Mini-Project: Deploying Python Application With Nginx

Husni B. · Follow
Published in The Startup
5 min read · Aug 3, 2020

Listen Share More

At this moment, I will present guidance on how to deploying a “Hello World” Python application with Nginx. Is it sounds strange to you? Likewise with me before trying this one. Before heading to the core of the tutorial, I should provide some prerequisites to do it.

Python

Python is an interpreted, high-level, general-purpose programming language. I'll use Python 3.8.2 which is by default installed on my ubuntu 20.04 or Focal Fossa version. You can check the version of your Python 3 by this line.

```
# python3 -V
Python 3.8.2
```

If you have no Python3 installed on it, you can search by yourself on how to do it. Don't slothful!

PIP

Pip is the package installer for Python. You can use pip to install packages from the Python Package Index and other indexes. By default, Pip isn't installed yet but it's easy to have it by entering this line.

```
# apt install python3-pip
```

And you'll get the latest and stable Pip package that using Python 3 as an interpreter. Check the version to make sure that it is absolutely installed on your machine.

```
# pip3 -V
pip 20.0.2 from /usr/lib/python3/dist-packages/pip (python 3.8)
```

Nginx

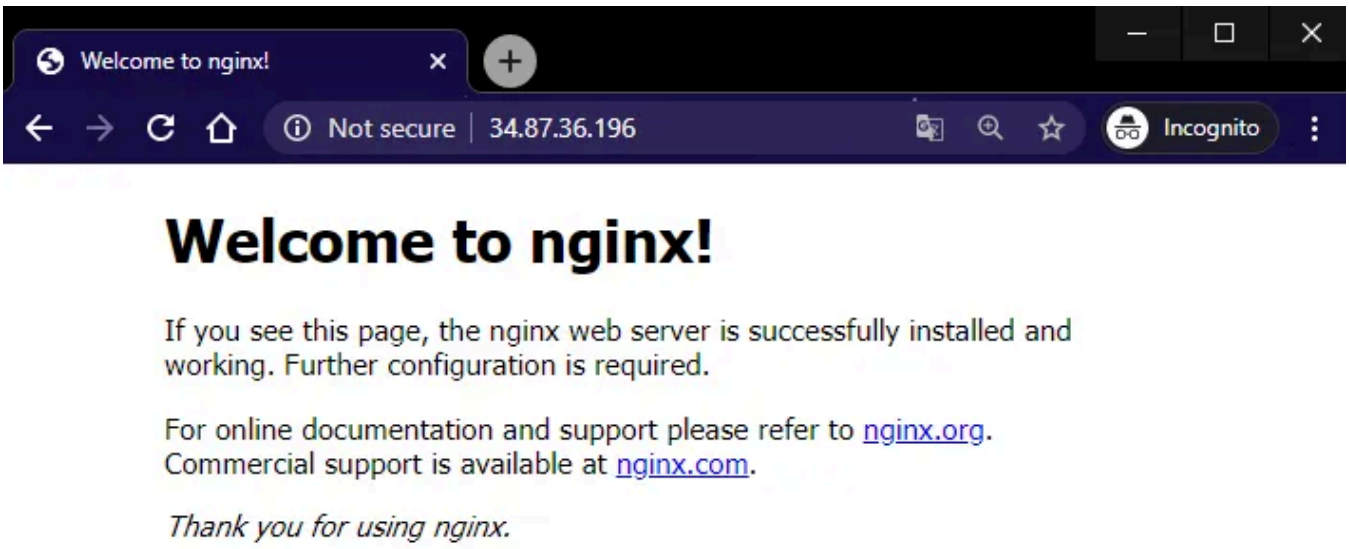
Nginx is a web server that can also be used as a reverse proxy, load balancer, mail proxy, and HTTP cache. It's easy to install that only using this clean line.

```
# apt install nginx
```

Check the version of it just to make sure...

```
# nginx -v
nginx version: nginx/1.18.0 (Ubuntu)
```

To make sure again for you that have a trust issue, you can check on your web by submitting your VM's IP. By default, Nginx running on port 80, but you can change to another port if you have another service that runs on port 80.



Step 1 — Setting Up Virtual Environment

We'll use one of python's features which is the virtual environment that could isolate our application from the other Python files on the system.

- Make a folder for the application

```
# mkdir python-app
# cd python-app
```

- Install python virtual environment package

```
# pip3 install virtualenv
```

- Create a virtual environment named py-env

```
# virtualenv py-env
```

- Activate the virtual environment

```
# source py-env/bin/activate
```

- Install python web framework and server gateway interface server

```
# pip3 install gunicorn flask
```

Step 2— Create the Application

- Create a Flask web application

```
# nano app.py
```

Now Flask is installed and we can create a simple flask application. Create a file named `app.py` .

```
from flask import Flask

app = Flask(__name__)
@app.route('/')
def hello_world():
 return return "<center>Hello World! Uooohh Mantaapp</center>"

if __name__ == '__main__':
 app.run(debug=True,host='0.0.0.0')
```

- Create the WSGI Entry Point

```
# nano wsgi.py
```

Create a file that will serve as the entry point for our application. This will tell the uWSGI server how to interact with it. Create a file named `wsgi.py` .

```
from app import app

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

Step 3 — Test the Application

- Test the Flask application

```
# python3 app.py
```

Then, you can visit via a web browser using your VM's IP with port 5000 which is the Flask default port.

- Testing Gunicorn's ability

```
# gunicorn --bind 0.0.0.0:5000 wsgi:app
```

By default, Gunicorn listening at localhost with port 8000. If your VM running on public IP, you have to manually bind it.

- Deactivate the environment

Make sure the folder structure just like this.

```
python-app
  |____ app.py
  |____ wsgi.py
  |____ py-env
```

Once done with the application. Deactivate it.

```
# deactivate
```

Step 4— Make the Application Run as a Service

Systemd unit file will allow Ubuntu's init system to automatically start Gunicorn and serve our Flask application whenever the server boots.

- Create a unit

```
# nano /etc/systemd/system/py-app.service
```

Create a unit file named **py-app.service** within the `/etc/systemd/system` directory. If you want to have a different name for the service, go ahead.

```
After=network.target

[Service]
User=root
Group=www-data

WorkingDirectory=/root/python-app
Environment="PATH=/root/python-app/py-env/bin"
ExecStart=/root/python-app/py-env/bin/gunicorn wsgi:app

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

- Start the Gunicorn service

```
# systemctl start py-app
```

Step 5— Configure Nginx

- Create a configuration file

Configure Nginx to pass web requests to that socket by making some small additions to its configuration file.

```
# nano /etc/nginx/sites-available/py-app
```

Create a configuration file named **py-app** within the `/etc/nginx/sites-available` directory. Open up a server block and tell Nginx to listen on the default port 80.

Add a location block that matches every request. Within this block, we'll include the **proxy_params** file that specifies some general proxying parameters that need to be set. We'll then pass the requests to the socket we defined using the **proxy_pass** directive.

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

location / {
 include proxy_params;
 proxy_pass http://127.0.0.1:8000;
 }
}
```

- Enable the configuration

To enable the Nginx server block configuration we've just created, link the file to the sites-enabled directory.

```
# ln -s /etc/nginx/sites-available/py-app /etc/nginx/sites-enabled
```

- Test the configuration syntax

Test for in case there are syntax errors.

```
# nginx -t
```

- Restart the service

If the test does not indicate any issues, we can restart the Nginx process to read the new config.

```
# systemctl restart nginx
```

- Check the service

You should now be able to go to your VM's IP in your web browser and see your app running.

References

DigitalOcean

How To Serve Flask Applications with uWSGI and Nginx on Ubuntu | DigitalOcean

In this guide, we will be setting up a simple Python application using the Flask micro-framework on Ubuntu 18.04. The...

www.digitalocean.com

Medium

Deploy flask app with Nginx using Gunicorn

Today we're going to deploy a micro flask app with Nginx using gunicorn.

medium.comm

Python Programming Web Development Nginx Deployment

Follow

Written by Husni B.

18 Followers · Writer for The Startup

Bad at writing stuff