# Semgrep Static code scanning

**START HERE**

`semgrep scan --config auto`
Scan with recommended rules

`semgrep scan --config <rules>`
Use a rule file or ruleset

`semgrep -e '<pattern>' --lang <lang>`
Ad-hoc pattern search

`semgrep scan --severity ERROR`
Only serious findings

`semgrep scan --json -o <file>`
Machine-readable output

`semgrep scan --error`
Exit 1 on findings

`semgrep ci`
Run in CI on the diff

`# nosemgrep`
Ignore a line

**Rules** YAML / registry → `--config` → **Targets** files, ignores applied → `scan` → **Findings** severity, location → `--json` / `--sarif` → **Report** exit code, files

## CONCEPTS

Semgrep matches code structure, not text: a pattern like foo($X) finds the call however it is spaced or nested.

Without --error, scan exits 0 even when it finds issues. Add it (or use semgrep ci) to fail a pipeline.

Rules from the registry need the network, and by default metrics are sent; use --metrics off to disable, --config <file> to stay local.

## SCANNING

`semgrep scan`
Scan the current directory
*Needs a --config, or a local semgrep config file is not picked up automatically.*

`semgrep scan --config auto`
Registry rules chosen for your code
*Requires login-free network access; sends project metadata.*

`semgrep scan --config <file|dir>`
Run local rule files

`semgrep scan --config p/<ruleset>`
Run a registry ruleset
*For example p/security-audit or p/python.*

`semgrep scan <path>`
Scan only a file or folder

`semgrep scan --severity <level>`
Filter by INFO, WARNING, ERROR
*Repeat the flag for several levels.*

`semgrep scan --exclude-rule <id>`
Skip a rule

`semgrep scan --include <glob>`
Only paths matching glob

`semgrep scan --exclude <glob>`
Skip paths matching glob

## SCANNING (cont.)

`semgrep scan -j <n>`
Number of parallel jobs

`semgrep scan --timeout <sec>`
Per-rule per-file time limit
*Files over 1 MB are skipped by default.*

`semgrep scan --max-target-bytes <n>`
Raise the file size limit

`semgrep scan --metrics off`
Disable metrics

## AD-HOC PATTERNS

`semgrep -e '<pattern>' -l <lang> <path>`
Search with a pattern

`semgrep -e 'eval(...)' -l python .`
Find any eval call
*... matches any arguments.*

`semgrep -e '$X == $X' -l <lang> .`
Metavariable, same value twice
*$X binds to any expression; repeating it requires equality.*

`semgrep -e '<pat>' --replacement '<new>' -l <lang> --autofix`
Rewrite matches
*Add --dryrun to preview the change.*

**Pattern syntax**

| Syntax | Matches |
|---|---|
| `...` | any arguments or statements |
| `$X` | any expression (metavariable) |
| `<... p ...>` | p nested anywhere inside |

## OUTPUT

`semgrep scan --json`
JSON to stdout

`semgrep scan --sarif -o <file>`
SARIF report to a file
*For code-scanning uploads such as GitHub.*

`semgrep scan --json-output <file>`
JSON file, text on screen

`semgrep scan --junit-xml`
JUnit XML output

## OUTPUT (cont.)

`semgrep scan --gitlab-sast`
GitLab SAST format

`semgrep scan --dataflow-traces`
Show taint paths in text

`semgrep scan -q`
Quiet: findings only

`semgrep scan -v`
Verbose, show skipped files

## CI & BASELINE

`semgrep ci`
Scan in CI, fail on blocking findings
*Works with a Semgrep account; reports to the platform.*

`semgrep ci --baseline-commit <ref>`
Only findings new since ref

`semgrep scan --baseline-commit <ref>`
Same for plain scan
*Compares against the given commit; needs a git repo.*

`semgrep scan --error`
Exit 1 when findings exist

`semgrep ci --subdir <dir>`
Scan only a subdirectory

`semgrep login`
Authenticate with semgrep.dev

`semgrep ci --code`
Run only code (SAST) scans

`semgrep scan --strict`
Fail on invalid target code

## IGNORING

`# nosemgrep`
Ignore this line or the next
*Put it as a comment on the matching line or the line above.*

`# nosemgrep: <rule-id>`
Ignore one rule only
*Use the full rule id with namespace; separate several with commas.*

`.semgrepignore`
Paths to skip, gitignore syntax
*Replaces the default ignore list, which skips tests/, vendor/, node_modules/ and more.*

## IGNORING (cont.)

`:include .gitignore`
Reuse gitignore in .semgrepignore

`semgrep scan --no-git-ignore`
Also scan gitignored files

`semgrep scan --disable-nosem`
Report nosemgrep lines anyway

## RULES

`semgrep scan --validate --config <file>`
Check rule files are valid

`semgrep scan --test <dir>`
Run rule tests
*Test files use # ruleid: and # ok: comments above the line.*

`semgrep scan --pro`
Pro engine, cross-file analysis
*Needs login and the Pro engine (install-semgrep-pro).*

`semgrep scan --oss-only`
Never use Pro features

`semgrep scan --secrets`
Secrets scanning
*Account required.*

**Exit codes**

| Code | Meaning |
|---|---|
| 0 | ok, or findings without --error |
| 1 | findings with --error |
| 2 | fatal error |
| 4 | invalid pattern in rule |
| 7 | invalid rule or missing config |

## RECIPES

**Fail CI on serious findings**

```
semgrep scan --config auto \
  --severity ERROR --error
```

**Only new findings in a pull request**

```
semgrep scan --config p/ci \
  --baseline-commit origin/main
--error
```

**Quick custom rule check**

```
semgrep -e 'requests.get(..., verify=False)' \
  -l python src/
```

**Report for code scanning**

```
semgrep scan --config auto \
  --sarif -o semgrep.sarif
```