



# AWS IoT Greengrass V2 の Tips & トラブルシューティ ング

2021/03/10

AWS IoT Deep Dive #3



## 自己紹介

名前：

市川 純（いちかわ じゅん）

@sparkgene

所属：

アマゾン ウェブ サービス ジャパン株式会社

デジタルトランスフォーメーション本部

プロトタイピングソリューション アーキテクト

担当：

IoT に関するプロトタイピング



# 本セッションの内容

## 紹介すること

- AWS IoT Greengrass を使い始めるにあたって、知っておきたい注意点や Tips について

## 紹介しないこと

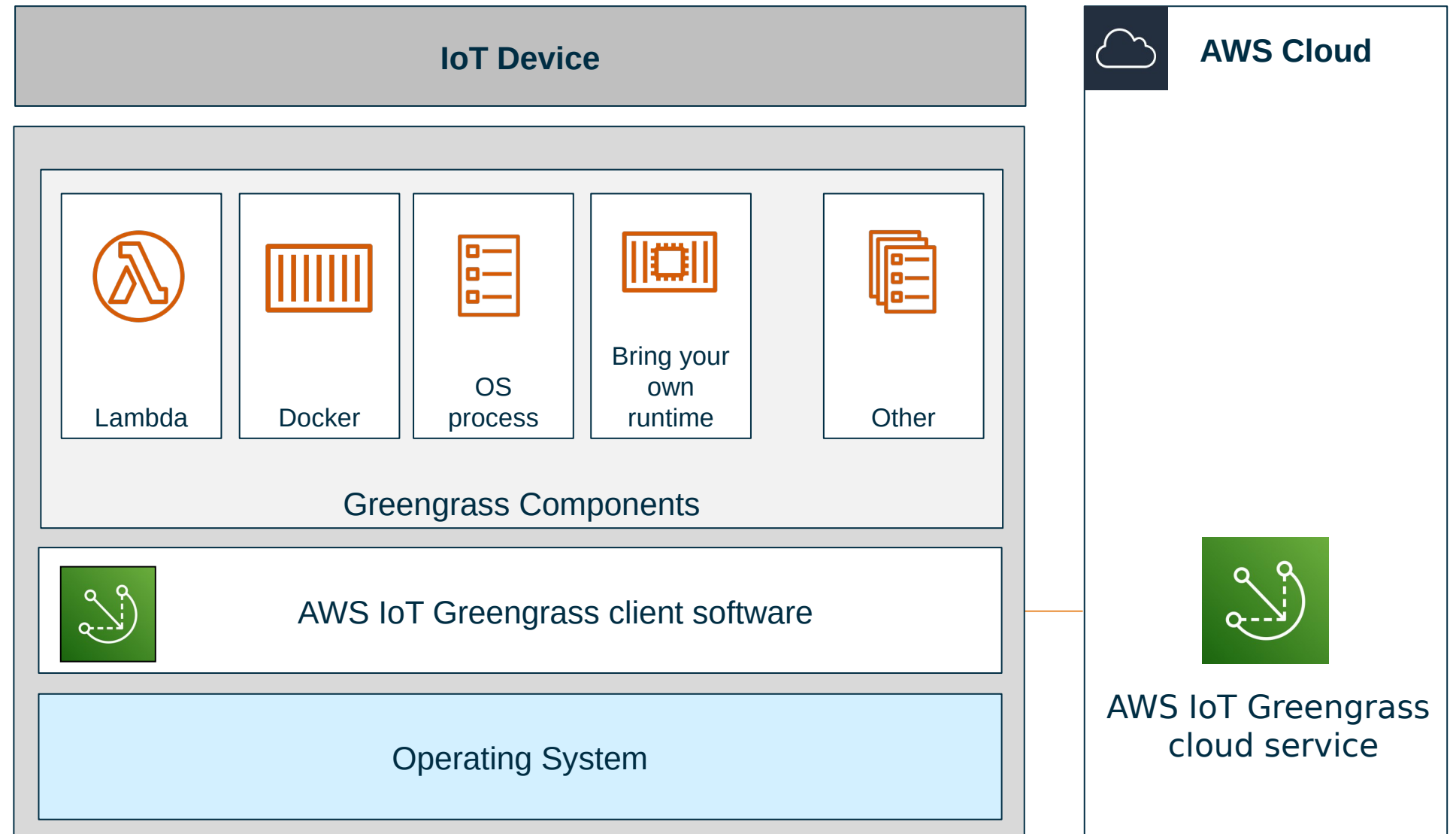
- AWS IoT Greengrass V2 の機能の詳細など
- AWS IoT Core が関連するような話（前のセッションで紹介）

# AWS IoT Greengrass - デバイスのソフトウェアの開発を早く

デバイス向けのソフトウェア開発を速くすることで、本番への投入を速くし、開発コストを下げます

## 特徴

- Greengrass クライアントソフトウェア自体がオープンソース
- デバイス側での開発ツール - ローカル向けの CLI とコンソールで開発を速く
- HW、OS、ランタイムの選択にまたがる移植性
- ビルド済みまたはカスタムソフトウェアを使用したモジュラー開発



# Greengrass V2 の Tips& トラブルシューティング

- プロビジョニング
- IoT Policy、IAM 権限
- ログの確認
- 設定情報の確認
- コンポーネントのデプロイ、OTA
- ローカルでバグコンソール
- Recipe ファイルの注意点

# Greengrass V2 の Tips& トラブルシューティング

- プロビジョニング
- IoT Policy、IAM 権限
- ログの確認
- 設定情報の確認
- コンポーネントのデプロイ、OTA
- ローカルでバグコンソール
- Recipe ファイルの注意点

# プロビジョニング - Quick セットアップ

1. Greengrass Core ソフトウェアをダウンロード
2. 環境変数からクレデンシャルを指定
3. 引数を指定してプロビジョニング

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
-jar ./GreengrassCore/lib/Greengrass.jar \
--aws-region ap-northeast-1 \
--thing-name MyGreengrassCore \
--thing-group-name MyGreengrassCoreGroup \
--tes-role-name MyGreengrassV2TokenExchangeRole \
--tes-role-alias-name MyGreengrassCoreTokenExchangeRoleAlias \
--component-default-user ggc_user:ggc_group \
--provision true \
--setup-system-service true \
--deploy-dev-tools true
```

引数の情報でプロビジョニングさせる

サービスとして登録

greengrass-cli をデプロイ

<https://docs.aws.amazon.com/greengrass/v2/developerguide/quick-installation.html>

# プロビジョニング - サービスとして登録しなかった場合

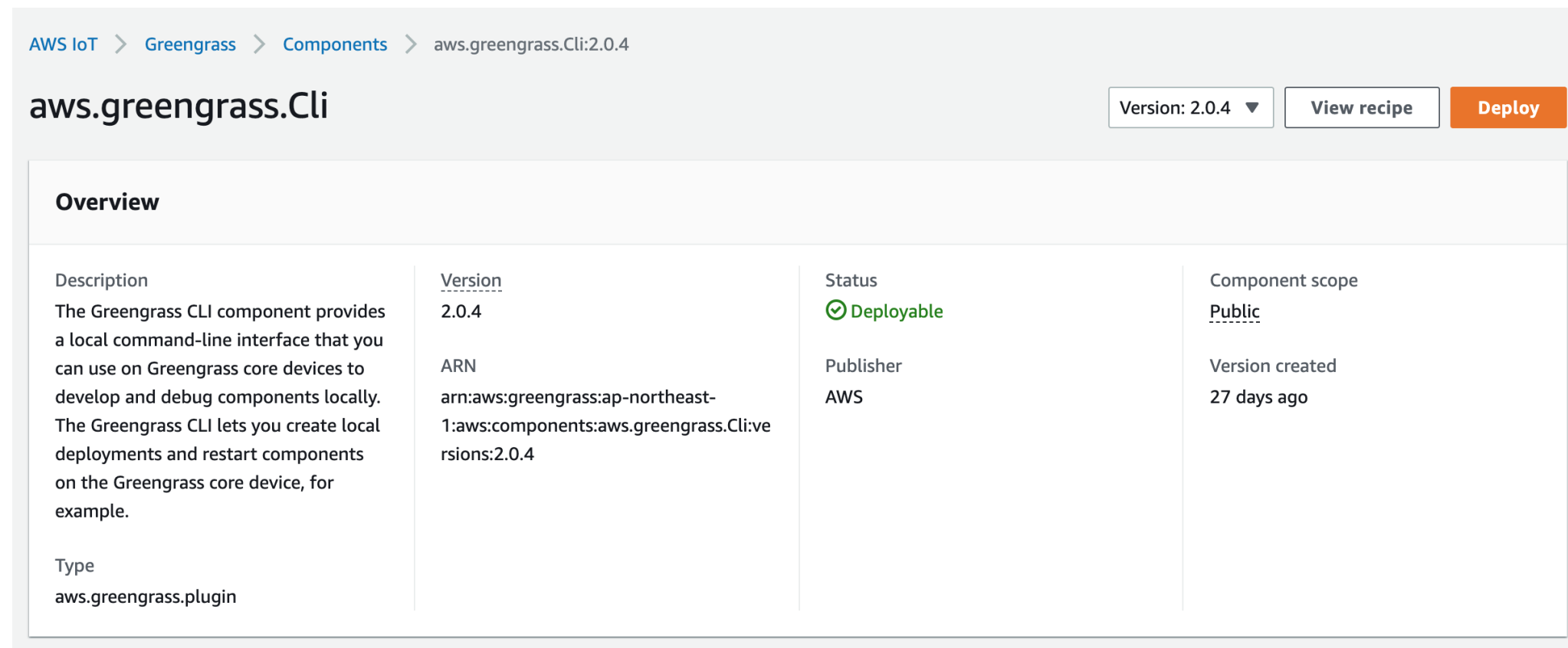
```
sudo /greengrass/v2/alts/current/distro/bin/loader
```

このコマンドで、Greengrass が起動できます。



# プロビジョニング – greengrass-cli を入れてなかった場合

greengrass-cli 用のコンポーネントがあるので、これをデプロイすれば良い  
(プロビジョニング時に含めた場合は、実は裏で Deployments が作成され、デプロイが実行されてから初めて使えるようになる)



The screenshot shows the AWS IoT Greengrass console interface for the component 'aws.greengrass.Cli'. The breadcrumb navigation is 'AWS IoT > Greengrass > Components > aws.greengrass.Cli:2.0.4'. The component name 'aws.greengrass.Cli' is displayed on the left, with a 'Version: 2.0.4' dropdown and 'View recipe' and 'Deploy' buttons on the right. Below this is an 'Overview' section with a table of metadata.

Overview			
<b>Description</b> The Greengrass CLI component provides a local command-line interface that you can use on Greengrass core devices to develop and debug components locally. The Greengrass CLI lets you create local deployments and restart components on the Greengrass core device, for example.	<b>Version</b> 2.0.4	<b>Status</b> ✔ Deployable	<b>Component scope</b> Public
<b>Type</b> aws.greengrass.plugin	<b>ARN</b> arn:aws:greengrass:ap-northeast-1:aws:components:aws.greengrass.Cli:versions:2.0.4	<b>Publisher</b> AWS	<b>Version created</b> 27 days ago

# プロビジョニング - 一つ一つ順番にセットアップ

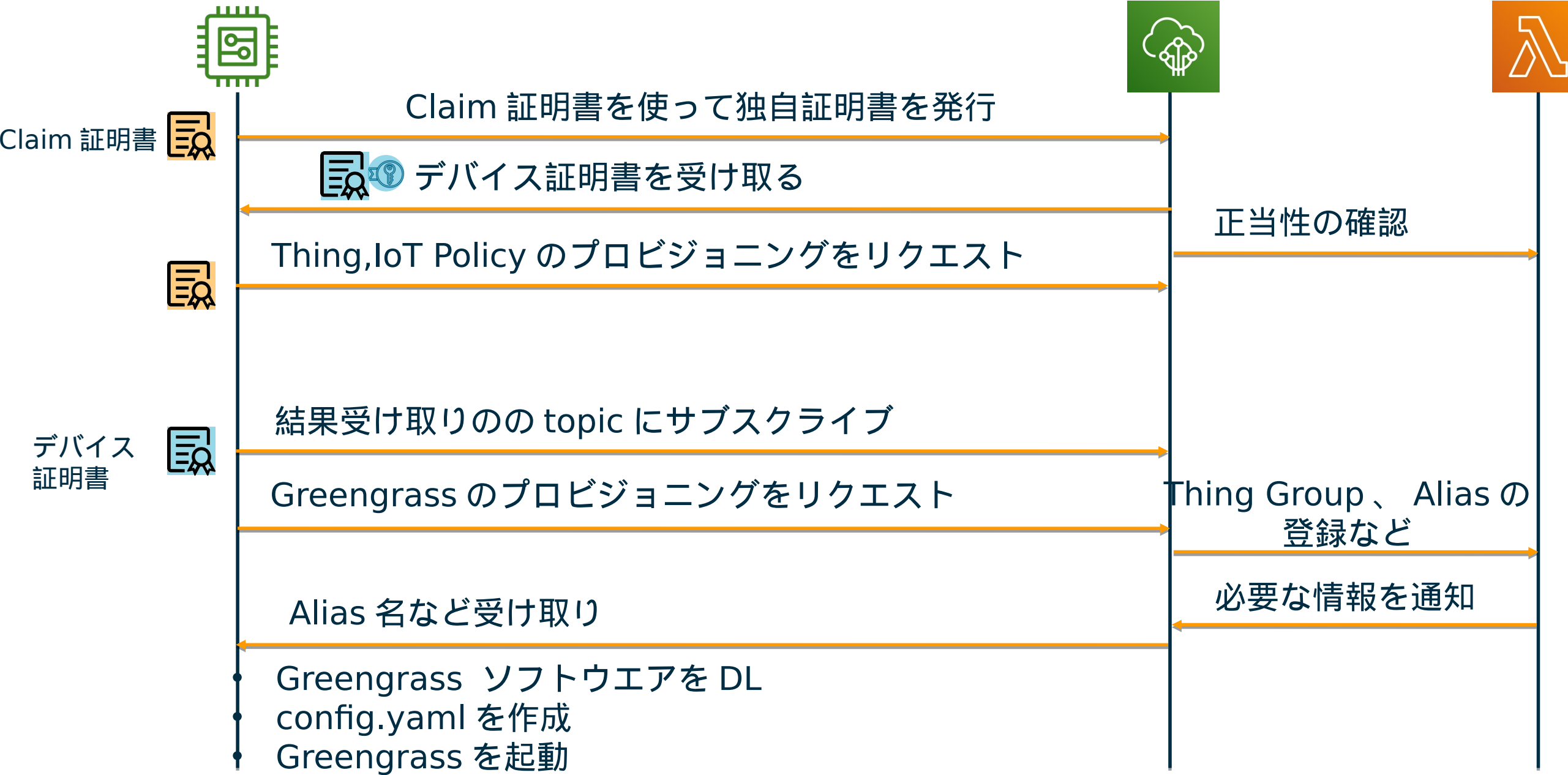
1. Thing を作成
2. 秘密鍵と証明書を作成
3. IoT Policy を作成
4. 証明書に Thing と IoT Policy を紐付け
5. Thing Group を作成し、Thing を紐付け
6. TES 用の IAM Role と Policy を作成し、Role Alias を作成
7. IoT Policy に Role Alias に assume role 出来る Policy を追加
8. Greengrass の config.yaml を作成
9. Greengrass Core ソフトウェアをダウンロード
10. 引数に config.yaml を指定して起動

# プロビジョニング - フリートプロビジョニングと組み合わせ

Greengrass Device

AWS IoT Core

AWS Lambda



注意) フリートプロビジョニングの仕組みを使い、この仕組みを自作する必要があります。  
AWS IoT Device SDK にフリープロビジョニングの実装サンプルがありますので、参考にしてください。  
フリープロビジョニングの詳細は、こちらから <https://aws.amazon.com/jp/blogs/news/aws-iot-deep-dive-1/>

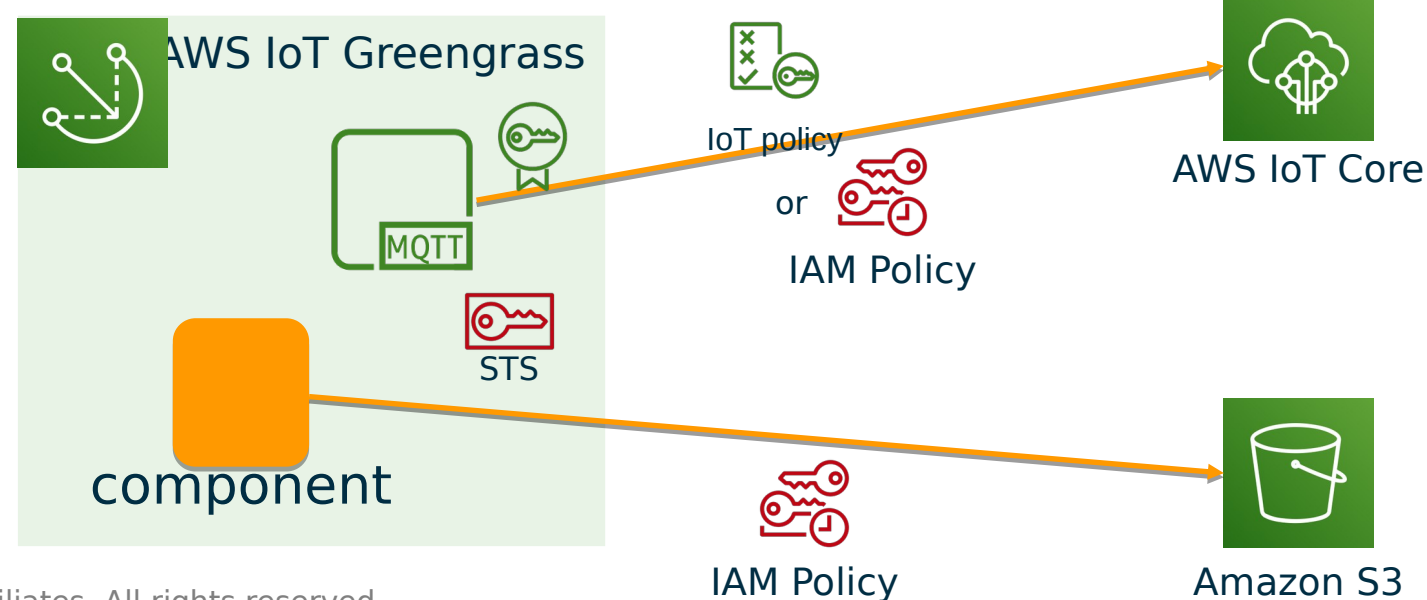


# Greengrass V2 の Tips& トラブルシューティング

- プロビジョニング
- IoT Policy、IAM 権限
- ログの確認
- 設定情報の確認
- コンポーネントのデプロイ、OTA
- ローカルでバグコンソール
- Recipe ファイルの注意点

# Greengrass V2 で設定するポリシー

- 証明書に紐づく IoT Policy
  - Greengrass 自身が IoT Core の MQTT ブローカーと通信するための権限
  - この証明書を使って TES の AliasRole に対して Assume Role が出来る権限
- TES(Token Exchange Service) に紐づく IAM Policy
  - Proxy 経由で Greengrass サービスや IoT Core と通信する場合
  - AWS のサービスを利用するために必要な権限
    - 自作のコンポーネントをデプロイする際の S3 のバケットに対してとか



# Greengrass V2 で Proxy を使う場合の注意

- Greengrass が Proxy を通してクラウドに繋げる場合は、Nucleus の設定でできる
- MQTT over 443 となる場合、証明書を使った認証ではなく、IAM Role を使った認証が採用されます。
- 先程の説明にあった、TES で利用している Role の権限が適用されますので、**iot** アクションの追加を忘れずに。
  - <https://docs.aws.amazon.com/greengrass/v2/developerguide/configure-greengrass-core-v2.html#configure-network-proxy>

# Greengrass V2 の Tips& トラブルシューティング

- プロビジョニング
- IoT Policy、IAM 権限
- ログの確認
- 設定情報の確認
- コンポーネントのデプロイ、OTA
- ローカルでバグコンソール
- Recipe ファイルの注意点

# ログの確認

- Greengrass のインストール先が `/greengrass/v2` であれば `/greengrass/v2/logs` に出力される
- コンポーネントのログは、`コンポーネント名.log` というファイルが作成される

```
ls -la /greengrass/v2/logs/
-rw-r--r--  1 root root 696547 Mar  3 10:56 Greengrass_HelloWorld_QcnGW.log
-rw-r--r--  1 root root 256272 Mar  2 07:59
Greengrass_HelloWorld_QcnGW_2021_03_02_07_0.log
-rw-r--r--  1 root root      0 Feb 26 07:15 aws.greengrass.LambdaLauncher.log
-rw-r--r--  1 root root      0 Feb 26 07:15 aws.greengrass.LambdaRuntimes.log
-rw-r--r--  1 root root      0 Feb 26 07:22
aws.greengrass.LegacySubscriptionRouter.log
-rw-r--r--  1 root root    339 Feb 26 07:12 aws.greengrass.Nucleus.log
-rw-r--r--  1 root root 111079 Mar  3 09:48 greengrass.log
-rw-r--r--  1 root root 108629 Mar  2 07:46 greengrass_2021_03_02_07_0.log
-rw-r--r--  1 root root      0 Feb 26 07:12 main.log
```



# Greengrass V2 の Tips& トラブルシューティング

- プロビジョニング
- IoT Policy、IAM 権限
- ログの確認
- **設定情報の確認**
- コンポーネントのデプロイ、OTA
- ローカルでバグコンソール
- Recipe ファイルの注意点

# 設定情報の確認

- 設定情報は、v1 と違い手動で変更することはありません
- 保存先は、 `/greengrass/v2/config/`
- ローカルデプロイ、クラウドからのデプロイなどで設定変更が行われると、`config.tlog` に履歴が追加されていきます。

```
" :1614323517282, "TP": ["system", "rootCaPath"], "W": "changed", "V": "/greengrass/v2/rootCA.pem"}
" :1614323517281, "TP": ["system", "thingName"], "W": "changed", "V": "component_test"}
" :1614323517282, "TP": ["system", "privateKeyPath"], "W": "changed", "V": "/greengrass/v2/privKey.key"}
" :1614323517282, "TP": ["system", "certificateFilePath"], "W": "changed", "V": "/greengrass/v2/thingCert.crt"}
" :1, "TP": ["system", "rootpath"], "W": "changed", "V": "/greengrass/v2"}
" :1614671164813, "TP": ["services", "aws.greengrass.LambdaRuntimes", "componentType"], "W": "changed", "V": "GENERAL"}
" :1614671164813, "TP": ["services", "aws.greengrass.LambdaRuntimes", "version"], "W": "changed", "V": "2.0.3"}
" :1614671164813, "TP": ["services", "aws.greengrass.LambdaRuntimes", "dependencies"], "W": "changed", "V": []}
" :0, "TP": ["services", "DeploymentService", "version"], "W": "changed", "V": "0.0.0"}
```

- Greengrass のプロセスを停止すると、`effectiveConfig.yaml` にダンプされます

# Greengrass V2 の Tips& トラブルシューティング

- プロビジョニング
- IoT Policy、IAM 権限
- ログの確認
- 設定情報の確認
- コンポーネントのデプロイ、OTA
- ローカルでバグコンソール
- Recipe ファイルの注意点

# コンポーネントのデプロイ - ローカル

以下のようなディレクトリを用意します

```
.
├── artifacts
│   ├── com.example.Pub
│   │   └── 1.0.0
│   │       └── publisher.py
│   └── recipes
│       └── com.example.Pub-1.0.0.yaml
```

- artifacts には、ソースコードや実行ファイルを置きます
- recipes には、レシピファイルを置きます

# コンポーネントのデプロイ - ローカル

## デプロイ

```
sudo /greengrass/v2/bin/greengrass-cli --ggcRootPath /greengrass/v2 deployment create --recipeDir ./recipes \ --artifactDir ./artifacts \ --merge "com.example.Pub=1.0.0"
```

## 状況を確認

```
sudo /greengrass/v2/bin/greengrass-cli deployment status -i デプロイ ID
```

## デプロイしたものを削除

```
sudo /greengrass/v2/bin/greengrass-cli --ggcRootPath /greengrass/v2 deployment create --recipeDir ./recipes \ --remove com.example.Pub
```

# コンポーネントのデプロイ - クラウドから

クラウドからコンポーネントをデプロイする場合に必要なのは

- artifact(ソースコードや実行ファイル)をS3にアップロード
- TESのPolicyにS3のバケットに対してs3:GetObjectを許可するポリシーを追加
- recipeファイルにartifactの情報を追加

```
Lifecycle:
```

```
...
```

```
Artifacts:
```

```
- URI:
```

```
s3://<bucket名>/artifacts/com.example.Pub/1.0.0/publisher.py
```

```
aws greengrassv2 create-component-version \  
--inline-recipe-fileb://recipes/com.example.Pub-1.0.0.yaml
```

CLIのコマンドは、 **greengrassv2**

# コンポーネントのデプロイ - 複数の ThingGroup

- V2 より Thing Group に対してもデプロイを実行できるようになりました。
- デプロイ自体は、デプロイメントという単位で作成され、その中で指定されたコンポーネントがデプロイされます
- もし、複数の Thing Group に登録してデプロイしたりすると、コンポーネントの管理がおかしくなるため、**複数のデプロイメントを作らないようにしましょう**

# コンポーネントのデプロイ - 挙動

- v2 ではモジュールという考え方が登場しました（ようは、コンポーネント）
- コンポーネント単位で追加したり、することで不要な機能が含まれず、リソースの節約にもなるという考えです
- コンポーネントをデプロイすると、対象のコンポーネントだけが再起動します。
- ただし、Nucleus(Greengrass のコンポーネント) をデプロイした場合は、全体が再起動します



# コンポーネントのデプロイ (OTA) - Nucleus の扱い

アクション	デプロイの定義	Nucleus のアップデート
既存のデプロイメントの対象となっている Thing Group に新しい Thing が追加された	<ul style="list-style-type: none"><li>• Nucleus が明示的に定義されていない</li><li>• AWS 提供のコンポーネントが対象に含まれている (カスタムコンポーネントから依存されている)</li></ul>	<ul style="list-style-type: none"><li>• 新しく追加されたデバイスには、依存関係を満たすように最新の Nucleus がデプロイされる</li><li>• 既存のデバイスには何も影響はない</li></ul>
	<ul style="list-style-type: none"><li>• Nucleus が明示的に定義されている</li></ul>	<ul style="list-style-type: none"><li>• 新しく追加されたデバイスには、指定されているバージョンの Nucleus がデプロイされる</li><li>• 既存のデバイスには何も影響はない</li></ul>
新しいデプロイメントが作成された、または既存のデプロイメントが更新された	<ul style="list-style-type: none"><li>• Nucleus が明示的に定義されていない</li><li>• AWS 提供のコンポーネントが対象に含まれている (カスタムコンポーネントから依存されている)</li></ul>	<ul style="list-style-type: none"><li>• デプロイメントの対象となるデバイス全てに対して、依存関係を満たすように最新の Nucleus がデプロイされる</li></ul>
	<ul style="list-style-type: none"><li>• Nucleus が明示的に定義されている</li></ul>	<ul style="list-style-type: none"><li>• 指定されているバージョンの Nucleus がデプロイされる</li></ul>

<https://docs.aws.amazon.com/greengrass/v2/developerguide/update-greengrass-core-v2.html>

# Greengrass V2 の Tips& トラブルシューティング

- プロビジョニング
- IoT Policy、IAM 権限
- ログの確認
- 設定情報の確認
- コンポーネントのデプロイ、OTA
- ローカルでバグコンソール
- Recipe ファイルの注意点

# ローカルデバッグコンソールのデプロイ

The screenshot shows the AWS IoT Greengrass console interface. The left sidebar has the 'Components' menu item highlighted. The main area shows a list of public components, with 'aws.greengrass.LocalDebugConsole' selected. A modal window titled 'Add to deployment' is open, showing the option to 'Add to existing deployment' and a table of existing deployments. The 'Deployment for 20210118\_DEMO\_Group' is selected in the table. A 'Deploy' button is visible in the top right of the component details view.

1) local debug console を選択

2) local debug console を deploy

3) 既存の deployment があればそれを選択

Name	Version	Oper...
aws.greengrass.SNS	2.0.3	All
aws.greengrass.LegacySubscriptionRouter	2.0.3	linux
variant.ImageClassification.ModelStore	2.0.4	linux
aws.greengrass.KinesisFirehose	2.0.3	All
aws.greengrass.Cli	2.0.3	linux, darwin
aws.greengrass.TokenExchangeService	2.0.3	All
aws.greengrass.Nucleus	2.0.7	
aws.greengrass.DIImageClassification	2.0.1	
aws.greengrass.LocalDebugConsole	2.0.3	
aws.greengrass.LogManager	2.0.1	

Deployment	Target	Status	Deployment created
Deployment for 20210118_DEMO_Group	20210118_DEMO_Group	Active	7 minutes ago

# ローカルデバッグコンソールのデプロイ

## Configure components - optional

You can configure the version and configuration parameters of each component to deploy. Components define default configuration parameters that you can customize in this deployment.

### Selected components (2)

Find by name, operating system, or architecture

Configure component

Name	Version	Modified?
<input checked="" type="radio"/> aws.greengrass.LocalDebugConsole	2.0.3	-
<input type="radio"/> aws.greengrass.Cli	2.0.3	-

4) local debug console を選択して、  
設定変更の画面に

5) デフォルトの設定を変更してデプ  
ロイ

### Configuration to merge

The configuration to merge with the configuration on each core device. The deployment merges this JSON object after it resets the values that you specify in the list of reset paths. [Learn more](#)

```
1 {  
2   "bindHostname": "0.0.0.0",  
3   "port": "8080",  
4   "websocketPort": "8081"  
5 }
```

Cancel

Confirm

# local debug console のパスワード取得

```
sudo /greengrass/v2/bin/greengrass-cli get-debug-password
```

## local debug console を開く

ブラウザで

`http://<デバイスの IP>:8080/`

にアクセス

(この手順だとポートを 8080 に変えていますが、デフォルトは 1441)

The screenshot shows the AWS IoT Greengrass console interface for the Local debug console. The left sidebar contains navigation options: Console, Device Details, and Documentation. The main content area displays the Local debug console components and their dependency graph.

**Components (5)**

Name	Version	Status	Origin
aws.greengrass.Cli	2.0.4	Running	User
aws.greengrass.LocalDebugConsole	2.0.3	Running	User
aws.greengrass.Nucleus	2.0.4	Finished	User
com.example.java.publisher	1.0.0	Running	User
main	-	Finished	User

**Dependency Graph**

```
graph TD; main[main: Finished] --> publisher[com.example.java.publisher: Running]; main --> localdebugconsole[aws.greengrass.LocalDebugConsole: Running]; main --> cli[aws.greengrass.Cli: Running]; main --> nucleus[aws.greengrass.Nucleus: Finished]; publisher -.-> cli; localdebugconsole -.-> cli; localdebugconsole -.-> nucleus;
```

# Greengrass V2 の Tips& トラブルシューティング

- プロビジョニング
- IoT Policy、IAM 権限
- ログの確認
- 設定情報の確認
- コンポーネントのデプロイ、OTA
- ローカルでバグコンソール
- Recipe ファイルの注意点

# Recipe ファイルの注意点 (1/3)

```
---  
RecipeFormatVersion: '2020-01-25'  
ComponentName: com.example.Pub  
ComponentVersion: '1.0.0'  
ComponentDescription: A component that publishes messages.  
ComponentPublisher: Amazon
```

ComponentDependencies:

aws.greengrass.Nucleus:

VersionRequirement: '>=2.0.4'

DependencyType: HARD

名称、バージョンなど  
基本的な情報

このコンポーネントの  
依存関係

先に起動してる必要があるコンポー  
ネントの様な依存関係があればここ  
に書く

# Recipe ファイルの注意点 (2/3)

ComponentConfiguration:

DefaultConfiguration:

Message: world

accessControl:

aws.greengrass.ipc.mqttproxy:

'com.example.Pub:publisher:1':

policyDescription: Allows access to publish to demo/topic topic.

operations:

- 'aws.greengrass#PublishToIoTCore'

resources:

- 'demo/topic'

コンポーネントの設定

Recipe 内で参照したい  
情報など

コンポーネント間通信  
、IoT Core に対しての  
通信の許可設定

この例だと IoT Core に対し  
て、`demo/topic` というトピックで  
の publish のみ許可



# Recipe ファイルの注意点 (3/3)

## Manifests:

### - Platform:

os: linux

### Lifecycle:

#### Install:

RequiresPrivilege: true

#### Script: |

```
python3 -m pip install awsiotsdk
```

```
Run: python3 {artifacts:path}/publisher.py '{configuration:/Message}'
```

Greengrass のユーザーでインストール出来ない場合は、 RequiresPrivilege:true で root 権限で実行

プラットフォームごとの設定

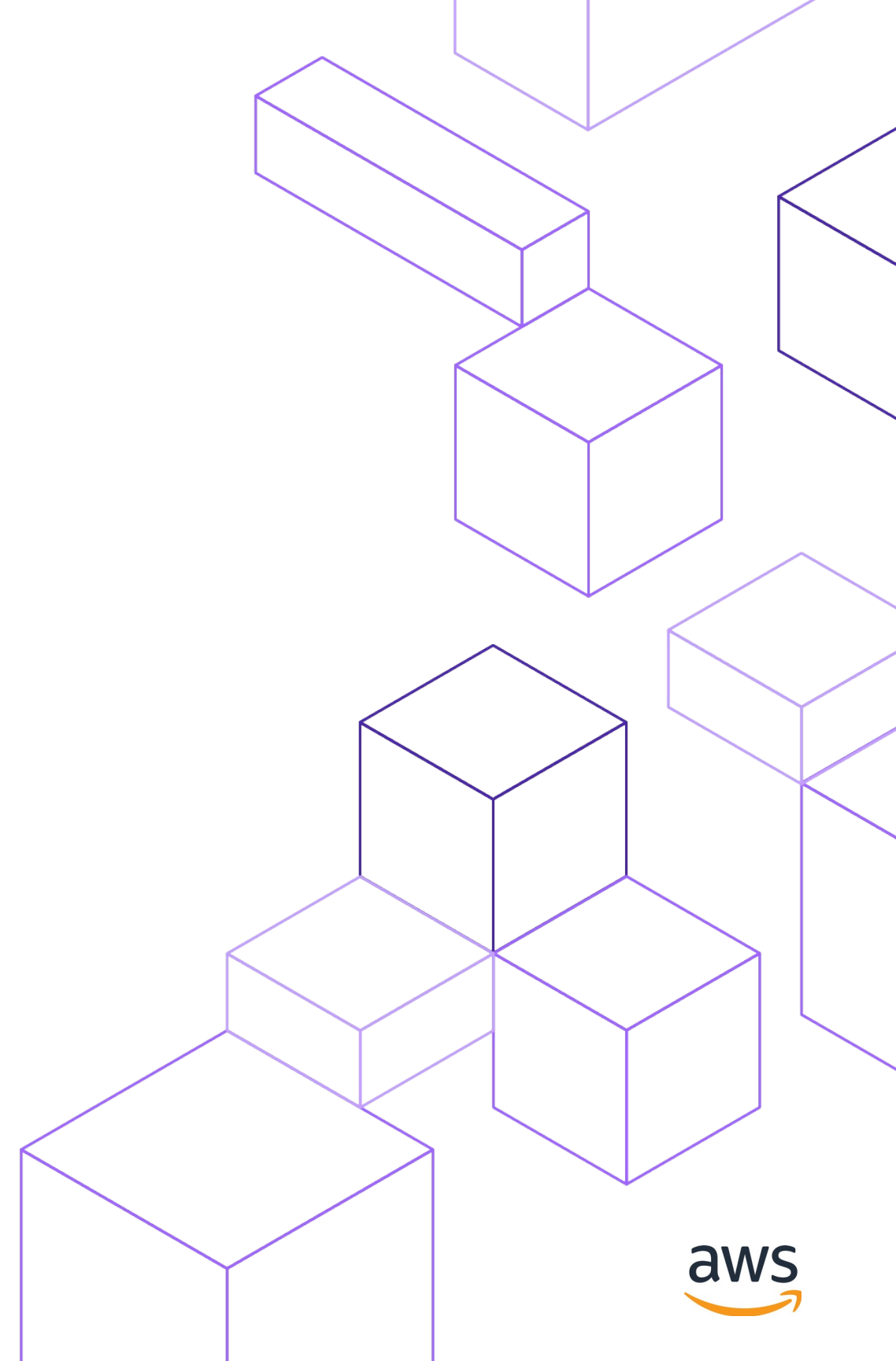
コンポーネントのライフサイクル毎の設定

DefaultConfiguration で設定した内容が参照可能

# component のライフサイクル

ステータス	状態	Recipe の Lifecycle
NEW	初めてデプロイされ、ストレージ上にある状態。バージョンが変わった場合も。	Bootstrap
INSTALLED	GG Core にインストール (Recipe の Install 完了) された。GG Core 起動時にも Install は実行される	Install
STARTING	別プロセスでバックグラウンドで実行するような物は、これを使う。例えば、MySQL の起動とか。正常終了を返すと RUNNING ステートに変わり、この component に依存している component は、実行を開始する。	Startup
RUNNING	スクリプトなどアプリケーションが実行中の間は終了しないような場合は、こちらを利用。アプリケーションが正常終了を返すと、FINISHED ステートに遷移する。	Run
FINISHED	component の実行が終了した。	
STOPPING	component を終了させる時に実行するスクリプトがあれば実行するとこのステートに遷移。例えば、MySQL の停止とか。	Shutdown
ERRORED	component を実行した際にエラーが発生したらこの状態になる。3 回 Recover を実行。	Recover
BROKEN	Recover を実行しても ERRORED 状態が変わらない場合にこの状態になる。 デプロイし直さないと復帰しない	

# Appendix



# AWS IoT Greengrass V2 を詳しく知りたい場合

- AWS IoT Greengrass v2 Developer Guide
  - <https://docs.aws.amazon.com/greengrass/v2/developerguide/what-is-iot-greengrass.html>
- AWS IoT Deep Dive #2 AWS re:Invent 2020 IoT Updates
  - <https://aws.amazon.com/jp/blogs/news/aws-iot-deep-dive-2/>
- JAWS-UG IoT 専門支部 「 re:Invent 2020 を味見する会 」
  - <https://www.slideshare.net/junichikawa1/cloud9aws-iot-greengrass-v2>

# Thank you!

