

Cloud-based artificial intelligence had to be limited in various solution developments due to the limitations of constant network connectivity. LGE has developed the On-Device artificial intelligence chip, the LG8111, that can provide AI functions even when the network is disconnected.

The LG8111 supports hardware processing of artificial intelligence functions categorized as video, voice, and control intelligence. The LG8111 utilizes LG's unique AI processor to efficiently process the deep learning algorithm (low power, low latency) to achieve "On-Device" artificial intelligence, which escapes the various limitations of always-on network connectivity, which has been pointed out as difficult for artificial intelligence.

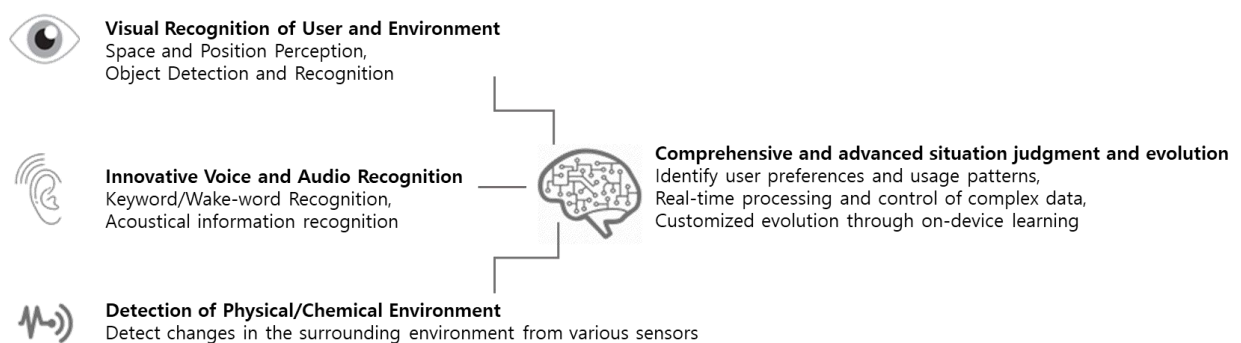


Figure 1. LG8111 Solution Highlights

The AWS IoT Greengrass extends AWS IoT to the device, taking advantage of the cloud, and providing an environment in which data generated through the device can be processed locally. With the On-Device Artificial Intelligence feature mentioned above, the LG8111 can provide improved Sensor Data collection/analysis and ML Inference performance. At the same time, the Eris board, the development board of the LG8111, allows you to integrate the AWS IoT Greengrass more easily for a variety of solutions.

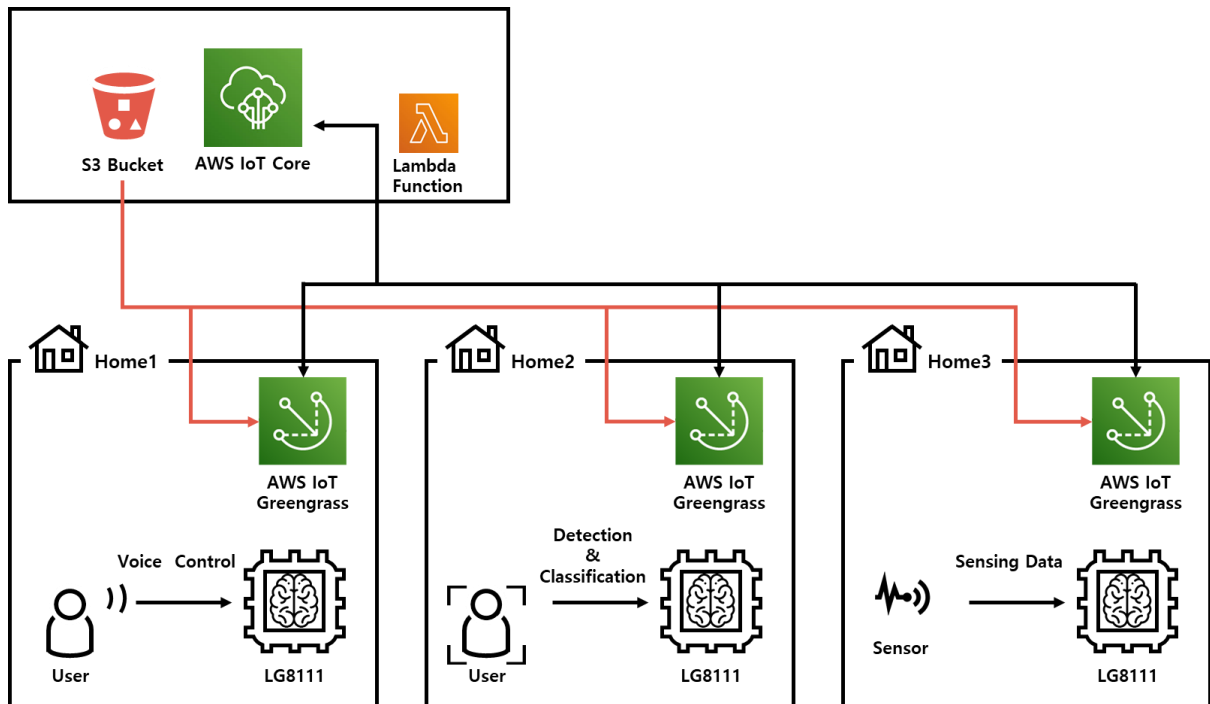


Figure 2. LG8111 and AWS IoT Greengrass

LG8111 Development Board

[LG8111 Development Board Configuration]

The LG8111 development board supports Ubuntu 18.04 environment and consists of Reset, USB Host, UART Debug Port, and Power.

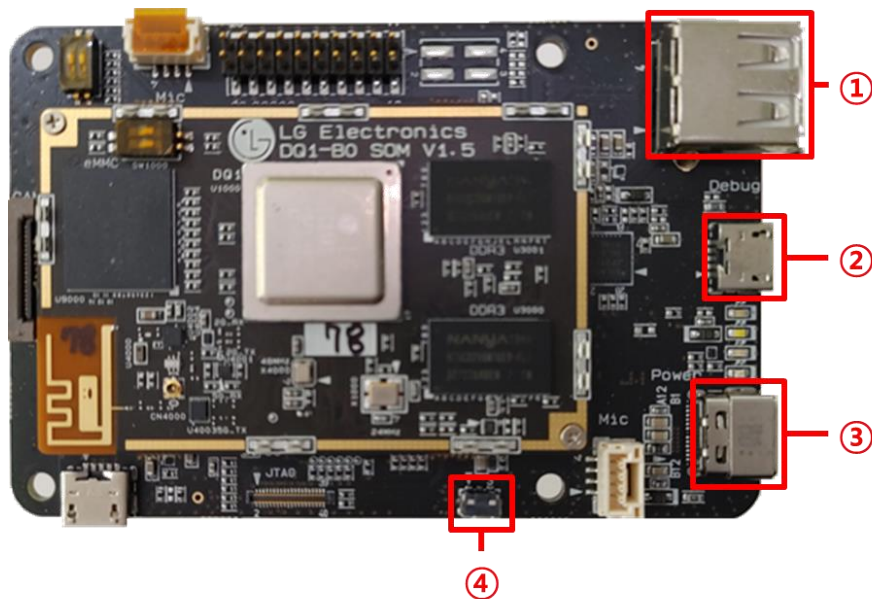


Figure 3. Reference Board(Eris) Embedding LG8111 AI Chip

NUM	REGION NAME	DESCRIPTION
1	USB Host	This is the port used to connect the USB Device.
2	UART Debug Port	Serial Port.
3	Power	Connect the USB Type-C to the Power and turn it on.
4	Reset	Reset the board.

[Board Connect]

Describes how to connect to the LG8111 development board.

1. Connect the Host (PC) and USB wire to the UART Debug Port on the LG8111 Development Board.
2. Grant power to the LG8111 Development Board Power.

The settings required to connect UART to the LG8111 development board are as follows:

RULES	VALUE
BAUD RATE	115200 bps
DATE BITS	8 bit
PARITY SET	N
STOP BITS	1

LG8111 uses AWS Greengrass on its development board

Follow the AWS IoT Greengrass Developer's Guide (https://docs.aws.amazon.com/ko_kr/greengrass/latest/developerguide/setup-filter.rpi.html) for configuring the AWS IoT Greengrass environment on the LG8111 before installing the Greengrass environment settings and the AWS IoT Greengrass core software.

[Obtaining AWS IoT Greengrass Certification]

To run the AWS IoT Greengrass, you must obtain a unique certificate for each device. These unique certificates are separated by group in the AWS IoT Greengrass.

Create a group from the AWS IoT Greengrass start page to obtain a certificate.



Monitor
Onboard
Manage
Greengrass
Groups
Cores
Devices
Secure
Defend
Act
Test

Welcome to AWS IoT Greengrass

AWS IoT Greengrass lets your devices process the data they generate locally, while still taking advantage of AWS services when an internet connection is available.

Get started today with Greengrass on a [Raspberry Pi](#), or read our [system requirements and list of compatible devices](#).

Create a Greengrass Group

Create a Group and provision a Core in one step, or walk through the process step by step.

Create a Group

Set up your Greengrass Group

Setting up your Group requires you to provision a Core device in the IoT Registry, acquire a certificate for your Core, and assign an IAM role to your Group. If you're unfamiliar with any of these steps we recommend the default Group creation. Finally, you'll need to install Greengrass software on your Core device.

Default Group creation (recommended)

This process will automatically provision a Core in the registry, use default settings to generate a new Group, and provide your Core with a new certificate and a key pair.

Use default creation

Advanced Group creation

This customizable process will take you step-by step through the Core provisioning and will allow you to customize the IAM Role for your Group and the certificate for your Core, and provide a key pair.

Customize

Cancel

Use default creation

Enter the name of the group that you want to generate, then select:

SET UP YOUR GREENGRASS GROUP

Name your Group

The Greengrass Group is a cloud-configured managed collection of local devices and Lambda functions that can be programmed to communicate with each other through a Core device. Groups can contain up to 200 local devices.

Group Name

LG8111

Apply tags to the Group (optional) ▾

Cancel

Back

Next

Select the following using the basic configuration settings for the AWS IoT Greengrass group core name:

SET UP YOUR GREENGRASS GROUP

Every Group needs a Core to function

Every Greengrass Group requires a device running Core software. It enables communication between Devices, local Lambda functions, and AWS cloud computing services. Adding information to the Registry is the first step in provisioning a device as your Greengrass Core.

Name

LG8111_Core

Show optional configuration (this can be done later) ▾

Cancel

Back

Next

On the [Run a scripted easy group creation] page, select [Create Group and Core] to verify that the configuration is progressing without problems. If the settings are satisfactory, all settings are green and automatically proceed to the next page.

SET UP YOUR GREENGRASS GROUP

Review Group creation

In order to speed up and simplify Group creation AWS IoT Greengrass will handle the following processes and use default settings. By proceeding to the next step, you are giving permission for us to complete the following steps.

AWS IoT Greengrass will take these actions on your behalf using default settings:

Create a new Greengrass Group in the cloud	Learn more
Provision a new Core in the IoT Registry and add to the Group	Learn more
Generate public and private key set for your Core	Learn more
Generate a new security certificate for the Core using the keys	Learn more
Attach a default security policy to the certificate	Learn more
Enable stream manager on the Core device	Learn more

[Cancel](#)

[Create Group and Core](#)

SET UP YOUR GREENGRASS GROUP

Review Group creation

In order to speed up and simplify Group creation AWS IoT Greengrass will handle the following processes and use default settings. By proceeding to the next step, you are giving permission for us to complete the following steps.

AWS IoT Greengrass will take these actions on your behalf using default settings:

✓ Create a new Greengrass Group in the cloud	Learn more
✓ Provision a new Core in the IoT Registry and add to the Group	Learn more
✓ Generate public and private key set for your Core	Learn more
✓ Generate a new security certificate for the Core using the keys	Learn more
✓ Attach a default security policy to the certificate	Learn more
✓ Enable stream manager on the Core device	Learn more

[Cancel](#)

[Create Group and Core](#)

Upon successful generation of the AWS IoT Greengrass group, you will receive core resources. You will never be able to download the downloaded core resource file again, so you must save and manage it. If you do, you will need to regenerate the core resources.

Connect your Core device


The final steps are to load the Greengrass software and then connect your Core device to the cloud. You can defer connecting your device at this time, but **you must download your public and private keys now as these cannot be retrieved later.**

Download and store your Core's security resources

A certificate for this Core	e892362d7d.cert.pem
A public key	e892362d7d.public.key
A private key	e892362d7d.private.key
Core-specific config file	config.json

Download these resources as a tar.gz

If the AWS IoT Greengrass group is successfully generated, the AWS IoT Greengrass start page displays the generated group in the group window.



- Monitor
- Onboard
- Manage
- Greengrass**
 - Groups**
 - Cores
 - Devices
- Secure
- Defend
- Act
- Test

Greengrass Groups

LG8111

GREENGRASS GROUP

Run AWS IoT Greengrass on the LG8111 Development Board

This is an example of inference of Camera Input data using local resources (Camera, LG AI processor) of the LG8111 development board, and sending the result of the inference to AWS Greengrass to check the result.

You will need to download the AWS IoT Greengrass SDK for python provided by the AWS IoT Greengrass from the Developer's Guide page (https://docs.aws.amazon.com/ko_kr/greengrass/latest/developerguide/what-is-gg.html#g-g-core-sdk-download) for example execution.

This example was implemented in python2.7. Every 5 seconds, images captured by the camera are inferred using an artificial intelligence algorithm called MobileNet, and the inferred results are sent to AWS IoT Greengrass.

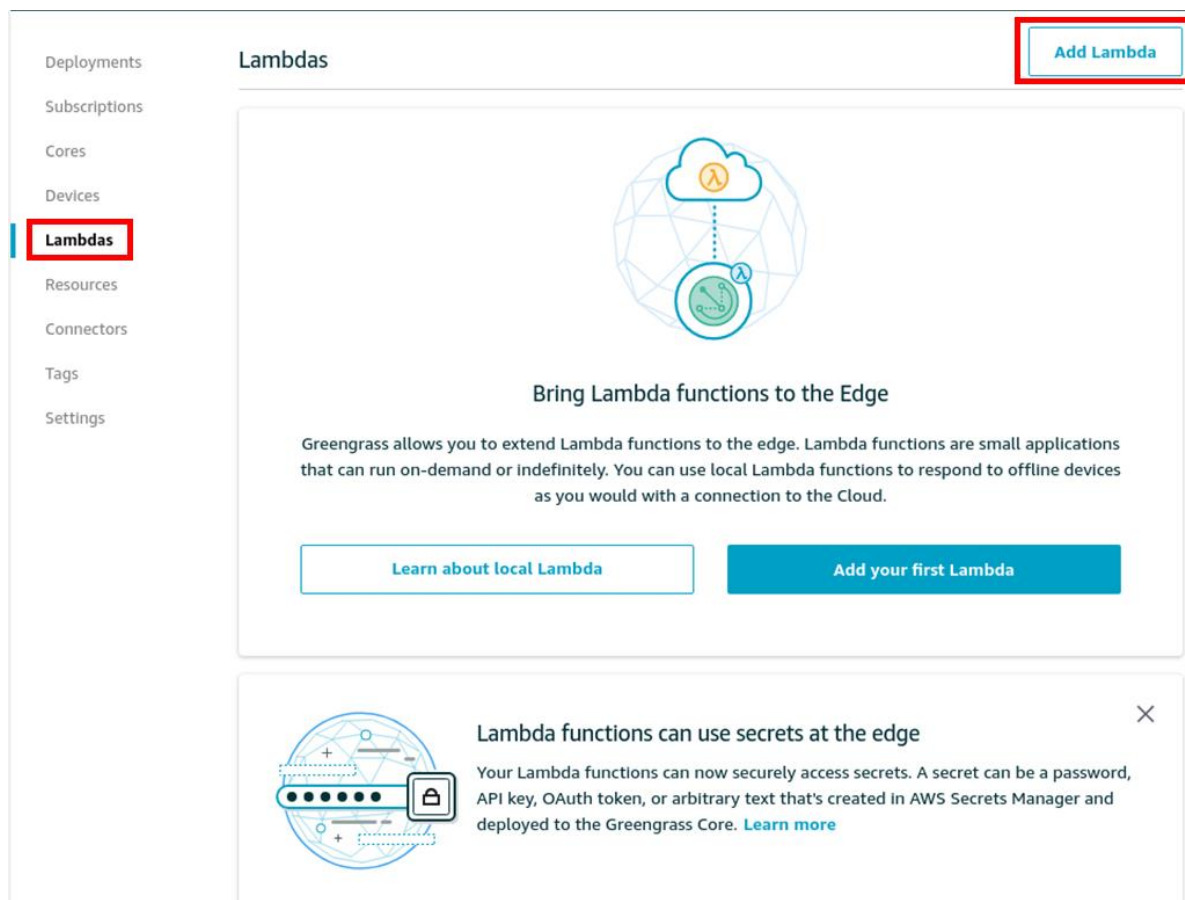
Simply route the contents in the device's "/home/ubuntu/sample/getting_started" folder and the AWS IoT Greengrass SDK for python and compress the same file.

```
$ zip -r greengrassMobilenet.zip labels.txt Mobilenet.lne greengrassMobilenet.py greengrasssdk
```

1. Lambda Registration

Connect with LG8111 development board group registered with AWS IoT Greengrass group.

Register a new Lambda by selecting Lambda -> Add Lambda for the connected group.



Select [Create new Lambda] to add a Lambda function to test the AWS IoT Greengrass on the development board of LG8111.

Add a Lambda to your Greengrass Group

Local Lambdas are hosted on your Greengrass Core and connected to each other and devices by Subscriptions, but they can also be deployed individually to your Group.

Create a new Lambda function

You will be taken to the AWS Lambda Console and can author a new Lambda function.

Create new Lambda

Use an existing Lambda function

You will choose from a list of existing Lambda functions.

Use existing Lambda

Cancel

Back

Use existing Lambda

The Function name is an example of a Python 2.7, registered with a Lambda-distinctive name and created based on python 2.7, so select the Runtime option to python 2.7, then move on to the next stage.

Basic information

Function name

Enter a name that describes the purpose of your function.

GG_Mobilenet

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)

Choose the language to use to write your function.

Python 2.7

The Lambda function that runs AWS IoT Greengrass on the development board of LG8111 is in .zip format, so code entry type is zip and Runtime is python 2.7 to run Lambda on python 2.7.

Handler is "[Lambda file name] [Handler name in Lambda file]" is specified in. Register as "greengrassMobileet.function_handler" because the example code has the file name is "greengrassMobilenet" and the handler name in the example title lambda file is "function_handler".

Finally, press the Upload button to select and register a zip file with the example code compressed.

Function code [Info](#)

Code entry type

Upload a .zip file

Runtime

Python 2.7

Handler [Info](#)

greengrassMobilenet.function_handler

Function package

Upload greengrassMobilenet.zip (13.4 MB)

For files larger than 10 MB, consider uploading using Amazon S3.

Press the save button in the upper right corner of the screen to save your registration, select Publish new version in Actions, and enter a description for the saved version.

The screenshot shows a web interface with a top bar containing buttons: 'Throttle', 'Qualifiers', 'Actions', a dropdown 'Select a test event', 'Test', and 'Save'. The 'Actions' dropdown is open, showing options: 'Publish new version' (highlighted with a red box), 'Create alias', 'Export function', and 'Delete function'. Below this is a modal dialog titled 'Publish new version from \$LATEST' with a close button 'X'. The dialog contains the text: 'Publishing a new version saves a snapshot of the code and configuration of the \$LATEST version. You can't edit the new version's code. Click to confirm.' Below this is a label 'Version description - optional' and a text input field containing 'First Version|' (the text is highlighted with a red box). At the bottom right of the dialog are 'Cancel' and 'Publish' buttons (the 'Publish' button is highlighted with a red box).

Saves the description for the version and saves it with alias and version for that version. (If you select Version with #LATEST, it may not work.)

The screenshot shows the same web interface as above, but the 'Actions' dropdown is open and 'Create alias' is highlighted with a red box. Below the dropdown is a modal dialog with a light green background and a close button 'X'. The dialog is empty.

Create a new alias

×

An alias is a pointer to one or two versions. Choose each version that you want the alias to point to.

Name*

GG_Mobilenet

Description

Version*

\$LATEST

1

Additional version

based on weights (%) that you assign. Click [here](#) to learn more.

Cancel

Create

Select Lambda->Add Lambda and select "Use existing Lambda" to use the saved lambda.

Add a Lambda to your Greengrass Group

Local Lambdas are hosted on your Greengrass Core and connected to each other and devices by Subscriptions, but they can also be deployed individually to your Group.

Create a new Lambda function

You will be taken to the AWS Lambda Console and can author a new Lambda function.

Create new Lambda

Use an existing Lambda function

You will choose from a list of existing Lambda functions.

Use existing Lambda

Cancel

Back

Use existing Lambda

Select Lambda name, Lambda version of Lambda that you registered for testing.

ADD A LAMBDA TO YOUR GREENGRASS GROUP

Use existing Lambda

Select a Lambda

<input type="radio"/>	helloWorld	Python 2.7
<input type="radio"/>	GG_MobileNet	Python 2.7
<input type="radio"/>	cloud9-GGMLInference-GGMLInference-1O39DN17CWN94	Python 3.7
<input checked="" type="radio"/>	GG_Mobilenet	Python 2.7
<input type="radio"/>	GG_MobileNet_label	Python 2.7
<input type="radio"/>	GG_LT_LeNet	Python 2.7
<input type="radio"/>	GGMLWorkshop-BootstrapC9InstanceLambdaFunction-14AH42JZYCOTH	Python 2.7
<input type="radio"/>	GG_Inceptionv3	Python 2.7

Cancel

Back

Next

ADD A LAMBDA TO YOUR GREENGRASS GROUP

Select a Lambda version

Select a Lambda version

<input checked="" type="radio"/>	Alias: GG_Mobilenet
<input type="radio"/>	Version 1

Cancel

Back

Finish

For the test, you must modify the configuration of the Lambda function that you registered. Select Edit configuration by pressing the three circles to the right of the Lambda function added for this purpose.

GREENGRASS GROUP

DQ1_4

● Successfully completed

Actions ▾

Deployments

Subscriptions

Cores

Devices

Lambdas

Resources

Connectors

Tags

Settings

Lambdas


Add Lambda

GG_Mobilenet
LAMBDA FUNCTION

USING

Edit configuration

Remove function



Lambda functions can use secrets at the edge

Your Lambda functions can now securely access secrets. A secret can be a password, API key, OAuth token, or arbitrary text that's created in AWS Secrets Manager and deployed to the Greengrass Core. [Learn more](#)

Correct the Memory limit, Timeout, and Lambda lifetime during the Lambda configuration.

Memory limit sets the memory limit to 128 MB when running the Lambda function.

Timeout is the Lambda function operation wait time, set to 25 seconds because Test Lambda transfers results every 5 seconds.

Lambda lifecycle saves the changes made by setting it to "Make this function log-live and keep it running indefinitely" running in the lifecycle setting of the Lambda function so that it can always operate for Test.

GG_Mobilenet

[View function in AWS Lambda](#)

Alias GG_Mobilenet

[Remove version](#)

Run as [?](#)

☒ Use group default (currently: ggc_user/ggc_group)

☐ Another user ID/group ID

Containerization [?](#)

☒ Use group default (currently: Greengrass container)

☐ Greengrass container (always)

☐ No container (always)

Memory limit

128

MB

Timeout

25

Second

Lambda lifecycle

☐ On-demand function

☒ Make this function long-lived and keep it running indefinitely

Read access to /sys directory

☒ Disable

☐ Enable

Input payload data type

☒ JSON

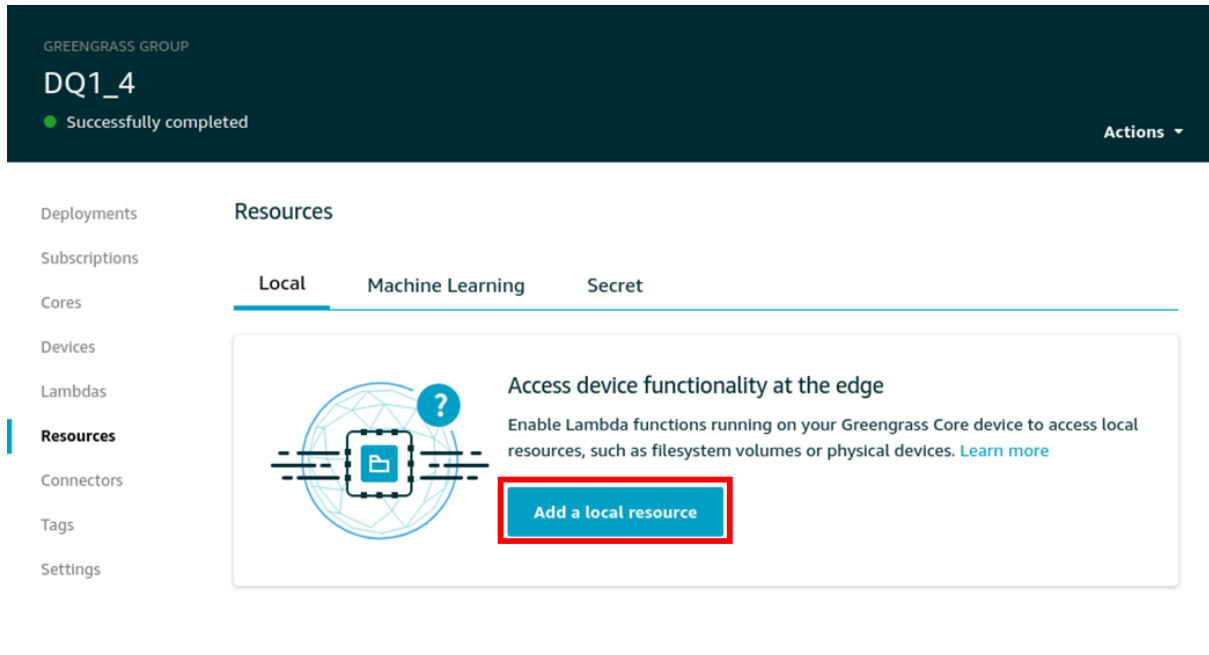
2. Local Resource Registration

Register the local resources (ex. camera, sensor, LG AI processors, etc.) of the Device in the development board of LG8111, which is used to run the AWS IoT Greengrass. The registered local resource is the local resource used by the Lambda function to operate.

NAME	DEVICE PATH	DESCRIPTION
------	-------------	-------------

LG AI PROCESSOR	/dev/dq1_lne	LG-specific AI processor that efficiently processes deep learning algorithms (power saving, low latency)
CAMERA	/dev/video0	LG8111 Development Device의 Camera

Select the Local tab for the Resource to add local resource.



2-1. Added LG AI Processor Local Resource

Configure the device path and lambda function to use the local resource in order to add local resource for the LG AI processor. At this time, set the setting for Lambda function to read and write access.

Add a new local resource

Resource name

LNE_Driver

Resource type

- ☒ Device
- ☐ Volume

Device path

/dev/dq1_lne

Group owner file access permission

An AWS IoT Greengrass Lambda function process normally runs without an OS Group. However, you can give additional file access permissions to the Lambda function process.

- ☐ No OS group
- ☒ Automatically add OS group permissions of the Linux group that owns the resource
- ☐ Specify another OS group to add permission

Lambda function affiliations

Resources must be affiliated with a Lambda function before deployment

Done

Find

GG_Mobilenet

Add a new local resource

Resource name

LNE_Driver

Resource type

☒ Device

☐ Volume

Device path

/dev/dq1_lne

Group owner file access permission

An AWS IoT Greengrass Lambda function process normally runs without an OS Group. However, you can give additional file access permissions to the Lambda function process.

☐ No OS group

☒ Automatically add OS group permissions of the Linux group that owns the resource

☐ Specify another OS group to add permission

Lambda function affiliations

GG_Mobilenet

READ AND WRITE ACCESS

Done

Specify the permission this Lambda will have to the resource.

☐ Read-only access

☒ Read and write access

After the local resource registration of the LG AI processor is completed, the status changes to green when it is successfully connected to Lambda function.

GREENGRASS GROUP

DQ1_4

● Successfully completed

Actions ▾

Deployments

Subscriptions

Cores

Devices

Lambdas

Resources

Connectors

Tags

Settings

Resources

LocalMachine LearningSecret

Add local resource

Name	Resource Type ▾	Status	Local path ▾
LNE_Driver	Device	● Affiliated	/dev/dq1_lne

2-2. Added Camera Local Resource

In order to use the LG8111 development board's Camera, the same process will be used to change only the Device path and register the LG AI processor as local resource.

Add a new local resource

Resource name

Resource type

☒ Device

☐ Volume

Device path

Group owner file access permission

An AWS IoT Greengrass Lambda function process normally runs without an OS Group. However, you can give additional file access permissions to the Lambda function process.

☐ No OS group

☒ Automatically add OS group permissions of the Linux group that owns the resource

☐ Specify another OS group to add permission

Lambda function affiliations

Resources must be affiliated with a Lambda function before deployment [Done](#)

Add a new local resource

Resource name

Camera_Driver

Resource type

- ☒ Device
☐ Volume

Device path

/dev/video0

Group owner file access permission

An AWS IoT Greengrass Lambda function process normally runs without an OS Group. However, you can give additional file access permissions to the Lambda function process.

- ☐ No OS group
☒ Automatically add OS group permissions of the Linux group that owns the resource
☐ Specify another OS group to add permission

Lambda function affiliations

GG_Mobilenet

READ AND WRITE ACCESS

Done

Specify the permission this Lambda will have to the resource.

- ☐ Read-only access
☒ Read and write access

GREENGRASS GROUP

DQ1_4

● Successfully completed

Actions ▾

Deployments

Subscriptions

Cores

Devices

Lambdas

Resources

Connectors

Tags

Settings

Resources

Local

Machine Learning

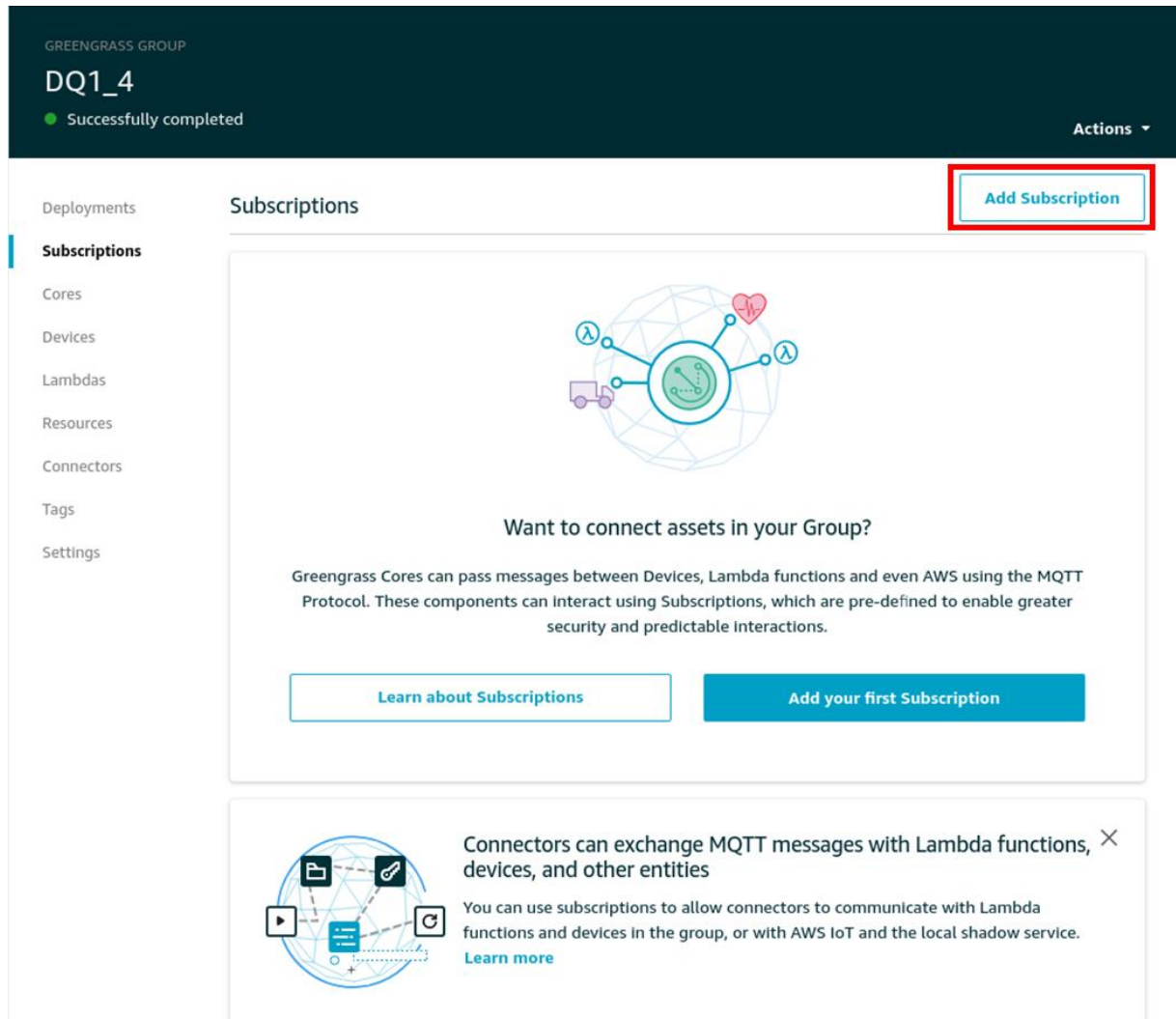
Secret

Add local resource

Name	Resource Type ▾	Status	Local path ▾	
Camera_Driver	Device	● Affiliated	/dev/video0	...
LNE_Driver	Device	● Affiliated	/dev/dq1_lne	...

2-3. Subscription & Lambda Deploy

Before deploying Lambda, you must configure which Topic to subscribe to. Select and configure Add Subscriptions in Subscriptions.



In the Subscription, source is the name of the Lambda that you registered, and target is the AWS IoT Greengrass, so select IoT Cloud and enter the Topic name. In this test, we set the topic name to "tflite/lne", so we set the topic to "tflite/lne".

greengrassMobilenet.py

```
..
85         client.publish(topic="tflite/lne", queueFullPolicy="AllOrException", w
86             payload="Result: {}".format(labels[lne_answer]))
...
```


Select your source and target

A Subscription consists of a source, target, and topic. The source is the originator of the message. The target is the destination of the message. The first step is selecting your source and target.

Select a source

No objects selected [Close](#)


Services Devices **Lambdas** Connectors


 GG_Mobilenet

Select a target

No objects selected [Close](#)

Services Devices Lambdas Connectors

 IoT Cloud

 Local Shadow Service

CREATE A SUBSCRIPTION

Filter your data with a topic

The source publishes data to the target. Topic filters are used to limit or control the data that the target receives. If a topic filter isn't defined, all messages from the source are sent to the target.

Source



GG_Mobilenet

LAMBDA

Topic filter

[How do I enter a topic filter?](#)

tf/ite/line

Target



IoT Cloud

SERVICE

Back

Next

AWS IoT Greengrass Daemon must be running on the development board of the LG8111 before Deploying the Lambda function. Run the AWS IoT Greengrass Daemon with the "greengrass start" command at the "/greengrass/gcc/core" location during installation, and the AWS IoT Greengrass Daemon with root privileges is located at the "/greengrass/gcc/core" location.

```
root@DQ1:/greengrass/gcc/core# pwd
/greengrass/gcc/core
root@DQ1:/greengrass/gcc/core# ./greengrassd start
Setting up greengrass daemon
Validating hardlink/softlink protection
Waiting for up to 1m10s for Daemon to start

Greengrass successfully started with PID: 2106
root@DQ1:/greengrass/gcc/core#
```

Run AWS IoT Greengrass Daemon on the development board of the LG8111, press the Actions button in the upper right corner of the screen in the AWS IoT Greengrass console, select Deploy, and send the saved Lambda to the development board of the LG8111. When Deploy is in progress, the circle on the left side of the screen changes from gray to yellow to green, and each signifies preparation, transfer in progress, and deployment is complete.

GREENGRASS GROUP

DQ1_4

● Successfully completed

Actions ▾

- Deploy
- Delete Group
- Reset Deployments

Deployments

Subscriptions

Cores

Devices

Lambdas

Resources

Connectors

Tags

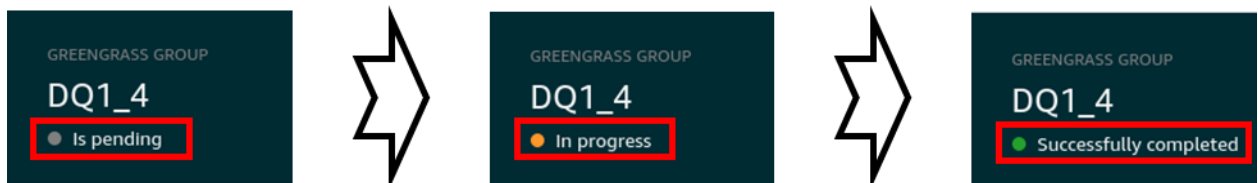
Settings

Source	Target	Topic
GG_Mobilenet	IoT Cloud	tflite/lne

Connectors can exchange MQTT messages with Lambda functions, devices, and other entities

You can use subscriptions to allow connectors to communicate with Lambda functions and devices in the group, or with AWS IoT and the local shadow service.

[Learn more](#)



2-4. AWS IoT Greengrass Test

Use the test provided by the AWS IoT Greengrass to verify that the Lambda function deployed from the AWS IoT Greengrass works properly on the LG8111 development board.

What you set up in Test is the part that sets up the format in which MYTT, the method of exchanging messages with the topic name that you are subscribing to, and the AWS IoT Greengrass.

The topic used in the example is "tflite/lne". Since the LG8111 Development Board will forward the result value as a string, set it as a string.

Once every five seconds when the configuration is complete, you can perform an inference through Mobilenet on the images received by the camera and see the result value sent to the AWS IoT Greengrass.



Monitor

Onboard

Manage

Greengrass

Secure

Defend

Act

Test

MQTT client [?](#)

Connected as **iotconsole-1582446297420-0**

Subscriptions

[Subscribe to a topic](#)

[Publish to a topic](#)

Subscribe
Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.

Subscription topic

tfllite/lne

[Subscribe to topic](#)

Max message capture [?](#)

100

Quality of Service [?](#)

- ☒ 0 - This client will not acknowledge to the Device Gateway that messages are received
☐ 1 - This client will acknowledge to the Device Gateway that messages are received

MQTT payload display

- ☐ Auto-format JSON payloads (improves readability)
☒ Display payloads as strings (more accurate)
☐ Display raw payloads (in hexadecimal)

MQTT client [?](#)

Connected as **iotconsole-1582446297420-0**

Subscriptions

[Subscribe to a topic](#)

[Publish to a topic](#)

tfllite/lne

x

Publish
Specify a topic and a message to publish with a QoS of 0.

tfllite/lne

[Publish to topic](#)

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

tfllite/lne

Feb 23, 2020 5:33:48 PM +0900

[Export](#) [Hide](#)

Result: water jug

tfllite/lne

Feb 23, 2020 5:33:42 PM +0900

[Export](#) [Hide](#)

Result: cartoon

tfllite/lne

Feb 23, 2020 5:33:36 PM +0900

[Export](#) [Hide](#)

Result: hand-held computer, hand-held microcomputer

AWS IoT Greengrass Operational References

Deploy completed while AWS IoT Greengrass is running, but you may need to verify that it works for debugging purposes.

At this time, the method used for debugging is as follows.

1. Error occurred during distribution

- /greengrass/ggc/var/log/system/runtime.log
- Log with errors seen during greengrass operation

2. Distributed but AWS IoT Greengrass Does Not Transfer Result Values

- /greengrass/ggc/var/log/user/[Region]/[public key]/[lambda function].log
- You can see the area by pressing the area displayed at the top of the AWS IoT Greengrass screen.

