Command syntax to create an image

docker build -t <image-name>:<tag-name> <location of Dockerfile>

eg : docker build -t cts/user-service:latest .

Note : (.) in case command is being run from the location of Dockerfile

Command syntax to launch a container

docker run -d -p <host-port>:<internal-port> <image-name>:<tag-name>

eg: docker run -d -p 9090:9090 cts/user-service:latest

Note : (-p) to map internal(docker internal network) port to host machine port

(-d) to run container in detached mode

Removing exited containers cache and stopped container

docker container prune

Building an Image (logical steps)

1. Build a Jar - /target/<application build>.jar
2. Setup the Prerequisites for Running the JAR - openjdk:8-jdk-alpine
3. Copy the jar
4. Run the jar

Docker Commands - Creating Image Manually

to launch alpine jdk container

- docker run -dit openjdk:8-jdk-alpine

to run command in running container

- docker container exec <container-name> ls /tmp

to copy jar file into running container

- docker container cp target/<application build>.jar <image-name>:/tmp

testing if it is copied

- docker container exec <container-name> ls /tmp

add an command into running container (env is configured now), create image of that environment

- docker container commit --change='CMD ["java","-jar","/tmp/<application build>.jar"]' <container-name> <new-image-name>:<tag>

launching container on newly created image

- docker run -p <host-port>:<container-port> <new-image-name>:<tag>

Creating image through Docker File

```
# Dockerfile
FROM openjdk:8-jdk-alpine
ADD target/<application build>.jar /tmp/<application build>.jar
CMD ["java","-jar","/tmp/<application build>.jar"]
```

Creating Docker image directly while using maven package (using plugin)

```
# plugin
<plugin>
    <groupId>com.spotify</groupId>
    <artifactId>dockerfile-maven-plugin</artifactId>
    <version>1.4.10</version>
    <executions>
        <execution>
            <id>default</id>
            <goals>
                <goal>build</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <repository>cts/${project.name}</repository>
        <tag>${project.version}</tag>
        <skipDockerInfo>true</skipDockerInfo>
    </configuration>
</plugin>
```

Database properties to set to use mysql container running on docker network

${<env-var-name>:<defualt-value>}

eg:

spring.datasource.url=jdbc:mysql://${RDS_HOSTNAME:localhost}:${RDS_PORT:3306}/${RDS_DB_NAME:user_db}

spring.datasource.username=${RDS_USERNAME:root}

spring.datasource.password=${RDS_PASSWORD:pass}

#spring.datasource.url=jdbc:mysql://localhost:3306/user_db

#spring.datasource.username=root

#spring.datasource.password=pass

running mysql docker container

docker run --detach --env MYSQL_ROOT_PASSWORD=<root-pass> --env MYSQL_DATABASE=<db-name> --name mysql --publish 3306:3306 mysql:5.7

to get into mysql container bash

docker container exec -it <containerid> bash

>mysql u<username> p<password>

launching Spring Boot container Connected with mysql container without network

docker container run -p 8080:8080 --link=mysql -e RDS_HOSTNAME=mysql <image-name>:<tag>

docker network related command

docker network ls

docker network create <network-name>

docker inspect <network-name>

running mysql docker container using Network

docker run --detach --env MYSQL_ROOT_PASSWORD=<root-pass> --env MYSQL_DATABASE=<db-name> --name mysql --publish 3306:3306 --network=<network name> mysql:5.7

launching Spring Boot container Connected with mysql container with network

docker container run -p 8080:8080 --network=<network name> -e RDS_HOSTNAME=mysql cts/<image-name>:<tag>

running mysql docker container using Network and volume

docker run --detach --env MYSQL_ROOT_PASSWORD=<root-pass> --env MYSQL_USER=<user> --env MYSQL_PASSWORD=<pass> --env MYSQL_DATABASE=<db-name> --name mysql --publish 3306:3306 --network=<network name> --volume mysql-database-volume:var/mysql/data mysql:5.7

```
## Steps to build angular docker image
# will create simple html launching content (index.html + plain js files)
ng build --prod

## Dockerfile structure for angular application
FROM nginx:1.17.1-alpine
COPY /dist/<app-name> /usr/share/nginx/html
```

```
## yml file structure for docker compose (microservices)
version: '3.7'
services:

  naming-server:
    image: cts/discovery-server:0.0.1-SNAPSHOT
    #build:
      #context: discovery-server
      #dockerfile: Dockerfile
    ports:
      - "8761:8761"
    restart: always
    networks:
      - pixogram-network
```

```
zuul-api-gateway:
  image: cts/api-gateway:0.0.1-SNAPSHOT
  #build:
    #context: api-gateway
    #dockerfile: Dockerfile

  ports:
    - "8765:8765"
  restart: always
  depends_on:
    - naming-server
    - mysql
  environment:
    RDS_HOSTNAME: mysql
    RDS_PORT: 3306
    RDS_DB_NAME: micro-user-db
    RDS_USERNAME: root
    RDS_PASSWORD: abc
  networks:
    - pixogram-network
```

```
user-service:
    image: cts/user-service:0.0.1-SNAPSHOT
    #build:
      #context: user-service
      #dockerfile: Dockerfile
    ports:
      - "9090:9090"
    restart: always
    depends_on:
      - naming-server
      - mysql
    environment:
      RDS_HOSTNAME: mysql
      RDS_PORT: 3306
      RDS_DB_NAME: micro-user-db
      RDS_USERNAME: root
      RDS_PASSWORD: abc
    networks:
      - pixogram-network
```

```
 mysql:
    image: mysql:5.7
    ports:
      - "3306:3306"
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: abc
      MYSQL_USER: root
      MYSQL_PASSWORD: root
      MYSQL_DATABASE: micro-user-db
    volumes:
      - mysql-database-data-volume:var/mysql/data
    networks:
      - pixogram-network

pixogram-client:
    image: cts/pixogram-client:latest
    ports:
      - "3000:80"
    restart: always
    networks:
      - pixogram-network

# Networks to be created to facilitate communication between containers
networks:
  pixogram-network:
```

NOTE :

in all microservices and zuul gateway service (leaving alone discovery) add this property in application.properties file to register with eureka server

eureka.client.service-url.default-zone=http://naming-server:8761/eureka/

naming-server is the name of discovery-server service in compose yml file