**Write an Ansible Playbook to Install and Configure Kubernetes Cluster on RHEL 9**

**Setup Kubernetes using Ansible Script**

Kubernetes has emerged as the go-to platform for container orchestration. It automates the deployment, scaling, and operations of application containers across clusters of hosts, providing a robust infrastructure for deploying applications consistently.

Setting up a Kubernetes cluster can be complex and time-consuming. This is where Ansible, a powerful automation tool, comes into play. By using Ansible scripts, you can automate the setup of Kubernetes clusters, ensuring a repeatable and error-free installation process. This article will guide you through setting up Kubernetes using Ansible scripts, making the process efficient and straightforward.

**Understanding Kubernetes and Ansible**

Kubernetes, often abbreviated as K8s, is an open-source platform designed to automate the deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery. Key features of Kubernetes include:

**Prerequisites**

# vim /etc/NetworkManager/NetworkManager.conf

[main]

dns=none

# systemctl restart NetworkManager.service

# vim /etc/resolv.conf

Comment out this two lines:

#nameserver 2001:558:feed::1

#nameserver 2001:558:feed::2

# systemctl restart NetworkManager.service

# more /etc/resolv.conf

Before setting up Kubernetes using Ansible, ensure you have the following prerequisites:

Hardware Requirements

At least 2 GB of RAM per machine.

At least 2 CPUs per machine.

At least 20 GB of disk space per machine

**Software Requirements**

* **Ansible:** Ensure Ansible is installed on your control node.
* **Kubernetes:** Kubernetes components (kubeadm, kubelet, kubectl) must be available for installation.
* **Virtual Machines or Cloud Instances:** You need multiple machines (virtual or physical) to form a Kubernetes cluster.

**Writing the Ansible Playbook**

**Structure of an Ansible Playbook**

Ansible playbooks are written in YAML and consist of plays, tasks, handlers, variables, and templates. Here is a basic structure:

- hosts: all  
 become: yes  
 tasks:  
 - name: Task description  
 module:   
 options:

**Creating a Playbook for Kubernetes Setup**

Below is a simplified example of an Ansible playbook to install Kubernetes:

# cd /home/hogege/ansible\_automation

# vim install\_k8s\_cluster.yaml

---

- name: Install and configure Kubernetes cluster on RHEL 9

hosts: master

become: yes

vars:

kube\_version: "1.31.1"

pod\_network\_cidr: "192.168.0.0/16"

tasks:

- name: Disable SELinux enforcement

ansible.posix.selinux:

state: permissive

policy: targeted

- name: Disable firewalld

ansible.builtin.systemd:

name: firewalld

state: stopped

enabled: no

- name: Disable swap

ansible.builtin.command: swapoff -a

when: ansible\_swaptotal\_mb > 0

- name: Remove swap from fstab

ansible.builtin.replace:

path: /etc/fstab

regexp: '^.\*swap.\*$'

replace: ''

- name: Install required packages

ansible.builtin.dnf:

name:

- curl

- wget

- vim

- git

- net-tools

- containerd

state: present

- name: Enable and configure containerd

block:

- name: Generate default config for containerd

ansible.builtin.command: containerd config default

register: containerd\_config

- name: Create containerd config file

ansible.builtin.copy:

dest: /etc/containerd/config.toml

content: "{{ containerd\_config.stdout }}"

- name: Enable SystemdCgroup in containerd

ansible.builtin.replace:

path: /etc/containerd/config.toml

regexp: 'SystemdCgroup = false'

replace: 'SystemdCgroup = true'

- name: Restart containerd

ansible.builtin.systemd:

name: containerd

state: restarted

enabled: yes

- name: Load br\_netfilter module

ansible.builtin.shell: modprobe br\_netfilter

- name: Apply sysctl params for Kubernetes networking

ansible.builtin.blockinfile:

path: /etc/sysctl.d/k8s.conf

block: |

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip\_forward = 1

net.bridge.bridge-nf-call-ip6tables = 1

create: yes

- name: Reload sysctl

ansible.builtin.command: sysctl --system

- name: Add Kubernetes YUM repository

ansible.builtin.yum\_repository:

name: kubernetes

description: Kubernetes Repo

baseurl: https://pkgs.k8s.io/core:/stable:/v{{ kube\_version.split('.')[0] }}.{{ k

ube\_version.split('.')[1] }}/rpm/

enabled: yes

gpgcheck: yes

repo\_gpgcheck: yes

gpgkey: https://pkgs.k8s.io/core:/stable:/v{{ kube\_version.split('.')[0] }}.{{ ku

be\_version.split('.')[1] }}/rpm/repodata/repomd.xml.key

- name: Install Kubernetes tools

ansible.builtin.dnf:

name:

- kubelet

- kubeadm

- kubectl

state: present

- name: Enable kubelet

ansible.builtin.systemd:

name: kubelet

enabled: yes

state: started

- name: Initialize Kubernetes control plane with kubeadm

shell: kubeadm init --pod-network-cidr={{ pod\_network\_cidr }} > /root/kubeadm-init.

log 2>&1

register: kubeadm\_init

ignore\_errors: yes

- name: Create .kube directory

file:

path: /root/.kube

state: directory

mode: '0755'

- name: Copy admin.conf to user's kube config

copy:

src: /etc/kubernetes/admin.conf

dest: /root/.kube/config

remote\_src: yes

owner: root

group: root

mode: '0644'

- name: Wait for Kubernetes API server to be ready

uri:

url: https://localhost:6443/healthz

method: GET

validate\_certs: no

status\_code: 200

register: result

retries: 20

delay: 10

until: result.status == 200

- name: Install Calico CNI

ansible.builtin.shell: |

kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/m

anifests/calico.yaml

environment:

KUBECONFIG: /etc/kubernetes/admin.conf

- name: Generate join command for worker nodes

shell: kubeadm token create --print-join-command

register: join\_cmd

- name: Set join command as a fact

set\_fact:

kube\_join\_command: "{{ join\_cmd.stdout }}"

- name: Save join command to a file for sharing with worker nodes

copy:

dest: /root/kubeadm-join.sh

content: "#!/bin/bash\n{{ kube\_join\_command }}\n"

mode: '0755'

- name: Join worker nodes to Kubernetes cluster

hosts: workers

become: yes

vars:

kube\_join\_command: ""

tasks:

- name: Fetch kubeadm join script from master

fetch:

src: /root/kubeadm-join.sh

dest: /tmp/kubeadm-join.sh

flat: yes

delegate\_to: k8s-master.henogez.com

- name: Copy join script to worker

copy:

src: /tmp/kubeadm-join.sh

dest: /root/kubeadm-join.sh

mode: '0755'

- name: Join node to cluster

shell: /root/kubeadm-join.sh

# sudo ansible-playbook install\_k8s\_cluster.yaml --syntax-check

# sudo ansible-playbook install\_k8s\_cluster.yaml

**Kubernetes Cluster Node/Master Node (k8s-master-henogez.com:**

# kubectl get nodes

**ERROR MESSAGE:**

# kubectl create namespace argocd

Error from server (Forbidden): namespaces is forbidden: User "system:node:k8s-master.henogez.com" cannot create resource "namespaces" in API group "" at the cluster scope

**RESOLUTION:**

# export KUBECONFIG=/etc/kubernetes/admin.conf

**ERROR MESSAGE:**

calico-node-tkhzr 0/1 CrashLoopBackOff 2 (60s ago) 4m11s

calico-node-whlqs 0/1 CrashLoopBackOff

**SOLUTION:**

**Downgrade calicoctl to Match Cluster**

If you want to avoid using the flag every time, you can install calicoctl v3.27.0 instead:

# curl -O -L https://github.com/projectcalico/calico/releases/download/v3.27.0/calicoctl-linux-amd64

# chmod +x calicoctl-linux-amd64

# sudo mv calicoctl-linux-amd64 /usr/local/bin/calicoctl

**Then test again:**

# calicoctl get bgppeers -o wide

**Edit the Calico config:**

# kubectl edit configmap calico-config -n kube-system

Find the key named **calico\_backend** and change its value:

calico\_backend: "vxlan"

**Restart Calico to Apply the Changes**

**Run:**

# kubectl rollout restart daemonset calico-node -n kube-system

**Then monitor:**

# kubectl get pods -n kube-system -l k8s-app=calico-node -w

**Step-by-Step: Fully Switch Calico to VXLAN**

We’ll do 3 things:

1. **Edit calico-config ConfigMap** (you did this).
2. **Patch the IPPool to use VXLAN instead of BGP**.
3. **Restart Calico to apply everything.**

**Use Correct Field Names for VXLAN Mode**

Instead of setting "encapsulation": "VXLAN", you need to disable IPIP and explicitly enable VXLAN using the vxlanMode field.

Correct Patch for VXLAN

**Run this:**

# calicoctl patch ippool default-ipv4-ippool --patch '{

"spec": {

"ipipMode": "Never",

"vxlanMode": "Always",

"natOutgoing": true,

"nodeSelector": "all()"

}

}' --allow-version-mismatch

**Then Verify:**

# calicoctl get ippools -o yaml --allow-version-mismatch

**Look for:**

ipipMode: Never

vxlanMode: Always

**Restart Calico After the Change:**

# kubectl rollout restart daemonset calico-node -n kube-system

**Then:**

# kubectl get pods -n kube-system -l k8s-app=calico-node -w

**Check interfaces on the node:**

# ip link show

# ip route show table 254

**Please make sure you did all cleanup prior to re-init:**

# kubeadm reset -f

# rm -rf /etc/kubernetes/pki /etc/kubernetes/manifests /var/lib/etcd

# rm -rf /etc/kubernetes/ /var/lib/etcd ~/.kube

# systemctl restart kubelet