Deploying a second API alongside the First API on the Ubuntu Server

This guide explains the process of deploying a second Flask API or App alongside an already running API on the Ubuntu Server.

Step 1: Connect to the remote server via SSH

Open Windows Terminal and enter the following command

```
$ ssh <username>@<IP_address>
```

Replace with your Ubuntu server username and <IP_address> with the server's IP Address.
For Example:

```
$ ssh ubuntu@172.15.3.94
```

Step 2: Update the Ubuntu System

Enter the following command:

```
$ sudo apt update && sudo apt upgrade
```

Step 3: Set Up the Flask API
In this guide we will follow the following directory structure:

```
project_directory/
├── app.py (first app)
├── app2.py (second app)
├── venv/ (virtual environment for both apps)
└── ...
```

Copy your second app in the same directory as your first one, so finally in our case, the directory structure will be like this:

```
python-apis/
├── app.py (first app)
├── app2.py (second app)
├── venv/ (virtual environment for both apps)
└── ...
```

Step 4: Activate the pre-existing Virtual Environment

```
source venv/bin/activate
```

Also, make sure the dependencies are installed in this step. Since our second app also uses the same dependencies, we did not need to install any more dependencies here, but you may have to.

Step 5: Run the Second App with Gunicorn

Run your second Flask app using Gunicorn on a designated port. In our second app we have designated the port 8988. Make sure to run it with the gunicorn from venv.

```
$  ./venv/bin/gunicorn -w 3 --bind 0.0.0.0:8988 app2:app
```

Replace `app2` and `app` with appropriate names as explained in the guide to deploy your Flask API on an Ubuntu server.

Step 5: Configure Nginx for the Second App

Create an Nginx server block configuration file for your second app:

```
$ sudo nano /etc/nginx/sites-available/app2
```

The file should be modified so:

```
server {
    listen 80;
    server_name 172.__.__.__;

    location / {
```

```
        include proxy_params;
        proxy_pass http://172.__.__.__:8988;
    }
}
```

Create a symbolic link to enable the configuration:

```
$ sudo ln -s /etc/nginx/sites-available/app2 /etc/nginx/sites-enabled
```

Test the Nginx configuration:

```
$ sudo nginx -t
```

Restart Nginx to apply the changes:

```
$ sudo systemctl restart nginx
```

Step 6: Configure Gunicorn systemd service

Add the following content:

```
[Unit]
Description=Gunicorn instance to serve your first Flask app
After=network.target

[Service]
User=<username>
Group=www-data
WorkingDirectory=/home/<username>/python-apis/
ExecStart=/home/ubuntutest/python-apis/venv/bin/gunicorn --workers 3 --bind
0.0.0.0:8988 app2:app

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

Make sure to replace your username and enter the correct port number without which the app will not run.

After creating these separate systemd service unit files, remember to start and enable each service:

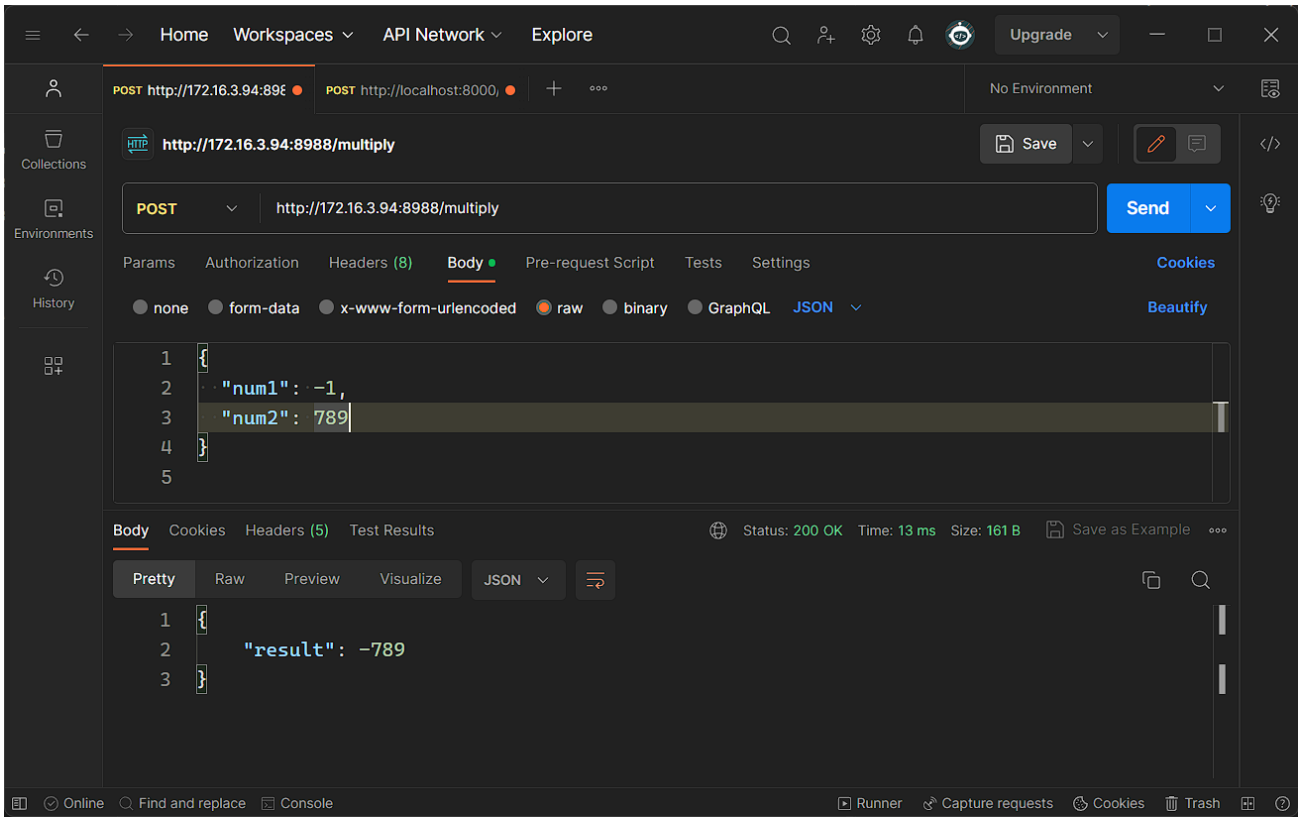
```
$ sudo systemctl daemon-reload
$ sudo systemctl start yourapp1
$ sudo systemctl enable yourapp1
$ sudo systemctl start yourapp2
$ sudo systemctl enable yourapp2
```

Make sure the services are up and running by using the following commands:

```
$ sudo systemctl status app1
$ sudo systemctl status app2
```

Step 7: Access the API via Postman

Send an appropriate request via Postman to your API and make sure it returns the appropriate response to confirm that it works.
Example:



To send the query as JSON, Click on `Body -> raw -> Text` and change it to JSON. Send your query.

For Example, for our API:

```
{
  "num1": -1,
  "num2": 789
}
```

Make sure an appropriate response is received as shown in screenshot.

Conclusion

Congratulations! You have deployed two apps on two ports alongside each other on the same Ubuntu Server!