

# Linux-Server-Configuration project Walkthrough

## Summary of Software/Package Installed

---

```
- apache2
- libapache2-mod-wsgi
- PostgreSQL
- git
- pip
- virtualenv
- flask
- sqlalchemy
- finger
- Apache2
- mod_wsgi
- httpplib2
- Python Requests
- oauth2client
- libpq-dev
- Psycopg2
```

## Step 1

---

### Setting Up Amazon Lightsail:

- Go to the Amazon Lightsail website
- **sign up** then enter your method payment info .. complete your registration from the message they'll send by email a window will show up asking for your contact info add your correct phone number, they will send you a code with 4 or 5 digit numbers, keep it.. answer their phone call .. enter the code number then they will inform you that your registration successfully activated
- Create your first instance click Create instance
- Choose an instance location (I choose Ohio zone A )
- Select Linux/Unix platform, OS Only and Ubuntu 16.04 LTS
- Choose an instance plan (I took the cheapest, \$3.5/month).
- Keep the default name provided by AWS or rename your instance.
- Click create then wait for the instance to start up.

## Step 2

---

## SSH into the server:

- Click on your instance once running then scroll down and click Account Page at the bottom
- Click on SSH keys tab and download the Default Private Key.
- Move this private key file named LightsailDefaultPrivateKey-\*.pem into the local folder ~/.ssh and rename it lightsail\_key.pem.
- To make our key secure type: `chmod 600 ~/.ssh/lightsail_key.pem` In your terminal,.
- To connect to the instance via the terminal: `ssh -i ~/.ssh/lightsail_key.pem ubuntu@00.00.00.00`, where 00.00.00.00 is the public IP address of the instance.
- you will log into the server as the user ubuntu@00.00.00.00, switch to the root user by typing `sudo su -`

## Step 3

### Secure the server

- ✓ Update and upgrade installed packages:

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

- ✓ Updated packages to most recent versions:

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

- Logged back in, and now I see this message:

```
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-1074-aws x86_64)
```

```
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage
```

```
Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud
```

```
0 packages can be updated.
0 updates are security updates.
```

```
New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
```

```
Last login: Wed Jan  9 18:58:37 2019 from 000.000.0.00
ubuntu@ip-00-00-00-00:~$
```

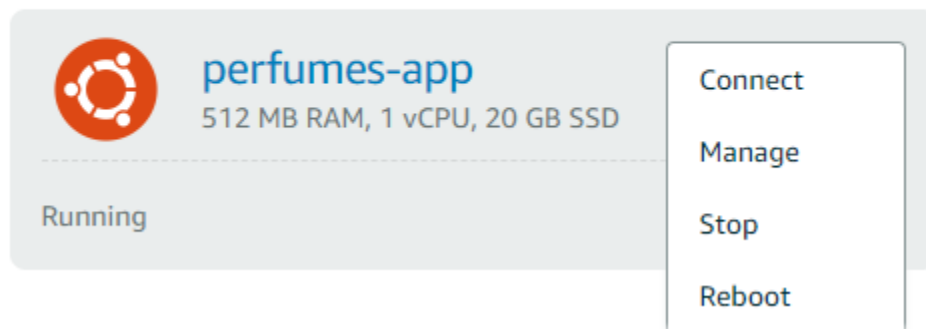
## Step 4

### Changing the SSH port from 22 to 2200

#### on SSH Server

- Edit the `/etc/ssh/sshd_config` file: Type `sudo nano /etc/ssh/sshd_config`.
- Change the port number on line 5 from 22 to 2200.
- Save and exit using CTRL+X and confirm with Y.
- Restart SSH: `sudo service ssh restart`.
- **on Amazon Lightsail firewall**
- Click on the Manage option of the Amazon Lightsail Instance, go to Networking tab, and then change the firewall configuration to match the internal firewall settings above.

Sort by **Date** ▼



- Allow ports 80(TCP), 123(UDP), and 2200(TCP), and deny the default port 22.

#### Firewall ?

You can control which ports on this instance accept connections.

Application	Protocol	Port range
HTTP	TCP	80
Custom	UDP	123
Custom	TCP	2200

[+ Add another](#)

[Edit rules](#) 

From your local terminal, run: `ssh -i ~/.ssh/lightsail_key.pem -p 2200 ubuntu@00.00.00.00`, where `00.00.00.00` is the public IP address of the instance.

## Step 5

---

### Configure the Uncomplicated Firewall (UFW)

- Configure the default firewall for Ubuntu to only allow incoming connections for SSH (port 2200), HTTP (port 80), and NTP (port 123).

```
sudo ufw status          # The UFW should be inactive.
sudo ufw default deny incoming # Deny any incoming traffic.
sudo ufw default allow outgoing # Enable outgoing traffic.
sudo ufw allow 2200/tcp      # Allow incoming tcp packets on port 2200.
sudo ufw allow www          # Allow HTTP traffic in.
sudo ufw allow 123/udp      # Allow incoming udp packets on port 123.
sudo ufw deny 22            # Deny tcp and udp packets on port 53.
```

- Turn UFW on: `sudo ufw enable`. The output should be like this:

```
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```

- Check the status of UFW to list current roles: `sudo ufw status`. The output should be like this:

```
Status: active

To Action From
--
2200/tcp ALLOW Anywhere
80/tcp ALLOW Anywhere
123/udp ALLOW Anywhere
22 DENY Anywhere
2200/tcp (v6) ALLOW Anywhere (v6)
80/tcp (v6) ALLOW Anywhere (v6)
123/udp (v6) ALLOW Anywhere (v6)
22 (v6) DENY Anywhere (v6)
```

- Exit the SSH connection: `exit`.

## Step 6

---

## Creating a new user account named grader

Create a new user account named `grader`

- While logged in as `ubuntu`, add user: `sudo adduser grader`.
- Enter a password (`your_password`) and fill out information for this new user.

## Step 7

---

### Giving grader access

Give `grader` the permission to `sudo`

- Edits the `sudoers` file: `sudo visudo`.
- Search for the line that looks like this:

```
root    ALL=(ALL:ALL) ALL
```

Below this line, add a new line to give `sudo` privileges to `grader` user.

```
root    ALL=(ALL:ALL) ALL
grader  ALL=(ALL:ALL) ALL
```

- Save and exit using `CTRL+X` and confirm with `Y`.
- Verify that `grader` has `sudo` permissions. Run `su - grader`, enter the password, run `sudo -l` and enter the password again. The output should be like this:

```
ubuntu@ip-10-10-10-13:~$ su - grader
Password:
grader@ip-10-10-10-13:~$ sudo -l
[sudo] password for grader:
Matching Defaults entries for grader on
ip-10-10-10-13.us-east-2.compute.internal:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User grader may run the following commands on
ip-10-10-10-13.us-east-2.compute.internal:
    (ALL : ALL) ALL
grader@ip-10-10-10-13:~$ |
```

- install **Finger** to see the users on this server.

```
$ sudo apt-get install finger.
```

## Step 8

---

### Creating an SSH key-gen

Create an SSH key pair for grader using the ssh-keygen tool

- On new local machine terminal run the command:
  - Run `ssh-keygen`
  - Enter file in which to save the key in the local directory `~/.ssh/grader_key`
  - Enter a passphrase (hebaahmad). Two files will be generated ( `~/.ssh/grader_key` and `~/.ssh/grader_key.pub` )
  - Run `cat ~/.ssh/grader_key.pub` and copy the **public key** of the file
  - Log in to the grader's virtual machine, run: `ssh -i ~/.ssh/lightstail_key.pem -p 2200 ubuntu@00.00.00.00` , where `00.00.00.00` is the public IP address of the instance.
- On the grader's virtual machine:
  - Back in the server terminal locate the folder for the user grader, it should be `/home/grader` . Run the command `$ cd /home/grader` to move to the folder.
  - Create a new directory called .ssh `mkdir .ssh`
  - Create a file to store the public key with the command `touch .ssh/authorized_keys`
  - Edit that file using `sudo nano .ssh/authorized_keys` and paste in the **public key** into this file, save and exit
  - change the permissions of the folder and its file by running `sudo chmod 700 /home/grader/.ssh` and `udo chmod 644 /home/grader/.ssh/authorized_keys`
  - Change the owner of the .ssh directory from root to grader by using the command `sudo chown -R grader:grader /home/grader/.ssh`
  - Check in `/etc/ssh/sshd_config` file if **PasswordAuthentication** is set to **no**, and Disable ssh login for root user change **PermitRootLogin** to **no**.
  - Restart SSH: `sudo service ssh restart`
  - Disconnect from the server `ctrl + D`
- On the local machine, run: `ssh -i ~/.ssh/grader_key -p 2200 grader@00.00.00.00`.

### Prepare to deploy the project

## Step 9

---

## Configure the local timezone to UTC

- While logged in as grader, Check the timezone with the `date` command. This will display the current timezone after the time.

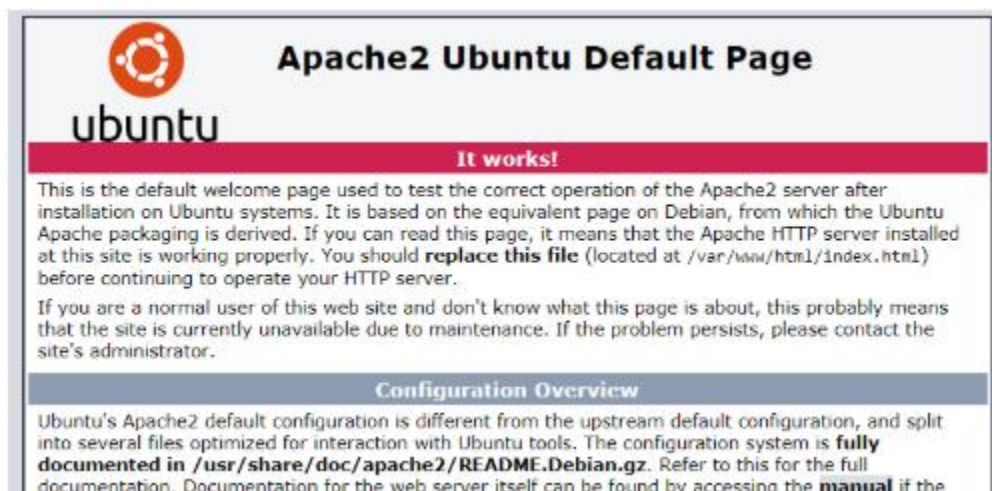
```
Wed Jan  9 22:11:15 UTC 2019
```

- If it's not UTC change it with this command: `sudo timedatectl set-timezone UTC`

## Step 10

### Install and configure Apache to serve and Python mod\_wsgi application

- While logged in as grader, install Apache:
- `sudo apt-get install apache2`
- Enter public IP of the Amazon Lightsail instance into browser. If Apache is working, you should see:



- My project is built with Python 2. So, I need to install the Python2 mod\_wsgi package:  
`sudo apt-get install libapache2-mod-wsgi python-dev.`

for python3 run the following command:

```
sudo apt-get install libapache2-mod-wsgi-py3
```

- Enable `mod_wsgi` using: `sudo a2enmod wsgi`.
- restart Apache `sudo service apache2 restart`

## Step 11:

### Install and configure PostgreSQL

- While logged in as grader, install PostgreSQL: `sudo apt-get install postgresql`.
- Check if no remote connections are allowed: run `sudo cat /etc/postgresql/9.5/main/pg_hba.conf` you should see:

```
# Database administrative login by Unix domain socket
local  all                postgres                                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only
local  all            all              peer
# IPv4 local connections:
host   all          all              127.0.0.1/32            md5
# IPv6 local connections:
host   all          all              ::1/128                  md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local  replication    postgres                                peer
#host   replication    postgres        127.0.0.1/32            md5
#host   replication    postgres        ::1/128                  md5
```

### Create a new PostgreSQL for catalog

- Switch to the postgres user: `sudo su - postgres`.
  - Open PostgreSQL interactive terminal with `psql`.
  - Create the catalog user with a password
- ```
postgres=# CREATE USER catalog WITH PASSWORD 'catalogdb';
CREATE ROLE
postgres=# ALTER USER catalog CREATEDB;
ALTER ROLE
postgres=# CREATE DATABASE catalog WITH OWNER catalog;
CREATE DATABASE
postgres=# \c catalog
You are now connected to database "catalog" as user "postgres".
catalog=# REVOKE ALL ON SCHEMA public FROM public;
```



```

REVOKE
catalog=# GRANT ALL ON SCHEMA public TO catalog;
GRANT
catalog=# \q

```

List the existing roles: `\du`. The output should be like this:

grader@ip-1

| List of roles |                                                            |           |
|---------------|------------------------------------------------------------|-----------|
| Role name     | Attributes                                                 | Member of |
| catalog       | Create DB                                                  | {}        |
| postgres      | Superuser, Create role, Create DB, Replication, Bypass RLS | {}        |

- Exit psql: `\q`.
- Run `psql` and then run `\l` to see that the new database has been created. The output should be like this:

List of databases

| Name      | Owner    | Encoding | Collate     | Ctype       | Access priv   |
|-----------|----------|----------|-------------|-------------|---------------|
| catalog   | catalog  | UTF8     | en_US.UTF-8 | en_US.UTF-8 |               |
| postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |               |
| template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres   |
| +         |          |          |             |             |               |
| postgres  |          |          |             |             | postgres=CTC/ |
| template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres   |
| +         |          |          |             |             |               |
| postgres  |          |          |             |             | postgres=CTC/ |

(4 rows)

- Exit psql: `\q`.
- Switch back to the grader user: `exit`.

## Step 12

### Install git

- While logged in as grader, install git: `sudo apt-get install git`.

## Step 13.1

## Clone and setup the Item Catalog project from the GitHub repository

- While logged in as grader, create a directory for our catalog application and make the user grader the owner.

```
$ cd /var/www
$ sudo mkdir catalog
$ sudo chown -R grader:grader catalog
$ cd catalog
```

- In this directory `catalog` we will have:

- our `catalog.wsgi` file `var/www/catalog/catalog.wsgi`,
- our `venv` virtual environment directory `/var/www/catalog/venv`,
- and `catalog` another directory which we'll sit inside of it our application `/var/www/catalog/catalog`.

➔ Cloning our Catalog Application repository by running the following:

- Make sure you are inside catalog directory `cd var/www/catalog/`
- clone the project `sudo git clone https://github.com/ your-project-name-on-github .git catalog`

➔ Create the `.wsgi` file by `$ sudo nano catalog.wsgi` and make sure your secret key matches with your project secret key:

```
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 = 'super_secret_key'
```

➔ Create our virtual environment, make sure you are in `/var/www/catalog`.

```
$ sudo pip install virtualenv
$ sudo virtualenv venv
$ source venv/bin/activate
$ sudo chmod -R 777 venv
```

(venv) will appear like this (venv)grader@ip00-00-00-00:/var/www/catalog\$ While our virtual environment is activated we need to install all packages required for our Flask application:

```
$ sudo apt-get install python-pip
$ sudo pip install flask
$ sudo pip install httplib2 oauth2client sqlalchemy psycopg2
$ sudo pip install requests
$ sudo pip install --upgrade oauth2client
$ sudo apt-get install libpq-dev
$ sudo pip install sqlalchemy_utils
```

- Deactivate the virtual environment by using the command : deactivate
- Change to the /var/www/catalog/catalog directory.
- Rename the application.py file to \_\_init\_\_.py using: mv application.py \_\_init\_\_.py.
- In \_\_init\_\_.py, change the following:

```
engine = create_engine("sqlite:///catalog.db")
engine = create_engine('postgresql://catalog:catalogdb@localhost/catalog')
```

the name and password of your database

Update path of client\_secrets.json file:

```
'/var/www/catalog/catalog/client_secrets.json'
```

And

```
app.run(host="0.0.0.0", port=8000, debug=True)
app.run()
```

- In database\_setup.py, and lotsofperfumes.py change the following by using the command \$ sudo nano database\_setup.py, \$ sudo nano lotsofperfumes.py :

```
engine = create_engine("sqlite:///catalog.db")
engine = create_engine('postgresql://catalog:catalogdb@localhost/catalog')
```

- Run the files by using python:

```
python database_setup.py
python lotsofperfumes.py
```

## Step 13.2:

### Authenticate login through Google

- Go to Google Cloud Platform: <https://console.cloud.google.com/>
- Click APIs & services on left menu.
- Click Credentials.
- Create an OAuth Client ID (under the Credentials tab), and add:
  - <http://00.00.00.00/> your public IP address and
  - <http://ec2-00-00-00-00.us-east-2.compute.amazonaws.com/> as authorized JavaScript origins.
  - Add <http://ec2-00-00-00-00.us-east-2.compute.amazonaws.com/oauth2callback> as authorized redirect URI.



## Configure and Enable our virtual host to run the site

```
sudo nano /etc/apache2/sites-available/catalog.conf
```

Paste in the following:

```
<VirtualHost *:80>
    ServerName 00.00.00.00 [Public IP]
    ServerAlias [Hostname]
    ServerAdmin admin@00.00.00.00
    WSGIDaemonProcess catalog python-
path=/var/www/catalog:/var/www/catalog/venv/lib/python2.7/site-packages
    WSGIProcessGroup catalog
    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>
```

If you need help finding your servers hostname use the following links and paste the IP address. Save and quit nano:

- <https://whatismyipaddress.com/ip-hostname>

- <http://www.nmonitoring.com/ip-to-domain-name.html>

## Useful commands

Below is a list of commands that could be useful while setting up the server.

- If at login the message **\*\*\* System restart required \*\*\*** is display, run the following command to reboot the machine:

```
```
$Sudo reboot
```
```

- Find out what version of Ubuntu is running

```
`lsb_release -a`
```

-Find out which user you are logged in as

```
`whoami`
```

```

- Find out which version of virtualenv is running
`virtualenv --version`

-Find out where Python has been installed
(Note: this is especially useful when making sure that the virtualenv is working
correctly; when the virtualenv is activated and set up correctly, running which
python should not return /usr/bin/python but rather the file path to the directory
where the virtualenv is located)
`which python`

-View Apache error logs, and open the file starting with the last line:
`sudo nano -f /var/log/apache2/error.log`.

- Delete a virtualenv and all of it's directories (be careful; this can't be undone):
`sudo rm -rf INSERT_NAME_OF_VIRTUALENV_HERE`

- if you made any change inside the database_setup.py or seeder.py files:
do the following before running the file again
delete database_setup.pyc inside catalog directory:
`sudo rm database_setup.pyc`
move to postgres: delete then recreate the database:
```
    postgres=# DROP DATABASE catalog;
    DROP DATABASE
    postgres=# CREATE DATABASE catalog WITH OWNER catalog;
    CREATE DATABASE
    postgres=# \q
```

- to stop and start Apache; stopping Apache breaks the database session, which makes
it possible to drop a database:
`sudo apachectl stop` and `sudo apachectl start`

- To get log messages from Apache server:
`sudo nano -f /var/log/apache2/error.log`.

- To restart Apache:
`sudo service apache2 restart`.

```

## Helpful Resources

- tutorialspoint tutorial on how to drop a PostgreSQL database:  
[https://www.tutorialspoint.com/postgresql/postgresql\\_drop\\_database.htm](https://www.tutorialspoint.com/postgresql/postgresql_drop_database.htm)
- Udacity course: [Configuring Linux Web Servers](https://classroom.udacity.com/courses/ud299-nd)  
<https://classroom.udacity.com/courses/ud299-nd>
- Udacity course: [Linux Command Line Basics](https://classroom.udacity.com/courses/ud595-nd)  
<https://classroom.udacity.com/courses/ud595-nd>
- DigitalOcean [How To Deploy a Flask Application on an Ubuntu VPS](https://www.digitalocean.com/community/tutorials/how-to-deploy-a-flask-application-on-an-ubuntu-vps)  
<https://www.digitalocean.com/community/tutorials/how-to-deploy-a-flask-application-on-an-ubuntu-vps>

- GitHub Repositories
  - <https://github.com/boisalai/udacity-linux-server-configuration>
  - <https://github.com/mulligan121/Udacity-Linux-Configuration>
  - <https://github.com/rjjoson/udacity-linux-server-configuration>

Thanks and Regards 😊

Heba Saada...