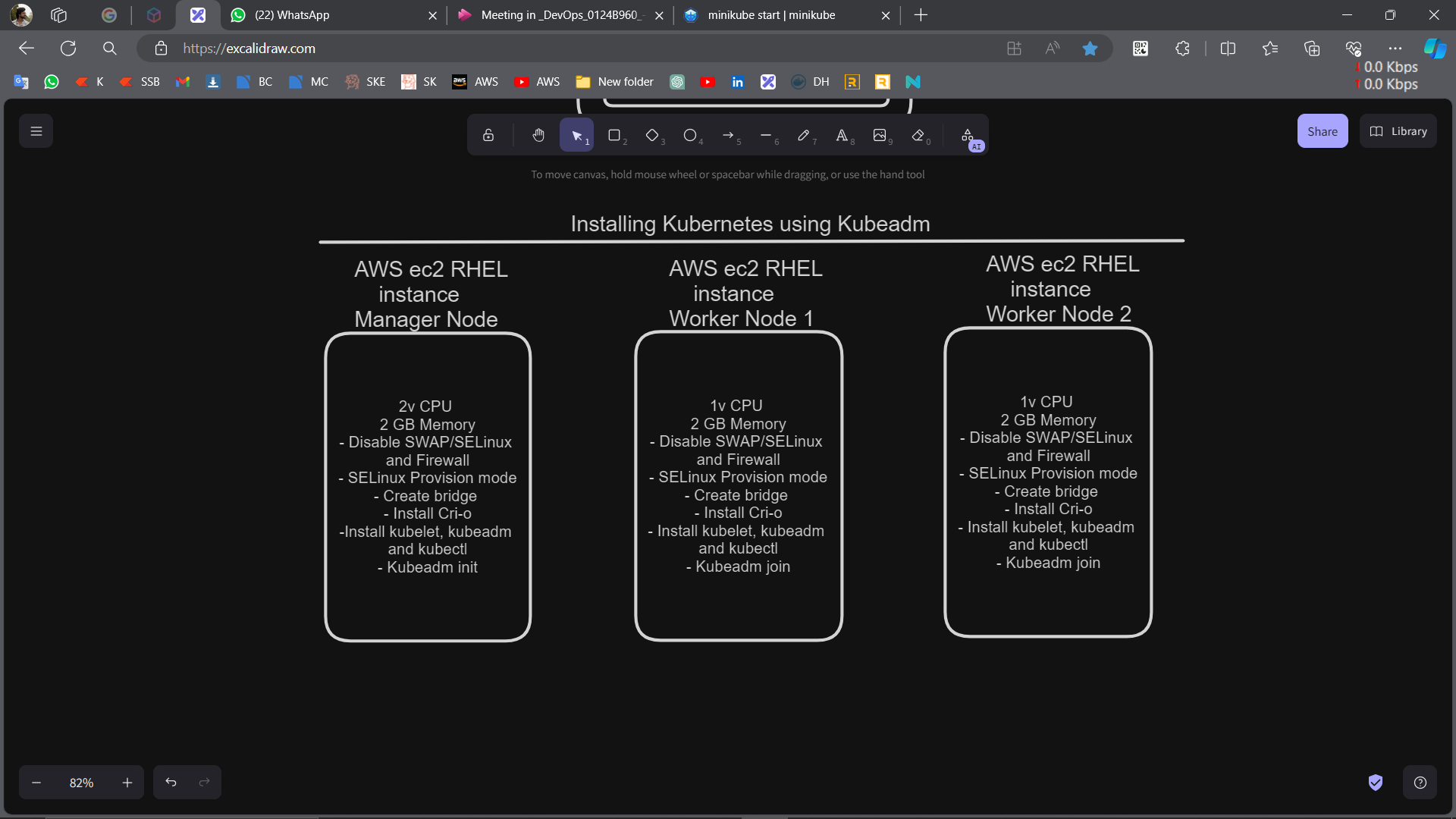
🚀 Project Update: Kubernetes Installation on AWS EC2 using kubeadm! 🚀

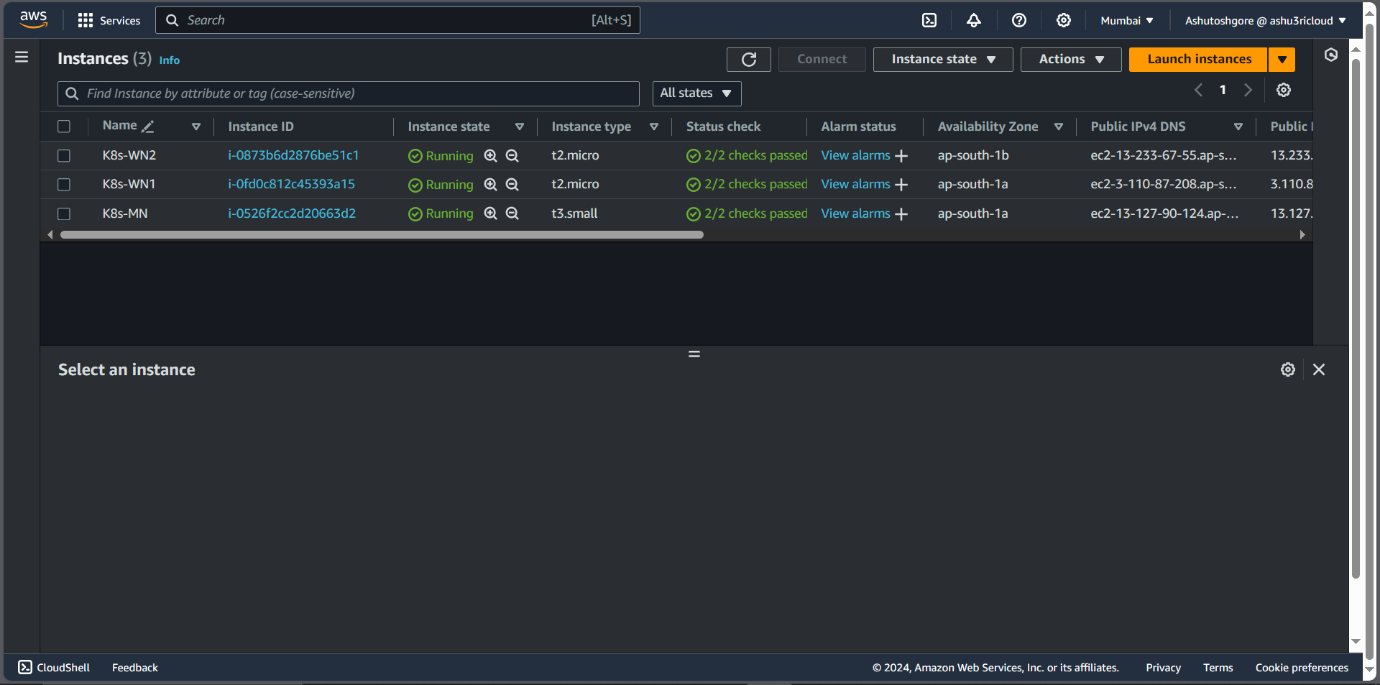
I'm excited to share the details of a recent project where I installed Kubernetes using kubeadm on three AWS EC2 RHEL instances. Kubernetes is a powerful tool for managing containerized applications, and kubeadm simplifies the setup process, making it easier to get a cluster up and running. Here's a breakdown of my setup:

🔹 Infrastructure:

Manager Node: 1 AWS EC2 RHEL t3.small instance (minimum requirement is 2 vCPU and 2 GiB Memory)

Worker Nodes: 2 AWS EC2 RHEL t2.micro instances (minimum requirement is 1 vCPU and 2 GiB Memory)





🔹 Installation Process:

1. Preparation on All Nodes:

* Switch to root and change the hostname.
* Disable swap to ensure Kubernetes can run correctly.
* Change SELinux from Enforcing to Provision.
* Install CRI-O as a Container Run Time.
* Set up the Kubernetes repository and install kubeadm, kubelet, and kubectl.

1. Manager Node:

* Initialize the Kubernetes control plane.
* Set up networking with a CNI plugin (e.g., Calico).

1. Worker Nodes:

* Join the worker nodes to the cluster using the token and hash from the manager node.

🔹 Outcome:

Successfully connected the two worker nodes to the manager node, forming a functional Kubernetes cluster.

🔹 Key Learnings:

Ensuring proper network configuration is critical for seamless node communication.

Managing Kubernetes resources with kubectl provides powerful control and insights.

This project has been a great learning experience, enhancing my skills in understanding Kubernetes architecture and the process of setting up a robust container orchestration environment. Looking forward to diving deeper into Kubernetes and exploring more advanced features!

Below I mention all the commands I run in each instance.

* + - Commands for all nodes: -
      * Sudo su-
      * Hostnamectl set-hostname <MNorWN1orWN2orWN3>
      * 1. yum update -y
      * 2. swapoff -a
      * 3. sed -i '/ swap / s/^\(.\*\)$/#/g' /etc/fstab
      * 4. getenforce
      * 5. setenforce 0
      * 6. sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
      * 7. cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

Overlay

br\_netfilter

EOF

* + - * 8. modprobe overlay
      * 9. modprobe br\_netfilter
      * 10. 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

* + - * 11. sysctl --system
      * We have to install CRI-O container runtime.
        + 1. VERSION=1.22
        + echo $VERSION
        + 2. curl -L -o /etc/yum.repos.d/devel:kubic:libcontainers:stable.repo https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable/CentOS\_8/devel:kubic:libcontainers:stable.repo
        + 3. curl -L -o /etc/yum.repos.d/devel:kubic:libcontainers:stable:crio:${VERSION}.repo https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable:cri-o:${VERSION}/CentOS\_8/devel:kubic:libcontainers:stable:cri-o:${VERSION}.repo
        + 4. dnf -y install cri-o cri-tools
        + 5. systemctl enable --now crio
        + 6. systemctl start crio
        + 7. systemctl status crio
      * We have to install Kubernetes.
        + 1. cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/

enabled=1

gpgcheck=1

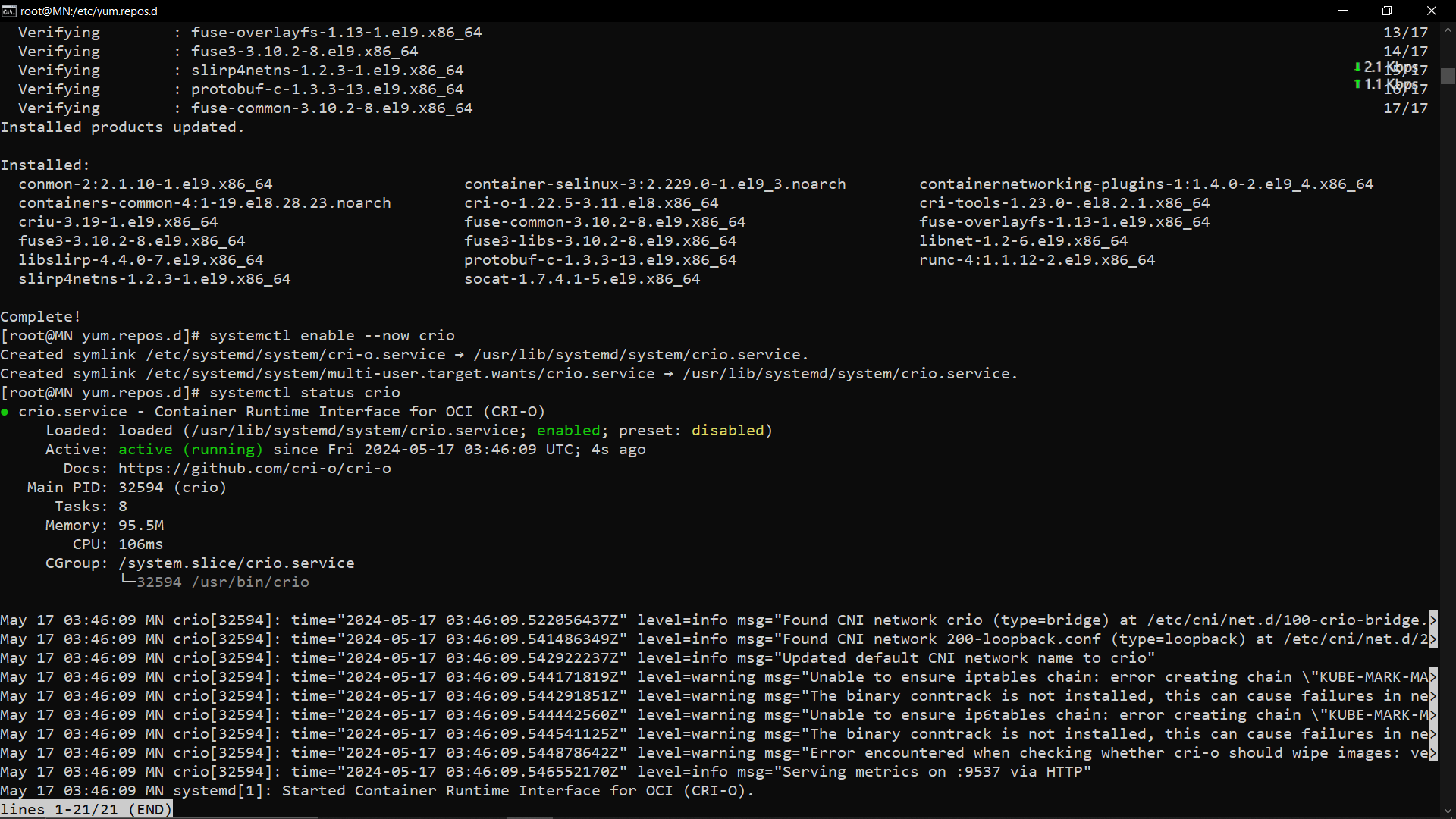
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key

EOF

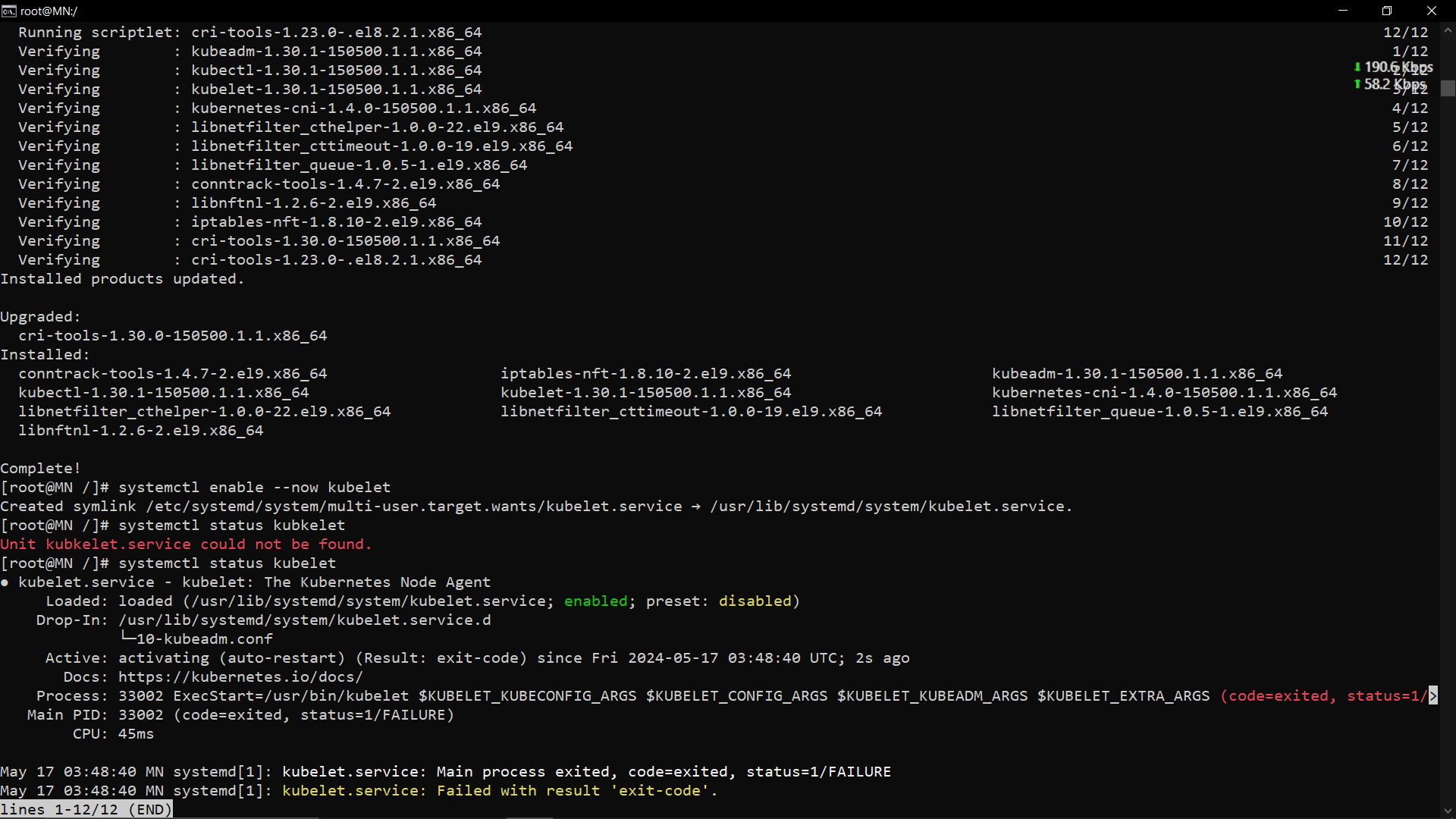
* + - * + 2. dnf install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
        + 3. systemctl enable --now kubelet
        + 4. systemctl start kubelet
        + 5. systemctl status kubelet
    - Commands for Master Node: -
      * + This below command no. 1 will make a node master node. And you need to put the instance public IP in command.
        + 1. sudo kubeadm init --control-plane-endpoint <Public\_IP>:6443
        + You’ll get an output in which you get the two tockens. One will make a node ‘control plane’ and another to make the nodes as a worker node
        + 1. mkdir -p $HOME/.kube
        + 2. cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
        + 3. chown $(id -u):$(id -g) $HOME/.kube/config
        + 4. export KUBECONFIG=/etc/kubernetes/admin.conf
* Commands for Worker Node: -
  + - * + Do this on workers only.
        + sudo kubeadm join <public\_IP>:6443 --token <random\_key> --discovery-token-ca-cert-hash sha256:<random\_key>

#Kubernetes #kubeadm #AWS #EC2 #RHEL #DevOps #CloudComputing #TechProjects

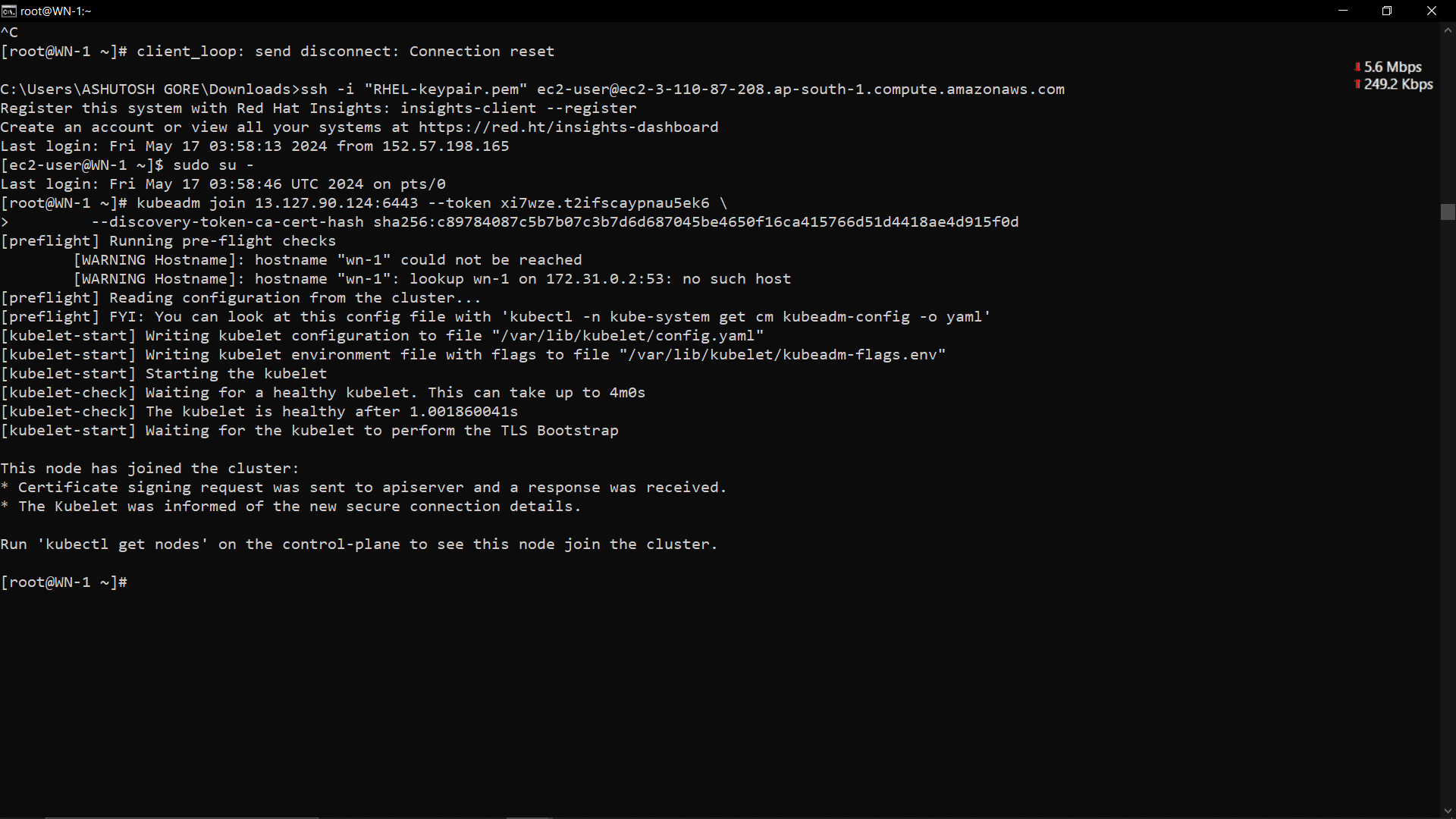
Crio installed



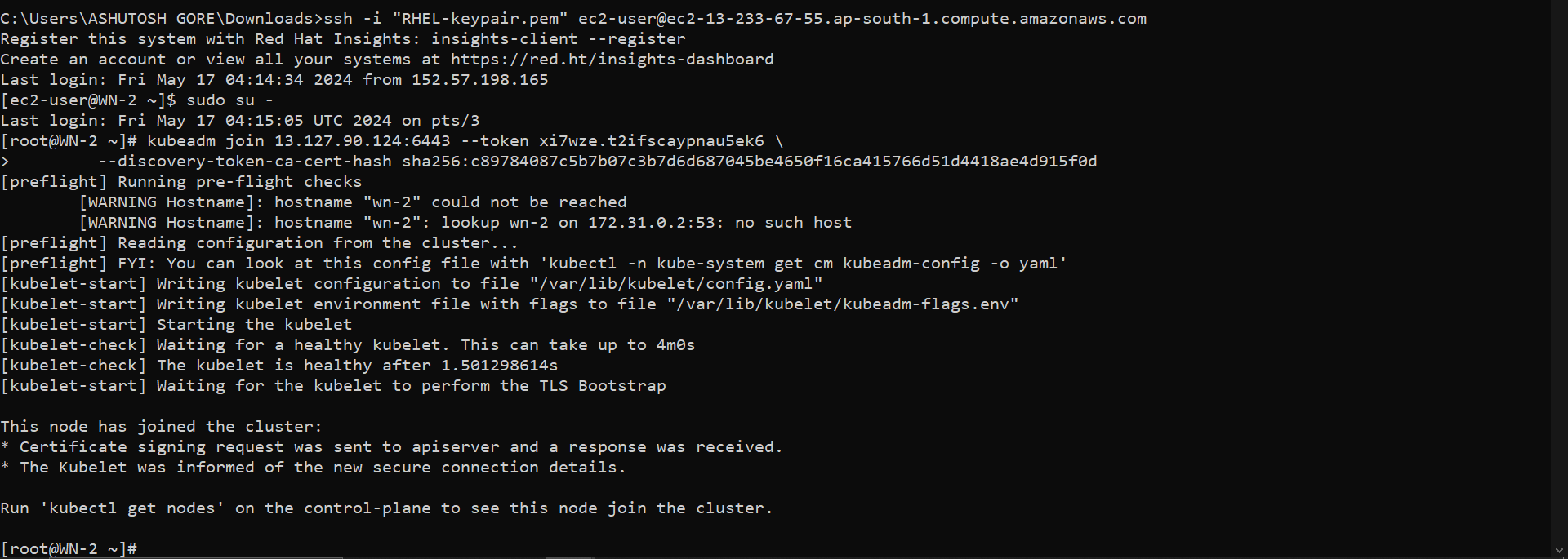
Installed kubelet



WN-1 connected



WN-2 connected



MN

