SUPPLYCHAIN SECURITYCON @ THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

April 10th, 2023

Securing Kubernetes manifests with Sigstore Cosign, what are your options?

Mathieu Benoit

Medium | Blog | LinkedIn

#ossummit

THE LINUX FOUNDATION

Securing Kubernetes manifests with Sigstore Cosign, what are your options?

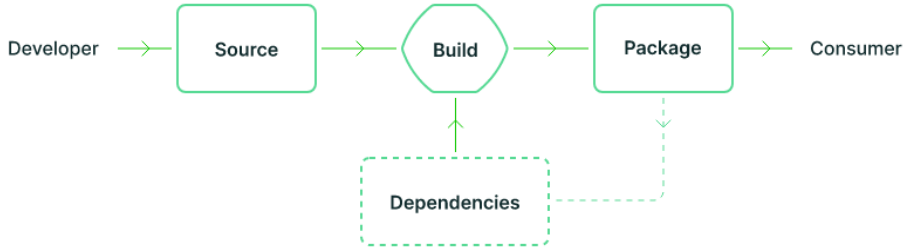
In this talk, we will explore the options to verify with Sigstore Cosign the provenance of Kubernetes manifests before actually being applied in your cluster.
Attendees will learn how Sigstore Cosign integrates with Kubernetes to provide secure solutions for signing and verifying container images and resource manifests, configuration files, and other critical components, bundled as generic OCI images.
We will also touch upon the use of GitOps tools like Flux and policy engines like Kyverno in combination with Sigstore Cosign to enforce security policies and prevent unwanted changes in your cluster.
Whether you are a seasoned Kubernetes user or just starting out, this talk will provide valuable insights and tips about your options for verifying in Kubernetes your Kubernetes manifests signed by Sigstore Cosign.

THE LINUX FOUNDATION

Agenda

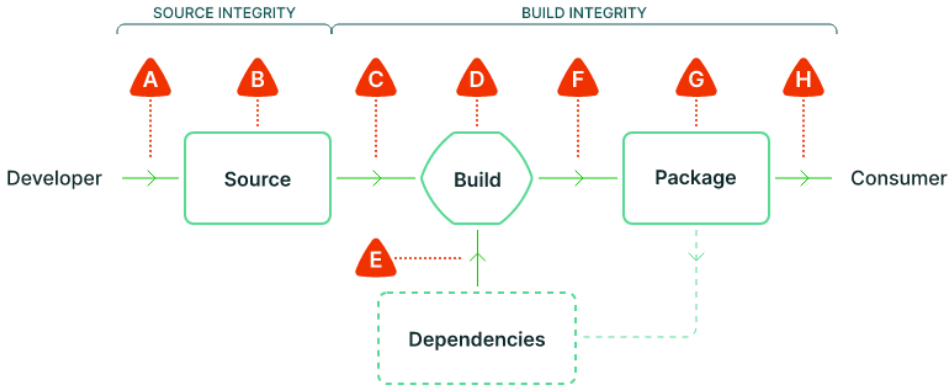
1. **Cosign** for Container images
2. **Kyverno** for Kubernetes manifests
3. **Flux** for Helm charts
4. **Flux** for OCI images

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				✓

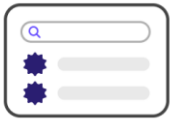
Sigstore

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.

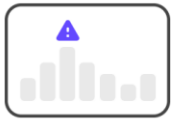
Sign code

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.



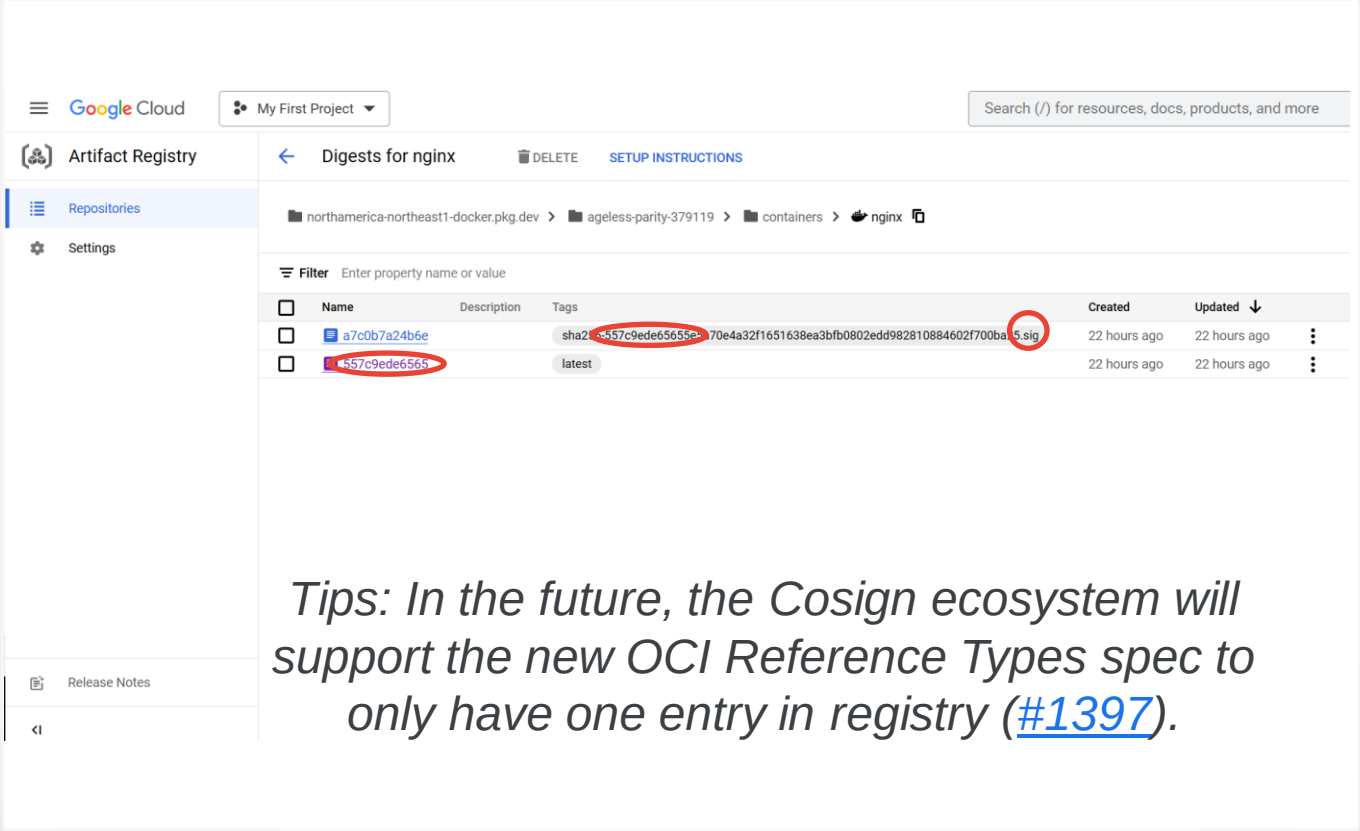
Monitor activity

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

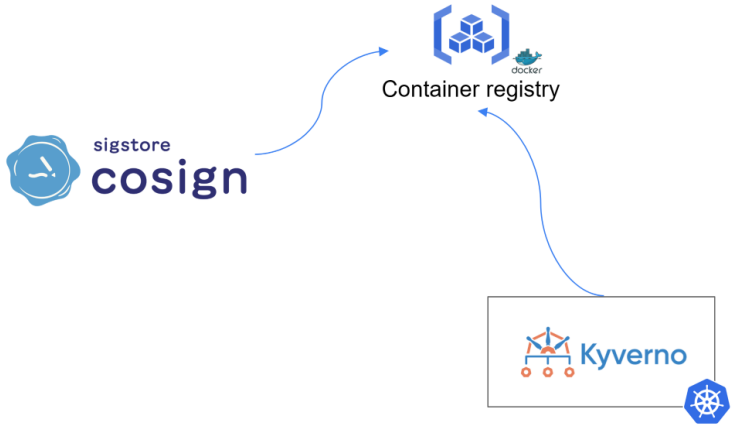
1. **Cosign** for Container images

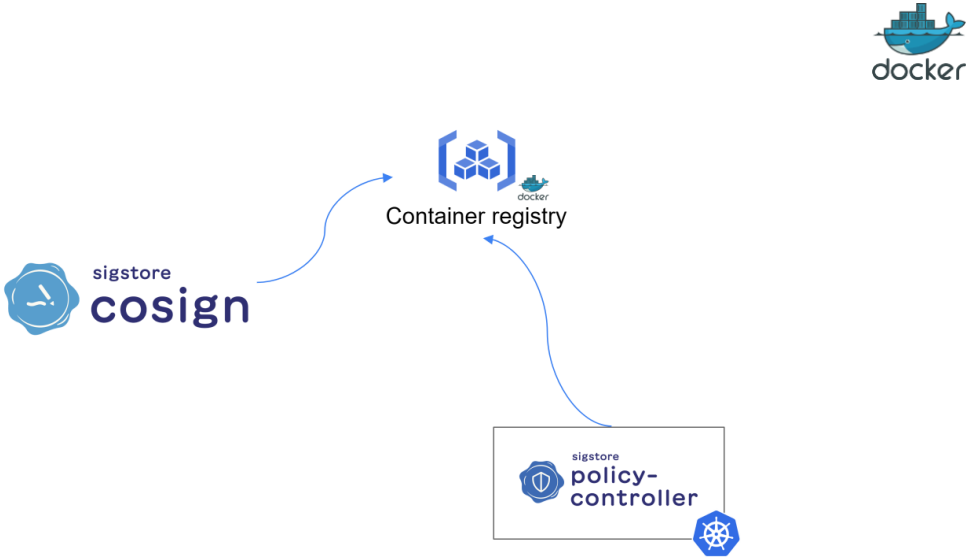
TERMINAL

```
docker build -t ${CONTAINER_IMAGE} .
docker push ${CONTAINER_IMAGE}

cosign generate-key-pair

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

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



Tips: In the future, the Cosign ecosystem will support the new OCI Reference Types spec to only have one entry in registry (#1397).





cluster-policy.yaml

```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:
              secret:
                name: cosign-pub
```

cluster-image-policy.yaml

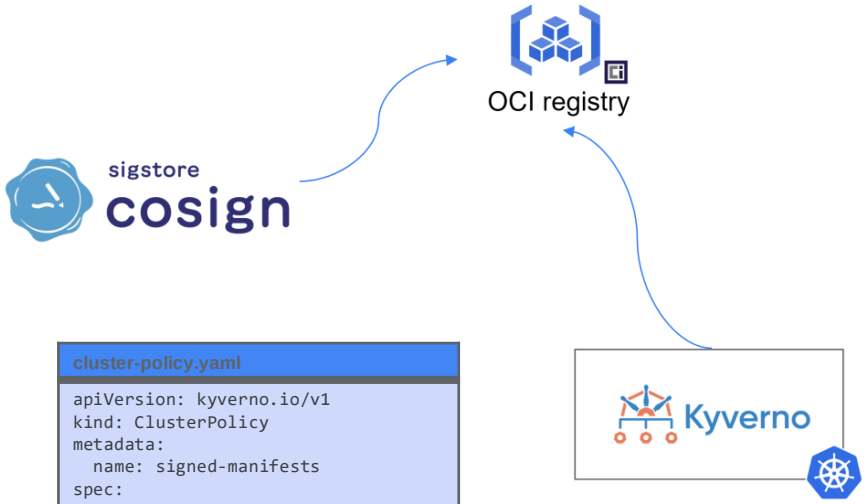
```yaml
apiVersion: policy.sigstore.dev/v1alpha1
kind: ClusterImagePolicy
metadata:
  name: private-signed-images-cip
spec:
  images:
  - glob: "**"
  authorities:
  - key:
      secret:
        name: cosign-pub
```

2. **Kyverno** for Kubernetes manifests

TERMINAL

```
kubectl sigstore sign \
    -f foo.yaml \
    --image ${OCI_IMAGE} \
    --key cosign.key

kubectl sigstore verify \
    -f foo.yaml \
    --image ${OCI_IMAGE} \
    --key cosign.pub
```

sigstore/k8s-manifest-sigstore: kubectl plugin for signing Kubernetes manifest



cluster-policy.yaml

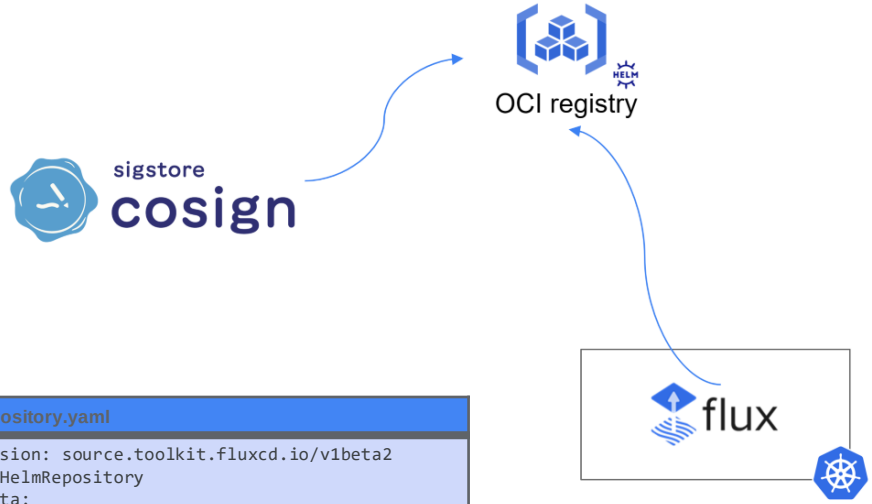
```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: signed-manifests
spec:
  validationFailureAction: Enforce
  background: true
  rules:
  - name: signed-manifests
    match:
      any:
      - resources:
          kinds:
          - Deployment
    validate:
      - manifests:
          attestors:
          - count: 1
            entries:
            - keys:
                secret:
                  name: cosign-pub
          ignoreFields:
          - objects:
            - kind: Deployment
            fields:
            - spec.replicas
```

3. **Flux** for Helm charts

TERMINAL

```
helm package ${HELM_CHART_NAME}   # --sign (#10644)
helm push oci://${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



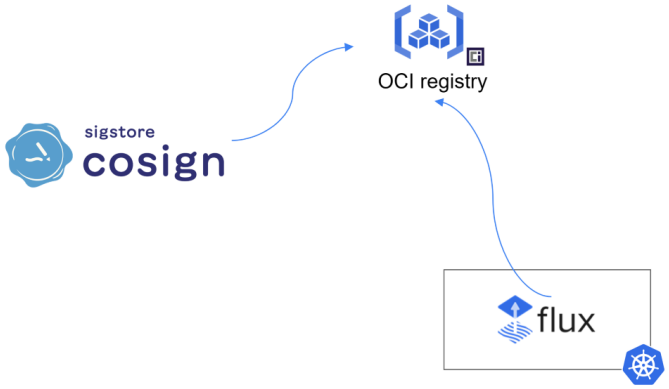
oci-repository.yaml

```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
```

4. **Flux** for OCI images

TERMINAL

```
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

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

That’s a wrap!

We demonstrated how to verify the Cosign signature of your Kubernetes manifests.

3 options were illustrated:

1. **Kyverno** for Kubernetes manifests
2. **Flux** for Helm charts
3. **Flux** for OCI images

Resources

- My first experience with Kyverno
- Cosign and Policy-controller with GKE, Artifact Registry and KMS
  - Associated talk
- Build and Deploy Cloud Native (OCI) Artifacts, the GitOps Way
- Securing Kubernetes Manifests with Sigstore and Kyverno

SUPPLYCHAIN SECURITYCON

@

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

NORTH AMERICA

THE LINUX FOUNDATION