RuboCop Ruby linter and formatter

CONCEPTS

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

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

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

rubocop --force-exclusion *<path>*
Honour Exclude for explicit paths

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

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

CHOOSING COPS (cont.)

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

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

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 -E
Show extra details in messages

CACHE AND SERVER

rubocop -C false
Disable the result cache

rubocop --server
Run via a background server

CONFIG FILES

rubocop --init
Write a starter .rubocop.yml

rubocop --auto-gen-config
Write .rubocop_todo.yml

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

YAML KEYS

inherit_from: *<file>*
Include other configs

plugins: - *<name>*
Load extension plugins

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

AllCops: NewCops: enable
Opt into pending cops

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>: **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