How To Deploy a Flask Application on an Ubuntu VPS

Posted July 3, 2013 353.2k PYTHON FRAMEWORKS APPLICATIONS APACHE PYTHON UBUNTU

By: Kundan Singh

What the Red Means

The lines that the user needs to enter or customize will be in red in this tutorial! The rest should mostly be copy-and-pastable.

Introduction

Flask is a micro-framework written in Python and based on the Werkzeug and Jinja2 template engine for developing web applications. It is intended for developing web apps quickly.

Setup

You need to have Apache already installed and running on your VPS. If this is not the case, follow Step One of our article on installing a LAMP stack on Ubuntu.

Step One— Install and Enable mod_wsgi

WSGI (Web Server Gateway Interface) is an interface between web servers and web apps for python. Mod_wsgi is an Apache HTTP server mod that enables Apache to serve Flask applications.

Open terminal and type the following command to install mod_wsgi:

```
sudo apt-get install libapache2-mod-wsgi python-dev
```

To enable mod_wsgi, run the following command:

```
sudo a2enmod wsgi
```

Step Two – Creating a Flask App

In this step, we will create a flask app. We will place our app in the */var/www* directory.

Use the following command to move to the /var/www directory:

```
cd /var/www
```

Create the application directory structure using *mkdir* as shown. Replace "FlaskApp" with the name you would like to give your application. Create the initial directory FlaskApp by giving following command:

```
sudo mkdir FlaskApp
```

Move inside this directory using the following command:

```
cd FlaskApp
```

Create another directory FlaskApp by giving following command:

```
sudo mkdir FlaskApp
```

Then, move inside this directory and create two subdirectories named *static* and *templates* using the following commands:

```
cd FlaskApp
sudo mkdir static templates
```

Your directory structure should now look like this:

```
|----FlaskApp
|---------FlaskApp
|--------------static
|--------------templates
```

Now, create the *__init__.py* file that will contain the flask application logic.

```
sudo nano __init__.py
```

Add following logic to the file:

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return "Hello, I love Digital Ocean!"
if __name__ == "__main__":
    app.run()
```

Save and close the file.

Step Three – Install Flask

Setting up a *virtual environment* will keep the application and its dependencies isolated from the main system. Changes to it will not affect the cloud server's system configurations.

In this step, we will create a virtual environment for our flask application.

We will use *pip* to install *virtualenv* and *Flask*. If *pip* is not installed, install it on Ubuntu through apt-get.

```
sudo apt-get install python-pip
```

If *virtualenv* is not installed, use *pip* to install it using following command:

```
sudo pip install virtualenv
```

Give the following command (where venv is the name you would like to give your temporary environment):

```
sudo virtualenv venv
```

Now, install Flask in that environment by activating the virtual environment with the following command:

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

Give this command to install Flask inside:

```
sudo pip install Flask
```

Next, run the following command to test if the installation is successful and the app is running:

```
sudo python __init__.py
```

It should display “Running on http://localhost:5000/” or "Running on http://127.0.0.1:5000/". If you see this message, you have successfully configured the app.

To deactivate the environment, give the following command:

```
deactivate
```

Step Four – Configure and Enable a New Virtual Host

Issue the following command in your terminal:

```
sudo nano /etc/apache2/sites-available/FlaskApp
```

NOTE: Newer versions of Ubuntu (13.10+) require a ".conf" extension for VirtualHost files -- run the following command instead:

```
sudo nano /etc/apache2/sites-available/FlaskApp.conf
```

Add the following lines of code to the file to configure the virtual host. Be sure to change the ServerName to your domain or cloud server's IP address:

```
<VirtualHost *:80>
                ServerName mywebsite.com
                ServerAdmin admin@mywebsite.com
                WSGIScriptAlias / /var/www/FlaskApp/flaskapp.wsgi
                <Directory /var/www/FlaskApp/FlaskApp/>
                        Order allow,deny
                        Allow from all
                </Directory>
                Alias /static /var/www/FlaskApp/FlaskApp/static
                <Directory /var/www/FlaskApp/FlaskApp/static/>
                        Order allow,deny
                        Allow from all
                </Directory>
                ErrorLog ${APACHE_LOG_DIR}/error.log
                LogLevel warn
                CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Save and close the file.

Enable the virtual host with the following command:

```
sudo a2ensite FlaskApp
```

Step Five – Create the .wsgi File

Apache uses the .wsgi file to serve the Flask app. Move to the /var/www/FlaskApp directory and create a file named flaskapp.wsgi with following commands:

```
cd /var/www/FlaskApp
sudo nano flaskapp.wsgi
```

Add the following lines of code to the flaskapp.wsgi file:

```
#!/usr/bin/python
import sys
```

```
import logging
logging.basicConfig(stream=sys.stderr)
sys.path.insert(0,"/var/www/FlaskApp/")

from FlaskApp import app as application
application.secret_key = 'Add your secret key'
```

Now your directory structure should look like this:

```
|--------FlaskApp
|----------------FlaskApp
|-----------------------static
|-----------------------templates
|-----------------------venv
|-----------------------__init__.py
|----------------flaskapp.wsgi
```

Step Six – Restart Apache

Restart Apache with the following command to apply the changes:

```
sudo service apache2 restart
```

You may see a message similar to the following:

```
Could not reliably determine the VPS's fully qualified domain name, using 127.0.0.1 for S
```

This message is just a warning, and you will be able to access your virtual host without any further issues. To view your application, open your browser and navigate to the domain name or IP address that you entered in your virtual host configuration.

You have successfully deployed a flask application.

Article Submitted by: Kundan Singh

By: Kundan Singh

Related Tutorials

How To Optimize Docker Images for Production

How to Set Up a Scalable Django App with DigitalOcean Managed Databases and Spaces

How To Build a Weather App with Angular, Bootstrap, and the APIXU API

How To Integrate MongoDB with Your Node Application

How To Install YunoHost on Debian 9

131 Comments

Leave a comment...

Log In to Comment

apkech *July 9, 2013*

1 One can simply use tornado and will avoid apache and setting virtual hosts.

coolit *July 5, 2016*

0 Old comment but...yeah you sure can, and i've actually been playing with tornado frontended with cloudflare as my nginx/cache/security. It works really well and multi-threading is pretty sweet too. Example deployment with tornado below...

Main
if **name** == "**main**":

```
# TORNADO WEB SERVER REGISTRATION
server = HTTPServer(WSGIContainer(app))
server.bind(80)
server.start(0)  # Forks multiple sub-processes, 1 per CPU
IOLoop.current().start()
```

garethprice *August 16, 2013*

0 I have followed this exactly, but instead of putting mywebsite.com i put flaskapp.myip in and tried going to http://flaskapp.myip. It came up with a "flaskapp.myip is unavailable or may not exist." error.

What should I do?

garethprice *August 16, 2013*

0 Okay, now I've got it to sort of work, but instead of it showing the "Hello I love Digital Ocean" message it just shows me a directory of my files... :/

kamaln7 MOD *August 16, 2013*

1 @garethprice: Disable Apache's default virtualhost:

```
sudo a2dissite default
```

Edit the virtualhost you created and set ServerName to e.g.

```
ServerName flaskapp.dev
```

and restart Apache.

```
sudo service apache2 restart
```

Edit

```
/etc/hosts
```

on your computer and add this line to the bottom:

```
1.2.3.4 flaskapp.dev
```

Where 1.2.3.4 is your droplet's IP address. Save it and point your browser to http://flaskapp.dev - it should load your flask app properly.

---

garethprice *August 16, 2013*

0 Thanks, Kamal. I'll give it a go.

But what if I want others to be able to access it?

---

kamaln7 MOD *August 17, 2013*

0

@garethprice: I assumed you don't have a domain name since you set it to something.ip at first. If you own a domain name:

- Follow this article to set up DNS records: https://www.digitalocean.com/community/articles/how-to-set-up-a-host-name-with-digitalocean

- Set ServerName to your domain name and restart Apache

---

garethprice *August 18, 2013*

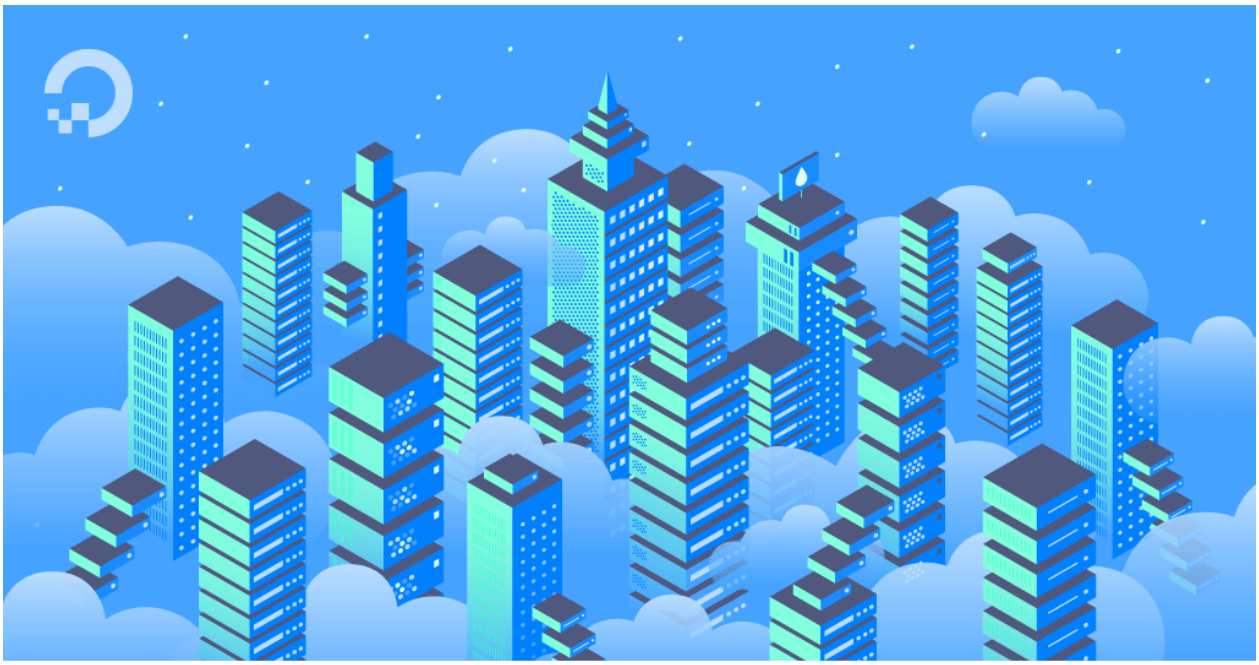
0

Kamal,

I set up a new droplet and followed this tutorial: https://www.digitalocean.com/community/articles/how-to-launch-your-site-on-a-new-ubuntu-12-04-server-with-lamp-sftp-and-dns

I did what you suggested and all I get is my list of files still. I've copied exactly your FlaskApp demonstration. You can see them at: http://opendiscovery.co.uk/FlaskApp/...

Sorry for being noobish.



How To Launch Your Site on a New Ubuntu 12.04 Server with LAMP, SFTP, and DNS

by Etel Sverdlov

This article will take you from a fresh, new server to an online, working site. It will combine the relevant articles into a cohesive step by step guide to set up your personal site on an Ubuntu server.

kamaln7 MOD *August 21, 2013*

0 @garethprice: Try running the following commands:

```
sudo a2dissite default
sudo service apache2 restart
```

Does that fix it?

garethprice *August 22, 2013*

0 Yes, thank you! Last question: is there a way of getting the normal apache to load html/php files in the root dir, and wsgi to handle Flask in a subdir (as achieve through WSGIScriptAlias)?

kamaln7 MOD *August 27, 2013*

0 @garethprice: The recommended way of doing that is to have php/html in a separate directory and your flask app in another directory and **not** mixing both apps together.

Create a separate virtualhost on a separate subdomain/domain for the php/html app.

itsanjalirk *September 2, 2013*

0 Hi
I have followed the exacts steps as recommended.
All the print statements from apache to flaskapp.wsgi to __init__.py file gets printed.
No error reported in apache-error log file
From from FlaskApp import views also works fine as it prints the first line in views.py
but when I type the url http://ip/hello in the browser and post I get 404.

View.py
#from FlaskApp import app
app = Flask(__name__)

```
@app.route("/hello")
def hello():
return "Hello, World!"
```

```
__init__.py
from flask import Flask
print "Importing 1"
#app = Flask(__name__)
app = Flask('FlaskApp')
from FlaskApp import views
```

Can you please guide on how to resolve

---

kamaln7 MOD *September 2, 2013*

0 @itsanjalirk: Please pastebin apache's config files.

---

john222195 *September 17, 2013*

0 Hi - What is the purpose of creating a FlaskApp directory inside another FlaskApp directory? Is it just for python package management? Couldn't you just have a main.py file inside one FlaskApp and then

from main import app

?

---

kamaln7 MOD *September 17, 2013*

0 @john: It's so that you can create files such as .wsgi that do not belong to the app itself but are required to make it work.

e.g. FlaskApp.wsgi is created in /var/www/FlaskApp

---

ljernejcic *September 17, 2013*

0 I am getting an import error:

ImportError: No module named flask

Isn't the WSGI file supposed to activate the virtual environment we created or something?

dogwynn *August 5, 2015*

0 If you modify your flaskapp.wsgi as such:

```
#!/usr/bin/python
activate_this = '/var/www/FlaskApp/FlaskApp/venv/bin/activate_this.py'
execfile(activate_this, dict(__file__=activate_this))

import sys
import logging
logging.basicConfig(stream=sys.stderr)
sys.path.insert(0,"/var/www/FlaskApp/")

from FlaskApp import app as application
application.secret_key = 'Add your secret key'
```

Lines 2 and 3 activate the virtual environment containing the Flask module you installed.

acronymcreations *May 6, 2017*

0 When I add those two lines of code, I get

```
No such file or directory: '/var/www/FlaskApp/FlaskApp/venv/bin/activate_this.p
```

error. How can I fix this?

kamaln7 MOD *September 18, 2013*

0 @ljernejcic:

```
ImportError: No module named flask
```

Flask isn't installed. Did you follow **Step Two – Install Flask**?

newaccount *September 21, 2013*

0 Correct me if I'm wrong but you create a virtualenv and then promptly set up an app which doesn't use the virtualenv... right?

---

brian.schiller *October 2, 2013*

0 This worked great for the simple Hello World example. I tried to adapt it to fit my own website, and it's no longer working. The browser says 500 Internal Error, which turns out to be an ImportError inside the apache logs. I think it's a different problem from ljernejcic because the ImportError is occurring on my own module.

The apache error.log output is at http://pastebin.com/aRau7TwA.

Thanks in advance for any ideas.

---

kamaln7 MOD *October 6, 2013*

0 @brian.schiller: That paste has been removed. Are you still experiencing this issue?

---

ultrafaca *October 10, 2013*

0 I have used your above described configuration to set up my flask application and I'm experiencing following issue:
Basic page (and every static page) is displayed with 404.html message. Only login page is displayed correctly.
I did my site similarly as http://www.youtube.com/watch?v=jELLsj1KPNQ; tested on my own linux machine, with flask development server. It works.
So, I have renamed main file (the one I called with application name to __init__.py, as described here.
Structure looks like this:

/myapp
myappwhatever.wsgi
/myapp
__init__.py
/templates --------- bunch of html files (with basic html ancestor and childs)
/static --------- css
module1.py
module2.py
module3.py

And I made Apache2 and wsgi file according your description. And it does not work as supposed.
Do you have any clue why?

Thanks

erikvmalmberg *April 11, 2017*

0 I can't for the love of God get my Flask app up and running with my own modules - it works fine with having one .py file in the directory, but as soon as I try to do some "import wtf*forms" or whatever, apache2 spits out an error saying "Module not found" - I have tried with the WSGIPythonPath, "activate*this", adding the path to the venv.

Why is it not working? I can't be the only one receiving "ImportError: No module named "XXXX" found".

(Running python3)

michelrobijns *April 28, 2017*

0 Are your modules installed in a virtual environment? Did you make sure WSGI actually loads the virtual environment?

acronymcreations *May 6, 2017*

0 How can I check if the WSGI is actually loading the virtualenv? I'm pretty sure this is what is causing my problems.

Load More Comments