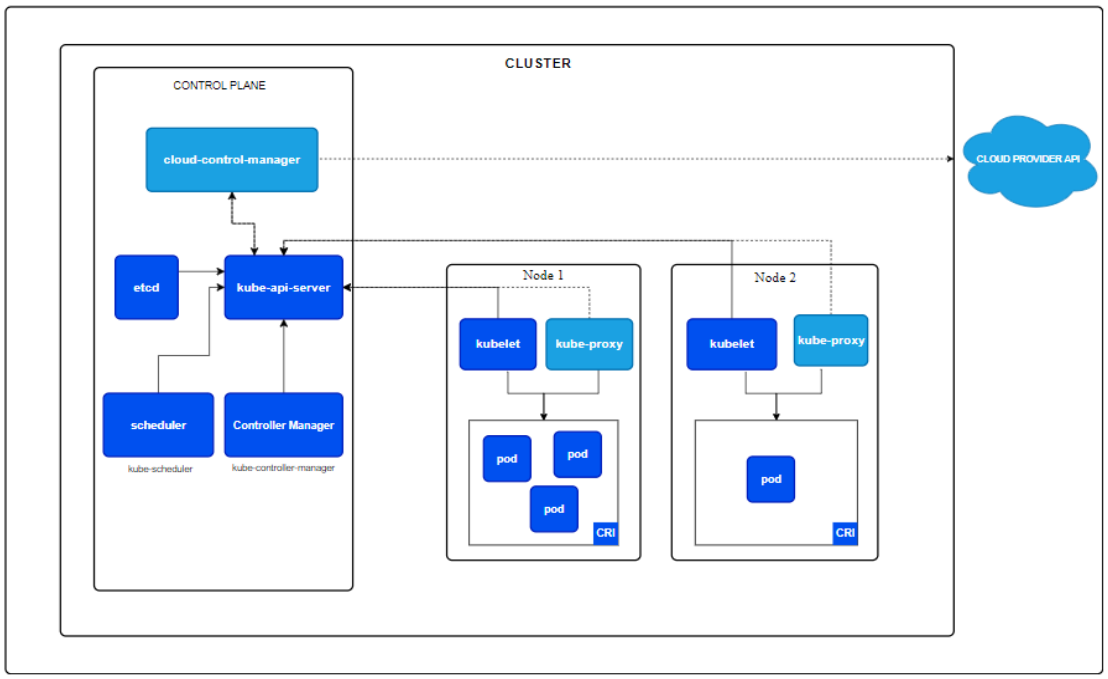
CREATING KUBERNETES CLUSTER USING KUBEADM



Prerequisites before creating the cluster

1. A compatible linux host
2. 2GB or more of RAM per machine
3. 2CPU'S or more
4. Full network connectivity between all machines in the cluster
5. Unique hostname, MAC address and product_uuid for every node
6. Certain ports are open on your machine

➔ **Kubernetes Control Plane Ports (Master Node)**

- **6443**: API server (kube-apiserver) for communication with the cluster.
- **2379-2380**: etcd server client API (used by the Kubernetes API server).
- **10250**: Kubelet API (for control plane to communicate with worker nodes).
- **10259**: kube-scheduler.
- **10257**: kube-controller-manager.

➔ **Kubernetes Worker Node Ports**

- **10250**: Kubelet API (for control plane to communicate with worker nodes).
- **30000-32767**: NodePort services (to expose services externally).

➔ **Overlay Network / Pod Communication (CNI)**

- **6783-6784**: Weave network (if using Weave).
- **8472**: Flannel (UDP) (if using Flannel).
- **8285/8472**: Calico networking.
- **179**: BGP (if using Calico in BGP mode).

➔ **DNS and Other Services**

- **53**: DNS (TCP/UDP) for CoreDNS.
- **9153**: CoreDNS monitoring.

➔ **Container Runtime Ports (if Docker is used)**

- **2375-2376**: Docker daemon (Docker-specific, for remote API, used in rare cases).

7. Swap configuration (turn-off)

   Swap is disabled during Kubernetes cluster creation to ensure consistent memory management and performance, as Kubernetes relies on precise resource allocation and does not handle swapping well. Enabling swap can lead to unpredictable behaviour and performance issues

Step to follow during creation of k8's cluster

❖ Control plane [minimum t2.medium(ubuntu) , cpu 2, memory 4GB]

1) Enabling IPV4 Forwarding
2) Disable swap
3) Installing a container runtime
4) Configure the container runtime
5) To use systemd Cgroups
6) Install kubeadm, kubelet and kubectl
7) Initialize the control plane node
8) Setup kubeconfig
9) Install pod network add-on (CNI)

❖ Work nodes [minimum t2.micro(ubuntu), cpu 1, memory 1 GB]

1) Enabling IPV4 Forwarding
2) Disable swap
3) Installing a container runtime
4) Configure the container runtime
5) To use systemd Cgroups
6) Install kubeadm, kubelet and kubectl
7) Join worker nodes

Check this k8's doc for more resource:

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

➔Enabling IPV4

Link: https://kubernetes.io/docs/setup/production-environment/container-runtimes/

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.ipv4.ip_forward = 1
EOF

sudo sysctl –system
```

CREATING KUBERNETES CLUSTER USING KUBEADM

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

```
sysctl net.ipv4.ip_forward
```

➔ Turn off swap

```
sudo swapoff -a
```

➔ Container runtime

link: https://docs.docker.com/engine/install/ubuntu/

Add Docker's official GPG key:

```
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/ubuntu/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/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

➔ Install container d

```
sudo apt-get install containerd.io
```

➔ Configure cgroup

https://kubernetes.io/docs/setup/production-environment/container-runtimes/#containerd

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

make the change in "**vim /etc/containerd/config.toml**" file

SystemdCgroup = false to SystemdCgroup = true , we can check in the link

Restart containerd

```
sudo systemctl restart containerd
```

CREATING KUBERNETES CLUSTER USING KUBEADM

➔Install kubeadm, kubelet and kubectl

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

for Ubuntu:

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
```

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/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:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

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

Run this command only on control plane node not on worker nodes

```
kubeadm init --apiserver-advertise-address <private ip> --pod-network-cidr <10.244.0.0/16> --cri-socket unix:///var/run/containerd/containerd.sock
```

https://kubernetes.io/docs/concepts/cluster-administration/addons/
--pod-network-cidr ➔ I chosen FLANNEL

--cri-socket : we install containerd
https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/#installing-runtime

➔Setup kubeconfig

and run the command to start using cluster give in the output, after the execution exit from root user to normal user

```
mkdir -p $HOME/.kube

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

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

CREATING KUBERNETES CLUSTER USING KUBEADM

Run this on worker node only

➔Join the worker node

In the output of **the control plane node** there will be a line and paste that line in **worker nodes** to join with the cluster

Kubeadm join <copy till end>

Kubectl get nodes – you can see nodes are still pending

➔Install pod network add-on (run on control plane)

Install CNI: calico/ fannel/ wevenet etc based on the requirement

https://kubernetes.io/docs/concepts/cluster-administration/addons/

weavenet: kubectl apply -f https://reweave.azurewebsites.net/k8s/v1.30/net.yaml

it will take few seconds

kubectl get pods -A kubectl get nodes – now nodes are in ready state