How to Deploy Flask Application with Nginx and Gunicorn on Ubuntu 20.04

February 23, 2022 by Jeff Wilson

Flask is a small, lightweight, and micro web framework written in Python. It allows you to develop web applications easily without any tools or libraries. This web application may be a blog, wiki page, web pages, web-based calendar application, or a commercial website. It is simple, easy to use, easy to learn, and beginner-friendly because it does not require any dependencies.

In this tutorial, we will show you how to deploy the Flask application with Gunicorn and Nginx on Ubuntu 20.04.

Table of Contents

Prerequisites

- A Ubuntu 20.04 VPS with root access enabled, or a user with sudo privileges.

Log in and Update Packages

First, we're going to need to log into our server using SSH. You can do that by entering this command:

```
ssh root@IP_Address -p Port_Number
```

Remember to replace `root` with your username if you are not using the root user. Change `IP_Address` and `Port_Number` according to your server's IP address and SSH port number.

Once you are logged in, you should update all of your packages to their latest available versions.

```
apt-get update -y
apt-get upgrade -y
```

Once all the packages are up-to-date, restart your server to apply the configuration changes.

Install Required Dependencies

Flask is a python-based application. So Python and other required dependencies must be installed on your server. If not installed you can install all of them with the following command:

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

Once all the dependencies are installed, install the Python virtual environment package using the following command:

```
apt-get install python3-venv -y
```

Once installed, you can proceed to the next step.

Install Nginx Web Server

In this tutorial, we will use Nginx as a reverse proxy for the Flask application. So you will need to install the Nginx web server package to your server. You can install it using the following command:

```
apt-get install nginx -y
```

Once the Nginx is installed, start and enable the Nginx service using the following command:

```
systemctl start nginx
systemctl enable nginx
```

Create a Virtual Environment for Flask Application

Next, you will need to create a virtual environment for the Flask application.

First, create a project directory with the following command:

```
mkdir ~/project
```

Next, change the directory to your project and create a Python virtual environment with the following command:

```
cd ~/project
python3 -m venv venv
```

Next, activate your environment with the following command:

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

Next, install Gunicorn, Flask, and other components with the following command:

```
pip install wheel
pip install gunicorn flask
```

Once you are finished, you can proceed to the next step.

Create a Flask Application

Next, you will need to create a sample Flask application for your project. Run the following command to create it inside your project directory:

```
nano ~/project/flaskapp.py
```

Add the following codes:

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return "Welcome to Flask Application!"
if __name__ == "__main__":
    app.run(host='0.0.0.0')
```

Save and close the file then verify your application with the following command:

```
cd ~/project/
python3 flaskapp.py
```

If everything is fine, you should get the following output:

```
 * Serving Flask app 'flaskapp' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production
deployment.
 * Running on http://69.28.84.227:5000/ (Press CTRL+C to quit)
```

Press CTRL+C to close the application.

Create a WSGI Entry Point for Gunicorn

Next, you will need to create a WSGI entry point to serve your application via Gunicorn.

Run the following command to create it:

```
nano ~/project/wsgi.py
```

Add the following lines:

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

Save and close the file then verify whether Gunicorn can serve the application correctly using the command below:

```
cd ~/project/
gunicorn --bind 0.0.0.0:5000 wsgi:app
```

If everything is fine, you should get the following output:

```
[2021-12-23 10:37:15 +0000] [9352] [INFO] Starting gunicorn 20.1.0
[2021-12-23 10:37:15 +0000] [9352] [INFO] Listening at: http://0.0.0.0:5000
(9352)
[2021-12-23 10:37:15 +0000] [9352] [INFO] Using worker: sync
[2021-12-23 10:37:15 +0000] [9354] [INFO] Booting worker with pid: 9354
```

Press CTRL+C to stop the application. Next, deactivate from the Python virtual environment with the following command:

```
deactivate
```

Create a Systemd Service File for Flask Application

Next, you will need to create a systemd unit file for the Flask application. You can create it with the following command:

```
nano /etc/systemd/system/flask.service
```

Add the following lines:

```
[Unit]
Description=Gunicorn instance to serve Flask
After=network.target
[Service]
User=root
Group=www-data
WorkingDirectory=/root/project
Environment="PATH=/root/project/venv/bin"
ExecStart=/root/project/venv/bin/gunicorn --bind 0.0.0.0:5000 wsgi:app
[Install]
WantedBy=multi-user.target
```

Save and close the file then set proper ownership and permission to flask project:

```
chown -R root:www-data /root/project
chmod -R 775 /root/project
```

Next, reload the systemd daemon with the following command:

```
systemctl daemon-reload
```

Next, start the flask service and enable it to start at system reboot:

```
systemctl start flask
systemctl enable flask
```

Next, verify the status of the flask with the following command:

```
systemctl status flask
```

Output:

```
● flask.service - Gunicorn instance to serve Flask
     Loaded: loaded (/etc/systemd/system/flask.service; disabled; vendor
preset: enabled)
     Active: active (running) since Thu 2021-12-23 10:38:26 UTC; 8s ago
   Main PID: 9376 (gunicorn)
      Tasks: 2 (limit: 2353)
     Memory: 27.8M
     CGroup: /system.slice/flask.service
             ├─9376 /root/project/venv/bin/python3
/root/project/venv/bin/gunicorn --bind 0.0.0.0:5000 wsgi:app
             └─9393 /root/project/venv/bin/python3
/root/project/venv/bin/gunicorn --bind 0.0.0.0:5000 wsgi:app

Dec 23 10:38:26 ubuntu2004 systemd[1]: Started Gunicorn instance to
serve Flask.
Dec 23 10:38:26 ubuntu2004 gunicorn[9376]: [2021-12-23 10:38:26 +0000]
[9376] [INFO] Starting gunicorn 20.1.0
Dec 23 10:38:26 ubuntu2004 gunicorn[9376]: [2021-12-23 10:38:26 +0000]
[9376] [INFO] Listening at: http://0.0.0.0:5000 (9376)
Dec 23 10:38:26 ubuntu2004 gunicorn[9376]: [2021-12-23 10:38:26 +0000]
[9376] [INFO] Using worker: sync
Dec 23 10:38:26 ubuntu2004 gunicorn[9393]: [2021-12-23 10:38:26 +0000]
[9393] [INFO] Booting worker with pid: 9393
```

Configure Nginx as a Reverse Proxy for Flask Application

Next, you will need to configure Nginx as a reverse proxy to serve the Flask application through port 80. To do so, create an Nginx virtual host configuration file:

```
nano /etc/nginx/conf.d/flask.conf
```

Add the following lines:


```
server {
    listen 80;
    server_name flask.example.com;
    location / {
        include proxy_params;
        proxy_pass  http://127.0.0.1:5000;
    }
}
```

Save and close the file then verify the Nginx for any syntax error:

```
nginx -t
```

You should see the following output:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Finally, restart the Nginx service to apply the changes:

```
systemctl restart nginx
```

Access Flask Application

At this point, your Flask application is installed, configured, and hosted with an Nginx proxy. You can now access it using the URL **http://flask.example.com**. You should see the following page:

Welcome to Flask Application!

Congratulations! you have successfully deployed the Flask application with Gunicorn and Nginx on Ubuntu 20.04 server.

However, if you are one of our Managed Ubuntu Hosting customers, or if you use one of our Managed VPS Hosting plans, you don't have to install the Flask application on your Ubuntu 20.04 VPS – simply ask our admins, sit back, and relax. Our admins will install the Flask application on Ubuntu 20.04 (or any other OS that you have with us) for you immediately.

PS. If you liked this post about how to install the Flask application on an Ubuntu 20.04 VPS, please share it with your friends on the social networks using the buttons below, or simply leave a comment in the comments section. Thanks.

Tutorials, Ubuntu

‹ How to Install WonderCMS with Nginx on Debian 11
› How to Fix the "There Has Been a Critical Error on Your Website" Error in WordPress

2 thoughts on "How to Deploy Flask Application with Nginx and Gunicorn on Ubuntu 20.04"

Owen
May 24, 2022 at 07:29

I've succeeded in following all your step until the end. But the http://flask.example.com is not accessible!

Please, I need guidance

Reply

admin
May 25, 2022 at 02:16

The subdomain is just a sample. You should use your own domain or subdomain and point the DNS A record to your server.

Reply

Leave a Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

To prove you are human please solve the following *

+ two = eight

Yes, add me to your new blog post notifications list

Post Comment