How to Deploy a Flask App on an AWS EC2 Instance

Aug 6, 2017 | Chris Tran | 3 Minute Read

Introduction

In this post, I will be showing you how I deployed a Flask app on an AWS Ubuntu Server 16.04 LTS server using nginx and Gunicorn. Gunicorn is a server that will be running our app and nginx will be used as our reverse-proxy server.

Server Setup

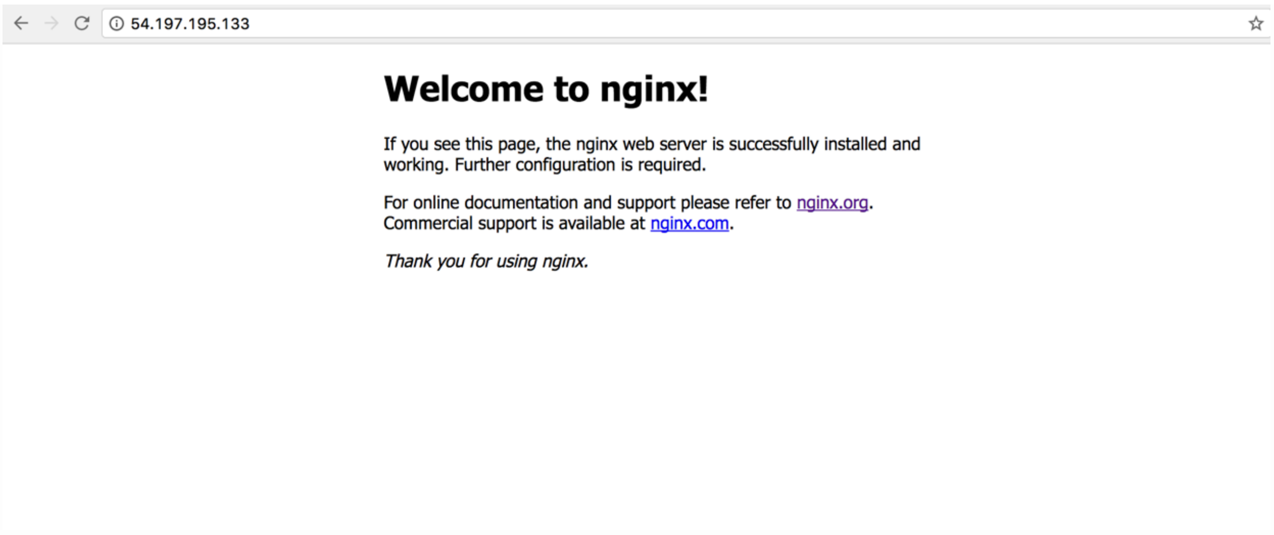
Before we get started, we will need to have an Ubuntu instance running with port 80 opened up to the public. A non-root user will need to be added with `sudo` privileges. I will not be going over how to set this up in this post. You can still continue running everything as the root user, but it is not recommended.

nginx Installation and Setup

First, we will install nginx.

```
$ sudo apt-get update
$ sudo apt-get install nginx
```

Once nginx is installed, you should be able to go to your public IP or public DNS and see the nginx welcome page.



54.197.195.133

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

We will remove the default page by deleting the default file.

```
$ sudo rm /etc/nginx/sites-enabled/default
```

We will create a new config file in the sites-available folder and create a symbolic link to it in the sites-enabled folder.

```
$ sudo vim /etc/nginx/sites-available/example.com
```

This is how the config file will look:

```
server {
        listen 80;

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

This config file will tell the nginx server to listen on port 80 and pass all requests with the '/' prefix to the server `http://127.0.0.1:8000/` We do this because Gunicorn will run your Flask app on port 8000.

Now, we will need to create a symbolic link from the sites-enabled directory that points to the `example.com` config file we created.

```
$ sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-
```

We will need to restart our nginx web server in order for our changes to take into effect.

```
$ sudo service nginx restart
```

Create Simple Flask App

We will install the venv library for Python 3. Python 2 can be used but in this example, we will be using Python 3.

```
$ sudo apt-get install python3-venv
```

We will create a directory for our app and create a virtual environment.

```
#we will be creating the directory in our home directory
$ cd ~
$ sudo mkdir example
$ cd example
$ sudo python -m venv example_env
```

This will create the `example_env` directory with all the files for our virtual environment.

Before we set up our Flask app, we will have to activate our virtual enviornment and install the necessary libraries.

```
$ source example_env/bin/activate
```

One you activate the virtual environment, you should see the name of your virtual environment in parentheses to the left of your shell prompt. Next, we will install the libraries needed to create and run our Flask app.

```
(example_env) $ pip install flask gunicorn
```

Once we have our libraries installed, we can create our Flask app.

```
(example_env) $ sudo vim helloworld.py
```

For our app, we will create a simple hello world example.

```
from flask import Flask


app = Flask(__name__)

@app.route('/')
def hello_world():
        return 'Hello World!'

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

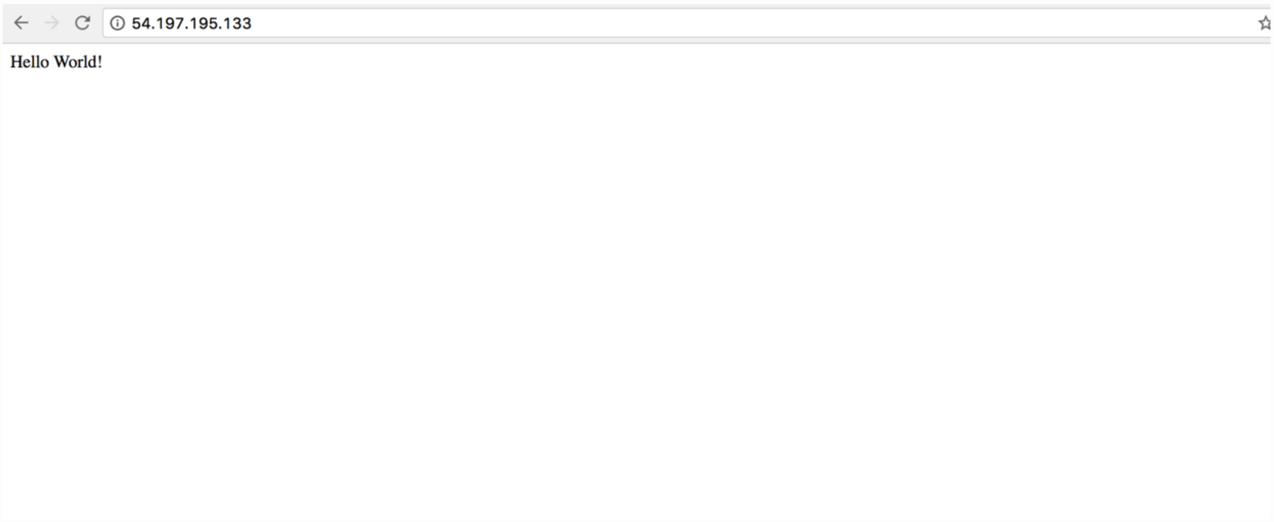
Run the App with Gunicorn

We will now run the Flask app with Gunicorn.

```
(example_env) $ gunicorn helloworld:app
```

Gunicorn should have started and should be listening at `http://127.0.0.1:8000`

If you go back to the public IP or public DNS of your server, you should be able to see that our app is running.



```
Hello World!
```

We now have a running Flask app on our Ubuntu EC2 Instance using nginx and Gunicorn