Topic: `pipenv`

Python Development Workflow for Humans

Presenter: Jay Hull

github.com/JayHull/pipenv-lightning-talk

pipenv

Time to start using it

https://github.com/pypa/pipenv

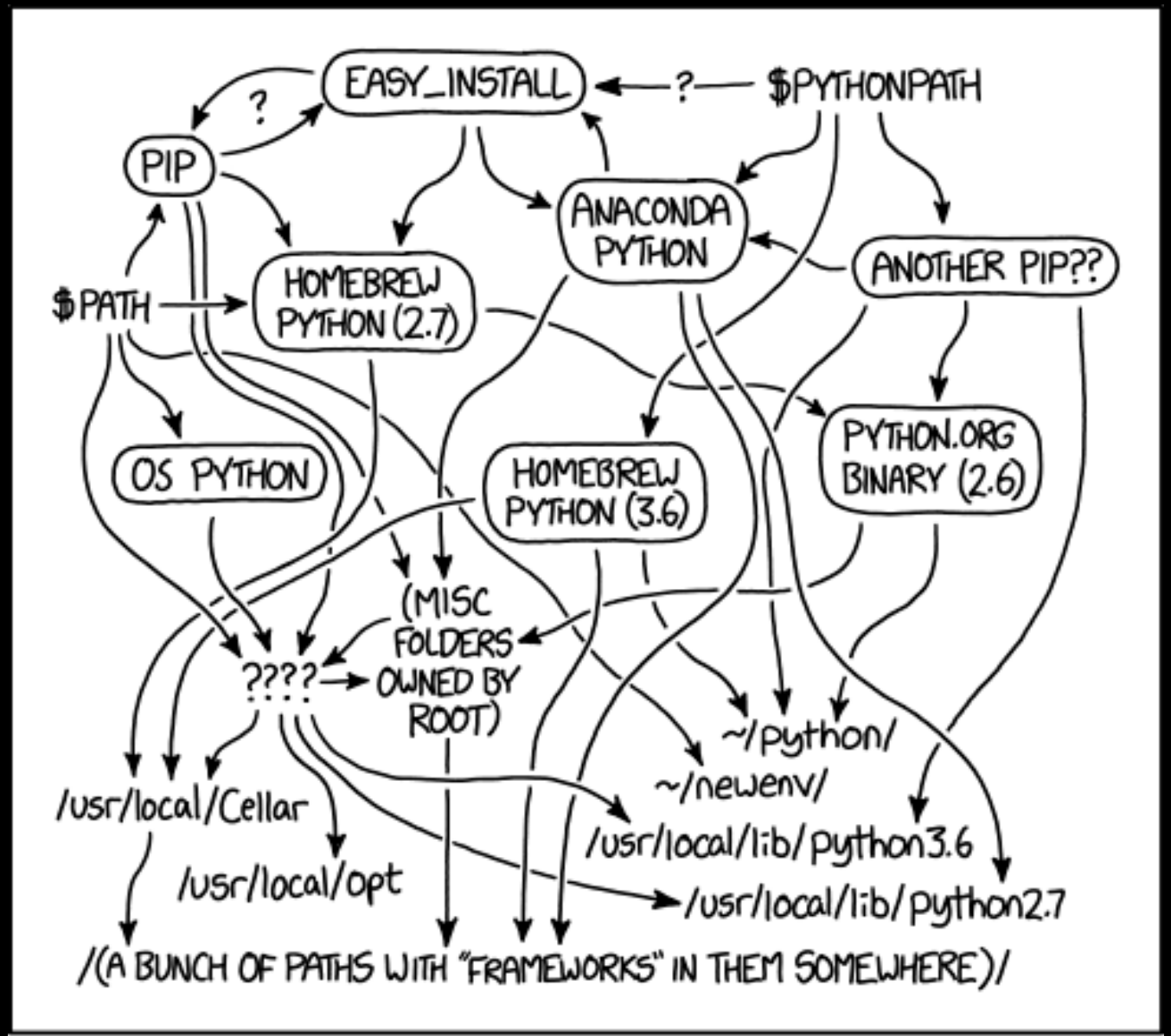
http://kennethreitz.org/

So you need to…

Manage your application’s dependencies…

One old way to manage environments:

```
$ python -m venv --copies .env
$ .env\Scripts\activate.bat
$ python -m pip install -U setuptools pip
```



"Python Environment"

https://www.xkcd.com/1987/

What is the official recommendation?

https://packaging.python.org/tutorials/managing-dependencies/#managing-dependencies

Managing Application Dependencies

The package installation tutorial covered the basics of getting set up to install and update Python packages.

However, running these commands interactively can get tedious even for your own personal projects, and thin
cult when trying to set up development environments automatically for projects with multiple contributors.

This tutorial walks you through the use of Pipenv to manage dependencies for an application. It will show you
the necessary tools and make strong recommendations on best practices.

Keep in mind that Python is used for a great many different purposes, and precisely how you want to mana
may change based on how you decide to publish your software. The guidance presented here is most direct
velopment and deployment of network services (including web applications), but is also very well suited to n
and testing environments for any kind of project.

Note: This guide is written for Python 3, however, these instructions should also work on Python 2.7.

Installing Pipenv

Pipenv is a dependency manager for Python projects. If you're familiar with Node.js' npm or Ruby's bundler,
those tools. While pip alone is often sufficient for personal use, Pipenv is recommended for collaborative pr
level tool that simplifies dependency management for common use cases.

Use `pip` to install Pipenv:

```
pip install --user pipenv
```

pipenv benefits…

- ✓ No need to remember where venv is located
- ✓ Creates environment in a separate user directory
- ✓ Manages dependencies
- ✓ Hashes are used everywhere
- ✓ Can `pipenv update`

Features:

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`).

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.

Problem:

User wants to try python 3.6, but only has python 3.5

AND

Doesn’t want to uninstall 3.5 or change environment variables

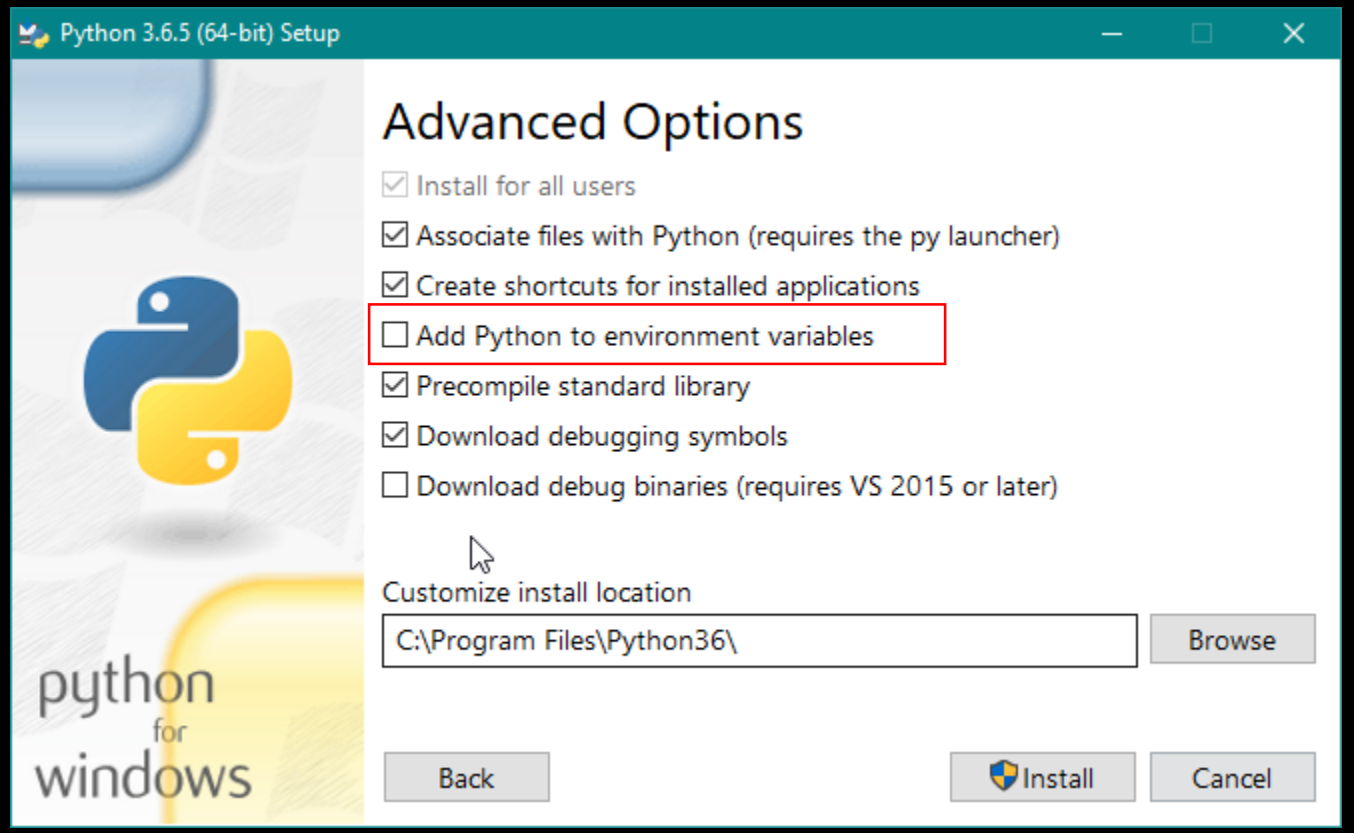
AND

Doesn’t like dealing with virtualenv

Step 1: Download python 3.6

https://www.python.org/downloads/release/python-365/

Step 2: Install python 3.6 && don’t add to env variables



Step 3: Install pipenv in system python

```
$ pip install pipenv -U
```

-or-

```
$ pip3 install pipenv -U
```

Step 4:

```
$ mkdir piedpiper
$ cd piedpiper
```

Step 5: make a new env with python 3.6

```
$ pipenv --python 3.6
```

```
$ pipenv --python /path/to/python
```

Example: using python from a conda env:

```
$ pipenv --python C:\Users\neo\Anaconda3\envs\py36\python
```

```
trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> pipenv --python 3.6
Creating a virtualenv for this project…
1 > Using C:\Program Files\Python36\python.exe (3.6.5) to create virtualenv…
Running virtualenv with interpreter C:\Program Files\Python36\python.exe
Using base prefix 'C:\\Program Files\\Python36'
2 > New python executable in C:\Users\trinity\.virtualenvs\piedpiper-NK61Z0Vn\Scripts\python.exe
Installing setuptools, pip, wheel...done.

3 > Virtualenv location: C:\Users\trinity\.virtualenvs\piedpiper-NK61Z0Vn
4 > Creating a Pipfile for this project…

trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
>
```

```
trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> python --version
Python 3.5.5
```

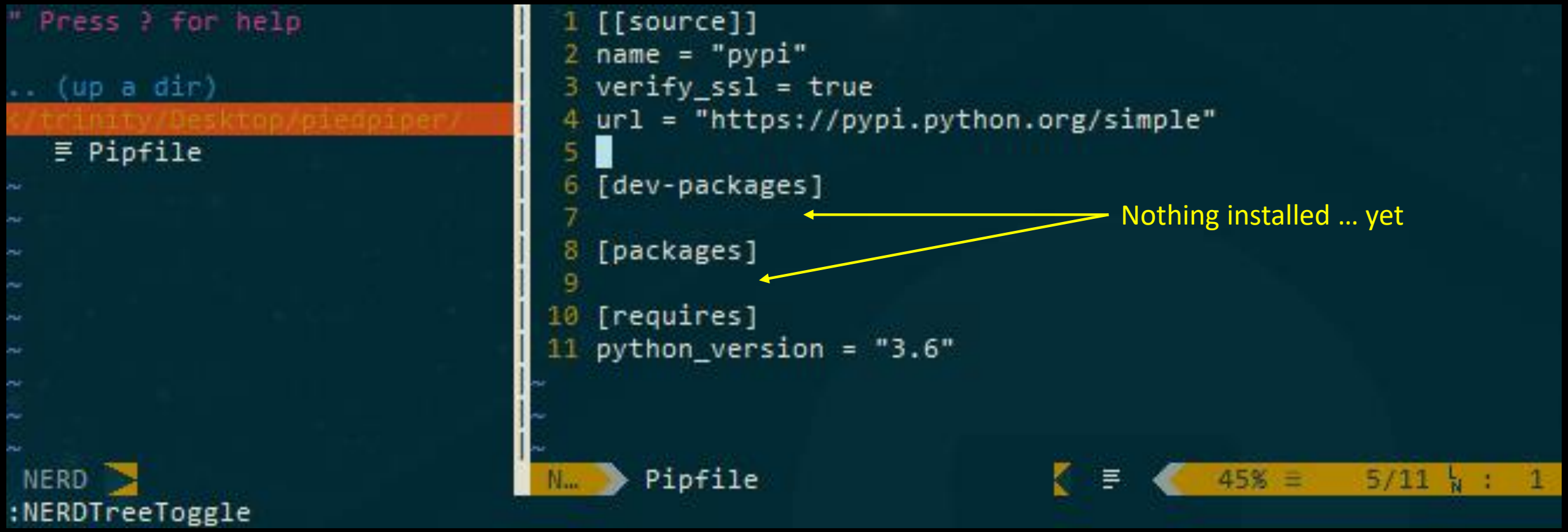
1. Automatically finds 3.6 we just installed
2. Creates new python executable in a virtualenv directory
3. virtualenv location: userhome/.virtualenvs/{project_dir}-{hash_Pipfile_location}/…
4. Finally, creates a Pipfile

From no python 3.6 to project environment in 5 steps:

1. Download python 3.6
2. Install python 3.6, don’t add to environment variables
3. `$ pip install pipenv –U`
4. `$ mkdir piedpiper && $ cd piedpiper`
5. `$ pipenv --python 3.6`

Digging into `pipenv`

Inside a brand spanking new Pipfile



Accessing the environment, what's in $ pip list

System python >

Access the environment >

`pipenv shell`

Environment python >

Starter pip list >

```
trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> python --version
Python 3.5.5

trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> pipenv shell
Launching subshell in virtual environment. Type 'exit' to return.
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> python --version
Python 3.6.5

trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> pip list
DEPRECATION: The default format will switch to columns in the future. You can use --format=(leg
acy|columns) (or define a format=(legacy|columns) in your pip.conf under the [list] section) to
 disable this warning.
pip (9.0.3)
setuptools (39.0.1)
wheel (0.31.0)
```

Step 6: install a dev package

```
trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> pipenv install pytest --dev
Installing pytest…
Collecting pytest
  Using cached pytest-3.5.0-py2.py3-none-any.whl
Collecting pluggy<0.7,>=0.5 (from pytest)
Collecting colorama; sys_platform == "win32" (from pytest)
  Using cached colorama-0.3.9-py2.py3-none-any.whl
Collecting py>=1.5.0 (from pytest)
  Using cached py-1.5.3-py2.py3-none-any.whl
Collecting more-itertools>=4.0.0 (from pytest)
  Using cached more_itertools-4.1.0-py3-none-any.whl
Collecting six>=1.10.0 (from pytest)
  Using cached six-1.11.0-py2.py3-none-any.whl
Requirement already satisfied: setuptools in c:\users\trinity\.virtualenvs\piedpiper-nk6lz0vn
\lib\site-packages (from pytest)
Collecting attrs>=17.4.0 (from pytest)
  Using cached attrs-17.4.0-py2.py3-none-any.whl
Installing collected packages: pluggy, colorama, py, six, more-itertools, attrs, pytest
Successfully installed attrs-17.4.0 colorama-0.3.9 more-itertools-4.1.0 pluggy-0.6.0 py-1.5.3
 pytest-3.5.0 six-1.11.0

Adding pytest to Pipfile's [dev-packages]…
Pipfile.lock not found, creating…
Locking [dev-packages] dependencies…
Locking [packages] dependencies…
Updated Pipfile.lock (db4635)!
Installing dependencies from Pipfile.lock (db4635)…
  ================================ 7/7 - 00:00:02
To activate this project's virtualenv, run the following:
 $ pipenv shell
```

`pipenv install pytest --dev`

< Collects all dependencies

- < pluggy
- < colorama
- < py
- < more-itertools
- < six
- < attrs

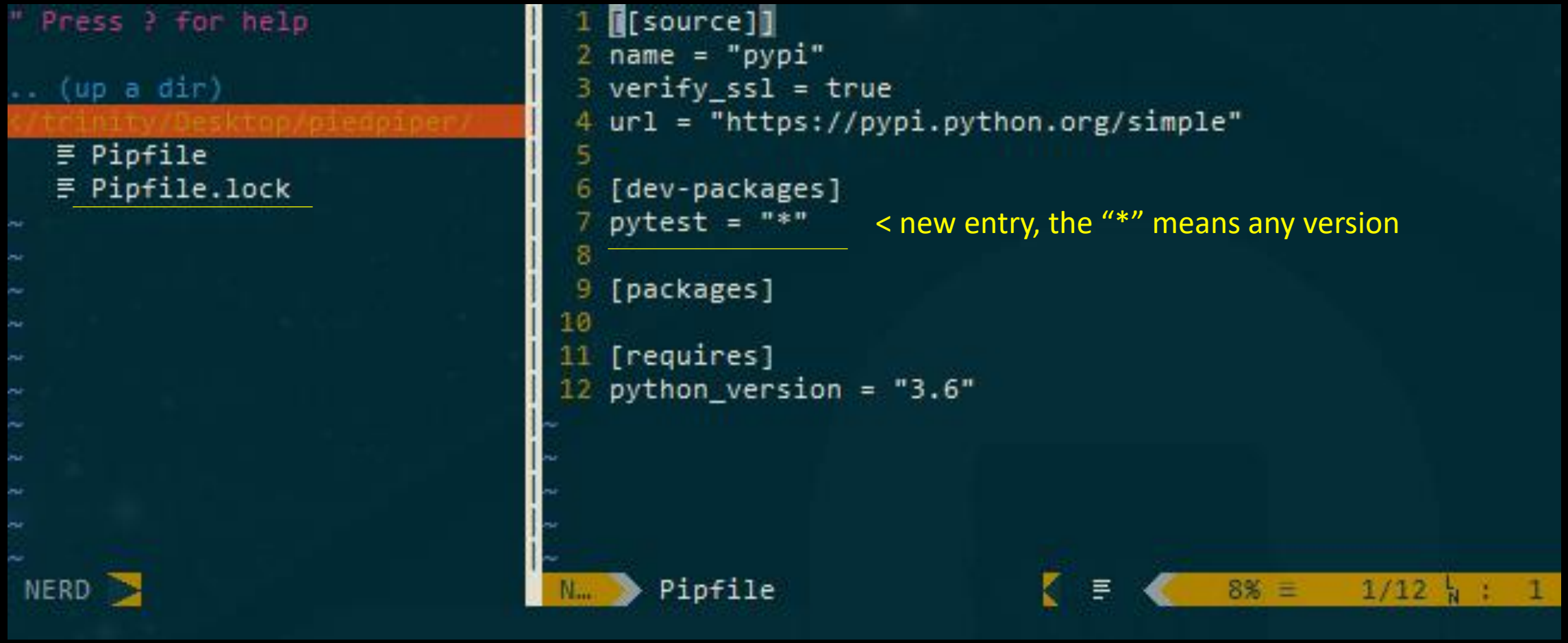
< Installs all dependencies

< Adds pytest to Pipfile’s [dev-packages]!

< Creates a Pipfile.lock

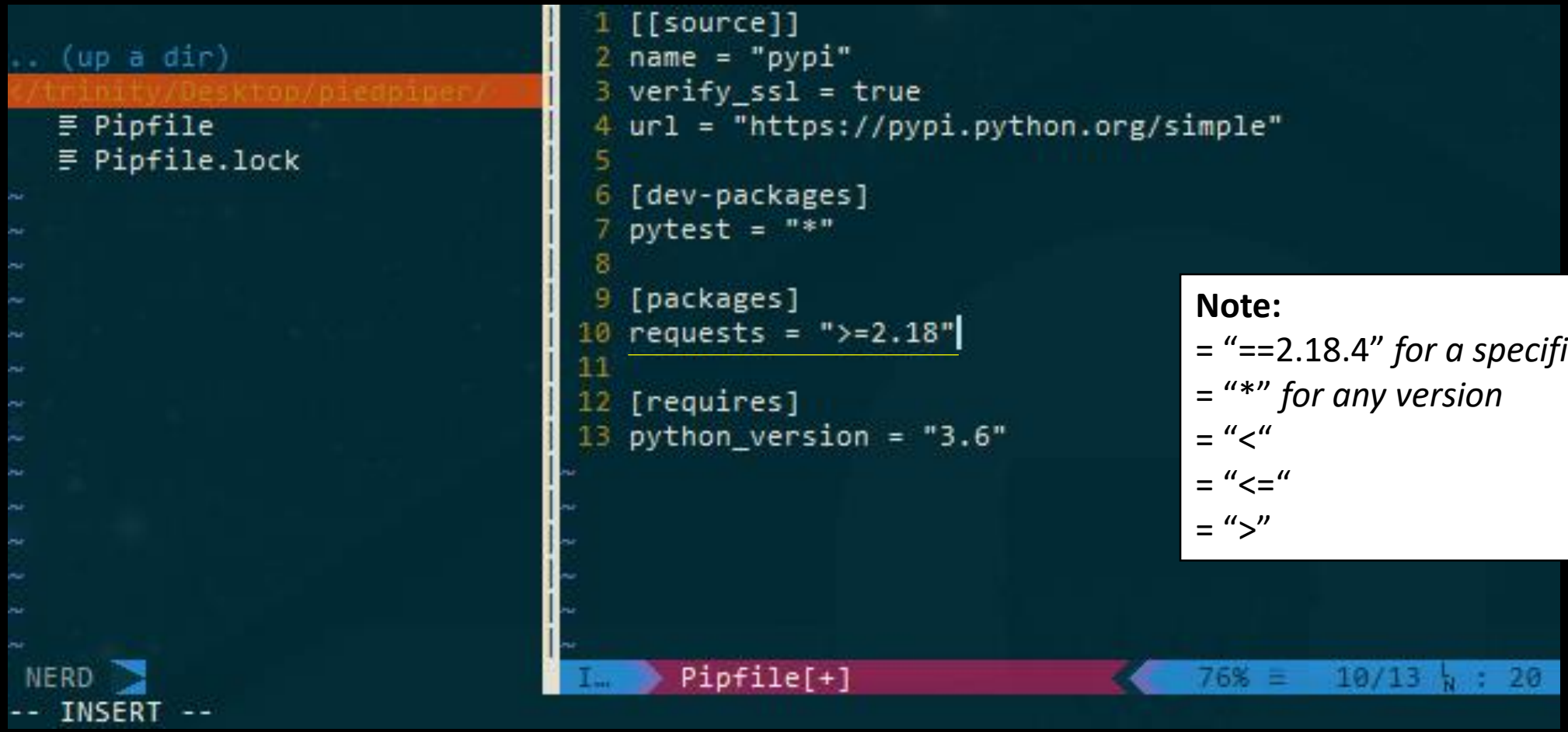
< Installs dependencies from pipfile.lock

Check the Pipfile again



< new entry, the “*” means any version

Can we install packages by putting them in Pipfile?



Note:

= “==2.18.4” *for a specific version*

= “*” *for any version*

= “<“

= “<=“

= “>”

Trying `$ pipenv install`

```
$ pipenv install
```

```
trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> pipenv install
Pipfile.lock (db4635) out of date, updating to (30c4ca)…
Locking [dev-packages] dependencies…
Locking [packages] dependencies…
Updated Pipfile.lock (30c4ca)!
Installing dependencies from Pipfile.lock (30c4ca)…
  ================================ 5/5 - 00:00:02
```

Installed 5 packages! >

… But what were they?

Let's check $ pip list

Oops! > Didn't need to pipenv shell

```
trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> pipenv shell
Shell for C:\Users\trinity\.virtualenvs\piedpiper-NK61Z0Vn already activated.
No action taken to avoid nested environments.

trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> pip list
DEPRECATION: The default format will switch to columns in the future. You can use -
-format=(legacy|columns) (or define a format=(legacy|columns) in your pip.conf unde
r the [list] section) to disable this warning.
attrs (17.4.0)
certifi (2018.1.18)
chardet (3.0.4)
colorama (0.3.9)
idna (2.6)
more-itertools (4.1.0)
pip (9.0.3)
pluggy (0.6.0)
py (1.5.3)
pytest (3.5.0)
requests (2.18.4)
setuptools (39.0.1)
six (1.11.0)
urllib3 (1.22)
wheel (0.31.0)
```

It worked! >

… But what about the other 4 packages?

Let's check `$ pipenv graph`

pytest >

and dependencies:

requests >

and 4 dependencies:

```
trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> pipenv graph
pytest==3.5.0
  - attrs [required: >=17.4.0, installed: 17.4.0]
  - colorama [required: Any, installed: 0.3.9]
  - more-itertools [required: >=4.0.0, installed: 4.1.0]
    - six [required: >=1.0.0,<2.0.0, installed: 1.11.0]
  - pluggy [required: >=0.5,<0.7, installed: 0.6.0]
  - py [required: >=1.5.0, installed: 1.5.3]
  - setuptools [required: Any, installed: 39.0.1]
  - six [required: >=1.10.0, installed: 1.11.0]
requests==2.18.4
  - certifi [required: >=2017.4.17, installed: 2018.1.18]
  - 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]
```

Let’s uninstall requests and see what happens

```
trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> pipenv uninstall requests
Un-installing requests…
Uninstalling requests-2.18.4:
  Successfully uninstalled requests-2.18.4

Removing requests from Pipfile…
Locking [dev-packages] dependencies…
Locking [packages] dependencies…
Updated Pipfile.lock (db4635)!
```

```
pipenv uninstall requests
```

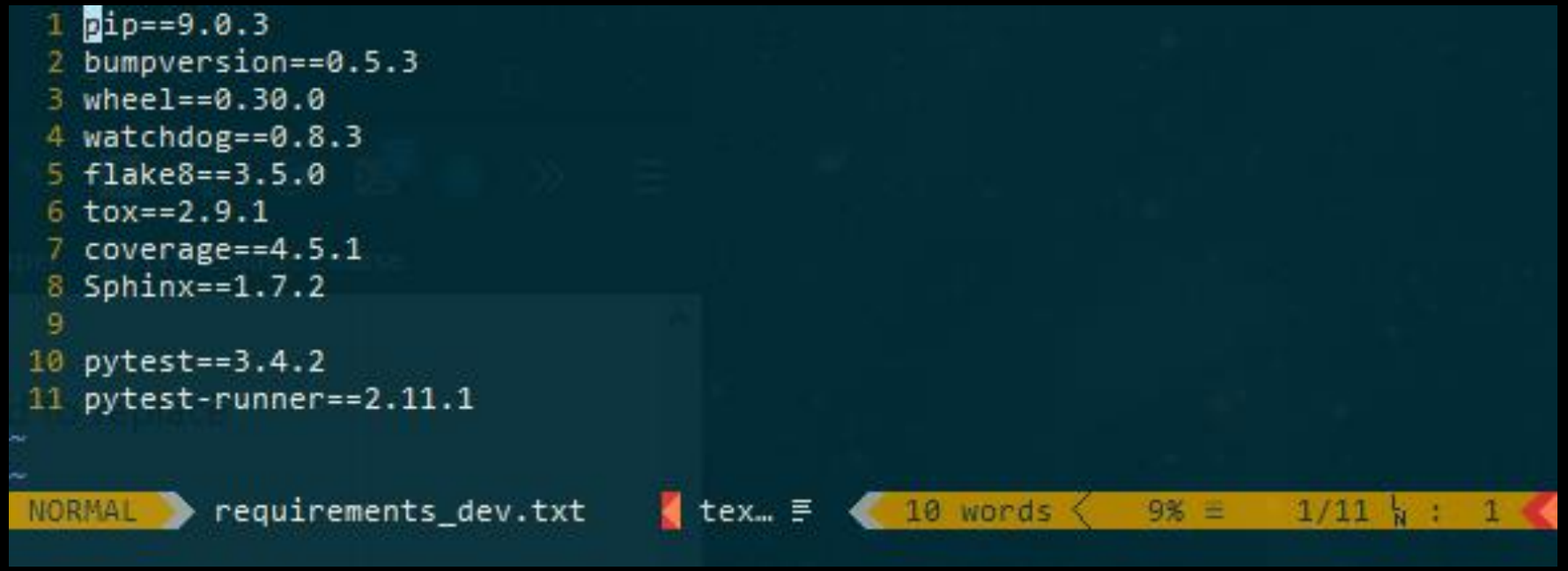
And in the Pipfile…

No more requests = “>=2.18”! >

```
 1 [[source]]
 2 name = "pypi"
 3 verify_ssl = true
 4 url = "https://pypi.python.org/simple"
 5
 6 [dev-packages]
 7 pytest = "*"
 8
 9 [packages]
10
11 [requires]
12 python_version = "3.6"
~
NORMAL  Pipfile                                  83% =  10/12 ℓN :  1
```

Problem:

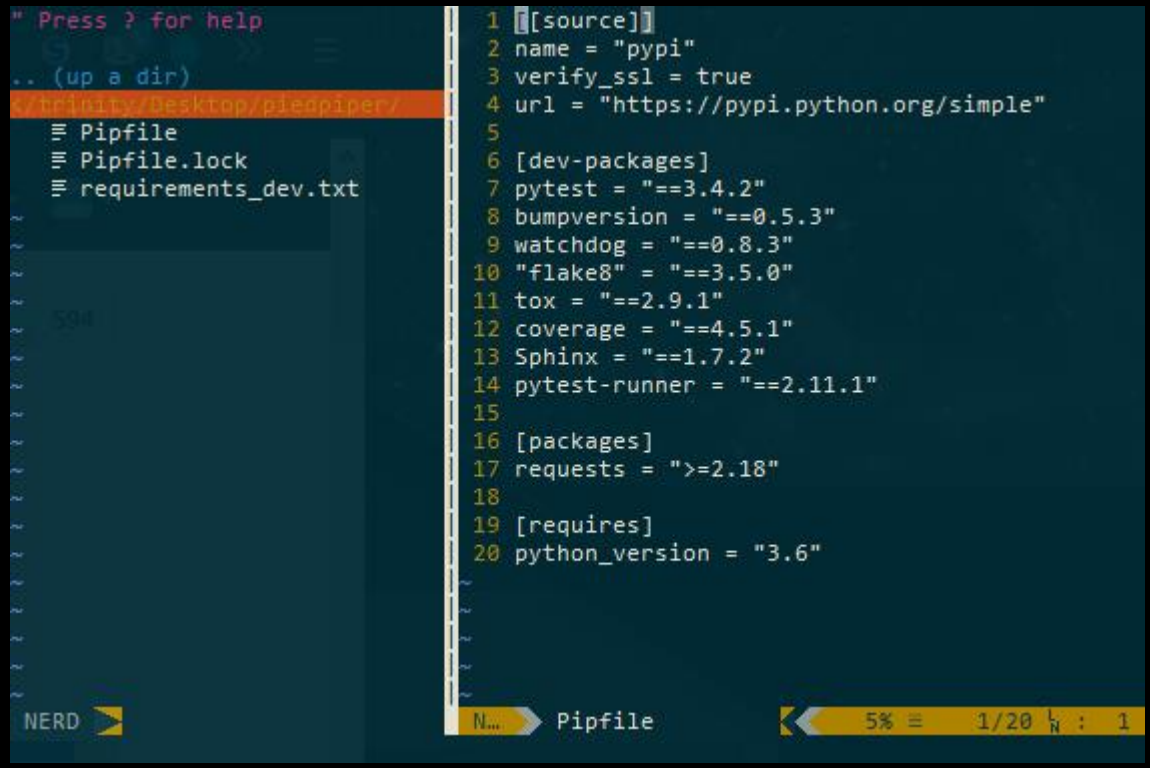
Uh, I’m lazy and already use requirements.txt...



$ pipenv install -r requirements.txt

```
trinity@TRINITY-CAVE C:\Users\trinity\Desktop\piedpiper
> pipenv install -r requirements_dev.txt --dev
Requirements file provided! Importing into Pipfile…
Pipfile.lock (30c4ca) out of date, updating to (de5c6d)…
Locking [dev-packages] dependencies…
Locking [packages] dependencies…
Updated Pipfile.lock (de5c6d)!
Installing dependencies from Pipfile.lock (de5c6d)…
  ================================ 42/42 - 00:00:51
To activate this project's virtualenv, run the following:
 $ pipenv shell
```

< Can pass the `--dev` flag to install under [dev-packages]



`pipenv --python 3.6`	Initiates env with python3.6
`pipenv install <package> --python 3.6`	Initiates env with python3.6 (if not exists) and installs <package>
`pipenv install -r requirements.txt`	Initiates env (if not exists) and installs packages from requirements.txt
`pipenv uninstall <package>`	Uninstalls <package> and dependencies and removes from Pipfile
`pipenv graph`	Show dependency graph
`pipenv --rm`	Remove env for this project, leaves Pipfile
`pipenv update`	Update all dependencies (!)
`pipenv shell`	Activate the environment (like activate
`exit`	Exit the environment
`pipenv run <command>`	Run a command in the env

Further reading…

pipenv docs

Pipfile vs. setup.py

Kenneth Reitz's Future of Python Dependency Management slides

Thank you.

Another nice python env management project is: **pipsi**

Which is for python applications you want to use from the command line but don’t want to mix up with your system python, such as:

youtube-dl

httpie

rtv