PACKAGING AND DISTRIBUTING PYTHON PROJECTS

NATALIE SEREBRYAKOVA,
SOFTWARE ENGINEER AUTOMATION AND TOOLING
@TESTINGRY

https://github.com/nserebry/foo

GOOD PRACTICE:

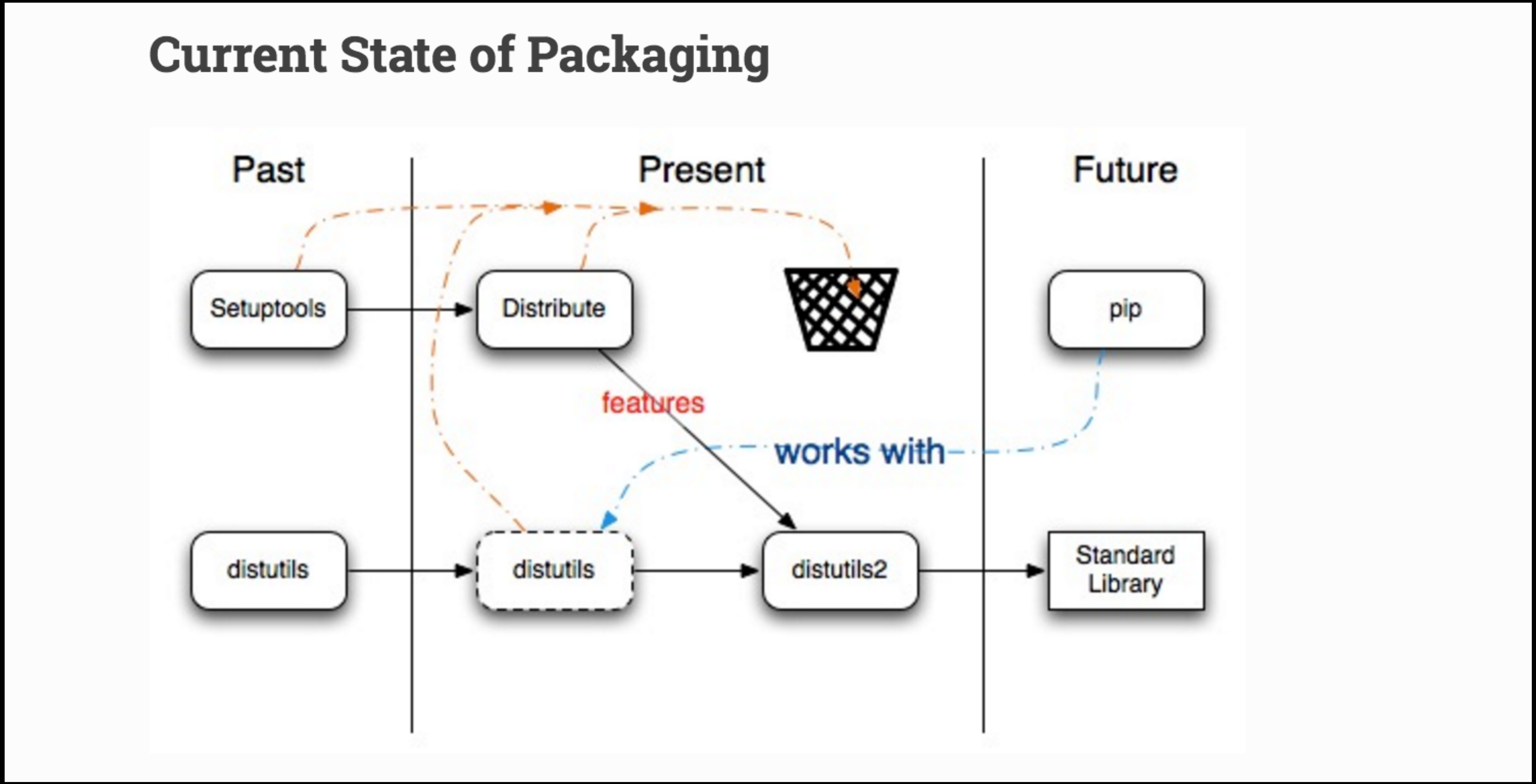
- Python Packaging User Guide (PyPUG) is the authoritative resource on how to package, publish and install Python distributions
- Benefits:

  dependency management
  versioning
  search
  maintenance (+ uninstall)

- Make sure your setup is updated before you start:

```
$ python3 -m ensurepip --user
$ python3 -m pip install --user --upgrade pip
$ python3 -m pip install --user --upgrade virtualenv
```

"THIRD PARTY" PYTHON MODULES AND PACKAGES (PACKAGING TOOLCHAIN)



PYTHON TERMS:

- **Modules :**
  single Python file (or files)
- **Package:**
  is a package (or a directory) of Python modules containing an additional
__init__.py file
- **root package:**
the root of the hierarchy of packages. (This isn't really a package, since it doesn't have an __init__.py file)

- **import package**: commonly called 'package'. Used to organize your code namespace (directory with __init__.py)
- **distribution package**: shareable/installable 'bundled up' python code. (sdist - source distribution with python source code and/or C ext. Run code to install; buildable - bdist_wheel (dropped on the system without anything being run)

SETUP.PY - THE MAIN SETUP CONFIG FILE

- there are no tools that help you to write and update a setup.py. Or a setup.python.json or something, so you actually need to do it yourself
- setup.py is a python file, which usually tells you that the module/package you are about to install have been packaged and distributed
- example:

```
#!/usr/bin/env python3
# from distutils.core import setup (BAD PRACTICE )

from setuptools import setup, find_packages
setup(
  name='foo',
  version='1.0',
  license='MIT',
  keywords='foo very useful module',
  description='Python Distribution Utilities',
  author='testingry',
  author_email='testingry@gmail.com',
  url='https://github.com/',
  packages=find_packages(exclude=['*test', 'doc']),
  entry_points={
      "console_scripts": [
         'foo=foo.main:main'] }
)
```

- If an end-user wishes to install your 'foo' module,

```
$ python3 setup.py install
```

“THIRD PARTY MODULES”

- *Distribute* is a collection of enhancements to the Python standard library module: distutils
- Using *pip* (The PyPA recommended tool for installing Python packages.)
- Pip is an installer for Python packages written by Ian Bicking. It can install packages, list installed packages, upgrade packages and uninstall packages.

```
$ pip install foo
 (pip install -e git+https://git.repo/foo_pkg.git#egg=Foo
$ pip list  (pip list -o)
$ pip install foo –upgrade
$ pip uninstall foo
```

REQUIREMENTS FILES:

- "Requirements files" are files containing a list of items to be installed using pip install like so:
- Each line of the requirements file indicates something to be installed
- Supports installing from a package index using a requirement specifier(composed of a project name followed by optional version specifiers)

```
$ pip install foo >= 1.3 ( ==, ~=, <=, [])
```

- Project can contain multiple requirements.txt files (test, deploy, etc)
- Can be generated by:

```
$ pip freeze > requirements.txt
$ pip install -r requirements.txt
```

README, LICENSE , MANIFEST (DISTRIBUTION FILES)

- README.txt (or README). Can be used for Pypi listing or index page(github)

- DESCRIPTIONS.rst Can be used for Pypi listing

- LICENSE

- MANIFEST.in manifest template is just a list of instructions for how to generate your manifest file. Listing of anything in your package that should be included from elsewhere.

PACKAGING YOUR PROJECT

- Create a **Distribution Package** : versioned archive file that contains Python packages, modules, and other resource files that are used to distribute a Release. The archive file is what an end-user will download from the internet and install.

- NOT an **Import Package** or **System Package**

- Can be done by Build distribution format: Wheel or Egg(old, don't use without good reason)

- Wheel : built distribution format that provides faster installation compared to Source Distributions (sdist), especially when a project contains compiled extensions.

  (Universal , Pure Python, Platform)

PYPI

- Package index
- Repo for python packages anyone can upload (registration is required)
-
- Create :

```
$ vim  $HOME/.pypirc
```

- Run :

```
$ python3 setup.py sdist
$ python3 setup.py bdist_wheel
$ twine upload dist/*
```

  twine is a utility for interacting with PyPI.

- Before releasing on main PyPI repo, you might prefer training with https://testpypi.python.org/pypi

APPLICATION DEVELOPMENT (DEMO)

- You can follow along : https://github.com/nserebry/foo

- Project example

```
.
|---data
|     |---data_file
|---MANIFEST.in
|---README.rst
|---foo:
|     |---__init__.py
|     |--- foo.py
|---setup.cfg
|---setup.py
|---tests:
      |---__init__.py
      |—test_foo.py
```

Q&A