**Lesson 03 Demo 02**

**Setting up Kubernetes Cluster on EC2 Instances**

**Objective:** To set up a Kubernetes cluster on Amazon EC2 instances, ensuring seamless communication between a master node and a worker node

**Tools required:** AWS Management Console

**Prerequisites:** None

Steps to be followed:

1. Set up the EC2 instances
2. Configure the Kubernetes master node
3. Connect the worker instance to the master node

**Step 1: Set up the EC2 instances**

1. Launch two EC2 instances using the Ubuntu image: one for the master node and one for the worker node. Ensure the instance type is **t2.medium** or higher.

A screenshot of a computer

Description automatically generated

|  |
| --- |
| **Note:** Refer to previous demos on how to launch an EC2 instance |

1. Install the necessary Kubernetes components using the commands given below:

**sudo apt update**

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

**sudo apt install docker.io -y**

**sudo systemctl enable --now docker**

**curl -fsSL "https://packages.cloud.google.com/apt/doc/apt-key.gpg" | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/kubernetes-archive-keyring.gpg**

**echo 'deb https://packages.cloud.google.com/apt kubernetes-xenial main' | sudo tee /etc/apt/sources.list.d/kubernetes.list**

**sudo apt update**

**sudo apt install kubeadm=1.20.0-00 kubectl=1.20.0-00 kubelet=1.20.0-00 -y**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Step 2: Configure the Kubernetes master node**

1. Run the following command to initialize the master node:

**sudo kubeadm init**

**A screenshot of a computer program

Description automatically generated**

1. Execute the following commands to set up local kubeconfig (both for root user and normal user):

**mkdir -p $HOME/.kube**

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

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

**A screenshot of a computer program

Description automatically generated**

1. Run the following command to deploy the weave network:

**kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml**

**A screenshot of a computer

Description automatically generated**

1. Run the following command to generate a token for joining the worker node:

**sudo kubeadm token create --print-join-command**

**A screenshot of a computer program

Description automatically generated**

|  |
| --- |
| **Note:** Save the **kubeadm** **join** command along with the generated token, as it will be required to connect the worker node |

1. Execute the following command to confirm the node’s status:

**kubectl get nodes**

**A screenshot of a computer

Description automatically generated**

You can see the master node is now ready and operational.

1. Open port **6443** in the security group to permit worker nodes to establish connections with the master node

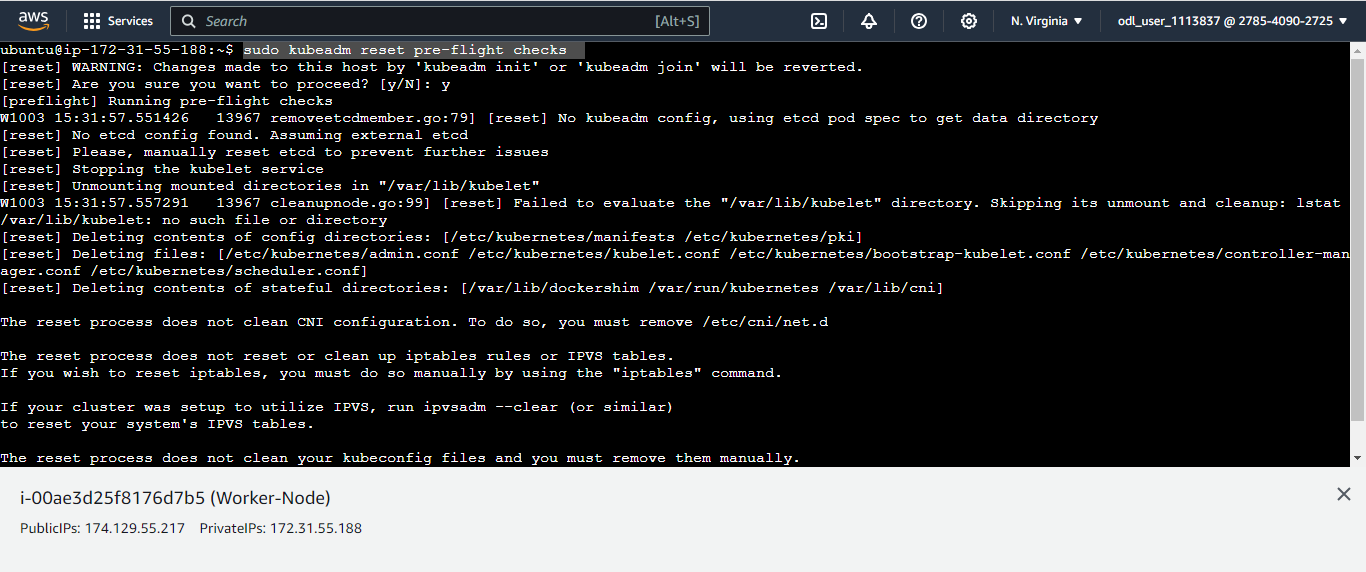
A screenshot of a computer

Description automatically generated

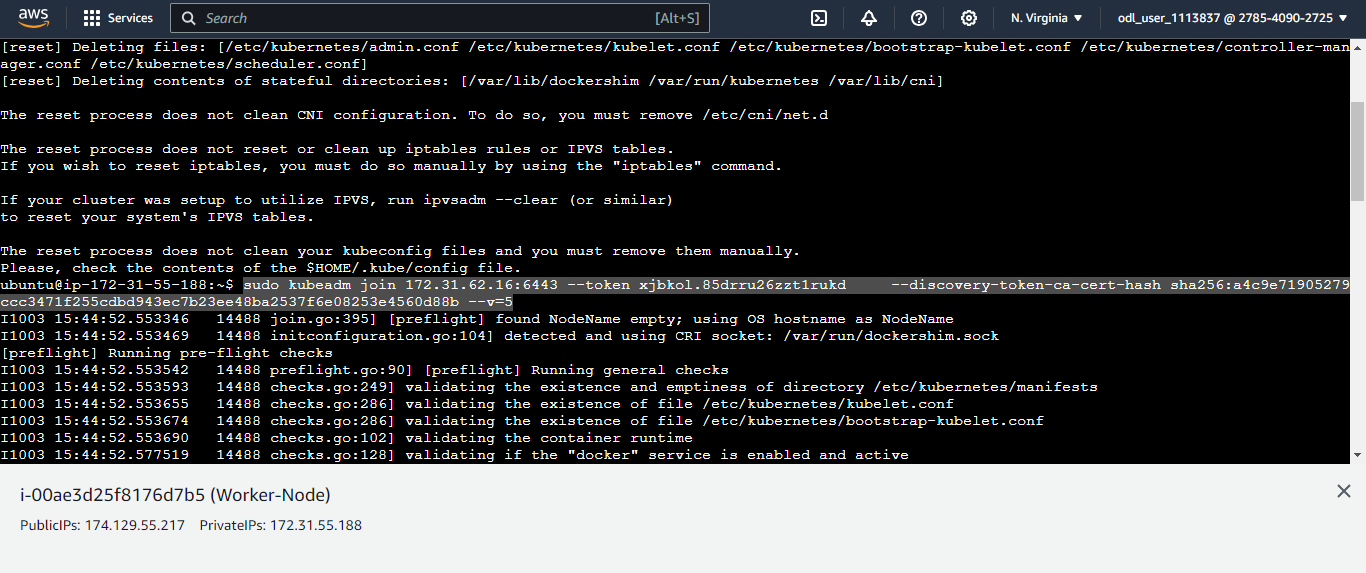
**Step 3: Connect the worker instance to the master node**

1. Execute the below command on the worker node to reset and perform pre-flight checks:

**sudo kubeadm reset pre-flight checks**

****

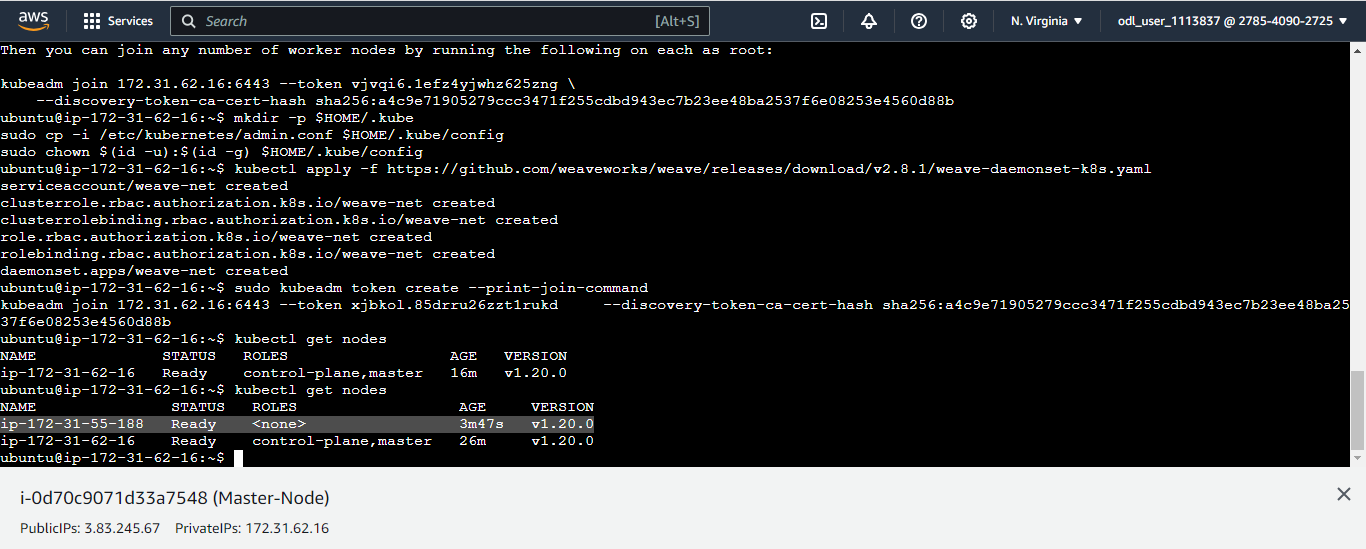
1. Paste the **kubeadm** **join** command you received from the master node and append **--v=5** to the end



|  |
| --- |
| **Note:** Make sure to use **sudo** before the command |

1. Return to the master node and run the following command to check if the worker node has successfully joined the cluster:

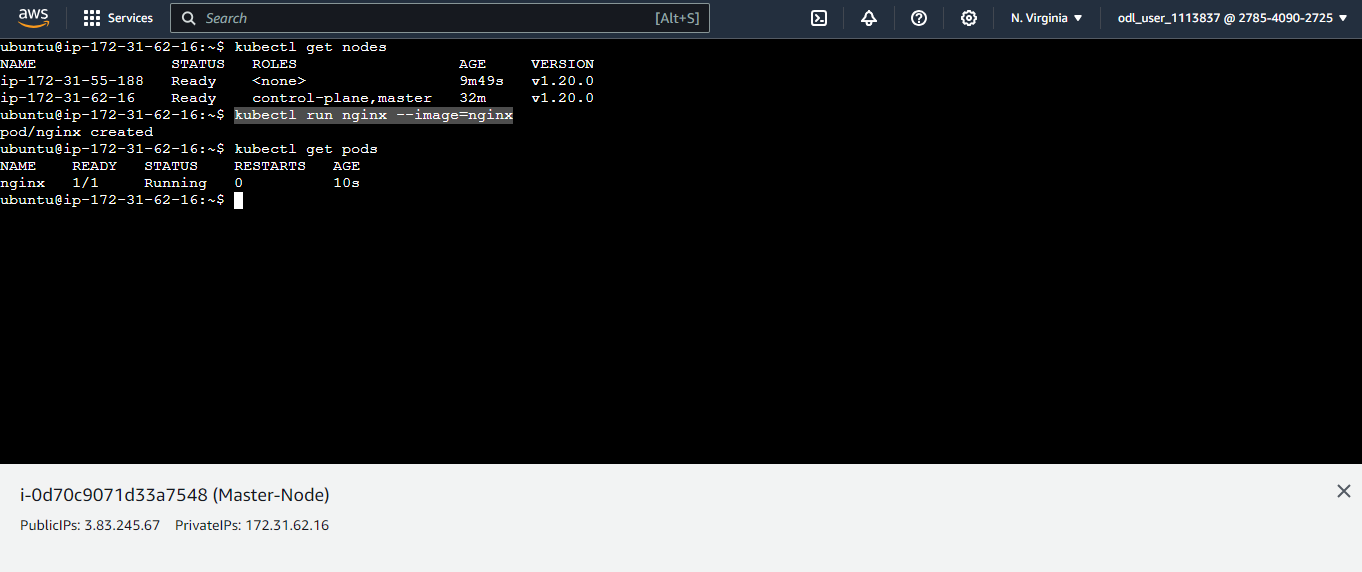
**kubectl get nodes**



You can see that the worker node has been successfully integrated into the cluster.

1. To deploy a sample pod for testing purposes, execute the given command:

**kubectl run nginx --image=nginx**



By following these steps, you have successfully established a Kubernetes cluster with a master node and a worker node on EC2 instances.