AWS Deployment

DASS Team-33

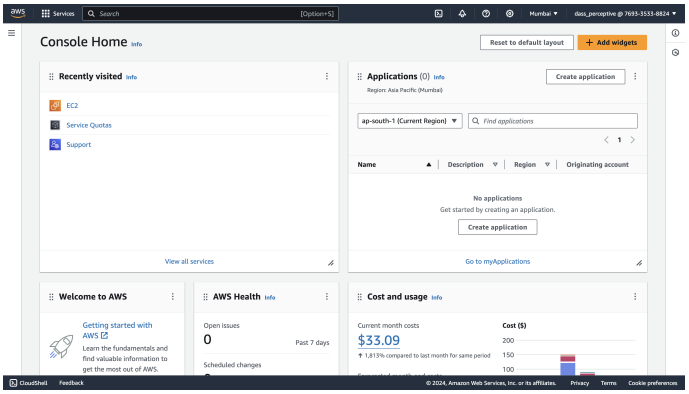
IIIT Hyderabad - April 19, 2024

Contents

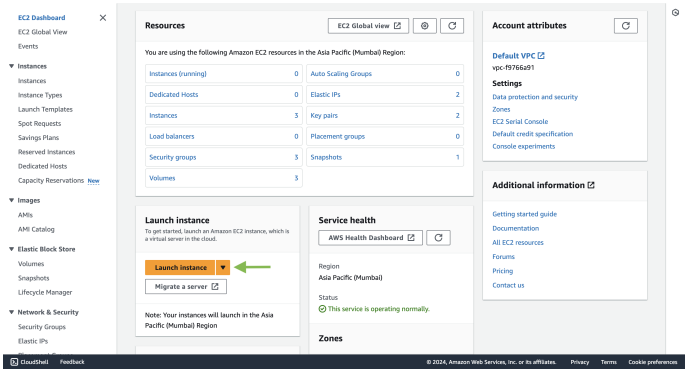
1 Creating AWS Instance

- Navigate to your AWS EC2 instance dashboard. Ensure that the location set in the dashboard is ap-south-1 or Asia Pacific(Mumbai).

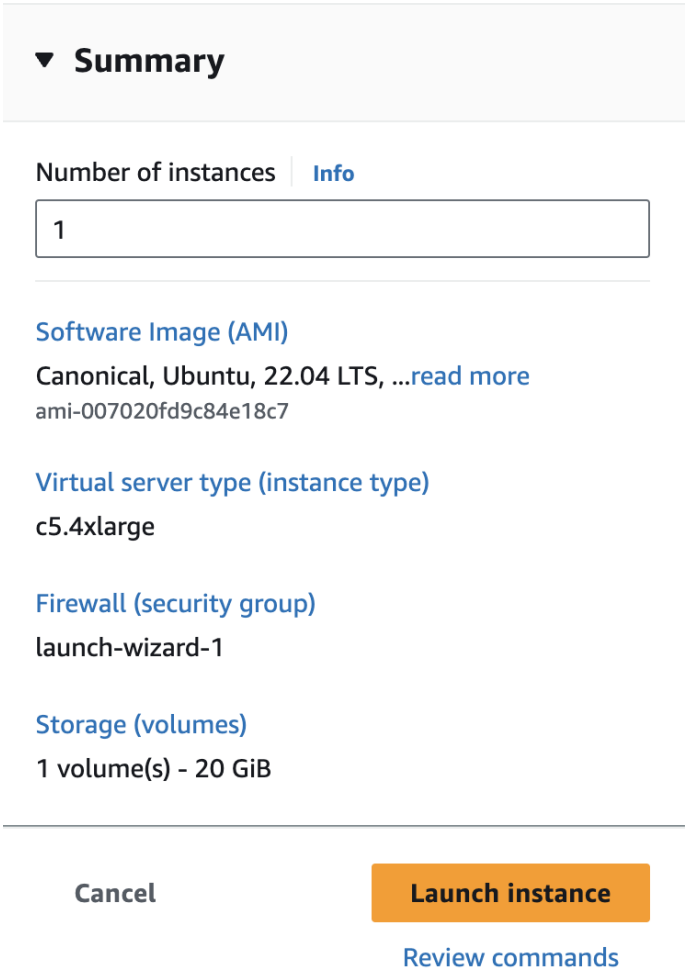
  INSERT IMAGE HERE



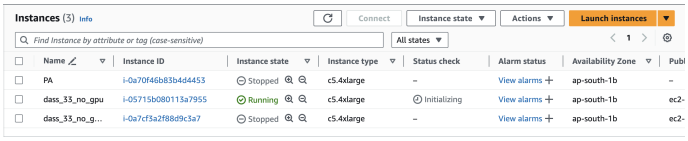
- Click on the "Launch Instance" option.



- Once you are navigated to the Launch Instance Page, select the following specs from all the options:
  - Amazon Machine Image (AMI): Ubuntu Server 22.04 LTS (HVM), SSD Volume Type, 64-bit (x86) architecture
  - Instance Type: c5.4xlarge
  - Key Pair Login: dass_33_k1.pem (provided in the codebase) or create a new RSA Key Pair
  - Network settings → Firewall (security groups): select existing security group → launch-wizard-1
  - Configure storage: 1x20 GiB gp2 Root Volume



- The security group "launch-wizard-1" allows all traffic through it, however if you want to customize a new security group, you can do so. But please make sure to have some idea on managing the requests.
- Do not add any extra EBS storage, 20 GB root will be sufficient.
- You can create your own .pem file, but make sure to have that file stored somewhere locally. The pem file acts as a key to our EC2 instance. The key that we have used (dass_33_k1.pem) is also attached in the codebase for you to refer.
- Now review the instance details and click launch. After few seconds our instance will be up and running. Navigate back to instance page, and reload. The instance state will be "Running".



2 SSH and SCP installation (for both Windows and Linux)

2.1 SSH Installation

- Press the following command in your terminal or command prompt.

```
ssh -i {path to .pem file} ubuntu@{public ipv4 address}
```

The .pem file should be locally stored in your local machine. IPv4 address can be easily obtained from the EC2 instance dashboard by clicking on the instance ID.

- If you find any error saying that the .pem file is public, execute the following command and try again the above one

```
sudo chmod 400 {path to .pem file}
```

- If you need the AWS doc procedure to connect, select the instance in your EC2 console and click on the connect button. You will be redirected to a new page named "Connect to instance", now select the SSH client and follow the steps.

2.2 SCP Installation

- SCP is used for transferring files between remote servers. In our scenario, we often need to add and remove images into our server. For the initial setup of backend files, we will be needing the backend directory to be sent into our server as well.
- Follow the following syntax for transferring any file/directory to the instance:

```
scp -i {path to .pem file} -r {path to file you want to send} \
ubuntu@{public ipv4 address}:{path to the directory on server}
```

For Example:

```
scp -i ./Downloads/dass_33_k1.pem -r ./Desktop/dass_33/code/backend \
ubuntu@23.333.45.67:/home/ubuntu
```

- Do not terminate the file transfer on your local host terminal, some files may take time to get uploaded given their larger sizes. Cross check on the EC2 instance whether the files are received or not after the command is completely executed on local host.
- If you have to transfer image/video files/directories to backend and modify their location, you also need to update **"/imagedetect"** and **"/videodetect"** routes in app.py file.
- Note: Do not enter the command with the backslash, it is just to indicate that it is a continuation command, with only a single space between the respective parts.

3 Environment Set Up in AWS Instance

- Follow the steps mentioned in the backend environment setup (quickstart.pdf). You will be good to go. Please do not forget to install the mongo db server.

4 Hosting the app on web

4.1 Gunicorn

- In the virtual environment created you have to install the gunicorn module by pressing the command:

```
pip install gunicorn
```

  Remember to activate the environment created before pressing any commands, with command:

```
source <location-to-envt>/bin/activate
```

- Systemd is a boot manager for Linux. We are using it to restart gunicorn if the EC2 restarts or reboots for some reason. We create a ¡projectname¿.service file in the /etc/systemd/system folder and specify what would happen to the gunicorn when the system reboots.
  - Execute the command:

```
sudo vim /etc/systemd/system/gbackend.service
```

    An empty file will be open on VIM.
  - Copy and Paste these service lines into the above file:

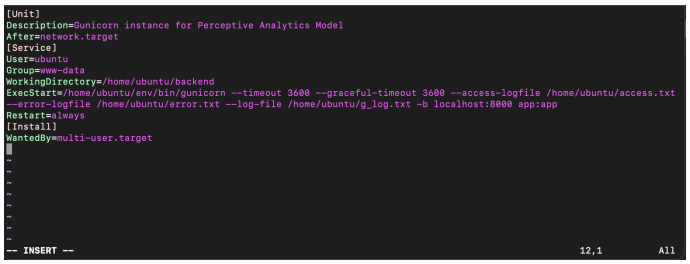
```
[Unit]
Description=Gunicorn instance for Perceptive Analytics Model
After=network.target


[Service]
User=ubuntu
Group=www-data
WorkingDirectory=/home/ubuntu/backend

ExecStart=/home/ubuntu/env/bin/gunicorn --timeout 3600 \
--graceful-timeout 3600 --access-logfile \
/home/ubuntu/access.txt --error-logfile \
/home/ubuntu/error.txt --log-file /home/ubuntu/g_log.txt \
-b localhost:8000 app:app
Restart=always

[Install]
WantedBy=multi-user.target
```

    This is supposed to look like this:



- **Note: Keep the name of virtual envt as "env" and store it in the /home-/ubuntu directory. backend should also be stored in /home/ubuntu**
- **The backslash here means that it is in a single line and is only intended for documentation purpose. All the pink part in the image indicates a single line command.**
- Save and Exit the vim file by clicking **ESC** and then press **:wq**.

4.2 Nginx

- Install the Nginx by using the commands:

```
sudo apt-get update
sudo apt-get install nginx
```

- Open the default file of Nginx file by using the command:

```
sudo vim /etc/nginx/sites-available/default
```

and make the following changes in the file:

- Add the upstream named backend and specify the localhost. Paste these lines of code in the file above the server block in the file you have just opened.

```
upstream backend {
    server 127.0.0.1:8000;
}
```

- Now add the created upstream proxy_pass into the "location /" block should look like this:

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

It will look like this:

```
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#
upstream backend {
        server 127.0.0.1:8000;
}

server {
        listen 80 default_server;
        listen [::]:80 default_server;

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index 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;
                proxy_pass http://backend;
        }

        # pass PHP scripts to FastCGI server
        #
        #location ~ \.php$ {
        #       include snippets/fastcgi-php.conf;
```

– Save and Exit the vim file by clicking **ESC** and then press **:wq**.

– Now open the Nginx config file by command:

```
sudo vim /etc/nginx/nginx.conf
```

– Now paste these lines in that file under "Basic Settings" comment:

```
proxy_read_timeout 3600;
proxy_connect_timeout 3600;
proxy_send_timeout 3600;
client_max_body_size 100M;
client_body_timeout 3600;
client_header_timeout 3600;
```

It will look like this:

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
        worker_connections 768;
        # multi_accept on;
}

http {

        ##
        # Basic Settings
        ##

        proxy_read_timeout 3600;
        proxy_connect_timeout 3600;
        proxy_send_timeout 3600;
        client_max_body_size 100M;
        client_body_timeout 3600;
        client_header_timeout 3600;

        sendfile on;
        tcp_nopush on;
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;
```

  - Save and Exit the vim file by clicking **ESC** and then press **:wq**.

- Now the configuration of both Nginx and Gunicorn is complete, and now we need to move to testing of our installations.

5 The Last Step and Verification

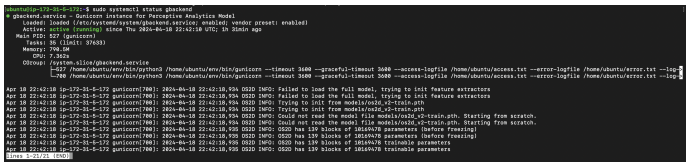
5.1 Last Step

- Once you have completed all the steps mentioned above, you have to restart the demon and configure minor changes as mentioned below.

```
sudo systemctl daemon-reload
sudo sysemctl start gbackend
sudo systemctl enable gbackend
sudo systemctl start nginx
sudo systemctl enable gbackend
sudo systemctl start mongod
sudo systemctl enable mongod
sudo systemctl daemon-reload
sudo systemctl restart gbackend
sudo systemctl restart nginx
```

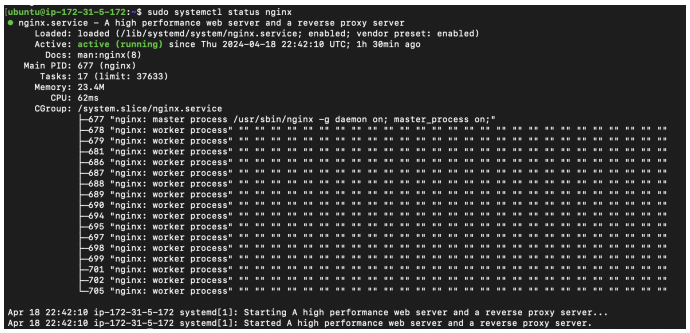
- To test whether gbackend and nginx are running, run the following commands:
  - `sudo systemctl status gbackend`

    which will give output as:



– `sudo systemctl status nginx`

which will give output as:



- Whenever any changes are made to any files in the backend, please reload the daemon, and restart both gbackend and nginx.

5.2 Verification

- Run the command:

```
curl localhost
```

and if everything is correctly installed, then the output will be:

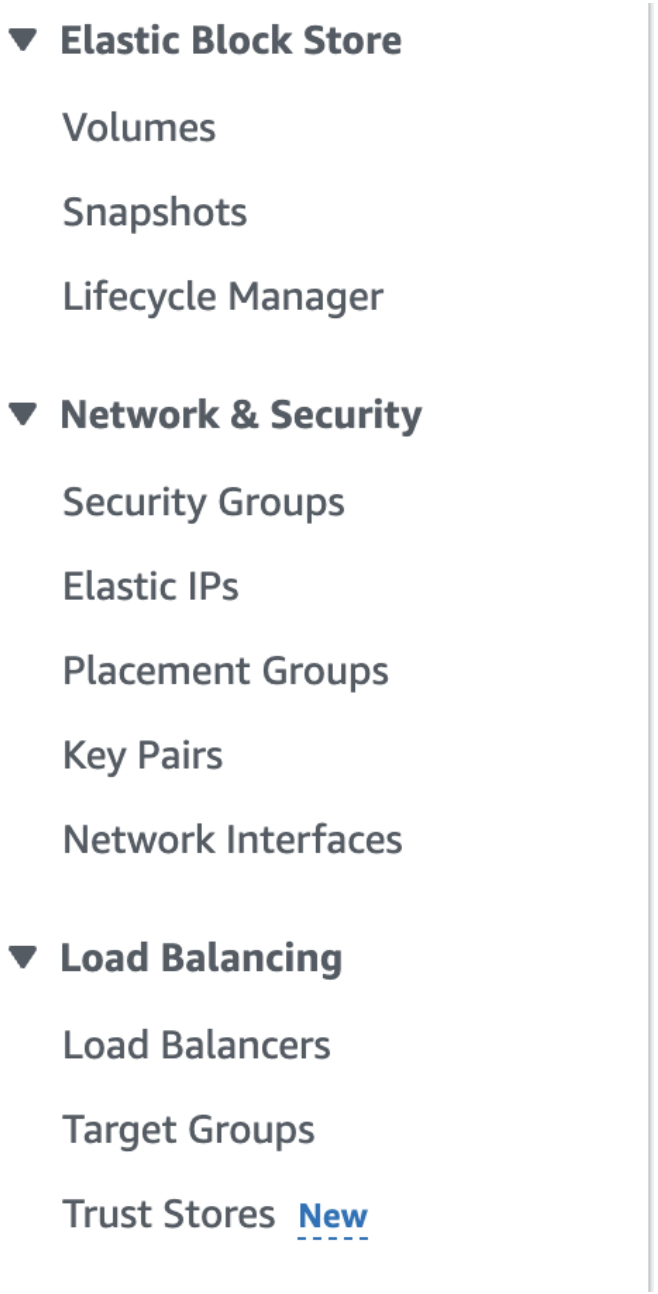
```
{"login":"YEAH"}
```

- In case of an error, if you need to have a look at the logs of Nginx and gunicorn. The log files are at the below location (they sometime also show information about the errors):
  - /var/log/nginx/error.log - Error log of Nginx
  - /var/log/nginx/access.log - Access log of Nginx
  - /home/ubuntu/access.txt - Access log of gunicorn
  - /home/ubuntu/error.txt - Error log of gunicorn
  - /home/ubuntu/g_log.txt - log of processing of gunicorn

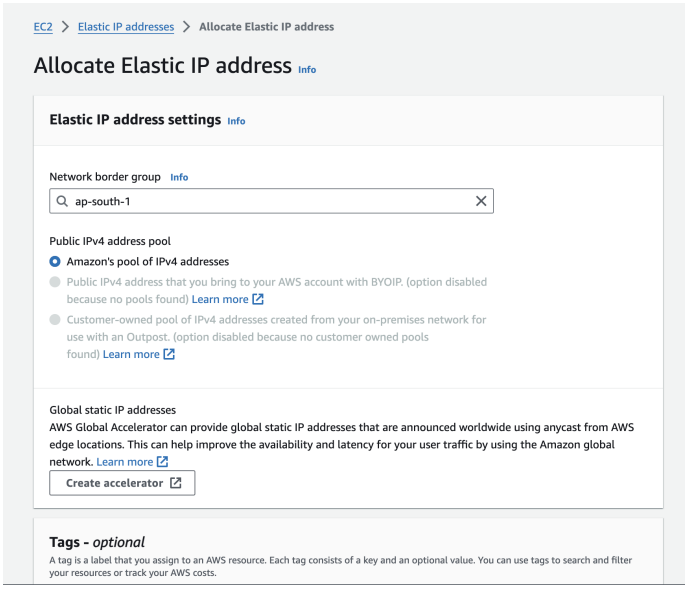
6 Attaching Elastic IP

You can attach an elastic IP to an EC2 instance for keeping the IP address constant after quick reboots. Follow the below steps to attach an elastic IP:

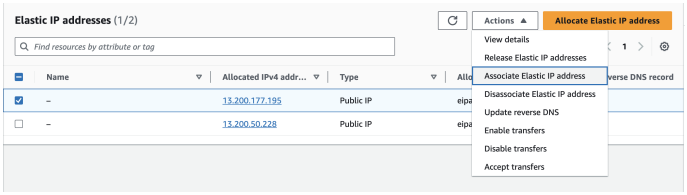
- Navigate to the elastic IP which can be seen on the left dashboard of the EC2 console (network and security section).



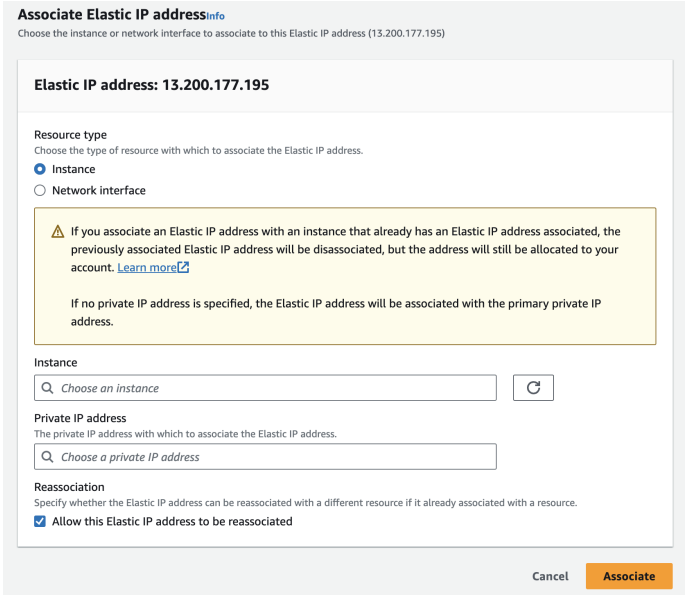
- Click on the Allocate Elastic IP address on the top right corner.
- Select amazon pool of IPv4 addresses. And click on allocate.



- After allocating click on actions of the selected/created IP address and click on the "Associate elastic IP address". It will ask you for the IPv4 of existing EC2 instances.



- Check box "Allow this Elastic IP address to be reassociated" and click on "Associate".



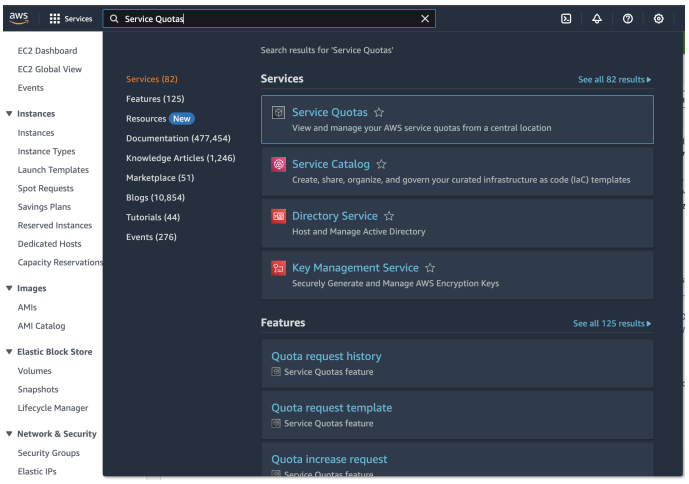
- If you have not allocated Elastic IP to an instance, and are using public IPv4 address (that changes everytime instance is started), you may get 404 error in access.log of nginx, so to avoid that, you need to use Elastic IP, and accordingly update "Constants.js" file in frontend.

7 GPU Instance Request

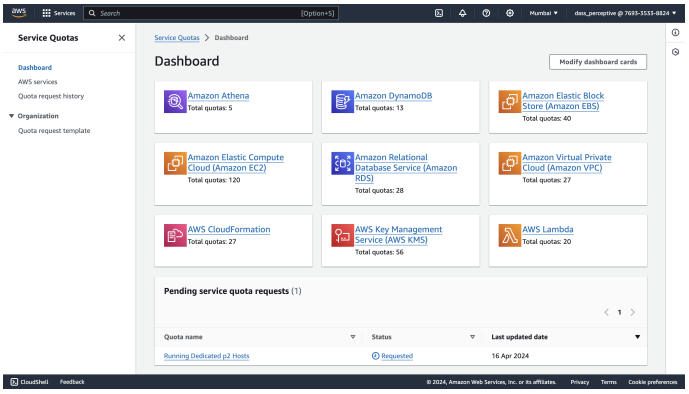
This is a tentative procedure for initiating a GPU instance, however due to lack of time in the project, we were not able to launch the instance and do the testing on it. Refer to this video for any references.

7.1 Requesting Quota Increase

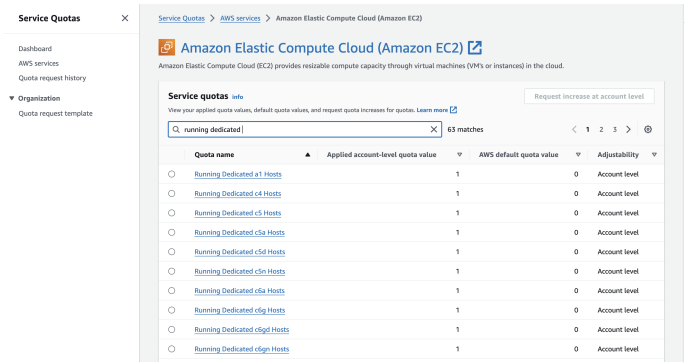
- Navigate to AWS Dashboard, and search for "Service Quotas" in the search bar in Dashboard.



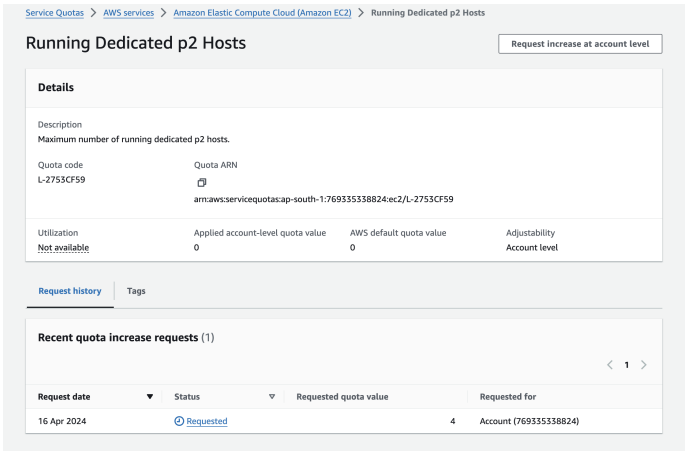
- On clicking on "Service Quota", you will be navigated to Service Quota Dashboard, where you have to select "Amazon Elastic Compute Cloud (Amazon EC2)" option.

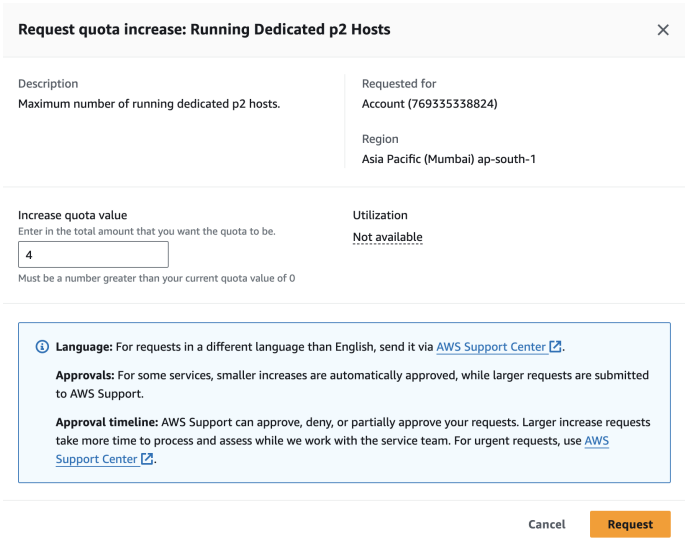


- You will be navigated to a search bar where you have to enter "running dedicated x hosts" where x is the instance that you want to be modified.



- For the sake of increasing the number of vCPUs to keep the computational powers of GPU instance same as the non-GPU instance (c5.4xlarge), we are increasing the number of vCPUs to 4 in p2.xlarge instance.
- Once you are at the dedicated x instance request page, you need to send a new request for quota increase if you want to modify the vCPU count. Click on the "Request Quota Increase" and fill the form as required.





Note: These requests are to be approved by AWS Support, so it may take some time to be approved, so send the request before.

7.2 Launching GPU Instance

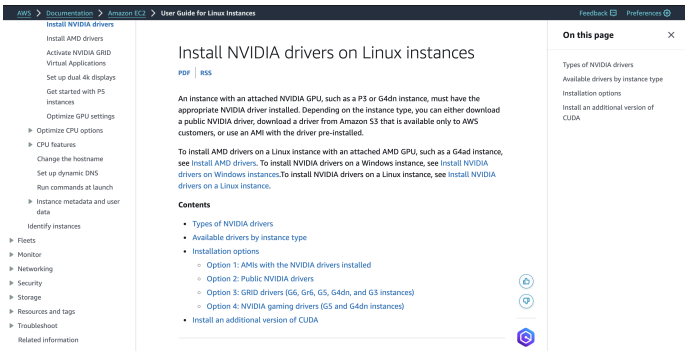
Please refer to Section 1

7.3 Installing NVIDIA Drivers on Instance

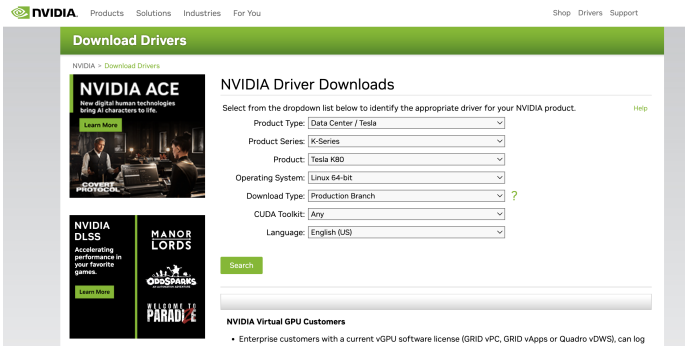
- First update the current packages in the instance by the command:

```
sudo apt -y update
```

- Follow this link to see what NVIDIA Driver is suitable for the instance that you have launched.



- Follow this link to install the corresponding NVIDIA Driver by filling the requirements that your instance satisfies.



- Copy the "Version" obtained after searching, and enter that version using the following command:

```
sudo apt install nvidia-driver-<copied_version> \
nvidia-dkms-<copied_version>
```

- To test our installation, restart the instance for the installation to take effect, and then enter the command:

```
nvidia-smi
```

  You will get the details of the graphic card displayed.

8 AWS Management

- Starting and stopping an Instance:
  - Navigate to amazon AWS console and select EC2. There you can see The number of instances created, number of running instances etc. Click on the instances button.
  - Now select the instance and you want to manipulate and click on instance Actions on the top, and then choose start or stop instance accordingly.
- **NOTE: NEVER TERMINATE THE WORKING INSTANCE PA!** If you did, you have create a fresh instance again and do the environment setup.
- When you connect to the Instance by default you will be in /home/ubuntu/
- If you are making any changes in the files uploaded, you need to reload the daemon and restart the gunicorn and nginx services.