



このコンテンツは公開から3年以上経過しており内容が古い可能性があります  
最新情報については[サービス別資料](#)もしくはサービスのドキュメントをご確認ください

# [AWS Black Belt Online Seminar]

## AWS IoT Events

サービスカットシリーズ

Prototyping Solutions Architect 市川 純

2020/9/16

AWS 公式 Webinar

<https://amzn.to/JPWebinar>



過去資料

<https://amzn.to/JPArchive>



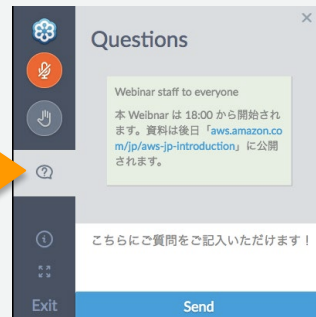
# AWS Black Belt Online Seminar とは

「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾンウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

## 質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- 今後のロードマップに関するご質問は  
お答えできませんのでご了承下さい

- ① 吹き出しをクリック
- ② 質問を入力
- ③ Sendをクリック



Twitter ハッシュタグは以下をご利用ください  
#awsblackbelt

# 内容についての注意点

- 本資料では2020年9月16日時点のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(<http://aws.amazon.com>)にてご確認ください。
- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。
- 価格は税抜表記となっております。日本居住者のお客様には別途消費税をご請求させていただきます。
- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at <http://aws.amazon.com/agreement/>. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

## 自己紹介

名前：

市川 純 (いちかわ じゅん)

所属：

アマゾン ウェブ サービス ジャパン株式会社

デジタルトランスフォーメーション本部

プロトタイピング ソリューション アーキテクト

担当：

IoTに関するプロトタイピングやお客様の支援

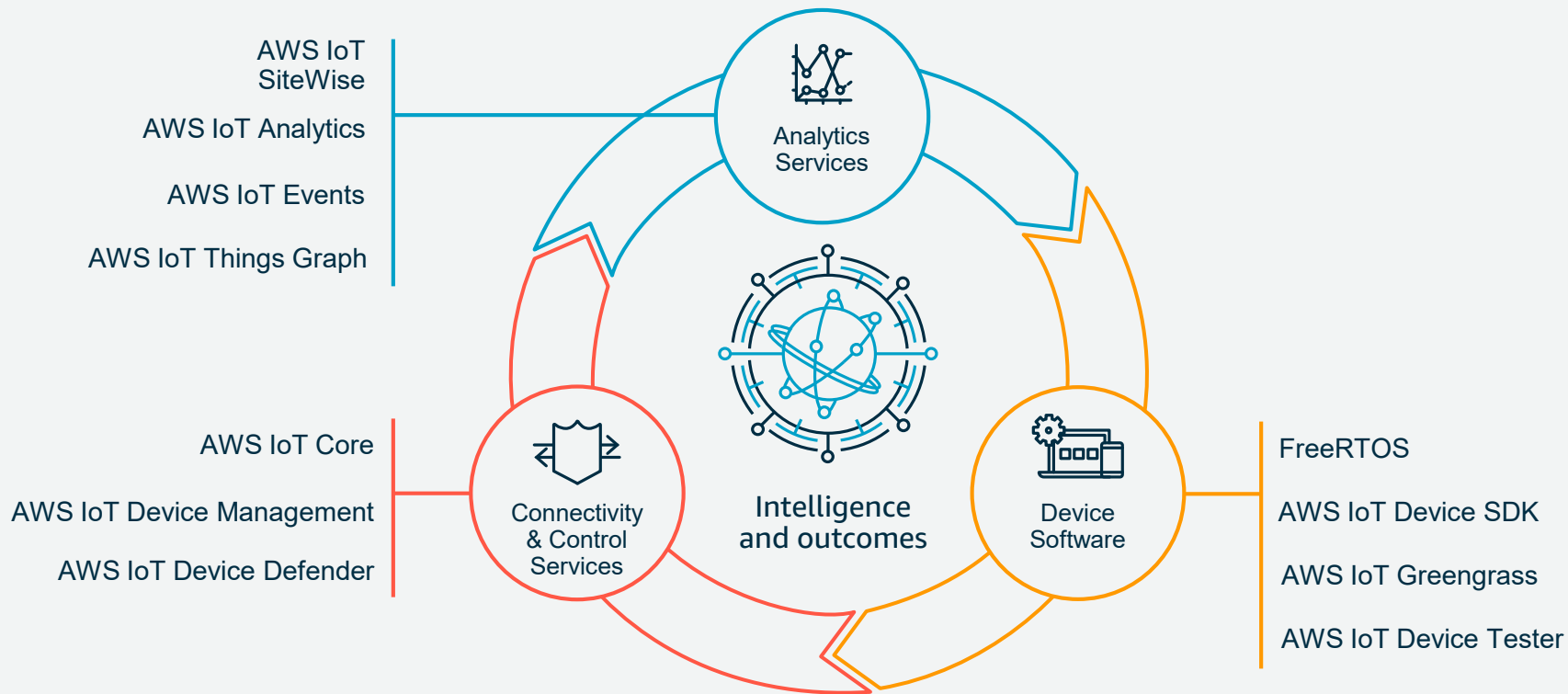


# 本日のアジェンダ

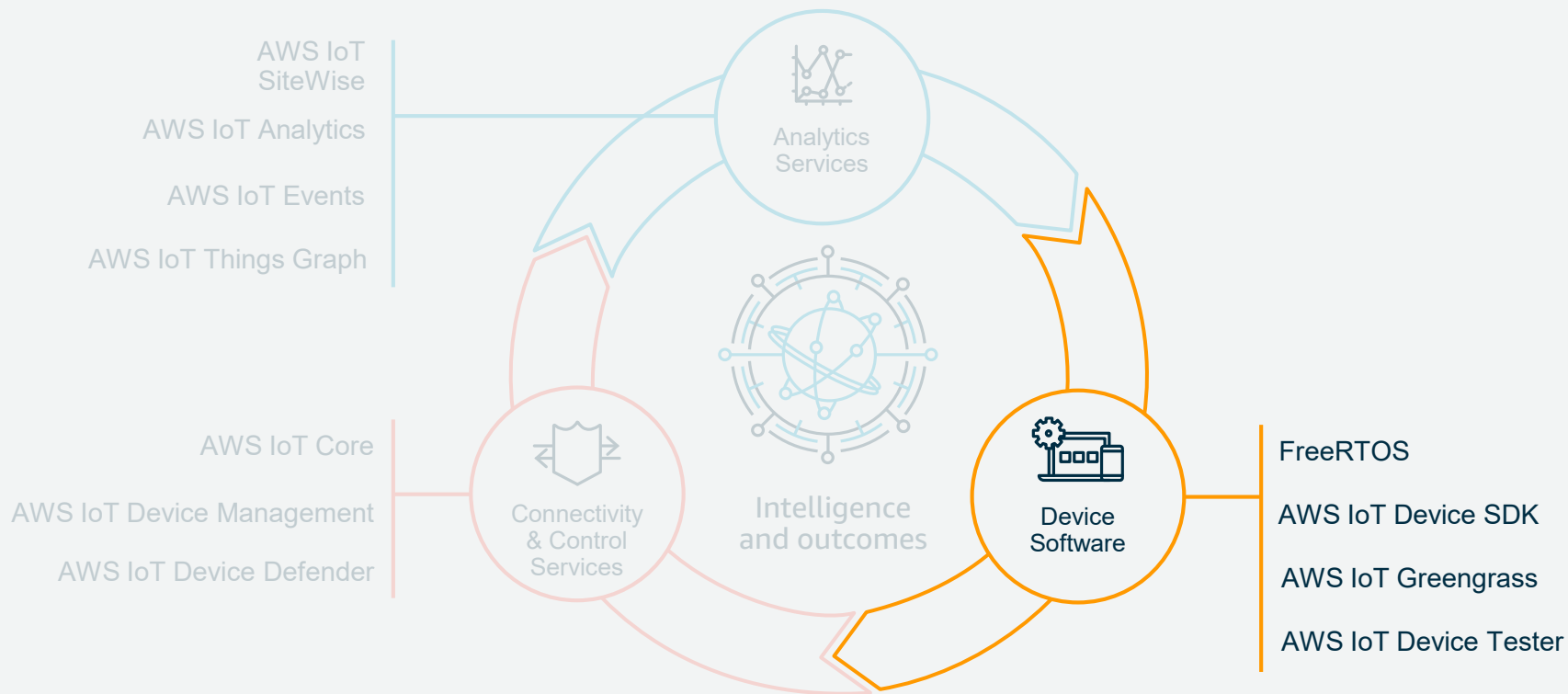
- AWS IoTサービスの紹介
- AWS IoT Eventsについて
- AWS IoT Eventsを使ったユースケースの紹介
- AWS IoT Eventsの始め方
- まとめ
- Q&A

# AWS IoT サービスの紹介

# AWS IoTのサービス

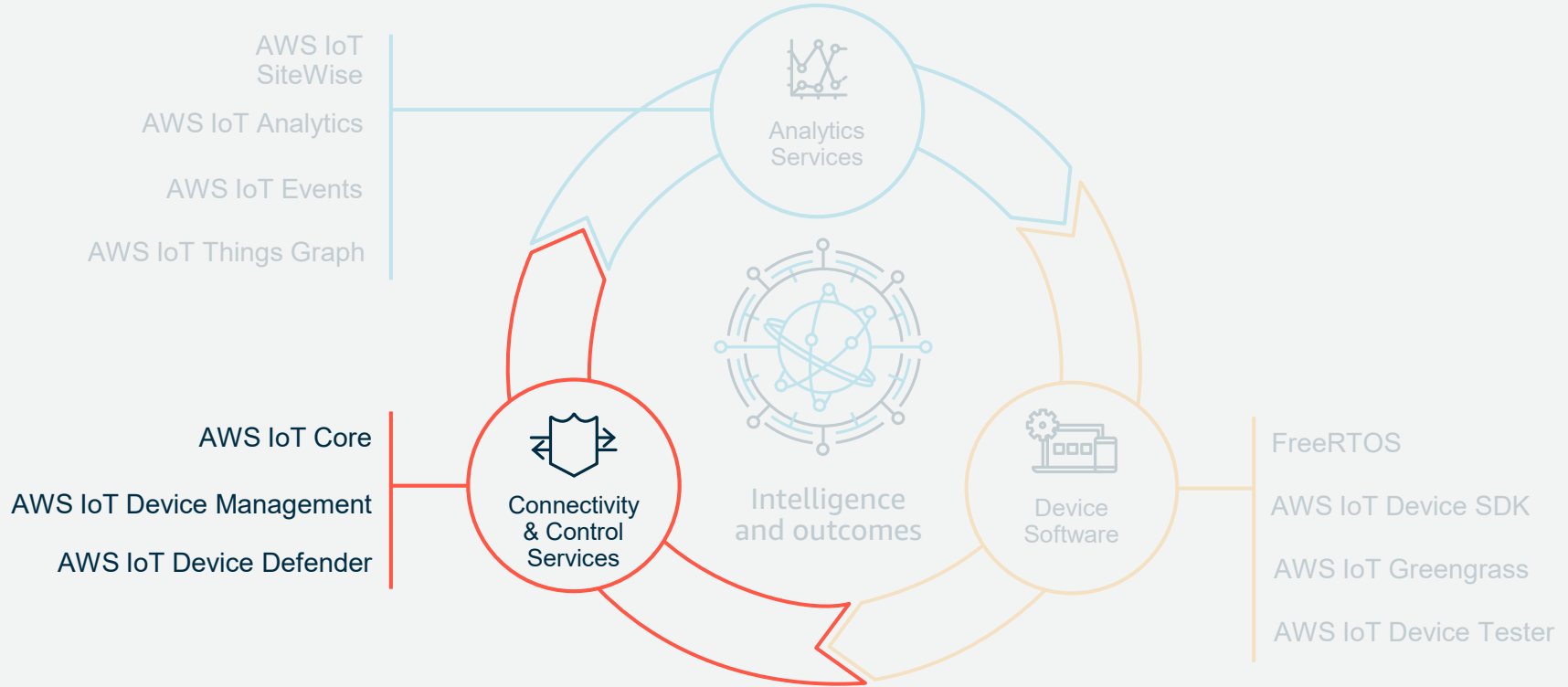


# AWS IoTのサービス

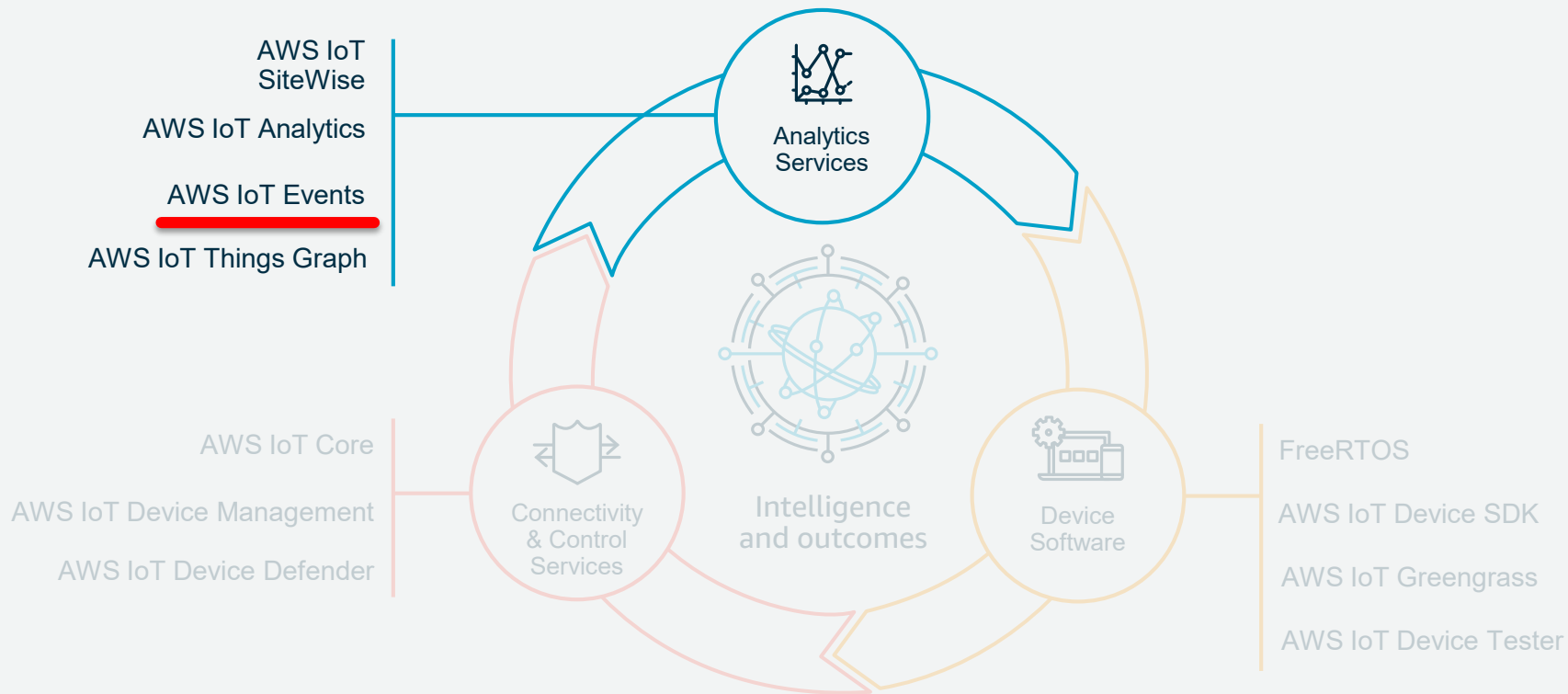




# AWS IoTのサービス



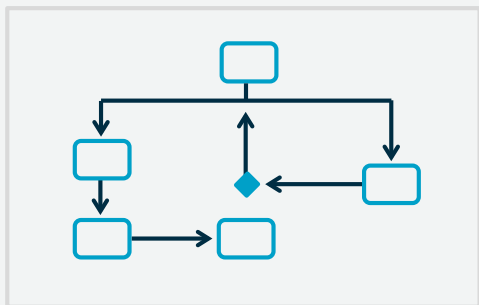
# AWS IoTのサービス



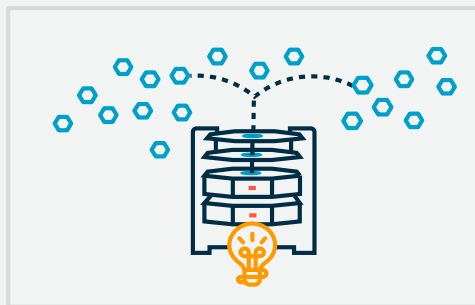
# AWS IoT Eventsについて

# AWS IoT Events

AWS IoT Eventsを使用すると、機器および一連のデバイスからのデータを継続的に監視し、イベントが発生したときに適切な対応をトリガーできます



複数の条件を評価しながら  
デバイスやシステムの状態を  
判断するロジックを構築



何千ものセンサーや他のソース  
のデータからイベントを検出

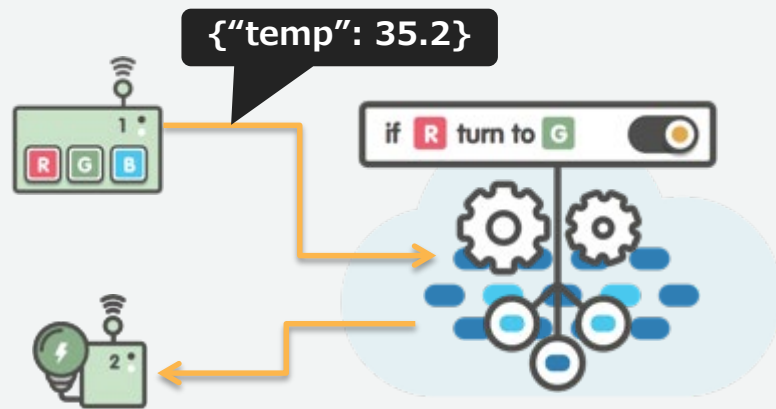


イベントの結果  
最適なアクションを実行

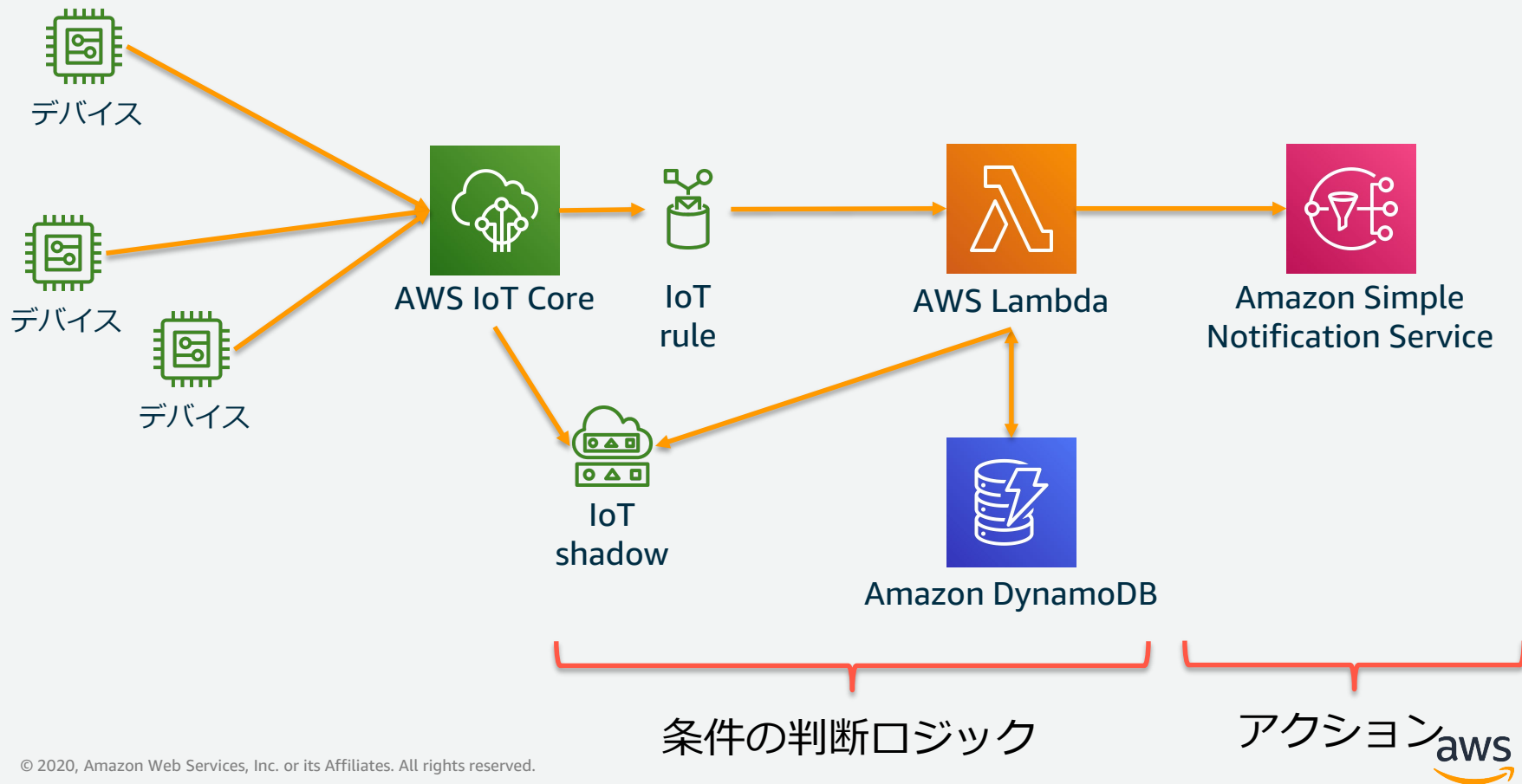
# IoT Coreのルールエンジンを利用した場合

- SQL文を使ったトピックのフィルタ
- オプションの**WHERE**句で条件を記述することが可能
- JSONをサポート
- トピックのフィルタの例

```
SELECT *  
FROM 'data/meguro_building/1f/+/sensor'  
WHERE temp > 35.0
```



# IoT Coreのルールエンジンを利用した場合



# AWS IoT Eventsの編集画面

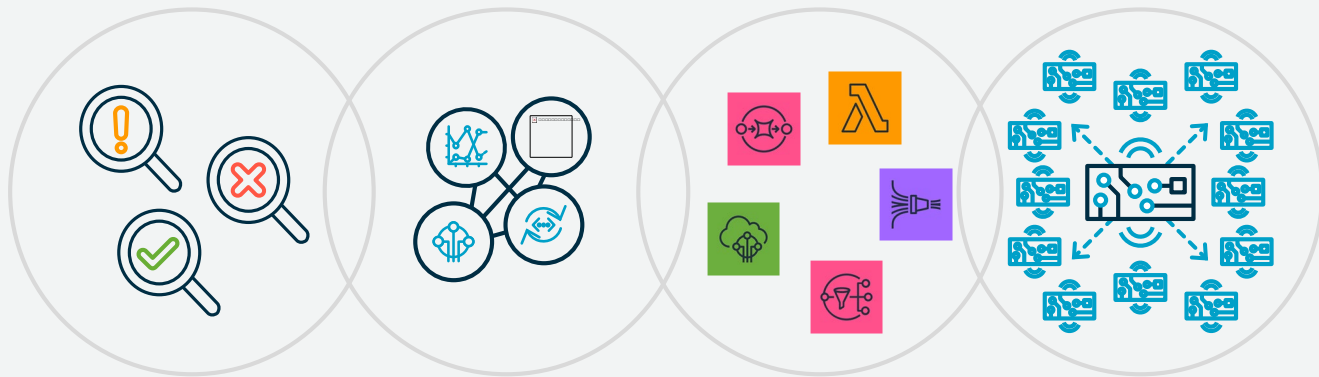
The screenshot displays the AWS IoT Events console interface for editing a state machine named 'device\_connected\_status'. The top navigation bar includes the AWS logo, 'サービス' (Services), 'リソースグループ' (Resource Groups), and location/language settings for '東京' (Tokyo) and 'サポート' (Support). The breadcrumb shows 'device\_connected\_status' with an '編集' (Edit) link. On the right, there are buttons for '入力の作成' (Create Input) and '発行' (Publish).

The main area contains a state machine diagram with three states: 'disconencted' (1 event), 'alert' (1 event), and 'Connected' (1 event). The transitions are as follows:

- 'disconencted' to 'alert' via 'to\_alert'
- 'disconencted' to 'Connected' via 'to\_disconnected'
- 'alert' to 'Connected' via 'to\_connected'
- 'Connected' to 'disconencted' via 'to\_connected'

A '開始' (Start) button is positioned below the diagram. On the right sidebar, the '探知器モデルパレット' (Detector Model Palette) section contains a search bar with the text '状態' (State).

# AWS IoT Eventsの特徴



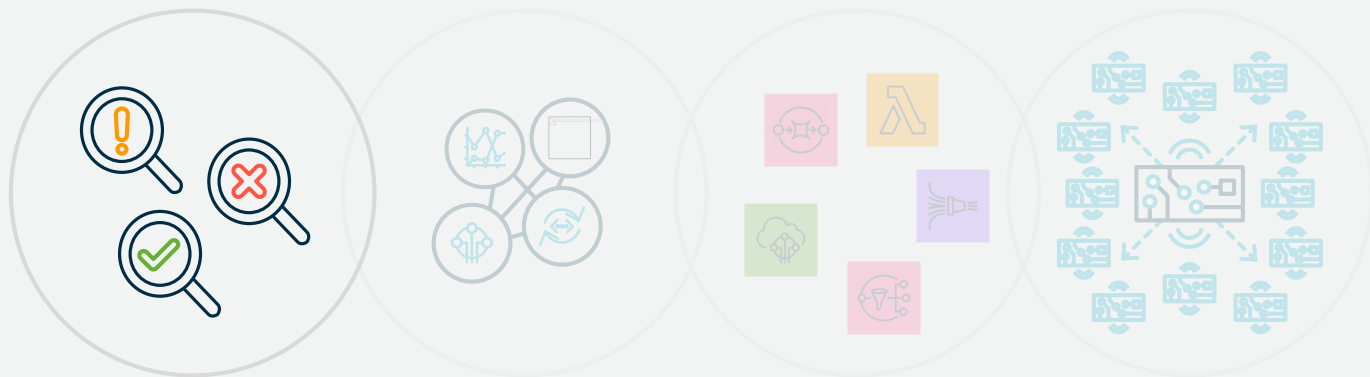
複数のソースから  
入力を受け入れ

簡単な論理式を使って  
複雑なイベントのパ  
ターンを認識

イベントに基づいた  
アクションの実行

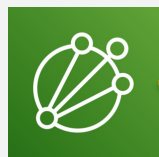
スケーラビリティ





## 複数のソースから入力を受け入れ

# 複数のソースから入力を受け入れ



AWS IoT Analytics



AWS IoT Core



AWS IoT Events



BatchPutMessage API



AWS IoT Events

# 入力データのフォーマットはIoT Events側で定義

- 入力データはJSON形式をサポート

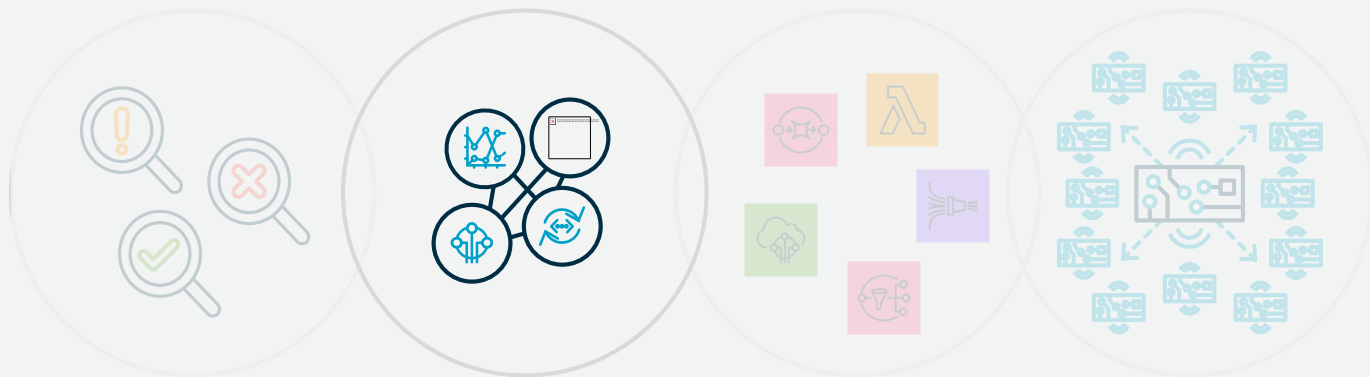
```
{  
  "motorid": "Fulton-A32",  
  "sensorData": {  
    "pressure": 23,  
    "temperature": 47  
  }  
}
```

- IoT Eventsの探知器モデルからは、このデータを以下のように参照します
  - \$input. **インプット名**.motorid
  - \$input. **インプット名**.sensorData.pressure
  - \$input. **インプット名**.sensorData.temperature

固定

自分で命名

JSONの階層を"."で区切る



## 簡単な論理式を使って複雑なイベントのパターンを認識

# 探知器モデル



サービス リソースグループ



東京

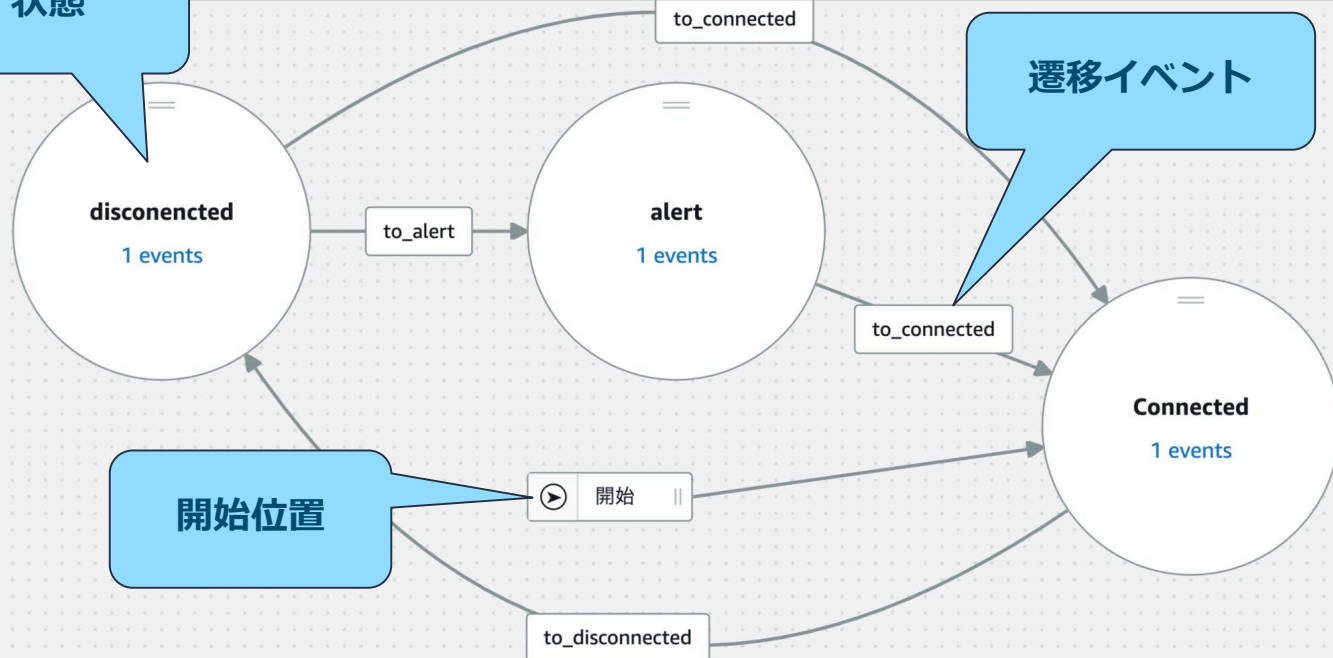
サポート

状態

編集

入力の作成

発行



## 探知器モデルパレット

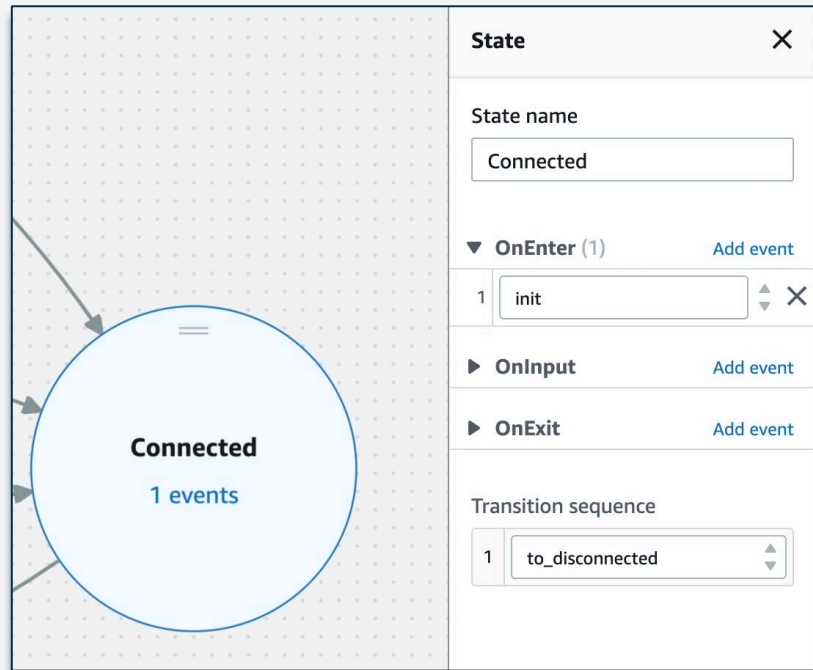
ディテクターモデルを構築するには、ドラッグアンドドロップして状態を追加します。ステート間に移行を追加するには、接続矢印を引っ張ります。



状態

# 状態

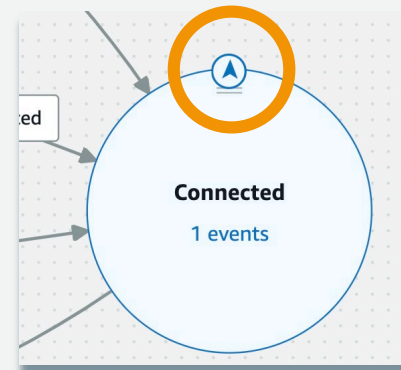
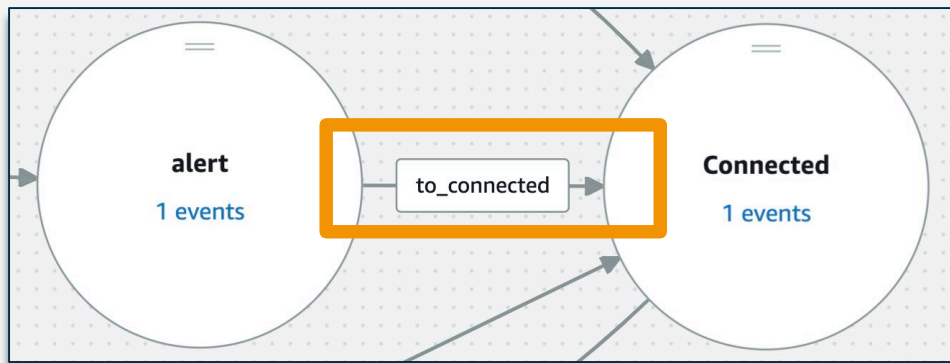
- 3種類のイベントを持つ
  - OnEnter
    - この状態に遷移してきた時に実行
  - OnInput
    - データが入力された時に実行
  - OnExit
    - この状態から抜ける時に実行



- それぞれのイベントでは条件を記述することが出来、その条件を満たしている時に、Actionを実行することが可能

# 遷移イベント

- 2つの状態間(Origin State, Destination State)の遷移条件を表します
- 条件を満たすと、Origin StateからDestination Stateに遷移する
- 遷移イベントが発生した際に、アクションを指定することが可能

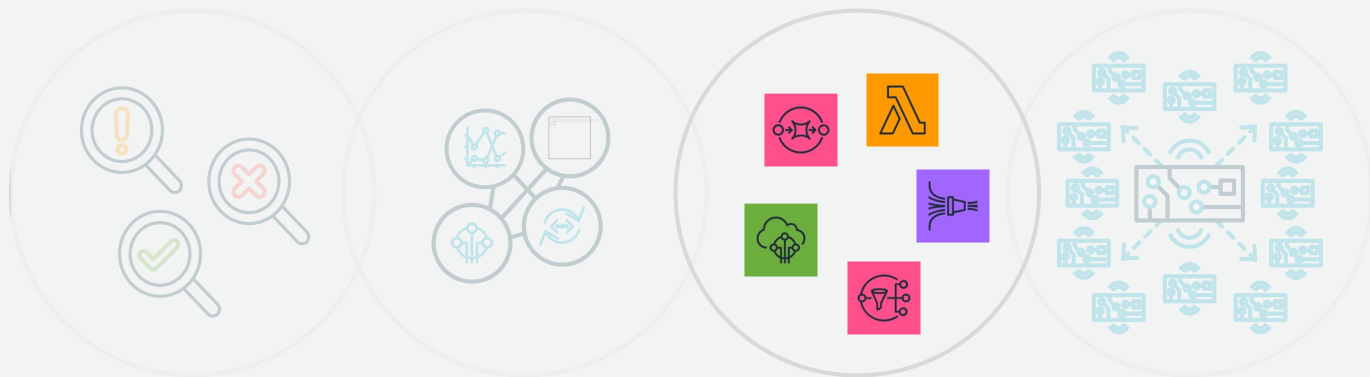


# 利用可能な条件式

- True, Falseを結果と返す式を指定
- 不等号(<, >, =)、四則演算(+, -, /, \*)、論理式(^, |, &, !, ~)が利用可能
- 組み込み関数
  - 型変換(convert)
  - 文字列マッチング(startWith, contains, など)
  - ビット演算(bitor, bitandなど)
  - timeout(タイマーの完了を判断)

<https://docs.aws.amazon.com/iotevents/latest/developerguide/expression-syntax.html>





## イベントに基づいたアクションの実行

# 利用できるアクション(ビルトイン)

- `setTimer`
  - タイマーの作成
- `resetTimer`
  - タイマーのリセット
- `clearTimer`
  - タイマーの削除
- `setVariable`
  - 変数の設定

# 利用できるアクション(AWSサービス連携)



Amazon Simple Notification Service



Amazon Simple Queue Service



Amazon Kinesis Data Firehose



Amazon DynamoDB



AWS IoT Events



AWS IoT SiteWise



AWS IoT Core



AWS Lambda

# カスタムペイロード

- アクションを定義する際に、対象のサービスにペイロード (データ)を文字列またはJSONで渡すことができます
- ペイロードで指定できる情報は
  - 固定の文字列
  - `$variable` で参照できる変数
  - `$input` で参照できる入力データ
- 置き換えテンプレートなど利用して、SNSに送る情報をデータだけではなく、文章として作ることもできます

[https://docs.aws.amazon.com/iotevents/latest/apireference/API\\_Payload.html](https://docs.aws.amazon.com/iotevents/latest/apireference/API_Payload.html)

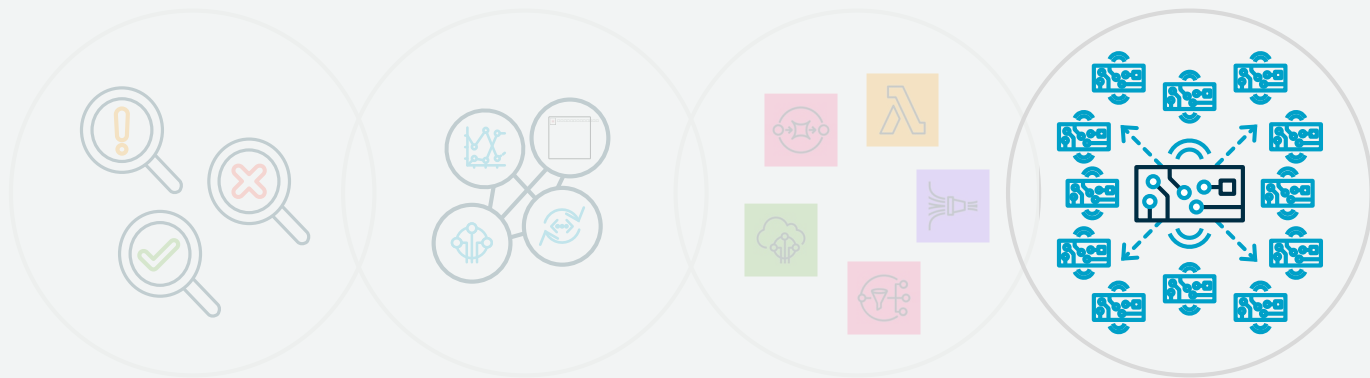
# 置き換えテンプレート

**`${}`**と書くことで置き換えテンプレートを利用できます。変数の参照や、関数の利用など

- 例1)
- 変数を参照**      **変数名**
- `'${'Sensor ' + variable.SensorID}'`
- “Sensor 10” のような変数の値と文字列を組み合わせた値が取得できる

- 例2)
- 入力を参照**      **入力名**      **入力の要素**
- `'Sensor 10: input.input_name.SensorID == 10}'`
- “Sensor 10: true” のように入力データを判定した結果が取得できる

<https://docs.aws.amazon.com/iotevents/latest/developerguide/expression-syntax.html#expression-substitution-template>



# スケーラビリティ

# 入力データに合わせてスケール

- デバイスごとに状態を参照することが可能(この例は一意的キー毎に探知器モデルを作成)
- 新しいデバイス(キー)が登場すると自動的に状態の管理を開始

ディテクター (28)

キーの値	作成日	最終更新時刻	現在の状態
AFRStick	Tue Apr 21 2020	Tue Apr 21 2020 9:38:45AM	alert
GGDiveDeep_Core	Mon Apr 20 2020	Tue Apr 21 2020 9:38:45AM	ok
GGDiveDeep_Core-c00	Mon Apr 20 2020	Tue Apr 21 2020 9:38:45AM	ok
clientid	Thu Apr 23 2020	Mon Apr 27 2020 9:38:45AM	ok
idt-8330302259920487884	Wed Apr 22 2020	Wed Apr 22 2020 9:38:45AM	ok
idt-8330302259920487884-c00	Wed Apr 22 2020	Wed Apr 22 2020 9:38:45AM	ok
idt-8435212291412673842	Fri Apr 24 2020	Fri Apr 24 2020 10:38:45AM	ok
idt-8435212291412673842-c00	Fri Apr 24 2020	Fri Apr 24 2020 10:38:45AM	ok
iotconsole-1587359545339-1	Mon Apr 20 2020	Mon Apr 20 2020 9:38:45AM	ok
iotconsole-1587359707554-2	Mon Apr 20 2020	Mon Apr 20 2020 9:38:45AM	ok

AFRStick

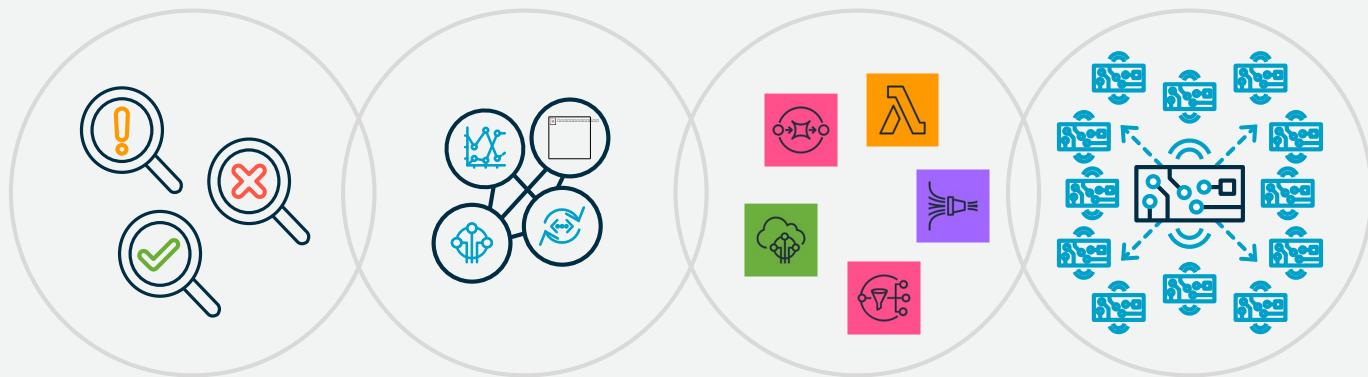
CloudWatch Logs で表示

一般的な情報

状態	作成日	最終更新時刻
alert	Tue Apr 21 2020	Tue Apr 21 2020 9:38:45AM

変数

名前	値
ipAddress	"1[REDACTED]"



# 探知器モデルについて



# 探知器の生成メソッド

- 一意のキー毎に探知器を作成
  - デバイス毎など、複数のものに対して状態を管理したい時に向いている
- 単一の探知器を作成
  - 複数のデバイス(情報)から構成されるシステムの状態を管理したい時に向いている

# 探知器モデルの評価方法

- バッチ評価
  - 変数が更新され、すべてのイベント条件が評価された後にのみイベントが実行されます
- シリアル評価
  - 変数が更新され、イベント条件はイベントが定義されている順序で評価されます

# 探知器の評価方法(例 シリアル)

▼ OnInput (3) <span>Add event</span>	
1	OnInput1 <span>⬆️</span> <span>⬆️</span> <span>✖</span>
2	OnInput2 <span>⬆️</span> <span>⬆️</span> <span>✖</span>
3	OnInput3 <span>⬆️</span> <span>⬆️</span> <span>✖</span>

## 入力データ

```
{  
  "tempture": 30  
}
```

それぞれのイベントの条件とアクション

### OnInput\_1

- 条件  
\$input.input\_name.tempture > 20
- アクション  
変数 tempture に \$input.input\_name.tempture を設定

### OnInput\_2

- 条件  
\$variable.tempture > 20
- アクション  
変数 status に文字列 "greater than 20" を設定

### OnInput\_3

- 条件  
\$variable.tempture <= 20
- アクション  
変数 status に文字列 "lower or equal 20" を設定

# 探知器の評価方法(例 バッチ)

▼ OnInput (3) <span>Add event</span>	
1	OnInput1 <span>⬆️</span> <span>⬆️</span> <span>✖</span>
2	OnInput2 <span>⬆️</span> <span>⬆️</span> <span>✖</span>
3	OnInput3 <span>⬆️</span> <span>⬆️</span> <span>✖</span>

それぞれのイベントの条件とアクション

## OnInput\_1

- 条件  
`$input.input_name.tempture > 20`
- アクション  
変数 `tempture` に `$input.input_name.tempture` を設定

## OnInput\_2

- 条件  
`$variable.tempture > 20`
- アクション  
変数 `status` に文字列 "greater than 20" を設定

## OnInput\_3

- 条件  
`$variable.tempture <= 20`
- アクション  
変数 `status` に文字列 "lower or equal 20" を設定

## 入力データ

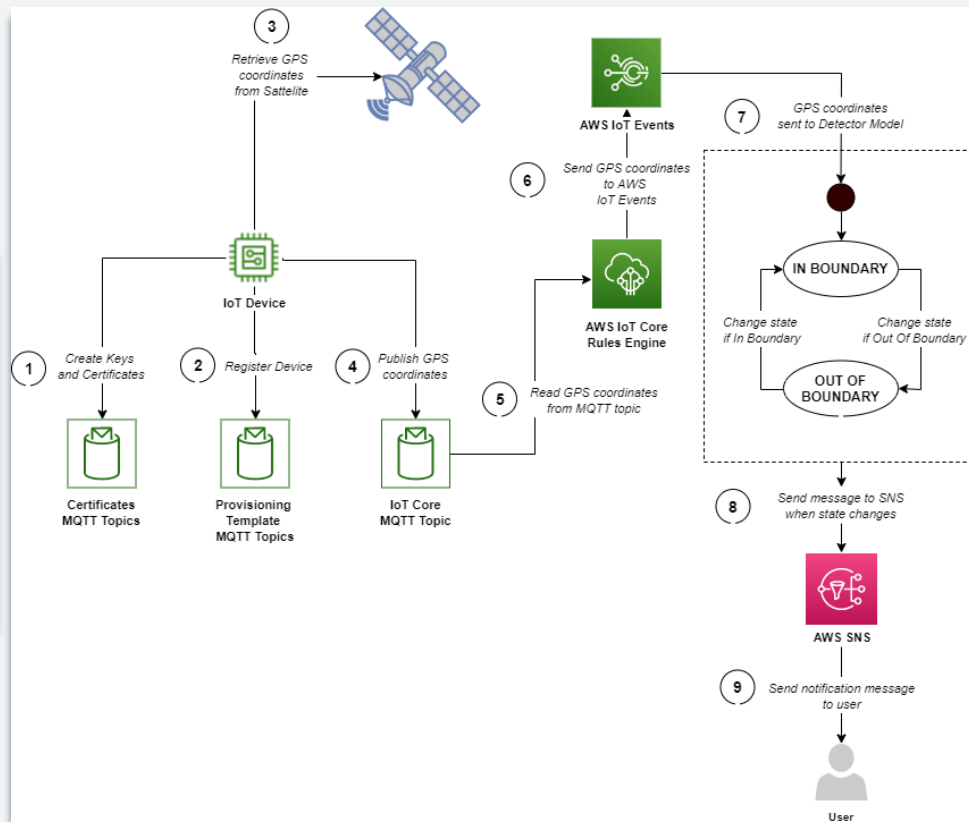
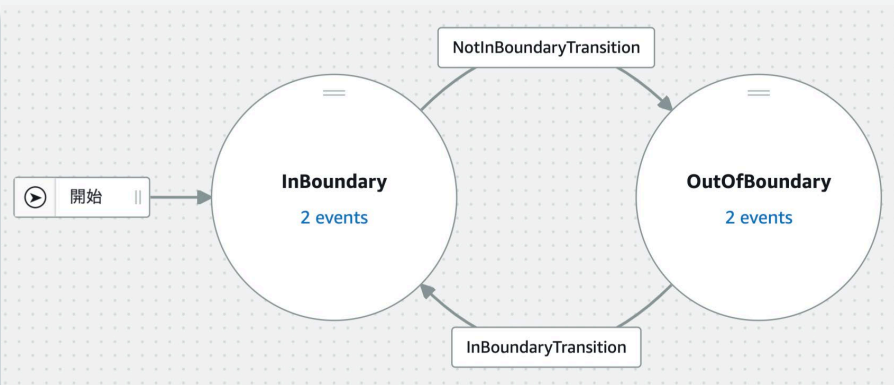
```
{  
  "tempture": 30  
}
```

# AWS IoT Eventsを使った ユースケースの紹介

# 1) デバイスの位置情報を監視

<https://aws.amazon.com/jp/blogs/news/monitor-iot-device-geolocation-with-aws-iot-events/>

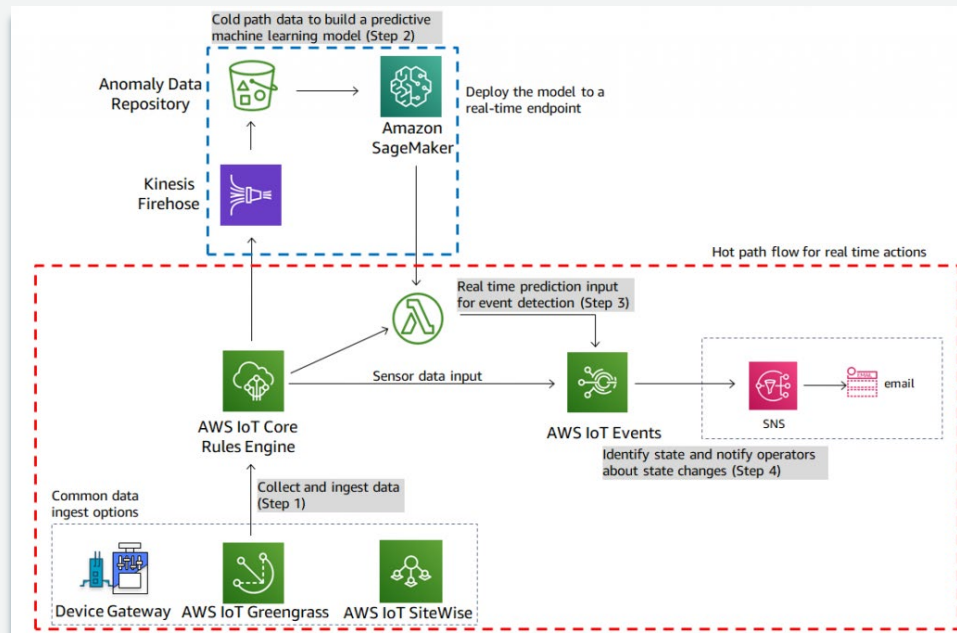
- 車両が予想されるジオロケーションの境界から外れるとサポートチームに通知



## 2) AWS IoT EventsとMLを組み合わせた設備の管理

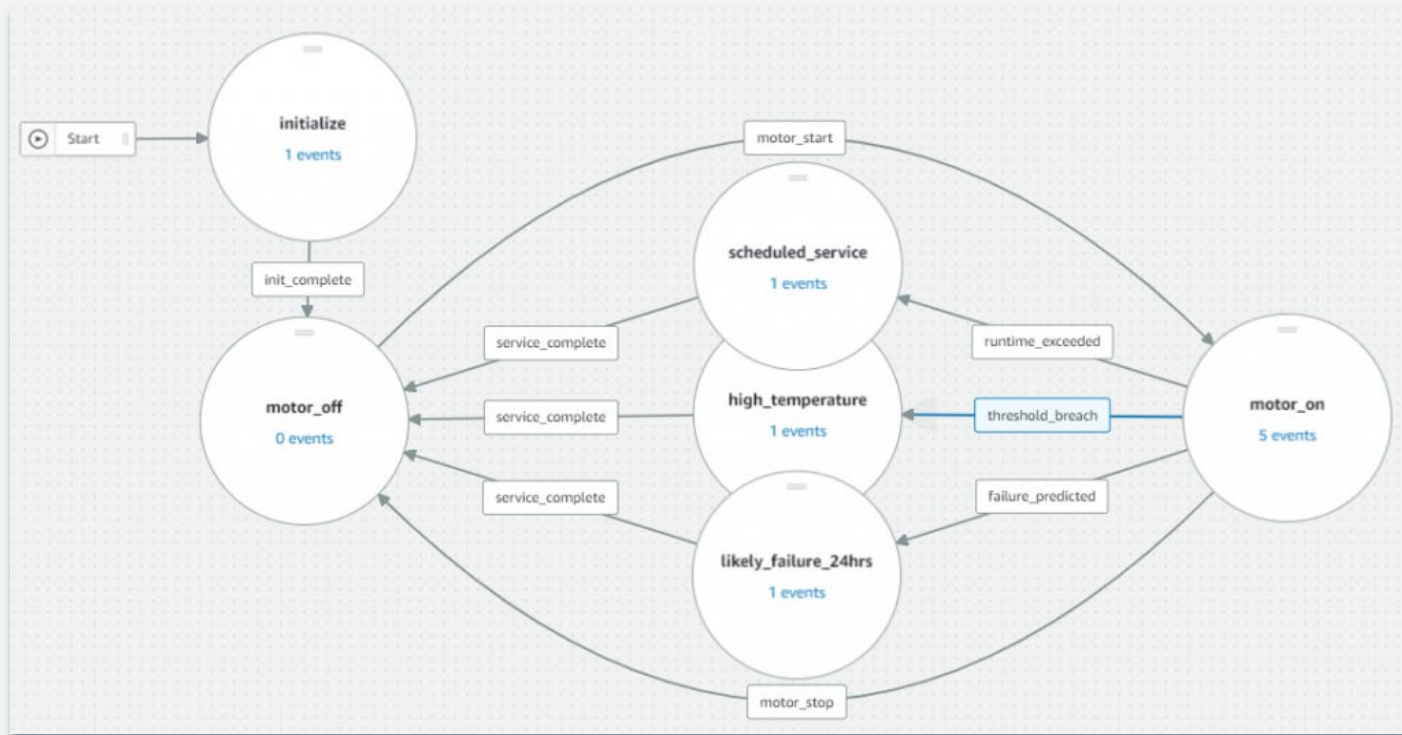
<https://aws.amazon.com/jp/blogs/iot/asset-maintenance-with-aws-iot-services-predict-and-respond-to-potential-failures-before-they-impact-your-business/>

- 工場設備の各種センサーの値を収集
- 一定時間異常な高温やポンプの異常を検出したら緊急メンテを通知
- 一定時間連続稼働したら計画停止



## 2) AWS IoT EventsとMLを組み合わせた設備の管理

<https://aws.amazon.com/jp/blogs/iot/asset-maintenance-with-aws-iot-services-predict-and-respond-to-potential-failures-before-they-impact-your-business/>

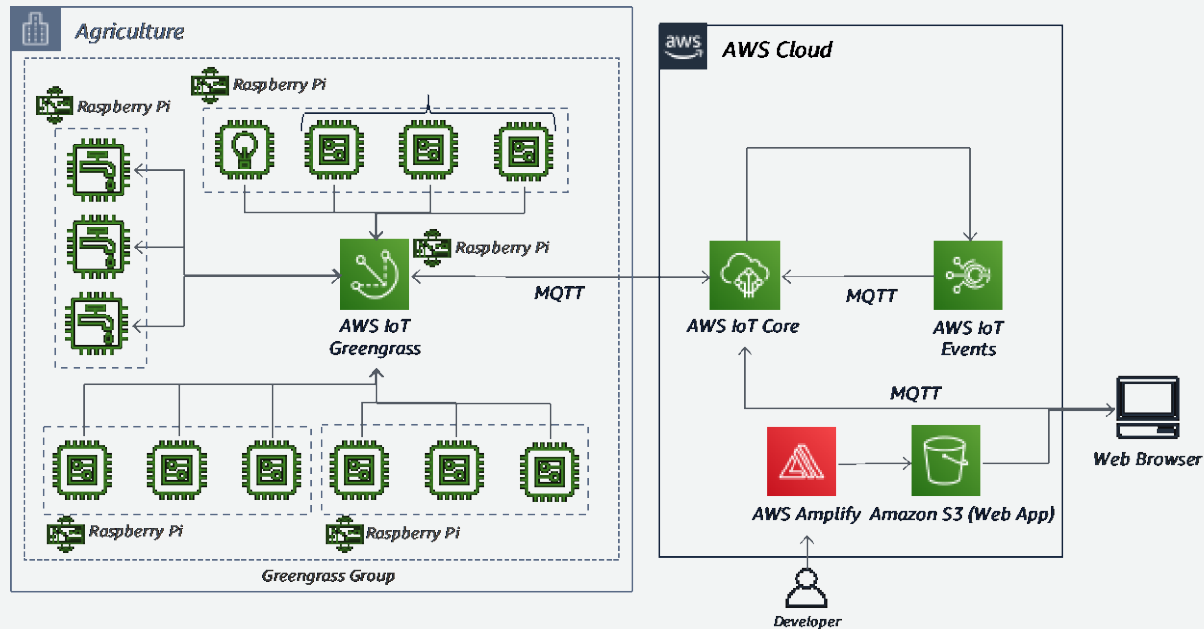




### 3) 畑の自動水やり

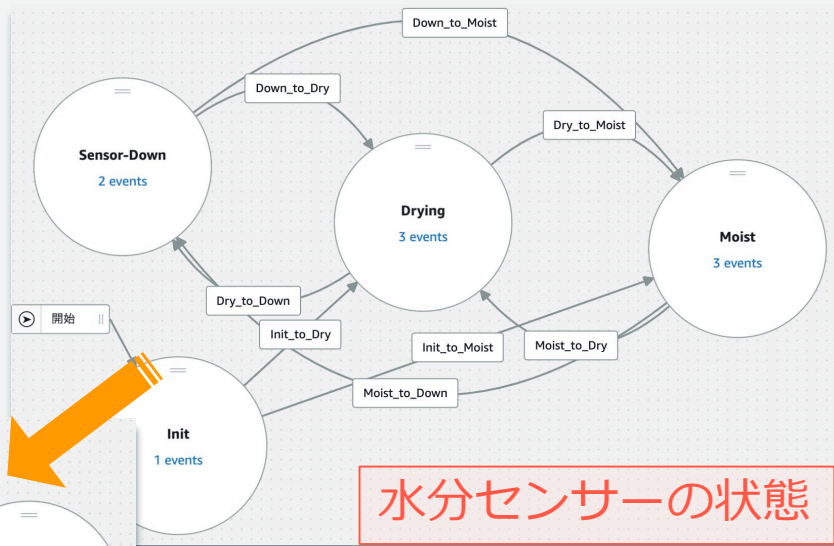
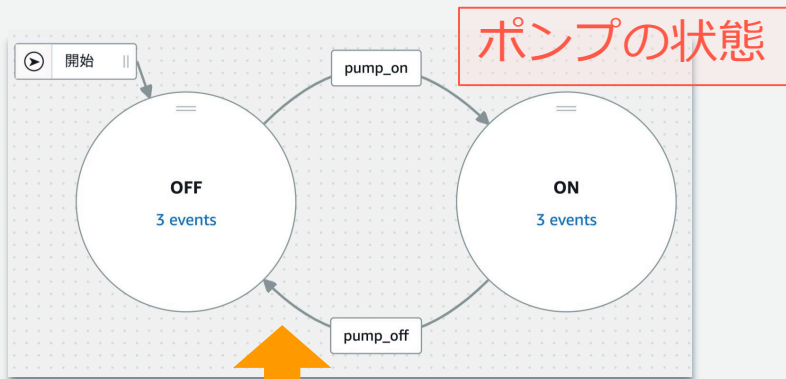
<https://github.com/aws-samples/aws-builders-fair-projects/tree/master/reinvent-2019/fully-automated-farm>

- 畑の自動水やり機では、水分量の減少を検出するとポンプを起動する
- 複数の探索器を連携



### 3) 畑の自動水やり

<https://github.com/aws-samples/aws-builders-fair-projects/tree/master/reinvent-2019/fully-automated-farm>



# AWS IoT Eventsの始め方

# テンプレートを使ってみよう



AWS IoT イベント

## 開始方法

開始するには、デバイスの状態を表す探知器モデルを作成します

[📄 探知器モデルのインポート](#)

### その他のリソース

[ドキュメント](#)

[API リファレンス](#)

[よくある質問](#)

[< 戻る](#)



### テンプレート

#### センサーとアプリケーションによるイベント検出

このディテクターモデルでは、デバイスセンサーとソフトウェアアプリケーションの両方からの入力を使用します。デバイス入力では GPS に位置と deviceId を送信します。ソフトウェアアプリケーションでは、deviceId、userId を使用してユーザー入力を行います。これらの入力の組み合わせを使用して、状態の変化を判断し、MQTT アクションを起動します。

ディテクターモデルテンプレート

開始

#### デバイスのハートビート

このディテクターモデルテンプレートは、通常/オフラインの2つのステートを定義します。入力が受信されるとタイマーが開始されます。別の入力を受信する前にタイマーが期限切れになると、オフライン状態に入り、SNS 通知を送信します。

ディテクターモデルテンプレート

開始

#### 単純なアラーム

このテンプレートは、ノーマル、アラーム、およびスヌーズの3つの状態がある単純なアラーム管理システムをモデル化したものです。入力のためのしきい値の設定を可能にし、しきい値に違反したときにアラームの状態を切り替えます。

ディテクターモデルテンプレート

開始

#### アラームライフサイクル管理 - ISA 18.2 規格

このテンプレートは、ISA 18.2 規格に基づいたアラームシステムをモデル化したものであり、異常なプロセス条件や機器の故障のオペレータへの通知を可能にし、あらゆる応答メカニズムをサポートします。

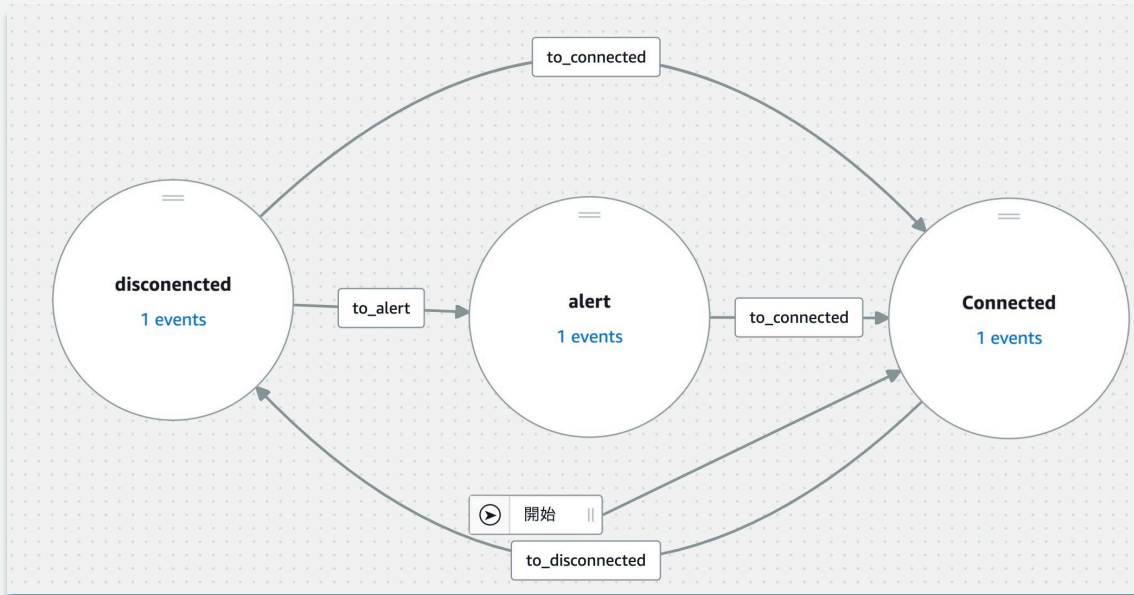
ディテクターモデルテンプレート

開始

# AWS IoT Eventsワークショップを試してみよう

<https://aws-iot-events-for-beginners.workshop.aws/>

- AWS IoT Coreのライフサイクルイベントを利用し、接続のステータスが一定時間disconnectedとなる場合に通知する仕組みを体験



# 探知器モデルの開発時のベストプラクティス

- AWS IoT EventsのCloudWatch ログを有効にしましょう
- 定期的にパブリッシュ(発行)を行い、探索器を保存しましょう
- 一定期間探索器を利用しない(データも送らない)場合は、モデルとデータを保存しましょう
  - 利用が無い場合は削除の30日前に通知をし、その後削除されます

<https://docs.aws.amazon.com/iotevents/latest/developerguide/best-practices.html>

# 探知器モデルのエクスポートとインポート

**AWS IoT イベント**

## 開始方法

開始するには、デバイスの状態を表す探知器モデルを作成します

**探知器モデルのインポート**

### 探知器モデルを作成する

- 作成する**  
新しいディテクターモデルを作成
- テンプレート**  
業界固有のテンプレートを選択する
- デモ**  
入力でサンプルディテクターモデルについて調べる

新しく作成する

IoT イベント > 探知器モデル > device\_connected\_status

## device\_connected\_status

削除 **エクスポート** データを送信する ▼ 編集

### 一般設定

説明	発行日	キー	バージョン
	Wed Jul 15 2020	clientId	2

評価方法

# 料金について

2020/9/16時点 東京リージョンの金額

料金の考え方は、メッセージの評価回数が基本

外部アクションの実行も、評価回数としてカウントする。1つのメッセージ評価に対して、2つのアクション実行は無料

- 最初の1億回のメッセージ評価
  - 100万回のメッセージ評価あたり 16.50USD
- 次の4億回のメッセージ評価
  - 100万回のメッセージ評価あたり 11.00USD
- 次の45億回のメッセージ評価
  - 100万回のメッセージ評価あたり 5.50USD
- 50億回以上のメッセージ評価
  - 100万回のメッセージ評価あたり 3.30USD

<https://aws.amazon.com/jp/iot-events/pricing/>



# まとめ

# まとめ

- AWS IoT Eventsを利用することで複雑な条件を組み合わせながら、デバイスやシステムの状態を管理する仕組みをコードレスで作成できる
- 複数のサービスからの入力を受付、AWSの他のサービスに対してアクションを起こすことができる
- マネージドサービスなので、デバイスが増えてきてもスケールする

# Q&A

お答えできなかったご質問については

AWS Japan Blog 「<https://aws.amazon.com/jp/blogs/news/>」にて

後日掲載します。

# AWS の日本語資料の場所「AWS 資料」で検索



The screenshot shows the AWS Japanese website header with the logo, navigation links for '日本語', 'アカウント', and 'サポート', and a 'サインイン' button. The main content area features the title 'AWS クラウドサービス活用資料集トップ' and a paragraph of introductory text. Below the text are four buttons: 'AWS Webinar お申込', 'AWS 初心者向け', '業種・ソリューション別資料', and 'サービス別資料'.

aws

日本語 日本担当チームへお問い合わせ サポート アカウント

コンソールにサインイン

製品 ソリューション 料金 ドキュメント 学習 パートナー AWS Marketplace その他

## AWS クラウドサービス活用資料集トップ

アマゾン ウェブ サービス (AWS) は安全なクラウドサービスプラットフォームで、ビジネスのスケールと成長をサポートする処理能力、データベースストレージ、およびその他多種多様な機能を提供します。お客様は必要なサービスを選択し、必要な分だけご利用いただけます。それらを活用するために役立つ日本語資料、動画コンテンツを多数ご提供しております。(本サイトは主に、AWS Webinar で使用した資料およびオンデマンドセミナー情報を掲載しています。)

AWS Webinar お申込 »

AWS 初心者向け »

業種・ソリューション別資料 »

サービス別資料 »

<https://amzn.to/JPArchive>

# AWS Well-Architected 個別技術相談会

毎週“W-A個別技術相談会”を実施中

- AWSのソリューションアーキテクト(SA)に  
対策などを相談することも可能

- **申込みはイベント告知サイトから**

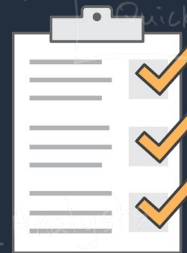
(<https://aws.amazon.com/jp/about-aws/events/>)

AWS イベント

で[検索]



AWS Well-Architected



# ご視聴ありがとうございました

AWS 公式 Webinar

<https://amzn.to/JPWebinar>



過去資料

<https://amzn.to/JPArchive>

