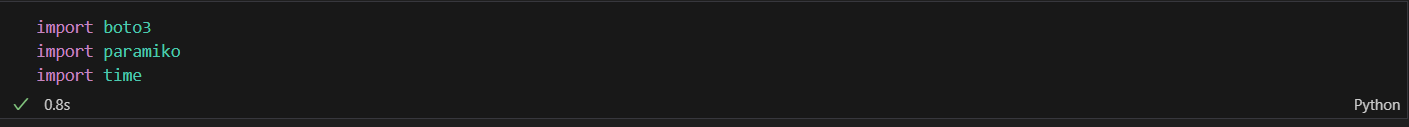
**Graded Assignment on Monitoring, Scaling and Automation**

**DEVELOP A SYSTEM THAT AUTOMATICALLY MANAGES THE LIFECYCLE OF A WEB APP HOSTED ON EC2 INSTANCES, MONITORS ITS HEALTH, AND REACTS TO CHANGES IN TRAFFIC BY SCALING RESOURCES. FURTHERMORE, ADMINISTERS RECEIVE NOTIFICATIONS REGARDING THE INFRASTRUCTURE’S HEALTH AND SCALING EVENTS.**

1. **Web App Deployment**

**Use boto3 to create an S3 bucket to store static files and launch an instance and configure its webserver.**

This question is completed via a jupyter notebook as I considered this as a easier platform to create the whole infra on while testing each cell.



Libraries imported – boto3, paramiko and time. Paramiko was imported to connect the EC2 instance SSH via the code and help run command on the instance itself.

A black screen with white text

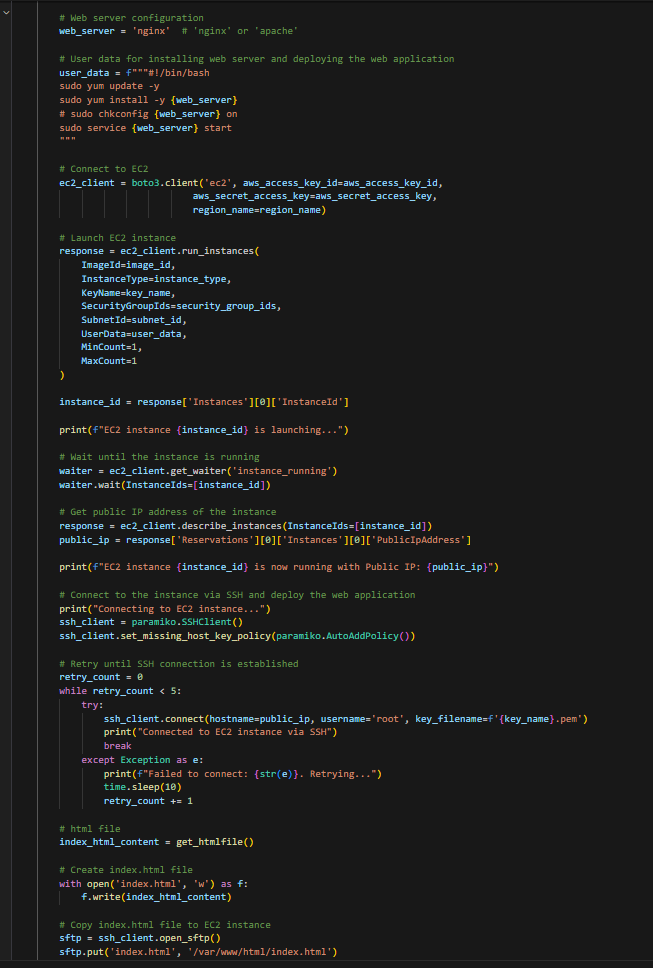
Description automatically generated

Configured my AWS CLI with my own personal AWS credentials since my authentication method (password) for the Hero Vired account isn’t working and the account also doesn’t have the right permissions.

A computer screen shot of a black screen

Description automatically generated

Created an S3 bucket. Stored a file manually in the bucket.



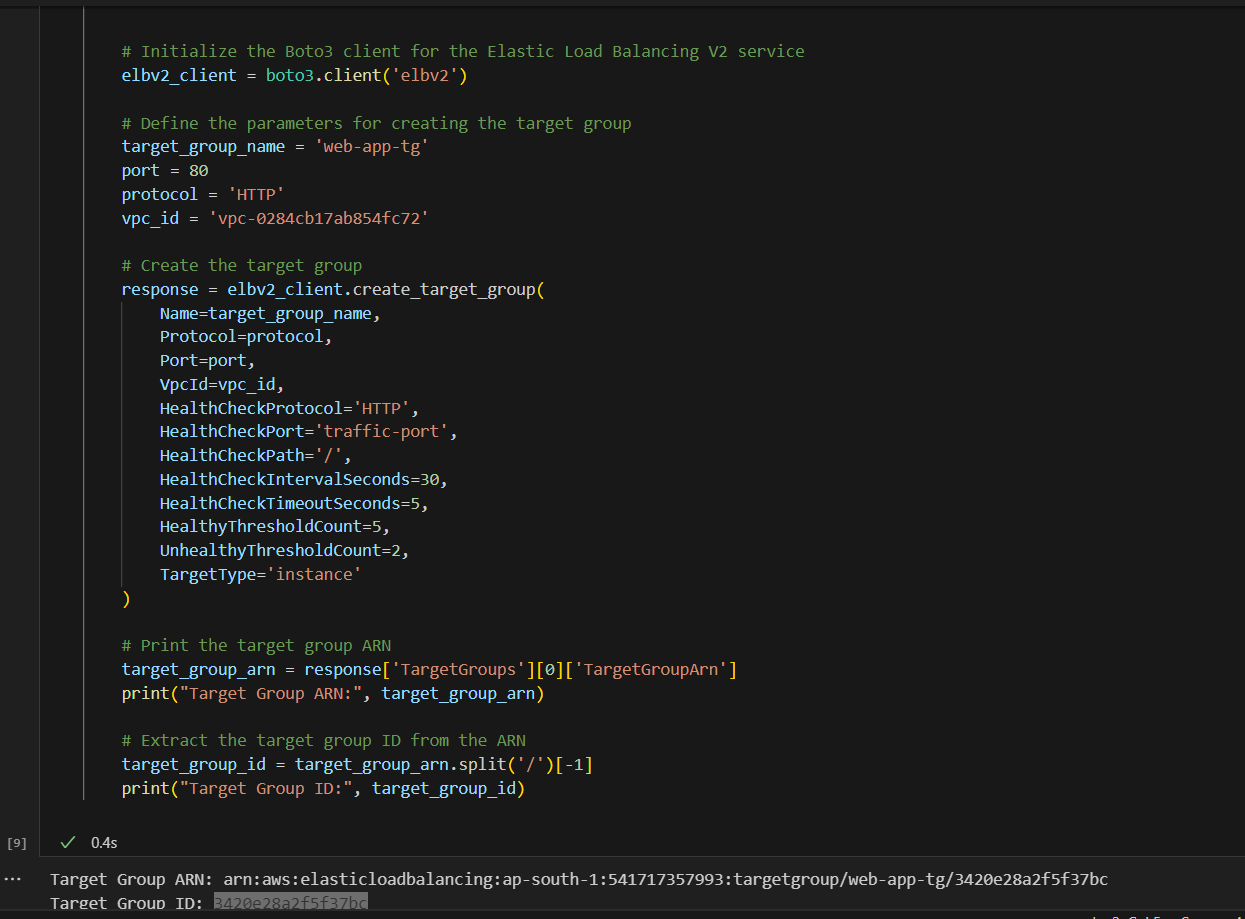
Created an EC2 instance and configured the webserver on the instance via the boto3 itself since this will help in automating the whole infrastructure.

A screenshot of a computer

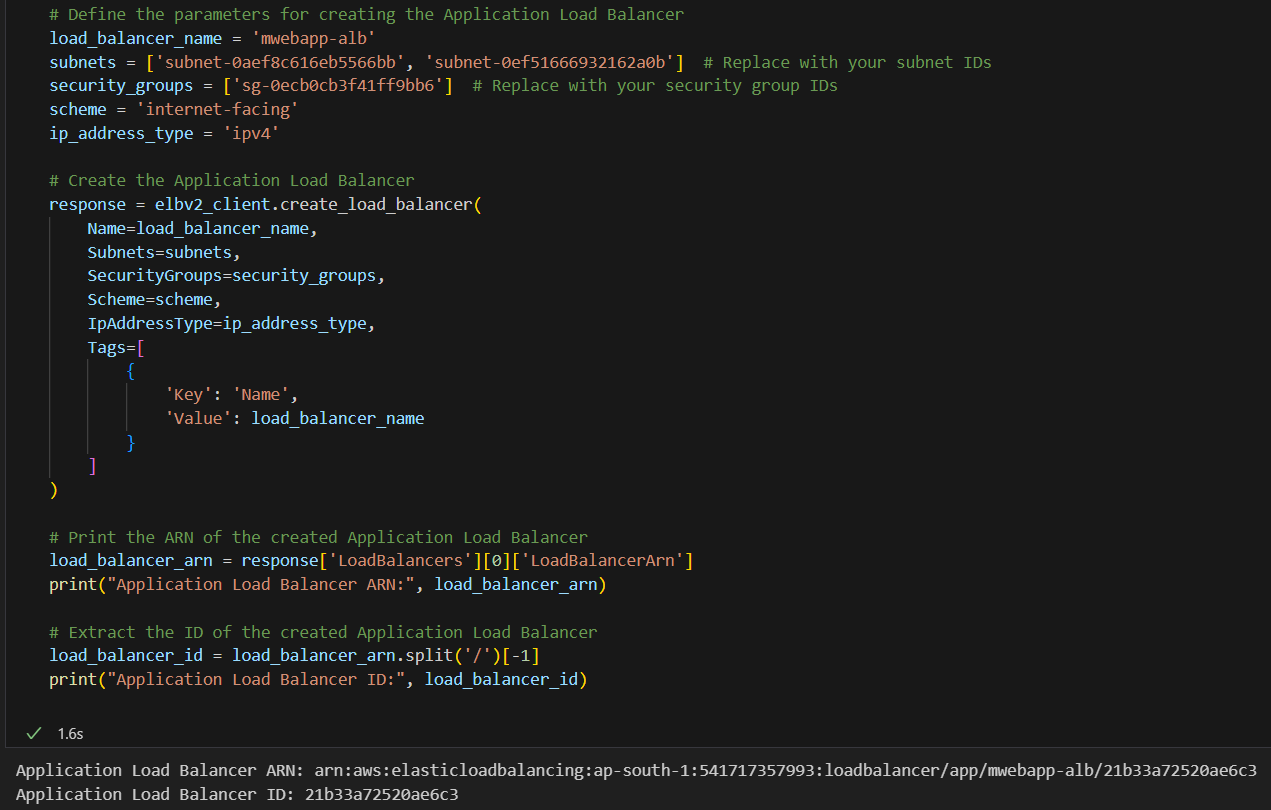
Description automatically generated

Web server running on Public IP of the server.

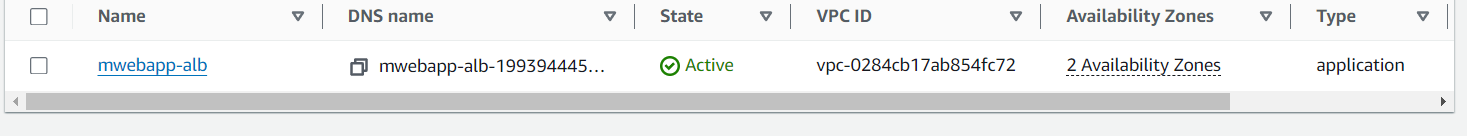
1. **Deploy an application load balancer using boto3 and register it with your instance.**

****

First and foremost, a target group was created using boto3. Printed out the target group ID to add it in the load balancer code cell.

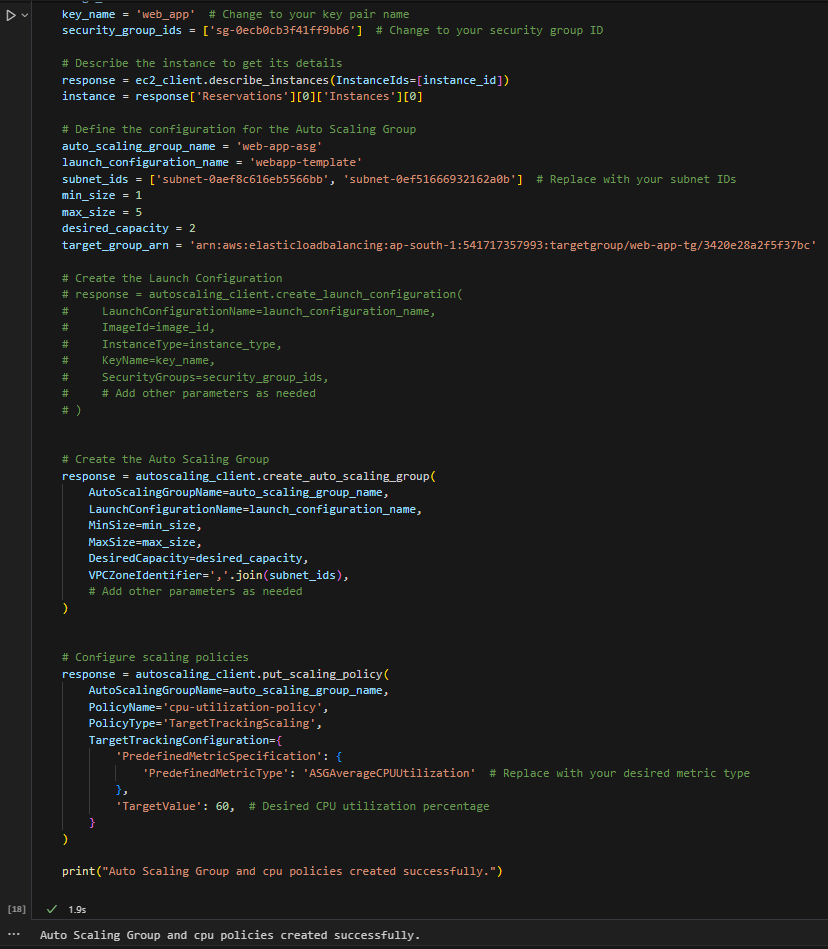


Load balancer was configured using boto3 and the everything above was also visible on the AWS Platform also.



1. **Using boto3, configure an ASG with the deployed EC2 instance as a template. Configure scaling policies to scale in/out based on metrics like CPU Utilisation or network traffic.**

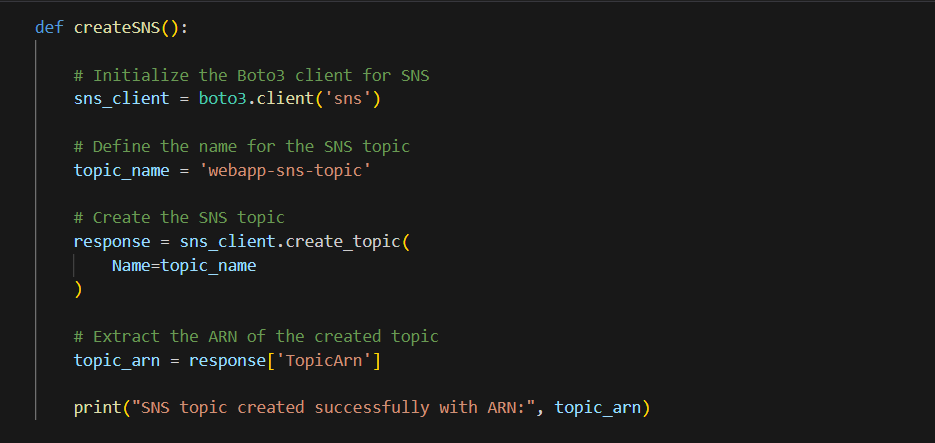
Created a launch template manually for add it in the code snippet directly for the ASG. Once done, using boto3, an ASG was created.



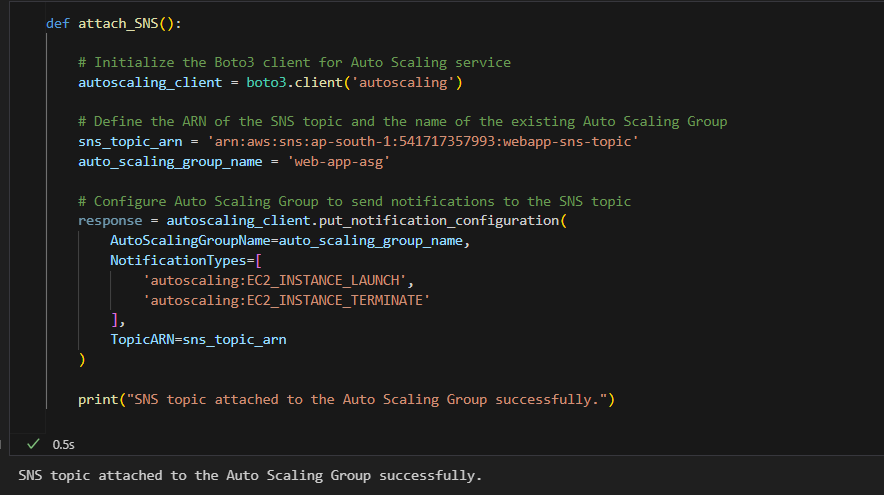
The ASG configured also has the CPU Utilisation as a metric. Predefined metric specification was used and the instance ID which is connected to the ALB was also used while creating the Auto Scaling Group.

1. **Develop a Lambda function to periodically check the health of the web app. If the health check fails, then Lambda function should create a snapshot of the failure and terminate the problematic instance. This should allow the ASG to replace it and send a SNS notification.**

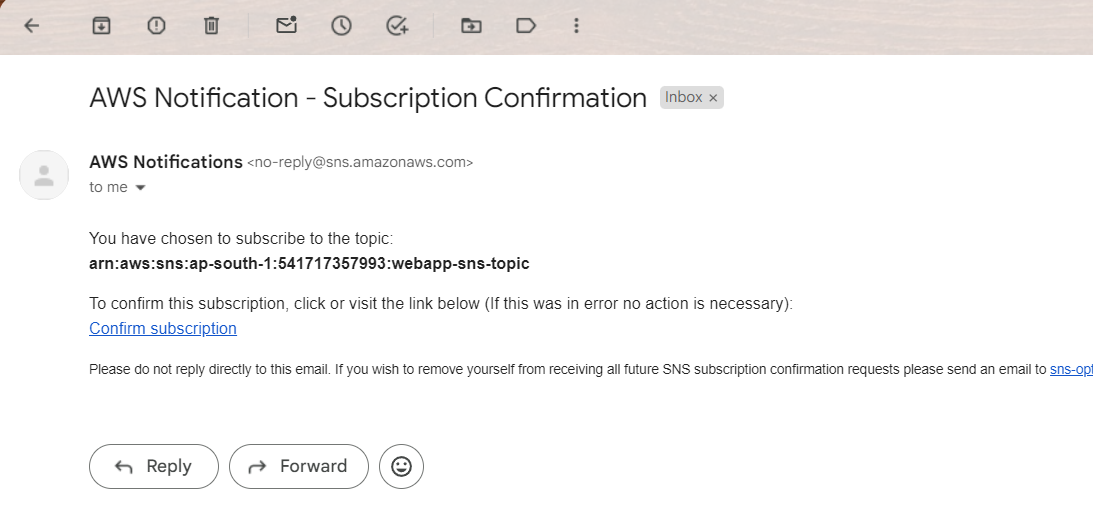
Creating a basic SNS topic to associate with the ASG. This will help when creating a lambda function to automatically notify the admin.



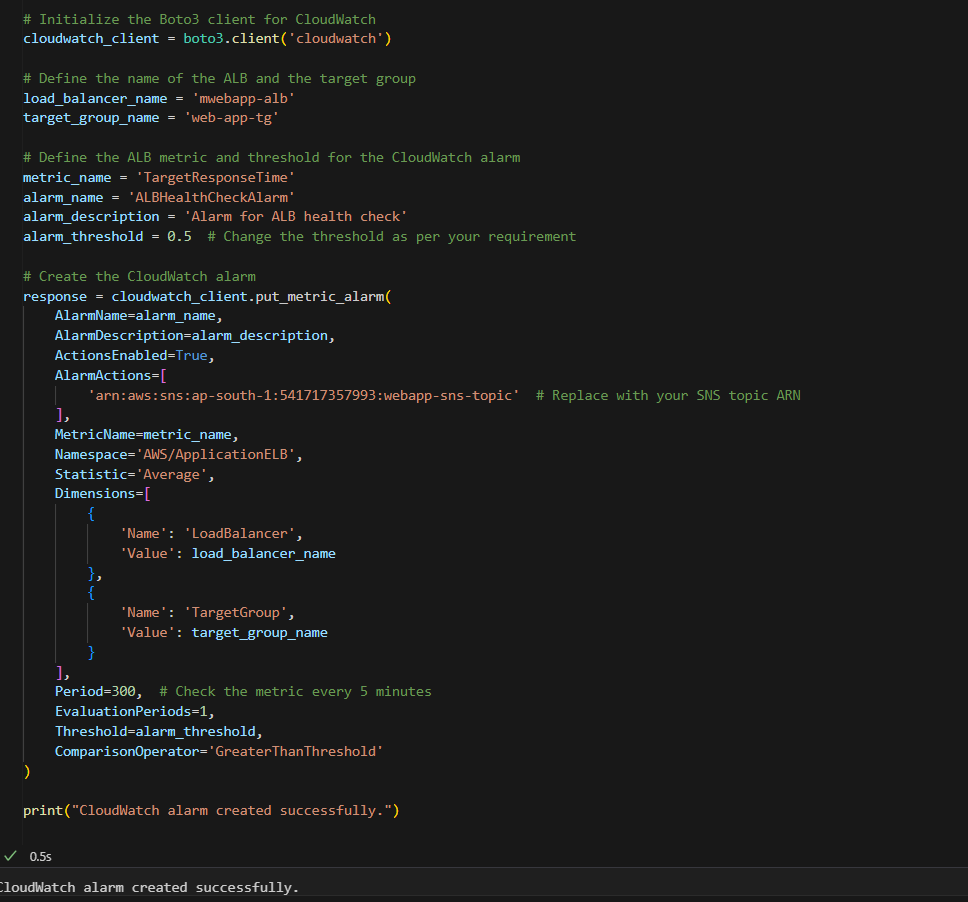
A basic SNS topic was created via this. Next up, I’ll attach this SNS to the ASG.



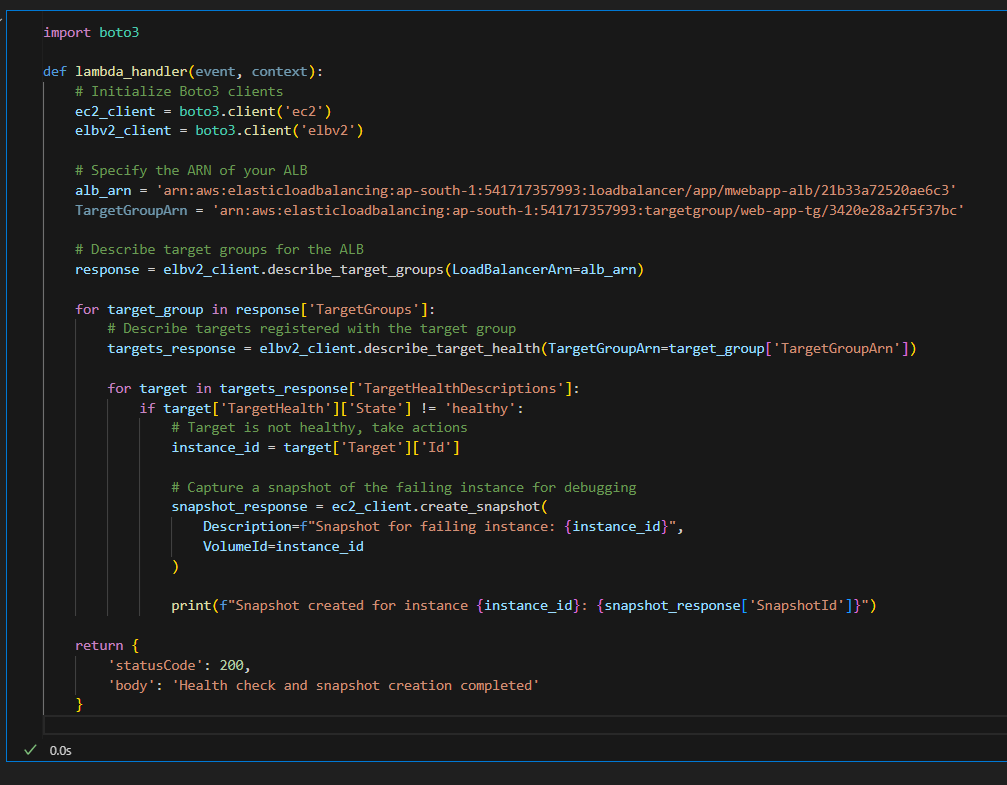
SNS attached to ASG.



SNS subscription confirmed. The subscription ID is also mentioned as follows.



After the SNS configuration, a cloudwatch alarm was configured which help trigger the lambda trigger. Once the alarm notices a health check failure of the instance, which is connected via the ALB, the lambda function will get triggered.



I manually created a lambda function in which, the lambda handler code was written using the python base. The above snipped consists of the python base code of the handler. The event of the same was configured manually.

The above lambda function does as described: Once the instance fails a health check which is monitored by the cloudwatch, it will create a snapshot of the failed health check. Then later on, ASG will terminate the instance since the health check is failed and since the SNS is attached, a notification will be sent to the admin that the instance has been terminated due to health check failure.

1. **Configure an ALB to send access logs to S3 bucket and create a Lambda function to that triggers whenever a new log is added to the S3 bucket.**

ALB was configured manually to send logs to the S3 bucket.



S3 bucket policy was created. This was then attached the IAM role of the ALB which allowed ALB to get the permission to take full control of the S3 bucket. This helped allow log the access files on the S3 bucket.

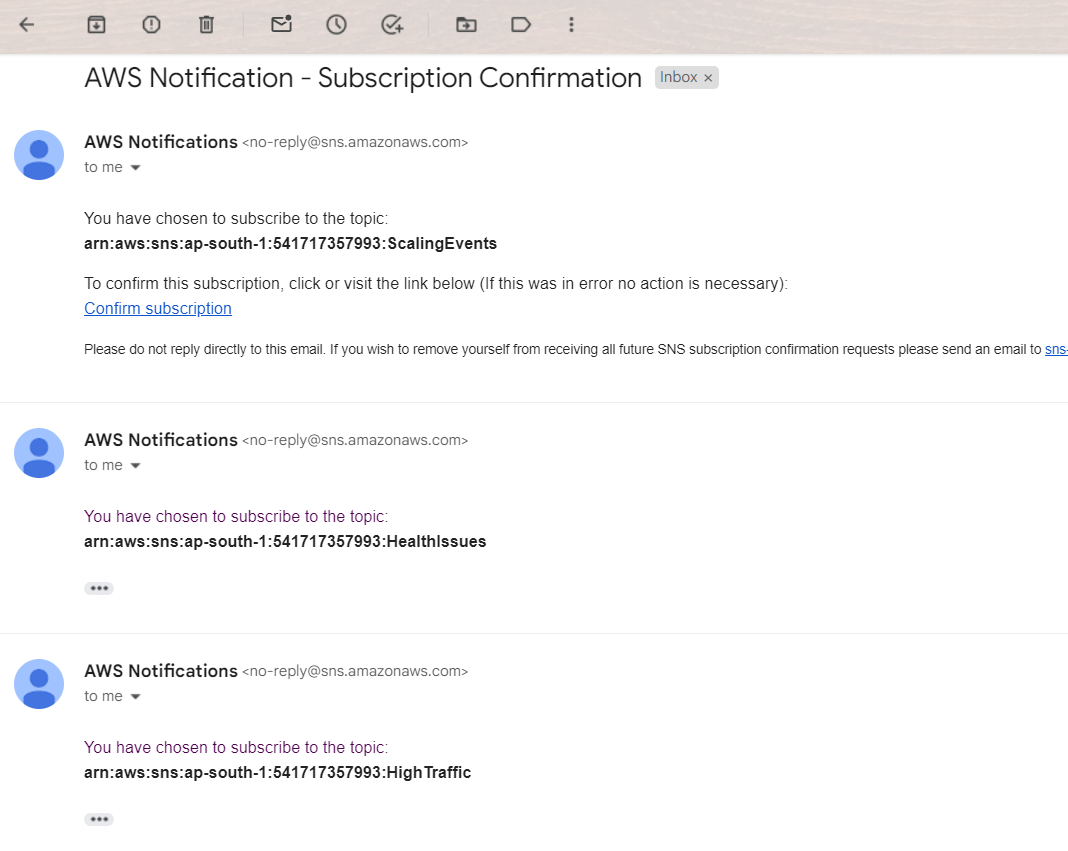
Later on, a new Lambda function was created to read the details of the S3 bucket. Whenever a new log files gets generated, a lambda function gets triggered. This will generate an alert whenever the function gets triggered.

1. **Configure various SNS topics.**

A screen shot of a computer program

Description automatically generated

Multiple SNS topics were configured all subscribed to my email ID.



All the above subscriptions were later confirmed manually.

1. **Single script to deploy the infrastructure and then tear down the infrastructure.**

A computer screen shot of a computer code

Description automatically generated

Single script created to call all the above functions.

Now to tear it all down

A black screen with white text

Description automatically generated

Some of the functions were deleted prior to writing the code to prevent the billing charges on my personal account so might not work. Since multiple things have been hard coded, **this is not a completely automated infrastructure but an attempt from my side to try to automate it as much as possible**.