tox Test automation

START HERE

`tox quickstart`
Create a config

`tox run -e <e1>,<e2>`
Run several envs

`tox`
Run default envs

`tox p`
Run envs in parallel

`tox -e <env>`
Run one env

`tox -r -e <env>`
Rebuild the env

`tox -e <env> -- <args>`
Pass args to commands

`tox l`
List environments

Package build sdist/wheel → **Env** virtualenv in .tox/ → **Install** deps + package → **Run** exit code = result

CONCEPTS

tox builds a fresh virtualenv per environment, installs your package and deps, then runs the commands. Same steps locally and in CI.

Environment names are factors split on dashes: py313-django has factors py313 and django, which deps can key on.

Config lives in tox.ini, tox.toml or pyproject.toml ([tool.tox]). tox.toml and native TOML need tox 4.22+.

RUNNING

`tox`
Run the envs in env_list

`tox run`
Same, explicit subcommand (alias r)

`tox -e <env>`
Run a single environment

`tox -e <e1>,<e2>`
Run several named environments

`tox -e <env> -- <args>`
Forward args to {posargs}

`tox -e ALL`
Run every defined environment

`tox -r -e <env>`
Recreate the env from scratch

`tox -f <factor>`
Run envs containing a factor

Several names after one -f must all match; repeat -f for OR.

`tox -m <label>`
Run envs under a label

`tox --skip-env <regex>`
Exclude envs matching a regex

`tox -s`
Skip missing interpreters

`tox --fail-fast`
Stop after the first failing env

`tox -n -e <env>`
Build the env, skip the commands

`tox -i`
Disable output capture

`tox -v`
More verbose (repeat for more)

`tox -c <file>`
Use another config file or folder

`tox -x <env>.<key>=<val>`
Override one config value

RUNNING (cont.)

`tox --installpkg <file>`
Test a prebuilt sdist or wheel

PARALLEL

`tox p`
Run envs in parallel (alias of run-parallel)

`tox p -p <n>`
Limit to n parallel envs

Default is auto = CPU count; 0 disables.

`tox p -o`
Stream output live while running

`tox p -e <e1>,<e2>`
Parallel run of chosen envs

INSPECT

`tox l`
List envs with descriptions

`tox l -d`
List only default envs

`tox c -e <env>`
Show the resolved config of an env

`tox c -e <env> -k <key>`
Show only chosen keys

`tox c -e ALL`
Resolved config of every env

`tox c --core`
Show core (non-env) settings

`tox c --format toml`
Output as ini, json or toml

`tox depends`
Show env run order

`tox --version`
Show tox and plugin versions

OTHER COMMANDS

`tox quickstart`
Interactively create a config (alias q)

`tox exec -e <env> -- <cmd>`
Run any command inside an env

`tox devenv <dir>`
Create a dev virtualenv from an env

-e picks the env; default is py.

CONFIG FILE

`[tox]`
Core section in tox.ini

`env_list = <e1>, <e2>`
Default environments

CONFIG FILE (cont.)

`requires = tox≥4.2`
Needed tox and plugins (auto-provisioned)

`labels = <name>=<e1>,<e2>`
Name a group of envs

`[testenv]`
Base section for all envs

`[testenv:<env>]`
Settings for one env

`description = <text>`
Shown by tox l

`deps = <pkg>`
Packages to install

`commands = <cmd> {posargs}`
Commands to run, one per line

`commands_pre = <cmd>`
Run before commands

`commands_post = <cmd>`
Run after commands, even on failure

`extras = <name>`
Install optional dependencies

`pass_env = <VAR>`
Let env vars through to the env

`set_env = <VAR>=<val>`
Set env vars

`change_dir = <dir>`
Working dir for commands

`allowlist_externals = <cmd>`
Allow commands from outside the env

Without this tox refuses system commands such as make.

`base_python = python3.13`
Pick the interpreter

`skip_install = true`
Do not install your package

`package = wheel`
sdist, wheel, editable or skip

`dependency_groups = <group>`
Install PEP 735 groups (tox 4.22+)

`depends = <env>`
Run only after another env (parallel)

`ignore_outcome = true`
Report failure without failing the run

`ignore_errors = true`
Continue after a failing command

SUBSTITUTION & FACTORS

`{posargs}`
Args after -- on the command line

`{posargs:<default>}`
Args, or a default if none given

`{env_name}`
Name of the current env

`{tox_root}`
Directory of the config file

`{env_python}`
Interpreter of the env

`{env:<VAR>:<default>}`
Read an environment variable

`<factor>: <value>`
Line applies only if the env has the factor

`py3{12,13}-{a,b}`
Expand into every combination

RECIPES

Minimal tox.ini

```
[tox]
env_list = lint, py313

[testenv]
deps = pytest
commands = pytest {posargs}

[testenv:lint]
skip_install = true
deps = ruff
commands = ruff check .
```

Matrix with factors

```
[tox]
env_list = py3{12,13}-
{django,flask}


[testenv]
deps =
  pytest
  django: django
  flask: flask
commands = pytest
```

Native tox.toml

```
env_list = ["lint", "3.13"]

[env_run_base]
dependency_groups = ["test"]
commands = [["pytest", { replace
= "posargs", extend = true }]]

[env.lint]
skip_install = true
deps = ["ruff"]
commands = [["ruff", "check",
"."]]
```

One test, fresh env

```
tox -r -e py313 -- -x -k <name>
```

Same matrix in CI

```
# one job per Python
tox -e py313
# or all at once on a runner
tox p
```