Setup Kubernetes

Kubernetes Nodes

Di cluster Kubernetes, Anda akan menemukan dua kategori node yang berbeda:

Node Master: Node ini memainkan peran penting dalam mengelola panggilan API kontrol untuk berbagai komponen dalam cluster Kubernetes. Ini termasuk mengawasi pod, pengontrol replikasi, layanan, node, dan banyak lagi.

Node worker: Node worker bertanggung jawab untuk menyediakan lingkungan runtime untuk container. Perlu dicatat bahwa sekelompok pod container dapat diperluas ke beberapa node worker, memastikan alokasi dan pengelolaan sumber daya yang optimal.

Setup Node Master

```
apt update && apt upgrade
```

Untuk meningkatkan kinerja Kubernetes, nonaktifkan swap dan atur parameter kernel penting. Jalankan perintah berikut pada semua node untuk menonaktifkan semua swap:

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

Load modul kernel yang diperlukan di semua node:

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

sudo modprobe overlay
sudo modprobe br_netfilter
```

Konfigurasikan parameter kernel penting untuk Kubernetes menggunakan perintah berikut:

```
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
```

Kemudian, muat ulang perubahannya:

```
sudo sysctl --system
```

Kita akan menggunakan runtime containerd. Instal containerd dan dependensinya dengan perintah berikut:

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

Aktifkan repositori Docker:

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg

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

Perbarui daftar paket dan instal containerd:

```
sudo apt update

sudo apt install -y containerd.io
```

Konfigurasikan containerd untuk mulai menggunakan systemd sebagai cgroup:

```
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
```

Mulai ulang dan aktifkan layanan containerd:

```
sudo systemctl restart containerd
```

```
sudo systemctl enable containerd
```

Paket Kubernetes tidak tersedia di repositori default Ubuntu 22.04. Tambahkan repositori Kubernetes dengan perintah berikut:

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/kubernetes-xenial.gpg

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

Setelah menambahkan repositori, instal komponen penting Kubernetes, termasuk kubectl, kubelet, dan kubeadm, di semua node dengan perintah berikut:

```
sudo apt update
sudo apt install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

Dengan semua prasyarat yang ada, inisialisasi cluster Kubernetes pada node master menggunakan perintah Kubeadm berikut:

```
sudo kubeadm init
```

```
root@master:~# sudo kubeadm init
[init] Using Kubernetes version: v1.28.3
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
W1102 19:06:53.288119   10840 checks.go:835] detected that the sandbox image "registry.k8s.io/pause:3.6" of the container runtime is inconsistent with that used by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local master] and IPs [10.96.0.1 146.190.135.86]
[certs] Generating "apiserver-kubelet-client" certificate and key
```

```
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master]
and IPs [146.190.135.86 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master]
and IPs [146.190.135.86 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in
"/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[wait-control-plane] Waiting for the kubelet to boot up the control plane
as static Pods from directory "/etc/kubernetes/manifests". This can take up
to 4m0s
[apiclient] All control plane components are healthy after 8.002720 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-
config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system
with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node master as control-plane by adding the
labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-
from-external-load-balancers]
[mark-control-plane] Marking the node master as control-plane by adding the
taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: f1h95l.u4nkex9cw8d0g63w
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap,
RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to
get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to
post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller
automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for
all node client certificates in the cluster
```

```
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-
public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a
rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular
user:

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

Alternatively, if you are the root user, you can run:

 export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
 https://kubernetes.io/docs/concepts/cluster-administration/addons/

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

kubeadm join 146.190.135.86:6443 --token f1h95l.u4nkex9cw8d0g63w \
       --discovery-token-ca-cert-hash
sha256:6d15f2a79bdb38d1666af50c85f060b9fadc73f13c932e0e2a9eeef08f51f91a
```

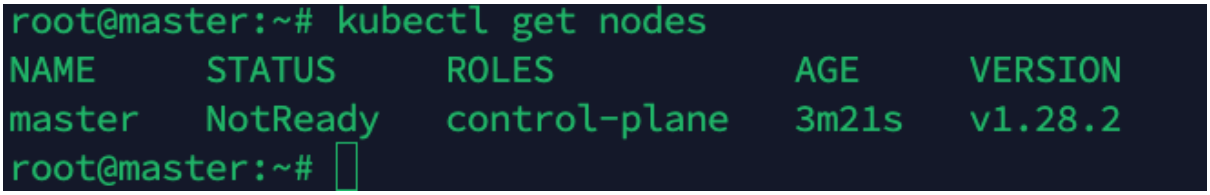
Setelah inisialisasi selesai, catat dan simpan perintah kubeadm join untuk konfigurasi worker node.

Jalankan perintah berikut pada node master:

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

Selanjutnya, gunakan perintah kubectl untuk memeriksa status cluster dan node:

```
kubectl get nodes
```



Instal Plugin Jaringan Kubernetes (node master)

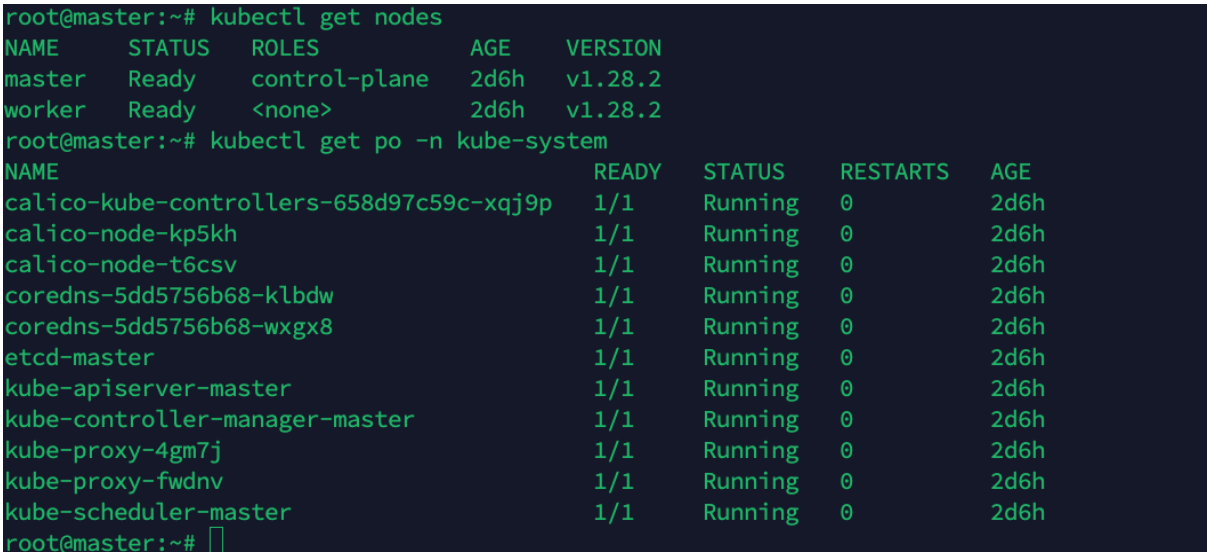
Untuk mengaktifkan komunikasi antar pod di cluster, Anda memerlukan plugin jaringan. Instal plugin jaringan Calico dengan perintah berikut dari node master:

```
kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manif
ests/calico.yaml
```

Verifikasi cluster dan uji (node master)

Terakhir, kita coba memverifikasi apakah cluster kami berhasil dibuat.

```
kubectl get pods -n kube-system
kubectl get nodes
```



Deploy aplikasi pengujian pada cluster (node master)

```
root@master:~# kubectl get po
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          46s
root@master:~#
```

```
kubectl run nginx --image=nginx
```

Setup Node Worker

```
apt update && apt upgrade
```

Untuk meningkatkan kinerja Kubernetes, nonaktifkan swap dan atur parameter kernel penting. Jalankan perintah berikut pada semua node untuk menonaktifkan semua swap:

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

Load modul kernel yang diperlukan di semua node:

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

sudo modprobe overlay
sudo modprobe br_netfilter
```

Konfigurasikan parameter kernel penting untuk Kubernetes menggunakan perintah berikut:

```
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
```

Kemudian, muat ulang perubahannya:

```
sudo sysctl --system
```

Kita akan menggunakan runtime containerd. Instal containerd dan dependensinya dengan perintah berikut:

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

Aktifkan repositori Docker:

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg
```

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

Perbarui daftar paket dan instal containerd:

```
sudo apt update

sudo apt install -y containerd.io
```

Konfigurasikan containerd untuk mulai menggunakan systemd sebagai cgroup:

```
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
```

Mulai ulang dan aktifkan layanan containerd:

```
sudo systemctl restart containerd

sudo systemctl enable containerd
```

Paket Kubernetes tidak tersedia di repositori default Ubuntu 22.04. Tambahkan repositori Kubernetes dengan perintah berikut:

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo
gpg --dearmour -o /etc/apt/trusted.gpg.d/kubernetes-xenial.gpg

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

Setelah menambahkan repositori, instal komponen penting Kubernetes, termasuk kubectl, kubelet, dan kubeadm, di semua node dengan perintah berikut:

```
sudo apt update
sudo apt install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

Pada setiap node worker, gunakan perintah kubeadm join yang Anda catat sebelumnya:

```
kubeadm join 146.190.135.86:6443 --token f1h95l.u4nkex9cw8d0g63w         --
discovery-token-ca-cert-hash
sha256:6d15f2a79bdb38d1666af50c85f060b9fadc73f13c932e0e2a9eeef08f51f91a
```