
IoT Hands-On Powered by OpenBlocks IoT | SORACOM | AWS / Documentation

リリース *1.0.0*

Kohei MATSUSHITA

2016年05月20日

Contents

1	概要	3
1.1	全体構成	3
1.2	ハンズオンの前に	4
1.3	BX1 の Wi-Fi AP 設定と SORACOM Air(3G ネットワーク) 設定	6
1.4	センサーと BX1 の接続	14
1.5	Amazon Elasticsearch Service のインスタンス作成と設定	25
1.6	AWS IoT の設定	31
1.7	BX1 と AWS IoT の接続	44
1.8	片付け&まとめ	51
1.9	自習室	51
1.10	自習室: SORACOM Beam で AWS IoT の認証処理をオフロード	53
1.11	自習室: AWS IoT の Thing Shadow でパトライトを制御する	66
1.12	時間配分	70

Auhtor: Kohei MATSUSHITA

Chapter 1

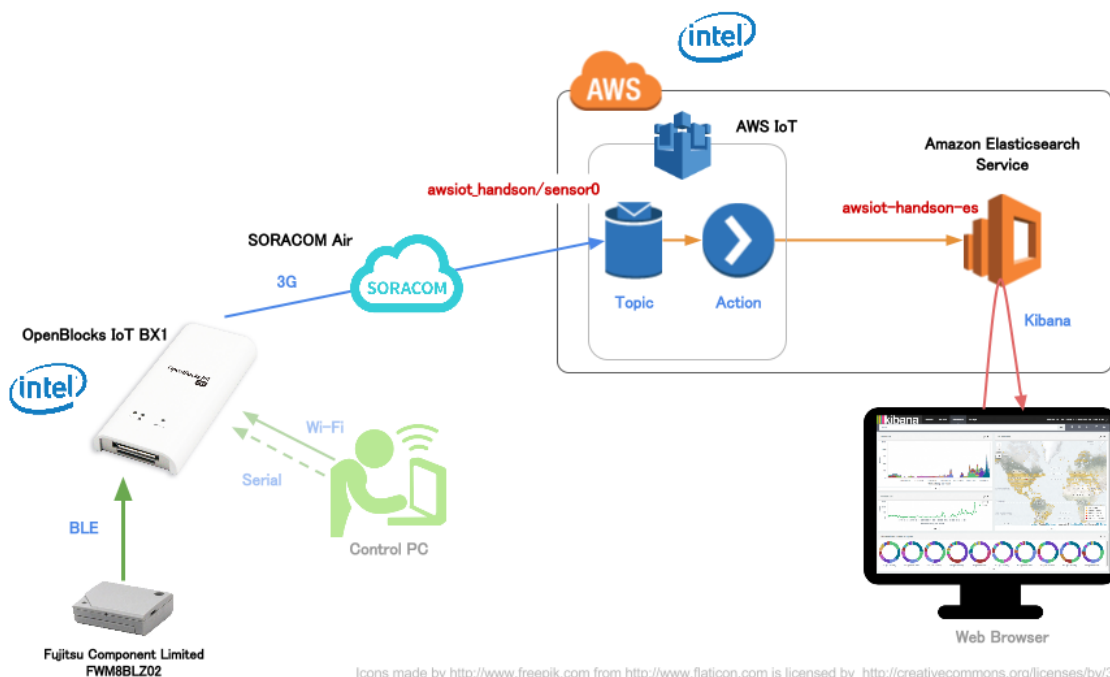
概要

OpenBlocks IoT BX1 (以下 BX1) を使用し、富士通コンポーネント社製 温度・加速度センサーデバイス “FWM8BLZ02” のデータを “SORACOM Air” の回線で “AWS IoT” に送信し、Amazon Elasticsearch Service(以下 Amazon ES) 上の Kibana でグラフ化るところまでを解説します

全体構成

OpenBlocks IoT BX1 (以下 BX1) を使用し、富士通コンポーネント社製 温度・加速度センサーデバイス “FWM8BLZ02” のデータを “SORACOM Air” の回線で “AWS IoT” に送信し、”Amazon Elasticsearch Service” (以下 Amazon ES) 上の Kibana でグラフ化るところまでを解説します

全体構成図と流れは以下のようになります



上図 SVG

1. BX1 の Wi-Fi AP 設定と SORACOM Air(3G ネットワーク) 設定

2. センサーと BX1 の接続
3. Amazon Elasticsearch Service のインスタンス作成と設定
4. AWS IoT の設定
5. BX1 と AWS IoT の接続
6. 片付け&まとめ
7. 自習室: SORACOM Beam で AWS IoT の認証処理をオフロード
8. 自習室: AWS IoT の Thing Shadow でパトライトを制御する

テキストの PDF は [こちら](#) (5.5MB P74)

ハンズオンの前に

対象者 (前提条件・スキル等)

- AWS Management Console の基本的な操作がわかる
- シリアルコンソールとターミナル等による機器操作スキル (FTDI シリアルドライバのインストール等含む)
- AWS IoT の概要を理解している (AWS IoT ハンズオン ~基本編~」を参考にしますと、より実践的な内容を理解することが出来ます。)

AWS のアカウントをお持ちでない方は

Amazon Web Services をご利用いただくために、事前のアカウント取得をお願いいたします

アカウントの取得は [こちら](#)

<http://aws.amazon.com/jp/register-flow/>

アカウント取得はクレジットカードの番号入力が必要となります
アカウントを取得しただけでは料金はかかりません

ソラコムアカウント取得・SORACOM Air のアクティベートが済んでいない方は

SORACOM Air を使用するため、事前に SORACOM アカウントの取得並びに SIM のアクティベートを完了しておいていただく必要があります

アカウントの取得と SIM のアクティベートは [こちら](#)

<https://dev.soracom.io/jp/start/console/>

アクティベートにはクレジットカードの番号入力が必要となります (アカウント登録には必要ありません)

部材等の確認

準備するものをご確認ください(不足している場合はチューターにお声がけください)

- 支給されるもの
 - 持ち帰り OK
 - * SORACOM Air <miniSIM サイズ> x 1 枚
 - 貸し出し物 (返却いただきます)
 - * OpenBlocks IoT BX1 x 1 ケ
 - * BX1 用 電源兼シリアルコンソール USB ケーブル x 1 本
 - * センサーデバイス FWM8BLZ02 (電池入り) x 1 ケ
- 用意いただくもの <必須>
 - ノート PC
 - * Wi-Fi 接続可
 - * USB ポート使用可 (A タイプ 1 ポート以上)
 - * FTDI シリアルドライバーインストール済
 - * シリアルコンソール制御用ターミナルソフトインストール済 (e.g. TeraTerm, GNU screen)
 - * curl や wget など HTTP コールが可能なツール
 - * Chrome 等のモダンブラウザ (Internet Explorer は NG ですご注意ください)
 - 参加者個人の AWS アカウント
 - 有効なクレジットカード (SORACOM Air のアクティベーションに使用)
- あると便利なもの
 - スマートフォン (本ドキュメントを参照しながらの作業)

注釈: curl 等のツール類は BX1 の中に入っていますので、それを利用することも可能です (ただし BX1 から利用する場合は 3G 回線費用が別途かかる可能性が有ります。その他、EC2 インスタンスを用意して、そちらで実施することも可能です)

注意事項

- 章毎に達成状況を確認しながら進めていきますが、時間の都合上未達者がいても次に進む場合がありますのでご了承ください
- AWS や SORACOM 等、ハンズオンで発生する費用については参加者の自己負担となります
- 不明な点はお気軽にチューターへご相談ください

ソラコム の料金詳細は [SORACOM Air のご利用料金](#) をご覧ください

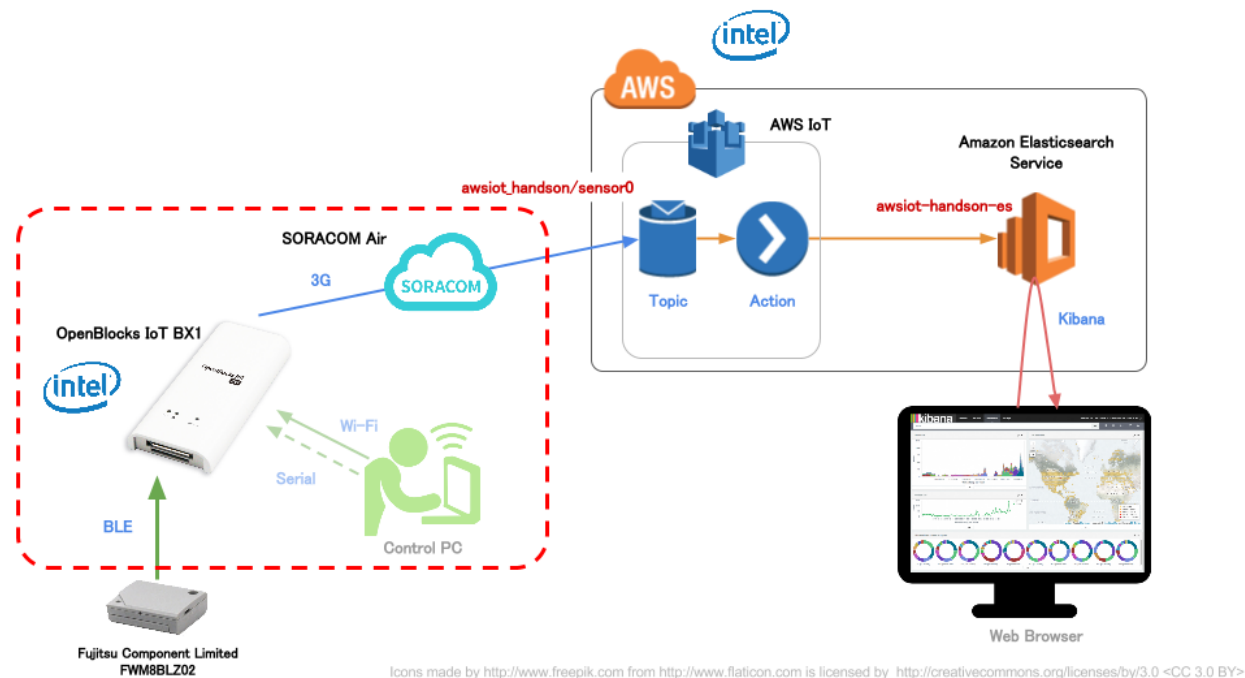
SORACOM Air には無料分利用枠があり、本ハンズオンでは無料枠内で収まる程度の通信となる見込みですが、その限りではありませんのでご承知ください

[BX1 の Wi-Fi AP 設定と SORACOM Air\(3G ネットワーク\) 設定 へ進む](#)

BX1 の Wi-Fi AP 設定と SORACOM Air(3G ネットワーク) 設定

本章のゴール: ping を WebUI 上から行い、3G 回線がつながっていることを確認する

作業の位置づけ;



上図 SVG

BX1 の取り扱い (電源 ON/OFF ・再起動、SIM 挿入)

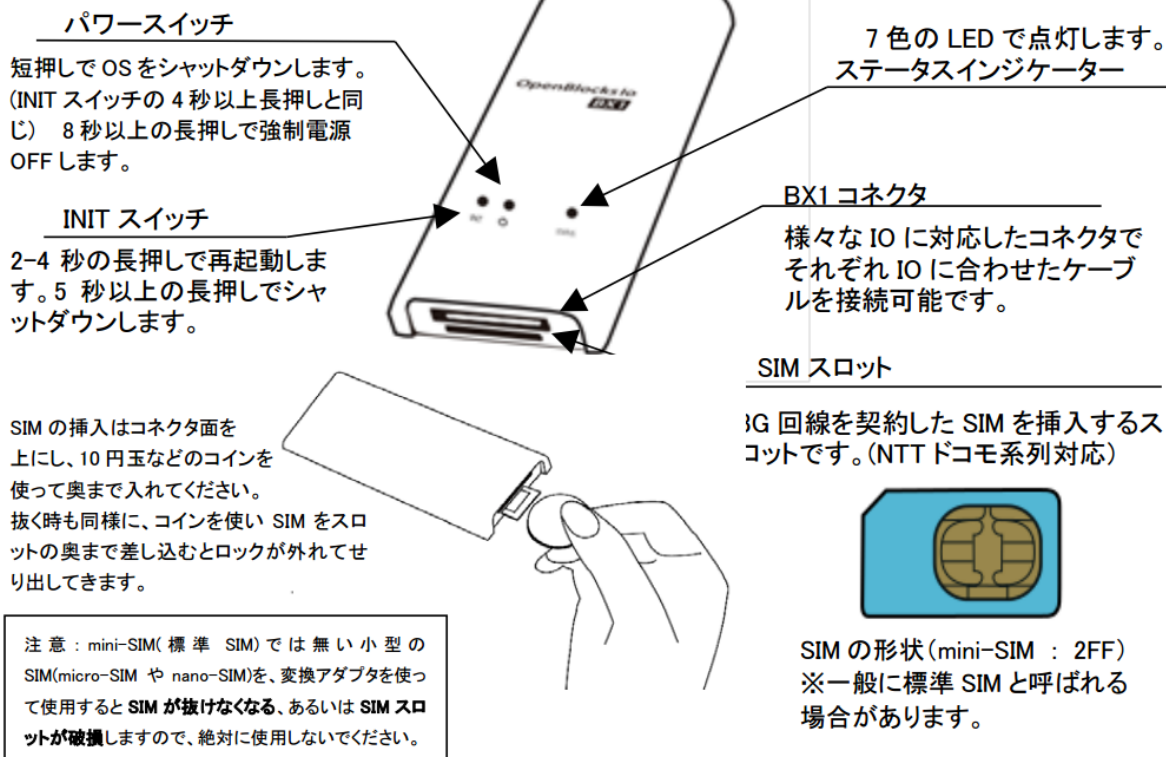
BX1 は給電開始と共に電源 ON となります。USB ケーブルを接続すると起動を開始します
起動完了 (使用できるようになるまで) に 4 分ほどかかります

注釈:

- INIT ボタンや電源ボタンは OS のシャットダウンと再起動に利用されますので、必要時以外は押さないでください
- SIM は電源 ON 前に挿すようにしてください。起動後に挿しても認識しません

部位の名前や SIM 挿入方法については、下記を参照してください

OpenBlocks IoT BX1 本体



(OpenBlocks IoT Family WEB UI セットアップガイド P7 より抜粋)

ステータスインジケータ (LED の表示色)

BX1 は *STATUS* の LED にて状態を把握することができます

本ハンズオンにおいては、起動後は白色、水色、青色が望ましく、それ以外の色の場合は不具合がある可能性がありますので、チューターにご相談ください

具体的な LED 色と状態については、下記を参照してください

状態	色	点灯状態	備考
OS 起動中	黄	点灯 ↓ 消灯 ↓ 点滅	OS 起動が終わるとモバイル回線電波受信チェックへ移行します。 ※モバイル回線が起動できない時は緑点灯。
3G/LTE 未使用での運用	緑	点滅	SIM がない状態での正常稼働状態
3G/LTE 電波(強)	白	点滅	極めて電波強度が良好です。 (-87dbm 以上)
3G/LTE 電波(中)	水色	点滅	通信には問題ないレベルです。 (-88dbm ~ -108dbm)
3G/LTE 電波(弱)	青	点滅	この青色表示は極めて通信エラーが起こりやすいので可能であれば水色の電波強度まで設置位置を変えてください。 (-109dbm ~ -112dbm)
3G/LTE 圏外	紫	点滅	通信不可能です。(-113dbm 以下)
INIT ボタンを押下(リブート)	黄	点灯	OS リブート。
電源ボタンを押下 シャットダウン電源 OFF	赤	点灯	LED が消灯するまで長押しが必要

(OpenBlocks IoT Family WEB UI セットアップガイド P7 より抜粋)

Web 管理画面 (WebUI) へのログイン

警告:

- BX1 に接続した PC やスマートフォンは、BX1 の Wi-Fi に接続している限り、後ほど設定する SORACOM Air の設定が終わるまではインターネットに接続することが出来ません
- また、SORACOM Air の設定が完了すると、インターネットへのアクセスは “PC =[Wi-Fi]=> BX1 =[SORACOM Air]=> インターネット” という経路になり、通信料が発生する可能性がありますので、Dropbox 等の共有ソフトの動作を OFF にすることを強く推奨いたします

BX1 と Wi-Fi で接続する

BX1 の起動が完了すると、BX1 は Wi-Fi のアクセスポイントとして動作を開始します

お手持ちの PC やスマートフォンから SSID を探し、接続してください

- SSID: `iotfamily_BX1` シリアル番号
- Password: `openblocks`

802.11g, WPA-PSK の設定で接続できます

お手持ちの **BX1** のシリアル番号確認方法

シリアル番号は、BX1 のウラ面のバーコード上の文字と数字の組み合わせになります

下記例では **F2A00788** がシリアル番号です (そのため SSID は **iotfamily_F2A00788** となります)



WebUI を表示する

BX1 に Wi-Fi で接続できたら、下記 URL にて WebUI を表示します

<http://192.168.254.254:880>

下記の画面が出れば正常に接続ができています



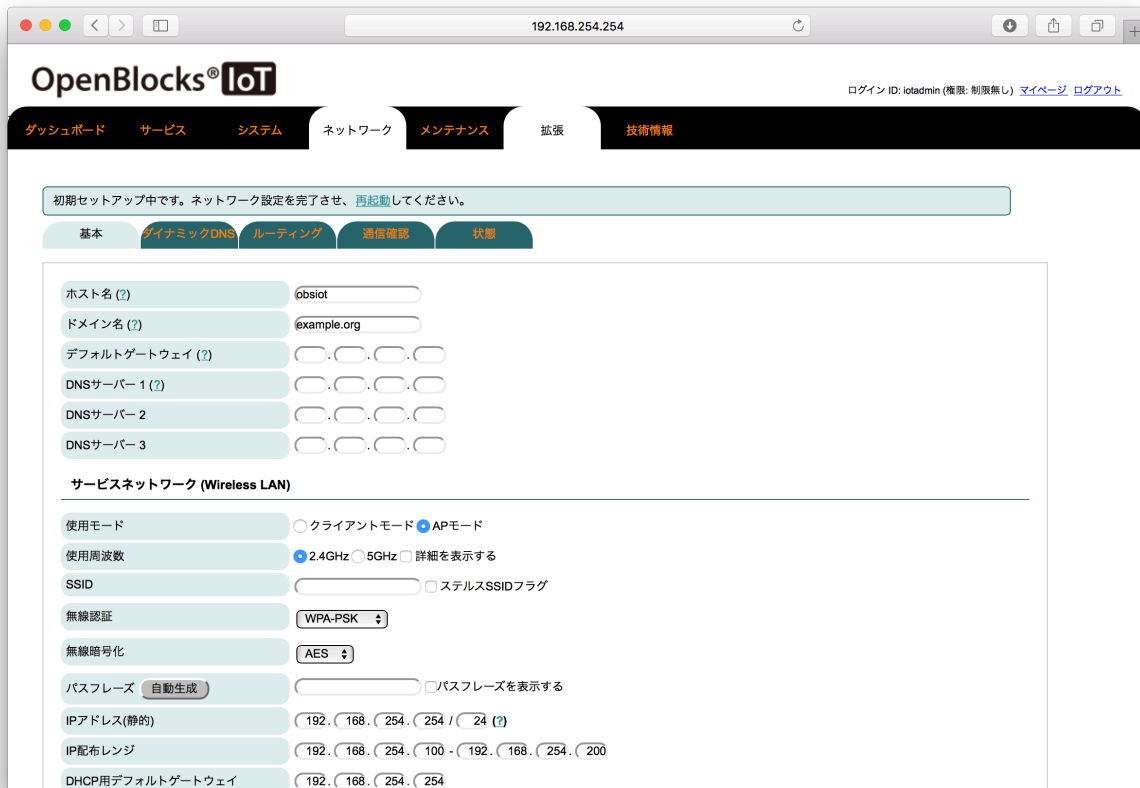
使用許諾の確認と管理ユーザアカウントの設定

使用許諾の“同意する”ボタンを押すと、WebUI 管理用アカウントの作成を求められます

下記にて設定願います（本来は複雑な組み合わせが好ましいのですが、ハンズオン時のトラブルシュート対策です）

ID	iot (アイ オー ティー)
Password	iot (アイ オー ティー)

成功すると、ネットワーク設定画面が表示されます



Wi-Fi AP 設定と SORACOM Air(3G ネットワーク) 設定

ネットワーク設定画面を開きます

(前項から正しく遷移していれば表示されているはずですが、そうでない場合は [ネットワーク] - [基本] を選択するようにしてください)

ネットワーク設定画面では各項目をそれぞれ下記のとおり設定してください

ホスト名	<シリアル番号> (例 F2A00788)
ドメイン	openblocks.local
Wi-Fi / 使用チャンネル	<シリアル番号 最後の値> (例:F2A00788 なら “8”) (0の方は10 / A~Eの方は11)
Wi-Fi / SSID	iotfamily_<シリアル番号> (例: iotfamily_F2A00788)
Wi-Fi / 無線認証	WPA-PSK
Wi-Fi / 無線暗号化	AES
Wi-Fi / パスフレーズ	openblocks
モバイル回線 / APN	soracom.io
モバイル回線 / ユーザ名	sora
モバイル回線 / パスワード	sora

注釈:

- Wi-Fi のチャンネル設定は 詳細を表示する をチェックすることで表示されます

- モバイル回線の設定は サービスネットワーク (モバイル回線) を使用する にチェックすることで表示されます

警告:

- 無線認証を WPA2-PSK に変更すると、OS によっては再接続ができなくなる事があります

再起動

APN の設定は再起動で反映されるため、BX1 を再起動します

1. WebUI から [メンテナンス] - [停止・再起動] を表示
2. 再起動を実施します (最後に確認ダイアログがあるので見逃さないようにしてください)

注釈: 再起動は 5 分程度かかります



ping で確認

再起動が無事終了すれば BX1 は 3G ネットワークに接続された状態となっています

WebUI 上から ping を発信して確認してみます

1. WebUI にログインした後 [ネットワーク] - [通信確認]

2. 宛先ホスト: metadata.soracom.io



OpenBlocks IoT

ログイン ID: iotadmin (権限: 制限無し) [マイページ](#) [ログアウト](#)

ダッシュボード サービス システム ネットワーク メンテナンス 拡張 技術情報

基本 ダイナミックDNS ルーティング 通信確認 状態

通信確認

宛先ホスト

コマンド

```
PING metadata.soracom.io (100.127.100.127) 56(84) bytes of data:
64 bytes from 100.127.100.127: icmp_req=1 ttl=64 time=106 ms
64 bytes from 100.127.100.127: icmp_req=2 ttl=64 time=105 ms
64 bytes from 100.127.100.127: icmp_req=3 ttl=64 time=104 ms
64 bytes from 100.127.100.127: icmp_req=4 ttl=64 time=104 ms
64 bytes from 100.127.100.127: icmp_req=5 ttl=64 time=103 ms

--- metadata.soracom.io ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 103.244/104.674/106.108/1.007 ms
```

Version 1.0.7

© 2015 - 2016 PlatHome Co., Ltd. All rights reserved.

ここまで到達できればゴールです

[センサーと BX1 の接続 へ進む](#)

トラブルシュート

1. 当該 SIM のアクティベーションは済んでいますか？
2. モバイル回線は接続できていますか？

モバイル回線の接続状況確認と接続方法

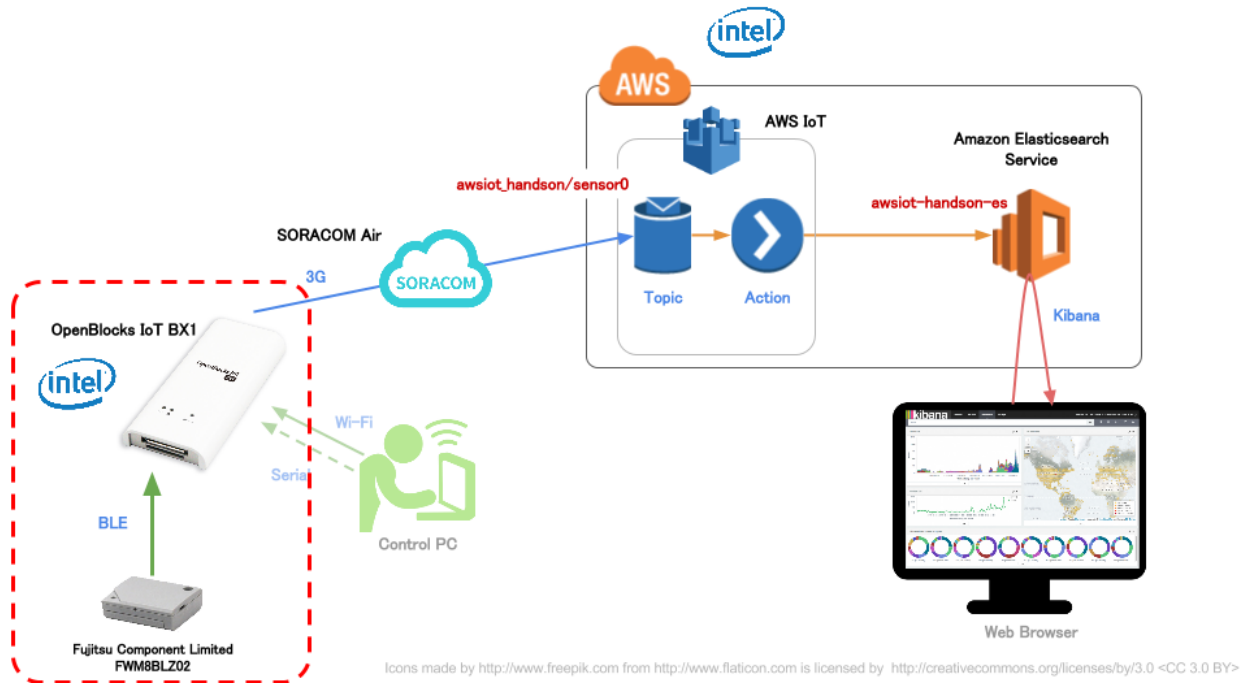
WebUI の “ダッシュボード” にてモバイル回線の接続状況確認と接続作業を行うことができます

The screenshot shows the OpenBlocks IoT WebUI dashboard. The browser address bar displays '192.168.254.254:880'. The page title is 'OpenBlocks IoT' with a logo. A navigation menu includes 'ダッシュボード', 'サービス', 'システム', 'ネットワーク', 'メンテナンス', '拡張', and '技術情報'. The main content area is titled 'システム全体の概要' (System Overview) with a '更新' (Refresh) button. It is divided into three sections: 'ハードウェアリソース' (Hardware Resources) showing 'メインメモリ: 338 MB / 961 MB' and 'ストレージ: 621 MB / 2283 MB'; 'ネットワーク (設定)' (Network (Settings)) showing 'FQDN: F2A00788.openblocks.local', 'IPアドレス (wlan0): 192.168.254.254', and 'モバイル回線状況: 未接続(電波: 中)' (Mobile network status: Not connected (Signal: Medium)) with a '接続' (Connect) button. The footer shows 'Version 1.0.7' and a copyright notice: '(C) 2015 - 2016 PlatHome Co., Ltd. All rights reserved.'

センサーと BX1 の接続

本章のゴール: センサーデバイス FWM8BLZ02(以下、センサー) の温度データを BX1 で受信し、WebUI 上でグラフ化される

作業の位置づけ;

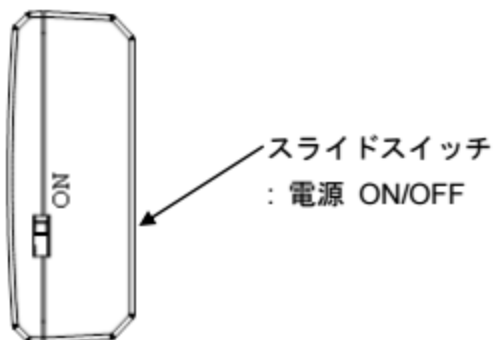


上図 SVG

センサーを **BX1** に登録する

センサーの電源を **ON** にする

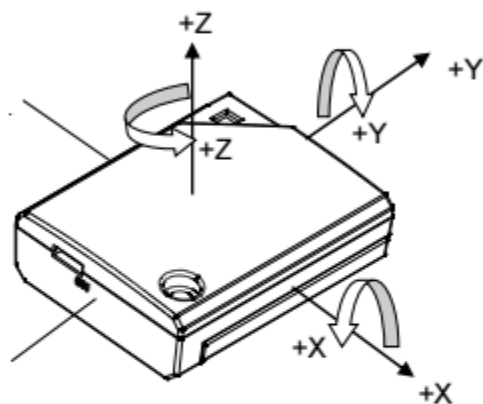
センサー側面の電源スイッチを ON にしてください



センサーの仕様

5-9-1-1. 軸の方向性と回転の極性

軸の方向性と回転の極性を下記に示します。



5-9-2. 温度センサ

Items	Symbol	Min	Typ	Max	Unit
動作温度	Ta	-30	25	+60	°C

Bluetooth の使用を開始する

1. WebUI から [サービス] - [基本] を表示
2. Bluetooth の使用設定を使用する にして [保存]

The screenshot shows a web browser window titled "OpenBlocks IoT - Mozilla Firefox" with the URL "192.168.254.254:880/apps/m2miot/basic.php". The page header includes the "OpenBlocks IoT" logo and a login ID: "webuiadm (権限: 制限無し) [マイページ](#) [ログアウト](#)". The main navigation menu contains "ダッシュボード", "サービス", "システム", "ネットワーク", "メンテナンス", "拡張", and "技術情報". The "サービス" menu is expanded to show "基本", "Bluetooth関連", "BLEメンテナンス", and "状態". The "Bluetooth" settings page is displayed, featuring sections for "Bluetooth", "データ収集", and "操作". Under "Bluetooth", the "使用設定" (Usage Settings) section has a radio button for "使用する" (checked) and "使用しない" (unchecked). Under "データ収集" (Data Collection), there are two sections: "データ収集" with radio buttons for "使用する" (unchecked) and "使用しない" (checked), and "PD Handler" with radio buttons for "使用する" (unchecked) and "使用しない" (checked). A "保存" (Save) button is located at the bottom of the form. The footer of the page shows "Version 1.0.6" and a copyright notice: "(C) 2015 PlatHome Co., Ltd. All rights reserved."

センサーを検出して登録する

1. WebUI から [サービス] - [Bluetooth 関連] を表示
2. **Bluetooth LE** デバイス検出の [検出] をクリック
3. 一覧の中から自分のデバイスを探し 使用設定 にチェックをして [保存]

警告:

- Bluetooth デバイス検出 ではなく **Bluetooth LE** デバイス検出 を押すようにしてください

The screenshot shows the OpenBlocks IoT web interface in a Mozilla Firefox browser. The page title is "OpenBlocks IoT" and the URL is "192.168.254.254:880/apps/m2miot/bluetooth.php". The interface includes a navigation menu with options like "ダッシュボード", "サービス", "システム", "ネットワーク", "メンテナンス", "拡張", and "技術情報". The main content area is titled "Bluetooth(2)" and contains a "Bluetoothデバイス検出" section with a "検出" button. Below this, there is a "Bluetooth LEデバイス検出時間(秒)" input field set to "15" and another "Bluetooth LEデバイス検出" section with a "検出" button and a "検出終了" status. A table displays the discovered devices with columns for "使用設定" (Usage Setting), "項目" (Item), and "内容" (Content).

使用設定	項目	内容
<input type="checkbox"/>	Device Name	s58
	Device Address	D5:01:AA:82:C5:FD
	Memo	
<input checked="" type="checkbox"/>	Device Name	SensorTag
	Device Address	5C:31:3E:C0:2E:E6
	Memo	
<input type="checkbox"/>	Device Name	(unknown)
	Device Address	75:9B:00:94:EF:E8
	Memo	
<input type="checkbox"/>	Device Name	FWM8BLZ02
	Device Address	EF:D4:43:9F:7E:DB
	Memo	
<input type="checkbox"/>	Device Name	FCLWirelessModule
	Device Address	E9:DF:F2:80:EB:9A
	Memo	

自分のデバイスの探し方

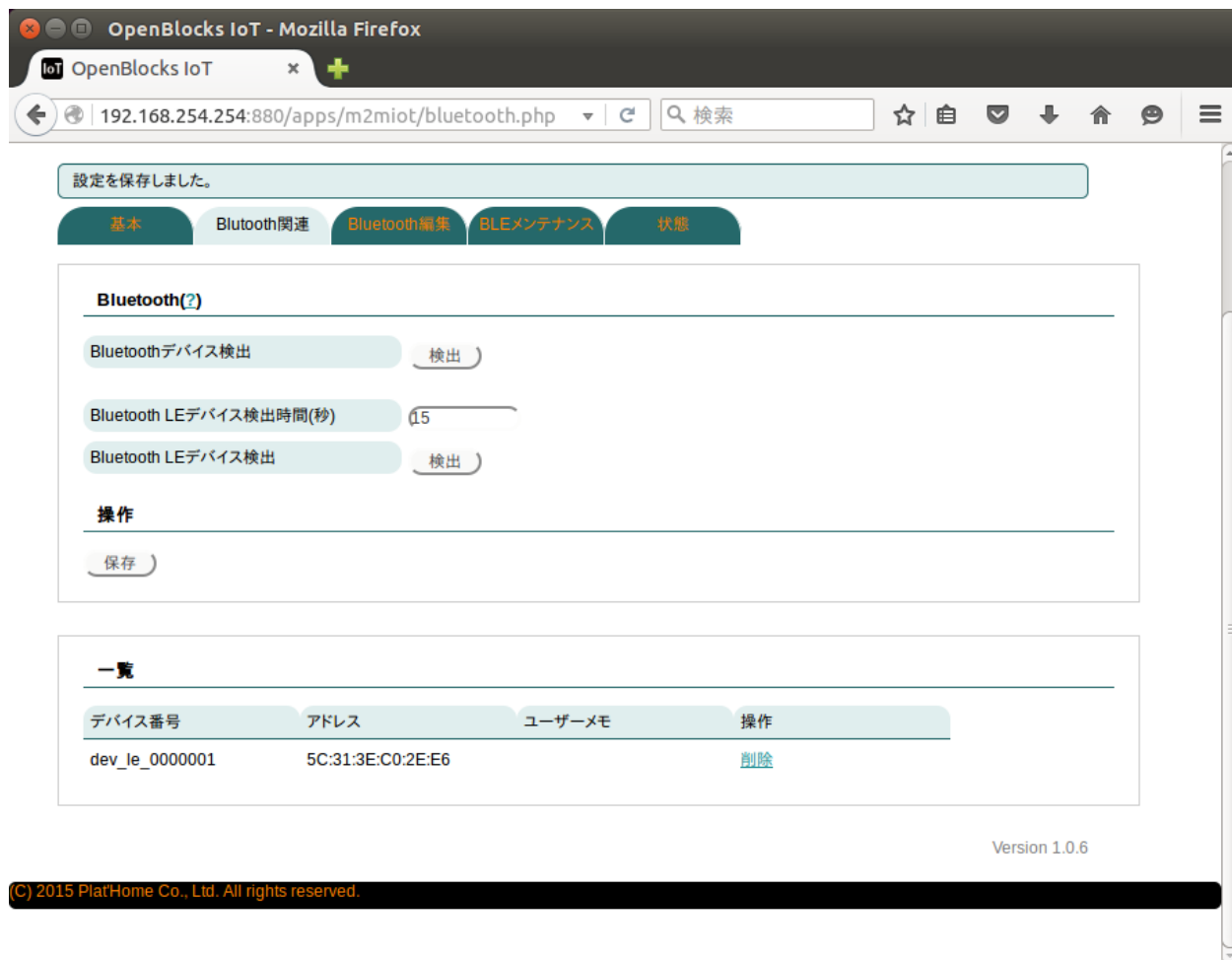
Device Name = FWM8BLZ02 が大量に表示される可能性があります

お配りしたセンサーにはアドレスのタグが書かれており、画面上に表示されている *Device Address* (= MAC Address) と一致したものが、ご自分のセンサーとなりますので確認してください



登録状況の確認

保存すると WebUI は下記のようになります



設定を保存しました。

基本 Bluetooth関連 Bluetooth編集 BLEメンテナンス 状態

Bluetooth(2)

Bluetoothデバイス検出

Bluetooth LEデバイス検出時間(秒)

Bluetooth LEデバイス検出

操作

一覧

デバイス番号	アドレス	ユーザーメモ	操作
dev_le_0000001	5C:31:3E:C0:2E:E6		削除

Version 1.0.6

(C) 2015 PlatHome Co., Ltd. All rights reserved.

以上でセンサーを BX1 に登録できました

BX1 のデータ収集設定

BX1 のデータ収集機能を開始する

1. WebUI から [サービス] - [基本] を表示
2. データ収集 における データ収集 ならびに PD Handler BLE をそれぞれ使用する にして [保存]

注釈:

- “PD Handler” は “データ収集” を <使用する> にすると表示されます

The screenshot shows a web browser window titled "OpenBlocks IoT - Mozilla Firefox" with the URL "192.168.254.254:880/apps/m2miot/basic.php". The page header includes the "OpenBlocks IoT" logo and a user login ID: "webuiadm (権限: 制限無し) [マイページ](#) [ログアウト](#)". A navigation menu contains "ダッシュボード", "サービス", "システム", "ネットワーク", "メンテナンス", "拡張", and "技術情報". Below this, a sub-menu highlights "Bluetooth関連", "Bluetooth編集", "BLEメンテナンス", and "状態". The main content area is titled "Bluetooth" and contains several settings sections:

- Bluetooth**
 - 使用設定: 使用する 使用しない
- データ収集**
 - データ収集: 使用する 使用しない
 - PD Handler: 使用する 使用しない
 - PD自動再起動設定: 使用する 使用しない
- 操作**
 - 保存

Version 1.0.6

(C) 2015 PlatHome Co., Ltd. All rights reserved.

注釈:

- データ収集を開始すると、新たなデバイスの登録ができなくなります
- デバイス登録をする場合は、まず、データ収集を行わないように設定を変更してください

BX1 内へのデータ収集機能を開始する

1. WebUI から [サービス] - [収集設定] を表示
2. 本体内 (local) を使用する

OpenBlocks IoT - Mozilla Firefox

OpenBlocks IoT

192.168.254.254:880/apps/m2miot/set_collect.php

OpenBlocks® IoT

ログイン ID: webuiadm (権限: 制限無し) [マイページ](#) [ログアウト](#)

ダッシュボード サービス システム ネットワーク メンテナンス 拡張 技術情報

基本 収集設定 収集ログ Bluetooth関連 Bluetooth編集 BLEメンテナンス 状態

送信先設定

本体内(local) 使用する 使用しない

デバイス一括設定 一括有効 一括無効

PD Exchange 使用する 使用しない

Amazon Kinesis 使用する 使用しない

AWS IoT 使用する 使用しない

IBM Bluemix 使用する 使用しない

MQTTサーバ 使用する 使用しない

WEBサーバ(PLAIN) 使用する 使用しない

ビーコン送信設定(?)

送信対象 送信する 送信しない

デバイス情報送信設定 送信対象一括有効 送信対象一括無効

ページ後半に移動し `dev_le_0000001` の設定を下記のようにします

送信対象	送信する
取得時間間隔 (ms)	1000
送信先設定	local のみにチェック

以上を確認し [保存]

PD Exchange 使用する 使用しない

Amazon Kinesis 使用する 使用しない

AWS IoT 使用する 使用しない

IBM Bluemix 使用する 使用しない

MQTTサーバ 使用する 使用しない

WEBサーバ(PLAIN) 使用する 使用しない

ビーコン送信設定(?)

送信対象 送信する 送信しない

デバイス情報送信設定 送信対象一括有効 送信対象一括無効

デバイス番号 dev_le_0000001

送信対象 送信する 送信しない

アドレス E7:3A:40:83:9D:8A

ユーザーメモ

センサー信号強度[dbm] 0

取得時間間隔[ms] 1000

送信先設定 local PD KINESIS AWSIOT BLUEMIX MQTT PLAIN

操作

保存

Version 1.0.7

© 2015 - 2016 PlatHome Co., Ltd. All rights reserved.

グラフの表示

local へのデータ収集が開始されると、WebUI 内のグラフに表示が開始されます

[サービス] - [データ表示] で、[グラフ表示] の [表示する] を選ぶとグラフで確認できます
 グラフが表示されるまで多少時間が必要です。グラフが表示されない場合は少し待ってから [再描画] をクリックしてください



ここまで到達できればゴールです

[Amazon Elasticsearch Service のインスタンス作成と設定 へ進む](#)

トラブルシューティング

データ収集状況ログの確認

WebUI から [サービス] - [収集ログ] にて、動作確認が可能です

ログ選択はそれぞれ下記のとおりです

pd-handler-stdout.log	センサー <-> BX1 間の送受信状況
pd-emitter.log	BX1 <-> 送信先 (local や AWS IoT 等) 間の送受信状況

下記画面はセンサーからのデータ読み出しが成功している場合のログ画面です。JSON が表示されているのが見てわかります

pd-handler-stdout.log に `timeout: ...` などと表示されている

センサーとの BLE 接続確立に失敗している可能性があります

1. しばらく待つ (再接続するため)
2. データ収集プロセスを再起動する (チューターにご相談ください)

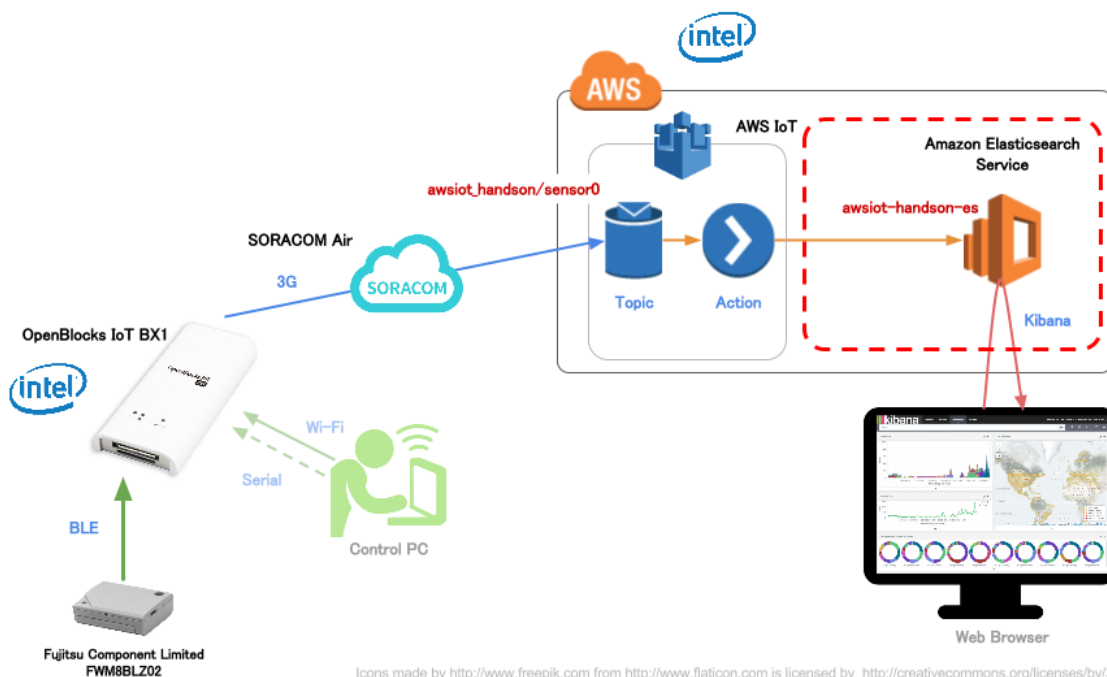
これらで対処可能です

Amazon Elasticsearch Service のインスタンス作成と設定

本章のゴール: Amazon Elasticsearch Service 上の Kibana でテストデータのグラフが表示される

以下、文中では Amazon Elasticsearch Service を以下 Amazon ES と略します

作業の位置づけ;



上図 SVG

Amazon ES のクラスタを作成

Get started (もしくはダッシュボードの Create a new domain) から Amazon ES のインスタンス作成のウィザードを開始します

それぞれのステップでは、下記項目を変更するようにしてください(それ以外の項目はデフォルト値で OK です)

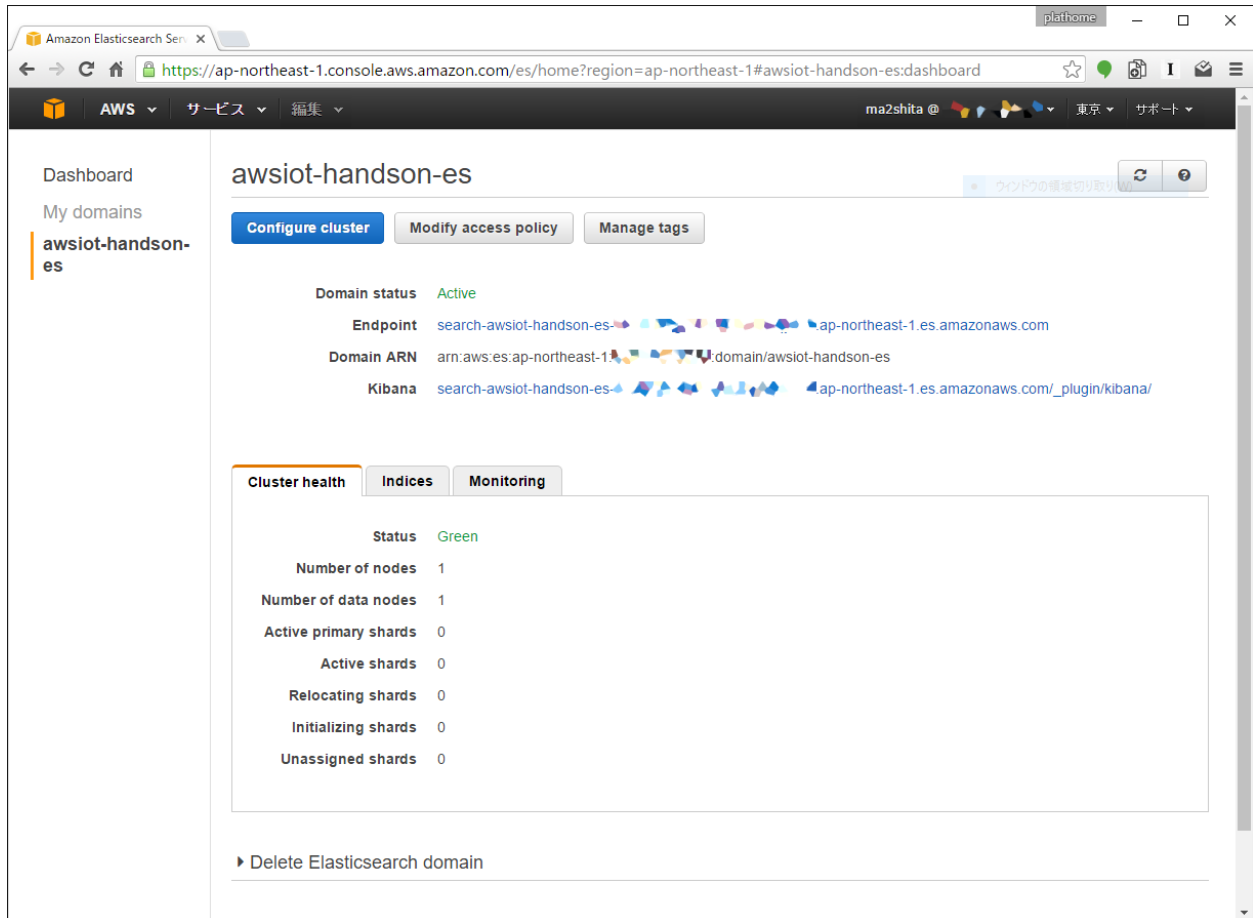
Elasticsearch domain name	awsiot-handson-es
> [Next] を押して次のステップへ	
Instance type	t2.micro.elasticsearch
Storage type	EBS
> [Next] を押して次のステップへ	
Set the domain access policy to	Allow open access to the domain
> [Next] を押すと確認画面がでるので、確認後 [Confirm and create] で作成します	

警告: 今回は時間の関係上 “Allow open access to the domain” (=すべてのアクセスを許可) としています。本番運用時にはアクセスコントロールを行うようにしてください

注釈: Amazon ES が使えるようになるまで 10 分程度かかるため、次章の作業 (AWS IoT の設定) を行うとよいでしょう

Elasticsearch の Endpoint と Kibana の URL を確認する

Amazon ES のダッシュボードから ES の Endpoint と Kibana の URL を確認します



テストデータの投入と表示

Amazon ES が利用可能になったら、動作確認を行います

データの投入

注釈:

- ここでは確認のために `curl` コマンドを使用します。準備願います (Windows の方向け)
- `date` コマンドが下記の通り使えない場合は `2016-05-19T10:10:50+0900` というフォーマットの日付文字列に置き換えてください
- `${YOUR_ES_ENDPOINT}` は、Amazon ES のダッシュボードで確認した Endpoint に置き換えてください

```
curl -X PUT ${YOUR_ES_ENDPOINT}/es-test/es0/1 -d '{"deviceId":"es-test0","field1":1,"timestamp":1,"@timestamp":1}'
```

Windows の方: 下記の通りダブルクォーテーションのエスケープを変更してください

```
curl -X PUT %YOUR_ES_ENDPOINT%/es-test/es0/1 -d '{"\"deviceId\": \"es-test0\", \"field1\": 1, \"timestamp\": 1, \"@timestamp\": 1}'
```

コマンドの投入結果は下記ようになります

```
{"_index": "es-test", "_type": "es0", "_id": "1", "_version": 1, "created": true}
```

Kibana での表示

Index を作成する

Amazon ES のダッシュボードで得た Kibana の URL にアクセスします
インデックスの設定を下記のとおりにします

Index name or pattern	es-test
Time-field name	@timestamp

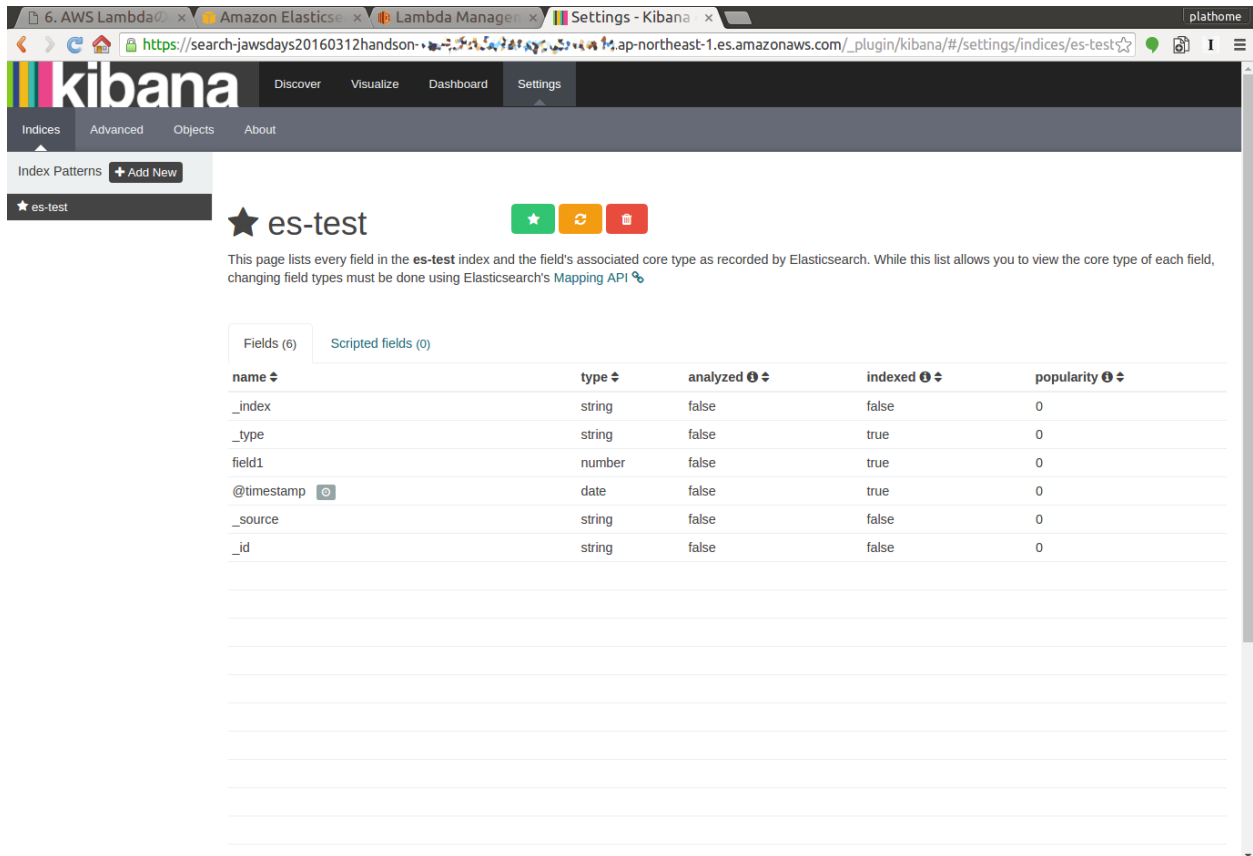
The screenshot shows the Kibana 'Configure an index pattern' interface. It includes a warning message: 'Warning No default index pattern. You must select or create one to continue.' The configuration options are as follows:

- Index contains time-based events
- Use event times to create index names
- Index name or pattern: es-test
- Time-field name: @timestamp

A green 'Create' button is located at the bottom of the configuration form.

テストデータが正しく投入されていれば“Create”ボタンが押せるようになるはずですので、押してください
下記の通り、インデックスのカラム一覧が表示されれば成功です

そうでない場合、テストデータの投入に失敗している可能性があります。コマンドの実行結果等を確認してください



The screenshot shows the Kibana Settings page for the 'es-test' index. The page displays a table of fields with their associated core types and properties. The table has columns for name, type, analyzed, indexed, and popularity. The fields listed are: _index (string), _type (string), field1 (number), @timestamp (date), _source (string), and _id (string).

name	type	analyzed	indexed	popularity
_index	string	false	false	0
_type	string	false	true	0
field1	number	false	true	0
@timestamp	date	false	true	0
_source	string	false	false	0
_id	string	false	false	0

データを表示する

“Discover” をクリックするとデータの中身を表示することができます



ここまで到達できればゴールです

[AWS IoT の設定 へ進む](#)

トラブルシューティング

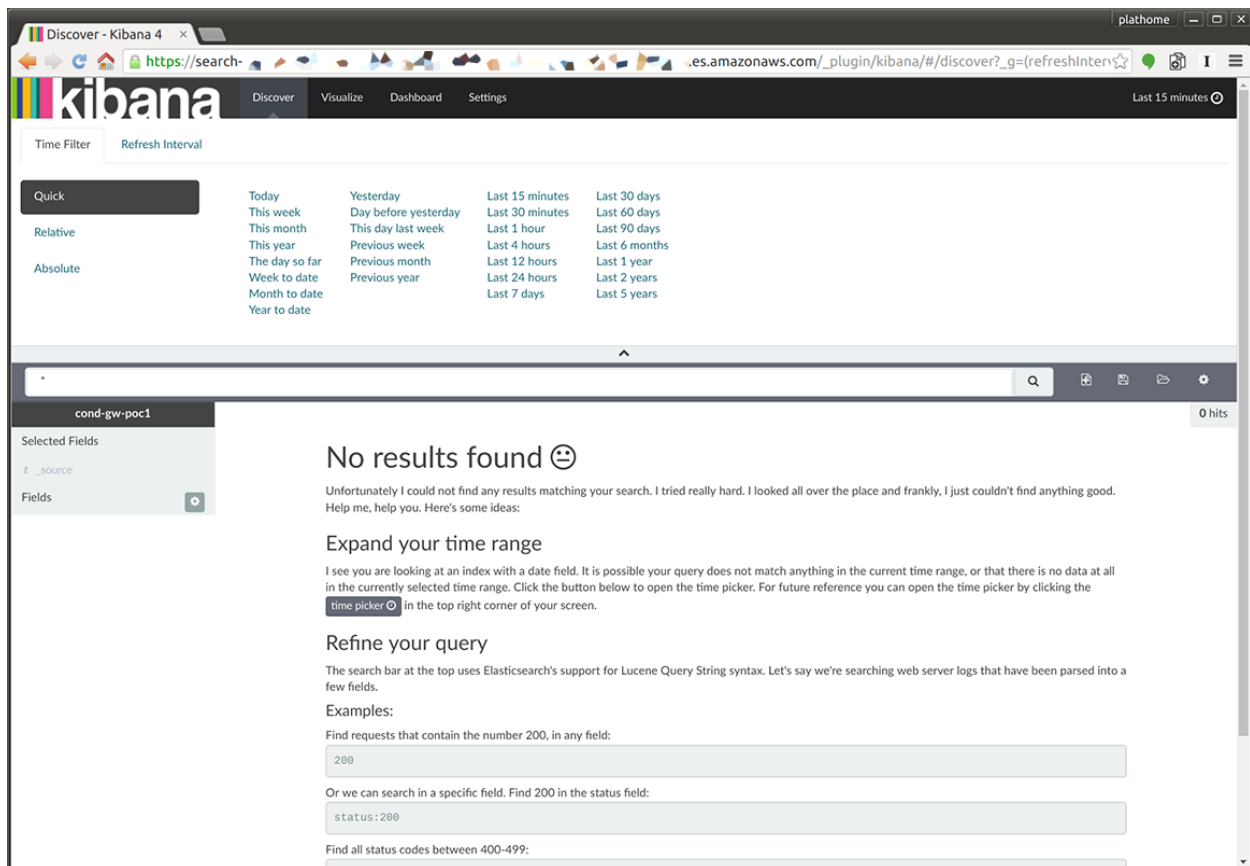
テストデータの投入に失敗した (ようなので) インデックスを削除する

```
$ curl -X DELETE ${YOUR_ES_ENDPOINT}/es-test
```

データ投入に成功した (はず) が、ダッシュボードにデータが表示されない

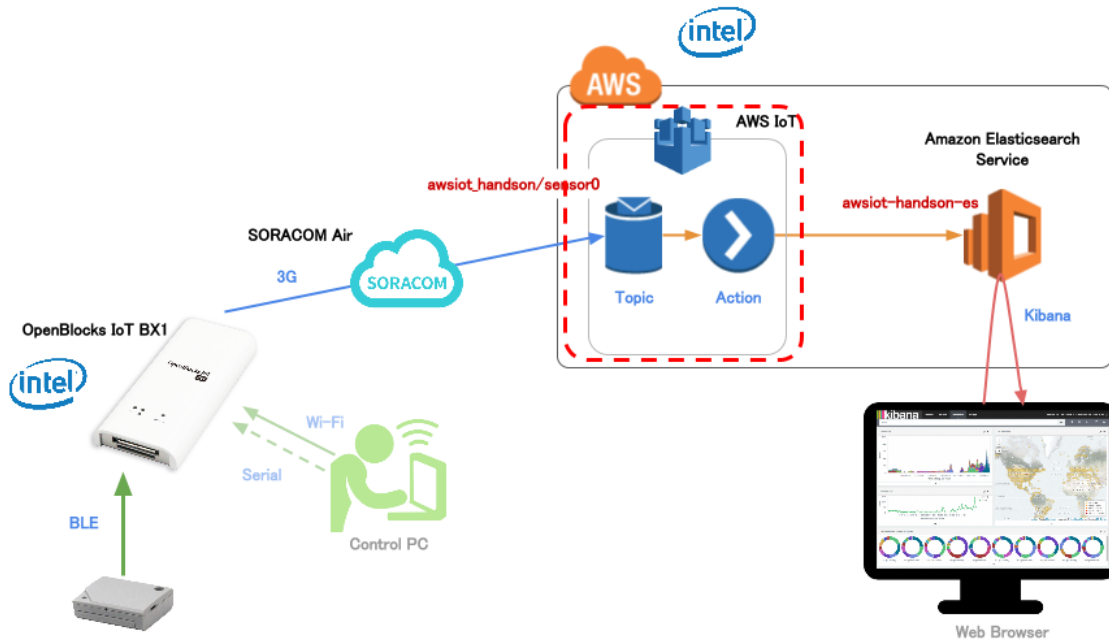
データ表示期間の範囲が適切でない場合があります

右上の “Time Picker” のアイコンから “Time Filter” で、例えば “Last 30 Days” に設定してみてください



AWS IoT の設定

本章のゴール: AWS IoT で MQTT(AWS IoT 上のテストサイト) による受信ができることを確認する作業の位置づけ;



Icons made by <http://www.freepik.com> from <http://www.flaticon.com> is licensed by <http://creativecommons.org/licenses/by/3.0> <CC 3.0 BY>

上図 SVG

概要

AWS IoT の設定は若干ステップが多いため、まずステップの概要を確認します

1. IAM ロールの作成
2. AWS IoT ポリシの作成
3. “モノ” の作成
4. 証明書の作成, ポリシの割り当て, “モノ” の割り当て
5. ルールの作成

IAM ロールの作成

AWS IoT のルールを作成する前に、AWS IoT から Amazon Elasticsearch Service へデータ送信するための権限ルールを作成します

ロールの作成

IAM コンソールのロール一覧から [新しいロールの作成] をクリックし、ウィザードを開始します

ウィザードの各項目は下記のようにしてください

ロール名	awsiot_handson_put_to_es
ロールタイプの選択	AWS サービスロール => AWS IoT を選択
ポリシーのアタッチ	<なにも選択せず>

インラインポリシーの設定

1. IAM コンソールのローラー一覧から、先ほど作成した `awsiot_handson_put_to_es` ロールを選択
2. [インラインポリシー] - [ここをクリックしてください] をクリック
3. 次の画面では [カスタムポリシー] - [選択] をクリック
4. 下記の通りポリシー名とポリシードキュメントを設定し [ポリシーの検証] をした後 [ポリシーの適用] をクリック

ポリシー名	<code>awsiot_handson_put_to_es_adhoc_policy</code>
-------	--

ポリシードキュメント

`AWS_ACCOUNT_ID` は各自の AWS AccountID に読み替えるようにしてください

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "es:ESHttpPut",
    "Resource": [
      "arn:aws:es:ap-northeast-1:AWS_ACCOUNT_ID:domain:awsiot-handson-es/*"
    ]
  }
}
```

以上でロールの作成は完了です _

AWS IoT ポリシの作成

“モノ” が AWS IoT にアクセスする際の制限を設定することができます
今回はフルコントロールとします

注釈: IAM のポリシーとは別のものになります

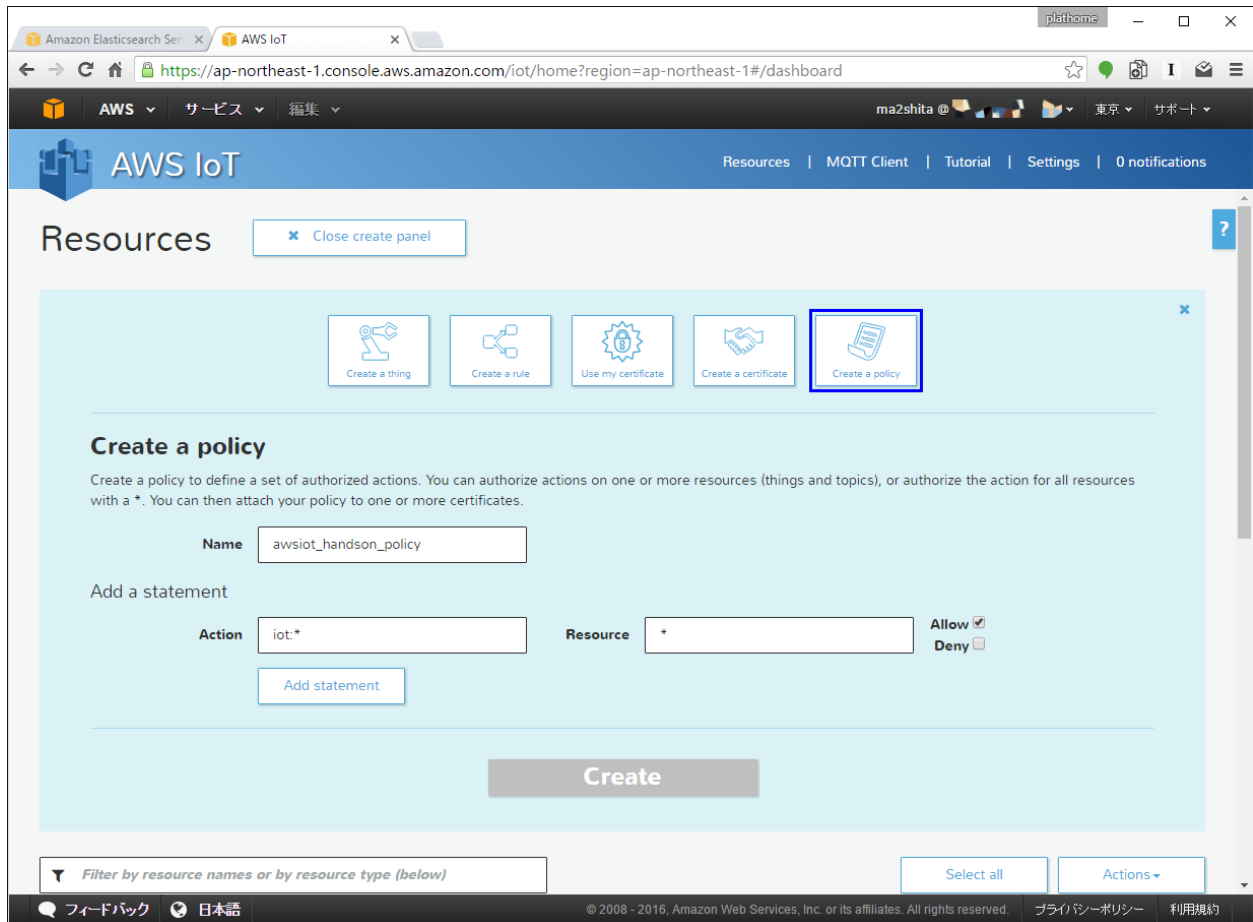
作業

1. AWS IoT のコンソールから “Create a resource” をクリックし “Create a policy” をクリック
2. 下記を入力したら “Add statement” をクリックし “Create” をクリック

Name	<code>awsiot_handson_policy</code>
Action	<code>iot:*</code>
Resource	テキストボックスに * を入力、Allow にチェック

これでポリシが作成されました

下図は “Add statement” 直前の画面です



“モノ”の作成

AWS IoT 上で“モノ”として認識できるようにします

実物の“モノ”の状態を管理するための機能である Thing shadow を使用する際に、特に必要となります

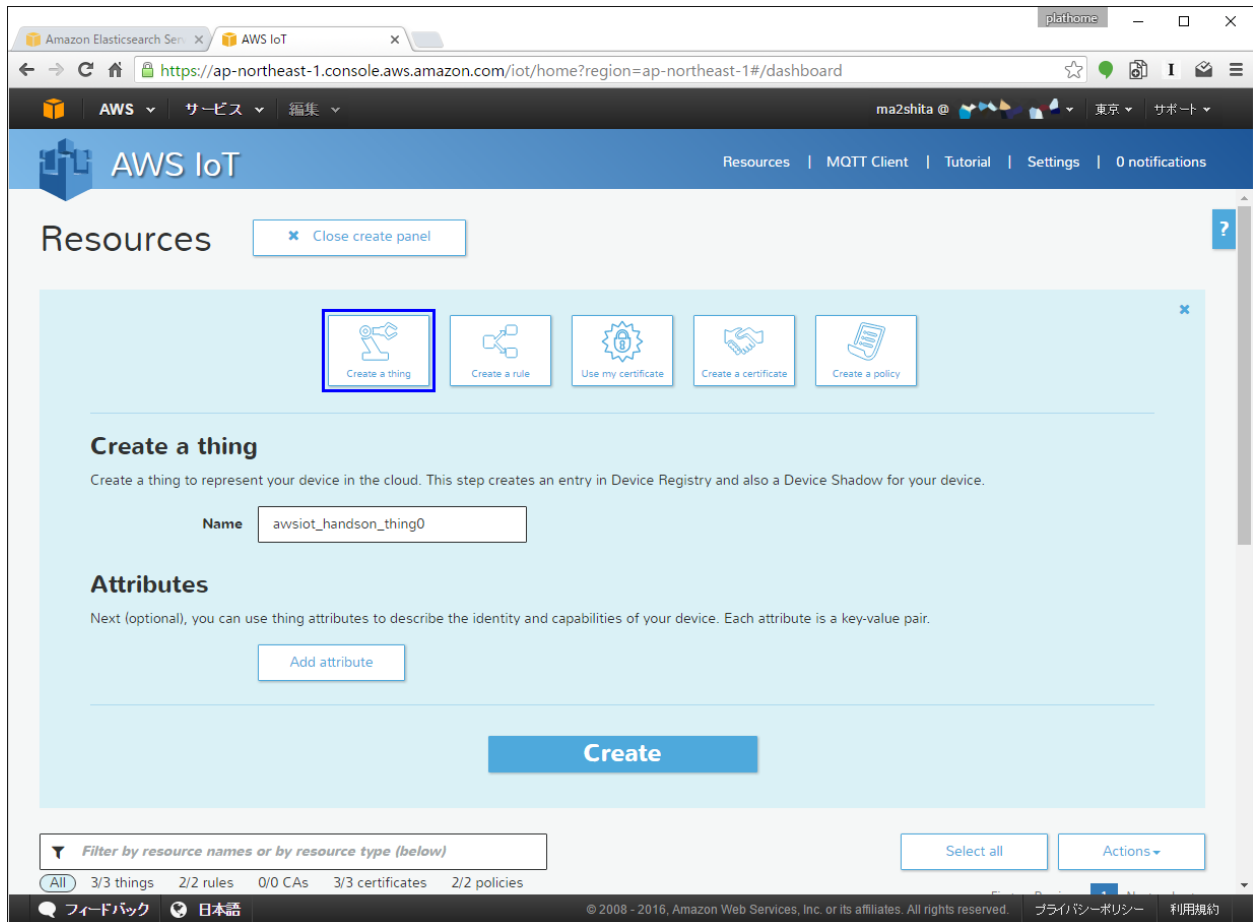
作業

1. AWS IoT のコンソールから“Create a thing”をクリック
2. 下記を入力したら“Create”をクリック

Name

これで“モノ”が作成されました

下図は“Create”直前の画面です



証明書の作成, ポリシの割り当て, “モノ”の割り当て

“モノ”がAWS IoTにアクセスする際に使用する証明書(キーペア)を作成します

“モノ”に公開鍵を持たせてAWS IoTにアクセスすることで認証としています

証明書は有効(活動中)/無効(非活動)というステータスを持っており、証明書が有効だとしても非活動の場合はAWS IoTへのアクセスができないといった制御が可能です

また、この証明書にポリシーと“モノ”を割り当てることで、その証明書を持っている“モノ”の制限をすることができますという仕組みです

すでに存在するキーペアから作成することも可能ですが、今回はAWS IoTに発行してもらいます

注釈: AWS IoT 接続トラブルの原因の80%が、証明書に起因するものですので丁寧に実施してください

作業

- 証明書の作成
 1. AWS IoTのコンソールから“Create a certificate”をクリック
 2. “1-Click certificate create”をクリック <“INACTIVE”と書かれた証明書が作成されます>

3. 画面上の “Download private key” と “Download certificate” をクリックし、それぞれ .pem.key ファイルと .pem.crt ファイルを取得する

警告:

- private key ファイルはこのタイミングでのみダウンロード可能です。あとでダウンロードできないので、必ず取得してください

- ポリシを証明書に割り当て
 1. 作成された証明書のチェックボックスをクリック (ついていれば次へ)
 2. [Actions] の中から [Attach a policy] をクリック
 3. Policy name にポリシ名 `awsiot_handson_policy` を入力し “Attach” をクリック
- “モノ” を証明書に割り当て
 1. 証明書のチェックボックスをクリック (ついていれば次へ)
 2. [Actions] の中から [Attach a thing] をクリック
 3. Thing name にポリシ名 `awsiot_handson_thing0` を入力し “Attach” をクリック
- 証明書のアクティベート
 1. 証明書のチェックボックスをクリック (ついていれば次へ)
 2. [Actions] の中から [Activate] をクリック <証明書が “ACTIVE” に変化します>

下図は 証明書にチェックを入れた後 “Actions” をクリックした直後の画面です

Screenshot of the AWS IoT console showing the "Resources" page. The "Create a certificate" button is highlighted with a blue box. Below the buttons, a message states: "Your new certificate has been created. You can attach a certificate to a thing so it can connect to AWS IoT and attach a policy to give it permissions. Please download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys will not be retrievable after closing this form." Below this message are three links: "Download public key", "Download private key", and "Download certificate". At the bottom, there is a table of resources with columns for "Filter by resource names or by resource type (below)", "Select all", and "Actions". The table shows three resources: a certificate (INACTIVE), a thing (awsiot_handson_thing_0), and a policy (awsiot_handson_policy). The "Actions" dropdown menu is open, showing options like "Activate", "Deactivate", "Revoke", "Delete", "Accept transfer", "Reject transfer", "Revoke transfer", "Start transfer", "Attach a policy", and "Attach a thing".

ルールの作成

AWS IoT では、MQTT や REST で送信されてきたデータに対して、どのようにアクションするか設定でき、これをルールと呼びます

作業

1. AWS IoT のコンソールから “Create a rule” をクリック
2. 下記を入力したら “Add action” をクリックし “Create” をクリック

Name	awsiot_handson_rule0
Description	awsiot_handson_rule0
SQL version	2016-03-23-beta
Attribute	*
Topic filter	awsiot_handson/sensor0
Condition	<なにも入力しません>
Chosen an action	Amazon Elasticsearch Service
Domain name	awsiot-handson-es
ID	\${newuuid() }
Index	awsiot_handson
Type	fwm8blz02
Role	awsiot_handson_put_to_es

これでルールが作成されました

下図は “Add action” 直前の画面です

AWS IoT
R

Create a rule

Create a rule to evaluate inbound messages published into AWS IoT. Your rule can deliver a message to the topic of another device.

Name your rule and add an optional description.

Name

Description

Indicate the source of the messages you want to process with this rule.

Rule query statement

SQL version

Attribute

Topic filter

Condition

Select one or more actions to happen when the above rule query is matched by an inbound message. Actions define additional actions, such as storing data in a database, invoking cloud functions, or sending notifications. (* required)

Choose an action

This action will send the message to an Amazon Elasticsearch cluster.

***Domain name** [Create a new resource](#)

***Endpoint**

***ID**

***Index**

***Type**

***Role name** [Create a new role](#)

注釈: Amazon Elasticsearch Service のインスタンスが完了してない場合は Endpoint が `https://null` となり、設定が完了できません。Amazon ES のインスタンス作成の完了を待ってからルール作成を行ってください

AWS IoT 上の MQTT クライアントツールを使用した確認

AWS IoT には MQTT クライアントツールがあり、それを使って簡単に動作確認をすることができます

作業

1. AWS IoT コンソールの右上 “MQTT Client” をクリック
2. “Generate client ID” をクリック <Client ID に任意の文字列が入ります>
3. “Connect” をクリック
4. “Publish to topic” をクリック
5. 下記を入力して “Publish” をクリック

Publish topic	awsiot_handson/sensor0
Payload	{"state":{"reported":{"deviceId":"awsiot-test0","field1":3,"time":"2016-05-19T10:"

ptions.

ck "Subscribe"

opic by clicking the 'x' in the right corner

MQTT Client Actions

Device Gateway connection ✓

Subscribe to topic 📄

Publish to topic ➤

Publish log 🔄

Publish topic

awsiot_handson/sensor0

Quality of service
(QoS) 0 1

Payload

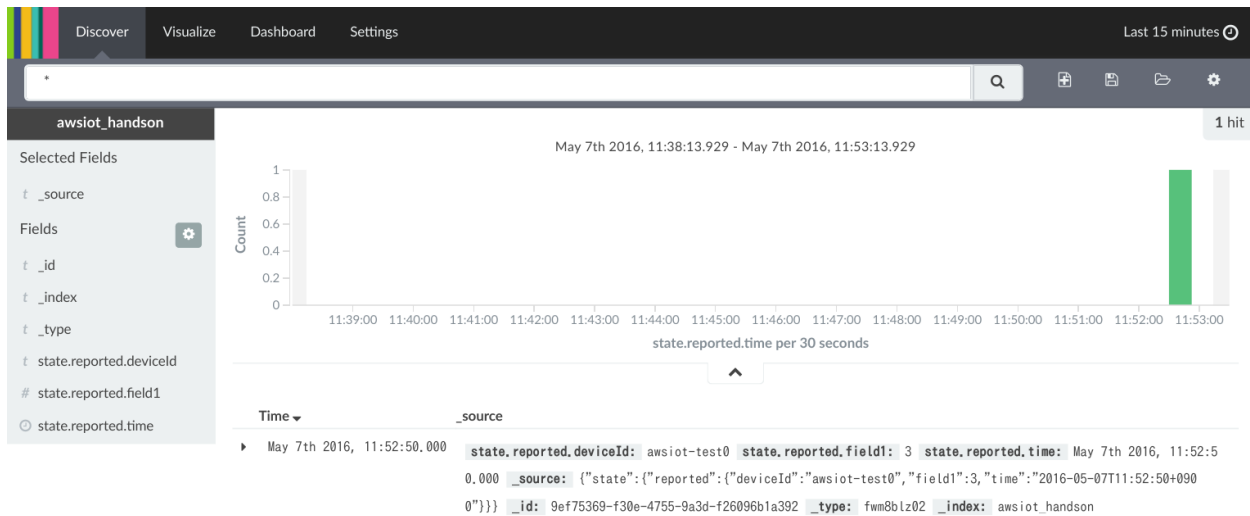
```
1 {"state":{"reported":{"deviceId":"awsiot-test0","field1":3,"f
```

Publish

Kibana 上に上記 payload のデータが入っていれば成功です

注釈: Amazon Elasticsearch Service のインスタンス作成と設定 では `es-test` という Index にデータを入れましたが、この章で使用している Index は `awsiot_handson` です

Kibana の Settings 画面から新規に `awsiot_handson` を基に Index を作成するようにしてください



ここまで到達できればゴールです

[BX1 と AWS IoT の接続 へ進む](#)

トラブルシューティング

AWS IoT のログ

CloudWatch で確認することができます

The screenshot shows the AWS CloudWatch console interface. On the left is a navigation menu with options like 'ダッシュボード', 'アラーム', 'イベント', 'ログ', and 'メトリックス'. The main content area is titled 'ロググループ' (Log Groups). It features a search filter and a table with the following data:

ロググループ	次の期間経過後にイベントを失効	メトリックスフィルタ	サブスクリプション
/aws/lambda/ansible-playbook-broker	失効しない	0 フィルタ	なし
/aws/lambda/jawsdays20160312_to_es	失効しない	0 フィルタ	なし
/aws/lambda/solar_demo_add_item_to_dynamodb	失効しない	0 フィルタ	なし
/aws/lambda/solar_demo_get_item_from_dynamodb	失効しない	0 フィルタ	なし
AWSIoTLogs	1 か月 (30 日間)	0 フィルタ	なし

Certificate ファイルや Private key ファイルのダウンロードを忘れた

AWS IoT 上で証明書を作成しなおしてください

また、ファイルを失ってしまった証明書は削除してください

AWS IoT の証明書が削除できない

証明書を削除できる条件は 1. モノやポリシーが割り当てられていない 2. Deactivate 状態である この 2 つが満たされている必要があります

割り当て済みのポリシーや“モノ”を解除する

1. 証明書をクリック
2. 右側に現れたウィンドウの [Detail] で モノやポリシーを “dettach” します

Deactivate 状態にする

1. 証明書のチェックボックスをクリック (ついていれば次へ)
2. [Actions] の中から [Deactivate] をクリック <証明書が “INACTIVE” に変化します>

Rule 作成時に **Elasticsearch Service** のインスタンスが見つからない

リージョンを確認してください

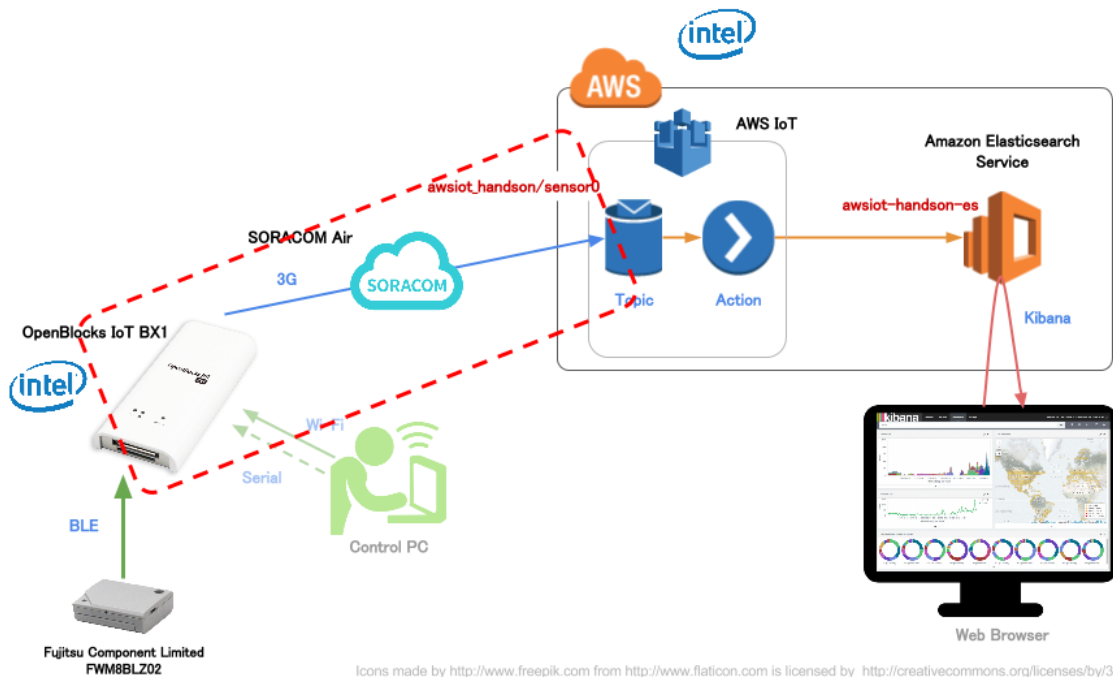
AWS IoT と Elasticsearch Service のリージョンは一致している必要があります

Kibana にデータが表示されない

表示対象の Index が `es-test` になっていませんか？

BX1 と AWS IoT の接続

本章のゴール: センサーからのデータが AWS IoT を通じて Amazon ES 上の Kibana 上で確認できる作業の位置づけ;



上図 SVG

準備

AWS IoT から入手しておくもの

- 証明書 (Certificate) ファイル <拡張子 `.pem.crt`>
- プライベートキー (PrivateKey) ファイル <拡張子 `.pem.key`>
- エンドポイント

エンドポイントの確認方法

最終的には“XXXXXXXXX.iot.REGION.amazonaws.com”というフォーマットの文字列を入手します

下記どちらかの方法で確認してください

- AWS CLI で `aws iot describe-endpoint` で確認
- AWS IoT コンソールの“Things”の REST API endpoint から抜き出す

The screenshot shows the AWS IoT console interface. On the left, there is a list of Things with IDs like 5c313ec11d02 and 5c313ec11d06. The main panel displays the details for a Thing with ID 5c313ec11d00. The REST API endpoint is shown as `https://XXXXXXXXX.iot.ap-northeast-1.amazonaws.com/things/5c313ec11d00/shadow`. A red box highlights the host part of the URL, and a red annotation reads: “http://”から後の、host名の部分を必要とします. Other details include the MQTT topic, last update time, and shadow state.

その他、入手しておくもの

- ルート証明書ファイル

警告:

- 上記 URL にはファイル名にスペースが含まれています。しかし BX1 ではファイル名にスペース文字が使用できないため、処置をしておいてください(本ハンズオンではスペースを `_` に変更しています)

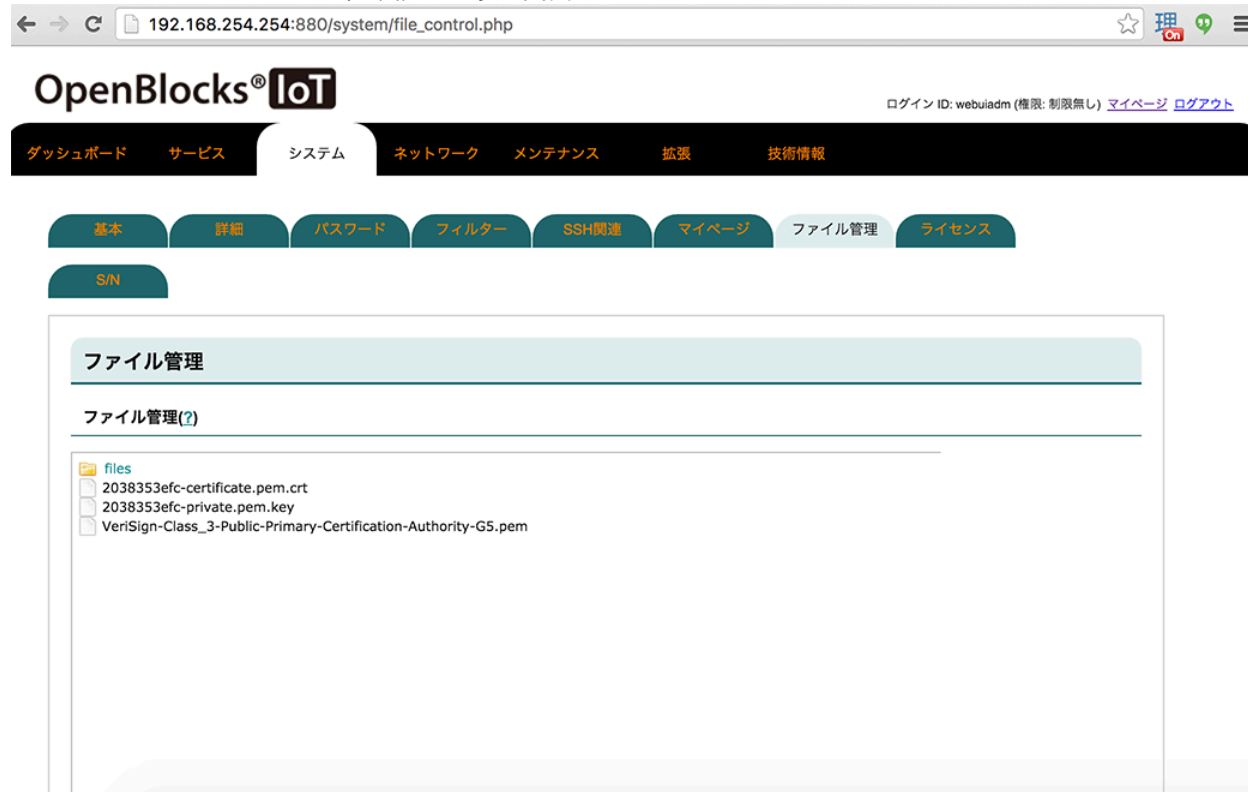
BX1 設定

ファイルアップロード

AWS IoT の証明書ファイル・プライベートキーファイル・ルート証明書ファイルの 3 つを BX1 にアップロードします

1. WebUI から [システム] - [ファイル管理] を表示
2. 本画面からファイルを 3 つ、それぞれアップロード

すべてアップロードされると、下記のような画面となります



収集設定 / AWS IoT に対する設定

1. WebUI から [サービス] - [収集設定] を表示
2. AWS IoT を使用する (クリックで設定が展開します)
3. 下記の通り設定します

インターバル	3
送信先ホスト	<エンドポイント URL>
root 証明書	/var/webui/upload_dir/VeriSign-Class_3-Public-Primary-Certification-Authority-G5

注釈:

- 先のファイルアップロード画面でアップロードされたファイルは BX1 内の `/var/webui/upload_dir/` にアップロードされます。それ以下のパスを指定することでファイルの読み込みが可能です

保存せず、引き続きページ下部へ移動します

収集設定 / センサーのデータ送信先に **AWS IoT** を加える

1. dev_le_0000001 の 送信先設定 で **AWSIoT** にチェックを付けます (クリックで設定が展開します)
2. 下記の通り設定します

トピック名	awsiot_handson/sensor0
証明書 (AWS IoT)	/var/webui/upload_dir/<アップロードした .pem.crt ファイル>
プライベートキー (AWS IoT)	/var/webui/upload_dir/<アップロードした .pem.key ファイル>

192.168.254.254

MQTTサーバ 使用する 使用しない

WEBサーバ(PLAIN) 使用する 使用しない

ビーコン送信設定(?)

送信対象 送信する 送信しない

デバイス情報送信設定

デバイス番号 dev_le_0000001

送信対象 送信する 送信しない

アドレス E7:3A:40:83:9D:8A

ユーザーメモ

センサー信号強度[dbm] 0

取得時間間隔[ms] 1000

送信先設定 local PD KINESIS AWSIOT BLUEMIX MQTT PLAIN

クライアントID (AWS IoT) 673a40839d8a

Thing Shadows(AWSIoT) 使用しない

トピック名(AWSIoT) /awsdays2016/sensor0

証明書(AWSIoT) /var/webui/upload_dir/505cad28c0-certifica

プライベートキー(AWSIoT) /var/webui/upload_dir/505cad28c0-private.

操作

Version 1.0.7

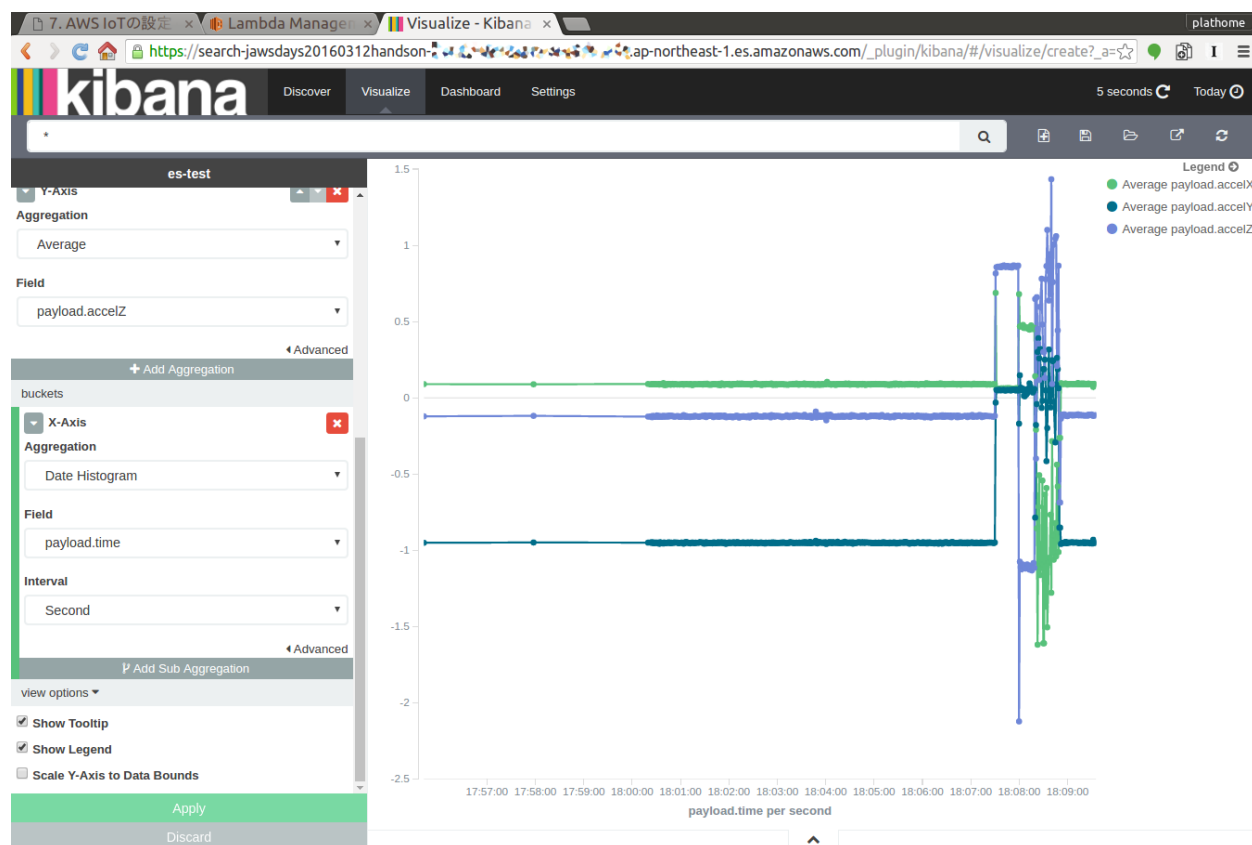
保存後、即座に AWS IoT への送信が始まります。Amazon ES の Kibana でデータ受信ができていないか確認してください



データの変化を楽しむ

加速度をつかってこんなグラフを作ることができます

Kibana の使い方は本ハンズオンの範疇を超えてしまうため、興味のある方はチューターにお声がけください



センサーのデータ取得間隔が1秒毎となっているため、加速度の数値を変化させたい場合は若干振り続ける等するようにしてください

くれぐれも降ってる最中に投げ出す事のないように、ご注意事項です！

お疲れ様でした！

無事ハンズオンのゴール達成です！

あとかたづけ作業をお忘れなく！

[片付け&まとめへ進む](#)

トラブルシューティング

以下を確認してみてください

1. センサーからのデータを受信できているか (pd-handler-stdout.log を確認してください)
2. 3G 通信ができているか (ping を試してみてください)
3. AWS IoT へ接続・送信できているか (CloudWatch を確認してください)
 - AWS IoT のエンドポイントはあっているか
 - トピック名はあっているか
 - 証明書があっているか
 - 証明書は ACTIVE になっているか

- 有効なポリシーがアタッチされているか
 - ルールは正しく Amazon Elasticsearch Service へデータを PUT できるようになっているか
4. Amazon Elasticsearch Service へのデータ PUT (CloudWatch を確認してください)
- 有効なロールがセットされているか

BX1 から AWS IoT への送信状況の確認

WebUI から [サービス] - [収集ログ] にて、動作確認が可能です

ログ選択はそれぞれ下記のとおりです

pd-handler-stdout.log	センサー <-> BX1 間の送受信状況
pd-emitter.log	BX1 <-> 送信先 (local や AWS IoT 等) 間の送受信状況

片付け&まとめ

AWS リソース

本ハンズオンで作成した AWS リソース等は継続的に費用がかかるものがあるため、不要ならば削除するようにしてください

- AWS IoT
 - ルール
 - 証明書
 1. アタッチされている“モノ”とポリシーをデタッチ
 2. 証明書の Deactivate
 - “モノ”
 - ポリシ
- IAM ロール
- Amazon ES インスタンス (課金されるので注意)

自習室

参考サイト

- [太陽光パネルの発電量を AWS IoT と Amazon Elasticsearch Service を使って可視化してみる](#)

自習室: SORACOM Beam で AWS IoT の認証処理をオフロード

[自習室: SORACOM Beam で AWS IoT の認証処理をオフロードへ](#)

自習室: AWS IoT の Thing Shadow でパトライトを制御する

[自習室: AWS IoT の Thing Shadow でパトライトを制御するへ](#)

センサーデータフォーマット

センサーデータは、BX1 を通過すると下記のような JSON データとなります
このフォーマットを AWS IoT 内の Rule Engine や、Lambda で処理することが可能です
(jq を使って整形済み。コメントは付与されません)

富士通コンポーネント社製 温度・加速度センサーデバイス “FWM8BLZ02”

```
{
  "deviceId": "e73a40839d8a", // BX1 の設定画面で設定したデバイス ID (BX1 で付与)
  "time": "2016-03-09T18:15:53.764+0900", // 計測日時 (BX1 で付与)
  "temperature": 27.22, // 温度
  "accelX": 0.092, // X 方向 加速度 G
  "accelY": -0.952, // Y 方向 加速度 G
  "accelZ": -0.113 // Z 方向 加速度 G
}
```

BX1 の “AWSIOT” 経由でデータ送信された場合、`{state: {reported: DATA}}` とした AWS IoT の Device Shadow のフォーマットに変形されます

```
{
  "state": {
    "reported": {
      "deviceId": "e73a40839d8a",
      "time": "2016-03-09T18:15:53.764+0900",
      "temperature": 27.22,
      "accelX": 0.092,
      "accelY": -0.952,
      "accelZ": -0.113
    }
  }
}
```

Texas Instruments 社製 多機能/開発用センサー SensorTag CC2541DK

```
{
  "deviceId": "5c313ec027e1", // BX1 の設定画面で設定したデバイス ID (BX1 で付与)
  "humidity": 40.7, // 湿度 %
  "temperature": 28.6, // 温度 (湿度センサー内)
  "objectTemp": 23.8, // 物体温度
  "ambientTemp": 28.4, // 周辺温度
  "gyroX": -1.4, // ジャイロ (X 軸) deg/s <角速度>
  "gyroY": 4, // ジャイロ (Y 軸) deg/s
  "gyroZ": 0.2, // ジャイロ (Z 軸) deg/s
  "pressure": 1015.6, // 気圧 hPa
  "accelX": 0.1, // 加速度 (X 軸) G
  "accelY": 0.3, // 加速度 (Y 軸) G
  "accelZ": 3.8, // 加速度 (Z 軸) G
  "magX": -53.9, // 地磁気 (X 軸) μT <テスラ>
  "magY": -5.2, // 地磁気 (Y 軸) μT
  "magZ": 102.7, // 地磁気 (Z 軸) μT
  "time": "2015-11-19T10:29:20.529+0900" // 計測日時 (BX1 で付与)
}
```


BX1 の "AWSIoT" 経由でデータ送信された場合、`{state: {reported: DATA}}` とした AWS IoT の Device Shadow のフォーマットに変形されます

```
{
  "state": {
    "reported": {
      "deviceId": "5c313ec027e1",
      "humidity": 40.7,
      "temperature": 28.6,
      "objectTemp": 23.8,
      "ambientTemp": 28.4,
      "gyroX": -1.4,
      "gyroY": 4,
      "gyroZ": 0.2,
      "pressure": 1015.6,
      "accelX": 0.1,
      "accelY": 0.3,
      "accelZ": 3.8,
      "magX": -53.9,
      "magY": -5.2,
      "magZ": 102.7,
      "time": "2015-11-19T10:29:20.529+0900"
    }
  }
}
```

BX1 ヘシリアルコンソールでログインする

BX1 は給電用 USB ケーブルが、シリアルコンソールを兼任しています

FTDI のシリアルポートドライバがインストール済みの Windows / Mac OS X や、Linux ならば追加ドライバ不要でアクセス可能です

ID	root
Password	0BSIoT (ゼロ ビー エス アイ ゼロ ティー)

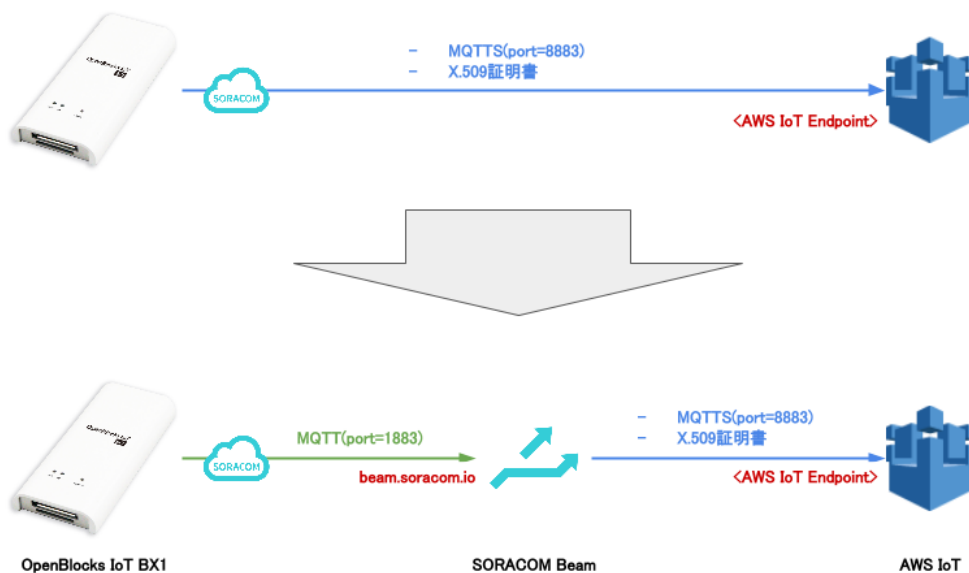
e.g.) screen コマンドによるアクセス

```
screen /dev/ttyUSB0 115200
```

自習室: SORACOM Beam で AWS IoT の認証処理をオフロード

本章のゴール: BX1 の設定を AWS IoT から MQTT に切り替えても AWS IoT ならびに Amazon ES へのデータ送信がされる

作業の位置づけ;



SORACOM Beam とは？

IoT デバイスにかかる暗号化等の高負荷処理や接続先の設定といった面倒な処理を、クラウドにオフロードできるサービスです

Beam でサポートしている機能は下記のとおりです (2016 年 5 月現在)

- プロトコル変換 (証明書添付含む)
- 接続先切り替え

詳細は [SORACOM Beam のページ](#) をご覧ください

自習室で行うこと

BX1 と AWS IoT の接続 では、AWS IoT に接続するために必要な証明書 (ファイル 3 つ) を BX1 に格納していました

この証明書を BX1 から SORACOM へ移動し、SORACOM Beam を使って AWS IoT に接続します

前提条件

すでにハンズオンすべてを完了させており、センサーデータが Amazon Elasticsearch Service の Kibana 上で確認できることが必要です

BX1 の仕様上の注意点

SORACOM Beam への切り替えに先立って、BX1 の仕様上の注意を確認します

1. AWS IoT 向けと MQTT サーバ向けでデータの payload フォーマットが違います。詳細は [センサーデータフォーマット](#) をご覧ください

- MQTT サーバ向けの送信先トピックは <トピックプレフィックス><ユニーク ID(MQTT)> と、ユニーク ID が自動的に付与されます

作業手順

- BX1*: AWS IoT への送信を停止
- AWS: AWS IoT にルールを追加する
- SORACOM: 証明書ストアへ X.509 証明書の登録
- SORACOM: SIM グループの作成と Beam の設定
- SORACOM: SIM をグループに所属させる
- BX1*: データを SORACOM Beam(MQTT) へ送信

BX1: AWS IoT への送信を停止

AWS IoT への送信を停止します

- WebUI から [サービス] - [収集設定] を表示
- AWS IoT を使用しないに変更
- [保存] をクリック

保存すると AWS IoT への送信を即時停止します (BX1 内のグラフ描画は継続されます)

Kibana 上にも送信されていないことを確認してください

注釈: デバイス情報送信設定の送信先設定には AWS IoT の設定情報が残っていますが、使用されません

AWS: AWS IoT にルールを追加する

AWS IoT にルールを追加します

BX1 固有の注意点にも記載したとおり、MQTT の送信先トピックが `awsiot_handson/sensors/<ユニーク ID (MQTT)>` となったため、これを受けられる AWS IoT のルールを新設します

ルールの作成の仕方は [ルールの作成](#) を参照してください

Name	awsiot_handson_rule1
Description	awsiot_handson_rule1
SQL version	2016-03-23-beta
Attribute	*
Topic filter	awsiot_handson/sensors/#
Condition	<なにも入力しません>
Choose an action	Amazon Elasticsearch Service
Domain name	awsiot-handson-es
ID	\${newuuid() }
Index	awsiot_handson_w_beam
Type	fwm8blz02
Role	awsiot_handson_put_to_es

[ルールの作成](#) と違う部分を特に強調してあります

SORACOM: 証明書ストアへ X.509 証明書の登録

ソラコムのコソールから [セキュリティ] - [認証情報ストア] - [認証情報を登録] を順にクリック
各項目は下記のようにしてください

認証情報 ID	AWSIoT_cert_0
概要	AWSIoT cert 0
種別	X.509 証明書
秘密鍵 (key)	<CertID>-private.pem.key の中身をペースト
証明書 (cert)	<CertID>-certificate.pem.crt の中身をペースト
CA 証明局	VeriSign-Class 3-Public-Primary-Certification-Authority-G5.pem の中身をペースト

認証情報を登録

認証情報 ID *

使用できる文字: A-Z, a-z, 0-9, -, _
1 文字以上 100 文字以内としてください

概要

種別 *

秘密鍵 (key) *

証明書 (cert) *

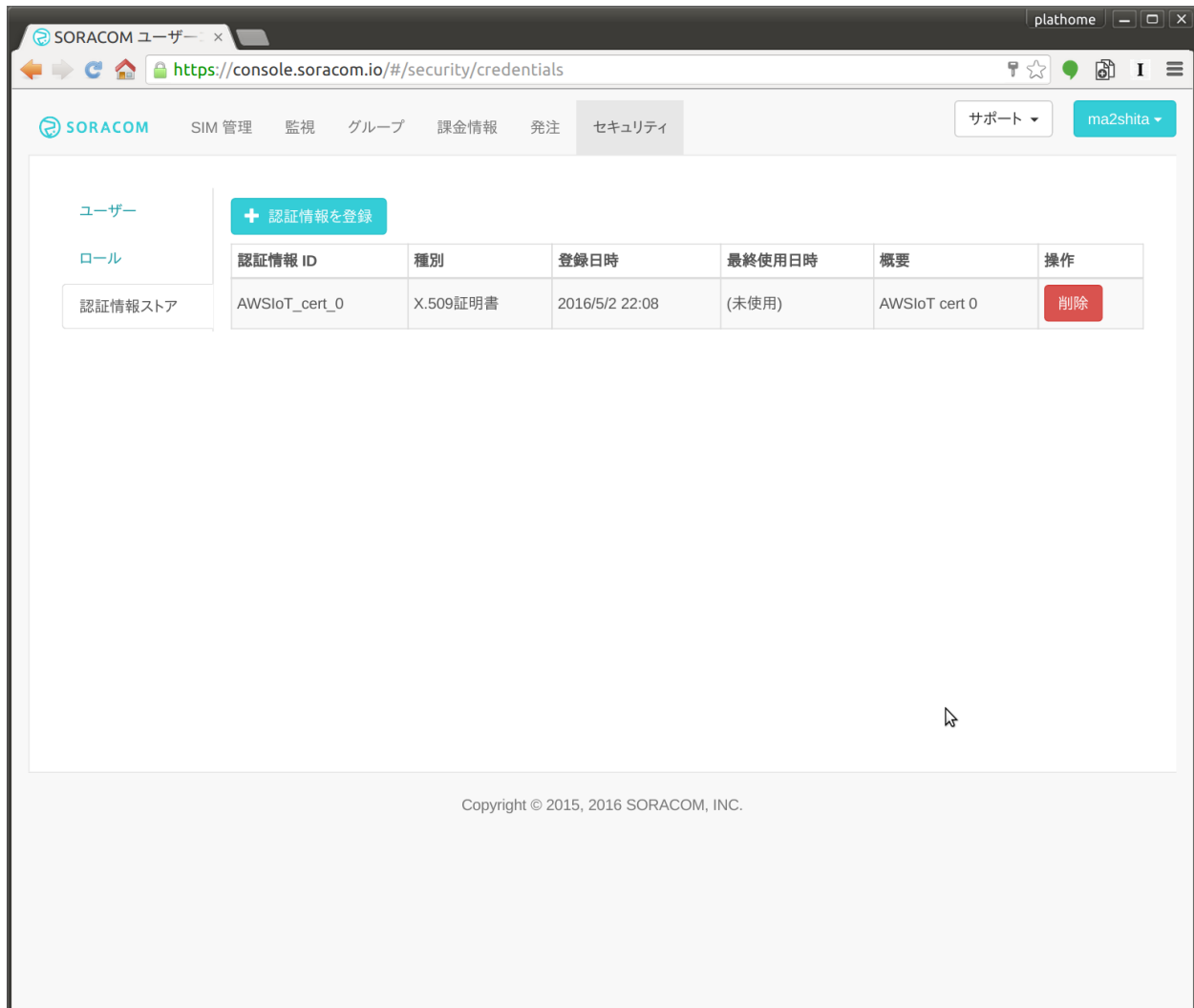
CA証明局 *

*** 必須項目**

キャンセル 登録

設定を確認したら [登録] をクリック

無事登録されると認証情報ストアに下図のように表示されます



注釈: 認証情報ストアには複数の認証情報を格納することができるほか、内容の編集・削除も可能です

SORACOM: SIM グループの作成と Beam の設定

SIM グループの作成

ソラコムのコソールから [グループ] - [追加] を順にクリック

各項目は下記のようにしてください

グループ名



グループ作成

グループ名 *

mqtt2awsiot

* は必須項目です

キャンセル グループ作成

設定を確認したら [グループ作成] をクリック

SORACOM Beam の設定

ソラコムのコソールから [グループ] - [mqtt2awsiot] - [SORACOM Beam 設定] を順にクリック

[+] をクリックし、リストの中から [MQTT エントリポイント] をクリック

各項目は下記のようにしてください

設定名	AWS IoT
転送先 / プロトコル	MQTTS
転送先 / ホスト名	<AWS IoT エンドポイント>
転送先 / ポート番号	8883
転送先 / ユーザ名	<空>
転送先 / パスワード	<空>
転送先 / 証明書	ON
転送先 / 証明書タイプ	AWSIoT_cert_0 (AWSIoT cert 0)

SORACOM Beam - MQTT 設定

設定名

AWS IoT ON

エントリーポイント

プロトコル	MQTT
ホスト名	beam.soracom.io
ポート番号	1883

転送先

プロトコル	MQTTS
ホスト名	iot.ap-northeast-1.amazonaws.com
ポート番号	8883
ユーザ名	
パスワード	

証明書 ON
クライアント証明書を使って認証します

証明書タイプ* AWSIoT_cert_0 (AWSIoT cert 0)

設定を確認したら [保存] をクリック

無事登録されると下図のように表示されます

名前	エントリポイント	状態	転送先	サマリ
AWS IoT	mqtt://beam.soracom.io:1883	有効	mqts://[redacted].iot.ap-northeast-1.amazonaws.com:8883	[...]

+ -

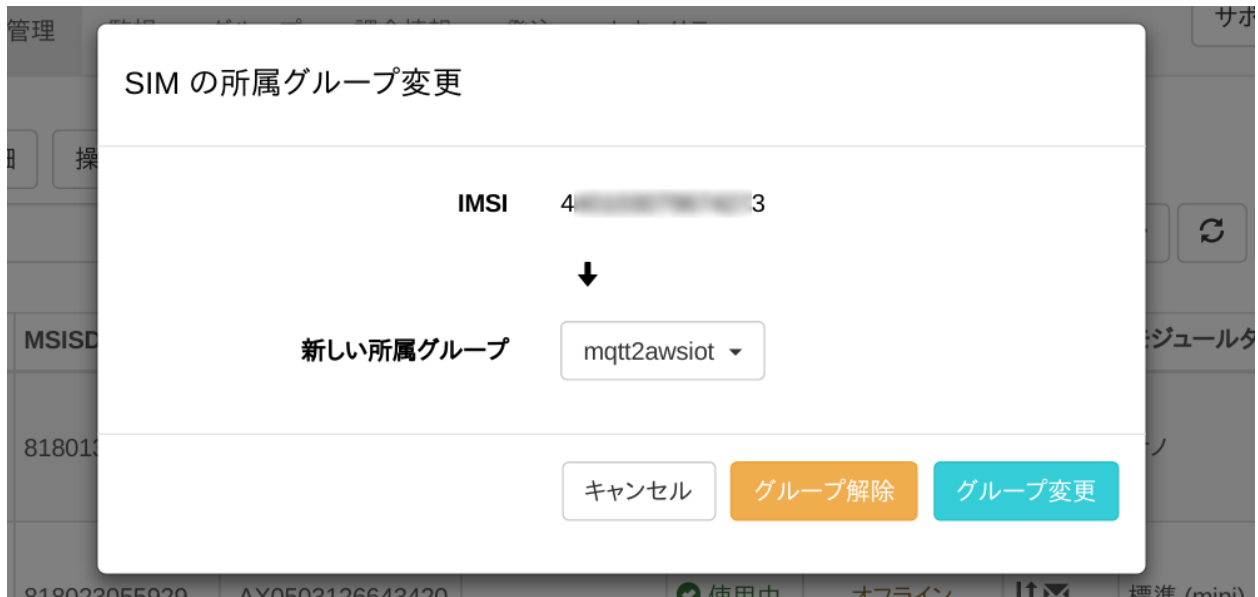
SORACOM: SIM をグループに所属させる

ソラコムのコソールから [SIM 管理] をクリック

さきほど作成した `mqtt2awsiot` グループへ所属させたい SIM のチェックボックスをつけたあと、[操作] - [所属グループ変更] をクリック

The screenshot shows the SORACOM console interface. At the top, there are navigation tabs: SIM 管理, 監視, グループ, 課金情報, 発注, and セキュリティ. Below the tabs, there are buttons for '+ SIM 登録', '詳細', and '操作'. A search bar labeled '名前検索' is present. A table lists SIMs with columns for IMSI, MSISDN, グループ, 状態, and セッション. A dropdown menu is open over the table, showing options: 使用開始, 休止, 所属グループ変更 (highlighted), 速度クラス変更, 有効期限変更, 他のオペレーターへ譲渡, and 解約. The status of the SIMs is '使用中' (In Use).

グループ変更画面で `mqtt2awsiot` を選択し [グループ変更] をクリック



無事変更されると下図のように表示されます

[-]	IMSI ?	MSISDN ?	名前 ?	グループ ?	状態 ?
<input checked="" type="checkbox"/>	4[redacted]3	8[redacted]0	nano・データのみ [redacted]	mqtt2awsiot	🟢 使用
			標準・SMS付		

以上の操作で対象の SIM から beam.soracom.io への MQTT 通信は、AWS IoT へ X.509 証明書による MQTTS 通信として転送されるようになりました

BX1: データを SORACOM Beam(MQTT) へ送信

これまでのハンズオンでは BX1 は AWS IoT へ直接 MQTTS 通信していたので、それを beam.soracom.io へ MQTT 通信するように切り替えます

収集設定 / AWS IoT の OFF と MQTT サーバへの設定

BX1 の WebUI から [サービス] - [収集設定] を表示

下記の通りにします

1. MQTT サーバ を [使用する] に変更し、下記の通り設定します

インターバル [sec]	2
送信ホスト	beam.soracom.io
送信先ポート	1883
QoS	1
クライアント ID	bx1-mqtt-client0
トピックプレフィックス	awsiot_handson/sensors
ユーザ名	<空>
パスワード	<空>
プロトコル	TCP

OpenBlocks[®] IoT

ダッシュボード

サービス

システム

ネットワーク

メンテナンス

拡張

技術情報

基本

収集設定

収集ログ

データ表示

Bluetooth関連

Bluetooth編集

BLE

送信先設定

本体内(local) 使用する 使用しない

デバイス一括設定

一括有効

一括無効

PD Exchange 使用する 使用しない

Amazon Kinesis 使用する 使用しない

AWS IoT 使用する 使用しない

Watson IoT(Device) 使用する 使用しない

Watson IoT(Gateway) 使用する 使用しない

MQTTサーバ 使用する 使用しない

インターバル[sec]

有効時間[sec]

送信先ホスト

送信先ポート

QoS

クライアントID

トピックプレフィックス

ユーザー名

パスワード

プロトコル ↓

デバイス一括設定

一括有効

一括無効

保存せず、引き続きページ下部へ移動します

収集設定 / センサデータの送信先設定

`dev_le_0000001` の送信先設定 で MQTT にチェックを付けます

ユニーク ID(MQTT) の項目が増え、クライアント ID(AWS IoT) と同じ値が入ります

デバイス情報送信設定 送信対象一括有効 送信対象一括無効

デバイス番号	dev_le_0000001
送信対象	<input checked="" type="radio"/> 送信する <input type="radio"/> 送信しない
アドレス	E7:3A:40:83:9D:8A
ユーザーメモ	
センサー信号強度[dbm]	0
取得時間間隔[ms]	1000
送信先設定	<input checked="" type="checkbox"/> local <input type="checkbox"/> PD <input type="checkbox"/> KINESIS <input checked="" type="checkbox"/> AWSIoT <input type="checkbox"/> Watson IoT(Device) <input type="checkbox"/> Watson IoT(Gateway) <input checked="" type="checkbox"/> MQTT <input type="checkbox"/> PLAIN
クライアントID (AWS IoT)	e73a40839d8a <input type="button" value="編集"/>
Thing Shadows(AWSIoT)	使用しない <input type="button" value="↓"/>
トピック名(AWSIoT)	@awsiot_handson/sensor <input type="button" value="編集"/>
証明書(AWSIoT)	(var/webui/upload_dir/33097b3bb2-certificate.p <input type="button" value="編集"/>
プライベートキー(AWSIoT)	(var/webui/upload_dir/33097b3bb2-private.pri <input type="button" value="編集"/>
ユニークID (MQTT)	e73a40839d8a <input type="button" value="編集"/>

以上を確認したら保存してください。保存と同時に (若干のラグがありますが) 送信が開始されます

Kibana 上での確認

新規に `awsiot_handson_w_beam` という Index に格納するようにしているので、そちらを参照するように変更するのを忘れなく

まとめ

SORACOM Beam を使用することで、証明書ファイルをデバイス (BX1) に格納せずとも AWS IoT へアクセスできるようになりました

デバイスへの設定が少なく済むというのは、大量のデバイスを展開する際にもコスト面で有利です

あとかたづけ

作成したリソースは削除しておきましょう

- SORACOM 上
 1. SIM のグループを解除
 2. SIM グループを削除
 3. 認証ストアの削除
- AWS 上
 1. Amazon Elasticsearch インスタンス
 2. AWS IoT モノ、証明書、ポリシー、ルール

3. IAM ロール

トラブルシューティング

SORACOM Beam には転送されたデータが正しく転送されたか否かを確認する方法がありません (2016 年 5 月現在)

そのため、うまく転送されていないと思われる場合は各種設定を見直すようにしてください

とくに見直すポイントは下記です

1. SORACOM: 証明書ストアへ保存した証明書や秘密鍵
2. AWS: AWS IoT のエンドポイント

転送がうまく作動していれば、あとは AWS の CloudWatch でログが確認できるので、処置可能です

自習室: AWS IoT の Thing Shadow でパトライトを制御する

本章のゴール: Thing Shadow の適用場面、構成コンポーネントやデータフローの理解

本章は、最初に AWS IoT の Thing Shadow のデモに触れてみることで全体像を確認し、その次に各コンポーネントの解説を行います

デモ

https://s3-ap-northeast-1.amazonaws.com/ma2shita/patlite_control_w_awsiot.html

デモはセットアップが完了していないと機能しません。詳しくはハンズオンスタッフまで

Patlite controller using AWS IoT

Current status

RED: OFF

YELLOW: OFF

GREEN: OFF

Control panel

RED

YELLOW

GREEN

Submit

Activity log

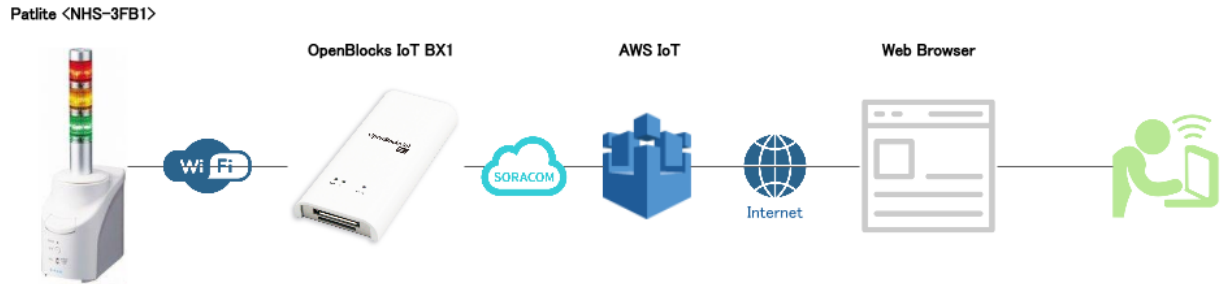
- ReceiveTopic: \$aws/things/patlite0/shadow/get/accepted | {"state":{"reported":{"red":"off","yellow":"off","green":"off"},"metadata":{"reported":{"red":{"timestamp":1462705368},"yellow":{"timestamp":1462705368},"green":{"timestamp":1462705368}}},"version":1,"timestamp":1462714999}}

デモ画面は左側に現在のパトライトの状態、右側にパトライトに対してのライト ON/OFF 制御を持っています

1. まずは右側の ON/OFF 制御を行ってみましょう
2. 気づいた事を話し合ってみましょう

概要

システムの概要は下図の通りです

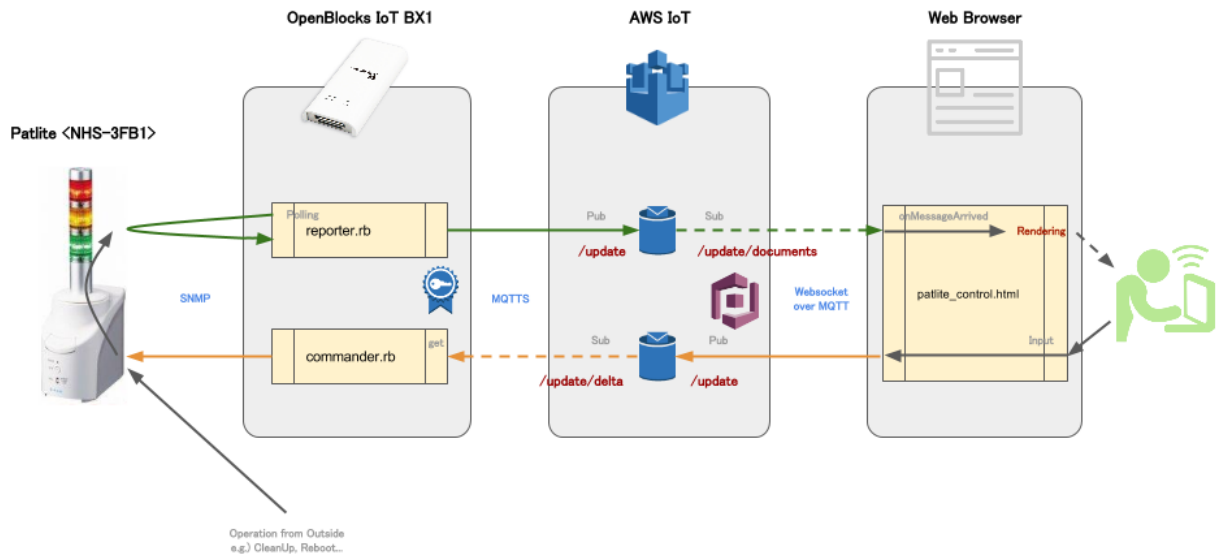


Icons made by <http://www.freepik.com> from <http://www.flaticon.com> is licensed by <http://creativecommons.org/licenses/by/3.0> <CC 3.0 BY>

デモの画面は一番右の **Web Browser** になります

コンポーネントとデータフロー

概要の詳細化を行い、コンポーネントとデータフローを記載したのが下図です



• 通信の部

- Patlite <NHS-3FB1> → BX1 との通信は **SNMP**

- * 認証/認可: SNMP の機構を使用
- BX1 → AWS IOT との通信は **MQTTS**
 - * 認証: AWS IoT が払いだした X.509 証明書を使用
 - * 認可: X.509 証明書に割り当てた AWS IoT の policy を使用
- AWS IoT → Web Browser との通信は **Websocket**
 - * 認証: なし (Cognito の Unauth(=Anonymous) を使用)
 - * 認可: Cognito を使用し Web Browser に一時的な IAM ロールを割り当て、AWS IoT(の data アクセス) に対して権限を付与
- プログラムの部
 - BX1 に 2 つのプログラムを配置
 - * `reporter.rb`
 - ・ 定期的に Patlite のステータスを取得し、AWS IoT へ **reported** を送信する
 - * `commander.rb`
 - ・ `/update/delta` を監視し、データが着信したら内容に応じて Patlite へコマンドを送信する
 - Web Browser に Javascript アプリケーションを実行させる
 - * `patlite_control_w_awsiot.html` (核となる部分を抽出したもの)
 1. `/update/documents` を監視し、データの着信したら内容に応じて表示の更新をする
 2. Web Browser への入力を基に AWS IoT へ **desired** を送信する
- データフローの部
 - 緑の線は、`reporter.rb` を起点に `/update/documents` を通じて Web Browser の内容が更新されるまで
 - オレンジの線は、Web Browser への入力を起点に `/update/delta` を通じて Patlite にコマンドが送信されるまで

システム設計のポイント

双方向システムのポイントは、データフローがループ構成になるように設計するべきです

- ただし、無限ループを防止するためにもブレーカーを必ず設置しましょう。本デモでは人間をブレーカー代わりにしています
- ループ外からの操作にも対応できるようにするべきです

双方向が不要ならば緑の線もしくはオレンジの線の上のコンポーネントのみで十分です。しかしながら、その場合に Thing Shadow を使う理由が見当たりません

- AWS IoT を **MQTT** ブローカー として見るか ステートマシン として位置づけるか、これが Thing Shadow を使う 1 つめの見極めポイントになります
- Thing Shadow の最大の特徴はデバイス操作の要求に対し 本当に操作すべき対象の抽出 を “`/update/delta`” への差分抽出 で実現/実装済みということです。この差分抽出は 現状 (=reported) と 要求 (=desired) という 2 つが必要となります。このような機構が不要であれば Thing Shadow を使用する必要はありません (e.g. センサーデータのアップロードにのみ AWS IoT を使う場合)

`commander.rb` による実行結果は `reporter.rb` に回収させるようにするのが、デバイスの稼働状況を確認するためにも推奨されます

プログラム改造のポイント

Patlite を用意できない場合もありますので、その場合は reporter.rb や commander.rb を改造する必要があります

改造ポイントはデバイス通信部になります

- reporter.rb では **Flow: Fetch state from real-device** の部分
- commander.rb では **Flow: Execute command to real-device** の部分

<NHS-3FB1>は内部に状態を持ち、SNMP で状態の読み出しができる「ステートフル・デバイス」であったため、reporter.rb はポーリング・パターン 1 で実装しました

しかし RS-232C といった I/F なデバイスはデータが I/F を通じて流れてくる「ストリーム・デバイス」であることが多いです。この場合はリスニング・パターン 2 を実装する必要があります

- 1 定期的にデータを取得しに行く定期実行型
- 2 TCP ソケットのようなポート待受型

参考資料

- [AWS IoT の MQTT over WebSocket を Cognito \(Unauth\) で認証して使ってみた | Developers.IO](#)
- [Paho - Open Source messaging for M2M](#)
- [dwyll/learn-aws-iot: Learn how to use Amazon Web Services Internet of Things \(IoT\) service to build connected applications.](#)
- [Class: AWS.Credentials AWS SDK for JavaScript](#)
- [Device Shadow MQTT Topics - AWS IoT](#)
- [テンプレートリテラルが実装された - JS.next](#)

[全体構成](#) へ進む

時間配分

1. [10m] 冒頭説明
2. [20m] BX1 の Wi-Fi AP 設定と SORACOM Air(3G ネットワーク) 設定
3. [20m] センサーと BX1 の接続
4. [10m] Amazon ES のインスタンス作成と設定
5. [20m] AWS IoT の設定
6. [20m] BX1 と AWS IoT の接続
7. [10m] 片付け&まとめ

冒頭説明から片付け&まとめまで約 120 分のコンテンツです

- [20m] 自習室: SORACOM Beam で AWS IoT の認証処理をオフロード
- [20m] 自習室: Thing Shadow でパトライトを制御