SSE

INTEGRITY OF DOCKER IMAGES

Signatures, Verification and a Tool for k8s

Who am I?

- Bachelor in computer science **@ FU Berlin**
- Master in computer science **@ FU Berlin**
- Master thesis about the **Meltdown** attack
- IT Security Engineer **@ SSE**

Philipp Belitz

- ✓ Cycling enthusiast
- ✓ Magic the Gathering player

MOTIVATION - SUPPLY CHAIN ATTACKS

Shadowhammer[1]

NotPetya[2]

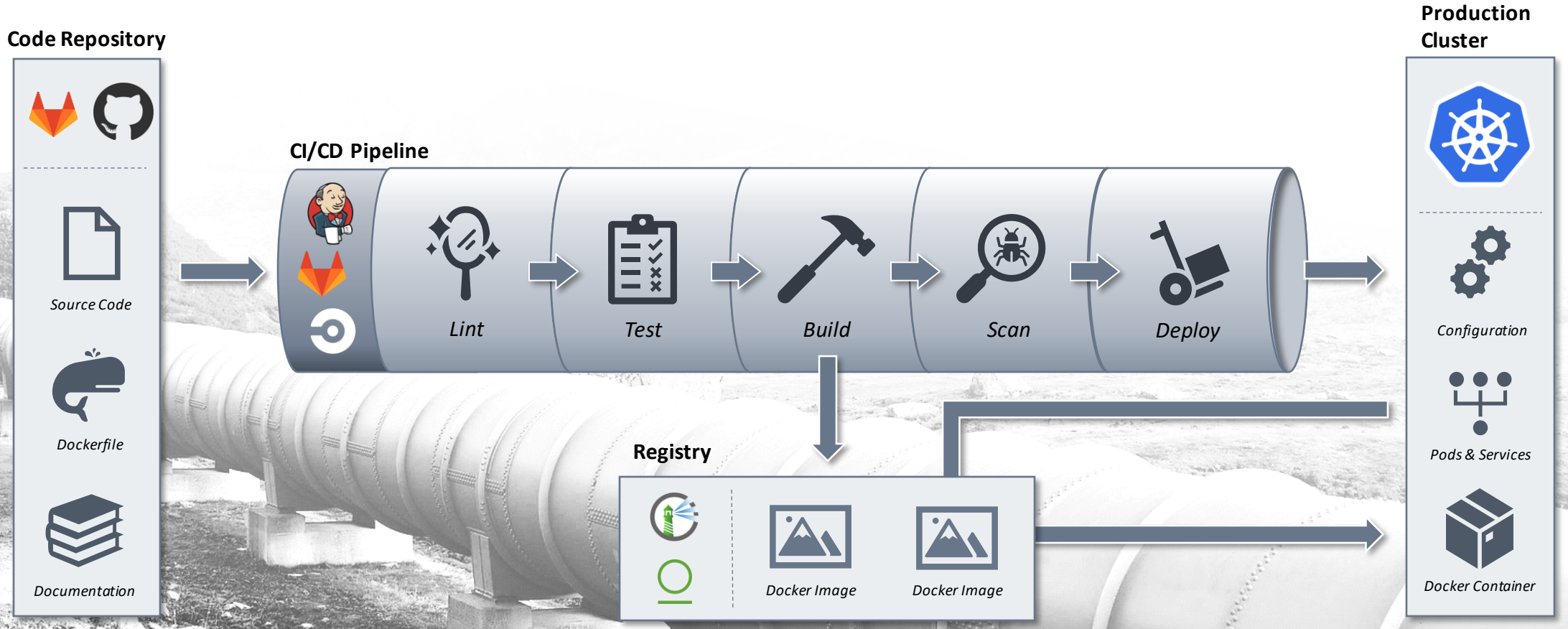
3 Years After NotPetya, Many Organizations Still in Danger of Similar Attacks

The same gaps that enabled ransomware to spread remain in patching, network segmentation, backup practices, security experts say.

Three years after the NotPetya ransomware outbreak overwhelmed numerous businesses in Ukraine and more than 60 other countries, many enterprises remain as vulnerable as ever to similar attacks.

[1] Quelle: https://www.cpomagazine.com/cyber-security/asus-supply-chain-attack-highlights-new-security-vulnerability-for-tech-giants
[2] Quelle: https://www.darkreading.com/threat-intelligence/3-years-after-notpetya-many-organizations-still-in-danger-of-similar-attacks/d/d-id/1338200

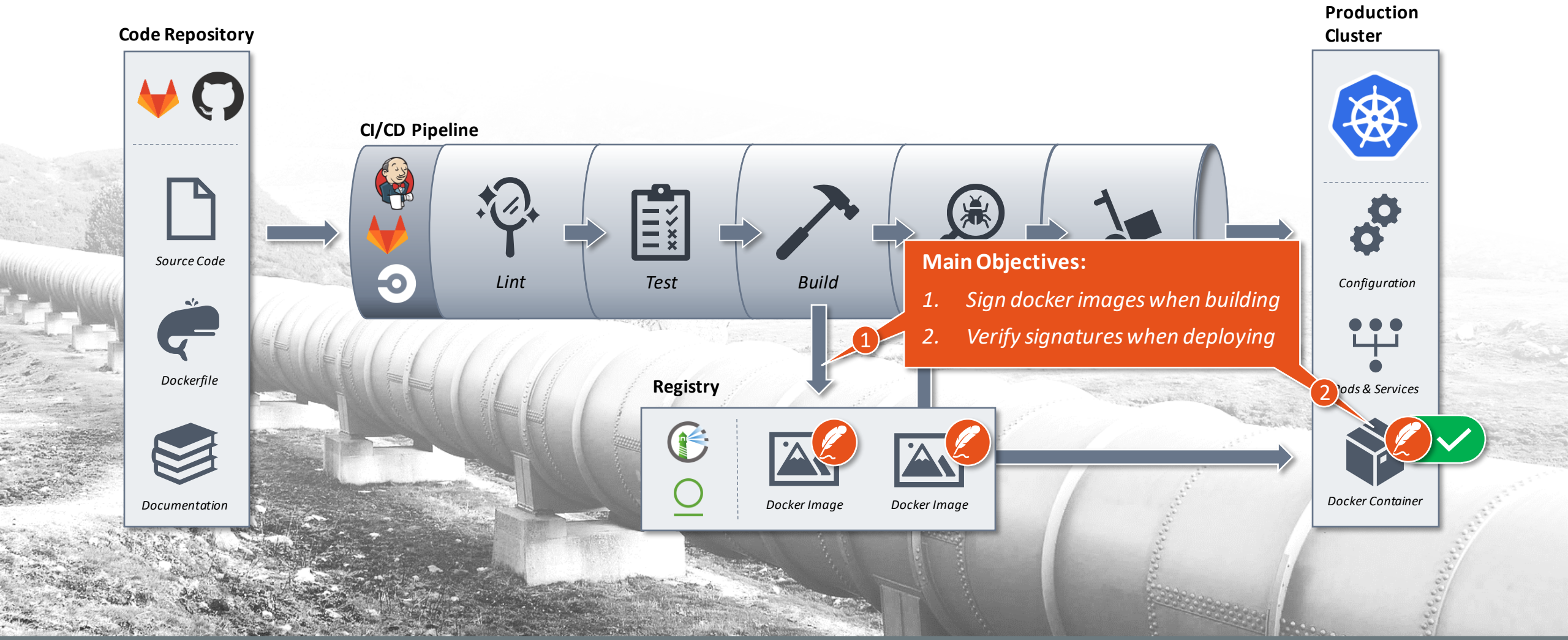
SUPPLY CHAIN ATTACKS – AN EXAMPLE



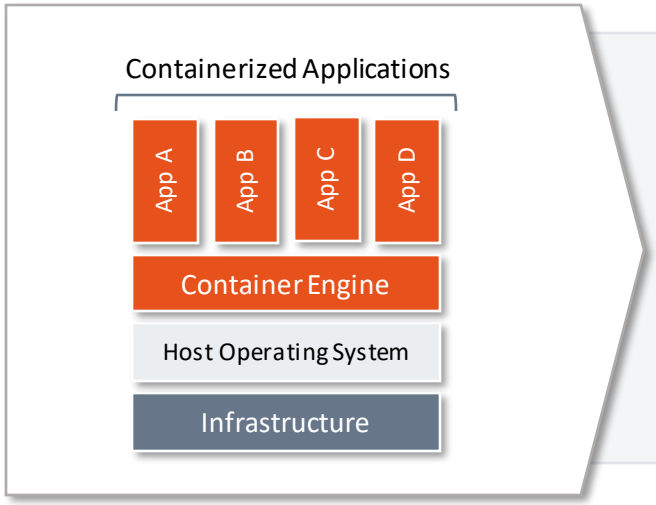
SUPPLY CHAIN ATTACKS – AN EXAMPLE

SUPPLY CHAIN ATTACKS – AN EXAMPLE

SUPPLY CHAIN ATTACKS – AN EXAMPLE



CONTAINERS – BASICS

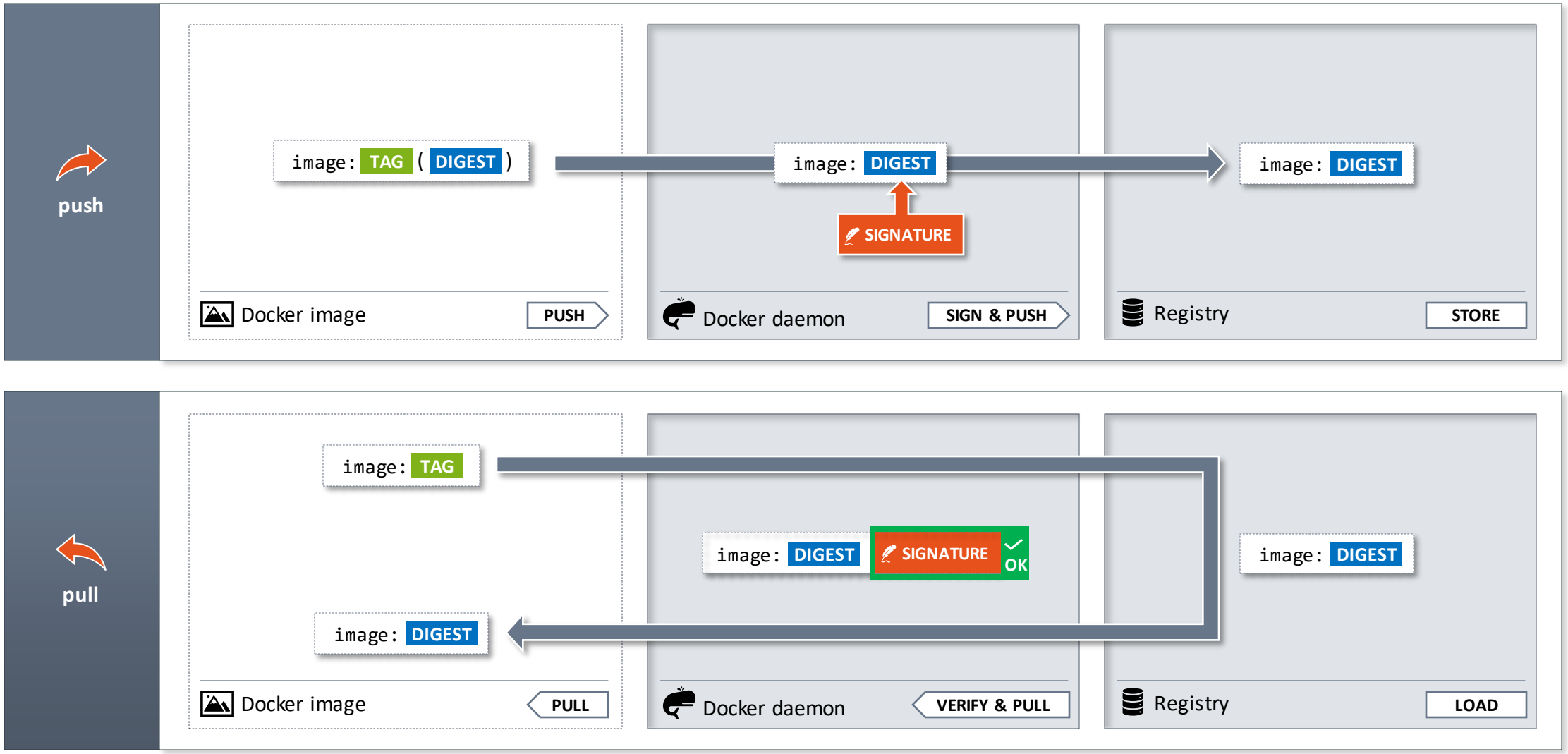


- **Container:** Full-featured unit of software (application) running in an isolated space on a host
- **Container Image:** Standalone, executable snapshot that becomes a container at runtime
- **Container Engine**: Runtime for containers, e.g. *Docker*, *CRI-O*
- **Container Orchestrator**: Manager for many containers at once, *e.g. Kubernetes*

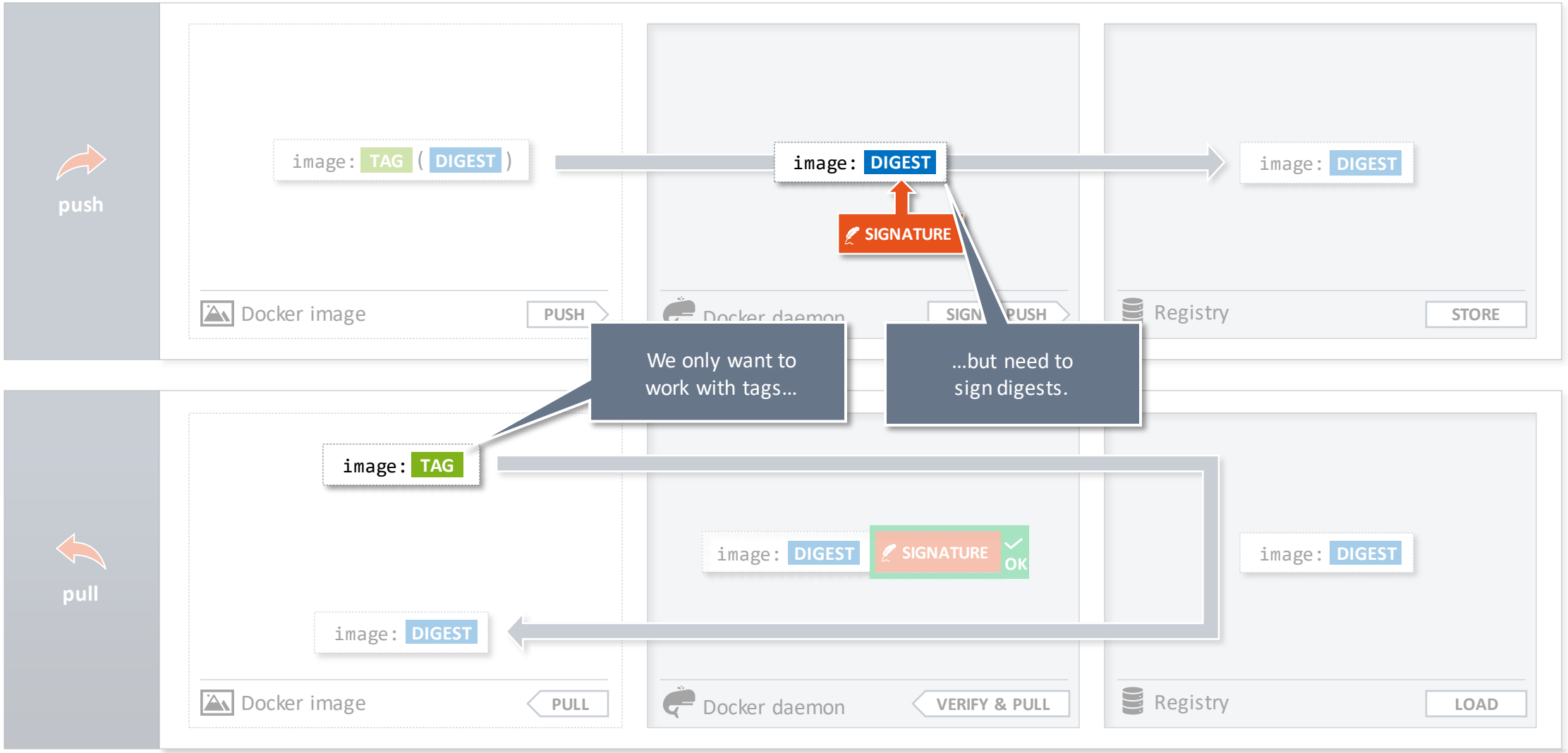
How to differentiate container images?

- Container images can be identified by their **name** → `docker.io/nginx:1.18`
- **Tag**: a mutable, human-readable description → `…/nginx:1.18.0`
- **Digest**: an immutable, unique SHA256 hash of the container's content → `…/nginx@sha256:69d4...5c00`
- Images always have a digest, but not necessarily a tag

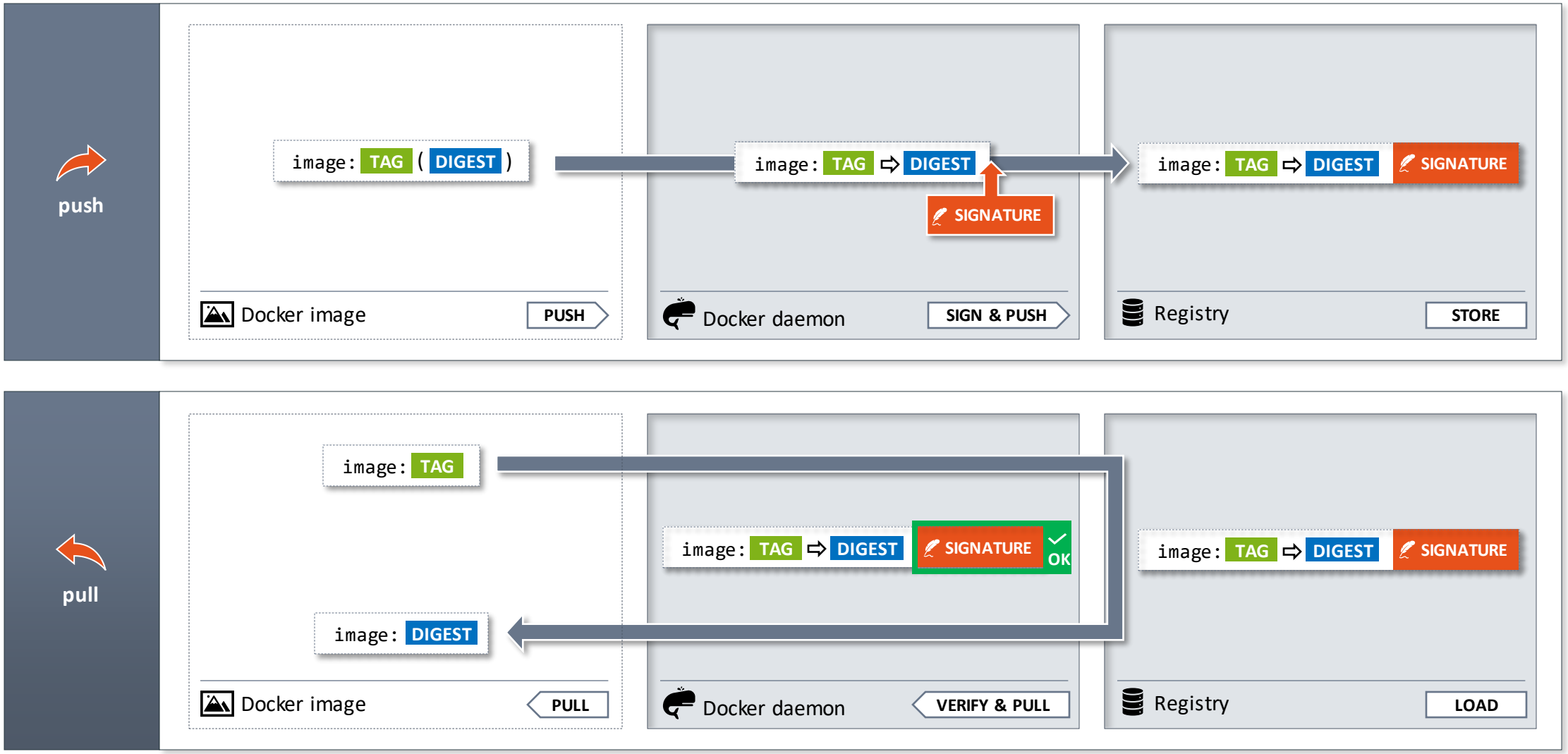
OBJECTIVE


push
image: TAG (DIGEST)
Docker image
PUSH
image: DIGEST
SIGNATURE
Docker daemon
SIGN & PUSH
image: DIGEST
Registry
STORE
pull
image: TAG
Docker image
PULL
image: DIGEST
SIGNATURE
OK
Docker daemon
VERIFY & PULL
image: DIGEST
Registry
LOAD
image: DIGEST

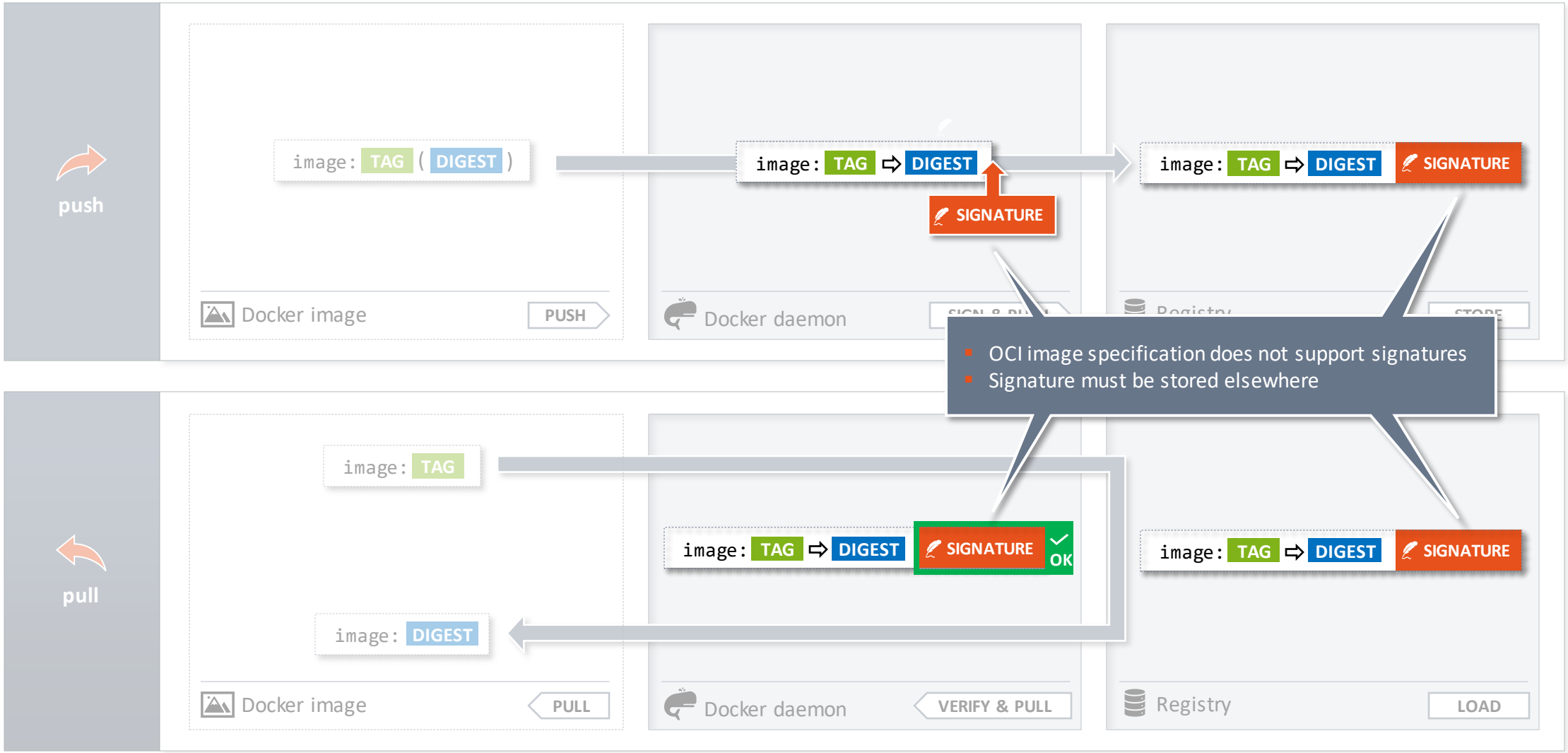
OBJECTIVE


push
image: TAG (DIGEST)
image: DIGEST
image: DIGEST
SIGNATURE
Docker image
PUSH
Docker daemon
SIGN & PUSH
Registry
STORE
We only want to work with tags...
...but need to sign digests.
pull
image: TAG
image: DIGEST
SIGNATURE
OK
image: DIGEST
image: DIGEST
Docker image
PULL
Docker daemon
VERIFY & PULL
Registry
LOAD

OBJECTIVE


push
image: TAG (DIGEST)
Docker image
PUSH
image: TAG ⇨ DIGEST
SIGNATURE
Docker daemon
SIGN & PUSH
image: TAG ⇨ DIGEST
SIGNATURE
Registry
STORE
pull
image: TAG
image: DIGEST
Docker image
PULL
image: TAG ⇨ DIGEST
SIGNATURE
OK
Docker daemon
VERIFY & PULL
image: TAG ⇨ DIGEST
SIGNATURE
Registry
LOAD

OBJECTIVE


push
image: TAG (DIGEST)
Docker image
PUSH
image: TAG ⇨ DIGEST
SIGNATURE
Docker daemon
image: TAG ⇨ DIGEST
SIGNATURE
OCI image specification does not support signatures
Signature must be stored elsewhere
pull
image: TAG
image: DIGEST
Docker image
PULL
image: TAG ⇨ DIGEST
SIGNATURE
OK
Docker daemon
VERIFY & PULL
image: TAG ⇨ DIGEST
SIGNATURE
Registry
LOAD

NOTARY AND TUF

Where to store the signatures?

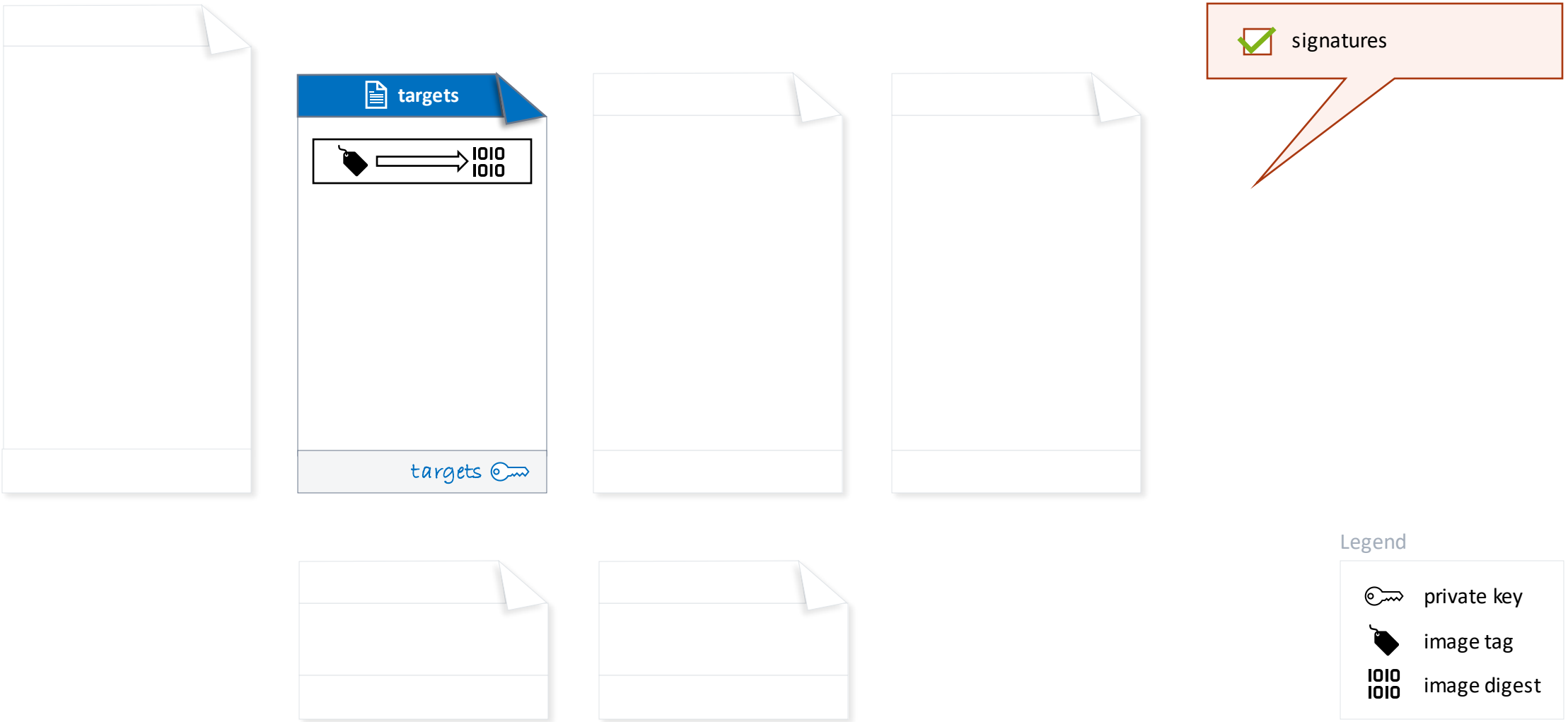
notary

- A Cloud Native Computing Foundation (CNCF) Project
- Works as server that stores signature information
- Implements *The Update Framework*

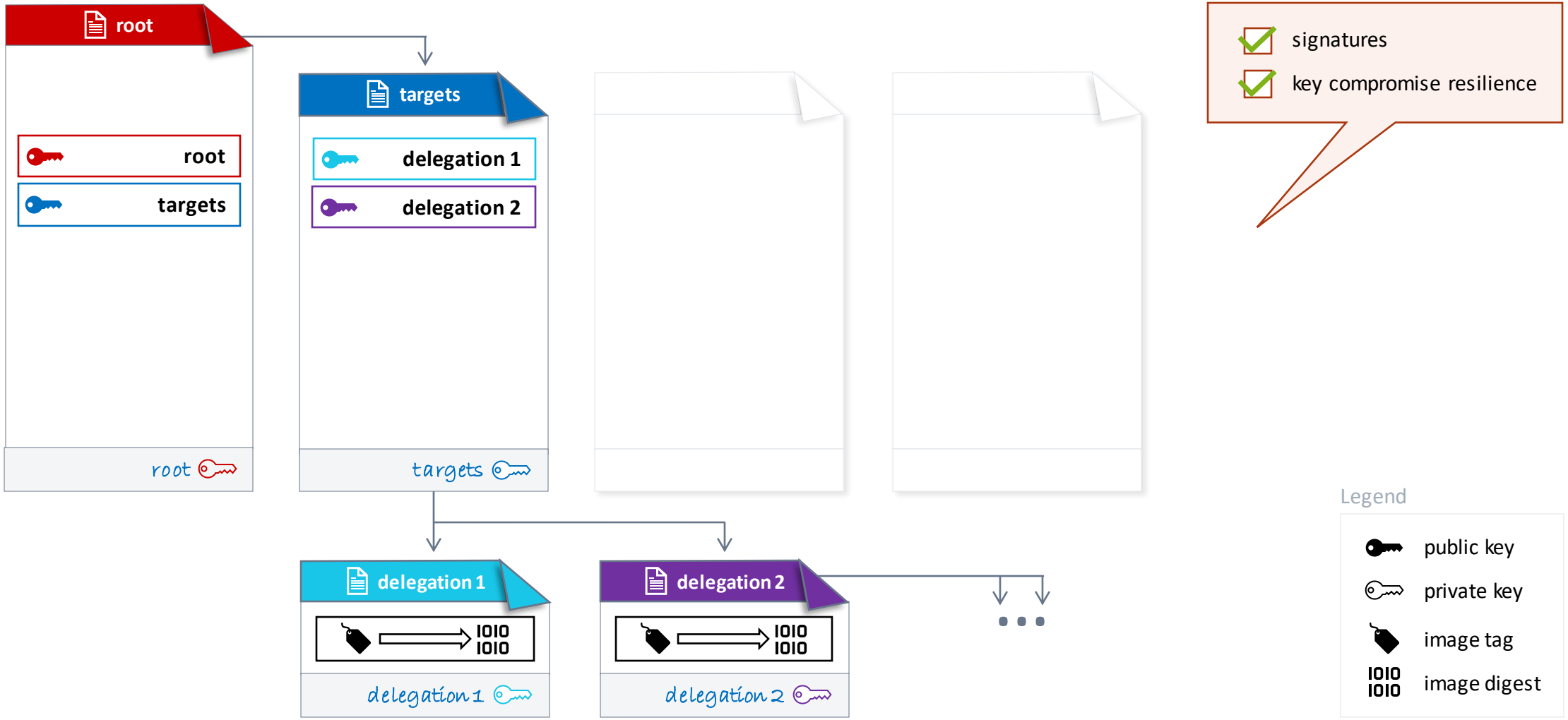
The Update Framework

- General Framework for securing software update systems
- Also CNCF
- Has several design goals:
  - Easy to integrate
  - Key compromise resiliance
  - Freshness

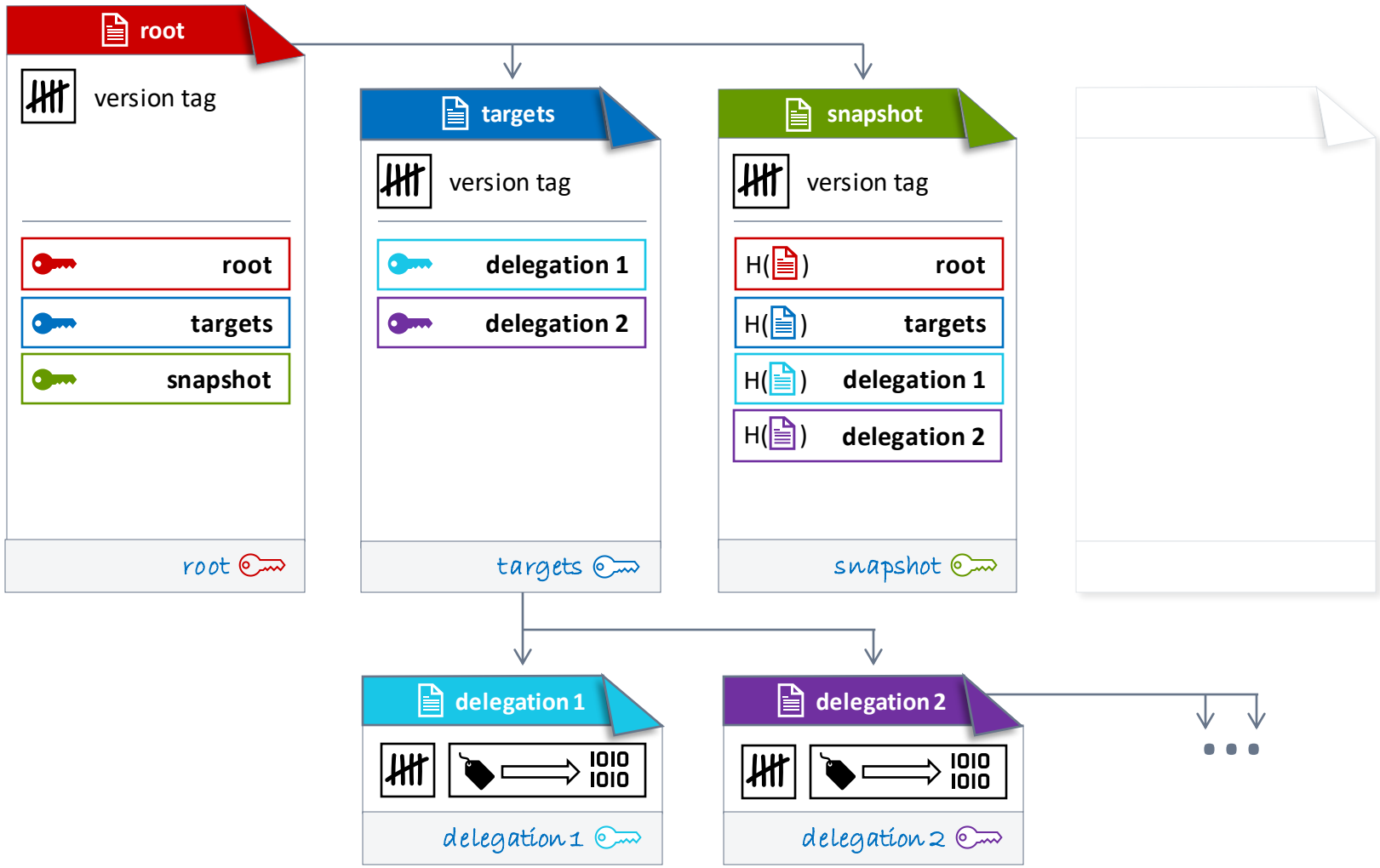
NOTARY AND TUF – STRUCTURE



NOTARY AND TUF – STRUCTURE



NOTARY AND TUF – STRUCTURE

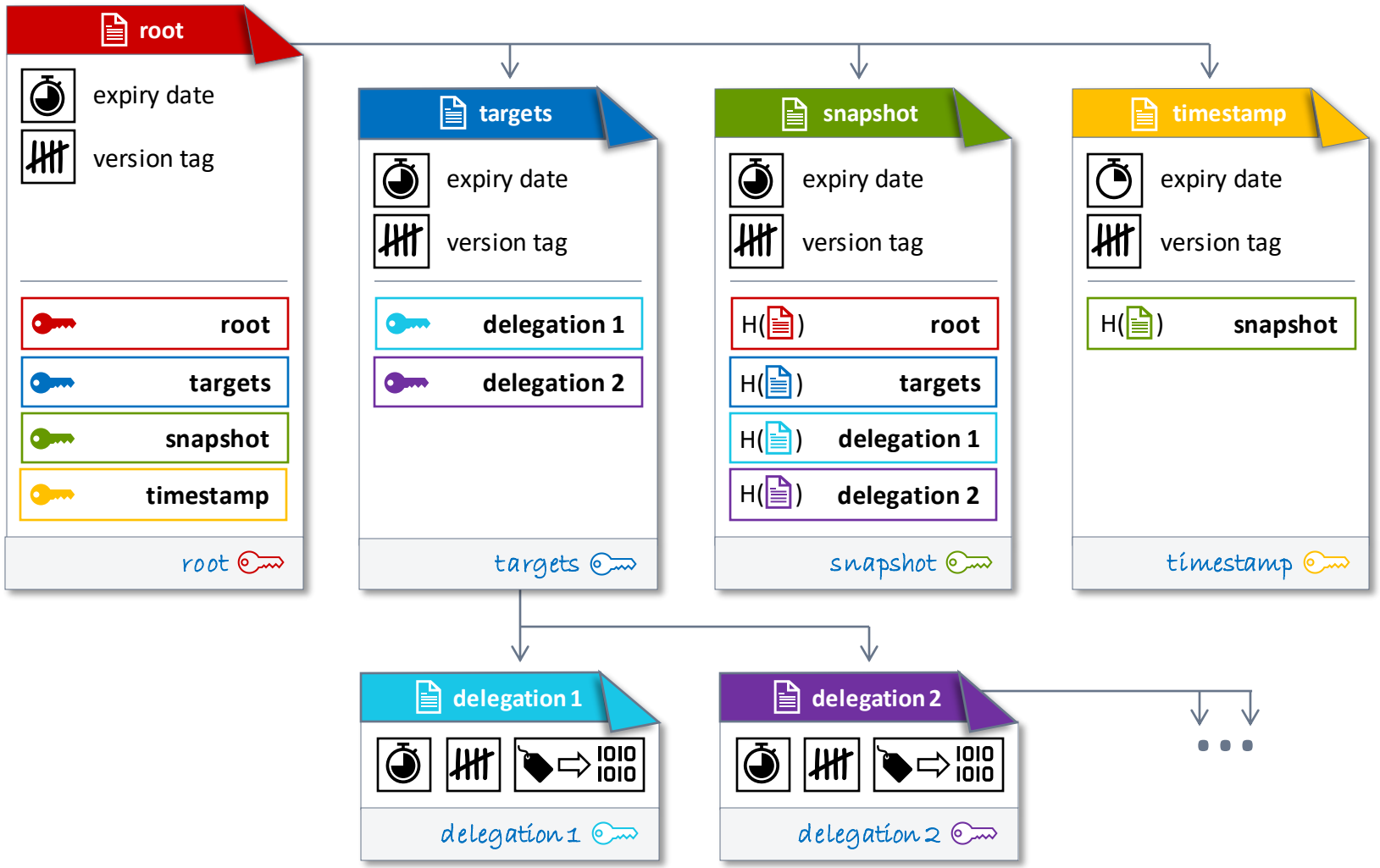


- signatures
- key compromise resilience
- rollback protection

Legend

- public key
- private key
- H( ) hash function
- version tag
- image tag
- 1010 1010 image digest

NOTARY AND TUF – STRUCTURE

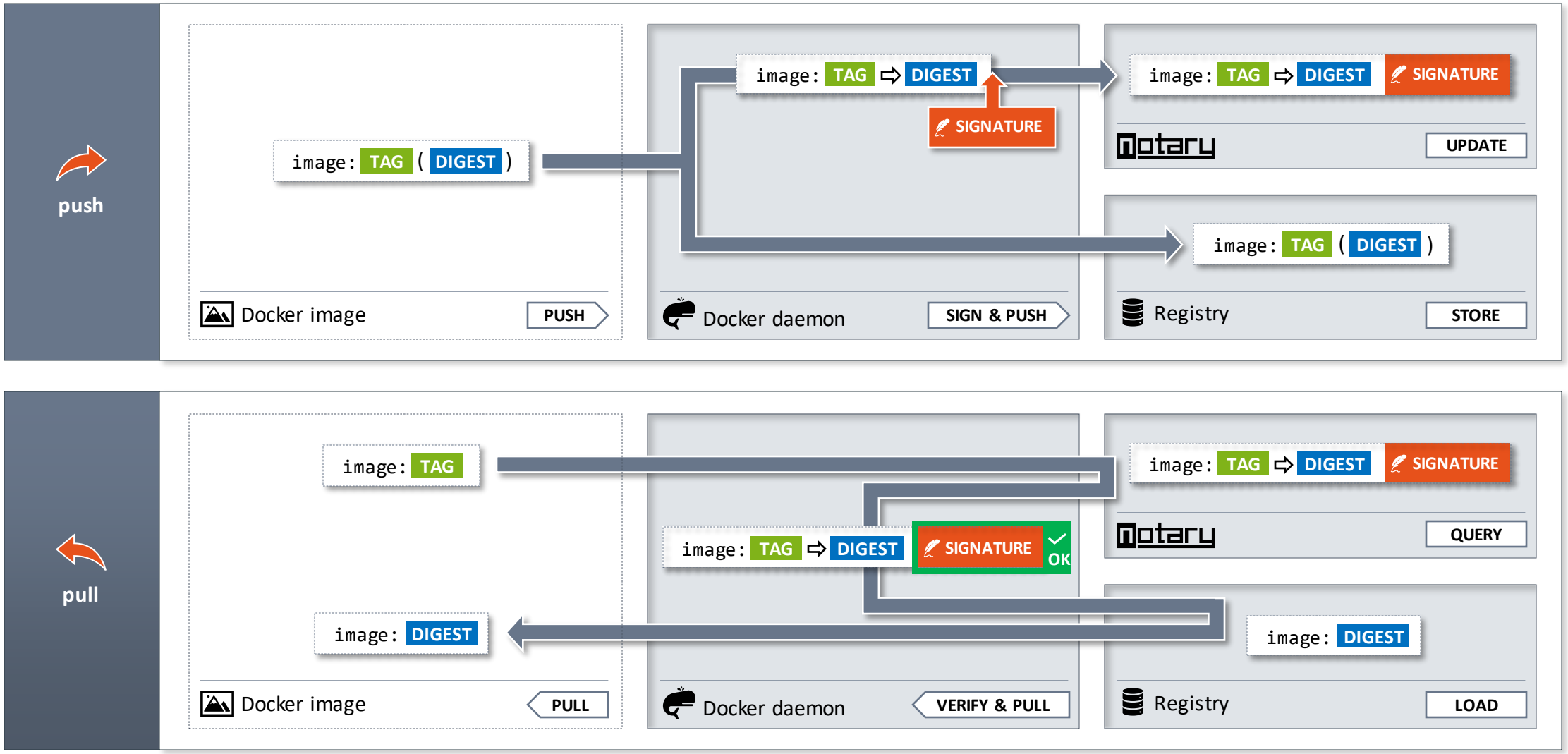


- signatures
- key compromise resilience
- rollback protection
- freeze protection

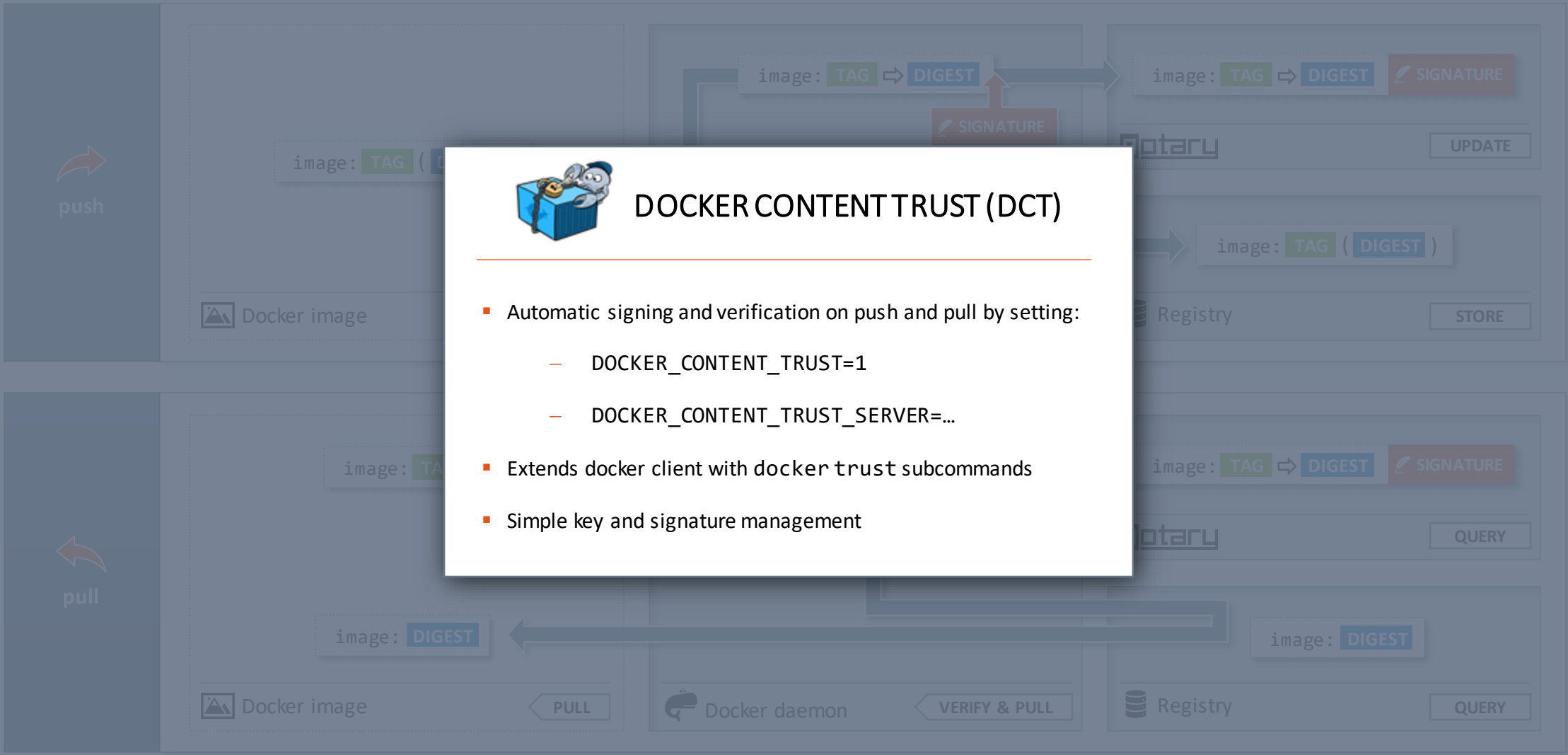
Legend

- public key
- private key
- H( ) hash function
- version tag
- expiry date
- image tag
- 1010 1010 image digest

OBJECTIVE


push
image: TAG (DIGEST)
Docker image
PUSH
image: TAG ⇨ DIGEST
SIGNATURE
Docker daemon
SIGN & PUSH
image: TAG ⇨ DIGEST
SIGNATURE
notary
UPDATE
image: TAG (DIGEST)
Registry
STORE
pull
image: TAG
Docker image
PULL
image: TAG ⇨ DIGEST
SIGNATURE
OK
Docker daemon
VERIFY & PULL
image: TAG ⇨ DIGEST
SIGNATURE
notary
QUERY
image: DIGEST
Registry
LOAD
image: DIGEST

OBJECTIVE



DOCKER CONTENT TRUST (DCT)

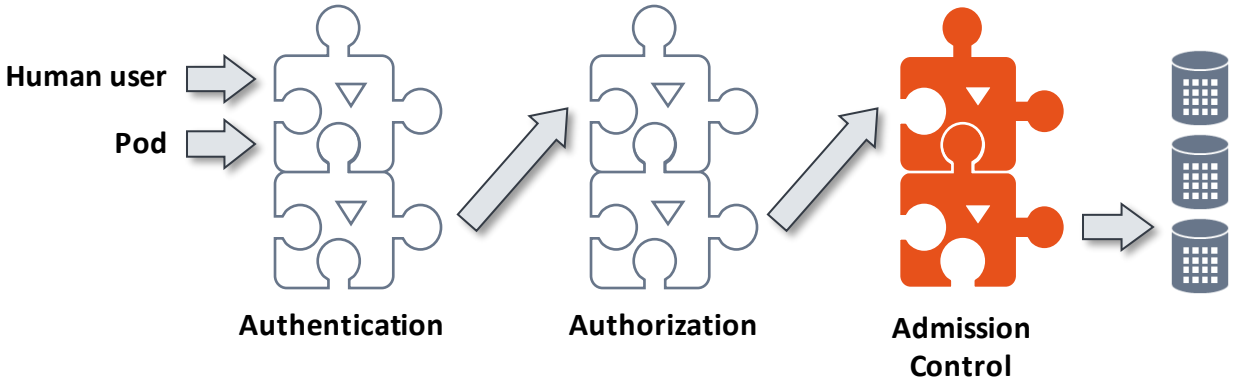
- Automatic signing and verification on push and pull by setting:
  - `DOCKER_CONTENT_TRUST=1`
  - `DOCKER_CONTENT_TRUST_SERVER=…`
- Extends docker client with `docker trust` subcommands
- Simple key and signature management

DOCKER CONTENT TRUST IN KUBERNETES

Kubernetes does not support Docker Content Trust!

How to integrate it into Kubernetes nevertheless

- Use Kubernetes Admission Controllers
- They intercept requests sent to Kubernetes + apply user-defined controls on them
- Two types: validating and mutating Admission Controller
- Use this for doing image signature verification

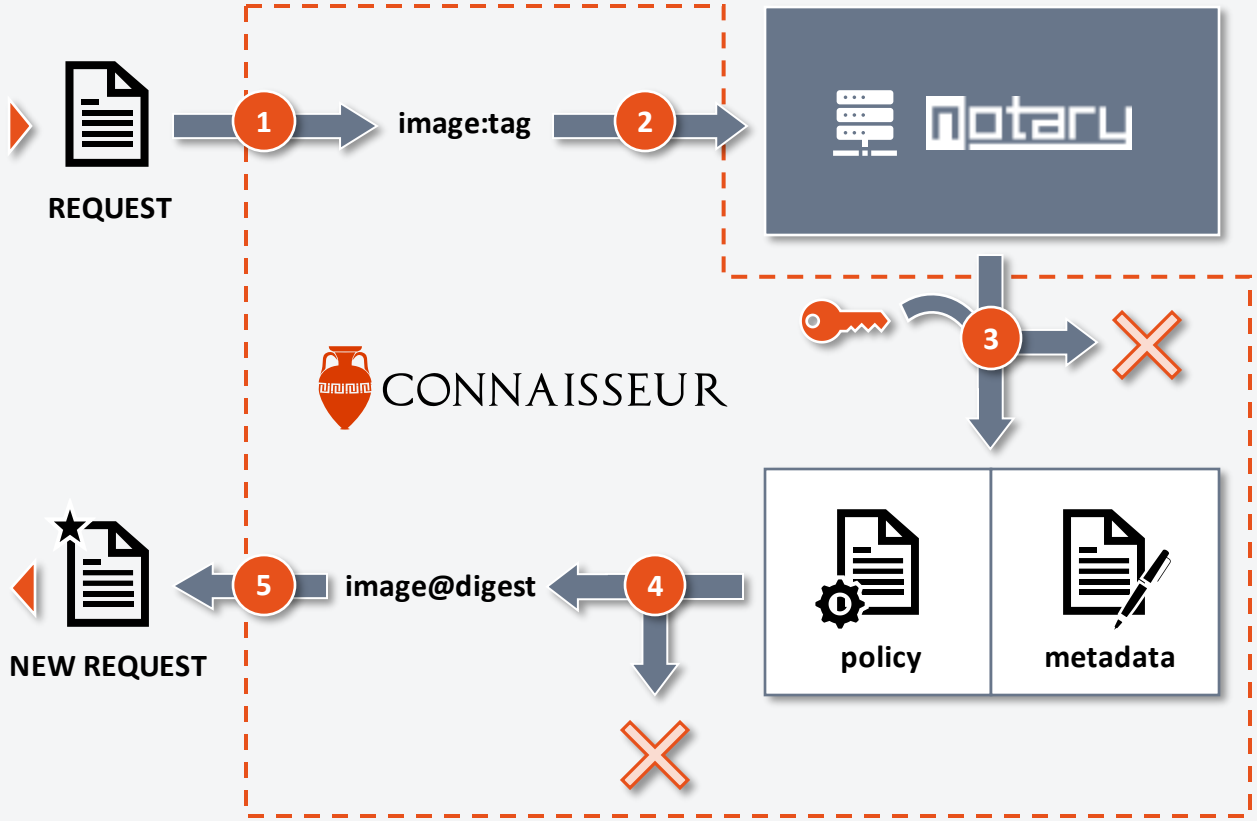


CONNAISSEUR

- ☑ Signature Verification
- ☑ Trust Pinning (public root key)

CONNAISSEUR

1. Extract image reference from request
2. Call notary API for trust data
3. Validate trust data
   - Deny request if no or invalid trust data
   - Proceed
4. Apply policy
   - Deny request if trust data doesn't comply to policy
   - Extract "signed" digest
5. Modify request with digest




DEMO

CONNAISSEUR

- Admission controller for Kubernetes
  - Signature Verification
  - Trust Pinning
- Open Source
- Design principles
  - Simplicity
  - Compatibility (AKS, EKS, GKE, K3s, MicroK8s, Minikube, SysEleven Metakube, …)
- Features
  - Allow-Listing
  - Detection Mode
  - (Alerting)
- Alternatives
  - Open Policy Agent (https://siegert-maximilian.medium.com/ensure-content-trust-on-kubernetes-using-notary-and-open-policy-agent-485ab3a9423c)
  - Portieris (IBM only)

OUTLOOK

- Notary v2 (https://github.com/notaryproject/nv2)
  - No more storing of signature in external server
  - Signature planned to be stored inside image, thus changing the OCI image specification
- Check out our GitHub Repository (https://github.com/sse-secure-systems/connaisseur)
- Also read our blog post for more details (https://medium.com/sse-blog/container-image-signatures-in-kubernetes-19264ac5d8ce)
- **Cheers**