How to set up Flask on Digital Ocean

Things you need before getting started

- Hosting website
    - Digital Ocean
    - We will be using Digital Ocean for this web app
- Domain name
    - You don't need this right away since you can just visit the IP address
    - You need to create the droplet first before you do this step
    - Godaddy Example
        - change the @ host to the IP address of the droplet that it points to
        - Then you want to change the nameservers
            - And all the nameservers for digitalocean
            - You can get these from the DNS tab in Digital Ocean after you link the domain name and the IP together
            - The name servers at the time
                - NS1.DIGITALOCEAN.COM
                - NS2.DIGITALOCEAN.COM
                - NS3.DIGITALOCEAN.COM
            - Takes some time for the changes to happen

On Digital Ocean

- Make and log into your account
- Create a droplet
    - Give it a name
    - Select the size you need
        - The size chosen was the following:
            - 512 MB/ 1 CPU
            - 20GB SSD DISK
            - 1TB Transfer
    - Choose a region
        - New York 2 was chosen
    - Choose Ubuntu
        - Version chosen is 14.04 x64
    - You want to enable backup for production
    - Enable VirtIO
    - Wait for droplet to be created

How to connect to your server on DO

- Windows
  - You can use putty
- Mac
  - You don't anything you can just use the terminal

- To log into the server using SSH:
  - ssh root@SERVER_IP_ADDRESS
  - Then log in with your password (if its the first time digital ocean will email it to you then you can change it after you log in)
  - For first time log in update the server using
    - *sudo apt-get update*
    - *sudo apt-get upgrade*

To Stop the SSH to time out quickly:

To start, we continue with a few setting changes in our server. The default SSH timeout setting is in place for security purposes, but it can also be quite annoying, especially in a development environment. You will find yourself disconnected often after searching Google for some answers, getting a snack... etc.

So we want to extend the SSH timeout. To do this:

```
cd /etc/ssh
```

```
sudo nano sshd_config
```

`control + w`, searching for "keepalive"

Now we want to add in the following below the TCPKeepAlive yes:

```
ClientAliveInterval 30
```

```
ClientAliveCountMax 99999
```

Now we won't be kicked out so quickly on our development server. Keep in mind that you may be best off keeping the default parameters here on your live production server, since it is a security precaution.

Making the Flask App on Digital Ocean:

Now that you've got your server and domain set up, it is time to set up Flask and get your very first web application up! There are many commands that we will need to run.

We need to get WSGI, so run the following:

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

Once we have that, we need to make sure we've enabled WSGI with the following:

```
sudo a2enmod wsgi
```

It is probably already enabled from the installation, but it is a good idea to make sure.

Next we are ready to set up our Flask environment.

Run:

```
cd /var/www/
```

Now let's make our Flask environment directory:

```
sudo mkdir FlaskApp
```

Move into that directory:

```
cd FlaskApp
```

Now make the actual application directory:

```
sudo mkdir FlaskAppName
```

Now let's go in there:

`cd FlaskAppName/`

Now we're going to make two directories, static and template:

`sudo mkdir static`

`sudo mkdir templates`

Now we're ready to create the main file for your first Flask App:

`sudo nano __init__.py`

Here is where we have our initialization script for our Flask application. You can actually keep all of your main website code right here for simplicity's sake, and that's what we'll be doing just for this tutorial to get you guys started. Within your __init__.py file, you will type:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def homepage():
    return "Hi there, how are you doing? It’s a beautiful day, let’s go play outside "

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

Make sure to save the file.

Now we should probably actually get Flask. Let's do that now.

Since this is likely a new server for you, you will want to go ahead and run if you didn’t run these when you first logged onto the server

`sudo apt-get update`

```
sudo apt-get upgrade
```

To get Flask, we're going to use pip, so you will need to first get pip if you do not already have it:

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

Now that we have pip, we also need virtualenv to create the virtual environment for Flask to run Python and your application in:

```
sudo pip install virtualenv
```

Now to set up the virtualenv directory:

```
sudo virtualenv venv
```

Activate the virtual environment:

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

Now install Flask within your virtual environment

```
sudo pip install Flask
```

Find out if everything worked out by going:

```
sudo python __init__.py
```

If you didn't get any major errors, congrats you got it working! :)

Type deactivate to stop the virtual environment running locally. This is only a local version, so you won't be able to type in anything to your browser to access it.

So now we need to set up our Flask configuration file:

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

This is where your Flask configuration goes, which will apply to your live website. Here's the code that you need to include:

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

For your notes, if you want to add more domains that point to the same Flask App, or a different app entirely, then you just copy and paste the above code and change the domain name. So, if you want www.yourapp.com and yourapp.com to both work, you will need two versions. We are now ready to enable the server.

Run:

`sudo a2ensite FlaskApp`

`service apache2 reload`

Almost done... now we just need to configure our WSGI file. To do this:

`cd /var/www/FlaskApp`

`sudo nano flaskapp.wsgi`

Within the wsgi file, enter:

```
#!/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 = 'your secret key. If you share your website, do NOT share it with this key.'
```

Save and exit.

Once that is done, run:

`service apache2 restart`

Get used to running the above command. Flask is very finicky about your python file changes. Every .py file change you make to your webapp, you need to run this command. Also as a side note you do not need the secret key it's used for production purposes.

Once you have done all of this, you are ready to visit your domain name in your browser. You should see the "Hi there, how you doing? It's a beautiful day, let's play outside" string that we put in your __init__.py file.

Link your Digital Ocean flask app to Github

You want to save all your changes on a repo. Below is a link to how to set up your github repo with your flask app on digital ocean.

https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/

Database

Installing PostgreSQL on DO(you will need this in your 2nd phase of your project not on the 1st phase):
https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-14-04