Kubernetes Assignment

Objective:

- Create a simple web application (e.g., a "Hello, World!" application) and Dockerize it.
- Deploy the application to a Kubernetes cluster using kubectl.
- Expose the application using a Kubernetes Service to access it externally.
- Scale the application by increasing the number of replicas.

Step 1: Open WSL ubunut terminal and Execute below command to install Kubectl and make sure you should have already installed docker in your wsl and then execute $ chmod +x ./kubectl command to change the permission of the file and then execute $ sudo mv ./kubectl /usr/local/bin/kubectl command, it moves the kubectl binary from the current directory (./kubectl) to the directory /usr/local/bin with the name kubectl and then execute $ kubectl version –client command to verify the installation.

```
aryan@IN-CD1D5S3: /mnt/c/\

C:\Users\aryanverma>wsl
[sudo] password for aryan:
 * Starting Docker: docker                                                    aryan@IN-CD1D
5S3:/mnt/c/Users/aryanverma$ curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s ht
tps://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 53.7M  100 53.7M    0     0  5183k      0  0:00:10  0:00:10 --:--:-- 8186k
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$ $ chmod +x ./kubectl
$: command not found
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$ chmod +x ./kubectl
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$ sudo mv ./kubectl /usr/local/bin/kubectl
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$ kubectl version --client
Client Version: v1.31.0
```

Step 2: Now execute $ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

Command to download minikube binary and then execute $ sudo install minikube-linux-amd64 /usr/local/bin/minikube to install minikube now just execute minikube start to start the minikube.

```
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 99.0M  100 99.0M    0     0  6872k      0  0:00:14  0:00:14 --:--:-- 8271k
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$ minikube start
😄  minikube v1.34.0 on Ubuntu 20.04 (amd64)
✨  Automatically selected the docker driver. Other choices: none, ssh
📌  Using Docker driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.45 ...
💾  Downloading Kubernetes v1.31.0 preload ...
    > preloaded-images-k8s-v18-v1...:  326.69 MiB / 326.69 MiB  100.00% 5.24 Mi
    > gcr.io/k8s-minikube/kicbase...:  487.90 MiB / 487.90 MiB  100.00% 4.80 Mi
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: default-storageclass, storage-provisioner
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Step 3: Check the cluster info kubernetes using the below command

```
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$ kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:32769
CoreDNS is running at https://127.0.0.1:32769/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$ kubectl get node
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   97s   v1.31.0
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$
```

Step 4: Execute the below command to tell the minikube about the docker environment we are using and point our shell to docker.

```
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$ minikube docker-env
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://127.0.0.1:32771"
export DOCKER_CERT_PATH="/home/aryan/.minikube/certs"
export MINIKUBE_ACTIVE_DOCKERD="minikube"

# To point your shell to minikube's docker-daemon, run:
# eval $(minikube -p minikube docker-env)
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$ eval $(minikube -p minikube docker-env)
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma$
```

Step 5: Build the docker image using your docker file

```
aryan@IN-CD1D5S3: /mnt/c/U

aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ docker build -t springboot-kubernetes .
[+] Building 46.9s (8/8) FINISHED                                                                          docker:default
 => [internal] load build definition from Dockerfile                                                                   0.1s
 => => transferring dockerfile: 188B                                                                                   0.0s
 => [internal] load metadata for docker.io/library/openjdk:11                                                          3.1s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                                         0.0s
 => [internal] load .dockerignore                                                                                      0.1s
 => => transferring context: 2B                                                                                        0.1s
 => [internal] load build context                                                                                      0.4s
 => => transferring context: 19.67MB                                                                                   0.4s
 => [1/2] FROM docker.io/library/openjdk:11@sha256:99bac5bf83633e3c7399aed725c8415e7b569b54e03e4599e580fc9cdb7c21ab   42.4s
 => => resolve docker.io/library/openjdk:11@sha256:99bac5bf83633e3c7399aed725c8415e7b569b54e03e4599e580fc9cdb7c21ab    0.0s
 => => sha256:d9d4b9b6e964657da49910b495173d6c4f0d9bc47b3b44273cf82fd32723d165 5.16MB / 5.16MB                         0.9s
 => => sha256:99bac5bf83633e3c7399aed725c8415e7b569b54e03e4599e580fc9cdb7c21ab 1.04kB / 1.04kB                         0.0s
 => => sha256:e81b7f317654b0f26d3993e014b04bcb29250339b11b9de41e130feecd4cd43c 1.79kB / 1.79kB                         0.0s
 => => sha256:47a932d998b743b9b0bcce55aa8ede77de94a6a183c8a67dec9d5e3b8ce0faa7 6.26kB / 6.26kB                         0.0s
 => => sha256:001c52e26ad57e3b25b439ee0052f6692e5c0f2d5d982a00a8819ace5e521452 55.00MB / 55.00MB                      10.1s
 => => sha256:2068746827ec1b043b571e4788693eab7e9b2a95301176512791f8c317a2816a 10.88MB / 10.88MB                       5.5s
 => => sha256:9daef329d35093868ef75ac8b7c6eb407fa53abbcb3a264c218c2ec7bca716e6 54.58MB / 54.58MB                      16.8s
 => => sha256:d85151f15b6683b98f21c3827ac545188b1849efb14a1049710ebc4692de3dd5 5.42MB / 5.42MB                         9.0s
 => => sha256:66223a710990a0ae7162aeed80417d30303afa3f24aafa57aa30348725e2230b 213B / 213B                             9.3s
 => => sha256:db38d58ec8ab4111b072f6700f978a51985acd252aabce3be377f25162e68301 202.07MB / 202.07MB                    35.7s
 => => extracting sha256:001c52e26ad57e3b25b439ee0052f6692e5c0f2d5d982a00a8819ace5e521452                              6.1s
 => => extracting sha256:d9d4b9b6e964657da49910b495173d6c4f0d9bc47b3b44273cf82fd32723d165                              0.6s
 => => extracting sha256:2068746827ec1b043b571e4788693eab7e9b2a95301176512791f8c317a2816a                              0.5s
 => => extracting sha256:9daef329d35093868ef75ac8b7c6eb407fa53abbcb3a264c218c2ec7bca716e6                              6.0s
 => => extracting sha256:d85151f15b6683b98f21c3827ac545188b1849efb14a1049710ebc4692de3dd5                              0.5s
 => => extracting sha256:66223a710990a0ae7162aeed80417d30303afa3f24aafa57aa30348725e2230b                              0.0s
 => => extracting sha256:db38d58ec8ab4111b072f6700f978a51985acd252aabce3be377f25162e68301                              6.4s
 => [2/2] ADD target/Springboot-kubernetes.jar Springboot-kubernetes.jar                                               0.8s
 => exporting to image                                                                                                 0.1s
 => => exporting layers                                                                                                0.1s
 => => writing image sha256:cb5b7c0ebafa47671478dac3796884bbe5090200d39bd8da5bbef3dfc72eb021                           0.0s
```

Step 6: Make a deployment.yaml file to deploy your application on Kubernetes.

```
eclipse-workspace - Springboot-kubernetes/deployment.yaml - Eclipse IDE
File Edit Source Navigate Search Project Run Window Help

Package Explorer
ApiGateway
App1Assignment5
App2Assignment5
Asssignment3
bankingportal
blog-app-apis
BootDemoApp
CarInsuranceCalculator
ConfigServer
CoreJavaAssignment1
CustomerService
DsaAssignment
exittest
ExitTestD
FoodOrderingApp
InsurancePremiumCalculator
Java-Mini-Assignment2
NoteBackend
Servers
ServiceRegistry
Springboot-kubernetes
  src/main/java
  src/main/resources
  src/test/java
  src/test/resources
  JRE System Library [JavaSE-17]
  Maven Dependencies
  src
  target
  app.yaml

Dockerfile | BootDemoAppA... | Temperature... | Temperature... | Temperature... | springboot-k... | deployment.yaml | Dockerfile

1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: springboot-k8s
5 spec:
6   selector:
7     matchLabels:
8       app: springboot-k8s
9   replicas: 3
10  template:
11    metadata:
12      labels:
13        app: springboot-k8s
14    spec:
15      containers:
16        - name: springboot-k8s
17          image: springboot-kubernetes
18          imagePullPolicy: IfNotPresent
19          ports:
20            - containerPort: 8080

Problems | Declaration | Progress | Console | JUnit | Console | Servers | Coverage
<terminated> C:\Program Files\Java\jdk-19\bin\javaw.exe (12-Sept-2024, 6:01:16 pm) [pid: 5348]
\aryanverma\.m2\repository\com\javatechie\springboot-k8s-demo\0.0.1-SNAPSHOT\springboot-k8s-demo-0.0.1-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  8.844 s
[INFO] Finished at: 2024-09-12T18:01:27+05:30
[INFO] ------------------------------------------------------------------------

Writable | Insert | 13 : 28 : 226
```

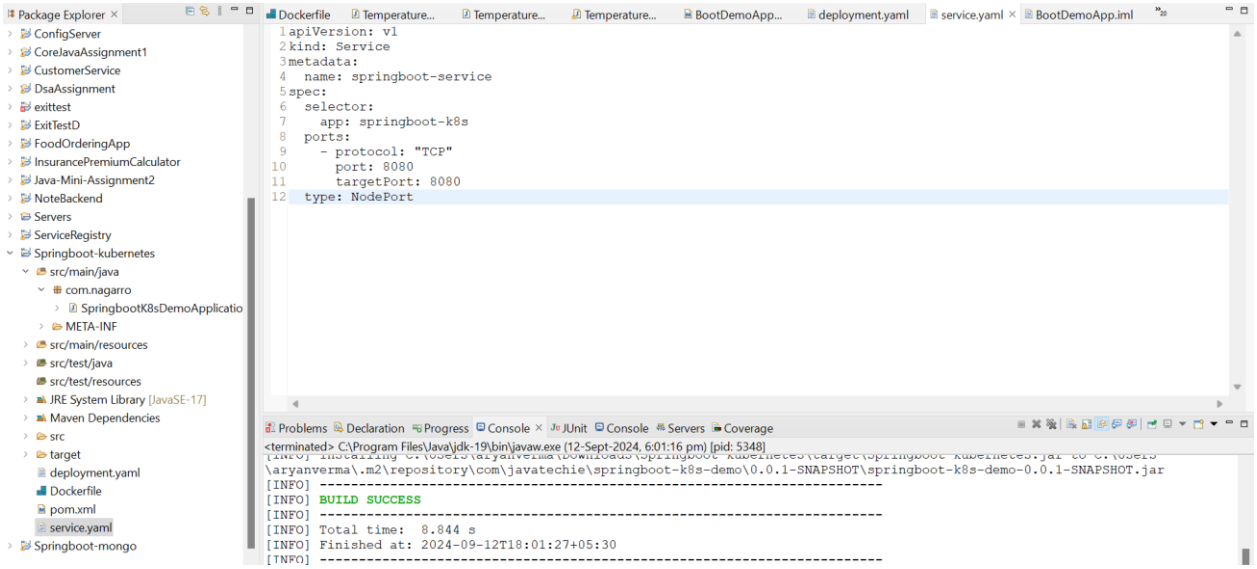
Step 7: Deploy the application using deployment.yaml file to kubernetes

```
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ ls
Dockerfile  deployment.yaml  pom.xml  src  target
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ kubectl apply -f deployment.yml
error: the path "deployment.yml" does not exist
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ kubectl apply -f deployment.yaml
deployment.apps/springboot-k8s created
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$
```

Step 8: Verify the deployments on the Kubernetes.

```
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ kubectl apply -f deployment.yaml
deployment.apps/springboot-k8s created
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ kubectl get deployments
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
springboot-k8s   3/3     3            3           8s
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
springboot-k8s-59467c5b8d-2qd8v   1/1     Running   0          19s
springboot-k8s-59467c5b8d-f726k   1/1     Running   0          19s
springboot-k8s-59467c5b8d-w8k5b   1/1     Running   0          19s
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$
```

Step 9: Create a service.yaml to expose the application to access it externally.



Step 10: Apply the services to the Kubernetes.

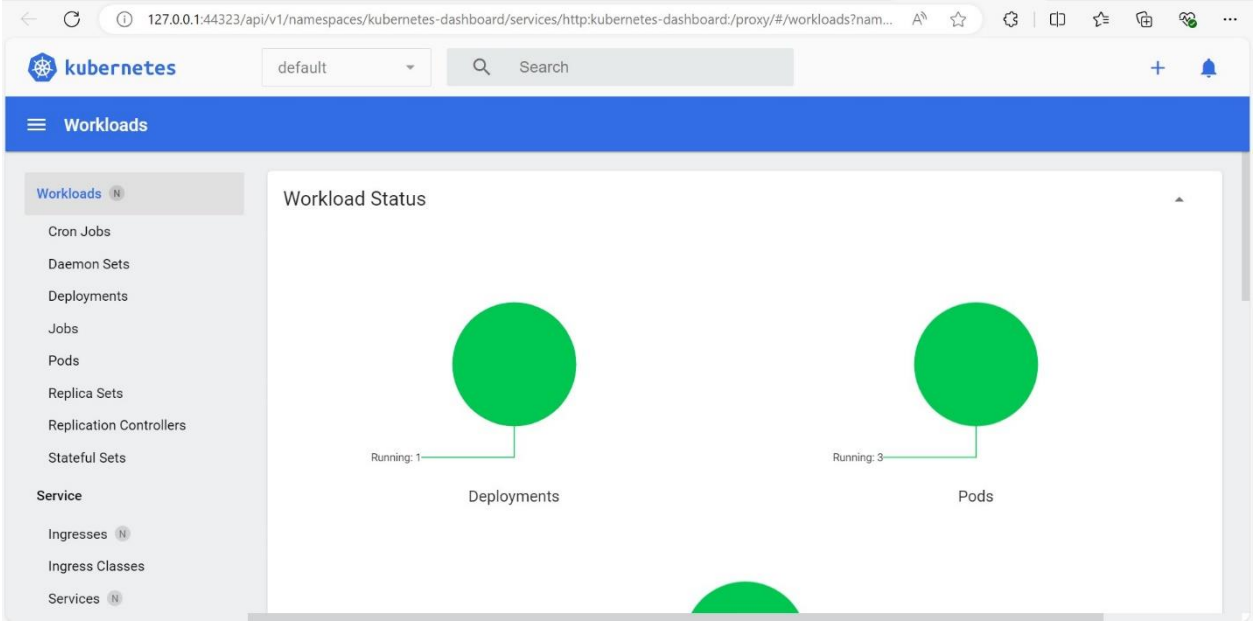
```
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ kubectl apply -f service.yaml
service/springboot-service created
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ kubectl get service
NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
kubernetes           ClusterIP   10.96.0.1     <none>        443/TCP          92m
springboot-service   NodePort    10.110.0.47   <none>        8080:32395/TCP   35s
```

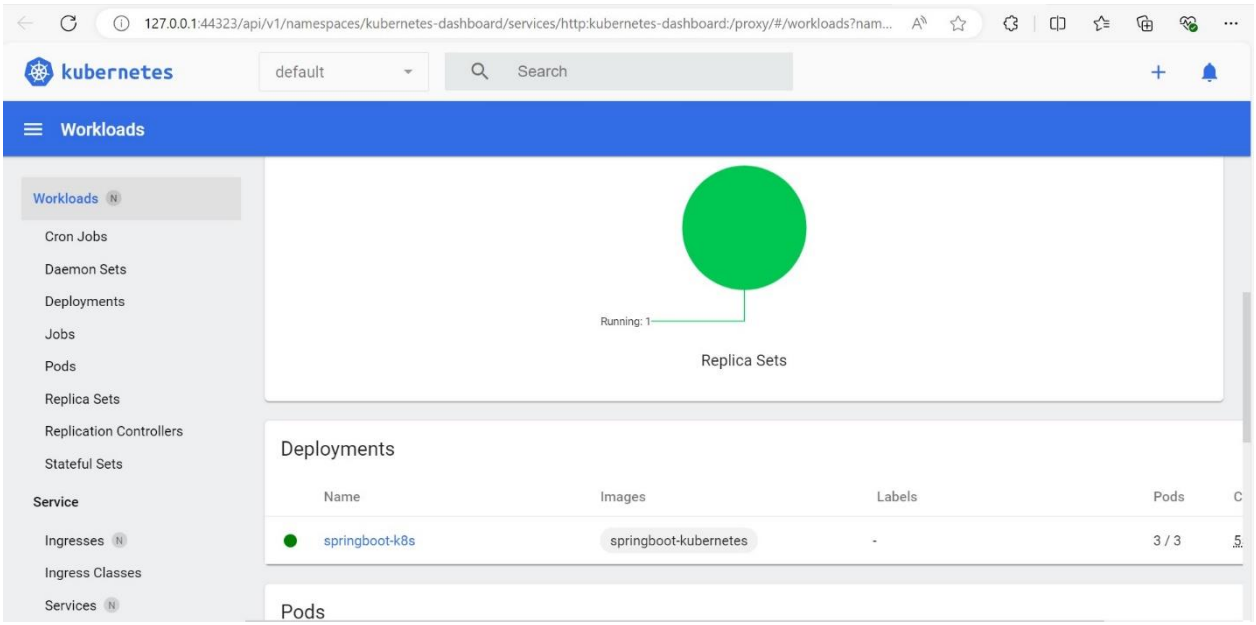
Step 11: Note the port on which your service is running which is 32395 and the minikube ip us 192.168.49.2 and verify the endpoints that service is accessible or not.

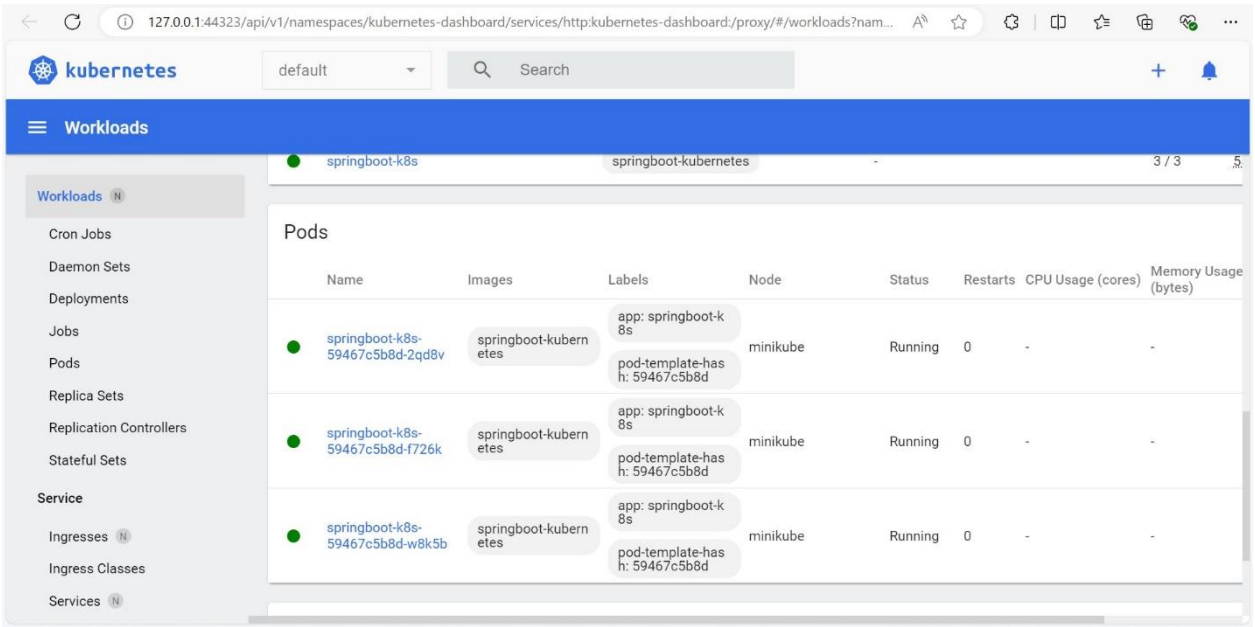
```
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ kubectl get service
NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
kubernetes           ClusterIP   10.96.0.1     <none>        443/TCP          92m
springboot-service   NodePort    10.110.0.47   <none>        8080:32395/TCP   35s
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ kubectl get nodes -o wide
NAME       STATUS   ROLES           AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION
      CONTAINER-RUNTIME
minikube   Ready    control-plane   93m   v1.31.0   192.168.49.2   <none>        Ubuntu 22.04.4 LTS   5.10.102.1-microsoft-standard-W
SL2   docker://27.2.0
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ minikube ip
192.168.49.2
```

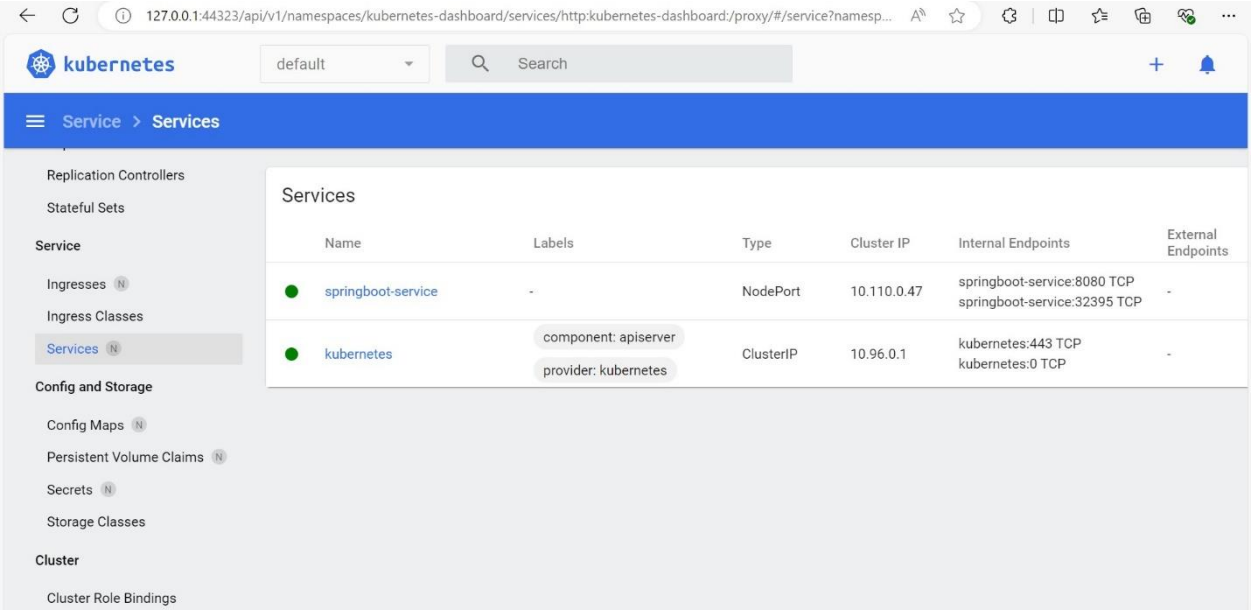
```
ot-kubernetes$ curl http://192.168.49.2:32395/message
Congratulation you successfully deployed your application to kubernetes !!aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springbo
```

Step 11: Execute the command minikube dashboard to see the deployments and services. Below is your deployments and pods details.



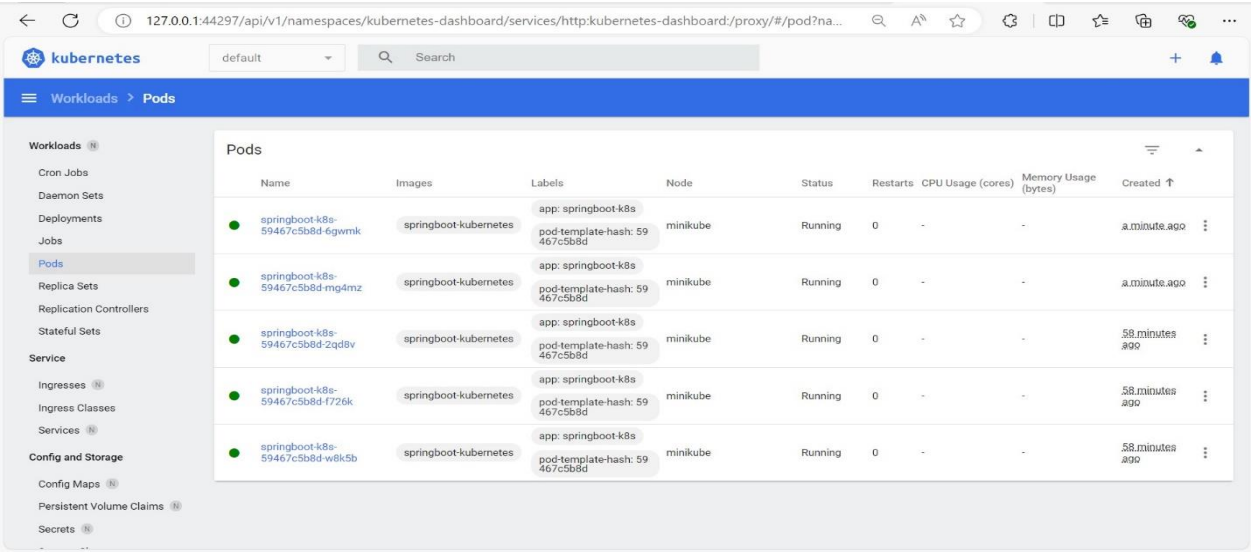


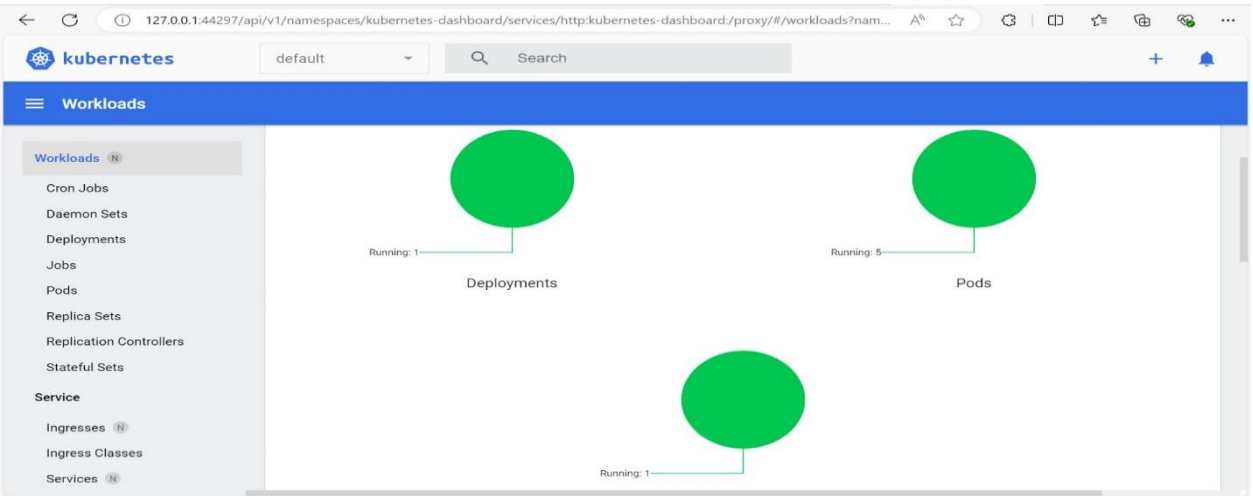


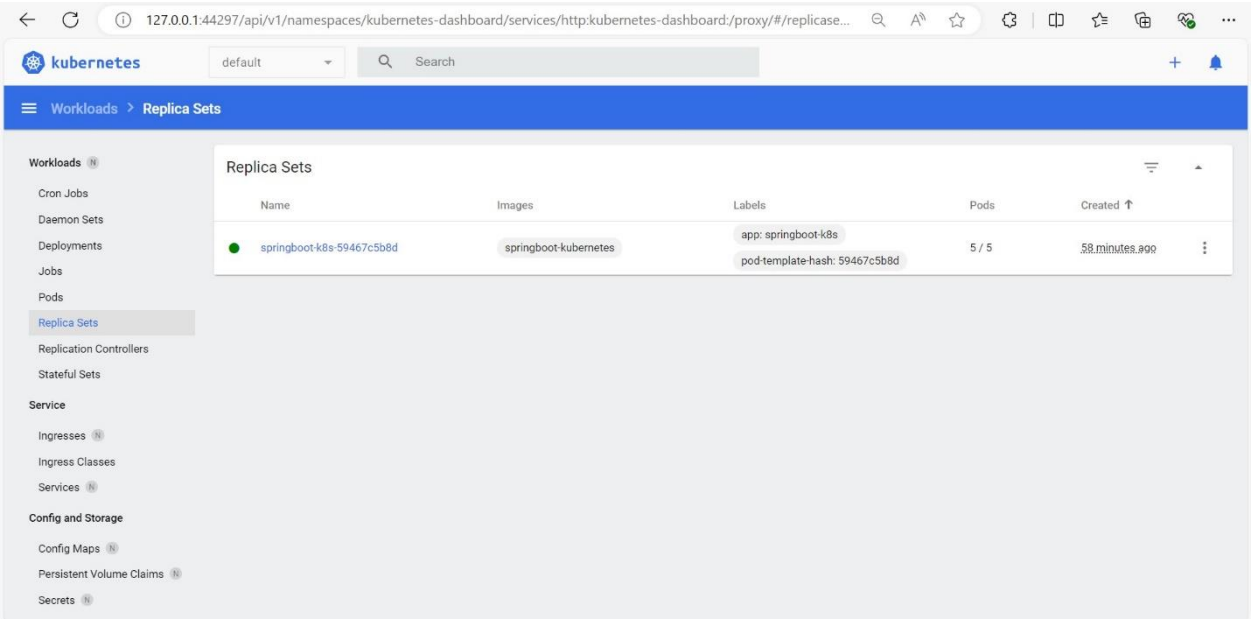


Step 12: Scale the application by increasing the number of replicas from 3 to 5 and verify using dashboard.

```
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ kubectl scale deployment springboot-k8s --replicas=5
deployment.apps/springboot-k8s scaled
aryan@IN-CD1D5S3:/mnt/c/Users/aryanverma/Downloads/Springboot-kubernetes$ minikube dashboard
🤔  Verifying dashboard health ...
🚀  Launching proxy ...
🤔  Verifying proxy health ...
🎉  Opening http://127.0.0.1:44297/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default
```






127.0.0.1:44297/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/replicase...
kubernetes
default
Search
Workloads > Replica Sets
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service
Ingresses
Ingress Classes
Services
Config and Storage
Config Maps
Persistent Volume Claims
Secrets
Replica Sets
Name
Images
Labels
Pods
Created
springboot-k8s-59467c5b8d
springboot-kubernetes
app: springboot-k8s
pod-template-hash: 59467c5b8d
5 / 5
58 minutes ago