

Deploying Spring Boot Application on Kubernetes with Minikube

This document provides step-by-step instructions to deploy a Spring Boot application using Kubernetes (Minikube) and expose it using a NodePort service.

Step 1: Install Minikube & Kubectl

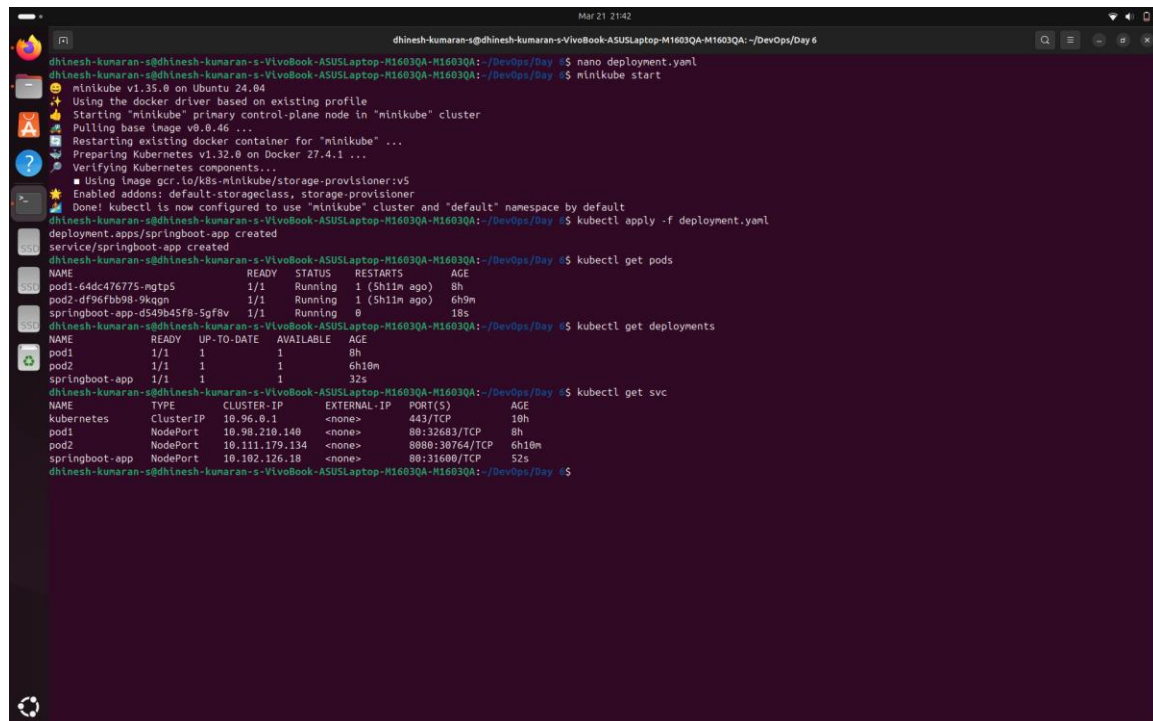
Ensure Minikube and kubectl are installed on your system. Run the following commands:

- Install Minikube:

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

- Install kubectl:

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
```



```
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6 $ nano deployment.yaml
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6 $ minikube start
minikube v1.35.0 on Ubuntu 24.04
Using the docker driver based on existing profile
Starting "minikube" primary control-plane node in "minikube" cluster
Pulling base image v0.0.46 ...
Restarting existing docker container for "minikube" ...
Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
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
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6 $ kubectl apply -f deployment.yaml
deployment.apps/springboot-app created
service/springboot-app created
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6 $ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
pod1-64dc476775-ngtp5              1/1     Running   1 (31m ago)   8h
pod2-df96fbb98-9kqgn              1/1     Running   1 (31m ago)   6h9m
springboot-app-d549b45f8-5gf8v     1/1     Running   0           18s
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6 $ kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
pod1    1/1     1             1           8h
pod2    1/1     1             1           6h10m
springboot-app  1/1     1             1           32s
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6 $ kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes      ClusterIP   10.96.0.1        <none>         443/TCP           18h
pod1            NodePort    10.98.210.140    <none>         80:32683/TCP      8h
pod2            NodePort    10.111.179.134   <none>         8080:30764/TCP    6h10m
springboot-app  NodePort    10.102.126.18    <none>         80:31600/TCP      52s
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6 $
```

Step 2: Start Minikube

Start a Kubernetes cluster using Minikube with Docker as the driver.

- Command:

```
minikube start --driver=docker
```

Step 3: Deploy the Application

Save the following YAML configuration as deployment.yaml and apply it using kubectl.

```
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: ar8888/ar
          imagePullPolicy: Always
          ports:
            - containerPort: 80
              name: http
              protocol: TCP
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: springboot-app
    k8s-app: springboot-app
  name: springboot-app
spec:
```

ports:

- name: http

port: 80

protocol: TCP

targetPort: 80

type: NodePort

selector:

app: springboot-app

- Apply the configuration using:

kubectl apply -f deployment.yaml

Step 4: Verify Deployment

- Check if the pods are running:

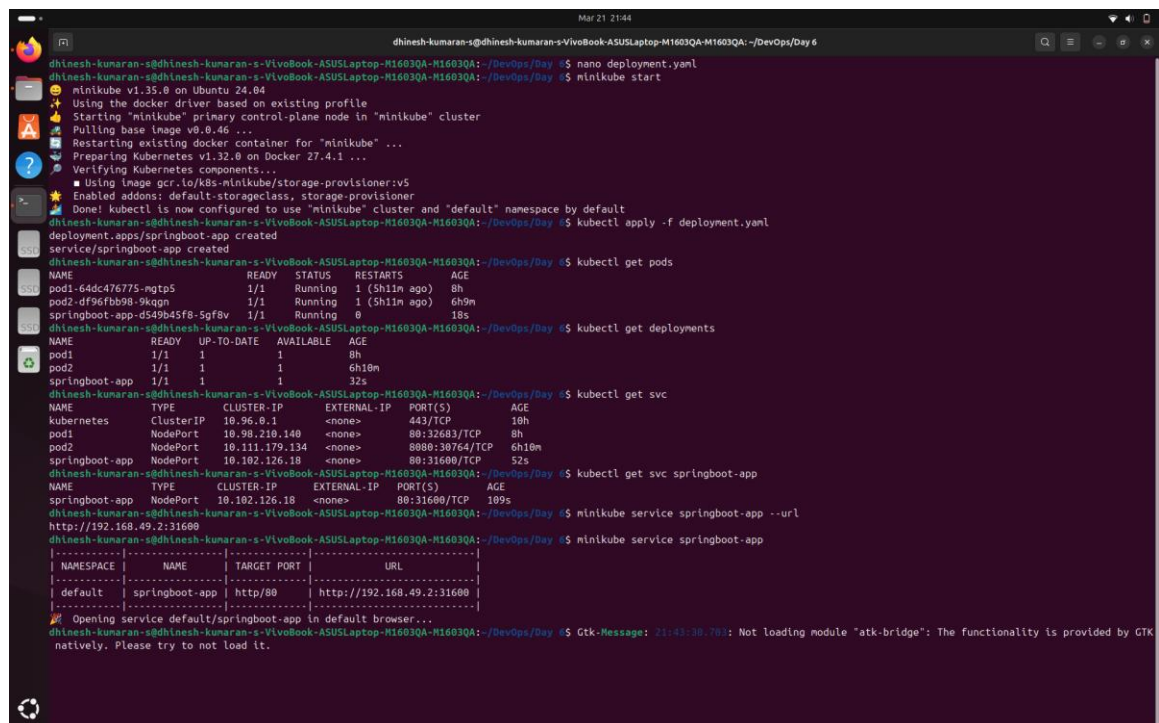
kubectl get pods

- Check deployment status:

kubectl get deployments

- Check service details:

kubectl get svc



```
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6$ nano deployment.yaml
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6$ minikube start
minikube v1.35.0 on Ubuntu 24.04
Using the docker driver based on existing profile
Starting "minikube" primary control-plane node in "minikube" cluster
Pulling base image v0.0.46...
Restarting existing docker container for "minikube" ...
Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
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
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6$ kubectl apply -f deployment.yaml
deployment.apps/springboot-app created
service/springboot-app created
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
pod1-64dc476775-mgtp5               1/1     Running   1 (311m ago)   8h
pod2-df96fbb98-9kqgn                1/1     Running   1 (311m ago)   6h9m
springboot-app-d549b45f8-5gf8v      1/1     Running   0             18s
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6$ kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
pod1    1/1     1             1            8h
pod2    1/1     1             1            6h10m
springboot-app  1/1     1             1            32s
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6$ kubectl get svc
NAME    TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes  ClusterIP   10.96.0.1      <none>          443/TCP           18h
pod1      NodePort    10.98.219.140  <none>          80:32683/TCP      8h
pod2      NodePort    10.111.179.134 <none>          8080:30764/TCP    6h18m
springboot-app  NodePort    10.102.126.18  <none>          80:31600/TCP      52s
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6$ kubectl get svc springboot-app
NAME    TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
springboot-app  NodePort    10.102.126.18  <none>          80:31600/TCP      109s
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6$ minikube service springboot-app --url
http://192.168.49.2:31600
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6$ minikube service springboot-app
|-----|
| NAMESPACE | NAME          | TARGET PORT | URL                  |
|-----|
| default   | springboot-app | http/80      | http://192.168.49.2:31600 |
|-----|
Opening service default/springboot-app in default browser...
dhinesh-kumaran-s@dhinesh-kumaran-s-VivoBook-ASUSLaptop-M1603QA-M1603QA: ~/DevOps/Day 6$ Gtk-Message: 21:45:38.783: Not loading module "atk-bridge": The functionality is provided by GTK
natively. Please try to not load it.
```

Step 5: Access the Application

- Get the assigned NodePort:

```
kubectl get svc springboot-app
```

- Get the Minikube service URL:

```
minikube service springboot-app --url
```

Debugging Commands

- View pod logs:

```
kubectl logs -f <pod-name>
```

- Delete deployment:

```
kubectl delete deployment springboot-app
```

- Delete service:

```
kubectl delete svc springboot-app
```

