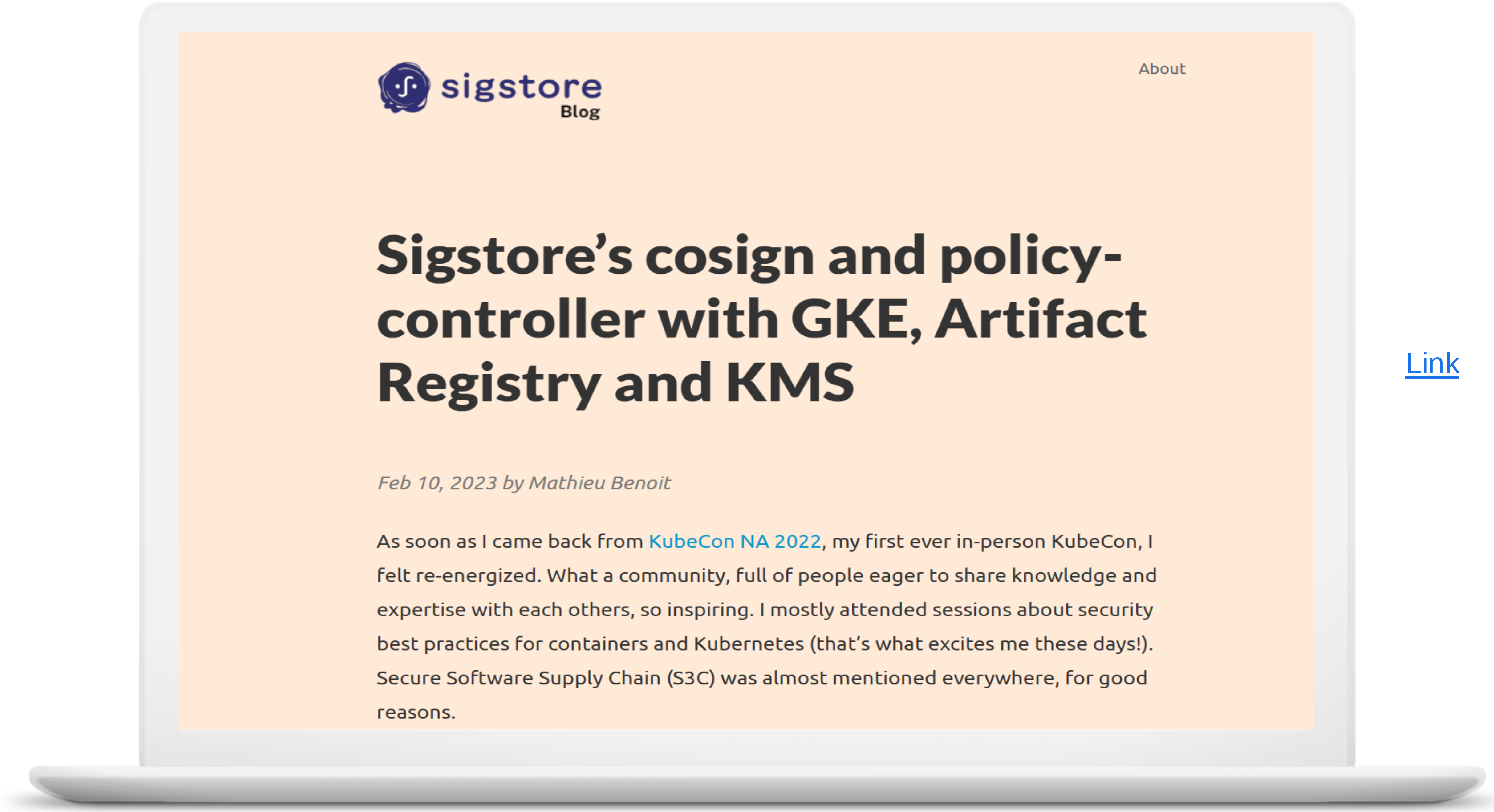
March 29th, 2023
GDG Cloud Montréal

Sigstore's Cosign with GKE, Artifact Registry and KMS

Mathieu Benoit
GKE/Anthos DevRel Engineer - Google Cloud
CNCF Ambassador
Medium | Blog | LinkedIn



sigstore Blog

About

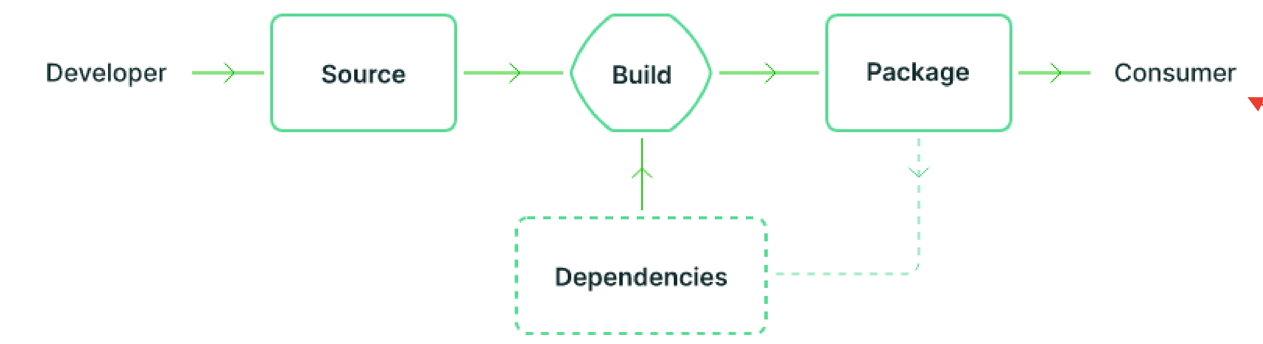
Sigstore's cosign and policy-controller with GKE, Artifact Registry and KMS

Feb 10, 2023 by Mathieu Benoit

As soon as I came back from KubeCon NA 2022, my first ever in-person KubeCon, I felt re-energized. What a community, full of people eager to share knowledge and expertise with each others, so inspiring. I mostly attended sessions about security best practices for containers and Kubernetes (that's what excites me these days!). Secure Software Supply Chain (S3C) was almost mentioned everywhere, for good reasons.

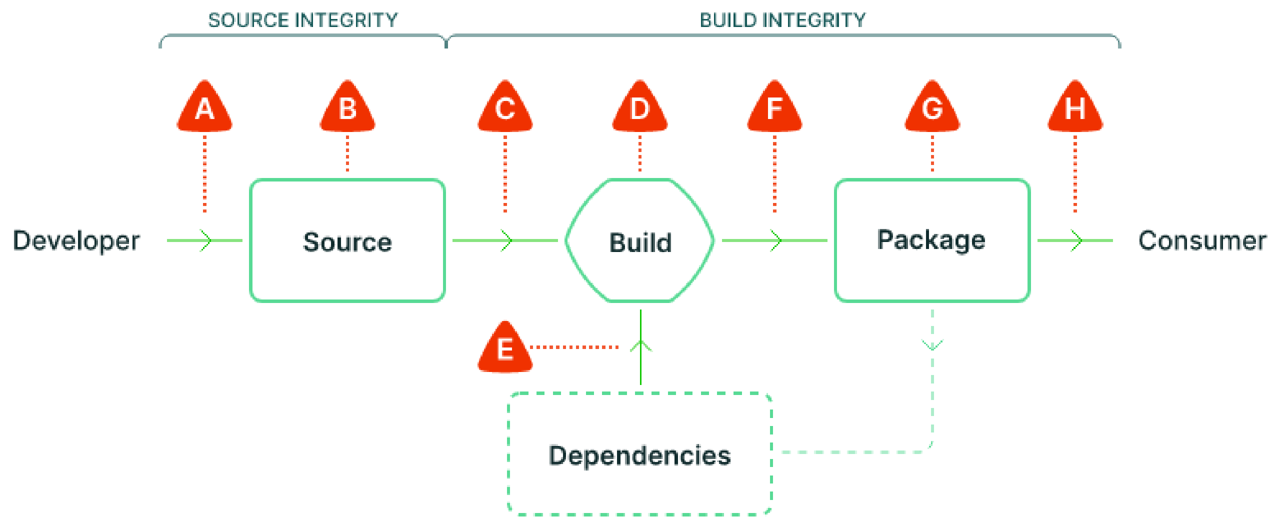
Link

Zero Trust with Software Supply Chain - slsa.dev

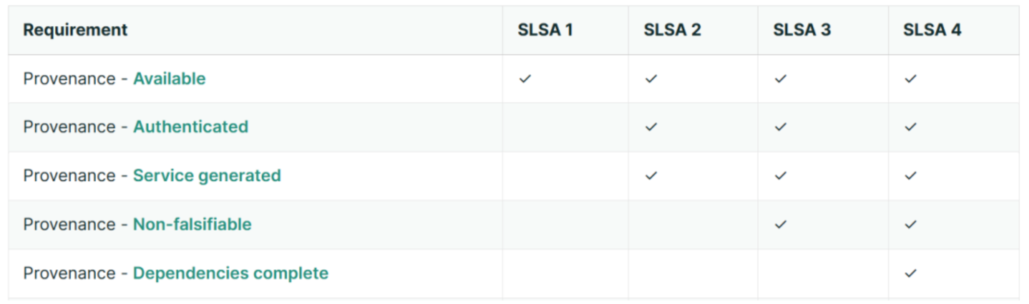


You use an artifact from the right place, but it’s not what the owner intended:

- ○ Compromised account
- ○ Compromised build process
- ○ Compromised package repository

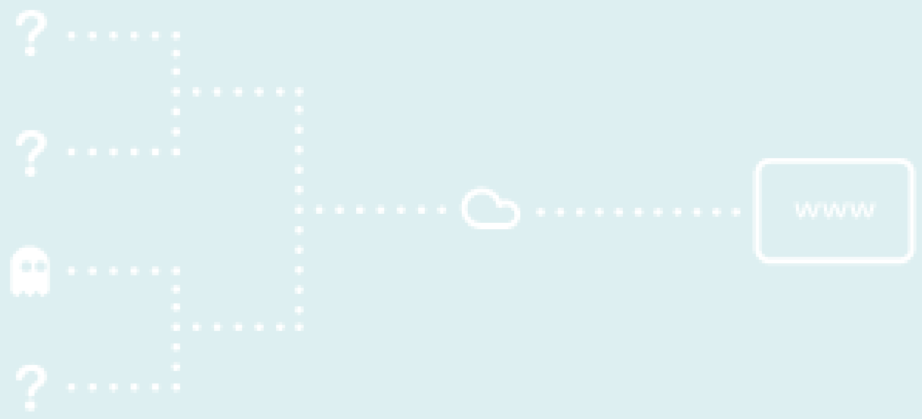


A Submit unauthorized change
B Compromise source repo
C Build from modified source
D Compromise build process
E Use compromised dependency
F Upload modified package
G Compromise package repo
H Use compromised package



Requirement	SLSA 1	SLSA 2	SLSA 3	SLSA 4
Provenance - Available	✓	✓	✓	✓
Provenance - Authenticated		✓	✓	✓
Provenance - Service generated		✓	✓	✓
Provenance - Non-falsifiable			✓	✓
Provenance - Dependencies complete				✓

Sigstore



sigstore was started to improve supply chain technology for anyone using open source projects. It's for open source maintainers, by open source maintainers.

And it's a direct response to today's challenges, a work in progress for a future where the integrity of what we build and use is up to standard.

Sign code

Easy authentication and smart cryptography work in the background. Just push your code.

Verify signatures

A transparency log stores data like who created something and how, so you know it hasn't been changed.

Monitor activity

Logged data is readily auditable, for future monitors and integrations to build into your security workflow.

Agenda

1 Sign

2 Verify signature with **Sigstore policy-controller** and **Kyverno**

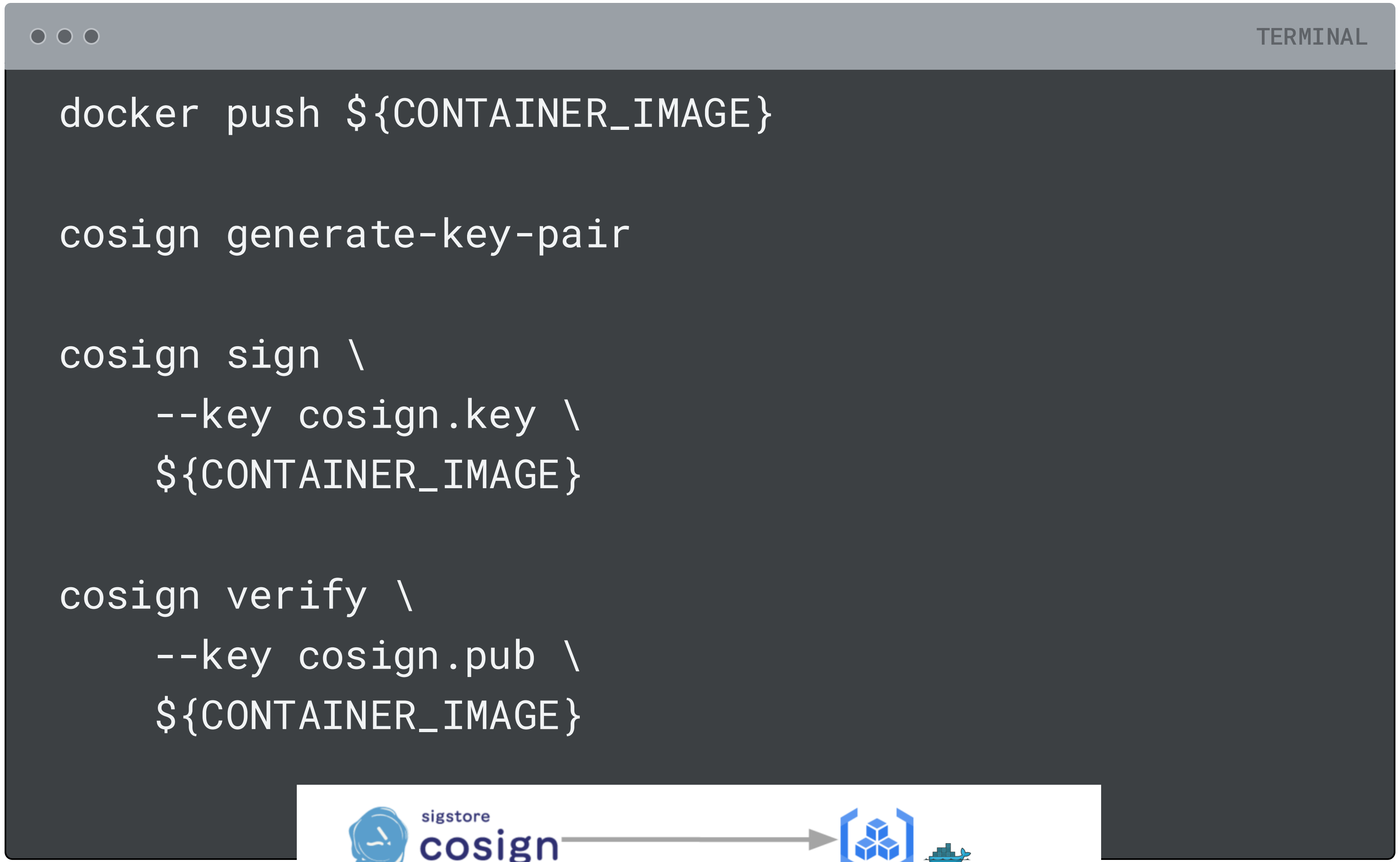
3 Sign

4 Verify signature with **Flux**

5 Sign

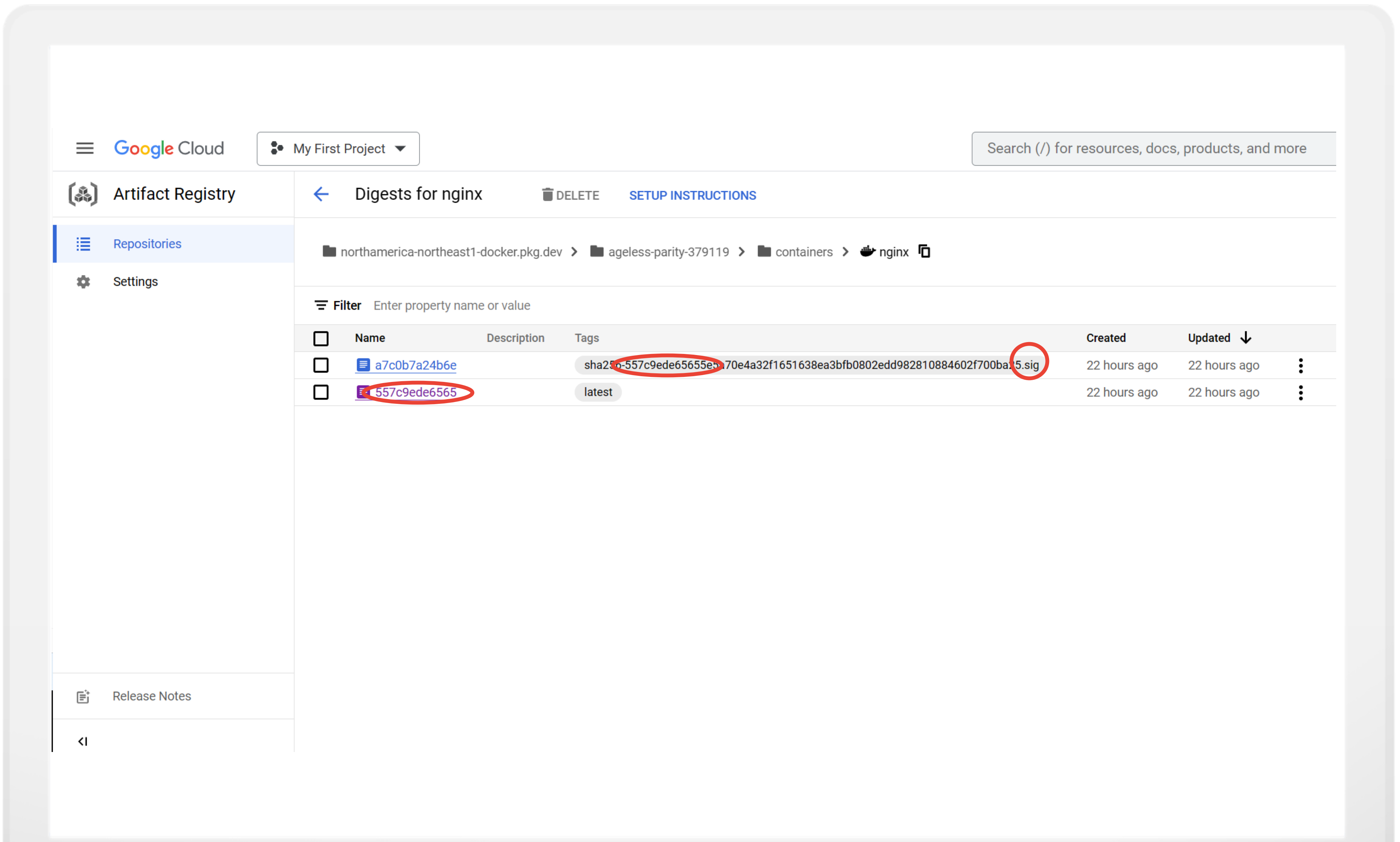
6 Verify signature with **Flux**

Sign a container image locally with Cosign


TERMINAL

```
docker push ${CONTAINER_IMAGE}

cosign generate-key-pair

cosign sign \
    --key cosign.key \
    ${CONTAINER_IMAGE}

cosign verify \
    --key cosign.pub \
    ${CONTAINER_IMAGE}
```

sigstore
cosign
docker
Google Artifact Registry



Digests for nginx

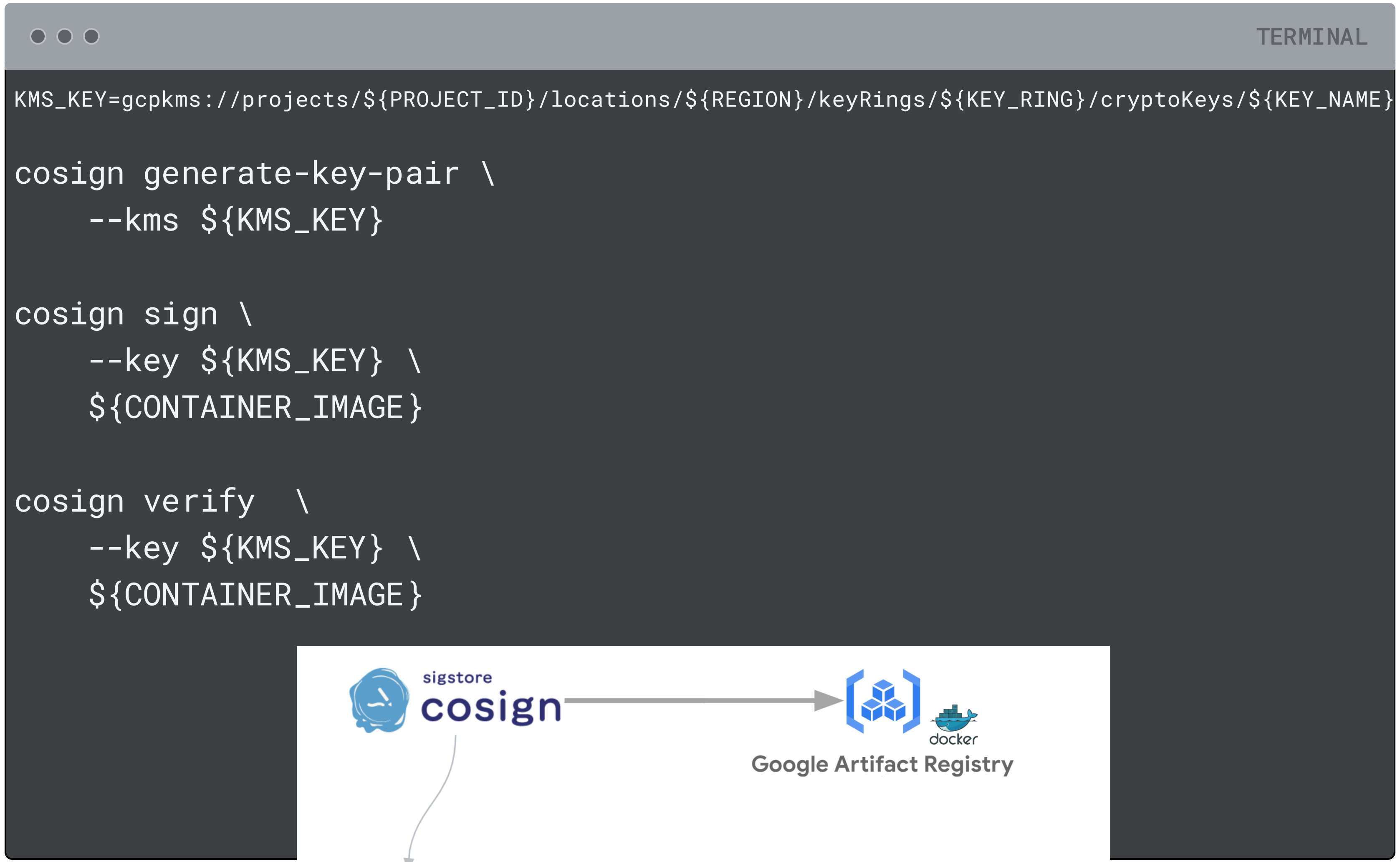
DELETE SETUP INSTRUCTIONS

northamerica-northeast1-docker.pkg.dev > ageless-parity-379119 > containers > nginx

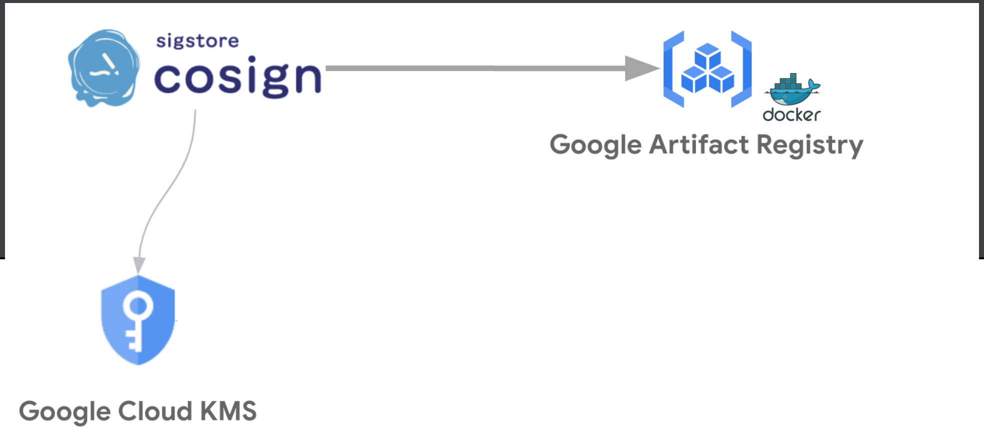
Filter Enter property name or value

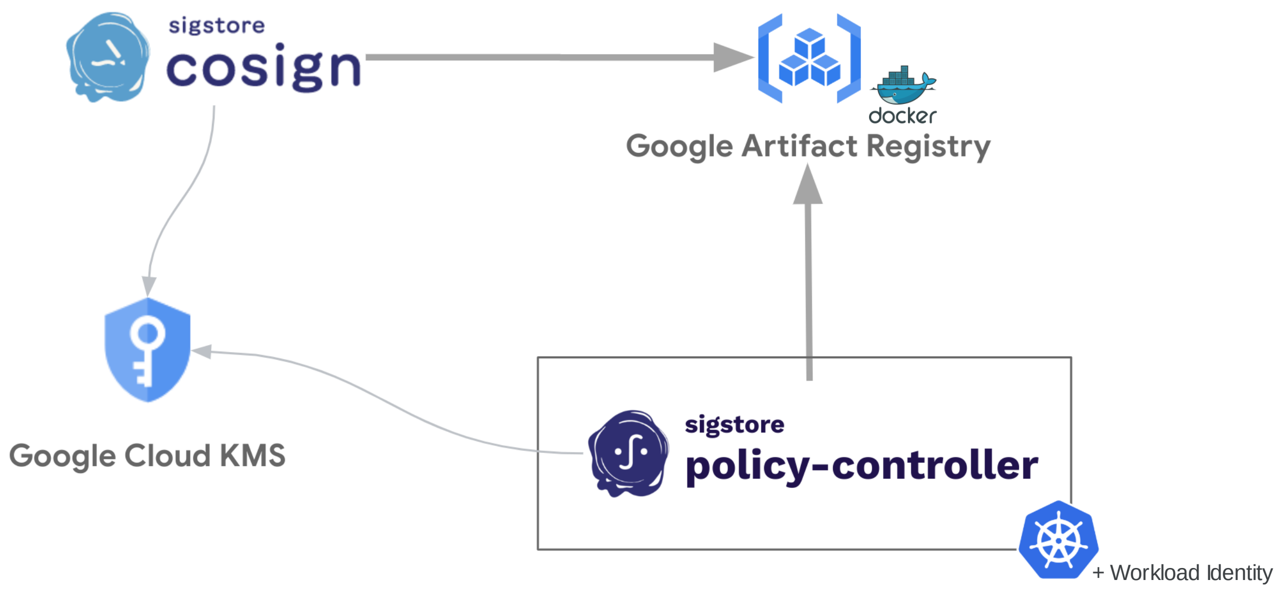
Name	Description	Tags	Created	Updated
a7c0b7a24b6e		sha256-557c9ede65655e5a70e4a32f1651638ea3bfb0802edd982810884602f700ba25.sig	22 hours ago	22 hours ago
557c9ede6565		latest	22 hours ago	22 hours ago

Sign a container image via Cloud KMS with Cosign


TERMINAL

```
KMS_KEY=gcpkms://projects/${PROJECT_ID}/locations/${REGION}/keyRings/${KEY_RING}/cryptoKeys/${KEY_NAME}

cosign generate-key-pair \
    --kms ${KMS_KEY}

cosign sign \
    --key ${KMS_KEY} \
    ${CONTAINER_IMAGE}

cosign verify  \
    --key ${KMS_KEY} \
    ${CONTAINER_IMAGE}
```



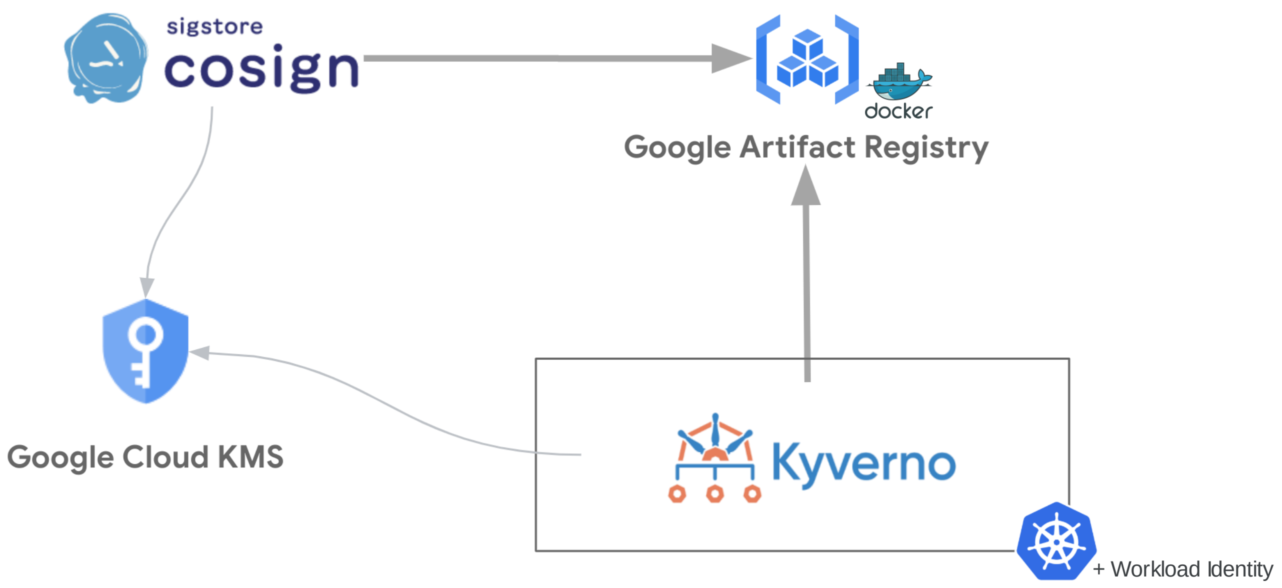


[Link]

cluster-image-policy.yaml

```
apiVersion: policy.sigstore.dev/v1alpha1
kind: ClusterImagePolicy
metadata:
  name: private-signed-images-cip
spec:
  images:
  - glob: "**"
  authorities:
  - key:
      kms: ${KMS_KEY}
```

Demo!



Link

cluster-policy.yaml

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: private-signed-images-cp
spec:
  validationFailureAction: Enforce
  background: true
  rules:
  - name: private-signed-images
    match:
      any:
      - resources:
          kinds:
          - Pod
    verifyImages:
      - imageReferences:
        - "*"
        attestors:
        - count: 1
          entries:
          - keys:
              kms: ${KMS_KEY}
```

1 Sign

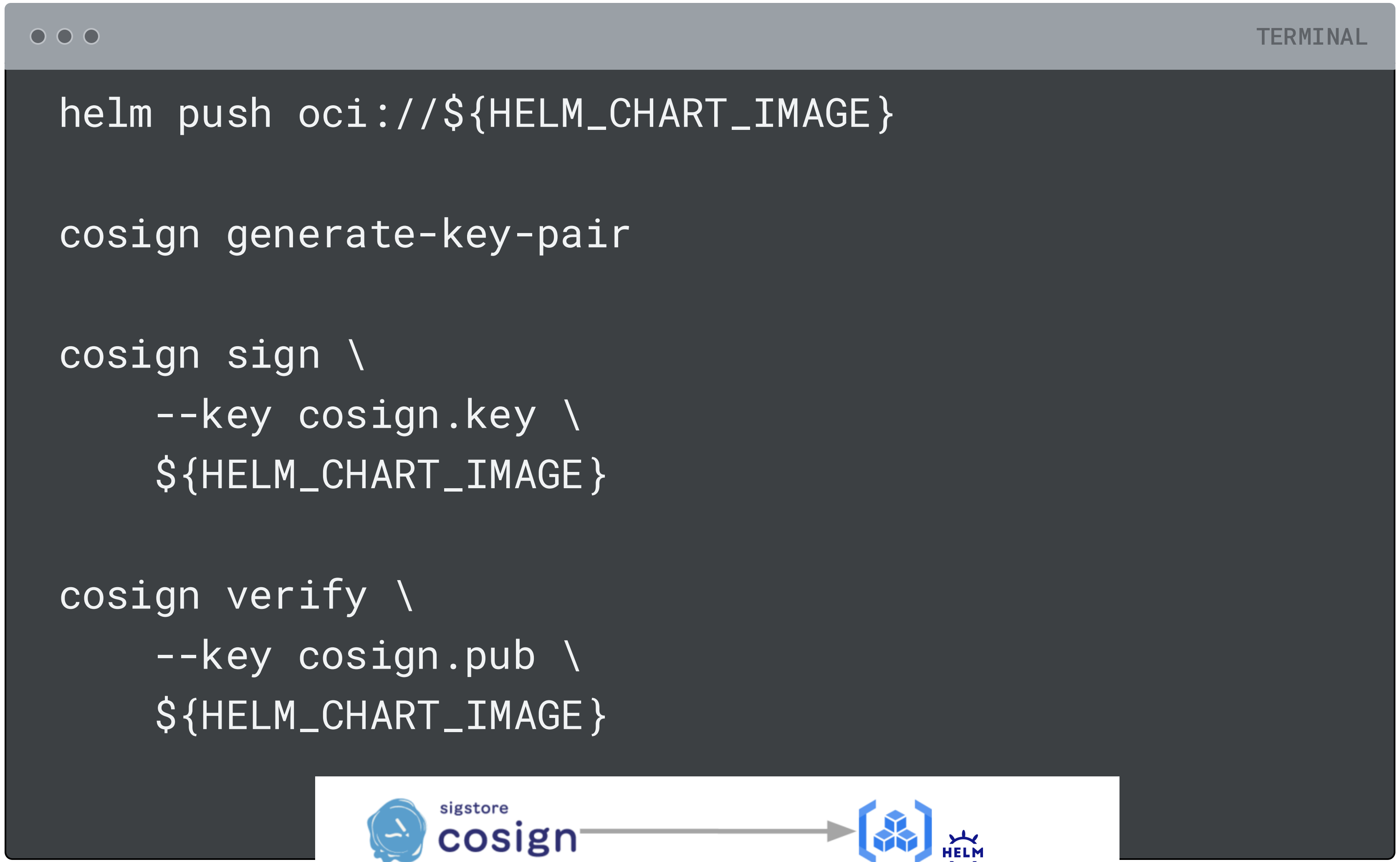
2 Verify signature with **Sigstore policy-controller** and **Kyverno**

Congrats! 🎉

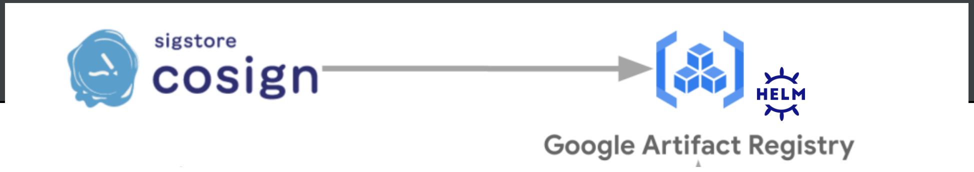
What about my Helm charts and OCI images?

3. Sign
4. Verify signature with **Flux**
5. Sign
6. Verify signature with **Flux**

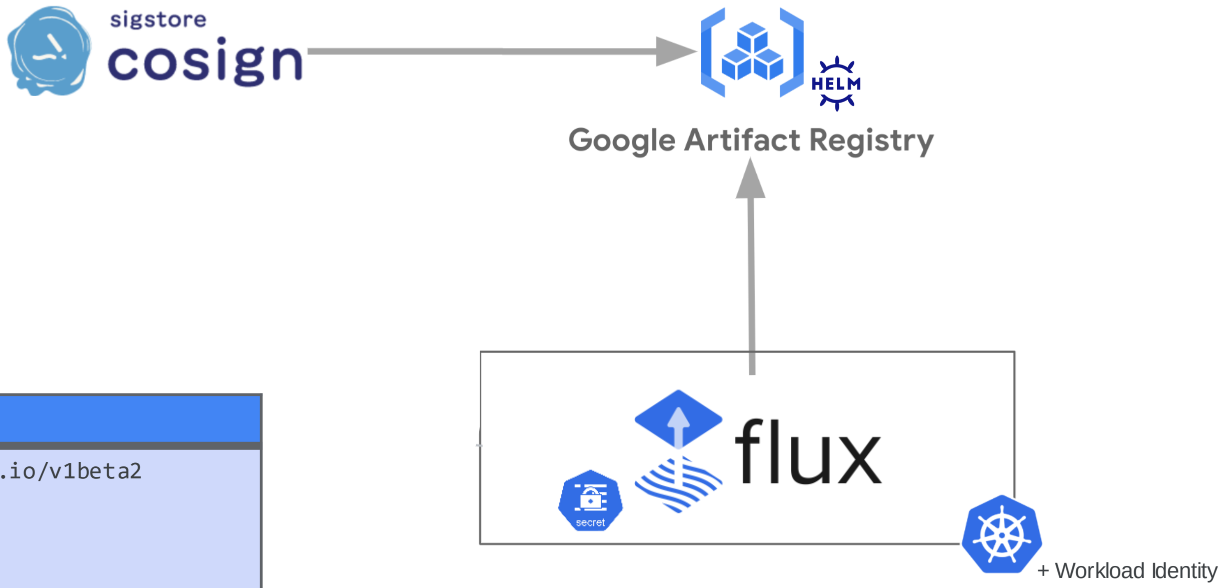
Sign an Helm chart


TERMINAL

```
helm push oci://${HELM_CHART_IMAGE}

cosign generate-key-pair

cosign sign \
    --key cosign.key \
    ${HELM_CHART_IMAGE}

cosign verify \
    --key cosign.pub \
    ${HELM_CHART_IMAGE}
```



Helm supply chain security · Issue #10644 - helm package --sign



Link

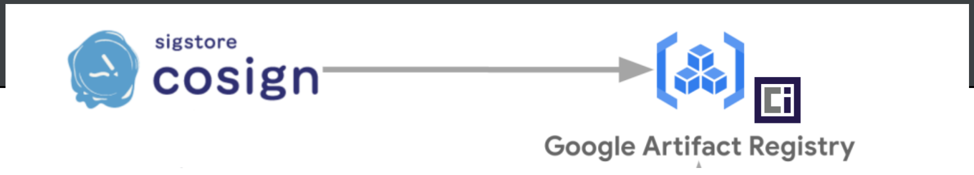
oci-repository.yaml

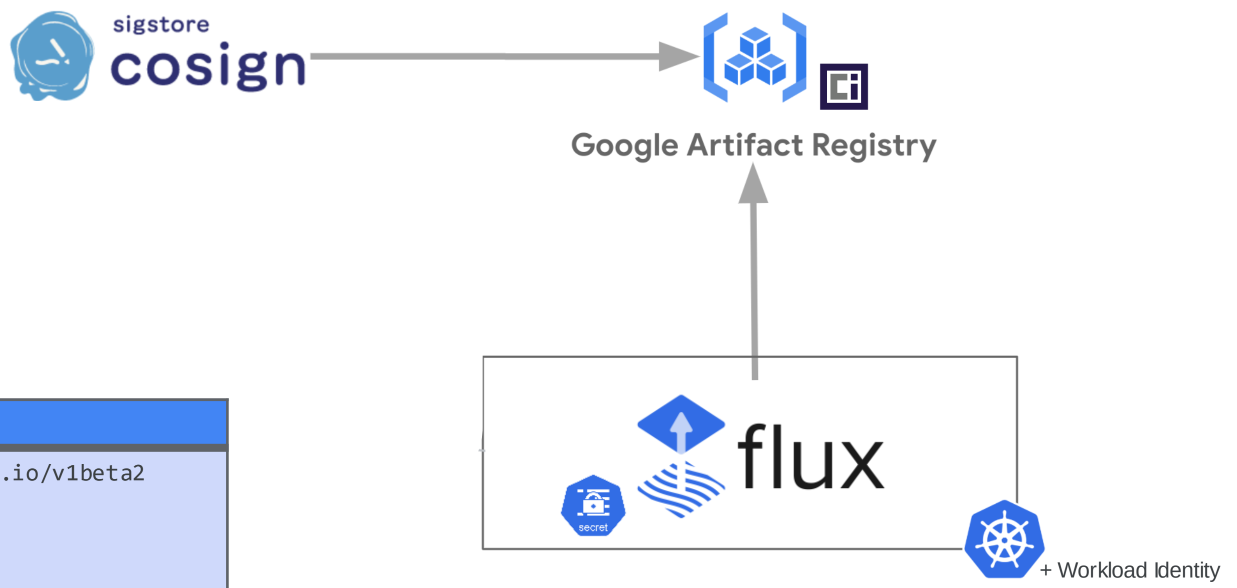
```
apiVersion: source.toolkit.fluxcd.io/v1beta2
kind: HelmRepository
metadata:
  name: my-helm-registry
spec:
  type: oci
  interval: 5m
  provider: gcp
  url: oci://${HELM_REPO}
---
apiVersion: source.toolkit.fluxcd.io/v1beta2
kind: HelmChart
metadata:
  name: my-helm-chart
spec:
  verify:
    provider: cosign
    secretRef:
      name: cosign-pub
```

HelmChart / OCIRepository - KMS Support for spec.verify · Issue #1060

Sign an OCI image

```
oras push ${OCI_IMAGE}

cosign generate-key-pair

cosign sign \
    --key cosign.key \
    ${OCI_IMAGE}

cosign verify \
    --key cosign.pub \
    ${OCI_IMAGE}
```





oci-repository.yaml

```
apiVersion: source.toolkit.fluxcd.io/v1beta2
kind: OCIRepository
metadata:
  name: my-oci-image
spec:
  interval: 5m
  url: oci://${IMAGE}
  provider: gcp
  ref:
    semver: "*"
  verify:
    provider: cosign
    secretRef:
      name: cosign-pub
```

Link

HelmChart / OCIRepository - KMS Support for spec.verify · Issue #1060

1. Sign
2. Verify signature with **Sigstore policy-controller** and **Kyverno**
3. Sign
4. Verify signature with **Flux**
5. Sign
6. Verify signature with **Flux**

That's a wrap!

Complementary resources

- Out of scope of this presentation:
  - BinAuthz (Kritis/Grafeas)
  - Kubewarden and Connaisseur
  - Portieris (Notary) or Ratify/Gatekeeper (Notary V2)

- More context:
  - OCI Artifacts Explained
  - Signature Formats
  - Sigstore Or: How We Learned to Stop Trusting Registries and Love Signatures

- Sigstore not just for Open Source projects:
  - A Guide to Running Sigstore Locally
  - The Road to SLSA4 – Applying the Sigstore Ecosystem in a Corporate Environment
  - Using Sigstore to meet FedRAMP Compliance at Autodesk

- Gatekeeper + Cosign is not working yet... needs contributors

Mathieu Benoit

Medium | Blog | LinkedIn

Thanks!