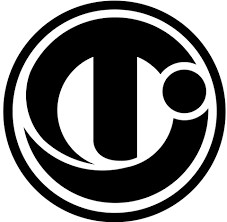
LAPORAN

PEMROGRAMAN MICROSERVICE

Install 1 Kubernetes dan 2 Docker pada Sistem Operasi Linux



DISUSUN OLEH

RASIHAN YUSRI MAZID

2301081011

PROGRAM STUDI TEKNIK KOMPUTER

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI PADANG

2025

**Persiapan AWS EC2 Instances**

# Langkah 1: Membuat 3 EC2 Instances

1. Login ke AWS Management Console
2. Navigasi ke layanan EC2
3. Klik “Launch Instances” 4. Beri nama untuk instans:
   * Kubernetes-master
   * Worker-node-1
   * Worker-node-2

5. Pilih AMI: Ubuntu Server 24.04 LTS (HVM), SSD Volume Type 6. Pilih tipe instans:

* Master node: minimal t2.medium (2 vCPU, 4 GB RAM)
* Worker nodes: minimal t2.micro (1 vCPU, 1 GB RAM)

1. Konfigurasi key pair untuk SSH
2. Pada konfigurasi jaringan, buat security group dengan port:
   * SSH (22)
   * Kubernetes API (6443)
   * NodePort range (30000-32767)
   * Izinkan semua traffic antar node dalam group
3. Luncurkan instance

# Konfigurasi Dasar pada Semua Node

Hubungkan ke semua node (master dan workers) melalui SSH dan jalankan perintah berikut pada masing-masing server: # Update system packages sudo apt update sudo apt upgrade -y

# Install prerequisites sudo apt install -y apt-transport-https ca-certificates curl software-properties-common

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

# Konfigurasi modul kernel cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf overlay br\_netfilter

EOF

sudo modprobe overlay sudo modprobe br\_netfilter

# Konfigurasi sysctl untuk Kubernetes cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf net.bridge.bridge-nf-call-iptables = 1 net.bridge.bridge-nf-call-ip6tables = 1 net.ipv4.ip\_forward = 1

EOF

# Apply sysctl settings sudo sysctl --system

# Install Docker

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o

/etc/apt/keyrings/docker.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list

sudo apt update sudo apt install -y docker-ce docker-ce-cli containerd.io

# Konfigurasi Docker untuk menggunakan systemd sebagai cgroup driver

sudo mkdir -p /etc/docker cat <<EOF | sudo tee /etc/docker/daemon.json

{

"exec-opts": ["native.cgroupdriver=systemd"],

"log-driver": "json-file",

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2"

}

EOF

sudo systemctl enable docker sudo systemctl daemon-reload sudo systemctl restart docker

# Tambahkan user ke group docker sudo usermod -aG docker $USER

# Konfigurasi containerd sudo mkdir -p /etc/containerd

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

# Restart containerd sudo systemctl restart containerd

# Install tools Kubernetes

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o

/etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

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

# Atur endpoint crictl sudo crictl config --set runtime-endpoint=unix:///run/containerd/containerd.sock

# Restart kubelet sudo systemctl restart kubelet

Konfigurasi Node Master Kubernetes

Pada node master (kubernetes-master), jalankan:

# Inisialisasi cluster Kubernetes

sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --control-plane-endpoint=$(curl -s http://169.254.169.254/latest/meta-data/public-ipv4)

# Setelah inisialisasi berhasil, setup konfigurasi untuk user saat ini mkdir -p $HOME/.kube sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config

# Deploy Flannel CNI network plugin

kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kubeflannel.yml

# Simpan join command untuk node worker kubeadm token create --print-join-command > join-command.txt **Join Worker Nodes ke Cluster**

1. Pada node master, tampilkan perintah untuk join worker: Cat join-command.txt
2. Salin output perintah tersebut
3. Pada setiap worker node, jalankan perintah yang disalin dengan menambahkan sudo di depannya: sudo kubeadm join 172.31.xx.xx:6443 --token xxxxxx.xxxxxxxxxxxxxxxx --discoverytoken-ca-cert-hash sha256:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
4. Jika terjadi error “container runtime is not running", pastikan containerd dan kubelet sudah dikonfigurasi dan berjalan dengan benar:

# Verifikasi status containerd

sudo systemctl status containerd

# Jika perlu, restart containerd

sudo systemctl restart containerd

# Verifikasi status kubelet

sudo systemctl status kubelet

# Jika perlu, restart kubelet

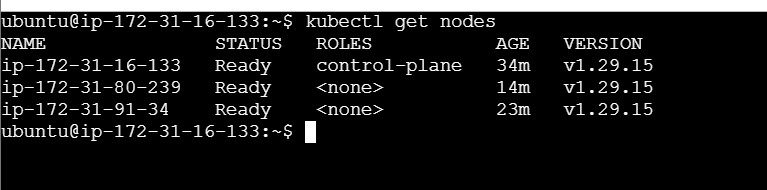
sudo systemctl restart kubelet

# Coba join lagi dengan perintah sebelumnya

# Verifikasi Cluster

Pada node master, periksa status cluster: kubectl get nodes

Output seharusnya menampilkan 3 node dengan status Ready: NAME STATUS ROLES AGE VERSION ip-172-31-16-133 Ready control-plane 20m v1.29.15 ip-172-31-80-239 Ready <none> 8s v1.29.15 ip-172-31-91-34 Ready <none> 8m18s v1.29.15



# Deploy 2 Container Docker

Setelah semua node status Ready, deploy container-container Docker pada cluster. Jalankan perintah berikut pada master node:

**Container 1: NGINX Web Server** # Buat file nginx-deployment.yaml cat <<EOF > nginx-deployment.yaml apiVersion: apps/v1 kind: Deployment metadata: name: nginx-deployment spec:

replicas: 1 selector: matchLabels: app: nginx template: metadata: labels:

app: nginx spec: containers:

* name: nginx

image: nginx:latest

ports:

* containerPort: 80

---

apiVersion: v1 kind: Service metadata:

name: nginx-service spec:

type: NodePort selector: app: nginx ports: - port: 80 targetPort: 80 nodePort: 30080

EOF

# Deploy NGINX kubectl apply -f nginx-deployment.yaml **Container 2: Redis Cache**

# Buat file redis-deployment.yaml cat <<EOF > redis-deployment.yaml apiVersion: apps/v1 kind: Deployment metadata:

name: redis-deployment spec: replicas: 1 selector:

matchLabels: app: redis template: metadata: labels: app: redis spec: containers: - name: redis image: redis:latest ports:

- containerPort: 6379

---

apiVersion: v1 kind: Service metadata:

name: redis-service spec: selector: app: redis ports: - port: 6379 targetPort: 6379

EOF

# Deploy Redis kubectl apply -f redis-deployment.yaml **Verifikasi Deployment**

# Periksa status deployment

kubectl get deployments

# Periksa status pods kubectl get pods

# Periksa status services kubectl get services

# Lihat di node mana pods berjalan kubectl get pods -o wide

# Akses Aplikasi

Untuk mengakses aplikasi NGINX, gunakan NodePort yang telah dikonfigurasi (30080):

# Menggunakan IP publik salah satu node http://<public-ip>:30080

Pastikan port 30080 sudah dibuka di security group AWS.

# Setup Monitoring Basic (Opsional)

# Deploy Metrics Server

kubectl apply -f https://github.com/kubernetes-sigs/metricsserver/releases/latest/download/components.yaml

# Tunggu beberapa menit sleep 180

# Periksa penggunaan sumber daya node kubectl top nodes

# Periksa penggunaan sumber daya pod kubectl top pods

**Troubleshooting Umum**

**Error 1: CRI runtime not running**

Jika mendapat error: [ERROR CRI]: container runtime is not running

# Reset konfigurasi containerd sudo rm -rf /etc/containerd/config.toml sudo systemctl restart containerd

# Buat konfigurasi baru dengan SystemdCgroup = true sudo mkdir -p /etc/containerd

cat <<EOF | sudo tee /etc/containerd/config.toml version = 2

[plugins]

[plugins."io.containerd.grpc.v1.cri"] sandbox\_image = "registry.k8s.io/pause:3.9"

[plugins."io.containerd.grpc.v1.cri".containerd]

[plugins."io.containerd.grpc.v1.cri".containerd.runtimes] [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc] runtime\_type = "io.containerd.runc.v2"

[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]

SystemdCgroup = true

EOF

# Restart layanan sudo systemctl restart containerd sudo systemctl restart kubelet

**Error 2: Tidak dapat join cluster setelah token kedaluwarsa** Jika token join kedaluwarsa (24 jam), buat token baru pada master node: sudo kubeadm token create --print-join-command

Kemudian gunakan perintah baru tersebut pada worker node.

# Error 3: DNS tidak berfungsi dalam cluster

# Verifikasi status CoreDNS pods kubectl get pods -n kube-system

# Jika pods tidak running, restart kubectl -n kube-system delete pods -l k8s-app=kube-dns

# Penutup

Selamat! Anda telah berhasil menginstall dan mengkonfigurasi cluster Kubernetes dengan 1 node master dan 2 worker node pada AWS menggunakan Ubuntu Server 24.04 LTS. Anda juga telah men-deploy 2 container Docker (NGINX dan Redis) pada cluster tersebut.

Kubernetes cluster ini sekarang siap digunakan untuk aplikasi produksi atau pengembangan lebih lanjut.