**SHRAVANI ANIL PATIL** **D15A-38**

**Advanced DevOps Lab**
**Experiment:4**

**Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.**

**Overview of Kubernetes and Kubectl**

**What is Kubernetes?**

Kubernetes, often referred to as K8s, is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. Originally developed by Google, it has become the industry standard for managing container workloads due to its flexibility and robust features.

**Core Concepts of Kubernetes**

Containers: These are lightweight, portable packages that include everything needed to run an application, ensuring consistency across different environments.

Pods: The smallest deployable units in Kubernetes, pods can contain one or more containers that share storage and network resources.

Nodes: A node is a worker machine in the Kubernetes cluster that runs at least one pod. Nodes can be either physical or virtual machines.

Clusters: A cluster comprises multiple nodes that run containerized applications. The control plane manages the cluster's state.

Services: Services provide stable endpoints for accessing pods and facilitate load balancing and service discovery.

Deployments: A deployment manages the lifecycle of pods, allowing users to specify the number of replicas and facilitating rolling updates and rollbacks.

**Architecture of Kubernetes**

**Kubernetes follows a client-server architecture consisting of:**

Control Plane: Manages the cluster and includes components like the API server (the front-end for the control plane), scheduler (assigns pods to nodes), controller manager (regulates cluster state), and etcd (a distributed key-value store for cluster data).

Worker Nodes: Each node runs components like kubelet (ensures containers are running), kube-proxy (manages network communication), and a container runtime (e.g., Docker).

**Role of Kubectl in Kubernetes**

**What is Kubectl?**

Kubectl is the command-line interface used to interact with the Kubernetes API server. It enables users to manage resources within a Kubernetes cluster effectively.

**Configuration Files**

Configuration files are essential for defining how resources should be created or modified within Kubernetes. Users can employ declarative configurations (using YAML/JSON files) or imperative commands directly in the terminal.

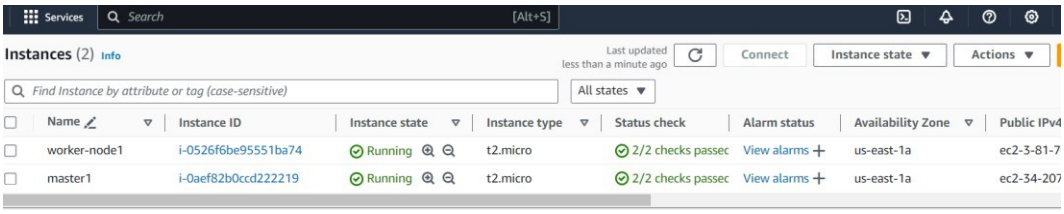
**Deploying Applications on Kubernetes**

**Application Deployment Lifecycle**

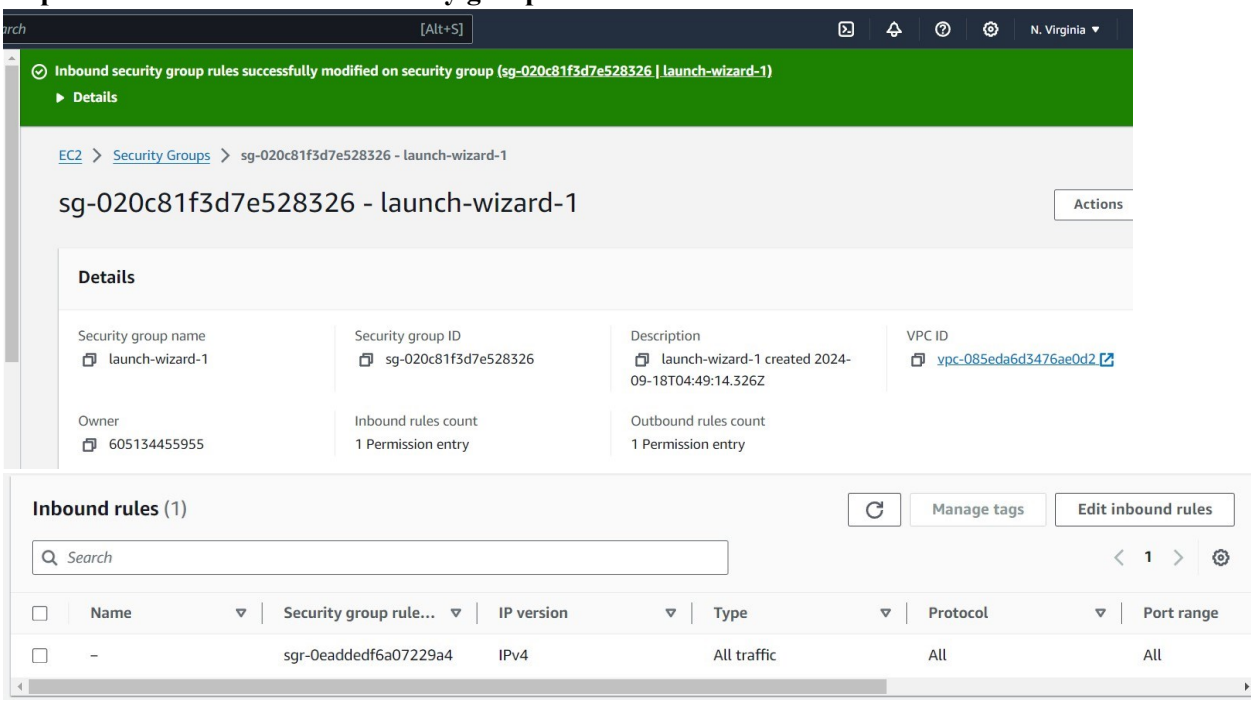
1. Define Application Requirements: Identify necessary resources such as CPU, memory, storage, etc.
2. Create Deployment Configurations: Write deployment manifests specifying container images, replicas for scaling, health checks, etc.
3. Deploying with Kubectl: Use kubectl commands like kubectl apply to deploy applications based on these configurations.
4. Monitoring and Scaling Applications: Monitor performance metrics and adjust deployments based on traffic demands.
5. Updating Applications: Modify deployment configurations for updates; Kubernetes supports rolling updates by default.
6. Rollback Capabilities: If an update causes issues, kubectl allows easy rollback to previous versions using commands like kubectl rollout undo.

## Implementation

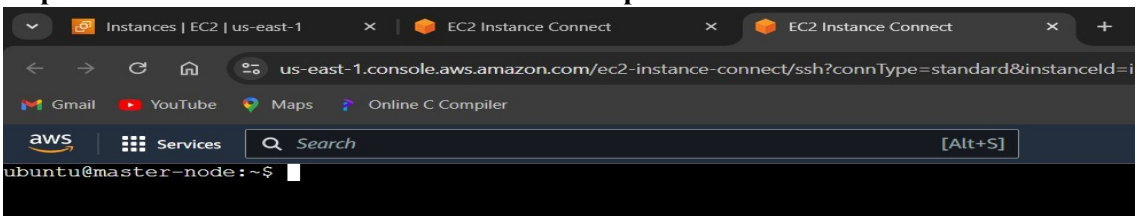
### Step 1.Creation of 2 EC2 Ubuntu Instances on AWS.




### Step 2.Edit inbound rules of security group 'launch-wizard-1' and set 'All Traffic'

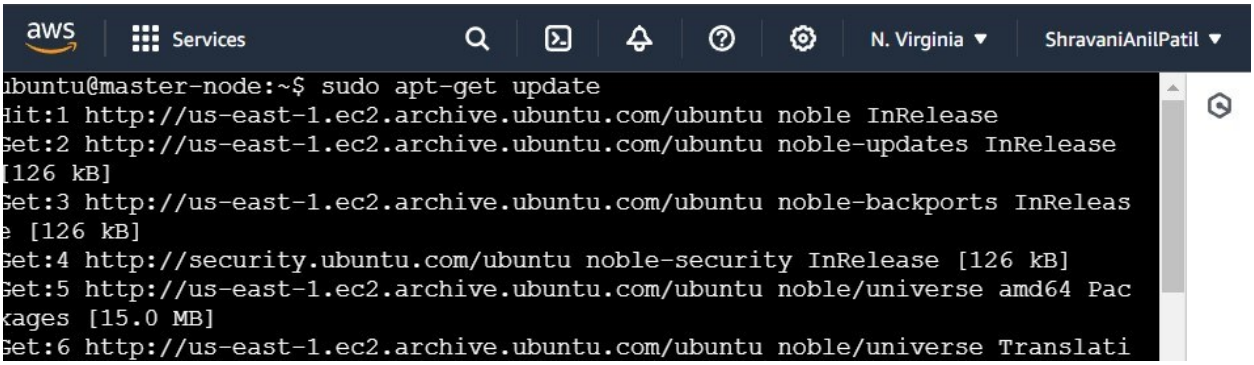



### Step 3. Set master and worker as hostname on respective servers




### Step 4.Installation of docker

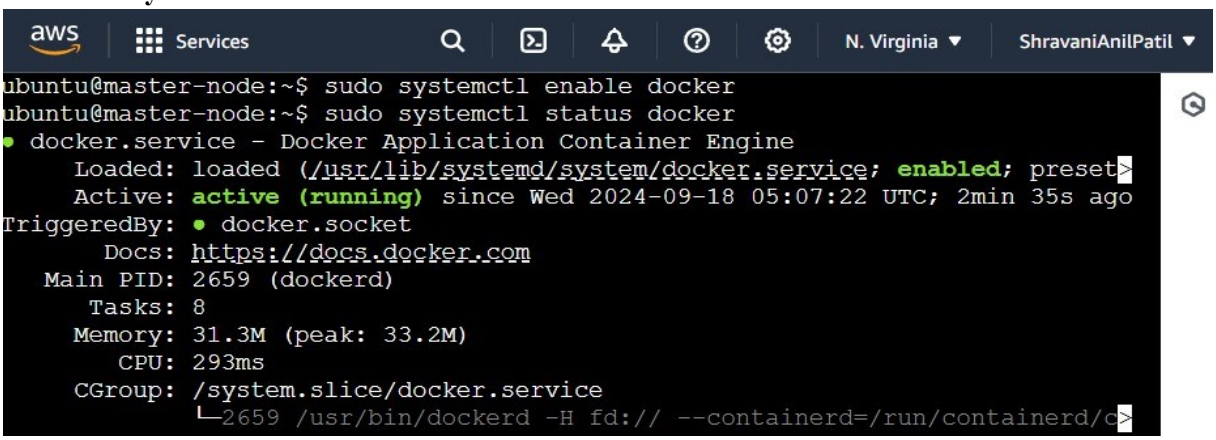
### 4.1 - sudo apt-get update




**4.2 - sudo apt-get install docker.io**


**4.3 – sudo systmectl enable docker**

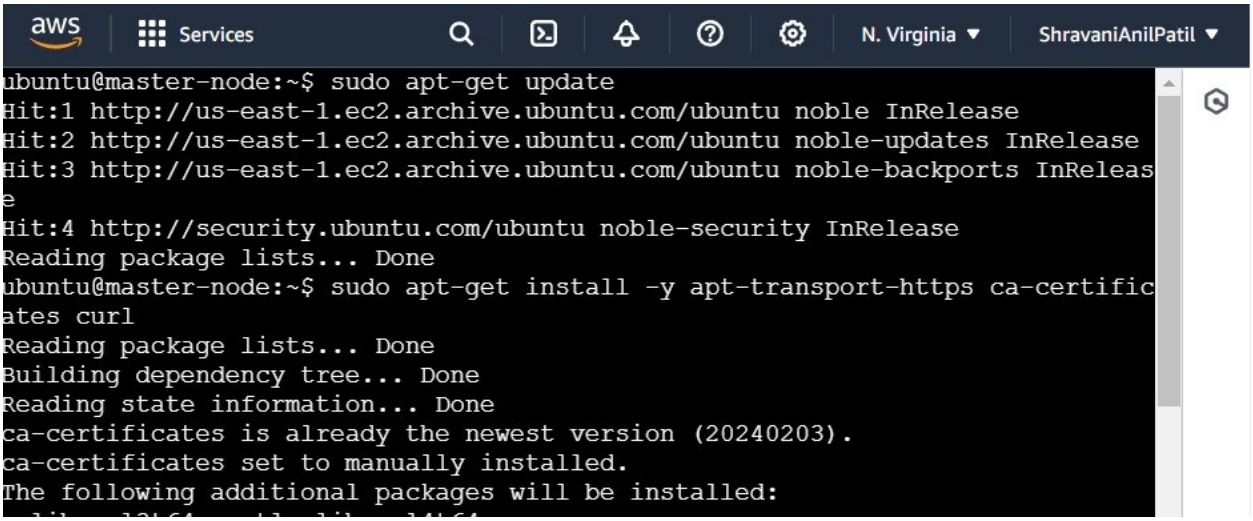
**4.4 – sudo systemctl status docker**



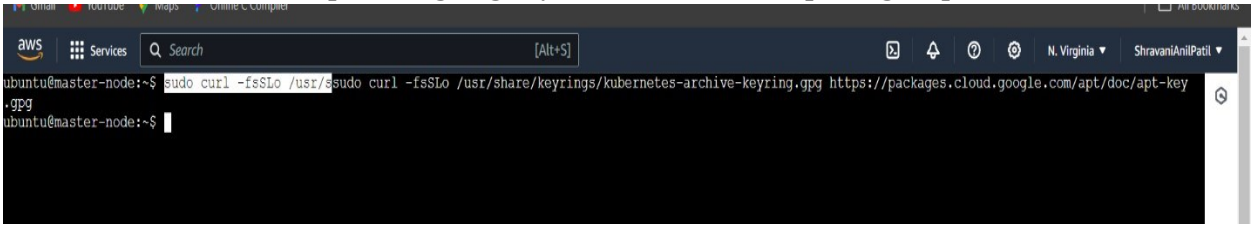
**Step 5.Installation of Kubernetes-**

**5.1 sudo apt-get update**

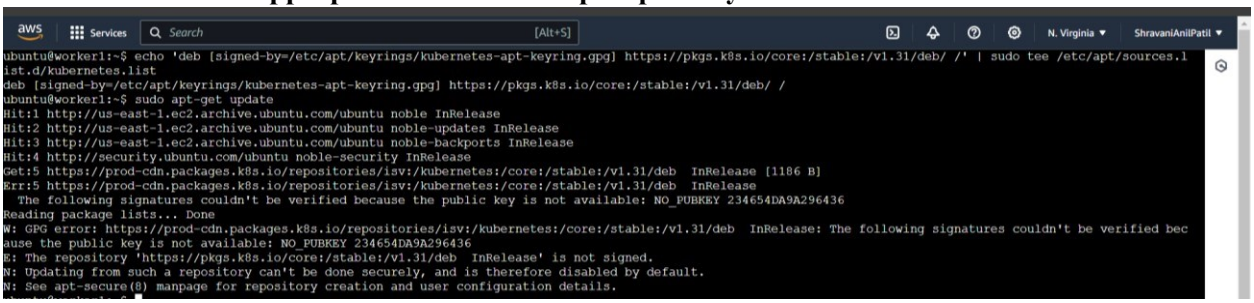
**5.2 install ca certificate**




**5.3 Download the public signing key for the Kubernetes package repositories.**

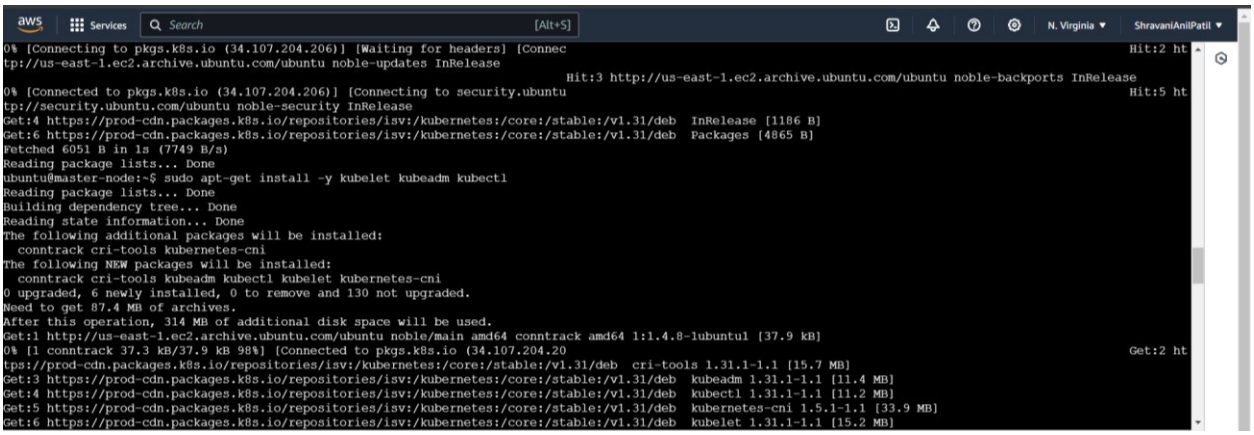



**5.4 Add the appropriate Kubernetes apt repository**

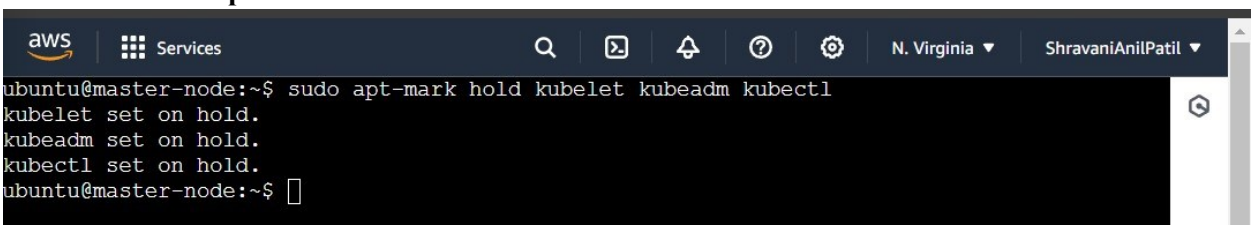



**5.5 sudo apt-get update**

**5.6 sudo apt-get install -y kubelet kubeadm kubectl**



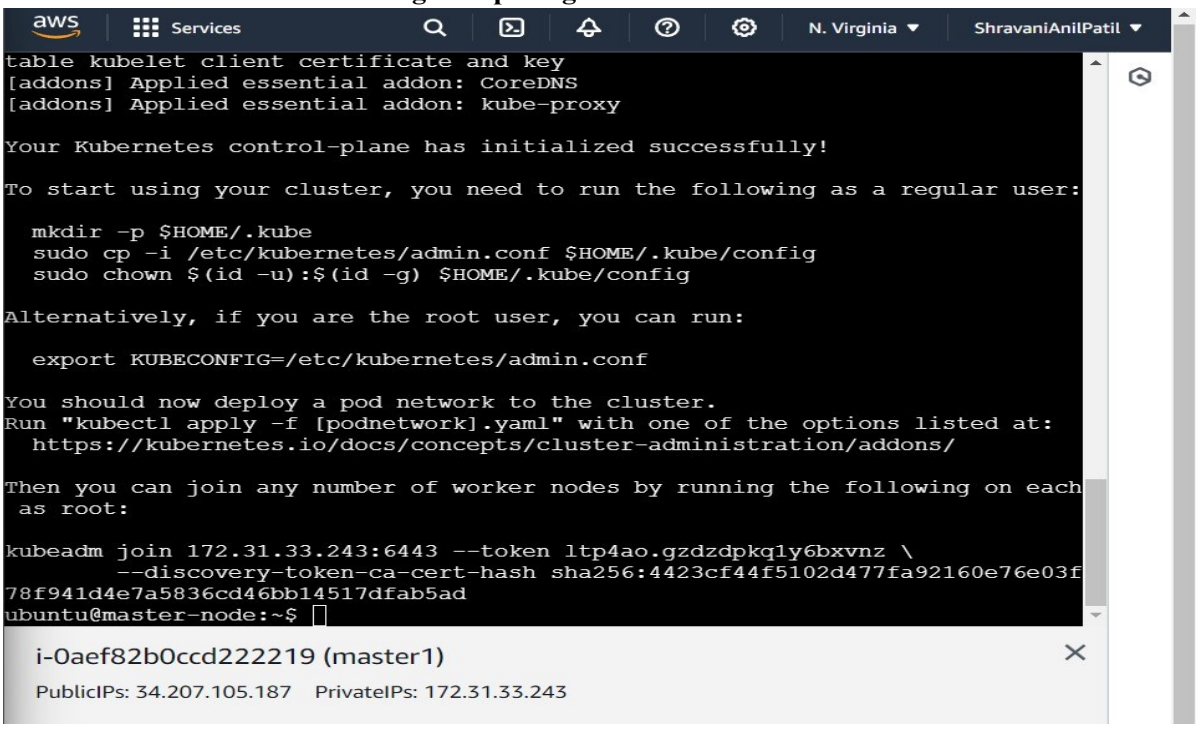

**5.7 sudo apt-mark hold kubelet kubeadm kubectl**




**Step.6 Kubernetes Deployment**

**6.1 sudo swapoff –a**

**6.2 Initialize Kubernetes on Master Node - sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all**



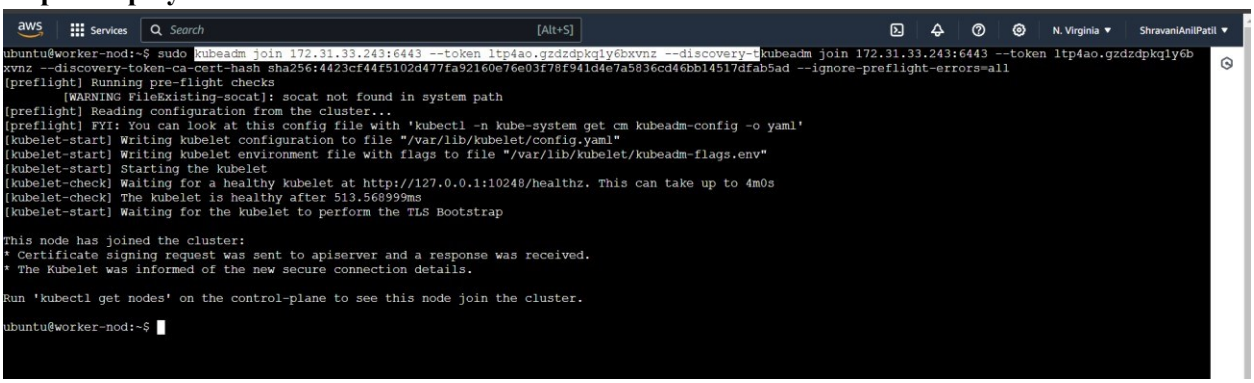
**Step 7.to create a directory for the cluster:**

**7.1mkdir -p $HOME/.kube**

**7.2sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config**

**7.3sudo chown HOME/.kube/config**

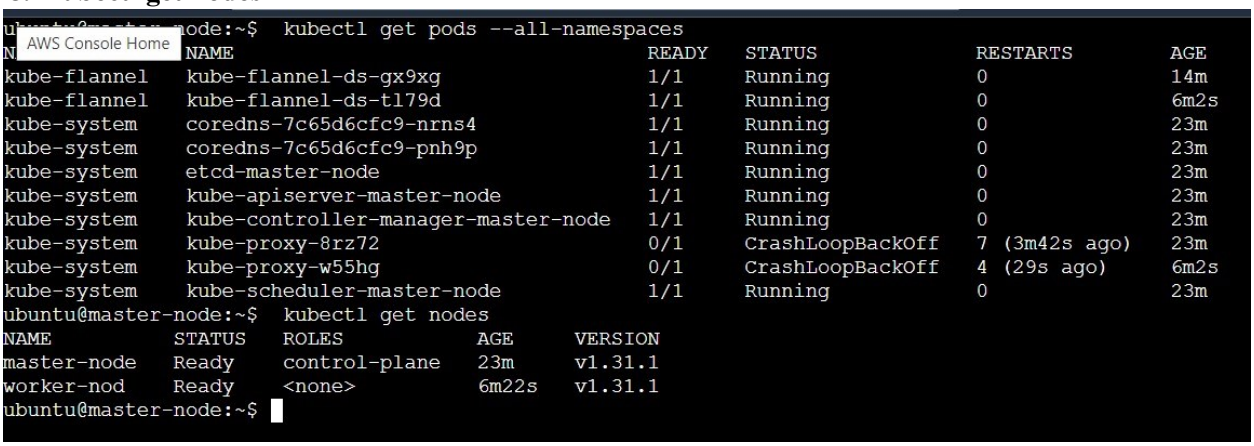
**Step 8. Deploy Pod Network to Cluster and Join Worker Node to Cluster**




**Verify that everything is running and communicating:**

**8.1kubectl get pods --all-namespaces**

**8.2kubectl get nodes**



**Step 9. Create one file deploy.yaml**

```
ubuntu@master-node:~$ sudo nano deploy.yaml
ubuntu@master-node:~$
```

```
ubuntu@master-node:~$ cat deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
ubuntu@master-node:~$
```

**Step 10 : Create Deployment**

```
aws | Services | Search [Alt+S]
ubuntu@master-node:~$ kubectl create -f deploy.yaml
deployment.apps/nginx-deployment created
ubuntu@master-node:~$
```

**(EXTRA) – kubectl get namepaces**

```
ubuntu@master-node:~$ kubectl get namespaces
NAME              STATUS   AGE
default           Active   9h
kube-flannel      Active   8h
kube-node-lease   Active   9h
kube-public       Active   9h
kube-system       Active   9h
ubuntu@master-node:~$
```

**Step 11. After deployment verify the same:**

```
ubuntu@master-node:~$ kubectl get deploy
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           3m13s
```

**Step 12: Expose the Application: Create a service to expose the deployment.**

```
ubuntu@master-node:~$ kubectl expose deployment.apps/nginx-deployment \
> --type="LoadBalancer"
service/nginx-deployment exposed
ubuntu@master-node:~$
```

**Step 13:Verfiy the service**

```
service/nginx-deployment exposed
ubuntu@master-node:~$ kubectl get svc
NAME               TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes         ClusterIP      10.96.0.1       <none>        443/TCP        4h43m
nginx-deployment   LoadBalancer   10.101.59.94    <pending>     80:31041/TCP   4m34s
ubuntu@master-node:~$
```

**Step 14: Access the application**