MAG(i)C Random in Python on Ubuntu 18.04 LTS

Overview

This is a simple overview of how a functioning implementation of the random number generator was achieved. Most of the guidance on how to do this was from the DigitalOcean article on how to set up Flask, uWSGI, and Nginx on Ubuntu 18.04 LTS, located here: DigitalOcean and Flask

Prerequisites

- Ubuntu 18.04 LTS server
- Nginx (installed)
- MAG(i)C (found in the GitHub repository)

Software Versions

The MAG(i)C Random Number Generator in Python relies on:

- Python 3.6 or 3.7 (will not work on 2.7)
- Flask 1.0.2
- Ubuntu 18.04 LTS

Keep in mind that you may have success running other versions, however it is not guaranteed to work.

Installing Required Packages from Ubuntu Repositories

To install the required packages, run the following in the terminal of your VM:

```
$ apt update
$ sudo apt install python3-pip python3-dev build-essential libssl-dev libffi-dev py
```

Creating a Python Virtual Environment

To ensure that our implementation is isolated from the rest of the system, we need to create a Python virtual environment.

To do this, we need the Python `venv` tool.

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

We then need to make a folder for all of our files to reside. Making this in your user's home directory is adequate.

```
$ mkdir ~/myproject
$ cd ~/myproject
```

To create the environment itself, run the following:

```
$ python3.6 -m venv myprojectenv
```

To install applications in this virtual environment, we need to start it:

```
$ source myprojectenv/bin/activate
```

Setting up Flask

Since we are in our virtual environment, we can install Flask and uWSGI.

To start, install `wheel` with the instance of `pip` (the Python package manager).

```
$ pip install wheel
```

With `wheel` installed, we can go ahead and install Flask and uWSGI.

```
(myprojectenv) $ pip install uwsgi flask
```

Flask Application (myproject.py)

With the environment set up and everything installed, we can pull the myproject.py file found in the GitHub repository into '~/myproject'. Here it is for reference, but please see the GitHub repository for the latest version:

```
from flask import Flask
import random
app = Flask(__name__)

@app.route("/")
def hello():
    return str(random.randint(0, 1000000))

if __name__ == "__main__":
    app.run(host='0.0.0.0')
```

Simple, right?

Now that we have the Random app Python file in place, we can begin setting up uWSGI.

Setting Up uWSGI: (wsgi.py)

We need a file that is an "entry point" for our web application that will tell our uWSGI server how to interact with our code.

```
from myproject import app

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

Test that the uWSGI server is working by running the following and trying to access http://server_ip:5000 after running the following command:

```
$ sudo ufw allow 5000
(myprojectenv) $ uwsgi --socket 0.0.0.0:5000 --protocol=http -w wsgi:app
```

If this does not work, you need to to some troubleshooting that is outside the context of this documentation.
Exit this application by pressing 'CTRL-C' and go ahead and exit the project by typing `deactivate` in the terminal.

uWSGI Initialization Parameters (myproject.ini)

We need to be able to tell uWSGI to apply certain settings outlined in the myproject.ini file, namely:

- Module Name (what code to run)
- To spawn 30 processes
- To use myproject.sock to communicate (UNIX sockets are much faster and more secure than using the network)
- To set the correct permissions for the socket.
- Other attributes that will not be explained here.

By default, uWSGI will use the 'uwsgi' protocol, which is very fast. Nginx natively understands this protocol, which helps greatly when experiencing large amounts of load.

systemd Unit File (myproject.service)

Note: The `myproject.service` file is the first file that is not recommended to copy from the GitHub repository. This is due to there being a lot of specific file paths that will be different on each configuration. Included below is a template:

```
# Template from DigitalOcean
# <user> = username of account on server

[Unit]
Description=uWSGI instance to serve myproject
After=network.target

[Service]
User=<user>
Group=www-data
WorkingDirectory=/home/<user>/myproject
Environment="PATH=/home/<user>/myproject/myprojectenv/bin"
ExecStart=/home/<user>/myproject/myprojectenv/bin/uwsgi --ini myproject.ini

[Install]
WantedBy=multi-user.target
```

Once the `myproject.service` file is created, we can start the uWSGI service and set it to start at boot.

```
$ sudo systemctl start myproject
$ sudo systemctl enable myproject
```

Now we can check the status to see if it is running:

```
$ sudo systemctl status myproject
```

You should see that the service is 'active (running)'. If there are any errors, those need to be resolved before continuing.

Nginx Configuration

Now that uWSGI is set up and running, we can create the Nginx server configuration.
Start by creating a configuration file in the `sites-available` directory.

```
$ sudo nano /etc/nginx/sites-available/myproject
```

Nginx uses "server blocks" to know what sites it is hosting, and where to look for their data. Below is a template from DigitalOcean for this file:

```
server {
    listen 80;
    server_name your_domain www.your_domain;

    location / {
        include uwsgi_params;
        uwsgi_pass unix:/home/sammy/myproject/myproject.sock;
    }
}
```

Save and close this file after you have completed the configuration.

To enable the server block just created, link the file to the `sites-enabled` directory.

```
$ sudo ln -s /etc/nginx/sites-available/myproject /etc/nginx/sites-enabled
```

Test the file using:

```
$ sudo nginx -t
```

If successful, restart Nginx and configure the firewall using:

```
$ sudo systemctl restart nginx
$ sudo ufw delete allow 5000
$ sudo ufw allow 'Nginx Full'
```

You should now be able to access the website!

```
http://yourdomain_or_ip/
```

Troubleshooting

Please look in the following locations for error/connection logs:

```
$ sudo less /var/log/nginx/error.log
$ sudo less /var/log/nginx/access.log
$ sudo journalctl -u nginx
$ sudo journalctl -u myproject
```