Experiment: Edge Computing with AWS IoT Greengrass Foundation

This experiment will start with the basics of AWS Greengrass for Edge & IoT and move to more advanced topics.

During this experiement you will learn how to install the AWS Edge IoT platform AKA Greengrass, connect Edge devices to AWS IoT and Greengrass, and create communication scenarios with those Edge Computing devices and AWS Greengrass.

The experiment will be exercised on an Elastic Compute Cloud (EC2) instance which will be provisioned through AWS CloudFormation. CloudFormation is one of the Infrastructure as Code (IaC) tools in AWS. Amazon Linux is used as operating system and the standard username is ***ec2-user***.

The OpenSource Cloud9 IDE will be installed by you on and EC2 instance. Cloud9 is an integrated development environment (IDE) that lets you write, run, and debug your code with just a browser. It also provides shell access in the browser which is needed for the exercises in this workshop. We will access the IDE through a web browser interface and you will set the username/password for that access. Ensure that you take notes if you change the username and password from the information provided in the experiment. The only way to reset those credentials is to delete our CloudFormation stack and recreate should you forget them.

For the experiment virtual devices are provisioned into your Cloud9 IDE EC2 virtual machine instance. You will find the Python device driver source located in the following files:

- **/home/ec2-user/greengrass-bootcamp/ggad-1/GGBootcampPubSub.py**
- **/home/ec2-user/greengrass-bootcamp/ggad-2/GGBootcampPubSub.py**

The **awscli** is installed and configured on the Cloud9 Integrated Development Environment (IDE). Where the experiment instructions refer to the **awscli** use the one that is installed on the EC2 instance we built with CloudFormation for our Cloud9 IDE.

Edge & IoT Foundation Steps

- Getting Started
- Edge Computing with AWS IoT
- Edge & IoT - AWS Greengrass
- Experiment Architecture
- Launch EC2 Instance with CloudFormation
- Enable logging for AWS IoT
- Connect a thing to AWS IoT

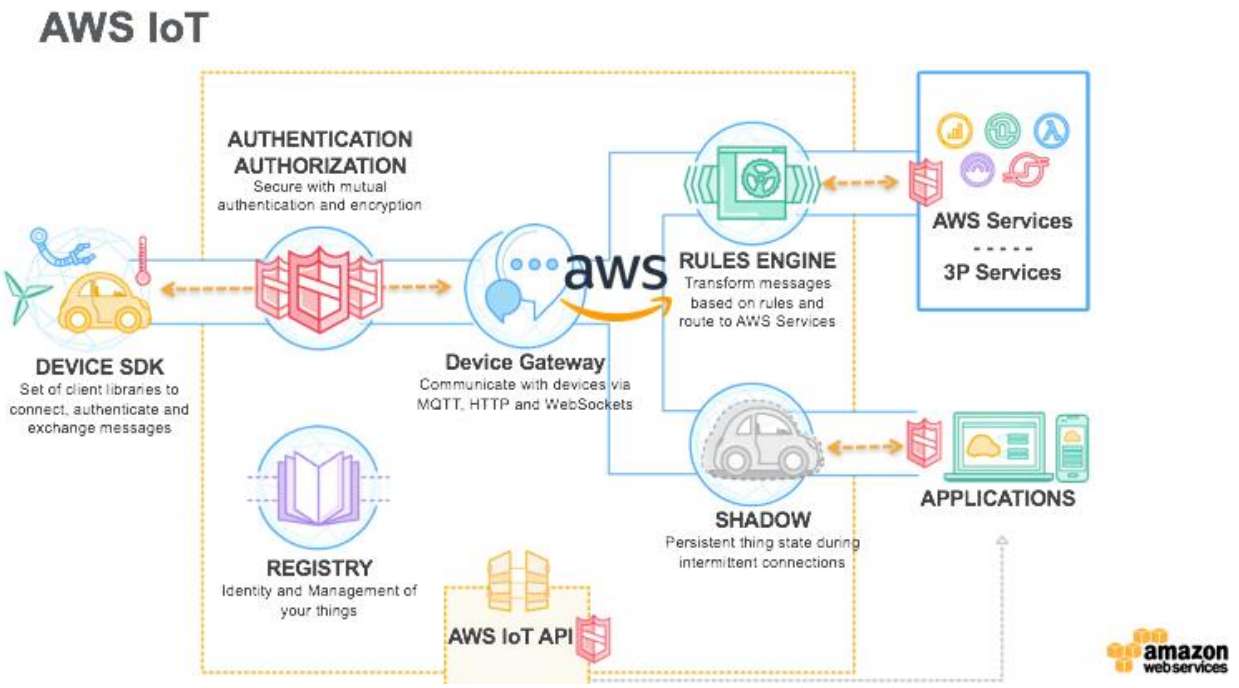
- Send sensor data to AWS IoT

Prerequisites

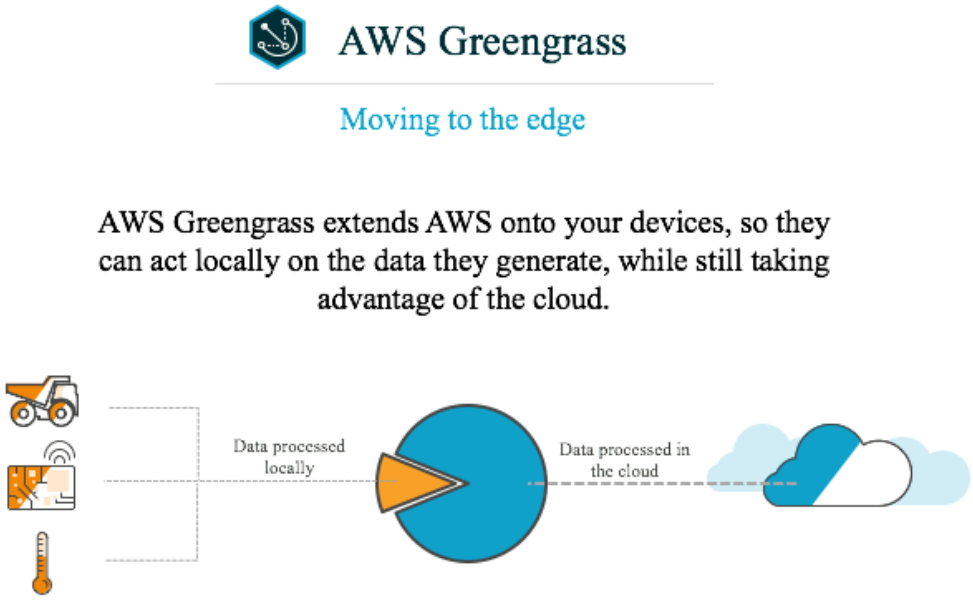
To conduct this experiment, we need:

- AWS Account with admin credentials provided by the instructor
- Your computer (desktop, laptop or tablet)
- Browser (Chrome preferred)
- Basic Linux knowledge

Edge Computing with AWS IoT

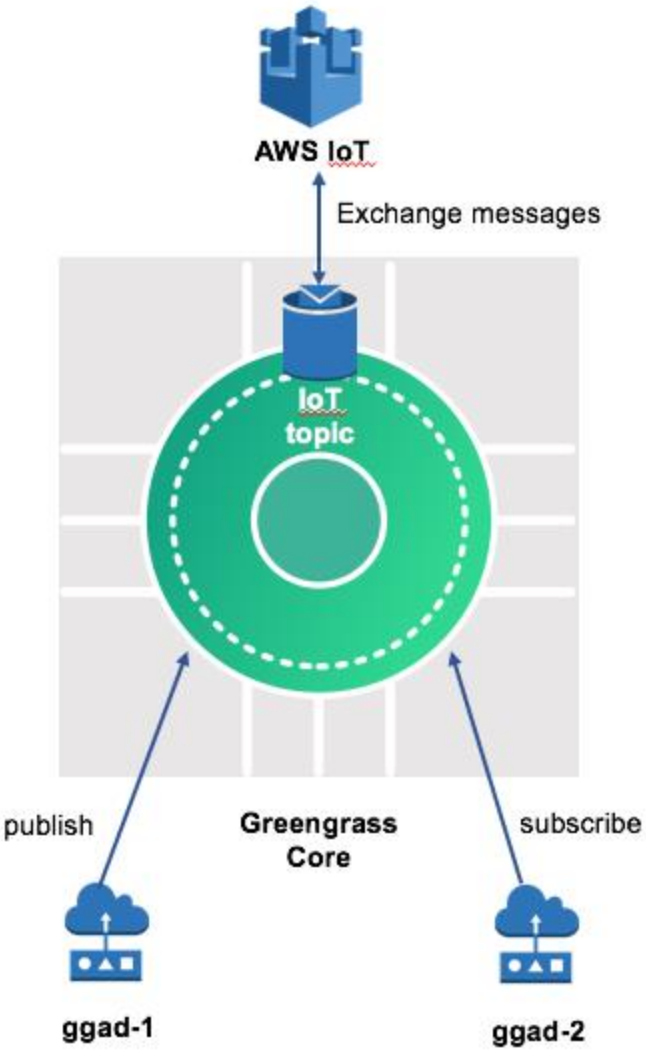


Edge & IoT - AWS Greengrass



Experiment Architecture

You will build this architecture:



AWS Greengrass Requirements

Minimum Hardware:

- CPU single 1GHz
- 128MB RAM
- x86 and Arm
- Linux (Ubuntu, Amazon, Raspbian)

We do provide a dependencies checker (not needed for the workshop) which can be used to verify if your Linux distribution fulfils the requirements to run Greengrass. The Checker can be found at GitHub.

Part 1: Launch EC2 Instance with CloudFormation IaC

We will use a CloudFormation stack to create an EC2 instance which is prepared to run the Greengrass Software. We will use us-east-1 AWS region. This is where we want to launch our Edge Computing and IoT stack.

The link below will automatically redirected to the CloudFormation section of the AWS Console where your stack will be launched.

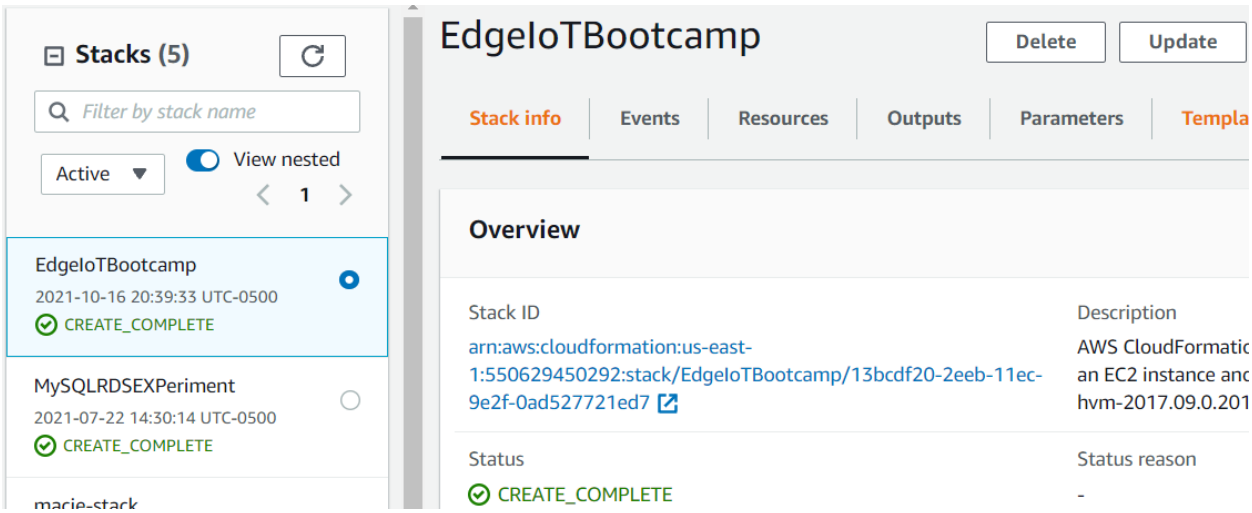
The CloudFormation Stack creates the following resources:

- **S3 Bucket** required for Bulk Provisioning
- **IoT Policy** for provisioning scenarios
- **VPC with public subnet + Security Group** for an EC2 instance
- **EC2 instance** where you do your work
- **Instance Profile** for your EC2 instance
- **IAM Role** required for provisioning scenarios

- Launch CloudFormation stack in us-east-1 (N. Virginia) - https://console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/create/review?stackName=EdgeIoTBootcamp&templateURL=http://s3-eu-central-1.amazonaws.com/aws-workshop-cfn/cfn/cfn-iot-workshop-v1-20190722.json

After you have been redirected to the **Quick create stack** page at the AWS CloudFormation console take the following steps to launch you stack:

1. Parameters
2. 01C9User: Enter a username for your Cloud9 user – **c9user**
3. 02C9Passwd: Enter a password to access Cloud9 – **c9password**
4. (Optional) Select an instance type. The preselected t2.medium is sufficient to execute our experiment
5. (Optional) Select a SageMaker Notebook instance type. The preselected ml.t2.medium is sufficient to execute our experiment
6. Capabilities -> check "I acknowledge that AWS CloudFormation might create IAM resources." at the bottom of the page
7. Select **Create stack**



8. Wait until the complete stack is created. It could take up to 10 minutes for the stack to complete. While you're holding check the Events/Resources/Outputs/Parameters tabs to view the progress and details of the Infrastructure as Code (IaC) deployment

In the **Outputs** section for your stack in the CloudFormation console you find several values for resources that has been created.

You can go back at any time to the Outputs section to find these values.

Access the Cloud9 IDE

Select the Cloud9 URL (you'll find it in the **Outputs** section of your CloudFormation stack) and the authentication dialog for the Cloud9 IDE that was created will be loaded. The HTTP connection to the Cloud9 IDE is encrypted with SSL and a self-signed certificate.

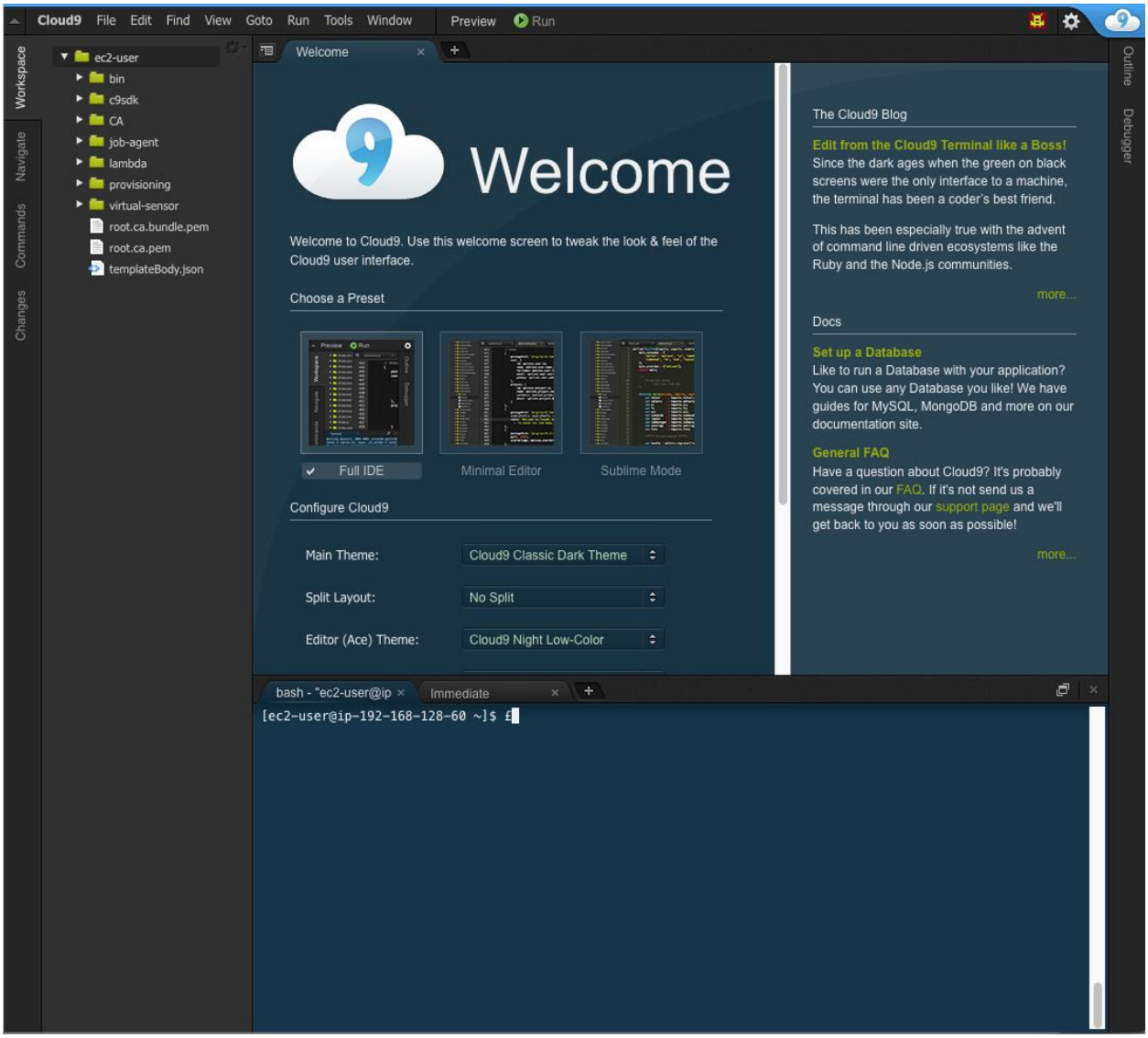
Note: Most web browsers will complain that the issuer of the certificate is unknown and will issue a security warning.

Depending on your browser you need to confirm that you want to visit the page or you need to add an exception to your browser.

- Firefox: Advanced -> Add Exception... -> Confirm Security Exception
- Chrome: ADVANCED -> Proceed to <name_of_your_EC2_instance>
- Safari: Show Details -> ...visit this website.
- Internet Explorer & Edge: Continue to this website (not recommended).

Click on the URL information to view the self-signed certificate issuer information.

After confirming that you want to access the Cloud9 website enter username and password that you choose when the stack was created. You should see a website similar to this one:



Opening a Terminal in Cloud9

The Cloud9 IDE offers a builtin terminal that is used to type commands on the EC2 instance. A terminal can be opened in the following way:

- Click the + in the tab-bar
- New Terminal

Copying Files from/to the EC2 instance

Files could either be uploaded directly with the Cloud9 IDE or indirectly via an S3 bucket.

Cloud9 IDE

- Upload a file: In the menu choose **File -> Upload Local Files...**

- Download a file: **Right-click on the filename -> Download**

S3 Object Storage Bucket

The CloudFormation stack has created a S3 Bucket for you. You can find the bucket name in the outputs section of the CloudFormation stack. A shell variable named "$S3_BUCKET" also holds the name of the bucket.

```
$>  echo $S3_BUCKET
```

Use the S3 bucket to copy files to/from your EC2 instance. The AWS S3 console can be used to up/download files to/from the S3 bucket.

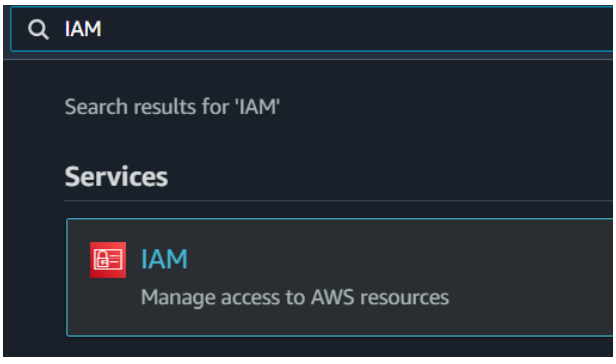
In a terminal in the Cloud9 IDE you will use the awscli to copy files to/from the bucket.

```
# copy files from the bucket
aws s3 cp s3://$S3_BUCKET/my_object .

# copy files to the bucket
aws s3 cp my_file s3://$S3_BUCKET/my_file
```

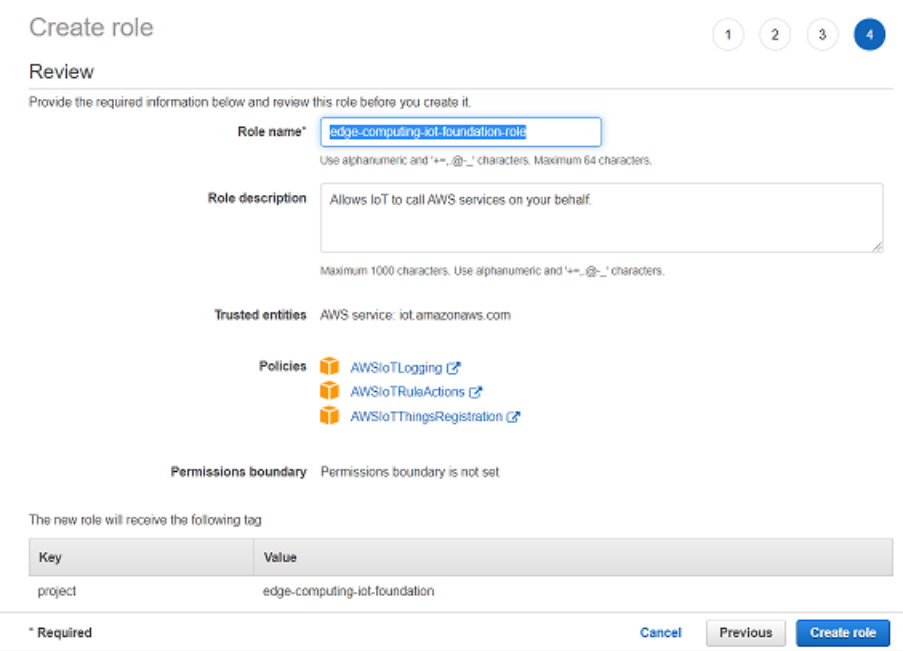
Enable logging for AWS IoT

Go to the AWS IAM console by navigating to the following link - https://console.aws.amazon.com/iam/

Or you can use the service search dialog

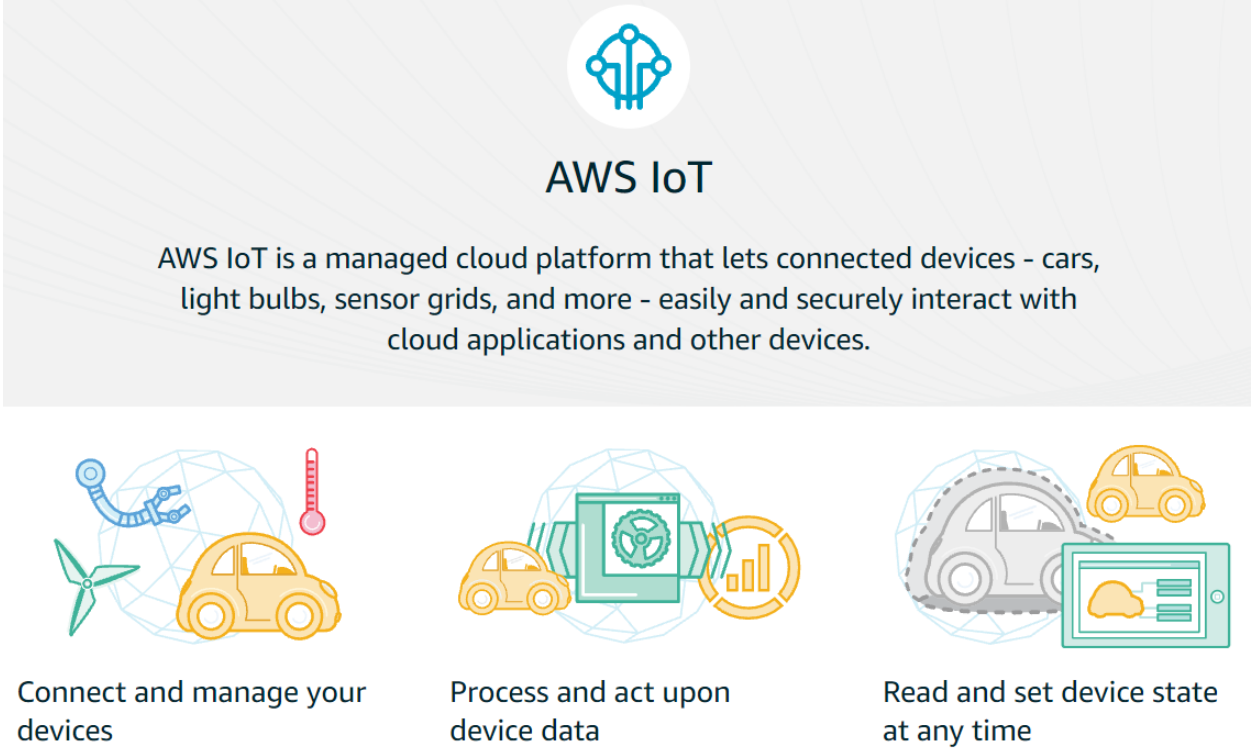


1. Select **Roles**
2. Select **Create role** -> verify AWS service is default selection-> Select **IoT** service
3. Select Use Case – **IoT Allows IoT to call AWS services on your behalf.**
4. Select **Next: Permissions**
5. This will show 3 attached permissions policy:
   - AWSIoTLogging, AWSIoTRuleActions, and AWSIoTThingsRegistration
6. Select **Next: Tags**
7. Enter **project** for Key and **edge-computing-iot-foundation** for tag Value
8. Select **Next: Review**
7. Set a role name - **edge-computing-iot-foundation-role**

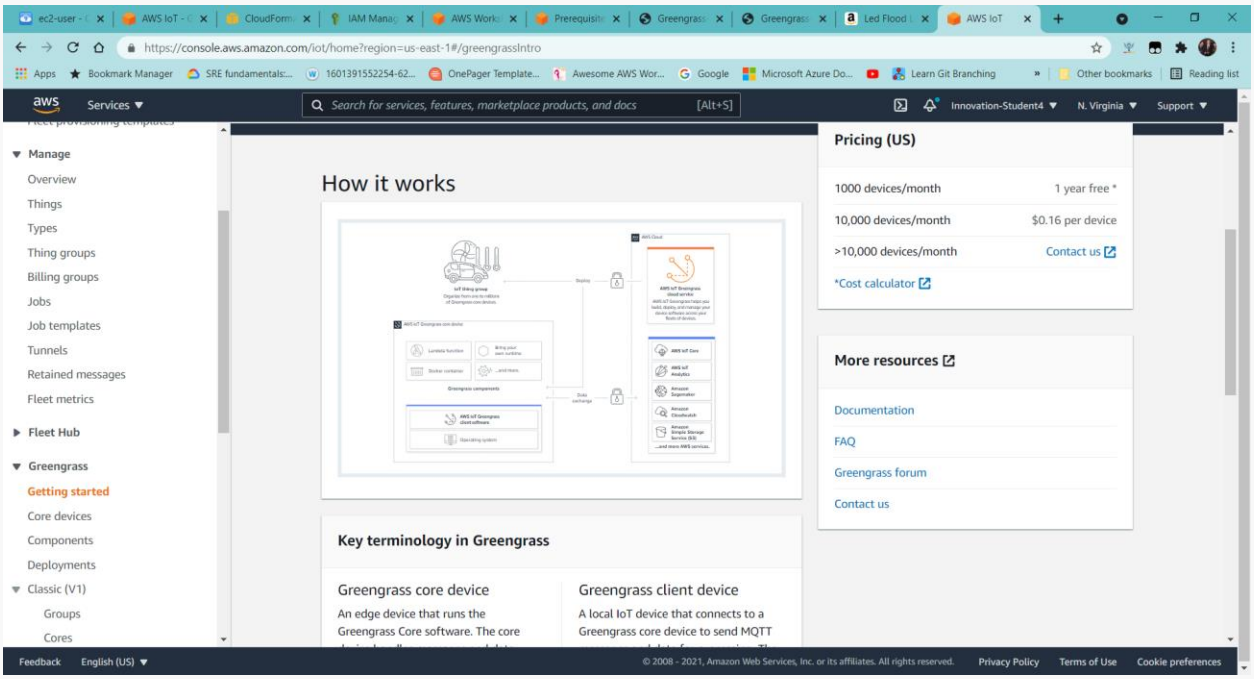


8. Select **Create role**

Go to the AWS IoT Core console - https://console.aws.amazon.com/iot/

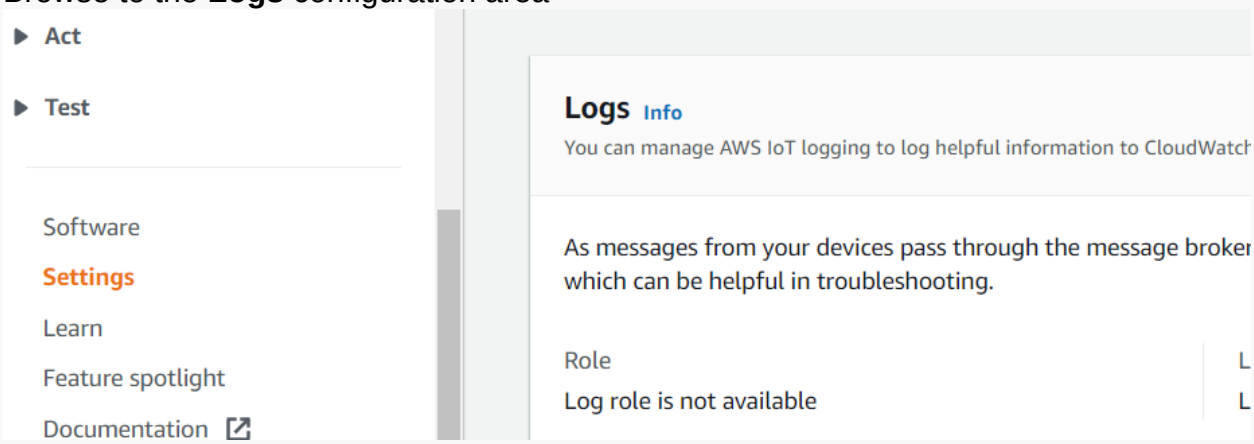


1. Select **Monitor**
2. Note that some views are created for IoT for a variety of metrics
3. Select to expand Greengrass -> select Getting started

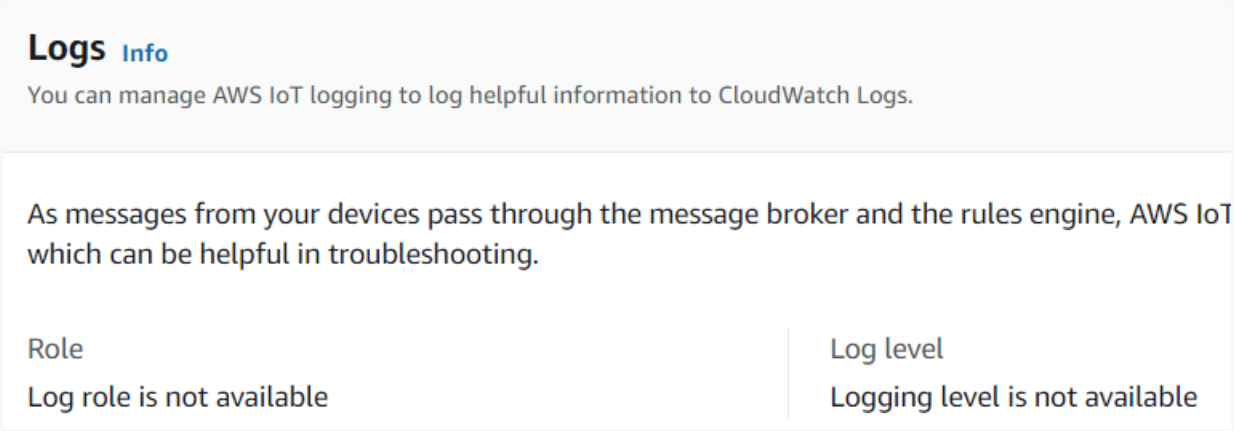


4. Select **Settings**

Browse to the **Logs** configuration area



5. Select **Manage logs**
6. Select **edge-computing-iot-foundation-**role for the logging role
7. Select **Debug (most verbosity)** for the logging level
8. Select **Update**



The log files from AWS IoT are sent to **Amazon CloudWatch**. The AWS console can be used to look at these logs.

Part 2: Connect an Edge Device/Thing to AWS IoT

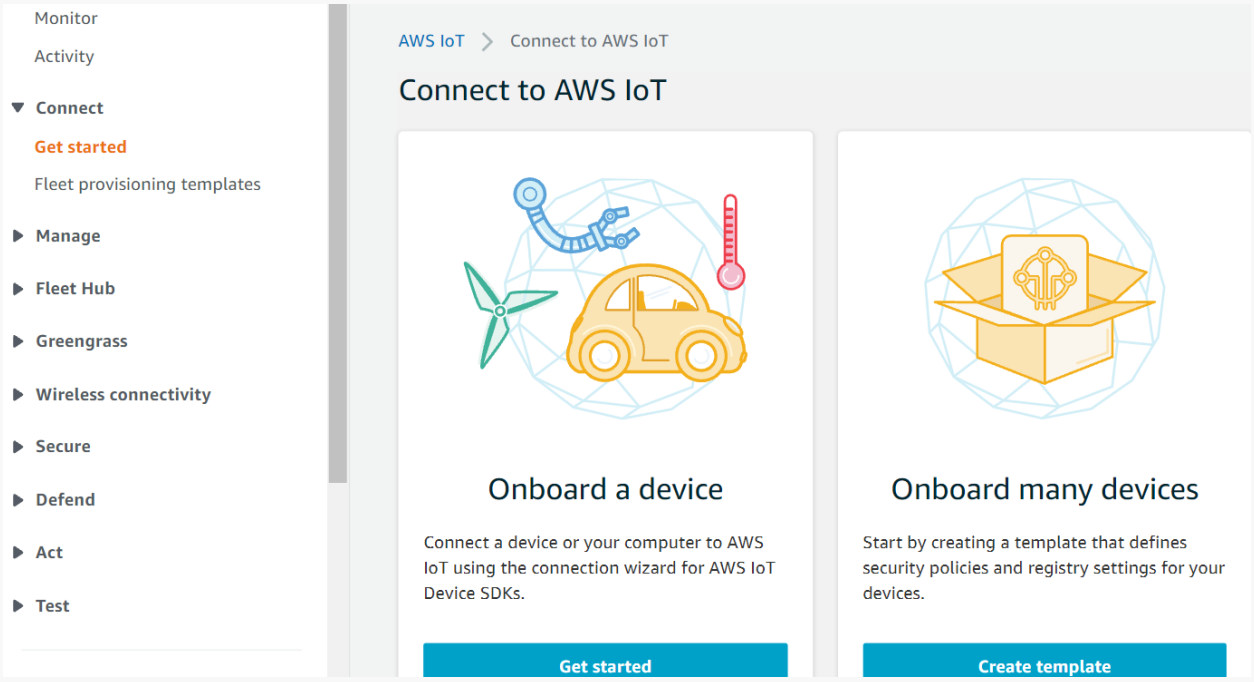
Utilizing a Greengrass Core or GGAD:

Connecting a device to AWS IoT, a Greengrass Core to AWS IoT or a Greengrass Aware Device (GGAD) to a core works in the same way by using X.509 certificates and connectivity information. At first we will connect a device to AWS IoT.

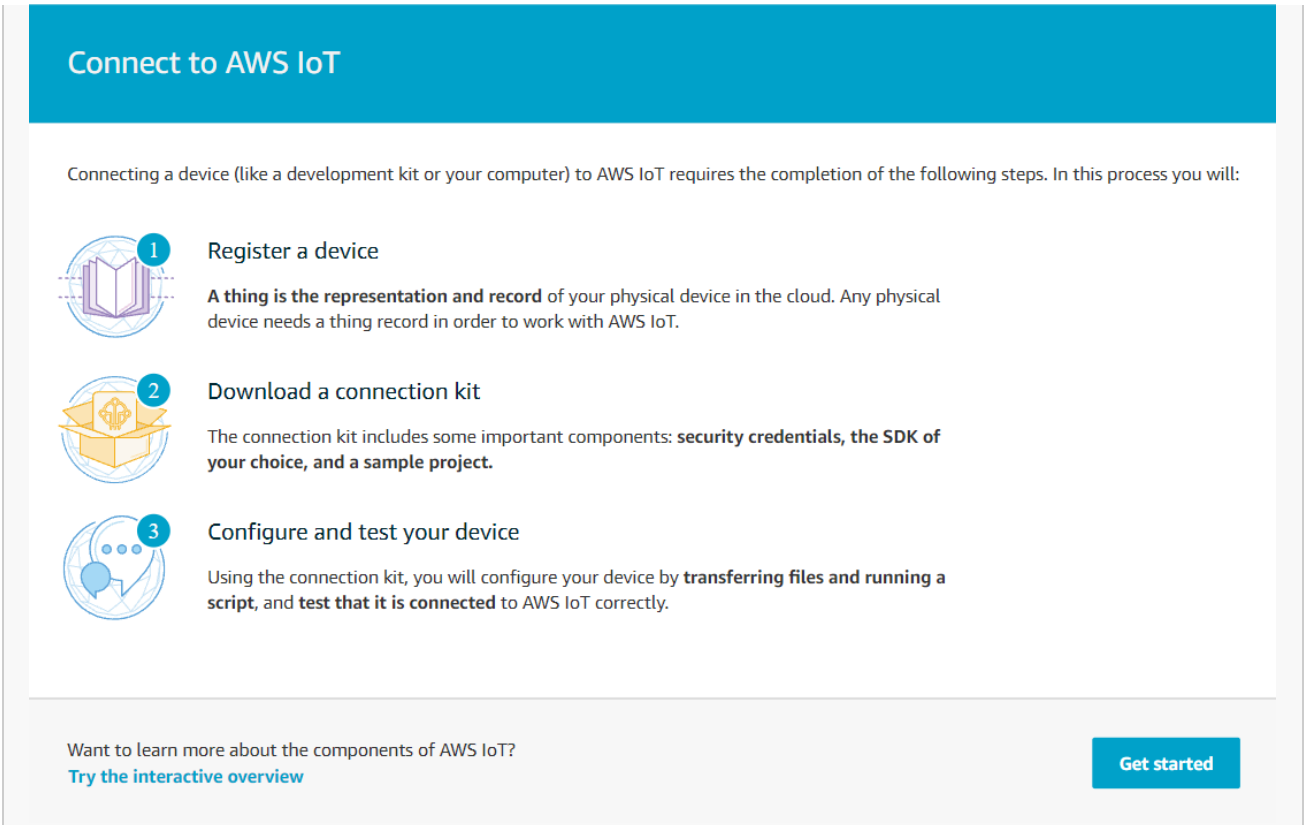
We'll utilizing the quickstart to provision a **thing** through the AWS IoT console. A zip file will be provided containing keys and certificates and a script to install further required software.

Go to the AWS IoT Core console if you've closed the previous IoT console view

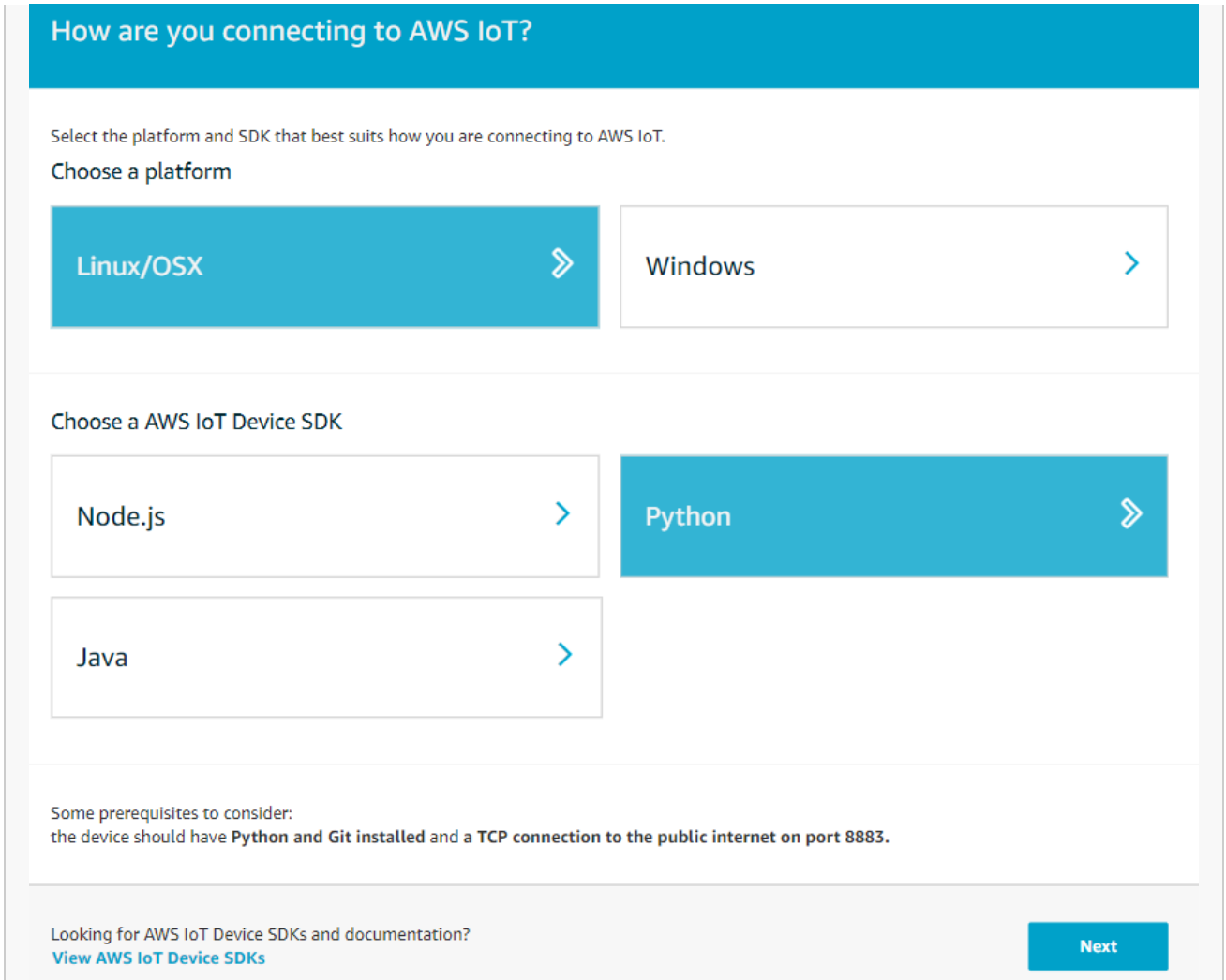
1. Expand **Connect**
2. Select **Get Started**
3. Onboard a device -> **Get started**



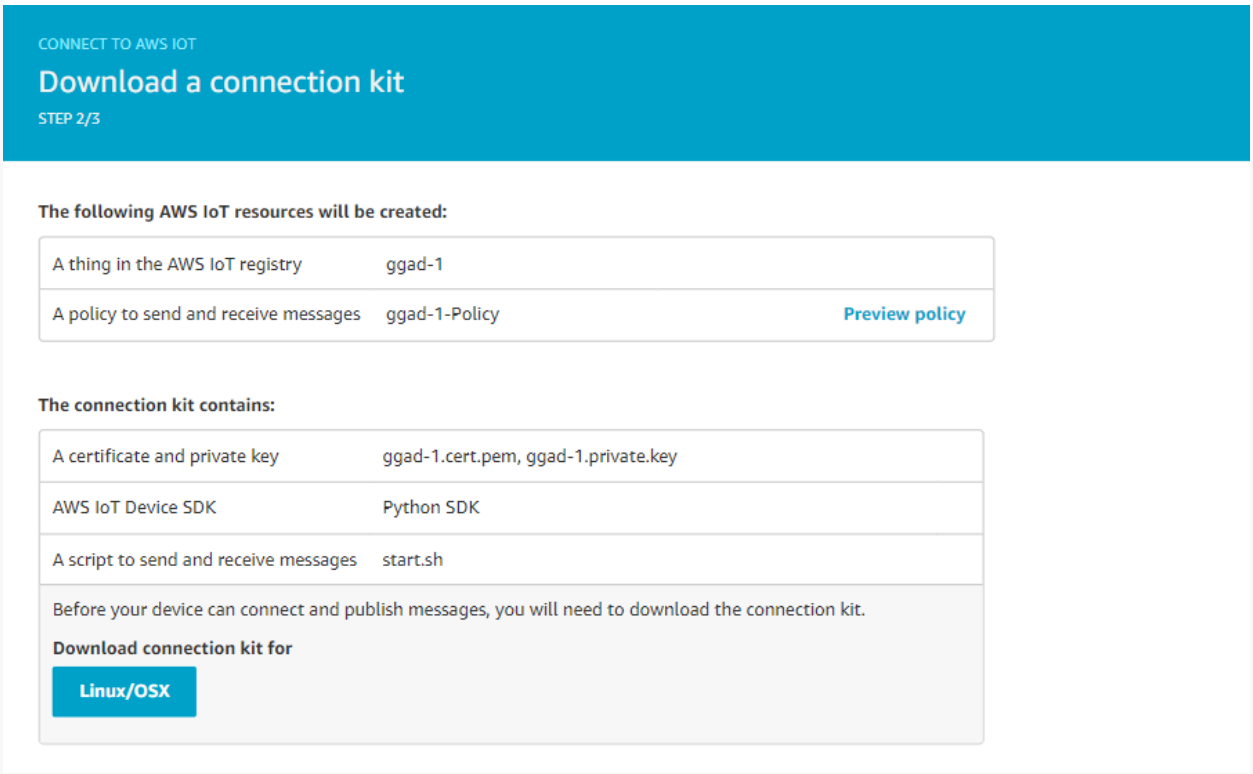
4. Connect to AWS IoT -> **Get started**



5. Choose Linux/OSX and Python -> Next



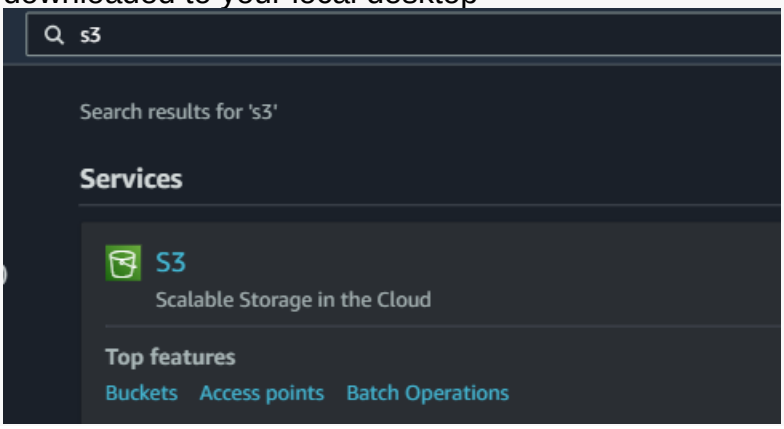
6. Register a thing Name **ggad-1**
7. Select **`Show optional configuration (this can be done later)`**
8. Enter **Manufacturer** for the Attribute key and **Cisco** for the Value -> Select **Next step**
7. Download a connection kit -> Linux/OSX -> A file "connect_device_package.zip" will be download.

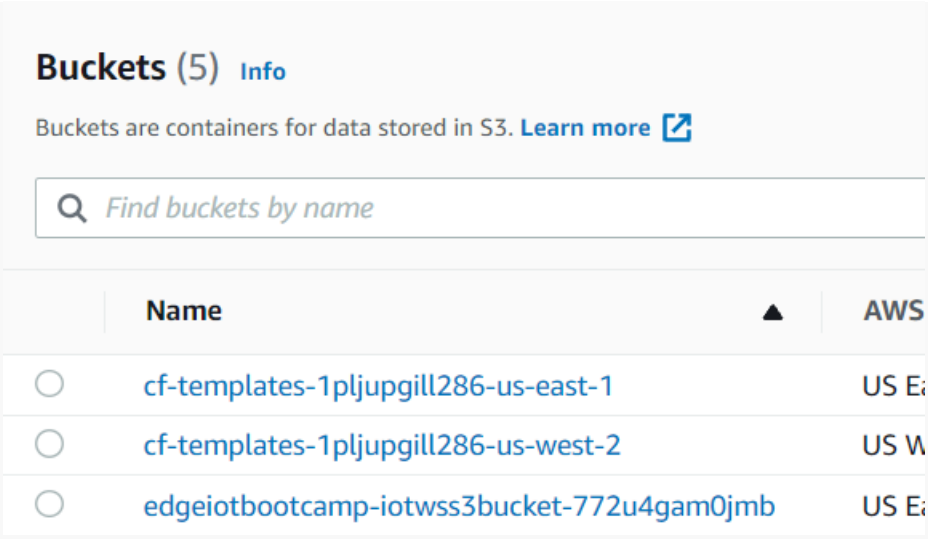


10. Select the Preview policy link to allow you to view, noting that it displays information about Java, Python and NodeJS.
11. Proceed to the next step after download is finished when the button becomes un-grayed.
11. Select **Next step**

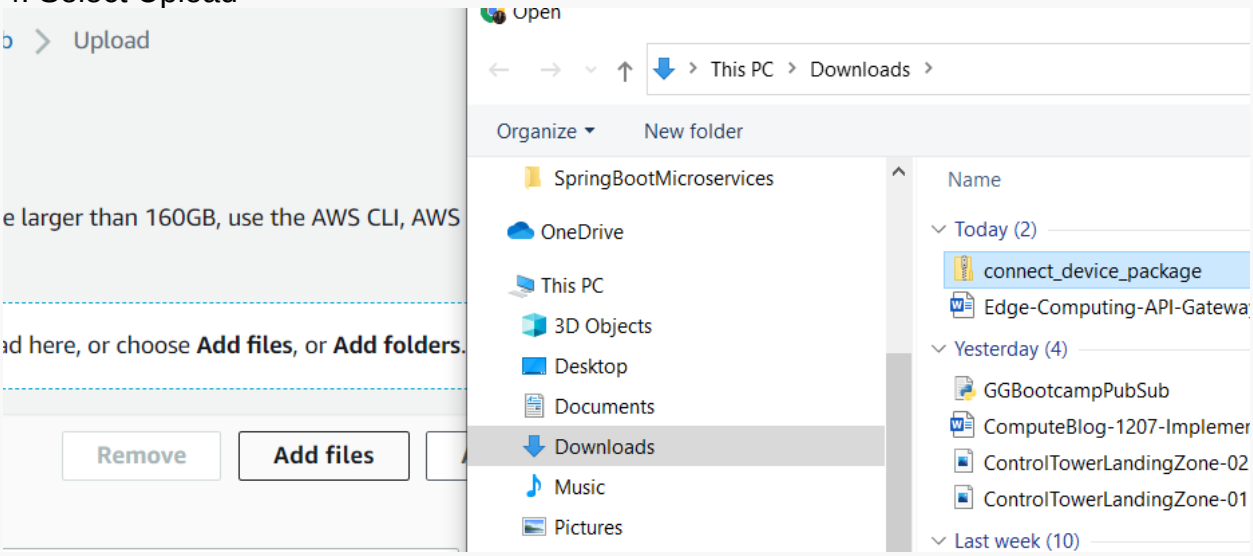
Add the Device Zip to your Experiment S3 Bucket

1. Copy (use S3/Cloud9 IDE as mentioned above) the file onto the EC2 instance into the directory /home/ec2-user/greengrass-bootcamp/ggad-1/
2. Navigate to S3 -> Buckets in the Management Console and upload the file you downloaded to your local desktop





3. Select the edgeiotbootcamp… folder
4. Select Upload



5. Select **Add files** and select the **connect_device_package.zip** that you downloaded.
6. Select **Upload**

Copy the File from S3 to the Cloud9 IDE

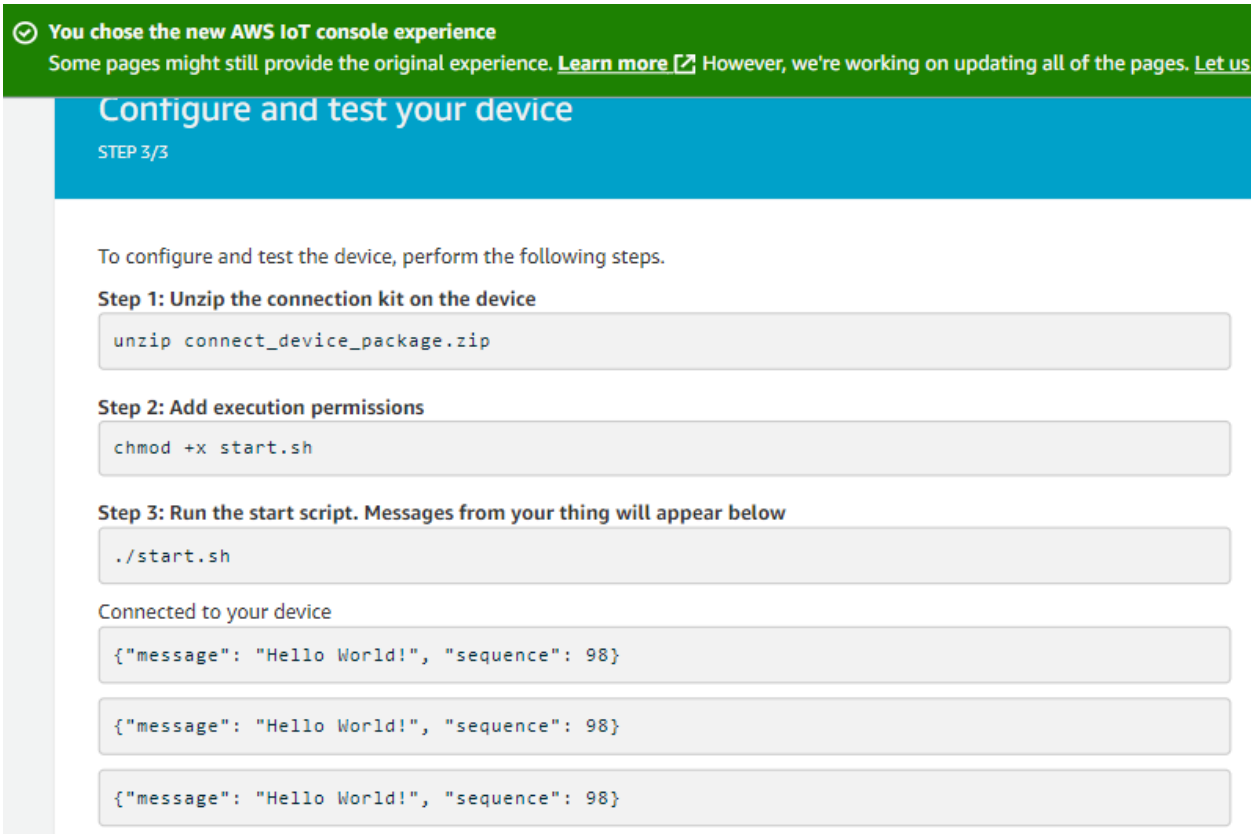
1. Make sure you're in the device folder
 $> **cd ~/greengrass-bootcamp/ggad-1/**
2. Copy the file with the AWS CLI
[ec2-user ggad-1]$ **aws s3 cp s3://$S3_BUCKET/connect_device_package.zip .**

```
download: s3://edgeiotbootcamp-iotwss3bucket-
772u4gam0jmb/connect_device_package.zip to ./connect_device_package.zip

2. Verify the file
[ec2-user@ip-192-168-128-124 ggad-1]$ ls
connect_device_package.zip  GGBootcampPubSub.py  gg_discovery_api.py

3. Unzip the Device Package
$> unzip connect_device_package.zip
Archive:  connect_device_package.zip
  inflating: ggad-1.private.key
  inflating: ggad-1.public.key
  inflating: ggad-1.cert.pem
  inflating: start.sh

4. Make start.sh executable
[ec2-user ggad-1]$ chmod 755 start.sh

5. Execute the start.sh script to deploy your device
[ec2-user ggad-1]$ sudo ./start.sh
```

Sometimes when launching the script **start.sh** errors are encountered:

- start.sh does not start: On some systems an explicit shebang line is needed. So add a first line "#!/bin/bash" or "#!/bin/sh" to start.sh
- start.sh installs the AWS IoT Python SDK in the local directory. This can result in "error: could not delete... : Permission denied" error. The reason for that is that the installation procedure tries to delete files in global directories. Simply fire up start.sh again or try to launch start.sh as root.

You should see messages arriving in the AWS IoT console.



By seeing messages arriving in the AWS IoT console you have successfully sent data from a device to AWS IoT.

Part 3: Edge Computing and IoT Structured Data Processing

You will have observed that in Part 1 of our experiment, we sent unstructured data to our message processing engine in AWS IoT with Greengrass Device Connection. In this section you will send sensor data in JSON format to AWS IoT and watch the incoming data in the MQTT client which is built into the AWS IoT console.

For sending random sensor data we do will use the following scripts:

- GGBootcampPubSub.py
- gg_discovery_api.py

Use your preferred method to view the files. This could be the **vi** or **nano** text editors, **cat** or **more** unix utilities, or any other mechanism.

Note that gg_discovery_api.py is a python class using the Greengrass Discovery API. This will return the response document for a given thing which could be any sensor, device, CDN or other edge virtual or physical device.

Note: To read more review
http://docs.aws.amazon.com/greengrass/latest/developerguide/gg-discover-api.html

The script "start.sh" in the directory "ggad-1" must be modified to call GGBootcampPubSub.py instead of the example script from the AWS IoT Python SDK:

1. Modify "start.sh" to start "GGBootcampPubSub.py" instead of "aws-iot-device-sdk-python/samples/basicPubSub/basicPubSub.py"

Replace "aws-iot-device-sdk-python/samples/basicPubSub/basicPubSub.py"
with
"GGBootcampPubSub.py"

The resulting line in "start.sh" should look like:

```
python GGBootcampPubSub.py -e
<YOUR_ENDPOINT>.<AWS_REGION>.amazonaws.com -r root-CA.crt -c ggad-
1.cert.pem -k ggad-1.private.key
```

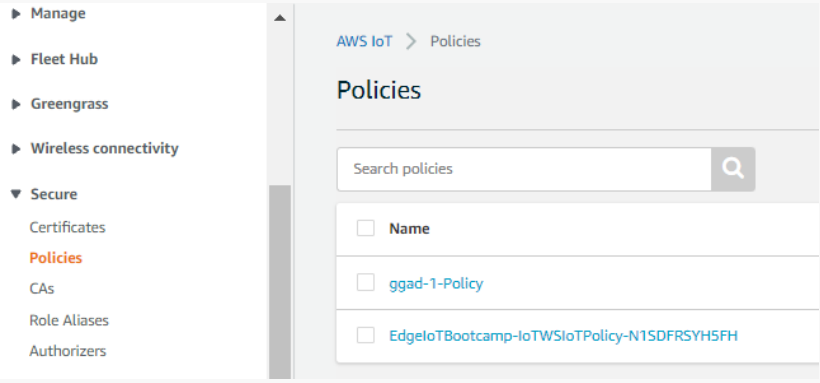
For example:
python GGBootcampPubSub.py -e a2sbyngxd290ib-ats.iot.us-east-1.amazonaws.com -r root-CA.crt -c ggad-1.cert.pem -k ggad-1.private.key

Modify IoT Policy

By default an IoT policy with least privileges is created. For this experiment we will use a more open IoT policy for simplicity. This is **not recommended** in production environments.

Go to the AWS IoT Core console

1. Expand **Secure**
2. Select **Policies**



3. Click the **ggad-1-Policy**
4. Select **Edit policy document**

5. Replace the existing policy with the following policy:

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

6. Select **Save as new version**

Subscribe to the MQTT client built into the AWS IoT console to see the messages which will be sent later on. Subscribe to the topics:

- sdk/test/Python
- $aws/events/#

Go to the AWS IoT Core console

1. Expand **Test**
2. Select **MQTT test client**
3. Subscribe to a topic
4. Topic filter: **sdk/test/Python** -> Select **Subscribe**
4. Subscribe to a topic
3. Topic filter: **$aws/events/#** -> Select **Subscribe**

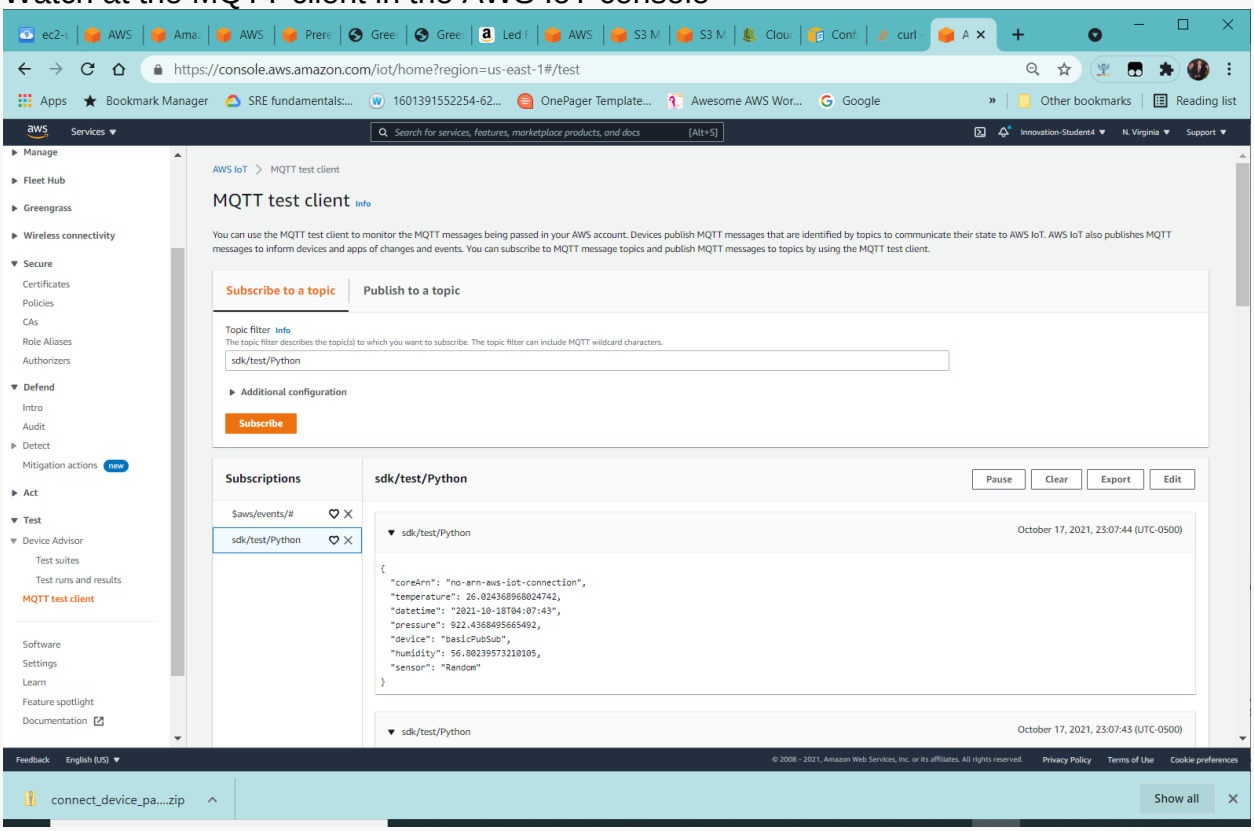
Now it is time to send sensor data to AWS IoT.

In a Cloud9 terminal:

Launch our Edge device
$ ggad-1> **sudo ./start.sh**

At the AWS IoT Core console

Watch at the MQTT client in the AWS IoT console



In a Cloud9 terminal:

Stop the script "start.sh" by pressing Ctrl+C

Provision a second device **ggad-2** the same way that you did for the device ggad-1.

Note: ***!!! Ensure it is named ggad-2, not ggad-1 and copy the newly created file connectdevicepackage.zip to the directory /home/ec2-user/greengrass-bootcamp/ggad-2/ !!!***

The scripts are acting as MQTT clients. A MQTT client uses a clientId when connecting to AWS IoT and the clientId must be unique.

By default the scripts are connecting with the clientId "basicPubSub". To make the clientId unique we want them to connect as "ggad-1" and "ggad-2".

As the scripts are taking command line arguments it is simple to make them send another clientId with the parameter "--clientId". This parameter must be added in "start.sh".

Change the clientId for "ggad-1" on your device:

Current line in start.sh will look similar to this one:

```
python GGBootcampPubSub.py -e <YOUR_ENDPOINT>.<AWS_REGION>.amazonaws.com -r root-CA.crt -c ggad-1.cert.pem -k ggad-1.private.key
```

Line after modification similar to this one:

```
python GGBootcampPubSub.py -e <YOUR_ENDPOINT>.<AWS_REGION>.amazonaws.com -r root-CA.crt -c ggad-1.cert.pem -k ggad-1.private.key --clientId ggad-1
```

To test if the clientId setting works go to the AWS IoT console and subscribe to

```
$aws/events/#
```

Start the script and check in the AWS IoT console if it connects with the modified clientId.

Change start.sh script for "ggad-2" to use "ggad-2" as clientId.

Start your Edge devices (ggad-1 and ggad-2)

View in the MQTT Test Client in the IoT Console