**Spring Boot Docker Kubernetes – Handle Deployment Portability Scalability of Microservice Using Docker**

**Step -1 – Introducing to challenges while building, deploying microservices**

**Step -2 – What are containers and how they are different from VMs**

**Step -3 – What are containers, containerization and docker**

**Step -4 – Introduction to Docker components and its architecture**

**Step -5 – Docker installation and docker hub introduction**

**Step -6 – Introduction to 3 approaches for Docker image generation**

**Step -7 – Generate docker image for Accounts microservice with Dockerfile – Part 1**Add this to every project (accounts,loans,cards) pom.xml

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.1</version>

<configuration>

<source>17</source>

<target>17</target>

<fork>true</fork>

<executable>C:\Program Files\Eclipse Adoptium\jdk-17.0.8.7-hotspot\bin\javac</executable>

</configuration>

</plugin>

\github\eazybytes\_ms\_k8s\accounts> .\mvnw.cmd clean package

This will create a jar file at  
\github\eazybytes\_ms\_k8s\accounts\target\accounts-0.0.1-SNAPSHOT.jar  
  
You can run the accounts application using java command using this jar  
 \github\eazybytes\_ms\_k8s\accounts>java -jar target/accounts-0.0.1-SNAPSHOT.jar

Same thing applied to loans and cards microservices

\github\eazybytes\_ms\_k8s\loans> .\mvnw.cmd clean package

\github\eazybytes\_ms\_k8s\cards> .\mvnw.cmd clean package

**Step -8 – Generate docker image for Accounts microservice with Dockerfile – Part 2**

**accounts\Dockerfile**

FROM openjdk:17-jdk-slim

COPY target/accounts-0.0.1-SNAPSHOT.jar accounts-0.0.1-SNAPSHOT.jar

ENTRYPOINT ["java","-jar","/accounts-0.0.1-SNAPSHOT.jar"]

loans\Dockerfile  
FROM openjdk:17-jdk-slim

COPY target/loans-0.0.1-SNAPSHOT.jar loans-0.0.1-SNAPSHOT.jar

ENTRYPOINT ["java","-jar","/loans-0.0.1-SNAPSHOT.jar"]

cards\Dockerfile  
FROM openjdk:17-jdk-slim

COPY target/cards-0.0.1-SNAPSHOT.jar cards-0.0.1-SNAPSHOT.jar

ENTRYPOINT ["java","-jar","/cards-0.0.1-SNAPSHOT.jar"]

**Step -9 – Generate docker image for Accounts microservice with Dockerfile – Part 3**

PS D:\Nilesh\github\eazybytes\_ms\_k8s\accounts> docker build . -t nileshzarkar/accounts:0.0.1

PS D:\Nilesh\github\eazybytes\_ms\_k8s\loans> docker build . -t nileshzarkar/loans:0.0.1

PS D:\Nilesh\github\eazybytes\_ms\_k8s\cards> docker build . -t nileshzarkar/cards:0.0.1

PS C:\Users> docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

nileshzarkar/cards 0.0.1 **d3e6a109dd76** About a minute ago 462MB

nileshzarkar/accounts 0.0.1 90400d546f90 About a minute ago 462MB

nileshzarkar/loans 0.0.1 dc830add8bcc About a minute ago 462MB

PS C:\Users> docker inspect d3e

"Env": [

"PATH=/usr/local/openjdk-17/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",

"JAVA\_HOME=/usr/local/openjdk-17",

"LANG=C.UTF-8",

"JAVA\_VERSION=17.0.2"

],

**Step -10 – Running accounts microservice as docker container**

PS C:\Users> docker run -p 8080:8080 nileshzarkar/accounts:0.0.1  
PS C:\Users> docker run -p 8090:8090 nileshzarkar/loans:0.0.1  
PS C:\Users> docker run -p 9000:9000 nileshzarkar/cards:0.0.1  
  
Test the basic endpoints

**Step -11 – Challenges with Dockerfile approach to generate a Docker image**

**Step -12 – Generate Docker image of Loans microservice with BuildPacks**Update accounts\pom.xml loans\pom.xml and cards\pom.xml

<plugin>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<image>

<name>nileshzarkar/${project.artifactId}:0.0.1</name>

</image>

</configuration>

</plugin>

PS D:\Nilesh\github\eazybytes\_ms\_k8s\accounts> .\mvnw.cmd spring-boot:build-image

PS D:\Nilesh\github\eazybytes\_ms\_k8s\loans> .\mvnw.cmd spring-boot:build-image

PS D:\Nilesh\github\eazybytes\_ms\_k8s\cards> .\mvnw.cmd spring-boot:build-image  
  
PS C:\Users> docker run -p 8080:8080 nileshzarkar/accounts:0.0.1  
PS C:\Users> docker run -p 8090:8090 nileshzarkar/loans:0.0.1  
PS C:\Users> docker run -p 9000:9000 nileshzarkar/cards:0.0.1  
  
Test the basic endpoints

**Step -13 – Generate Docker image of Cards microservice with Google Jib**<https://github.com/GoogleContainerTools/jib/tree/master/jib-maven-plugin#quickstart>  
**Google Container Registry**<configuration>

<to>

<image>**gcr.io**/my-gcp-project/my-app</image>

</to>

</configuration>  
**Amazon Elastic Container Registry**

<configuration>

<to>

<image>**aws\_account\_id.dkr.ecr.region.amazonaws.com**/my-app</image>

</to>

</configuration>  
**Docker Hub Registry**

<configuration>

<to>

<image>**docker.io**/my-docker-id/my-app</image>

</to>

</configuration>

Update accounts\pom.xml loans\pom.xml and cards\pom.xml

<plugin>

<groupId>com.google.cloud.tools</groupId>

<artifactId>jib-maven-plugin</artifactId>

<version>3.4.0</version>

<configuration>

<to>

<image>nileshzarkar/${project.artifactId}:0.0.1</image>

</to>

</configuration>

</plugin>  
  
\github\eazybytes\_ms\_k8s\accounts> .\mvnw.cmd compile jib:dockerBuild  
\github\eazybytes\_ms\_k8s\loans> .\mvnw.cmd compile jib:dockerBuild  
\github\eazybytes\_ms\_k8s\cards> .\mvnw.cmd compile jib:dockerBuild  
  
PS C:\Users> docker run -p 8080:8080 nileshzarkar/accounts:0.0.1  
PS C:\Users> docker run -p 8090:8090 nileshzarkar/loans:0.0.1  
PS C:\Users> docker run -p 9000:9000 nileshzarkar/cards:0.0.1  
  
Test the basic endpoints

**Step -14 – Compare Dockerfile, Buildpacks and Jib approaches**

**Step -15 – Pushing Docker images from your local to remote Docker hub**

**Using docker command**PS C:\Users > docker login

PS C:\Users > docker image push docker.io/nileshzarkar/accounts:0.0.1  
PS C:\Users > docker image push docker.io/nileshzarkar/loans:0.0.1

PS C:\Users > docker image push docker.io/nileshzarkar/cards:0.0.1  
  
**Maven home: C:\Users\niles\.m2\wrapper\dists\apache-maven-3.9.4-bin\32a55694\apache-maven-3.9.4**

<server>

<id>registry.hub.docker.com</id>

<username>XXXX </username>

<password>XXXXX@91</password>

</server>

**Using JIB plugin – Add this plugin to all services**<plugin>

<groupId>com.google.cloud.tools</groupId>

<artifactId>jib-maven-plugin</artifactId>

<version>3.4.0</version>

<configuration>

<from>

<image>openjdk:17-jdk-slim</image>

</from>

<to>

<image>registry.hub.docker.com/nileshzarkar/${project.artifactId}:${project.version}</image>

</to>

</configuration>

<executions>

<execution>

<id>build-and-push-docker-image</id>

<phase>package</phase>

<goals>

<goal>build</goal>

</goals>

</execution>

</executions>

</plugin>

\github\eazybytes\_ms\_k8s\accounts> .\mvnw.cmd clean package

**Step -16 – Introduction to Docker-Compose**

\github\eazybytes\_ms\_k8s\accounts> docker build . -t nileshzarkar/accounts:0.0.1  
\github\eazybytes\_ms\_k8s\accounts> docker build . -t nileshzarkar/loans:0.0.1  
\github\eazybytes\_ms\_k8s\accounts> docker build . -t nileshzarkar/cards:0.0.1  
  
**docker-compose\docker-compose.yml**  
services:

accounts:

image: "nileshzarkar/accounts:0.0.1"

container\_name: accounts-ms

ports:

- "8080:8080"

deploy:

resources:

limits:

memory: 700m

networks:

- eazybank

loans:

image: "nileshzarkar/loans:0.0.1"

container\_name: loans-ms

ports:

- "8090:8090"

deploy:

resources:

limits:

memory: 700m

networks:

- eazybank

cards:

image: "nileshzarkar/cards:0.0.1"

container\_name: cards-ms

ports:

- "9000:9000"

deploy:

resources:

limits:

memory: 700m

networks:

- eazybank

networks:

eazybank:

driver: "bridge"

**Step -17 – Running all microservices containers using Docker-Compose**

\github\eazybytes\_ms\_k8s\docker-compose> docker-compose -f docker-compose.yml up

Test all the end-points  
  
\github\eazybytes\_ms\_k8s\docker-compose> docker-compose -f docker-compose.yml down

This command also deletes the containers

**Step -18 – Demo of Docker compose commands**

\github\eazybytes\_ms\_k8s\docker-compose> docker-compose -f docker-compose.yml stop

This command just stops the containers do not delete  
\github\eazybytes\_ms\_k8s\docker-compose> docker-compose -f docker-compose.yml start

This command just starts the stopped containers  
\github\eazybytes\_ms\_k8s\docker-compose> docker-compose -f docker-compose.yml up -d  
This command just create and starts the containers in background

**Step -19 – Deep Dive into Docker commands**

**Step -20 – Introduction to Docker extensions vscode**