

KES IoT Logic.com

KES IoT Logic.simple

ハンズオン



AWS IoT Greengrass設定編



©Kanazawa Engineering Systems Inc.

株式会社金沢エンジニアリングシステムズ

アジェンダ



事前準備



AWS IoT Greengrass の 概念



AWS IoT Greengrass の 設定 / 確認

事前準備

Greengrassを設定する前の環境準備



事前準備

• 以下の機材があることをご確認ください。ハンズオン 接続編 を一通り実施しておくことをお勧めします。

• PC



Chrome をインストールしておいてください。



Wi-Fi、有線LAN、またはモバイル経由でインターネットにアクセスできるようにしてください。

AWSのアカウントがあることが前提です。

• USB LANアダプタ



PCにLANポートがある場合は不要ですが
IP設定を固定し切替ながらセットアップするため、
専用にもう1つあると変便利です。

• LANケーブル



• KES IoT Logic.comp/.simple +GG



• SIM挿入済みであること
• AWS IoT へのアップ、ダウンが完了していること
(ハンズオン 接続編が完了していること)

• 証明書/プライベートキー



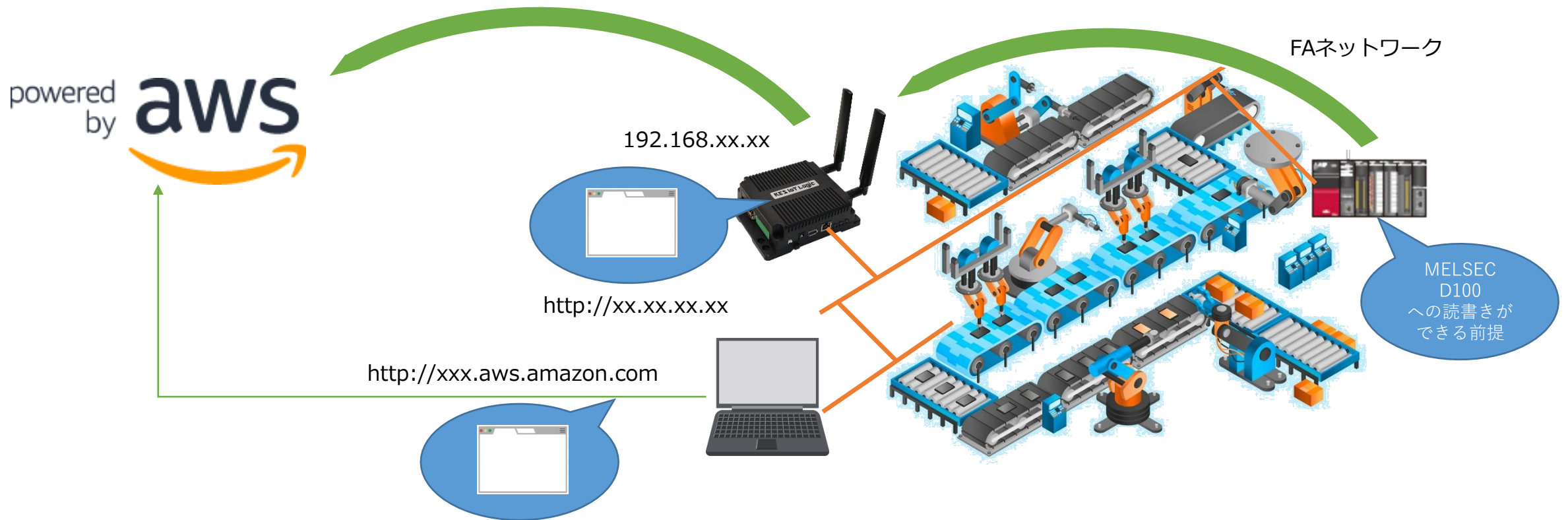
接続編で用意した証明書類やモノの名前を再度使います。4

Greengrassを設定する前の環境準備



AWS IoT への接続が完了していること

PLC <-> KES IoT Logic <-> AWS IoT と Publish ができていることを確認してください。



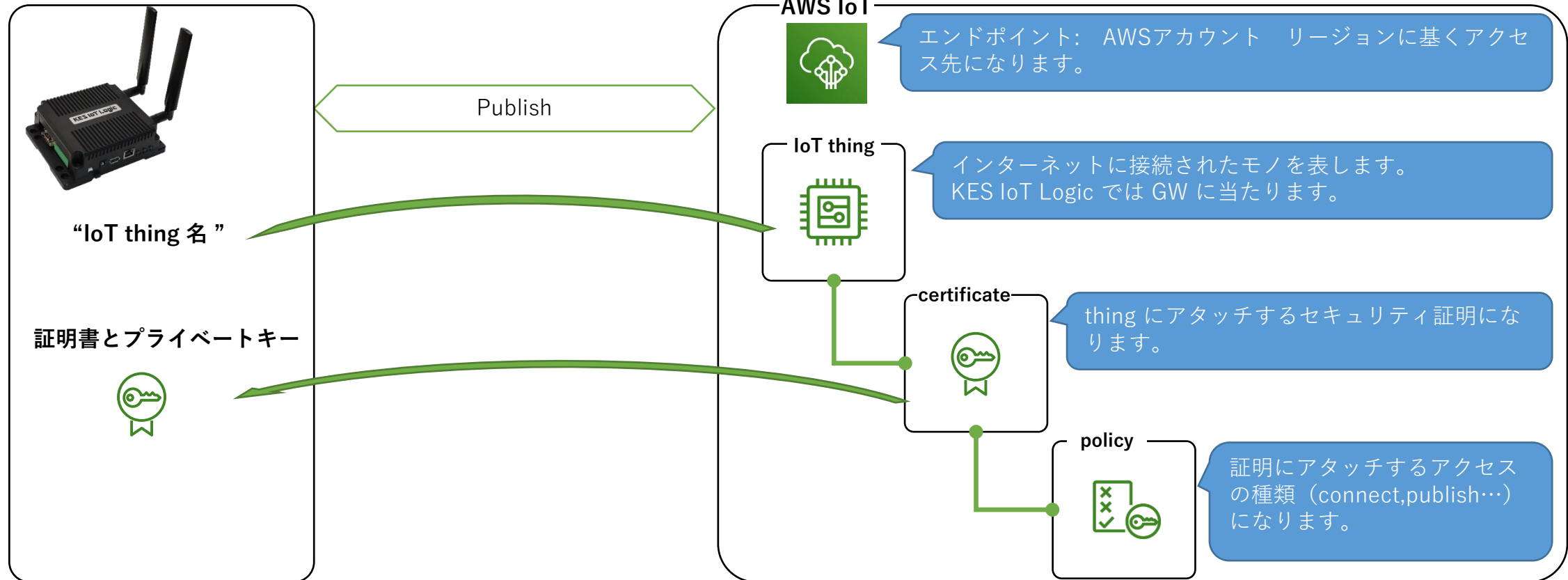
AWS IoT Greengrass の概念

Greengrass の 概念



AWS IoT と GW の関係性

AWS IoT の復習です。接続編では、メッセージブローカーを利用したデータ送受信を説明しました。

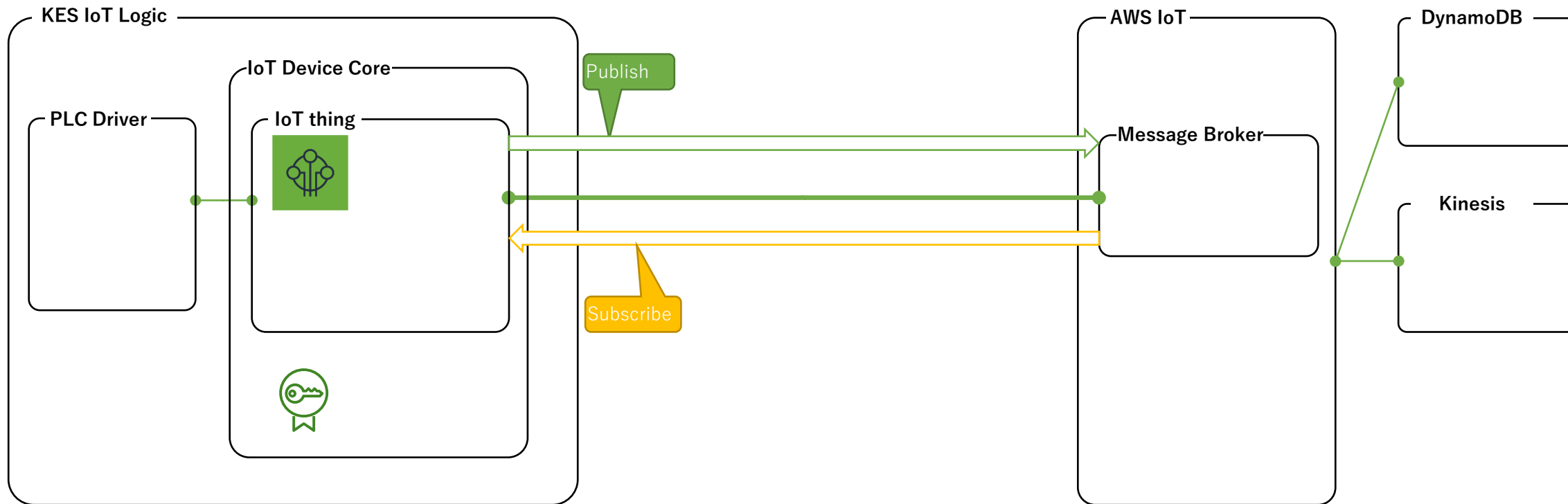


Greengrass の 概念



AWS IoT の メッセージブローカー について

メッセージブローカーはMQTTを利用した、thingとAWS IoT 間の通信を担うサービスです。

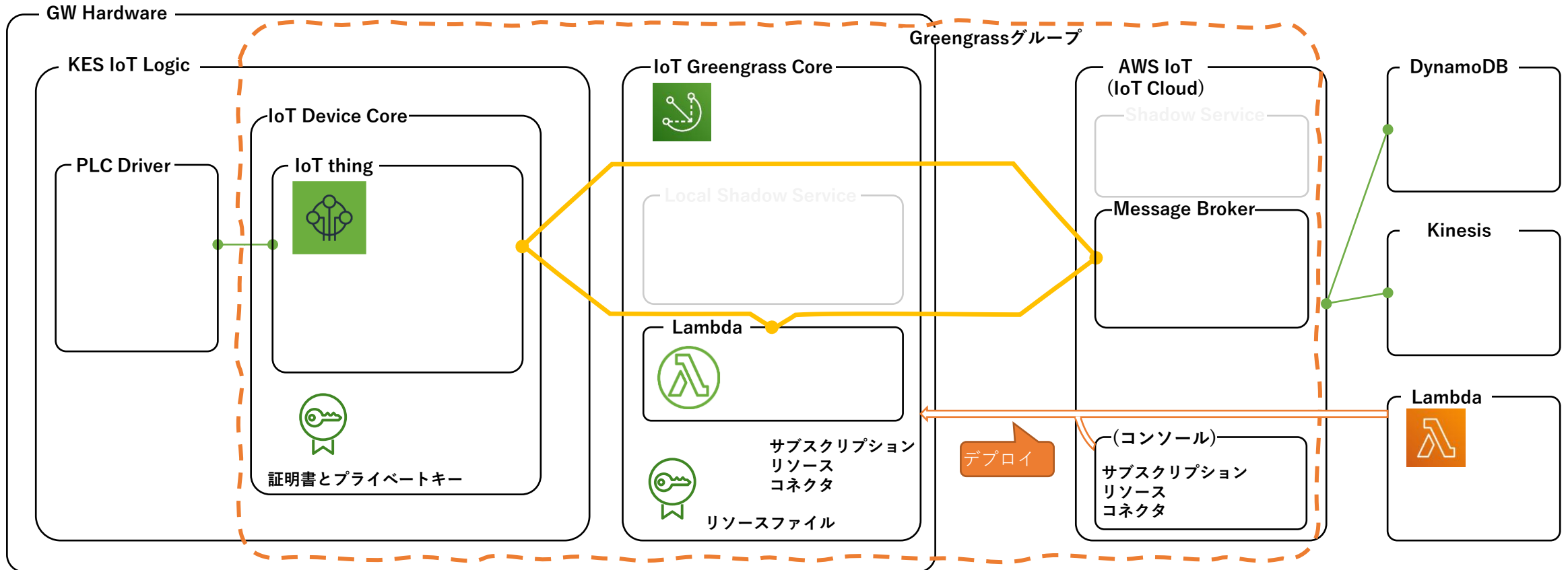


Greengrass の概念



Thing - Greengrass - AWS IoT の関係性

IoT Thing - GG - AWS IoT の関係を理解しましょう。



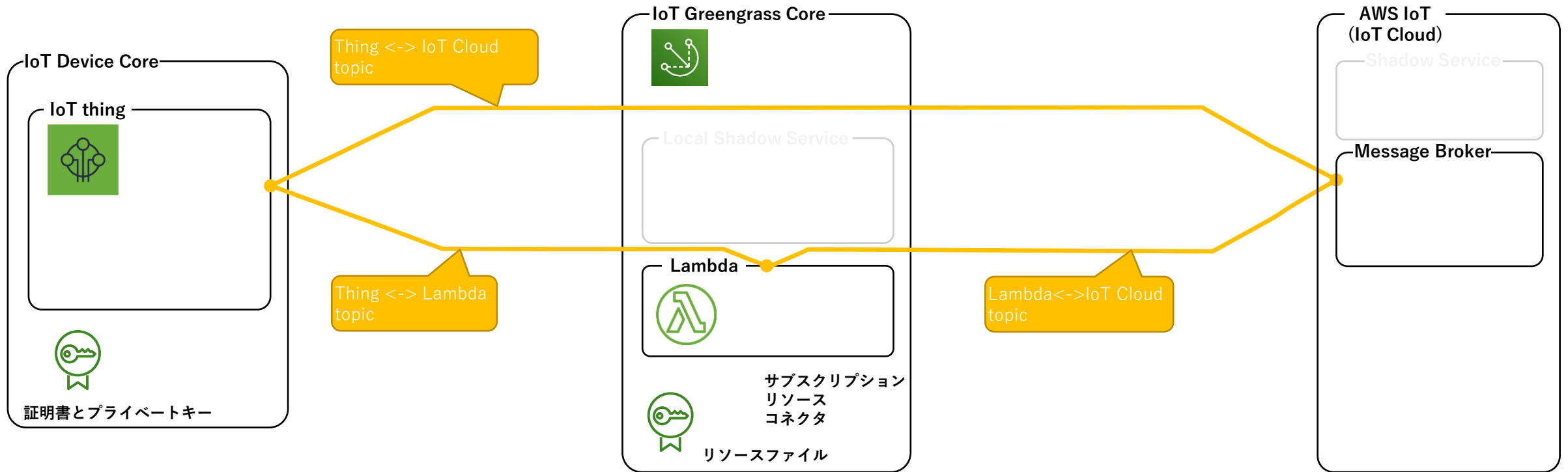
Greengrass の 概念



Greengrassグループ の詳細説明

GG においても、メッセージブローカーの利用が可能です。ブローカーを利用して、Lambdaが処理したデータをクラウドへ送信できます。

Greengrassグループ



AWS IoT Greengrass の 設定 / 確認

AWS IoT Greengrass の準備



サインイン

ルートユーザー
無制限アクセスを必要とするタスクを実行するアカウント所有者。 [詳細はこちら](#)

IAM ユーザー
日常的なタスクを実行するアカウント内のユーザー。 [詳細はこちら](#)

ルートユーザーのEメールアドレス

次へ

— AWSのご利用は初めてですか? —

新しいAWSアカウントの作成

AWSでのワークロードの 起動に役立つリソースセンター

どなたでも簡単にAWSを開始できるチュートリアルや
コース別ガイド、トレーニング等
ご活用ください

[詳細はこちら »](#)



ご自身のアカウントでログイン

Amazon.comのサインインについて

アマゾンウェブサービスは、Amazon.com アカウントの情報を使用して、ユーザーを識別し、アマゾンウェブサービスへのアクセスを許可します。このサイトのご利用は、

AWS マネジメントコンソール

画面を下へスクロール

AWS のサービス

サービスを検索する

名称、キーワード、頭文字を入力できます。

🔍 例 : *Relational Database Service*、データベース、RDS

▼ 最近アクセスしたサービス

📍 IoT Core

▼ すべてのサービス

📁 コンピューティング

EC2

Lightsail [🔗](#)

Lambda

Batch

Elastic Beanstalk

Serverless Application

Repository

📁 Machine Learning

Amazon SageMaker

Amazon CodeGuru

Amazon Comprehend

Amazon Forecast

Amazon Fraud Detector

Amazon Kendra

外出先でも AWS リソースに常時 接続



AWS コンソールモバイルアプリを
iOS または Android モバイルデバイ
スにダウンロードします。

[詳細はこちらから](#) [🔗](#)

AWS を試す

Amazon Redshift

データレイクにクエリを拡張できる、高速かつシンプルで、費用対効果の高いデータウェアハウス。 [詳細はこちらから](#) [🔗](#)

AWS Fargate を使ってサーバーレスコンテナを実行

[Trusted Advisor](#)[Control Tower](#)[AWS License Manager](#)[AWS Well-Architected Tool](#)[Personal Health Dashboard](#)[AWS Chatbot](#)[Launch Wizard](#)[AWS Compute Optimizer](#)

メディアサービス

[Elastic Transcoder](#)[Kinesis Video Streams](#)[MediaConnect](#)[MediaConvert](#)[MediaLive](#)[MediaPackage](#)[MediaStore](#)[MediaTailor](#)[Elemental Appliances & Software](#)[AppStream 2.0](#)[WorkDocs](#)[WorkLink](#)

IoT

[IoT Core](#)[FreeRTOS](#)[IoT 1-Click](#)[IoT Analytics](#)[IoT Device Defender](#)[IoT Device Management](#)[IoT Events](#)[IoT Greengrass](#)[IoT SiteWise](#)[IoT Things Graph](#)

ゲーム開発

[Amazon GameLift](#)

IoT Greengrass をクリック

ソリューションの構築

シンプルなウィザードと自動化されたワークフローで作業を開始しましょう。

[仮想マシンを起動する](#)[ウェブアプリを構築する](#)

us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/greengrassIntro

aws サービス リソースグループ

AWS IoT

モニタリング

▶ オンボード

▶ 管理

▼ Greengrass

開始方法

グループ

コア

デバイス

▶ 安全性

▶ 防御

▶ ACT

テスト

AWS IoT Greengrass へようこそ

AWS IoT Greengrass では、インターネット接続が利用できるときに AWS のサービスを活用しながら、デバイスでローカルに生成されるデータを処理できます。

[Raspberry Pi](#) の Greengrass で今すぐ使用を開始します。あるいは、[system requirements](#) および [list of compatible devices](#) を読みます。

クリック

グループの作成

Greengrass グループを作成する

グループを作成し、ワンステップで Core をプロビジョンするか、またはプロセスをステップバイステップに実行します。

フィードバック 日本語

© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 プライバシーポリシー 利用規約

Greengrass グループを設定する

グループの設定では、IoT レジストリでの Core デバイスのプロビジョニング、Core の証明書の取得、グループへの IAM ロールの割り当てが必要です。これらのステップに慣れていない場合は、デフォルトのグループ作成をお勧めします。最後に、Core デバイスに Greengrass ソフトウェアをインストールする必要があります。

デフォルトのグループ作成 (推奨)

このプロセスでは、レジストリで Core を自動的にプロビジョニングし、デフォルト設定を使用して新しいグループを生成し、新しい証明書とキーペアで Core を提供します。

クリック

デフォルト作成を使用

詳細グループ作成

このカスタマイズ可能なプロセスでは、Core のプロビジョニングを段階的に実行します。グループの IAM ロールと Core の証明書をカスタマイズして、キーペアを提供することができます。

カスタマイズ

キャンセル

デフォルト作成を使用

GREENGRASS グループを設定する

グループに名前を付ける

Greengrass グループは、Core デバイスを通じて相互に通信するようプログラムできる、ローカルデバイスと Lambda 関数のクラウドで設定されるマネージド型のコレクションです。グループには、最大 200 個のローカルデバイスを含めることができます。

グループ名

TestGG

① 名前を付ける
任意の名前でOK

(オプション) グループにタグを適用

② クリック

キャンセル

戻る

次へ

GREENGRASS グループを設定する

各グループには Core 機能が必要

各 Greengrass グループには、Core ソフトウェアを実行するデバイスが必要です。これにより、デバイス、ローカル Lambda 関数、および AWS クラウドコンピューティングサービス間の通信が可能になります。レジストリへの情報の追加は、デバイスを Greengrass Core としてプロビジョニングするための最初のステップです。

名前

Coreの名前は自動で振られます
そのままOK

オプション設定の表示 (後でも実行できます) ▾

クリック

[キャンセル](#)[戻る](#)[次へ](#)

GREENGRASS グループを設定する

グループ作成を確認

Greengrassグループの作成は自動で行われます

グループ作成のスピードアップとシンプル化のために、AWS IoT Greengrass は以下のプロセスを処理し、デフォルト設定を使用します。次のステップに進むと、以下のステップの完了が許可されます。

画面を下へスクロール

AWS IoT Greengrass は、デフォルト設定を使用して次のアクションをお客様に代わって行います。

クラウドで新しい Greengrass グループを作成する	詳細はこちら
IoT レジストリに新しい Core をプロビジョニングし、グループに追加する	詳細はこちら
Core のパブリックキーとプライベートキーセットを生成する	詳細はこちら
キーを使用して Core の新しいセキュリティ証明書を生成する	詳細はこちら
証明書にデフォルトのセキュリティポリシーを添付する	詳細はこちら

この操作は、AWS IoT Greengrass をインストールするためのプロセスの一部として、AWS IoT Greengrass は以下のプロセスを処理し、デフォルト設定を使用します。次のステップに進むと、以下のステップの完了が許可されます。

AWS IoT Greengrass は、デフォルト設定を使用して次のアクションをお客様に代わって行います。

クラウドで新しい Greengrass グループを作成する	詳細はこちら
IoT レジストリに新しい Core をプロビジョニングし、グループに追加する	詳細はこちら
Core のパブリックキーとプライベートキーセットを生成する	詳細はこちら
キーを使用して Core の新しいセキュリティ証明書を生成する	詳細はこちら
証明書にデフォルトのセキュリティポリシーを添付する	詳細はこちら
Core デバイスでストリームマネージャーを有効にする	詳細はこちら

キャンセル

クリック

グループと Core の作成

この操作は、AWS IoT Greengrass をインストールするためのプロセスの一部として、AWS IoT Greengrass は以下のプロセスを処理し、デフォルト設定を使用します。次のステップに進むと、以下のステップの完了が許可されます。

AWS IoT Greengrass は、デフォルト設定を使用して次のアクションをお客様に代わって行います。

- | | |
|--|------------------------|
| ✔️ クラウドで新しい Greengrass グループを作成する | 詳細はこちら |
| ✔️ IoT レジストリに新しい Core をプロビジョニングし、グループに追加する | 詳細はこちら |
| ✔️ Core のパブリックキーとプライベートキーセットを生成する | 詳細はこちら |
| ✔️ キーを使用して Core の新しいセキュリティ証明書を生成する | 詳細はこちら |
| ✔️ 証明書にデフォルトのセキュリティポリシーを添付する | 詳細はこちら |
| ✔️ Core デバイスでストリームマネージャーを有効にする | 詳細はこちら |

作成のアクションが順次実行され、画面が遷移します

キャンセル

🔄 グループと Core の作成

Core デバイスへの接続

最終ステップでは、Greengrass ソフトウェアをロードしてから、Core デバイスをクラウドに接続します。この時点でデバイスの接続を延期できますが、今すぐパブリックキーとプライベートキーをダウンロードする必要があります。これらを後で取得することはできません。

② 画面を下へスクロール

Core のセキュリティリソースのダウンロードと保存

このCoreの証明書	XXXXXXXXXX.cert.pem
パブリックキー	XXXXXXXXXX.public.key
プライベートキー	XXXXXXXXXX.private.key
Core 固有の設定ファイル	config.json

これらのリソースは tar.gz としてダウンロードしてください

① ダウンロードします

これらのリソースは tar.gz としてダウンロードしてください

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。

[ルート CA を選択](#)

Core デバイスでの前提条件をインストール

Java 8 ランタイム

ストリームマネージャーでの必須事項

[詳細はこちら](#)

最新の Greengrass Core ソフトウェアをダウンロードする

このソフトウェアをダウンロードすると、[Greengrass Core ソフトウェアのライセンス契約](#)に同意したと見なされます。Core に Greengrass をインストールするには、パッケージをダウンロードして[入門ガイド](#)の手順に従います。

[プラットフォームを選択](#)

クリックし、完了します

完了

AWS IoT

us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/greengrass/grouphub

aws サービス リソースグループ

AWS IoT

モニタリング

▶ オンボード

▶ 管理

▼ Greengrass

開始方法

グループ

コア

デバイス

▶ 安全性

▶ 防御

▶ ACT

テスト

Greengrass グループ (2) 情報

Greengrass グループは、デバイス、Lambda 関数、およびその他のローカルコンポーネントを整理します。

削除 グループを作成

名前、ID、または最新バージョン ID でグループを検索する

<input type="checkbox"/>	名前	ID	最終更新日
<input type="checkbox"/>	TestGG		
<input type="checkbox"/>			1 year ago

① 作成したグループがあることを確認

② クリック

フィードバック 日本語

© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 プライバシーポリシー 利用規約



GREENGRASS グループ

TestGG

デプロイされていません

アクション

デプロイ

グループ履歴の概要

デプロイメント別

サブスクリプション

この Greengrass グループのデプロイメントはまだありません

コア

デバイス

クリック

Lambda

リソース

コネクタ

タグ

設定



GREENGRASS グループ

TestGG

デプロイされていません

アクション

デプロイ

デバイス

クリック

デバイスの追加

サブスクリプション

コア

デバイス

Lambda

リソース

コネクタ

タグ

設定



デバイスに Core を認識させる

Greengrass では、デバイスをエッジに接続することができます。Greengrass デバイスは IoT Things と同じ SDK を共有します。サブスクリプションを使用して、相互に、Lambda 関数に、または直接クラウドサービスに接続することができます。

デバイスの追加

デバイスの追加

接続編にて作成したThingをGreengrassグループに使用します

レジストリから既存の IoT Thing を再利用するか、新しいレジストリ項目を作成してから、Greengrass グループに追加することで、Greengrass デバイスを作成できます。

新しいデバイスの作成

新しいデバイスを作成し、証明書、プライベートキー、およびパブリックキーを生成します。

新しいデバイスの作成

既存の IoT Thing をデバイスとして使用する

既存の IoT Thing をグループに追加できます。

クリック

IoT Thing を選択する

キャンセル

新規にデバイスを追加いただいても問題ありません
接続編 p22～p42まで実施してください

戻る

新しいデバイスの作成

デバイスの追加

既存の IoT Thing をデバイスとして使用する

既存の AWS IoT Thing を選択し、デバイスとして Greengrass グループにインポートできます。

② 画面を下へスクロール

モノの選択

モノの検索

 [redacted] test_egg_Thing eggtest_Thing Test_Thing sample_Thing TestThing

① 接続編にて作成した「TestThing」にチェックを付けます

既存の AWS IoT Thing を選択し、デバイスとして Greengrass グループにインポートできます。

モノの選択

- [Redacted]
- test_egg_Core
- ggtest_Core
- Test_Core
- sample_Core
- TestThing
- test_Core

キャンセル

戻る

完了

クリック



GREENGRASS グループ

TestGG

デプロイされていません

これでGreengrassグループにデバイスが追加されました

アクション

デプロイ

デバイス

デバイスの追加

サブスクリプション

コア

デバイス

Lambda

リソース

コネクタ

タグ

設定

TestThing

デバイス

ローカルシャドウのみ

クリック

us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/thing/TestThing

aws サービス リソースグループ

TestThing

モノ
TestThing
タイプなし

アクション

クリック

セキュリティ

モノの ARN 編集

モノの Amazon リソースネーム (ARN) はこのモノを一意に識別します。

arn:aws:iot:us-west-2:123456789012:thing/TestThing

タイプ

タイプなし

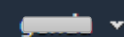
フィードバック 日本語

© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 プライバシーポリシー 利用規約



サービス

リソースグループ



オレゴン

サポート



TestThing

モノ

TestThing

タイプなし

アクション

詳細

証明書

セキュリティ

証明書を作成

他のオプションの表示

モノのグループ

請求グループ

シャドウ

操作

アクティビティ

ジョブ

違反

Defender メトリクス

xxxxxxxxxxxxxxxxxxxx...

クリック

TestThing > [redacted]

証明書

アクティブ

② クリック

① ドロップダウンを開いて、「有効化」がグレースアウト(有効になっている)ことを確認してください
有効になっていない場合は、有効化してください

アクション

詳細

ポリシー

モノ

コンプライアンス違反

証明書 ARN

証明書の Amazon リソースネーム (ARN) はこの証明書を一意に識別します。 [詳細はこちら](#)

arn:aws:iot:us-west-2:[redacted]

詳細

発行者

OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US

件名

CN=AWS IoT Certificate

作成日

5月 26, 2020, 15:20:42 (UTC+0900)

発効日



TestThing > [redacted]



証明書

アクティブ

アクション

詳細

ポリシー

ポリシー

モノ

コンプライアンス違
反

TestThing-policy



クリック



TestThing > [redacted] > TestThing-policy

ポリシー

TestThing-policy

アクション

概要

ポリシー ARN

証明書

ポリシー ARN は、このポリシーを一意に識別します。 [詳細はこちら](#)

バージョン

グループ

`arn:aws:iot:us-west-2:431099347647:policy/TestThing-policy`

コンプライアンス違反

作成日

5月 26, 2020, 15:20:42 (UTC+0900)

ポリシードキュメント

ポリシードキュメントでは、リクエストの権限を定義しています。 [詳細はこちら](#)

下へスクロール

ポリシードキュメントでは、リクエストの権限を定義しています。 [詳細はこちら](#)



バージョン 2

クリック

ポリシードキュメントの編集

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "greengrass:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

ポリシードキュメント

ポリシードキュメントでは、リクエストの権限を定義しています。詳細は

編集後、保存し、Greengrassグループへ戻ります

デフォルトとして設定

```

1 [
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "iot:*"
8       ],
9       "Resource": [
10        "*"
11      ]
12    },
13    {
14      "Effect": "Allow",
15      "Action": [
16        "greengrass:*"
17      ],
18      "Resource": [
19        "*"
20      ]
21    }
22  ]
23 ]

```

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "greengrass:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}

```

新しいバージョンとして保存。

Greengrassのアクセスを許可します



GREENGRASS グループ

TestGG

デプロイされていません

これでデバイス(モノ) Greengrass間の
アクセスが追加されました

アクション

デプロイ

デバイス

デバイスの追加

サブスクリプション

コア

デバイス

Lambda

リソース

コネクタ

タグ

設定

TestThing

デバイス

ローカルシャドウのみ

クリック



GREENGRASS グループ

TestGG

デプロイされていません

アクション

デプロイ

コア

サブスクリプション

コア

デバイス

Lambda

リソース

コネクタ

タグ

設定

TestGG_Core

CORE

クリック



GREENGRASS CORE

TestGG_Core

詳細

モノの ARN

編集

シャドウ

モノの Amazon リソースネーム (ARN) はこのモノを一意に識別します。

操作

接続

`arn:aws:iot:us-west-2:██████████:thing/TestGG_Core`

クリック

タイプ

🔍 タイプなし

ローカル接続の管理

ローカル Core 接続の情報

Greengrass は接続データをデバイスレジストリに保存します。この情報がないと、デバイスは Core で接続エンドポイントを検出できず、Core に接続できません。

[詳細はこちら](#)

別の接続エンドポイントの追加

別の接続の追加

クリック

キャンセル

更新

Core と Thing が構成するローカルネットワークの Core の初期設定を行います

KES IoT Logic GG対応では、同一GW内に Thing と Core が存在する構成です

ローカル接続の管理

ローカル Core 接続の情報

Greengrass は接続データをデバイスレジストリに保存します。この情報がないと、デバイスは Core で接続エンドポイントを検出できず、Core に接続できません。

[詳細はこちら](#)

エンドポイント

① 127.0.0.1 を入力

ポート

② 8883 を入力

オプションの接続情報

③ 画面を下へスクロール



サービス

リソースグループ



オレゴン

サポート

ポート

8883

オプションの接続情報

追加の接続メタデータ

削除

別の接続エンドポイントの追加

別の接続の追加

キャンセル

クリック

更新

クリック



GREENGRASS CORE

TestGG_Core

詳細

Core エンドポイント

[編集](#)

シャドウ

127.0.0.1

操作

ポート8883

接続



GREENGRASS グループ

TestGG

デプロイされていません

アクション

デプロイ

コア

サブスクリプション

コア

TestGG_Core

CORE

デバイス

Lambda

リソース

コネクタ

タグ

設定

クリック

コアの設定は完了です。

次に グループ の設定です。



GREENGRASS グループ

TestGG

デプロイされていません

アクション

デプロイ

グループのロール

サブスクリプション

TestGGグループにロールが割り当てられていません

コア

デバイス

グループ ID

Lambda

リソース

コネクタ

タグ

設定

認証機関 (CA) とローカル接続の設定

MQTT サーバー証明書の有効期間

Greengrass デバイスは、ローカルの MQTT サーバー証明書を使用して Greengrass コアで認証します。この証明書は 7 日毎に失効します。この設定を使用して、証明書の更新期間を制御することができます。

③ 画面を下へスクロール

選択しませぬ。 [詳細はこちら](#)

Greengrass コンテナ

コンテナなし

デフォルトの Lambda 実行設定を更新する

ストリームマネージャー

クリック

編集

ストリームマネージャーで、Core はデータストリームを取り込み、処理し、クラウドターゲットにエクスポートすることができます。 [詳細はこちら](#)

ステータス

有効

ストリームマネージャーはデフォルトで有効です
KES IoT Logic GGはストリームマネージャーを構成するJava8ランタイムを搭載しておりません
この機能を無効に変更します
(注意)これが有効になっていると、greengrassコアへのデプロイにてJava8ランタイムを読み込もうとしたりエラーになり、デプロイが失敗します

CloudWatch ログ設定

設定されている CloudWatch

編集

ローカルログ設定

設定されているローカルログはありません

編集

ストリームマネージャーを設定

ストリームマネージャー機能で、Core はデータストリームを取り込み、処理し、クラウドターゲットにエクスポートすることができます。ストリームマネージャーでは、Core デバイスに Java 8 ランタイムをインストールする必要があります。[詳細はこちら](#)

有効化

無効化

クリック

▶ カスタム設定

キャンセル

戻る

保存

ストリームマネージャーを設定

ストリームマネージャー機能で、Core はデータストリームを取り込み、処理し、クラウドターゲットにエクスポートすることができます。ストリームマネージャーでは、Core デバイスに Java 8 ランタイムをインストールする必要があります。[詳細はこちら](#)

- 有効化
- 無効化

キャンセル

戻る

保存

クリック



REENGRASS グループ

TestGG

デプロイされていません

クリック

グループの設定は完了です。

アクション

デプロイ

サブスクリプション

コア

デバイス

Lambda

リソース

コネクタ

タグ

設定

グループのロール

ロールの追加

TestGGグループにロールが割り当てられていません

グループ ID

認証機関 (CA) とローカル接続の設定

MQTT サーバー証明書の有効期間

Greengrass デバイスは、ローカルの MQTT サーバー証明書を使用して Greengrass コアで認証します。この証明書は 7 日毎に失効します。この設定を使用して、証明書の更新期間を制御することができます。

ストリームマネージャー設定が更新されました。



GREENGRASS グループ

TestGG

デプロイされていません

基本設定はここまです。

アクション

デプロイ

サブスクリプション

サブスクリプションの追加

サブスクリプション

コア

デバイス

Lambda

リソース

コネクタ

タグ

設定



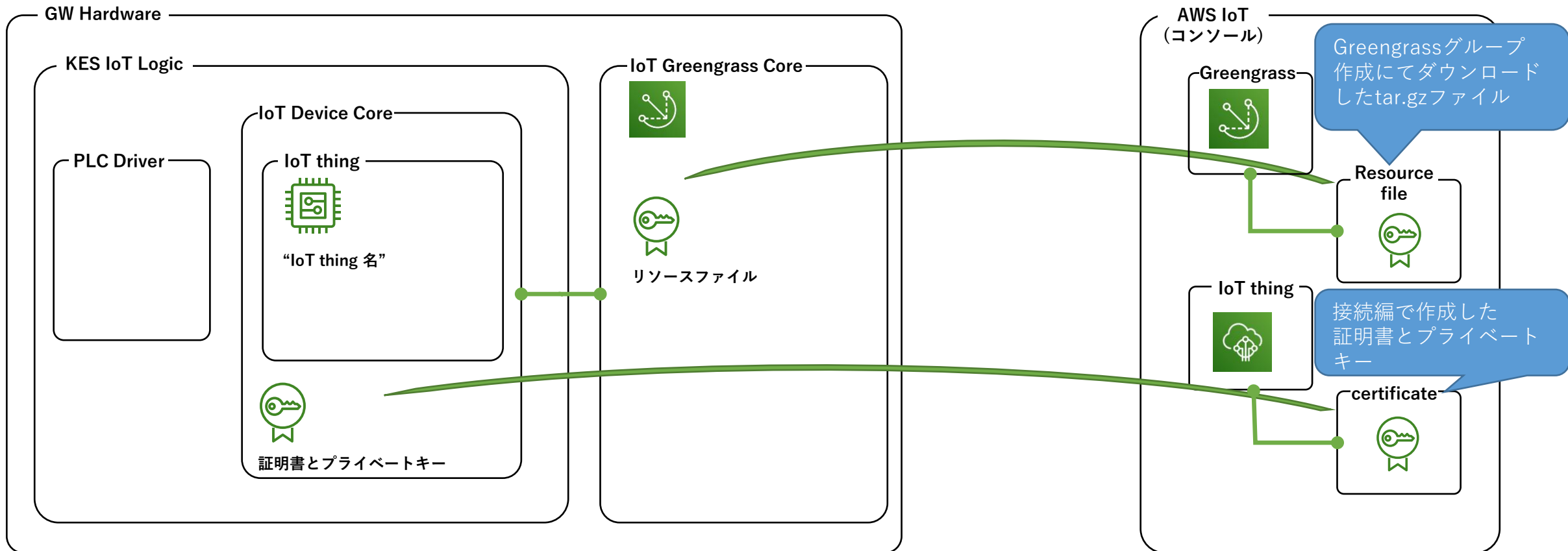
グループのアセットを接続しますか？

Greengrass Core は、MQTT プロトコルを使用して、デバイス、Lambda 関数、さらに AWS のサービス間でメッセージを送受信できます。これらのコンポーネントは、サブスクリプションを使用して連携できます。サブスクリプションは、より高いセキュリティと予測可能な連携を可能にするために事前定義されま

GG のリソースファイル類を GW に設定

Greengrass の 準備

これで Greengrass の基本設定が完了です。次にGWに証明書類を設定していきます。



192.168. [redacted]

192.168. [redacted]

ログイン

http://192.168. [redacted]

このサイトへの接続ではプライバシーが保護されません

ユーザー名

パスワード

ログイン キャンセル

①Chromeを起動
192.168.xxx.xxx
を開く

②ログイン

メニュー

☑ 初期設定

☑ デバイス設定

☑ ゲートウェイ設定

☁ 接続確認

👤 ユーザー設定

🔄 バックアップ



🔄 ゲートウェイ再起動

② 下へスクロール 次へをクリック

初期設定

設定 1 ~ 4 まで順番に設定して下さい。

1

初期設定①

2

初期設定②

3

初期設定③

4

初期設定④

上位接続ルート設定



3G/LTE設定

APN

(半角で入力して下さい)

ユーザー名

(半角で入力して下さい)

パスワード

(半角で入力して下さい)

クラウド設定以外は設定が完了している前提で進めます。

初期設定

設定 1 ~ 4 まで順番に設定して下さい。

1

初期設定①

2

初期設定②

3

初期設定③

4

初期設定④

メニュー

☑ 初期設定

☑ デバイス設

☑ ゲートウェイ設定

☁ 接続確認

👤 ユーザー設

🔄 バックアッ

①AWS IoT Greengrassを選択します

クラウド設定

クラウド選択

AWS IoT Greengrass

Greengrass Core設定

セキュリティリソースファイル

ファイル選択

②Greengrass Core作成にて
ダウンロードしたセキュリティリソ
ースファイルを選択します

Thing設定

エンドポイント

(半角で入力して下さい)

モノの名前

Thing設定

エンドポイント (半角で入力して下さい)

greengrass-ats.iot.[redacted].amazonaws.com

モノの名前

TestTing

D2C

publish

Publish トピック

pub_topic

Publish タグ

tag

Publish シーケンス

seq

Publish メッセージメンバー

msg

C2D

subscribe

Subscribe トピック

sub_topic

CA証明書

ファイル選択

エンドポイント

greengrass-ats.iot.リージョン.amazonaws.com

を設定します

他、

モノの名前 “AWS IoT コンソールにあるもの”

D2C “publish”

C2D “subscribe”

その他は任意の文字を設定します

Publish タグ

Publish シーケンス

Publish メッセージメンバー

C2D

Subscribe トピック

CA証明書

ファイル選択

プライベートキー

ファイル選択

クリック

保存

次へ

戻る

モノに設定している証明書を選択します

既に用意してあるモノに関連づいている証明書を選択してください

ここは変更しません

🔄 ゲートウェイ再起動

下へスクロール 次へをクリック



- メニュー
- ☑ 初期設定
- ☑ デバイス設定
- ☑ ゲートウェイ設定
- ☁ 接続確認
- 👤 ユーザー設定
- 📁 バックアップ

初期設定

設定 1 ~ 4 まで順番に設定して下さい。



時刻設定

NTPによる自動取得 ON OFF

タイムゾーン

サーバー名 (半角で入力して下さい)

確認周期(時間毎) (半角で入力して下さい)

アップロード共通データ設定

PLC収集時刻通知キー (半角で入力して下さい)

PLC収集時刻通知フォーマット (半角で入力して下さい)



ここは変更しません

🔄 ゲートウェイ再起動

KES IoT Logic



メニュー

☑ 初期設定

☑ デバイス設定

☑ ゲートウェイ設定

☁ 接続確認

👤 ユーザー設定

🗄️ バックアップ

初期設定

設定 1 ~ 4 まで順番に設定して下さい。

1

初期設定①

2

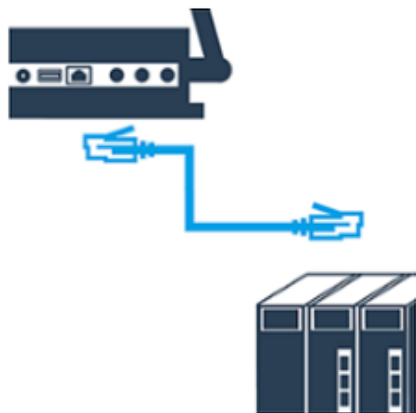
初期設定②

3

初期設定③

4

初期設定④



LAN設定

IPアドレス

(半角で入力して下さい 例 : 192.168.10.1)

192.168.

サブネットマスク

(半角で入力して下さい 例 : 255.255.255.0)

255.255.255.0

保存する

保存

次へ

戻る

ゲートウェイ再起動

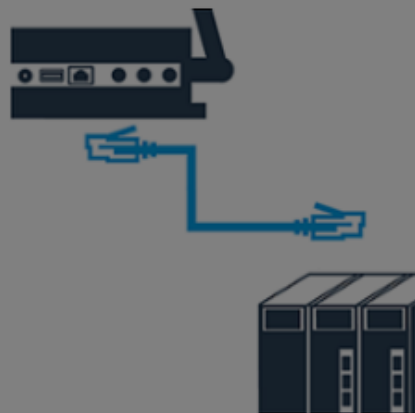
設定項目を反映するためにはゲートウェイの再起動が必要です。今すぐ再起動しますか？

“はい”を押します

4

初期設定④

メニュー

 初期設定 デバイス設定 ゲートウェイ設定 接続確認 ユーザー設定 バックアップ

LAN設定

IPアドレス

(半角で入力して下さい 例: 192.168.10.1)

192.168.

サブネットマスク

(半角で入力して下さい 例: 255.255.255.0)

255.255.255.0

KES IoT Logic

メニュー

☑ 初期設定

☑ デバイス設定

☑ ゲートウェイ設定

☁ 接続確認

👤 ユーザー設定

🔄 バックアップ

🔄 ゲートウェイ再起動

ページ移動

新しいネットワーク先のページに移動します。
ページ移動後に正しく表示されない場合は、ゲートウェイが再起動中である可能性があります。
ページ移動後、しばらく時間をおいてから再読み込みを実施して下さい。

“ページ移動”します
再起動のため2~3分待つ画面
に移動し、その後自動で切り
替わります

🔄 ページ移動

4

初期設定④

IPアドレス (半角で入力して下さい 例: 192.168.10.1)

192.168.

サブネットマスク (半角で入力して下さい 例: 255.255.255.0)

255.255.255.0

再起動後、ブラウザは閉じて
もらってOKです

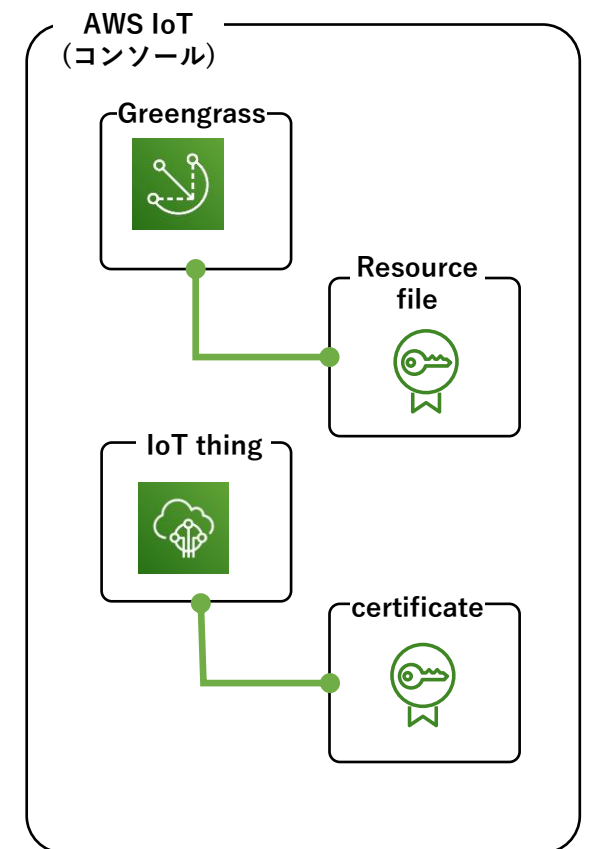
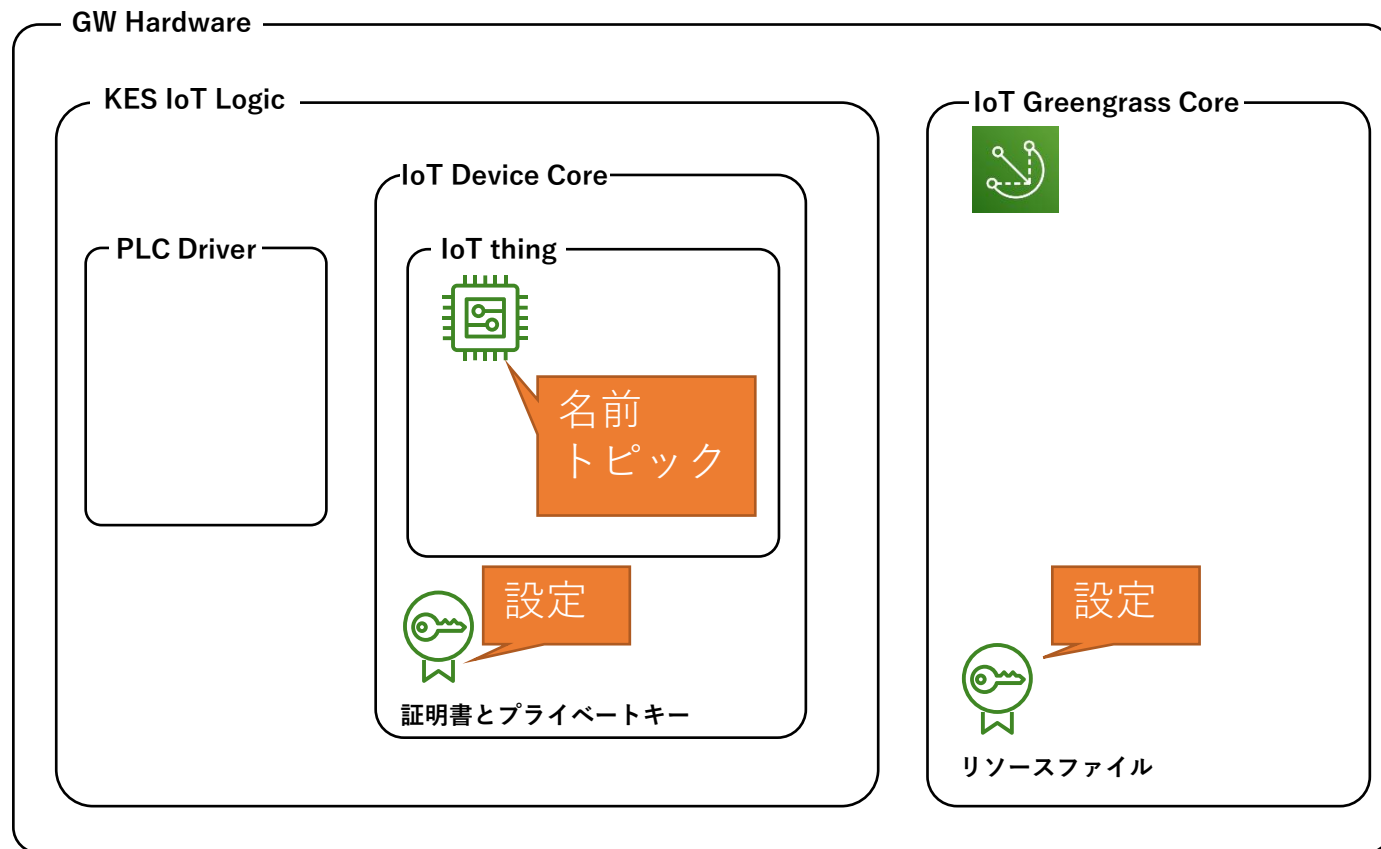
保存

次へ

戻る

KES IoT Logic の 準備

GW に証明書類の設定ができたことで、Greengrass グループに属する GW と デバイスになりました。次のステップより、Greengrassグループ内のデータの流れを構築します。



メッセージブローカーでのやり取り

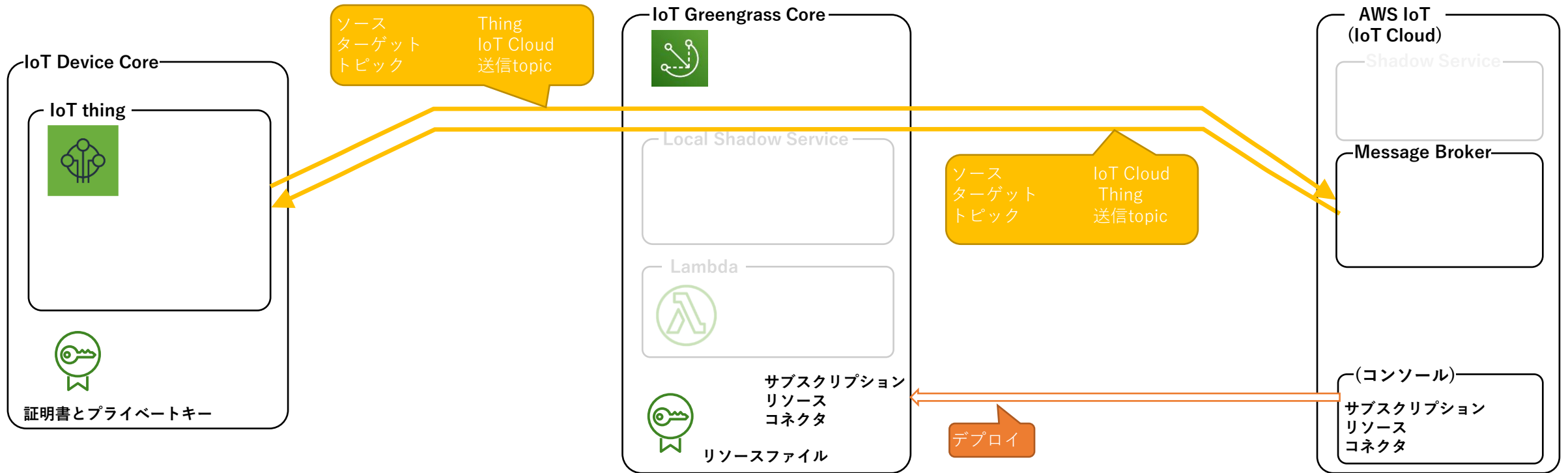
メッセージブローカーでのやり取り



IoT Thing と Cloud 間の通信

Greengrass Core の中継によりメッセージブローカーと通信する設定を行います。

Greengrassグループ



Greengrassグループにてこのデータフローを作成し、Greengrass Core へデプロイします。



GREENGRASS グループ

TestGG

デプロイされていません

アクション

クリック

サブスクリプションの追加

デプロイ

サブスクリプション

サブスクリプション

コア

デバイス

Lambda

リソース

コネクタ

タグ

設定



グループのアセットを接続しますか？

Greengrass Core は、MQTT プロトコルを使用して、デバイス、Lambda 関数、さらに AWS のサービス間でメッセージを送受信できます。これらのコンポーネントは、サブスクリプションを使用して連携できます。サブスクリプションは、より高いセキュリティと予測可能な連携を可能にするために事前定義されま

サブスクリプションの作成

ソースとターゲットの選択

サブスクリプションはソース、ターゲット、およびトピックで構成されます。ソースはメッセージの送信元です。ターゲットはメッセージの送信先です。最初のステップは、ソースとターゲットの選択です。

ソースの選択

オブジェクトが選択されていません

選択

クリック

ターゲットの選択

オブジェクトが選択されていません

選択

キャンセル

戻る

次へ

サブスクリプションの作成

ソースとターゲットの選択

サブスクリプションはソース、ターゲット、およびトピックで構成されます。ソースはメッセージの送信元です。ターゲットはメッセージの送信先です。最初のステップは、ソースとターゲットの選択です。

ソースの選択

オブジェクトが選択されていません

閉じる

サービス

デバイス

Lambda

コネクタ

🔍 検索

クリック

🔗 IoT Cloud

🔗 Local Shadow Service

サブスクリプションの作成

ソースとターゲットの選択

サブスクリプションはソース、ターゲット、およびトピックで構成されます。ソースはメッセージの送信元です。ターゲットはメッセージの送信先です。最初のステップは、ソースとターゲットの選択です。

ソースの選択

オブジェクトが選択されていません

閉じる


サービス

デバイス

Lambda

コネクタ

🔍 検索

 TestThing

選択

サブスクリプションの作成

ソースとターゲットの選択

サブスクリプションはソース、ターゲット、およびトピックで構成されます。ソースはメッセージの送信元です。ターゲットはメッセージの送信先です。最初のステップは、ソースとターゲットの選択です。

ソースの選択



TestThing

GREENGRASS デバイス

編集

ターゲットの選択

オブジェクトが選択されていません

閉じる

サービス

デバイス

Lambda

コネクタ



検索



IoT Cloud

選択

同じ要領で、ターゲットの選択では、サービス→IoT Cloud を選択

サブスクリプションの作成

ソースとターゲットの選択

サブスクリプションはソース、ターゲット、およびトピックで構成されます。ソースはメッセージの送信元です。ターゲットはメッセージの送信先です。最初のステップは、ソースとターゲットの選択です。

ソースの選択



TestThing

GREENGRASS デバイス

編集

ターゲットの選択



IoT Cloud

サービス

編集

キャンセル

戻る

次へ

クリック

サブスクリプションの作成

トピックでデータをフィルタリングする

ソースは、データをターゲットに公開します。トピックフィルターは、ターゲットが受信するデータを制限または制御するために使用されます。トピックフィルターが定義されていない場合、ソースからのすべてのメッセージがターゲットに送信されます。

ソース



TestThing

GREENGRASS デバイス

トピックのフィルター

トピックのフィルターの入力方法

pub_topic

① KES IoT Logic 初期設定② クラウド設定の Publishトピック に設定しているトピック名を入力

ターゲット



IoT Cloud

サービス

② クリック

サブスクリプションの作成

サブスクリプションの確認と保存

サブスクリプションが完了し、オブジェクトがこのグループで接続されました。新しいグループ定義を保存し、デプロイしてこの変更を有効にできます。



TestThing

GREENGRASS デバイス

pub_topic



IoT Cloud

サービス

戻る

完了

クリック



GREENGRASS グループ

TestGG

デプロイされていません

アクション

デプロイ

サブスクリプション

サブスクリプションの追加

サブスクリプション

コア

デバイス

Lambda

リソース

コネクタ

タグ

設定

ソース

ターゲット

トピック

TestThing

IoT Cloud

pub_topic



これで
KES IoT Logic -> AWS IoT
への topic "pub_topic"
が設定されました。

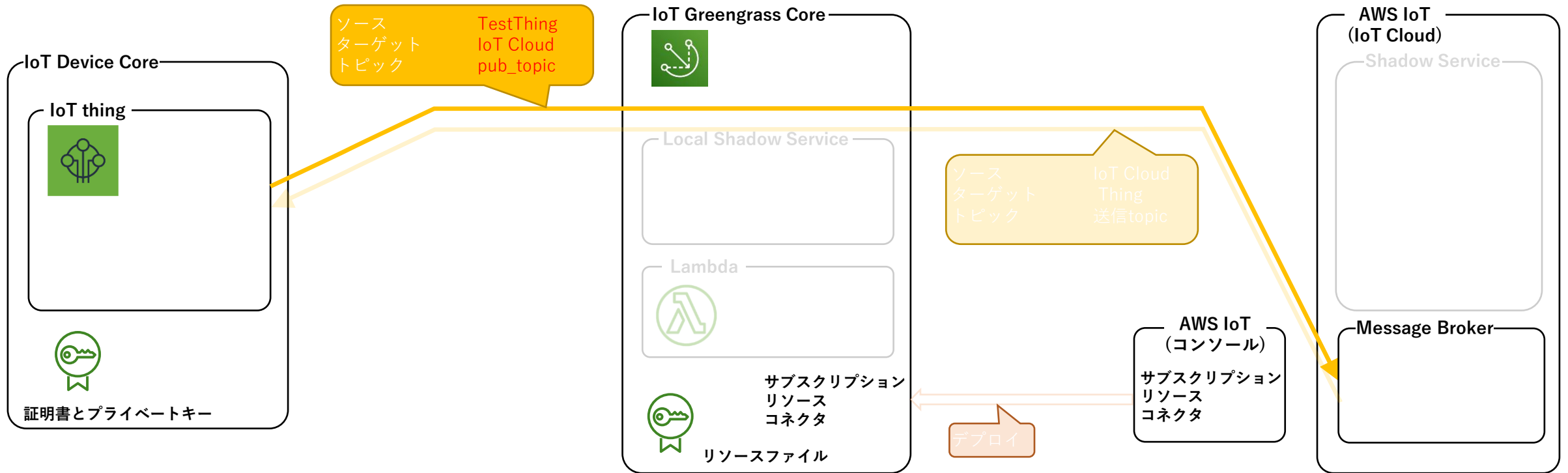
メッセージブローカーでのやり取り



IoT Thing と Cloud 間の通信

設定した部分は以下となります。

Greengrassグループ





GREENGRASS グループ

TestGG

デプロイされていません

クリック

アクション

サブスクリプションの追加

デプロイ

サブスクリプション

サブスクリプション

コア

デバイス

Lambda

リソース

コネクタ

タグ

設定

ソース

ターゲット

トピック

TestThing

IoT Cloud

pub_topic



サブスクリプションの作成

サブスクリプションの確認と保存

サブスクリプションが完了し、オブジェクトがこのグループで接続されました。新しいグループ定義を保存し、デプロイしてこの変更を有効にできます。

IoT Cloud

sub_topic

TestThing

①

先ほどとは逆の設定を行います
KES IoT Logic 初期設定② クラウド設定の
Subscribeトピック に設定しているトピック名

② クリック

戻る

完了



GREENGRASS グループ

TestGG

デプロイされていません

アクション

デプロイ

サブスクリプション

サブスクリプションの追加

サブスクリプション

コア

デバイス

Lambda

リソース

コネクタ

タグ

設定

ソース

ターゲット

トピック

IoT Cloud

TestThing

sub_topic



TestThing

IoT Cloud

pub_topic



これでメッセージのフローの設定
は完了です

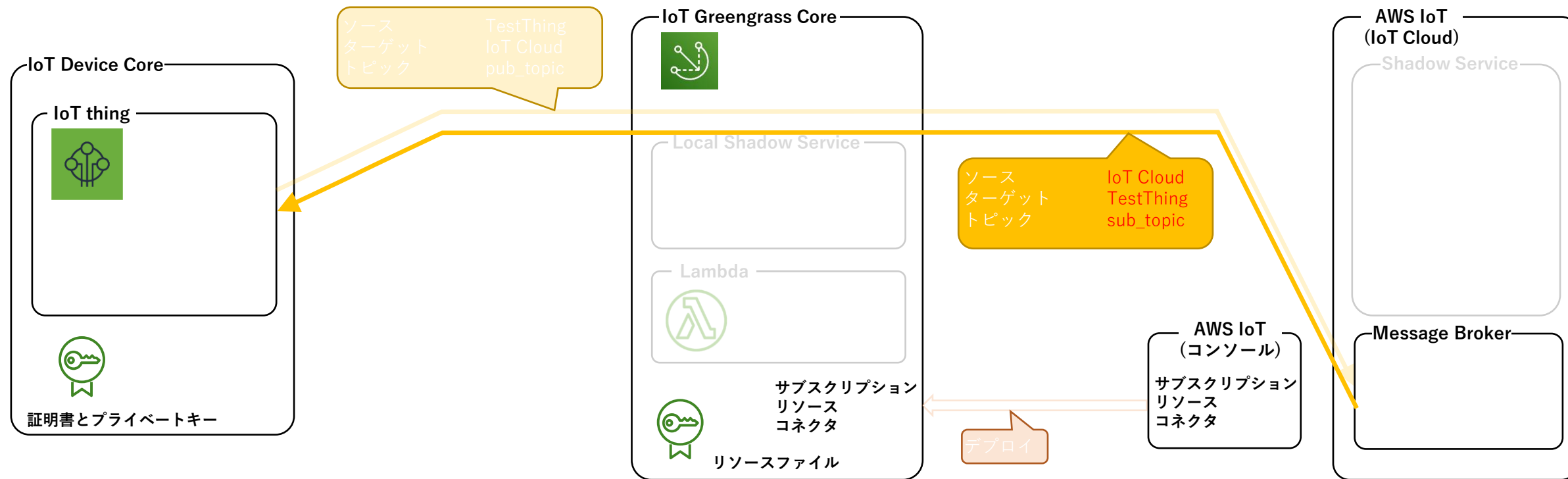
メッセージブローカーでのやり取り



IoT Thing と Cloud 間の通信

2つ目に設定した部分は以下となります。

Greengrassグループ





GREENGRASS グループ

TestGG

デプロイされていません

クリック

アクション

デプロイ
グループの削除
デプロイのリセット

デプロイ

サブスクリプション

サブスクリプション

コア

デバイス

Lambda

リソース

コネクタ

タグ

設定

ソース

ターゲット

トピック

IoT Cloud

TestThing

sub_topic

TestThing

IoT Cloud

pub_topic

これまでの設定を Greengrass Coreに
デプロイします

us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/greengrass/groups/

aws サービス リソースグループ

TestGG

● 正常に完了しました

完了するとメッセージが出ます

アクション

サブスクリプションの追加

ソース	ターゲット	トピック
IoT Cloud	TestThing	sub_topic
TestThing	IoT Cloud	pub_topic

これで Greengrass グループに設定した内容が GW にデプロイできました

フィードバック 日本語

© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 プライバシーポリシー 利用規約

192.168.xx.xx で開きます

Google

Google で検索または URL を入力

+

ショートカッ...

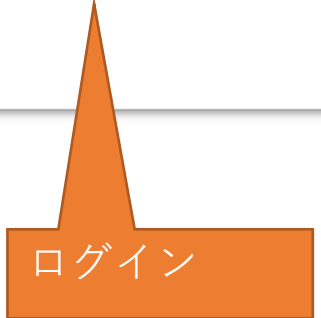
KES IoT Logic で動作を確認します

ログイン

http://192.168. [redacted]
このサイトへの接続ではプライバシーが保護されません

ユーザー名

パスワード





メニュー

☑ 初期設定

☑ デバイス設定

☑ ゲートウェイ設定

☁ 接続確認

👤 ユーザー設定

🗄️ バックアップ

クリック

初期設定

設定 1 ~ 4 まで順番に設定して下さい。

1

初期設定①

2

初期設定②

3

初期設定③

4

初期設定④

🔄 ゲートウェイ再起動

上位接続ルート設定

3G/LTE設定

APN

(半角で入力して下さい)

ユーザー名

(半角で入力して下さい)

パスワード

(半角で入力して下さい)

☑ 初期設定

☑ デバイス設定

☑ ゲートウェイ設定

☁ 接続確認

👤 ユーザー設定

🗄️ バックアップ



IPアドレス

(半角で入力して下さい)

⇒ 実行

接続機器の通信確認

選択したデバイスNo.に設定されたデバイス情報の通信状態を確認します。

デバイスNo.

⇒ 実行

通信確認結果

```
{
  "entryDatetime": "2019-12-13T22:30:49+09:00",
  "mac_address": "00:11:0c:1d:00:86",
  "device_type": "test_melsec",
  "event_type": "ver1",
  "ConnectError": 0,
  "D100": 1000,
  "D101": 1010,
  "D102": 1020,
  "D103": 1030,
  "D104": 1040
```

設定時刻の確認



本体に設定されている現在の時刻を表示します。
NTPによる時刻同期が動作しているか確認できます。



モニタリング

サンプル期間

時間範囲

クラウド側で通信の確認を行います

メッセージブローカー確認は“接続編”クラウド確認と同じ手順になります
GW側の設定は“MELSEC” ”D100“~”D149“ の範囲が設定されている前提です

(以下 “接続編”からの抜粋)

モニタリング

オンボード

管理

Greengrass

安全性

防御

ACT

テスト

テストへ移動



ソフトウェア

設定

学習

メッセージ



MQTT クライアント

iotconsole-1576244246402-3として接続

モニタリング

オンボード

管理

Greengrass

安全性

防御

ACT

テスト

ソフトウェア

設定

学習

サブスクリプション

トピックへサブスクライブする

トピックへの発行

サブスクリプション

デバイスは、トピックに関する MQTT メッセージを発行します。このクライアントを使用すると、トピックにサブスクライブしてこれらのメッセージを受信できます。

トピックのサブスクリプション

pub_topic

トピック...

① 初期設定のPublishトピックにて設定した文字を入力

メッセージキャプチャの最大数

100

② クリック

サービスの品質

0 - このクライアントは、メッセージを受信されるデバイスゲートウェイに対して認証されません。

1 - このクライアントは、メッセージを受信されるデバイスゲートウェイに対して認証されます。

MQTT クライアント

iotconsole-1576244246402-3として接続

サブスクリプション pub_topic エクスポート クリア 一時停止

トピックへサブスクライブする
トピックへの発行

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。

pub_topic [トピック...]

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

しばらく待ち、データ
が上がってくることを
確認する

Publishタグ、Publishシーケンス、
Publishメッセージメンバーに設定し
た文字が含まれています。

pub_topic 2019/12/13 22:39:41 エクスポート 非表示

```
{
  "tag": "pub_tag",
  "pub_seq": 0,
  "pub_member": {
```



AWS IoT

モニタリング

オンボード

管理

Greengrass

安全性

防御

ACT

テスト

ソフトウェア

設定

学習

サンプリングデバイス設定にて登録したデータがすべて入っていることを確認

```
D126 : 1260,  
"D127": 1270,  
"D128": 1280,  
"D129": 1290,  
"D130": 1300,  
"D131": 1310,  
"D132": 1320,  
"D133": 1330,  
"D134": 1340,  
"D135": 1350,  
"D136": 1360,  
"D137": 1370,  
"D138": 1380,  
"D139": 1390,  
"D140": 1400,  
"D141": 1410,  
"D142": 1420,  
"D143": 1430,  
"D144": 1440,  
"D145": 1450,  
"D146": 1460,  
"D147": 1470,  
"D148": 1480,  
"D149": 1490
```



MQTT クライアント

iotconsole-1576244246402-3として接続

モニタリング

オンボード

管理

Greengrass

安全性

防御

ACT

テスト

ソフトウェア

設定

学習

サブスクリプション

pub_topic

エクスポート

クリア

一時停止

トピックへサブスクライ

トピックへの発行

トピックの発行へ移動

トピックとメッセージを指定します。

pub_topic

トピック...

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

pub_topic

2019/12/13 22:46:01

エクスポート

非表示

```
{
  "tag": "pub_tag",
  "pub_seq": 0,
  "pub_member": {
```



モニタリング

オンボード

管理

Greengrass

安全性

防御

ACT

テスト

ソフトウェア

設定

学習

MQTT ペイロード表示

- JSON ペイロードを自動フォーマット (読みやすさが向上)
- ペイロードを文字列として表示 (より正確)
- 未加工のペイロードを表示 (16 進数)

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。

sub_topic

トピック...

```
1 {
2   "device_type": "test_melsec",
3   "dataName": "D100",
4   "value": "900",
5   "transactionId": "1"
6 }
```

① Subscribe トピック
に登録した文字を入力

③ クリック

② JSON フォーマットで、デフォルトの内容以下を書き変える

- "device_type" ユニット名 を入力
- "dataName" データ名 を入力
- "value" 任意の値 を入力 **ダブルクォート付**
- "transactionId" 任意の数値 を入力 **ダブルクォート付**

us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/test

aws サービス リソースグループ

トピックへサブスクライブする
トピックへの発行

pub_topic

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。

pub_topic トピック...

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

① Publishトピックへ移動

② スクロールさせ、
Publishに対する応答を探し確認

```
{  
  "tag": "pub_tag",  
  "pub_seq": 0,  
  "pub_member": {  
    "entryDatetime": "2019-12-13T22:53:27+09:00",  
    "device_type": "test_melsec",  
    "event_type": "ver1",  
    "transactionId": "1",  
    "returnCode": "OK",  
    "ConnectError": 0  
  }  
}
```

"transactionId"が一致するデータが応答データになります。
"returnCode"にて送信結果が返ります。

pub_topic 2019/12/13 22:53:26 エクスポート 非表示

フィードバック 日本語 © 2008 - 2019, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。 プライバシーポリシー 利用規約



モニタリング

オンボード

管理

Greengrass

安全性

防御

ACT

テスト

ソフトウェア

設定

学習

pub_topic

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

pub_topic 2019/12/13 22:53:31

エクスポート 非表示

```
{
  "tag": "pub_tag",
  "pub_seq": 0,
  "pub_member": {
    "entryDatetime": "2019-12-13T22:53:30+09:00",
    "mac_address": "00:11:0c:1d:00:86",
    "device_type": "test_melsec",
    "event_type": "ver1",
    "ConnectError": 0,
    "D100": 900,
    "D101": 1010,
    "D102": 1020,
    "D103": 1030,
    "D104": 1040,
    "D105": 1050,
    "D106": 1060,
    "D107": 1070,
    "D108": 1080
```

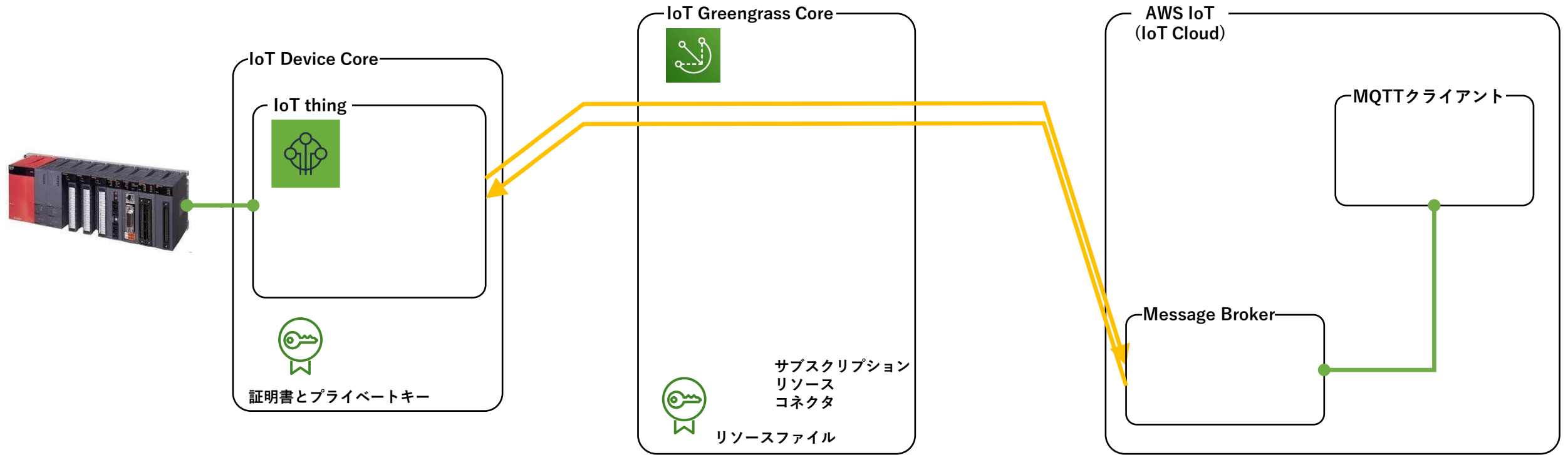
次のPublishデータにて、
PLCに書き込んだデータを
読み込んでいることを確認

メッセージブローカーでのやり取り



IoT Thing と Cloud 間の通信

これで Greengrass Core の中継によるメッセージブローカーとの通信ができました。



お疲れさまでした！
これでGreengrass設定編は完了です！



©Kanazawa Engineering Systems Inc.