uv Foundations

Understanding uv and getting started



Mohamed Echout

Senior Technical Trainer

@MEchout | www.mohamedechout.com

uv and the problem with traditional Python tooling

Meet Maya, our Python developer



5 Python services

Each with its own dependencies and version



Mixed team toolchain

Some use **`pip`** and **`venv`**, others **`poetry`**, others **`pyenv`**



Works on her machine

Onboarding takes days, CI is slow, builds drift

The fragmented Python toolchain



`pip`



`poetry`



`venv`



`pipx`



`virtualenv`



`pyenv`

Installs are slow and builds are slower



`pip + venv`

Cold install of 3 packages



`uv`

Same install, plus venv and lockfile creation



Onboarding takes days

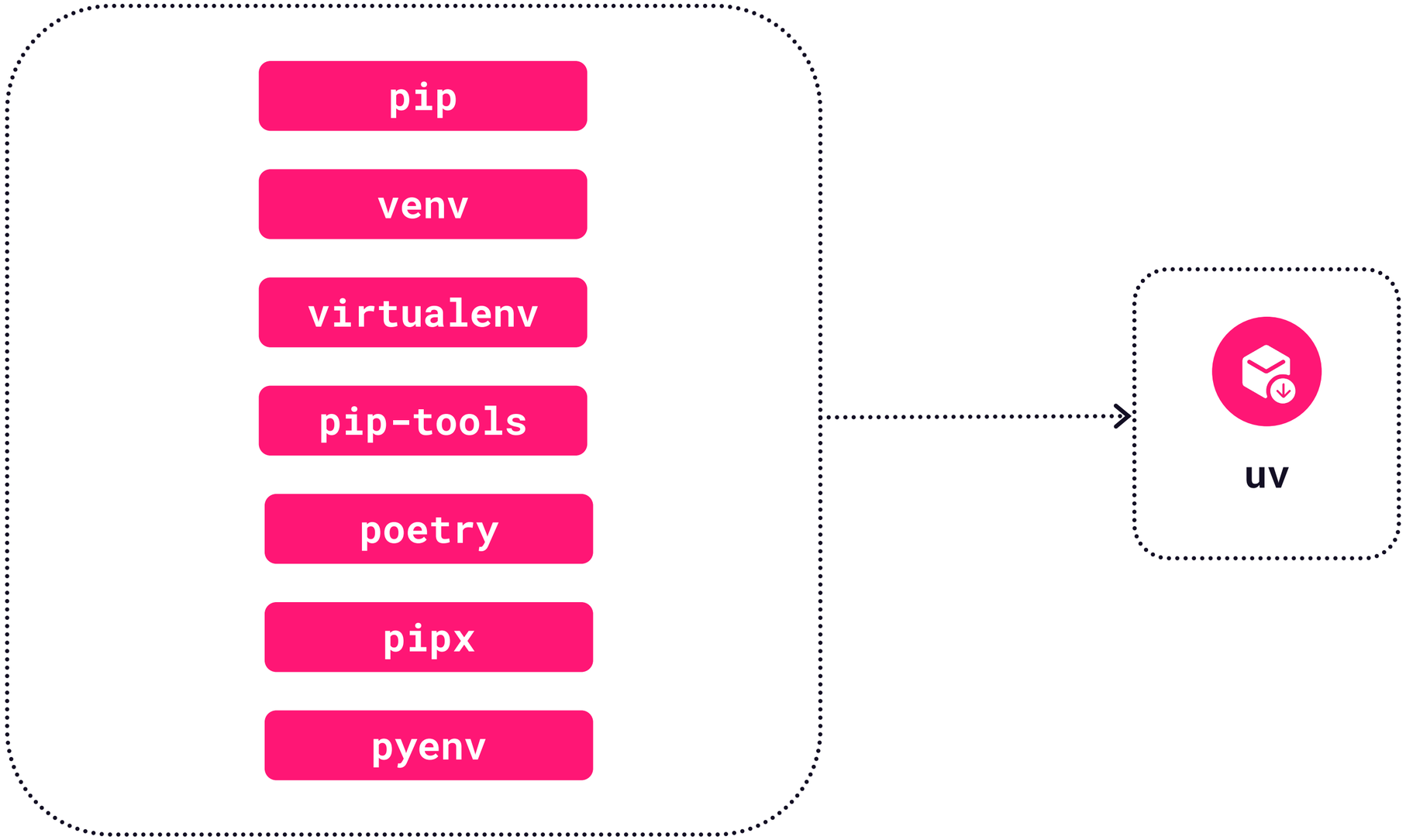
1. Install **pyenv**
2. Install the right Python
3. Create a **venv**
4. Activate the **venv**
5. Install requirements



Pain point

Every step has its own docs, its own version mismatches, and its own platform

What if one tool replaced them all?



What is uv and where does it fit in the SDLC?

Three pain points



Slow installs

`pip + venv` takes time



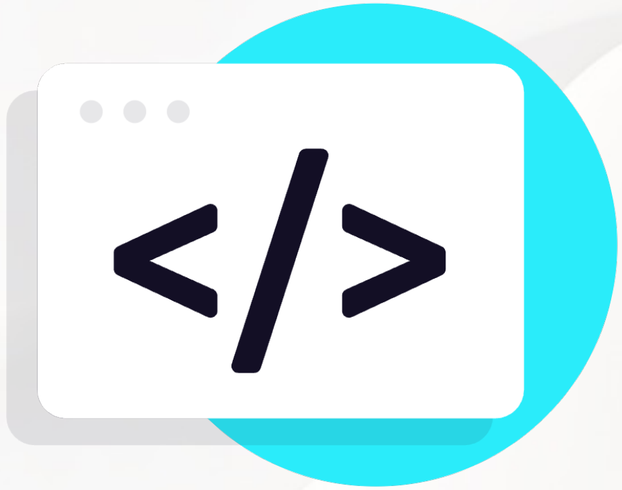
Drift across machines

`requirements.txt` isn't a real lock



Onboarding takes days

Many manual steps just to start



uv

A Python package and project manager

Meet uv



10 to 100 times faster than **`pip`**



Replaces **`pip`**, **`venv`**, **`pip-tools`**, **`Poetry`**, **`pipx`**, **`pyenv`**



Standards-based and supports **`pyproject.toml`** and PEP

uv handles five things

Python

Environments

Dependencies

Lockfiles

CLI tools



uv across the development cycle



Stage 1

Project setup



Stage 2

Daily development



Stage 3

CI and build



Stage 4

Onboard and deploy

What this looks like in practice

```
$ uv init my-app    # Start a new project

$ uv add fastapi    # Add a runtime dependency

$ uv add --dev pytest    # Add a dev dependency

$ uv run main.py    # Run code in the env

$ uv sync --frozen    # Reproduce in CI
```



What this means for Maya



Before uv

- 7 different tools to learn
- **`pip install`** takes 50 seconds
- **`requirements.txt`** drift in CI
- Onboarding takes days of setup
- Different setup per machine



After uv

- 1 tool and one mental model
- **`uv`** add takes 10 seconds
- **`uv.lock`** is deterministic across CI
- Onboarding takes one command
- Same setup everywhere

Demo

Installing uv and managing Python versions

Demo

Initializing a project and exploring the layout

Demo

uv vs. pip and poetry, a side-by-side comparison