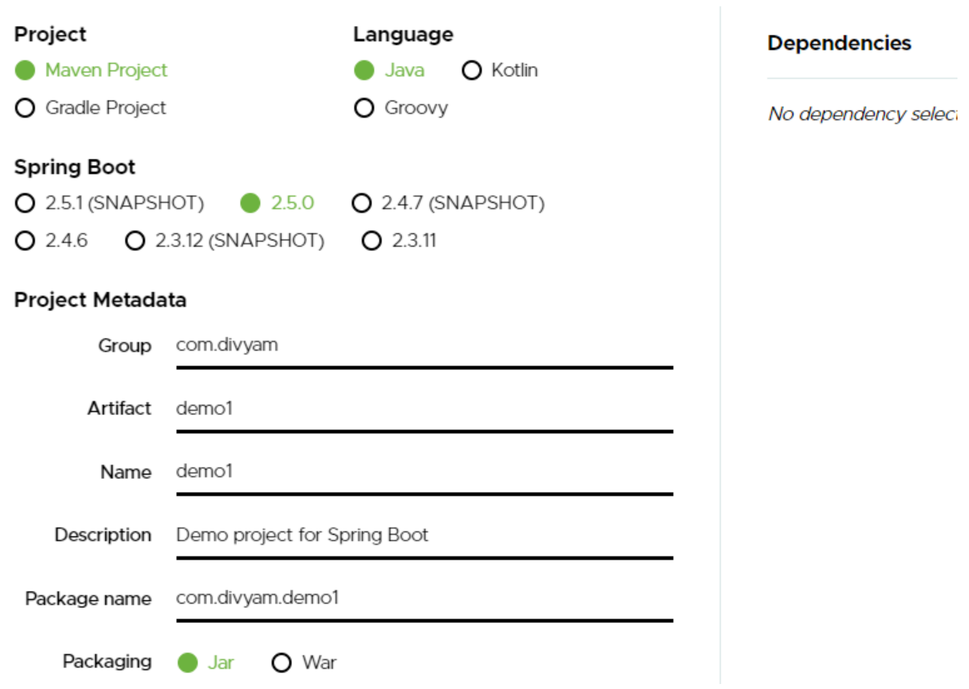
DEMO: CREATING JAVA APP, CONTAINERIZING WITH DOCKER & DEPLOYING ON KUBERNETES CLUSTER.

STEP 1: CREATE A SPRINGBOOT JAVA MAVEN PROJECT USING SPRING INITIALIZER

- Add the required dependencies e.g., SPRING WEB.
- Chose the java version according to the needs.

LINK-Spring Initializr



Project
[✓ Maven Project] Gradle Project

Language
[✓ Java] Kotlin Groovy

Spring Boot
2.5.1 (SNAPSHOT) [✓ 2.5.0] 2.4.7 (SNAPSHOT) 2.4.6 2.3.12 (SNAPSHOT) 2.3.11

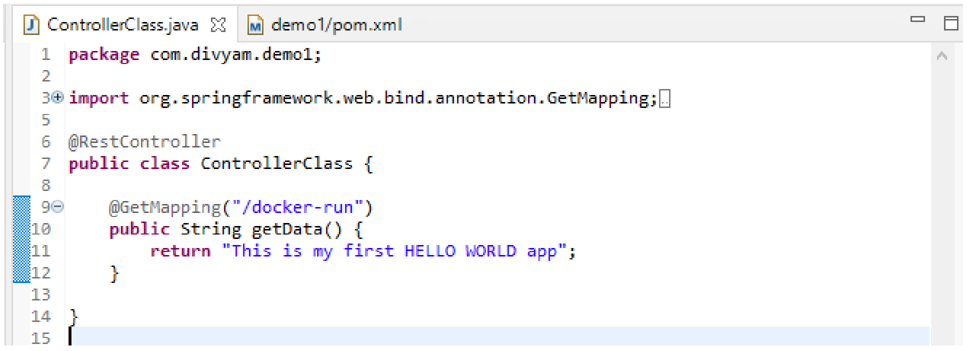
Project Metadata

Group	com.divyam
Artifact	demo1
Name	demo1
Description	Demo project for Spring Boot
Package name	com.divyam.demo1
Packaging	[✓ Jar] War

Dependencies

No dependency selec

This is a simple java application with functionality to display the string when user hits “docker-run” URL.

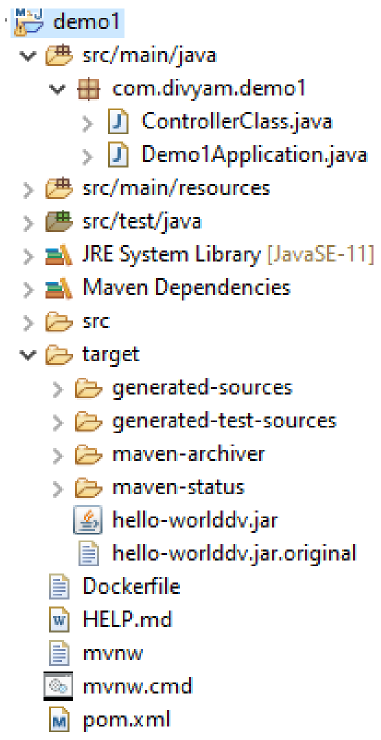


```
package com.divyam.demo1;

import org.springframework.web.bind.annotation.GetMapping;

@RestController
public class ControllerClass {

    @GetMapping("/docker-run")
    public String getData() {
        return "This is my first HELLO WORLD app";
    }

}
```

- Run and test the application on the port where it is running.

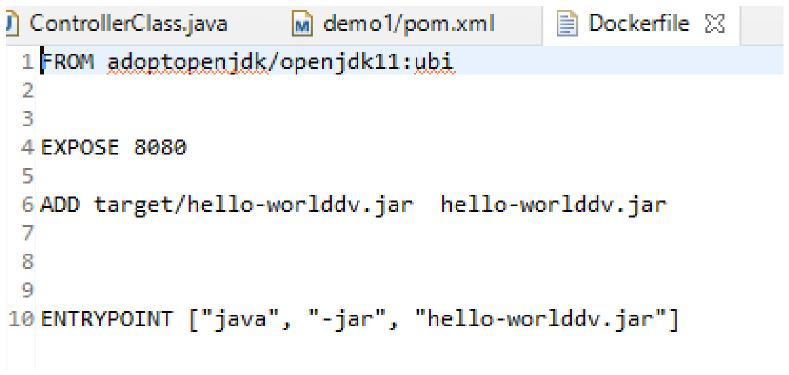
STEP 2: Getting the JAR in target folder using clean install command.

PROJECT-> RUN AS -> MAVEN BUILD -> GOALS (CLEAN INSTALL)



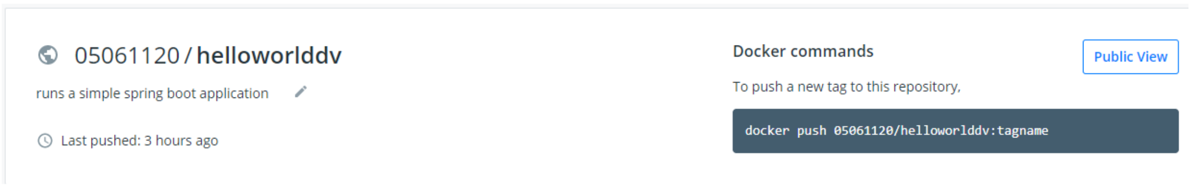
```
demo1
  src/main/java
    com.divyam.demo1
      ControllerClass.java
      Demo1Application.java
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-11]
  Maven Dependencies
  src
  target
    generated-sources
    generated-test-sources
    maven-archiver
    maven-status
    hello-worlddv.jar
    hello-worlddv.jar.original
  Dockerfile
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

STEP 3: CREATE A Docker file with this name and no extension. (Docker build reads from this file the instructions and builds the images accordingly).



```
FROM adoptopenjdk/openjdk11:ubi


EXPOSE 8080

ADD target/hello-worlddv.jar  hello-worlddv.jar



ENTRYPOINT ["java", "-jar", "hello-worlddv.jar"]
```

STEP 4: CREATE A DOCKER HUB REPOSITORY TO STORE YOUR IMAGE.



05061120 / helloworlddv

runs a simple spring boot application

Last pushed: 3 hours ago

Docker commands

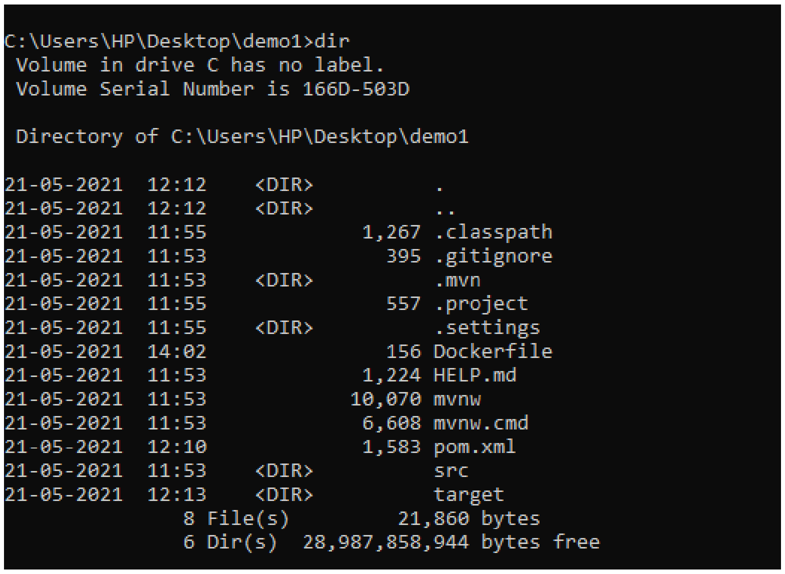
To push a new tag to this repository,

```
docker push 05061120/helloworlddv:tagname
```

Sample Repository where I will store my image.

STEP 5: BUILDING THE DOCKER IMAGE & PUSHING IT TO THE DOCKER HUB REPOSITORY.

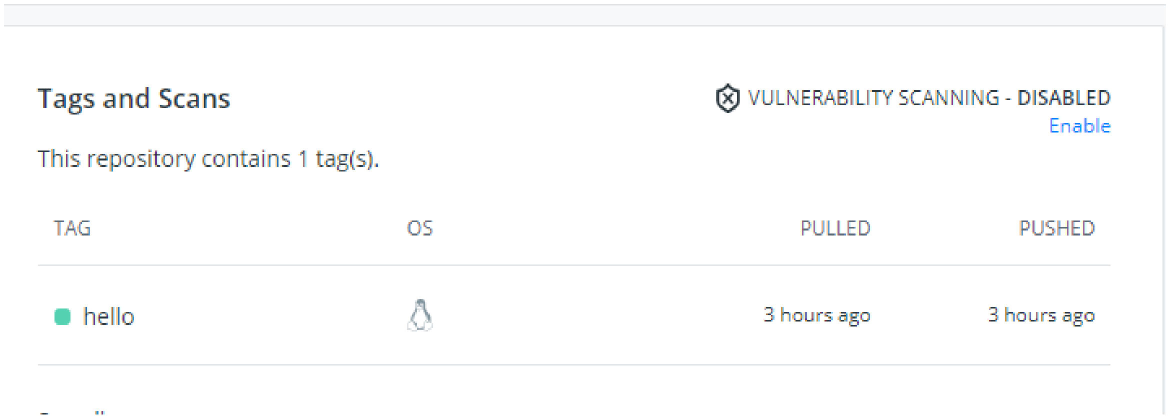
- Go into the directory where the Dockerfile is stored.
- Then build that docker file using the command.
    - //Docker build .



```
C:\Users\HP\Desktop\demo1>dir
 Volume in drive C has no label.
 Volume Serial Number is 166D-503D

 Directory of C:\Users\HP\Desktop\demo1

21-05-2021  12:12    <DIR>          .
21-05-2021  12:12    <DIR>          ..
21-05-2021  11:55             1,267 .classpath
21-05-2021  11:53               395 .gitignore
21-05-2021  11:53    <DIR>          .mvn
21-05-2021  11:55               557 .project
21-05-2021  11:55    <DIR>          .settings
21-05-2021  14:02               156 Dockerfile
21-05-2021  11:53             1,224 HELP.md
21-05-2021  11:53            10,070 mvnw
21-05-2021  11:53             6,608 mvnw.cmd
21-05-2021  12:10             1,583 pom.xml
21-05-2021  11:53    <DIR>          src
21-05-2021  12:13    <DIR>          target
               8 File(s)         21,860 bytes
               6 Dir(s)  28,987,858,944 bytes free
```

- Tag the image according to the version with the command
    - // docker tag <image name/id> <userid>/<reponame>: <tag name>
- Push the image to the repository:
    - docker push 05061120/helloworlddv:tagname

The image will look like this in the repository.



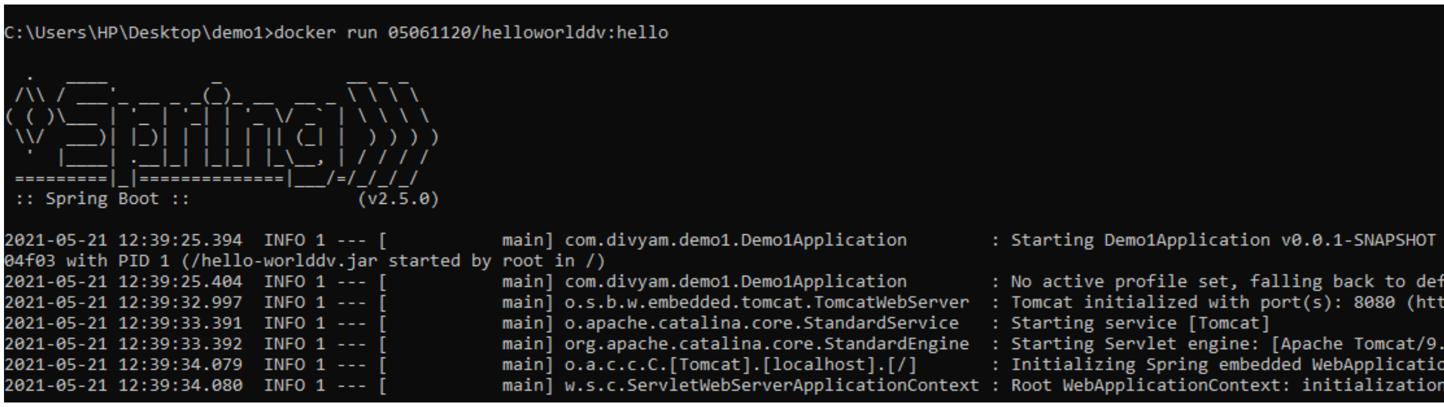
Tags and Scans

VULNERABILITY SCANNING - **DISABLED**
Enable

This repository contains 1 tag(s).

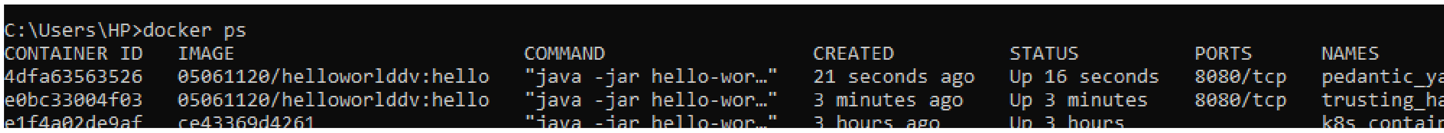
TAG	OS	PULLED	PUSHED
🟩 hello	🐧	3 hours ago	3 hours ago

- Delete the image from your local system. To check if the docker run command pulls it from docker hub repo.
- Docker run <image name>



```
C:\Users\HP\Desktop\demo1>docker run 05061120/helloworlddv:hello

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.5.0)

2021-05-21 12:39:25.394  INFO 1 --- [           main] com.divyam.demo1.Demo1Application        : Starting Demo1Application v0.0.1-SNAPSHOT
04f03 with PID 1 (/hello-worlddv.jar started by root in /)
2021-05-21 12:39:25.404  INFO 1 --- [           main] com.divyam.demo1.Demo1Application        : No active profile set, falling back to def
2021-05-21 12:39:32.997  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (htt
2021-05-21 12:39:33.391  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-05-21 12:39:33.392  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.
2021-05-21 12:39:34.079  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicatio
2021-05-21 12:39:34.080  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization
```

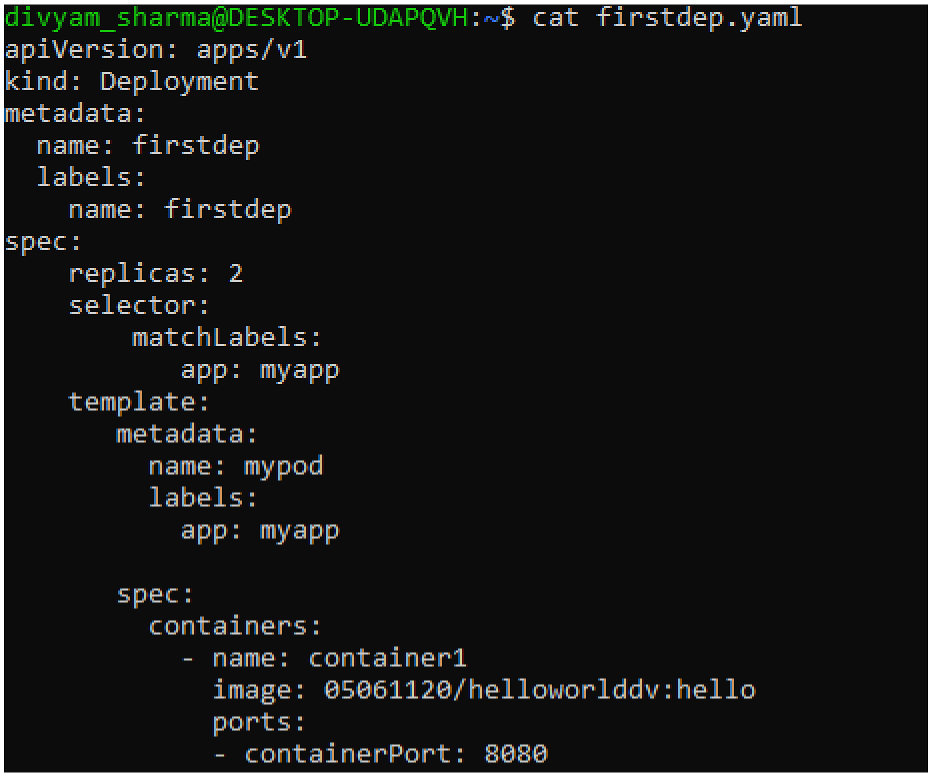
The image runs successfully!



```
C:\Users\HP>docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS      NAMES
4dfa63563526   05061120/helloworlddv:hello   "java -jar hello-wor…"   21 seconds ago   Up 16 seconds   8080/tcp   pedantic_ya
e0bc33004f03   05061120/helloworlddv:hello   "java -jar hello-wor…"   3 minutes ago    Up 3 minutes    8080/tcp   trusting_ha
e1f4a02de9af   ce43369d4261                  "java -jar hello-wor…"   3 hours ago      Up 3 hours                 k8s_contain
```

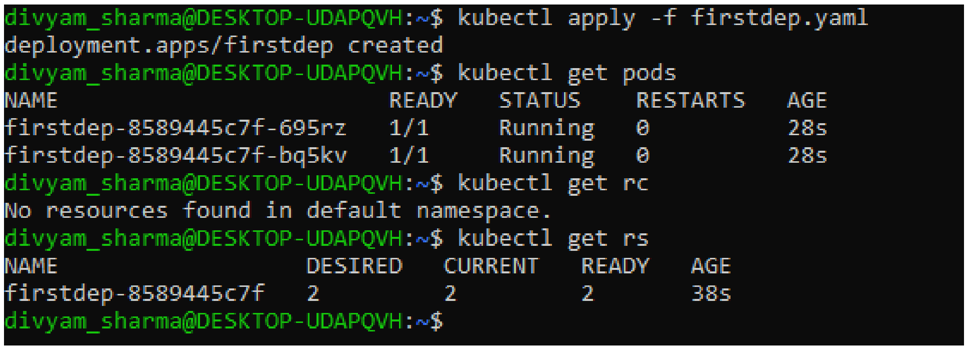
Container on which this image is running.

STEP 6: CREATING A DEPLOYMENT WITH YAML FILE AND DEPLOYING IT.



```
divyam_sharma@DESKTOP-UDAPQVH:~$ cat firstdep.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: firstdep
  labels:
    name: firstdep
spec:
    replicas: 2
    selector:
        matchLabels:
           app: myapp
    template:
       metadata:
         name: mypod
         labels:
           app: myapp

       spec:
         containers:
           - name: container1
             image: 05061120/helloworlddv:hello
             ports:
             - containerPort: 8080
```

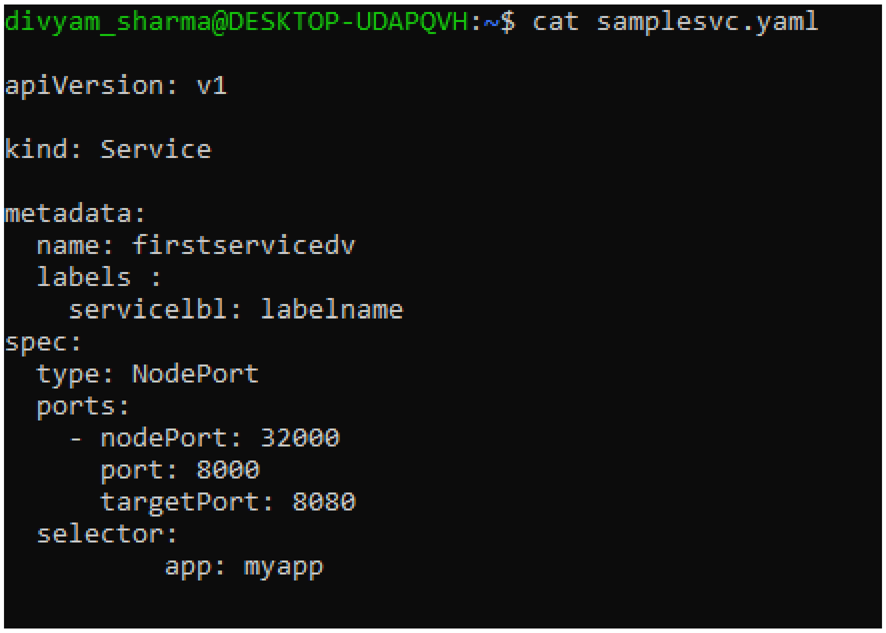
The yaml file which contains all the declarative details for the pod creation , replication control and port listing.



```
divyam_sharma@DESKTOP-UDAPQVH:~$ kubectl apply -f firstdep.yaml
deployment.apps/firstdep created
divyam_sharma@DESKTOP-UDAPQVH:~$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
firstdep-8589445c7f-695rz   1/1     Running   0          28s
firstdep-8589445c7f-bq5kv   1/1     Running   0          28s
divyam_sharma@DESKTOP-UDAPQVH:~$ kubectl get rc
No resources found in default namespace.
divyam_sharma@DESKTOP-UDAPQVH:~$ kubectl get rs
NAME                  DESIRED   CURRENT   READY   AGE
firstdep-8589445c7f   2         2         2       38s
divyam_sharma@DESKTOP-UDAPQVH:~$
```

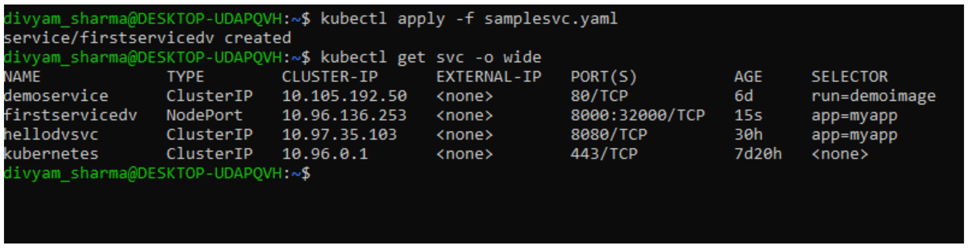
STEP 7: EXPOSING THE DEPLOYMENT TO A SERVICE

The yaml file for the svc looks like this.



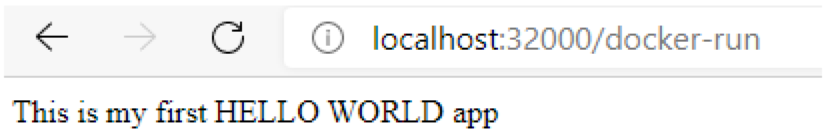
```
divyam_sharma@DESKTOP-UDAPQVH:~$ cat samplesvc.yaml

apiVersion: v1

kind: Service

metadata:
  name: firstservicedv
  labels :
    servicelbl: labelname
spec:
  type: NodePort
  ports:
    - nodePort: 32000
      port: 8000
      targetPort: 8080
  selector:
          app: myapp
```

Here we are using a service of type NodePort to expose the deployment on a node port which directs to another port which then sends the request to the containers port.



```
divyam_sharma@DESKTOP-UDAPQVH:~$ kubectl apply -f samplesvc.yaml
service/firstservicedv created
divyam_sharma@DESKTOP-UDAPQVH:~$ kubectl get svc -o wide
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE    SELECTOR
demoservice      ClusterIP   10.105.192.50   <none>        80/TCP           6d     run=demoimage
firstservicedv   NodePort    10.96.136.253   <none>        8000:32000/TCP   15s    app=myapp
hellodvsvc       ClusterIP   10.97.35.103    <none>        8080/TCP         30h    app=myapp
kubernetes       ClusterIP   10.96.0.1       <none>        443/TCP          7d20h  <none>
divyam_sharma@DESKTOP-UDAPQVH:~$
```

Let's try to access our application on port 32000 now with the given url.



localhost:32000/docker-run

This is my first HELLO WORLD app

THANKS!