NOVEMBER 30, 2018 • READ IN 5 MINUTES
LAST UPDATED MARCH 25, 2019

How to Install Django on Debian 9 Linux

Django is the most popular Python web framework designed to help developers build secure, scalable and maintainable web applications.

Django can be installed system-wide or in a Python virtual environment using pip. Django packages are included in the official Debian repositories and can be installed using the `apt` package manager. This is the easiest method to install Django on Debian 9, but not as flexible as installing in a virtual environment. Also, the version included in the repositories is always several releases behind the latest version of Django.

The main goal of Python virtual environments is to create an isolated environment for different Python projects. This way you can have multiple different Django environments on a single computer and install a specific version of a module on a per project basis without worrying that it will affect your other Django installations. If you install Django into the global environment then you can install only one Django version on your computer.

Installing Django on Debian 9

Perform the following steps to install Django in a Python virtual environment on Debian 9.

1. Installing Python 3 and venv

Debian 9 ships with Python 3.5 by default. You can verify that Python 3 is installed on your system by typing:

```
$ python3 -V
```

The output should look like this:

```
Output
Python 3.5.3
```

The recommended way to create a virtual environment is by using the `venv` module.

The `venv` module is included in the `python3-venv` package. Install it by typing the following command:

Once the module is installed we are ready to create a virtual environment for our Django application.

2. Creating Virtual Environment

Start by navigating to the directory where you would like to store your Python 3 virtual environments. It can be your home directory or any other directory where your user has read and write permissions.

Create a new directory for your Django application and navigate into it:

```
$ mkdir my_django_app
$ cd my_django_app
```

From inside the directory, execute the following command to create a new virtual environment:

The command above creates a directory called `venv`, which contains a copy of the Python binary, the Pip package manager, the standard Python library and other supporting files. You can use any name you want for the virtual environment.

To start using the virtual environment, activate it by running the `activate` script:

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

Once activated, the virtual environment's bin directory will be added at the beginning of the `$PATH` variable. Also your shell's prompt will change and it will show the name of the virtual environment you're currently using. In our case that is `venv`.

3. Installing Django

Now that the virtual environment is active, install Django using the Python package manager `pip`:

```
(venv) $ pip install django
```

> Within the virtual environment, you can use the command `pip` instead of `pip3` and `python` instead of `python3`.

Verify the installation using the following command which will print the Django version:

```
(venv) $ python -m django --version
```

At the time of writing this article, the latest Django version is `2.1.2`.

```
Output
2.1.3
```

4. Creating a Django Project

Use the `django-admin` command-line utility to create a new Django project named `mydjangoapp` :

```
(venv) $ django-admin startproject mydjangoapp
```

The command above will create a `mydjangoapp` directory in your current directory.

```
(venv) $ tree  mydjangoapp/
```

Output

```
mydjangoapp/
|-- manage.py
`-- mydjangoapp
    |-- __init__.py
    |-- settings.py
    |-- urls.py
    `-- wsgi.py
```

Inside that directory, you will find the main script for managing projects named `manage.py` and another directory including database configuration, and Django and application-specific settings.

Let’s migrate the database and create an administrative user.

First, navigate to the `mydjangoapp` directory:

```
(venv) $ cd mydjangoapp
```

SQLite is the default database for Django. For production applications, you can use PostgreSQL, MariaDB, Oracle or MySQL Database.

Migrate the database by typing:

```
(venv) $ python manage.py migrate
```

The output will look something like the following:

```
Output
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
```

```
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying sessions.0001_initial... OK
```

Once the database is migrated, create an administrative user will be used to access the Django admin interface:

```
(venv) $ python manage.py createsuperuser
```

The command will prompt you for a username, an email address, and a password.

```
Output

Username (leave blank to use 'linuxize'): admin
Email address: admin@linuxize.com
Password:
Password (again):
Superuser created successfully.
```

Start the development web server using the `manage.py` script followed by the `runserver` option:

```
(venv) $ python manage.py runserver
```

You'll see the following output:

```
Output
Performing system checks...

System check identified no issues (0 silenced).
October 20, 2018 - 11:16:28
Django version 2.1.2, using settings 'mydjangoapp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

> If you installed Django on a virtual machine and you want to access Django development server then you'll need to edit the `settings.py` file and add the server IP address inside the `ALLOWED_HOSTS` list.

Open `http://127.0.0.1:8000` in your web browser and you will be presented with the default Django landing page:

To access the Django admin interface, add `/admin` to the end of the URL (`http://127.0.0.1:8000/admin/`). This will take you to the admin login screen:

Enter your username and password and you will be redirected to the Django admin page:

To stop the development server type `CTRL-C` in your terminal.

Once done with your work, deactivate the environment, by typing `deactivate` and you will return to your normal shell.

```
(venv) $ deactivate
```

Conclusion

You have learned how to create a Python virtual environment and install Django on your Debian 9 system. To create additional Django development environments repeat the steps outlined in this tutorial.

If you are new to Django, visit the Django documentation page and learn how to develop your first Django app.

If you are facing any problem, feel free to leave a comment.

python debian

JUN 21, 2018

How to install Pip on Debian 9

APR 28, 2019

How to Install TensorFlow on Debian 9

MAR 27, 2019

How to Install Python 3.7 on Debian 9

Write a comment