Docker file: root of project location

```
From tomcat:9.0.24-jdk8-openjdk-slim
RUN rm -rf /usr/local/tomcat/webapps/*
COPY ./target/<your>.war /usr/local/tomcat/webapps/<dest>.war
CMD ["catalina.sh","run"]
```

Create Network

```
docker network create <network name>
docker network ls
```

Create a db schema file : schema-mysql.sql

launch mysql container

```
docker container run
--name <custom-container-name>
--network <network-name>
-e MYSQL_ROOT_PASSWORD=<password>
-e MYSQL_DATABASE=<dbname>
-d
mysql:8
```

Check if mysql with db running

```
docker container exec -it <container-id> bash
mysql -uroot -p<password>
```

application.properties

```
spring.datasource.url=jdbc:mysql://<custom-container-name of mysql>/<dbname>
spring.datasource.username=root
spring.datasource.password=<password>
spring.datasource.platform=mysql
spring.datasource.initialization-mode=always
```

Create image

```
docker image build -t <image name> .
```

Launch Container

```
docker container run
--network <network name>
--name <custom-container-name >
-p 8080:8080
-d <image name>
```

Jenkins Build

```
┌───────────── minute (0 - 59)
│ ┌───────────── hour (0 - 23)
│ │ ┌───────────── day of month (1 - 31)
│ │ │ ┌───────────── month (1 - 12)
│ │ │ │ ┌───────────── day of week (0 - 6) (Sunday to Saturday;
│ │ │ │ │                   7 is also Sunday on some systems)
│ │ │ │ │
│ │ │ │ │
* * * * *
```

Build every hour:

H * * * *

Build every 20 minutes:

H/20 * * * *

Build every 20 minutes 2am to 11pm:

H/20 14-23 * * *

Build every 20 minutes, work time/days (8am-6pm, MON-FRI) only:

H/20 8-18 * * 1-5

Build every hour MON-WED and FRI only:

H * * * 1-3,5

Build every hour, in April and December:

H * * 4,12 *

Build at 8.30am on July 4:

30 8 4 7 *

```xml
<plugin>
           <groupId>org.apache.maven.plugins</groupId>
           <artifactId>maven-surefire-plugin</artifactId>
            <version>2.19.1</version>
           <configuration>
           <testFailureIgnore>true</testFailureIgnore>
           </configuration>
       </plugin>
```

```xml
<dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>4.0.1</version>
        <scope>provided</scope>
    </dependency>

<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
<dependency>
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>javax.servlet.jsp-api</artifactId>
        <version>2.3.3</version>
        <scope>provided</scope>
    </dependency>
```

Integrating Docker:

Manage Jenkins ->Manage Plugin->Available : Docker Cloud Provider

Configuring Docker:

#Manage Jenkins->Configure System->Add New Cloud (Docker)

#Host URI

#Docker Agent Template:

Label : docker-agent

☐ Docker image : benhall/dind-jenkins-agent:v2

Container Settings:Volume : /var/run/docker.sock:/var/run/docker.sock

#Connect Method : Connect with SSH

#Enable container and agent

Configuring jenkins build job for Docker

Configuring Project :

 # Restrict where this project can be run : docker-agent

 # Add Build Steps : Execute Shell

```
docker build -t <image-name>:${BUILD_NUMBER} .
```

#pulling docker image

```
==>docker pull <image-name>
```

# launching container base on images

```
==> docker container run <image-name>
```

launch nginx server

```
==> docker container run -p 8282:2000 nginx
```

Docker file Command

- FROM : other docker images
- LABEL :
- RUN :
- COPY
- WORKDIR
- CMD
- ENTRYPOINT

Creating a docker images

```
docker build -t <image-name> .
```

Docker file for Jar packaged boot application

```
FROM java:8-jdk-alpine


COPY ./target/<src.jar> /usr/app/


WORKDIR /usr/app


ENTRYPOINT ["java","-jar","<src>.jar"]
```

Angular build for production (before creating angular image)

```
ng build --prod
```

nginx.config file

```
worker_processes  1;


events {
    worker_connections  1024;
}


http {
    server {
        listen 80;
        server_name  localhost;


        root   /usr/share/nginx/html;
        index  index.html index.htm;
        include /etc/nginx/mime.types;


        gzip on;
        gzip_min_length 1000;
        gzip_proxied expired no-cache no-store private auth;
        gzip_types text/plain text/css application/json application/javascript application/x-
javascript text/xml application/xml application/xml+rss text/javascript;


        location / {
            try_files $uri $uri/ /index.html;
        }
    }
}
```

Efficient approach of Angular Docker image

```
FROM nginx:alpine


COPY dist/<app-name>/nginx.conf /etc/nginx/nginx.conf


WORKDIR /usr/share/nginx/html
COPY dist/<app-name>  .
```

Alternate approach for Angular Docker images

```
FROM node:12.2.0

# set working directory
WORKDIR /app

# install and cache app dependencies
COPY package.json /app/package.json
RUN npm install
RUN npm install -g @angular/cli

# add app
COPY . /app

# start app
CMD ["ng", "serve"]
```