

# Django Complete Project Setup with AWs, Nginx and git

- Narendra Allam, Copyright 2018.

## os requirements:

- Ubuntu Desktop Linux 16.0.4/18.04

## Steps :

- Open terminal
- Update 'sudo ' for latest system packages

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

## make directory:

- create a directory for storing all the project files in that directory

```
$ mkdir dev
```

```
$ cd dev
```

## installing pip:

- install 'pip3' for installing python packages

```
$ sudo apt-get install python3-pip
```

## virtual environment:

- it is a tool to create isolated python environments. It creates a folder which contains all the necessary packages that a python project would need.

```
$ sudo apt-get install virtualenv
```

- creating a folder with virtual environment

```
$ virtualenv venv
```

- here "venv" is the virtual environment folder name , the folder name is of as your wish

- activating the virtual environment

`$ source venv/bin/activate`

### Mysql database setup:

`$ sudo apt-get install python3-dev mysql-server libmysqlclient-dev`

`$ pip install mysql-python`

`$ pip install mysql-connector==2.1.6`

### login to mysql:

- While installing mysql, copy and save the default password provided by the installer.

Installer prompts and show

- you the default password. save it and alter root password as below

`$ mysql -u root -p[root password]`

- Change the password

`$ mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass';`

- Create a user

`$ mysql> CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'password';`

- Give privileges

`$ mysql > GRANT ALL PRIVILEGES ON *.* TO 'newuser'@'localhost';`

- Create database

`$ mysql > create database databasename;`

### Install django & restframework:

- Install Django and Django REST framework into the virtualenv

```
$ pip install django
```

```
$ pip install djangorestframework
```

### **Installing atom:**

```
$ sudo apt-get install snapd
```

```
$ snap install atom --classic
```

### **Setting up a new project :**

```
$ django-admin startproject projectname
```

### **Open the project with text editor:**

```
$ atom projectname
```

- now in atom you can see the project and app folders with files in it.

### **Move to projects directory:**

```
$ cd projectname
```

### **create an app:**

```
$ python manage.py startapp appname
```

### **Make changes in settings.py file:**

- in atom go to settings.py file and change the database credentials and also under the installed apps give your app name and rest\_framework

```

26  DEBUG = True
27
28  ALLOWED_HOSTS = []
29
30
31  # Application definition
32
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'squares',
41      'rest_framework',
42  ]

```

```

# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'count',
        'USER': 'ram',
        'PASSWORD': 'Raghuram@9',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}

```

### Create super user:

- Super user is created so that the admin the authentication to login to the application page
  - \$ python manage.py createsuperuser
- While creating superuser it ask you to give username, mail, password

### Wrting models:

- It contains the essential fields and behaviors of the data you're storing. Generally, each model maps to a single database table. Each model is a Python class that subclasses `django.db.models.Model`

```

from __future__ import unicode_literals

from django.db import models

class Square(models.Model):
    number = models.IntegerField()
    sqr = models.IntegerField()

```

- After writing models next you have to see the changes and want to migrate them  
\$ python manage.py makemigrations
- Now you can see the changes then migrate them  
\$ python manage.py migrate
- After migrating the models check the database whether the tables are created or not

### Admin page:

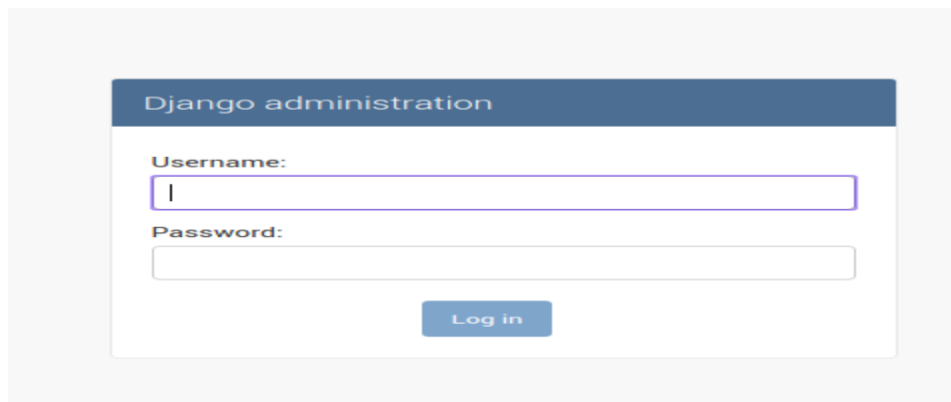
- This administrate Django groups and users, and all registered models in your app. This interface gives you the ability to Create, Read, Update, Delete operations on your registered models.

```

1  # -*- coding: utf-8 -*-
2  from __future__ import unicode_literals
3
4  from django.contrib import admin
5  from .models import Square
6  class SquareDetail(admin.ModelAdmin):
7      list_display=('number','sqr')
8  admin.site.register(Square,SquareDetail)
9
10 # Register your models here.
11

```

- **Now run the server.**  
`$ python manage.py runserver`
- **You will see an link in the terminal open it in browser.**
- **“it’s working” is displayed on the browser because your application is created successfully.**
- **Now add ‘/admin’ at the end of link then it will redirect to the admin page.**



- **Login to the django administration page.**

## Django administration

Site administration

The screenshot displays the Django administration interface. At the top, there's a header 'Django administration'. Below it, the 'Site administration' section is visible. The main content area is divided into two sections: 'AUTHENTICATION AND AUTHORIZATION' and 'SQUARES'. The 'AUTHENTICATION AND AUTHORIZATION' section contains two rows: 'Groups' and 'Users', each with '+ Add' and 'Change' links. The 'SQUARES' section contains one row: 'Squares', also with '+ Add' and 'Change' links. On the right side, there's a sidebar titled 'Recent actions' which lists 'My actions'. Under 'My actions', there are three entries: 'Square object' (with a red 'X' icon), 'Square object' (with a red 'X' icon), and 'squares object' (with a green '+' icon).

### Writing views:

- Your entire logic should be here.
- A view function, is simply a Python function that takes a Web request and returns a Web response. Return response can be the HTML contents of a Web page, or a 404 error, or an XML document, or an image . . . or anything.
- If you are using `django-rest-framework` the response will be in json format.
- The below one is a simple `api_view` get function.

```

from __future__ import unicode_literals
from django.contrib.messages.views import SuccessMessageMixin
from django.shortcuts import render
from rest_framework.decorators import api_view
from rest_framework.response import Response
from django.http import JsonResponse
from .models import Square
from .serializers import SquareSerializer
from rest_framework import status

@api_view(['GET'])
def getsquare(request, _number):

    cal = Square.objects.filter(number=_number)
    print (list(cal))
    if cal:
        serializer = SquareSerializer(cal, many=True)
        return Response(serializer.data)

```

### Serializers:

- Django's serialization framework provides a mechanism for translating Django models into other formats.

```

from rest_framework import serializers
from .models import Square

class SquareSerializer(serializers.ModelSerializer):

    class Meta:
        model = Square
        fields = '__all__'

```

### Urls:

- Create urls.py file in the application folder.
- Map the application url with project url.



### Project urls.py

```
from django.conf.urls import url, include
from django.contrib import admin
from django.http import HttpResponse
from squares import urls

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^squares/', include('squares.urls')),
]
```

### Application urls.py

```
from django.conf.urls import url, include
from django.http import HttpResponse
from .views import getsquare, postnum, cal_list

urlpatterns = [
    url(r'^calc/(?P<_number>\d+)', getsquare),
    url(r'^post/', postnum),
    url(r'^list/(?P<_number>\d+)', cal_list),
]
```

- Now run the server and check it in the browser.

### Requirements file:

- This file is to save the packages that you have used for this project.  
\$ pip freeze
- Now you can see all the packages that you have used for this project like this.

```
(venv) ubuntu@ip-172-31-45-254:~/Square$ pip freeze
Django==2.0.6
djangorestframework==3.8.2
mysqlclient==1.3.13
Pillow==5.2.0
pkg-resources==0.0.0
pytz==2018.5
```

- You can save it for further use by the following command.

```
$ pip freeze >requirements.txt
```



## Pushing code to github:

- if you don't have a github account , Create an account in github. Create a repository for storing and managing the folders.
- Create a repository to manage a project or folders at any time.

---


Owner


Repository name

  octocat /

Great repository names are short and memorable. Need inspiration? How about **petulant-computing-machine**.


Description (optional)

☒  **Public**  
Anyone can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

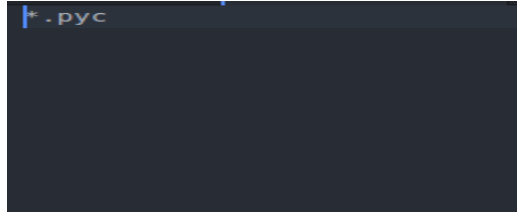
☐ **Initialize this repository with a README**  
This will allow you to `git clone` the repository immediately. Skip this step if you are going to import your existing repository or if you have already run `git init` locally.

Add .gitignore: **None** ▾

Add a license: **None** ▾ 

---

- Create a .gitignore file in your project folder to ignore the specific files and mention which files you want to ignore. In below screenshot we can see the example “\*.pyc” it ignores all .pyc files in your project while commit to github.

A screenshot of a code editor showing the content of a .gitignore file. The text "\*.pyc" is visible at the top, indicating that all Python compiled files are to be ignored by Git.

**Then follow the below commands to push the projects to repository:**

- `sudo apt-get install git`
- `git init.`
- `git add “your project name/”.`
- `git status.`
- `git commit -m "message".`
- `git remote add origin “repository root”.`
- `git push -u origin master.`

### **Aws Instance:**

- An EC2 instance is a virtual server in Amazon's Elastic Compute Cloud (EC2) for running applications on the AWS.

### **How to launch a AWS Instance:**

- Create an account in aws.
- After going into console dashboard go to EC2
- Click on launch Instance to launch the instance.

- EC2 Dashboard
- Events
- Tags
- Reports
- Limits
- INSTANCES
  - Instances
  - Launch Templates
  - Spot Requests
  - Reserved Instances
  - Dedicated Hosts
- IMAGES
  - AMIs
  - Bundle Tasks
- ELASTIC BLOCK STORE
  - Volumes
  - Snapshots
- NETWORK & SECURITY
  - Security Groups
  - Elastic IPs
  - Placement Groups

## Resources

You are using the following Amazon EC2 resources in the US East (Ohio) region:

2 Running Instances	2 Elastic IPs
0 Dedicated Hosts	0 Snapshots
2 Volumes	0 Load Balancers
5 Key Pairs	8 Security Groups
0 Placement Groups	

Learn more about the latest in AWS Compute from AWS re:Invent 2017 by viewing the [EC2 Videos](#).

## Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

[Launch Instance](#)

Note: Your instances will launch in the US East (Ohio) region

## Service Health

### Service Status:

✓ US East (Ohio):  
This service is operating normally

### Availability Zone Status:

## Scheduled Events

### US East (Ohio):

No events

➤ **Select an ubuntu server 16.04 AMI in below screenshot.**

1. Choose AMI 2. Choose instance type 3. Configure instance 4. Add storage 5. Add tags 6. Configure Security Group 7. Review

## Step 1: Choose an Amazon Machine Image (AMI)

[Cancel and Exit](#)



**Red Hat Enterprise Linux 7.5 (HVM), SSD Volume Type** - ami-03291866

Red Hat

Free tier eligible

Red Hat Enterprise Linux version 7.5 (HVM), EBS General Purpose (SSD) Volume Type  
Root device type: ebs Virtualization type: hvm

Select

64-bit



**SUSE Linux Enterprise Server 12 SP3 (HVM), SSD Volume Type** - ami-f4e6da91

SUSE Linux

Free tier eligible

SUSE Linux Enterprise Server 12 Service Pack 3 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.

Root device type: ebs Virtualization type: hvm

Select

64-bit



**Ubuntu Server 16.04 LTS (HVM), SSD Volume Type** - ami-6a003c0f

Ubuntu

Free tier eligible

Ubuntu Server 16.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Root device type: ebs Virtualization type: hvm

Select

64-bit

- **Select an instance type t2.micro and Click on review and launch .**

## Step 2: Choose an Instance Type

Filter by: All instance types Current generation [Show/Hide Columns](#)

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)


	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Configure Instance Details](#)

- **Click on launch.**

## AMI Details

[Edit AMI](#)

 **Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-6a003c0f**

Free tier eligible

Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).  
Root Device Type: ebs    Virtualization type: hvm

## Instance Type

[Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

## Security Groups

[Edit security groups](#)

**Security group name** launch-wizard-8  
**Description** launch-wizard-8 created 2018-07-02T17:10:51.267+05:30

Type	Protocol	Port Range	Source	Description
------	----------	------------	--------	-------------

[Cancel](#) [Previous](#) [Launch](#)

- Select create a new key pair and give a name in key value pair , and click on download key pair and click on launch to launch the instance.

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.


Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair

Key pair name

aws

Download Key Pair



You have to download the **private key file** (\*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel

Launch Instances

- select a instance and click on Elastic ip in network&security to set the static ip.

EC2 Dashboard

Events

Tags

Reports

Limits

INSTANCES

**Instances**

Launch Templates

Spot Requests

Reserved Instances

Dedicated Hosts

IMAGES

AMIs

Bundle Tasks

ELASTIC BLOCK STORE

Volumes

Snapshots

NETWORK & SECURITY

Security Groups

Elastic IPs

Launch Instance Connect Actions

Filter by tags and attributes or search by keyword

<input type="checkbox"/>	name	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
<input type="checkbox"/>	report2		i-060d5ae8b41525547	t2.micro	us-east-2b	running	2/2 checks passed
<input checked="" type="checkbox"/>			i-03d9447d9afd7b60f	t2.micro	us-east-2c	running	2/2 checks passed

Instance: **i-03d9447d9afd7b60f** Elastic IP: **13.59.189.14**

Description Status Checks Monitoring Tags

Instance ID	i-03d9447d9afd7b60f	Public DNS (IPv4)	ec2-13-59-189-14.us-east-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	13.59.189.14
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs	13.59.189.14*	Private DNS	ip-172-31-33-135.us-east-2.compute.internal
Availability zone	us-east-2c	Private IPs	172.31.33.135
Security groups	launch-wizard-7, view inbound rules, view outbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-c9e2bfa1

- click on allocate new address.

Allocate new address Actions

Filter by tags and attributes or search by keyword

<input type="checkbox"/>	name	Name	Elastic IP	Allocation ID	Instance	Private IP
<input type="checkbox"/>			18.188.49.245	eipalloc-09892bddc...	i-060d5ae8b41525547	172.31.33.135
<input type="checkbox"/>			13.59.189.14	eipalloc-0aa0f46291...	i-03d9447d9afd7b60f	172.31.33.135

- Click on allocate.

esses > Allocate new address

## Allocate new address

Allocate a new Elastic IP address by selecting the scope in which it will be used

WS Command Line Interface command

Cancel Allocate

- Now you can have the elastic ip address.
- Then you have to sync it to the instance.

Addresses > Allocate new address

## Allocate new address

✓ New address request succeeded

Elastic IP 52.14.170.123

Close

- Select the ip address , click actions and select associate address.

Allocate new address Actions ^

Filter by tags and attributes

<input type="checkbox"/>	name	Name	Allocation ID	Instance
<input checked="" type="checkbox"/>		52.14.170.123	eipalloc-0f8f0d8610...	-
<input type="checkbox"/>		18.188.49.245	eipalloc-09892bddc...	i-060d5ae8b41525547
<input type="checkbox"/>		13.59.189.14	eipalloc-0aa0f46291...	i-03d9447d9afd7b60f

Release addresses  
Associate address  
Disassociate address  
Add/Edit Tags



- **Select your running instance id and select private ip ,Click on associate.**

Addresses > Associate address

### Associate address

Select the instance OR network interface to which you want to associate this Elastic IP address (52.14.170.123)

Resource type ☒ Instance ?  
☐ Network interface

Instance  ↻

Private IP  ↻ ?

Reassociation ☐ Allow Elastic IP to be reassociated if already attached ?

**Warning**

If you associate an Elastic IP address with your instance, your current public IP address is released. [Learn more](#).

▶ AWS Command Line Interface command

Cancel Associate

- **Check whether the Elastic ip address is associated with instance or not.**

Allocate new address

Actions

Filter by tags and attributes or search by keyword

1 to 2 of 2

<input type="checkbox"/>	name	Name	Elastic IP	Allocation ID	Instance	Private IP address	Scope	Associa
<input checked="" type="checkbox"/>			18.188.49.245	eipalloc-09892bddc...	i-060d5ae8b41525547	172.31.21.136	vpc	eipassoc
<input type="checkbox"/>			13.59.189.14	eipalloc-0aa0f46291...	i-03d9447d9afd7b60f	172.31.33.135	vpc	eipassoc

k

- Click on instances.

The screenshot shows the AWS Management Console interface for the EC2 service. On the left is a navigation sidebar with the following menu items: EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES (which is expanded to show Instances, Launch Templates, Spot Requests, Reserved Instances, and Dedicated Hosts), IMAGES (expanded to show AMIs), and AMIs. The main content area has a top bar with a blue 'Allocate new address' button and a grey 'Actions' dropdown menu. Below this is a search bar with a magnifying glass icon and the text 'Filter by tags and attributes or search by keyword'. A table displays the list of instances with columns: a checkbox, 'name', 'Name', and 'Elastic IP'. Two instances are listed: the first has an Elastic IP of 18.188.49.245, and the second has an Elastic IP of 13.59.189.14.

<input type="checkbox"/>	name	Name	Elastic IP
<input type="checkbox"/>			18.188.49.245
<input type="checkbox"/>			13.59.189.14

- Select the instance and click on security groups.

EC2 Dashboard

Events

Tags

Reports

Limits

INSTANCES

**Instances**

Launch Templates

Spot Requests

Reserved Instances

Dedicated Hosts

IMAGES

AMIs

Bundle Tasks

ELASTIC BLOCK STORE

Volumes

Snapshots

NETWORK & SECURITY

Security Groups

Elastic IPs

Launch Instance

Connect

Actions

Filter by tags and attributes or search by keyword

	name	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
<input type="checkbox"/>	report2		i-060d5ae8b41525547	t2.micro	us-east-2b	running	2/2 checks passed
<input checked="" type="checkbox"/>			i-03d9447d9afd7b60f	t2.micro	us-east-2c	running	2/2 checks passed

Instance: **i-03d9447d9afd7b60f** Elastic IP: **13.59.189.14**

Description

Status Checks

Monitoring

Tags

Instance ID	i-03d9447d9afd7b60f	Public DNS (IPv4)	ec2-13-59-189-14.us-east-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	13.59.189.14
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs	13.59.189.14*	Private DNS	ip-172-31-33-135.us-east-2.compute.internal
Availability zone	us-east-2c	Private IPs	172.31.33.135
Security groups	launch-wizard-7. view inbound rules. view outbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-c9e2bfa1

➤ Select your instance group, Click on actions to edit inbound rules.

Create Security Group

Actions

Filter by tags and attributes

Delete Security Group

Add/Edit Tags

Copy to new

Edit inbound rules

Edit outbound rules

	Name	Group Name	VPC ID	Description
<input type="checkbox"/>	sg-03a4f9b98527715e0	launch-wizard-4	vpc-c9e2bfa1	launch-wizard-4
<input checked="" type="checkbox"/>	sg-03a4f9b98527715e0	launch-wizard-7	vpc-c9e2bfa1	launch-wizard-7
<input type="checkbox"/>	sg-04c49cd998e6b3846	launch-wizard-6	vpc-c9e2bfa1	launch-wizard-6
<input type="checkbox"/>	sg-0de6b9a8171f324c2	launch-wizard-5	vpc-c9e2bfa1	launch-wizard-5
<input type="checkbox"/>	sg-c826aca2	default	vpc-c9e2bfa1	default

- By default it has 22 port and remaining port ranges and types you have to set like below mentioned in the screen shot and click on save.

Edit inbound rules

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom ::/0	e.g. SSH for Admin Desktop
Custom TCP	TCP	8000	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP	TCP	8000	Custom ::/0	e.g. SSH for Admin Desktop
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

Add Rule

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Cancel
Save

- Click on instances.

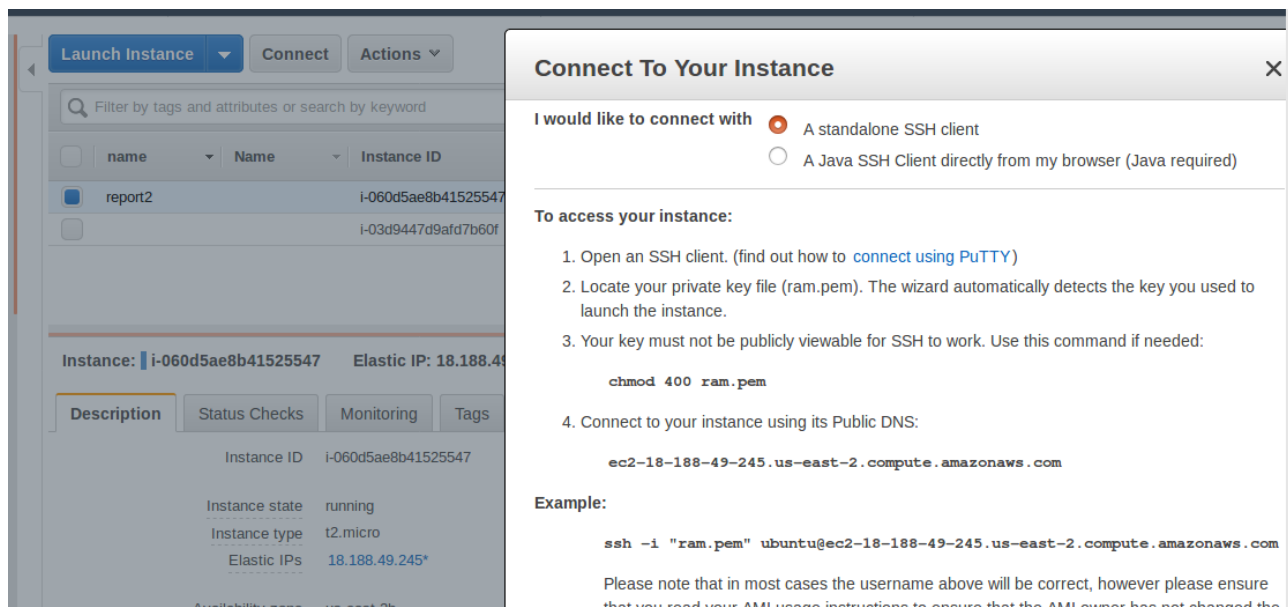
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Launch Templates
Spot Requests
Reserved Instances
Dedicated Hosts

Create Security Group
Actions

Filter by tags and attributes or search by keyword

	Name	Group ID	Group Name
<input type="checkbox"/>		sg-0188c147723c7506c	launch-wizard-4
<input checked="" type="checkbox"/>		sg-03a4f9b98527715e0	launch-wizard-7
<input type="checkbox"/>		sg-04c49cd998e6b3846	launch-wizard-6
<input type="checkbox"/>		sg-0de6b9a8171f324c2	launch-wizard-5
<input type="checkbox"/>		sg-c826aca2	default

- Select the instance and click on connect.



- You can get the dialogue box like this.
- You can connect in your instance in your systems terminal.

### Steps to connect in terminal:

- Open terminal and change the directory to where you downloaded the .pem file .

`$cd downloads.`

- Copy and paste the below “ `chmod 400 ram.pem` ”in terminal.
- Next copy the line just below the example its like `ssh`

```

hp@hp-HP-ProBook-4445s:~$ cd Downloads
hp@hp-HP-ProBook-4445s:~/Downloads$ chmod 400 ram.pem
hp@hp-HP-ProBook-4445s:~/Downloads$ ssh -i "ram.pem" ubuntu@ec2-18-188-49-245.us
-east-2.compute.amazonaws.com

Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-1060-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

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

*** System restart required ***
Last login: Fri Jun 29 11:01:53 2018 from 14.142.116.211

```

- Now you are inside the local Ubuntu server
- Here once again you have to update sudo
- Here again you have install pip3,mysql,virtualenv,python3-dev
- Activate your virtualenv
- You have to clone the project folder from github now

\$ git clone [your repository root]

- If you have multiple folders in your repository, you have clone only the folder that you wants

### Cloning a particular folder from git:

\$ git clone --depth 1 [repository root] [name of the folder]

\$ cd [folder name]

\$ git filter-branch --prune-empty --subdirectory-filter [folder name] HEAD

- NOTE: No need of square brackets.

- Here again you have to do Mysql database setup and login to mysql shell.
- Create a database and come out of mysql shell.
- Install requirements.txt here.

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

### **Edit files:**

- Here you cannot install atom so you can edit the files in terminal.  
\$ vi [filename]
- After opening the file with vi editor press i to make changes after making changes press 'esc' and ":wq" to save the file.
- Here you have to make changes in settings.py file. In allowed hosts you have to give your ip address and 0.0.0.0 or '\*' in single or double quotes. '\*' means any one.
- if your database credentials are not same then change them in allowed hosts.

- Now you have to migrate.

```
$ python manage.py makemigrations.
```

```
$ python manage.py migrate.
```

- Now your database tables are created.
- Create superuser again and insert the data.
- Now you can run the server by command.

```
$ python manage.py runserver 0.0.0.0:8000
```

- Now you can check the output in the browser with ip address. And also you can see it in your mobiles also.

## **Nginx setup:**

### **What is Nginx?**

- **Nginx is one of the most popular webserver for its load balancing, reverse proxy and security. It hosts some of the largest traffic sites on the internet.**

## **Installation Steps:**

- **we have to update local package index , so that we can install recent packages.**

**\$ sudo apt-get update**

- **install nginx.**

**\$ sudo apt-get install nginx**

- **Adjusting the firewall: To configure our firewall to allow access to the service.**

**\$ sudo ufw app list**

- **You should get a listing of the application profiles:**

**Output:  
Available applications:  
Nginx Full  
Nginx HTTP  
Nginx HTTPS  
OpenSSH**

- **As you can see, there are three profiles available for Nginx:**
- **Nginx Full: This profile opens both port 80 (normal, unencrypted web traffic) and port 443 (TLS/SSL encrypted traffic)**
- **Nginx HTTP: This profile opens only port 80 (normal, unencrypted web traffic)**
- **Nginx HTTPS: This profile opens only port 443 (TLS/SSL encrypted traffic)**



- You can enable this by typing:  
\$ sudo ufw allow 'Nginx HTTP'
- you can verify the change by seeing the status.  
\$ sudo ufw status
- you can see the HTTP traffic allowed.

Output:

Status: active

To	Action	From
--	-----	----
OpenSSH	ALLOW	Anywhere
Nginx HTTP	ALLOW	Anywhere
OpenSSH (v6)	ALLOW	Anywhere (v6)
Nginx HTTP (v6)	ALLOW	Anywhere (v6)

- In case status is inactive follow these steps:  
\$ sudo ufw enable  
\$ sudo ufw status
- Check the Nginx service is active or not.  
\$ systemctl status nginx

```
(venv) ubuntu@ip-172-31-33-135:~$ deactivate
ubuntu@ip-172-31-33-135:~$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: en
   Active: active (running) since Fri 2018-06-29 11:57:54 UTC; 3 days ago
     Main PID: 13564 (nginx)
        Tasks: 2
       Memory: 2.5M
          CPU: 182ms
      CGroup: /system.slice/nginx.service
              └─13564 nginx: master process /usr/sbin/nginx -g daemon on; master_pr
                 └─13565 nginx: worker process
```

- When you have your server's IP address or domain, enter it into your browser's address bar:

`http://server_domain_or_IP`

- You should see the default Nginx landing page, which should look something like this:

# Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](http://nginx.org). Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

- This page is simply included with Nginx to show you that the server is running correctly.

## Uwsgi:

- UWSGI: it is an application server which can communicate with applications through interface called wsgi.

## UWSGI Installion Steps:

- install uwsgi by using pip

`$ sudo -H pip install uwsgi`

- For instance, we can tell it to serve our first project by typing:

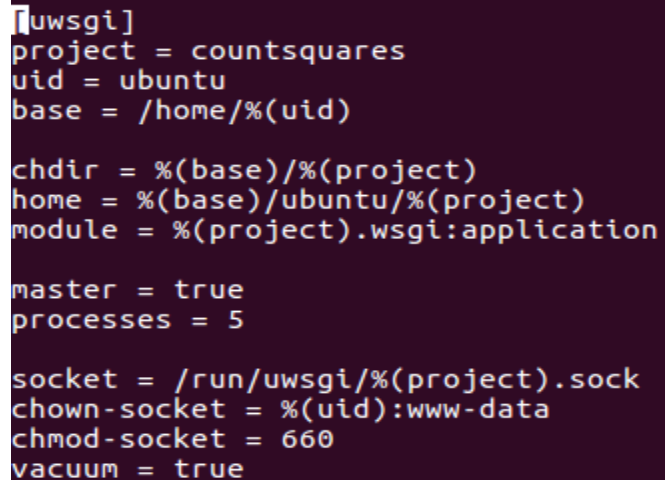
`$ uwsgi --http :8080 --home /home/ubuntu/venv --chdir /home/ubuntu/projectname -w projectname.wsgi`

- create a directory to store configure files.

**\$ sudo mkdir -p /etc/uwsgi/sites**

- **create a file for your project and open in a text editor.**

**\$ sudo nano /etc/uwsgi/sites/firstsite.ini**



```
[uwsgi]
project = countsquares
uid = ubuntu
base = /home/%(uid)

chdir = %(base)/%(project)
home = %(base)/ubuntu/%(project)
module = %(project).wsgi:application

master = true
processes = 5

socket = /run/uwsgi/%(project).sock
chown-socket = %(uid):www-data
chmod-socket = 660
vacuum = true
```

- **in above screenshot you type mention project = “your projectname” , uid = “your system id”**
- **Create a systemd file to automate the uwsgi.**

**\$ sudo nano /etc/systemd/system/uwsgi.service**

```
[Unit]
Description=uWSGI Emperor service

[Service]
ExecStartPre=/bin/bash -c 'mkdir -p /run/uwsgi; chown ubuntu:www-data /run/uwsgi'
ExecStart=/usr/local/bin/uwsgi --emperor /etc/uwsgi/sites
Restart=always
KillSignal=SIGQUIT
Type=notify
NotifyAccess=all

[Install]
WantedBy=multi-user.target
```

- **Create a server block configuration file where our project can access.**

```
server {
    listen 80;
    server_name 13.59.189.14;

    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        root /home/ubuntu/countsquares;
    }

    location / {
        include uwsgi_params;
        uwsgi_pass unix:/run/uwsgi/countsquares.sock;
    }
}
```

- **link your configuration file to Nginx sites-enabled directory to enable them.**

**\$ sudo ln -s /etc/nginx/sites-available/projectname /etc/nginx/sites-enabled**

- Check the configuration by typing.

**\$ sudo nginx -t**

```
ubuntu@ip-172-31-33-135:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
ubuntu@ip-172-31-33-135:~$
```

- if no errors are detected, you restart your nginx service to load the new configuration.

**\$ sudo systemctl restart nginx**

- start uwsgi

**\$ sudo systemctl start uwsgi**

- Let's delete the UFW rule to port 8000 and instead allow access to our Nginx server.

**\$ sudo ufw delete 'allow 8000'**

**\$ sudo ufw allow 'Nginx Full'**

- you can enable both of the services to start automatically at boot by typing.

**\$ sudo systemctl enable nginx**

**\$ sudo systemctl enable uwsgi**

- Now you can see the output in browser as nginx is running your server.

**Refer the below links if you have any doubts in nginx setup:**

- 1) <https://www.digitalocean.com/community/tutorials/how-to-serve-django-applications-with-uwsgi-and-nginx-on-ubuntu-16-04>
- 2) <https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-16-04>