# rke2安装调研

## rke2 server安装

```
curl -sfL http://rancher-mirror.rancher.cn/rke2/install.sh |
INSTALL_RKE2_MIRROR=cn sh -

systemctl enable rke2-server.service
systemctl start rke2-server.service    ## 可能时间比较长，需要拉取所需的镜像
```

```
[root@node1 ~]# ls -l /var/lib/rancher/rke2/bin/    ## rke2提供的命令
-rwxr-xr-x 1 root root  54120392 5月  18 16:49 containerd
-rwxr-xr-x 1 root root   7369488 5月  18 16:49 containerd-shim
-rwxr-xr-x 1 root root  11527464 5月  18 16:49 containerd-shim-runc-v1
-rwxr-xr-x 1 root root  11539944 5月  18 16:49 containerd-shim-runc-v2
-rwxr-xr-x 1 root root  34985144 5月  18 16:49 crictl
-rwxr-xr-x 1 root root  20463560 5月  18 16:49 ctr
-rwxr-xr-x 1 root root  48746480 5月  18 16:49 kubectl
-rwxr-xr-x 1 root root 119735752 5月  18 16:49 kubelet
-rwxr-xr-x 1 root root  11068888 5月  18 16:50 runc

ln -s /var/lib/rancher/rke2/bin/kubectl  /usr/bin/kubectl
ln -s /var/lib/rancher/rke2/bin/ctr /usr/bin/ctr
ln -s /var/lib/rancher/rke2/bin/crictl /usr/bin/crictl
```

```
[root@node1 ~]# /var/lib/rancher/rke2/bin/crictl --runtime-endpoint unix:///run/k3s/containerd/containerd.sock image ls
IMAGE                                                TAG                          IMAGE ID            SIZE
docker.io/rancher/hardened-calico                    v3.21.4-build20220228        3d20d59cd627c       197MB
docker.io/rancher/hardened-cluster-autoscaler        v1.8.5-build20211119         0e6424d22fd5c       43.6MB
docker.io/rancher/hardened-coredns                   v1.9.1-build20220318         a9c20758166a0       48MB
docker.io/rancher/hardened-etcd                      v3.5.3-k3s1-build20220413    65bbabcdc0fce       49.1MB
docker.io/rancher/hardened-flannel                   v0.17.0-build20220317        f776f3ce534ab       95.5MB
docker.io/rancher/hardened-k8s-metrics-server        v0.5.0-build20211119         57533a88f34ca       49.7MB
docker.io/rancher/hardened-kubernetes                v1.22.9-rke2r2-build20220428 a4eae6fd14a2d       221MB
docker.io/rancher/klipper-helm                       v0.7.1-build20220407         4adfa32cd74b1       83MB
docker.io/rancher/mirrored-ingress-nginx-kube-webhook-certgen  v1.1.1             c41e9fcadf5a2       18.9MB
docker.io/rancher/nginx-ingress-controller           nginx-1.2.0-hardened6        9f7db473ac573       231MB
docker.io/rancher/pause                              3.6                          6270bb605e12e       299kB
docker.io/rancher/rke2-cloud-provider                v0.0.3-build20211118         029e4a095f53d       49.5MB
```

rke2 server端所需要的镜像

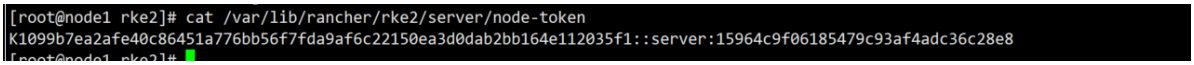
```
export KUBECONFIG=/etc/rancher/rke2/rke2.yaml       ## 配置KUBECONFIG

或
mkdir -p ~/.kube/
cp /etc/rancher/rke2/rke2.yaml ~/.kube/config
```

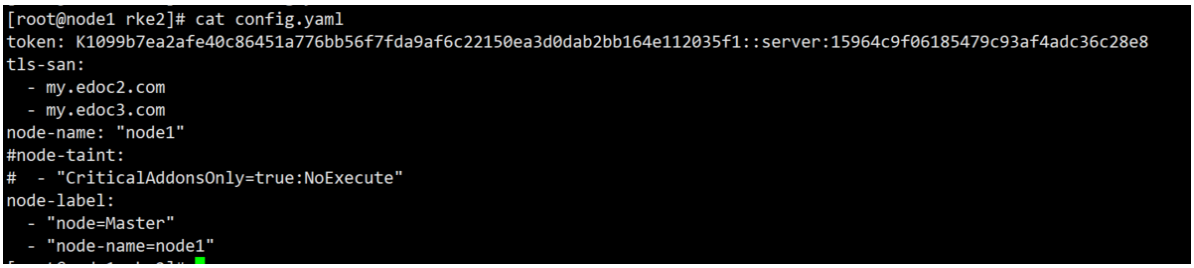
```
[root@node1 ~]# /var/lib/rancher/rke2/bin/kubectl get pod -A
NAMESPACE     NAME                                                READY   STATUS      RESTARTS      AGE
kube-system   cloud-controller-manager-node1                      1/1     Running     3 (17m ago)   17m
kube-system   etcd-node1                                          1/1     Running     0             17m
kube-system   helm-install-rke2-canal--1-xcsrx                    0/1     Completed   0             17m
kube-system   helm-install-rke2-coredns--1-w95g4                  0/1     Completed   0             17m
kube-system   helm-install-rke2-ingress-nginx--1-p7fll            0/1     Completed   0             17m
kube-system   helm-install-rke2-metrics-server--1-rqpkv           0/1     Completed   0             17m
kube-system   kube-apiserver-node1                                1/1     Running     0             17m
kube-system   kube-controller-manager-node1                       1/1     Running     2 (17m ago)   17m
kube-system   kube-proxy-node1                                    1/1     Running     0             17m
kube-system   kube-scheduler-node1                                1/1     Running     0             17m
kube-system   rke2-canal-7hx2m                                    2/2     Running     0             16m
kube-system   rke2-coredns-rke2-coredns-687554ff58-9ks27          1/1     Running     0             16m
kube-system   rke2-coredns-rke2-coredns-autoscaler-7566b44b85-nss77  1/1     Running     0             16m
kube-system   rke2-ingress-nginx-controller-qtcvt                 1/1     Running     0             9m20s
kube-system   rke2-metrics-server-8574659c85-zbv7d                1/1     Running     0             10m
```

server端启动之后启动的pod

**配置rke2 server config.yaml**

获取节点token




```
# cat /etc/rancher/rke2/config.yaml   ## 创建此文件
server: https://my.edoc2.com:9345
token:
K1099b7ea2afe40c86451a776bb56f7fda9af6c22150ea3d0dab2bb164e112035f1::server:1596
4c9f06185479c93af4adc36c28e8
tls-san:
  - my.edoc2.com
  - my.edoc3.com # 都是集群的别名，是tls证书所认证的别名或域名，需要认证的别名罗列在这里就可
以被tls认证
node-name: "node1"
#node-taint:
#  - "CriticalAddonsOnly=true:NoExecute"
node-label:
  - "node=Master"
  - "node-name=node1"
```

重启 rke2-server 服务

```
# systemctl restart rke2-server.service
# /var/lib/rancher/rke2/bin/kubectl get node
NAME    STATUS   ROLES                       AGE   VERSION
node1   Ready    control-plane,etcd,master   49m   v1.22.9+rke2r2
```

此时一个节点的rke2 安装完成

# 添加rke2 server节点

配置hosts解析

```
192.168.251.244 node1 my.edoc2.com my.edoc3.com
192.168.251.98 node2
192.168.251.249 node3
192.168.20.223 node4
```

第二个节点同样需要配置 config.yaml 文件（**这个文件需要在第二节点部署之前就创建完成**）

```
[root@node2 rke2]# cat config.yaml
server: https://192.168.251.244:9345
token: K1099b7ea2afe40c86451a776bb56f7fda9af6c22150ea3d0dab2bb164e112035f1::server:15964c9f06185479c93af4adc36c28e8
tls-san:
  - my.edoc2.com
  - my.edoc3.com
node-name: "node2"
#node-taint:
#  - "CriticalAddonsOnly=true:NoExecute"
node-label:
  - "node=Master"
  - "node-name=node2"
```

```
mkdir -p /etc/rancher/rke2
# cat /etc/rancher/rke2/config.yaml
server: https://my.edoc2.com:9345     ## 指定第一个server地址
token:
K1099b7ea2afe40c86451a776bb56f7fda9af6c22150ea3d0dab2bb164e112035f1::server:1596
4c9f06185479c93af4adc36c28e8    ## 第一个server的token
tls-san:
  - my.edoc2.com
  - my.edoc3.com
node-name: "node2"
#node-taint:
#  - "CriticalAddonsOnly=true:NoExecute"
node-label:
  - "node=Master"
  - "node-name=node2"
```

```
## 进行第二个节点的安装
curl -sfL http://rancher-mirror.rancher.cn/rke2/install.sh |
INSTALL_RKE2_MIRROR=cn sh -

systemctl enable rke2-server.service
systemctl start rke2-server.service    ## 可能时间比较长，需要拉取所需的镜像
```

```
[root@node1 ~]# kubectl get node
NAME    STATUS   ROLES                       AGE   VERSION
node1   Ready    control-plane,etcd,master   37m   v1.24.0+rke2r1
node2   Ready    control-plane,etcd,master   39s   v1.24.0+rke2r1
```

**同样的方法添加第三个节点**

**重启第二个节点，如果启动不了，出现以下报错**

```
May 18 07:24:19 logserver rke2[22407]: time="2022-05-18T07:24:19-04:00" level=fatal msg="/var/lib/rancher/rke2/server/tls/server-ca.key, /var/lib/rancher/rke2/server/cred/encryption-state.j
son, /var/lib/rancher/rke2/server/tls/etcd/server-ca.key, /var/lib/rancher/rke2/server/tls/etcd/server-ca.crt, /var/lib/rancher/rke2/server/cred/ipsec.psk, /var/lib/rancher/rke2/server/cred
/passwd, /var/lib/rancher/rke2/server/tls/etcd/peer-ca.crt, /var/lib/rancher/rke2/server/tls/etcd/peer-ca.key, /var/lib/rancher/rke2/server/tls/request-header-ca.key, /var/lib/rancher/rke2/
server/tls/request-header-ca.crt, /var/lib/rancher/rke2/server/tls/service.key, /var/lib/rancher/rke2/server/tls/client-ca.crt, /var/lib/rancher/rke2/server/tls/server-ca.crt, /var/lib/ranc
her/rke2/server/tls/client-ca.key, /var/lib/rancher/rke2/server/cred/encryption-config.json newer than datastore and could cause a cluster outage. Remove the file(s) from disk and restart t
o be recreated from datastore."
```

这个是因为，进行 start rke2-server的时候，没有提前准备好 /etc/rancher/rke2/config.yaml（去配置第一个server节点的信息），启动的时候作为了初始化节点；
先进行了start rke2-server,后续添加config.yaml文件,在进行重启rke2-server，就会导致出现以上报错

# 添加rke2 agent 节点

```
curl -sfL http://rancher-mirror.rancher.cn/rke2/install.sh |
INSTALL_RKE2_MIRROR=cn INSTALL_RKE2_TYPE="agent"  sh -
```

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

配置hosts解析

```
192.168.251.244 node1 my.edoc2.com my.edoc3.com
192.168.251.98 node2
192.168.251.249 node3
192.168.20.223 node4
```

配置 config.yml 文件

```
[root@node4 ~]# cat /etc/rancher/rke2/config.yaml
server: https://my.edoc2.com:9345
token:
K1099b7ea2afe40c86451a776bb56f7fda9af6c22150ea3d0dab2bb164e112035f1::server:1596
4c9f06185479c93af4adc36c28e8
node-name: "node4"
node-label:
  - "node=worker"
  - "node-name=node4"
```

启动 rke2-agent

```
systemctl start rke2-agent.service      ## 注意启动的agent服务
```

## 部署应用测试

```
[root@node1 snapshots]# kubectl create deployment test --image=busybox:1.28  --replicas=4   -- sleep 30000
deployment.apps/test created
```

```
[root@node1 snapshots]# kubectl get pod -o wide
NAME                   READY   STATUS    RESTARTS   AGE   IP          NODE    NOMINATED NODE   READINESS GATES
test-569cff9d8d-g2hl7   1/1     Running   0          33s   10.42.2.3   node3   <none>           <none>
test-569cff9d8d-m72k5   1/1     Running   0          33s   10.42.3.2   node4   <none>           <none>
test-569cff9d8d-tlp8t   1/1     Running   0          33s   10.42.1.2   node2   <none>           <none>
test-569cff9d8d-xnnzr   1/1     Running   0          33s   10.42.0.5   node1   <none>           <none>
```

## 外部访问集群高可用配置

Apiserver统一入口（可选），为了方便外部访问集群，需要在集群实现统一入口，可以通过L4负载均衡器或vip地址或智能轮询DNS。集群内部已经通过rke2-agent实现了worker访问api-server的多入口反向代理

nginx配置示例：

```
events {
  worker_connections  1024;  ## Default: 1024
}
stream {
    upstream kube-apiserver {
        server node1:6443     max_fails=3 fail_timeout=30s;
        server node2:6443     max_fails=3 fail_timeout=30s;
        server node3:6443     max_fails=3 fail_timeout=30s;
    }
    upstream rke2 {
        server node1:9345     max_fails=3 fail_timeout=30s;
        server node2:9345     max_fails=3 fail_timeout=30s;
        server node3:9345     max_fails=3 fail_timeout=30s;
```

```
    }
    server {
        listen 6443;
        proxy_connect_timeout 2s;
        proxy_timeout 900s;
        proxy_pass kube-apiserver;
    }
    server {
        listen 9345;
        proxy_connect_timeout 2s;
        proxy_timeout 900s;
        proxy_pass rke2;
    }
}
```

server端的 tls-san：

```
tls-san:
  - xxx.xxx.xxx.xxx    ## IP
  - www.xxx.com        ## 域名
## 此处填写LB的统一入口ip地址或域名
```

kubeconfig配置：


## RKE2内部高可用实现

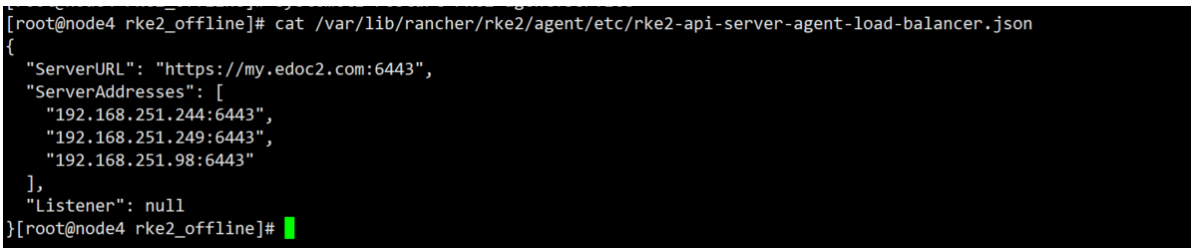
k8s集群的高可用针对：

- etcd：通过本身的 Raft 算法 Leader 选主机制，组成ETCD集群，实现 etcd 高可用。
- controller manager：leader election 选举竞争锁的机制来保证高可用
- scheduler：leader election 选举竞争锁的机制来保证高可用。
- apiserver：无状态，通过前端负载均衡实现高可用。

rke2集群中，containerd、kubelet组件集成到了rke2服务中，这点和k3s非常相式，同时在rke2服务中还集成了nginx服务，主要用于做为kubelet连接api-server的方向代理，RKE2 会帮助其他组件自动做HA。

当api-server有统一入口，所有请求走LB连接到api-server；

如果api-server没有统一入口，kubelet和rke2-agent去连接rke2-server时，会用一个server地址去注册即可，然后agent会获取 所有rke2 server 的地址，然后存储到 /var/lib/rancher/rke2/agent/etc/rke2-api-server-agent-load-balancer.json中，生成nginx反向代理配置




**rke2-agent Down:**

不会影响业务正常运行，因为containerd创建容器是通过containerd-shim-runc-v2调用runc创建，当containerd出现问题时containerd-shim-runc-v2会被init进程托管，不会导致退出影响现有业务POD。但需要注意的是**rke2-agent退出后kubelet也退出**了，对应的业务状态探测就没有了，在默认超时5分钟后，Controller-manager会将业务pod重建。

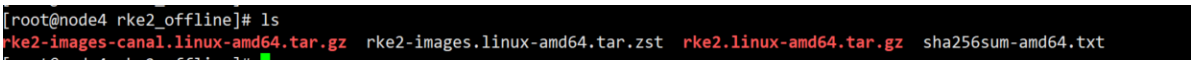
## 使用离线包安装

下载离线包

下载地址：https://github.com/rancher/rke2/releases

- rke2-images.linux-amd64.tar
- rke2.linux-amd64.tar.gz
- rke2-images-canal.linux-amd64.tar.gz（根据使用的网络插件）
- sha256sum-amd64.txt （hash文件）

```
# mkdir /root/rke2_offline && cd /root/rke2_offline
curl -OLs
https://github.com/rancher/rke2/releases/download/v1.21.5%2Brke2r2/rke2-
images.linux-amd64.tar.zst
curl -OLs
https://github.com/rancher/rke2/releases/download/v1.21.5%2Brke2r2/rke2.linux-
amd64.tar.gz
curl -OLs
https://github.com/rancher/rke2/releases/download/v1.21.5%2Brke2r2/sha256sum-
amd64.txt
```




下载安装脚本

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

进行安装

```
INSTALL_RKE2_ARTIFACT_PATH=/root/rke2_offline sh install.sh    # server端安装
INSTALL_RKE2_ARTIFACT_PATH=/root/rke2_offline INSTALL_RKE2_TYPE="agent" sh
install.sh    # agent端安装
## 会把安装包放到相应的位置，配置好rke2 服务，并没有进行实际的安装
```

启动服务，进行安装

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

## 配置自有仓库

配置registry.yaml 配置文件

```
mirrors:
  edoc2.com:
    endpoint:
      - "https://registry.edoc2.com:5000"
configs:
```

```
  "edoc2.com":
    auth:
      username: "ci"
      password: "1qaz@WSX"
#    tls:
#      cert_file:
#      key_file:
#      ca_file:
#      insecure_skip_verify: true
```

# 升级

```
## 再次执行安装脚本，会升级到最新的稳定版本
curl -sfL http://rancher-mirror.rancher.cn/rke2/install.sh |
INSTALL_RKE2_MIRROR=cn sh -

## 升级到指定版本
curl -sfL https://get.rke2.io | INSTALL_RKE2_VERSION=vX.Y.Z-rc1 sh -
```

# 备份与恢复

**配置备份计划时间**

rke2会自动的进行快照的备份，默认每12小时生成一次快照。保存路径在：/var/lib/rancher/rke2/server/db/snapshots 下

更改rke2 的配置文件：（**所有master节点配置一致**）

```
[root@node1 ~]# cat /etc/rancher/rke2/config.yaml
server: https://my.edoc2.com:9345
token:
K10d8ea6e640267b8a8019b43a8f4f19c39bf4a77d9880a4d4abdd058aa2db0a657::server:30b6
5290a2ab54ddd579282c40f82b9b
tls-san:
  - my.edoc2.com
  - my.edoc3.com
node-name: "node1"
#node-taint:
#  - "CriticalAddonsOnly=true:NoExecute"
node-label:
  - "node=Master"
  - "node-name=node1"
etcd-snapshot-retention: 2                    ## 快照的保存个数
etcd-snapshot-schedule-cron: "*/2 * * * *"  ## 每两分钟备份一次
```

```
[root@node1 snapshots]# pwd
/var/lib/rancher/rke2/server/db/snapshots
[root@node1 snapshots]# ls
etcd-snapshot-node1-1652958840
```

```
[root@node1 snapshots]# ls
etcd-snapshot-node1-1652958960  etcd-snapshot-node1-1652959080
[root@node1 snapshots]# du -sh *
4.0M    etcd-snapshot-node1-1652958960
4.0M    etcd-snapshot-node1-1652959080
```

**进行恢复**

在进行集群恢复的时候mater节点需要停止所有的服务，worker节点只需要停掉rke2-agent服务

**第一个master节点操作：**

```
[root@node1 snapshots]# rke2-killall.sh
+ systemctl stop rke2-server.service
+ systemctl stop rke2-agent.service
+ killtree 5012 5015 5174 5194 5746 6718 6848 7304 7850 10085 10125
+ kill -9 5012 5052 10624 5015 5059 10635 5174 5218 10717 5194 5225 5264 5746 5766 6239 6295 6300 6296
```

使用rke2-killall.sh 停止所有服务

```
[root@node1 snapshots]# ls /var/lib/rancher/rke2/server/db/snapshots/etcd-snapshot-node1-1652960280
/var/lib/rancher/rke2/server/db/snapshots/etcd-snapshot-node1-1652960280
[root@node1 snapshots]# rke2 server --cluster-reset --cluster-reset-restore-path=/var/lib/rancher/rke2/server/db/snapshots/etcd-snapshot-node1-1652960280
```

```
rke2 server --cluster-reset --cluster-reset-restore-
path=/var/lib/rancher/rke2/server/db/snapshots/etcd-snapshot-node1-1652960280
```

```
INFO[0148] Defragmenting etcd database
INFO[0148] etcd data store connection OK
INFO[0148] ETCD server is now running
INFO[0148] rke2 is up and running
INFO[0148] Waiting for API server to become available
INFO[0148] Tunnel server waiting for runtime core to become available
WARN[0148] bootstrap key already exists
INFO[0148] Reconciling etcd snapshot data in rke2-etcd-snapshots ConfigMap
INFO[0149] Defragmenting etcd database
INFO[0149] Waiting to retrieve kube-proxy configuration; server is not ready: https://127.0.0.1:6444/v1-rke2/readyz: 500 Internal Server Error
INFO[0149] Reconciling bootstrap data between datastore and disk
INFO[0149] Cluster reset: backing up certificates directory to /var/lib/rancher/rke2/server/tls-1652960833
INFO[0153] Tunnel server waiting for runtime core to become available
INFO[0154] Waiting to retrieve kube-proxy configuration; server is not ready: https://127.0.0.1:6444/v1-rke2/readyz: 500 Internal Server Error
INFO[0154] Managed etcd cluster membership has been reset, restart without --cluster-reset flag now. Backup and delete ${datadir}/server/db on each peer etcd server and rejoin the nodes
```

恢复完成后，启动第一个节点

```
[root@node1 snapshots]# systemctl start rke2-server
```

第一个节点启动后会有以下状态

```
[root@node1 snapshots]# kubectl get node
NAME    STATUS     ROLES                       AGE     VERSION
node1   Ready      control-plane,etcd,master   3h34m   v1.24.0+rke2r1
node2   NotReady   control-plane,etcd,master   177m    v1.24.0+rke2r1
node3   NotReady   control-plane,etcd,master   169m    v1.24.0+rke2r1
node4   Ready      control-plane,etcd,master   143m    v1.24.0+rke2r1
```

**第二个master节点操作：**

第二个节点**备份db目录后，删除db数据目录**，重新从第一个节点恢复的数据同步最新数据

```
[root@node2 server]# pwd
/var/lib/rancher/rke2/server
[root@node2 server]# rm -rf db/          ## master 删除db目录
[root@node2 server]# systemctl start rke2-server   ## 重新启动server
```

第三个节点的操作和第二个节点操作相同

agent节点再master节点恢复后，重新启动 rke2-agent服务即可。

```
[root@node1 snapshots]# kubectl get node
NAME    STATUS   ROLES                       AGE   VERSION
node1   Ready    control-plane,etcd,master   16h   v1.24.0+rke2r1
node2   Ready    control-plane,etcd,master   16h   v1.24.0+rke2r1
node3   Ready    control-plane,etcd,master   16h   v1.24.0+rke2r1
node4   Ready    <none>                      97s   v1.24.0+rke2r1
```

## 配置k8s组件参数

在/etc/rancher/rke2/config.yaml 文件中，按照对应组件，添加对应的参数，如apiserver对应为kube-apiserver-arg，组件对应参数为etcd-arg。kube-controller-manager-arg、kube-scheduler-arg、kubelet-arg、kube-proxy-arg

```
[root@node1 snapshots]# cat /etc/rancher/rke2/config.yaml
server: https://my.edoc2.com:9345
token:
K10d8ea6e640267b8a8019b43a8f4f19c39bf4a77d9880a4d4abdd058aa2db0a657::server:30b6
5290a2ab54ddd579282c40f82b9b
tls-san:
  - my.edoc2.com
  - my.edoc3.com
node-name: "node1"
#node-taint:
#  - "CriticalAddonsOnly=true:NoExecute"
node-label:
  - "node=Master"
  - "node-name=node1"
etcd-snapshot-retention: 2
etcd-snapshot-schedule-cron: "*/2 * * * *"
kubelet-arg:
  - "eviction-hard=nodefs.available<1%,memory.available<10%"
  - "eviction-soft=nodefs.available<5%,imagefs.available<1%"
  - "eviction-soft-grace-period=nodefs.available=30s,imagefs.available=30s"
```

配置kubelet 进行pod驱逐的限制

重启rke2 server，查看kubelet进行参数

```
[root@node1 snapshots]# ps -ef | grep kubelet | grep available
root     29040 28701  3 19:29 ?        00:00:03 kubelet --volume-plugin-dir=/var/lib/kubelet/volumeplugins --file-check-frequency=5s --sync-frequency=30s --address=0.0.0.0 --alsologtostderr
=false --anonymous-auth=false --authentication-token-webhook=true --authorization-mode=Webhook --cgroup-driver=systemd --client-ca-file=/var/lib/rancher/rke2/agent/client-ca.crt --cluster-d
ns=10.43.0.10 --cluster-domain=cluster.local --container-runtime-endpoint=unix:///run/k3s/containerd/containerd.sock --eviction-hard=nodefs.available<1%,memory.available<10% --eviction-mini
mum-reclaim=imagefs.available=10%,nodefs.available=10% --eviction-soft=nodefs.available<5%,imagefs.available<1% --eviction-soft-grace-period=nodefs.available=30s,imagefs.available=30s --fai
l-swap-on=false --healthz-bind-address=127.0.0.1 --hostname-override=node1 --kubeconfig=/var/lib/rancher/rke2/agent/kubelet.kubeconfig --log-file=/var/lib/rancher/rke2/agent/logs/kubelet.lo
g --log-file-max-size=50 --logtostderr=false --node-labels=node=Master,node-name=node1 --pod-infra-container-image=index.docker.io/rancher/pause:3.6 --pod-manifest-path=/var/lib/rancher/rke
2/agent/pod-manifests --read-only-port=0 --resolv-conf=/etc/resolv.conf --serialize-image-pulls=false --stderrthreshold=FATAL --tls-cert-file=/var/lib/rancher/rke2/agent/serving-kubelet.crt
 --tls-private-key-file=/var/lib/rancher/rke2/agent/serving-kubelet.key
```

## 停止服务及卸载

```
[root@node1 ~]# rke2-
rke2-killall.sh    rke2-uninstall.sh
```

执行以上命令即可

## 导出rke2安装所需要的镜像

安装配置 nerdctl

```
# mkdir nerdctl && cd nerdctl
# wget https://github.com/containerd/nerdctl/releases/download/v0.20.0/nerdctl-
0.20.0-linux-amd64.tar.gz
# tar -xf nerdctl-0.20.0-linux-amd64.tar.gz
# cp nerdctl /usr/bin/
```

配置nerdctl 配置文件


rke2 使用containerd指定的配置文件
/var/lib/rancher/rke2/agent/etc/containerd/config.toml启动
指定 -a /run/k3s/containerd/containerd.sock socket地址

```
[root@node2 containerd]# cat /var/lib/rancher/rke2/agent/etc/containerd/config.toml | grep -Ev "^$"
[plugins.opt]
  path = "/var/lib/rancher/rke2/agent/containerd"    containerd data root
[plugins.cri]
  stream_server_address = "127.0.0.1"
  stream_server_port = "10010"
  enable_selinux = true
  sandbox_image = "index.docker.io/rancher/pause:3.6"
[plugins.cri.containerd]
  snapshotter = "overlayfs"
  disable_snapshot_annotations = true
[plugins.cri.containerd.runtimes.runc]
  runtime_type = "io.containerd.runc.v2"
```

```
# mkdir /etc/nerdctl
cat > /etc/nerdctl/nerdctl.toml << 'EOF'
debug              = false
debug_full         = false
address            = "unix:///run/k3s/containerd/containerd.sock"
data_root          = "/var/lib/rancher/rke2/agent/containerd"
namespace          = "default"
snapshotter        = "overlayfs"
cgroup_manager     = "cgroupfs"
insecure_registry = true
EOF
```

**拉取镜像**

```
nerdctl pull busybox
```

```
[root@node1 containerd]# nerdctl pull busybox
WARN[0000] skipping verifying HTTPS certs for "docker.io"
docker.io/library/busybox:latest:                                 resolved       |++++++++++++++++++++++++++++++++++++++|
index-sha256:d2b53584f580310186df7a2055ce3ff83cc0df6caacf1e3489bff8cf5d0af5d8:    done           |++++++++++++++++++++++++++++++++++++++|
manifest-sha256:52f431d980baa76878329b68ddb69cb124c25efa6e206d8b0bd797a828f0528e: done           |++++++++++++++++++++++++++++++++++++++|
config-sha256:1a80408de790c0b1075d0a7e23ff7da78b311f85f36ea10098e4a6184c200964:   done           |++++++++++++++++++++++++++++++++++++++|
layer-sha256:50e8d59317eb665383b2ef4d9434aeaa394dcd6f54b96bb7810fdde583e9c2d1:    done           |++++++++++++++++++++++++++++++++++++++|
elapsed: 15.9s                                                    total:  4.2 Ki (269.0 B/s)
```

```
[root@node1 containerd]# nerdctl images
REPOSITORY    TAG       IMAGE ID        CREATED           PLATFORM       SIZE       BLOB SIZE
busybox       latest    d2b53584f580    52 seconds ago    linux/amd64    1.2 MiB    758.9 KiB
```

使用nerdctl可以指定namespace去查看rke2的镜像，rke2的镜像在k8s.io名称空间

```
[root@node1 containerd]# nerdctl namespace ls
NAME       CONTAINERS    IMAGES    VOLUMES    LABELS
default    0             1         0
k8s.io     30            28        0
```

**指定名称空间**

```
[root@node1 containerd]# nerdctl images --namespace k8s.io
REPOSITORY                                    TAG                                 IMAGE ID        CREATED         PLATFORM       SIZE         BLOB SIZE
rancher/hardened-calico                       v3.21.4-build20220228               24a4890d8793    2 hours ago     linux/amd64    555.2 MiB    548.6 MiB
rancher/hardened-cluster-autoscaler           v1.8.5-build20211119                9220a7ac9606    2 hours ago     linux/amd64    118.8 MiB    114.7 MiB
rancher/hardened-coredns                      v1.9.1-build20220318                78bae6b08b9f    2 hours ago     linux/amd64    133.0 MiB    130.9 MiB
rancher/hardened-dns-node-cache               1.21.2-build20211119                6a7489d21ad1    2 hours ago     linux/amd64    0.0 B        137.9 MiB
rancher/hardened-etcd                         v3.5.3-k3s1-build20220413           be2fd21b3518    2 hours ago     linux/amd64    127.4 MiB    125.3 MiB
rancher/hardened-flannel                      v0.17.0-build20220317               10577c31fcb2    2 hours ago     linux/amd64    270.5 MiB    261.9 MiB
rancher/hardened-k8s-metrics-server           v0.5.0-build20211119                f6661b94e539    2 hours ago     linux/amd64    140.2 MiB    136.2 MiB
rancher/hardened-kubernetes                   v1.24.0-rke2r1-build20220505        4bdc2f4f6661    2 hours ago     linux/amd64    707.8 MiB    701.7 MiB
rancher/klipper-helm                          v0.7.1-build20220407                6dc7ee009192    2 hours ago     linux/amd64    228.5 MiB    228.2 MiB
rancher/mirrored-ingress-nginx-kube-webhook-certgen  v1.1.1                       423064c5804a    2 hours ago     linux/amd64    48.8 MiB     46.8 MiB
rancher/nginx-ingress-controller              nginx-1.2.0-hardened6               43ea317d7e30    2 hours ago     linux/amd64    559.7 MiB    553.3 MiB
```

**导出镜像到文件**

```
[root@node1 ~]# nerdctl image save --namespace k8s.io rancher/hardened-kubernetes:v1.24.0-rke2r1-build20220505 -o k8s.1.24.tar
[root@node1 ~]# ls k8s.1.24.tar
k8s.1.24.tar
[root@node1 ~]# du -sh k8s.1.24.tar
702M    k8s.1.24.tar
```

# 参考文档：


- https://mp.weixin.qq.com/s/GxrxKWaBUEx-bHWMgSvsmg
- https://blog.csdn.net/m0_49654228/article/details/120287498