Linux & Unix Servers

How to deploy web.py applications

In this article you will learn how to deploy a web.py application under Linux / UNIX environments.

You can refer to our article titled, “How to install web.py” if you don’t know what web.py is or if web.py is currently not installed on your machine / server.

The "main" application file in web.py is typically called "code.py".

Note: We will be deploying a simple "Hello World" application to test our installation of web.py. You will learn how to follow this procedure to deploy web.py applications in the future.

First, you need to create all child and parent directories by using the command below:

```
sudo mkdir -p /var/www/example.com/application
```

Save a file called “code.py” under directory /var/www/example.com/application. Please note that this directory should also be same in the Apache virtual host configuration which we will cover later in this article.

```
sudo vim /var/www/example.com/application/code.py
```

```
import web



urls = (

    '(.*)', 'hello'

)
```

```
app = web.application(urls, globals())



class hello:

    def GET(self, name):

        if not name:

            name = 'World'

        return 'Hello, ' + name + '!'



if __name__ == "__main__":

    app.run()
```

We all know there are numerous ways to deploy a web.py application. There are a number of web servers already in the market like Apache, Lighttpd, NGINX, and Cherokee and more to come.

We will be using the most famous Apache web server along with the module mod_wsgi, which is used to parse and serve python applications.

WSGI is an evolution of the CGI standard and has performance comparable to FastCGI and embedded interpreter application deployments.

We assume at this stage that you already have installed Apache, python, mod_wsgi configured and it is ready to be used.

If you don’t know how to install Apache, python, mod_wsgi, please refer to one of our pervious articles on “How to setup python with mod_wsgi and apache”.

Enable Apache web server’s mod_rewrite which is used for the URL rewriting by

using the command below.

```
sudo a2enmod rewrite
```

WSGI requires a very minor change to be made to your web.py application. Add the content below to end of the file we created earlier called “code.py”.

```
app = web.application(urls, globals(), autoreload=False)

application = app.wsgifunc()
```

Please refer to your previous created virtual host, which we did in our previous article called “example.com”.

The virtual host file for “example.com” looks like below after the modification.

```
<VirtualHost example.org:80>

  ServerAdmin admin@example.com

  ServerName example.com

    ServerAlias www.example.com

    DocumentRoot /var/www/example.com/public_html/

    ErrorLog /var/www/example.com/logs/error.log

    CustomLog /var/www/example.com/logs/access.log combined



  WSGIScriptAlias / /var/www/example.com/application

  Alias /static /var/www/example.com/public_html



  <Directory /var/www/example.com/application>

   SetHandler wsgi-script
```

```
    Options ExecCGI
  </Directory>

  AddType text/html .py

  <Location />
    RewriteEngine on
    RewriteBase /
    RewriteCond %{REQUEST_URI} !^/static
    RewriteCond %{REQUEST_URI} !^(/.*)+code.py/
    RewriteRule ^(.*)$ code.py/$1 [PT]
  </Location>
</VirtualHost>
```

If your Apache virtual host is not yet enabled, you may enable it using the command below. If it is already enabled there is no need to do it now and you can kindly skip this step.

```
sudo a2ensite example.com
```

Finally, restart your Apache web server by using the command below:

```
sudo /etc/init.d/apache2 restart

or

sudo service apache2 restart
```

A small overview of what we have done above, all the requests for the example.com domain will be handled by WSGI, with the application files located in "/var/www/example.com/application".

All static files can be stored in /var/www/example.com/public_html and served directly by Apache. Furthermore, the rewrite rules convert requests so that paths beneath example.com are handled by the Web.py application without including code.py in the URL.

For example, the request for http://example.com/about would be processed as http://example.com/code.py/about but requests for http://example.com/static would not be rewritten by the mod_rewrite of Apache and content would be served from the directory "/var/www/example.com/public_html".

By opening the web browser, you should be able to see our nice “Hello world” application based on web.py

If yes, congratulations! You have now deployed a web application based on web.py

Unique solution ID: #1316
Author: Pat Lathem
Last update: 2015-12-03 20:39