pre-commit Git hook manager

START HERE

`pip install pre-commit`
Install the tool

`pre-commit sample-config > .pre-commit-config.yaml`
Create a config

`pre-commit install`
Hook into git commit

`pre-commit run --all-files`
Check every file

`pre-commit run <hook-id>`
Run one hook

`pre-commit autoupdate`
Bump hook versions

`SKIP=<hook-id> git commit`
Skip a hook once

`git commit --no-verify`
Skip all hooks

git commit staged files → **hook script** → **pre-commit** reads config → **filters files** → **Hooks** cached envs → **exit code** → **Commit** or abort

CONCEPTS

Hooks only see staged files on commit. A fresh config does nothing until you run --all-files once.

A hook that edits files counts as a failure: review the changes, git add them and commit again.

Hook environments are built once per repo and rev and cached, so the first run is slow.

INSTALL & SETUP

`pip install pre-commit`
Install with pip

`pre-commit install`
Install the git pre-commit hook
Run once per clone; the hook lives in .git/hooks.

`pre-commit install --install-hooks`
Install hook and build environments

`pre-commit install -t <hook-type>`
Install another hook type
E.g. pre-push or commit-msg.

`pre-commit install-hooks`
Build environments only

`pre-commit uninstall`
Remove the git hook

`pre-commit sample-config`
Print a starter config

`pre-commit init-templatedir <dir>`
Hooks for every new clone
Pair with git config init.templateDir.

`pre-commit install -f`
Overwrite an existing hook

RUNNING

`pre-commit run`
Run hooks on staged files

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

`pre-commit run <hook-id>`
Run a single hook

`pre-commit run <hook-id> --all-files`
One hook on all files

`pre-commit run --files <file>...`
Run on named files

RUNNING (cont.)

`pre-commit run --from-ref <ref> --to-ref <ref>`
Run on files changed between refs
Typical CI use: --from-ref origin/main --to-ref HEAD.

`pre-commit run --show-diff-on-failure`
Print the diff after failures

`pre-commit run --hook-stage <stage>`
Run hooks of one stage
Use manual for hooks that never run automatically.

`pre-commit run -v`
Show output of passing hooks

`pre-commit run --fail-fast`
Stop at the first failure

SKIPPING

`SKIP=<id>,<id> git commit`
Skip hooks by id
Ids are comma-separated; other hooks still run.

`git commit --no-verify`
Skip all git hooks
Also skips non-pre-commit hooks.

`SKIP=<id> pre-commit run -a`
Skip a hook in a manual run

Prefer SKIP over --no-verify: the other checks still protect you.

UPDATING & CACHE

`pre-commit autoupdate`
Bump every rev to the latest tag

`pre-commit autoupdate --repo <url>`
Update one repo only

`pre-commit autoupdate --freeze`
Pin commit SHAs, keep tag as comment

`pre-commit try-repo <repo> <hook-id>`
Test a hook repo without adding it
Uses the repo's HEAD unless --ref is given.

`pre-commit clean`
Delete the whole cache

`pre-commit validate-config <file>`
Check a config file

CONFIG FILE

Top-level keys

Key	Meaning
`repos`	list of hook sources (required)
`default_stages`	stages hooks use by default
`default_install_hook_types`	what install sets up
`default_language_version`	e.g. python: python3.12
`files / exclude`	global regex filters
`fail_fast`	stop after first failing hook
`minimum_pre_commit_version`	refuse older versions

Repo entry

Key	Meaning
`repo`	git URL, local or meta
`rev`	tag or SHA to use, not a branch
`hooks`	list of hook mappings

Hook keys

Key	Meaning
`id`	which hook (required)
`args`	extra arguments before filenames
`files / exclude`	regex on file paths
`types / types_or`	filter by file type
`stages`	git stages it runs on
`additional_dependencies`	extra packages for the env
`always_run`	run even without matching files
`pass_filenames`	local hooks: pass files or not
`language_version`	override the runtime version

files and exclude are regexes matched against the path, not globs. Use ^docs/ rather than docs/*.

CONFIG FILE (cont.)

Always pin rev to a tag or SHA. A moving branch name is rejected.

Common stages

Stage	Fires on
`pre-commit`	git commit (default)
`commit-msg`	after message is written
`pre-push`	git push
`pre-merge-commit`	before a merge commit
`manual`	only if you ask for it

LOCAL & META HOOKS

`repo: local`
Hooks defined in your own repo
Needs id, name, entry, language and files or types.

`language: unsupported`
Run a tool already on the PATH
Older configs say system; same meaning.

CI

`pre-commit run --all-files --show-diff-on-failure`
Typical CI step

`pre-commit run --from-ref origin/main --to-ref HEAD`
Check only the changed files

RECIPES

Minimal config

```
# .pre-commit-config.yaml
repos:
  - repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v5.0.0
    hooks:
      - id: end-of-file-fixer
      - id: trailing-whitespace
      - id: check-yaml
```

Add a local hook

```
  - repo: local
    hooks:
      - id: pytest
        name: pytest
        entry: pytest -q
        language: unsupported
        pass_filenames: false
        stages: [pre-push]
```

Adopt in an existing repo

```
pre-commit install
pre-commit run --all-files
git add -u
git commit -m "Apply pre-commit fixes"
```