Hatch Python project manager

CONCEPTS

Environments are created on first use and rebuilt when their dependencies change in pyproject.toml.

pyproject.toml envs, scripts, version
↓ `hatch run`
Environment isolated virtualenv
↓ `hatch build`
dist/ sdist and wheel
↓ `hatch publish`
Index PyPI
From source to release

Config lives under [tool.hatch.envs.<name>] in pyproject.toml, or in envs.<name> of a separate hatch.toml.

PROJECTS

`hatch new <name>`
Create a project skeleton

`hatch new --cli <name>`
Include a command line entry point

`hatch new --init`
Convert an existing project

`hatch project metadata`
Show the resolved metadata

ENVIRONMENTS

`hatch shell`
Enter the default env shell

`hatch shell <env>`
Enter a named env shell

`hatch env create`
Create the default env

`hatch env create <env>`
Create a named env

`hatch env show`
List the configured envs

`hatch env find <env>`
Print an env's path

`hatch env remove <env>`
Delete an env

`hatch env prune`
Delete all envs

RUNNING COMMANDS

`hatch run <cmd>`
Run a command in the default env

`hatch run <env>:<cmd>`
Run a command in a named env

`hatch run <script>`
Run a script defined in config

`hatch -e <env> run <cmd>`
Pick the env with a flag

`hatch run +py=3.12 <env>:<cmd>`
Select matrix envs by variable

TESTING AND QUALITY

`hatch test`
Run tests in the matrix

`hatch test -c`
Measure coverage

`hatch test -a`
Test every matrix env

`hatch test -py 3.12`
Test one Python version

`hatch test -p`
Run tests in parallel

`hatch test <path>::<test>`
Pass arguments to pytest

`hatch fmt`
Format and fix lint errors

`hatch fmt --check`
Check only, change nothing

`hatch fmt -l`
Run the linter only

`hatch fmt -f`
Run the formatter only

VERSIONING

`hatch version`
Show the project version

`hatch version <x.y.z>`
Set an exact version

`hatch version patch`
Bump the patch number

`hatch version minor`
Bump the minor number

`hatch version major`
Bump the major number

BUILD AND PUBLISH

`hatch build`
Build sdist and wheel into dist/

`hatch build -t wheel`
Build one target only

`hatch build -c`
Clean old artifacts first

`hatch clean`
Remove build artifacts

`hatch publish`
Upload artifacts to the index

`hatch publish -r test`
Upload to the test index

`hatch publish -u __token__ -a <token>`
Authenticate with a token

DEPENDENCIES AND PYTHON

`hatch dep show requirements`
List env dependencies

`hatch dep show table`
List dependencies as a table

`hatch python show`
Show available Python builds

`hatch python install <version>`
Install a managed Python

CONFIG IN PYPROJECT.TOML

`[tool.hatch.version]`
Where the version comes from

`[tool.hatch.envs.<name>]`
Env deps and scripts

`[[tool.hatch.envs.<name>.matrix]]`
Env variants

Env scripts can call each other by name and use {args} to forward extra arguments.

RECIPES

Env with scripts and a matrix

```
# pyproject.toml
[tool.hatch.envs.test]
dependencies = ["pytest"]
[tool.hatch.envs.test.scripts]
run = "pytest {args}"
[[tool.hatch.envs.test.matrix]]
python = ["3.11", "3.12"]
```