pdm-build-locked

sigma67, Frost Ming

Feb 16, 2024

ABOUT

pdm-build-locked is a pdm plugin to enabling reproducible installs of Python CLI tools – no breakage on dependency updates.

It achieves this by adding the pinned packages from PDM's lockfile as additional optional dependency groups to the distribution metadata.

CHAPTER

ONE

WHAT IT DOES

Normally, you would only be able to use your locked dependencies from your lockfile when using pdm in your dev environment.

This plugin enables using the lockfile in a **deployment scenario**.

Thus, your users may install your package as an exact reproduction of your dev environment lockfile by running:

```
pip install mypkg[locked]
```

So for `mypkg` with `optional-dependencies` group `extras` you will end up with the following groups:

- `locked`
- `extras-locked`

To install both, you would run:

```
pip install mypkg[locked, extras-locked]
```

Hint: It achieves this by adding optional-dependencies groups that allow the user to opt-in to installing the dependencies that were pinned in the lockfile:

- `[locked]` - contains all default dependencies as pinned version from lockfile
- for each optional-dependencies group `<group>`:
  - `[<group>-locked]` - contains optional dependencies for group `<group>` as pinned version from lockfile

---

CHAPTER

TWO

WHEN TO USE

To avoid misuse, we recommend deciding whether to use this plugin based on your project type:

- **CLI tool**: If your package is a CLI tool that will be installed in an isolated virtualenv, for example using `pipx`.
- **CLI tool and library**: Recommended. Advise your users to only use the [locked] group when used as an executable (never in `pyproject.toml` dependencies!).
- **Library only**: Do not use.

Danger: The following example is highly discouraged and should never be used, as it will easily lead to dependency conflicts.

`pyproject.toml`

```
dependencies = [
    my-library[locked]==1.1.1,  # this will break your install sooner rather than␣
↪later
    some-other-library
]
```

CHAPTER

THREE

WHICH PLUGIN SHOULD I USE?

3.1 PDM CLI Plugin

If you only care about reproducible installs after publishing your package, you may use the *PDM CLI plugin*.

Compatible package managers:

- **pdm** only

Compatible build backends:

- **any PEP 517 compatible** build backend (`setuptools`, `flit-core`, `pdm-backend`, `` `hatchling `` etc.)

```
pipx install mypkg[locked]
```

3.2 Backend Plugin

If you want to be able to install your package from a local directory or a git repository, you need to use the *Build Backend Plugin*.

Compatible package managers:

- **any PEP 621 compatible** package manager (`poetry`, `flit`, `pdm`, `hatch` etc.)

Compatible build backends:

- **pdm-backend** and **hatchling**

```
pipx install "mypkg[locked] @ git+https://github.com/myorg/mypkg"
```

CHAPTER

FOUR

PDM CLI PLUGIN

The PDM CLI plugin replaces the `pdm build` command with a locked build. Essentially it

- edits your `pyproject.toml` and adds the additional optional-dependency groups with pins from the lockfile to it
- hides the file changes from git to avoid a dirty status on build
- calls the actual `pdm build` command
- restores the original `pyproject.toml`

The result is a wheel that is installable reproducibly, as it contains the exact dependencies you used to build it.

It only requires the user to add `[locked]` on install.

4.1 Installing the plugin

To install the plugin, you need to run `pdm install --plugins`.

Alternatively, you can activate the plugin globally by running `pdm self add pdm-build-locked`.

4.2 Plugin registration

`pyproject.toml`

```
[tool.pdm]
plugins = [
    "pdm-build-locked"
]
```

This registers the plugin with pdm.

4.3 Enabling the plugin

To enable locked builds, set the `locked` entry in `pyproject.toml`:

```
[tool.pdm.build]
package-dir = "src"
locked = true
```

This will enable locked releases in the pipeline, as it also affects a basic `pdm build` call.

Hint: Locally, the following options also work.

- run `pdm build --locked`
- set `PDM_BUILD_LOCKED` env var to `true`

CHAPTER

FIVE

BUILD BACKEND PLUGIN

You can even use this plugin without PDM. This is enabled by build backend hooks.

Currently, both pdm-backend and hatchling are supported.

To set it up, a few configuration steps are required.

5.1 Lockfile configuration

Your lockfile must be configured with the `include_metadata` strategy (`pdm>=2.11`) and include locks for the optional-dependencies groups you want to publish locked.

5.2 pyproject configuration

If you already have a `[project.optional-dependencies]` section, **skip this step**.

Else, add the following to the start of your `pyproject.toml`:

Listing 1: pyproject.toml

```toml
[project]
dynamic = ["optional-dependencies"]
```

5.3 buildsystem configuration

This step depends on the build-system you use and requires you to add the following to your `pyproject.toml`.

5.3.1 pdm-backend

Listing 2: pyproject.toml

```toml
[build-system]
requires = ["pdm-backend", "pdm-build-locked"]
build-backend = "pdm.backend"

[tool.pdm.build]
locked = true
```

5.3.2 hatchling

Listing 3: pyproject.toml

```toml
[build-system]
requires = ["hatchling", "pdm-build-locked"]
build-backend = "hatchling.build"

[tool.hatch.metadata.hooks.build-locked]
```