Setting up a k3s cluster

Background

This document captures my "copy/paste" and other details as I try to setup a k3 cluster.

Walkthrough

Using the k3s-ansible repository

There is a k3s-ansible repository that may be easier to work with. I'll fork it and then use it as a basis of my own version.

Repository	https://github.com/tlhakhan/k3s-ansible

```
# get helm
https://github.com/helm/helm/releases

# get the linux amd64 version
wget https://get.helm.sh/helm-v3.8.2-linux-amd64.tar.gz
tar -zxvf helm-v3.8.2-linux-amd64.tar.gz
cd linux-amd64
mv helm /sbin
cd ..
rm -rf linux-amd64

# add helm stable repo
helm repo add rancher-stable https://releases.rancher.com/server-charts/stable

# add namespace
kubectl create namespace cattle-system

# helm needs to know location of kubeconfig file
export KUBECONFIG=/etc/rancher/k3s/k3s.yaml

# setup
helm install rancher rancher-stable/rancher \
  --namespace cattle-system \
  --set hostname=rancher.tenzin.io \
  --set replicas=3 \
  --set ingress.tls.source=secret

# DNS was created for rancher.tenzin.io and CNAMEd to kube-1, the master node.
# VMware vSwitch needs to be configured with promiscuous mode, forged transmits and MAC changes.
```

Playbook breakdown

Inventory breakdown

group_vars/all.yml

```
---
k3s_version: v1.22.3+k3s1
ansible_user: sysuser
systemd_dir: /etc/systemd/system
master_ip: "{{ hostvars[groups['master'][0]]['ansible_host'] | default(groups['master'][0]) }}"
extra_server_args: ""
extra_agent_args: ""
```

Notes

- The `group_vars` has most of the important details easy to configure.
- The slightly complex piece is the selection of the `master_ip` . It's trying to be smart in its "default" selection of the master host's IP address.
    - It appears that if you define an `ansible_host` as an override in the hosts file then it can take precedence over the inventory_hostname.

hosts.ini

```
[master]
kube-1

[node]
kube-2
kube-3

[k3s_cluster:children]
master
node
```

Notes

- The hosts file has two main groups, master and node.
- The k3s_cluster is composed of the two groups.
- This would create a single node master, which may not be good from a fault tolerance standpoint.

Entrypoint breakdown

```
---
- hosts: k3s_cluster
  gather_facts: yes
  become: yes
  roles:
    - role: prereq
    - role: download

- hosts: master
  become: yes
  roles:
    - role: k3s/master

- hosts: node
  become: yes
  roles:
    - role: k3s/node
```

Notes

- Simple use of multiple playbook stanzas in a single playbook. It begins by running the `prereq` and `download` roles on the entire cluster members.
- It then runs specific roles on the two k3s_cluster sub-groups.
    - The `k3s/master` role would need to be extended to support high availability.

Prereq role breakdown

```
prereq
 tasks
     main.yml
```

This role appears to setup some general OS specific requirements for Kubernetes to work.

- It turns off SELinux if the OS is CentOS.
- It enables IPv4 and IPv6 forwarding.
- It enables and loads the `br_netfilter` module if CentOS.
    - I checked on Debian 11 and it appears to be loaded by default.

```
# lsmod | grep netfilter
br_netfilter           32768  0
bridge                253952  1 br_netfilter
```

- It wants the bridge-nf-call-iptables to be enabled.
    - I checked on Debian 11 and it appears to be enabled by default.

```
# sysctl -a | grep net.bridge.bridge-nf-call-ip
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
```

- It updates the /etc/sudoers and adds /usr/local/bin to the secure_path list./

Download role breakdown

```
download
 tasks
     main.yml
```

This role focuses only on downloading the binary and verifying its checksum. It tries to target three platforms: x64, arm64 and armhf.

- It places the binary into /usr/local/bin. The k3s is a single binary Kubernetes distribution.

k3s/master role breakdown

```
k3s/master
 defaults
    main.yml
 tasks
    main.yml
 templates
     k3s.service.j2
```

This role is run on the master node(s). Presently it's designed for a single master and will need to be enhanced for a HA master setup.

The defaults contain only a single variable.

```
k3s_server_location: /var/lib/rancher/k3s
```

- This folder appears to be the location of the `node-token` file. It appears to be important and most likely used in the the `k3s/node` role.
- This variable is used in the `ExecStart` of the systemctl service file. It is referenced as the `--data-dir` argument.

The k3s.service.j2 template.

k3s.service.j2

```
[Unit]
Description=Lightweight Kubernetes
Documentation=https://k3s.io
After=network-online.target

[Service]
Type=notify
ExecStartPre=-/sbin/modprobe br_netfilter
ExecStartPre=-/sbin/modprobe overlay
ExecStart=/usr/local/bin/k3s server --data-dir {{ k3s_server_location }} {{ extra_server_args | default("") }}
KillMode=process
Delegate=yes
# Having non-zero Limit*s causes performance problems due to accounting overhead
# in the kernel. We recommend using cgroups to do container-local accounting.
LimitNOFILE=1048576
LimitNPROC=infinity
LimitCORE=infinity
TasksMax=infinity
TimeoutStartSec=0
Restart=always
RestartSec=5s

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

The tasks does the following:

- Setup the systemd service. The service is called `k3s.service`.
  - The service is enabled and restarted.
- The role then waits for the `node-token` file to be created by the service start up.
  - It does something weird, where it saves the permission mode, then changes to +rx on group and owner, reads the file and then reverts the permission mode. I'm not sure why this is necessary.
  - The `node-token` is then saved as a fact after base64 decode and trimming new line.
- It creates the `~/.kube` folder.
  - It then copies the `k3s.yaml` located in `/etc/rancher/k3s` folder to the `~/.kube/config` file.
- It then updates the the kubectl config, where the --server is updated to the `master_ip`.

```
- name: Replace https://localhost:6443 by https://master-ip:6443
  command: >-
    k3s kubectl config set-cluster default
      --server=https://{{ master_ip }}:6443
      --kubeconfig ~{{ ansible_user }}/.kube/config
  changed_when: true
```

- It then closes by updating symlinks, for crictl and kubectl to point to the `/usr/local/bin/k3s` binary.

k3s/node role breakdown

```
k3s/node
 tasks
    main.yml
 templates
     k3s.service.j2
```

This role runs on the worker node and setups the service `k3s-node.service` and joins it to the control plane (master).

The template sets up the systemd service.

- It uses the `node-token` content set as a fact on the master host in the `ExecStart` line.

k3s.service.j2

```
[Unit]
Description=Lightweight Kubernetes
Documentation=https://k3s.io
After=network-online.target

[Service]
Type=notify
ExecStartPre=-/sbin/modprobe br_netfilter
ExecStartPre=-/sbin/modprobe overlay
ExecStart=/usr/local/bin/k3s agent --server https://{{ master_ip }}:6443 --token {{ hostvars[groups['master']
[0]]['token'] }} {{ extra_agent_args | default("") }}
KillMode=process
Delegate=yes
# Having non-zero Limit*s causes performance problems due to accounting overhead
# in the kernel. We recommend using cgroups to do container-local accounting.
LimitNOFILE=1048576
LimitNPROC=infinity
LimitCORE=infinity
TasksMax=infinity
TimeoutStartSec=0
Restart=always
RestartSec=5s

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

Appendix

	Document	Link
1	k3s.io	https://k3s.io/
2	Latest k3s release	https://github.com/rancher/k3s/releases/latest
3	Installing rancher on k3s	https://rancher.com/docs/rancher/v2.5/en/installation/install-rancher-on-k8s/