

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.

To create a VM instance, select one of the options:

- New VM instance
Create a single VM instance from scratch
- New VM instance from template
Create a single VM instance from an existing template
- New VM instance from machine image
Create a single VM instance from an existing machine image
- Marketplace
Deploy a ready-to-go solution onto a VM instance

Name * **instance-1**

Labels **+ ADD LABELS**

Region * **us-central1 (Iowa)** Zone * **us-central1-a**

Machine configuration

Machine family **GENERAL-PURPOSE**

Series **E2**

Machine type **e2-small (2 vCPU, 2 GB memory)**

vCPU: 1 shared core | Memory: 2 GB

Monthly estimate **\$13.23**

That's about \$0.02 hourly
Pay for what you use: No upfront costs and per second billing

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

To create a VM instance, select one of the options:

- New VM instance
Create a single VM instance from scratch
- New VM instance from template
Create a single VM instance from an existing template
- New VM instance from machine image
Create a single VM instance from an existing machine image
- Marketplace
Deploy a ready-to-go solution onto a VM instance

Service accounts **Service account**
Compute Engine default service account

Access scopes **Allow default access**

Firewall **Allow HTTP traffic, Allow HTTPS traffic**

Monthly estimate **\$13.23**

That's about \$0.02 hourly
Pay for what you use: No upfront costs and per second billing

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

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

The screenshot shows the Google Cloud Platform Compute Engine VM instances page. The left sidebar is collapsed. The main area displays a table of VM instances. One instance, named "m21cs007", is listed with the following details:

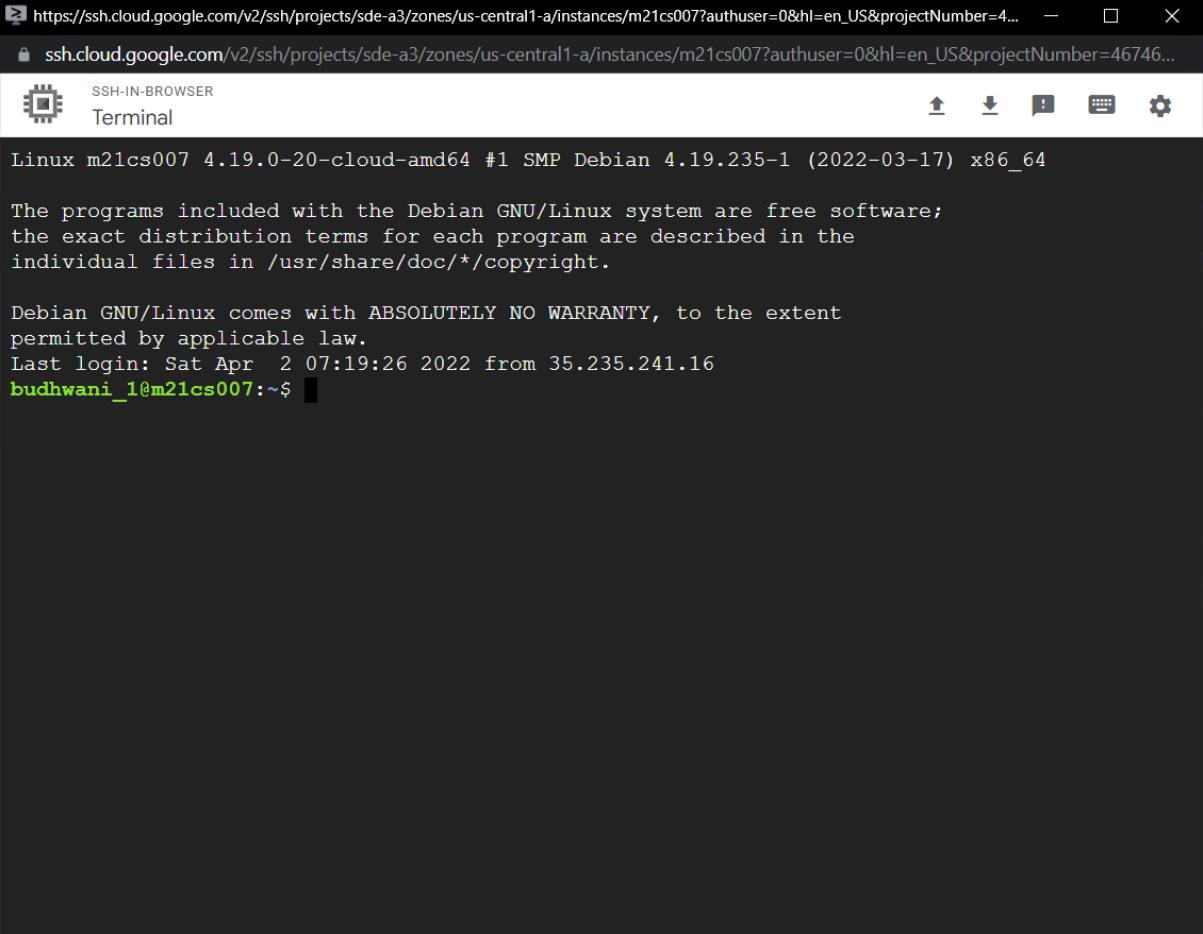
Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
Up	m21cs007	us-central1-a			10.128.0.3 (nic0)	34.135.107.217	SSH

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

The screenshot shows the same Google Cloud Platform Compute Engine VM instances page as before. A context menu has been opened over the "m21cs007" instance. The "Connect" option in the menu has a dropdown list with the following options:

- Open in browser window
- Open in browser window on custom port
- Open in browser window using provided private SSH key
- View gcloud command
- Use another SSH client

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



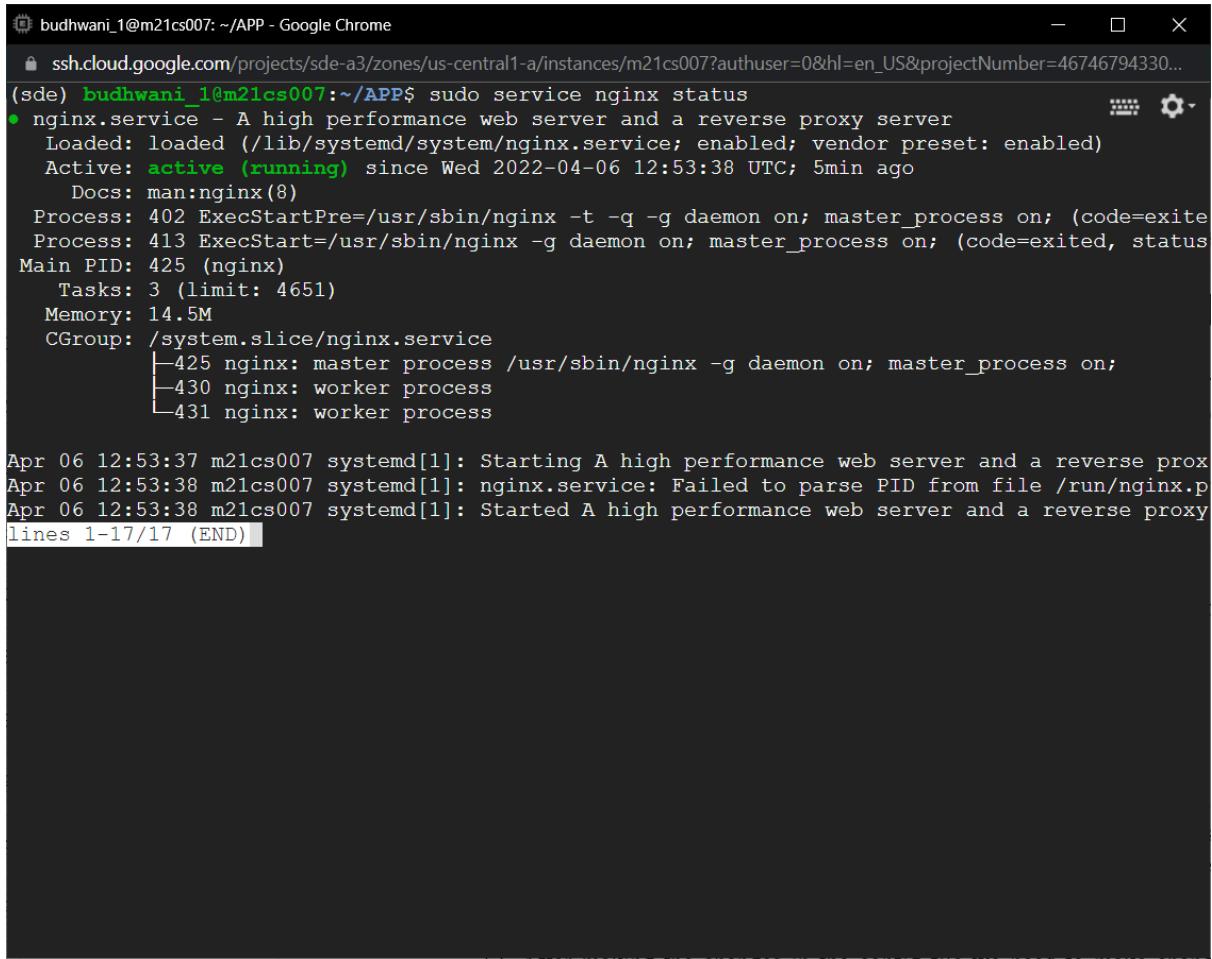
The screenshot shows an SSH session in a browser window titled "SSH-IN-BROWSER". The terminal window displays a Debian 4.19.235-1 (x86_64) system prompt. The output includes the kernel version, copyright information, and a login history entry for "budhwani_1@m21cs007".

```
https://ssh.cloud.google.com/v2/ssh/projects/sde-a3/zones/us-central1-a/instances/m21cs007?authuser=0&hl=en_US&projectNumber=4... - □ X
ssh.cloud.google.com/v2/ssh/projects/sde-a3/zones/us-central1-a/instances/m21cs007?authuser=0&hl=en_US&projectNumber=46746...
SSH-IN-BROWSER
Terminal
Linux m21cs007 4.19.0-20-cloud-amd64 #1 SMP Debian 4.19.235-1 (2022-03-17) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Apr  2 07:19:26 2022 from 35.235.241.16
budhwani_1@m21cs007:~$
```

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.

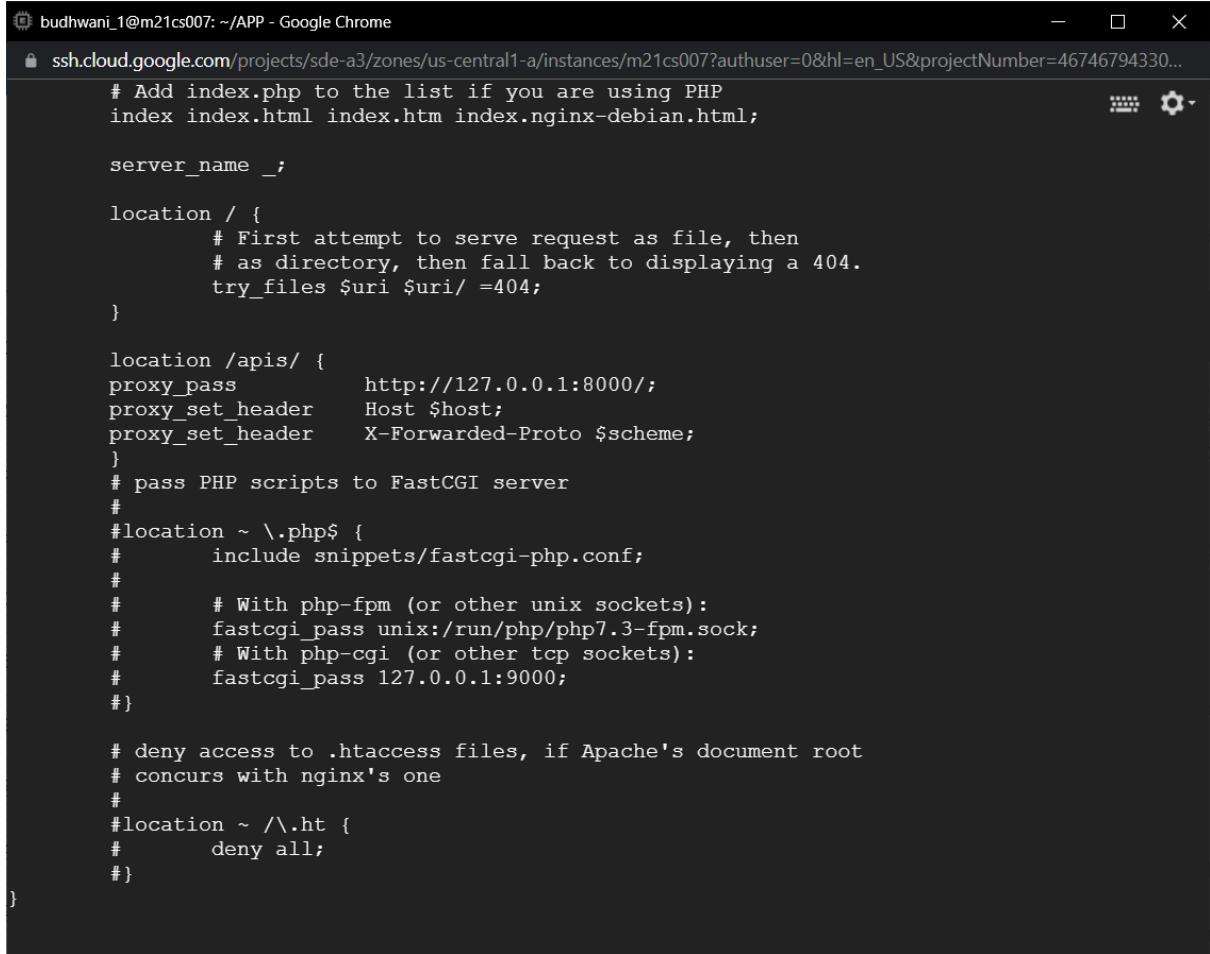


The screenshot shows a terminal window titled "budhwani_1@m21cs007: ~/APP - Google Chrome". The command "sudo service nginx status" is run, displaying the following output:

```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-04-06 12:53:38 UTC; 5min ago
     Docs: man:nginx(8)
  Process: 402 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0)
  Process: 413 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0)
 Main PID: 425 (nginx)
    Tasks: 3 (limit: 4651)
   Memory: 14.5M
      CGrou...  
Apr 06 12:53:37 m21cs007 systemd[1]: Starting A high performance web server and a reverse prox...
Apr 06 12:53:38 m21cs007 systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.p...
Apr 06 12:53:38 m21cs007 systemd[1]: Started A high performance web server and a reverse proxy
```

lines 1-17/17 (END)

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.



```
budhwani_1@m21cs007:~/APP - Google Chrome
ssh.cloud.google.com/projects/sde-a3/zones/us-central1-a/instances/m21cs007?authuser=0&hl=en_US&projectNumber=46746794330...
# Add index.php to the list if you are using PHP
index index.php index.html index.htm index.nginx-debian.html;

server_name _;

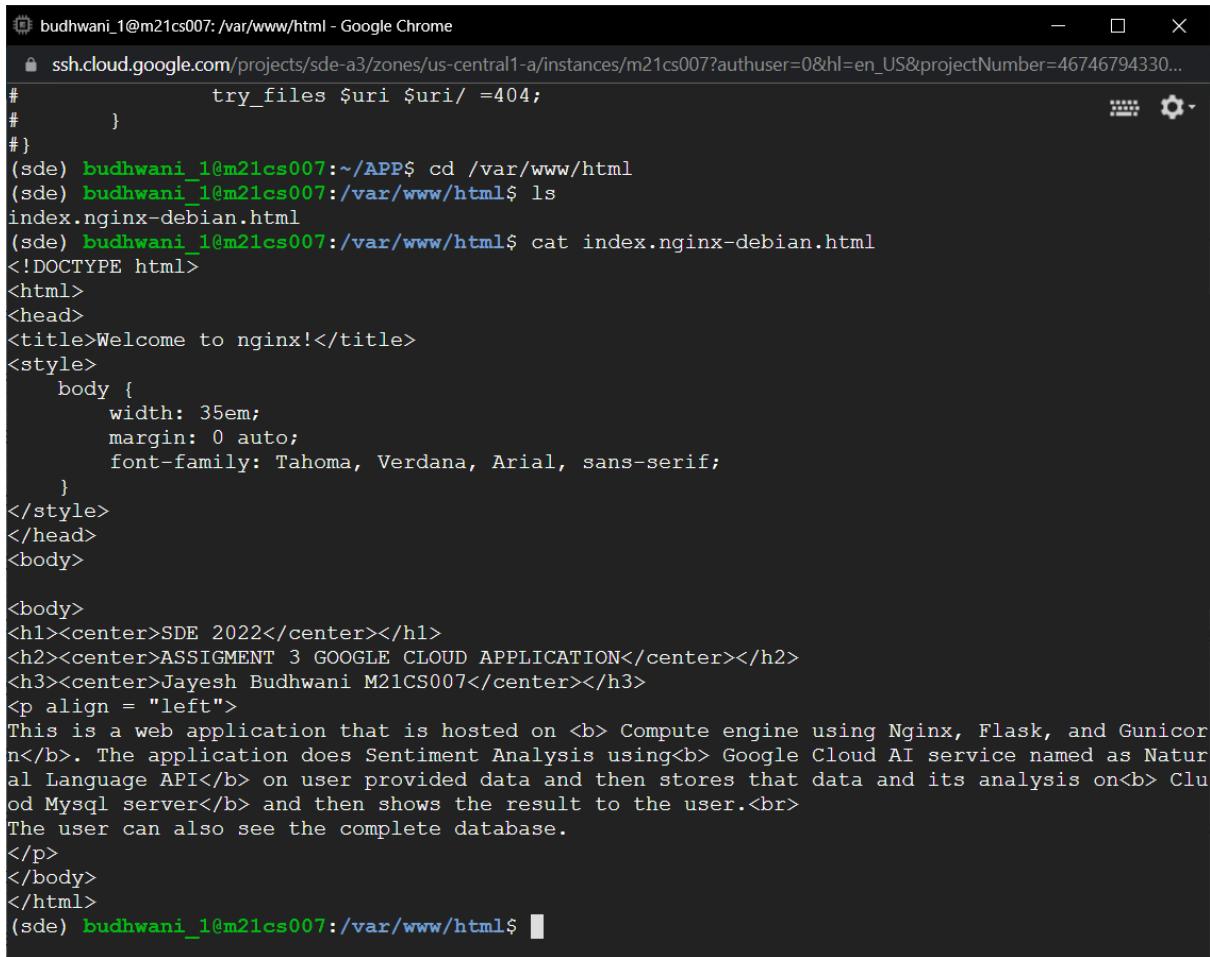
location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    try_files $uri $uri/ =404;
}

location /apis/ {
proxy_pass          http://127.0.0.1:8000/;
proxy_set_header    Host $host;
proxy_set_header    X-Forwarded-Proto $scheme;
}
# pass PHP scripts to FastCGI server
#
#location ~ \.php$ {
#    include snippets/fastcgi-php.conf;
#
#    # With php-fpm (or other unix sockets):
#    fastcgi_pass unix:/run/php/php7.3-fpm.sock;
#    # With php-cgi (or other tcp sockets):
#    fastcgi_pass 127.0.0.1:9000;
#}

# deny access to .htaccess files, if Apache's document root
# concurs with nginx's one
#
#location ~ /\.ht {
#    deny all;
#}

}
```

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`.



The screenshot shows a terminal window titled "budhwani_1@m21cs007: /var/www/html - Google Chrome". The window displays the content of the file "index.nginx-debian.html". The code is a standard HTML document with some CSS styles. It includes sections for the title, head, and body. The body contains a center-aligned h1, an h2, an h3, and a paragraph describing the application's purpose. The paragraph mentions Compute engine, Nginx, Flask, and Gunicorn, and describes the sentiment analysis process involving Google Cloud AI and Natural Language API.

```
#          try_files $uri $uri/ =404;
#
#}
(sde) budhwani_1@m21cs007:~/APP$ cd /var/www/html
(sde) budhwani_1@m21cs007:/var/www/html$ ls
index.nginx-debian.html
(sde) budhwani_1@m21cs007:/var/www/html$ cat index.nginx-debian.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
}
</style>
</head>
<body>

<body>
<h1><center>SDE 2022</center></h1>
<h2><center>ASSIGMENT 3 GOOGLE CLOUD APPLICATION</center></h2>
<h3><center>Jayesh Budhwani M21CS007</center></h3>
<p align = "left">
This is a web application that is hosted on <b> Compute engine using Nginx, Flask, and Gunicor n</b>. The application does Sentiment Analysis using<b> Google Cloud AI service named as Natur al Language API</b> on user provided data and then stores that data and its analysis on<b> Clu od Mysql server</b> and then shows the result to the user.<br>
The user can also see the complete database.
</p>
</body>
</html>
(sde) budhwani_1@m21cs007:/var/www/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**.

```
budhwani_1@m21cs007: ~ - Google Chrome
ssh.cloud.google.com/projects/sde-a3/zones/us-central1-a/instances/m21cs007?authuser=0&hl=en_US&projectNumber=46746794330...
(sde) budhwani_1@m21cs007:/var/www/html$ cd
(sde) budhwani_1@m21cs007:~$ sudo service nginx reload
(sde) budhwani_1@m21cs007:~$ sudo service nginx status
● nginx.service - A high performance web server and a reverse proxy server
  Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2022-04-06 12:53:38 UTC; 8min ago
    Docs: man:nginx(8)
Process: 402 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0)
Process: 413 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0)
Process: 618 ExecReload=/usr/sbin/nginx -g daemon on; master_process on; -s reload (code=exited, status=0)
Main PID: 425 (nginx)
  Tasks: 3 (limit: 4651)
 Memory: 15.1M
 CGroup: /system.slice/nginx.service
        ├─425 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
        ├─619 nginx: worker process
        └─620 nginx: worker process

Apr 06 12:53:37 m21cs007 systemd[1]: Starting A high performance web server and a reverse proxy...
Apr 06 12:53:38 m21cs007 systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid: Invalid argument
Apr 06 12:53:38 m21cs007 systemd[1]: Started A high performance web server and a reverse proxy...
Apr 06 13:02:25 m21cs007 systemd[1]: Reloading A high performance web server and a reverse proxy...
Apr 06 13:02:25 m21cs007 systemd[1]: Reloaded A high performance web server and a reverse proxy...
lines 1-20/20 (END)
(sde) budhwani_1@m21cs007:~$
```

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**

The screenshot shows a terminal window titled "budhwani_1@m21cs007: ~/APP - Google Chrome". The terminal displays the following steps:

- System status: Tasks: 3 (limit: 4651), Memory: 15.1M, CGroup: /system.slice/nginx.service, listing processes 425, 619, and 620.
- Log entries from April 6, 2022, showing the start of an nginx service and its reload.
- Terminal session history:
 - (sde) budhwani_1@m21cs007:~\$
 - (sde) budhwani_1@m21cs007:~\$ cd APP
 - (sde) budhwani_1@m21cs007:~/APP\$ ls
 - 'Voice List.csv' h.txt sde-a3-7511ca9379bf.json server.py test.py
 - __pycache__ sde sde-a3-ed0f1cb3a84e.json templates
- Gunicorn command execution:
 - (sde) budhwani_1@m21cs007:~/APP\$ source sde/bin/activate
 - (sde) budhwani_1@m21cs007:~/APP\$ gunicorn server:app
- Log output from gunicorn:
 - [2022-04-06 13:03:52 +0000] [628] [INFO] Starting gunicorn 20.1.0
 - [2022-04-06 13:03:52 +0000] [628] [INFO] Listening at: http://127.0.0.1:8000 (628)
 - [2022-04-06 13:03:52 +0000] [628] [INFO] Using worker: sync
 - [2022-04-06 13:03:52 +0000] [631] [INFO] Booting worker with pid: 631
- MySQL connection information: Engine(mysql+pymysql://root:***@34.122.45.49/NLP)

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.

The screenshot shows the Google Cloud Platform APIs & Services dashboard. On the left, there's a sidebar with 'Enabled APIs & services' listed: Library, Credentials, OAuth consent screen, Domain verification, and Page usage agreements. The main area has tabs for 'Traffic' and 'Median latency'. A search bar at the top right is set to 'Natural Language'. Below it, under 'PRODUCTS & PAGES', is a section titled 'Natural Language' with a single result: 'Learn to use Natural Language Interactive Tutorial'. Under 'DOCUMENTATION & TUTORIALS', there are links to 'Cloud Natural Language | Cloud Natural Language Documentation', 'Natural Language API Basics | Cloud Natural Language API Documentation', 'Natural Language release notes | Cloud Natural Language API Documentation', 'Healthcare Natural Language API | Cloud Healthcare API Documentation', and 'Natural Language client libraries | Cloud Natural Language API Documentation'. Under 'MARKETPLACE', there are links to 'Cloud Natural Language API | Google Enterprise API' and 'Natural Language Accelerator | Calculated Systems LLC'.

2. Enable the Cloud Natural Language API.

The screenshot shows the Cloud Natural Language API product page. At the top, there's a navigation bar with tabs for 'MANAGE', 'TRY THIS API', and 'API Enabled' (which is checked). Below that, there are 'OVERVIEW' and 'DOCUMENTATION' tabs, with 'OVERVIEW' currently selected. The 'OVERVIEW' section contains a large icon of a document with three lines, the text 'Cloud Natural Language API', 'Google Enterprise API', and a brief description: 'Provides natural language understanding technologies, such as sentiment analysis, entity...'. The 'DOCUMENTATION' section is partially visible below. To the right, under 'Additional details', it lists: Type: SaaS & APIs, Last updated: 7/23/21, Category: Machine learning, Google Enterprise APIs, and Service name: language.googleapis.com.

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

The screenshot shows the 'Create service account' dialog box in the Google Cloud Platform. The left sidebar is titled 'IAM & Admin' and includes options like IAM, Identity & Organization, Policy Troubleshooter, Policy Analyzer, Organization Policies, Service Accounts (which is selected), Workload Identity Federation, Labels, Tags, Settings, and Release Notes. The main area is titled 'Create service account' and contains three steps:

- 1 Service account details**: Fields include 'Service account name' (assignment3), 'Display name for this service account', 'Service account ID' (assignment3), 'Email address' (assignment3@sde-a3.iam.gserviceaccount.com), and 'Service account description'. A 'CREATE AND CONTINUE' button is present.
- 2 Grant this service account access to project (optional)**
- 3 Grant users access to this service account (optional)**

At the bottom are 'DONE' and 'CANCEL' buttons.

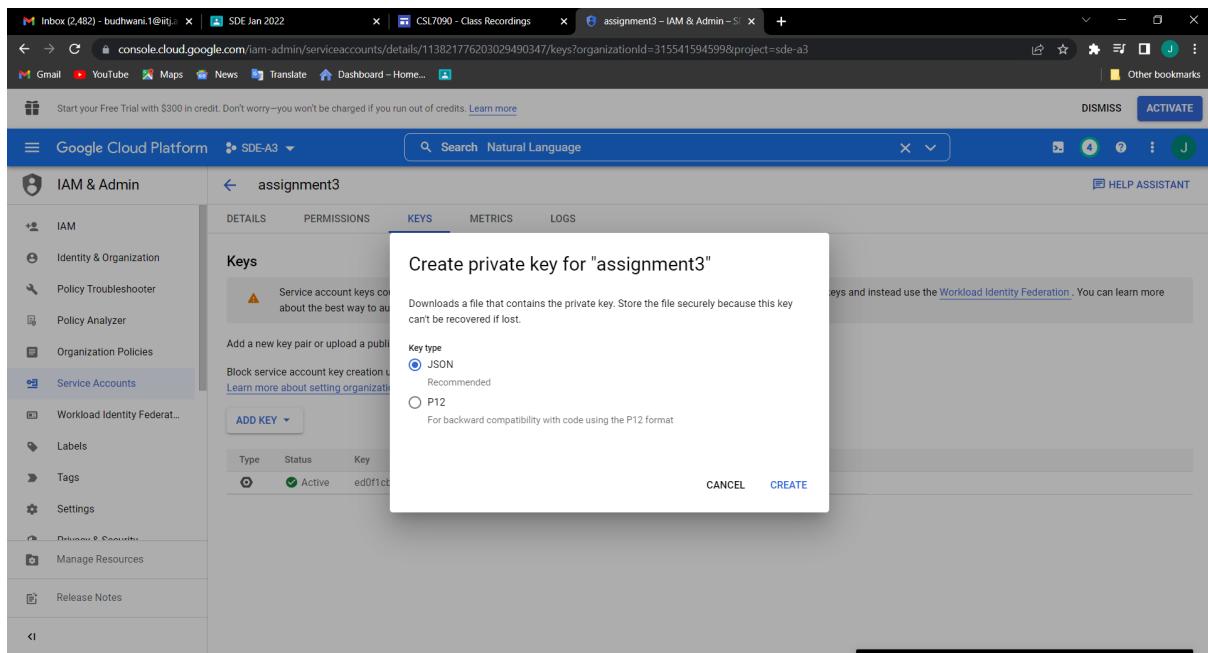
4. After creating the service account **create an account key in JSON format** under the **Manages keys** option.

The screenshot shows the 'Service accounts' list page in the Google Cloud Platform. The left sidebar is identical to the previous screenshot. The main area lists service accounts for the project 'SDE-A3'. There are three entries:

Email	Status	Name	Description	Key ID	Key creation date	OAuth 2	Actions
assignment3@sde-a3.iam.gserviceaccount.com	✓	assignment3		7511ca9379bf5106e2c1cdf44d7a35b395fc3e8	Mar 30, 2022	108160	⋮
assignment3@sde-a3.iam.gserviceaccount.com	✓	assignment3		ed0f1cb3a84ed8d753d051a16920552a57c9dcda	Mar 30, 2022	113821	⋮
467467943308-compute@developer.gserviceaccount.com	✓	Compute Engine default service account	No keys				⋮

A context menu is open over the third row, listing options: Manage details, Manage permissions, Manage keys, View metrics, View logs, Disable, and Delete.

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 like **Mysql**, Postgres, and SQL server. Mysql was chosen for this assignment.

The screenshot shows the 'Create an instance' page for Google Cloud SQL. At the top, there's a navigation bar with tabs for Gmail, YouTube, Maps, News, Translate, Dashboard, and Home. Below the navigation bar, there's a search bar with 'Search Natural Language'. The main content area is titled 'Choose your database engine' and lists three options: MySQL, PostgreSQL, and SQL Server. Each option has a small icon, a version range, and a 'Choose [Engine]' button. A note at the bottom says 'Want more context on the Cloud SQL database engines? Learn more'.

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.

The screenshot shows the 'Create a MySQL instance' configuration page. The left sidebar has a 'Create a MySQL instance' link. The main form is divided into sections: 'Instance info' (with fields for Instance ID, Password, and Database version), 'Choose region and zonal availability' (with Region set to 'us-central1 (Iowa)' and Zonal availability set to 'Single zone'), and a 'Summary' table on the right. The 'Summary' table contains the following data:

Region	us-central1 (Iowa)
DB Version	MySQL 5.7
vCPUs	4 vCPU
Memory	26 GB
Storage	100 GB
Network throughput (MB/s)	1,000 of 2,000
Disk throughput (MB/s)	Read: 48.0 of 240.0 Write: 48.0 of 240.0
IOPS	Read: 3,000 of 15,000 Write: 3,000 of 15,000
Connections	Public IP
Backup	Automated
Availability	Multiple zones (Highly available)
Point-in-time recovery	Enabled

Choose region and zonal availability

For better performance, keep your data close to the services that need it. Region is permanent, while zone can be changed any time.

Region

us-central1 (Iowa)

Zonal availability

Single zone
In case of outage, no failover. Not recommended for production.

Multiple zones (Highly available)
Automatic failover to another zone within your selected region. Recommended for production instances. Increases cost.

SPECIFY ZONES

Customize your instance

You can also customize instance configurations later

SHOW CONFIGURATION OPTIONS

CREATE INSTANCE **CANCEL**

Summary	
Region	us-central1 (Iowa)
DB Version	MySQL 5.7
vCPUs	4 vCPU
Memory	26 GB
Storage	100 GB
Network throughput (MB/s)	1,000 of 2,000
Disk throughput (MB/s)	Read: 48.0 of 240.0 Write: 48.0 of 240.0
IOPS	Read: 3,000 of 15,000 Write: 3,000 of 15,000
Connections	Public IP
Backup	Automated
Availability	Multiple zones (Highly available)
Point-in-time recovery	Enabled

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

Instances

SQL

Instance ID	Type	Public IP address	Private IP address	Instance connection name	High availability	Location	Storage used	Labels	Actions
m21cs007-sql	MySQL 5.7	34.122.45.49	10.128.0.3	sde-a3:us-central1:m21cs007-sql	ENABLED	us-central1-f	1 GB of 100 GB		⋮

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

The screenshot shows the Google Cloud Platform SQL Overview page for the instance 'm21cs007-sql'. The left sidebar lists 'PRIMARY INSTANCE' options: Overview (selected), Connections, Users, Databases, Backups, Replicas, and Operations. Below this are 'Release Notes' and a 'Connect to this instance' section with a Public IP address field containing '34.122.45.49'. The main area features a chart titled 'CPU utilization' showing a fluctuating line graph from April 2, 2022, at 2:01:11 AM to 10:00 PM. The chart includes a legend for 'CPU utilization' and a time range selector. To the right is a 'Configuration' section showing 'vCPUs: 4', 'Memory: 26 GB', and 'SSD storage: 100 GB'.

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

The screenshot shows the Google Cloud Platform SQL Connections Networking page for the instance 'm21cs007-sql'. The left sidebar is identical to the previous screenshot. The main area has tabs for 'NETWORKING' (selected), 'SECURITY', and 'CONNECTIVITY TESTS'. Under 'NETWORKING', there are sections for 'Instance IP assignment' (with 'Public IP' selected) and 'Authorized networks' (with a note about adding 0.0.0.0/0). A warning message at the bottom states: 'You have added 0.0.0.0/0 as an allowed network. This prefix will allow any IPv4 client to pass the network firewall and make login attempts to your instance.' Learn more.

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

The screenshot shows the Google Cloud Platform SQL Connections page. On the left, there's a sidebar with options like Overview, Connections (which is selected), Users, Databases, Backups, Replicas, and Operations. The main area is titled 'Connections' and shows a dropdown menu with 'ALL (0.0.0.0/0)' selected. A modal window titled 'New network' is open, with the 'Name' field containing 'ALL'. Below it, there's a 'Network' field with '0.0.0.0/0' selected. At the bottom of the modal are 'CANCEL' and 'DONE' buttons. Below the modal is a large 'ADD NETWORK' button. At the very bottom of the page, there are 'SAVE' and 'DISCARD CHANGES' buttons.

- 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:

- 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.
- 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.
- 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.
- Finally, after creating these files 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`
- Finally, go to the link `http://<IP_of_instance>` like to see the Main Home page.
- Go to `http://<IP_of_instance>/apis` like to see sub home page.
- Go to `http://<IP_of_instance>/apis/show` like to see the records present in Mysql Database.
- And finally go 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.

The screenshot shows the Google Cloud Platform Compute Engine Instance templates page. The left sidebar is collapsed, showing options like VM instances, Instance templates (which is selected), Sole-tenant nodes, Machine images, TPUs, Committed use discounts, and Migrate for Compute Engine. The main content area has a title 'Instance templates' with a sub-section 'Instance templates'. It includes a 'Filter' section with columns for Name, Machine type, Image, Disk type, Placement policy, In use by, Creation time, and Actions. Below this is a large graphic of a globe with colored dots (yellow, blue, red) representing different regions or data centers. A text box explains what an instance template is: 'An instance template lets you describe a VM instance. You can then create groups of identical instances based on the template.' At the bottom is a prominent blue 'CREATE INSTANCE TEMPLATE' button.

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

The screenshot shows the 'Create an instance template' dialog. The left sidebar is collapsed, showing the same list of Compute Engine options as the previous screenshot. The main form has a title 'Create an instance template'. It starts with a descriptive text: 'Describe a VM instance once and then use that template to create groups of identical instances' with a 'Learn more' link. Below this is a 'Name' field containing 'm21cs007-sde'. There's also a 'Labels' section with a '+ ADD LABELS' button. The next section is 'Machine configuration' with a 'Machine family' dropdown set to 'GENERAL-PURPOSE'. Underneath are 'Series' (set to 'E2'), 'CPU platform selection based on availability', and 'Machine type' (set to 'e2-small (2 vCPU, 2 GB memory)'). On the right side, there's a note about estimated costs: 'These are estimated costs for a VM instance created using this template: Monthly estimate \$13.23 That's about \$0.02 hourly' and a note about billing: 'Pay for what you use: No upfront costs and per second billing'. At the bottom are 'CREATE', 'CANCEL', and 'EQUIVALENT COMMAND LINE' buttons.

- After creating the template it will look as below.

The screenshot shows the Google Cloud Platform Compute Engine Instance templates page. The left sidebar is collapsed, and the main header says "Instance templates". Below the header, there's a search bar and several action buttons: "CREATE INSTANCE TEMPLATE", "REFRESH", "CREATE INSTANCE GROUP", "COPY", "DELETE", "SHOW INFO PANEL", and "LEARN". A banner at the top encourages a free trial with \$300 in credit. The main content area displays a table titled "Filter: Filter instance templates". The table has columns: Name, Machine type, Image, Disk type, Placement policy, In use by, Creation time, and Actions. There is one entry: "m21cs007-sde" (e2-small, debian-10-buster-v20220317, Balanced persistent disk, No policy, None, Apr 6, 2022, 12:16:46 PM UTC+05:30). The left sidebar lists other Compute Engine resources like VM instances, Sole-tenant nodes, Machine images, TPUs, Committed use discounts, and Migrate for Compute Engine.

Step 2: Create an Instance Group

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

The screenshot shows the Google Cloud Platform Compute Engine Instance template details page for "m21cs007-sde". The left sidebar is collapsed, and the main header says "m21cs007-sde - Compute Engine". Below the header, there's a search bar and several action buttons: "+ CREATE VM", "+ CREATE SIMILAR", "+ CREATE INSTANCE GROUP", and "DELETE". A banner at the top encourages a free trial with \$300 in credit. The main content area is divided into sections: "Basic information", "Machine configuration", and "Networking". The "Basic information" section contains fields: Name (m21cs007-sde), Type (Instance Template), Creation time (Apr 6, 2022, 12:16:46 PM UTC+05:30), In use by (None), Reservations (Automatically choose), Labels (None), Placement policy (No policy), and Confidential VM service (Disabled). The "Machine configuration" section contains fields: Machine type (e2-small), Minimum CPU platform (None), vCPUs to core ratio (—), Display device (Disabled), and GPUs (None). The "Networking" section contains fields: Public DNS PTR Record (None), Total egress bandwidth tier (—), and NIC type (—).

- 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.

The screenshot shows the 'Create Instance Group' page in the Google Cloud Platform. The 'New managed instance group (stateless)' option is selected. The 'Name' field is set to 'm21cs007-group'. The 'Description' field is empty. The 'Instance template' dropdown is set to 'm21cs007-sde'. The 'Number of instances' dropdown is set to 'Based on autoscaling configuration'. Under the 'Location' section, the 'Multiple zones' radio button is selected. At the bottom, there are 'CREATE', 'CANCEL', and 'EQUIVALENT COMMAND LINE' buttons.

The screenshot shows the 'Create Instance Group' page with more detailed configurations. In the 'LOCATION' section, 'Multiple zones' is selected, and the 'Region' dropdown is set to 'us-central1 (Iowa)' with 'Zones' set to 'us-central1-c, us-central1-f, and us-central1-b'. The 'Target distribution shape' dropdown is set to 'Even'. Under the 'AUTOSCALING' section, 'Enable instance redistribution' is checked. The 'AUTOSCALING MODE' dropdown is set to 'On: add and remove instances to the group'. The 'Minimum number of instances' is 1 and the 'Maximum number of instances' is 2. A note at the bottom states: 'To maximize availability the minimum number of instances should be at least equal to the number of zones.' Additional notes mention 'Autoscaling mode' and 'Autoscaling metric'.

The screenshot shows the 'Create Instance Group' page in the Google Cloud Platform. The 'Autoscaling metrics' section is highlighted, showing the configuration for scaling based on CPU utilization (target 70%). The 'Predictive autoscaling' section is also visible, with the option 'Optimize for availability' selected.

- 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.

The screenshot shows the 'Create Instance Group' page in the Google Cloud Platform. The 'Health Check' section is highlighted, showing the configuration for a health check named 'm21cs007-health-check' with protocol 'HTTP' and port '80'. The 'Logs' section is also visible, with the option 'Off' selected.

Health Check

Proxy protocol: NONE

Request path: /

Logs: Turning on Health check logs can increase costs in Cloud Logging.
 On
 Off

Health criteria

Define how health is determined: how often to check, how long to wait for a response, and how many successful or failed attempts are decisive.

Check interval *	5 seconds	Timeout *	5 seconds
Healthy threshold *	2 consecutive successes		
Unhealthy threshold *	2 consecutive failures		

CREATE **CANCEL** **EQUIVALENT COMMAND LINE** **SAVE** **CANCEL**

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

Instance groups - Compute Engine

DISMISS ACTIVATE

Search Products, resources, docs (/)

Compute Engine Instance groups CREATE INSTANCE GROUP REFRESH DELETE HELP ASSISTANT LEARN

Virtual machines

- VM instances
- Instance templates
- Sole-tenant nodes
- Machine images
- TPUs
- Committed use discounts
- Migrate for Compute Eng...

Storage

- Disks
- Snapshots
- Marketplace
- Release Notes

Instance groups are collections of VM instances that use load balancing and automated services, like autoscaling and autohealing. [Learn more](#)

Filter Enter property name or value

Status	Name	Instances	Template	Group type	Creation time	Recommendation	Autoscaling	Zone	In Use By
<input checked="" type="checkbox"/>	m21cs007-group	1	m21cs007-sde	Managed	Apr 6, 2022, 12:27:00 PM UTC+05:30		On: Target CPU utilization 70%	us-central1 (3/4)	

Instance group is transforming

Creating instance group "m21cs007-group"...

- 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.

The screenshot shows the Google Cloud Platform Network services page. The left sidebar has 'Network services' selected, with 'Load balancing' highlighted. The main content area features a section titled 'Get real-time analytics with Network Intelligence Center' and a 'LOAD BALANCERS' tab. Below this, there's a 'Load balancing' section with a 'CREATE LOAD BALANCER' button. A note at the bottom says 'To edit load balancing resources like forwarding rules and target proxies, go to the'.

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

The screenshot shows the 'Create a load balancer' page. The left sidebar is identical to the previous screenshot. The main content area displays three options: 'HTTP(S) Load Balancing' (selected), 'TCP Load Balancing', and 'UDP Load Balancing'. Each option has a 'Configure' section and an 'Options' section. Buttons for 'START CONFIGURATION' are located at the bottom of each section.

- Provide the below options.

Please answer a few questions to help us select the right load balancing type for your application

Internet facing or internal only

Do you want to load balance traffic from the Internet to your VMs or serverless services, or only between VMs in your network?

From Internet to my VMs or serverless services

Only between my VMs

Global or Regional

Do you want to deploy your application in global, regional, or classic mode?

Global HTTP(S) Load Balancer (classic)

Global HTTP(S) Load Balancer PREVIEW

Regional HTTP(S) Load Balancer PREVIEW

CONTINUE

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

Name * — m21cs007-backend

Description

Backend type — Instance group

Protocol — HTTP Named port * — http

Timeout * — 30 seconds

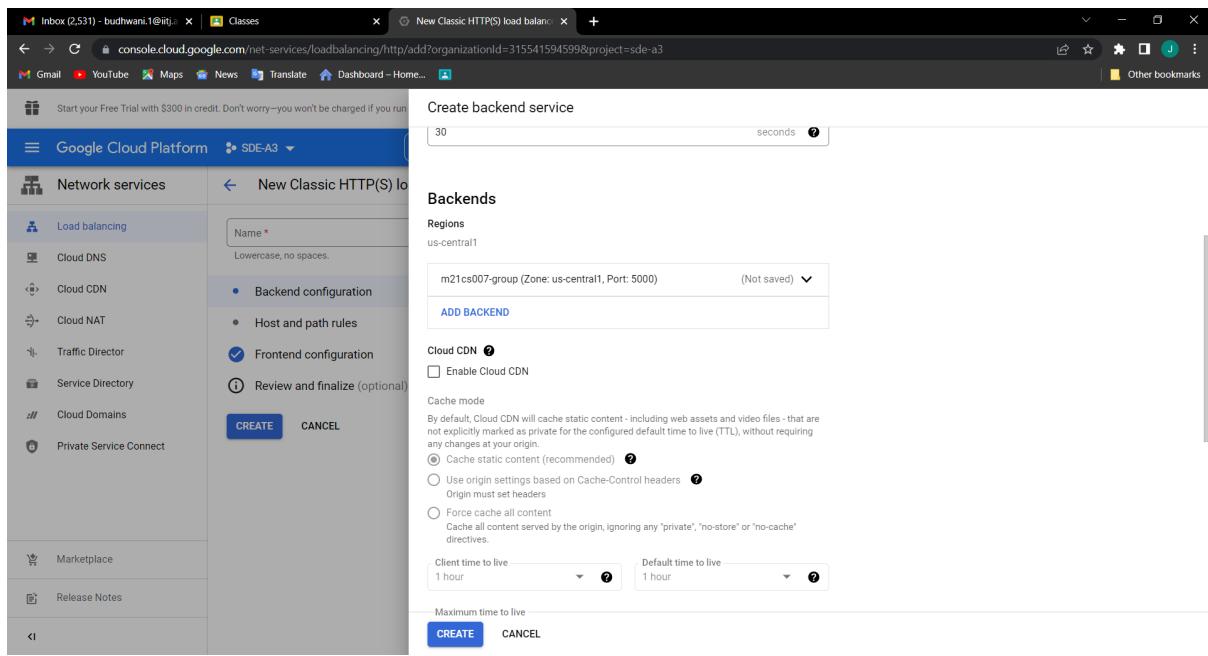
Backends

Regions — us-central1

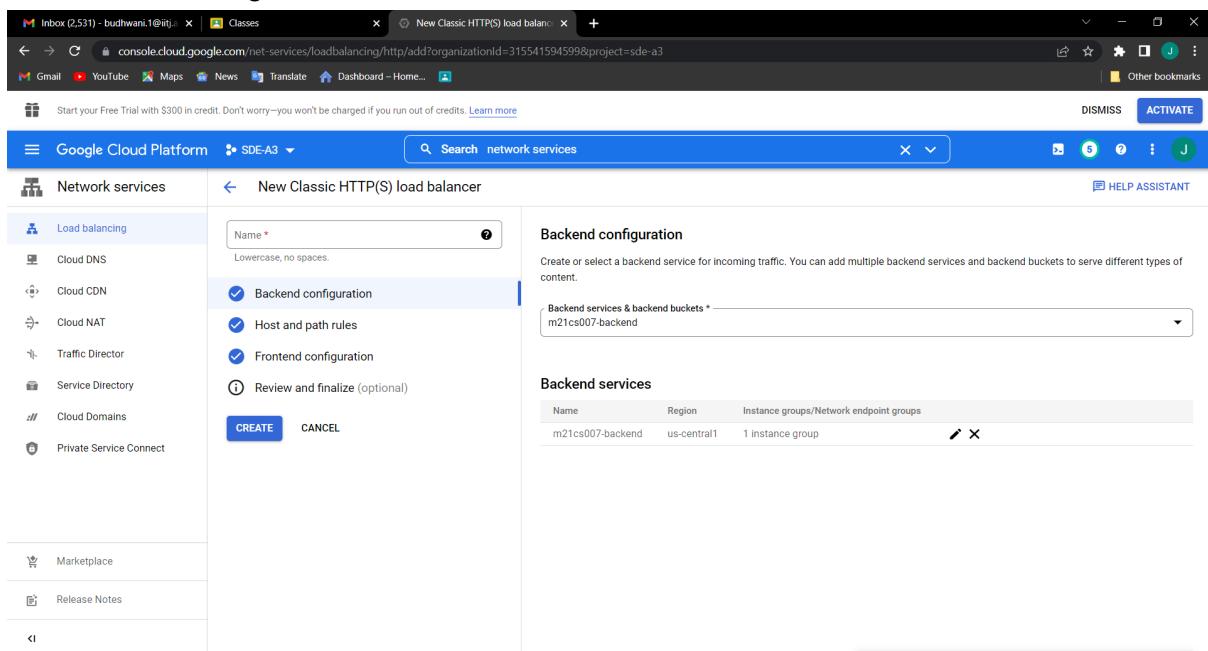
New backend

Instance group * — m21cs007-group

CREATE



- 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.

The screenshot shows the Google Cloud Platform Network services Load balancing page. The left sidebar is collapsed, showing options like Load balancing, Cloud DNS, Cloud CDN, Cloud NAT, Traffic Director, Service Directory, Cloud Domains, Private Service Connect, Marketplace, and Release Notes. The main content area has a header with a search bar and buttons for CREATE LOAD BALANCER, REFRESH, and DELETE. Below the header, there are tabs for LOAD BALANCERS, BACKENDS, and FRONTENDS. A callout message about Cloud CDN and Cloud Armor is visible. A table lists the load balancer, with one row for m21cs007-loadbalancer (HTTP(S) (Classic), HTTP protocol, 1 backend service). A note at the bottom suggests viewing or deleting forwarding rules and target proxies.

The screenshot shows the Google Cloud Platform Network services Load balancer details page for the m21cs007-loadbalancer. The left sidebar is collapsed, showing the same set of network services options. The main content area shows the load balancer details. At the top, there are buttons for EDIT, DELETE, and VIEW IN NETWORK TOPOLOGY. The Frontend section shows a single entry for HTTP on port 34.149.43.252:80 with a Premium Network Tier. The Host and path rules section shows a single unmatched rule for all hosts and paths pointing to the m21cs007-backend. The Backend section shows the m21cs007-backend with its configuration: Endpoint protocol (HTTP), Named port (http), Timeout (30 seconds), Health check (m21cs007-health-check), and Cloud CDN (Disabled). The ADVANCED CONFIGURATIONS section shows a table for the m21cs007-group, listing Name (m21cs007-group), Type (Instance group), Zone (us-central1), Healthy (2 of 2), Autoscaling (On: Target CPU utilization 70%), Balancing mode (Max RPS: 1 (per instance)), Selected ports (5000), and Capacity (100%).

- 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.

The screenshot shows the Google Cloud Platform Compute Engine Instance Groups page. The instance group 'm21cs007-group' has 2 instances, both marked as '100% healthy'. The 'OVERVIEW' tab is selected. The left sidebar shows 'Compute Engine' and 'Instance groups' is selected. The right sidebar shows 'Autoscaling' set to 'On (min 2, max 3)'.

Status	Name	Creation Time	Template	Per instance config	Internal IP	External IP	Health Check Status	Connect
<input type="checkbox"/>	m21cs007-group-cs7k	Apr 6, 2022, 1:20:58 PM UTC+05:30	m21cs007-sde		10.128.0.8 (nic0)	35.232.198.124	<input checked="" type="checkbox"/> Healthy	SSH
<input type="checkbox"/>	m21cs007-group-kxhs	Apr 6, 2022, 12:27:29 PM UTC+05:30	m21cs007-sde		10.128.0.5 (nic0)	35.225.138.182	<input checked="" type="checkbox"/> Healthy	SSH

- 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**

The screenshot shows a browser window with multiple tabs. The active tab displays the text 'SDE 2022' and 'ASSIGMENT 3 GOOGLE CLOUD APPLICATION' followed by 'Jayesh Budhwani M21CS007'. Below this, a paragraph describes the application's functionality, mentioning Compute engine using Nginx, Flask, and Gunicorn, and its integration with Google Cloud AI services like Natural Language API and Cloud MySQL server.

SDE 2022

ASSIGMENT 3 GOOGLE CLOUD APPLICATION

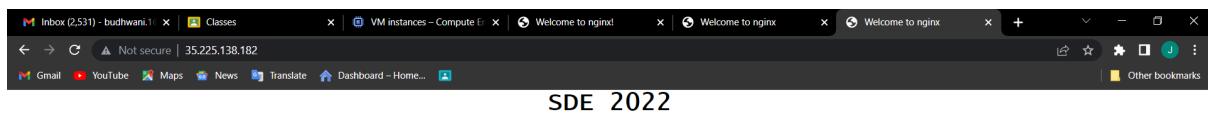
Jayesh Budhwani M21CS007

This is a web application that is hosted on **Compute engine using Nginx, Flask, and Gunicorn**. The application does Sentiment Analysis using **Google Cloud AI service named as Natural Language API** on user provided data and then stores that data and its analysis on **Cloud MySQL server** and then shows the result to the user. The user can also see the complete database.

- **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>