From Chaos to Clarity

How uv transformed my Python workflow

by Karim Lalani
for AIMUG SXSW Lightning Talk 2025

About Me - Karim Lalani

- **Home**: Leander, TX
- **Work**: Software Engineer @ Office the Governor
- **Background**: Full Stack Engineer, Gen AI
- **FOSS**: Docker / Kubernetes, C#, Python, PHP, Rust
- **Using LangChain**: Experimentation, learning
- **Socials**:
  - Linkedin https://www.linkedin.com/in/-karim-lalani/
  - Github https://github.com/lalanikarim/
  - Medium https://medium.com/@klcoder

The Pain Points of Python Dev

- Managing multiple Python versions
- Environment chaos with virtualenv
- Docker's overhead for simple projects
- Nix/Devbox complexity with CUDA

Managing multiple Python versions

- Installed Python versions manually
- Conflicting dependencies (e.g., tensorflow, torch, etc)
- Constant "Which Python is active?" confusion

Manual Environment Management

Steps I repeat:

```
virtualenv env
source env/bin/activate
pip install -r requirements.txt
```

Package duplication across different projects

PIP IS SLOWWWWWW!

Containerization Trade-offs

✅ Pros:

- Reproducible dependencies

❌ Cons:

- Slow local development (rebuilding images for small changes)
- Debugging felt clunky in containers

Nix/Devbox Frustrations

Declarative Overkill

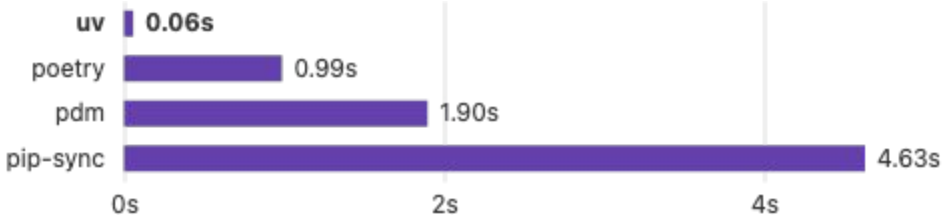
- CUDA version configs required deep Nix expertise
- "Overkill for simple projects, insufficient for complex ones"
- Learn an unrelated skill

How uv Fixed These Issues

- Automatic Python Version Selection
- Zero-Friction Environments
- Reproducibility Without Docker Overhead
- Simpler Dependency Management

Introducing uv from astral.sh

An extremely fast Python package and project manager, written in Rust.



Installing Trio's dependencies with a warm cache.

Highlights

- 🚀 A single tool to replace `pip`, `pip-tools`, `pipx`, `poetry`, `pyenv`, `twine`, `virtualenv`, and more.
- ⚡ 10-100x faster than `pip`.
- 🗂️ Provides comprehensive project management, with a universal lockfile.
- ❇️ Runs scripts, with support for inline dependency metadata.
- 🐍 Installs and manages Python versions.
- 🛠️ Runs and installs tools published as Python packages.
- 🔩 Includes a pip-compatible interface for a performance boost with a familiar CLI.
- 🏢 Supports Cargo-style workspaces for scalable projects.
- 💾 Disk-space efficient, with a global cache for dependency deduplication.
- ⏬ Installable without Rust or Python via `curl` or `pip`.
- 🖥️ Supports macOS, Linux, and Windows.

uv is backed by Astral, the creators of Ruff.

https://docs.astral.sh/uv/#highlights

uv Magic

- No more manual Python installations

  uv python pin 3.12

- Dependencies handled declaratively in python scripts

  uv add langgraph langgraph-checkpoint-sqlite –script script.py

- Run script

  uv run script.py

hil-func-input-sqlite-async.py

```
# /// script
# requires-python = ">=3.12"
# dependencies = [
#     "langgraph",
#     "langgraph-checkpoint-sqlite",
# ]
# ///
```

@lalanikarim

Goodbye Virtualenv, Hello uv

Old workflow:

- Create virtual environment
- Activate
- Install
- Run

New workflow:

- uv run

What about Projects?

- uv init - creates a project in the current directory with pyproject.toml
- uv add <dependencies> - add dependencies to pyproject.toml
- uv run script.py - run script
- uv build - build distribution
- uv publish - publish package to pypi.org

Key Takeaways

uv Saved Me:

- Reduced environment setup time considerably
- Made dependency management understandable again
- Allowed focus on coding instead of tooling
- Made python scripts portable, finally

Final Thoughts

uv = Python Development Unchained

No more version hell, environment chaos, or over complicated tools.

Thank You