Phase 5

Lesson 5

**Web Hosting**

This section will guide you to:

* Build a custom Docker image with Spring Boot application and deploy it to AWS EKS

Steps:

1. Creating a custom Docker image

2. Deploying a Spring Boot application to AWS EKS

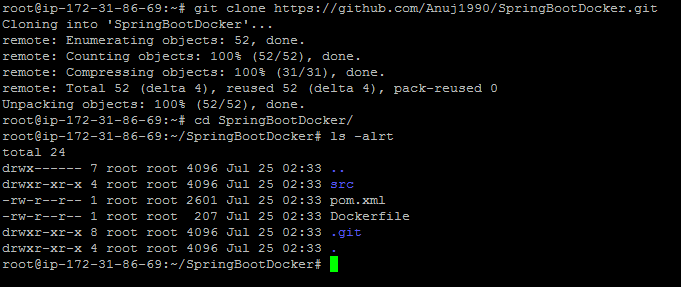
3. Pushin the code to GitHub repositories

**Step 1:** Creating a custom Docker image

* Follow the set of commands shown below to build a custom Docker image:

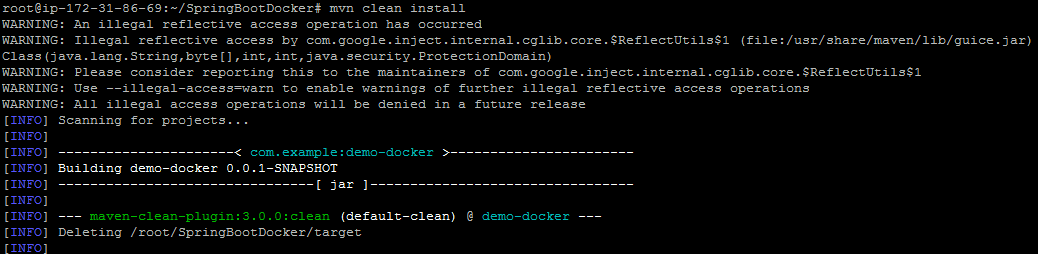
**git clone** [**https://github.com/Anuj1990/SpringBootDocker.git**](https://github.com/Anuj1990/SpringBootDocker.git)

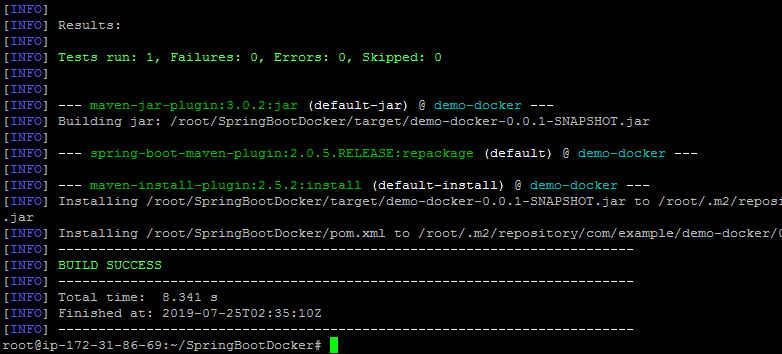
**ls -lart**



* Build source code to generate artifacts which can be deployed on Docker host.

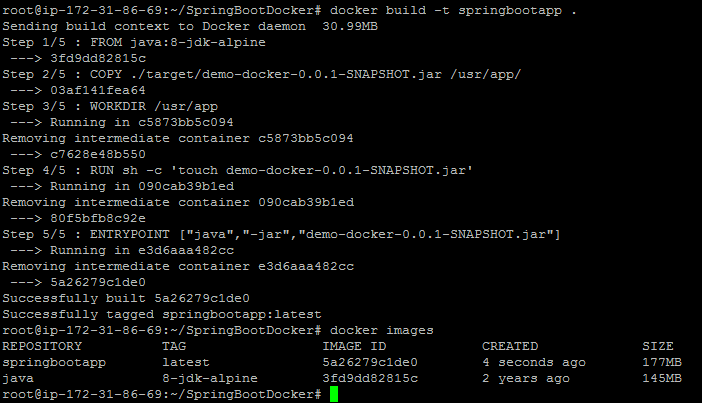
**mvn clean install**





* Deploy this artifact inside the custom Docker image using **docker build** command line. Follow the steps shown below to create the custom Docker image:

**docker build -t springbootapp .**

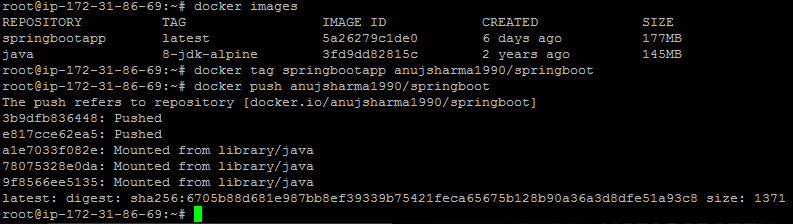


* Push this image to Docker Hub. Follow the command below to do so.

**docker images**

**docker tag springbootapp anujsharma1990/springboot**

**docker push anujsharma1990/springboot**

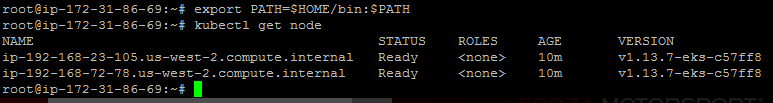


**Step 5.3.2:** Deploying a Spring Boot application to AWS EKS

* Configure **kubectl command line** and deploy containers to AWS EKS.

**export PATH=$HOME/bin:$PATH**

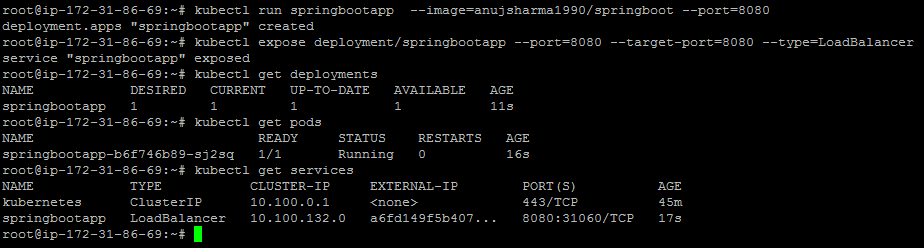
**kubectl get node**



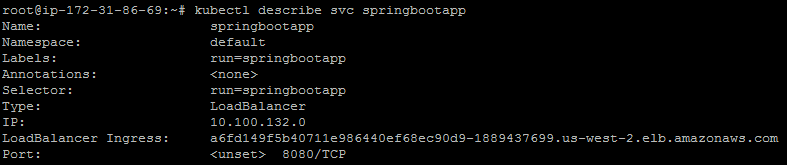
* Create Kubernetes deployment and service using the set of commands given below:

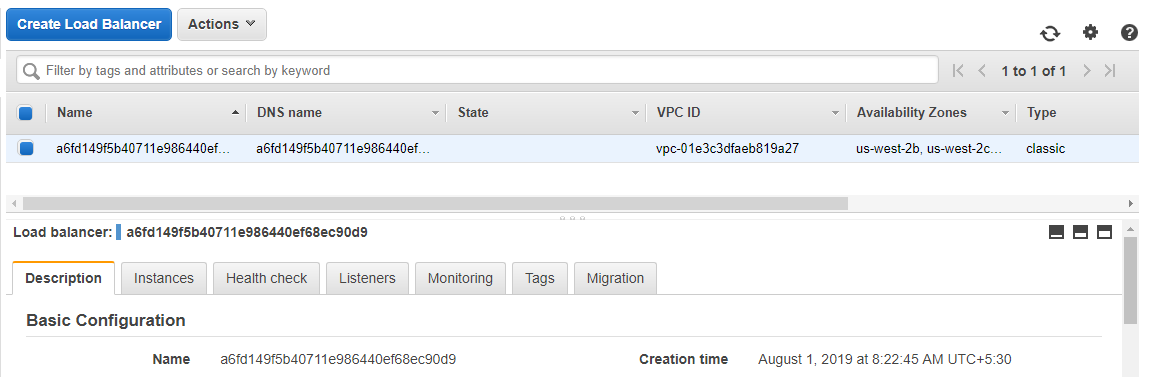
**kubectl run springbootapp--image=anujsharma1990/springboot --port=8080**

**kubectl expose deployment/springbootapp --port=8080 --target-port=8080 --type=LoadBalancer**



**Please Note:** Once the pod is deployed, we can get the Load Balancer URL from springbootapp EKS Service. EKS will automatically configure the Load Balancer in AWS.





* To access the Spring Boot application, use the **Load Balancer URL** as shown below.

**curl -w "\n" a6fd149f5b40711e986440ef68ec90d9-1889437699.us-west-2.elb.amazonaws.com:8080/greet/EKSSpringboot**

