**CSS 532 Internet of Things HW1 AWS IoT**

***Configure my Macbook Pro as an IoT device, which can communicate with AWS IoT.***

In order to configure my Macbook Pro as an IoT device that can communicate with AWS IoT, I firstly created a thing in AWS IoT console and named it as ***yiweitLaptop*** with proper policy attached. After creation completed, the device certificate, private key, public, and Amazon Root CA1 have been downloaded and saved in fold so that I could access later when connecting my laptop with AWS IoT.

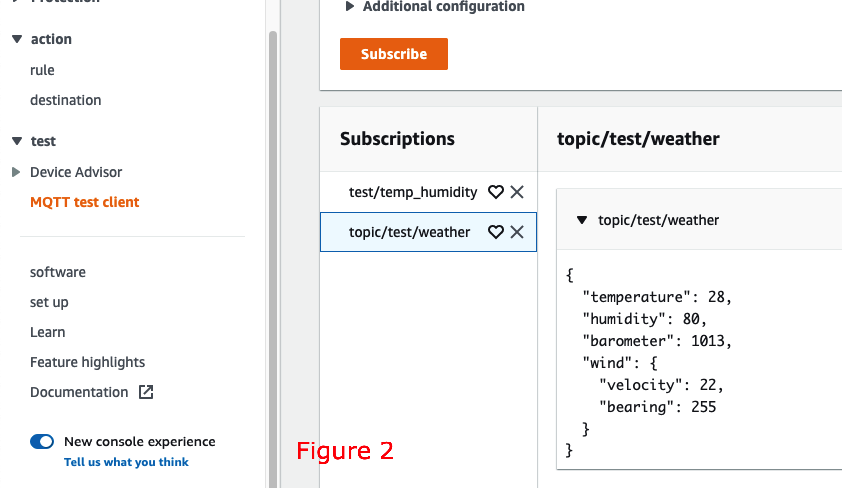
Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

***Configure AWS IoT to receive messages from your device (laptop) and show the result in the AWS IoT console***

In this step, I used the device certificate, private key, root CA and device endpoint found in AWS Setting page to connect to AWS IoT MQTT and publish a weather message to the topic ***“topic/test/weather”***. In order to see the weather message sent from my device, I subscribed the topic “***topic/test/weather”*** in MQTT test client in IoT console. After running the python script in Terminal, the message displayed in the IoT console. Figure 2 shows the message present on AWS IoT MQTT Test Client.



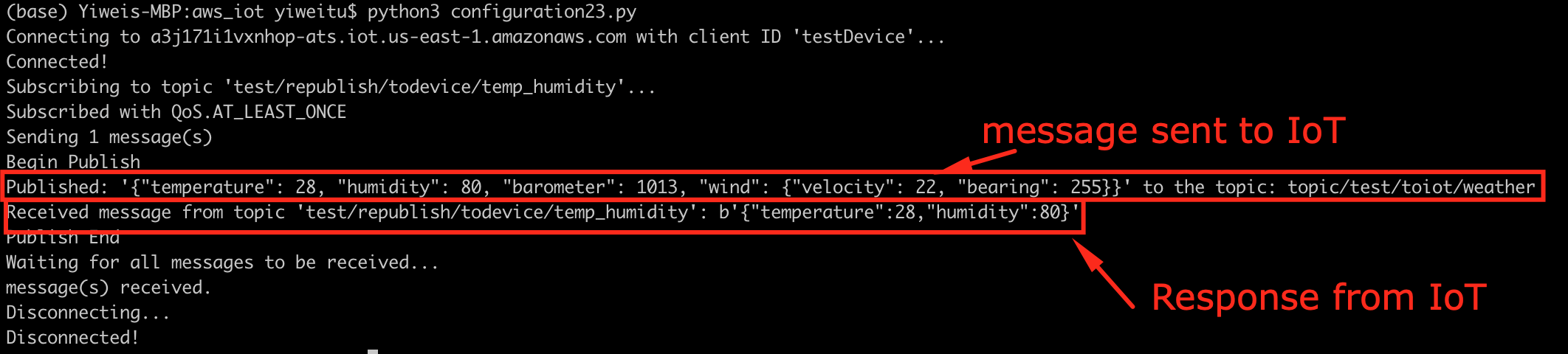
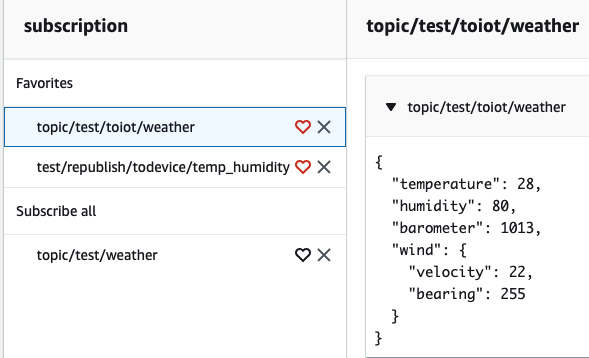
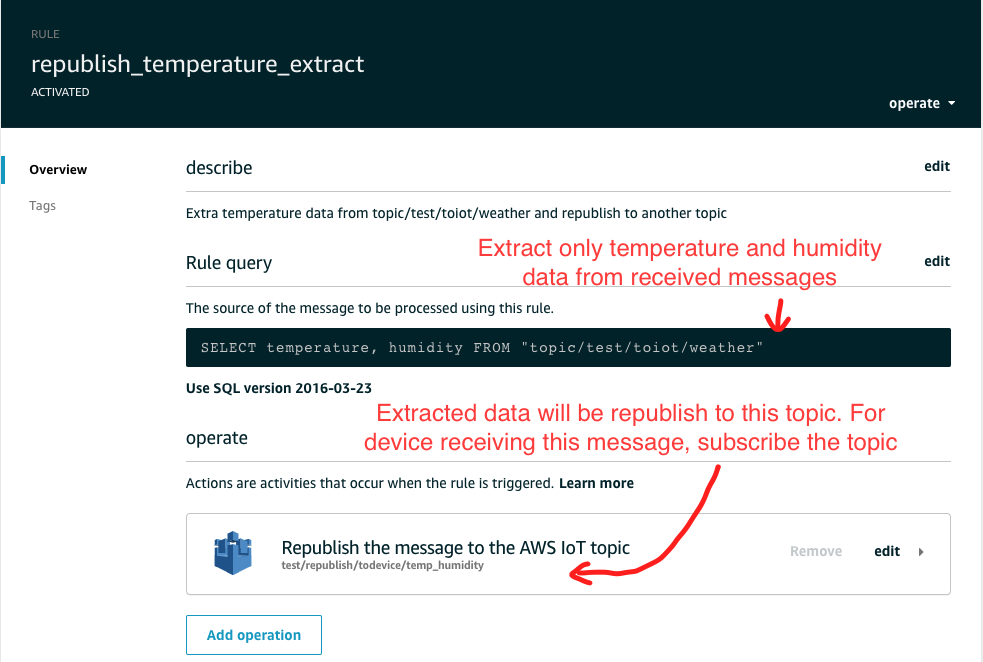
***Configure AWS IoT to automatically respond to messages sent from your device (laptop), and your device can display the received responses.***

In this section we will use two topics, one for publishing messages from device to IoT, called ***TOPIC\_PUBLISH***, and one for subscribing message from IoT to device, called ***TOPIC\_SUBSCRIBE***.

***TOPIC\_PUBLISH*** : “topic/test/toiot/weather” and

***TOPIC\_SUBSCRIBE***: “test/republish/todevice/temp\_humidity”

In the next step, I used “republish feature” in IoT Console to modify the received message from the device and republish the result to another topic ***TOPIC\_SUBSCRIBE****.* In order to do so, I firstly created a rule, called *“republish\_temperature\_extract”* with the rule query and operation shown in below figure. This rule will extract the temperature and humidity data from the message in ***TOPIC\_PUBLISH*** and then publish the result to ***TOPIC\_SUBSCRIBE***. In order to see the respond message on my device, I made my device subscribe the republish topic, ***TOPIC\_SUBSCRIBE*** and the message displayed when my device received message.

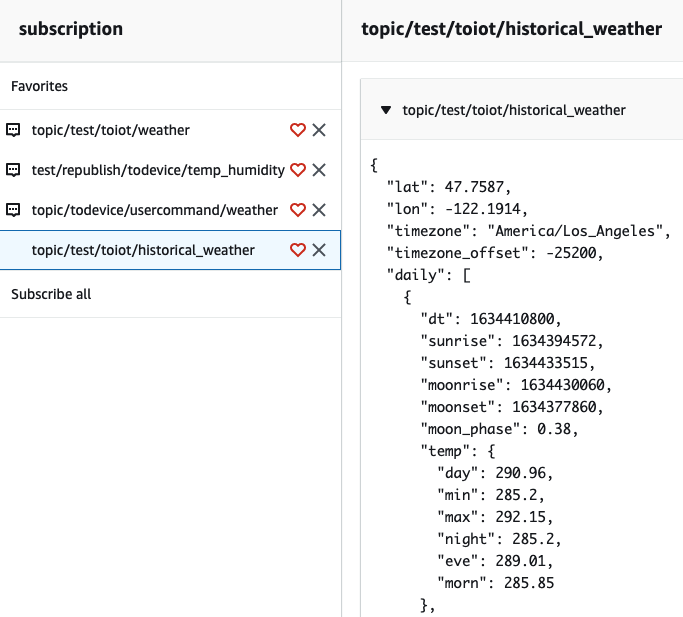
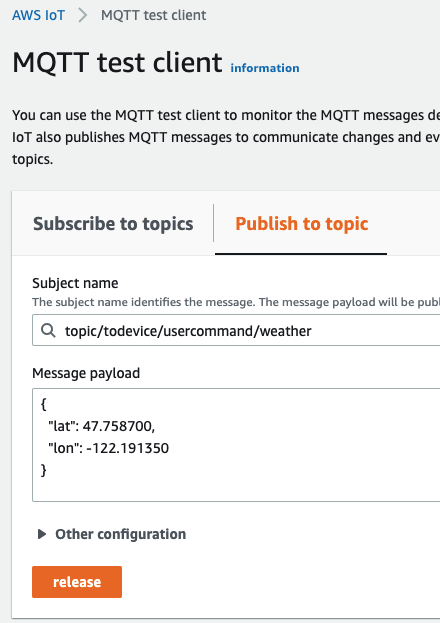


The code to process above two steps is provided in Configuration23.py in Code folder.

The last two configurations have been done together. The data flow is shown in the following image. And the code is provided in Configuration45.py in Code folder.

1. Diagram

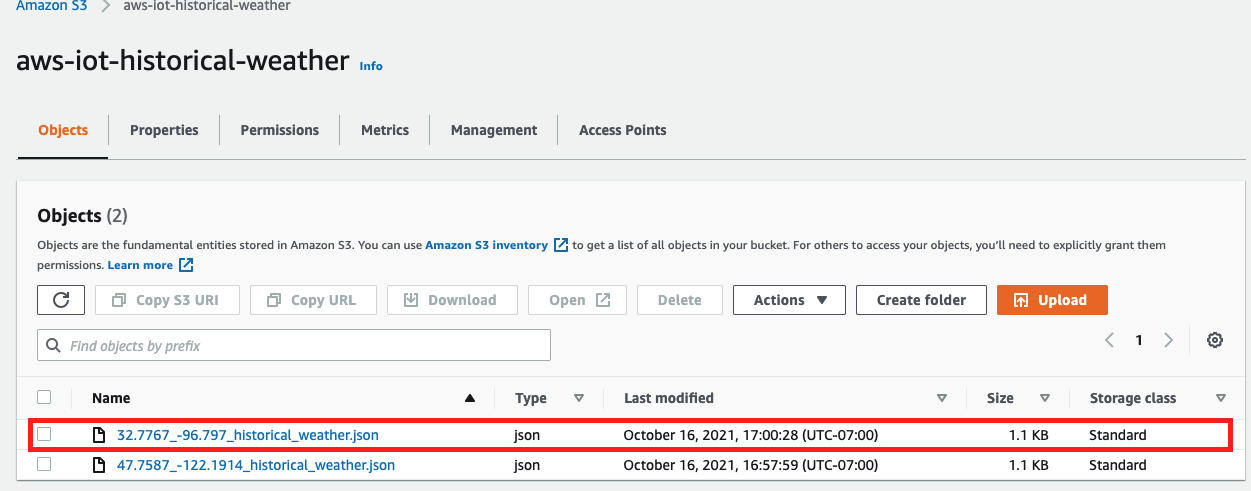
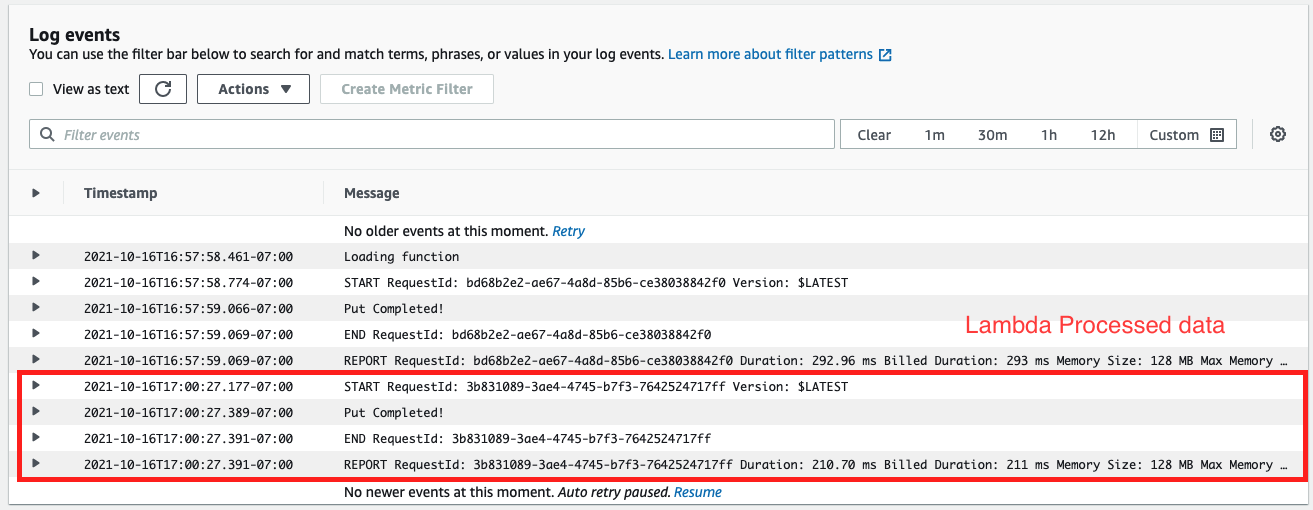
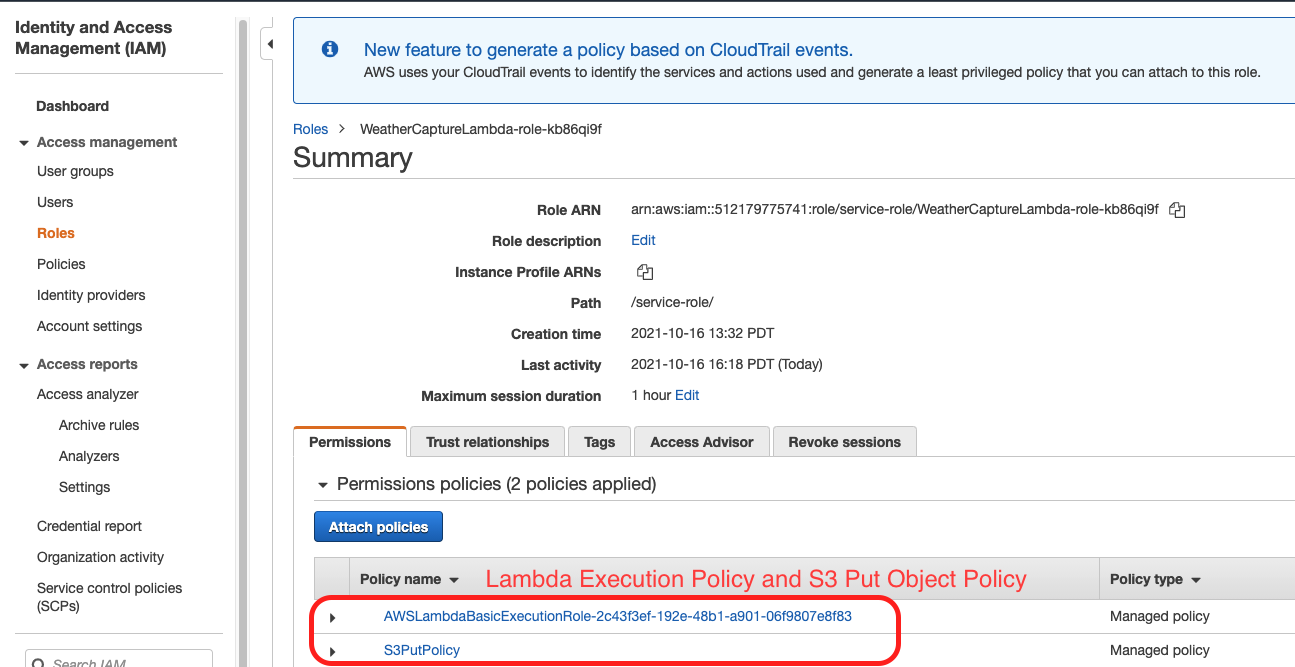
   Description automatically generatedSend user command to IoT: this step has been done via publish topic function in MQTT Test Client. I published a topic, called ***"topic/todevice/usercommand/weather"***, with the message showing above on the let.



1. Text

   Description automatically generatedSend user command from IoT to Device: In order to receive user command from IoT, the device subscribed the topic published in step 1 and extracted keywords in the command, the longitude and latitude. Used the two location keywords, I called a weather API to fetch the weather data on that location. Then I published the json message of the received data to ***“topic/test/toiot/historical\_weather”***.
2. In order to send the data received by IoT to Lambda for further data processing, I created a rule called ***“iot\_weather\_lambda”*** which will select all information from ***“topic/test/toiot/historical\_weather”*** and send it to a Lambda Function, called ***“WeatherCaptureLambda”***.Graphical user interface, application, Teams

   Description automatically generated
3. In the Lambda Function ***“WeatherCaptureLambda”,*** I captured location information, such as Latitude, Longitude, and Timezone, daily sunrise and sunset time in the given timezone and daily average temperature. Also I grouped the maximum, minimum and average temperatures for the given location as temperature stats. All of them were stored in a json object to be sent to S3 bucket. The python code in Lambda is provided in LambdaFunction\_Weather.py.



1. In order to send processed data to S3 bucket, I first created a s3 bucket named as “aws-iot-historical-weather” and a role with s3 put object policy and Lambda execution policy. Then using boto3 package in Python to upload bytes object to S3 bucket.

**Problems encountered:**

1. When programming for subscribing followed by publishing, I continuously received the error message of “myCallback() missing 3 required positional arguments: 'dup', 'qos', and 'retain'”. After many times of code modification and comparison with sample codes, this problem was finally solved.
2. I found that there are many Python packages cannot be imported in Lambda Functions, such as numpy, panda, and pytz (package to convert timezone-offset to timezone). In order to fulfill my needs, I used the alternate packages or write my own functions to perform the calculation.

**Estimated time spent on this assignment:**

1. About 1 week.
2. I used the most of my time to test and debug python code for publishing and subscribing topics and about 1 day to work on Lambda and S3.