Kubernetes v1.25.1 Setup

Last update 15.09.2022

Contents

[Preconditions 1](#_Toc115196021)

[Install Control Plane 2](#_Toc115196022)

[Preconditions 2](#_Toc115196023)

[Install Work node 5](#_Toc115196024)

[Install Network Plugin on control plane 7](#_Toc115196025)

[Install Helm charts 8](#_Toc115196026)

[Generate Infrastructure configuration using Terraform scripts 8](#_Toc115196027)

[Global Kubernetes namespace setup 8](#_Toc115196028)

[Create Service Account for Terraform Script 8](#_Toc115196029)

[Create Private Docker Images Registry 8](#_Toc115196030)

[Create Ingress controller and resource 10](#_Toc115196031)

[Create SSL Issues repository 10](#_Toc115196032)

[Create Kafka Broker Pod 10](#_Toc115196033)

[Create Spring Boot Docker images 10](#_Toc115196034)

[Create Dummy Spring Boot Docker image 11](#_Toc115196035)

[Deploy Spring Boot Application 11](#_Toc115196036)

[Troubleshooting 11](#_Toc115196037)

# Preconditions

Table Hardware Requirements

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Server | CPU | RAM | HDD | Network | Description |
| kubernetes\_1 | 4 vCPU Cores | 8 GB | 200 GB | 1 Network interface | Control Plane |
| kubernetes\_2 | 4 vCPU Cores | 8 GB | 200 GB | 1 Network interface | Work Node |
|  |  |  |  |  |  |

Ref <https://www.linuxtechi.com/install-kubernetes-on-ubuntu-22-04/>

# Install Control Plane

## Preconditions

Install Ubuntu Server 22.04 TLS

Min 2 CPU cores

Min 4 GB ram

Set static IPs for all servers. Set network adapter to bridged if the server is VM running on Virtualbox.

*sudo apt-get install vim*

*sudo apt-get install net-tools*

*sudo vi /etc/netplan/00-installer-config.yaml*

network:

version: 2

renderer: networkd

ethernets:

enp0s3:

addresses:

- 192.168.1.126/24

nameservers:

addresses: [8.8.8.8, 8.8.4.4]

routes:

- to: default

via: 192.168.1.1

*sudo netplan --debug apply*

Change hostname

sudo hostnamectl set-hostname kubernetes1

Configure username

*sudo adduser kubernetes*

*sudo swapoff -a*

Remove following line from /etc/fstab

/swap.img none swap sw 0 0

sudo swapoff -a

*sudo sed -i '/ swap / s/^\(.\*\)$/#\1/g' /etc/fstab*

sudo tee /etc/modules-load.d/containerd.conf <<EOF

overlay

br\_netfilter

EOF

*sudo modprobe overlay*

*sudo modprobe br\_netfilter*

sudo tee /etc/sysctl.d/kubernetes.conf <<EOF

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

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

net.ipv4.ip\_forward = 1

EOF

*sudo sysctl --system*

sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

sudo apt update

sudo apt install -y containerd.io

containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1

sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml

sudo systemctl restart containerd

sudo systemctl enable containerd

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

sudo apt update

sudo apt install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

Execute the following steps only on the Control plane

sudo kubeadm init --control-plane-endpoint=<IP>

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

# Install Work node

Install Ubuntu Server 22.04 TLS

Min 2 CPU cores

Min 2 GB ram

Set static IPs for all servers. Set network adapter to bridged if the server is VM running on Virtualbox.

*sudo vi /etc/netplan/00-installer-config.yaml*

network:

version: 2

renderer: networkd

ethernets:

enp0s3:

addresses:

- 192.168.1.138/24

nameservers:

addresses: [8.8.8.8, 8.8.4.4]

routes:

- to: default

via: 192.168.1.1

*sudo netplan --debug apply*

Change hostname

sudo hostnamectl set-hostname kubernetes2

sudo adduser kubernetes

apt-get install vim

apt-get install net-tools

swapoff -a

Remove following line from /etc/fstab

/swap.img none swap sw 0 0

sudo swapoff -a

sudo sed -i '/ swap / s/^\(.\*\)$/#\1/g' /etc/fstab

sudo tee /etc/modules-load.d/containerd.conf <<EOF

overlay

br\_netfilter

EOF

sudo modprobe overlay

sudo modprobe br\_netfilter

sudo tee /etc/sysctl.d/kubernetes.conf <<EOF

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

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

net.ipv4.ip\_forward = 1

EOF

sudo sysctl --system

sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

sudo apt update

sudo apt install -y containerd.io

containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1

sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml

sudo systemctl restart containerd

sudo systemctl enable containerd

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

sudo apt update

sudo apt install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

Then you can join any number of worker nodes by running the following on each as root:

sudo kubeadm join 192.168.1.133:6443 --token ltwcrr.1uy0u591nnodmktr \

--discovery-token-ca-cert-hash sha256:b307e7a28d2c01fad33c9e8e4c1c9a6bc712cb5df351d10bc833ce63bff034aa

Create a command to join Kubernetis cluster:

kubeadm token create --print-join-command

# Install Network Plugin on control plane

*curl https://projectcalico.docs.tigera.io/manifests/calico.yaml -O*

*kubectl apply -f calico.yaml*

*kubectl get pods -n kube-system*

# Install Helm charts

cd /opt

sudo curl -fsSL -o get\_helm.sh <https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3>

sudo chmod 700 get\_helm.sh

sudo ./get\_helm.sh

# Generate Infrastructure configuration using Terraform scripts

Ref <https://medium.com/@stefanprodan/running-kubernetes-on-scaleway-bare-metal-with-terraform-and-kubeadm-1cf18aae32d5>

Run Terraform script

# Global Kubernetes namespace setup

kubectl create namespace registry-space

# Create Service Account for Terraform Script

# Create Private Docker Images Registry

Source <https://www.linuxtechi.com/setup-private-docker-registry-kubernetes/>

Private registry deployed as Docker image will be used to store and distribute images into the Kubernetes cluster

Execute on every work node:

Generate Self-Signed Certificate

cd /opt

sudo mkdir certs

cd certs

sudo touch registry.key

cd /opt

sudo openssl req -newkey rsa:4096 -nodes -sha256 -keyout \

./certs/registry.key -x509 -days 365 -out ./certs/registry.crt

ls -l certs/

Create registry folder

cd /opt

sudo mkdir registry

sudo chmod -R 777 /opt

Deploy private registry as deployment via private-registry.yaml file

cd /opt/registry

kubectl create -f ingress-controller.yaml

kubectl create -f private-registry.yaml

kubectl create -f service-load-balancer.yaml

kubectl get pods --all-namespaces

kubectl get service private-repository-service -n registry-space

# Create Ingress controller and resource

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.3.0/deploy/static/provider/baremetal/deploy.yaml

kubectl get po -A

kubectl expose deployment private-repository-k8s --type=NodePort --port=5000

# Create SSL Issues repository

Create Internal SSL Issues repository to verify SSL certificates.

# Create Kafka Broker Pod

Deploy Kafka broker

# Create Spring Boot Docker images

# Create Dummy Spring Boot Docker image

kubectl create deployment web --image=gcr.io/google-samples/hello-app:1.0

kubectl expose deployment web --type=NodePort --port=8080

To get service port use

kubectl get service web

GET <http://192.168.1.138:32647>

# Deploy Spring Boot Application

Install on Control Plane docker package

sudo apt-get install docker-ce

dpkg -l|grep docker

Deploy private registry as deployment via spring-boot.yaml file

# Troubleshooting

Login as root

kubectl version

kubectl get nodes

kubectl get pods --all-namespaces

journalctl -f

kubectl describe nodes command

kubectl describe po private-repository-k8s-6cc59cbcb-gxmpt

Delete pod

kubectl get deployments --all-namespaces

kubectl delete -n default deployment private-repository-k8s

kubectl get nodes

kubectl describe node kubernetes1

kubectl describe node kubernetes2

// delete node

kubectl drain kubernetes4

kubectl delete node kubernetes4