SUPPLY CHAIN SECURITY KUBERNETES

MICHAEL HAJJAR
CLOUDYLINUX

INHOUDSOPGAVE

Part 1: digital identity

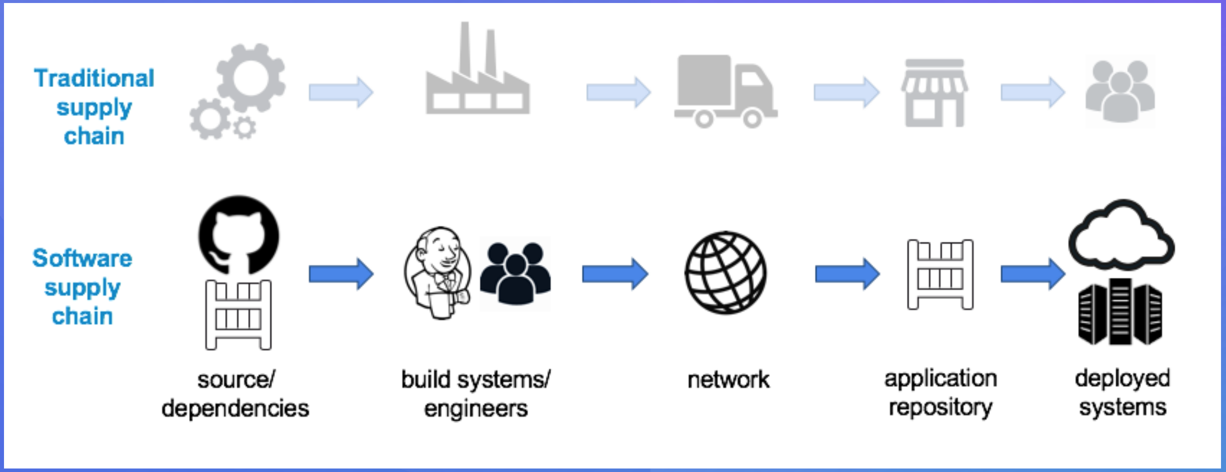
- Supply Chain?
- Supply Chain Security?
- Enter SLSA
- Enter Sigstore

Part 2: Scanning & hardening

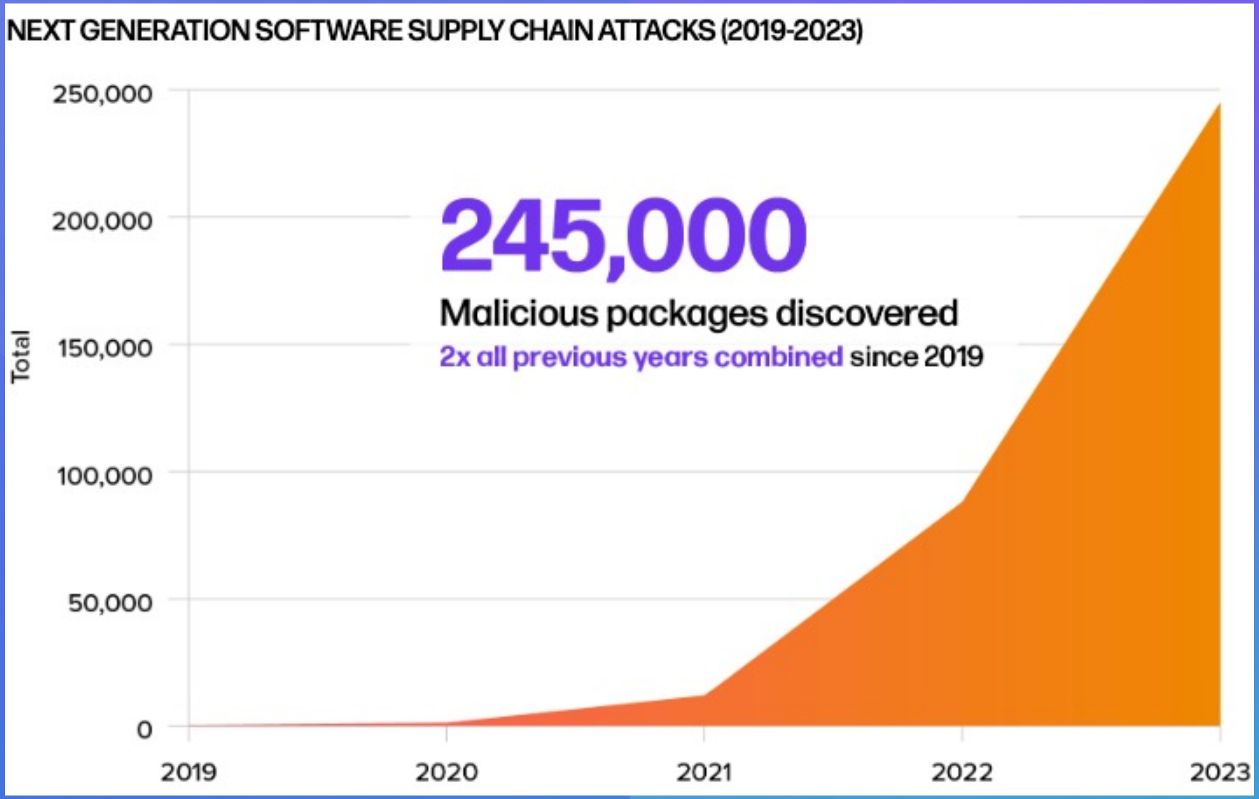
- Container images
- Scanning
- Enter ChainGuard
- QA

INCOMING BUZZWORD

SUPPLY CHAIN?

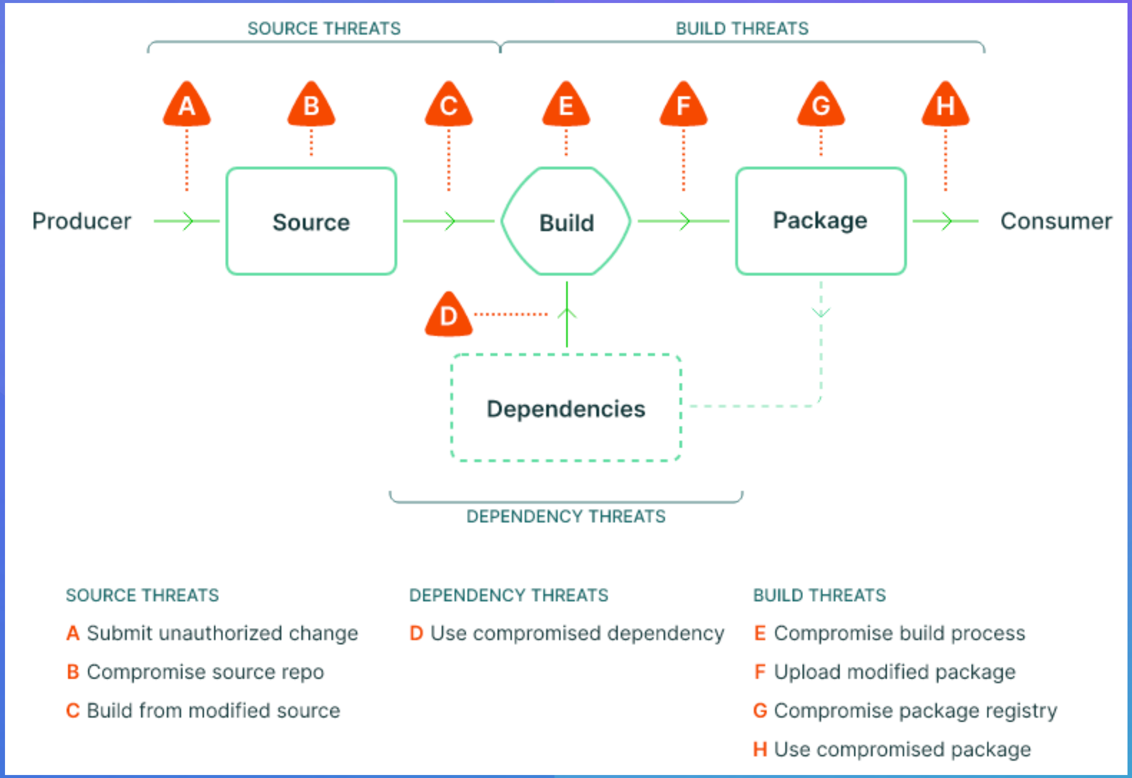


SUPPLY CHAIN SECURITY?



Bron: Sonatype annual state of Software Supply Chain 2023

ENTER SLSA



SOURCE THREATS

A Submit unauthorized change

B Compromise source repo

C Build from modified source

DEPENDENCY THREATS

D Use compromised dependency

BUILD THREATS

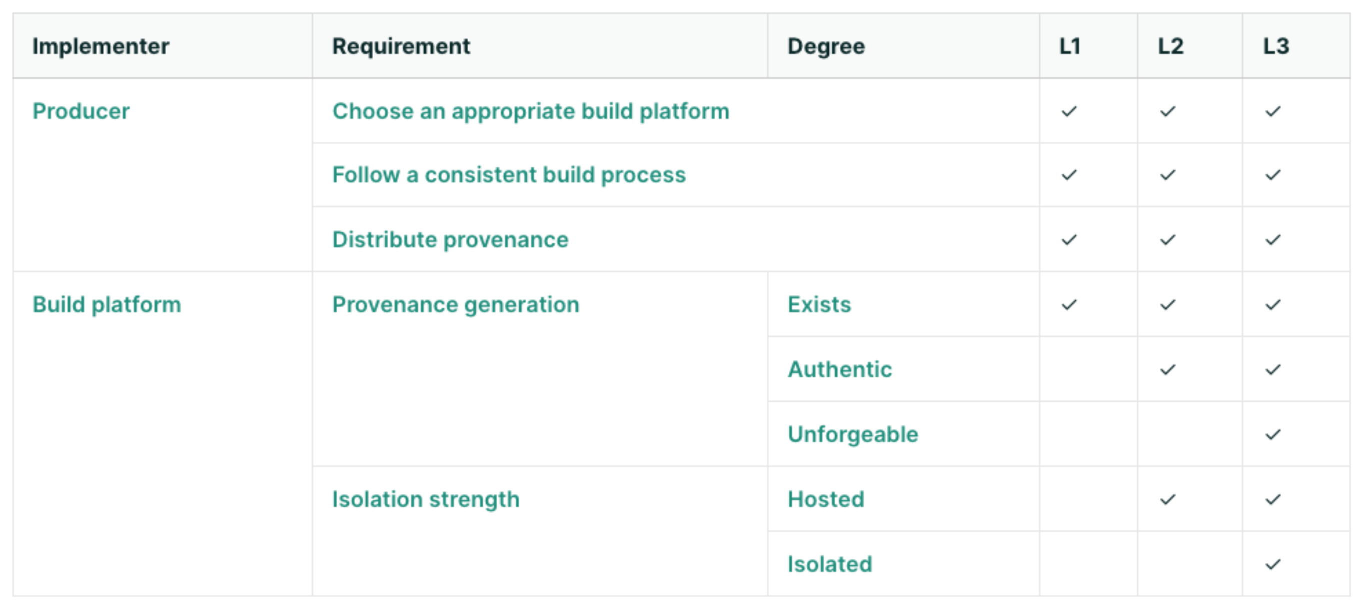
E Compromise build process

F Upload modified package

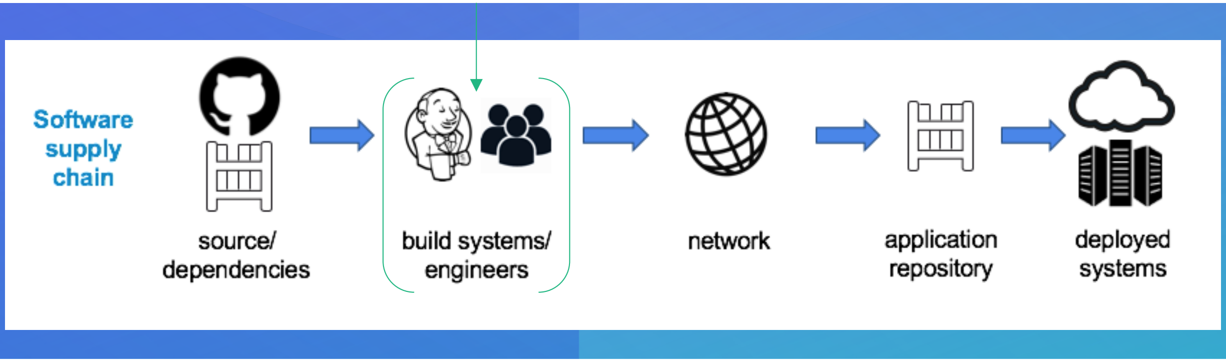
G Compromise package registry

H Use compromised package

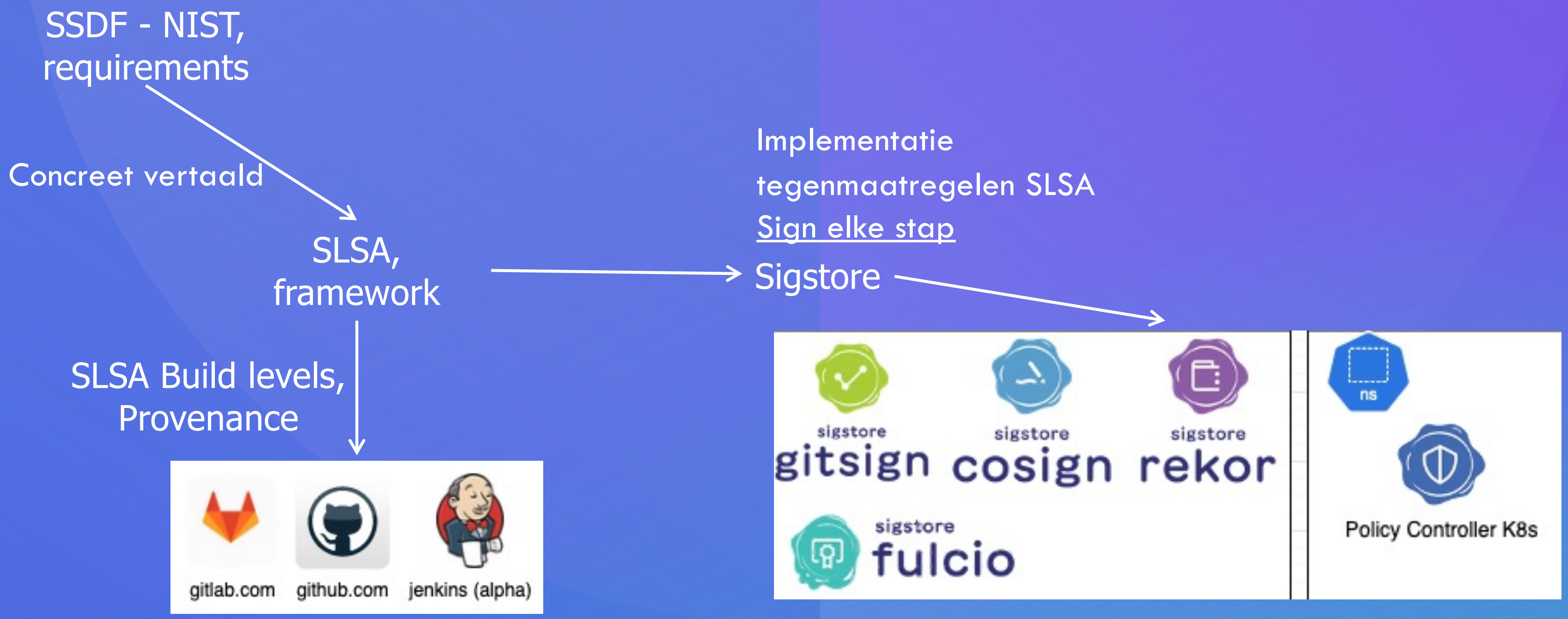
SLSA BUILD LEVELS



Implementer	Requirement	Degree	L1	L2	L3
Producer	Choose an appropriate build platform		✓	✓	✓
	Follow a consistent build process		✓	✓	✓
	Distribute provenance		✓	✓	✓
Build platform	Provenance generation	Exists	✓	✓	✓
		Authentic		✓	✓
		Unforgeable			✓
	Isolation strength	Hosted		✓	✓
		Isolated			✓



ENTER SIGSTORE



SSDF - NIST, requirements

Concreet vertaald

SLSA, framework

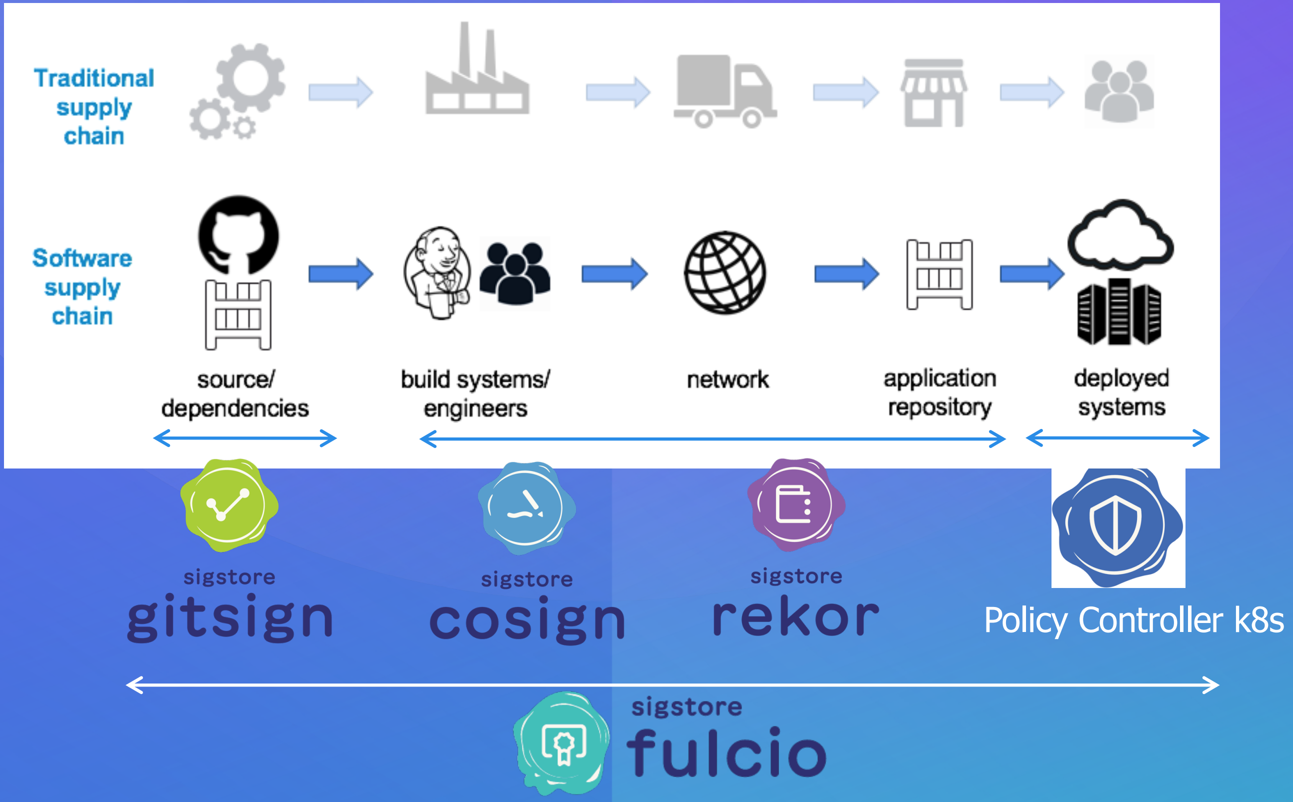
SLSA Build levels, Provenance

Implementatie tegenmaatregelen SLSA

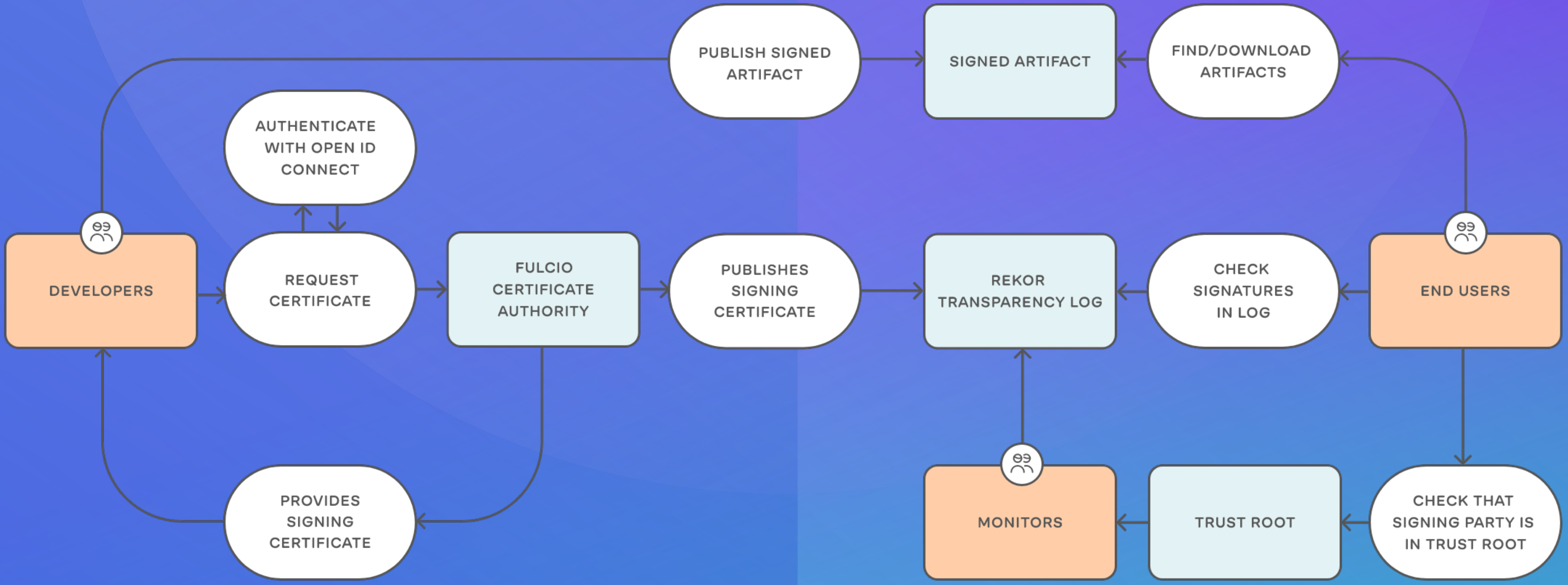
Sign elke stap

Sigstore

TOEGEPAST


Traditional supply chain
Software supply chain
source/ dependencies
build systems/ engineers
network
application repository
deployed systems
sigstore gitsign
sigstore cosign
sigstore rekor
Policy Controller k8s
sigstore fulcio

SIGSTORE HAPPY FLOW



DEVELOPERS

AUTHENTICATE WITH OPEN ID CONNECT

REQUEST CERTIFICATE

FULCIO CERTIFICATE AUTHORITY

PUBLISHES SIGNING CERTIFICATE

PROVIDES SIGNING CERTIFICATE

PUBLISH SIGNED ARTIFACT

SIGNED ARTIFACT

FIND/DOWNLOAD ARTIFACTS

REKOR TRANSPARENCY LOG

CHECK SIGNATURES IN LOG

END USERS

CHECK THAT SIGNING PARTY IS IN TRUST ROOT

TRUST ROOT

MONITORS

CONCLUSIE PART 1

- Supply Chain Security risico’s
- SLSA build security – Provenance
- Sigstore – elke stap signen binnen SSCS en controleren
- Kubernetes – controle op **Pods/Container Images**

CONTAINER IMAGES

Grote surface, lage beheerlast

debian
Normal
/slim

Ubi/micro

SUSE
Linux Enterprise 15

Alpine Linux /
BusyBox

kleine surface, grote beheerlast

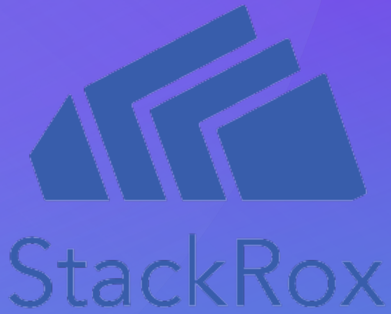
Google Distroless
(Debian)

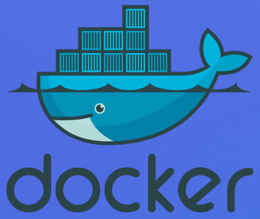
chiselled
Ubuntu

ko
KO Go

Docker Scratch

VULN. SCANNING





Docker
Scout

NeuVector
BY SUSE

PROBLEEM / KANS

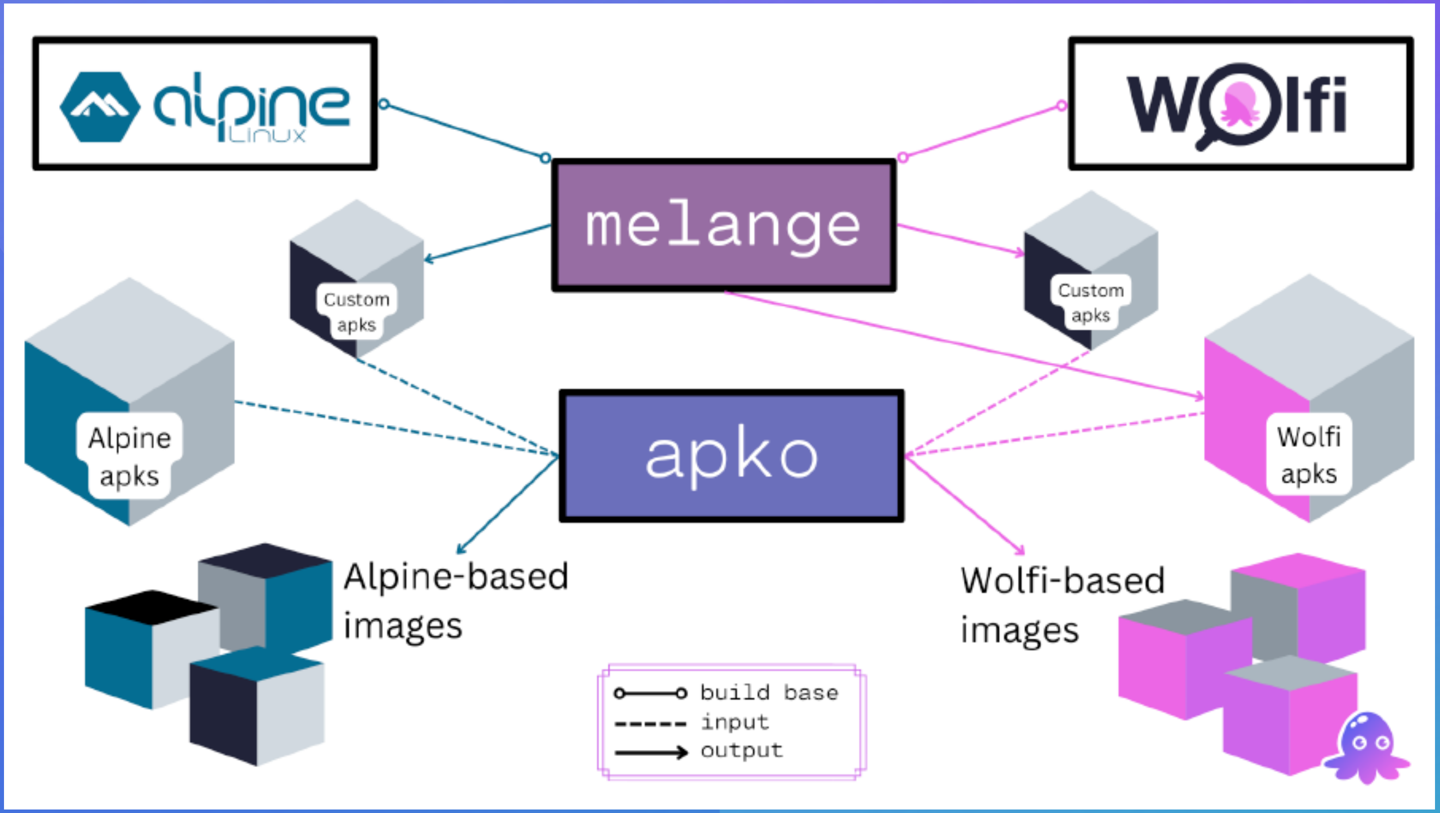
Probleem:

- Google Distroless te complex (Bazel)
- Scratch geen optie (trucjes uithalen)
- Coverage niet 100% van SBOM/Scans
- Beheerlast verminderen
- Complexiteit Sigstore/

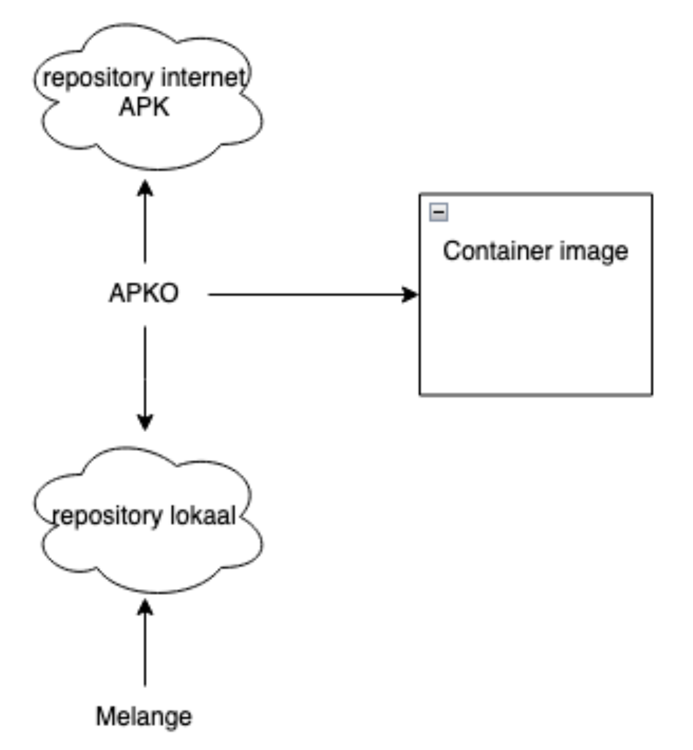
Oplossings richting:

- Alles via package manager laten gaan
- SBOM by-design
- Proces versimpelen image hardening
- In YAML

ENTER CHAINGUARD



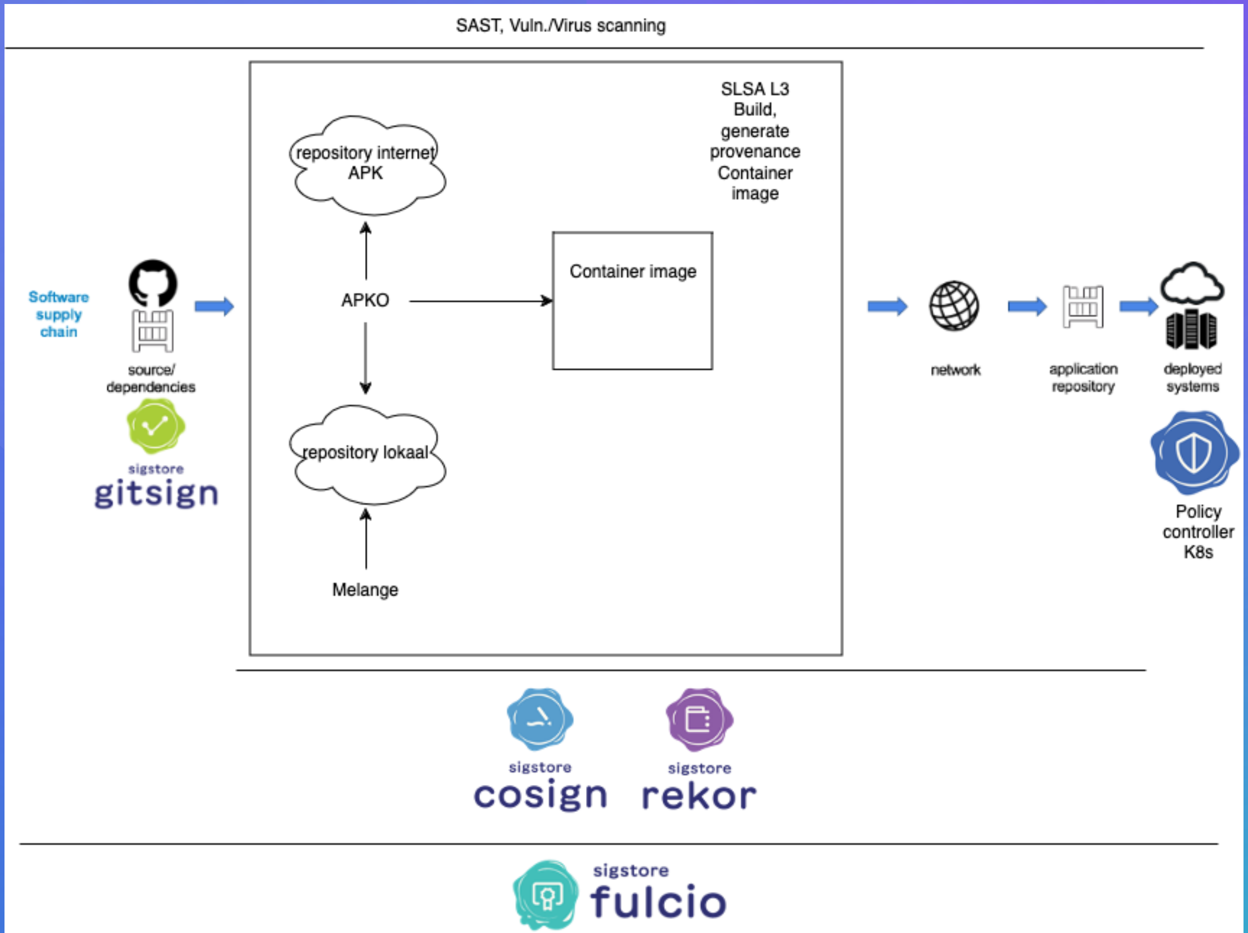
CHAINGUARD HAPPY FLOW



CONCLUSIE PART 2

- Attack surface verminderen
- Coverage SBOM/Scans verhogen
  (alles via Pkg manager doen)
- Bewuste keuze maken over gebruik base-images
  en controleren leverancier base-images

COMBINATIE PART 1/2



SAST, Vuln./Virus scanning

Software supply chain

source/ dependencies

sigstore gitsign

SLSA L3 Build, generate provenance Container image

repository internet APK

APKO

Container image

repository lokaal

Melange

network

application repository

deployed systems

Policy controller K8s

sigstore cosign

sigstore rekor

sigstore fulcio

QA

Bronnen / Credits:
https://edu.chainguard.dev/
https://slsa.dev
https://sigstore.dev
https://www.sonatype.com