Set up Django Project:

*{{ }} - These are place holders*

\*\*Make sure your djangoEnv is running\*\*

\*\*Make sure you’re in the folder your project is in\*\*

Step 1: Getting Started

pip freeze > requirements.txt

*In your text editor, open your requirements.txt file and, if they exist, remove pygraphviz, pydot and anything with MySQL in it.*

## Step 2: Committing

## ***Important!*** *We’re about to initialize a new git repo. Your git repo must be initialized within the outer project folder. This is the same level as your manage.py file.*

touch .gitignore

*Open you’re .gitignore file in your text editor and add the lines:*

\*.pyc

venv/

*We know this is familiar, but here’s a reminder of how to initialize a new repo:*

git init

git add .

git commit -m "initial commit"

*Create a new github repo and, back in terminal run these commands, replacing the repo url with your own.*

git remote add origin ***{{GITHUB URL}}***

git push origin master

## Step 3: Creating an EC2 server instance

1. Click EC2
2. Click Launch Instance
3. Select Ubuntu Server
4. Select t2.micro
5. Click Review and Launch
6. Click Edit security groups

*SSH option should be there already. Reset SSH source from the dropdown menu to MyIP. This is the ip address of your current location. If you go home, for example, you will have to set this again to get it to be your home ip.*

1. Click the add a rule button twice to add HTTP and HTTPS, source set to Anywhere, and then click Review and Launch

*You’ll be asked to create a key file. This is what will let us connect and control the server from our local machine.*

*Name your pem key whatever makes the most sense to you as shown in the next step. Give it a generic name, not the name of your project, as we will be re-using this instance.*

*The key will automatically be saved to your downloads folder when you click Download Key Pair, but you will want to move it. See the next item for more information on this critical step.*

1. Click Launch Instances

*This next part is very important! Put your pem key in a file that has no chance of EVER being pushed to github. You should not send this file via email, or in any other way make it publicly available:*

*After launching your instance, you will see a rather confusing screen with some information, as shown below. In order to move on, scroll to the bottom of the page and confirm that you would like to view your instance.*

*Once you have several instances running, you will want to be able to identify what your different instances are for. We have the option of naming our instance, so let’s do so now by clicking on our instance’s name column as shown.*

Step 4: Connecting to your remote server

*Back in your terminal, cd to the folder that holds the key file you just downloaded.*

cd /projects/AWS

*Back in your AWS console click the connect button at the top of your screen here:*

* Run the chmod command in your terminal.
* Copy and paste the line starting with ssh (below) and paste the text into your terminal.

You might have to type yes and wait for a few seconds to a minute before you are connected, but if all goes well, you should be in your Ubuntu cloud server. Your terminal should show something like this in the far left of your prompt:

ubuntu@54.162.31.253:~$ #Commands you write appear here

Step 5: Server Configuration

*First, let’s install python, python dev, pip, nginx, and git*

sudo apt-get update && sudo apt-get install python-pip python-dev nginx git

*Now that we’ve installed some packages using apt-get, let’s run update again to make sure apt-get knows we’ve done those installations.*

*In addition, you’ll use your newly installed pip to install the virtual environment program so that you can now build a brand new virtual environment on your new computer.*

sudo apt-get update && sudo pip install virtualenv

*Now you’ll clone your project from GitHub.*

git clone ***{{GITHUB URL}}***

*Navigate into this project and run ls in your terminal. If you don’t see manage.py as one of the files, STOP. Review the setting up GitHub/Git pieces from earlier.*

cd **{{repoName}}**

*If everything looks good, let’s make that virtual environment, and activate it.*

virtualenv venv

source venv/bin/activate

pip install -r requirements.txt && pip install django-extensions && pip install gunicorn

Step 6: Editing Settings

*IMPORTANT: NOTE FOR STEP 6 and BELOW*

*Anywhere you see {{myRepoName}} – replace that whole thing INCLUDING the {{}} with your outer folder name.*

*Anywhere you see {{projectName}} – replace that whole thing INCLUDING the {{}} with the project folder name.*

*Anywhere you see {{yourEC2.public.ip}} – replace that whole thing INCLUDING the {{}} with the public IP address of your newly created server.*

*\*If you named your repo something different from your project, the repo name and project name may be different, but it is ok if they are the same.*

*Navigate into your main project directory (where settings.py lives). We’re going to use a built-in text editor in the terminal to update the code in settings.py. For example:*

cd **{{projectName}}**

sudo vim settings.py

*You'll need to add a line that allows you to serve static content. You'll also need to modify a couple of lines, as follows:*

*# Inside settings.py*

*# modify these lines*

DEBUG = False

ALLOWED\_HOSTS = ['**{{yourEC2.public.ip}}**']

# add the line below to the bottom of the file

STATIC\_ROOT = os.path.join(BASE\_DIR, "static/")

Save your changes and quit.

*Go back to where your manage.py file is*

cd ..

python manage.py collectstatic

Step 7: Gunicorn

*You may remember that Gunicorn is our process manager. Let's test Gunicorn by directing it to our Django project's wsgi.py file, which is the entry point to our application.*

gunicorn --bind 0.0.0.0:8000 **{{projectName}}**.wsgi:application

*If your Gunicorn process ran correctly, you will see something like the following printed to the terminal:*

[2016-12-27 05:45:56 +0000] [8695] [INFO] Starting gunicorn 19.6.0

[2016-12-27 05:45:56 +0000] [8695] [INFO] Listening at: http://0.0.0.0:8000 (8695)

[2016-12-27 05:45:56 +0000] [8695] [INFO] Using worker: sync

[2016-12-27 05:45:56 +0000] [8700] [INFO] Booting worker with pid: 8700

*To exit the process ctrl-c*

*Now, deactivate your virtual environment.*

Set up gunicorn:

sudo vim /etc/systemd/system/gunicorn.service

*In the vim text editor copy and paste the following code. Don’t forget to type i before copying and pasting the lines below, otherwise vim may cut off a few characters at the beginning!*

[Unit]

Description=gunicorn daemon

After=network.target

[Service]

User=ubuntu

Group=www-data

WorkingDirectory=/home/ubuntu/**{{repoName}}**

ExecStart=/home/ubuntu/**{{repoName}}**/venv/bin/gunicorn --workers 3 --bind unix:/home/ubuntu/**{{repoName}}**/**{{projectName}}**.sock **{{projectName}}**.wsgi:application

[Install]

WantedBy=multi-user.target

sudo systemctl daemon-reload && sudo systemctl start gunicorn && sudo systemctl enable gunicorn

*Note:* if any additional changes are made to the gunicorn.service the previous three commands will need to be run in order to sync things up and restart our service.

Step 8: Nginx

*One final file to edit.*

sudo vim /etc/nginx/sites-available/**{{projectName}}**

*Add the following to your new document, editing what’s inside curly braces {{}}:*

server {

listen 80;

server\_name **{{yourEC2.public.ip}}**;

location = /favicon.ico { access\_log off; log\_not\_found off; }

location /static/ {

root /home/ubuntu/**{{myRepoName}}**;

}

location / {

include proxy\_params;

proxy\_pass http://unix:/home/ubuntu/**{{myRepoName}}**/**{{projectName}}**.sock;

}

}

*Save and exit.*

sudo ln -s /etc/nginx/sites-available/**{{projectName}}** /etc/nginx/sites-enabled && sudo nginx -t

Step 9 - Finally:

*We will remove the Nginx default site display from directory sites-enabled, by running the following in your terminal. Now, all that is left to do is restart your Nginx server.*

sudo rm /etc/nginx/sites-enabled/default && sudo service nginx restart

VIM

*VIM is a terminal based file editor. Press i to enter INSERT mode. You should see****–INSERT–****at the bottom left corner of your terminal. Now use your arrow keys to move the cursor to where you want to edit and make your changes.*

*Once you are done, press the esc key to exit INSERT mode. Type a colon to enter the vim command interface. You should now see a colon at the bottom left corner of your terminal. Now, type wq and press return to write (save) and quit.*

*If you want to quit without saving, type q! after the colon.*

*If you'd like to save without quitting, type w after the colon.*