Experiment N0. 06

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy your first Kubernetes application.

LOs:

Theory:

This process involves installing the command-line tool for Kubernetes and using it to deploy a complete Nginx web server application.

Step 1: Install Kubectl

kubectl is the primary tool you'll use to interact with your Kubernetes cluster's API Server.

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

```
ubuntu@ip-172-31-45-31:~$ curl -LO "https://dl.k8s.io/release/$(cu
rl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kub
ectl"
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubect
l
  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current
                                 Dload  Upload   Total   Spent
Left  Speed
100   138  100   138    0     0   1469      0 --:--:-- --:--:-- --
:--:--  1483
100 57.7M  100 57.7M    0     0   140M      0 --:--:-- --:--:-- --
:--:--  140M
```

Step 2: Execute Kubectl Commands to Deploy an Application

We will deploy a simple Nginx application by defining its desired state in YAML manifest files.

1. **Create a Namespace:** A namespace provides a scope for names, allowing you to organize your resources.

   ```
   kubectl create namespace my-nginx-app
   ```

   ```
   ubuntu@ip-172-31-45-31:~$ kubectl create namespace my-nginx-app
   namespace/my-nginx-app created
   ```

2. **Create a Deployment Manifest:** A **Deployment** manages your application's Pods, ensuring a specified number of replicas are always running. This command creates a **`deployment.yaml`** file.

```
cat <<EOF > deployment.yaml
# API version for Deployments
apiVersion: apps/v1
# Type of object
kind: Deployment
metadata:
  name: my-nginx-deployment
  # Place it in our new namespace
  namespace: my-nginx-app
spec:
  # We want 2 identical copies (Pods) of our app
  replicas: 2
  # This selector tells the Deployment which Pods to manage
  selector:
    matchLabels:
      app: nginx
  # This is the template, or blueprint, for the Pods
  template:
    metadata:
      # Pods created will have this label
      labels:
        app: nginx
    spec:
      containers:
      # Define the container to run inside the Pod
      - name: nginx-container
        image: nginx # The Docker image to use
        ports:
        - containerPort: 80 # The port the app listens on
EOF
```

3. **Create a Service Manifest:** A **Service** exposes your application to the network. This command creates a **`service.yaml`** file.

```
cat <<EOF > service.yaml
apiVersion: v1
kind: Service
metadata:
  name: my-nginx-service
  namespace: my-nginx-app
spec:
  # NodePort exposes the service on a static port on the Node's IP
  type: NodePort
  # This selector finds Pods with the label 'app: nginx'
  selector:
    app: nginx
  ports:
    # Define port mapping
  - protocol: TCP
    port: 80 # The service's internal port
    targetPort: 80 # The port on the Pod to forward traffic to
EOF
```

```
ubuntu@ip-172-31-45-31:~$ cat <<EOF > service.yaml
apiVersion: v1
kind: Service
metadata:
  name: my-nginx-service
  namespace: my-nginx-app
spec:
  # NodePort exposes the service on a static port on the Node's IP
  type: NodePort
  # This selector finds Pods with the label 'app: nginx'
  selector:
    app: nginx
  ports:
```

4. **Apply the Manifests:** These commands send the configuration files to the Kubernetes API Server, which then creates the resources.

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
```

```
ubuntu@ip-172-31-45-31:~$ kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
deployment.apps/my-nginx-deployment created
service/my-nginx-service created
```

5. **Verify the Deployment:** Check the status of all the resources you created in your namespace.

kubectl get all --namespace my-nginx-app

```
ubuntu@ip-172-31-45-31:~$ kubectl get all --namespace my-nginx-app
NAME                                    READY   STATUS    RESTARTS   AGE
pod/my-nginx-deployment-599786d4bc-7zbcn   1/1     Running   0          15s
pod/my-nginx-deployment-599786d4bc-m6fkg   1/1     Running   0          15s

NAME                       TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/my-nginx-service   NodePort   10.99.151.217   <none>        80:32018/TCP   15s

NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-nginx-deployment   2/2     2            2           15s
```

6. **Access the Application from Your Browser:** This command forwards traffic from your EC2 instance's port 8080 to the Nginx service. This terminal must be left open.

kubectl port-forward --namespace my-nginx-app service/my-nginx-service 8080:80 --address 0.0.0.0

```
ubuntu@ip-172-31-45-31:~$ kubectl port-forward --namespace my-ngin
x-app service/my-nginx-serkubectl port-forward --namespace my-ngin
x-app service/my-nginx-service 8080:80 --address 0.0.0.0
Forwarding from 0.0.0.0:8080 -> 80
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
```

7. **Final Step: View in Your Browser**
Now, on your **local computer**, open a web browser and go to this address:

http://<Your-EC2-Public-IP>:8080