**How To Install the Django Web Framework on Ubuntu 14.04**

Mar 5, 2015 [Django](https://www.digitalocean.com/community/tags/django?type=tutorials), [Python Frameworks](https://www.digitalocean.com/community/tags/python-frameworks?type=tutorials), [Python](https://www.digitalocean.com/community/tags/python?type=tutorials) [Ubuntu](https://www.digitalocean.com/community/tags/ubuntu?type=tutorials)

**Introduction**

Django is a full-featured Python web framework for developing dynamic websites and applications. Using Django, you can quickly create Python web applications and rely on the framework to do a good deal of the heavy lifting.

In this guide, we will show you how to get Django up and running on an Ubuntu 14.04 server. After installation, we'll show you how to start a new project to use as the basis for your site.

**Different Methods**

There are a number of different ways in which you can install Django depending upon your needs and how you want to configure your development environment. These have different advantages and one method may lend itself better to your specific situation than others.

Some of the different methods are below:

* **Global Install from Packages**: The official Ubuntu repositories contain Django packages that can be installed easily with the conventional apt package manager. This is very simple, but not as flexible as some other methods. Also, the version contained in the repositories may lag behind the official versions available from the project.
* **Global Install through pip**: The pip tool is a package manager for Python packages. If you install pip, you can easily install Django on the system level for use by any user. This should always contain the latest stable release. Even so, global installations are inherently less flexible.
* **Install through pip in a Virtualenv**: The Python virtualenv package allows you to create self-contained environments for various projects. Using this technology, you can install Django in a project directory without affecting the greater system. This allows you to provide per-project customizations and packages easily. Virtual environments add some slight mental and process overhead in comparison to globally accessible installation, but provide the most flexibility.
* **Development Version Install through git**: If you wish to install the latest development version instead of the stable release, you will have to acquire the code from the git repo. This is necessary to get the latest features/fixes and can be done globally or locally. Development versions do not have the same stability guarantees, however.

With the above caveats and qualities in mind, select the installation method that best suites your needs out of the below instructions.

**Global Install from Packages**

If you wish to install Django using the Ubuntu repositories, the process is very straight forward.

First, update your local package index with apt, and then install the python-django package:

sudo apt-get update

sudo apt-get install python-django

You can test that the installation was successful by typing:

django-admin --version

1.6.1

This means that the software was successfully installed. You may also notice that the Django version is not the latest stable. To learn a bit about how to use the software, skip ahead to learn [how to create sample project](https://www.digitalocean.com/community/tutorials/how-to-install-the-django-web-framework-on-ubuntu-14-04#creating-a-sample-project).

**Global Install through pip**

If you wish to install the latest version of Django globally, a better option is to use pip, the Python package manager. First, we need to install the pip package manager. Refresh your apt package index:

sudo apt-get update

Now you can install pip. If you plan on using Python version 2, install using the following commands:

sudo apt-get install python-pip

If, instead, you plan on using Python 3, use this command:

sudo apt-get install python3-pip

Now that you have pip, we can easily install Django. If you are using Python 2, you can type:

sudo pip install django

If you are using Python 3, use the pip3 command instead:

sudo pip3 install django

You can verify that the installation was successful by typing:

django-admin --version

1.7.5

As you can see, the version available through pip is more up-to-date than the one from the Ubuntu repositories (yours will likely be different from the above).

**Install through pip in a Virtualenv**

Perhaps the most flexible way to install Django on your system is with the virtualenv tool. This tool allows you to create virtual Python environments where you can install any Python packages you want without affecting the rest of the system. This allows you to select Python packages on a per-project basis regardless of conflicts with other project's requirements.

We will begin by installing pip from the Ubuntu repositories. Refresh your local package index before starting:

sudo apt-get update

If you plan on using version 2 of Python, you can install pip by typing:

sudo apt-get install python-pip

If, instead, you plan on using version 3 of Python, you can install pip by typing:

sudo apt-get install python3-pip

Once pip is installed, you can use it to install the virtualenv package. If you installed the Python 2 pip, you can type:

sudo pip install virtualenv

If you installed the Python 3 version of pip, you should type this instead:

sudo pip3 install virtualenv

Now, whenever you start a new project, you can create a virtual environment for it. Start by creating and moving into a new project directory:

mkdir ~/newproject

cd ~/newproject

Now, create a virtual environment within the project directory by typing:

virtualenv newenv

This will install a standalone version of Python, as well as pip, into an isolated directory structure within your project directory. We chose to call our virtual environment newenv, but you should name it something descriptive. A directory will be created with the name you select, which will hold the file hierarchy where your packages will be installed.

To install packages into the isolated environment, you must activate it by typing:

source newenv/bin/activate

Your prompt should change to reflect that you are now in your virtual environment. It will look something like (newenv)username@hostname:~/newproject$.

In your new environment, you can use pip to install Django. Regardless of whether you are using version 2 or 3 of Python, it should be called just pip when you are in your virtual environment. Also note that you *do not* need to use sudo since you are installing locally:

pip install django

You can verify the installation by typing:

django-admin --version

1.7.5

To leave your virtual environment, you need to issue the deactivate command from anywhere on the system:

deactivate

Your prompt should revert to the conventional display. When you wish to work on your project again, you should re-activate your virtual environment by moving back into your project directory and activating:

cd ~/newproject

source newenv/bin/activate

**Development Version Install through git**

If you need a development version of Django, you will have to download and install Django from its git repository.

To do so, you will need to install git on your system with apt. Refresh your local package index by typing:

sudo apt-get update

Now, we can install git. We will also install the pip Python package manager. We will use this to handle the installation of Django after it has been downloaded. If you are using Python 2, you can type:

sudo apt-get install git python-pip

If you are using Python 3 instead, you should type this:

sudo apt-get install git python3-pip

Once you have git, you can clone the Django repository. Between releases, this repository will have more up-to-date features and bug fixes at the possible expense of stability. You can clone the repository to a directory called django-dev within your home directory by typing:

git clone git://github.com/django/django ~/django-dev

Once the repository is cloned, you can install it using pip. We will use the -e option to install in "editable" mode, which is needed when installing from version control. If you are using version 2 of Python, type:

sudo pip install -e ~/django-dev

If you are using Python 3, type:

sudo pip3 install -e ~/django-dev

You can verify that the installation was successful by typing:

django-admin --version

1.9.dev20150305171756

Note that you can also combine this strategy with the use of virtualenv above if you wish to install a development version of Django in a single environment.

**Creating a Sample Project**

Now that you have Django installed, we can show you briefly how to get started on a project.

You can use the django-admin command to create a project:

django-admin startproject projectname

cd projectname

This will create a directory called projectname within your current directory. Within this, a management script will be created and another directory called projectname will be created with the actual code.

**Note**: If you were already in a project directory that you created for use with the virtualenv command, you can tell Django to place the management script and inner directory into the current directory without the extra layer by typing this (notice the ending dot):

django-admin startproject projectname .

To bootstrap the database (this uses SQLite by default) on more recent versions of Django, you can type:

python manage.py migrate

If the migrate command doesn't work, you likely are using an older version of Django. Instead, you can type:

python manage.py syncdb

You will be asked to create an administrative user as part of this process. Select a username, email address, and password for the user.

If you used the migrate command above, you'll need to create the administrative user manually. You can create an administrative user by typing:

python manage.py createsuperuser

You will be prompted for a username, an email address, and a password for the user.

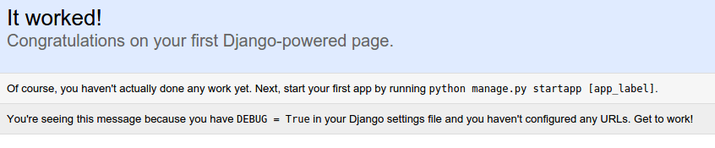
Once you have a user, you can start up the Django development server to see what a fresh Django project looks like. You should only use this for development purposes. Run:

python manage.py runserver 0.0.0.0:8000

Visit your server's IP address followed by :8000 in your web browser

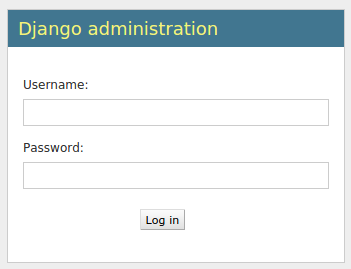
server\_ip\_address:8000

You should see something that looks like this:

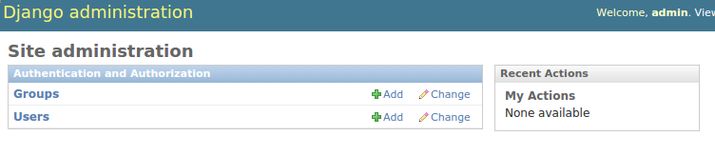


Now, append /admin to the end of your URL to get to the admin login page:

server\_ip\_address:8000/admin



If you enter the admin username and password that you just created, you should be taken to the admin section of the site:



When you are finished looking through the default site, you can stop the development server by typing CTRL-C in your terminal.

The Django project you've created provides the structural basis for designing a more complete site. Check out the Django documentation for more information about how to build your applications and customize your site.

**Conclusion**

You should now have Django installed on your Ubuntu 14.04 server, providing the main tools you need to create powerful web applications. You should also know how to start a new project and launch the developer server. Leveraging a complete web framework like Django can help make development faster, allowing you to concentrate only on the unique aspects of your applications