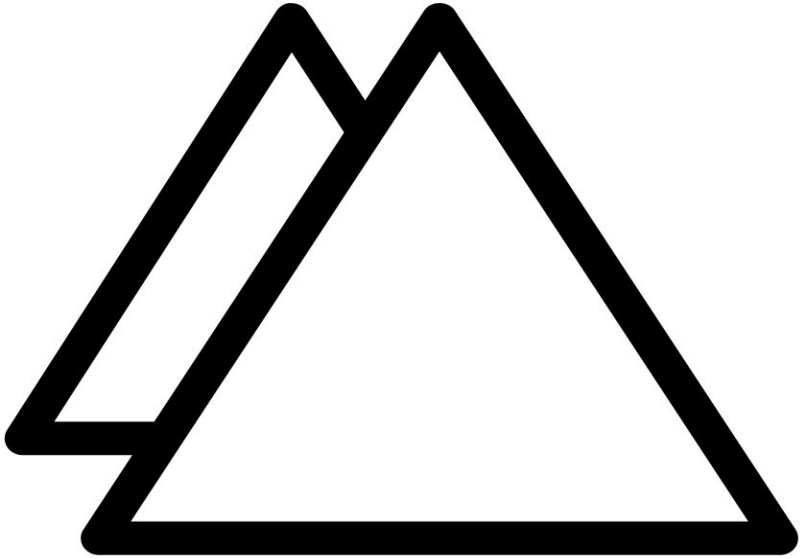
Kubernetes Supply Chain Security: The Software Factory

a.k.a. Who's Afraid of the Big Bad Supply Chain?

KubeCon NA, October 2021

@sublimino & @controlplaneio



controlplane

Author:

- SANS SEC584
  - *Cloud Native Security: Defending Containers and K8s*
- O'Reilly
  - *Hacking Kubernetes*
  - *Kubernetes Threat Modelling*
- ControlPlane

Trainer:

- Hashicorp
- Docker

I'm:

- **Andy**
- **Dev**-like
- **Sec**-ish
- **Ops**-y

https://learning.oreilly.com/library/view/hacking-kubernetes/9781492081722/

Hacking Kubernetes

With such favourites as:

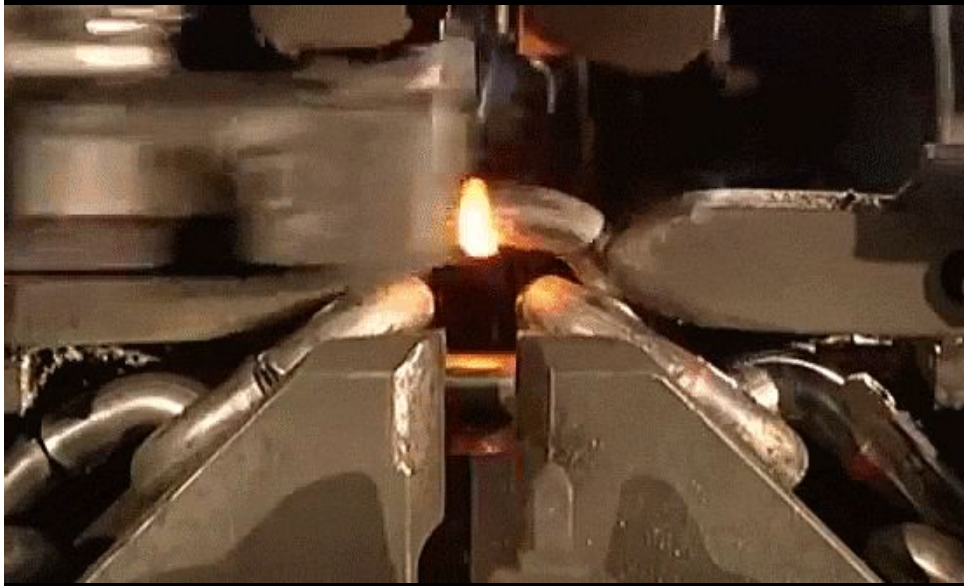
- Kubernetes YOLO moves
- Threat models ‘n’ attack trees
- Running Kubernetes “securely”
- Exploiting your Kuberneteses
- The future state of containers...

🔥🔥 *Pre-orderable*: https://smile.amazon.co.uk/dp/1492081736 🔥🔥

controlplane

TL;DR

- Supply Chain **ELI5**
- How to **Attack a Supply Chain**
- **Sign** me to the Moon
- The **Software Factory**



Prior Art, References, and Thanks

- https://github.com/sigstore
  - https://mobile.twitter.com/lorenc_dan
  - https://mobile.twitter.com/lukehinds
  - cosign, fulcio, rekor (see: other talks)
- In-Toto
  - https://mobile.twitter.com/torresariass
- TUF
- SPIFFE/SPIRE
- SBOM standards
  - SPDX
  - CycloneDX
- TektonCD
- CNCF TAG Security

controlplane

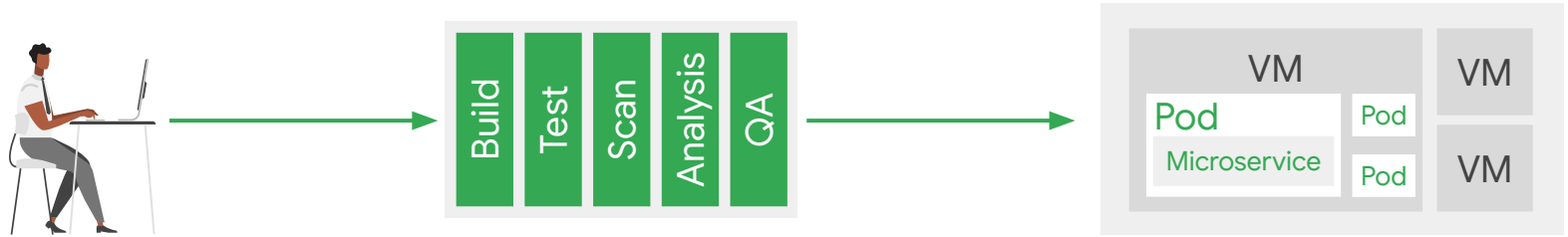
Supply Chains

- En vogue / so hot right now
- Is anything we depend upon (military, pharmaceutical, food, manufacture...software!)
- Beyond our direct control as consumers
- Reliant on “Trust” in software providers

Supply Chains

- En vogue / so hot right now
- Is anything we depend upon (military, pharmaceutical, food, manufacture...software!)
- Beyond our direct control as consumers
- Reliant on “Trust” in software providers

What is a **Software** Supply Chain?



Developer

CI/CD pipeline

Production environment

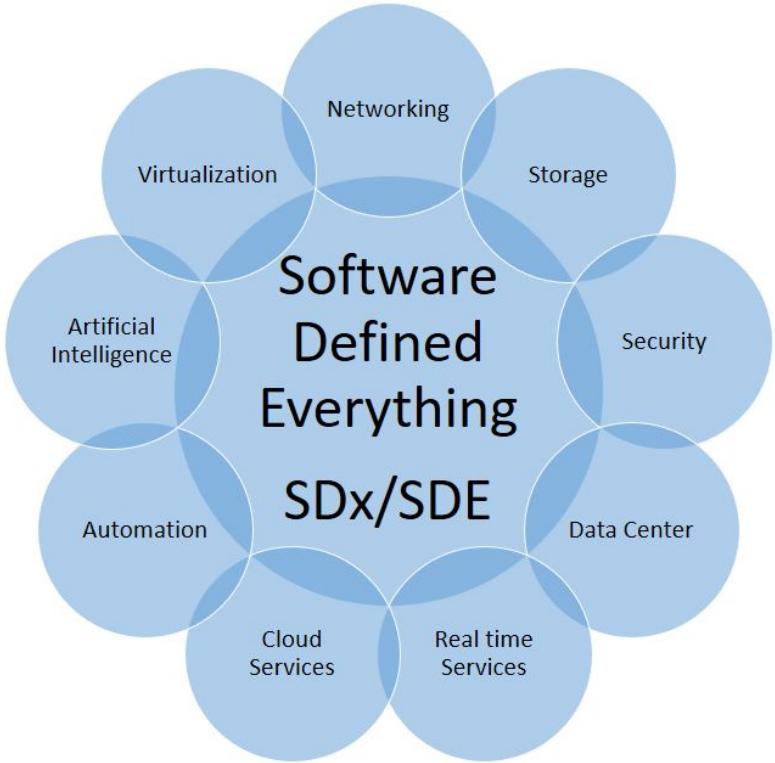
Any code that ends up running in production

What could possibly go wrong?

controlplane

Everything is Software

- Software types:
  - Applications
  - Infrastructure
  - Security
  - Policy
  - ...Configuration
- **Similar controls** can be applied to entire classes of software, containers, and systems (**static analysis**, **composition scanning**, etc)



[Software Defined Everything - Benefit for All: Myth or Reality?]

controlplane

Types of pipeline

Application Pipelines

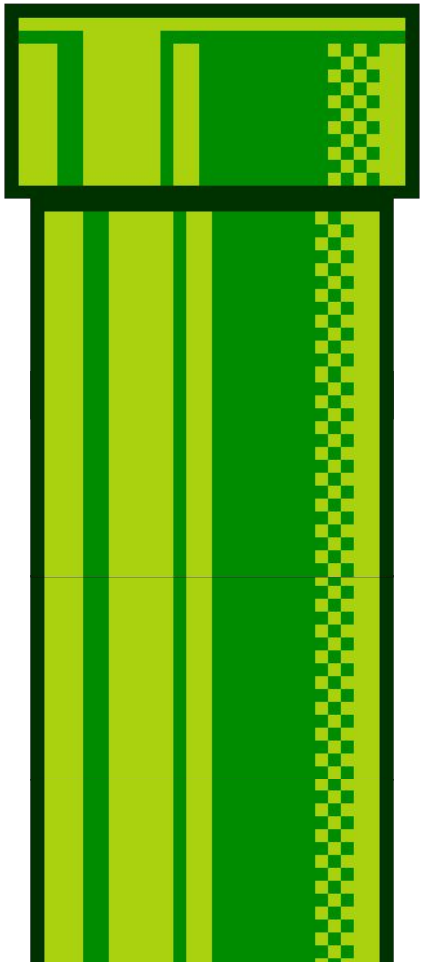
- Building an artefact (building and testing containers and applications, security testing)
- Deploying an artefact (GitOps for Kubernetes, CaaS with GCP Cloud Run or AWS Fargate)

Infrastructure (Infrastructure as Code) Pipelines

- Building an artefact (AMI, container image, FaaS archive)
- Deploying infrastructure (Terraform, Cloud Formation)
- Provisioning installations (Ansible, Puppet)

Security and Non-functional (Security as Code) Pipelines

- Testing security and mis-configuration (Network services, DNS, TLS, routing, response time and availability)
- Performance and Load (System responsiveness and availability)



controlplane

Supply Chain ELI5

- 👩🏾‍💻 **Everything is software**
- 👥 Software is made of **other people's software**
  - Applications, libraries, compilers, operating system, firmware, microcode, …
- 🔒 We **rely on the producers** of software for safety
  - We want **correct code**
  - And also **non-violent code**

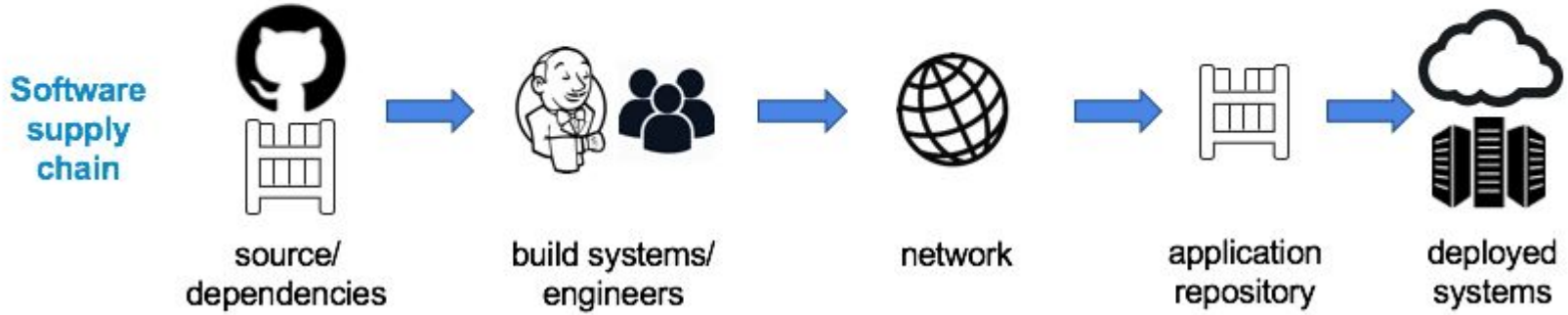
Supply Chain ELI5

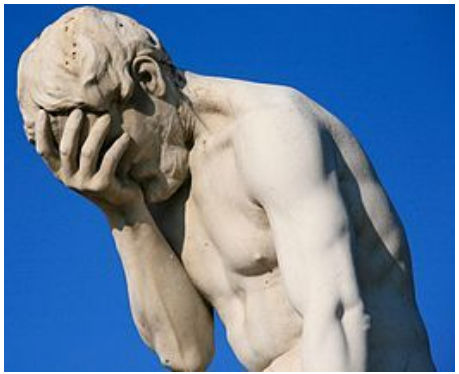
- 👩‍💻 **Everything is software**
- 👥 Software is made of **other people's software**
  - Applications, libraries, compilers, operating system, firmware, microcode, …
- 🔒 We **rely on the producers** of software for safety
  - We want **correct code**
  - And also **non-violent code**
- ⛓️ Alice -> *Bob* -> Charlie -> **production**
  - **Bob** delivers Alice's code to Charlie
  - If *Bob drops **malware** into the build*
  - **Malware runs in production**
- 🗺️ Supply chains are **long**, **difficult** to secure…
- 🛡️...and **The New Software Security Frontier (for ~40y)**

controlplane

What must we Trust in a Supply Chain?

...almost everything





https://blog.convisoappsec.com/en/is-your-software-supply-chain-secure/

controlplane

Trust These Men

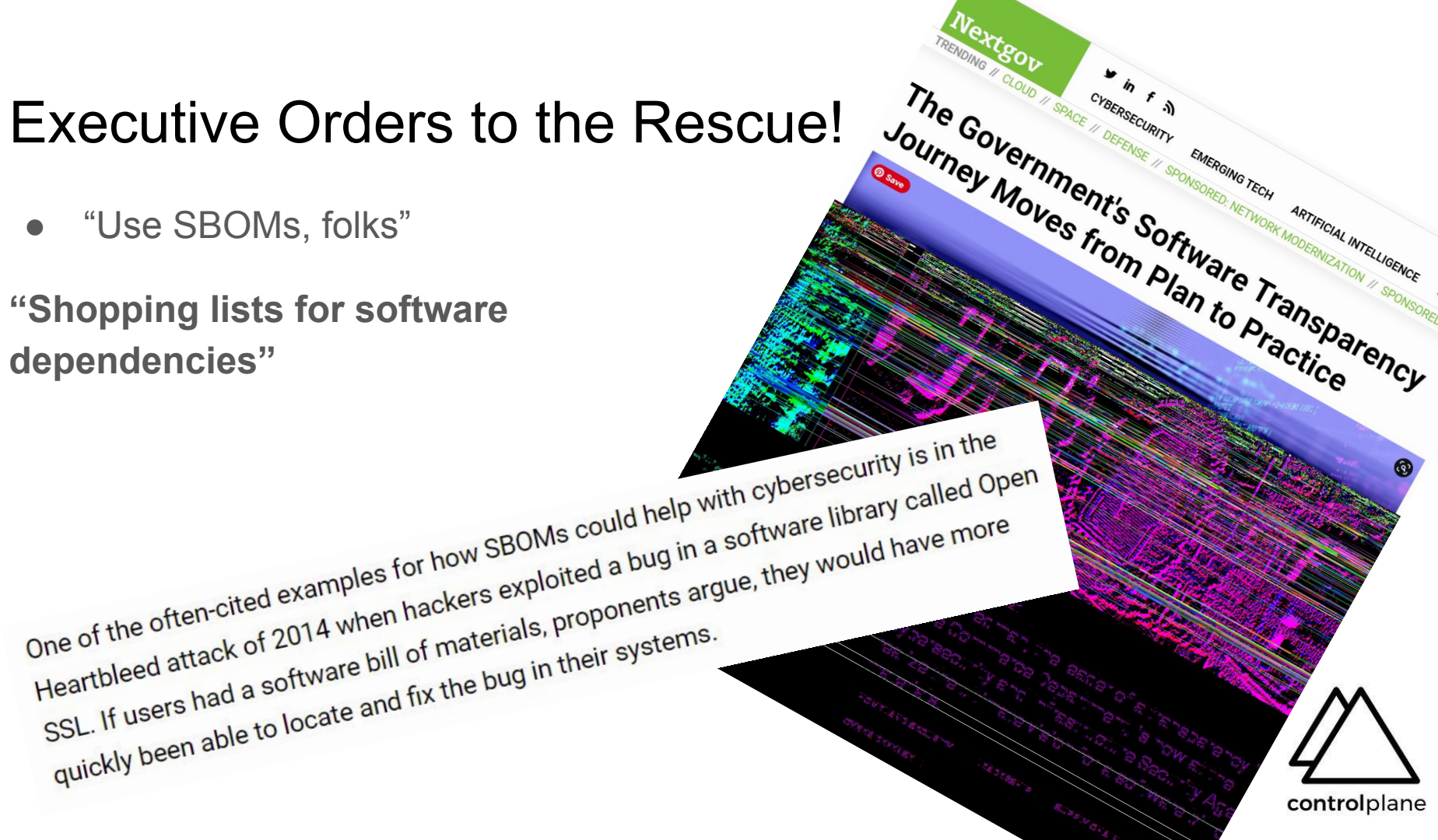
We must also trust our build tools…

- Reflections on Trusting Trust (Thompson) Communications of the ACM, 27:8, Aug 1984

Executive Orders to the Rescue!

- “Use SBOMs, folks”

“Shopping lists for software dependencies”



controlplane

How to Attack a Supply Chain

controlplane

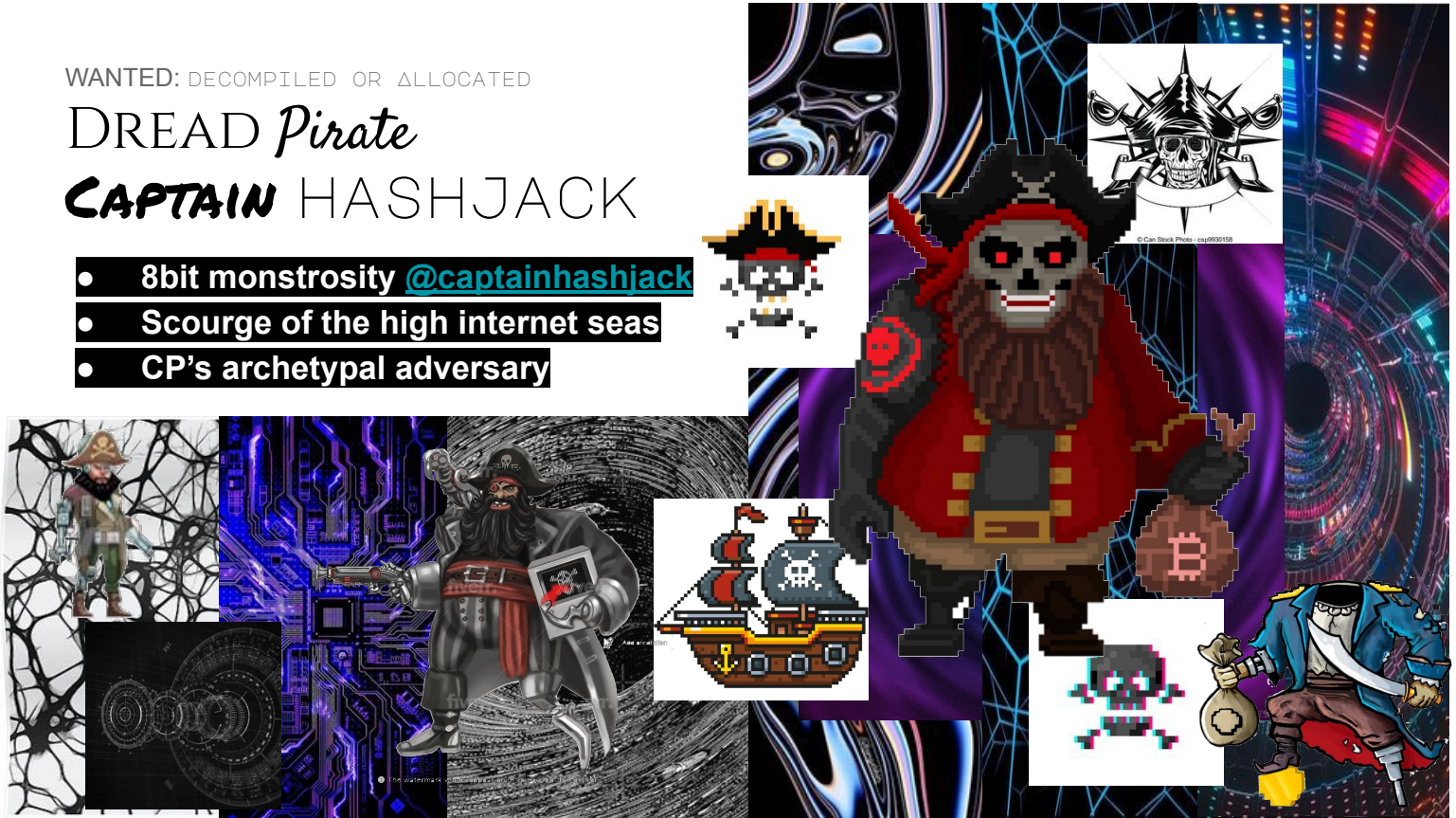
A Pirate's Life for Me: Malicious Code in Production

WANTED: DECOMPILED OR ALLOCATED

DREAD *Pirate*

CAPTAIN HASHJACK

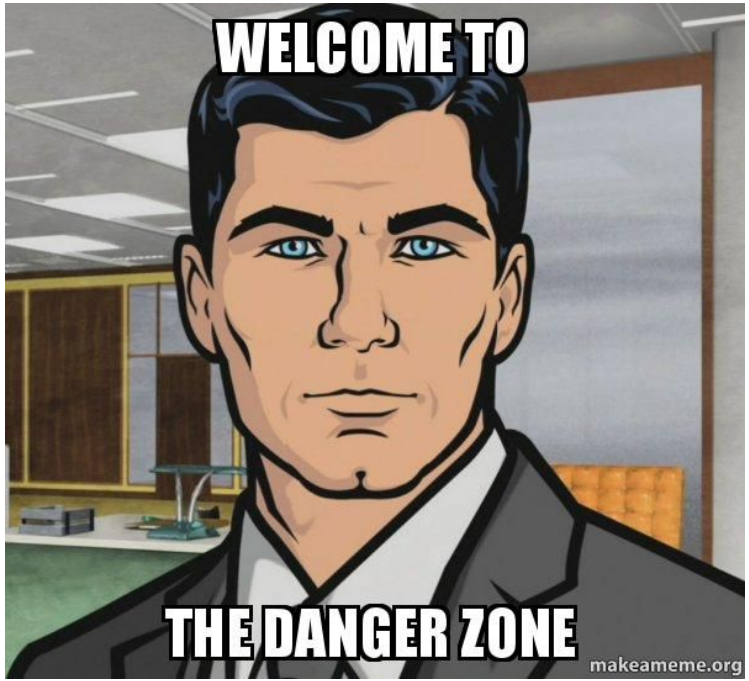
- 8bit monstrosity @captainhashjack
- Scourge of the high internet seas
- CP's archetypal adversary



controlplane

Welcome to the Dangerzone

- REvil (Ransomware Evil; aka Sodinokibi)
- SolarWinds
- Codecov
- Homebrew
- VSCode
- Xcodeghost



controlplane

Catalog of Supply Chain Compromises

- Source Code
- Dev Tooling
- Publishing Infrastructure
- Trust and Signing
- Attack Chaining
- Negligence

https://github.com/cncf/tag-security/tree/main/supply-chain-security/compromises

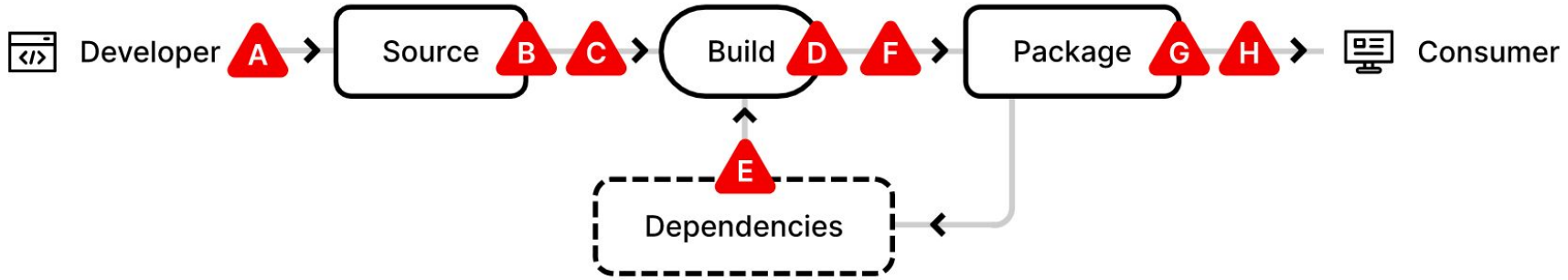
Name	Year	Type of compromise	Link
PHP self-hosted git server	2021	Dev Tooling	1
Homebrew	2021	Dev Tooling	1, 2
Codecov	2021	Source Code	1
VSCode GitHub	2021	Dev Tooling	1
SUNBURST/SUNSPOT/Solarigate	2020	Publishing Infrastructure	1, 2, 3
The Great Suspender	2020	Malicious Maintainer	1,2
Abusing misconfigured SonarQube applications	2020	Dev Tooling	1, 2
Octopus Scanner	2020	Dev Tooling	1,2
NPM reverse shells and data mining	2020	Dev Tooling	1
Webmin backdoor	2019	Dev Tooling	1, 2
purescript-npm	2019	Source Code	1 and 2
electron-native-notify	2019	Source Code	1, 2
PyPI typosquatting	2019	Negligence	1
ROS build farm compromise	2019	Trust and Signing Publishing Infrastructure	1, 2
ShadowHammer	2019	Attack Chaining	1, 2
PEAR Breach	2019	Publishing Infrastructure	1, 2
Canonical's GitHub org compromised	2019	Dev Tooling Source Code Publishing infrastructure	1
The event-stream vulnerability	2018	Malicious Maintainer	1,2
Dofoil	2018	Publishing Infrastructure	1
Operation Red	2018	Publishing Infrastructure	1
RCE in go get -u	2018	Dev Tooling	1, 2
acroread compromised in AUR	2018	Malicious Maintainer	1, 2

How to Attack a Supply Chain

- Infect the source code
  - Developer’s machine & git 🗝
  - Source repository
  - Build infrastructure
- Infect a trusted supplier
  - Infect a dependency the source code pulls in
  - Infect a trusted package that runs as root
- Infect the runtime environment the source is running in
  - Legitimate application code, compromised OS

Every Step You Take: SLSA Dangers

- https://slsa.dev/



A Bypassed code review
B Compromised source control system
C Modified code after source control
D Compromised build platform
E Using a bad dependency
F Bypassed CI/CD
G Compromised package repo
H Using a bad package

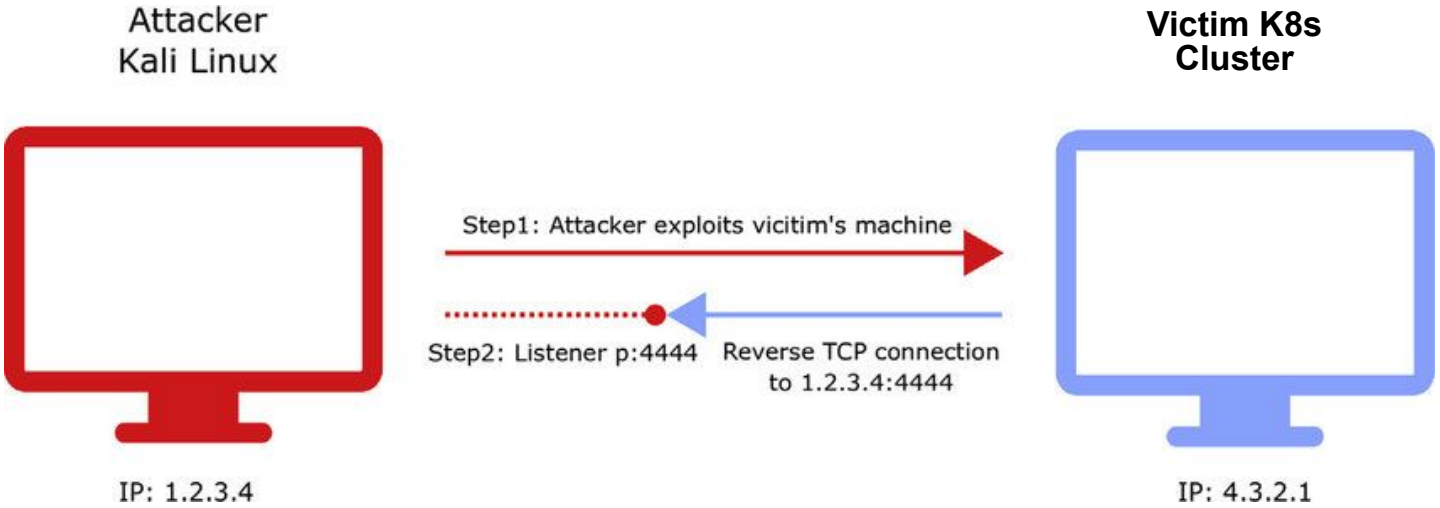
Demo: AppDepSec

- Malware in **application dependency** attacks EUD
- **EUD** -> pulls **malicious NPM package** -> package installer runs **malware**
  - Game over approaching
  - **Steals** identities, **tampers** with code, persists with **keylogger**
- Fix?
  - 2FA for signing keys and SSH
  - Avoid plaintext credentials anywhere
  - Hide yo' cryptowallets
  - Air-gapped development environment (unlikely to be practical for >90% orgs)

controlplane

Demo #2: Container-rama Drama

- Malware in container takes over container in Kubernetes
- **K8s** -> pulls **malicious container image** -> container runs malware
  - Fires reverse shell
  - Gives control of the container to a remote attacker

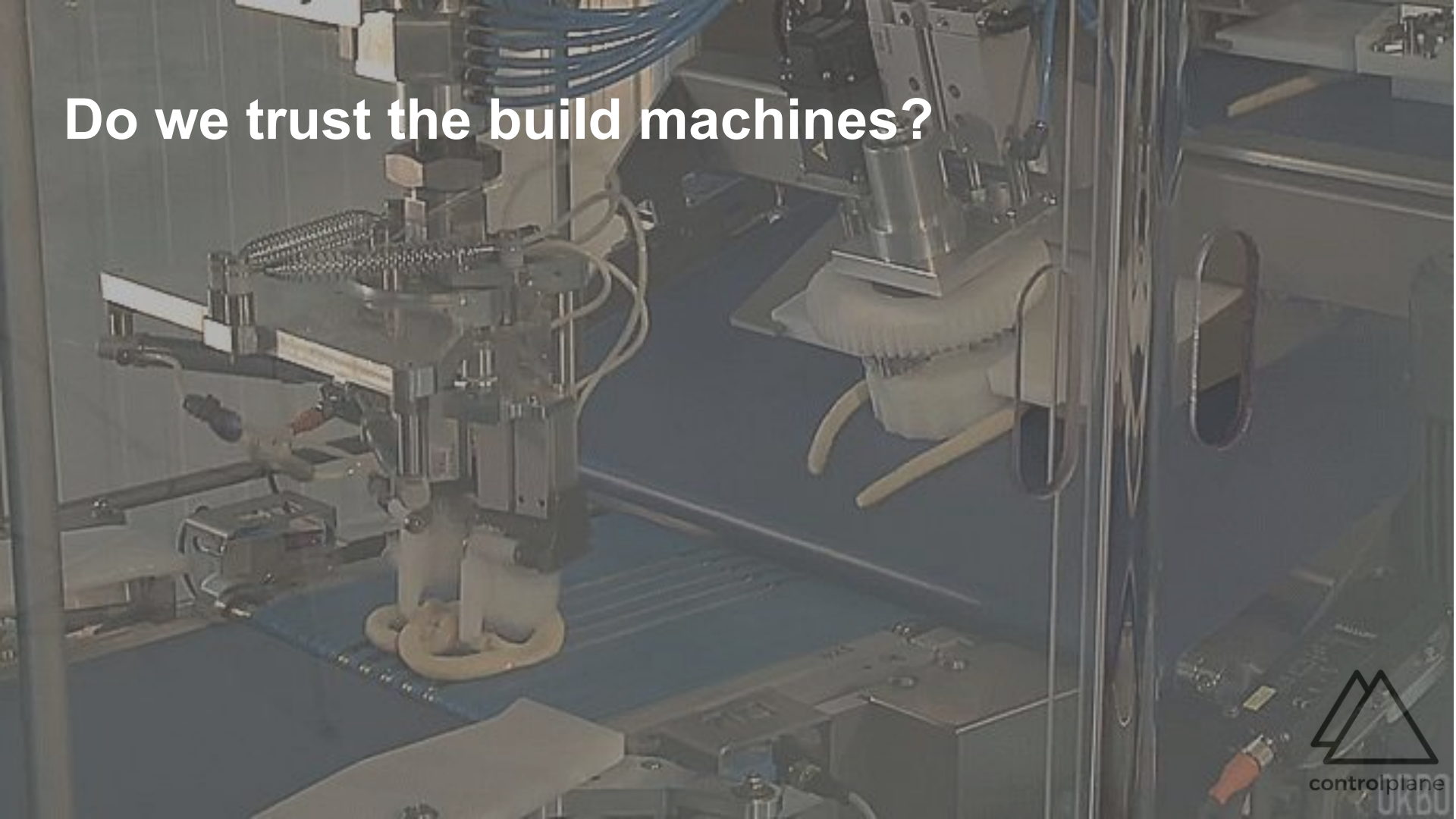


controlplane

Demo #2: Container-rama Drama

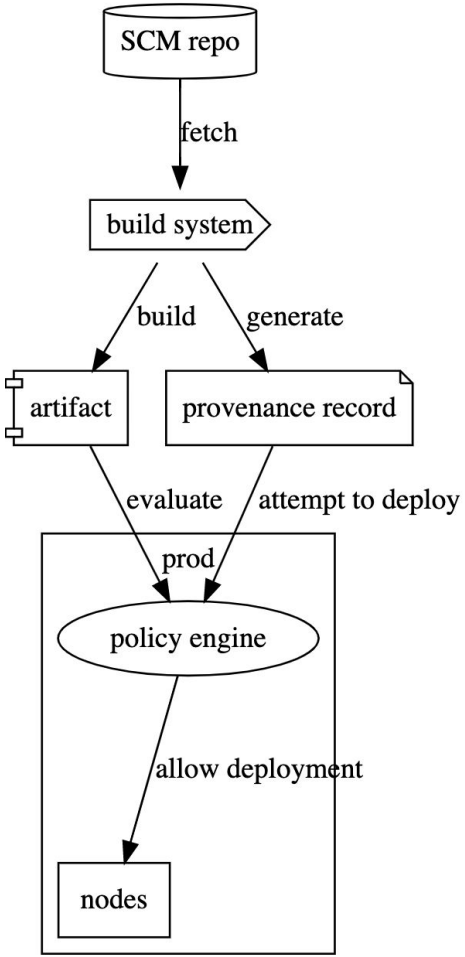
- Malware in container takes over container in Kubernetes
- K8s -> pulls malicious container image -> container runs malware
  - Fires reverse shell
  - Gives control of the container to a remote attacker
- Fix?
  - Image scanning in the pipeline (minimum viable cloud native security)
    - Trivy and crew
  - IDS (image scanning can't catch everything)
    - Falco and pals
  - Build-time behavioural analysis (trust must be earned)
    - Tracee and eBPF friends
  - **PROPER POD `securityContexts` and CAPABILITIES** (hashtag https://kubesec.io/)
    - Tracee, oci-seccomp-bpf-hook

controlplane


Do we trust the build machines?
controlplane

Policy and Attestation

- Build an artefact
- Generate provenance
- Verify at production deployment



https://dlorenc.medium.com/policy-and-attestations-89650fd6f4fa

controlplane

Other Demos: Build Infrastructure Compromise

SupplyChainSecurityCon!

Getting Started with Supply Chain Security is Easier Than You Think: Perspectives From a Highly Regulated Industry - Michael Lieberman & Timothy Miller, CitiBank

https://github.com/mlieberman85/supply-chain-examples

Other Demos: Build Infrastructure Compromise

```
mlieberman@dev:~/Projects/supply-chain-examples/kubernetes/scripts$ kubectl run foo --image ghcr.io/mlieberman85/kaniko-chains-demo
:latest
Error from server: admission webhook "mutate.kyverno.svc" denied the request:

resource Pod/default/foo was blocked due to the following policies

verify-image:
  verify-image: 'image verification failed for ghcr.io/mlieberman85/kaniko-chains-demo:latest:
    failed to verify image: fetching signatures: getting signature manifest: GET https://ghcr.io/v2/mlieberman85/kaniko-chains-demo
/manifests/sha256-6767c6093b7be776e1ebc9f91676cab3a9db9c2b9798d5f594437e6d63251d41.sig:
    MANIFEST_UNKNOWN: manifest unknown'
```

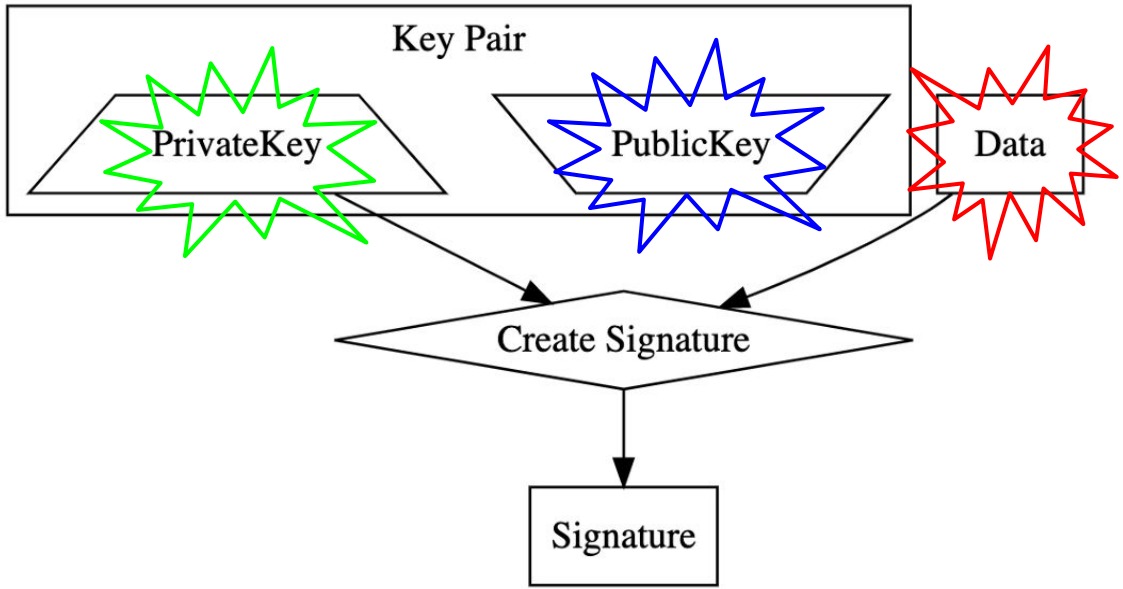
controlplane

Sign me to the Moon

controlplane

What is Signing?

- Take 1 item of "data"
  - Could be a file, or
  - Structured data about other files, or
  - A container image, tarball, or other collection of files, or
  - ...anything else
- Combine it with a private key to create a signature
  - That can be verified with the private key's **public part**
  - Public key distributed to users
- Distribute signature alongside data

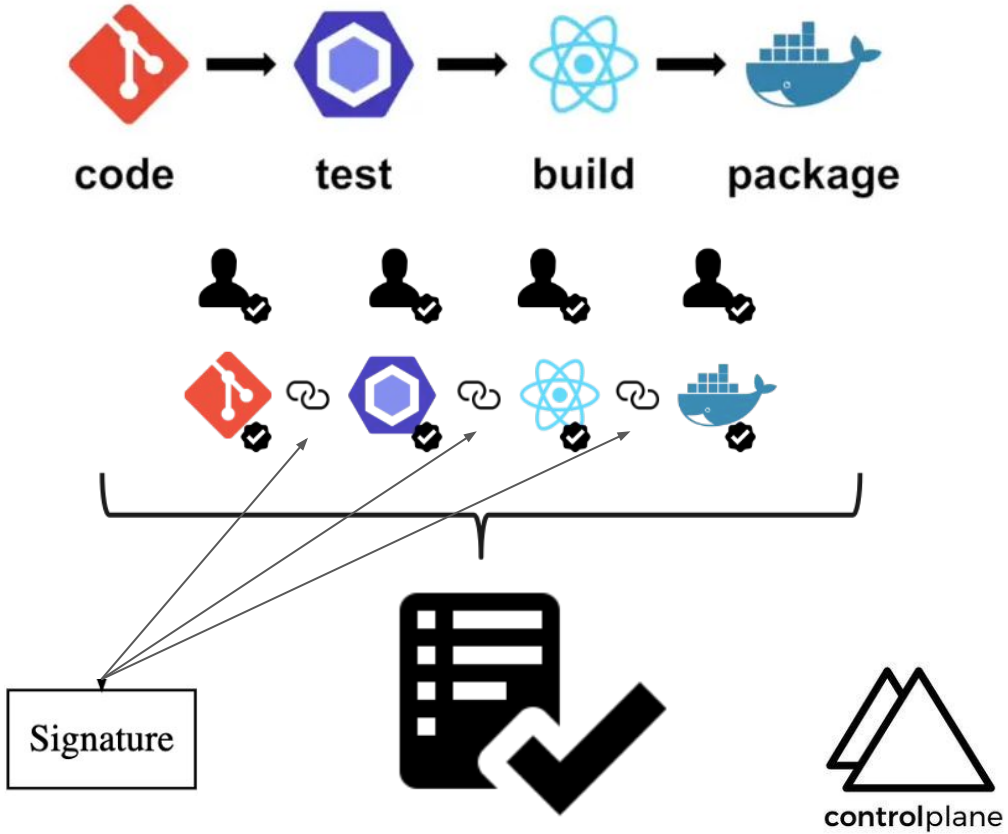


https://dlorenc.medium.com/signature-formats-9b7b2a127473

controlplane

in-toto

- Checks for **compliance** using **Link metadata** and the **Layout metadata**
- in-toto doesn't care what you're verifying
  - It's just verifying a **chain of signatures**

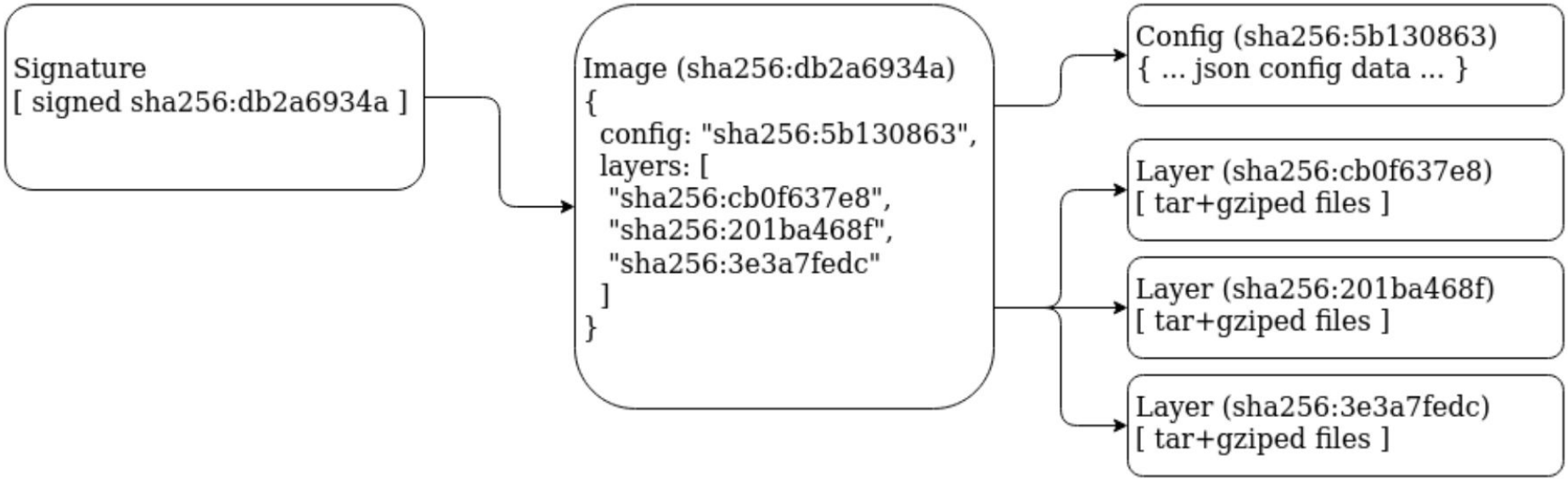


in-toto: layouts

```
"_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" : [...]
```

controlplane

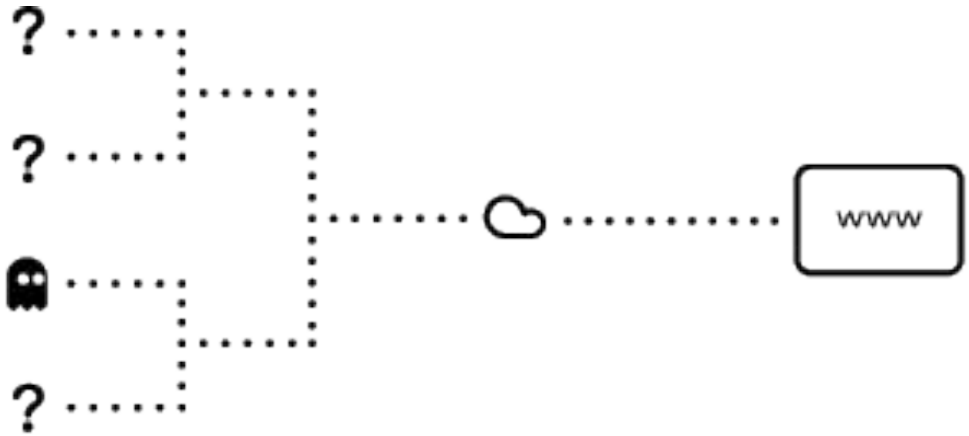
Signing Container Images



controlplane

sigstore

- rekor
  - Transparency logs for metadata and signatures
  - Validate other people's signatures
- cosign
  - Signing for container images

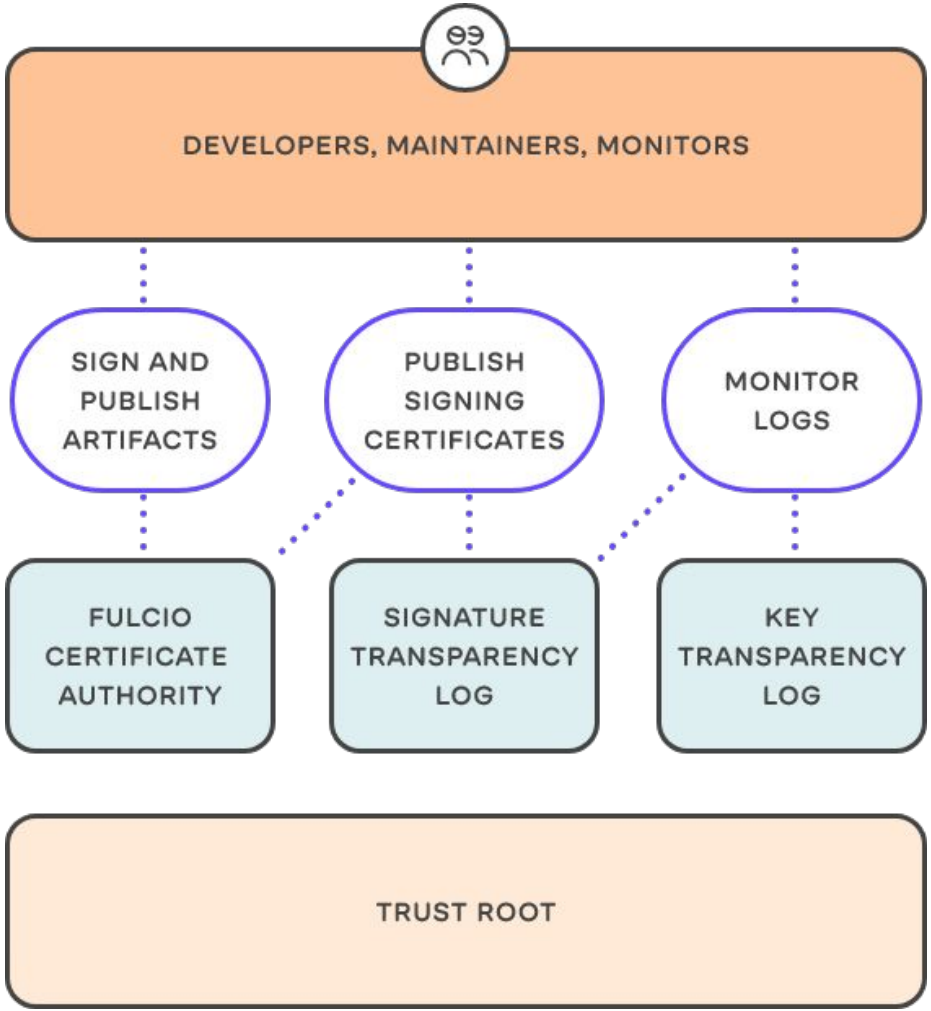


Notary v2 will also enter this space

controlplane

sigstore Ecosystem

- This is about trusting people
- Software does not compromise itself
- Beware of companies that don't provide transparency into their software composition

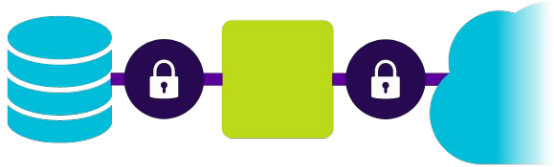


controlplane

SPIFFE: universal identity

- A **universal identity control plane** for distributed systems
- **Cryptographic identity** of workload can be **confirmed from the root of trust**
- **SPIRE** is an implementation of SPIFFE
- Cryptographic proofs from the software factory enable **runtime and network security**

Secure microservices communication automatically with Envoy, X.509 PKI, or JWT



Authenticate securely to common databases or platforms without passwords or API keys

Build, bridge, and extend **service mesh** across organizations without sharing keys

https://spiffe.io/

controlplane

https://github.com/boxboat/in-toto-golang

- in-toto fork with hooks into the SPIRE workload API
- verify signatures based on certificate constraints
- wraps all build commands with in-toto

https://boxboat.com/2021/05/04/supply-chain-security-by-verification/

controlplane

The Software Factory

controlplane

Software Factory

- Build pipelines for other build pipelines
- CI/CD at hyperspeed
- Aggressive automation enables more secure systems
- Welcomes many signing approaches

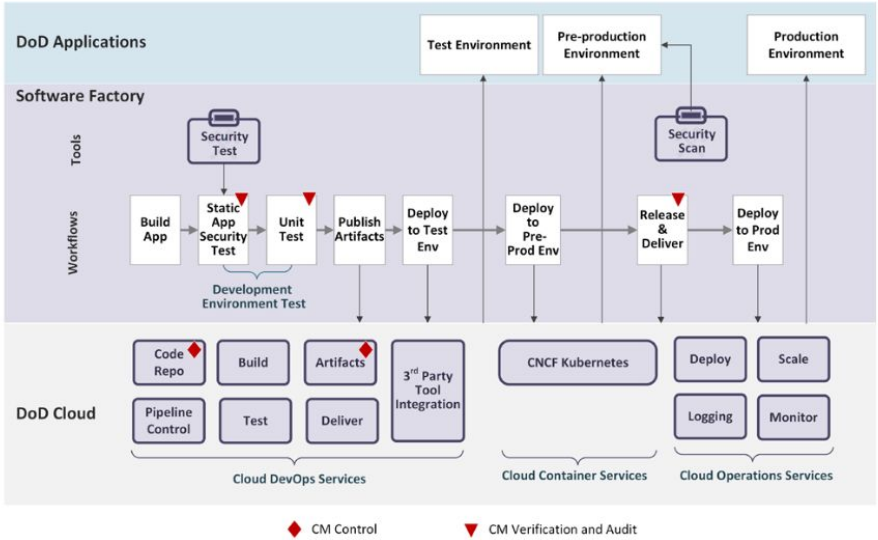
Layers

- Infra
  - Hardware, cloud, K8s
  - Alerting and monitoring integration
- Platform
  - User-facing DSL (a la https://sap.github.io/jenkins-library/)
  - CI orchestrator (Tekton, Jenkins, another layer)
  - Build step containers for CI and local (git, lint, test, sign, etc)
  - Remote build steps (scanning source, containers, etc)
  - Chain of trust (verificable signatures)
- Data
  - Git apps/config, containers, OS/app packages/deps, XML feeds, …
- App
  - Delivery target (web-facing, batch, async, etc)

controlplane

Building Systems with a Software Factory

- Multiple CI/CD pipelines can be composed into a **complex build and deployment system**
- This is called a "Software Factory", and is used to **securely build and deploy all components of a system**, until the entire system is composed and deployed
- **Everything is automated, including security**

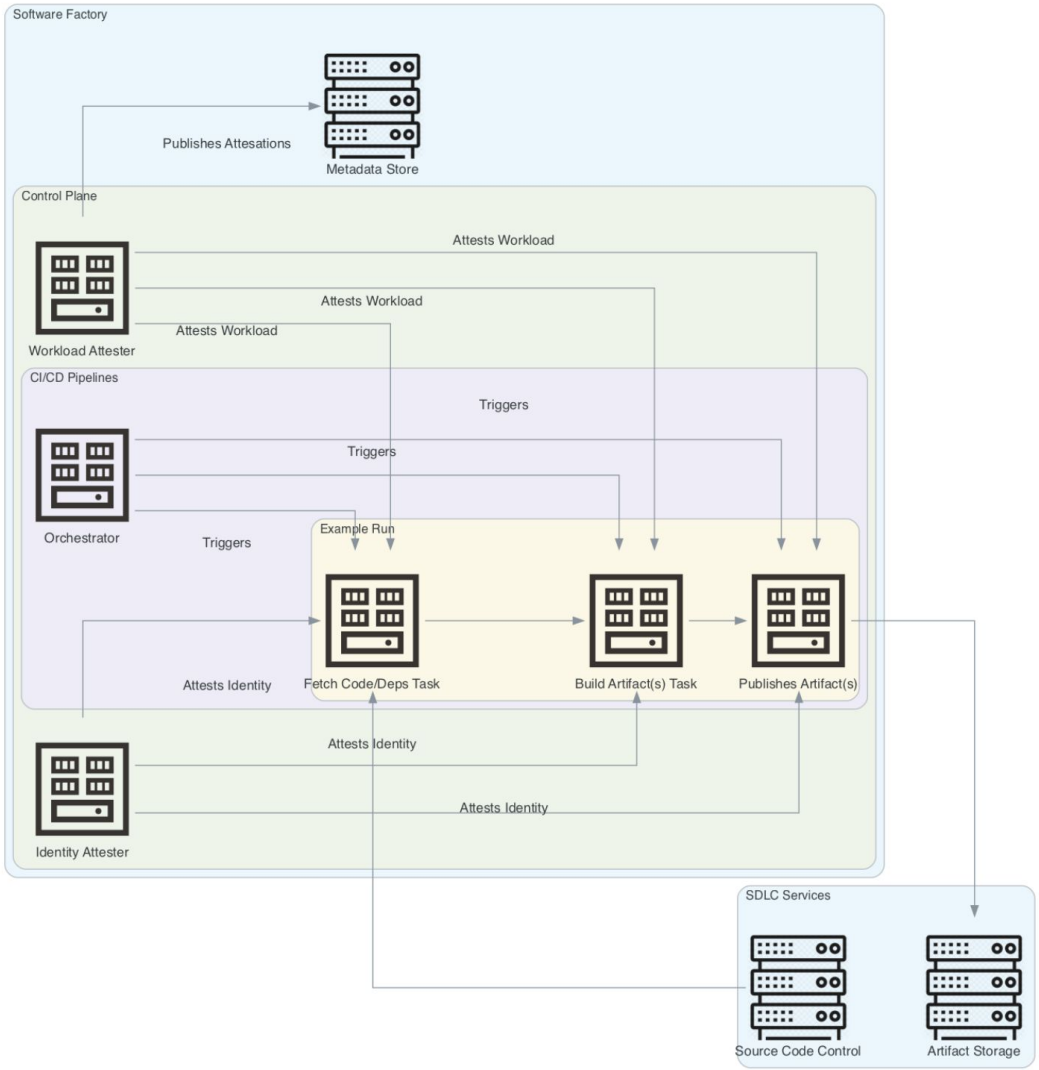


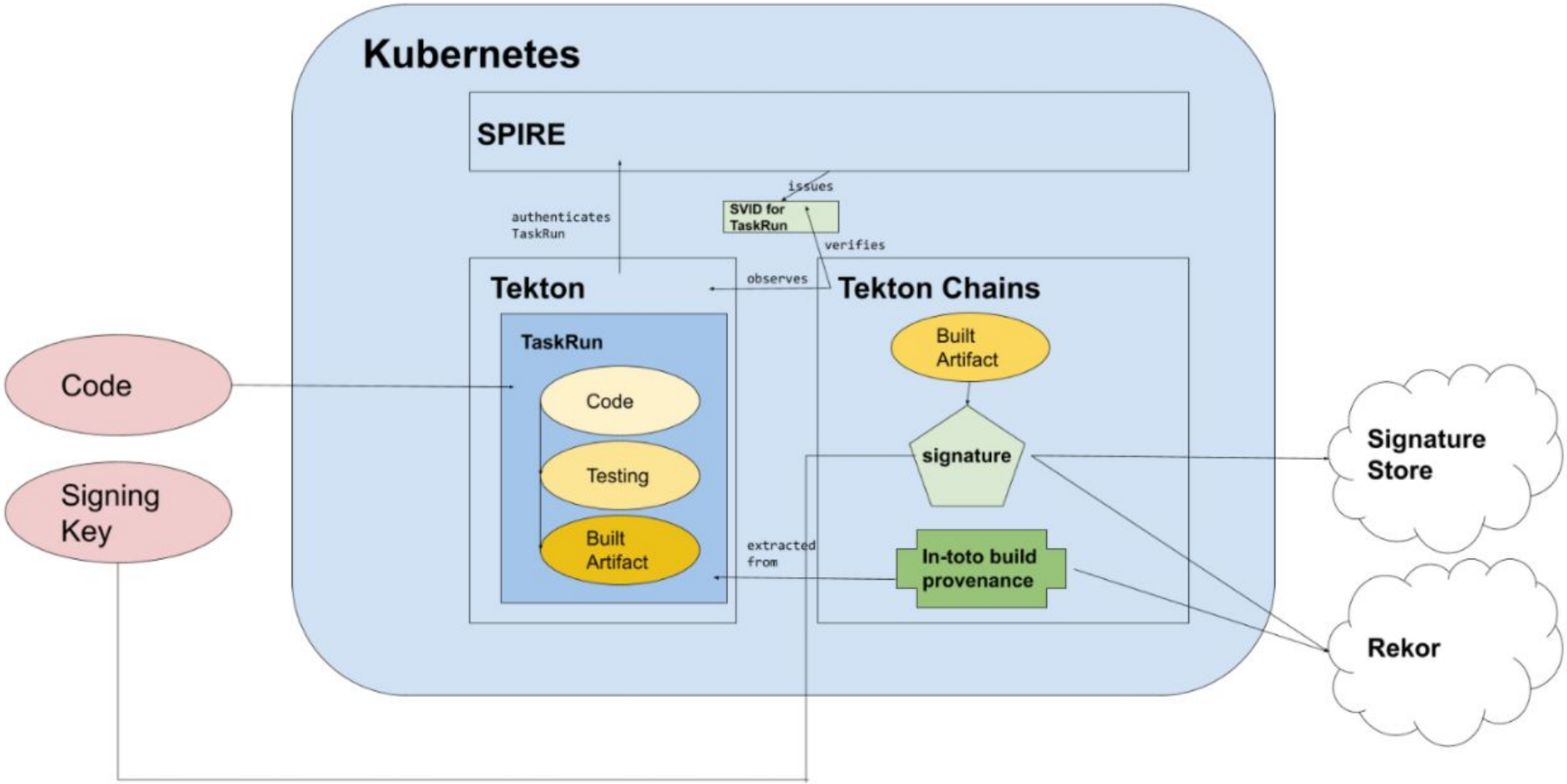
[DoD Enterprise DevSecOps Reference Design v1.0]

controlplane

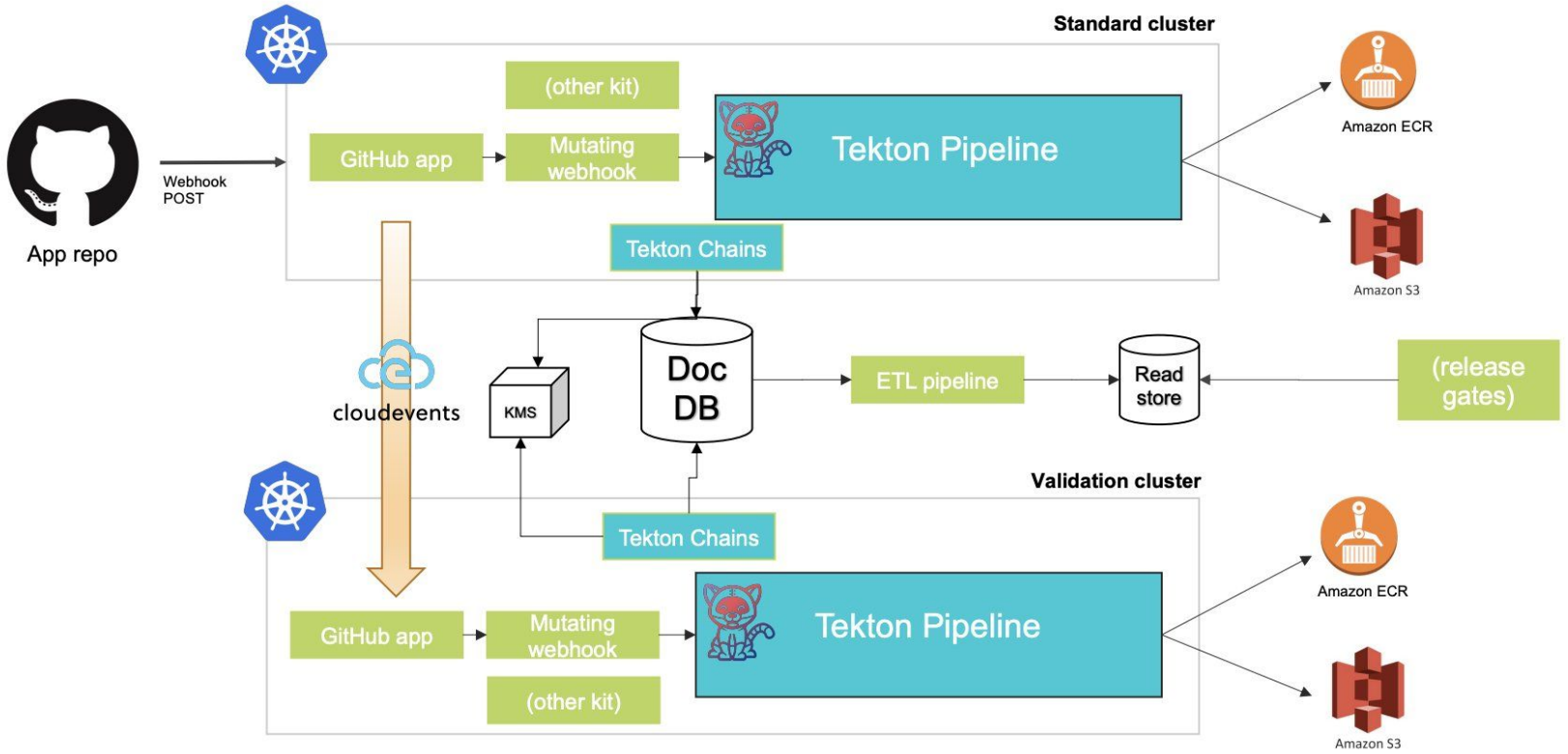
Secure Software Factory Reference Architecture

[Secure Software Factory Reference Architecture]




Kubernetes
SPIRE
issues
SVID for TaskRun
authenticates TaskRun
verifies
Tekton
observes
Tekton Chains
TaskRun
Code
Testing
Built Artifact
Built Artifact
signature
extracted from
In-toto build provenance
Code
Signing Key
Signature Store
Rekor

SolarWinds Trebuchet: Redundant Tekon Pipelines



Whitepaper: Zero-Trust Supply Chains

- https://github.com/sigstore/community/blob/main/docs/zero-trust-supply-chains.pdf

Zero-Trust Supply Chains

Dan Lorenc
June 26th 2021

Introduction

Supply-chain security and technology have lagged behind network and service security for a number of reasons, **and it's time to fix that!** Zero-trust technologies have dramatically improved and simplified other forms of enterprise infrastructure, but haven't been applied to supply-chain security yet. This document explores how zero-trust could be applied to build systems, through the Sigstore and SPIFFE/SPIRE projects.

TektonCD is used as an example build system and In-Toto as an example provenance format. Other systems and formats would work as well.

controlplane


controlplane
https://control-plane.io/
w.JJ.AM