**Docker Commands**

1. Dockerfile

FROM openjdk:8

ADD target/demo-0.0.1-SNAPSHOT.jar demo-0.0.1-SNAPSHOT.jar

EXPOSE 8082

ENTRYPOINT ["java","-jar", "demo-0.0.1-SNAPSHOT.jar"]

1. docker -v: Docker Version
2. docker build -f Dockerfile -t my-docker-app .
3. docker images: show docker images in local container
4. docker run -p 8082:8082 my-docker-app

**Docker MYSQL + REST Commands**

*(Make sure to put mysql container name in place of localhost in app.props)*

*(host.docker.internal)*

1. docker pull mysql
2. docker images
3. docker run --name mysql-test -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=test -e MYSQL\_USER=sa -e MYSQL\_PASSWORD=root -d mysql
4. docker container logs mysql-test
5. docker container ls
6. docker build -t user-mysql .
7. docker run -d -p 8089:8089 --name user-mysql --link mysql-test:mysql user-mysql
8. docker container logs user-mysql
9. docker container stop user-mysql
10. docker container rm user-mysql

**For calling a dockerized rest api externally**

--net=host makes the docker’s network same as the host network

docker run -p 8082:8082 my-docker-app --net=host

**For calling two dockerized rest apis**

Proper way is to use concept of docker networks.

Shortcut is to use the ip address of the machine on which dockers are running.

Or use docker.host.internal in place of localhost.

**Using networks**

1. Create a docker network

docker network create rest-network

1. Run the first-app

docker run -p 8082:8082 --name first-app --network rest-network first-app

1. Run the second-app

docker run -p 8081:8081 --name second-app --network rest-network second-app