

The vote is over, but the fight for net neutrality isn't. Show your support for a free and open internet.

Learn more

 **pypa / pipenv**

Python Development Workflow for Humans. <https://docs.pipenv.org/>  
[#pip](#) [#python](#) [#packaging](#) [#virtualenv](#) [#pipfile](#) [#kennethreitz](#)

🔗 2,688 commits

🌿 6 branches

📦 212 releases

👥 120 contributors

📄 MIT

Branch: master ▾


New pull request

Create new file























Upload files


Find file

Clone or download ▾


 nateprewitt Merge pull request #1250 from joshfriend/python-3.6.4 ...


Latest commit e45b597 6 days ago


 .github	Fix typo in GitHub issue template	2 months ago
 _appveyor	setup appveyor for pipenv	8 months ago
 docs	Update install.rst	13 days ago
 examples	update examples	3 months ago
 pipenv	Update default python for 3.6.x	6 days ago
 tests	Add docker pin to test to fix resolver error	11 days ago
 .gitignore	Display warning message if PYENV_ROOT is not set and abort if install...	3 months ago
 .travis.yml	cache issues with pypy	3 months ago
 CODE_OF_CONDUCT.md	adding Code of Conduct	3 months ago
 CONTRIBUTING.rst	Add some basic development docs	3 months ago
 Dockerfile	Update default python for 3.6.x	6 days ago
 HISTORY.txt	v9.0.1	17 days ago
 LICENSE	Wrap lines in license text	11 months ago
 MANIFEST.in	Add cacert.pem from requests to releases. Fixes #871	3 months ago
 Makefile	make man	3 months ago
 NOTICES	Fix typos:	a month ago
 Pipfile	hope this does not break build	24 days ago
 README.rst	Update README.rst	13 days ago
 appveyor.yml	more coverage	3 months ago
 get-pipenv.py	vendor all the things!	10 months ago
 setup.cfg	added flake8 back in and added jobs for style and unit tests	4 months ago
 setup.py	moving on up	26 days ago


 **README.rst**


# Pipenv: Python Development Workflow for Humans


 **v9.0.1**


 **MIT**

 **2.7, 3.3, 3.4, 3.5, 3.6**

 **passing**

 **passing**

 **Say Thanks**



**Pipenv** — the officially recommended Python packaging tool from [Python.org](https://python.org), free (as in freedom).

Pipenv is a tool that aims to bring the best of all packaging worlds (bundler, composer, npm, cargo, yarn, etc.) to the Python world. *Windows is a first-class citizen, in our world.*

It automatically creates and manages a virtualenv for your projects, as well as adds/removes packages from your `Pipfile` as you install/uninstall packages. It also generates the ever-important `Pipfile.lock`, which is used to produce deterministic builds.

<http://media.kennethreitz.com.s3.amazonaws.com/pipenv.gif>

The problems that Pipenv seeks to solve are multi-faceted:

- You no longer need to use `pip` and `virtualenv` separately. They work together.
- Managing a `requirements.txt` file [can be problematic](#), so Pipenv uses the upcoming `Pipfile` and `Pipfile.lock` instead, which is superior for basic use cases.
- Hashes are used everywhere, always. Security. Automatically expose security vulnerabilities.
- Give you insight into your dependency graph (e.g. `$ pipenv graph`).
- Streamline development workflow by loading `.env` files.

## Installation

---

```
$ pip install pipenv
```



## 🍷 User Testimonials

---

**Jannis Leidel, former `pip` maintainer—**

*Pipenv is the porcelain I always wanted to build for `pip`. It fits my brain and mostly replaces `virtualenvwrapper` and manual `pip` calls for me. Use it.*

**Justin Myles Holmes—**

*Pipenv is finally an abstraction meant to engage the mind instead of merely the filesystem.*

**Isaac Sanders—**

*Pipenv is literally the best thing about my day today. Thanks, Kenneth!*

## 🍷 Features

---

- Enables truly *deterministic builds*, while easily specifying *only what you want*.
- Generates and checks file hashes for locked dependencies.
- Automatically install required Pythons, if `pyenv` is available.
- Automatically finds your project home, recursively, by looking for a `Pipfile`.
- Automatically generates a `Pipfile`, if one doesn't exist.
- Automatically creates a virtualenv in a standard location.
- Automatically adds/removes packages to a `Pipfile` when they are un/installed.
- Automatically loads `.env` files, if they exist.

The main commands are `install`, `uninstall`, and `lock`, which generates a `Pipfile.lock`. These are intended to replace `$ pip install` usage, as well as manual virtualenv management (to activate a virtualenv, run `$ pipenv shell`).

## Basic Concepts

- A virtualenv will automatically be created, when one doesn't exist.
- When no parameters are passed to `install`, all packages [packages] specified will be installed.
- To initialize a Python 3 virtual environment, run `$ pipenv --three`.
- To initialize a Python 2 virtual environment, run `$ pipenv --two`.
- Otherwise, whatever virtualenv defaults to will be the default.

## Other Commands

- `shell` will spawn a shell with the virtualenv activated.
- `run` will run a given command from the virtualenv, with any arguments forwarded (e.g. `$ pipenv run python`).
- `check` asserts that PEP 508 requirements are being met by the current environment.
- `graph` will print a pretty graph of all your installed dependencies.

## Shell Completion

For example, with fish, put this in your `~/.config/fish/completions/pipenv.fish`:

```
eval (pipenv --completion)
```

Alternatively, with bash, put this in your `.bashrc` or `.bash_profile`:

```
eval "$(pipenv --completion)"
```

Magic shell completions are now enabled! There is also a [fish plugin](#), which will automatically activate your subshells for you!

Fish is the best shell. You should use it.

## Usage

```
$ pipenv
Usage: pipenv [OPTIONS] COMMAND [ARGS]...
```

Options:

<code>--update</code>	Update Pipenv & pip to latest.
<code>--where</code>	Output project home information.
<code>--venv</code>	Output virtualenv information.
<code>--py</code>	Output Python interpreter information.
<code>--envs</code>	Output Environment Variable options.
<code>--rm</code>	Remove the virtualenv.
<code>--bare</code>	Minimal output.
<code>--completion</code>	Output completion (to be eval'd).
<code>--man</code>	Display manpage.
<code>--three / --two</code>	Use Python 3/2 when creating virtualenv.
<code>--python TEXT</code>	Specify which version of Python virtualenv should use.
<code>--site-packages</code>	Enable site-packages for the virtualenv.
<code>--jumbotron</code>	An easter egg, effectively.
<code>--version</code>	Show the version and exit.
<code>-h, --help</code>	Show this message and exit.

Usage Examples:

```
Create a new project using Python 3.6, specifically:
$ pipenv --python 3.6
```

```
Install all dependencies for a project (including dev):
$ pipenv install --dev
```

```
Create a lockfile containing pre-releases:
$ pipenv lock --pre

Show a graph of your installed dependencies:
$ pipenv graph

Check your installed dependencies for security vulnerabilities:
$ pipenv check

Install a local setup.py into your virtual environment/Pipfile:
$ pipenv install -e .
```

```
Commands:
check      Checks for security vulnerabilities and...
graph      Displays currently-installed dependency graph...
install    Installs provided packages and adds them to...
lock       Generates Pipfile.lock.
open       View a given module in your editor.
run        Spawns a command installed into the...
shell      Spawns a shell within the virtualenv.
uninstall  Un-installs a provided package and removes it...
update     Uninstalls all packages, and re-installs...
```

Locate the project:

```
$ pipenv --where
/Users/kennethreitz/Library/Mobile Documents/com~apple~CloudDocs/repos/kr/pipenv/test
```

Locate the virtualenv:

```
$ pipenv --venv
/Users/kennethreitz/.local/share/virtualenvs/test-Skyy4vre
```

Locate the Python interpreter:

```
$ pipenv --py
/Users/kennethreitz/.local/share/virtualenvs/test-Skyy4vre/bin/python
```

Install packages:

```
$ pipenv install
Creating a virtualenv for this project...
...
No package provided, installing all dependencies.
Virtualenv location: /Users/kennethreitz/.local/share/virtualenvs/test-EJkjoYts
Installing dependencies from Pipfile.lock...
...

To activate this project's virtualenv, run the following:
$ pipenv shell
```

Install a dev dependency:

```
$ pipenv install pytest --dev
Installing pytest...
...
Adding pytest to Pipfile's [dev-packages]...
```

Show a dependency graph:

```
$ pipenv graph
requests==2.18.4
- certifi [required: >=2017.4.17, installed: 2017.7.27.1]
- chardet [required: >=3.0.2,<3.1.0, installed: 3.0.4]
- idna [required: >=2.5,<2.7, installed: 2.6]
- urllib3 [required: <1.23,>=1.21.1, installed: 1.22]
```

Generate a lockfile:

```
$ pipenv lock
Assuring all dependencies from Pipfile are installed...
Locking [dev-packages] dependencies...
Locking [packages] dependencies...
Note: your project now has only default [packages] installed.
To install [dev-packages], run: $ pipenv install --dev
```

Install all dev dependencies:

```
$ pipenv install --dev
Pipfile found at /Users/kennethreitz/repos/kr/pip2/test/Pipfile. Considering this to be the project home.
Pipfile.lock out of date, updating...
Assuring all dependencies from Pipfile are installed...
Locking [dev-packages] dependencies...
Locking [packages] dependencies...
```

Uninstall everything:

```
$ pipenv uninstall --all
No package provided, un-installing all dependencies.
Found 25 installed package(s), purging...
...
Environment now purged and fresh!
```

Use the shell:

```
$ pipenv shell
Loading .env environment variables...
Launching subshell in virtual environment. Type 'exit' or 'Ctrl+D' to return.
$ █
```

## Documentation

Documentation resides over at [pipenv.org](https://pipenv.org).