






Python Application Dependency management

Liuyang Wan

About me

-  Software Engineer @ Zendesk
-  Python and Open source
-  Cycling & Taiwan
-  @sfdye



Agenda

Pip basics/terminology

Dependencies issues & How to resolve them

Behavior of different pip versions

Best practices & Recommendations

Q&A



Basics

- What is a Python package?
- Where?
 - PyPI (public)
 - Artifactory/Github Packages (private)
 - Git submodule (vendor)

The package manager

pip - **P**ython **I**nstall **P**ackage

\$ pip install SomePackage # install one requirement

\$ pip install -r requirements.txt # install requirements specified in a file

```
Flask==0.12.4
Werkzeug==0.15.3
Jinja2==2.10.1
MySQL-python==1.2.5 ; python_version < '3.0'
mysqlclient==1.4.2.post1 ; python_version > '3.0'
celery==3.1.26.post2
ipaddress==1.0.23
python-dateutil==2.8.1
```

An example requirements file

Which version will be installed

- Version Specifier ([PEP-440](#))
 - ==, !=
 - >, <,
 - >=, <=
 - ...
 - >= 1.0, != 1.3.*, < 2.0

Problem 1 - Indeterministic behavior

```
$ pip install requests
```

Question: what version of **requests** will be installed?

Answer:

- It depends, will install the latest version from PyPI at the time of running the pip command

Results: Breaking changes introduced in newer version break CI/CD/App

Solution: pin the version

Problem 2 - Version conflict

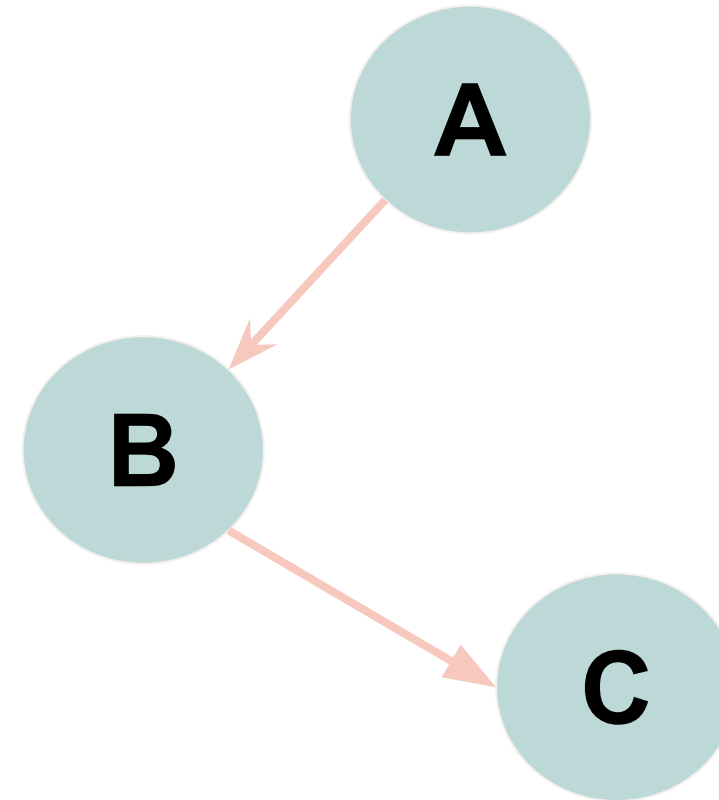
```
$ pip install 'pyopenssl==19.1.0 cryptography==2.7'
```

```
ERROR: pyopenssl 19.1.0 has requirement cryptography>=2.8, but  
you'll have cryptography 2.7 which is incompatible.
```


Dependencies issues - Why

A dependency can have its own its dependency.

- **A** depends on **B**
- **B** depends on **C**



What data structure? Tree! A dependency tree

Visualize the dependency tree

```
$ pipdeptree
```

```
pyOpenSSL==19.1.0
```

- **cryptography [required: >=2.8, installed: 2.7]**
 - asn1crypto [required: >=0.21.0, installed: 1.4.0]
 - cffi [required: >=1.8,!=1.11.3, installed: 1.14.1]
 - pycparser [required: Any, installed: 2.20]
 - six [required: >=1.4.1, installed: 1.15.0]
- six [required: >=1.5.2, installed: 1.15.0]

Possible ways to resolve version conflicts

```
$ pip install 'pyopenssl==19.1.0 cryptography==2.7'
```

ERROR: pyopenssl 19.1.0 has requirement cryptography>=**2.8**, but you'll have cryptography **2.7** which is incompatible.

Option 1: bump cryptography to >=2.8

Option 2: downgrade pyopenssl version to loosen constraint

Option 3: remove pyopenssl if not used

Behavior of different pip versions

pip 8.x

- Does not support python environment markers such as *python_version*
e.g.
`someLib==X.Y ; python-version > '3.0'`

pip 9.x

```
$ pip install oslo.utils==1.4.0
```

Successfully installed Babel-2.8.0
importlib-resources-3.0.0
iso8601-0.1.12 netaddr-0.8.0
netifaces-0.10.9 **oslo.i18n-5.0.0**
oslo.utils-1.4.0 **pbr-0.11.1** pytz-2020.1
six-1.15.0 zipp-3.1.0

pip 19.x

```
$ pip install oslo.utils==1.4.0
```

ERROR: oslo-i18n 5.0.0 has
requirement pbr!=2.1.0,>=2.0.0, but
you'll have pbr 0.11.1 which is
incompatible.

Successfully installed Babel-2.8.0
importlib-resources-3.0.0
iso8601-0.1.12 netaddr-0.8.0
netifaces-0.10.9 **oslo.i18n-5.0.0**
oslo.utils-1.4.0 **pbr-0.11.1** pytz-2020.1
six-1.15.0 zipp-3.1.0

pip's new resolver

pip 20.1 (alpha)

Released in May 2020

```
$ pip install
```

```
--unstable-feature=resolver
```

```
oslo.utils==1.4.0
```

Successfully installed Babel-2.8.0

importlib-resources-3.0.0

iso8601-0.1.12 netaddr-0.8.0

netifaces-0.10.9 **oslo.i18n-2.1.0**

oslo.utils-1.4.0 **pbr-0.11.1** pytz-2020.1

six-1.15.0 zipp-3.1.0

pip 20.2 (beta)

Release in July 2020

```
$ pip install
```

```
--use-feature=2020-resolver
```

```
oslo.utils==1.4.0
```

Successfully installed Babel-2.8.0

importlib-resources-3.0.0

iso8601-0.1.12 netaddr-0.8.0

netifaces-0.10.9 **oslo.i18n-2.1.0**

oslo.utils-1.4.0 **pbr-0.11.1** pytz-2020.1

six-1.15.0 zipp-3.1.0

pip 20.3 (stable)

To be released in Oct 2020

Pinning version: where and how?

- Library VS Application

Library: setup.py/setup.cfg/pyproject.toml

Application: requirements.txt/main.txt/test.txt

```
from setuptools import setup

setup(
    name='zopim-utils',
    install_requires=[
        "six>=1.10.0,<2",
        "requests>=2.22.0,<3",
        "python-dateutil>=2.8.0,<3",
        "MySQL-python>=1.2.5,<2 ; platform_python_implementation != 'PyPy'",
        "mysqlclient>=1.4.2.post1,<2 ; platform_python_implementation != 'PyPy'",
        "simplejson>=3.16.0,<4",
        "dogstatsd-python>=0.5.1,<1",
    ],
)
```

setup.py for a library

```
Flask==0.12.4
Werkzeug==0.15.3
Jinja2==2.10.1
MySQL-python==1.2.5 ; python_version < '3.0'
mysqlclient==1.4.2.post1 ; python_version > '3'
celery==3.1.26.post2
ipaddress==1.0.23
python-dateutil==2.8.1
```

requirements.txt for an application

**Rule of thumb: requirements.txt
should contain only ==, while setup.py
everything except ==.**

**Always pin versions and use tools like
dependabot to keep your
dependencies up-to-date and secure.**

Take-away

- Always pin the version (ROT: requirements.txt should contain only ==, while setup.py everything except ==)
- Upgrade pip to 20.x to prepare for the new resolver
- Use tools like dependabot to keep dependencies up-to-date and secure
- Same principle applies to other languages and their package managers

thank you