# Jacinto Connected Cars on AWS: Getting Started Guide

**Version 2021.10.25.1**

## 1 Overview

Jacinto 7 processors are popular choices by top-tier automotive suppliers for vehicle compute, addressing the rapid growth of telematics applications such as Vehicle-to-cloud (V2C), Vehicle-to-infrastructure (V2I) and Vehicle-to-Vehicle (V2V) technologies. With the tremendously increased volume of vehicle data, it is essential to have a reliable cloud infrastructure to manage and extract intelligence based on these data, and provide valuable services to end users and critical road data to vehicle manufactures for maintenance, over-the-air updates, and Artificial Intelligence (AI) model improvements.

Amazon Web Services (AWS) Connected Mobility Solutions (CMS) provides a suite of solutions for OEMs and mobility service providers to quickly deploy services on the AWS cloud.

*Figure 1* shows an example TCU/Domain Controller software architecture based on the DRA82x Processor SDK. A TCU can be deployed as a single IOT device (a Thing) or each of the domains can be a separate device. A Greengrass core is deployed, for filtering, analyzing, and buffering of telematics data at the vehicle premise. Greengrass also buffers vehicle data when network is not available as the vehicle moves.

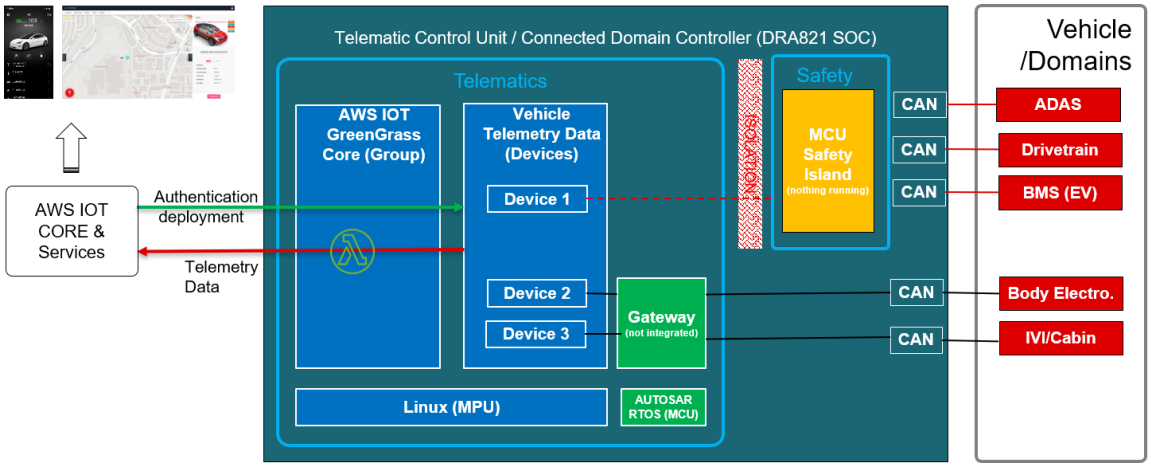



Figure 1: Vehicle Compute (Telematics) Architecture based on DRA82X Gateway Processors

In this guide, we walk through detailed steps to install AWS Greengrass application on the DRA829 and DRA821 device EVMs, connect to AWS IOT core, and then integrate with Fleet Management demo with CMS solutions.

This guide is primarily written based on TI Processor SDK (https://www.ti.com/tool/PROCESSOR-SDK-J721E) and the J721E EVM, with additional instructions given for:

- TDA4VM processor starter kit for Edge AI vision systems (SK-TDA4VM)
- DRA821x/J7200 EVM
- Subsequent Processor SDK releases from TI
- Subsequent Greengrass releases from AWS

Therefore, attention shall be given on minor changes in future release configurations.

# 2 Required Hardware Platform (Target Board)

Required target boards including the Jacinto 7 Common Processor board and one of the Jacinto 7 evaluation modules:

- Common Processor board for Jacinto™ 7 processors (J721EXCPXEVM, https://www.ti.com/tool/J721EXCPXEVM)
- TDA4VM and DRA829V system-on-module (J721EXSOMXEVM, https://www.ti.com/tool/J721EXSOMXEVM)
- DRA821 system-on-module (J7200XSOMXEVM, https://www.ti.com/tool/J7200XSOMXEVM)
- TDA4VM processor starter kit for Edge AI vision systems (SK-TDA4VM, https://www.ti.com/tool/SK-TDA4VM).

Note that the Common Processor Board is required for either TDA4 or DRA821 SOM modules, but the SK-TDA4VM does not require the Common Processor Board.

Refer to the Quick Start Guide here and the User's Guide here for important information regarding setting up the TDA4 EVM as well as technical specifications. Similar documentation for DRA821 and SK-TDA4VM can be found in the product pages listed above.

# 3 Prepare Software on Target Board

Greengrass 2.x can be installed to the standard DRA82x Linux file system. This section provides instructions to:

- Create a standard boot SD card
- Build and Install Greengrass

If a user already has a bootable SD card or obtained required binary packages for Greengrass, steps to build them can be skipped.

These build and installation steps are also verified with Greengrass v1.1, with slight difference of dependent packages.

## 3.1 Creating a standard boot SD card

Note: SDK version numbers in URLs given in this section may be updated to the latest versions as needed. Additionally, for DRA821 and SK-TDA4VM, use their specific SDK releases.

Step 1:
Download latest Processor SDK at:
https://www.ti.com/tool/PROCESSOR-SDK-J721E (DRA829/TDA4)
https://www.ti.com/tool/PROCESSOR-SDK-J7200 (DRA821x)
https://www.ti.com/tool/download/PROCESSOR-SDK-LINUX-SK-TDA4VM (SK-TDA4VM)
and install to a Linux host. Detailed instructions are available at:
https://software-dl.ti.com/jacinto7/esd/processor-sdk-linux-jacinto7/08_00_00_08/exports/docs/linux/Overview/Download_and_Install_the_SDK.html

Step 2:
Use the supplied shell script to prepare the SD card with boot files and Linux file system, by issue:

```
sudo <SDK_INSTALL_DIR>/bin/mksdboot.sh --device /dev/sdX --sdk <SDK INSTALL DIR>
#Replace the /dev/sdX with appropriate device name of the SD card
```

Refer to the SDK Getting Started Guide at:
https://software-dl.ti.com/jacinto7/esd/processor-sdk-linux-jacinto7/08_00_00_08/exports/docs/linux/Overview/Processor_SDK_Linux_Formatting_SD_Card.html

in case errors or further details are needed.

Alternatively, the out-of-box demo software, such as:
https://software-dl.ti.com/jacinto7/esd/processor-sdk-rtos-jacinto7/08_00_00_12/exports/docs/psdk_rtos/docs/user_guide/out_of_box_j721e.html

also provide binary and instructions to prepare bootable Linux SD card.

Step 3:
Test the SD card. Insert the prepared card to the MicroSD card slot on the common processor board, and confirm the board is configured for SD card boot:

```
SW8[1-10]: 10000010 (Switch 1 to 8 order, "1" = ON)
SW9: All switches in OFF
```

If having trouble booting the SD card, try one-time reset the u-boot environment variables by:

1. Stop the boot at u-boot
2. Issue following command at u-boot:
   ```
   env default -a -f
   saveenv
   ```
3. Power off and reboot

Additional details and troubleshooting can be referenced at:
https://software-dl.ti.com/jacinto7/esd/processor-sdk-linux-jacinto7/08_00_00_08/exports/docs/linux/Foundational_Components/U-Boot/UG-Memory.html

### 3.2 Build and Install Greengrass

This section we build the Greengrass and required dependencies and install to the target filesystem on the SD card created in the previously section.

If you are using a preinstalled SD card from the EVM kit, you may following Step 1 in Section 3.1 to install Processor SDK Linux on a host PC first.

Step 1:
Download the toolchain to the host Linux PC:

```
mkdir $HOME/toolchains

wget https://developer.arm.com/-/media/Files/downloads/gnu-a/9.2-2019.12/binrel/gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabihf.tar.xz

tar xvf gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabihf.tar.xz -C $HOME/toolchains

wget https://developer.arm.com/-/media/Files/downloads/gnu-a/9.2-2019.12/binrel/gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz

tar xvf gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz -C $HOME/toolchains
```

Step 2:
Update Yocto build layers with the latest meta-aws release

On the host PC, edit:
<SDK_INSTALL_DIR>yocto-build/build/processor-sdk-linux/processor-sdk-linux-<SDK_VERSION>.txt

Update meta-aws release tag to "10b45cf5d9459122e55d2e9891e8fea7b4f38911". Latest tag can be obtained from:
https://github.com/aws/meta-aws/tree/dunfell

make sure to choose the branch that matches the Yocto build (dunfell).

Step 3:
Add Greengrass as a yocto build layer

```
cd <SDK_INSTALL_DIR>yocto-build
./oe-layertool-setup.sh -f configs/processor-sdk-linux/processor-sdk-linux-
<SDK_VERSION>.txt
cd build
echo "INHERIT += \"own-mirrors\"" >> conf/local.conf
echo "SOURCE_MIRROR_URL = \"https://software-dl.ti.com/processor-sdk-mirror/sources/\""
>> conf/local.conf
echo "ARAGO_BRAND = \"psdkla\"" >> conf/local.conf
echo "DISTRO_FEATURES_append = \" virtualization\"" >> conf/local.conf
echo "IMAGE_INSTALL_append = \" docker\"">> conf/local.conf
echo "IMAGE_INSTALL_append = \" greengrass-bin\"">> conf/local.conf
echo "IMAGE_INSTALL_append = \" corretto-11-bin\"">> conf/local.conf
. conf/setenv
```

Step 4:
Build Greengrass packages
Issue build commands from:

```
cd <SDK_INSTALL_DIR>yocto-build/build
```

For DRA829/TDA4VM:

```
TOOLCHAIN_BASE=<TOOLCHAIN_BASE> MACHINE=j7-evm bitbake -k greengrass-bin
```

For DRA821:

```
TOOLCHAIN_BASE=<TOOLCHAIN_BASE> MACHINE=j7200-evm bitbake -k greengrass-bin
```

For Greengrass V1.1, user shall replace `greengrass-bin` with `greengrass`.

Note that TOOLCHAIN_BASE need to be updated with path where user's toolchain is installed.

Upon successful build, installers will be at:

```
<SDK_INSTALL_DIR>yocto-build/build/arago-tmp-external-arm-glibc/deploy/ipk/aarch64

        greengrass_<VERSION>_aarch64.ipk
        sqlite3_<VERSION>_aarch64.ipk
        corretto-11-bin_<VERSION>_aarch64.ipk
```

Note that `sqlite3` package is only required for Greengrass Version 1.1, and `corretto` is only required by Greengrass Version 2.x. Copy these installer binaries to a directory on the target SD card.

Debug Information

In case of misconfigurations or errors, user can remove all source and build files by manually deleting the following directories:

```
cd <SDK_INSTALL_DIR>yocto-build
rm -Rf build sources
```

then restart from Step 2 above.

Step 5:
Install Greengrass to target file system

Boot the board with the SD card, go to the directory where the installers are copied to, issue:

```
opkg install corretto-11-bin_<VERSION>_aarch64.ipk
opkg install greengrass_<VERSION>_aarch64.ipk
```

```
opkg install sqlite3_<Version>_aarch64.ipk  [only needed for V1.1]
```

Once this completes, verify that java has been installed using the command:

```
java – version
java -jar <GG_Install_Dir>/lib/Greengrass.jar --version
```

where <GG_Install_Dir> is the installed Greengrass directory. If this command executes successfully, the pre-requisites for Greengrass v2 are complete.

Alternatively, latest Greengrass software may be downloaded and installed with the following command on the host:

```
wget https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip
unzip greengrass-nucleus-latest.zip -d <GGCoreInstall>
rm greengrass-nucleuslatest.zip
```

Replace `<GGCoreInstall>` with the folder that you want to use.

# 4 Setup Your AWS Account and Permissions

Step 1:
Setup an AWS account

If you don't have an AWS account, please refer to the instructions at Set up your AWS Account. Follow the steps outlined in the following sections to create your account and a user and get started:

- Sign up for an AWS account and
- Create a user and grant permissions.
- Open the AWS IoT console

Pay special attention to the Notes. Note that certain AWS services used in the demo are configured beyond the free-tier and may incur cost. Therefore, it is not recommended to use the free-tier account.

Step 2:
Create an IAM user account

It is recommended to deploy CMS under an IAM user account, instead of using the root account. To create a IAM user:

1. Log in to the AWS root account just created, go to the AWS management console and choose IAM services
2. Under Access management, select Users and Add a new user, then following the prompt to create a new user
3. Make sure to save the access key ID and Secret Access Key. Note the secret access key can only be saved when a user is created.

Step 3 (optional):
Install the AWS Command Line Interface

Installing the AWS CLI is needed to complete the instructions in this guide.
To install the AWS CLI on your host machine, refer to the instructions at Installing the AWS CLI v2.
AWS CLI can also be installed to the target board. See instructions later.

# 5 Provision Greengrass to IOT Core (Automated Provisioning)

It is recommended that you use the quick installation option, leaving the task of setting up Greengrass to the installer.
To enable this, the `--provision true` option will be used. For more details, refer to https://docs.aws.amazon.com/greengrass/v2/developerguide/quick-installation.html

Step 1:
Provide your credentials

On the target board, set AWS access credentials:

```
export AWS_ACCESS_KEY_ID=<the access key id for your user>
export AWS_SECRET_ACCESS_KEY=<the secret access key for your user>
```

Those two key parameters are from your account – those credentials are for the user you want to act as.

Step 2:
Boot the DRA8xx target board, go to the installed Greengrass directory. Run the installer as shown below. Update values below to reflect the configuration of your device, account and region.

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
-jar ./GGCoreInstall/lib/Greengrass.jar \
--aws-region region \
--thing-name MyGreengrassCore \
--thing-group-name MyGreengrassCoreGroup \
--tes-role-name GreengrassV2TokenExchangeRole \
--tes-role-alias-name GreengrassCoreTokenExchangeRoleAlias \
--component-default-user ggc_user:ggc_group \
--provision true \
--setup-system-service true \
--deploy-dev-tools true
```

You may replace thing-name, thing-group-name with your own. If all goes well, amongst other messages you will see the following output on the device console:

```
Successfully configured Nucleus with provisioned resource details!
Configured Nucleus to deploy aws.greengrass.Cli component
Successfully set up Nucleus as a system service
```

**NOTE**: Pay special attention to installer messages in the console output. Some manual instructions (such as attaching a policy) may be given.

Verify the following:

```
The GreengrassV2TokenExchangeRoleAccess policy has been attached to the role
GreengrassV2TokenExchangeRole.
```

The local development tools (specified by the `--deploy-dev-tools true` option) take some time to deploy.

If you installed AWS CLI on a host, the following command can be used to check the status of this deployment:

```
aws greengrassv2 list-effective-deployments --core-device-thing-name MyGreengrassCore
```

When the status is SUCCEEDED, run the following command to verify that the Greengrass CLI is

installed and runs on your device. Replace /greengrass/v2 with the path to the base folder on your device as needed.

```
/greengrass/v2/bin/greengrass-cli help
```

Step 3
Create Hello World Component

In Greengrass v2, components can be created on the edge device and uploaded to the cloud, or vice versa.

Follow the instructions online under the section To create a Hello World component to create, deploy, test, update and manage a simple component on your device.

Step 4
Upload the Hello World component

Follow the instructions online at Upload your component to upload your component to the cloud, where it can be deployed to other devices as needed.

### Troubleshooting

Refer to the instructions at Troubleshooting Greengrass v2 for information on:

- How to View AWS IoT Greengrass Core software logs
- How to View component logs
- AWS IoT Greengrass Core software issues
- AWS IoT Greengrass cloud issues
- Core device deployment issues
- Core device component issues

You can also refer to Logging and Monitoring to learn how to log API calls, gather system health telemetry data, and check core device status.

## 6 Deploy CMS Fleet Management Demo to the AWS Account

To visually demonstrate connectivity of the DRA8xx EVM as an IOT Device, we deploy AWS CDF and CMS Fleet Management demo to visualize data sent to the IOT core.

Refer to: Section 1, *Build and Deploy CMS from Source*
**Connected Mobility Solution Developer Guide: Documents and tools for building CMS-based Solutions**
To be published at:
https://aws.amazon.com/solutions/implementations/aws-connected-vehicle-solution

Note that a Fleet Manager UI web access URL will be given at the end of the deployment, for user to log in and view the map interface.

To verify correct functionality of the Fleet Manager, one can publish single vehicle data to the CMS, or use a software simulation to add vehicles to the Fleet. See

Section 6: *Connecting a Telematics Source (TCU) to CMS with AWS Greengrass* from source
**Connected Mobility Solution Developer Guide: Documents and tools for building CMS-based Solutions**
To be published at:

https://aws.amazon.com/solutions/implementations/aws-connected-vehicle-solution

for detailed instructions.

# 7 Simulate Vehicle Telemetry Data from Target Board to IoT Core

At this point, we successfully provisioned a Greengrass core on the DRA82x board. We are ready to generate some simulated vehicle telematics data to the IOT core from the Greengass core on the target board. We use aws-sample software, aws-connected-mobility-solution-telemetry-device-demo to achieve this goal.

Step 1:
Install latest aws-cms-telemetry-demo to DRA8xx target board. On the DRA8xx console, issue:

```
git clone https://github.com/aws-samples/aws-cms-telemetry-demo
```

this will install latest application on the board.

Step 2:
If this is the first the board is connecting to this AWS account, create AWS account credentials, by issuing:

```
$ mkdir ~/.aws
$ cat >> ~/.aws/config
[default]
aws_access_key_id=YOUR_ACCESS_KEY_HERE
aws_secret_access_key=YOUR_SECRET_ACCESS_KEY
region=YOUR_REGION (such as us-west-2, us-west-1, etc)
```

Step 3:
Install AWS CLI on the DRA8xx EVM:

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-aarch64.zip" -o
"awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

see
https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html#ARM
if you run into any debug issues.

Step 4:
Install required Python packages. Go to the installed `aws-cms-telemetry-demo` directory, execute:

```
python3 -m pip install -r requirements.txt
```

This will install all required python packages shown below:

```
appdirs==1.4.4
awscrt==0.9.15
awsiotsdk==1.5.3
boto3==1.16.43
botocore==1.19.43
certifi==2020.12.5
chardet==4.0.0
distlib==0.3.1
filelock==3.0.12
idna==2.10
importlib-metadata==3.4.0
jmespath==0.10.0
pbr==5.5.1
pyfiglet==0.8.post1
python-dateutil==2.8.1
requests==2.25.1
```

```
s3transfer==0.3.3
six==1.15.0
stevedore==3.3.0
typing-extensions==3.7.4.3
urllib3==1.26.2
virtualenv==20.3.0
virtualenv-clone==0.5.4
virtualenvwrapper==4.8.4
zipp==3.4.0
```

Step 5:
Customize the GPS data file in the similar.

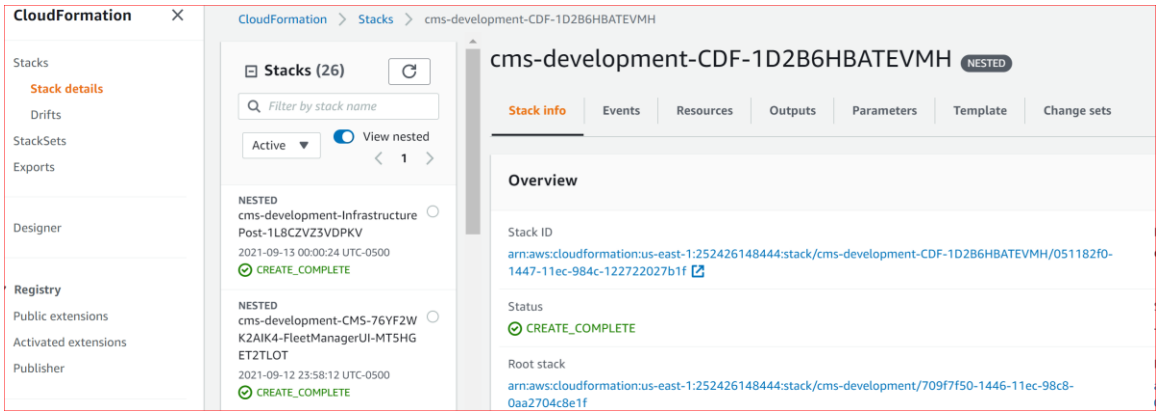
Refer to Appendix I, Creating Some Telemetry to build a customized latLong.csv file and replace the one in the demo. This will allow the demo to generate vehicle trip data of your desire locations.

Step 6:
Obtain CDFStackName from AWS

We would like the simulator to publish vehicle telematics data to the existing CMS that we deployed in previous steps. Thus we need to give the following information:

```
Template profile: default (from Section 6)
stackName: cms-development (from Section 6)
UserName: user name to log into the FleetManager web UI (from Section 6)
Password: password to log into the FleetManager Web UI (from Section 6)
CDFStackName (see below)
```

To obtain the CDFStackName, log into AWS management console, select CloudFormation, click on Stacks on the left pane, uncheck “View Nested” on the right pane, click on “cms-development”, then click on “Resources” menu, and click the weblink for CDF ID, then you will see the CDF stack ID. See screenshot below.




Step 7:
Set up DRA8xx as a vehicle. In the demo directory, issue:

```
python3 ./setupSingleVehicle.py -SkipSetupProvisioning Templates --profile=default --
stackName=cms-development --VIN=LSH14J4C4LA046512 --FirstName=DRA8 --LastName=Jacinto --
Username=jacinto --Password=Jacinto# --CDFstackName cms-development-CDF-1D2B6HBATEVMH
```

Note the option `-SkipSetupProvisioning` tells the script to use existing CMS deployment.
Upon successful execution, user shall see the vehicle on the Fleet Manager UI screen, and also given a web UI URL, which should be the same as the URL before.

Step 8:

Generate simulated trip data

```
python3 ./generateTelemetry.py --profile=default --VIN LSH14J4C4LA046512
```

This script execute simulated trip/route data and publish to the CMS, where the trip can be shown on the Fleet Manager UI.

User shall be able to see live motion of the vehicle on the Fleet Manager web UI.

## 8 Sending an Instant Text Message upon Vehicle Event (future plan only)

In this demo, a Lambda function in the IOT core triggers an instant text message when a vehicle event is sent to the IOT core. A preconfigured cell phone number is provided.

## 9 Enable Edge Inference as Greengrass Component (future plan only)

In this demo, a set of simulated inverter motor operating data is injected to a CAN interface on the target board, simulating a vehicle TCU where powertrain module data is collected. A set of Lambda functions are deployed in the Greengrass core, which perform model-based machine learning and predictive maintenance using the analytics engines on the processor. A subset of the operating data is also sent to the IOT core via Greengrass, allowing the ML model to be refined and then deployed on the edge device.

This demo integrated two existing TI demos:

- Gateway demo that integrates CAN and Ethernet bridges
- Predictive maintenance

## Appendix I: Create some Telemetry

To create a CSV of lat/long coordinates to create a proper simulation of a vehicle along a route, the quickest implementation is to utilize an online maps resource and export a route.
This will provide the most accurate data to simulate your trips and begin build upon other features available in CMS. Below is the procedure to develop that data to be stored in assets/latLong.csv as exported.

1. Go to maps.google.com
2. Click on the hamburger menu and select 'Your Places'
3. At the bottom of the sidebar, select 'CREATE MAP'
4. When the map creation interface loads in a new tab, click the 'Add directions' under the search bar
5. Put in two local landmarks in the city of your choice and the route should appear on the map
6. Click on the 'Untitled Map' dot menu, and select 'Export to KML/KMZ'
7. Select the dropdown and select just the route directions and select download.
8. Find the Placemark/coordinates within the markup language and copy that section (without the tags) into your latLong.csv

## Appendix II: Manual provisioning steps

Instead of automatic provisioning, an AWS account owner may choose to create a Thing in the AWS management console, obtain credentials, and authenticate the Thing with the credentials. This methods is useful to add an IOT device to an existing account.

More details of this approach is available at:
https://docs.aws.amazon.com/greengrass/v2/developerguide/manual-installation.html.

Step 1
Create an AWS IoT thing to represent your Greengrass Core device

Go to the AWS IoT console
In the navigation pane on the left, choose **Manage**.
Choose **Things**
Under **You don't have any things yet**, choose **Register a thing,** or if your account already has some things, choose **Create**.

Under **Creating AWS IoT things**, choose **Create a single thing**.

On the **Add your device to the thing registry** page, in the **Name** field, enter a name for your thing, such as *J7GGCore*.

When naming things, choose the name carefully, because you can't change a thing name after you create it.

You can skip the **Thing Type** and **Add this thing to a group** steps for now.
Choose **Next**

Under **Add a certificate for your thing**, choose **Create certificate**.
Under **Certificate Created!**, choose each of the Download links to download the following onto your host machine:

- A certificate for this thing
- A public key
- A private key

You also need to download the root CA certificate: RSA 2048 bit key: Amazon Root CA 1
Choose **Activate** to activate the certificate.
Choose **Done** for now. The policy will be attached later.

The certificates and keys will be required later.

Step 2:
Create a Policy

The Greengrass core device needs explicit permissions to access AWS services and resources as well as the operations it can perform on them. Granting permissions is done by creating and attaching an AWS IoT policy that defines the AWS IoT permissions for your Greengrass core device.

Go to the AWS IoT console
In the navigation pane on the left, choose **Secure**.
Choose **Policies**
Choose **Create**
Give the policy a name
Under **Add statements**, choose **Advanced mode**
Paste the following into the template text box, overwriting all the template text

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

```
    {
      "Effect": "Allow",
        "Action": [
          "iot:Publish",
          "iot:Subscribe",
          "iot:Receive",
          "iot:Connect",
          "greengrass:*"
        ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

**NOTE**:
The policy example shown allows access to all MQTT topics and Greengrass operations, so your device works with custom applications and future changes that require new Greengrass operations. It is recommended that you restrict this policy to grant the least privileges required by your use case.

Choose **Create** to create the policy and save it. You will see a confirmation message "Successfully created policy".

Step 3:
Attach the Policy

The policy just created in Create a Policy needs to be attached to the certificate associated with the Greengrass Core thing created in Create an AWS IoT thing to represent your Greengrass Core device.

Go to the AWS IoT console
In the navigation pane on the left, choose **Manage**.
Choose **Things**
In the Search box, enter the name of the Greengrass Core thing
Click on the name of the thing
Choose **Security**
Under **Certificates**, choose the certificate shown
In the **Actions** drop-down menu, choose **Attach policy**
Find and select the policy you created in Create a Policy
Choose **Attach** to attach this policy to the certificate. The permissions in the policy are now attached to the Greengrass Core device.

Step 4:
Retrieve AWS IoT endpoints

Get the AWS IoT data and credentials endpoints for your AWS account, and save them for use later. Your device uses these endpoints to connect to AWS IoT.

**Device Data Endpoint**
Go to the AWS IoT console
In the navigation panel on the left, choose **Settings**
The endpoint can be found under **Device data endpoint**. It will look similar to this: xxxxxxxxxxxxxx-ats.iot.xxxxxxxxx.amazonaws.com

Copy and save this endpoint - it will be used in the configuration file for the installer.

**Credentials Endpoint**
The credentials endpoint can be obtained only through the AWS CLI.

Issue the following command:

```
aws iot describe-endpoint --endpoint-type iot:CredentialProvider
```

The response looks similar to the following example, if the request succeeds.

```
{
  "endpointAddress": "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
}
```

Copy and save this endpoint - it will be used in the configuration file for the installer.

Step 5:
Create a token exchange role and alias

Not all AWS services support certificate based authentication. To authorize calls to such services, the Greengrass core device will use an IAM service role, called the token exchange role. The device uses the AWS IoT credentials provider (through the credentials endpoint) to get temporary AWS credentials for this role.

For more details, refer to Authorizing direct calls to AWS services.

For ease of management, instead of directly using the token exchange role, we will use an AWS IoT role alias for the Greengrass core device. Role aliases enable you to change the token exchange role for a device without having to change the device configuration.

Now create a token exchange IAM role and an AWS IoT role alias that points to that role.

Step 6:
Create the access policy for the token exchange role

First create the access policy required by the token exchange role.

Navigate to the AWS IAM console
In the navigation pane on the left, choose **Policies**
Select **Create policy**
Select the **JSON** tab

Paste the following policy document in the text box, replacing the existing template text:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:DescribeCertificate",
        "logs:CreateLogGroup",
```

```
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

Choose **Next: Tags**
Choose **Next: Review**
Enter a policy **Name** and **Description**
Click on **Create policy** to create and save the policy.

***NOTE*** *– The examples in this document are intended only for dev environments. All devices in your fleet must have credentials with privileges that authorize only intended actions on specific resources. The specific permission policies can vary for your use case. Identify the permission policies that best meet your business and security requirements. For more information, refer to Example policies and Security Best practices*

Step 7:
Create an S3 artifacts access policy

Navigate to the AWS IAM console
In the navigation pane on the left, choose **Policies**
Select **Create policy**
Select the **JSON** tab
Paste the following policy document in the text box, replacing the existing template text:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET-NAME/*"
    }
  ]
}
```

Change DOC-EXAMPLE-BUCKET-NAME to your S3 bucket name.
Choose **Next: Tags**
Choose **Next: Review**
Enter a policy **Name** and **Description**
Click on **Create policy** to create and save the policy.

Step 8:
Create the token exchange IAM role

Navigate to the AWS IAM console
In the navigation pane on the left, choose **Roles**
Select **Create Role**

Choose **AWS Service**, then scroll below to select **IoT**
Under **Select your use case**, choose **IoT**
Choose **Next: Permissions**
Choose **Next: Tags**
Choose **Next: Review**
Enter a role name and description
Choose **Create Role** to create the role.

In the confirmation message at the top of the page - "The role <rolename> has been created", click on the <rolename>.
Choose the **Trust Relationships** tab
Select **Edit trust relationship**

Paste the following policy statement into the text box, overwriting all template text:

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

The above trust policy allows the AWS IoT credentials provider to assume the role on behalf of the device.

Choose **Update Trust Policy** to save the changes.

Select the **Permissions** tab
Choose **Attach Policies**
In the Search field, enter the name of the policy you created in Create the access policy for the token exchange role.
In the search results, select this policy using the checkbox
Choose **Attach policy**

Repeat the above steps for the policy you created in Create an S3 artifacts access policy.

Step 9:
Create the role alias

Create the role alias to be used by the Greengrass Core device.

Navigate to the AWS IoT console
In the navigation pane on the left, select **Secure**
Choose Role Aliases
Select **Create**
Enter a **Name** for the role alias
Under **Select a role to alias**, choose the name of the token exchange role.
Select Create role alias

Step 10:

Create the configuration file

Use a text editor to create a configuration file named config.yaml to provide to the installer.
For example, you can run the following command to use GNU nano to create the config.yaml file.

```
nano GGCoreInstall/config.yaml
```

Copy the following YAML content into the file. This partial configuration file specifies system parameters and Greengrass nucleus parameters.

```
---
system:
  certificateFilePath: "/tmp/certs/device.pem.crt"
  privateKeyPath: "/greengrass/v2/private.pem.key"
  rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
  rootpath: "/greengrass/v2"
  thingName: "J7GGCore"
services:
  aws.greengrass.Nucleus:
    componentType: "NUCLEUS"
    version: "2.2.0"
    configuration:
      awsRegion: "us-west-2"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      iotDataEndpoint: "a302j5u5tb9gbt-ats.iot.us-west-2.amazonaws.com "
      iotCredEndpoint: " c1rlm199xh42f1.credentials.iot.us-west-2.amazonaws.com"
```

NOTE: Update the values in the above file to reflect the configuration of your device, account and region.

Step 11:
Load the certificates and keys on the device

Create the Greengrass root folder on the device. You install the AWS IoT Greengrass Core software to this folder later.

```
sudo mkdir -p /greengrass/v2
```

Copy the downloaded certificates and keys to the device as follows:

- The device certificate : /tmp/certs/device.pem.crt (or other path)
- The private key: /greengrass/v2/private.pem.key
- The root CA certificate: /greengrass/v2/AmazonRootCA1.pem

Step 12:
Install Greengrass V2, similar to the automatic provisioning

Run the installer as shown below.

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
-jar ./GGCoreInstall/lib/Greengrass.jar \
--aws-region region \
--thing-name MyGreengrassCore \
--thing-group-name MyGreengrassCoreGroup \
--tes-role-name GreengrassV2TokenExchangeRole \
--tes-role-alias-name GreengrassCoreTokenExchangeRoleAlias \
--component-default-user ggc_user:ggc_group \
--setup-system-service true \
--deploy-dev-tools true
```

If all goes well, you will see the following output (amongst other messages) on the device console:

```
Successfully configured Nucleus with provisioned resource details!
Configured Nucleus to deploy aws.greengrass.Cli component
```

The installer prints **Successfully set up Nucleus as a system service** if it set up and ran the software as a service

**NOTE**: Pay special attention to installer messages in the console output. Some manual instructions (such as attaching a policy) may be given.

Verify the following:

- The GreengrassV2TokenExchangeRoleAccess policy has been attached to the role GreengrassV2TokenExchangeRole.
- The policy configured in Create an S3 artifacts access policy has been attached to GreengrassV2TokenExchangeRole.

The local development tools (specified by the `--deploy-dev-tools true` option) take some time to deploy. The following command can be used to check the status of this deployment:

```
aws greengrassv2 list-effective-deployments --core-device-thing-name MyGreengrassCore
```

When the status is SUCCEEDED, run the following command to verify that the Greengrass CLI is installed and runs on your device. Replace /greengrass/v2 with the path to the base folder on your device as needed.

```
/greengrass/v2/bin/greengrass-cli help
```

If this does not work, deploy the public component aws.greengrass.Cli from the console. For more details, refer to the online documentation.