Protected: CSC-394 HW-6 Tooling Setup EC2 Setup

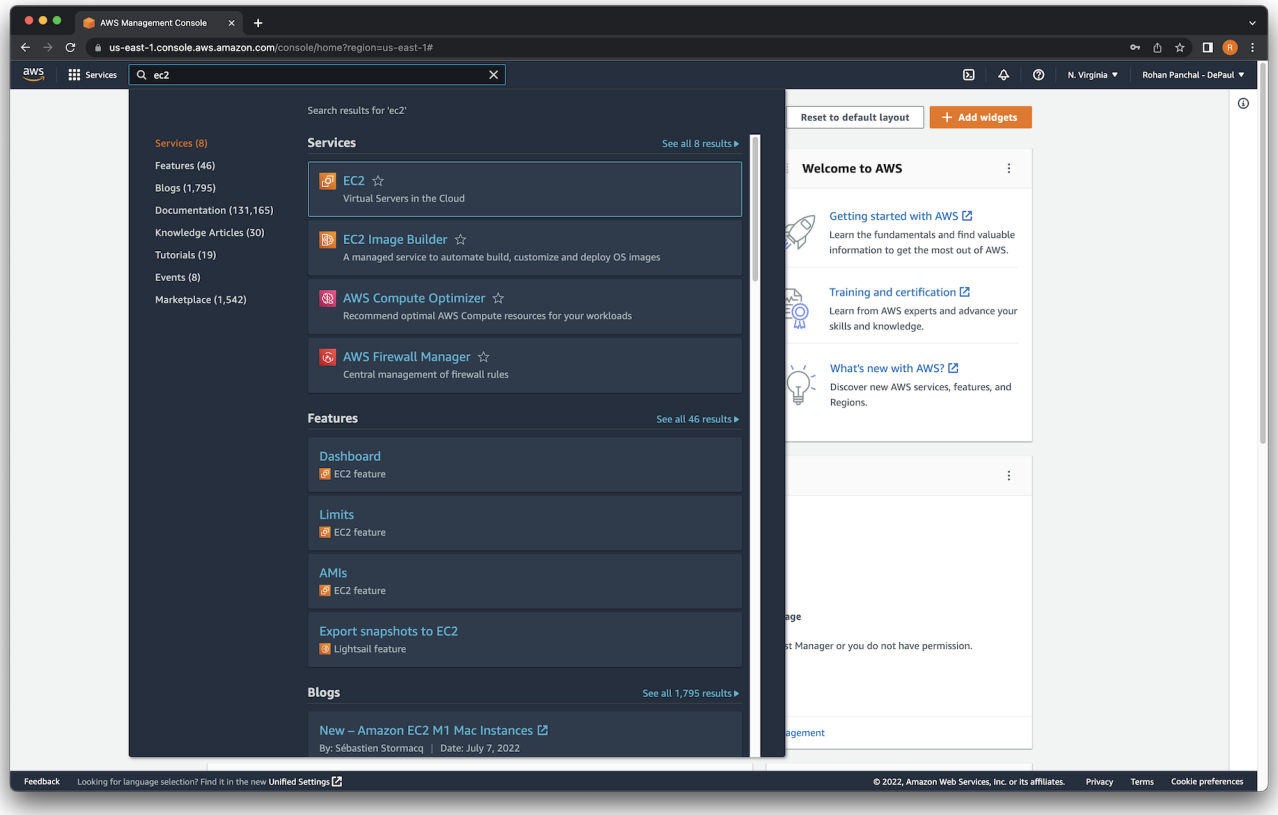
Step 1 – Create an EC2 instance

What is EC2?

EC2 stands for `Elastic Compute Cloud` . It is a service provider of Cloud Computing in AWS's data centers. It allows you to provision, and deploy a Linux, Windows, macOS server in the cloud! Accessible from anywhere you like!

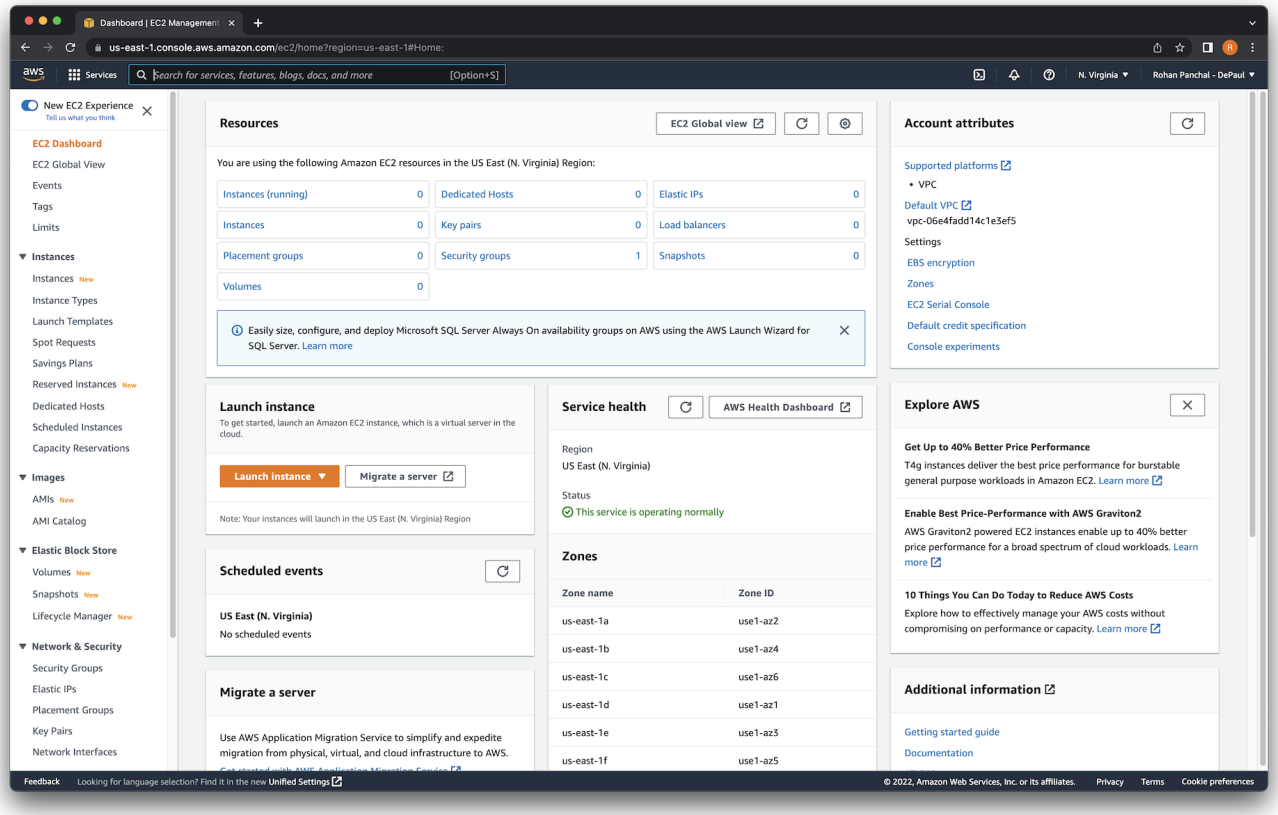
For this assignment we will use EC2 to provision a server to host the django application we created in the previous assignment.

In the search bar at the top of the `management console` type in EC2, and select the EC2 icon in the dropdown.



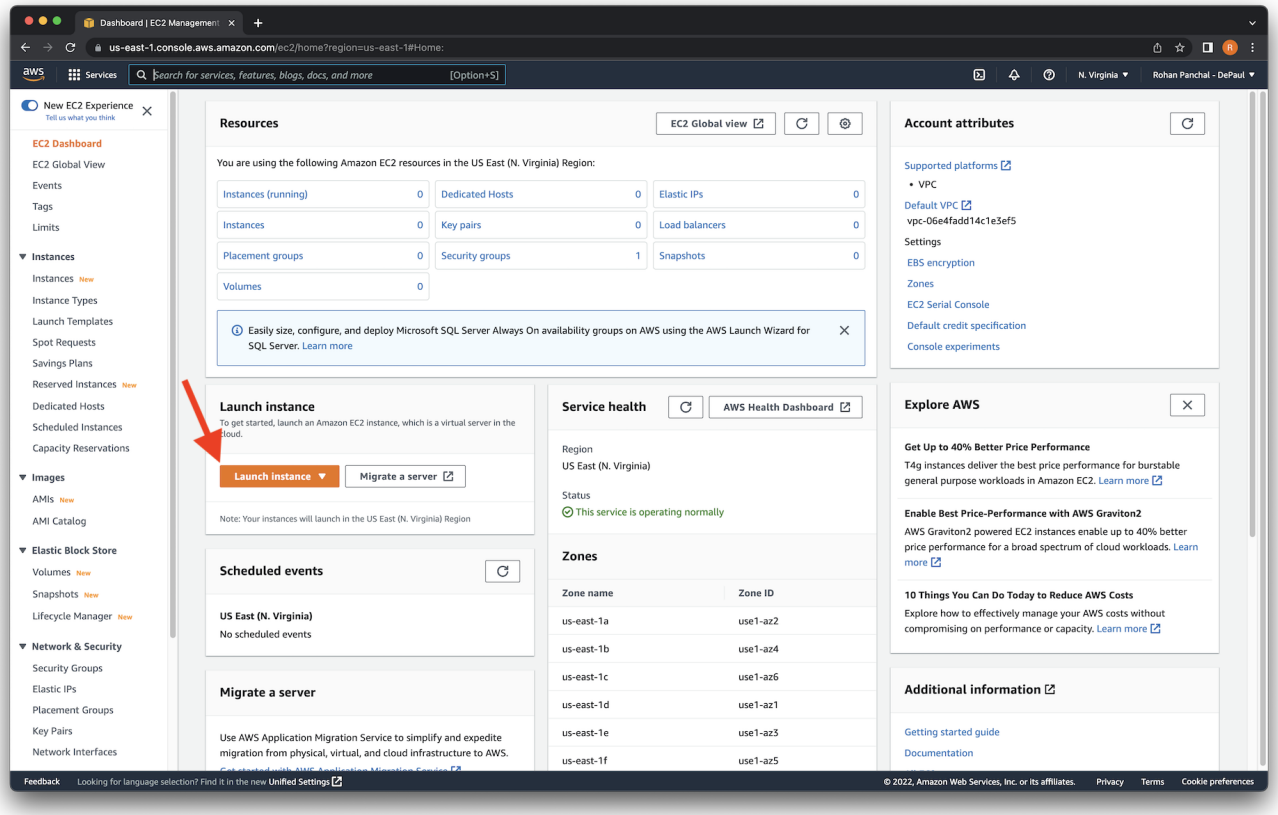
You should now see the EC2 Management Console.

It has A LOT of options and buttons. And can look quite intimidating. But for now we only need to focus on a few areas.

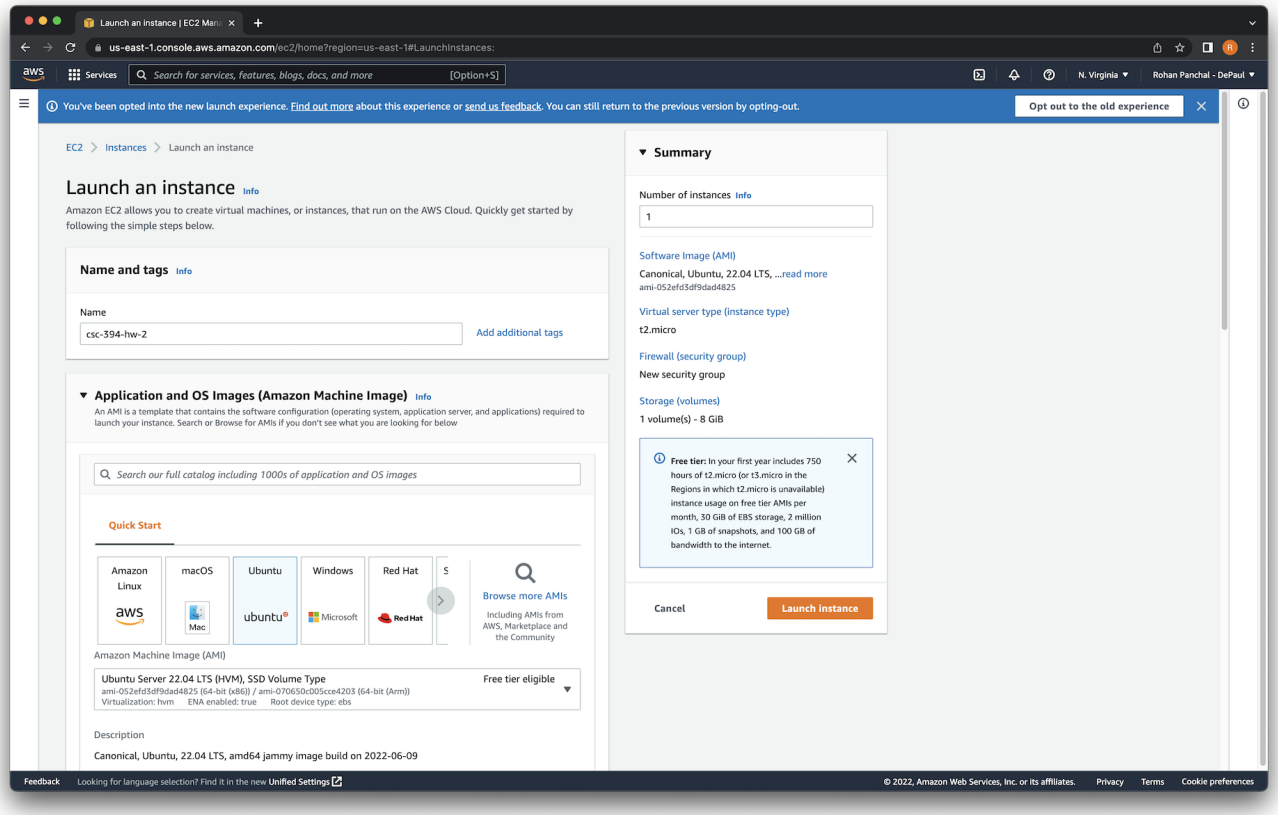


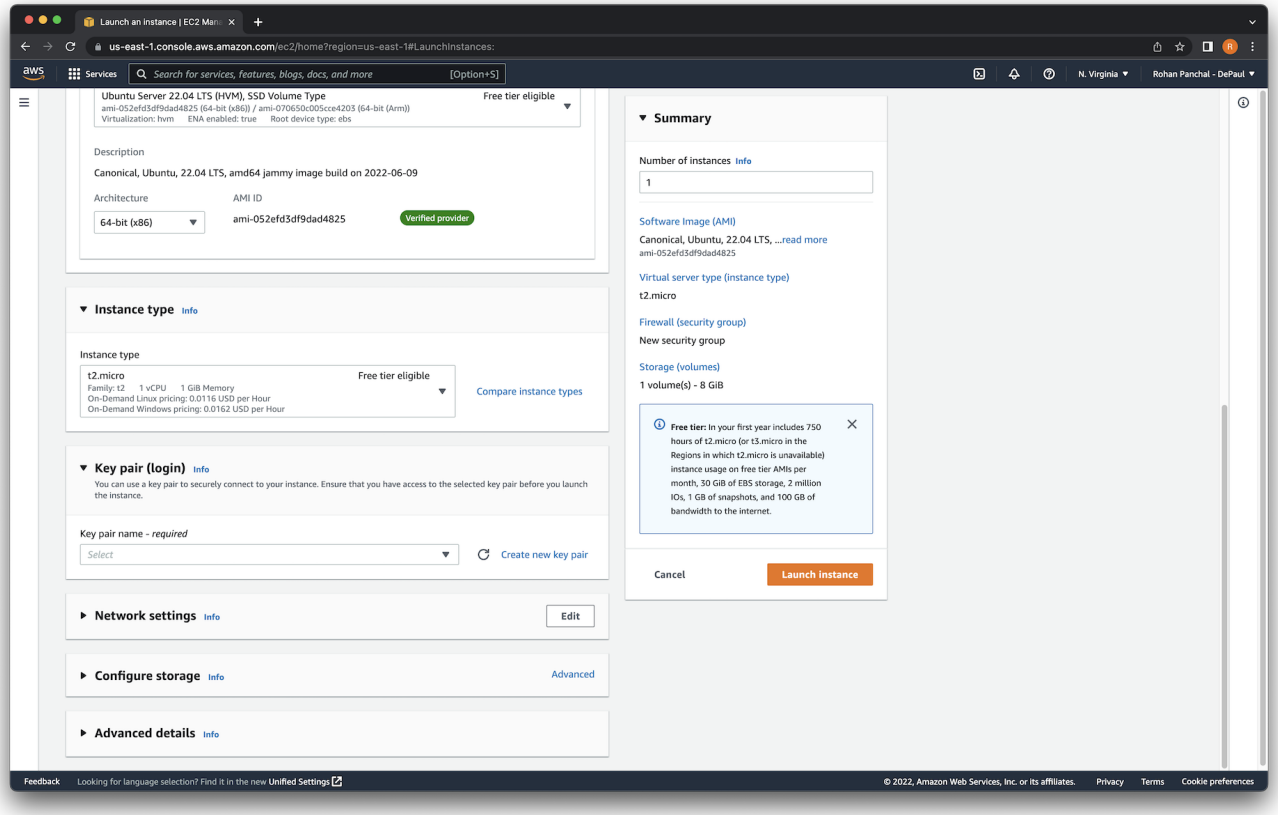
AWS EC2 Management Console

Click the `Launch instance` button



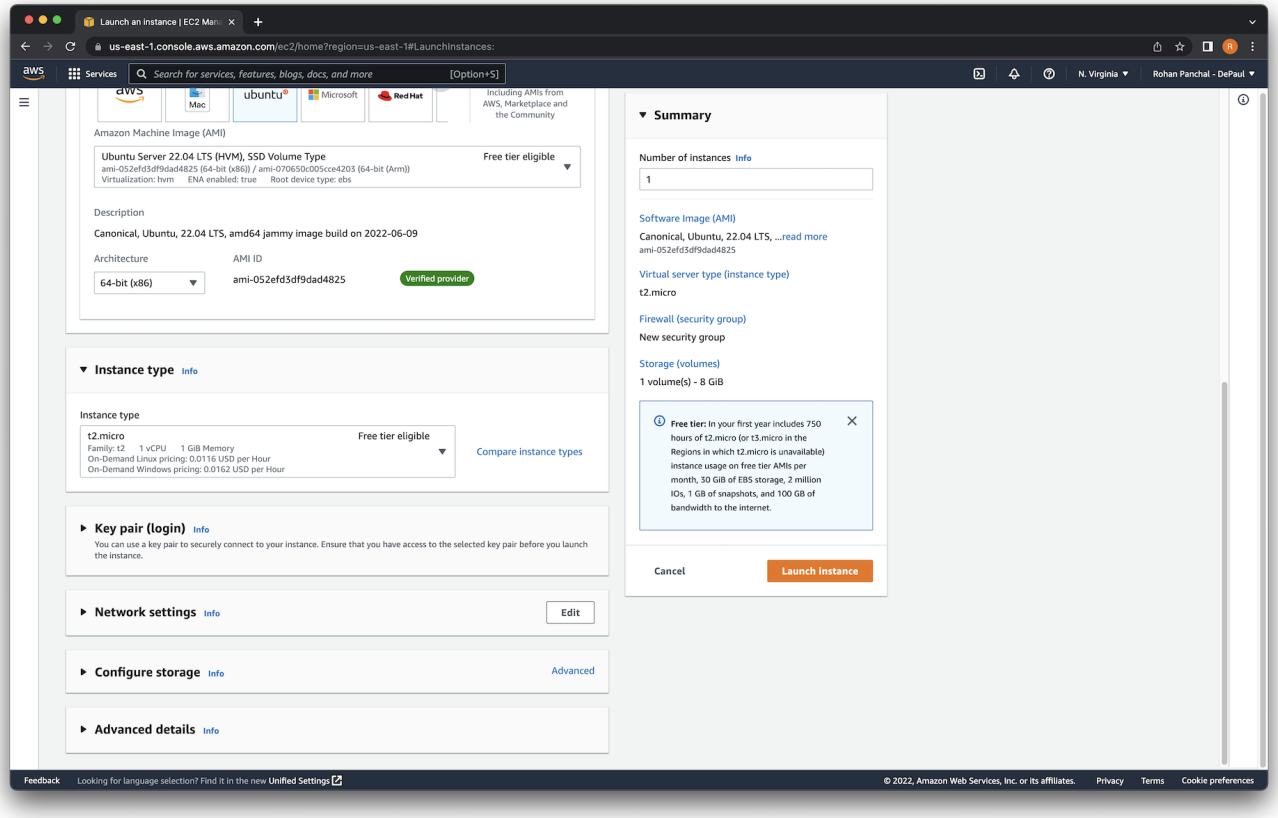
- Enter a name for your `EC2 instance`
- In the quick start section choose the `Ubuntu` option.



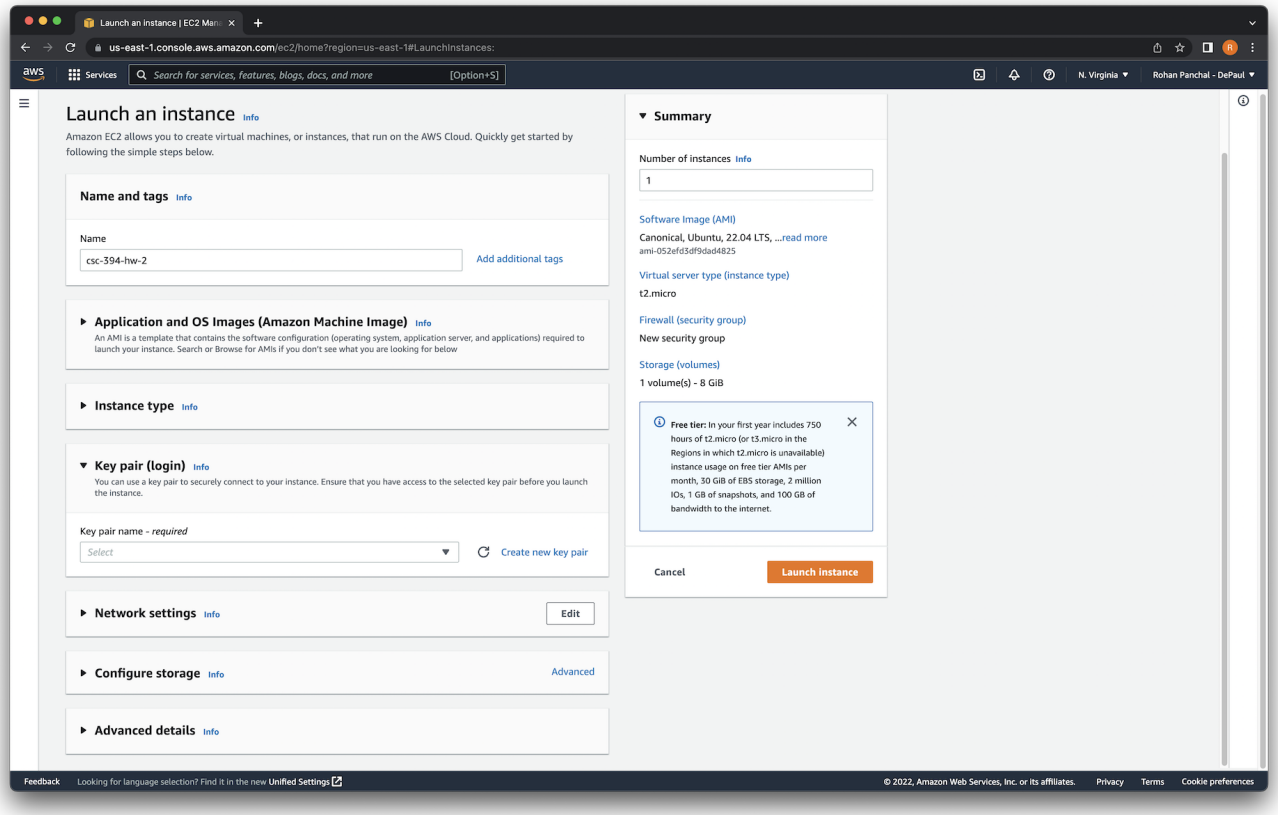


The next set of steps are a little more complex.

First ensure that the `Instance type` is set to `t2.micro`.



Then select the option to `Create a new key pair` in the `Key pair (login)` section.

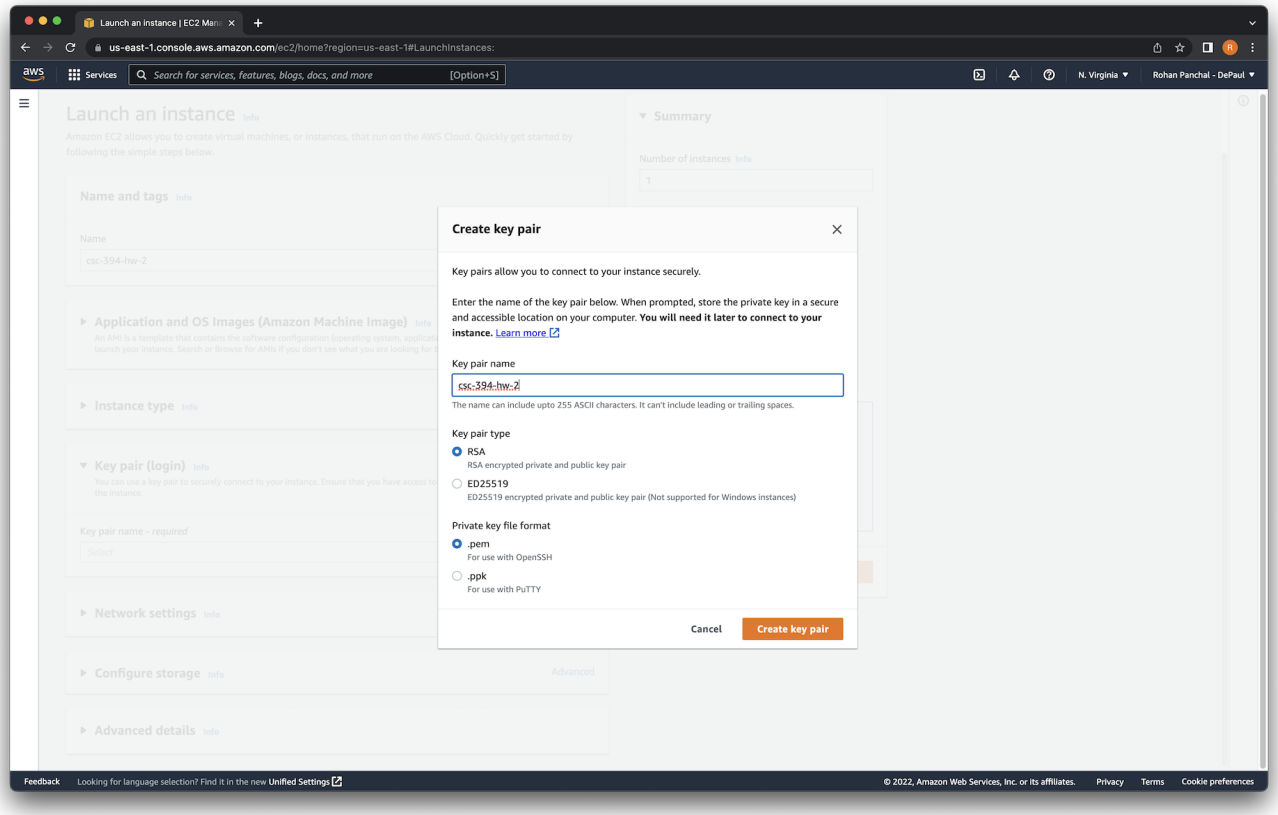


Enter a `key pair name`

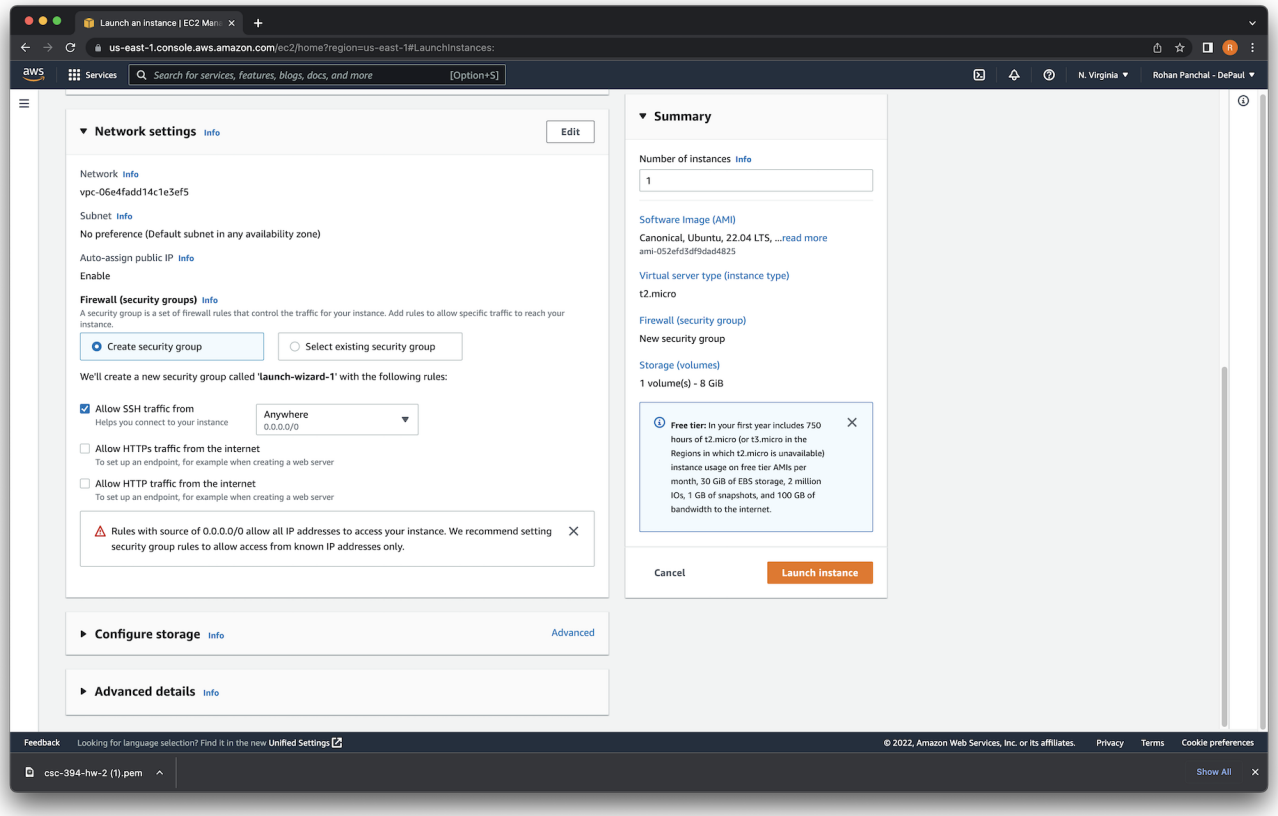
Make sure the key pair type is `RSA`

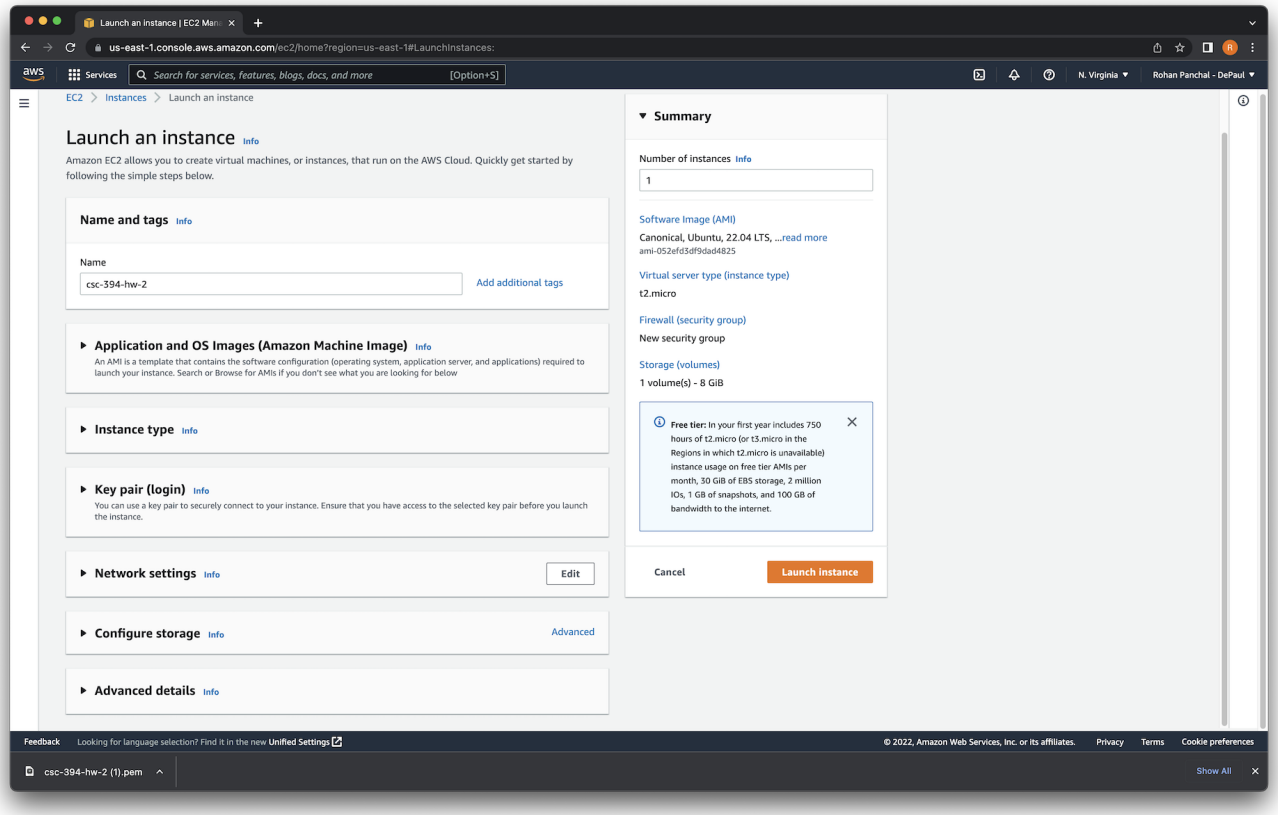
And the private key file format should either be .pem or .ppk if you are using PuTTY

Click `Create key pair` and it will download the `.pem` file for you.

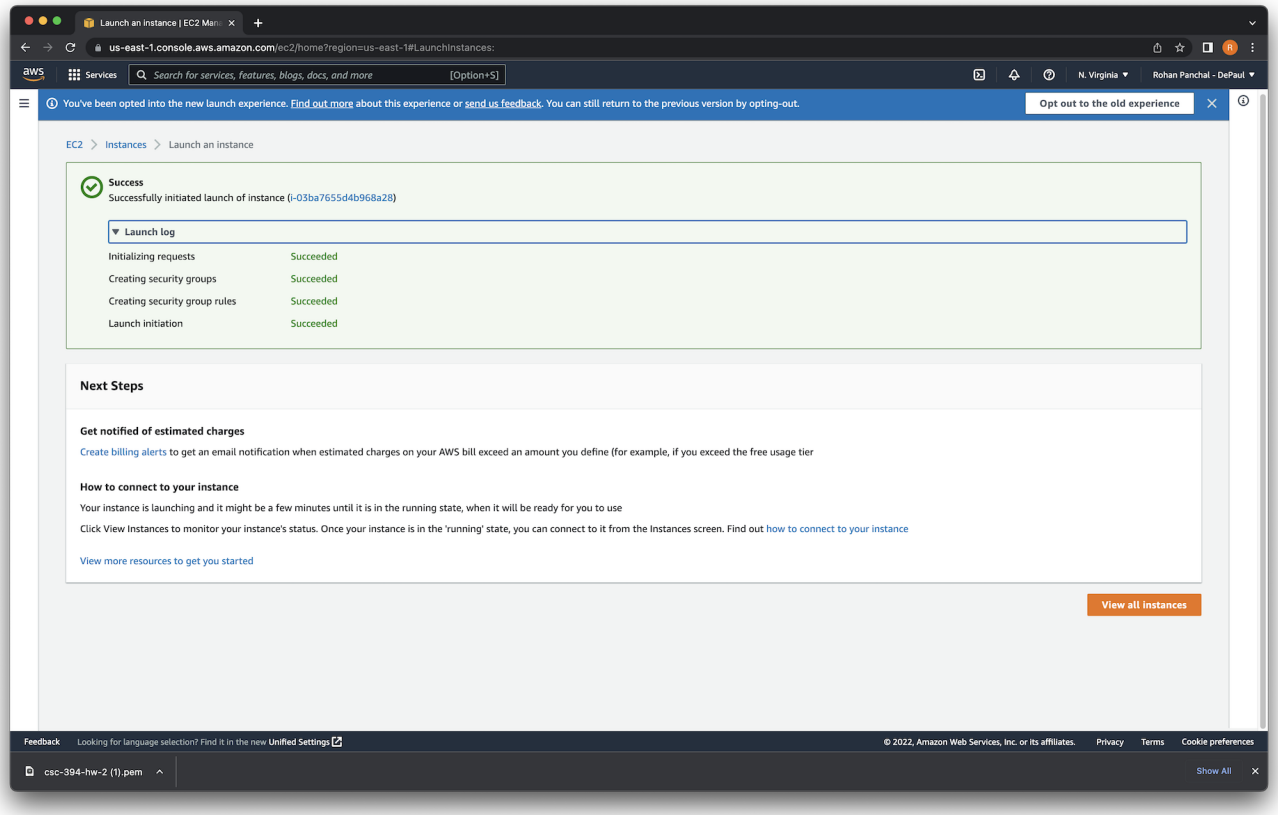


In the `Network settings` opt into the defaults as shown here:

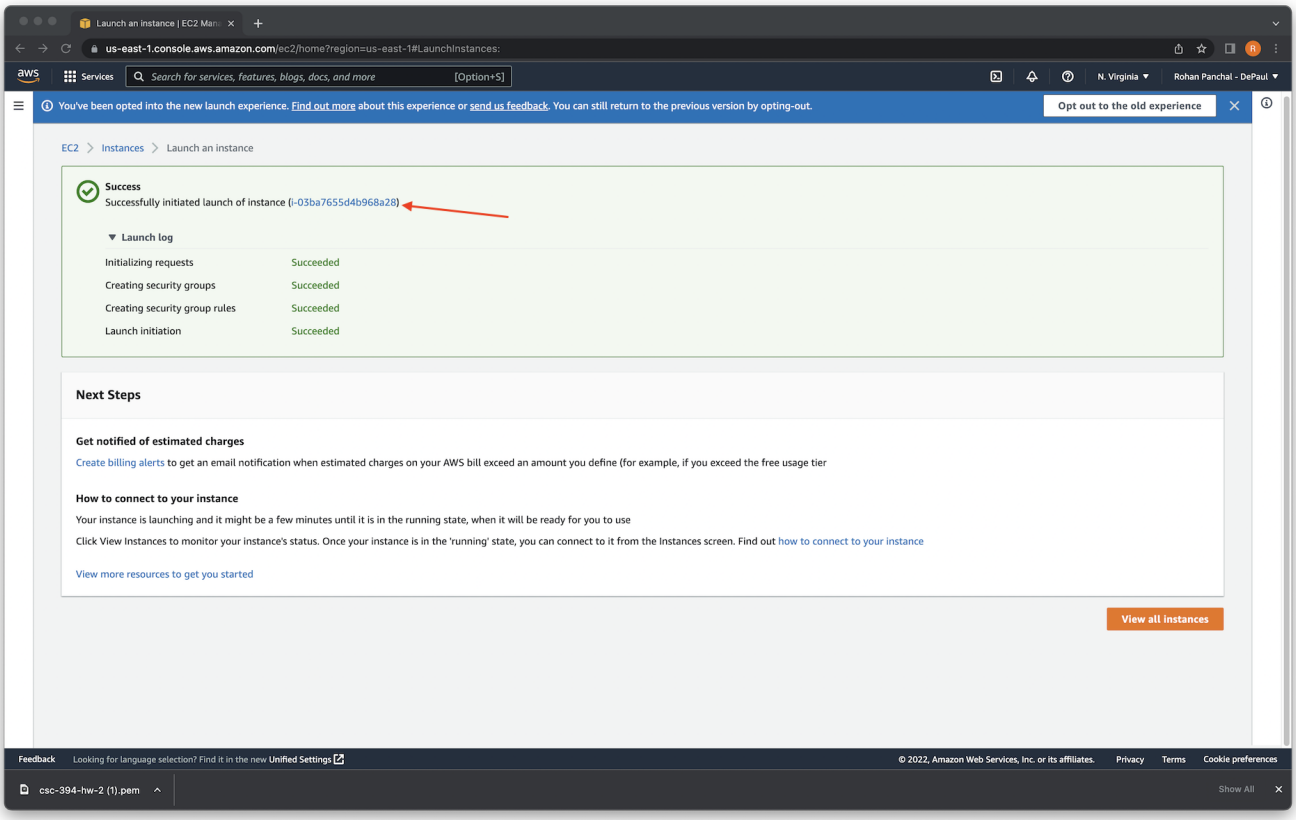




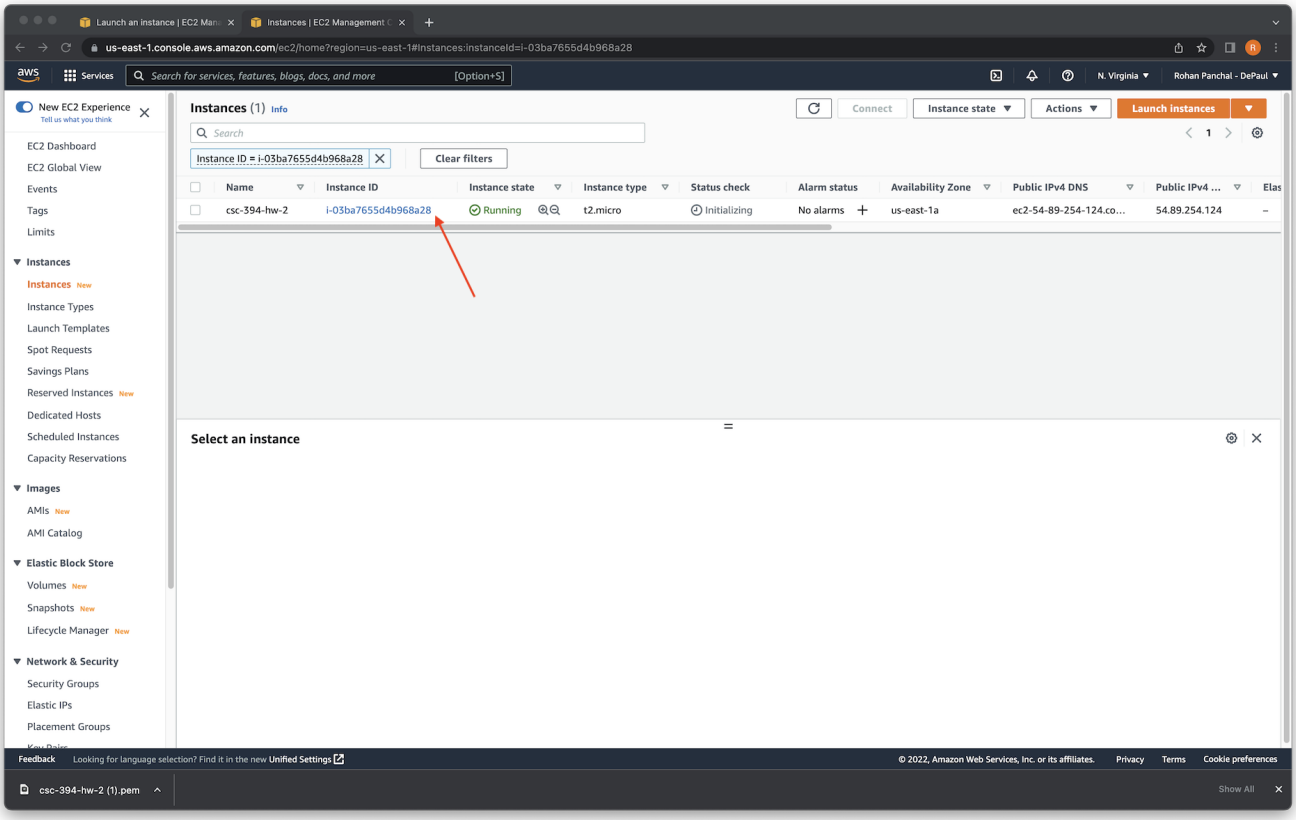
Leave the `Configure storage` and `Advanced details` sections to their default options. And then click the `Launch instance` button on the right sidebar.

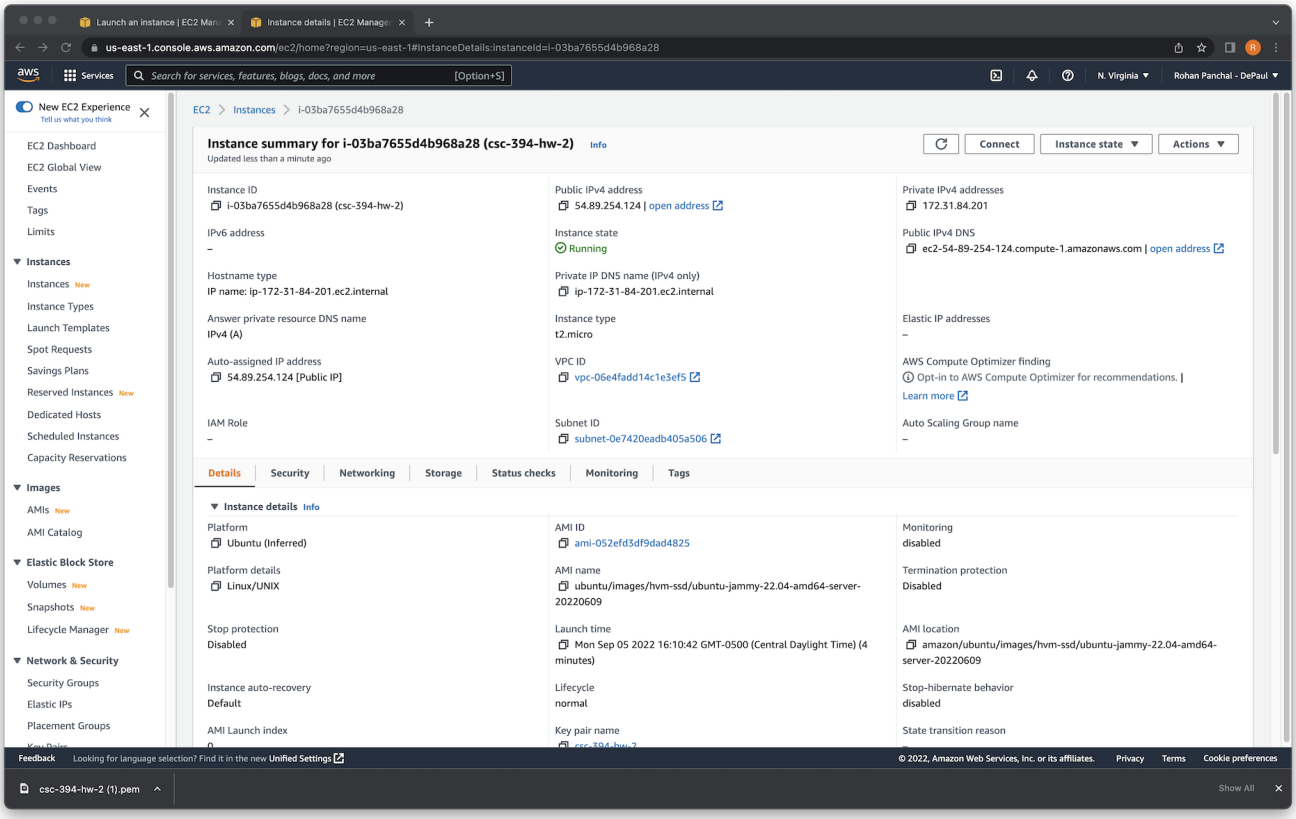


Click on the instance number to navigate to the `EC2 Instance List`



Click the `Instance ID` to navigate to the `EC2 Instance Dashboard`



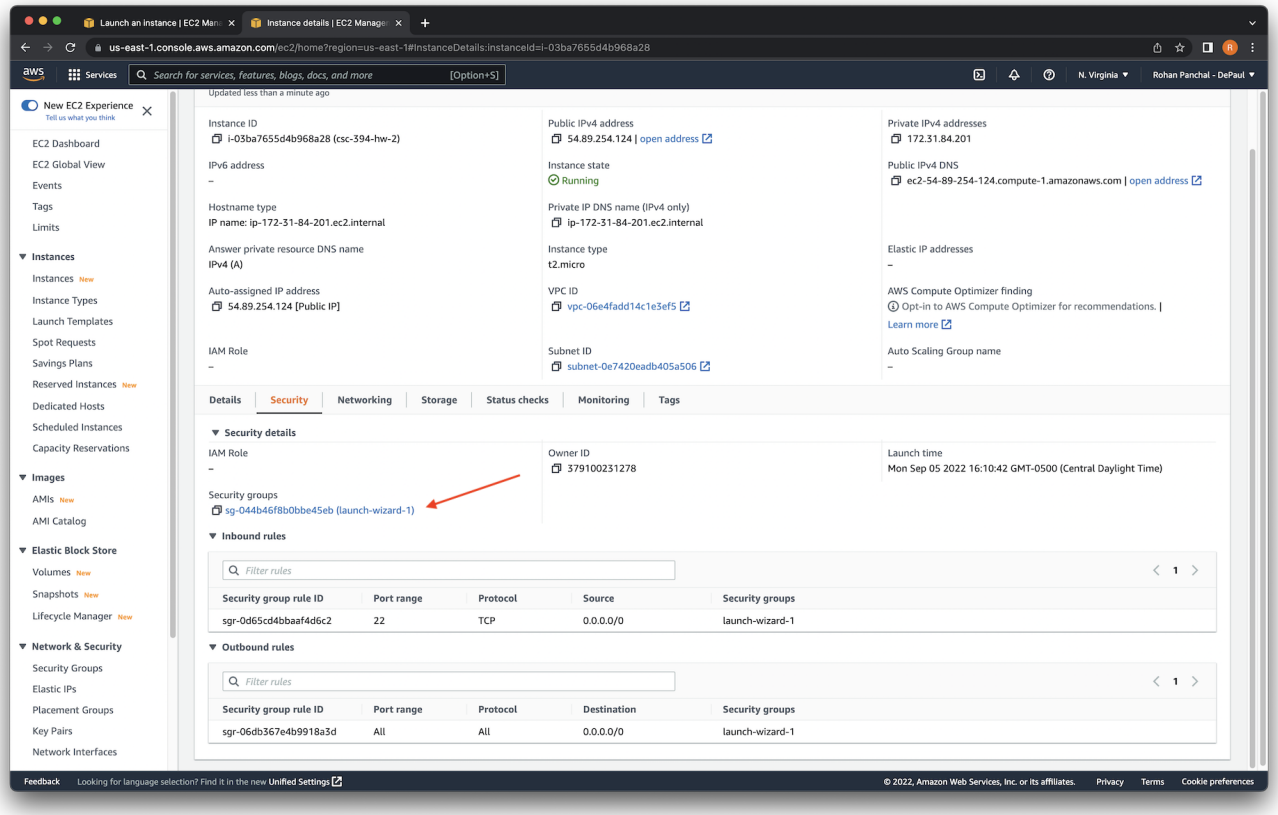


Success! You have an EC2 instance setup!

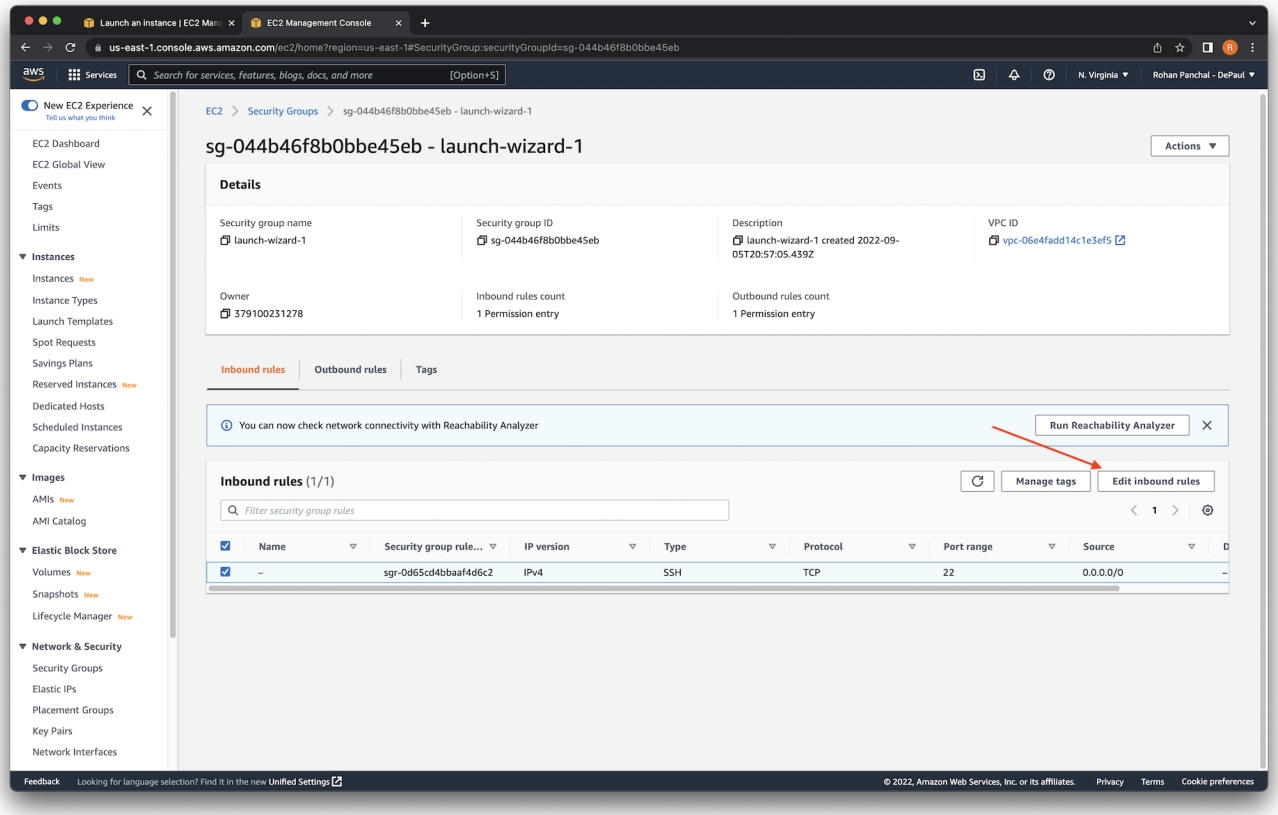
Updating EC2 security

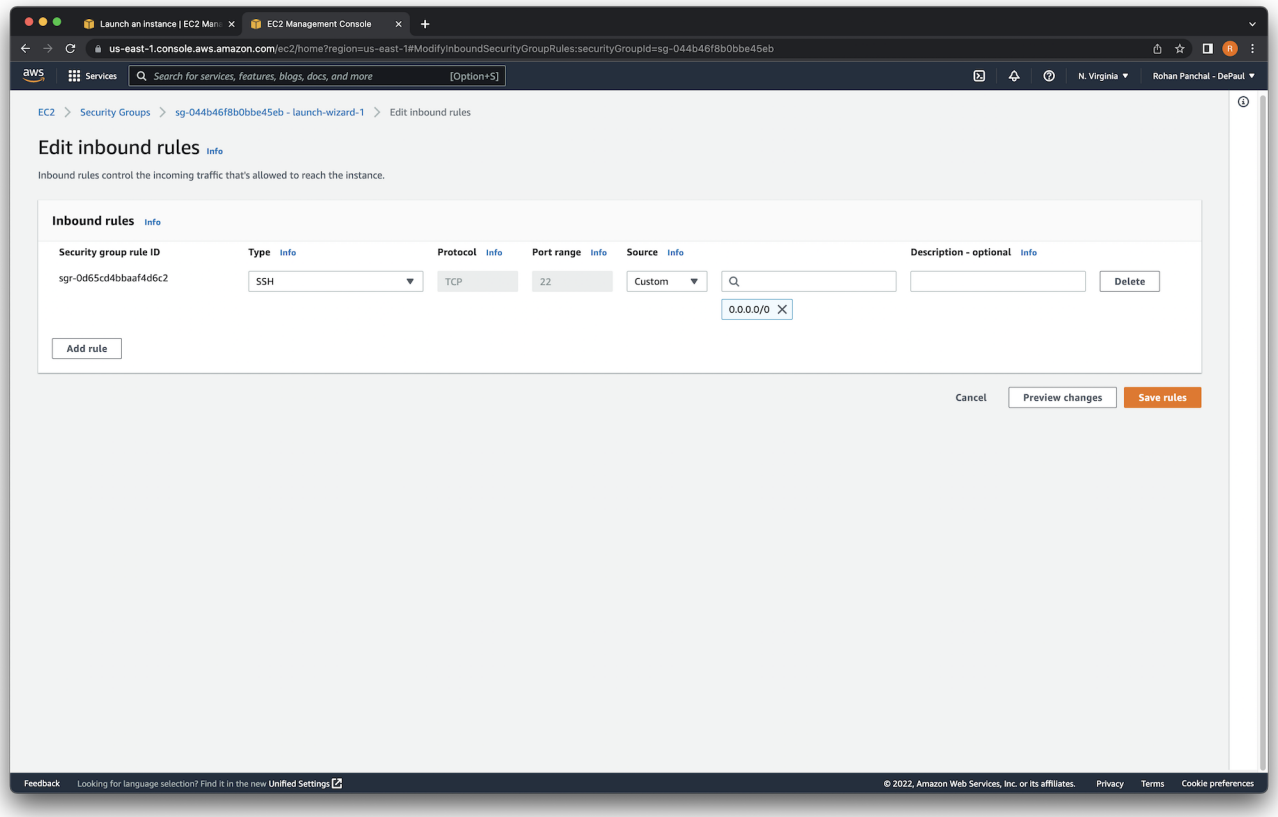
Back in your EC2 instance dashboard, click on the security tab, and click on the

`Security Group instance ID`



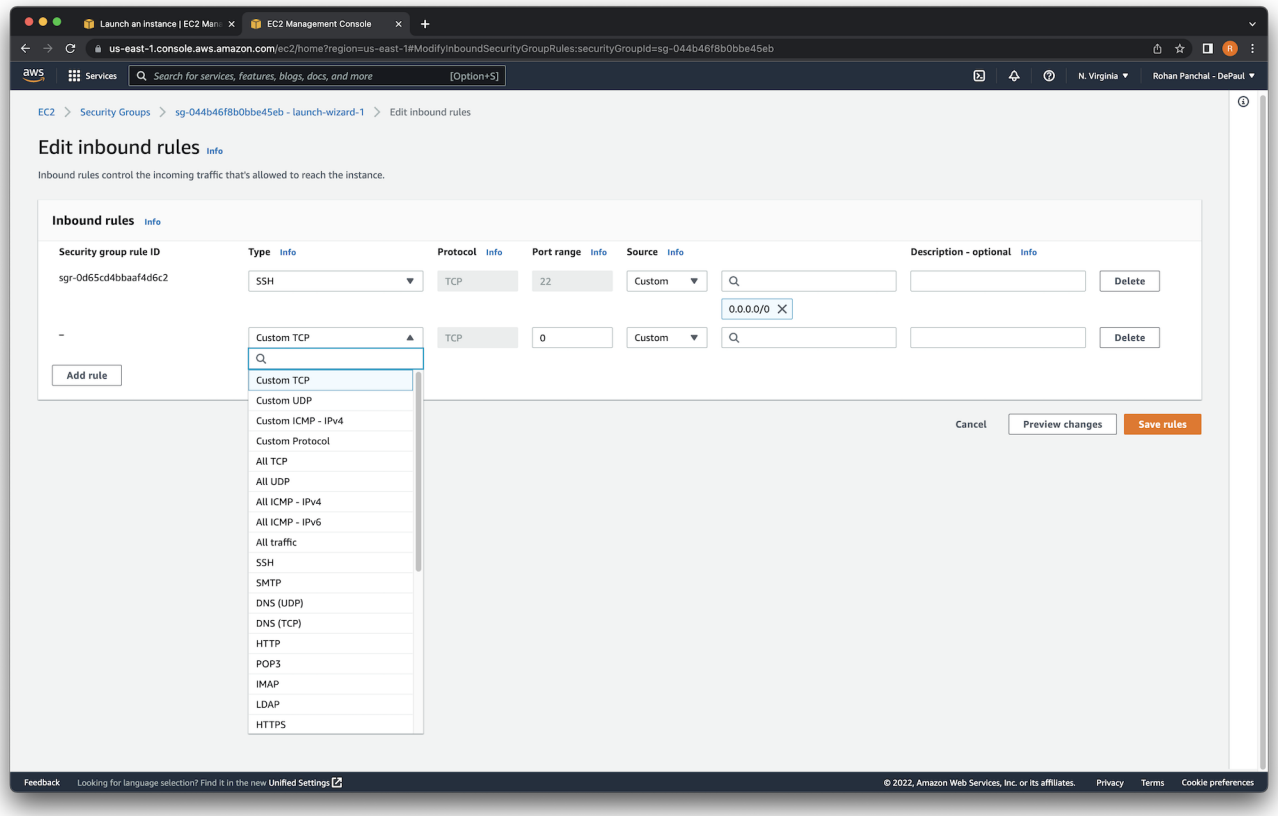
Click on the `Edit inbound rules` button





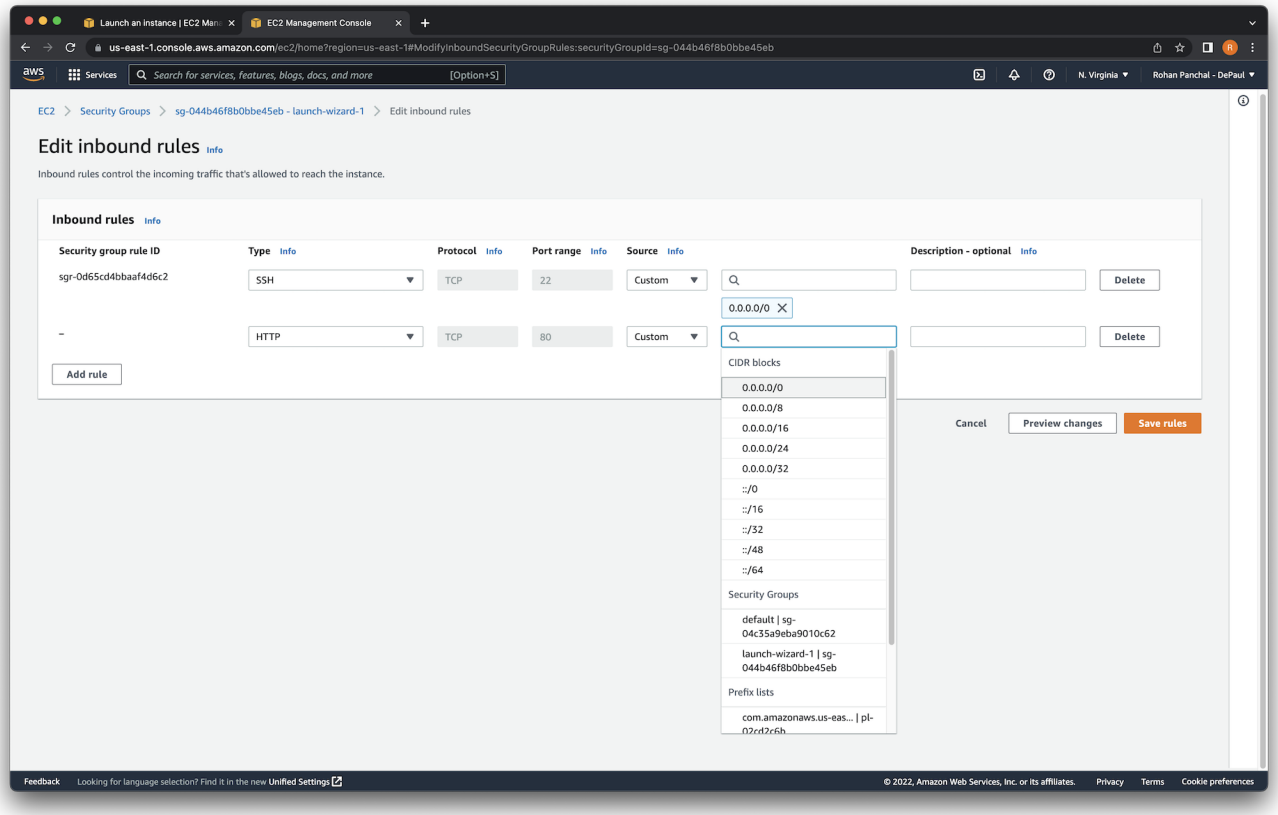
Click `Add Rule`

In the new line item, click the `Type` dropdown and select `HTTP`

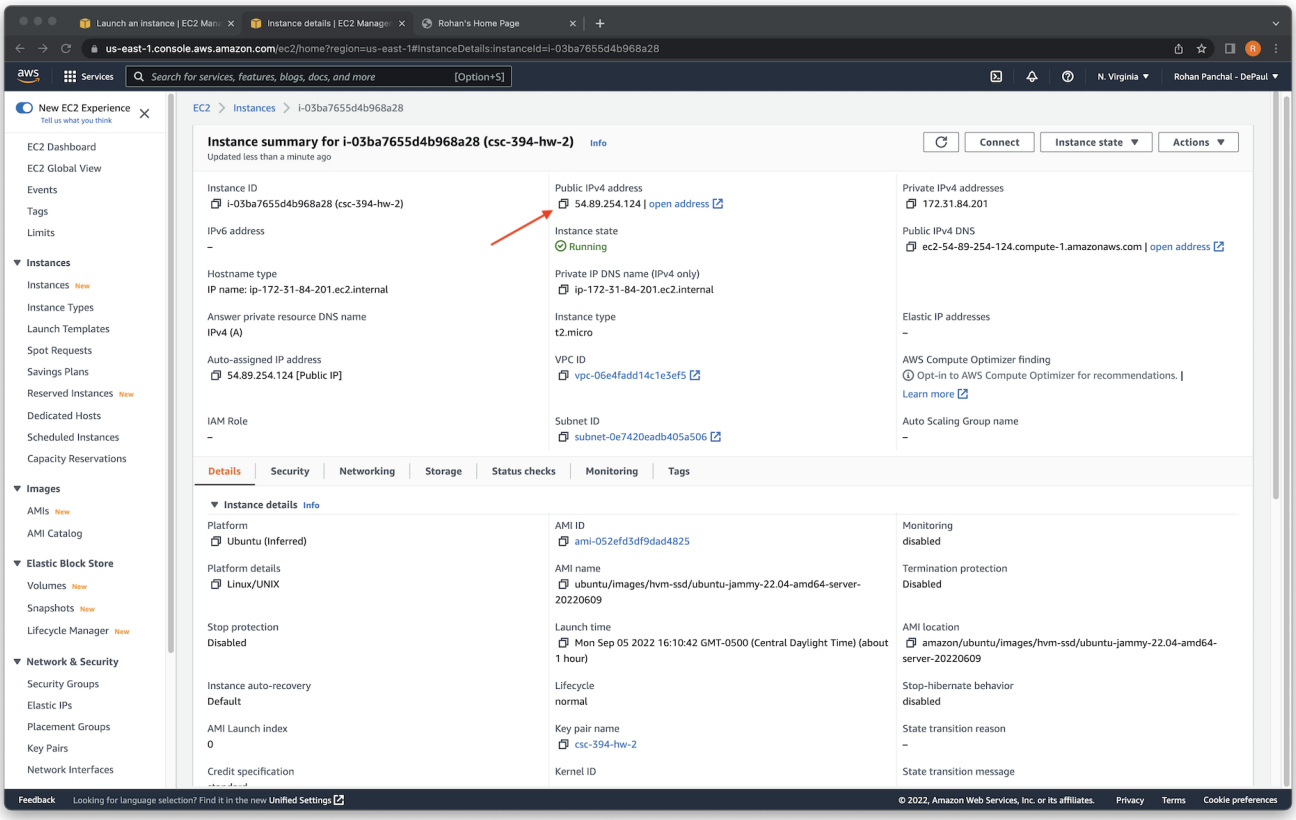


In the dropdown next to the `Source` column, select the `0.0.0.0/0` entry.

Click `Save Rules`



Navigate back to the `EC2 instance dashboard` and copy the `Public IPv4 address`



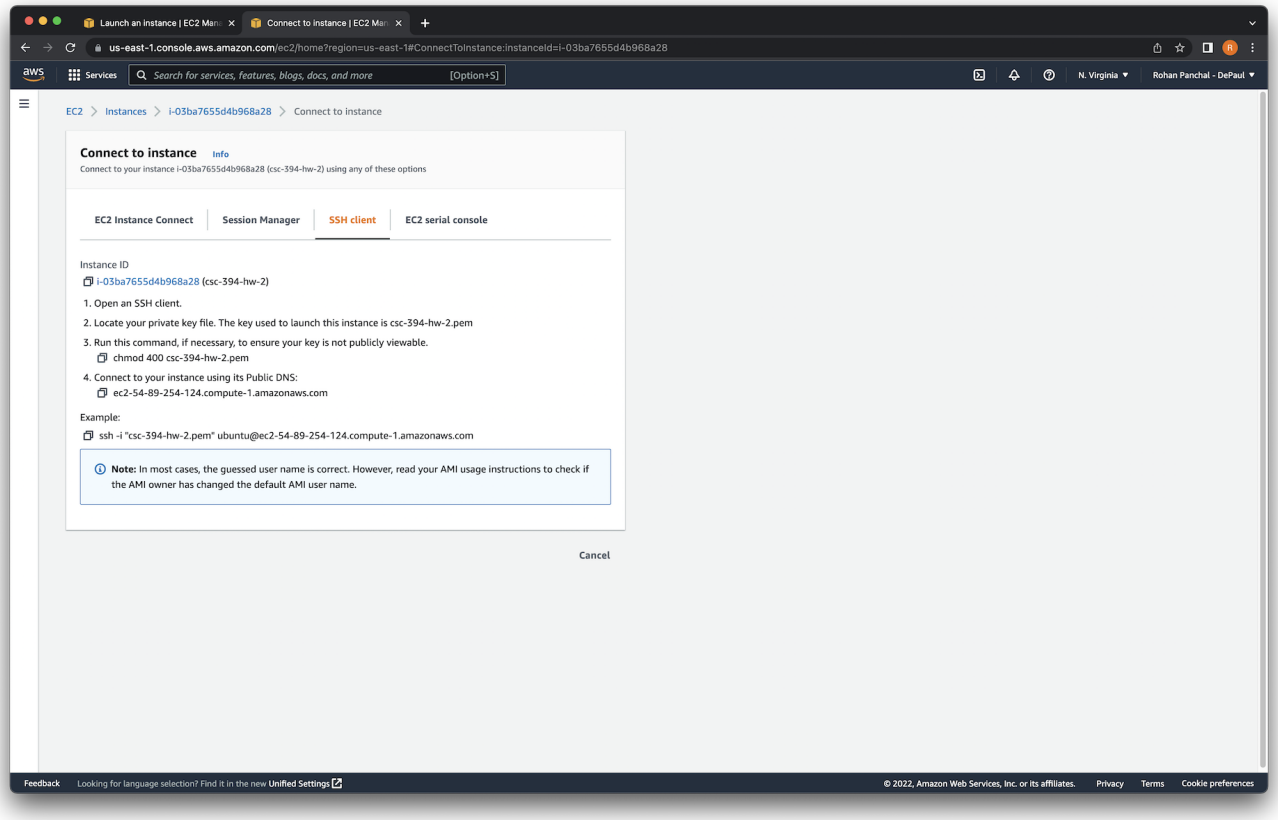
Step 2 – Deploy your site on an EC2 instance

Click the `Connect` button on the top right of the screen

You now see the `Connect to instance` selector

Click on the `SSH client` selector

And then copy the `example` at the bottom.



My ssh connection command was:

```
ssh -i "csc-394-hw-project.pem" ubuntu@ec2-54-89-254-124.compute-1.amazonaws.com
```

Yours will have a different `ip address` and the location of your `.pem` file may be at a different directory (mine was in my `Downloads` directory). But with this command you

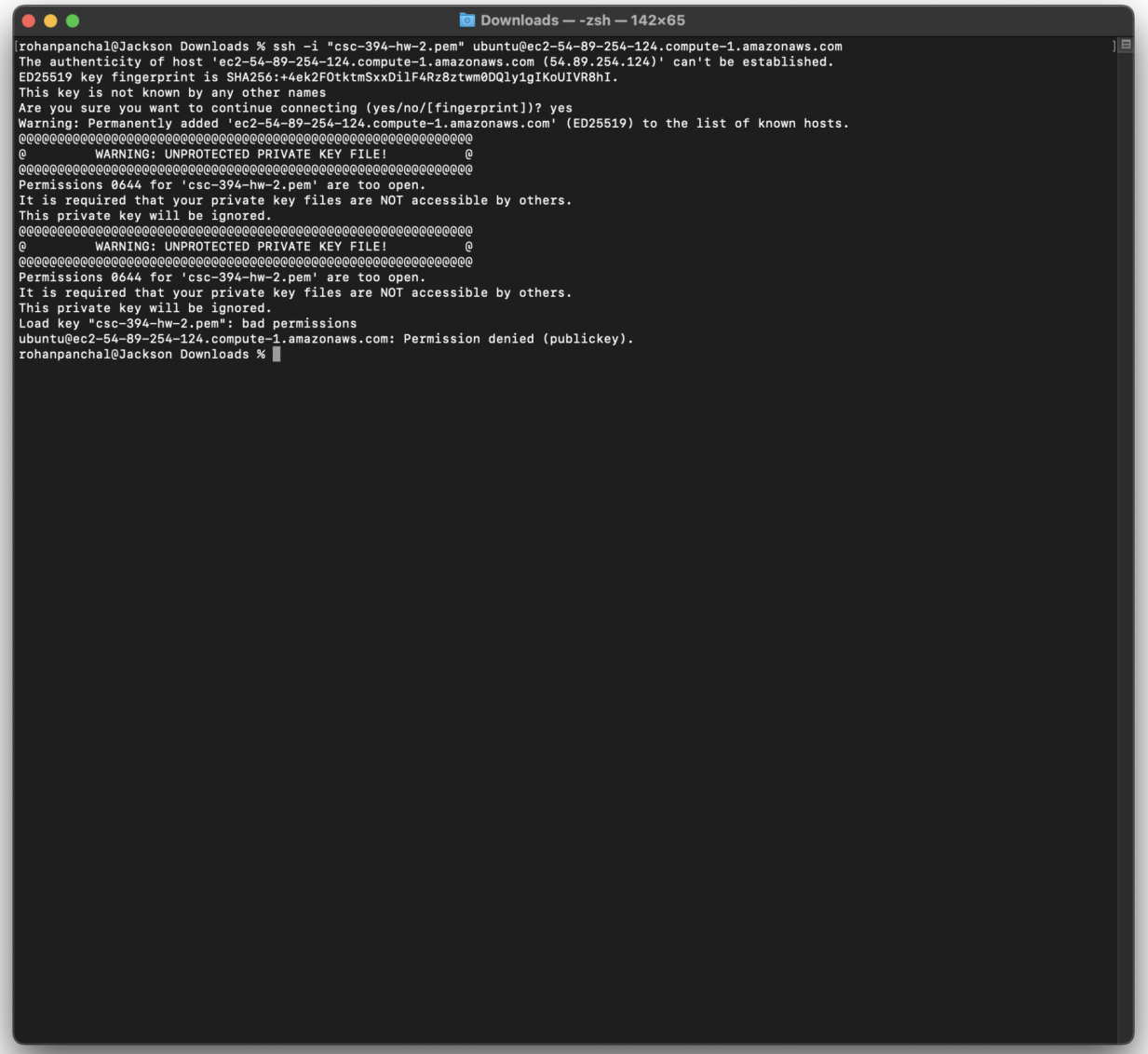
can launch a `terminal` or `command prompt` and connect to your linux instance!

Connect to your instance

In your terminal paste the command to connect via SSH to your EC2 instance.

Note:

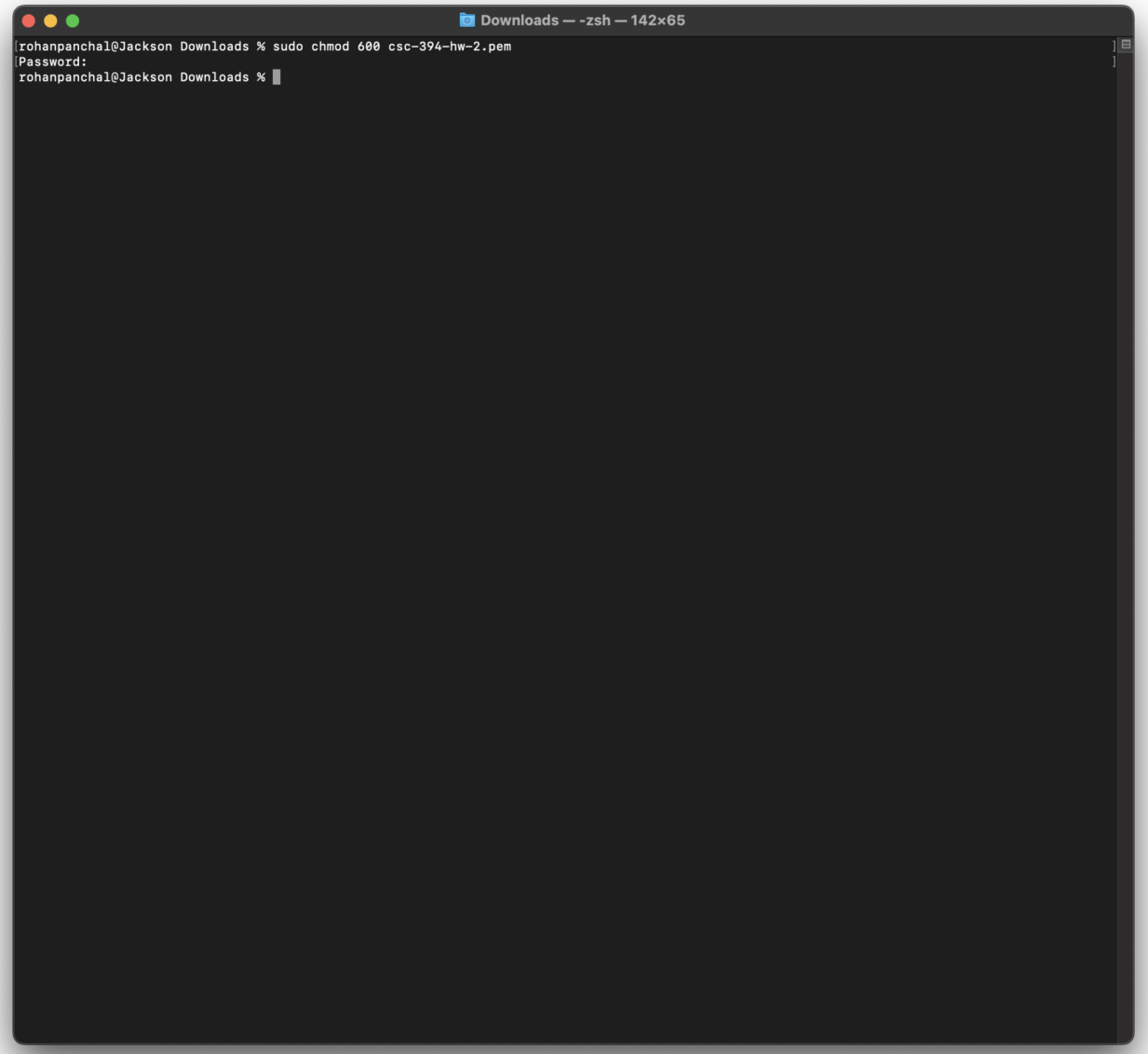
If you run into a WARNING as follows:



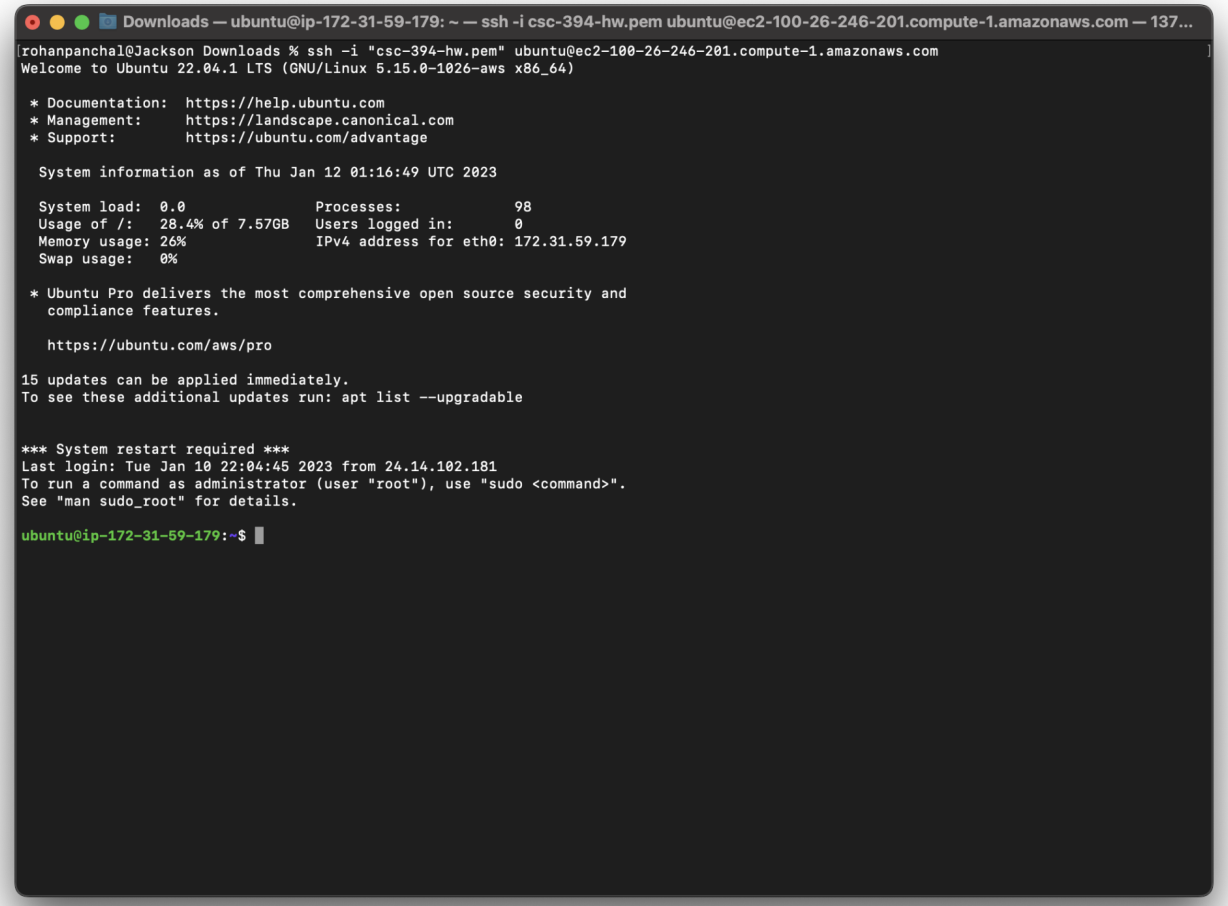
You need to modify the permissions of your .pem file.

Execute the following command

```
sudo chmod 400 csc-394-hw-2.pem
```



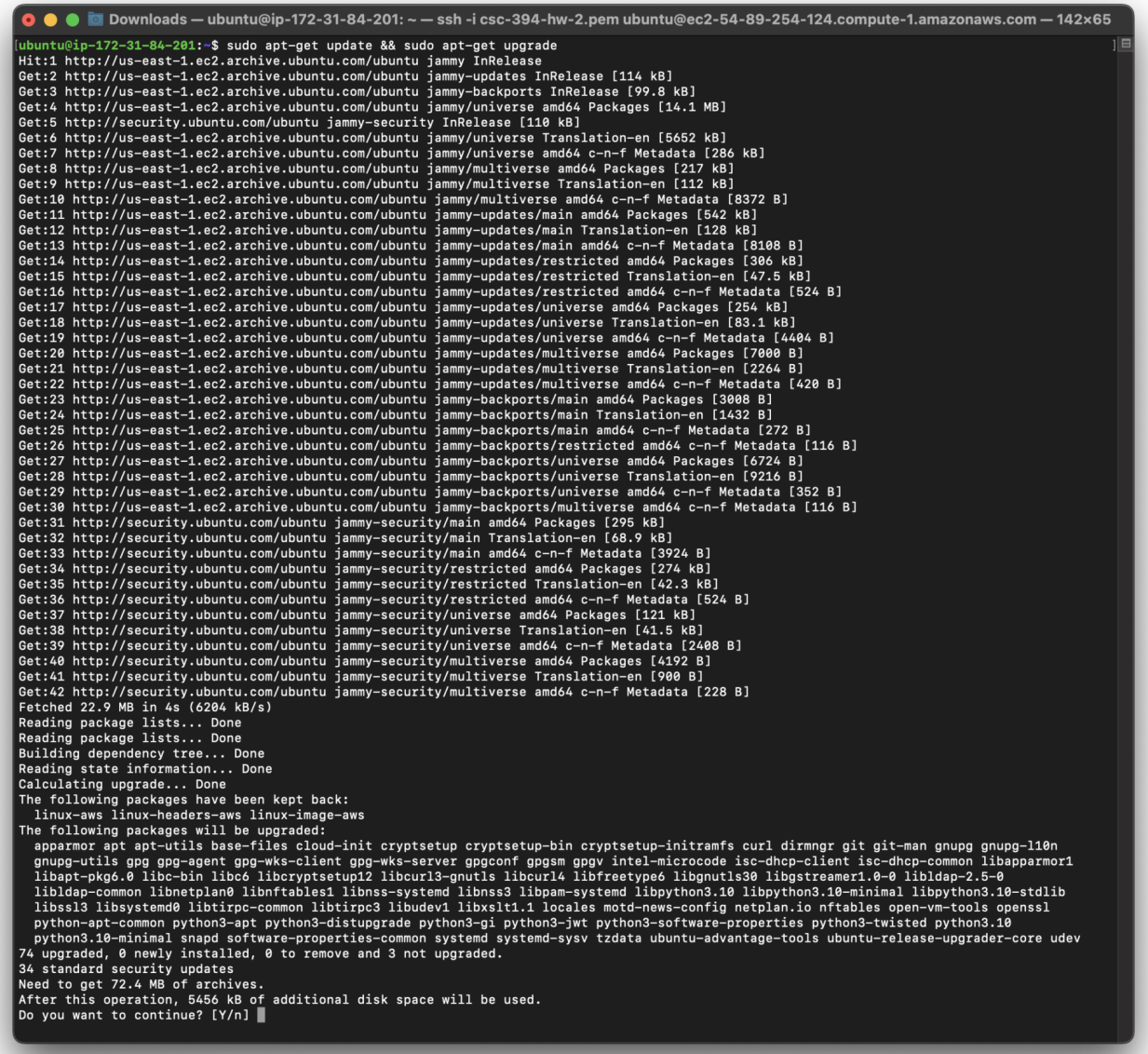
Reattempt connecting to your ssh instance and you should see the following:



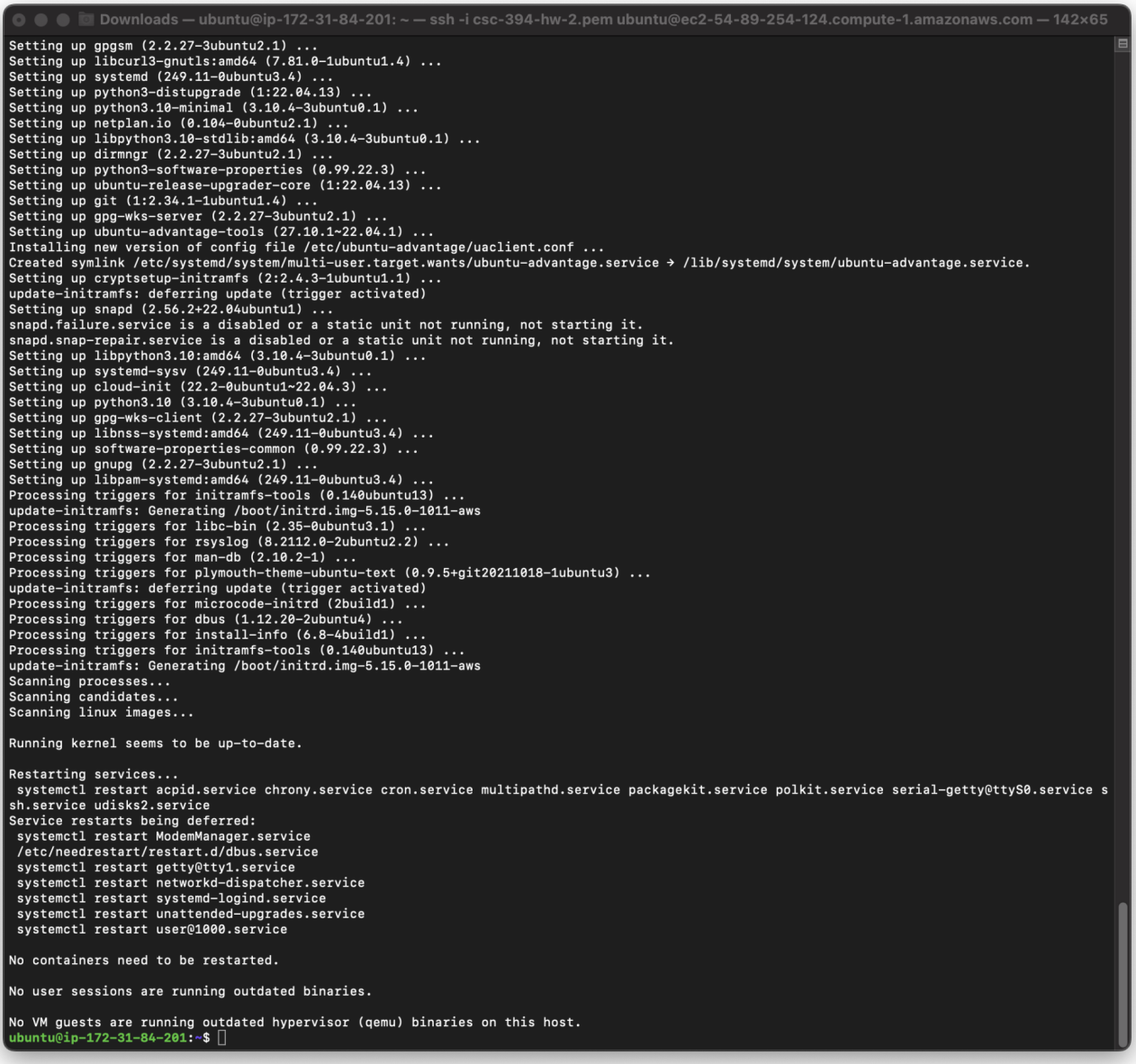
Update your EC2 instance's tooling

Execute the following command:

```
sudo apt-get update && sudo apt-get upgrade
```



This may take a few seconds. At some point you will see a prompt to continue. Type Y and hit enter.



EC2 box updated

Install development packages

Execute the following command:

```
sudo apt-get install build-essential python3-dev libpq-dev
```

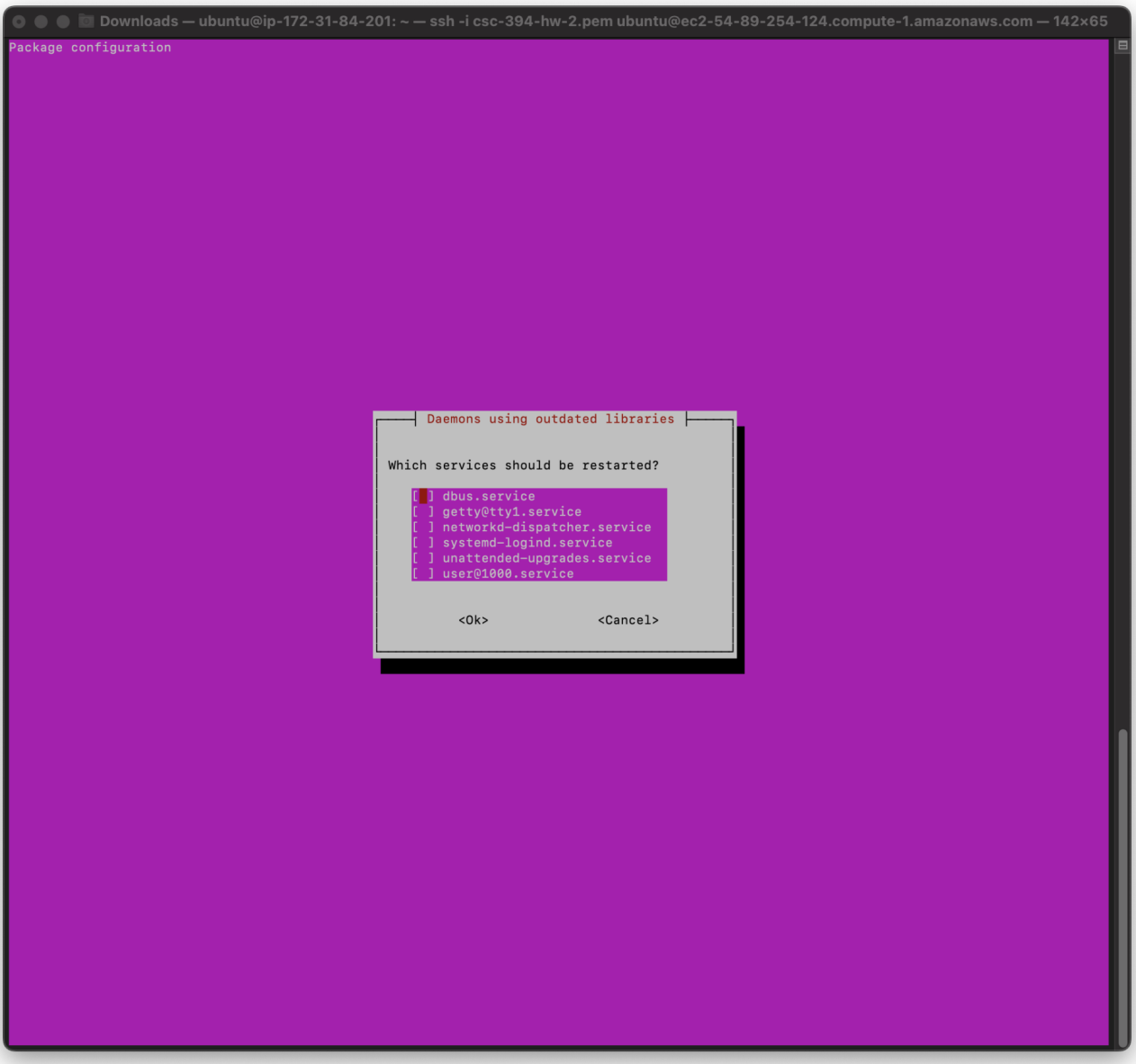
Install NGINX

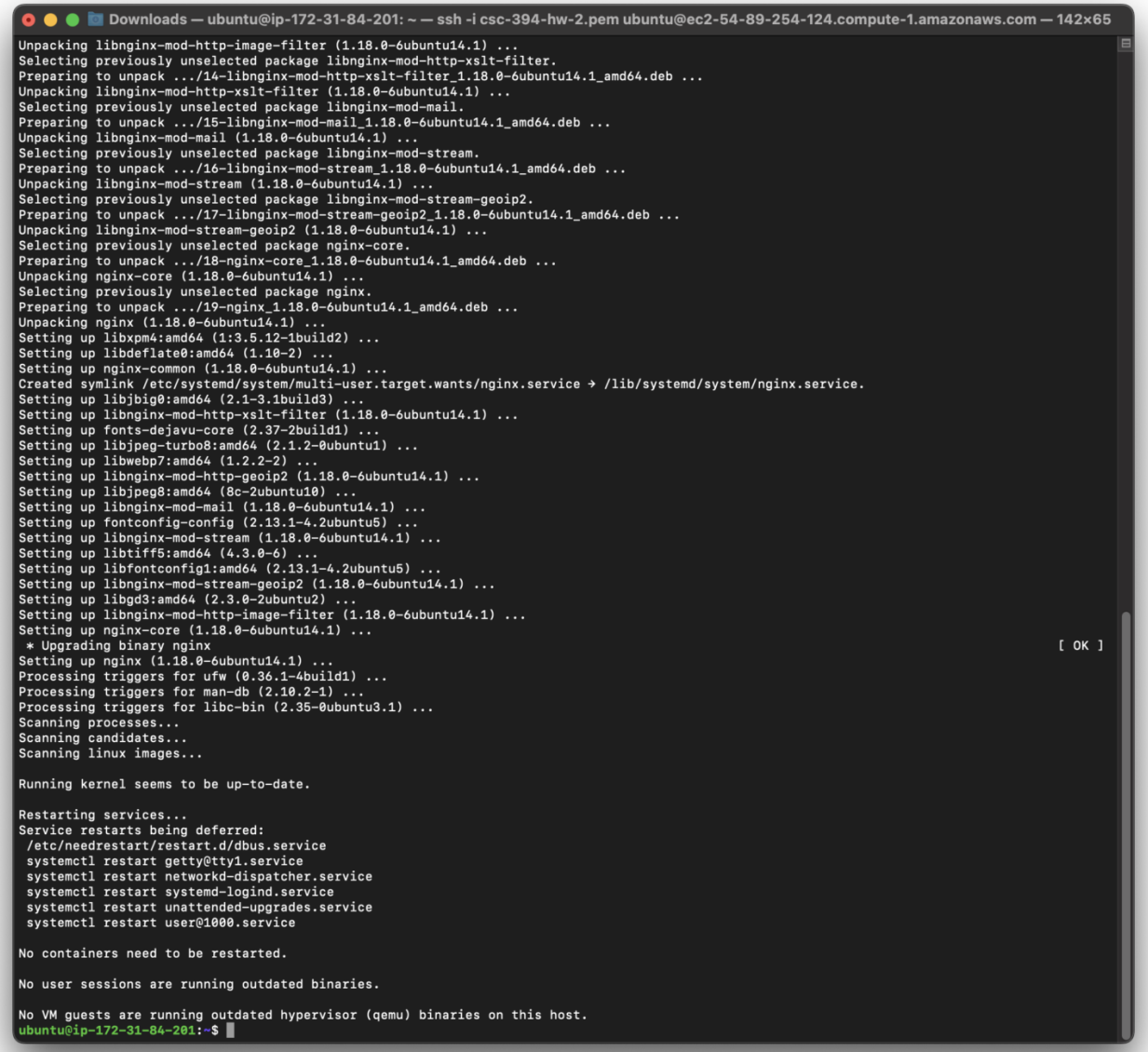
Execute the following command:

```
sudo apt-get install nginx
```

This may take a few seconds. At some point you will see a prompt to continue. Type Y and hit enter.

You may see a large purple screen asking you to restart services on your EC2 box. Just hit `Enter` .





NGINX installed!

Install PostgreSQL on your EC2 instance.

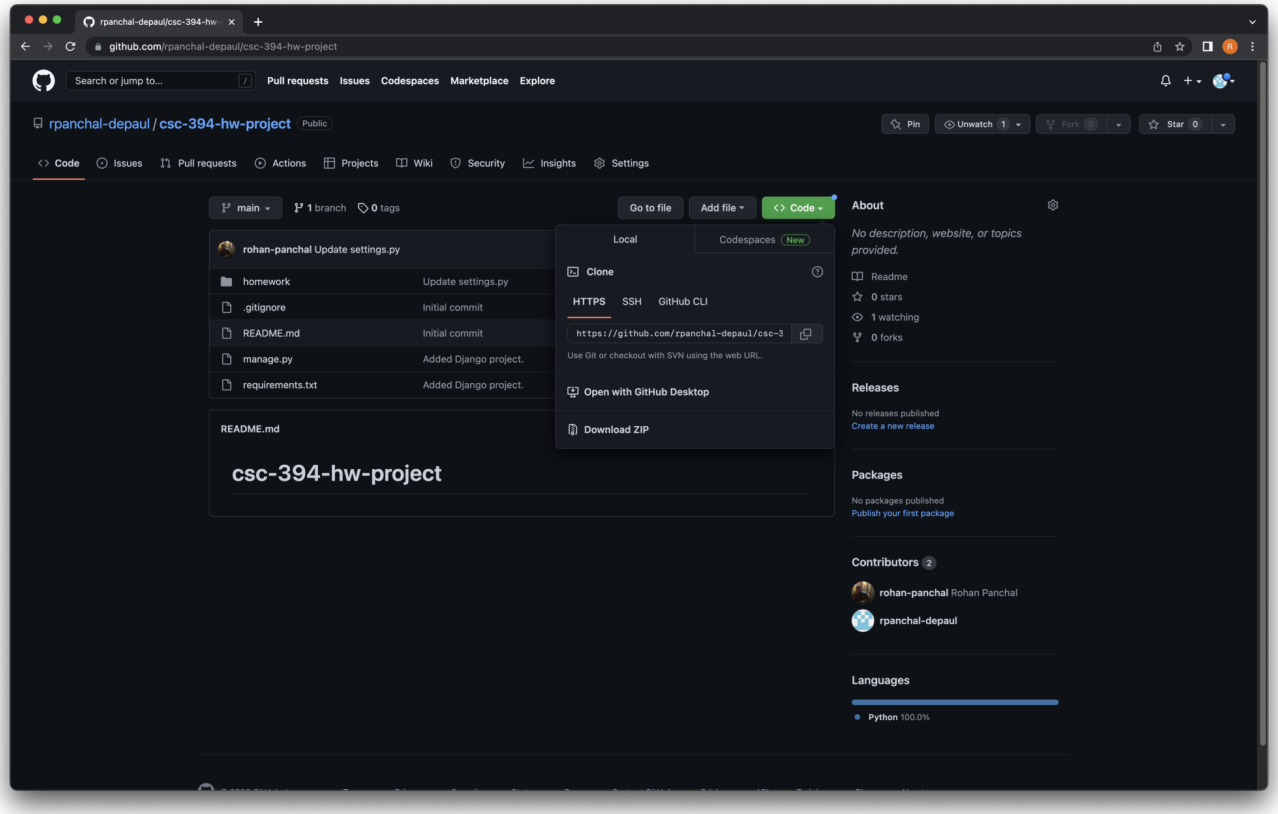
Execute the command:

```
sudo apt install postgresql postgresql-contrib libpq-dev
```

Clone your GitHub repo

Navigate to your GitHub repository you created for your django app.

Copy the url in the `Code` dropdown.



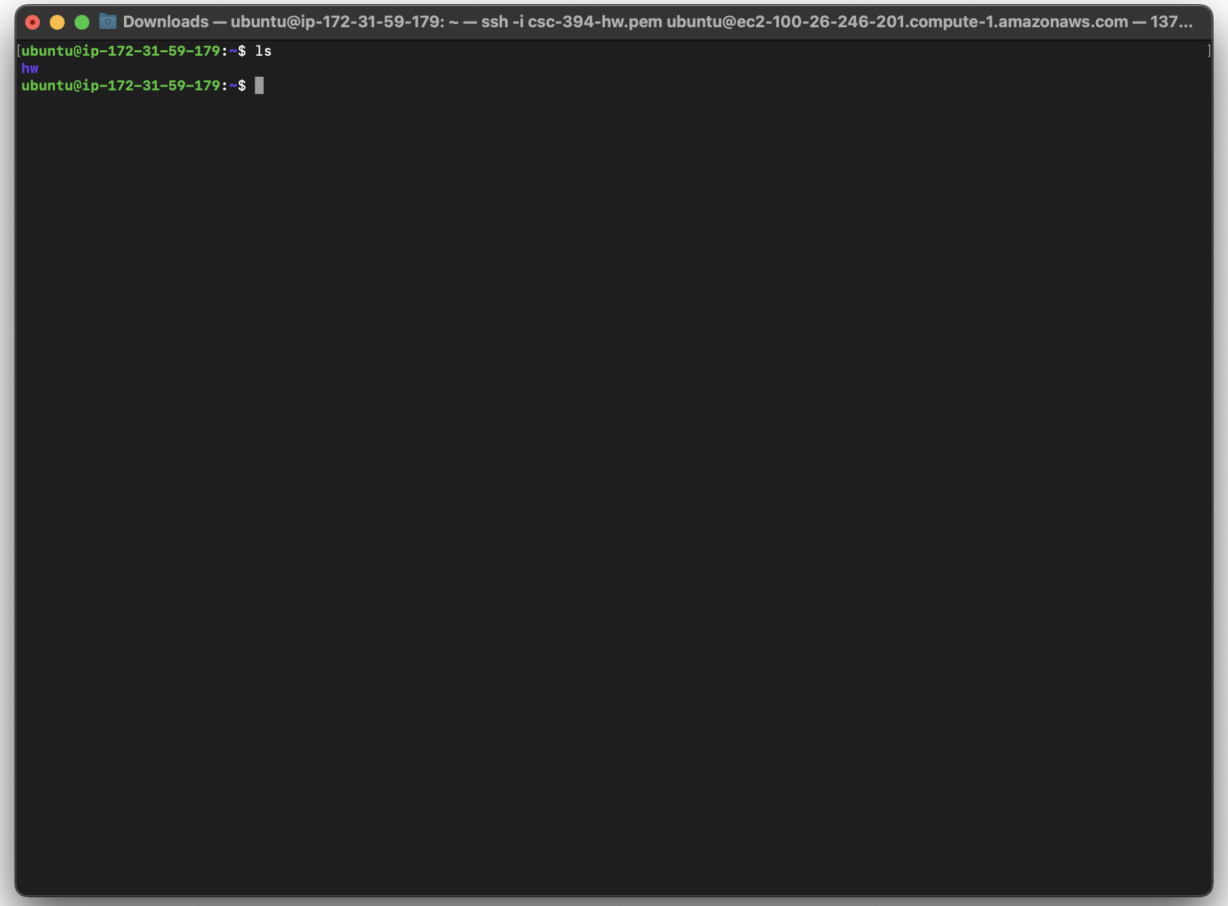
In your terminal window, execute the following command:

```
git clone {YOUR GITHUB REPO URL HERE} hw
```

Example:

```
git clone https://rpanchal-depaul/hw.git hw
```

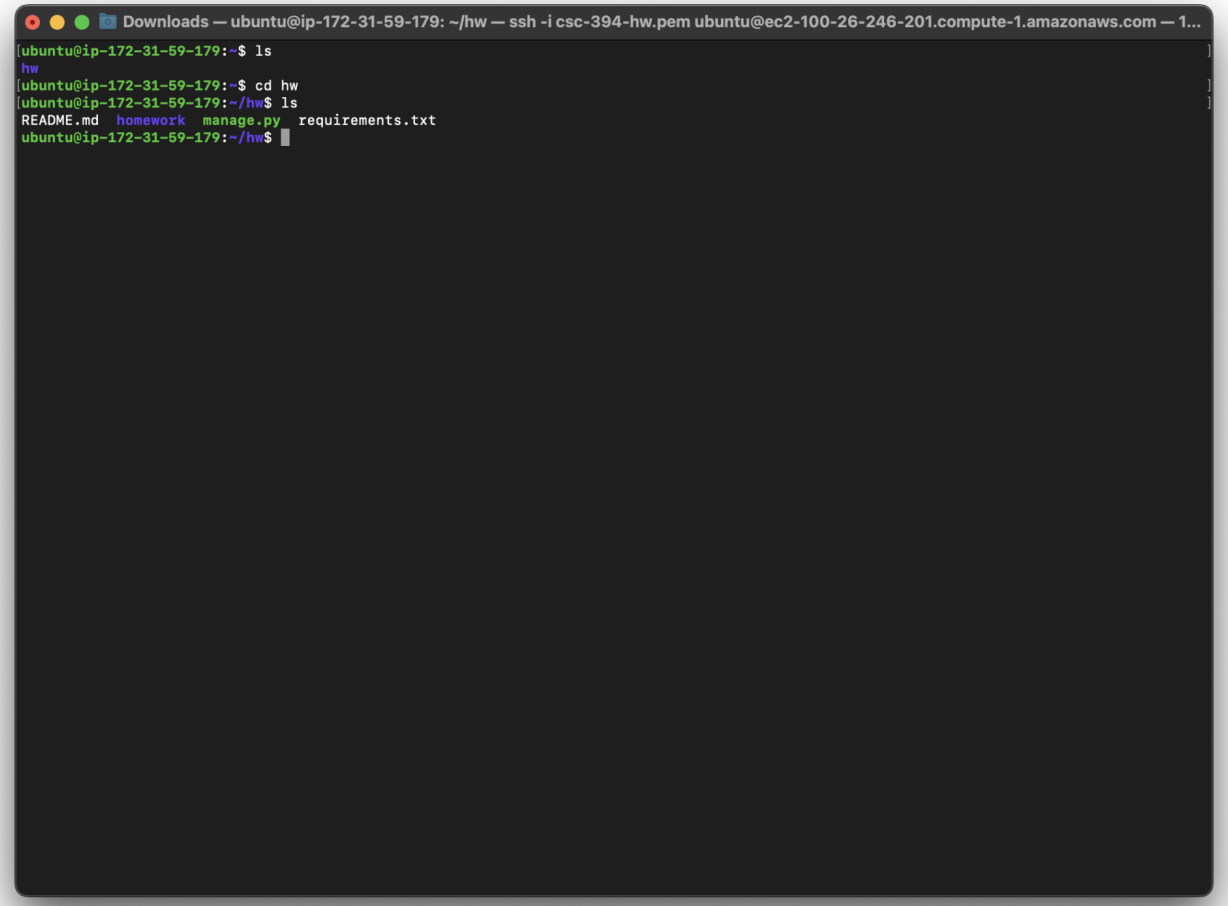
If you execute the `ls` command you should now see the following:



If we execute the `cd` command (change directory) as follows:

```
cd hw
```

And execute the command `ls` :



We will see the contents of our project here now!

Step 3 – Setup Django on your EC2 Server.

Let's install virtualenv on our EC2 server.

Execute the following command:

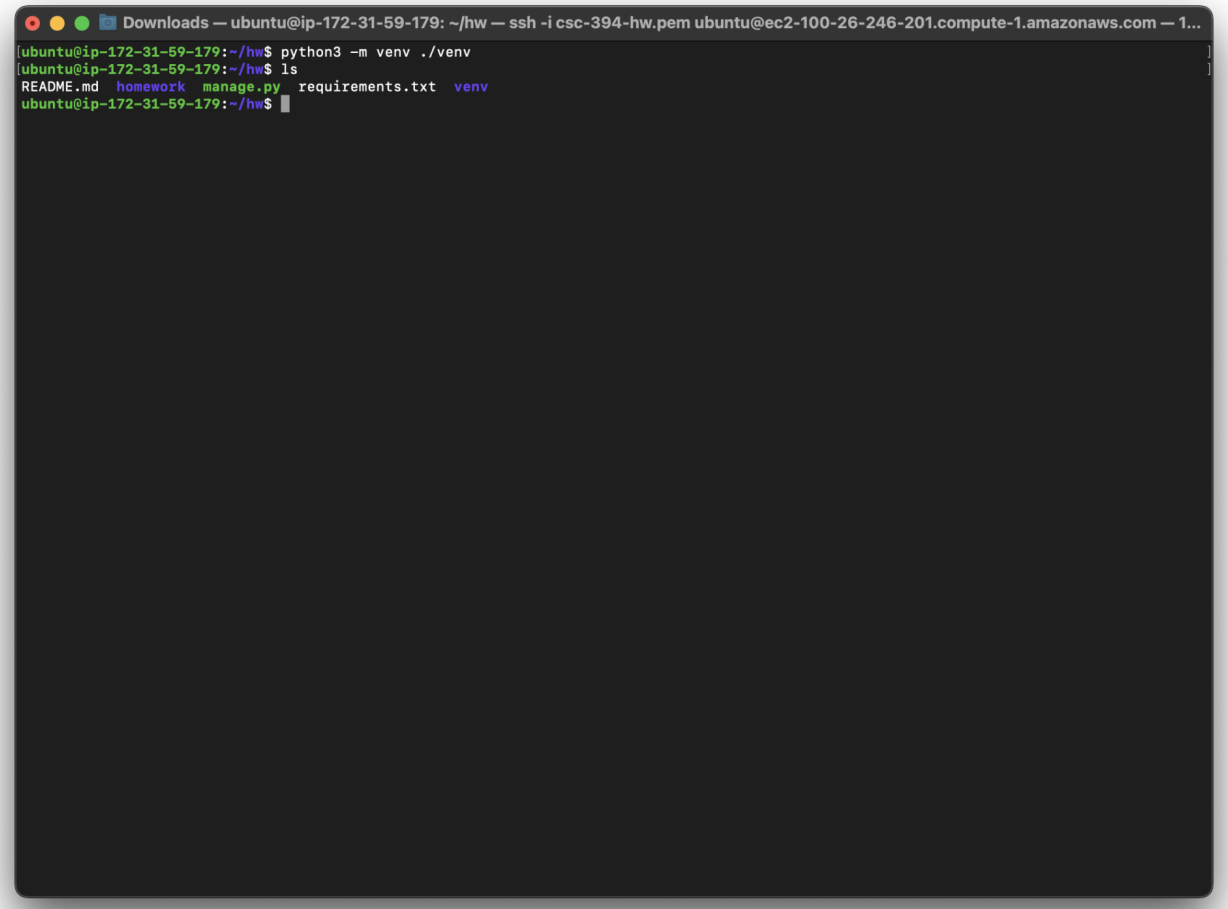
```
sudo apt install python3.10-venv
```

And then let's create a `virtualenv .`

Execute the following command:

```
python3 -m venv ./venv
```

We should now see the following directories:



Activate the `virtualenv` using the command:

```
source ./venv/bin/activate
```

And then install the requirements using the command:

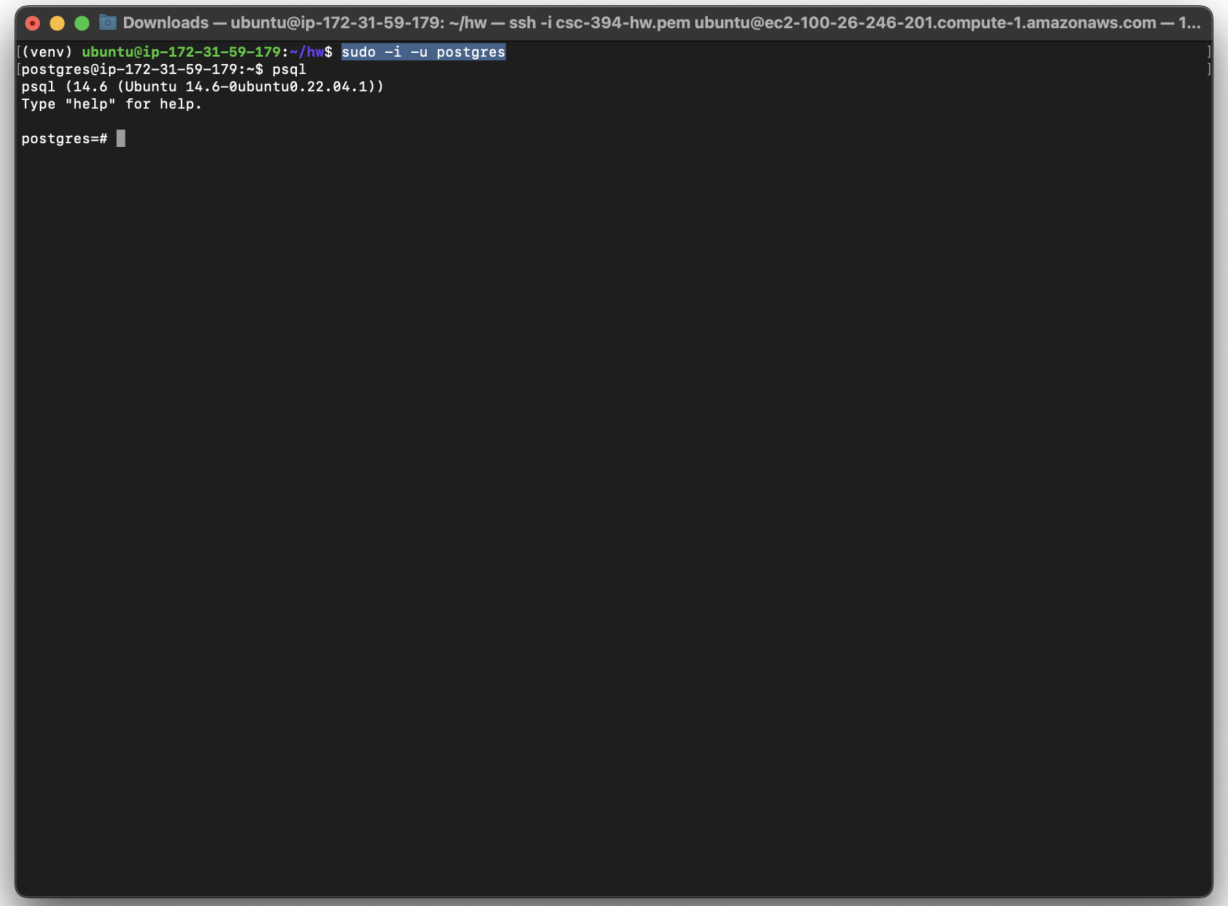
```
pip install -r requirements.txt
```

Setup postgres database on the server

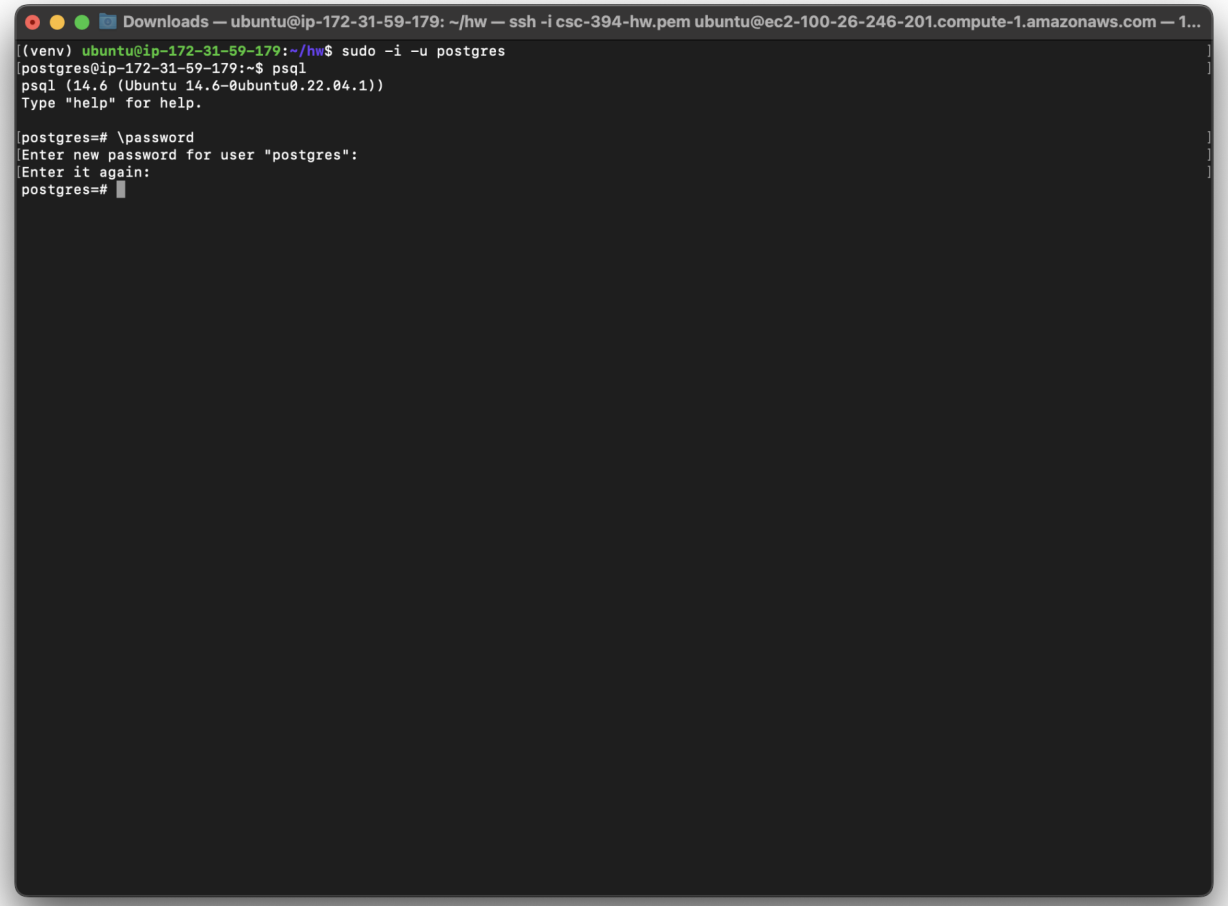
Sign in to the postgres user on Ubuntu:

```
sudo -i -u postgres
```

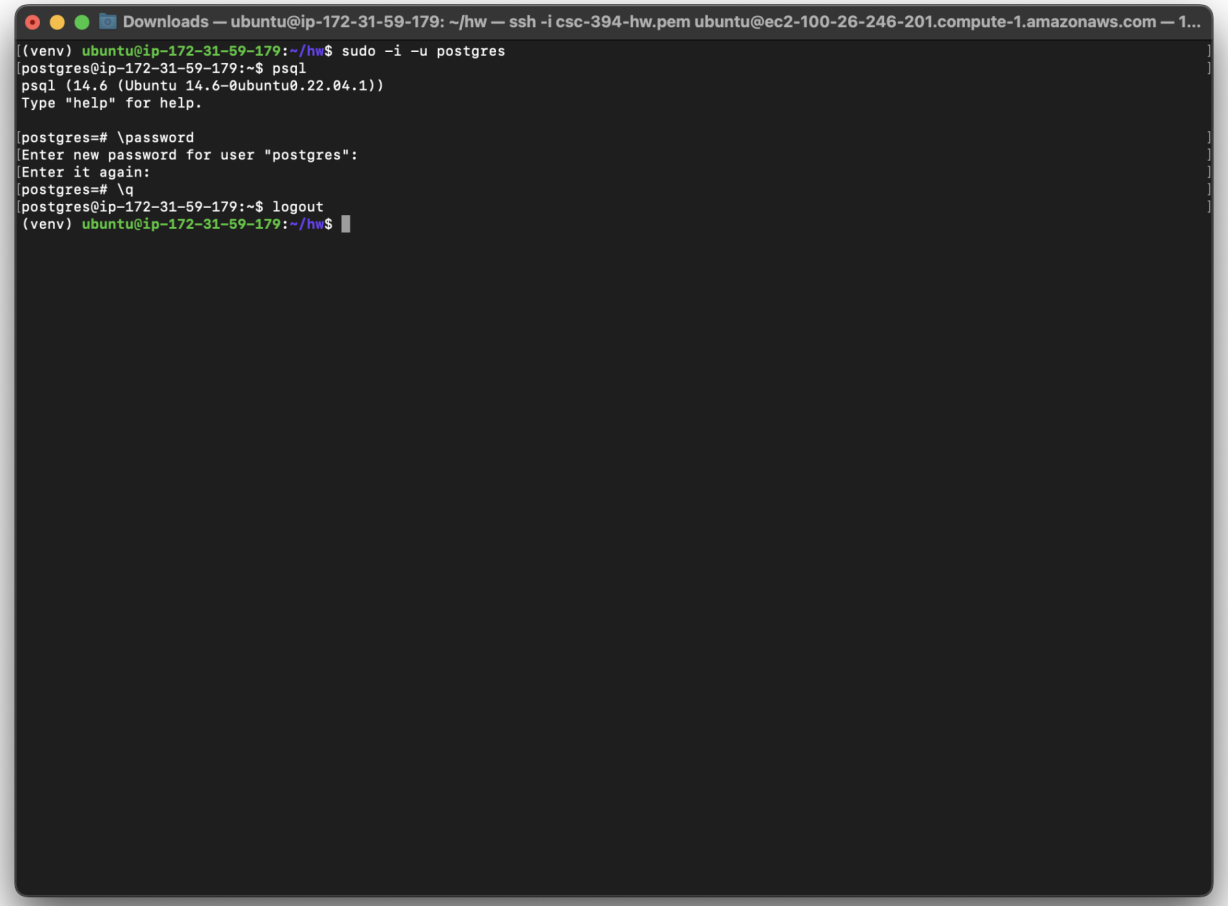
Followed by executing the command: `psql` .



Execute the command `\password` and fill in values for new passwords.

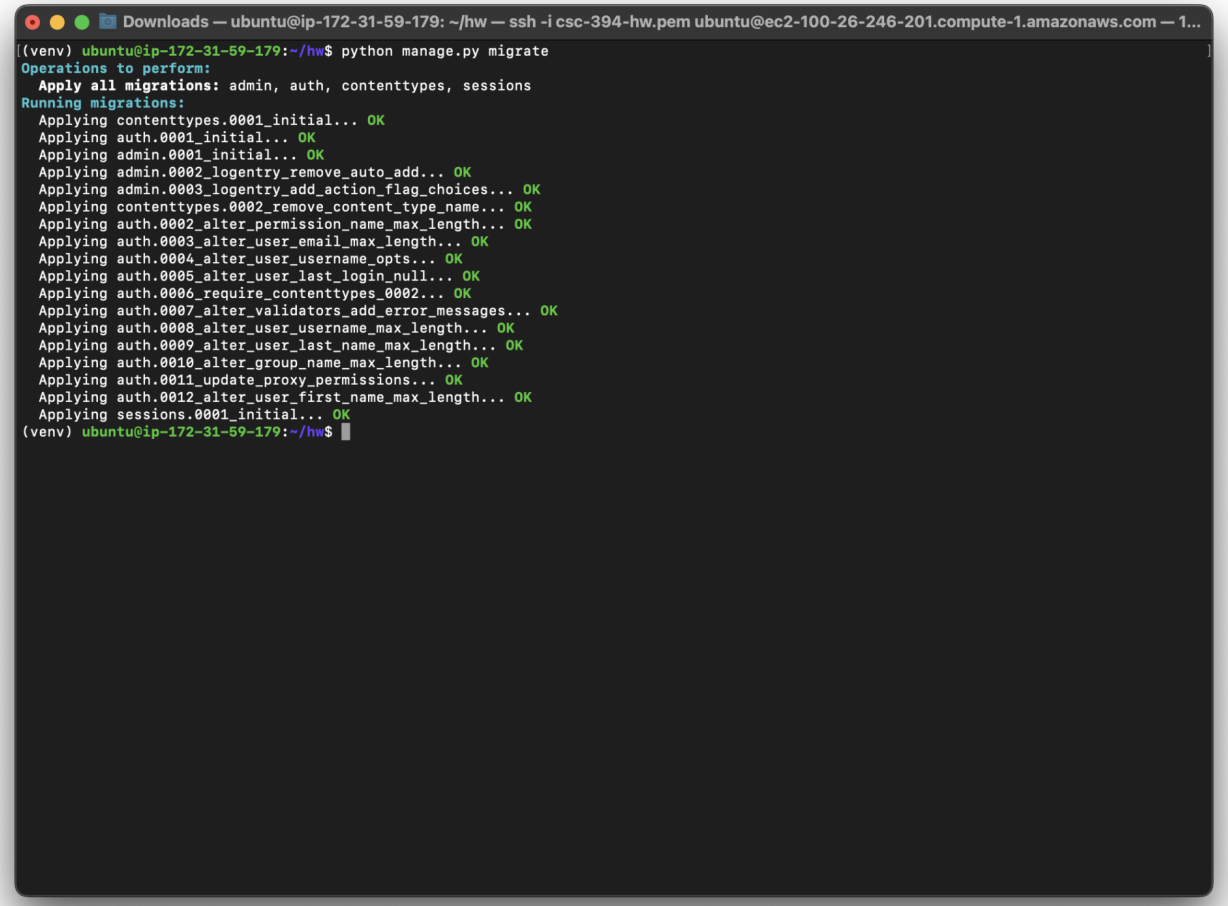


Quit by executing the command: `\q` followed by `logout`

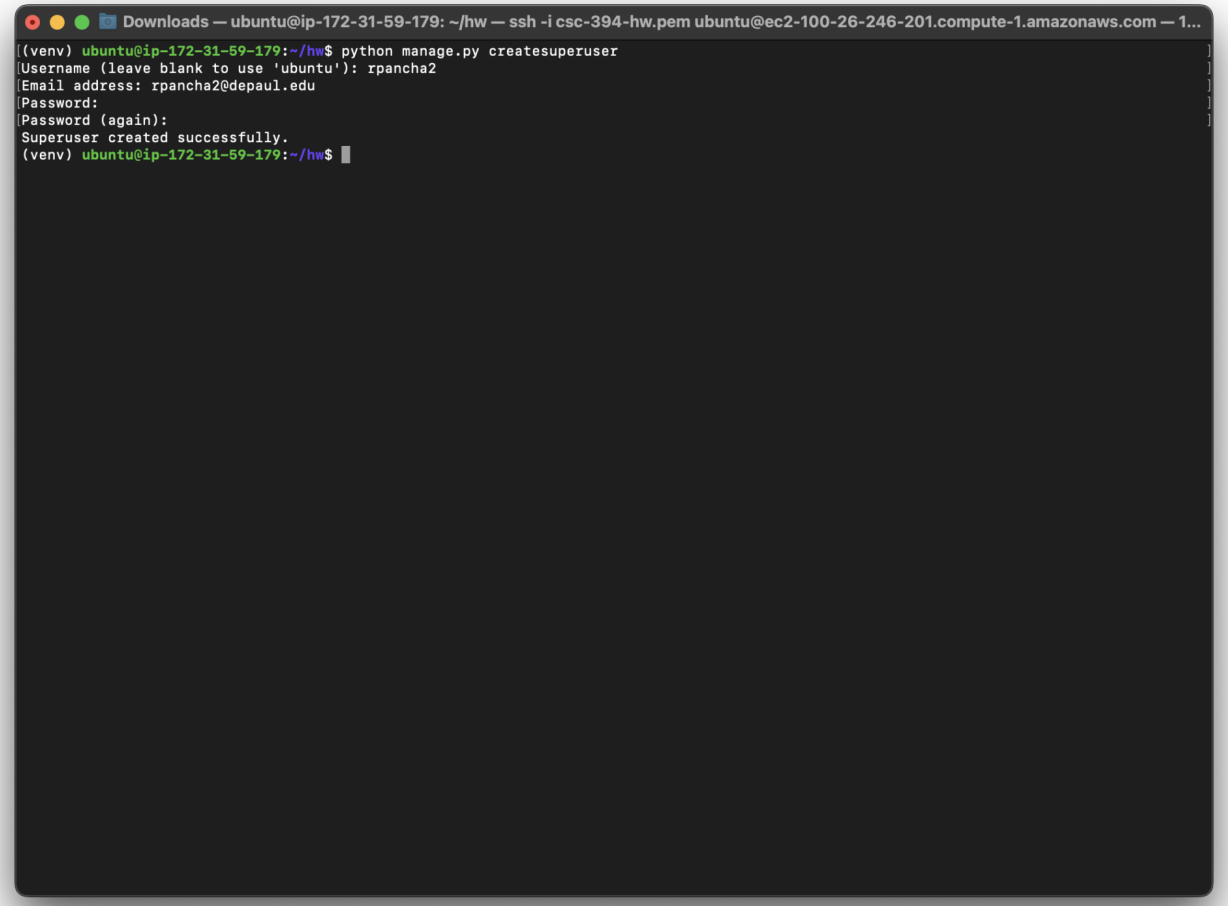


Run django migrations

Execute the django migration command: `python manage.py migrate .`



Followed by creating a super user: `python manage.py createsuperuser`



Configure Nginx

We have the database setup, let's point our nginx server to our django app.

First let's clear out the existing configuration for our `NginX` install.

```
sudo rm /etc/nginx/sites-enabled/default.net
sudo rm /etc/nginx/sites-available/default.conf
```

```
python manage.py
```

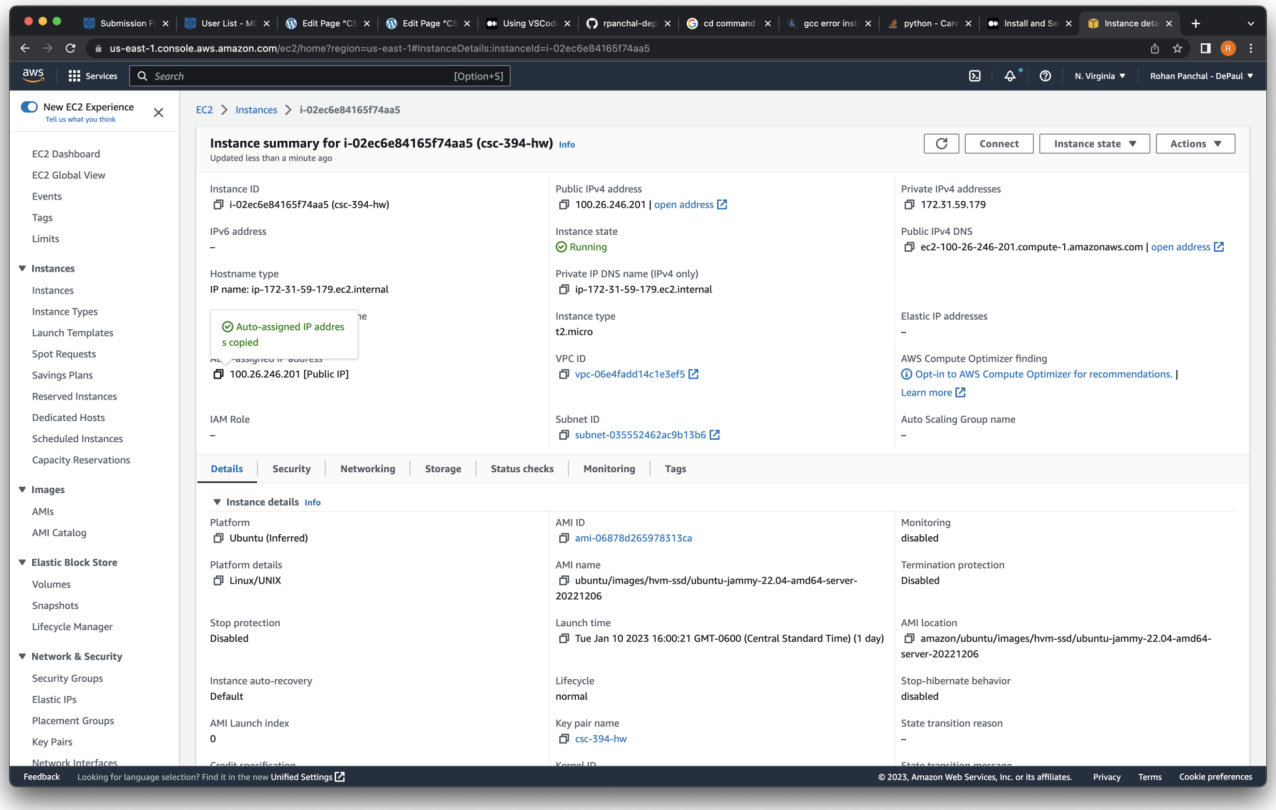
Now let's create a new configuration file.

```
sudo vi /etc/nginx/sites-available/hw.conf
```

And let's paste the following:

```
upstream hw {
    # Change the port based on what port your app is using.
    server localhost:8000;
}


server {
    listen 80;

    # Change the following IP Address to your machine's public IP
address.
    server_name 0.0.0.0;

    location / {
        proxy_pass http://hw/;

        proxy_set_header X-Real-IP $remote_addr;

        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;
    }
}
```

To find the appropriate IP address, for the `server_name`, look at the EC2 instance on AWS.



Save and quit by hitting the escape key, followed by hitting `:`, followed by typing `wq` and wacking `enter`.

Next, execute the following command:

```
sudo ln -v -s /etc/nginx/sites-available/hw.conf /etc/nginx/sites-enabled/hw.net
```

Wonderful, the appropriate configuration files have been created.

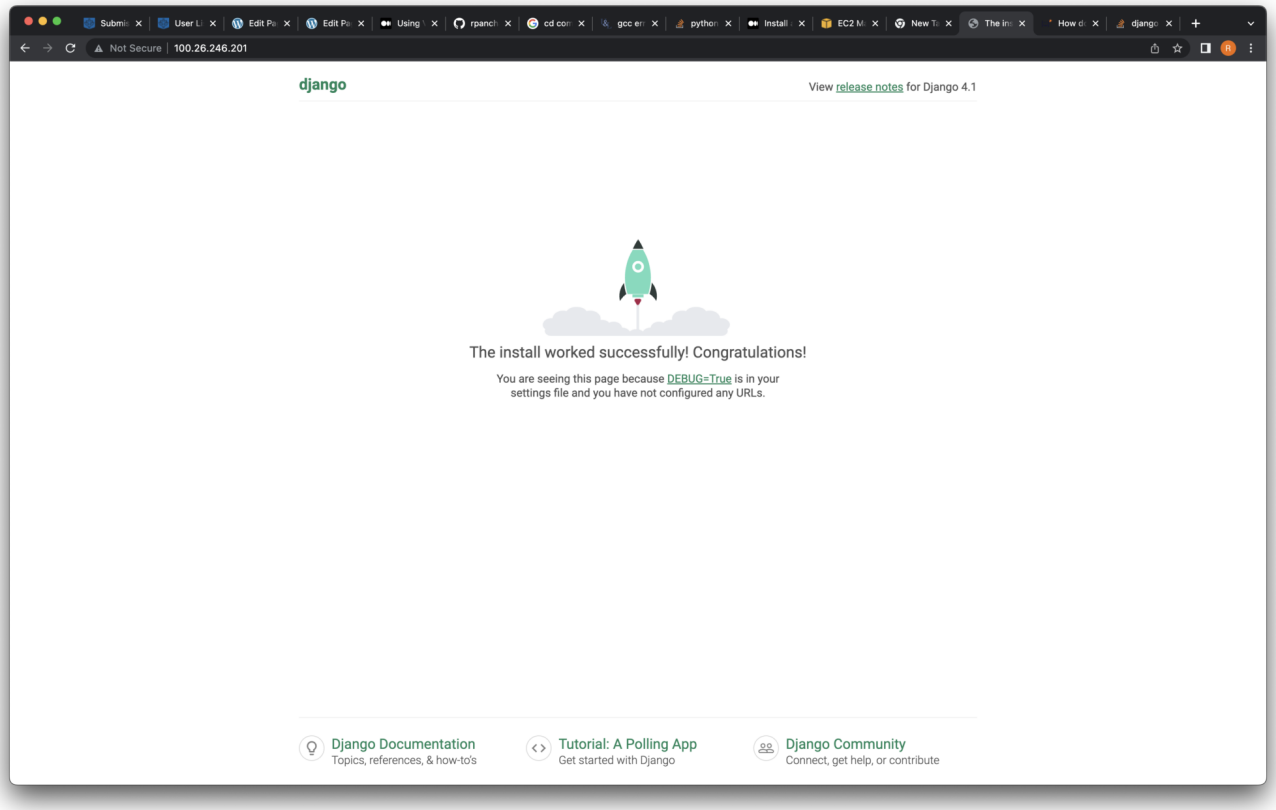
Let's restart the `NginX` server, execute the following command:

```
sudo service nginx restart
```

Back in the `hw` directory, start your django app!

```
python manage.py runserver
```

And now navigate to the ip address from earlier on your web browser, and see what happens!



Submission

Submit the following to D2L:

- A link to your IP Address
- A screenshot of your website running, showing the ip address in your url bar.

Theme: Noto Simple