**Docker:**

[**https://github.com/in28minutes/docker-crash-course/**](https://github.com/in28minutes/docker-crash-course/)

>docker run **-p 5000:5000** in28min/todo-rest-api-h2:1.0.0.RELEASE

>docker container run **-p 5000:5000** in28min/todo-rest-api-h2:1.0.0.RELEASE (above one is same )

>docker run **-p 5001:5000** in28min/todo-rest-api-h2:1.0.0.RELEASE (Another container running)

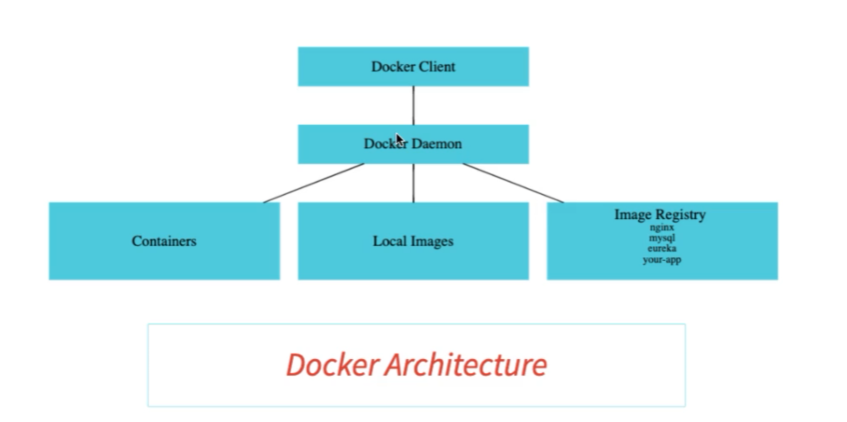
>docker run -p 5000:5000 **-d** in28min/todo-rest-api-h2:1.0.0.RELEASE

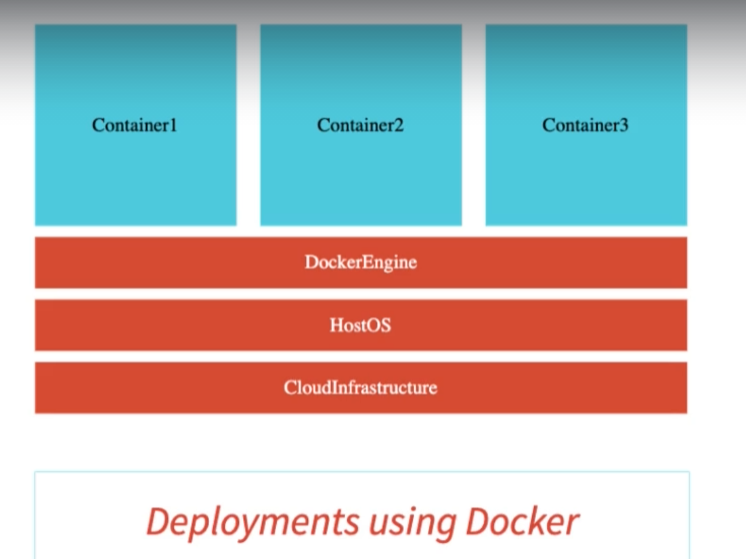
>docker logs -f <id of the container >

>docker container ls

>docker images

>docker container stop <id of the container>





>docker tag in28min/todo-rest-api-h2:1.0.0.RELEASE in28min/todo-rest-api-h2:1.0.0.latest (new image)

>docker pull mysql (mysql image)

>docker search mysql

>docker image history

>docker container stop <id of the container>

>docker container kill <id of the container >

>docker container pause <id of the container> (Stope serving)

>docker container unpause <id of the container> (resuming)

>docker container inspect <id of the container>

>docker container prune (remove all the stopped container)

>docker events

>docker top <id of the container>

>docket stats

>docker run -m 512mb –cpu-quota 5000

>docker system df

**Build docker manually:**

* Building an Image
* Build a Jar - /target/hello-world-rest-api.jar
* Setup the Prerequisites for Running the JAR - openjdk:8-jdk-alpine
* Copy the jar
* Run the jar

**Command for doing this manual docker.**

* docker run -dit openjdk:8-jdk-alpine
* docker container exec naughty\_knuth ls /tmp (Naught\_knuth is container name )
* docker container cp target/hello-world-rest-api.jar naughty\_knuth:/tmp
* docker container exec naughty\_knuth ls /tmp
* docker container commit naughty\_knuth in28min/hello-world-rest-api:manual1
* docker run in28min/hello-world-rest-api:manual1
* docker container ls
* docker container commit --change='CMD ["java","-jar","/tmp/hello-world-rest-api.jar"]' naughty\_knuth in28min/hello-world-rest-api:manual2
* docker run -p 8080:8080 in28min/hello-world-rest-api:manual2

**Build docker Using docker File:**

FROM openjdk:8-jdk-alpine

EXPOSE 8080

ADD target/hello-world-rest-api.jar hello-world-rest-api.jar

ENTRYPOINT ["sh", "-c", "java -jar /hello-world-rest-api.jar"]

>docker build -t in28min/hello-world-rest-api:dockerFile

**Spotify plugin**

[**https://github.com/in28minutes/docker-crash-course/tree/master/01-hello-world-rest-api**](https://github.com/in28minutes/docker-crash-course/tree/master/01-hello-world-rest-api)

**.dockerFile**

**Spring boot Application: WAR**

@Configuration

public class SecurityConfiguration extends WebSecurityConfigurerAdapter{

//Create User - in28Minutes/dummy

@Autowired

public void configureGlobalSecurity(AuthenticationManagerBuilder auth)

throws Exception {

auth.inMemoryAuthentication().withUser("in28minutes").password("{noop}dummy")

.roles("USER", "ADMIN");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests().antMatchers("/login", "/h2-console/\*\*").permitAll()

.antMatchers("/", "/\*todo\*/\*\*").access("hasRole('USER')").and()

.formLogin();

http.csrf().disable();

http.headers().frameOptions().disable();

}

}

**Dockerfile**

FROM tomcat:8.0.51-jre8-alpine

EXPOSE 8080

RUN rm -rf /usr/local/tomcat/webapps/\*

COPY target/\*.war /usr/local/tomcat/webapps/ROOT.war

CMD ["catalina.sh","run"]

**Pom.xml**

**Mvn clean pacakage**

<build>

<finalName>todo-web-application-h2</finalName>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

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

</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>in28min/${project.name}</repository>

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

<skipDockerInfo>true</skipDockerInfo>

</configuration>

</plugin>

</plugins>

</build>

> docker container run **-p 5000:5000** in28min/todo-web-application-h2:0.0.1.SNAPHSHOT

> docker push in28min/todo-web-application-h2:0.0.1.SNAPHSHOT

**Spring boot Application: mysql (two layer )**

**Running my sql docker**

>docker run --detach

--env MYSQL\_ROOT\_PASSWORD=dummypassword

--env MYSQL\_USER=todos-user

--env MYSQL\_PASSWORD=dummytodos

--env MYSQL\_DATABASE=todos

--name mysql

--publish 3306:3306 mysql:5.7

>mysqlsh

Web application to run the link to mysql and environment variable to property file. (link is deprecated one )

> docker container run -p 8080:8080 **--link=mysql** -e **RDS\_HOSTNAME=mysql** in28min/todo-web-application-mysql:0.0.1-SNAPSHOT

**Running With Custom network (local host wont work Container(mysql to webapp)**

>docker network ls (bridge,host,none)

>docker network create web-application-mysql-network

> docker run --detach --env MYSQL\_ROOT\_PASSWORD=dummypassword --env MYSQL\_USER=todos-user --env MYSQL\_PASSWORD=dummytodos --env MYSQL\_DATABASE=todos --name mysql --publish 3306:3306 **--network=web-application-mysql-network** mysql:5.7

>docker container run -p 8080:8080

**--network=web-application-mysql-network**

-e RDS\_HOSTNAME=mysql in28min/todo-web-application-mysql:0.0.1-SNAPSHOT

Network commands

>docker inspect web-application-mysql-network

**Volume (persisting On Rerun)**docker run --detach --env MYSQL\_ROOT\_PASSWORD=dummypassword --env MYSQL\_USER=todos-user --env MYSQL\_PASSWORD=dummytodos --env MYSQL\_DATABASE=todos --name mysql --publish 3306:3306 --network=web-application-mysql-network --volume mysql-database-volume:/var/lib/mysql mysql:5.7

**Full stack Web application**

**Building front end**

* Npm install
* Npm run build
* Nginx

## Stage 1 - Lets build the "deployable package"

FROM node:7.10 as frontend-build

WORKDIR /fullstack/frontend

# Step 1 - Download all package dependencies first.

# We will redownload dependencies only when packages change.

COPY package.json package-lock.json ./

RUN npm install

# Step 2 - Copy all source and run build

COPY . ./

RUN npm run build

## Stage 2 - Let's build a minimal image with the "deployable package"

FROM nginx:1.12-alpine

COPY --from=frontend-build /fullstack/frontend/build /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

>docker build -t in28minutes/todo-frontend:snaphsont

>docker run -p 4000:80 in28minutes/todo-frontend:snaphsont

**Builind restfull Services**

### Step 1 - Copy pom.xml and download project dependencies

# Dividing copy into two steps to ensure that we download dependencies

# only when pom.xml changes

COPY pom.xml .

# dependency:go-offline - Goal that resolves all project dependencies,

# including plugins and reports and their dependencies. -B -> Batch mode

RUN mvn dependency:go-offline -B

### Step 2 - Copy source and build "deployable package"

COPY src src

RUN mvn install -DskipTests

# Unzip

RUN mkdir -p target/dependency && (cd target/dependency; jar -xf ../\*.jar)

##### Stage 2 - Let's build a minimal image with the "deployable package"

FROM openjdk:8-jdk-alpine

VOLUME /tmp

ARG DEPENDENCY=/fullstack/backend/target/dependency

COPY --from=backend-build ${DEPENDENCY}/BOOT-INF/lib /app/lib

COPY --from=backend-build ${DEPENDENCY}/META-INF /app/META-INF

COPY --from=backend-build ${DEPENDENCY}/BOOT-INF/classes /app

ENTRYPOINT ["java","-cp","app:app/lib/\*","com.in28minutes.rest.webservices.restfulwebservices.RestfulWebServicesApplication"]

>docker build .

**Docker compose**

>docker-compose version

>docker-compose up

>docker-compose up -e

>docker-compose down

>docker-compose events

>docker-compose images

>docker-compose ps

>docker-compose top

>docker-compose pause

>docker-compose unpause

>docker-compose build

services:

todo-frontend:

image: in28min/todo-front-end:0.0.1-SNAPSHOT

#build:

#context: frontend/todo-app

#dockerfile: Dockerfile

ports:

- "4200:80"

restart: always

depends\_on: # Start the depends\_on first

- todo-api

networks:

- fullstack-application-network

todo-api:

image: in28min/rest-api-full-stack:0.0.1-SNAPSHOT

ports:

- "8080:8080"

restart: always

networks:

- fullstack-application-network

# Networks to be created to facilitate communication between containers

networks:

fullstack-application-network:

<><><><><>

services:

todo-web-application:

image: in28min/todo-web-application-mysql:0.0.1-SNAPSHOT

#build:

#context: .

#dockerfile: Dockerfile

ports:

- "8080:8080"

restart: always

depends\_on: # Start the depends\_on first

- mysql

environment:

RDS\_HOSTNAME: mysql

RDS\_PORT: 3306

RDS\_DB\_NAME: todos

RDS\_USERNAME: todos-user

RDS\_PASSWORD: dummytodos

networks:

- todo-web-application-network

mysql:

image: mysql:5.7

ports:

- "3306:3306"

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: root

MYSQL\_ROOT\_PASSWORD: dummypassword

MYSQL\_USER: todos-user

MYSQL\_PASSWORD: dummytodos

MYSQL\_DATABASE: todos

volumes:

- mysql-database-data-volume:/var/lib/mysql

networks:

- todo-web-application-network

# Volumes

volumes:

mysql-database-data-volume:

networks:

todo-web-application-network:

version: '3.7'

services:

rabbitmq:

image: rabbitmq:3.5.3-management

ports:

- "5672:5672"

- "15672:15672"

restart: always

networks:

- currency-compose-network

naming-server:

#image: in28min/netflix-eureka-naming-server:0.0.1-SNAPSHOT

build:

context: netflix-eureka-naming-server

dockerfile: Dockerfile

ports:

- "8761:8761"

restart: always

networks:

- currency-compose-network

zipkin-server:

image: openzipkin/zipkin

container\_name: zipkin

environment:

STORAGE\_TYPE: mem

RABBIT\_URI: amqp://guest:guest@rabbitmq:5672

ports:

- "9411:9411"

restart: always

depends\_on:

- rabbitmq

networks:

- currency-compose-network

zuul-api-gateway:

#image: in28min/netflix-zuul-api-gateway-server:0.0.1-SNAPSHOT

build:

context: netflix-zuul-api-gateway-server

dockerfile: Dockerfile

environment:

RABBIT\_URI: amqp://guest:guest@rabbitmq:5672

ports:

- "8765:8765"

restart: always

depends\_on:

- naming-server

- rabbitmq

- zipkin-server

networks:

- currency-compose-network

currency-exchange-service:

#image: in28min/currency-exchange-service:0.0.1-SNAPSHOT

build:

context: currency-exchange-service

dockerfile: Dockerfile

environment:

RABBIT\_URI: amqp://guest:guest@rabbitmq:5672

ports:

- "8000:8000"

restart: always

depends\_on:

- naming-server

- rabbitmq

- zipkin-server

networks:

- currency-compose-network

currency-conversion-service:

#image: in28min/currency-conversion-service:0.0.1-SNAPSHOT

build:

context: currency-conversion-service

dockerfile: Dockerfile

ports:

- "8100:8100"

restart: always

environment:

#CURRENCY\_EXCHANGE\_URI: http://currency-exchange-service:8000

RABBIT\_URI: amqp://guest:guest@rabbitmq:5672

depends\_on:

- currency-exchange-service

- naming-server

- rabbitmq

- zipkin-server

networks:

- currency-compose-network

# Networks to be created to facilitate communication between containers

networks:

currency-compose-network: