Kubernetes Cluster Setup Guide

Table of Contents

---

1. Introduction

Overview

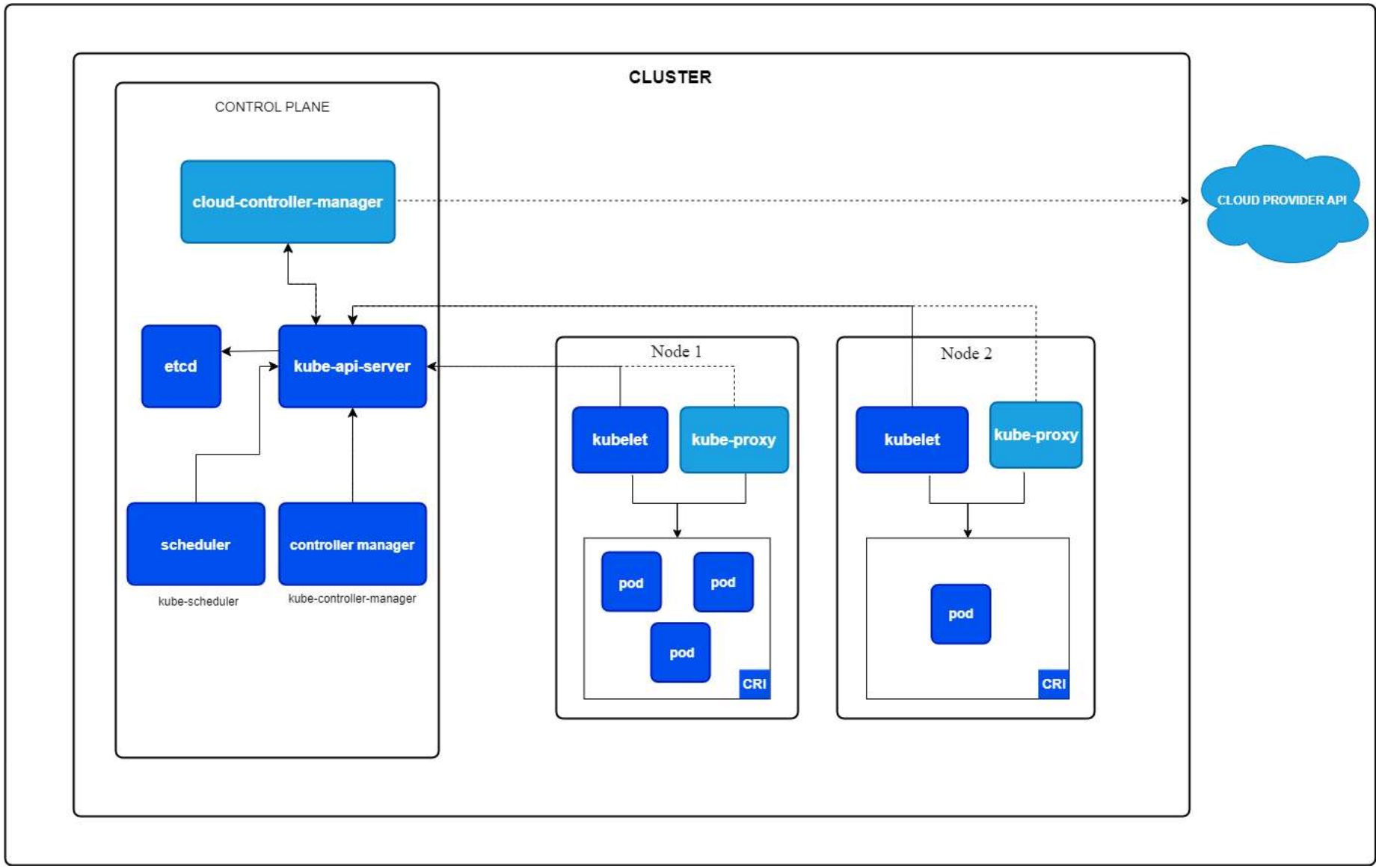
Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. It allows organizations to run applications efficiently across multiple nodes, ensuring high availability, scalability, and fault tolerance. Kubernetes is widely used in cloud environments and supports various networking, storage, and security configurations.

Setting up a Kubernetes cluster provides a centralized platform to manage workloads, making it easier to deploy and maintain applications. By configuring a cluster with a master node and worker nodes, users can take advantage of Kubernetes features like self-healing, load balancing, rolling updates, and automated scaling.

Architecture

I have used below Architecture for the setup:

- **Number of nodes** (1 Master, 2 Worker Nodes)
- **Kubernetes version used**
    - Kubernetes v1.32
- **Network Plugin**
    - **Flannel -** an overlay network provider that can be used with Kubernetes.


CLUSTER
CONTROL PLANE
cloud-controller-manager
CLOUD PROVIDER API
etcd
kube-api-server
Node 1
Node 2
kubelet
kube-proxy
kubelet
kube-proxy
scheduler
controller manager
kube-scheduler
kube-controller-manager
pod
pod
pod
pod
CRI
CRI

2. System Requirements & Prerequisites

List system requirements:

- OS Version - Ubuntu 22.04.2 LTS
- Hardware Specifications (CPU, RAM, Storage)
    - CPU: Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz
    - RAM: 4 GB
    - Storage: 30G
- Network Configuration
    - Single Adaptor configured with 192.168.95.14 ip

Prerequisite Configurations

- Update System Packages (perform on all machines)
-

```
tausif@controlnode:~$ sudo apt-get update
Hit:1 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.32/deb  InRelease [1,186
B]
Hit:5 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.32/deb  Packages [3,954 B
]
Fetched 5,140 B in 1s (3,892 B/s)
Reading package lists... Done
```

- Disable Swap (perform on all machines)
    - The default behavior of a kubelet is to fail to start if swap memory is detected on a node. This means that swap should either be disabled or tolerated by kubelet.
- ```
  tausif@controlnode:~$ sudo swapoff -a
  tausif@controlnode:~$ free -h
                 total        used        free      shared  buff/cache   available
  Mem:           3.8Gi       666Mi       1.3Gi        13Mi       1.8Gi       2.9Gi
  Swap:             0B          0B          0B
  ```
- Enable IP Forwarding (perform on all machines)
    - Kubernetes networking required IP forwarding to be enabled in kernel. Perform below steps to enable it in kernel.
  ```

```
tausif@controlnode:~$ sudo vim /proc/sys/net/ipv4/ip_forward
tausif@controlnode:~$ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
tausif@controlnode:~$ sudo vim /etc/sysctl.conf
tausif@controlnode:~$ sudo sysctl -p
net.ipv4.ip_forward = 1
tausif@controlnode:~$ cat /proc/sys/net/ipv4/ip_forward
1
```

---

3. Installing Required Packages

Download the public signing key for the Kubernetes package Repositories. (Perform on all Machines)

```
tausif@controlnode:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.32/deb/Release.key | sudo gpg --dearmor -
o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
File '/etc/apt/keyrings/kubernetes-apt-keyring.gpg' exists. Overwrite? (y/N) y
```

Add the appropriate Kubernetes apt repository (Perform on all Machines)

```
tausif@controlnode:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/co
re:/stable:/v1.32/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.32/deb/ /
```

Install Kubernetes Components

sudo apt install -y kubeadm kubelet kubectl

```
tausif@controlnode:~$ sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 179 not upgraded.
Need to get 92.7 MB of archives.
After this operation, 338 MB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 conntrack amd64 1:1.4.6-2build2 [33.5 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.32/deb  cri-tools 1.32.0-
1.1 [16.3 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.32/deb  kubeadm 1.32.1-1.
1 [12.2 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.32/deb  kubectl 1.32.1-1.
1 [11.3 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.32/deb  kubernetes-cni 1.
6.0-1.1 [37.8 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.32/deb  kubelet 1.32.1-1.
1 [15.2 MB]
Fetched 92.7 MB in 2s (46.2 MB/s)
Selecting previously unselected package conntrack.
(Reading database ... 204860 files and directories currently installed.)
Preparing to unpack .../0-conntrack 1%3a1.4.6-2build2 amd64.deb ...
```

```
Unpacking cri-tools (1.32.0-1.1) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../2-kubeadm_1.32.1-1.1_amd64.deb ...
Unpacking kubeadm (1.32.1-1.1) ...
Selecting previously unselected package kubectl.
Preparing to unpack .../3-kubectl_1.32.1-1.1_amd64.deb ...
Unpacking kubectl (1.32.1-1.1) ...
Selecting previously unselected package kubernetes-cni.
Preparing to unpack .../4-kubernetes-cni_1.6.0-1.1_amd64.deb ...
Unpacking kubernetes-cni (1.6.0-1.1) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../5-kubelet_1.32.1-1.1_amd64.deb ...
Unpacking kubelet (1.32.1-1.1) ...
Setting up conntrack (1:1.4.6-2build2) ...
Setting up kubectl (1.32.1-1.1) ...
Setting up cri-tools (1.32.0-1.1) ...
Setting up kubernetes-cni (1.6.0-1.1) ...
Setting up kubeadm (1.32.1-1.1) ...
Setting up kubelet (1.32.1-1.1) ...
Processing triggers for man-db (2.10.2-1) ...
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
```

Verify the Kubeadm installation

```
tausif@controlnode:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"32", GitVersion:"v1.32.1", GitCommit:"e9c9be4007d1664e68796af0
2b8978640d2c1b26", GitTreeState:"clean", BuildDate:"2025-01-15T14:39:14Z", GoVersion:"go1.23.4", Compiler:"gc",
 Platform:"linux/amd64"}
```

Install Docker/Container Runtime

sudo apt update && sudo apt install -y containerd

```
tausif@controlnode:~$ sudo apt update
Hit:1 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.32/deb  InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
179 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

```
tausif@controlnode:~$ sudo apt install containerd -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  runc
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 0 to remove and 179 not upgraded.
Need to get 45.7 MB of archives.
After this operation, 176 MB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.12-0ubuntu2~22.04.1 [8,405 kB
]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.24-0ubuntu1~22.04.1 [37
.3 MB]
Fetched 45.7 MB in 5s (8,786 kB/s)
Selecting previously unselected package runc.
(Reading database ... 204919 files and directories currently installed.)
Preparing to unpack .../runc_1.1.12-0ubuntu2~22.04.1_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu2~22.04.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../containerd_1.7.24-0ubuntu1~22.04.1_amd64.deb ...
Unpacking containerd (1.7.24-0ubuntu1~22.04.1) ...
Setting up runc (1.1.12-0ubuntu2~22.04.1) ...
```

Configuring a cgroup driver

To check the current cgroup driver:
kubelet --version

To change the driver if needed, update the config:
sudo sed -i 's/systemd/cgroupfs/g' /etc/containerd/config.toml
sudo systemctl restart containerd kubelet

Verify that both Container run time and Kubelet have a same cgroup drive. It will be either “systemd” or “cgroup”. By default it will be system

```
tausif@controlnode:~$ ps -p 1
    PID TTY          TIME CMD
      1 ?        00:00:03 systemd
```

Configuring containerd runtime to use systemd as a cgroup driver

Create a directory containerd inside /etc

```
tausif@controlnode:~$ sudo mkdir /etc/containerd
```

Verify the value of “SystemdCgroup = true” under [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options] using containerd config default cmd (by default value will not be present)

```
tausif@controlnode:~$ containerd config default
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
```

Create a file called config.toml inside /etc/containerd and pass the argument to add value as **“SystemdCgroup = true” under [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]**

```
tausif@controlnode:~$ containerd config default | sed 's/SystemdCgroup = false/SystemdCgroup = true/' | sudo te
e /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0
```

Once execute verify the value:

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
  BinaryName = ""
  CriuImagePath = ""
  CriuPath = ""
  CriuWorkPath = ""
  IoGid = 0
  IoUid = 0
  NoNewKeyring = false
  NoPivotRoot = false
  Root = ""
  ShimCgroup = ""
  SystemdCgroup = true
```

```
tausif@controlnode:~$ cat /etc/containerd/config.toml | grep -i SystemdCgroup -B 50
      ignore_blockio_not_enabled_errors = false
      ignore_rdt_not_enabled_errors = false
      no_pivot = false
      snapshotter = "overlayfs"

      [plugins."io.containerd.grpc.v1.cri".containerd.default_runtime]
        base_runtime_spec = ""
        cni_conf_dir = ""
        cni_max_conf_num = 0
        container_annotations = []
        pod_annotations = []
        privileged_without_host_devices = false
        privileged_without_host_devices_all_devices_allowed = false
        runtime_engine = ""
        runtime_path = ""
        runtime_root = ""
        runtime_type = ""
        sandbox_mode = ""
        snapshotter = ""

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

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

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
  BinaryName = ""
  CriuImagePath = ""
  CriuPath = ""
  CriuWorkPath = ""
  IoGid = 0
  IoUid = 0
  NoNewKeyring = false
  NoPivotRoot = false
  Root = ""
  ShimCgroup = ""
  SystemdCgroup = true
```

Restart the containerd service to apply the changes

```
tausif@controlnode:~$ sudo systemctl restart containerd
```

---

4. Initializing the Control Plane

Run Kubeadm Init Command

sudo kubeadm init --apiserver-advertise-address=<Master-IP> --pod-network-cidr=10.244.0.0/16

```
tausif@controlnode:~$ sudo kubeadm init --apiserver-advertise-address 192.168.95.14 --pod-network-cidr "10.244.
0.0/16" --upload-certs
[init] Using Kubernetes version: v1.32.1
[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 beforehand using 'kubeadm config images pull'
W0208 18:48:36.591319    5666 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the
 container runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.
10" 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 [controlnode kubernetes kubernetes.default kubernetes.de
fault.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.95.14]
[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 [controlnode localhost] and IPs [192.168.95.14 127.0.0
.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [controlnode localhost] and IPs [192.168.95.14 127.0.0.1
```

```
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 192.168.95.14:6443 --token 5x44wj.fmkrkxb9j6s0pvog \
        --discovery-token-ca-cert-hash sha256:6c6d887174c1d5ffd4a5d0fada34095bf77bc8739ab2248721ccd080f31df2cb
```

Configure kubectl for Admin User

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

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

Verify the installation:

```
tausif@controlnode:~$ kubectl get pods -A
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
kube-system   coredns-668d6bf9bc-5r8cl              0/1     Pending   0          3m2s
kube-system   coredns-668d6bf9bc-l7w48              0/1     Pending   0          3m2s
kube-system   etcd-controlnode                      1/1     Running   0          3m6s
kube-system   kube-apiserver-controlnode            1/1     Running   0          3m6s
kube-system   kube-controller-manager-controlnode   1/1     Running   0          3m6s
kube-system   kube-proxy-s56zc                      1/1     Running   0          3m2s
kube-system   kube-scheduler-controlnode            1/1     Running   0          3m6s
tausif@controlnode:~$ kubectl get nodes
NAME          STATUS     ROLES           AGE     VERSION
controlnode   NotReady   control-plane   3m56s   v1.32.1
```

NOTE: controlnode status will be “NotReady” till the time you configured Network based pod. **You must deploy a Container Network Interface (CNI) based Pod network add-on so that your Pods can communicate with each other. Cluster DNS (CoreDNS) will not start up before a network is installed.**

---

5. Deploying the Pod Network

Install Flannel Network Plugin

Flannel is chosen as the CNI plugin because it provides a simple, reliable overlay network using VXLAN tunnels. It helps Kubernetes pods communicate across nodes by encapsulating network traffic.

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

```
tausif@controlnode:~$ wget https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
--2025-02-08 19:21:24--  https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
Resolving github.com (github.com)... 20.207.73.82
Connecting to github.com (github.com)|20.207.73.82|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github.com/flannel-io/flannel/releases/download/v0.26.4/kube-flannel.yml [following]
--2025-02-08 19:21:24--  https://github.com/flannel-io/flannel/releases/download/v0.26.4/kube-flannel.yml
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/21704134/9ececd7d-10a3-4
ebd-96c3-1f0298db33b3?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250208%2Fus-
east-1%2Fs3%2Faws4_request&X-Amz-Date=20250208T135124Z&X-Amz-Expires=300&X-Amz-Signature=80dde06c9a72467c3e4c19
4493bb88c9a6d97f27bf963880647fe58dd66d8104&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%
20filename%3Dkube-flannel.yml&response-content-type=application%2Foctet-stream [following]
--2025-02-08 19:21:24--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/21704134/
9ececd7d-10a3-4ebd-96c3-1f0298db33b3?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2
F20250208%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250208T135124Z&X-Amz-Expires=300&X-Amz-Signature=80dde06
c9a72467c3e4c194493bb88c9a6d97f27bf963880647fe58dd66d8104&X-Amz-SignedHeaders=host&response-content-disposition
=attachment%3B%20filename%3Dkube-flannel.yml&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.110.133, 18
5.199.111.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4409 (4.3K) [application/octet-stream]
```

```
tausif@controlnode:~$ kubectl apply -f kube-flannel.yml
namespace/kube-flannel created
serviceaccount/flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
tausif@controlnode:~$ kubectl get pods -A
NAMESPACE      NAME                                READY   STATUS             RESTARTS      AGE
kube-flannel   kube-flannel-ds-j4bl5               0/1     CrashLoopBackOff   1 (13s ago)   27s
kube-system    coredns-668d6bf9bc-5r8cl            0/1     ContainerCreating  0             33m
kube-system    coredns-668d6bf9bc-l7w48            0/1     ContainerCreating  0             33m
kube-system    etcd-controlnode                    1/1     Running            0             33m
kube-system    kube-apiserver-controlnode          1/1     Running            0             33m
kube-system    kube-controller-manager-controlnode 1/1     Running            0             33m
kube-system    kube-proxy-s56zc                    1/1     Running            0             33m
kube-system    kube-scheduler-controlnode          1/1     Running            0             33m
```

NOTE: when I tried to create flanner network pod, it was failed to start because “br_netfilter” kernel module was missing and “bridge-nf-call-iptables” configuration was also missing. You can see below steps of troubleshooting.

- I check the logs why pod was not starting

```
tausif@controlnode:~$ kubectl logs pods -n kube-flannel kube-flannel-ds-j4bl5
error: error from server (NotFound): pods "pods" not found in namespace "kube-flannel"
tausif@controlnode:~$ kubectl logs -n kube-flannel kube-flannel-ds-j4bl5
Defaulted container "kube-flannel" out of: kube-flannel, install-cni-plugin (init), install-cni (init)
I0208 13:54:04.146867       1 main.go:211] CLI flags config: {etcdEndpoints:http://127.0.0.1:4001,http://127.0.
0.1:2379 etcdPrefix:/coreos.com/network etcdKeyfile: etcdCertfile: etcdCAFile: etcdUsername: etcdPassword: vers
ion:false kubeSubnetMgr:true kubeApiUrl: kubeAnnotationPrefix:flannel.alpha.coreos.com kubeConfigFile: iface:[]
 ifaceRegex:[] ipMasq:true ifaceCanReach: subnetFile:/run/flannel/subnet.env publicIP: publicIPv6: subnetLeaseR
enewMargin:60 healthzIP:0.0.0.0 healthzPort:0 iptablesResyncSeconds:5 iptablesForwardRules:true netConfPath:/et
c/kube-flannel/net-conf.json setNodeNetworkUnavailable:true}
W0208 13:54:04.147077       1 client_config.go:618] Neither --kubeconfig nor --master was specified.  Using the
 inClusterConfig.  This might not work.
I0208 13:54:04.162594       1 kube.go:139] Waiting 10m0s for node controller to sync
I0208 13:54:04.162624       1 kube.go:469] Starting kube subnet manager
I0208 13:54:05.163395       1 kube.go:146] Node controller sync successful
I0208 13:54:05.163429       1 main.go:231] Created subnet manager: Kubernetes Subnet Manager - controlnode
I0208 13:54:05.163440       1 main.go:234] Installing signal handlers
I0208 13:54:05.163878       1 main.go:468] Found network config - Backend type: vxlan
E0208 13:54:05.164108       1 main.go:268] Failed to check br_netfilter: stat /proc/sys/net/bridge/bridge-nf-ca
ll-iptables: no such file or directory
```

- Load the “br_netfilter” module in kernel

```
tausif@controlnode:~$ sudo modprobe br_netfilter
```

- Above setting will be revoked after restart, so make it persistent and enable bridge traffic filtering

```
tausif@controlnode:~$ echo 'br_netfilter' | sudo tee -a /etc/modules-load.d/kubernetes.conf
br_netfilter
tausif@controlnode:~$ sudo sysctl -w net.bridge.bridge-nf-call-iptables=1
sudo sysctl -w net.bridge.bridge-nf-call-ip6tables=1
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
```

- Adding settings to sysctl.conf file so that it will make persistent. And apply the changes

```
tausif@controlnode:~$ sudo vim /etc/sysctl.conf
tausif@controlnode:~$ sudo sysctl --system
```

- Verify the output

```
* Applying /etc/sysctl.d/99-sysctl.conf ...
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
* Applying /etc/sysctl.conf ...
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
```

- Now delete the pods, it will create again and works

```
tausif@controlnode:~$ kubectl delete pod -n kube-flannel --all
pod "kube-flannel-ds-j4bl5" deleted
tausif@controlnode:~$ kubectl get pods -A
NAMESPACE      NAME                                   READY   STATUS    RESTARTS   AGE
kube-flannel   kube-flannel-ds-7mtql                  1/1     Running   0          16s
kube-system    coredns-668d6bf9bc-5r8cl               1/1     Running   0          90m
kube-system    coredns-668d6bf9bc-l7w48               1/1     Running   0          90m
kube-system    etcd-controlnode                       1/1     Running   0          90m
kube-system    kube-apiserver-controlnode             1/1     Running   0          90m
kube-system    kube-controller-manager-controlnode    1/1     Running   0          90m
kube-system    kube-proxy-s56zc                       1/1     Running   0          90m
kube-system    kube-scheduler-controlnode             1/1     Running   0          90m
```

Check the node status

```
tausif@controlnode:~$ kubectl get nodes
NAME          STATUS   ROLES           AGE   VERSION
controlnode   Ready    control-plane   93m   v1.32.1
tausif@controlnode:~$
```

6. Adding Worker Nodes

Retrieve Join Command

kubeadm token create --print-join-command

Run Join Command on Worker Nodes

sudo kubeadm join <Master-IP>:6443 --token <TOKEN> --discovery-token-ca-cert-hash sha256:<HASH>

```
tausif@node01:~$ sudo kubeadm join 192.168.95.14:6443 --token 5x44wj.fmkrkxb9j6s0pvog        --discovery-token
-ca-cert-hash sha256:6c6d887174c1d5ffd4a5d0fada34095bf77bc8739ab2248721ccd080f31df2cb
[preflight] Running pre-flight checks
[preflight] Reading configuration from the "kubeadm-config" ConfigMap in namespace "kube-system"...
[preflight] Use 'kubeadm init phase upload-config --config your-config.yaml' to re-upload it.
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.501725128s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

```
tausif@node02:~$ sudo kubeadm join 192.168.95.14:6443 --token 5x44wj.fmkrkxb9j6s0pvog        --discovery-token
-ca-cert-hash sha256:6c6d887174c1d5ffd4a5d0fada34095bf77bc8739ab2248721ccd080f31df2cb
[preflight] Running pre-flight checks
[preflight] Reading configuration from the "kubeadm-config" ConfigMap in namespace "kube-system"...
[preflight] Use 'kubeadm init phase upload-config --config your-config.yaml' to re-upload it.
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.00178725s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

NOTE:

If swap space is not disabled on worker nodes and ip_forwarding was not enabled the worker node will not able to join the master node.

See the below error:

```
tausif@node01:~$ sudo kubeadm join 192.168.95.14:6443 --token 5x44wj.fmkrkxb9j6s0pvog        --discovery-token
-ca-cert-hash sha256:6c6d887174c1d5ffd4a5d0fada34095bf77bc8739ab2248721ccd080f31df2cb
[sudo] password for tausif:
[preflight] Running pre-flight checks
        [WARNING Swap]: swap is supported for cgroup v2 only. The kubelet must be properly configured to use sw
ap. Please refer to https://kubernetes.io/docs/concepts/architecture/nodes/#swap-memory, or disable swap on the
 node
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set t
o 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
```

```
tausif@node02:~$ sudo kubeadm join 192.168.95.14:6443 --token 5x44wj.fmkrkxb9j6s0pvog \
        --discovery-token-ca-cert-hash sha256:6c6d887174c1d5ffd4a5d0fada34095bf77bc8739ab2248721ccd080f31df2cb
[sudo] password for tausif:
[preflight] Running pre-flight checks
        [WARNING Swap]: swap is supported for cgroup v2 only. The kubelet must be properly configured to use sw
ap. Please refer to https://kubernetes.io/docs/concepts/architecture/nodes/#swap-memory, or disable swap on the
 node
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set t
o 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
tausif@node02:~$ sudo swapoff -a
tausif@node02:~$ sudo vim /etc/sysctl.conf
tausif@node02:~$ sudo sysctl -p
net.ipv4.ip_forward = 1
tausif@node02:~$ cat /proc/sys/net/ipv4/ip_forward
```

You need to swapoff and enable the ip_forwarding to solve the issues.

---

7. Verifying Cluster Status

Check Node Status

kubectl get nodes

```
tausif@controlnode:~$ kubectl get nodes
NAME          STATUS     ROLES           AGE    VERSION
controlnode   Ready      control-plane   101m   v1.32.1
node01        Ready      <none>          3m8s   v1.32.1
node02        NotReady   <none>          7s     v1.32.1
tausif@controlnode:~$ kubectl get nodes
NAME          STATUS   ROLES           AGE     VERSION
controlnode   Ready    control-plane   102m    v1.32.1
node01        Ready    <none>          3m23s   v1.32.1
node02        Ready    <none>          22s     v1.32.1
```

Check System Pods

kubectl get pods -A

---

8. Deploying a Sample Application

Create Deployment

kubectl create deployment nginx --image=nginx

Expose the Application

kubectl expose deployment nginx --type=NodePort --port=80

(*Include screenshots of deployment and service creation*)

---

9. Troubleshooting Common Issues

- Node Not Ready: Check kubectl describe node <node-name>

  Additional Troubleshooting Steps:
  1. Check kubelet logs:
     journalctl -u kubelet -f

  2. Verify API server connectivity:
     nc -vz <master-ip> 6443

- Pods Stuck in Pending: Check kubectl describe pod <pod-name>
- Network Issues: Verify sysctl net.bridge.bridge-nf-call-iptables

---

10. Conclusion

Congratulations! You have successfully set up a Kubernetes cluster with a control plane and worker nodes, deployed a networking solution, and verified the cluster's health. This guide provided step-by-step instructions on:

- Installing and configuring Kubernetes components.
- Setting up a control plane and adding worker nodes.
- Deploying a pod network for inter-pod communication.
- Verifying cluster status and ensuring nodes are ready.
- Running a sample application to validate the setup.

With the cluster up and running, you can now explore advanced Kubernetes features, such as:

- **Deploying Stateful Applications** – Learn how to manage databases and persistent storage with Persistent Volumes (PVs) and Persistent Volume Claims (PVCs).
- **Implementing Ingress Controllers** – Set up ingress controllers to expose applications via custom domain names.
- **Securing the Cluster** – Apply Role-Based Access Control (RBAC), Network Policies, and Pod Security Standards.
- **Monitoring & Logging** – Use Prometheus, Grafana, and Fluentd to monitor and analyze cluster activity.
- **Scaling & Auto-healing** – Learn about Horizontal Pod Autoscaler (HPA) and cluster auto-scaling.

This Kubernetes setup lays the foundation for container orchestration and microservices deployment. As a next step, consider integrating CI/CD pipelines and service mesh technologies like Istio to enhance cluster efficiency.

🚀 **Keep experimenting, deploying, and optimizing your Kubernetes cluster for real-world workloads!**

Next Steps:

- **Backup and Disaster Recovery** - Use `etcdctl snapshot save` to back up etcd.
- **Upgrading Kubernetes** - Follow best practices for updating Kubernetes versions.
- **Setting Up Monitoring** - Implement Prometheus and Grafana for cluster monitoring.

Appendix

This Kubernetes Cluster Setup Guide is intended for **educational purposes only**. The content, including commands, configurations, and explanations, is meant to help learners understand and deploy Kubernetes in a controlled environment.

For inquiries, collaboration, or further assistance, feel free to reach out to:

📌 **Tausif Shaikh**
📧 **Email**: shaikh.only@gmail.com
📞 **Contact**: +91 99244 25668