Deploy an End-to-End IoT Application

**SPL-168 - Version 1.1.21**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Overview**

In this lab, you will: connect virtual things using AWS IoT, publish messages and visualize real-time data using a serverless web application leveraging AWS Lambda, Amazon API Gateway and Amazon S3.

LAB DESCRIPTION

By the end of this lab, you will be able to:

* Set up AWS IoT
* Create an IoT thing, policy, and certificate
* Run a device simulator on Amazon EC2
* Process and Visualize streaming data using a serverless app

**Lab Setup**

The lab comes pre-configured with an EC2 instance that you use as an IoT device simulator. It also configures an API Gateway, a Lambda function and DynamoDB tables that will be used by you to set up a serverless visualization dashboard.

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

**Caution:** You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**Warning:** Do not change the **Region** unless instructed.

COMMON SIGN-IN ERRORS

**Error: Choosing Start Lab has no effect**

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

* **Command:** A command that you must run.
* **Expected output:** A sample output that you can use to verify the output of a command or edited file.
* **Note:** A hint, tip, or important guidance.

**Set up AWS IoT**

In this task, you create the resources needed in the AWS IoT console. There are 4 components that need to be created:

* **Thing** – A thing is a logical representation of a device stored in IoT’s Registry. A thing supports attributes, as well as Device Shadows, which can be used to store device state & define desired state.
* **Policy** – You attach a policy to a certificate to dictate what the certificate (or rather, a Thing using that certificate) is entitled to do on AWS IoT.
* **Certificates** – Certificates are used for authentication. Things can communicate with AWS IoT via MQTT, MQTT over WebSockets or HTTPS. MQTT is a machine-to-machine pub-sub protocol well-suited for IoT use cases given its low overhead and low resource requirements. MQTT transmission to your AWS IoT gateway is encrypted using TLS and authenticated using certificates that you create.
* **Rule** – Rules leverages AWS IoT’s Rules Engine to dictate how messages sent from Things to AWS IoT are handled. You configure rules that send data published to an MQTT topic to a variety of AWS Services.

**Task 1: Create an IoT Thing**

1. In the **AWS Management Console**, to the right of **Services** menu, in the search bar, search for

**IoT Core**

 and then choose **IoT Core** from the list.

**Note:** If you see a message that says “Introducing the new AWS IoT console experience”, you can dismiss the message by selecting the X to the right of the message.

1. In the navigation pane at the left of the **AWS IoT** page, under the **Manage** category, choose **All devices** -> **Things**.

**Note:** There will not be any things listed in this section yet.

1. On the **Things** page, choose **Create things** .
2. On the **Create things** page, in the **Number of things to create** section, select **Create a single thing**.
3. Choose **Next** .
4. On the **Specify thing properties** page, in the **Thing properties** section:

* **Thing name:**

iotThing

* Keep all other default settings

1. At the bottom of the page, choose **Next** .
2. On the **Configure device certificate -** ***optional*** page, choose **Skip creating a certificate at this time**.
3. Choose **Create thing** .

**Expected output:** A message should display stating: *You successfully created thing iotThing*.

1. In the AWS IoT navigation pane, choose **Settings**.
2. In the **Device data endpoint** section, copy the **Endpoint** to your text editor.

**Expected output:** It should look like this: **a1xduq2dksdz70-ats.iot.us-west-2.amazonaws.com**. You need this endpoint to configure the device simulator later in the lab.

**Task 2: Review the IoT Policy**

In this task, you review the IoT policy that was created for this lab.

1. In the AWS IoT navigation pane, choose **Security** > **Policies**.
2. Choose **iotPolicy**.
3. The policy actions are shown in the version tab. This policy was created in advance for this lab.

In this policy, only your specific Amazon Resource Name (arn) is given specific rights to publish, receive, and subscribe to the MQTT topic **device/\*/rechargeAlert** and **device/\*/devicePayload**. No other topics are allowed. Also, this policy limits the devices that can send messages to any IoT thing that is attached. **iot:Connection.Thing.IsAttached**. This means that if a thing cannot publish or subscribe unless it exists in the AWS IoT Core thing registry.

**Task 3: Connect to Your EC2 IoT Device Simulator**

In this task you connect to your EC2 IoT instance.

1. In the **AWS Management Console**, to the right of **Services** menu, in the search bar, search for

**EC2**

 and then choose **EC2** from the list.

1. In the navigation pane at the left of the page, choose **Instances**.
2. Select **Ec2InstanceDeviceSimulator** and then, at the top-right of the page, choose **Connect** .
3. On the **Connect to instance** page, choose the **Session Manager** tab, and then choose **Connect** .

**Expected output:** A new browser tab opens with a console connection to the instance. A set of commands are run automatically when you connect to the instance that change to the user’s home directory and display the path of the working directory, similar to this:

cd HOME; pwd

sh-4.2$ cd HOME; pwd

/home/ec2-user

sh-4.2$

**Note:** On a Windows-based computer, you might need to use **Ctrl + Shift + V** or open the context menu (right-click) to paste text into the console window.

**Task 4: Attach the Policy and Thing to the IoT Certificate**

In this task, you associate the certificate from your EC2 IoT Simulator to your policy and thing. While it is possible to create the device certificates in the AWS Management Console, these have been created for you during the CloudFormation stack creation via a script on your EC2 device simulator instance.

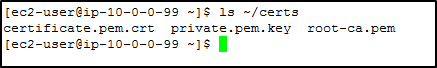
1. In the terminal, enter:

ls ~/certs

**Expected output:**

You should have 3 files in the directory:

* certificate.pem.crt
* private.pem.key
* root-ca.pem



*Image description: In the preceding image, a console output is displayed, listing three files.*

1. In the **AWS Management Console**, to the right of **Services** menu, in the search bar, search for

**IoT Core**

 and then choose **IoT Core** from the list to return to the IoT Core service screen.

1. In the AWS IoT navigation pane on the left, choose **Security** > **Certificates**.

**Note:** A certificate has been pre-created by the CloudFormation template in your lab. The certificate must be associated with the thing and the policy that you created earlier. If there isn’t a certificate listed, wait 30 seconds and then refresh the page.

1. Choose on the name of the listed certificate.
2. Select **Actions** .
3. Choose **Attach policy**.
4. In Attach policies to certificate(s) section:

* Select the  **iotPolicy**
* Choose **Attach policies** .

1. Select **Actions** .
2. Choose **Attach to things**.
3. In Attach things to certificate(s) section

* Select the  **iotThing**
* Choose **Attach to thing** .

**Task 5: Configure and Run the IoT Device Simulator**

An example script is provided that will send messages containing current battery charge, simulated GPS location data, as well as other telemetry data. The AWS IoT Service will process these messages and send them to the appropriate AWS services based on the rule actions that you configure.

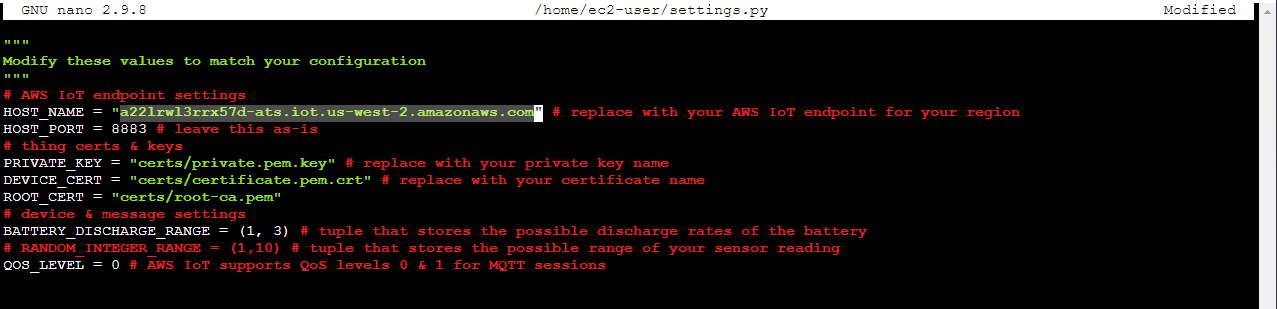
1. Use the arrow keys on your keyboard to navigate through the editor. If using a mouse, use the right mouse button to paste into the editor.

 In the terminal window, edit the settings.py file by entering:

nano ~/settings.py

**Note:** You are welcome to use any other text editor. This lab guide uses the nano editor.

1. Replace the **HOST\_NAME** value with the Device data endpoint of your thing that you copied earlier to the clipboard. If you forgot to copy the endpoint, you can obtain it again by going to: **Settings** > **Device data endpoint**.



*Image description: In the preceding image, the content of the settings.py script is displayed.*

1. Save the settings.py file by:

* Pressing **Ctrl + X**
* Typing

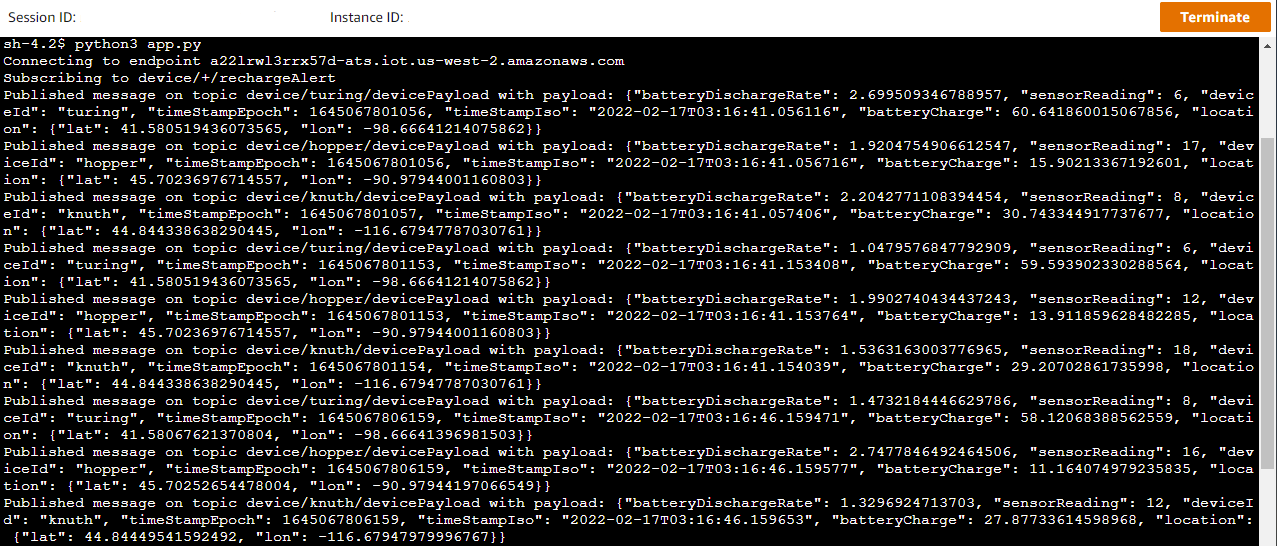
y

 and then pressing **Enter**

1. Start the device simulator by entering:

python3 app.py

**Expected output:** IoT Core receives and displays data when the devices connect.



*Image description: In the preceding image, a series of messages are displayed.*

**Task 6: Create an IoT Rule and Action**

IoT Rule Actions give your devices the ability to interact with AWS services. Rules are analyzed and actions are performed based on the MQTT topic stream. The simulated IoT devices report current battery charge percentage which decreases over time. We will create a rule action that will monitor the reported battery charge and publish a message to a new topic when it is time to recharge. The device is subscribed to this topic and will “take action” to recharge.

1. In the **AWS IoT Console**, in the navigation pane at the left of the page, under the **Message routing** category, choose **Rules**.
2. Choose **Create rule** .
3. In the **Specify rule properties** section, configure:

* Under **Rule properties**, for **Rule name**, enter

gsRecharge

.

1. Choose **Next** .
2. In the **Configure SQL statement** section, in the **SQL statement** code block, delete the existing text and replace it with the following:

SELECT \* FROM 'device/+/devicePayload' WHERE batteryCharge <=0

1. Choose **Next** .
2. In **Attach rule actions** page, under **Rule actions** section, for **Action 1** form, configure:

* **Action 1:** Select the **Republish to AWS IoT topic** option from the dropdown.
* **Topic:**

device/${topic(2)}/rechargeAlert

* **IAM role:** Select **AwsIotRepublishRole** from the dropdown.

1. Choose **Next** .
2. Then choose **Create** .

**Expected output:** Once complete, a message appears stating that your rule was successfully created.

**Task 7: View Device Messages with the AWS IoT MQTT Client**

Devices publish MQTT messages on topics. You can use the AWS IoT MQTT client to subscribe to these topics to receive the content of these messages. You now use the AWS IoT MQTT client to confirm that the IoT messages are being sent back and forth between the devices and the AWS IoT Device Gateway.

1. In the Navigation pane, Choose **MQTT test client**.
2. Under **Subscribe to a topic** section, enter

#

 as Topic filter.

This is a wild card symbol.

1. Select **Subscribe** .

**Expected output:** If the devices are successfully configured, the MQTT messages that are received display on the page. Next, you confirm that the Recharge Rule is configured correctly.



*Image description: In the preceding image, a sample MQTT message is displayed.*

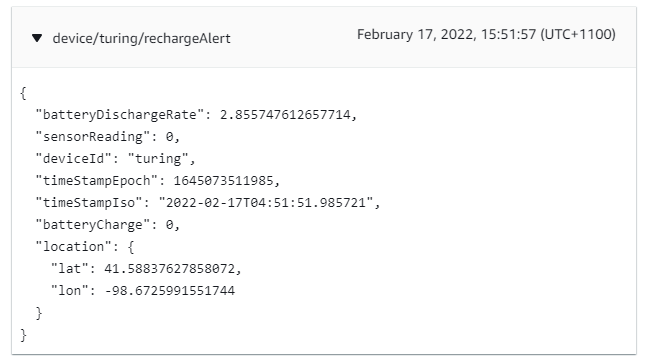
1. Delete the **#** symbol in the **Subscriptions** field.
2. Under **Subscribe to a topic** section, enter

device/+/rechargeAlert

 as Topic filter.

1. Select **Subscribe** .
2. Wait for messages to be displayed.

**Expected output:** It may take 2-5 minutes before the messages appear.



*Image description: In the preceding image, a sample recharge alert message in json format is shown.*

1. In the left navigation pane, choose **Monitor**.

**Expected output:** The **Messages published** count should increase. It may take 2-5 minutes to see the increase, and you may need to refresh the page.

1. Take a moment to review the rest of the Dashboard.

**Task 8: Process and Visualize Streaming Data**

In this task, you save time-series data from the devices to Amazon DynamoDB tables. You create a rule with two actions, to query the incoming messages and capture the payload. The first rule will write time series data from devices to a DynamoDB table called IoTDynamoTimeSeriesTable. The second rule will write the latest received messages to a DynamoDB table called IoTDynamoDeviceStatusTable. The actual DynamoDB table names will be prefixed by *qls…*.

1. In the **AWS IoT Console**, in the navigation pane at the left of the page, under the **Message routing** category, choose **Rules**.
2. Choose **Create rule** .
3. In the **Specify rule properties** section, configure:

* Under **Rule properties**, for **Rule name**, enter

IoTToDynamo

.

1. Choose **Next** .
2. In the **Configure SQL statement** section, in the **SQL statement** code block, delete the existing text and replace it with the following:

SELECT \* FROM 'device/+/devicePayload'

1. Choose **Next** .
2. In **Attach rule actions** page, under **Rule actions** section, for **Action 1** form, configure:

* **Action 1:** Select the **DynamoDB Insert a message into a DynamoDB table** option from the dropdown.
* **Table name:** Select **GSDynamoTimeSeries** from the dropdown.
* **Partition key:**

deviceId

* **Partition key value:**

${topic(2)}

* **Sort key:**

payloadTimestamp

* **Sort key value:**

${timeStampEpoch}

* **IAM role:** Select **AwsIotToDynamoRole** from the dropdown.

**Note:** This action will write the payload to DynamoDB table using the timestamp as a range key value.

Next, you create a table of connected devices using the same IoT rule that reports the last reported value from the devices. You create an additional action to accomplish this.

* Under **Rule actions**, choose **Add rule action** .

1. For **Action 2** form, configure:

* **Action 2:** Select the **DynamoDB Insert a message into a DynamoDB table** option from the dropdown.
* **Table name:** Select **GSDynamoDeviceStatus** from the dropdown.
* **Partition key:**

deviceId

* **Partition key value:**

${topic(2)}

* **IAM role:** Select **AwsIotToDynamoRole** from the dropdown.

1. Choose **Next** .
2. Choose **Create** .

The rule and actions are now configured. In the next step you enable the APIs that read the DynamoDB table and return devices data in an API GET method.

**Task 9: Test the APIs**

In this task you test that the API works, you use a command line to read the data via HTTP. The API definition and the backing AWS Lambda function that support the API were configured for you during the lab setup. In the next section, you “hook” the APIs into a dashboard to visualize the data in the website. The CloudFormation template has already configured a production stage called **prod**.

1. In the **AWS Management Console**, to the right of **Services** menu, in the search bar, search for

**API Gateway**

 and then choose **API Gateway** from the list.

1. Under the name column, choose the available **API name**.
2. In the left navigation menu, choose **Stages**.
3. Under **Stages**, choose **prod**.
4. Review the current API configuration.
5. Choose the **Invoke URL** link.

**Expected output:** Device data appears in JSON format.



*Image description: In the preceding image, device data in json format is shown.*

1. Refresh the page every few seconds and watch the **timeStampIso** value change.
2. Copy the URL endpoint to your text editor.

You need it in the next section.

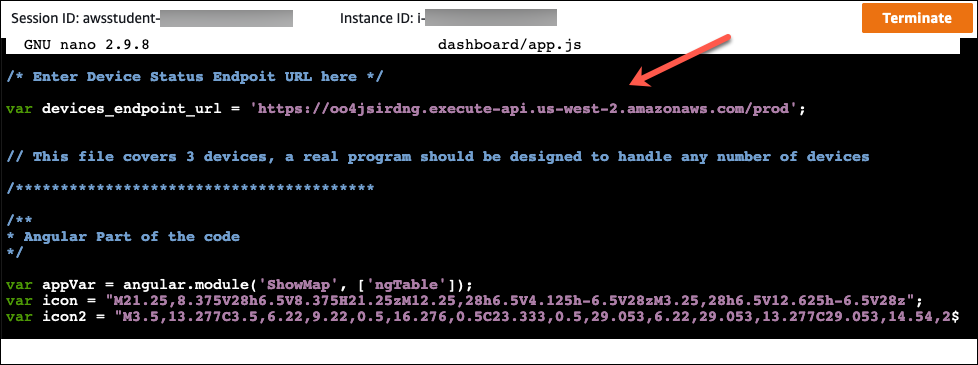
**Task 10: Deploy the Real-Time Dashboard**

In this task, you visualize device data in a dashboard. The dashboard will place the devices in a map based on Geolocation (lon,lat), Battery Charge and Battery Discharge Rate will be displayed in a line chart. First, you download the dashboard code and update the API endpoint. Last, you setup a static website on S3 to host your dashboard.

1. Open the browser tab with the console connection to the EC2 instance. If the device simulator is still running, press CTRL-C to return to the command prompt.
2. Edit app.js by entering

nano ~/dashboard/app.js

1. At the top of the file, set the **devices\_endpoint\_url** value to the API endpoint you copied to your text editor.



*Image description: In the preceding image, the content of the app.js file is shown.*

1. Save the app.js file by:

* Pressing **Ctrl + X**
* Typing

y

 and then pressing **Enter**

**Host a Static Website on Amazon S3**

In this section, you configure a static website on an S3 bucket. To host your static website, you configure an Amazon S3 bucket for website hosting and then upload your website content to the bucket. The website is then available at the region-specific website endpoint of the bucket.

1. In the **AWS Management Console**, to the right of **Services** menu, in the search bar, search for

**S3**

 and then choose **S3** from the list.

During the lab setup, a bucket was pre-created to hold the Dashboard. The bucket name should contain **iotgss3bucket** in the middle of the name.

1. Choose the **iotgss3bucket**.
2. Update your **iotgss3bucket** policy by:

* Choose the **Permissions** tab
* Under **Bucket policy**, choose the **Edit** button
* Copy and paste the following code snippet into the Bucket Policy Editor:

{

"Version":"2012-10-17",

"Statement":[

{

"Sid":"Allow Public Access to All Objects",

"Effect":"Allow",

"Principal":"\*",

"Action":"s3:GetObject",

"Resource":"arn:aws:s3:::BUCKET/\*"

}

]

}

1. Replace **BUCKET** in the bottom line, with the name of **your S3 bucket**. For your convenience, the bucket name is located to the left of these instructions. You need to copy the value of **S3BucketName**.



*Image description: In the preceding image, a sample of the preceding json snippet is shown with an example bucket name inserted in place of BUCKET.*

This policy enables anyone to read the bucket (run a GET HTTP command).

* Choose **Save changes** .

1. To enable static website hosting on your **iotgss3bucket**:

* Choose the **Properties** tab
* Under **Static website hosting**, choose the **Edit** button.
* Select  **Enable**
* In the **Index document** field, enter

index.html

.

* In the **Error document** field, enter

index.html

. For this lab, there is no error handling.

* Choose **Save changes** .

1. Under **Static website hosting**, copy the **Bucket website endpoint** URL to your text editor.

Your dashboard will be available at the bucket’s endpoint.

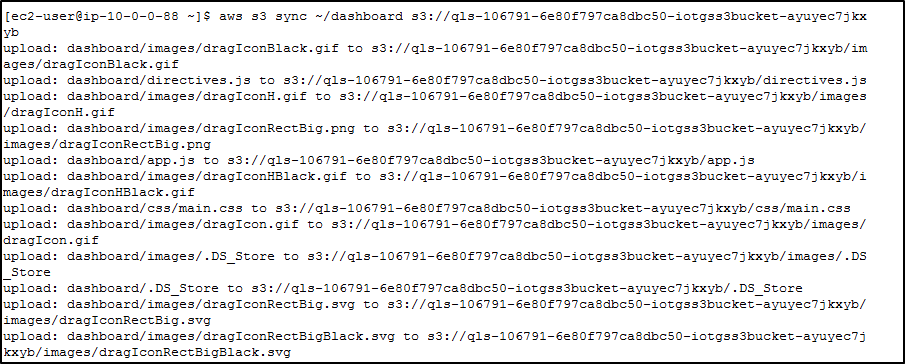
1. Copy the dashboard code to the S3 bucket by configuring the following:

* Copy the following command into your terminal window:

aws s3 sync ~/dashboard s3://BUCKET

* Replace **BUCKET** with the value of **S3BucketName** shown to the left of these instructions
* Run the command by pressing **Enter**

**Expected output:** This will copy the content of your local directory into the S3 bucket.



*Image description: In the preceding image, expected console output from the aws command is shown.*

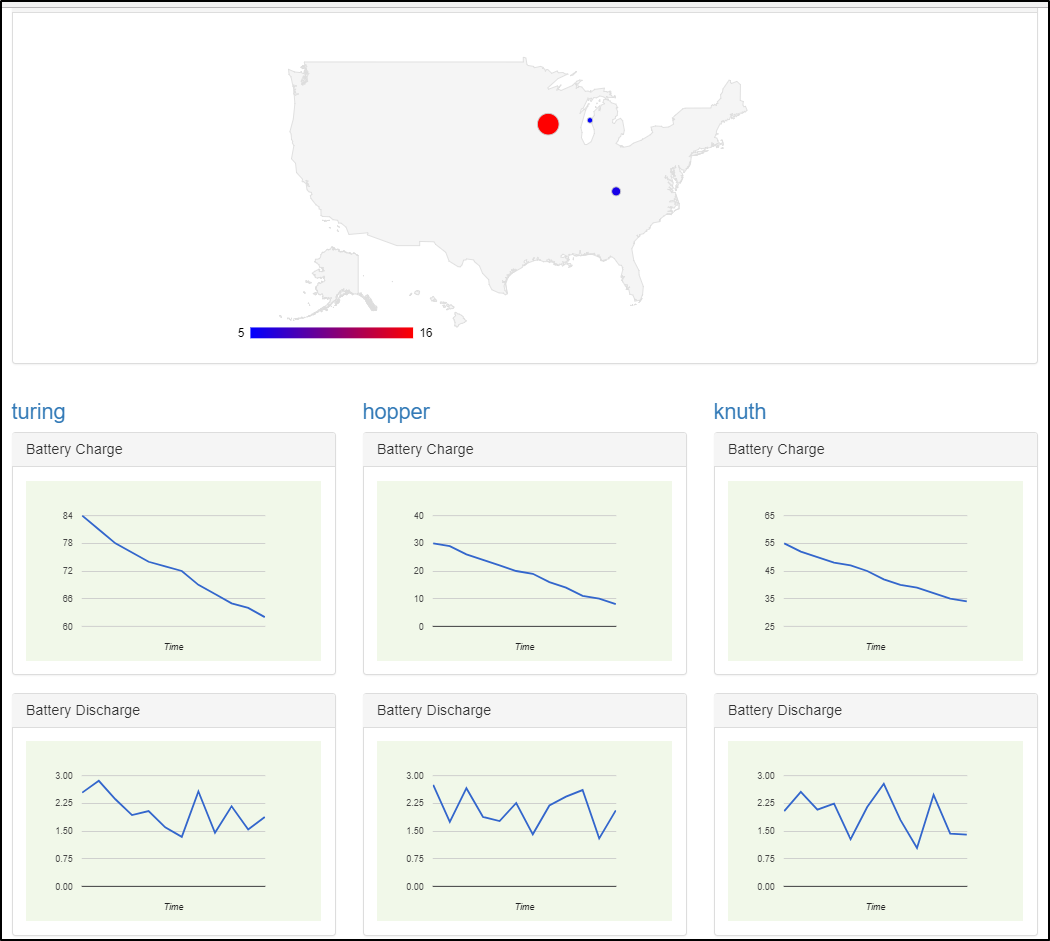
1. Restart the device simulator by entering:

python3 app.py

1. In a browser:

* Paste the S3 Bucket website endpoint URL that your copied earlier.
* Press **Enter**

**Expected output:** You can visualize the devices and its related parameters using the dashboard.



*Image description: In the preceding image, a sample of the expected website console is shown.*

**Conclusion**

 Congratulations! You now have successfully:

* Set up AWS IoT
* Ran a device simulator on EC2
* Processed and Visualized streaming data using a serverless app

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional Resources**

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).