Setup k8s cluster with kudeadm :

- Reference link - https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/
- Steps to be followed :
    - Take 3 ubuntu24 vms(cloud/0npremise) . 1vm of master and two vms for worker nodes. Open all trafffic in SG.
    - Look at pre-requesites and implement on all the master & Worker nodes
    - Install containerd on all Master & Worker nodes. Setup sytemd cgroupdrivers as well
    - Install kubeadm, , Kubelet, Kubectl on all the Master & Worker nodes
    - Initialize kubeadm on Master node
        - Add admin.conf to the $HOME/.kube/config
        - Utilize the kubeadm join command from above step and add all worker nodes to the master node
    - Setup weavenet for pod communication

Step-1: Prerequesites (Master & Worker)

- A compatible Linux host. The Kubernetes project provides generic instructions for Linux distributions based on Debian and Red Hat, and those distributions without a package manager.
- 2 GB or more of RAM per machine (any less will leave little room for your apps).
- 2 CPUs or more.
- Full network connectivity between all machines in the cluster (public or private network is fine).
- Unique hostname, MAC address, and product_uuid for every node. See here for more details.
- Certain ports are open on your machines. See here for more details.
- Swap configuration. The default behavior of a kubelet was to fail to start if swap memory was detected on a node. See Swap memory management for more details. * You MUST disable swap if the kubelet is not properly configured to use swap. For example, sudo swapoff -a will disable swapping temporarily. To make this change persistent across reboots, make sure swap is disabled in config files like /etc/fstab, systemd.swap, depending how it was configured on your system.

Step-2: Install Container Runtime - containerd on all servers(Master & Worker)

- Reference Link : - https://docs.docker.com/engine/install/debian/
- To run containers in Pods, Kubernetes uses a container runtime.
- Prerequesites:
    - Reference 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

sysctl net.ipv4.ip_forward
```

- Install using the apt repository

```
# Add Docker's official GPG key:

sudo apt-get update -y
sudo apt-get install ca-certificates curl gnupg -y

# Add Docker’s official GPG key:
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Use the following command to set up the repository:
echo \
    "deb [arch="$(dpkg --print-architecture)" signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
    "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
    sudo tee /etc/apt/sources.list.d/docker.list > /dev/null


 sudo apt-get install  containerd.io

 sudo systemctl restart containerd
```

- To use the systemd cgroup driver in /etc/containerd/config.toml with runc, set
    - Reference Link : https://kubernetes.io/docs/setup/production-environment/container-runtimes/#:~:text=Configuring%20the%20systemd,cgroup%20v2.

```
ps -p 1
```

- Add below content to the /etc/containerd/config.toml , delete all data before adding .

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

- CNI Plugin Installtion :

```
sudo wget
https://github.com/containernetworking/plugins/releases/download/v1.5.1/cni-
plugins-linux-amd64-v1.5.1.tgz
```

```
sudo mkdir -p /opt/cni/bin
sudo tar Cxzvf /opt/cni/bin cni-plugins-linux-amd64-v1.1.1.tgz
```

Step-3: Install kubeadm on all the servers (Master & Worker)

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

```
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


# If the directory `/etc/apt/keyrings` does not exist, it should be created before
the curl command, read the note below.
# sudo mkdir -p -m 755 /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --
dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

# This overwrites any existing configuration in
/etc/apt/sources.list.d/kubernetes.list
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.30/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

sudo systemctl enable --now kubelet
```

step-4: Initialize kudem on Master :

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

- You will see output as , follow the steps in the output :

```
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 172.31.49.100:6443 --token 9kex8p.nr65o6s3ws4dztbb \
        --discovery-token-ca-cert-hash
sha256:5c6b3de527c8003c76e7bdbf9b53c17f1aa2867b4554140a7392871a8d56b08d
```

step-5: Setup Weavenet for pod communication

- Reference link : https://github.com/rajch/weave#using-weave-on-kubernetes

```
kubectl apply -f https://reweave.azurewebsites.net/k8s/v1.29/net.yaml
```