trutz.io

SPICKZETTEL · GEHÄRTETES K3S

Gehärtetes k3s

Die vier Config-Zeilen, die vor der Installation auf der Platte liegen, die Admission-Datei, die Prüf-Kommandos und das Wegwerf-Rezept. Begleitmaterial zum Video: ein k3s-Cluster, der von seiner ersten Sekunde an gehärtet ist.

DIE EINE REGEL, AUS DER ALLES FOLGT

Ein gehärteter Cluster ist einer, der nie ungehärtet war.

01 Die vier Zeilen, bevor k3s startet

Video 03:29

```
# cloud-init legt sie auf die Platte,
# bevor die Installation läuft:
# /etc/rancher/k3s/config.yaml
secrets-encryption: true
tls-san:
  - 10.101.0.1
disable:
  - traefik
kube-apiserver-arg:
  - admission-control-config-file=/etc/rancher/k3s/
psa.yaml
```

Erst schreibt cloud-init die Config, dann installiert `runcmd` k3s mit **gepinnter Version**. Es gibt keine Sekunde ungehärtetes Kubernetes: **nie ungehärtet gewesen.**

02 secrets-encryption nachweisen

Video 05:04

```
kubectl create secret generic db-password \
  --from-literal=password=sesam-oeffne-dich-4711
# dieselbe Suche wie am Anfang:
sudo strings /var/lib/rancher/k3s/server/db/state.db \
  | grep -c sesam
# 0 Treffer, statt Klartext
sudo k3s secrets-encrypt status
# Encryption Status: Enabled
```

base64 ist eine Kodierung, keine Verschlüsselung. `secrets-encryption` schützt die Wege, auf denen die Datenbank den Server verlässt: **Backup, Snapshot, kopiertes Volume, alte Platte.** Nachträglich einschalten verschlüsselt nur Neues.

03 Pod Security Standards scharf ab Pod 1

Video 08:28

```
# /etc/rancher/k3s/psa.yaml
defaults:
  enforce: baseline
  warn: restricted
  audit: restricted
exemptions:
  namespaces:
    - kube-system
```

Baseline durchgesetzt: kein privileged, kein Host-Netz, keine Host-Pfade. Ein privilegierter Pod wird **abgelehnt, bevor er entsteht.** Dazu gehört RBAC, sonst labelt ein Benutzer seinen Namespace einfach auf `privileged` zurück.

04

Video 10:20

Der Zombie-Pod: nachträglich reicht nicht

```
# Namespace nachträglich scharf:
kubectl label --overwrite ns nachtraeglich \
  pod-security.kubernetes.io/enforce=baseline
# der vorher gestartete privilegierte
# Pod läuft trotzdem weiter:
kubectl get pod -n nachtraeglich
# altlast   Running
```

Pod Security Standards prüfen **beim Erzeugen**, nicht im Betrieb. Was schon läuft, prüft niemand nach. Erst der nächste Rebuild greift. Wer vor dem ersten Start härtet, muss nicht wissen, was aus der ungehärteten Zeit übrig ist.

05

Video 13:14

API dicht, Cluster zum Wegwerfen

```
# von außen: kein Port 6443
nc -z -v -w3  6443
# Connection timed out
# wegwerfen und neu bauen:
tofu destroy -target=hcloud_server.lab
tofu apply
# frische Zertifikate: kubeconfig neu
ssh-keygen -R 10.101.0.1
scp root@10.101.0.1:/etc/rancher/k3s/k3s.yaml \
  ~/.kube/config-lab
sed -i 's/127.0.0.1/10.101.0.1/' ~/.kube/config-lab
```

`tls-san` trägt die Tunnel-Adresse ins Zertifikat, die API ist nur durch WireGuard erreichbar. Nach dem Neubau ist alles frisch: gleiche IP, gleiche Version, gleiche Härtung, **weil jede Zeile in Git liegt.**

Der Cluster von vorhin ist tot, und es vermisst ihn niemand. Genau das ist ein Cluster, den du besitzt.

Dieser Spickzettel gehört zum Video über den gehärteten Wegwerf-Cluster. Jede Woche praxisnahe Antworten auf Docker- und Kubernetes-Probleme: der Newsletter.

trutz.io

Newsletter