mise Tool versions, env and tasks

START HERE

`mise use node@24`
Install and pin a tool

`mise install`
Install what the config lists

`mise ls`
Installed and active tools

`mise x node@24 -- node -v`
Run once, no config

`mise set <KEY>=<value>`
Add an env variable

`mise run <task>`
Run a project task

`mise upgrade`
Update tools

`mise doctor`
Diagnose problems

mise.toml [tools] [env] [tasks] → `mise install` → **Installed tools** in the mise data dir → `activate or shims` → **PATH + env** per directory → `mise run` → **Tasks** run with these tools

What mise does in a project

CONCEPTS

mise manages tool versions, env variables and tasks from one mise.toml per project. Commit it so everyone gets the same setup.

Activation (mise activate in your shell rc) updates PATH on every prompt. Shims suit IDEs and scripts that skip rc files; mise exec and mise run need neither.

Config with [env], hooks or templates loads only once trusted. mise use, set, install, exec and run trust it for you; for a cloned repo read it, then mise trust.

Nearer mise.toml files override parents, and ~/.config/mise/config.toml is the global layer. mise.local.toml is for personal overrides, mise.<env>.toml for -E <env>.

INSTALL AND ACTIVATE

`curl -fsSL https://mise.run | sh`
Install mise to ~/.local/bin

`eval "$(mise activate bash)"`
Activate in bash (put in ~/.bashrc)

`eval "$(mise activate zsh)"`
Activate in zsh (put in ~/.zshrc)

`mise activate fish | source`
Activate in fish (config.fish)

`mise completion bash`
Print shell completions

`mise reshim`
Rebuild shims for installed tools

`mise self-update`
Update mise itself

`mise version`
Show the mise version

TOOLS

`mise use <tool>@<version>`
Install and add to mise.toml

`mise use -g <tool>@<version>`
Same, in the global config

TOOLS (cont.)

`mise use --pin <tool>@<version>`
Record the exact resolved version

`mise use --remove <tool>`
Drop a tool from the config

`mise install`
Install every tool the config lists

`mise install <tool>@<version>`
Install without touching config

`mise ls`
List installed and active tools

`mise ls --missing`
Requested but not installed

`mise ls-remote <tool>`
Versions available to install

`mise latest <tool>`
Latest version matching a request

`mise outdated`
Show tools with newer versions

`mise upgrade`
Upgrade within configured ranges

`mise upgrade --bump`
Upgrade and rewrite the versions

`mise unuse <tool>`
Remove from config and prune

`mise prune`
Delete versions nothing uses

`mise uninstall <tool>@<version>`
Remove an installed version

`mise search <text>`
Search for available tools

`mise registry`
List tool shorthand names

`mise where <tool>`
Install dir of a tool version

`mise which <bin>`
Path of the executable mise resolves

RUNNING WITH TOOLS

`mise exec <tool>@<version> -- <cmd>`
Run a command with that tool

`mise x -- <cmd>`
Run with the project's tools and env

`mise exec <tool> -c "<a> && <b>"`
Run a shell string (pipes, &&)

RUNNING WITH TOOLS (cont.)

`mise shell <tool>@<version>`
Use a version in this shell only

`mise shell -u <tool>`
Undo a mise shell override

`mise en`
Subshell with tools and env loaded

`mise exec --deny-net -- <cmd>`
Run with network blocked

`mise bin-paths`
List bin dirs of active tools

ENVIRONMENT

`mise set <KEY>=<value>`
Set a variable in mise.toml

`mise set`
List variables and their sources

`mise set -E <env> <KEY>=<value>`
Write to mise.<env>.toml

`mise unset <KEY>`
Remove a variable from mise.toml

`mise env`
Print the env mise would set

`mise env --dotenv > .env`
Export as a dotenv file

`eval "$(mise env -s bash)"`
Load the env into this shell once

`mise -E <env> <cmd>`
Use mise.<env>.toml for this run

TASKS

`mise run <task>`
Run a task and its dependencies

`mise <task>`
Shorthand for mise run <task>

`mise tasks`
List tasks

`mise run <a> ::: <b>`
Run several tasks in parallel

`mise run 'test:*'`
Run tasks matching a pattern

`mise run <task> <args>`
Pass arguments to a task

`mise run -n <task>`
Show run order, execute nothing

`mise tasks add <task> -- <cmd>`
Create a task

TASKS (cont.)

`mise watch <task>`
Rerun a task when files change

Task fields in [tasks.<name>]

Field	Effect
`run`	Command, or array run in order
`description`	Shown in mise tasks
`depends`	Tasks that must succeed first
`alias`	Short name for mise run
`dir`	Working directory
`env`	Variables for this task only
`sources / outputs`	Skip when outputs are current
`hide`	Hide from the list, still runnable
`confirm`	Ask before running

CONFIG AND TRUST

`mise trust`
Trust the nearest untrusted config

`mise trust --show`
Trust status of configs here and above

`mise config`
List the config files in use

`mise settings`
Show or change mise settings

`mise doctor`
Check the setup for problems

RECIPES

Pin a toolchain for a repo

```
cd my-project
mise use node@24 python@3.13
git add mise.toml
mise ls
```

mise.toml with env and a task

```
[tools]
node = "24"

[env]
NODE_ENV = "development"

[tasks.test]
description = "Run tests"
depends = ["build"]
run = "npm test"
```

CI without activation

```
curl -fsSL https://mise.run | sh
~/.local/bin/mise install
~/.local/bin/mise run test
```

Try a version without committing

```
mise x node@22 -- node -v
mise shell node@22
mise shell -u node
```