AWS Delivery Challenge

作成したロボットアプリケーションで実際のロボットを動かす

ここではこれまでクラウドシミュレータで動かしていたロボットアプリケーションを実際のロボットで動かすための手順を説明します。

この手順書は次のような方のために用意しました。

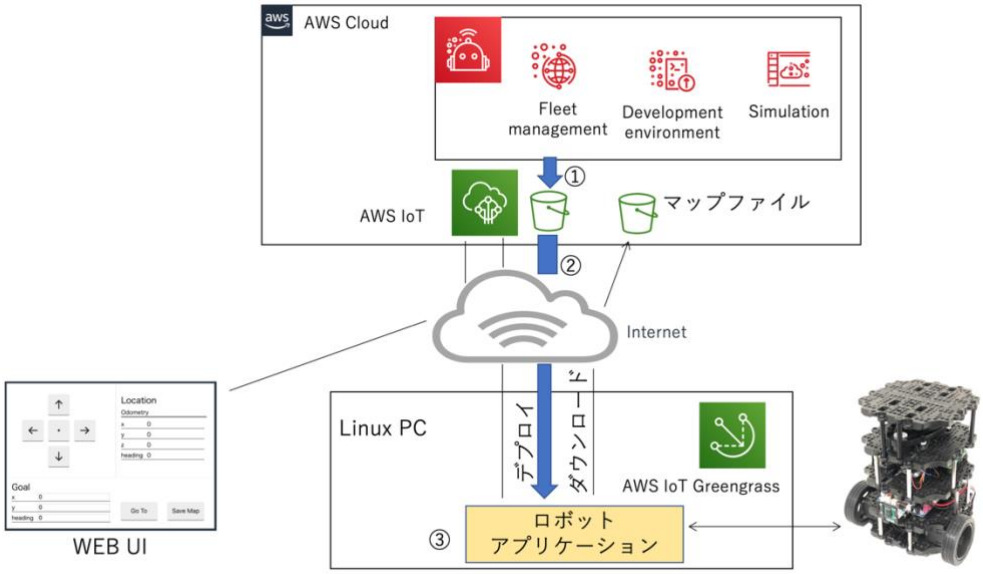
- シミュレーションで動かしたアプリケーションを実際のロボットでも動かしてみたい
- 本戦に向けて実際のロボットで動作を確認したい

※これはサンプルアプリケーションを実際のロボット上で実行したい方のための手順書です。
予選は全てシミュレーション環境で競います。予選リーグはこの作業を行わなくともトライすることができます。また本戦においてもロボットは事務局が行うのでここでの作業を行わなくても本戦に挑むことができます。

今回の作業を行うには次が必要です。

- AWS アカウント(現在お使いの AWS Educate スターターアカウント)
- Linux PC (Ubuntu 16.04 or Ubuntu 18.04)
- WiFi ネットワーク
- Turtlebot 3 Burger Delivery Challenge Edition (https://aws.amazon.com/jp/robot-delivery-challenge/bom-robot/)

今回の作業を行うことで次のような構成でロボットアプリケーションが動かすことができるようになります。



① ロボットアプリケーションをクラウドストレージ Amazon S3 に保存し、ロボットアプリケーションとして登録します。

② AWS RoboMaker コンソールからデプロイの指示をすることで、Amazon S3 に保存されたアプリケーションがデプロイターゲット(Linux PC) にダウンロードされ、展開されます。クラウドからの指示を受け取り、ファイルをダウンロードし、展開、実行する一連のプロセスはターゲットデバイスにインストールされた AWS IoT Greengrass が行います

③ ロボットアプリケーションがターゲット PC 内で起動します。
リモートコントロールを行うための Web UI との通信は AWS IoT を介して行われます。
ナビゲーションに必要なマップファイルは Amazon S3 に保存されているものをダウンロードして使用します。

作業は次の手順で進めていきます。

準備

1. ターゲットデバイス実行用の Launch ファイルの追加

2. 実機用の IAM ロールの作成

3. フリートの作成

4. RoboMaker へのデプロイターゲットの登録

5. デプロイターゲットのセットアップ

6. ターゲットロボット (Turtlebot3 Burger) のセットアップ

デプロイ

7. ロボットアプリケーションのビルド、バンドル、S3 へのアップロード

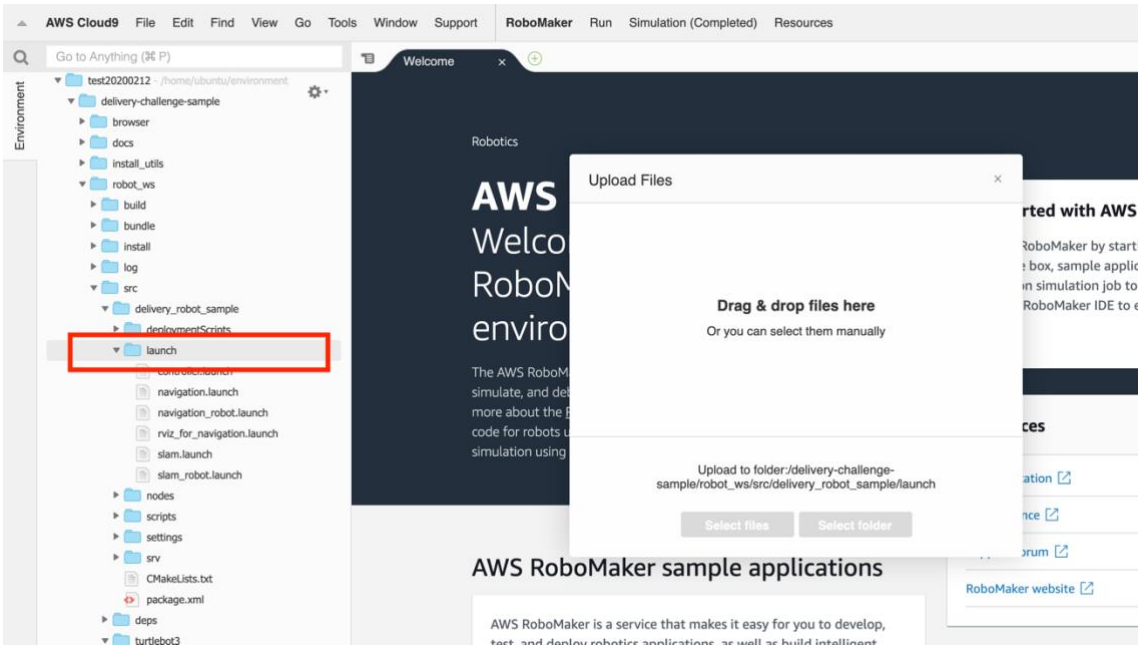
8. ロボットアプリケーションのデプロイ

1. ターゲットデバイス実行用の Launch ファイルの追加

ターゲットデバイスでのアプリケーションの起動パラメータと起動するプログラムを指定する Launch ファイルをサンプルアプリケーションの中に追加します。

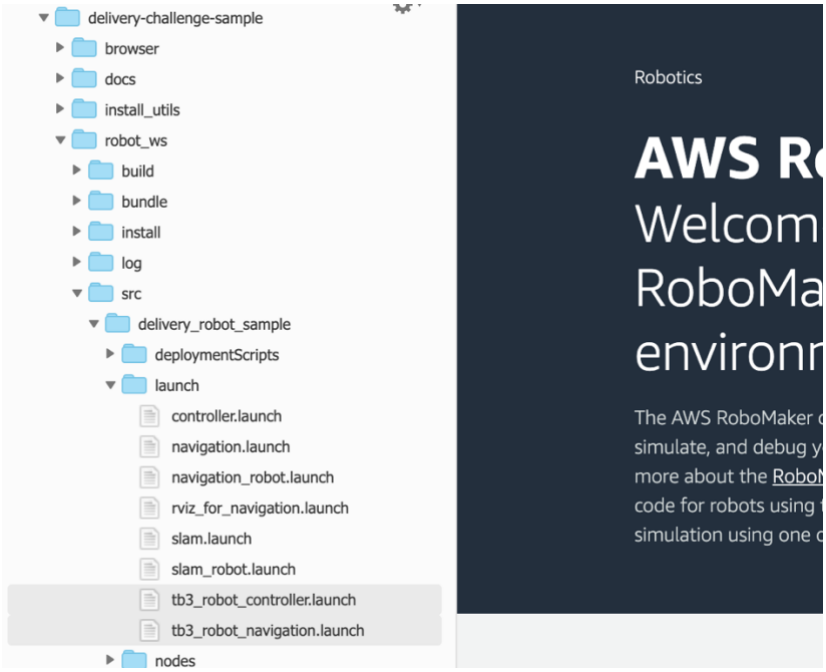
> **Launch ファイルをカスタマイズまたは新規に作成された場合：**
> ご自身で Launch ファイルをカスタマイズまたは新規に作成された場合、そのファイルをターゲットデバイスで動くように変更、新規作成してください。アプリケーションを Turtlebot3 Burger の実機で動くかすために配慮が必要なパラメータの一つは use_sim_time です、このパラメータ は true にならないようにしてください。

1-1. これまで作業してきた AWS RoboMaker の開発環境を開いてください。フォルダー delivery_challenge_sample -> robot_ws -> src -> delivery_robot_sample -> launch を選択し、メニュー File -> Upload Local Files... を選択し、「Upload Files」ウィンドウを開きます。



今回提供のファイル、tb3_robot_controller.launch と tb3_robot_navigation.launch を「Drag & drop files here」の領域にドラッグ&ドロップするか [Select files] ボタンをクリックしてそれぞれのファイルを選択するかして、それぞれのファイルを launch フォルダー配下に追加します。

1-2. それぞれのファイルが launch フォルダー内に追加されていることを確認します。

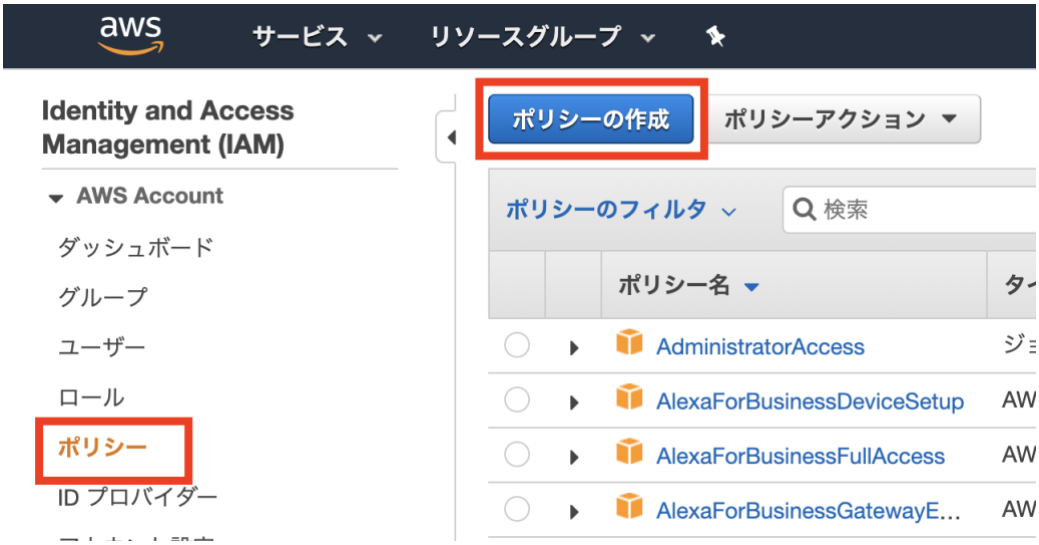


2. 実機用の IAM ロールの作成

ここではターゲットデバイスでロボットアプリケーションを実行するための IAM ポリシーと、IAM ロールを作成の手順を示します。 RoboMaker では ロボットを特定の IAM ロールと紐づけられることで、ロボットがどの AWS のサービスにアクセスできるかをコントロールしています。

AWS マネジメントコンソールで「Identity and Access Management (IAM)」を開きます。
(https://console.aws.amazon.com/iam)

左のナビゲーションペインから「**ポリシー**」を選び、リストの一番上の「**ポリシーの作成**」ボタンをクリックします。



ポリシーの作成画面ではタブを「**JSON**」に切り替え次を入力します。

<**bucket name**> は **ws_settings.yaml** の **bucket_name** の値で書き換えます。(同じ内容は HOWTO_deploy_worksheet.txt ファイルにも収録しています。HOWTO_deploy_worksheet.txt から内容をコピー&ペーストすると良いでしょう)

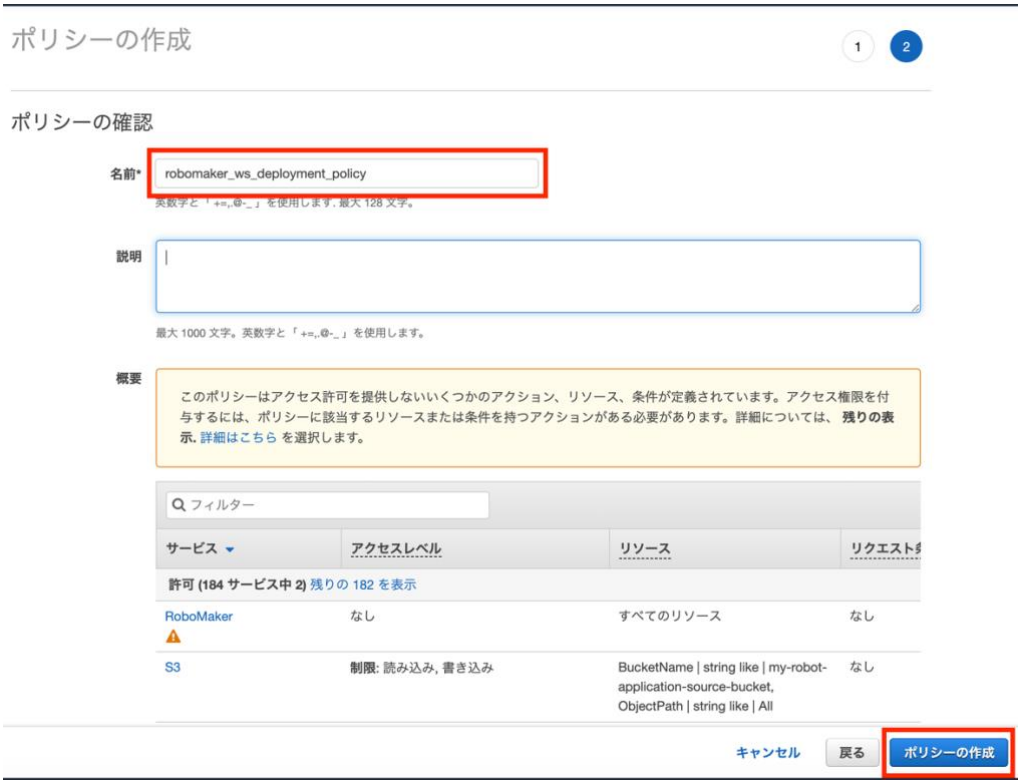
code 2-1

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

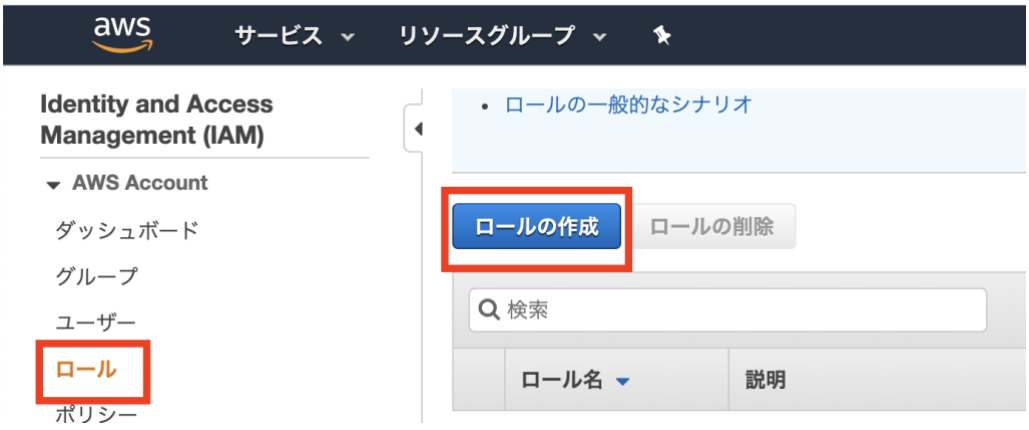
```
            "Action": [
                "robomaker:UpdateRobotDeployment"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:List*",
                "s3:Get*"
            ],
            "Resource": ["arn:aws:s3:::<bucket name>/*"]
        }
    ]
}
```

[**ポリシーの確認**] をクリック、「**ポリシーの作成**」画面では名前に任意の名前を設定し、[**ポリシーの作成**] ボタンを押してポリシーの作成を完了させます。

（一つの AWS アカウントを共有している場合は、それぞれで重複しない名前をつけます）



左のナビゲーションペインから 「**ロール**」を選び、リストの一番上の 「**ロールの作成**」ボタンをクリックします。



ステップ 1 の「**信頼されたエンティティの種類を選択**」では Greengrass を選択し、右下、[**次のステップ：アクセス権限**] ボタンをクリックします。

ステップ 2 の「**ポリシーの作成**」では、先ほど作成したポリシーを選択し、[**次のステップ：タグ**] をクリックします。

「**タグの追加**」はそのまま、「**次のステップ：確認**」で次に進みます。

「**確認**」画面では、名前に任意の名前を設定し、[**ロールの作成**]ボタンを押してロールを作成します。

（一つの AWS アカウントを共有している場合は、それぞれ重複しない名前をつけます）

作成した IAM ロールを再度開きます。開かれたら、その中の「**信頼関係**」タブを開き、タブの中にある [**信頼関係の編集**] ボタンをクリックします。（信頼関係 (Trust relationships) では、このロールを引き受けることができる Principal (特定のユーザーや AWS サービスなど) を指定しています。 ここに lambda.amazonaws.com を加えることで、Greengrass 内で AWS RoboMaker 連携するために動作されている Lambda に必要なアクセス権限を与えています。）

「**信頼関係の編集**」画面で次のようにポリシードキュメントを変更します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "lambda.amazonaws.com",
          "greengrass.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. フリートの作成

AWS RoboMaker のフリート管理はロボットを管理する機能です。フリート管理機能を用いて複数のロボットに遠隔からソフトウェアの更新を行うことができます。

ここではまずフリートを作成し、次にロボットを登録、フリートに追加します。フリートは複数のロボットを束ねるグループのようなもので、ロボットアプリケーションのデプロイなどの管理作業はこのフリートの単位で行うことになります。

AWS マネジメントコンソールで AWS RoboMaker を開きます。
(https://console.aws.amazon.com/robomaker)

左のナビゲーションペインから 「**フリート管理**」の「**フリート**」を選び、開かれた「**フリート**」の一覧の右上の [**フリートを作成する**] ボタンをクリックします。

フリートの作成画面が開かれます。名前に任意の名前を設定して [**作成**] ボタンをクリックします。

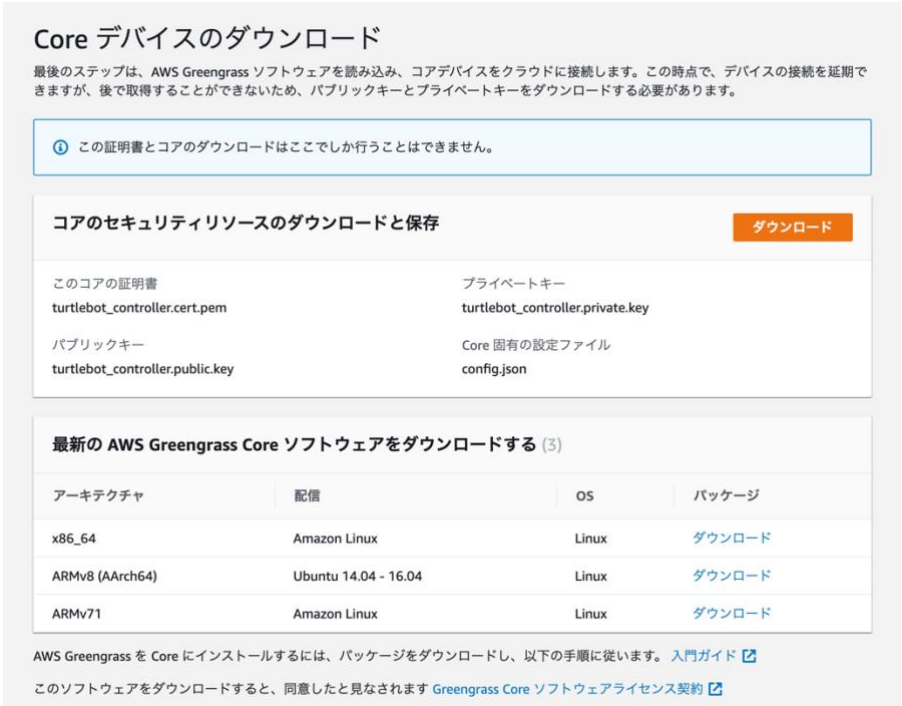
4. RoboMaker へのデプロイターゲットの登録

次に RoboMaker からロボットを管理できるようロボットを登録します。
左のナビゲーションペインから 「**フリート管理**」の「**ロボット**」を選び、開かれた「**ロボット**」の一覧の右上の [**ロボットの作成**] ボタンをクリックします。

ロボットの作成ウィンドウが開かれます。

- 「**名前**」はこれから登録するロボットにつける名前です。わかりやすい任意の名前を入力します。
- 「**アーキテクチャ**」はアプリケーションをインストールするデバイスの CPU を選択します。今回は **X86_64** になります。
- 「**Greengrass setup method**」 は 「手動セットアップ方法を使用する」を選択します。
- 「**AWS Greengrass グループ**」は「**新規作成**」を選びます。
- 「**AWS Greengrass プリフィクス**」には「名前」に入力したものが表示されているはずです。そのまま設定を受け入れます。
- 「**IAM ロール**」は「2. 実機用の IAM ロールの作成」で作成した IAM ロールを設定します。

- 右下［**作成**］ボタンをクリックします。作成がうまくいくと、「**Core デバイスのダウンロード**」ウィンドウが開かれます



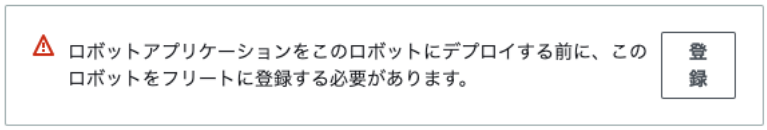
- 「**Core デバイスのダウンロード**」ではロボット本体に配置する各種ファイルをダウンロードします。

  特にここからダウンロードできる証明書とプライベートキーはここからしかダウンロードすることができず、またこの画面に再び戻ることはできませんので、[**ダウンロード**] ボタンをクリックして確実にこれらのファイルをダウンロードしておきます。

- AWS Greengrass ソフトウェアは今回のターゲットである **x86_64** 用のものをダウンロードしておきます。

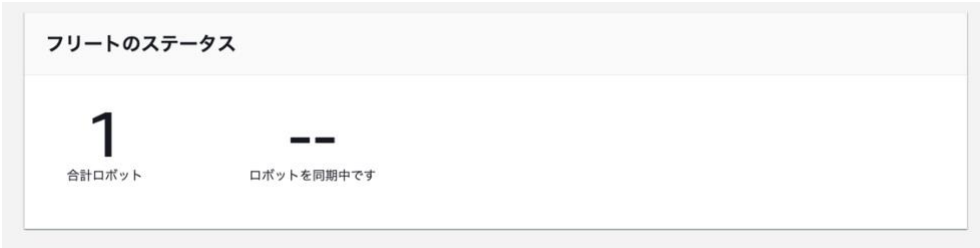
- ダウンロードが終わったら右下の [**ロボットの表示**] ボタンをクリックします。

- 「**詳細**」画面が開かれます。次のような表示が現れているかと思います。ここの[**登録**] ボタンをクリックして、フリートへの登録処理に進みましょう。または ナビゲーションペイン、**フリート管理**の**フリート**からフリートの一覧を開いても同じようにロボットの登録に進めることができます。



- 登録したフリートを確認します

- フリートのステータスで、合計ロボット数が 1 になっていれば正しくフリートにロボットが登録されている状態です。

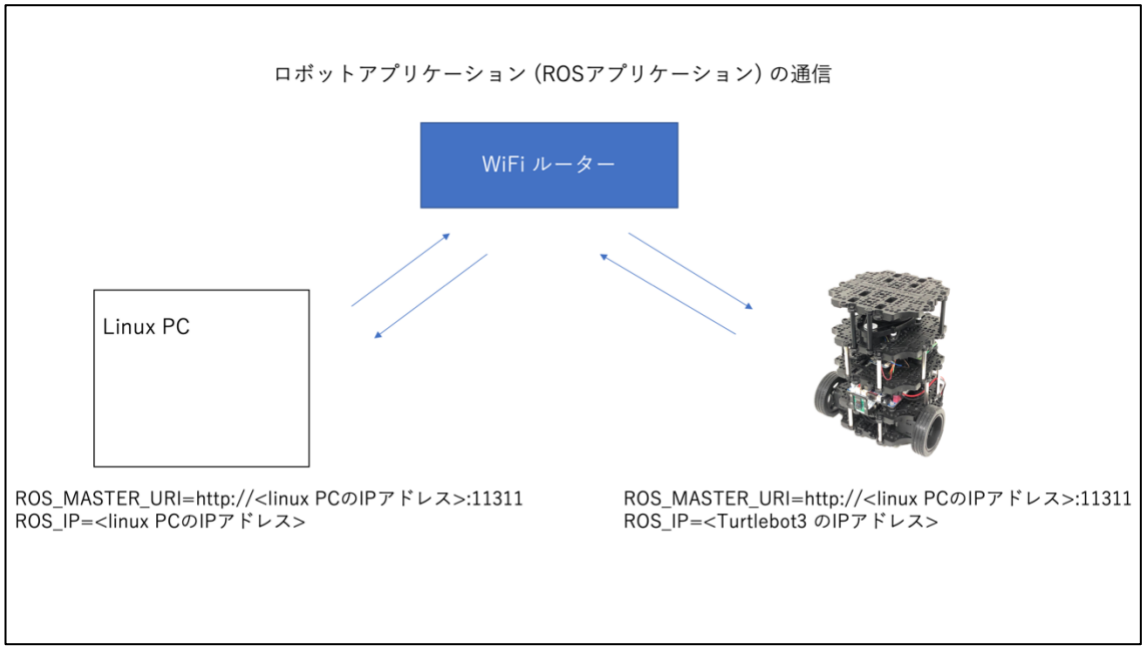


これでロボットにアプリケーションをデプロイするためのクラウド側の準備は整いました。次はロボットの設定に進みましょう。

5. デプロイターゲットのセットアップ

今回のロボットを動かす構成ではロボット本体にアプリケーションを配置するではなく、Linux PC に配置し、これがロボット本体をコントロールするようにします。このような構成を取る理由はいくつかありますが、最大の理由は処理能力です。より高性能な Linix PC 上でアプリケーションを動かす構成を取ることで、ナビゲーションなどの重たい処理も処理することが可能です。

ロボットと Linux PC は ROS (Robot Operating System) の通信方法にしたがって通信します。お互いが正しく通信できるようにするためには「お互いが見える」ように設定する必要があります。これは ROS_MASTER_URI と ROS_IP という環境変数で実現できます。次のように構成します。



（詳しくは http://wiki.ros.org/ja/ROS/EnvironmentVariables を参照してください）

RoboMaker と連携できるようロボットをコントロールする Linux PC を設定していきます。一連の設定が完了することで、ロボットは RoboMaker に登録されたロボットと連動するようになります。
（このステップは、次の内容をもとに記載しています
https://docs.aws.amazon.com/ja_jp/greengrass/latest/developerguide/setup-filter.other.html ）

Linux-PC で次を実行して ggc_user というユーザーと ggc_group というグループを作成します

code 5-1

```
sudo adduser --system ggc_user
```

```
sudo addgroup --system ggc_group
```

(お使いのシステムが addgroup コマンドを使用できない場合は、代わりに次を実行してください
sudo groupadd --system ggc_group)

次を実行して Greengrass を実行するために必要なコンポーネントが全て整っているか確認してください。

code 5-2

```
mkdir greengrass-dependency-checker-GGCv1.9.x
cd greengrass-dependency-checker-GGCv1.9.x
wget https://github.com/aws-samples/aws-greengrass-samples/raw/master/greengrass-dependency-checker-GGCv1.9.x.zip
unzip greengrass-dependency-checker-GGCv1.9.x.zip
cd greengrass-dependency-checker-GGCv1.9.x
sudo ./check_ggc_dependencies | more
```

出力にエラーが表示されなければ AWS IoT Greengrass はデバイスで正常に実行することができます。エラーが表示された場合、エラーの内容に従ってソフトウェアの追加インストールなどを行います。

AWS IoT Greengrass のインストールファイルを Linux-PC にコピーして、ターミナルを開きファイルをコピーし次のコマンドでファイルを展開します。

code 5-3

```
wget https://d1onfpft10uf5o.cloudfront.net/greengrass-core/downloads/1.9.4/greengrass-linux-x86-64-1.9.4.tar.gz
sudo tar -xzvf greengrass-linux-x86-64-1.9.4.tar.gz -C /
```

次のコマンドを Linux-PC で実行しルート CA 証明書をダウンロードします。

code 5-4

```
cd /greengrass/certs/
sudo wget -O root.ca.pem https://www.amazontrust.com/repository/AmazonRootCA1.pem
```

「**4. RoboMaker へのロボットの登録**」でダウンロードしてきたコアのセキュリティーリソースファイル（ロボット名-setup.zip）を Linux-PC にアップロードします。

Linux-PC で次のコマンドを実行してアップロードしたファイルを展開します。途中ファイルを上書きして良いか確認された場合、全て上書きするようにしてください

code 5-5

```
sudo unzip <robot name>-setup.zip -d /greengrass
sudo rm -rf /home/ggc_user/roboMakerDeploymentPackage
```

(<robot name>-setup.zip は実際のファイルの名前で読み替えてください)

次のコマンドを Linux-PC で実行します。「Greengrass successfully started」メッセージが表示されたらセットアップは成功です。

code 5-6

```
sudo /greengrass/ggc/core/greengrassd start
```

これでデプロイターゲットの準備は完了です。続いてロボットの設定をしていきます。

6. ターゲットロボット (Turtlebot3 Burger) のセットアップ

今回ロボットは Turtulebot3 Burger をベースにカメラとテーブルを追加しています。ハードウェア構成、追加部品などの詳細は次を参照してください。

https://aws.amazon.com/jp/robot-delivery-challenge/bom-robot/

Turtlebot3 Burger のセットアップは ROBOTIS 社の e-Manual を参考にしてください。なお、このドキュメントで **PC Setup となっている箇所が今回は AWS RoboMaker からアプリケーションがデプロイされてくる箇所になりますので、PC Setup の箇所は作業不要です。また作業の中で ROS_MASTER_URI と ROS_HOSTNAME に設定する IP アドレスは 5. でセットアップした Linux PC の IP アドレスを設定します。**
http://emanual.robotis.com/docs/en/platform/turtlebot3/setup/#setup

7. ロボットアプリケーションのビルド、バンドル、S3 へのアップロード

ロボットアプリケーションのビルドとバンドルをします。

メニューから *Run -> Workflow -> Delivery Robot app build -> bundle* を選び、ロボットアプリケーションのビルドとバンドルを行います。これにより、「1.ターゲットデバイス実行用の Launch ファイルの追加」で追加したファイルをロボットアプリケーションに反映しています。

デスクトップ Linux 環境で作業をしている場合：
robot_ws ディレクトリで次を実行します

```
rosdep install --from-paths src --ignore-src -r -y

colcon build

colcon bundle
```

最新のバンドルを S3 にアップロードします。
AWS RoboMaker 開発環境のターミナルで次を実行します。

code 7-1

```
export robot_app_name="<robot_app_name>"
export bucket="<bucket_name>"
export ros_version="<ros_version>"

aws s3 cp ~/environment/delivery-challenge-sample/robot_ws/bundle/output.tar ¥
    s3://${bucket}/delivery-challenge-sample/robot_ws/bundleoutput.tar
app_name_arn=`aws robomaker list-robot-applications | grep -o ¥
    '¥"arn:.*'${robot_app_name}'.*¥"'`
aws robomaker update-robot-application --application ${app_name_arn:1:-1} ¥
      --sources s3Bucket=${bucket},s3Key=delivery-challenge-
sample/robot_ws/bundle/output.tar,architecture=X86_64 ¥
      --robot-software-suite name=ROS,version=${ros_version}
```

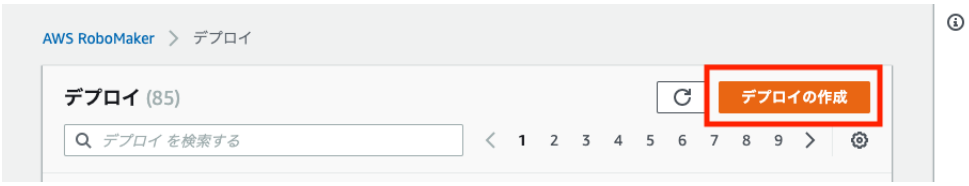
<robot_app_name> <bucket_name> <ros_version> それぞれ **ws_resources.yaml** の **robot_app_name、bucket_name, ws_info.yaml** の ros_version の値に書き換えてください

これで最新のアプリケーションコードがデプロイ可能な状態になりました。デプロイを開始しましょう。

8. ロボットアプリケーションのデプロイ

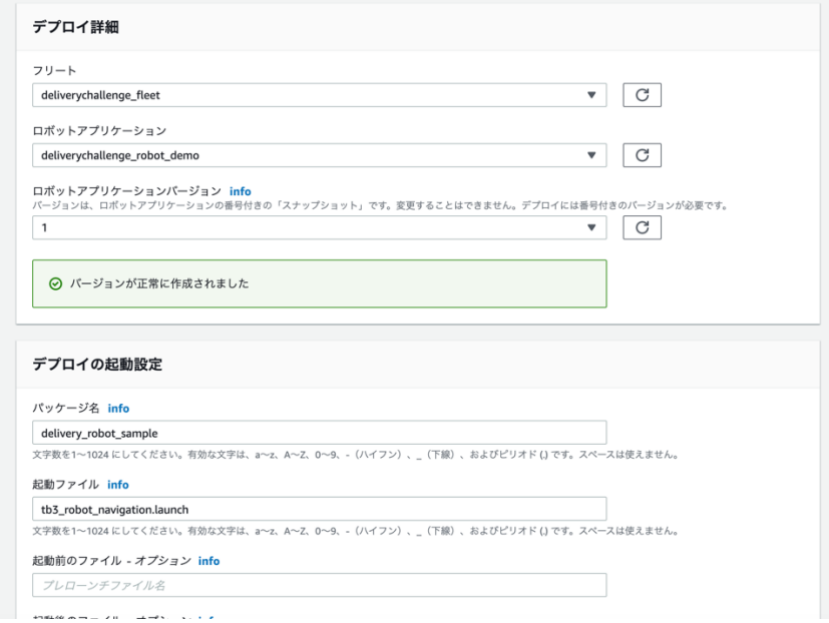
この構成を実現するためにまず、Linux PC の IP アドレスを調べておきます。

AWS マネジメントコンソール、AWS RoboMaker で **フリート管理** の、**デプロイ** を開き、デプロイ画面右上の [**デプロイの作成**] ボタンをクリックします。

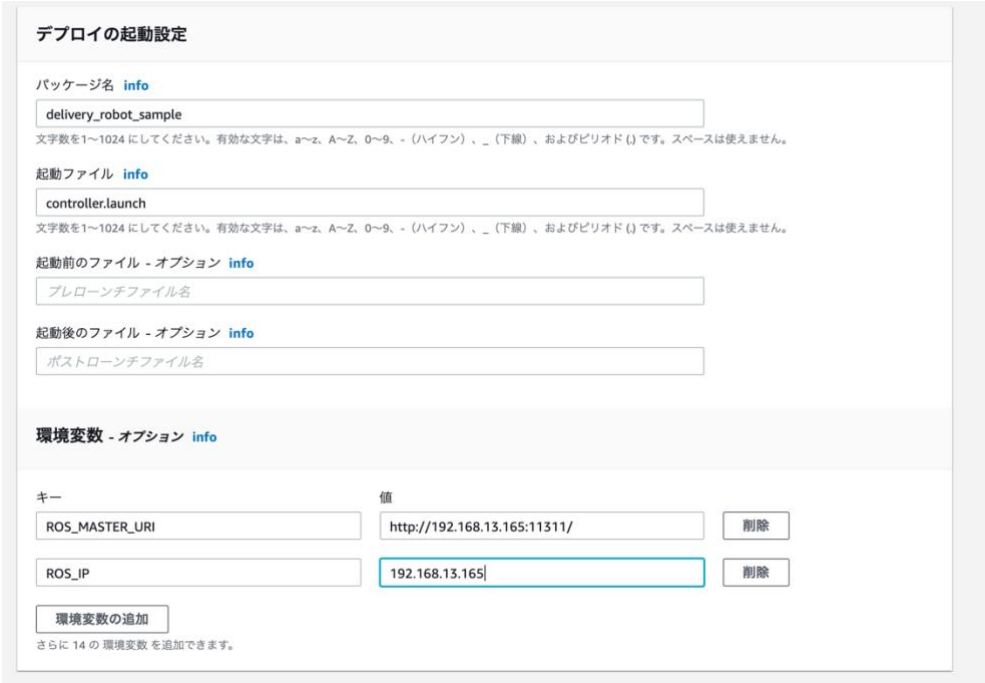


「**デプロイの作成**」画面が開かれます。それぞれ項目を入力します。

- **フリート** は、今回作成したフリートを選びます
- **ロボットアプリケーション** は、今回作成したロボットアプリケーション（**ws_settings.yaml** の **robot_app_name** の値)を選びます
- **ロボットアプリケーションバージョン** は「**新規作成**」 を選びます。新しいバージョンの作成を確認するポップアップウィンドウが現れますので、[**作成**] ボタンをクリックしてこれを受け入れます。
- デプロイ起動設定セクションもそれぞれ入力します。
- **パッケージ名**は **delivery_robot_sample** と入力します。
- **起動ファイル**は **tb3_robot_navigation.launch** と入力します。(これはナビゲーションプログラムを起動する場合。マップを作りたい場合や、リモートコントロールだけ行いたい場合は起動ファイルは **tb3_robot_controller.launch** と入力)



- **環境オプション**で、ROS_MASTER_URI と ROS_IP を次のように設定します。192.168.13.165 の箇所はターゲットハードウェア(LinuxPC) の IP アドレスで読み替えてください



右下 [作成] ボタンをクリックします。デプロイが開始されます。進行状況が表示されます。

デプロイが成功するとロボットのステータスが 「**同期中**」に変わります。

ロボットの電源を入れます。
ロボット本体では次のコマンドを実行します。

code 8-1

```
export ROS_HOSTNAME=<ロボットの IP アドレス>
export ROS_MASTER_URI=http://<PC の IP アドレス>:11311/
roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

Web UI を開いてロボットがコントロールできるか確認してください。

tb3_robot_navigation.launch を指定してアプリケーションをデプロイした場合は次のコマンドを Linux PC で実行することでナビゲーションの状況をビジュアライズすることができます。

code 8-2

```
for dp in `find /home/ggc_user/roboMakerDeploymentPackage -maxdepth 2 -name setup.sh`;
do
BUNDLE_CURRENT_PREFIX=${dp:0:-9}
source ${dp} --extend
done
export TURTLEBOT3_MODEL=burger
roslaunch delivery_robot_sample rviz_for_navigation.launch
```

tb3_robot_controller.launch を指定してアプリケーションをデプロイした場合は次のコマンドを Linux PC で実行することで地図の作成を開始することができます。(地図の保存を行うと既存の地図を上書きすることになるので気をつけてください。シミュレーション環境で作った地図がある場合、AWS コンソール、Amazon S3 にアクセスしてあらかじめ地図のファイルをダウンロード、退避しておくと良いでしょう)

```
for dp in `find /home/ggc_user/roboMakerDeploymentPackage -maxdepth 2 -name setup.sh`;
do
BUNDLE_CURRENT_PREFIX=${dp:0:-9}
source ${dp} --extend
done
export TURTLEBOT3_MODEL=burger
roslaunch delivery_robot_sample slam.launch
```

以上で全ての作業が終了しました。
プログラムを変更した場合は、「7.ロボットアプリケーションのビルド、バンドル、S3 へのアップロード」から作業を繰り返すことで最新のプログラムを実際のロボットで試すことができます。