SDE Assignment 3

Part A.: Creating Application

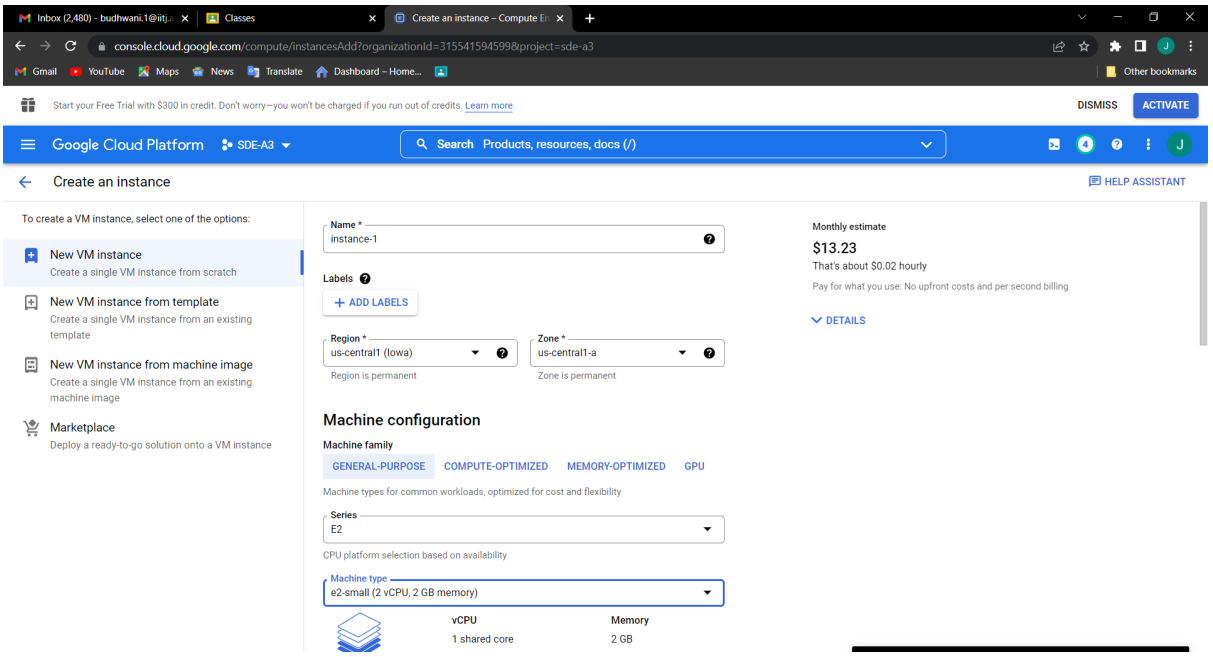
In this Assignment, we were required to create an application using three google cloud services which were **Cloud Compute service, Cloud Storage, and Cloud AI service**.

First, we need to **Create a Project SDE-A3** and then do all the below steps in created project.

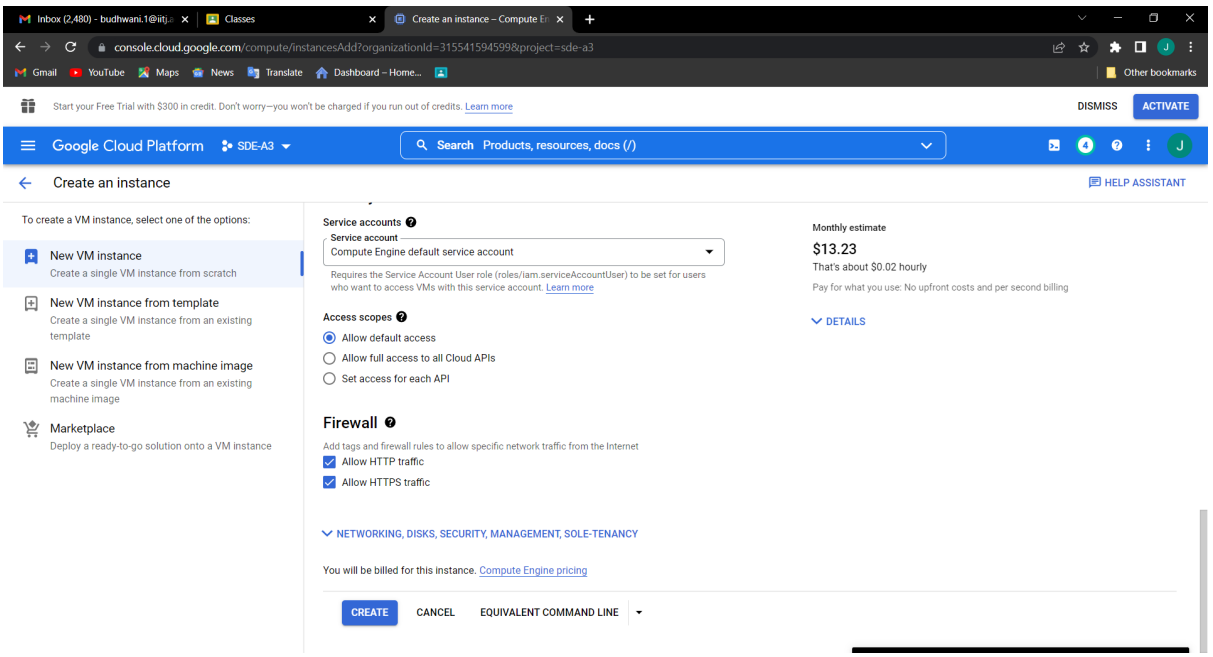
Creating Services:

First Create a **Cloud Compute service** and host our site:

1. To use the compute engine first create an instance of the engine.
2. Go to **compute engine** google cloud -> **create instance** -> give instance name and select appropriate CPUs and memory required for your application.

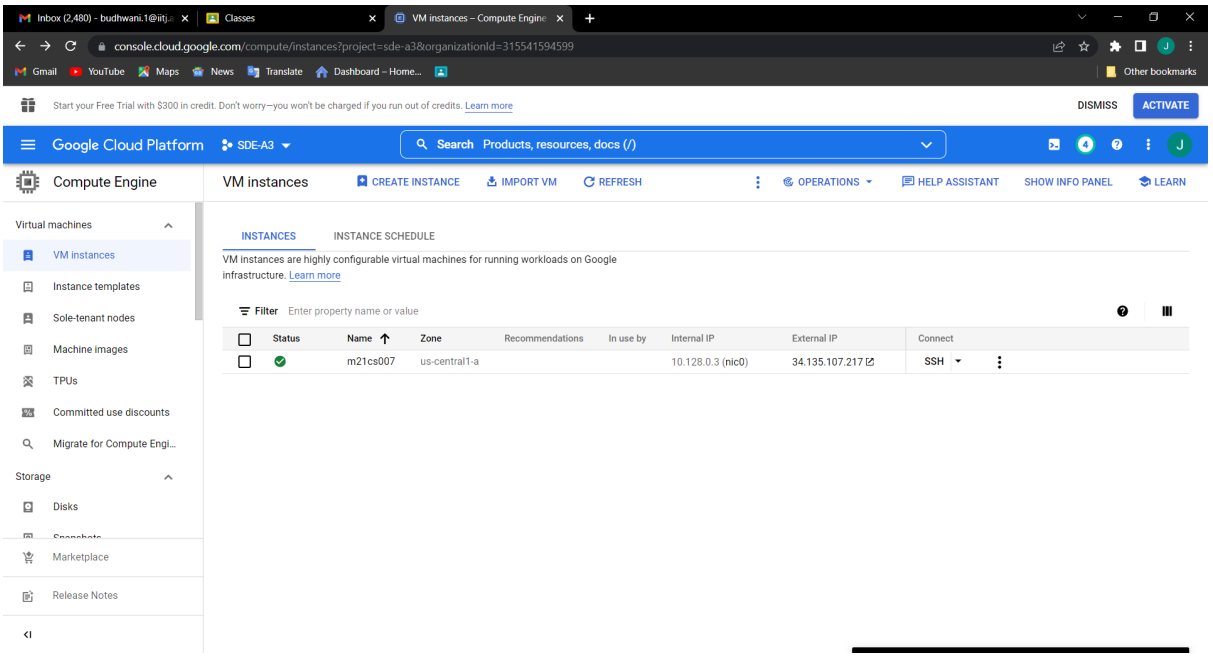


3. Under the firewall, the option **enables HTTP and HTTPS** to enable requests and responses.

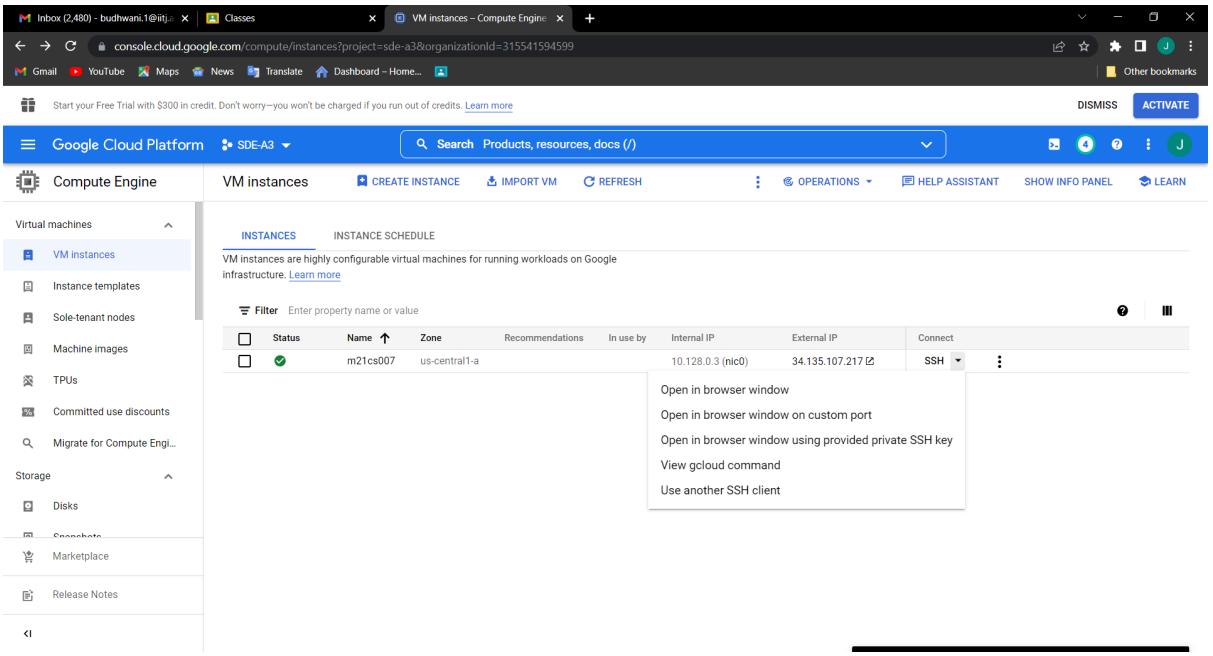


4. Finally, creating the instance will take some time.

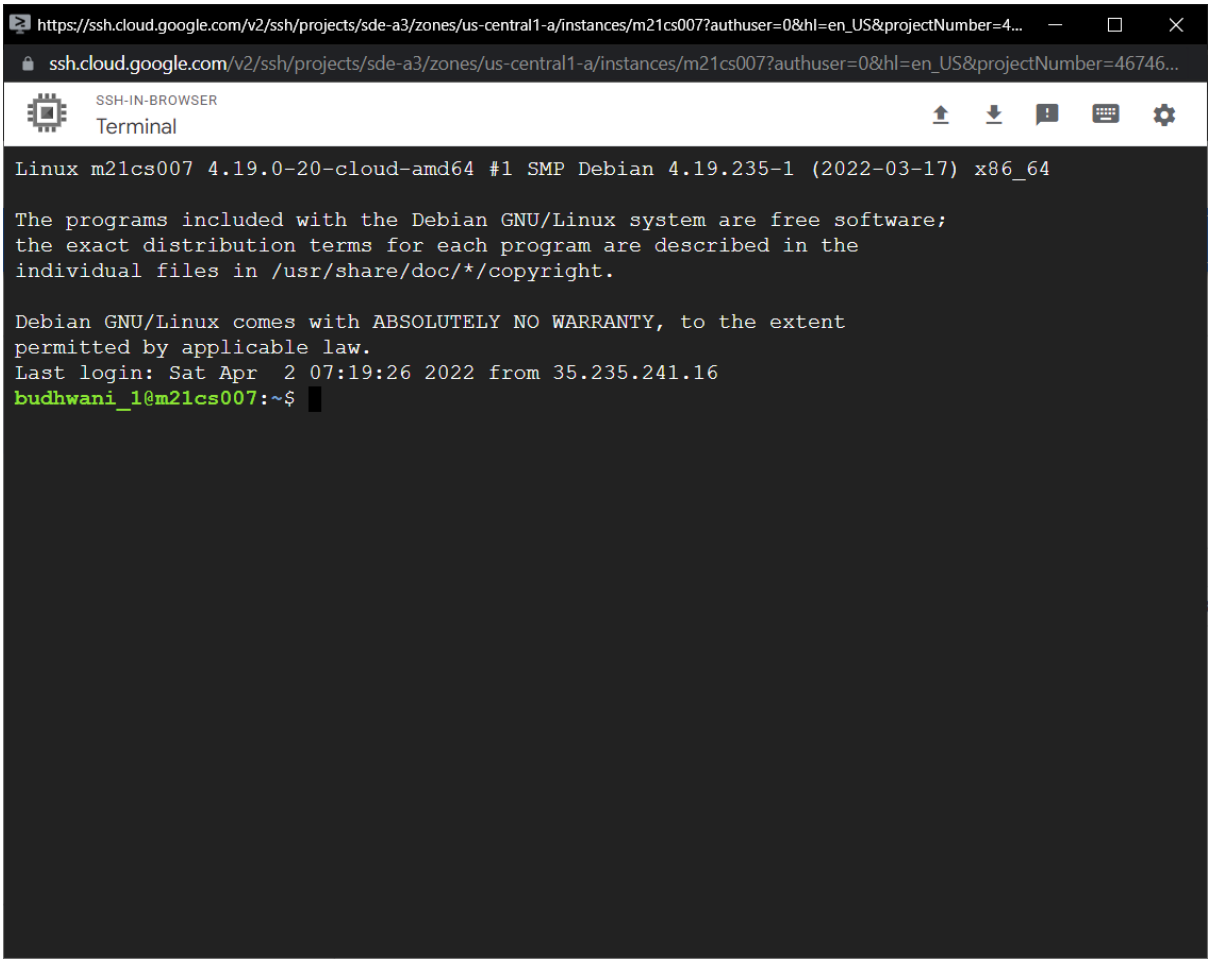
5. After the instance is created you will see a screen below.



6. After creating the instance connect to it using the ssh connection by any of the options provided below.

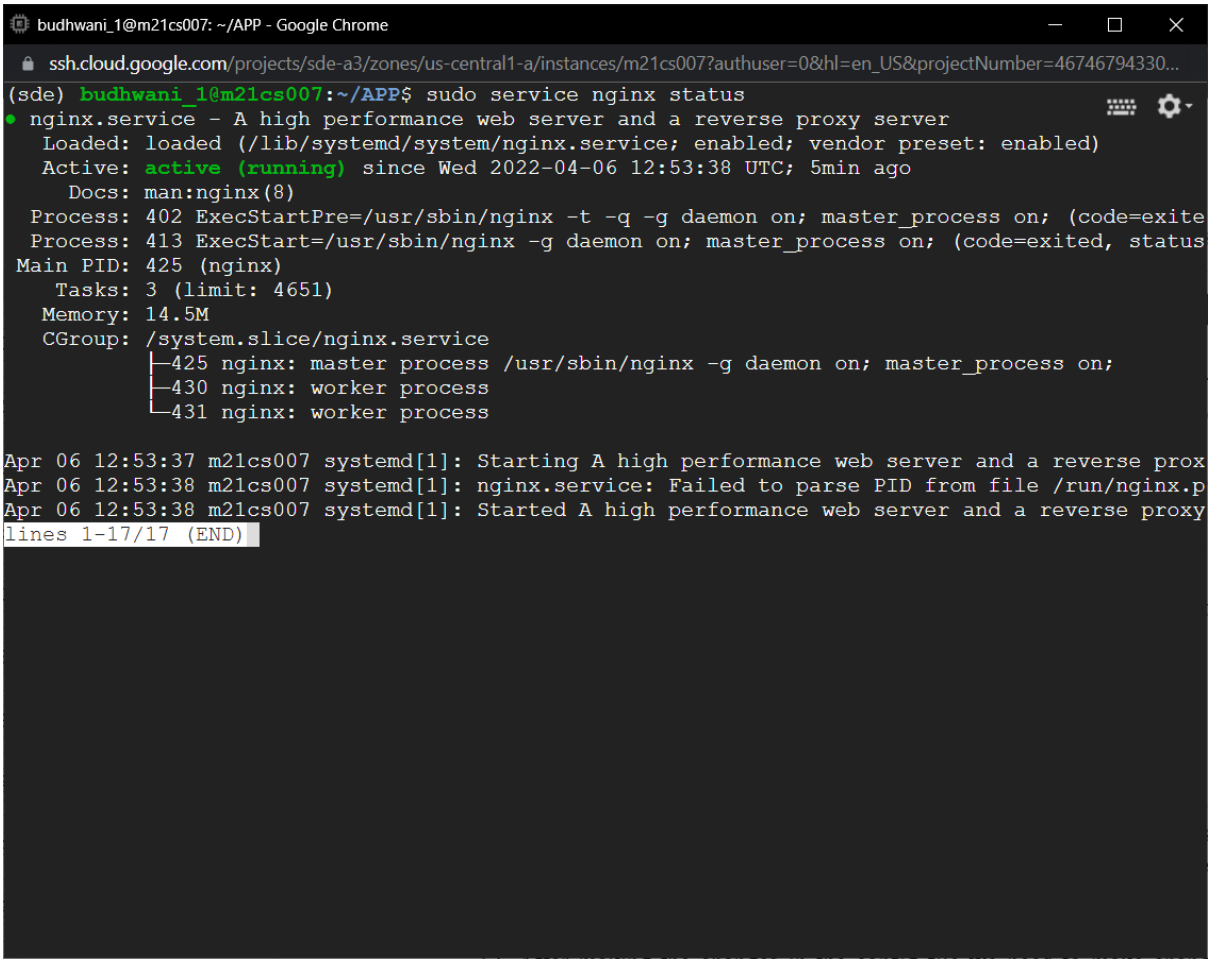


7. For this assignment, the open in the browser window is selected After selecting you will see a screen as below.

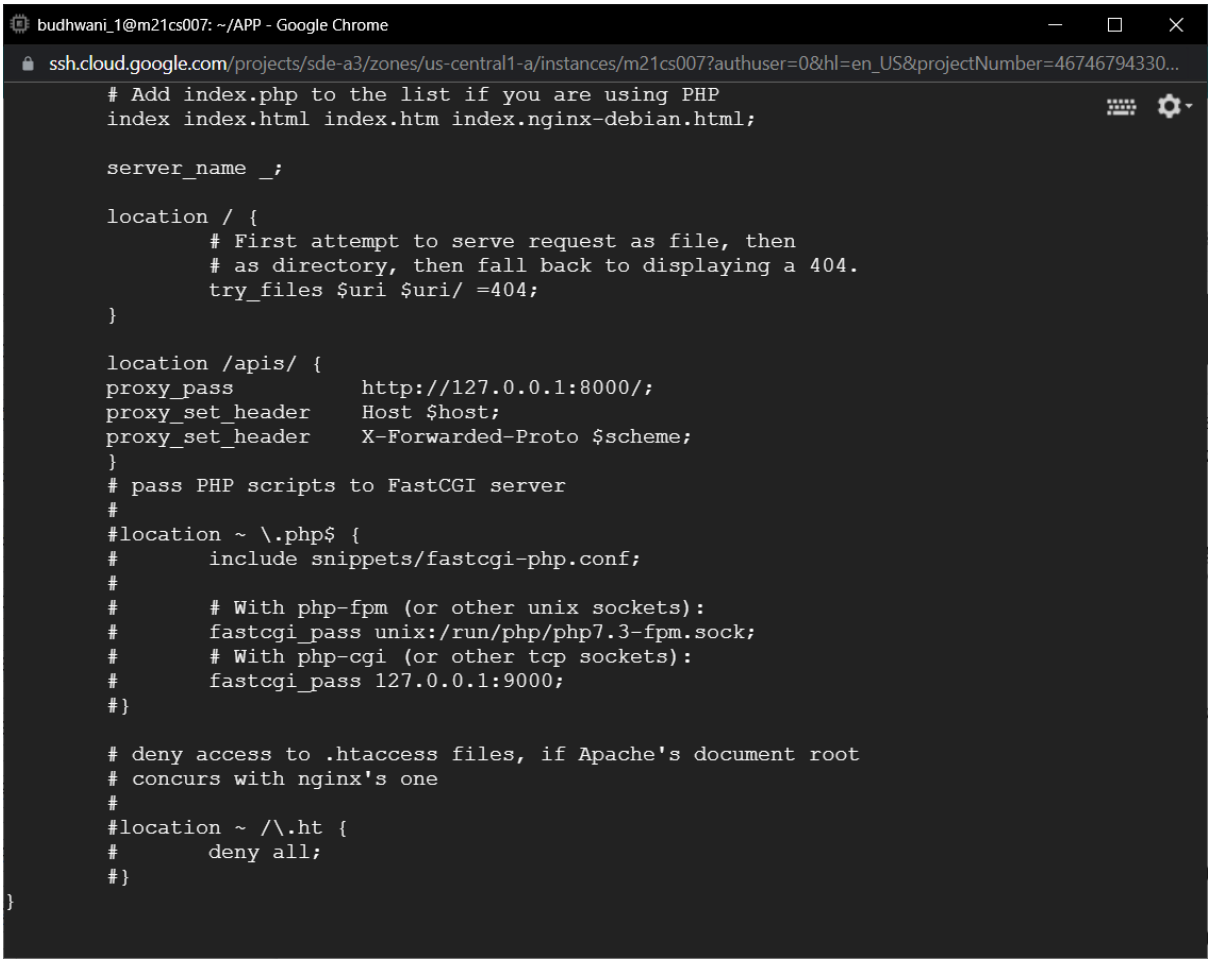


8. Now create a Folder named **APP** that contains all of our required libraries and codes.
9. For coding, the **Flask framework** is used and for hosting the Flask application **gunicorn** with **Nginx server** is used.
10. For setting up the environment reference used the **link** or Commands used for setting up the environment are:
    - Update the shell to recent libraries - **sudo apt-get upgrade && sudo apt-get update**
    - For Installing the latest version of python - **sudo apt-get install Python3**
    - For Installing the latest version of pip - **sudo apt-get install Python Python3-pip**
    - For Installing Nginx - **sudo apt-get install nginx**
    - For starting Nginx - **sudo service nginx start**
    - For checking the status of nginx - **sudo service nginx status**

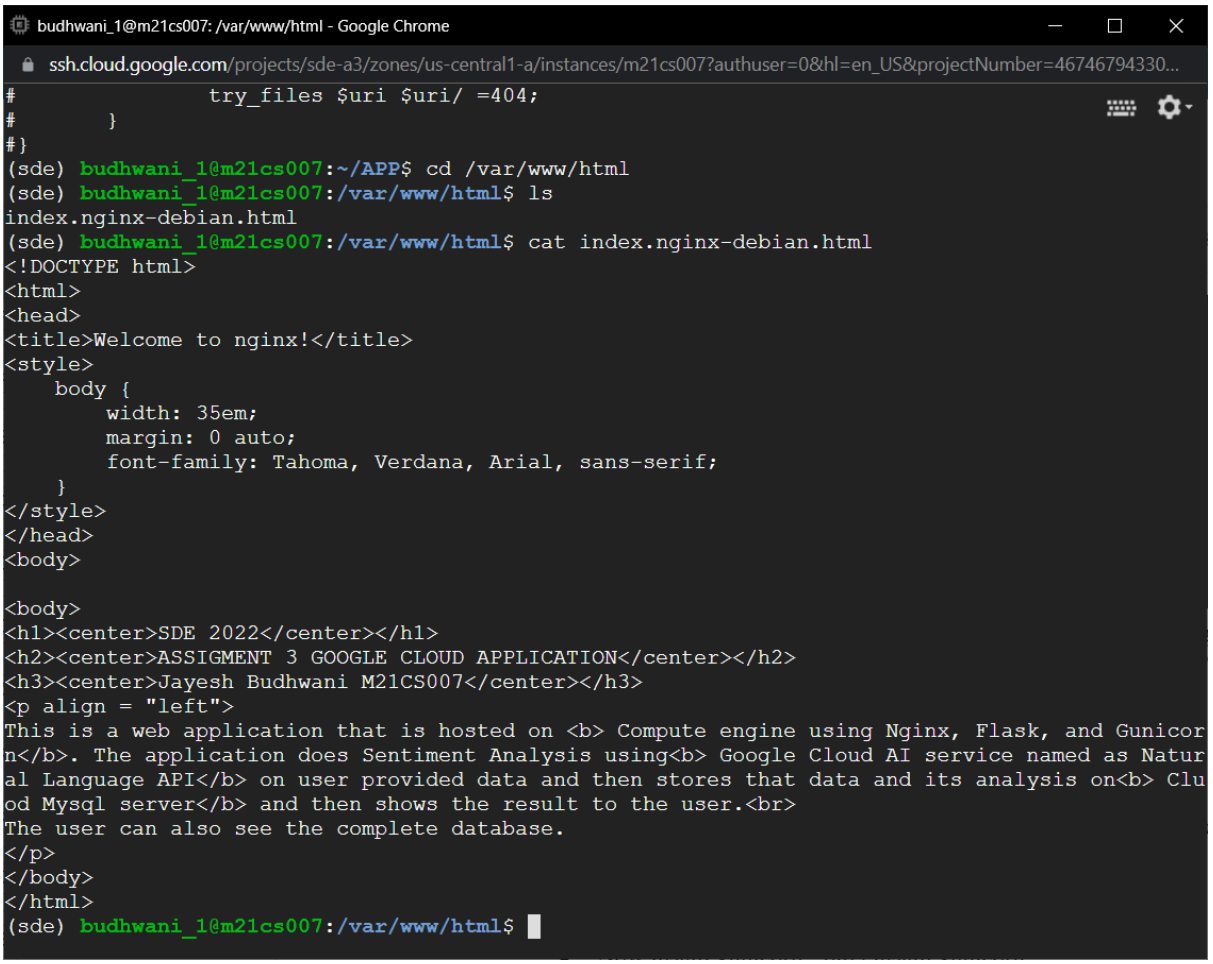
After the previous steps status of nginx looks like this.



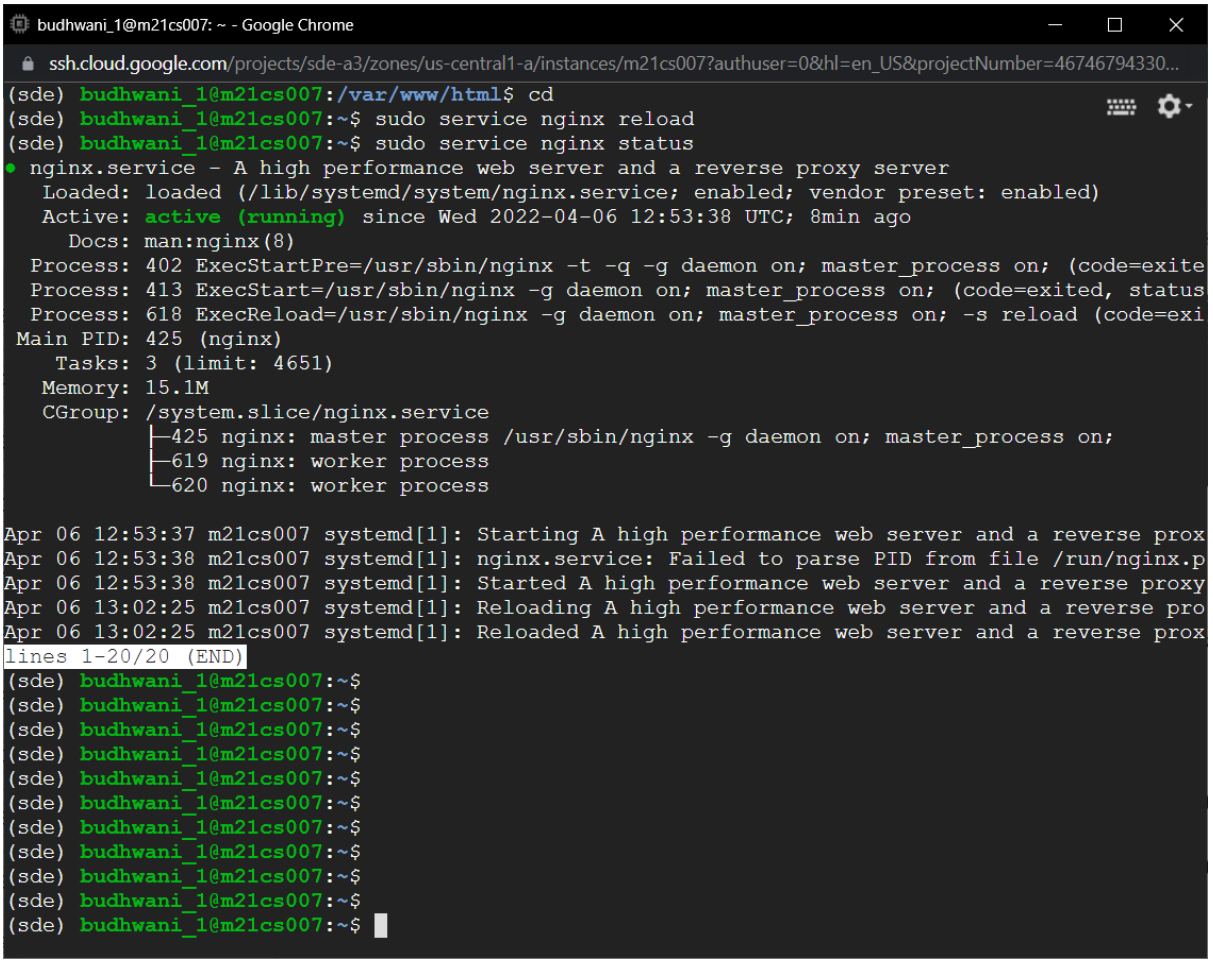
11. Now that we have installed the nginx we need to provide the request points in the conf file which can be found at **cd /etc/nginx/sites-available/default** for the assignment **/apis/** is used and in **/apis/** further bifurcation is made for each operation after making changes we further need to again start the nginx server.



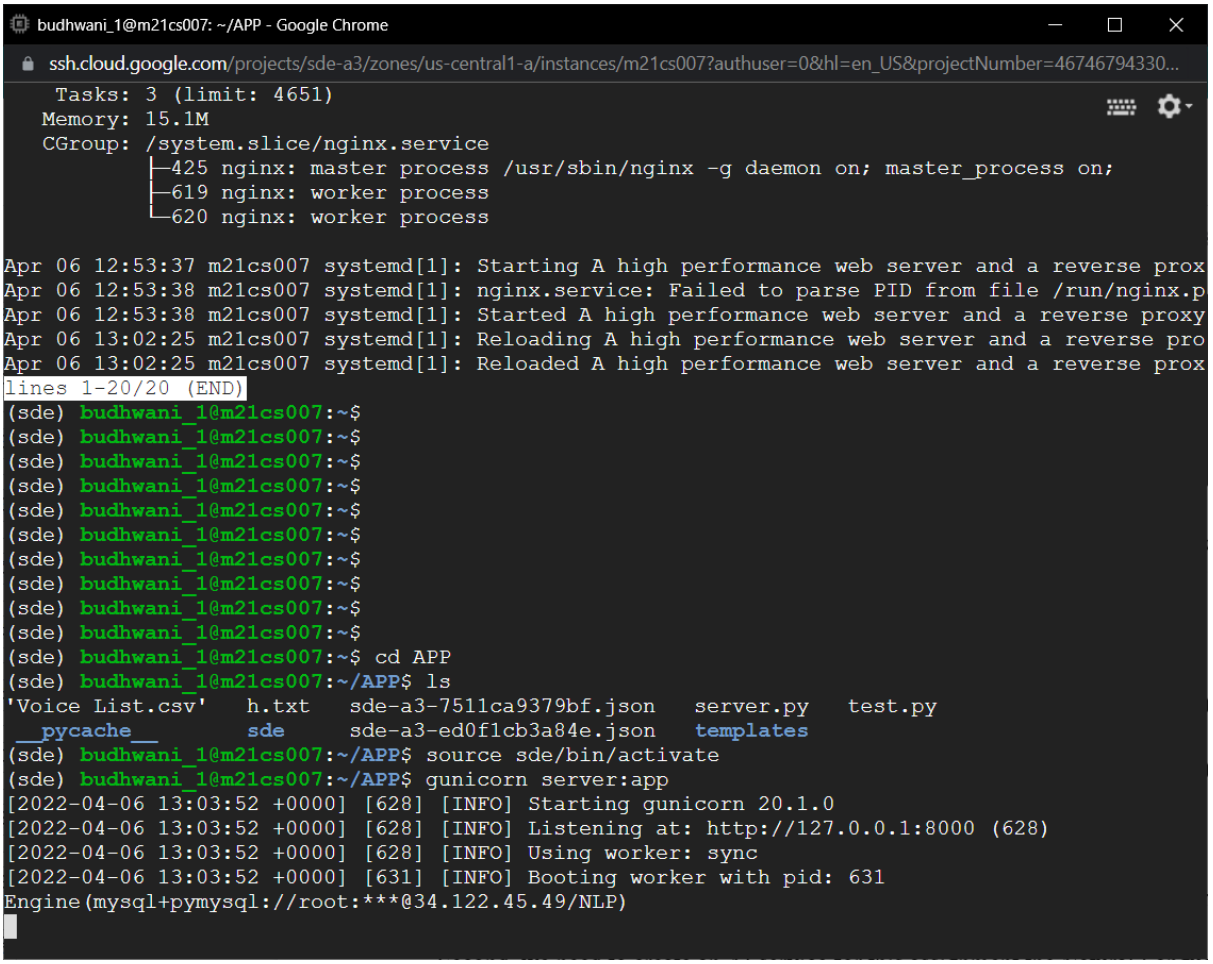
12. After making the changes in the config file we need to make changes in the index.html to create a webpage for our appropriate needs which can be found at **cd /var/www/html/index.html.**



13. After making the changes in the nginx server we can see our hosted site after again starting our nginx service using **sudo service nginx reload && sudo service nginx status**.



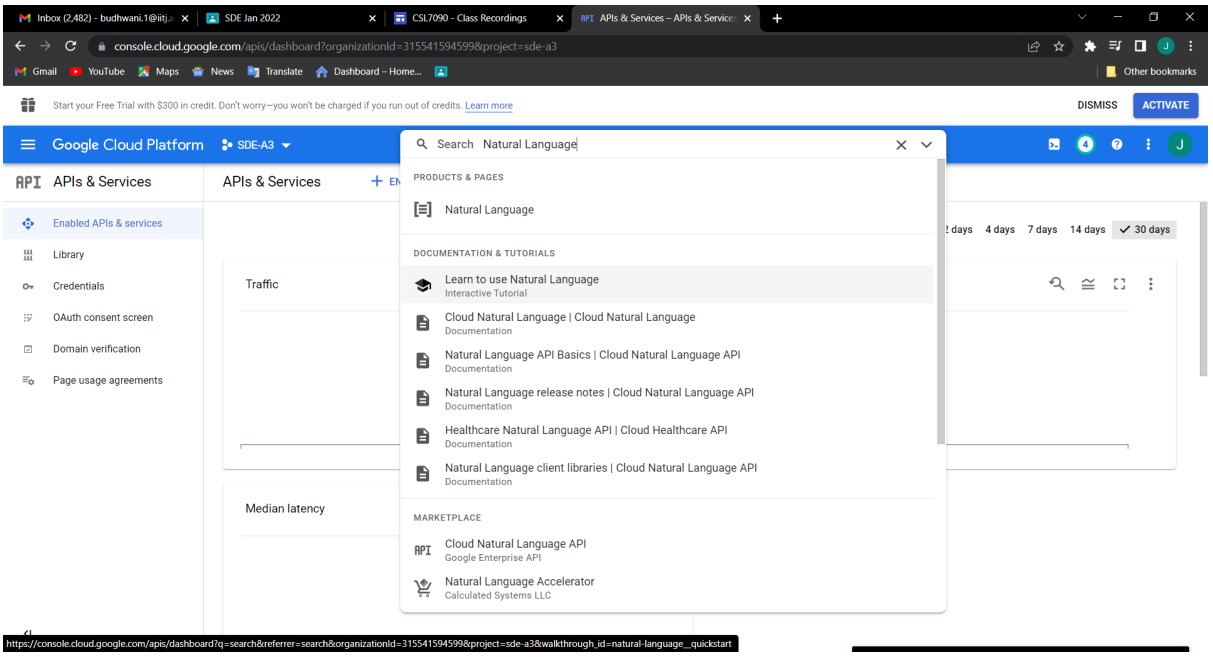
14. Now to set up our flask project and environment use commands.
    - First, create a virtual environment so that all of our libraries can be in a single place using the command - **python3 -m venv sde** (sde is name of our virtual environment).
    - Now enable the virtual environment - **source sde/bin/activate**
    - Now install Flask - **pip3 install flask**
    - Now install gunicorn - **pip3 install gunicorn**
    - Now create a folder named **templates** for our HTML templates and create **server.py** which contains the flask code.
    - As we know the flask run locally so we use gunicorn to host this local flask on our compute instance.
    - Run gunicorn using command **gunicorn server:app**



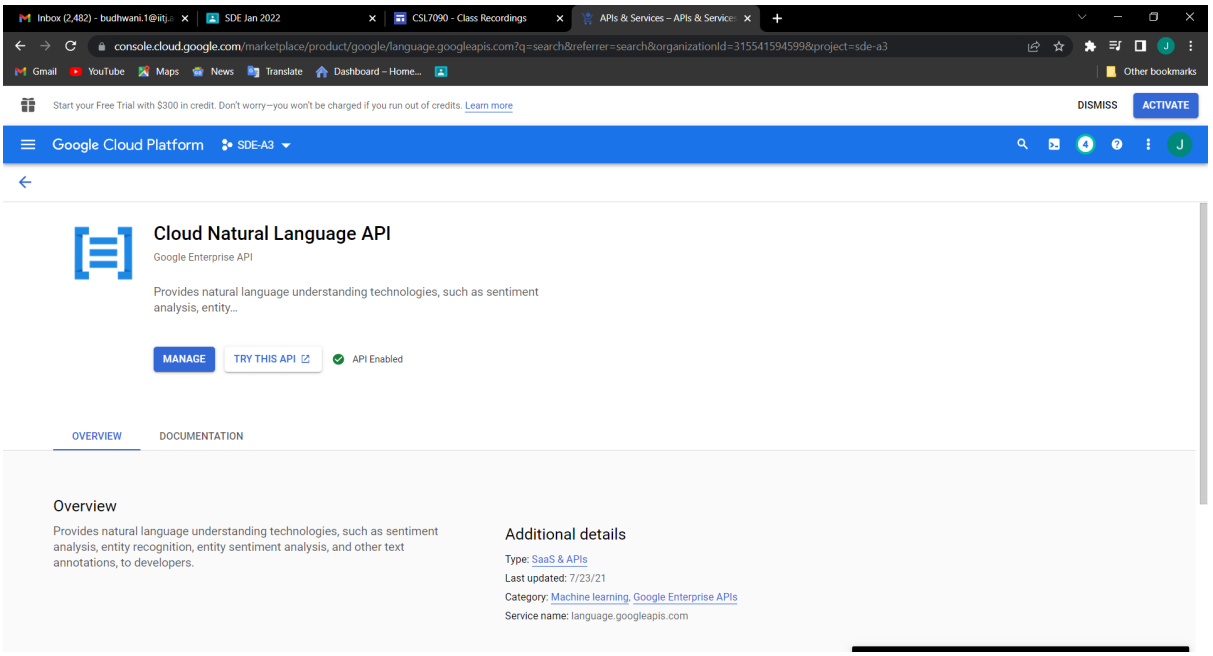
15. After setting up the gunicorn and flask add the functionalities of AI services and Cloud SQL.

Second, we need to create an AI service for this assignment the Natural Language AI service was used for Sentiment Analysis.

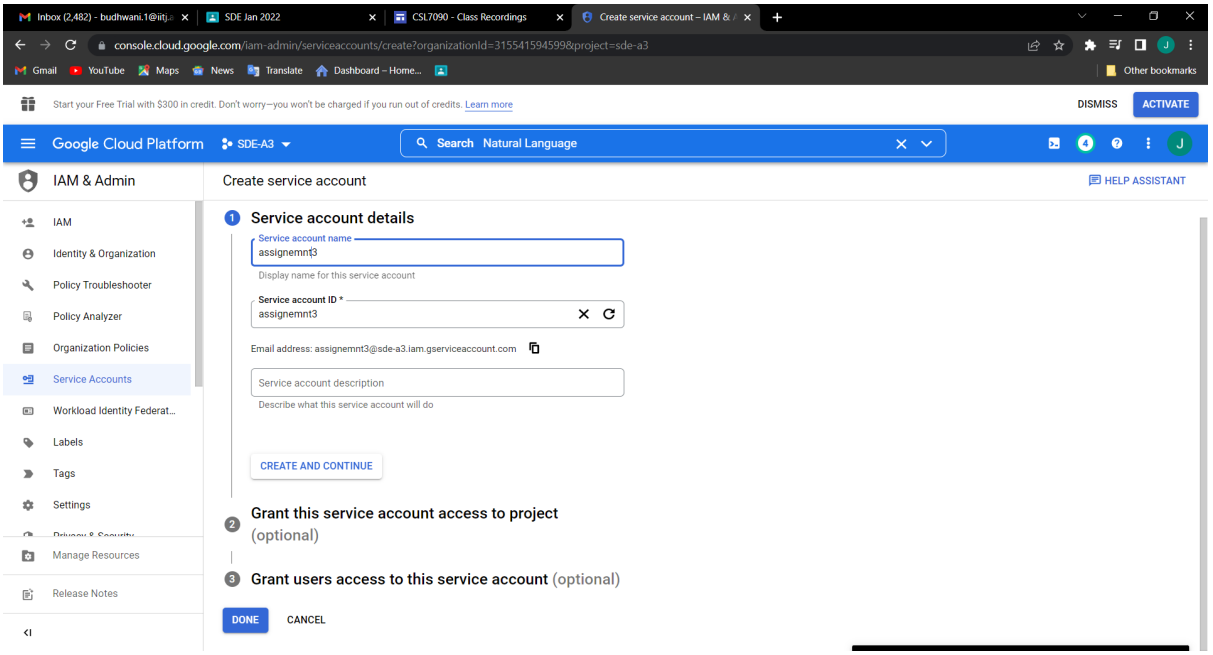
1. Go to **Cloud API and Services**, then search for **Natural Language API.**



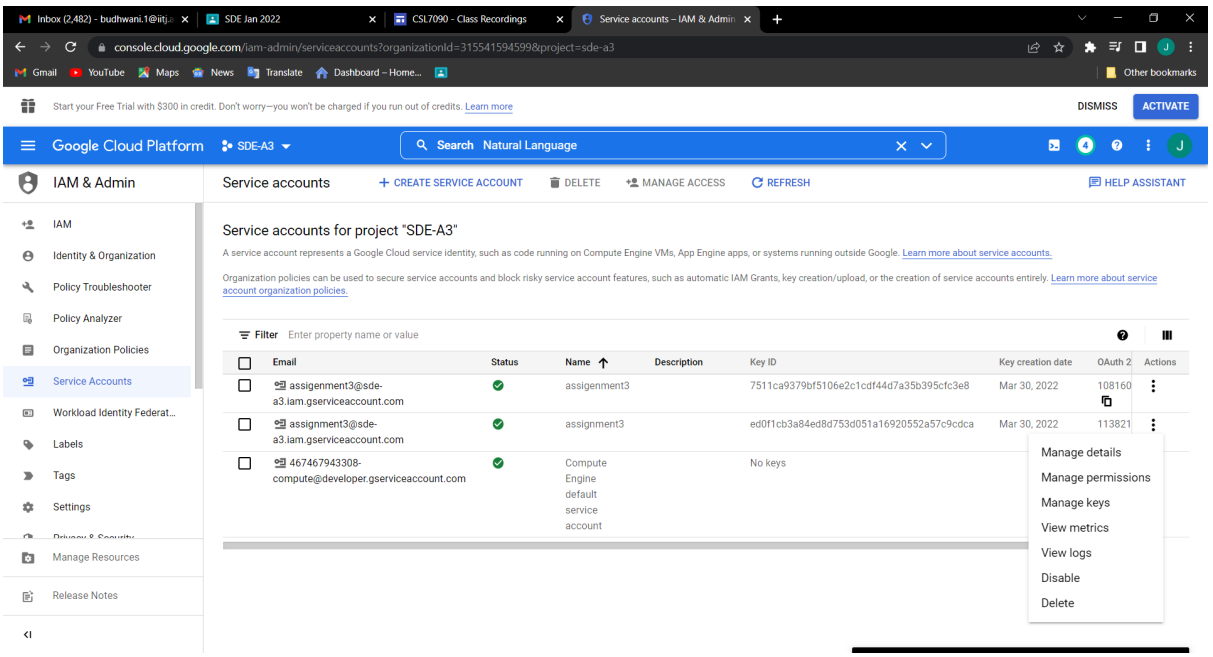
2. **Enable** the Cloud Natural Language API.



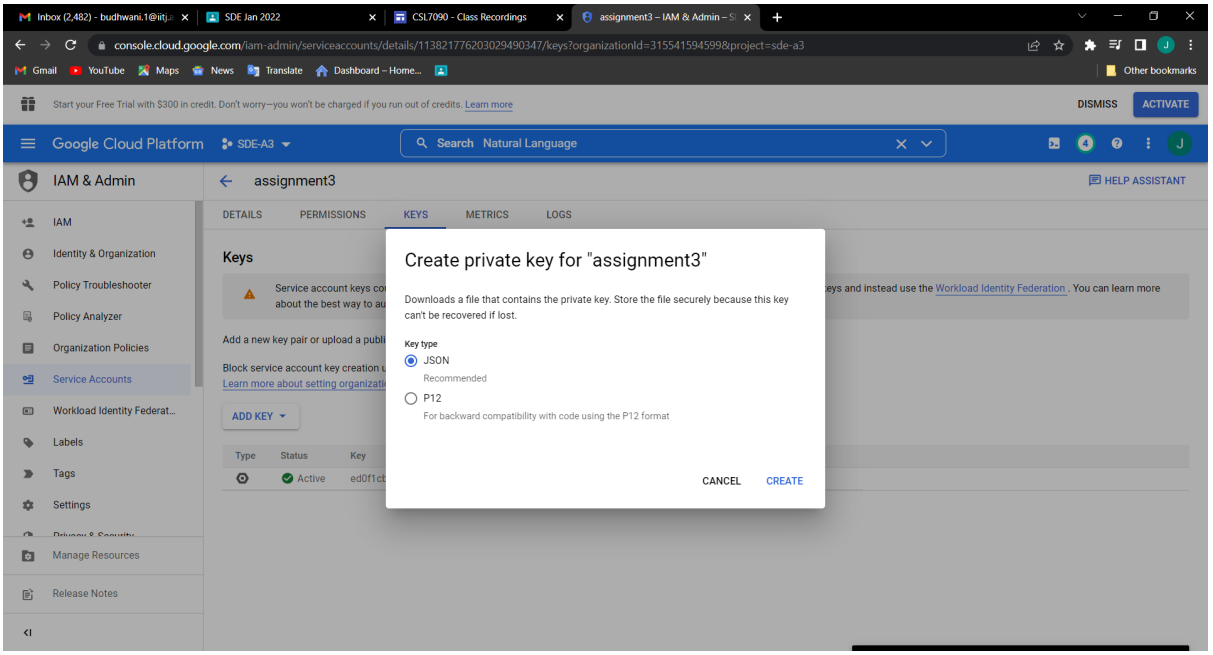
3. Next, create a **Service Account** to Access the API in **IAM and Admin** panel by adding appropriate details



4. After creating the service account **create an account key in JSON** format under the Manages keys option.



5. Then under the add key option, there is an option of creating new key use that to create a new key in JSON format.

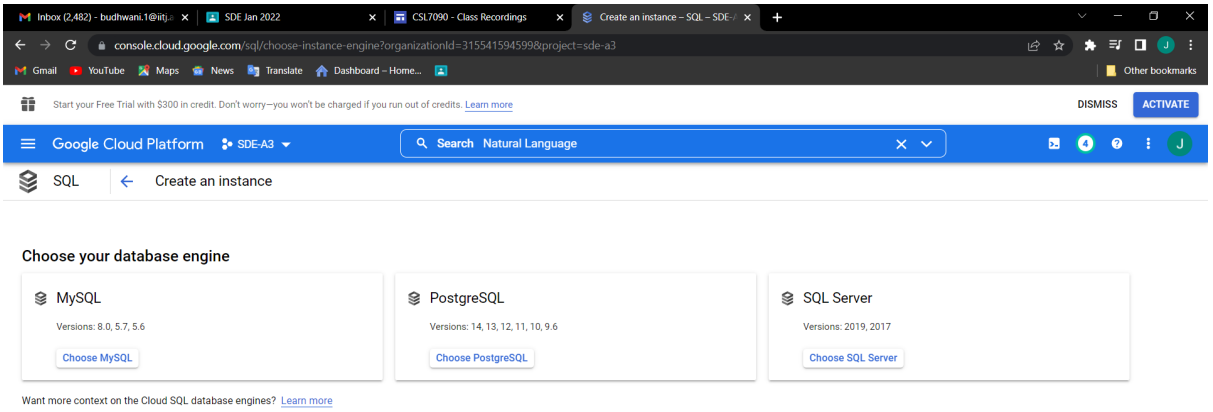


6. Then save this key and upload the same key to the previously created instance.

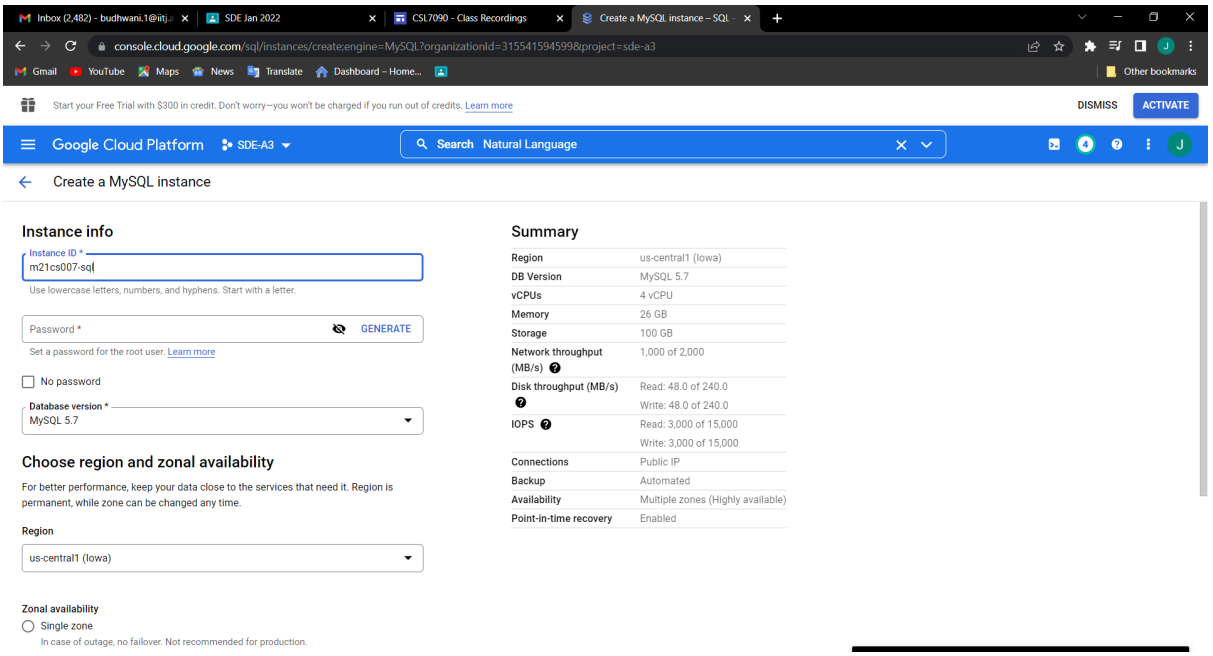
Now we also have a Cloud AI service with Compute Instance Finally we need Cloud storage to store the data.

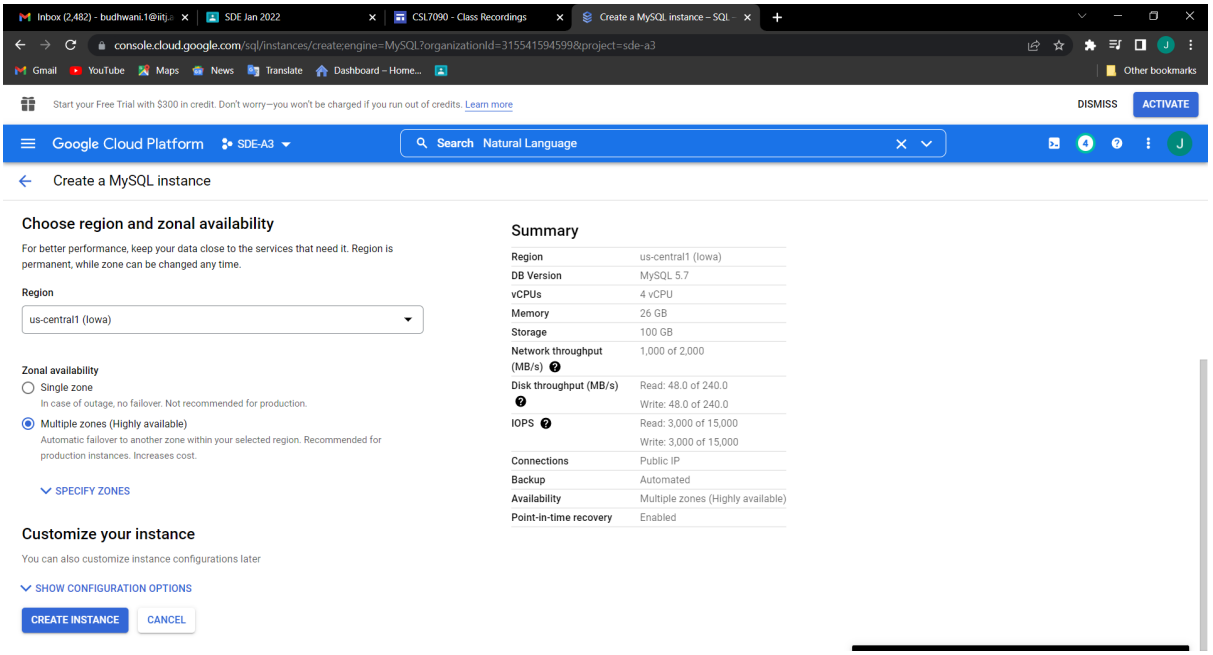
For Cloud Storage, Cloud SQL was used, to create a Cloud SQL instance use the following steps.

1. Go to **SQL in Cloud Left side Pane**, Click on **Create Instance** then Select the Type of Instance you need line **Mysql**, Postgres, and SQL server. Mysql was chosen for this assignment.

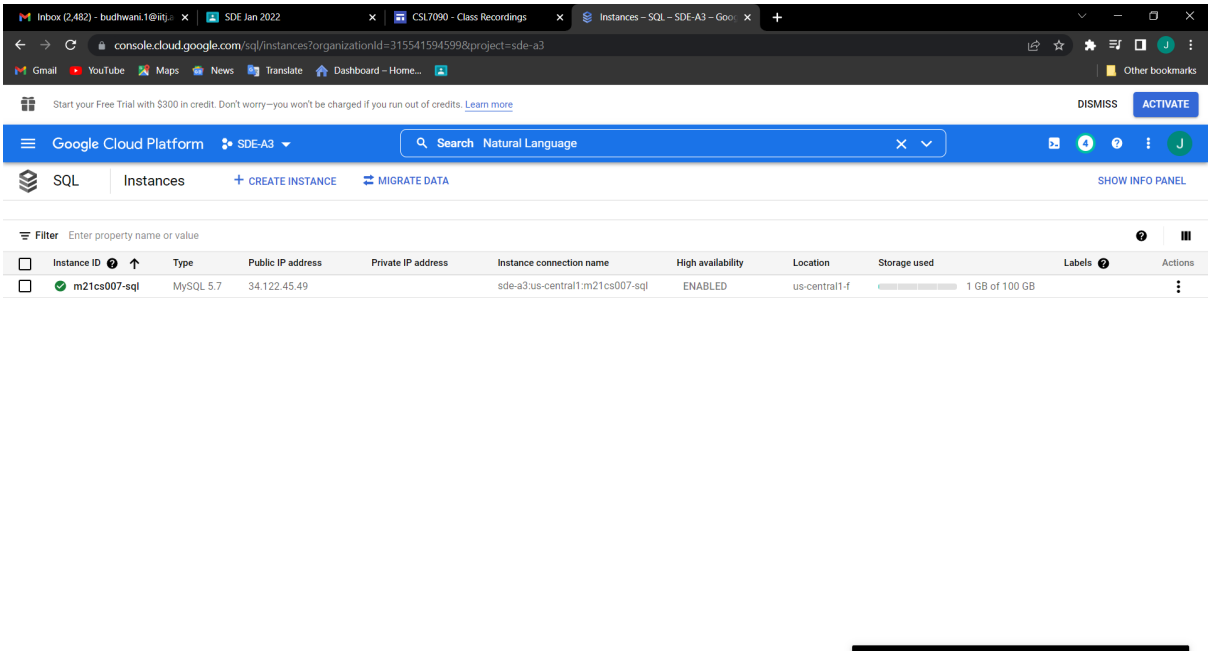


2. Then after selecting Mysql Enter the instance name and appropriate details like **size, CPUs, Password, etc. Click on create to create the instance will take around 5-6 minutes**.

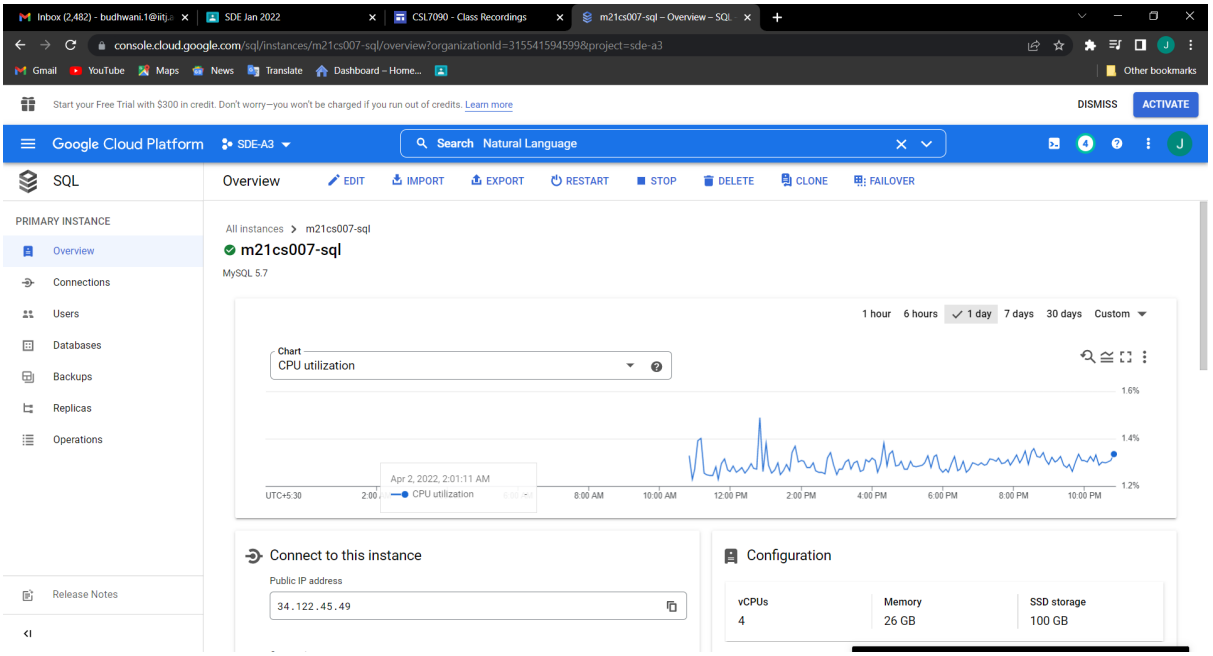




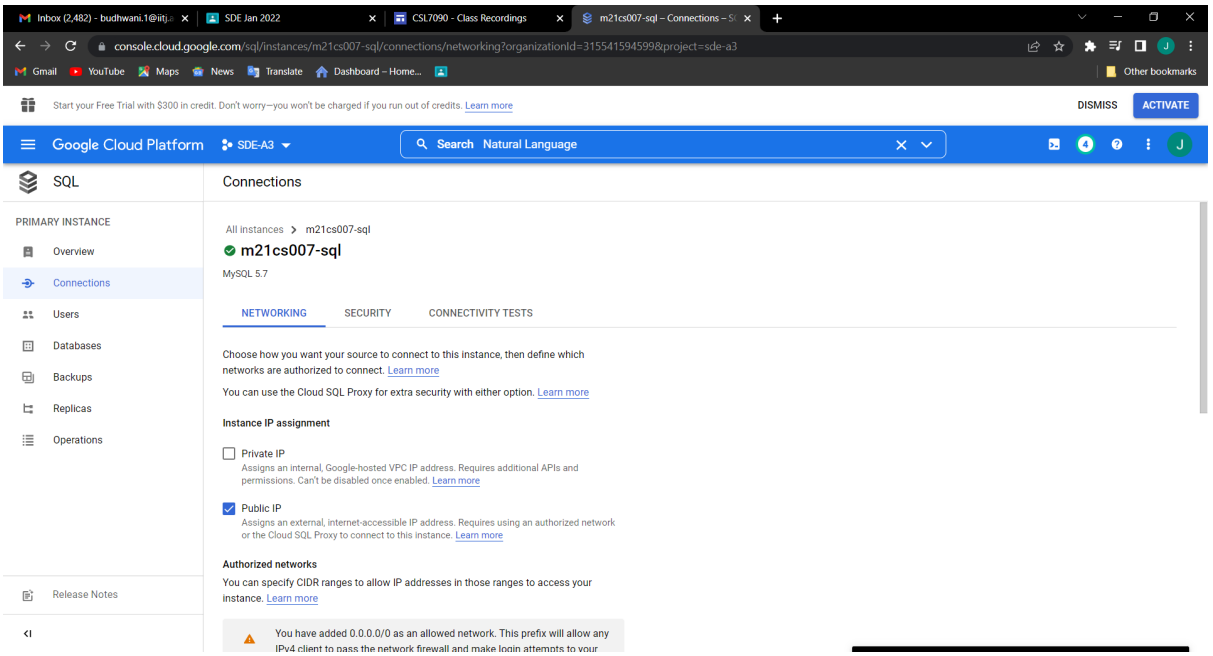
3. Then after the instance is created it looks like this.



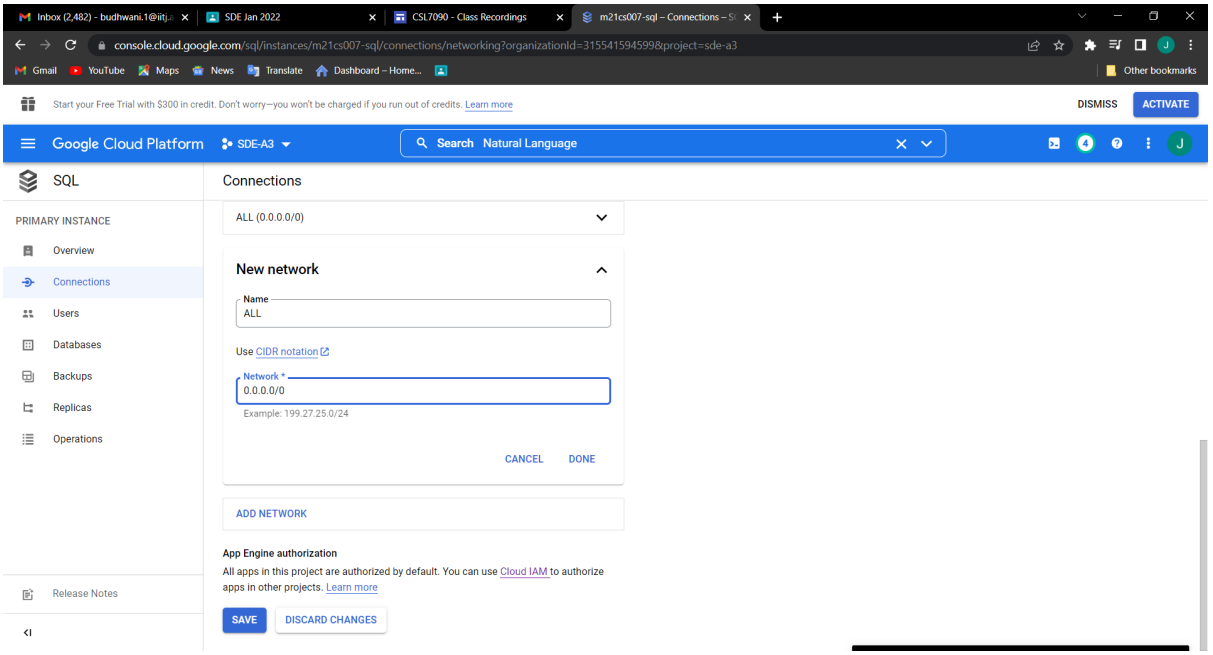
4. Then click on the instance and see all activity.



5. After this, we need to give access to users for accessing the Databases for this go-to **connection -> networking.**



6. Then under add network add a network like **0.0.0.0/0 with the name ALL** to give access to all public networks.



7. Then Click on Save this will take some time to restart the instance.

Finally, we have created all three services. Now combining all three into an application.
For this application, we are using the computer engine to host a flask application that takes user input then perform the sentiment analysis, entity analysis, and classification analysis, and then provide the result and store that result in the Cloud SQL for future query and also it shows all the previous done experiments in the application.
For this we need to perform some steps:

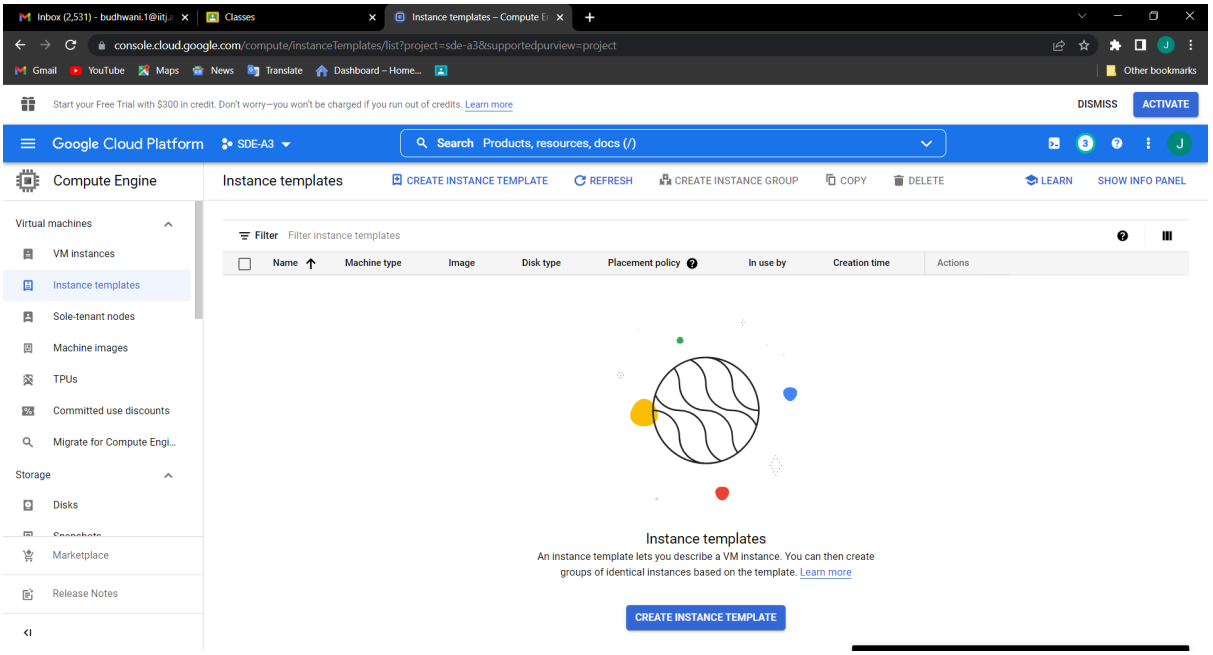
1. We need to create a file **server.py** that contains the required code of flask and also we need to create the folder templates that contain the HTML files which are used to perform operations and show results.
2. In the server.py file, we need to give access to our Cloud Natural Language API which can be done using the os library provided by python
   os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = "<key_file.json>"
   and also we need to access the cloud Mysql which can be done using
   engine = sqlalchemy.create_engine('mysql+pymysql://root:root@34.122.45.49/NLP')
   where **34.122.45.49** is the public IP of Mysql and **NLP** is the name of the database and the User name and Password are both root respectively.
3. For running this file without errors some libraries need to be installed using pip or pip3 command which are mysql.connector, sqlalchemy, pymysql, google.cloud, google-cloud-language, etc.
4. Finally, after creating these fles we need to run the flask project using the gunicorn so that our hosting on the nginx server can be done without any problem.
   - Use command gunicorn server:app
5. Finally, go to the link http://<IP_of_instance> like to see the Main Home page.
6. Go to http://<IP_of_instance>/apis like to see sub home page.
7. Go to http://<IP_of_instance>/apis/show like to see the records present in Mysql Database.
8. And finally fo to http://<IP_of_instance>/apis/data like to perform new sentiment analysis.

For verification of the application, you can all see that the records are entered into the database.

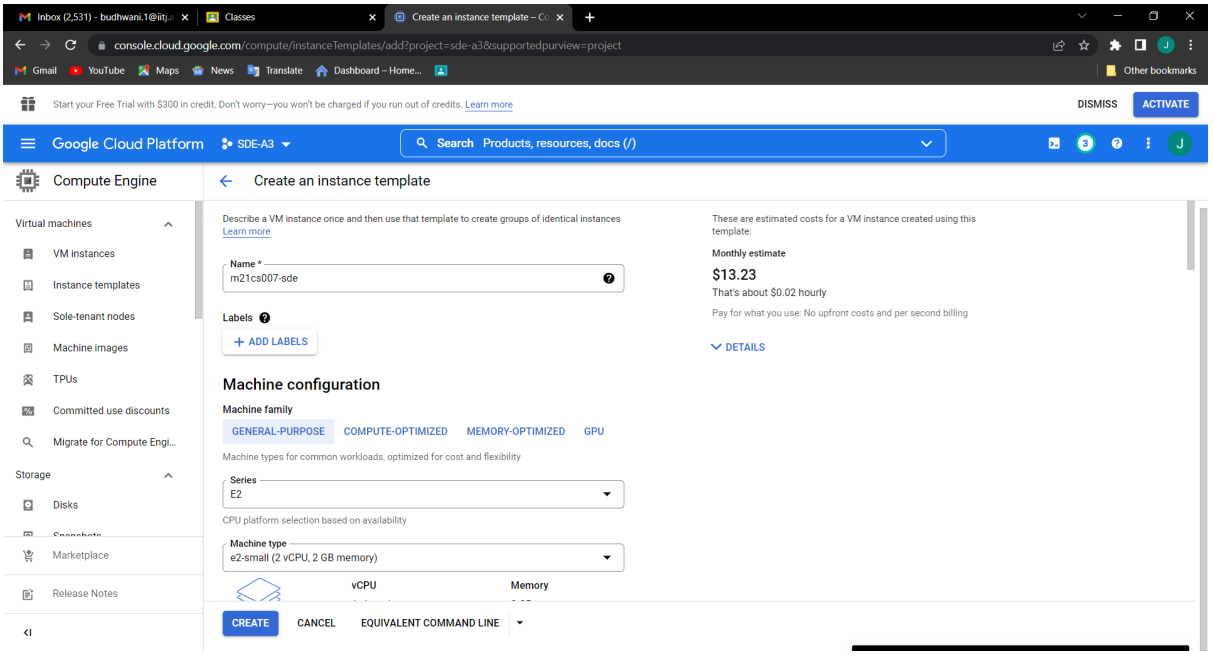
Part B: Creating a Load Balancer.

Step 1: Create an Instance template.

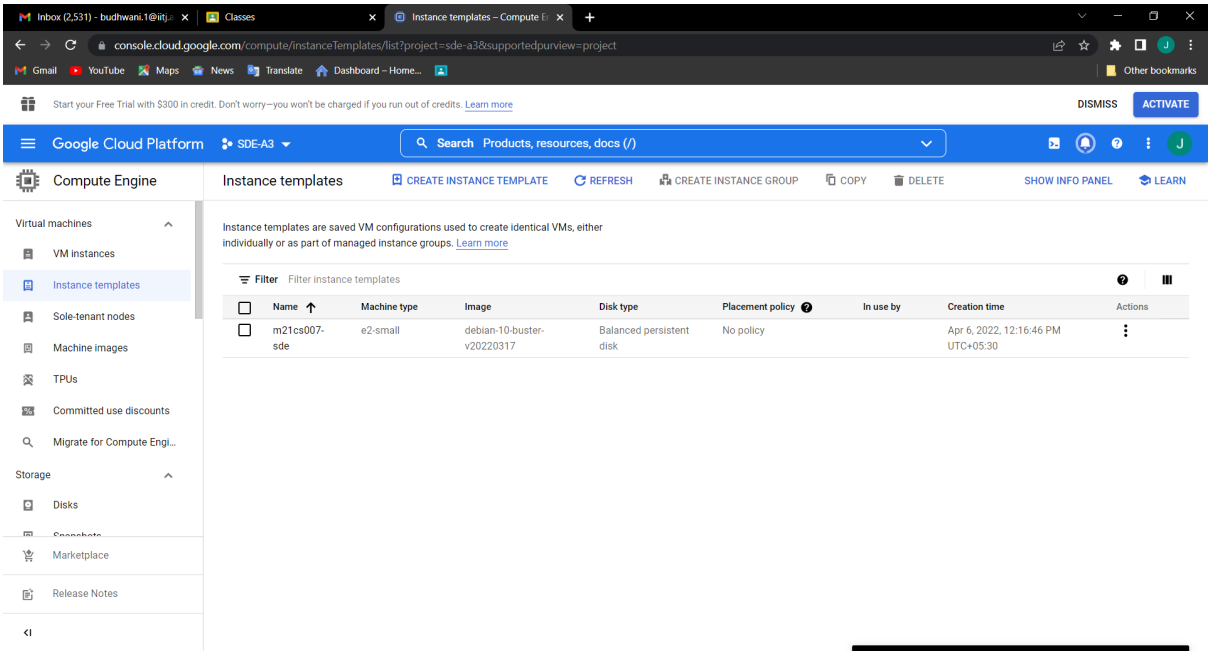
- First, we need to go to **the engine instance template**, and then we need to **create the instance template**.



- After this **enter the appropriate name** for the template and **click on create the template**.

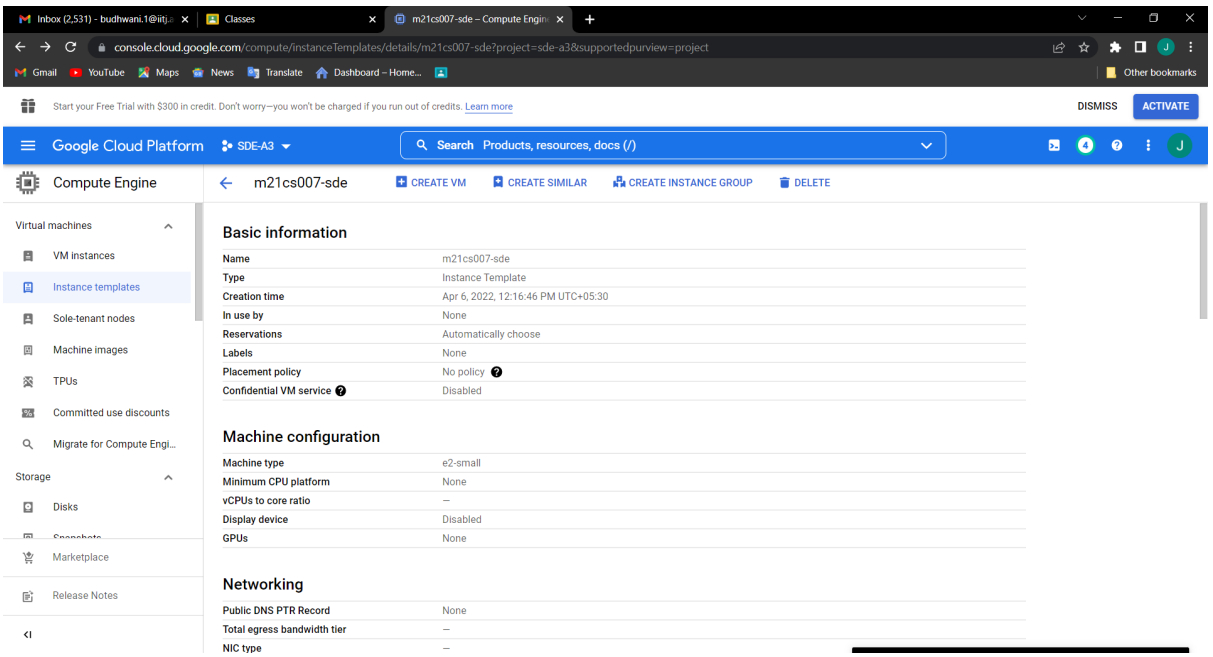


- After creating the template it will look as below.

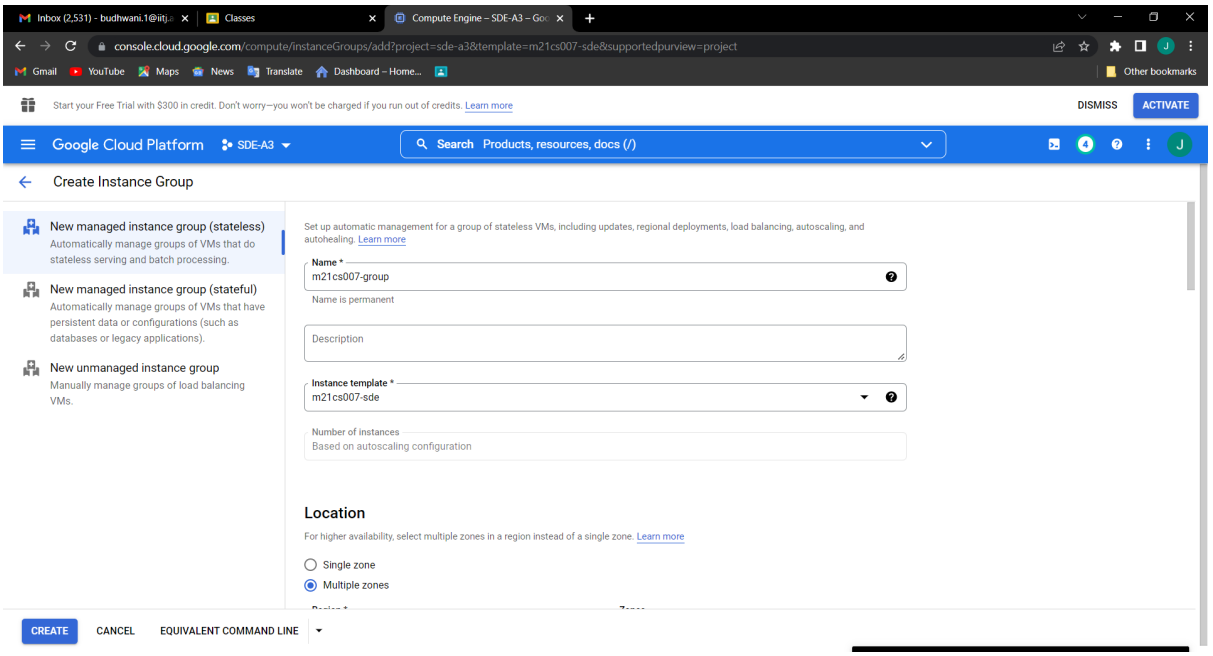


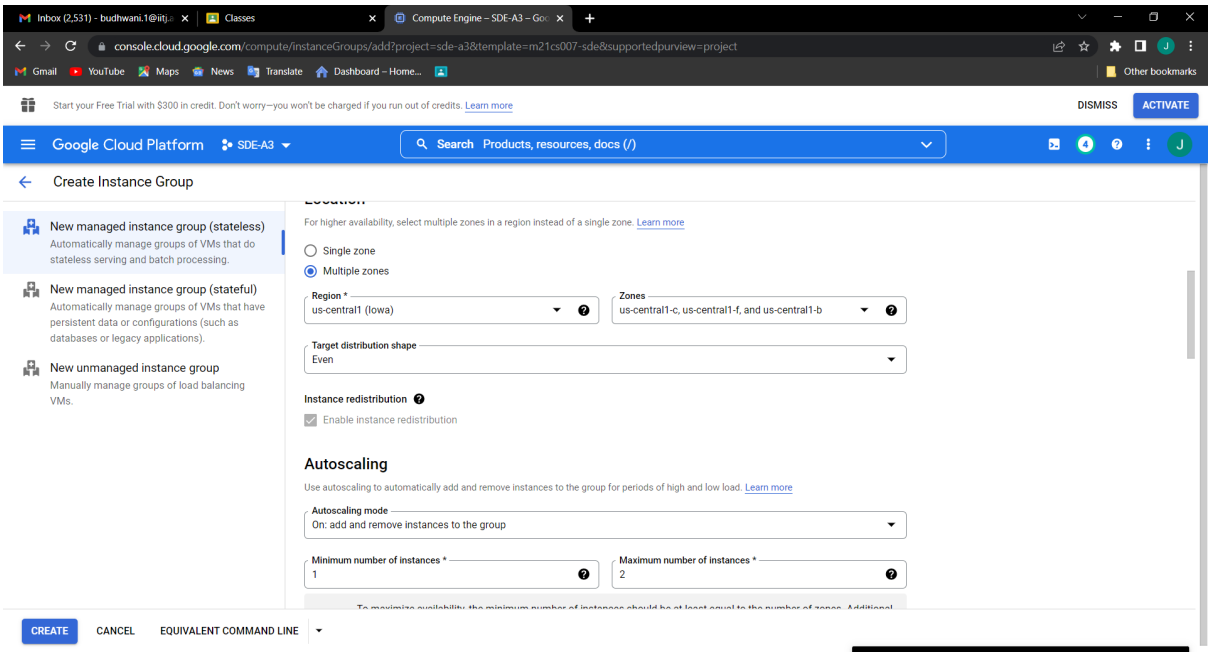
Step 2: Create an Instance Group

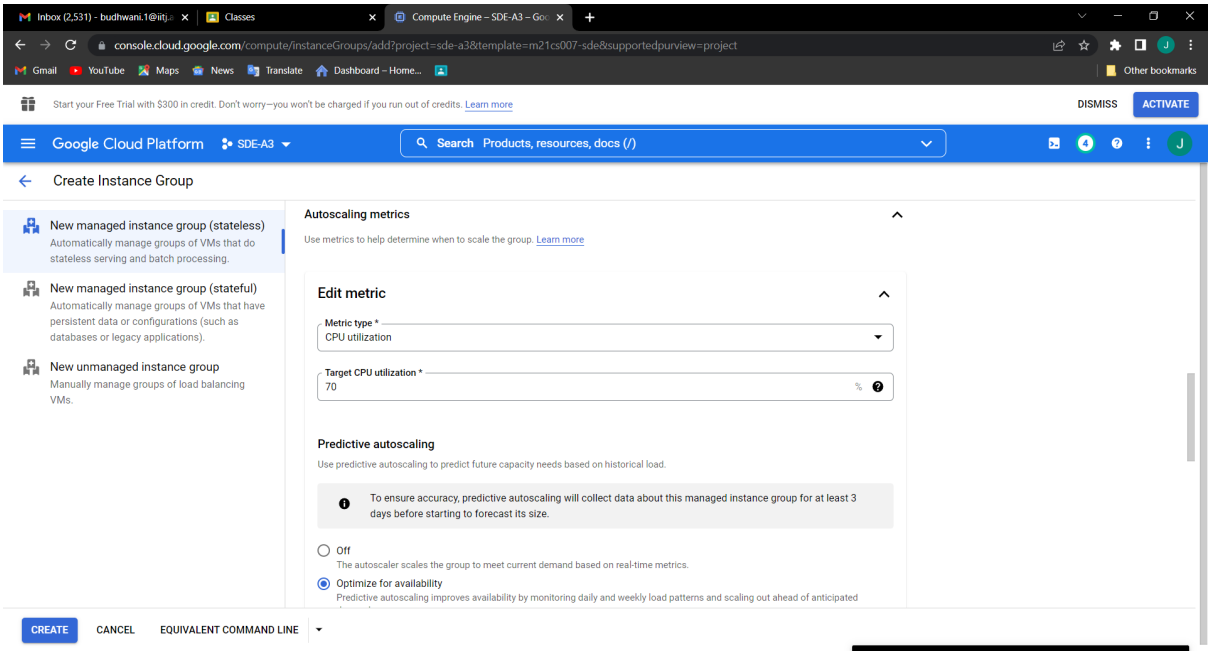
- Go to Instance Template created in the previous step in compute engine and click on **create instance group.**



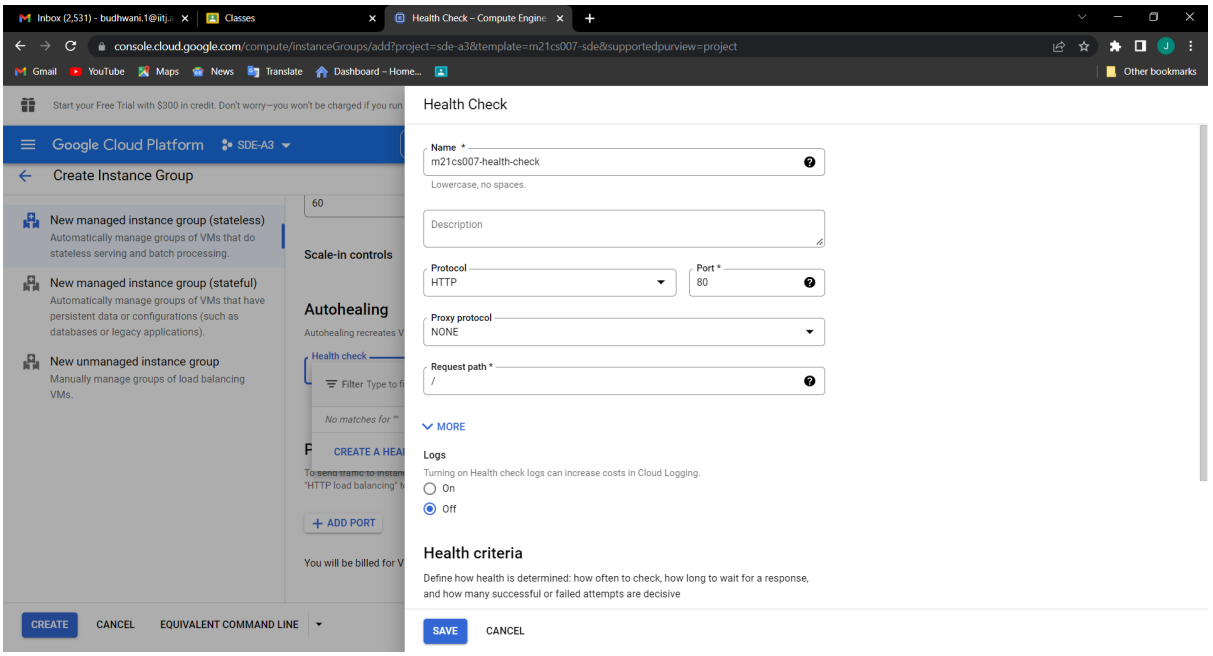
- Provide the details like **Group name, select Multiple zones, number of instances** managed should be 2 under autoscaling, under autoscaling metric select **70% CPU utilization** for using another instance.

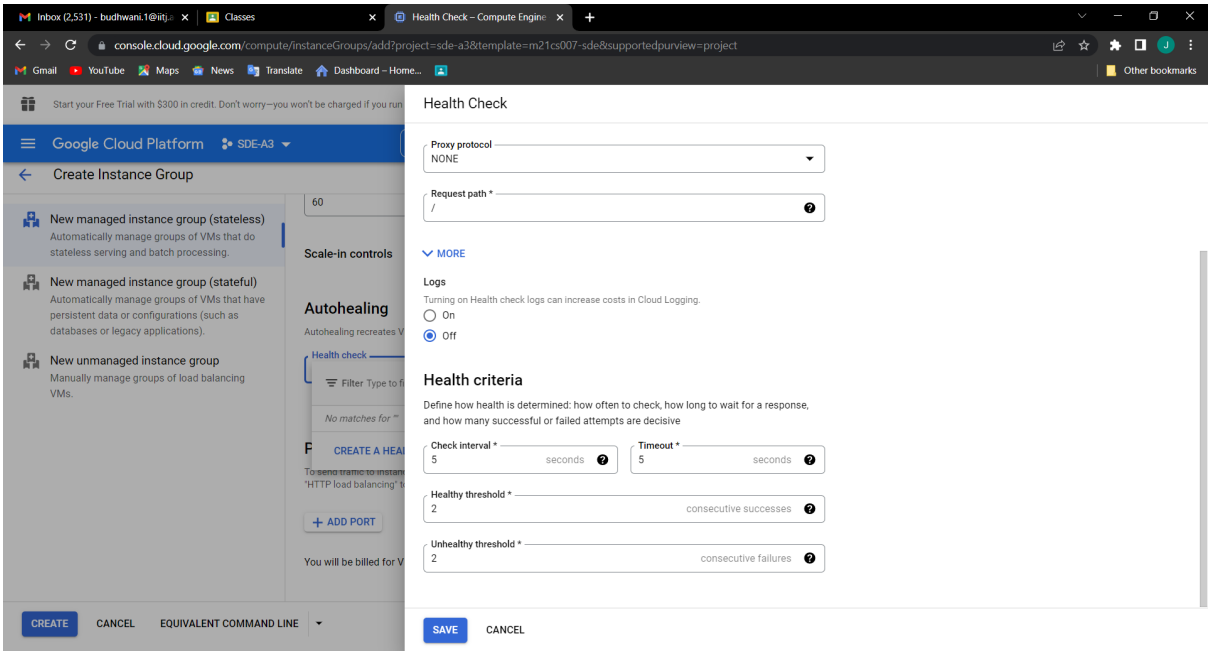




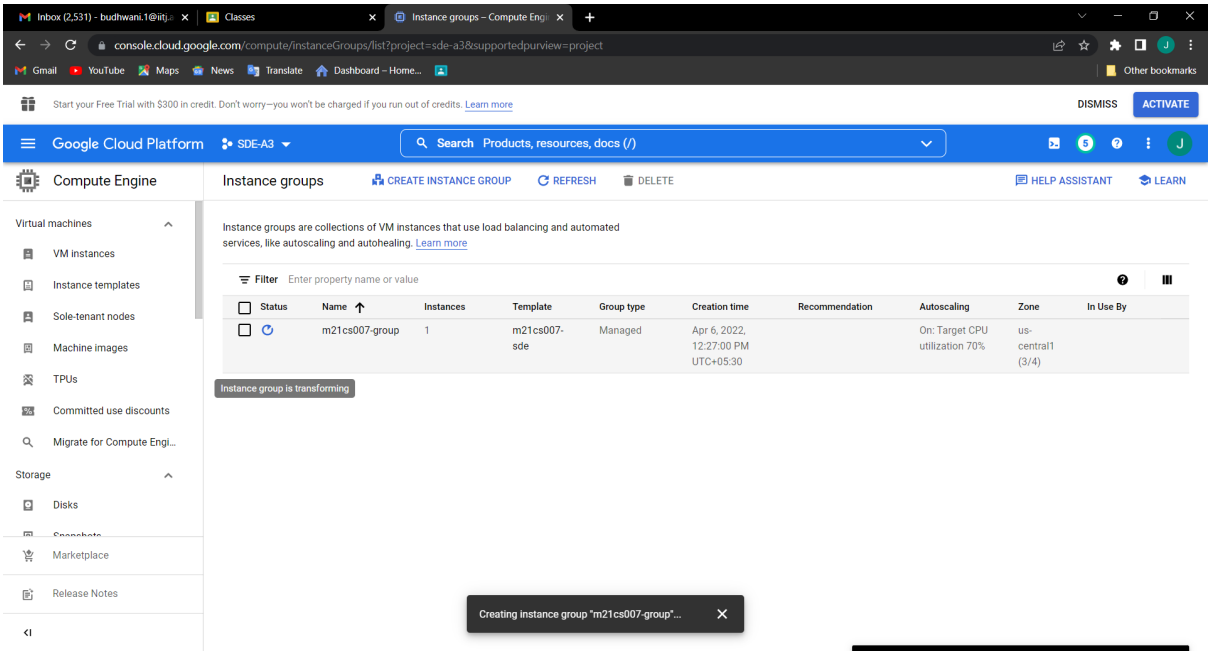


- After setting the above there is one more thing that is most important which is creating a **health check** to check whether the instances are healthy and if not after how much time they need to be created again or deleted.





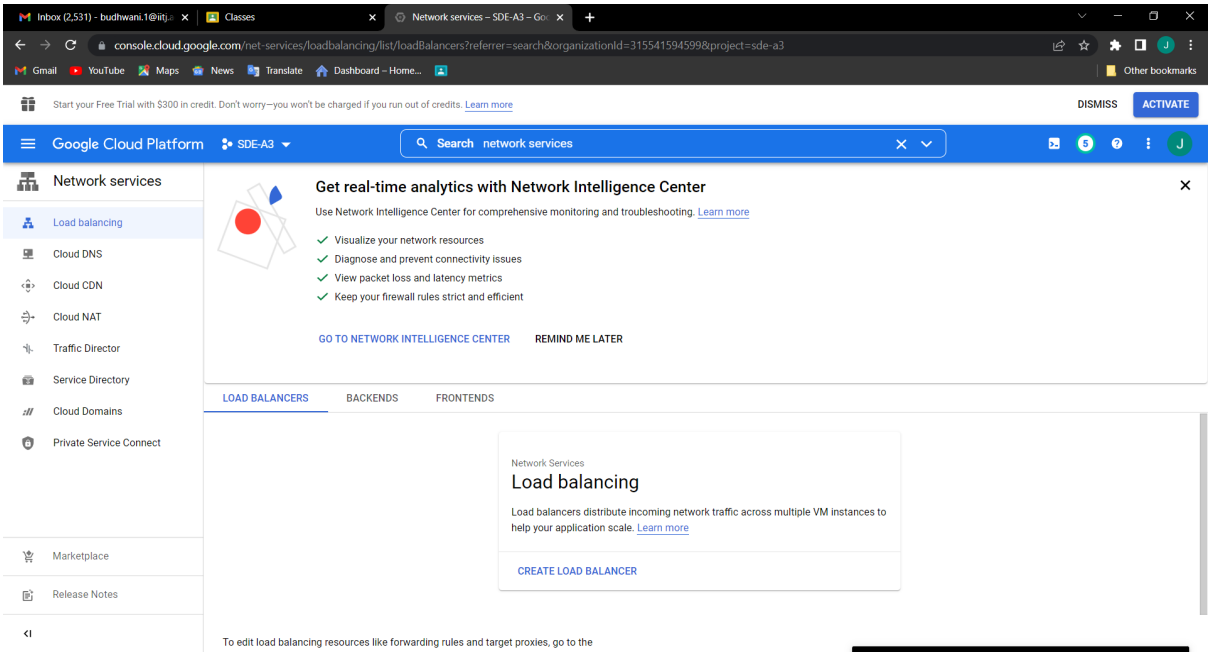
- After this, we need to create this instance group and after creation, it looks like the below.



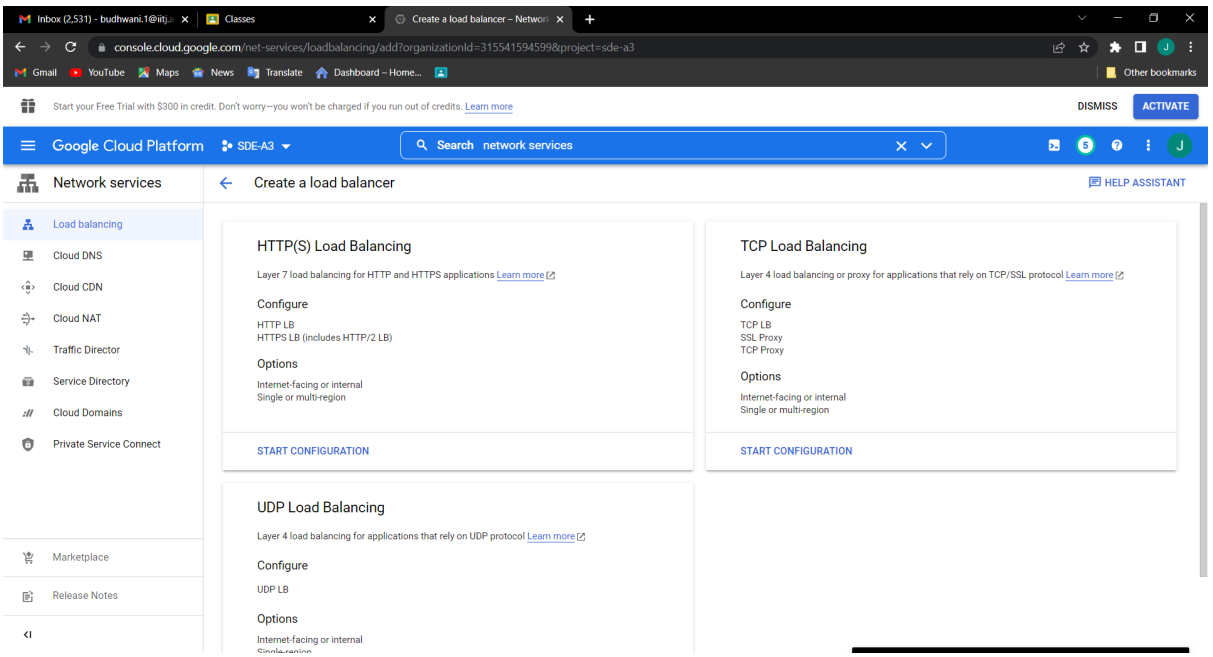
- It will take some time till the instance group is running.

Step 3: Creating Load Balancer.

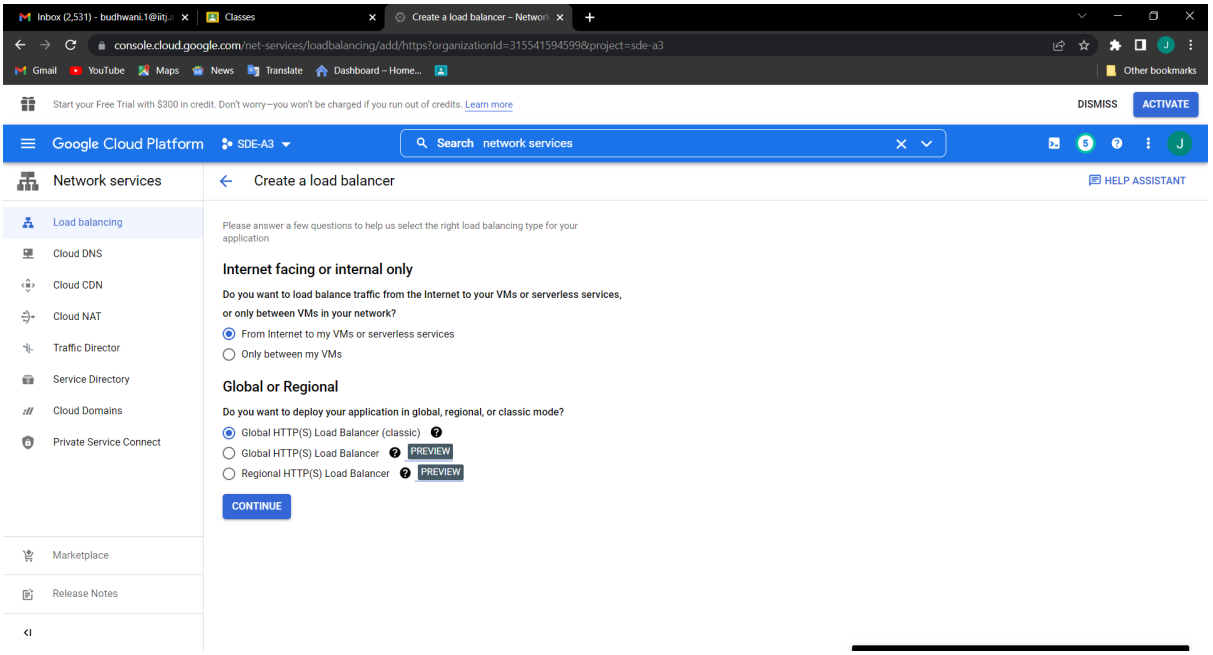
- Go to **Network services** to create a load balancer.



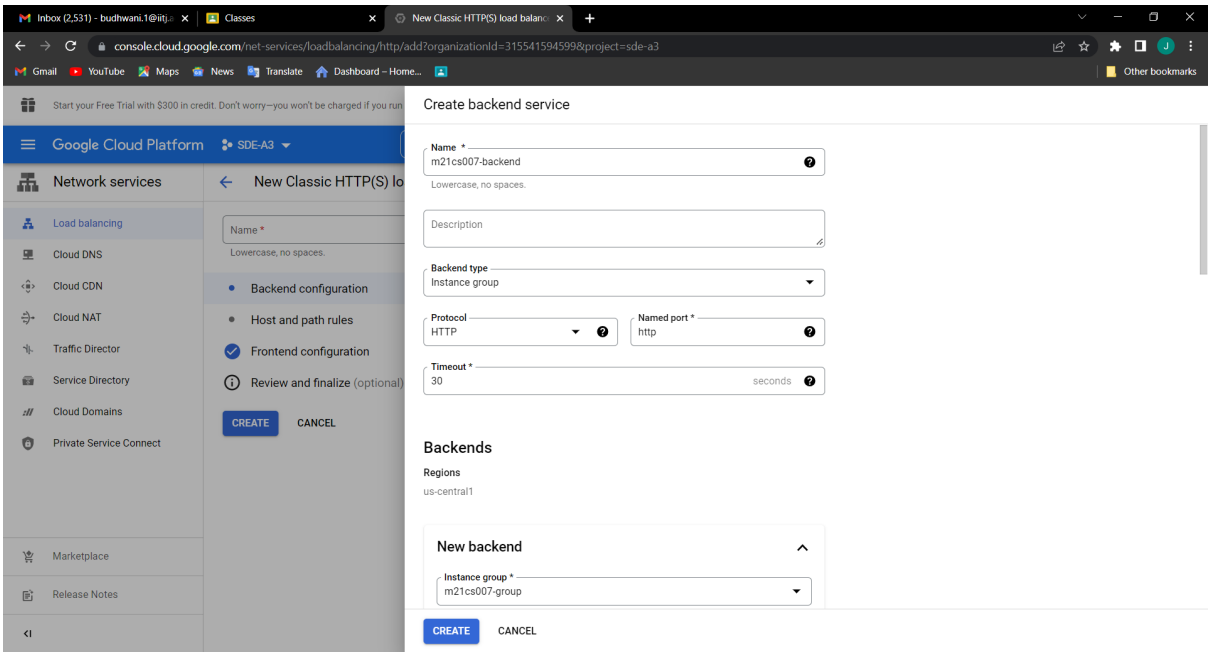
- Under create, the load balancer option selects **HTTP Load balancer.**

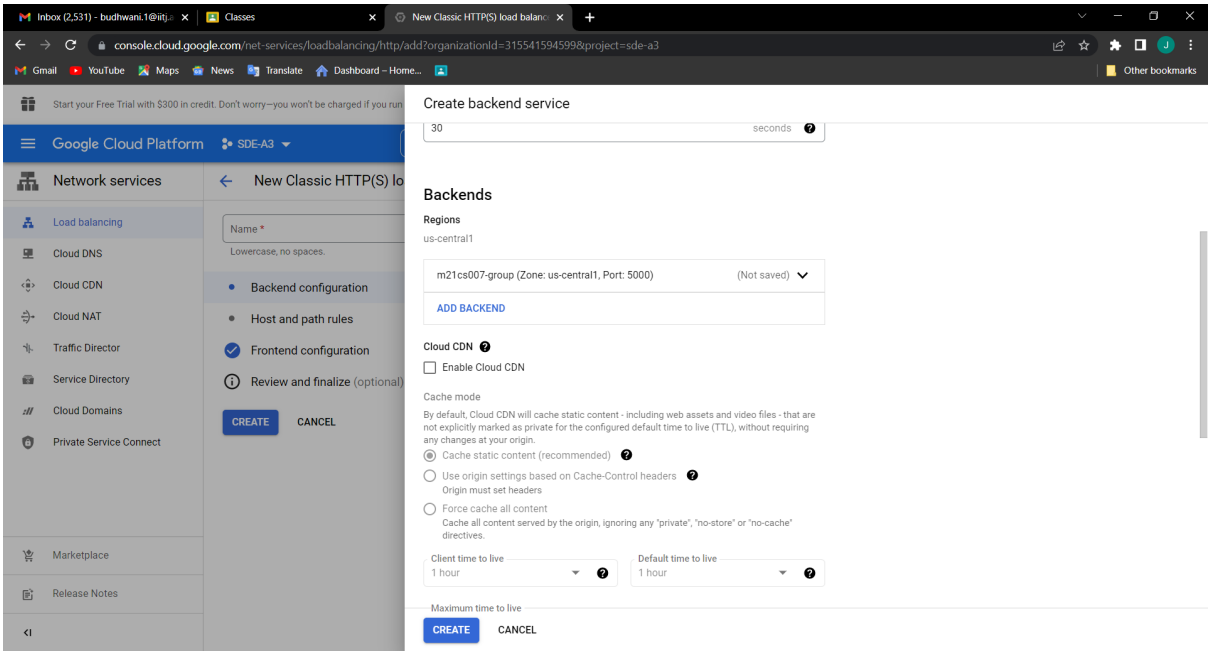


- Provide the below options.

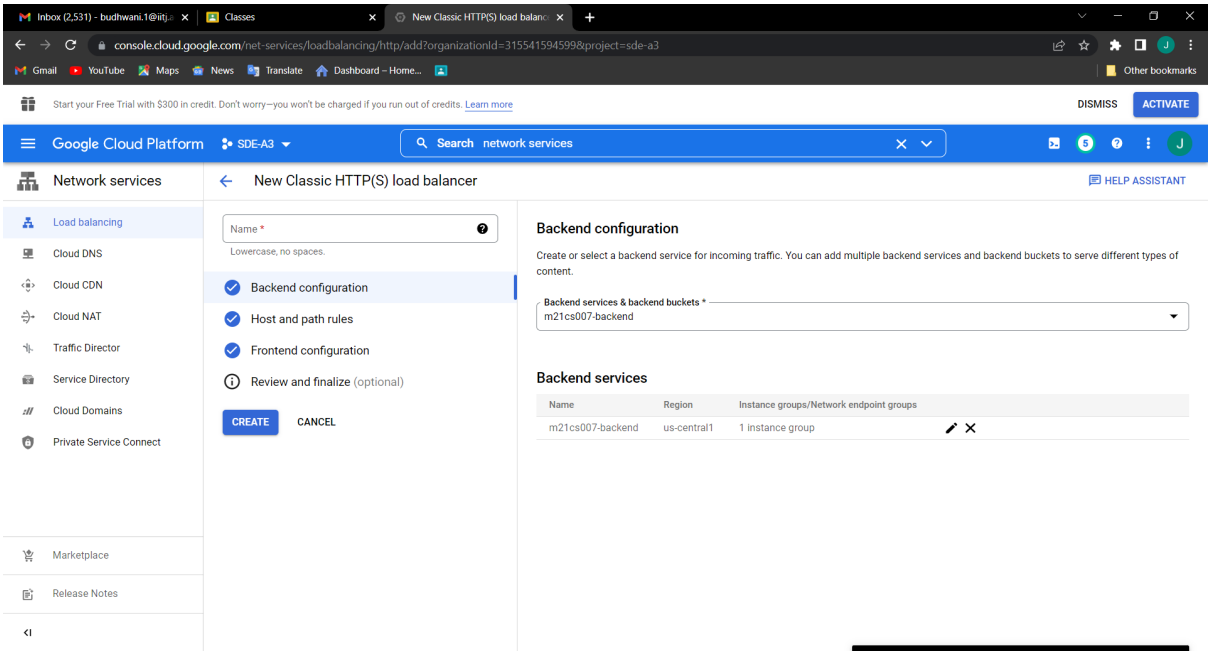


- Now we need to set the **backend configuration.**

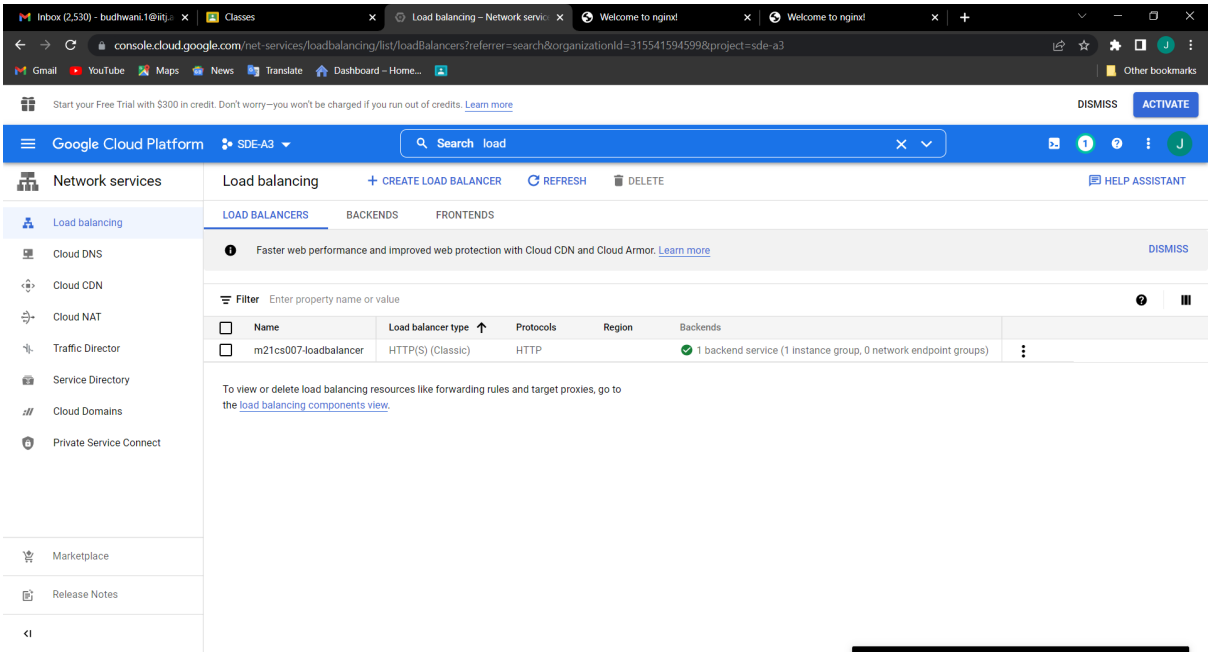


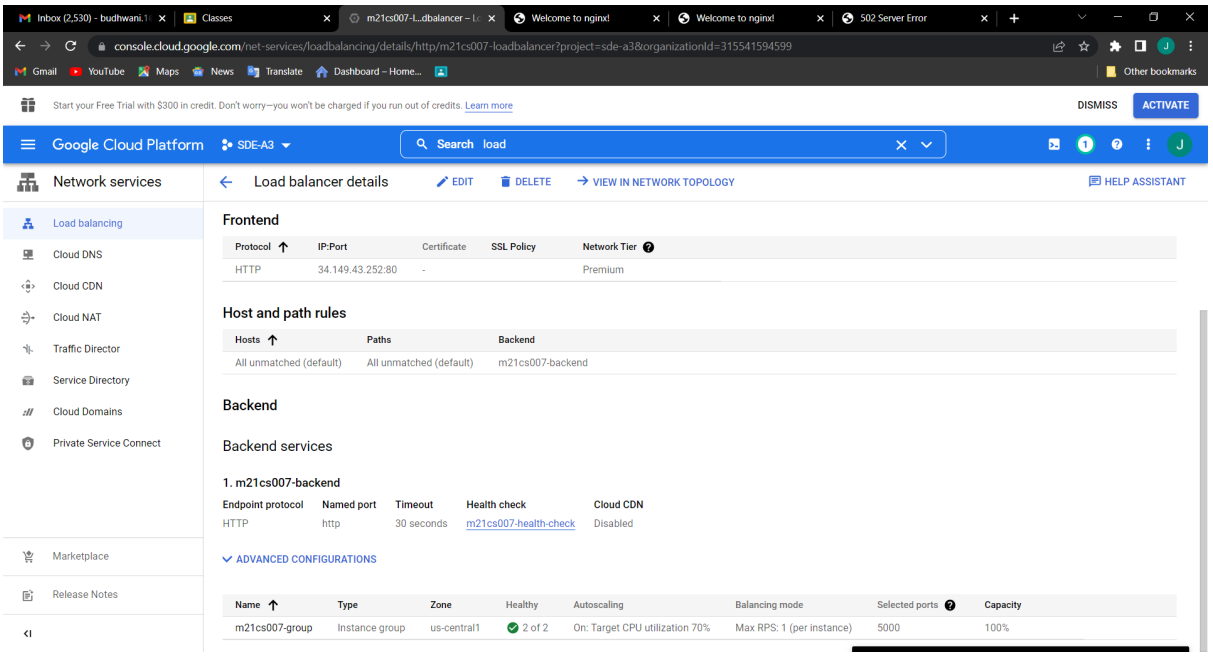


- After creating the backend service it looks like the below.

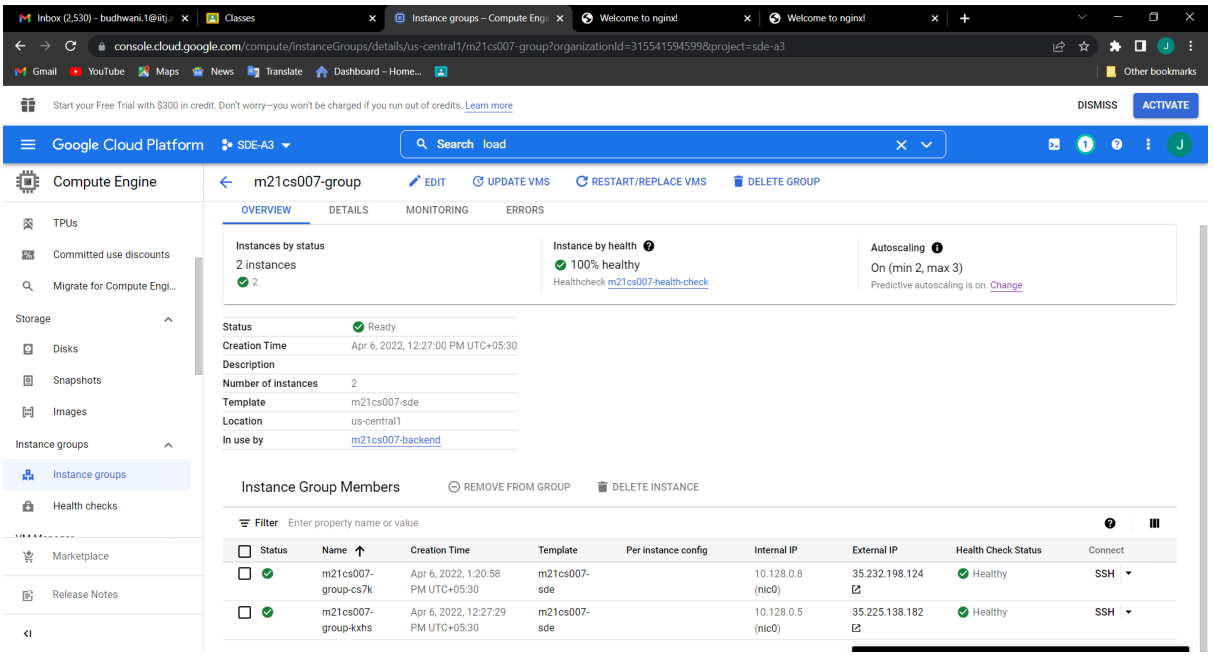


- Finally, select create to create a load balancer. It will take some time to create.



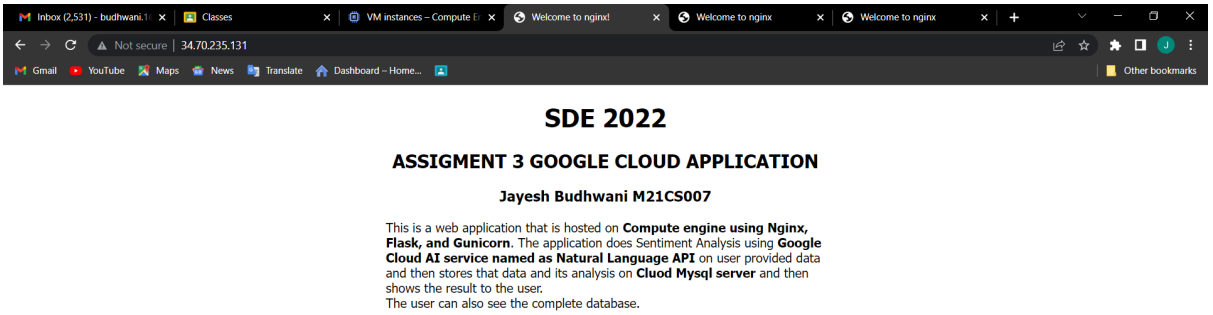


- After Creating the load balancer we can see that there would be two new instances created in the compute engine instances as shown below. The First one is the base application.

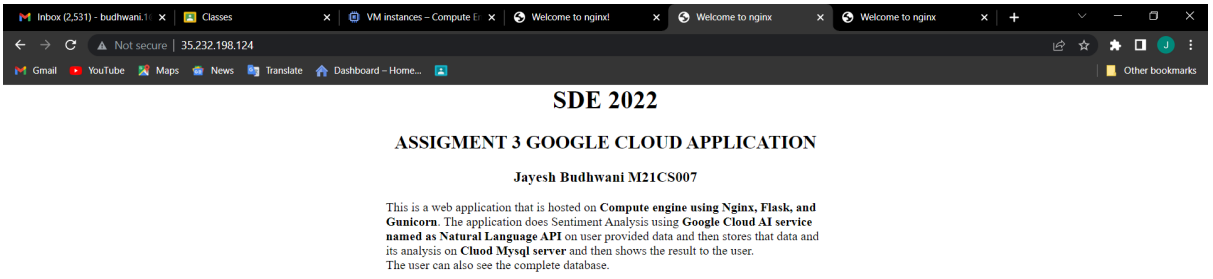


- Now to show how the application would run using a load balancer is really simple just use the load balancer Ip address to access either of the two applications. We have differentiated the **two applications using different text styles**.

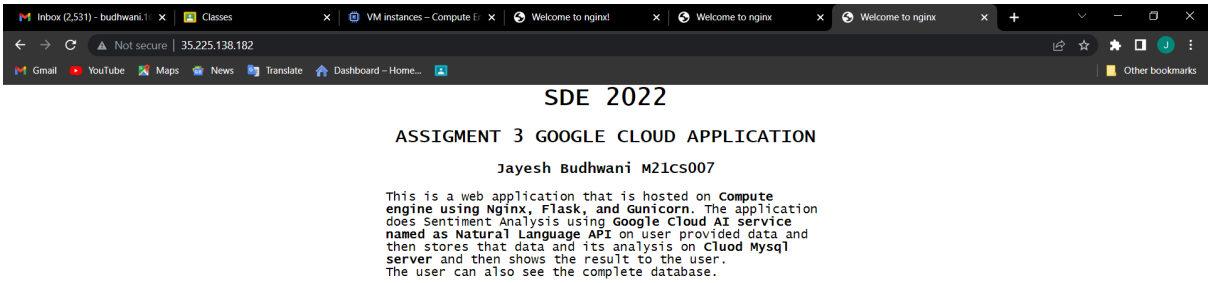
- **Original**



- **First Load Balancer Instance**



- **Second Load Balancer Instance**



As we can see all the three instances are working perfectly with different text styles to differentiate between them the first one is not a part of the load balancer the other two are a part of the load balancer.

YOU-TUBE Links:

1. For creating the services: https://youtu.be/QsRuHuQLt1c
2. For Implementing the application using Flask: https://youtu.be/m6yVVZJGIw8
3. Demo of Application: https://youtu.be/Np22lG_Cezg

References:

1. https://cloud.google.com/natural-language/docs/reference/libraries
2. https://cloud.google.com/sql/docs/mysql/admin-api/libraries