K3 Official Setup Log

K3 Lab Setup using RHEL KVM & Pod Deployments using NFS PV/C

Suggestion: Getting familiar with containers, repos, networking, storage and set up some containers using Docker or Podman before possibly setting up a K3 lab. This is how I started.

Note: All items in the lab are FOSS Free and Open Source. I have this all up and running and did not have to pay anything. The only registering I had to do was register my Red Hat servers for repo access.

Logical Steps:

1. Setup LACP NIC Team (Optional)
    a. Setup LCAP Trunk
2. Setup Network Bridge
3. Install KVM as needed
4. Spin up VMs for K3s use
5. Disable SeLinux on all VM's for K3
6. Disable firewalls on all VM's for K3 (Optional)
7. Install and setup cluster IP (keepalived)
8. Setup NFS server on Hypervisor host
9. Install K3's on 1 VM (System Controller)
10. Retrieve K3 Token and register K3 Worker Node
11. Install NFS CSI Driver on K3 System Controller

K3

A) Create Namespace
B) Create Storage Class
C) Create PV
D) Create PVC
E) Create Deployment
F) Create Service

Notes:

- K3 lab setup:
    - 1 controller 1 worker
    - PV and PVC setups
    - Deployment | Pod | Container
        - Node Failover
        - Node Affinity
        - Pod Dynamic Recreation
        - Persistent Data
        - Port Mapping Service
- Team bandwidth & failover *capable switch
- Disable SeLinux recommended setting
- Firewall requires lots of management
- Cluster IP, pods are advertised on all nodes
- NFS for PV storage across all nodes

I utilized KVM for this process to allow easy fallback to older versions using snapshots and simple backups process.

I used NFS on the KVM host since it does not require outside network access and traverses at high speeds from the VM - Br0 - NVMe SSD.

Must use bridge (Direct Access) network option does not permit network access from VM to hypervisor host. Also, VM and Bridge have direct kernel access so super-fast data transfers.

K3 Setup:

A) Install K3 on first machine to be the System Controller
   a. sudo curl -sfL https://get.k3s.io | sh –
B) Retrieve the K3 token from the K3 System Controller
   a. sudo cat /var/lib/rancher/k3s/server/node-token

As per design you will most likely get a permissions issue accessing kubectl. Follow these commands to copy over the configs to your local profile and correct the issue:

 # Copy K3 config items

mkdir -p ~/.kube

sudo cp /etc/rancher/k3s/k3s.yaml ~/.kube/config

sudo chown $(whoami):$(whoami) ~/.kube/config

 # Edit and add to profile

nano ~/.bashrc

export KUBECONFIG=$HOME/.kube/config

 # Reload Profile

source ~/.bashrc

Initial setup of config points to localhost or 127.0.0.1 and you need to update the kube 'config' to point to your system controllers IP address as shown below to allow you to properly register worker nodes:

```
[shead@rhel981 ~]$ cat ~/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUS
kRRMU1USTVPRGt3SGhjTk1qVXdOREV6TURJMU5qSTVXaGNOTXpVd0
BRTUJCd05DQUFTV1ljamlrRlY5b2E0TGtWbEpaSWVIcDVzT0M5OWo
WUjBUQVFIL0JBVXdBd0VCL3pBZEJnTlZIUTRFRmdRVWF6NWtkdXBJ
SVFDbXdXNVBXUFNMSzVQWkk1czdGSjdUbnpjbWxSV2xYZzhOVWlrc
    server: https://192.168.254.81:6443
  name: default
contexts:
- context:
```

K3 Kubernetes | NFS Storage | KVM

C) Login to worker node VM and register it with K3 System Controller
   a. curl -sfL https://get.k3s.io | K3S_URL=https://(Controller IP):6443 K3S_TOKEN=(TOKEN HERE) sh
D) Log into K3 System Controller and designate as worker node
   a. kubectl label node (hostname) node-role.kubernetes.io/worker=
E) Setup node affinity labels:
   b. kubectl label nodes (hostnameA) mysql-role=primary
   c. kubectl label nodes (hostnameB) mysql-role=secondary
F) Install on all nodes:
   d. sudo dnf install -y nfs-utils
G) Install on K3 System Controller:
   e. sudo dnf install -y git
H) Install CSI NFS Driver on System Controller:
   f. cd ~
   g. git clone https://github.com/kubernetes-csi/csi-driver-nfs.git
      i. Deploy RBAC Driver:
         1. cd ~/csi-driver-nfs/deploy/
         2. kubectl apply -f rbac-csi-nfs.yaml
         3. kubectl apply -f csi-nfs-driverinfo.yaml
         4. kubectl apply -f csi-nfs-controller.yaml
         5. kubectl apply -f csi-nfs-node.yaml
I) Set up NFS Server and share and chmod to 777 and limit access by IP:
   a. sudo nano /etc/exportfs
      /home/shead/nfs-mysql 192.168.254.81(rw,no_root_squash,async,no_subtree_check) \
      192.168.254.82(rw,no_root_squash,async,no_subtree_check) \
      192.168.254.100(rw,no_root_squash,async,no_subtree_check) \
      192.168.254.201(rw,no_root_squash,async,no_subtree_check)
J) (Unrelated to CSI) Verify NFS access on all nodes:
   A) mkdir /home/shead/nfs-mysql
   B) mount 192.168.254.70/home/shead/nfs-mysql /home/shead/nfs-mysql
K) Run Namespace YAML file:
   a. Kubectl apply -f namespace.yml
L) Run Storage Class YAML file:
   a. Kubectl apply -f StorageClass.yml
M) Check Storage Class:
   a. kubectl get storageclass -n mysql-namespace
   b. kubectl describe storageclass nfs-csi-sc -n mysql-namespace
N) Run PV YAML file:
   a. Kubectl apply -f pv.yml

O) Check PV status
    a. Kubectl get pv –n mysql-namespace
    b. kubectl describe pv nfs-pv -n mysql-namespace
P) Run PVC YAML file:
    a. Kubectl apply -f pvc.yml
Q) Check PVC status
    a. kubectl describe pvc nfs-pvc -n mysql-namespace
    b. Kubectl get pvc –n mysql-namespace
R) Run Deployment YAML file:
    a. Kubectl apply -f deployment.yml
S) Check Deployment and Pod
    a. Kubectl get deployment –n mysql-namespace
    b. Kubectl get pods –n mysql-namespace -o wide
T) Run Service YAML file:
    a. Kubectl apply -f service.yml
U) Check Service
    a. kubectl get svc mysql-service
    b. kubectl describe svc mysql-service