ROS

Doosan Robot

M0609 | M0617 | M1013 | M1509

ROS Programming Manual

DOOSAN

1. Doosan Robotics ROS in AWS

- 본 매뉴얼은 Doosan Robotics ROS 패키지를 AWS RoboMaker 와 연동하여 사용하는 방법에 대하여 설명 합니다.

- 본 문서에서는 Doosan Robotics ROS 패키지를 AWS RoboMaker 환경에서 로봇 시뮬레이션, 로봇 App 생성, 배포에 대한 기본적인 내용만을 다룹니다. 이 주제를 벗어나는 사용법이나 어플리케이션은 AWS Robomaker 매뉴얼이나 세부 문서를 참고하시기 바랍니다.

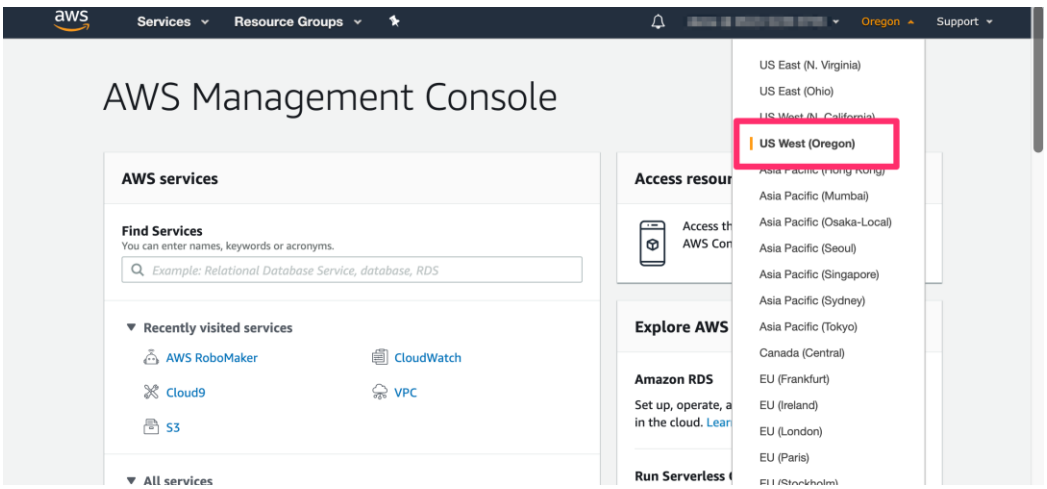
  https://aws.amazon.com/robomaker

- AWS 클라우드 환경이 아닌 자신의 로컬 환경에서 Doosan Robotics ROS 패키지를 사용해 보시면, 좀 더 명확하고 빠르게 본 내용을 이해할 수 있습니다.

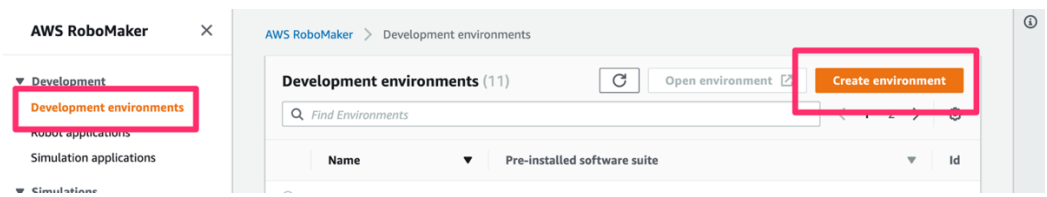
  https://github.com/doosan-robotics/doosan-robot

2. Setup development environment

- **[2.1]** AWS Management Console (**https://console.aws.amazon.com**) 열고, region을 US West Oregon 으로 선택합니다.

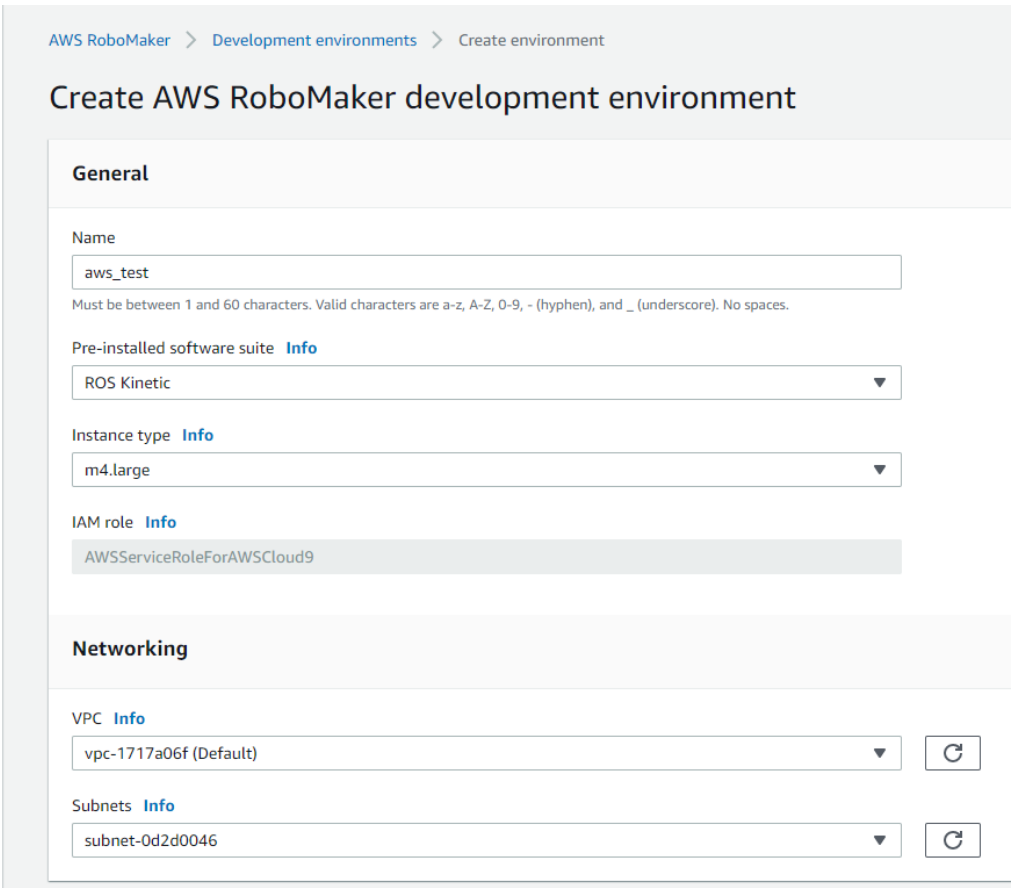


- **[2.2]** 왼쪽 탐색 메뉴에서 ***Development ->Development environments*** 를 선택한 후, "Development environment" 페이지의 오른쪽 위 **[Create environment]** 버튼을 클릭합니다.

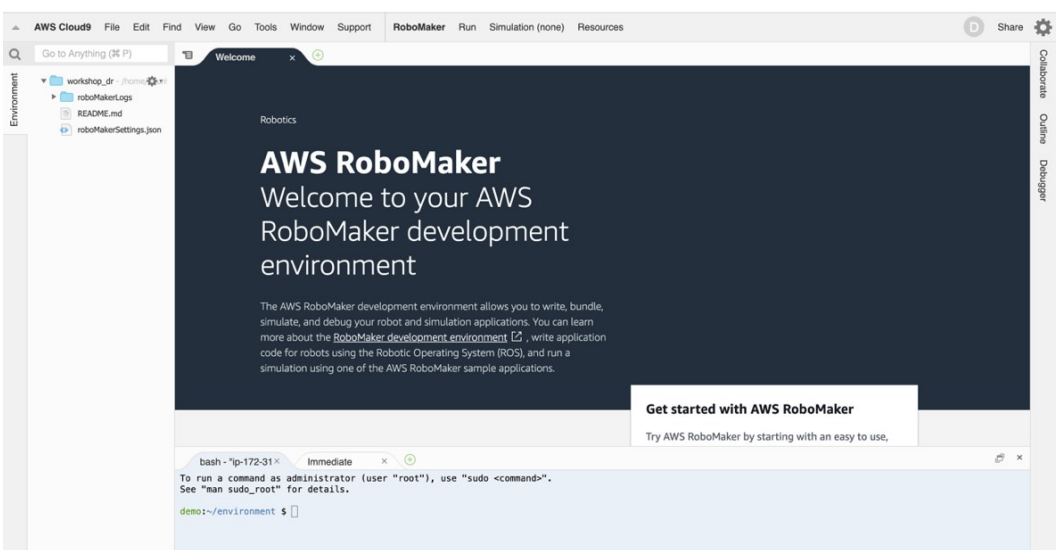


- **[2.3]** 자신의 환경 "Name"을 설정합니다. Pre-installed software suite 에서 ROS Kinetic 을 선택합니다. Instance type은 m4.large 로 선택합니다. 네트워킹 설정의 경우 목록에서 default VPC를 선택하고 VPC를 선택한 후 목록에서 서브 넷 하나를 선택하십시오. 오른쪽 하단 [Create] 버튼을 누르면 개발 환경 구성이 시작됩니다. (2~3분 정도 소요 됨)

  - 환경 생성 실패 시, 생성 환경을 지우고 다른 서브넷을 선택하여 진행합니다.

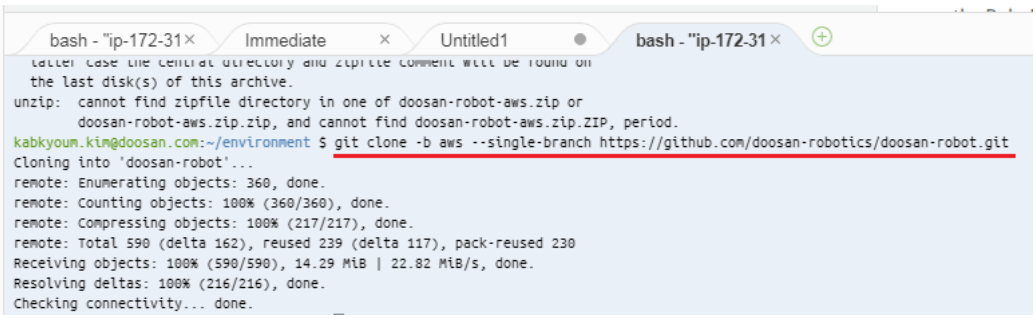


- **[2.4]** Cloud9 (AWS의 Cloud IDE 서비스)를 기반으로 하는 RoboMaker 개발 환경이 시작됩니다.



- **[2.6] Doosan ROS package** 소스를 생성된 클라우드 환경으로 업로드 하기 위해서 터미널 창에 하기의 명령을 입력합니다.

git clone -b aws --single-branch https://github.com/doosan-robotics/doosan-robot.git



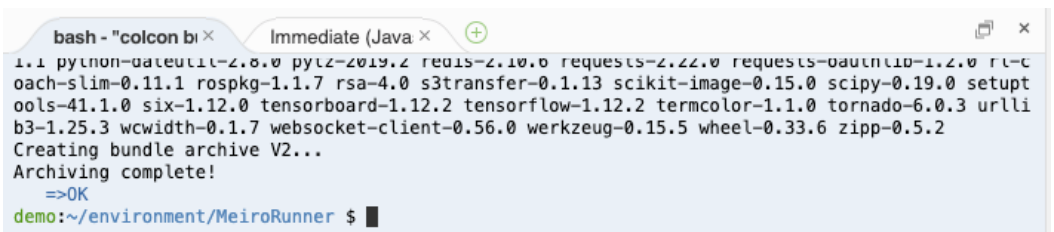
- **[2.7]** 설정 스크립트를 실행하기 위해서 다음 명령을 입력하십시오

cd doosan-robot

chmod +x setup.sh

./setup.sh

- **[2.8]** 설정 스크립트가 시작됩니다. 완료하는데 약 30 분 정도 걸립니다. 완료되면 터미널은 다음과 같이 표시됩니다.



방금 실행 한 스크립트의 기능은 다음과 같습니다.

- 개발 환경 업데이트.
- Prepare the resources

.Bundle 및 시뮬레이션 작업의 출력을 저장하는 S3 버킷 생성.

(S3 bucket name : robomaker-ws-us-west-2-[AWS Account Number]-[YYMMDD] –[HHMMSS]
e.g.: robomaker-ws-us-west-2-123456789012-190808-135139)

. 시뮬레이션 실행을 위한 IAM role 설정.

. 로봇 코드를 배포하기 위한 IAM role 설정.

. 로봇 및 시뮬레이션 어플리케이션의 이름 생성

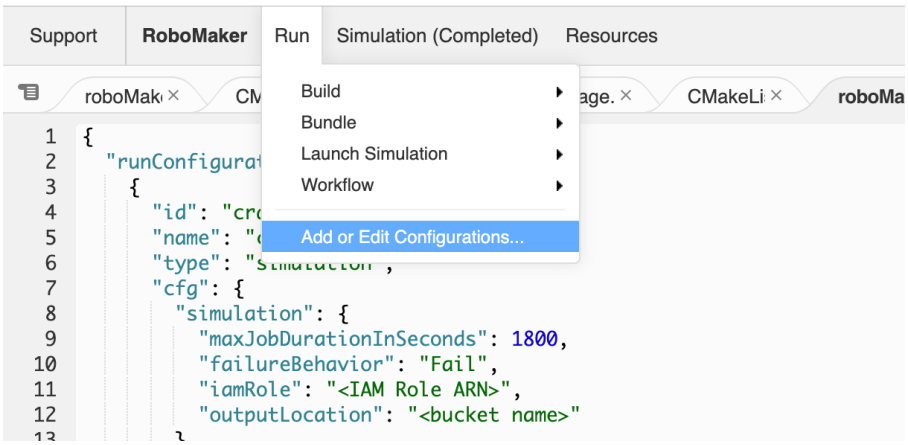
. 프로젝트 설정 파일 roboMakerSettings.json을 업데이트

. 초기 빌드 및 Bundle 소스 코드를 실행.

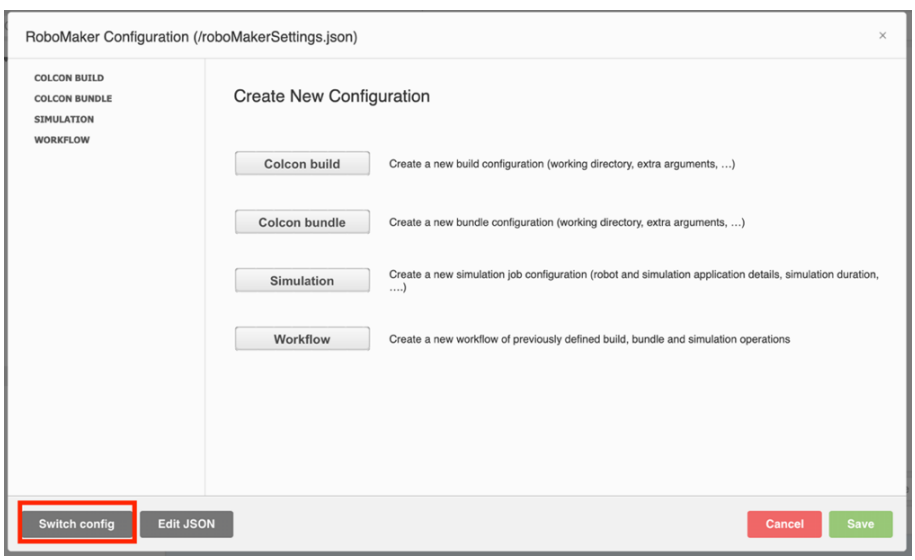
3. Running the application on simulator

- **[3.1]** 개발 환경 구성 파일을 개발 환경에 로드 합니다.

  - 메뉴에서, ***Run -> Add or Edit Configurations...*** 선택하십시오.

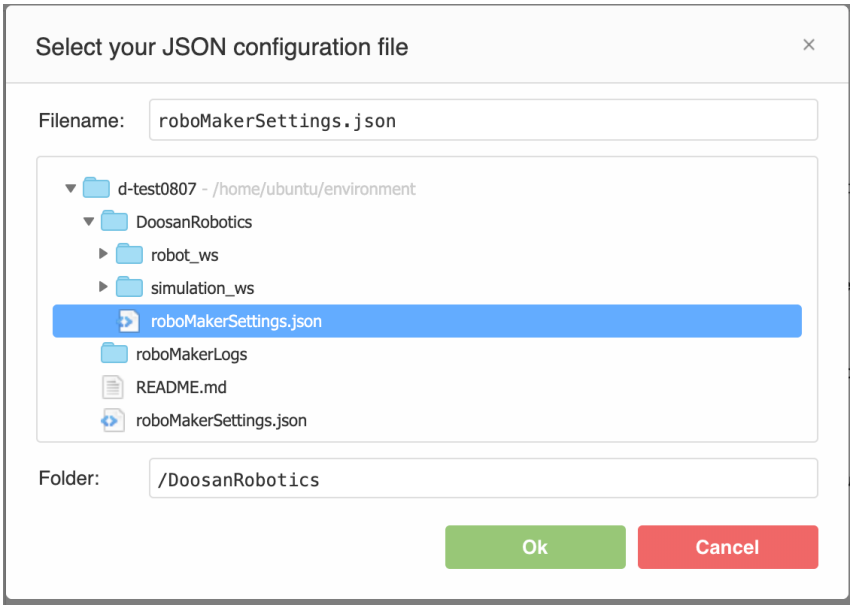


- **"RoboMaker Configuration"** 창 이 열리게 되고 왼쪽 하단 **[Switch Config]** 버튼을 클릭하십시오.

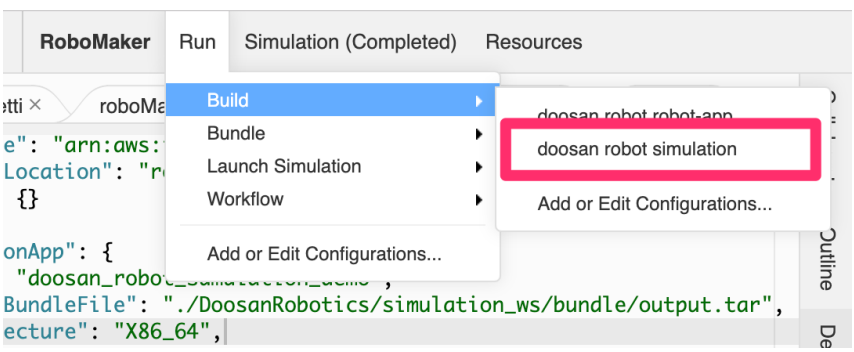


- **doosan-robot 디렉터리**를 열고 **roboMakerSettings.json** 파일을 선택하십시오. **[OK]** 버튼을 클릭 한 후, **"RoboMaker Configuration"** 창의 오른쪽 하단에 있는

[Save] 버튼을 클릭하십시오.

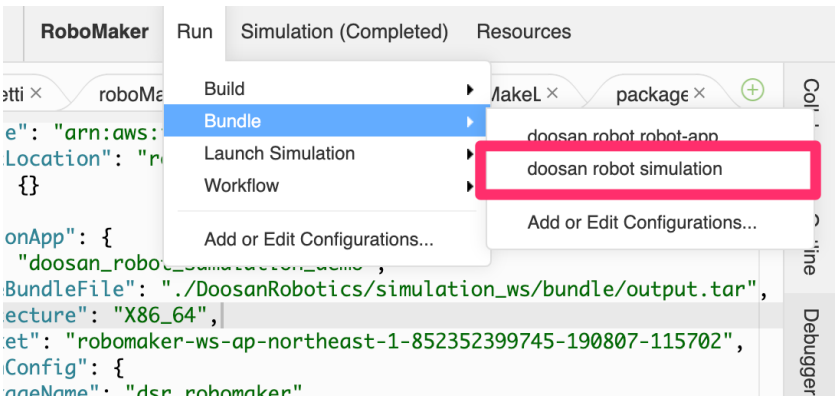


- **[3.2]** 이제 이 프로젝트에 대한 메뉴 항목이 생기게 됩니다. 메뉴에서 ***Run -> Build -> doosan robot simulation*** 을 선택하십시오. simulation application 을 빌드하게 됩니다.

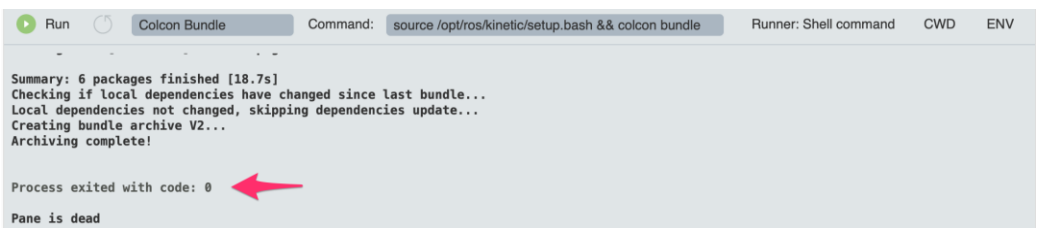


Running for building Doosan robot simulation 의 결과는 하기와 같습니다;

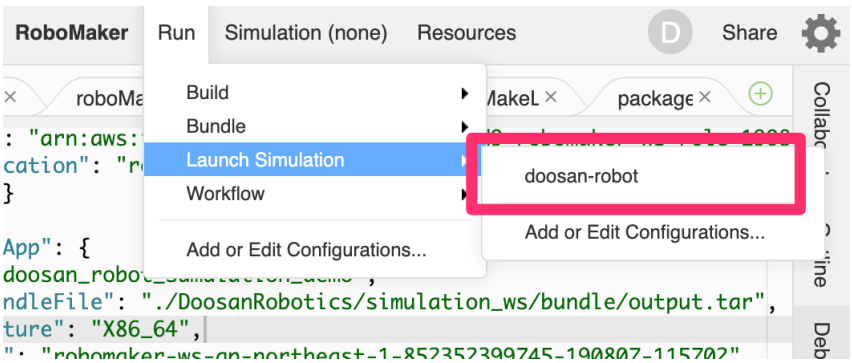
- **[3.3]** 메뉴에서 ***Run -> Bundle -> doosan robot simulation.*** 을 선택하십시오. 그러면 애플리케이션 용 **bundle** 이 생성됩니다. **Bundle** 은 ROS 애플리케이션을 압축하는 프로세스입니다. RoboMaker는 **Bundle** 파일을 사용하여 시뮬레이션 환경 / 로봇 하드웨어에서 응용 프로그램을 시작합니다. **Bundle** 파일은 환경으로 다운로드 되고 시뮬레이션 환경 / 로봇 하드웨어에서 실행하기 위해 압축이 풀립니다.



Doosan robot simulation의 번들링 결과는 다음과 같습니다.



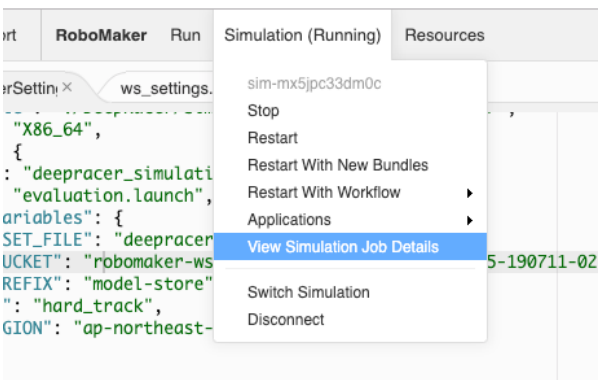
- **[3.4]** 메뉴에서 ***Run -> Launch Simulation -> doosan-robot*** 을 선택하십시오. 시뮬레이션 응용 프로그램이 시작됩니다.



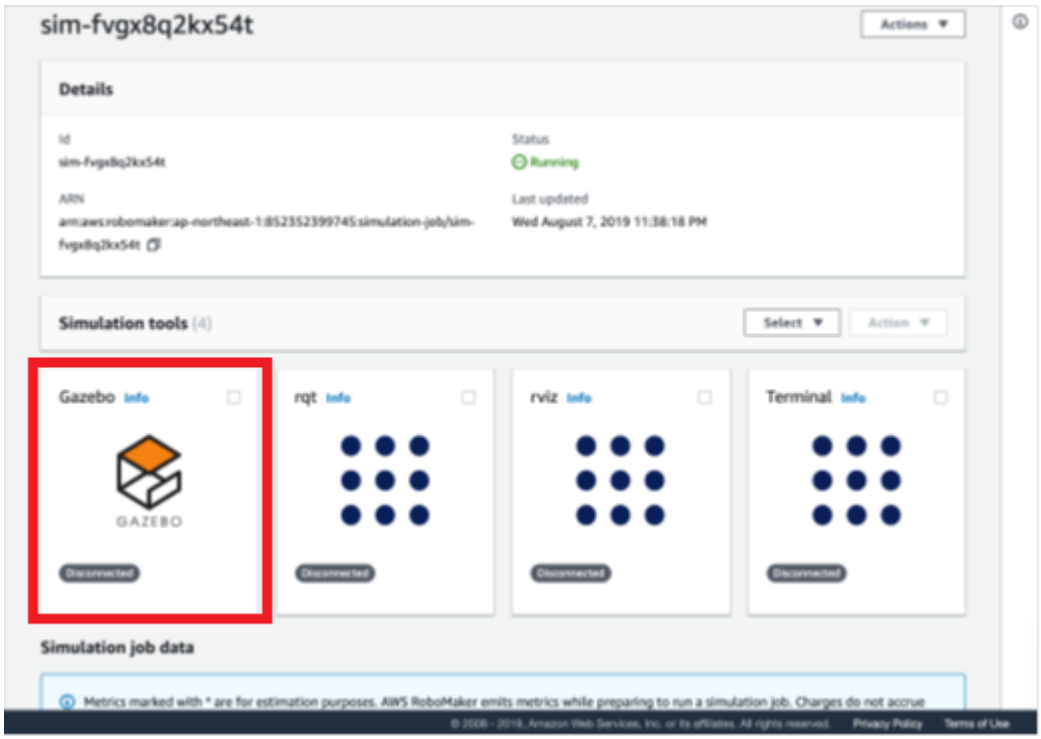
이 작업은 번들 파일을 **S3**(클라우드 파일 스토리지)에 복사하고 시뮬레이션 작업을 시작합니다. 그런 다음 시뮬레이션 작업은 S3에서 파일을 로드하고 압축을 풀고 응용 프로그램을 시작합니다.

두산 로봇 시뮬레이션 작업 결과는 다음과 같습니다.

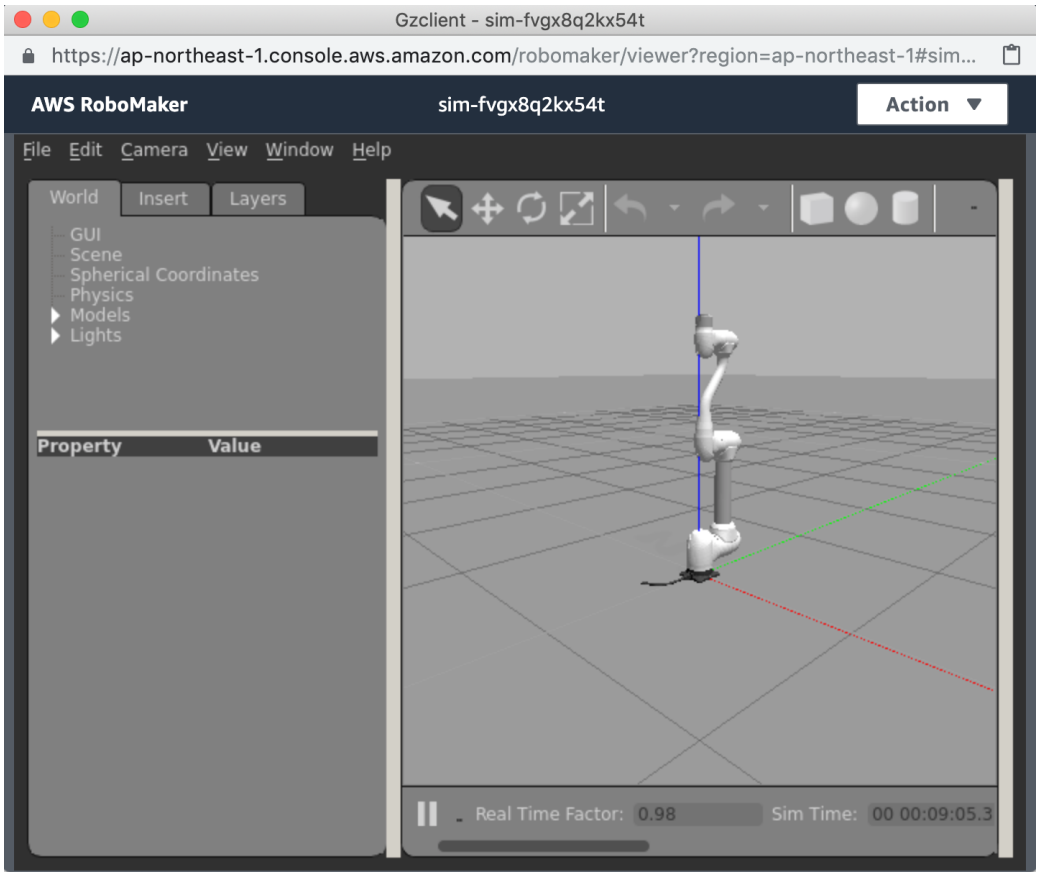
- **[3.5]** 시뮬레이션 작업이 시작되었습니다. 메뉴에서 ***Simulation -> View Simulation Job Details***. 을 선택하십시오. 시뮬레이션 작업 세부 사항 페이지가 열립니다.



- **[3.6]** 약 3분 정도 기다리면 **status** 가 **'preparing'** 에서 **'Running'** 으로 바뀌게 되고 하기의 상태가 됩니다.



- **Gazebo**는 ROS 애플리케이션을 시뮬레이션 하는 시뮬레이터입니다. **Gazebo** 아이콘을 클릭하면 Gazebo 시뮬레이션 창과 상호 작용할 수 있습니다.



- 로봇 모델이 로드 되고 시뮬레이션이 시작됩니다.

Note:

simulation_ws/dsr_launcher/launch/**aws_bringup_simulation.launch**

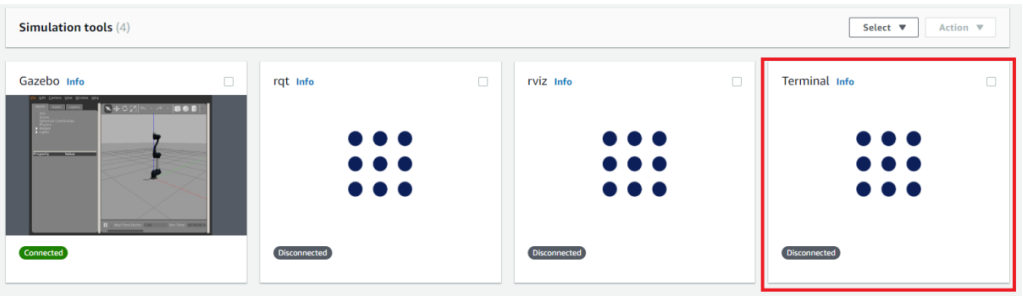
이 파일은 시뮬레이션 응용 프로그램을 시작하는 데 사용됩니다. 다음과 같이 **roboMakerSettings.json** 파일에 지정됩니다.

```
…
"simulationApp": {
  "name": "doosan_robot_sumulation_annakie",
  "sourceBundleFile": "./doosan-robot/simulation_ws/bundle/output.tar",
  "architecture": "X86_64",
  "s3Bucket": "robomaker-ws-us-west-2-000000000000-000000-000000",
  "launchConfig": {
  "packageName": "dsr_launcher ",
```

```
    "launchFile": "aws_bringup_simulation.launch",
    "environmentVariables": {}
    },
    "simulationSoftwareSuite": {
    "name": "Gazebo",
    "version": "7"
    },
    "renderingEngine": {
    "name": "OGRE",
    "version": "1.x"
    },
    "robotSoftwareSuite": {
    "name": "ROS",
    "version": "Kinetic"
    }
}
…
```

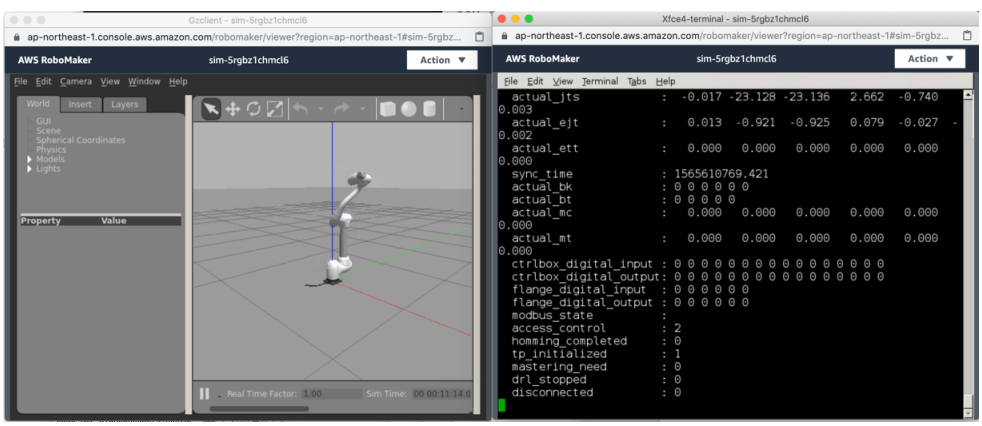
- 모든 준비가 되었고, ROS 응용 프로그램을 통해 두산 로봇을 제어할 수 있습니다.

[3.7] 예제 프로그램들을 구동해 보겠습니다. 터미널 아이콘을 클릭하여 터미널 창을 시작합니다.



- 다음 명령을 입력하십시오.

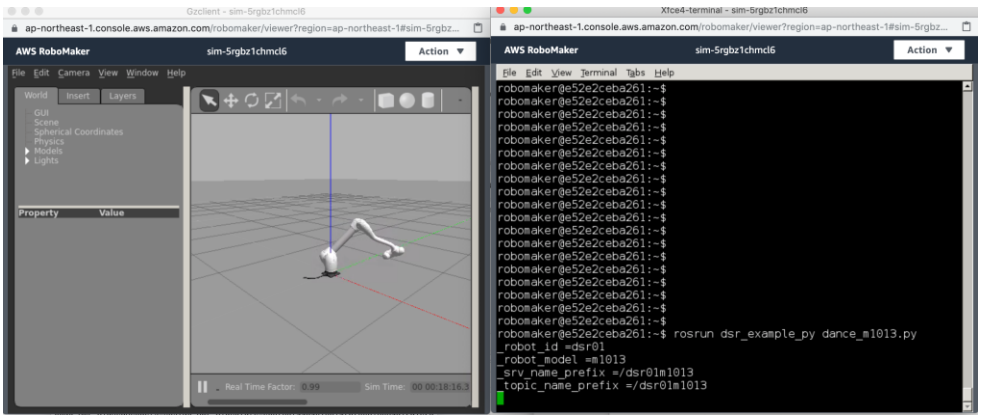
```
rosrun dsr_example_py single_robot_simple.py
```



- 터미널 창에서 Ctl+c 눌러 single_robot_simle.py 실행을 중지 시키고,

다음 명령을 입력하십시오.

```
rosrun dsr_example_py dance_m1013.py
```



- 시뮬레이션 구성을 변경하여 다중 로봇을 제어할 수 있습니다.

다음과 같이 **aws_bringup_simulation.launch** 를 변경 한 후,

(simulation_ws/src/dsr_launcher/launch/aws_bringup_simulation.launch)

```
<?xml version="1.0"?>
<launch>
  <!-- node name="drcf" pkg="common" type="run_drcf.sh" output="screen" required="true"/-->
  </include>
  <!-- Start Gazebo with an empty world. -->
  <!--include file="$(find dsr_launcher)/launch/single_robot_gazebo.launch"-->
  <include file="$(find dsr_launcher)/launch/multi_robot_gazebo.launch">
    <arg name="mode"      value="virtual"/>
    <arg name="model"     value="m1013"/>
    <arg name="color"     value="white"/>
    <arg name="gripper" value="none"/>
    <arg name="mobile"    value="none"/>
  </include>
</launch>
```

- 시뮬레이션을 다시 빌드하고 번들링 하십시오.

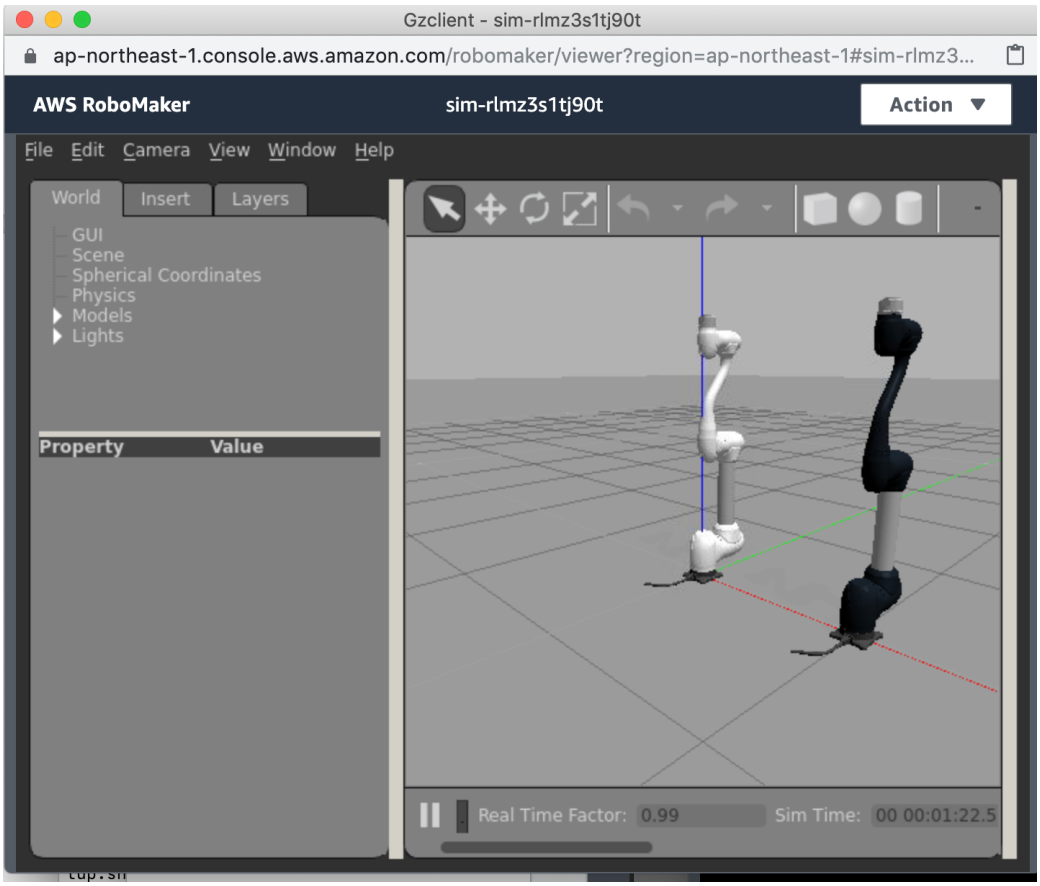
  Run -> Build -> doosan robot simulation.

  Run -> Bundle -> doosan robot simulation.

- 시뮬레이션을 시작하면 이제 Gazebo에 두 개의 로봇이 표시됩니다.

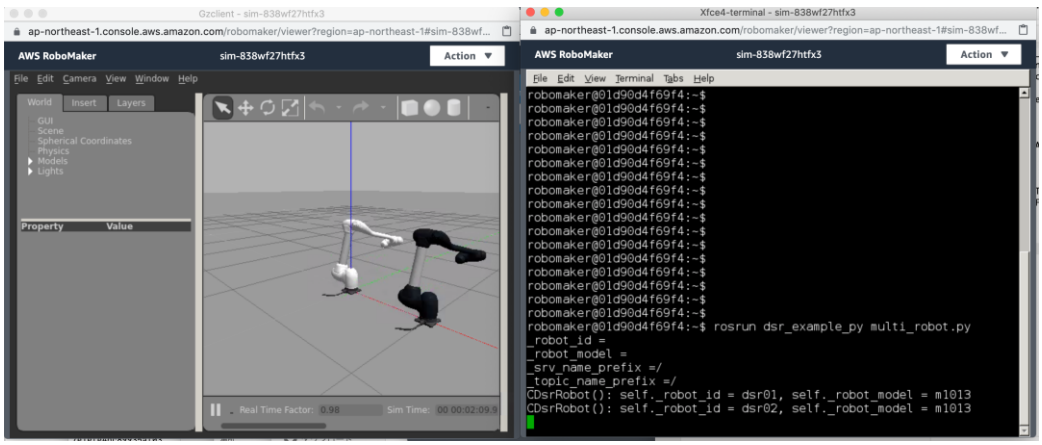
  Run -> Launch Simulation-> doosan-robot.

  Simulation -> View Simulation Job Details.

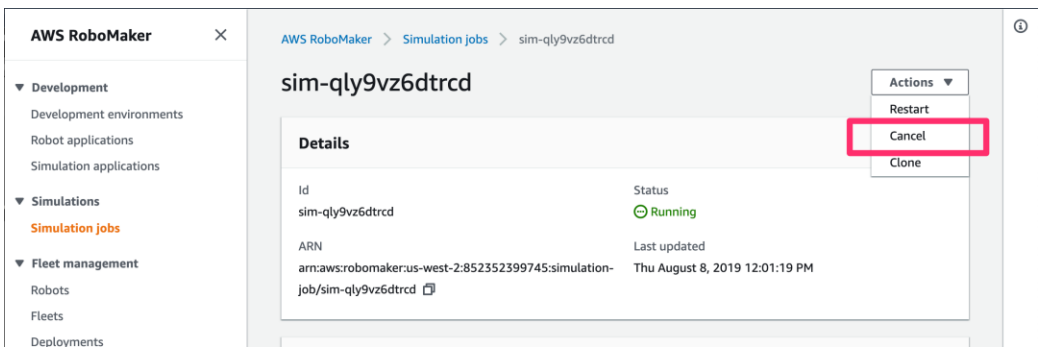


- 터미널 아이콘을 클릭하여 터미널 창을 시작하고 다음 명령을 입력하십시오.

```
rosrun dsr_example_py multi_robot.py
```



- **[3.8]** 시뮬레이션을 마치려면 시뮬레이션 작업 세부 사항 페이지로 돌아가서 **[Actions]** -> **[Cancel]** 를 누릅니다.



4. Create robot application

AWS RoboMaker는 ROS 애플리케이션을 "**Robot Application**"와 "**Simulation Application**" 두 가지 유형으로 유지 관리합니다. "**Robot Application**"은 시뮬레이션과 실제 로봇을 모두 시작할 수 있는 응용 프로그램입니다. 반면 "**Simulation Application**"으로 유지 관리되는 ROS 응용 프로그램은 시뮬레이션 환경에서만 시작할 수 있습니다.

	In Simulation Job	Deploy to robot	Main purpose
Simulation Application	Mandatory	N/A	- Launch simulation environment - Launch simulation robot
Robot Application	Optional (Simulation can be launched without robot application)	Yes	- Control robot - ROS node which interact with HW

지금까지는 **Simulation applicatiton** 에서만 다루었습니다. 이제 **Robot application** 을 살표 보겠습니다. Doosan ROS 패키지 소스를 AWS RoboMaker RoboMaker 연동을 위해서 더 적합한 디렉터리 구조를 재 구성했습니다. 이를 통하여 Robot application 및 Simulation application 으로 구성되었습니다.

다음은 재 구성된 기본 파일 구조입니다.

doosan-robot
+ robot_ws --- Robot application
  + src -- source directory for robot app. ROS packages related to the application stays
+ simulation_ws --- Robot application
  + src -- source directory for simulation app. ROS packages related to the application stays
roboMakerSettings.json --- setting file for this project.

두산 ROS 패키지를 AWS RoboMaker에서 사용하기 위해서는 약간의 수정이 필요합니다. 이 수정된 부분에는 **"for aws robomaker"** 라는 주석이 달려 있고, **"for aws robomaker"**를 검색하여 수정 된 부분을 찾을 수 있습니다. 이러한 변경이 필요한 이유를 이해하려면 다음의 링크가 도움이 될 것입니다.

https://aws.amazon.com/jp/blogs/opensource/building-bundling-ros-app-aws-robomaker/

Package	SIMULATION	ROBOT	Note
common	✔	✔	
doosan_robot	✔	✔	
dsr_control	✔	✔	
dsr_description	✔	✔	
dsr_example	✔	✔	
dsr_gazebo	✔		
dsr_launcher	✔	✔	
dsr_msgs	✔	✔	
moveit_config_m0609	✔	✔	
moveit_config_m0617	✔	✔	
moveit_config_m1013	✔	✔v	
moveit_config_m1509	✔	✔	

simulation job 은 이미 구성되었지만 Robot Application 이 포함되지 않았습니다. 이제 Robot Application 을 포함시켜보겠습니다.

- **[4.1] roboMakerSettings.json** 파일을 엽니다. robot application에 simulation job을 포함하도록 파일을 편집하십시오.

  - "simulation"과"simulationApp"사이에 아래 항목을 추가하십시오.

  - 다음은 추가 할 항목입니다.

```
"robotApp": {
   "name": "<robot app name>",
   "s3Bucket": "<bucket name>",
   "sourceBundleFile": "./doosan-robot/robot_ws/bundle/output.tar",
   "architecture": "X86_64",
   "robotSoftwareSuite": {
      "version": "Kinetic",
      "name": "ROS"
      },
   "launchConfig": {
      "packageName": "dsr_launcher",
      "launchFile": "aws_bringup_robot.launch"
      }
},
```

항목을 추가 한 후 다음 부분을 수정하십시오:

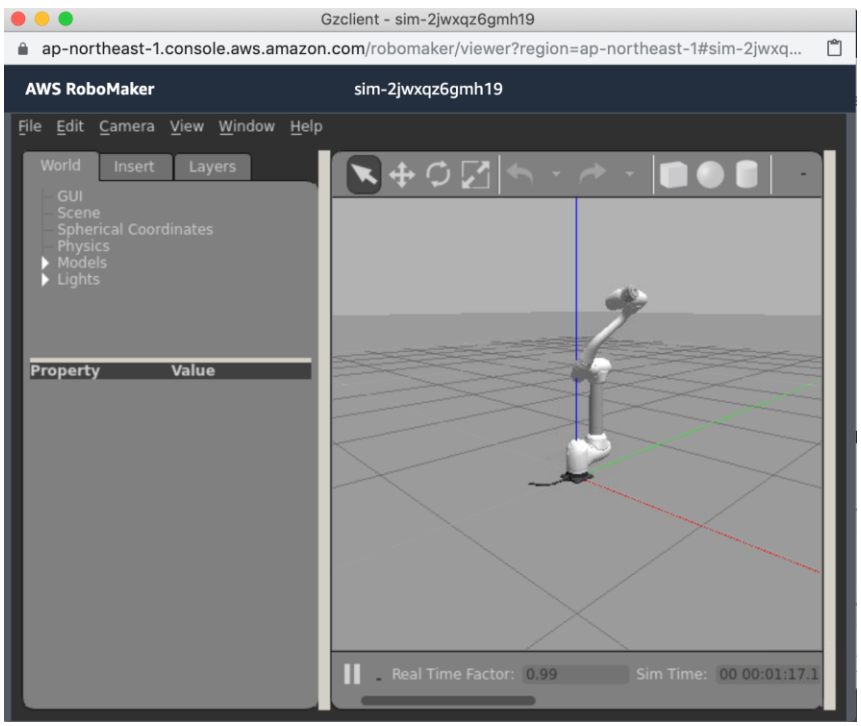
- **<robot app name>**의 경우 **doosan-robot/ws_settings.yaml** 의 **robot_app_name** 로 바꾸십시오.
- <**bucket name**>의 경우 **doosan-robot/ws_settings.yaml**의 **bucket_name** 으로 바꾸십시오.
- [Ctrl] + s를 눌러 변경 사항을 저장합니다.

(Note: ws_settings.yaml 파일은 setup.sh 설정 스크립트에 의해서 기록됩니다.)

- **[4.2]** 메뉴에서 ***Run-> Build-> doosan robot robot-app***를 선택하여 로봇 응용 프로그램을 빌드하고 ***Run-> Bundle-> doosan robot robot-app***를 선택하여 로봇 응용 프로그램을 번들링합니다.

- **[4.3]** 메뉴에서 ***Run-> Launch Simulation-> doosan-robot***을 선택하여 시뮬레이션 작업을 다시 시작하십시오.

- **[4.4]** 이 때 로봇 응용 프로그램과 시뮬레이션 응용 프로그램이 생성되고 시뮬레이터상에서 실행됩니다. 로봇 응용 프로그램은 **dance_m1013.py** (**dsr_launcher** 패키지의 **aws_bringup_robot.launch**를 통해)를 시작되므로 시뮬레이션이 시작되면 로봇이 움직이기 시작합니다.



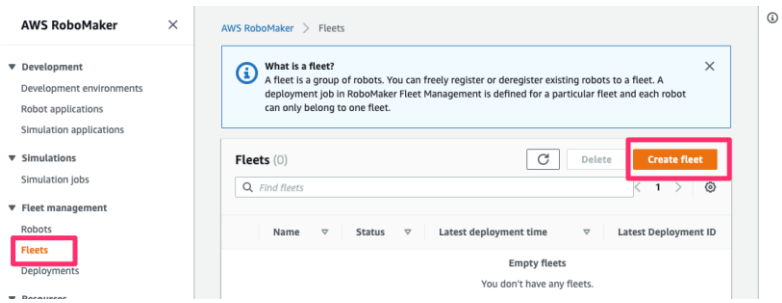
5. Create fleet

Fleet Management 는 로봇을 관리하는 기능입니다. 이를 사용하면 로봇 응용 프로그램을 로봇에 원격으로 설치할 수 있습니다.

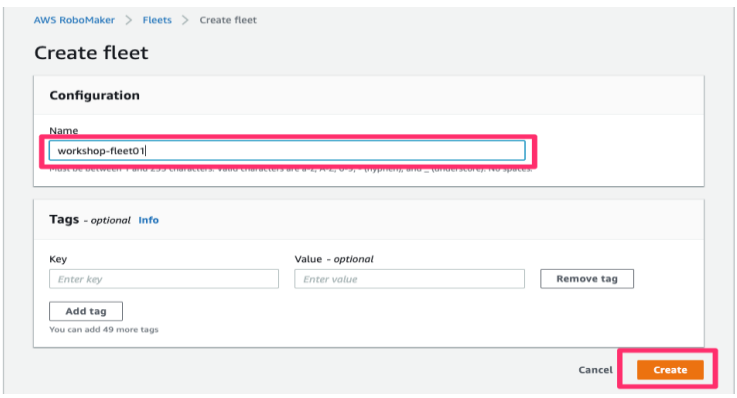
Fleet management 사용 방법을 이해하려면, 먼저 **fleet**을 생성 한 다음 로봇을 등록하고 로봇을 fleet에 추가하십시오. **fleet**은 로봇 응용 프로그램을 유지 관리하는 일종의 그룹입니다. 로봇 응용 프로그램을 fleet을 통해 배포하면 **fleet**에 속해있는 모든 로봇이 자동으로 응용 프로그램을 다운로드 하여 자체 설치합니다.

- **[5.1]** AWS RoboMaker (**https://console.aws.amazon.com/robomaker**) 을 엽니다.

- **[5.2]** 왼쪽의 탐색 트리에서 **Fleets**을 선택하면. Fleet 목록이 표시됩니다. 오른쪽 상단의 **[Create fleet]** 버튼을 선택하십시오.



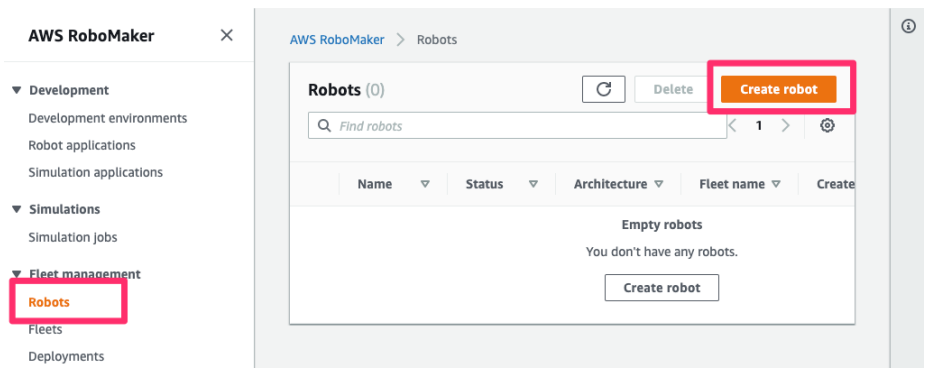
- **[5.3] Create fleet** 창이 표시됩니다. **Name** 필드에 fleet 이름을 입력하고 **[Create]** 버튼을 클릭하십시오.



6. Register a robot to AWS RoboMaker

다음으로 fleet management 에 로봇을 등록하겠습니다. 그러면 AWS RoboMaker는 로봇을 제어할 수 있습니다.

- **[6.1]** 왼쪽의 탐색 메뉴에서 **Fleet management** 의 **Robots** 을 선택하십시오. 로봇 목록이 표시되고, 오른쪽 상단의 **[Create robot]** 버튼을 선택하십시오.



- **[6.2]** Robot 생성 창이 나타나게 되고, 입력 필드를 적절히 설정하십시오.

  - **Name**: 로봇 이름입니다. 쉽게 알아 볼 수 있는 임의의 이름을 지정.

  - **Architecture**: 로봇 응용 프로그램이 실행될 CPU 아키텍처. **X86_64**를 선택.

  - **AWS Greengrass group**: 선택 **[Create new]**

  - **AWS Greengrass prefix**: 로봇의 이름이 자동으로 적용됩니다. 그냥 동의합니다.

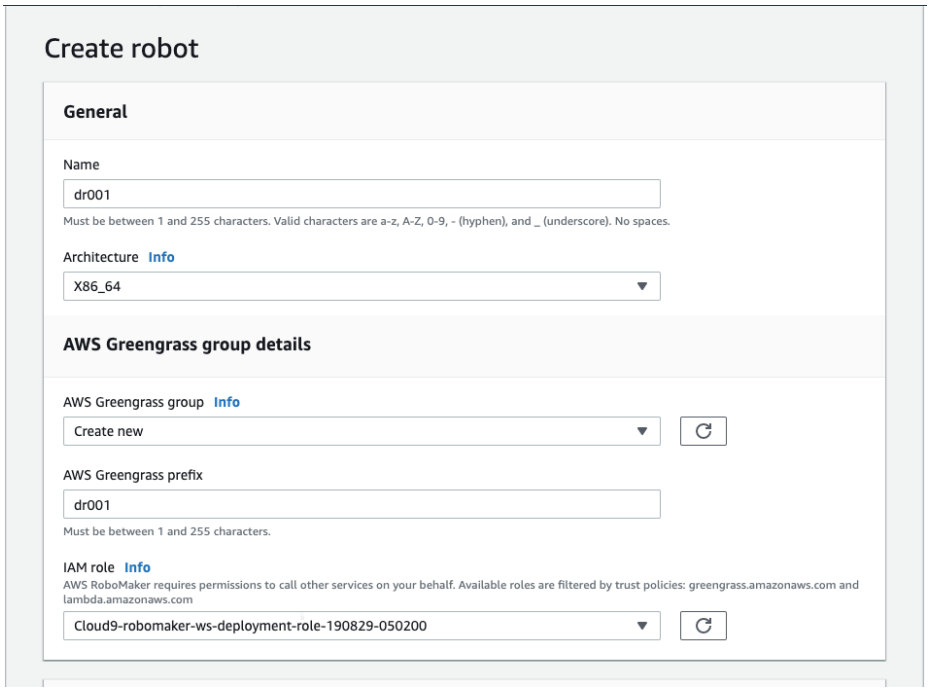
  - **IAM Role**: **doosan-robot/ws_settings.yaml** 파일에서 **iam_role_for_deployment**의 값을 찾으십시오.

    설정 스크립트에 의해 role 이 생성되었습니다.

    (이 IAM role은 AWS 리소스의 액세스 권한을 정의하는 데 사용됩니다.

    이 role 을 수동으로 생성하려면 다음을 참조하십시오.

    https://docs.aws.amazon.com/robomaker/latest/dg/create-robot.html#create-robot-role)

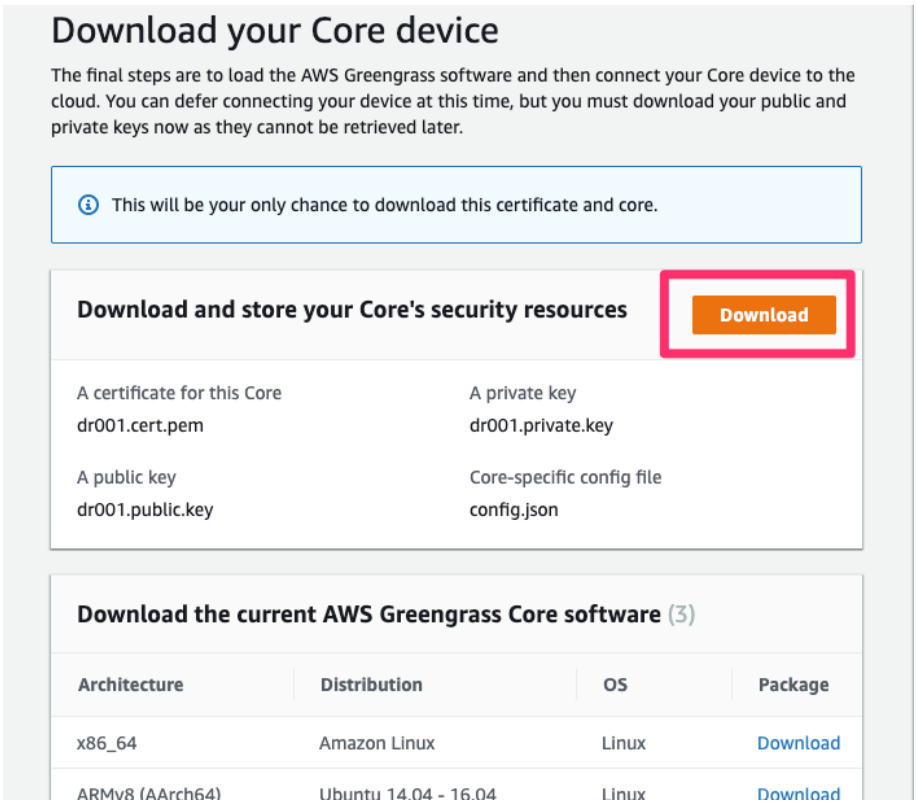


- **[6.3]** 오른쪽 하단의 **[Create]** 버튼을 클릭하면 **"Download your Core device"** 창이 표시됩니다.

- **[6.4]** **"Download your Core device"** 에서 다운로드 한 파일을 로봇을 제어하는 ROS Master PC에 설치해야 합니다. 먼저 **Download and store your Core's security resources**. 옆에 있는 **[Download]** 버튼을 클릭하십시오. 이 보안 키 파일 및 설정을 다운로드 합니다. 여기에서만 개인 키를 다운로드 할 수 있으며 이 페이지로 돌아올 수 없습니다. 따라서 파일 권한을 다운로드 하여 안전한 위치에 저장하십시오.

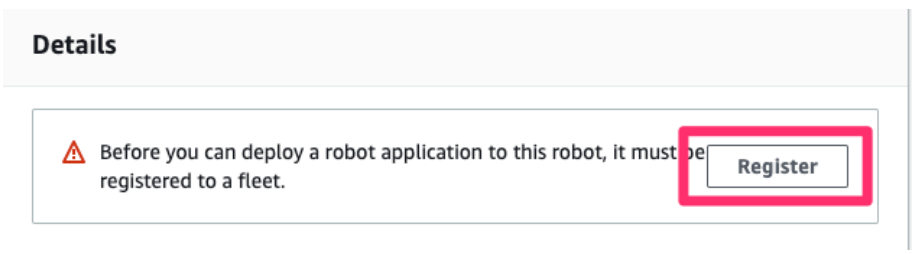
이 저장된 압축 파일은 **7장 Setup the ROS Master PC** 에서 사용될 예정입니다.

- **[6.5]** **Download the current AWS Greengrass Core software** 에서 AWS IoT **Greengrass** 소프트웨어를 다운로드 할 수 있습니다. 이 소프트웨어를 로봇을 제어하는 ROS Master PC에 설치해야 합니다. **Greengrass를** ROS Master PC에 설치하는 방법은 **7장 Setup the ROS Master PC** 를 참조하시길 바랍니다.

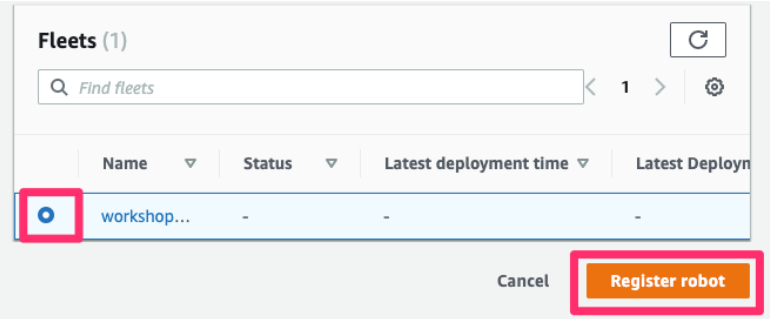


- 우측 하단의 **[View robot]** 버튼을 누릅니다.

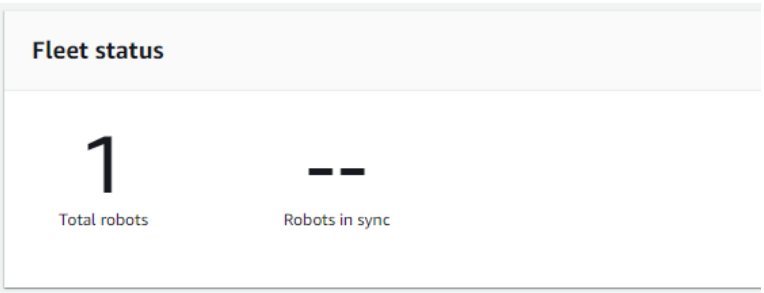
- **"Details"** 페이지가 표시됩니다. 왼쪽 상단에 **[Register]** 버튼이 있습니다. 이를 통해 로봇을 **fleet**에 등록 할 수 있습니다. 버튼을 클릭하고 방금 생성 한 **fleet**에 로봇을 등록합니다.



- "**Select fleet to register robot to**" 창이 나타납니다. 라디오 버튼을 선택하고 **[Register robot]** 버튼을 클릭합니다.



- 로봇은 이제 방금 생성 한 fleet에 속합니다. fleet 의 세부 정보 페이지 아래에 있는 링크를 클릭하면 fleet 페이지로 이동합니다. Fleet 페이지에 총 로봇 수가 1로 표시됩니다.



이제 배포를 위한 AWS RoboMaker 설정이 완료되었습니다. 그럼 다음 장에서 로봇 구동하는 ROS Master PC 을 설정하겠습니다.

7. Setup the ROS Master PC

AWS RoboMaker에 연결할 수 있도록 로봇을 제어하는 ROS Master PC를 설정하겠습니다. 이 단계를 완료하면 ROS Master PC가 AWS RoboMaker 에 연결됩니다. ROS Master PC 에 Greengrass 를 설치 해야만 합니다. 상세한 내용은 하기 링크를 참고 바랍니다.

https://docs.aws.amazon.com/greengrass/latest/developerguide/setup-filter.other.html

하기 작업은 ROS Master PC 상에서 이루어 져야 합니다.

- **[7.1]** Setup to install AWS IoT Greengrass

  - 하기 명령을 실행하여 ggc_group 그룹에 ggc_user 계정을 생성합니다.

```
sudo adduser --system ggc_user
sudo addgroup --system ggc_group
```

  - 하기 명령을 실행하여 AWS IoT Greengrass 설치 전에 필요한 소프트웨어를 찾습니다..

```
cd /
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
```

  - Greengrass 설치 전 요구하는 소프트웨어를 설치합니다.

  - Ubuntu 16.04 기준으로 python 업그레이드, nodejs, java8 의 설치가 필요 합니다.

  해당 프로그램 설치 후, 실행파일들을 Greengrass 가 요구하는 이름으로 심볼릭 링크를 걸어야 함을 유의바랍니다.

- **[7.2]** Download greengrass-linux-x86-64-1.9.2.tar.gz

  https://d1onfpft10uf5o.cloudfront.net/greengrass-core/downloads/1.9.2/greengrass-linux-x86-64-1.9.2.tar.gz

  - and extract the file to root

```
sudo tar -xzvf greengrass-linux-x86-64-1.9.2.tar.gz -C /
```

- **[7.3]** Execute following to place root CA.

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

- **[7.4] 6장. Register a robot to AWS RoboMaker** 에서 다운로드 한 보안 리소스 파일 (**<robot name>-setup.zip**)을 ROS master PC에 복사하십시오.

- **[7.5]** ROS Master PC 에서 다음 명령을 실행하여 리소스 파일을 greengrass 폴더에 압축 푸십시오. 기존 파일을 새 파일로 덮어 쓸지 묻는 메시지가 표시 될 수 있습니다. 모든 것을 덮어 쓰려면 “A”라고 답하십시오.

```
sudo unzip <robotname>-setup.zip -d /greengrass
```

  (<robotname> -setup.zip을 복사 한 실제 파일 이름으로 바꾸십시오.)

- **[7.6]** ROS Master PC 에서 다음 명령을 실행합니다. 성공하면 "Greengrass가 성공적으로 시작되었습니다"라는 메시지가 표시됩니다.

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

이제 로봇이 준비되었습니다. 다음 장에서 배포를 시작해 보도록 하겠습니다.

8. Build & Bundle robot app

4장이 완료된 후에 소스 코드를 변경하지 않았으면 이 장을 수행 할 필요가 없습니다. 소스 코드를 변경하고 시뮬레이션에서 테스트하기 전에 소스 코드를 배포하려면 이 단계가 필요합니다.

- **[8.1]** Do build and bundle.

  - 개발 환경 메뉴에서 ***Run -> Build -> doosan robot robot-app*** 실행, ***Run -> Bundle -> doosan robot robot-app*** 을 실행하십시오. 로봇 응용 프로그램이 컴파일 되고 번들링 됩니다.

- **[8.2]** 최신으로 생성된 **bundle** 을 **S3** 로 업로드

  - 본 작업은 자동 혹은 수동으로 할 수 있습니다. 자동으로 수행하는 방법을 추천합니다.

  - [자동으로 수행하는 방법]

    시뮬레이션 작업 ***Run -> Launch Simulation -> doosan-robot*** 을 실행하면 자동으로 bundle 이 S3로 업로드 됩니다..

  - [수동으로 수행하는 방법]

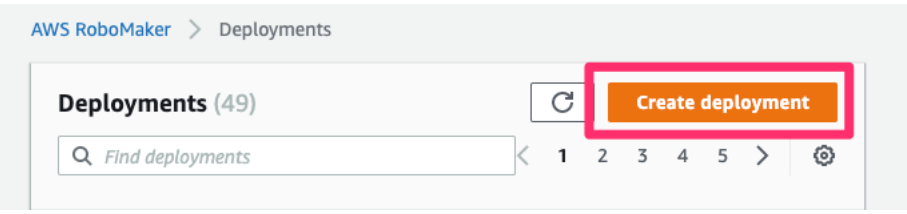
    AWS RoboMaker 개발 환경의 터미널에서 다음을 명령을 수행 하십시오.

```
bucket=<buckt_name>
aws s3 cp ~/environment/doosan-robot/robot_ws/bundle/output.tar
s3://${bucket}/robot_ws/bundle/output.tar
```

<bucket_name>을 doosan-robot/ws_settings.yaml 의 bucket_name으로 바꾸십시오.

9. Deploy to a robot

- **[9.1]** AWS Management Console에서 AWS RoboMaker를 엽니다. 왼쪽의 탐색 메뉴에서 **Fleet management**. 의 **Deployments** 를 선택합니다. 오른쪽 상단의 **[Create deployment]** 버튼을 클릭하십시오.

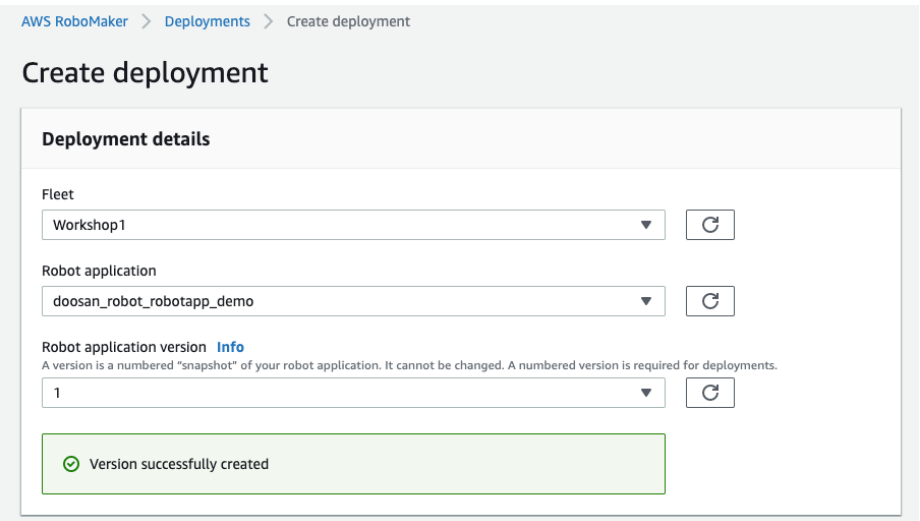


- **[9.2]** 배포 생성 창이 표시됩니다.

  - **Fleet:** 방금 만든 fleet을 선택하십시오.

  - **Robot application:** 만든 응용 프로그램을 선택하십시오. 시뮬레이션을 시작할 때 이름이 자동으로 생성되고 등록됩니다. **doosan-robot/ws_settings.yaml** 파일의 **robot_app_name** 값에서 로봇 응용 프로그램 이름을 찾을 수 있습니다.

  - **Robot application version:** 새 버전을 만들려면 새로 만들기를 선택하십시오.



- **[9.3]** 배포 시작 구성에서 다음과 같이 설정하십시오:

  - **Package name:** dsr_launcher

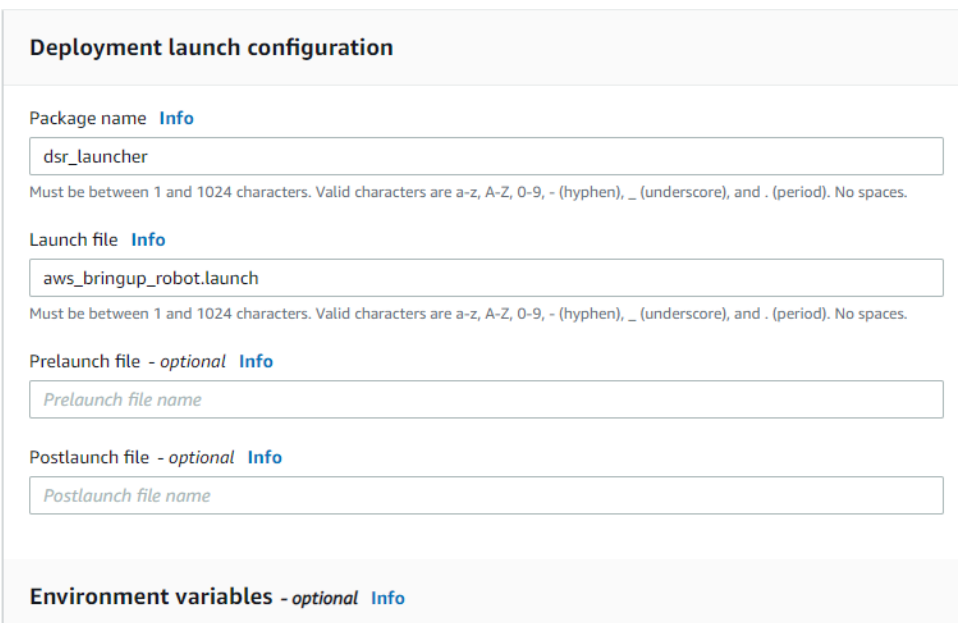
  - **Launch file:** aws_bringup_robot.launch

  - **Prelaunch file:** (optional) 여기에서 설정하면 ROS 응용 프로그램이 시작되기 전에 스크립트 파일을 실행할 수 있습니다.

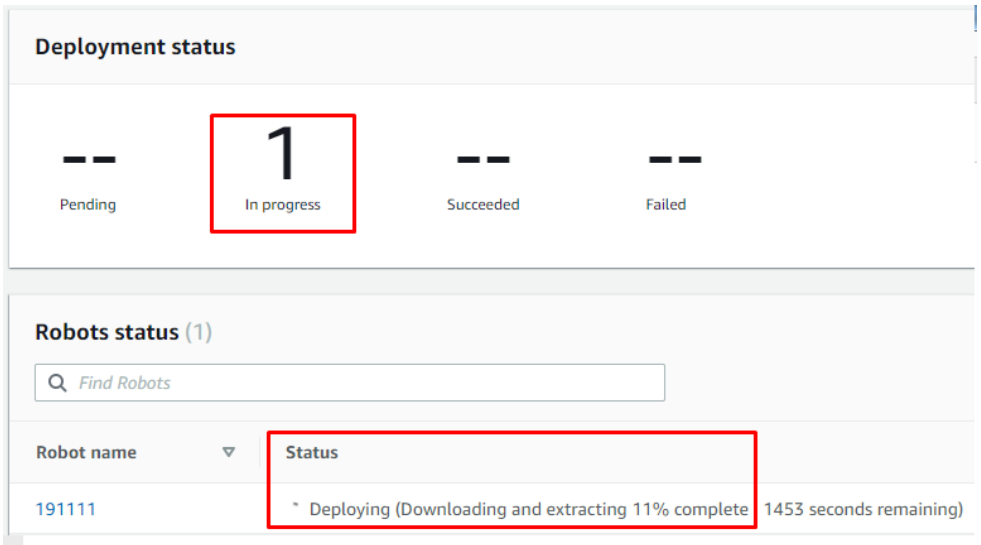
  - **Postlaunch file:** (optional). 여기에서 설정하면 ROS 응용 프로그램이 시작된 직후 스크립트를 실행할 수 있습니다.

  - **Environment variables:** (Optional) 여기에서 환경 변수를 정의 할 수 있습니다.

  환경 변수는 시작 ROS 애플리케이션에 제공됩니다.



- **[9.4]** 우측 하단의 **[Create]** 버튼을 클릭하십시오. 배포가 시작됩니다.



- 모든 것이 올바르게 되면 성공 횟수는 1이되고 로봇은 움직이기 시작합니다.

- **Note:**

  - 첫 배포에는 시간이 다소 걸립니다. 처음에는 모든 bundle을 다운로드하기 때문입니다. 약간만 변경하고 배포하면 배포 시간이 단축됩니다. fleet management 에서 변경된 부분 만 다운로드하기 때문입니다.

  - 동일한 설정으로 다시 배포하려는 경우 기존 배포에서 **[Action] -> [Close]**를 선택할 수 있으며 배포 구성 부분을 다시 설정할 필요가 없습니다.

10. References

- **Links:**

  - **Working with Robot Applications**

    https://docs.aws.amazon.com/robomaker/latest/dg/managing-robot-applications.html

  - **Working with Simulation Applications**

    https://docs.aws.amazon.com/robomaker/latest/dg/managing-simulation-applications.html

  - **Fleet management**

    https://docs.aws.amazon.com/robomaker/latest/dg/fleets.html

Doosan Robotics

www.doosanrobotics.com