aws

Panduan Developer

AWS IoT Events

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

AWS IoT Events: Panduan Developer

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

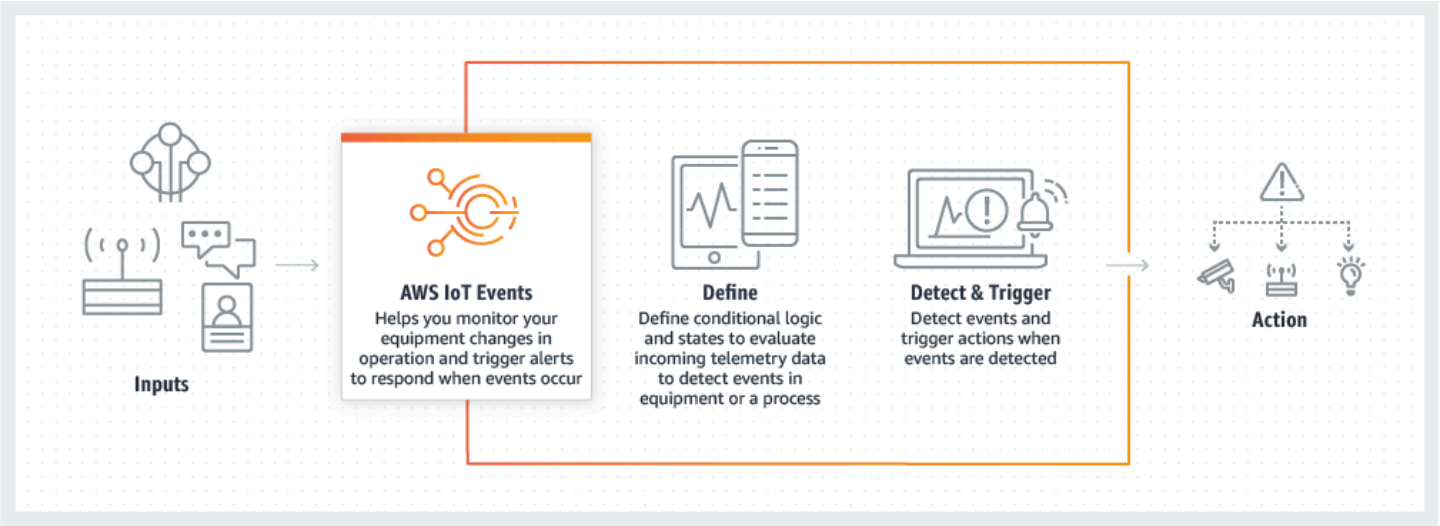
Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain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

Table of Contents

Apa itu AWS IoT 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 atauAPIs.



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 API antarmuka standar (`BatchPutMessage`API) atau konsol. AWS IoT Events

Untuk informasi lebih lanjut tentang memulai AWS IoT Events, lihatMemulai 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, lihatEkspresi 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 (AmazonSNS),, Lambda AWS IoT Core, Amazon, SQS 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 Amazon Connect, atau aplikasi perencanaan sumber daya perusahaan Anda sendiri. ERP

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, lihatTindakan yang didukung untuk menerima data dan memicu tindakan.

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, lihatAWS 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.

Melacak 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.

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 Akun AWS

Mendaftar untuk Akun AWS

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

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/pendaftaran.
2. Ikuti petunjuk online.

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

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua AWS layanan dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

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

Buat pengguna dengan akses administratif

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

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.
2. Aktifkan otentikasi multi-faktor (MFA) untuk pengguna root Anda.

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

Buat pengguna dengan akses administratif

1. Aktifkan Pusat IAM Identitas.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .
2. Di Pusat IAM Identitas, berikan akses administratif ke pengguna.

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

Masuk sebagai pengguna dengan akses administratif

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

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

Tetapkan akses ke pengguna tambahan

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

   Untuk petunjuk, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.
2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Menyiapkan izin untuk 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 IAM Pengguna memiliki informasi lebih rinci tentang mengendalikan izin untuk mengakses AWS sumber daya secara aman. Untuk informasi khusus AWS IoT Events, lihat Tindakan, sumber daya, dan kunci kondisi untuk AWS IoT Events.

Untuk menggunakan IAM konsol untuk membuat dan mengelola peran dan izin, lihat IAMtutorial: Mendelegasikan akses di seluruh AWS akun menggunakan IAM peran.

> **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* and *account-id* Referensi seperti yang diperlukan. Jika memungkinkan, Anda juga harus mengubah wildcard (*) untuk merujuk ke sumber daya tertentu yang akan diakses. Anda dapat menggunakan IAM konsol untuk memberikan izin AWS IoT Events untuk mengirim SNS peringatan Amazon 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 suatu MQTT topik.
- `iotEvents`untuk mengirim data 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 SQS antrian Amazon.

Example Kebijakan

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": "iot:Publish",
       "Resource": "arn:aws:iot:<region>:<account_id>:topic/*"
```

```
    },
    {
      "Effect": "Allow",
      "Action": "iotevents:BatchPutMessage",
      "Resource": "arn:aws:iotevents:<region>:<account_id>:input/*"
    },
    {
      "Effect": "Allow",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "dynamodb:PutItem",
      "Resource": "arn:aws:dynamodb:<region>:<account_id>:table/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ],
      "Resource": "arn:aws:firehose:<region>:<account_id>:deliverystream/*"
    },
    {
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:<region>:<account_id>:function:*"
    },
    {
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:<region>:<account_id>:*"
    },
    {
      "Effect": "Allow",
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:<region>:<account_id>:*"
    }
  ]
}
```

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`.

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

Anda dapat menentukan daftar masukan Amazon Resource Names (ARNs) 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

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

Kebijakan peran:

```
{
```

```
    "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:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [

          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Anda juga memerlukan kebijakan IAM izin 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 IAMPanduan Pengguna.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": [
          "iam:GetRole",
          "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::<account-id>: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 SNS perpesanan Amazon

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

Kebijakan peran:

```
{
```

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

Kebijakan kepercayaan:

```
{
  "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 SNS pesan Amazon 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 SNS pesan Amazon lain 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 BatchPutMessageoperasi.
- 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 CreateDatasetoperasi 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 SNS pesan Amazon 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 di`CreateDetectorModel`. 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 diTutorial untuk kasus AWS IoT Events penggunaan. Anda dapat menggunakan JSON contoh ini untuk membantu Anda mengikuti tutorial.

Topik

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 (AmazonSNS).

   Tutorial ini (dan contoh yang sesuai) berasumsi bahwa Anda membuat dua SNS topik Amazon. 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-nilai ini dengan SNS topik Amazon yang Anda buat. ARNs Untuk informasi lebih lanjut, lihat Panduan Developer Layanan Notifikasi Sederhana Amazon.

   Sebagai alternatif untuk menerbitkan peringatan ke SNS topik Amazon, Anda dapat meminta detektor mengirim MQTT pesan 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 tersebut. MQTT Anda juga dapat menentukan nama MQTT topik 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 Bekerja dengan AWS Management Console di Memulai dengan AWS Management Console.

Buat masukan untuk model

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 JSON input

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.

Membuat dan mengkonfigurasi input

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 JSON file, pilih `input.json` file yang Anda buat di ikhtisarBuat file JSON input. 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.

Buat masukan dalam Model Detektor

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

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 input, masukkan, Deskripsi opsional InputName, dan pilih Unggah file. Di kotak dialog yang ditampilkan, pilih `input.json` file yang Anda buat dalam ikhtisarBuat file JSON input.
6. Untuk Pilih atribut input, pilih atribut yang akan digunakan, dan pilih Buat. Dalam contoh ini, kita pilih motorIddan sensorData.pressure.

Buat model detektor

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 lingkar 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 SNS pesan.

    a. Untuk SNStopik, masukkan target ARN SNS topik Amazon 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 ARN menggunakan SNS topik Amazon 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 lingkar 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 diPembatasan 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, konfigurasikan 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 akan 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 (inJSON) membuat ulang atau memperbarui model detektor atau digunakan sebagai templat untuk membuat model detektor lain.

Anda dapat melakukan ini dari konsol atau dengan menggunakan CLI perintah 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

Ada beberapa cara untuk menerima data telemetri di AWS IoT Events (lihatTindakan yang didukung untuk menerima data dan memicu tindakan). 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 MQTT klien AWS IoT konsol untuk mengirim pesan pengujian. Anda dapat menggunakan metode ini untuk memasukkan data telemetri AWS IoT Events saat perangkat Anda dapat mengirim MQTT pesan menggunakan broker AWS IoT pesan.

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

        > **Note**
        >
        > Jangan gunakan spasi.

        - Deskripsi aturan. Ini bersifat opsional.
        - Pilih Berikutnya.
    2. Langkah 2. Konfigurasikan SQL pernyataan. Lengkapi bidang berikut:
        - SQLversi. Pilih opsi yang sesuai dari daftar.
        - SQLpernyataan. 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 berikut:

            - Nama masukan. Masukkan `PressureInput`.
            - Deskripsi. Ini bersifat opsional.

- Unggah JSON file. Unggah salinan JSON file 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 bersifat opsional.

Pilih Buat.

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

b. Mode Batch. Ini bersifat opsional. Jika payload adalah array pesan, pilih opsi ini.

c. ID pesan. Ini memang opsional, tetapi direkomendasikan.

d. IAMperan. 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 MQTT uji.
5. Pilih Publikasikan ke topik. Lengkapi bidang berikut:

- Nama topik. Masukkan nama untuk mengidentifikasi pesan, sepertimotors/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.

Instance detektor yang Anda buat menghasilkan dan mengirimi Anda SNS pesan Amazon. 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 SNS pesan Amazon 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 SNS pesan Amazon 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 step-by-step Contoh sederhana 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.
- Jika ada sesuatu yang tidak bekerja, lihatPemecahan 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

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) KeyValueuntuk 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

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

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 BatchPutMessageoperasi.
- Tentukan `iotEvents`saturan-tindakan untuk mesin aturan.AWS IoT Core Aturan-tindakan meneruskan data pesan dari masukan Anda ke. AWS IoT Events

- Di AWS IoT Analytics, gunakan CreateDatasetoperasi untuk membuat kumpulan data dengan`contentDeliveryRules`. Aturan ini menentukan AWS IoT Events input tempat konten kumpulan data dikirim secara otomatis.
- Tentukan iotEventstindakan dalam model AWS IoT Events detektor`onInput`, `onExit` atau `transitionEvents` peristiwa. 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 inputnya, dalam bentuk JSON muatan, 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 SNS pesan Amazon 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 di`CreateDetectorModel`.) 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 proses sebagai input ke AWS IoT Events detektor yang tidak mengharuskan Anda untuk 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 mengirimi 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 status kedua, jika acara `condition` terpenuhi.) Ketika Anda mencapai status kedua, Anda mengirim pesan segera setelah Anda memasuki negara bagian. (Anda menggunakan sebuah `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.

step-by-step Contoh sederhana

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 SNS (Amazon) 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 SNS pesan Amazon lain sebagai konfirmasi bahwa kondisi telah dihapus. Kami memerlukan tiga pembacaan berturut-turut di bawah ambang tekanan untuk menghilangkan kemungkinan kegagapan

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 BatchPutMessageoperasi. Metode ini mudah tetapi mengharuskan perangkat atau proses Anda dapat mengakses AWS IoT Events API melalui SDK atau 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 CreateDatasetoperasi 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 Anda ingin melacak masing-masing perangkat secara terpisah, tentukan bidang di setiap input yang mengidentifikasi perangkat tertentu atau memproses input berasal. Lihat `key` bidang di`CreateDetectorModel`. Ketika

perangkat baru diidentifikasi (nilai baru terlihat di bidang input yang diidentifikasi olehkey), instance detektor dibuat. Instans detektor baru terus merespons input yang berasal dari perangkat tertentu hingga model detektornya diperbarui atau dihapus. Anda memiliki detektor unik (instance) sebanyak nilai unik di bidang inputkey.

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 HVAC unit 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 input untuk menangkap data perangkat

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 contoh pesan sebagai JSON file dari perangkat atau proses Anda. Anda dapat menggunakannya untuk membuat input dari konsol atauCLI.

Buat model detektor untuk mewakili status perangkat

DiBuat 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 SNS pesan Amazon sebagai peringatan. Jika pembacaan tekanan

kembali normal (kurang dari 70) untuk tiga input berturut-turut, detektor kembali ke status "`Normal`" dan SNS mengirim pesan Amazon lain sebagai semua jelas.

Model detektor contoh ini mengasumsikan Anda telah membuat dua SNS topik Amazon yang Amazon Resource Names (ARNs) ditampilkan dalam definisi sebagai "`targetArn`": "`arn:aws:sns:us-east-1:123456789012:underPressureAction`" 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 CreateTopicoperasi di APIReferensi Layanan Pemberitahuan Sederhana Amazon.

Contoh ini juga mengasumsikan Anda telah membuat peran AWS Identity and Access Management (IAM) dengan izin yang sesuai. ARNPeran 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 peran di tempat yang sesuai dalam definisi model detektor. ARN

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": "pressureThresholdBreached",
                    "value": "0"
                  }
```

```
                }
              ]
            }
          ]
        },
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "Overpressurized",
              "condition": "$input.PressureInput.sensorData.pressure > 70",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "pressureThresholdBreached",
                    "value": "$variable.pressureThresholdBreached + 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

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

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 SNS titik akhir Amazon yang ARN ada`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 (misalnya) untuk motor "`Fulton-A32`" mengirim pesan ke SNS titik akhir Amazon "`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 SNS pesan 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":{
        "pressureThresholdBreached":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":{
        "pressureThresholdBreached":0
      },
      "timers":{}
    }
  },
  "eventName":"Normal Pressure Restored"
}
```

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

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

Pembatasan dan batasan model detektor

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

Cara menggunakan **`actions`** bidang

`actions`Bidang 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.pressureThresholdBreached"` dikurangi oleh satu `"Pressure Okay"` jika kondisi terpenuhi (ketika input saat ini memiliki tekanan kurang dari atau sama dengan 70).

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

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

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 masukan

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 bersyarat, Anda harus merujuk ke nilai kunci tersebut menggunakan backticks juga.

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

Contoh yang dikomentari: kontrol HVAC suhu

Beberapa JSON file contoh berikut memiliki komentar sebaris, yang membuatnya tidak validJSON. Versi lengkap dari contoh-contoh ini, tanpa komentar, tersedia diContoh: Menggunakan kontrol HVAC suhu.

Latar Belakang

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.

Definisi input untuk model detektor

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.

CLIperintah:

```
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.

CLIperintah:

```
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 detektor

`'areaDetectorModel'`Contohnya memiliki komentar sebaris.

CLIperintah:

```
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:coolOff"
                  }
                },
                {
                  "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:coolOn"
                  }
                },
                {
                  "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 && $variable.enteringNewState",
              // If the operational parameters specify that there should be a minimum
time that the
              //   heating and cooling units should be run before being shut off again,
we set
              //   a timer to ensure the proper operation here.
              "actions": [
                {
                  "setTimer": {
                    "timerName": "coolingTimer",
                    "seconds": 180
                  }
                },
                {
                  "setVariable": {
                    // We use this 'goodToGo' variable to store the status of the timer
expiration
                    //   for use in conditions that also use input variable values. If
                    //   'timeout()' is used in such mixed conditionals, its value is
lost.
                    "variableName": "goodToGo",
                    "value": "false"
                  }
                }
              ]
            },
            {
              "eventName": "dontDelay",
              "condition": "$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:coolOff"
                  }
                },
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
                  }
                },
                {
                  "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:coolOff"
                  }
                },
                {
                  "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

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.

CLIperintah:

```
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

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.

CLIperintah:

```
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.

CLIperintah:

```
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.

CLIperintah:

```
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": []
}
```

Pesan MQTT tertelan

Jika sumber daya komputasi sensor Anda tidak dapat menggunakan "BatchPutMessage"API, tetapi dapat mengirim datanya ke broker AWS IoT Core pesan menggunakan MQTT klien ringan, Anda dapat membuat aturan AWS IoT Core topik untuk mengarahkan data pesan ke AWS IoT Events input. Berikut ini adalah definisi aturan AWS IoT Events topik yang mengambil "areaId" dan "sensorId" memasukkan bidang dari MQTT topik, dan bidang dari "sensorData.temperature" bidang payload "temp" pesan, dan memasukkan data ini ke dalam kami. AWS IoT Events "temperatureInput"

CLIperintah:

```
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/anotheRole"
        }
      }
    ],
    "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 `"BatchPutMessage"` API panggilan berikut telah dilakukan.

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

Berkas: `spoofExample.json`

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

Hasilkan SNS pesan Amazon

Berikut ini adalah contoh SNS pesan 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"},"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"}
```

Konfigurasikan DescribeDetector API

Anda dapat menggunakan `DescribeDetector` operasi untuk melihat status saat ini, nilai variabel, dan timer untuk instance detektor.

CLIperintah:

```
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

Aturan berikut menerbitkan ulang AWS IoT Core MQTT pesan 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"`.

CLIperintah:

```
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]

CLIperintah:

```
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]

CLIperintah:

```
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]

CLIperintah:

```
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]

Tindakan yang didukung untuk menerima data dan memicu tindakan

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.

> **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:

- `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 suatu MQTT topik.
- `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 SQS antrian Amazon.

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 lebih lanjut, lihat SetTimerActiondi AWS IoT Events APIReferensi.

Atur ulang tindakan pengatur waktu

Reset timer action

`resetTimer`Tindakan ini memungkinkan Anda menyetel pengatur waktu 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 lebih lanjut, lihat ResetTimerActiondi AWS IoT Events APIReferensi.

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 lebih lanjut, lihat ClearTimerActiondi AWS IoT Events APIReferensi.

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 lebih lanjut, lihat SetVariableActiondi AWS IoT Events APIReferensi.

Bekerja dengan AWS layanan lain

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

- `iotTopicPublish`untuk mempublikasikan pesan tentang suatu MQTT topik.
- `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 SQS antrian Amazon.

> **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 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 AWS IoT Events APIReferensi.

AWS IoT Core

IoT topic publish action

AWS IoT Core Tindakan ini memungkinkan Anda mempublikasikan MQTT pesan melalui broker AWS IoT pesan. 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)

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

`mqttTopic`

MQTTTopik yang menerima pesan.

Anda dapat menentukan nama MQTT topik 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 Muatan di AWS IoT Events APIReferensi.

> **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 lebih lanjut, lihat IotTopicPublishActiondi AWS IoT Events APIReferensi.

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 Muatan di AWS IoT Events APIReferensi.

> **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 lebih lanjut, lihat IotEventsActiondi AWS IoT Events APIReferensi.

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 AWS IoT SiteWise Panduan Pengguna.

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 lebih lanjut, lihat AssetPropertydi AWS IoT SiteWise APIReferensi.

`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 lebih lanjut, lihat IotSiteWiseActiondi AWS IoT Events APIReferensi.

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 SQL Tanpa terkelola sepenuhnya yang memberikan kinerja yang cepat dan dapat diprediksi dengan skalabilitas yang mulus. Untuk informasi selengkapnya, 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`. Default-nya adalah `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`. Default-nya adalah `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 Muatan di AWS IoT Events APIReferensi.

Jika jenis payload yang ditentukan adalah string, `DynamoDBAction` mengirimkan non- JSON data ke tabel DynamoDB sebagai data biner. Konsol DynamoDB menampilkan data sebagai teks yang dikodekan Base64. 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 lebih lanjut, lihat D ynamoDBAction di AWS IoT Events APIReferensi.

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 SQL Tanpa terkelola sepenuhnya yang memberikan kinerja yang cepat dan dapat diprediksi dengan skalabilitas yang mulus. Untuk informasi selengkapnya, 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 Muatan di AWS IoT Events APIReferensi.

> **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 lebih lanjut, lihat D ynamoDBv 2Action di AWS IoT Events APIReferensi.

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 Muatan di AWS IoT Events APIReferensi.

> **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 lebih lanjut, lihat FirehoseActiondi AWS IoT Events APIReferensi.

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`

Fungsi Lambda untuk memanggil. ARN

`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 Muatan di AWS IoT Events APIReferensi.

> **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 lebih lanjut, lihat LambdaActiondi AWS IoT Events APIReferensi.

Amazon Simple Notification Service

SNS action

Tindakan mempublikasikan SNS topik Amazon memungkinkan Anda mempublikasikan SNS pesan Amazon. 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 selengkapnya, lihat Apa itu AmazonSNS? di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon.

> **Note**
>
> Tindakan publikasi SNS topik Amazon 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 SNS tindakan Amazon dimulai.

More information (2)

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

`targetArn`

SNSTarget Amazon yang menerima pesan. ARN

`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 Muatan di AWS IoT Events APIReferensi.

> **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 lebih lanjut, lihat SNSTopicPublishActiondi AWS IoT Events APIReferensi.

Amazon Simple Queue Service

SQS action

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

Amazon Simple Queue Service (AmazonSQS) 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**
>
> SQSTindakan Amazon 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 SQS tindakan Amazon dimulai.

More information (3)

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

`queueUrl`

SQSAntrian Amazon yang menerima data. URL

`useBase64`

(Opsional) AWS IoT Events mengkodekan data ke dalam teks Base64, jika Anda menentukan. `TRUE` Default-nya adalah `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 Muatan di AWS IoT Events APIReferensi.

> **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 lebih lanjut, lihat SNSTopicPublishActiondi AWS IoT Events APIReferensi.

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 Amazon Connect, 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

Anda dapat menggunakan template literal, operator, fungsi, referensi, dan substitusi dalam ekspresi. AWS IoT Events

Literal

- Bilangan Bulat
- Decimal
- String
- Boolean

Operator

Unary

- Tidak (Boolean): `!`
- Tidak (bitwise): `~`
- 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`
- AND: **&**

  Misalnya: `13 & 5` -> `5`
- XOR: **^**

  Misalnya: `13 ^ 5` -> `8`
- NOT: **~**

  Misalnya: `~13` -> `-14`

Boolean

- Kurang dari: **<**
- Kurang dari atau sama dengan: **<=**
- Sama dengan: **==**
- Tidak Sama Dengan: **!=**
- Lebih besar dari atau sama dengan: **>=**
- Lebih besar dari: **>**
- AND: **&&**

- 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 ekspresi

Fungsi Bawaan

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

Mengevaluasi `true` apakah timer yang ditentukan telah berlalu. Ganti"*`nama pengatur waktu`*"dengan nama timer yang Anda tentukan, dalam tanda kutip. Dalam tindakan peristiwa, Anda dapat menentukan pengatur waktu dan kemudian memulai pengatur waktu, mengatur ulang, atau menghapus pengatur waktu yang telah Anda tentukan sebelumnya. Lihat lapangan`detectorModelDefinition.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. Bagian *type* nilainya 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*")

Bagian *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 input 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 (ANDoperasi 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. `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 bitwise XOR dari ekspresi integer (XORoperasi 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 (NOToperasi 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.

Referensi untuk input dan variabel dalam ekspresi

Masukan

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

`input-name`adalah masukan yang Anda buat menggunakan CreateInputtindakan.

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 CreateDetectorModeltindakan.

Misalnya, jika Anda memiliki variabel bernama `TechnicianID` yang Anda definisikan menggunakan`detectorDefinition.states.onInputEvents.actions.setVariable.variabl` 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`* nilai, Anda harus referensi string dengan tanda kutip ganda.

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

> **Note**
>
> Untuk `convert(type, expression)` fungsi tersebut, 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
Integer, string
Bilangan bulat, Boolean
Desimal, string
Desimal, Boolean
Tali, Boolean

Templat substitusi untuk ekspresi

```
'${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 SetVariableActiontindakan untuk mendefinisikan variabel. `variableName` adalah `SensorID`, dan `value` adalah `10`. Anda dapat membuat template substitusi berikut.

Templat substitusi	String hasil
`'${'Sensor ' + $variable.SensorID}'`	`"Sensor 10"`

Templat substitusi	String hasil
`'Sensor ' + '${$variable.SensorID + 1}'`	`"Sensor 11"`
`'Sensor 10: ${$variable.SensorID == 10}'`	`"Sensor 10: true"`
`'{\"sensor\":\"${$variable.SensorID + 1}\"}'`	`"{\"sensor\":\"11\"}"`
`'{\"sensor\":${$variable.SensorID + 1}}'`	`"{\"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 HVAC suhu

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.
- Mengevaluasi 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.
- Ambil tindakan yang mengirim Amazon SNS dan MQTT pesan.

Definisi input untuk suatu HVAC sistem

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

CLIperintah 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.

CLIperintah 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 suatu HVAC sistem

`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.

CLIperintah 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:coolOff"
                }
              },
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
                }
              },
              {
                "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:coolOff"
                }
              },
              {
                "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"
    }
}
```

BatchPutMessageContoh untuk sebuah HVAC sistem

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

CLIperintah 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.

CLIperintah 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.

CLIperintah 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 SNS pesan Amazon 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"
    }
}
```

BatchUpdateDetectorContoh untuk sebuah HVAC sistem

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

CLIperintah 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:

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

AWS IoT Core aturan mesin

Aturan berikut menerbitkan ulang AWS IoT Events MQTT pesan 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`.

CLIperintah 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]

CLIperintah 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]

CLIperintah 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]

CLIperintah 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: Kondisi pendeteksi derek

Cerita latar belakang

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 SNS topik Amazon 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 SNS target.

Kirim perintah sebagai respons terhadap kondisi yang terdeteksi

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 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 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 memperbarui 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": "craneThresholdBreached",
                                        "value": "0"
                                    }
                                }
                            ]
                        }
                    ]
                },
                "onInput": {
                    "events": [
                        {
```

```
                            "eventName": "Overheated",
                            "condition": "$input.TemperatureInput.temperature > 35",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "craneThresholdBreached",
                                        "value": "$variable.craneThresholdBreached + 1"
                                    }
                                }
                            ]
                        },
                        {
                            "eventName": "Crane Threshold Breached",
                            "condition": "$variable.craneThresholdBreached > 5",
                            "actions": [
                                {
                                    "sns": {
                                        "targetArn": "arn:aws:sns:us-
east-1:123456789012:CraneSNSTopic"
                                    }
                                }
                            ]
                        },
                        {
                            "eventName": "Underheated",
                            "condition": "$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": "$input.TemperatureInput.temperature > 30",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "craneThresholdBreached",
                                        "value": "$variable.craneThresholdBreached + 1"
                                    }
```

```
                                }
                            ]
                        },
                        {
                            "eventName": "Crane Threshold Breached",
                            "condition": "$variable.craneThresholdBreached > 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": "craneThresholdBreached",
                                        "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": "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"
    },
    "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"
}
```

Input untuk pemantauan derek

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

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 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.

```
{
    "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

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": "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: ISA Alarm

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: Bangun alarm sederhana

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

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

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 mengkonfigurasi 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 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 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 IAM peran 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 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 JSON jalur 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, konfigurasikan 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 SMS pemberitahuan dan pemberitahuan email.

- Email - Alarm mengirimkan pemberitahuan email.
- Teks - Alarm mengirimkan SMS pemberitahuan.

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 IAM Identitas Anda sebelum Anda dapat menambahkannya ke model alarm Anda. Untuk informasi selengkapnya, lihat Mengelola akses Pusat IAM Identitas penerima alarm.

d. (Opsional) Untuk pesan kustom tambahan, masukkan pesan yang menjelaskan apa yang terdeteksi 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 IAM peran untuk Anda.
- Anda dapat Menggunakan IAM peran yang ada yang memungkinkan model alarm ini untuk 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, lihatBekerja 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 AWS IoT Events sumber daya Anda.
11. Pilih Buat.

Menanggapi alarm

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, lihatAkui aliran.

API

Untuk menanggapi satu atau lebih alarm, Anda dapat menggunakan AWS IoT Events API operasi berikut:

- BatchAcknowledgeAlarm
- BatchDisableAlarm
- BatchEnableAlarm
- BatchResetAlarm
- BatchSnoozeAlarm

Mengelola pemberitahuan alarm

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
- Menggunakan fungsi Lambda yang disediakan oleh AWS IoT Events
- Mengelola akses Pusat IAM Identitas penerima alarm

Membuat fungsi Lambda

AWS IoT Events menyediakan fungsi Lambda yang memungkinkan alarm untuk mengirim dan menerima email dan pemberitahuan. SMS

Persyaratan

Persyaratan berikut berlaku saat Anda membuat fungsi Lambda untuk alarm:

- Jika alarm Anda mengirim email atau SMS pemberitahuan, Anda harus memiliki IAM peran yang memungkinkan AWS Lambda untuk bekerja dengan Amazon SES dan AmazonSNS.

  Contoh kebijakan:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ses:GetIdentityVerificationAttributes",
                "ses:SendEmail",
                "ses:VerifyEmailIdentity"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish",
                "sns:OptInPhoneNumber",
                "sns:CheckIfPhoneNumberIsOptedOut"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:*:*:*"
        }
    ]
```

```
}
```

- 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

Deploying fungsi Lambda

Tutorial ini menggunakan AWS CloudFormation template untuk menyebarkan fungsi Lambda. Template ini secara otomatis membuat IAM peran yang memungkinkan fungsi Lambda bekerja dengan Amazon dan SES 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 AWS CLI dalam Panduan Pengguna AWS Command Line Interface .
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 Menginstal atau memperbarui ke versi terbaru dari Panduan AWS Command Line Interface Pengguna AWS CLI
3. Unduh CloudFormation template, notificationLambda.template.yaml.zip.

   > **Note**
   >
   > Jika Anda mengalami kesulitan mengunduh file, template juga tersedia di fileCloudFormation 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 antrean 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

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 SSO. 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.utcfromtimestamp(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"
              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 SSO.
 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.utcfromtimestamp(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')
```

Menggunakan fungsi Lambda yang disediakan oleh AWS IoT Events

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 (AmazonSES). Untuk informasi selengkapnya, lihat Memverifikasi alamat email di Amazon SES, 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 SMS notifikasi, 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.

  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 SMS pemberitahuan, Anda dapat menghubungi operator telepon. Operator dapat memblokir pesan.

Mengelola akses Pusat IAM Identitas penerima alarm

AWS IoT Events digunakan AWS IAM Identity Center untuk mengelola SSO akses penerima alarm. Untuk mengaktifkan alarm untuk mengirim pemberitahuan ke penerima, Anda harus mengaktifkan Pusat IAM Identitas dan menambahkan penerima ke toko Pusat IAM Identitas Anda. Untuk informasi selengkapnya, lihat Menambahkan AWS IAM Identity Center Pengguna di Panduan Pengguna.

Important

- Anda harus memilih AWS Wilayah yang sama untuk AWS IoT Events, AWS Lambda, dan Pusat IAM Identitas.
- AWS Organizations hanya mendukung satu Wilayah Pusat IAM Identitas pada satu waktu. Jika Anda ingin membuat Pusat IAM Identitas tersedia di Wilayah yang berbeda, Anda

harus terlebih dahulu menghapus konfigurasi Pusat IAM Identitas Anda saat ini. Untuk informasi selengkapnya, lihat Data Wilayah Pusat IAM Identitas 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 menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. 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

AWS Identity and Access Management (IAM) adalah AWS layanan yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. IAMadministrator mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya. AWS IoT Events IAMadalah AWS layanan yang dapat Anda gunakan tanpa biaya tambahan.

Topik

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan AWS IoT Events.

Pengguna layanan — Jika Anda menggunakan AWS IoT Events layanan untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak AWS IoT Events fitur untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara mengelola akses dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di AWS IoT Events, lihat Memecahkan masalah AWS IoT Events identitas dan akses.

Administrator layanan — Jika Anda bertanggung jawab atas AWS IoT Events sumber daya di perusahaan Anda, Anda mungkin memiliki akses penuh ke AWS IoT Events. Tugas Anda adalah menentukan AWS IoT Events fitur dan sumber daya mana yang harus diakses pengguna layanan Anda. Anda kemudian harus mengirimkan permintaan ke IAM administrator Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasarIAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakannya IAM AWS IoT Events, lihatBagaimana AWS IoT Events bekerja dengan IAM.

IAMadministrator - Jika Anda seorang IAM administrator, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses AWS IoT Events. Untuk melihat contoh kebijakan AWS IoT Events berbasis identitas yang dapat Anda gunakan, lihat. IAM AWS IoT Events contoh kebijakan berbasis identitas

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai IAM pengguna, atau dengan mengambil peranIAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensi yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (Pusat IAM Identitas), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas federasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan IAM peran. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensil Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang menggunakan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani AWS API permintaan di Panduan IAM Pengguna.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari lebih lanjut, lihat Autentikasi multi-faktor di Panduan AWS IAM Identity Center Pengguna dan Menggunakan otentikasi multi-faktor (MFA) AWS di Panduan Pengguna. IAM

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua AWS layanan dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan

untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensi pengguna root di IAMPanduan Pengguna.

Pengguna dan grup IAM

IAMPengguna adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya mengandalkan kredensi sementara daripada membuat IAM pengguna yang memiliki kredensi jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensi jangka panjang dengan IAM pengguna, kami sarankan Anda memutar kunci akses. Untuk informasi selengkapnya, lihat Memutar kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensi jangka panjang di IAMPanduan Pengguna.

IAMGrup adalah identitas yang menentukan kumpulan IAM pengguna. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup bernama IAMAdminsdan memberikan izin grup tersebut untuk mengelola sumber dayaIAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari lebih lanjut, lihat Kapan membuat IAM pengguna (bukan peran) di Panduan IAM Pengguna.

IAMperan

IAMPeran adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Ini mirip dengan IAM pengguna, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil IAM peran sementara AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan memanggil AWS CLI atau AWS API operasi atau dengan menggunakan kustomURL. Untuk informasi selengkapnya tentang metode penggunaan peran, lihat Menggunakan IAM peran di Panduan IAM Pengguna.

IAMperan dengan kredensi sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga di Panduan IAM Pengguna. Jika Anda menggunakan Pusat IAM Identitas, Anda mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah diautentikasi, Pusat IAM Identitas menghubungkan izin yang disetel ke peran. IAM Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin IAM pengguna sementara — IAM Pengguna atau peran dapat mengambil IAM peran untuk sementara mengambil izin yang berbeda untuk tugas tertentu.
- Akses lintas akun — Anda dapat menggunakan IAM peran untuk memungkinkan seseorang (prinsipal tepercaya) di akun lain mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa AWS layanan, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM Panduan Pengguna. IAM
- Akses lintas layanan — Beberapa AWS layanan menggunakan fitur lain AWS layanan. Misalnya, saat Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses teruskan (FAS) — Saat Anda menggunakan IAM pengguna atau peran untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FASmenggunakan izin dari pemanggilan utama AWS layanan, dikombinasikan dengan permintaan AWS layanan untuk membuat permintaan ke layanan hilir. FASPermintaan hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain AWS layanan atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan saat membuat FAS permintaan, lihat Meneruskan sesi akses.
  - Peran layanan — Peran layanan adalah IAMperan yang diasumsikan layanan untuk melakukan tindakan atas nama Anda. IAMAdministrator dapat membuat, memodifikasi, dan menghapus peran layanan dari dalamIAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke AWS layanan dalam IAMPanduan Pengguna.

  - Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. AWS layanan Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. IAMAdministrator dapat melihat, tetapi tidak mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan IAM peran untuk mengelola kredensil sementara untuk aplikasi yang berjalan pada EC2 instance dan membuat AWS CLI atau AWS API meminta. Ini lebih baik untuk menyimpan kunci akses dalam EC2 instance. Untuk menetapkan AWS peran ke EC2 instance dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensi sementara. Untuk informasi selengkapnya, lihat Menggunakan IAM peran untuk memberikan izin ke aplikasi yang berjalan di EC2 instans Amazon di IAMPanduan Pengguna.

Untuk mempelajari apakah akan menggunakan IAM peran atau IAM pengguna, lihat Kapan membuat IAM peran (bukan pengguna) di Panduan IAM Pengguna.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai JSON dokumen. Untuk informasi selengkapnya tentang struktur dan isi dokumen JSON kebijakan, lihat Ringkasan JSON kebijakan di Panduan IAM Pengguna.

Administrator dapat menggunakan AWS JSON kebijakan untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka butuhkan, IAM administrator dapat membuat IAM kebijakan. Administrator kemudian dapat menambahkan IAM kebijakan ke peran, dan pengguna dapat mengambil peran.

IAMkebijakan menentukan izin untuk tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasi. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan

`iam:GetRole`. Pengguna dengan kebijakan itu bisa mendapatkan informasi peran dari AWS Management Console, AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan JSON izin yang dapat dilampirkan ke identitas, seperti pengguna, grup IAM pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat IAM kebijakan di Panduan Pengguna. IAM

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan sebaris, lihat Memilih antara kebijakan terkelola dan kebijakan sebaris di IAMPanduan Pengguna.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batas izin — Batas izin adalah fitur lanjutan tempat Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas (pengguna atau peran). IAM IAM Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batas izin, lihat Batas izin untuk IAM entitas di IAMPanduan Pengguna.
- Kebijakan kontrol layanan (SCPs) — SCPs adalah JSON kebijakan yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam suatu organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCPMembatasi izin untuk entitas di akun anggota, termasuk masing-masing Pengguna root akun AWS. Untuk informasi selengkapnya tentang Organizations danSCPs, lihat Kebijakan kontrol layanan di Panduan AWS Organizations Pengguna.

- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan secara tegas dalam salah satu kebijakan ini membatalkan izin. Untuk informasi selengkapnya, lihat Kebijakan sesi di Panduan IAM Pengguna.

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 IAM Pengguna.

Pelajari selengkapnya

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 IAM fitur apa yang tersedia untuk digunakan AWS IoT Events. Untuk mendapatkan tampilan tingkat tinggi tentang cara AWS IoT Events dan AWS layanan lain bekerja denganIAM, lihat AWS layanan yang bekerja dengan IAM dalam Panduan IAM Pengguna.

Topik

- AWS IoT Events kebijakan berbasis identitas
- AWS IoT Events Kebijakan berbasis sumber daya
- Otorisasi berdasarkan tanda AWS IoT Events
- AWS IoT Events IAMperan

AWS IoT Events kebijakan berbasis identitas

Dengan kebijakan IAM berbasis identitas, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak serta kondisi di mana tindakan diizinkan atau ditolak. AWS IoT Events mendukung tindakan, sumber daya, dan kunci kondisi tertentu. Untuk mempelajari semua elemen yang Anda gunakan dalam JSON kebijakan, lihat referensi elemen IAM JSON kebijakan di Panduan IAM Pengguna.

Tindakan

`Action`Elemen kebijakan IAM berbasis identitas menggambarkan tindakan atau tindakan spesifik yang akan diizinkan atau ditolak oleh kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan AWS API operasi 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 AWS IoT Events `CreateInput` API operasi, Anda menyertakan `iotevents:CreateInput` tindakan tersebut dalam kebijakan mereka. Untuk memberikan izin kepada seseorang untuk mengirim masukan dengan AWS IoT Events `BatchPutMessage` API operasi, 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 AWS IoT Events tindakan, lihat Tindakan yang Ditentukan oleh AWS IoT Events di Panduan IAM Pengguna.

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 tersebut berlaku untuk semua sumber daya.

Sumber daya model AWS IoT Events detektor memiliki yang berikutARN:

```
arn:${Partition}:iotevents:${Region}:${Account}:detectorModel/${detectorModelName}
```

Untuk informasi selengkapnya tentang formatARNs, lihat Mengidentifikasi AWS sumber daya dengan Nama Sumber Daya Amazon (ARNs).

Misalnya, untuk menentukan model `Foobar` detektor dalam pernyataan Anda, gunakan yang berikut iniARN:

```
"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 AWS IoT Events API tindakan 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 AWS IoT Events sumber daya dan jenis sumber dayaARNs, lihat Sumber Daya yang Ditentukan oleh AWS IoT Events di Panduan IAM Pengguna. Untuk mempelajari tindakan

mana yang dapat Anda tentukan ARN dari 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 selengkapnya, lihat elemen IAM kebijakan: Variabel dan tag di Panduan IAM Pengguna.

AWS IoT Events tidak menyediakan kunci kondisi khusus layanan apa pun, tetapi mendukung penggunaan beberapa kunci kondisi global. Untuk melihat semua kunci kondisi AWS AWS global, lihat kunci konteks kondisi global di Panduan IAM Pengguna.

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 tanda 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 tag, berikan informasi tentang tag 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 tanda pada sumber daya tersebut, lihat Melihat AWS IoT Events input berdasarkan tag.

AWS IoT Events IAMperan

IAMPeran adalah entitas di dalam Anda Akun AWS yang memiliki izin khusus.

Menggunakan kredensi sementara dengan AWS IoT Events

Anda dapat menggunakan kredensi sementara untuk masuk dengan federasi, mengambil IAM peran, atau untuk mengambil peran lintas akun. Anda memperoleh kredensi keamanan sementara dengan memanggil AWS Security Token Service (AWS STS) API operasi seperti AssumeRoleatau. GetFederationToken

AWS IoT Events tidak mendukung penggunaan kredensi 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 IAM akun Anda dan dimiliki oleh layanan. IAMAdministrator dapat melihat tetapi tidak 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 IAM akun Anda dan dimiliki oleh akun. Ini berarti bahwa IAM administrator 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 AWS Management

Console, AWS CLI, atau AWS API. IAMAdministrator harus membuat IAM kebijakan yang memberikan izin kepada pengguna dan peran untuk melakukan API operasi tertentu pada sumber daya tertentu yang mereka butuhkan. Administrator kemudian harus melampirkan kebijakan tersebut ke pengguna atau grup yang memerlukan izin tersebut.

Untuk mempelajari cara membuat kebijakan IAM berbasis identitas menggunakan contoh dokumen JSON kebijakan ini, lihat Membuat kebijakan pada JSON tab di Panduan Pengguna. IAM

Topik

- Praktik terbaik kebijakan
- Menggunakan konsol AWS IoT Events
- Memungkinkan pengguna untuk melihat izin mereka sendiri
- Mengakses satu masukan AWS IoT Events
- Melihat 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 IAM Pengguna.
- 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 Berikan hak istimewa paling sedikit di Panduan IAM Pengguna.
- Aktifkan MFA untuk Operasi Sensitif — Untuk keamanan ekstra, pengguna harus menggunakan otentikasi multi-faktor (MFA) untuk mengakses sumber daya atau API operasi sensitif. Untuk informasi selengkapnya, lihat Menggunakan otentikasi multi-faktor (MFA) AWS di IAM Panduan Pengguna.

- 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 kondisi untuk mengizinkan permintaan hanya dalam tanggal atau rentang waktu tertentu, atau untuk meminta penggunaan SSL atauMFA. Untuk informasi selengkapnya, lihat elemen IAM JSON kebijakan: Kondisi dalam Panduan IAM Pengguna.

Menggunakan konsol AWS IoT Events

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 IAM Pengguna:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotevents-data:BatchPutMessage",
                "iotevents-data:BatchUpdateDetector",
                "iotevents:CreateDetectorModel",
                "iotevents:CreateInput",
                "iotevents:DeleteDetectorModel",
                "iotevents:DeleteInput",
                "iotevents-data:DescribeDetector",
                "iotevents:DescribeDetectorModel",
                "iotevents:DescribeInput",
                "iotevents:DescribeLoggingOptions",
                "iotevents:ListDetectorModelVersions",
                "iotevents:ListDetectorModels",
                "iotevents-data:ListDetectors",
                "iotevents:ListInputs",
```

```
                "iotevents:ListTagsForResource",
                "iotevents:PutLoggingOptions",
                "iotevents:TagResource",
                "iotevents:UntagResource",
                "iotevents:UpdateDetectorModel",
                "iotevents:UpdateInput",
                "iotevents:UpdateInputRouting"
            ],
            "Resource": "arn:${Partition}:iotevents:${Region}:${Account}:detectorModel/
${detectorModelName}",
            "Resource": "arn:${Partition}:iotevents:${Region}:${Account}:input/
${inputName}"
        }
    ]
}
```

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau. AWS API Sebagai gantinya, izinkan akses hanya ke tindakan yang cocok dengan API operasi yang Anda coba lakukan.

Memungkinkan pengguna untuk melihat izin mereka sendiri

Contoh ini menunjukkan cara Anda membuat kebijakan yang memungkinkan pengguna melihat kebijakan sebaris dan terkelola yang dilampirkan pada identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau secara terprogram menggunakan atau. AWS CLI AWS API

```
{
       "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": "*"
        }
    ]
}
```

Mengakses satu masukan AWS IoT Events

Dalam contoh ini, Anda ingin memberikan pengguna Akun AWS akses ke salah satu AWS IoT Events input Anda,`exampleInput`. Anda juga ingin mengizinkan pengguna untuk menambah, memperbarui, dan menghapus input.

Kebijakan memberikan`iotevents:ListInputs`,,, `iotevents:DescribeInput` `iotevents:CreateInputiotevents:DeleteInput`, dan `iotevents:UpdateInput` izin kepada pengguna. Untuk contoh panduan untuk Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) yang memberikan izin kepada pengguna dan mengujinya menggunakan konsol, lihat Mengontrol akses ke bucket dengan kebijakan pengguna.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ListInputsInConsole",
         "Effect":"Allow",
         "Action":[
            "iotevents:ListInputs"
         ],
         "Resource":"arn:aws:iotevents:::*"
```

```
      },
      {
         "Sid":"ViewSpecificInputInfo",
         "Effect":"Allow",
         "Action":[
            "iotevents:DescribeInput"
         ],
         "Resource":"arn:aws:iotevents:::exampleInput"
      },
      {
         "Sid":"ManageInputs",
         "Effect":"Allow",
         "Action":[
            "iotevents:CreateInput",
            "iotevents:DeleteInput",
            "iotevents:DescribeInput",
            "iotevents:ListInputs",
            "iotevents:UpdateInput"
         ],
         "Resource":"arn:aws:iotevents:::exampleInput/*"
      }
   ]
}
```

Melihat AWS IoT Events input berdasarkan tag

Anda dapat menggunakan syarat dalam kebijakan berbasis identitas Anda untuk mengontrol akses ke sumber daya AWS IoT Events berdasarkan tanda. Contoh ini menunjukkan bagaimana Anda dapat membuat kebijakan yang memungkinkan melihat *`input`*. Namun, izin hanya diberikan jika *`input`* tag `Owner` memiliki nilai nama pengguna tersebut. Kebijakan ini juga memberi izin yang diperlukan untuk menyelesaikan tindakan ini pada konsol tersebut.

```
{
    "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 IAM JSON kebijakan: Kondisi dalam Panduan IAM Pengguna.

Pencegahan confused deputy lintas layanan

> **Note**
>
> - AWS IoT Events Layanan ini hanya memungkinkan pelanggan untuk menggunakan peran untuk memulai tindakan di akun yang sama dengan sumber daya yang dibuat. Ini berarti bahwa serangan wakil yang membingungkan tidak dapat dilakukan dengan layanan ini.
> - Halaman ini berfungsi sebagai referensi bagi pelanggan untuk melihat bagaimana masalah wakil yang membingungkan bekerja dan dapat dicegah jika sumber daya lintas akun diizinkan dalam AWS IoT Events layanan.

Masalah deputi yang bingung adalah masalah keamanan di mana entitas yang tidak memiliki izin untuk melakukan tindakan dapat memaksa entitas yang lebih istimewa untuk melakukan tindakan. Pada tahun AWS, peniruan lintas layanan dapat mengakibatkan masalah wakil yang membingungkan.

Peniruan identitas lintas layanan dapat terjadi ketika satu layanan (layanan yang dipanggil) memanggil layanan lain (layanan yang dipanggil). Layanan pemanggilan dapat dimanipulasi menggunakan izinnya untuk bertindak pada sumber daya pelanggan lain dengan cara yang seharusnya tidak dilakukannya kecuali bila memiliki izin untuk mengakses. Untuk mencegah hal

ini, AWS sediakan alat yang membantu Anda melindungi data Anda untuk semua layanan dengan prinsip layanan yang telah diberikan akses ke sumber daya di akun Anda.

Sebaiknya gunakan kunci konteks kondisi `aws:SourceAccount`global `aws:SourceArn`dan dalam kebijakan sumber daya untuk membatasi izin yang AWS IoT Events memberikan layanan lain ke sumber daya. Jika `aws:SourceArn` nilainya tidak berisi ID akun, seperti bucket Amazon S3ARN, Anda harus menggunakan kedua kunci konteks kondisi global 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 hanya ingin satu sumber daya 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 wakil yang membingungkan adalah dengan menggunakan kunci konteks kondisi `aws:SourceArn` global dengan penuh ARN sumber daya. Jika Anda tidak mengetahui sumber daya yang lengkap ARN atau jika Anda menentukan beberapa sumber daya, gunakan kunci kondisi konteks `aws:SourceArn` global dengan wildcard (`*`) untuk bagian yang tidak diketahui dari file. ARN 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: Mengakses Model Detektor
- Contoh: Mengakses Model Alarm
- Contoh: Mengakses Sumber Daya di Wilayah Tertentu
- Contoh: Opsi Logging

Contoh: Mengakses Model Detektor

Dalam contoh ini, peran yang diberikan hanya dapat digunakan untuk mengakses model detektor bernama`foo`.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account_id"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:region:account_id:detectorModel/foo"
        }
      }
    }
  ]
 }
}
```

Contoh: Mengakses Model Alarm

Contoh berikut menunjukkan cara memberikan AWS IoT Events akses untuk mengirim data ke alarm sambil membatasi izin untuk mengubah konfigurasi alarm IAM menggunakan kebijakan.

Peran berikut hanya dapat digunakan untuk mengakses Model Alarm apa pun.

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

```
          "aws:SourceAccount": "account_id"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:region:account_id:alarmModel/*"
        }
      }
    }
  ]
}
```

Contoh: Mengakses Sumber Daya di Wilayah Tertentu

Contoh berikut menunjukkan peran yang dapat Anda gunakan untuk mengakses sumber daya di wilayah tertentu. Wilayah dalam contoh ini adalah *us-east-1*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account_id"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:us-east-1:account_id:*"
        }
      }
    }
  ]
}
```

Contoh: Opsi Logging

Untuk menyediakan peran untuk opsi logging, izinkan untuk diasumsikan oleh setiap AWS IoT Events sumber daya. Gunakan wildcard (*) untuk jenis sumber daya dan nama.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account_id"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:region:account_id:*"
        }
      }
    }
  ]
}
```

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 danIAM.

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 AWS Management Console 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 `mateojackson` IAM pengguna mencoba menggunakan konsol untuk melihat detail tentang *`input`* tetapi tidak memiliki `iotevents:`*`ListInputs`* izin kerja.

```
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 AWS layanan 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 IAM pengguna bernama `marymajor` mencoba menggunakan konsol 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 ini, lihatBagaimana 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 IAM pengguna lain Akun AWS yang Anda miliki di Panduan IAM Pengguna.
- 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 IAM Pengguna.
- Untuk mempelajari cara menyediakan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna yang diautentikasi secara eksternal (federasi identitas) di 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 Panduan Pengguna. IAM

Pemantauan AWS IoT Events

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 tujuan pemantauan Anda?
- Sumber daya manakah yang akan Anda pantau?
- Seberapa seringkah Anda akan memantau sumber daya ini?
- Apa sajakah alat pemantauan yang akan Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Langkah 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 AmazonEC2, Anda dapat memantau CPU pemanfaatan, disk I/O, dan penggunaan jaringan untuk instans Anda. Ketika kinerja berada di luar garis dasar yang ditetapkan, Anda mungkin perlu mengkonfigurasi ulang atau mengoptimalkan instance untuk mengurangi CPU pemanfaatan, meningkatkan I/O disk, atau mengurangi lalu lintas jaringan.

Topik

Alat pemantauan

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 mengautomasi 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:
  - Membuat dasbor yang disesuaikan untuk memantau layanan yang penting bagi Anda
  - 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 CloudWatch dengan Amazon

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 detektormemiliki 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 CloudWatch DEBUG tingkat 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 AWS IoT Events API panggilan 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 API panggilan untuk AWS IoT Events sebagai acara, 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 selengkapnya, lihat:

- Membuat jejak untuk AWS akun Anda

- CloudTrail layanan dan integrasi yang didukung
- Mengonfigurasi SNS notifikasi Amazon 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 siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Apakah permintaan dibuat dengan root atau kredensi IAM pengguna.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna terfederasi.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi lebih lanjut, lihat CloudTrail userIdentityelemen. AWS IoT Events tindakan didokumentasikan dalam AWS IoT Events APIreferensi.

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 manapun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari API panggilan publik, sehingga tidak muncul dalam urutan tertentu.

Saat CloudTrail logging diaktifkan di AWS akun Anda, sebagian besar API panggilan yang dilakukan untuk AWS IoT Events tindakan dilacak dalam file CloudTrail log di mana mereka ditulis dengan catatan AWS layanan lainnya. CloudTrailmenentukan 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 dibuat dengan root atau kredensi IAM pengguna.
- 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, file log Anda dienkripsi dengan enkripsi sisi server Amazon S3 (). SSE

Untuk diberi tahu saat pengiriman file log, Anda dapat mengonfigurasi CloudTrail untuk mempublikasikan SNS pemberitahuan Amazon saat file log baru dikirimkan. Untuk informasi selengkapnya, lihat Mengonfigurasi SNS notifikasi Amazon 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 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 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 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 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 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 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 berikut menunjukkan entri CloudTrail log yang menunjukkan `DescribeInput` 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": "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 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 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": "CkZEZXRlY3Rvck1vZGVsMl9saXN0ZGV0ZWN0b3Jtb2RlbHN0ZXN0X2VlOWJkZTk1YT",
    "maxResults": 3
  },
  "responseElements": null,
  "requestID": "6d70f262-da95-4bb5-94b4-c08369df75bb",
  "eventID": "2d01a25c-d5c7-4233-99fe-ce1b8ec05516",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

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 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 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": "CkhjYW5hcnlfdGVzdF9pbnB1dF9saXN0ZGV0ZWN0b3Jtb2RlbHN0ZXN0ZDU3OGZ",
    "maxResults": 3
  },
  "responseElements": null,
  "requestID": "dd6762a1-1f24-4e63-a986-5ea3938a03da",
  "eventID": "c500f6d8-e271-4366-8f20-da4413752469",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```


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 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 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"
}
```

Validasi kepatuhan untuk AWS IoT Events

Untuk mempelajari apakah an AWS layanan berada dalam lingkup program kepatuhan tertentu, lihat AWS layanan di Lingkup oleh Program Kepatuhan AWS layanan 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 AWS layanan ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar AWS yang berfokus pada keamanan dan kepatuhan.
- Arsitektur untuk HIPAA Keamanan dan Kepatuhan di Amazon Web Services — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat HIPAA aplikasi yang memenuhi syarat.

> **Note**
>
> Tidak semua AWS layanan HIPAA memenuhi syarat. Untuk informasi selengkapnya, lihat Referensi Layanan yang HIPAA Memenuhi Syarat.

- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWS Panduan Kepatuhan Pelanggan - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan AWS layanan dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi ()). ISO
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— Ini AWS layanan memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk sumber daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat Referensi kontrol Security Hub.
- Amazon GuardDuty — Ini AWS layanan mendeteksi potensi ancaman terhadap beban kerja Akun AWS, kontainer, dan data Anda dengan memantau lingkungan Anda untuk aktivitas mencurigakan dan berbahaya. GuardDuty dapat membantu Anda mengatasi berbagai persyaratan kepatuhan, seperti PCIDSS, dengan memenuhi persyaratan deteksi intrusi yang diamanatkan oleh kerangka kerja kepatuhan tertentu.
- AWS Audit ManagerIni AWS layanan membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

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 API panggilan yang AWS dipublikasikan untuk mengakses melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Transportasi (TLS). Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Suite cipher dengan kerahasiaan maju yang sempurna (PFS) seperti (Ephemeral Diffie-Hellman) atau DHE (Elliptic Curve Ephemeral Diffie-Hellman). ECDHE Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani dengan menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan IAM prinsipal. Atau Anda bisa menggunakan AWS Security Token Service (AWS STS) untuk membuat kredensial keamanan sementara guna menandatangani permintaan.

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 IAM kebijakan di Panduan AWS IoT Pengembang.

Untuk kemudahan penggunaan, Editor Tag di AWS Management Console 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 -8 UTF
- Panjang nilai maksimum - 255 karakter Unicode di -8 UTF
- Kunci dan nilai tanda peka huruf besar-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 penandaan 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 tag dengan IAM kebijakan

Anda dapat menerapkan izin tingkat sumber daya berbasis tag dalam kebijakan yang Anda gunakan untuk IAM tindakan. AWS IoT Events API Hal ini memberi Anda kontrol yang lebih baik atas sumber daya yang dapat dibuat, dimodifikasi, atau digunakan oleh pengguna.

Anda menggunakan `Condition` elemen (juga disebut `Condition` blok) dengan kunci konteks kondisi berikut dan nilai dalam IAM kebijakan untuk mengontrol akses pengguna (izin) berdasarkan tag 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 API permintaan 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 API permintaan untuk membuat atau memodifikasi sumber daya yang memungkinkan tag.

> **Note**
>
> Kunci konteks kondisi dan nilai dalam IAM kebijakan hanya berlaku untuk AWS IoT Events tindakan tersebut di mana pengenal untuk sumber daya yang dapat diberi tag adalah parameter wajib.

Mengontrol akses menggunakan tag di Panduan AWS Identity and Access Management Pengguna memiliki informasi tambahan tentang penggunaan tag. Bagian referensi IAM JSON kebijakan dari panduan itu memiliki sintaks terperinci, deskripsi, dan contoh elemen, variabel, dan logika evaluasi JSON kebijakan 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"`

```
{
    "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,,, kesalahanAPI, CLI 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

AWS IoT Events Masalah dan solusi umum

Lihat bagian berikut untuk memecahkan masalah kesalahan dan menemukan solusi yang mungkin untuk menyelesaikan masalah. AWS IoT Events

Kesalahan

Kesalahan pembuatan model detektor

Saya mendapatkan kesalahan ketika saya mencoba membuat model detektor.

Solusi

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 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 MQTT pesan atau SNS peringatan.

Solusi

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 versi sebelumnya dari model detektor. Anda mungkin terus menerima peringatan yang ditentukan oleh model detektor sebelumnya. Tunggu setidaknya tujuh menit sebelum Anda memeriksa ulang pembaruan atau melaporkan kesalahan.

Kegagalan pemicu tindakan (saat memenuhi suatu kondisi)

Detektor gagal memicu tindakan atau transisi ke keadaan baru ketika kondisi terpenuhi.

Solusi

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.

Solusi

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`.

Solusi

Jika Anda gunakan BatchPutMessageuntuk 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 fileAPI.

Solusi

Verifikasi bahwa Open SSL 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 Open. SSL

InvalidRequestException pesan

Saya mendapatkan InvalidRequestException ketika saya mencoba untuk menelepon `CreateDetectorModel` dan `UpdateDetectorModel`APIs.

Solusi

Periksa hal berikut untuk membantu menyelesaikan masalah. Untuk informasi lebih lanjut, lihat CreateDetectorModeldan 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>`).

`action.setTimer`Kesalahan Amazon CloudWatch Log

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.

Solusi

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 bahwa ekspresi durasi Anda mengacu pada nilai antara 60-31622400.

Solusi

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 bahwa ekspresi durasi Anda mengacu pada nilai antara 60-31622400.

Solusi

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.

Solusi

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.

Solusi

Lihat SetTimerActiontindakan 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.

Solusi

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.

Solusi

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 adalahJSON, jadi Anda harus menentukan ekspresi yang AWS IoT Events akan mengevaluasi ke string.

Solusi

Jika jenis payload yang ditentukan adalahJSON, AWS IoT Events pertama-tama periksa 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 agar payload tindakan validJSON. Jenis payload yang ditentukan adalahJSON.

Solusi

Pastikan bahwa AWS IoT Events dapat mengevaluasi ekspresi konten Anda untuk payload tindakan menjadi validJSON, jika Anda menentukan jenis payload sebagai. `JSON` AWS IoT Events menjalankan tindakan meskipun tidak AWS IoT Events dapat mengevaluasi ekspresi konten menjadi validJSON.

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>`

Solusi

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`* nilai, Anda harus referensi 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 Referensi untuk input dan variabel dalam ekspresi.

Gagal mengirim pesan ke AWS IoT Events

Pesan: Gagal mengirim pesan ke Acara Iot

Solusi

Anda mungkin mengalami kesalahan ini karena alasan berikut:

- Payload pesan masukan tidak berisi file. `Input attribute Key`
- Tidak `Input attribute Key` berada di JSON jalur yang sama seperti yang ditentukan dalam definisi input.
- Pesan input 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 untuk mempelajari cara mendefinisikan AWS IoT Events Input.

Memecahkan masalah model detektor dengan menjalankan analisis

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, lihatAnalisis model detektor dan informasi diagnostik.

Anda dapat menggunakan AWS IoT Events konsol, API, AWS Command Line Interface (AWS CLI), atau AWS SDKuntuk 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 Menganalisis model detektor (Konsol) atauMenganalisis model detektor (AWS CLI).

Topik

Analisis model detektor dan informasi diagnostik

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 bersifat khusus per Wilayah. 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

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

- Location
- supported-actions
- service-limits
- structure

- expression-syntax
- data-type
- referenced-data
- referenced-resource

`Location`

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, lihatTindakan yang didukung untuk menerima data dan memicu tindakan.

`supported-actions`

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, lihatTindakan yang didukung untuk menerima data dan memicu tindakan.
- Pesan: DetectorModel definisi memiliki *`aws-layanan`* tindakan, tetapi *`aws-layanan`* layanan tidak didukung di wilayah *`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.

`service-limits`

Hasil analisis dengan informasi tentang`service-limits`, sesuai dengan pesan kesalahan berikut:

- Pesan: Ekspresi Konten yang diizinkan dalam muatan melebihi batas *content-expression-size* byte dalam acara *event-name* di negara bagian *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 untuk timer *timer-name* Setidaknya harus *minimum-timer-duration* Detik panjang.

  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 acara 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 negara 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 acara yang diizinkan per negara 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 di *detector-models-per-input* rute model 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.

`structure`

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`* jenis. Harap dibagi menjadi Tindakan terpisah.

  Solusi: Anda mungkin menerima pesan galat ini jika Anda menetapkan dua atau lebih tindakan dalam satu bidang dengan menggunakan API operasi untuk membuat atau memperbarui model detektor Anda. Anda dapat menentukan array `Action` objek. Pastikan Anda mendefinisikan setiap tindakan sebagai objek terpisah.

- Pesan: The 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: Definisi initialStateName *`initial-state-name`* tidak sesuai dengan Negara 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 parameter`SetTimerAction`. Untuk informasi lebih lanjut, lihat SetTimerActiondi AWS IoT Events APIReferensi.

- 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 Ekspresi 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

`expression-syntax`

Hasil analisis dengan informasi tentang`expression-syntax`, sesuai dengan pesan kesalahan berikut:

- Pesan: Ekspresi payload Anda {*expression*} tidak valid. Jenis payload yang ditentukan adalahJSON, jadi Anda harus menentukan ekspresi yang AWS IoT Events akan mengevaluasi ke string.

  Solusi: Jika jenis payload yang ditentukan adalahJSON, AWS IoT Events pertama-tama periksa 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 lebih lanjut, lihat SetVariableAction, di AWS IoT Events APIReferensi.

- Pesan: Kami tidak dapat mengurai ekspresi atribut Anda (*attribute-name*) untuk tindakan DynamoDB. Masukkan ekspresi dengan sintaks yang benar.

  Solusi: Anda harus menggunakan ekspresi untuk semua parameter `DynamoDBAction` di. template substitusi. Untuk informasi lebih lanjut, lihat D ynamoDBAction di AWS IoT Events APIReferensi.

- Pesan: Kami tidak dapat mengurai ekspresi Anda tableName untuk tindakan D ynamoDBv 2. 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 lebih lanjut, lihat D ynamoDBv 2Action di AWS IoT Events APIReferensi.

- Pesan: Kami tidak dapat mengevaluasi ekspresi Anda menjadi validJSON. Tindakan D ynamoDBv 2 hanya mendukung jenis JSON payload.

  Solusi: Jenis muatan untuk `DynamoDBv2` harusJSON. Pastikan bahwa AWS IoT Events dapat mengevaluasi ekspresi konten Anda agar payload validJSON. Untuk informasi lebih lanjut, lihat D ynamoDBv 2Action, di AWS IoT Events APIReferensi.

- 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 ($ variabel.*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 Muatan di AWS IoT Events APIReferensi.

- 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.

- Pesan: Anda *fieldName* karena `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* untuk`IotSitewiseAction`. Pastikan *fieldName* menggunakan ekspresi yang AWS IoT Events dapat mengurai. Untuk informasi lebih lanjut, lihat IotSiteWiseActiondi AWS IoT Events APIReferensi.

`data-type`

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 untuk *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* nilai, Anda harus referensi 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 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*] ditemukan untuk *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 baik 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 JSON format.

  Solusi: Anda mungkin menerima kesalahan ini jika jenis payload yang Anda tentukan adalahJSON, 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 Anda {*payload-expression*} tidak akan dapat diuraikan saat runtime sebagai valid. JSON

  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 `IotSitewiseAction` 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 `IotSitewiseAction` 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`*] digunakan untuk `setTimer` tindakan tidak mengevaluasi `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`*] digunakan dengan operan dari operator perbandingan [*`operator`*] tidak kompatibel dalam ekspresi berikut: *`expression`*

  Solusi: Tipe data yang disimpulkan untuk operan *`operator`* dalam ekspresi kondisional (*`expression`*) dari 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 ekspresi.

`referenced-data`

Hasil analisis dengan informasi tentang`referenced-data`, sesuai dengan pesan kesalahan berikut:

- Pesan: Terdeteksi Timer rusak: timer *`timer-name`* digunakan dalam ekspresi tetapi tidak pernah diatur.

  Solusi: Anda mungkin menerima pesan galat ini jika Anda menggunakan timer yang tidak disetel. Anda harus mengatur timer sebelum Anda 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 diatur.

  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.

`referenced-resource`

Hasil analisis dengan informasi tentang`referenced-resource`, sesuai dengan pesan kesalahan berikut:

- Pesan: Definisi Model Detektor berisi referensi ke Input yang tidak ada.

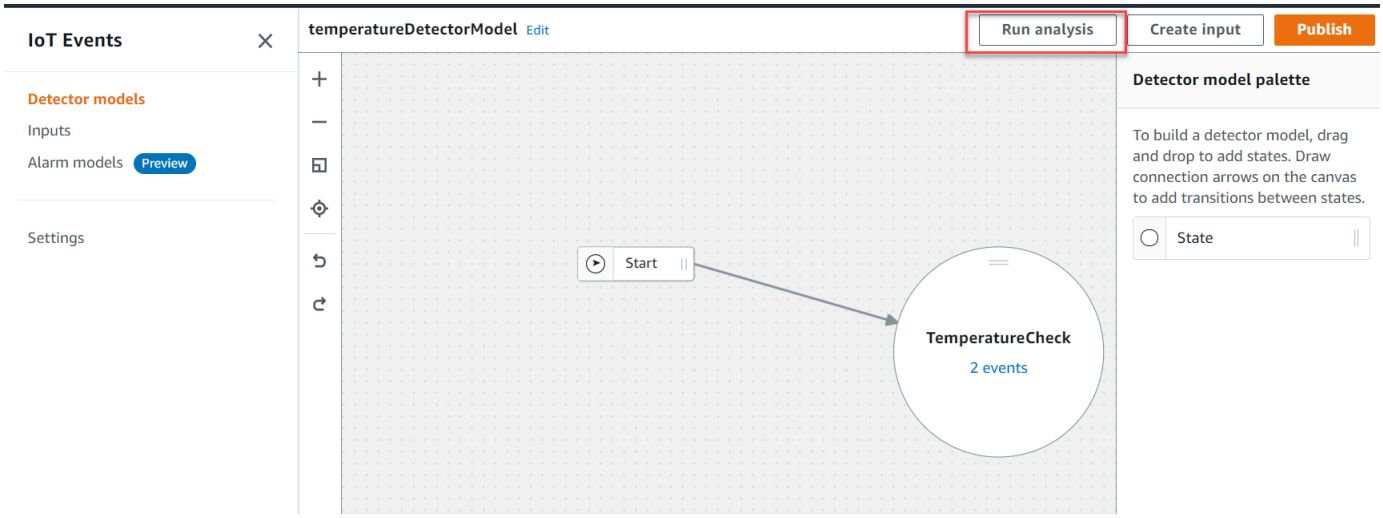
  Solusi: Anda mungkin menerima pesan galat ini jika Anda menggunakan ekspresi untuk mereferensikan masukan yang tidak ada. Pastikan ekspresi Anda mereferensikan input yang ada dan masukkan nama input yang benar. Jika Anda tidak memiliki masukan, buat terlebih dahulu.
- Pesan: Definisi Model Detektor berisi 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).

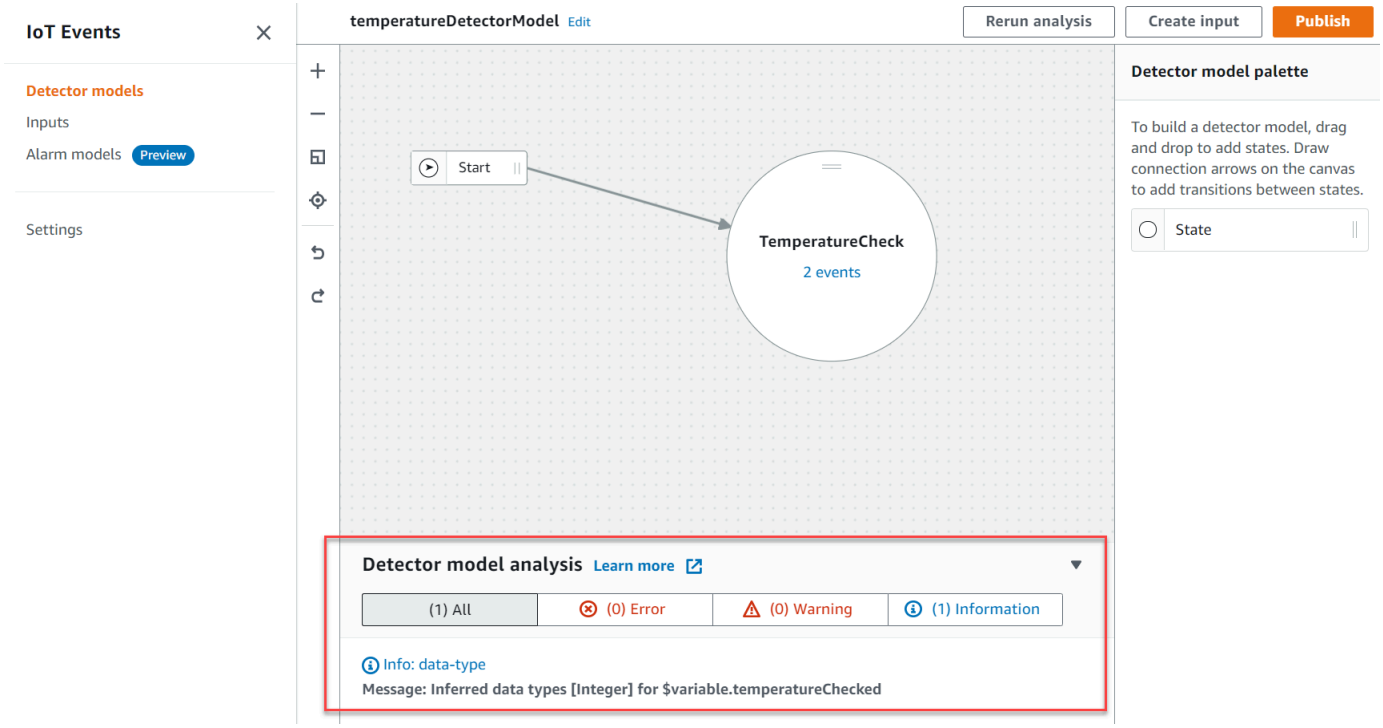
Menganalisis model detektor (Konsol)

Langkah-langkah berikut menggunakan AWS IoT Events konsol 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.



> **Note**
>
> Setelah AWS IoT Events mulai menganalisis model detektor Anda, Anda memiliki waktu hingga 24 jam untuk mengambil hasil analisis.

Menganalisis model detektor (AWS CLI)

Langkah-langkah berikut menggunakan AWS CLI untuk menganalisis model detektor.

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 diJSON.
  4. Buka JSON file model detektor.
  5. Anda hanya membutuhkan `detectorModelDefinition` objek. Hapus yang berikut ini:
     - Braket keriting pertama ({) di bagian atas halaman
     - `detectorModel`Garis
     - `detectorModelConfiguration`Objeknya
     - 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 diJSON.

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 mengarahkan Anda ke semua API operasi secara AWS IoT Events rinci, termasuk permintaan sampel, tanggapan, dan kesalahan untuk protokol layanan web yang didukung.

AWS IoT Events tindakan

Anda dapat menggunakan AWS IoT Events API perintah 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 AWS IoT Events APIReferensi.

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 API perintah AWS IoT Events Data 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 dalam AWS IoT Events APIReferensi.

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 RSS umpan.

Perubahan	Deskripsi	Tanggal
Peluncuran wilayah	AWS IoT Events sekarang tersedia di wilayah Asia Pasifik (Mumbai).	30 September 2021
Peluncuran wilayah	AWS IoT Events sekarang tersedia di Wilayah AWS GovCloud (AS-Barat).	22 September 2021
Memecahkan masalah model detektor dengan menjalankan analisis	AWS IoT Events Sekarang dapat menganalisis model detektor Anda dan menghasil kan hasil analisis yang dapat Anda gunakan untuk memecahkan masalah model detektor Anda.	23 Februari 2021
Peluncuran wilayah	Diluncurkan AWS IoT Events di China (Beijing).	30 September 2020
Penggunaan ekspresi	Menambahkan contoh untuk menunjukkan cara menulis ekspresi.	22 September 2020
Pemantauan dengan alarm	Alarm membantu Anda memantau data untuk perubahan. Anda dapat membuat alarm yang mengirim notifikasi saat ambang batas dilanggar.	1 Juni 2020

Pembaruan sebelumnya

Tabel berikut menjelaskan perubahan penting pada Panduan AWS IoT Events Pengembang sebelum 18 September 2020.

Perubahan	Deskripsi	Tanggal
Menambahkan validasi tipe ke referensi Ekspresi	Menambahkan informasi validasi tipe ke referensi Ekspresi.	3 Agustus 2020
Menambahkan peringatan Wilayah untuk layanan lain	Menambahkan peringatan tentang memilih wilayah yang sama untuk AWS IoT Events dan AWS layanan lainnya.	7 Mei 2020
Penambahan, pembaruan	• Fitur Kustomisasi Payload • Tindakan acara baru: Amazon DynamoDB dan AWS IoT SiteWise	27 April 2020
Menambahkan fungsi bawaan untuk ekspresi bersyarat model detektor	Menambahkan fungsi bawaan untuk ekspresi kondisional model detektor.	10 September 2019
Contoh model detektor yang ditambahkan	Menambahkan contoh untuk model detektor.	5 Agustus 2019
Ditambahkan tindakan acara baru	Ditambahkan tindakan acara baru untuk: • Lambda • Amazon SQS • Kinesis Data Firehose • AWS IoT Events masukan	19 Juli 2019
Penambahan, koreksi	• Deskripsi `timeout()` fungsi yang diperbarui.	11 Juni 2019

Perubahan	Deskripsi	Tanggal
	• Menambahkan praktik terbaik mengenai ketidakak tifan akun.	
Kebijakan izin yang diperbarui dan opsi debug konsol	• Memperbarui kebijakan izin konsol. • Diperbarui gambar halaman opsi debug konsol.	5 Juni 2019
Pembaruan	AWS IoT Events layanan terbuka untuk ketersediaan umum.	30 Mei 2019
Penambahan, pembaruan	• Informasi keamanan yang diperbarui. • Menambahkan contoh model detektor beranotasi.	22 Mei 2019
Menambahkan contoh dan izin yang diperlukan	Menambahkan contoh SNS payload Amazon; penambaha n izin yang diperlukan untuk. `CreateDetectorModel`	17 Mei 2019
Menambahkan informasi keamanan tambahan	Menambahkan informasi ke bagian keamanan.	9 Mei 2019
Rilis pratinjau terbatas	Rilis pratinjau dokumentasi terbatas.	28 Maret 2019

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