



2022年10月04日

## AWS IoT CoreではじめるIoTデータ蓄積



partner  
network



# 自己紹介

---

## □加藤 悠太 (Yuta Kato)

### □株式会社スタイルズ

- ▶ AWSを主としたIoTサービスの設計・構築担当

### □好きなAWSサービス

- ▶ IoT Events
- ▶ Step Functions



# 本日、お話する内容

---

## □対象

- ▶ IoTデータをクラウドに蓄積する方法について知りたい方
- ▶ AWSについて基本的な知識を持っている方

## □ゴール

- ▶ AWS IoT Coreを用いたデータ蓄積の方法を知る

# AWS IoT Coreとは

# IoTとは

---



- ▶ IoTはInternet of Things（モノのインターネット）の略
- ▶ 様々なモノがインターネットに繋がること、インターネットに繋がる様々なモノを指す
- ▶ モノがインターネットに繋がることで、今まで埋もれていたデータを利活用することができる

# AWS IoTサービスについて

- ▶ AWSにはIoTに活用できるサービスが多数用意されている
- ▶ 各サービスは役割で大きく3つに分類できる

## Data services



Amazon Kinesis  
Video Streams



AWS IoT  
Analytics



AWS IoT Events



AWS IoT  
FleetWise



AWS IoT  
SiteWise



AWS IoT  
TwinMaker

## Control services



AWS IoT Core



AWS IoT Device  
Advisor



AWS IoT Device  
Defender



AWS IoT Device  
Management

## Device software



AWS IoT Device  
SDKs



AWS IoT Device  
Tester



AWS IoT  
ExpressLink



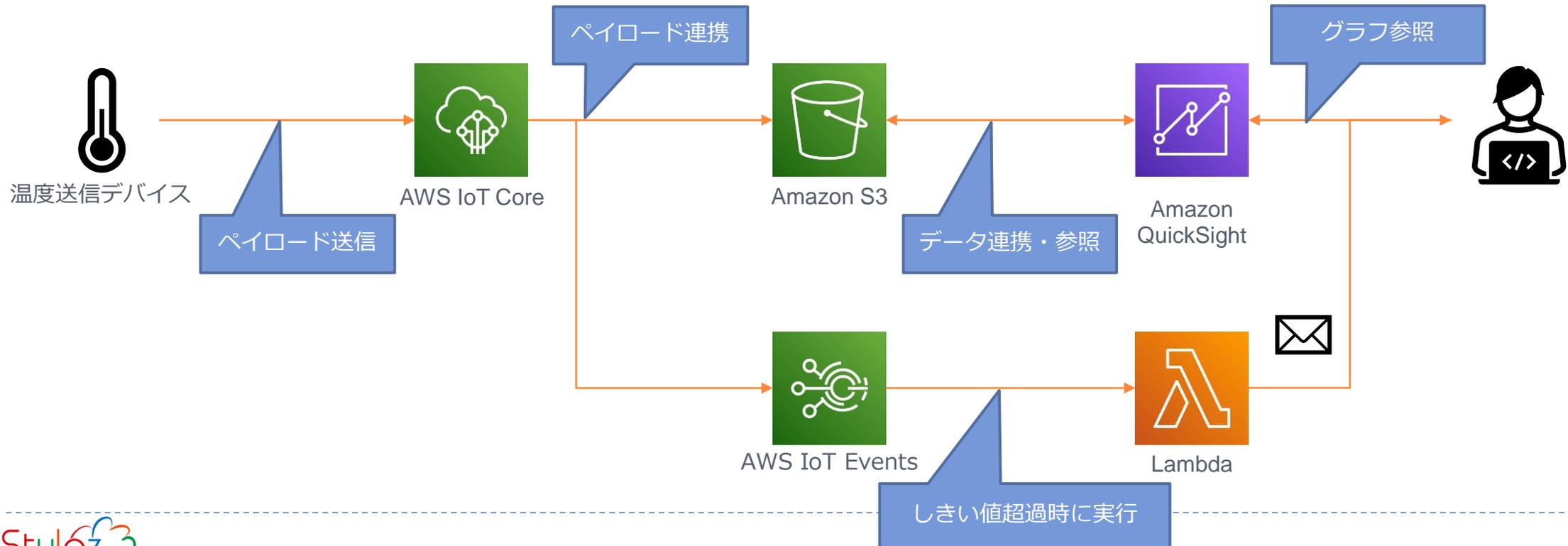
AWS IoT  
Greengrass



FreeRTOS

# AWS IoT Coreとは

- ▶ AWS IoT CoreはIoTデバイスを各種AWSサービスに接続するためのサービス
- ▶ 活用ケース①：ペイロードをデータストアに蓄積して可視化する
- ▶ 活用ケース②：ペイロードの値がしきい値を超えたら通知する



# AWS IoT Coreの機能

---

具体的には以下のような機能が用意されている（抜粋）

- ▶ デバイスゲートウェイ  
IoTデバイスがAWSに安全に接続するための機能
- ▶ メッセージブローカー  
IoTデバイスとAWSがメッセージをやり取りするための機能
- ▶ ルールエンジン  
受信したペイロードを他のサービスへと受け渡す機能
- ▶ デバイスプロビジョニング
- ▶ デバイスシャドウ

# AWS IoT Coreへの接続方法

# AWS IoT Coreへの接続に必要な情報（MQTT）

AWS IoT CoreとIoTデバイスを接続するまで



# モノ (Thing) の登録

- ▶ IoT Coreに接続するデバイスをモノ (Thing) として登録する
- ▶ 登録はマネジメントコンソールから簡単に行うことができる

AWS IoT > 管理 > モノ > モノを作成 > 1つのモノを作成

ステップ1  
モノのプロパティを指定

ステップ2 - オプション  
デバイス証明書を設定

ステップ3 - オプション  
証明書にポリシーをアタッチ

## モノのプロパティを指定 情報

モノのリソースは、AWS IoT の物理デバイスまたは論理エンティティのデジタル表現です。デバイスまたはエンティティは、Device Shadow、イベント、ジョブ、デバイス管理機能などの AWS IoT 機能を使用するために、レジストリにモノのオブジェクトを必要とします。

### モノのプロパティ

モノの名前

文字、数字、ハイフン、コロン、またはアンダースコアのみを含む一意の名前を入力します。モノの名前にスペースを含めることはできません。

### 追加設定

これらの設定を使用して、モノの整理、管理、検索に役立つ詳細を追加できます。

- ▶ モノのタイプ - オプション
- ▶ 検索可能なモノの属性 - オプション
- ▶ モノのグループ - オプション
- ▶ 請求グループ - オプション

# デバイス証明書の発行

- ▶ 登録したモノ（Thing）に対してデバイス証明書、秘密鍵、公開鍵、を発行する
- ▶ モノの登録時に、マネジメントコンソールから自動生成することができる

ステップ1  
モノのプロパティを指定

ステップ2 - オプション  
デバイス証明書を設定

ステップ3 - オプション  
証明書にポリシーをアタッチ

## デバイス証明書を設定 - オプション 情報

デバイスには、AWS IoT に接続するために証明書が必要です。今すぐデバイスの証明書を登録する方法を選択するか、後でデバイス用の証明書を作成して登録できます。適切なポリシーを含むアクティブな証明書がないと、デバイスから AWS IoT に接続することはできません。

### デバイス証明書

- 新しい証明書を自動生成 (推奨)  
AWS IoT の認証機関を使用して、証明書、パブリックキー、およびプライベートキーを生成します。
- 自分の証明書を使用  
独自の認証機関によって署名された証明書を使用します。
- CSR をアップロード  
CA を登録し、1 つまたは複数のデバイスに独自の証明書を使用します。
- 今回の証明書の作成をスキップ  
このモノの証明書を作成し、後で証明書にポリシーをアタッチできます。

キャンセル 戻る 次へ

# デバイス証明書の発行

## 証明書とキーをダウンロード



AWS に接続できるように、証明書とインストールするキーファイルをデバイスにダウンロードします。

### デバイス証明書

証明書は今すぐアクティブ化することも、後でアクティブ化することもできます。デバイスが AWS IoT に接続するためには、証明書がアクティブである必要があります

デバイス証明書

██████████.pem.crt

証明書を非アクティブ化

ダウンロード

### キーファイル

キーファイルはこの証明書に固有であり、このページを離れるとダウンロードできません。今すぐダウンロードして、安全な場所に保存してください。

 この証明書のキーファイルをダウンロードできるのは、この時点のみです。

パブリックキーファイル

██████████-public.pem.key

ダウンロード

プライベートキーファイル

██████████-private.pem.key

ダウンロード

## ルート CA 証明書

使用しているデータエンドポイントと暗号スイートのタイプに対応するルート CA 証明書ファイルをダウンロードします。ルート CA 証明書は後でダウンロードすることもできます。

Amazon 信頼サービスエンドポイント

RSA 2048 ビットキー: Amazon ルート CA 1

ダウンロード

Amazon 信頼サービスエンドポイント

ECC 256 ビットキー: Amazon ルート CA 3

ダウンロード

ここで必要なルート CA 証明書が表示されない場合、AWS IoT では追加のルート CA 証明書がサポートされます。これらのルート CA 証明書などは、デベロッパーガイドで入手できます。 [詳細はこちら](#)

# IoTポリシーとは

- ▶ IoTポリシーは接続できるクライアントの制限や、Publish/Subscribeできるトピックの制限などを行うことができる

バージョン	ターゲット	非標準	タグ
アクティブなバージョン: 4 <a href="#">情報</a>			<input type="button" value="ビルダー"/> <input type="button" value="JSON"/>
ポリシー効果	ポリシーアクション	ポリシーリソース	
Allow	iot:Connect	arn:aws:iot:ap-northeast-1:██████████:client/\${iot:Connection.Thing.ThingName}	
Allow	iot:Publish	arn:aws:iot:ap-northeast-1:██████████:topic/some/*	

# IoTポリシーのアタッチ

- ▶ IoTポリシーが既に用意されている場合は、マネジメントコンソールでモノ（Thing）を登録する手順の中で簡単に設定することができる



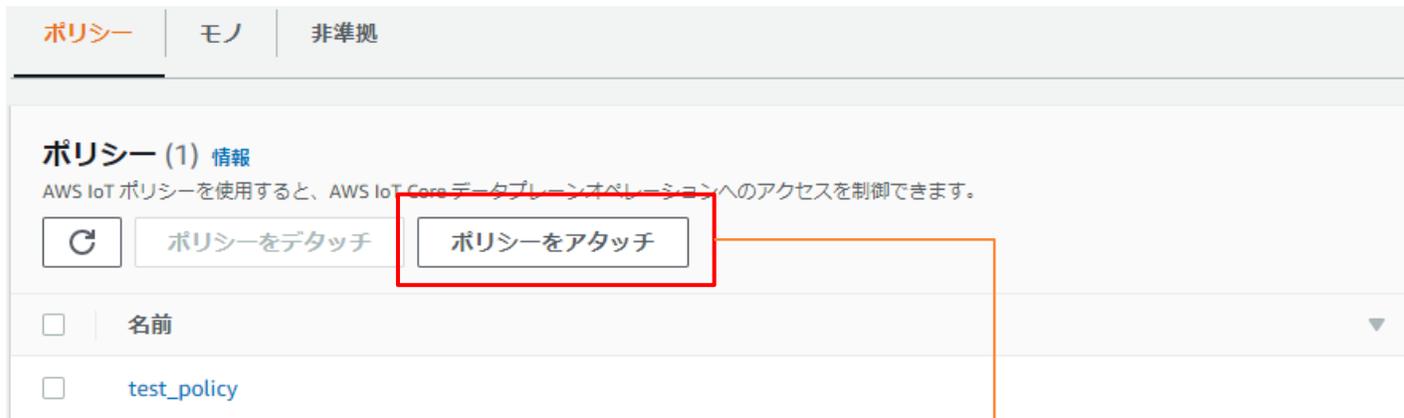
The screenshot shows the AWS IoT console interface for attaching a policy to a certificate. The breadcrumb trail is 'AWS IoT > 管理 > モノ > モノを作成 > 1つのモノを作成'. The current step is 'ステップ1: モノのプロパティを指定', with sub-steps 'ステップ2-オプション: デバイス証明書を設定' and 'ステップ3-オプション: 証明書にポリシーをアタッチ'. The main heading is '証明書にポリシーをアタッチ - オプション 情報'. A note states: 'AWS IoT ポリシーは、AWS IoT リソースへのアクセスを許可または拒否します。デバイス証明書にポリシーをアタッチすると、このデバイスからアクセスできるようになります。' Below this is a section titled 'ポリシー (1/23)' with a refresh button and a 'ポリシーを作成' button. A search bar contains 'フィルターポリシー'. A table lists policies, with 'test\_policy' selected.

名前
<input checked="" type="checkbox"/> test_policy

- ▶ 権限に制限を加えない場合でも、すべてを許可するIoTポリシーを作成しなければ通信できないので注意

# IoTポリシーのアタッチ

- ▶ モノ（Thing）の登録後であっても、デバイス証明書の詳細画面からIoTポリシーのアタッチ/デタッチは簡単に行うことができる



# デバイスデータエンドポイントとトピックの確認

- ▶ デバイスデータエンドポイントはマネジメントコンソールの設定から確認できる

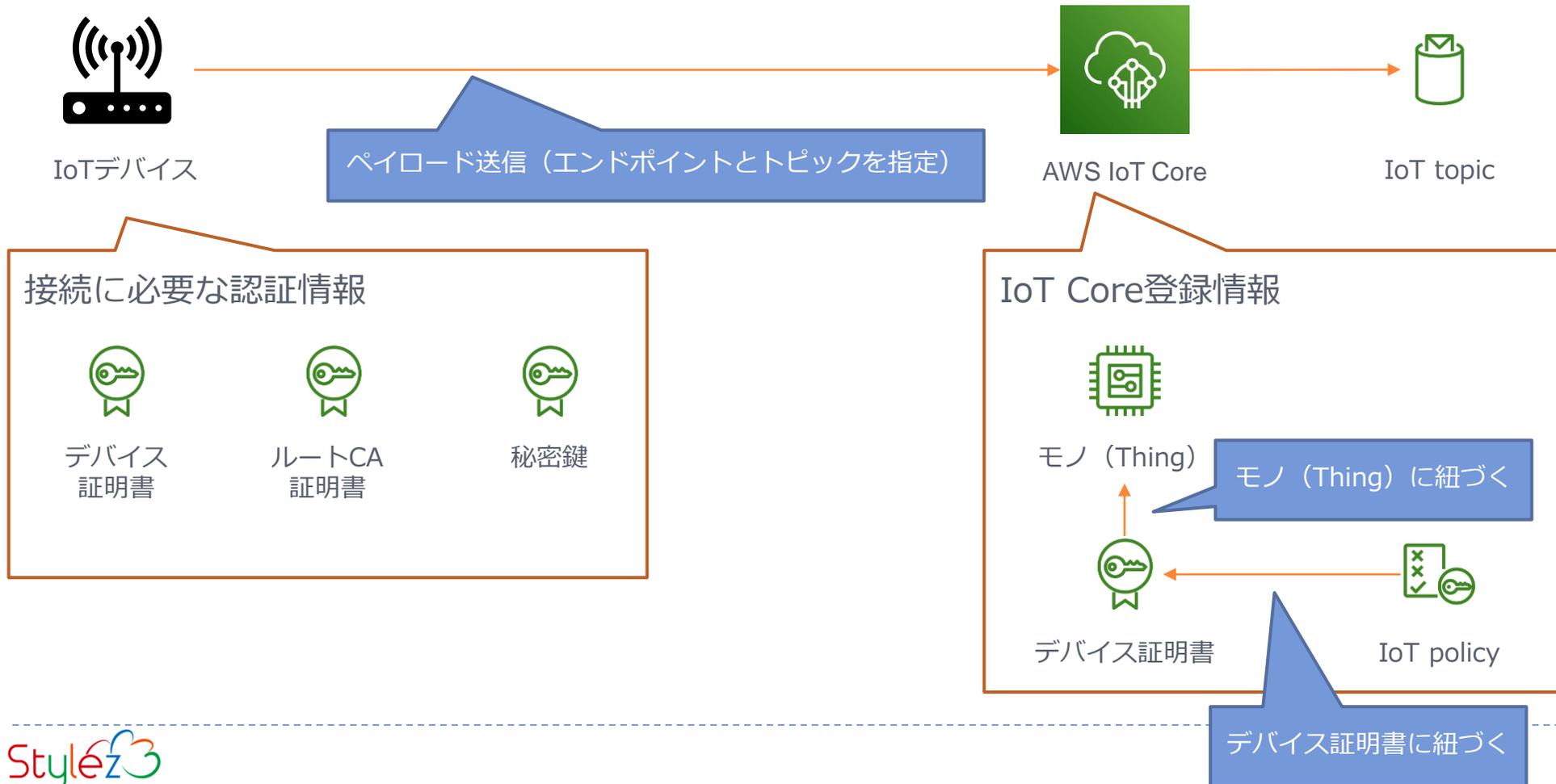


The screenshot shows the 'Device Data Endpoint Information' page in the AWS IoT console. The title is 'デバイスデータエンドポイント 情報' (Device Data Endpoint Information) with a refresh icon. Below the title, it states: 'デバイスは、アカウントのデバイスデータエンドポイントを使用して AWS に接続できます。' (Devices can connect to AWS using the device data endpoint of the account). A paragraph follows: '各モノには、このエンドポイントで使用可能な REST API があります。MQTT クライアントと [AWS IoT デバイス SDK](#) もこのエンドポイントを使用します。' (Each device has a REST API available at this endpoint. MQTT clients and the [AWS IoT Device SDK](#) also use this endpoint). At the bottom, there is a section labeled 'エンドポイント' (Endpoint) with a copy icon and the text 'ap-northeast-1.amazonaws.com', which is highlighted with a red box.

- ▶ ペイロード送信時は実行したいIoT rule（後述）がsubscribeしているトピックを指定する

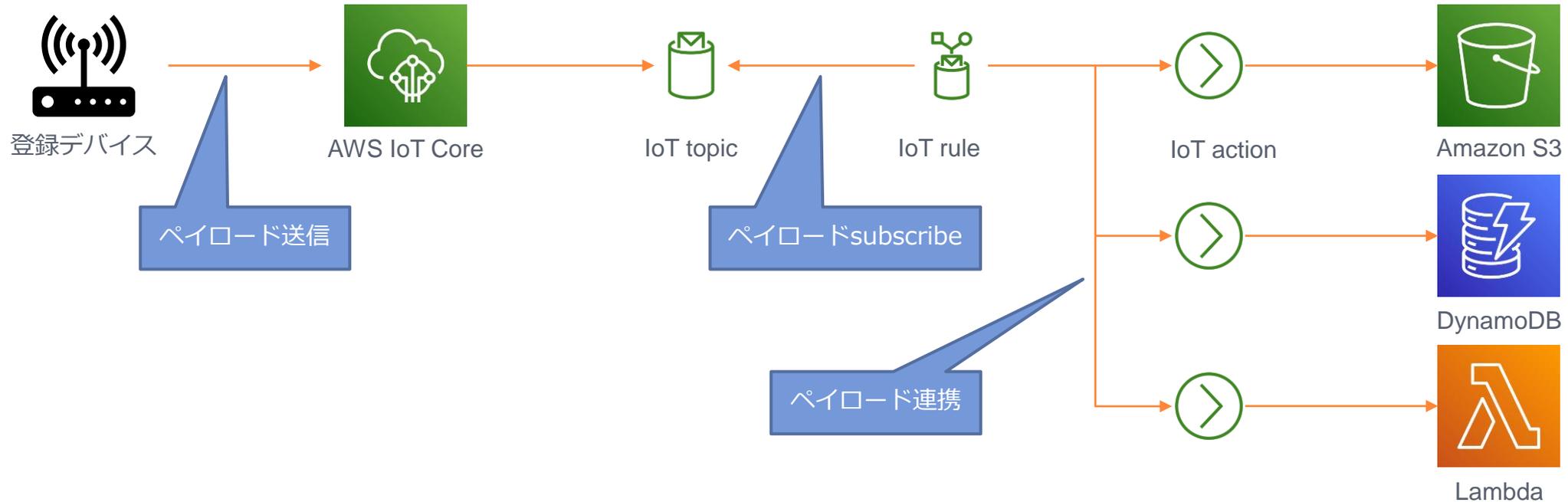
# AWS IoT Coreとの接続例

IoTデバイスからAWS IoT Coreまでペイロードを送るまで



## IoT ruleアクションによるデータ蓄積

# IoT ruleとは



- ▶ 特定のトピックで受信したペイロードのアクションを管理する
- ▶ SQLステートメント（後述）でアクションに渡すペイロードを加工できる

# データストアとの連携方法①

## □ S3アクション

以下を指定することで、受信したペイロードをS3に格納することができる

- ▶ バケット名、キー、既定ACL（オプション）、IAMロール
- ▶ キーには置換テンプレートを利用できる

**S3 bucket** ×

Amazon S3 バケットにメッセージを保存

バケット名 test-bucket-██████████	キー \${software}/\${timestamp()}.log	既定 ACL private
IAM ロール arn:aws:iam::██████████:role/sevice-role/S3UploadRole <a href="#">🔗</a>		

# データストアとの連携方法②

## □ DynamoDBアクション

**DynamoDB** ×  
DynamoDB テーブルにメッセージを挿入

テーブル名 TestTable	パーティションキー timestamp	パーティションキーのタイプ NUMBER
パーティションキーの値 \${timestamp()}	ソートキー - オプション -	レンジキーのタイプ -
レンジキーの値 -	この列にメッセージデータを書き込む - オプション payload	オペレーション - オプション INSERT

IAM ロール  
arn:aws:iam::[redacted]:role/service-role/RuleAction-DynamoDBInsert [🔗](#)

閉じる

テーブル名、PK、SKの各種情報、IAMロールの設定で、受信したペイロードをDynamoDBのテーブルに書き込むことができる

## データストアとの連携方法②

### □ DynamoDBアクション

- ▶ アクションに渡した値を単一attributeにまとめて登録する

アクションに渡されたペイロード

```
{  
  "temperature": 27.2,  
  "uuid": "9931a265-80f6-b85e-aa94-3f54aa0df28d",  
  "timestamp": "1664848800"  
}
```

DynamoDBアクションによる登録

timestamp ※PK	payload
1664848800	{"temperature": 27.2, "uuid": "9931a265-80f6-b85e-aa94-3f54aa0df28d", "timestamp": "1664848800"}

## データストアとの連携方法③

### □ DynamoDBv2アクション

テーブルとIAMロールの指定のみで、受信したペイロードをDynamoDBのテーブルに書き込むことができる

**DynamoDBv2** ×

DynamoDB テーブル (DynamoDBv2) の複数列にメッセージを分割する

テーブル名	IAM ロール
TestTable	<a href="#">arn:aws:iam::[redacted]:role/service-role/RuleAction-DynamoDBv2</a> 

閉じる

- ▶ ペイロード中にPK、SKとなるkey-valueは必ず含まれている必要がある

## データストアとの連携方法③

### □ DynamoDBv2アクション

- ▶ アクションに渡した値を各attributeに分けて登録する

```
アクションに渡されたペイロード  
{  
  "temperature": 27.2,  
  "uuid": "9931a265-80f6-b85e-aa94-3f54aa0df28d",  
  "timestamp": "1664848800"  
}
```

DynamoDBv2アクションによる登録

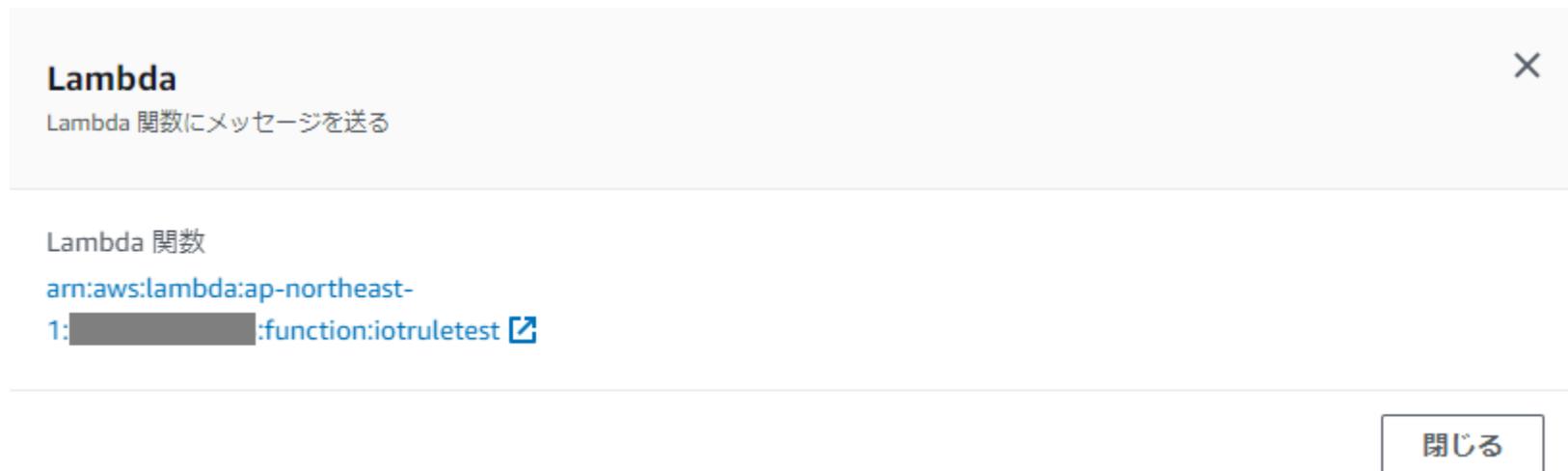
timestamp	※PK	temperature	uuid
1664848800		27.2	9931a265-80f6-b85e-aa94-3f54aa0df28d

## データストアとの連携方法④

---

### □ Lambdaアクション

受信したペイロードにより細かい処理を加えたいときはLambdaに連携する  
Lambda関数を指定するのみで良い



# SQLステートメントについて

---

## SQL ステートメント

SQL ステートメント	SQL のバージョン
<code>SELECT temperature, newuuid() AS uuid, timestamp() AS timestamp FROM 'some/test' WHERE temperature &lt; 30</code>	2016-03-23

- ▶ アクションに渡すペイロードをSELECT句で簡易に加工できる
- ▶ 組み込み関数も複数用意されている
- ▶ SubscribeするトピックはSQLステートメントのFROM句で指定する
- ▶ WHERE句でアクションを実行する条件を指定できる

# その他連携できるアクション

## DynamoDBv2

DynamoDB テーブル (DynamoDBv2) の複数列にメッセージを分割する

## Lambda

Lambda 関数にメッセージを送る

## Simple Notification Service (SNS)

SNS プッシュ通知としてメッセージを送信します

## Simple Queue Service (SQS)

SQS キューにメッセージを送信する

## Apache Kafka Cluster

VPC 内の Apache Kafka にメッセージを送信します

## Kinesis Stream

VPC に送信されるメッセージは、追加のアクションで計測されます。

## Republish to AWS IoT topic

AWS IoT のトピックにメッセージを再パブリッシュする

## S3 bucket

Amazon S3 バケットにメッセージを保存

## Kinesis Firehose stream

Amazon Kinesis Firehose ストリームにメッセージを送信

## CloudWatch metric

CloudWatch メトリクスにメッセージデータを送信

## CloudWatch alarm

CloudWatch アラームの状態を変更

## CloudWatch logs

CloudWatch Logs にメッセージデータを送信

## Amazon OpenSearch Service (Amazon Elasticsearch Service の後継サービス)

Amazon OpenSearch Service (Amazon Elasticsearch Service の後継サービス) にメッセージを送信します。

## Salesforce IoT

Salesforce IoT 入力ストリームにメッセージを送信

## IoT Analytics

IoT Analytics にメッセージを送る

## IoT Events

IoT Events 入力にメッセージを送信

## AWS IoT SiteWise のアセットプロパティにメッセージデータを送信

このアクションは、AWS IoT SiteWise のアセットプロパティにメッセージを送信します。

## Step Functions

Step Functions ステートマシンの実行を開始

## HTTPS endpoint

ダウンストリーム HTTPS エンドポイントにメッセージを送信します

## Timestream table

Timestream テーブルにメッセージを書き込む

## Basic Ingestによるコストカット

# IoT Coreのコスト

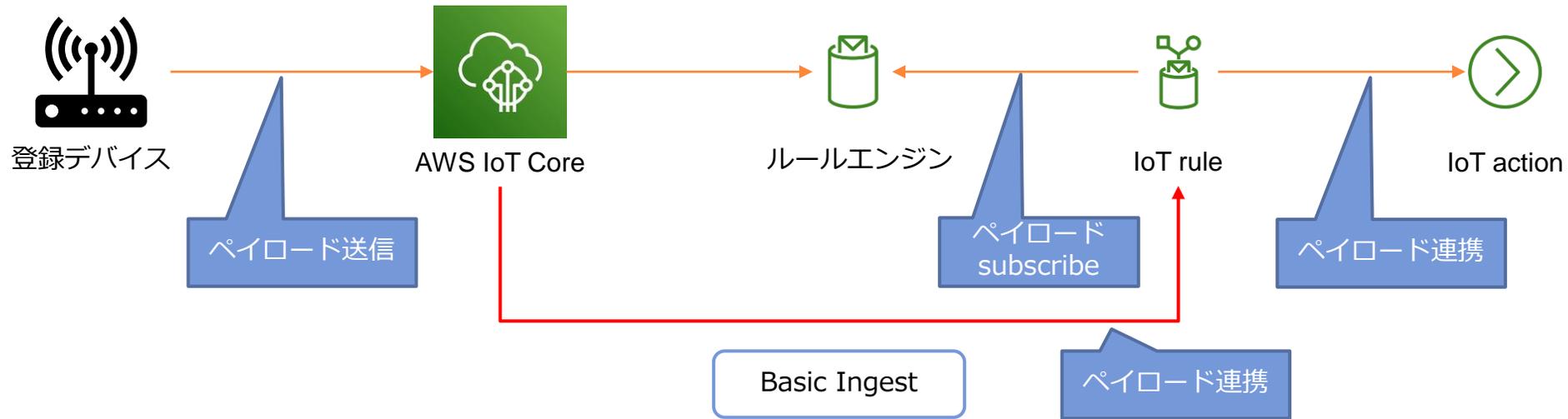
- ▶ IoT Coreとの接続時間、メッセージング件数、ルールとアクションの実行件数による従量課金

例) 100台が5分に1回ペイロードを送信し、2つのアクションを持つ1つのルールを毎回実行する。デバイスは常時接続とする

※以下はアクションによるデータストアへの書き込み料金等を含まない

コスト項目	料金
接続コスト	0.42 USD/月
メッセージングコスト	1.04 USD/月
ルールエンジンのコスト (ルール実行+アクション実行)	0.47 USD/月

# BasicIngestとは



- ▶ ルールエンジンをバイパスし、直接IoT ruleにペイロードを送ることでメッセージングコストが発生しなくなる
- ▶ 特定のトピックに送ったペイロードを複数ルールからsubscribeする、といったことはできなくなる

# BasicIngestによるコストカット

- ▶ Basic Ingestの利用には特別な設定は不要
- ▶ 「`$aws/rules/{実行したいIoT rule名}/{当該ルールがsubscribeしているトピック}`」に対してペイロードを送信する

例) 100台が5分に1回ペイロードを送信し、2つのアクションを持つ1つのルールを毎回実行する。デバイスは常時接続とする

※以下はアクションによるデータストアへの書き込み料金等を含まない

コスト項目	料金
接続コスト	0.42 USD/月
ルールエンジンのコスト (ルール実行+アクション実行)	0.47 USD/月

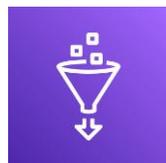
# AWS IoT Coreでデータを蓄積するメリット

# 蓄積したデータの活用

## □AWSサービスとの連携

蓄積したデータを他のAWSサービスから参照して可視化や分析が可能

分析サービス



AWS Glue



Amazon Athena



Amazon SageMaker



Amazon Lookout for Metrics

可視化サービス



Amazon QuickSight



Amazon Managed Grafana

まとめ

## まとめ

---

- AWS IoT Coreの活用でクラウドで簡単にIoTデータ収集ができる
- IoT ruleを利用してAWSのサービスと簡単に連携できる
- 蓄積したデータはAWSサービスを利用して分析・可視化などに活用できる



実績豊富なエンジニア集団の技術と開発ツールで  
「開発期間/コスト削減」「品質向上」を実現

株式会社スタイルズ

<https://www.stylez.co.jp>

東京都千代田区神田小川町1-2 風雲堂ビル6階

Tel:03-5244-4111

オープンソースソフトウェア推進