Setting Up a Customized Multi-Node Kubernetes Cluster with CRI-O and Calico Networking

Introduction

In the vast landscape of container orchestration, Kubernetes stands as a powerful tool, automating the deployment, scaling, and management of containerized applications. However, setting up a multi-node cluster can be challenging. This guide aims to demystify the process, exploring components, concepts, and tools involved in creating a customized Kubernetes cluster.

Components Overview

Kubernetes

Kubernetes, often abbreviated as K8s, is an open-source container orchestration platform, streamlining application deployment and management.

CRI-O

CRI-O is a lightweight container runtime interacting with the Kubernetes API server, crucial for efficient container management.

kubeadm

Kubeadm simplifies setting up Kubernetes clusters by installing and configuring necessary components, providing a streamlined cluster creation process.

Container Engine (CRI-O)

Chosen for its speed and adherence to standards, CRI-O integrates seamlessly with Kubernetes, making it an ideal choice for the cluster.

kubelet

Kubelet, an agent on each node, ensures containers are running in Pods, the smallest Kubernetes objects representing running processes.

kube-proxy

Kube-proxy, a network proxy on each node, maintains network rules allowing communication to Pods within and outside the cluster.

kubectl

kubectl, a command-line tool, serves as the main interface for interacting with Kubernetes clusters, essential for developers, operators, and administrators.

The Need for a Customized Approach

While cloud services and tools like Minikube offer pre-configured Kubernetes setups, a customized approach becomes essential for granular control. This guide focuses on Kubeadm for setting up Kubernetes tailored to specific requirements.

Prerequisites

Before diving into the cluster creation process, ensure a clear understanding of:

- kubectl Command: Essential for interacting with Kubernetes clusters.
- Container Basics: Fundamentals of containers, Docker/Podman, and container images.
- Kubernetes Services: Familiarity with various services and components within a Kubernetes environment.

Container Runtimes: Docker, Podman, and CRI-O

Choosing a container runtime is critical. While Docker and Podman are popular, CRI-O stands out for speed and adherence to standards. This guide opts for CRI-O due to its efficiency and seamless integration with Kubernetes.

Networking in a Multi-Node Cluster

In a multi-node Kubernetes cluster, networking plays a pivotal role. This guide employs Software-Defined Networking (SDN) with VLANs to connect independent nodes, facilitating communication between nodes and containers.

Step-by-Step Guide

Prerequisites

Infrastructure: Utilize a cloud service like AWS EC2 to launch three instances - one master node and two worker nodes.
Instance Configuration: Choose a reliable operating system (e.g., RHEL 9.2) with 4GB RAM and 2 CPUs for each worker node.
Container Engine: Opt for CRI-O as the container engine for its lightweight nature.
Kubernetes Tools: Install essential Kubernetes tools, primarily kubeadm, assisting in configuring the master and worker nodes.

Cluster Initialization

Disable Swap Configuration: Swap configurations can interfere with Kubernetes. Disable them using `swapoff -a`.
Traffic Control Utility: Install the traffic control utility package for effective network management: `dnf install -y iproute-tc`.
Overlay for Networking: Enable overlay for networking using:

```
modprobe overlay
modprobe br_netfilter
```

Network Drivers: Configure net filter drivers permanently:

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
```

Enable IP Forwarding and Network Settings:

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
sysctl -p
```

SELINUX Configuration:

```
setenforce 0
sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

Installing CRI-O and Kubernetes Tools

Install CRI-O:

```
PROJECT_PATH=prerelease:/main
cat <<EOF | tee /etc/yum.repos.d/cri-o.repo
[cri-o]
name=CRI-O
baseurl=https://pkgs.k8s.io/addons:/cri-o:/$PROJECT_PATH/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/addons:/cri-o:/$PROJECT_PATH/rpm/repodata/repomd.xml
.key
EOF

dnf install -y cri-o
systemctl enable –now crio
systemctl status cri-o
```

Install Kubernetes Tools:

```
KUBERNETES_VERSION=v1.29
cat <<EOF | tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/$KUBERNETES_VERSION/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/$KUBERNETES_VERSION/rpm/repodata/repo
md.xml.key
EOF

dnf install -y cri-o kubelet kubeadm kubectl
systemctl enable –now kubelet
systemctl status kubelet
```

Install Net-tools:

yum install net-tools

Master Node Configuration

Initialize Kubernetes on Master Node:

kubeadm init –pod-network-cidr=192.168.0.0/16

Configure kubectl for User:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Disable Taint on Master Node
kubectl taint nodes –all node-role.kubernetes.io/control-plane-

Joining Worker Nodes

Obtain Join Token from Master Node:
kubeadm token create –print-join-command

kubeadm token create –print-join-command

Run this command on the master node.
Paste Token on Worker Nodes:
On each worker node, paste the token obtained from the master node.

In Master Node

Labeling and Verifying

Label Worker Nodes:

kubectl label node [worker-node-hostname] node-role.kubernetes.io/worker=worker

Verify Cluster Information:

kubectl cluster-info

Installing Calico for Networking

Delete Existing Pods and Deployments:

kubectl delete all –all

Download and Apply Calico:

After Installing Calico CNI, nodes state will change to Ready state, DNS service inside the cluster would be functional and containers can start communicating with each other.

To install Calico CNI, run the following command from the master node

$ kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml

or

$ kubectl create -f
https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/manifests/tigera-operator.yaml

Before creating this manifest, read its contents and make sure its settings are correct for your environment. For example, you may need to change the default IP pool CIDR to match your pod network CIDR.

$ kubectl create -f
https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/manifests/custom-resources.yaml

$ watch kubectl get pods -n calico-system or

curl -O
https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml

or

(this link work perfectly)

https://raw.githubusercontent.com/projectcalico/calico/master/manifests/calico.yaml

Modify Calico File for Network Card Detection: Update the `calico.yaml` file to specify the correct network interface.

vi calico.yaml

Auto-detect the BGP IP address.

name: IP

value: “autodetect”

name: IP_AUTODETECTION_METHOD

value: “interface=eth0”

Apply Calico:

kubectl apply -f calico.yaml

kubectl get pods -n kube-system

Deploying Applications and Services

Create a Deployment:
kubectl create deployment lwdeploy1 -image=vimal13/apache-webserver-php -replicas=5

Expose Deployment with NodePort Service:

kubectl expose deployment lwdeploy1 –type=NodePort –port=80

Access Services from Outside:

Use the public IP address of your instances along with the assigned NodePort to access your deployed services.

***Make Sure to Allow Firewall and Configure inbound rules in Security groups under Security section Of AWS Cloud instance if doing setup on AWS cloud similar for any other platforms as well.**

Conclusion

Congratulations! You've successfully set up a multi-node Kubernetes cluster with Calico for efficient networking and CRI-O as the container engine. This robust cluster is ready to deploy and scale containerized applications seamlessly.

Check_Out_Detailed_Blog:-https://medium.com/@srivastavayushmaan1347/comprehensive-guide-to-setting-up-a-customized-multi-node-kubernetes-cluster-with-cri-o-and-calico-e3e65c39b8b9