29Oct2022

Day**17**

Kubernetes Add, Remove, Rejoin Node to & from Cluster

Remove Node from k8S Cluster

```
# kubectl get nodes
master1.example.com   Ready   control-plane   139d   v1.24.1
node1.example.com     Ready   <none>          139d   v1.24.1
node2.example.com     Ready   <none>          139d   v1.24.1
node3.example.com     Ready   <none>          22m    v1.24.1
node4.example.com     Ready   <none>          22m    v1.24.1
node5.example.com     Ready   <none>          22m    v1.24.1
# kubectl create deployment dpl1 --image redis --replicas 7 -o yaml --dry-run=client >dpl1.yaml
# ls
dpl1.yaml
# kubectl create deployment dpl1 --image redis --replicas 7
# kubectl get pods -o wide
```

Now, select node to remove. node3.example.com

before migration/cordon, create DaemonSet to check DaemonSet's behavior when a Node connect/disconnect to/from Cluster.

https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/

```
# vim rsyslogdmnst.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: rsyslogdmnst
  namespace: kube-system
  labels:
    app: rsyslog
spec:
  selector:
    matchLabels:
      name: rsyslogdmnst
  template:
    metadata:
      labels:
        name: rsyslogdmnst
    spec:
      containers:
      - name: rsyslogcnt
        image: rsyslog
:wq!
# kubectl apply -f rsyslogdmnst.yaml --dry-run=client
# kubectl apply -f rsyslogdmnst.yaml
# kubectl get daemonsets.apps  --namespace  kube-system
NAME          DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR        AGE
rsyslogdmnst  5         5         5       5            5           <none>               5m52s
```

Safely Drain a Node

https://kubernetes.io/docs/tasks/administer-cluster/safely-drain-node/

step1-disbale kube-scheduling on target node through 'cordon'

Cordon

```
# kubectl get nodes
node3.example.com    Ready    <none>        29m    v1.24.1
# kubectl cordon node3.example.com
# kubectl get nodes
node3.example.com    Ready,SchedulingDisabled   <none>        30m    v1.24.1
```

uncorden

```
# kubectl uncordon node3.example.com
# kubectl get nodes
```

step2-Drain target node.

migrate workloads from target node to other nodes in cluster.

Drain

```
# kubectl drain node3.example.com --force --ignore-daemonsets
node/node3.example.com already cordoned
WARNING: ignoring DaemonSet-managed Pods: kube-system/calico-node-n7jdc, kube-system/kube-proxy-k62mt, kube-system/rsyslogdmnst-tmg2r
evicting pod default/dpl1-58c79557cc-qd2wx
evicting pod default/dpl1-58c79557cc-n8279
pod/dpl1-58c79557cc-qd2wx evicted
pod/dpl1-58c79557cc-n8279 evicted
node/node3.example.com drained
# kubectl get pods -o wide
```

step3-remove target node/nodes from Cluster

Delete

```
# kubectl delete nodes node3.example.com
```

Mohammad Ali Naghval
ali@prodevans.com

Add new Node to K8s Cluster

install new Linux Box.
OS: Ubuntu 18.04LTS
u: devops
p: ubuntu
After finish Linux Box installation, login through console and do basic configuration:
u: devops
p: ubuntu
$ sudo -i
password: ubuntu
passwd
hostnamectl set-hostname node6.example.com
ip a s
route -n
cat /etc/rsyslog.conf
vim /etc/netplan/00-installer-config.yaml

```
# This is the network config written by 'subiquity'
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: false
      addresses: [192.168.29.113/24]
      gateway4: 192.168.29.1
      nameservers:
       addresses: [8.8.8.8,8.8.4.4,192.168.29.1]
```

:wq!
netplan apply
ping 8.8.8.8
Now, append node6 record in to '/etc/hosts' of all nodes in K8s Cluster
cat /etc/hsost

```
192.168.29.104  master1.example.com    master1
192.168.29.110  master2.example.com    master2
192.168.29.111  master3.example.com    master3
192.168.29.105  node1.example.com      node1
192.168.29.106  node2.example.com      node2
192.168.29.107  node3.example.com      node3
192.168.29.108  node4.example.com      node4
192.168.29.109  node5.example.com      node5
192.168.29.113  node6.example.com      node6
192.168.29.112  haproxylb.example.com   haproxylb
```

step1.prepare Node/Basic configuration
step2. install CRI, CNI, kubectl, kubeadm, kubelet
step3. check token on MasterNode
step4. join New Node to Cluster

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kube
hostname
node6.example.com
sudo cat /sys/class/dmi/id/product_uuid
Container Runtimes
https://kubernetes.io/docs/setup/production-environment/container-runtimes/
Forwarding IPv4 and letting iptables see bridged traffic
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

```
overlay
br_netfilter
EOF
```

sudo modprobe overlay
sudo modprobe br_netfilter
#cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

```
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF
```

sudo sysctl --system

containerd

Installing containerd
Option 1: From the official binaries
https://github.com/containerd/containerd/blob/main/docs/getting-started.md
https://github.com/containerd/containerd/releases
wget https://github.com/containerd/containerd/releases/download/v1.6.9/containerd-1.6.9-linux-amd64.tar.gz
ls
containerd-1.6.9-linux-amd64.tar.gz
tar Cxzvf /usr/local containerd-1.6.9-linux-amd64.tar.gz

systemd

mkdir -p /usr/local/lib/systemd/system/
vim /usr/local/lib/systemd/system/containerd.service

```
# Copyright The containerd Authors.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

[Unit]
Description=containerd container runtime
Documentation=https://containerd.io
After=network.target local-fs.target

[Service]
#uncomment to enable the experimental sbservice (sandboxed) version of containerd/cri integration
#Environment="ENABLE_CRI_SANDBOXES=sandboxed"
ExecStartPre=-/sbin/modprobe overlay
ExecStart=/usr/local/bin/containerd

Type=notify
Delegate=yes
KillMode=process
Restart=always
RestartSec=5
# Having non-zero Limit*s causes performance problems due to accounting overhead
# in the kernel. We recommend using cgroups to do container-local accounting.
LimitNPROC=infinity
LimitCORE=infinity
LimitNOFILE=infinity
# Comment TasksMax if your systemd version does not supports it.
# Only systemd 226 and above support this version.
TasksMax=infinity
OOMScoreAdjust=-999

[Install]
WantedBy=multi-user.target
```

:wq!

systemctl daemon-reload
systemctl enable --now containerd

Installing runc

https://github.com/opencontainers/runc/releases

wget https://github.com/opencontainers/runc/releases/download/v1.1.4/runc.amd64
ls
containerd-1.6.9-linux-amd64.tar.gz runc.amd64
install -m 755 runc.amd64 /usr/local/sbin/runc

Installing CNI plugins

https://github.com/containernetworking/plugins/releases

wget https://github.com/containernetworking/plugins/releases/download/v1.1.1/cni-plugins-linux-amd64-v1.1.1.tgz
ls
cni-plugins-linux-amd64-v1.1.1.tgz containerd-1.6.9-linux-amd64.tar.gz runc.amd64
mkdir -p /opt/cni/bin
tar Cxzvf /opt/cni/bin cni-plugins-linux-amd64-v1.1.1.tgz

CRI-O

https://github.com/cri-o/cri-o/blob/main/install.md#readme

APT based operating systems

NOTE

set the environment variable $OS as the appropriate OS as you installed
set the environment variable $VERSION as the appropriate CRI-O version as you want installed

echo "deb [signed-by=/usr/share/keyrings/libcontainers-archive-keyring.gpg] https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/**xUbuntu_18.04**/ /" > /etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list

echo "deb [signed-by=/usr/share/keyrings/libcontainers-crio-archive-keyring.gpg] https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable:/cri-o:/**1.24**/**xUbuntu_18.04**/ /" > /etc/apt/sources.list.d/devel:kubic:libcontainers:stable:cri-o:**1.24**.list

mkdir -p /usr/share/keyrings

curl -L https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/**xUbuntu_18.04**/Release.key | gpg --dearmor -o /usr/share/keyrings/libcontainers-archive-keyring.gpg

curl -L https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable:/cri-o:/**1.24**/**xUbuntu_18.04**/Release.key | gpg --dearmor -o /usr/share/keyrings/libcontainers-crio-archive-keyring.gpg

apt-get update

apt-get install cri-o cri-o-runc -y

systemctl daemon-reload

systemctl enable crio

systemctl start crio

Docker Engine

https://docs.docker.com/engine/install/ubuntu/

sudo apt-get update
sudo apt-get install ca-certificates curl gnupg lsb-release
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

cri-dockerd

https://github.com/Mirantis/cri-dockerd
https://www.mirantis.com/blog/how-to-install-cri-dockerd-and-migrate-nodes-from-dockershim/
wget https://github.com/Mirantis/cri-dockerd/releases/download/v0.2.0/cri-dockerd-v0.2.0-linux-amd64.tar.gz
tar xvf cri-dockerd-v0.2.0-linux-amd64.tar.gz
sudo mv ./cri-dockerd /usr/local/bin/
cri-dockerd --help
wget https://raw.githubusercontent.com/Mirantis/cri-dockerd/master/packaging/systemd/cri-docker.service
wget https://raw.githubusercontent.com/Mirantis/cri-dockerd/master/packaging/systemd/cri-docker.socket
sudo mv cri-docker.socket cri-docker.service /etc/systemd/system/
sudo sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,' /etc/systemd/system/cri-docker.service
systemctl daemon-reload
systemctl enable cri-docker.service
systemctl enable --now cri-docker.socket

verify

systemctl status cri-docker.socket
systemctl status docker.service
systemctl status containerd.service
systemctl status crio.service

Installing kubeadm, kubelet and kubectl

sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
kubectl version --short

Join NEW node to existing K8s Cluster

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/

Joining your nodes

to Join new node 'node6.example.com' new to write this command:

kubeadm join --token **<token> <control-plane-host>:<control-plane-port>** --discovery-token-ca-cert-hash sha256:**<hash>**

Now, back to MasterNode and check/create token and ca.crt

kubeadm token list

-to Create Token

kubeadm token create

ludpbj.vqvvwywe2q5qoslk

-to create discovery-token-ca-cert-hash sha256

openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | openssl dgst -sha256 -hex | sed 's/^.* //'

33db1c4ef050c21e798eddb51bf8a526b071f4f546be014faf0ef312a0546945

hostname -i

192.168.29.104

Now, back to new node, 'node6.example.com'

kubeadm join --token <token> <control-plane-host>:<control-plane-port> --discovery-token-ca-cert-hash sha256:<hash>

swapoff -a

free -m

rm -rf /etc/containerd/config.toml

systemctl restart containerd.service

kubeadm join --token ludpbj.vqvvwywe2q5qoslk 192.168.29.104:6443 --discovery-token-ca-cert-hash sha256:33db1c4ef050c21e798eddb51bf8a526b071f4f546be014faf0ef312a0546945 --cri-socket=/var/run/containerd/containerd.sock --v=5

NOTE

1-before join Debian-based OS Linux Box to existing/new K8s Cluster delete /etc/containerd/**config.toml**

root@node**6**:~# rm -rf /etc/containerd/config.toml

root@node6:~# systemctl restart containerd.service

root@node6:~# kubeadm join --token ludpbj.vqvvwywe2q5qoslk 192.168.29.104:6443 --discovery-token-ca-cert-hash sha256:33db1c4ef050c21e798eddb51bf8a526b071f4f546be014faf0ef312a0546945 --cri-socket /var/run/containerd/containerd.sock --v=5

2-if in joining time any issue happens and procedure got failed, just delete Kubernetes directory from node and again try

root@node6:~# rm -rf /etc/kubernetes/

Rejoin Deleted Node from K8s Cluster again to K8s Cluster

after delete and maintenance, need rejoin **node3.example.com** to K8s Cluster.

on node3.example.com

swapoff -a

free -m

rm -rf /etc/kubernetes/

kubeadm join --token <token> <control-plane-host>:<control-plane-port> --discovery-token-ca-cert-hash sha256:<hash>

Now, back to MasterNode:

kubectl get nodes

NAME	STATUS	ROLES	AGE	VERSION
master1.example.com	Ready	control-plane	139d	v1.24.1
node1.example.com	Ready	<none>	139d	v1.24.1
node2.example.com	Ready	<none>	139d	v1.24.1
node3.example.com	Ready	<none>	2m1s	v1.24.1
node4.example.com	Ready	<none>	150m	v1.24.1
node5.example.com	Ready	<none>	150m	v1.24.1
node6.example.com	Ready	<none>	13m	v1.25.3