NGINX, uWSGI, and Flask Installation Guide

Update Ubuntu

Uninstall `Anaconda` if you have otherwise it will not work

You need to keep your system up-to-date

```
sudo apt update
```

```
sudo apt install python3-pip python3-dev build-essential libssl-dev libffi-dev python3-setuptools
```

```
1
```

Install Python Virtual Environment

Install python virtual environment in order to isolate our new installation from previous working modules. This has to be done, incase anything breaks.

```
sudo apt install python3-venv
```

Make a parent directory for our project

Here are few guidelines which we will follow

```
mkdir ~/bertmodel
```

```
cd ~/bertmodel
```

- project directory: `bertmodel`
- virtual environment: `bertenv`
- username: `ubuntu`
- server_ip: `EC2 Instance Public IP`
- default http port: `80`
- flask port: `5000`

Create a Virtual Environment

```
python -m venv bertenv
```

this will install local copy of Python packages into virtual environment

Now you need to activate virtual environment

```
source bertenv/bin/activate
```

Now you should see in terminal like this

```
(bertenv)ubuntu@host:~/bertmodel$
```

Congrats! Virtual Environment is activated

Setting up Flask Application

Now we need to install necessary python packages inside `bertenv`

```
pip install wheel
```

```
pip install uwsgi flask
```

```
sudo ufw allow 5000
```

Now test your flask application

Setting up `uWSGI`

We need to create a `wsgi.py` file inside the `bertmodel` project dir

```
nano ~/bertmodel/wsgi.py
```

Inside `wsgi.py` file add below code

```
from bertmodel import app

if __name__ == "__main__":
    app.run()
```

Testing `uWSGI`

specify the socket, so that it will be started on a publicly available interface, as well as the protocol, so that it will use `HTTP` instead of the uwsgi binary protocol. We'll use the same port number, `5000` , that we opened earlier

```
uwsgi --socket 0.0.0.0:5000 --protocol=http -w wsgi:app
```

now test server by visiting pubplic ip address

```
http://public_ip:5000
```

Now deactivate the virtual environment

```
deactivate
```

```
1
```

Configuring `uWSGI`

We need to setup a robust system for deployment

```
nano ~/bertmodel/bertmodel.ini
```

we will start off with the `[uwsgi]` header so that `uWSGI` knows to apply the settings. We'll specify two things:

- the module itself, by referring to the `wsgi.py` file minus the extension,
- and the callable within the file, `app`:

```
[uwsgi]
module = wsgi:app

master = true
processes = 5

socket = bertmodel.sock
chmod-socket = 660
vacuum = true

die-on-term = true
```

The last thing we'll do is set the `die-on-term` option. This can help ensure that the init system and `uWSGI` have the same assumptions about what each process signal means.

You may have noticed that we did not specify a protocol like we did from the command line. That is because by default, `uWSGI` speaks using the `uwsgi` protocol, a fast binary protocol designed to communicate with other servers. `NGINX` can speak this protocol natively, so it's better to use this than to force communication by `HTTP` .

Creating `systemd` Unit File

Creating a systemd unit file will allow Ubuntu's init system to automatically start uWSGI and serve the Flask application whenever the server boots.

```
sudo nano /etc/systemd/system/bertmodel.service
```

Write this below code

```
[Unit]
Description=uWSGI instance to serve bertmodel
After=network.target

[Service]
User=ubuntu
Group=www-data
WorkingDirectory=/home/ubuntu/bertmodel
Environment="PATH=/home/ubuntu/bertmodel/bertenv/bin"
ExecStart=/home/ubuntu/bertmodel/bertenv/bin/uwsgi --ini bertmodel.ini

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

Start `uWSGI` services

We can now start the uWSGI service we created and enable it so that it starts at boot:

```
sudo systemctl start bertmodel
```

```
sudo systemctl enable bertmodel
```

```
sudo systemctl status bertmodel
```

```

```

Installing NGINX

```
sudo apt update
```

```
sudo apt install nginx
```

Nginx registers itself as a service with `ufw` upon installation, making it straightforward to allow `Nginx` access.

```
sudo ufw app list
```

We should allow only `HTTPS` but for time being we will use only `HTTP` . We will need additional `SSL` certificate to encrypt connection of `https` .

```
sudo ufw allow 'Nginx HTTP'
sudo ufw status
```

```
systemctl status nginx
```

Check your `NGINX` is working?

```
http://public_ip
```

```

```

Managing `NGINX` process

```
sudo systemctl stop nginx
```

```
sudo systemctl start nginx
```

```
sudo systemctl restart nginx
```

```
sudo systemctl reload nginx
```

```

```

Configuring `NGINX`

```
sudo nano /etc/nginx/sites-available/bertmodel
```

```
server {
    listen 80;
    server_name bertmodel public_ip;

    location / {
        include uwsgi_params;
        uwsgi_pass unix:/home/ubuntu/bertmodel/bertmodel.sock;
    }
}
```

To enable the Nginx server block configuration you've just created, link the file to the `sites-enabled` directory:

```
sudo ln -s /etc/nginx/sites-available/bertmodel /etc/nginx/sites-enabled
```

```
sudo nginx -t
```

`sudo systemctl restart nginx`

`sudo ufw delete allow 5000`

`sudo ufw allow 'Nginx Full'`

`http://public-ip`

```
```

Errors checking

`sudo less /var/log/nginx/error.log`

`sudo less /var/log/nginx/access.log`

`sudo journalctl -u nginx`

`sudo journalctl -u bertmodel`