**Linux Server Configuration**

**Project Overview**

The objective of this project is to have our web application, Item-Catalog run live on a secure web server. We take a baseline installation of a Linux server instance on a virtual machine, using Amazon Lightsail. We prepare it to host our web application by securing it from a number of attack vectors and by installing and configuring a web server and a database server. Finally, we deploy our existing web application on it.

**Link to Project:** **http://34.220.148.31/**

* **Public IP Address:** 34.220.148.31
* **Accessible SSH port:** 2200

**Steps to configure a secured Linux Server:**

**1. Create Development Environment Instance**

* Set up an Amazon Web Services account and start a new Ubuntu Linux Server Instance on Amazon Lightsail.
* Used putty to ssh into the instance as ubuntu user.
* Download private key provided and note down your public IP address.

**2. Launch VM and access SSH to the instance**

* Move the private key file into the folder ~/.ssh (~ 🡪 home directory)
  + *$ mv /(current\_private\_key\_address)/udacity\_key.rsa ~/.ssh/*
* Change the key permission so that only owner can read and write
  + *$ chmod 600 ~/.ssh/udacity\_key.rsa*
* SSH into the instance either using PUTTY and PUTTY Gen or local command line
  + *$ ssh -i ~/.ssh/id\_rsa username@public\_IP\_address -p 2200*

**3. Create New User - Grader**

* Add User grader
  + $ *sudo adduser grader*
* Give Sudo Access to grader

*$ sudo vim /etc/sudoers.d/grader* –add this line to the file - *grader ALL=(ALL:ALL) ALL*

**4. Configure the key-based authentication for grader user**

* Generate an encryption key on your local machine

Go to the directory where you want to save the key, and run the following command:  
*$ ssh-keygen -t rsa key\_name*

* Place the public key on the server that we want to use:  
  *$ ssh-copy-id grader@XX.XX.XX.XX -i (key\_name.pub)*
* Log into remote machine as grader  
   $ ssh -i udacity\_key.rsa grader@XX.XX.XX.XX
  + sudo su - grader
  + mkdir .ssh
  + touch .ssh/authorized\_keys
  + sudo chmod 700 .ssh
  + sudo chmod 600 .ssh/authorized\_keys
  + nano .ssh/authorized\_keys
  + Then paste the contents of the public key created on the local machine

**5. Change the SSH port from 22 to 2220, Enforce key-based authentication & Disable login for root user**

* + Open sshd\_config file  
    *sudo vim /etc/ssh/sshd\_config*
  + Change the following in the file:
    - Find the Port line and edit it to 2200.
    - Find the PasswordAuthentication line and edit it to no.
    - Find the PermitRootLogin line and edit it to no.
  + Save the file and run *sudo service ssh restart*

**6. Configure the Uncomplicated Firewall (UFW)**

* *$ sudo ufw default deny incoming*
* *$ sudo ufw default allow outgoing*
* *$ sudo ufw allow 2200/tcp*
* *$ sudo ufw allow www*
* *$ sudo ufw allow ntp*
* *$ sudo ufw enable*

**7. Change timezone to UTC and Fix language issues**

* + Change the timezone to UTC using following command

*$ sudo timedatectl set-timezone UTC.*

* + Fix language issues  
    *$ sudo update-locale LANG=en\_US.utf8 LANGUAGE=en\_US.utf8 LC\_ALL=en\_US.utf8*

**8. Update all currently installed packages**

* *$ sudo apt-get update.*
* *$ sudo apt-get upgrade.*

**9. Install and Configure Apache2, mod-wsgi and Git**

$ *sudo apt-get install apache2 libapache2-mod-wsgi git*

Enable mod\_wsgi:  
$ *sudo a2enmod wsgi*

**10. Install and configure PostgreSQL**

*$sudo apt-get install libpq-dev python-dev  
$sudo apt-get install postgresql postgresql-contrib  
$sudo su – postgres  
$psql*

Once in psql command line, do these steps:

*CREATE USER catalog WITH PASSWORD 'password';  
CREATE DATABASE catalog WITH OWNER catalog;  
\c catalog  
REVOKE ALL ON SCHEMA public FROM public;  
GRANT ALL ON SCHEMA public TO catalog;  
\q  
exit*

**Note:** In your catalog project you should change database engine to:  
engine = create\_engine('postgresql://catalog:password@localhost/catalog')

**11. Install Flask and other dependencies**

*$ sudo apt-get install python-pip  
 $ sudo pip install Flask  
 $ sudo pip install httplib2 oauth2client sqlalchemy psycopg2 sqlalchemy\_utils  
 $ sudo pip install requests*

**12. Cloning & Configuring the Catalog app from Github**

* Make a *catalog* named directory in */var/www  
  $ sudo mkdir /var/www/catalog*
* Change the owner of the directory *catalog  
  $ sudo chown -R grader:grader /var/www/catalog*
* Clone the **Item Catalog** to the catalog directory:  
  *$ git clone https://github.com/kalps08/Item-Catalog.git*
* Make a item\_catalog.wsgi file to serve the application over the mod\_wsgi. with content:  
  *$ touch item\_catalog.wsgi &&vim item\_catalog.wsgi*

import sys  
sys.stdout = sys.stderr  
sys.path.insert(0, "/var/www/catalog/")  
from app import app as application

* Inside *app.py, models.py & lotsofitems.py,*  database connection is now performed with:  
  *engine = create\_engine('postgresql://catalog:password@localhost/catalog')*
* Run the models.py and lotsofitems.py once to setup the database and populate it with test data.

$ python models.py  
$ python lotsofitems.py

**13. Configure Apache Server**

Open apache server config file and– add following content:  
*$ sudo vim /etc/apache2/sites-available/000-default.conf*

# serve catalog app  
<VirtualHost \*:80>  
 ServerName <IP\_Address or Domain>  
 ServerAdmin <Email>  
 DocumentRoot /var/www/catalog  
 WSGIDaemonProcess catalog user=grader group=grader  
 WSGIScriptAlias / /var/www/catalog/catalog.wsgi

<Directory /var/www/catalog>  
 WSGIProcessGroup catalog  
 WSGIApplicationGroup %{GLOBAL}  
 Require all granted  
 </Directory>

ErrorLog ${APACHE\_LOG\_DIR}/error.log  
 LogLevel warn  
 CustomLog ${APACHE\_LOG\_DIR}/access.log combined  
</VirtualHost>

**14. Restart Apache to launch the app** *$sudo service apache2 restart*

**Resources & References:**

* [Amazon EC2 Linux Instances](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html)
* [Flask mod\_wsgi (Apache)](http://flask.pocoo.org/docs/0.12/deploying/mod_wsgi/)
* [Apache Server Configuration Files](https://httpd.apache.org/docs/current/configuring.html)
* [Deploy a Flask Application on an Ubuntu VPS](https://www.digitalocean.com/community/tutorials/how-to-deploy-a-flask-application-on-an-ubuntu-vps)
* [Set Up Apache Virtual Hosts on Ubuntu](https://www.digitalocean.com/community/tutorials/how-to-set-up-apache-virtual-hosts-on-ubuntu-14-04-lts)
* [mod\_wsgi documentation](https://modwsgi.readthedocs.io/en/develop/)
* [Automatic Security Updates](https://help.ubuntu.com/community/AutomaticSecurityUpdates#Using_the_.22unattended-upgrades.22_package)
* [Ask Ubuntu](https://askubuntu.com/)
* [Stack Overflow](https://stackoverflow.com/)