

TASK 4 KUBERNET USING SHELL SCRIPT

Step 1: MiniKube

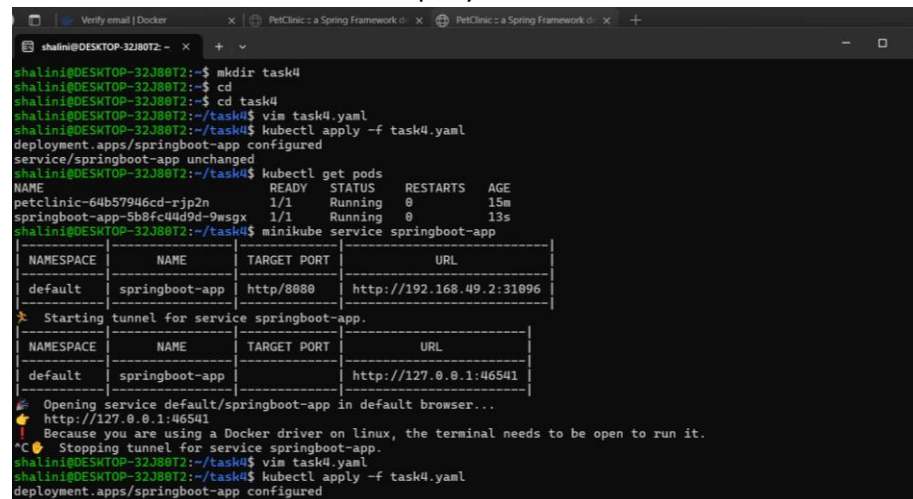
Start the minikube using minikube start command

Step 2: Folder Creation

Create a folder named task

Step 3: New Yaml File

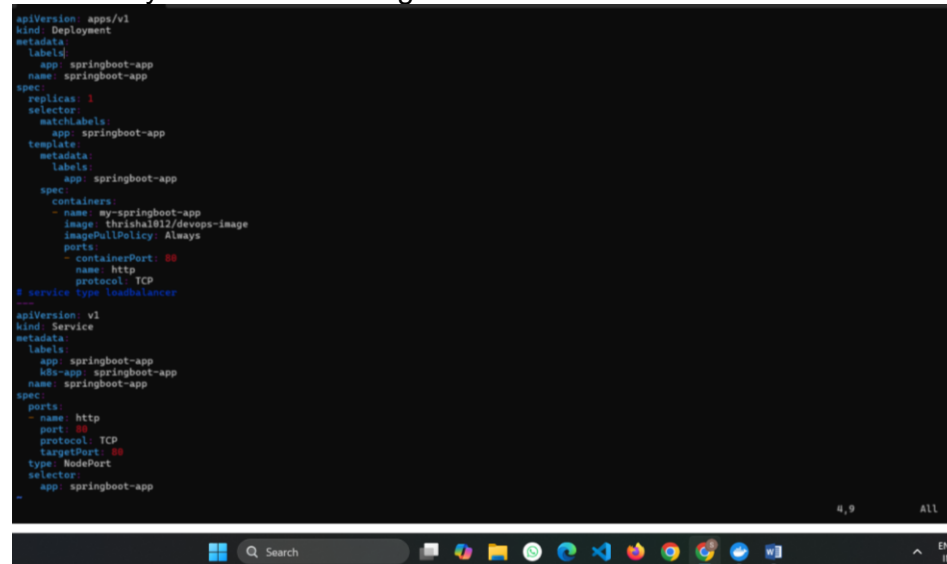
Create a new vim file named scripts.yaml



```
shalini@DESKTOP-32J80T2:~$ mkdir task4
shalini@DESKTOP-32J80T2:~$ cd
shalini@DESKTOP-32J80T2:~$ cd task4
shalini@DESKTOP-32J80T2:~/task4$ vim task4.yaml
shalini@DESKTOP-32J80T2:~/task4$ kubectl apply -f task4.yaml
deployment.apps/springboot-app configured
service/springboot-app unchanged
shalini@DESKTOP-32J80T2:~/task4$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
petclinic-64b57946cd-rjp2n         1/1     Running   0           15m
springboot-app-5b8fc4d9d-9wsgx     1/1     Running   0           13s
shalini@DESKTOP-32J80T2:~/task4$ minikube service springboot-app
NAMESPACE   NAME           TARGET PORT   URL
default     springboot-app http/8080      http://192.168.49.2:31096
Starting tunnel for service springboot-app.
NAMESPACE   NAME           TARGET PORT   URL
default     springboot-app http/8080      http://127.0.0.1:46541
Opening service default/springboot-app in default browser...
http://127.0.0.1:46541
! Because you are using a Docker driver on linux, the terminal needs to be open to run it.
^C Stopping tunnel for service springboot-app.
shalini@DESKTOP-32J80T2:~/task4$ vim task4.yaml
shalini@DESKTOP-32J80T2:~/task4$ kubectl apply -f task4.yaml
deployment.apps/springboot-app configured
```

Step 4: Yaml file

Enter the yaml file code using the insert



```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: springboot-app
  name: springboot-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: springboot-app
  template:
    metadata:
      labels:
        app: springboot-app
    spec:
      containers:
        - name: my-springboot-app
          image: thrishal912/devops-image
          imagePullPolicy: Always
          ports:
            - containerPort: 80
              name: http
              protocol: TCP
      serviceType: loadbalancer
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: springboot-app
  name: springboot-app
spec:
  ports:
    - name: http
      port: 80
      protocol: TCP
      targetPort: 80
  type: NodePort
  selector:
    app: springboot-app
```

Step 5: Apply

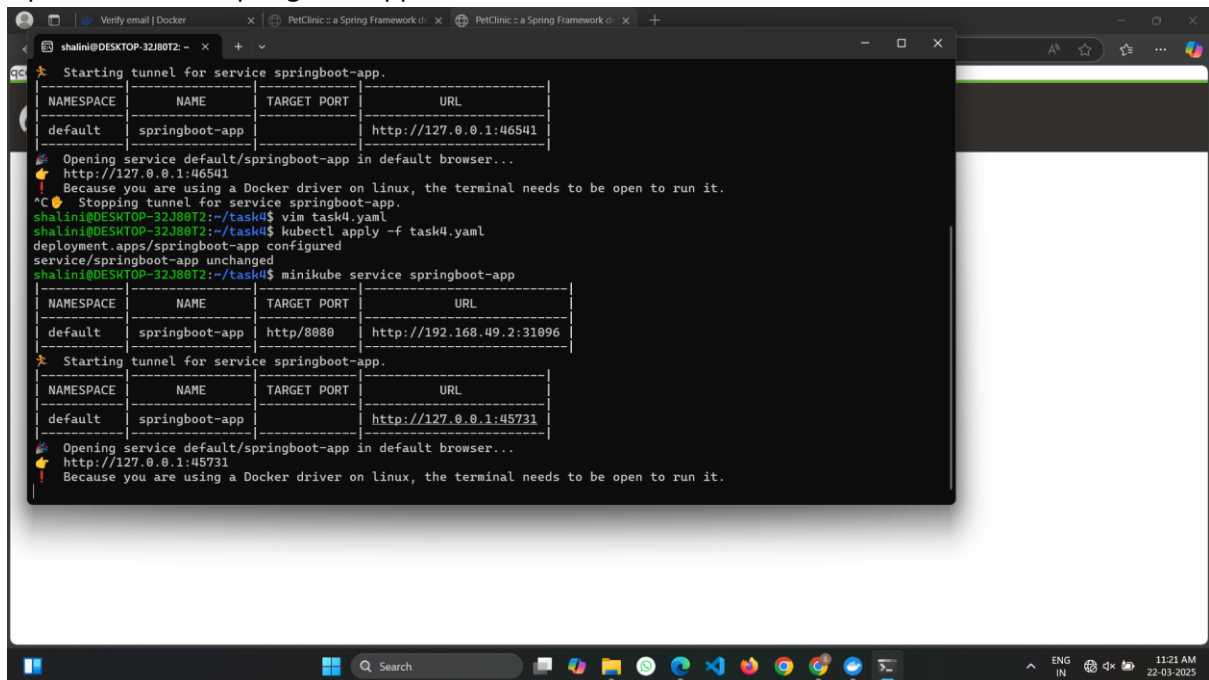
Apply the changes made in the devops.yaml file

Step 6: Get Pods

Get the pods information to check if it is running or not.

Step 7: Service

Open the service springboot-app in the browser



```
shalini@DESKTOP-32J80T2: ~  
* Starting tunnel for service springboot-app.  
NAMESPACE   NAME       TARGET PORT   URL  
-----  
default     springboot-app   http://127.0.0.1:46541  
* Opening service default/springboot-app in default browser...  
http://127.0.0.1:46541  
! Because you are using a Docker driver on linux, the terminal needs to be open to run it.  
^C * Stopping tunnel for service springboot-app.  
shalini@DESKTOP-32J80T2:~/task4$ vim task4.yaml  
shalini@DESKTOP-32J80T2:~/task4$ kubectl apply -f task4.yaml  
deployment.apps/springboot-app configured  
service/springboot-app unchanged  
shalini@DESKTOP-32J80T2:~/task4$ minikube service springboot-app  
NAMESPACE   NAME       TARGET PORT   URL  
-----  
default     springboot-app   http://192.168.49.2:31096  
* Starting tunnel for service springboot-app.  
NAMESPACE   NAME       TARGET PORT   URL  
-----  
default     springboot-app   http://127.0.0.1:45731  
* Opening service default/springboot-app in default browser...  
http://127.0.0.1:45731  
! Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```

Step 8: Output

The output is shown in the browser in the localhost url present

