EXPERIMENT NO. 4

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

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

Theory:

What is kubectl?

kubectl is the command-line interface (CLI) used to interact with a Kubernetes cluster. It allows users to manage cluster resources, deploy applications, inspect and manage cluster components, and much more. Using kubectl, you can communicate with the Kubernetes API server to issue commands and queries.

Common kubectl commands:

- kubectl get: View information about resources.
- kubectl describe: Detailed description of resources.
- kubectl create/apply: Create or update resources.
- kubectl delete: Delete resources.

kubectl plays a crucial role in the day-to-day operation of a Kubernetes cluster.

Basic Concepts in Kubernetes

Before diving into the application deployment process, it's important to understand a few key Kubernetes objects:

1. **Pods**: The smallest deployable unit in Kubernetes. A pod encapsulates one or more containers (usually a single container) that share the same network namespace and storage.
2. **Deployments**: A Kubernetes resource that defines how to create and manage pods. It ensures the specified number of pod replicas are running at any given time and handles updates and rollbacks.
3. **Services**: An abstraction that defines how to access the pods. A service allows you to expose your pods to internal or external clients.
4. **ReplicaSets**: Ensures that a specified number of pod replicas are running at all times. It is managed by a Deployment, but can also be used independently.

Step 1: Install Kubectl on Ubuntu

1.1 Add Kubernetes APT repository

First, add the Kubernetes repository to your system.

1. Install prerequisites:

sudo apt-get update

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

```
ubuntu@ip-172-31-44-131:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri-o:/prerelease:/main/deb  InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  InRelease
Reading package lists... Done
```

```
ubuntu@ip-172-31-44-131:~$ sudo apt-get install -y apt-transport-https ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apt-transport-https is already the newest version (2.7.14build2).
ca-certificates is already the newest version (20240203).
curl is already the newest version (8.5.0-2ubuntu10.4).
0 upgraded, 0 newly installed, 0 to remove and 10 not upgraded.
```

2. Add the GPG key for Kubernetes:

sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg

3. Add the Kubernetes repository:

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-focal main" | sudo tee

/etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-44-131:~$ echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-focal main" | sudo
 tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-focal main
```

1.2 Install kubectl Now install kubectl: sudo apt-get update

sudo apt-get install -y kubectl

```
ubuntu@ip-172-31-44-131:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri-o:/prerelease:/main/deb  InRelease
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:5 https://packages.cloud.google.com/apt kubernetes-focal InRelease
Err:7 https://packages.cloud.google.com/apt kubernetes-focal Release
  404  Not Found [IP: 142.250.76.206 443]
Reading package lists... Done
E: The repository 'https://apt.kubernetes.io kubernetes-focal Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
```

```
ubuntu@ip-172-31-44-131:~$ sudo apt-get install -y kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
kubectl is already the newest version (1.29.0-1.1).
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.
ubuntu@ip-172-31-44-131:~$
```

Verify the installation(extra): kubectlversion --client

```
ubuntu@ip-172-31-44-131:~$ kubectl version --client
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Step 2: Deploying Your Application on Kubernetes

2.1 Set up Kubernetes Cluster

1. If you haven't already set up a Kubernetes cluster (e.g., with kubeadm), use minikube or any managed Kubernetes service (like EKS, GKE, etc.) to get a cluster running.

2. Once your cluster is ready, verify the nodes:

kubectl get nodes

```
ubuntu@ip-172-31-44-131:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-40-114    Ready    <none>          9m55s   v1.29.0
ip-172-31-44-131    Ready    control-plane   33m     v1.29.0
```

Step 3:**Create the Deployment YAML file**

a) Create the YAML file: Use a text editor to create a file named nginx-deployment.yaml
Add the Deployment Configuration: Copy and paste the following YAML content into the file. Save and exit the editor (Press Ctrl+X, then Y, and Enter).

```
 GNU nano 7.2                                  nginx-deployment.yaml *
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80

Save modified buffer?
 Y Yes
 N No           ^C Cancel
```

Step 4:Create the Service YAML File

a) Create the YAML File: Create another file named nginx-service.yaml Add the Service Configuration: Copy and paste the following YAML content into the file given below.

```
 GNU nano 7.2                                  nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: my-nginx
  labels:
    run: my-nginx
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    run: my-nginx

                                [ Read 12 lines ]
^G Help       ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo        M-A Set Mark    M-]
^X Exit       ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo        M-6 Copy        ^Q
```

Step 5:Apply the YAML Files

a) Deploy the Application: Use kubectl to create the Deployment and Service from the YAML files.

Verify the Deployment: Check the status of your Deployment,Pods and Services.

Describe the deployment(Extra)

```
ubuntu@ip-172-31-44-131:~$ ubuntu@ip-172-31-44-131:~$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-44-131:~$ kubectl apply -f nginx-service.yaml
service/my-nginx created
```

Step 6:Ensure Service is Running

6.1 **Verify Service**: Run the following command to check the services running in your cluster:

Kubectl get deployment

Kubectl get pods

kubectl get service

```
ubuntu@ip-172-31-44-131:~$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           74s
ubuntu@ip-172-31-44-131:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-86dcfdf4c6-8d7rx   1/1     Running   0          81s
nginx-deployment-86dcfdf4c6-bdbcm   1/1     Running   0          81s
ubuntu@ip-172-31-44-131:~$ kubectl get services
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP   48m
my-nginx     ClusterIP   10.111.168.255   <none>        80/TCP    55s
```

Step 7:Forward the Service Port to Your Local Machine

kubectl port-forward allows you to forward a port from your local machine to a port on a service running in the Kubernetes cluster.

1. **Forward the Service Port**: Use the following command to forward a local port to the service's target port.

kubectl port-forward service/<service-name> <local-port>:<service-port>

This command will forward local port 8080 on your machine to port 80 of the service nginx-service running inside the cluster.

```
ubuntu@ip-172-31-44-131:~$ kubectl describe deployments
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Tue, 17 Sep 2024 17:00:22 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=nginx
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   nginx-deployment-86dcfdf4c6 (2/2 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  2m9s  deployment-controller  Scaled up replica set nginx-deployment-86dcfdf4c6 to 2
```

2. This means port forwarding is now active, and any traffic to localhost:8080 will be routed to the nginx-service on port 80.

```
ubuntu@ip-172-31-44-131:~$ kubectl get services
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP   49m
my-nginx     ClusterIP   10.111.168.255   <none>        80/TCP    2m9s
ubuntu@ip-172-31-44-131:~$
```

```
ubuntu@ip-172-31-44-131:~$ nano nginx-services.yaml
ubuntu@ip-172-31-44-131:~$ nano nginx-service.yaml
ubuntu@ip-172-31-44-131:~$ kubectl apply -f nginx-service.yaml
service/nginx-service created
ubuntu@ip-172-31-44-131:~$ kubectl get services
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP      10.96.0.1        <none>        443/TCP        71m
my-nginx        ClusterIP      10.111.168.255   <none>        80/TCP         23m
nginx-service   LoadBalancer   10.105.174.168   <pending>     80:31376/TCP   10s
ubuntu@ip-172-31-44-131:~$ kubectl port-forward service/nginx-service 8088:80
Forwarding from 127.0.0.1:8088 -> 80
Forwarding from [::1]:8088 -> 80
^Cubuntu@ip-172-31-44-131:~kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-86dcfdf4c6-8d7rx   1/1     Running   0          26m
nginx-deployment-86dcfdf4c6-bdbcm   1/1     Running   0          26m
```

Step 8: Access the Application Locally

1. **Open a Web Browser**: Now open your web browser and go to the following URL:

http://localhost:8080
You should see the application (in this case, Nginx) that you have deployed running in the Kubernetes cluster, served locally via port 8080.

In case the port 8080 is unavailable, try using a different port like 8081