

1. Lambda 関数の作成

lambda関数

lambda関数の作成 => lambdaに飛ぶ => [lambdaの関数を作成]

[一から作成]

Name:AWSIoTcore2Lambda

ランタイム : python3.7

```
import json, time, boto3

def lambda_handler(event, context):
    print(event["sample_time"])
    print(event["humidity"])
    print(event["barometer"])
    time.sleep(1)

    return "OK"
```

=> {Deploy}

2 AWS IOT COREでルール（アクション）の作成

[AWS Iot Core] => 管理 => メッセージのルーティング => ルール

[Create rule] (ルールの作成)

name : to_lambda

[SQL statement]

2016-03-23

SELECT *

FROM 'device/+/data'

ルールアクション

=> [Lambda](Lambda関数にメッセージを送る)

Lambda関数 : AWSIoTcore2Lambda (さっき作ったやつ)

Lambda関数のバージョン : \$LATEST

ステップ 2: SQL ステートメント		編集
SQL ステートメント		
SQL のバージョン 2016-03-23		
SQL クエリ SELECT * FROM 'device/+/data'		
ステップ 3: ルールアクション		編集
アクション		
Lambda Lambda 関数にメッセージを送る		
Lambda 関数 arn:aws:lambda:ap-northeast-1:980023311172:function:AWSIoTcore2Lambda 🔗	Lambda 関数のバージョン \$LATEST	

[作成]

AWS IoT > メッセージのルーティング > ルール

ルール (2) 情報

ルールを使用すると、モノは他のサービスとインタラクションできます。ルールは分析され、デバイスによって発行されたメッセージに基づいて特定のアクションが実行されます。


 有効化 無効化 編集 削除 **ルールを作成**


🔍 ルールを検索

<input type="checkbox"/>	名前 ▲	ステータス ▼	ルールのトピック ▼	作成日
<input type="checkbox"/>	to_lambda	✔ アクティブ	device/+ /data	April 01,
<input type="checkbox"/>	wx_data_ddb	✔ アクティブ	device/+ /data	March 29

LmbdaでAWS IOTがトリガーになっている

▼ 関数の概要 情報

 **AWS IoT**

 **AWSIoTcore2Lambda**
Layers (0)

+ トリガーを追加

3 Cloud Watchで確認

```
$ python3 testMQTT.py
```

Cloud Watch

ログイベント

下のフィルターバーを使用して、ログイベント内の用語、語句、値の検索や照合ができます。[フィルターパターンの詳細](#)

アクション ▼

メトリクスフィルターを作成

🔍 イベントをフィルター

クリア

1m

30m

1h

12h

カスタム

表示 ▼

▶

タイムスタンプ

メッセージ

▶

2023-04-01T21:53:07.277+09:00

現時点では古いイベントはありません。[再試行](#)

▶

2023-04-01T21:53:07.575+09:00

INIT_START Runtime Version: python:3.7.v23 Runtime Version ARN: arn:aw...

▶

2023-04-01T21:53:07.575+09:00

START RequestId: 80471093-2942-4135-9b09-099ef4a90e13 Version: \$LATEST

▶

2023-04-01T21:53:07.575+09:00

2023-04-01 21:53:06.900650

▶

2023-04-01T21:53:07.575+09:00

6

▶

2023-04-01T21:53:07.575+09:00

463

▶

2023-04-01T21:53:08.577+09:00

END RequestId: 80471093-2942-4135-9b09-099ef4a90e13

DynamoDB

Items returned (15)

Export to CSV

#	sample_time ⚡	device_id ⚡	wind_velocity ⚡	barometer ⚡	humidity ⚡	temperature
1	2023-04-01 21:53...	0	1	463	6	3
2	2023-04-01 21:53...	33	20	183	1	3
3	2023-04-01 21:53...	6	24	177	6	4