**Creating image for MySQL database for Auth2 Server**

Create the MySQL database image.

docker run --name mysql-auth2 -e MYSQL\_ROOT\_PASSWORD=root -p 3306:3306 -v C:/Users/bprajapati/Desktop/2019/TOOLs/docker-mount-dir/mysql:/var/lib/mysql -d mysql:5.7

IF you want to save image along with data don’t use volume option,

docker run --name mysql-auth2 -e MYSQL\_ROOT\_PASSWORD=root -p 3306:3306 -d mysql:5.7

First commit the changes you made for the schema you created with initial data

docker commit container\_id mysql-auth2

Tag the image with your docker hub prefix

docker tag mysql-auth2 docker-io-acc-id/mysql-auth2

Login to docker.io

docker login -u username

Push the image

docker push docker-io-acc-id/mysql-auth2

Remove the above container and images

docker rm mysql-auth2

docker rmi mysql-auth2

docker rmi docker-io-acc-id/mysql-auth2

Pull from your docker hub account and run to see it has database setup or not

docker run --name mysql-auth2 -e MYSQL\_ROOT\_PASSWORD=root -p 3306:3306 -v C:/Users/bprajapati/Desktop/2019/TOOLs/docker-mount-dir/mysql:/var/lib/mysql -d itbrijesh143/mysql-auth2

MySQL commands: <http://g2pc1.bu.edu/~qzpeng/manual/MySQL%20Commands.htm>

* docker exec -it container\_id /bin/bash
* mysql -uroot -proot
* mysql use oauth2\_dev

**IMPORTAN:** Database in unix system is case sensitive if you see no connection refuse error but table not found then it is the problem with case of your tables. 😊

* Setup Centos operating system

docker run -d centos tail -f /dev/null

Get into centos host system using command,

docker run -it container\_id bash

Install java in this centos host system

yum install java

* Create a Dockerfile with below content

FROM centos

RUN yum install -y java

VOLUME /tmp

ADD /target/eureka-naming-server-0.0.1-SNAPSHOT.jar eureka-server.jar

RUN sh -c 'touch /eureka-server.jar'

ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/eureka-server.jar"]

* Generate the image for above docker file.

Docker build -t docker-eureka-server .

Note: we used (.) because name of docker file is ‘Dockerfile’. If you have other file then you have to pass name of file as well.

* Now generate the container for the above build image and map to port for external access

docker run -d -p 8761:8761 docker-eureka-server

**Automatic build using io.fabric8 maven plugin|**

* Add below plugin in pom.xml

<!-- Added for Dockerfile Docker fabric.io -->

<plugin>

<groupId>io.fabric8</groupId>

<artifactId>docker-maven-plugin</artifactId>

<!-- <version>0.20.0</version> -->

<configuration>

<dockerHost>http://127.0.0.1:2375</dockerHost>

<!-- <dockerHost>unix:///var/run/docker.sock</dockerHost> -->

<verbose>true</verbose>

<images>

<image>

<name>${docker.image.prefix}/${docker.image.name}</name>

<build>

<dockerFileDir>${project.basedir}/src/main/docker/</dockerFileDir>

<!--copies artficact to docker build dir in target-->

<assembly>

<descriptorRef>artifact</descriptorRef>

</assembly>

<tags>

<tag>latest</tag>

<tag>${project.version}</tag>

</tags>

</build>

</image>

</images>

</configuration>

</plugin>

* Configure below properties used in plugin

<!-- Added for Dockerfile Docker fabric.io -->

<!--set this to your docker acct name-->

<docker.image.prefix>itbrijesh143</docker.image.prefix>

<!--Set to name of project-->

<docker.image.name>spring-eureka-server</docker.image.name>

* Create new folder in src/main/docker and create Dockerfile using below content

FROM openjdk

VOLUME /tmp

ADD maven/eureka-naming-server-0.0.1-SNAPSHOT.jar eureka-server.jar

RUN sh -c 'touch /eureka-server.jar'

ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/eureka-server.jar"]

* Do maven clean build

mvn clean package

* Do docker build

mvn clean package docker:build

* Push the generate image to docker.io using below command. But before that you need to setup your docker.io username and password.

Option 1: Set in environment

docker.username, docker.password

Option 2: Provide through command

mvn -Ddocker.username=myuserid -Ddocker.password=mypass docker:push

Option 3: Configure in ../m2/settings.xml

<servers>

<server>

<id>docker.io</id>

<username>jolokia</username>

<password>s!cr!t</password>

</server>

....

</servers>

mvn clean package docker:build docker:push

* In above Dockerfile the name of the jar file provided is hardcoded, we can make it dynamic using below steps,

Add below plugin in pom.xml

<!-- Added for Dockerfile Docker fabric.io -->

<plugin>

<groupId>org.codehaus.gmavenplus</groupId>

<artifactId>gmavenplus-plugin</artifactId>

<version>1.5</version>

<executions>

<execution>

<phase>prepare-package</phase>

<goals>

<goal>execute</goal>

</goals>

</execution>

</executions>

<configuration>

<scripts>

<script>file:///${project.basedir}/src/main/scripts/BuildDockerfile.groovy</script>

</scripts>

</configuration>

<dependencies>

<dependency>

<groupId>org.codehaus.groovy</groupId>

<artifactId>groovy-all</artifactId>

<!-- any version of Groovy \>= 1.5.0 should work here -->

<version>2.4.8</version>

<scope>runtime</scope>

</dependency>

</dependencies>

</plugin>

Generate file with name BuildDockerfile.groovy inside the folder src/main/scripts

String template = new File("${project.basedir}/src/main/docker/DockerfileTemplate".toString()).getText()

def dockerFileText = new groovy.text.SimpleTemplateEngine().createTemplate(template)

.make([fileName: project.build.finalName])

println "writing dir " + "${project.basedir}/target/dockerfile"

new File("${project.basedir}/target/dockerfile/".toString()).mkdirs()

println "writing file"

File dockerFile = new File("${project.basedir}/target/dockerfile/Dockerfile".toString())

dockerFile.withWriter('UTF-8') { writer ->

writer.write(dockerFileText)

}

Create a new file with name DockerfileTemplate under src/main/docker with below content

FROM openjdk

VOLUME /tmp

ADD maven/${fileName}.jar ${fileName}.jar

RUN sh -c 'touch /${fileName}.jar'

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/${fileName}.jar"]

* Running container from maven.

Add below section in pom.xml inside <image> and after <build> tag.

<run>

<ports>

<port>8761:8761</port>

</ports>

</run>

Use command to run from maven, mvn docker:run or docker:start to run in background.

* dgg

**Generating docker images for Angular 7 app**

* Generate the Dockerfile with below content
* sdfsd