LAPORAN

PEMROGRAMAN MICROSERVICE

Install 1 Kubernetes dan 2 Docker pada Sistem Operasi Linux

Disusun Oleh:

Nama	**: Hafizh Fadhlurrohman**
NIM	**: 2301081006**
Kelas	**: 2-A**
Dosen	**: Ervan Asri, S.Kom., M.Kom**

PROGRAM STUDI TEKNIK KOMPUTER

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI PADANG

2025

Tutorial Install 1 Kubernetes dan 2 Docker Pada Sistem Operasi Linux

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

Konfigurasi Dasar pada Semua Node

Hubungkan ke semua node (master dan workers) melalui SSH dan jalankan perintah berikut pada masing-masing server:

```
sudo apt update

sudo apt upgrade -y

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

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

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

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

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

sudo systemctl restart containerd


# Install Kubernetes tools

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
sudo systemctl restart kubelet
```

Konfigurasi Node Master 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)

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

kubectl apply -f https://github.com/flannel-
io/flannel/releases/latest/download/kubeflannel.yml
kubeadm token create --print-join-command > join-command.txt
```

Join Worker Nodes ke Cluster

```
# Di master node:
cat join-command.txt
```

```
# Di setiap worker node:
sudo kubeadm join <ip-master>:6443 --token <token> --discovery-token-ca-cert-hash
sha256:<hash>
```

Verifikasi Cluster

```
kubectl get nodes
```

Deploy 2 Container Docker

NGINX Web Server

```
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

kubectl apply -f nginx-deployment.yaml
```

Redis Cache

```
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

kubectl apply -f redis-deployment.yaml
```

Akses Aplikasi

```
http://<public-ip-node>:30080
```