Lockfiles and scaling uv to larger projects



Mohamed Echout

Senior Technical Trainer

@MEchout | www.mohamedechout.com

Understanding lockfiles and reproducible builds

The reproducibility problem

Dev laptop

```
$ pip install -r requirements.txt

Installed:
   requests==2.32.0
   certifi==2025.11.5
   urllib3==2.2.4

Tests pass.
```

CI runner

```
$ pip install -r requirements.txt

Installed:
   requests==2.33.1
   certifi==2026.2.25
   urllib3==2.6.3

Tests fail.
```

Comparing what each file does



`requirements.txt`

- Lists direct deps only
- Constraints, not versions
- No transitive deps recorded
- No hashes by default
- No platform info
- Resolves at install time



`uv.lock`

- Records every package
- Exact versions, pinned
- Full transitive graph
- Secure algorithm per file
- Platform markers included
- Resolved once, reused

What's inside uv.lock?

An entry from a lockfile

```
[[package]]

name = "requests“

version = "2.33.1“

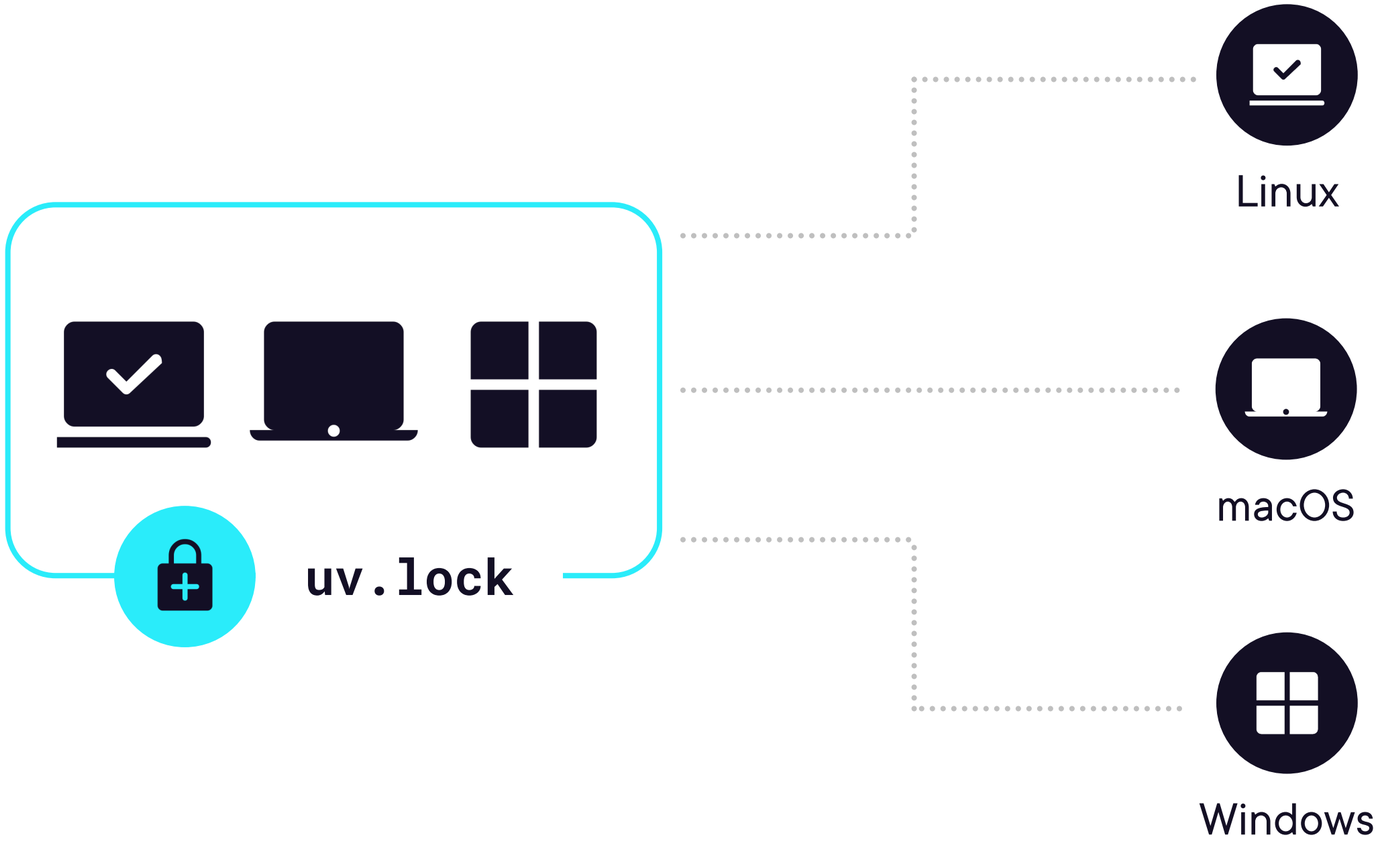
source = {registry = "https://pypi.org/simple"}

dependencies = {name = "urllib3"}

sdist = {url = "URL", hash = "sha256:abc123"}
```



One lockfile for every platform



The reproducibility triangle



`pyproject.toml`

What you want



`uv.lock`

What you got



`uv sync`

What you install



Demo

Generating, updating, and syncing with uv.Lock

Demo

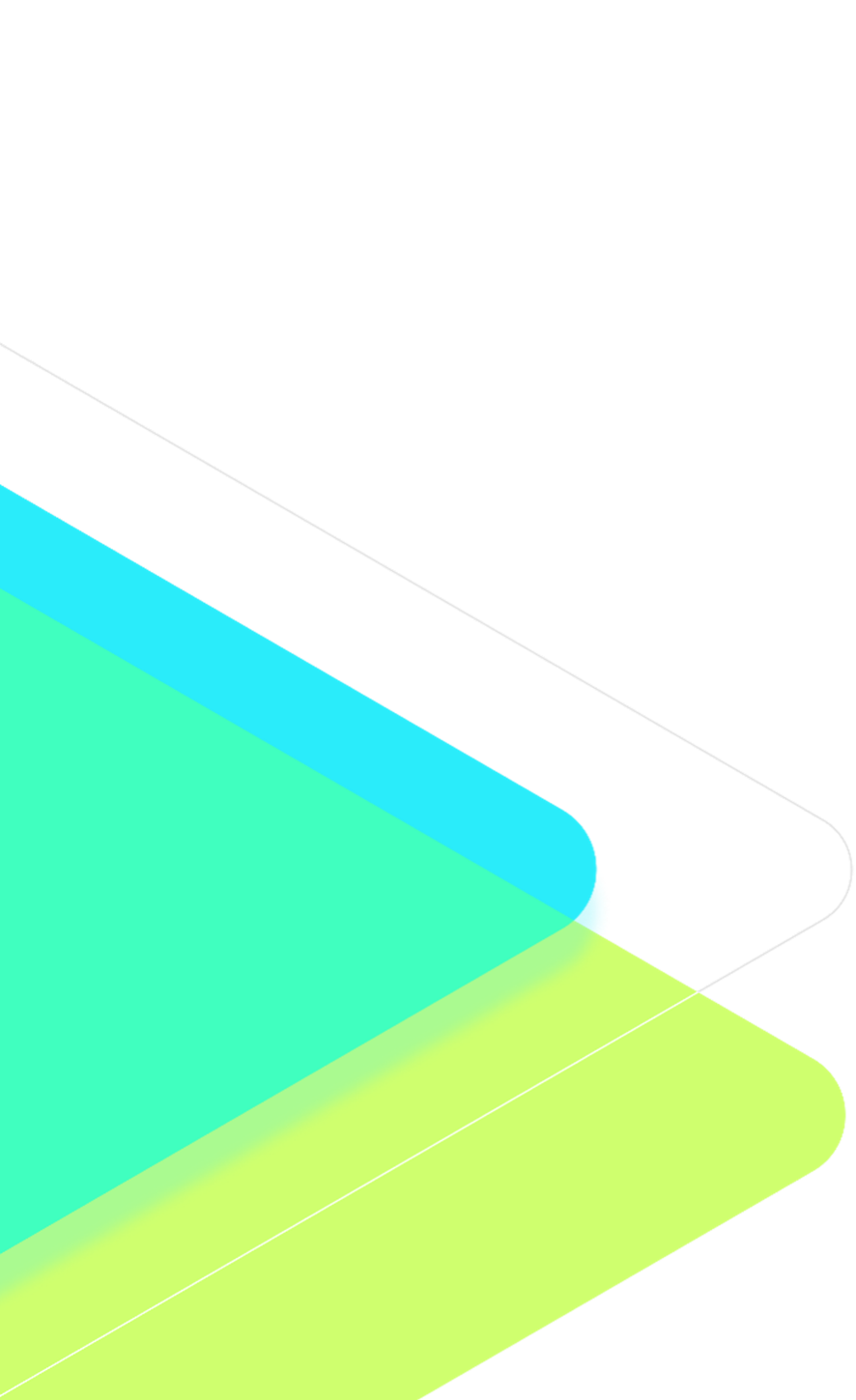
Controlling dependency resolution with strategies and overrides

Demo

Scaling with uv workspaces for multi-package repos

Demo

Using uv in CI for fast, reproducible builds



Pluralsight

Thank you for your time!

@MEchout | www.mohamedechout.com