

pip & packaging

Xavier Fernandez

DjangoCong 2016



Résolution de dépendances dans pip

Résolution de dépendances dans pip



problem?

(non-)Résolution des dépendances

- Premier arrivé premier servi
- Pip choisit la version la plus récente du 1^{er} specifier
- Il ignore les suivants

(non-)Résolution des dépendances

- Imaginons que C existe en v1, v1.5, v2
- Si A demande $C==1$ et B demande $C==2$
 - `pip install A` B installe $C==1$
 - `pip install B` A installe $C==2$
- Si A demande $C \geq 1$ et B demande $C < 2$
 - `pip install A` B installe $C==2$
 - `pip install B` A installe $C==1.5$

Les contraintes

Fichier de contraintes

- Depuis pip 7.1, via l'option `--constraint/-c`
- Comme les `requirements.txt`, mais:
 - Ils n'occasionnent aucune installation
 - Ils ont le dernier mot sur la version installée
- Pratique pour déclarer des politiques générales du genre:
 - Si un paquet a besoin de `spyne`, il doit utiliser `spyne==2.12.10`
`+polyconseil`

Rolling release de librairies internes

- Des SIs qui ne fixent pas ces dépendances
- Uniquement des combinaisons testées en prod
- Solution:
 - On installe et teste en rolling
 - On `pip freeze` pour connaître la combinaison gagnante
 - Pas de `pip install -r freeze.txt`
 - Mais `pip install si_package -c freeze.txt`

Packaging & PEPs

Specifications

- PEP 440 -- Version Identification and Dependency Specification
- PEP 503 -- Simple Repository API
- PEP 508 -- Dependency specification for Python Software Packages
- `pip install` packaging

PEP 518 -- Specifying Minimum Build System Requirements for Python Projects

- Un fichier `pyproject.toml`
 - `[build-system]`
 - # Minimum requirements for the build system to execute.
 - `requires = ["setuptools", "wheel"]` # PEP 508 specifications.
- Premier pas pour autoriser d'autres outils de build que `setuptools`
 - `flit` ou `bento`
 - cf les draft PEP 516/517

Merci :)