Configuring virtualenv

From Haverford Digital Scholarship

Contents

Introduction

This is a step-by-step set of instructions for how to create a *new* Django-based project on ds-web within a python virtualenv from an *existing* repo, and configuring Nginx and uWSGI.

Clone the Github Repository

```
cd /srv
```

This folder contains the repos of all the Django (or other) projects

```
git clone https://url-to-the-repo.git
chown www-data:www-data reponame
```

Now change to root

```
sudo su -
```

From now on, do not sudo your commands, especially pip installs, as you may end up installing components on the system level instead of the virtualenv level (this is bad)!

Create the virtualenv

```
cd /usr/local/lib/python-virtualenv
```

Within this folder, there will likely be existing projects. You need to create a new project with the following command:

```
virtualenv projectname
```

Now you can activate the virtualenv for this project:

```
cd projectname
source bin/activate
pip install django==1.6 (or whatever version this project uses)
pip freeze (shows what's currently installed)
```

```
cd /srv/reponame
pip freeze > requirements.txt
```

This dumps the contents of "pip freeze" into the requirements.txt file so we can re-deploy the project with all its dependencies in the future if necessary.

Now, check to make sure it's running

```
python manage.py runserver
```

If you want to exit the virtualenv, use the following command:

```
deactivate
```

Configure uWSGI

The first thing to do here is to create a .ini file for the project in the apps-available directory. Then when we want to enable the project, we can create a symlink to it in apps-enabled.

```
cd /etc/uwsgi
cd apps-available
vim projectname.ini
```

Paste the following into the .ini file you've just created, replacing projectname and reponame where appropriate.

```
[uwsgi]

# Environmental settings
uid = www-data
gid = www-data
socket = /run/uwsgi/app/projectname/projectname.socket

# For testing: curl http://localhost:5000
#socket = localhost:5000
#protocol = http
```

```
# Application settings
plugins = python
chdir = /srv/reponame
virtualenv = /usr/local/lib/python-virtualenv/projectname
module=reponame.wsgi:application

# Performance tuning
processes = 4
threads = 2
```

Comment out the file socket (line beginning with (socket) under "# Environmental Settings" and uncomment the testing socket and protocol beneath it to do a TCP socket test first. Save and quit.

Then, make a symbolic link in apps-enabled to apps-available:

```
cd apps-enabled
ln -s projectname.ini ../apps-available/projectname.ini
```

And restart uwsgi

```
service uwsgi restart
```

Configure Nginx

```
cd /etc/nginx
vi projectname
```

Paste in the following, replacing projectname and reponame where appropriate:

```
server {
    listen 80;
    server_name projectname.haverford.edu;

    location /static {
        alias /srv/reponame/static;
    }

     location / {
         include /srv/reponame/uwsgi_params;
         uwsgi_pass unix:/run/uwsgi/app/projectname/projectname.socket;
     }
}
```

Save and quit. Also, make sure /srv/reponame/uwsgi_params exists! If it doesn't, create the file and paste in the following:

```
  uwsgi_param     QUERY_STRING            $query_string;
  uwsgi_param     REQUEST_METHOD          $request_method;
  uwsgi_param     CONTENT_TYPE            $content_type;
  uwsgi_param     CONTENT_LENGTH          $content_length;

  uwsgi_param     REQUEST_URI             $request_uri;
  uwsgi_param     PATH_INFO               $document_uri;
  uwsgi_param     DOCUMENT_ROOT           $document_root;
```

```
  uwsgi_param     SERVER_PROTOCOL         $server_protocol;

  uwsgi_param     REMOTE_ADDR             $remote_addr;
  uwsgi_param     REMOTE_PORT             $remote_port;
  uwsgi_param     SERVER_PORT             $server_port;
  uwsgi_param     SERVER_NAME             $server_name;
```

Save and quit, then restart Nginx:

```
service restart nginx
```

Test the TCP Socket and set up the file socket

Now test the TCP socket we've temporarily created with the curl statement in the .ini file:

```
curl http://localhost:5000
```

If you get HTML, then uWSGI has been configured correctly! Now we can disable the TCP socket, and enable the file socket by commenting the section under "# For testing" and uncommenting the "socket" line under "# Environmental settings". Restart uWSGI again with:

```
service uwsgi restart
```

Then go to /run/uwsgi/app/projectname/ and look for the file socket. If it's there, you've successfully configured the project!

Test full configuration by fooling your local machine

Home stretch! Now you have to fool your local machine into sending requests to ds-web.haverford.edu when you put the project's URL into your browser. On your local station, modify your host name file to point to it and do testing.

On most Linux and Mac machines, this file is in /etc/hosts file. Paste this line into that file:

```
165.82.124.18 projectname.haverford.edu
```

You should be able to put projectname.haverford.edu into your browser and see your project!

Lastly, open a ticket with Prodesk to create the DNS record (or switch an existing one)

You're done! Congrats!

- This page was last modified on 17 April 2015, at 13:36.

- This page has been accessed 19 times.