Hatch Python project manager

START HERE

`hatch new <name>`
Create a project

`hatch version patch`
Bump the version

`hatch shell`
Enter the default env

`hatch build`
Make sdist and wheel

`hatch run <cmd>`
Run in an env

`hatch run test:<cmd>`
Run in the test env

`hatch test`
Run the test matrix

`hatch fmt`
Format and lint

pyproject.toml envs, scripts, version → `hatch run` → **Environment** isolated virtualenv → `hatch build` → **dist/** sdist and wheel → `hatch publish` → **Index** PyPI

From source to release

CONCEPTS

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

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
Reads setup.py or requirements files and writes a pyproject.toml.

`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

ENVIRONMENTS (cont.)

`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_ENV does the same from the environment.

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

`hatch run -py=3.9 <env>:<cmd>`
Exclude 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

`hatch fmt --sync`
Sync the default Ruff config

`hatch check`
Run code, format and type checks

VERSIONING

`hatch version`
Show the project version

VERSIONING (cont.)

`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 lock`
Write a lockfile for the env
Alias of hatch dep lock; locks the env chosen with -e.

`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.

CONFIG IN PYPROJECT.TOML (cont.)

Start a script command with - to ignore its exit code and carry on with the next one.

`hatch config show`
Print the user config

`hatch self update`
Upgrade Hatch itself

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"]
```

Version from a file

```
# pyproject.toml
[project]
dynamic = ["version"]
[tool.hatch.version]
path = "src/demo/__about__.py"
```

Release a new version

```
hatch version minor
hatch build -c
hatch publish -r test
hatch publish
```