Vitest Vite-native test runner

CONCEPTS

Plain vitest watches when run in a terminal and runs once when in CI or without a TTY. Use vitest run for a guaranteed single run.

Filter path, -t, --changed
↓ select
Run workers, pool, env
↓ report
Report --reporter, --coverage
↓ iterate
Watch a / f / u / p / t keys

CLI flags override the config file (vitest.config.ts or the test block in vite.config.ts); keep settings there and use flags for one-off runs.

RUN TESTS

`npx vitest`
Start watch mode (or one run in CI)

`npx vitest run`
Run once, no watch

`npx vitest --run`
Same as run, for any command

`npx vitest <filter> <filter>`
Run files whose path contains a filter

`npx vitest <file>:<line>`
Run the test at a line number

`npx vitest -t "<pattern>"`
Run tests whose full name matches a regexp

`npx vitest run <file> -t "<pattern>"`
Combine file and name filters

`npx vitest related <file> --run`
Run tests that import the given files

`npx vitest run --changed`
Run tests affected by uncommitted changes

`npx vitest run --changed HEAD~1`
Tests affected since a commit or branch

RUN TESTS (cont.)

`npx vitest run --passWithNoTests`
Do not fail when nothing matches

WATCH MODE KEYS

`a / Enter`
Rerun all tests

`f`
Rerun only failed tests

`r`
Rerun the current pattern

`u`
Update failing snapshots

`p`
Filter by file name

`t`
Filter by test name regexp

`h`
Show the key help

`q`
Quit

OUTPUT

`npx vitest run --reporter=verbose`
One line per test

`npx vitest run --reporter=dot`
Compact dots

`npx vitest run --reporter=json --outputFile=<file>`
Write JSON results to a file

`npx vitest run --reporter=junit --outputFile=<file>`
JUnit XML for CI

`npx vitest list`
Print the collected tests without running

`npx vitest --ui`
Open the browser dashboard

COVERAGE

`npx vitest run --coverage`
Run with coverage report

COVERAGE (cont.)

`npx vitest run --coverage --coverage.reporter=html`
Pick report formats

`npx vitest run --coverage --coverage.include="src/**"`
Count uncovered files too

`npx vitest run --coverage --coverage.thresholds.lines=80`
Fail below 80% lines

SNAPSHOTS AND FAILURES

`npx vitest run -u`
Rewrite all failing snapshots

`npx vitest run --bail 1`
Stop after the first failure

`npx vitest run --retry 2`
Retry failed tests twice

SPEED AND CI

`npx vitest run --shard 1/3`
Run one third of the suite

`npx vitest run --no-file-parallelism`
Run files one after another

`npx vitest run --project <name>`
Run one workspace project

ENVIRONMENT AND DEBUGGING

`npx vitest run --environment=jsdom`
Test in a DOM environment

`npx vitest --inspectBrk --no-file-parallelism`
Debug with the Node inspector

RECIPES

CI run with reports and coverage

```
npx vitest run --coverage \
  --reporter=default --
reporter=junit \
  --outputFile.junit=reports/
junit.xml
```

Pre-commit: tests for staged files

```
git diff --name-only --cached \
  | xargs npx vitest related --run
```