BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Trust But Verify: Bringing Supply Chain Integrity To CD GitOps

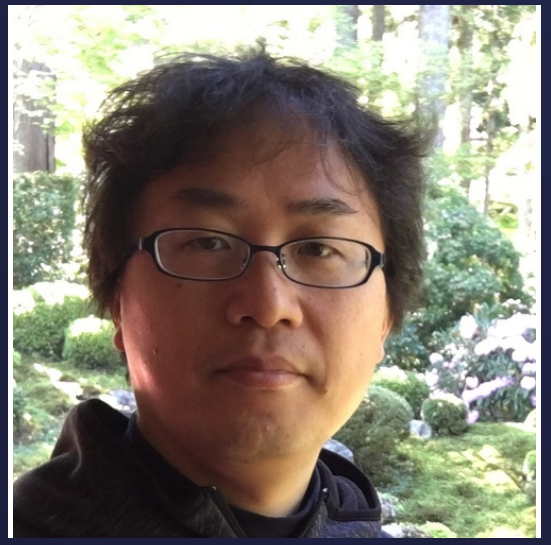
Yuji Watanabe & Hirokuni Kitahara, IBM Research

Trust But Verify: Bringing Supply Chain Integrity To CD GitOps

KubeCon | CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022

KubeCon | CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022

October 24-28, 2021



Yuji Watanabe
Senior Technical Staff Member
IBM Research

Hirokuni Kitahara
Research Scientist
IBM Research

Agenda

- End-to-end software supply chain integrity
- YAML manifest signature and enforcement
- Gaps in CD GitOps - manifest build
- ArgoCD Interlace – extension to fill the gap
- Demo

End-to-end supply chain

source

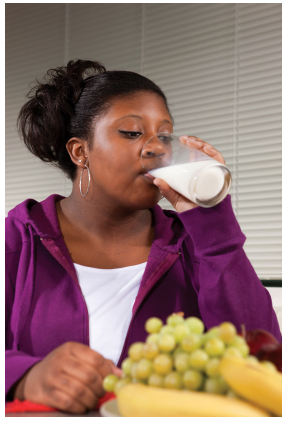
build

package

deliver

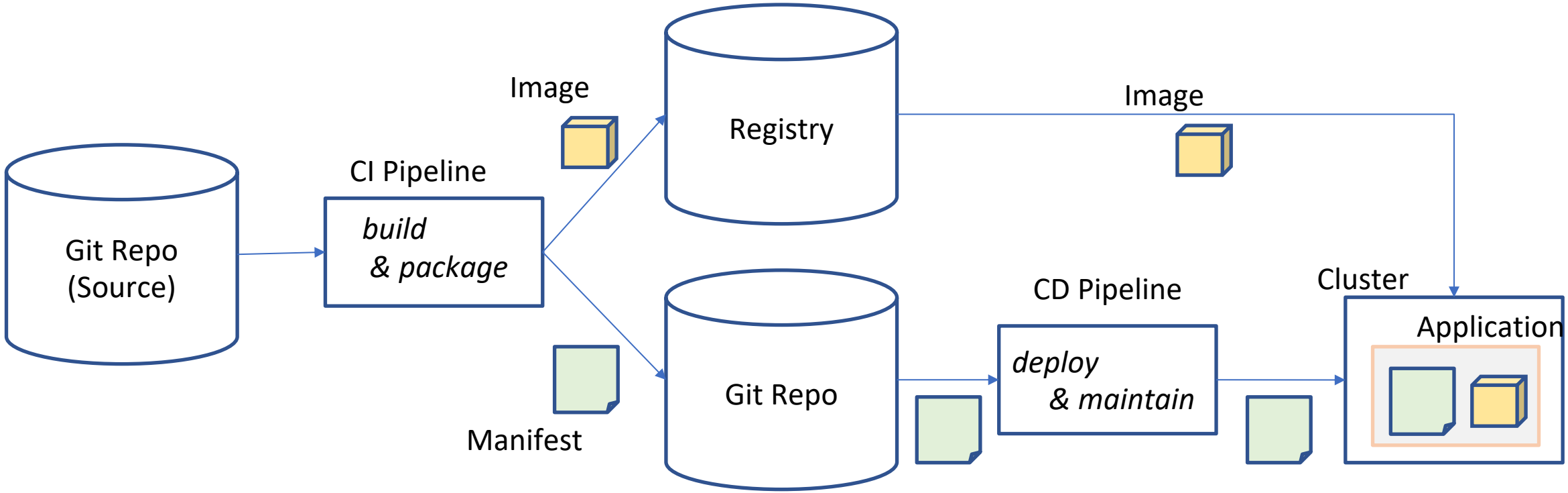
maintain

use

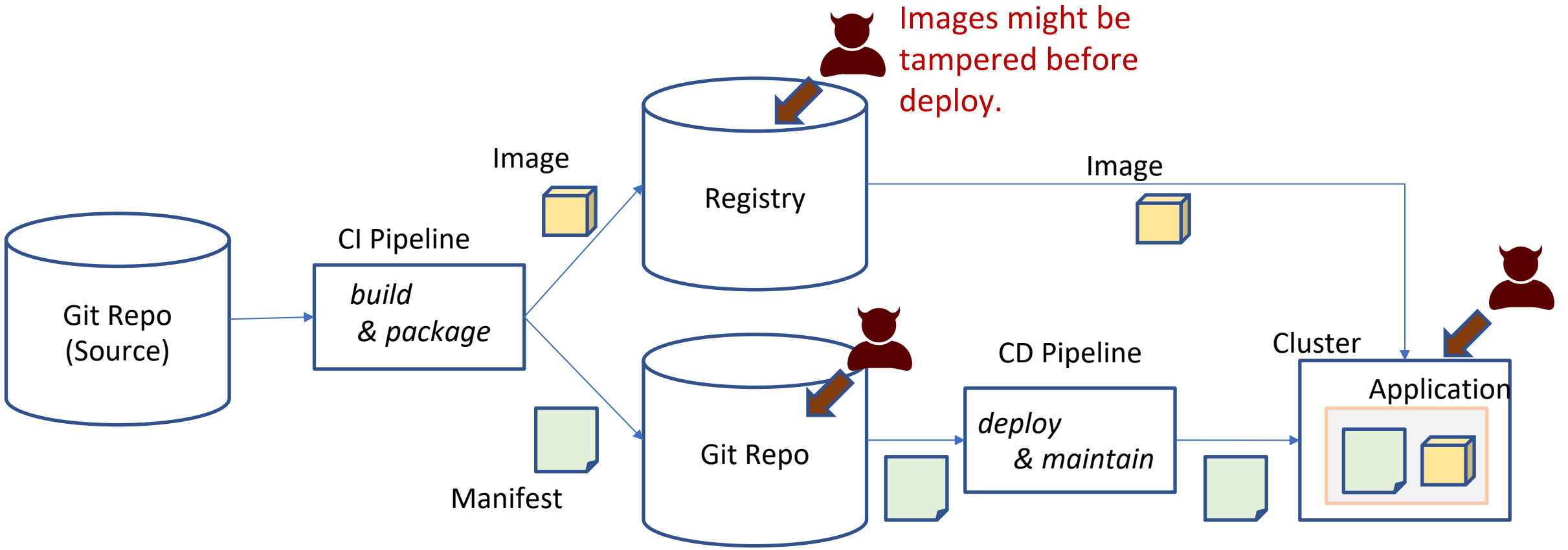


Delivery application to clusters

Images and YAML manifests are delivered to deploy an application.

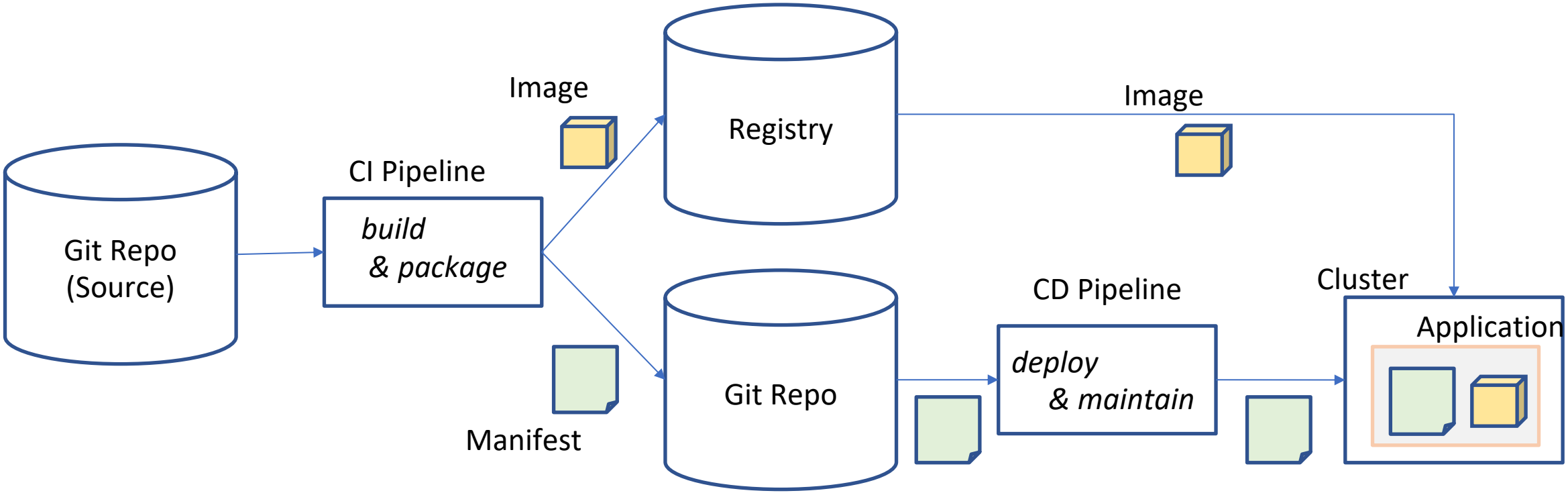


Integrity issues in delivery



End-to-end software supply chain integrity

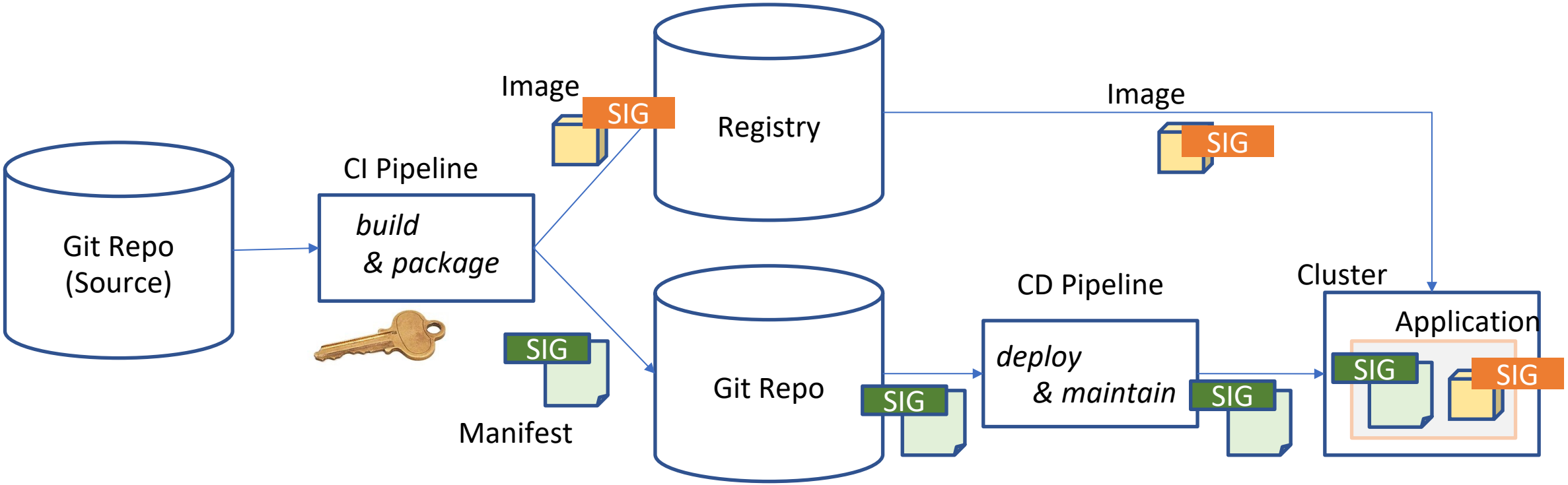
To protect delivery ...



End-to-end software supply chain integrity

To protect delivery ...

Use signature for integrity assurance and source verification of **manifests** and images



YAML manifest signing

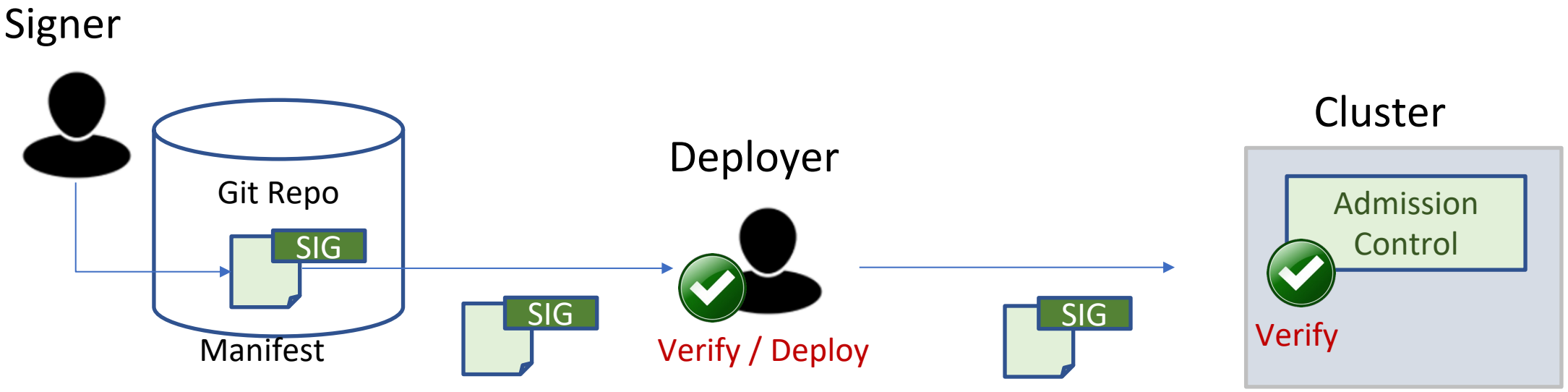
https://github.com/sigstore/k8s-manifest-sigstore

kubectl sigstore sign -f test-deployment.yaml -k cosign.key

```yaml
metadata:
  annotations:
    cosign.sigstore.dev/message: H4sIAAAAAAAA/wBZAab+H4sIAAAAAAAA/+ySu87bMAyFPfsp+A
    cosign.sigstore.dev/signature: MEUCIQCetBNqM7j4JBBgewjUIMsIPlR3gxKF+flZ8rGYBZE2
  labels:
    app: nginx
  name: test-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
```

Verify signed YAML manifests

- Both deployer and cluster side admission control can verify the signature.



protection during delivery

check at the time of admission time & even after

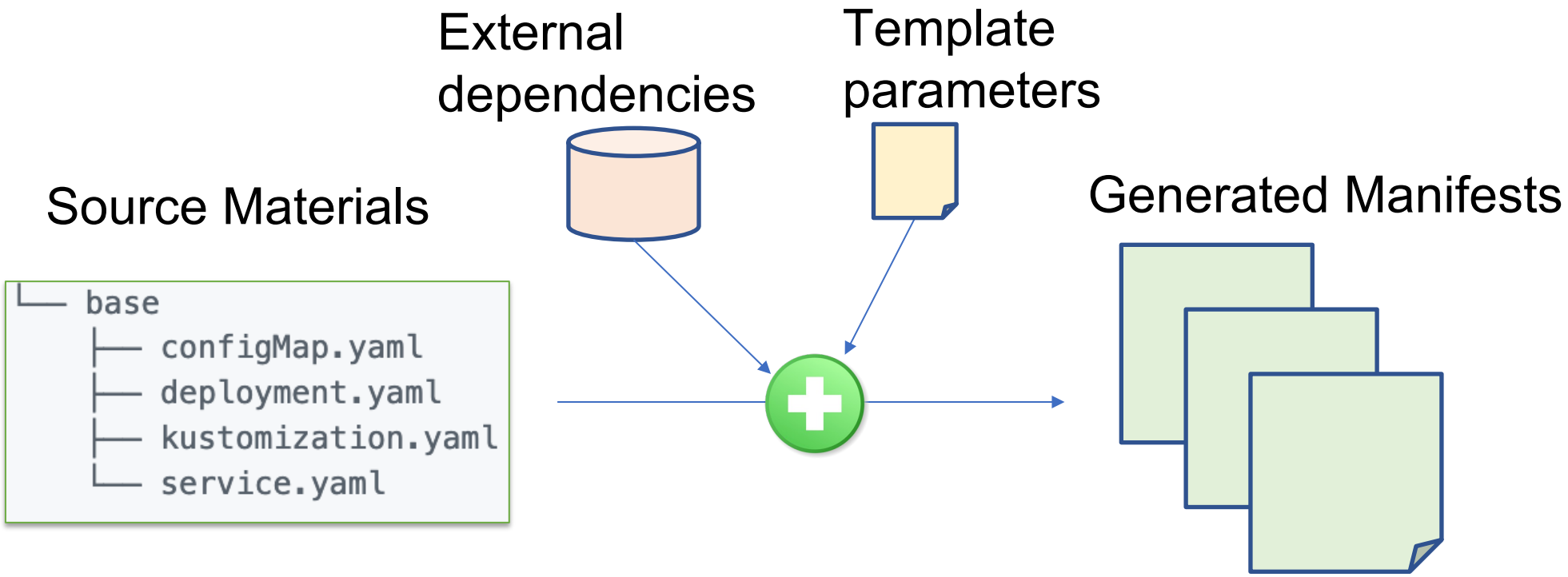
Admission control for signed YAML manifests

- Integrity Shield
  - OPA/Gatekeeper (PEP) + Integrity Shield API backend(PDP)
  - Signature verification at admission time + continuous check

- Kyverno
  - validate.manifests() to enable YAML manifest signature validation (v1.8.0-)

- You can enable protection of your YAML manifests with signature using k8s-manifest-sigstore tool via CLI or pipeline.

Manifest Build (kustomize)



kubectl apply –k <base_dir>

kustomize build <base_dir> | kubectl apply -f -

Nested dependency in manifest build

App repo (parent)

guestbook-ui-deployment.yaml

guestbook-ui-svc.yaml

kustomization.yaml

```
namePrefix: sample-kustomize-
commonLabels:
  project: sample-kustomize-app
resources:
- https://github.com/hirokuni-kitahara/sample-kustomize-base.git
- guestbook-ui-deployment.yaml
- guestbook-ui-svc.yaml
```

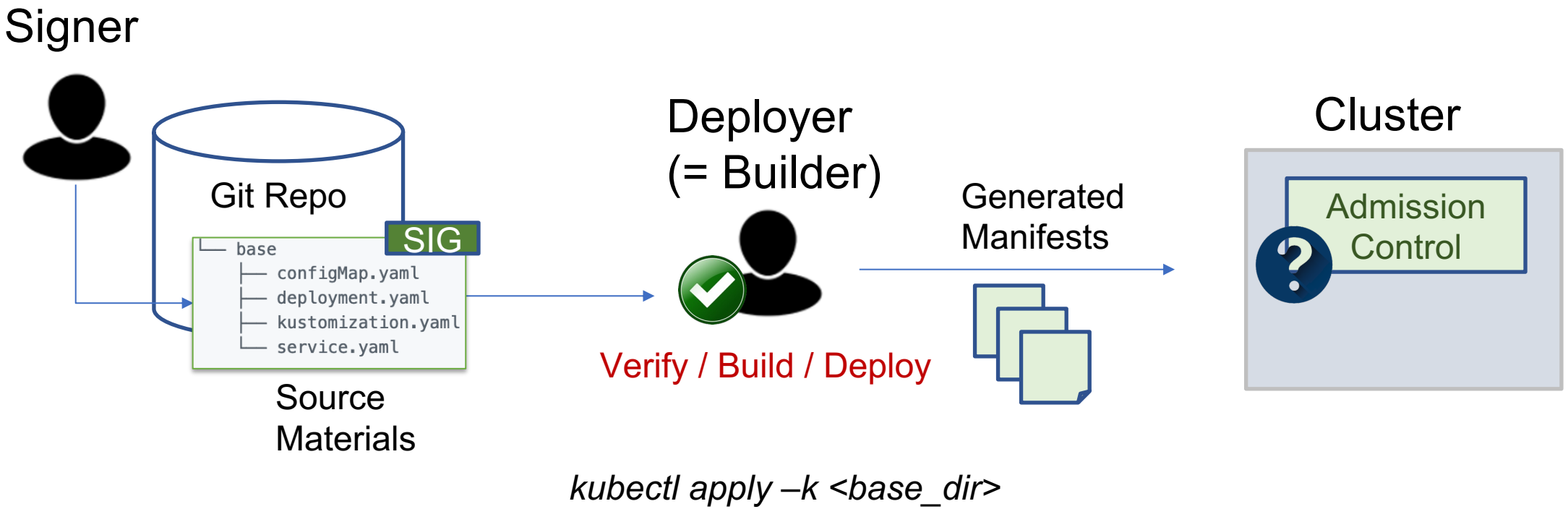
Config repo (child)

base-configmap.yaml

kustomization.yaml

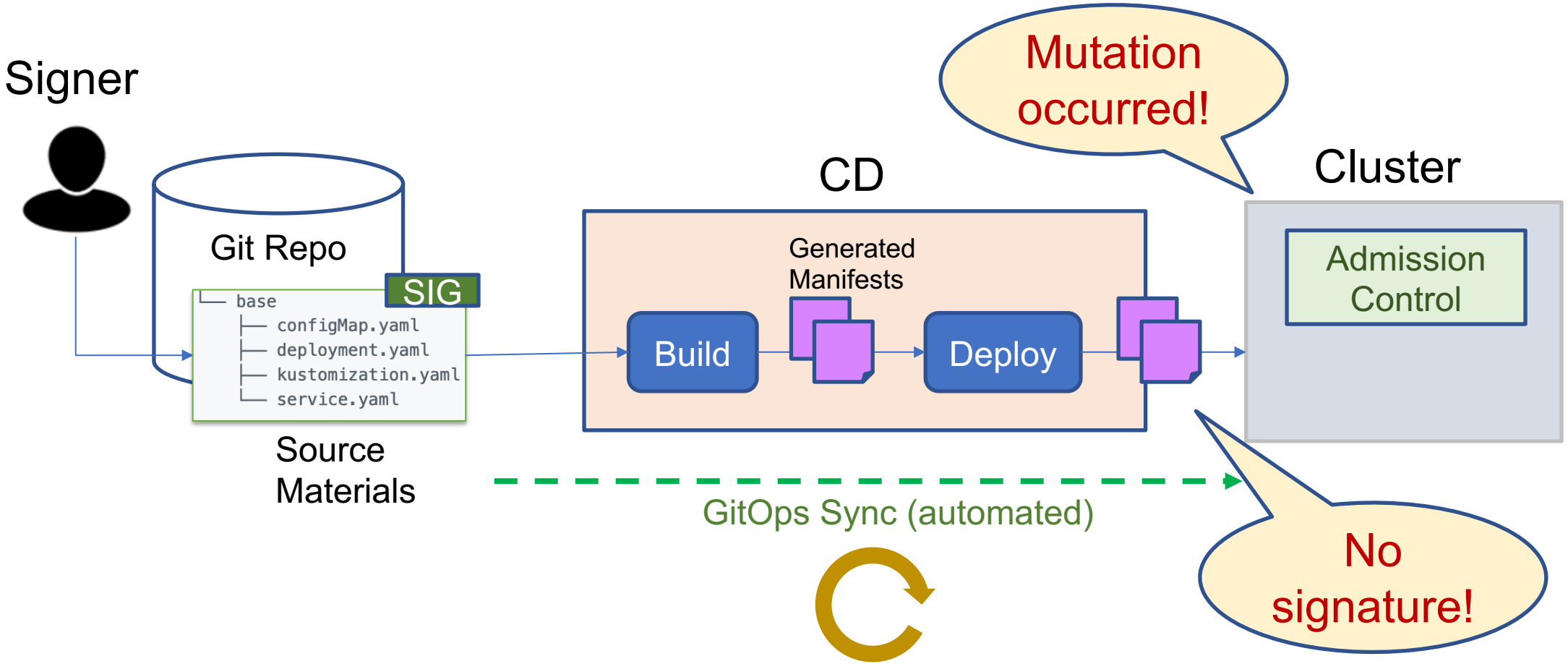
Cluster needs to rely on deployer

- Deployer can verify before build, but the cluster side cannot.
  → Cluster side need to rely on deployer's pre-check

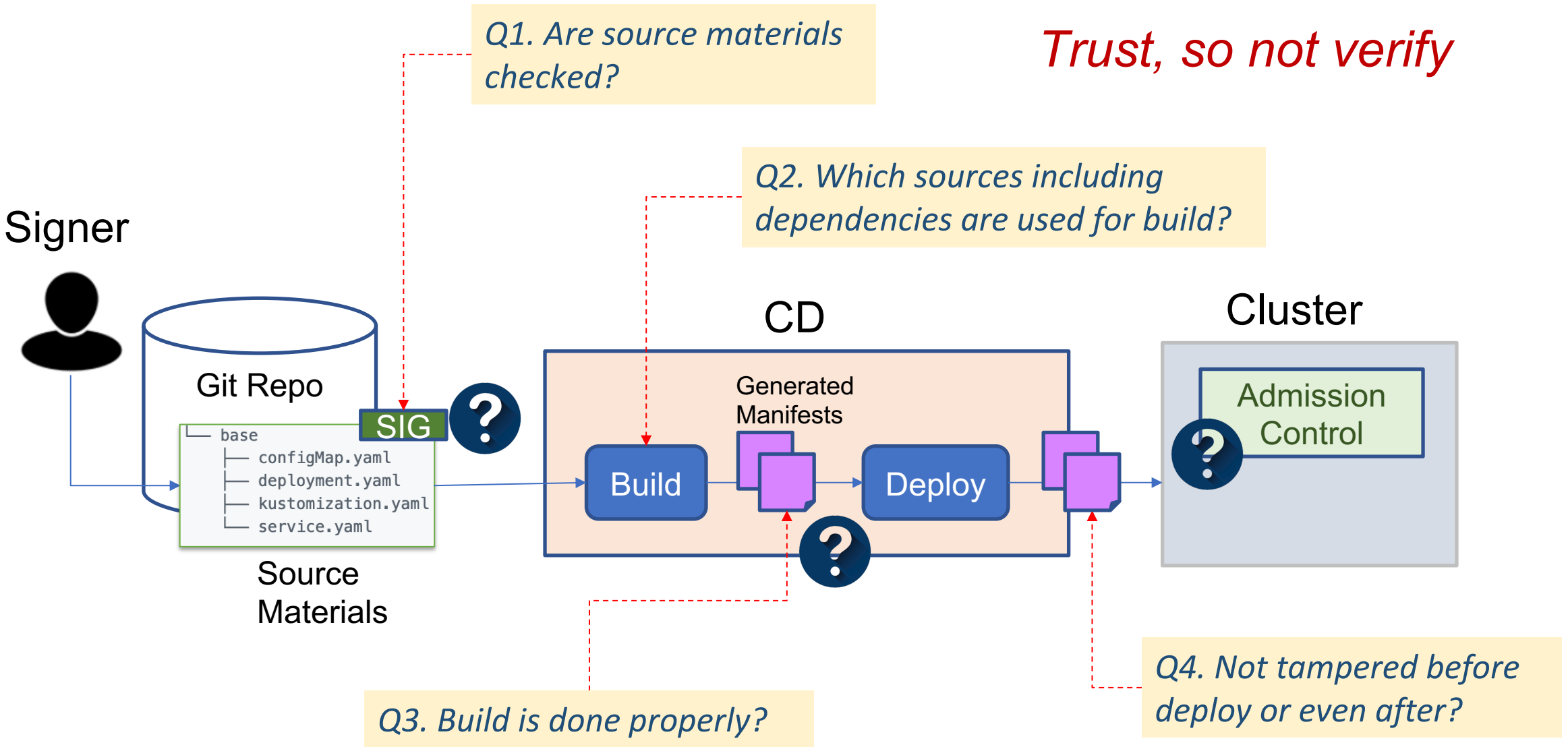


CD GitOps

Modern software deployment mutated the source materials before deploying final manifests.

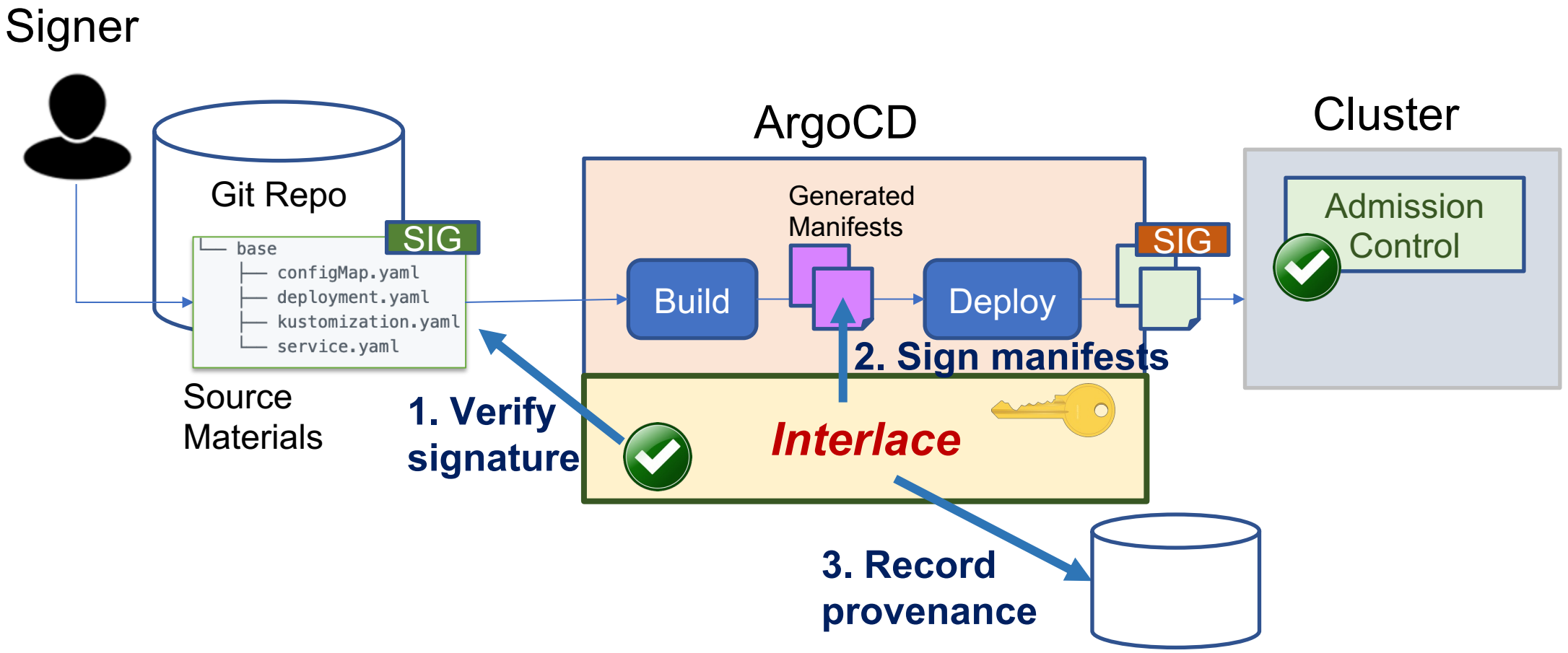


Gaps in CD GitOps



Interlace – "trust but verify" approach

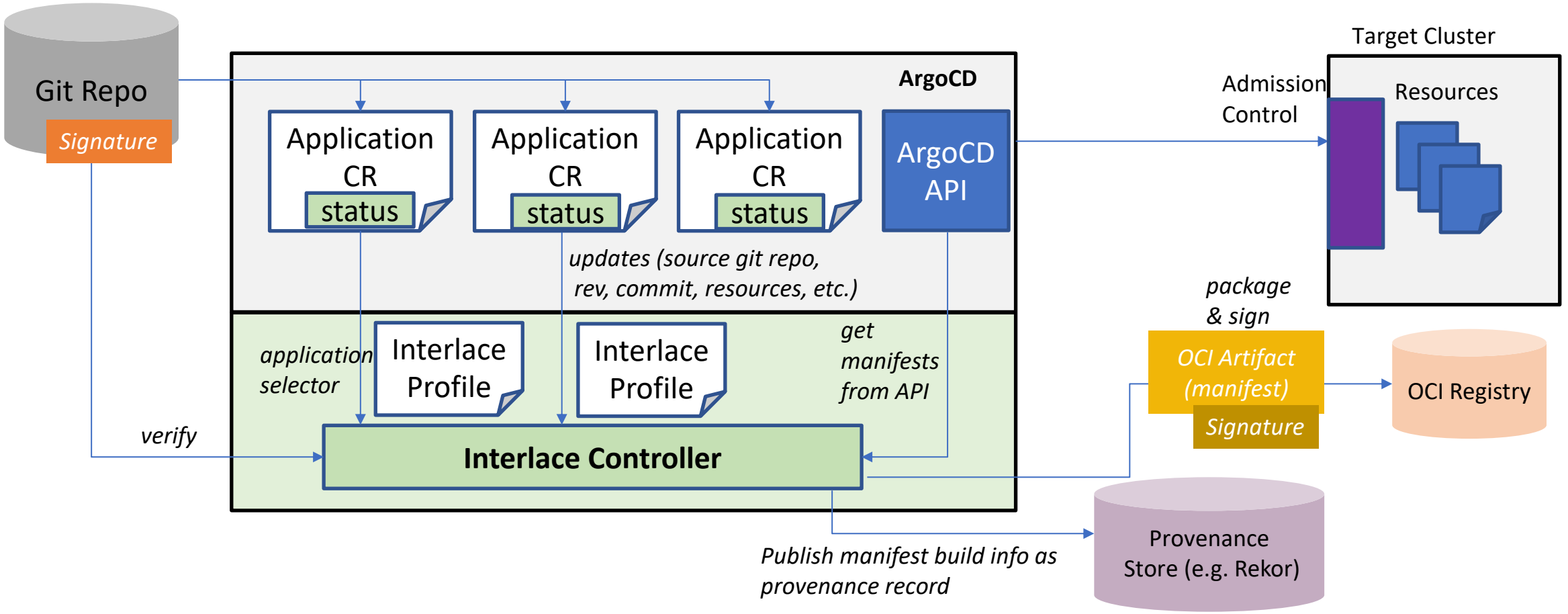
Interlace controller extends GitOps sync in ArgoCD by three functions - verify, sign, provenance.



Interlace

https://github.com/argoproj-labs/argocd-interlace

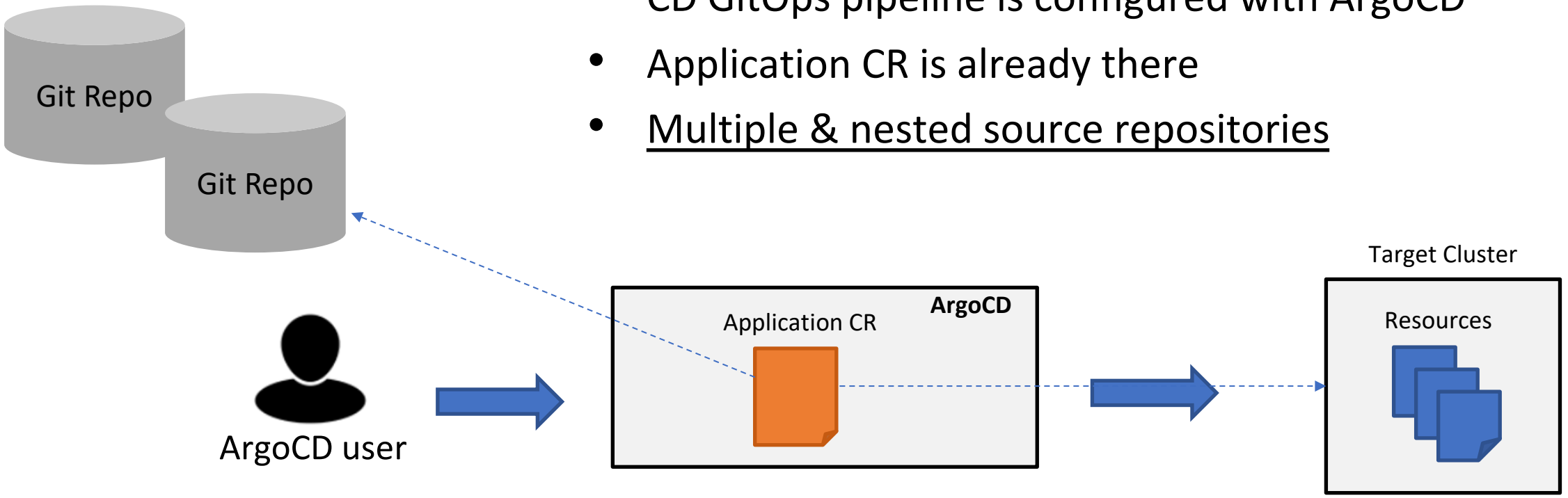
- An Application custom resource defines source (git repo) and target (cluster).
- ArgoCD observes source git repo, then auto-sync with target.
- Interlace detects the sync event and then do the verify/sign/provenance recording.



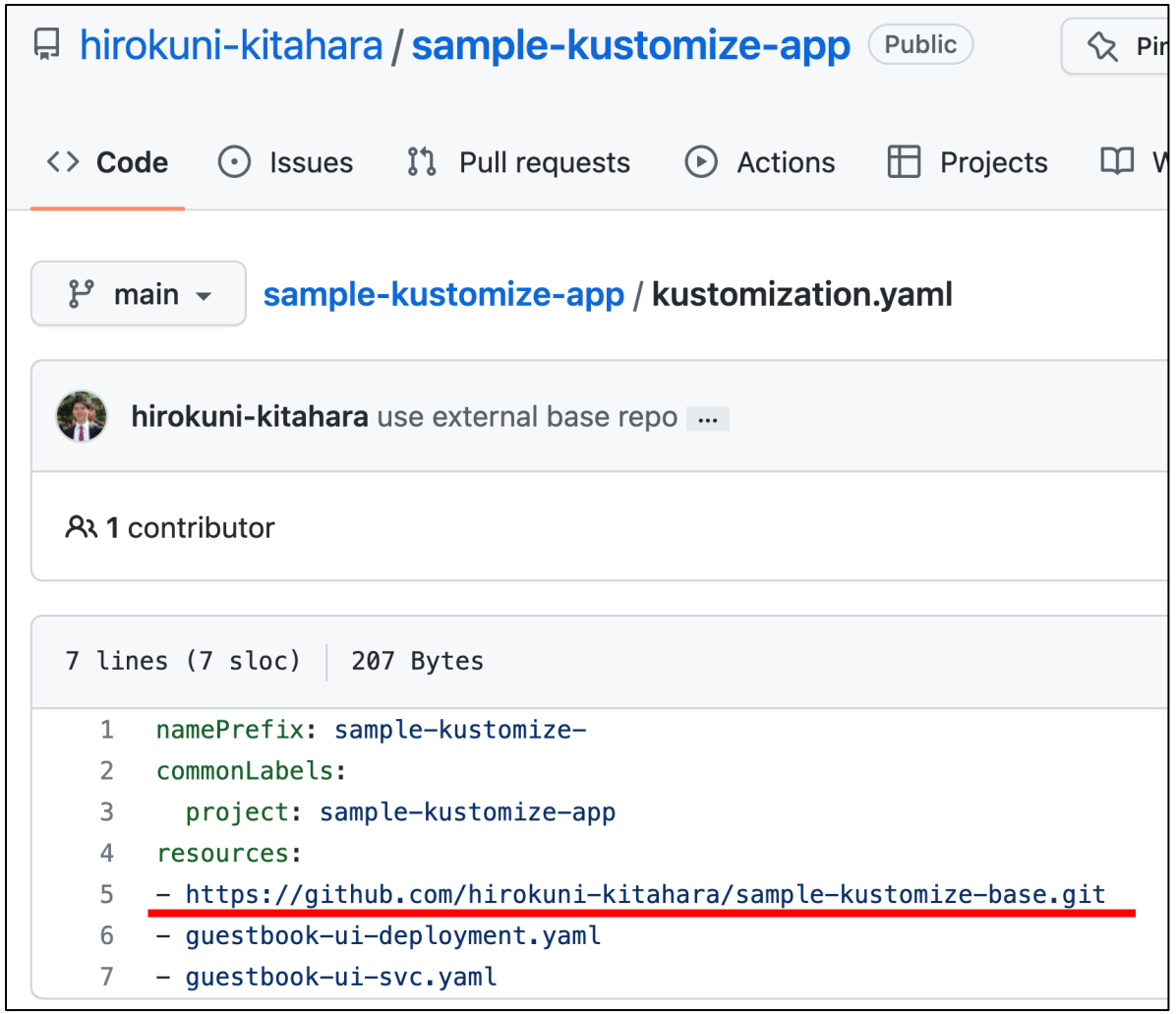
Demo

Demo Scenario

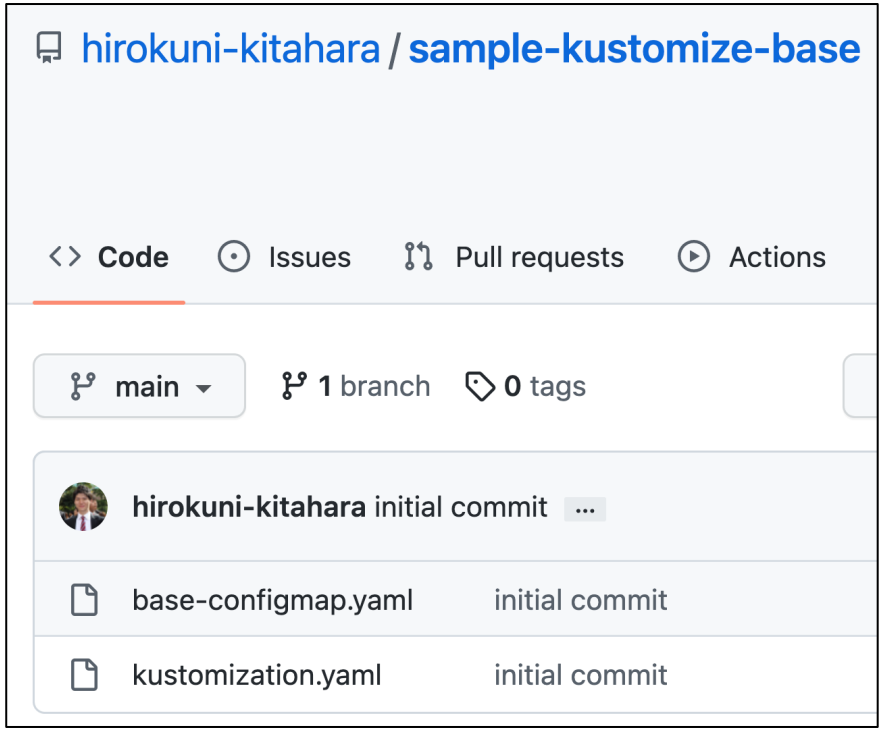
- CD GitOps pipeline is configured with ArgoCD
- Application CR is already there
- Multiple & nested source repositories



Source repositories in demo

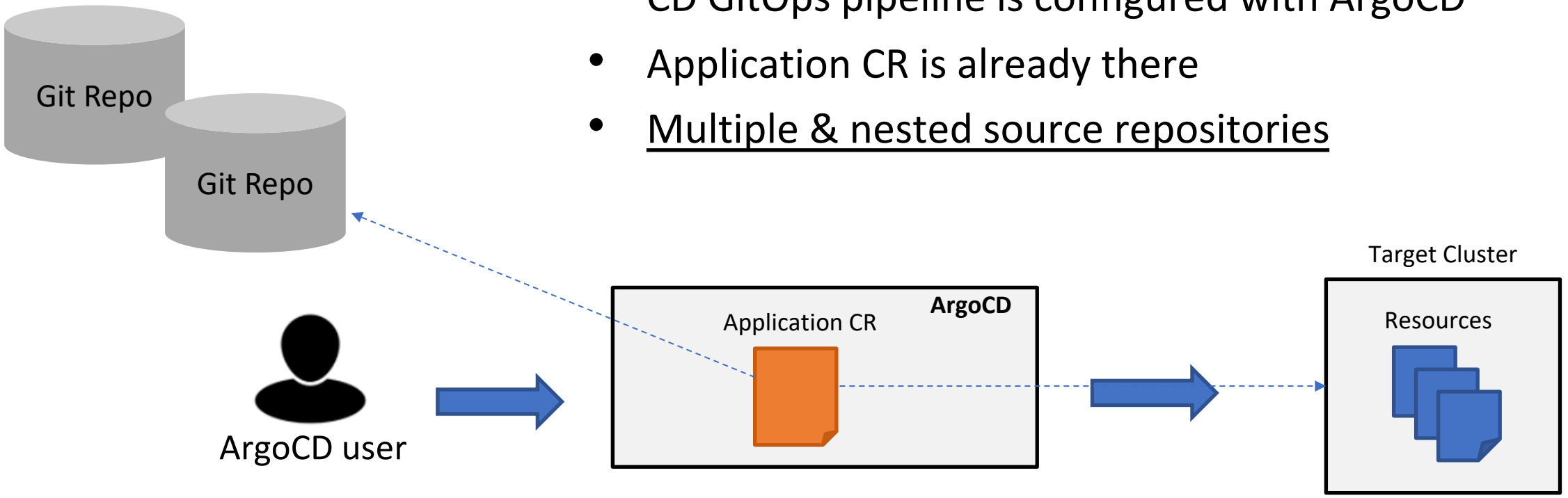


External dependency

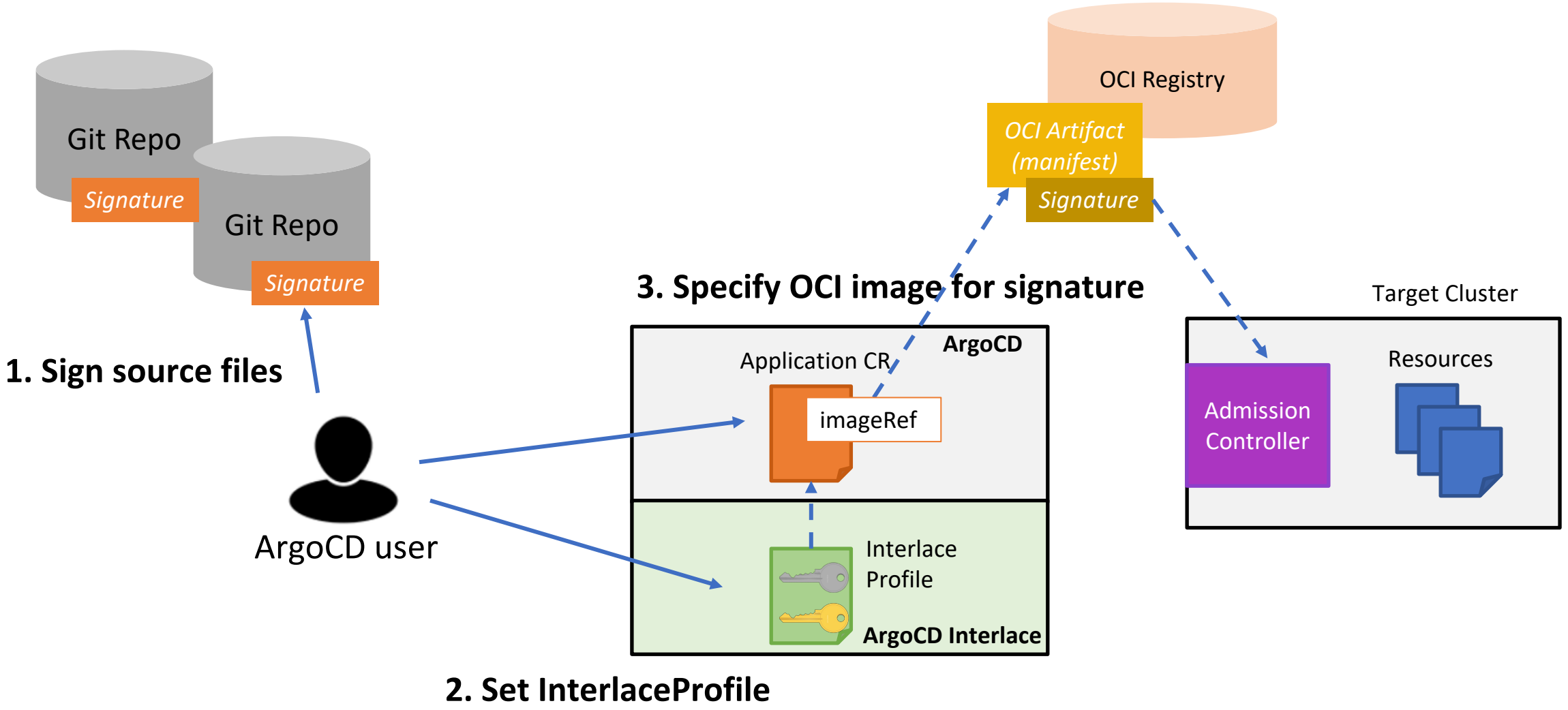


Demo Scenario

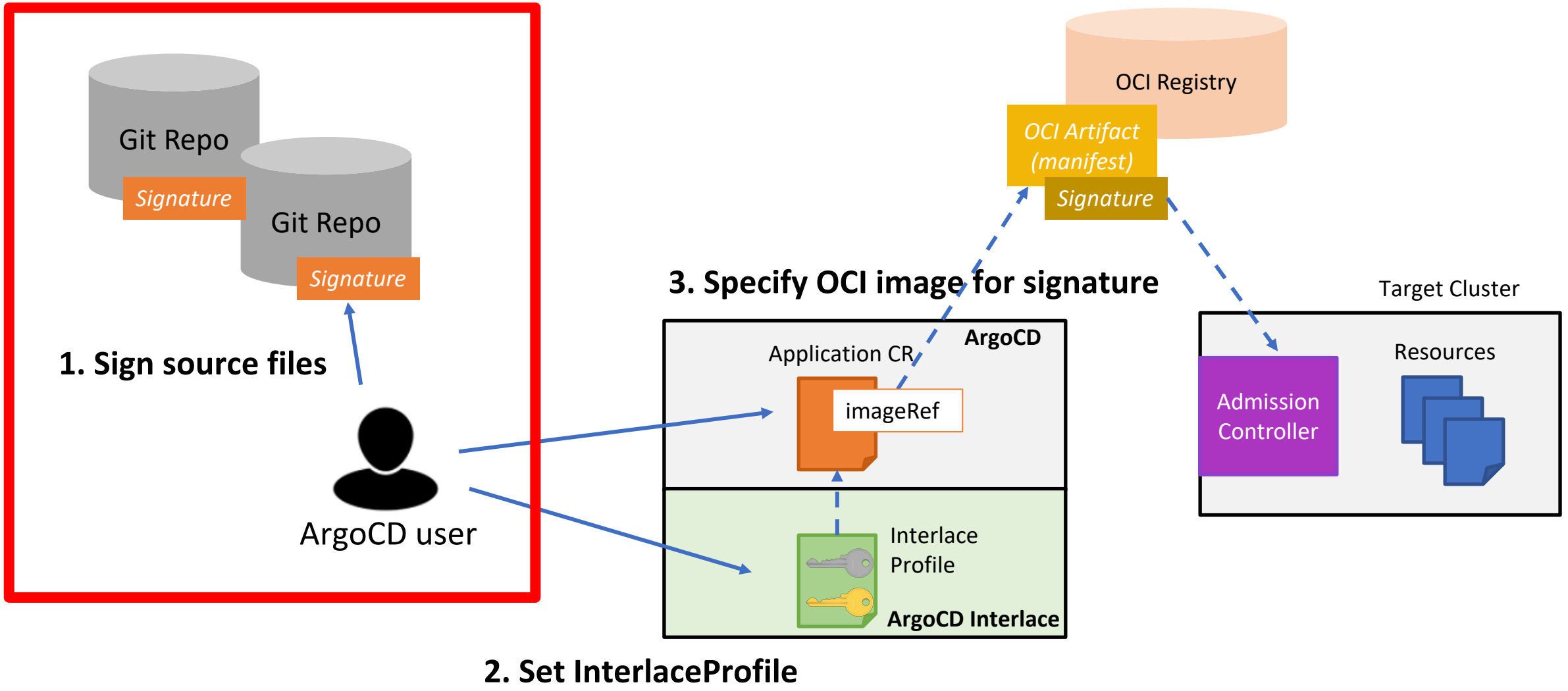
- CD GitOps pipeline is configured with ArgoCD
- Application CR is already there
- Multiple & nested source repositories



Demo Scenario



Demo Scenario



Sign source


```
$ git ls-tree -r HEAD --name-only | grep -v source-materials | xargs sha256sum > source-materials
```

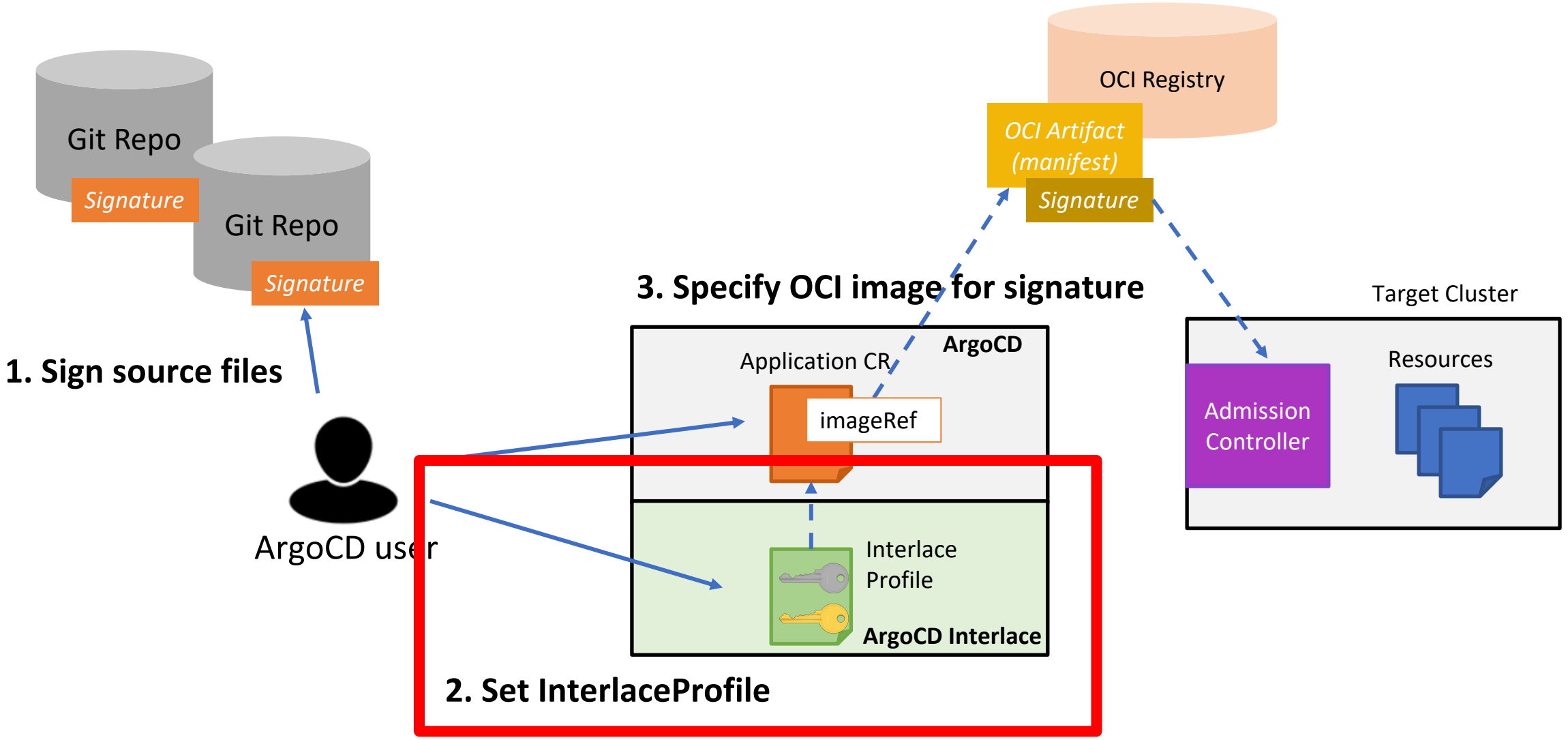
```
$ cosign sign-blob --key <SIGNING_KEY> --output-signature source-materials.sig source-materials
```

hirokuni-kitahara sign files ... 2972ead 2 minutes ago **85** commits

guestbook-ui-deploym...	reset	3 hours ago
guestbook-ui-svc.yaml	initial commit	14 months ago
kustomization.yaml	use external base repo	27 days ago
source-materials	sign files	2 minutes ago
source-materials.sig	sign files	2 minutes ago

Signature files

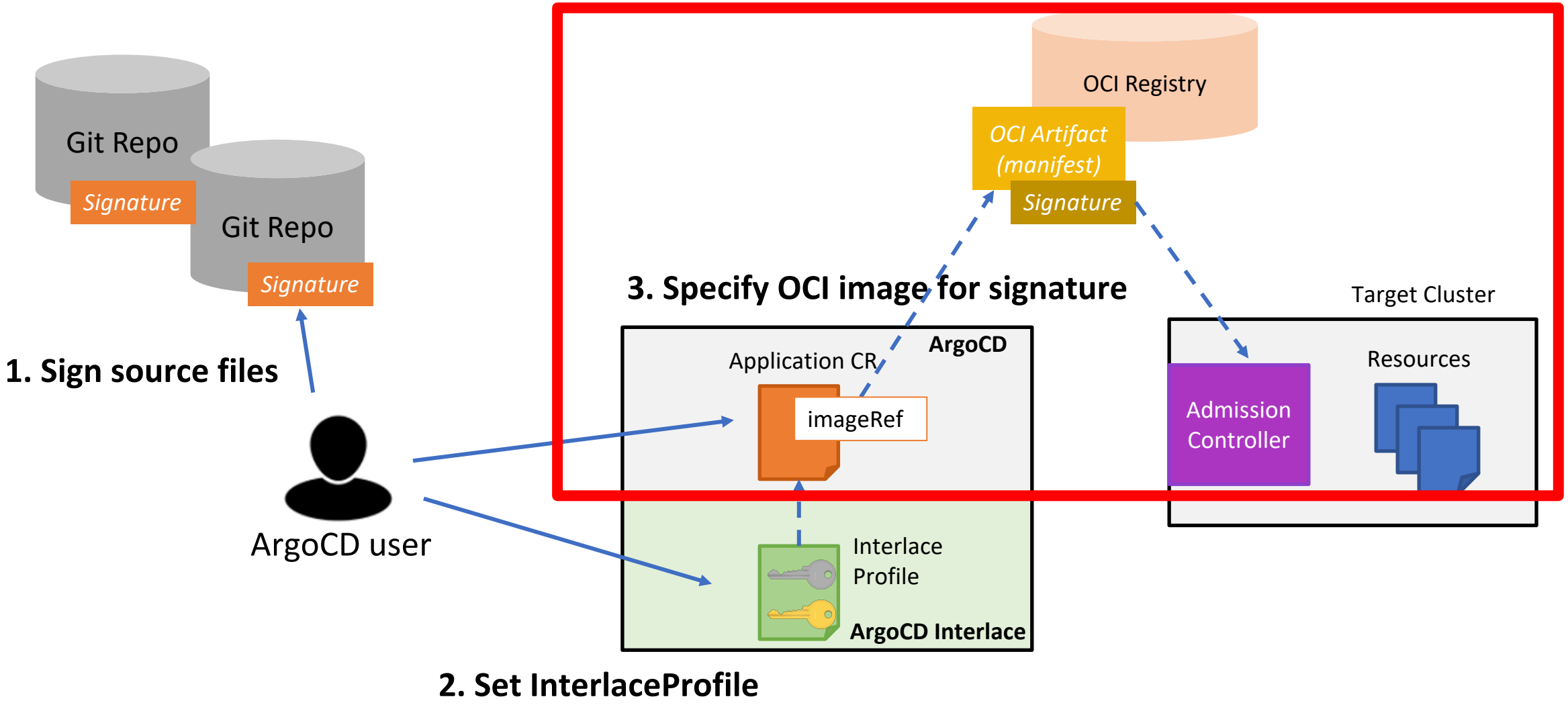
Demo Scenario



Set InterlaceProfile


```yaml
apiVersion: interlace.argocd.dev/v1beta1
kind: InterlaceProfile
metadata:
  name: kubecon-demo-profile
  namespace: argocd-interlace
spec:
  applicationSelector:
  - matchLabels:
      kubecon: 'demo'
  verifyConfig:
    key:
      PEM: |
        -----BEGIN PUBLIC KEY-----
        MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE5znfqEW3LfyMW
        jl7n/wWiZxWxtDskyTWFUgWYtILbpJ+j9lJqrPfBlmyp4LxtO
        -----END PUBLIC KEY-----
  signConfig:
    key:
      secret: kubecon-demo-sign-key
    match:
    - kind: Deployment
    - kind: Service
```

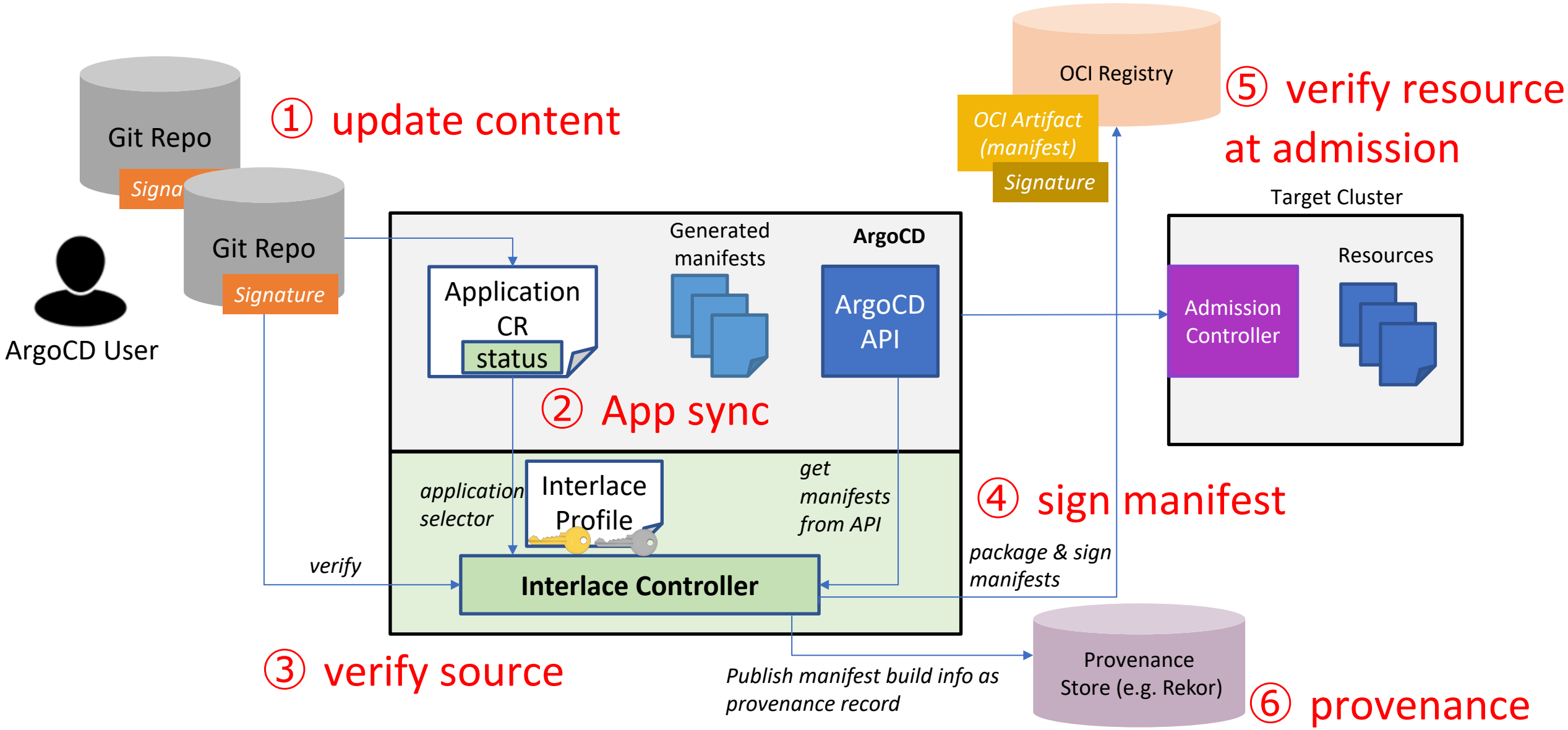
Demo Scenario



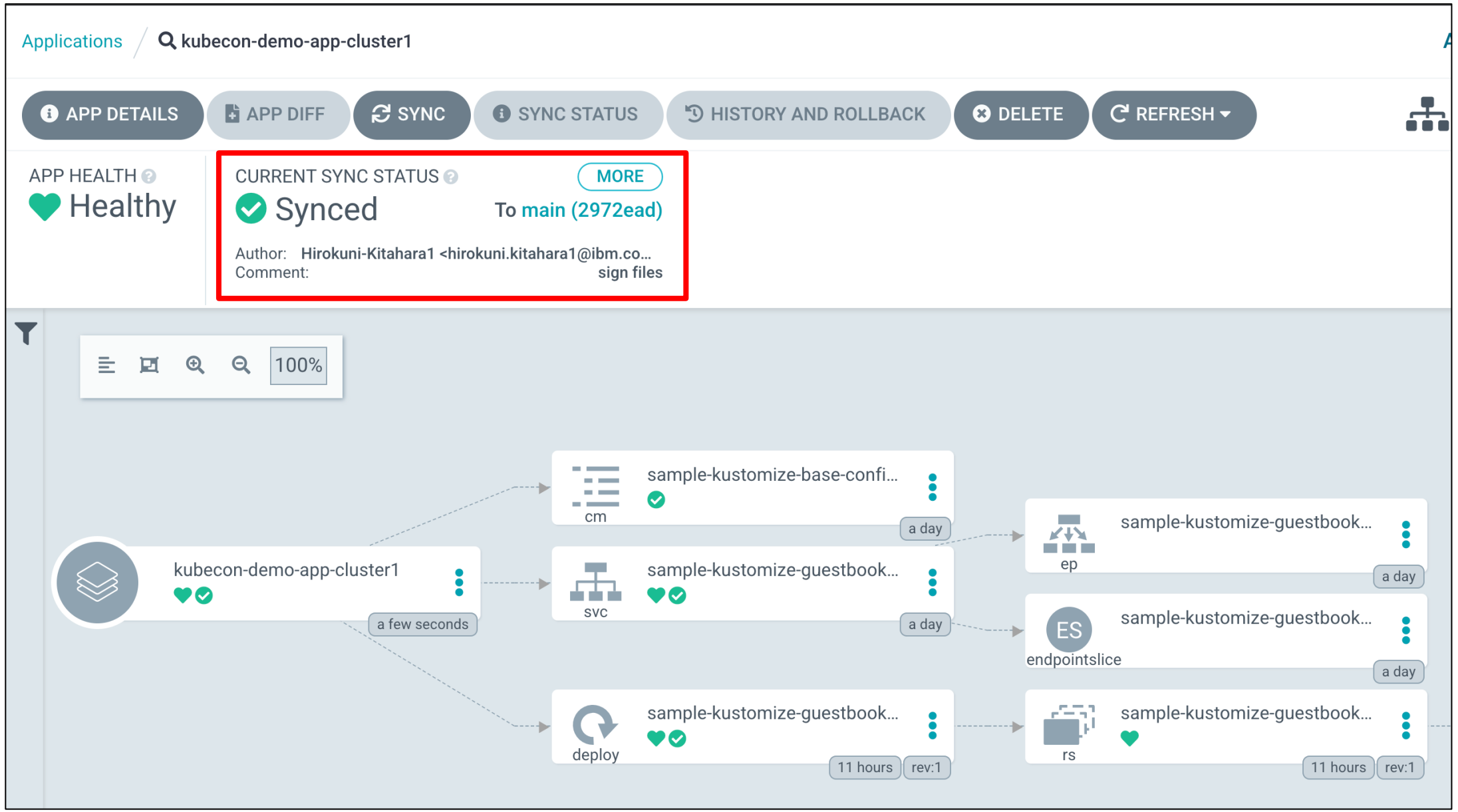
Specify OCI Image for signature


```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: kubecon-demo-app-cluster1
  namespace: argocd
  annotations:
    interlace.argocd.dev/manifestImage: ghcr.io/hirokuni-kitahara/kubecon-demo-app-manifest:dev
  labels:
    kubecon: demo
spec:
```

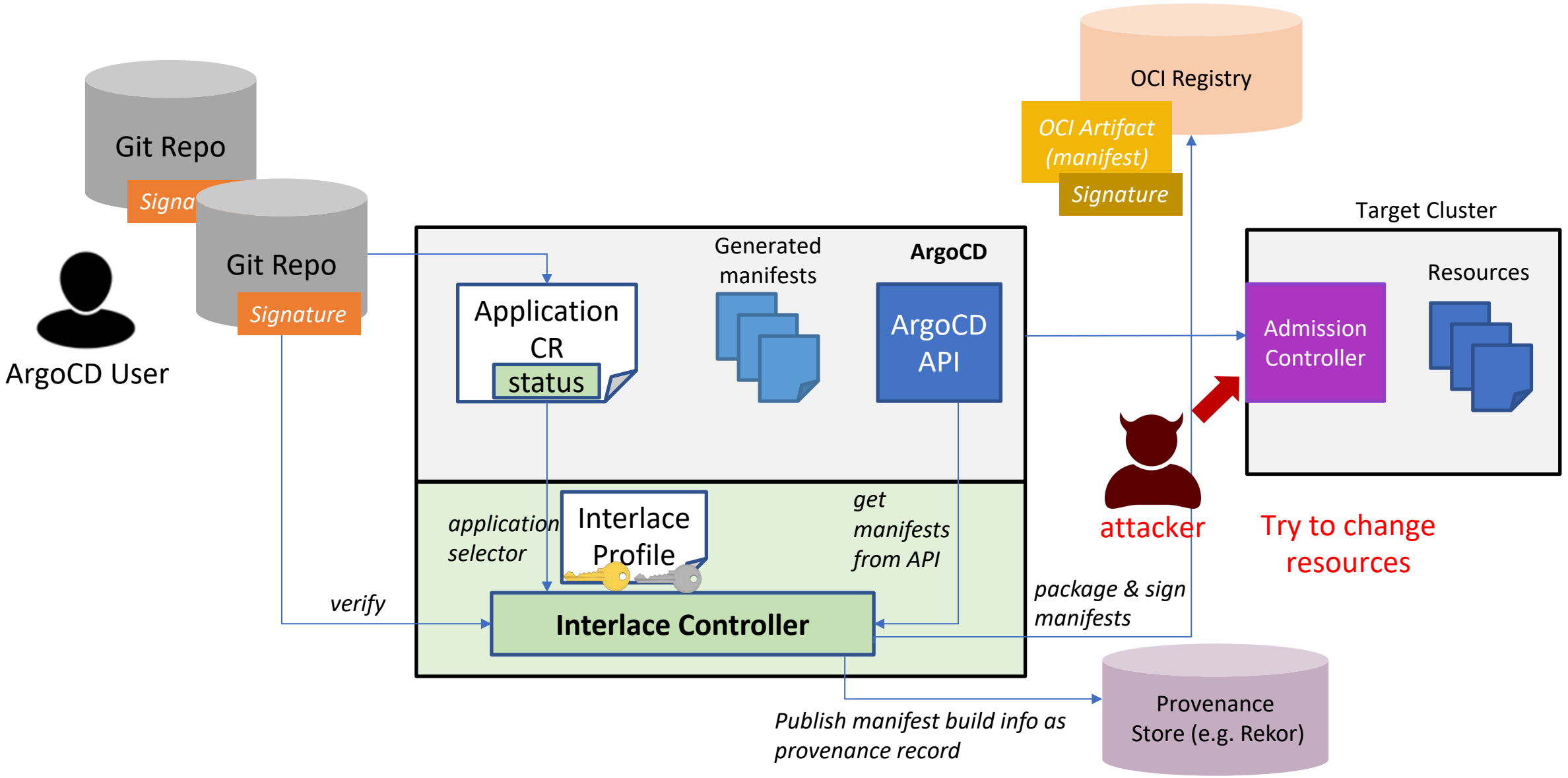
End-to-end signature protection



Successful deployment with valid signature



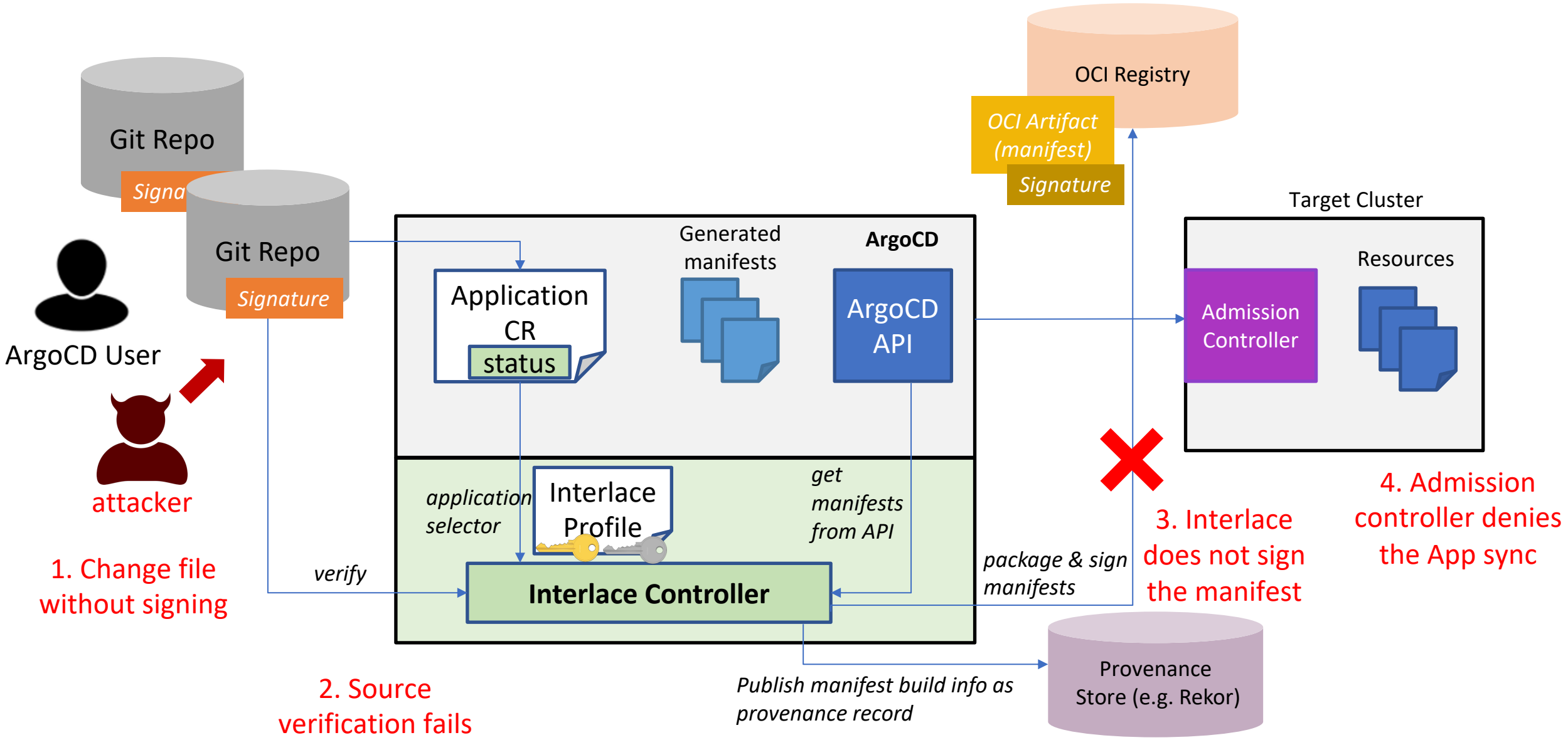
Unauthorized change to resource



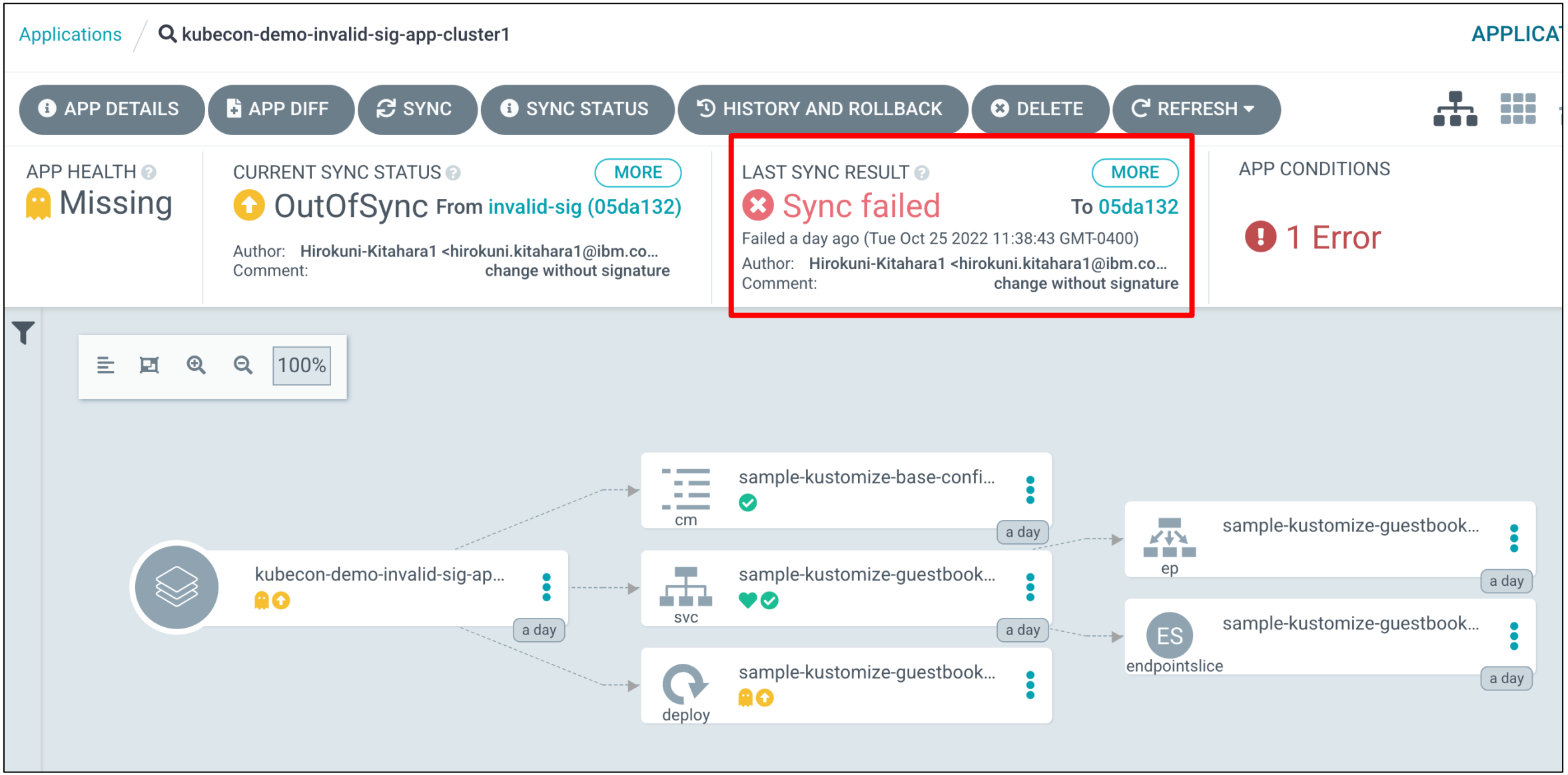
Blocked by admission controller


```
target-cluster:$ kubectl edit deploy -n kubecon-demo sample-kustomize-guestbo
ok-ui
error: deployments.apps "sample-kustomize-guestbook-ui" could not be patched:
 admission webhook "validation.gatekeeper.sh" denied the request: [enforce-ru
le-sample-profile-0] denied; {"allow": false, "message": "Signature verificat
ion is required for this request, but failed to verify signature. diff found:
 {\"items\":[{\"key\":\"spec.template.spec.containers.0.ports.0.containerPort
\",\"values\":{\"after\":12345,\"before\":80}}]}"}
You can run `kubectl replace -f /tmp/kubectl-edit-4020396247.yaml` to try thi
s update again.
```

Sync failed because Interlace does not sign



Sync failed because Interlace does not sign

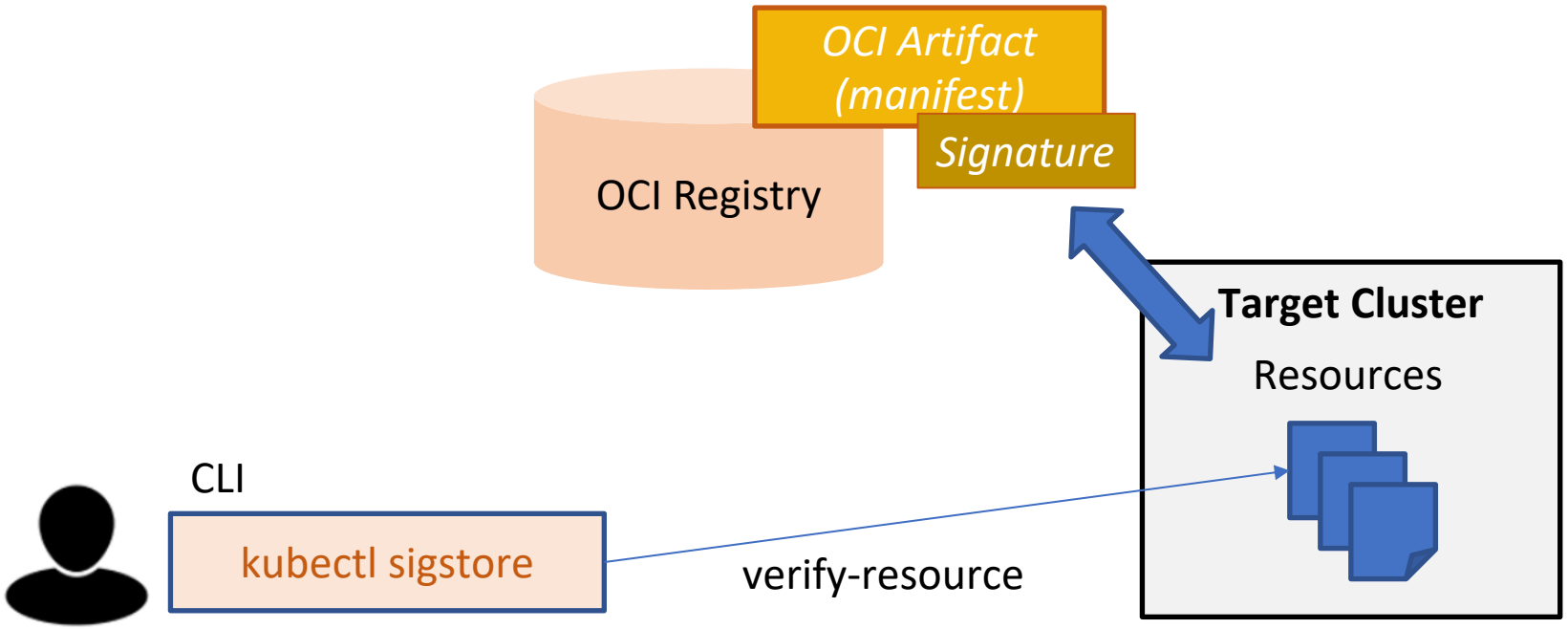


Verify resource with OCI artifact at cluster

- Verify the deployed resources with public key & OCI artifact

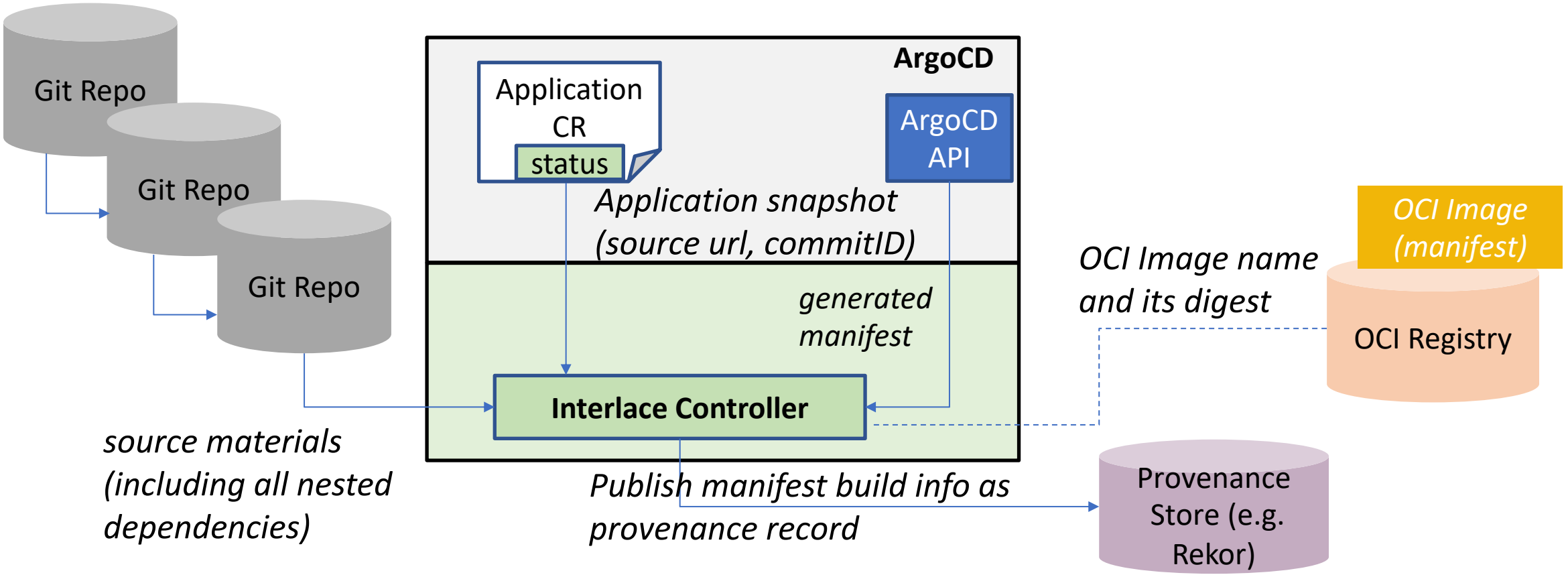
```
$ kubectl sigstore verify-resource <KIND> -k <PUBLIC_KEY> -i <OCI_ARTIFACT>
```

```
$ kubectl sigstore verify-resource deploy -n sample-ns -k ~/.sigstore/cosign.pub -i ghcr.io/hirokuni
-kitahara/sample-app-manifest:dev --allow-insecure-registry
NAME                            SIGNED   MANIFEST IMAGE                                          AGE
sample-kustomize-guestbook-ui   true     ghcr.io/hirokuni-kitahara/sample-app-manifest:dev       14h
```



Recording provenance

- Source materials (URL, commitID) including nested dependencies
- Snapshot of Application CR
- Manifest build artifacts (OCI image name and digest)



Get provenance from provenance store


```
$ provenance.sh -i ghcr.io/hirokuni-kitahara/sample-app-manifest:dev -k ~/.sigstore/cosign.pub
NAME                                          SIGNED  SIGNER  PROVENANCE
ghcr.io/hirokuni-kitahara/sample-app-manifest:dev  true    N/A     found

IMAGE DIGEST
sha256:64442cdc308687261d60e03929dbeac25e7bc9748eccf323648c0ad573022fdb

SOURCE REPO                                                COMMIT ID
https://github.com/hirokuni-kitahara/sample-kustomize-app.git   5c6172b89473fbf15d8df8e4e61203cd06b87b49
https://github.com/hirokuni-kitahara/sample-kustomize-base.git  bb8adffb03c7d8be5b1ca0af5eb5ee1c27182db6

BUILD TYPE        BUILT AT             ADDITIONAL INFO
argocd-interlace  2022-10-13T00:01:21  manifest generated for Application "sample-application-cluster1"
$
```

Get provenance from provenance store

Source maeteriarls (url, commit id, etc.)


```
"materials": [
  {
    "uri": "https://github.com/hirokuni-kitahara/sample-kustomize-app.git",
    "digest": {
      "commit": "8af6b72bb2232d500f6b8ca291fc221f89c522a6",
      "path": "./",
      "revision": "main"
    }
  },
  {
    "uri": "https://github.com/hirokuni-kitahara/sample-kustomize-base.git",
    "digest": {
      "commit": "bb8adffb03c7d8be5b1ca0af5eb5ee1c27182db6",
      "path": "",
      "revision": ""
    }
  }
]
```


Snapshot of Application CR (including status)


```
"parameters": {
  "applicationSnapshot": "{\"kind\":\"Application\",
}
```


OCI Image reference, digest


```
"subject": [
  {
    "name": "ghcr.io/hirokuni-kitahara/sample-app-manifest:dev",
    "digest": {
      "sha256": "b9b7ccdb6c10d7f9cf156f8b7257baf431e2524257b3ae51
    }
  }
],
```

Summary

- Modern CD GitOps engines dynamically generate YAML manifests, but it causes a problem of supply chain integrity
- YAML manifest signing improves the level of protection in delivery time.
- Interlace controller extends ArgoCD to fill the gap in manifest signing in CD GitOps.
- Trust but verify approach by Interlace makes application deployment transparent and accountable.

Thank you!
Q & A

Yuji Watanabe
twitter.com/ywatan1
linkedin.com/in/yuji-watanabe-0108322/
github.com/yuji-watanabe-jp

Hirokuni Kitahara
twitter.com/HirokuniKitaha1
linkedin.com/in/hirokuni-kitahara-a7450a165
github.com/hirokuni-kitahara

Resources

Interlace - https://github.com/argoproj-labs/argocd-interlace
Manifest Signing - https://github.com/sigstore/k8s-manifest-sigstore
Integrity Shield - https://github.com/stolostron/integrity-shield
Kyverno - https://github.com/kyverno/kyverno

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

https://sched.co/182HF