](#) untuk mengirim data ke properti aset di AWS IoT SiteWise.
- [dynamoDB](#) untuk mengirim data ke tabel Amazon DynamoDB.
- [dynamoDBv2](#) untuk mengirim data ke tabel Amazon DynamoDB.
- [firehose](#) untuk mengirim data ke aliran Amazon Data Firehose.
- [lambda](#) untuk memanggil AWS Lambda fungsi.

- [sns](#) untuk mengirim data sebagai pemberitahuan push.
- [sqs](#) untuk mengirim data ke antrian Amazon SQS.

## Gunakan timer AWS IoT Events bawaan dan tindakan variabel

AWS IoT Events mendukung tindakan berikut yang memungkinkan Anda menggunakan timer atau mengatur variabel:

- [setTimer](#) untuk membuat timer.
- [resetTimer](#) untuk mengatur ulang timer.
- [clearTimer](#) untuk menghapus timer.
- [setVariable](#) untuk membuat variabel.

### Atur tindakan pengatur waktu

#### Set timer action

`setTimer` Tindakan ini memungkinkan Anda membuat timer dengan durasi dalam hitungan detik.

#### More information (2)

Saat Anda membuat timer, Anda harus menentukan parameter berikut.

##### **timerName**

Nama timer.

##### **durationExpression**

(Opsional) Durasi timer, dalam hitungan detik.

Hasil evaluasi dari ekspresi durasi dibulatkan ke bawah ke bilangan bulat terdekat. Misalnya, jika Anda mengatur timer ke 60,99 detik, hasil evaluasi dari ekspresi durasi adalah 60 detik.

Untuk informasi selengkapnya, lihat [SetTimerAction](#) di dalam Referensi API AWS IoT Events .

## Atur ulang tindakan pengatur waktu

### Reset timer action

`resetTimer`Tindakan ini memungkinkan Anda menyetel timer ke hasil ekspresi durasi yang dievaluasi sebelumnya.

#### More information (1)

Saat Anda mengatur ulang timer, Anda harus menentukan parameter berikut.

#### **timerName**

Nama timer.

AWS IoT Events tidak mengevaluasi kembali ekspresi durasi saat Anda mengatur ulang pengatur waktu.

Untuk informasi selengkapnya, lihat [ResetTimerAction](#) di dalam Referensi API AWS IoT Events .

## Hapus tindakan pengatur waktu

### Clear timer action

`clearTimer`Tindakan ini memungkinkan Anda menghapus timer yang ada.

#### More information (1)

Saat Anda menghapus timer, Anda harus menentukan parameter berikut.

#### **timerName**

Nama timer.

Untuk informasi selengkapnya, lihat [ClearTimerAction](#) di dalam Referensi API AWS IoT Events .

## Tetapkan tindakan variabel

### Set variable action

`setVariable`Tindakan ini memungkinkan Anda membuat variabel dengan nilai tertentu.

## More information (2)

Saat Anda membuat variabel, Anda harus menentukan parameter berikut.

### **variableName**

Nama dari variabel.

### **value**

Nilai baru dari variabel.

Untuk informasi selengkapnya, lihat [SetVariableAction](#) di dalam Referensi API AWS IoT Events .

## AWS IoT Events bekerja dengan AWS layanan lain

AWS IoT Events mendukung tindakan berikut yang memungkinkan Anda bekerja dengan AWS layanan:

- [iotTopicPublish](#) untuk mempublikasikan pesan tentang topik MQTT.
- [iotEvents](#) untuk mengirim data ke AWS IoT Events sebagai nilai input.
- [iotSiteWise](#) untuk mengirim data ke properti aset di AWS IoT SiteWise.
- [dynamoDB](#) untuk mengirim data ke tabel Amazon DynamoDB.
- [dynamoDBv2](#) untuk mengirim data ke tabel Amazon DynamoDB.
- [firehose](#) untuk mengirim data ke aliran Amazon Data Firehose.
- [lambda](#) untuk memanggil AWS Lambda fungsi.
- [sns](#) untuk mengirim data sebagai pemberitahuan push.
- [sqs](#) untuk mengirim data ke antrian Amazon SQS.

### Important

- Anda harus memilih AWS Wilayah yang sama untuk keduanya AWS IoT Events dan AWS layanan yang akan digunakan. Untuk daftar Wilayah yang didukung, lihat [AWS IoT Events titik akhir dan kuota](#) di. Referensi Umum Amazon Web Services

- Anda harus menggunakan AWS Wilayah yang sama saat membuat AWS sumber daya lain untuk AWS IoT Events tindakan tersebut. Jika Anda beralih AWS Wilayah, Anda mungkin mengalami masalah saat mengakses AWS sumber daya.

Secara default, AWS IoT Events menghasilkan muatan standar di JSON untuk tindakan apa pun. Payload tindakan ini berisi semua pasangan nilai atribut yang memiliki informasi tentang instance model detektor dan peristiwa yang memicu aksi. Untuk mengonfigurasi payload tindakan, Anda dapat menggunakan ekspresi konten. Untuk informasi selengkapnya, lihat [Ekspresi untuk memfilter, mengubah, dan memproses data peristiwa](#) dan tipe data [Payload](#) di Referensi AWS IoT Events API.

## AWS IoT Core

### IoT topic publish action

AWS IoT Core Tindakan ini memungkinkan Anda mempublikasikan pesan MQTT melalui broker pesan. AWS IoT Untuk daftar Wilayah yang didukung, lihat [AWS IoT Core titik akhir dan kuota](#) di Referensi Umum Amazon Web Services

Broker AWS IoT pesan menghubungkan AWS IoT klien dengan mengirim pesan dari klien penerbitan ke klien berlangganan. Untuk informasi selengkapnya, lihat [Protokol komunikasi perangkat](#) di Panduan AWS IoT Pengembang.

### More information (2)

Ketika Anda mempublikasikan pesan MQTT, Anda harus menentukan parameter berikut.

#### **mqttTopic**

Topik MQTT yang menerima pesan.

Anda dapat menentukan nama topik MQTT secara dinamis saat runtime menggunakan variabel atau nilai input yang dibuat dalam model detektor.

#### **payload**

(Opsional) Payload default berisi semua pasangan nilai atribut yang memiliki informasi tentang contoh model detektor dan peristiwa memicu tindakan. Anda juga dapat menyesuaikan payload. Untuk informasi selengkapnya, lihat [Payload](#) di Referensi AWS IoT Events API.

**Note**

Pastikan bahwa kebijakan yang dilampirkan pada peran AWS IoT Events layanan Anda memberikan `iot:Publish` izin. Untuk informasi selengkapnya, lihat [Identitas dan manajemen akses untuk AWS IoT Events](#).

Untuk informasi selengkapnya, lihat [IoTTopicPublishAction](#) di dalam Referensi API AWS IoT Events .

## AWS IoT Events

### IoT Events action

AWS IoT Events Tindakan ini memungkinkan Anda mengirim data AWS IoT Events sebagai input. Untuk daftar Wilayah yang didukung, lihat [AWS IoT Events titik akhir dan kuota](#) di. Referensi Umum Amazon Web Services

AWS IoT Events memungkinkan Anda untuk memantau peralatan atau armada perangkat Anda untuk kegagalan atau perubahan dalam operasi, dan untuk memicu tindakan ketika peristiwa tersebut terjadi. Untuk informasi lebih lanjut, lihat [Apa itu AWS IoT Events?](#) di Panduan AWS IoT Events Pengembang.

### More information (2)


Saat Anda mengirim data ke AWS IoT Events, Anda harus menentukan parameter berikut.

**inputName**

Nama AWS IoT Events input yang menerima data.

**payload**

(Opsional) Payload default berisi semua pasangan nilai atribut yang memiliki informasi tentang contoh model detektor dan peristiwa memicu tindakan. Anda juga dapat menyesuaikan payload. Untuk informasi selengkapnya, lihat [Payload](#) di Referensi AWS IoT Events API.

 Note

Pastikan bahwa kebijakan yang dilampirkan pada peran AWS IoT Events layanan Anda memberikan `iotevents:BatchPutMessage` izin. Untuk informasi selengkapnya, lihat [Identitas dan manajemen akses untuk AWS IoT Events](#).

Untuk informasi selengkapnya, lihat [lotEventsAction](#) di dalam Referensi API AWS IoT Events .

## AWS IoT SiteWise


### IoT SiteWise action

AWS IoT SiteWise Tindakan ini memungkinkan Anda mengirim data ke properti aset di AWS IoT SiteWise. Untuk daftar Wilayah yang didukung, lihat [AWS IoT SiteWise titik akhir dan kuota](#) di Referensi Umum Amazon Web Services

AWS IoT SiteWise adalah layanan terkelola yang memungkinkan Anda mengumpulkan, mengatur, dan menganalisis data dari peralatan industri dalam skala besar. Untuk informasi selengkapnya, lihat [Apa itu AWS IoT SiteWise?](#) dalam Panduan Pengguna AWS IoT SiteWise .

### More information (11)

Ketika Anda mengirim data ke properti aset di AWS IoT SiteWise, Anda harus menentukan parameter berikut.

 Important

Untuk menerima data, Anda harus menggunakan properti aset yang ada di AWS IoT SiteWise.

- Jika Anda menggunakan AWS IoT Events konsol, Anda harus menentukan `propertyAlias` untuk mengidentifikasi properti aset target.
- Jika Anda menggunakan AWS CLI, Anda harus menentukan salah satu `propertyAlias` atau keduanya `assetId` dan `propertyId` untuk mengidentifikasi properti aset target.

Untuk informasi selengkapnya, lihat [Memetakan pengaliran data industri ke properti aset](#) di AWS IoT SiteWise Panduan Pengguna.

**propertyAlias**

(Opsional) Alias properti aset. Anda juga dapat menentukan ekspresi.

**assetId**

(Opsional) ID aset yang memiliki properti yang ditentukan. Anda juga dapat menentukan ekspresi.

**propertyId**

(Opsional) ID properti aset. Anda juga dapat menentukan ekspresi.

**entryId**

(Opsional) Pengidentifikasi unik untuk entri ini. Anda dapat menggunakan ID entri untuk melacak entri data yang menyebabkan kesalahan jika terjadi kegagalan. Default-nya adalah pengidentifikasi unik baru. Anda juga dapat menentukan ekspresi.

**propertyValue**

Struktur yang berisi rincian tentang nilai properti.

**quality**

(Opsional) Kualitas nilai properti aset. Nilai harus berupa GOOD, BAD, atau UNCERTAIN. Anda juga dapat menentukan ekspresi.

**timestamp**

(Opsional) Struktur yang berisi informasi stempel waktu. Jika Anda tidak menentukan nilai ini, defaultnya adalah waktu acara.

**timeInSeconds**

Timestamp, dalam hitungan detik, dalam format jangka waktu Unix. Kisaran valid adalah antara 1-31556889864403199. Anda juga dapat menentukan ekspresi.

**offsetInNanos**

(Opsional) Offset nanodetik dikonversi dari `timeInSeconds`. Kisaran valid adalah antara 0-999999999. Anda juga dapat menentukan ekspresi.

## value

Struktur yang berisi nilai properti aset.

### Important

Anda harus menentukan salah satu dari jenis nilai berikut, tergantung dari `dataType` dari properti aset yang ditentukan. Untuk informasi selengkapnya, lihat [AssetProperty](#) di dalam Referensi API AWS IoT SiteWise .

### **booleanValue**

(Opsional) Nilai properti aset adalah nilai Boolean yang harus TRUE atau FALSE. Anda juga dapat menentukan ekspresi. Jika Anda menggunakan ekspresi, hasil yang dievaluasi harus berupa nilai Boolean.

### **doubleValue**

(Opsional) Nilai properti aset adalah ganda. Anda juga dapat menentukan ekspresi. Jika Anda menggunakan ekspresi, hasil yang dievaluasi harus ganda.

### **integerValue**

(Opsional) Nilai properti aset adalah bilangan bulat. Anda juga dapat menentukan ekspresi. Jika Anda menggunakan ekspresi, hasil yang dievaluasi harus berupa bilangan bulat.

### **stringValue**

(Opsional) Nilai properti aset adalah string. Anda juga dapat menentukan ekspresi. Jika Anda menggunakan ekspresi, hasil yang dievaluasi harus berupa string.

### Note

Pastikan bahwa kebijakan yang dilampirkan pada peran AWS IoT Events layanan Anda memberikan `iotsitewise:BatchPutAssetPropertyValue` izin. Untuk informasi selengkapnya, lihat [Identitas dan manajemen akses untuk AWS IoT Events](#).

Untuk informasi selengkapnya, lihat [lotSiteWiseAction](#) di dalam Referensi API AWS IoT Events .

# Amazon DynamoDB

## DynamoDB action

Tindakan Amazon DynamoDB memungkinkan Anda mengirim data ke tabel DynamoDB. Satu kolom tabel DynamoDB menerima semua pasangan atribut-nilai dalam muatan tindakan yang Anda tentukan. Untuk daftar Wilayah yang didukung, lihat [titik akhir Amazon DynamoDB](#) dan kuota di. Referensi Umum Amazon Web Services

Amazon DynamoDB adalah layanan basis data NoSQL terkelola sepenuhnya yang menyediakan performa cepat dan dapat diprediksi dengan skalabilitas tanpa hambatan. Untuk informasi lebih lanjut, lihat [Apa itu DynamoDB?](#) di Panduan Pengembang Amazon DynamoDB.

## More information (10)

Ketika Anda mengirim data ke satu kolom tabel DynamoDB, Anda harus menentukan parameter berikut.

### **tableName**

Nama tabel DynamoDB yang menerima data. `tableName`Nilai harus sesuai dengan nama tabel DynamoDB tabel. Anda juga dapat menentukan ekspresi.

### **hashKeyField**

Nama kunci hash (juga disebut kunci partisi). `hashKeyField`Nilai harus cocok dengan kunci partisi dari tabel DynamoDB. Anda juga dapat menentukan ekspresi.

### **hashKeyType**

(Opsional) Tipe data dari kunci hash. Nilai dari tipe kunci hash harus `STRING` atau `NUMBER`. Nilai default-nya `STRING`. Anda juga dapat menentukan ekspresi.

### **hashKeyValue**

Nilai kunci hash. `hashKeyValue`Menggunakan template substitusi. Templat ini menyediakan data pada saat runtime. Anda juga dapat menentukan ekspresi.

### **rangeKeyField**

(Opsional) Nama tombol rentang (juga disebut tombol sortir). `rangeKeyField`Nilai harus cocok dengan kunci sort dari tabel DynamoDB. Anda juga dapat menentukan ekspresi.

## **rangeKeyType**

(Opsional) Tipe data dari tombol rentang. Nilai dari tipe kunci hash harus `STRING` atau `NUMBER`. Nilai default-nya `STRING`. Anda juga dapat menentukan ekspresi.

## **rangeKeyValue**

(Opsional) Nilai tombol rentang. `rangeKeyValue` menggunakan template substitusi. Templat ini menyediakan data pada saat runtime. Anda juga dapat menentukan ekspresi.

## **operation**

(Opsional) Jenis operasi yang harus dilakukan. Anda juga dapat menentukan ekspresi. Nilai operasi harus salah satu dari nilai berikut:

- `INSERT` - Masukkan data sebagai item baru ke dalam tabel DynamoDB. Ini adalah nilai default.
- `UPDATE` - Perbarui item tabel DynamoDB yang sudah ada dengan data baru.
- `DELETE` - Hapus item yang ada dari tabel DynamoDB.

## **payloadField**

(Opsional) Nama kolom DynamoDB yang menerima muatan tindakan. Nama defaultnya adalah `payload`. Anda juga dapat menentukan ekspresi.

## **payload**

(Opsional) Payload default berisi semua pasangan nilai atribut yang memiliki informasi tentang contoh model detektor dan peristiwa memicu tindakan. Anda juga dapat menyesuaikan payload. Untuk informasi selengkapnya, lihat [Payload](#) di Referensi AWS IoT Events API.

Jika jenis payload yang ditentukan adalah string, `DynamoDBAction` mengirimkan data non-JSON ke tabel DynamoDB sebagai data biner. Konsol DynamoDB menampilkan data sebagai teks. Base64-encoded Nilai `payloadField` adalah `payload-field_raw`. Anda juga dapat menentukan ekspresi.

### Note

Pastikan bahwa kebijakan yang dilampirkan pada peran AWS IoT Events layanan Anda memberikan `dynamodb:PutItem` izin. Untuk informasi selengkapnya, lihat [Identitas dan manajemen akses untuk AWS IoT Events](#).

Untuk informasi selengkapnya, lihat [DynamoDBAction](#) di Referensi API.AWS IoT Events

## Amazon DynamoDB (v2)

### DynamoDBv2 action

Tindakan Amazon DynamoDB (v2) memungkinkan Anda menulis data ke tabel DynamoDB. Kolom terpisah dari tabel DynamoDB menerima satu pasangan atribut-nilai dalam muatan tindakan yang Anda tentukan. Untuk daftar Wilayah yang didukung, lihat [titik akhir Amazon DynamoDB](#) dan kuota di. Referensi Umum Amazon Web Services

Amazon DynamoDB adalah layanan basis data NoSQL terkelola sepenuhnya yang menyediakan performa cepat dan dapat diprediksi dengan skalabilitas tanpa hambatan. Untuk informasi lebih lanjut, lihat [Apa itu DynamoDB?](#) di Panduan Pengembang Amazon DynamoDB.

### More information (2)

Ketika Anda mengirim data ke beberapa kolom tabel DynamoDB, Anda harus menentukan parameter berikut.

#### **tableName**

Nama tabel DynamoDB yang menerima data. Anda juga dapat menentukan ekspresi.

#### **payload**

(Opsional) Payload default berisi semua pasangan nilai atribut yang memiliki informasi tentang contoh model detektor dan peristiwa memicu tindakan. Anda juga dapat menyesuaikan payload. Untuk informasi selengkapnya, lihat [Payload](#) di Referensi AWS IoT Events API.

#### Important

Jenis payload harus JSON. Anda juga dapat menentukan ekspresi.

#### Note

Pastikan bahwa kebijakan yang dilampirkan pada peran AWS IoT Events layanan Anda memberikan `dynamodb:PutItem` izin. Untuk informasi selengkapnya, lihat [Identitas dan manajemen akses untuk AWS IoT Events](#).

Untuk informasi selengkapnya, lihat [DynamoDBv2Action](#) di dalam Referensi API AWS IoT Events

## Amazon Data Firehose

### Firehose action

Tindakan Amazon Data Firehose memungkinkan Anda mengirim data ke aliran pengiriman Firehose. Untuk daftar Wilayah yang didukung, lihat [titik akhir Amazon Data Firehose dan kuota](#) di bagian. Referensi Umum Amazon Web Services

Amazon Data Firehose adalah layanan yang dikelola sepenuhnya untuk mengirimkan data streaming real-time ke tujuan seperti Amazon Simple Storage Service (Amazon Simple Storage Service), Amazon Redshift, OpenSearch Amazon OpenSearch Service (Service), dan Splunk. Untuk informasi selengkapnya, lihat [Apa itu Amazon Data Firehose?](#) di Panduan Pengembang Firehose Data Amazon.

### More information (3)

Saat Anda mengirim data ke aliran pengiriman Firehose, Anda harus menentukan parameter berikut.

#### **deliveryStreamName**

Nama aliran pengiriman Firehose yang menerima data.

#### **separator**

(Opsional) Anda dapat menggunakan pemisah karakter untuk memisahkan data kontinu yang dikirim ke aliran pengiriman Firehose. Nilai pemisah harus `'\n'` (baris baru), `'\t'` (tab), `'\r\n'` (baris baru Windows), atau `' , '` (koma).

#### **payload**

(Opsional) Payload default berisi semua pasangan nilai atribut yang memiliki informasi tentang contoh model detektor dan peristiwa memicu tindakan. Anda juga dapat menyesuaikan payload. Untuk informasi selengkapnya, lihat [Payload](#) di Referensi AWS IoT Events API.

**Note**

Pastikan bahwa kebijakan yang dilampirkan pada peran AWS IoT Events layanan Anda memberikan `firehose:PutRecord` izin. Untuk informasi selengkapnya, lihat [Identitas dan manajemen akses untuk AWS IoT Events](#).

Untuk informasi selengkapnya, lihat [FirehoseAction](#) di dalam Referensi API AWS IoT Events .

## AWS Lambda

### Lambda action

AWS Lambda Tindakan ini memungkinkan Anda memanggil fungsi Lambda. Untuk daftar Wilayah yang didukung, lihat [AWS Lambda titik akhir dan kuota](#) di Referensi Umum Amazon Web Services

AWS Lambda adalah layanan komputasi yang memungkinkan Anda menjalankan kode tanpa menyediakan atau mengelola server. Untuk informasi lebih lanjut, lihat [Apa itu AWS Lambda?](#) di Panduan AWS Lambda Pengembang.

### More information (2)

Saat Anda memanggil fungsi Lambda, Anda harus menentukan parameter berikut.

#### **functionArn**

ARN dari fungsi Lambda untuk memanggil.

#### **payload**

(Opsional) Payload default berisi semua pasangan nilai atribut yang memiliki informasi tentang contoh model detektor dan peristiwa memicu tindakan. Anda juga dapat menyesuaikan payload. Untuk informasi selengkapnya, lihat [Payload](#) di Referensi AWS IoT Events API.

**Note**

Pastikan bahwa kebijakan yang dilampirkan pada peran AWS IoT Events layanan Anda memberikan `lambda:InvokeFunction` izin. Untuk informasi selengkapnya, lihat [Identitas dan manajemen akses untuk AWS IoT Events](#).

Untuk informasi selengkapnya, lihat [LambdaAction](#) di dalam Referensi API AWS IoT Events .

## Layanan Notifikasi Sederhana Amazon

### SNS action

Tindakan mempublikasikan topik Amazon SNS memungkinkan Anda mempublikasikan pesan Amazon SNS. Untuk daftar Wilayah yang didukung, lihat [titik akhir dan kuota Amazon Simple Notification Service](#) di. Referensi Umum Amazon Web Services

Amazon Simple Notification Service (Amazon Simple Notification Service) adalah layanan web yang mengoordinasikan dan mengelola pengiriman atau pengiriman pesan ke titik akhir atau klien berlangganan. Untuk informasi lebih lanjut, lihat [Apa itu Amazon SNS](#) di Panduan Developer Amazon Simple Notification Service.

#### Note

Tindakan publikasi topik Amazon SNS tidak mendukung topik Amazon SNS FIFO (masuk pertama, keluar pertama). Karena mesin aturan adalah layanan terdistribusi penuh, pesan mungkin tidak ditampilkan dalam urutan tertentu saat tindakan Amazon SNS dimulai.

### More information (2)

Saat mempublikasikan pesan Amazon SNS, Anda harus menentukan parameter berikut.

#### **targetArn**

ARN dari target Amazon SNS yang menerima pesan.

#### **payload**

(Opsional) Payload default berisi semua pasangan nilai atribut yang memiliki informasi tentang contoh model detektor dan peristiwa memicu tindakan. Anda juga dapat menyesuaikan payload. Untuk informasi selengkapnya, lihat [Payload](#) di Referensi AWS IoT Events API.

**Note**

Pastikan bahwa kebijakan yang dilampirkan pada peran AWS IoT Events layanan Anda memberikan `sns:Publish` izin. Untuk informasi selengkapnya, lihat [Identitas dan manajemen akses untuk AWS IoT Events](#).

Untuk informasi selengkapnya, lihat [SNSTopicPublishAction](#) di dalam Referensi API AWS IoT Events .

## Amazon Simple Queue Service

### SQS action

Tindakan Amazon SQS memungkinkan Anda mengirim data ke antrian Amazon SQS. Untuk daftar Wilayah yang didukung, lihat [titik akhir dan kuota Layanan Antrian Sederhana Amazon](#) di Referensi Umum Amazon Web Services

Amazon Simple Queue Service (Amazon SQS) menawarkan antrian host yang aman, tahan lama, dan tersedia yang memungkinkan Anda mengintegrasikan dan memisahkan sistem dan komponen perangkat lunak terdistribusi. Untuk informasi selengkapnya, lihat [Apa itu Layanan Antrian Sederhana Amazon](#) > di [Panduan Pengembang Layanan Antrian Sederhana Amazon](#).

**Note**

Tindakan Amazon SQS tidak mendukung topik > Amazon SQS FIFO (masuk pertama, keluar pertama). Karena mesin aturan adalah layanan terdistribusi penuh, pesan mungkin tidak ditampilkan dalam urutan tertentu saat tindakan Amazon SQS dimulai.

### More information (3)

Saat Anda mengirim data ke antrian Amazon SQS, Anda harus menentukan parameter berikut.

**queueUrl**

URL antrian Amazon SQS yang menerima data.

## useBase64

(Opsional) AWS IoT Events mengkodekan data ke dalam teks Base64, jika Anda menentukan. TRUE Nilai default-nya FALSE.

## payload

(Opsional) Payload default berisi semua pasangan nilai atribut yang memiliki informasi tentang contoh model detektor dan peristiwa memicu tindakan. Anda juga dapat menyesuaikan payload. Untuk informasi selengkapnya, lihat [Payload](#) di Referensi AWS IoT Events API.

### Note

Pastikan bahwa kebijakan yang dilampirkan pada peran AWS IoT Events layanan Anda memberikan sqs : SendMessage izin. Untuk informasi selengkapnya, lihat [Identitas dan manajemen akses untuk AWS IoT Events](#).

Untuk informasi selengkapnya, lihat [SNSTopicPublishAction](#) di dalam Referensi API AWS IoT Events .

Anda juga dapat menggunakan Amazon SNS dan mesin AWS IoT Core aturan untuk memicu suatu AWS Lambda fungsi. Hal ini memungkinkan untuk mengambil tindakan menggunakan layanan lain, seperti Connect Customer, atau bahkan aplikasi perencanaan sumber daya perusahaan perusahaan (ERP).

### Note

Untuk mengumpulkan dan memproses aliran besar catatan data secara real time, Anda dapat menggunakan AWS layanan lain, seperti [Amazon Kinesis](#). Dari sana, Anda dapat menyelesaikan analisis awal dan kemudian mengirim hasilnya AWS IoT Events sebagai input ke detektor.

# Ekspresi untuk memfilter, mengubah, dan memproses data peristiwa

Ekspresi digunakan untuk mengevaluasi data yang masuk, melakukan perhitungan, dan menentukan kondisi di mana tindakan tertentu atau transisi status harus terjadi. AWS IoT Events menyediakan beberapa cara untuk menentukan nilai saat Anda membuat dan memperbarui model detektor. Anda dapat menggunakan ekspresi untuk menentukan nilai literal, atau AWS IoT Events dapat mengevaluasi ekspresi sebelum Anda menentukan nilai tertentu.

## Topik

- [Sintaks untuk memfilter data perangkat dan menentukan tindakan di AWS IoT Events](#)
- [Contoh ekspresi dan penggunaan untuk AWS IoT Events](#)

## Sintaks untuk memfilter data perangkat dan menentukan tindakan di AWS IoT Events

Ekspresi menawarkan sintaks untuk memfilter data perangkat dan menentukan tindakan. Anda dapat menggunakan template literal, operator, fungsi, referensi, dan substitusi dalam ekspresi. AWS IoT Events Dengan menggabungkan komponen-komponen ini, Anda dapat membuat ekspresi yang kuat dan fleksibel untuk memproses data IoT, melakukan perhitungan, memanipulasi string, dan membuat keputusan logis dalam model detektor Anda.

## Literal

- Bilangan Bulat
- Decimal
- String
- Boolean

## Operator

### Unary

- Tidak (Boolean): !

- Tidak (bitwise):  $\sim$
- Minus (aritmatika):  $-$

## String

- Penggabungan:  $+$

Kedua operan harus berupa string. String literal harus diapit dalam tanda kutip tunggal (').

Misalnya: `'my' + 'string' -> 'mystring'`

## Aritmatika

- Penambahan:  $+$

Kedua operan harus numerik.

- Pengurangan:  $-$
- Divisi:  $/$

Hasil pembagian adalah nilai integer bulat kecuali setidaknya salah satu operan (pembagi atau dividen) adalah nilai desimal.

- Perkalian:  $*$

## Bitwise (Bilangan bulat)

- ATAU:  $|$

Misalnya: `13 | 5 -> 13`

- DAN:  $\&$

Misalnya: `13 & 5 -> 5`

- XOR:  $\wedge$

Misalnya: `13 ^ 5 -> 8`


- TIDAK:  $\sim$

Misalnya: `\sim 13 -> -14`

## Boolean

- Kurang dari:  $<$
- Kurang dari atau sama dengan:  $<=$
- Sama dengan:  $==$

- Tidak Sama Dengan: `!=`
- Lebih besar dari atau sama dengan: `>=`
- Lebih besar dari: `>`
- DAN: `&&`
- ATAU: `||`

 Note

Ketika subexpression `||` berisi data yang tidak ditentukan, subexpression itu diperlakukan sebagai `false`

### Tanda kurung

Anda dapat menggunakan tanda kurung untuk mengelompokkan istilah dalam ekspresi.

## Fungsi untuk digunakan dalam AWS IoT Events ekspresi

AWS IoT Events menyediakan serangkaian fungsi bawaan untuk meningkatkan kemampuan ekspresi model detektor Anda. Fungsi-fungsi ini memungkinkan manajemen timer, konversi tipe, pemeriksaan null, identifikasi tipe pemicu, verifikasi input, manipulasi string, dan operasi bitwise. Dengan memanfaatkan fungsi-fungsi ini, Anda dapat membuat logika AWS IoT Events pemrosesan responsif, meningkatkan efektivitas keseluruhan aplikasi IoT Anda.

### Fungsi Bawaan


**`timeout("timer-name")`**

Mengevaluasi `true` apakah timer yang ditentukan telah berlalu. Ganti `timer-name` dengan nama timer yang Anda tentukan, dalam tanda kutip. Dalam tindakan peristiwa, Anda dapat menentukan timer dan kemudian memulai timer, mengatur ulang, atau menghapus salah satu yang Anda tentukan sebelumnya. Lihat `lapangandetectorModelDefinition.states.onInput|onEnter|onExit.events.actions.setTimer.timerName`.

Timer yang disetel dalam satu status dapat direferensikan dalam keadaan yang berbeda. Anda harus mengunjungi negara bagian di mana Anda membuat timer sebelum Anda memasukkan status di mana timer direferensikan.

Misalnya, model detektor memiliki dua status, `TemperatureChecked` dan `RecordUpdated`. Anda membuat timer di `TemperatureChecked` negara bagian. Anda harus mengunjungi `TemperatureChecked` negara bagian terlebih dahulu sebelum Anda dapat menggunakan timer di `RecordUpdated` negara bagian.

Untuk memastikan akurasi, waktu minimum pengatur waktu harus diatur adalah 60 detik.

 Note

`timeout()` mengembalikan `true` hanya pertama kali diperiksa setelah kedaluwarsa timer aktual dan kembali `false` setelahnya.

### **`convert(type, expression)`**

Mengevaluasi nilai ekspresi yang dikonversi ke tipe yang ditentukan. *type* Nilai harus `String`, `Boolean`, atau `Decimal`. Gunakan salah satu kata kunci ini atau ekspresi yang mengevaluasi string yang berisi kata kunci. Hanya konversi berikut yang berhasil dan mengembalikan nilai yang valid:

- Boolean -> string

Mengembalikan string `"true"` atau `"false"`.

- Desimal -> string
- String -> Boolean
- String -> desimal

String yang ditentukan harus merupakan representasi yang valid dari angka desimal, atau `convert()` gagal.

Jika `convert()` tidak mengembalikan nilai yang valid, ekspresi bahwa itu adalah bagian dari juga tidak valid. Hasil ini setara dengan `false` dan tidak akan memicu transisi `actions` atau ke yang `nextState` ditentukan sebagai bagian dari peristiwa di mana ekspresi terjadi.

### **`isNull(expression)`**

Mengevaluasi `true` jika ekspresi mengembalikan `null`. Misalnya, jika input `MyInput` menerima pesan `{ "a": null }`, maka yang berikut ini mengevaluasi `true`, tetapi `isUndefined($input.MyInput.a)` mengevaluasi ke `false`

```
isNull($input.MyInput.a)
```

### **isUndefined**(*expression*)

Mengevaluasi `true` apakah ekspresi tidak terdefinisi. Misalnya, jika input `MyInput` menerima pesan `{ "a": null }`, maka yang berikut ini mengevaluasi `false`, tetapi `isNull($input.MyInput.a)` mengevaluasi ke `true`

```
isUndefined($input.MyInput.a)
```

### **triggerType**("type")

*type* Nilainya bisa `"Message"` atau `"Timer"`. Mengevaluasi `true` apakah kondisi peristiwa di mana itu muncul sedang dievaluasi karena timer telah kedaluwarsa seperti pada contoh berikut.

```
triggerType("Timer")
```

Atau pesan masukan diterima.

```
triggerType("Message")
```

### **currentInput**("input")

Mengevaluasi `true` apakah kondisi acara di mana itu muncul sedang dievaluasi karena pesan input yang ditentukan telah diterima. Misalnya, jika input `Command` menerima pesan `{ "value": "Abort" }`, maka yang berikut ini akan dievaluasi `true`

```
currentInput("Command")
```

Gunakan fungsi ini untuk memverifikasi bahwa kondisi sedang dievaluasi karena input tertentu telah diterima dan timer belum kedaluwarsa, seperti pada ekspresi berikut.

```
currentInput("Command") && $input.Command.value == "Abort"
```

## Fungsi Pencocokan String

### **startsWith**(*expression1*, *expression2*)

Mengevaluasi `true` apakah ekspresi string pertama dimulai dengan ekspresi string kedua. Misalnya, jika input `MyInput` menerima pesan `{ "status": "offline" }`, maka yang berikut ini akan dievaluasi. `true`

```
startsWith($input.MyInput.status, "off")
```

Kedua ekspresi harus mengevaluasi nilai string. Jika salah satu ekspresi tidak mengevaluasi nilai string, maka hasil dari fungsi tersebut tidak terdefinisi. Tidak ada konversi yang dilakukan.

### **endsWith**(*expression1*, *expression2*)

Mengevaluasi `true` apakah ekspresi string pertama berakhir dengan ekspresi string kedua. Misalnya, jika input `MyInput` menerima pesan `{ "status": "offline" }`, maka yang berikut ini akan dievaluasi. `true`

```
endsWith($input.MyInput.status, "line")
```

Kedua ekspresi harus mengevaluasi nilai string. Jika salah satu ekspresi tidak mengevaluasi nilai string, maka hasil dari fungsi tersebut tidak terdefinisi. Tidak ada konversi yang dilakukan.

### **contains**(*expression1*, *expression2*)

Mengevaluasi `true` apakah ekspresi string pertama berisi ekspresi string kedua. Misalnya, jika input `MyInput` menerima pesan `{ "status": "offline" }`, maka yang berikut ini akan dievaluasi. `true`

```
contains($input.MyInput.value, "fli")
```

Kedua ekspresi harus mengevaluasi nilai string. Jika salah satu ekspresi tidak mengevaluasi nilai string, maka hasil dari fungsi tersebut tidak terdefinisi. Tidak ada konversi yang dilakukan.

## Fungsi Manipulasi Bitwise Integer

### **bitor**(*expression1*, *expression2*)

Mengevaluasi bitwise OR dari ekspresi integer (operasi OR biner dilakukan pada bit yang sesuai dari bilangan bulat). Misalnya, jika input `MyInput` menerima pesan `{ "value1": 13, "value2": 5 }`, maka yang berikut ini akan dievaluasi. `13`

```
bitor($input.MyInput.value1, $input.MyInput.value2)
```

Kedua ekspresi harus mengevaluasi ke nilai integer. Jika salah satu ekspresi tidak mengevaluasi nilai integer, maka hasil dari fungsi tersebut tidak terdefinisi. Tidak ada konversi yang dilakukan.

### **bitand**(*expression1*, *expression2*)

Mengevaluasi bitwise AND dari ekspresi integer (operasi biner AND dilakukan pada bit yang sesuai dari bilangan bulat). Misalnya, jika input MyInput menerima pesan{ "value1": 13, "value2": 5 }, maka yang berikut ini akan dievaluasi. 5

```
bitand($input.MyInput.value1, $input.MyInput.value2)
```

Kedua ekspresi harus mengevaluasi ke nilai integer. Jika salah satu ekspresi tidak mengevaluasi nilai integer, maka hasil dari fungsi tersebut tidak terdefinisi. Tidak ada konversi yang dilakukan.

### **bitxor**(*expression1*, *expression2*)

Mengevaluasi XOR bitwise dari ekspresi integer (operasi XOR biner dilakukan pada bit yang sesuai dari bilangan bulat). Misalnya, jika input MyInput menerima pesan{ "value1": 13, "value2": 5 }, maka yang berikut ini akan dievaluasi. 8

```
bitxor($input.MyInput.value1, $input.MyInput.value2)
```

Kedua ekspresi harus mengevaluasi ke nilai integer. Jika salah satu ekspresi tidak mengevaluasi nilai integer, maka hasil dari fungsi tersebut tidak terdefinisi. Tidak ada konversi yang dilakukan.

### **bitnot**(*expression*)

Mengevaluasi bitwise NOT dari ekspresi integer (operasi NOT biner dilakukan pada bit integer). Misalnya, jika input MyInput menerima pesan{ "value": 13 }, maka yang berikut ini akan dievaluasi. -14

```
bitnot($input.MyInput.value)
```

Kedua ekspresi harus mengevaluasi ke nilai integer. Jika salah satu ekspresi tidak mengevaluasi nilai integer, maka hasil dari fungsi tersebut tidak terdefinisi. Tidak ada konversi yang dilakukan.

## AWS IoT Events referensi untuk input dan variabel dalam ekspresi

### Masukan

`$input.input-name.path-to-data`

`input-name` adalah masukan yang Anda buat menggunakan [CreateInput](#) tindakan.

Misalnya, jika Anda memiliki masukan bernama `TemperatureInput` yang Anda tetapkan `inputDefinition.attributes.jsonPath` entri, nilainya mungkin muncul di bidang yang tersedia berikut.

```
{
  "temperature": 78.5,
  "date": "2018-10-03T16:09:09Z"
}
```

Untuk mereferensikan nilai `temperature` bidang, gunakan perintah berikut.

```
$input.TemperatureInput.temperature
```

Untuk bidang yang nilainya adalah array, Anda dapat mereferensikan anggota array menggunakan `[n]`. Misalnya, diberikan nilai-nilai berikut:

```
{
  "temperatures": [
    78.4,
    77.9,
    78.8
  ],
  "date": "2018-10-03T16:09:09Z"
}
```

Nilai `78.8` dapat direferensikan dengan perintah berikut.

```
$input.TemperatureInput.temperatures[2]
```

### Variabel

`$variable.variable-name`

*variable-name* Ini adalah variabel yang Anda definisikan menggunakan [CreateDetectorModel](#) tindakan.

Misalnya, jika Anda memiliki variabel bernama `TechnicianID` yang Anda definisikan menggunakan `detectorDefinition.states.onInputEvents.actions.setVariable.variable` Anda dapat mereferensikan nilai (string) yang terakhir diberikan ke variabel dengan perintah berikut.

```
$variable.TechnicianID
```

Anda dapat mengatur nilai variabel hanya menggunakan `setVariable` tindakan. Anda tidak dapat menetapkan nilai untuk variabel dalam ekspresi. Variabel tidak dapat di-unset. Misalnya, Anda tidak dapat menetapkan nilainya `null`.

#### Note

Dalam referensi yang menggunakan pengidentifikasi yang tidak mengikuti pola (ekspresi reguler) `[a-zA-Z][a-zA-Z0-9_]*`, Anda harus menyertakan pengidentifikasi tersebut di backticks (```). Misalnya, referensi ke input bernama `MyInput` dengan bidang bernama `_value` harus menentukan bidang ini sebagai `$input.MyInput.`_value``.

Saat Anda menggunakan referensi dalam ekspresi, periksa hal berikut:

- Bila Anda menggunakan referensi sebagai operan dengan satu atau beberapa operator, pastikan semua tipe data yang Anda referensikan kompatibel.

Misalnya, dalam ekspresi berikut, integer 2 adalah operan dari kedua operator `==` dan `&&`. Untuk memastikan bahwa operan kompatibel, `$variable.testVariable + 1` dan `$variable.testVariable` harus mereferensikan bilangan bulat atau desimal.

Selain itu, integer 1 adalah operan dari operator `+`. Oleh karena itu, `$variable.testVariable` harus referensi bilangan bulat atau desimal.

```
'$variable.testVariable + 1 == 2 && $variable.testVariable'
```

- Bila Anda menggunakan referensi sebagai argumen yang diteruskan ke fungsi, pastikan bahwa fungsi tersebut mendukung tipe data yang Anda referensikan.

Misalnya, `timeout("time-name")` fungsi berikut membutuhkan string dengan tanda kutip ganda sebagai argumen. Jika Anda menggunakan referensi untuk `timer-name` nilainya, Anda harus mereferensikan string dengan tanda kutip ganda.

```
timeout("timer-name")
```

#### Note

Untuk `convert(type, expression)` fungsi, jika Anda menggunakan referensi untuk `type` nilai, hasil evaluasi dari referensi Anda harus `String`, `Decimal`, atau `Boolean`.

AWS IoT Events ekspresi mendukung tipe data integer, desimal, string, dan Boolean. Tabel berikut menyediakan daftar pasangan jenis yang tidak kompatibel.

#### Pasangan tipe yang tidak kompatibel

Bilangan bulat, string

Bilangan bulat, Boolean

Desimal, string

Desimal, Boolean

Tali, Boolean

## Templat substitusi untuk ekspresi AWS IoT Events

```
'${expression}'
```

`${}` Mengidentifikasi string sebagai string interpolasi. Itu `expression` bisa berupa AWS IoT Events ekspresi apa saja. Ini termasuk operator, fungsi, dan referensi.

Misalnya, Anda menggunakan [SetVariableAction](#) tindakan untuk mendefinisikan variabel. `variableName` adalah `SensorID`, dan `value` adalah `10`. Anda dapat membuat template substitusi berikut.

Templat substitusi	String hasil
<code>'\${'Sensor ' + \$variable.SensorID}'</code>	"Sensor 10"
<code>'Sensor ' + '\${\$variable.SensorID + 1}'</code>	"Sensor 11"
<code>'Sensor 10: \${\$variable.SensorID == 10}'</code>	"Sensor 10: true"
<code>'{"sensor\":"\${\$variable.SensorID + 1}\"}'</code>	"{"sensor\":"11\"}"
<code>'{"sensor\":"\${\$variable.SensorID + 1}}'</code>	"{"sensor\":"11}"

## Contoh ekspresi dan penggunaan untuk AWS IoT Events

Anda dapat menentukan nilai dalam model detektor dengan cara berikut:

- Masukkan ekspresi yang didukung di AWS IoT Events konsol.
- Lewati ekspresi ke parameter AWS IoT Events APIs as.

Ekspresi mendukung literal, operator, fungsi, referensi, dan templat substitusi.

### Important

Ekspresi Anda harus mereferensikan nilai integer, desimal, string, atau Boolean.

## Menulis AWS IoT Events ekspresi

Lihat contoh berikut untuk membantu Anda menulis AWS IoT Events ekspresi Anda:

## Literal

Untuk nilai literal, ekspresi harus berisi tanda kutip tunggal. Nilai Boolean harus salah satu `true` atau `false`.

```
'123'      # Integer
'123.12'   # Decimal
'hello'    # String
'true'     # Boolean
```

## Referensi

Untuk referensi, Anda harus menentukan variabel atau nilai input.

- Input berikut mereferensikan angka desimal, `10.01`

```
$input.GreenhouseInput.temperature
```

- Variabel berikut referensi string, `Greenhouse Temperature Table`.

```
$variable.TableName
```

## Templat substitusi

Untuk templat substitusi, Anda harus menggunakan `${}`, dan templat harus berada dalam tanda kutip tunggal. Templat substitusi juga dapat berisi kombinasi dari templat literal, operator, fungsi, referensi, dan substitusi.

- Hasil evaluasi dari ekspresi berikut adalah string, `50.018 in Fahrenheit`.

```
'${$input.GreenhouseInput.temperature * 9 / 5 + 32} in Fahrenheit'
```

- Hasil evaluasi dari ekspresi berikut adalah string, `{"sensor_id\":\"Sensor_1\", \"temperature\":\"50.018\"}`.

```
'{\"sensor_id\":\"${$input.GreenhouseInput.sensors[0].sensor1}\", \"temperature\": \"${$input.GreenhouseInput.temperature*9/5+32}\"}'
```

## Penggabungan string

Untuk rangkaian string, Anda harus menggunakan `+`. Rangkaian string juga dapat berisi kombinasi dari templat literal, operator, fungsi, referensi, dan substitusi.

- Hasil evaluasi dari ekspresi berikut adalah string,Greenhouse Temperature Table 2000-01-01.

```
'Greenhouse Temperature Table ' + $input.GreenhouseInput.date
```

# AWS IoT Events contoh model detektor

Halaman ini menyediakan daftar contoh kasus penggunaan yang menunjukkan cara mengkonfigurasi berbagai AWS IoT Events fitur. Contohnya berkisar dari deteksi dasar seperti ambang suhu hingga deteksi anomali yang lebih maju dan skenario pembelajaran mesin. Setiap contoh mencakup prosedur dan cuplikan kode untuk membantu Anda mengatur AWS IoT Events deteksi, tindakan, dan integrasi. Contoh-contoh ini menunjukkan fleksibilitas AWS IoT Events layanan dan bagaimana hal itu dapat disesuaikan untuk beragam aplikasi IoT dan kasus penggunaan. Lihat halaman ini saat menjelajahi AWS IoT Events kemampuan atau jika Anda memerlukan panduan untuk menerapkan alur kerja deteksi atau otomatisasi tertentu.

## Topik

- [Contoh: Menggunakan kontrol suhu HVAC dengan AWS IoT Events](#)
- [Contoh: Derek mendeteksi kondisi menggunakan AWS IoT Events](#)
- [Kirim perintah sebagai respons terhadap kondisi yang terdeteksi di AWS IoT Events](#)
- [Model AWS IoT Events detektor untuk pemantauan derek](#)
- [AWS IoT Events input untuk pemantauan derek](#)
- [Kirim alarm dan pesan operasional dengan AWS IoT Events](#)
- [Contoh: deteksi AWS IoT Events peristiwa dengan sensor dan aplikasi](#)
- [Contoh: Perangkat HeartBeat untuk memantau koneksi perangkat dengan AWS IoT Events](#)
- [Contoh: Alarm ISA di AWS IoT Events](#)
- [Contoh: Membangun alarm sederhana dengan AWS IoT Events](#)

## Contoh: Menggunakan kontrol suhu HVAC dengan AWS IoT Events

### Cerita latar belakang

Contoh ini mengimplementasikan model kontrol suhu (termostat) dengan fitur-fitur ini:

- Satu model detektor yang Anda tentukan yang dapat memantau dan mengontrol beberapa area. (Sebuah instance detektor akan dibuat untuk setiap area.)

- Setiap instance detektor menerima data suhu dari beberapa sensor yang ditempatkan di setiap area kontrol.
- Anda dapat mengubah suhu yang diinginkan (titik setel) untuk setiap area kapan saja.
- Anda dapat menentukan parameter operasional untuk setiap area dan mengubah parameter ini kapan saja.
- Anda dapat menambahkan sensor ke atau menghapus sensor dari suatu area kapan saja.
- Anda dapat mengaktifkan unit pemanasan dan pendingin waktu minimum untuk melindunginya dari kerusakan.
- Detektor akan menolak, dan melaporkan, pembacaan sensor anomali.
- Anda dapat menentukan titik setel suhu darurat. Jika ada satu sensor yang melaporkan suhu di atas atau di bawah titik setel yang telah Anda tentukan, unit pemanas atau pendingin akan segera diaktifkan, dan detektor akan melaporkan lonjakan suhu tersebut.

Contoh ini menunjukkan kemampuan fungsional berikut:

- Buat model detektor acara.
- Buat input.
- Menelan input ke dalam model detektor.
- Evaluasi kondisi pemicu.
- Lihat variabel keadaan dalam kondisi dan atur nilai variabel tergantung pada kondisi.
- Lihat pengatur waktu dalam kondisi dan atur pengatur waktu tergantung pada kondisi.
- Lakukan tindakan yang mengirim pesan Amazon SNS dan MQTT.

## Definisi input untuk sistem HVAC di AWS IoT Events

A `seedTemperatureInput` digunakan untuk membuat instance detektor untuk suatu area dan menentukan parameter operasionalnya.

Mengkonfigurasi input untuk sistem HVAC penting untuk pengendalian iklim AWS IoT Events yang efektif. Contoh ini menunjukkan cara mengatur input yang menangkap parameter seperti, suhu, kelembaban, hunian, dan data konsumsi energi. Pelajari cara menentukan atribut input, mengonfigurasi sumber data, dan menyiapkan aturan pra-pemrosesan untuk membantu model detektor Anda menerima informasi yang akurat dan tepat waktu untuk manajemen dan efisiensi yang optimal.

Perintah CLI yang digunakan:

```
aws iotevents create-input --cli-input-json file://seedInput.json
```

Berkas: seedInput.json

```
{
  "inputName": "seedTemperatureInput",
  "inputDescription": "Temperature seed values.",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "areaId" },
      { "jsonPath": "desiredTemperature" },
      { "jsonPath": "allowedError" },
      { "jsonPath": "rangeHigh" },
      { "jsonPath": "rangeLow" },
      { "jsonPath": "anomalousHigh" },
      { "jsonPath": "anomalousLow" },
      { "jsonPath": "sensorCount" },
      { "jsonPath": "noDelay" }
    ]
  }
}
```

Respons:

```
{
  "inputConfiguration": {
    "status": "ACTIVE",
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/seedTemperatureInput",
    "lastUpdateTime": 1557519620.736,
    "creationTime": 1557519620.736,
    "inputName": "seedTemperatureInput",
    "inputDescription": "Temperature seed values."
  }
}
```

A temperatureInput harus dikirim oleh setiap sensor di setiap area, seperlunya.

Perintah CLI yang digunakan:

```
aws iotevents create-input --cli-input-json file://temperatureInput.json
```

Berkas: temperatureInput.json

```
{
  "inputName": "temperatureInput",
  "inputDescription": "Temperature sensor unit data.",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorId" },
      { "jsonPath": "areaId" },
      { "jsonPath": "sensorData.temperature" }
    ]
  }
}
```

Respons:

```
{
  "inputConfiguration": {
    "status": "ACTIVE",
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/temperatureInput",
    "lastUpdateTime": 1557519707.399,
    "creationTime": 1557519707.399,
    "inputName": "temperatureInput",
    "inputDescription": "Temperature sensor unit data."
  }
}
```

## Definisi model detektor untuk sistem HVAC menggunakan AWS IoT Events

`areaDetectorModel` mendefinisikan bagaimana setiap instance detektor bekerja. Setiap state machine instance akan menelan pembacaan sensor suhu, kemudian mengubah status dan mengirim pesan kontrol tergantung pada pembacaan ini.

Perintah CLI yang digunakan:

```
aws iotevents create-detector-model --cli-input-json file://areaDetectorModel.json
```

## Berkas: areaDetectorModel.json

```
{
  "detectorModelName": "areaDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "start",
        "onEnter": {
          "events": [
            {
              "eventName": "prepare",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "sensorId",
                    "value": "0"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "reportedTemperature",
                    "value": "0.1"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "resetMe",
                    "value": "false"
                  }
                }
              ]
            }
          ]
        }
      }
    ],
    "onInput": {
      "transitionEvents": [
        {
          "eventName": "initialize",
          "condition": "$input.seedTemperatureInput.sensorCount > 0",
          "actions": [
            {
              "setVariable": {
```

```
        "variableName": "rangeHigh",
        "value": "$input.seedTemperatureInput.rangeHigh"
    }
},
{
    "setVariable": {
        "variableName": "rangeLow",
        "value": "$input.seedTemperatureInput.rangeLow"
    }
},
{
    "setVariable": {
        "variableName": "desiredTemperature",
        "value": "$input.seedTemperatureInput.desiredTemperature"
    }
},
{
    "setVariable": {
        "variableName": "averageTemperature",
        "value": "$input.seedTemperatureInput.desiredTemperature"
    }
},
{
    "setVariable": {
        "variableName": "allowedError",
        "value": "$input.seedTemperatureInput.allowedError"
    }
},
{
    "setVariable": {
        "variableName": "anomalousHigh",
        "value": "$input.seedTemperatureInput.anomalousHigh"
    }
},
{
    "setVariable": {
        "variableName": "anomalousLow",
        "value": "$input.seedTemperatureInput.anomalousLow"
    }
},
{
    "setVariable": {
        "variableName": "sensorCount",
        "value": "$input.seedTemperatureInput.sensorCount"
    }
}
```

```

    }
  },
  {
    "setVariable": {
      "variableName": "noDelay",
      "value": "$input.seedTemperatureInput.noDelay == true"
    }
  }
],
"nextState": "idle"
},
{
  "eventName": "reset",
  "condition": "($variable.resetMe == true) &&
($input.temperatureInput.sensorData.temperature < $variable.anomalousHigh &&
$input.temperatureInput.sensorData.temperature > $variable.anomalousLow)",
  "actions": [
    {
      "setVariable": {
        "variableName": "averageTemperature",
        "value": "((( $variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
      }
    }
  ],
  "nextState": "idle"
}
]
},
"onExit": {
  "events": [
    {
      "eventName": "resetHeatCool",
      "condition": "true",
      "actions": [
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
          }
        },
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOff"
          }
        }
      ]
    }
  ]
}
}

```

```

    },
    {
      "iotTopicPublish": {
        "mqttTopic": "hvac/Heating/Off"
      }
    },
    {
      "iotTopicPublish": {
        "mqttTopic": "hvac/Cooling/Off"
      }
    }
  ]
}
]
},
},
{
  "stateName": "idle",
  "onInput": {
    "events": [
      {
        "eventName": "whatWasInput",
        "condition": "true",
        "actions": [
          {
            "setVariable": {
              "variableName": "sensorId",
              "value": "$input.temperatureInput.sensorId"
            }
          },
          {
            "setVariable": {
              "variableName": "reportedTemperature",
              "value": "$input.temperatureInput.sensorData.temperature"
            }
          }
        ]
      },
      {
        "eventName": "changeDesired",
        "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",

```

```

    "actions": [
      {
        "setVariable": {
          "variableName": "desiredTemperature",
          "value": "$input.seedTemperatureInput.desiredTemperature"
        }
      }
    ],
    {
      "eventName": "calculateAverage",
      "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
      "actions": [
        {
          "setVariable": {
            "variableName": "averageTemperature",
            "value": "((( $variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
          }
        }
      ]
    }
  ],
  "transitionEvents": [
    {
      "eventName": "anomalousInputArrived",
      "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
      "actions": [
        {
          "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/anomaly"
          }
        }
      ],
      "nextState": "idle"
    },
    {
      "eventName": "highTemperatureSpike",

```

```
    "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
    "actions": [
      {
        "iotTopicPublish": {
          "mqttTopic": "temperatureSensor/spike"
        }
      },
      {
        "sns": {
          "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
        }
      },
      {
        "iotTopicPublish": {
          "mqttTopic": "hvac/Cooling/On"
        }
      },
      {
        "setVariable": {
          "variableName": "enteringNewState",
          "value": "true"
        }
      }
    ],
    "nextState": "cooling"
  },
  {
    "eventName": "lowTemperatureSpike",
    "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
    "actions": [
      {
        "iotTopicPublish": {
          "mqttTopic": "temperatureSensor/spike"
        }
      },
      {
        "sns": {
          "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
        }
      }
    ]
  }
}
```

```

        "iotTopicPublish": {
            "mqttTopic": "hvac/Heating/On"
        }
    },
    {
        "setVariable": {
            "variableName": "enteringNewState",
            "value": "true"
        }
    }
],
"nextState": "heating"
},

{
    "eventName": "highTemperatureThreshold",
    "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) >
($variable.desiredTemperature + $variable.allowedError))",
    "actions": [
        {
            "sns": {
                "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
            }
        },
        {
            "iotTopicPublish": {
                "mqttTopic": "hvac/Cooling/On"
            }
        },
        {
            "setVariable": {
                "variableName": "enteringNewState",
                "value": "true"
            }
        }
    ]
},
"nextState": "cooling"
},

{
    "eventName": "lowTemperatureThreshold",

```

```

        "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) <
($variable.desiredTemperature - $variable.allowedError))",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
                }
            },
            {
                "iotTopicPublish": {
                    "mqttTopic": "hvac/Heating/On"
                }
            },
            {
                "setVariable": {
                    "variableName": "enteringNewState",
                    "value": "true"
                }
            }
        ],
        "nextState": "heating"
    }
]
}
},

{
    "stateName": "cooling",
    "onEnter": {
        "events": [
            {
                "eventName": "delay",
                "condition": "!$variable.noDelay && $variable.enteringNewState",
                "actions": [
                    {
                        "setTimer": {
                            "timerName": "coolingTimer",
                            "seconds": 180
                        }
                    },
                    {
                        "setVariable": {

```

```
        "variableName": "goodToGo",
        "value": "false"
      }
    ]
  },
  {
    "eventName": "dontDelay",
    "condition": "$variable.noDelay == true",
    "actions": [
      {
        "setVariable": {
          "variableName": "goodToGo",
          "value": "true"
        }
      }
    ]
  },
  {
    "eventName": "beenHere",
    "condition": "true",
    "actions": [
      {
        "setVariable": {
          "variableName": "enteringNewState",
          "value": "false"
        }
      }
    ]
  }
],
"onInput": {
  "events": [
    {
      "eventName": "whatWasInput",
      "condition": "true",
      "actions": [
        {
          "setVariable": {
            "variableName": "sensorId",
            "value": "$input.temperatureInput.sensorId"
          }
        }
      ]
    }
  ]
}
```

```
    },
    {
      "setVariable": {
        "variableName": "reportedTemperature",
        "value": "$input.temperatureInput.sensorData.temperature"
      }
    }
  ]
},
{
  "eventName": "changeDesired",
  "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
  "actions": [
    {
      "setVariable": {
        "variableName": "desiredTemperature",
        "value": "$input.seedTemperatureInput.desiredTemperature"
      }
    }
  ]
},
{
  "eventName": "calculateAverage",
  "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
  "actions": [
    {
      "setVariable": {
        "variableName": "averageTemperature",
        "value": "((( $variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
      }
    }
  ]
},
{
  "eventName": "areWeThereYet",
  "condition": "(timeout(\"coolingTimer\"))",
  "actions": [
    {
      "setVariable": {
        "variableName": "goodToGo",
```

```

        "value": "true"
      }
    }
  ],
  "transitionEvents": [
    {
      "eventName": "anomalousInputArrived",
      "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
      "actions": [
        {
          "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/anomaly"
          }
        }
      ],
      "nextState": "cooling"
    },
    {
      "eventName": "highTemperatureSpike",
      "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
      "actions": [
        {
          "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/spike"
          }
        }
      ],
      "nextState": "cooling"
    },
    {
      "eventName": "lowTemperatureSpike",
      "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
      "actions": [
        {
          "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/spike"
          }
        }
      ]
    }
  ]
}

```

```

    }
  },
  {
    "sns": {
      "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0ff"
    }
  },
  {
    "sns": {
      "targetArn": "arn:aws:sns:us-west-2:123456789012:heat0n"
    }
  },
  {
    "iotTopicPublish": {
      "mqttTopic": "hvac/Cooling/Off"
    }
  },
  {
    "iotTopicPublish": {
      "mqttTopic": "hvac/Heating/On"
    }
  },
  {
    "setVariable": {
      "variableName": "enteringNewState",
      "value": "true"
    }
  }
],
"nextState": "heating"
},
{
  "eventName": "desiredTemperature",
  "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) <=
($variable.desiredTemperature - $variable.allowedError)) && $variable.goodToGo ==
true",
  "actions": [
    {
      "sns": {
        "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0ff"
      }
    }
  ]
},

```

```
        {
          "iotTopicPublish": {
            "mqttTopic": "hvac/Cooling/Off"
          }
        }
      ],
      "nextState": "idle"
    }
  ]
},

{
  "stateName": "heating",
  "onEnter": {
    "events": [
      {
        "eventName": "delay",
        "condition": "!$variable.noDelay && $variable.enteringNewState",
        "actions": [
          {
            "setTimer": {
              "timerName": "heatingTimer",
              "seconds": 120
            }
          },
          {
            "setVariable": {
              "variableName": "goodToGo",
              "value": "false"
            }
          }
        ]
      },
      {
        "eventName": "dontDelay",
        "condition": "$variable.noDelay == true",
        "actions": [
          {
            "setVariable": {
              "variableName": "goodToGo",
              "value": "true"
            }
          }
        ]
      }
    ]
  }
}
```

```

    }
  ]
},
{
  "eventName": "beenHere",
  "condition": "true",
  "actions": [
    {
      "setVariable": {
        "variableName": "enteringNewState",
        "value": "false"
      }
    }
  ]
}
]
},
],
},
"onInput": {
  "events": [
    {
      "eventName": "whatWasInput",
      "condition": "true",
      "actions": [
        {
          "setVariable": {
            "variableName": "sensorId",
            "value": "$input.temperatureInput.sensorId"
          }
        },
        {
          "setVariable": {
            "variableName": "reportedTemperature",
            "value": "$input.temperatureInput.sensorData.temperature"
          }
        }
      ]
    },
    {
      "eventName": "changeDesired",
      "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
      "actions": [
        {

```

```

        "setVariable": {
            "variableName": "desiredTemperature",
            "value": "$input.seedTemperatureInput.desiredTemperature"
        }
    }
]
},
{
    "eventName": "calculateAverage",
    "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
    "actions": [
        {
            "setVariable": {
                "variableName": "averageTemperature",
                "value": "((( $variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
            }
        }
    ]
},
{
    "eventName": "areWeThereYet",
    "condition": "(timeout(\"heatingTimer\"))",
    "actions": [
        {
            "setVariable": {
                "variableName": "goodToGo",
                "value": "true"
            }
        }
    ]
}
],
"transitionEvents": [
    {
        "eventName": "anomalousInputArrived",
        "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
        "actions": [
            {
                "iotTopicPublish": {

```

```
        "mqttTopic": "temperatureSensor/anomaly"
      }
    }
  ],
  "nextState": "heating"
},
{
  "eventName": "highTemperatureSpike",
  "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
  "actions": [
    {
      "iotTopicPublish": {
        "mqttTopic": "temperatureSensor/spike"
      }
    },
    {
      "sns": {
        "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
      }
    },
    {
      "sns": {
        "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
      }
    },
    {
      "iotTopicPublish": {
        "mqttTopic": "hvac/Heating/Off"
      }
    },
    {
      "iotTopicPublish": {
        "mqttTopic": "hvac/Cooling/On"
      }
    },
    {
      "setVariable": {
        "variableName": "enteringNewState",
        "value": "true"
      }
    }
  ]
},
],
```

```

        "nextState": "cooling"
    },
    {
        "eventName": "lowTemperatureSpike",
        "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
        "actions": [
            {
                "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/spike"
                }
            }
        ],
        "nextState": "heating"
    },
    {
        "eventName": "desiredTemperature",
        "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) >=
($variable.desiredTemperature + $variable.allowedError)) && $variable.goodToGo ==
true",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
                }
            },
            {
                "iotTopicPublish": {
                    "mqttTopic": "hvac/Heating/Off"
                }
            }
        ],
        "nextState": "idle"
    }
]
}
}
],
"initialStateName": "start"

```

```

},
"key": "areaId",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}

```

Respons:

```

{
  "detectorModelConfiguration": {
    "status": "ACTIVATING",
    "lastUpdateTime": 1557523491.168,
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "creationTime": 1557523491.168,
    "detectorModelArn": "arn:aws:iotevents:us-west-2:123456789012:detectorModel/areaDetectorModel",
    "key": "areaId",
    "detectorModelName": "areaDetectorModel",
    "detectorModelVersion": "1"
  }
}

```

## BatchPutMessage contoh untuk sistem HVAC di AWS IoT Events

Dalam contoh ini, BatchPutMessage digunakan untuk membuat instance detektor untuk suatu area dan menentukan parameter operasi awal.

Perintah CLI yang digunakan:

```
aws iotevents-data batch-put-message --cli-input-json file://seedExample.json --cli-binary-format raw-in-base64-out
```

Berkas: seedExample.json

```

{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "seedTemperatureInput",
      "payload": "{\"areaId\": \"Area51\", \"desiredTemperature\": 20.0, \"allowedError\": 0.7, \"rangeHigh\": 30.0, \"rangeLow\": 15.0, \"anomalousHigh\": 60.0, \"anomalousLow\": 0.0, \"sensorCount\": 10, \"noDelay\": false}"
    }
  ]
}

```

```
}
]
}
```

Respons:

```
{
  "BatchPutMessageErrorEntries": []
}
```

Dalam contoh ini, BatchPutMessage digunakan untuk melaporkan pembacaan sensor suhu untuk sensor tunggal di suatu area.

Perintah CLI yang digunakan:

```
aws iotevents-data batch-put-message --cli-input-json file://temperatureExample.json --cli-binary-format raw-in-base64-out
```

Berkas: temperatureExample.json

```
{
  "messages": [
    {
      "messageId": "00005",
      "inputName": "temperatureInput",
      "payload": "{\"sensorId\": \"05\", \"areaId\": \"Area51\", \"sensorData\": {\"temperature\": 23.12} }"
    }
  ]
}
```

Respons:

```
{
  "BatchPutMessageErrorEntries": []
}
```

Dalam contoh ini, BatchPutMessage digunakan untuk mengubah suhu yang diinginkan untuk suatu daerah.

Perintah CLI yang digunakan:

```
aws iotevents-data batch-put-message --cli-input-json file://seedSetDesiredTemp.json --cli-binary-format raw-in-base64-out
```

Berkas: seedSetDesiredTemp.json

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "seedTemperatureInput",
      "payload": "{\"areaId\": \"Area51\", \"desiredTemperature\": 23.0}"
    }
  ]
}
```

Respons:

```
{
  "BatchPutMessageErrorEntries": []
}
```

Contoh pesan Amazon SNS yang dihasilkan oleh instance Area51 detektor:

```
Heating system off command> {
  "eventTime":1557520274729,
  "payload":{
    "actionExecutionId":"f3159081-bac3-38a4-96f7-74af0940d0a4",
    "detector":{
      "detectorModelName":"areaDetectorModel",
      "keyValue":"Area51",
      "detectorModelVersion":"1"
    },
    "eventTriggerDetails":{
      "inputName":"seedTemperatureInput",
      "messageId":"00001",
      "triggerType":"Message"
    },
    "state":{
      "stateName":"start",
      "variables":{
```

```

    "sensorCount":10,
    "rangeHigh":30.0,
    "resetMe":false,
    "enteringNewState":true,
    "averageTemperature":20.0,
    "rangeLow":15.0,
    "noDelay":false,
    "allowedError":0.7,
    "desiredTemperature":20.0,
    "anomalousHigh":60.0,
    "reportedTemperature":0.1,
    "anomalousLow":0.0,
    "sensorId":0
  },
  "timers":{}
}
},
"eventName":"resetHeatCool"
}

```

```

Cooling system off command> {
  "eventTime":1557520274729,
  "payload":{
    "actionExecutionId":"98f6a1b5-8f40-3cdb-9256-93afd4d66192",
    "detector":{
      "detectorModelName":"areaDetectorModel",
      "keyValue":"Area51",
      "detectorModelVersion":"1"
    },
    "eventTriggerDetails":{
      "inputName":"seedTemperatureInput",
      "messageId":"00001",
      "triggerType":"Message"
    },
    "state":{
      "stateName":"start",
      "variables":{
        "sensorCount":10,
        "rangeHigh":30.0,
        "resetMe":false,
        "enteringNewState":true,
        "averageTemperature":20.0,

```

```

    "rangeLow":15.0,
    "noDelay":false,
    "allowedError":0.7,
    "desiredTemperature":20.0,
    "anomalousHigh":60.0,
    "reportedTemperature":0.1,
    "anomalousLow":0.0,
    "sensorId":0
  },
  "timers":{}
}
},
"eventName":"resetHeatCool"
}

```

Dalam contoh ini, kita menggunakan DescribeDetector API untuk mendapatkan informasi tentang keadaan saat ini dari instance detektor.

```
aws iotevents-data describe-detector --detector-model-name areaDetectorModel --key-value Area51
```

Respons:

```

{
  "detector": {
    "lastUpdateTime": 1557521572.216,
    "creationTime": 1557520274.405,
    "state": {
      "variables": [
        {
          "name": "resetMe",
          "value": "false"
        },
        {
          "name": "rangeLow",
          "value": "15.0"
        },
        {
          "name": "noDelay",
          "value": "false"
        },
        {
          "name": "desiredTemperature",

```

```
        "value": "20.0"
      },
      {
        "name": "anomalousLow",
        "value": "0.0"
      },
      {
        "name": "sensorId",
        "value": "\"01\""
      },
      {
        "name": "sensorCount",
        "value": "10"
      },
      {
        "name": "rangeHigh",
        "value": "30.0"
      },
      {
        "name": "enteringNewState",
        "value": "false"
      },
      {
        "name": "averageTemperature",
        "value": "19.572"
      },
      {
        "name": "allowedError",
        "value": "0.7"
      },
      {
        "name": "anomalousHigh",
        "value": "60.0"
      },
      {
        "name": "reportedTemperature",
        "value": "15.72"
      },
      {
        "name": "goodToGo",
        "value": "false"
      }
    ],
    "stateName": "idle",
```

```
        "timers": [
          {
            "timestamp": 1557520454.0,
            "name": "idleTimer"
          }
        ],
        "keyValue": "Area51",
        "detectorModelName": "areaDetectorModel",
        "detectorModelVersion": "1"
      }
    }
  }
```

## BatchUpdateDetector contoh untuk sistem HVAC di AWS IoT Events

Dalam contoh ini, BatchUpdateDetector digunakan untuk mengubah parameter operasional untuk instance detektor kerja.

Manajemen sistem HVAC yang efisien seringkali memerlukan pembaruan batch ke beberapa detektor. Bagian ini menunjukkan cara menggunakan AWS IoT Events fitur pembaruan batch untuk detektor. Pelajari cara memodifikasi beberapa parameter kontrol secara bersamaan, memperbarui nilai ambang batas, sehingga Anda dapat menyesuaikan tindakan respons di seluruh armada perangkat, meningkatkan kemampuan Anda untuk mengelola sistem skala besar secara efektif.

Perintah CLI yang digunakan:

```
aws iotevents-data batch-update-detector --cli-input-json file://areaDM.BUD.json
```

Berkas: areaDM.BUD.json

```
{
  "detectors": [
    {
      "messageId": "0001",
      "detectorModelName": "areaDetectorModel",
      "keyValue": "Area51",
      "state": {
        "stateName": "start",
        "variables": [
          {
            "name": "desiredTemperature",
```

```
    "value": "22"
  },
  {
    "name": "averageTemperature",
    "value": "22"
  },
  {
    "name": "allowedError",
    "value": "1.0"
  },
  {
    "name": "rangeHigh",
    "value": "30.0"
  },
  {
    "name": "rangeLow",
    "value": "15.0"
  },
  {
    "name": "anomalousHigh",
    "value": "60.0"
  },
  {
    "name": "anomalousLow",
    "value": "0.0"
  },
  {
    "name": "sensorCount",
    "value": "12"
  },
  {
    "name": "noDelay",
    "value": "true"
  },
  {
    "name": "goodToGo",
    "value": "true"
  },
  {
    "name": "sensorId",
    "value": "0"
  },
  {
    "name": "reportedTemperature",
```

```

        "value": "0.1"
      },
      {
        "name": "resetMe",
        "value": "true"
      }
    ],
    "timers": [
    ]
  }
}
]
}

```

Respons:

```

{
  "message": "An error occurred (InvalidRequestException) when calling the BatchUpdateDetector operation: Number of variables in the detector exceeds the limit 10"
}

```

## AWS IoT Core Aturan mesin dan AWS IoT Events

Aturan berikut menerbitkan ulang pesan AWS IoT Events MQTT sebagai pesan permintaan pembaruan bayangan. Kami berasumsi bahwa AWS IoT Core hal-hal didefinisikan untuk unit pemanas dan unit pendingin untuk setiap area yang dikendalikan oleh model detektor.

Dalam contoh ini, kita telah mendefinisikan hal-hal bernama `Area51HeatingUnit` dan `Area51CoolingUnit`.

Perintah CLI yang digunakan:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowCoolOffRule.json
```

Berkas: `ADMShadowCoolOffRule.json`

```

{
  "ruleName": "ADMShadowCoolOff",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Cooling/Off'",
  }
}

```

```

    "description": "areaDetectorModel mqtt topic publish to cooling unit shadow
request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"
        }
      }
    ]
  }
}

```

Tanggapan: [kosong]

Perintah CLI yang digunakan:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowCoolOnRule.json
```

Berkas: ADMShadowCoolOnRule.json

```

{
  "ruleName": "ADMShadowCoolOn",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Cooling/On'",
    "description": "areaDetectorModel mqtt topic publish to cooling unit shadow
request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"
        }
      }
    ]
  }
}

```

Tanggapan: [kosong]

Perintah CLI yang digunakan:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowHeatOffRule.json
```

Berkas: ADMSHadowHeatOffRule.json

```
{
  "ruleName": "ADMSHadowHeatOff",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Heating/Off'",
    "description": "areaDetectorModel mqtt topic publish to heating unit shadow
request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}HeatingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMSHadowRole"
        }
      }
    ]
  }
}
```

Tanggapan: [kosong]

Perintah CLI yang digunakan:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowHeatOnRule.json
```

Berkas: ADMSHadowHeatOnRule.json

```
{
  "ruleName": "ADMSHadowHeatOn",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Heating/On'",
    "description": "areaDetectorModel mqtt topic publish to heating unit shadow
request",
```

```
"ruleDisabled": false,
"awsIotSqlVersion": "2016-03-23",
"actions": [
  {
    "republish": {
      "topic": "$$aws/things/${payload.detector.keyValue}HeatingUnit/shadow/
update",
      "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"
    }
  }
]
```

Tanggapan: [kosong]

## Contoh: Derek mendeteksi kondisi menggunakan AWS IoT Events

Operator dari banyak crane ingin mendeteksi kapan mesin membutuhkan perawatan atau penggantian dan memicu pemberitahuan yang sesuai. Setiap derek memiliki motor. Motor memancarkan pesan (input) dengan informasi tentang tekanan dan suhu. Operator menginginkan dua tingkat detektor peristiwa:

- Detektor peristiwa tingkat derek
- Detektor peristiwa tingkat motor

Menggunakan pesan dari motor (yang berisi metadata dengan kedua `craneId` dan `motorid`), operator dapat mengeksekusi kedua tingkat detektor peristiwa menggunakan perutean yang sesuai. Ketika kondisi acara terpenuhi, pemberitahuan harus dikirim ke topik Amazon SNS yang sesuai. Operator dapat mengonfigurasi model detektor sehingga pemberitahuan duplikat tidak dinaikkan.

Contoh ini menunjukkan kemampuan fungsional berikut:

- Buat, Baca, Perbarui, Hapus (CRUD) input.
- Buat, Baca, Perbarui, Hapus (CRUD) model detektor peristiwa dan berbagai versi detektor peristiwa.
- Merutekan satu input ke beberapa detektor peristiwa.
- Menelan input ke dalam model detektor.

- Evaluasi kondisi pemicu dan peristiwa siklus hidup.
- Kemampuan untuk merujuk pada variabel keadaan dalam kondisi dan menetapkan nilainya tergantung pada kondisi.
- Orkestrasi runtime dengan definisi, status, evaluator pemicu, dan pelaksana tindakan.
- Eksekusi tindakan `ActionsExecutor` dengan target SNS.

## Kirim perintah sebagai respons terhadap kondisi yang terdeteksi di AWS IoT Events

Halaman ini memberikan contoh untuk menggunakan AWS IoT Events perintah untuk mengatur input, membuat model detektor, dan mengirim data sensor simulasi. Contoh menunjukkan bagaimana memanfaatkan AWS IoT Events untuk memantau peralatan industri seperti motor dan derek.

```
#Create Pressure Input
aws iotevents create-input --cli-input-json file://pressureInput.json
aws iotevents describe-input --input-name PressureInput
aws iotevents update-input --cli-input-json file://pressureInput.json
aws iotevents list-inputs
aws iotevents delete-input --input-name PressureInput

#Create Temperature Input
aws iotevents create-input --cli-input-json file://temperatureInput.json
aws iotevents describe-input --input-name TemperatureInput
aws iotevents update-input --cli-input-json file://temperatureInput.json
aws iotevents list-inputs
aws iotevents delete-input --input-name TemperatureInput

#Create Motor Event Detector using pressure and temperature input
aws iotevents create-detector-model --cli-input-json file://motorDetectorModel.json
aws iotevents describe-detector-model --detector-model-name motorDetectorModel
aws iotevents update-detector-model --cli-input-json file://
updateMotorDetectorModel.json
aws iotevents list-detector-models
aws iotevents list-detector-model-versions --detector-model-name motorDetectorModel
aws iotevents delete-detector-model --detector-model-name motorDetectorModel

#Create Crane Event Detector using temperature input
aws iotevents create-detector-model --cli-input-json file://craneDetectorModel.json
aws iotevents describe-detector-model --detector-model-name craneDetectorModel
```

```
aws iotevents update-detector-model --cli-input-json file://
updateCraneDetectorModel.json
aws iotevents list-detector-models
aws iotevents list-detector-model-versions --detector-model-name craneDetectorModel
aws iotevents delete-detector-model --detector-model-name craneDetectorModel

#Replace craneIds
sed -i ' ' "s/100008/100009/g" messages/*

#Replace motorIds
sed -i ' ' "s/200008/200009/g" messages/*

#Send HighPressure message
aws iotevents-data batch-put-message --cli-input-json file://messages/
highPressureMessage.json --cli-binary-format raw-in-base64-out

#Send HighTemperature message
aws iotevents-data batch-put-message --cli-input-json file://messages/
highTemperatureMessage.json --cli-binary-format raw-in-base64-out

#Send LowPressure message
aws iotevents-data batch-put-message --cli-input-json file://messages/
lowPressureMessage.json --cli-binary-format raw-in-base64-out

#Send LowTemperature message
aws iotevents-data batch-put-message --cli-input-json file://messages/
lowTemperatureMessage.json --cli-binary-format raw-in-base64-out
```

## Model AWS IoT Events detektor untuk pemantauan derek

Pantau peralatan atau armada perangkat Anda untuk kegagalan atau perubahan dalam operasi, dan memicu tindakan ketika peristiwa tersebut terjadi. Anda menentukan model detektor di JSON yang menentukan status, aturan, dan tindakan. Ini memungkinkan Anda untuk memantau input seperti suhu dan tekanan, melacak pelanggaran ambang batas, dan mengirim peringatan. Contoh menunjukkan model detektor untuk derek dan motor, mendeteksi masalah panas berlebih dan memberi tahu oleh Amazon SNS ketika ambang batas terlampaui. Anda dapat memperbaiki model untuk memperbaiki perilaku tanpa mengganggu pemantauan.

Berkas: craneDetectorModel.json

```
{
```

```

"detectorModelName": "craneDetectorModel",
"detectorModelDefinition": {
  "states": [
    {
      "stateName": "Running",
      "onEnter": {
        "events": [
          {
            "eventName": "init",
            "condition": "true",
            "actions": [
              {
                "setVariable": {
                  "variableName": "craneThresholdBreach",
                  "value": "0"
                }
              }
            ]
          }
        ]
      },
      "onInput": {
        "events": [
          {
            "eventName": "Overheated",
            "condition": "$input.TemperatureInput.temperature > 35",
            "actions": [
              {
                "setVariable": {
                  "variableName": "craneThresholdBreach",
                  "value": "$variable.craneThresholdBreach + 1"
                }
              }
            ]
          },
          {
            "eventName": "Crane Threshold Breached",
            "condition": "$variable.craneThresholdBreach > 5",
            "actions": [
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-
east-1:123456789012:CraneSNSTopic"
                }
              }
            ]
          }
        ]
      }
    }
  ]
}

```

```

        }
      ]
    },
    {
      "eventName": "Underheated",
      "condition": "$input.TemperatureInput.temperature < 25",
      "actions": [
        {
          "setVariable": {
            "variableName": "craneThresholdBreached",
            "value": "0"
          }
        }
      ]
    }
  ]
},
"initialStateName": "Running"
},
"key": "craneid",
"roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}

```

Untuk memperbarui model detektor yang ada. Berkas: `updateCraneDetectorModel.json`

```

{
  "detectorModelName": "craneDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Running",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "craneThresholdBreached",
                    "value": "0"
                  }
                }
              ]
            }
          ]
        }
      }
    ]
  }
}

```

```

    }
  },
  {
    "setVariable": {
      "variableName": "alarmRaised",
      "value": "'false'"
    }
  }
]
},
"onInput": {
  "events": [
    {
      "eventName": "Overheated",
      "condition": "$input.TemperatureInput.temperature > 30",
      "actions": [
        {
          "setVariable": {
            "variableName": "craneThresholdBreach",
            "value": "$variable.craneThresholdBreach + 1"
          }
        }
      ]
    },
    {
      "eventName": "Crane Threshold Breached",
      "condition": "$variable.craneThresholdBreach > 5 &&
$variable.alarmRaised == 'false'",
      "actions": [
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-
east-1:123456789012:CraneSNSTopic"
          }
        }
      ],
      "setVariable": {
        "variableName": "alarmRaised",
        "value": "'true'"
      }
    }
  ]
}
]

```

```

        },
        {
            "eventName": "Underheated",
            "condition": "$input.TemperatureInput.temperature < 10",
            "actions": [
                {
                    "setVariable": {
                        "variableName": "craneThresholdBreach",
                        "value": "0"
                    }
                }
            ]
        }
    ],
    "initialStateName": "Running"
},
"roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}

```

Berkas: motorDetectorModel.json

```

{
  "detectorModelName": "motorDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Running",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "motorThresholdBreach",
                    "value": "0"
                  }
                }
              ]
            }
          ]
        }
      }
    ]
  }
}

```

```

        }
      ]
    },
    "onInput": {
      "events": [
        {
          "eventName": "Overheated And Overpressurized",
          "condition": "$input.PressureInput.pressure > 70 &&
$input.TemperatureInput.temperature > 30",
          "actions": [
            {
              "setVariable": {
                "variableName": "motorThresholdBreach",
                "value": "$variable.motorThresholdBreach + 1"
              }
            }
          ]
        },
        {
          "eventName": "Motor Threshold Breached",
          "condition": "$variable.motorThresholdBreach > 5",
          "actions": [
            {
              "sns": {
                "targetArn": "arn:aws:sns:us-
east-1:123456789012:MotorSNSTopic"
              }
            }
          ]
        }
      ]
    }
  ],
  "initialStateName": "Running"
},
"key": "motorId",
"roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}

```

Untuk memperbarui model detektor yang ada. Berkas: `updateMotorDetectorModel.json`

```
{
```

```

"detectorModelName": "motorDetectorModel",
"detectorModelDefinition": {
  "states": [
    {
      "stateName": "Running",
      "onEnter": {
        "events": [
          {
            "eventName": "init",
            "condition": "true",
            "actions": [
              {
                "setVariable": {
                  "variableName": "motorThresholdBreached",
                  "value": "0"
                }
              }
            ]
          }
        ],
      },
      "onInput": {
        "events": [
          {
            "eventName": "Overheated And Overpressurized",
            "condition": "$input.PressureInput.pressure > 70 &&
$input.TemperatureInput.temperature > 30",
            "actions": [
              {
                "setVariable": {
                  "variableName": "motorThresholdBreached",
                  "value": "$variable.motorThresholdBreached + 1"
                }
              }
            ]
          }
        ],
      },
      {
        "eventName": "Motor Threshold Breached",
        "condition": "$variable.motorThresholdBreached > 5",
        "actions": [
          {
            "sns": {
              "targetArn": "arn:aws:sns:us-
east-1:123456789012:MotorSNSTopic"
            }
          }
        ]
      }
    ]
  }
}

```

```
    ],
    "initialStateName": "Running"
  },
  "roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}
```

## AWS IoT Events input untuk pemantauan derek

Dalam contoh ini, kami mendemonstrasikan cara mengatur input untuk menggunakan AWS IoT Events sistem pemantauan derek. Ini menangkap input tekanan dan suhu untuk menggambarkan bagaimana menyusun input untuk pemantauan peralatan industri yang kompleks.

Berkas: `pressureInput.json`

```
{
  "inputName": "PressureInput",
  "inputDescription": "this is a pressure input description",
  "inputDefinition": {
    "attributes": [
      {"jsonPath": "pressure"}
    ]
  }
}
```

Berkas: `temperatureInput.json`

```
{
  "inputName": "TemperatureInput",
  "inputDescription": "this is temperature input description",
  "inputDefinition": {
    "attributes": [
      {"jsonPath": "temperature"}
    ]
  }
}
```

```
}
```

## Kirim alarm dan pesan operasional dengan AWS IoT Events

Penanganan pesan yang efektif penting dalam sistem pemantauan derek. Bagian ini menampilkan cara mengkonfigurasi AWS IoT Events untuk memproses dan menanggapi berbagai jenis pesan dari sensor derek. Menyiapkan alarm berdasarkan pesan tertentu dapat membantu Anda mengurai, memfilter, dan merutekan pembaruan status untuk memicu tindakan yang sesuai.

Berkas: `highPressureMessage.json`

```
{
  "messages": [
    {
      "messageId": "1",
      "inputName": "PressureInput",
      "payload": "{\"craneid\": \"100009\", \"pressure\": 80, \"motorid\": \"200009\"}"
    }
  ]
}
```

Berkas: `highTemperatureMessage.json`

```
{
  "messages": [
    {
      "messageId": "2",
      "inputName": "TemperatureInput",
      "payload": "{\"craneid\": \"100009\", \"temperature\": 40, \"motorid\": \"200009\"}"
    }
  ]
}
```

Berkas: `lowPressureMessage.json`

```
{
  "messages": [
    {
```

```

        "messageId": "1",
        "inputName": "PressureInput",
        "payload": "{\"craneid\": \"100009\", \"pressure\": 20, \"motorid\":
        \"200009\"}"
    }
]
}

```

Berkas: lowTemperatureMessage.json

```

{
  "messages": [
    {
      "messageId": "2",
      "inputName": "TemperatureInput",
      "payload": "{\"craneid\": \"100009\", \"temperature\": 20, \"motorid\":
      \"200009\"}"
    }
  ]
}

```

## Contoh: deteksi AWS IoT Events peristiwa dengan sensor dan aplikasi

Model detektor ini adalah salah satu templat yang tersedia dari AWS IoT Events konsol. Ini termasuk di sini untuk kenyamanan Anda.

Contoh ini menunjukkan AWS IoT Events deteksi peristiwa aplikasi menggunakan data sensor. Ini menunjukkan bagaimana Anda dapat membuat model detektor yang memantau peristiwa tertentu sehingga Anda dapat memicu tindakan yang sesuai. Anda dapat membuat beberapa input sensor, menentukan kondisi peristiwa yang kompleks, dan menyiapkan mekanisme respons bertingkat.

```

{
  "detectorModelName": "EventDetectionSensorsAndApplications",
  "detectorModelDefinition": {
    "states": [
      {
        "onInput": {
          "transitionEvents": [],
          "events": []
        },
      },
    ],
  },
}

```

```

    "stateName": "Device_exception",
    "onEnter": {
      "events": [
        {
          "eventName": "Send_mqtt",
          "actions": [
            {
              "iotTopicPublish": {
                "mqttTopic": "Device_stolen"
              }
            }
          ],
          "condition": "true"
        }
      ]
    },
    "onExit": {
      "events": []
    }
  },
  {
    "onInput": {
      "transitionEvents": [
        {
          "eventName": "To_in_use",
          "actions": [],
          "condition": "$variable.position !=
$input.AWS_IoTEvents_Blueprints_Tracking_DeviceInput.gps_position",
          "nextState": "Device_in_use"
        }
      ],
      "events": []
    },
    "stateName": "Device_idle",
    "onEnter": {
      "events": [
        {
          "eventName": "Set_position",
          "actions": [
            {
              "setVariable": {
                "variableName": "position",
                "value":
"$input.AWS_IoTEvents_Blueprints_Tracking_DeviceInput.gps_position"

```

```

        }
    },
    ],
    "condition": "true"
}
],
},
"onExit": {
    "events": []
}
},
{
    "onInput": {
        "transitionEvents": [
            {
                "eventName": "To_exception",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_Tracking_UserInput.device_id !=
$input.AWS_IoTEvents_Blueprints_Tracking_DeviceInput.device_id",
                "nextState": "Device_exception"
            }
        ],
        "events": []
    },
    "stateName": "Device_in_use",
    "onEnter": {
        "events": []
    },
    "onExit": {
        "events": []
    }
}
],
"initialStateName": "Device_idle"
}
}

```

## Contoh: Perangkat HeartBeat untuk memantau koneksi perangkat dengan AWS IoT Events

Model detektor ini adalah salah satu templat yang tersedia dari AWS IoT Events konsol. Ini termasuk di sini untuk kenyamanan Anda.

Contoh Defective Heart Beat (DHB) menggambarkan bagaimana AWS IoT Events dapat digunakan dalam pemantauan perawatan kesehatan. Contoh ini menunjukkan bagaimana Anda dapat membuat model detektor yang menganalisis data detak jantung, mendeteksi pola tidak teratur, dan memicu respons yang sesuai. Belajarlah untuk mengatur input, menentukan ambang batas, dan mengonfigurasi peringatan untuk potensi masalah jantung, menampilkan keserbagunaan dalam aplikasi perawatan AWS IoT Events kesehatan terkait.

```
{
  "detectorModelDefinition": {
    "states": [
      {
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "To_normal",
              "actions": [],
              "condition":
"currentInput(\"AWS_IoTEvents_Blueprints_Heartbeat_Input\")",
              "nextState": "Normal"
            }
          ],
          "events": []
        },
        "stateName": "Offline",
        "onEnter": {
          "events": [
            {
              "eventName": "Send_notification",
              "actions": [
                {
                  "sns": {
                    "targetArn": "sns-topic-arn"
                  }
                }
              ]
            }
          ]
        }
      }
    ],
  },
}
```

```

        "condition": "true"
      }
    ]
  },
  "onExit": {
    "events": []
  }
},
{
  "onInput": {
    "transitionEvents": [
      {
        "eventName": "Go_offline",
        "actions": [],
        "condition": "timeout(\"awake\")",
        "nextState": "Offline"
      }
    ],
    "events": [
      {
        "eventName": "Reset_timer",
        "actions": [
          {
            "resetTimer": {
              "timerName": "awake"
            }
          }
        ],
        "condition":
"currentInput(\"AWS_IoTEvents_Blueprints_Heartbeat_Input\")"
      }
    ]
  },
  "stateName": "Normal",
  "onEnter": {
    "events": [
      {
        "eventName": "Create_timer",
        "actions": [
          {
            "setTimer": {
              "seconds": 300,
              "timerName": "awake"
            }
          }
        ]
      }
    ]
  }
}

```

```

        }
        ],
        "condition":
"$input.AWS_IoTEvents_Blueprints_Heartbeat_Input.value > 0"
    }
  ]
},
"onExit": {
  "events": []
}
},
],
"initialStateName": "Normal"
}
}

```

## Contoh: Alarm ISA di AWS IoT Events

Model detektor ini adalah salah satu templat yang tersedia dari AWS IoT Events konsol. Ini termasuk di sini untuk kenyamanan Anda.

```

{
  "detectorModelName": "AWS_IoTEvents_Blueprints_ISA_Alarm",
  "detectorModelDefinition": {
    "states": [
      {
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "unshelve",
              "actions": [],
              "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unshelve\" &&
$variable.state == \"rtnunack\"",
              "nextState": "RTN_Unacknowledged"
            },
            {
              "eventName": "unshelve",
              "actions": [],
              "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unshelve\" &&
$variable.state == \"ack\"",

```

```

        "nextState": "Acknowledged"
    },
    {
        "eventName": "unshelve",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unshelve\" &&
$variable.state == \"unack\"",
        "nextState": "Unacknowledged"
    },
    {
        "eventName": "unshelve",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unshelve\" &&
$variable.state == \"normal\"",
        "nextState": "Normal"
    }
],
"events": []
},
"stateName": "Shelved",
"onEnter": {
    "events": []
},
"onExit": {
    "events": []
}
},
{
    "onInput": {
        "transitionEvents": [
            {
                "eventName": "abnormal_condition",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value > $variable.higher_threshold ||
$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value < $variable.lower_threshold",
                "nextState": "Unacknowledged"
            },
            {
                "eventName": "acknowledge",
                "actions": [],

```

```

        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"acknowledge\"",
        "nextState": "Normal"
    },
    {
        "eventName": "shelve",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"shelve\"",
        "nextState": "Shelved"
    },
    {
        "eventName": "remove_from_service",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"remove\"",
        "nextState": "Out_of_service"
    },
    {
        "eventName": "suppression",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"suppressed\"",
        "nextState": "Suppressed_by_design"
    }
],
"events": []
},
"stateName": "RTN_Unacknowledged",
"onEnter": {
    "events": [
        {
            "eventName": "State Save",
            "actions": [
                {
                    "setVariable": {
                        "variableName": "state",
                        "value": "\"rtnunack\""
                    }
                }
            ]
        },
        {
            "condition": "true"
        }
    ]
}
]

```

```

    },
    "onExit": {
      "events": []
    }
  },
  {
    "onInput": {
      "transitionEvents": [
        {
          "eventName": "abnormal_condition",
          "actions": [],
          "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value > $variable.higher_threshold ||
$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value < $variable.lower_threshold",
          "nextState": "Unacknowledged"
        },
        {
          "eventName": "shelve",
          "actions": [],
          "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"shelve\"",
          "nextState": "Shelved"
        },
        {
          "eventName": "remove_from_service",
          "actions": [],
          "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"remove\"",
          "nextState": "Out_of_service"
        },
        {
          "eventName": "suppression",
          "actions": [],
          "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"suppressed\"",
          "nextState": "Suppressed_by_design"
        }
      ],
      "events": [
        {
          "eventName": "Create Config variables",
          "actions": [
            {
              "setVariable": {

```

```

                "variableName": "lower_threshold",
                "value":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.lower_threshold"
            }
        },
        {
            "setVariable": {
                "variableName": "higher_threshold",
                "value":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.higher_threshold"
            }
        }
    ],
    "condition": "$variable.lower_threshold !=
$variable.lower_threshold"
}
]
},
"stateName": "Normal",
"onEnter": {
    "events": [
        {
            "eventName": "State Save",
            "actions": [
                {
                    "setVariable": {
                        "variableName": "state",
                        "value": "\"normal\""
                    }
                }
            ]
        }
    ],
    "condition": "true"
}
]
},
"onExit": {
    "events": []
}
},
{
    "onInput": {
        "transitionEvents": [
            {
                "eventName": "acknowledge",

```

```

        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"acknowledge\"",
        "nextState": "Acknowledged"
    },
    {
        "eventName": "return_to_normal",
        "actions": [],
        "condition":
"($input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value <= $variable.higher_threshold
&& $input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value >=
$variable.lower_threshold)",
        "nextState": "RTN_Unacknowledged"
    },
    {
        "eventName": "shelve",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"shelve\"",
        "nextState": "Shelved"
    },
    {
        "eventName": "remove_from_service",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"remove\"",
        "nextState": "Out_of_service"
    },
    {
        "eventName": "suppression",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"suppressed\"",
        "nextState": "Suppressed_by_design"
    }
],
"events": []
},
"stateName": "Unacknowledged",
"onEnter": {
    "events": [
        {
            "eventName": "State Save",
            "actions": [

```

```

        {
            "setVariable": {
                "variableName": "state",
                "value": "\"unack\""
            }
        }
    ],
    "condition": "true"
}
]
},
"onExit": {
    "events": []
}
},
{
    "onInput": {
        "transitionEvents": [
            {
                "eventName": "unsuppression",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unsuppressed\" &&
$variable.state == \"normal\"",
                "nextState": "Normal"
            },
            {
                "eventName": "unsuppression",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unsuppressed\" &&
$variable.state == \"unack\"",
                "nextState": "Unacknowledged"
            },
            {
                "eventName": "unsuppression",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unsuppressed\" &&
$variable.state == \"ack\"",
                "nextState": "Acknowledged"
            },
            {
                "eventName": "unsuppression",

```

```

        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unsuppressed\" &&
$variable.state == \"rtnunack\"",
        "nextState": "RTN_Unacknowledged"
    }
  ],
  "events": []
},
"stateName": "Suppressed_by_design",
"onEnter": {
  "events": []
},
"onExit": {
  "events": []
}
},
{
  "onInput": {
    "transitionEvents": [
      {
        "eventName": "return_to_service",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"add\" && $variable.state
== \"rtnunack\"",
        "nextState": "RTN_Unacknowledged"
      },
      {
        "eventName": "return_to_service",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"add\" && $variable.state
== \"unack\"",
        "nextState": "Unacknowledged"
      },
      {
        "eventName": "return_to_service",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"add\" && $variable.state
== \"ack\"",
        "nextState": "Acknowledged"
      }
    ]
  }
}

```

```

        {
            "eventName": "return_to_service",
            "actions": [],
            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"add\" && $variable.state
== \"normal\"",
            "nextState": "Normal"
        }
    ],
    "events": []
},
"stateName": "Out_of_service",
"onEnter": {
    "events": []
},
"onExit": {
    "events": []
}
},
{
    "onInput": {
        "transitionEvents": [
            {
                "eventName": "re-alarm",
                "actions": [],
                "condition": "timeout(\"snooze\")",
                "nextState": "Unacknowledged"
            },
            {
                "eventName": "return_to_normal",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"reset\"",
                "nextState": "Normal"
            },
            {
                "eventName": "shelve",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"shelve\"",
                "nextState": "Shelved"
            },
            {
                "eventName": "remove_from_service",

```

```

        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"remove\"",
        "nextState": "Out_of_service"
    },
    {
        "eventName": "suppression",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"suppressed\"",
        "nextState": "Suppressed_by_design"
    }
],
"events": []
},
"stateName": "Acknowledged",
"onEnter": {
    "events": [
        {
            "eventName": "Create Timer",
            "actions": [
                {
                    "setTimer": {
                        "seconds": 60,
                        "timerName": "snooze"
                    }
                }
            ],
            "condition": "true"
        },
        {
            "eventName": "State Save",
            "actions": [
                {
                    "setVariable": {
                        "variableName": "state",
                        "value": "\"ack\""
                    }
                }
            ],
            "condition": "true"
        }
    ]
},

```

```

        "onExit": {
            "events": []
        }
    },
    ],
    "initialStateName": "Normal"
},
"detectorModelDescription": "This detector model is used to detect if a monitored
device is in an Alarming State in accordance to the ISA 18.2.",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
"key": "alarmId"
}

```

## Contoh: Membangun alarm sederhana dengan AWS IoT Events

Model detektor ini adalah salah satu templat yang tersedia dari AWS IoT Events konsol. Ini termasuk di sini untuk kenyamanan Anda.

```

{
  "detectorModelDefinition": {
    "states": [
      {
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "not_fixed",
              "actions": [],
              "condition": "timeout(\"snoozeTime\")",
              "nextState": "Alarming"
            },
            {
              "eventName": "reset",
              "actions": [],
              "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.command == \"reset\"",
              "nextState": "Normal"
            }
          ],
          "events": [
            {
              "eventName": "DND",
              "actions": [

```

```

        {
            "setVariable": {
                "variableName": "dnd_active",
                "value": "1"
            }
        }
    ],
    "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.command == \"dnd\""
    }
]
},
"stateName": "Snooze",
"onEnter": {
    "events": [
        {
            "eventName": "Create Timer",
            "actions": [
                {
                    "setTimer": {
                        "seconds": 120,
                        "timerName": "snoozeTime"
                    }
                }
            ],
            "condition": "true"
        }
    ]
},
"onExit": {
    "events": []
}
},
{
    "onInput": {
        "transitionEvents": [
            {
                "eventName": "out_of_range",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.value > $variable.threshold",
                "nextState": "Alarming"
            }
        ]
    },

```

```
        "events": [
          {
            "eventName": "Create Config variables",
            "actions": [
              {
                "setVariable": {
                  "variableName": "threshold",
                  "value":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.threshold"
                }
              }
            ],
            "condition": "$variable.threshold != $variable.threshold"
          }
        ]
      },
      "stateName": "Normal",
      "onEnter": {
        "events": [
          {
            "eventName": "Init",
            "actions": [
              {
                "setVariable": {
                  "variableName": "dnd_active",
                  "value": "0"
                }
              }
            ],
            "condition": "true"
          }
        ]
      },
      "onExit": {
        "events": []
      }
    },
    {
      "onInput": {
        "transitionEvents": [
          {
            "eventName": "reset",
            "actions": [],

```

```

        "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.command == \"reset\"",
        "nextState": "Normal"
    },
    {
        "eventName": "acknowledge",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.command == \"acknowledge\"",
        "nextState": "Snooze"
    }
],
"events": [
    {
        "eventName": "Escalated Alarm Notification",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-
west-2:123456789012:escalatedAlarmNotification"
                }
            }
        ],
        "condition": "timeout(\"unacknowledgeTime\")"
    }
]
},
"stateName": "Alarming",
"onEnter": {
    "events": [
        {
            "eventName": "Alarm Notification",
            "actions": [
                {
                    "sns": {
                        "targetArn": "arn:aws:sns:us-
west-2:123456789012:alarmNotification"
                    }
                },
                {
                    "setTimer": {
                        "seconds": 300,
                        "timerName": "unacknowledgeTime"
                    }
                }
            ]
        }
    ]
}

```

```
        }
      ],
      "condition": "$variable.dnd_active != 1"
    }
  ]
},
"onExit": {
  "events": []
}
],
"initialStateName": "Normal"
},
"detectorModelDescription": "This detector model is used to detect if a monitored
device is in an Alarming State.",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
"key": "alarmId"
}
```

# Pemantauan dengan alarm di AWS IoT Events

AWS IoT Events alarm membantu Anda memantau data Anda untuk perubahan. Data dapat berupa metrik yang Anda ukur untuk peralatan dan proses Anda. Anda dapat membuat alarm yang mengirim notifikasi saat ambang batas dilanggar. Alarm membantu Anda mendeteksi masalah, merampingkan pemeliharaan, dan mengoptimalkan kinerja peralatan dan proses Anda.

Alarm adalah contoh model alarm. Model alarm menentukan apa yang harus dideteksi, kapan harus mengirim pemberitahuan, siapa yang mendapat pemberitahuan, dan banyak lagi. Anda juga dapat menentukan satu atau beberapa [tindakan yang didukung](#) yang terjadi ketika status alarm berubah. AWS IoT Events merutekan [atribut input](#) yang berasal dari data Anda ke alarm yang sesuai. Jika data yang Anda pantau berada di luar rentang yang ditentukan, alarm akan dipanggil. Anda juga dapat mengenali alarm atau mengaturnya ke mode tunda.

## Bekerja dengan AWS IoT SiteWise

Anda dapat menggunakan AWS IoT Events alarm untuk memantau properti aset di AWS IoT SiteWise. AWS IoT SiteWise mengirimkan nilai properti aset ke AWS IoT Events alarm. AWS IoT Events mengirimkan status alarm ke AWS IoT SiteWise.

AWS IoT SiteWise juga mendukung alarm eksternal. Anda dapat memilih alarm eksternal jika Anda menggunakan alarm di luar AWS IoT SiteWise dan memiliki solusi yang mengembalikan data status alarm. Alarm eksternal berisi properti pengukuran yang menelan data status alarm.

AWS IoT SiteWise tidak mengevaluasi keadaan alarm eksternal. Selain itu, Anda tidak dapat mengakui atau menunda alarm eksternal saat status alarm berubah.

Anda dapat menggunakan fitur SiteWise Monitor untuk melihat status alarm eksternal di portal SiteWise Monitor.

Untuk informasi selengkapnya, lihat [Memantau data dengan alarm](#) di Panduan AWS IoT SiteWise Pengguna dan [Pemantauan dengan alarm](#) di Panduan Aplikasi SiteWise Monitor.

## Akui aliran

Saat membuat model alarm, Anda memilih apakah akan mengaktifkan alur pengakuan. Jika Anda mengaktifkan alur pengakuan, tim Anda akan diberi tahu saat status alarm berubah. Tim Anda dapat mengenali alarm dan meninggalkan catatan. Misalnya, Anda dapat menyertakan informasi alarm

dan tindakan yang akan Anda ambil untuk mengatasi masalah tersebut. Jika data yang Anda pantau berada di luar rentang yang ditentukan, alarm akan dipanggil.

Alarm memiliki status berikut:

#### DISABLED

Ketika alarm dalam DISABLED keadaan, itu tidak siap untuk mengevaluasi data. Untuk mengaktifkan alarm, Anda harus mengubah alarm ke NORMAL negara.

#### NORMAL

Ketika alarm dalam NORMAL keadaan, itu siap untuk mengevaluasi data.

#### ACTIVE

Jika alarm dalam ACTIVE keadaan, alarm dipanggil. Data yang Anda pantau berada di luar rentang yang ditentukan.

#### ACKNOWLEDGED

Ketika alarm dalam ACKNOWLEDGED keadaan, alarm dipanggil dan Anda mengakui alarm.

#### LATCHED

Alarm dipanggil, tetapi Anda tidak mengakui alarm setelah jangka waktu tertentu. Alarm secara otomatis berubah ke NORMAL status.

#### SNOOZE\_DISABLED

Ketika alarm dalam SNOOZE\_DISABLED keadaan, alarm dinonaktifkan untuk jangka waktu tertentu. Setelah waktu tunda, alarm secara otomatis berubah ke status. NORMAL

## Membuat model alarm di AWS IoT Events

Anda dapat menggunakan AWS IoT Events alarm untuk memantau data Anda dan mendapatkan pemberitahuan ketika ambang batas dilanggar. Alarm menyediakan parameter yang Anda gunakan untuk membuat atau mengonfigurasi model alarm. Anda dapat menggunakan AWS IoT Events konsol atau AWS IoT Events API untuk membuat atau mengonfigurasi model alarm. Saat Anda mengonfigurasi model alarm, perubahan berlaku saat data baru tiba.

## Persyaratan

Persyaratan berikut berlaku saat Anda membuat model alarm.

- Anda dapat membuat model alarm untuk memantau atribut input di AWS IoT Events atau properti aset di AWS IoT SiteWise.
  - Jika Anda memilih untuk memantau atribut input di AWS IoT Events, [Buat masukan untuk model di AWS IoT Events](#) sebelum Anda membuat model alarm.
  - Jika Anda memilih untuk memantau properti aset, Anda harus [membuat model aset](#) AWS IoT SiteWise sebelum membuat model alarm.
- Anda harus memiliki peran IAM yang memungkinkan alarm Anda melakukan tindakan dan mengakses AWS sumber daya. Untuk informasi selengkapnya, lihat [Menyiapkan izin untuk AWS IoT Events](#).
- Semua AWS sumber daya yang digunakan tutorial ini harus berada di AWS Wilayah yang sama.

## Membuat model alarm (konsol)

Berikut ini menunjukkan cara membuat model alarm untuk memantau AWS IoT Events atribut di AWS IoT Events konsol.

1. Masuk ke [konsol AWS IoT Events](#) tersebut.
2. Di panel navigasi, pilih Model alarm.
3. Pada halaman Model alarm, pilih Buat model alarm.
4. Di bagian Detail model alarm, lakukan hal berikut:
  - a. Masukkan nama yang unik.
  - b. (Opsional) Masukkan deskripsi.
5. Di bagian Target alarm, lakukan hal berikut:

### Important

Jika Anda memilih properti AWS IoT SiteWise aset, Anda harus telah membuat model aset di AWS IoT SiteWise.

- a. Pilih atribut AWS IoT Events input.
- b. Pilih input.
- c. Pilih kunci atribut input. Atribut input ini digunakan sebagai kunci untuk membuat alarm. AWS IoT Events rute input yang terkait dengan kunci ini ke alarm.

**⚠ Important**

Jika payload pesan input tidak berisi kunci atribut input ini, atau jika kunci tidak berada di jalur JSON yang sama yang ditentukan dalam kunci, maka pesan akan gagal dalam konsumsi. AWS IoT Events

6. Di bagian Definisi Threshold, Anda menentukan atribut input, nilai ambang batas, dan operator perbandingan yang AWS IoT Events digunakan untuk mengubah status alarm.

a. Untuk atribut Input, pilih atribut yang ingin Anda pantau.

Setiap kali atribut input ini menerima data baru, itu dievaluasi untuk menentukan status alarm.

b. Untuk Operator, pilih operator perbandingan. Operator membandingkan atribut input Anda dengan nilai ambang untuk atribut Anda.

Anda dapat memilih dari opsi ini:

- > lebih besar dari
- >= lebih besar dari atau sama dengan
- < kurang dari
- <= kurang dari atau sama dengan
- = sama dengan
- != tidak sama dengan

c. Untuk Nilai ambang batas, masukkan angka atau pilih atribut dalam AWS IoT Events input. AWS IoT Events membandingkan nilai ini dengan nilai atribut input yang Anda pilih.

d. (Opsional) Untuk Tingkat Keparahan, Gunakan nomor yang dipahami tim Anda untuk mencerminkan tingkat keparahan alarm ini.

7. (Opsional) Di bagian Pengaturan pemberitahuan, konfigurasi pengaturan notifikasi untuk alarm.

Anda dapat menambahkan hingga 10 notifikasi. Untuk Notifikasi 1, lakukan hal berikut:

a. Untuk Protokol, pilih dari opsi berikut:

- Email & teks - Alarm mengirimkan pemberitahuan SMS dan pemberitahuan email.

- Email - Alarm mengirimkan pemberitahuan email.
  - Teks - Alarm mengirimkan pemberitahuan SMS.
- b. Untuk Pengirim, tentukan alamat email yang dapat mengirim pemberitahuan tentang alarm ini.

Untuk menambahkan lebih banyak alamat email ke daftar pengirim, pilih Tambahkan pengirim.

- c. (Opsional) Untuk Penerima, pilih penerima.

Untuk menambahkan lebih banyak pengguna ke daftar penerima, pilih Tambahkan pengguna baru. Anda harus menambahkan pengguna baru ke toko Pusat Identitas IAM Anda sebelum Anda dapat menambahkannya ke model alarm Anda. Untuk informasi selengkapnya, lihat [Mengelola akses IAM Identity Center penerima alarm di AWS IoT Events](#).

- d. (Opsional) Untuk pesan kustom tambahan, masukkan pesan yang menjelaskan apa yang dideteksi alarm dan tindakan apa yang harus dilakukan penerima.
8. Di bagian Instans, Anda dapat mengaktifkan atau menonaktifkan semua instance alarm yang dibuat berdasarkan model alarm ini.
9. Di bagian Pengaturan lanjutan, lakukan hal berikut:
- a. Untuk alur Akui, Anda dapat mengaktifkan atau menonaktifkan notifikasi.
- Jika Anda memilih Diaktifkan, Anda menerima pemberitahuan saat status alarm berubah. Anda harus mengakui pemberitahuan sebelum status alarm dapat kembali normal.
  - Jika Anda memilih Dinonaktifkan, tidak ada tindakan yang diperlukan. Alarm secara otomatis berubah ke keadaan normal ketika pengukuran kembali ke kisaran yang ditentukan.

Untuk informasi selengkapnya, lihat [Akui aliran](#).

- b. Untuk Izin, pilih salah satu opsi berikut:

- Anda dapat Membuat peran baru dari templat AWS kebijakan dan AWS IoT Events secara otomatis membuat peran IAM untuk Anda.
- Anda dapat Menggunakan peran IAM yang ada yang memungkinkan model alarm ini melakukan tindakan dan mengakses AWS sumber daya lainnya.

Untuk informasi selengkapnya, lihat [Identitas dan manajemen akses untuk AWS IoT Events](#).

- c. Untuk pengaturan pemberitahuan tambahan, Anda dapat mengedit AWS Lambda fungsi Anda untuk mengelola pemberitahuan alarm. Pilih salah satu opsi berikut untuk AWS Lambda fungsi Anda:

- Buat AWS Lambda fungsi baru - AWS IoT Events membuat AWS Lambda fungsi baru untuk Anda.
- Gunakan AWS Lambda fungsi yang ada - Gunakan AWS Lambda fungsi yang ada dengan memilih nama AWS Lambda fungsi.

Untuk informasi lebih lanjut tentang kemungkinan tindakan, lihat [AWS IoT Events bekerja dengan AWS layanan lain](#).

- d. (Opsional) Untuk tindakan Atur status, Anda dapat menambahkan satu atau beberapa AWS IoT Events tindakan yang akan diambil saat status alarm berubah.
10. (Opsional) Anda dapat menambahkan Tag untuk mengelola alarm Anda. Untuk informasi selengkapnya, lihat [Menandai sumber daya AWS IoT Events](#).
11. Pilih Buat.

## Menanggapi alarm di AWS IoT Events

Menanggapi alarm secara efektif merupakan aspek penting dalam mengelola sistem IoT dengan AWS IoT Events Jelajahi berbagai cara untuk mengonfigurasi dan menangani alarm, termasuk: menyiapkan saluran notifikasi, menentukan prosedur eskalasi, dan menerapkan tindakan respons otomatis. Pelajari cara membuat kondisi alarm bernuansa, memprioritaskan peringatan, dan berintegrasi dengan AWS layanan lain untuk membangun sistem manajemen alarm responsif untuk aplikasi IoT Anda.

Jika Anda mengaktifkan [alur pengakuan](#), Anda akan menerima pemberitahuan saat status alarm berubah. Untuk menanggapi alarm, Anda dapat mengakui, menonaktifkan, mengaktifkan, mengatur ulang, atau menunda alarm.

### Console

Berikut ini menunjukkan kepada Anda cara merespons alarm di AWS IoT Events konsol.

1. Masuk ke [konsol AWS IoT Events](#) tersebut.
2. Di panel navigasi, pilih Model alarm.
3. Pilih model alarm target.
4. Di bagian Daftar alarm, pilih alarm target.
5. Anda dapat memilih salah satu opsi berikut dari Tindakan:
  - Akui - Alarm berubah ke ACKNOWLEDGED negara.
  - Nonaktifkan - Alarm berubah ke DISABLED negara bagian.
  - Aktifkan - Alarm berubah ke NORMAL negara bagian.
  - Reset - Alarm berubah ke NORMAL status.
  - Tunda, lalu lakukan hal berikut:
    1. Pilih panjang Tunda atau masukkan panjang Tunda khusus.
    2. Pilih Simpan.

Alarm berubah ke SNOOZE\_DISABLED negara

Untuk informasi lebih lanjut tentang status alarm, lihat [Akui aliran](#).

## API

Untuk merespons satu atau beberapa alarm, Anda dapat menggunakan operasi AWS IoT Events API berikut:

- [BatchAcknowledgeAlarm](#)
- [BatchDisableAlarm](#)
- [BatchEnableAlarm](#)
- [BatchResetAlarm](#)
- [BatchSnoozeAlarm](#)

## Mengelola notifikasi alarm di AWS IoT Events

AWS IoT Events terintegrasi dengan Lambda, menawarkan kemampuan pemrosesan acara khusus. Bagian ini mengeksplorasi cara menggunakan fungsi Lambda dalam model detektor AWS IoT Events

Anda, memungkinkan Anda menjalankan logika kompleks, berinteraksi dengan layanan eksternal, dan menerapkan penanganan peristiwa yang canggih.

AWS IoT Events menggunakan fungsi Lambda untuk mengelola pemberitahuan alarm. Anda dapat menggunakan fungsi Lambda yang disediakan oleh AWS IoT Events atau membuat yang baru.

Topik

- [Membuat fungsi Lambda di AWS IoT Events](#)
- [Menggunakan fungsi Lambda yang disediakan oleh AWS IoT Events](#)
- [Mengelola akses IAM Identity Center penerima alarm di AWS IoT Events](#)

## Membuat fungsi Lambda di AWS IoT Events

AWS IoT Events menyediakan fungsi Lambda yang memungkinkan alarm untuk mengirim dan menerima pemberitahuan email dan SMS.

### Persyaratan

Persyaratan berikut berlaku saat Anda membuat fungsi Lambda untuk alarm:

- Jika alarm Anda mengirimkan notifikasi SMS, pastikan Amazon SNS dikonfigurasi untuk mengirimkan pesan SMS.
  - Untuk informasi selengkapnya, lihat dokumentasi berikut ini:
    - [Pesan teks seluler dengan Amazon SNS](#) dan [identitas Originasi untuk pesan SMS Amazon SNS di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon](#).
    - [Apa itu SMS Pesan Pengguna Akhir AWS?](#) dalam AWS SMS User Guide.
- Jika alarm Anda mengirim email atau pemberitahuan SMS, Anda harus memiliki peran IAM yang memungkinkan AWS Lambda untuk bekerja dengan Amazon SES dan Amazon SNS.

Contoh kebijakan:

JSON

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

```

    {
      "Effect": "Allow",
      "Action": [
        "ses:GetIdentityVerificationAttributes",
        "ses:SendEmail",
        "ses:VerifyEmailIdentity"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish",
        "sns:OptInPhoneNumber",
        "sns:CheckIfPhoneNumberIsOptedOut",
        "sms-voice:DescribeOptedOutNumbers"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:*:*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource" : "*"
    }
  ]
}

```

- Anda harus memilih AWS Wilayah yang sama untuk keduanya AWS IoT Events dan AWS Lambda. Untuk daftar Wilayah yang didukung, lihat [AWS IoT Events titik akhir dan kuota serta AWS Lambda titik akhir dan kuota](#) di. Referensi Umum Amazon Web

## Menerapkan fungsi Lambda untuk menggunakan AWS IoT Events CloudFormation

Tutorial ini menggunakan CloudFormation template untuk menyebarkan fungsi Lambda. Template ini secara otomatis membuat peran IAM yang memungkinkan fungsi Lambda bekerja dengan Amazon SES dan Amazon SNS.

Berikut ini menunjukkan cara menggunakan AWS Command Line Interface (AWS CLI) untuk membuat CloudFormation tumpukan.

1. Di terminal perangkat Anda, jalankan `aws --version` untuk memeriksa apakah Anda menginstal file AWS CLI. Untuk informasi selengkapnya, lihat [Menginstal atau memperbarui ke versi terbaru](#) dari Panduan AWS Command Line Interface Pengguna. AWS CLI
2. Jalankan `aws configure list` untuk memeriksa apakah Anda mengkonfigurasi AWS CLI di AWS Wilayah yang memiliki semua AWS sumber daya Anda untuk tutorial ini. Untuk informasi selengkapnya, lihat [Mengatur dan melihat pengaturan konfigurasi menggunakan perintah](#) di Panduan AWS Command Line Interface Pengguna
3. Unduh CloudFormation template, [NotificationLambda.Template.yaml.zip](#).

### Note

Jika Anda mengalami kesulitan mengunduh file, template juga tersedia di file [CloudFormation Template](#).

4. Unzip konten dan simpan secara lokal sebagai `notificationLambda.template.yaml`.
5. Buka terminal di perangkat Anda dan arahkan ke direktori tempat Anda mengunduh `notificationLambda.template.yaml` file.
6. Untuk membuat CloudFormation tumpukan, jalankan perintah berikut:

```
aws cloudformation create-stack --stack-name notificationLambda-stack --template-body file://notificationLambda.template.yaml --capabilities CAPABILITY_IAM
```

Anda dapat memodifikasi CloudFormation template ini untuk menyesuaikan fungsi Lambda dan perilakunya.

**Note**

AWS Lambda mencoba ulang kesalahan fungsi dua kali. Jika fungsi tidak memiliki kapasitas yang cukup untuk menangani semua permintaan yang masuk, peristiwa mungkin menunggu dalam antrian selama beberapa jam atau hari untuk dikirim ke fungsi tersebut. Anda dapat mengonfigurasi antrian pesan tidak terkirim (DLQ) pada fungsi untuk menangkap peristiwa yang tidak berhasil diproses. Untuk informasi selengkapnya, lihat [Invokasi asinkron](#) di Panduan Developer AWS Lambda .

Anda juga dapat membuat atau mengonfigurasi tumpukan di CloudFormation konsol. Untuk informasi selengkapnya, lihat [Bekerja dengan tumpukan](#), di Panduan AWS CloudFormation Pengguna.

## Membuat fungsi Lambda khusus untuk AWS IoT Events

Anda dapat membuat fungsi Lambda atau memodifikasi yang disediakan oleh AWS IoT Events

Persyaratan berikut berlaku saat Anda membuat fungsi Lambda kustom.

- Tambahkan izin yang memungkinkan fungsi Lambda Anda melakukan tindakan tertentu dan AWS mengakses sumber daya.
- Jika Anda menggunakan fungsi Lambda yang disediakan oleh AWS IoT Events, pastikan Anda memilih runtime Python 3.7.

Contoh fungsi Lambda:

```
import boto3
import json
import logging
import datetime
logger = logging.getLogger()
logger.setLevel(logging.INFO)
ses = boto3.client('ses')
sns = boto3.client('sns')
def check_value(target):
    if target:
        return True
    return False
```

```
# Check whether email is verified. Only verified emails are allowed to send emails to
or from.
def check_email(email):
    if not check_value(email):
        return False
    result = ses.get_identity_verification_attributes(Identities=[email])
    attr = result['VerificationAttributes']
    if (email not in attr or attr[email]['VerificationStatus'] != 'Success'):
        logging.info('Verification email for {} sent. You must have all the emails
verified before sending email.'.format(email))
        ses.verify_email_identity(EmailAddress=email)
        return False
    return True

# Check whether the phone holder has opted out of receiving SMS messages from your
account
def check_phone_number(phone_number):
    try:
        result = sns.check_if_phone_number_is_opted_out(phoneNumber=phone_number)
        if (result['isOptedOut']):
            logger.info('phoneNumber {} is not opt in of receiving SMS messages. Phone
number must be opt in first.'.format(phone_number))
            return False
        return True
    except Exception as e:
        logging.error('Your phone number {} must be in E.164 format in SS0. Exception
thrown: {}'.format(phone_number, e))
        return False

def check_emails(emails):
    result = True
    for email in emails:
        if not check_email(email):
            result = False
    return result

def lambda_handler(event, context):
    logging.info('Received event: ' + json.dumps(event))
    nep = json.loads(event.get('notificationEventPayload'))
    alarm_state = nep['alarmState']
    default_msg = 'Alarm ' + alarm_state['stateName'] + '\n'
    timestamp =
datetime.datetime.utcnow().timestamp(float(nep['stateUpdateTime'])/1000).strftime('%Y-
%m-%d %H:%M:%S')
```

```

alarm_msg = "{} {} {} at {} UTC ".format(nep['alarmModelName'], nep.get('keyValue',
'Singleton'), alarm_state['stateName'], timestamp)
default_msg += 'Sev: ' + str(nep['severity']) + '\n'
if (alarm_state['ruleEvaluation']):
    property = alarm_state['ruleEvaluation']['simpleRule']['inputProperty']
    default_msg += 'Current Value: ' + str(property) + '\n'
    operator = alarm_state['ruleEvaluation']['simpleRule']['operator']
    threshold = alarm_state['ruleEvaluation']['simpleRule']['threshold']
    alarm_msg += '({} {} {})' .format(str(property), operator, str(threshold))
default_msg += alarm_msg + '\n'

emails = event.get('emailConfigurations', [])
logger.info('Start Sending Emails')
for email in emails:
    from_adr = email.get('from')
    to_adrs = email.get('to', [])
    cc_adrs = email.get('cc', [])
    bcc_adrs = email.get('bcc', [])
    msg = default_msg + '\n' + email.get('additionalMessage', '')
    subject = email.get('subject', alarm_msg)
    fa_ver = check_email(from_adr)
    tas_ver = check_emails(to_adrs)
    ccas_ver = check_emails(cc_adrs)
    bccas_ver = check_emails(bcc_adrs)
    if (fa_ver and tas_ver and ccas_ver and bccas_ver):
        ses.send_email(Source=from_adr,
                        Destination={'ToAddresses': to_adrs, 'CcAddresses': cc_adrs,
'BccAddresses': bcc_adrs},
                        Message={'Subject': {'Data': subject}, 'Body': {'Text': {'Data':
msg}}}))
        logger.info('Emails have been sent')

logger.info('Start Sending SNS message to SMS')
sns_configs = event.get('smsConfigurations', [])
for sns_config in sns_configs:
    sns_msg = default_msg + '\n' + sns_config.get('additionalMessage', '')
    phone_numbers = sns_config.get('phoneNumbers', [])
    sender_id = sns_config.get('senderId')
    for phone_number in phone_numbers:
        if check_phone_number(phone_number):
            if check_value(sender_id):
                sns.publish(PhoneNumber=phone_number, Message=sns_msg,
MessageAttributes={'AWS.SNS.SMS.SenderID':{'DataType': 'String', 'StringValue':
sender_id}}))

```

```
else:
    sns.publish(PhoneNumber=phone_number, Message=sns_msg)
logger.info('SNS messages have been sent')
```

Untuk informasi selengkapnya, lihat [Apa itu AWS Lambda?](#) dalam Panduan Pengguna AWS Lambda

## CloudFormation Template

Gunakan CloudFormation template berikut untuk membuat fungsi Lambda Anda.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Notification Lambda for Alarm Model'
Resources:
  NotificationLambdaRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: sts:AssumeRole
      Path: "/"
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/AWSLambdaExecute'
    Policies:
      - PolicyName: "NotificationLambda"
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: "Allow"
              Action:
                - "ses:GetIdentityVerificationAttributes"
                - "ses:SendEmail"
                - "ses:VerifyEmailIdentity"
              Resource: "*"
            - Effect: "Allow"
              Action:
                - "sns:Publish"
                - "sns:OptInPhoneNumber"
                - "sns:CheckIfPhoneNumberIsOptedOut"
                - "sms-voice:DescribeOptedOutNumbers"
```

```

        Resource: "*"
    - Effect: "Deny"
      Action:
        - "sns:Publish"
      Resource: "arn:aws:sns:*:*:*"
NotificationLambdaFunction:
  Type: AWS::Lambda::Function
  Properties:
    Role: !GetAtt NotificationLambdaRole.Arn
    Runtime: python3.7
    Handler: index.lambda_handler
    Timeout: 300
    MemorySize: 3008
    Code:
      ZipFile: |
        import boto3
        import json
        import logging
        import datetime
        logger = logging.getLogger()
        logger.setLevel(logging.INFO)
        ses = boto3.client('ses')
        sns = boto3.client('sns')
        def check_value(target):
            if target:
                return True
            return False

        # Check whether email is verified. Only verified emails are allowed to send
        emails to or from.
        def check_email(email):
            if not check_value(email):
                return False
            result = ses.get_identity_verification_attributes(Identities=[email])
            attr = result['VerificationAttributes']
            if (email not in attr or attr[email]['VerificationStatus'] != 'Success'):
                logging.info('Verification email for {} sent. You must have all the
                emails verified before sending email.'.format(email))
                ses.verify_email_identity(EmailAddress=email)
                return False
            return True

        # Check whether the phone holder has opted out of receiving SMS messages from
        your account

```

```

def check_phone_number(phone_number):
    try:
        result = sns.check_if_phone_number_is_opted_out(phoneNumber=phone_number)
        if (result['isOptedOut']):
            logger.info('phoneNumber {} is not opt in of receiving SMS messages.
Phone number must be opt in first.'.format(phone_number))
            return False
        return True
    except Exception as e:
        logging.error('Your phone number {} must be in E.164 format in SS0.
Exception thrown: {}'.format(phone_number, e))
        return False

def check_emails(emails):
    result = True
    for email in emails:
        if not check_email(email):
            result = False
    return result

def lambda_handler(event, context):
    logging.info('Received event: ' + json.dumps(event))
    nep = json.loads(event.get('notificationEventPayload'))
    alarm_state = nep['alarmState']
    default_msg = 'Alarm ' + alarm_state['stateName'] + '\n'
    timestamp =
datetime.datetime.utcnow().timestamp(float(nep['stateUpdateTime']/1000)).strftime('%Y-
%m-%d %H:%M:%S')
    alarm_msg = "{} {} {} at {} UTC ".format(nep['alarmModelName'],
nep.get('keyValue', 'Singleton'), alarm_state['stateName'], timestamp)
    default_msg += 'Sev: ' + str(nep['severity']) + '\n'
    if (alarm_state['ruleEvaluation']):
        property = alarm_state['ruleEvaluation']['simpleRule']['inputProperty']
        default_msg += 'Current Value: ' + str(property) + '\n'
        operator = alarm_state['ruleEvaluation']['simpleRule']['operator']
        threshold = alarm_state['ruleEvaluation']['simpleRule']['threshold']
        alarm_msg += '({} {} {})'.format(str(property), operator, str(threshold))
    default_msg += alarm_msg + '\n'

    emails = event.get('emailConfigurations', [])
    logger.info('Start Sending Emails')
    for email in emails:
        from_addr = email.get('from')
        to_addrs = email.get('to', [])

```

```

cc_adrs = email.get('cc', [])
bcc_adrs = email.get('bcc', [])
msg = default_msg + '\n' + email.get('additionalMessage', '')
subject = email.get('subject', alarm_msg)
fa_ver = check_email(from_adr)
tas_ver = check_emails(to_adrs)
ccas_ver = check_emails(cc_adrs)
bccas_ver = check_emails(bcc_adrs)
if (fa_ver and tas_ver and ccas_ver and bccas_ver):
    ses.send_email(Source=from_adr,
                   Destination={'ToAddresses': to_adrs, 'CcAddresses':
cc_adrs, 'BccAddresses': bcc_adrs},
                   Message={'Subject': {'Data': subject}, 'Body': {'Text':
{'Data': msg}}})
    logger.info('Emails have been sent')

logger.info('Start Sending SNS message to SMS')
sns_configs = event.get('smsConfigurations', [])
for sns_config in sns_configs:
    sns_msg = default_msg + '\n' + sns_config.get('additionalMessage', '')
    phone_numbers = sns_config.get('phoneNumbers', [])
    sender_id = sns_config.get('senderId')
    for phone_number in phone_numbers:
        if check_phone_number(phone_number):
            if check_value(sender_id):
                sns.publish(PhoneNumber=phone_number, Message=sns_msg,
MessageAttributes={'AWS.SNS.SMS.SenderID':{'DataType': 'String', 'StringValue':
sender_id}})
            else:
                sns.publish(PhoneNumber=phone_number, Message=sns_msg)
    logger.info('SNS messages have been sent')

```

## Menggunakan fungsi Lambda yang disediakan oleh AWS IoT Events

Dengan pemberitahuan alarm, Anda dapat menggunakan fungsi Lambda yang disediakan oleh AWS IoT Events untuk mengelola pemberitahuan alarm.

Persyaratan berikut berlaku saat Anda menggunakan fungsi Lambda yang disediakan oleh AWS IoT Events untuk mengelola notifikasi alarm Anda:

- Anda harus memverifikasi alamat email yang mengirimkan pemberitahuan email di Amazon Simple Email Service (Amazon SES). Untuk informasi selengkapnya, lihat [Memverifikasi identitas alamat email](#), di Panduan Pengembang Layanan Email Sederhana Amazon.

Jika Anda menerima tautan verifikasi, klik tautan untuk memverifikasi alamat email Anda. Anda juga dapat memeriksa folder spam Anda untuk email verifikasi.

- Jika alarm Anda mengirimkan notifikasi SMS, Anda harus menggunakan format nomor telepon internasional E.164 untuk nomor telepon. Format ini berisi `+<country-calling-code><area-code><phone-number>`.

Contoh nomor telepon:

Negara	Nomor telepon lokal	Nomor yang diformat E.164
Amerika Serikat	206-555-0100	+12065550100
Britania Raya	020-1234-1234	+442012341234
Lithuania	8+601+12345	+37060112345

Untuk menemukan kode panggilan negara, buka [countrycode.org](http://countrycode.org).

Fungsi Lambda disediakan dengan AWS IoT Events memeriksa apakah Anda menggunakan nomor telepon berformat E.164. Namun, itu tidak memverifikasi nomor telepon. Jika Anda memastikan bahwa Anda memasukkan nomor telepon yang akurat tetapi tidak menerima pemberitahuan SMS, Anda dapat menghubungi operator telepon. Operator dapat memblokir pesan.

## Mengelola akses IAM Identity Center penerima alarm di AWS IoT Events

AWS IoT Events digunakan AWS IAM Identity Center untuk mengelola akses SSO penerima alarm. Menerapkan IAM Identity Center untuk penerima AWS IoT Events notifikasi dapat meningkatkan keamanan dan pengalaman pengguna. Untuk mengaktifkan alarm mengirim notifikasi ke penerima, Anda harus mengaktifkan Pusat Identitas IAM dan menambahkan penerima ke toko Pusat Identitas IAM Anda. Untuk informasi selengkapnya, lihat [Menambahkan AWS IAM Identity Center Pengguna](#) di Panduan Pengguna.

**⚠ Important**

- Anda harus memilih AWS Region yang sama untuk AWS IoT Events, AWS Lambda, dan IAM Identity Center.
- AWS Organizations hanya mendukung satu Wilayah Pusat Identitas IAM pada satu waktu. Jika Anda ingin membuat Pusat Identitas IAM tersedia di Wilayah yang berbeda, Anda harus terlebih dahulu menghapus konfigurasi Pusat Identitas IAM Anda saat ini. Untuk informasi selengkapnya, lihat [Data Wilayah Pusat Identitas IAM](#) di Panduan AWS IAM Identity Center Pengguna.

# Keamanan di AWS IoT Events

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](#) menggambarkan hal ini sebagai keamanan dari cloud dan keamanan di 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. Efektivitas keamanan kami diuji dan diverifikasi secara rutin oleh auditor pihak ketiga sebagai bagian dari [program kepatuhan AWS](#). Untuk mempelajari tentang program kepatuhan yang berlaku AWS IoT Events, lihat [AWS layanan dalam cakupan 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, persyaratan perusahaan, serta hukum dan peraturan yang berlaku.

Dokumentasi ini akan membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan AWS IoT Events. Topik berikut menunjukkan cara mengonfigurasi AWS IoT Events untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga akan belajar cara menggunakan AWS layanan lain yang dapat membantu Anda memantau dan mengamankan AWS IoT Events sumber daya Anda.

## Topik

- [Identitas dan manajemen akses untuk AWS IoT Events](#)
- [Pemantauan AWS IoT Events untuk menjaga keandalan, ketersediaan, dan kinerja](#)
- [Validasi kepatuhan untuk AWS IoT Events](#)
- [Ketahanan di AWS IoT Events](#)
- [Keamanan infrastruktur di AWS IoT Events](#)

# Identitas dan manajemen akses untuk AWS IoT Events

AWS Identity and Access Management (IAM) adalah AWS layanan 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 Events IAM adalah AWS layanan yang dapat Anda gunakan tanpa biaya tambahan.

## Topik

- [Audiens](#)
- [Mengautentikasi dengan identitas](#)
- [Mengelola akses menggunakan kebijakan](#)
- [Lebih lanjut tentang identitas dan manajemen akses](#)
- [Bagaimana AWS IoT Events bekerja dengan IAM](#)
- [AWS IoT Events contoh kebijakan berbasis identitas](#)
- [Pencegahan deperiti kebingungan lintas layanan untuk AWS IoT Events](#)
- [Memecahkan masalah AWS IoT Events identitas dan akses](#)

## Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda berdasarkan peran Anda:

- Pengguna layanan - minta izin dari administrator Anda jika Anda tidak dapat mengakses fitur (lihat [Memecahkan masalah AWS IoT Events identitas dan akses](#))
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat [Bagaimana AWS IoT Events bekerja dengan IAM](#))
- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat [AWS IoT Events contoh kebijakan berbasis identitas](#))

## Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensial dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), otentikasi masuk tunggal, atau kredensial. Google/Facebook Untuk informasi selengkapnya tentang cara masuk, lihat [Cara masuk ke Akun AWS Anda](#) dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat [AWS Signature Version 4 untuk permintaan API](#) dalam Panduan Pengguna IAM.

## Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat [Tugas yang memerlukan kredensial pengguna root](#) dalam Panduan Pengguna IAM.

## Pengguna dan grup IAM

[Pengguna IAM](#) adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk informasi selengkapnya, lihat [Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensi sementara](#) di Panduan Pengguna IAM.

[Grup IAM](#) menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat [Kasus penggunaan untuk pengguna IAM](#) dalam Panduan Pengguna IAM.

## Peran IAM

[Peran IAM](#) adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan [beralih dari pengguna ke peran IAM \(konsol\)](#) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat [Metode untuk mengambil peran](#) dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat [Akses sumber daya lintas akun di IAM](#) dalam Panduan Pengguna IAM.

## Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat [Gambaran umum kebijakan JSON](#) dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

### Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat [Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada banyak identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat [Pilih antara kebijakan terkelola dan kebijakan inline](#) dalam Panduan Pengguna IAM.

### Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat [Batasan izin untuk entitas IAM](#) dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCPs) — Tentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat [Kebijakan kontrol layanan](#) dalam Panduan Pengguna AWS Organizations .

- Kebijakan kontrol sumber daya (RCPs) — Tetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat [Kebijakan kontrol sumber daya \(RCPs\)](#) di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. 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.

## Lebih lanjut tentang identitas dan manajemen akses

Untuk informasi lebih lanjut tentang identitas dan manajemen akses AWS IoT Events, lanjutkan ke halaman berikut:

- [Bagaimana AWS IoT Events bekerja dengan IAM](#)
- [Memecahkan masalah AWS IoT Events identitas dan akses](#)

## Bagaimana AWS IoT Events bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses AWS IoT Events, Anda harus memahami fitur IAM apa yang tersedia untuk digunakan. AWS IoT Events Untuk mendapatkan pandangan tingkat tinggi tentang bagaimana AWS IoT Events dan AWS layanan lain bekerja dengan IAM, lihat [AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM](#).

### Topik

- [AWS IoT Events kebijakan berbasis identitas](#)
- [AWS IoT Events kebijakan berbasis sumber daya](#)
- [Otorisasi berdasarkan tag AWS IoT Events](#)
- [AWS IoT Events Peran IAM](#)

## AWS IoT Events 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 Events 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

Elemen `Action` kebijakan berbasis identitas IAM menjelaskan tindakan atau tindakan tertentu yang akan diizinkan atau ditolak oleh kebijakan tersebut. Tindakan kebijakan biasanya memiliki nama yang sama dengan operasi AWS API terkait. Tindakan ini digunakan dalam kebijakan untuk memberikan izin guna melakukan operasi terkait.

Tindakan kebijakan AWS IoT Events menggunakan awalan berikut sebelum tindakan: `iotevents:`. Misalnya, untuk memberikan izin kepada seseorang untuk membuat AWS IoT Events input dengan operasi AWS IoT Events `CreateInput` API, Anda menyertakan `iotevents:CreateInput` tindakan tersebut dalam kebijakan mereka. Untuk memberikan izin kepada seseorang untuk mengirim input dengan operasi AWS IoT Events `BatchPutMessage` API, Anda menyertakan `iotevents-data:BatchPutMessage` tindakan tersebut dalam kebijakan mereka. Pernyataan kebijakan harus mencakup salah satu `Action` atau `NotAction` elemen. AWS IoT Events 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": [
  "iotevents:action1",
  "iotevents:action2"
```

Anda dapat menentukan beberapa tindakan menggunakan wildcard (\*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "iotevents:Describe*"
```

Untuk melihat daftar tindakan, lihat AWS IoT Events [Tindakan yang Ditentukan oleh AWS IoT Events](#) dalam Panduan Pengguna IAM.

## Sumber daya

Elemen `Resource` menentukan objek di mana tindakan berlaku. Pernyataan harus mencakup elemen `Resource` atau `NotResource`. Anda menentukan sumber daya menggunakan ARN atau menggunakan wildcard (\*) untuk menunjukkan bahwa pernyataan berlaku untuk semua sumber daya.

Sumber daya model AWS IoT Events detektor memiliki ARN berikut:

```
arn:${Partition}:iotevents:${Region}:${Account}:detectorModel/${detectorModelName}
```

Untuk informasi selengkapnya tentang format ARNs, lihat [Mengidentifikasi AWS sumber daya dengan Nama Sumber Daya Amazon \(ARNs\)](#).

Misalnya, untuk menentukan model Foobar detektor dalam pernyataan Anda, gunakan ARN berikut:

```
"Resource": "arn:aws:iotevents:us-east-1:123456789012:detectorModel/Foobar"
```

Untuk menentukan semua instans milik akun tertentu, gunakan wildcard (\*):

```
"Resource": "arn:aws:iotevents:us-east-1:123456789012:detectorModel/*"
```

Beberapa AWS IoT Events tindakan, seperti untuk membuat sumber daya, tidak dapat dilakukan pada sumber daya tertentu. Dalam kasus tersebut, Anda harus menggunakan wildcard (\*).

```
"Resource": "*" 
```

Beberapa tindakan AWS IoT Events API melibatkan banyak sumber daya. Misalnya, `CreateDetectorModel` referensi input dalam pernyataan kondisinya, sehingga pengguna harus memiliki izin untuk menggunakan input dan model detektor. Untuk menentukan beberapa sumber daya dalam satu pernyataan, pisahkan ARNs dengan koma.

```
"Resource": [  
  "resource1",  
  "resource2"
```

Untuk melihat daftar jenis sumber daya dan jenis AWS IoT Events sumber daya ARNs, lihat [Sumber Daya yang Ditentukan oleh AWS IoT Events](#) dalam Panduan Pengguna IAM. Untuk mempelajari

tindakan mana yang dapat menentukan ARN setiap sumber daya, lihat [Tindakan yang Ditentukan oleh AWS IoT Events](#).

## Kunci syarat

Elemen Condition (atau blok Condition) memungkinkan Anda menentukan ketentuan yang mengizinkan Anda untuk menerapkan pernyataan. Elemen Condition bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan [operator kondisi](#), seperti sama dengan atau kurang dari, untuk mencocokkan ketentuan dalam kebijakan dengan nilai dalam permintaan.

Jika Anda menentukan beberapa elemen Condition dalam pernyataan, atau beberapa kunci dalam satu elemen Condition, AWS mengevaluasinya menggunakan operasi AND. 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. Misalnya, Anda dapat memberikan izin pengguna untuk mengakses sumber daya hanya jika ditandai dengan nama pengguna mereka. Untuk informasi lebih lanjut, lihat [Elemen kebijakan IAM: Variabel dan tanda](#) dalam Panduan Pengguna IAM.

AWS IoT Events tidak menyediakan kunci kondisi khusus layanan apa pun, tetapi mendukung penggunaan beberapa kunci kondisi global. Untuk melihat semua kunci kondisi AWS global, lihat [kunci konteks kondisi AWS global](#) di Panduan Pengguna IAM.

## Contoh

Untuk melihat contoh kebijakan AWS IoT Events berbasis identitas, lihat. [AWS IoT Events contoh kebijakan berbasis identitas](#)

## AWS IoT Events kebijakan berbasis sumber daya

AWS IoT Events tidak mendukung kebijakan berbasis sumber daya.” Untuk melihat contoh halaman detail kebijakan berbasis sumber daya, lihat <https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html>.

## Otorisasi berdasarkan tag AWS IoT Events

Anda dapat melampirkan tag ke AWS IoT Events sumber daya atau meneruskan tag dalam permintaan AWS IoT Events. Untuk mengendalikan akses berdasarkan tanda, berikan informasi tentang tanda di [elemen kondisi](#) dari kebijakan menggunakan kunci kondisi `iotevents:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`. Untuk

informasi selengkapnya tentang penandaan sumber daya AWS IoT Events , lihat [Menandai sumber daya Anda AWS IoT Events](#).

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat [Lihat AWS IoT Events input berdasarkan tag](#).

## AWS IoT Events Peran IAM

[Peran IAM](#) adalah entitas di dalam Anda Akun AWS yang memiliki izin khusus.

### Menggunakan kredensi sementara dengan AWS IoT Events

Anda dapat menggunakan kredensial sementara untuk masuk dengan gabungan, menjalankan IAM role, atau menjalankan peran lintas akun. Anda memperoleh kredensi keamanan sementara dengan memanggil AWS Security Token Service (AWS STS) operasi API seperti [AssumeRole](#) atau [GetFederationToken](#).

AWS IoT Events tidak mendukung penggunaan 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 Events 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 Events mendukung peran layanan.

## AWS IoT Events contoh kebijakan berbasis identitas

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi AWS IoT Events sumber daya. Mereka juga tidak dapat melakukan tugas menggunakan Konsol Manajemen

AWS, 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 AWS IoT Events konsol](#)
- [Izinkan pengguna untuk melihat izin mereka sendiri di AWS IoT Events](#)
- [Akses satu AWS IoT Events masukan](#)
- [Lihat AWS IoT Events input berdasarkan tag](#)

## Praktik terbaik kebijakan

Kebijakan berbasis identitas adalah pilihan yang sangat tepat. Mereka menentukan apakah seseorang dapat membuat, mengakses, atau menghapus AWS IoT Events 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:

- **Memulai Menggunakan Kebijakan AWS Terkelola** — Untuk mulai menggunakan AWS IoT Events dengan cepat, gunakan kebijakan AWS terkelola untuk memberi karyawan Anda izin yang mereka butuhkan. Kebijakan ini sudah tersedia di akun Anda dan dikelola, serta diperbarui oleh AWS. Untuk informasi selengkapnya, lihat [Memulai menggunakan izin dengan kebijakan AWS terkelola](#) di Panduan Pengguna IAM.
- **Berikan hak akses terkecil** – Saat Anda membuat kebijakan khusus, berikan izin yang diperlukan untuk melaksanakan tugas saja. Mulai dengan satu set izin minimum dan berikan izin tambahan sesuai kebutuhan. Melakukan hal tersebut lebih aman daripada memulai dengan izin yang terlalu fleksibel, lalu mencoba memperketatnya nanti. Untuk informasi selengkapnya, lihat [Pemberian hak istimewa terendah](#) dalam Panduan Pengguna IAM.
- **Aktifkan MFA untuk Operasi Sensitif** — Untuk keamanan ekstra, pengguna harus menggunakan otentikasi multi-faktor (MFA) untuk mengakses sumber daya sensitif atau operasi API. Untuk informasi selengkapnya, lihat [Menggunakan autentikasi multifaktor \(MFA\) dalam AWS](#) dalam Panduan Pengguna IAM.

- Gunakan Kondisi Kebijakan untuk Keamanan Tambahan – Selama praktis, tentukan ketentuan di mana kebijakan berbasis identitas Anda memungkinkan akses ke sumber daya. Misalnya, Anda dapat menulis persyaratan untuk menentukan jangkauan alamat IP yang diizinkan untuk mengajukan permintaan. Anda juga dapat menulis persyaratan untuk mengizinkan permintaan hanya dalam rentang tanggal atau waktu tertentu, atau untuk mewajibkan penggunaan SSL atau autentikasi multifaktor (MFA). Untuk informasi selengkapnya, lihat [Elemen kebijakan JSON IAM: Syarat](#) dalam Panduan Pengguna IAM.

## Menggunakan AWS IoT Events konsol

Untuk mengakses AWS IoT Events konsol, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang AWS IoT Events 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 Events konsol, lampirkan juga kebijakan AWS terkelola berikut ke entitas. Untuk informasi selengkapnya, lihat [Menambahkan izin ke pengguna di Panduan Pengguna](#) IAM:

### JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotevents:BatchPutMessage",
        "iotevents:BatchUpdateDetector",
        "iotevents:CreateDetectorModel",
        "iotevents:CreateInput",
        "iotevents>DeleteDetectorModel",
        "iotevents>DeleteInput",
        "iotevents:DescribeDetector",
        "iotevents:DescribeDetectorModel",
        "iotevents:DescribeInput",
        "iotevents:DescribeLoggingOptions",
        "iotevents:ListDetectorModelVersions",
        "iotevents:ListDetectorModels",

```

```

        "iotevents:ListDetectors",
        "iotevents:ListInputs",
        "iotevents:ListTagsForResource",
        "iotevents:PutLoggingOptions",
        "iotevents:TagResource",
        "iotevents:UntagResource",
        "iotevents:UpdateDetectorModel",
        "iotevents:UpdateInput",
        "iotevents:UpdateInputRouting"
    ],
    "Resource": "arn:aws:iotevents:us-
east-1:123456789012:detectorModel/your-detector-model-name",
    "Resource": "arn:aws:iotevents:us-east-1:123456789012:input/your-input-name"
  }
]
}

```

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 pengguna untuk melihat izin mereka sendiri di AWS IoT Events

Contoh ini menunjukkan cara Anda membuat kebijakan yang memungkinkan pengguna melihat kebijakan sebaris dan terkelola yang dilampirkan pada identitas pengguna mereka. Memungkinkan pengguna untuk melihat izin IAM mereka sendiri berguna untuk kesadaran keamanan dan kemampuan layanan mandiri. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI AWS

### JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",

```

```

        "iam:ListUserPolicies",
        "iam:GetUser"
    ],
    "Resource": [
        "arn:aws:iam::*:user/${aws:username}"
    ]
},
{
    "Sid": "NavigateInConsole",
    "Effect": "Allow",
    "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
    ],
    "Resource": "*"
}
]
}

```

## Akses satu AWS IoT Events masukan

Kontrol akses granular ke AWS IoT Events input penting untuk menjaga keamanan di lingkungan multi-pengguna atau multi-tim. Bagian ini menunjukkan cara membuat kebijakan IAM yang memberikan akses ke AWS IoT Events input tertentu sambil membatasi akses ke orang lain.

Dalam contoh ini, Anda dapat memberikan pengguna Akun AWS akses ke salah satu AWS IoT Events input Anda, `exampleInput`. Anda juga dapat mengizinkan pengguna untuk menambah, memperbarui, dan menghapus input.

Kebijakan memberikan `iotevents:ListInputs`, `iotevents:DescribeInput`, `iotevents>CreateInput`, `iotevents>DeleteInput`, dan `iotevents:UpdateInput` izin kepada pengguna. Untuk contoh panduan untuk Amazon Simple Storage Service (Amazon S3) yang memberikan izin kepada pengguna dan mengujinya menggunakan konsol, [lihat Mengontrol](#) akses ke bucket dengan kebijakan pengguna.

## JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListInputsInConsole",
      "Effect": "Allow",
      "Action": [
        "iotevents:ListInputs"
      ],
      "Resource": "arn:aws:iotevents:us-east-2:123456789012:input/*"
    },
    {
      "Sid": "ViewSpecificInputInfo",
      "Effect": "Allow",
      "Action": [
        "iotevents:DescribeInput"
      ],
      "Resource": "arn:aws:iotevents:us-east-1:123456789012:input/inputName"
    },
    {
      "Sid": "ManageInputs",
      "Effect": "Allow",
      "Action": [
        "iotevents:CreateInput",
        "iotevents>DeleteInput",
        "iotevents:DescribeInput",
        "iotevents:ListInputs",
        "iotevents:UpdateInput"
      ],
      "Resource": "arn:aws:iotevents:us-east-1:123456789012:input/*"
    }
  ]
}
```

## Lihat AWS IoT Events input berdasarkan tag

Tag membantu Anda mengatur AWS IoT Events sumber daya. Anda dapat menggunakan kondisi dalam kebijakan berbasis identitas untuk mengontrol akses ke AWS IoT Events sumber daya berdasarkan tag. Contoh ini menunjukkan bagaimana Anda dapat membuat kebijakan yang

memungkinkan melihat *input*. Namun, izin diberikan hanya jika *input* tag Owner memiliki nilai nama pengguna pengguna tersebut. Kebijakan ini juga memberi izin yang diperlukan untuk menyelesaikan tindakan ini pada konsol tersebut.

## JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListInputsInConsole",
      "Effect": "Allow",
      "Action": "iotevents:ListInputs",
      "Resource": "*"
    },
    {
      "Sid": "ViewInputsIfOwner",
      "Effect": "Allow",
      "Action": "iotevents:ListInputs",
      "Resource": "arn:aws:iotevents:*:*:input/*",
      "Condition": {
        "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
      }
    }
  ]
}
```

Anda dapat melampirkan kebijakan ini ke pengguna di akun Anda. Jika pengguna bernama *richard-roe* mencoba untuk melihat AWS IoT Events *input*, *input* harus ditandai `Owner=richard-roe` atau `owner=richard-roe`. Jika tidak, aksesnya akan ditolak. Kunci tanda syarat Owner cocok dengan Owner dan owner karena nama kunci syarat tidak terpengaruh huruf besar/kecil. Untuk informasi selengkapnya, lihat [Elemen kebijakan IAM JSON: Syarat](#) dalam Panduan Pengguna IAM.

## Pencegahan deputi kebingungan lintas layanan untuk AWS IoT Events

### Note

- AWS IoT Events Layanan ini hanya memungkinkan Anda untuk menggunakan peran untuk memulai tindakan di akun yang sama di mana sumber daya dibuat. Ini membantu mencegah serangan wakil yang membingungkan AWS IoT Events.
- Halaman ini berfungsi sebagai referensi bagi Anda untuk melihat bagaimana masalah wakil yang membingungkan bekerja dan dapat dicegah jika sumber daya lintas akun diizinkan dalam AWS IoT Events layanan.

Masalah "confused deputy" adalah masalah keamanan di mana entitas yang tidak memiliki izin untuk melakukan tindakan dapat memengaruhi entitas yang memiliki hak akses lebih tinggi 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 dalam kebijakan sumber daya untuk membatasi izin yang AWS IoT Events memberikan layanan lain ke sumber daya. Jika nilai `aws:SourceArn` tidak berisi ID akun, seperti ARN bucket Amazon S3, Anda harus menggunakan kedua kunci konteks kondisi global tersebut untuk membatasi izin. Jika Anda menggunakan kunci konteks kondisi global dan nilai `aws:SourceArn` berisi ID akun, nilai `aws:SourceAccount` dan akun dalam nilai `aws:SourceArn` harus menggunakan ID akun yang sama saat digunakan dalam pernyataan kebijakan yang sama.

Gunakan `aws:SourceArn` jika Anda ingin hanya satu sumber daya yang akan dikaitkan dengan akses lintas layanan. Gunakan `aws:SourceAccount` jika Anda ingin mengizinkan sumber daya apa pun di akun tersebut dikaitkan dengan penggunaan lintas layanan. Nilai `aws:SourceArn` harus Model Detektor atau model Alarm yang terkait dengan `sts:AssumeRole` permintaan.

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:iotevents*:123456789012:*`.

Contoh berikut menunjukkan bagaimana Anda dapat menggunakan kunci konteks kondisi `aws:SourceAccount` global `aws:SourceArn` dan AWS IoT Events untuk mencegah masalah wakil yang membingungkan.

## Topik

- [Contoh: Akses aman ke model AWS IoT Events detektor](#)
- [Contoh: Akses aman ke model AWS IoT Events alarm](#)
- [Contoh: Mengakses AWS IoT Events sumber daya di wilayah tertentu](#)
- [Contoh: Konfigurasi opsi logging untuk AWS IoT Events](#)

## Contoh: Akses aman ke model AWS IoT Events detektor

Contoh ini menunjukkan cara membuat kebijakan IAM yang secara aman memberikan akses ke model detektor tertentu di AWS IoT Events. Kebijakan menggunakan kondisi untuk memastikan bahwa hanya AWS akun dan AWS IoT Events layanan tertentu yang dapat mengambil peran tersebut, menambahkan lapisan keamanan tambahan. Dalam contoh ini, peran hanya dapat mengakses model detektor bernama `WindTurbine01`.

## JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
```

```

        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:us-
east-1:123456789012:detectorModel/WindTurbine01"
        }
      }
    ]
  }
}

```

## Contoh: Akses aman ke model AWS IoT Events alarm

Contoh ini menunjukkan cara membuat kebijakan IAM yang memungkinkan AWS IoT Events untuk mengakses model alarm dengan aman. Kebijakan menggunakan ketentuan untuk memastikan bahwa hanya AWS akun dan AWS IoT Events layanan yang ditentukan yang dapat mengambil peran tersebut.

Dalam contoh ini, peran dapat mengakses model alarm apa pun dalam AWS akun yang ditentukan, seperti yang ditunjukkan oleh \* wildcard dalam model alarm ARN. Kondisi `aws:SourceAccount` dan `aws:SourceArn` kondisi bekerja sama untuk mencegah masalah wakil yang membingungkan.

## JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {

```

```

        "aws:SourceArn": "arn:aws:iotevents:us-
east-1:123456789012:alarmModel/*"
    }
}
]
}

```

## Contoh: Mengakses AWS IoT Events sumber daya di wilayah tertentu

Contoh ini menunjukkan cara mengkonfigurasi peran IAM untuk mengakses AWS IoT Events sumber daya di wilayah tertentu AWS . Dengan menggunakan wilayah khusus ARNs dalam kebijakan IAM Anda, Anda dapat membatasi akses ke AWS IoT Events sumber daya di berbagai wilayah geografis. Pendekatan ini dapat membantu menjaga keamanan dan kepatuhan dalam penerapan multi-wilayah. Wilayah dalam contoh ini adalah *us-east-1*.

### JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:us-east-1:123456789012:*"
        }
      }
    }
  ]
}

```

## Contoh: Konfigurasi opsi logging untuk AWS IoT Events

Pencatatan yang tepat penting untuk memantau, men-debug, dan mengaudit aplikasi Anda AWS IoT Events . Bagian ini memberikan ikhtisar opsi pencatatan yang tersedia di AWS IoT Events.

Contoh ini menunjukkan cara mengkonfigurasi peran IAM yang memungkinkan AWS IoT Events untuk mencatat data ke CloudWatch Log. Penggunaan wildcard (\*) di ARN sumber daya memungkinkan pencatatan komprehensif di seluruh AWS IoT Events infrastruktur Anda.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:us-east-1:123456789012:*"
        }
      }
    }
  ]
}
```

## Memecahkan masalah AWS IoT Events identitas dan akses

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan AWS IoT Events dan IAM.

Topik

- [Saya tidak berwenang untuk melakukan tindakan di AWS IoT Events](#)
- [Saya tidak berwenang untuk melakukan iam:PassRole](#)
- [Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses AWS IoT Events sumber daya saya](#)

## Saya tidak berwenang untuk melakukan tindakan di AWS IoT Events

Jika Konsol Manajemen AWS memberitahu Anda bahwa Anda tidak berwenang untuk melakukan tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberikan nama pengguna dan kata sandi Anda.

Contoh kesalahan berikut terjadi ketika pengguna mateojackson IAM mencoba menggunakan konsol untuk melihat detail tentang *input* tetapi tidak memiliki `iotevents:ListInputs` izin.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
iotevents:ListInputs on resource: my-example-input
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya untuk mengizinkan dia mengakses sumber daya *my-example-input* menggunakan tindakan `iotevents:ListInput`.

## 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 Events.

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 Events. 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 Events 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 (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Konsultasikan topik berikut untuk menentukan opsi terbaik Anda:

- Untuk mempelajari apakah AWS IoT Events mendukung fitur-fitur ini, lihat [Bagaimana AWS IoT Events 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](#).

## Pemantauan AWS IoT Events untuk menjaga keandalan, ketersediaan, dan kinerja

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja AWS IoT Events 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. Sebelum

Anda mulai memantau AWS IoT Events, Anda harus membuat rencana pemantauan yang mencakup jawaban atas pertanyaan-pertanyaan berikut:

- Apa saja sasaran pemantauan Anda?
- Sumber daya manakah yang akan Anda pantau?
- Seberapa seringkah Anda akan memantau sumber daya ini?
- Apa sajakah alat pemantauan yang akan Anda gunakan?
- Siapakah yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Langkah selanjutnya adalah menetapkan dasar untuk AWS IoT Events kinerja normal di lingkungan Anda, dengan mengukur kinerja pada berbagai waktu dan dalam kondisi beban yang berbeda. Saat Anda memantau AWS IoT Events, simpan data pemantauan historis agar Anda dapat membandingkannya dengan data performa baru, mengidentifikasi pola performa normal dan anomali performa, dan merancang metode untuk mengatasi masalah.

Misalnya, jika Anda menggunakan Amazon EC2, Anda dapat memantau pemanfaatan CPU, disk I/O, and network utilization for your instances. When performance falls outside your established baseline, you might need to reconfigure or optimize the instance to reduce CPU utilization, improve disk I/O, atau mengurangi lalu lintas jaringan.

Topik

- [Alat yang tersedia untuk memantau AWS IoT Events](#)
- [Pemantauan AWS IoT Events dengan Amazon CloudWatch](#)
- [Pencatatan panggilan AWS IoT Events API dengan AWS CloudTrail](#)

## Alat yang tersedia untuk memantau AWS IoT Events

AWS menyediakan berbagai alat yang dapat Anda gunakan untuk memantau AWS IoT Events. Anda dapat mengonfigurasi beberapa alat ini untuk melakukan pemantauan untuk Anda, sementara 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 Events dan melaporkan ketika ada sesuatu yang salah:

- Amazon CloudWatch Logs — Pantau, simpan, dan akses file log Anda dari AWS CloudTrail atau sumber lain. Untuk informasi selengkapnya, lihat [Menggunakan CloudWatch dasbor 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 [Bekerja dengan file CloudTrail log](#) di Panduan AWS CloudTrail Pengguna.

## Alat pemantauan manual

Bagian penting lainnya dari pemantauan AWS IoT Events melibatkan pemantauan secara manual item-item yang tidak tercakup oleh CloudWatch alarm. Dasbor AWS konsol AWS IoT Events CloudWatch, dan lainnya memberikan at-a-glance tampilan status AWS lingkungan Anda. Kami menyarankan Anda juga memeriksa file log AWS IoT Events.

- AWS IoT Events Konsol menunjukkan:
  - Model detektor
  - Detektor
  - Masukan
  - Pengaturan
- CloudWatch Halaman beranda menunjukkan:
  - Alarm dan status saat ini
  - Grafik alarm dan sumber daya
  - Status kesehatan layanan

Selain itu, Anda dapat menggunakan CloudWatch untuk melakukan hal berikut:

- Buat [Membuat CloudWatch dasbor](#) untuk memantau layanan yang Anda pedulikan
- Data metrik grafik untuk memecahkan masalah dan mengungkap tren
- Cari dan telusuri semua metrik AWS sumber daya Anda
- Membuat dan mengedit alarm untuk menerima notifikasi terkait masalah

## Pemantauan AWS IoT Events dengan Amazon CloudWatch

Saat Anda mengembangkan atau men-debug model AWS IoT Events detektor, Anda perlu tahu apa yang AWS IoT Events sedang dilakukan, dan kesalahan apa pun yang ditemuinya. Amazon CloudWatch memantau AWS sumber daya Anda dan aplikasi yang Anda jalankan AWS secara real time. Dengan CloudWatch, Anda mendapatkan visibilitas sistem ke dalam penggunaan sumber daya, kinerja aplikasi, dan kesehatan operasional. [Aktifkan CloudWatch pencatatan Amazon saat mengembangkan model AWS IoT Events detektor](#) memiliki informasi tentang cara mengaktifkan CloudWatch logging untuk AWS IoT Events. Untuk menghasilkan log seperti yang ditunjukkan di bawah ini, Anda harus mengatur Level verbositas ke 'Debug' dan menyediakan satu atau beberapa Target Debug yang merupakan Nama Model Detektor dan opsional. KeyValue

Contoh berikut menunjukkan entri log tingkat CloudWatch DEBUG yang dihasilkan oleh AWS IoT Events.

```
{
  "timestamp": "2019-03-15T15:56:29.412Z",
  "level": "DEBUG",
  "logMessage": "Summary of message evaluation",
  "context": "MessageEvaluation",
  "status": "Success",
  "messageId": "SensorAggregate_2th846h",
  "keyValue": "boiler_1",
  "detectorModelName": "BoilerAlarmDetector",
  "initialState": "high_temp_alarm",
  "initialVariables": {
    "high_temp_count": 1,
    "high_pressure_count": 1
  },
  "finalState": "no_alarm",
  "finalVariables": {
    "high_temp_count": 0,
    "high_pressure_count": 0
  },
  "message": "{\"temp\": 34.9, \"pressure\": 84.5}",
  "messageType": "CUSTOMER_MESSAGE",
  "conditionEvaluationResults": [
    {
      "result": "True",
      "eventName": "alarm_cleared",
      "state": "high_temp_alarm",
      "lifeCycle": "OnInput",
    }
  ]
}
```

```
    "hasTransition": true
  },
  {
    "result": "Skipped",
    "eventName": "alarm_escalated",
    "state": "high_temp_alarm",
    "lifeCycle": "OnInput",
    "hasTransition": true,
    "resultDetails": "Skipped due to transition from alarm_cleared event"
  },
  {
    "result": "True",
    "eventName": "should_recall_technician",
    "state": "no_alarm",
    "lifeCycle": "OnEnter",
    "hasTransition": true
  }
]
}
```

## Pencatatan panggilan AWS IoT Events API dengan AWS CloudTrail

AWS IoT Events terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di AWS IoT Events. CloudTrail menangkap semua panggilan API untuk AWS IoT Events sebagai peristiwa, termasuk panggilan dari AWS IoT Events konsol dan dari panggilan kode ke AWS IoT Events APIs.

Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk AWS IoT Events. 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 Events, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat [Panduan AWS CloudTrail Pengguna](#).

### AWS IoT Events informasi di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas terjadi di AWS IoT Events, aktivitas tersebut dicatat dalam suatu CloudTrail peristiwa dengan peristiwa AWS layanan

lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di AWS akun Anda. Untuk informasi selengkapnya, lihat [Bekerja dengan riwayat CloudTrail Acara](#).

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk AWS IoT Events, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua AWS Wilayah. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan bertindak atas data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi lebih lanjut, lihat:

- [Membuat jejak untuk AWS akun Anda](#)
- [CloudTrail layanan dan integrasi yang didukung](#)
- [Mengonfigurasi notifikasi Amazon SNS untuk CloudTrail](#)
- [Menerima file CloudTrail log dari beberapa wilayah](#) dan [Menerima file CloudTrail log dari beberapa akun](#)

Setiap entri peristiwa atau log berisi informasi tentang entitas 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 [CloudTrailUserIdentity](#). AWS IoT Events tindakan didokumentasikan dalam [referensi AWS IoT Events API](#).

## Memahami entri file AWS IoT Events log

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. AWS 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.

Saat CloudTrail logging diaktifkan di AWS akun Anda, sebagian besar panggilan API yang dilakukan untuk AWS IoT Events tindakan dilacak dalam file CloudTrail log tempat mereka ditulis dengan catatan AWS layanan lainnya. CloudTrail menentukan kapan harus membuat dan menulis ke file baru berdasarkan periode waktu dan ukuran file.

Setiap entri log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas pengguna dalam entri log membantu Anda menentukan hal berikut:

- 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.

Anda dapat menyimpan berkas log dalam bucket Amazon S3 selama yang diinginkan, tetapi Anda juga dapat menentukan aturan siklus hidup Amazon S3 untuk mengarsipkan atau menghapus berkas log secara otomatis. Secara default, berkas log Anda dienkripsi dengan menggunakan enkripsi sisi server (SSE) Amazon S3.

Untuk diberi tahu saat pengiriman file log, Anda dapat mengonfigurasi CloudTrail untuk mempublikasikan notifikasi Amazon SNS saat file log baru dikirimkan. Untuk informasi lebih lanjut, lihat [Mengonfigurasi pemberitahuan Amazon SNS untuk CloudTrail](#).

Anda juga dapat menggabungkan file AWS IoT Events log dari beberapa AWS Wilayah dan beberapa AWS akun ke dalam satu bucket Amazon S3.

Untuk informasi selengkapnya, lihat [Menerima file CloudTrail log dari beberapa wilayah](#) dan [Menerima file CloudTrail log dari beberapa akun](#).

Contoh: DescribeDetector tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan DescribeDetector tindakan.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:sts::123456789012:assumed-role/Admin/bertholt-brecht",
    "accountId": "123456789012",
    "accessKeyId": "access-key-id",
```

```

"sessionContext": {
  "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-02-08T18:53:58Z"
  },
  "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/Admin",
    "accountId": "123456789012",
    "userName": "Admin"
  }
},
"eventTime": "2019-02-08T19:02:44Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "DescribeDetector",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-cli/1.15.65 Python/3.7.1 Darwin/16.7.0 botocore/1.10.65",
"requestParameters": {
  "detectorModelName": "pressureThresholdEventDetector-brecht",
  "keyValue": "1"
},
"responseElements": null,
"requestID": "00f41283-ea0f-4e85-959f-bee37454627a",
"eventID": "5eb0180d-052b-49d9-a289-0eb8d08d4c27",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Contoh: CreateDetectorModel tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan CreateDetectorModel tindakan.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-Lambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEvents-RoleForIotEvents-ABC123DEF456/IotEvents-Lambda",

```

```

"accountId": "123456789012",
"accessKeyId": "AKIAI44QH8DHBEXAMPLE",
"sessionContext": {
  "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-02-07T22:22:30Z"
  },
  "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABC123DEF456",
    "accountId": "123456789012",
    "userName": "IotEventsLambda-RoleForIotEvents-ABC123DEF456"
  }
},
"eventTime": "2019-02-07T23:54:43Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "CreateDetectorModel",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"requestParameters": {
  "detectorModelName": "myDetectorModel",
  "key": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "roleArn": "arn:aws:iam::123456789012:role/events_action_execution_role"
},
"responseElements": null,
"requestID": "cecfbfa1-e452-4fa6-b86b-89a89f392b66",
"eventID": "8138d46b-50a3-4af0-9c5e-5af5ef75ea55",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Contoh: CreateInput tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan CreateInput tindakan.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-Lambda",

```

```

    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABC123DEF456/IotEvents-Lambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABC123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABC123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:54:43Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "CreateInput",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "inputName": "batchputmessagedetectorupdated",
    "inputDescription": "batchputmessagedetectorupdated"
  },
  "responseElements": null,
  "requestID": "fb315af4-39e9-4114-94d1-89c9183394c1",
  "eventID": "6d8cf67b-2a03-46e6-bbff-e113a7bded1e",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}

```

Contoh: DeleteDetectorModel tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan DeleteDetectorModel tindakan.

```

{
  "eventVersion": "1.05",

```

```

"userIdentity": {
  "type": "AssumedRole",
  "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
  "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
  "accountId": "123456789012",
  "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
  "sessionContext": {
    "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2019-02-07T22:22:30Z"
    },
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AKIAI44QH8DHBEXAMPLE",
      "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
      "accountId": "123456789012",
      "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
    }
  }
},
"eventTime": "2019-02-07T23:54:11Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "DeleteDetectorModel",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"requestParameters": {
  "detectorModelName": "myDetectorModel"
},
"responseElements": null,
"requestID": "149064c1-4e24-4160-a5b2-1065e63ee2e4",
"eventID": "7669db89-dcc0-4c42-904b-f24b764dd808",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Contoh: DeleteInput tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan DeleteInput tindakan.

```

{
  "eventVersion": "1.05",

```

```
"userIdentity": {
  "type": "AssumedRole",
  "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
  "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
  "accountId": "123456789012",
  "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
  "sessionContext": {
    "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2019-02-07T22:22:30Z"
    },
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AKIAI44QH8DHBEXAMPLE",
      "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
      "accountId": "123456789012",
      "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
    }
  }
},
"eventTime": "2019-02-07T23:54:38Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "DeleteInput",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"errorCode": "ResourceNotFoundException",
"errorMessage": "Input of name: NoSuchInput not found",
"requestParameters": {
  "inputName": "NoSuchInput"
},
"responseElements": null,
"requestID": "ce6d28ac-5baf-423d-a5c3-afd009c967e3",
"eventID": "be0ef01d-1c28-48cd-895e-c3ff3172c08e",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}
```

## Contoh: DescribeDetectorModel tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan DescribeDetectorModel tindakan.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AAKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:54:20Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DescribeDetectorModel",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "detectorModelName": "myDetectorModel"
  },
  "responseElements": null,
  "requestID": "18a11622-8193-49a9-85cb-1fa6d3929394",
  "eventID": "1ad80ff8-3e2b-4073-ac38-9cb3385beb04",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

```
}
```

Contoh: DescribeInput tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan DescribeInput tindakan.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AAKIAI44QH8DHBEXAMPLE",
    "sessionContext": {

      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:56:09Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DescribeInput",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "inputName": "input_createinput"
  },
  "responseElements": null,
  "requestID": "3af641fa-d8af-41c9-ba77-ac9c6260f8b8",
  "eventID": "bc4e6cc0-55f7-45c1-b597-ec99aa14c81a",
  "eventType": "AwsApiCall",
```

```
"recipientAccountId": "123456789012"  
}
```

Contoh: DescribeLoggingOptions tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan DescribeLoggingOptions tindakan.

```
{  
  "eventVersion": "1.05",  
  "userIdentity": {  
    "type": "AssumedRole",  
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",  
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-  
ABCD123DEF456/IotEvents-EventsLambda",  
    "accountId": "123456789012",  
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",  
    "sessionContext": {  
      "attributes": {  
        "mfaAuthenticated": "false",  
        "creationDate": "2019-02-07T22:22:30Z"  
      },  
      "sessionIssuer": {  
        "type": "Role",  
        "principalId": "AKIAI44QH8DHBEXAMPLE",  
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-  
ABCD123DEF456",  
        "accountId": "123456789012",  
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"  
      }  
    }  
  },  
  "eventTime": "2019-02-07T23:53:23Z",  
  "eventSource": "iotevents.amazonaws.com",  
  "eventName": "DescribeLoggingOptions",  
  "awsRegion": "us-east-1",  
  "sourceIPAddress": "192.168.0.1",  
  "userAgent": "aws-internal/3",  
  "requestParameters": null,  
  "responseElements": null,  
  "requestID": "b624b6c5-aa33-41d8-867b-025ec747ee8f",  
  "eventID": "9c7ce626-25c8-413a-96e7-92b823d6c850",  
  "eventType": "AwsApiCall",  
}
```

```
"recipientAccountId": "123456789012"
}
```

Contoh: ListDetectorModels tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan ListDetectorModels tindakan.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:53:23Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "ListDetectorModels",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "nextToken": "CkZEZXRlY3Rvck1vZGVsMl9saXN0ZGV0ZWNoY3Jtb2R1bHN0ZXN0X2VlOWJkZTk1YT",
    "maxResults": 3
  },
  "responseElements": null,
}
```

```

"requestID": "6d70f262-da95-4bb5-94b4-c08369df75bb",
"eventID": "2d01a25c-d5c7-4233-99fe-ce1b8ec05516",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Contoh: ListDetectorModelVersions tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan ListDetectorModelVersions tindakan.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:53:33Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "ListDetectorModelVersions",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "detectorModelName": "myDetectorModel",

```

```

    "maxResults": 2
  },
  "responseElements": null,
  "requestID": "ebecb277-6bd8-44ea-8abd-fbf40ac044ee",
  "eventID": "fc6281a2-3fac-4e1e-98e0-ca6560b8b8be",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}

```

Contoh: ListDetectors tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan ListDetectors tindakan.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:53:54Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "ListDetectors",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {

```

```

    "detectorModelName": "batchputmessagedetectorinstancecreated",
    "stateName": "HIDDEN_DUE_TO_SECURITY_REASONS"
  },
  "responseElements": null,
  "requestID": "4783666d-1e87-42a8-85f7-22d43068af94",
  "eventID": "0d2b7e9b-afe6-4aef-afd2-a0bb1e9614a9",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}

```

Contoh: ListInputs tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan ListInputs tindakan.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:53:57Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "ListInputs",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",

```

```

"requestParameters": {
  "nextToken": "CkhjYW5hcnlfdGVzdF9pbmB1dF9saXN0ZGV0ZWN0b3Jtb2R1bHN0ZXN0ZDU3OGZ",
  "maxResults": 3
},
"responseElements": null,
"requestID": "dd6762a1-1f24-4e63-a986-5ea3938a03da",
"eventID": "c500f6d8-e271-4366-8f20-da4413752469",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Contoh: PutLoggingOptions tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan PutLoggingOptions tindakan.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:56:43Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "PutLoggingOptions",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",

```

```

"userAgent": "aws-internal/3",
"requestParameters": {
  "loggingOptions": {
    "roleArn": "arn:aws:iam::123456789012:role/logging__logging_role",
    "level": "INFO",
    "enabled": false
  }
},
"responseElements": null,
"requestID": "df570e50-fb19-4636-9ec0-e150a94bc52c",
"eventID": "3247f928-26aa-471e-b669-e4a9e6fbc42c",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Contoh: UpdateDetectorModel tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan UpdateDetectorModel tindakan.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      }
    },
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AKIAI44QH8DHBEXAMPLE",
      "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
      "accountId": "123456789012",
      "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
    }
  }
}

```

```

},
"eventTime": "2019-02-07T23:55:51Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "UpdateDetectorModel",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"requestParameters": {
  "detectorModelName": "myDetectorModel",
  "roleArn": "arn:aws:iam::123456789012:role/Events_action_execution_role"
},
"responseElements": null,
"requestID": "add29860-c1c5-4091-9917-d2ef13c356cf",
"eventID": "7baa9a14-6a52-47dc-aea0-3cace05147c3",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Contoh: UpdateInput tindakan untuk CloudTrail

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan UpdateInput tindakan.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      }
    },
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AKIAI44QH8DHBEXAMPLE",
      "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
      "accountId": "123456789012",
      "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
    }
  }
}

```

```

    }
  },
  "eventTime": "2019-02-07T23:53:00Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "UpdateInput",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "errorCode": "ResourceNotFoundException",
  "errorMessage": "Input of name: NoSuchInput not found",
  "requestParameters": {
    "inputName": "NoSuchInput",
    "inputDescription": "this is a description of an input"
  },
  "responseElements": null,
  "requestID": "58d5d2bb-4110-4c56-896a-ee9156009f41",
  "eventID": "c2df241a-fd53-4fd0-936c-ba309e5dc62d",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}

```

Contoh: BatchPutMessage tindakan untuk CloudTrail

AWS IoT Events dapat menggunakan CloudTrail integrasi untuk pencatatan API bidang data. Contoh ini menambahkan detail tentang peristiwa data melalui BatchPutMessage tindakan.

```

{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:PrincipalId",
    "arn": "arn:aws:sts::123456789012:assumed-role/my-iam-role/my-iam-role-
entity",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/my-iam-role",
        "accountId": "123456789012",
        "userName": "sample_user_name"
      }
    }
  },

```

```
        "attributes": {
            "creationDate": "2024-11-22T18:32:41Z",
            "mfaAuthenticated": "false"
        }
    },
    "eventTime": "2024-11-22T18:57:35Z",
    "eventSource": "iotevents.amazonaws.com",
    "eventName": "BatchPutMessage",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "3.239.107.128",
    "userAgent": "aws-internal/3",
    "requestParameters": {
        "messages": [
            {
                "messageId": "e306d827-b2e4-4439-9c86-411d4242a397",
                "payload": "HIDDEN_DUE_TO_SECURITY_REASONS",
                "inputName": "my_input_name"
            }
        ]
    },
    "responseElements": {
        "batchPutMessageErrorEntries": []
    },
    "requestID": "cefc6b63-9ccf-4e31-9177-4aec8e701bfe",
    "eventID": "b994b52c-6011-4e3c-ad5f-e784e732fde0",
    "readOnly": false,
    "resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::IoTEvents::Input",
            "ARN": "arn:aws:iotevents:us-east-1:123456789012:input/
my_input_name"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "123456789012",
    "eventCategory": "Data",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "iotevents.us-east-1.amazonaws.com"
    }
}
```

```
},
```

## Validasi kepatuhan untuk AWS IoT Events

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. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat [Dokumentasi AWS Keamanan](#).

## Ketahanan di AWS IoT Events

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Zona Ketersediaan yang terpisah dan terisolasi secara fisik, yang terhubung dengan jaringan latensi rendah, throughput tinggi, dan 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 AWS Wilayah dan Availability Zone, lihat [infrastruktur AWS global](#).

## Keamanan infrastruktur di AWS IoT Events

Sebagai layanan terkelola, AWS IoT Events dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat [Keamanan AWS Cloud](#). Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat [Perlindungan Infrastruktur dalam Kerangka Kerja](#) yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses AWS IoT Events melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

# AWS kuota layanan untuk sumber daya AWS IoT Events

Referensi Umum AWS Panduan ini menyediakan kuota default AWS IoT Events untuk AWS akun. Kecuali ditentukan, setiap kuota per AWS Wilayah. Untuk informasi selengkapnya, lihat [AWS IoT Events titik akhir dan kuota serta Service AWS Quotas](#) dalam Panduan.Referensi Umum AWS

Untuk meminta peningkatan kuota layanan, kirimkan kasus dukungan di konsol [Pusat Dukungan](#). Untuk informasi selengkapnya, lihat [Meminta peningkatan kuota](#) di Panduan Pengguna Service Quotas.

## Note

- Semua nama untuk model detektor dan input harus unik dalam akun.
- Anda tidak dapat mengubah nama untuk model dan input detektor setelah dibuat.

# Menandai sumber daya Anda AWS IoT Events

Untuk membantu Anda mengelola dan mengatur model dan input detektor, 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.

## Dasar-dasar tag

Tag memungkinkan Anda untuk mengkategorikan AWS IoT Events sumber daya Anda dengan cara yang berbeda, misalnya, berdasarkan tujuan, pemilik, atau lingkungan. Hal ini berguna jika Anda memiliki banyak sumber daya dengan jenis yang sama. Anda dapat mengidentifikasi sumber daya tertentu dengan cepat berdasarkan tag yang Anda tetapkan padanya.

Setiap tanda terdiri dari kunci dan nilai opsional, yang keduanya Anda tentukan. Misalnya, Anda dapat menentukan satu set tag untuk input Anda yang membantu Anda melacak perangkat yang mengirim input ini berdasarkan jenisnya. 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, menggunakan tag untuk mengkategorikan dan melacak biaya, dan juga menggunakan tag untuk mengontrol akses ke sumber daya Anda seperti yang dijelaskan dalam [Menggunakan tag dengan kebijakan IAM](#) di Panduan Pengembang AWS IoT .

Untuk kemudahan penggunaan, Editor Tag di Konsol Manajemen AWS menyediakan cara terpusat dan terpadu untuk membuat dan mengelola tag Anda. Untuk informasi selengkapnya, lihat [Memulai Editor Tag](#) di AWS Sumber Daya Penandaan dan Panduan Pengguna Editor Tag.

Anda juga dapat bekerja dengan tag menggunakan AWS CLI dan AWS IoT Events API. Anda dapat mengaitkan tag dengan model detektor dan input saat Anda membuatnya dengan menggunakan "Tags" bidang dalam perintah berikut:

- [CreateDetectorModel](#)
- [CreateInput](#)

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.

Untuk informasi selengkapnya, lihat [Praktik Terbaik untuk Menandai Sumber Daya AWS](#)

## Pembatasan dan batasan tanda

Batasan dasar berikut berlaku untuk tag:

- Jumlah maksimum tag per sumber daya – 50
- Panjang kunci maksimum - 127 karakter Unicode di UTF-8
- Panjang nilai maksimum – 255 karakter Unicode dalam UTF-8
- Kunci dan nilai tag sensitif huruf besar dan kecil.
- Jangan gunakan "aws : " awalan dalam nama atau nilai tag Anda karena itu 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 pemberian tag Anda digunakan di beberapa layanan dan sumber daya , ingatlah bahwa layanan lain mungkin memiliki pembatasan pada karakter yang diizinkan. Secara umum, karakter yang diizinkan adalah: 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 Events 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 dan nilai konteks syarat dalam kebijakan IAM hanya berlaku untuk tindakan AWS IoT Events tersebut di mana pengidentifikasi untuk sumber daya yang dapat ditandai adalah parameter yang diperlukan.

[Mengontrol akses menggunakan tag](#) di Panduan AWS Identity and Access Management Pengguna memiliki informasi tambahan tentang penggunaan tag. Bagian [Referensi kebijakan JSON IAM](#) dari panduan tersebut menyajikan sintaks, deskripsi, dan contoh terperinci elemen, variabel, dan logika evaluasi kebijakan JSON di IAM.

Kebijakan contoh berikut memberlakukan dua pembatasan berbasis tanda. Pengguna yang dibatasi oleh kebijakan ini:

- Tidak dapat memberikan sumber daya tag “env=prod” (dalam contoh, lihat baris `"aws:RequestTag/env" : "prod"`)
- Tidak dapat mengubah atau mengakses sumber daya yang memiliki tag “env=prod” yang ada (dalam contoh, lihat baris). `"aws:ResourceTag/env" : "prod"`

## JSON

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Deny",
    "Action": [
      "iotevents:CreateDetectorModel",
      "iotevents:CreateAlarmModel",
      "iotevents:CreateInput",
      "iotevents:TagResource"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/env": "prod"
      }
    }
  },
  {
    "Effect": "Deny",
    "Action": [
      "iotevents:DescribeDetectorModel",
      "iotevents:DescribeAlarmModel",
      "iotevents:UpdateDetectorModel",
      "iotevents:UpdateAlarmModel",
      "iotevents>DeleteDetectorModel",
      "iotevents>DeleteAlarmModel",
      "iotevents>ListDetectorModelVersions",
      "iotevents>ListAlarmModelVersions",
      "iotevents:UpdateInput",
      "iotevents:DescribeInput",
      "iotevents>DeleteInput",
      "iotevents:ListTagsForResource",
      "iotevents:TagResource",
      "iotevents:UntagResource",
      "iotevents:UpdateInputRouting"
    ],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "aws:ResourceTag/env": "prod"
      }
    }
  },
  {
```

```
        "Effect": "Allow",
        "Action": [
            "iotevents:*"
        ],
        "Resource": "*"
    }
]
```

Anda juga dapat menentukan beberapa nilai tag untuk kunci tag yang diberikan dengan melampirkannya dalam daftar, sebagai berikut.

```
"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.

# Pemecahan masalah AWS IoT Events

Panduan pemecahan masalah ini memberikan solusi untuk masalah umum yang mungkin Anda temui saat menggunakan AWS IoT Events. Jelajahi topik untuk mengidentifikasi dan menyelesaikan masalah dengan mendeteksi peristiwa, mengakses data, izin, integrasi layanan, konfigurasi perangkat, dan banyak lagi. Dengan saran pemecahan masalah untuk AWS IoT Events konsol, API, CLI, kesalahan, latensi, dan integrasi, panduan ini bertujuan untuk menyelesaikan masalah Anda dengan cepat sehingga Anda dapat membangun aplikasi berbasis peristiwa yang andal dan dapat diskalakan.

## Topik

- [Biasa AWS IoT Events masalah dan solusi](#)
- [Memecahkan masalah model detektor dengan menjalankan analisis di AWS IoT Events](#)

## Biasa AWS IoT Events masalah dan solusi

Lihat bagian berikut untuk memecahkan masalah kesalahan dan menemukan solusi yang mungkin untuk menyelesaikan masalah. AWS IoT Events

### Kesalahan

- [Kesalahan pembuatan model detektor](#)
- [Pembaruan dari model detektor yang dihapus](#)
- [Kegagalan pemicu tindakan \(saat memenuhi suatu kondisi\)](#)
- [Kegagalan pemicu tindakan \(saat melewati ambang batas\)](#)
- [Penggunaan status salah](#)
- [Pesan koneksi](#)
- [InvalidRequestException pesan](#)
- [Kesalahan Amazon CloudWatch Logs Action.setTimer](#)
- [Kesalahan CloudWatch payload Amazon](#)
- [Tipe data yang tidak kompatibel](#)
- [Gagal mengirim pesan ke AWS IoT Events](#)

## Kesalahan pembuatan model detektor

Saya mendapatkan kesalahan ketika saya mencoba membuat model detektor.

Saat Anda membuat model detektor, Anda harus mempertimbangkan batasan berikut.

- Hanya satu tindakan yang diperbolehkan di setiap `action` bidang.
- `condition` diperlukan untuk `transitionEvents`. Ini opsional untuk `OnEnter`, `OnInput`, dan `OnExit` acara.
- Jika `condition` bidang kosong, hasil evaluasi dari ekspresi kondisi setara `true` dengan.
- Hasil evaluasi dari ekspresi kondisi harus menjadi nilai Boolean. Jika hasilnya bukan nilai Boolean, itu setara dengan `false` dan tidak memicu `actions` atau transisi ke yang `nextState` ditentukan dalam acara tersebut.

Untuk informasi selengkapnya, lihat [AWS IoT Events pembatasan dan batasan model detektor](#).

## Pembaruan dari model detektor yang dihapus

Saya memperbarui atau menghapus model detektor beberapa menit yang lalu tetapi saya masih mendapatkan pembaruan status dari model detektor lama melalui pesan MQTT atau peringatan SNS.

Jika Anda memperbarui, menghapus, atau membuat ulang model detektor (lihat [UpdateDetectorModel](#)), ada penundaan sebelum semua instance detektor dihapus dan model baru digunakan. Selama waktu ini, input mungkin terus diproses oleh contoh model detektor versi sebelumnya. Anda mungkin terus menerima peringatan yang ditentukan oleh model detektor sebelumnya. Tunggu setidaknyanya tujuh menit sebelum Anda memeriksa ulang pembaruan atau melaporkan kesalahan.

## Kegagalan pemacu tindakan (saat memenuhi suatu kondisi)

Detektor gagal memicu tindakan atau transisi ke keadaan baru ketika kondisi terpenuhi.

Verifikasi bahwa hasil evaluasi dari ekspresi kondisional detektor adalah nilai Boolean. Jika hasilnya bukan nilai Boolean, itu setara dengan `false` dan tidak memicu `action` atau transisi ke yang `nextState` ditentukan dalam acara tersebut. Untuk informasi selengkapnya, lihat [Sintaks ekspresi bersyarat](#).

## Kegagalan pemicu tindakan (saat melewati ambang batas)

Detektor tidak memicu tindakan atau transisi peristiwa ketika variabel dalam ekspresi kondisional mencapai nilai tertentu.

Jika Anda memperbarui `setVariable` untuk `onInput`, `onEnter`, atau `onExit`, nilai baru tidak digunakan saat mengevaluasi apa pun `condition` selama siklus pemrosesan saat ini. Sebaliknya, nilai asli digunakan sampai siklus saat ini selesai. Anda dapat mengubah perilaku ini dengan menyetel `evaluationMethod` parameter dalam definisi model detektor. Ketika `evaluationMethod` diatur ke `SERIAL`, variabel diperbarui dan kondisi peristiwa dievaluasi dalam urutan bahwa peristiwa didefinisikan. Ketika `evaluationMethod` diatur ke `BATCH` (default), variabel diperbarui dan peristiwa dilakukan hanya setelah semua kondisi acara dievaluasi.

## Penggunaan status salah

Detektor memasuki status yang salah ketika saya mencoba mengirim pesan ke input dengan menggunakan `BatchPutMessage`.

Jika Anda gunakan [BatchPutMessage](#) untuk mengirim beberapa pesan ke input, urutan pemrosesan pesan atau input tidak dijamin. Untuk menjamin pemesanan, kirim pesan satu per satu dan tunggu setiap kali `BatchPutMessage` untuk mengakui keberhasilan.

## Pesan koneksi

Saya mendapatkan ( 'Connection aborted.', `error(54, 'Connection reset by peer' )` ) kesalahan ketika saya mencoba memanggil atau memanggil API.

Verifikasi bahwa OpenSSL menggunakan TLS 1.1 atau versi yang lebih baru untuk membuat koneksi. Ini harus menjadi default di sebagian besar distribusi Linux atau Windows versi 7 dan yang lebih baru. Pengguna macOS mungkin perlu memutakhirkan OpenSSL.

## InvalidRequestException pesan

Saya dapatkan `InvalidRequestException` ketika saya mencoba menelepon `CreateDetectorModel` dan `UpdateDetectorModel` API.

Periksa hal berikut untuk membantu menyelesaikan masalah. Untuk informasi selengkapnya, lihat [CreateDetectorModel](#) dan [UpdateDetectorModel](#).

- Pastikan Anda tidak menggunakan keduanya `seconds` dan `durationExpression` sebagai parameter pada `SetTimerAction` saat yang sama.
- Pastikan bahwa ekspresi string Anda `durationExpression` valid. Ekspresi string dapat berisi angka, variabel (`$variable.<variable-name>`), atau nilai input (`$input.<input-name>.<path-to-datum>`).

## Kesalahan Amazon CloudWatch Logs `Action.setTimer`

Anda dapat mengatur Amazon CloudWatch Logs untuk memantau instance model AWS IoT Events detektor. Berikut ini adalah kesalahan umum yang dihasilkan oleh AWS IoT Events, ketika Anda menggunakan `action.setTimer`.

- Kesalahan: Ekspresi durasi Anda untuk pengatur waktu bernama tidak `<timer-name>` dapat dievaluasi ke angka.

Pastikan bahwa ekspresi string Anda untuk `durationExpression` dapat dikonversi ke angka. Tipe data lainnya, seperti Boolean, tidak diperbolehkan.

- Kesalahan: Hasil evaluasi ekspresi durasi Anda untuk pengatur waktu bernama `<timer-name>` lebih besar dari 31622440. Untuk memastikan akurasi, pastikan ekspresi durasi Anda mengacu pada nilai antara 60-31622400.

Pastikan durasi timer Anda kurang dari atau sama dengan 31622400 detik. Hasil yang dievaluasi dari durasi dibulatkan ke bilangan bulat terdekat.

- Kesalahan: Hasil evaluasi ekspresi durasi Anda untuk pengatur waktu bernama `<timer-name>` kurang dari 60. Untuk memastikan akurasi, pastikan ekspresi durasi Anda mengacu pada nilai antara 60-31622400.

Pastikan durasi timer Anda lebih besar dari atau sama dengan 60 detik. Hasil yang dievaluasi dari durasi dibulatkan ke bilangan bulat terdekat.

- Kesalahan: Ekspresi durasi Anda untuk pengatur waktu bernama tidak `<timer-name>` dapat dievaluasi. Periksa nama variabel, nama input, dan jalur ke data untuk memastikan bahwa Anda merujuk ke variabel dan input yang ada.

Pastikan bahwa ekspresi string Anda mengacu pada variabel dan input yang ada. Ekspresi string dapat berisi angka, variabel (`$variable.<variable-name>`), dan nilai input (`$input.<input-name>.<path-to-datum>`).

- Kesalahan: Gagal mengatur timer bernama `<timer-name>`. Periksa ekspresi durasi Anda, dan coba lagi.

Lihat [SetTimerAction](#) tindakan untuk memastikan bahwa Anda menentukan parameter yang benar, dan kemudian mengatur timer lagi.

Untuk informasi selengkapnya, lihat [Mengaktifkan CloudWatch pencatatan Amazon saat mengembangkan model AWS IoT Events detektor](#).

## Kesalahan CloudWatch payload Amazon

Anda dapat mengatur Amazon CloudWatch Logs untuk memantau instance model AWS IoT Events detektor. Berikut ini adalah kesalahan umum dan peringatan yang dihasilkan oleh AWS IoT Events, ketika Anda mengonfigurasi payload tindakan.

- Kesalahan: Kami tidak dapat mengevaluasi ekspresi Anda untuk tindakan tersebut. Pastikan bahwa nama variabel, nama input, dan jalur ke data mengacu pada variabel yang ada dan nilai input. Juga, verifikasi bahwa ukuran muatan kurang dari 1 KB, ukuran muatan maksimum yang diizinkan.

Pastikan Anda memasukkan nama variabel yang benar, nama input, dan jalur ke data. Anda mungkin juga menerima pesan galat ini jika payload tindakan lebih besar dari 1 KB.

- Kesalahan: Kami tidak dapat mengurai ekspresi konten Anda untuk muatan. `<action-type>` Masukkan ekspresi konten dengan sintaks yang benar.

Ekspresi konten dapat berisi string (`'string'`), variabel (`$variable.variable-name`), nilai input (`$input.input-name.path-to-datum`), rangkaian string, dan string yang berisi `${}`

- Kesalahan: Ekspresi payload Anda `{expression}` tidak valid. Jenis payload yang ditentukan adalah JSON, jadi Anda harus menentukan ekspresi yang AWS IoT Events akan mengevaluasi ke string.

Jika jenis payload yang ditentukan adalah JSON, periksa AWS IoT Events terlebih dahulu apakah layanan dapat mengevaluasi ekspresi Anda ke string. Hasil yang dievaluasi tidak bisa berupa Boolean atau angka. Jika validasi gagal, Anda mungkin menerima kesalahan ini.

- Peringatan: Tindakan telah dijalankan, tetapi kami tidak dapat mengevaluasi ekspresi konten Anda untuk payload tindakan ke JSON yang valid. Jenis payload yang ditentukan adalah JSON.

Pastikan bahwa AWS IoT Events dapat mengevaluasi ekspresi konten Anda untuk payload tindakan ke JSON yang valid, jika Anda menentukan jenis payload sebagai JSON. AWS IoT Events menjalankan tindakan meskipun tidak dapat mengevaluasi ekspresi konten ke JSON yang valid.

Untuk informasi selengkapnya, lihat [Mengaktifkan CloudWatch pencatatan Amazon saat mengembangkan model AWS IoT Events detektor](#).

## Tipe data yang tidak kompatibel

Pesan: Tipe data yang tidak kompatibel [`<inferred-types>`] ditemukan `<reference>` dalam ekspresi berikut: `<expression>`

Anda mungkin menerima kesalahan ini karena salah satu alasan berikut:

- Hasil evaluasi referensi Anda tidak kompatibel dengan operan lain dalam ekspresi Anda.
- Jenis argumen yang diteruskan ke fungsi tidak didukung.

Saat Anda menggunakan referensi dalam ekspresi, periksa hal berikut:

- Bila Anda menggunakan referensi sebagai operan dengan satu atau beberapa operator, pastikan semua tipe data yang Anda referensikan kompatibel.

Misalnya, dalam ekspresi berikut, integer 2 adalah operan dari kedua operator `==` dan `&&`. Untuk memastikan bahwa operan kompatibel, `$variable.testVariable + 1` dan `$variable.testVariable` harus mereferensikan bilangan bulat atau desimal.

Selain itu, integer 1 adalah operan dari operator `+`. Oleh karena itu, `$variable.testVariable` harus referensi bilangan bulat atau desimal.

```
'$variable.testVariable + 1 == 2 && $variable.testVariable'
```

- Bila Anda menggunakan referensi sebagai argumen yang diteruskan ke fungsi, pastikan bahwa fungsi tersebut mendukung tipe data yang Anda referensikan.

Misalnya, `timeout("time-name")` fungsi berikut membutuhkan string dengan tanda kutip ganda sebagai argumen. Jika Anda menggunakan referensi untuk `timer-name` nilainya, Anda harus mereferensikan string dengan tanda kutip ganda.

```
timeout("timer-name")
```

### Note

Untuk `convert(type, expression)` fungsi, jika Anda menggunakan referensi untuk *type* nilai, hasil evaluasi dari referensi Anda harus `String`, `Decimal`, atau `Boolean`.

Untuk informasi selengkapnya, lihat [AWS IoT Events referensi untuk input dan variabel dalam ekspresi](#).

## Gagal mengirim pesan ke AWS IoT Events

Pesan: Gagal mengirim pesan ke Acara IoT

Anda mungkin mengalami kesalahan ini karena alasan berikut:

- Payload pesan masukan tidak berisi file. Input `attribute Key`
- Tidak Input `attribute Key` berada di jalur JSON yang sama seperti yang ditentukan dalam definisi input.
- Pesan masukan tidak cocok dengan skema, seperti yang didefinisikan dalam AWS IoT Events input.

### Note

Konsumsi data dari layanan lain juga akan mengalami kegagalan.

## Example

Misalnya di AWS IoT Core, AWS IoT aturan akan gagal dengan pesan berikut `Verify the Input Attribute key`.

Untuk mengatasi hal ini, pastikan bahwa skema pesan payload input sesuai dengan definisi AWS IoT Events Input dan lokasi cocok. Input `attribute Key` Untuk informasi selengkapnya, lihat [Buat masukan untuk model di AWS IoT Events](#) untuk mempelajari cara mendefinisikan AWS IoT Events Input.

# Memecahkan masalah model detektor dengan menjalankan analisis di AWS IoT Events

AWS IoT Events dapat menganalisis model detektor Anda dan menghasilkan hasil analisis tanpa mengirim data input ke model detektor Anda. AWS IoT Events melakukan serangkaian analisis yang dijelaskan di bagian ini untuk memeriksa model detektor Anda. Solusi pemecahan masalah lanjutan ini juga merangkum informasi diagnostik, termasuk tingkat keparahan dan lokasi, sehingga Anda dapat dengan cepat menemukan dan memperbaiki potensi masalah dalam model detektor Anda. Untuk informasi selengkapnya tentang jenis kesalahan diagnostik dan pesan untuk model detektor Anda, lihat [Analisis model detektor dan informasi diagnostik untuk AWS IoT Events](#).

Anda dapat menggunakan AWS IoT Events konsol, [API](#), [AWS Command Line Interface \(AWS CLI\)](#), atau [AWS SDK](#) untuk melihat pesan kesalahan diagnostik dari analisis model detektor Anda.

## Note

- Anda harus memperbaiki semua kesalahan sebelum dapat mempublikasikan model detektor Anda.
- Kami menyarankan Anda meninjau peringatan dan mengambil tindakan yang diperlukan sebelum Anda menggunakan model detektor Anda di lingkungan produksi. Jika tidak, model detektor mungkin tidak berfungsi seperti yang diharapkan.
- Anda dapat memiliki hingga 10 analisis dalam RUNNING status secara bersamaan.

Untuk mempelajari cara menganalisis model detektor Anda, lihat [Analisis model detektor untuk AWS IoT Events \(Konsol\)](#) atau [Analisis model detektor di AWS IoT Events \(AWS CLI\)](#).

## Topik

- [Analisis model detektor dan informasi diagnostik untuk AWS IoT Events](#)
- [Analisis model detektor untuk AWS IoT Events \(Konsol\)](#)
- [Analisis model detektor di AWS IoT Events \(AWS CLI\)](#)

## Analisis model detektor dan informasi diagnostik untuk AWS IoT Events


Analisis model detektor mengumpulkan informasi diagnostik berikut:

- **Tingkat** — Tingkat keparahan hasil analisis. Berdasarkan tingkat keparahan, hasil analisis terbagi dalam tiga kategori umum:
  - **Informasi (INFO)** — Hasil informasi memberi tahu Anda tentang bidang penting dalam model detektor Anda. Jenis hasil ini biasanya tidak memerlukan tindakan segera.
  - **Warning (WARNING)** — Hasil peringatan menarik perhatian khusus pada bidang yang dapat menyebabkan masalah pada model detektor Anda. Kami menyarankan Anda meninjau peringatan dan mengambil tindakan yang diperlukan sebelum Anda menggunakan model detektor Anda di lingkungan produksi. Jika tidak, model detektor mungkin tidak berfungsi seperti yang diharapkan.
  - **Error (ERROR)** - Hasil kesalahan memberi tahu Anda tentang masalah yang ditemukan dalam model detektor Anda. AWS IoT Events secara otomatis melakukan serangkaian analisis ini ketika Anda mencoba mempublikasikan model detektor. Anda harus memperbaiki semua kesalahan sebelum Anda dapat mempublikasikan model detektor.
- **Lokasi** - Berisi informasi yang dapat Anda gunakan untuk menemukan bidang dalam model detektor Anda yang dirujuk oleh hasil analisis. Lokasi biasanya mencakup nama negara, nama peristiwa transisi, nama acara, dan ekspresi (misalnya, `in state TemperatureCheck in onEnter in event Init in action setVariable`).
- **Jenis** — Jenis hasil analisis. Jenis analisis termasuk dalam kategori berikut:
  - **supported-actions**— AWS IoT Events dapat memanggil tindakan ketika peristiwa tertentu atau peristiwa transisi terdeteksi. Anda dapat menentukan tindakan bawaan untuk menggunakan timer atau mengatur variabel, atau mengirim data ke AWS layanan lain. Anda harus menentukan tindakan yang bekerja dengan AWS layanan lain di AWS Wilayah tempat AWS layanan tersedia.
  - **service-limits** Kuota layanan, juga dikenal sebagai batas, adalah jumlah maksimum atau minimum sumber daya layanan atau operasi untuk AWS akun Anda. Kecuali dinyatakan lain, setiap kuota adalah Region-specific. Tergantung pada kebutuhan bisnis Anda, Anda dapat memperbarui model detektor Anda untuk menghindari menghadapi batasan atau meminta peningkatan kuota. Anda dapat meminta kenaikan untuk beberapa kuota, dan kuota lainnya tidak dapat ditingkatkan. Untuk informasi lebih lanjut, lihat [Kuota](#).
- **structure**— Model detektor harus memiliki semua komponen yang diperlukan seperti status dan mengikuti struktur yang AWS IoT Events mendukung. Model detektor harus memiliki setidaknya satu status dan kondisi yang mengevaluasi data input yang masuk untuk mendeteksi peristiwa penting. Ketika suatu peristiwa terdeteksi, model detektor bertransisi ke status berikutnya dan dapat memanggil tindakan. Peristiwa ini dikenal sebagai peristiwa transisi. Peristiwa transisi harus mengarahkan status berikutnya untuk masuk.

- **expression-syntax**— AWS IoT Events menyediakan beberapa cara untuk menentukan nilai saat Anda membuat dan memperbarui model detektor. Anda dapat menggunakan template literal, operator, fungsi, referensi, dan substitusi dalam ekspresi. Anda dapat menggunakan ekspresi untuk menentukan nilai literal, atau AWS IoT Events dapat mengevaluasi ekspresi sebelum Anda menentukan nilai tertentu. Ekspresi Anda harus mengikuti sintaks yang diperlukan. Untuk informasi selengkapnya, lihat [Ekspresi untuk memfilter, mengubah, dan memproses data peristiwa](#).

Ekspresi Model Detektor AWS IoT Events dapat mereferensikan data atau sumber daya tertentu.

- **data-type**— AWS IoT Events mendukung tipe data integer, desimal, string, dan Boolean. Jika AWS IoT Events dapat secara otomatis mengonversi data dari satu tipe data ke tipe data lainnya selama evaluasi ekspresi, tipe data ini kompatibel.

 Note

- Integer dan desimal adalah satu-satunya tipe data yang kompatibel yang didukung oleh AWS IoT Events
- AWS IoT Events tidak dapat mengevaluasi ekspresi aritmatika karena tidak AWS IoT Events dapat mengonversi bilangan bulat menjadi string.

- **referenced-data**— Anda harus menentukan data yang direferensikan dalam model detektor Anda sebelum Anda dapat menggunakan data. Misalnya, jika Anda ingin mengirim data ke tabel DynamoDB, Anda harus menentukan variabel yang mereferensikan nama tabel sebelum Anda dapat menggunakan variabel dalam ekspresi (`$.variable.TableName`
- **referenced-resource**— Sumber daya yang digunakan model detektor harus tersedia. Anda harus menentukan sumber daya sebelum Anda dapat menggunakannya. Misalnya, Anda ingin membuat model detektor untuk memantau suhu rumah kaca. Anda harus menentukan input (`$.input.TemperatureInput`) untuk merutekan data suhu yang masuk ke model detektor Anda sebelum Anda dapat menggunakan `$.input.TemperatureInput.sensorData.temperature` untuk mereferensikan suhu.

Lihat bagian berikut untuk memecahkan masalah kesalahan dan menemukan solusi yang mungkin dari analisis model detektor Anda.

## Memecahkan masalah kesalahan model detektor di AWS IoT Events

Jenis kesalahan yang dijelaskan di atas memberikan informasi diagnostik tentang model detektor dan sesuai dengan pesan yang mungkin Anda ambil. Gunakan pesan ini dan solusi yang disarankan untuk memecahkan masalah kesalahan dengan model detektor Anda.

Pesan dan solusi

- [Lokasi](#)
- [tindakan yang didukung](#)
- [batas layanan](#)
- [struktur](#)
- [sintaks ekspresi](#)
- [tipe data](#)
- [data yang direferensikan](#)
- [sumber daya referensi](#)

### Lokasi

Hasil analisis dengan informasi tentang `Location`, sesuai dengan pesan kesalahan berikut:

- Pesan - Berisi informasi tambahan tentang hasil analisis. Ini bisa berupa informasi, peringatan, atau pesan kesalahan.

Solusi: Anda mungkin menerima pesan galat ini jika Anda menetapkan tindakan yang AWS IoT Events saat ini tidak mendukung. Untuk daftar tindakan yang didukung, lihat [Tindakan yang didukung untuk menerima data dan memicu tindakan di AWS IoT Events](#).

### tindakan yang didukung

Hasil analisis dengan informasi tentang `supported-actions`, sesuai dengan pesan kesalahan berikut:

- Pesan: Jenis tindakan tidak valid yang ada dalam definisi tindakan: *action-definition*

Solusi: Anda mungkin menerima pesan galat ini jika Anda menetapkan tindakan yang AWS IoT Events saat ini tidak mendukung. Untuk daftar tindakan yang didukung, lihat [Tindakan yang didukung untuk menerima data dan memicu tindakan di AWS IoT Events](#).

- Pesan: `DetectorModel` definisi memiliki `aws-service` tindakan, tetapi `aws-service` layanan tidak didukung di wilayah tersebut `region-name`.

Solusi: Anda mungkin menerima pesan galat ini jika tindakan yang Anda tentukan didukung oleh AWS IoT Events, tetapi tindakan tersebut tidak tersedia di Wilayah Anda saat ini. Ini mungkin terjadi ketika Anda mencoba mengirim data ke AWS layanan yang tidak tersedia di Wilayah. Anda juga harus memilih Wilayah yang sama untuk keduanya AWS IoT Events dan AWS layanan yang Anda gunakan.

## batas layanan

Hasil analisis dengan informasi tentang `service-limits`, sesuai dengan pesan kesalahan berikut:

- Pesan: Ekspresi Konten yang diizinkan dalam payload melebihi batas `content-expression-size` byte jika terjadi `event-name` dalam status. `state-name`

Solusi: Anda mungkin menerima pesan galat ini jika ekspresi konten untuk muatan tindakan Anda lebih besar dari 1024 byte. Ukuran ekspresi konten untuk payload bisa sampai 1024 byte.

- Pesan: Jumlah status yang diizinkan dalam definisi model detektor melebihi batas `states-per-detector-model`.

Solusi: Anda mungkin menerima pesan kesalahan ini jika model detektor Anda memiliki lebih dari 20 status. Model detektor dapat memiliki hingga 20 status.

- Pesan: Durasi pengatur waktu `timer-name` harus setidaknya `minimum-timer-duration` beberapa detik.

Solusi: Anda mungkin menerima pesan galat ini jika durasi timer Anda kurang dari 60 detik. Kami merekomendasikan bahwa durasi timer adalah antara 60 dan 31622400 detik. Jika Anda menentukan ekspresi untuk durasi timer Anda, hasil evaluasi dari ekspresi durasi dibulatkan ke bawah ke bilangan bulat terdekat.

- Pesan: Jumlah tindakan yang diizinkan per peristiwa melebihi batas `actions-per-event` dalam definisi model detektor

Solusi: Anda mungkin menerima pesan galat ini jika acara memiliki lebih dari 10 tindakan. Anda dapat memiliki hingga 10 tindakan untuk setiap peristiwa dalam model detektor Anda.

- Pesan: Jumlah peristiwa transisi yang diizinkan per status melebihi batas `transition-events-per-state` dalam definisi model detektor.

Solusi: Anda mungkin menerima pesan galat ini jika status memiliki lebih dari 20 peristiwa transisi. Anda dapat memiliki hingga 20 peristiwa transisi untuk setiap status dalam model detektor Anda.

- Pesan: Jumlah peristiwa yang diizinkan per status melebihi batas *events-per-state* dalam definisi model detektor

Solusi: Anda mungkin menerima pesan galat ini jika status memiliki lebih dari 20 peristiwa. Anda dapat memiliki hingga 20 acara untuk setiap status dalam model detektor Anda.

- Pesan: Jumlah maksimum model detektor yang dapat dikaitkan dengan satu input mungkin telah mencapai batas. Input *input-name* digunakan dalam rute model *detector-models-per-input* detektor.

Solusi: Anda mungkin menerima pesan peringatan ini jika Anda mencoba merutekan input ke lebih dari 10 model detektor. Anda dapat memiliki hingga 10 model detektor berbeda yang terkait dengan model detektor tunggal.

## struktur

Hasil analisis dengan informasi tentang `structure`, sesuai dengan pesan kesalahan berikut:

- Pesan: Tindakan mungkin hanya memiliki satu jenis yang ditentukan, tetapi menemukan tindakan dengan *number-of-types* tipe. Harap dibagi menjadi Tindakan terpisah.

Solusi: Anda mungkin menerima pesan galat ini jika Anda menetapkan dua atau beberapa tindakan dalam satu bidang dengan menggunakan operasi API untuk membuat atau memperbarui model detektor Anda. Anda dapat menentukan array `Action` objek. Pastikan Anda mendefinisikan setiap tindakan sebagai objek terpisah.

- Pesan: `TransitionEvent` *transition-event-name* Transisi ke keadaan yang tidak ada. *state-name*

Solusi: Anda mungkin menerima pesan galat ini jika AWS IoT Events tidak dapat menemukan status berikutnya yang direferensikan oleh peristiwa transisi Anda. Pastikan bahwa status berikutnya ditentukan dan Anda memasukkan nama negara yang benar.

- Pesan: `DetectorModelDefinition` Memiliki nama negara bersama: status ditemukan *state-name* dengan *number-of-states* pengulangan.

Solusi: Anda mungkin menerima pesan galat ini jika Anda menggunakan nama yang sama untuk satu atau beberapa status. Pastikan Anda memberikan nama unik untuk setiap status dalam model

detektor Anda. Nama negara harus memiliki 1-128 karakter. Karakter yang valid: a-z, A-Z, 0-9, \_ (garis bawah), dan - (tanda hubung).

- Pesan: Awal Definisi StateName *initial-state-name* tidak sesuai dengan Status yang ditentukan.

Solusi: Anda mungkin menerima pesan galat ini jika nama status awal salah. Model detektor tetap dalam keadaan awal (mulai) sampai input tiba. Setelah input tiba, model detektor segera beralih ke status berikutnya. Pastikan bahwa nama negara awal adalah nama negara yang ditentukan dan Anda memasukkan nama yang benar.

- Pesan: Definisi Model Detektor harus menggunakan setidaknya satu Input dalam suatu kondisi.

Solusi: Anda mungkin menerima kesalahan ini jika Anda tidak menentukan input dalam suatu kondisi. Anda harus menggunakan setidaknya satu input dalam setidaknya satu kondisi. Jika tidak, AWS IoT Events tidak mengevaluasi data yang masuk.

- Pesan: Hanya satu detik dan durationExpression dapat diatur. SetTimer

Solusi: Anda mungkin menerima pesan galat ini jika Anda menggunakan keduanya seconds dan durationExpression untuk timer Anda. Pastikan Anda menggunakan salah satu seconds atau durationExpression sebagai parameterSetTimerAction. Untuk informasi selengkapnya, lihat [SetTimerAction](#) di dalam Referensi API AWS IoT Events .

- Pesan: Tindakan dalam model detektor Anda tidak dapat dijangkau. Periksa kondisi yang memulai tindakan.

Solusi: Jika tindakan dalam model detektor Anda tidak dapat dijangkau, kondisi acara dievaluasi menjadi false. Periksa kondisi acara yang berisi tindakan, untuk memastikan bahwa itu mengevaluasi menjadi benar. Ketika kondisi acara dievaluasi menjadi benar, tindakan harus dapat dijangkau.

- Pesan: Atribut input sedang dibaca, tetapi ini mungkin disebabkan oleh kedaluwarsa timer.

Solusi: Nilai atribut input dapat dibaca ketika salah satu dari berikut ini terjadi:

- Nilai input baru telah diterima.
- Ketika timer di detektor telah kedaluwarsa.

Untuk memastikan bahwa atribut input sedang dievaluasi hanya ketika nilai baru untuk input tersebut diterima, sertakan panggilan ke `triggerType("Message")` fungsi dalam kondisi Anda sebagai berikut:

Kondisi asli yang sedang dievaluasi dalam model detektor:

```
if ($input.HeartBeat.status == "OFFLINE")
```

akan menjadi mirip dengan yang berikut:

```
if ( triggerType("MESSAGE") && $input.HeartBeat.status == "OFFLINE")
```

di mana panggilan ke `triggerType("Message")` fungsi datang sebelum input awal yang disediakan dalam kondisi. Dengan menggunakan teknik ini, `triggerType("Message")` fungsi akan mengevaluasi menjadi benar dan memenuhi kondisi menerima nilai input baru. Untuk informasi selengkapnya tentang penggunaan `triggerType` fungsi, cari `triggerType` di bagian [Ekspres](#) di Panduan AWS IoT Events Pengembang

- Pesan: Status dalam model detektor Anda tidak dapat dijangkau. Periksa kondisi yang akan menyebabkan transisi ke keadaan yang diinginkan.

Solusi: Jika status dalam model detektor Anda tidak dapat dijangkau, kondisi yang menyebabkan transisi masuk ke status tersebut dievaluasi menjadi false. Periksa apakah kondisi transisi yang masuk ke keadaan yang tidak dapat dijangkau dalam model detektor Anda mengevaluasi ke true, sehingga status yang diinginkan dapat dijangkau.

- Pesan: Timer kedaluwarsa dapat menyebabkan jumlah pesan yang tidak terduga dikirim.

Solusi: Untuk mencegah model detektor Anda masuk ke dalam keadaan tak terbatas mengirim pesan dalam jumlah tak terduga karena pengatur waktu telah kedaluwarsa, pertimbangkan untuk menggunakan panggilan ke `triggerType("Message")` fungsi tersebut, dalam kondisi model detektor Anda sebagai berikut:

Kondisi asli yang sedang dievaluasi dalam model detektor:

```
if (timeout("awake"))
```

akan diubah menjadi kondisi yang terlihat mirip dengan berikut ini:

```
if (triggerType("MESSAGE") && timeout("awake"))
```

di mana panggilan ke `triggerType("Message")` fungsi datang sebelum input awal yang disediakan dalam kondisi.

Perubahan ini mencegah memulai tindakan pengatur waktu di detektor Anda, mencegah pengulangan pesan tak terbatas yang dikirim. Untuk informasi selengkapnya tentang cara menggunakan tindakan pengatur waktu di detektor, lihat halaman [Menggunakan tindakan bawaan](#) dari Panduan AWS IoT Events Pengembang

## sintaks ekspresi

Hasil analisis dengan informasi tentang `expression-syntax`, sesuai dengan pesan kesalahan berikut:

- Pesan: Ekspresi payload Anda `{expression}` tidak valid. Jenis payload yang ditentukan adalah JSON, jadi Anda harus menentukan ekspresi yang AWS IoT Events akan mengevaluasi ke string.

Solusi: Jika jenis payload yang ditentukan adalah JSON, periksa AWS IoT Events terlebih dahulu apakah layanan dapat mengevaluasi ekspresi Anda ke string. Hasil yang dievaluasi tidak bisa berupa Boolean atau angka. Jika validasi tidak berhasil, Anda mungkin menerima kesalahan ini.

- Pesan: `SetVariableAction.value` harus berupa ekspresi. Gagal mengurai nilai "`variable-value`"

Solusi: Anda dapat menggunakan `SetVariableAction` untuk mendefinisikan variabel dengan `name` dan `value`. Itu `value` bisa berupa string, angka, atau nilai Boolean. Anda juga dapat menentukan ekspresi untuk `value`. Untuk informasi selengkapnya, lihat [SetVariableAction](#), di Referensi AWS IoT Events API.

- Pesan: Kami tidak dapat mengurai ekspresi atribut (`attribute-name`) Anda untuk tindakan DynamoDB. Masukkan ekspresi dengan sintaks yang benar.

Solusi: Anda harus menggunakan ekspresi untuk semua parameter `DynamoDBAction` di template substitusi. Untuk informasi selengkapnya, lihat [DynamodBaction](#) di Referensi API AWS IoT Events

- Pesan: Kami tidak dapat mengurai ekspresi `TableName` Anda untuk tindakan `DynamoDBv2`. Masukkan ekspresi dengan sintaks yang benar.

Solusi: `tableName` In `DynamoDBv2Action` harus berupa string. Anda harus menggunakan ekspresi untuk `tableName`. Ekspresi menerima templat literal, operator, fungsi, referensi, dan

substitusi. Untuk informasi selengkapnya, lihat [DynamoDBv2Action](#) di dalam Referensi API AWS IoT Events .

- Pesan: Kami tidak dapat mengevaluasi ekspresi Anda ke JSON yang valid. Tindakan DynamoDBv2 hanya mendukung jenis payload JSON.

Solusi: Jenis payload untuk DynamoDBv2 harus JSON. Pastikan itu AWS IoT Events dapat mengevaluasi ekspresi konten Anda untuk payload ke JSON yang valid. Untuk informasi selengkapnya, lihat [DynamoDBv2Action](#), di Referensi AWS IoT Events API.

- Pesan: Kami tidak dapat mengurai ekspresi konten Anda untuk muatan. *action-type* Masukkan ekspresi konten dengan sintaks yang benar.

Solusi: Ekspresi konten dapat berisi string ('*string*'), variabel ( $\$$ variable. *variable-name*), nilai masukan ( $\$$  input. *input-name.path-to-datum*), rangkaian string, dan string yang berisi.  $\{\}$

- Pesan: Muatan yang Disesuaikan harus tidak kosong.

Solusi: Anda mungkin menerima pesan galat ini, jika memilih Payload khusus untuk tindakan Anda dan tidak memasukkan ekspresi konten di AWS IoT Events konsol. Jika Anda memilih Payload kustom, Anda harus memasukkan ekspresi konten di bawah Payload kustom. Untuk informasi selengkapnya, lihat [Payload](#) di Referensi AWS IoT Events API.

- Pesan: Gagal mengurai ekspresi durasi '*duration-expression*' untuk timer '*timer-name*'.

Solusi: Hasil evaluasi ekspresi durasi Anda untuk pengatur waktu harus bernilai antara 60—31622400. Hasil yang dievaluasi dari durasi dibulatkan ke bilangan bulat terdekat.

- Pesan: Gagal mengurai ekspresi '*expression*' untuk *action-name*

Solusi: Anda mungkin menerima pesan ini jika ekspresi untuk tindakan yang ditentukan memiliki sintaks yang salah. Pastikan Anda memasukkan ekspresi dengan sintaks yang benar. Untuk informasi selengkapnya, lihat [Sintaks untuk memfilter data perangkat dan menentukan tindakan di AWS IoT Events](#).

- Pesan: *fieldName* Untuk Anda IotSiteWiseAction tidak dapat diuraikan. Anda harus menggunakan sintaks yang benar dalam ekspresi Anda.

Solusi: Anda mungkin menerima kesalahan ini jika AWS IoT Events tidak dapat mengurai *fieldName* for IotSiteWiseAction Anda. Pastikan *fieldName* menggunakan ekspresi yang AWS IoT Events dapat mengurai. Untuk informasi selengkapnya, lihat [IotSiteWiseAction](#) di dalam Referensi API AWS IoT Events .

## tipe data

Hasil analisis dengan informasi tentang data-type, sesuai dengan pesan kesalahan berikut:

- Pesan: Ekspresi durasi *duration-expression* untuk timer *timer-name* tidak valid, itu harus mengembalikan nomor.

Solusi: Anda mungkin menerima pesan galat ini jika AWS IoT Events tidak dapat mengevaluasi ekspresi durasi untuk timer Anda ke nomor. Pastikan bahwa Anda *durationExpression* dapat dikonversi ke nomor. Tipe data lainnya, seperti Boolean, tidak didukung.

- Pesan: Ekspresi *condition-expression* bukan ekspresi kondisi yang valid.

Solusi: Anda mungkin menerima pesan galat ini jika AWS IoT Events tidak dapat mengevaluasi nilai Boolean Anda *condition-expression*. Nilai Boolean harus salah satu TRUE atau FALSE. Pastikan bahwa ekspresi kondisi Anda dapat dikonversi ke nilai Boolean. Jika hasilnya bukan nilai Boolean, itu setara dengan FALSE dan tidak memanggil tindakan atau transisi ke yang *nextState* ditentukan dalam acara tersebut.

- Pesan: Tipe data yang tidak kompatibel [*inferred-types*] ditemukan *reference* dalam ekspresi berikut: *expression*

Solusi: Solusi: Semua ekspresi untuk atribut input atau variabel yang sama dalam model detektor harus mereferensikan tipe data yang sama.

Gunakan informasi berikut untuk menyelesaikan masalah:

- Bila Anda menggunakan referensi sebagai operan dengan satu atau beberapa operator, pastikan semua tipe data yang Anda referensikan kompatibel.

Misalnya, dalam ekspresi berikut, integer 2 adalah operan dari kedua operator `==` dan `&&`. Untuk memastikan bahwa operan kompatibel, `$variable.testVariable + 1` dan `$variable.testVariable` harus mereferensikan bilangan bulat atau desimal.


Selain itu, integer 1 adalah operan dari operator `+`. Oleh karena itu, `$variable.testVariable` harus referensi bilangan bulat atau desimal.

```
'$variable.testVariable + 1 == 2 && $variable.testVariable'
```

- Bila Anda menggunakan referensi sebagai argumen yang diteruskan ke fungsi, pastikan bahwa fungsi tersebut mendukung tipe data yang Anda referensikan.

Misalnya, `timeout("time-name")` fungsi berikut membutuhkan string dengan tanda kutip ganda sebagai argumen. Jika Anda menggunakan referensi untuk `timer-name` nilainya, Anda harus mereferensikan string dengan tanda kutip ganda.

```
timeout("timer-name")
```

 Note

Untuk `convert(type, expression)` fungsi, jika Anda menggunakan referensi untuk `type` nilai, hasil evaluasi dari referensi Anda harus `String`, `Decimal`, atau `Boolean`.

Untuk informasi selengkapnya, lihat [AWS IoT Events referensi untuk input dan variabel dalam ekspresi](#).

- Pesan: Tipe data yang tidak kompatibel [*inferred-types*] digunakan dengan *reference*. Ini dapat menyebabkan kesalahan runtime.

Solusi: Anda mungkin menerima pesan peringatan ini jika dua ekspresi untuk atribut input yang sama atau referensi variabel dua tipe data. Pastikan bahwa ekspresi Anda untuk atribut input atau variabel yang sama mereferensikan tipe data yang sama dalam model detektor.

- Pesan: Tipe data [*inferred-types*] yang Anda masukkan untuk operator [*operator*] tidak kompatibel untuk ekspresi berikut: '*expression*'

Solusi: Anda mungkin menerima pesan galat ini jika ekspresi Anda menggabungkan tipe data yang tidak kompatibel dengan operator tertentu. Misalnya, dalam ekspresi berikut, operator `+` kompatibel dengan tipe data `Integer`, `Decimal`, dan `String`, tetapi bukan operan tipe data `Boolean`.

```
true + false
```

Anda harus memastikan bahwa tipe data yang Anda gunakan dengan operator kompatibel.

- Pesan: Tipe data [*inferred-types*] yang ditemukan *input-attribute* tidak kompatibel dan dapat menyebabkan kesalahan runtime.

Solusi: Anda mungkin menerima pesan galat ini jika dua ekspresi untuk atribut input yang sama mereferensikan dua tipe data untuk status, atau untuk status `OnInputLifecycle` dan `OnExitLifecycle` status. `OnEnterLifecycle` Pastikan ekspresi Anda dalam

`OnEnterLifecycle` (atau, keduanya `OnInputLifecycle` dan `OnExitLifecycle`) mereferensikan tipe data yang sama untuk setiap status model detektor Anda.

- Pesan: Ekspresi payload [*expression*] tidak valid. Tentukan ekspresi yang akan mengevaluasi string saat runtime karena jenis payload adalah format JSON.

Solusi: Anda mungkin menerima kesalahan ini jika jenis payload yang Anda tentukan adalah JSON, tetapi tidak AWS IoT Events dapat mengevaluasi ekspresinya ke String. Pastikan hasil yang dievaluasi adalah String, bukan Boolean atau angka.

- Pesan: Ekspresi interpolasi Anda {*interpolated-expression*} harus mengevaluasi ke integer atau nilai Boolean saat runtime. Jika tidak, ekspresi payload {*payload-expression*} Anda tidak akan dapat diuraikan saat runtime sebagai JSON yang valid.

Solusi: Anda mungkin menerima pesan galat ini jika AWS IoT Events tidak dapat mengevaluasi ekspresi interpolasi Anda ke bilangan bulat atau nilai Boolean. Pastikan ekspresi interpolasi Anda dapat dikonversi ke integer atau nilai Boolean, karena tipe data lain, seperti tring, tidak didukung.

- Pesan: Jenis ekspresi di `IotSitetwiseAction` bidang *expression* didefinisikan sebagai tipe *defined-type* dan disimpulkan sebagai tipe *inferred-type*. Tipe yang ditentukan dan tipe yang disimpulkan harus sama.

Solusi: Anda mungkin menerima pesan galat ini jika ekspresi Anda di `propertyValue` of `IotSitetwiseAction` memiliki tipe data yang ditentukan secara berbeda dari tipe data yang disimpulkan oleh AWS IoT Events. Pastikan Anda menggunakan tipe data yang sama untuk semua contoh ekspresi ini dalam model detektor Anda.

- Pesan: Tipe data [*inferred-types*] yang digunakan untuk `setTimer` tindakan tidak dievaluasi Integer untuk ekspresi berikut: *expression*

Solusi: Anda mungkin menerima pesan galat ini jika tipe data yang disimpulkan untuk ekspresi durasi Anda bukan Integer atau Desimal. Pastikan Anda `durationExpression` dapat dikonversi ke nomor. Tipe data lainnya, seperti Boolean dan String, tidak didukung.

- Pesan: Tipe data [*inferred-types*] yang digunakan dengan operan operator perbandingan [*operator*] tidak kompatibel dalam ekspresi berikut: *expression*

Solusi: Tipe data yang disimpulkan untuk operan *operator* dalam ekspresi bersyarat (*expression*) model detektor Anda tidak cocok. Operan harus digunakan dengan tipe data yang cocok di semua bagian lain dari model detektor Anda.

**Tip**

Anda dapat menggunakan `convert` untuk mengubah tipe data ekspresi dalam model detektor Anda. Untuk informasi selengkapnya, lihat [Fungsi untuk digunakan dalam AWS IoT Events ekspresi](#).

**data yang direferensikan**

Hasil analisis dengan informasi tentang `referenced-data`, sesuai dengan pesan kesalahan berikut:

- Pesan: Timer rusak yang terdeteksi: timer *timer-name* digunakan dalam ekspresi tetapi tidak pernah disetel.

Solusi: Anda mungkin menerima pesan galat ini jika Anda menggunakan timer yang tidak disetel. Anda harus mengatur timer sebelum menggunakannya dalam ekspresi. Juga, pastikan Anda memasukkan nama timer yang benar.

- Pesan: Variabel rusak yang terdeteksi: variabel *variable-name* digunakan dalam ekspresi tetapi tidak pernah disetel.

Solusi: Anda mungkin menerima pesan galat ini jika Anda menggunakan variabel yang tidak disetel. Anda harus menetapkan variabel sebelum Anda menggunakannya dalam ekspresi. Juga, pastikan bahwa Anda memasukkan nama variabel yang benar.

- Pesan: Variabel rusak yang terdeteksi: variabel digunakan dalam ekspresi sebelum disetel ke nilai.

Solusi: Setiap variabel harus ditetapkan ke nilai sebelum dapat dievaluasi dalam ekspresi. Tetapkan nilai variabel sebelum setiap penggunaan sehingga nilainya dapat diambil. Juga, pastikan bahwa Anda memasukkan nama variabel yang benar.

**sumber daya referensi**

Hasil analisis dengan informasi tentang `referenced-resource`, sesuai dengan pesan kesalahan berikut:

- Pesan: Definisi Model Detektor berisi referensi ke Input yang tidak ada.

Solusi: Anda mungkin menerima pesan galat ini jika Anda menggunakan ekspresi untuk mereferensikan masukan yang tidak ada. Pastikan ekspresi Anda mereferensikan input yang ada dan masukkan nama input yang benar. Jika Anda tidak memiliki masukan, buat terlebih dahulu.

- Pesan: Definisi Model Detektor berisi tidak valid InputName: *input-name*

Solusi: Anda mungkin menerima pesan galat ini jika model detektor Anda berisi nama input yang tidak valid. Pastikan Anda memasukkan nama input yang benar. Nama input harus memiliki 1-128 karakter. Karakter yang valid: a-z, A-Z, 0-9, \_ (garis bawah), dan - (tanda hubung).

## Analisis model detektor untuk AWS IoT Events (Konsol)

AWS IoT Events memungkinkan Anda untuk memantau dan bereaksi terhadap data IoT dengan mendeteksi peristiwa dan memicu tindakan dengan API. AWS IoT Events Langkah-langkah berikut menggunakan AWS IoT Events konsol untuk menganalisis model detektor.

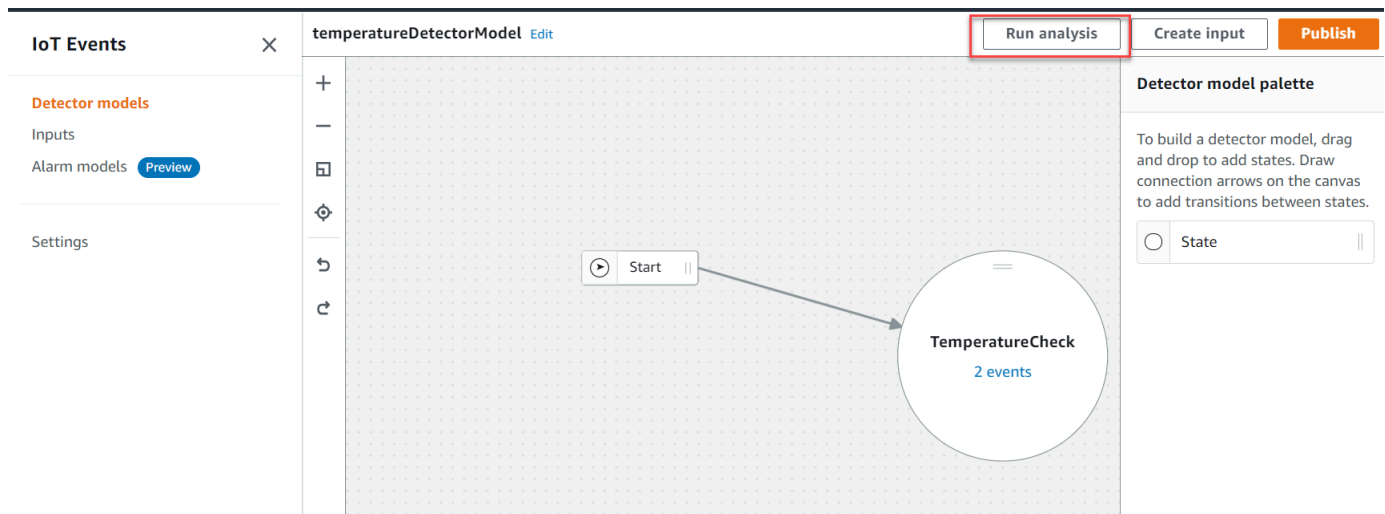
### Note

Setelah AWS IoT Events mulai menganalisis model detektor Anda, Anda memiliki waktu hingga 24 jam untuk mengambil hasil analisis.

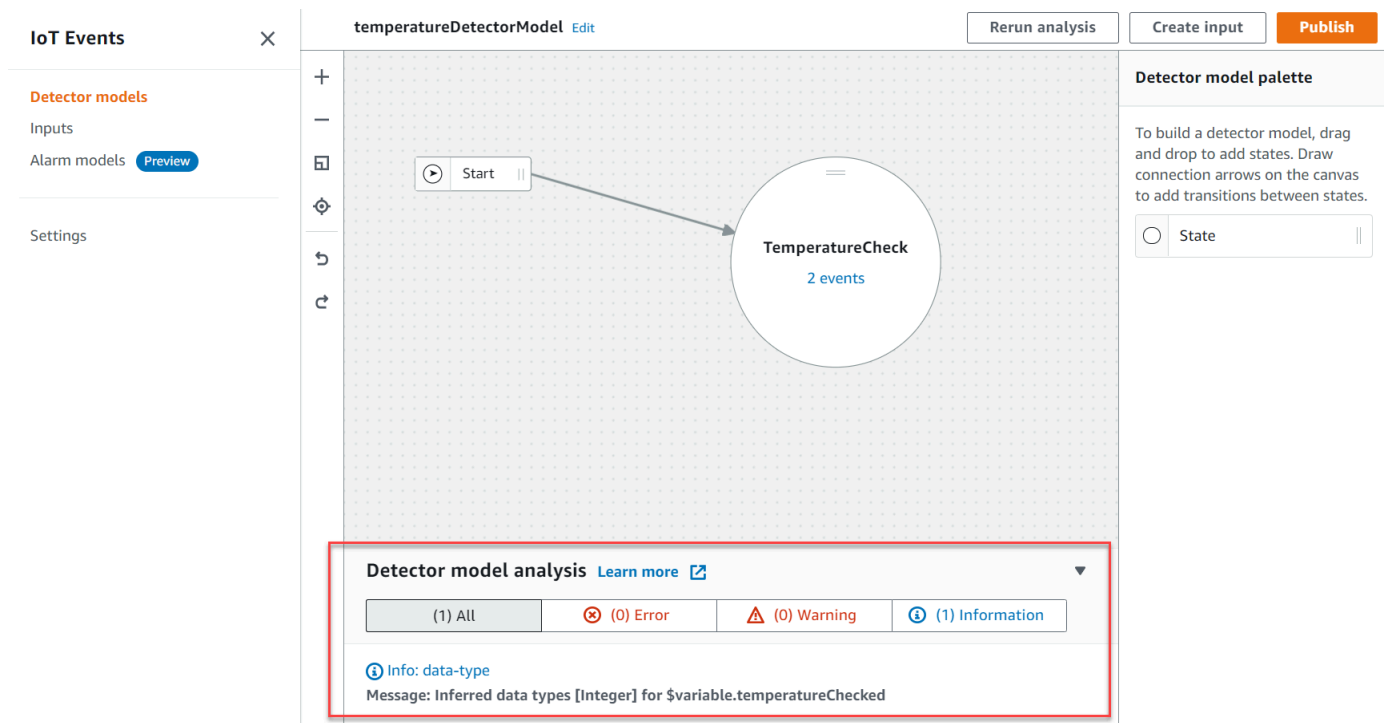
Analisis model detektor dapat membantu Anda mengoptimalkan model Anda, mengidentifikasi potensi masalah, dan memastikan mereka berfungsi sebagaimana dimaksud. Misalnya, pada ladang angin, analisis model detektor dapat mengungkapkan jika model mengidentifikasi dengan benar potensi kegagalan gigi berdasarkan pola getaran abnormal. Atau, jika model secara akurat memicu peringatan pemeliharaan ketika kecepatan angin melebihi ambang operasi yang aman. Dengan menyempurnakan model berdasarkan analisis, Anda dapat meningkatkan pemeliharaan prediktif, mengurangi waktu henti, dan meningkatkan efisiensi produksi energi secara keseluruhan.

Untuk menganalisis model detektor

1. Masuk ke [konsol AWS IoT Events](#) tersebut.
2. Di panel navigasi, pilih Model detektor.
3. Di bawah model Detektor, pilih model detektor target.
4. Pada halaman model detektor Anda, pilih Edit.
5. Di sudut kanan atas, pilih Jalankan analisis.



Berikut ini adalah contoh hasil analisis di AWS IoT Events konsol.



## Analisis model detektor di AWS IoT Events (AWS CLI)

Menganalisis model AWS IoT Events detektor Anda secara terprogram memberikan wawasan berharga tentang struktur, perilaku, dan kinerjanya. Pendekatan berbasis API ini memungkinkan analisis otomatis, integrasi dengan alur kerja Anda yang ada, dan kemampuan untuk melakukan operasi massal di beberapa model detektor. Dengan memanfaatkan [StartDetectorModelAnalysis](#) API, Anda dapat memulai pemeriksaan mendalam terhadap model Anda, membantu Anda


mengidentifikasi potensi masalah, mengoptimalkan alur logika, dan memastikan bahwa pemrosesan peristiwa IoT Anda selaras dengan kebutuhan bisnis Anda.

Langkah-langkah berikut menggunakan AWS CLI untuk menganalisis model detektor.

Untuk menganalisis model detektor menggunakan AWS CLI

1. Jalankan perintah berikut untuk memulai analisis.

```
aws iotevents start-detector-model-analysis --cli-input-json file://file-name.json
```

 Note

Ganti *file-name* dengan nama file yang berisi definisi model detektor.

### Example Definisi model detektor

```
{
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "TemperatureCheck",
        "onInput": {
          "events": [
            {
              "eventName": "Temperature Received",
              "condition":
"isNull($input.TemperatureInput.sensorData.temperature)==false",
              "actions": [
                {
                  "iotTopicPublish": {
                    "mqttTopic": "IoTEvents/Output"
                  }
                }
              ]
            }
          ]
        },
        "transitionEvents": []
      },
      {
        "onEnter": {
          "events": [
```

```

        {
            "eventName": "Init",
            "condition": "true",
            "actions": [
                {
                    "setVariable": {
                        "variableName": "temperatureChecked",
                        "value": "0"
                    }
                }
            ]
        },
        "onExit": {
            "events": []
        }
    ],
    "initialStateName": "TemperatureCheck"
}
}

```

Jika Anda menggunakan AWS CLI untuk menganalisis model detektor yang ada, pilih salah satu dari berikut ini untuk mengambil definisi model detektor:

- Jika Anda ingin menggunakan AWS IoT Events konsol, lakukan hal berikut:
  1. Di panel navigasi, pilih Model detektor.
  2. Di bawah model Detektor, pilih model detektor target.
  3. Pilih Ekspor model detektor dari Tindakan untuk mengunduh model detektor. Model detektor disimpan di JSON.
  4. Buka file JSON model detektor.
  5. Anda hanya membutuhkan `detectorModelDefinition` objek. Hapus yang berikut ini:
    - Braket keriting pertama (`{`) di bagian atas halaman
    - `detectorModelGaris`
    - `detectorModelConfigurationObjeknya`
    - Braket keriting terakhir (`}`) di bagian bawah halaman
  6. Simpan file tersebut.

- Jika Anda ingin menggunakan AWS CLI, lakukan hal berikut:

1. Jalankan perintah berikut di terminal.

```
aws iotevents describe-detector-model --detector-model-name detector-model-name
```

2. Ganti *detector-model-name* dengan nama model detektor Anda.
3. Salin `detectorModelDefinition` objek ke editor teks.
4. Tambahkan kurung keriting (`{}`) di luar `detectorModelDefinition`
5. Simpan file di JSON.

#### Example Contoh tanggapan

```
{  
  "analysisId": "c1133390-14e3-4204-9a66-31efd92a4fed"  
}
```

2. Salin ID analisis dari output.
3. Jalankan perintah berikut untuk mengambil status analisis.

```
aws iotevents describe-detector-model-analysis --analysis-id "analysis-id"
```

#### Note

Ganti *analysis-id* dengan ID analisis yang Anda salin.


#### Example Contoh tanggapan

```
{  
  "status": "COMPLETE"  
}
```

Nilai bisa jadi salah satu dari yang berikut:

- **RUNNING**— AWS IoT Events Menganalisis model detektor Anda. Proses ini bisa memakan waktu hingga satu menit untuk menyelesaikannya.
- **COMPLETE**— AWS IoT Events selesai menganalisis model detektor Anda.

- FAILED— AWS IoT Events tidak dapat menganalisis model detektor Anda. Coba lagi nanti.
4. Jalankan perintah berikut untuk mengambil satu atau lebih hasil analisis model detektor.


 Note

Ganti *analysis-id* dengan ID analisis yang Anda salin.

```
aws iotevents get-detector-model-analysis-results --analysis-id "analysis-id"
```

### Example Contoh tanggapan

```
{
  "analysisResults": [
    {
      "type": "data-type",
      "level": "INFO",
      "message": "Inferred data types [Integer] for
$variable.temperatureChecked",
      "locations": []
    },
    {
      "type": "referenced-resource",
      "level": "ERROR",
      "message": "Detector Model Definition contains reference to Input
'TemperatureInput' that does not exist.",
      "locations": [
        {
          "path": "states[0].onInput.events[0]"
        }
      ]
    }
  ]
}
```

 **Note**

Setelah AWS IoT Events mulai menganalisis model detektor Anda, Anda memiliki waktu hingga 24 jam untuk mengambil hasil analisis.

# AWS IoT Events perintah

Bab ini memberikan panduan komprehensif untuk semua operasi API yang tersedia di AWS IoT Events. Ini menawarkan penjelasan rinci, termasuk permintaan sampel, tanggapan, dan potensi kesalahan untuk setiap operasi di seluruh protokol layanan web yang didukung. Memahami operasi API ini membantu Anda mengintegrasikan secara efektif AWS IoT Events ke dalam aplikasi IoT Anda dan mengotomatiskan alur kerja deteksi dan respons peristiwa Anda.

## AWS IoT Events tindakan

Anda dapat menggunakan perintah AWS IoT Events API untuk membuat, membaca, memperbarui, dan menghapus input dan model detektor, dan untuk membuat daftar versinya. Untuk informasi selengkapnya, lihat [tindakan](#) dan [tipe data](#) yang didukung oleh AWS IoT Events dalam Referensi AWS IoT Events API.

[AWS IoT Events Bagian](#) dalam AWS CLI Command Reference mencakup AWS CLI perintah yang dapat Anda gunakan untuk mengelola dan memanipulasi AWS IoT Events.

## AWS IoT Events data

Anda dapat menggunakan perintah AWS IoT Events Data API untuk mengirim input ke detektor, detektor daftar, dan melihat atau memperbarui status detektor. Untuk informasi selengkapnya, lihat [tindakan](#) dan [tipe data](#) yang didukung oleh AWS IoT Events Data di Referensi AWS IoT Events API.

[Bagian AWS IoT Events data](#) dalam AWS CLI Command Reference mencakup AWS CLI perintah yang dapat Anda gunakan untuk memproses AWS IoT Events data.

## Riwayat dokumen untuk AWS IoT Events

Tabel berikut menjelaskan perubahan penting pada Panduan AWS IoT Events Pengembang setelah 17 September 2020. Untuk informasi lebih lanjut tentang pembaruan dokumentasi ini, Anda dapat berlangganan umpan RSS.

Perubahan	Deskripsi	Tanggal
<a href="#">Pemberitahuan akhir dukungan</a>	Pemberitahuan akhir dukungan: Pada 20 Mei 2026, AWS akan menghentikan dukungan untuk AWS IoT Events Setelah 20 Mei 2026, Anda tidak akan lagi dapat mengakses AWS IoT Events konsol atau AWS IoT Events sumber daya.	Mei 20, 2025
<a href="#">Peluncuran wilayah</a>	AWS IoT Events sekarang tersedia di wilayah Asia Pasifik (Mumbai).	30 September 2021
<a href="#">Peluncuran wilayah</a>	AWS IoT Events sekarang tersedia di Wilayah AWS GovCloud (AS-Barat).	22 September 2021
<a href="#">Memecahkan masalah model detektor dengan menjalankan analisis</a>	AWS IoT Events Sekarang dapat menganalisis model detektor Anda dan menghasilkan hasil analisis yang dapat Anda gunakan untuk memecahkan masalah model detektor Anda.	23 Februari 2021
<a href="#">Peluncuran wilayah</a>	Diluncurkan AWS IoT Events di Tiongkok (Beijing).	30 September 2020

<a href="#">Penggunaan ekspresi</a>	Menambahkan contoh untuk menunjukkan cara menulis ekspresi.	22 September 2020
<a href="#">Pemantauan dengan alarm</a>	Alarm membantu Anda memantau data untuk perubahan. Anda dapat membuat alarm yang mengirim notifikasi saat ambang batas dilanggar.	1 Juni 2020

## Pembaruan lebih awal

Tabel berikut menjelaskan perubahan penting pada Panduan AWS IoT Events Pengembang sebelum 18 September 2020.

Perubahan	Deskripsi	Tanggal
<a href="#">Menambahkan validasi tipe ke referensi Ekspresi</a>	Menambahkan informasi validasi tipe ke referensi Ekspresi.	3 Agustus 2020
<a href="#">Menambahkan peringatan Wilayah untuk layanan lain</a>	Menambahkan peringatan tentang memilih wilayah yang sama untuk AWS IoT Events dan AWS layanan lainnya.	7 Mei 2020
Penambahan, pembaruan	<ul style="list-style-type: none"> <li>Fitur Kustomisasi Payload</li> <li>Tindakan acara baru: Amazon DynamoDB dan AWS IoT SiteWise</li> </ul>	27 April 2020
Menambahkan fungsi bawaan untuk ekspresi bersyarat model detektor	Menambahkan fungsi bawaan untuk ekspresi kondisional model detektor.	10 September 2019

Perubahan	Deskripsi	Tanggal
<a href="#">Contoh model detektor yang ditambahkan</a>	Menambahkan contoh untuk model detektor.	5 Agustus 2019
Ditambahkan tindakan acara baru	Ditambahkan tindakan acara baru untuk: <ul style="list-style-type: none"> <li>• Lambda</li> <li>• Amazon SQS</li> <li>• Kinesis Data Firehose</li> <li>• AWS IoT Events masukan</li> </ul>	19 Juli 2019
Penambahan, koreksi	<ul style="list-style-type: none"> <li>• Deskripsi timeout() fungsi yang diperbarui.</li> <li>• Menambahkan praktik terbaik mengenai ketidakaktifan akun.</li> </ul>	11 Juni 2019
<a href="#">Kebijakan izin</a> yang diperbarui dan opsi debug konsol	<ul style="list-style-type: none"> <li>• Memperbarui kebijakan izin konsol.</li> <li>• Diperbarui gambar halaman opsi debug konsol.</li> </ul>	5 Juni 2019
Pembaruan	AWS IoT Events layanan terbuka untuk ketersediaan umum.	30 Mei 2019
Penambahan, pembaruan	<ul style="list-style-type: none"> <li>• Informasi keamanan yang diperbarui.</li> <li>• Menambahkan contoh model detektor beranotasi.</li> </ul>	22 Mei 2019
Menambahkan contoh dan izin yang diperlukan	Menambahkan contoh payload Amazon SNS; penambahan izin yang diperlukan untuk <code>CreateDetectorModel</code>	17 Mei 2019

Perubahan	Deskripsi	Tanggal
<a href="#">Menambahkan informasi keamanan tambahan</a>	Menambahkan informasi ke bagian keamanan.	9 Mei 2019
Rilis pratinjau terbatas	Rilis pratinjau dokumentasi terbatas.	28 Maret 2019