Lefthook Fast git hooks manager

START HERE

`lefthook install`
Install the git hooks

`lefthook add <hook>`
Add a hook

`lefthook run pre-commit`
Run a hook by hand

`lefthook uninstall`
Remove the hooks

`lefthook run pre-commit --all-files`
Run on every file

`lefthook validate`
Check the config

`lefthook dump`
Show merged config

`LEFTHOOK=0 git commit`
Skip hooks once

git commit git event → `runs .git/hooks` → **lefthook run** reads lefthook.yml → `filters files` → **Jobs** glob, skip, root → `exit code` → **Commit** allowed or aborted

CONCEPTS

Hooks are read from lefthook.yml on every git event, so editing the config needs no reinstall.

Run lefthook install after cloning; the npm package does it for you in a postinstall script.

Config order: lefthook.yml, extends, remotes, then lefthook-local.yml (wins).

INSTALL & CLI

`brew install lefthook`
Install with Homebrew

`npm install lefthook --save-dev`
Install as a dev dependency

`go install github.com/evilmartians/lefthook/v2@latest`
Install with Go

`gem install lefthook`
Install as a gem

`pipx install lefthook`
Install with pipx

`lefthook install`
Install hooks, create lefthook.yml if missing

`lefthook install <hook> ...`
Install only some hooks

INSTALL & CLI (cont.)

`lefthook install --force`
Install even if core.hooksPath is set

`lefthook uninstall`
Remove hooks lefthook installed

`lefthook add <hook> --dirs`
Add a hook and its script directory

`lefthook validate`
Check the config for errors

`lefthook dump`
Print the merged configuration

RUNNING

`lefthook run <hook>`
Run a hook's jobs now

`lefthook run <hook> --all-files`
Treat all tracked files as input

`lefthook run <hook> --file <path>`
Run on chosen files only
Repeat the flag for several files. Ignored if --all-files is also given.

`lefthook run <hook> --job <name>`
Run only the named job

`lefthook run <hook> --tag <tag>`
Run only jobs with a tag

`LEFTHOOK_VERBOSE=1 lefthook run <hook>`
Print verbose output

RUNNING (cont.)

`LEFTHOOK_EXCLUDE=<tag>,<name>`
Exclude tags or commands for one run

`git commit --no-verify`
Bypass pre-commit and commit-msg hooks

`LEFTHOOK=0 git <cmd>`
Disable lefthook for one command

HOOK & JOB OPTIONS

`pre-commit:`
Hook name as top-level key
Any git hook name works, plus custom names you run with lefthook run.

`jobs: [ {name, run} ]`
Ordered list of tasks in a hook

`commands: <name>: run:`
Older map form of jobs

`run: <shell command>`
Command to execute (via sh)

`parallel: true`
Run a hook's jobs concurrently

`piped: true`
Stop at the first failing job
Cannot be combined with parallel: true.

`group: {parallel, jobs}`
Nest jobs with their own flow

HOOK & JOB OPTIONS (cont.)

`script: <file> runner: bash`
Run a script from .lefthook/<hook>/

`root: <dir>`
Run the job from a subdirectory

`env: {KEY: value}`
Set environment variables for a job

`tags: [<tag>]`
Label a job for filtering

FILES & PLACEHOLDERS

`{staged_files}`
Files staged for this commit

`{push_files}`
Commits not yet pushed, as files

`{all_files}`
Every file tracked by git

`{files}`
Output of the job's files command

`{1} / {2}`
Arguments git passed to the hook

`glob: "*.{js,ts}"`
Only run for matching files
If a glob matches no staged files, the job is skipped.

FILES & PLACEHOLDERS (cont.)

`glob: ["*.ts", "*.js"]`
List of globs

`exclude: [<glob>]`
Leave matching files out

`files: git diff --name-only main`
Custom command listing files

`stage_fixed: true`
git add files after the job (pre-commit)

SKIPPING & CONDITIONS

`skip: true`
Disable a job or hook

`skip: [merge, rebase]`
Skip in merge or rebase state

`skip: merge-commit`
Skip when HEAD is a merge commit

`skip: [{ref: main}]`
Skip on a branch (globs ok)

`skip: [{run: <cmd>}]`
Skip if the command succeeds

`only: [{ref: dev/*}]`
Run only when the condition holds
skip takes precedence over only.

`lefthook-local.yml`
Personal overrides, not committed

CONFIG FILE

`extends: [<file>]`
Merge other YAML files (globs ok)

`remotes: [{git_url, ref, configs}]`
Share config from another repo

RECIPES

Lint staged files and restage

```
pre-commit:
  parallel: true
  jobs:
    - name: eslint
      glob: "*.{js,ts}"
      run: npx eslint --fix
{staged_files}
      stage_fixed: true
    - name: tests
      run: npm test
```

Per-folder jobs in a monorepo

```
pre-commit:
  jobs:
    - name: web
      root: web/
      glob: "web/**/*.ts"
      run: npx tsc --noEmit
    - name: api
      root: api/
      glob: "api/**/*.go"
      run: go vet ./...
```

Check commit messages

```
commit-msg:
  jobs:
    - name: commitlint
      run: npx commitlint --
edit {1}
```