CODESTUDIO

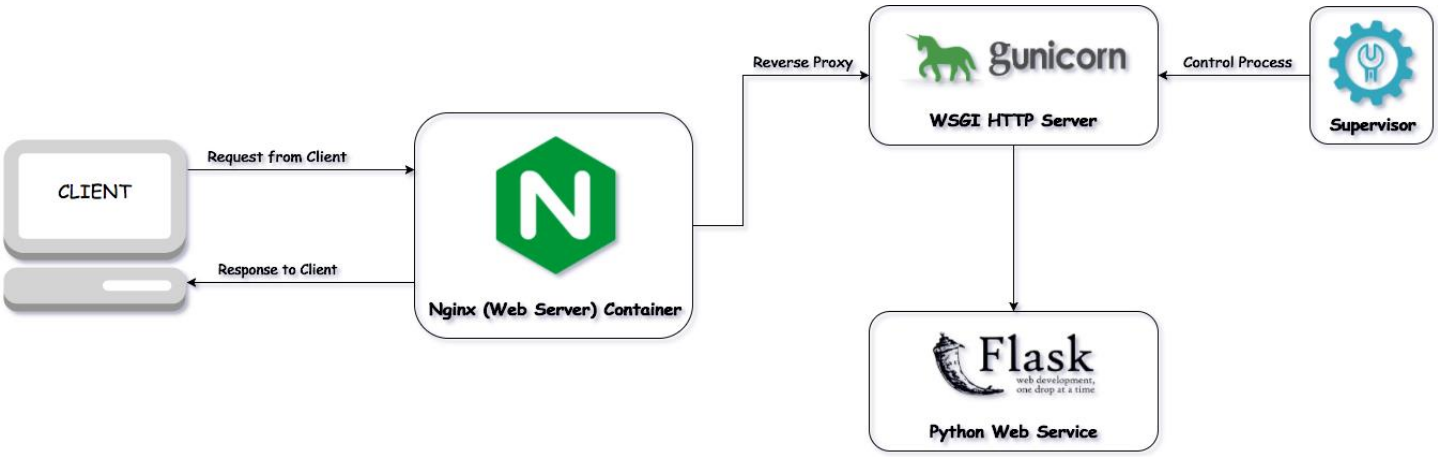
Model Deployment Tutorial 5 - ML Model in Microsoft Azure Cloud (IAAS)

ML Deployment in Azure with NGINX, Guinicorn, Supervisor

Architecture of Deployment

We will be using the following technologies:

- ➔ Nginx: Reverse proxy, web server
- ➔ Flask: Server backend
- ➔ Gunicorn: To run flask app
- ➔ Supervisor: Monitor and control gunicorn process

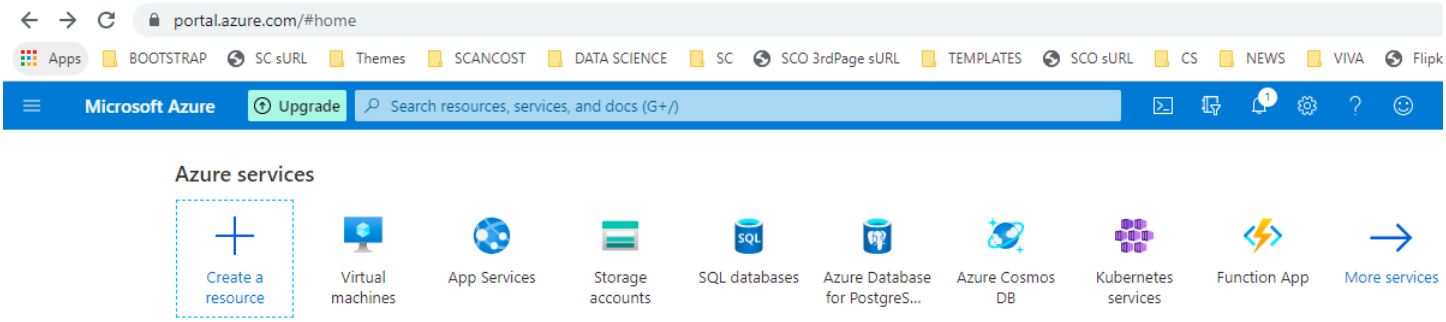


Here is the architecture of this deployment, where client could be web browser or mobile device etc. NGINX as the web server and reverse proxy. This means that NGINX will sit between your Flask application and external clients and forward all client requests to your running Flask application. Gunicorn (Green Unicorn), is a Python web server gateway interface (WSGI) HTTP Server for UNIX. It will be used to forward requests from your NGINX web server to your Flask application and finally Supervisor is a client/server system that allows its users to monitor and control a number of processes on UNIX-like operating systems. Supervisor can handle auto-reloading Gunicorn if it crashes or if your server is rebooted unexpectedly

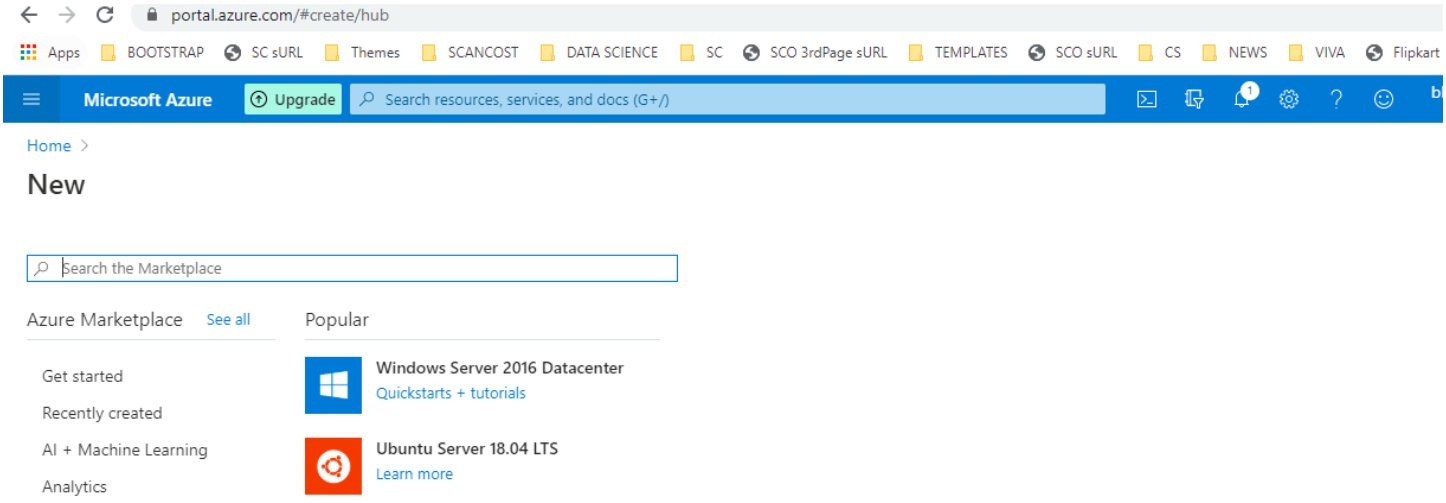
Starting up an instance in Microsoft Azure

1. Login to the azure console here https://portal.azure.com/#home

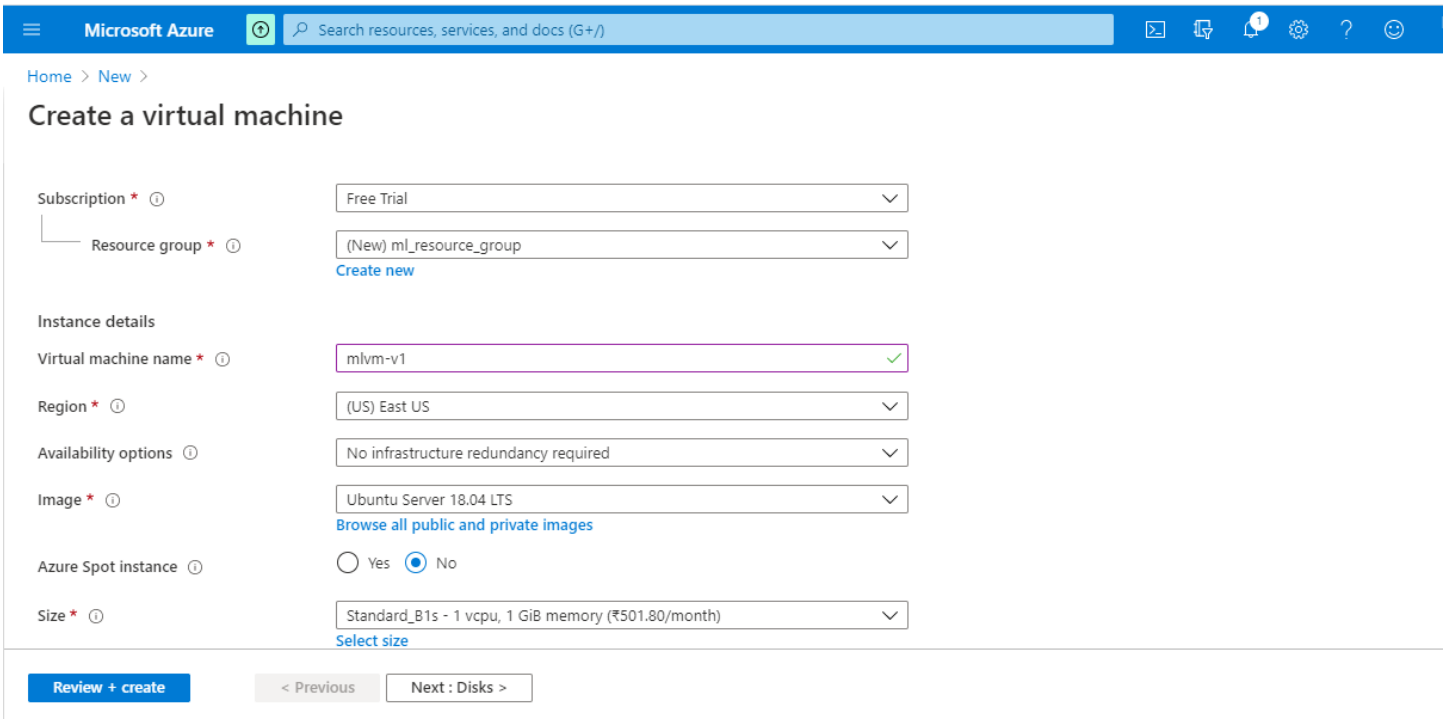
➔ Create virtual machine by clicking on create a resource

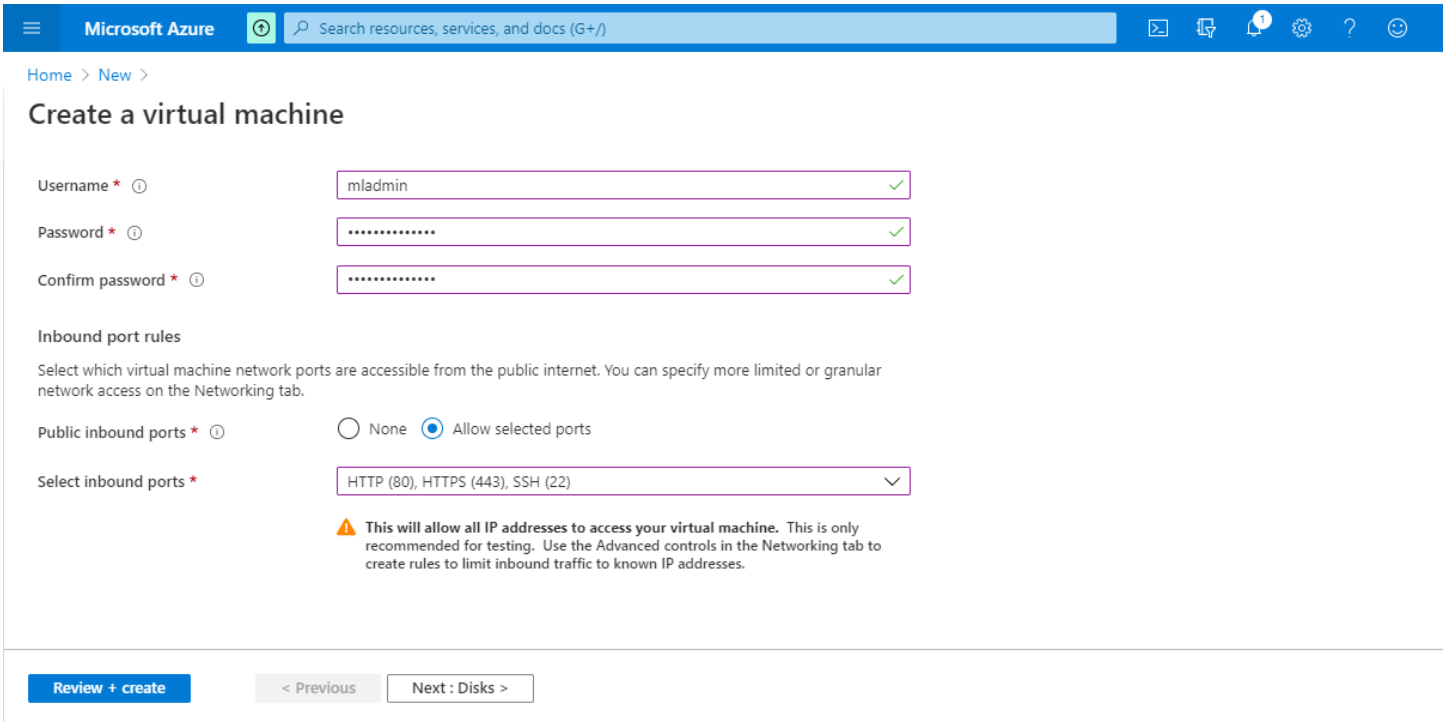


➔ Select the Ubuntu server



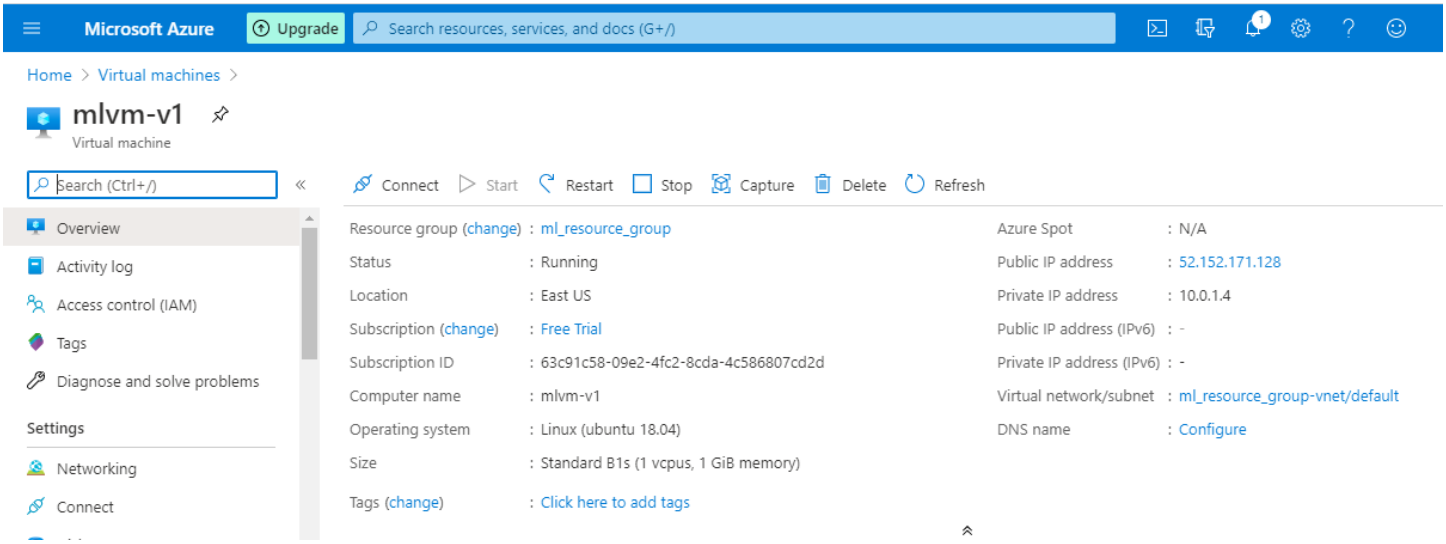
➔ Enter required fields





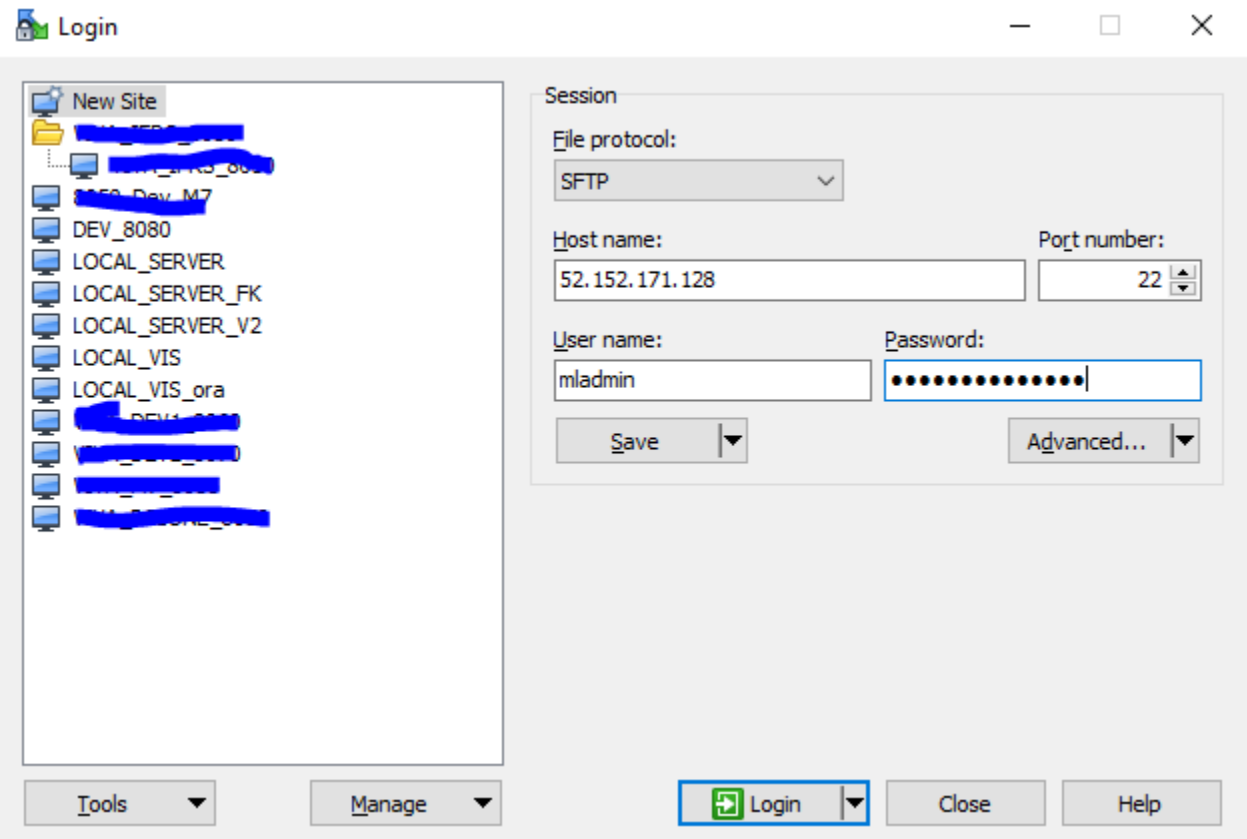
Please note down user name and password which to be used to connect server from third party tool such as putty, WinScp.

➔ Click on next button to fill other required fields. And finally review and create the virtual machine



➔ Note down the instance public IP, public DNS name and User
Public IP: 52.152.171.128
User: mladmin

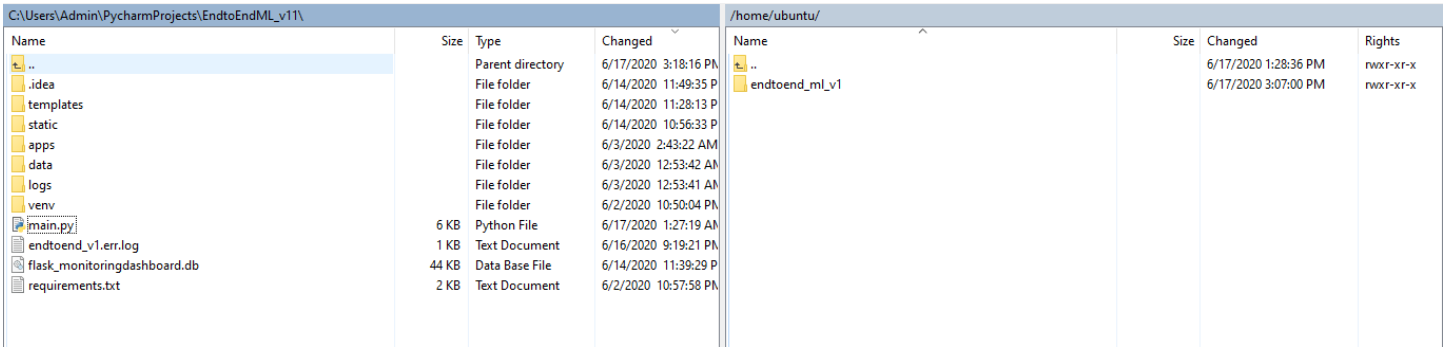
2. Connect WINSCP to upload code files



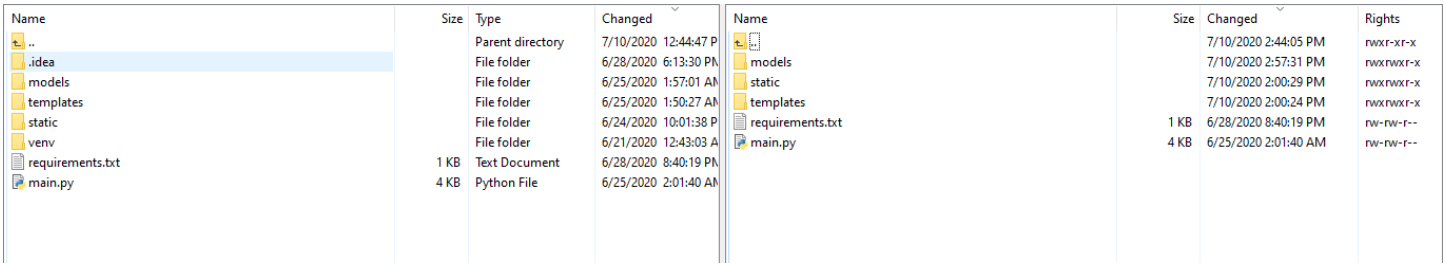
➔ Finally click on Login Button to connect server

Please note that home directory may be different for your deployment.

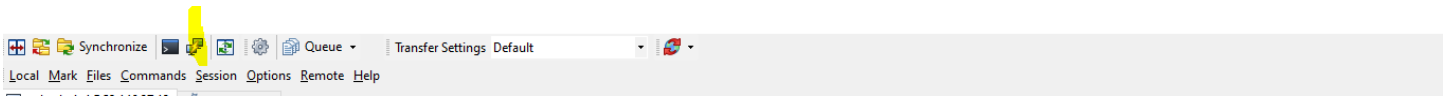
➔ Create new folder in home directory to upload all code files



➔ Upload required code files into server



3. Connect putty to install webserver called NGINX, Flask Web framework and all the required libraries



4. Run below command to upgrade pip install utility in Ubuntu server before installing all required libraries

➔ In your home directory, install Python 3:

```
sudo apt install python3
```

➔ Install pip3, the standard package manager for Python

```
sudo apt-get update && sudo apt-get install python3-pip
```

5. Install nginx web server by running below command

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

Above will install nginx as well as run it.

Check status of nginx using
sudo service nginx status
Here are the commands to start/stop/restart nginx
sudo service nginx start
sudo service nginx stop
sudo service nginx restart

➔ Check the website by hitting public ip or with public DNS. It displays default page of nginx sever
http://52.152.171.128/

➔ remove the default page by deleting the default file to redirect our custom index.html page

```
sudo rm /etc/nginx/sites-enabled/default
```

➔ create a new config file in the sites-available folder and create a symbolic link to it in the sites-enabled folder.

```
sudo vim /etc/nginx/sites-available/endtoend_v1.conf

server {
   listen 80;
   server_name 52.152.171.128;

   root /home/ubuntu/endtoend_v1;
```

```
    access_log /home/ubuntu/endtoend_v1/logs/nginx_log/access.log;
    error_log /home/ubuntu/endtoend_v1/logs/nginx_log/error.log;

    location / {
        proxy_set_header X-Forward-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;
        if (!-f $request_filename) {
            proxy_pass http://127.0.0.1:8000;
            break;
        }
    }

    location /static {
        alias  /home/ubuntu/endtoend_v1/static/;
        autoindex on;
    }
}
:wq to save and exit
```

➔ We have to create the directory for our nginx logs

```
mkdir -p ~endtoend_v1/logs/nginx_log
```

➔ Create symlink for this file in /etc/nginx/sites-enabled by running this command,

```
sudo ln -s /etc/nginx/sites-available/endtoend_v1.conf /etc/nginx/sites-enabled/endtoend_v1.conf
```

➔ Restart nginx server

```
sudo service nginx restart
```

6. Create virtual environment

```
sudo apt install virtualenv
```

➔ create a virtual environment and activate

```
cd /home/ubuntu/endtoend_v1
mkdir /home/ubuntu/endtoend_v1/.virtualenvs && cd /home/ubuntu/endtoend_v1/.virtualenvs
virtualenv -p python3 endtoend_v1_venv
```

➔ Activate the virtual env

```
source /home/ubuntu/endtoend_v1/.virtualenvs/endtoend_v1_venv/bin/activate
```

7. Install dependencies using requirement.txt. run below command

```
pip3 install -r /home/ubuntu/endtoend_v1/requirements.txt
```

8. Install Gunicorn. It act as python WSGI HTTP server for Unix

   pip3 install gunicorn

➔ Let's start a Gunicorn process to serve your Flask app.

   cd /home/ubuntu/endtoend_v1
   gunicorn main:app -w 3

   This will set your Gunicorn process off running in the background, which will work fine for your purposes here. An improvement that can made here is to run Gunicorn via Supervisor.

9. Install supervisor lib which Supervisor allows to monitor and control a number of processes on UNIX-like operating systems. Supervisor will look after the Gunicorn process and make sure that they are restarted if anything goes wrong, or to ensure the processes are started at boot time.

   sudo apt install supervisor

➔ Create a supervisor file in /etc/supervisor/conf.d/ and configure it according to your requirements.

   sudo vim /etc/supervisor/conf.d/endtoend_v1.conf

   [program:endtoend_v1]
   directory=/home/ubuntu/endtoend_v1/
   command=/home/ubuntu/endtoend_v1/.virtualenvs/endtoend_v1_venv/bin/gunicorn main:app
   autostart=true
   autorestart=true
   stopasgroup=true
   killasgroup=true
   stderr_logfile=/home/ubuntu/endtoend_v1/logs/supervisor_log/endtoend_v1.err.log
   stdout_logfile=/home/ubuntu/endtoend_v1/logs/supervisor_log/endtoend_v1.out.log

➔ Create the log directories and files listed in the endtoend_v1.conf file. Make sure to replace endtoend_v1 if it was modified in the Supervisor script above:

   sudo mkdir /home/ubuntu/endtoend_v1/logs/supervisor_log
   sudo touch /home/ubuntu/endtoend_v1/logs/supervisor_log/endtoend_v1.err.log
   sudo touch /home/ubuntu/endtoend_v1/logs/supervisor_log/endtoend_v1.out.log

➔ To enable the configuration, run the following commands:

   sudo supervisorctl reread
   sudo supervisorctl update
   sudo supervisorctl reload

   additional command
   sudo service supervisor restart

sudo service supervisor stop

➔ This should start a new process. To check the status of all monitored apps, use the following command:

sudo supervisorctl status

10. run URL into browser to see our custom index.html page