Set up a Kubernetes cluster

Set up a Kubernetes cluster

INSTALLING KUBERNETES

INSTALL CONTAINERD

```
# update packages in apt package manager
sudo apt update && apt dist-uppgrade

# install containerd using the apt package manager
# containerd is lightwieght, reliable and fast (CRI native)
sudo apt-get install -y containerd

# create /etc/containerd directory for containerd configuration
sudo mkdir -p /etc/containerd

# Generate the default containerd configuration
# Change the pause container to version 3.10 (pause container holds the linux ns for
Kubernetes namespaces)
# Set `SystemdCgroup` to true to use same cgroup drive as kubelet
containerd config default \
| sed 's/SystemdCgroup = false/SystemdCgroup = true/' \
| sed 's|sandbox_image = ".*"|sandbox_image = "registry.k8s.io/pause:3.10"|' \
| sudo tee /etc/containerd/config.toml > /dev/null

# Restart containerd to apply the configuration changes
sudo systemctl restart containerd

# Kubernetes doesn’t support swap unless explicitly configured under cgroup v2
sudo swapoff -a
```

INSTALL KUBEADM, KUBELET, and KUBECTL

```
# update packages
sudo apt update

# install apt-transport-https ca-certificates curl and gpg packages using
# apt package manager in order to fetch Kubernetes packages from
# external HTTPS repositories
sudo apt-get install -y apt-transport-https ca-certificates curl gpg

# create a secure directory for storing GPG keyring files
# used by APT to verify trusted repositories.
# This is part of a newer, more secure APT repository layout that
# keeps trusted keys isolated from system-wide GPG configurations
sudo mkdir -p -m 755 /etc/apt/keyrings

# download the k8s release gpg key FOR 1.33
sudo curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.33/deb/Release.key | sudo gpg --dearmor -
o /etc/apt/keyrings/kubernetes-apt-keyring.gpg



# Download and convert the Kubernetes APT repository's GPG public key into
# a binary format (`.gpg`) that APT can use to verify the integrity
# and authenticity of Kubernetes packages during installation.
# This overwrites any existing configuration in
# /etc/apt/sources.list.d/kubernetes.list FOR 1.33
# (`tee` without `-a` (append) will **replace** the contents of the file)
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.33/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list

# update packages in apt
sudo apt-get update

apt-cache madison kubelet
apt-cache madison kubectl
apt-cache madison kubeadm



KUBE_VERSION="1.33.2-1.1"
```

```
# install kubelet, kubeadm, and kubectl at version 1.33.2-1.1
sudo apt-get install -y kubelet=$KUBE_VERSION kubeadm=$KUBE_VERSION kubectl=$KUBE_VERSION

# hold these packages at version
sudo apt-mark hold kubelet kubeadm kubectl
```

ENABLE IP FORWARDING

```
# enable IP packet forwarding on the node, which allows the Linux kernel
# to route network traffic between interfaces.
# This is essential in Kubernetes for pod-to-pod communication
# across nodes and for routing traffic through the control plane
# or CNI-managed networks
sudo sysctl -w net.ipv4.ip_forward=1

# uncomment the line in /etc/sysctl.conf enabling IP forwarding after reboot
sudo sed -i '/^#net\.ipv4\.ip_forward=1/s/^#//' /etc/sysctl.conf

# Apply the changes to sysctl.conf
# Any changes made to sysctl configuration files take immediate effect without requiring a
reboot
sudo sysctl -p
```

INITIALIZE THE CLUSTER (ONLY FROM CONTROL PLANE)

```
########################################
# ⚠ WARNING ONLY ON THE CONTROL PLANE #
########################################
# Initialize the cluster specifying containerd as the container runtime, ensuring that the --
cri-socket argument includes the unix:// prefix
# containerd.sock is a Unix domain socket used by containerd
# The Unix socket mechanism is a method for inter-process communication (IPC) on the same
host.
sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --cri-
```

```
socket=unix:///run/containerd/containerd.sock

# HOW TO RESET IF NEEDED
# sudo kubeadm reset --cri-socket=unix:///run/containerd/containerd.sock
# sudo rm -rf /etc/kubernetes /var/lib/etcd

# ONLY ON CONTROL PLANE (also in the output of 'kubeadm init' command)
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

# ONLY FOR FLANNEL: Load `br_netfilter` and enable bridge networking
# ONLY FOR FLANNEL: echo "br_netfilter" | sudo tee /etc/modules-load.d/br_netfilter.conf

# install flannel
# kubectl apply -f https://raw.githubusercontent.com/flannel-
io/flannel/master/Documentation/kube-flannel.yml



# install calcio
kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/manifests/calico.yaml




# Install the Tigera Calico CNI operator and custom resource definitions
# kubectl create -f
https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manifests/tigera-operator.yaml

# Install Calico CNI by creating the necessary custom resource
# kubectl create -f
https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manifests/custom-resources.yaml
```

JOIN THE WORKER NODES TO THE CLUSTER

```
########################################
# ⚠ WARNING: DO NOT USE THIS COMMAND #
########################################
# Get this command from the 'kubeadm init' output instead (above)
sudo kubeadm join x.x.x.x:6443 --token lllrj0.pystabmhlyt2svty --discovery-token-ca-cert-hash
sha256:9d2fd15886eb176466640067f361ed2295de38188b057becf31d3bf5a4fb0b73

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

kubectl -n kube-system get po -w
```

source

https://gitlab.com/chadmcrowell/container-security-course/-/blob/main/0-Introduction-and-Setup/Installing-Kubernetes.md