



Simple Django Blog



Why Django?

Django is a Web framework written in Python. A Web framework is a software that supports the development of dynamic Web sites, applications, and services. It provides a set of tools and functionalities that solves many common problems associated with Web development, such as security features, database access, sessions, template processing, URL routing, internationalization, localization, and much more.

Using a Web framework, such as Django, enables us to develop secure and reliable Web applications very quickly in a standardized way, without having to reinvent the wheel.





Simple Django Blog



In this tutorial we are going to use AWS instance – a virtual machine, running Ubuntu

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with links like EC2 Dashboard, Events, Tags, Reports, Limits, Instances (which is selected), Launch Templates, and Spot Requests. The main area has tabs for Launch Instance, Connect, and Actions. Below that is a search bar and a table with columns: Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, Public DNS (IPv4), IPv4 Public IP, IPv6 IPs, and Key Name. Two instances are listed:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name
	i-062354ac4d2528...	t2.micro	us-east-2c	running	2/2 checks passed	None	ec2-18-218-252-191....	18.218.252.191	-	grivis-emr-20180123
	i-0ea8013e0f11b7339	t2.micro	us-east-2a	running	2/2 checks passed	None	ec2-18-219-66-163.us... ec2-18-219-66-163.us...	18.219.66.163	-	grivis-emr-20180123

This screenshot shows the detailed view for the instance i-0ea8013e0f11b7339. The top bar shows the instance ID and public DNS. Below is a tab navigation bar with Description, Status Checks, Monitoring, and Tags. The main content is divided into sections:

- Description:** Instance ID: i-0ea8013e0f11b7339, Instance state: running, Instance type: t2.micro, Availability zone: us-east-2a, Security groups: ElasticMapReduce-master, AMI ID: ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20180126 (ami-965e6bf3), Platform: -, IAM role: -, Key pair name: grivis-emr-20180123.
- EBS-optimized:** False
- Root device type:** ebs
- Block devices:** /dev/sda1
- Elastic GPU:** -
- Elastic GPU type:** -
- Elastic GPU status:** -
- Networking:** Public DNS (IPv4): ec2-18-219-66-163.us-east-2.compute.amazonaws.com, IPv4 Public IP: 18.219.66.163, Private DNS: ip-172-31-1-192.us-east-2.compute.internal, Private IPs: 172.31.1.192, Secondary private IPs: VPC ID: vpc-2a889043, Subnet ID: subnet-9f826cf7, Network interfaces: eth0, Sourced/dest. check: True, T2 Unlimited: Disabled, Owner: 662692220389, Launch time: March 23, 2018 at 8:45:12 AM UTC+3 (less than one hour).
- Termination protection:** False
- Lifecycle:** normal
- Monitoring:** basic
- Alarm status:** None
- Kernel ID:** -
- RAM disk ID:** -
- Placement group:** -
- Virtualization:** hvm
- Reservation:** r-014e0fd327b587e82
- AMI launch index:** 0



Simple Django Blog

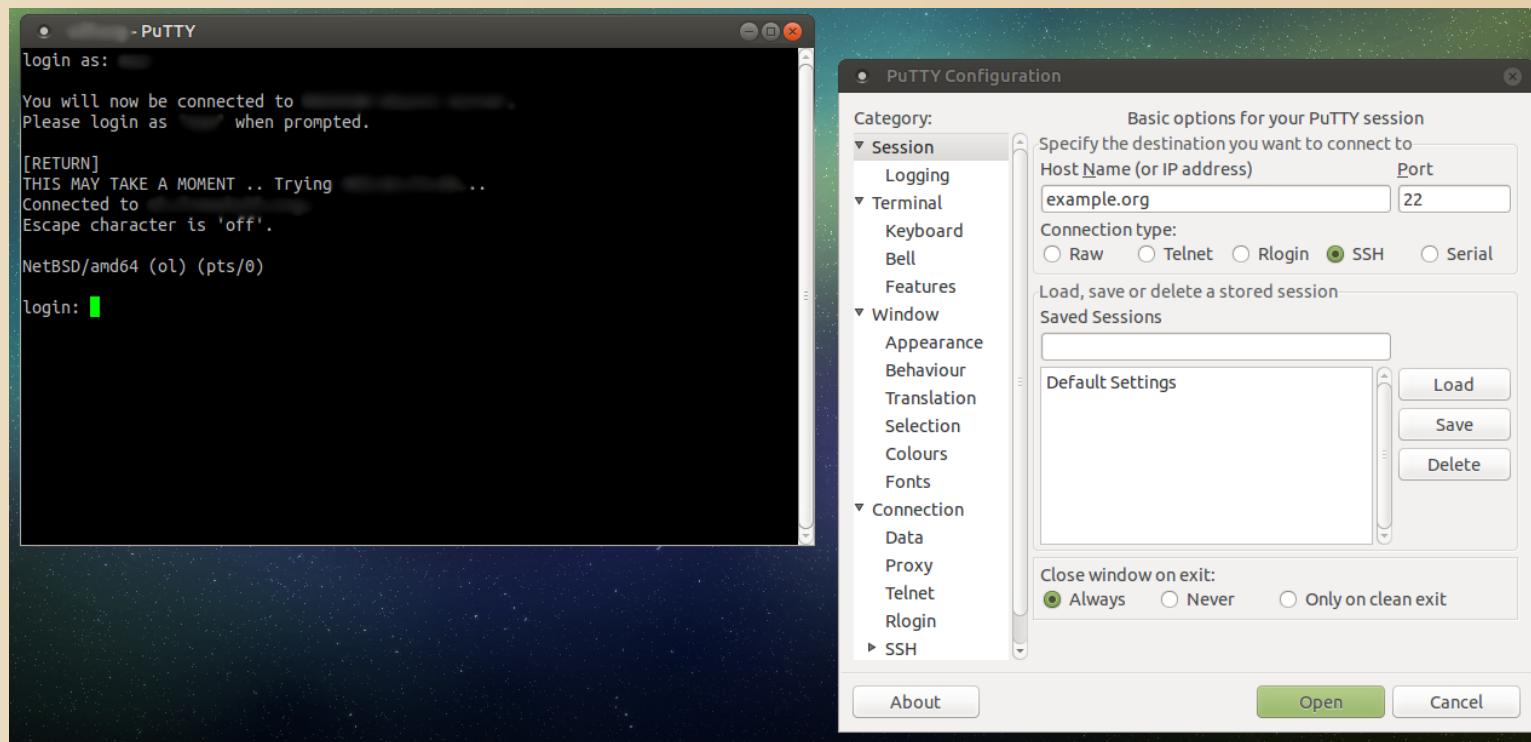


If you are not familiar with AWS, you may read here about Amazon Elastic Cloud and how to create and run instances

The screenshot shows a Mozilla Firefox browser window with the title bar "Amazon EC2 - Mozilla Firefox". The address bar displays the URL "https://aws.amazon.com/ec2/". The main content area shows the Amazon EC2 landing page. At the top of the page is a navigation bar with links for "Menu", the "aws" logo, "English", "My Account", and a yellow "Sign In to the Console" button. Below the navigation bar are tabs for "Amazon EC2", "Overview", "Features", "Pricing", "Getting Started", "Resources", and "Instance Types". The main content features a large banner with the text "Amazon EC2" and "Secure and resizable compute capacity in the cloud. Launch applications when needed without upfront commitments." A yellow "Get started with Amazon EC2" button is located at the bottom left of the banner. The background of the page is a blurred image of a futuristic data center or network infrastructure.



To connect to remote Ubuntu server from Windows environment you can use PuTTY



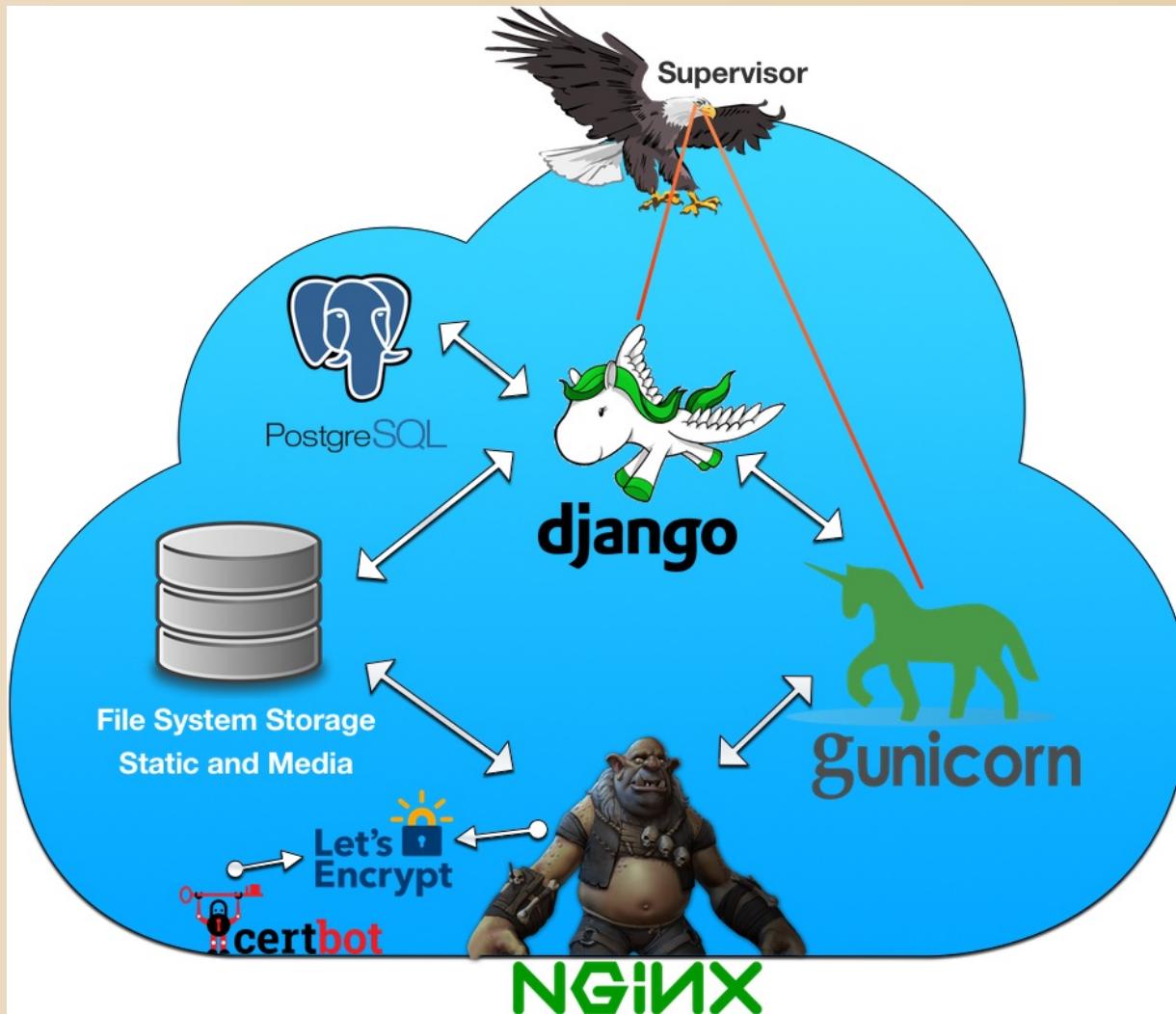
<https://www.chiark.greenend.org.uk/~sgtatham/putty/>

django

Simple Django Blog



Our final goal is to create a real web blog in production environment using the most advanced tools

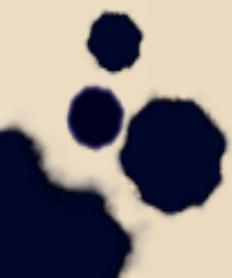




Simple Django Blog



The first thing we need to do is install some programs on our machine so to be able to start playing with Django. The basic setup consists of installing Python, Virtualenv, and Django.

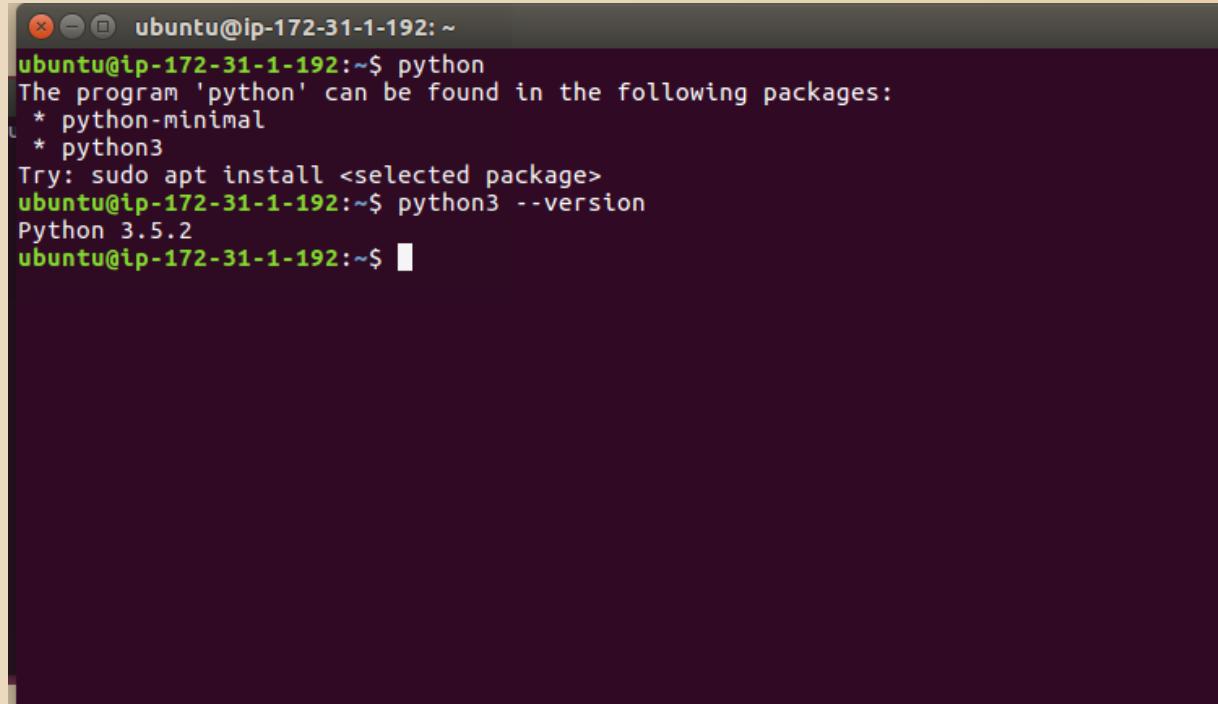




Simple Django Blog



For this tutorial, we will be using Ubuntu 16.04 as an example. Ubuntu 16.04 already comes with both Python 2 (available as `python`), and Python 3 (available as `python3`) installed. We can test the installation by opening the Terminal and checking the versions:

A screenshot of a terminal window on an Ubuntu 16.04 system. The window title bar says "ubuntu@ip-172-31-1-192:~". The terminal output shows:

```
ubuntu@ip-172-31-1-192:~$ python
The program 'python' can be found in the following packages:
 * python-minimal
 * python3
Try: sudo apt install <selected package>
ubuntu@ip-172-31-1-192:~$ python3 --version
Python 3.5.2
ubuntu@ip-172-31-1-192:~$
```

The text is in white on a dark background, with command prompts in green.



Simple Django Blog



If you are using Ubuntu 16.04 or an older version, first add the following repository:

```
sudo add-apt-repository ppa:deadsnakes/ppa
```

If you are using Ubuntu 16.10, 17.04 or 17.10 you don't need to perform the step above. Now everyone executes the following commands to install the latest Python 3 distribution:

```
sudo apt-get update  
sudo apt-get install python3.6
```

The new installation will be available under python3.6, which is fine:



```
ubuntu@ip-172-31-1-192: ~  
Get cloud support with Ubuntu Advantage Cloud Guest:  
http://www.ubuntu.com/business/services/cloud  
  
65 packages can be updated.  
21 updates are security updates.  
  
Last login: Fri Mar 23 05:47:32 2018 from 185.145.38.234  
ubuntu@ip-172-31-1-192:~$ python  
The program 'python' can be found in the following packages:  
* python-minimal  
* python3  
Try: sudo apt install <selected package>  
ubuntu@ip-172-31-1-192:~$ python3  
Python 3.5.2 (default, Nov 23 2017, 16:37:01)  
[GCC 5.4.0 20160609] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>>  
ubuntu@ip-172-31-1-192:~$ python3.6  
Python 3.6.4 (default, Jan 28 2018, 17:52:01)  
[GCC 5.4.0 20160609] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>>  
ubuntu@ip-172-31-1-192:~$
```



Simple Django Blog



We are going to use pip, a tool to manage and install Python packages, to install virtualenv. First let's install pip for our Python 3.6.2 version:

```
wget https://bootstrap.pypa.io/get-pip.py  
sudo python3.6 get-pip.py
```

Now we can install virtualenv:
`sudo pip3.6 install virtualenv`

```
ubuntu@ip-172-31-1-192:~$ sudo python3.6 get-pip.py  
The directory '/home/ubuntu/.cache/pip/http' or its parent directory is not owned by the current user and  
the cache has been disabled. Please check the permissions and owner of that directory. If executing pip wi  
th sudo, you may want sudo's -H flag.  
The directory '/home/ubuntu/.cache/pip' or its parent directory is not owned by the current user and cachi  
ng wheels has been disabled. check the permissions and owner of that directory. If executing pip with sudo  
, you may want sudo's -H flag.  
Collecting pip  
  Downloading pip-9.0.3-py2.py3-none-any.whl (1.4MB)  
    100% |██████████| 1.4MB 994kB/s  
Collecting setuptools  
  Downloading setuptools-39.0.1-py2.py3-none-any.whl (569kB)  
    100% |██████████| 573kB 2.3MB/s  
Collecting wheel  
  Downloading wheel-0.30.0-py2.py3-none-any.whl (49kB)  
    100% |██████████| 51kB 11.4MB/s  
Installing collected packages: pip, setuptools, wheel  
Successfully installed pip-9.0.3 setuptools-39.0.1 wheel-0.30.0  
ubuntu@ip-172-31-1-192:~$ sudo pip3.6 install virtualenv  
The directory '/home/ubuntu/.cache/pip/http' or its parent directory is not owned by the current user and  
the cache has been disabled. Please check the permissions and owner of that directory. If executing pip wi  
th sudo, you may want sudo's -H flag.  
The directory '/home/ubuntu/.cache/pip' or its parent directory is not owned by the current user and cachi  
ng wheels has been disabled. check the permissions and owner of that directory. If executing pip with sudo  
, you may want sudo's -H flag.  
Collecting virtualenv  
  Downloading virtualenv-15.2.0-py2.py3-none-any.whl (2.6MB)  
    100% |██████████| 2.6MB 571kB/s  
Installing collected packages: virtualenv  
Successfully installed virtualenv-15.2.0  
ubuntu@ip-172-31-1-192:~$
```



Simple Django Blog



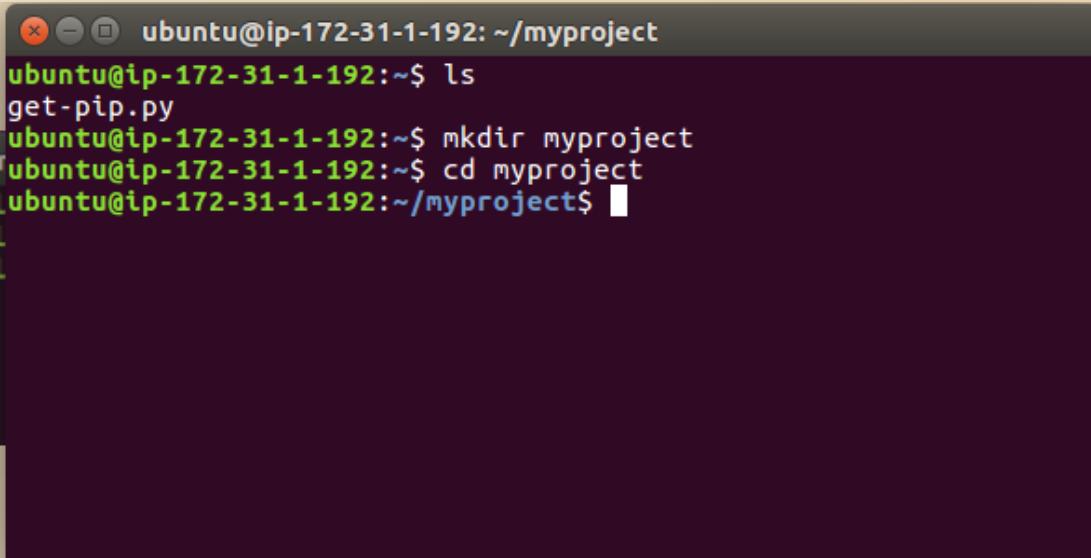
All installations that we performed were system-wide. From now on, everything we install, including Django itself, will be installed inside a Virtual Environment.

Think of it like this: for each Django project you start, you will first create a Virtual Environment for it. It's like having a sandbox for each Django project. So you can play around, install packages, uninstall packages without breaking anything.

We suggest to create a folder named Development on my personal computer. Then, we use it to organize all my projects and websites. But you can follow the next steps creating the directories wherever it feels right for you.

Usually, we start by creating a new folder with the project name inside my Development folder. Since this is going to be our very first project, we don't need to pick a fancy name or anything. For now, we can call it myproject.

```
mkdir myproject  
cd myproject
```



A screenshot of a terminal window titled "ubuntu@ip-172-31-1-192: ~/myproject". The window shows the following command history:

```
ubuntu@ip-172-31-1-192:~/myproject$ ls  
get-pip.py  
ubuntu@ip-172-31-1-192:~/myproject$ mkdir myproject  
ubuntu@ip-172-31-1-192:~/myproject$ cd myproject  
ubuntu@ip-172-31-1-192:~/myproject$
```



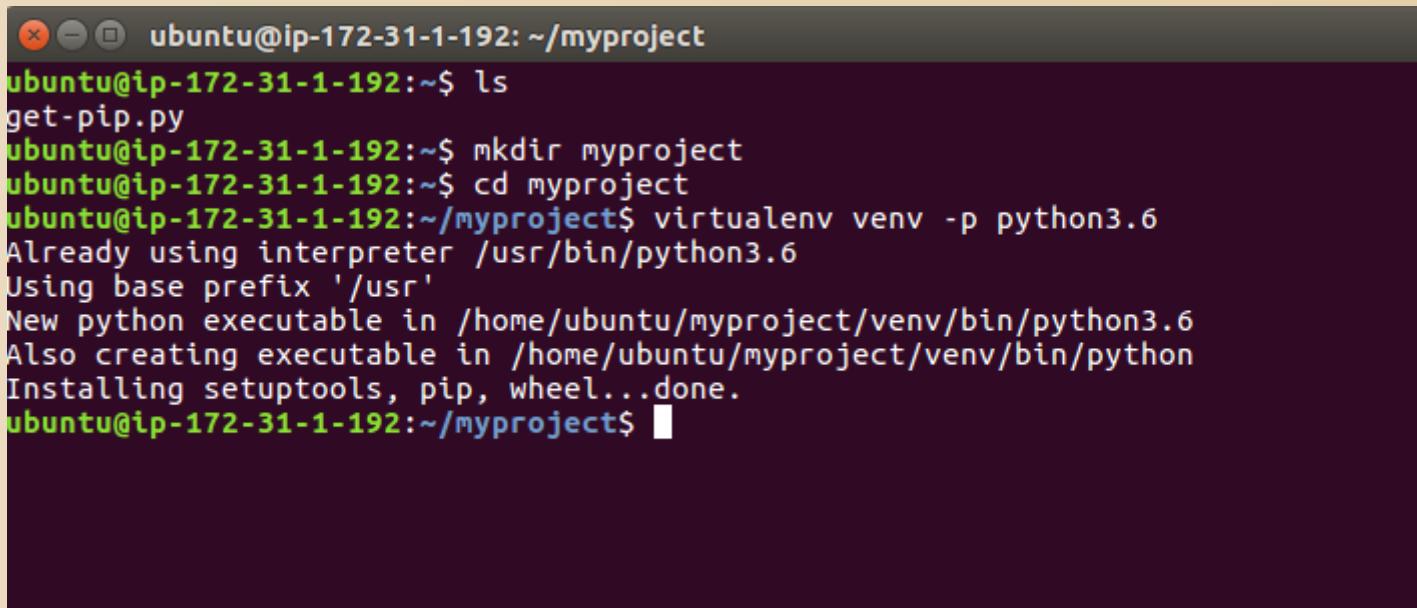
Simple Django Blog



We start by creating our very first virtual environment and installing Django.

Inside the myproject folder:

```
virtualenv venv -p python3.6
```

A screenshot of a terminal window on an Ubuntu system. The title bar says "ubuntu@ip-172-31-1-192: ~/myproject". The terminal shows the command "virtualenv venv -p python3.6" being run and its output, which includes creating a virtual environment and setting up pip and setuptools. The terminal window has a dark background with light-colored text.

```
ubuntu@ip-172-31-1-192:~/myproject
ubuntu@ip-172-31-1-192:~$ ls
get-pip.py
ubuntu@ip-172-31-1-192:~$ mkdir myproject
ubuntu@ip-172-31-1-192:~$ cd myproject
ubuntu@ip-172-31-1-192:~/myproject$ virtualenv venv -p python3.6
Already using interpreter /usr/bin/python3.6
Using base prefix '/usr'
New python executable in /home/ubuntu/myproject/venv/bin/python3.6
Also creating executable in /home/ubuntu/myproject/venv/bin/python
Installing setuptools, pip, wheel...done.
ubuntu@ip-172-31-1-192:~/myproject$ █
```



Simple Django Blog



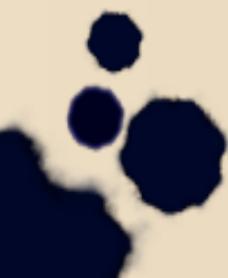
Our virtual environment is created. Now before we start using it, we need to activate:

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

You will know it worked if you see (venv) in front of the command line, like this:

A screenshot of a terminal window titled "ubuntu@ip-172-31-1-192: ~ /myproject". The terminal shows the following sequence of commands:

```
ubuntu@ip-172-31-1-192: ~ /myproject
ubuntu@ip-172-31-1-192: ~ /myproject$ ls
venv
ubuntu@ip-172-31-1-192: ~ /myproject$ source venv/bin/activate
(venv) ubuntu@ip-172-31-1-192: ~ /myproject$
```

The terminal has a dark background with light-colored text. The prompt "(venv)" indicates that the virtual environment is now active.



Simple Django Blog

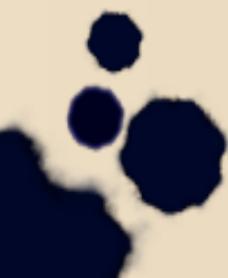


When we have the venv activated, we will use the command `python` (instead of `python3.6`) to refer to Python 3.6.2, and just `pip` (instead of `pip3.6`) to install packages.

By the way, to deactivate the venv run the command below:

deactivate

But let's keep it activated for the next steps.



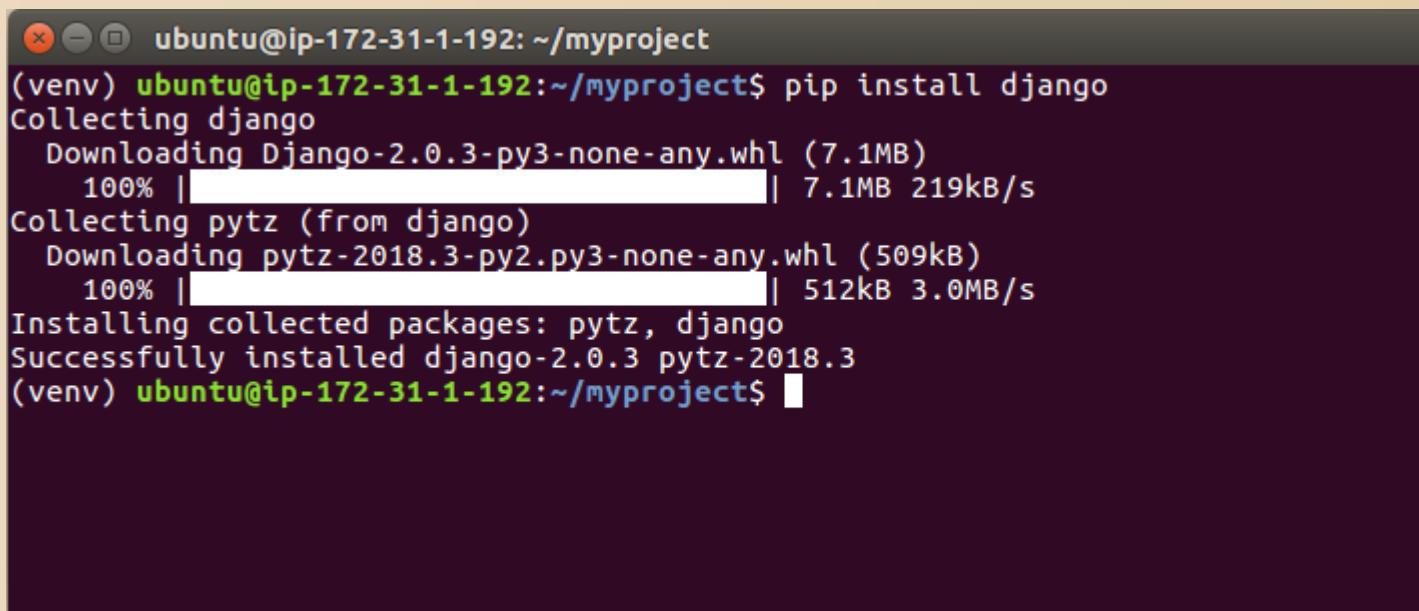


Installing Django



It's very straightforward. Now that we have the venv activated, run the following command to install Django:

```
pip install django
```



A screenshot of a terminal window on an Ubuntu system. The terminal shows the command `pip install django` being run. The process involves collecting packages (django and pytz), downloading them (Django-2.0.3 and pytz-2018.3), and then installing them. The output includes progress bars for the downloads and a confirmation message at the end.

```
ubuntu@ip-172-31-1-192: ~/myproject
(venv) ubuntu@ip-172-31-1-192:~/myproject$ pip install django
Collecting django
  Downloading Django-2.0.3-py3-none-any.whl (7.1MB)
    100% |██████████| 7.1MB 219kB/s
Collecting pytz (from django)
  Downloading pytz-2018.3-py2.py3-none-any.whl (509kB)
    100% |██████████| 512kB 3.0MB/s
Installing collected packages: pytz, django
Successfully installed django-2.0.3 pytz-2018.3
(venv) ubuntu@ip-172-31-1-192:~/myproject$
```



Installing Django



Another useful utility is tree:

```
ubuntu@ip-172-31-1-192:~/myproject

(venv) ubuntu@ip-172-31-1-192:~/myproject$ sudo apt-get install tree
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  tree
0 upgraded, 1 newly installed, 0 to remove and 62 not upgraded.
Need to get 40.6 kB of archives.
After this operation, 138 kB of additional disk space will be used.
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu xenial/universe amd64 tre
e amd64 1.7.0-3 [40.6 kB]
Fetched 40.6 kB in 0s (3516 kB/s)
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
```



Starting a New Project



To create a new Django project, run the command below:

```
django-admin startproject myproject
```

The command-line utility django-admin is automatically installed with Django.

After we run the command above, it will generate the base folder structure for a Django project.

Right now, our myproject directory looks like this:

```
myproject/
|-- myproject/           <-- higher level folder
|   |-- __init__.py      <-- django project folder
|   |-- settings.py
|   |-- urls.py
|   |-- wsgi.py
|   `-- manage.py
+-- venv/                <-- virtual environment folder
```

```
ubuntu@ip-172-31-1-192:~/myproject/myproject
(venv) ubuntu@ip-172-31-1-192:~/myproject$ ls
(venv) ubuntu@ip-172-31-1-192:~/myproject$ django-admin startproject myproject
myproject  venv
(venv) ubuntu@ip-172-31-1-192:~/myproject$ cd myproject/
(venv) ubuntu@ip-172-31-1-192:~/myproject/myproject$ tree
.
└── manage.py
    ├── myproject
    │   ├── __init__.py
    │   ├── settings.py
    │   ├── urls.py
    │   └── wsgi.py
1 directory, 5 files
(venv) ubuntu@ip-172-31-1-192:~/myproject/myproject$
```



Starting a New Project



Our initial project structure is composed of five files:

- **manage.py**: a shortcut to use the **django-admin** command-line utility. It's used to run management commands related to our project. We will use it to run the development server, run tests, create migrations and much more.
- **__init__.py**: this empty file tells Python that this folder is a Python package.
- **settings.py**: this file contains all the project's configuration. We will refer to this file all the time!
- **urls.py**: this file is responsible for mapping the routes and paths in our project. For example, if you want to show something in the URL `/about/`, you have to map it here first.
- **wsgi.py**: this file is a simple gateway interface used for deployment. You don't have to bother about it. Just let it be for now.



Starting a New Project



Django comes with a simple web server installed. It's very convenient during the development, so we don't have to install anything else to run the project locally. We can test it by executing the command:

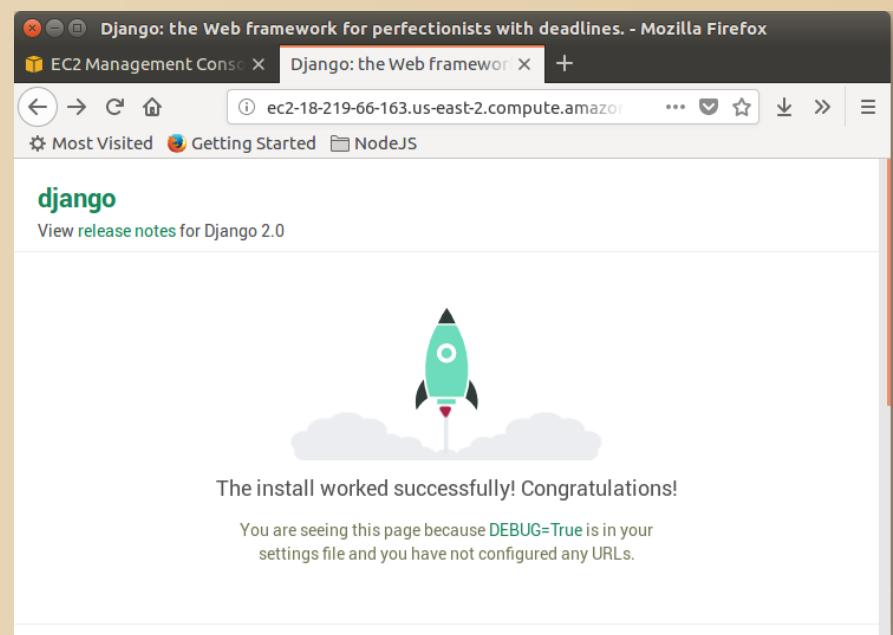
```
python manage.py runserver 0.0.0.0:8000
```

For now, you can ignore the migration errors; we will get to that later.

Now open the following URL in a Web browser:

<http://ec2-18-219-66-163.us-east-2.compute.amazonaws.com:8000/>

and you should see the following page:





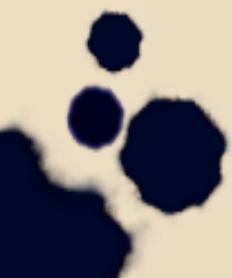
Django Apps



In the Django philosophy we have two important concepts:

- **app**: is a Web application that does something. An app usually is composed of a set of models (database tables), views, templates, tests.
- **project**: is a collection of configurations and apps. One project can be composed of multiple apps, or a single app.

It's important to note that you can't run a Django **app** without a **project**. Simple websites like a blog can be written entirely inside a single app, which could be named **blog** or **weblog** for example.





To illustrate let's create a simple Web Forum or Discussion Board. To create our first app, go to the directory where the manage.py file is and executes the following command:

```
django-admin startapp boards
```

Notice that we used the command startapp this time.

```
myproject/
|-- myproject/
|   |-- boards/           <-- our new django app!
|   |   |-- migrations/
|   |   |   `-- __init__.py
|   |   |-- __init__.py
|   |   |-- admin.py
|   |   |-- apps.py
|   |   |-- models.py
|   |   |-- tests.py
|   |   `-- views.py
|   |-- myproject/
|   |   |-- __init__.py
|   |   |-- settings.py
|   |   |-- urls.py
|   |   |-- wsgi.py
|   `-- manage.py
`-- venv/
```

```
(venv) ubuntu@ip-172-31-1-192:~/myproject/myproject$ tree
.
├── boards
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   └── migrations
│       └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
└── db.sqlite3
└── manage.py
myproject
├── __init__.py
└── __pycache__
    ├── __init__.cpython-36.pyc
    ├── settings.cpython-36.pyc
    ├── urls.cpython-36.pyc
    └── wsgi.cpython-36.pyc
├── settings.py
└── urls.py
wsgi.py

4 directories, 17 files
(venv) ubuntu@ip-172-31-1-192:~/myproject/myproject$
```



Django Apps



- **migrations/**: here Django store some files to keep track of the changes you create in the **models.py** file, so to keep the database and the **models.py** synchronized.
- **admin.py**: this is a configuration file for a built-in Django app called **Django Admin**.
- **apps.py**: this is a configuration file of the app itself.
- **models.py**: here is where we define the entities of our Web application. The models are translated automatically by Django into database tables.
- **tests.py**: this file is used to write unit tests for the app.
- **views.py**: this is the file where we handle the request/response cycle of our Web application.



Django Apps



Now open the settings.py and try to find the INSTALLED_APPS variable:

settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

```
ubuntu@ip-172-31-1-192: ~/myproject/myproject
GNU nano 2.5.3      File: myproject/settings.py

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['ec2-18-219-66-163.us-east-2.compute.amazonaws.com']

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'boards'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',

```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'boards',
]
```

We added a new application



Hello, World!



Let's write our first view. We will explore it in great detail in the next tutorial. But for now, let's just experiment how it looks like to create a new page with Django.

views.py

```
from django.http import HttpResponse

def home(request):
    return HttpResponse('Hello, World!')
```

urls.py

```
from django.conf.urls import url
from django.contrib import admin

from boards import views

urlpatterns = [
    url(r'^$', views.home, name='home'),
    url(r'^admin/', admin.site.urls),
]
```



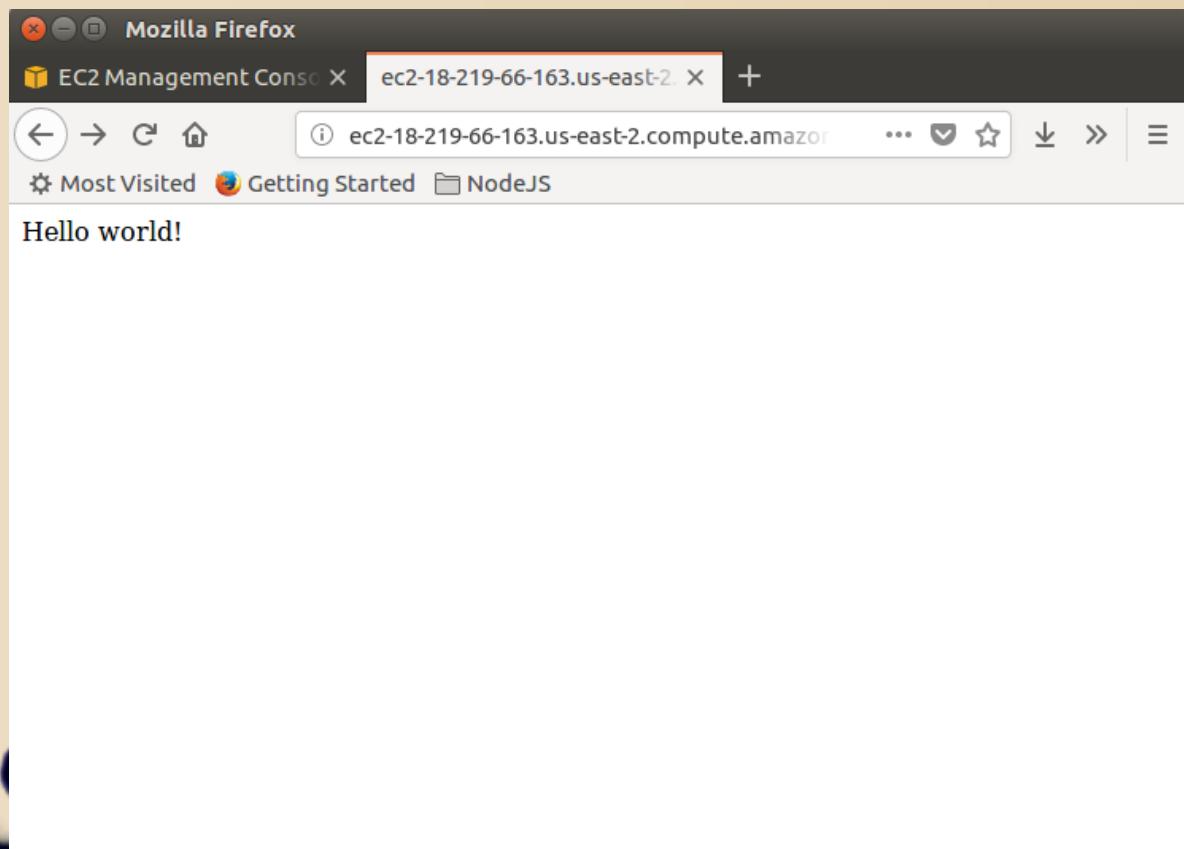
Hello, World!



No we can run our server again:

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

In a Web browser, open the <http://ec2-18-219-66-163.us-east-2.compute.amazonaws.com:8000/>:



django

Django Blog



The End of Part One

Au revoir! Adios!
HEY HEY! Arrivederci!
Good bye! Anito!
Ciao! salut! Tschüss!
adeus!

