PDM Modern Python package manager

START HERE

`pdm` **`init`**
Create pyproject.toml

`pdm` **`list --tree`**
Show dependency tree

`pdm` **`add`** *`<pkg>`*
Add a dependency

`pdm` **`use`** *`<python>`*
Pick the interpreter

`pdm` **`add -d`** *`<pkg>`*
Add a dev dependency

`pdm` **`install`**
Lock if needed, then install

`pdm` **`run`** *`<cmd>`*
Run inside the env

`pdm` **`lock`**
Write pdm.lock

pyproject.toml what you want → `pdm lock` → **pdm.lock** exact pins + hashes → `pdm sync` → **Virtualenv** installed packages → `pdm run` → **Your command** runs in the env

CONCEPTS

Commit pyproject.toml and pdm.lock; pdm install reproduces the same set anywhere.

PDM keeps dependencies in standard pyproject.toml (PEP 621) fields, not a private format.

PROJECT SETUP

`pdm` **`init`**
Create pyproject.toml interactively

`pdm` **`init --python`** *`<ver>`*
Init with a Python version or path

`pdm` **`init --lib`**
Init a distributable package

`pdm` **`new`** *`<path>`*
Create a new project folder

`pdm` **`import`** *`<file>`*
Import from Pipfile, Poetry or requirements

`pdm` **`use`** *`<python>`*
Select the project interpreter

`pdm` **`info`**
Show project and environment info

ADD AND REMOVE

`pdm` **`add`** *`<pkg>`*
Add a dependency and install it

ADD AND REMOVE (cont.)

`pdm` **`add`** *`<pkg>==<ver>`*
Add with a version pin

`pdm` **`add -d`** *`<pkg>`*
Add a development dependency

`pdm` **`add -G`** *`<group>`* *`<pkg>`*
Add to a named group

`pdm` **`add -e`** *`<path>`*
Add a local package in editable mode

`pdm` **`add --no-sync`** *`<pkg>`*
Edit pyproject.toml, skip installing

`pdm` **`remove`** *`<pkg>`*
Remove a dependency

`pdm` **`remove -d`** *`<pkg>`*
Remove a dev dependency

LOCK AND INSTALL

`pdm` **`install`**
Install from the lock, locking if stale

`pdm` **`lock`**
Resolve and write pdm.lock

`pdm` **`sync`**
Install exactly what the lock says
Never changes the lock; use it in CI and on servers.

`pdm` **`install --frozen-lockfile`**
Install without touching the lock

LOCK AND INSTALL (cont.)

`pdm` **`install --check`**
Fail if the lock is out of date

`pdm` **`install --prod`**
Skip dev dependencies

`pdm` **`install -G`** *`<group>`*
Include an extra group

`pdm` **`install -G :all`**
Include every group

`pdm` **`install --no-self`**
Skip installing the project itself

`pdm` **`lock --check`**
Check the lock is current and quit

`pdm` **`lock -G :all`**
Lock all declared groups

`pdm` **`sync --clean`**
Remove packages not in the lock

UPDATE

`pdm` **`update`**
Update all packages and re-lock

`pdm` **`update`** *`<pkg>`*
Update one package

`pdm` **`outdated`**
List packages with newer versions

`pdm` **`update --dry-run`**
Preview an update

`pdm` **`update --top`**
Only update direct dependencies

RUN

`pdm` **`run`** *`<cmd>`*
Run a command in the env

`pdm` **`run python`** *`<file>`*
Run a script with the env's Python

`pdm` **`run`** *`<script>`* **`[args]`**
Run a [tool.pdm.scripts] entry

`pdm` **`run -l`**
List defined scripts

`pdm` **`run --env-file`** *`<file>`*
Load a dotenv file for the run

INSPECT

`pdm` **`list`**
List installed packages

`pdm` **`list --tree`**
Show the dependency tree

`pdm` **`list --tree -r`**
Reverse tree: who needs this?

`pdm` **`list --freeze`**
Output in pip freeze format

`pdm` **`show`** *`<pkg>`*
Show package metadata

VIRTUALENVS AND PYTHON

`pdm` **`venv create`**
Create a virtualenv for the project

`pdm` **`venv list`**
List the project's virtualenvs

`pdm` **`venv activate`**
Print the activate command

VIRTUALENVS AND PYTHON (cont.)

`pdm` **`venv remove`** *`<name>`*
Delete a virtualenv

`pdm` **`python install`** *`<ver>`*
Download a Python interpreter

BUILD AND PUBLISH

`pdm` **`build`**
Build sdist and wheel into dist/

`pdm` **`publish`**
Build and upload to PyPI

`pdm` **`publish -r`** *`<repo>`*
Upload to another repository

`pdm` **`publish --no-build`**
Upload existing files in dist/

`pdm` **`export -o requirements.txt`**
Export the lock as requirements

`pdm` **`export --no-hashes`**
Export without hashes

CONFIGURATION

`pdm` **`config`**
Show all settings

`pdm` **`config`** *`<key>`* *`<value>`*
Set a user-level setting

`pdm` **`config -l`** *`<key>`* *`<value>`*
Set it for this project only

`pdm` **`self update`**
Update PDM itself

PYPROJECT.TOML

Project and groups

```
[project]
name = "app"
requires-python = "⩾3.10"
dependencies =
["requests⩾2.31"]

[project.optional-dependencies]
web = ["flask"]

[dependency-groups]
dev = ["pytest"]

[tool.pdm]
distribution = false
```

Script shortcuts

```
[tool.pdm.scripts]
start = "flask run -p 54321"
test = {cmd = "pytest -q", help
= "Run tests"}
lint = {shell = "ruff check . |
head"}
main = {call = "app.cli:main"}
all = {composite = ["lint",
"test"]}
test.env = {APP_ENV = "test"}
```

Extra arguments are appended to a script; put {args} in the command to choose where they go.

RECIPES

New project

```
mkdir app && cd app
pdm init
pdm add requests
pdm add -d pytest
pdm run pytest
```