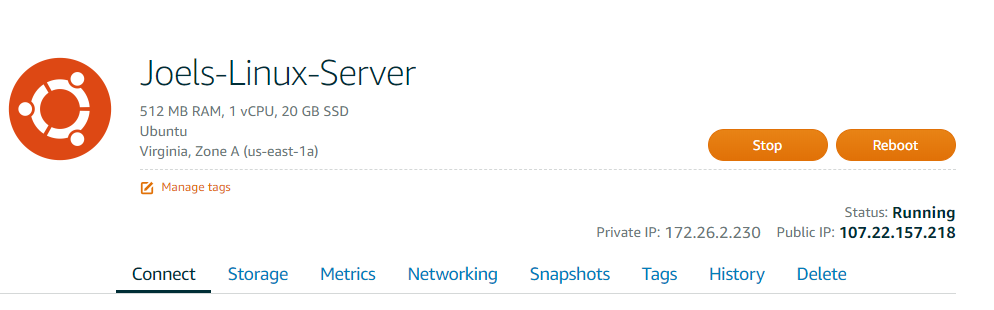
Amazon Saillight log in: [joelmagee@sbcglobal.net](mailto:joelmagee@sbcglobal.net)

Password: Disney9648-

AWS Name = JoelsUdacity

1. Sign into or create your Amazon Web Services Account (lightsail)
2. Use Udacity Linux Server Project steps to create an instance in lightsail of an Ubuntu ran virtual machine.

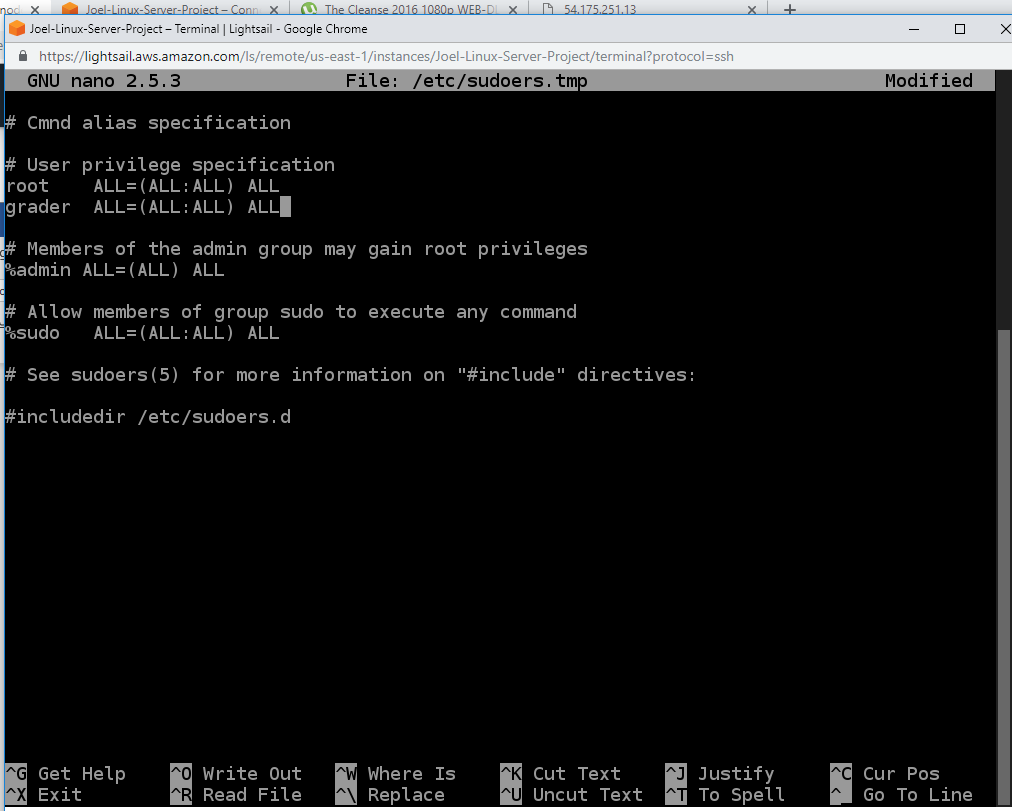


1. Connect to instance using ssh to begin project.
2. Update all current packages. ( **sudo apt-get update** THEN **sudo apt-get upgrade**)sudo
3. Install FINGER ( Program to find information about users) (**sudo apt install finger**)
4. Create new user grader ( **sudo adduser grader)**

User: grader

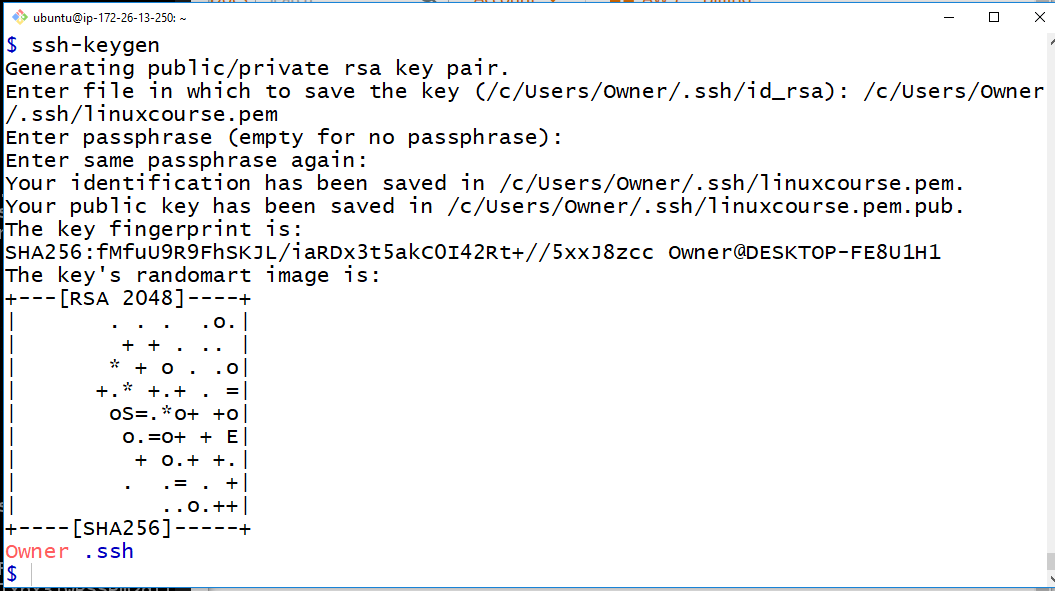
Password: grader

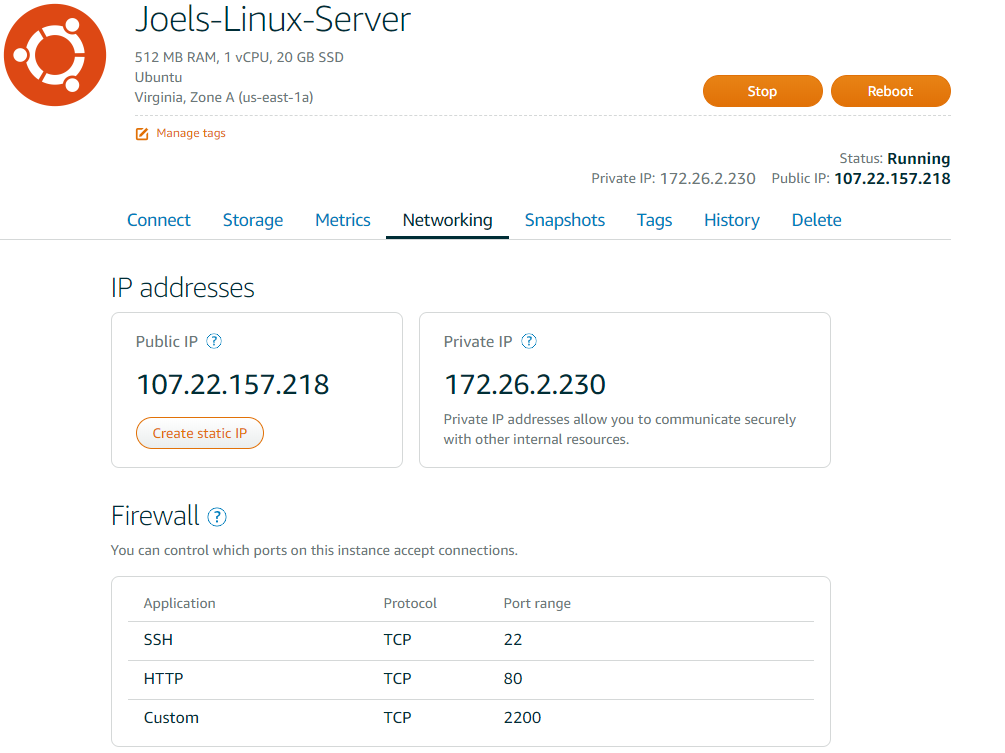
Fullname: Udacity Grader

1. Giver grade sudo access. (**sudo visudo**) add grader to USER PRIVILEDGE SPECIFICATION
2. In lightsail Networking – Firewall settings, add a custom listening port 2200. (Allows me to ssh into this sever once I change the port settings later. **REBOOT** Instance in lightsail.

Set SSH Login Using Key login in

1. On LOCAL computer generate key pair using **ssh-keygen** (my passphrase: joelsserver):

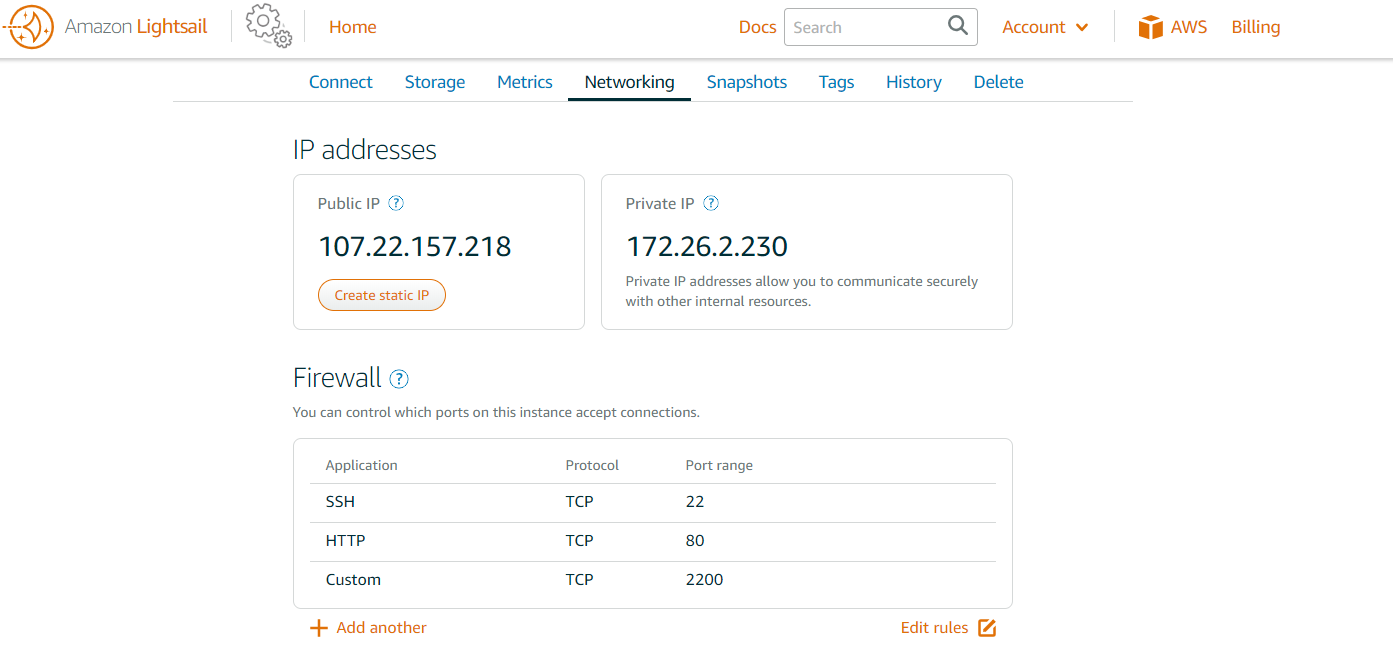




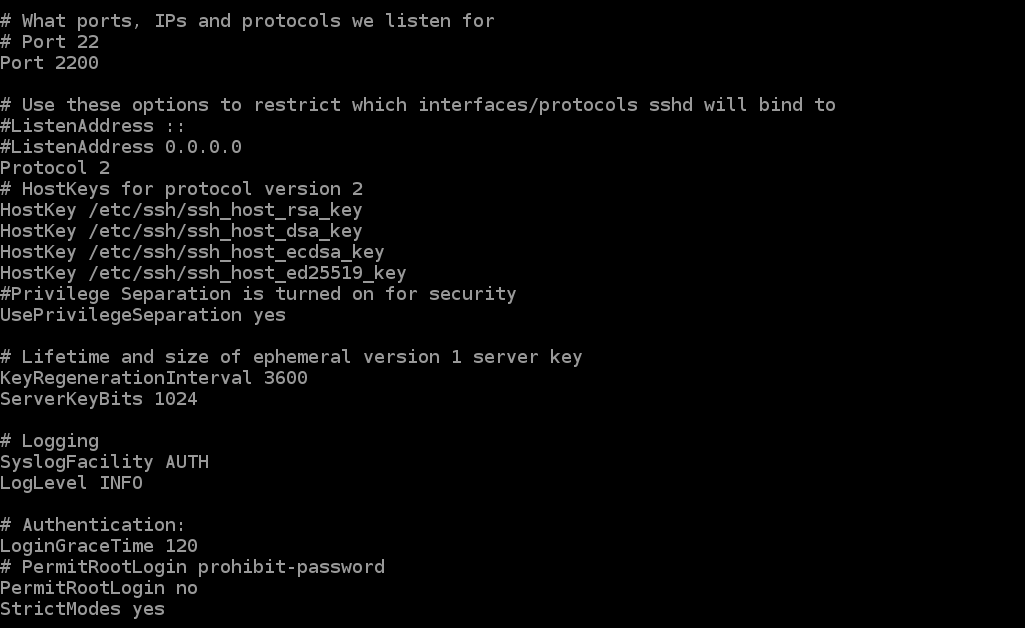
1. On server, use (**su grader** followed by password: grader). Switch to grader user to load in private key file.
2. Create new file to load in public key (**sudo** **touch .ssh/authorized\_keys**)
3. Goto local machine copy public key (**cat .ssh/linuxcouse.pub**) then paste into Servers authorized\_key file (**nano .ssh/authorized\_keys)**
4. When you log into the lightsail server, verify that you are in the /home/grader directory. (**pwd**)
5. Logged in as grader, use (**sudo chmod 700 .ssh**) and (**chmod 644 .ssh/authorized\_keys**)
6. Restart ssh (**sudo service ssh restart**)

Change SSH Port to 2200

1. In lightsail add a new listening Port 2200

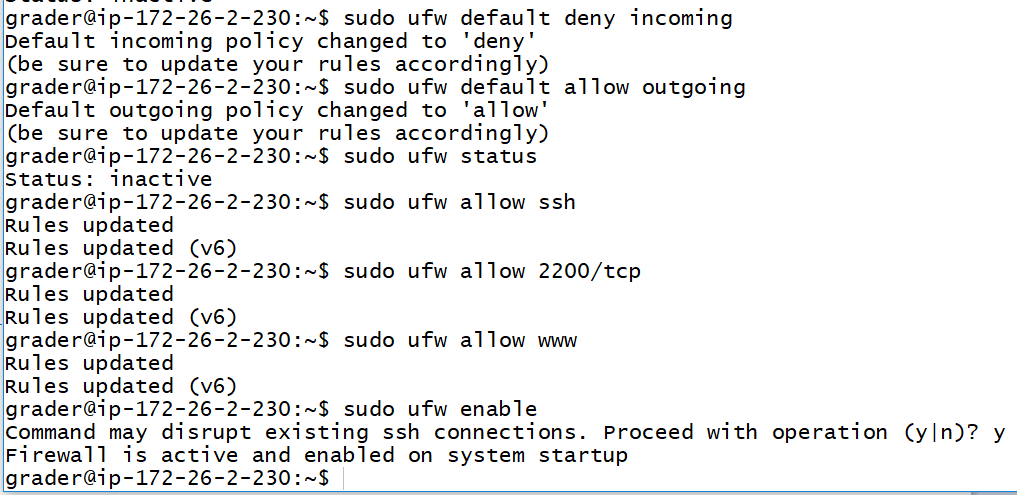


1. Log into server and change sshd\_config. (**nano /etc/ssh/sshd\_config**) Edit **the # What ports, IPs and protocols we listen for** and **#Authentication** to what you see below:



Configure Firewall

1. Use the follow commands that will configure your firewall ports then enable the firewall:



Change Time Zone to UTC

1. Type (**sudo dpkg-reconfigure tzdata**). Scroll down too **not seen**, then select **UTC.**

Install Apache2, mod\_wsgi, PostgreSQL and Git

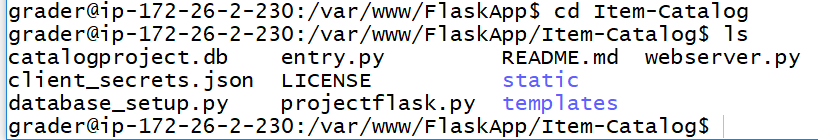
1. Install Apache2 (**sudo apt-get install apache2**)
2. Install mod\_wsgi (**sudo apt-get install libapache2-mod-wsgi**)
3. Restart apache2 (**sudo service apache2 restart**)
4. Install postgresql (**sudo apt-get install postgresql**)
5. Install Git (**sudo apt-get install git**)

Configure Postgersql (Reference: [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)%20) Database= catalog, User = catalog, Password = password

1. (**sudo vim /etc/postgresql/9.5/main/pg\_hba.conf**) Use this command and verify that all connections except local connections are commented out **(#).**
2. Log into Postgresql (**sudo su – postgres**)
3. Connect to the system (**psql**)
4. Create a new user and database (Used Udacity Fullstack Nano Degree Lesson 3. The Backend: Databases & Applications to help)
5. (**CREATE DATABASE catalog**;)
6. (**CREATE USER catalog;**)
7. Set password (**ALTER ROLE catalog WITH PASSWORD ‘password’;)**
8. (**\du**) Lists current roles and attributes, then (**\q**) to exit
9. Connect to database and limit access a. (**\c catalog**) b. (**REVOKE ALL ON SCHEMA public FROM public** ) c.(**GRANT ALL ON SCHEMA public TO catalog**)
10. Exit (**\q**), (**exit**)

Download Item catalog from github and set it up for virtual host (Reference: <https://www.digitalocean.com/community/tutorials/how-to-deploy-a-flask-application-on-an-ubuntu-vps>)

1. Move to directory www (**cd /var/www**) and make new directory FlaskApp (**sudo mkdir FlaskApp**)
2. Move to FlaskApp and clone in my github repository (**sudo** **git clone** [**https://github.com/jm2826/Item-Catalog.git**](https://github.com/jm2826/Item-Catalog.git)



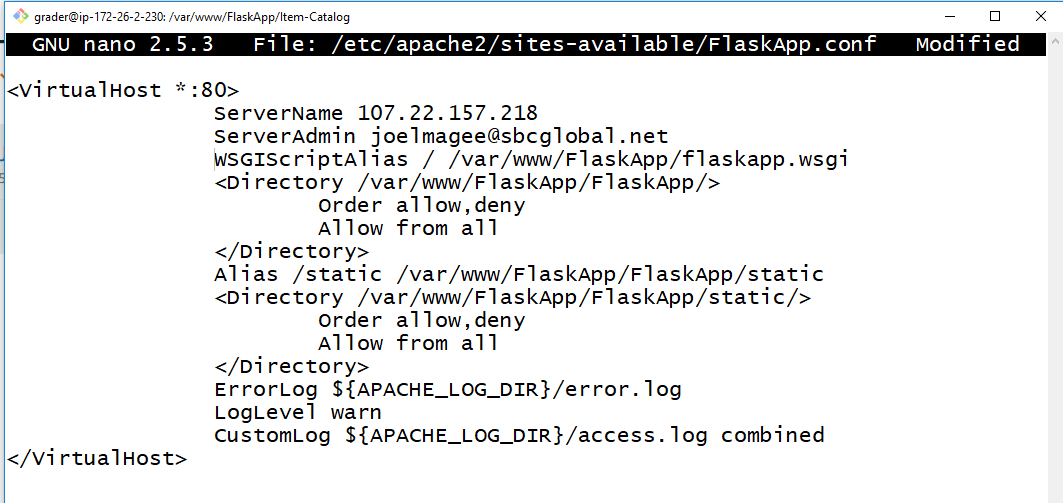
1. Renamed Item-Catalog to Flaskapp (**sudo mv ./Item-catalog ./FlaskApp**)
2. Move to Item-Catalog and rename projectflask.py to \_\_init\_\_.py (**sudo mv projectflask.py \_\_init\_\_.py**)
3. Edit database\_setup.py and webserver.py files from sqlite to postgresql with the catalog login information. (**sudo nano ‘filename’**)



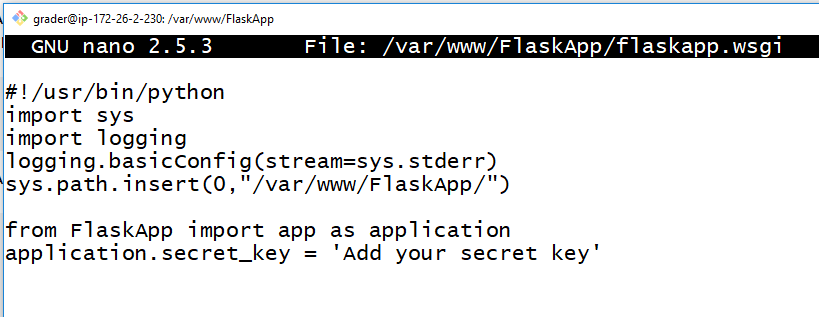
1. Install pip then SQLAlchemy (**sudo apt-get install python-pip**) (**sudo pip install SQLAlchemy**)
2. Install python passlib (**sudo apt-get install -y python-passlib**)
3. Create database by running python file. (**sudo python database\_setup.py**)

Create and configure Virtual Host

1. Create configuration file and then edit it. (**sudo nano /etc/apache2/sites-available/FlaskApp.conf**)
2. Type in this:



1. Enable virtual host (**sudo a2ensite FlaskApp**)
2. Restart apache server (**sudo service apache reload** )
3. Create the flaskapp.wsgi file (**sudo nano /var/www/FlaskApp/flaskapp.wsgi)** and type this in:



1. Restart apache again (**sudo service apache2 restart**)

Google OAUTH2CLIENT Setup

<https://developers.google.com/api-client-library/python/auth/web-app>

Due to Errors Installed more Apps and adjusted server settings

1. oauth2client (**sudo pip install –upgrade oauth2client**)
2. Passlib (**sudo apt-get install -y python-passlib**)
3. Virtualenv (**sudo pip install virtualenv**)
4. Flask (**sudo pip install Flask**)
5. Used (**sudo cat /var/log/apache2/error.log**) to check error log in apache2.
6. **Sudo ufw allow 123/upd** then **sudo ufw enable**

This page can be used to find the IP of a host machine (convert host to IP) or domain name (convert domain name to ip address) or find the name of one of the hosts at an IP address (convert ip address).

It will also show the location of IP address. The [country data](http://www.hcidata.info/host2ip.htm#country) is about 94% accurate.

## Details of 107.22.157.218

IP Address : 107.22.157.218

Location : United States (95% accuracy)

Host Name : ec2-107-22-157-218.compute-1.amazonaws.com

## Details of Computer Using This Web Service

Below is shown the name of the computer that is using this web service. If this computer is behind a firewall or uses a proxy server, the name shown will be that of the firewall computer or proxy server. Many ISPs route internet traffic via a proxy server to reduce network traffic. Most commercial organisations protect their internal network (Intranet) by routing all traffic to and from the Internet via a firewall. A firewall computer of proxy server is sometimes called a 'gateway'.

IP Address : 99.127.92.113

Host Name : 99-127-92-113.lightspeed.cicril.sbcglobal.net

### GitHub OAuth App

For 3rd party OAuth provision to function properly, you must create an OAuth App in GitHub. You may create an app [here](https://github.com/settings/applications/new). Set "Homepage URL" to "[http://localhost:5000](http://localhost:5000/)", and "Authorization callback URL" to "<http://localhost:5000/login/github/authorized>". The other fields maybe filled in however you want. Register your app and find values for "Client ID" and "Client Secret". Create environment variables with these values, named "IC\_CLIENT\_ID" and "IC\_CLIENT\_SECRET" respectively.

* If setting the environment variables is unsuccessful, you may insert the values directly into your code for demonstration purposes only.

blueprint = make\_github\_blueprint(

client\_id="insert-your-clinet-id",

client\_secret="insert-your-client-secret"

)

Note that this is an insecure means of incorporating these values in your code.

### Other Requirements

This application uses Flask-Dance to implement 3rd party OAuth provision. By default, Flask-Dance requires HTTPS, or the application will not run. This requirement may be suppressed by running the following:

$ export OAUTHLIB\_INSECURE\_TRANSPORT=1

* Note that this is an insecure means of running the application and should only be used when testing the application on a local device.

# Authenticating login through google

From Google Cloud Plateform.

Click APIs & services on left menu then Credentials.

Create an OAuth Client ID, add and add [http://52.29.32.59.xip.io](http://52.29.32.59.xip.io/) as authorized JavaScript origins.

in authorized redirect URI add

<http://52.29.32.59.xip.io/oauth2callback>

<http://52.29.32.59.xip.io/login>

<http://52.29.32.59.xip.io/gconnect>

Download the corresponding JSON file, open it then copy the contents and paste them into /var/www/catalog/Catalog/client\_secrets.json.

Replace the client ID of templates/login.html file in the project directory.

Add the path of client\_secrets.json to **init**.py ""/var/www/catalog/Catalog/client\_secrets.json""

**Modifying the application (where needed)**

Created a new OAuth 2.0 client ID credentials for Google OUATH

* In Google Developer's consoloe
  + set the **javascript origin** of the project to the IPv4 & Default DNS
  + set the **redirect\_uris** to http://ec2-18-188-108-29.us-east-2.compute.amazonaws.com/gconnect
* downloaded the new credentials
* updated the credentials in the project in g\_client\_secrets.json & in /templates/login.html
* updated the g\_client\_secrets's path by appending /srv/path to it in the auth.py file (where needed)
* updated the URL for the database in app.py & db\_setup.py to DB\_URL = "postgresql://sportsbazar:password@localhost/sportsbazar"

1. Go to google console and add authorized domains xip.io
2. Add authorized Javascript origins: [http://35.180.75.141.xip.io](http://35.180.75.141.xip.io/)
3. Add authorised redirect URIs: <http://35.180.75.141.xip.io/gconnect> and <http://35.180.75.141.xip.io/login>
4. Downlaod the updated JSON file, copy its content and paste it in your server, run the following command :
   * sudo nano client\_secrets.json
5. restart ssh server, run the following commands :
   * sudo a2dissite 000-default.conf
   * sudo a2ensite catalog.conf
   * sudo service apache2 reload
   * sudo service apache2 restart