Michele Novack Abugosh

Project : Linux Server Configuration

This project call to setup and configure a Linux (Ubuntu) web server using Amazon AWS. The server must be secure and serve and application previously developed in the course (Catalog) . Prepare it to host your web application to include installing updates, securing it from attacks.

Table of Contents:

|  |  |
| --- | --- |
| **Name** | **Value** |
| **IP Address** | 3.84.147.247 |
| **SSH Port** | **2200** |
| **Username** | **grader** |
| **URL of Applicaiton** |  |

ec2-3-84-147-247.compute-1.amazonaws.com (3.84.147.247)

To connect to EC2 instance you need the Lightsail\_key.rsa ( supplied separately in the submit process)

|  |
| --- |
| ssh -i Lightsail\_key.rsa ubuntu@3.84.147.247 |

Launch your EC2 Terminal , Then access the EC2 instance using the following command:

|  |
| --- |
| ssh -i Lightsail\_key.rsa grader@3.84.147.247 |

Check for updates:

|  |
| --- |
| sudo apt-get update  sudo apt-get upgrade  sudo apt-get dist-upgrade |

Create New User: Create a new user named grader and grant this user sudo access permissions

|  |
| --- |
| sudo adduser grader |

Followed the instructions in command line and added a secure password. After that I GRANTED (sudo) permission to the grader by editing the /etc/sudoers.d/ directory

|  |
| --- |
| Sudo nano /etc/sudoers.d/grader  grader ALL-(ALL:ALL)ALL - close and save |

Update All Currently installed applications:

|  |
| --- |
| Sudo apt-get update  Sudo apt-get upgrade |

Configure the local timezone to UTC: select OK for more option and select UTC save

<http://askubuntu.com/questions/138423/how-do-i-change-my-timezone-to-utc-gmt/138442>

|  |
| --- |
| Sudo dpkg-reconfigure tzdata |

Set time to sync with NTP and add additional servers to the /etc/ntp.config file

|  |
| --- |
| Sudo apt-get install ntp |

To add additional servers:

|  |
| --- |
| Server ntp.ubuntu.com  Server pool.ntp.org |

Restart NTP service

|  |
| --- |
| Sudo service ntp reload |

ADD Python environment

|  |
| --- |
| Sudo apt-get install python-psycopg2  Sudo apt-get python-flask python-sqlalchemy  Sudo apt-get install python-pip |

Then Log in temporarily into the grader :

|  |
| --- |
| Sudo -su grader |

Securing the Server:

Added a Key based login to new grader user:

|  |
| --- |
| su - grader |

Then added directory .ssh

|  |
| --- |
| mkdir .ssh |

Added file .ssh/authorized\_keys and copy the public key contents of the Lightsail\_key to authorized\_keys and finally restrict permission to the .shh and authorized\_key

|  |
| --- |
| Chmod 700 .ssh  Chmod 644 .ssh/authorized\_keys |

Only allow Key Based Authentication

To force key based authentication edit the /etc/ssh/sshd\_config file

|  |
| --- |
| sudo nano /etc/ssh/sshd\_config |

Update the following field # Change to no to disable tunnelled clear text passwords

|  |
| --- |
| PasswordAuthentication yes |

TO :

|  |  |
| --- | --- |
| **PasswordAuthentication NO** |  |

Then, restart the ssh service

|  |
| --- |
| Sudo service ssh restart |

The SSH is hosted on non-default port - Change the Default to host on Port 2200 - edit the /etc/ssh/sshd\_config file

|  |
| --- |
| Port 22 |

TO

|  |
| --- |
| Port 2200 |

The Restart the Service:

|  |
| --- |
| Sudo service ssh restart |

**# in AWS Lightsail update the Security Group , add port 2200 as the inbound custom TCP rule port**

Configure the Firewall (UFW) to only allow incoming connect for SSH(port 2200), HTTP (port 80) and NTP (port 123)

To set up the UFE , first check the firewall status: it should show inactive

|  |
| --- |
| Sudo ufw status |

Then, DENY incoming Traffic

|  |
| --- |
| Sudo ufw default deny incoming |

Then ALLOW outgoing traffic

|  |
| --- |
| Sudo ufw allow outgoing |

Establish the rules for SSH (port 2200) –

|  |
| --- |
| Sudo ufw allow 2200/tcp |

For HTTP (port 80)

|  |
| --- |
| Sudo ufw allow www |

And for NTP (port 123)

|  |
| --- |
| Sudo ufw allow ntp |

Then ENABLE UFW

|  |
| --- |
| Sudo ufw enable |

#

DISABLE remote login of the root user. Edit the /etc/ssh/sshd\_config file and update the following:

|  |
| --- |
| Sudo nano /etc/ssh/sshd\_config |

Edit the following

|  |
| --- |
| PermitRootLogin with-out-password |

TO

|  |
| --- |
| PermitRootLogin NO |

Then Restart the SHH server:

|  |
| --- |
| Sudo service ssh restart |

Confirm that root can SSH and login from local computer, ssh -i ~/.ssh/udacity\_key.rsa -p 2200 root@AWS\_IP\_ADDRESS If yes, then proceed. If not, repeat the steps above since you are locked out of the server.

**Now you are able to log in using : ssh -I Lightsail\_key.rsa -p 2200 grader@3.84.147.247**

**Additional Security**

Add feature to protect against attackers, I installed fail2ban a software package that blocks up address with multiple failed login attempts within a certain about of time

To install fail2ban :

|  |
| --- |
| Sudo apt-get update  Sudo apt-get fail2ban |

Configuring fail2ban was simple, the default configurations settings were adequate. I had to copy the default configuration file to new file to prevent updates from overwriting my configurations setting:

|  |
| --- |
| Sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local |

Then with the new jail.local file I changed the ssh port from 22 to 2200

|  |
| --- |
| Sudo nano /etc/fail2ban/jail.local |

Then restart fail2ban service

|  |
| --- |
| Sudo service fail2ban restart |

Reference Documents : [DigitalOcean](https://www.digitalocean.com/community/tutorials/how-to-protect-ssh-with-fail2ban-on-ubuntu-14-04), [Reddit](https://www.reddit.com/r/linuxadmin/comments/2lravs/fail2ban_does_not_detect_my_ssh_privatekey/).

**INSTALL PACKAGES FOR SERVER NEEDS**

**Apache2 HTTP Server**:

|  |
| --- |
| Sudo apt-get install apache2 |

**mod\_wsgi**

|  |
| --- |
| Sudo apt-get install libapache2-mod-wsgi  Sudo apt-get install python-setuptools libapache2-mod-wsgi |

Install and configure DEMO WSGI app by editing file /etc/apache2/sites-enabled/000-default.conf and have Apache to handle request using WSGI

|  |
| --- |
| Sudo nano /etc/apache2/sites-enable/000-default.conf |

Edit the following <Vitrualhost\*.80) block, right before closing add this line :

|  |
| --- |
| WSGIScriptAlias / /var/www/html/myapp.wsgi |

Add the following

|  |
| --- |
| Sudo nano /etc/apache2/apache2.conf |

INSERT ANWHERE: exit and save

|  |
| --- |
| ServerName localhost |

Enable mod\_wsgi

|  |
| --- |
| Sudo a2enmod wsgi |

**Update packages**

|  |
| --- |
| Sudo apt-get update |

Restart the Apache2 server:

|  |
| --- |
| Sudo service apache2 restart |

**# YOU WILL GET AN ERROR ON PAGE UNTIL WE RECONFIGURE Apache to Serve WSGI**

**Configure Apache to serve basic WSGI Application**

Create the following file /var/www/html/myapp.wsgi

|  |
| --- |
| Sudo nano /var/www/html/myapp.wsgi |

Within the file , write the following application

|  |
| --- |
| def application(environ, start\_response):  status = ‘200 OK’  output = ‘Hello World’  response\_headers = [(‘Content-type’. ‘text-plain’), (‘Content-length’ , str(len(output))]  start\_response(status, response\_headers)  return [output] |

Refresh the page and the text in the script above will displayed

**PostgreSQL**

Install PostgreSQL

|  |
| --- |
| Sudo apt-get install postgresql postgresql-contrib |

Check that remote connects are NOT Allowed:

|  |
| --- |
| Sudo less /etc/postgresql/9.3/main/pg\_hba.conf |

(by default remote connect to the database are disabled for security reason when installing PostgreSQL from Ubuntu repositories

Basic server setup:

|  |
| --- |
| Sudo -u postgres psql postgres |

Set-up password for the user postgres and enter a password : grader

|  |
| --- |
| \password postgres |

**Create a new database user name with limited permission to the database. Connect to database as the user postgres**

|  |
| --- |
| Sudo su - postgres |

To generate PostgreSQL prompt type:

|  |
| --- |
| psql |

**Create new User**

|  |
| --- |
| CREATE USER catalog2 WITH PASSWORD ‘your\_passwd’; |

Confirm user was created

|  |
| --- |
| du |

**Limit permission to new database user**

Run \du to see what permission the user catalog2 has

To see possible user roles type

|  |
| --- |
| \h CREATE ROLE |

Update permission for catalog2 user:

|  |
| --- |
| ALTER ROLE catalog2 WITH LOGIN;  ALTER USER catalot2 CREATEDB; |

Set password for user catalog2

|  |
| --- |
| ALTER ROLE catalog2 WITH PASSWD ‘password’; |

**Create a new database named catalog2**

|  |
| --- |
| CREATE DATABASE catalog2 WITH OWNER catalog2; |

Login to the database:

|  |
| --- |
| \c catalog2 |

Revoke all rights

|  |
| --- |
| REVOKE ALL ON SCHEMA public FROM public; |

Grant only access to the catalog role:

|  |
| --- |
| GRANT ALL ON SCHEMA public TO catalog2; |

Exit out of PostgreSQL and the postgres user

|  |
| --- |
| \q  exit |

Restart postgresql

|  |
| --- |
| Sudo service postgresql restart |

Reference Documentation:  <https://www.digitalocean.com/community/tutorials/how-to-secure-postgresql-on-an-ubuntu-vps>

<https://www.digitalocean.com/community/tutorials/how-to-secure-postgresql-on-an-ubuntu-vps>  
<https://help.ubuntu.com/community/PostgreSQL>

**GIT**

Install Git so that you can clone the Catalog2 app from github

|  |
| --- |
| Sudo apt-get install git |

**Clone Udacity Project 2 catalog2 app to the AWS server**

Create a folder inside the /var/www folder called “catalog2” and then CD into this folder(this is python flask app and not just html)

|  |
| --- |
| cd/var/www |

Make the directory and name it catalog2

|  |
| --- |
| sudo mkdir catalog2  cd catalog2 |

**Change the owner of the created direcrtory**

|  |
| --- |
| Sudo chown -R grader:grader catalog2 |

**Clone your project from github**

|  |
| --- |
| git clone <https://github.com/mxor111/catalog2.git> catalog2 |

**The project is now at /var/www/catalog2/catalog2**

**Make sure the git directory is not publicly accessible via a browser**

**At the root of the web directory , add .htaccess file and include this line:**

|  |
| --- |
| RedirectMatch 404 /\.git |

Reference Documentation: <http://stackoverflow.com/questions/6142437/make-git-directory-web-inaccessible>

**Flask -**

Set up a virtual environment to keep the application and it dependencies isolated from the main system. Create a Virtual Environment for Flask Applications . Using pip to install flask

Run this if you have not already install in your python environment:

|  |
| --- |
| Sudo apt-get install python-pip |

**Install Flask**

|  |
| --- |
| Sudo pip install flask |

Check to see if install was success:

|  |
| --- |
| Sudo python \_\_init\_\_.py |

It should display “Running on http://localhost:5000/” or "Running on http://127.0.0.1:5000/". If you see this message, you have successfully configured the app.

To deactivate environment : give following command

|  |
| --- |
| deactivate |

Configure and Enable Vitrual Host:

|  |
| --- |
| sudo nano /etc/apache2/sites-available/FlaskApp.conf |

**Add the following lines of code to the file to configure the virtual hose. Be sure to change the ServerName to your domain or cloud server’s IP address:**

|  |
| --- |
| <VirtualHost \*:80>  ServerName ec2-3.84.147.247.compute-1.amazonaws.com  ServerAdmin mishtay@yahoo.com  WSGIScriptAlias / /var/www/FlaskApp/flaskapp.wsgi  <Directory /var/www/FlaskApp/FlaskApp/>  Order allow,deny  Allow from all  </Directory>  Alias /static /var/www/FlaskApp/FlaskApp/static  <Directory /var/www/FlaskApp/FlaskApp/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 the Virtual Host:**

|  |
| --- |
| Sudo a2ensite FlaskApp |

**Reload the server**

|  |
| --- |
| Sudo service apache2 reload |

**Create the .wsgi file**

Create a new File under /var/www/FlaskApp

|  |
| --- |
| cd /var/www/Flaskapp  sudo nano flaskapp.wsgi |

Update the following content to /var/www/FlaskApp

#!/usr/bin/python

import sys

import logging

logging.basicConfig(stream=sys.stderr)

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

from FlaskApp import app as application

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

**Restart Apache**

|  |
| --- |
| sudo service apache2 restart |

Reference documentation: <https://www.digitalocean.com/community/tutorials/how-to-deploy-a-flask-application-on-an-ubuntu-vps>

**Install App Dependencies for Flask and Database**

|  |
| --- |
| sudo pip install Flask  sudo pip install sqlalchemy  sudo pip install Flask-SQLAlchemy  sudo pip install psycopg2  sudo pip install flask-seasurf  sudo pip install oauth2client  sudo pip install httplib2  sudo pip install requests |

**Update oAuth information for Google+ and Facebook Logins**

Go to Google Development console:

Click on Enable and Manage APIs, then click on Credentials in the left-hand menu

Select Catalog App

Add URLs to  
Authorized Javascript origins, both local URL and EC2 version, e.g. <http://3.84.147.247> and <http://ec2-3-84-147.247.us-east-1.compute.amazonaws.com/>

Authorized redirect URIs, <http://ec2-3-84-147.247.us-east-1.compute.amazonaws.com/login>

 and<http://ec2-3-84-147.247.us-east-1.compute.amazonaws.com/gconnect>

NOTE: Needed to restart Apache and Python app to get it all working

Downgrade packages to enable Google+ Login

|  |
| --- |
| Pip install werzeug==0.8.3  Pip install flask==0.9  Pip install Flask-Login==0.1.3 |

Go to Facebook Development console:

Click on Android Events App

Click on Settings and navigate to Valid oAuth redirect URIs section

Add URLS for local and EC@ instance then save

**Flask**

|  |
| --- |
| Sudo pip install flask |

**Create a directory for Flask App**

**In the home directory to work in and link to it from the site-root defined in apache’s configuration (/var/www/html) see /etc/apache2/sites-enabled/000-default.conf**

|  |
| --- |
| mkdir ~/flaskapp  sudo 1n -sT ~/flaskapp /var/www/html/flaskapp |

**You can check if this is working - by creating a Index.html file – then $cd ~/flaskapp echo “hello world” > index.html (you should see Hello World displayed of you navigate to you instance public DNS/flaskapp)**