

# Citus™ IoT Ecosystem















This document introduce about the Citus™ IoT Ecosystem with bootstrap instructions which is used to provision an IoT Platform in Citus™ IoT Ecosystem using Docker Compose and AWS CloudFormation on AWS.



Fpt Software

# Description

Citus™ IoT Ecosystem (<https://apps.citus.io/>) is a complete IoT solution which allows consumers start to develop, integrate their IoT products, visualize sensors data in a centralized platform and rapidly building their own sharing economy business model through the Citus™ IoT Platform. It also supports dedicated infrastructure and shared infrastructure deployment.

No.	Primary Service	Hits	Image Info
1	citus-iot-ecosystem-website		
2	citus-application-gateway		
3	device-lifecycle-service		
4	citus-elasticsearch-svc		
5	sensor-remote-dashboard		
6	citus-sensor-analytics		
7	seniot-gateway		

# Features

## Web Portal

GUI Web Portal that concentrates users, devices and applications together in one place with separated workspace for each consumer or tenant user. *This feature is still in reviewing for multi-tenant security concern using kubernetes.*

- User Identity/User Groups/Roles Management using Auth0 (<https://auth0.com>) as an external service.
- Protect device/application accesses by API Gateway using API Secret Key Authentication feature.

## Application Platform

Container-based application engine is designed for Microservices architecture which is easily to deploy on Docker-Compose, Docker Swarm or Kubernetes.

- Publish or consume Docker-based applications across users.
- Continuous Delivery Support w/ Docker Hub using Web Hook.

## Manage Your Device

Device Lifecycle Management service and device security process that help you enhancing the device provisioning and communication security of the AWS IoT as well as providing Over-The-Air software update for IoT devices.

- Device Provisioning/Activation/Management.
- Device Software Update (OTA) with CI/CD.

## Data Analytics

A set of featured (default) services that allow user consuming their IoT telemetry data into business instances such as anomaly detection, face detection or plate recognition.

- Statistical Anomaly Detection
- Plate Recognition (3rd Party)
- Face Detection (3rd Party)



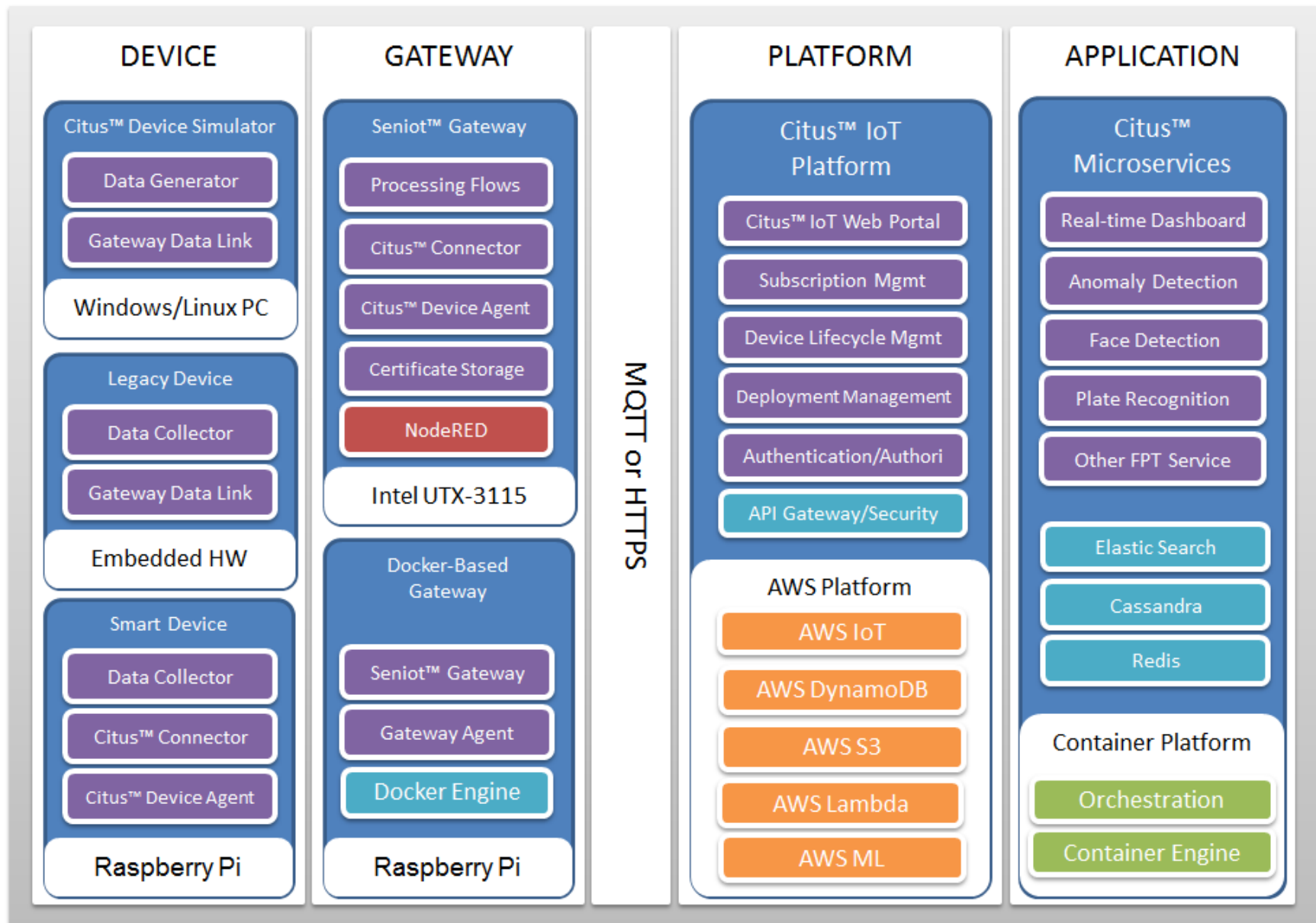
## Monitoring & Control

A set of featured Real-time Dashboards which is used to display, monitor and control your IoT devices directly using Web Portal.

- Sensor Remote Dashboard
- Citus Sensor Analytics

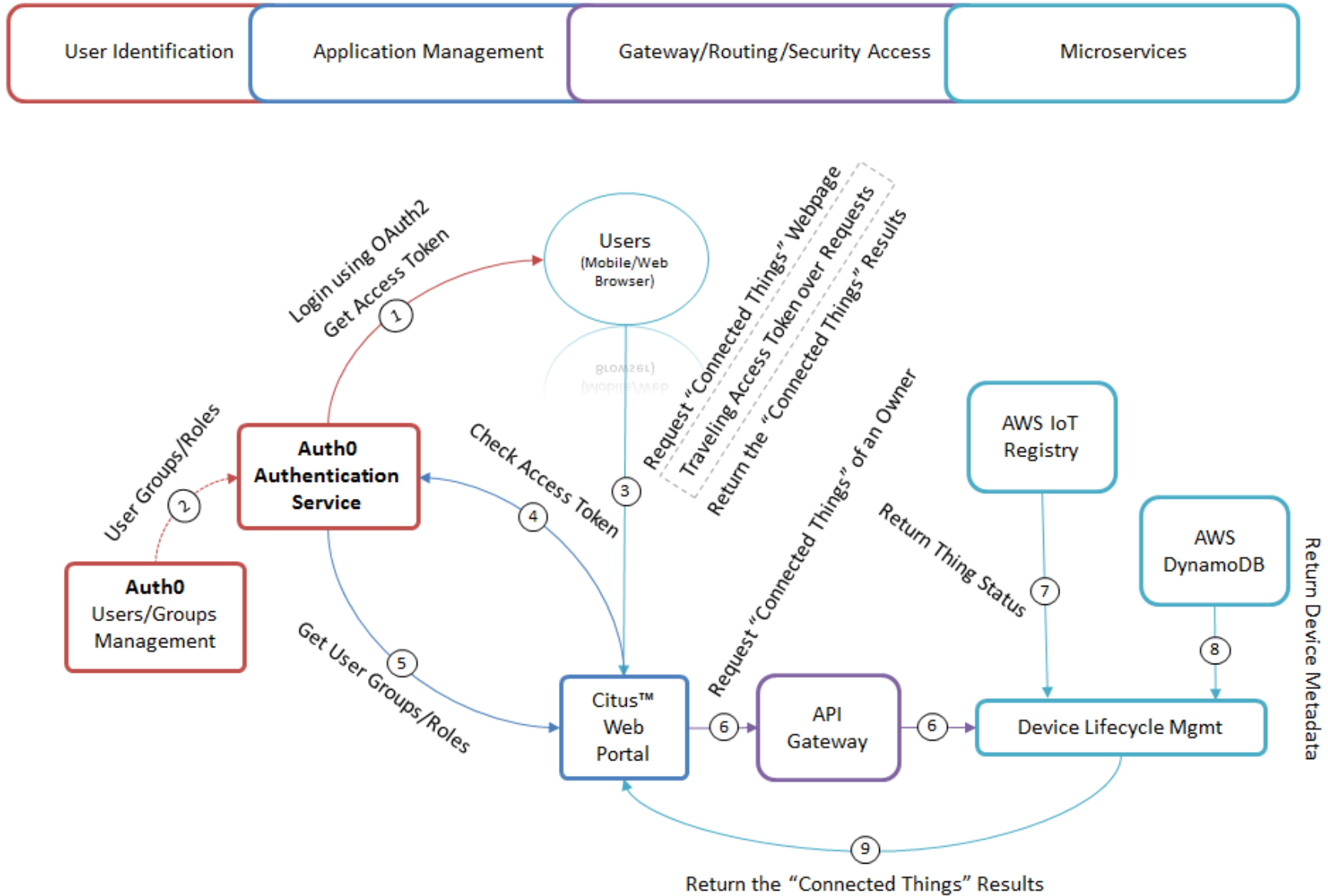
**Architecture**

## Citus™ IoT Ecosystem – System Architecture

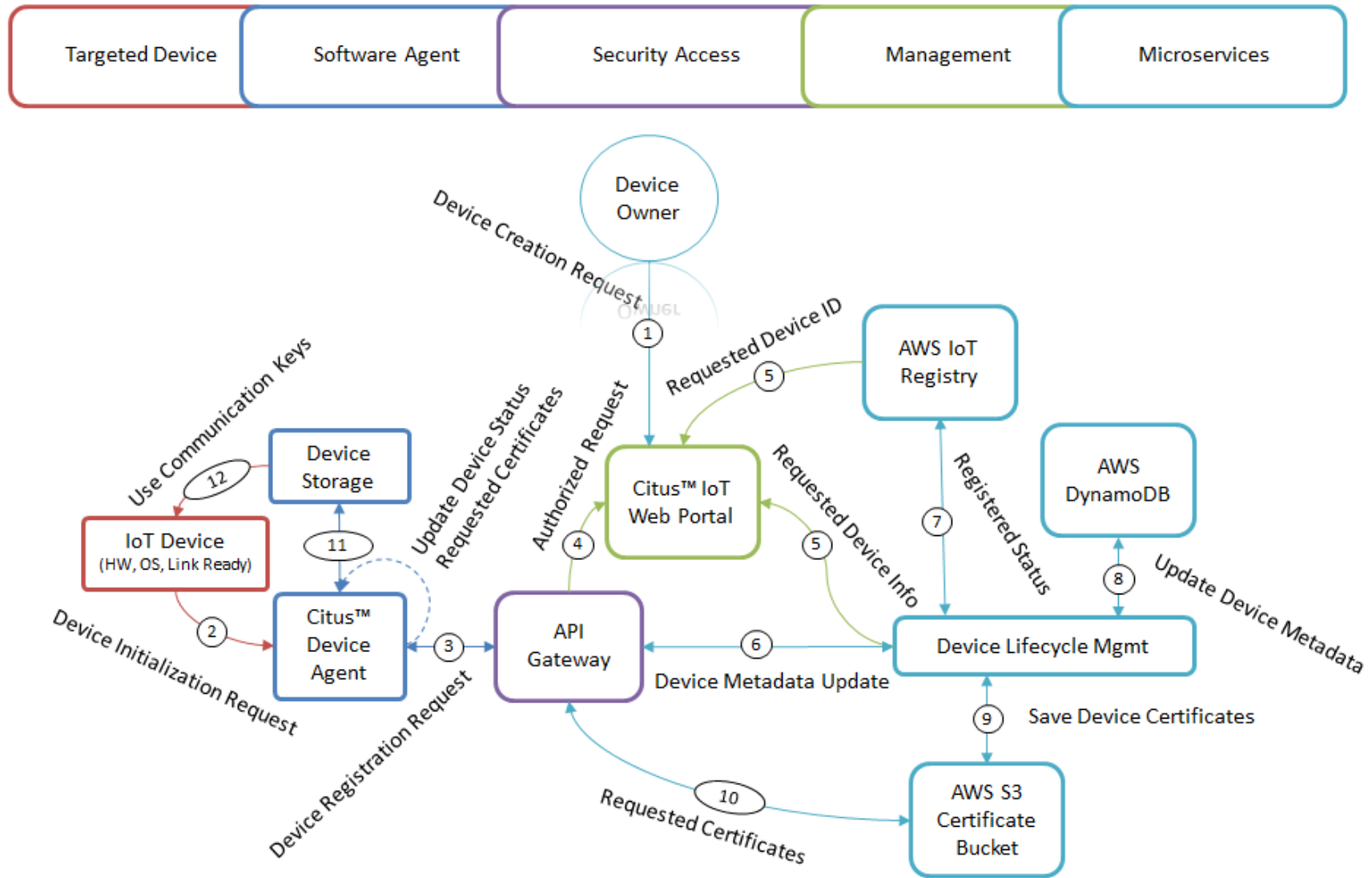


# User Story

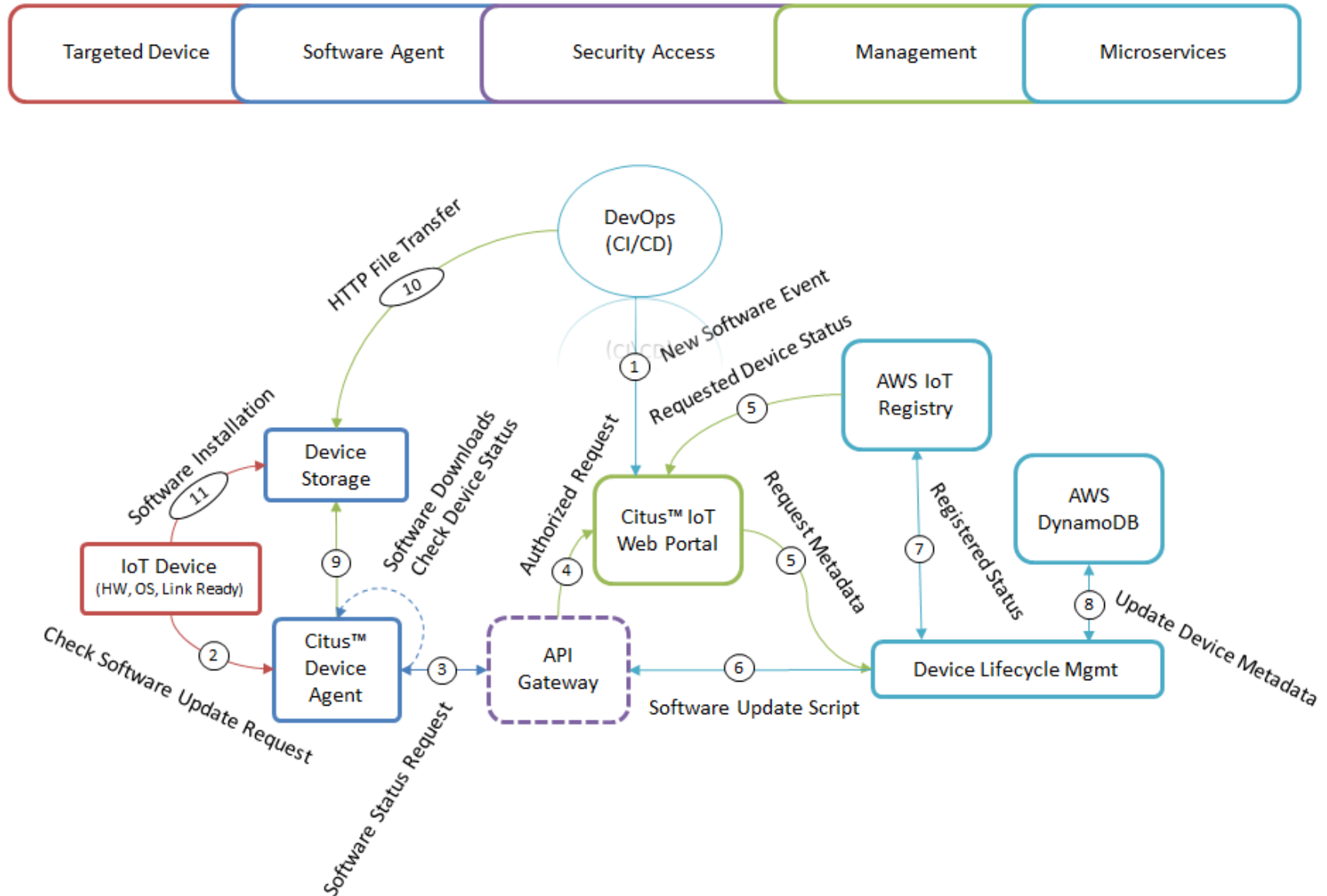
**As an Embedded SE, I want to declare my device on the Citus™ IoT Platform.**



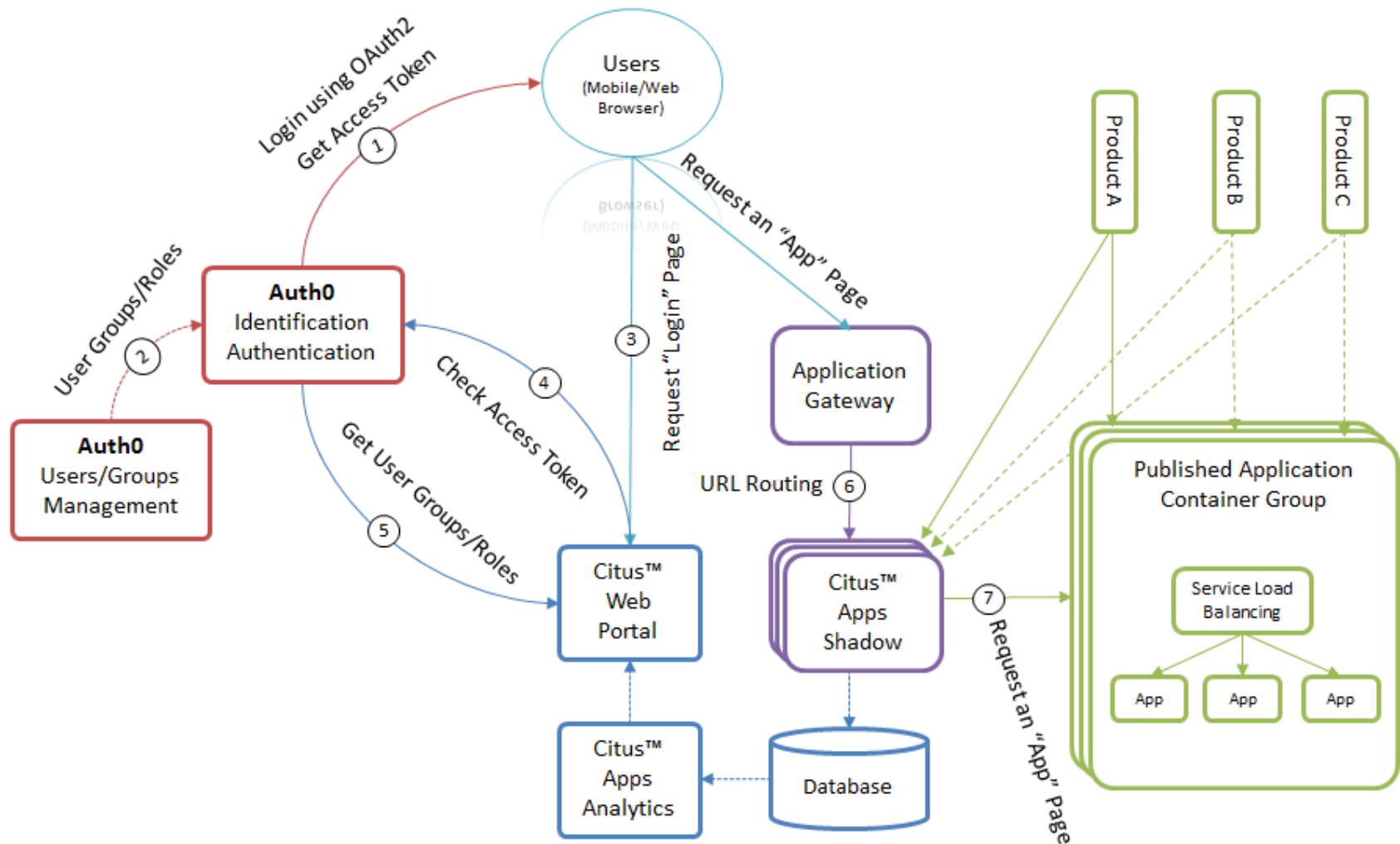
As an Embedded SE, I want my device be able to connect to Citus™ IoT Platform easily.



**As an Embedded SE, I want to update my software over-the-air when it has a new version.**

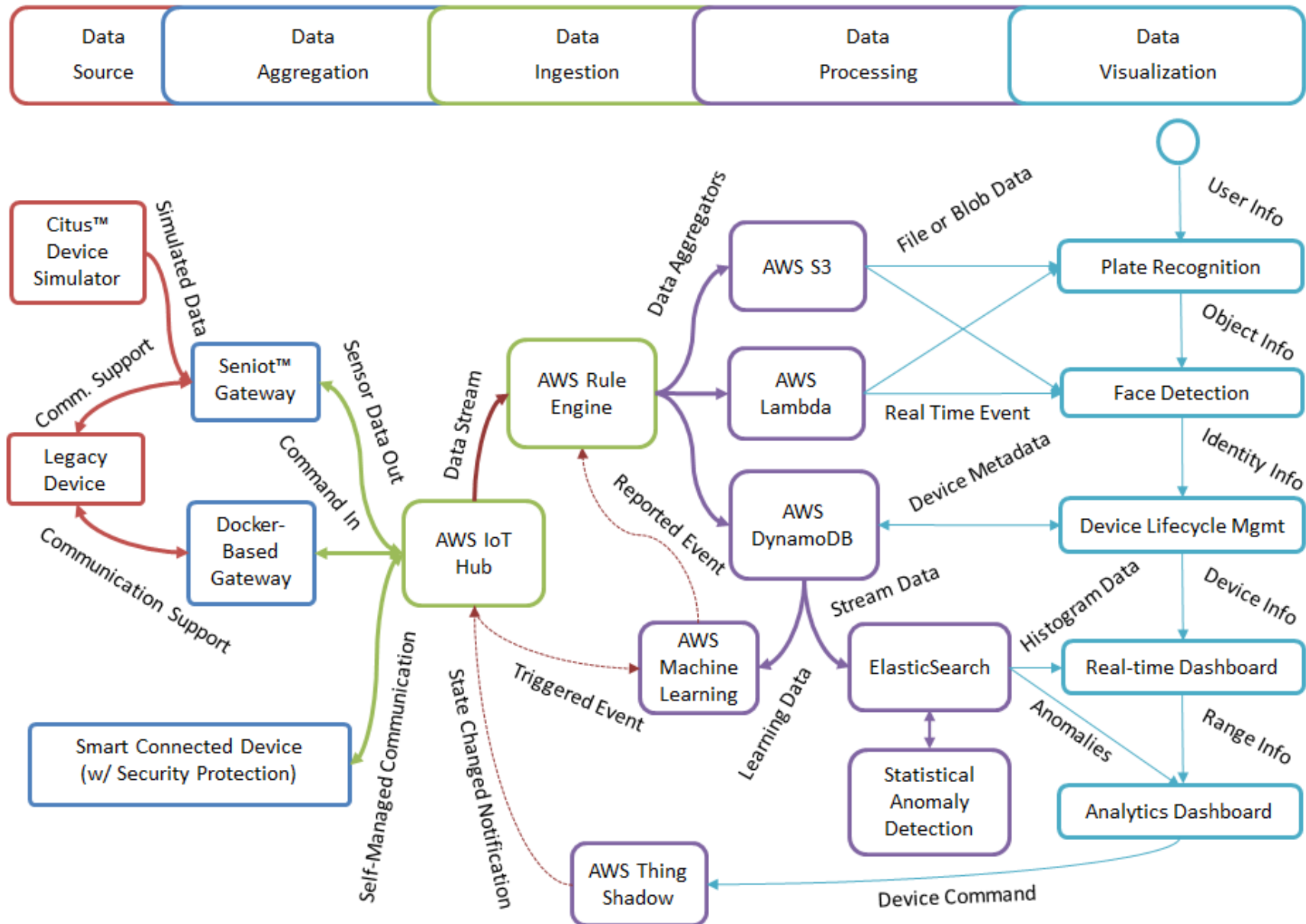


## As a Software Developer, I want to submit my application into the Citus™ IoT Platform.

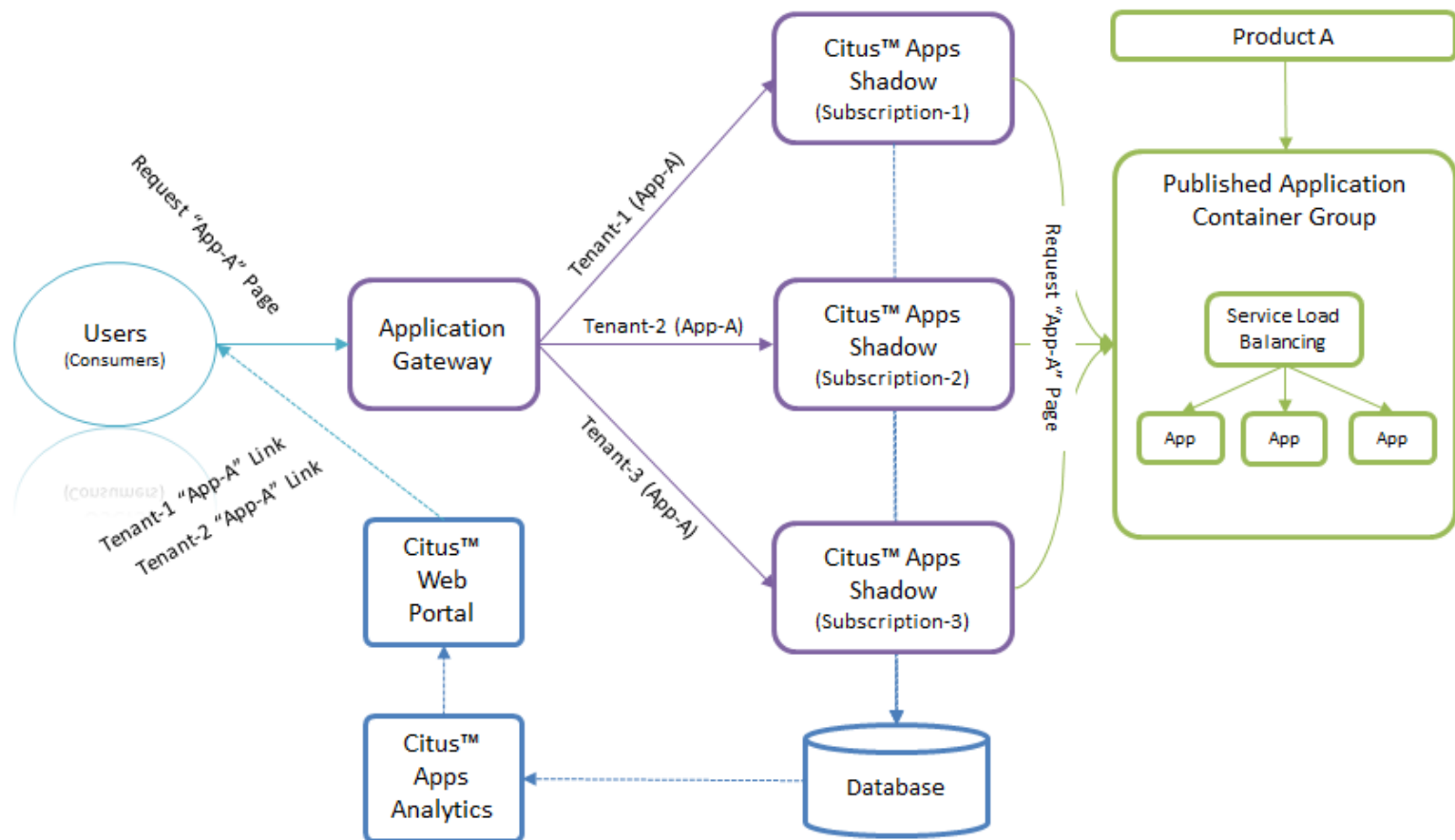




**As a Consumer, I want to ingest my sensor data in the cloud to be consumed later on.**



**As a Consumer, I want to analyze my sensor data using published application of the platform**



**Technology**

## Platforms

- AWS Cloud Computing Basic Services (VPC, EC2, Route53, Elastic IP, IAM, S3)
- AWS IoT (Hub, Registry, Rule Engine, ThingShadow)
- Cassandra/DynamoDB w/Streamming
- ElasticSearch/Logstash
- Kong API Gateway
- Docker/DockerHub
- Docker-Compose
- Docker Swarm
- Kubernetes
- Node-RED

## Languages

- HTML5/CSS3
- NodeJS
- AngularJS
- D3JS
- Nginx
- Python
- Bash Shell

# Deployment

# Prerequisites

## I. AWS Environment

(Supported Region: *ap-northeast-1* as default if using template)

1. Create [AWS IAM User](#) and manage [Access Key](#)
2. Setup [DynamoDB Table](#) with [Stream Enabled](#)

Property	Value
Database name	your-dynamodb-table-name
Table name	telemetry-sensors (default)
Primary partition key	topic (String)
Primary sort key	epoch (Number)
Stream enabled	Yes (used for citus-elasticsearch-svc)
View type	New and old images

3. Create [AWS IoT Policy](#) with at least `iot:Publish`, `iot:Receive` permissions for IoT devices sending and receiving MQTT topic messages. Then named as *your-iot-thing-policy-name*

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



4. Create [AWS IoT DynamoDB Rule](#) to forward telemetry sensor topic data into DynamoDB.
5. Create a [AWS S3 Bucket](#) and named as *your-s3-certificate-bucket-name* to store temporary device certificates.
6. Create a VPC with (YOUR-VPC-ID) and at least one public subnet (YOUR-VPC-SUBNET-ID) to host your system.
7. Create a [Hosted Domain](#) with YOUR-ROUTE53-DOMAIN-NAME and retrieve YOUR-ROUTE53-HOSTED-ZONE-ID

## II. Kubernetes Environment

1. Setup [Container Cluster on AWS using kube-aws](#)
2. Configure this cluster to use for Citus™ IoT Ecosystem (TBD)

# Step By Step

## I. Setup Development Environment

1. Install Docker Engine and Docker Compose following this link <https://docs.docker.com/compose/install/>.
2. On Windows or Mac OSX Operating System: Launch Kitematic to start docker machine then run

```
$ eval "$$(docker-machine env default)"
```

3. On Ubuntu/RHEL/CentOS: execute shell command "\$ docker-compose --version" to make sure it's running.
4. Checkout this repository **git clone** <https://github.com/cuongquay/citus-iot-ecosystem.git> or download the zipped package and extract to a folder.
5. Setup the shell environment variables which will be used by *docker-compose.yaml*

```
export AWS_DEFAULT_REGION=ap-northeast-1
export AWS_ACCESS_KEY_ID=your-s3-iot-hub-access-key-id
export AWS_SECRET_ACCESS_KEY=your-s3-iot-hub-secret-key
export AWS_IOT_CERT_BUCKET=your-s3-certificate-bucket-name
export AWS_IOT_DEVICE_POLICY=your-iot-thing-policy-name
export AWS_DYN_TABLE_NAME=your-dynamodb-table-name
```

6. Start deploying by running this shell command

```
$ cd citus-iot-ecosystem-bootstrap  
$ docker-compose up -d --force-recreate
```

7. Wait for cluster is initialised and stable. It takes about 5 minutes to pull docker images and initialize states.

8. Access to the Web Portal at <http://192.168.99.100/> on Windows/Mac OSX or <http://127.0.0.1> on Ubuntu/RHEL/CentOS.

9. Terminate the system by running this shell command

```
$ docker-compose down
```

## II. Run on AWS Cloud Formation Stack

Download [Cloud Formation Stack Template](#)

You need to change these parameters before applying the AWS CloudFormation template:

1. YOUR-ROUTE53-HOSTED-ZONE-ID
2. YOUR-AWS-EC2-SSH-KEYPAIR
3. YOUR-DNS-PREFIX-xxx1/2/3
4. YOUR-ROUTE53-DOMAIN-NAME
5. YOUR-VPC-SUBNET-ID
6. YOUR-VPC-ID

Update your AWS Credentials for your AWS IoT Hub by encoding the script below into Base64 format

```
#!/bin/bash
set -e -x

export AWS_DEFAULT_REGION=ap-northeast-1
export AWS_ACCESS_KEY_ID=your-s3-iot-hub-access-key-id
export AWS_SECRET_ACCESS_KEY=your-s3-iot-hub-secret-key
export AWS_IOT_CERT_BUCKET=your-s3-certificate-bucket-name
export AWS_IOT_DEVICE_POLICY=your-iot-thing-policy-name
export AWS_DYN_TABLE_NAME=your-dynamodb-table-name

yum update -y
yum install git -y

git clone https://github.com/cuongquay/citus-iot-ecosystem-bootstrap
cd /usr/share/citus-iot-ecosystem
chmod +x setup.sh
./setup.sh
```

Replace the **Base64UserData.Default** with the encoded value in the Cloud Formation template above.

```
"Base64UserData": {  
    "Type": "String",  
    "Default": "IyEvYm1uL2Jhc2gNCnNldCAtZSAteCANCg0KZXhwb3J  
}
```



You need to setup a corrected AWS IoT environment for AWS IoT Policy, AWS IoT Rule, AWS DynamoDB (Stream Enabled) to use with this platform. For more information, please contact us by email: [cuongdd1@fsoft.com.vn](mailto:cuongdd1@fsoft.com.vn)!

# Author

DUONG Dinh Cuong