**Step1**) I have cloned the repo from <https://github.com/spring-guides/gs-spring-boot.git>

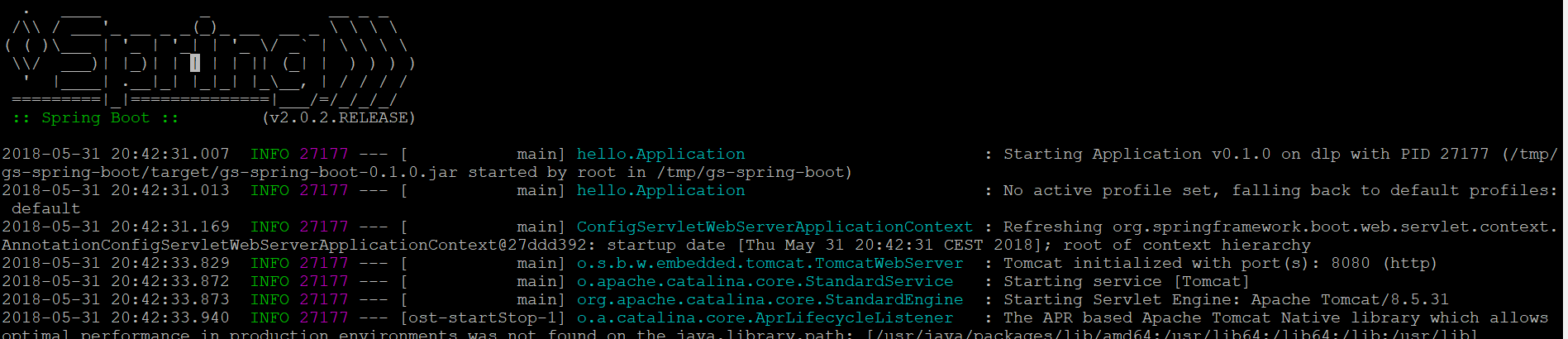
**Step2**) To run the springboot app I have created HelloController.java, Application.java & pom.xml as mentioned in the link <https://spring.io/guides/gs/spring-boot/>

**Step3**) Use below command to test the application and it will generate the gs-spring-boot-0.1.0.jar file under target directory.

a) cd gs-spring-boot

b) Run the mvn package && java -jar target/gs-spring-boot-0.1.0.jar

**Step4**) Once we run the command “mvn package && java -jar target/gs-spring-boot-0.1.0.jar “ springboot app will come up and start serving at port 8080



**Step5)** To containerize the springboot application I have created the below **Dockerfile** and build the image from it named as rsthakur83/springboot , also pushed the same image to my docker hub account so that it can be pulled from internet.

FROM java:8

VOLUME /tmp

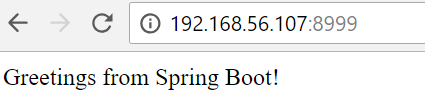
ADD target/gs-spring-boot-0.1.0.jar app.jar

RUN bash -c 'touch /app.jar'

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]

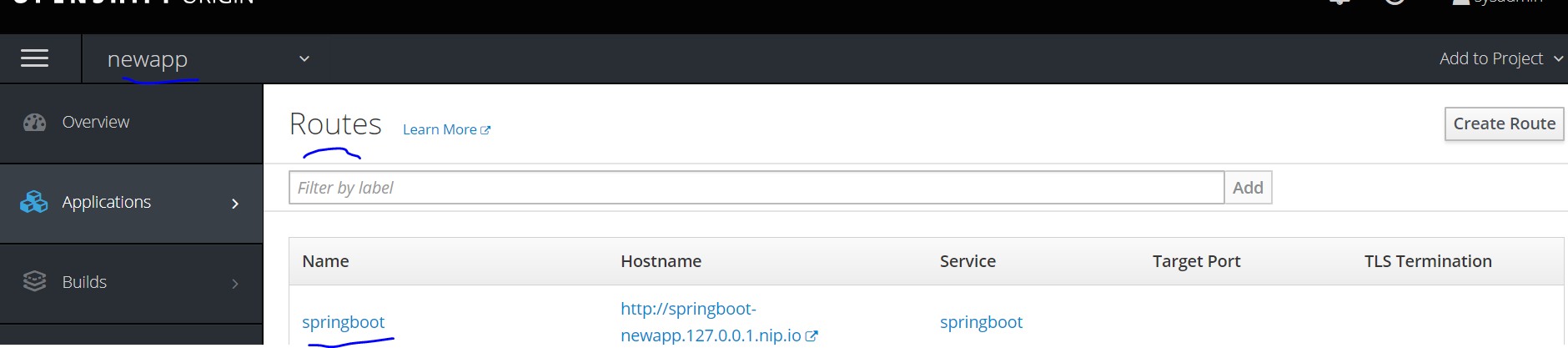
To test the springboot container I have run “docker run -p 8999:8080 rsthakur83/java-app” command and expose port **8080** to **8999 .**

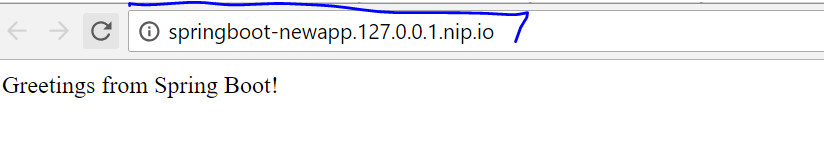




**Step6)** I have created the deployment & service config for the **openshift** environment and can be deployed using commands. Similar config can be used on kubernetes with slight changes in the service & deployment config files.

1. **oc create namespace newapp**
2. **cd gs-spring-boo**
3. **oc create –f springboot.yaml**
4. **oc create –f service.yaml**
5. **Create the external route for the service**
6. **Click on the url/hostname and we can access the springboot app**





**NOTE:** Anyone can run this after installing docker on machine-> **docker run -p 8999:8080 rsthakur83/java-app**