CSS 532 Internet of Things

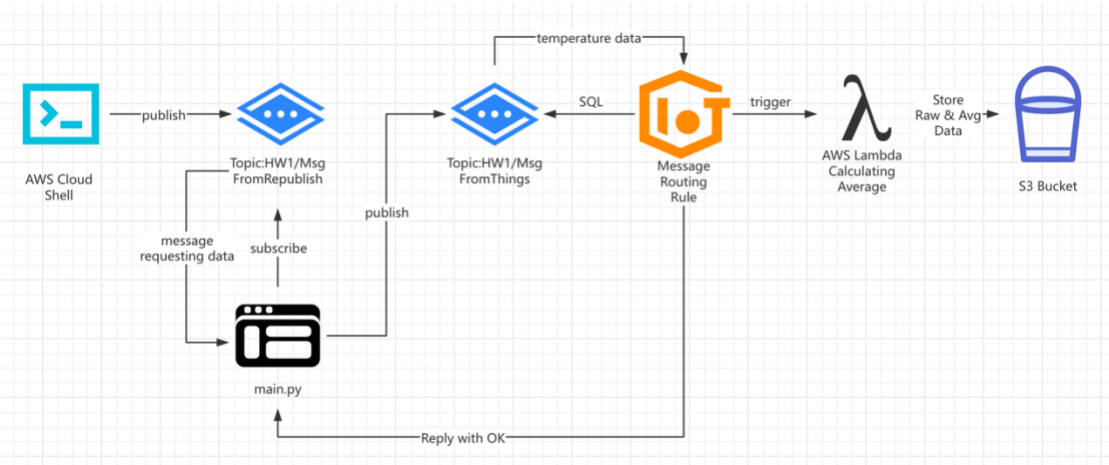
HW1 Report

Student: *Dazhi* Li; **NetID**: *dazhili*; **Student ID**: *2400330*

1.Impelemation

1.1Implementation hierarchy

The whole program starts from running main.py at first to subscribe messages from AWS CLI(CloudShell Service). If a message is sent from AWS CLI with a json form requesting temperature data from Thing(my own laptop), temperature data is going to be sent to AWS IoT Core for further processing. The customized rule in AWS IoT Core will trigger AWS Lambda service, which is going to calculating the average temperature from those temperature data and store them (both raw data and calculated average data) into S3 bucket. The whole figure is shown below.



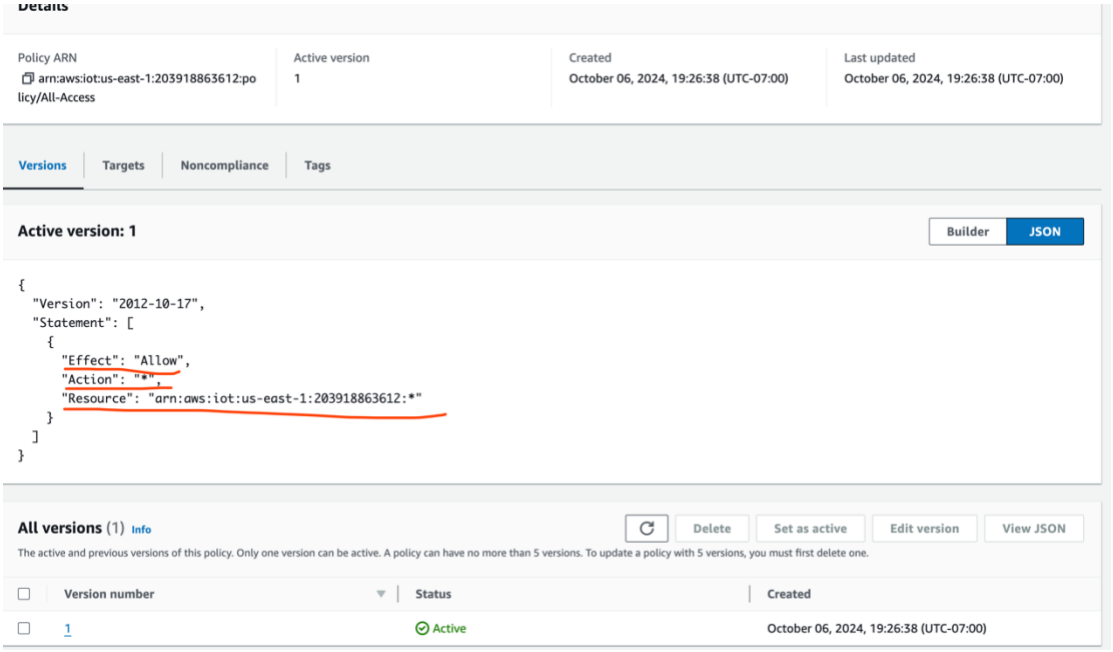
1.2Implementation Checklist

1.2.1 Configure Laptop as a Thing

It is easy to accomplish this step by clicking the left menu button “connect one device”. The most important thing is to keep the device’s private key, certification.

Besides, changing the policy of the device really matters. The policy could assign topics,

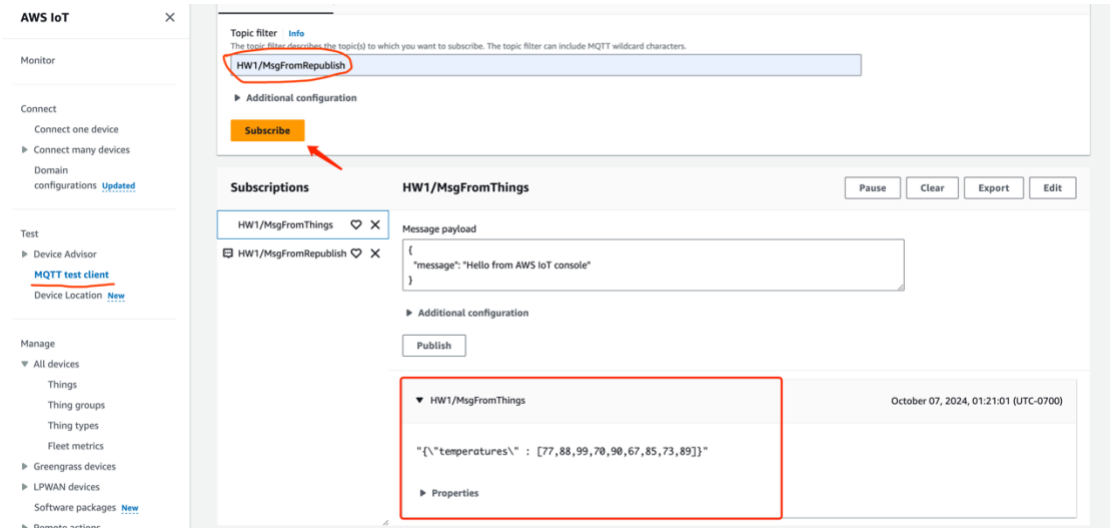
subscription and publishment behaviors.



For better debugging, I created a full access for my laptop. It allows my laptop to publish any content on any topics and subscribe any topics.

1.2.2 Configure receiving messages from things and show it

For this step, coding a MQTT5 client on my main.py program is especially important. Although IoT Core show some best practice for us, we still need to manage MQTT5 connection first. Then I tried to publish message on topic topic/HW1/MsgFromThings. To prove that my message is successfully sent, I used to MQTT test client to subscribe on the same topic. Then, the message will be shown on IoT core console.



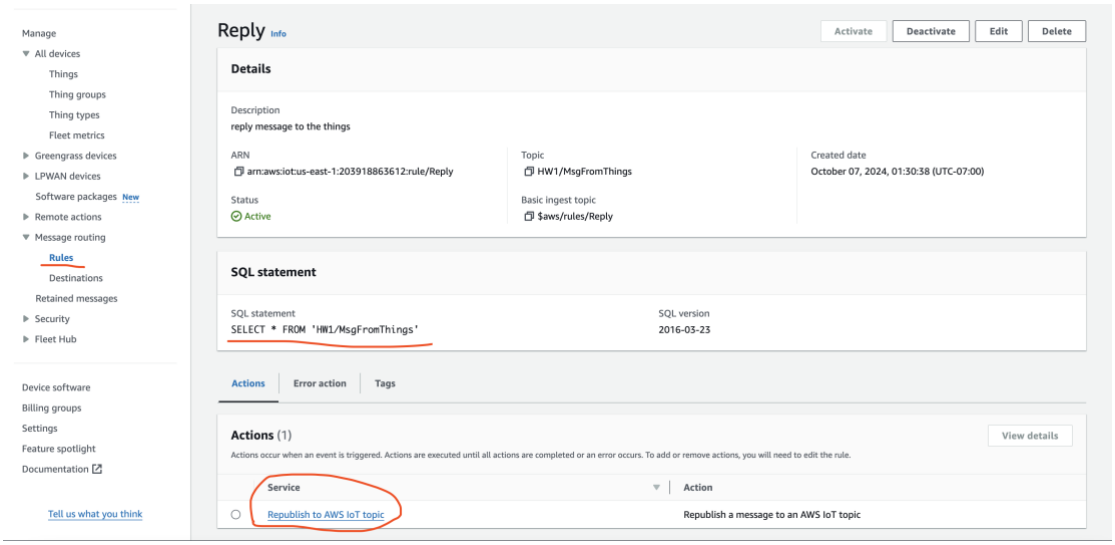
1.2.3 Configure AWS IoT Reply and Show it

For this part, the essential skills is learning how to create a message routing rules. The

republish technology is actually built-in there.

But before learning how to create rules, I utilized IAM to create an assume role for AWS IoT. This step is necessary for AWS IoT to get privileges on republish messages on MQTT.

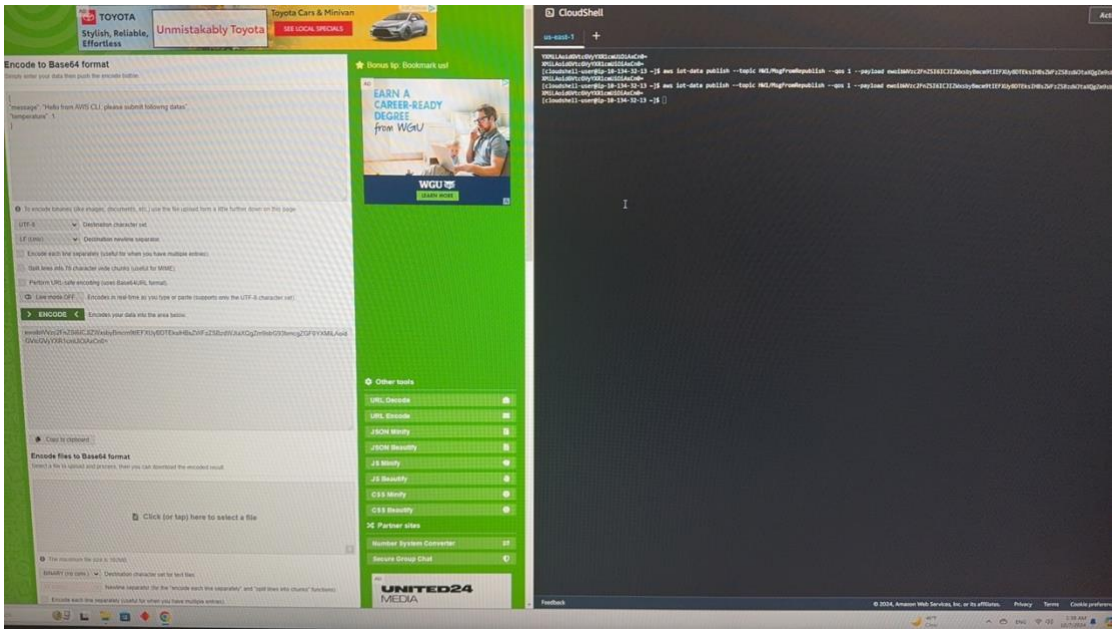
After the role was set, I began to create a rule for republish. It requires a SQL sentence to grab data from one topic and then forward it to another topic.



1.2.4Configure sending commands to Things and reply

In my understanding, a cloud consoles refers to the AWS cloud shell service, which is exactly what I used. I used the command aws iot-data publish –topic –qos –payload to publish message from the cloud console. After the message it sent, my main program will receive it by subscribing on the same topic and reply a message just like in step 1.2.2

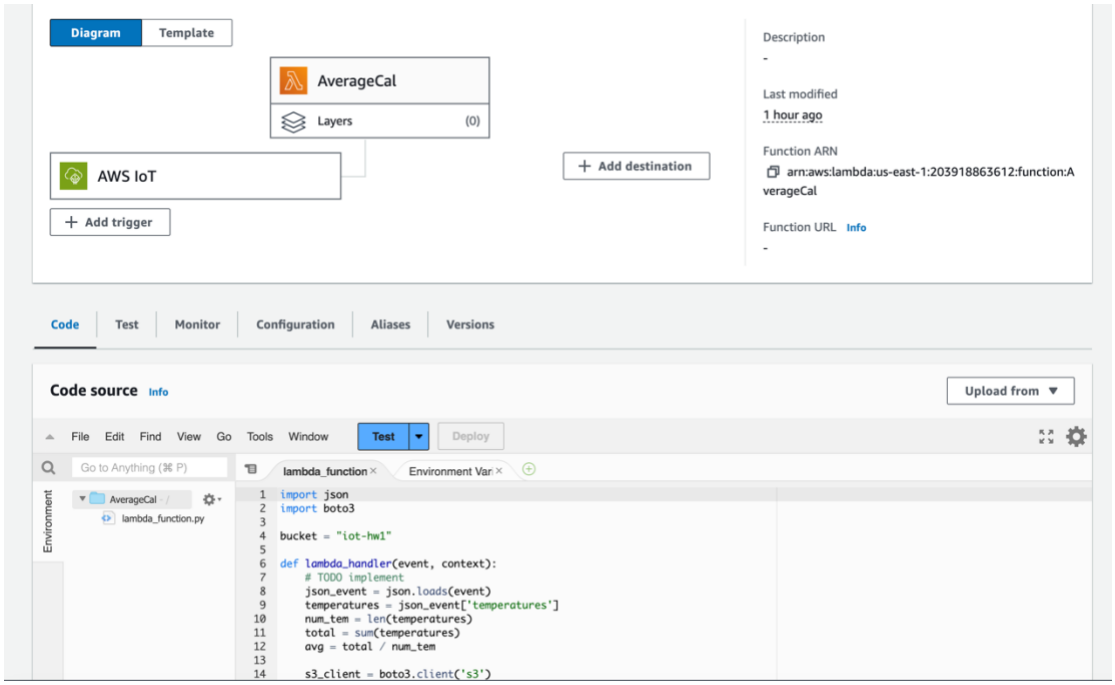
However, it should be mentioned that the payload of this command cannot be plain text. I used base64 to encode my message so that it could be published by the iot-data.



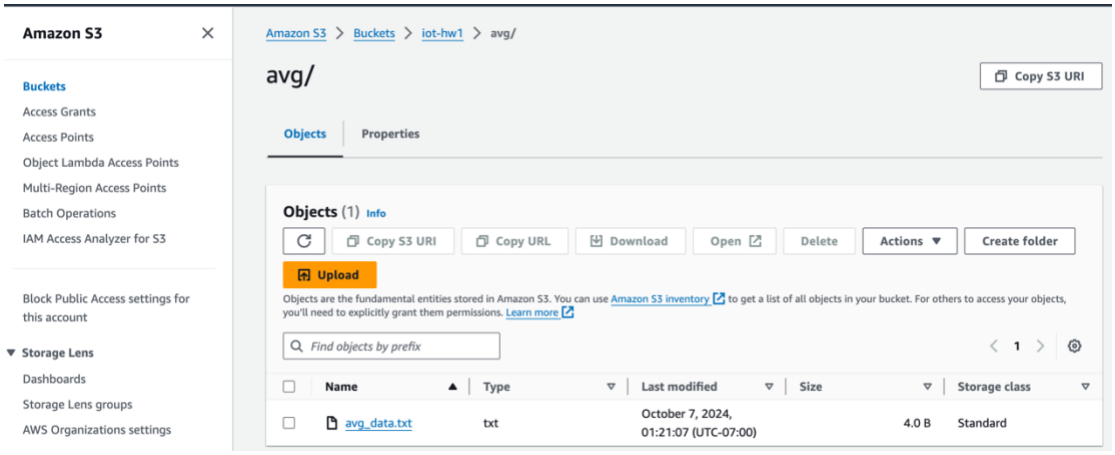
1.2.5 Configure Lambda for further data processing

The last step is creating another message routing rules on AWS IoT Core. Once the AWS IoT Core received temperature data from my laptop, it will trigger lambda service to do average temperature calculation.

Besides, lambda service also needs a role for processing data. Policy that allowing access to S3 and IoT should be attached with this role.



Once the lambda program is finished, I tried to send data to one topic, which is subscribed by a rule. This rule will forward the message to lambda. Lambda will be responsible for computation and storage.



2.Encountered Problems

2.1 How to establish a MQTT connection and subscribe/publish

This problem is a very tough at the beginning of this project. As what AWS demo shows is completely far from what we are going to achieve. I struggled a lot from adding codes one step by step to understand the meaning of each line of codes. I strongly suggest that student could begin with the connection part. Once the connection is established, student could come to publish a message and receive it on online MQTT client test.

But before this, be sure to change the default policy of the IoT devices, or we cannot connect with our desired client id and subscribe/publish message on our desired topics.

2.2Do not use us-west-2 region

I cannot even believe I could meet such kind of problem. As we know us-west-2 is closer to Washington State. On that consideration, I though us-west-2 provide lower latency and better service quality. However, when I am doing the third request which is asking us to receive republish from IoT Core, I found that I could not receive any message from IoT Core. The MQTT test client could receive the reply from the rule, which means my rule is totally correct. However, my main.py cannot receive any message. Then I started the AWS demo to see what I did is wrong. I found that the AWS

demo could not receive message from topics either. I tried many hours on thinking how could things be like that until I start a brand new demo on us-east-1.

I found my new demo works totally right on us-east-1. That made me understands that us-west-2 IoT Core Service did have some bugs in it. That problem it the most unlucky problem I have ever met recently.

2.3Document finding

I spent a lot of time on searching related documents. Actually, I found that some documents on AWS is out of date. For an instance, the role creation for message routing rule part. Also, many document is hard to find. I spent some time on searching how to establish a command on cloud console to my laptop. At first I thought it was the IoT job creation, which apparently is not the answer. Finally I found iot-data, which could publish message on MQTT.

Even if I found the right command, it is still not easy for me to implement it. There is no examples on how to use it. Only some parameters that I have to try and try again. The document does not mention the base64 encoding at all.

2.4Lambda function debug

Lambda function is really convenient, but it dose not provide any debug tools. I spent lots of time on debugging just several lines of codes. The only way to do it is reading the lambda monitor logs to find the problems.

2.5 Lambda to S3

It was very embarrassing for me to find that lambda has built-in s3 SDK dependencies. At first I compress a whole file of my lambda function with s3 python SDK included. Once the file size exceeds 10 MB, lambda inline editor does not work at all. It made me crazy on how to debug this. Also, I tried to use Access Key and Secret Key first to upload files to S3. This method exposes my credentials in my codes. After some documents finding period, I found that I could grant access directly to lambda service to prevent compromising my credentials.

3 Time Spent

I estimated the time for finishing this homework is around 6 hours. However, I spent totally 16 hours on it. It contains 2 hours study, 8 hours coding and debugging, 4 hours document searching, 2 hours report writing