Kubernetes:

Installation – Version -1.34

Disable swap:

Kubernetes relies on Kublets.

Kubelets determines a nodes available resources(CPU & Memory), helps while scheduling PODs.

Kublelet is designed to base its Scheduling Decisions on PHYSICAL MEMORY.

If Swap is Present , Kubelet gets inaccurate memory capacity leading to Over Commitment & unpredictable behaviour.

Kubernetes has OOMKill Mechanism , Swap prevents this from working as intended , affects stability.

[OOMKill – Kubernete's Event that signifies a container has been terminated by the Linux Kernel's Out-Of-Memory(OOM) Killer because it exceeded its Allocated Memory Limit.]

[OOMKill identification → Run "kubectl describe pod <pod name>" → Exit Code 137 , Status OOMKilled. Remedy → Increse memory/fix memory leak using monitoring tools]

SWAP Slower than RAM, latency Spikes.

CLI to Disable SWAP:

```
 sudo swapoff -a                    #Disables all Swaps, Temporary – enabled after Reboot.
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab    # Stream Editor used to edit /etc/fstab , Permanent.
```

Enabling Kernel Modules & IP Forwarding:

```
   sudo modprobe overlay   # Loads Overlay module , Essential to stack image layers , sharing common files.
   sudo modprobe br_netfilter # Loads Bridge Netfilter module, Essential to pass traffic to IPTables , Firewalls to process.
```

#Most CNI Plugins (like Calico, Flannel, etc) and Kube-Proxy rely heavily on iptables rules for Service , pod to pod routing/communication.

# Below lines writes new config file sets kernel parameters (iptables, ipforwarding)

```
   printf "%s\n" "net.bridge.bridge-nf-call-ip6tables = 1" "net.bridge.bridge-nf-call-iptables = 1" "net.ipv4.ip_forward = 1" | sudo tee /etc/sysctl.d/kubernetes.conf
```

# applying new network settings to running kernel without reboot

```
   sudo sysctl --system
```

*! Before installing Containerd , Run Below Script to Update Docker Repository to system's package sources(/etc/apt/sources.list.d). Containerd.io located hosted in Docker Repository. Else we get Error "E: Unable to locate package containerd.io"

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


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
```

Installing Container Runtime [Ex- Containerd]

[Notations: CNI – Container Network Interface , CRI – Container Runtime Interface]

```
   sudo apt update # Update local package index
   sudo apt install -y containerd.io # Installs Containerd – responsible for container lifecycle within pods
   sudo mkdir -p /etc/containerd  # Creates config directory , where Containerd config file(config.toml) resides.
#Below Command Generates default config and inserting to config.toml, necessary to load SYSTEMD CGROUP DRIVER(used for Resource Management with Host System)
   sudo containerd config default | sudo tee /etc/containerd/config.toml
   sudo systemctl restart containerd  # Restart Containerd Services.
   sudo systemctl enable containerd # Enables on boot
```

Add Kubernetes APT Repository:

```
    sudo apt update     #Refreshes Local Package Index.
   sudo apt install -y apt-transport-https ca-certificates curl # Necessary utility installation.
   sudo curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.34/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg  # Download & Install Kubernetes GPG Signing key.
   echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.34/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list #Adds K8s Repo
   sudo apt update   # Refreshes Packages again
```

Install kubeadm, kubelet, and kubectl:

[k8s Core Components – kubeadm , kubelet , kubectl]

[kubeadm (Cluster Bootstrap Tool) – initialize the control-plane (master node) & to join worker nodes, automates cluster component setup]

[kubelet (Node Agent) – Primary agent runs on every node , register node to control plane, responsible for managing pods lifecycle , also for containers by communicating with container run time(containerd).]

[kubectl (CLI Tool) – Client Utility(Users) to interact with control plane(Actions: Deploy, inspect, log view).]

# Below command installs Core Components mentioned

```
sudo apt install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl   # Version Locking – k8s has high sensitivity on this
```

Initialize the Control Plane (Master Node Only):

# Initializes k8s Control Plane Node (Master Node).

# kubeadm init – performs critical actions below

~ Generates Security Certificates (for API Server, etcd, etc)

~Generates kubeconfig files for administrators and components

~Installs Control Plane Components (API Server, Control Manager, Scheduler, etcd) as static Pods in the "/etc/kubernetes/manifests" directory.

~Installs CoreDNS and kube-proxy

~Prints "kubeadm join" Command for worker nodes.

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16 # Use a suitable Pod network CIDR
```

[notes: 10.244.0.0/16 – Standard default CIDR for CNI Plugins like Flannel, Calico]

[Flannel – Creates Simple Overlay Network by , Assigns unique subnets to each nodes, uses encapsulation VXLAN (IP-in-IP), making communication between Pods across nodes (Cluster Wide) as on same local network, used in smaller cluster environments.]

[Calico – Operates at OSI Layer 3 , Uses BGP to route packets between nodes directly, designed for production grade clusters, high performance networking, robust network policy enforcement. Allows defining fine-grained rules to control ingress and egress traffic for pods, namespaces, etc,. enabling strong network segmentations and security.]

[Canal – CNI Plugin – Combines Flannel's Networking Capabilities & Calico's Network Policy Enforcement]

We get kubeadm join command for worker nodes.

Setting Up Kubectl (Master Node Only):

```
mkdir -p $HOME/.kube    #creates .kube (user's home directory) , which has configuration file(config).
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config #Copies admin.conf to .kube/config
```

[/etc/Kubernetes/admin.conf – generated after kubeadm init , contains Cluster definition, User Credentials and Security Certificates to authenticate administrator]

[$HOME/.kube/config – Kubectl uses it automatically as its configuration file]

# Changes file permission to Current-Non Root User(UID) and Group(GID)

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

Install a Pod Network Add-on (Master Node Only):

#Below command fetch and apply Flannel CNI Plugin [kube-flannel.yml – Configuration file of Flannel, Deploying Flannel Agent as DaemonSet, Necessary permissions (ServiceAccount & ClusterRole) created so the Flannel Agent reads POD CIDR and update node information to Kubernetes API Server in Master Node, Creates Cross-Node Tunnels-Cluster Wide.]

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

#Generate kubeadm join command and save in a file.

```
sudo kubeadm token create --print-join-command > k8s-join
```

Join Worker Nodes Using kubeadm join command provided from kubeadm init command output.

Kubernetes Installation Setup is Done , Run “kubectl get nodes” to view nodes and Ready status.

In Worker Nodes:

Before Executing “kubeadm join” setup below prerequisites.

```
sudo swapoff -a

sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab

sudo modprobe overlay

sudo modprobe br_netfilter

printf "%s\n" "net.bridge.bridge-nf-call-ip6tables = 1" "net.bridge.bridge-nf-call-iptables = 1"
"net.ipv4.ip_forward = 1" | sudo tee /etc/sysctl.d/kubernetes.conf

sudo sysctl --system

sudo apt update

sudo apt install -y ca-certificates curl gnupg

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


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 update
   sudo apt install -y containerd.io
   sudo mkdir -p /etc/containerd
   sudo containerd config default | sudo tee /etc/containerd/config.toml
   sudo systemctl restart containerd

   sudo systemctl enable containerd

   sudo apt update

   sudo apt install -y apt-transport-https ca-certificates curl

   sudo curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.34/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:/v1.34/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

   sudo apt update

   sudo apt install -y kubelet kubeadm kubectl

   sudo apt-mark hold kubelet kubeadm kubectl

   sudo kubeadm join <ControlPlane-IP>:<Port> --token <Token> --discovery-token-ca-cert-hash sha256:<Hash>
```

*! If Error occurs Run “sudo kubeadm reset -f” and Rerun from 1st Step in Worker node to Join Cluster.