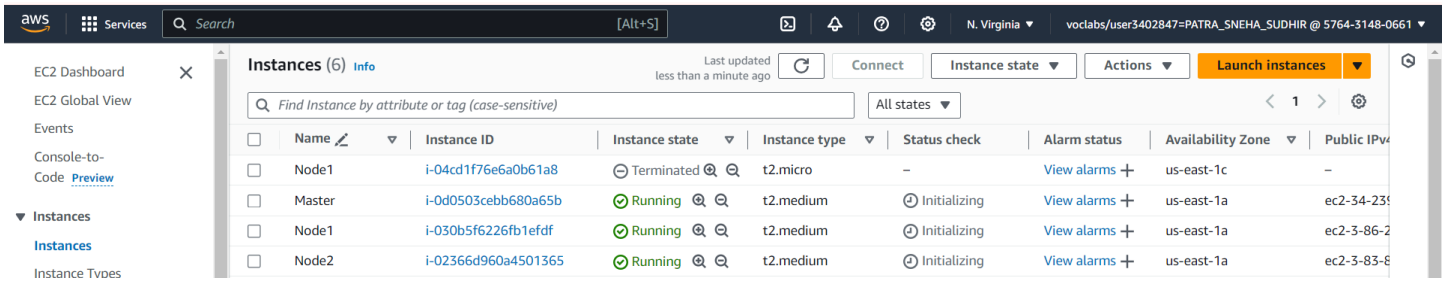
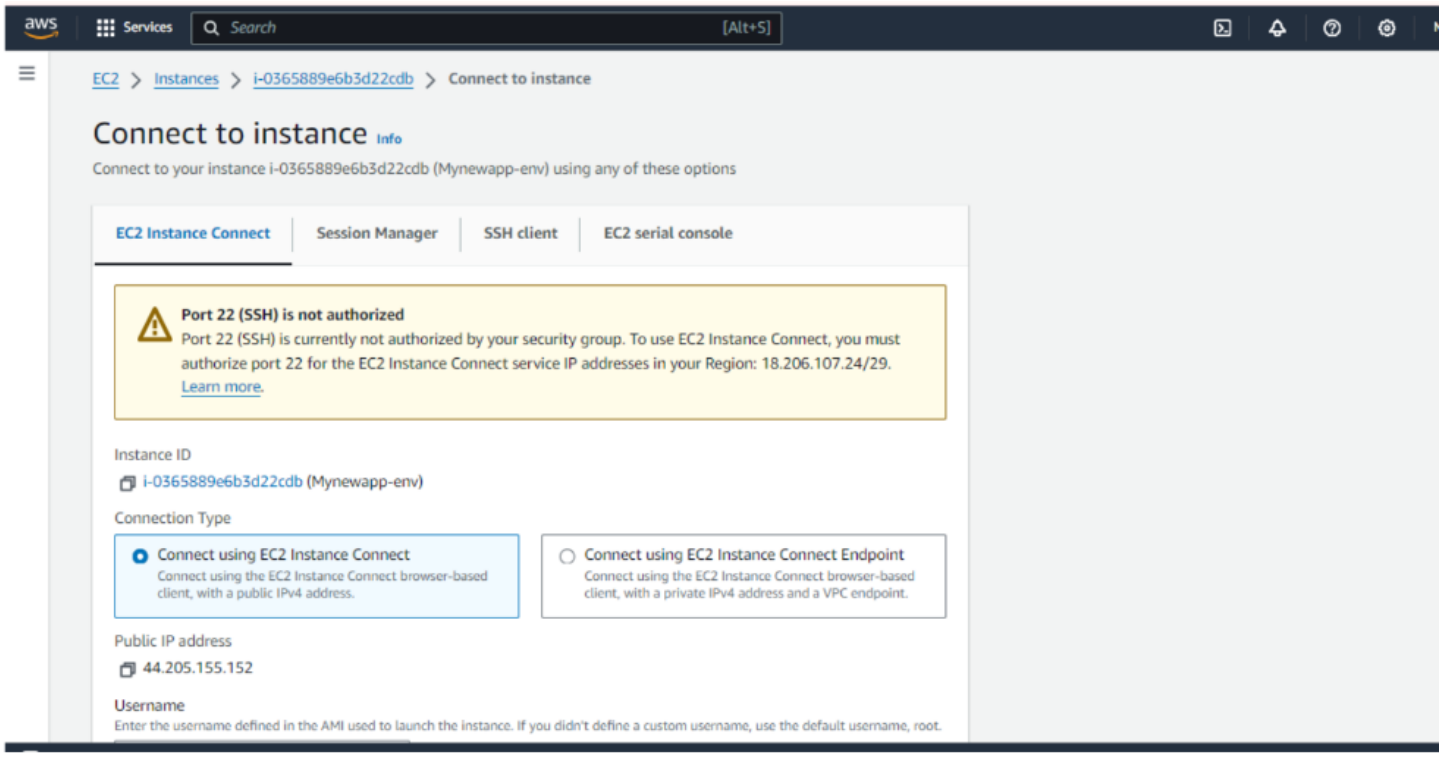
Advanced DevOps Experiment 3

Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

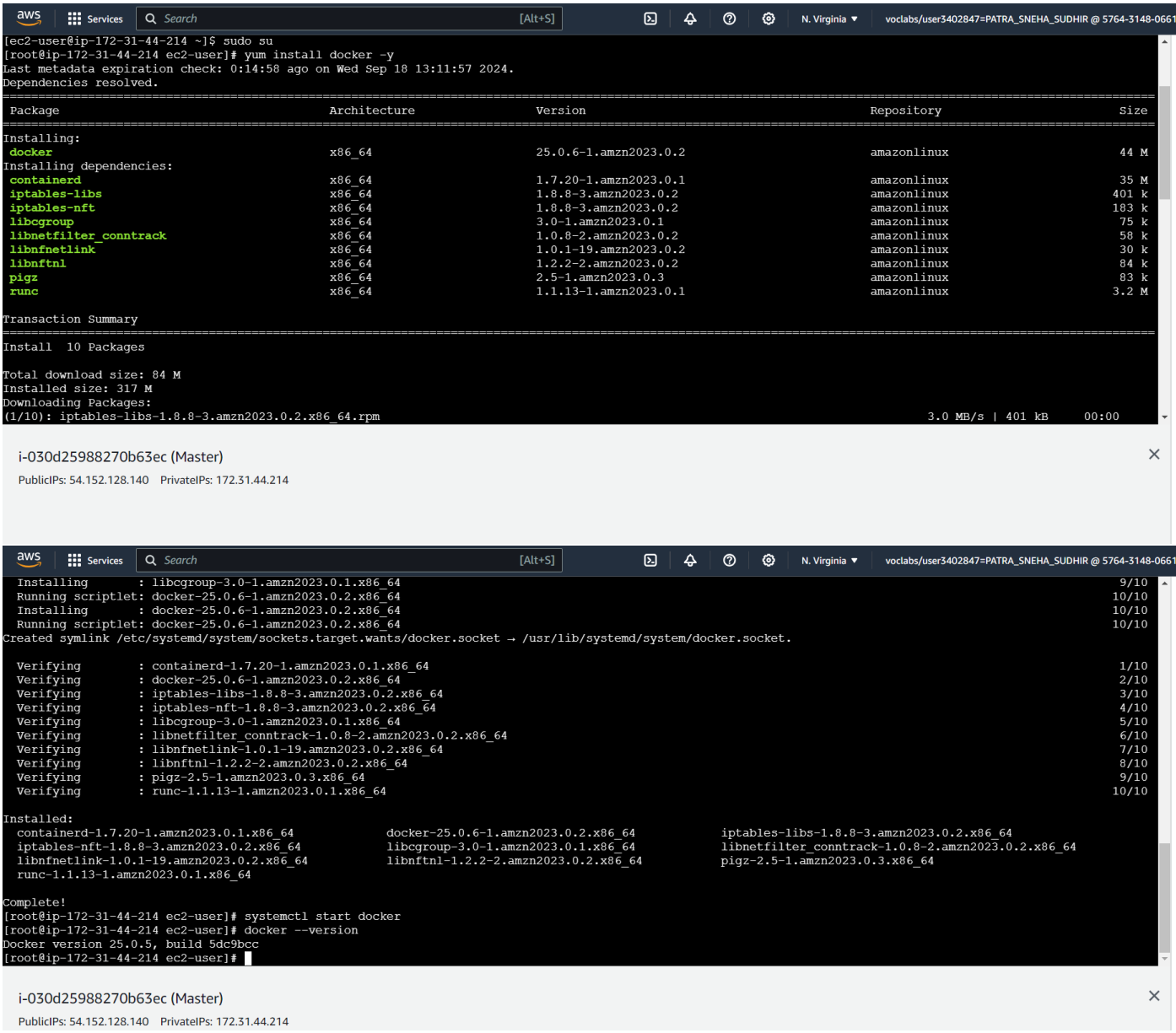
Step 1: Go to AWS Academy in services select EC2 and create 3 instance with and name them as master, node1, node2 and remember to select instance type as t2.medium.



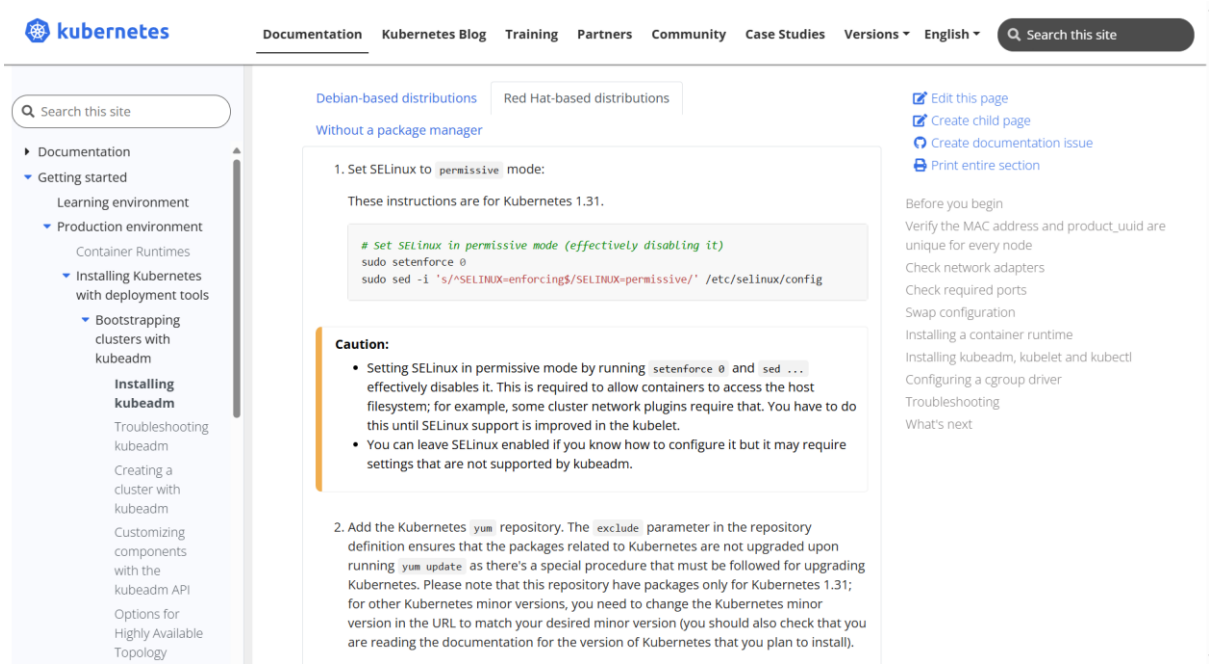


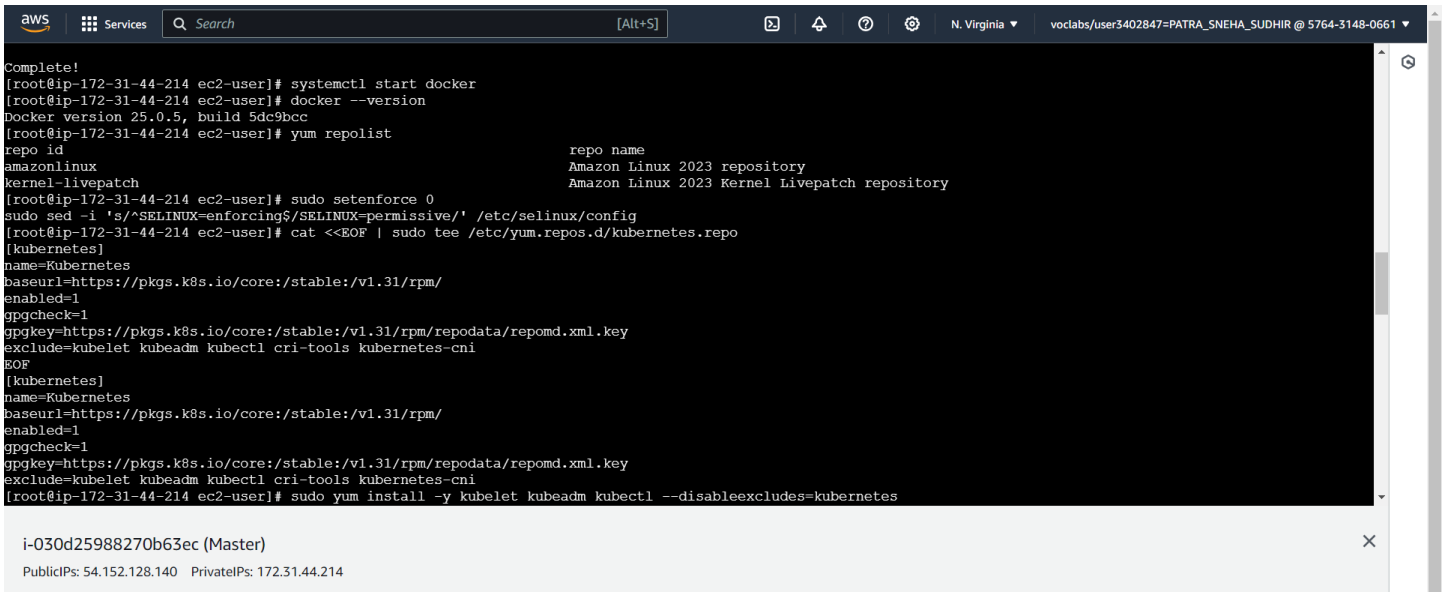
Step 2: Then, Select and connect each instance and run the following commands inside the console of each instance.

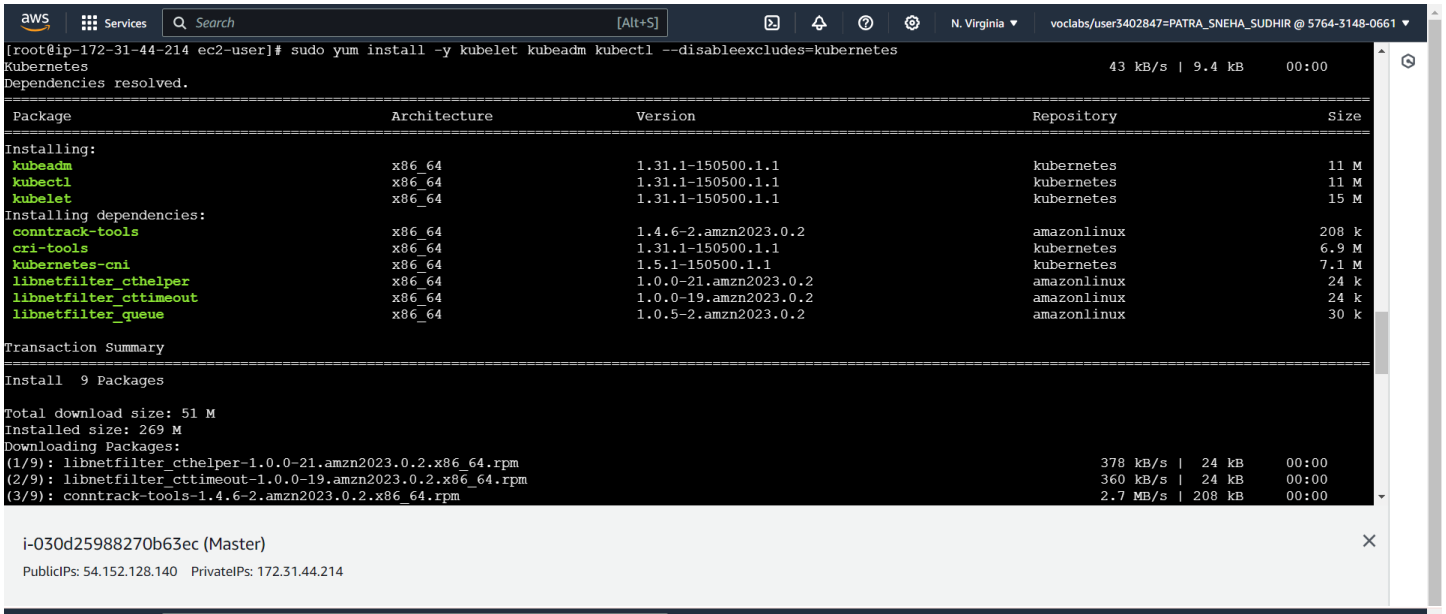
- sudo su
- yum install docker -y
- systemctl start docker
- docker –version
- yum repolist

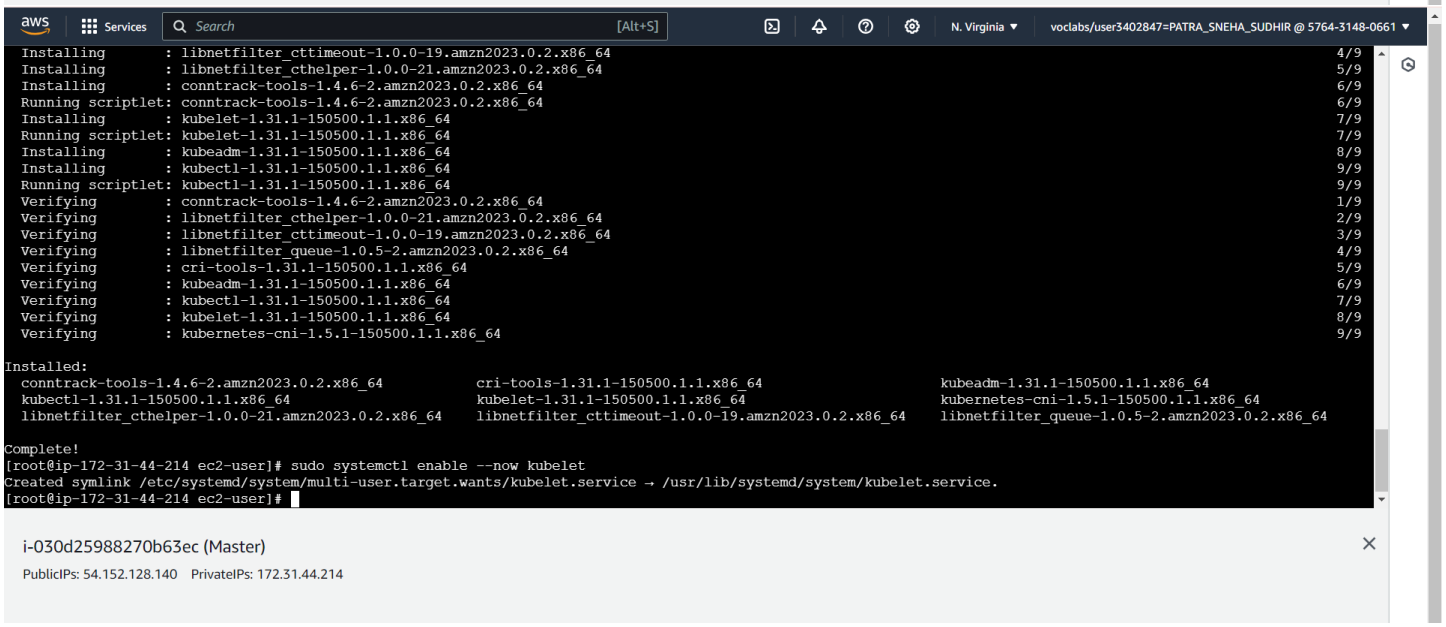


Step 3: Now, visit the following link https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/ and scroll down till you find Red-Hat and then select Red-Hat based distributions tab copy all the commands one by one in each console of instance.

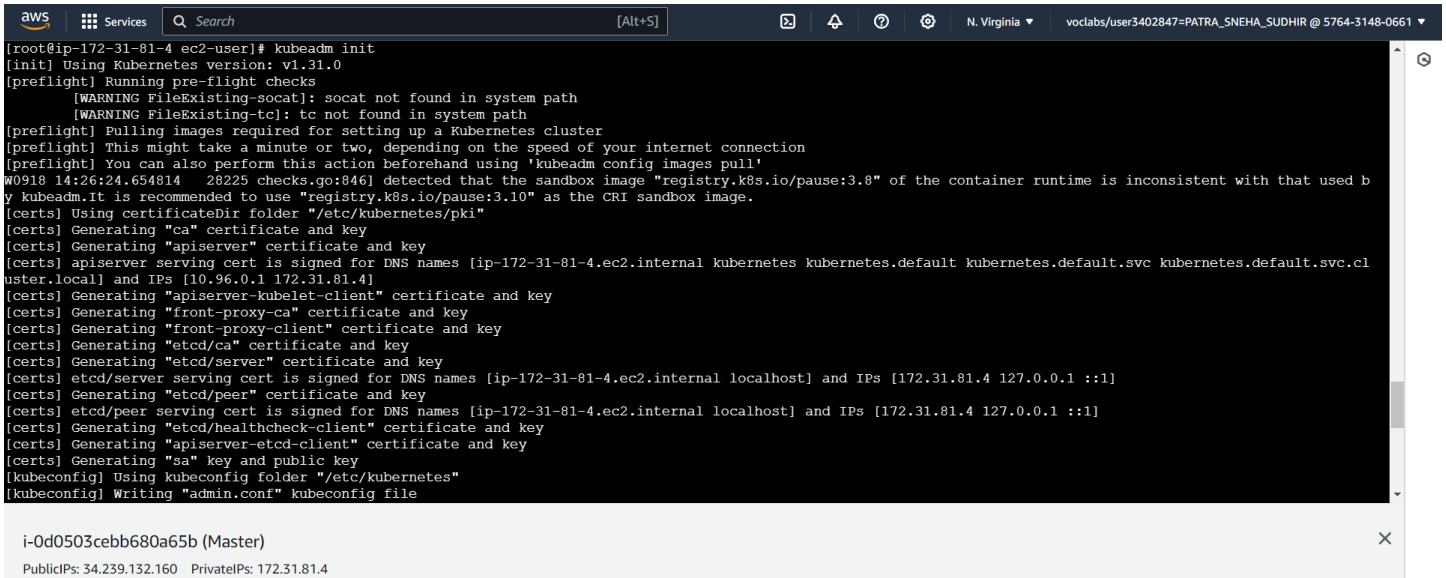

kubernetes
Documentation
Kubernetes Blog
Training
Partners
Community
Case Studies
Versions
English
Search this site
Search this site
Documentation
Getting started
Learning environment
Production environment
Container Runtimes
Installing Kubernetes with deployment tools
Bootstrapping clusters with kubeadm
Installing kubeadm
Troubleshooting kubeadm
Creating a cluster with kubeadm
Customizing components with the kubeadm API
Options for Highly Available Topology
Debian-based distributions
Red Hat-based distributions
Without a package manager
1. Set SELinux to permissive mode:
These instructions are for Kubernetes 1.31.
# Set SELinux in permissive mode (effectively disabling it)
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
Caution:
Setting SELinux in permissive mode by running setenforce 0 and sed ... effectively disables it. This is required to allow containers to access the host filesystem; for example, some cluster network plugins require that. You have to do this until SELinux support is improved in the kubelet.
You can leave SELinux enabled if you know how to configure it but it may require settings that are not supported by kubeadm.
2. Add the Kubernetes yum repository. The exclude parameter in the repository definition ensures that the packages related to Kubernetes are not upgraded upon running yum update as there's a special procedure that must be followed for upgrading Kubernetes. Please note that this repository have packages only for Kubernetes 1.31; for other Kubernetes minor versions, you need to change the Kubernetes minor version in the URL to match your desired minor version (you should also check that you are reading the documentation for the version of Kubernetes that you plan to install).
Edit this page
Create child page
Create documentation issue
Print entire section
Before you begin
Verify the MAC address and product_uuid are unique for every node
Check network adapters
Check required ports
Swap configuration
Installing a container runtime
Installing kubeadm, kubelet and kubectl
Configuring a cgroup driver
Troubleshooting
What's next


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3402847=PATRA_SNEHA_SUDHIR @ 5764-3148-0661
Complete!
[root@ip-172-31-44-214 ec2-user]# systemctl start docker
[root@ip-172-31-44-214 ec2-user]# docker --version
Docker version 25.0.5, build 5dc9bcc
[root@ip-172-31-44-214 ec2-user]# yum repolist
repo id repo name
amazonlinux Amazon Linux 2023 repository
kernel-livepatch Amazon Linux 2023 Kernel Livepatch repository
[root@ip-172-31-44-214 ec2-user]# sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[root@ip-172-31-44-214 ec2-user]# cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[root@ip-172-31-44-214 ec2-user]# sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
i-030d25988270b63ec (Master)
PublicIPs: 54.152.128.140 PrivateIPs: 172.31.44.214



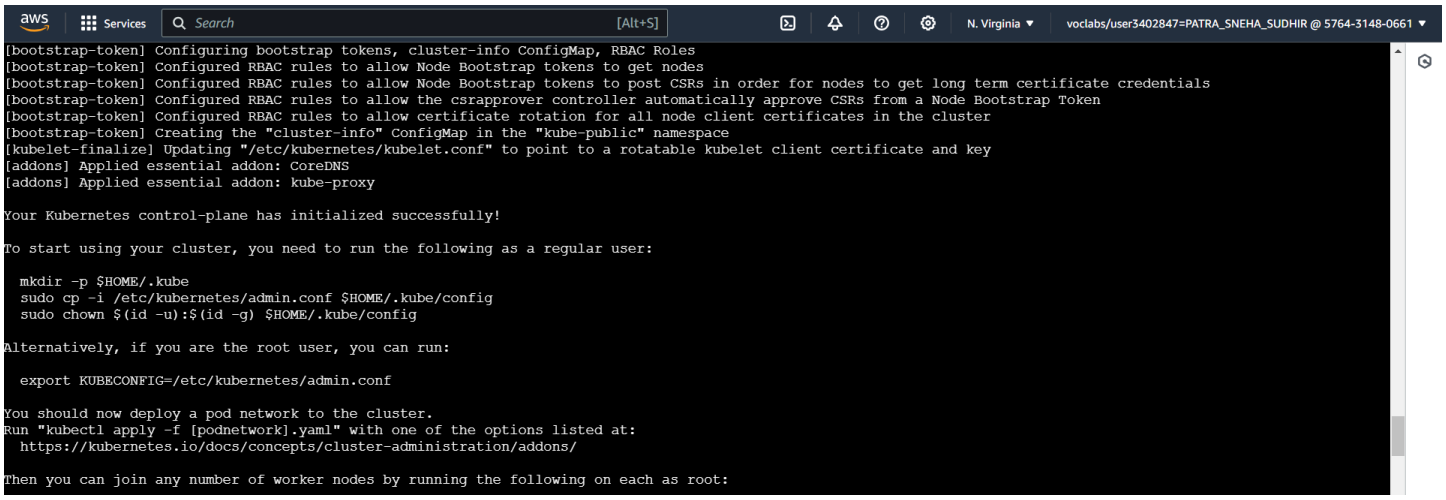


Step 4: Now, run the following command in the mater instance - kubeadm init

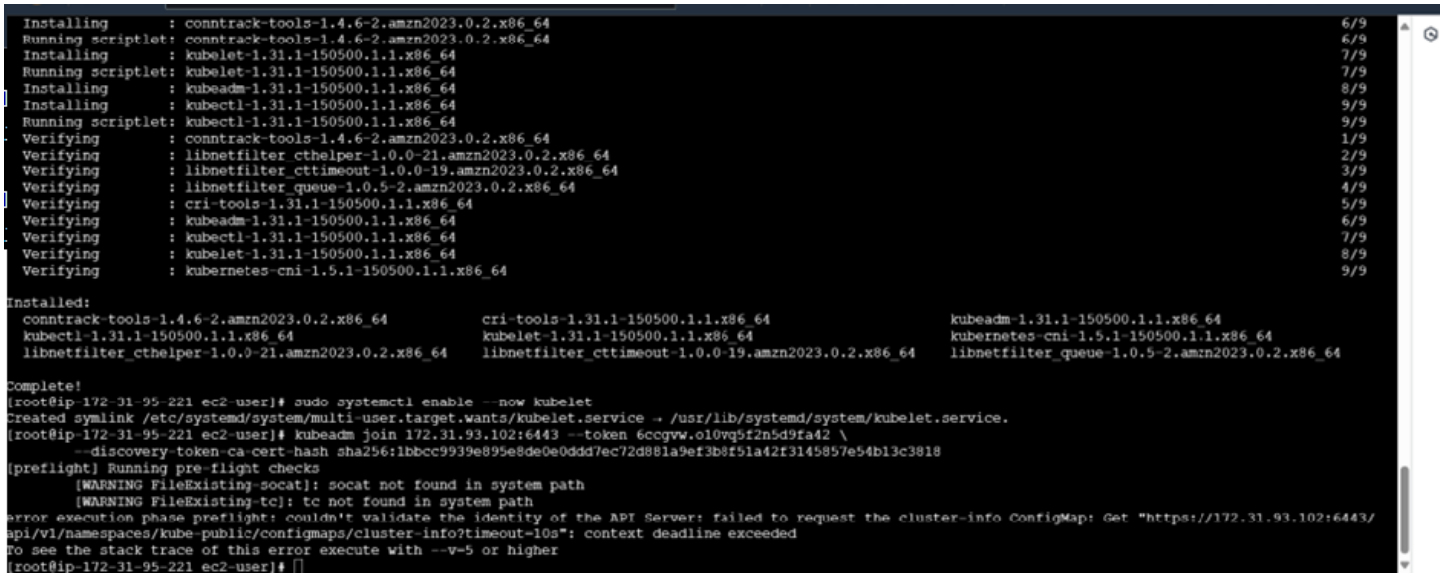


Step 5: Now, run the following commands in master instance's console –

a. mkdir -p $HOME/.kube
   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
   sudo chown $(id -u):$(id -g) $HOME/.kube/config
b. export KUBECONFIG=/etc/kubernetes/admin.conf
c. kubeadm join 172.31.93.102:6443 --token 6ccgvw.o10vq5f2n5d9fa42 \
   --discovery-token-ca-cert-hash
   sha256:1bbcc9939e895e8de0e0ddd7ec72d881a9ef3b8f51a42f3145857e54b13c3818



Step 6: Run this command in node1 and node2 -
kubeadm join 172.31.93.102:6443 --token 6ccgvw.o10vq5f2n5d9fa42 \
--discovery-token-ca-cert-hash sha256:1bbcc9939e895e8de0e0ddd7ec72d881a9ef3b8f51a42f3145857e54b13c3818



```
  Installing       : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                                  6/9
  Running scriptlet: conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                                  6/9
  Installing       : kubelet-1.31.1-150500.1.1.x86_64                                                             7/9
  Running scriptlet: kubelet-1.31.1-150500.1.1.x86_64                                                             7/9
  Installing       : kubeadm-1.31.1-150500.1.1.x86_64                                                             8/9
  Installing       : kubectl-1.31.1-150500.1.1.x86_64                                                             9/9
  Running scriptlet: kubectl-1.31.1-150500.1.1.x86_64                                                             9/9
  Verifying        : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                                  1/9
  Verifying        : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                                           2/9
  Verifying        : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64                                          3/9
  Verifying        : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64                                               4/9
  Verifying        : cri-tools-1.31.1-150500.1.1.x86_64                                                           5/9
  Verifying        : kubeadm-1.31.1-150500.1.1.x86_64                                                             6/9
  Verifying        : kubectl-1.31.1-150500.1.1.x86_64                                                             7/9
  Verifying        : kubelet-1.31.1-150500.1.1.x86_64                                                             8/9
  Verifying        : kubernetes-cni-1.5.1-150500.1.1.x86_64                                                       9/9

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64          cri-tools-1.31.1-150500.1.1.x86_64          kubeadm-1.31.1-150500.1.1.x86_64
  kubectl-1.31.1-150500.1.1.x86_64                     kubelet-1.31.1-150500.1.1.x86_64            kubernetes-cni-1.5.1-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64   libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64   libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64

Complete!
[root@ip-172-31-94-95 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-94-95 ec2-user]# kubeadm join 172.31.93.102:6443 --token 6ccgvw.ol0vq5f2n5d9fa42 \
        --discovery-token-ca-cert-hash sha256:1bbcc9939e095e8de0e0ddd7ec72d081a9ef3b8f51a42f3145857e54b13c3818
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
error execution phase preflight: couldn't validate the identity of the API Server: failed to request the cluster-info ConfigMap: Get "https://172.31.93.102:6443/
api/v1/namespaces/kube-public/configmaps/cluster-info?timeout=10s": context deadline exceeded
To see the stack trace of this error execute with --v=5 or higher
[root@ip-172-31-94-95 ec2-user]#
```

Step 7: Run the following command in master instance console -
kubectl get nodes

```
[root@ip-172-31-81-4 ec2-user]# kubectl get nodes
NAME                         STATUS     ROLES           AGE   VERSION
ip-172-31-81-4.ec2.internal  NotReady   control-plane   26m   v1.31.1
[root@ip-172-31-81-4 ec2-user]# kubectl get nodes
NAME                         STATUS     ROLES           AGE   VERSION
ip-172-31-81-4.ec2.internal  NotReady   control-plane   26m   v1.31.1
ip-172-31-94-95.ec2.internal NotReady   <none>          17s   v1.31.1
ip-172-31-95-221.ec2.internal NotReady  <none>          13s   v1.31.1
[root@ip-172-31-81-4 ec2-user]#
```