Shah Nawaz Shuvo
Posted on 8 Sept 2020 • Originally published at blog.nawaz.info

Deploy FastAPI Application on Ubuntu with Nginx, Gunicorn and Uvicorn

#python #fastapi #nginx #deployment

FastAPI is a promising new Python framework that supports concurrency and type system out of the box. It has many cool features that I like and it's fast. In this post, I will briefly go over the process of deploying a simple FastAPI application on Ubuntu running on an EC2 instance. This post will assume that you know and already configured quite a few things and will only focus on deploying the actual application.

Pre-requisites

1. First of all, you will need a running EC2 instance with Ubuntu 18.04 or later. This will also work for VPS servers like DigitalOcean or Linode.
2. If you are using EC2 instance, make sure you configured the VPC and security groups properly and your **port 80** is open for inbound and outbound traffic.
3. You have to have Nginx installed and running.
4. You will need Python 3.7 or above installed and any Python virtual environment tool like **virtualenv**, **pipenv**, **conda**, etc. installed.
5. You need to have your FastAPI application on Github/Bitbucket/Gitlab for easy deployment.

After you have ensured that all of the above requirements are met you can move on to the next step. The actual deployment.

Prepare the Application

Now, SSH into the server, create and navigate to the directory that you want your application to be stored into. Say it's `/var/www/myapp`.

```
$ mkdir /var/www/myapp
$ cd /var/www/myapp
```

We have to make sure that our user has proper read-write access to this directory. In this post, I will use `virtualenv` to manage a Python virtual environment. You can use whichever you like or prefer. Inside `/var/www/myapp` we have to do the following one by one:

```
$ virtualenv -p python3.8 venv
$ mkdir src
$ . venv/bin/activate
(venv) $ cd src
```

Once inside the src directory, we have to pull our application source code there. Our application source code should contain a `requirements.txt` file with all the dependencies unless we are using a `PIPFILE`.

```
(venv) $ git init
(venv) $ git remote add origin <your-repo-url>
(venv) $ git pull origin <your-branch-name>
(venv) $ pip install -r requirements.txt
(venv) $ pip install gunicorn uvicorn
```

Configure Nginx

Now our application is ready to be run and tested. To be able to serve the application over HTTP we have to make an Nginx config for our application.

```
(venv) $ sudo vim /etc/nginx/sites-available/myapp
```

Put the followings on that file:

```
server{
       server_name <your-site-name>;
       location / {
           include proxy_params;
           proxy_pass http://127.0.0.1:8000;
       }
}
```

Now we save the file and exit. Then we make a symbolic link to this config file in the `/etc/nginx/sites-enabled` directory.

```
› ln -s /etc/nginx/sites-available/myapp /etc/nginx/sites-enabled/
```

Then we restart the Nginx service.

```
(venv) $ sudo systemctl restart nginx.service
```

Now we can start our uvicorn server to check if our application is working or not.

```
(venv) $ gunicorn -w 4 -k uvicorn.workers.UvicornWorker main:app
```

In the place of `main:app` we can use whatever is correct for our application. Now that our application is running and the proxy server is configured properly we should be able to visit the URL and see our application from a browser.

Configure ASGI Server

Now that our application is deployed and configured properly one last thing to do is to create a service for the Gunicorn server so that it is always running and it automatically starts when the server is rebooted. We will user `systemd` to create the service.

```
(venv) $ deactivate
$ sudo vim /etc/systemd/system/myapp.service
```

In this new file we have to put the following:

```
[Unit]
Description=Gunicorn instance to serve MyApp
After=network.target

[Service]
User=<username>
Group=www-data
WorkingDirectory=/var/www/myapp/src
Environment="PATH=/var/www/myapp/venv/bin"
ExecStart=/var/www/myapp/venv/bin/gunicorn -w 4 -k uvicorn.workers

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

Modify the locations as of your own setup and then save and exit the file.

```
$ sudo systemctl start myapp.service
```

This will start our new service and our ASGI server will be running in the background.

Read next

How to Build a Predictive Machine Learning Site With React and Python (Part Three: Frontend Development)
Gospel Darlington - Oct 4

Django REST Framework : #9 Category API Routing And CRUD Operations
Abhishek Ezhava - Oct 4

IndexError: too many indices for array
Srinivas - Sep 12

Rock, Paper, Scissors game in Python
JRRyan606 - Oct 4

Shah Nawaz Shuvo
Follow
A Full Stack Developer and Tech Enthusiast from Bangladesh who wants to work and collaborate in the global ecosystem of technology and innovation. I mostly write about things I am learning myself.

LOCATION
Dhaka, Bangladesh

EDUCATION
BSc in Computer Science & Engineering

WORK
Software Engineer at PIPELINE Security

JOINED
26 Aug 2018

More from Shah Nawaz Shuvo

Deploy Laravel Application with Nginx on Ubuntu 18.04 on DigitalOcean
#laravel #nginx #ubuntu #devops

Laravel Horizon with Nginx and Ubuntu 18.04 on DigitalOcean
#php #laravel #nginx #digitalocean