Module 2a: Kubernetes Cluster Installation Guide

This comprehensive lab guide walks system administrators through installing a production-ready Kubernetes 1.33.1 cluster on Ubuntu 24.04 LTS. The installation covers all essential components including containerd runtime, CNI networking with Cilium, and proper system configuration. By following these steps, you'll have a fully functional single-node cluster ready for development or testing workloads.

Author: Kelvin Lai

System Requirements: 2 vCPU, 4GB RAM, Ubuntu 24.04.03 LTS Desktop

Initial System Configuration

Switch to Root

Begin by elevating privileges to root for system-level configurations.

```
sudo -i
```

Set Hostname

Configure the master node hostname for cluster identification and DNS resolution.

```
hostnamectl hostname master
cat /etc/hosts
echo "192.168.0.8 master" >> /etc/hosts
```

01

Elevate Privileges

Switch to root user for administrative access

02

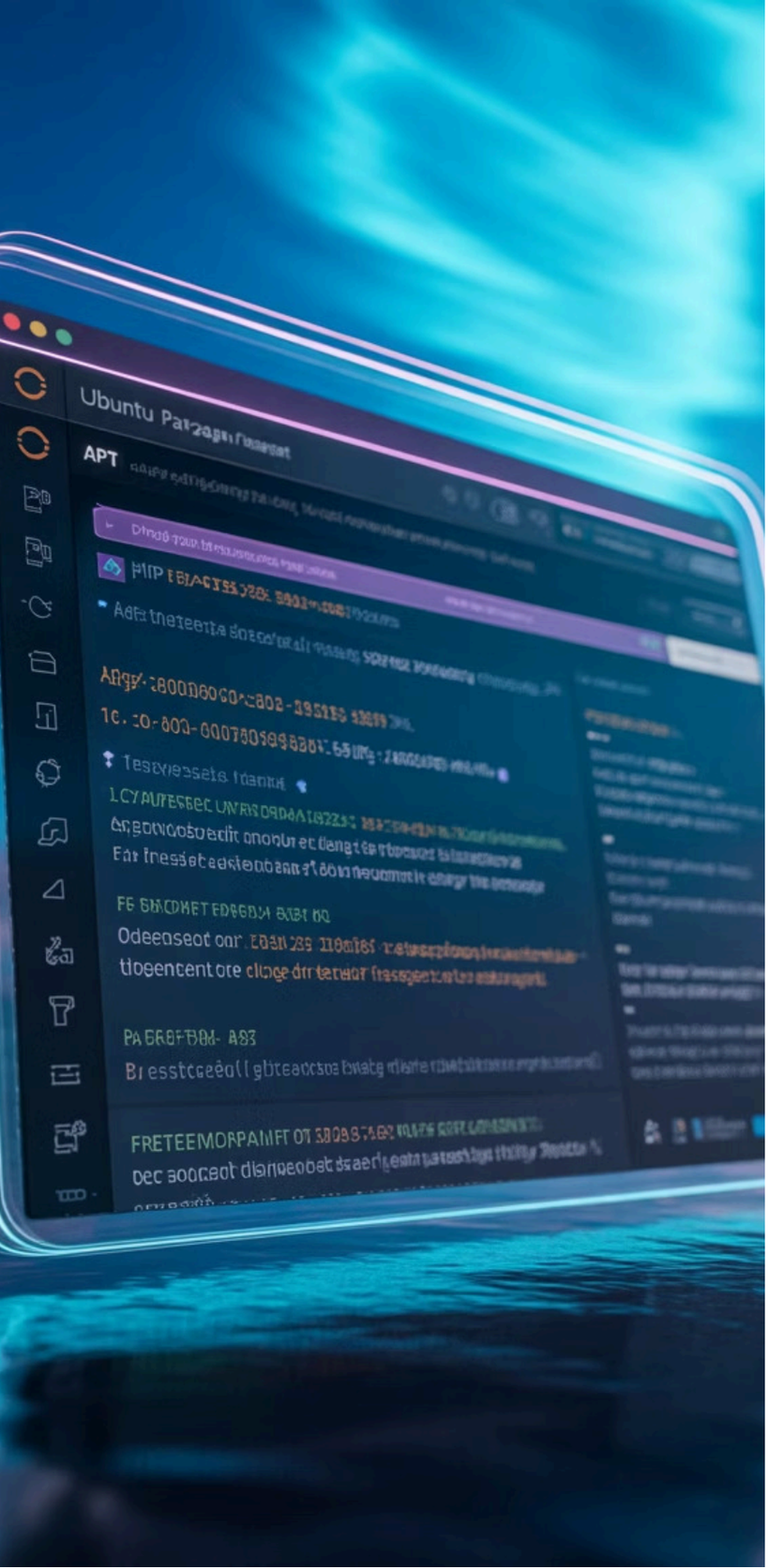
Configure Hostname

Set master node identifier for cluster networking

03

Update Host Resolution

Ensure proper DNS and hostname mapping



Package Management & Dependencies

Update the system package index and upgrade existing packages to ensure security patches and compatibility. Then install prerequisite packages required for Kubernetes container runtime and networking components.

System Update

```
apt update && apt upgrade -y
```

Refreshes package lists and upgrades all installed packages to latest versions

Install Prerequisites

```
apt install -y apt-transport-https \
software-properties-common ca-certificates \
socat vim curl gnupg2 lsb-release wget \
bash-completion tree
```

Installs essential tools for HTTPS transport, certificate management, and container operations

Kernel & System Optimization

Disable Swap

Kubernetes requires swap to be disabled for proper memory management and pod scheduling.

```
swapoff -a
sed -i '/^[^#].* swap .*/s/^/#/' /etc/fstab
```

Load Kernel Modules

Enable overlay and br_netfilter modules for container networking capabilities.

```
cat << EOF | tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
modprobe overlay
modprobe br_netfilter
```

Configure Kernel Parameters

Set networking parameters for iptables bridge processing and IP forwarding.

```
cat << EOF | tee /etc/sysctl.d/kubernetes.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
sysctl --system
```

Repository Configuration

Add GPG keys and repositories for Kubernetes and containerd packages. This ensures package authenticity and enables installation from official sources.

Add GPG Keys

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/$K8S_GPG_VER/deb/Release.key | \
sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Add Package Repositories

```
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] \
https://pkgs.k8s.io/core:/stable:/$K8S_GPG_VER/deb/ /"

echo "deb [arch=$(dpkg --print-architecture) \
signed-by=/etc/apt/keyrings/docker.gpg] \
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

Container Runtime Installation

Install and configure containerd as the container runtime. Enable systemd cgroup driver for proper resource management and update the pause container image version for compatibility with Kubernetes 1.33.

1 Install Containerd

```
apt update && apt install containerd.io -y
containerd config default | tee /etc/containerd/config.toml
```

2 Configure Runtime

```
sed -e 's/SystemdCgroup = false/SystemdCgroup = true/g' \
-i /etc/containerd/config.toml
sed -e 's/pause:3.8/pause:3.10/' \
-i /etc/containerd/config.toml
```

3 Enable Service

```
systemctl restart containerd
systemctl enable containerd
```

Kubernetes Components Installation

Install Core Components

Install kubeadm, kubectl, and kubelet at version 1.33.1. These components form the foundation of your Kubernetes cluster. Hold the packages to prevent unintended upgrades.

```
apt update
apt install -y kubeadm=1.33.1-* \
kubectl=1.33.1-* kubelet=1.33.1-*

apt-mark hold kubeadm kubectl kubelet
```

Configure Container Runtime

Set the crictl runtime and image endpoints to use containerd socket.

```
crictl config \
--set runtime-endpoint=unix:///run/containerd/containerd.sock
\
--set image-endpoint=unix:///run/containerd/containerd.sock
```


kubeadm

Cluster bootstrap and lifecycle management tool

kubectl

Command-line interface for cluster operations

kubelet

Node agent managing pod lifecycle

Cluster Initialization & Configuration

Initialize the Kubernetes control plane with specific networking parameters. Configure kubectl access for the student user and enable bash completion for enhanced command-line productivity.

Initialize Cluster

```
kubeadm init --kubernetes-version=1.33.1 \
--pod-network-cidr=10.0.0.0/16 \
--upload-certs --node-name=master \
--control-plane-endpoint=master:6443
```

Configure kubectl Access

```
mkdir -p /home/student/.kube
cp /etc/kubernetes/admin.conf /home/student/.kube/config
chown -R student:student /home/student/.kube
```

Enable Bash Completion

```
mkdir -p /etc/bash_completion.d
kubectl completion bash > /etc/bash_completion.d/kubectl
```

Important: The pod network CIDR 10.0.0.0/16 must not overlap with your node network. This network will be used for inter-pod communication.

Network Plugin Deployment

Install Cilium CNI

Deploy Cilium 1.16.1 as the Container Network Interface using Helm. Cilium provides advanced networking, security, and observability features powered by eBPF technology.

```
snap install helm --classic

helm repo add cilium https://helm.cilium.io/
helm repo update

helm template cilium cilium/cilium \
--version 1.16.1 --namespace kube-system >
cilium.yaml

KUBECONFIG=/etc/kubernetes/admin.conf \
kubectl apply -f cilium.yaml
```

Layer 3/4 Networking

Pod-to-pod communication across nodes

Network Policies

Fine-grained traffic control and security

Load Balancing

Efficient service traffic distribution

Verification & Worker Node Addition

Verify Cluster Status

Check node status and remove control-plane taint to allow workload scheduling on the master node in single-node configurations.

```
kubectl get node
kubectl describe node master
kubectl describe node | grep -i taint

kubectl taint node --all node-role.kubernetes.io/control-plane-
ip a
```

Step 1: Prepare Worker Node

Execute Steps 1-9 on the worker node to install prerequisites, container runtime, and Kubernetes components

Step 2: Configure DNS Resolution

Ensure master and worker hostnames are resolvable. Add entries to /etc/hosts if DNS is unavailable

Step 3: Generate Join Token

On master node, run: `kubeadm token create --print-join-command`

Step 4: Join the Cluster

Copy and execute the kubeadm join command output from Step 3 on the worker node

Success! Your Kubernetes cluster is now operational. Use `kubectl get nodes` to verify all nodes are in Ready status.