

## はじめての IoT ～AWS IoT Core ハンズオン～

アマゾン ウェブ サービス ジャパン

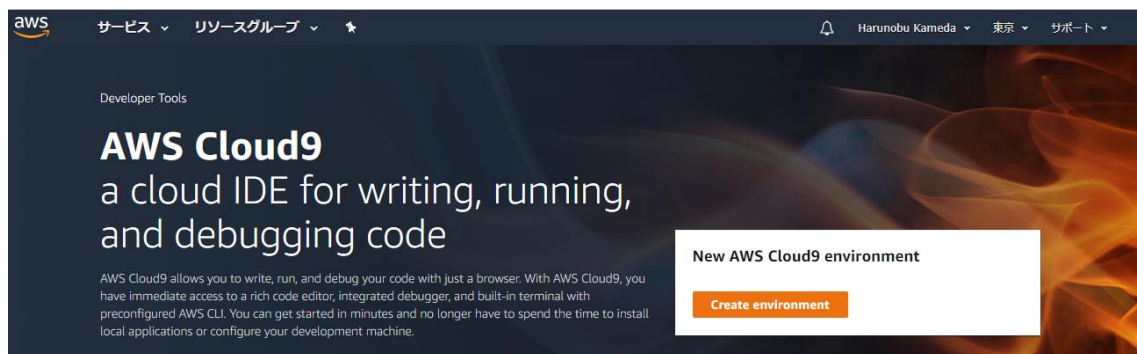
エバンジェリスト

亀田 治伸

### IoT ダミークライアントの作成

AWS IoT Core と通信（MQTT 及び HTTP）を行う IoT クライアントを構築します。このハンズオンでは、デバイスを用いず、AWS Cloud9 に AWS IoT SDK をインストールします。

- 1-1. AWS Cloud 9 の画面にアクセスします。（ブラウザの別タブで開くことをお勧めします）



- 1-2. 【Create Environment】を押します。

適当な名前を付けて、【Next Step】を押します。

Step 1  
Name environment

Step 2  
Configure settings

Step 3  
Review

## Name environment

### Environment name and description

**Name**  
The name needs to be unique per user. You can update it at any time in your environment settings.

Name

Limit: 60 characters

**Description - Optional**  
This will appear on your environment's card in your dashboard. You can update it at any time in your environment settings.

Write a short description for your environment

Limit: 200 characters

CancelNext step

aws

サービス

リソースグループ

AWS Cloud9 > Environments > Create environment

Step 1  
Name environment

Step 2  
Configure settings

Step 3  
Review

## Configure settings

### Environment settings

**Environment type** [Info](#)  
Choose between creating a new EC2 instance for your new environment or connecting directly to your server over SSH.

☒ **Create a new instance for environment (EC2)**  
Launch a new instance in this region to run your new environment.

☐ **Connect and run in remote server (SSH)**  
Display instructions to connect remotely over SSH and run your new environment.

**Instance type**

☒ **t2.micro (1 GiB RAM + 1 vCPU)**  
Free-tier eligible. Ideal for educational users and exploration.

☐ **t2.small (2 GiB RAM + 1 vCPU)**  
Recommended for small-sized web projects.

☐ **m4.large (8 GiB RAM + 2 vCPU)**  
Recommended for production and general-purpose development.

☐ **Other instance type**  
Select an instance type.  
t2.nano

**Platform**

☒ **Amazon Linux**

☐ **Ubuntu Server 18.04 LTS**


Cost-saving setting

1-3. 「Create a new instance for environment (EC2)」にチェックを入れる。


Cloud9 は専用環境を EC2 で起動する形式と、既存サーバへ SSH 経由でログインし環境を設定する形式があります。このハンズオンでは、デフォルトの EC2 形式を活用します。オ

ンプレミス環境のサーバなどでも設定が可能です。**OS は Amazon Linux でなく Ubuntu**  
**を選択してください。**


▼ Network settings

 **No default VPC**  
Your account has multiple VPCs but there are no defaults so we automatically selected one for you. You can change this if you want.

Network (VPC)  
Launch your EC2 instance into an existing Amazon Private Cloud (VPC) or create a new one.

vpc-aa1644cf ▼  [Create new VPC](#)

Subnet  
Select a range of IP addresses in your VPC to isolate EC2 resources from each other.

subnet-2a2d7d73 | Non-default in ap-northeast-1c ▼  [Create new subnet](#)

Cancel [Previous step](#) [Next step](#)

1-4. 任意の VPC を選択し、Public サブネットを選びます。不明な場合は、スタッフに聞いてください。AWS は初期状態でデフォルト VPC というものが存在しています。特にこだわりがない場合、デフォルト VPC 及びそのパブリックサブネットを指定してください。

Cost-saving settings

After 30 minutes (default)

IAM role

AWSServiceRoleForAWSCloud9 (generated)



**We recommend the following best practices for using your AWS Cloud9 environment**

- Use **source control and backup** your environment frequently. AWS Cloud9 does not perform automatic backups.
- Perform regular **updates of software** on your environment. AWS Cloud9 does not perform automatic updates on your behalf.
- **Turn on AWS CloudTrail in your AWS account** to track activity in your environment. [Learn more](#)
- Only share your environment with **trusted users**. Sharing your environment may put your AWS access credentials at risk. [Learn more](#)

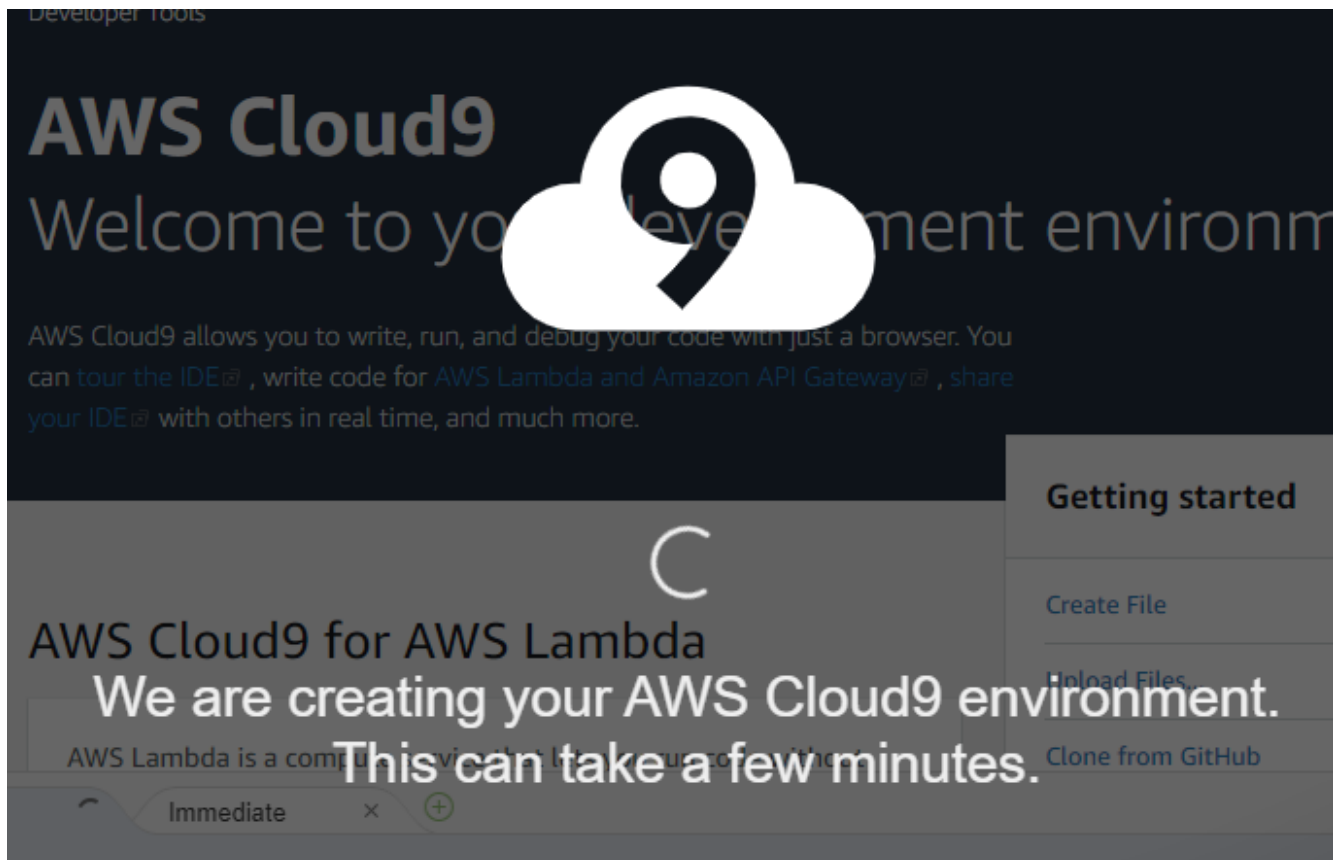
Cancel

Previous step

Create environment

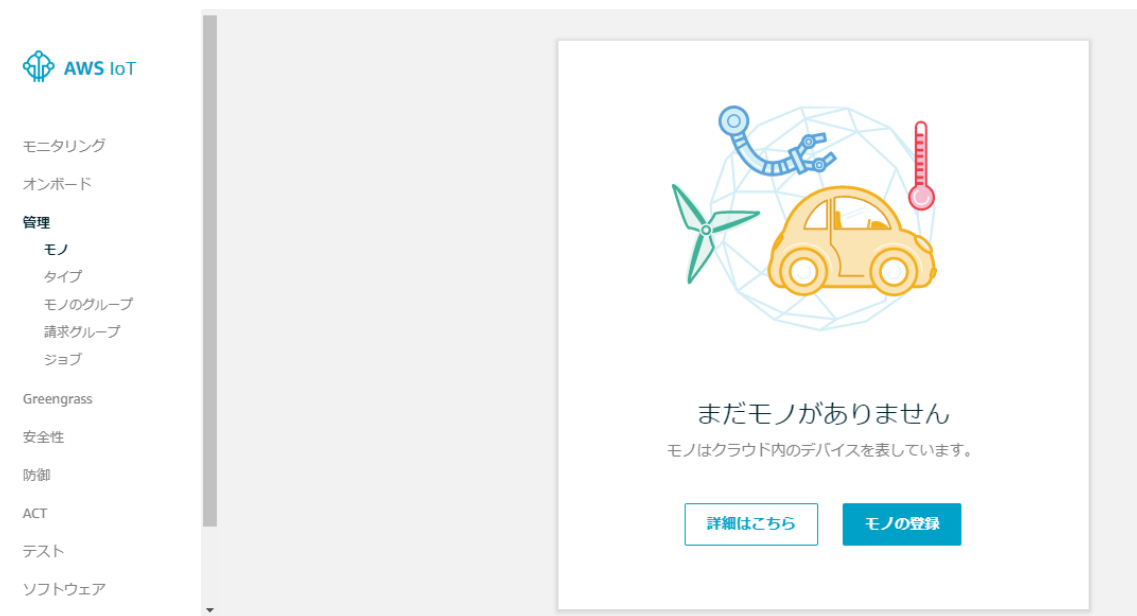
1-5. 「Next step」を押して出てきた確認画面で「Create environment」を押す

起動まで数分まちます。



## 2. AWS IoT クライアントの設定

### 2-1. AWS IoT のトップ画面へアクセスします



### 2-2. 画面左下の【設定】を押します

安全性

防御

ACT

テスト

ソフトウェア

設定

学習

2-3. 【エンドポイント】をコピーしておき、テキストファイルなどにペーストしておきます。IoT クライアントが通信を行う先の URI です。はじめて利用する場合、右上のボタンを操作し、【有効】と表示されるように設定してください。

### カスタムエンドポイント

有効

AWS IoT に接続することができるカスタムエンドポイントです。モノにはそれぞれ、このエンドポイントで利用できる REST API があります。これは、MQTT クライアントまたは AWS IoT 「[デバイス SDK](#)」を使用する際に挿入される重要なプロパティでもあります。

エンドポイントはプロビジョンされ、使用を開始できるようになりました。これで、トピックのパブリッシュとサブスクライブを開始できます。

エンドポイント

afhmd7pja59at-ats.iot.ap-northeast-1.amazonaws.com

2-4. 【安全性】 → 【Policy】 を選びます。

### 安全性

証明書

ポリシー

CA

ロールエイリアス

オーソライザー

2-5. 【ポリシーの作成】 をクリックします。



ポリシーはまだ作成されていません。

AWS IoT ポリシーは、AWS IoT リソース (その他のモノ、MQTT トピック、デバイス、Thing Shadow など) へのアクセス許可をモノに付与します。

[詳細はこちら](#)

ポリシーの作成

2-6. 適当な名前を付けます。ここで作成したポリシーは、AWS IoT Core と通信を行う

クライアントが持つべきセキュリティポリシー（AWS IoT Core の複数の機能と連携できる・できない等）になります。

## ポリシーの作成

ポリシーを作成して、認可アクションのセットを定義します。1 つ以上のリソース (モノ、トピック、トピックフィルター) のアクションを承認できます。IoT ポリシーの詳細については、「[AWS IoT ポリシーのドキュメントページ](#)」を参照してください。

名前

ステートメントを追加

ポリシー構文は、リソースで実行できるアクションの種類を定義します。

アドバンスドモード

アクション

カンマを使用してアクションを区切ってください (例: iot:Publish, iot:Subscribe)

リソース ARN

2-7. 以下の表示と同じ値を入力し、【作成】を押します。このハンズオンでは AWS IoT のすべての機能を使えるポリシーを作成します。(AWS のその他リソースを操作できる権限ではないことに注意してください) 【iot:\*】【\*】

ステートメントを追加

ポリシー構文は、リソースで実行できるアクションの種類を定義します。

アドバンスドモード

アクション

iot:\*

リソース ARN

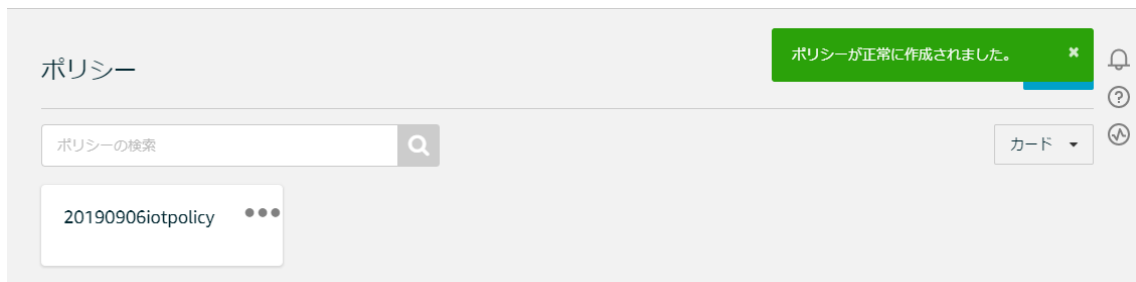
\*

効果

☒ 許可 ☐ 拒否

削除

2-8. ポリシーが作成されました。



2-9. 【管理】→【モノ】を選んでください。モノ、は AWS IoT が管理する IoT クライアント（デバイス）になります。このハンズオンではダミークライアントとして Cloud9 を使います。商用環境では大量の登録が発生するため、CLI 等プログラム化しておくことをお勧めしています。



モニタリング

オンボード

管理

モノ

タイプ

モノのグループ

請求グループ

ジョブ

2-10. 【モノの登録】を押します。





## まだモノがありません

モノはクラウド内のデバイスを表しています。

[詳細はこちら](#)

[モノの登録](#)

2-11. 【単一のモノを作成する】を選びます。

## AWS IoT モノを作成する

IoT の「モノ」とはクラウド内部の物理デバイスの表現とレコードを意味します。物理デバイスが AWS IoT と連携するには、モノのレコードが必要です。[詳細はこちら](#)。

単一の AWS IoT モノの登録  
レジストリにモノを作成します

単一のモノを作成する

AWS IoT モノの一括登録  
すでに AWS IoT を使用している多数のデバイスのモノをレジストリに作成します。または、AWS IoT に接続できるようにデバイスを登録します。

多数のモノの作成

キャンセル

単一のモノを作成する

2-12. 適当な名前を入力します。

モノの作成

ステップ 1/3

## Thing Registry にデバイスを追加

このステップは、デバイスの Thing Registry と Thing Shadow にエントリーを作成します。

名前

20190906things

このモノにタイプを適用

モノのタイプを使用すると、タイプを共有するモノに対して一貫した登録データを提供することで、デバイス管理が簡単になります。タイプは、デバイスのアイデンティティと機能を説明する共通の属性と説明を提供します。

モノのタイプ

タイプが選択されていません

タイプの作成

2-13. その他の設定は行わず、画面下の【次へ】を押します

検索可能なモノの属性の設定 (オプション)

レジストリ内のモノを検索できるように、これらの属性 (複数可) に値を入力してください。

属性キー

属性キー (メーカーなど) を指定します

値

属性値 (Acme-Corporation など) を指定します。

別のを追加

Thing Shadow の表示 ▼

クリア

キャンセル

戻る

次へ

## 2-14. 【証明書の作成】を押します

AWS IoT Core は通信及びデバイスのセキュリティ管理、制御に電子証明書を用いるため、認証局の機能を内蔵しています。ここで発行された認証局をクライアント (Cloud9) に組み込むことによって通信が可能となります。

モノの作成

モノに証明書を追加

ステップ 2/3

証明書は、AWS IoT へのデバイスの接続を認証するために使用されます。

1-Click 証明書作成 (推奨)

AWS IoT の認証局を使用して証明書、パブリックキー、プライベートキーを作成します。

証明書の作成

CSR による作成

所有しているプライベートキーに基づいて固有の証明書署名リクエスト (CSR) をアップロードします。

CSR による作成

お持ちの証明書を使用する

CA 証明書を登録し、1 つ以上のデバイスに独自の証明書を使用します。

開始方法

## 2-15. すべてを DL して【有効化】のボタンを押します。

デバイスを接続するには、次の情報をダウンロードします。

このモノの証明書	d8fee7d0d2.cert.pem	<a href="#">ダウンロード</a>
パブリックキー	d8fee7d0d2.public.key	<a href="#">ダウンロード</a>
プライベートキー	d8fee7d0d2.private.key	<a href="#">ダウンロード</a>

また、AWS IoT のルート CA をダウンロードする必要があります。

AWS IoT のルート CA [ダウンロード](#)

有効化

2-16. 【ポリシーのアタッチ】を押します。ポリシーは先ほど作成したデバイスが操作可能な AWS IoT の権限が設定されたものです。AWS IoT Core はデバイスの制御に証明書を使いますので、ポリシーを証明書に結び付けることになります。

デバイスを接続するには、次の情報をダウンロードします。

このモノの証明書	d8fee7d0d2.cert.pem	<a href="#">ダウンロード</a>
パブリックキー	d8fee7d0d2.public.key	<a href="#">ダウンロード</a>
プライベートキー	d8fee7d0d2.private.key	<a href="#">ダウンロード</a>

また、AWS IoT のルート CA をダウンロードする必要があります。  
AWS IoT のルート CA [ダウンロード](#)

無効化

キャンセル

完了

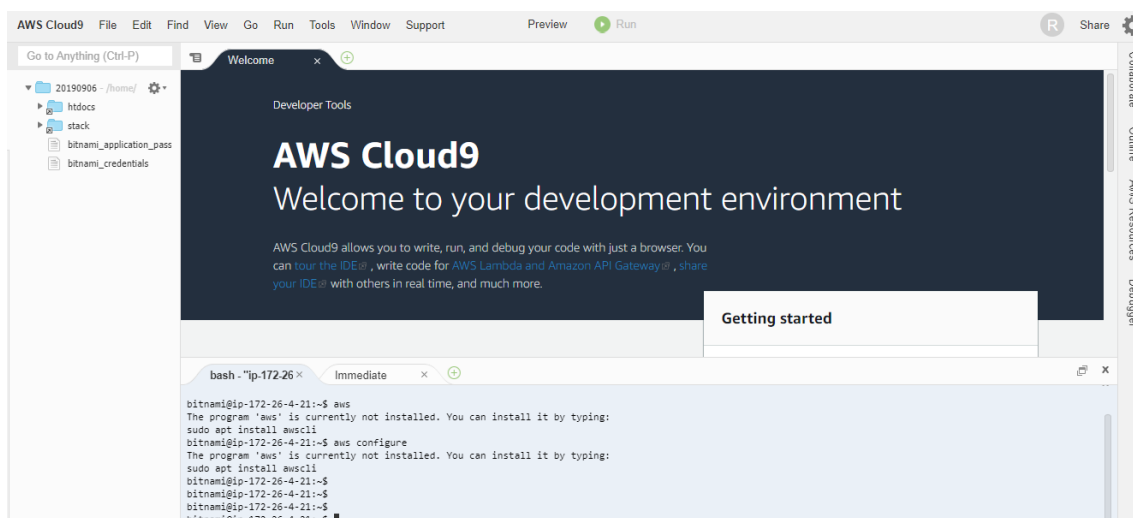
ポリシーをアタッチ

2-17. 先ほど作成したポリシーを選び【モノの登録】を押します。モノが作成されまし

た。



2-18. 作成した Cloud9 でターミナルの画面にいきます。



2-19. 以下のコマンドを実行

– `sudo ln -s /usr/bin/pip-3.6 /usr/bin/pip3`

ln: failed to create symbolic link '/usr/bin/pip3': File exists

が表示された場合、すでに Python3.6 がインストール済みですので、先に作業を進めてください。#IoT 用 SDK は Python3 が動作に必要です。

2-20. `pip3 -V` を実行し、以下の表示がされたらインストール完了です。

```
ec2-user:~/environment $ pip3 -V  
pip 9.0.3 from /usr/lib/python3.6/dist-packages (python 3.6)
```

2-21. Python SDK をインストールします。以下のコマンドを実行します。

- `sudo pip3 install AWSIoTPythonSDK`

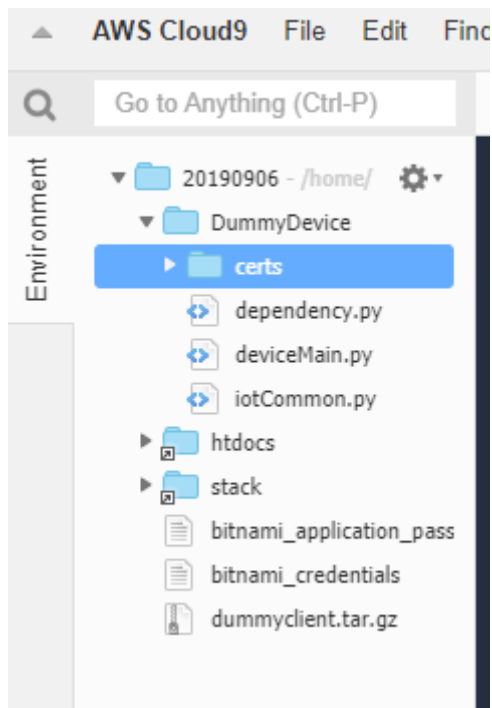
2-22. ダミークライアントのソースコードをダウンロードします。

- `wget http://bit.ly/2QggRgx -O dummyclient.tar.gz`

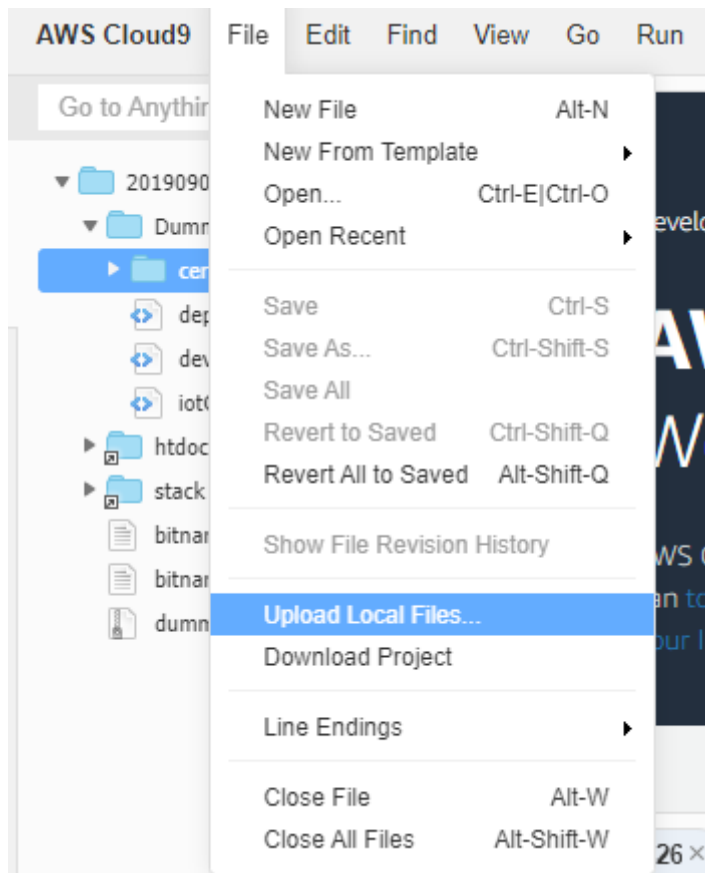
2-23. ダウンロードしたクライアントを解凍します。

- `tar -zxvf dummyclient.tar.gz`

2-24. 今の解凍で新しいフォルダができていますので、【DummyDevice】【certs】を選んで開きます。

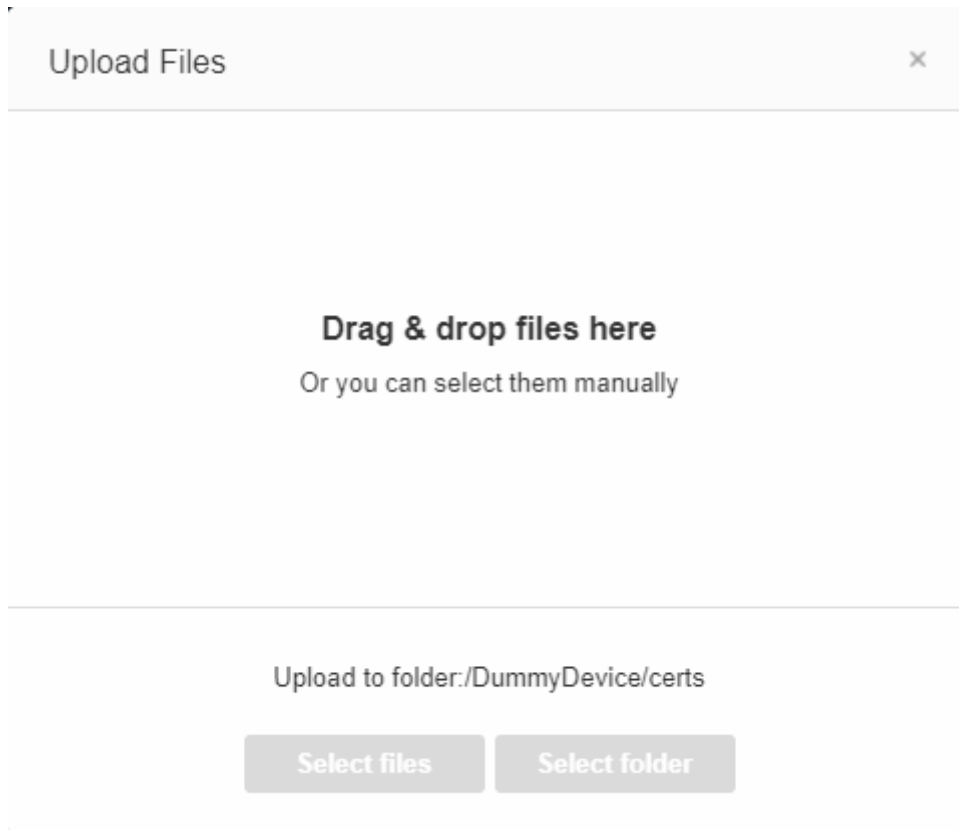


2-25. 【File】【Upload Local Files】を開きます。

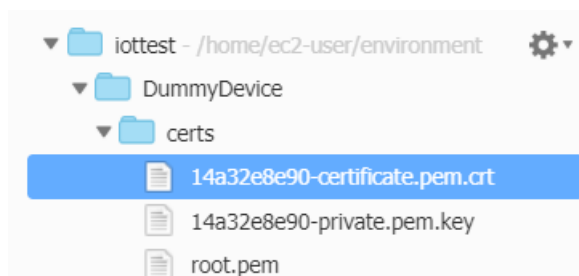


2-26. 先ほど DL した 2 つの電子証明書ファイル【\*\*-certificate.pem.crt】【\*\*-private.pem.key】を Upload します。

注意 : Windows 環境であれば、\*.crt ファイルは証明書を表すアイコンマークとなり、拡張子が表示されず\*.pem ファイルとなっています。



2-27. ファイルがコピーされたことを確認します。



2-28. \*\*\*private.pem.key のファイル名を private.pem に変更します。Cloud9 のシェルで変更してもいいですし、エクスプローラーで rename を選んでもいいです。

2-28. ディレクトリをシェル上で移動します。移動先は【DummyDevice】です。TAB を使うことができますので、たとえば cd D だけ入力して TAB を押すと残りは自動で補完さ



れます。

```
bash - "ip-172-26-4-21:~$ tar -zxvf dummyclient.tar.gz
DummyDevice/
DummyDevice/deviceMain.py
DummyDevice/dependency.py
DummyDevice/certs/
DummyDevice/certs/root.pem
DummyDevice/iotCommon.py
bitnami@ip-172-26-4-21:~$ ls
bitnami_application_password bitnami_credentials dummyclient.tar.gz DummyDevice httdocs stack
bitnami@ip-172-26-4-21:~$ cd DummyDevice/
bitnami@ip-172-26-4-21:~/DummyDevice$
```

2-29. 以下のコマンドを入力します。赤字は置き換えてください。

python3 deviceMain.py --device\_name **ご自分の作ったモノ名** --endpoint **AWS IoT の endpoint\_url**

2-30. 疎通が完了すると以下のような画面が表示されます。

```
bitnami@ip-172-26-4-21:~/DummyDevice$ python3 deviceMain.py --device_name 20190906things --endpoint afhmd7pja59at-ats.iot.ap-northeast-1.amazonaws.com
start >>>
device_name: 20190906things
endpoint: afhmd7pja59at-ats.iot.ap-northeast-1.amazonaws.com
rootca cert: ./certs/root.pem
private key: ./certs/private.pem
certificate: ./certs/d8fee7d0d2-certificate.pem.crt
connect to AWS IoT >>>
topic: data/20190906things
send back state payload:{"state": {"reported": {"wait_time": 5}}}
```

2-31. AWS IoT の管理画面でテストを選びます。



モニタリング

オンボード

管理

Greengrass

安全性

防御

ACT

テスト

2-32. 【トピックのサブスクリプション】の欄に data/{モノの名前}を入力し【サブス  
クライブ】ボタンを押します。{モノの名前}は皆さんが作成した名前です。

data/20190906things

エクスポート   クリア   一時停止

発行

QoS を 0 にして発行するトピックとメッセージを指定します。

data/20190906things

トピックに発行

```
1 {
2   "message": "Hello from AWS IoT console"
3 }
```

data/20190906things

2019/09/06 16:39:11

エクスポート   非表示

```
{
  "TIMESTAMP": "2019-09-06T07:39:11",
  "DEVICENAME": "20190906things",
  "VALUE": 25
}
```

2-33. ダミークライアントの設定が 5 秒間隔でのステータス同期となっているので、5 秒ごとにデータが 1 個ずつ増えていきます。

data/20190906things

2019/09/06 16:40:37

エクスポート   非表示

```
{
  "TIMESTAMP": "2019-09-06T07:40:37",
  "DEVICENAME": "20190906things",
  "VALUE": 22
}
```

data/20190906things

2019/09/06 16:40:32

エクスポート   非表示

```
{
  "TIMESTAMP": "2019-09-06T07:40:32",
  "DEVICENAME": "20190906things",
  "VALUE": 21
}
```

data/20190906things

2019/09/06 16:40:27

エクスポート   非表示

```
{
  "TIMESTAMP": "2019-09-06T07:40:27",
  "DEVICENAME": "20190906things",
  "VALUE": 20
}
```

## 1. デバイスシャドウによるデバイスの操作

AWS IoT にはデバイスシャドウという機能があります。クライアントデバイスが送ってきたステータスを、管理者側が書き換えることでクライアントデバイスの挙動を変更させることができます。上記でテストした 5 秒おきに送られてくるデータを 10 秒おきに送られてくるように変更します。

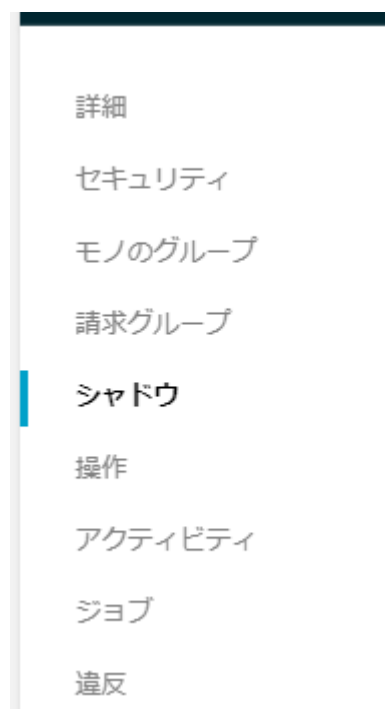
### 3-1. データ間隔が 5 秒おきになっていることを確認します。

data/20190906things	2019/09/06 16:52:58	<a href="#">エクスポート</a> <a href="#">非表示</a>
<pre>{   "TIMESTAMP": "2019-09-06T07:52:58",   "DEVICENAME": "20190906things",   "VALUE": 18 }</pre>		
data/20190906things	2019/09/06 16:52:53	<a href="#">エクスポート</a> <a href="#">非表示</a>
<pre>{   "TIMESTAMP": "2019-09-06T07:52:53",   "DEVICENAME": "20190906things",   "VALUE": 15 }</pre>		

### 3-2. 【管理】【モノ】を選びます。



3-3. 【シャドウ】を選択します。



3-4. データが 5 秒間隔で上がってくることが定義されています。これを 10 秒に書き

換えるため【編集】を押します。

シャドウ ARN は、この Thing Shadow を一意に識別します。詳細はこちら

```
arn:aws:iot:ap-northeast-1:294963776963:thing/20190906things
```

シャドウドキュメント

削除 編集

最終更新日: 2019/09/06 16:32:18

シャドウステータス:

```
{
  "reported": {
    "wait_time": 5
  }
}
```

こちらの値が空欄の場合、Cloud9 上の Dummy Client を一度停止して、再度起動し 5 秒待ってください。

3-5. 以下のように置換し【保存】を押します。

シャドウドキュメント

削除 キャンセル 保存

最終更新日: 2019/09/06 16:32:18

シャドウステータス:

```
1
2 {
3   "desired": {
4     "wait_time": 10
5   },
6   "reported": {
7     "wait_time": 5
8   }
9 }
10
```

3-6. Cloud9 の画面に戻ると、指示を受信した旨が表示されています。

```
-----
send back state payload:{"state": {"reported": {"wait_time": 5}}}
delta payload:{"version":4,"timestamp":1567756601,"state":{"wait_time":10},"metadata":{"wait_time":{"timestamp":1567756601}}}
{"state": {"reported": {"wait_time": 10}}}
send back state payload:{"state": {"reported": {"wait_time": 10}}}
```

3-7. AWS IoT Core のテスト画面で、同期間隔が 10 秒になっていることを確認します。

data/20190906things	2019/09/06 16:58:14	<a href="#">エクスポート</a> <a href="#">非表示</a>
<pre>{   "TIMESTAMP": "2019-09-06T07:58:14",   "DEVICENAME": "20190906things",   "VALUE": 23 }</pre>		
data/20190906things	2019/09/06 16:58:04	<a href="#">エクスポート</a> <a href="#">非表示</a>
<pre>{   "TIMESTAMP": "2019-09-06T07:58:04",   "DEVICENAME": "20190906things",   "VALUE": 19 }</pre>		

## 2. ルールエンジン

AWS IoT Core にはルールエンジンという機能が存在しています。クライアントデバイスから上がってきたデータの中身をもとに、SQL を実行し、データの中身によりその後の AWS 上の挙動を変更させます。このハンズオンでは、データが[s3]という文字列であった場合のみ、s3 にデータを保存する手順を行います。

4-1. 【ACT】を押します。



モニタリング

オンボード

管理

Greengrass

安全性

防御

**ACT**

テスト

4-2. 【ルール作成】を押します。



ルールはまだ作成されていません。

ルールを使用すると、AWS などのウェブサービスとやり取りする権限をモノに許可できます。ルールは、モノより送信されるメッセージに基づいて、分析し、アクションを実行します。

[詳細はこちら](#)

[ルールの作成](#)

4-3. 適当な名前を入力します。



## ルールの作成

モノにより送信されるメッセージを評価し、メッセージを受信したときの処理を指定するルールを作成します (DynamoDB テーブルにデータを書き込む、Lambda 関数を呼び出すなど)。

名前

20190906s3rule

説明

4-4. 以下の SQL を入力します。

```
select name from 'data/{モノの名前}' where name = 's3'
```

### ルールクエリステートメント

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. SQL ステートメントを構築する方法については、「[AWS IoT SQL リファレンス](#)」を参照してください。

```
1 select name from 'data/20190906things' where name = 's3'
```

4-5. 【アクションの追加】を押します。

### 1 つ以上のアクションを設定する

インバウンドメッセージが上記のルールに一致すると、1 つ以上のアクションが選択されます。メッセージ受信時に発生する追加アクティビティ (データベースへの格納、クラウド関数の呼び出し、通知の送信など) を定義するアクション。(\*必須)

アクションの追加

4-6. 複数の AWS リソースとの連携が用意されています。S3 を選びます。

<input type="radio"/>	 Amazon Kinesis ストリームにメッセージを送信する AMAZON KINESIS
<input type="radio"/>	 AWS IoT のトピックにメッセージを再パブリッシュする AWS IoT の再パブリッシュ
<input type="radio"/>	 Amazon S3 バケットにメッセージを格納する S3
<input type="radio"/>	 Amazon Kinesis Firehose ストリームにメッセージを送信する AMAZON KINESIS FIREHOSE
<input type="radio"/>	 CloudWatch にメッセージデータを送信する CLOUDWATCH メトリクス
<input type="radio"/>	 CloudWatch アラームの状態を変更する CLOUDWATCH アラーム

4-7. 【アクションの設定】を押します。

キャンセル
アクションの設定

4-8. 【新しいリソースを作成する】を押します。

### アクションの設定

 Amazon S3 バケットにメッセージを格納する  
S3

このアクションによって、メッセージは S3 バケットのファイルに書き込まれます。

\*S3 バケット

リソースの選択

↺

新しいリソースを作成する

\*キー 

4-9. 【バケットを作成する】を押します。



4-10. 適当なバケット名を入力し、すべてデフォルトのまま【次へ】を3回押し【バケットの作成】を押します。

4-11. IoT の画面に戻り、ぐるぐるしたマークをおすと、先ほど作成したバケットが表示されますので、選択します。



4-12. キーに「test」と入力します。



4-13. 【ロールの作成】を押します。

このアクションを実行するための AWS IoT アクセス権限を付与するロールを選択または作成します。

選択されたロールがありません	更新	ロールの作成	閉じる
<input type="text" value="IAM ロールを検索"/>			

4-14. 適当な名前を付けて【ロールの作成】を押します。

### 新しいロールの作成

新しい IAM ロールがお客様のアカウントに作成されます。AWS IoT がお客様に代わってリソースにアクセスすることを許可するスコープダウンされたアクセス許可を提供するロールにインラインポリシーがアタッチされます。

名前

このフィールドは必須です。

キャンセル

ロールの作成

4-15. 【アクションの追加】 ボタンを押します。

このアクションを実行するための AWS IoT アクセス権限を付与するロールを選択または作成します。

20190906iottestrole	アタッチされたポリシー ✓	ロールの作成	選択
---------------------	---------------	--------	----

キャンセル

アクションの追加

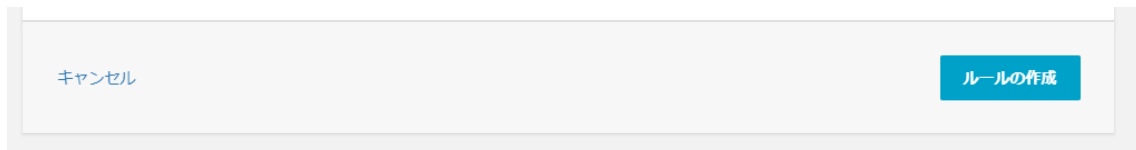
4-16. S3 への書き込み設定が完了しました。

1 つ以上のアクションを設定する

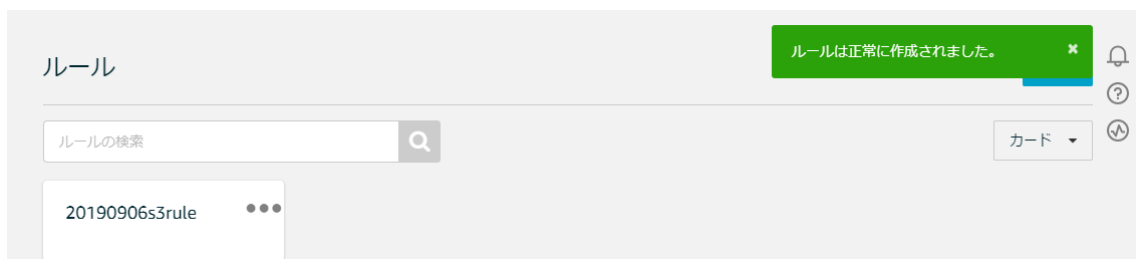
インバウンドメッセージが上記のルールに一致すると、1 つ以上のアクションが選択されます。メッセージ受信時に発生する追加アクティビティ (データベースへの格納、クラウド関数の呼び出し、通知の送信など) を定義するアクション。(\*必須)

	Amazon S3 バケットにメッセージを格納する 20190906iottest	削除	編集 ▶
---	--	----	------

4-17. 【ルール作成】を押します。



4-18. ルールが作成されています。これで「s3」というデータを含んだ通信が来た際に、S3 上にファイルが作成されるようになります。

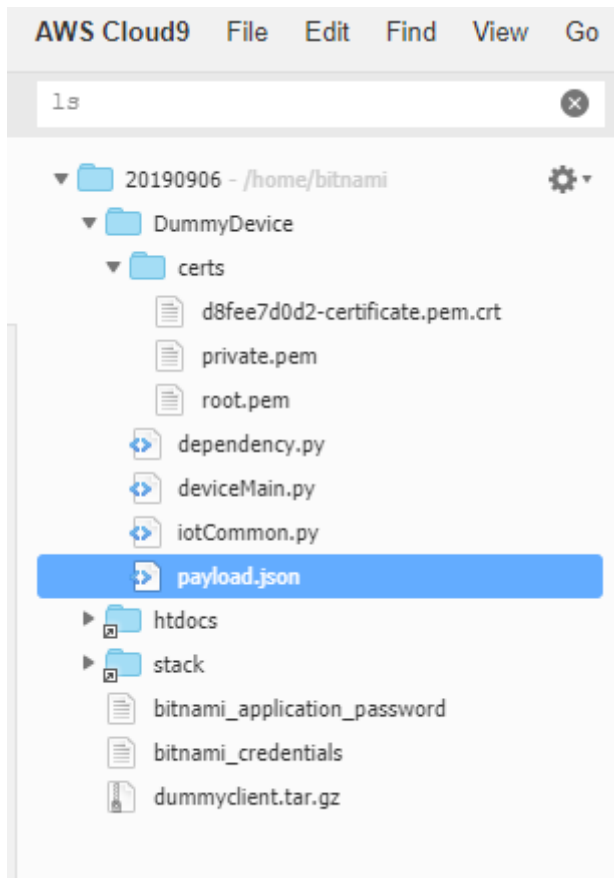


4-19. テキストファイルを開いて以下のコマンドを入力します。(AWS IoT のデータは JSON 形式です。)(シェルでの作業に慣れている方は、Cloud9 上でそのままファイルを作成しても問題ありません。)

```
{ "name": "s3" }
```

入力後、ファイル名を【payload.json】で保存します。

4-20. 保存したファイルを Cloud9 上の【DummyDevice】 フォルダにアップロードします。



4-21. シェルで `ctr + c` を押して、先ほどの通信を止めます。

4-22. 以下のコマンドを入力します。

```
curl -D - --tlsv1.2 -X POST --cert ./certs/{証明書ファイル名} --
```

```
key ./certs/private.pem --cacert ./certs/root.pem https://{エンドポイン
```

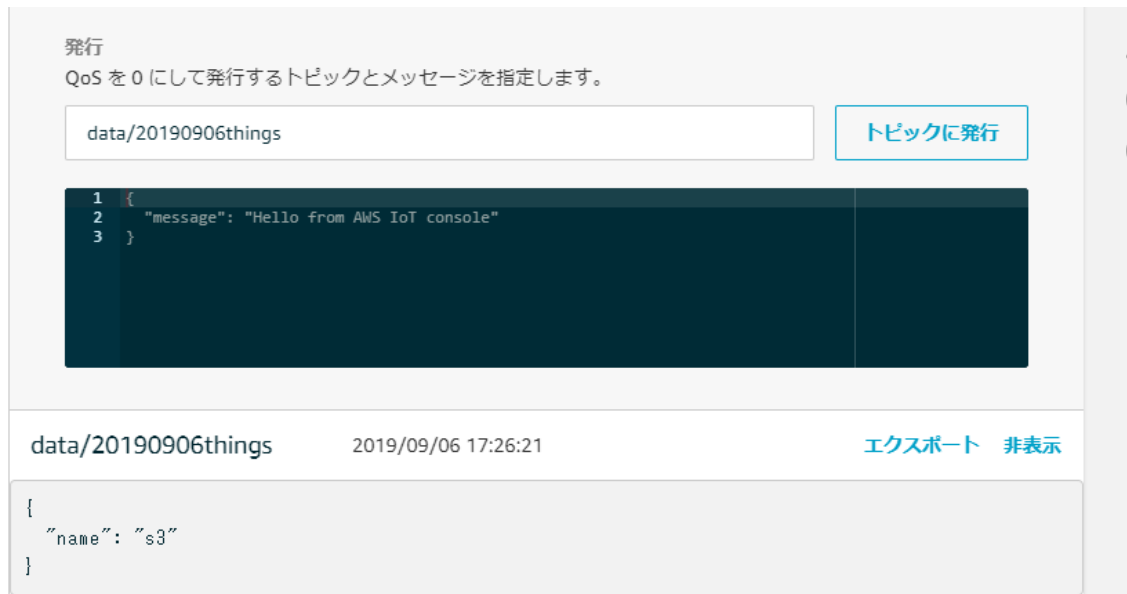
```
ト}:8443/topics/data/{モノの名前}?qos=0 -d @payload.json
```

動作すると以下の表示になります。

```
HTTP/1.1 200 OK
content-type: application/json
content-length: 65
date: Fri, 06 Sep 2019 08:22:31 GMT
x-amzn-RequestId: 4202a6c5-a439-c1de-de50-f79bc2f4990c
connection: keep-alive

{"message":"OK","traceId":"4202a6c5-a439-c1de-de50-f79bc2f4990c"}bitnami@ip-172-26-4-21:~/DummyDevice$
```

4-23. テスト画面で受信したデータの確認が可能です。[s3]と表示されていれば成功です。この画面を閉じてしまっており、新しく設定する場合、メッセージが表示されていないはずなので、もう一度 curl のコマンドを実行してください。



発行  
QoS を 0 にして発行するトピックとメッセージを指定します。

data/20190906things [トピックに発行](#)

```
1 {  
2   "message": "Hello from AWS IoT console"  
3 }
```

data/20190906things 2019/09/06 17:26:21 [エクスポート](#) [非表示](#)

```
{  
  "name": "s3"  
}
```

4-24. 作成した s3 バケットを見てみましょう。データが保存されています。



Amazon S3 > 20190906iottest

概要 **プロパティ** アクセス権限 管理

🔍 プレフィックスを入力し、Enter キーで検索します。ESC を押してクリアします。

[アップロード](#) [フォルダの作成](#) [ダウンロード](#) [アクション](#) アジアパシフィック (東京) 🔄

表示中 1 ~ 1			
<input type="checkbox"/> 名前 ▼	最終更新日時 ▼	サイズ ▼	ストレージクラス ▼
<input type="checkbox"/> 📄 test	9月 6, 2019 5:26:22 午後 GMT+0900	13.0 B	スタンダード

データを送るたびに最終更新日時が上書きされています。この手順ではデータが単一ファイルを上書きしていきますが、Timestamp をベースとして都度都度ファイル名を変更させることができます。また、payload.json の中身を書き換えて、s3 が含まれていない通

信は、s3 のファイル更新日が上書きされないことを確認しましょう。

### 3. 削除

お疲れ様でした！以上でハンズオン終了です。

以下を必ず削除してください。

- AWS Cloud9
- AWS IoT のモノ