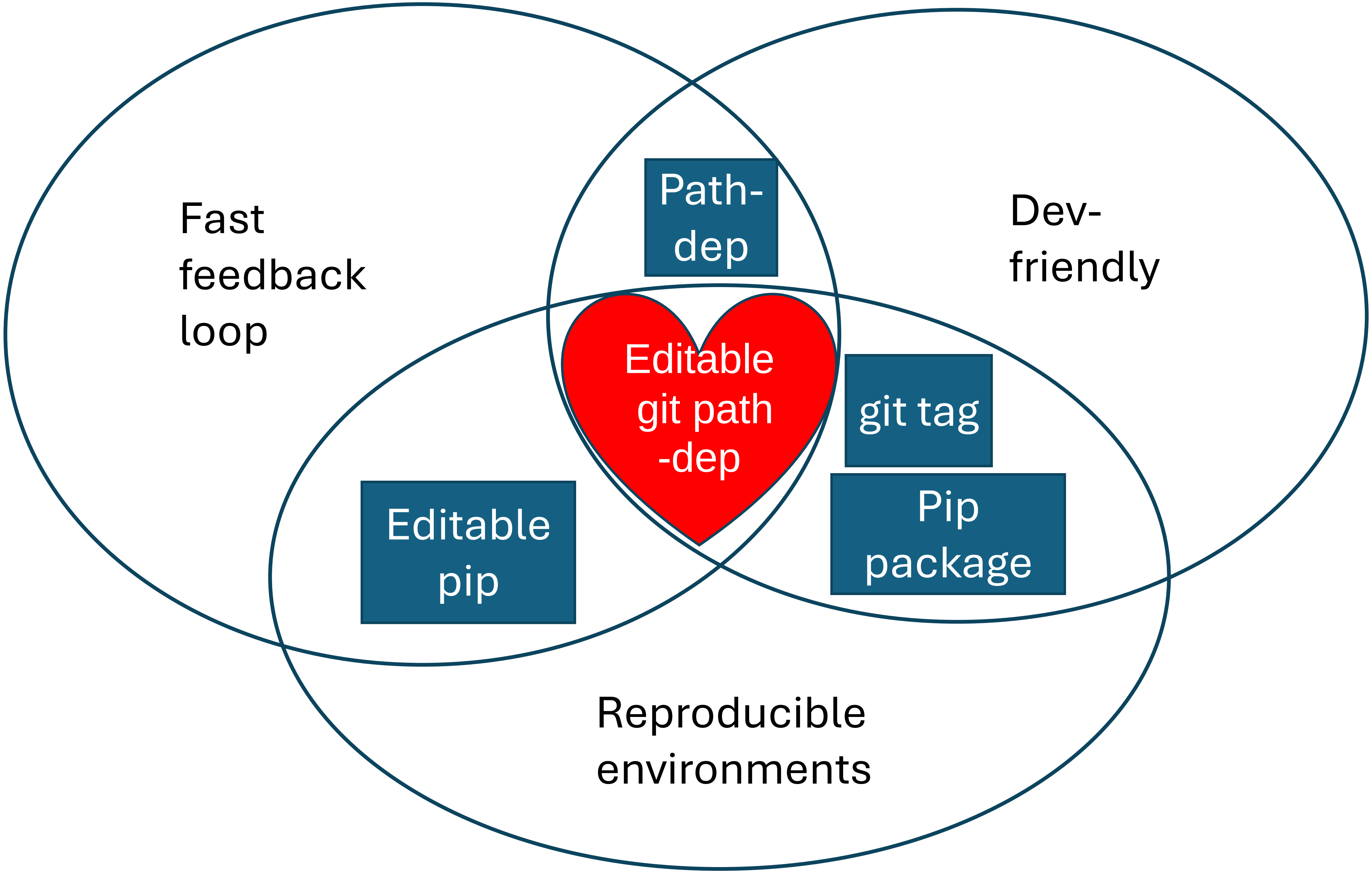
uv editable git-dependency

David Horvath

@systematicguy

Swiss Python Summit 2025

⚡ lightning talk ⚡


Fast feedback loop
Dev-friendly
Path-dep
Editable git path -dep
git tag
Pip package
Editable pip
Reproducible environments

Circumstances

- 📁 Application code is split into multiple repos.
- 🫂 There is shared code.
- 🔒 Private repos.

Skill expectation

- 📥 Developer can git clone the repos.
- 💡 Developer knows basic uv usage.
- 😴 Developer is super lazy and will forget any extra steps to perform.

Architectural Requirements

- Multiple app repos consume the same lib.
  - 📁 lib (1.0.0, 1.1.0, ...)
  - 📁 app1 (lib@1.0.0)
  - 📁 app2 (lib@1.1.0)
- 🔗 1.0.1 prod refers to specific version of lib.
  - Apps update their dependencies separately.

Development requirements

- 😴 Serve the lazy forgetful developer.
- 📦 pyproject.toml works out of the box.

- 🛠️ Default setup assumes local clone of library.
- 🔄 Fast feedback loop refactoring:
  - 💥 Local changes MUST have immediate effects.
  - ⚡ No commit, no push, no local package index, no commands.

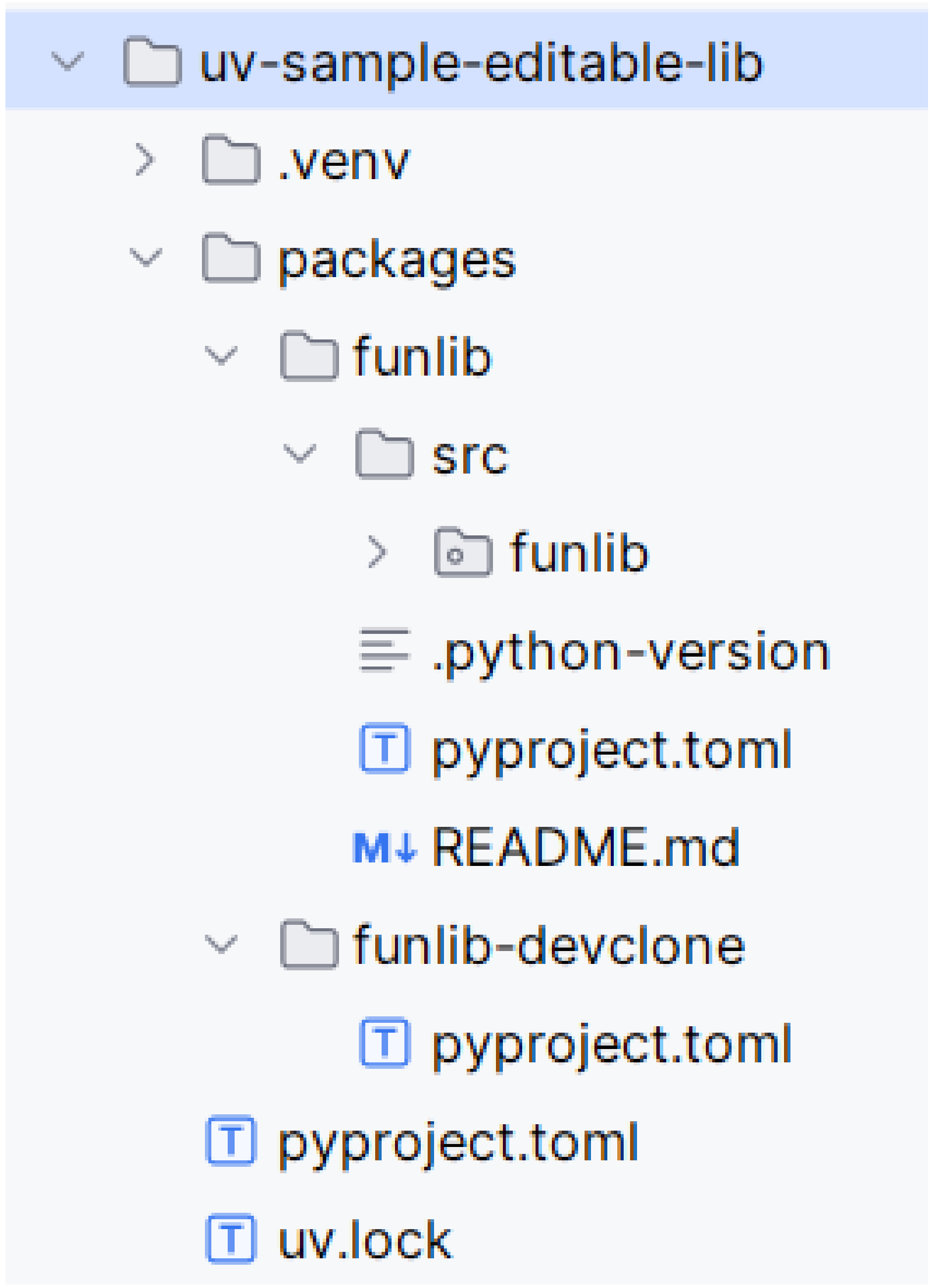
Acceptable

- 🤖 CI yamls and scripts are allowed to do fancy extra steps.
- ✨ As few hacks as possible.
- 💥 Transient breaking apps in local dev env.
  - 🚧 While the shared lib is under heavy refactoring.

Opinions, choices

	says who
no package index	Me
No git submodules	Me
no symlink	Me
We check out all repos in a flat hierarchy.	Me
libs under packages/	Me
src/ layout	Uv
single lock file per repo	Uv
lib git tags follow semantic versions	Me

lib structure



```
uv-sample-editable-lib
  .venv
  packages
    funlib
      src
        funlib
      .python-version
      pyproject.toml
      README.md
    funlib-devclone
      pyproject.toml
  pyproject.toml
  uv.lock
```

lib packages/funlib

```
[project]
name = "funlib"
version = "0.1.1"
description = "shared library for business logic"
readme = "README.md"
authors = [
    { name = "David Horvath", email = "david.h@systematicguy.com" }
]
requires-python = ">=3.13"
dependencies = []

[build-system]
requires = ["uv_build>=0.8.13,<0.9.0"]
build-backend = "uv_build"
```

lib packages/funlib-devclone

```toml
[project]
name = "funlib-devclone"
version = "0.0.0"
description = "Development proxy for funlib package - exposes local funlib source"
requires-python = ">=3.13"
dependencies = []

[build-system]
requires = ["setuptools", "wheel"]
build-backend = "setuptools.build_meta"

[tool.setuptools.packages.find]
where = ["../funlib/src"]
include = ["funlib*"]

[tool.setuptools.package-dir]
railib = "../funlib/src/funlib"
```

lib root workspace

```toml
[project]
name = "uv-sample-editable-lib"
version = "0.1.0"
requires-python = ">=3.13"
dependencies = [
    "funlib",
]

[tool.uv.workspace]
members = [
    "packages/funlib",
    "packages/funlib-devclone",
]

[tool.uv.sources]
funlib = { workspace = true }
```

consumer app – exact and devclone

```
dependencies = []

[build-system]
requires = ["uv_build>=0.8.13,<0.9.0"]
build-backend = "uv_build"

[dependency-groups]
devclone = ["funlib-devclone"]
exact = ["funlib"]

[tool.uv.sources]
funlib = { git = "https://github.com/systematicguy/uv-sample-editable-lib.git",
  subdirectory = "packages/funlib", tag = "0.1.1" }
funlib-devclone = { path = "../uv-sample-editable-lib/packages/funlib-devclone",
  editable = true }
```

consumer app – conflicts and defaults

```toml
[tool.uv]
default-groups = ["devclone"]

# exact and devclone groups are marked as conflicting
# to prevent installing both at the same time
conflicts = [
    [
      { group = "exact" },
      { group = "devclone" },
    ],
]
```

The Hack – CI-only

```
# eliminate editable devclone dependencies
# because they are not checked out in the pipeline
uvx --from toml-cli toml set --toml-path pyproject.toml
  "dependency-groups.devclone" --to-array "[]"

uv lock
uv sync --locked --no-install-project --no-group devclone --group exact

# if we have pytest:
uv run --no-sync pytest
```

Thank you

example on GitHub