Poetry Python packaging and dependencies

CONCEPTS

Commit poetry.lock for applications so everyone installs the same versions. poetry install needs a lock that matches pyproject.toml; run poetry lock after editing dependencies by hand.

PROJECTS

poetry new *<name>*
Create a project in a new directory

poetry init
Create pyproject.toml interactively

DEPENDENCIES

poetry add *<pkg>*...
Add dependencies and install them

poetry add *<pkg>*@*<constraint>*
Add with a version constraint

poetry add --group *<name>* *<pkg>*
Add to a dependency group

poetry add --dev *<pkg>*
Shortcut for the dev group

poetry remove *<pkg>*...
Remove dependencies

INSTALL AND LOCK

poetry install
Install the project and its locked dependencies

poetry install --no-root
Skip installing the project itself

poetry install --without *<group>*
Exclude a group

poetry sync
Make the env match the lock exactly

poetry lock
Refresh poetry.lock for pyproject.toml

poetry update
Upgrade all packages within constraints

INSTALL AND LOCK (cont.)

poetry update *<pkg>*...
Upgrade only these packages

RUNNING AND ENVIRONMENTS

poetry run *<cmd>*
Run a command in the project env

poetry run python
Python REPL with project deps

poetry shell
Spawn a shell in the env (plugin)

poetry env info
Show env details

poetry env use *<python>*
Pick or create the env for a Python

INSPECTING

poetry show
List installed packages

poetry show --tree
Dependencies as a tree

poetry show --outdated
Only packages with newer versions

BUILD AND PUBLISH

poetry build
Build sdist and wheel into dist/

poetry publish
Upload dist/ to PyPI

CONFIG AND MAINTENANCE

poetry config --list
Show current settings

poetry config *<key>* *<value>*
Set a setting

poetry config virtualenvs.in-project true
Create .venv inside the project