Checkov IaC security scanner

START HERE

`checkov -d <dir>`
Scan a directory

`checkov -f <file>`
Scan one file

`checkov -d <dir> --framework terraform`
Only one framework

`checkov -d <dir> --compact --quiet`
Short, failures only

`checkov -d <dir> -c <id>`
Run only this check

`checkov -d <dir> --skip-check <id>`
Skip a check

`checkov -d <dir> -o json`
Machine-readable output

`checkov -d <dir> --baseline <file>`
Report only new failures

IaC files tf, yaml, Dockerfile → `checkov -d` → **Framework runners** auto-detected → `policies` → **Checks** CKV_* IDs → `-o` → **Report** pass / fail / skip

CONCEPTS

Checkov scans infrastructure-as-code (Terraform, Kubernetes, Dockerfiles, CloudFormation, CI configs) for misconfigurations without deploying anything.

Every check has an ID like CKV_AWS_20; you use these IDs to select, skip and suppress checks.

By default all matching frameworks run; narrow with --framework to speed up large repos.

SETUP

`pip install checkov`
Install with pip

`checkov --version`
Show the version

`checkov --help`
List all options

`checkov --list`
List available checks

`checkov --show-config`
Show settings and their sources

SCANNING

`checkov -d <dir>`
Scan a directory recursively

`checkov -f <file>`
Scan a single file

Cannot be combined with -d; the file type limits which frameworks run.

`checkov -f <a.tf> -f <b.tf>`
Scan several files

`checkov -f tfplan.json --repo-root-for-plan-enrichment <dir>`
Scan a Terraform plan

Export the plan with terraform show -json; the root option links findings back to source files.

`checkov -d <dir> --skip-path <regex>`
Ignore paths matching a regex

`checkov -d <dir> --var-file <file>`
Add a variable file

`checkov -d <dir> --download-external-modules true`
Fetch remote Terraform modules

SCANNING (cont.)

`checkov -d <dir> --external-modules-download-path <dir>`
Where downloaded modules go

FRAMEWORKS

`checkov -d <dir> --framework <name>`
Run only these frameworks

Repeat names after the flag: --framework terraform secrets.

`checkov -d <dir> --skip-framework <name>`
Leave frameworks out

`checkov -d <dir> --framework secrets`
Hunt for hard-coded secrets

`checkov -d <dir> --framework kubernetes helm`
Scan manifests and charts

`checkov -d <dir> --framework dockerfile`
Lint Dockerfiles

`checkov -d <dir> --framework github_actions`
Scan GitHub workflows

Common names: terraform, terraform_plan, kubernetes, helm, kustomize, dockerfile, cloudformation, arm, bicep, github_actions, gitlab_ci, secrets.

CHOOSING CHECKS

`checkov -d <dir> -c <id>,<id>`
Run only these checks

Also accepts severities such as HIGH when a platform key is set.

`checkov -d <dir> --check <id>`
Long form of -c

`checkov -d <dir> --skip-check <id>,<id>`
Skip these checks

`checkov -d <dir> --external-checks-dir <dir>`
Load custom Python or YAML checks

OUTPUT

`checkov -d <dir> --compact`
Hide code snippets

`checkov -d <dir> --quiet`
Show only failed checks

OUTPUT (cont.)

`checkov -d <dir> -o <format>`
Pick an output format

Formats: cli, json, junitxml, sarif, csv, cyclonedx, spdx, gitlab_sast, github_failed_only.

`checkov -d <dir> -o json`
JSON report

`checkov -d <dir> -o sarif`
SARIF for code scanning

`checkov -d <dir> -o junitxml`
JUnit report for CI

`checkov -d <dir> -o cli -o json --output-file-path console,<file>`
Print and save in one run

Entries in --output-file-path pair up with the -o formats in order; console keeps that format on screen.

`checkov -d <dir> --output-file-path <dir>`
Save reports in a folder

EXIT CODES AND CI

`checkov -d <dir> --soft-fail`
Always exit 0

`checkov -d <dir> --hard-fail-on <id|sev>`
Fail only on these checks

With a severity such as HIGH this needs a platform key; IDs work offline.

`checkov -d <dir> --soft-fail-on <id|sev>`
Pass when only these fail

Exit codes

Code	Meaning
0	All checks passed (or soft-fail)
1	At least one check failed
2	Crash or platform failure

BASELINES

`checkov -d <dir> --create-baseline`
Save current findings as a baseline

Writes .checkov.baseline in the scanned directory.

`checkov -d <dir> --baseline <file>`
Report only new failures

Baselines let you adopt Checkov on a legacy repo: block new problems now, fix old ones later.

SUPPRESSING FINDINGS

Terraform: inside the resource

```
resource "aws_s3_bucket" "site"
{
#checkov:skip=CKV_AWS_20:public
static site
}
```

Kubernetes: annotation

```
metadata:
  annotations:
    checkov.io/skip1:
CKV_K8S_20=needs root for init
```

Dockerfile: any line

```
#checkov:skip=CKV_DOCKER_2: no
healthcheck for jobs
```

The reason after the second colon is optional but makes reviews easier; skips show up in the report as SKIPPED.

CONFIG FILE

`checkov --config-file <file>`
Use a YAML config file

A .checkov.yaml in the working directory is picked up automatically.

`checkov -d <dir> --create-config <file>`
Write current flags to a config

.checkov.yaml

```
directory:
  - infra
framework:
  - terraform
  - secrets
skip-check:
  - CKV_AWS_20
compact: true
quiet: true
```

RECIPES

First scan of a Terraform repo

```
checkov -d . --framework
terraform --compact --quiet
# fix or suppress, then add a
baseline for the rest
checkov -d . --create-baseline
```

CI gate with a SARIF report

```
checkov -d . --framework
terraform \
  -o cli -o sarif --output-file-
path console,results.sarif
# exit code 1 fails the job
```

Scan a plan with resolved values

```
terraform plan -out tf.plan
terraform show -json tf.plan >
tfplan.json
checkov -f tfplan.json --repo-
root-for-plan-enrichment .
```