

Title: Deployment of Spring Boot Application on Kubernetes Using Helm

Introduction

- Brief introduction to Spring Boot and Kubernetes.
- Importance of container orchestration and the role of Helm in simplifying deployments.
- Overview of what will be covered in the blog post.

Prerequisites

- Basic understanding of Spring Boot, Docker, Kubernetes, and Helm.
- Tools and technologies required:
 - Spring Boot application
 - Docker
 - Kubernetes cluster (Minikube or any cloud provider)
 - Helm
 - Jenkins for CI/CD (optional)

Step 1: Setting Up the Spring Boot Application

- Create a simple Spring Boot application.
- Add a simple REST controller that returns a "Hello World!" message.
- Test the application locally to ensure it's working.

- 1) mvn compile
- 2) mvn package

Step 2: Dockerizing the Spring Boot Application

- Create a Dockerfile for the Spring Boot application.
- Build the Docker image locally.
- Push the Docker image to DockerHub (or any container registry).
- Example Dockerfile



```
1 FROM openjdk:17-jdk
2
3 WORKDIR /testhello
4
5 ADD target/testhello-0.0.1-SNAPSHOT.jar testhello.jar
6
7 EXPOSE 8040
8
9 ENTRYPOINT ["java", "-jar", "testhello.jar"]
```

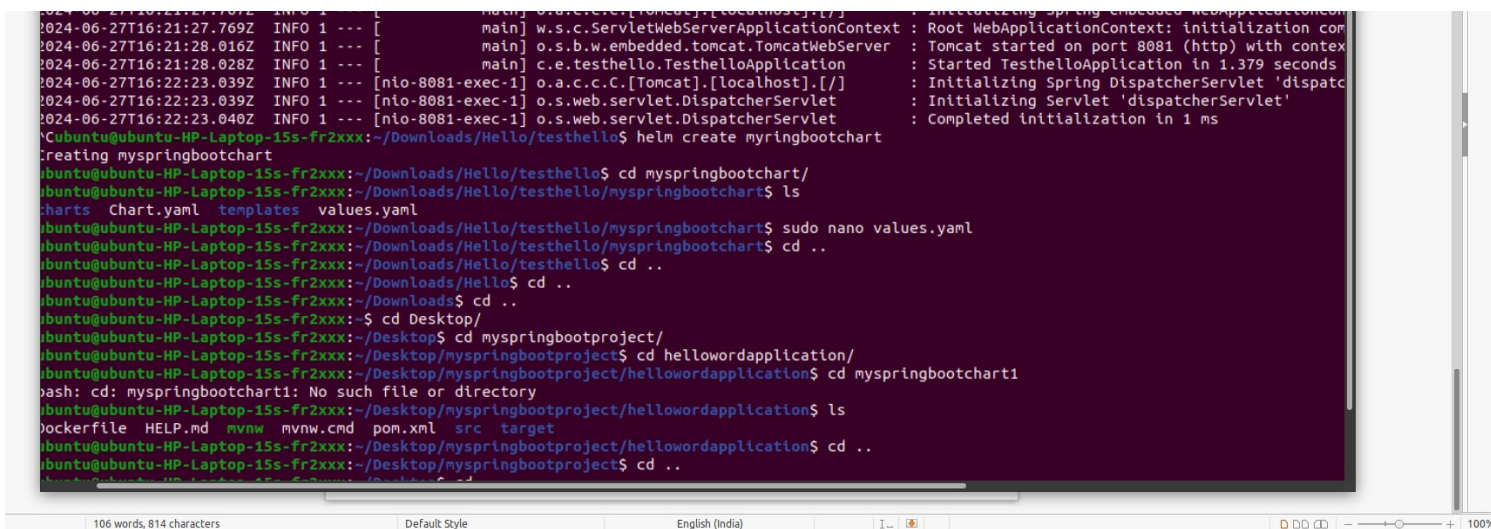
```
$ docker run -p 8040:8081 testhello
```



- Set up a local Kubernetes cluster using Minikube or use a cloud provider (e.g., GKE, EKS, AKS).
- Ensure `kubectl` is configured to interact with the cluster.
- `$ minikube start`
- `$ minikube status`

Step 4: Creating Helm Charts

- Introduction to Helm and its components (Chart.yaml, values.yaml, templates, etc.).
- Create a Helm chart for the Spring Boot application.
- **\$ helm create myspringbootchart**
 - Define the deployment and service templates.
 - Customize `values.yaml` for different environments (e.g., development, production).



Edit this part:

image:

repository: pramila188/testhello

tag: latest

pullPolicy: IfNotPresent

service:

type: NodePort

port: 8081

Example Helm chart structure:

```
my-springboot-chart/
├── Chart.yaml
├── values.yaml
└── templates/
    ├── deployment.yaml
    └── service.yaml
```

```
$ eval $(minikube docker -env)
```

again run docker build and run command

Step 5: Deploying the Application Using Helm

- Package the Helm chart.
- Install the Helm chart on the Kubernetes cluster.
- Verify the deployment and ensure the application is running.

```
$ helm install hellochart myspringbootchart
```

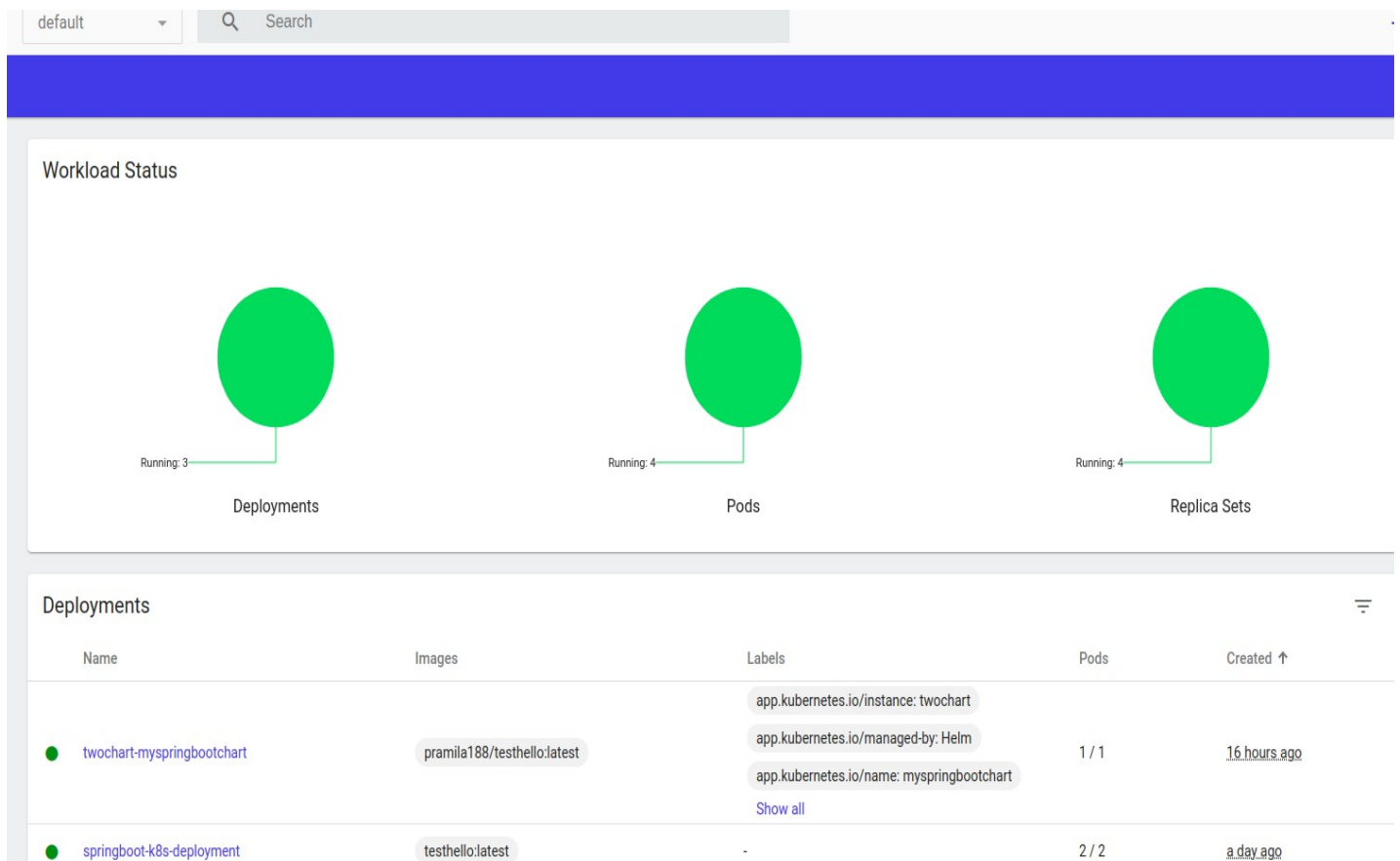
```
$ kubectl get pod
```

```
$ kubectl get svc
```

```
$ kubectl get deployment
```

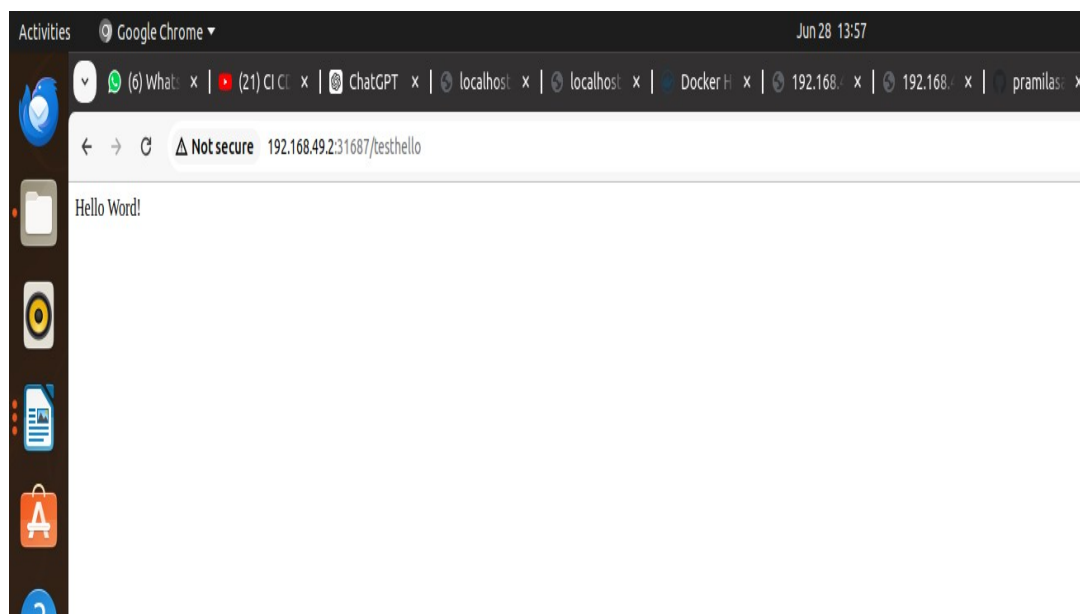
```
NAME: onechart
LAST DEPLOYED: Fri Jun 28 11:04:36 2024
NAMESPACE: test
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  http://chart-example.local/
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/Downloads/Hello/testhello$ kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
minikube            Ready     control-plane  7d18h  v1.30.0
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/Downloads/Hello/testhello$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
hellochart-myspringbootchart1-d88f85574-pckg6  1/1     Running   10 (13h ago)  7d18h
springboot-k8s-deployment-68dbc55d54-75jkn    1/1     Running   2 (13h ago)   23h
springboot-k8s-deployment-68dbc55d54-ncmrn    1/1     Running   3 (94m ago)   23h
twochart-myspringbootchart-597b9779b9-f7cfp    0/1     ImagePullBackOff  0           6m35s
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/Downloads/Hello/testhello$ kubectl get pods --test
error: unknown flag: --test
See 'kubectl get --help' for usage.
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/Downloads/Hello/testhello$ helm list
NAME          NAMESPACE    REVISION    UPDATED                               STATUS    CHART
hellochart    default       1            2024-06-20 16:30:52.022648932 +0530 IST deployed  myspringbootchart1
onechart      default       1            2024-06-27 23:27:24.963134825 +0530 IST deployed  myspringbootchart-
twochart      default       1            2024-06-27 23:40:11.749108258 +0530 IST deployed  myspringbootchart-
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/Downloads/Hello/testhello$ kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
hellochart-myspringbootchart1-d88f85574-pckg6  1/1     Running   10 (13h ago)  7d18h
springboot-k8s-deployment-68dbc55d54-75jkn    1/1     Running   2 (13h ago)   23h
springboot-k8s-deployment-68dbc55d54-ncmrn    1/1     Running   3 (98m ago)   23h
twochart-myspringbootchart-597b9779b9-f7cfp    0/1     ImagePullBackOff  0           9m52s
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/Downloads/Hello/testhello$ kubectl get pod -A
NAMESPACE    NAME                                READY   STATUS    RESTARTS   AGE
default      hellochart-myspringbootchart1-d88f85574-pckg6  1/1     Running   10 (13h ago)  7d18h
default      springboot-k8s-deployment-68dbc55d54-75jkn    1/1     Running   2 (13h ago)   23h
default      springboot-k8s-deployment-68dbc55d54-ncmrn    1/1     Running   3 (98m ago)   23h
default      twochart-myspringbootchart-597b9779b9-f7cfp    0/1     ImagePullBackOff  0           9m52s
```

```
$ minikube dashboard
```



\$ minikube service hellochart myspringbootchart --url

Final output:



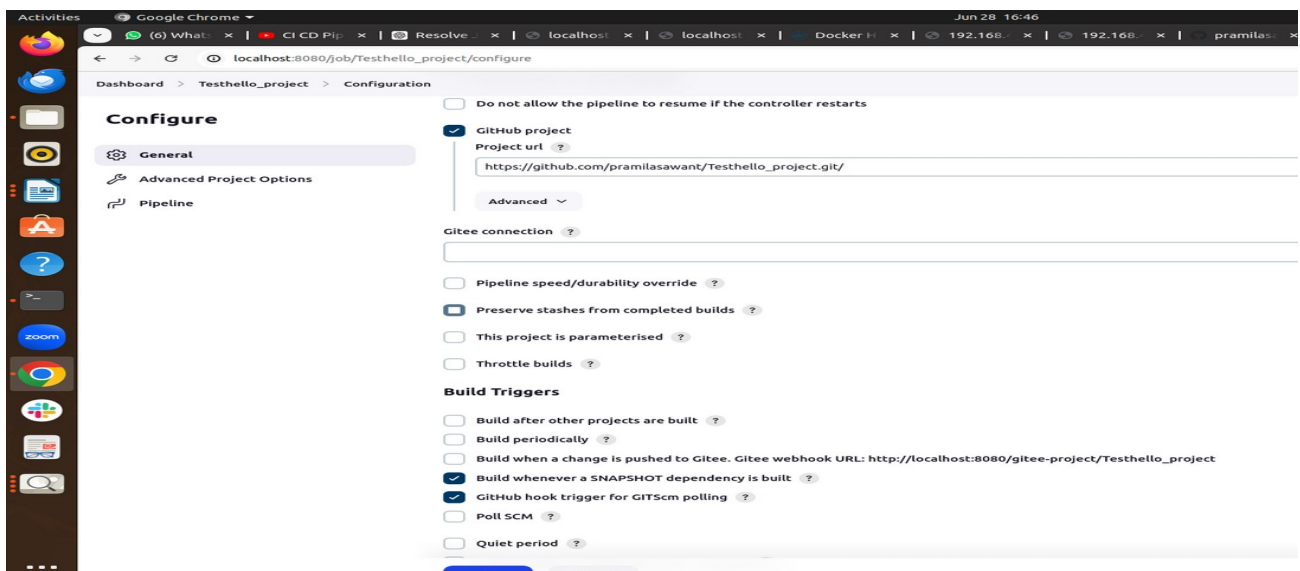
Push this project folder into new github repo.

Step 6: Setting Up Continuous Deployment with Jenkins (Optional)

- Introduction to Jenkins and its role in CI/CD.
- Create a Jenkins pipeline to automate the build, Docker image creation, and Helm deployment.

Open jenkins dashboard create new pipeline job with added plugin maven,Docker,kubernetes,kubernetes continous deployment etc...

create new pipeline job with adding project github url



1

Step by step run the script:

- 1) Clone code
- 2) build maven project
- 3) build and push docker image into dockerhub

Dashboard > Testhello_project > Configuration

Configure

Advanced ▾

- General
- Advanced Project Options**
- Pipeline

Pipeline

Definition

Pipeline script

```
1 pipeline {
2   agent any
3
4   tools{
5     maven'testhello'
6     // Define Docker installation name outside of any stage
7     dockerTool'testhello'
8   }
9
10  stages {
11    stage('Clone code') {
12      steps {
13        script {
14          echo 'Cloning code...'
15          git branch: 'main', url: 'https://github.com/pramilasawant/Testhello_project.git'
16        }
17      }
18    }
19
20    stage('Build Maven project') {
21      steps {
22        dir('/var/lib/jenkins/workspace/Testhello_project/Hello/testhello') {
23          echo 'Building Maven project...'
24          sh 'mvn clean install'
25        }
26      }
27    }
28
29    stage('Build Docker image') {
30      steps {
31        script {
32          echo 'Building dockerimage...'
33          // Use double quotes for variable substitution
34          docker build -t pramila188/testhello .
35        }
36      }
37    }
38
39    stage('Push Docker image') {
40      steps {
41        script {
42          echo 'Push dockerimage...'
43          // Use double quotes for variable substitution
44          docker push pramila188/testhello:latest
45        }
46      }
47    }
48  }
49 }
```

Save Apply

Pipeline script is as follows:

```
pipeline {
  agent any

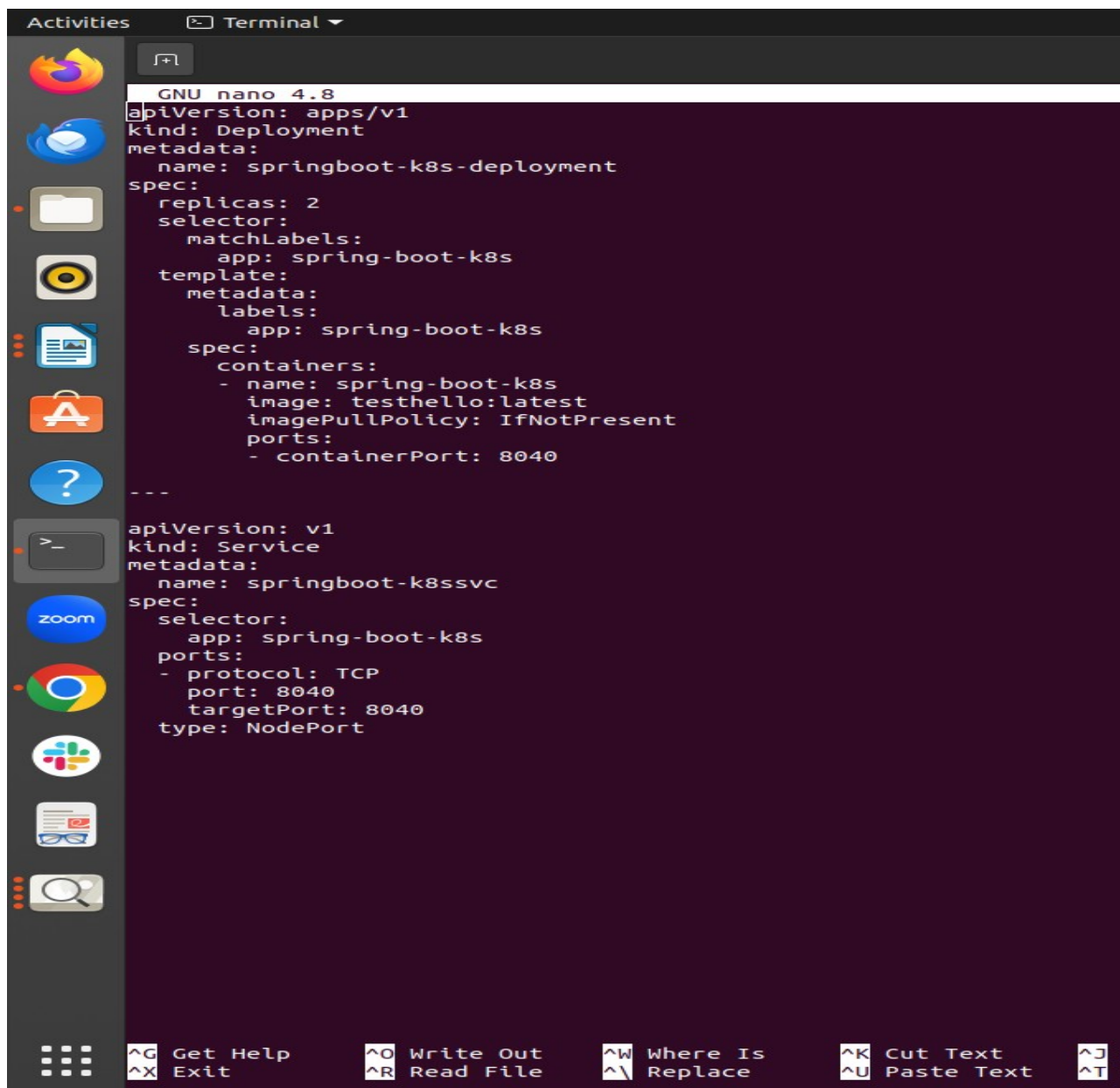
  tools{
    maven'testhello'
    // Define Docker installation name outside of any stage
    dockerTool'testhello'
  }

  stages {
    stage('Clone code') {
      steps {
        script {
          echo 'Cloning code...'
```

```

        git branch: 'main', url:
'https://github.com/pramilasawant/Testhello_project.git'
    }
}
stage('Build Maven project') {
    steps {
        dir('/var/lib/jenkins/workspace/Testhello_project/Hello/testhello')
{
        echo 'Building Maven project...'
        sh 'mvn clean install'
    }
}
stage('Build Docker image') {
    steps {
        script {
            echo 'Building dockerimage...'
            // Use double quotes for variable substitution
            'docker build -t pramila188/testhello .'
        }
    }
}
stage('Push Docker image') {
    steps {
        script {

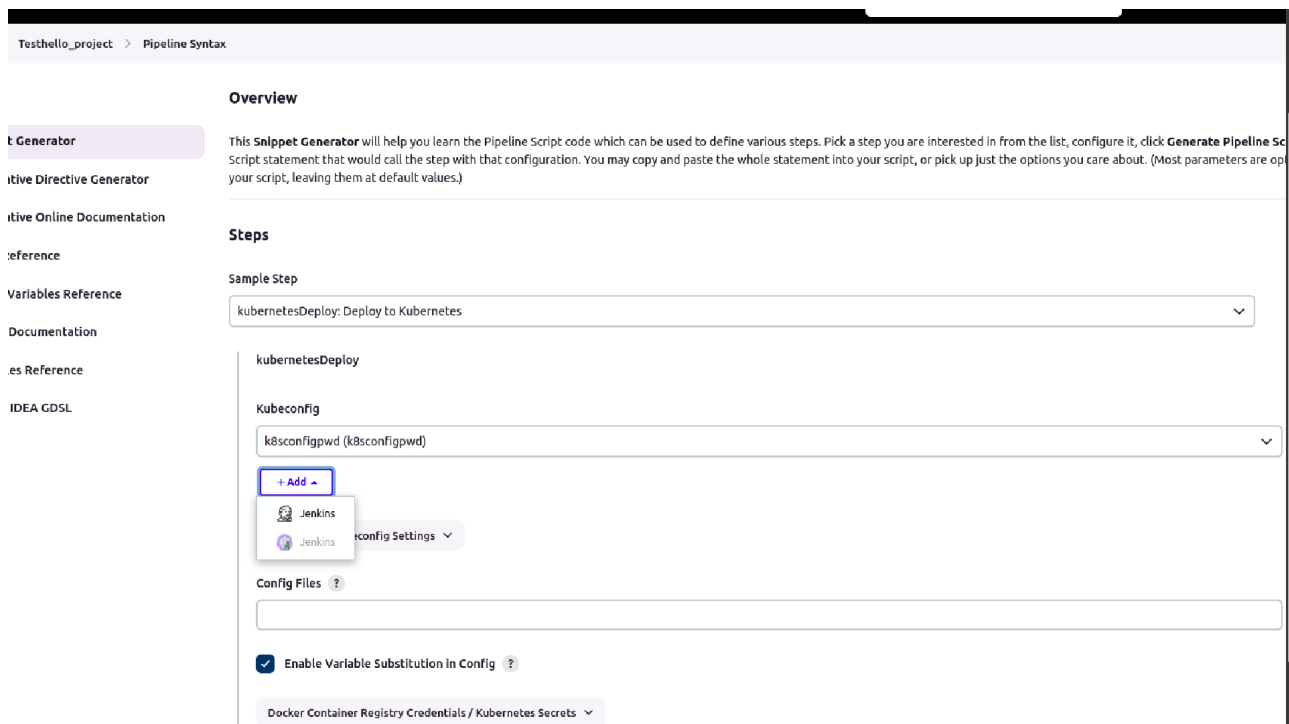
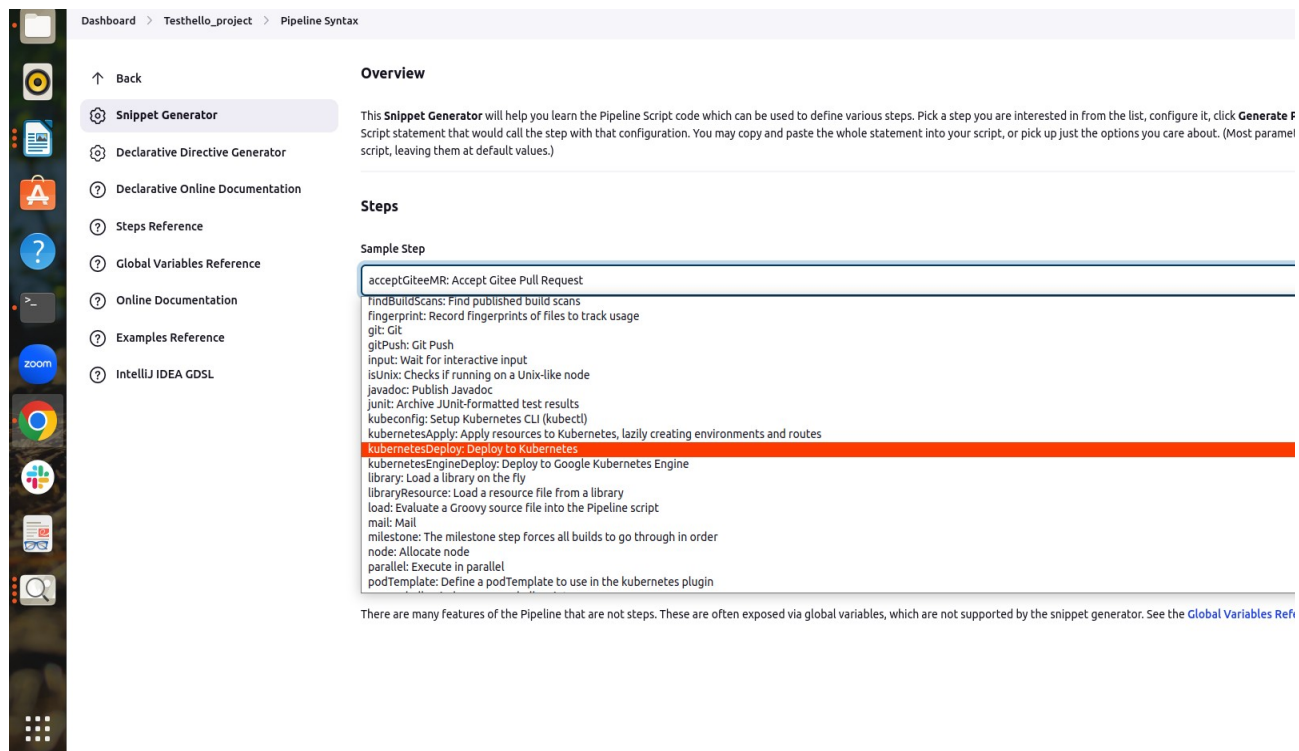
```

The image shows a terminal window with the GNU nano 4.8 editor. The editor contains two YAML configurations for a Kubernetes application. The first configuration is a Deployment named 'springboot-k8s-deployment' with 2 replicas, using the 'testhello:latest' image, and exposing port 8040. The second configuration is a Service named 'springboot-k8ssvc' of type 'NodePort', also exposing port 8040. The terminal window has a sidebar with application icons and a bottom status bar with nano editor shortcuts.

```
GNU nano 4.8
apiVersion: apps/v1
kind: Deployment
metadata:
  name: springboot-k8s-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: spring-boot-k8s
  template:
    metadata:
      labels:
        app: spring-boot-k8s
    spec:
      containers:
        - name: spring-boot-k8s
          image: testhello:latest
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 8040
---
apiVersion: v1
kind: Service
metadata:
  name: springboot-k8ssvc
spec:
  selector:
    app: spring-boot-k8s
  ports:
    - protocol: TCP
      port: 8040
      targetPort: 8040
  type: NodePort
```

Open this pipeline job and click on pipeline syntax
click into kubernetes Deploy:Deploy to kubernetes
in kubeconfig section click on jenkins and select kubeconfigpwd



Add the Credentials:

Username: Kubernetes configuration(Kubeconfig)

Password: kubeconfigpwd or any something

click on Enter Directly

open the terminal and go to root directory

\$ ls -la

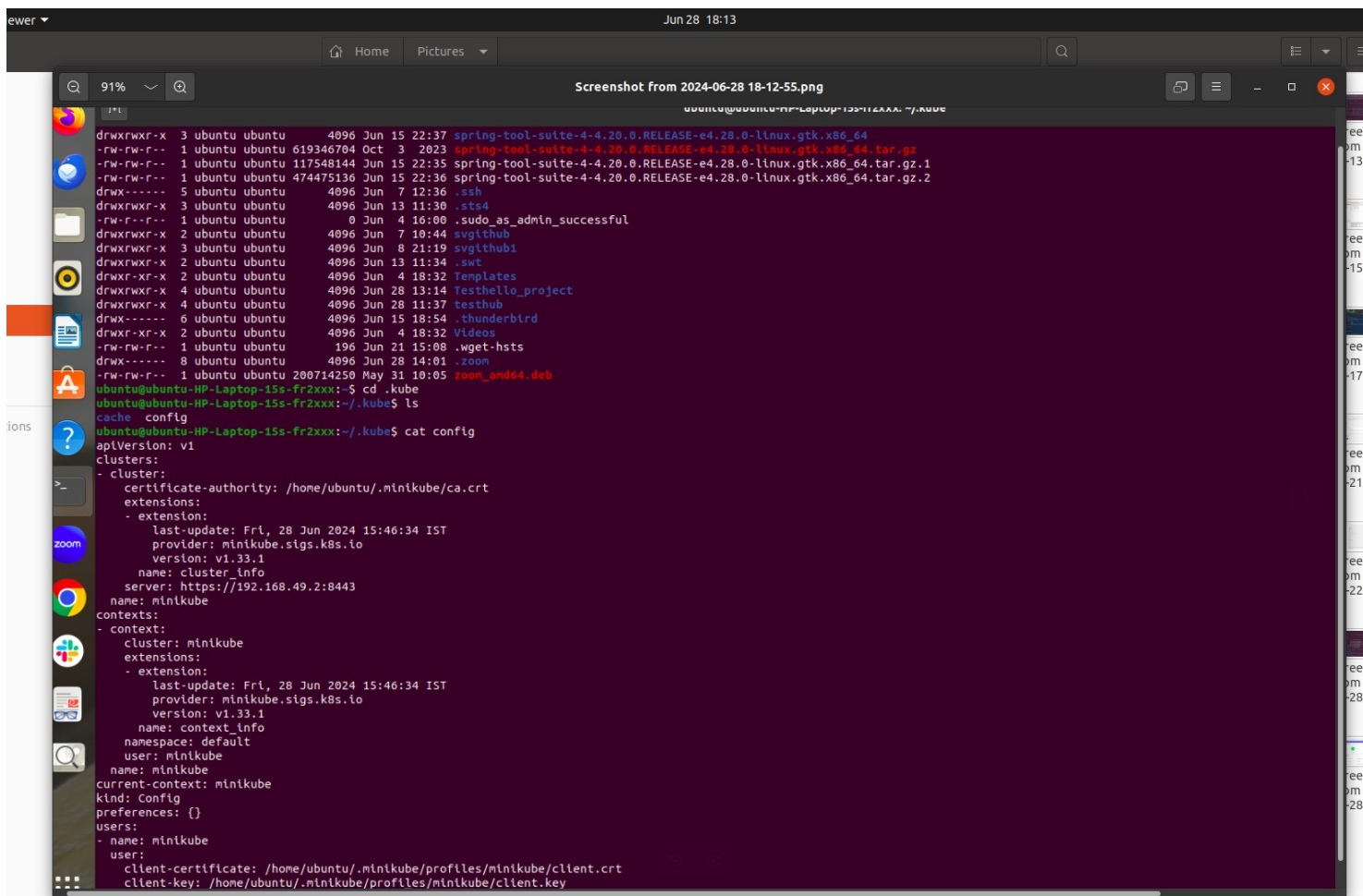
show all hidden fileopen the .kube

\$ cd .kube

\$ ls

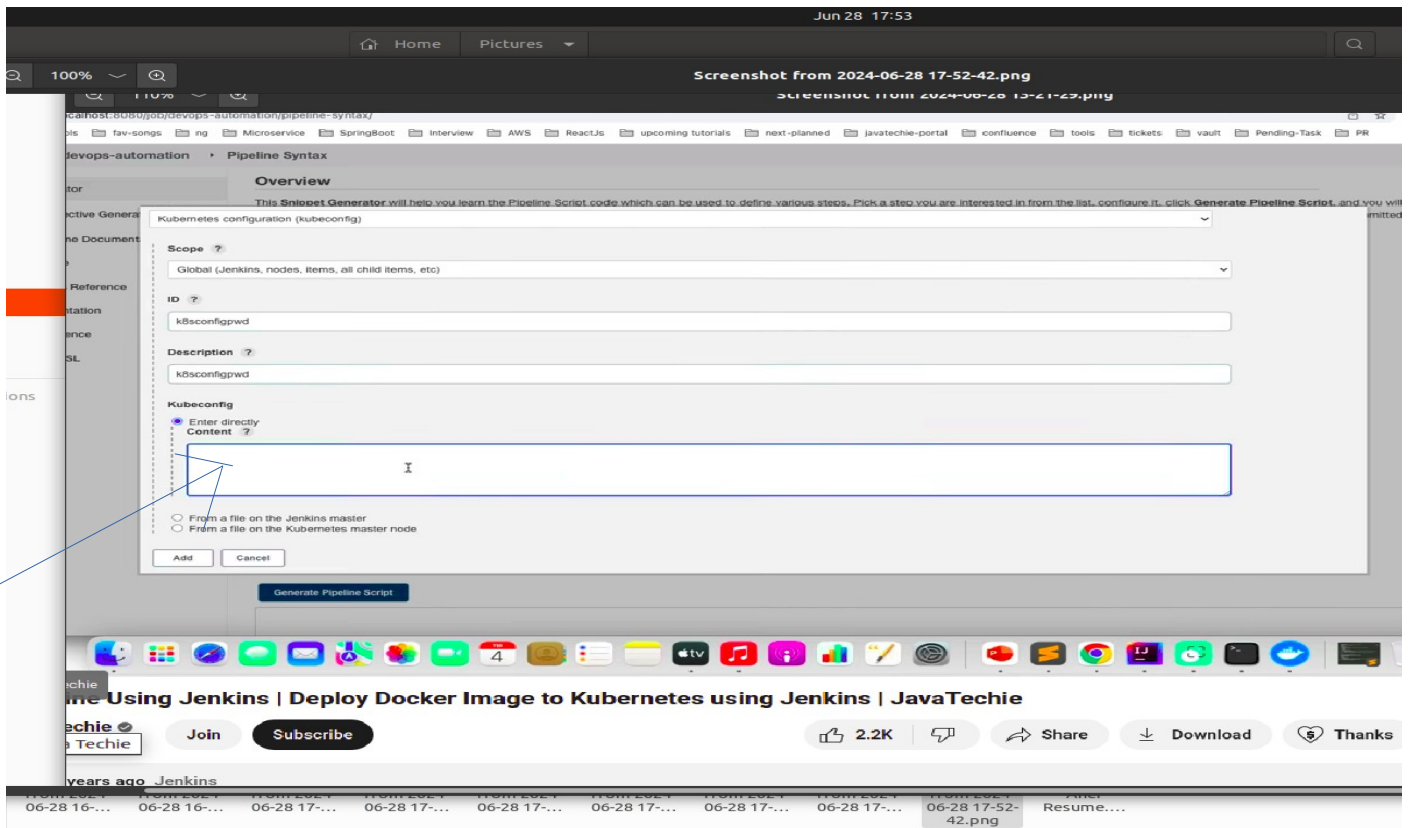
open config using

\$ cat config



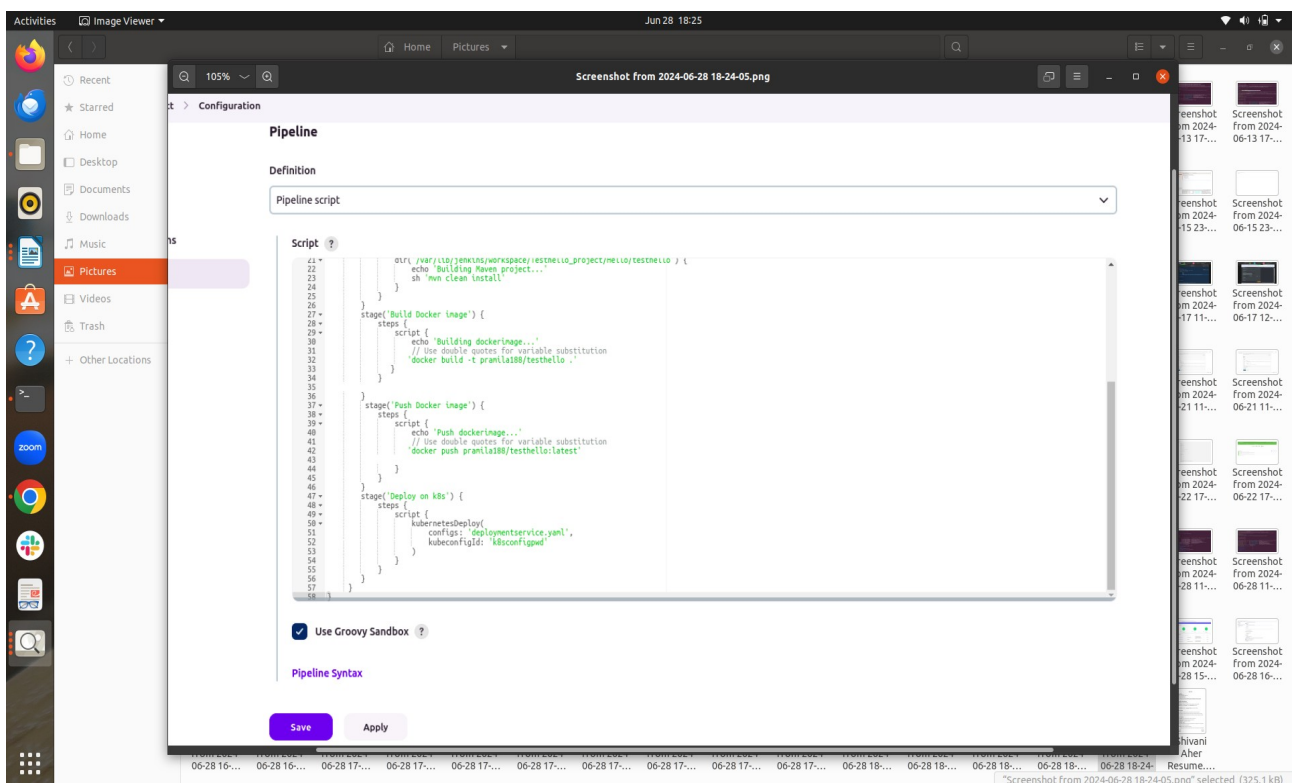
```
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/.kube$ cat config
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /home/ubuntu/.minikube/ca.crt
    extensions:
    - extension:
        last-update: Fri, 28 Jun 2024 15:46:34 IST
        provider: minikube.sigs.k8s.io
        version: v1.33.1
        name: cluster_info
        server: https://192.168.49.2:8443
      name: minikube
    contexts:
    - context:
        cluster: minikube
        extensions:
        - extension:
            last-update: Fri, 28 Jun 2024 15:46:34 IST
            provider: minikube.sigs.k8s.io
            version: v1.33.1
            name: context_info
            namespace: default
            user: minikube
          name: minikube
        current-context: minikube
      kind: Config
      preferences: {}
      users:
      - name: minikube
        user:
          client-certificate: /home/ubuntu/.minikube/profiles/minikube/client.crt
          client-key: /home/ubuntu/.minikube/profiles/minikube/client.key
```

copy this config file and into highlighted box



click on Add and Generate pipeline script

copy this content and paste into k8s deployment script



save the script and Build now

when build success open terminal

\$ kubectl get pod

\$ kubectl get node

\$ kubectl get deployment

\$ kubectl get svc

\$ minikube service service name --url

```
Activities Terminal Jun 28 13:57
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx: ~/kube
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~$ cd .kube
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/kube$ kubectl get nodes
NAME STATUS ROLES AGE VERSION
minikube Ready control-plane 7d21h v1.30.0
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/kube$ kubectl get pods
NAME READY STATUS RESTARTS AGE
hellochart-myspringbootchart1-d88f85574-pckg6 1/1 Running 10 (15h ago) 7d20h
springboot-k8s-deployment-6d5488b4f5-gr6nb 1/1 Running 0 4m41s
springboot-k8s-deployment-6d5488b4f5-pd4c5 1/1 Running 0 4m40s
twochart-myspringbootchart-597b9779b9-f7cfp 1/1 Running 0 146m
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/kube$ kubectl get svc
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
hellochart-myspringbootchart1 NodePort 10.108.252.148 <none> 1800:31865/TCP 7d20h
kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 7d21h
onechart-myspringbootchart NodePort 10.97.118.111 <none> 8081:31128/TCP 13h
springboot-k8ssvc NodePort 10.106.189.44 <none> 8040:30237/TCP 25h
twochart-myspringbootchart NodePort 10.99.181.191 <none> 8081:31687/TCP 13h
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/kube$ kubectl get deployment
NAME READY UP-TO-DATE AVAILABLE AGE
hellochart-myspringbootchart1 1/1 1 1 7d20h
springboot-k8s-deployment 2/2 2 2 25h
twochart-myspringbootchart 1/1 1 1 13h
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/kube$ kubectl logs twochart-myspringbootchart-597b9779b9-f7cfp
:: Spring Boot :: (v3.3.1)

2024-06-28T05:53:36.336Z INFO 1 --- [ main] c.e.testhello.TestHelloApplication : Starting TestHelloApplication v0.0.1-SNAPSHOT using Java 17.0.2 with PID 1 (/testhello/testhello.jar star
ted by root in /testhello)
2024-06-28T05:53:36.340Z INFO 1 --- [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : No active profile set, falling back to 1 default profile: "default"
2024-06-28T05:54:21.040Z INFO 1 --- [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8081 (http)
2024-06-28T05:54:22.338Z INFO 1 --- [ main] o.apache.catalina.core.StandardEngine : Starting service [Tomcat]
2024-06-28T05:54:22.339Z INFO 1 --- [ main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.25]
2024-06-28T05:54:28.536Z INFO 1 --- [ main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-06-28T05:54:28.537Z INFO 1 --- [ main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 49599 ms
2024-06-28T05:54:51.141Z INFO 1 --- [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8081 (http) with context path '/'
2024-06-28T05:54:53.239Z INFO 1 --- [ main] c.e.testhello.TestHelloApplication : Started TestHelloApplication in 83.199 seconds (process running for 115.878)
2024-06-28T05:54:53.938Z INFO 1 --- [nio-8081-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2024-06-28T05:54:53.938Z INFO 1 --- [nio-8081-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2024-06-28T05:54:54.037Z INFO 1 --- [nio-8081-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 98 ms
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/kube$ kubectl get svc
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
hellochart-myspringbootchart1 NodePort 10.108.252.148 <none> 1800:31865/TCP 7d21h
kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 7d21h
onechart-myspringbootchart NodePort 10.97.118.111 <none> 8081:31128/TCP 14h
springboot-k8ssvc NodePort 10.106.189.44 <none> 8040:30237/TCP 25h
twochart-myspringbootchart NodePort 10.99.181.191 <none> 8081:31687/TCP 14h
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/kube$ minikube service twochart-myspringbootchart --url
http://192.168.49.2:31687
ubuntu@ubuntu-HP-Laptop-15s-fr2xxx:~/kube$ kubectl get svc ns test
```

open this url shows final output:

