K3sInstallationLab

k3s Installation

Introduction

In this lab we will be installing k3s.

Prerequisite

All of our labs are built upon one another. Please make sure all previous labs are completed before starting this lab.

Workflow

Hardware

For this lab we are going to need at least two virtual machine or a physical machine to install k3s on. The machines need to be of a particular architecture outlined here:

- x86_64
- armhf
- arm64/aarch64
- s390x

Each of these machines need the minimum requirements:

- CPU: 1 core
- Ram: 512 MB

Recommended:

- CPU: 2 cores
- RAM: 1 GB

Operating System

K3s will work on most modern Linux system. Note there are differences for CentOS/Red Hat and Raspberry Pi OS.

For these labs we will be using openSUSE Leap.

---

k3s Version

In this course we are going to do an upgrade to a newer version of k3s. To do this we are going to need to find the latest version of k3s and take the stable release one minor version back.

For example if the latest release of the latest version of k3s is `v1.27.4+k3s1` we are going to want to use the `v1.26.7+k3s1` of k3s.

You can find the latest versions of k3s here.

NOTE: don't try this lab with any marked as `Pre-release`.

Server Script Install

One of the easiest ways to install k3s is by the installation script.

You will need to `ssh` in to the `server` node.

The install script lives here https://get.k3s.io/. To run this we are going to use a few flags:

```
curl -sfL https://get.k3s.io | INSTALL_K3S_VERSION=v1.26.7+k3s1
INSTALL_K3S_EXEC="server --cluster-init --node-ip 192.168.3.2 --node-
external-ip 192.168.3.2" sh -s -
```

NOTE: the k3s version might not be the one prior to the latest! Check here for the correct version.

This setups up the `server` installation on our `server` VM.

To test to see if everything is running *k3s* comes with `kubectl` so while still logged in to our `server` instance we can do a quick check to make sure all is running:

```
k3s kubectl get pods -A
```

And you should see the following output:

```
NAMESPACE     NAME                                      READY   STATUS
 RESTARTS   AGE
kube-system   local-path-provisioner-957fdf8bc-sd297    1/1     Running
0           107s
kube-system   coredns-77ccd57875-sbrsl                  1/1     Running
0           107s
kube-system   helm-install-traefik-crd-7926x            0/1     Completed
0           107s
kube-system   helm-install-traefik-rswvl                0/1     Completed
1           107s
kube-system   metrics-server-648b5df564-5xbcw           1/1     Running
```

```
0           107s
kube-system   svclb-traefik-ea396d7d-4h9f6             2/2     Running
0           53s
kube-system   traefik-64f55bb67d-5cqxm                 1/1     Running
0           53s
```

Once you see that the system pods are either `Running` or `Completed` we need to get the token from the server to do so run the following command:

```
sudo cat /var/lib/rancher/k3s/server/node-token
```

You will get the token needed for adding the agent (Note: it will not be this exact key):

```
K105b9c68df80752c8f9d498097b764a35f9ba2c0220b2ea8951cef3aca111d9f33::serve
r:2855bde078f38f3964f3f36e6e37dfbb
```

To logout of the `server` vm.

Agent Script Install

Now we need to install `k3s` on the `agent` vm.

First let's remote in to our `agent` vm.

Again the install script lives here https://get.k3s.io/. To run this we are going to use a few flags (Note: this will take a few moments.):

```
curl -sfL https://get.k3s.io | INSTALL_K3S_VERSION=v1.26.7+k3s1
INSTALL_K3S_EXEC="agent --server https://192.168.3.2:6443 --node-ip
192.168.3.3 --node-external-ip 192.168.3.3"
K3S_TOKEN=K105b9c68df80752c8f9d498097b764a35f9ba2c0220b2ea8951cef3aca111d9
f33::server:2855bde078f38f3964f3f36e6e37dfbb K3S_NODE_NAME=agent sh -
```

NOTE: the k3s version might not be the one prior to the latest! Check here for the correct version.

NOTE: The `K3S_TOKEN` is the token we pulled from the `server` earlier.

`kubectl` Remotely

Just so we don't have to remain logged into the `server` VM let's copy down the config:

```
ssh server -c "sudo cat /etc/rancher/k3s/k3s.yaml" > k3s.yaml
```

This will copy down the `KUBECONFIG` file down from the server.

We are going to need to modify the `k3s.yaml` file:

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRU
    server: https://127.0.0.1:6443
  name: default
contexts:
- context:
    cluster: default
    user: default
  name: default
current-context: default
kind: Config
preferences: {}
users:
- name: default
  user:
    client-certificate-data: LS0tLS1CRU
    client-key-data: LS0tLS1CRU
```

(Note: the certificate data has been cut down.)

We are going to need to change the variable of the `server` from:

```
server: https://127.0.0.1:6443
```

To the following:

```
server: https://192.168.3.2:6443
```

Now we are going to copy this `k3s.yaml` file to our `.kube` directory:

```
cp k3s.yaml $HOME/.kube
```

From there we are going to set the `KUBECONFIG` variable to the location of the `k3s.yaml` file:

```
export KUBECONFIG=$HOME/.kube/k3s.yaml
```

To test to see if this works we can run the following this command:

```
kubectl get nodes
```

And you should see something like this:

```
NAME     STATUS   ROLES                        AGE    VERSION
agent    Ready    <none>                       21s    v1.26.7+k3s1
server   Ready    control-plane,etcd,master    3m9s   v1.26.7+k3s1
```
