PDM Modern Python package manager

CONCEPTS

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

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

PROJECT SETUP

pdm init
Create pyproject.toml interactively

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

pdm use *<python>*
Select the project interpreter

ADD AND REMOVE

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

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 remove *<pkg>*
Remove a 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

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

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Fail if the lock is out of date

pdm install --prod
Skip dev dependencies

LOCK AND INSTALL (cont.)

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

UPDATE

pdm update
Update all packages and re-lock

pdm update *<pkg>*
Update one package

pdm outdated
List packages with newer versions

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

INSPECT

pdm list
List installed packages

pdm list --tree
Show the dependency tree

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

VIRTUALENVS AND PYTHON

pdm venv create
Create a virtualenv for the project

pdm venv activate
Print the activate command

BUILD AND PUBLISH

pdm build
Build sdist and wheel into dist/

pdm publish
Build and upload to PyPI

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

CONFIGURATION

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