Cosign Sign and verify artifacts

OIDC login Google, GitHub, ... → `identity token` → **Fulcio** short-lived cert → `sign with ephemeral key` → **Signature** stored in registry → `log entry` → **Rekor** transparency log

Keyless signing

CONCEPTS

Sign by digest (image@sha256:...), not by tag: a tag can move to a different image after you sign.

Keyless verify needs both an expected identity and an expected OIDC issuer; without them anyone with any Sigstore identity would pass.

Signatures and attestations are stored as extra artifacts next to the image in the same registry, so signing needs push access.

KEYS

`cosign generate-key-pair`
Create cosign.key and cosign.pub

`cosign generate-key-pair --output-key-prefix <name>`
Choose the key file names

`--key awskms:// / gcpkms:// / azurekms:// / hashivault://`
Key in a KMS (URI schemes)

SIGN

`cosign sign --key cosign.key <image>@<digest>`
Sign with a local key

`cosign sign <image>@<digest>`
Keyless sign (OIDC login)

`cosign sign --yes <image>@<digest>`
Skip confirmation prompts (CI)

VERIFY

`cosign verify --key cosign.pub <image>`
Verify with a public key

`cosign verify --certificate-identity <id> --certificate-oidc-issuer <url> <image>`
Verify a keyless signature

`cosign verify --key cosign.pub <image> | jq`
Read the verified payload

ATTESTATIONS

`cosign attest --key cosign.key --predicate <file> <image>@<digest>`
Attach a signed predicate

ATTESTATIONS (cont.)

`cosign attest --type <type> --predicate <file> <image>@<digest>`
Set the predicate type

`cosign verify-attestation --key cosign.pub <image>`
Verify attestations

`cosign verify-attestation --type <type> --policy <file> --key cosign.pub <image>`
Check predicate against a policy

`cosign verify-attestation ... <image> | jq -r .payload | base64 -d`
Decode the statement

BLOBS

`cosign sign-blob --bundle <bundle> <file>`
Keyless sign a file into a bundle

`cosign verify-blob --bundle <bundle> --certificate-identity <id> --certificate-oidc-issuer <url> <file>`
Verify a keyless file signature

BLOBS (cont.)

`cosign sign-blob --key cosign.key <file>`
Sign a file with a key

`cosign verify-blob --key cosign.pub --bundle <bundle> <file>`
Verify a file with a public key

`cosign sign-blob --yes --bundle <bundle> <file>`
Non-interactive file signing

INSPECT AND MANAGE

`cosign tree <image>`
Show signatures, SBOMs, attestations

`cosign login <registry>`
Log in to a registry

`cosign version`
Print the cosign version

ENVIRONMENT

`COSIGN_PASSWORD=<pw>`
Key password without a prompt

`SIGSTORE_ID_TOKEN=<token>`
Supply an OIDC token, skip login

ENVIRONMENT (cont.)

`SIGSTORE_ROOT_FILE=<pem>`
Use a custom root certificate

`cosign env --show-descriptions`
List all variables cosign reads

In CI (GitHub Actions) keyless signing picks up the workflow's OIDC token automatically; the job needs id-token: write.

RECIPES

Key-based sign and verify

```
cosign generate-key-pair
cosign sign --key cosign.key
ghcr.io/me/
app@sha256:<digest>
cosign verify --key cosign.
pub ghcr.io/me/
app@sha256:<digest>
```

Keyless in CI, verify by identity

```
cosign sign --yes ghcr.io/
me/app@sha256:<digest>
cosign verify ghcr.io/me/
app@sha256:<digest> \
  --certificate-identity-
regexp 'https://github.com/
me/app/.*' \
  --certificate-oidc-issuer
https://token.actions.
githubusercontent.com
```