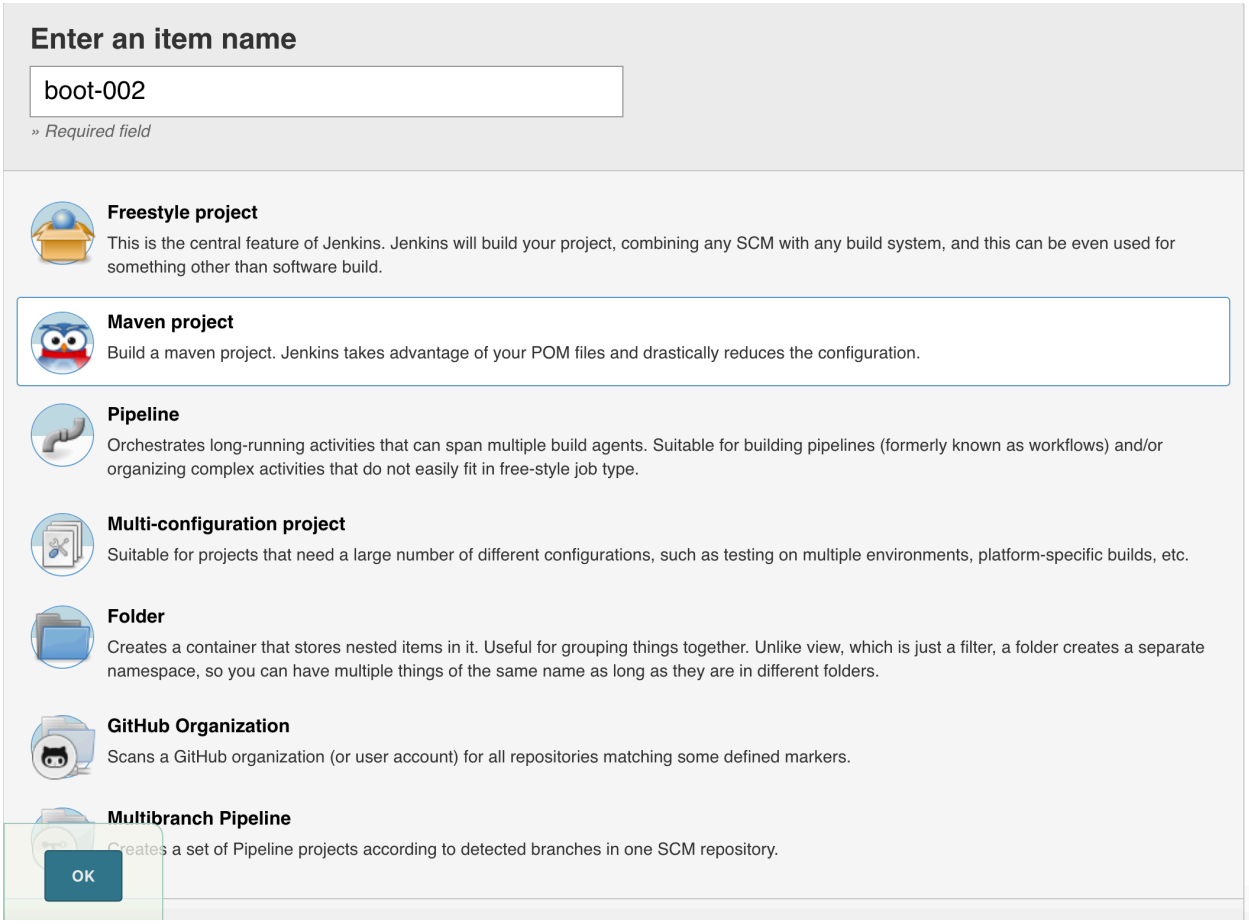
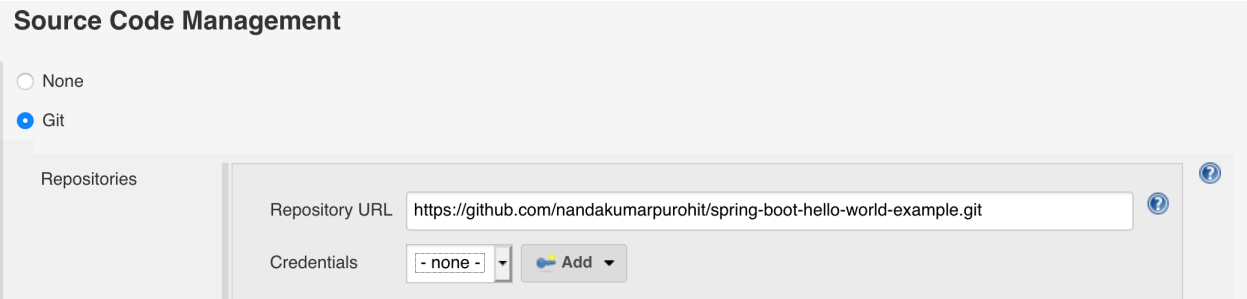
Configure Jenkins Job to create Docker Image and spin up Docker Container for your Spring Boot Application

1. Click on **New Item** link

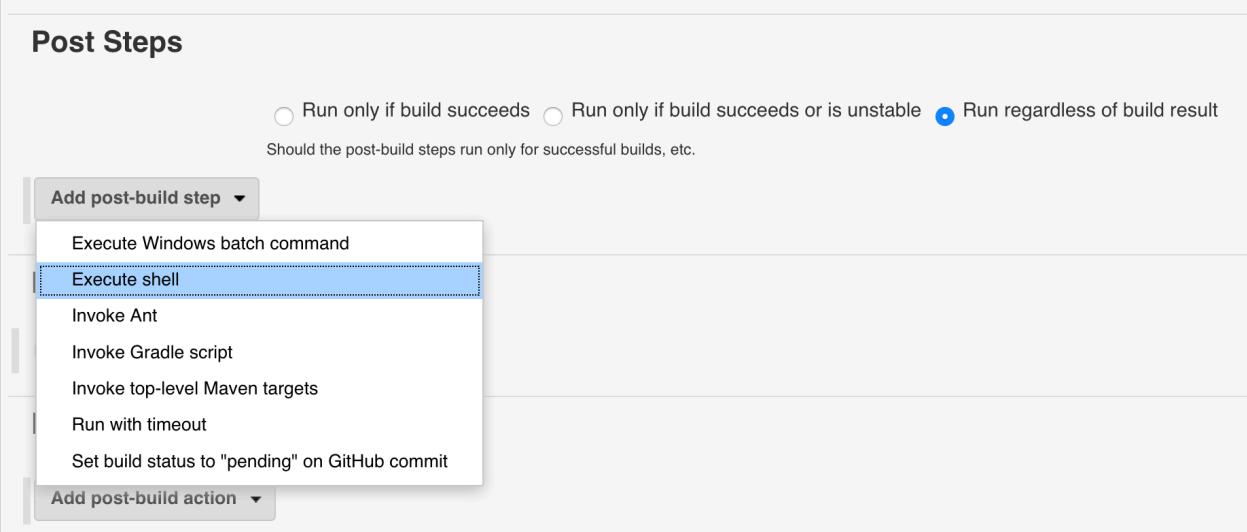
2. Enter the **job name**, select **Maven Project** and click on **OK**



3. Specify the Git Repo URL

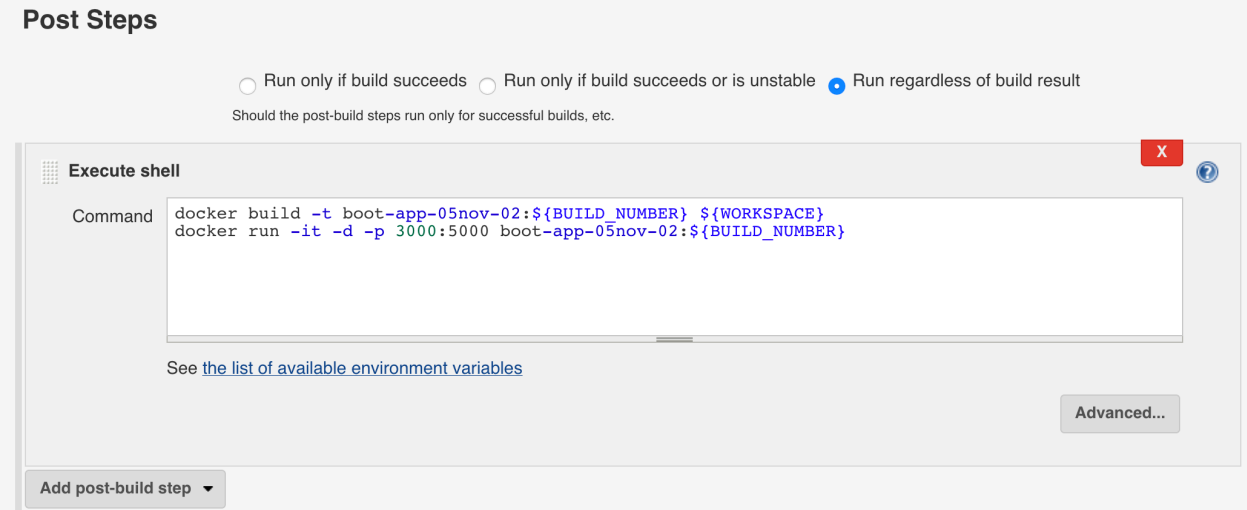


4. Click on **Add post-build step** and select **Execute shell**



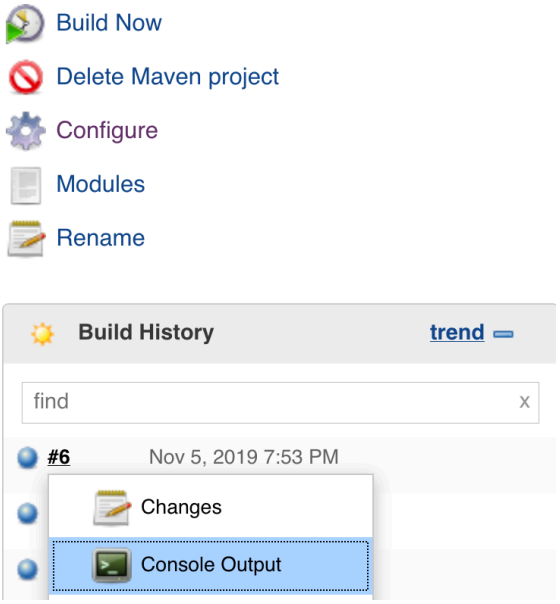
5. Enter the docker build & docker run commands which you tested on command prompt in earlier exercise.

docker build –t boot–app–05nov–02:${BUILD_NUMBER} ${WORKSPACE}
docker run –it –d –p 3000:5000 boot–app–05nov–02:${BUILD_NUMBER}



6. Click on **Save** button

7. Click on **Build Now** link and go to the **Console Output** of the current build in progress



8. You can observe in the console that Jenkins build process picks up docker build command, generates the docker image and also creates a Docker container

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.071 s
[INFO] Finished at: 2019-11-05T19:53:22-05:00
[INFO] ------------------------------------------------------------------------
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /Users/nanda/.jenkins/workspace/boot-002/pom.xml to com.demo.springboot/spring-boot-hello-world-example/0.0.1-SNAPSHOT
/spring-boot-hello-world-example-0.0.1-SNAPSHOT.pom
[JENKINS] Archiving /Users/nanda/.jenkins/workspace/boot-002/target/spring-boot-hello-world-example-0.0.1-SNAPSHOT.jar to
com.demo.springboot/spring-boot-hello-world-example/0.0.1-SNAPSHOT/spring-boot-hello-world-example-0.0.1-SNAPSHOT.jar
[boot-002] $ /bin/sh -xe /var/folders/vv/9x81ghcj5_jd7_4zpyqny5zc0000gn/T/jenkins9150309080332977211.sh
channel stopped
+ docker build -t boot-app-05nov-02:6 /Users/nanda/.jenkins/workspace/boot-002
Sending build context to Docker daemon  16.55MB

Step 1/4 : FROM openjdk:8-jdk-alpine
 ---> a3562aa0b991
Step 2/4 : EXPOSE 8080
 ---> Using cache
 ---> c084233bb3ce
Step 3/4 : ADD target/spring-boot-hello-world-example-0.0.1-SNAPSHOT.jar app.jar
 ---> 124c0dd8b9c2
Step 4/4 : ENTRYPOINT ["java","-jar","app.jar"]
 ---> Running in 5c9a80cf5752
Removing intermediate container 5c9a80cf5752
 ---> 737aa8842580
Successfully built 737aa8842580
Successfully tagged boot-app-05nov-02:6
+ docker run -it -d -p 3000:5000 boot-app-05nov-02:6
f177593f163a1de2feaca72ade8d94e68e10e22c7bdafc19137217f9ef2cedc2
Finished: SUCCESS
```

9. Go to CMD prompt and verify that the new decker image is created

 # decker images

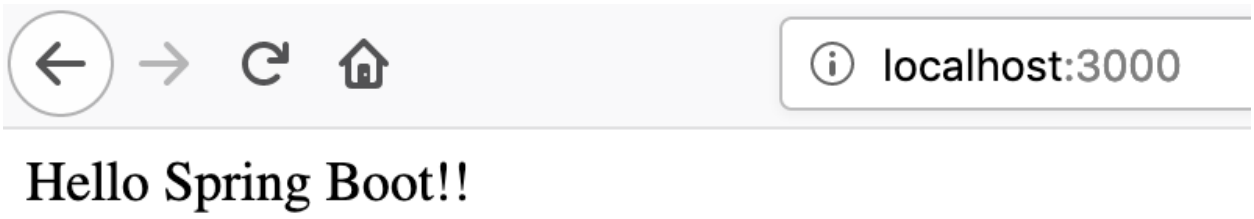
```
Nandakumars-MacBook-Pro:boot-002 nanda$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
boot-app-05nov-02   6                   737aa8842580        15 seconds ago      121MB
boot-app-05nov-01   1.0                 ed031beab71c        5 hours ago         121MB
openjdk             8-jdk-alpine        a3562aa0b991        5 months ago        105MB
```

10. Also verify that the Docker container is also stared for the image as part of this Jenkins Job

 # docker ps

```
Nandakumars-MacBook-Pro:boot-002 nanda$ docker ps
CONTAINER ID        IMAGE                   COMMAND               CREATED             STATUS              PORTS
 NAMES
f177593f163a        boot-app-05nov-02:6     "java -jar app.jar"   25 seconds ago      Up 23 seconds       8080/tcp, 0.0.0.0:3000->5000/tcp
 priceless_tereshkova
fa76d0fb68e5        boot-app-05nov-01:1.0   "java -jar app.jar"   3 hours ago         Up 3 hours          8080/tcp, 0.0.0.0:4000->5000/tcp
 crazy_snyder
```

11. Visit ***http://localhost:3000***



Hello Spring Boot!!