ProLUG – K3s

Required Materials

Rocky 9 or equivalent

Root or sudo command access

EXERCISES (Warmup and quick review)

1. curl -sfL https://get.k3s.io > /tmp/k3_installer.sh
2. more /tmp/k3_installer.sh
   #What do you notice in the installer?
   #What checks are there?
3. grep -i arch /tmp/k3_installer.sh
   #What is the name of the variable holding the architecture?
   #How is the system finding that variable?
4. uname -m
   #Verify your system architecture

5. grep -iE “selinux|sestatus” /tmp/k3_installer.sh
   #Does K3s check if selinux is running, or no?

Installing k3s and looking at how it interacts with your Linux system

1. Install k3s

```
curl -sfL https://get.k3s.io | sh -
```

2. What was installed on your system?

```
rpm -qa –last | tac
```

Your tasks in this lab are designed to get you thinking about how container deployments interact with

3. Verify the service is running

```
systemctl status k3s
```

4. Check it systemd configuration

```
systemctl cat k3s
```

5. See what files and ports it is using

```
ss -ntulp | grep <pid from 3>
lsof -p <pid from 3>

#Do you notice any ports that you did not expect?
```

6. Verify simple kubectl call to API

```
kubectl get nodes
```

7. Verify K3s is set to start on boot and then cycle the service

```
systemctl is-enabled k3s
systemctl stop k3s
#Recheck your steps 3-6
  #What error do you see, and is it expected?
  #What is the API port that kubectl is failing on?
systemctl start k3s
#Verify your normal operations again.
```

Looking at the K3 environment

1. Check out components

```
kubectl version

kubectl get nodes

kubectl get pods -A

kubectl get namespaces

kubectl get configmaps -A

kubectl get secrets -A

  #Which namespace seems to be the most used?
```

Creating Pods, Deployments, and Services

It’s possible that the lab will fail in this environment. Continue as you can and identify the problem using the steps at the end of this section.

1. Create a pod named webpage with the image nginx

```
kubectl run webpage --image=nginx
```

2. Create a pod named database with the image redis and labels set to tier=database

```
kubectl run database --image=redis --labels=tier=database
```

3. Create a service with the name redis-service to expose the database pod within the cluster on port 6379 (default Redis port)

```
kubectl expose pod database --port=6379 --name=redis-service --
type=ClusterIP
```

4. Create a deployment called web-deployment using the image nginx that has 3 replicas

```
kubectl create deployment web-deployment --image=nginx --
replicas=3
```

5. Verify that the pods are created

```
kubectl get deployments
```

```
kubectl get pods
```

6. Create a new namespace called my-test

```
kubectl create namespace my-test
```

7. Create a new deployment called redis-deploy with the image redis in your my-test namespace with 2 replicas

```
kubectl create deployment redis-deploy -n my-test --image=redis -
-replicas=2
```

Do some of your same checks from before. What do you notice about the pods you created? Did they all work?

If this breaks in the lab, document the error. Check your disk space and RAM, the two tightest constraints in the lab. Using systemctl restart k3s and journalctl -xe can you figure out what is failing? (Rocky boxes may have limitations that cause this to not fully deploy, can you find out why?)

Conclusion

There are a lot of ways to use these tools. There are a lot of ways you will support them. At the end of the day you’re a Linux System Administrator, you’re expected to understand everything that goes on in your system. To this end, we want to know the build process and run processes so we can help the engineers we support keep working in a Linux environment.
We did not declaratively deploy anything, but read about how that would work and we’ll keep working it into the course in the future.

Notes and resources used:
https://kubernetes.io/docs/concepts/overview/