OMNI52 RKE2

Architektur

Ein Binary, zwei Rollen

rke2-server / rke2-agent: je eine systemd-Unit, ein statisch gelinktes Binary. Der Supervisor startet und ueberwacht die eigene **containerd**-Instanz, das **kubelet** und – auf Servern – die Control-Plane als statische Pods.
Kein Docker: eigenes containerd unter **/run/k3s/containerd**, Socket **containerd.sock**.

Server-Node

embedded etcd: Default-Datastore (Raft), Quorum ueber ungerade Server-Zahl.
Control-Plane als static Pods: apiserver, controller-manager, scheduler, etcd, cloud-controller – Manifeste in **/var/lib/rancher/rke2/agent/pod-manifests**, vom kubelet gestartet (nicht in-process wie k3s).

CNI & Netzwerk

Default Canal (Calico-Policy + Flannel-VXLAN). Alternativen per **cni:**-Feld: Cilium, Calico, Multus (Multi-NIC, kombiniert mit einem Meta-CNI).
Bundled: CoreDNS, ingress-nginx, metrics-server, snapshot-controller – alle als HelmChart-CRs ausgerollt.

```
export KUBECONFIG=/etc/rancher/rke2/rke2.yaml
export PATH=$PATH:/var/lib/rancher/rke2/bin
kubectl get nodes -o wide
systemctl status rke2-server
```

Installation

Server aufsetzen

Installer zieht Binary + systemd-Unit, startet aber nicht selbst:

```
curl -sfL https://get.rke2.io | sh -
systemctl enable --now rke2-server
```

Agent: **INSTALL_RKE2_TYPE=agent** vor dem Installer, dann **rke2-agent** aktivieren.

Wichtige Pfade

Config: **/etc/rancher/rke2/config.yaml**
kubeconfig: **/etc/rancher/rke2/rke2.yaml**
Binaries (kubectl, crictl, ctr): **/var/lib/rancher/rke2/bin**
Node-Token: **/var/lib/rancher/rke2/server/node-token**

```
# /etc/rancher/rke2/config.yaml (erster Server)
token: <shared-secret>
tls-san:
  - rke2.example.com      # LB / VIP-Hostname
  - 10.0.0.10
write-kubeconfig-mode: "0640"
cni: canal
node-taint:
  - "CriticalAddonsOnly=true:NoExecute"
disable:
  - rke2-ingress-nginx
```

HA Control-Plane

Quorum & Registration

Ungerade Server-Zahl (3 oder 5) fuer etcd-Quorum.
Feste Registration-Address (LB/VIP) vor allen Servern – muss in **tls-san** stehen, sonst x509-Fehler ueber den LB.
Der erste Server bootstrappt (kein **server:**-Feld), weitere joinen ueber **server:** + gleichen **token**.

Ports

9345: RKE2 Supervisor/Registration (nicht 6443!) – darauf zeigt **server:**.
6443: kube-apiserver.
2379/2380: etcd client/peer.

```
# config.yaml (zweiter/dritter Server)
server: https://rke2.example.com:9345
token: <shared-secret>
tls-san:
  - rke2.example.com
```

Bundled Components

Auto-Deploy aus manifests/

Alles in **/var/lib/rancher/rke2/server/manifests/** wird vom Deploy-Controller angewandt (HelmChart-CRs: rke2-canal, rke2-coredns, rke2-ingress-nginx, rke2-metrics-server).
Nicht direkt editieren – beim Restart ueberschrieben.

Anpassen & Abschalten

Customizing ueber HelmChartConfig-CR (gleicher Name/Namespace, **valuesContent** wird gemergt).
Abschalten mitgelieferter Komponenten ueber **disable:** in der config.yaml (z.B. **rke2-ingress-nginx**, **rke2-metrics-server**).

```
apiVersion: helm.cattle.io/v1
kind: HelmChartConfig
metadata:
  name: rke2-ingress-nginx
  namespace: kube-system
spec:
  valuesContent: |-
    controller:
      config:
        use-forwarded-headers: "true"
      allowSnippetAnnotations: "false"
```

Security & Hardening

CIS-Profil

profile: cis in der config.yaml aktiviert die CIS-Benchmark-Haertung: PSA **restricted** als Default (ausser System-Namespaces), NetworkPolicies fuer System-NS, strenge Datei-Rechte.
Host-Prereqs: **etcd**-User + Kernel-Sysctls (via **rke2-cis-sysctl**-Paket bzw. **/etc/sysctl.d**).

Weitere Defaults

Secrets-Encryption at rest: standardmaessig an (AES-CBC), verwaltet via **rke2 secrets-encrypt**.
SELinux: unterstuetzt (**selinux: true** + rke2-selinux Policy).
Pod Security Admission: Cluster-weite Default-Config im cis-Profil.
Kein Docker: reduzierte Angriffsflaeche, nur containerd.

```
rke2 secrets-encrypt status
rke2 secrets-encrypt rotate-keys   # Key-Rotation
kubectl get --raw='/readyz?verbose'
```

etcd Snapshot / Restore

Snapshots

Automatisch an by default: **etcd-snapshot-schedule-cron** (Default alle 12h), **etcd-snapshot-retention** (Default 5).
Ablage: **/var/lib/rancher/rke2/server/db/snapshots**.
Off-node sichern: **etcd-s3**-Optionen (S3-Bucket) – sonst ist der Snapshot mit dem Node verloren.

```
rke2 etcd-snapshot save --name pre-upgrade
rke2 etcd-snapshot ls
# Restore (auf EINEM Server, andere vorher stoppen):
systemctl stop rke2-server
rke2 server \
  --cluster-reset \
  --cluster-reset-restore-path=/var/lib/rancher/rke2/\
server/db/snapshots/pre-upgrade-<ts>
systemctl start rke2-server
```

Restore-Ablauf

Restore laeuft auf einem Server, alle anderen Server werden gestoppt und ihr **db/**-Verzeichnis geleert; danach re-joinen sie den zurueckgesetzten Cluster ueber **server:** + **token**.

Upgrades

Zwei Wege

system-upgrade-controller (SUSE-Pattern): deklarative **Plan**-CRs, node-selektiert, Server zuerst, dann Agents.
Manuell: **INSTALL_RKE2_VERSION** setzen, Installer erneut laufen lassen, **rke2-server** neustarten.
Channels: **INSTALL_RKE2_CHANNEL** (**stable**|**latest**|**v1.32**).

```
apiVersion: upgrade.cattle.io/v1
kind: Plan
metadata: {name: server-plan, namespace: system-upgrade}
spec:
  concurrency: 1
  nodeSelector:
    matchExpressions:
    - {key: node-role.kubernetes.io/control-plane, operator:
       Exists}
  serviceAccountName: system-upgrade
  cordon: true
  upgrade: {image: rancher/rke2-upgrade}
  version: v1.32.5+rke2r1
```

Air-Gap

Images & Binary

Images-Tarball (`rke2-images.linux-amd64.tar.zst`) nach `/var/lib/rancher/rke2/agent/images/` legen – RKE2 importiert ihn beim Start in containerd.
Binary + `install.sh` offline via `INSTALL_RKE2_ARTIFACT_PATH`.

Private Registry

`/etc/rancher/rke2/registries.yaml`: Mirror-Endpunkte + Auth + TLS. Gilt fuer System-Images und Workloads.

```
# /etc/rancher/rke2/registries.yaml
mirrors:
  docker.io:
    endpoint:
      - "https://registry.internal:5000"
configs:
  registry.internal:5000:
    auth: {username: robot, password: <token>}
    tls: {ca_file: /etc/ssl/registry-ca.pem}
```

Diagnose

Werkzeuge

`journalctl -u rke2-server -f` (bzw. `rke2-agent`).
`crictl` unter `/var/lib/rancher/rke2/bin` – Container/Images direkt am Node (containerd, nicht Docker).
Static-Pod-Logs: `crictl logs` oder `/var/log/pods`.

```
export
    CRI_CONFIG_FILE=/var/lib/rancher/rke2/agent/etc/crictl.yaml
/var/lib/rancher/rke2/bin/crictl ps
/var/lib/rancher/rke2/bin/crictl logs <container-id>
journalctl -u rke2-server --no-pager | tail -50
```

Reset & Removal

`rke2-killall.sh`: stoppt alle rke2-Prozesse + Container, laesst Daten liegen.
`rke2-uninstall.sh`: vollstaendige Entfernung (Binaries, Daten, Unit).

Anti-Patterns

Was du nicht tun solltest

Gerade Server-Zahl (2/4): kein Quorum-Gewinn, Split-Brain-Risiko – immer 3 oder 5.
Snapshots nur on-node: Node-Verlust = etcd-Verlust. S3/off-node konfigurieren.
Docker erwarten: RKE2 hat ein eigenes containerd; kein Docker-Socket, kein `docker ps`. Nutze `crictl`.
tls-san vergessen: API-Zertifikat kennt den LB/VIP-Namen nicht – x509-Fehler, sobald kubectl ueber den LB geht.
Server als Worker ohne Taint in Prod: Workloads konkurrieren mit etcd/Control-Plane – `CriticalAddonsOnly`-Taint setzen.
manifests/ direkt editieren: beim Restart ueberschrieben – `HelmChartConfig` statt Datei-Patch.

rke2-cheatsheet.de

RKE2 Cheatsheet von OMNI52

Weiterfuehren, nicht selbst neu erfinden

RKE2-Rollout & Cluster-Hardening

OMNI52™ hilft Plattform-Teams, RKE2 fuer regulierte Enterprise-Umgebungen sauber aufzustellen.

HA-Design mit ungerader Server-Zahl und externem LB, CIS-Profil-Rollout ohne Produktions-Riss, etcd-Snapshot-Strategie mit Off-Node-Backup, Air-Gap-Installation und Upgrade-Automatisierung via system-upgrade-controller. Output landet als GitOps-Repo bei euch im Cluster: Helm-Charts, Runbooks, Diagnose-Skripte. Euer Team operiert weiter, nicht wir.

OMNI52 GmbH, Ebern · istio-consulting.de · kostenloses 30-min Initial-Assessment

Aus der OMNI52 Cheatsheet-Fabrik

Verwandte Sheets: rancher-cheatsheet.de (Rancher-Management) · kubernetes-cheatsheet.de (Kubernetes Core) · istio-cheatsheet.de (Service-Mesh-Layer).

Lizenz & Weiterverteilung

Trademarks