```
resource "aws_instance" "k8s_master" {
  ami           = "ami-080e1f13689e07408" # Example AMI, replace with a Kubernetes supported one
  instance_type = "t3.xlarge"
  key_name      = aws_key_pair.ssh_key.key_name
  subnet_id     = aws_subnet.main.id  # Ensure this is the corrected subnet ID
  iam_instance_profile = aws_iam_instance_profile.ec2_instance_profile.name
  vpc_security_group_ids = [aws_security_group.k8s_master_sg.id]
  tags = {
    Name = "KubernetesMaster"
  }
  root_block_device {
    volume_size = 50  # Specify the size of the root volume in GiB
    volume_type = "gp2"
  }
user_data = base64encode(<<-EOSH
#!/bin/bash
# Update the system
sudo su
sudo apt-get update -y

sudo apt-get install -y iproute-tc
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
sudo apt-get -y install vim git curl wget
sudo swapoff -a
sudo sed -i '/swap/d' /etc/fstab
```

```
sudo mount -a

free -h

# Set up required modules

cat <<EOC | sudo tee /etc/modules-load.d/k8s.conf >/dev/null

overlay

br_netfilter

EOC


sudo modprobe overlay

sudo modprobe br_netfilter


  # sysctl params required by setup, params persist across reboots

sudo cat <<EOK | sudo tee /etc/sysctl.d/k8s.conf > /dev/null

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

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

net.ipv4.ip_forward                 = 1

EOK


# Apply sysctl params without reboot

sudo sysctl --system

sudo apt-get install -y lvm2



# Install crictl (the version should match with the Kubernetes version you are using)

curl                                                                                          -L
```

```
https://github.com/kubernetes-sigs/cri-tools/releases/download/v1.29.0/crictl-v1.29.0-linux-amd64.tar
.gz --output crictl.tar.gz
sudo tar zxvf crictl.tar.gz -C /usr/local/bin
rm -f crictl.tar.gz


sudo apt install -y containerd


# Configure the containerd cgroup driver to systemd
mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml
# Ensure the kubelet.service.d directory exists
sudo mkdir -p /etc/systemd/system/kubelet.service.d
sudo cat <<EOKUBE > /etc/systemd/system/kubelet.service.d/0-containerd-cgroup-driver.conf
[Service]
Environment="KUBELET_EXTRA_ARGS=--cgroup-driver=systemd"
EOKUBE



curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check kubectl.sha256
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
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:/v1.29/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:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
# Restart containerd to apply the configuration changes
sudo systemctl restart containerd
# Reload systemd, enable and start kubelet
sudo systemctl daemon-reload
sudo systemctl enable --now kubelet containerd



sudo kubeadm config images pull --cri-socket unix:///run/containerd/containerd.sock
# Initialize the Kubernetes cluster
PUBLIC_IP=$(hostname -I | awk '{print $1}')
sudo kubeadm init --apiserver-advertise-address=$PUBLIC_IP --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all --v=10

# Set up kubeconfig for the root user
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
export KUBECONFIG=$HOME/.kube/config
```

```
# Example readiness check loop for the kube-system namespace
for i in {1..10}; do
    kubectl get pods -n kube-system && break || sleep 15
done



# Apply network plugin if not already applied (idempotent operation)
echo "Applying Flannel CNI plugin..."
aws s3 cp s3://swarmskube/cogvlm_deployment.yml /tmp/cogvlm_deployment.yml
aws s3 cp s3://swarmskube/qwenvl_deployment.yml /tmp/qwenvl_deployment.yml
aws s3 cp s3://swarmskube/cogvlm_service.yml /tmp/cogvlm_service.yml
aws s3 cp s3://swarmskube/qwenvl_service.yml /tmp/qwenvl_service.yml
aws s3 cp s3://swarmskube/hpa.yml /tmp/hpa.yml
aws s3 cp s3://swarmskube/kubeflannel.yml.yml /tmp/kubeflannel.yml.yml
aws s3 cp s3://swarmskube/hpa.yml /tmp/router.yml
kubectl apply -f /tmp/kubeflannel.yml
kubectl apply -f /tmp/cogvlm_deployment.yml
kubectl apply -f /tmp/qwenvl_deployment.yml
kubectl apply -f /tmp/cogvlm_service.yml
kubectl apply -f /tmp/qwenvl_service.yml
kubectl apply -f /tmp/hpa.yml
kubectl apply -f /tmp/router.yml

curl https://baltocdn.com/helm/signing.asc | sudo apt-key add -
echo "deb https://baltocdn.com/helm/stable/debian/ all main" | sudo tee
```

```
/etc/apt/sources.list.d/helm-stable-debian.list
sudo apt-get update
sudo apt-get install -y helm
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update
helm install prometheus prometheus-community/prometheus --namespace monitoring --create-namespace
helm install grafana grafana/grafana --namespace monitoring --create-namespace
sudo kubeadm token create --print-join-command | sudo tee /tmp/k8s-join-command.sh
sudo apt-get install -y awscli
aws s3 cp /tmp/k8s-join-command.sh s3://swarmskube/k8s-join-command.sh
aws s3 cp $HOME/.kube/config s3://swarmskube/kubeconfig
EOSH
)
}

resource "aws_launch_template" "k8s_worker" {
  name_prefix = "k8s-worker-"
  image_id      = "ami-080e1f13689e07408" # Example AMI, replace with a Kubernetes supported one
  instance_type = "p3.2xlarge"
  key_name      = aws_key_pair.ssh_key.key_name

  iam_instance_profile {
    name = aws_iam_instance_profile.ec2_instance_profile.name
```

```
  }
  vpc_security_group_ids = [aws_security_group.k8s_worker_sg.id]

  block_device_mappings {
    device_name = "/dev/sda1"
    ebs {
      volume_size           = 50
      volume_type           = "gp3"
      delete_on_termination = true
    }
  }



  user_data = base64encode(<<-EOF
              #!/bin/bash
              # Download and execute the Kubernetes join command securely
sudo su
sudo apt-get update -y

sudo apt-get install -y iproute-tc
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
sudo apt-get -y install vim git curl wget
sudo swapoff -a
sudo sed -i '/swap/d' /etc/fstab
```

```
sudo mount -a

free -h

# Set up required modules

cat <<EOC | sudo tee /etc/modules-load.d/k8s.conf >/dev/null

overlay

br_netfilter

EOC


sudo modprobe overlay

sudo modprobe br_netfilter


  # sysctl params required by setup, params persist across reboots

sudo cat <<EOK | sudo tee /etc/sysctl.d/k8s.conf > /dev/null

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

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

net.ipv4.ip_forward                 = 1

EOK


# Apply sysctl params without reboot

sudo sysctl --system

sudo apt-get install -y lvm2



# Install crictl (the version should match with the Kubernetes version you are using)

curl -L
```

```
https://github.com/kubernetes-sigs/cri-tools/releases/download/v1.29.0/crictl-v1.29.0-linux-amd64.tar
.gz --output crictl.tar.gz
sudo tar zxvf crictl.tar.gz -C /usr/local/bin
rm -f crictl.tar.gz


sudo apt install -y containerd


# Configure the containerd cgroup driver to systemd
mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml
# Ensure the kubelet.service.d directory exists
sudo mkdir -p /etc/systemd/system/kubelet.service.d
sudo cat <<EOKUBE > /etc/systemd/system/kubelet.service.d/0-containerd-cgroup-driver.conf
[Service]
Environment="KUBELET_EXTRA_ARGS=--cgroup-driver=systemd"
EOKUBE



curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check kubectl.sha256
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
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:/v1.29/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:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
# Restart containerd to apply the configuration changes
sudo systemctl restart containerd
# Reload systemd, enable and start kubelet
sudo systemctl daemon-reload
sudo systemctl enable --now kubelet containerd
sudo apt-get install -y awscli


        while true; do
          if aws s3 ls "s3://swarmskube/k8s-join-command.sh"; then
            aws s3 cp s3://swarmskube/k8s-join-command.sh /tmp/k8s-join-command.sh
            chmod +x /tmp/k8s-join-command.sh
            /tmp/k8s-join-command.sh
            break
          else
            echo "Waiting for the master node to initialize..."
            sleep 30
          fi
        done
```

```
            EOF
  )



  tags = {
    resource_type = "instance"
    Name = "KubernetesWorker"
  }
}
```