Configuring Flask With uWSGI+NGINX

|  |  |  |
| --- | --- | --- |
| **Version** | **Author** | **Revision date** |
| 1.0 | Eric Matte | 2016-10-16 |

## Description

In this guide, we will be setting up a simple Python application using the Flask micro-framework on Ubuntu 16.04.1 LTS. The bulk of this article will be about how to set up the uWSGI application server to launch the application and Nginx to act as a front end reverse proxy.

# Prerequisites

Before starting on this guide, you should have a non-root user configured on your server. This user needs to have sudo privileges so that it can perform administrative functions.

# Install the Components from the Ubuntu Repositories

Install all the required packages:

|  |
| --- |
| sudo apt-get update  sudo apt-get install python-pip python-dev nginx uwsgi |

# Create a Python Virtual Environment

Next, we need to create a virtual environment for our flask application.

Install the **virtualenv** package using **pip**:

|  |
| --- |
| sudo pip install virtualenv |

By conventions, most people uses **~/.virtualenvs** as the home for virtual environment.

Create your new virtual environment:

|  |
| --- |
| mkdir ~/.virtualenvs  cd ~/.virtualenvs  sudo virtualenv -p python3 --no-site-packages **<venv\_name>**  source ~/.virtualenvs/**<venv\_name>**/bin/activate |

You should now be operating within your virtual environment as shown below:

|  |
| --- |
| (**<venv\_name>**)**<user>**@**<host>**:~/.virtualenvs$ |

# Set Up the Flask Application

Install flask:

|  |
| --- |
| pip install flask Jinja2 |

Get the flask application from the git repository:

|  |
| --- |
| cd ~/  git clone **<url\_to\_git\_repo>**  ln -s **<repo\_name>/api** /opt/flask-api |

Test that your application is working by running its main file:

|  |
| --- |
| python /opt/flask-api/**<app\_name>**.py |

If you got a message like the one below, your application is working.

|  |
| --- |
| \* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit) |

You can now stop quit the application and deactivate the virtual environment:

|  |
| --- |
| deactivate |

# Create the WSGI Entry Point

In your project, you should have a file named **wsgi.py**. If not, create it and link it to your flask main python file:

|  |
| --- |
| from **<app\_name>** import app  if \_\_name\_\_ == "\_\_main\_\_":  app.run(threaded=True) |

# Configure uWSGI

You need to create a ini file to hook uWSGI with your project in the **apps-available** directory:

|  |
| --- |
| sudo nano /etc/uwsgi/apps-available/**<app\_name>**.ini |

Add this to the file :

|  |
| --- |
| [uwsgi]  # working paths  virtualenv = /home/ubuntu/.virtualenvs/**<venv\_name>**  chdir = /opt/flask-api  module = wsgi # point to the wsgi.py file in chdir  callable = app # the name of the var in module (for app.run())  # server settings  processes = 5  manage-script-name = true  plugin = python3  master = True  vacuum = True  die-on-term = true  # user settings  setuid ubuntu  setgid www-data  # connection settings  chmod-socket = 660  socket = /tmp/%n.sock |

Now enable your ini file and start uWSGI with your application:

|  |
| --- |
| sudo ln -s /etc/uwsgi/apps-available/**<app\_name>**.ini /etc/uwsgi/apps-enabled/  sudo service uwsgi restart |

Do a **ls** command in the **/tmp** directory. If you see **<app\_name>.sock**, everything is good.

# Configuring Nginx to Proxy Requests

Create a new server block configuration file for the project in Nginx **sites-available** directory.

|  |
| --- |
| sudo nano /etc/nginx/sites-available/**<app\_name>** |

Add this to the file :

|  |
| --- |
| server {  listen 80;  server\_name **<server\_address>**;  location / {  include uwsgi\_params;  uwsgi\_pass unix:/tmp/**<app\_name>**.sock;  }  ## Example for a localhost server proxy  # location /f/ {  # proxy\_pass http://0.0.0.0:8000/;  # proxy\_set\_header X-Real-IP $remote\_addr;  # proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  # proxy\_set\_header Host $host;  # }  } |

Enable the configuration file and restart nginx :

|  |
| --- |
| sudo ln -s /etc/nginx/sites-available/**<app\_name>** /etc/nginx/sites-enabled/  sudo service nginx restart |

You should be now all set!