Setting up our Django-server with Linode

# General information

We are going to setup our djano-server with linode. This is a server hosting site. We are going to create an account and make an ubuntu based server for this project

# Logging into our server with SSH

SSH stands for secure shell . We are going to use SSH in order to log in into our newly created server.

In order to do this, on windows we can use putty or we can install Bash (emulated linux terminal on windows). In this tutorial we are going to use BASH

In order to log in we just have to open BASH and type the following

ssh [root@172.104.135.165](mailto:root@172.104.135.165)

This will prompt us for the password we set up when we created our server.

After this we are going to do

apt-get upgrade && apt-get update

This is in order to get our OS up to date.

# Setting the hostname for the machine

We are going to change the hostname with

: hostnamectl set-hostname django-server

then type

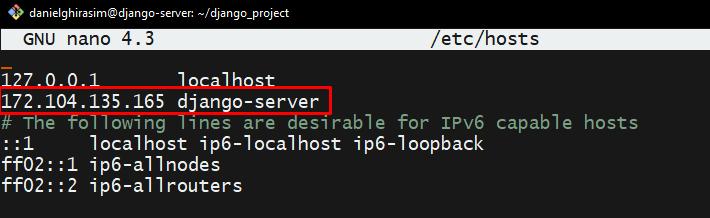
: hostname

and we will get our newly made hostname

# Adding the IP address to our hosts file

We are going to navigate to /etc/ and edit the hosts file with the nano text editor.

nano /etc/hosts/



# Adding a default user – Best practices

It is a good practice to not do stuff around in your server as root because many things can go wrong. A good practice is making another user that doesn’t have all the root access and using that instead.

To make a user you simply type:

: adduser username

Choose a password and fill out the additional stuff if you want. (This is optional)

## Adding SUDO to new user (root privileges)

Now we are going to add sudo privileges to our new user by typing

: adduser username sudo

Now this added our user to the group sudo

After we added this new user we can just use it log in to our server with this user.And the format is the following:

ssh newuser@serveripaddress

ssh danielghirasim@<http://172.104.135.165/>

# Setting up SSH Key based authentication

More on this (<https://www.youtube.com/watch?v=vpk_1gldOAE>)

Next, we are going to set up SSH Key based authentication this helps us logging in into our server by using a SSH key instead of using passwords. This is a good practice and also more secure and convenient.

On our webservers home folder we want to create a new file called ssh.

To check what folder we are in we can just type pwd into the terminal



## Making the ssh file

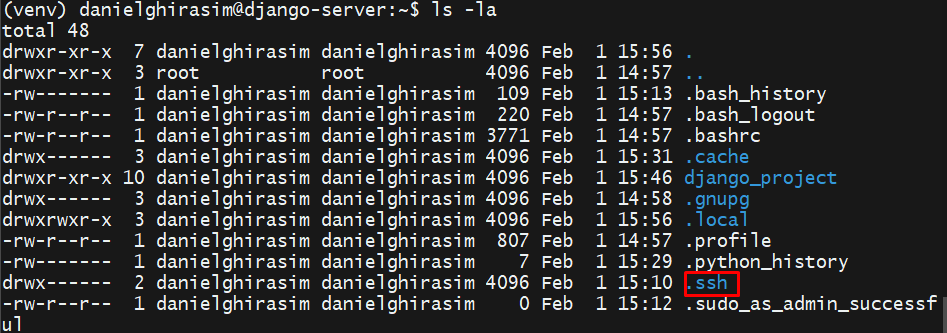
mkdir –p ~/.ssh

mkdir 🡪 make directory

-p 🡪 Make whole folder tree

~ 🡪 means our current folder

.ssh 🡪 the actual file

Now if we type ls -la we will see that our .ssh file is there

## Making the SSH key on the machine we want to log in from

We are going to open bash and type

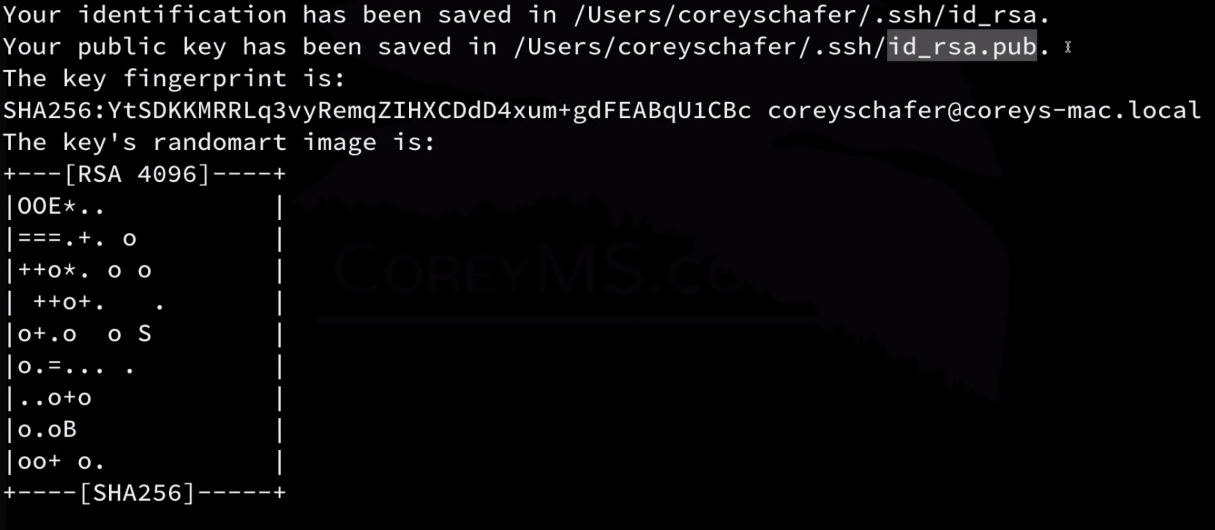
ssh-keygen -b 4096

-b stands for bytes

4096 stands for the size

After this we are prompted for the directory but you can just leave the default, then it’s going to ask you for a password you can also leave that blank. (adding one makes it even more secure).

After this 2 keys are created



Now we want to put the public key on the server so we can log in without a server. By adding the public key to the server and the other key being on the machine we want to login from, they are gonna match and we can the login without a password

## Sending the public key to the server

On windows we can use FTP or filezilla to transfer this public key to the server but we are going to use bash for this also.

We are going to use the scp command. SCP stands for Secure Copy:

scp /c/users/myuser/ssh/public key danielghirasim@http://172.104.135.165:~/.ssh/authorized\_keys

So SCP > FILEPATH+FILEYOUWANT TO SSH to [user@ip.com:locationonthe](mailto:user@ip.com:locationonthe) server where you want it saved.

If everything goes fine a password is prompted and the file should be there

Check with ls -la

## Updating permissions for users using SSH

Here we are going to set the permissions for the ssh directory on our server so that specific users have read/write capabilities

To do that we are going to type:

sudo chmod 700 ~/.ssh/

chmod - In Unix-like operating systems, the chmod command sets the permissions of files or directories.

700 – number of permissions

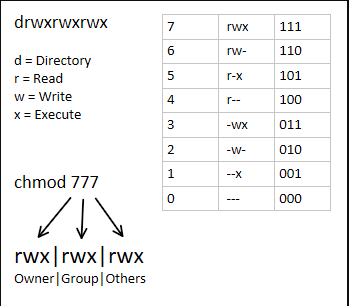
Then we are going to do 600 on all of the files of the directories.

sudo chmod 600 ~/.ssh/

Now we should be able to login without a password

### ReadWriteExecute graph

The first number of the 3-digit number is the permission for the owner, the second is the perm for the group and the 3rd is the permissions for everyone else



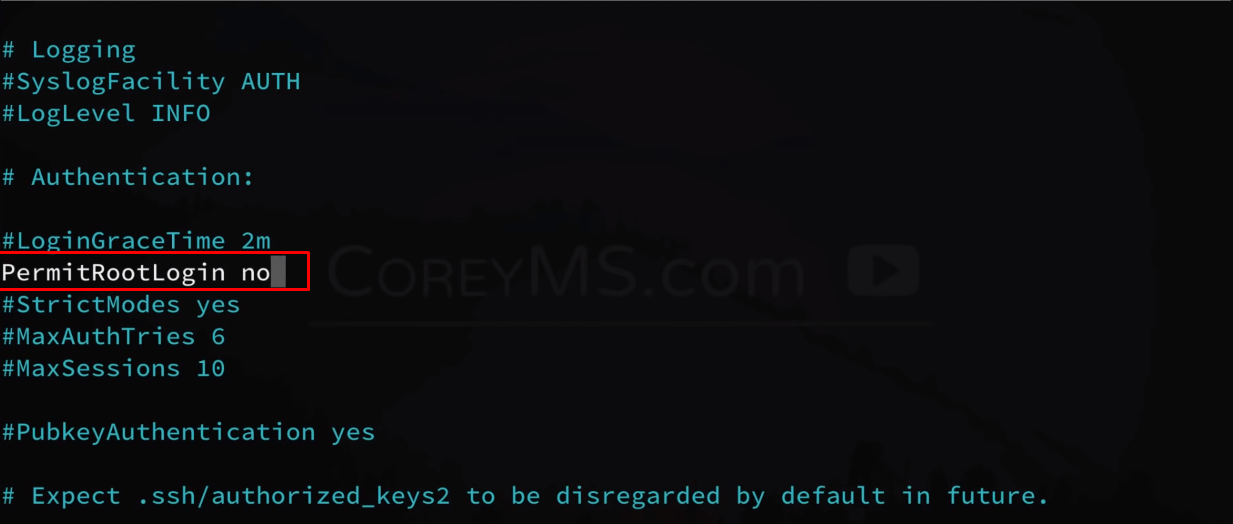
## Enabling only SSH key login to our server

This is another good practice, since we set up SSH key based authentication we want users that want to login to our server just to use the SSH key, so if they don’t have one they cannot log in in order to do this we have to changes a couple lines of code in our SSH config file

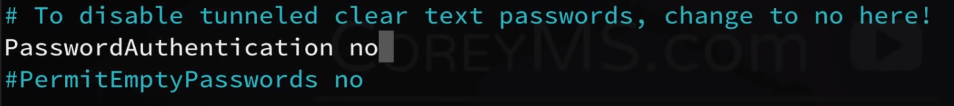
sudo nano /etc/ssh/sshd\_config



We are going to disable PermitRootLogin which doesn’t let us logging in at root as first so root@myipaddress won’t work



And we also gonna set password authentication to no



Save with CTRL + X

No we have to restart the ssh service with the following command:



# Setting up a firewall – Good practice

We are going to set up a firewall for more security, in order to do this we are installing ufw – stands for uncomplicated firewall.



Now we are going to set up a few basic rules.



Allows outgoing traffic

Denying incoming traffic but we are going to set a few exceptions



We must add some exceptions to this in order for us to log in and the server to properly work.

To allow SSH

sudo ufw allow ssh



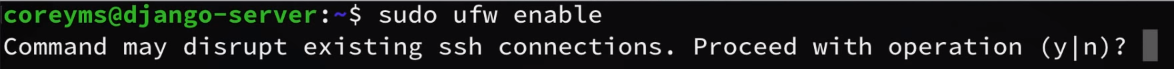
We are not going to allow http just yet but we are going to allow just a certain port until we finish with the server for testing

Our Django development server runs on port 8000 so we are going to allow that port for now.

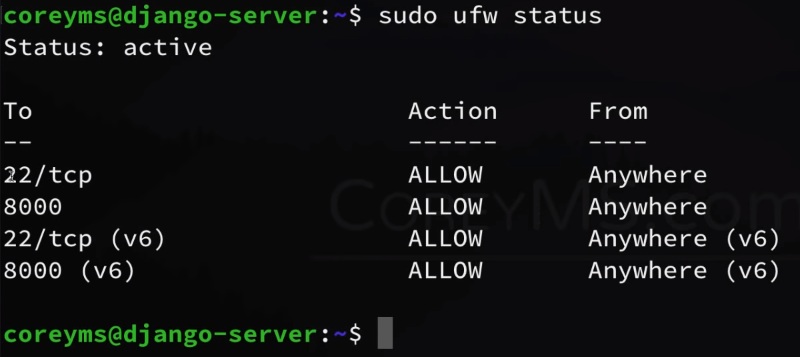


If this is all working, later we are going to allow HTTP incom ing requests trough port 80.

**All that is left is to activate our firewall**



If everything was setup correctly you shouldn’t be disconnected.

And now we can check ufw status with:

Here we can see all our allowed ports:

22 – is the ssh port

8000 – Django default port

# Transfering our Django project to the server

In this step we are going to transfer the project to the server . For this we are going to use the scp command again from our local machine because it looks cool. But first we are going to make a requirments.txt file for our existing Django\_env.

## Making a requirements.txt file

We are going to activate our Django\_env and make a requirments.txt file because this way it’s easier to set it up on our server.

In order to do this we have to activate the virtual environment and type:

pip freeze > requirments.txt

This will make us a file with all the requirements of the environment. Copy this file to the Django\_project and proceed with the copy

## Copying the project with SCP

Navigate to the project directory folder and type

scp -r project\_name danielghirasim@myip:~/

-r 🡪 stands for recursive copy, this means it will copy everything within the directory

and we are going to place it on our server in ~/ which is our home folder

Now we can check with ls if it’s there



## Installing Python3-Pip and other required stuff

We are going to install all the applications we need in order for our Django server to run on our linux server

sudo apt-get install python3-pip

sudo apt-get install python3-venv

## Creating a new venv

Now we are going to make a new virtual environment on our linux server inside of our newly copied Django\_project folder:

python3 -m venv Django\_project/venv

This makes a new virtual environment in our Django projects folder

Now we are going to activate that venv with:

cd Django\_project

source venv/bin/activate

If everything goes right, the environment should be activated.



Installing the requirements:

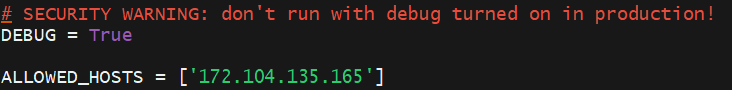
Just simply say:

pip install -r requirements.txt

-r 🡪 this says that pip expects a requirements.txt file.

# Running the Django\_project from our linux server

Before we run this we have to make some modifications in our settings.py file. We have to add our ip into ALLOWED\_HOSTS – This is for security reasons



## Migrating our static files

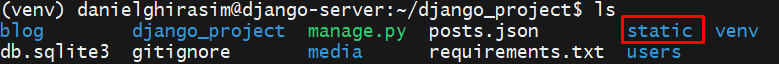
Static files are handled differently in development then in production so in production we have to put in a place where the static files are going to exist and that is the STATIC\_ROOT, we are going to add this to our settings.py files also



And now we use a simple command within our shell to migrate all of the existing static files to this folder by typing:

**python manage.py collectstatic**

Now when we check the root of our Django\_project we will see all of the static files migrated there.



## Running the server after the modifications:

To run the server we aren’t going to use a simple python manage.py runserver, instead we are going to type:

python manage.py runserver 0.0.0.0:8000

We opened up port 8000 on the server that’s why we use that.

# Apache

We aren’t going to run our server with Django’s built in server, instead we are going to use Apache (alternatives: enginex etc) because it’s more secure and reliable, in order to set this up we want to install a couple of things before.