KubeCon | CloudNativeCon

China 2018

The State of Your Supply Chain

Andrew Martin, Control Plane
Maya Kaczorowski, Google Cloud

Nov 15 2018

Hi!

KubeCon | CloudNativeCon
China 2018

Maya Kaczorowski

Security PM, Google

@MayaKaczorowski

Google Cloud

controlplane

Andy Martin

Founder, Control Plane

Dev-like, sec-ish, ops-y

@sublimino

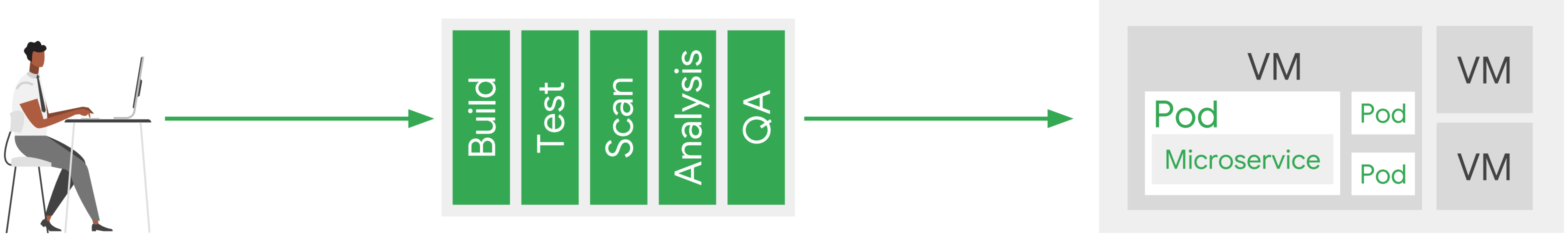
controlplane

What is a supply chain?

Anything that we depend upon

- e.g., the military need to know where all their hardware and software comes from and who builds them, to protect against state attacks
- e.g., pharmaceutical companies likewise need to know the provenance of their ingredients

What is a **software** supply chain?



Build

Test

Scan

Analysis

QA

VM

Pod

Microservice

Pod

Pod

VM

VM

Developer

CI/CD pipeline

Production environment

Any code that ends up running in production

Software supply chains can be exploited

- Vulnerabilities in dependencies, e.g., open-source packages
- Deliberate backdoors
- Compromised downloads, e.g., typosquatting

Software supply chains can be exploited

- Vulnerabilities in dependencies, e.g., open-source packages
- Deliberate backdoors
- Compromised downloads, e.g., typosquatting

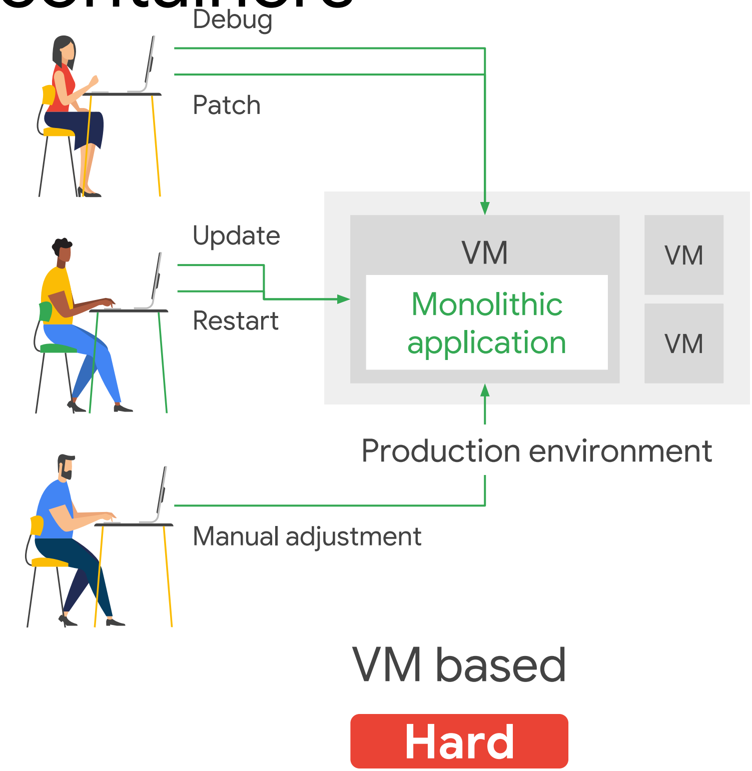
EQUIFAX

Apache Struts vulnerability

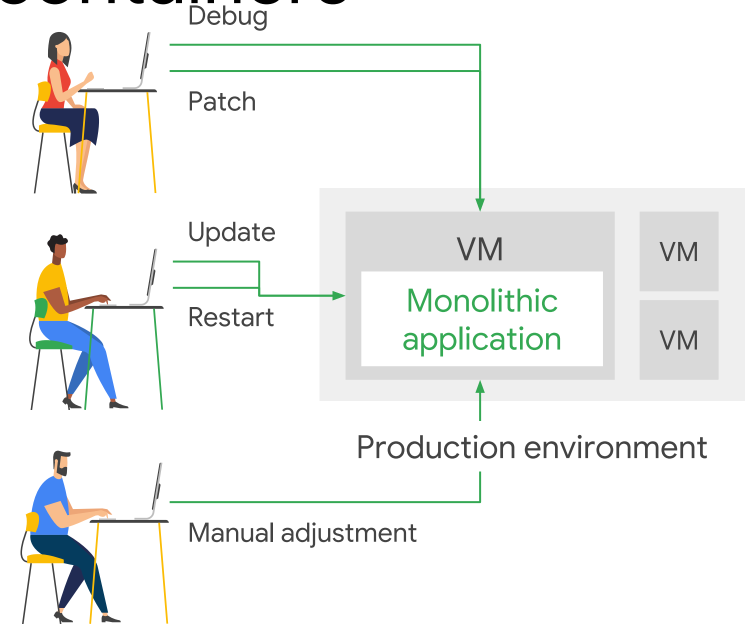
Malicious signed binary

Compromised software update server

What's different about supply chains with containers



What's different about supply chains with containers


Debug

Patch

Update

Restart

Manual adjustment

VM

Monolithic application

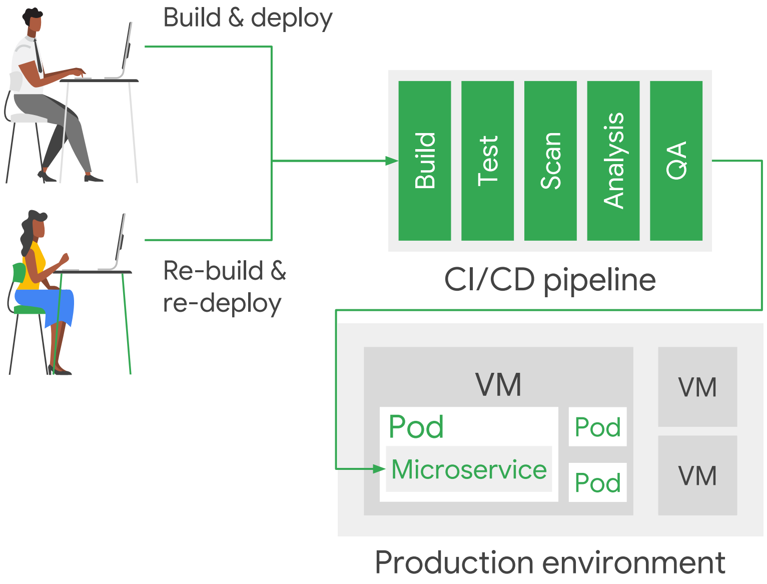
VM

VM

Production environment

VM based

Hard


Build & deploy

Re-build & re-deploy

Build

Test

Scan

Analysis

QA

CI/CD pipeline

VM

Pod

Microservice

Pod

Pod

VM

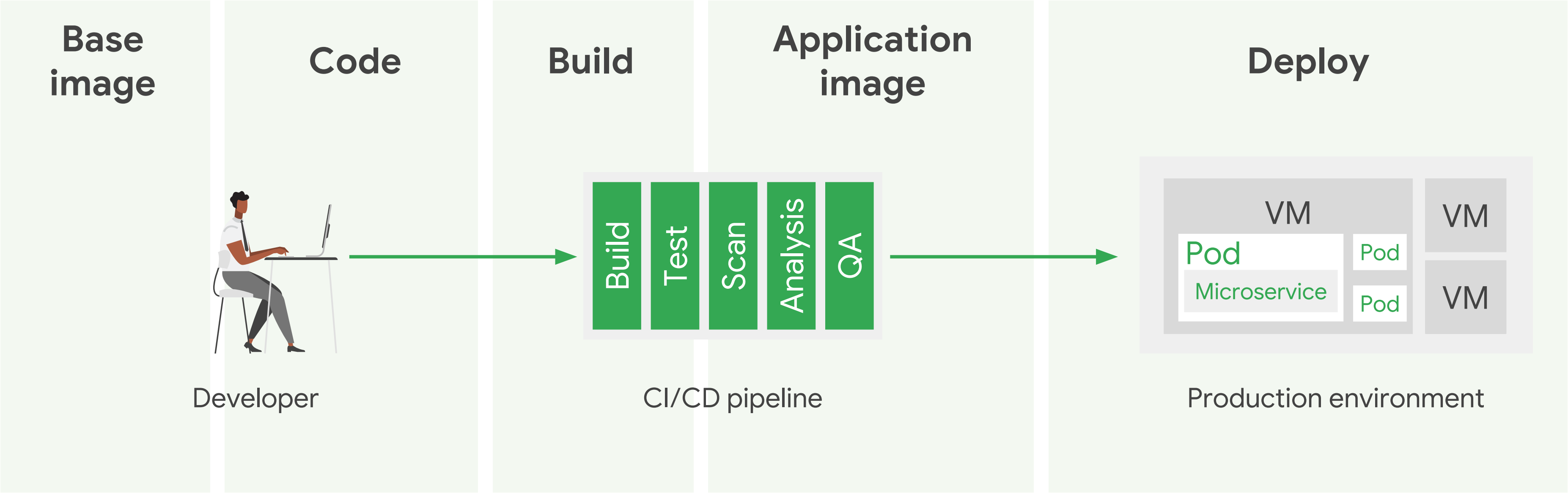
VM

Production environment

Container based

Easy

Stages of the CDLC (Container Delivery Lifecycle)



Base image

Code

Build

Application image

Deploy

Build

Test

Scan

Analysis

QA

VM

Pod

Microservice

Pod

Pod

VM

VM

Developer

CI/CD pipeline

Production environment

“I find your lack of security disturbing.”

https://www.slideshare.net/FranklinMosley/devsecops-a-new-hope-for-security-in-cicd

Base Image

- **Controlled base images**: official external images, copied into the organisation and promoted through dedicated pipelines
    - e.g. Docker Hub official images
- **Hash based addressing**: image has a verifiable “identity”
    - Hashes help ensure we have immutable images
    - Hashes are static - whereas tags are transitory and a possible risk

Code

- **Static analysis:** of code in-IDE (style, AST-analysis, atoms of confusion)
- **Dependency analysis:** Immediate and transitive (pom.xml, package.json, requirements.txt and pals)

Build

- **Hermetic builds**: Isolated build environment
    - No inter-build data or artefact leakage
- **Reproducible builds**: Repeatable build from source to binary
    - Build dependencies cached within an organisation's estate
    - Pinned versions for deterministic builds
    - Only helps security if you actually do reproduce it - not great for incremental builds
- The future: **rootless builds**: Build without privileged access
    - Tools like umoci, img, buildah, kaniko are moving towards a safer build environment
    - The class of build-time attacks this is mitigating against are aspirational rather than in-the-wild right now

Application Image scans

- **Vulnerability scanning:** CVE scans (operating system components, installed binaries/JARs/tarballs)
  - Patching
  - Removing packages
  - Smaller distribution
- **Configuration scanning:** Make it easy to do the right thing
  - Secrets in code
  - Images running as root
  - Misconfigurations
- **Policy:** filesystem configuration and Discretionary Access Controls, xattrs SUID/GUID, runtimes and debug tools, etc.

Deploy

- **Admission control:** Gated admission to production based on policy, compliance, and other metadata from previous build stages
- **Runtime configurations:** Adherence to PodSecurityPolicy and Kubesec.io risk based on runtime configuration of the images that comprise a pod

Enforced Governance

Containers are short lived and frequently re-deployed, **you can constantly be patching.**

Containers are immutable, **you can control what is deployed in your environment.**

Ideal, security-hardened container supply chain

Base image	Code	Build	Application image	Deploy
Controlled base images Hash based addressing	Static analysis Dependency analysis	Hermetic Reproducible Rootless	Vulnerability scanning Configuration scanning	Admission control Runtime configurations

State of the Ecosystem

Open-source supply chain today

Base image

Images: Docker Hub

Code

Updates: TUF, Notary

TUF

notary

Build

Pipeline metadata: Grafeas, in-toto

Grafeas

Application image

Vulnerability scanning: Clair, Micro Scanner, Anchore Open Source Engine

clair

aqua MicroScanner

Deploy

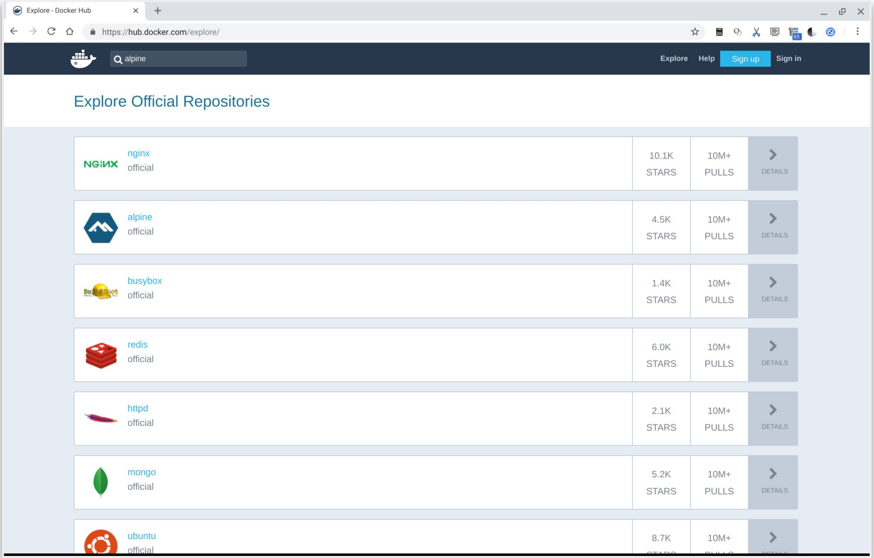
Admission control: K8s admission controllers, Kritis, Portieris

PORTIERIS

KUBESEC.IO

Images

Docker Hub



Explore Official Repositories

nginx official	10.1K STARS	10M+ PULLS	DETAILS
alpine official	4.5K STARS	10M+ PULLS	DETAILS
busybox official	1.4K STARS	10M+ PULLS	DETAILS
redis official	6.0K STARS	10M+ PULLS	DETAILS
httpd official	2.1K STARS	10M+ PULLS	DETAILS
mongo official	5.2K STARS	10M+ PULLS	DETAILS
ubuntu official	8.7K	10M+	

- Offers hundreds of ‘official’ images, including base images
  - Alpine
  - Debian
  - Ubuntu
- Best practices
  - Pull latest
  - Don’t trust blindly: check when last patched, scan for vulnerabilities

Updates

TUF vs Notary

The Update Framework (TUF) is a secure distribution mechanism, for signing software package updates

Notary is an implementation of TUF for container images specifically

Both CNCF projects

The Update Framework (TUF)

- Software package signing
- Secure key distribution mechanism
  - Update keys delegated by root key
  - Offline rotation
  - Temporal expiration
  - Resistant to replay attacks

Notary

- Implementation of TUF for image distribution
  - Server + database
  - Signer + database
- Signs and validates images
  - Signed collections
  - Key delegation
- Best practices
  - Store the master root key offline
  - Key rotation

Pipeline metadata

Why track pipeline metadata?

- Pipeline metadata is rich and varied
  - Initiating user(s) and/or events
  - Installed dependencies and their versions
  - Veracity test data, e.g., unit/integration/acceptance/&c tests
  - Security test data
  - Compliance and policy
- Data can be used for recording (audit) and reporting/enforcing (policy)

Grafeas

- Structured artifact metadata repository
  - Meant to be used as part of a container registry
- Spec includes multiple kinds of metadata
  - Package, Vulnerabilities, Discovery, Builds, Image basis, Deployment history, Attestation
- Can use multiple metadata providers
  - Providers include other scanning companies, e.g., JFrog, Red Hat, IBM, Black Duck, Twistlock, and Aqua
- You can use this metadata for enforcing restrictions on which containers get deployed
  - E.g., use “Admission” metadata with an admission controller to ensure compliance with your policies before deploying

Grafeas: concepts

- **Notes** are the definition of something that can be found or detected through analysis
- **Occurrences** are instances of a Note
- **Providers** are sources of metadata
- **Projects** are namespaces for metadata
- **Attestations** are cryptographic signatures
  - They aren't a separate object - but rather a metadata type part of Notes and Occurrences

in-toto

- Framework to provide whole software supply chain security
- Provides tooling and a metadata format to ensure all steps:
  - Are performed by the right party
  - Follow the expected policy
  - Use the right artefacts
  - Report the artefacts that were produced

in-toto: layouts

```
“_type”: “layout”,
“expires”: “2018-11-30T12:44:15Z”,
“keys”: {
    “0c6c50”: {…}
},
“signatures”: {…}
“steps”: [{
    “_type”: “step”,
    “name”: “checkout-code”,
    “expected_command”: [“git”, “clone”, “...”],
    “expected_materials”: [ ],
    “expected_products”: [[“CREATE”, “demo-project/foo.py”], … ]
    “pubkeys”: [“0c6c50”],
    “threshold”: 1
}, … ]
“inspections” : [...]
```

in-toto: execution parties and links

- Three types of parties
  - **Project owner**: defines a policy
  - **Functionary**: carries out a step and produces a statement as link metadata
  - **Verifier**: ensures all the link metadata matches the layout policy
- Links are cryptographically signed by the functionary

```
“_type”: “link”,
“name”: “build”,
“byproducts”: {“stderr: “”, “stdout”: “”},
“command: [...],
“materials”: {...},
“products”: {
    “foo”: {“sha256”: “...”}
},
“return_value”: 0,
“signatures”: [...]
```

in-toto: verification

- Checks for compliance using Link metadata and the Layout metadata
- Verification can be done in many steps:
  - Continuously (e.g. polling the Docker API endpoint)
  - Upon installation (e.g. hooking the package manager)
  - Before deployment (e.g. a Kubernetes admission controller)
- in-toto doesn’t care what you’re verifying
  - It’s just verifying a chain of signatures
  - With a little change-management tooling integration, it could help automate bureaucratic releases processes

Grafeas vs in-toto

Grafeas

- Strict opinionated API schema - “on rails”
- Supported by Google
- Limited documentation

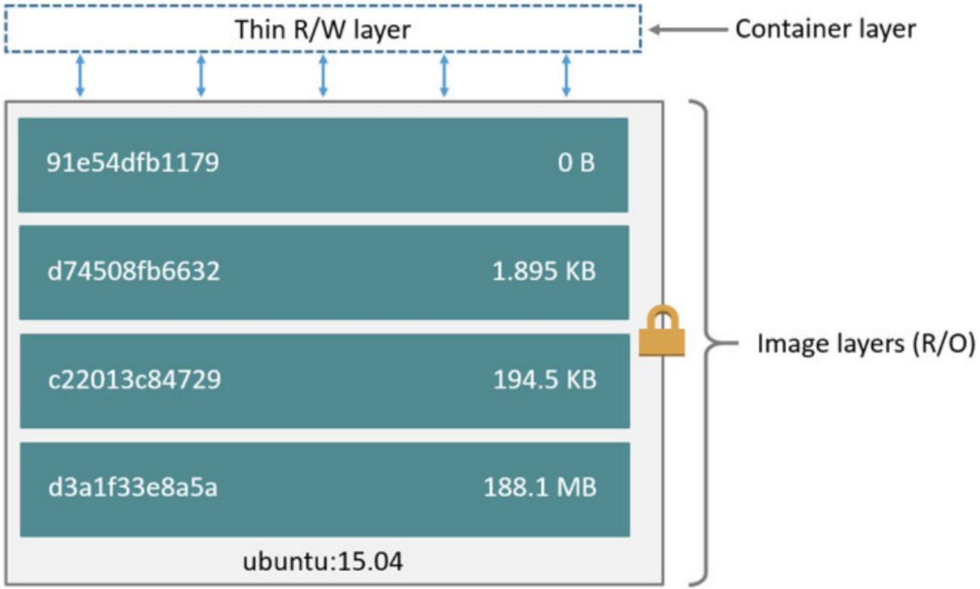
in-toto

- Adaptable to your environment, supports unstructured data
- Can chain together attestations to assert the integrity of a whole supply chain
- Can use different storage backends

Integration between Grafeas & in-toto proposed

Vulnerability scanning

Image vulnerability scanning approaches



- Components to scan: package-level vs. code-level
  - OS packages
  - App library packages
  - JARs, WARs, TARs, etc.
  - Malware
  - Misconfigurations, e.g., secrets
- Scan type
  - Layer-by-layer
  - UnionFS top layer only

https://sysdig.com/blog/container-security-docker-image-scanning/

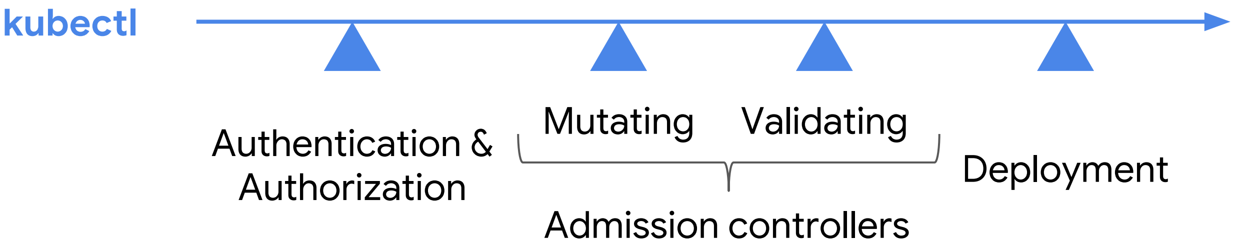
Clair vs. MicroScanner vs. Anchore

	Scanning depth	OS covered	Maintainer
clair	Packages	Alpine, CentOS, Debian, Oracle Linux, RHEL, Ubuntu	CoreOS
aqua MicroScanner	Packages		Aqua Security
anchore	Packages, files, software artifacts		Anchore

Admission control

Kubernetes admission controllers

- Admission controllers are a concept built into Kubernetes
  - **Mutating:** can modify objects
  - **Validating**: can't modify objects
- Can customize for whatever you want to check



kubectl

Authentication & Authorization

Mutating

Validating

Admission controllers

Deployment

Kritis

- Signing and deploy enforcement tool for Kubernetes
  - Implemented as a Kubernetes admission controller
  - Integrates with Grafeas attestation metadata APIs
- Generate attestations based on your requirements
  - Build provenance
  - Vulnerability findings

Kritis: ImageSecurityPolicy example

```
apiVersion: kritis.grafeas.io/v1beta1
kind: ImageSecurityPolicy
metadata:
 name: my-isp
spec:
 imageWhitelist:
  - gcr.io/kritis-int-test/nginx-digest-whitelist:latest
  - gcr.io/kritis-int-test/nginx-digest-whitelist\
@sha256:56e0af16f4a9d2401d3f55bc8d214d519f070b5317512c87568603f315a8be72
 packageVulnerabilityRequirements:
   maximumSeverity: HIGH # BLOCKALL|LOW|MEDIUM|HIGH|CRITICAL
   whitelistCVEs:
     - providers/goog-vulnz/notes/CVE-2017-1000082
     - providers/goog-vulnz/notes/CVE-2017-1000081
```

Portieris

- Notary Admission Controller
- Portieris enforces Content Trust
  - Different levels of trust for different images
- A mutating admission webhook ensures Kubernetes pulls the signed version
- Enforces trust pinning, and blocks the creation of resources that use untrusted images
- Supports IBM Cloud Container Registry, Quay.io, Docker Hub

Summary

Ideal, security-hardened container supply chain

Base image	Code	Build	Application image	Deploy
Controlled base images Hash based addressing	Static analysis Dependency analysis	Hermetic Reproducible Rootless	Vulnerability scanning Configuration scanning	Admission control Runtime configurations

Open-source supply chain today

Base image

Images: Docker Hub

Code

Updates: TUF, Notary

TUF

notary

Build

Pipeline metadata: Grafeas, in-toto

Grafeas

Application image

Vulnerability scanning: Clair, Micro Scanner, Anchore Open Source Engine

clair

aqua MicroScanner

Deploy

Admission control: K8s admission controllers, Kritis, Portieris

PORTIERIS

KUBESEC.IO

www.JJ.AM