Introduction to Kubernetes

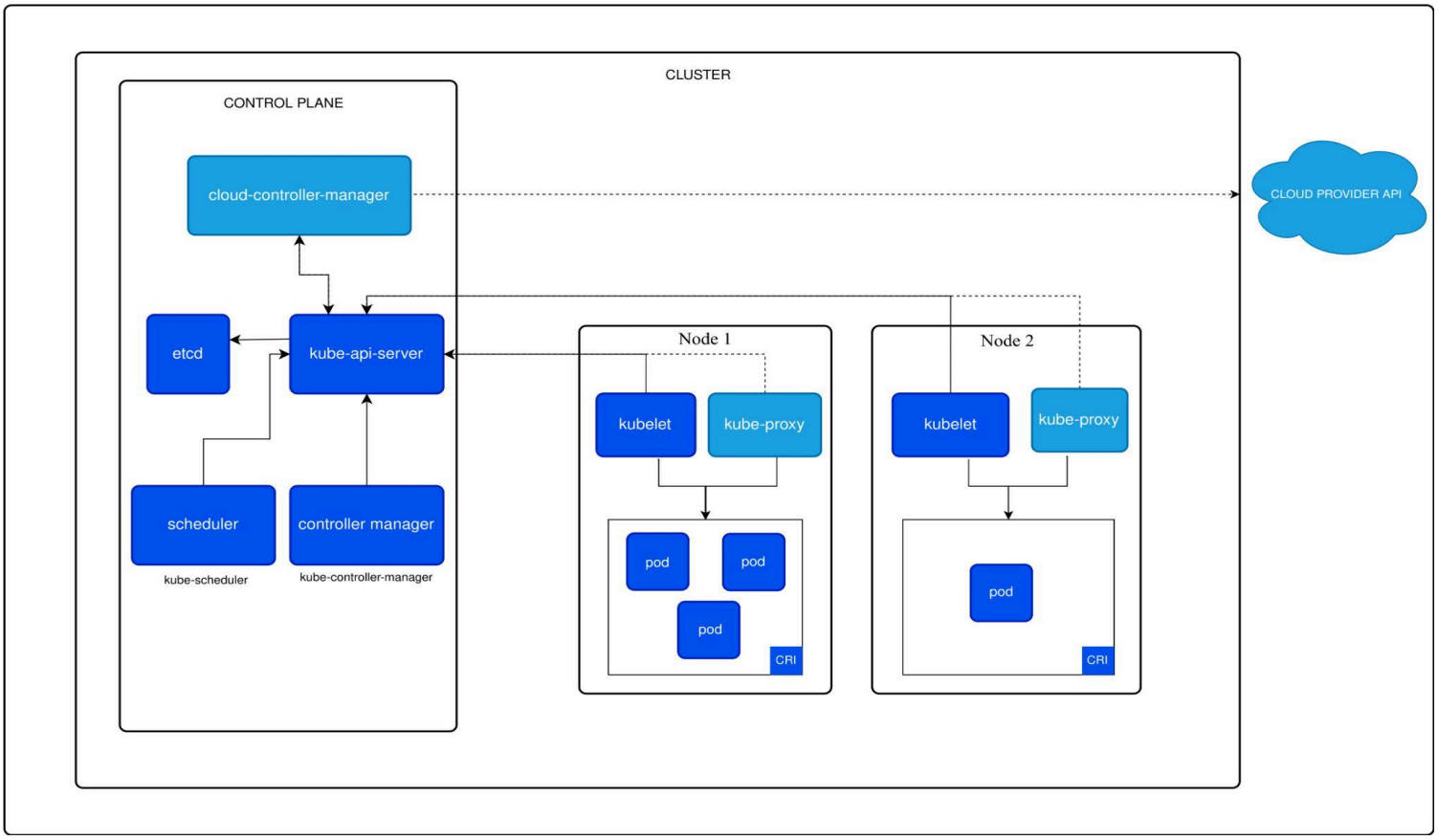
Kubernetes (K8s) is an open-source container orchestration platform for automating deployment, scaling, and management of containerized applications. It was originally developed by Google and is now maintained by the Cloud Native Computing Foundation (CNCF).It provides a robust and scalable system for managing microservices across multiple nodes. Understanding Kubernetes architecture is crucial for effectively deploying and managing applications in a distributed environment.

Kubernetes Architecture Overview

Kubernetes follows a **master-worker architecture** where the **Control Plane** manages the cluster, and **Worker Nodes** execute application workloads.

Key Components of Kubernetes Architecture:

1. **Control Plane** (Manages the cluster)
2. **Worker Nodes** (Run application workloads)
3. **Networking & Storage Components** (Enable seamless communication and data persistence)



- ## Control Plane/Master Node Components

The Control Plane is responsible for managing the cluster, maintaining the desired state, scheduling workloads, and handling API requests. It includes:

a) API Server (`kube-apiserver`)

- Acts as the frontend of the Kubernetes cluster.
- Exposes REST APIs for interaction with the cluster.
- All internal and external requests go through this server.

b) Controller Manager (`kube-controller-manager`)

- Runs various controllers to maintain the cluster's desired state.
- Key controllers include:
    - **Node Controller**: Detects and responds when a node fails.
    - **Replication Controller**: Ensures the specified number of pod replicas are running.
    - **Endpoint Controller**: Manages service endpoints.

c) Scheduler (`kube-scheduler`)

- Assigns Pods to nodes based on resource availability, constraints, and policies.
- Factors like CPU, memory, and affinity rules are considered during scheduling.

d) etcd (Distributed Key-Value Store)

- Stores all cluster data, including configuration, policies, and state information.
- Provides high availability and consistency across the cluster.

- ## Worker Node Components

Worker Nodes are responsible for running application workloads. Each node contains:

a) Kubelet

- A lightweight agent that ensures the required containers are running on a node.
- Communicates with the API Server to receive instructions.

b) Kube Proxy

- Manages network rules and facilitates communication between different services and pods.
- Implements service discovery and load balancing.

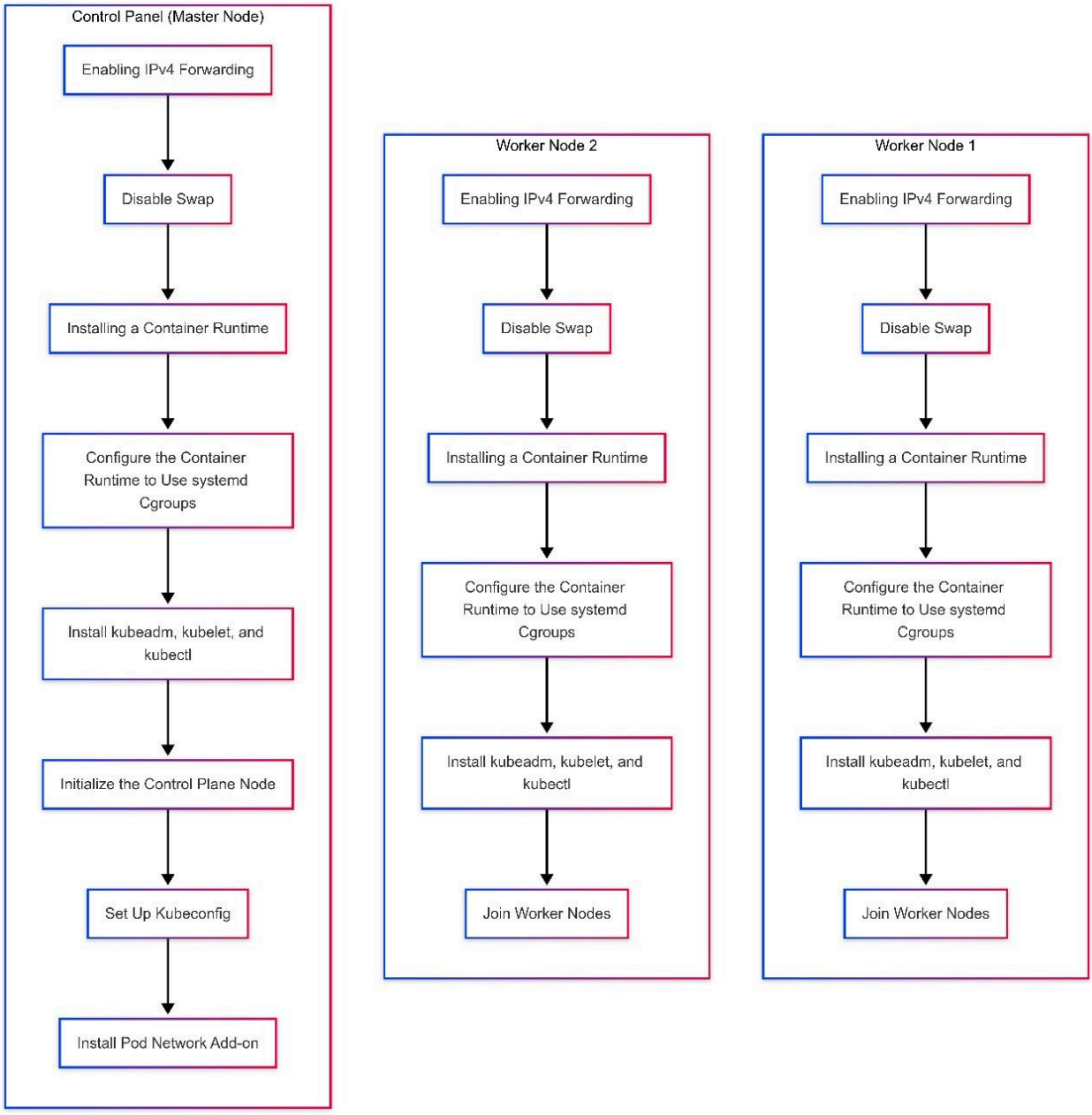
c) Container Runtime

- Runs and manages containers inside pods.
- Common runtimes include Docker, containerd, and CRI-O.

“ Kubernetes architecture is designed to be modular, scalable, and resilient. The **Control Plane** ensures cluster management, while **Worker Nodes** execute workloads. The networking and storage subsystems enhance application reliability and performance.

Understanding these core components helps DevOps engineers and cloud professionals deploy and manage Kubernetes . “

Production Ready Kubernetes Cluster setup in 2025 — A Complete Guide



Step 1 : Enable IPv4 packet forwarding :-

To manually enable IPv4 packet forwarding:

 # sysctl params required by setup, params persist across reboots

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.ipv4.ip_forward = 1

EOF

 # Apply sysctl params without reboot

sudo sysctl --system

Verify that net.ipv4.ip_forward is set to 1 with:

sysctl net.ipv4.ip_forward

Step 2: Disable swap

Follow this step in all nodes . using following command :-

sudo swapoff -a

sudo vim /etc/fstab

commentout swap commandline -

#/swapfile none swap sw 0 0

Step 3: Install Container Runtime Interface

Follow this step in all nodes .

To run Kubernetes you need at least one CRI which could be –

- containerd
- CRI-O
- Docker Engine
- Mirantis Container Runtime

In this tutorial we are going with most commonly used CRI - Containerd .

To install containerd simply run these commands in your terminal -

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

Add the repository to Apt sources:

echo \

  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/debian \

  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \

  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

#Copy completey from deb to null

  sudo apt-get update

  sudo apt-get install containerd.io

Congratulations you have successfully installed containerd .

Check the containerd status using following command :-

Sudo systemctl status containerd

Step 4: Configure Cgroup Driver ---

Do Follow this step in all nodes .

There are two cgroup drivers available:

- cgroupfs
- systemd

In This Tutorial we are going with **systemd cgroup driver**

First reset the containerd configuration with -

containerd config default > /etc/containerd/config.toml

To use the systemd cgroup driver in /etc/containerd/config.toml with runc,

Open the file /etc/containerd/config.toml using vim/nano and search for mentioned plugin carefully

And now set the value SystemdCgroup = true

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]
  ...
  [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
            SystemdCgroup = true
```

That’s it my friend you have successfully configured the cgroup driver .

Step 5: Installing kubeadm, kubelet, kubectl

These instructions are for Kubernetes v1.32. and do follow this step in all of your nodes

Update the apt package index and install packages needed to use the Kubernetes apt repository:

```
sudo apt-get update
# apt-transport-https may be a dummy package; if so, you can skip that package
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
```

#Download the public signing key for the Kubernetes package repositories. The same signing key is used for all repositories so you can disregard the version in the URL:

If the directory `/etc/apt/keyrings` does not exist, it should be created before the curl command

sudo mkdir -p -m 755 /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.32/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:/v1.32/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

#change the v1.32 to the latest one if it's available .

Update the apt package index, install kubelet, kubeadm and kubectl, and pin their version:

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

(Optional) Enable the kubelet service before running kubeadm:

sudo systemctl enable --now kubelet

That's it my friend you have successfully Installed Kubernetes .

Step 6: Creating Cluster

Follow this guide in Masternode only-

Run the following Command to create the cluster :-

kubeadm init --apiserver-advertise-address 192.168.1.108 --pod-network-cidr 10.244.0.0/16 --cri-socket unix:///var/run/containerd/containerd.sock

please be patient until you see something like this -

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 192.168.194.176:6443 --token 3dwbkq.spy8i3awii1r25s4 \
        --discovery-token-ca-cert-hash sha256:6935c5600d5fcbdb7d1f942221c353f26b4b7493eabc4d14d331a4632bcb3a12
```

Now follow the instruction given by kubeadm init command

1st - mkdir -p $HOME/.kube

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

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

Run These Commands One By one to enable Kubectl command for a Regular User .

2nd - Alright ! at this point all you just need to do is to join with worker nodes { as a root user } … simply copy the kubeadm join …. { followed by token } command given by kube init as mentioned and paste it to the worker nodes that's it you have successfully created a production ready cluster .

Note –

If you are facing any problem joining with worker node you can also try these commands –

1st – kubeadm reset

2nd – systemctl restart containerd

Step 7: Container Network Interface (CNI) Setup

At this point you just have need to install a CNI to create a Proper Netwrok interface . to do so you have variety of options available as follows :-

- ACI provides integrated container networking and network security with Cisco ACI.
- Antrea operates at Layer 3/4 to provide networking and security services for Kubernetes, leveraging Open vSwitch as the networking data plane. Antrea is a CNCF project at the Sandbox level.
- Calico is a networking and network policy provider. Calico supports a flexible set of networking options so you can choose the most efficient option for your situation, including non-overlay and overlay networks, with or without BGP. Calico uses the same engine to enforce network policy for hosts, pods, and (if using Istio & Envoy) applications at the service mesh layer.
- Canal unites Flannel and Calico, providing networking and network policy.
- Cilium is a networking, observability, and security solution with an eBPF-based data plane. Cilium provides a simple flat Layer 3 network with the ability to span

multiple clusters in either a native routing or overlay/encapsulation mode, and can enforce network policies on L3-L7 using an identity-based security model that is decoupled from network addressing. Cilium can act as a replacement for kube-proxy; it also offers additional, opt-in observability and security features. Cilium is a CNCF project at the Graduated level.

- CNI-Genie enables Kubernetes to seamlessly connect to a choice of CNI plugins, such as Calico, Canal, Flannel, or Weave. CNI-Genie is a CNCF project at the Sandbox level.
- Contiv provides configurable networking (native L3 using BGP, overlay using vxlan, classic L2, and Cisco-SDN/ACI) for various use cases and a rich policy framework. Contiv project is fully open sourced. The installer provides both kubeadm and non-kubeadm based installation options.
- Contrail, based on Tungsten Fabric, is an open source, multi-cloud network virtualization and policy management platform. Contrail and Tungsten Fabric are integrated with orchestration systems such as Kubernetes, OpenShift, OpenStack and Mesos, and provide isolation modes for virtual machines, containers/pods and bare metal workloads.
- Flannel is an overlay network provider that can be used with Kubernetes.
- Gateway API is an open source project managed by the SIG Network community and provides an expressive, extensible, and role-oriented API for modeling service networking.
- Knitter is a plugin to support multiple network interfaces in a Kubernetes pod.
- Multus is a Multi plugin for multiple network support in Kubernetes to support all CNI plugins (e.g. Calico, Cilium, Contiv, Flannel), in addition to SRIOV, DPDK, OVS-DPDK and VPP based workloads in Kubernetes.
- OVN-Kubernetes is a networking provider for Kubernetes based on OVN (Open Virtual Network), a virtual networking implementation that came out of the Open vSwitch (OVS) project. OVN-Kubernetes provides an overlay based networking implementation for Kubernetes, including an OVS based implementation of load balancing and network policy.
- Nodus is an OVN based CNI controller plugin to provide cloud native based Service function chaining(SFC).
- NSX-T Container Plug-in (NCP) provides integration between VMware NSX-T and container orchestrators such as Kubernetes, as well as integration between NSX-T and container-based CaaS/PaaS platforms such as Pivotal Container Service (PKS) and OpenShift.
- Nuage is an SDN platform that provides policy-based networking between Kubernetes Pods and non-Kubernetes environments with visibility and security monitoring.

- Romana is a Layer 3 networking solution for pod networks that also supports the NetworkPolicy API.
- Spiderpool is an underlay and RDMA networking solution for Kubernetes. Spiderpool is supported on bare metal, virtual machines, and public cloud environments.
- Weave Net provides networking and network policy, will carry on working on both sides of a network partition, and does not require an external database.

We are going With **Weave Net** CNI , just copy the below command and paste it to the master node - kubectl apply -f https://reweave.azurewebsites.net/k8s/v1.29/net.yaml

Run - " kubectl get no " to see the Status

if it is Ready which mean you are all set .

Authors

Ram Nivas , Mohni , Anuradha , Farhat , Divya

Mentor

Anil Kumar Dahiya

DevOs