Kubernetes Assignment:- 2

Create Helm charts for a sample application (e.g., a web server) with configurable values.

command : helm create web-app

This command will generate a Helm chart template in a directory called spring-boot-app. The default Helm chart includes directories like templates, and files such as Chart.yaml and values. yaml.

go to web -app command **: cd web-app**

Modify the values.yaml file.

Edit the `values.yaml` file to include configurable values for your Spring Boot application. Set the Docker image reference, container port and replica set.
Using this command: vi values.yaml

Modify the deployment.yaml file.

Open the `deployment.yaml` file located in the `templates` directory and update it to use the Spring Boot Docker image and container port.

command : vi ./templates/deployment.yaml

Save the file after making the changes.

```
anil@IN-PG02P670:~$ ls
helm-v3.11.0-linux-amd64.tar.gz  java-project  linux-amd64  minikube-linux-amd64  snap
anil@IN-PG02P670:~$ helm create web-app
Creating web-app
anil@IN-PG02P670:~$ ls
helm-v3.11.0-linux-amd64.tar.gz  java-project  linux-amd64  minikube-linux-amd64  snap  web-app
anil@IN-PG02P670:~$ cd web-app
anil@IN-PG02P670:~/web-app$ ls
Chart.yaml  charts  templates  values.yaml
anil@IN-PG02P670:~/web-app$ vi values.yaml
anil@IN-PG02P670:~/web-app$ cd templates
anil@IN-PG02P670:~/web-app/templates$ ls
NOTES.txt  _helpers.tpl  deployment.yaml  hpa.yaml  ingress.yaml  service.yaml  serviceaccount.yaml  tests
anil@IN-PG02P670:~/web-app/templates$ vi deployment.yaml
```

Use the Helm charts to deploy the application to a Kubernetes cluster.

Install the Helm Chart:

Deploy the Spring Boot application to the Kubernetes cluster using the helm install command:

- `helm install my-web-app ./web-app`

```
anil@IN-PG02P670:~$ helm install my-web-app web-app
NAME: my-web-app
LAST DEPLOYED: Tue Oct  1 10:25:47 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services my-web-app)
  export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
anil@IN-PG02P670:~$ helm list -a
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP
VERSION
my-web-app      default         1               2024-10-01 10:25:47.99259272 +0000 UTC  deployed        web-app-0.1.0   1.16
.0
```

Verify the Deployment:

Using this command :

- kubectl get pods
- kubectl get services

Upgrade the application by changing the Helm chart values and performing a Helm upgrade.

Update the `values.yaml` file**:** For example, update the `replicaCount` value from 1 to 2

Apply the upgrade command :

helm upgrade my-web-app ./web-app

Verify the upgrade command:

kubectl get deployment spring-boot-app

```
anil@IN-PG02P670:~$ helm upgrade my-web-app web-app
Release "my-web-app" has been upgraded. Happy Helming!
NAME: my-web-app
LAST DEPLOYED: Tue Oct  1 18:02:39 2024
NAMESPACE: default
STATUS: deployed
REVISION: 2
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services my-web-app)
  export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
anil@IN-PG02P670:~$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
my-web-app-7bd64d867-4g442    1/1     Running   0          7h27m
my-web-app-7bd64d867-nqp2m    1/1     Running   0          2m39s
my-web-app-7bd64d867-rs6cz    1/1     Running   0          2m39s
my-web-app-7bd64d867-zfm2m    1/1     Running   0          70s
anil@IN-PG02P670:~$ kubectl get deployment my-web-app
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
my-web-app   4/4     4            4           7h31m
```

Implement a namespace strategy for your Kubernetes cluster, organizing applications and resources into separate namespaces based on their environments (e.g., development, staging, production).

Create Namespaces using this command:

```
kubectl create namespace development
kubectl create namespace staging
kubectl create namespace production
```

Deploy the Application to Different Namespaces:

```
helm install spring-boot-app-dev ./spring-boot-app --namespace development
helm install spring-boot-app-staging ./spring-boot-app --namespace staging
helm install spring-boot-app-prod ./spring-boot-app --namespace production
```

Verify Deployments in Different Namespaces:

```
kubectl get pods -n development
kubectl get pods -n staging
kubectl get pods -n production
```

```
anil@IN-PG02P670:~$ kubectl create namespace development
namespace/development created
anil@IN-PG02P670:~$ helm install my-web-app-dev ./web-app --namespace development
NAME: my-web-app-dev
LAST DEPLOYED: Tue Oct  1 18:07:27 2024
NAMESPACE: development
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace development -o jsonpath="{.spec.ports[0].nodePort}" services my-web-app-dev)
  export NODE_IP=$(kubectl get nodes --namespace development -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
anil@IN-PG02P670:~$ kubectl create namespace staging
namespace/staging created
anil@IN-PG02P670:~$ helm install my-web-app-staging ./web-app --namespace staging
NAME: my-web-app-staging
LAST DEPLOYED: Tue Oct  1 18:11:38 2024
NAMESPACE: staging
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace staging -o jsonpath="{.spec.ports[0].nodePort}" services my-web-app-staging)
  export NODE_IP=$(kubectl get nodes --namespace staging -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
anil@IN-PG02P670:~$ kubectl create namespace prod
namespace/prod created
anil@IN-PG02P670:~$ helm install my-web-app-prod ./web-app --namespace prod
```

Rollback the application to a previous version using Helm rollback.

View the Helm Release History:

```
helm history spring-boot-app
```

Rollback to a Previous Version:

Use the helm rollback command followed by the release name and revision number:

```
helm rollback my-web-app 2
```

Verify the Rollback:

Check the status of the deployment to ensure it has been rolled back successfully:

```
kubectl get deployment my-web-app
```

```
anil@IN-PG02P670:~$ helm history my-web-app
REVISION        UPDATED                         STATUS          CHART           APP VERSION     DESCRIPTION
1               Tue Oct  1 10:34:53 2024        superseded      web-app-0.1.0   1.16.0          Install complete
2               Tue Oct  1 18:02:39 2024        deployed        web-app-0.1.0   1.16.0          Upgrade complete
anil@IN-PG02P670:~$ helm rollback my-web-app 1
Rollback was a success! Happy Helming!
anil@IN-PG02P670:~$ helm rollback my-web-app 2
Rollback was a success! Happy Helming!
anil@IN-PG02P670:~$
```