はじめての IoT ~AWS IoT Core ハンズオン~

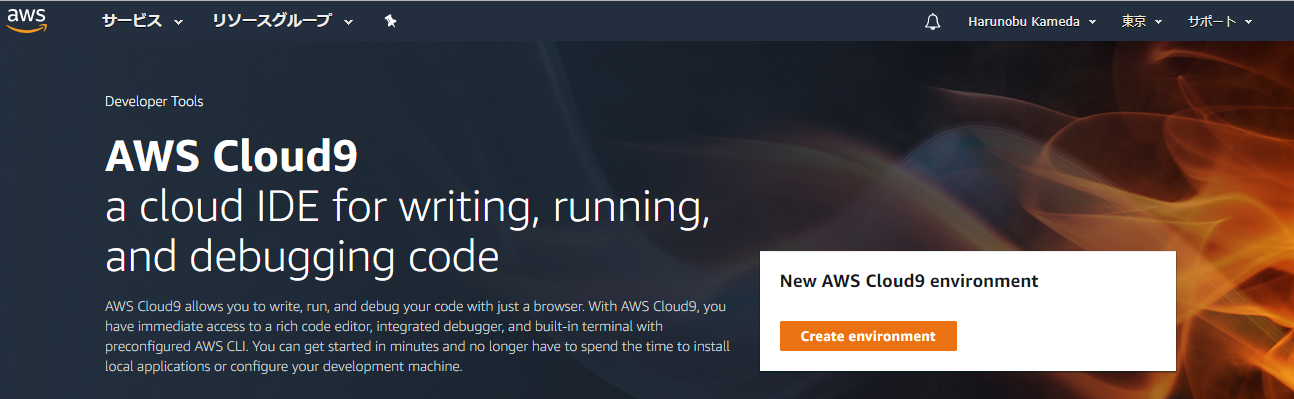
アマゾン ウェブ サービス ジャパン  
エバンジェリスト

亀田 治伸

IoTダミークライアントの作成

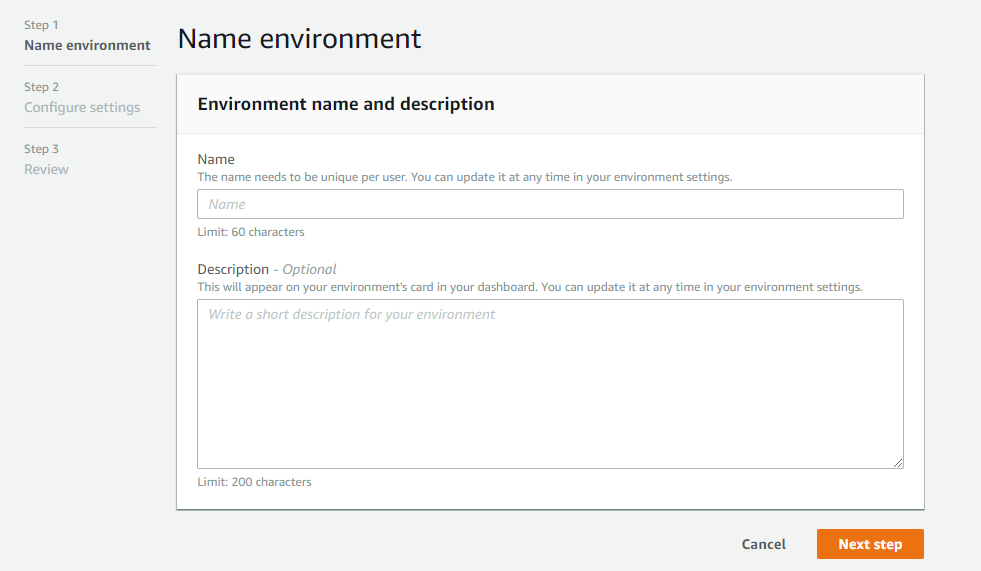
AWS IoT Coreと通信（MQTT及びHTTP）を行うIoTクライアントを構築します。このハンズオンでは、デバイスを用いず、AWS Cloud9にAWS IoT SDKをインストールします。

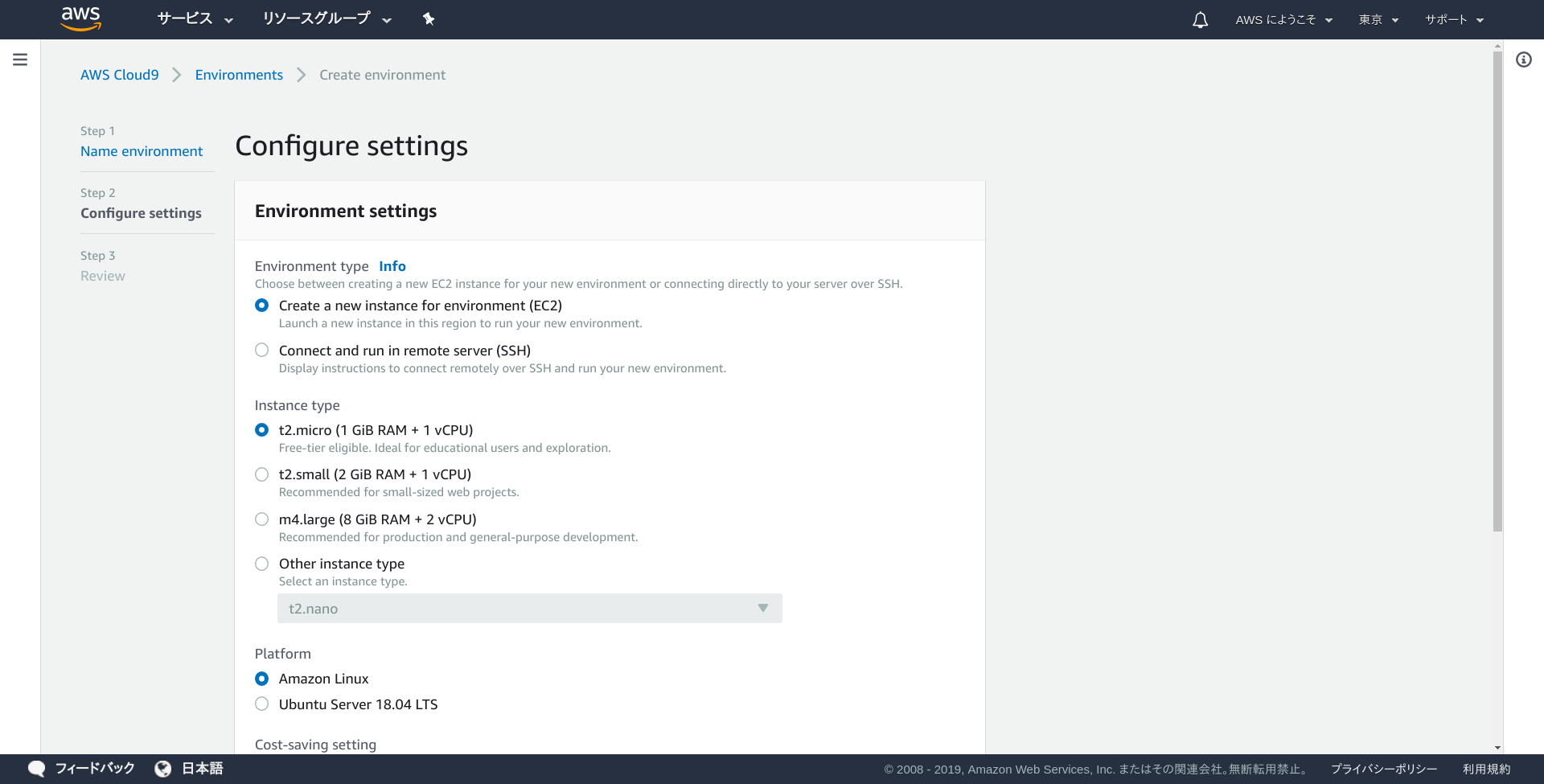
* 1. AWS Cloud９の画面にアクセスします。（ブラウザの別タブで開くことをお勧めします）



1-2.　【Create Environment】を押します。

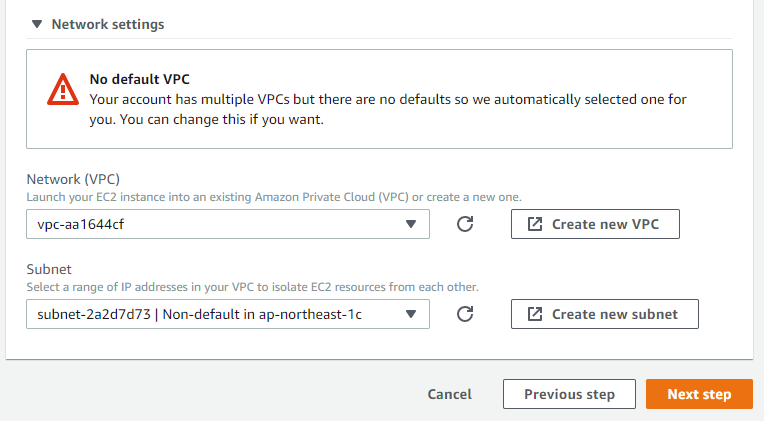
　　適当な名前を付けて、【Next Step】を押します。



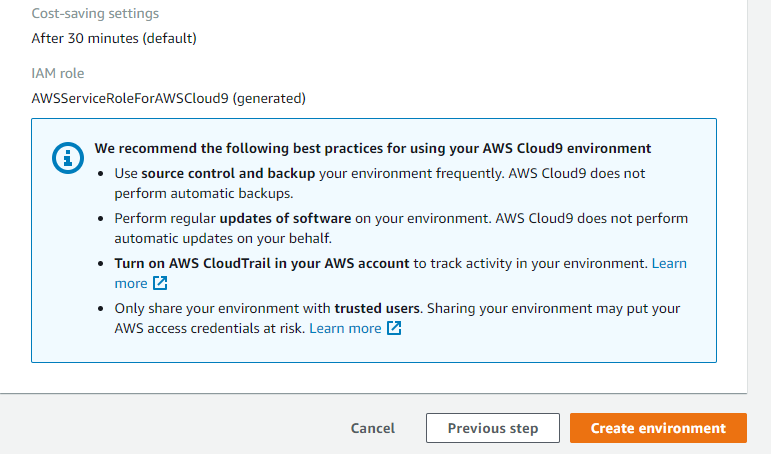


1-3.　 「Create a new instance for environment (EC2)」にチェックを入れる。

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。**

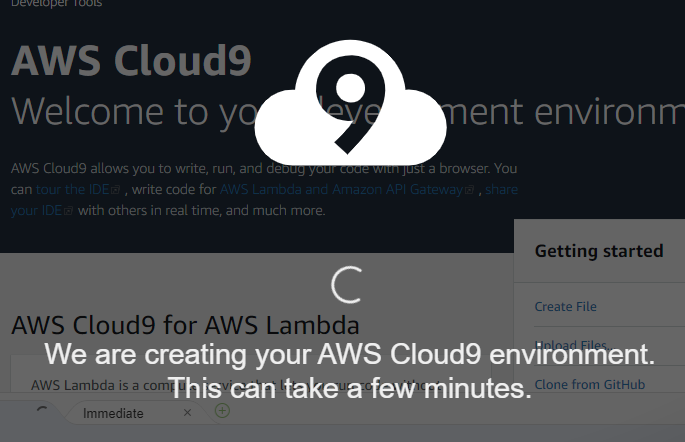


1-4. 任意のVPCを選択し、Publicサブネットを選びます。不明な場合は、スタッフに聞いてください。



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。

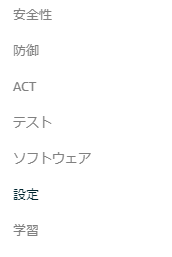


1. AWS IoT クライアントの設定

2-1.　AWS IoTのトップ画面へアクセスします



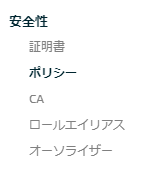
2-2.　画面左下の【設定】を押します



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。



2-4.　【安全性】→　【Policy】を選びます。



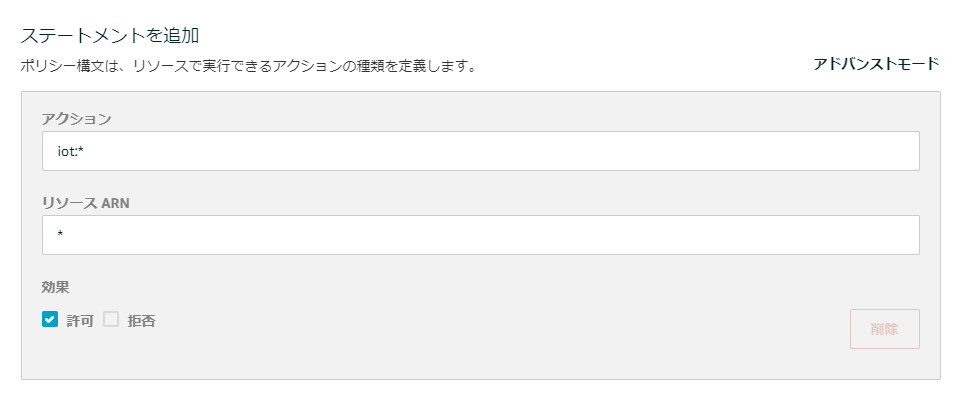
2-5．　【ポリシーの作成】をクリックします。



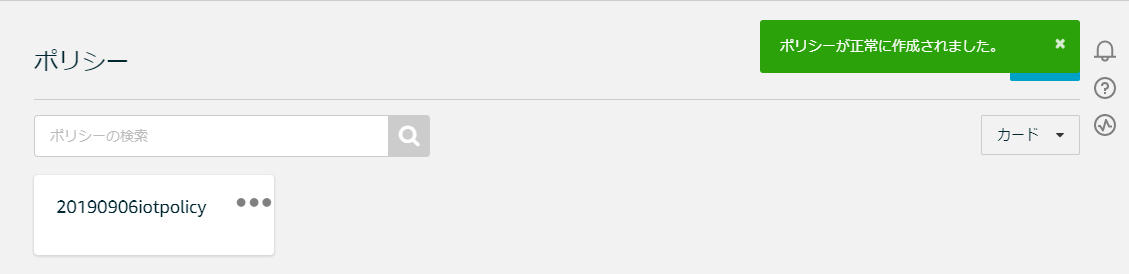
2-6.　適当な名前を付けます。ここで作成したポリシーは、AWS IoT Coreと通信を行うクライアントが持つべきセキュリティポリシー（AWS IoT Coreの複数の機能と連携できる・できない等）になります。



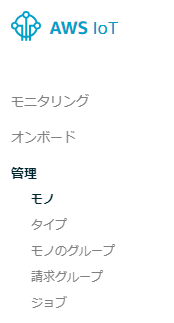
2-7.　以下の表示と同じ値を入力し、【作成】を押します。このハンズオンではAWS IoTのすべての機能を使えるポリシーを作成します。（AWSのその他リソースを操作できる権限ではないことに注意してください）【iot:\*】【\*】



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。



2-10.　【モノの登録】を押します。



2-11．　【単一のモノを作成する】を選びます。



2-12．　適当な名前を入力します。



2-13.　その他の設定は行わず、画面下の【次へ】を押します



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。



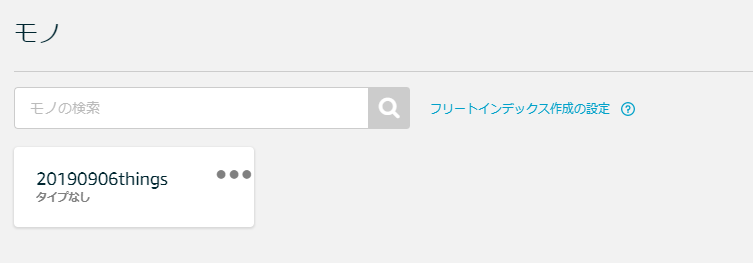
2-15.　すべてをDLして【有効化】のボタンを押します。



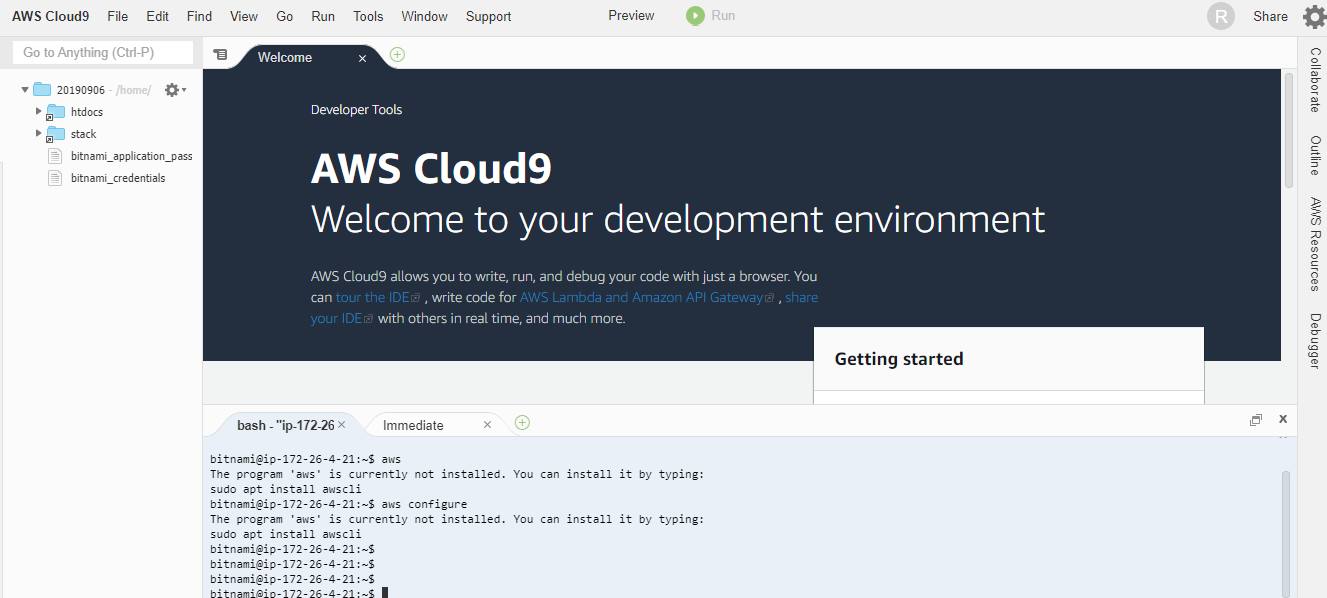
2-16.　【ポリシーのアタッチ】を押します。ポリシーは先ほど作成したデバイスが操作可能なAWS IoTの権限が設定されたものです。AWS IoT Coreはデバイスの制御に証明書を使いますので、ポリシーを証明書に結び付けることになります。



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。



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

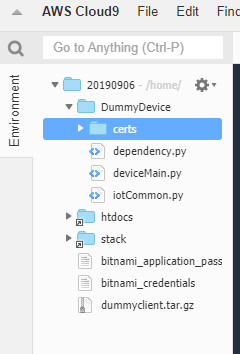
2-22．　ダミークライアントのソースコードをダウンロードします。

* + wget <http://bit.ly/2QggRgx> [-O dummyclient.tar.gz](http://bit.ly/2GxsLxU%20-O%20dummyclient.tar.gz)

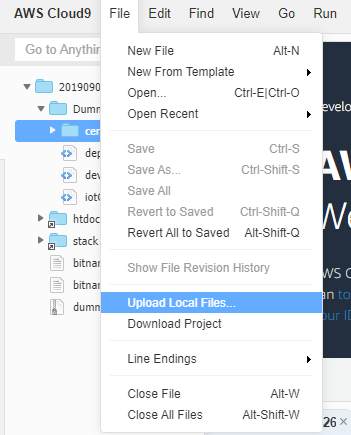
2-23.　ダウンロードしたクライアントを解凍します。

* + 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。

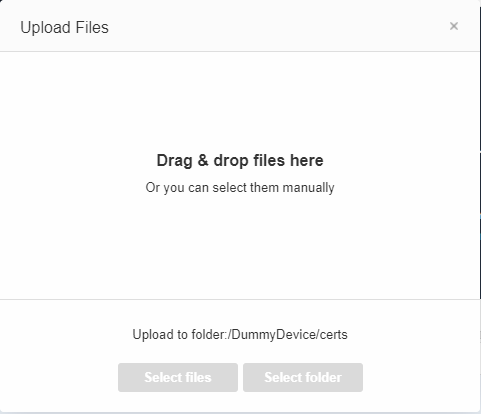


2-25.　【File】【Upload Local Files】を開きます。

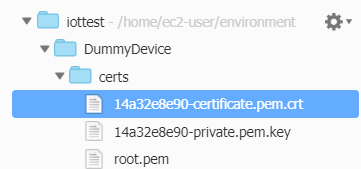


2-26.　先ほどDLした2つの電子証明書ファイル【\*\*-certificate.pem.crt 】【\*\*-private.pem.key】をUploadします。

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。

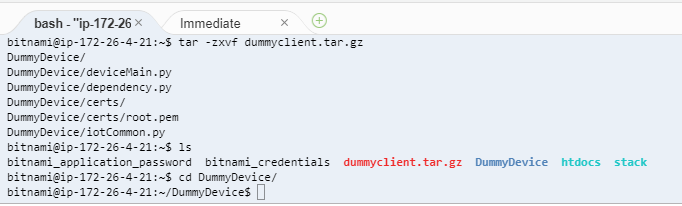


2-27.　ファイルがコピーされたことを確認します。



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。

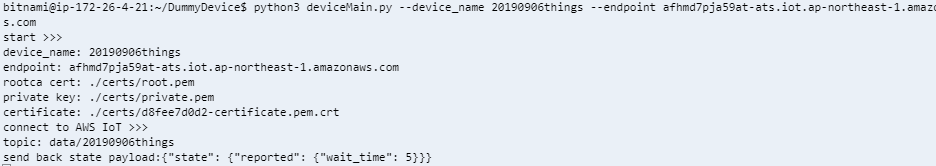
2-28.　ディレクトリをシェル上で移動します。移動先は【DummyDevice】です。TABを使うことができますので、たとえば cd D だけ入力してTABを押すと残りは自動で補完されます。



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。



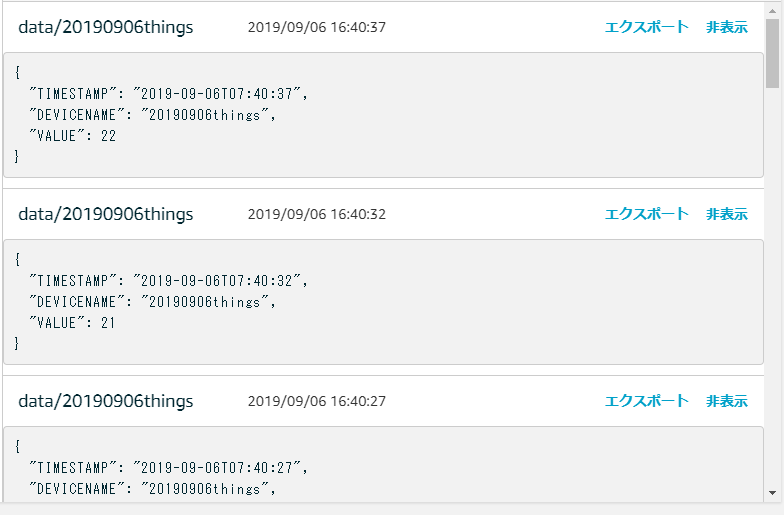
2-31.　AWS IoT の管理画面でテストを選びます。



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。



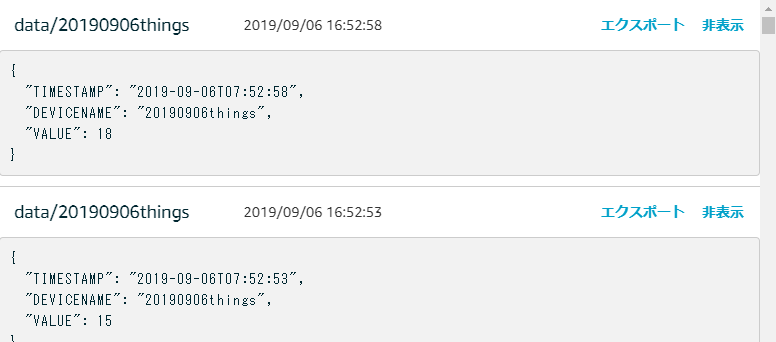
2-33.　ダミークライアントの設定が5秒間隔でのステータス同期となっているので、5秒ごとにデータが1個づつ増えていきます。



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。



3-2．　【管理】【モノ】を選びます。



3-3．　【シャドウ】を選択します。

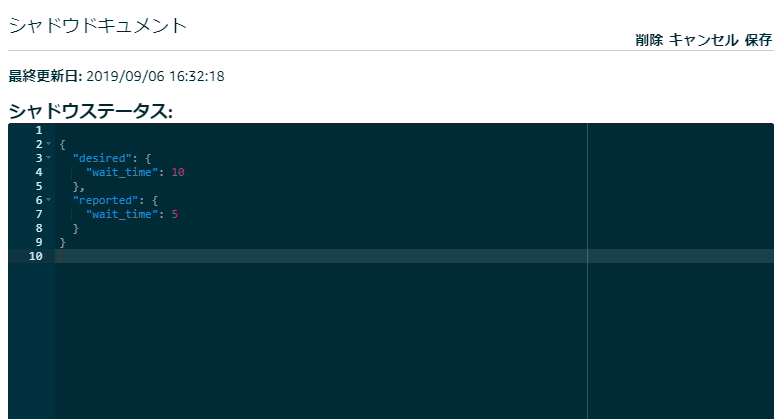


3-4．　データが5秒間隔で上がってくることが定義されています。これを10秒に書き換えるため【編集】を押します。

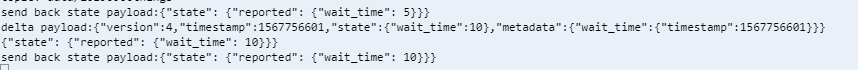


こちらの値が空欄の場合、Cloud9上のDummy Clientを一度停止して、再度起動し5秒待ってください。

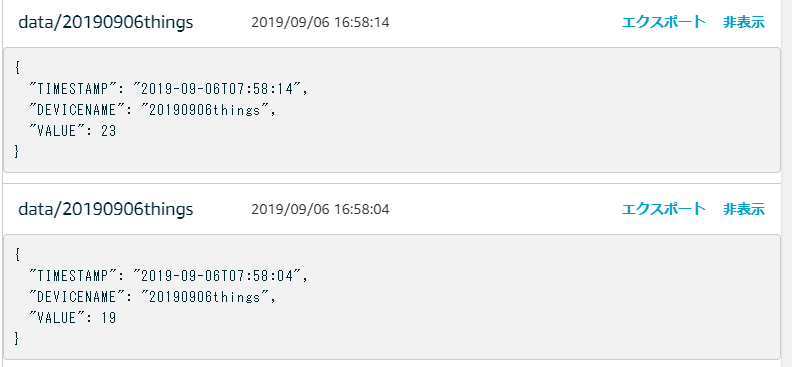
3-5.　以下のように置換し【保存】を押します。



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。



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。



1. 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。

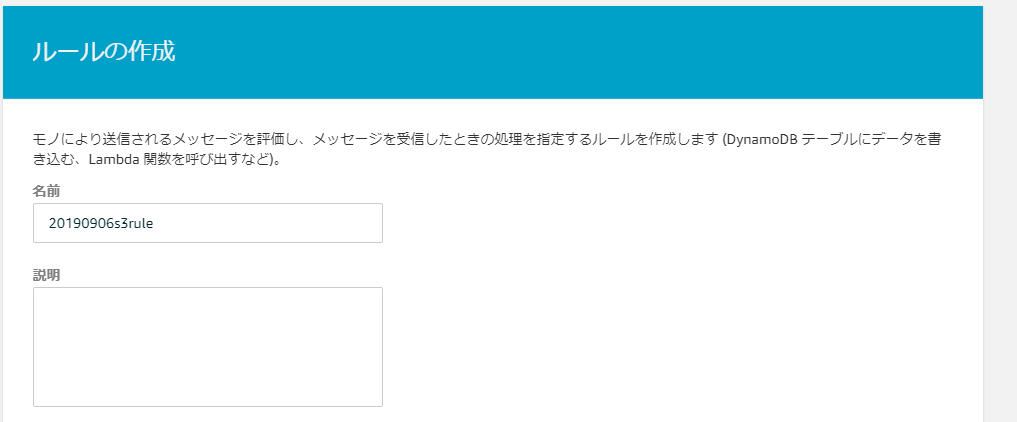
4-1．【ACT】を押します。



4-2．　【ルールの作成】を押します。

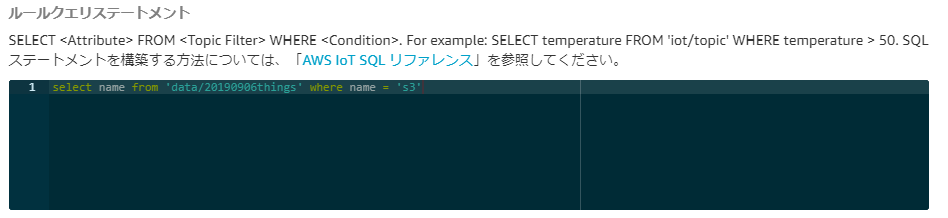


4-3．　適当な名前を入力します。

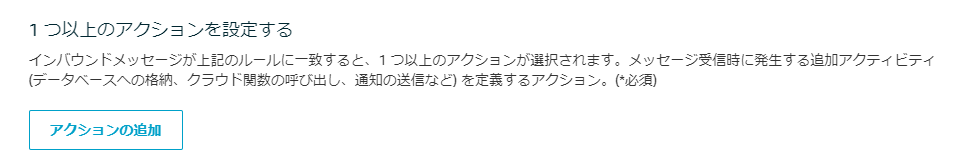


4-4．　以下のSQLを入力します。

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

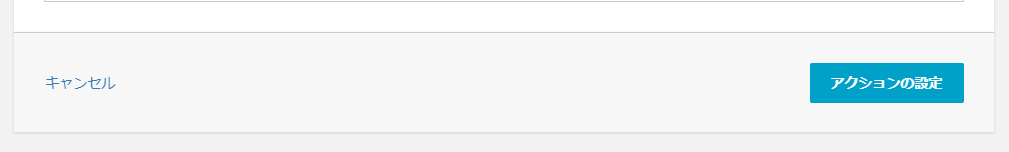


4-5.　【アクションの追加】を押します。

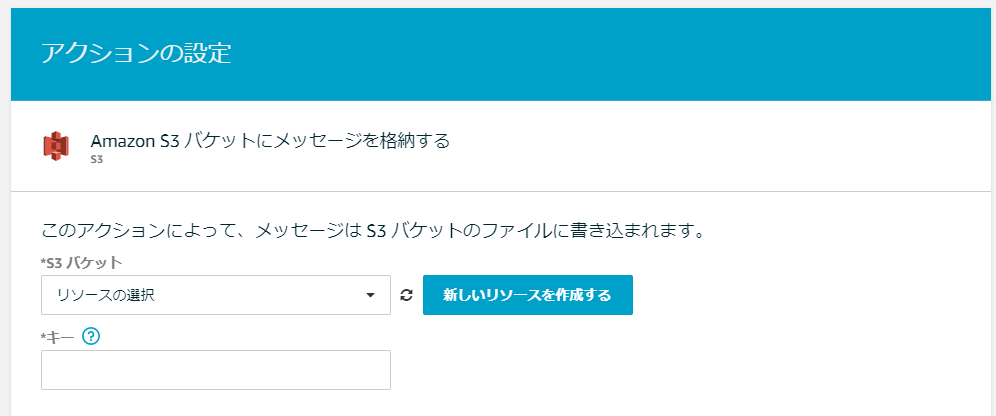
4-6.　複数のAWSリソースとの連携が用意されています。S3を選びます。



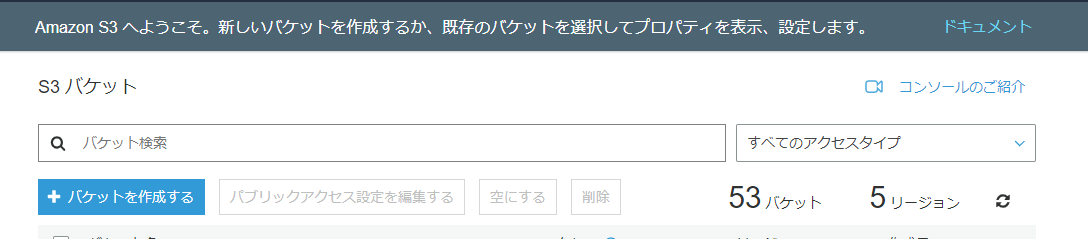
4-7.　【アクションの設定】を押します。



4-8.　【新しいリソースを作成する】を押します。



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。

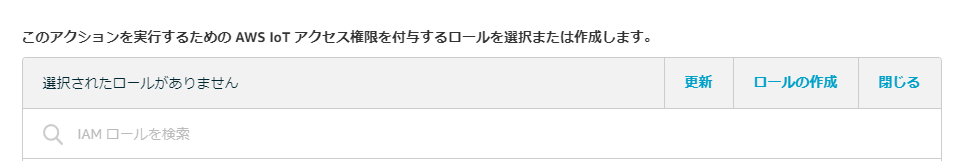
4-11.　IoTの画面に戻り、ぐるぐるしたマークをおすと、先ほど作成したバケットが表示されますので、選択します。



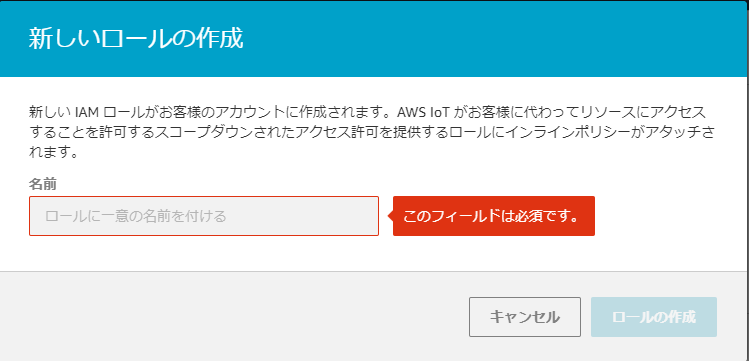
4-12.　キーに「test」と入力します。



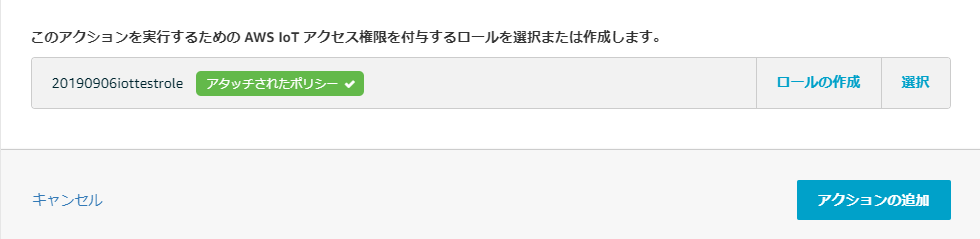
4-13.　【ロールの作成】を押します。



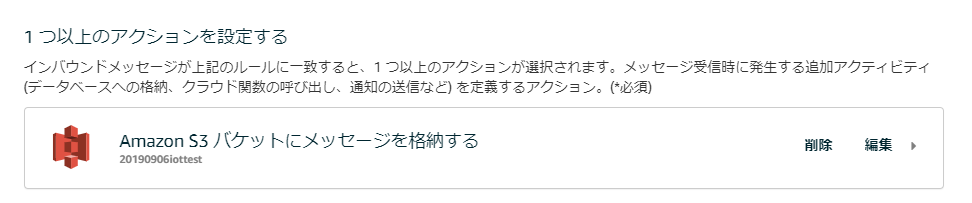
4-14.　適当な名前を付けて【ロールの作成】を押します。



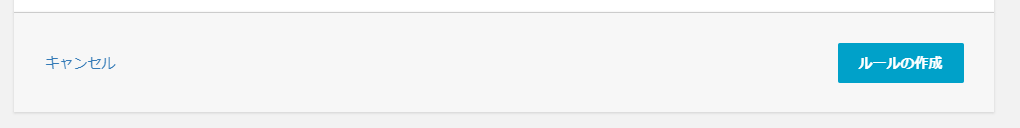
4-15.　【アクションの追加】ボタンを押します。



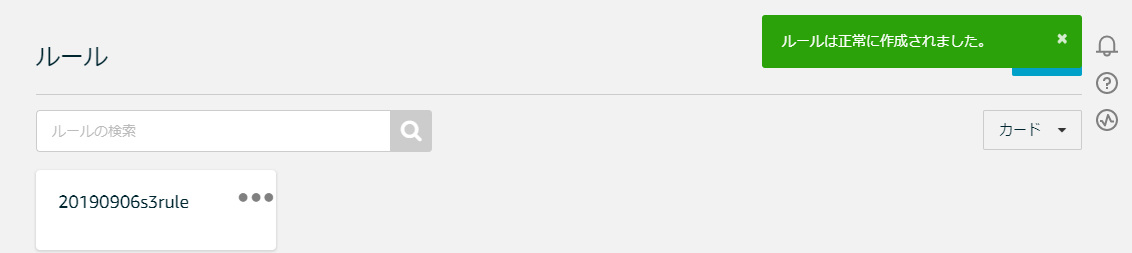
4-16.　S3への書き込み設定が完了しました。



4-17.　【ルールの作成】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。

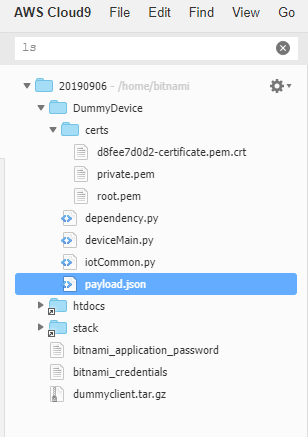


4-19.　テキストファイルを開いて以下のコマンドを入力します。（AWS IoTのデータはJSON形式です。）（シェルでの作業に慣れている方は、Cloud9上でそのままファイルを作成しても問題ありません。）

{ "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。

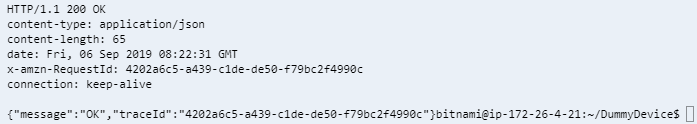


4-21.　シェルで ctr + cを押して、先ほどの通信を止めます。

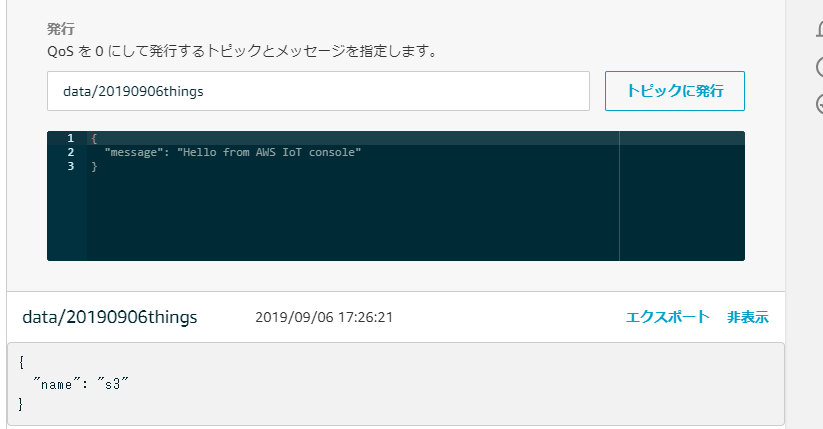
4-22.　以下のコマンドを入力します。

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。



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、メッエ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。



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。



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。

1. 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。

・Amazon Lightsail

・AWS Cloud9

・AWS IoTのモノ