AWS IoT Specific

Nilesh Devdas

Logistics & Special notes from Vodafone L&D

Trainer : Nilesh Devdas

Training timing

- Time 10:00 am to 06:30 pm
- Two 15 min break & Lunch Break 30-45 mins

Special Notes

- Participants want to leave early/come late need manager approval as well as approval from training team i.e. Ravi Kant Pandey
- Please make it a point to sign the attendance sheet when it is circulated in the class. If your signature is not on the attendance sheet, you will be considered as a “No Show” and a mail will be sent to your manager, Service line lead and Service line training focal.

For any queries, issues or suggestion please contact.

Mr. Vipin Ahuja

+91-9607933302

SPOC Details

Exposure to Node Programming

Hands on experience of messaging and Http

Hands on experience in AWS Cloud basics

Knowledge of IoT basics

Knowledge of security certificates

If you do-not meet the prerequisites you may not be able to take full advantage of the class.

Prerequisites

Objectives

- AWS IoT Core
- AWS IoT Greengrass
- AWS IoT Integration with nodejs
- AWS IoT sdk for nodejs
- IoT Device Gateway
- Device Registry/Shadow Management
- Device Shadow integration using nodejs and real device
- AWS IoT Rule Engine
- AWS IoT Device Defender
- AWS IoT Message Broker
- AWS IoT Analytics
- More focus on real device based training with Adruino or any real device and Usecase using nodejs as programming language
- Develop Rule Engine using Real devices, MQTT, nodejs, lambda, Kinesis Streaming, RedShift Cluster

AWS IoT Core
AWS IoT Greengrass
AWS IoT Integration with nodejs
AWS IoT sdk for nodejs
IoT Device Gateway
Device Registry/Shadow Management
Device Shadow integration using nodejs and real device
AWS IoT Rule Engine
AWS IoT Device Defender
AWS IoT MessageBroker
AWS IoT Analytics
More focus on real device based training with Adruino or any real device and Usecase using nodejs as programming language
Develop Rule Engine using Real devices, MQTT, nodejs, lambda, Kinesis Streaming, RedShift Cluster

Course Agenda

Introduction to aws and IoT
AWS IoT Core concepts
Managing devices
Device security
Creating certificates
Attaching policy
Testing a device
Device metrics
Device monitoring
Raspberry pi on boarding
Device shadows
Arduino on boarding
AWS IoT Rule Engine
AWS IoT Device Defender Hands on use-cases
AWS IoT Message Broker

Day-1

Introduction

Register aws green grass

Lambda runtime

Shadows implementation

Message manager

Group management

Discovery service

Local resource access

Local secrets manager

Connectors with built-in integration with services, protocols, and softwargr

Day-2

AWS IoT Integration with node-js

AWS IoT sdk for nodejs

IoT Device Gateway

Device Shadow integration using node-js and real device

AWS IoT Analytics

More focus on real device based training with Arduino or any real device and

Use case using NodeJS as application

Develop Rule Engine using

Real devices, MQTT demo

Analytics demo

Day -3

Feedback QR Code

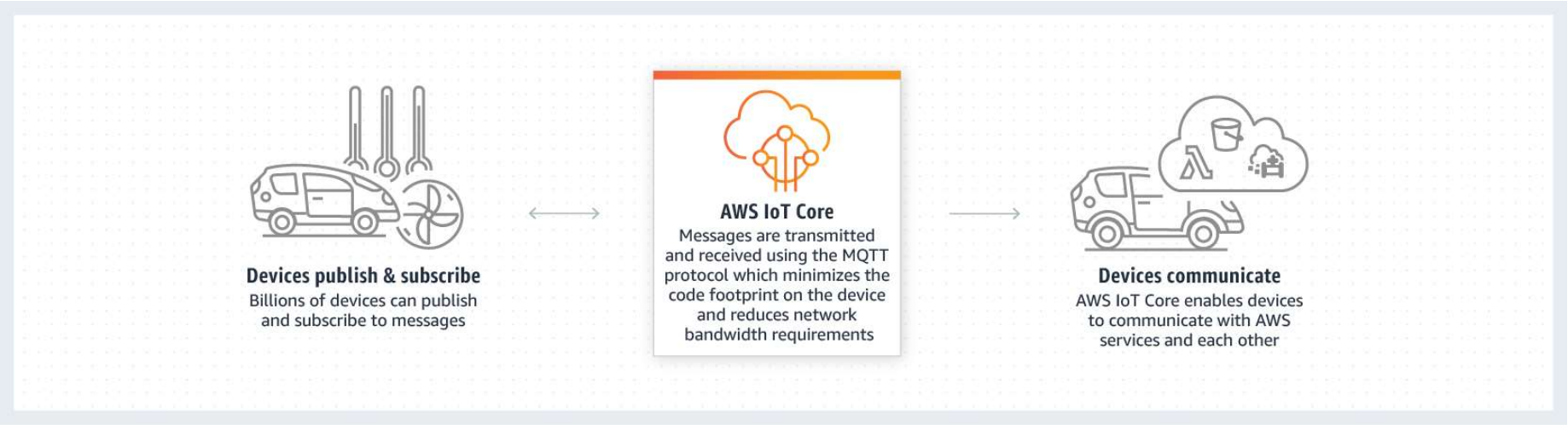
Introduction to AWS & IoT

T

AWS IoT

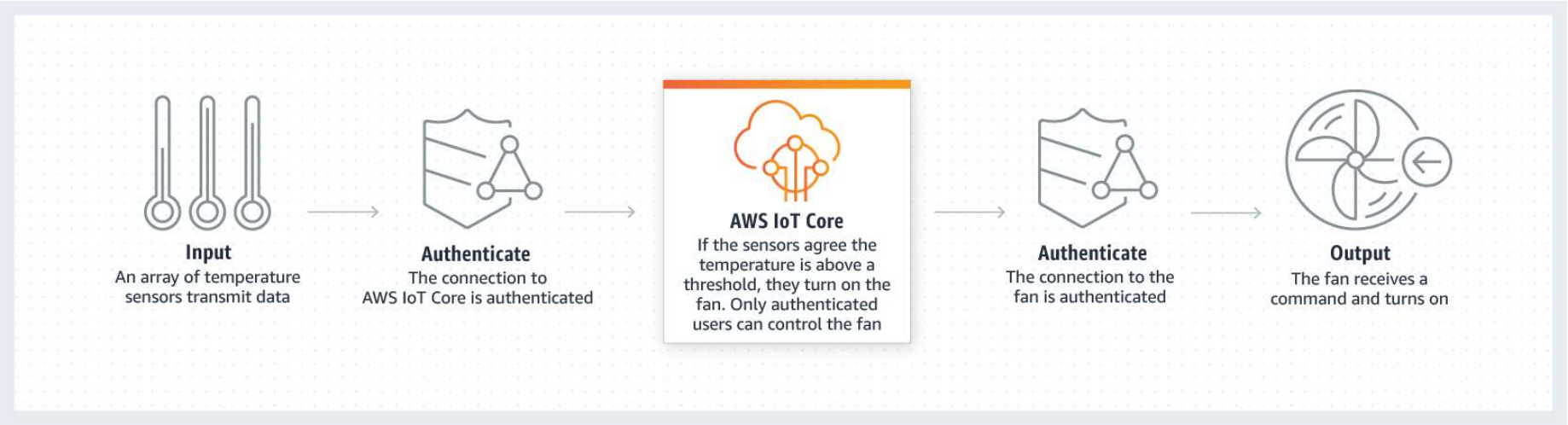
What is AWS IoT

AWS IoT provides secure, bi-directional communication between Internet-connected devices such as sensors, actuators, embedded micro-controllers, or smart appliances and the AWS Cloud. This enables you to collect telemetry data from multiple devices, and store and analyze the data. You can also create applications that enable your users to control these devices from their phones or tablets.



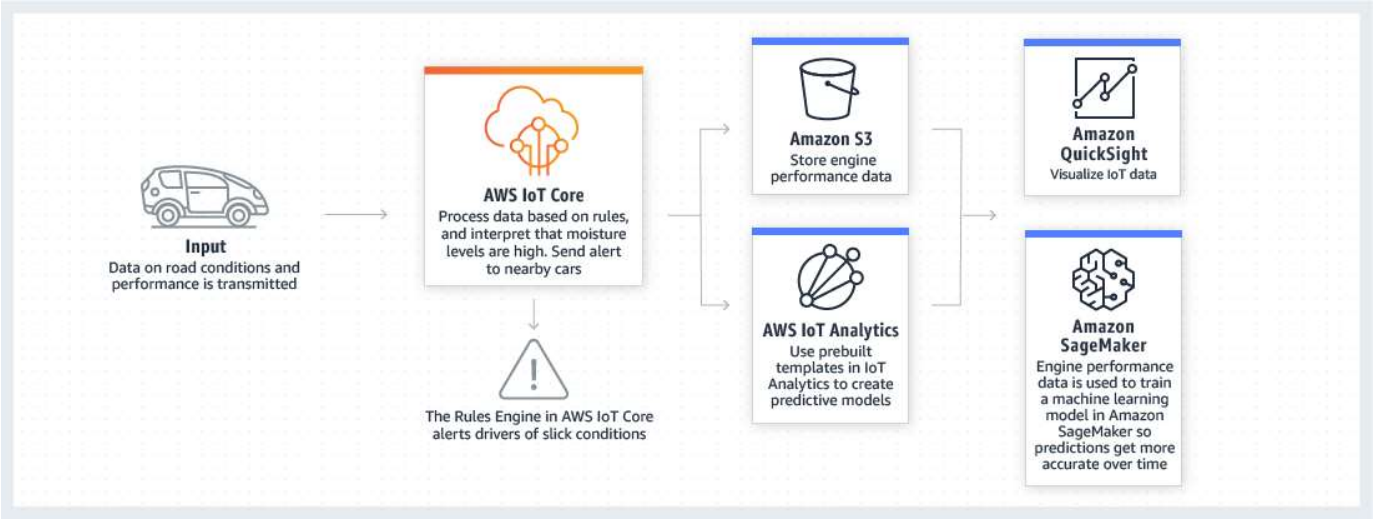
How does AWS IoT Core work?

- AWS IoT Core allows you to easily connect devices to the cloud and to other devices.
- AWS IoT Core supports HTTP, WebSockets, and MQTT



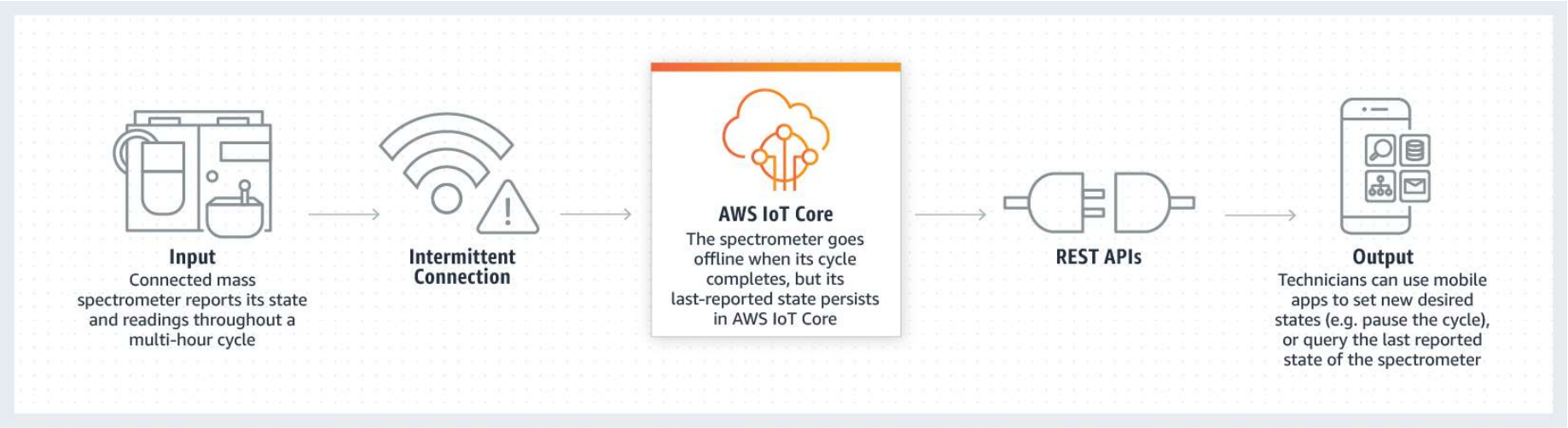
AWS Secure device connections and data

- AWS IoT Core provides authentication and end-to-end encryption throughout all points of connection, so that data is never exchanged between devices and AWS IoT Core without proven identity.
- You can secure access to your devices and applications by applying policies with granular permissions.



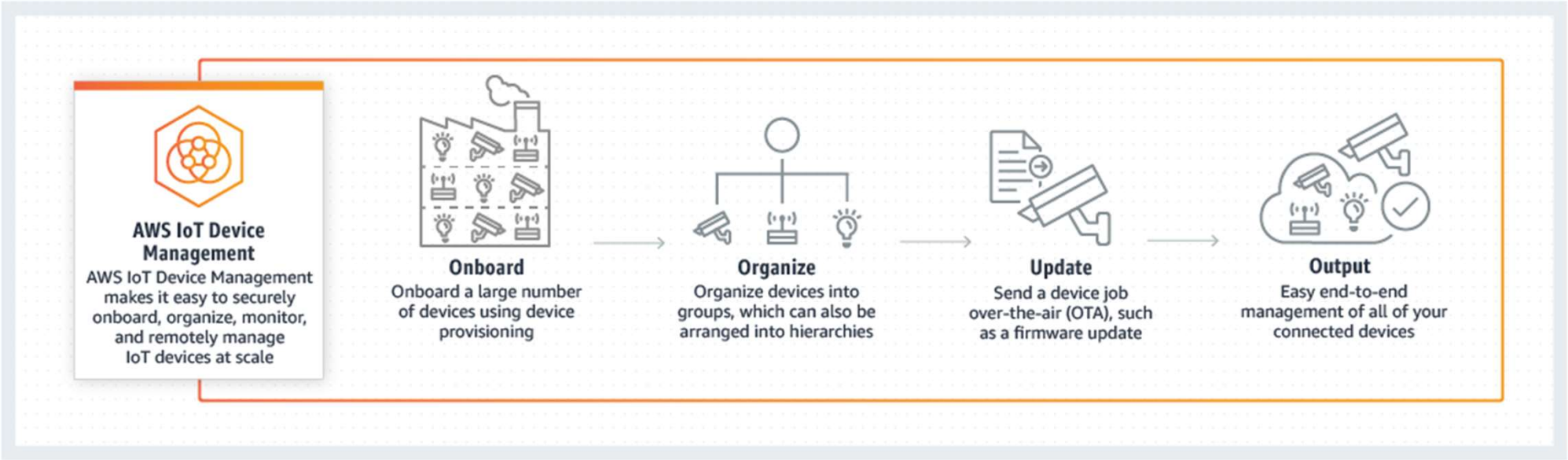
Process and act upon device data

- You can filter, transform, and act upon device data on the fly, based on business rules you define.
- You can update your rules to implement new device and application features at any time.
- AWS IoT Core makes it easy to use AWS services like AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, Amazon CloudWatch, and Amazon Elasticsearch Service for even more powerful IoT applications.



Read and set device state at any time

- AWS IoT Core stores the latest state of a connected device so that it can be read or set at anytime, making the device appear to your applications as if it were online all the time.
- This means that your application can read a device's state even when it is disconnected, and also allows you to set a device state and have it implemented when the device reconnects.



Managing devices

Manage & Onboard Devices

AWS IoT Core

Concepts

AWS IoT Components

Device gateway

- Enables devices to securely and efficiently communicate with AWS IoT.

Message broker

- Provides a secure mechanism for devices and AWS IoT applications to publish and receive messages from each other. You can use either the MQTT protocol directly or MQTT over WebSocket to publish and subscribe. You can use the HTTP REST interface to publish

Rules engine

- Provides message processing and integration with other AWS services. You can use an SQL-based language to select data from message payloads, and then process and send the data to other services, such as Amazon S3, Amazon DynamoDB, and AWS Lambda. You can also use the message broker to republish messages to other subscribers.

AWS IoT Components

Security and Identity service

- Provides shared responsibility for security in the AWS Cloud. Your devices must keep their credentials safe in order to securely send data to the message broker.
- The message broker and rules engine use AWS security features to send data securely to devices or other AWS services.

Registry

- Organizes the resources associated with each device in the AWS Cloud. You register your devices and associate up to three custom attributes with each one.

Group registry

- Groups allow you to manage several devices at once by categorizing them into groups.
- Groups can also contain groups—you can build a hierarchy of groups.
- Any action you perform on a parent group will apply to its child groups, and to all the devices in it and in all of its child groups as well.
- Permissions given to a group will apply to all devices in the group and in all of its child groups

AWS IoT Components

Device shadow

- A JSON document used to store and retrieve current state information for a device.

Device Shadow service

- Provides persistent representations of your devices in the AWS Cloud. You can publish/retrieve updated state information to a device's shadow, and your device can synchronize its state when it connects.

Device Provisioning service

- Provision devices using a template that describes the resources required for your device: a thing, a certificate, and one or more policies.
- A thing is an entry in the registry that contains attributes that describe a device.
- Devices use certificates to authenticate with AWS IoT. Policies determine which operations a device can perform in AWS IoT.

AWS IoT Components

Jobs service

- Allows you to define a set of remote operations that are sent to and executed on one or more devices connected to AWS IoT.

Custom Authentication

- You can define custom authorizers that allow you to manage your own authentication and authorization strategy using a custom authentication service and a Lambda function

Accessing AWS IoT (Available interfaces)

AWS Command Line Interface (AWS CLI)

- Run commands for AWS IoT on Windows, macOS, and Linux.
- These commands allow you to create and manage things, certificates, rules, and policies

AWS IoT API

- Build your IoT applications using HTTP or HTTPS requests.
- These API actions allow you to programmatically create and manage things, certificates, rules, and policies.

AWS SDKs

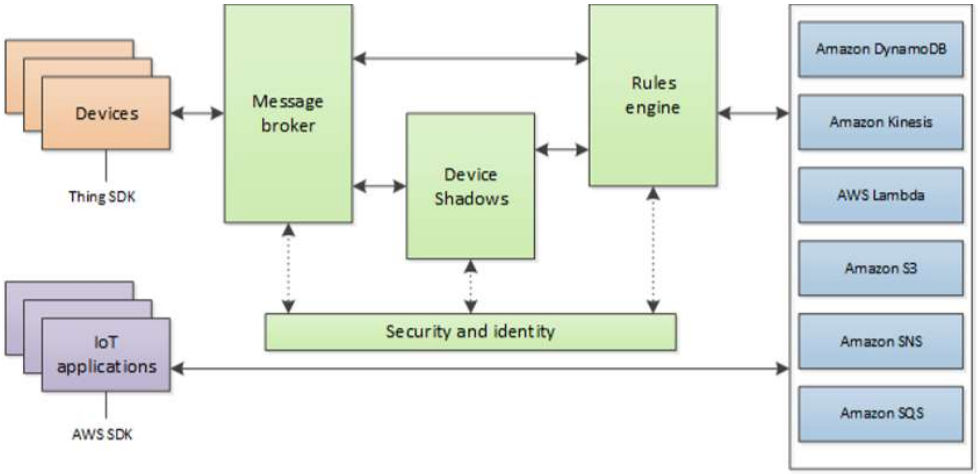
- Build your IoT applications using language-specific APIs. These SDKs wrap the HTTP/ HTTPS API and allow you to program in any of the supported languages

AWS IoT Device SDKs

- Build applications that run on devices that send messages to and receive messages from AWS IoT

AWS IoT Related Services

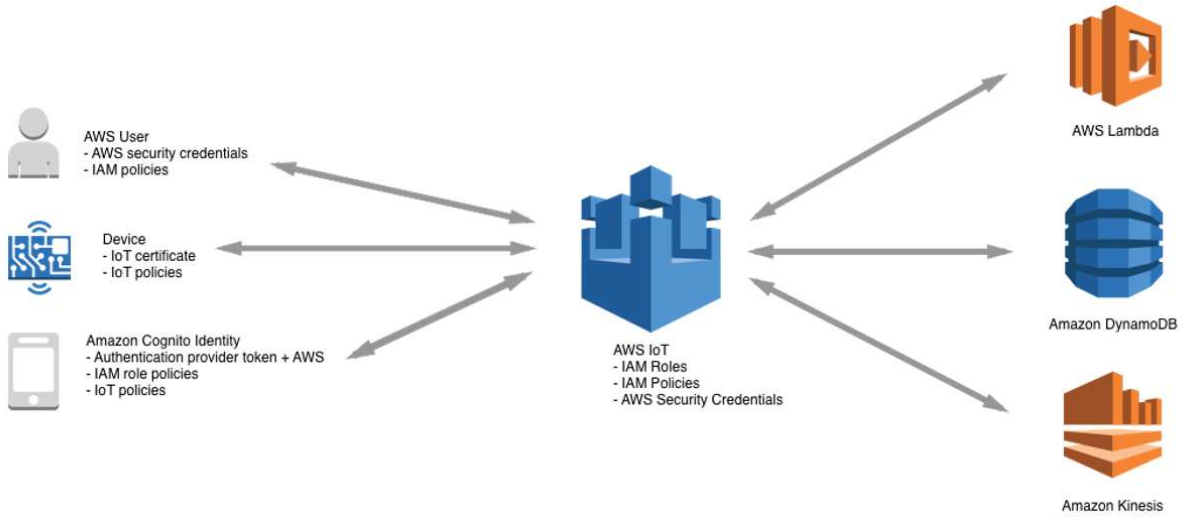
- Amazon Simple Storage Service
- Amazon DynamoDB
- Amazon Kinesis
- AWS Lambda
- Amazon Simple Notification Service
- Amazon Simple Queue Service



- Devices report their state by publishing messages, in JSON format, on MQTT topics. Each MQTT topic has a hierarchical name that identifies the device whose state is being updated.
- When a message is published on an MQTT topic, the message is sent to the AWS IoT MQTT message broker, which is responsible for sending all messages published on an MQTT topic to all clients subscribed to that topic.
- Communication between a device and AWS IoT is protected through the use of X.509 certificates.
- AWS IoT can generate a certificate for you or you can use your own. In either case, the certificate must be registered and activated with AWS IoT, and then copied onto your device.
- When your device communicates with AWS IoT, it presents the certificate to AWS IoT as a credential.
- You can create rules that define one or more actions to perform based on the data in a message. For example, you can insert, update, or query a DynamoDB table or invoke a Lambda function.
- Rules use expressions to filter messages. When a rule matches a message, the rules engine triggers the action using the selected properties.
- Rules also contain an IAM role that grants AWS IoT permission to the AWS resources used to perform the action.

AWS FLOW

Device Security



AWS Security Fundamentals

- Devices should connect using X.509 certificates
- The AWS IoT rules engine forwards device data to other devices and other AWS services according to rules you define.
- You are responsible for assigning unique identities to each device and managing the permissions for a device or group of devices.
- The message broker is responsible for authenticating your devices, securely ingesting device data, and adhering to the access permissions you place on devices using policies.

AWS IoT Authentication

- X.509 certificates
- IAM users, groups, and roles
- Amazon Cognito identities
- Federated identities

Creating certificates

X.509 Certificates

X.509 certificates are digital certificates that use the X.509 public key infrastructure standard to associate a public key with an identity contained in a certificate.

X.509 certificates are issued by a trusted entity called a certification authority (CA).

The CA maintains one or more special certificates called CA certificates that it uses to issue X.509 certificates.

Only the certification authority has access to CA certificates.

Aws recommend's that you give each device a unique certificate.

Supported Certificate-signing algorithms

SHA256WITHRSA
SHA384WITHRSA
SHA512WITHRSA
RSASSAPSS
DSA_WITH_SHA256
ECDSA-WITH-SHA256
ECDSA-WITH-SHA384
ECDSA-WITH-SHA512

Creating a Certificate.

Sign in	Sign in to the AWS Management Console and open the AWS IoT console.
Choose	In the left navigation pane, choose Security, choose Certificates, and then choose Create.
Choose	Choose One-click certificate creation - Create certificate. Or to generate a certificate with a certificate signing request (CSR), choose Create with CSR.
Use	Use the links to the public key, private key, and certificate to download each to a secure location.
Choose	Choose Activate.

IAM Users, Group and Roles

- IAM roles also allow AWS IoT to access other AWS resources in your account on your behalf.
  - Example: if you want to have a device publish its state to a DynamoDB table, IAM roles allow AWS IoT to interact with Amazon DynamoDB

AWS IoT Policies

- AWS IoT defines a set of policy actions that describe the operations and resources to which you can grant or deny access.
  - iot:Connect represents permission to connect to the AWS IoT message broker.
  - iot:Subscribe represents permission to subscribe to an MQTT topic or topic filter.
  - iot:GetThingShadow represents permission to get a device's shadow.
- AWS IoT policies allow you to control access to the AWS IoT data plane.
- The AWS IoT data plane consists of operations that allow you to connect to the AWS IoT message broker, send and receive MQTT messages, and get or update a device's shadow.

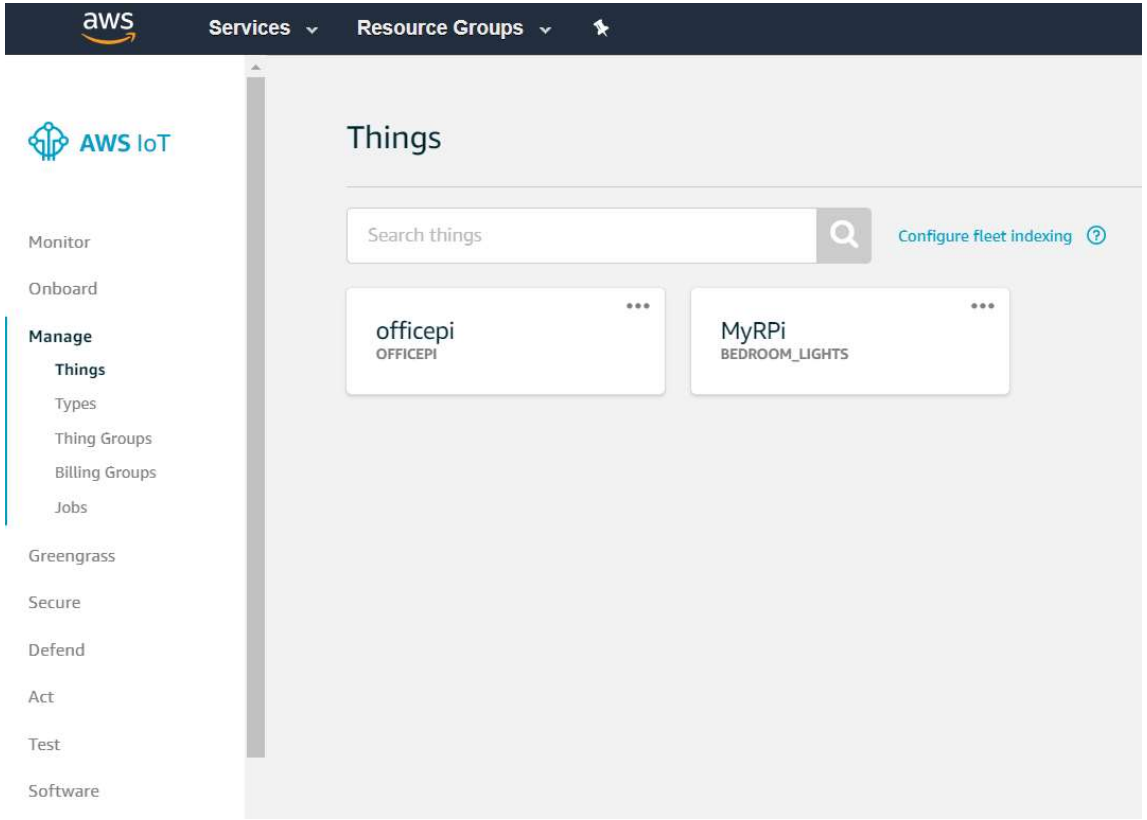
Attaching policy

Attaching Policy

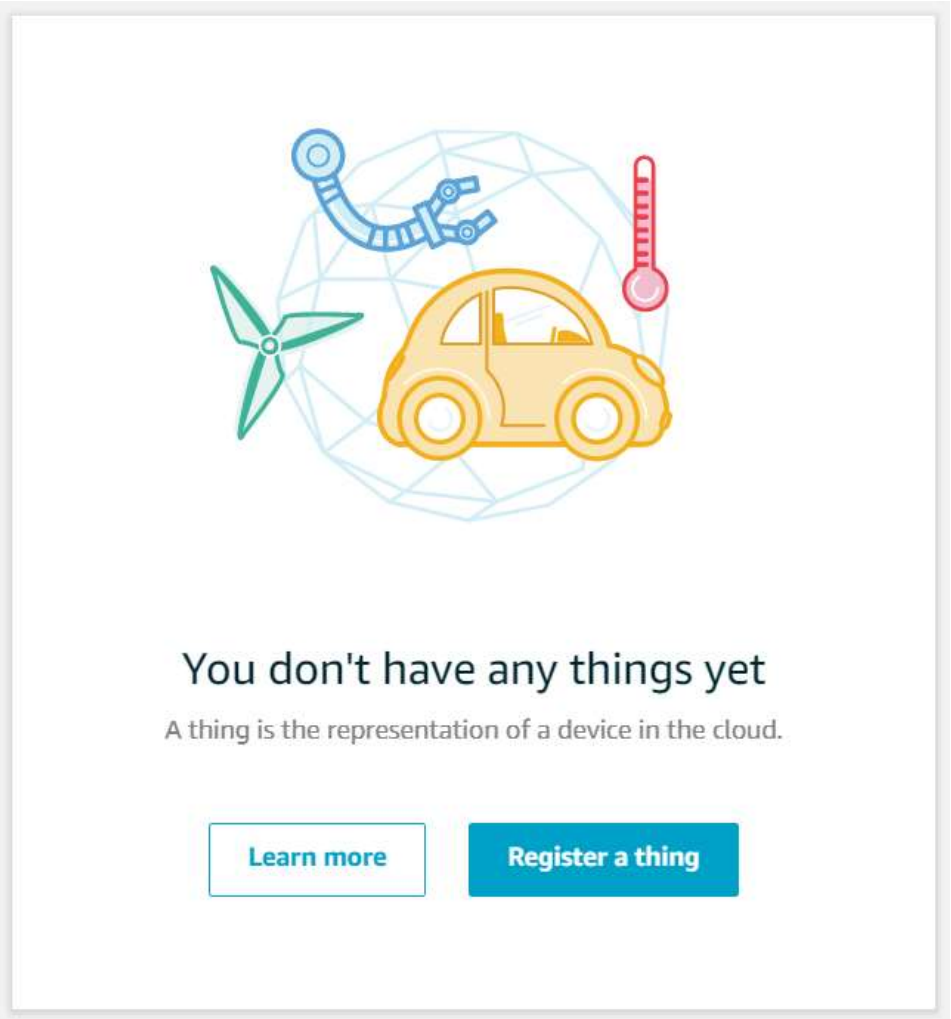
- choose Secure and Policies.
- Choose Create
- On the Create a policy page:
  - Enter a Name for the policy, such as MyIotPolicy.
  - For Action, enter iot:*. For Resource ARN, enter *.
  - Under Effect, choose Allow, and then choose Create.

Register a Device in the Registry

On the Welcome to the AWS IoT Console page, in the navigation pane, choose Manage -> Things

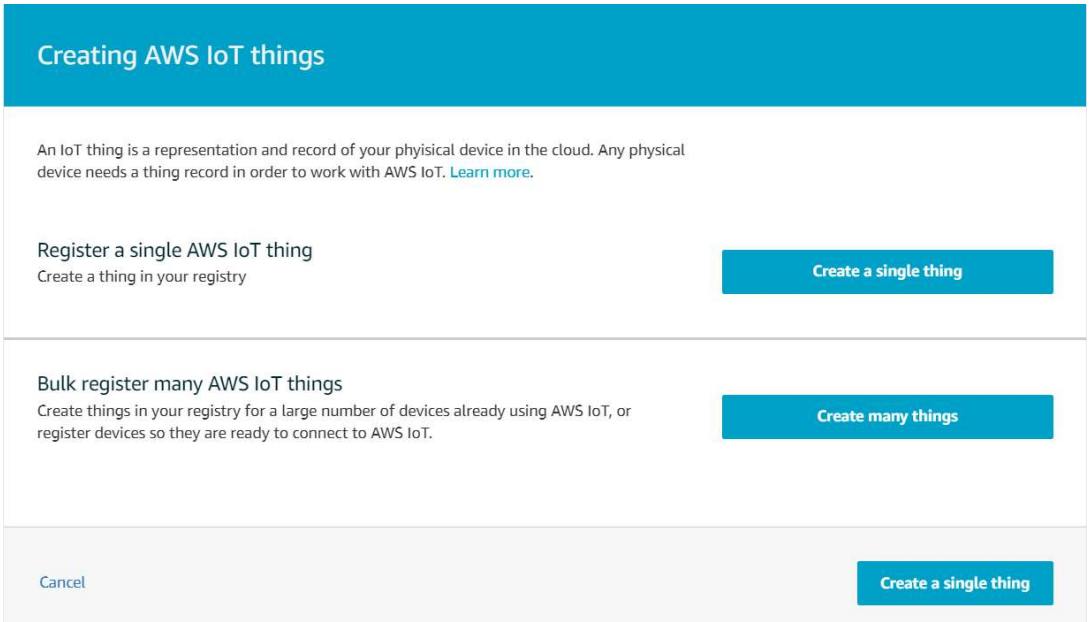


On the You don't have any things yet page, choose Register a thing



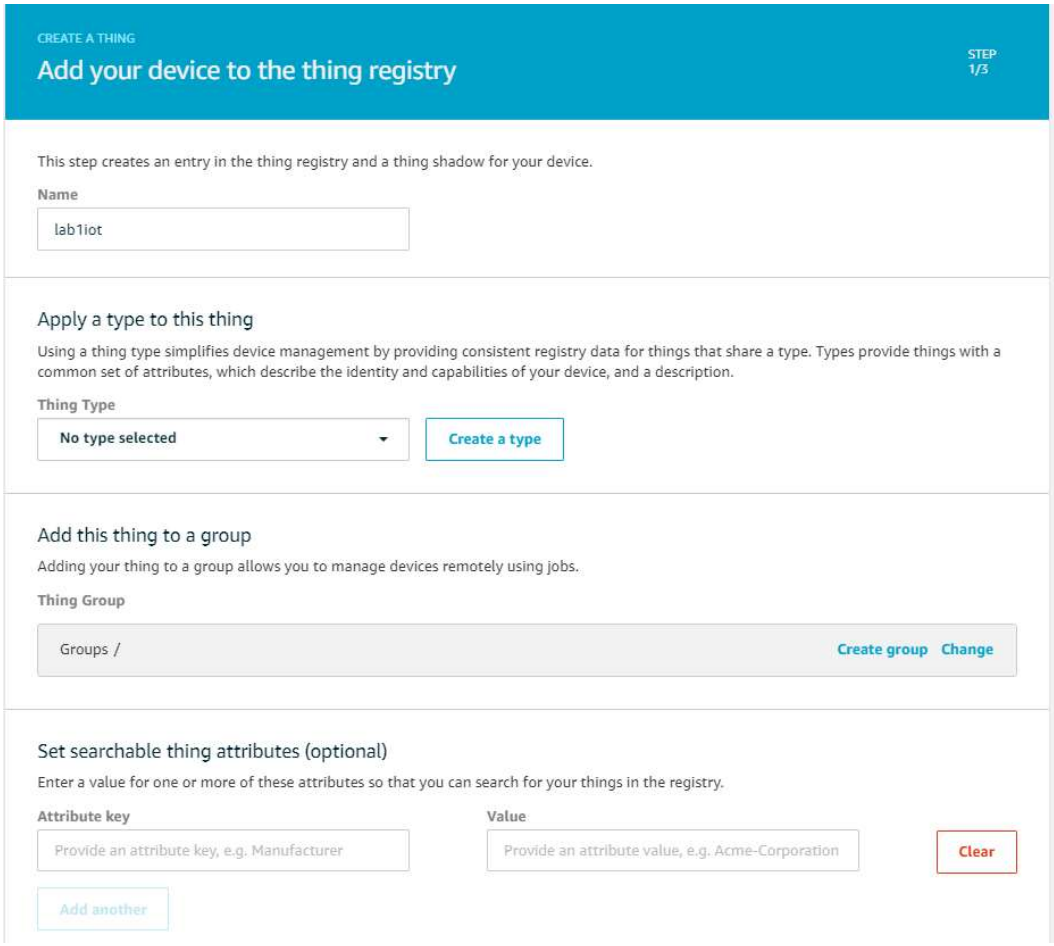
Register a thing

- On the Creating AWS IoT things page, choose Create a single thing.



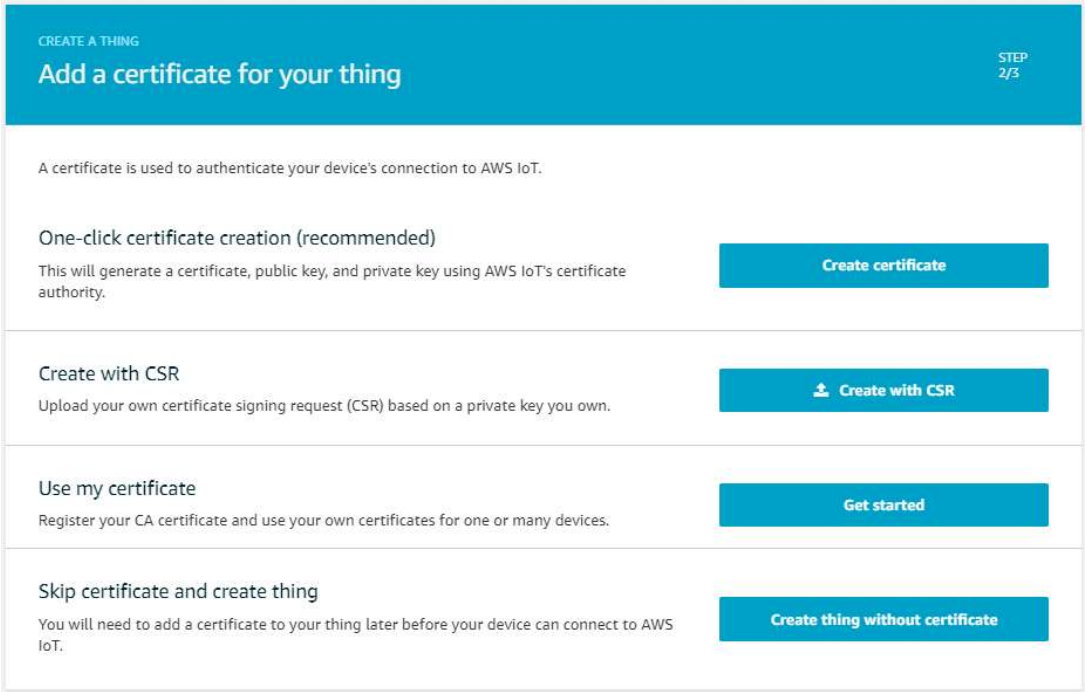
Device Naming

- On the Create a thing page, in the Name field, type a name for your thing, such as MyIotThing.
- Choose Next.
- *Note AWS does not recommend using personally identifiable information in your thing name.



Add a certificate for your thing

- On the Add a certificate for your thing page,
- Choose Create certificate. This generates an X.509 certificate and key pair.



Download the Certificates

On the Certificate created! page, download your public and private keys, certificate, and root certificate authority (CA):

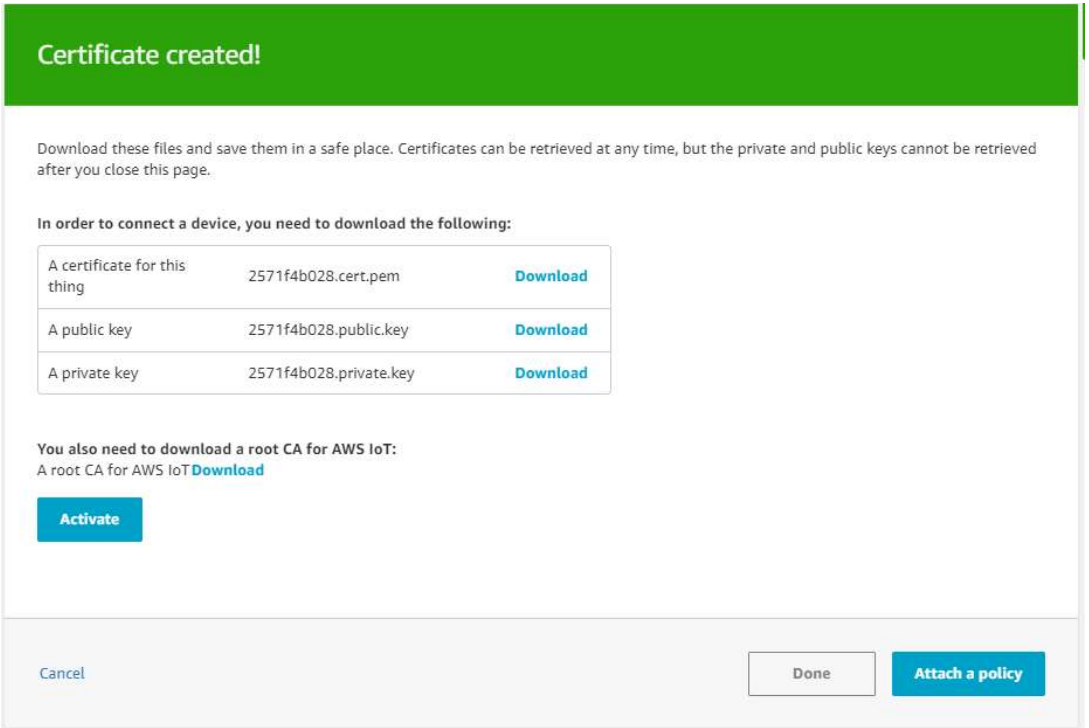
a) Choose Download for your certificate.

b) Choose Download for your private key.

c) Choose Download for the Amazon root CA.

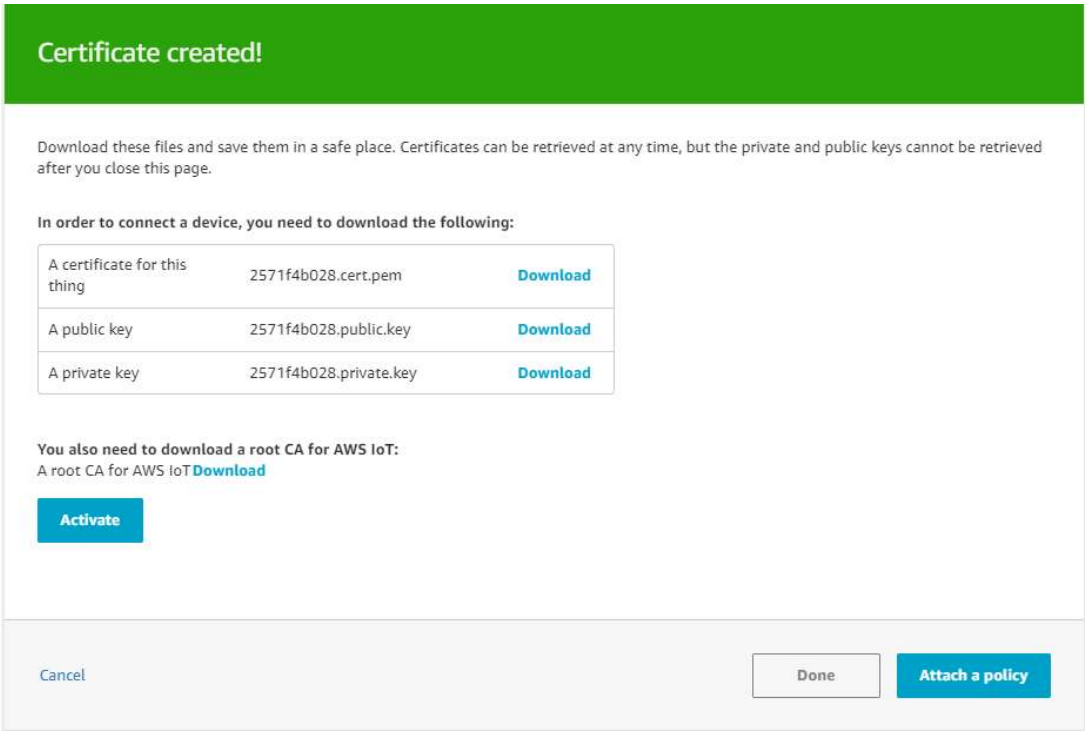
This will display a new web page. Choose RSA 2048 bit key: Amazon Root CA 1. This opens another web page with the text of the root CA certificate.

Copy this text and paste it into a file named Amazon_Root_CA_1.pem.



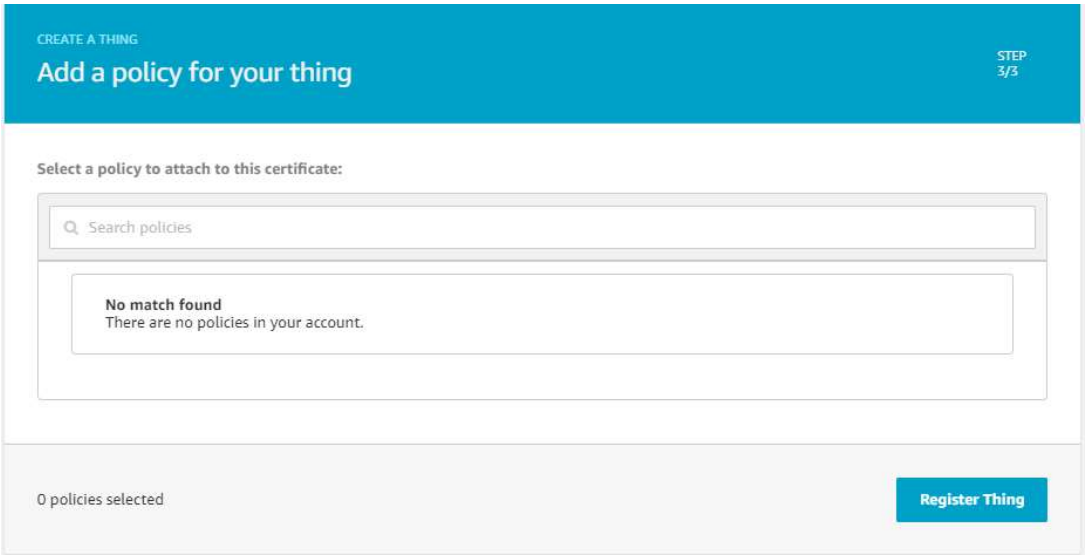
Activate & Attach Policy

- Choose Activate to activate the X.509 certificate, and then choose Attach a policy.



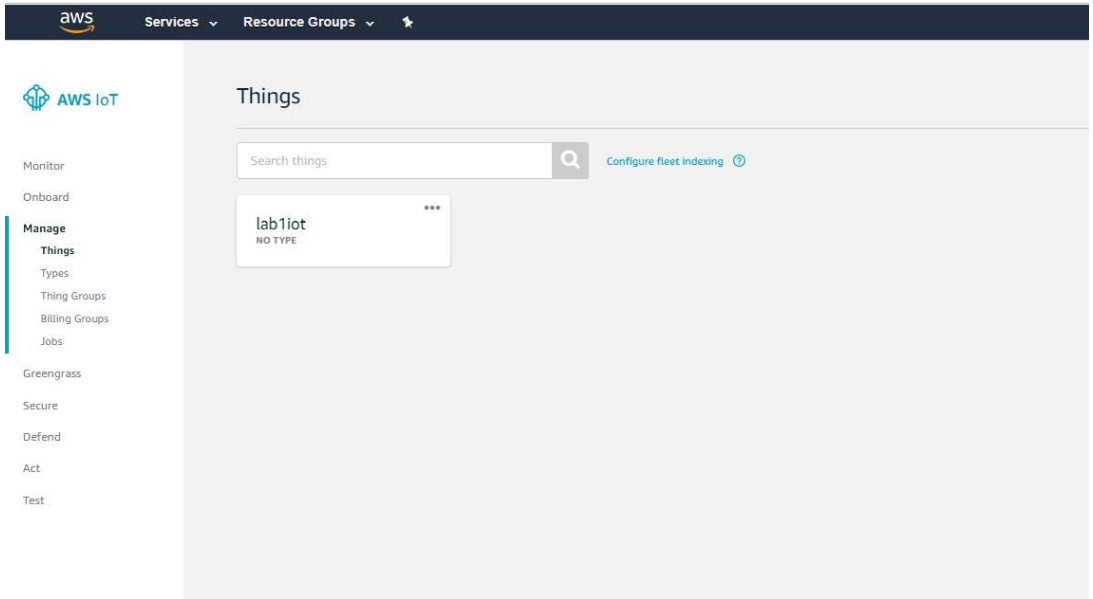
Attaching a policy

- On the Add a policy for your thing page, choose Register Thing.
- After you register your thing, you will need to create and attach a new policy to the certificate



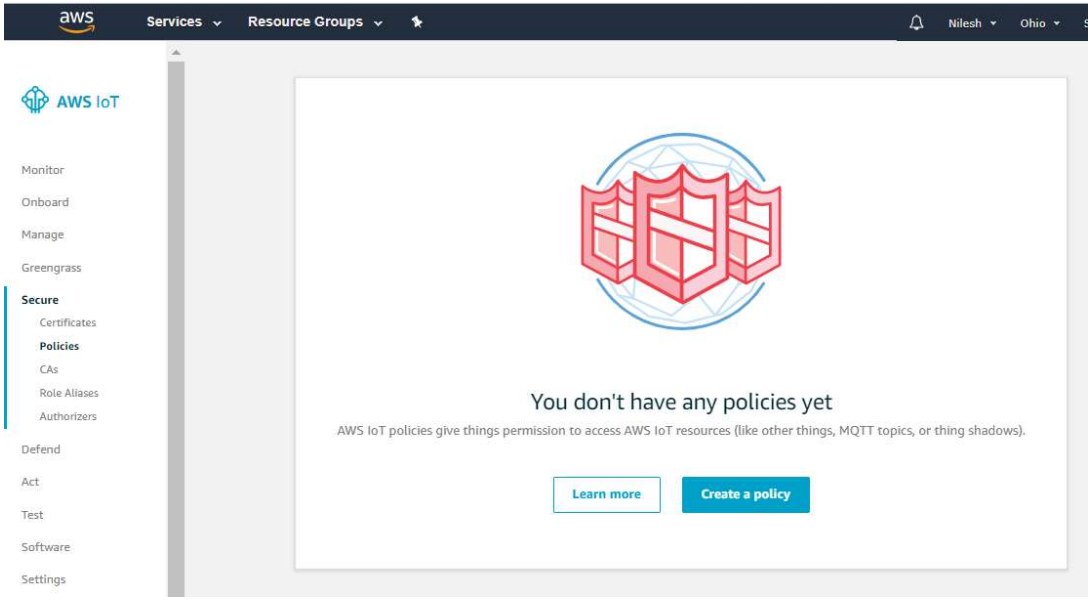
Device Registered (No Policy Attached)

- The Device is registered but does not have any policy attached to it



Creating Policies

- On the AWS IoT console, in the navigation pane, choose Secure and Policies.



Create Policy

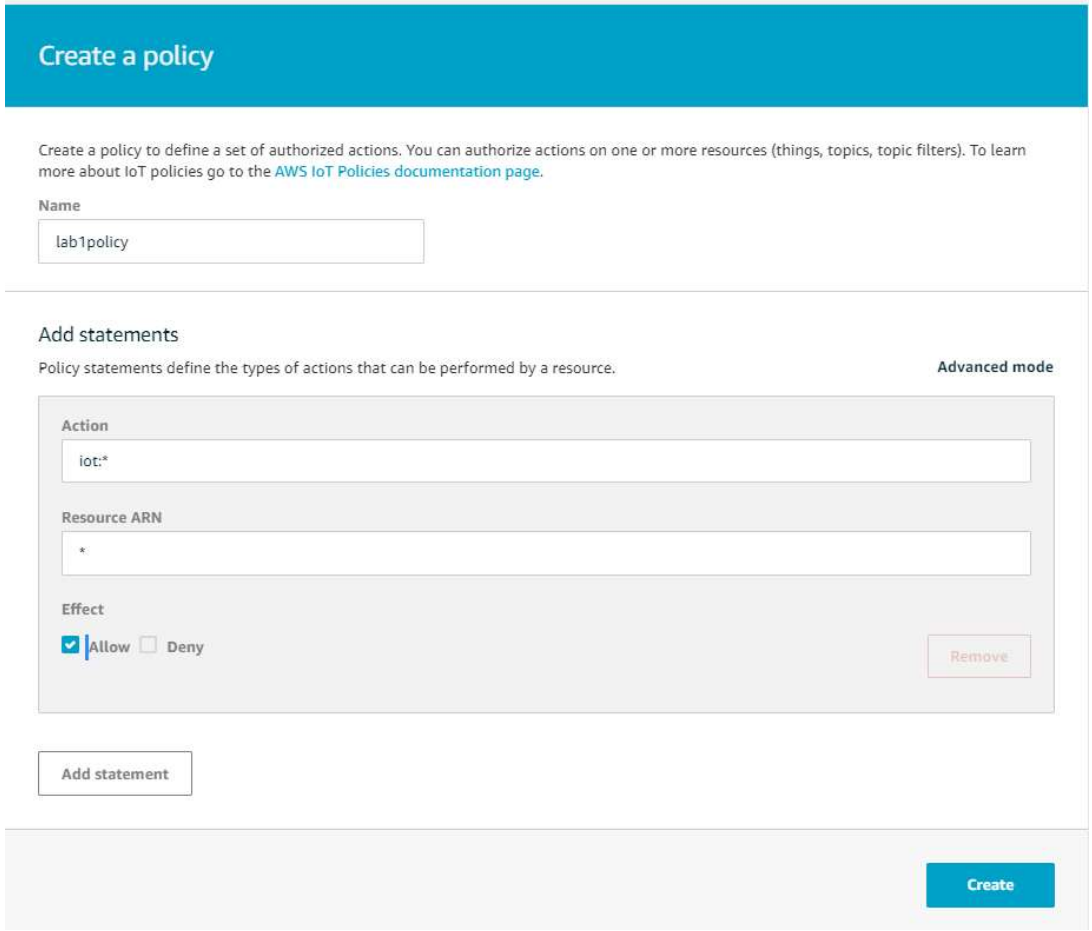
On the Create a policy page:

a) Enter a Name for the policy, such as MyIotPolicy.

b) For Action, enter iot:*.

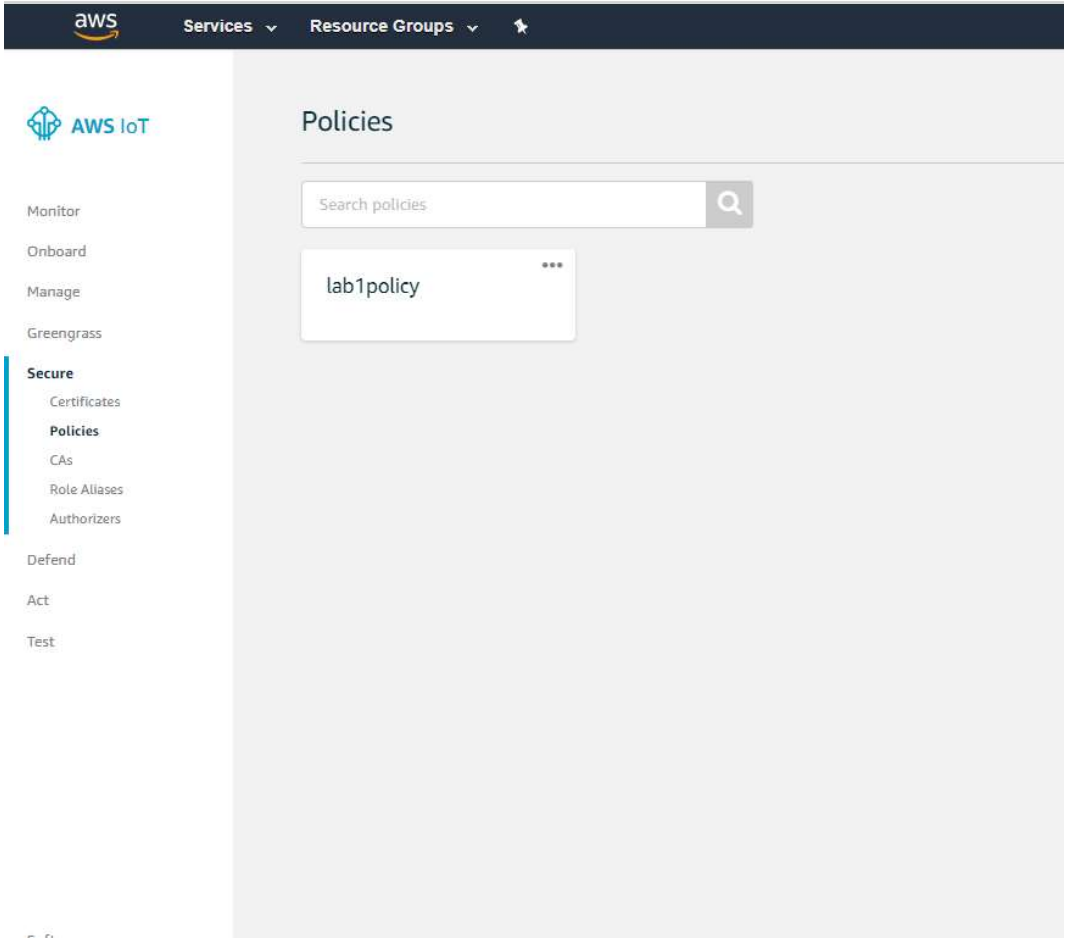
c) For Resource ARN, enter *.

d) Under Effect, choose Allow, and then choose Create.



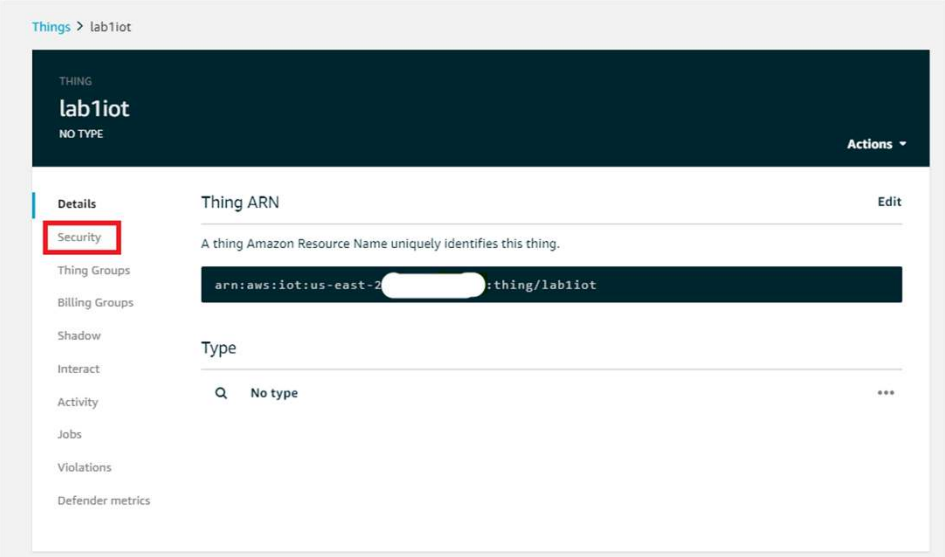
Policy Created

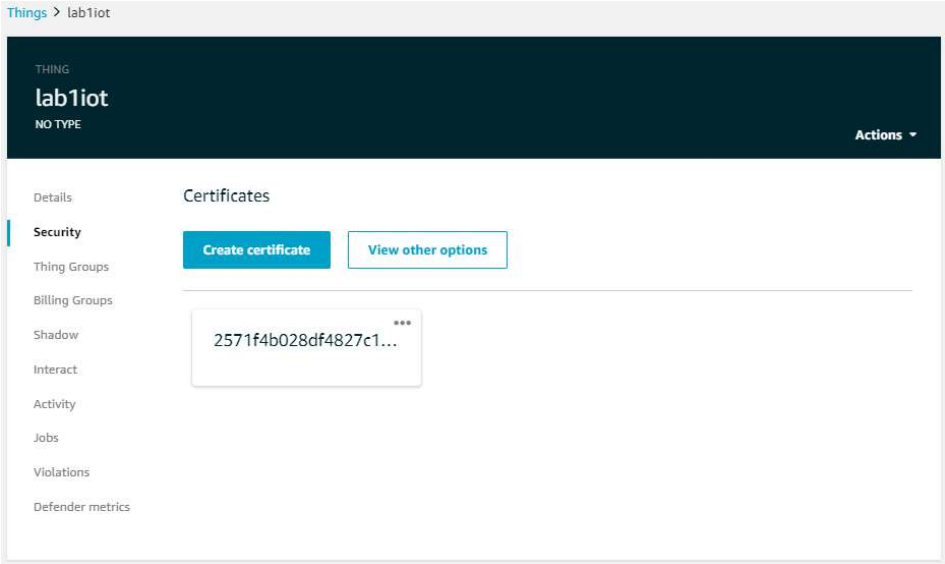
- Once the policy I s created you will need to manage the thing to attach the policy to the thing



Attach Policy

Choose your certificate an





Att

- In cy.

CERTIFICATE

2571f4b028df4827c1662566fb99ccb8a644f00469e30080c1b48ec6393240f4

ACTIVE

Actions

Details

Policies

Things

Non-compliance

Certificate ARN

A certificate Amazon Resc

arn:aws:iot:us-ea

Details

Issuer
OU=Amazon Web Service
Subject
CN=AWS IoT Certificate
Create date
Jul 7, 2019 11:13:11 AM +
Effective date
Jul 7, 2019 11:11:11 AM +
Expiration date
Jan 1, 2050 5:29:59 AM +

Attach policies to certificate(s)

Policies will be attached to the following certificate(s):

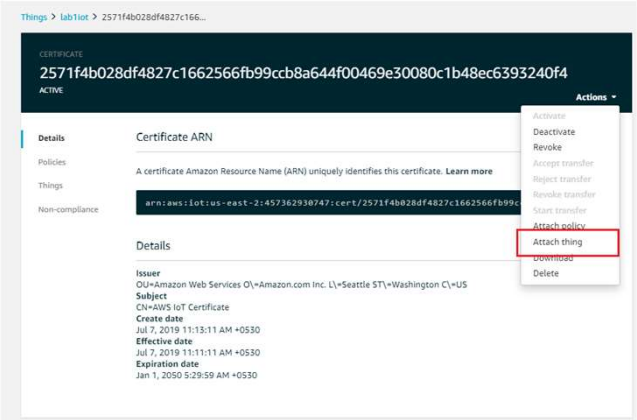
2571f4b028df4827c1662566fb99ccb8a644f00469e30080c1b48ec6393240f4

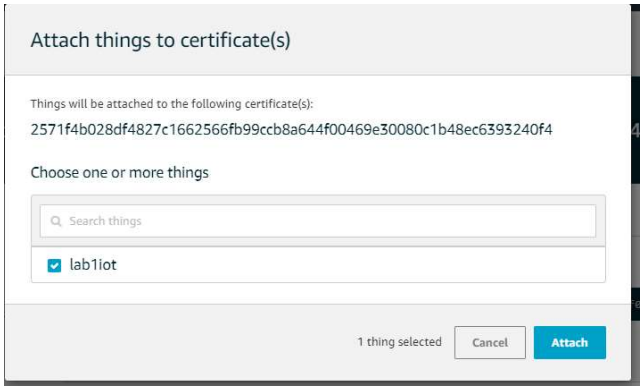
Choose one or more policies

Search policies

lab1policy View

1 policy selected Cancel Attach

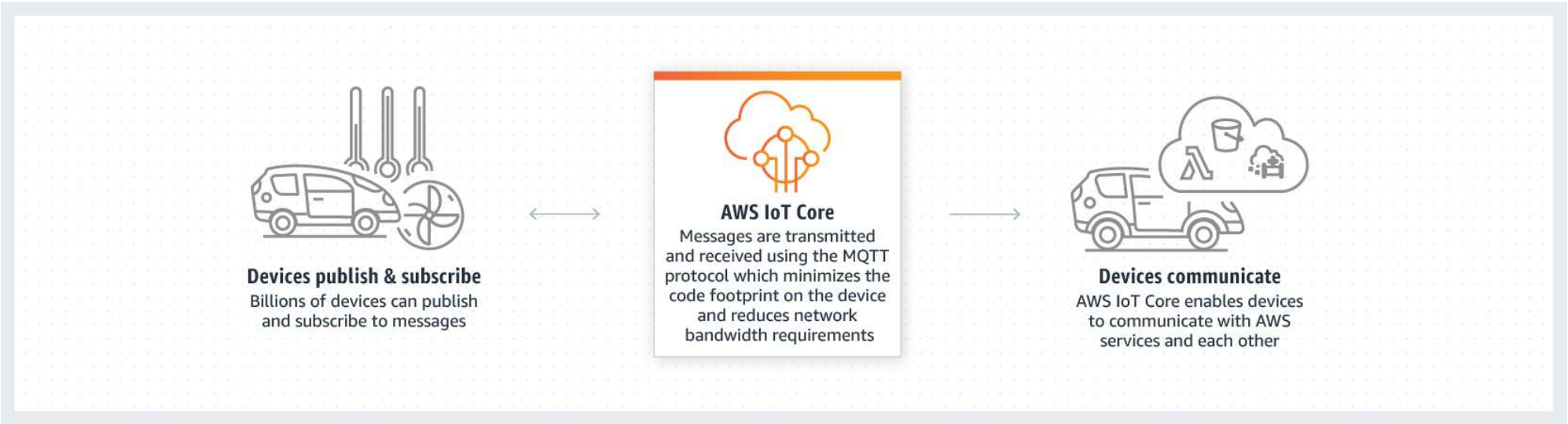




Attach A Thing to the policy

- To attach a certificate to the thing representing your device in the registry:
- 1. In the box for the certificate you created, choose ... to open a drop-down menu, and then choose Attach thing.

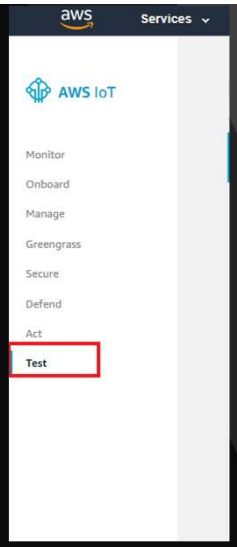
Testing a device

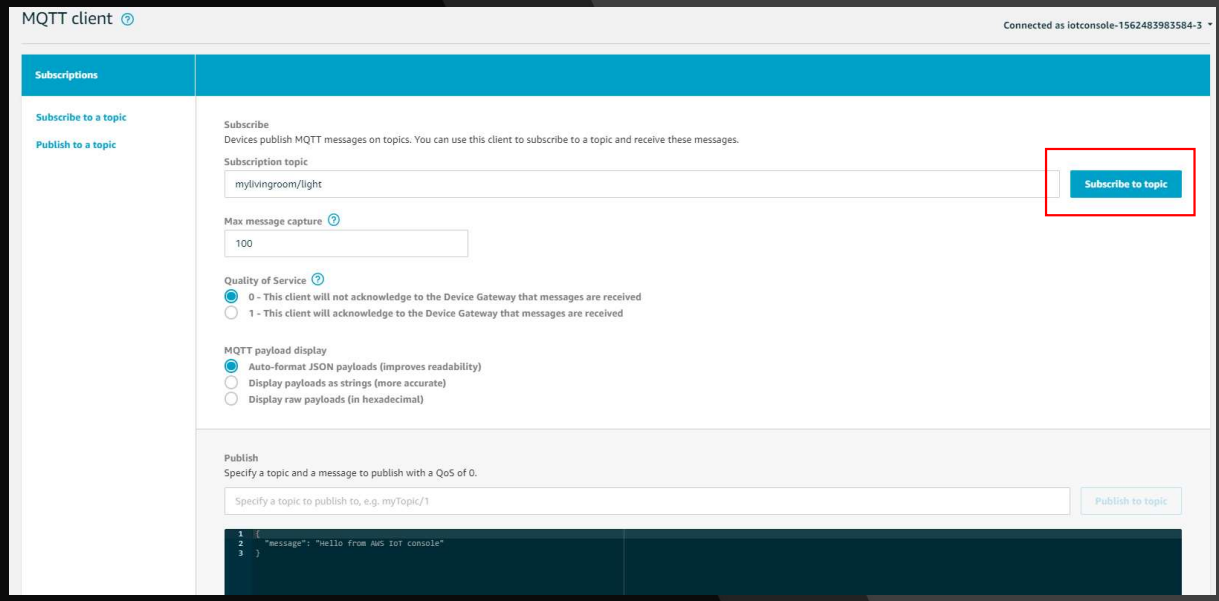


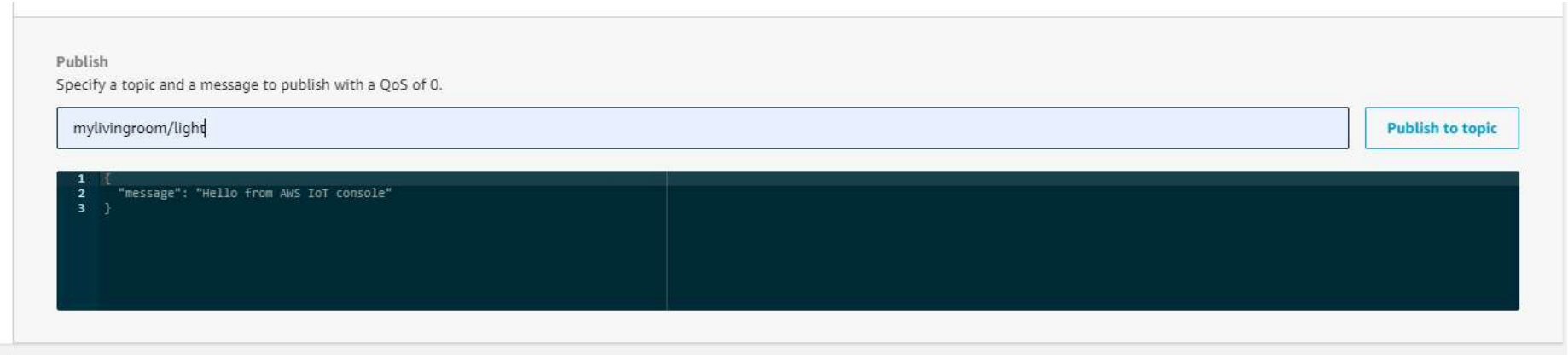
Testing the Thing

Testing the Device

- In the AWS IoT console, in the left navigation pane, choose Test.
- Subscribe to the topic on which your IoT thing publishes. Continuing with this example, in Subscribe to a topic, in the Subscription topic field, type my/topic, and then choose Subscribe to topic.







Publish a test Message

- Sending a message to the given topic thing
- Click the publish to the topic

Connect programmatically using node js

Create a project

Step 1 Create Directory awsthingtest

Step 2 npm init to create a package.json

[npm is installed with node which is already in your machine]

Step 3 npm install aws-iot-device-sdk

Step 4 type code . to start the visual studio editor

[code opens visual studio code which is already installed on you lab machine]

Connect the device using Node.JS

API : aws-iot-device-sdk

The SDK is built on top of mqtt.js and provides three classes:

- 'device',
- 'thingShadow'
- 'jobs'.

The 'device' class wraps mqtt.js to provide a secure connection to the AWS IoT platform and expose the mqtt.js interfaces upward.

Instantiating the device

```
var awsIot = require('aws-iot-device-sdk’);
var device = awsIot.device({
    keyPath: <YourPrivateKeyPath>,
    certPath: <YourCertificatePath>,
    caPath: <YourRootCACertificatePath>,
    clientId: <YourUniqueClientIdentifier>,
    host: <YourCustomEndpoint>
});
```

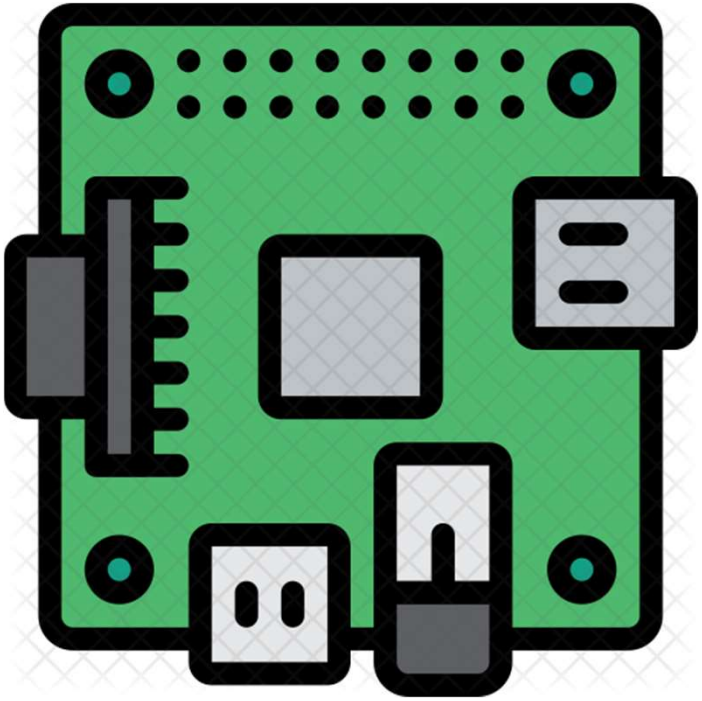
Device is an instance returned by mqtt.Client(),

```
device
  .on('connect', function() {
    console.log('connect');
    device.subscribe('topic_1');
    device.publish('topic_2', JSON.stringify({ test_data: 1}));
  });
device
  .on('message', function(topic, payload) {
    console.log('message', topic, payload.toString());
  });
```

The Device Class

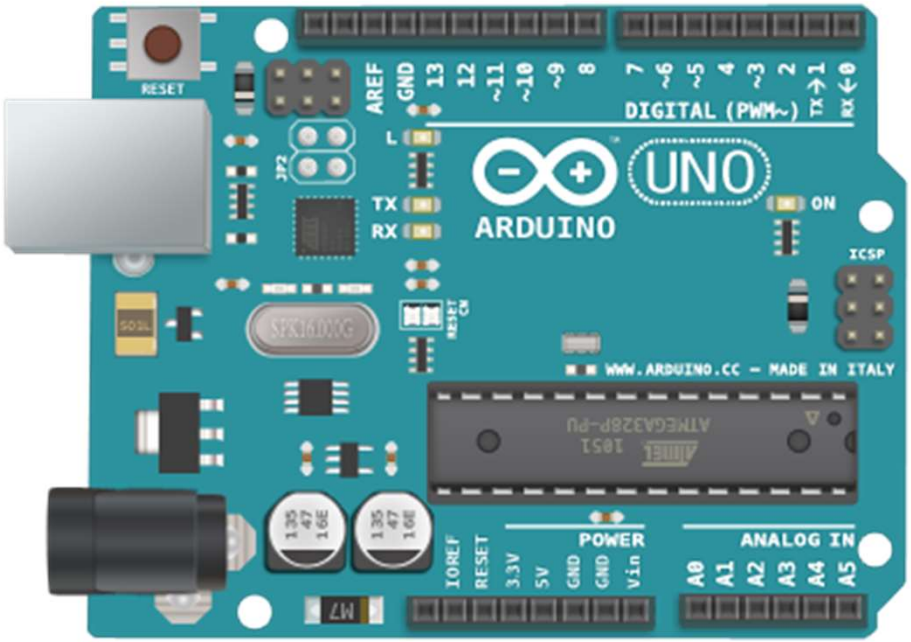
Device metrics

Device Monitoring

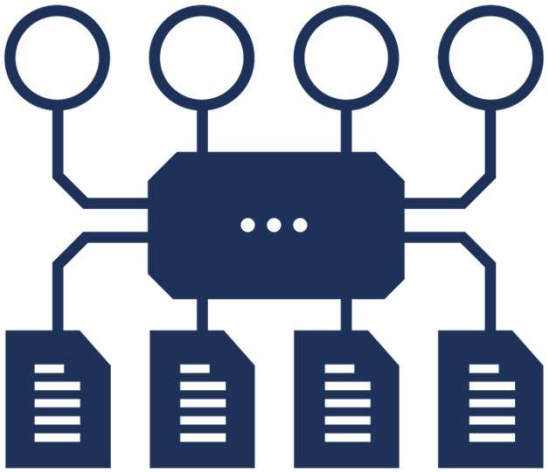


Raspberry PI on boarding

Device shadows

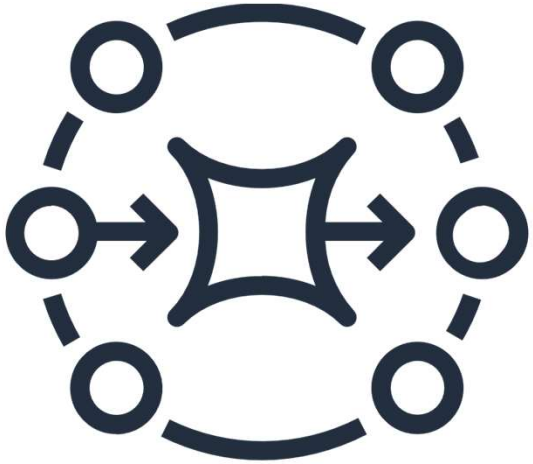


Arduino on boarding



AWS IoT Rule Engine

AWS IoT Device Defender



AWS IoT Message Broker

IoT Greengrass

Lambda Runtime

AWS IoT Integration with node-js