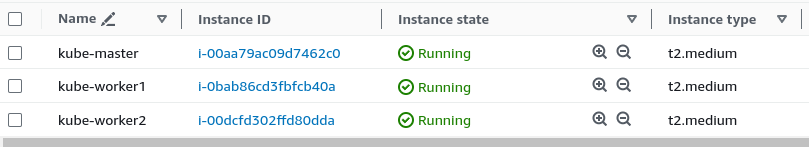
# Experiment No. 3

**Objective:** To understand the Kubernetes Cluster Architecture and to set up a Kubernetes Cluster on Linux Machines/Cloud.

# Procedure:

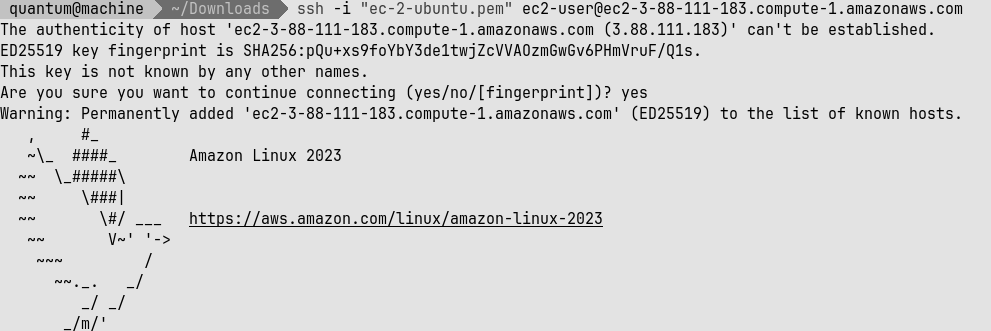
1. **Instance Setup:**
   * Create three EC2 instances using Amazon Linux as the operating system.
   * Ensure that SSH traffic is allowed from any source.
   * For optimal performance, choose an instance type of at least t2.medium, as Kubernetes recommends a minimum of 2 vCPUs.



# SSH Access:

* + SSH into each of the three machines using separate terminal windows: ssh -i

<keyname>.pem ubuntu@<public\_ip\_address>



# Docker Installation and Configuration:

* + On all three machines, install Docker with the command: sudo yum install docker -y
  + Configure Docker to use systemd as the cgroup driver by creating and editing the

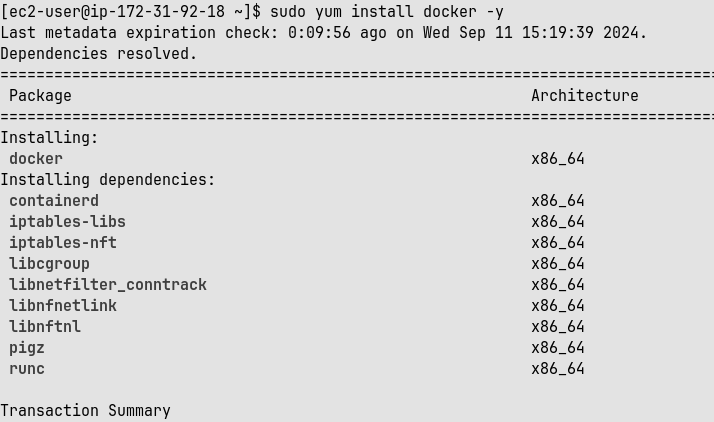
daemon.json file:

* + - Change directory to /etc/docker
    - Use the command cat <<EOF | sudo tee

/etc/docker/daemon.json followed by the JSON configuration details and end with EOF

* + Enable and restart Docker: sudo systemctl enable docker sudo systemctl daemon-reload sudo systemctl restart docker docker

-v



# Kubernetes Installation:

* + Disable SELinux before configuring kubelet: sudo setenforce 0 sudo sed

-i 's/^SELINUX=enforcing$/SELINUX=permissive/'

/etc/selinux/config

* + Add the Kubernetes repository and install Kubernetes components:
    - Use the command cat <<EOF | sudo tee

/etc/yum.repos.d/kubernetes.repo followed by the repository configuration details and end with EOF sudo yum update sudo yum install -y kubelet kubeadm kubectl

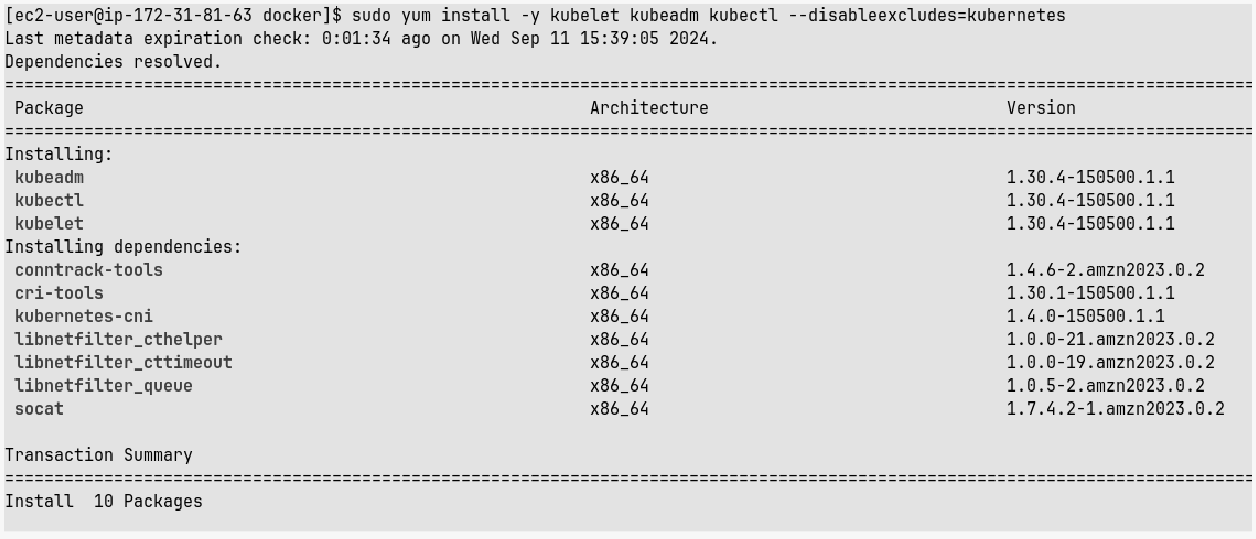
0--disableexcludes=kubernetes

* + Configure networking for bridging: sudo swapoff -a echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a

/etc/sysctl.conf sudo sysctl -p







# Master Node Setup:

* + Initialize the Kubernetes master node (perform only on the master machine): sudo kubeadm init --pod-network-cidr=10.244.0.0/16

--ignore-preflight-errors=all

* + After initialization, set up the Kubernetes configuration on the master node: mkdir -p

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

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

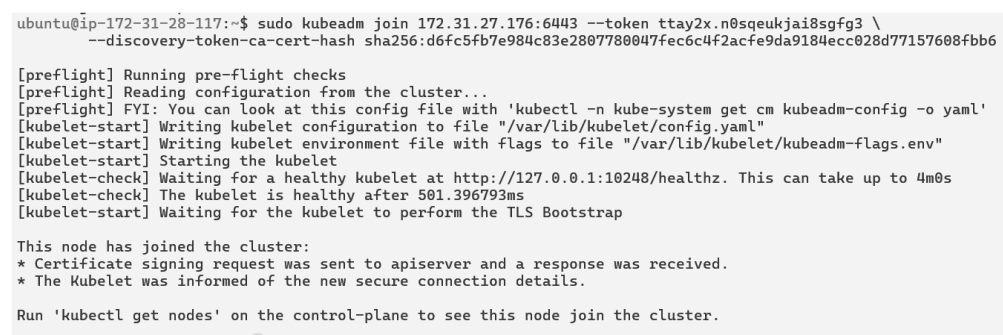
$HOME/.kube/config

* Save the generated join command from the output for worker nodes. This command is unique and specific to your cluster setup: kubeadm join 172.31.91.120:6443

--token

r8j60r.n1j6h0klbewvoka5\--discovery-token-ca-cert-hashsha256

:dd8426260174d673303aef17717f740772fcf7ee782245bc653eecf4a13 05da7

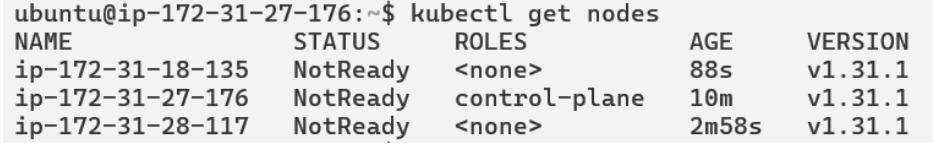


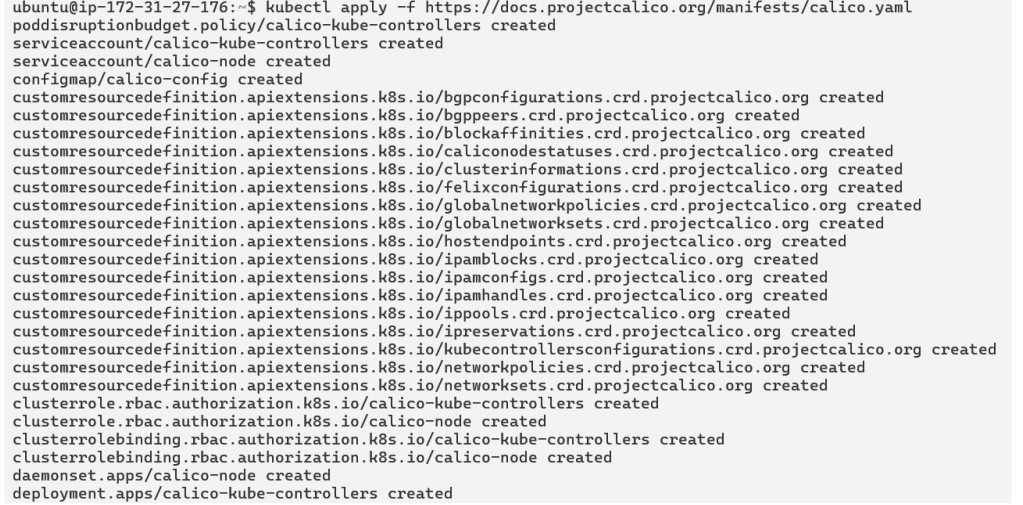
* Deploy the Flannel networking plugin to enable pod communication: kubectl apply

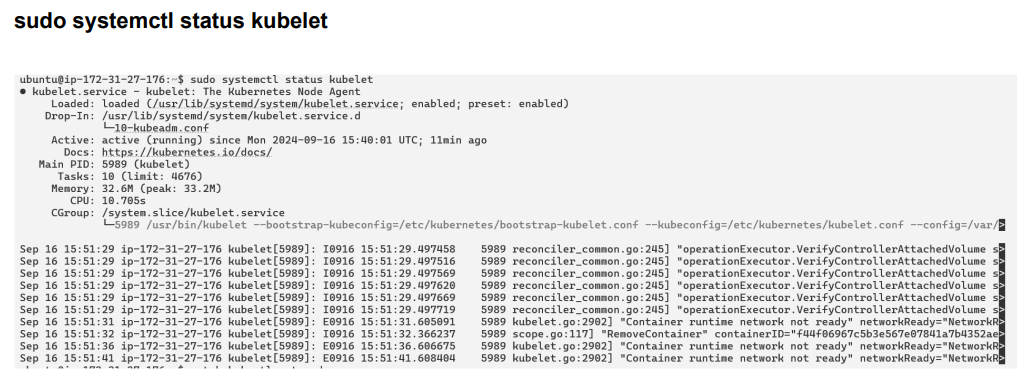
-f [https://raw.githubusercontent.com/coreos/flannel/master/Docu](https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml) [mentation/kube-flannel.yml](https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml)

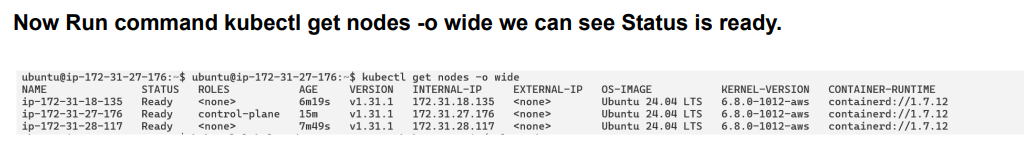
* Check the status of the pods to ensure they are running: kubectl get pods

--all-namespaces









# Worker Node Setup:

* + On each worker node, install the required package and configure kubelet: sudo yum install iproute-tc -y sudo systemctl enable kubelet sudo systemctl restart kubelet
  + Join the worker nodes to the Kubernetes cluster using the join command from the master node: kubeadm join 172.31.91.120:6443 --token r8j60r.n1j6h0klbewvoka5\--discovery-token-ca-cert-hashsha256

:dd8426260174d673303aef17717f740772fcf7ee782245bc653eecf4a13 05da7

# 

1. **Verify Node Status:**
   * On the master node, verify that the worker nodes have successfully joined the cluster by running: watch kubectl get nodes



# Conclusion:

Setting up the Kubernetes cluster involved several challenges. Network configuration issues initially hindered the deployment of the Flannel plugin, requiring open ports and a functional Kubernetes API server. Disabling SELinux and adjusting firewall rules were essential for proper communication between

components. Worker nodes experienced difficulties with the kubelet service, which needed to be correctly configured and restarted. Additionally, accurate copying of the join command, including the token and discovery-token-ca-cert-hash, was crucial for integrating worker nodes into the cluster. These issues underscored the need for precise configuration and troubleshooting to achieve a stable Kubernetes setup.