golangci-lint Go linters runner

START HERE

`golangci-lint run`
Lint the module

`golangci-lint run -E <linter>`
Add a linter

`golangci-lint run --fix`
Lint and auto-fix

`golangci-lint run --new-from-rev main`
New issues only

`golangci-lint fmt`
Format the code

`//nolint:<linter>`
Silence one line

`golangci-lint linters`
What is enabled?

`golangci-lint config verify`
Check .golangci.yml

Packages ./... → load → **Linters** default + enable → report → **Filters** exclusions, nolint, new-from → print → **Output** text, json, sarif...

CONCEPTS

One binary runs many linters in parallel over a shared parse, with a cache. Exit code 1 means issues were found.

Coming from v1? Config needs version: "2", exclude rules moved to linters.exclusions, formatters (gofmt, goimports) have their own section. Run golangci-lint migrate.

INSTALL AND INFO

`curl -sSfL https://golangci-lint.run/install.sh | sh -s -- -b $(go env GOPATH)/bin v<version>`
Install a pinned release binary
go install is not guaranteed to work; prefer the release binary.

`docker run --rm -v $(pwd):/app -w /app golangci/golangci-lint golangci-lint run`
Run from the official image

`golangci-lint --version`
Show version and Go it was built with

`golangci-lint help linters`
Describe every available linter

RUN

`golangci-lint run`
Lint all packages below here
Same as run ./... ; it lints packages, not single files.

`golangci-lint run <dir>/...`
Lint one subtree

`golangci-lint run --fix`
Apply fixes linters can make
Only linters marked [auto-fix] in the linters list rewrite code.

`golangci-lint run --timeout 5m`
Abort after a time limit
No timeout by default.

`golangci-lint run --tests=false`
Skip *_test.go files

`golangci-lint run --build-tags <tag>`
Lint with build tags set

`golangci-lint run -c <file>`
Use a specific config file

`golangci-lint run --no-config`
Ignore any config file

RUN (cont.)

`golangci-lint run -v`
Show config used, linters and timings

`golangci-lint run --issues-exit-code 0`
Report but never fail

`golangci-lint cache clean`
Clear the analysis cache
Try this first when results look stale.

PICK LINTERS

`golangci-lint linters`
List enabled and disabled linters
Tags show [fast] and [auto-fix] linters.

`golangci-lint run -E <linter>,<linter>`
Enable extra linters

`golangci-lint run -D <linter>`
Disable a linter

`golangci-lint run --enable-only <linter>`
Run just this linter
Overrides the linters section of the config.

`golangci-lint run --default all`
Start from every linter
Noisy: combine with -D or a config. Sets: standard, all, none, fast.

`golangci-lint run --default none -E <linter>`
Start empty, add only these

`golangci-lint run --fast-only`
Keep only fast linters
Good for editor save hooks.

The standard set (default)

Linter	Catches
`errcheck`	unchecked errors
`govet`	go vet checks
`staticcheck`	bugs, simplifications, style
`unused`	unused code
`ineffassign`	assignments never read

ONLY NEW ISSUES

`golangci-lint run --new-from-rev <rev>`
Issues introduced since a commit
The way to adopt it on a big legacy codebase.

`golangci-lint run --new-from-merge-base main`
Issues since the branch left main

ONLY NEW ISSUES (cont.)

`golangci-lint run -n`
Issues in uncommitted changes
Falls back to HEAD~ when the tree is clean; in CI prefer --new-from-rev HEAD~.

`golangci-lint run --new-from-patch <file>`
Issues within a patch file

OUTPUT

`golangci-lint run --output.json.path stdout`
Machine-readable JSON
Add --output.text.path stderr to keep the human output too.

`golangci-lint run --max-issues-per-linter 0 --max-same-issues 0`
Show every issue, no caps
By default 50 per linter and 3 with the same text are shown.

FORMAT

`golangci-lint fmt`
Format files with enabled formatters
Rewrites files in place; with no formatters enabled it uses gofmt.

`golangci-lint fmt -d`
Show diff, change nothing

`golangci-lint fmt -E goimports`
Format with a given formatter
Formatters: gofmt, goimports, gofumpt, gci, golines, swaggo.

`golangci-lint formatters`
List enabled formatters

CONFIG FILE

Searched as .golangci.yml/.yaml/.toml/.json from the linted path up to /, then in $HOME.

.golangci.yml

```yaml
version: "2"
linters:
  default: standard
  enable: [gosec, revive, misspell]
  disable: [errcheck]
  settings:
    misspell:
      locale: US
  exclusions:
    generated: lax
    presets: [std-error-handling]
    paths: [third_party/]
    rules:
      - path: _test\.go
        linters: [gosec]
formatters:
  enable: [gofumpt, goimports]
run:
  timeout: 5m
```

CONFIG FILE (cont.)

`golangci-lint config verify`
Validate config against the schema

`golangci-lint config path`
Which config file is used

`golangci-lint migrate`
Convert a v1 config to v2
Rewrites in place, keeps a .bck copy, drops comments.

Exclusion presets

Preset	Hides
`comments`	missing doc-comment reports
`std-error-handling`	unchecked Close, print, os.Remove
`common-false-positives`	noisy gosec rules
`legacy`	old v1 default exclusions

SUPPRESS

`//nolint:<linter>`
Silence a linter on this line
No spaces anywhere: // nolint or // nolint: x are ignored.

`//nolint:<linter> // <why>`
Silence with a reason

`//nolint:<a>,<b>`
Silence several linters

`//nolint:all`
Silence every linter

A //nolint line above a func, var block or package clause covers that whole declaration or file.

RECIPES

Adopt on a legacy repo

```
# fail only on new issues
golangci-lint run --new-from-merge-base origin/main
# then clean up per linter
golangci-lint run --enable-only errcheck --max-issues-per-linter 0
```

CI job

```
golangci-lint config verify
golangci-lint fmt -d
golangci-lint run --timeout 5m \
  --output.text.path stdout \
  --output.sarif.path lint.sarif
```

Move from v1 to v2

```
golangci-lint migrate  # keeps .bck
golangci-lint config verify
golangci-lint run
```