RKE2 Install Guid

Mehr Informationen bekommt man von der offiziellen Dokumentationsseite von RKE2

Server Node Installation

Überprüfe ob curl und iptables auf dem System installiert ist

Konfiguration für den Cluster erstellen

```
mkdir -p /etc/rancher/rke2/
vim /etc/rancher/rke2/config.yaml
```

Und mit folgendem Inhalt befüllen

```
write-kubeconfig-mode: 0600
tls-san:
  - 10.6.8.20           # tls-san nur wenn HA
disable:
  - rke2-ingress-nginx
cni: calico
cluster-cidr: "100.73.0.0/16"
service-cidr: "100.74.0.0/16"
```

Damit geben wir gleich die service und cluster cidr an. Also welche IP Range sollen die Services bekommen und welche die Container.

1. Installer ausführen

   ```
   curl -sfL https://get.rke2.io | sh -
   ```

   Wenn eine bestimmte Version gewünscht ist

   ```
   curl -sfL https://get.rke2.io | INSTALL_RKE2_VERSION=v1.30.5+rke2r1 sh
   -
   ```

2. Aktivieren des rke2-server service

   ```
   systemctl enable rke2-server.service
   ```

3. Starten des Services

   ```
   systemctl start rke2-server.service
   ```

   oder

   ```
   systemctl start --no-block rke2-server.service
   ```

4. Prüfe das Log nach belieben

```
journalctl -u rke2-server -f
```

After running this installation:

1. The rke2-server service will be installed. The rke2-server service will be configured to automatically restart after node reboots or if the process crashes or is killed.
2. Additional utilities will be installed at /var/lib/rancher/rke2/bin/. They include: kubectl, crictl, and ctr. Note that these are not on your path by default.
3. Two cleanup scripts, rke2-killall.sh and rke2-uninstall.sh, will be installed to the path at:
    1. /usr/local/bin for regular file systems
    2. /opt/rke2/bin for read-only and brtfs file systems
    3. INSTALL_RKE2_TAR_PREFIX/bin if INSTALL_RKE2_TAR_PREFIX is set
4. A kubeconfig file will be written to /etc/rancher/rke2/rke2.yaml.
5. A token that can be used to register other server or agent nodes will be created at /var/lib/rancher/rke2/server/node-token

RKE2 HA

Wenn der rke2 Cluster HA fähig sein soll dann empfiehlt es sich dies mit kube-vip als Loadbalancer zu machen. Eine Beispielkonfiguration für eine kube-vip.yml findest Du hier im Wiki im Bereich Kubernetes.

Eine Mögliche HA Konfig mittels kube-vip Daemonset

> RBAC.yml

```
kubectl apply -f https://kube-vip.io/manifests/rbac.yaml
```

> daemonset.yml

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  annotations:
  name: kube-vip-ds
  namespace: kube-system
spec:
  selector:
    matchLabels:
      name: kube-vip-ds
  template:
    metadata:
      creationTimestamp: null
      labels:
        name: kube-vip-ds
```

```
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: node-role.kubernetes.io/master
                operator: Exists
            - matchExpressions:
              - key: node-role.kubernetes.io/control-plane
                operator: Exists
      containers:
      - args:
        - manager
        env:
        - name: address
          value: need-to-be-set           # use your own kubevip
        - name: vip_arp
          value: "true"
        - name: port
          value: "6443"
        - name: vip_interface             # use your own network interface
          value: ens3                     #     e.g. ens3, eth0
        - name: vip_cidr
          value: "32"
        - name: cp_enable
          value: "true"
        - name: cp_namespace
          value: kube-system
        - name: vip_ddns
          value: "false"
        - name: svc_enable
          value: "true"
        - name: vip_leaderelection
          value: "true"
        - name: vip_leaseduration
          value: "5"
        - name: vip_renewdeadline
          value: "3"
        - name: vip_retryperiod
          value: "1"
        image: ghcr.io/kube-vip/kube-vip:v1.0.1
        imagePullPolicy: Always
        name: kube-vip
        resources: {}
        securityContext:
          capabilities:
            add:
            - NET_ADMIN
            - NET_RAW
            - SYS_TIME
```

```
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      hostNetwork: true
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: kube-vip
      serviceAccountName: kube-vip
      terminationGracePeriodSeconds: 30
      tolerations:
      - effect: NoSchedule
        operator: Exists
      - effect: NoExecute
        operator: Exists
  updateStrategy:
    rollingUpdate:
      maxSurge: 0
      maxUnavailable: 1
    type: RollingUpdate
```

Verzeichnisse für Manifeste und Static/MirrorPods Bei RKE2 gibt es zwei besondere Verzeichnisse für Manifeste und Static- oder Mirror- Pods: /var/lib/rancher/rke2/server/manifests/ Alle Kubernetes-Manifeste, die in diesem Verzeichnis abgespeichert sind, werden automatisch in den Cluster übertragen („applied“), schon beim Start des RKE2 oder auch zur Laufzeit. Wird ein bestehendes Manifest aus dem Verzeichnis ge- löscht, wird das Manifest aus dem Cluster ebenso entfernt! /var/lib/rancher/rke2/agent/static-pods/ Verzeichnis für die Static- oder MirrorPods unter Kubernetes für RKE2. Liegen dort Pod-Manifest-Dateien, werden diese ebenso in den Cluster übertragen. Die Besonderheit ist dabei, dass das kubelet für das Anstarten des Pods den API- Server nicht benötigt. (Das gilt nur für Pod-Manifeste; nicht aber für Services usw.)

Einrichten der zweiten Control Plane Node

Erstellen einer passenden Cluster-Konfigurationsdatei (/etc/rancher/rke2/config.yaml):

```
write-kubeconfig-mode: 0600
tls-san:
  - 10.6.8.20
disable:
  - rke2-ingress-nginx
cni: calico
cluster-cidr: "100.73.0.0/16"
service-cidr: "100.74.0.0/16"
server: https://10.6.8.20:9345
token: <token from server node>
```

Achte darauf das die IP Adresse 10.6.8.20 bei der HA Konfiguration NICHT die IP einer Node sein darf sondern die IP welche Du bei der kube-vip Konfig als ENV `address` konfiguriert hast.

Linux Agent (Worker) Node installieren

Überprüfe ob curl und iptables auf dem System installiert ist

Konfiguration für den Cluster erstellen

```
mkdir -p /etc/rancher/rke2/
vim /etc/rancher/rke2/config.yaml
```

Und mit folgendem Inhalt befüllen

```
server: https://10.6.8.20:9345
token: <token from server node>
```

Server: Der rke2-Serverprozess wartet auf Port 9345 auf die Registrierung neuer Knoten. Die Kubernetes-API wird weiterhin wie gewohnt auf Port 6443 bereitgestellt.

Token: Unter /var/lib/rancher/rke2/server/node-token wird ein Token erstellt, der zum Registrieren anderer Server- oder Agentenknoten verwendet werden kann

cluster-cidr: und **service-cidr:** Damit geben wir gleich die service und cluster cidr an. Also welche IP Range sollen die Services bekommen und welche die Container.

1. Installer ausführen

   ```
   curl -sfL https://get.rke2.io | INSTALL_RKE2_TYPE="agent" sh -
   ```

   Wenn eine bestimmte Version gewünscht ist

   ```
   curl -sfL https://get.rke2.io | INSTALL_RKE2_VERSION=v1.30.5+rke2r1
   INSTALL_RKE2_TYPE="agent" sh -
   ```

2. Aktivieren des rke2-server service

   ```
   systemctl enable rke2-agent.service
   ```

3. Starten des Services

   ```
   systemctl start rke2-agent.service
   ```

4. Prüfe das Log nach belieben

   ```
   journalctl -u rke2-agent -f
   ```

Konfigurationsänderungen

Beenden der RKE2-Prozesse

```
rke2-killall.sh
```

rke2 deinstallieren

```
rke2-uninstall.sh
```

Administration mit kubectl

```
export KUBECONFIG=/etc/rancher/rke2/rke2.yaml
export PATH=/var/lib/rancher/rke2/bin/kubectl:$PATH
```

Kontrolle mit crictl

```
export PATH=/var/lib/rancher/rke2/bin:$PATH
export CRI_CONFIG_FILE=/var/lib/rancher/rke2/agent/etc/crictl.yaml
```