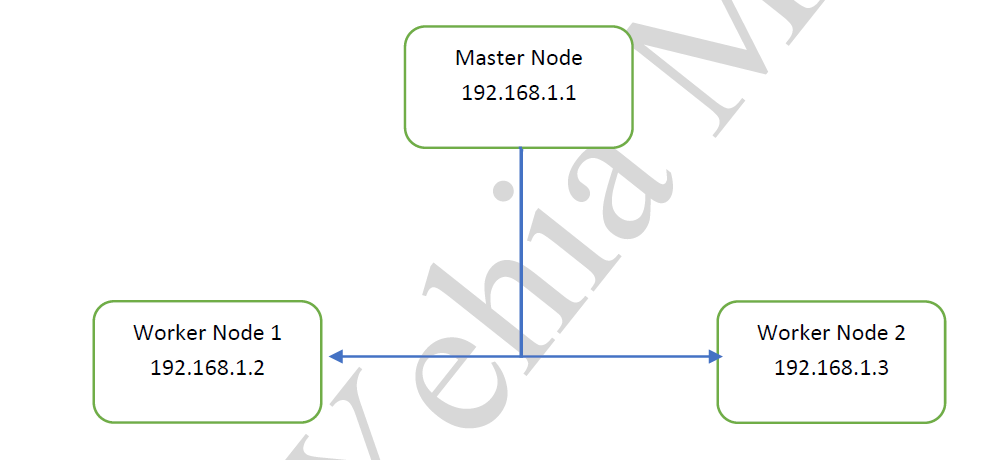
* **Install Kubernates(k8s) from scratch: -**
* ➢ Requirements: -
* 2 ubuntu machines – OS: Ubuntu 20.04.
* 2- 2 GB or more of RAM per machine.
* 3- 2 CPUs or more.
* 4- Full network connectivity between all machines in the cluster, (you can disable firewall just testing environment).
* 5- Unique hostname, MAC address, and product\_uuid for every node.
* 6- Root user.

Architecture: -



**Installation Steps: -**

**a) Command run through ( Master node – Worker Node 1 – Worker Node 2 ) :-**

**1- You MUST disable swap $$ sudo swapoff -a**

**Then disable swap as a below $$ nano /etc/fstab**

1. **- Set up the IPV4 bridge on all nodes ( run all below at one time).**

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

overlay

br\_netfilter

EOF

sudo modprobe overlay

sudo modprobe br\_netfilter

# sysctl params required by setup, params persist across reboots

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

net.bridge.bridge-nf-call-iptables = 1

net.bridge.bridge-nf-call-ip6tables = 1

net.ipv4.ip\_forward = 1

EOF

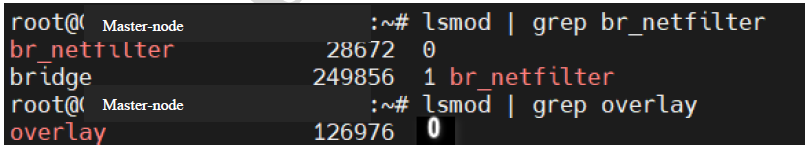
# Apply sysctl params without reboot

sudo sysctl --system

**3- Verify that the br\_netfilter, overlay modules are loaded by running the following commands:**

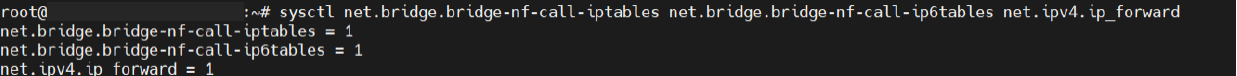
**$$lsmod | grep br\_netfilter**

**$$lsmod | grep overlay**

****

**Verify that the net.bridge.bridge-nf-call-iptables, net.bridge.bridge-nf-call-ip6tables, and net.ipv4.ip\_forward system variables are set to 1 in your sysctl config by running the following command:**

**$$sysctl net.bridge.bridge-nf-call-iptables net.bridge.bridge-nf-call-ip6tables net.ipv4.ip\_forward**



**Disable firewall (to make sure 3 machines can connect together): -**

**$$ufw status**



**- Install Containerd from docker website: -**

**$$sudo mkdir /etc/apt/keyrings**

**$$sudo apt-get update**

**$$sudo apt-get install 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

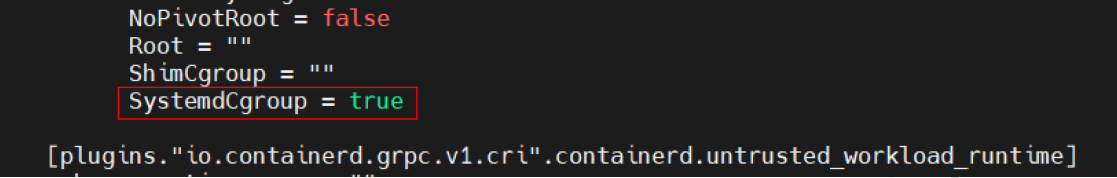
**$$sudo apt-get update**

**$$sudo apt-get install containerd.io**

**$$systemctl status containerd.service**

**Set up Cgroup as a true (systemd)**

**$$ nano /etc/containerd/config.toml**



**Then restart service to read new configuration.**

$$ systemctl restart containerd.service

1. **- Let’s install kubelet, kubeadm, and kubectl to create a Kubernetes cluster. They play an important role in managing a Kubernetes cluster.**

**$$ sudo apt-get update**

**$$ sudo apt-get install -y apt-transport-https ca-certificates curl**

**$$ mkdir -p /etc/apt/keyrings**

**$$ curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-archive-keyring.gpg**

**$$ echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list**

**$$ sudo apt-get update**

**$$ sudo apt install -y kubelet=1.25.4-00 kubeadm=1.25.4-00 kubectl=1.25.4-00**

**$$ sudo apt-mark hold kubelet kubeadm kubectl**

**To check the installation of kubelet-kubeadm-kubectl :-**

**$$ kubeadm version**

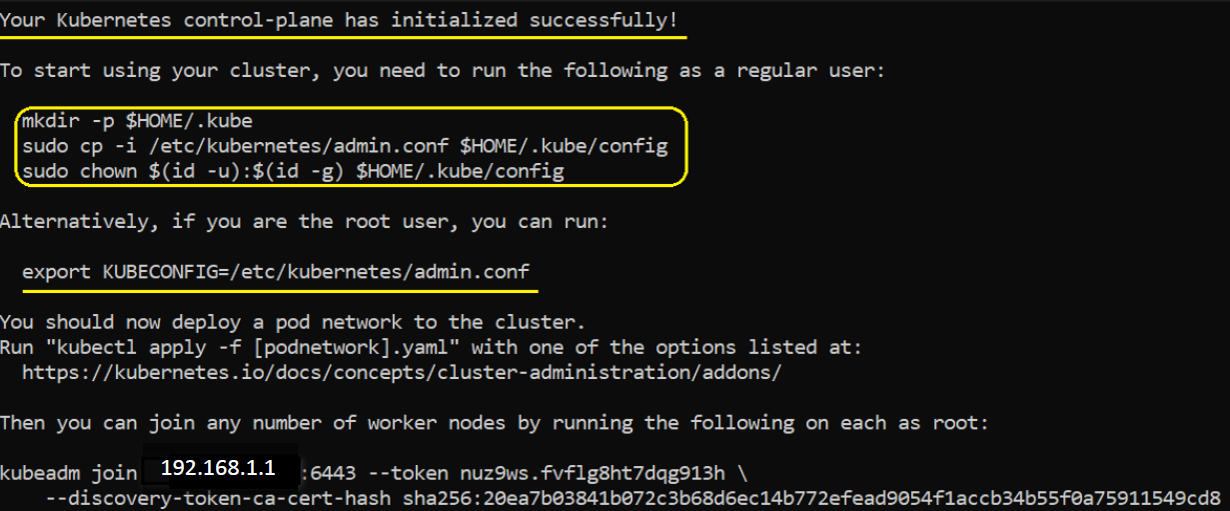
**$$ kubelet --version**

**$$ kubectl version –short**

1. **Command run through ( Master node only – 192.168.1.1 ) :-**
2. **- initialize your master node. The --pod-network-cidr flag is setting the IP address range for the pod network (Must use range 10.244.0.0 for flannel network package – Next step).**
3. **192.168.1.1 is the IP of master node.**

**$$kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=192.168.1.1**

The result of initialize is below:



1. **- Set up config file.**

**$$ mkdir -p $HOME/.kube**

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

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

1. **c) Join workers node to the cluster through (WorkerNode1 192.168.1.2 - WorkerNode2 192.168.1.3) in upper photo.**
2. **- From machine WorkerNode1 run**

**$$ kubeadm join 192.168.1.1:6443 --token d2tgh8.f9q3vjf1i5t1uneu \**

**--discovery-token-ca-cert-hash sha256:59b6ac2294eb69ccf84743fc2b9ea5113b64bbe5ea0d5372938b1e81468c47da**

**Copy certificate from MasterNode path /root/.kube/config**

**Then paste it in WorkerNode1 in same path /root/.kube/config**

**- From machine WorkerNode2 run**

**$$ kubeadm join 192.168.1.1:6443 --token d2tgh8.f9q3vjf1i5t1uneu \**

**--discovery-token-ca-cert-hash sha256:59b6ac2294eb69ccf84743fc2b9ea5113b64bbe5ea0d5372938b1e81468c47da**

**Copy certificate from MasterNode path /root/.kube/config**

**Then paste it in WorkerNode2 in same path /root/.kube/config**

1. **d) Command run through ( Master node only 192.168.1.1 ) :-**
2. **- Install flannel package for network solution.**

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

1. Finally,

**$$ kubectl get nodes**

