

D15A 43

Step 1: Deploying Your Application on Kubernetes**1.1 Set up Kubernetes Cluster**

1. If you haven't already set up a Kubernetes cluster (e.g., with kubectl), 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:

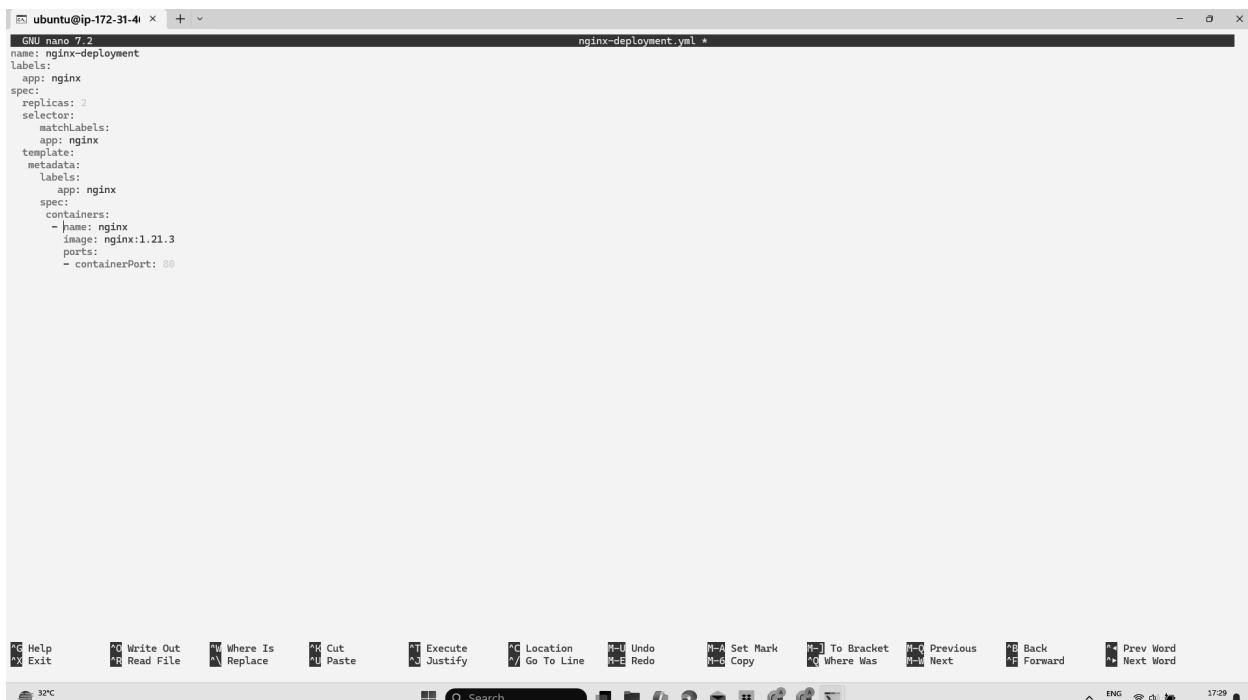
kubectl get nodes

```
ubuntu@ip-172-31-46-220:~$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
ip-172-31-36-212	Ready	<none>	47s	v1.29.0
ip-172-31-46-220	Ready	control-plane	16m	v1.29.0
ip-172-31-47-26	Ready	<none>	29s	v1.29.0

Step 2: 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).



```
ubuntu@ip-172-31-46-220:~$ nano nginx-deployment.yaml
name: nginx-deployment
labels:
  app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.21.3
        ports:
        - containerPort: 80
```

The screenshot shows a terminal window with the nano text editor open. The file being edited is named 'nginx-deployment.yaml'. The content of the file is a Kubernetes Deployment configuration for nginx. The configuration includes labels, replicas (2), a selector, and a template with a single container named 'nginx' using the 'nginx:1.21.3' image and exposing port 80. The terminal window also shows various nano editor shortcuts at the bottom and system status at the very bottom.

Step 3: 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

```
ubuntu@ip-172-31-41:~$ cat nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-server
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

Step 4: 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-172-31-46-220:~$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
```

```
ubuntu@ip-172-31-46-220:~$ kubectl apply -f nginx-service.yaml
service/nginx-server created
```

Step 5: 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-172-31-46-220:~$ kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment    3/3     3             3           7m27s
```

```
ubuntu@ip-172-31-46-220:~$ kubectl get services
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes          ClusterIP      10.96.0.1        <none>         443/TCP          85m
nginx-server        LoadBalancer  10.111.218.213   <pending>     80:30798/TCP     110s
```

Step 6: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/` :

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

```
ubuntu@ip-172-31-46-220:~$ kubectl describe deployments
Name:          nginx-deployment
Namespace:     default
CreationTimestamp: Sat, 21 Sep 2024 12:30:54 +0000
Labels:        app=nginx
Annotations:   deployment.kubernetes.io/revision: 1
Selector:      app=nginx
Replicas:      3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:  RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
    nginx:
      Image:      nginx:1.16
      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-854bc88786 (3/3 replicas created)
Events:
  Type           Reason             Age   From               Message
  ----           -
  Normal        ScalingReplicaSet   11m   deployment-controller   Scaled up replica set nginx-deployment-854bc88786 to 3
```

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-172-31-46-220:~$ kubectl port-forward service/nginx-server 8080:80
```

Step 7:

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

