Trivy Security scanner

Target image, fs, repo, k8s → `analyze` → **Scanners** vuln, secret, misconfig, license → `match` → **Databases** cached vuln DB and checks → `filter` → **Report** table, json, sarif, SBOM

CONCEPTS

Pick the target type (image, fs, repo, config, k8s, sbom), then choose what to look for with --scanners.

Defaults: image, fs and repo run vuln + secret; config runs misconfig only. License scanning is off unless requested.

The first run downloads the vulnerability DB into the cache; later runs reuse it, so offline CI needs a pre-warmed cache.

SCAN TARGETS

`trivy image` *`<image>`*
Scan an image (local daemon, then registry)

`trivy image --input` *`<file.tar>`*
Scan an image saved as tar

`trivy fs` *`<dir>`*
Scan a local project, incl. lock files

`trivy repo` *`<path|url>`*
Scan a local or remote git repo

SCAN TARGETS (cont.)

`trivy rootfs` *`<dir>`*
Scan an extracted root filesystem

`trivy sbom` *`<file>`*
Scan an existing SBOM for vulns

`trivy image --platform` *`<os/arch>`*
Scan a non-default platform

`trivy registry login` *`<registry>`*
Store registry credentials

SCANNERS

`--scanners vuln`
Only known vulnerabilities

`--scanners vuln,secret,misconfig`
Combine several scanners

`--scanners misconfig`
IaC and Dockerfile misconfigurations

`--scanners secret`
Hard-coded secrets only

`--scanners license`
Report package licenses

`trivy config` *`<dir>`*
Misconfig scan of Terraform, K8s, Helm, Dockerfile

FILTERING

`--severity` *`<list>`*
Keep only these severity levels

`--ignore-unfixed`
Skip vulns that have no fix yet

`--ignore-status` *`<list>`*
Hide statuses like affected,will_not_fix

`.trivyignore`
One finding ID per line; # comments

`--ignorefile` *`<file>`*
Use another ignore file (.trivyignore.yaml)

`--skip-dirs` *`<glob>`*
Do not scan matching directories

`--skip-files` *`<glob>`*
Do not scan matching files

OUTPUT AND CI

`-f table`
Human-readable table (default)

`-f json -o` *`<file>`*
Write JSON to a file

`-f sarif -o` *`<file>`*
SARIF for code-scanning tools

OUTPUT AND CI (cont.)

`--exit-code` *`<n>`*
Exit with n when issues are found

`-q`
Quiet: no progress bar or log noise

`--no-progress`
Hide the progress bar only

`-d`
Debug logging

SBOM

`-f cyclonedx -o` *`<file>`*
Generate a CycloneDX SBOM

`-f spdx-json -o` *`<file>`*
Generate an SPDX JSON SBOM

KUBERNETES

`trivy k8s --report summary`
Cluster overview (default report)

`trivy k8s --report all`
Detailed findings for the cluster

`trivy k8s --scanners misconfig`
Only misconfigurations in the cluster

KUBERNETES (cont.)

`--include-namespaces` *`<list>`*
Limit to these namespaces

DATABASE AND CACHE

`--skip-db-update`
Use cached vuln DB, no download

`--download-db-only`
Fetch the DB, do not scan

`--cache-dir` *`<dir>`*
Move the cache (CI caching)

RECIPES

Gate a CI build on serious, fixable issues

```
trivy image --severity
HIGH,CRITICAL \
  --ignore-unfixed --exit-
code 1 <image>
```

Project audit with a JSON report

```
trivy fs --scanners
vuln,secret,misconfig \
  -f json -o report.json .
```

Build an image and scan it before pushing

```
docker build -t <image> .
trivy image --exit-code 1 --
severity CRITICAL <image>
docker push <image>
```