Syed Muhammad Asad Follow
Oct 11, 2020 · 8 min read · Listen

Deploying Flask App with WSGI and Apache Server on Ubuntu 20.04

Description

Flask app is a popular framework for developing minimal apps or often creating restful APIs. In this article I'm going to discuss about how to deploy a flask app using WSGI and Apache server over Ubuntu 20.04. This article will be helpful to those people who are deploying flask app for the first time and I have also discussed that how to find some of the errors which may occur during deployment and how to tackle them. Recently, I deployed a flask app on AWS and there I were faced some difficulties. So, main purpose of this article is to share all those difficulties which a beginner may face too. In this article I'm skipping the section about how to create a Ubuntu server on online web service like Amazon Web Service etc. but I'll suggest first to google it by following the keywords "How to create a Ubuntu server on AWS" or "How to launch and AWS EC2 server and set up ubuntu on it". Once when you'll have a Ubuntu server follow this article to deploy your app. Here are some further intuitions which are needed to deploy a Flask app

- Create an AWS EC2 instance and setup Ubuntu server on it.
- Push your Flask app code on GitHub. Because, we are going to upload our flask app code on AWS Ubuntu server using GitHub repository.

Logging In to Virtual Machine

Once you'll create your virtual mach 97 2 or any other web service then you'll be able to access it using ssh. Let first login to AWS by running ssh on terminal by running

```
$ ssh -i yourSecretKey.pem ubuntu@ec2-XX-XXX-XXX-XX.yourServerLocation.compute.amazonaws.com
```

Once you'll logged in successfully, then you may be able to see the following screen

```
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-1024-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Oct 10 18:40:54 UTC 2020

  System load:  0.25              Processes:             109
  Usage of /:   34.9% of 9.63GB   Users logged in:       0
  Memory usage: 58%               IPv4 address for eth0: [redacted]
  Swap usage:   0%

 * Kubernetes 1.19 is out! Get it in one command with:

     sudo snap install microk8s --channel=1.19 --classic

   https://microk8s.io/ has docs and details.

10 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable


*** System restart required ***
Last login: [redacted]
ubuntu@ip-[redacted]:~$
```

Now, we are running our AWS VM in terminal. Let's deploy our Flask app now.

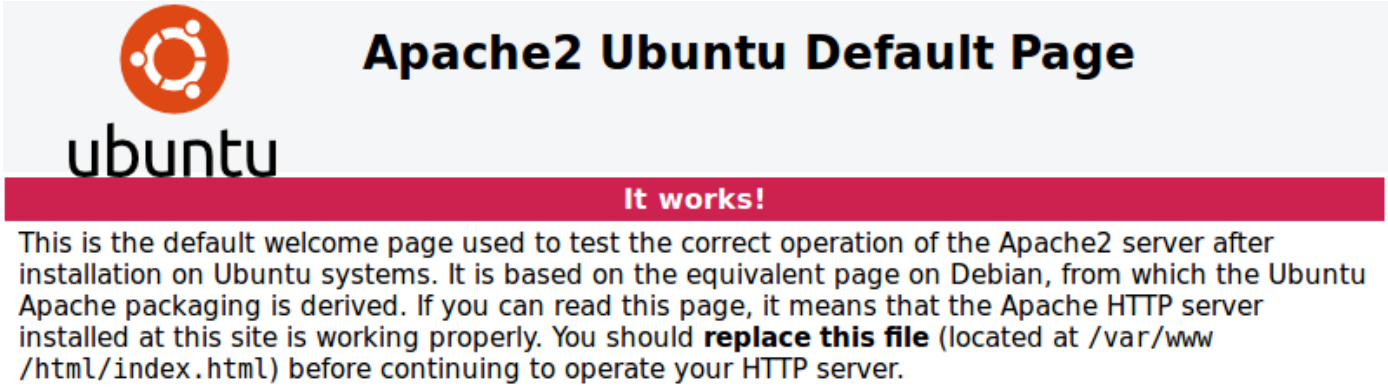
Installing Required Packages of Ubuntu

Let first install following required packages:

- MOD WSGI (Remember to install mod wsgi over python 3)

```
$ sudo apt-get install python3
$ sudo apt-get install python3-pip
$ sudo apt-get install apache2
$ sudo apt-get install libapache2-mod-wsgi-py3
```

Once apache will be installed successfully, you should be able to see this page on your public domain address when you'll run in the browser,



Note: In some articles `libapache2-mod-wsgi` is suggested to install. But I'll recommend to install mod WSGI library for apache over python 3 using `libapache2-mod-wsgi-py3` package. Because, often you may face error on running mod WSGI server or it may give errors that module flask is not installed. This is happen due to your app run on default python version of Ubuntu (i.e. python 2.x) instead of python 3.x which you installed by yourself. So to avoid the error and future difficulties we will install mod WSGI over python 3.

Installing Required Packages for Flask App

Now lets install the required packages for your app.

```
$ sudo pip3 install flask, numpy, pandas
```

flask app even though you have installed them. If you face this error then try installing packages with `sudo`.

Cloning Project On Server

Now lets clone our app code from GitHub into server. First we will install git in our machine if it was not already installed by running following command.

```
$ sudo apt install git
```

As it is supposed that you have already uploaded your code on GitHub repository and so we can access it now by cloning it. Once git is installed then we can clone the project into server by running following command:

```
$ git clone YOUR_GITHUB_PROJECT_HTTP_LINK
```

Let say project was titled as **flaskapp** so we may found a folder with name **flaskapp**. Make it clear by running following command that you have cloned your project in server successfully:

```
$ ls | grep flaskapp
```

Out:

```
flaskapp
```

Our **flaskapp** directory should must have two important following files:

- flaskapp.wsgi (Root file for wsgi, which we will configure with apache server to run flaskapp)

Now lets take a look at the structures of both these files

```
# flaskapp.py
# This is a "hello world" app sample for flask app. You may have a
different file.

from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
   return 'Hello from Flask!'
if __name__ == '__main__':
   app.run()
```

And WSGI file as

```
import sys
sys.path.insert(0, '/var/www/html/flaskapp')
from flaskapp import app as application
```

In the above WSGI file, `/var/www/html/flaskapp` is the location of apache server where we will create a symbolic link of our app directory `~/flaskapp` with apache server. And in the third line `from flaskapp import` is the file `flaskapp.py` which is the root point of our flask app.

Configuration of Flask App with Apache Server Using WSGI

First make sure that you're on the root directory by running following command:

```
$ cd ~
```

`OUR_FLASK_APP_DIRECTORY /var/www/html/flaskapp` as following:

```
$ sudo ln -sT ~/flaskapp /var/www/html/flaskapp
```

Once we have created a link, now lets move to the next step to enable mod WSGI. We have to made some configurations in `/etc/apache2/sites-enabled/000-default.conf` to serve our flaskapp instead of static html pages which apache serve by default. Lets open configuration file by using command:

```
$ sudo vi /etc/apache2/sites-enabled/000-default.conf
```

It may ask you for different actions. Go to edit mode then you see a file with following code:

```
<VirtualHost *:80>
        # The ServerName ...
        # the server uses ...
        # redirection URLs. In the ...
        # specifies what hostname ...
        # match this virtual host. For ...
        # value is not decisive as ...
        # However, you must set ...
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace...
        # error, crit, alert, emerg.
        # It is also possible ...
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
        # following line enables the CGI configuration for this...
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
         Header set Access-Control-Allow-Origin "*"
</VirtualHost>
```

Then after the line `DocumentRoot /var/www/html` add the following code to configure our `flaskapp.wsgi` file with apache:

```
        WSGIDaemonProcess flaskapp threads=5
        WSGIScriptAlias / /var/www/html/flaskapp/flaskapp.wsgi
        WSGIApplicationGroup %{GLOBAL}
        <Directory flaskapp>
             WSGIProcessGroup flaskapp
             WSGIApplicationGroup %{GLOBAL}
             Order deny,allow
             Allow from all
        </Directory>
```

If you have read other articles related to deploying flask app then you may seem that I have added an extra line `WSGIApplicationGroup %{GLOBAL}` after the line `WSGIScriptAlias / /var/www/html/flaskapp/flaskapp/wsgi`. This is to avoid error which you may face when your app has a package with name `__init__.py` created by yourself. In some cases, when we add a module including `__init__.py` file and hit url of our app, and server goes on loading and takes too long and don't give us response. So to avoid such future errors and difficulties, add `WSGIApplicationGroup %{GLOBAL}` line also before `<Directory>` markup. Finally, your configuration file should look like as following:

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        WSGIDaemonProcess flaskapp threads=5
        WSGIScriptAlias / /var/www/html/flaskapp/flaskapp.wsgi
        WSGIApplicationGroup %{GLOBAL}
        <Directory flaskapp>
             WSGIProcessGroup flaskapp
             WSGIApplicationGroup %{GLOBAL}
             Order deny,allow
             Allow from all
        </Directory>

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
         Header set Access-Control-Allow-Origin "*"
                                                                                  1,8           Top
```

Now finally restart the apache server and serve your flaskapp by running following command:

```
$ sudo service apache2 restart
```

Now you should access your flaskapp using your domain url.

Some Error and Difficulties which You May Face

In this section, I'm going to discuss some difficulties or errors which I face and you may face too if you have deployed your app following any other tutorial or article. I'll discuss errors and their solutions but first let me to tell you how you'll find the errors.

When you found that your app is not running successfully and server is responding any kind of error, then try to looking into the error log file `error.log` to find the cause of error. The file is located at `/var/log/apache2/error.log`. You can look into the file by using commands `vi` or `cat`. The commands will be

```
$ vi /var/log/apache2/error.log
```

Some errors list is following:

1. If you found that your app is running on python 2 instead of python 3 then the error may be with installation of your `libapache2-mod-wsgi` package. It has installed with python 2 which is default in Ubuntu. You can found it in your `error.log` file as following:

```
ubuntu@ip-[redacted]:~$ cat /var/log/apache2/error.log
[Sun Oct 11 00:00:21.054562 2020] [mpm_event:notice] [pid 118986:tid 14
0242119162944] AH00489: Apache/2.4.41 (Ubuntu) mod_wsgi/4.6.8 Python/3.
8 configured -- resuming normal operations
[Sun Oct 11 00:00:21.054598 2020] [core:notice] [pid 118986:tid 1402421
19162944] AH00094: Command line: '/usr/sbin/apache2'
```

As you can see in the first line `mod_wsgi/4.6.8 python/3.8 configured`. If you're seeing `python/2.x` instead of `python/3.x` then you have to do following step.

Try to install `libapache2-mod-wsgi` package over python 3 (installed by you) by running following command:

```
$ sudo apt-get install libapache2-mod-wsgi-py3
```

This will install mod wsgi over python 3. Restart apache server and now you should see `python/3.x` instead of `python/2.x`.

2. If you found that you have installed some python packages but still getting error that `no module XXXX is installed` then you have to install you package with root privileges using `sudo`. Let say we found the error `no module flask is installed`, then we will install this module using `sudo` as following:

Since python 3.x is installed, it will install pip3 instead of pip. Then restart the server and your error should gone.

3. If you found that after deploying your flask app or when you uploaded your own python package which contain `__init__.py` file and when you access some of the URLs or APIs, the server is not responding and it's loading for a long time. In some case this could be a cause of importing modules or may be you have used `__init__.py` file in your own python package. This error may not record in `error.log` file. You have to do following step to resolve this error.

Go into the apache configuration file by running following command:

```
$ sudo vi /etc/apache2/sites-enabled/000-default.conf
```

Edit the file and add the line `WSGIApplicationGroup %{GLOBAL}` after the line `WSGIScriptAlias / /var/www/html/YOUR_APP_LINK/WSGI_FILE_NAME` as following:

```
WSGIDaemonProcess flaskapp threads=5
WSGIScriptAlias / /var/www/html/flaskapp/flaskapp.wsgi
WSGIApplicationGroup %{GLOBAL}
<Directory flaskapp>
     WSGIProcessGroup flaskapp
     WSGIApplicationGroup %{GLOBAL}
     Order deny,allow
     Allow from all
</Directory>
```

Then restart the apache server and error should gone. If error is still there, then you should find the cause by trying to deploy first a simple **Hello World Flask App** and then going on to find the cause.

This was all. Hope it will help you to deploy flask app using WSGI and apache server on

Thanks to you for reading this article. Give me a clap and follow me if this article helped you to get more articles on data science, machine learning, big data/data engineering and python.