

Maxime Serrano and Joshua Zimmerman
{mserrano,jzimmerm}@andrew.cmu.edu
10/24/2013

15-221
Fall 2013
Mixed

Homework 4: Instructions

How to Install Flask

1 Overview

When creating websites, it is often useful to have a system of functions and objects to abstract away the lower-level protocols for you. For example, rather than manually writing HTML to present to the user, many modern websites use *templating systems* to automatically generate it dynamically. Flask is a Python framework that allows for easy creation of complex websites.

This tutorial will teach you how to install Flask and set it up in a production environment. The tutorial has *5 steps* and should take up to *15 minutes* to complete, though depending on your environment, you may have to wait longer for some of the commands to complete. After completing these steps, you should be ready to write and deploy a custom, dynamic website using the Flask framework.

This tutorial is meant for *technical users* who are familiar with Linux, Python and the Bourne shell. Users should be experienced enough to use a package manager, understand the difference between a normal user and `root`, and edit text files.

For this tutorial you must use the Bourne shell or one of its variants on Ubuntu Linux. While these instructions may also work for other distributions, they are not guaranteed to do so. You will also need internet access.

2 Steps

2.1 Step 1 - Install Python and Pip

This step outlines how to install Python and Pip in the terminal.

1. Open a terminal.
2. Within the terminal, type the following commands:

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

You may be asked for your password. If you are, type it in. The characters you type for your password will not display - but rest assured that they are being kept track of. You may also be asked if you are sure, with the prompt `[Y/n]`. Simply hit enter to confirm.

3. Type `python` into your terminal, and hit enter. Your terminal should look like the following:

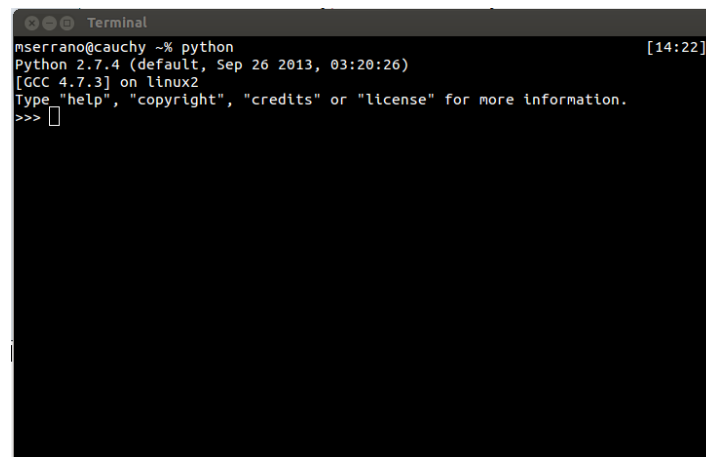
A screenshot of a terminal window titled "Terminal". The prompt is "mserrano@cauchy ~%". The user has entered "python", and the terminal displays "Python 2.7.4 (default, Sep 26 2013, 03:20:26)" and "[GCC 4.7.3] on linux2". It then shows the Python help text: "Type 'help', 'copyright', 'credits' or 'license' for more information." followed by a prompt ">>>" and a cursor.

Figure 1: A Python session.

4. Type `exit()` to exit out of Python. You now have Python and Pip installed!

2.2 Step 2 - Install Flask

This step outlines how to install Flask using Pip, a package manager for Python libraries and utilities.

1. Open a terminal, or use an already open terminal.
2. Within the terminal, type the following command:

```
sudo pip install Flask
```

You may be asked for your password. If you are, type it in. The characters you type for your password will not display - but rest assured that they are being kept track of. You may also be asked whether or not you are sure. When that prompt appears, simply hit the Enter key to accept.

3. Open your favorite text editor, such as `vim` or `emacs`.

4. Type the following text into your text editor:

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return "Hello world!"

app.run()
```

5. Save the file as `flask_test.py` and close it.
6. Run the file by typing `python flask_test.py`.
7. Open your web browser, and navigate to `http://localhost:5000/`. You should see something like the following:

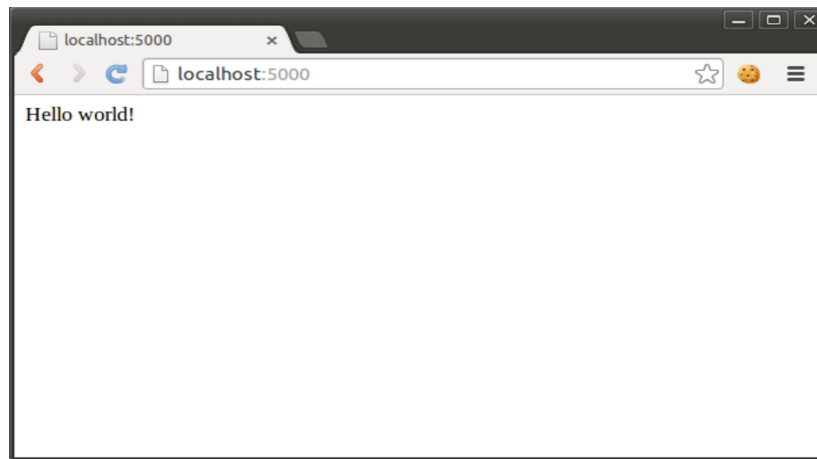


Figure 2: A browser visiting your development Flask website.

8. Go back to your terminal, and hit Ctrl-C.
9. You now have Flask installed for development!

2.3 Step 3 - Install Apache

This step outlines how to install Apache, a development-ready web server.

1. Open a terminal, or use an already open terminal.
2. Within the terminal, type the following command:

```
sudo apt-get install apache2 libapache2-mod-wsgi
sudo /etc/init.d/apache2 restart
```

You may be asked for your password. If so, type it in. The characters you type for your password will not display - but rest assured that they are being kept track of. You may also be asked whether or not you are sure. When that prompt appears, simply hit the Enter key to accept.

3. Open a web browser, and navigate to `http://localhost/`. You should see something like the following:

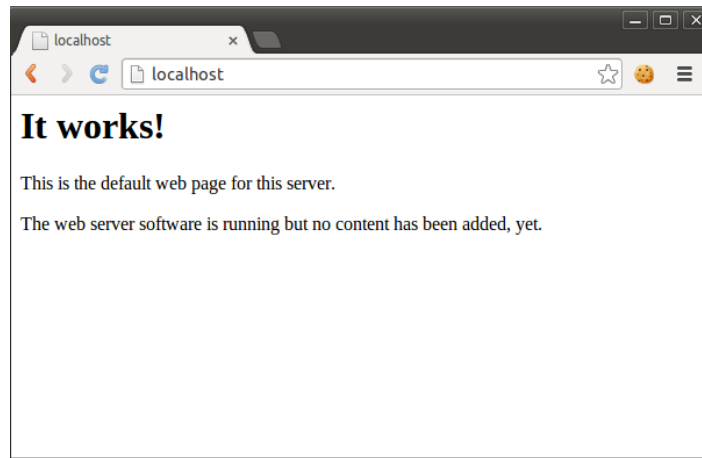


Figure 3: A browser visiting a default Apache install.

4. Within your terminal, type the command `sudo /etc/init.d/apache2 stop`.
5. You should now have Apache installed, though it will not yet be running.

2.4 Step 4 - Setup a Flask app in Apache

This step outlines how to create a Flask app that Apache will run for you.

1. Open a terminal, or use an already open terminal.
2. Type `sudo su` into your terminal, and provide your password. Your terminal now belongs to the `root` user - be very careful with the rest of your changes from now on.
3. Type `cd /var/www/` into your terminal. Now, type `mkdir app`; `cd app`.
4. Once you are safely in the `app` folder, your terminal should look like the following:

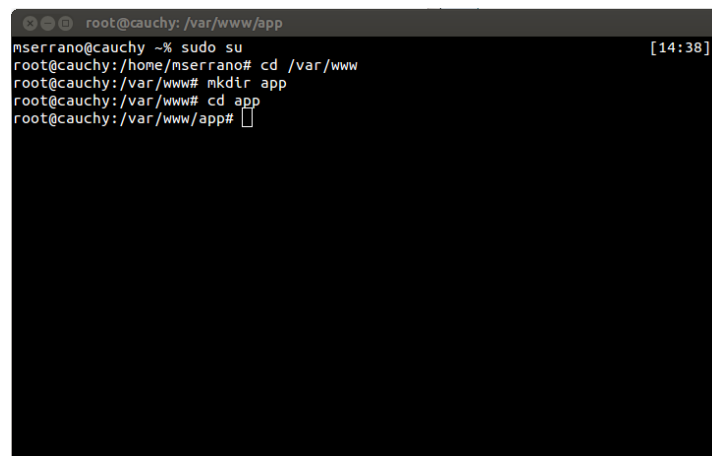


Figure 4: A root terminal in the app directory.

5. Open your favorite editor, and type the following into it:

```
import sys
sys.path.append('/var/www/app')
from website import app as application
```

6. Save and close the file as `application.wsgi`.
7. Open your favorite editor, and type the following into it:

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

@app.route("/")
def hello():
    return "Hello world, from Apache!"
```

8. Save and close the file as `website.py`.

2.5 Step 5 - Configure Apache to see the Flask App

This step outlines how to inform Apache of how to run your Flask app.

1. Type `cd /etc/apache2` into your root terminal.
2. Open the file `sites-available/default` with your favorite editor, and replace its contents with:

```
<VirtualHost *:80>
ServerName localhost
    WSGIScriptAlias / /var/www/app/application.wsgi
    <Directory /var/www/app>
        Order deny,allow
        Allow from all
    </Directory>
</VirtualHost>
```

3. Save and close the file.
4. Type `/etc/init.d/apache2 restart` into your terminal.
5. Open your web browser, and navigate to `http://localhost/`. You should see something like the contents of Figure 5 below.
6. Close your root terminal.
7. Congratulations! You now have Flask deployed in a production environment. When you make changes, simply edit the file `/var/www/app/website.py`, and type `sudo /etc/init.d/apache2 restart` to have the changes reflected by Apache.

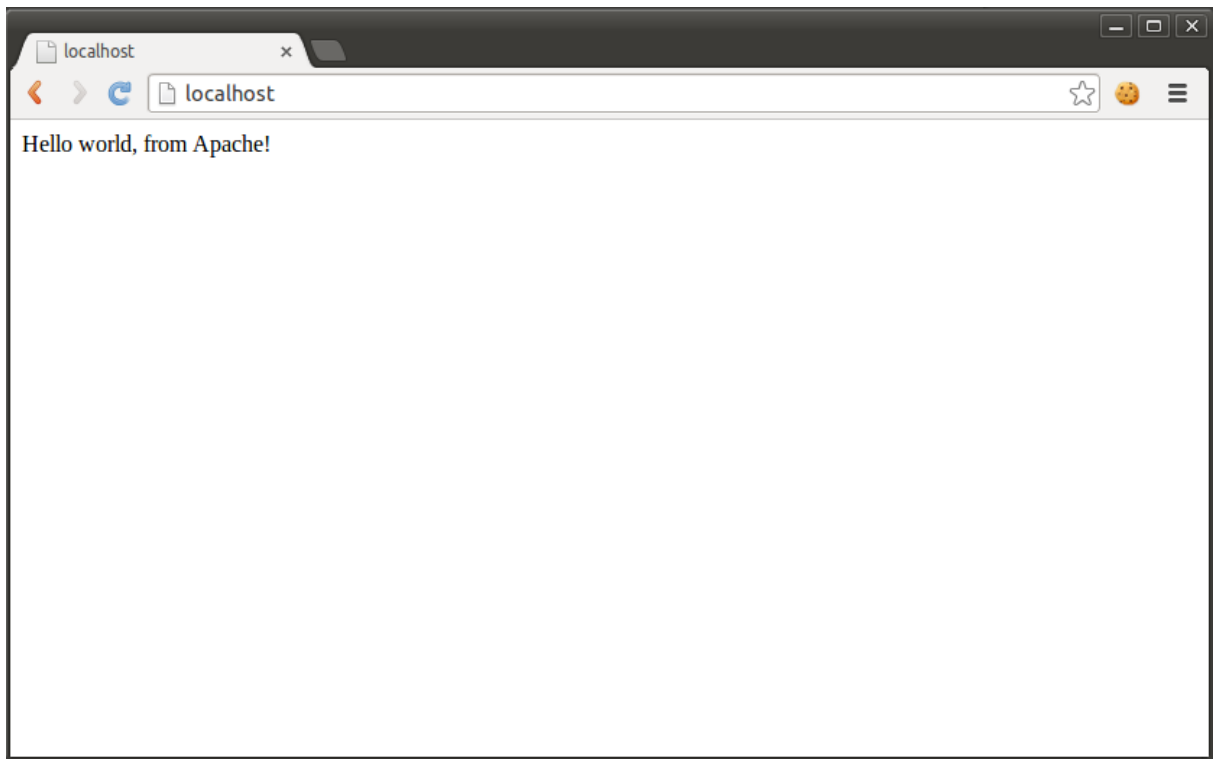


Figure 5: A web browser visiting your production Flask application.