The goal of this project is to take an Ubuntu server and configure it to securely host the catalog application from Project 3. What follows is step by step documentation of the process I went through in setting up my server as well as links to online documentation I found useful in the process.

Step 1 - Aquire the machine, complete initial login, update machine and install useful packages

AQUIRE MACHINE AT

https://www.udacity.com/course/viewer#!/c-nd004/l-3573679011/m-3620328723

SSH INTO VIRTUAL SERVER

```
local> ssh -i ~/.ssh/udacity_key.rsa root@SERVER_IP
```

UPDATE SERVER

```
# sudo apt-get update
# sudo apt-get upgrade
```

GET USEFUL PACKAGES

```
# sudo apt-get python-pip
# sudo apt-get finger
```

Step 2 - Add users, configure key access, remove root access

BASED ON DOCUMENTATION FROM:

https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-14-04

ADD USER

```
# adduser grader
```

ADD TO SUDO GROUP

```
# gpasswod -a grader sudo
```

GENERATE PUBLIC/PRIVATE KEYS

```
local> ssh-keygen
```

INSTALL PUBLIC KEY ON SERVER

```
local> cat ~/.ssh/id_rsa.pub
```

copy key to clipboard

LOGIN AS GRADER

```
# su - grader
```

CREATE DIRECTORY TO STORE KEYS AND SET ACCESS

```
$ mkdir .ssh
$ chmod 700 .ssh
```

CREATE FILE TO STORE KEYS AND PASTE IN KEY FROM CLIPBOARD

```
$ nano .ssh/authorized_keys
```

RESTRICT ACCESS TO THE AUTHORIZED_KEYS FILE

```
$ chmod 600 .ssh/authorized_keys
```

RETURN TO ROOT

```
$ exit
```

LOGIN AS GRADER

```
$ ssh grader@SERVER_IP
```

REMOVE ROOT LOGIN - EDIT DOCUMENT TO READ

```
$ sudo nano /etc/ssh/sshd_config

   # Authentication:
   LoginGraceTime 120
   PermitRootLogin no
   StrictModes yes
```

RESTART SSH

```
$ service ssh restart
```

TEST FOR CHANGE TO ROOT LOGIN

```
$ ssh -i ~/.ssh/udacity_key.rsa root@SERVER_IP
…Permission denied (publickey).
```

Step 3 Remove "sudo: unable to resolve host ip-10-20-57-3"

BASED ON DOCUMENTATION FROM:

http://askubuntu.com/questions/59458/error-message-when-i-run-sudo-unable-to-resolve-host-none

RUN ON SERVER

```
$ sudo nano /etc/hostname
```

EDIT DOCUMENT TO READ

```
   server
```

```
$ sudo nano /etc/hosts
```

EDIT DOCUMENT TO READ

```
   127.0.0.1 localhost server

   # The following lines are desirable for IPv6 capable hosts
```

Step 4 Update time zone and keep server time accurate

BASED ON DOCUMENTATION FROM:

http://askubuntu.com/questions/81293/what-is-the-command-to-update-time-and-date-from-internet

UPDATE LOCAL TIME

```
$ sudo dpkg-reconfigure tzdata
```

follow prompts to set time zone

KEEP TIME UPDATED

```
$ sudo apt-get install ntp
```

Step 5 Update software and configure for automatic updates

UPDATE LOCAL INSTALLED PACKAGES

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
```

KEEP IT UPDATED KEEP IT SAFE

```
$ sudo dpkg-reconfigure unattended-upgrades
…Creating config file /etc/apt/apt.conf.d/20auto-upgrades with new
version
```

Step 6 Secure SSH change for from 22 to 2200

BASED ON DOCUMENTATION FROM:

http://www.2daygeek.com/how-to-change-the-ssh-port-number/

EDIT CONFIG FILE

```
$ sudo nano /etc/ssh/sshd_config

   # Package generated configuration file
   # See the sshd_config(5) manpage for details

   # What ports, IPs and protocols we listen for
   Port 2200
   # Use these options to restrict which interfaces/protocols sshd will
   bind to
```

RESTART SERVICE

```
$ sudo service ssh restart
```

TIP - create ssh config on local machine (who wants to remember all this jargon?)

CREATE CONFIG FILE

```
local> nano ~/.ssh/config
   #contents of ~/.ssh/config
   Host HOST_NAME
      HostName SERVER_IP
      Port 2200
      User grader
```

NOW TO CONNECT SSH

```
local>ssh HOST_NAME
```

Step 7 Configure firewall

SET DEFAULTS

```
$ sudo ufw default deny incoming
$ sudo ufw default allow outgoing
```

OPEN PORTS NEEDED

```
$ sudo ufw allow 2200
$ sudo ufw allow 123
$ sudo ufw allow 80
```

ENABLE THE FIREWALL

```
$ sudo ufw enable
```

RESTART SERVICE

```
$ sudo service ufw restart
```

CHECK CONFIG

```
$ sudo ufw status
```

Step 8 Protect server from DOS attacks on SSH ports

BASED ON DOCUMENTATION FROM:

https://www.digitalocean.com/community/tutorials/how-to-protect-ssh-with-fail2ban-on-ubuntu-14-04

INSTALL FAIL2BAN

```
$ sudo apt-get install fail2ban
```

CREATE LOCAL VERSION, JAIL.CONF IS SUBJECT TO UPDATE CHANGES

```
$ sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

UPDATE ALERTS

```
$ sudo nano /etc/fail2ban/jail.local
```

```
# Destination email address used solely for the interpolations in
# jail.{conf,local} configuration files.
destemail = EMAIL_ADDRESS

#
# Name of the sender for mta actions
sendername = HOST_NAME.Fail2Ban

…
# ban & send an e-mail with whois report and relevant log lines
# to the destemail.
action_mwl = %(banaction)s[name=%(__name__)s, port="%(port)s",
protocol="%(protocol)s", chain="%(chain)s"]
               %(mta)s-whois-lines[name=%(__name__)s, dest="%
(destemail)s", logpath=%(logpath)s, chain="%(chain)s", sendername="%
(sendername)s"]

# Choose default action.  To change, just override value of 'action'
with the
# interpolation to the chosen action shortcut (e.g.  action_mw,
action_mwl, etc) in jail.local
# globally (section [DEFAULT]) or per specific section
action = %(action_mwl)s
```

RESTART SERVICE

```
$ sudo service fail2ban restart
```

Step 9 Install Apache2 and PostgreSQL

BASED ON DOCUMENTATION FROM:

https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu

GET APACHE

```
$ sudo apt-get install apache2
```

verify install by visiting SERVER _IP Address

GET POSTGRESQL

BASED ON DOCUMENTATION FROM:

http://help.ubuntu.com/stable/serverguide/postgresql.html

```
$ sudo apt-get install postgresql
```

LOGIN TO POSTGRES

```
$ sudo -u postgres psql template1
```

CHANGE PASSWORD

```
template1=#ALTER USER postgres with encrypted password 'your
password';
```

CREATE DATABASE

```
template1=#CREATE DATABASE gearopedia;
```

CREATE CATALOG USER AND SETUP USER

```
template1=#CREATE USER catalog;
template1=#ALTER USER catalog with encrypted password 'udacity';
template1=#ALTER USER catalog NOCREATEUSER NOCREATEDB;
template1=#ALTER USER catalog VALID UNTIL 'infinity';
```

VERIFY USER

```
template1=#\du
template1=#\q
```

SETUP MD5 PASSWORD ENCRYPTION

```
$ sudo nano /etc/postgresql/9.3/main/pg_hba.conf
```

RESTART

```
$ sudo nano service postresql restart
```

EDIT USER ACCESS PROFILES TO USE MD5 ENCRYPTION AND RESTRICT DATABASE ACCESS FOR CATALOG USER

BASED ON DOCUMENTATION FOUND AT:

http://stackoverflow.com/questions/17443379/psql-fatal-peer-authentication-failed-for-user-dev

```
$ sudo nano /etc/postgresql/9.3/main/pg_hba.conf
```

```
# Database administrative login by Unix domain socket
local   all             postgres                                md5
# TYPE  DATABASE        USER            ADDRESS
METHOD

# "local" is for Unix domain socket connections only
#local   all             all                                     md5
local   gearopedia      catalog                                 md5
```

TRY LOGIN TO SQL WITH NEW USER

```
$ psql -U catalog -d database -W
```

Step 10 Download and install the catalog application

GET GIT

```
$ sudo apt-get install git
```

CLONE THE REPO

```
$ cd /var/www/
$ sudo git clone https://github.com/stevenmarr/gearopedia.git
```

ALLOW ACCESS TO FILES FOR UPLOADS

```
$ sudo chmod -R 777 files/
```

SETUP FLASK ENVIROMENT

BASED ON DOCUMENTATION FOUND AT:

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

INSTALL MOD_WSGI

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

ENABLE MOD_WSGI

```
$ sudo a2enmod wsgi
```

INSTALL POSTGRESQL DEPENDENCY

```
$ sudo apt-get install postgresql-server-dev-9.3
```

NAVIGATE TO APP DIRECTORY

```
$ cd /var/www/gearopedia
```

CREATE VIRTUAL ENV

DOWNLOAD

```
$ sudo pip install virtualenv
```

ENABLE

```
$ sudo virtualenv venv
```

INSTALL APP REQUIREMENTS

```
$ sudo pip install -r requirements.txt
$ deactivate
```

CREATE INSTANCE FOLDER AND CONFIG.PY (APP SECRETS)

```
$ sudo mkdir instance
$ sudo nano config.py
```

PASTE INTO CONFIG

```
#!/usr/bin/python
SECRET_KEY = 'SUPER_SECRET'
```

CONFIGURE VIRTUAL HOST

```
$ sudo nano /etc/apache2/sites-available/gearopedia.conf
```

PLACE INTO FILE

```
<VirtualHost *:80>
                ServerName SERVER_NAME
                ServerAdmin EMAIL_ADDRESS
                WSGIScriptAlias / /var/www/APP/APP.wsgi
                <Directory /var/www/APP/APP/>
                        Order allow,deny
                        Allow from all
                </Directory>
                Alias /static /var/www/APP/APP/static
                <Directory /var/www/APP/APP/static/>
                        Order allow,deny
                        Allow from all
```

```
                    </Directory>
                    ErrorLog ${APACHE_LOG_DIR}/error.log
                    LogLevel warn
                    CustomLog ${APACHE_LOG_DIR}/access.log combined
    </VirtualHost>
```

VISIT APPLICATION AT HTTP://SERVER_NAME

TIP - Logs are your friends, access them via

```
$ sudo tail -f /var/log/apache2/error.log
$ sudo tail -f /var/log/apache2/access.log
```

Step 11 Secure Apache from attack

BASED ON DOCUMENTATION FOUND AT:

https://www.digitalocean.com/community/tutorials/how-to-protect-an-apache-server-with-fail2ban-on-ubuntu-14-04

OPEN THE CONFIG FILE

```
$ sudo nano /etc/fail2ban/jail.local
    #
    # HTTP servers
    #

    [apache]

    enabled  = true

    [apache-overflows]

    enabled  = true
```

RESTART THE SERVICE

```
$ sudo service fail2ban restart
```

Step 12 Sit back and enjoy - we are done!