Name: Siya Dadheech

Assignment - 5

Kubernetes Installation

1. Create a Kubernetes cluster using minikube
   Step 1 Launch an EC2 Instance
   Step 2 Install Docker
   - sudo apt install -y docker.io
   - sudo usermod -aG docker $USER

   Step 3 Install Kubectl

   - curl -LO "https://dl.k8s.io/release/$(curl -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
   - chmod +x kubectl
   - sudo mv kubectl /usr/local/bin/
   - kubectl version –client

   Step 4 Install Minikube

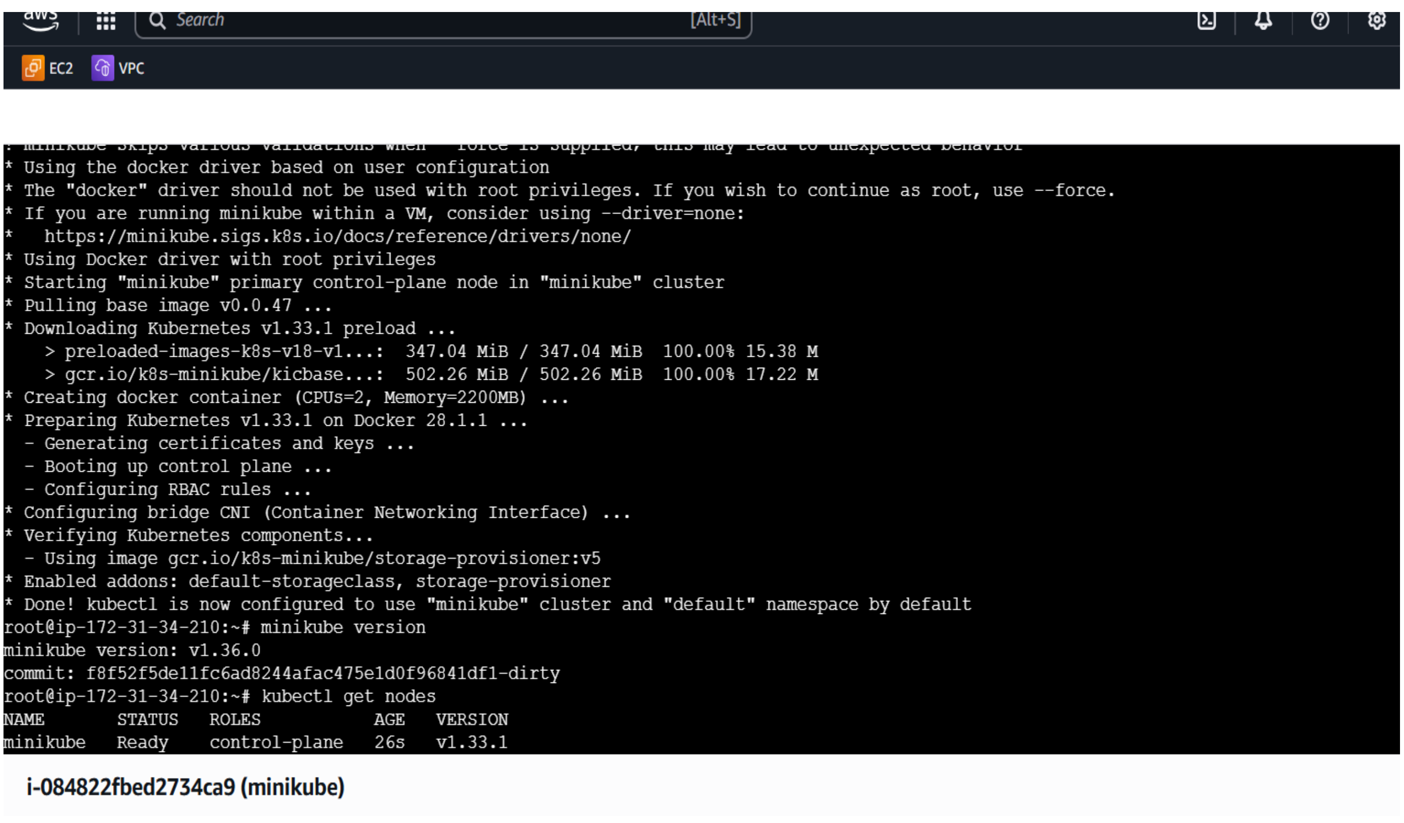
   - curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
   - sudo install minikube-linux-amd64 /usr/local/bin/minikube
   - minikube version

   Step 5 To start the minikube we need driver.

   - minikube start --driver=docker

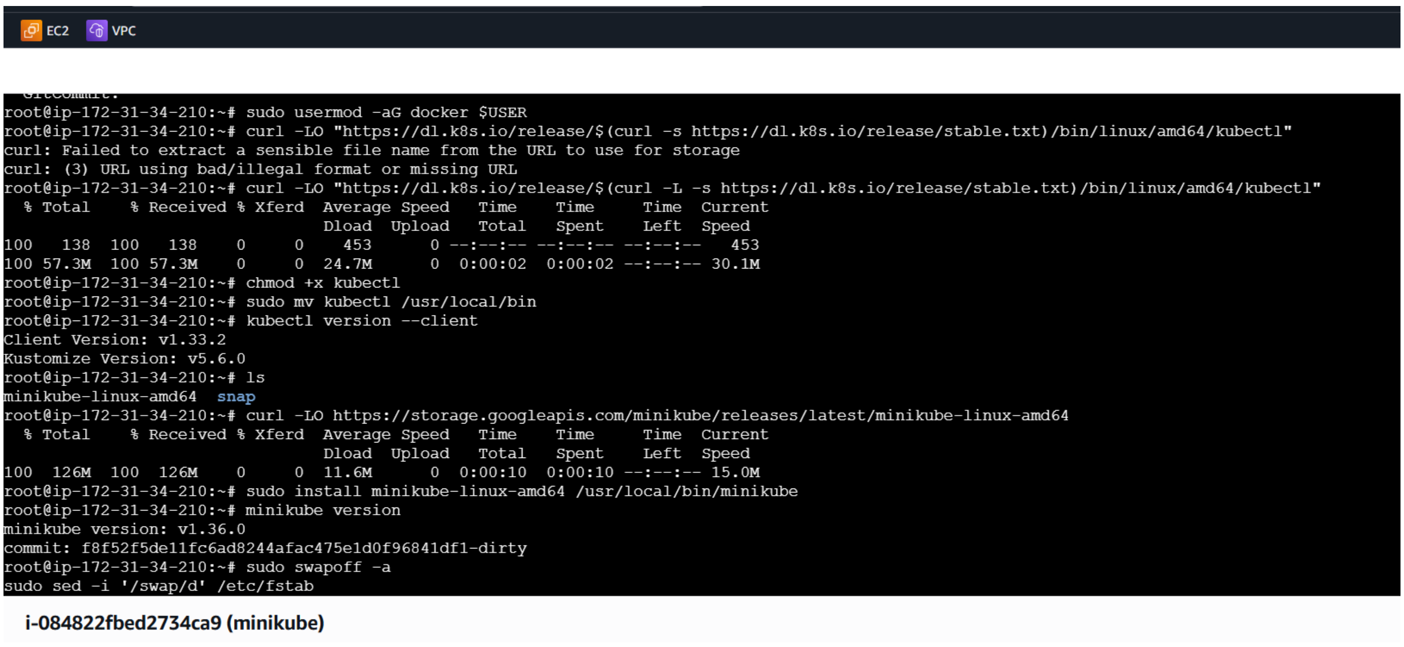
   Step 6 Verify Cluster

   - kubectl get nodes



```
* Using the docker driver based on user configuration
* The "docker" driver should not be used with root privileges. If you wish to continue as root, use --force.
* If you are running minikube within a VM, consider using --driver=none:
*   https://minikube.sigs.k8s.io/docs/reference/drivers/none/
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.47 ...
* Downloading Kubernetes v1.33.1 preload ...
    > preloaded-images-k8s-v18-v1...:  347.04 MiB / 347.04 MiB  100.00% 15.38 M
    > gcr.io/k8s-minikube/kicbase...:  502.26 MiB / 502.26 MiB  100.00% 17.22 M
* Creating docker container (CPUs=2, Memory=2200MB) ...
* Preparing Kubernetes v1.33.1 on Docker 28.1.1 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
root@ip-172-31-34-210:~# minikube version
minikube version: v1.36.0
commit: f8f52f5de11fc6ad8244afac475e1d0f96841df1-dirty
root@ip-172-31-34-210:~# kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   26s   v1.33.1
```

i-084822fbed2734ca9 (minikube)



```
root@ip-172-31-34-210:~# sudo usermod -aG docker $USER
root@ip-172-31-34-210:~# curl -LO "https://dl.k8s.io/release/$(curl -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
curl: Failed to extract a sensible file name from the URL to use for storage
curl: (3) URL using bad/illegal format or missing URL
root@ip-172-31-34-210:~# curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    453      0 --:--:-- --:--:-- --:--:--   453
100 57.3M  100 57.3M    0     0  24.7M      0  0:00:02  0:00:02 --:--:-- 30.1M
root@ip-172-31-34-210:~# chmod +x kubectl
root@ip-172-31-34-210:~# sudo mv kubectl /usr/local/bin
root@ip-172-31-34-210:~# kubectl version --client
Client Version: v1.33.2
Kustomize Version: v5.6.0
root@ip-172-31-34-210:~# ls
minikube-linux-amd64  snap
root@ip-172-31-34-210:~# curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  126M  100  126M    0     0  11.6M      0  0:00:10  0:00:10 --:--:-- 15.0M
root@ip-172-31-34-210:~# sudo install minikube-linux-amd64 /usr/local/bin/minikube
root@ip-172-31-34-210:~# minikube version
minikube version: v1.36.0
commit: f8f52f5de11fc6ad8244afac475e1d0f96841df1-dirty
root@ip-172-31-34-210:~# sudo swapoff -a
sudo sed -i '/swap/d' /etc/fstab
```

i-084822fbed2734ca9 (minikube)

2. Create a Kubernetes cluster using kubeadm
   Step 1: I Created one master and 2 worker node.
   Step 2: both on (worker and master node)

Step 1: Disable Swap on All Nodes

swapoff -a

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

Step 2: Enable IPv4 Packet Forwarding

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

net.ipv4.ip_forward = 1

EOF

Step 3: Verify IPv4 Packet Forwarding

sysctl net.ipv4.ip_forward

Step 4: Install containerd

# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

```
# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
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 enable --now containerd
```

Step 5: Install CNI Plugin

```
wget https://github.com/containernetworking/plugins/releases/download/v1.4.0/cni-plugins-linux-amd64-v1.4.0.tgz
mkdir -p /opt/cni/bin
tar Cxzvf /opt/cni/bin cni-plugins-linux-amd64-v1.4.0.tgz
```

Step 6: Forward IPv4 and Configure iptables

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter

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

sudo sysctl --system
sysctl net.bridge.bridge-nf-call-iptables net.bridge.bridge-nf-call-ip6tables net.ipv4.ip_forward
modprobe br_netfilter
sysctl -p /etc/sysctl.conf
```

Step 7: Modify containerd Configuration for systemd Support

```
Vim /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
```

```
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_cert = ""
  tcp_tls_key = ""
  uid = 0

[metrics]
  address = ""
  grpc_histogram = false

[plugins]

  [plugins."io.containerd.gc.v1.scheduler"]
    deletion_threshold = 0
    mutation_threshold = 100
    pause_threshold = 0.02
    schedule_delay = "0s"
    startup_delay = "100ms"
```

```
[plugins."io.containerd.grpc.v1.cri"]
  disable_apparmor = false
  disable_cgroup = false
  disable_hugetlb_controller = true
  disable_proc_mount = false
  disable_tcp_service = true
  enable_selinux = false
  enable_tls_streaming = false
  ignore_image_defined_volumes = false
  max_concurrent_downloads = 3
  max_container_log_line_size = 16384
  netns_mounts_under_state_dir = false
  restrict_oom_score_adj = false
  sandbox_image = "k8s.gcr.io/pause:3.5"
  selinux_category_range = 1024
  stats_collect_period = 10
  stream_idle_timeout = "4h0m0s"
  stream_server_address = "127.0.0.1"
  stream_server_port = "0"
  systemd_cgroup = false
  tolerate_missing_hugetlb_controller = true
  unset_seccomp_profile = ""

  [plugins."io.containerd.grpc.v1.cri".cni]
    bin_dir = "/opt/cni/bin"
    conf_dir = "/etc/cni/net.d"
    conf_template = ""
    max_conf_num = 1

  [plugins."io.containerd.grpc.v1.cri".containerd]
    default_runtime_name = "runc"
    disable_snapshot_annotations = true
    discard_unpacked_layers = false
    no_pivot = false
    snapshotter = "overlayfs"

    [plugins."io.containerd.grpc.v1.cri".containerd.default_runtime]
      base_runtime_spec = ""
  container_annotations = []
  pod_annotations = []
  privileged_without_host_devices = false
  runtime_engine = ""
  runtime_root = ""
  runtime_type = ""

  [plugins."io.containerd.grpc.v1.cri".containerd.default_runtime.options]

[plugins."io.containerd.grpc.v1.cri".containerd.runtimes]

  [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]
    base_runtime_spec = ""
    container_annotations = []
    pod_annotations = []
    privileged_without_host_devices = false
    runtime_engine = ""
    runtime_root = ""
    runtime_type = "io.containerd.runc.v2"

    [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
      BinaryName = ""
      CriuImagePath = ""
      CriuPath = ""
      CriuWorkPath = ""
      IoGid = 0
      IoUid = 0
      NoNewKeyring = false
      NoPivotRoot = false
      Root = ""
      ShimCgroup = ""
      SystemdCgroup = true

[plugins."io.containerd.grpc.v1.cri".containerd.untrusted_workload_runtime]
  base_runtime_spec = ""
  container_annotations = []
  pod_annotations = []
  privileged_without_host_devices = false
```

```
      runtime_engine = ""
      runtime_root = ""
      runtime_type = ""

      [plugins."io.containerd.grpc.v1.cri".containerd.untrusted_workload_runtime.options]

  [plugins."io.containerd.grpc.v1.cri".image_decryption]
    key_model = "node"

  [plugins."io.containerd.grpc.v1.cri".registry]
    config_path = ""

    [plugins."io.containerd.grpc.v1.cri".registry.auths]

    [plugins."io.containerd.grpc.v1.cri".registry.configs]

    [plugins."io.containerd.grpc.v1.cri".registry.headers]

    [plugins."io.containerd.grpc.v1.cri".registry.mirrors]

  [plugins."io.containerd.grpc.v1.cri".x509_key_pair_streaming]
    tls_cert_file = ""
    tls_key_file = ""

[plugins."io.containerd.internal.v1.opt"]
  path = "/opt/containerd"

[plugins."io.containerd.internal.v1.restart"]
  interval = "10s"

[plugins."io.containerd.metadata.v1.bolt"]
  content_sharing_policy = "shared"

[plugins."io.containerd.monitor.v1.cgroups"]
  no_prometheus = false

[plugins."io.containerd.runtime.v1.linux"]
  no_shim = false
```

```
    runtime = "runc"
    runtime_root = ""
    shim = "containerd-shim"
    shim_debug = false

  [plugins."io.containerd.runtime.v2.task"]
    platforms = ["linux/amd64"]

  [plugins."io.containerd.service.v1.diff-service"]
    default = ["walking"]

  [plugins."io.containerd.snapshotter.v1.aufs"]
    root_path = ""

  [plugins."io.containerd.snapshotter.v1.btrfs"]
    root_path = ""

  [plugins."io.containerd.snapshotter.v1.devmapper"]
    async_remove = false
    base_image_size = ""
    pool_name = ""
    root_path = ""

  [plugins."io.containerd.snapshotter.v1.native"]
    root_path = ""

  [plugins."io.containerd.snapshotter.v1.overlayfs"]
    root_path = ""

  [plugins."io.containerd.snapshotter.v1.zfs"]
    root_path = ""

[proxy_plugins]

[stream_processors]

  [stream_processors."io.containerd.ocicrypt.decoder.v1.tar"]
    accepts = ["application/vnd.oci.image.layer.v1.tar+encrypted"]
    args = ["--decryption-keys-path", "/etc/containerd/ocicrypt/keys"]
    env = ["OCICRYPT_KEYPROVIDER_CONFIG=/etc/containerd/ocicrypt/ocicrypt_keyprovider.conf"]
    path = "ctd-decoder"
    returns = "application/vnd.oci.image.layer.v1.tar"

  [stream_processors."io.containerd.ocicrypt.decoder.v1.tar.gzip"]
    accepts = ["application/vnd.oci.image.layer.v1.tar+gzip+encrypted"]
    args = ["--decryption-keys-path", "/etc/containerd/ocicrypt/keys"]
    env = ["OCICRYPT_KEYPROVIDER_CONFIG=/etc/containerd/ocicrypt/ocicrypt_keyprovider.conf"]
    path = "ctd-decoder"
    returns = "application/vnd.oci.image.layer.v1.tar+gzip"

[timeouts]
  "io.containerd.timeout.shim.cleanup" = "5s"
  "io.containerd.timeout.shim.load" = "5s"
  "io.containerd.timeout.shim.shutdown" = "3s"
  "io.containerd.timeout.task.state" = "2s"

[ttrpc]
  address = ""
  gid = 0
  uid = 0
```

Step 8: Restart containerd and Check the Status

```
sudo systemctl restart containerd && systemctl status containerd
```

On Master Node

Step 9: Install kubeadm, kubelet, and kubectl

```
sudo apt-get update

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


sudo mkdir -p -m 755 /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/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:/v1.30/deb/ /' |
sudo tee /etc/apt/sources.list.d/kubernetes.list
```

sudo apt-get update -y

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

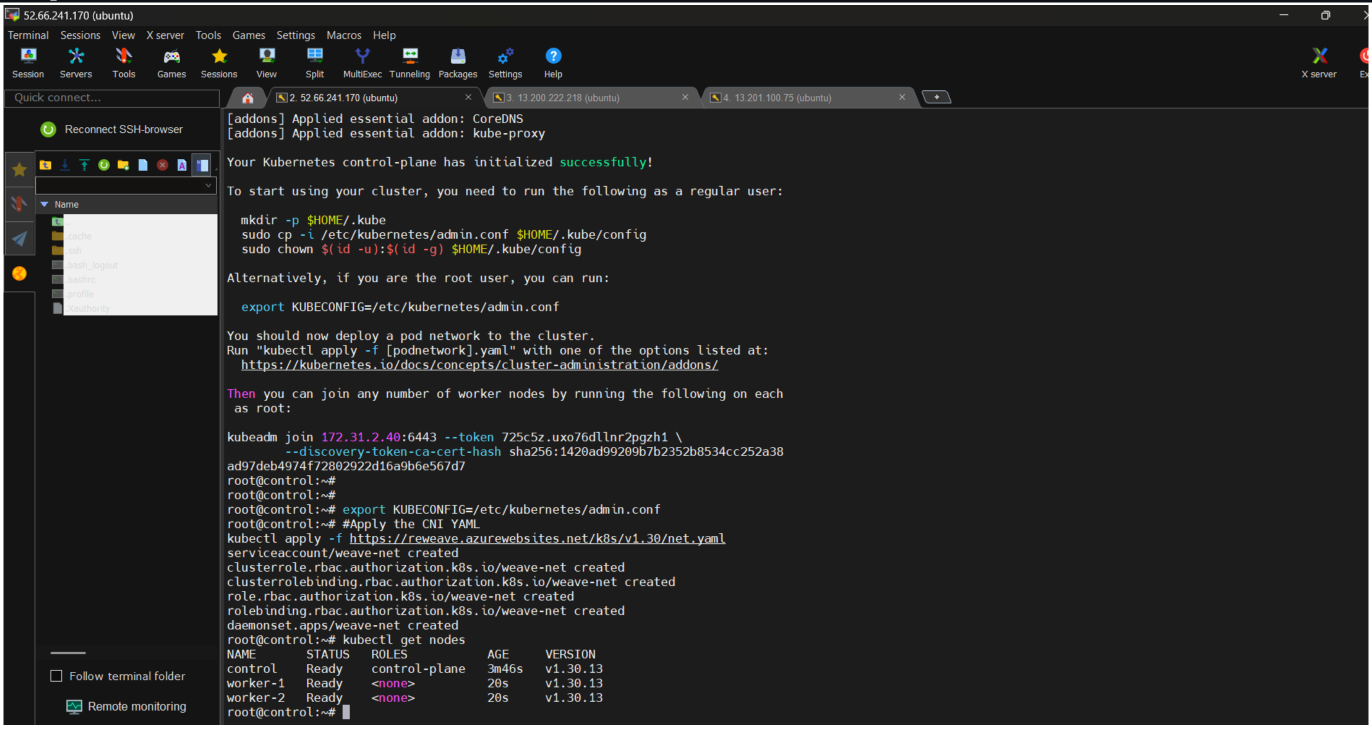
Step 10: Initialize the Cluster and Install CNI

sudo kubeadm config images pull

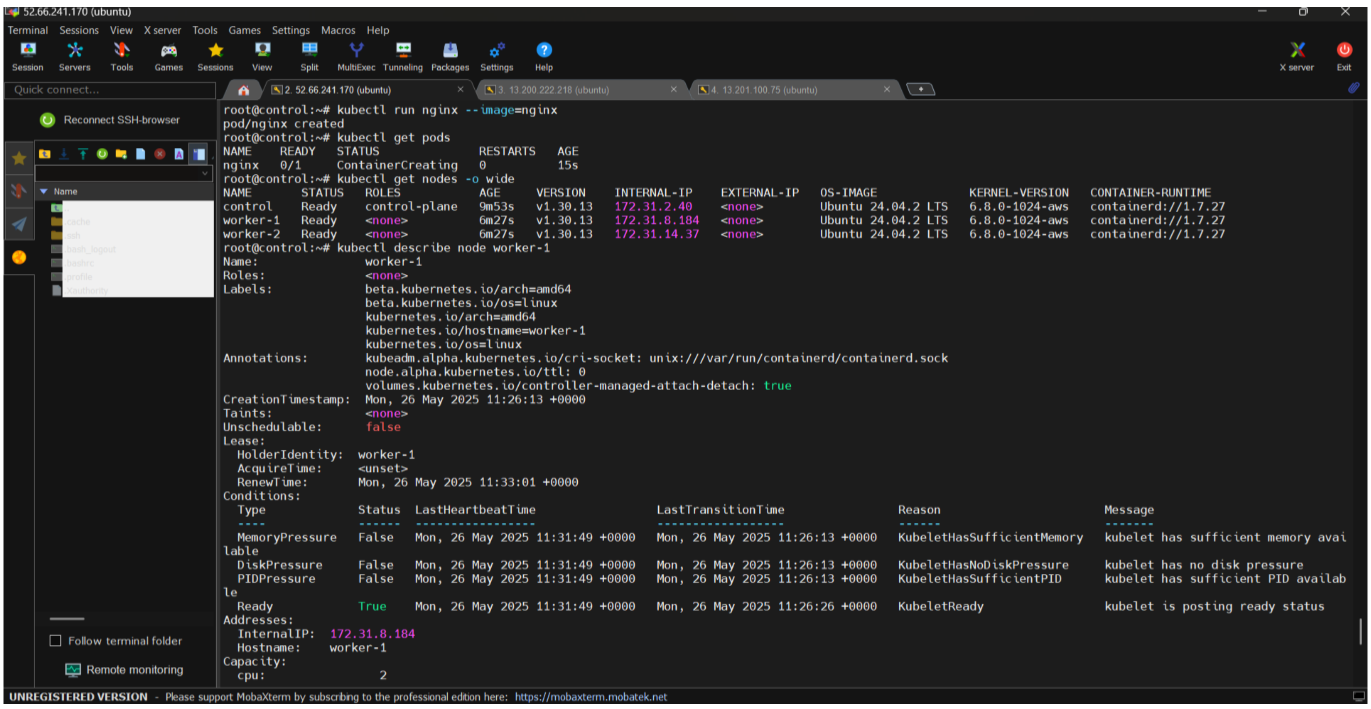
sudo kubeadm init

```
export KUBECONFIG=/etc/kubernetes/admin.conf
```

Step 11: Add the worker node to the cluster



```
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each
 as root:

kubeadm join 172.31.2.40:6443 --token 725c5z.uxo76dllnr2pgzh1 \
        --discovery-token-ca-cert-hash sha256:1420ad99209b7b2352b8534cc252a38
ad97deb4974f72802922d16a9b6e567d7
root@control:~#
root@control:~#
root@control:~# export KUBECONFIG=/etc/kubernetes/admin.conf
root@control:~# #Apply the CNI YAML
kubectl apply -f https://reweave.azurewebsites.net/k8s/v1.30/net.yaml
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
root@control:~# kubectl get nodes
NAME       STATUS   ROLES           AGE     VERSION
control    Ready    control-plane   3m46s   v1.30.13
worker-1   Ready    <none>          20s     v1.30.13
worker-2   Ready    <none>          20s     v1.30.13
root@control:~#
```



```
root@control:~# kubectl run nginx --image=nginx
pod/nginx created
root@control:~# kubectl get pods
NAME    READY   STATUS              RESTARTS   AGE
nginx   0/1     ContainerCreating   0          15s
root@control:~# kubectl get nodes -o wide
NAME       STATUS   ROLES           AGE     VERSION    INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION    CONTAINER-RUNTIME
control    Ready    control-plane   9m53s   v1.30.13   172.31.2.40    <none>        Ubuntu 24.04.2 LTS   6.8.0-1024-aws    containerd://1.7.27
worker-1   Ready    <none>          6m27s   v1.30.13   172.31.8.184   <none>        Ubuntu 24.04.2 LTS   6.8.0-1024-aws    containerd://1.7.27
worker-2   Ready    <none>          6m27s   v1.30.13   172.31.14.37   <none>        Ubuntu 24.04.2 LTS   6.8.0-1024-aws    containerd://1.7.27
root@control:~# kubectl describe node worker-1
Name:               worker-1
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=worker-1
                    kubernetes.io/os=linux
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: unix:///var/run/containerd/containerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Mon, 26 May 2025 11:26:13 +0000
Taints:             <none>
Unschedulable:      false
Lease:
  HolderIdentity:  worker-1
  AcquireTime:     <unset>
  RenewTime:       Mon, 26 May 2025 11:33:01 +0000
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----             ------  -----------------                 ------------------                ------                       -------
  MemoryPressure   False   Mon, 26 May 2025 11:31:49 +0000   Mon, 26 May 2025 11:26:13 +0000   KubeletHasSufficientMemory   kubelet has sufficient memory avai
lable
  DiskPressure     False   Mon, 26 May 2025 11:31:49 +0000   Mon, 26 May 2025 11:26:13 +0000   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure      False   Mon, 26 May 2025 11:31:49 +0000   Mon, 26 May 2025 11:26:13 +0000   KubeletHasSufficientPID      kubelet has sufficient PID availab
le
  Ready            True    Mon, 26 May 2025 11:31:49 +0000   Mon, 26 May 2025 11:26:26 +0000   KubeletReady                 kubelet is posting ready status
Addresses:
  InternalIP:  172.31.8.184
  Hostname:    worker-1
Capacity:
  cpu:                2
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net