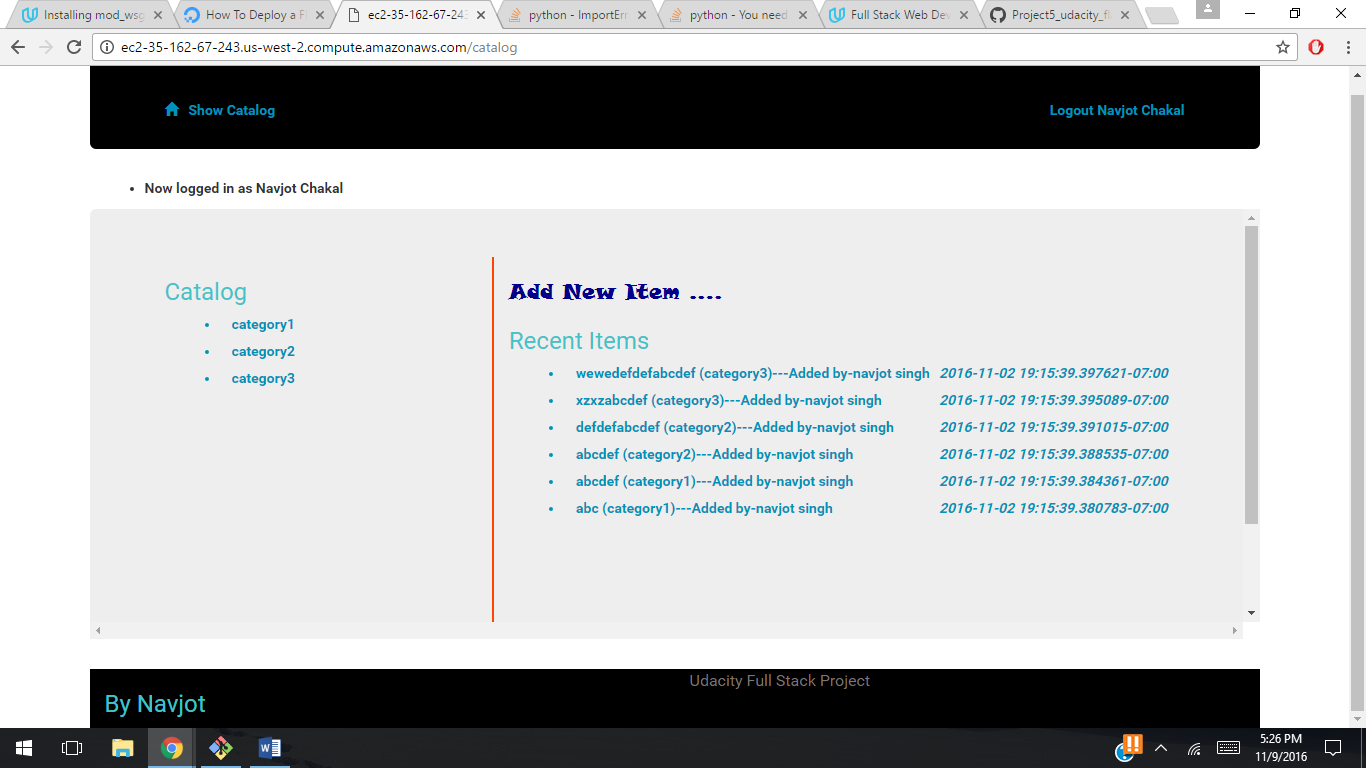
This project is about hosting a web application on virtual machine. This project includes steps to configure Linux distribution on virtual machine. I have used Amazon AWS instance to host the web application. The web application used here is project Item catalog developed using Flask framework and ORM, which was project 5 of Full Stack nanodegree. The project can be found on my github account. Below is screen shot of deployed project using aws instance.



This project is reachable under: <http://ec2-35-162-67-243.us-west-2.compute.amazonaws.com/>. This AWS instance was provided free by Udacity. And it may expire soon. But one can follow same steps to configure own virtual machine.

About Project

The Item Catalog project allows to edit/delete/update an item in particular category. The same framework can be used for creating restaurant menu, movie ticket application etc. For authorization, I have used oAuth2.0 authentication with facebook and google+. The original project uses sqlite database which is achieved through sqlAlchemy.

For deployment purpose, I will be using postgresql and apache2 web server.

Steps

1. **Launch your Virtual Machine with your [Udacity account](https://www.udacity.com/account" \l "!/development_environment" \t "_blank).**

* Get new public ip address from Udacity Account and download the private key.
* Move the private key into ~/.ssh folder. (if old key with same name exist, then rename or move key to other folder).
* Change owner’s permission to 700 using

$ chmod 700 ~/.ssh/udacity\_key.rsa

1. **SSH into your server**

* Ssh into AWS instance:

$ ssh –i ~/.ssh/udacity\_key.rsa root@public-ip-address

1. **Create a new user named grader**

* Now as we are login into remote machine i.e. the virtual machine that needs to be configured. Therefore we will be adding new user as:

$ adduser grader

1. **Give the grader the permission to sudo**

* Type and enter command:

$ visudo

* At the end of file, append following line:

grader ALL=(ALL:ALL) ALL

Save changes and exit

1. **Update all currently installed packages**

* Get list of packages to be upgrader:

$ sudo apt-get update

* Install all packages:

$ sudo apt-get upgrade

1. **Change the SSH port from 22 to 2200**

* By default, ssh is configured to port 22. One can change the port using:

$ vi /etc/ssh/sshd\_config

* + Change port 22 to 2200
  + Change password Authentication to yes
  + Append line : ‘Allowusers grader’
* Restart ssh service:

$ sudo service ssh restart

* Now exit the remote connection using command ‘exit’
* Reconnect to remote machine on port ‘2200’ and using user ‘grader’ using

$ ssh grader@public-ip-address –p 2200

* Creating ssh key-pair

One have to generate key at local machine, so using command ‘exit’, one can exit remote machine. Now enter the following command to generate key-pair:

$ ssh-keygen  
One can save key with any name, at any location. Also user can enter paraphrase for further security on prompt. The above command will generate two files: one with .pub extension which is public file and needs to be placed at server, and other file is private file which is used for connection with server if key based authentication is enabled.

* Login to remote machine as grader and create .ssh directory at home.

$ mkdir ~/.ssh

Create a file name: authorized\_keys in this folder and copy the content of public key in this file.

* Now change the permission of the folder .ssh to 700 and .ssh/authorized\_keys to 644.
* To enable key based authentication, open the file /etc/ssh/sshd\_config, and change PasswordAuthentication from No to Yes.
* To simplify Login from local machine, one can edit file ~/.ssh/config and add following line :

Host graderLogin

HostName 35.162.67.243

Port 2200

User grader

IdentityFile ~/.ssh/id\_rsa

Now to login into remote machine, one can simply type ‘ssh graderLogin’ and enter.

1. **Configure the Uncomplicated Firewall (UFW) to only allow incoming connections for SSH (port 2200), HTTP (port 80), and NTP (port 123**)

* First, check ufw status and if it is enabled, then disable it.

$ sudo ufw status

* Deny All Incoming conenctions

$ sudo ufw default deny incoming

* Allow tcp at port 80, ntp at port 123.

$ sudo ufw allow 2200/tcp

$ sudo ufw allow 80/tcp

$ sudo ufw allow 123/udp

* Enable the firewall

$ sudo ufw enable

1. **Configure the local timezone to UTC**

Type the following command and then select the time zone.

$ sudo dpkg-reconfigure tzdata

1. **Install and configure Apache to serve a Python mod\_wsgi application**

* Install Apache2

$ sudo apt-get install apache2

To check its working or not, open the public-address-ip on web browser.

Apache just returns a file requested or the index.html file if no file is defined within the URL.

* To configure Apache to hand-off certain requests to an application handler - mod\_wsgi. The first step in this process is to install mod\_wsgi:

$ sudo apt-get install libapache2-mod-wsgi python-dev

* To configure Apache to handle requests using the WSGI module. You’ll do this by editing the /etc/apache2/sites-enabled/000-default.conf file. This file tells Apache how to respond to requests, where to find the files for a particular site and much more.
* Restart Apache: $ sudo apache2ctl restart

1. **Install git and setup your project:**

* Install git

$ sudo apt-get install git

* Set your name and email id:

$ git config --global user.name "YOUR NAME"

$ git config --global user.email "YOUR EMAIL ADDRESS"

* 1. **Deploying Flask application on Ubuntu**
* Enable mod\_wsgi

$ sudo a2enmod wsgi

* Create the web application

$ cd /var/www

$ sudo mkdir catalog

$ cd catalog and $ sudo mkdir catalog

$ cd catalog and $ sudo mkdir static templates

$ sudo vi \_\_init\_\_.py

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route("/")

def hello():

return "Check: Successful"

if \_\_name\_\_ == "\_\_main\_\_":

app.run()

* Install FlasK

$ sudo apt-get install python-pip

$ sudo pip install virtualenv

$ sudo virtualenv venv

$ sudo chmod -R 777 venv

$ source venv/bin/activate

$ pip install Flask

$ pip install Flask-SQLAlchemy

$ sudo apt-get install libpq-dev python-dev

$ pip install psycopg2

* Run the application and check on localhost

python \_\_init\_\_.py

* Deactivate
* Configure and Enable a New Virtual Host

$ sudo vi /etc/apache2/sites-available/catalog.conf

<VirtualHost \*:80>

ServerName PUBLIC-IP-ADDRESS

ServerAdmin admin@PUBLIC-IP-ADDRESS

WSGIScriptAlias / /var/www/catalog/catalog.wsgi

<Directory /var/www/catalog/catalog/>

Order allow,deny

Allow from all

</Directory>

Alias /static /var/www/catalog/catalog/static

<Directory /var/www/catalog/catalog/static/>

Order allow,deny

Allow from all

</Directory>

ErrorLog ${APACHE\_LOG\_DIR}/error.log

LogLevel warn

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

* Enable virtual host

$ sudo a2ensite catalog

* Create the .wsgi File and Restart Apache

$ cd /var/www/catalog and $ sudo vim catalog.wsgi

#!/usr/bin/python

import sys

import logging

logging.basicConfig(stream=sys.stderr)

sys.path.insert(0,"/var/www/catalog/")

from catalog import app as application

application.secret\_key = 'Add your secret key'

* Restart Apache:  
  $ sudo service apache2 restart
  1. **Clone GitHub repository and make it web inaccessible**
* Clone project itemCatalog from github at localtion /var/www/catalog/catalog

Git clone <https://github.com/navjot77/Project5_udacity_flask/tree/master/project>

* Make github repo accessible

1. Create and open .htaccess file in directory /var/www/catalog.
2. Paste in the following:  
   RedirectMatch 404 /\.git

**10.3** **Install packages:**

* Activate virtual environment venv:  
  $ source venv/bin/activate
* Install httplib2 module in venv:  
  $ pip install httplib2
* Install requests module in venv:  
  $ pip install requests
* Install oauth2client.client:  
  $ sudo pip install --upgrade oauth2client
* Install SQLAlchemy:

$ pip install Flask-SQLAlchemy

* Install the Python PostgreSQL adapter psycopg:

$ pip install psycopg2

1. **Install and configure PostgreSQL:**
2. Install PostgreSQL:  
   $ sudo apt-get install postgresql postgresql-contrib
3. Open the database setup file and change the engine settings for postgresql:  
   $ sudo vim database\_setup.py  
   python engine = create\_engine('postgresql://catalog:PW-FOR-DB@localhost/catalog')
4. Change the same line in application.py and lotsofmeny.py respectively
5. Rename application.py:  
   $ mv application.py \_\_init\_\_.py
6. Create needed linux user for psql:  
   $ sudo adduser catalog
7. Change to default user postgres:  
   $ sudo su - postgre
8. Connect to the system:  
   $ psql
9. Add postgre user with password:
   * Create user with LOGIN role and set a password:  
     # CREATE USER catalog WITH PASSWORD 'PW-FOR-DB'; (# stands for the command prompt in psql and end the command with ;)
   * Allow the user to create database tables:  
     # ALTER USER catalog CREATEDB;
10. Create database:  
    # CREATE DATABASE catalog WITH OWNER catalog;
11. Connect to the database catalog # \c catalog
12. Revoke all rights:  
    # REVOKE ALL ON SCHEMA public FROM public;
13. Grant only access to the catalog role:  
    # GRANT ALL ON SCHEMA public TO catalog;
14. Exit out of PostgreSQl and the postgres user:  
    # \q, then $ exit
15. Create postgreSQL database schema:  
    $ python database\_setup.py
16. Create dummy menu catalog descriptions:

$ python lotsofmenu.py

1. **Run Application**

$ sudo service apache2 restart

1. **Get oAuth working**

* Get host name for your public-ip-address at  <http://www.hcidata.info/host2ip.cgi>
* Open file /etc/apache2/sites-available/catalog.conf and add line below ServerAdmin:

ServerAlias HOSTNAME ec2-35-162-67-243.us-west-2.compute.amazonaws.com

* Enable virtual host

$ sudo a2ensite catalog

* Vist google/facebook developers page and add the url of web page to redirect URIs and authorized java script origins.

1. **References**

* <https://www.digitalocean.com/community/tutorials/how-to-add-and-delete-users-on-an-ubuntu-14-04-vps>
* <http://askubuntu.com/questions/16650/create-a-new-ssh-user-on-ubuntu-server>
* <https://help.ubuntu.com/community/UFW>
* <https://www.digitalocean.com/community/tutorials/how-to-deploy-a-flask-application-on-an-ubuntu-vps>
* <https://www.digitalocean.com/community/tutorials/how-to-secure-postgresql-on-an-ubuntu-vps>
* <http://killtheyak.com/use-postgresql-with-django-flask/>
* <http://blog.trackets.com/2013/08/19/postgresql-basics-by-example.html>
* <http://stackoverflow.com/questions/10572498/importerror-no-module-named-sqlalchemy>
* <http://stackoverflow.com/questions/28253681/you-need-to-install-postgresql-server-dev-x-y-for-building-a-server-side-extensi>