ShellCheck Shell script linter

START HERE

`shellcheck <script.sh>`
Lint a script

`shellcheck -x <script.sh>`
Follow sourced files

`shellcheck -s bash -`
Lint stdin as bash

`shellcheck -S warning <file>`
Hide style and info

`shellcheck -e SC2034 <file>`
Skip one check

`shellcheck -f gcc <file>`
Editor-style output

`shellcheck -f diff <file> | git apply`
Apply auto-fixes

`# shellcheck disable=SC2086`
Silence one line

Script file or stdin → parse → **Shell dialect** -s, directive, shebang → check → **Findings** SC code + severity → filter → **Report** -f format, exit code

CONCEPTS

Every finding has an SC code. Search it on the ShellCheck wiki (github.com/koalaman/shellcheck/wiki/SC2086) for the reason and the fix.

Severity order is error, warning, info, style. Exit code 0 means clean, 1 means findings, 2 means a file could not be read.

Pin the ShellCheck version in CI: new releases add checks and can turn a green build red.

RUNNING

`shellcheck <file>...`
Check one or more scripts

`shellcheck -`
Read the script from stdin

`shellcheck -x <file>`
Also follow files it sources

`shellcheck -a <file>`
Report issues in sourced files too

`shellcheck --files-from <list>`
Read file names from a list

`find . -name '*.sh' -exec shellcheck {} +`
Check every script in a tree

`shellcheck --version`
Show the installed version

`apt install shellcheck`
Install on Debian/Ubuntu

`brew install shellcheck`
Install on macOS

`docker run --rm -v "$PWD:/mnt" koalaman/shellcheck:stable <file>`
Run without installing

DIALECT AND CHECKS

`-s <shell>`
Force sh, bash, dash, ksh or busybox

`-S <severity>`
Minimum level: error, warning, info, style

`-e <codes>`
Exclude codes, comma separated

`-i <codes>`
Report only these codes

`-o <names>`
Enable optional checks

DIALECT AND CHECKS (cont.)

`-o all`
Enable every optional check

`--list-optional`
List the optional checks

`-P <dirs>`
Search path for sourced files

`--extended-analysis=false`
Turn dataflow analysis off (faster)

`-C / --color[=<when>]`
Colour output: always, never, auto

`-W <n>`
Number of wiki links shown (0 hides)

OUTPUT AND EXIT CODES

`-f tty`
Readable text with code excerpts (default)

`-f gcc`
file:line:col lines for editors

`-f json1`
Machine-readable JSON with a comments array

`-f checkstyle`
Checkstyle XML for CI dashboards

`-f diff`
Unified diff of automatic fixes

`-f quiet`
No output, stop at the first finding

`echo $?`
Exit code: 0 clean, 1 findings

`SHELLCHECK_OPTS="--shell=bash -e SC2016"`
Default flags for every run

DIRECTIVES

`# shellcheck disable=SC2086`
Ignore a code for the next command

`# shellcheck disable=SC2034,SC2154`
Ignore several codes

`# shellcheck source=<path>`
Tell it which file a source loads

`# shellcheck source=/dev/null`
Do not follow this source

`# shellcheck source-path=SCRIPTDIR`
Look up sources next to the script

DIRECTIVES (cont.)

`# shellcheck shell=bash`
Set the dialect for this file

`# shellcheck enable=<name>`
Switch on an optional check (file-wide)

A directive on its own line before the first command applies to the whole file; anywhere else it covers only the next command or block.

CONFIG FILES

`.shellcheckrc`
Project config, found in script dir or parents

`shell=bash`
rc: default dialect

`disable=SC2034`
rc: ignore a code everywhere

`external-sources=true`
rc: always follow sourced files

`source-path=SCRIPTDIR`
rc: lookup directory for sources

`enable=require-variable-braces`
rc: turn on an optional check

`--norc`
Ignore .shellcheckrc files

`--rcfile <file>`
Use this config instead

Only the first rc file found is used; it falls back to ~/.shellcheckrc and then ~/.config/shellcheckrc.

COMMON WARNINGS

SC2086
Quote $var to stop splitting and globbing

SC2046
Quote $(...) to stop word splitting

SC2068
Use "$@" instead of unquoted $@

SC2034
Variable assigned but never used

SC2154
Variable used but never assigned

SC2164
Use cd ... || exit in case cd fails

SC2155
Declare and assign separately

COMMON WARNINGS (cont.)

SC2162
read without -r mangles backslashes

SC2006
Use $(...) instead of backticks

SC2016
Single quotes do not expand variables

SC2012
Do not parse ls; use a glob or find

SC2002
Useless cat; redirect with < file

SC2015
A && B || C is not if-then-else

SC2181
Test the command directly, not $?

SC2207
Use mapfile or read -a, not arr=($(cmd))

SC2115
Guard rm -rf "$dir/" against empty dir

SC2148
Add a shebang or a shell directive

SC1090 / SC1091
Cannot follow source; use -x or a source= directive

SC3010
[[is not POSIX; use [in sh scripts

SC2064
Quote trap commands with single quotes

RECIPES

Makefile target

```
lint:
shellcheck -x scripts/*.sh
```

pre-commit hook

```
# .pre-commit-config.yaml
repos:
  - repo: https://github.com/koalaman/shellcheck-precommit
    rev: v0.11.0
    hooks:
      - id: shellcheck
        args: ["--severity=warning"]
```

Project .shellcheckrc

```
# .shellcheckrc
shell=bash
external-sources=true
source-path=SCRIPTDIR
disable=SC2016
```

Fix what it can, then recheck

```
shellcheck -f diff deploy.sh | git apply
shellcheck -x -S warning deploy.sh
```