RuboCop Ruby linter and formatter

START HERE

`rubocop`
Lint the project

`rubocop --only <cop>`
Run one cop

`rubocop -a`
Fix safe offenses

`rubocop -x`
Fix layout only

`rubocop -A`
Fix all, incl. unsafe

`rubocop --auto-gen-config`
Write a TODO config

`rubocop <path>`
Check files or dirs

`rubocop --init`
Create .rubocop.yml

Run rubocop reads .rubocop.yml → *offenses* → **Autocorrect** -a safe, -A unsafe → *leftovers* → **Fix by hand** or disable a cop → *commit* → **Clean exit 0** CI gate

CONCEPTS

A cop is one rule; cops belong to departments such as Style, Layout, Lint, Metrics, Naming, Security.

Safe autocorrect (-a) keeps behaviour; unsafe (-A) may change it. Review the diff after -A.

Exit codes: 0 clean, 1 offenses found, 2 crash or bad config or options.

RUNNING

`rubocop`
Check all Ruby files here

`rubocop <path> ...`
Check given files or dirs

`rubocop -L`
List files that would be inspected

`rubocop -F`
Stop after first file with offenses

`rubocop --parallel`
Use all CPUs

`rubocop -s <path> < <file>`
Check stdin as if at path
Used by editor integrations.

`rubocop --force-exclusion <path>`
Honour Exclude for explicit paths
Needed so editors and pre-commit hooks skip excluded files.

`rubocop -c <file>`
Use another config file

`rubocop -r <lib>`
Require a Ruby file or extension

`rubocop -v / -V`
Show version / full versions

CHOOSING COPS

`rubocop --only <Cop>,<Dept>`
Run only these cops or departments

`rubocop --except <Cop>`
Run all but these

`rubocop -l`
Lint cops only

`rubocop --safe`
Skip cops not marked safe

`rubocop --show-cops [<Cop>]`
Print cops and their settings

`rubocop --enable-pending-cops`
Turn on pending cops

CHOOSING COPS (cont.)

`rubocop --disable-pending-cops`
Keep pending cops off

AUTOCORRECT

`rubocop -a`
Fix safe offenses

`rubocop -A`
Fix safe and unsafe offenses

`rubocop -x`
Fix layout offenses only

`rubocop -a --only <Cop>`
Fix one cop everywhere

`rubocop -A --only Style/<Cop>`
Roll out a style change

`rubocop -a --disable-uncorrectable`
Add todo comments for the rest
Inserts rubocop:todo comments where autocorrect is impossible.

`rubocop --fail-level A`
Count autocorrected as failures

OUTPUT

`rubocop -f <name>`
Pick an output formatter

`rubocop -f simple`
Offenses grouped by file

`rubocop -f offenses`
Count offenses per cop

`rubocop -f worst`
Files with most offenses

`rubocop -f files`
Only list offending files

`rubocop -f json -o <file>`
Machine readable report

`rubocop -f github`
GitHub Actions annotations

`rubocop -f emacs`
One line per offense

`rubocop -f <name> -f <name2> -o <file>`
Several formatters at once
Each -f applies to the -o that follows it.

`rubocop -E`
Show extra details in messages

`rubocop -d`
Debug output

`rubocop --[no-]color`
Force colours on or off

`rubocop --fail-level <sev>`
Minimum severity that fails

OUTPUT (cont.)

`rubocop --display-only-fail-level-offenses`
Hide offenses below fail level

CACHE AND SERVER

`rubocop -C false`
Disable the result cache

`rubocop --server`
Run via a background server
Reuses a warm process; much faster on repeated runs.

`rubocop --start-server / --stop-server`
Control the server

`rubocop --restart-server`
Restart after config change

`rubocop --no-server`
Bypass the server once

`rubocop --lsp`
Language server for editors

CONFIG FILES

`rubocop --init`
Write a starter .rubocop.yml

`rubocop --auto-gen-config`
Write .rubocop_todo.yml
Also adds inherit_from: .rubocop_todo.yml to .rubocop.yml.

`rubocop --regenerate-todo`
Redo the TODO file with last options

`rubocop --auto-gen-config --exclude-limit <n>`
Files listed before cop is disabled
Default 15.

`rubocop --auto-gen-only-exclude`
Only Exclude, never Max

YAML KEYS

`inherit_from: <file>`
Include other configs

`inherit_gem: { <gem>: <file> }`
Use a config shipped in a gem

`plugins: - <name>`
Load extension plugins
Newer extensions such as rubocop-rails use plugins; older style uses require.

`require: - <lib>`
Load extension by require

`AllCops: NewCops: enable`
Opt into pending cops

YAML KEYS (cont.)

`AllCops: TargetRubyVersion: <x.y>`
Set Ruby version to parse as

`AllCops: Exclude: - <glob>`
Skip paths everywhere

`<Dept>/<Cop>: Enabled: false`
Disable a cop

`<Dept>/<Cop>: Exclude: - <glob>`
Skip paths for one cop

`<Dept>/<Cop>: Max: <n>`
Tune a numeric limit

`<Dept>/<Cop>: Severity: <level>`
Change severity of a cop

`<Dept>: Enabled: false`
Disable a whole department

INLINE COMMENTS

`# rubocop:disable <Cop>`
Turn off from here

`# rubocop:enable <Cop>`
Turn back on

`<code> # rubocop:disable <Cop>`
Disable for one line

`# rubocop:disable <Cop1>, <Cop2>`
Several cops at once

`# rubocop:disable <Dept>`
Whole department

`# rubocop:disable all`
Every cop

`# rubocop:todo <Cop>`
Disable, marked as debt

RECIPES

Adopt on an old codebase

```
rubocop --auto-gen-config
rubocop -a
rubocop -f offenses
# then delete TODO entries as you fix them
```

Roll out one cop safely

```
rubocop --only Style/<Cop> -f offenses
rubocop -a --only Style/<Cop>
git diff --stat
```

CI gate with annotations

```
bundle exec rubocop --parallel -f github
# non-zero exit fails the job
```

Lint only changed files

```
git diff --name-only --diff-filter=d origin/main -- '*.rb' \
  | xargs rubocop --force-exclusion
```