

# Работаем в виртуальном окружении: что такое virtualenv

Опубликовано: [04.12.2014](#) | Автор: [Николай](#)

На ресурсах, посвященных *Python* или *Django* можно часто встретить упоминания неких виртуальных окружений и программы *virtualenv*. Информация на [официальном сайте](#) дает понять, что это кое-что полезное. И не зря его всячески рекомендуют к использованию.

Совсем недавно я переустановил систему и еще не успел поставить *Django* и пр. инструменты. Поэтому, самое время организовать всю дальнейшую работу на основе виртуальных окружений. В статье пойдет речь о том, что такое *virtualenv*, зачем он нужен, как его установить и использовать. **Черновая редакция.**

## Как установить *virtualenv* или «курица vs яйцо»

На официальном сайте *virtualenv* в разделе «*Installation*» рекомендуется устанавливать *virtualenv* через менеджер *Python*-пакетов *pip* (командой «*pip install virtualenv*»). Однако далеко не всегда *pip* установлен в системе по-умолчанию. Я не стал исключением: команду *pip* система не понимает. Идем на [официальный сайт](#) и видим, что *pip* рекомендуется использовать в пределах виртуального окружения *virtualenv*. При установке *virtualenv*, *pip* устанавливается автоматически. Выходит, официальные сайты обеих программ рекомендуют устанавливать *virtualenv* через *pip*, а *pip* — через *virtualenv*. Хм...

**Примечание:** есть, конечно, множество других способов установки того и другого (через *easy\_install*, скачивание *deb*-пакетов или *python*-установщиков) — все эти способы также описаны на официальных сайтах или на чьих-то блогах. Но все-таки что-то тянет меня придерживаться рекомендуемых способов от официальных разработчиков.

Если следовать концепции виртуальных окружений — логично использовать *pip* в пределах *virtualenv*, а не глобально во всей системе. Тем более ~~нахалству~~, что и поставится он автоматически вместе с *virtualenv*. Значит, прежде всего нужно устанавливать *virtualenv*. Как?

На мой взгляд, лучшее *Linux-way* решение - установка из репозитория (почему-то этот вариант не упоминается на официальном сайте *virtualenv*).

1. Для установки *virtualenv* набираем в терминале:

```
sudo apt-get install python-virtualenv
```

2. Создаем папку, внутри которой будут храниться папки будущих виртуальных окружений. Лучше всего создать такую папку в пределах своей домашней директории, чтобы не было проблем с правами доступа. Поскольку вручную копаться в ней вряд ли придется, сделаем ее скрытой и назовем «*.virtualenvs*»:

```
mkdir .virtualenvs
```

Внутри этой папки можно создавать виртуальные окружения под каждый из наших будущих проектов.

3. Создаем виртуальное окружение внутри папки *.virtualenvs*. Пусть наше первое виртуальное окружение будет называться «*project\_one*».

```
cd .virtualenvs
```

```
virtualenv project_one
```

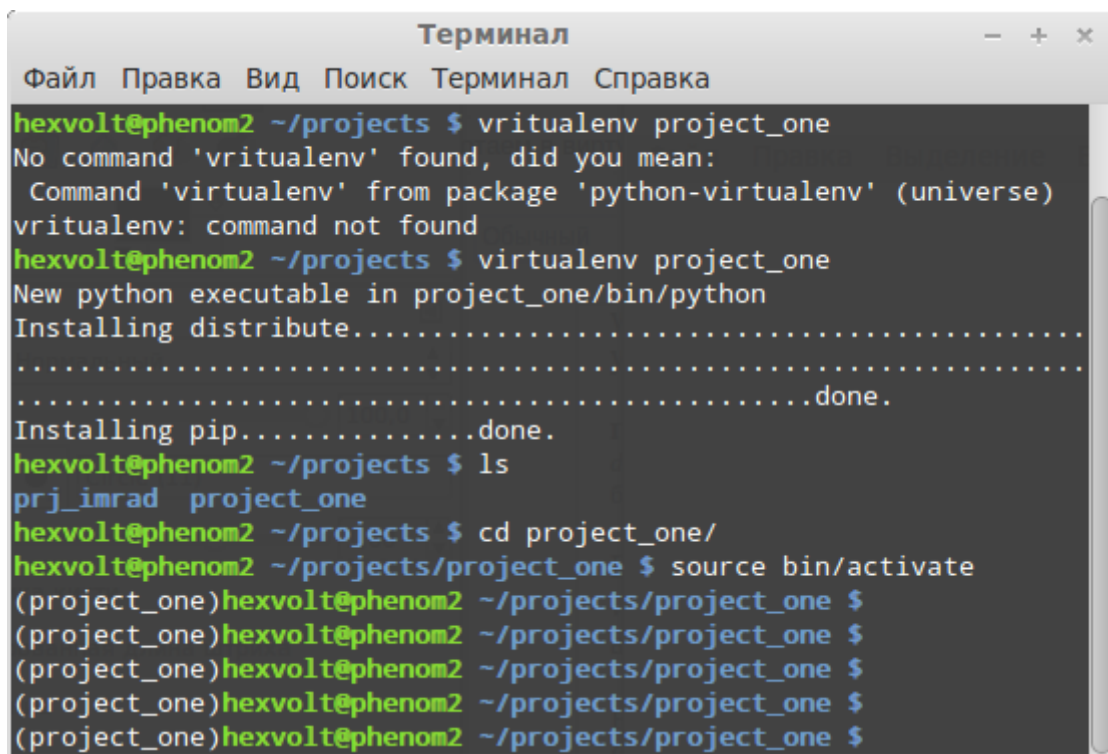
(Аналогично могут создаваться виртуальные окружения для каких-то других проектов).

В результате внутри папки *./virtualenvs/project\_one/* создается маленькая рабочая среда с папками *bin/*, *include/*, *lib/*, *local/*, содержащими минимальный «набор джентльмена» для работы — *python*, менеджеры пакетов *pip* и *easy\_install*. Сюда же могут доставляться все необходимые пакеты, фреймворки (в том числе *Django*) и утилиты. В пределах каждого виртуального окружения они будут изолированы друг от друга, не оказывая никакого взаимного «паразитного» влияния.

**Примечание:** во многих руководствах по работе с виртуальными окружениями рекомендуется выполнять команду `virtualenv` с ключом `--no-site-packages`. Применение этого ключа позволяет создавать виртуальное окружение, изолированное от системной питоновской папки `site-packages`, что повышает степень автономности. Так вот в новых версиях `virtualenv` указывать этот ключ не обязательно, поскольку в них эта опция включена по умолчанию.

4. Для активации необходимого виртуального окружения нужно зайти в его папку («`cd project_one`») и выполнить следующее:  
`source bin/activate`

После активации командная строка изменится: перед именем пользователя появится название виртуального окружения в скобках «`(project_one)имя_пользователя>@имя_компьютера ~`».



```
Терминал
Файл  Правка  Вид  Поиск  Терминал  Справка
hexvolt@phenom2 ~/projects $ virtualenv project_one
No command 'virtualenv' found, did you mean:
  Command 'virtualenv' from package 'python-virtualenv' (universe)
virtualenv: command not found
hexvolt@phenom2 ~/projects $ virtualenv project_one
New python executable in project_one/bin/python
Installing distribute.....
.....done.
Installing pip.....done.
hexvolt@phenom2 ~/projects $ ls
prj_imrad  project_one
hexvolt@phenom2 ~/projects $ cd project_one/
hexvolt@phenom2 ~/projects/project_one $ source bin/activate
(project_one)hexvolt@phenom2 ~/projects/project_one $
(project_one)hexvolt@phenom2 ~/projects/project_one $
(project_one)hexvolt@phenom2 ~/projects/project_one $
(project_one)hexvolt@phenom2 ~/projects/project_one $
(project_one)hexvolt@phenom2 ~/projects/project_one $
```

Теперь любые команды по установке пакетов (например, «`pip install django`») или по их удалению будут выполняться только в пределах активированного окружения.

Для выхода из виртуального окружения и перехода в обычный режим достаточно набрать:  
`deactivate`

**5.** Для более удобной работы с *virtualenv* рекомендуется поставить утилиту под названием ***virtualenvwrapper***. Это специальная надстройка над *virtualenv*, которая избавит от необходимости заходить в определенную папку для включения виртуального окружения и обеспечит прочие «ништяки». Поставив ее, можно будет переключаться между виртуальными окружениями ~~одним взмахом руки~~ одной командой, вызванной откуда угодно. Итак, устанавливаем:

```
sudo apt-get install virtualenvwrapper
```

Следуя, рекомендациям разработчиков, для настройки *virtualenvwrapper* на нашу папку с виртуальными окружениями, добавляем следующий текст в файл `~/.profile` (чтобы эти настройки выполнялись при каждой загрузке компа):

```
export WORKON_HOME=$HOME/.virtualenvs
source /etc/bash_completion.d/virtualenvwrapper
```

Чтобы наши изменения настроек вступили в силу без перезагрузки, запускаем на выполнение файл `.profile`:

```
source ~/.profile
```

**6.** Теперь работа с виртуальными окружениями стала еще проще:

<code>mkvirtualenv env_name1</code>	— создать в папке с вирт.окружениями окружение « <i>env_name1</i> »
<code>workon env_name1</code>	- переключится в окружение « <i>env_name1</i> »
<code>lsvirtualenv</code>	— вывести список доступных виртуальных окружений
<code>rmvirtualenv env_name1</code>	- удалить виртуальное окружение « <i>env_name1</i> »
<code>deactivate</code>	- выйти из текущего виртуального окружения