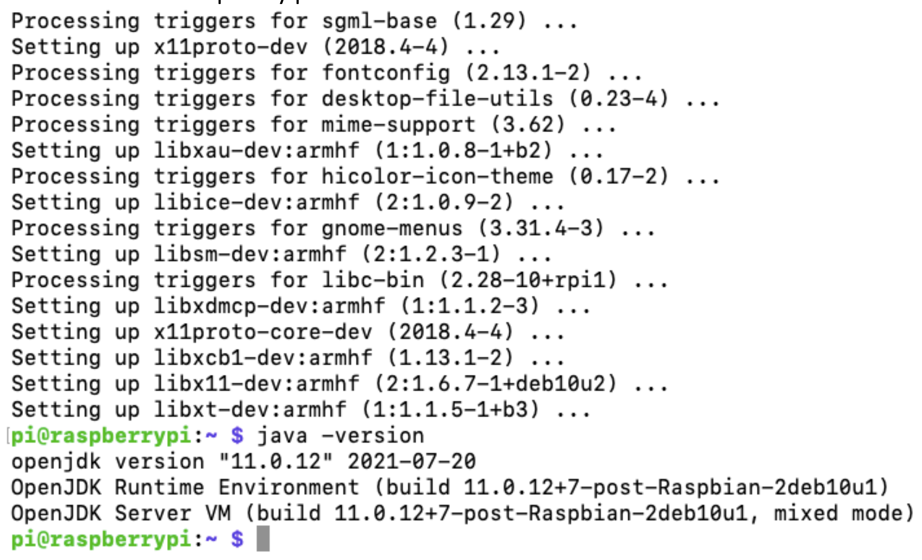
Homework -3

Sayali Kudale

## Part 1 : Configure IoT core and IoT Client

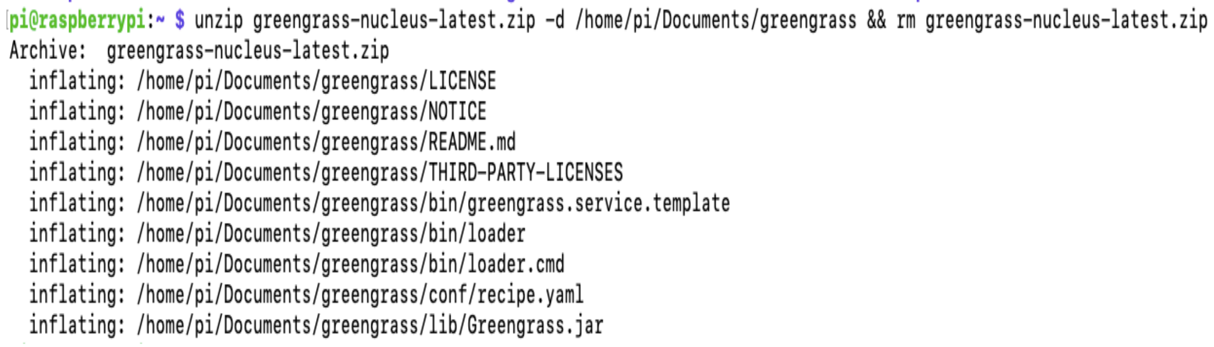
### Configuration of Core device on AWS green-grass version 2:

1. Selected Raspberry pi device as the core device to install aws IoT green-grass.
2. Enabled SSH on the Raspberry pi device to remotely connect from laptop.
3. Installed the Java SDK on raspberry pi.



```
Processing triggers for sgml-base (1.29) ...
Setting up x11proto-dev (2018.4-4) ...
Processing triggers for fontconfig (2.13.1-2) ...
Processing triggers for desktop-file-utils (0.23-4) ...
Processing triggers for mime-support (3.62) ...
Setting up libxau-dev:armhf (1:1.0.8-1+b2) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Setting up libice-dev:armhf (2:1.0.9-2) ...
Processing triggers for gnome-menus (3.31.4-3) ...
Setting up libsm-dev:armhf (2:1.2.3-1) ...
Processing triggers for libc-bin (2.28-10+rpi1) ...
Setting up libxdmcp-dev:armhf (1:1.1.2-3) ...
Setting up x11proto-core-dev (2018.4-4) ...
Setting up libxcb1-dev:armhf (1.13.1-2) ...
Setting up libx11-dev:armhf (2:1.6.7-1+deb10u2) ...
Setting up libxt-dev:armhf (1:1.1.5-1+b3) ...
pi@raspberrypi:~ $ java -version
openjdk version "11.0.12" 2021-07-20
OpenJDK Runtime Environment (build 11.0.12+7-post-Raspbian-2deb10u1)
OpenJDK Server VM (build 11.0.12+7-post-Raspbian-2deb10u1, mixed mode)
pi@raspberrypi:~ $
```

4. Installation and configuration of the **AWS IoT Greengrass nucleus version 2.5.**
   a. Download and unzip the greengrass-nucleus-latest.zip file.

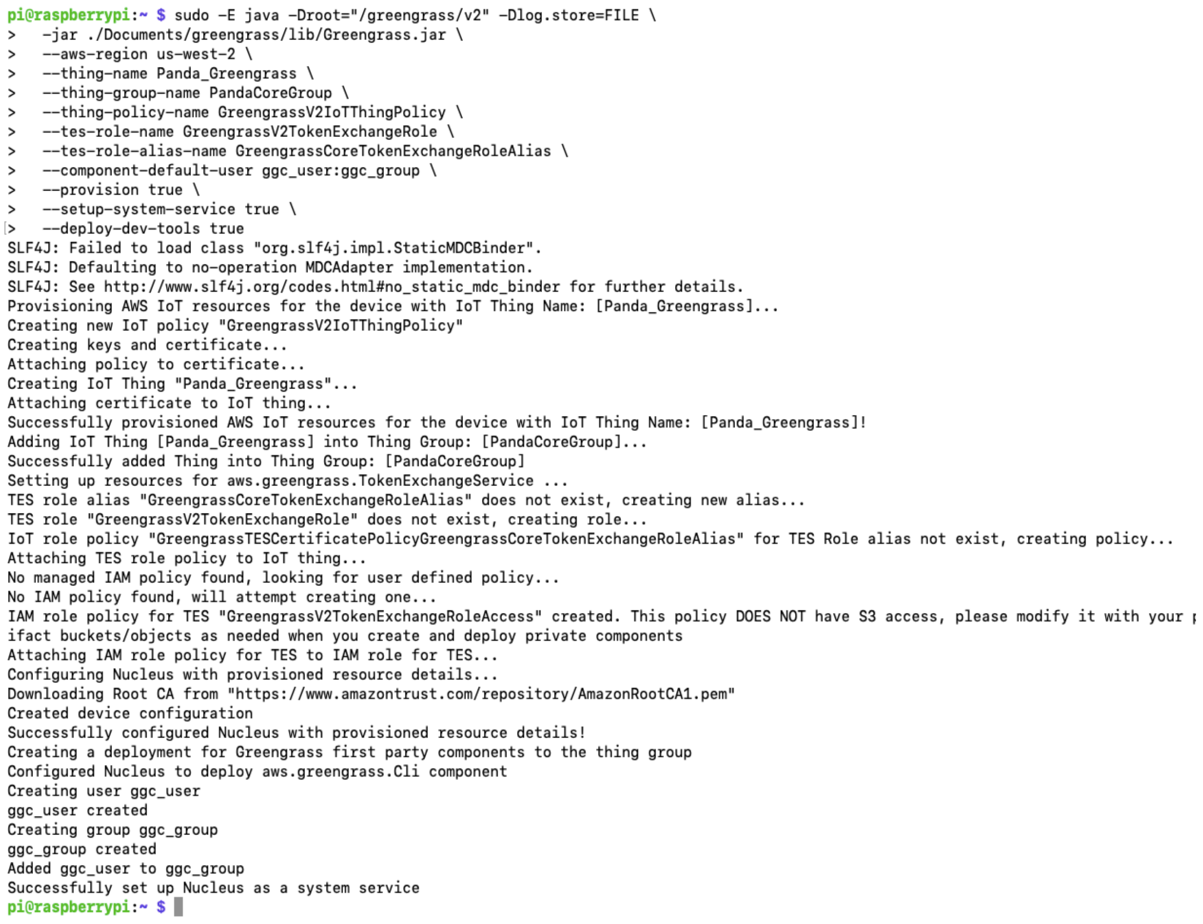


```
pi@raspberrypi:~ $ unzip greengrass-nucleus-latest.zip -d /home/pi/Documents/greengrass && rm greengrass-nucleus-latest.zip
Archive:  greengrass-nucleus-latest.zip
  inflating: /home/pi/Documents/greengrass/LICENSE
  inflating: /home/pi/Documents/greengrass/NOTICE
  inflating: /home/pi/Documents/greengrass/README.md
  inflating: /home/pi/Documents/greengrass/THIRD-PARTY-LICENSES
  inflating: /home/pi/Documents/greengrass/bin/greengrass.service.template
  inflating: /home/pi/Documents/greengrass/bin/loader
  inflating: /home/pi/Documents/greengrass/bin/loader.cmd
  inflating: /home/pi/Documents/greengrass/conf/recipe.yaml
  inflating: /home/pi/Documents/greengrass/lib/Greengrass.jar
```

   b. Provided AWS credentials so that the installer can provision the AWS IoT and IAM resources for core device by exporting them in path.

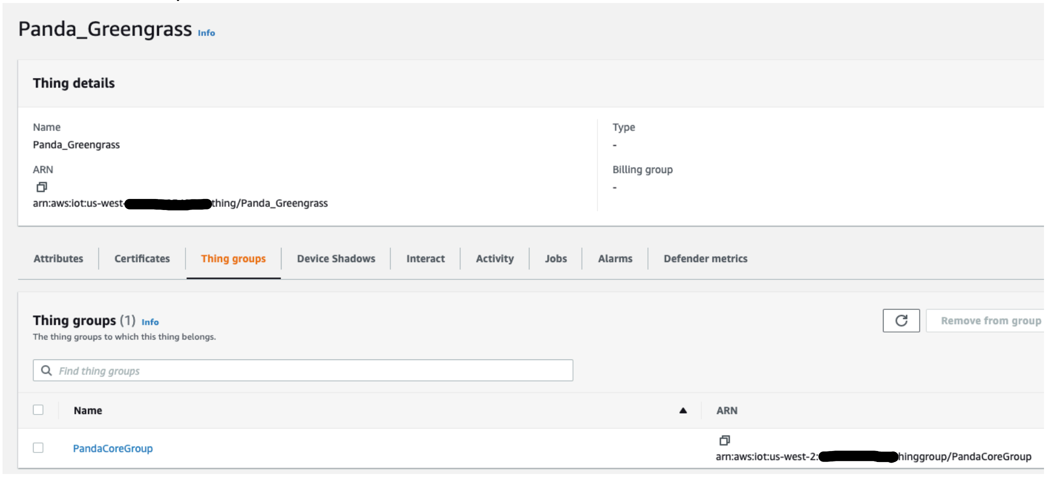
      pi@raspberrypi:~ $ export AWS_ACCESS_KEY_ID=A************O
      pi@raspberrypi:~$export AWS_SECRET_ACCESS_KEY=sn************n

   c. Executed below command to launch the installation of the AWS IoT Greengrass core software installer. This command automatically created new thing "PandaGreenGrass" and thing group "PandaCoreGroup", Also, role "GreengrassV2TokenExchangeRole", new role thing policy "GreengrassV2IoTThingPolicy" and new role "GreengrassV2TokenExchangeRole"
      With this installation, we are installing the software as a system service.



```
pi@raspberrypi:~ $ sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
>   -jar ./Documents/greengrass/lib/Greengrass.jar \
>   --aws-region us-west-2 \
>   --thing-name Panda_Greengrass \
>   --thing-group-name PandaCoreGroup \
>   --thing-policy-name GreengrassV2IoTThingPolicy \
>   --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
SLF4J: Failed to load class "org.slf4j.impl.StaticMDCBinder".
SLF4J: Defaulting to no-operation MDCAdapter implementation.
SLF4J: See http://www.slf4j.org/codes.html#no_static_mdc_binder for further details.
Provisioning AWS IoT resources for the device with IoT Thing Name: [Panda_Greengrass]...
Creating new IoT policy "GreengrassV2IoTThingPolicy"
Creating keys and certificate...
Attaching policy to certificate...
Creating IoT Thing "Panda_Greengrass"...
Attaching certificate to IoT thing...
Successfully provisioned AWS IoT resources for the device with IoT Thing Name: [Panda_Greengrass]!
Adding IoT Thing [Panda_Greengrass] into Thing Group: [PandaCoreGroup]...
Successfully added Thing into Thing Group: [PandaCoreGroup]
Setting up resources for aws.greengrass.TokenExchangeService ...
TES role alias "GreengrassCoreTokenExchangeRoleAlias" does not exist, creating new alias...
TES role "GreengrassV2TokenExchangeRole" does not exist, creating role...
IoT role policy "GreengrassTESCertificatePolicyGreengrassCoreTokenExchangeRoleAlias" for TES Role alias not exist, creating policy...
Attaching TES role policy to IoT thing...
No managed IAM policy found, looking for user defined policy...
No IAM policy found, will attempt creating one...
IAM role policy for TES "GreengrassV2TokenExchangeRoleAccess" created. This policy DOES NOT have S3 access, please modify it with your 
ifact buckets/objects as needed when you create and deploy private components
Attaching IAM role policy for TES to IAM role for TES...
Configuring Nucleus with provisioned resource details...
Downloading Root CA from "https://www.amazontrust.com/repository/AmazonRootCA1.pem"
Created device configuration
Successfully configured Nucleus with provisioned resource details!
Creating a deployment for Greengrass first party components to the thing group
Configured Nucleus to deploy aws.greengrass.Cli component
Creating user ggc_user
ggc_user created
Creating group ggc_group
ggc_group created
Added ggc_user to ggc_group
Successfully set up Nucleus as a system service
pi@raspberrypi:~ $
```

d. Check the deployment status of newly created thing on aws cloud:
We can see the Panda_Greengrass thing is deployed in the cloud and associated with the PandaCoreGroup.



Panda_Greengrass Info

**Thing details**

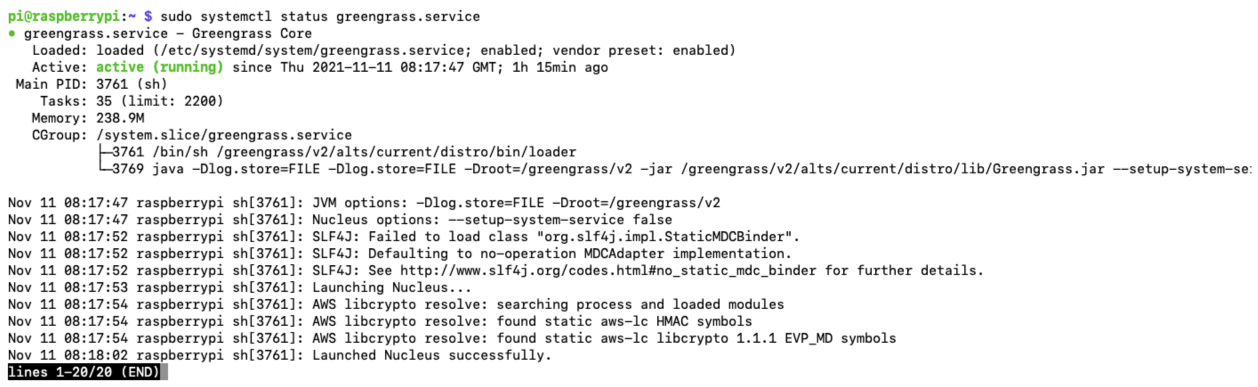
| | |
|---|---|
| Name | Type |
| Panda_Greengrass | - |
| ARN | Billing group |
| arn:aws:iot:us-west[redacted]thing/Panda_Greengrass | - |

**Thing groups** (1) Info

The thing groups to which this thing belongs.

| Name | ARN |
|---|---|
| PandaCoreGroup | arn:aws:iot:us-west-2:[redacted]hinggroup/PandaCoreGroup |

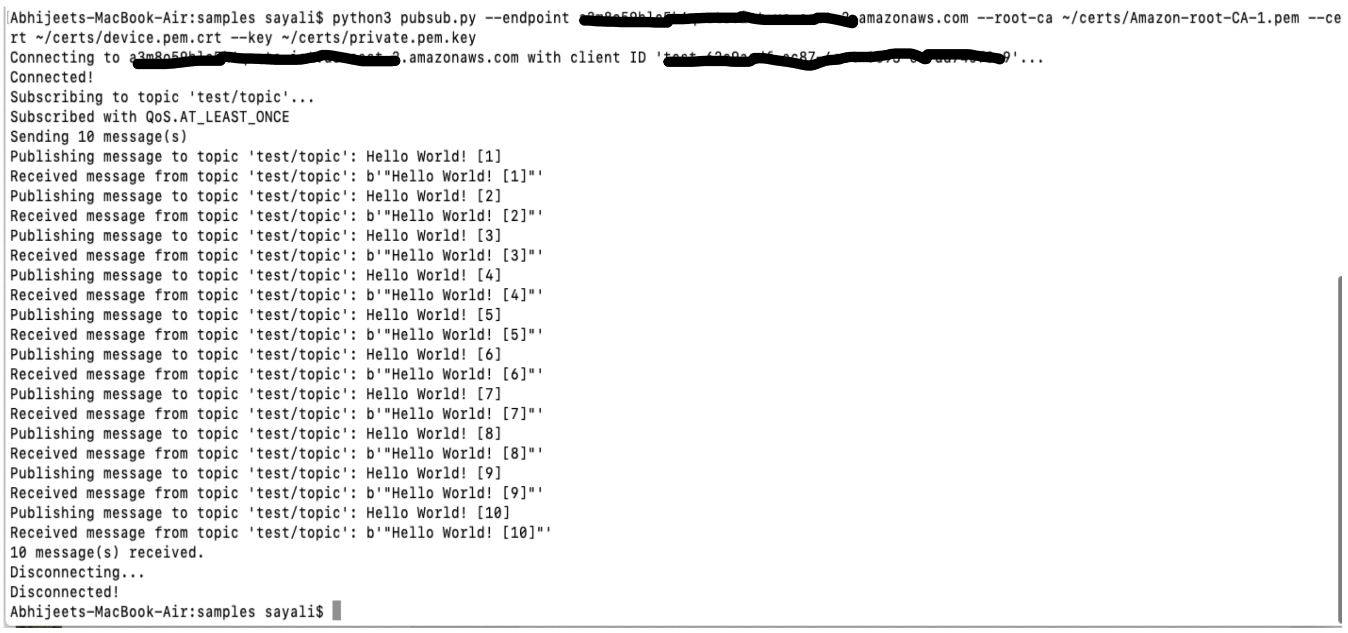
e. Greengrass service status is active and running in Raspberry Pi core device.



```
pi@raspberrypi:~ $ sudo systemctl status greengrass.service
● greengrass.service - Greengrass Core
   Loaded: loaded (/etc/systemd/system/greengrass.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2021-11-11 08:17:47 GMT; 1h 15min ago
 Main PID: 3761 (sh)
    Tasks: 35 (limit: 2200)
   Memory: 238.9M
   CGroup: /system.slice/greengrass.service
           ├─3761 /bin/sh /greengrass/v2/alts/current/distro/bin/loader
           └─3769 java -Dlog.store=FILE -Dlog.store=FILE -Droot=/greengrass/v2 -jar /greengrass/v2/alts/current/distro/lib/Greengrass.jar --setup-system-se

Nov 11 08:17:47 raspberrypi sh[3761]: JVM options: -Dlog.store=FILE -Droot=/greengrass/v2
Nov 11 08:17:47 raspberrypi sh[3761]: Nucleus options: --setup-system-service false
Nov 11 08:17:52 raspberrypi sh[3761]: SLF4J: Failed to load class "org.slf4j.impl.StaticMDCBinder".
Nov 11 08:17:52 raspberrypi sh[3761]: SLF4J: Defaulting to no-operation MDCAdapter implementation.
Nov 11 08:17:52 raspberrypi sh[3761]: SLF4J: See http://www.slf4j.org/codes.html#no_static_mdc_binder for further details.
Nov 11 08:17:53 raspberrypi sh[3761]: Launching Nucleus...
Nov 11 08:17:54 raspberrypi sh[3761]: AWS libcrypto resolve: searching process and loaded modules
Nov 11 08:17:54 raspberrypi sh[3761]: AWS libcrypto resolve: found static aws-lc HMAC symbols
Nov 11 08:17:54 raspberrypi sh[3761]: AWS libcrypto resolve: found static aws-lc libcrypto 1.1.1 EVP_MD symbols
Nov 11 08:18:02 raspberrypi sh[3761]: Launched Nucleus successfully.
lines 1-20/20 (END)
```

## Configuration of the Client device:

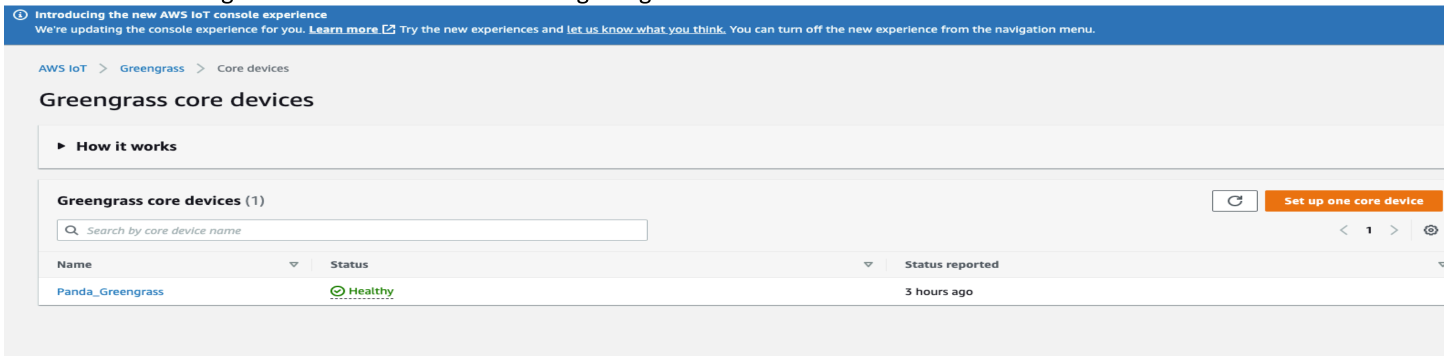
1. Selected laptop as the client device to install aws IoT SDK.
2. Cloned aws-iot-device-sdk-python-v2 repository.
3. If we run the sample application, we are able to publish and receive messages from aws iot console.
4. Set up the policy and run the sample application.



```
Abhijeets-MacBook-Air:samples sayali$ python3 pubsub.py --endpoint [redacted]amazonaws.com --root-ca ~/certs/Amazon-root-CA-1.pem --ce
rt ~/certs/device.pem.crt --key ~/certs/private.pem.key
Connecting to [redacted].amazonaws.com with client ID '[redacted]'...
Connected!
Subscribing to topic 'test/topic'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 10 message(s)
Publishing message to topic 'test/topic': Hello World! [1]
Received message from topic 'test/topic': b'"Hello World! [1]"'
Publishing message to topic 'test/topic': Hello World! [2]
Received message from topic 'test/topic': b'"Hello World! [2]"'
Publishing message to topic 'test/topic': Hello World! [3]
Received message from topic 'test/topic': b'"Hello World! [3]"'
Publishing message to topic 'test/topic': Hello World! [4]
Received message from topic 'test/topic': b'"Hello World! [4]"'
Publishing message to topic 'test/topic': Hello World! [5]
Received message from topic 'test/topic': b'"Hello World! [5]"'
Publishing message to topic 'test/topic': Hello World! [6]
Received message from topic 'test/topic': b'"Hello World! [6]"'
Publishing message to topic 'test/topic': Hello World! [7]
Received message from topic 'test/topic': b'"Hello World! [7]"'
Publishing message to topic 'test/topic': Hello World! [8]
Received message from topic 'test/topic': b'"Hello World! [8]"'
Publishing message to topic 'test/topic': Hello World! [9]
Received message from topic 'test/topic': b'"Hello World! [9]"'
Publishing message to topic 'test/topic': Hello World! [10]
Received message from topic 'test/topic': b'"Hello World! [10]"'
10 message(s) received.
Disconnecting...
Disconnected!
Abhijeets-MacBook-Air:samples sayali$
```

## <u>Part 2 : Register devices and AWS Greengrass group</u>

1. Registered raspberry pi as a core device. With the help of greengrass command in part 1. Raspberry pi device is registered as a core device in AWS greengrass.



Introducing the new AWS IoT console experience
We're updating the console experience for you. Learn more Try the new experiences and let us know what you think. You can turn off the new experience from the navigation menu.

AWS IoT > Greengrass > Core devices

Greengrass core devices

▸ How it works

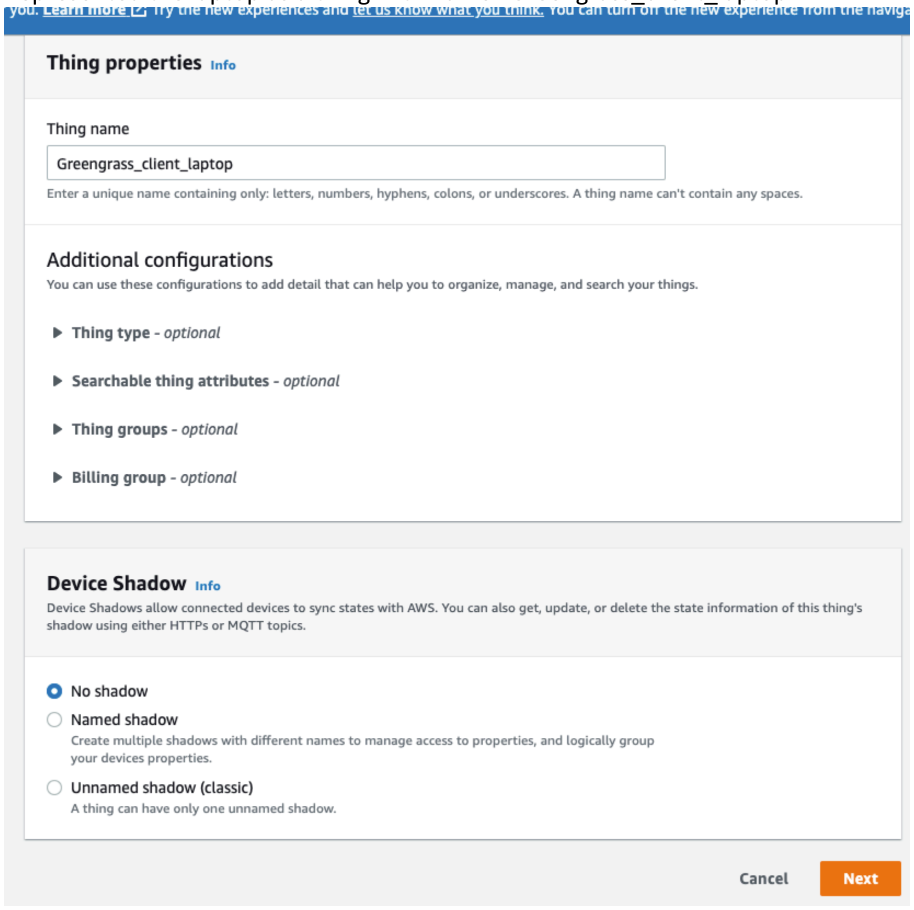
Greengrass core devices (1)

Set up one core device

Search by core device name

| Name | Status | Status reported |
|---|---|---|
| Panda_Greengrass | Healthy | 3 hours ago |

2. Registered laptop as a client device in AWS green-grass.
    a. Used the configured discovery as an option to connect the client devices (i.e laptop) to core devices.
    b. Represented the laptop as a thing with name “Greengrass_client_laptop”.



you. Learn more Try the new experiences and let us know what you think. You can turn off the new experience from the naviga

**Thing properties** Info

Thing name

Greengrass_client_laptop

Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

Additional configurations

You can use these configurations to add detail that can help you to organize, manage, and search your things.

▶ Thing type - *optional*

▶ Searchable thing attributes - *optional*

▶ Thing groups - *optional*

▶ Billing group - *optional*

**Device Shadow** Info

Device Shadows allow connected devices to sync states with AWS. You can also get, update, or delete the state information of this thing's shadow using either HTTPs or MQTT topics.

[✓ No shadow]

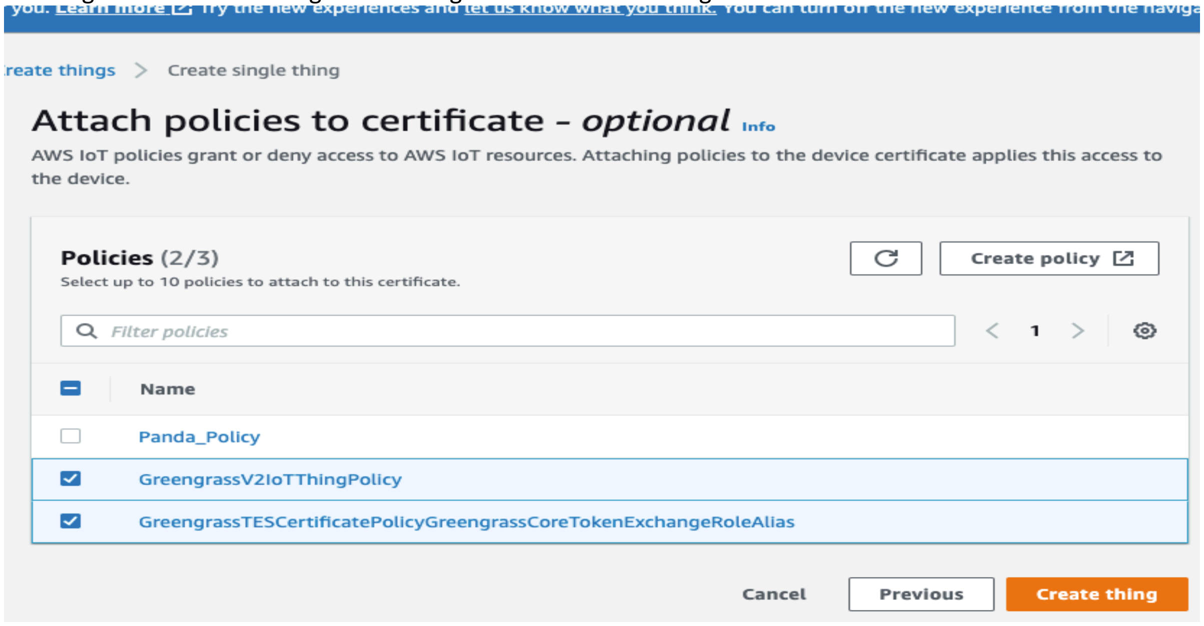
Named shadow

Create multiple shadows with different names to manage access to properties, and logically group your devices properties.

Unnamed shadow (classic)

A thing can have only one unnamed shadow.

    c. Attached the policies which are created in part 1 “GreengrassV2IoTThingPolicy” and “GreengrassV2TokenExchangeRoleGreengrassCoreTokenExchangeRoleAlias”.



you. Learn more Try the new experiences and let us know what you think. You can turn off the new experience from the naviga

reate things > Create single thing

**Attach policies to certificate - *optional*** Info

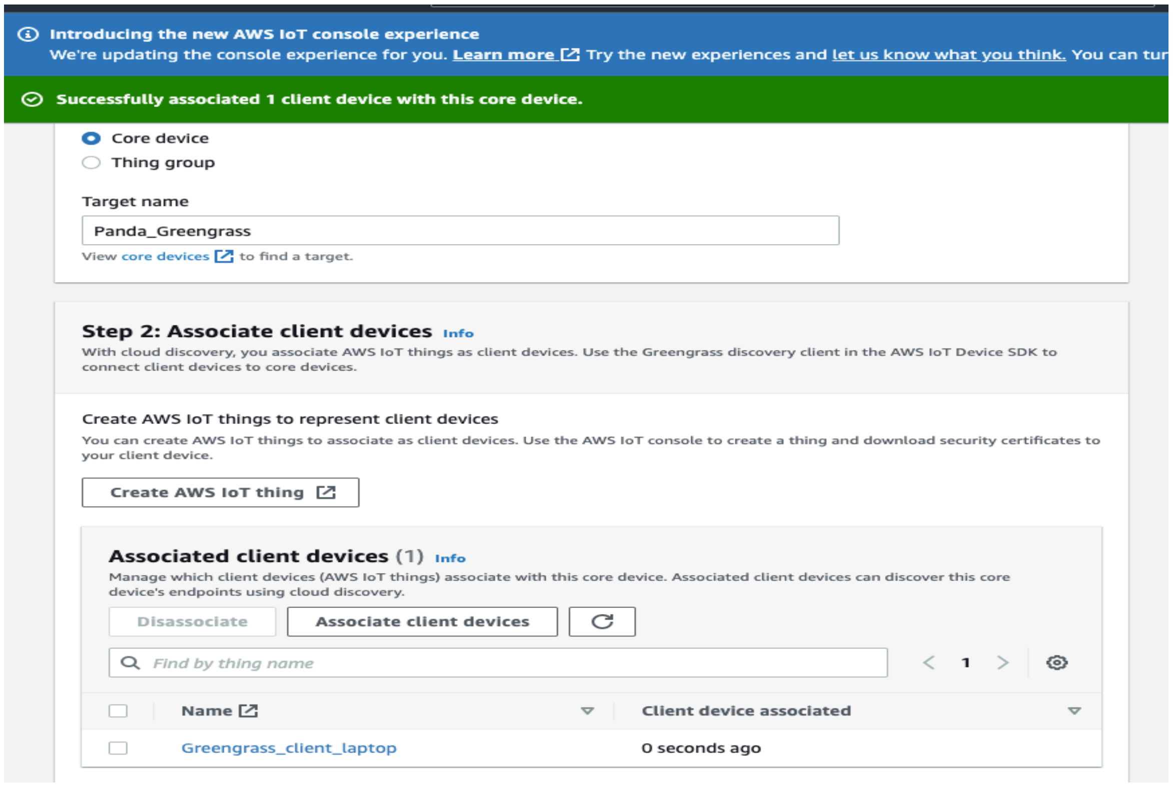
AWS IoT policies grant or deny access to AWS IoT resources. Attaching policies to the device certificate applies this access to the device.

**Policies** (2/3)

Select up to 10 policies to attach to this certificate.

| | Name |
|---|---|
| ☐ | Panda_Policy |
| ✓ | GreengrassV2IoTThingPolicy |
| ✓ | GreengrassTESCertificatePolicyGreengrassCoreTokenExchangeRoleAlias |

d. Associate the client device to core device.



Introducing the new AWS IoT console experience
We're updating the console experience for you. Learn more Try the new experiences and let us know what you think. You can tur

Successfully associated 1 client device with this core device.

Core device [✓]
Thing group

Target name
Panda_Greengrass
View core devices to find a target.

**Step 2: Associate client devices** Info
With cloud discovery, you associate AWS IoT things as client devices. Use the Greengrass discovery client in the AWS IoT Device SDK to connect client devices to core devices.

Create AWS IoT things to represent client devices
You can create AWS IoT things to associate as client devices. Use the AWS IoT console to create a thing and download security certificates to your client device.

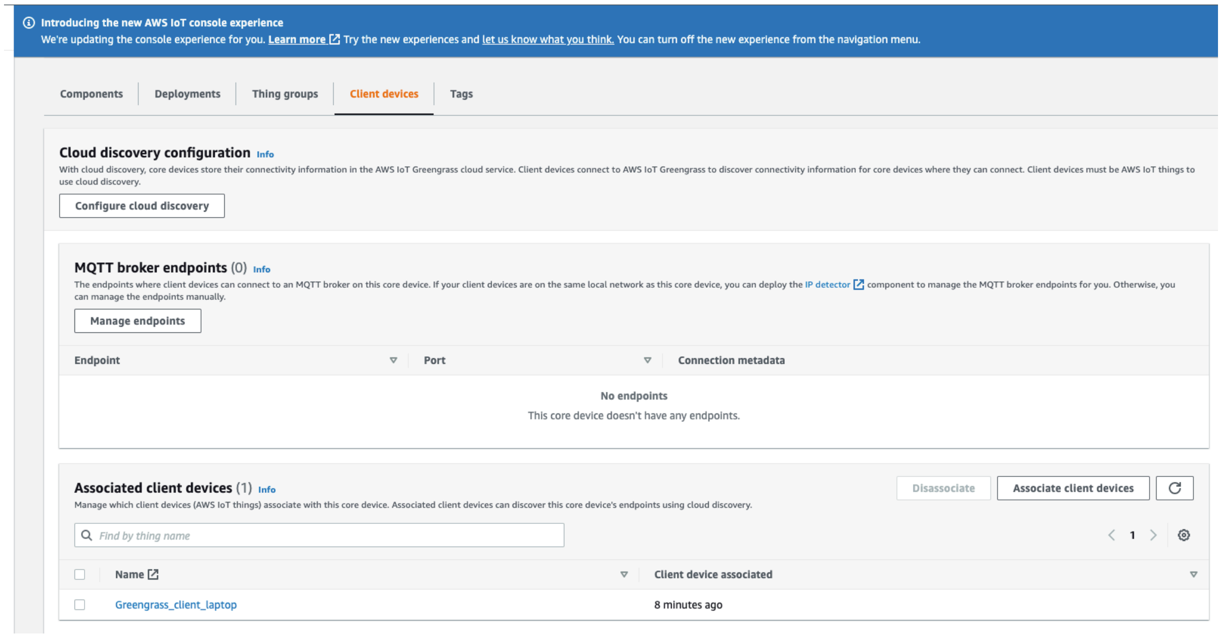
Create AWS IoT thing

**Associated client devices** (1) Info
Manage which client devices (AWS IoT things) associate with this core device. Associated client devices can discover this core device's endpoints using cloud discovery.

Disassociate | Associate client devices

Find by thing name

| Name | Client device associated |
|---|---|
| Greengrass_client_laptop | 0 seconds ago |



Introducing the new AWS IoT console experience
We're updating the console experience for you. Learn more Try the new experiences and let us know what you think. You can turn off the new experience from the navigation menu.

**Cloud discovery configuration** Info
With cloud discovery, core devices store their connectivity information in the AWS IoT Greengrass cloud service. Client devices connect to AWS IoT Greengrass to discover connectivity information for core devices where they can connect. Client devices must be AWS IoT things to use cloud discovery.

Configure cloud discovery

**MQTT broker endpoints** (0) Info
The endpoints where client devices can connect to an MQTT broker on this core device. If your client devices are on the same local network as this core device, you can deploy the IP detector component to manage the MQTT broker endpoints for you. Otherwise, you can manage the endpoints manually.

Manage endpoints

| Endpoint | Port | Connection metadata |
|---|---|---|

No endpoints
This core device doesn't have any endpoints.

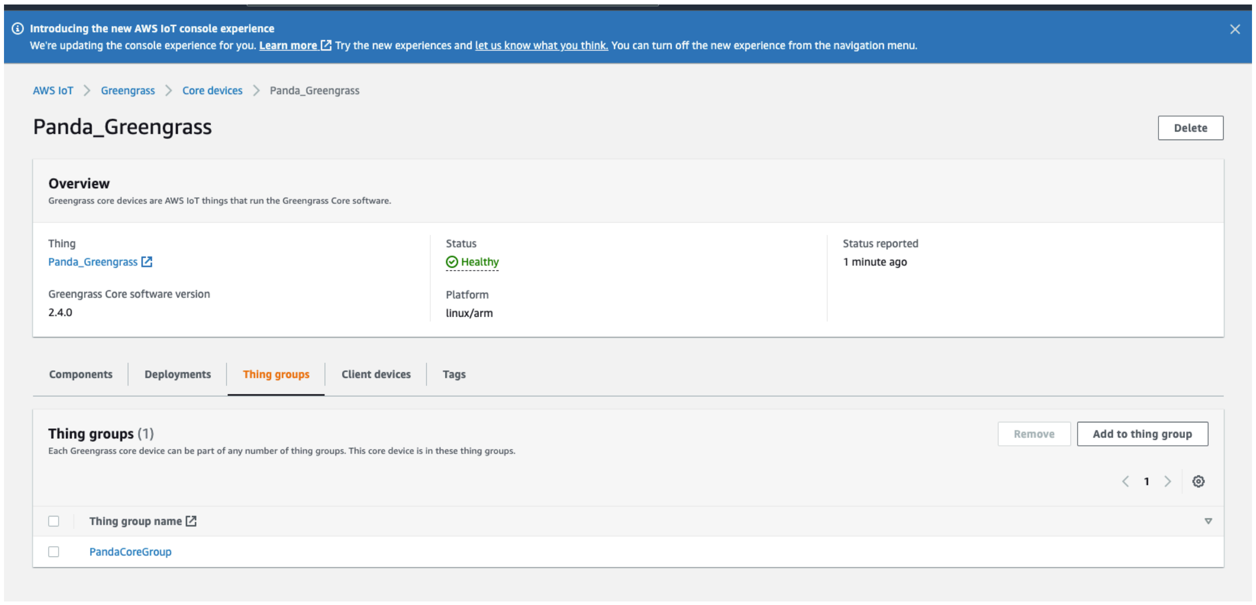
**Associated client devices** (1) Info
Manage which client devices (AWS IoT things) associate with this core device. Associated client devices can discover this core device's endpoints using cloud discovery.

Disassociate | Associate client devices

Find by thing name

| Name | Client device associated |
|---|---|
| Greengrass_client_laptop | 8 minutes ago |

3. AWS greengrass group created in part 1 which is represented by thing groups "PandaCoreGroup". The client device "Greengrass_client_laptop" is also part of this thing group.


Introducing the new AWS IoT console experience
We're updating the console experience for you. Learn more Try the new experiences and let us know what you think. You can turn off the new experience from the navigation menu.

AWS IoT > Greengrass > Core devices > Panda_Greengrass

Panda_Greengrass

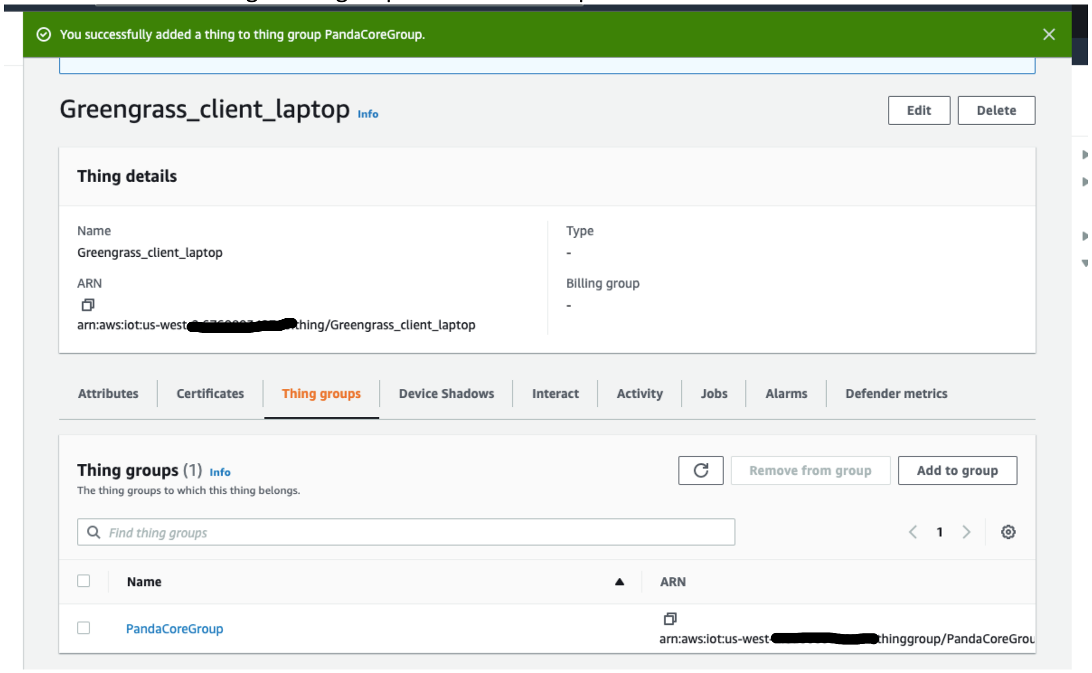
Overview
Greengrass core devices are AWS IoT things that run the Greengrass Core software.

Thing: Panda_Greengrass
Status: Healthy
Status reported: 1 minute ago
Greengrass Core software version: 2.4.0
Platform: linux/arm

Thing groups (1)
Each Greengrass core device can be part of any number of thing groups. This core device is in these thing groups.

| Thing group name |
|---|
| PandaCoreGroup |

4. Added the client thing to the group PandaCoreGroup.


You successfully added a thing to thing group PandaCoreGroup.

Greengrass_client_laptop

Thing details

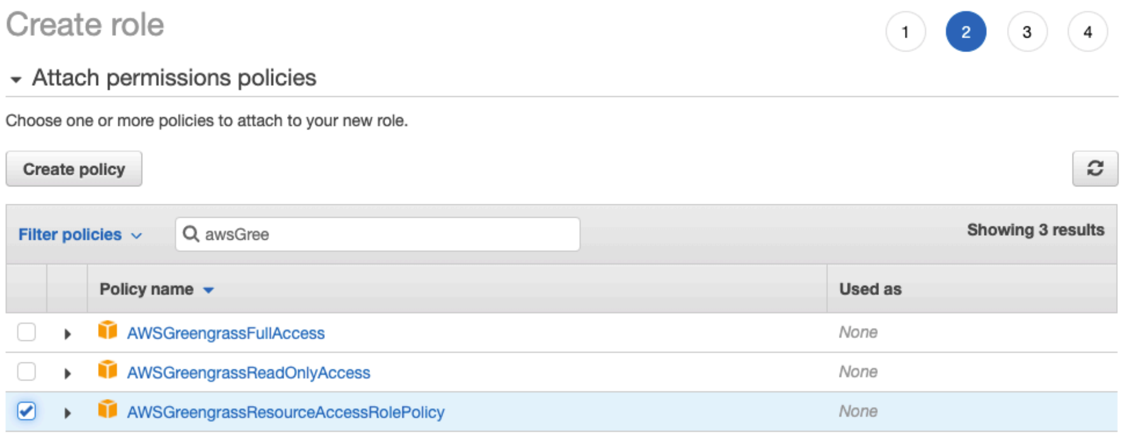
Name: Greengrass_client_laptop
Type: -
ARN: arn:aws:iot:us-west[redacted]thing/Greengrass_client_laptop
Billing group: -

Thing groups (1)
The thing groups to which this thing belongs.

| Name | ARN |
|---|---|
| PandaCoreGroup | arn:aws:iot:us-west[redacted]thinggroup/PandaCoreGrou |

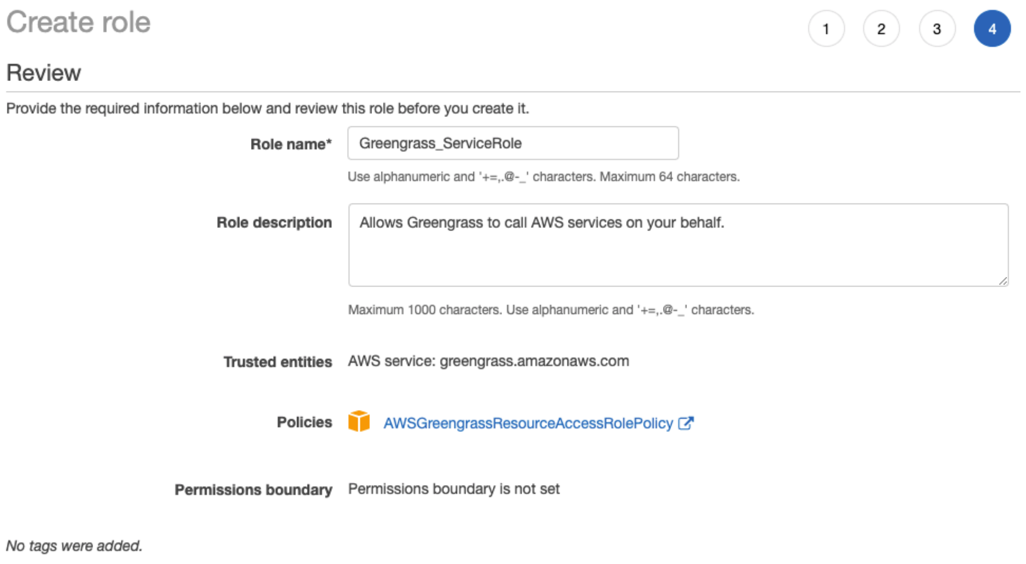
## Part 3 : Communication between core and client device

1. Created Greeengrass service role.
   a. Created a new greengrass service role “Greengrass_ServiceRole” with policy “AWSGreengrassResourceAccessRolePolicy”.



Choose one or more policies to attach to your new role.

| | | Policy name | Used as |
|---|---|---|---|
| ☐ | ▸ | AWSGreengrassFullAccess | *None* |
| ☐ | ▸ | AWSGreengrassReadOnlyAccess | *None* |
| ✓ | ▸ | AWSGreengrassResourceAccessRolePolicy | *None* |

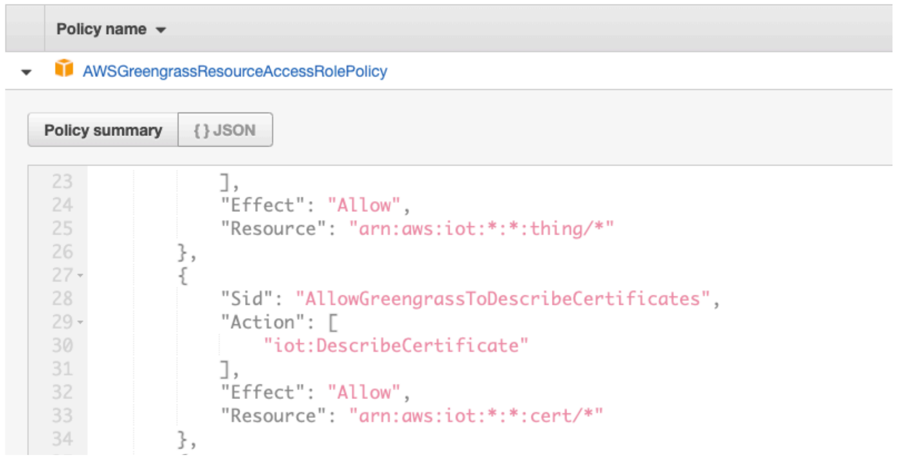


Provide the required information below and review this role before you create it.

| | |
|---|---|
| **Role name*** | Greengrass_ServiceRole |
| | Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters. |
| **Role description** | Allows Greengrass to call AWS services on your behalf. |
| | Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters. |
| **Trusted entities** | AWS service: greengrass.amazonaws.com |
| **Policies** | AWSGreengrassResourceAccessRolePolicy |
| **Permissions boundary** | Permissions boundary is not set |

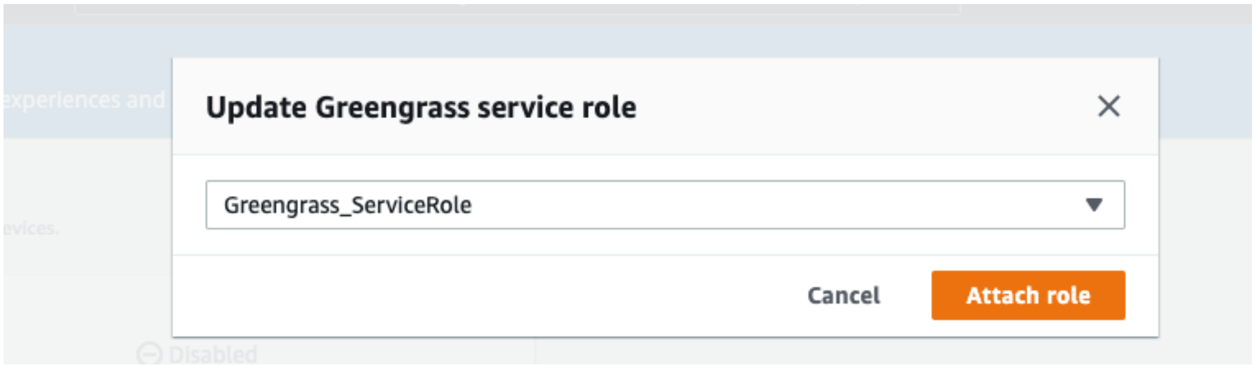
*No tags were added.*

b. This AWSGreengrassResourceAccessRolePolicy has the iot:DescribeCertificate permission.



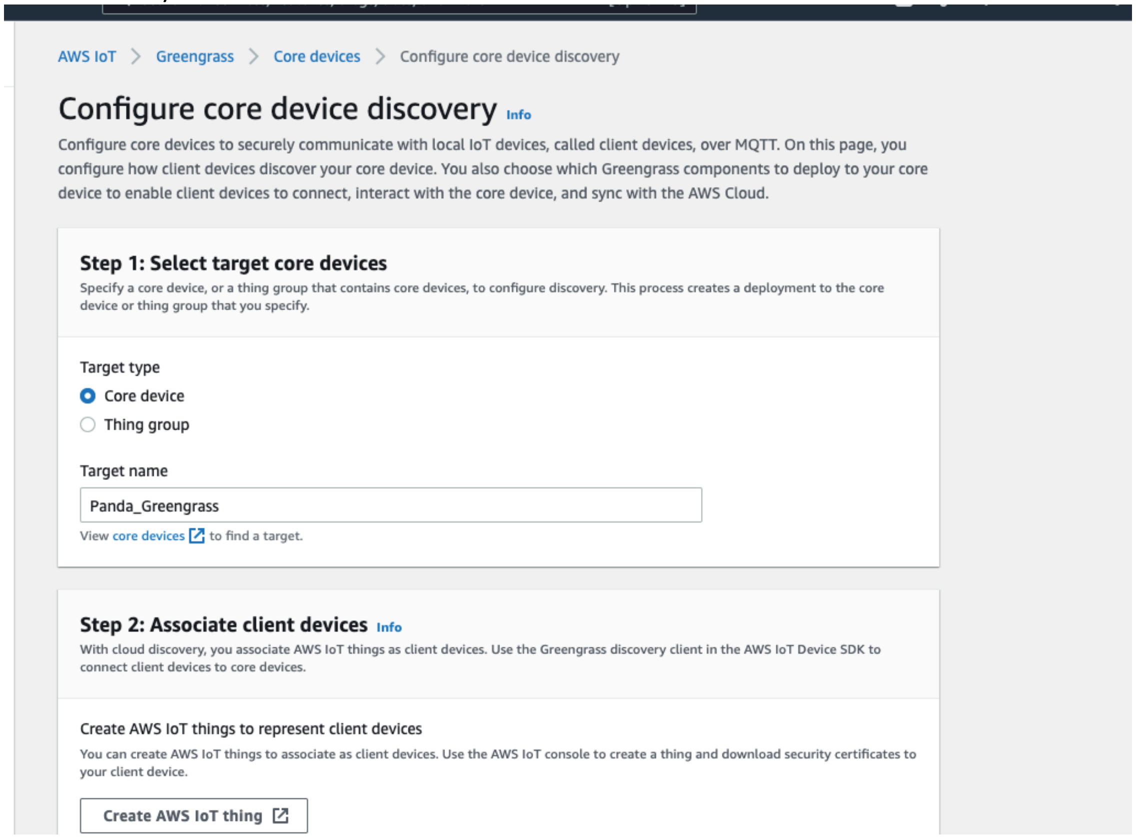
```
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:*:*:thing/*"
        },
        {
            "Sid": "AllowGreengrassToDescribeCertificates",
            "Action": [
                "iot:DescribeCertificate"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:*:*:cert/*"
        },
```

c. Attached this Greengrass_ServiceRole to the AWS account by updating in the greengrass serive role section.


Update Greengrass service role

Greengrass_ServiceRole

2. Reviewed the AWS IoT policies of client device. It has all the greengrass policies because permission was set to *.
3. Enable Client Device support.
   a. On the core devices page, we have selected the "Panda_Greengrass" which is the core device we have created in part 1. And in the client devices page we have selected the option to configure cloud discovery.


AWS IoT > Greengrass > Core devices > Configure core device discovery

Configure core device discovery Info

Configure core devices to securely communicate with local IoT devices, called client devices, over MQTT. On this page, you configure how client devices discover your core device. You also choose which Greengrass components to deploy to your core device to enable client devices to connect, interact with the core device, and sync with the AWS Cloud.

Step 1: Select target core devices

Specify a core device, or a thing group that contains core devices, to configure discovery. This process creates a deployment to the core device or thing group that you specify.

Target type

[✓ Core device]
Thing group

Target name

Panda_Greengrass

View core devices to find a target.

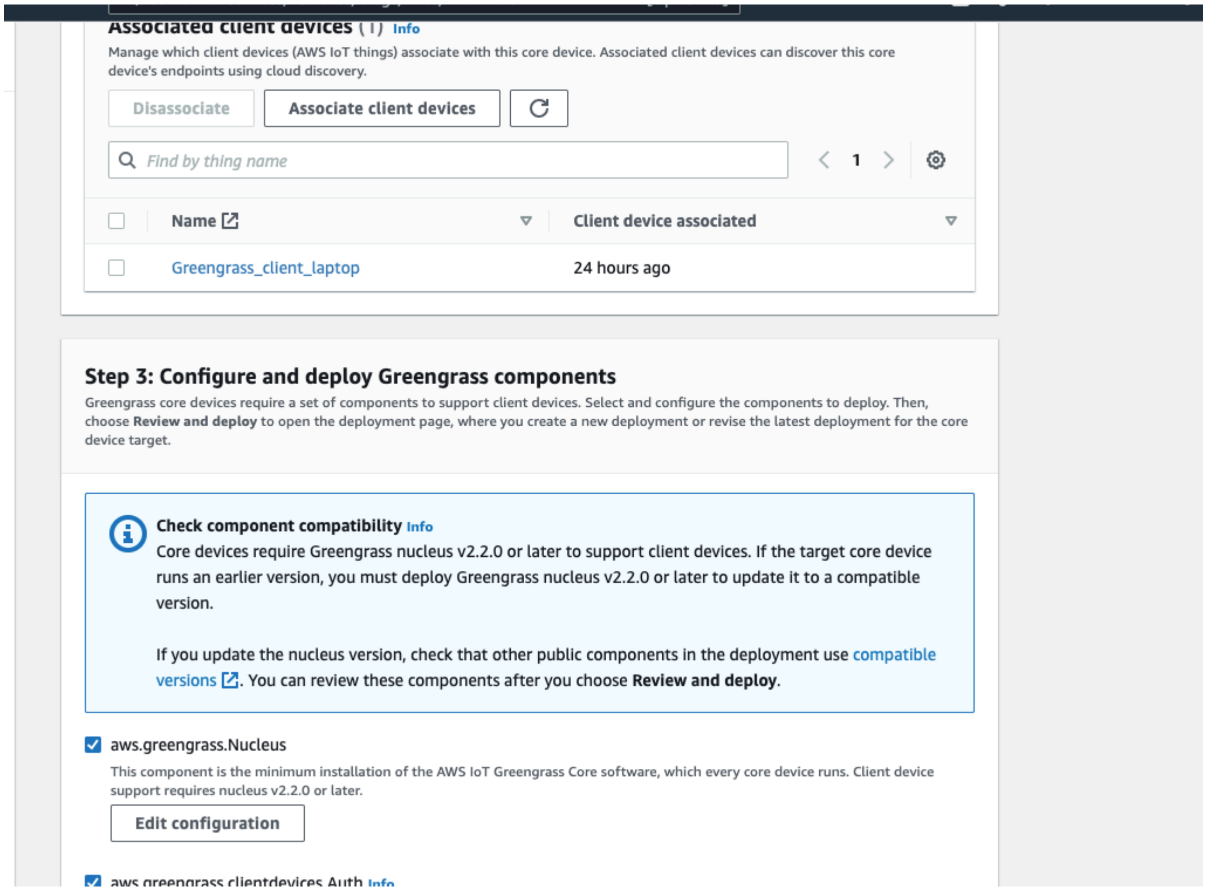
Step 2: Associate client devices Info

With cloud discovery, you associate AWS IoT things as client devices. Use the Greengrass discovery client in the AWS IoT Device SDK to connect client devices to core devices.

Create AWS IoT things to represent client devices

You can create AWS IoT things to associate as client devices. Use the AWS IoT console to create a thing and download security certificates to your client device.

   b. The target core device is automatically selected as "Panda_Greengrass" and In the associate client devices we have selected "Greengrass_client_laptop". In the part 2 we have already associated the client device "Greengrass_client_laptop" to the core device "Panda_Greengrass".


Associated client devices (1) Info

Manage which client devices (AWS IoT things) associate with this core device. Associated client devices can discover this core device's endpoints using cloud discovery.

Disassociate | Associate client devices

| Name | Client device associated |
|---|---|
| Greengrass_client_laptop | 24 hours ago |

**Step 3: Configure and deploy Greengrass components**

Greengrass core devices require a set of components to support client devices. Select and configure the components to deploy. Then, choose **Review and deploy** to open the deployment page, where you create a new deployment or revise the latest deployment for the core device target.

**Check component compatibility** Info

Core devices require Greengrass nucleus v2.2.0 or later to support client devices. If the target core device runs an earlier version, you must deploy Greengrass nucleus v2.2.0 or later to update it to a compatible version.

If you update the nucleus version, check that other public components in the deployment use compatible versions. You can review these components after you choose **Review and deploy**.

✓ aws.greengrass.Nucleus

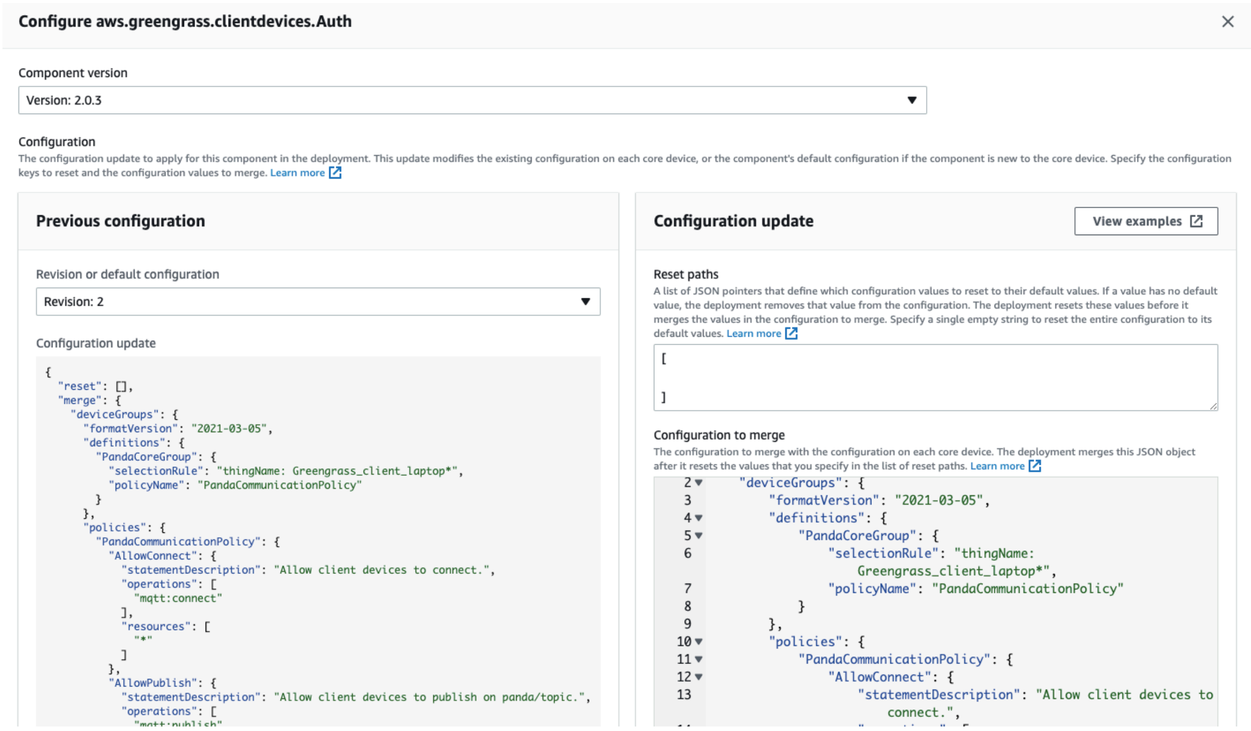
This component is the minimum installation of the AWS IoT Greengrass Core software, which every core device runs. Client device support requires nucleus v2.2.0 or later.

Edit configuration

✓ aws.greengrass.clientdevices.Auth Info

c. We will not modify the configuration of aws.greengrass.Nucleus; because on our core device we have installed the same version of nucleus component and hence both are compatible.
d. We will modify the configuration of aws.greengrass.clientdevices.Auth component. We will mention the policy with name “PandaCommunicationPolicy”. This policy will allow client device with name “Greengrass_client_laptop” to communicate over the mqtt protocol. This policy will specify the topic name to publish and subscribe. We have restricted the publish subcribe for the topics “panda/topic.” We have specified the below configuration:

```
{
 "deviceGroups": {
  "formatVersion": "2021-03-05",
  "definitions": {
   "PandaCoreGroup": {
    "selectionRule": "thingName: Greengrass_client_laptop*",
    "policyName": "PandaCommunicationPolicy"
   }
  },
  "policies": {
   "PandaCommunicationPolicy": {
    "AllowConnect": {
     "statementDescription": "Allow client devices to connect.",
     "operations": [
      "mqtt:connect"
     ],
     "resources": [
          "*"
        ]
      },
      "AllowPublish": {
        "statementDescription": "Allow client devices to publish on panda/topic.",
        "operations": [
          "mqtt:publish"
        ],
        "resources": [
          "mqtt:topic:panda/topic"
        ]
      },
      "AllowSubscribe": {
        "statementDescription": "Allow client devices to subscribe to
panda/topic/response.",
        "operations": [
          "mqtt:subscribe"
        ],
        "resources": [
          "mqtt:topicfilter:panda/topic/response"
        ]
      }
    }
  }
 }
}
```



**Configure aws.greengrass.clientdevices.Auth**

Component version

Version: 2.0.3

Configuration

The configuration update to apply for this component in the deployment. This update modifies the existing configuration on each core device, or the component's default configuration if the component is new to the core device. Specify the configuration keys to reset and the configuration values to merge. Learn more

**Previous configuration**

Revision or default configuration

Revision: 2

Configuration update

```
{
  "reset": [],
  "merge": {
    "deviceGroups": {
      "formatVersion": "2021-03-05",
      "definitions": {
        "PandaCoreGroup": {
          "selectionRule": "thingName: Greengrass_client_laptop*",
          "policyName": "PandaCommunicationPolicy"
        }
      },
      "policies": {
        "PandaCommunicationPolicy": {
          "AllowConnect": {
            "statementDescription": "Allow client devices to connect.",
            "operations": [
              "mqtt:connect"
            ],
            "resources": [
              "*"
            ]
          },
          "AllowPublish": {
            "statementDescription": "Allow client devices to publish on panda/topic.",
            "operations": [
```

**Configuration update**

Reset paths

A list of JSON pointers that define which configuration values to reset to their default values. If a value has no default value, the deployment removes that value from the configuration. The deployment resets these values before it merges the values in the configuration to merge. Specify a single empty string to reset the entire configuration to its default values. Learn more

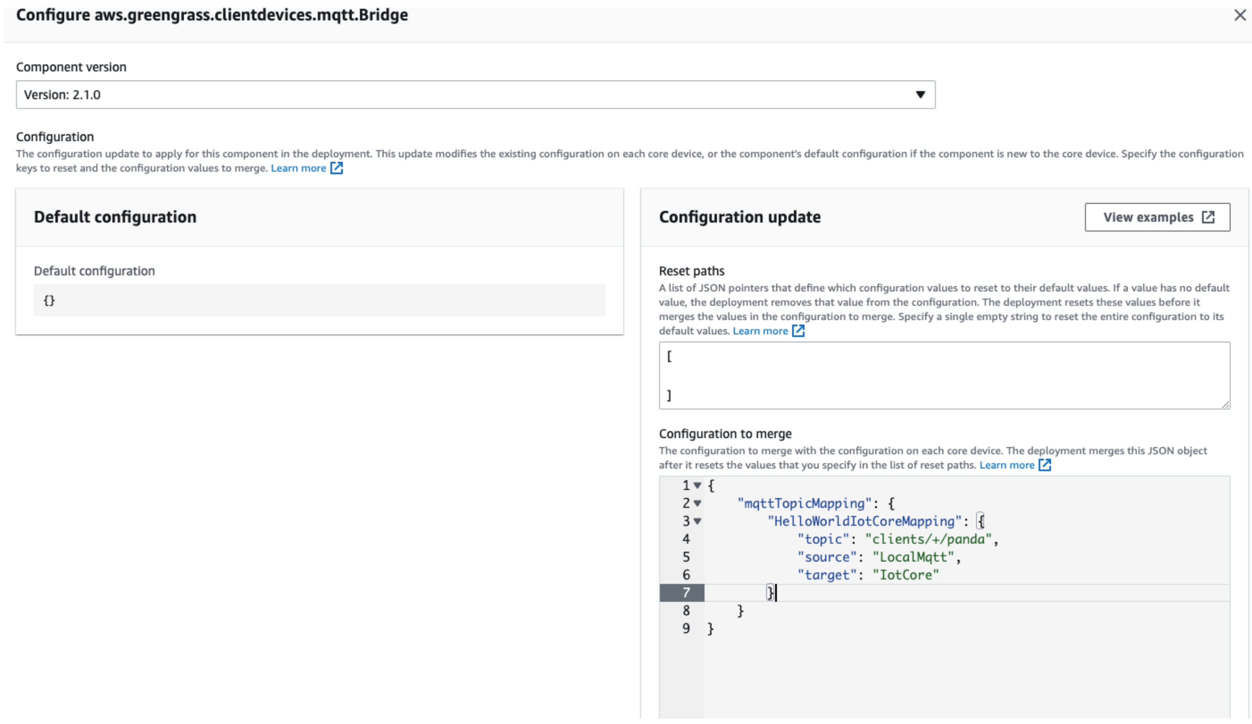
```
[

]
```

Configuration to merge

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

```
    "deviceGroups": {
        "formatVersion": "2021-03-05",
        "definitions": {
            "PandaCoreGroup": {
                "selectionRule": "thingName:
                    Greengrass_client_laptop*",
                "policyName": "PandaCommunicationPolicy"
            }
        },
        "policies": {
            "PandaCommunicationPolicy": {
                "AllowConnect": {
                    "statementDescription": "Allow client devices to
                        connect.",
```

e. We will modify the configuration of aws.greengrass.clientdevices.mqtt.Bridge component. We have specified the topic filter as “panda”; to allow only topic with name “Panda”


**Configure aws.greengrass.clientdevices.mqtt.Bridge**

Component version

Version: 2.1.0

Configuration

The configuration update to apply for this component in the deployment. This update modifies the existing configuration on each core device, or the component's default configuration if the component is new to the core device. Specify the configuration keys to reset and the configuration values to merge. Learn more

**Default configuration**

Default configuration

```
{}
```

**Configuration update**

Reset paths

A list of JSON pointers that define which configuration values to reset to their default values. If a value has no default value, the deployment removes that value from the configuration. The deployment resets these values before it merges the values in the configuration to merge. Specify a single empty string to reset the entire configuration to its default values. Learn more

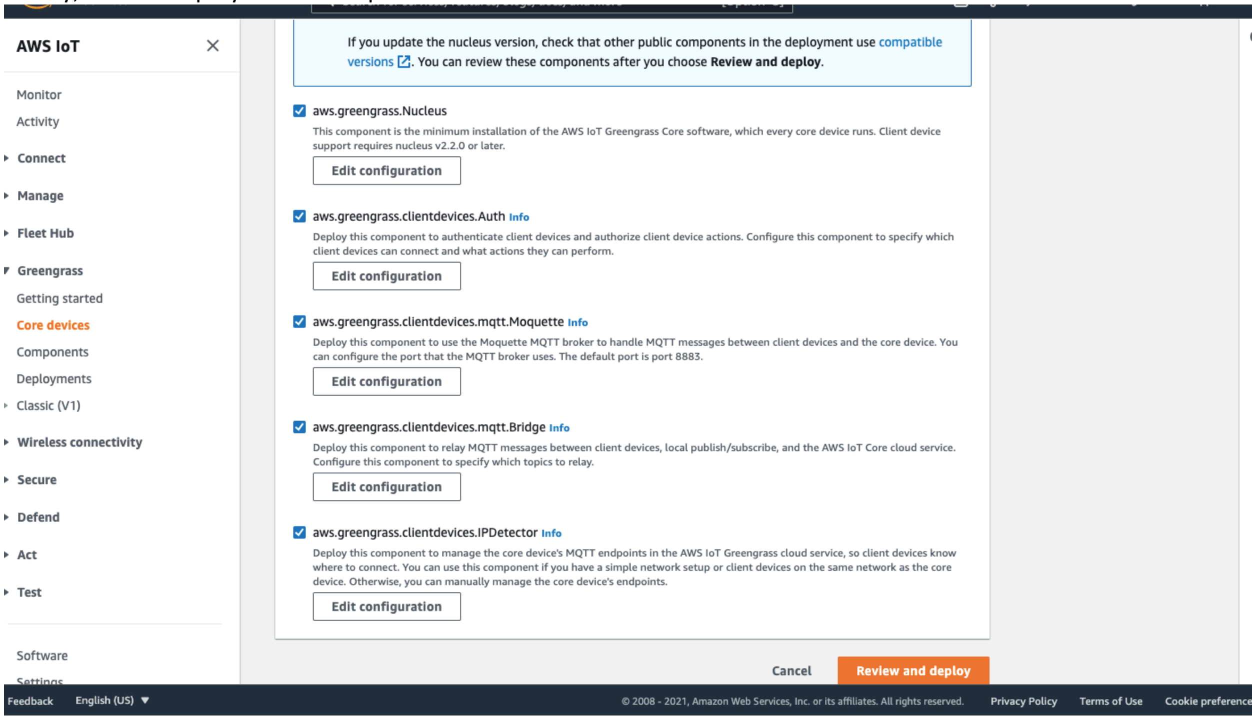
```
[

]
```

Configuration to merge

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

```
{
    "mqttTopicMapping": {
        "HelloWorldIotCoreMapping": {
            "topic": "clients/+/panda",
            "source": "LocalMqtt",
            "target": "IotCore"
        }
    }
}
```

f. Finally, we will deploy all the components.



If you update the nucleus version, check that other public components in the deployment use compatible versions. You can review these components after you choose **Review and deploy**.

☑ aws.greengrass.Nucleus
This component is the minimum installation of the AWS IoT Greengrass Core software, which every core device runs. Client device support requires nucleus v2.2.0 or later.
Edit configuration

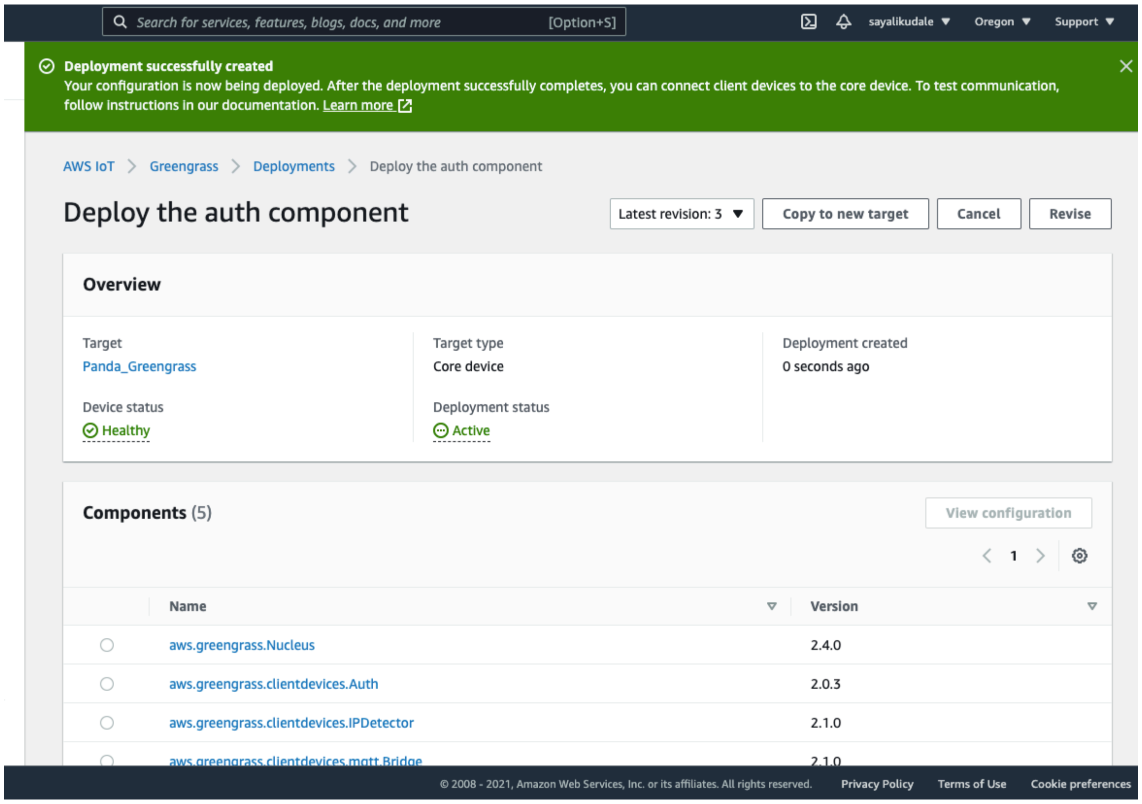
☑ aws.greengrass.clientdevices.Auth Info
Deploy this component to authenticate client devices and authorize client device actions. Configure this component to specify which client devices can connect and what actions they can perform.
Edit configuration

☑ aws.greengrass.clientdevices.mqtt.Moquette Info
Deploy this component to use the Moquette MQTT broker to handle MQTT messages between client devices and the core device. You can configure the port that the MQTT broker uses. The default port is port 8883.
Edit configuration

☑ aws.greengrass.clientdevices.mqtt.Bridge Info
Deploy this component to relay MQTT messages between client devices, local publish/subscribe, and the AWS IoT Core cloud service. Configure this component to specify which topics to relay.
Edit configuration

☑ aws.greengrass.clientdevices.IPDetector Info
Deploy this component to manage the core device's MQTT endpoints in the AWS IoT Greengrass cloud service, so client devices know where to connect. You can use this component if you have a simple network setup or client devices on the same network as the core device. Otherwise, you can manually manage the core device's endpoints.
Edit configuration

g. With this successful deployment client device will be able to retrieve the core devices' IP address and certificated and use to connect.


Search for services, features, blogs, docs, and more [Option+S]
sayalikudale ▼
Oregon ▼
Support ▼

**Deployment successfully created**
Your configuration is now being deployed. After the deployment successfully completes, you can connect client devices to the core device. To test communication, follow instructions in our documentation. Learn more

AWS IoT > Greengrass > Deployments > Deploy the auth component

Deploy the auth component

**Overview**

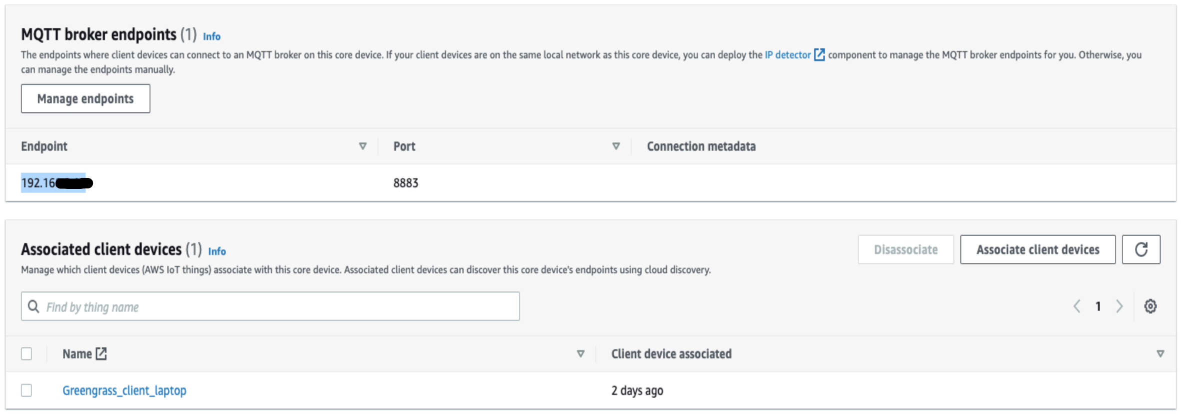
| | | |
|---|---|---|
| Target<br>Panda_Greengrass | Target type<br>Core device | Deployment created<br>0 seconds ago |
| Device status<br>✓ Healthy | Deployment status<br>Active | |

**Components** (5)

| | Name | Version |
|---|---|---|
| ○ | aws.greengrass.Nucleus | 2.4.0 |
| ○ | aws.greengrass.clientdevices.Auth | 2.0.3 |
| ○ | aws.greengrass.clientdevices.IPDetector | 2.1.0 |
| ○ | aws.greengrass.clientdevices.mqtt.Bridge | 2.1.0 |

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

h. Core device's IP address can be seen as MQTT broker endpoint in client devices section.


**MQTT broker endpoints** (1) Info

The endpoints where client devices can connect to an MQTT broker on this core device. If your client devices are on the same local network as this core device, you can deploy the IP detector component to manage the MQTT broker endpoints for you. Otherwise, you can manage the endpoints manually.

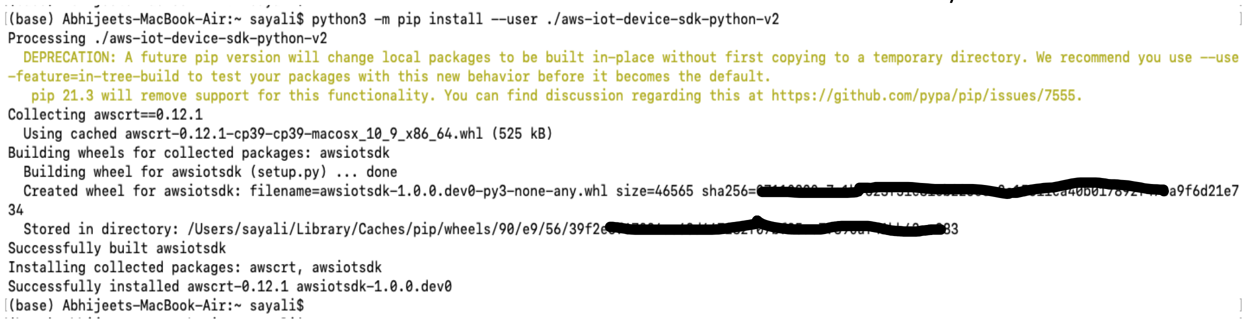
| Endpoint | Port | Connection metadata |
|---|---|---|
| 192.16[redacted] | 8883 | |

**Associated client devices** (1) Info

Manage which client devices (AWS IoT things) associate with this core device. Associated client devices can discover this core device's endpoints using cloud discovery.

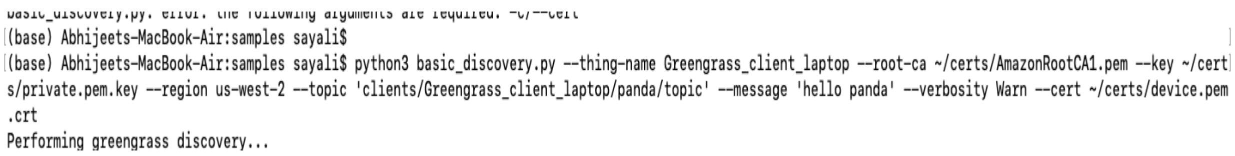
| | Name | Client device associated |
|---|---|---|
| ☐ | Greengrass_client_laptop | 2 days ago |

4. Test the client and core devices communication:
   a. In the client device we have downloaded and installed the AWS IoT Device SDK v2 for Python.



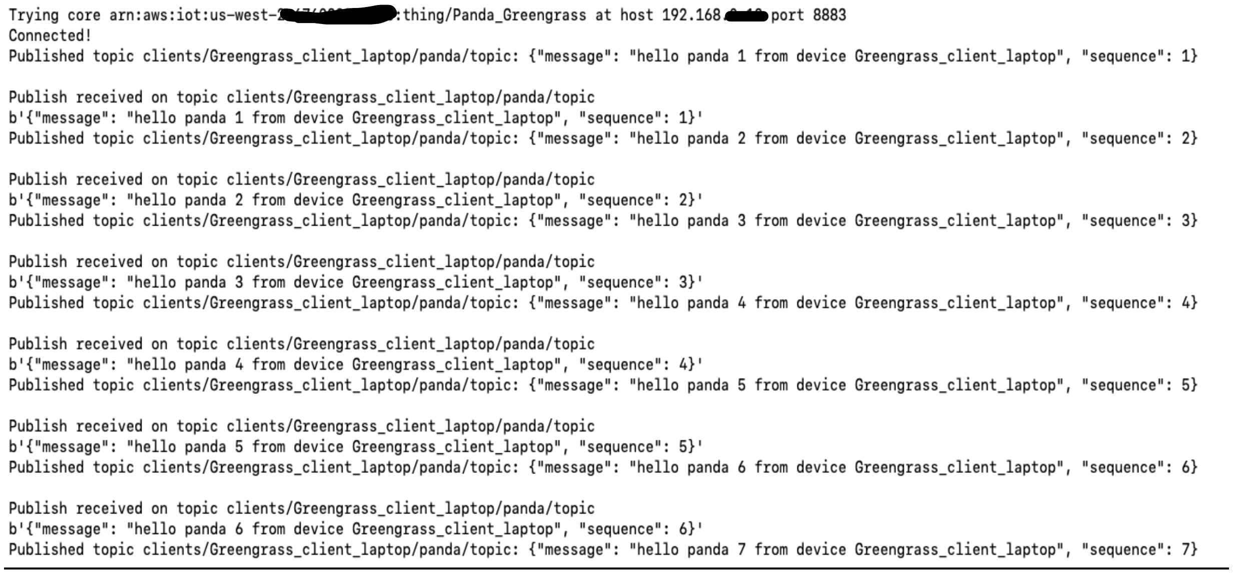
```
(base) Abhijeets-MacBook-Air:~ sayali$ python3 -m pip install --user ./aws-iot-device-sdk-python-v2
Processing ./aws-iot-device-sdk-python-v2
  DEPRECATION: A future pip version will change local packages to be built in-place without first copying to a temporary directory. We recommend you use --use
-feature=in-tree-build to test your packages with this new behavior before it becomes the default.
   pip 21.3 will remove support for this functionality. You can find discussion regarding this at https://github.com/pypa/pip/issues/7555.
Collecting awscrt==0.12.1
  Using cached awscrt-0.12.1-cp39-cp39-macosx_10_9_x86_64.whl (525 kB)
Building wheels for collected packages: awsiotsdk
  Building wheel for awsiotsdk (setup.py) ... done
  Created wheel for awsiotsdk: filename=awsiotsdk-1.0.0.dev0-py3-none-any.whl size=46565 sha256=[redacted]a9f6d21e7
34
  Stored in directory: /Users/sayali/Library/Caches/pip/wheels/90/e9/56/39f2[redacted]83
Successfully built awsiotsdk
Installing collected packages: awscrt, awsiotsdk
Successfully installed awscrt-0.12.1 awsiotsdk-1.0.0.dev0
(base) Abhijeets-MacBook-Air:~ sayali$
```

b. In the client device we will run the script “basic_discovery.py” and we will pass all the certs and keys., region, thing name, message.



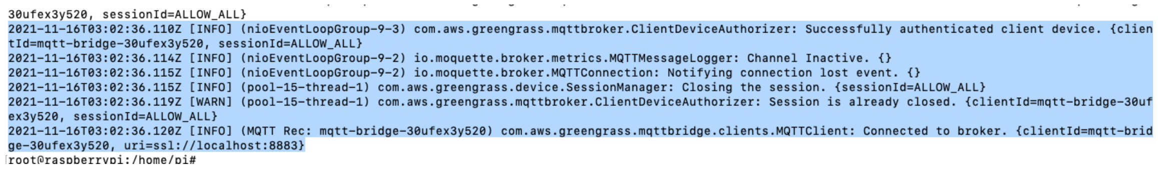
```
basic_discovery.py: error: the following arguments are required: -c/--cert
(base) Abhijeets-MacBook-Air:samples sayali$
(base) Abhijeets-MacBook-Air:samples sayali$ python3 basic_discovery.py --thing-name Greengrass_client_laptop --root-ca ~/certs/AmazonRootCA1.pem --key ~/cert
s/private.pem.key --region us-west-2 --topic 'clients/Greengrass_client_laptop/panda/topic' --message 'hello panda' --verbosity Warn --cert ~/certs/device.pem
.crt
Performing greengrass discovery...
```

c. The script will publish messages to the core device after every 10 secs. To get the core device endpoint we have used the **discovery api**. Also, in the message we have appended sequence number and device name.



```
Trying core arn:aws:iot:us-west-[redacted]:thing/Panda_Greengrass at host 192.168[redacted]port 8883
Connected!
Published topic clients/Greengrass_client_laptop/panda/topic: {"message": "hello panda 1 from device Greengrass_client_laptop", "sequence": 1}

Publish received on topic clients/Greengrass_client_laptop/panda/topic
b'{"message": "hello panda 1 from device Greengrass_client_laptop", "sequence": 1}'
Published topic clients/Greengrass_client_laptop/panda/topic: {"message": "hello panda 2 from device Greengrass_client_laptop", "sequence": 2}

Publish received on topic clients/Greengrass_client_laptop/panda/topic
b'{"message": "hello panda 2 from device Greengrass_client_laptop", "sequence": 2}'
Published topic clients/Greengrass_client_laptop/panda/topic: {"message": "hello panda 3 from device Greengrass_client_laptop", "sequence": 3}

Publish received on topic clients/Greengrass_client_laptop/panda/topic
b'{"message": "hello panda 3 from device Greengrass_client_laptop", "sequence": 3}'
Published topic clients/Greengrass_client_laptop/panda/topic: {"message": "hello panda 4 from device Greengrass_client_laptop", "sequence": 4}

Publish received on topic clients/Greengrass_client_laptop/panda/topic
b'{"message": "hello panda 4 from device Greengrass_client_laptop", "sequence": 4}'
Published topic clients/Greengrass_client_laptop/panda/topic: {"message": "hello panda 5 from device Greengrass_client_laptop", "sequence": 5}

Publish received on topic clients/Greengrass_client_laptop/panda/topic
b'{"message": "hello panda 5 from device Greengrass_client_laptop", "sequence": 5}'
Published topic clients/Greengrass_client_laptop/panda/topic: {"message": "hello panda 6 from device Greengrass_client_laptop", "sequence": 6}

Publish received on topic clients/Greengrass_client_laptop/panda/topic
b'{"message": "hello panda 6 from device Greengrass_client_laptop", "sequence": 6}'
Published topic clients/Greengrass_client_laptop/panda/topic: {"message": "hello panda 7 from device Greengrass_client_laptop", "sequence": 7}
```

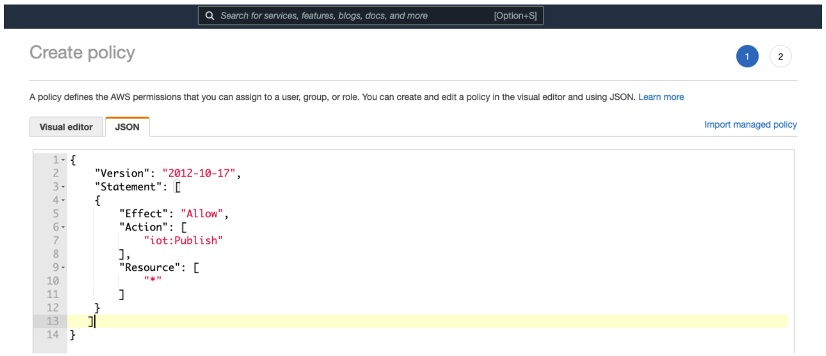
5. Core device authenticates the client.



```
30ufex3y520, sessionId=ALLOW_ALL}
2021-11-16T03:02:36.110Z [INFO] (nioEventLoopGroup-9-3) com.aws.greengrass.mqttbroker.ClientDeviceAuthorizer: Successfully authenticated client device. {clien
tId=mqtt-bridge-30ufex3y520, sessionId=ALLOW_ALL}
2021-11-16T03:02:36.114Z [INFO] (nioEventLoopGroup-9-2) io.moquette.broker.metrics.MQTTMessageLogger: Channel Inactive. {}
2021-11-16T03:02:36.115Z [INFO] (nioEventLoopGroup-9-2) io.moquette.broker.MQTTConnection: Notifying connection lost event. {}
2021-11-16T03:02:36.115Z [INFO] (pool-15-thread-1) com.aws.greengrass.device.SessionManager: Closing the session. {sessionId=ALLOW_ALL}
2021-11-16T03:02:36.119Z [WARN] (pool-15-thread-1) com.aws.greengrass.mqttbroker.ClientDeviceAuthorizer: Session is already closed. {clientId=mqtt-bridge-30uf
ex3y520, sessionId=ALLOW_ALL}
2021-11-16T03:02:36.120Z [INFO] (MQTT Rec: mqtt-bridge-30ufex3y520) com.aws.greengrass.mqttbridge.clients.MQTTClient: Connected to broker. {clientId=mqtt-brid
ge-30ufex3y520, uri=ssl://localhost:8883}
root@raspberrypi:/home/pi#
```

## Part 4 : Develop and deploy lambda function to core device

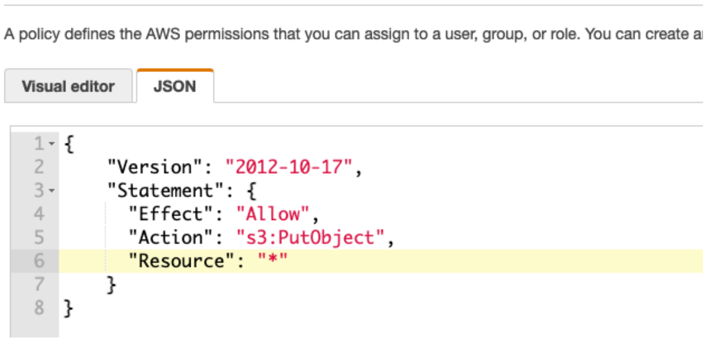
1. Developed the LambdaOnCoreD2C lambda function.
    a. Created the lambda function named LambdaOnCoreD2C using Python 3.7. The Core device already has Python 3.7 installed.
    b. Added the iot publish policy. To publish the MQTT messages from lambda function.


Search for services, features, blogs, docs, and more [Option+S]
Create policy
1
2
Visual editor
JSON
Import managed policy

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iot:Publish"
        ],
        "Resource": [
            "*"
        ]
    }
  ]
}
```

c. Added S3 database policy to save the files to S3 dynamo db.

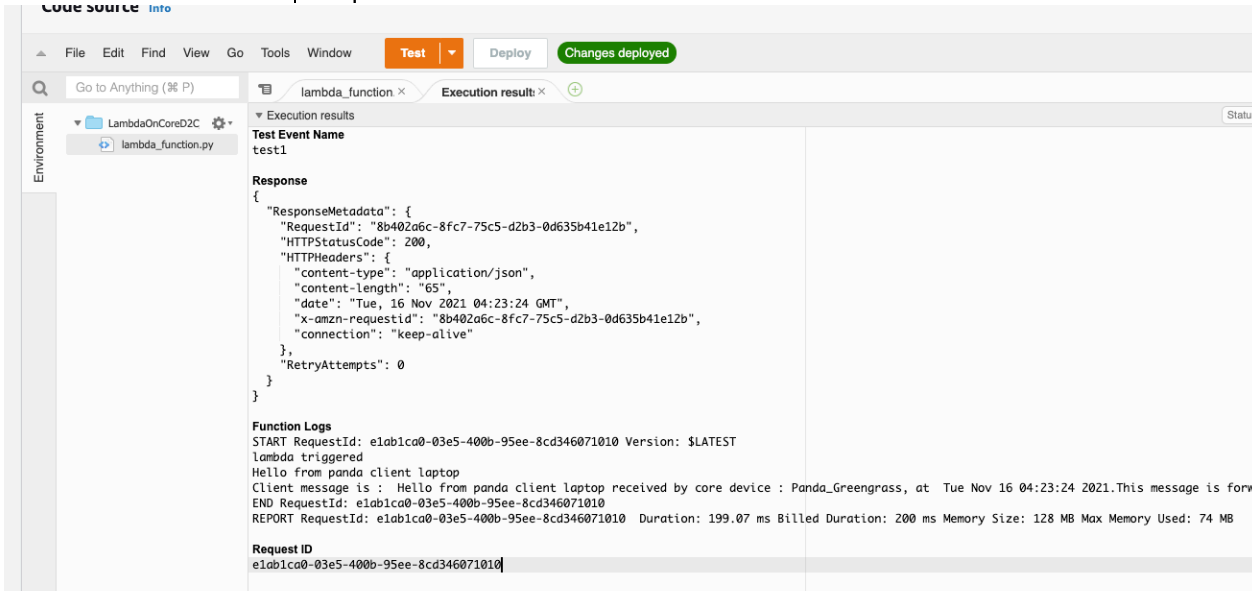


A policy defines the AWS permissions that you can assign to a user, group, or role. You can create a

```
{
    "Version": "2012-10-17",
    "Statement": {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "*"
    }
}
```

d. Lambda function will receive messages in the events and code is written to modify the message to add the core device name, timestamp and direction. Topic is “PandaCoreData” and Q0S used is 0.

e. Lambda is tested with sample input.

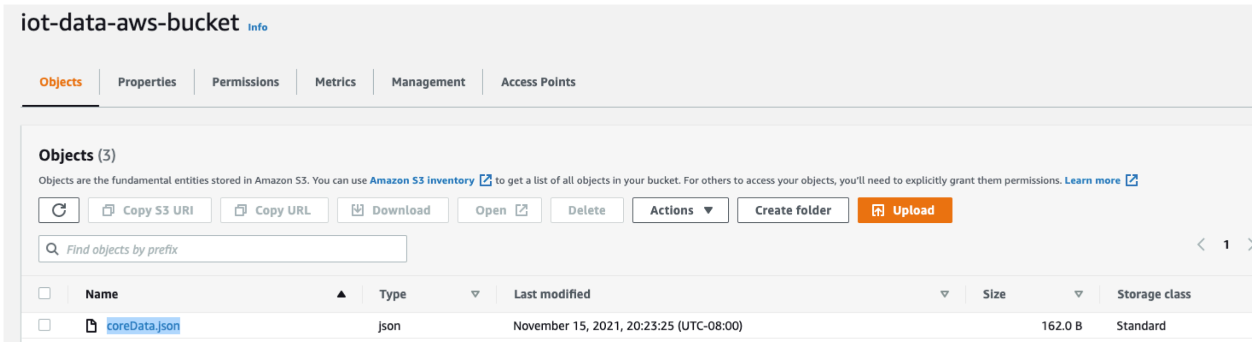


```
Test Event Name
test1

Response
{
  "ResponseMetadata": {
    "RequestId": "8b402a6c-8fc7-75c5-d2b3-0d635b41e12b",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "content-type": "application/json",
      "content-length": "65",
      "date": "Tue, 16 Nov 2021 04:23:24 GMT",
      "x-amzn-requestid": "8b402a6c-8fc7-75c5-d2b3-0d635b41e12b",
      "connection": "keep-alive"
    },
    "RetryAttempts": 0
  }
}

Function Logs
START RequestId: e1ab1ca0-03e5-400b-95ee-8cd346071010 Version: $LATEST
lambda triggered
Hello from panda client laptop
Client message is :  Hello from panda client laptop received by core device : Panda_Greengrass, at  Tue Nov 16 04:23:24 2021.This message is forw
END RequestId: e1ab1ca0-03e5-400b-95ee-8cd346071010
REPORT RequestId: e1ab1ca0-03e5-400b-95ee-8cd346071010  Duration: 199.07 ms Billed Duration: 200 ms Memory Size: 128 MB Max Memory Used: 74 MB

Request ID
e1ab1ca0-03e5-400b-95ee-8cd346071010
```

f. The Lambda function publish the formatted messages to the AWS IoT console.

g. The Data is also saved in the S3 dynamo db.

h. The lambda function is then published to the newer version.

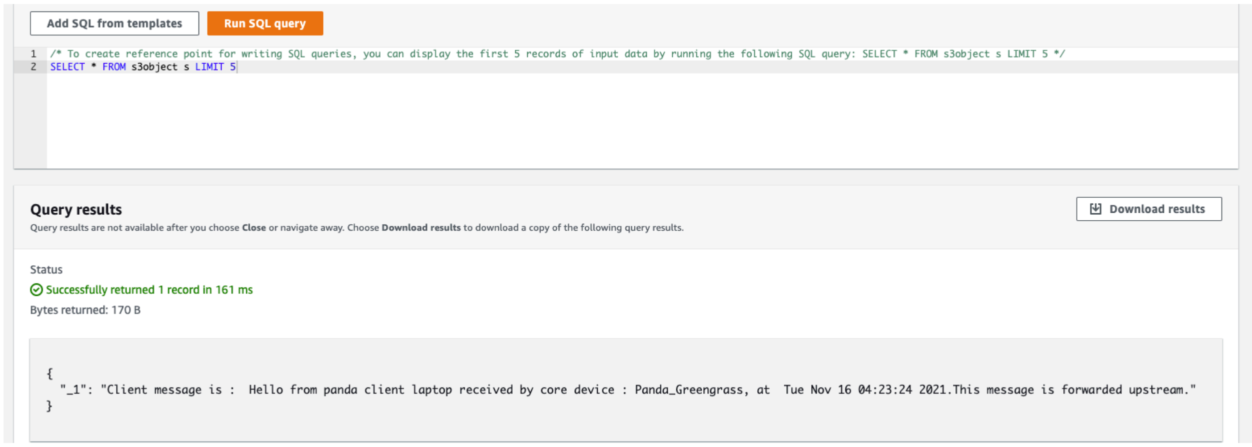

iot-data-aws-bucket Info

Objects | Properties | Permissions | Metrics | Management | Access Points

**Objects (3)**

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

| Name | Type | Last modified | Size | Storage class |
|---|---|---|---|---|
| coreData.json | json | November 15, 2021, 20:23:25 (UTC-08:00) | 162.0 B | Standard |


```
/* To create reference point for writing SQL queries, you can display the first 5 records of input data by running the following SQL query: SELECT * FROM s3object s LIMIT 5 */
SELECT * FROM s3object s LIMIT 5
```

**Query results**

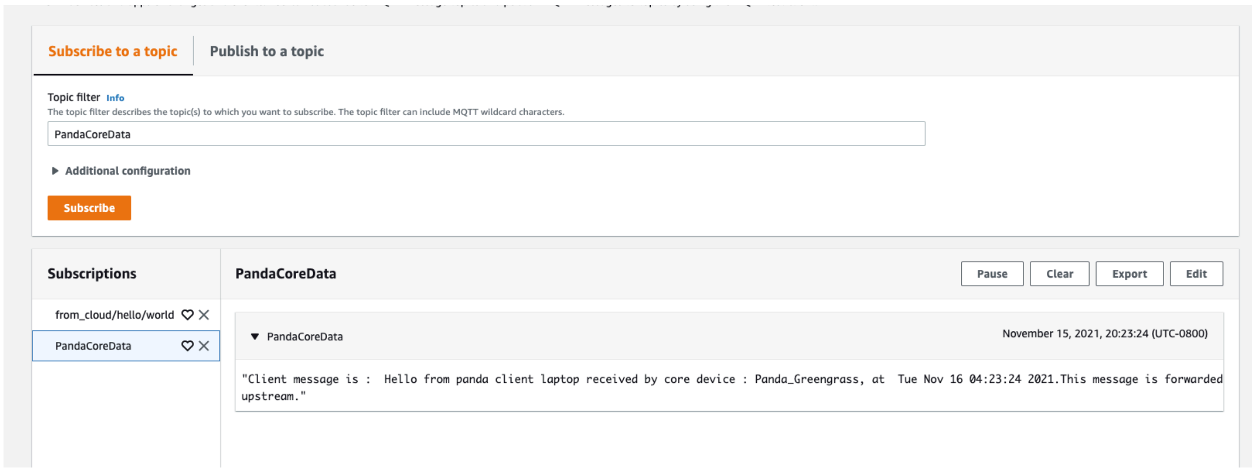
Query results are not available after you choose **Close** or navigate away. Choose **Download results** to download a copy of the following query results.

Status

Successfully returned 1 record in 161 ms

Bytes returned: 170 B

```
{
  "_1": "Client message is :  Hello from panda client laptop received by core device : Panda_Greengrass, at  Tue Nov 16 04:23:24 2021.This message is forwarded upstream."
}
```


Subscribe to a topic | Publish to a topic

Topic filter Info

The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

PandaCoreData

Additional configuration

**Subscriptions**

from_cloud/hello/world

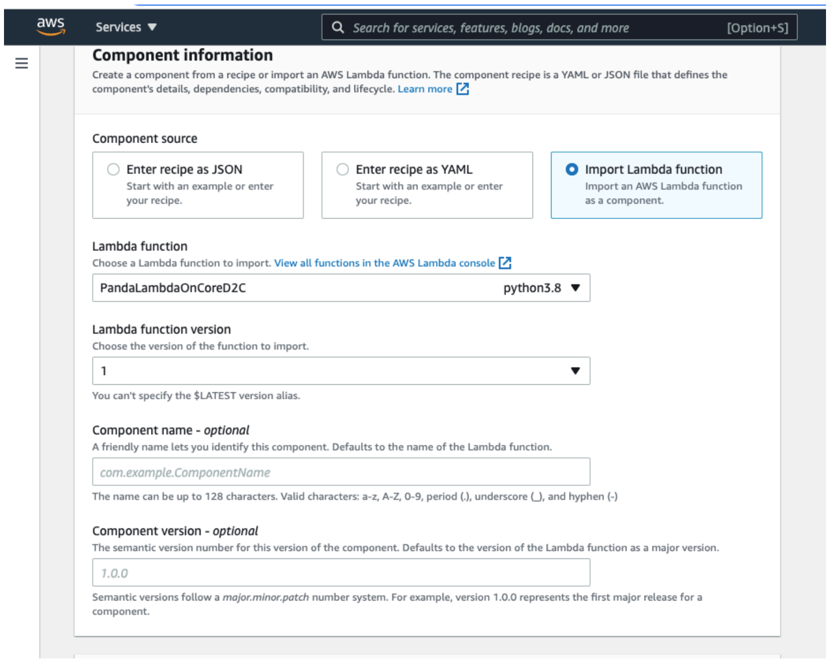
PandaCoreData

**PandaCoreData**

PandaCoreData November 15, 2021, 20:23:24 (UTC-0800)

```
"Client message is :  Hello from panda client laptop received by core device : Panda_Greengrass, at  Tue Nov 16 04:23:24 2021.This message is forwarded upstream."
```

2. Configure Lambda Function as Component.
    a. Lambda function LambdaOnCoreD2C is added as a component.


aws
Services ▼
Search for services, features, blogs, docs, and more [Option+S]

**Component information**

Create a component from a recipe or import an AWS Lambda function. The component recipe is a YAML or JSON file that defines the component's details, dependencies, compatibility, and lifecycle. Learn more

Component source

- ○ Enter recipe as JSON — Start with an example or enter your recipe.
- ○ Enter recipe as YAML — Start with an example or enter your recipe.
- ● Import Lambda function — Import an AWS Lambda function as a component.

Lambda function
Choose a Lambda function to import. View all functions in the AWS Lambda console
PandaLambdaOnCoreD2C python3.8

Lambda function version
Choose the version of the function to import.
1
You can't specify the $LATEST version alias.

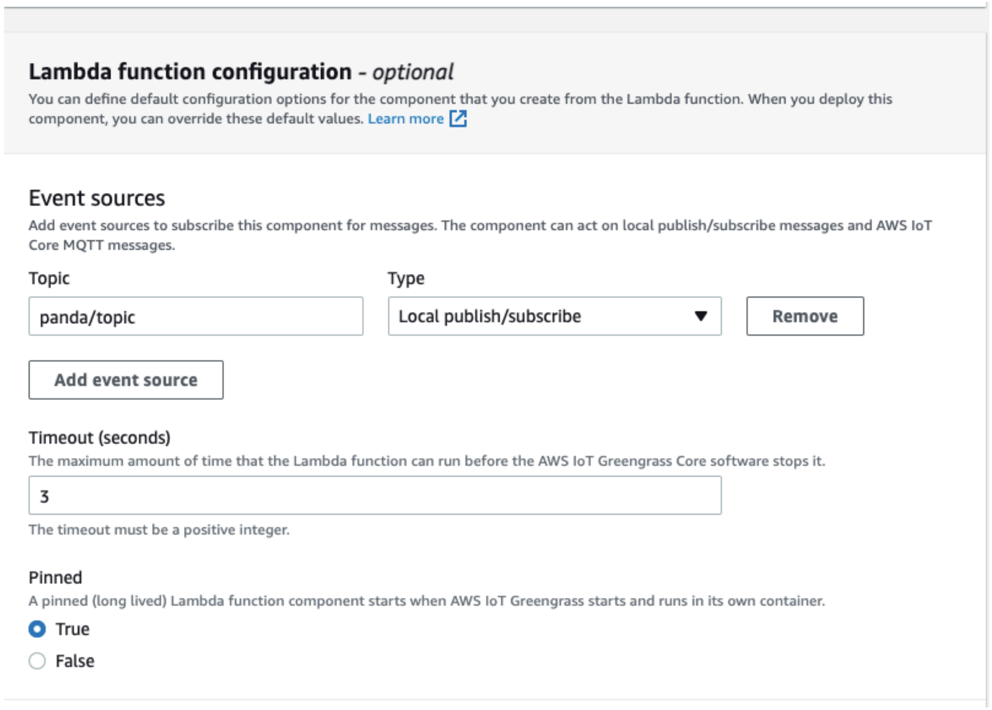
Component name - *optional*
A friendly name lets you identify this component. Defaults to the name of the Lambda function.
com.example.ComponentName
The name can be up to 128 characters. Valid characters: a-z, A-Z, 0-9, period (.), underscore (_), and hyphen (-)

Component version - *optional*
The semantic version number for this version of the component. Defaults to the version of the Lambda function as a major version.
1.0.0
Semantic versions follow a *major.minor.patch* number system. For example, version 1.0.0 represents the first major release for a component.

b. Configuration of the **Event Source** lambda parameters:

   i. The lambda function should receive the messages published by the client devices hence in the event source we should specify the topic name to subscribe to the messages and also the type of event source. Type of event source is specified as Local publish/subscribe.

   ii. This setting will be used to subscribe to the messages published by the client device in part 3 using topic panda/Topic.



**Lambda function configuration** - *optional*

You can define default configuration options for the component that you create from the Lambda function. When you deploy this component, you can override these default values. Learn more

Event sources

Add event sources to subscribe this component for messages. The component can act on local publish/subscribe messages and AWS IoT Core MQTT messages.

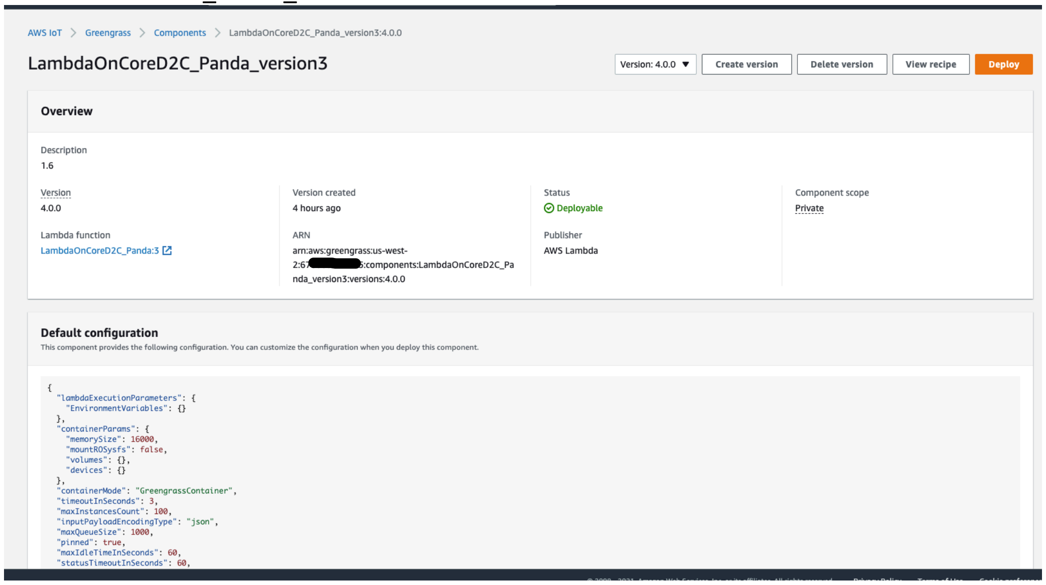
| Topic | Type |
|---|---|
| panda/topic | Local publish/subscribe |

Timeout (seconds)
The maximum amount of time that the Lambda function can run before the AWS IoT Greengrass Core software stops it.
3
The timeout must be a positive integer.

Pinned
A pinned (long lived) Lambda function component starts when AWS IoT Greengrass starts and runs in its own container.
[✓ True] False

c. All other parameters are used as default.

d. Review and create the lambda function component. Here component name is LamdaOnCoreD2C_Panda_version3



AWS IoT > Greengrass > Components > LambdaOnCoreD2C_Panda_version3:4.0.0

LambdaOnCoreD2C_Panda_version3

**Overview**

Description
1.6

Version
4.0.0

Version created
4 hours ago

Status
Deployable

Component scope
Private

Lambda function
LambdaOnCoreD2C_Panda:3

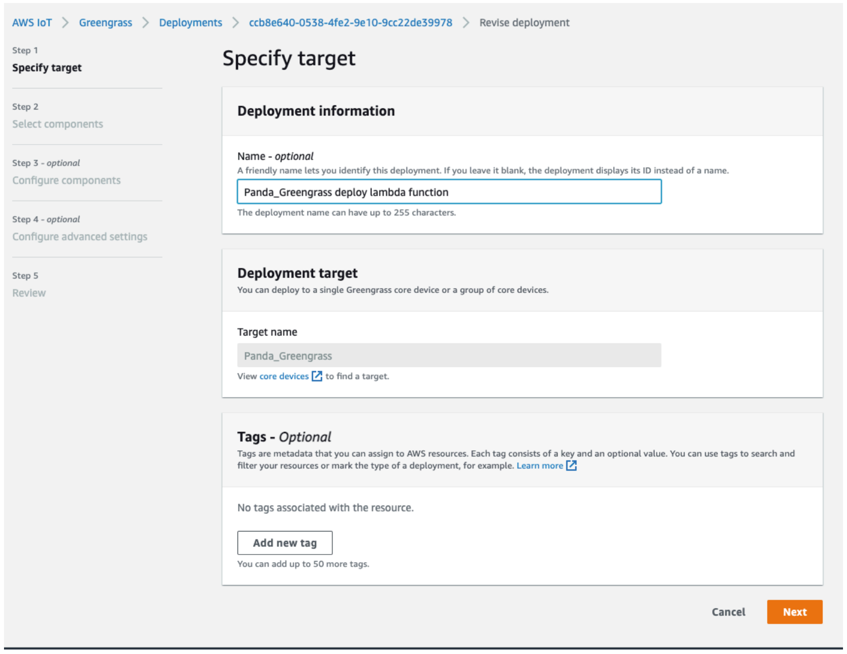
ARN
arn:aws:greengrass:us-west-2:67[redacted]5:components:LambdaOnCoreD2C_Panda_version3:versions:4.0.0

Publisher
AWS Lambda

**Default configuration**
This component provides the following configuration. You can customize the configuration when you deploy this component.

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {}
  },
  "containerParams": {
    "memorySize": 16000,
    "mountROSysfs": false,
    "volumes": {},
    "devices": {}
  },
  "containerMode": "GreengrassContainer",
  "timeoutInSeconds": 3,
  "maxInstancesCount": 100,
  "inputPayloadEncodingType": "json",
  "maxQueueSize": 1000,
  "pinned": true,
  "maxIdleTimeInSeconds": 60,
  "statusTimeoutInSeconds": 60,
```

3. Deploy the component to AWS core device.
   a. Specify the deployment name of and target will be the core device and specify the name of core device.



AWS IoT > Greengrass > Deployments > ccb8e640-0538-4fe2-9e10-9cc22de39978 > Revise deployment

Step 1
Specify target

Step 2
Select components

Step 3 - *optional*
Configure components

Step 4 - *optional*
Configure advanced settings

Step 5
Review

Specify target

**Deployment information**

Name - *optional*
A friendly name lets you identify this deployment. If you leave it blank, the deployment displays its ID instead of a name.
Panda_Greengrass deploy lambda function
The deployment name can have up to 255 characters.

**Deployment target**
You can deploy to a single Greengrass core device or a group of core devices.

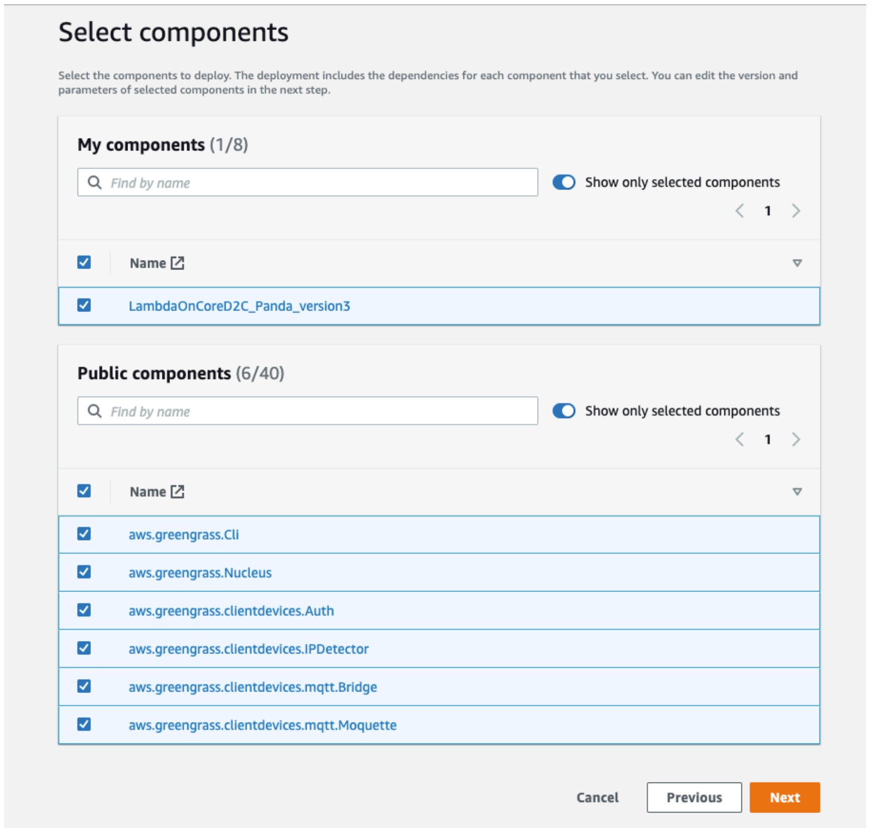
Target name
Panda_Greengrass
View core devices to find a target.

**Tags - *Optional***
Tags are metadata that you can assign to AWS resources. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or mark the type of a deployment, for example. Learn more

No tags associated with the resource.

You can add up to 50 more tags.

   b. Along with existing components Lambda function component is added in the selected component list that will be deployed on the core device.



## Select components

Select the components to deploy. The deployment includes the dependencies for each component that you select. You can edit the version and parameters of selected components in the next step.

**My components** (1/8)

Show only selected components

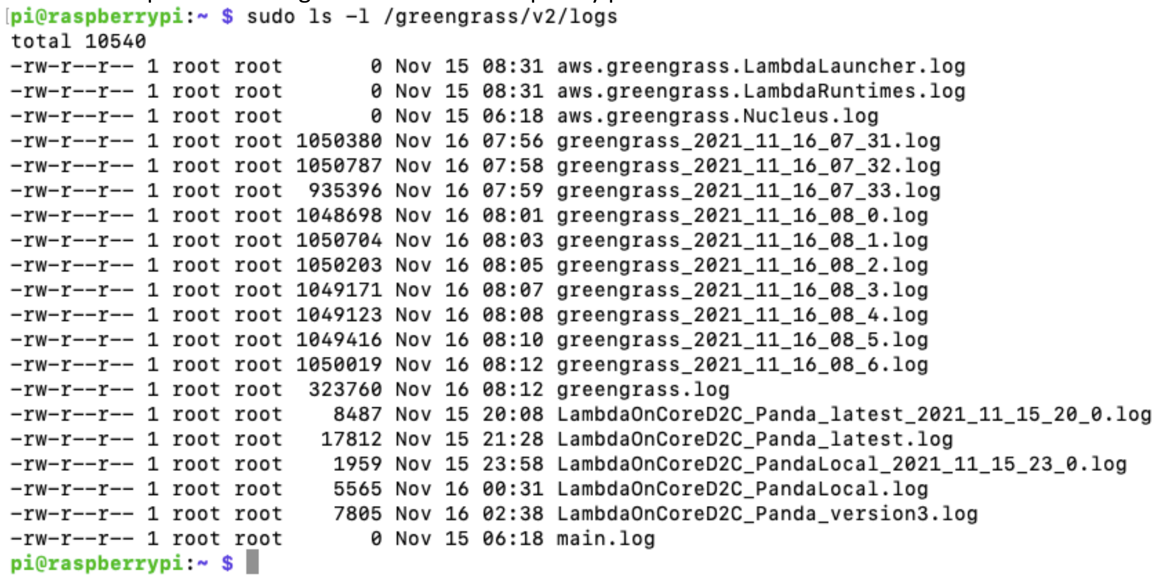
| Name |
|---|
| ✓ LambdaOnCoreD2C_Panda_version3 |

**Public components** (6/40)

Show only selected components

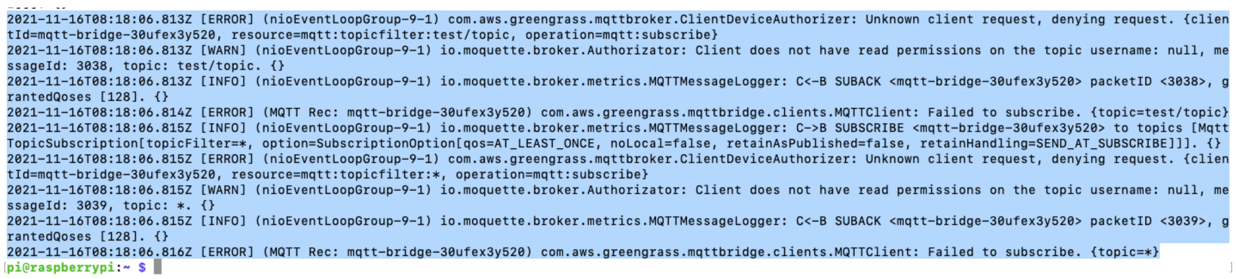
| Name |
|---|
| ✓ aws.greengrass.Cli |
| ✓ aws.greengrass.Nucleus |
| ✓ aws.greengrass.clientdevices.Auth |
| ✓ aws.greengrass.clientdevices.IPDetector |
| ✓ aws.greengrass.clientdevices.mqtt.Bridge |
| ✓ aws.greengrass.clientdevices.mqtt.Moquette |

c. Lambda function is successfully deployed on the core device. We can see the log file with lambda function component name got created in the raspberry pi core device.



```
pi@raspberrypi:~ $ sudo ls -l /greengrass/v2/logs
total 10540
-rw-r--r-- 1 root root       0 Nov 15 08:31 aws.greengrass.LambdaLauncher.log
-rw-r--r-- 1 root root       0 Nov 15 08:31 aws.greengrass.LambdaRuntimes.log
-rw-r--r-- 1 root root       0 Nov 15 06:18 aws.greengrass.Nucleus.log
-rw-r--r-- 1 root root 1050380 Nov 16 07:56 greengrass_2021_11_16_07_31.log
-rw-r--r-- 1 root root 1050787 Nov 16 07:58 greengrass_2021_11_16_07_32.log
-rw-r--r-- 1 root root  935396 Nov 16 07:59 greengrass_2021_11_16_07_33.log
-rw-r--r-- 1 root root 1048698 Nov 16 08:01 greengrass_2021_11_16_08_0.log
-rw-r--r-- 1 root root 1050704 Nov 16 08:03 greengrass_2021_11_16_08_1.log
-rw-r--r-- 1 root root 1050203 Nov 16 08:05 greengrass_2021_11_16_08_2.log
-rw-r--r-- 1 root root 1049171 Nov 16 08:07 greengrass_2021_11_16_08_3.log
-rw-r--r-- 1 root root 1049123 Nov 16 08:08 greengrass_2021_11_16_08_4.log
-rw-r--r-- 1 root root 1049416 Nov 16 08:10 greengrass_2021_11_16_08_5.log
-rw-r--r-- 1 root root 1050019 Nov 16 08:12 greengrass_2021_11_16_08_6.log
-rw-r--r-- 1 root root  323760 Nov 16 08:12 greengrass.log
-rw-r--r-- 1 root root    8487 Nov 15 20:08 LambdaOnCoreD2C_Panda_latest_2021_11_15_20_0.log
-rw-r--r-- 1 root root   17812 Nov 15 21:28 LambdaOnCoreD2C_Panda_latest.log
-rw-r--r-- 1 root root    1959 Nov 15 23:58 LambdaOnCoreD2C_PandaLocal_2021_11_15_23_0.log
-rw-r--r-- 1 root root    5565 Nov 16 00:31 LambdaOnCoreD2C_PandaLocal.log
-rw-r--r-- 1 root root    7805 Nov 16 02:38 LambdaOnCoreD2C_Panda_version3.log
-rw-r--r-- 1 root root       0 Nov 15 06:18 main.log
pi@raspberrypi:~ $ 
```

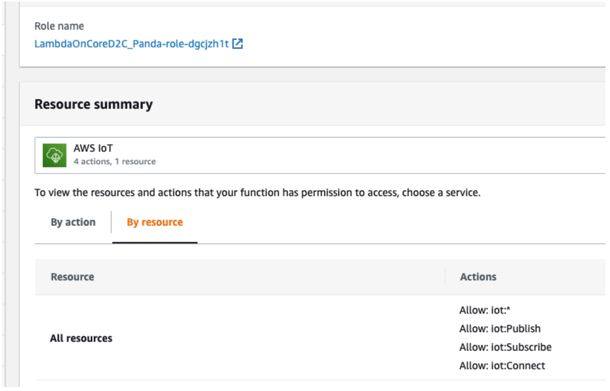
4. Test the Lambda function on the core device.
   a. As seen in part 3 the client device will publish the messages every 10 secs and core device should supposed to receive those messages.
   b. When we send messages from the client device the core device is not able to subscribe to the topics.
   c. We can see the errors in the greengrass.log file.



```
2021-11-16T08:18:06.813Z [ERROR] (nioEventLoopGroup-9-1) com.aws.greengrass.mqttbroker.ClientDeviceAuthorizer: Unknown client request, denying request. {clien
tId=mqtt-bridge-30ufex3y520, resource=mqtt:topicfilter:test/topic, operation=mqtt:subscribe}
2021-11-16T08:18:06.813Z [WARN] (nioEventLoopGroup-9-1) io.moquette.broker.Authorizator: Client does not have read permissions on the topic username: null, me
ssageId: 3038, topic: test/topic. {}
2021-11-16T08:18:06.813Z [INFO] (nioEventLoopGroup-9-1) io.moquette.broker.metrics.MQTTMessageLogger: C<-B SUBACK <mqtt-bridge-30ufex3y520> packetID <3038>, g
rantedQoses [128]. {}
2021-11-16T08:18:06.814Z [ERROR] (MQTT Rec: mqtt-bridge-30ufex3y520) com.aws.greengrass.mqttbridge.clients.MQTTClient: Failed to subscribe. {topic=test/topic}
2021-11-16T08:18:06.815Z [INFO] (nioEventLoopGroup-9-1) io.moquette.broker.metrics.MQTTMessageLogger: C->B SUBSCRIBE <mqtt-bridge-30ufex3y520> to topics [Mqtt
TopicSubscription[topicFilter=*, option=SubscriptionOption[qos=AT_LEAST_ONCE, noLocal=false, retainAsPublished=false, retainHandling=SEND_AT_SUBSCRIBE]]]. {}
2021-11-16T08:18:06.815Z [ERROR] (nioEventLoopGroup-9-1) com.aws.greengrass.mqttbroker.ClientDeviceAuthorizer: Unknown client request, denying request. {clien
tId=mqtt-bridge-30ufex3y520, resource=mqtt:topicfilter:*, operation=mqtt:subscribe}
2021-11-16T08:18:06.815Z [WARN] (nioEventLoopGroup-9-1) io.moquette.broker.Authorizator: Client does not have read permissions on the topic username: null, me
ssageId: 3039, topic: *. {}
2021-11-16T08:18:06.815Z [INFO] (nioEventLoopGroup-9-1) io.moquette.broker.metrics.MQTTMessageLogger: C<-B SUBACK <mqtt-bridge-30ufex3y520> packetID <3039>, g
rantedQoses [128]. {}
2021-11-16T08:18:06.816Z [ERROR] (MQTT Rec: mqtt-bridge-30ufex3y520) com.aws.greengrass.mqttbridge.clients.MQTTClient: Failed to subscribe. {topic=*}
pi@raspberrypi:~ $
```

   d. We are getting ClientDEviceAuthorizer error and Error says Unknown client request, denying request.

5. To resolve this issue, we have tried below things:
   a. Verified the iot permissions for the lambda function and added all the iot permissions.



Role name
LambdaOnCoreD2C_Panda-role-dgcjzh1t

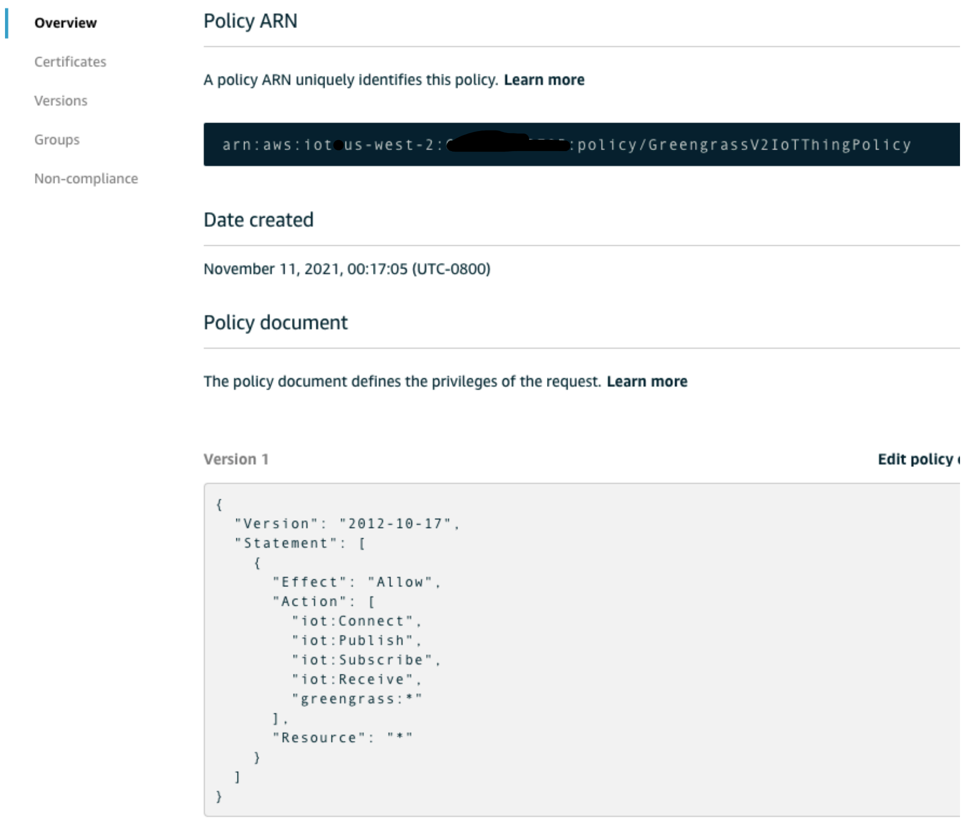
**Resource summary**

AWS IoT
4 actions, 1 resource

To view the resources and actions that your function has permission to access, choose a service.

| Resource | Actions |
|---|---|
| All resources | Allow: iot:*<br>Allow: iot:Publish<br>Allow: iot:Subscribe<br>Allow: iot:Connect |

   b. Verified the core device's policies. It has all iot publish subcribe, connect permissions.



Policy ARN

A policy ARN uniquely identifies this policy. **Learn more**

```
arn:aws:iot:us-west-2:[redacted]:policy/GreengrassV2IoTThingPolicy
```

Date created

November 11, 2021, 00:17:05 (UTC-0800)

Policy document

The policy document defines the privileges of the request. **Learn more**

Version 1

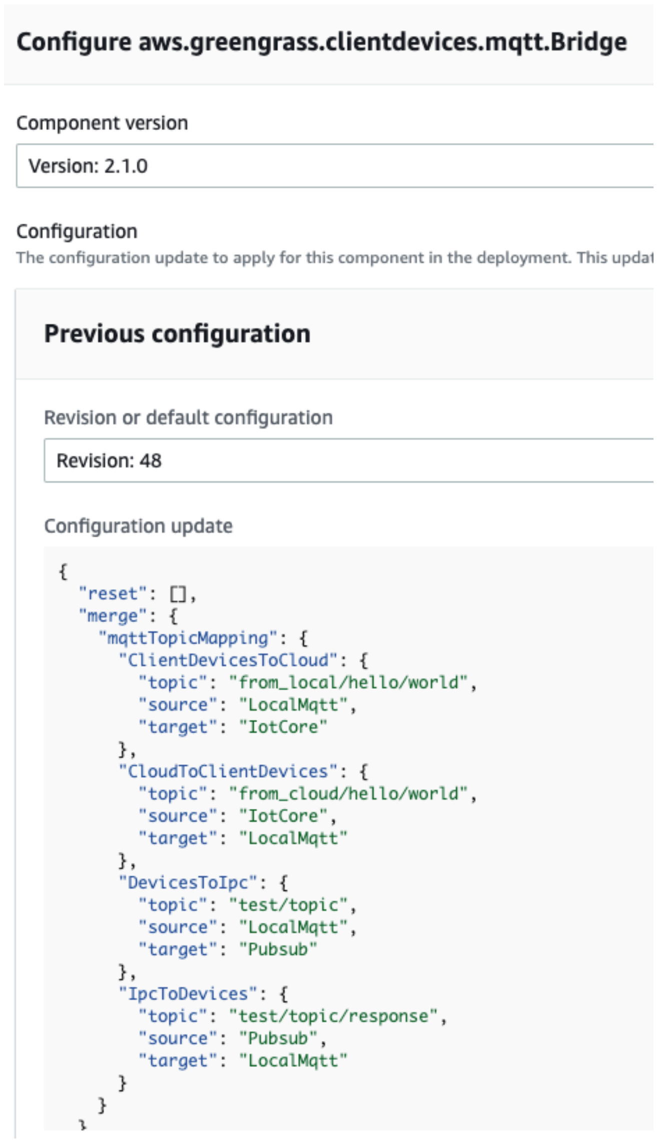
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "greengrass:*"
      ],
      "Resource": "*"
    }
  ]
}
```

   c. Changed the topic filter from cloud discover and changed it to the *.
   d. Changed the topic filter in the event source while creating the component. Mentioned the topic filter as *. So that it can accept the input messages from any published topic on local pub sub.

6. **aws.greengrass.clientdevices.Aut** component is responsible to authenticate client devices and authorize client device actions. The component performs this by matching AWS IoT Core thing names to client devices, and then applying a policy that informs the MQTT broker as to what MQTT operations and topics client devices can act upon. Hence, we have modified the Auth component configurations so that **Greengrass_client_laptop** which is a client thing will be authorized to communicate locally on topics and will also have access to cloud topics “from_local/hello/world” and “from_cloud/hello/world".
   as seen below:

```
{
      "deviceGroups": {
                  "formatVersion": "2021-03-05",
                  "definitions": {
                              "PandaCoreGroup": {
                                          "selectionRule": "thingName: greengrass_client OR thingName:
Greengrass_client_laptop",
                                          "policyName": "PandaCommunicationPolicy"
                              }
                  },
                  "policies": {
                              "PandaCommunicationPolicy": {
                                          "AllowConnect": {
                                                      "statementDescription": "Allow client devices to connect.",
                                                      "operations": [
                                                                  "mqtt:connect"
                                                      ],
                                                      "resources": [
                                                                  "*"
                                                      ]
                                          },
                                          "AllowPublish": {
                                                      "statementDescription": "Allow client devices to publish on.",
                                                      "operations": [
                                                                  "mqtt:publish"
                                                      ],
                                                      "resources": [
                                                                  "mqtt:topic:test/topic"
                                                      ]
                                          },
                                          "AllowSubscribe": {
                                                      "statementDescription": "Allow client devices to subscribe to
panda/topic/response.",
                                                      "operations": [
                                                                  "mqtt:subscribe"
                                                      ],
                                                      "resources": [
                                                                  "mqtt:topicfilter:test/topic/response"
                                                      ]
                                          },
                                          "AllowSubscribeToCloud": {
                                                      "statementDescription": "Allow client devices to subscribe to
the AWS Cloud originated topic",
                                                "operations": [
                                                        "mqtt:subscribe"
                                                ],
                                                "resources": [
                                                        "mqtt:topicfilter:from_cloud/hello/world"
                                                ]
                                        },
                                        "AllowPublishToCloud": {
                                                "statementDescription": "Allow client devices to publish on a
local topic that will be sent to the AWS Cloud",
                                                "operations": [
                                                        "mqtt:publish"
                                                ],
                                                "resources": [
                                                        "mqtt:topic:from_local/hello/world"
                                                ]
                                        }
                                }
                        }
                }
        }
```

7. According to above configuration **Greengrass_client_laptop** should connect to the Moquette MQTT broker and subscribes to the local response topic test/topic/response and also to the topic from_cloud/hello/world that receive messages from AWS IoT Core.
8. However, this is not happening, and core device is not able to subscribe to the topic and getting authentication issue.
9. Also, aws.greengrass.clientdevices.mqtt.Moquette broker is added as component in the core device and which is responsible to handles MQTT messages between client devices and a Greengrass core device.
10. Also, changes are done in **aws.greengrass.clientdevices.mqtt.Bridge** component configuration to allow bidirectional communication. With this configuration, This component can relays MQTT messages between client devices, local AWS IoT Greengrass publish/subscribe, and AWS IoT Core. You use this component to route messages between the Moquette MQTT broker component (LocalMqtt) and either AWS IoT Core (IotCore) or the IPC (Pubsub).



Configure aws.greengrass.clientdevices.mqtt.Bridge

Component version

Version: 2.1.0

Configuration

The configuration update to apply for this component in the deployment. This updat

Previous configuration

Revision or default configuration

Revision: 48

Configuration update

```
{
  "reset": [],
  "merge": {
    "mqttTopicMapping": {
      "ClientDevicesToCloud": {
        "topic": "from_local/hello/world",
        "source": "LocalMqtt",
        "target": "IotCore"
      },
      "CloudToClientDevices": {
        "topic": "from_cloud/hello/world",
        "source": "IotCore",
        "target": "LocalMqtt"
      },
      "DevicesToIpc": {
        "topic": "test/topic",
        "source": "LocalMqtt",
        "target": "Pubsub"
      },
      "IpcToDevices": {
        "topic": "test/topic/response",
        "source": "Pubsub",
        "target": "LocalMqtt"
      }
    }
  }
```

## <u>Part 5 : Develop and deploy lambda function for LambdaOnCoreC2D</u>

Find below the approach to achieve the communication from core device to the client device.

1. Developed the LambdaOnCoreD2C lambda function.
    a. Create the lambda function named LambdaOnCoreC2D using Python 3.7. The Core device already has Python 3.7 installed.
    b. Add the iot publish policy. To publish the MQTT messages from lambda function to the client device.
    c. Lambda function will receive messages in the events and write code to modify the message to add the core device name, timestamp and direction. Topic is “PandaCoreData” and Q0S used is 0.
2. How will lambda receive messages from the AWS IoT console?
    a. We are going to deploy this lambda as a component to the core device.
    b. While creating the component we can specify the event source of the lambda function.
    c. We will specify the event source as “**AWS IoT Core MQTT**” so that lambda function will be able to receive messages from the AWS IoT console.
    d. We will provide the custom topic name; for example “from_cloud/test1”

    e. From IoT test console we will publish to the topic “from_cloud/test1”.
3. How will lambda send messages to the client?
    a. The lambda function will publish the formatted messages to the mqtt broker using a custom topic name, lets say “from_Lambda/test”
    b. Now we will write a code in client device to subscribe to topic “from_Lambda/test”.
    c. In this way whatever messages are sent from lambda will be received by the client device
4. Import lambda function into component. Use the event source specified in the step 2.
5. Deploy component to the core device.
6. Permissions will be similar to the part 3. As in part 3 we have provided the component so that core device can perform bidirectional communication.

## Encountered Problems :

1. Very limited web help for the greengrass version 2 issues. All the resources on internet are mostly for the version 1. Also the documentation of greengrass version 2 is not easy to follow.
2. In the step 1; when I downloaded the latest version on greengrass by using curl command; by default version 2.4.0 got downloaded. It should have downloaded latest version 2.5.0. I was able to perform all the actions till step 3 with greengrass version 2.4.0. However, when I started deploying the lambda version all the dependent components such as **aws.greengrass.clientdevices.Aut** was having hard dependency on greengrass nucleus version 2.5.0. I tried upgrade to the latest version however the In the part 1 the java API already created the core thing with the version 2.4 and hence there was no way to directly upgrade the neucleus version to 2.5.0. I deleted all the configuration with version 2.4.0 and Install greengrass neucleus version 2.5.0 then performed all the steps from start.
3. If lambda function is written in Python version 3.9, we cannot create a component from aws console by directly importing lambda function into component. We have to create a lamda function in version below 3.8 or below then we will be able to select from the dropdown in import lambda section.
4. The core device is not able to subscribe to the topics; Although all permissions are given.
5. The core device is not able to authorize client device; Although all permissions are given.

## Time spent on this assignment: 25-30hrs