# Docker

Could play with : <https://labs.play-with-docker.com/> for 80% which will be available for 4 hours

Course Source: <https://github.com/up1/course-introduction-docker>

Slide: <https://github.com/up1/course-introduction-docker/blob/master/slide/SCK-INTRODUCTION-DOCKER-V3.pdf>

FB: Somkiat Puisungnoen

<https://store.docker.com/> vs hub.docker.com

Design pattern:

microservice.io

bilki martin fowler: how to break a Monolith into Microservices

microservice testing

Domain-Deriven Design: Tackling Complexity in the hear of Software by Eric Evans

Building Evolutionary Architectures: by Neal Ford; Rebecca Parsons; Patrik Kua

kitematic >> is the GUI to create container

Logging:

cAdvisor is the agent to sum log from every container then use grafana docker to show dashboard out

Tracing :

<http://zipkin.io/> (open tracing)>> tool to do tracing use with UDP and it’wont effect production performance.

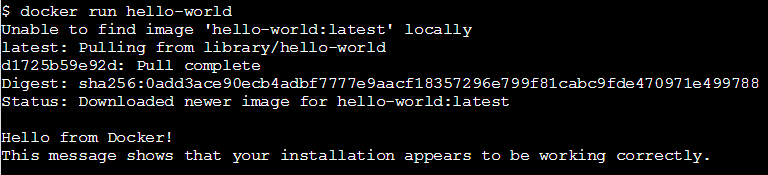
Jaeger >> the same but bigger storage

Istio >> the same but for K8S

1. Start with : DockerToolbox

Docker deamon = docker engine = docker server

Image(Class- immutable) -> container(Instance)



Docker run > this is legacy command to create container from Image ‘hello-world’; will auto pull

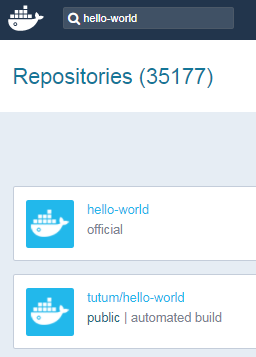
Docker container run hello-world > this is community command

1.1 ‘hello-world:[tag or version]’ -> ‘hello-world:[latest]’ -> please always identify version

1.2 Pull -> retrieve image from registry will first pull from hub.docker.com (public registry) ex: Nexus, Jfrog (private regitry)



1. Image in public register



First one is official but later is other version that other have updated

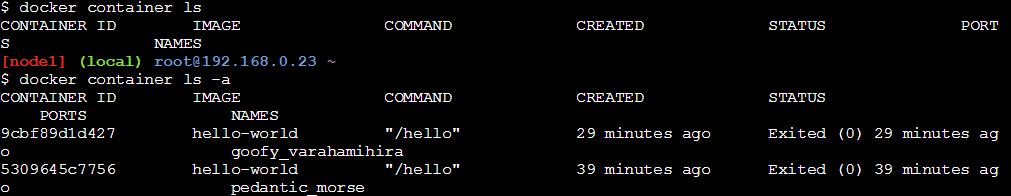
Docker daemon also provide rest api to invoke any method

1. Docker images = docker image ls >> check which image is in our machine

Docker image history hello-world -- > when create image please create it in low level



Docker container ls >> will see container if we run twice then we will see it two



1. Delete

Please stop container before delete it

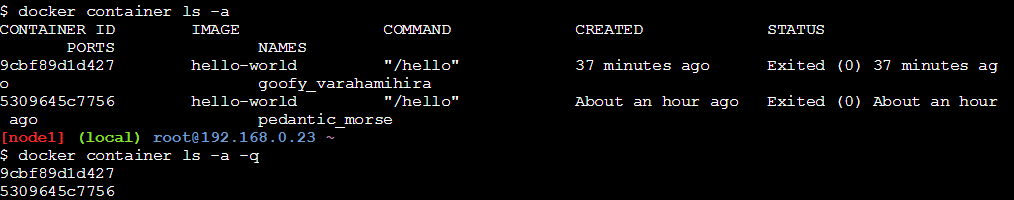
docker container rm [container name]

docker container ls -a -q >> snippet to list only image name that already stop

docker rm $(docker container ls -a -q) >> old command to remove docker from the server

docker rm -f $(docker container ls -a -q) >> old command to ‘force’ remove docker from the server

docker container prune >> new command that remove all container that is not running



Container is grateful shutdown but do our app have manage this

1. Container that still have process

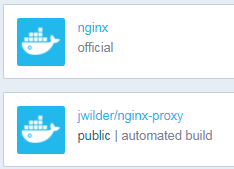
Hello-world is the image that start and dead

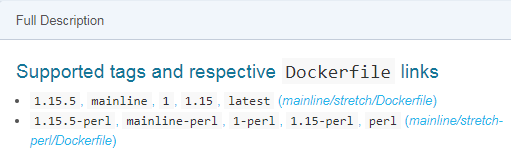
We will try nginx >

Official is to make sure that

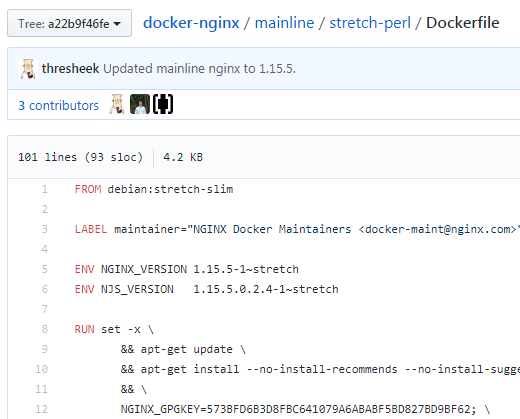
Then we select version that will be used

Alpine is OS that occur for container with 4 mb

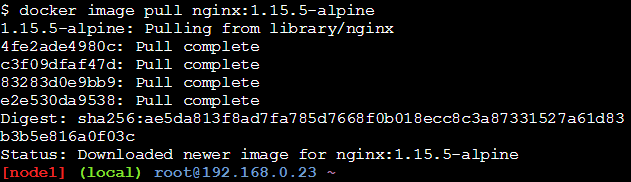


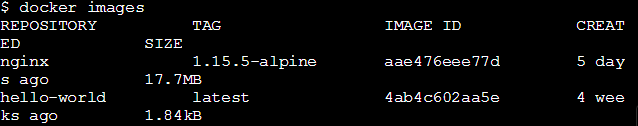


Inside file is command that will be execute



docker image pull nginx:1.15.5-alpine





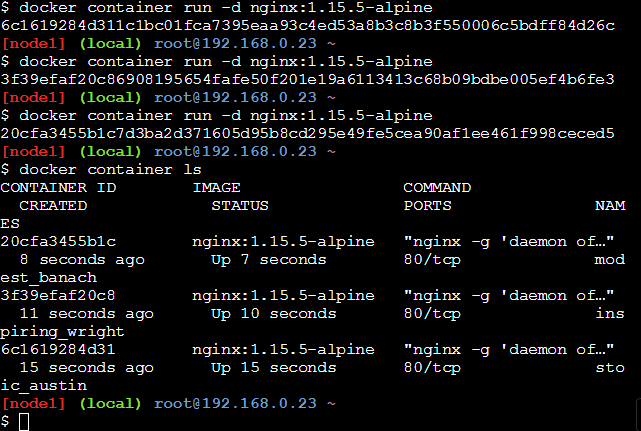
1. Run container

docker foreground >

docker background > daemon mode, detached mode > return id of the container back

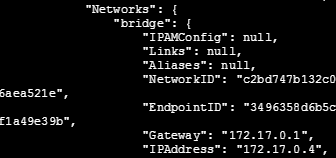
docker container run -d nginx:1.15.5-alpine

docker container ls >> will return information



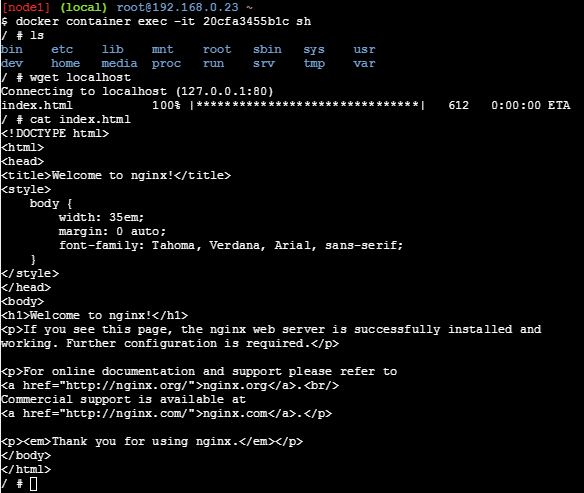
docker container inspect [container\_id]

docker container inspect 20cfa3455b1c



docker container exec -it 20cfa3455b1c sh >> we will get access to that container -i > mean interactive

ps -ef >> you willsee the process



Wget show that This docker is work but can’t access from external cause we still haven’t open to the external world

“Docker world : create delete create delete if config incorrect then delete and create new one”

1. Stop

docker container stop $(docker container ps -q)

docker container prune

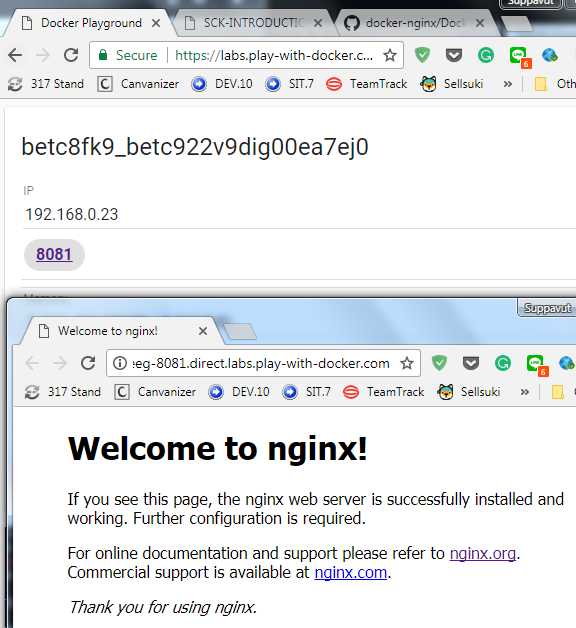
\*\*\* docker size must the same must not scale

1. Forward port (publish port)

docker container run -d -p 8081:80 nginx:1.15.5-alpine >> you will see available port

we need to check the document or inspect the image to see that which parameter that will effect these image

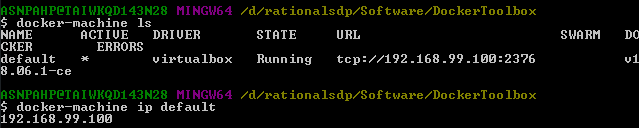
For play ground:



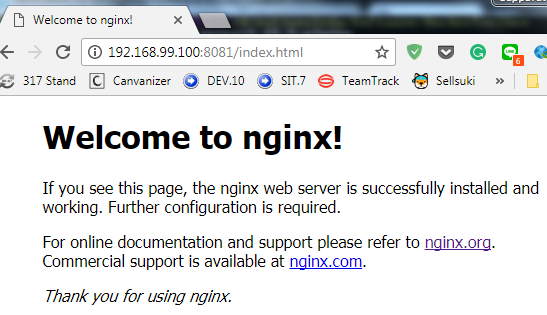
1. IP address or docker host

You will see it on start up or use command to see it





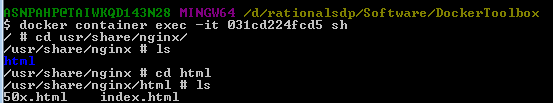
For Docerk, If you install nginx correctly you will see the index.html file

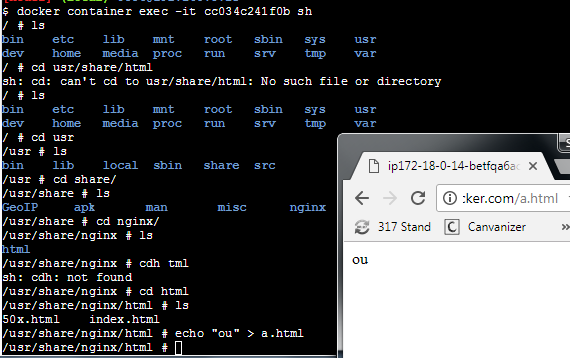


1. Deploy code to container

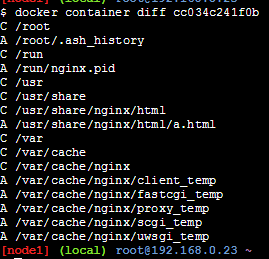
10.1 get in to the container and manual deploy > then create image for next deploy

10.1.1 you will see the index.html here

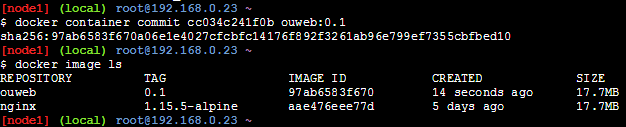




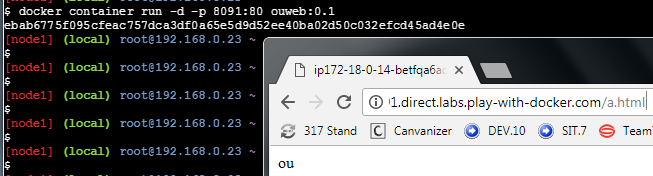
Docker container diff [image] >> you will see what different from base



10.1.2 docker container commit [container id] ouweb:0.1 >> create my image version 0.1 that ready to use!!!



10.1.3 you could create new container from that image by mount to other port



10.1.4 push image to public registry (docker hub) we are not official

> docker login > (oujai/123456)

there are pattern to push it up

> docker image push ouweb:0.1 >> this will be error >> we are not official then can’t do this

our image need prefix in the front this is convention

This is 3 convention

ouweb:0.1 >> need to be official

oujai/ouweb:0.1>> community from docker hub

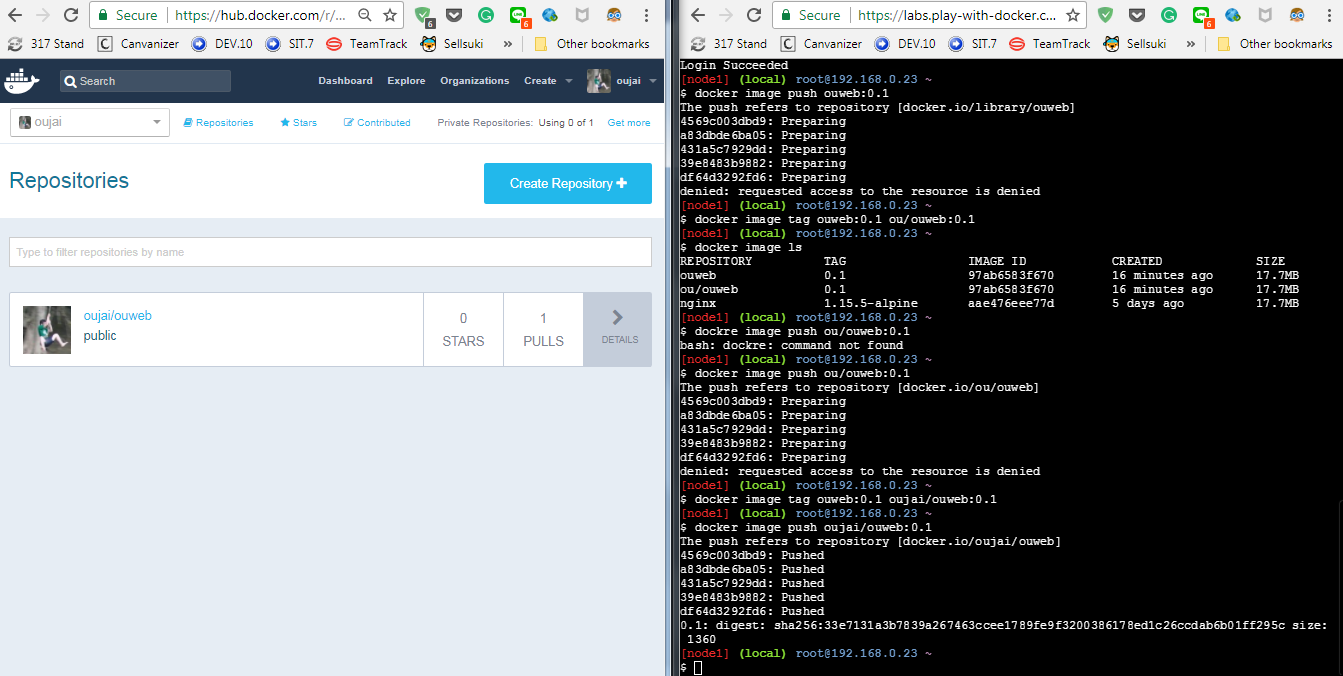
<https://xxx.com/oujai/ouweb:0.1> or <https://xxx.com/ouweb:0.1> >> if you use private registry

so we have 2 way to do it

1. rm
2. docker image tag [source] [target]

After you change the name you will be able to push it up to registry

This tag will only have the link layer. The old file will still there



10.2 make a hole by Mount path (mount volumn) by mapping between path in container and machine

Need to worry about permission of each image. Docker should install with specific user not root one.

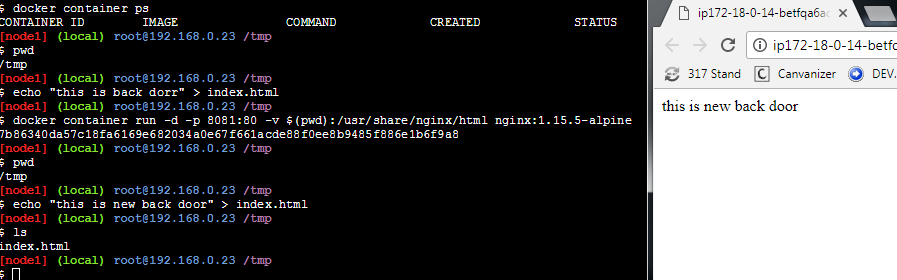
[use this style for **devlopment**, log path, upload path] so that our container won’t be getting bigger

This style is most easy and used most

We create file and mount path to tmp folder so you could see file in $(pwd) to the nginx folder so every time you change the file then you will see the change on browser

Otherwise you will see the problem about the authorize so check this command below

> chown -Rf nginx:nginx /root/test >> user and path is from specification of nginx image



docker container run -d -p 8081:80 -v $(pwd):/usr/share/nginx/html nginx:1.15.5-alpine

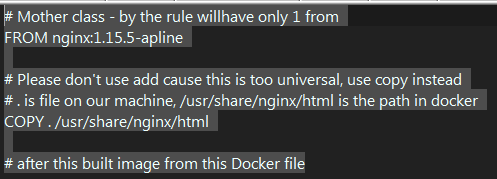
10.3 create container which already include our code

Dockerfile that don’t have suffix >> this use to create image: for example > create from NGINX + our index.html file

Docker way!!!

Add script every line

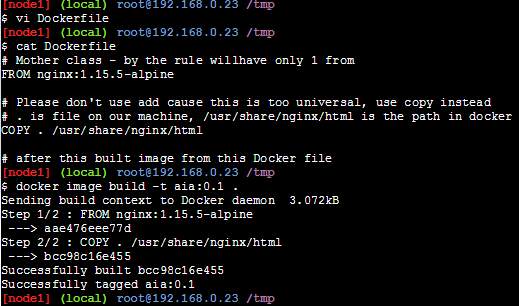
Professional will make it fast



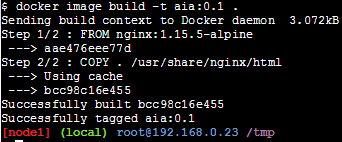
After create Dockerfile and build you will get image

Docker image build -t aia:0.1 .

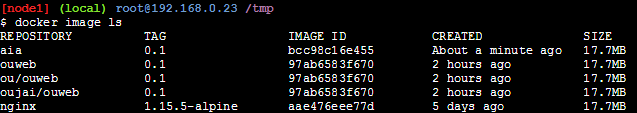
. on the bottom is a must that will indicate all the file that will up to daemon and pack as image



Recreate will get cached

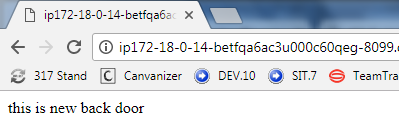


Here is the image file



So you will be able to create container from image:

docker container run -d -p 8099:80 -v $(pwd):/usr/share/nginx/html aia:0.1



1. Security

Will control at network not in container itself

1. – rm

docker container run –rm nginx >> this will auto remove after down. Normally use for adhoc which run only 1 times

docker ps -a

docker container run -d --rm hello-world >> -d is don’t have to print result out, --rm will remove the container

docker container run -d –restart always >> will auto restart if there have a problem

mean once you get in the docker and kill 1 then the container will auto restart itself

1. Dockerfile is the file with out suffix told you about how to build image. there are a tons command in the file need only to open other docker file and learn from them

<https://docs.docker.com/engine/reference/builder/>

<https://docs.docker.com/develop/develop-images/dockerfile_best-practices/>

sample >> don’t try at home cause file ubuntu is quite huge

FROM ubuntu

RUN apt-get update

RUN apt-get install -y wget

CMD and ENTRYPOINT >> use to create long process [page118]

CMD >> for example nignx use CMD[“nginx”, -g, “daemon off;”] >> only have 1 times in docker file

ENTRYPOINT >> use with CMD to dynamic the argument that pass in CMD (there are default define)

EXPOSE [port] >> port that will be used in container

If in docker file there are EXPOSE command then we could use -P so it will automatic bind port to

Example that container that we have multiple port

: Database, search engine

Perfomance test the network will be drop by 20 percent << we need to monitoring

Again please don’t you “ADD” command cause it not secure

Sample ADD <http://a.coom/web> /opt << this will be kind of backdoor

docker container run -d -P –name somkiat nginx:1.15.5-alpine

We identify name for this container

Example : we will expose 9999 and we will -P to automatic assign port

FROM nginx:1.15.5-alpine

WORKDIR /usr/share/nginx/html

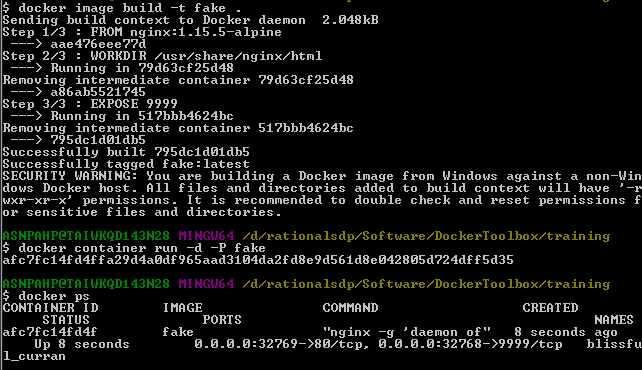
EXPOSE 9999

Run

$docker image build -t fake .

$docker container run -d -P fake

$docker container run -d -p 9001:80 -P fake



1. VOLUME : create data volume mount point

Becareful for this command

docker volume ls << to check volume in container

We have docker secret to manage password in container which we will use encode-decode

docker secret

Secret storage > commercial tool to do it

1. Best practice for write docker file

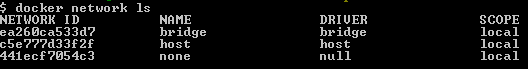
Don’t create line by line command >> combine it to one command line

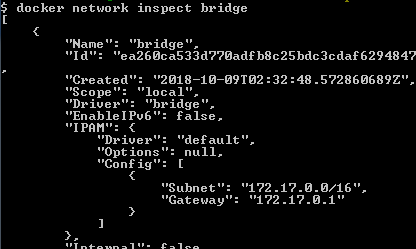
1. Volumn

Default will be keep at >> /var/lib/docker/volumes/….

1. Docker network ls : host is the most fastest in your machine

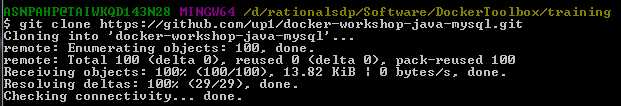
Over head >> from 10,000 concurrent /sec to 7,000 concurrent/sec





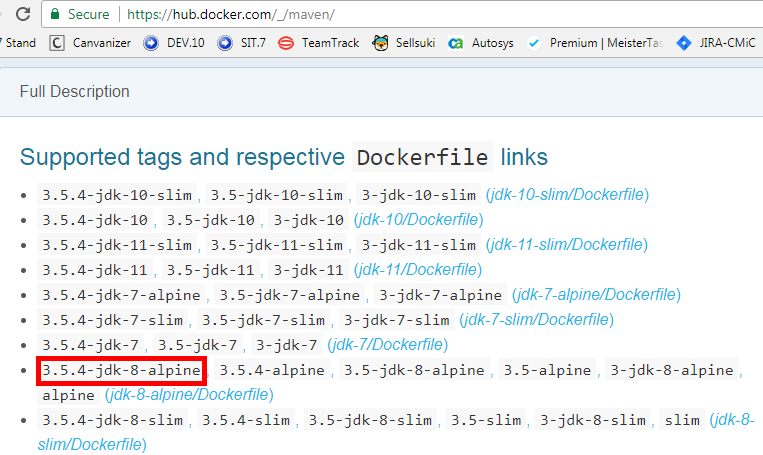
1. Git clone work shop from

> git clone <https://github.com/up1/docker-workshop-java-mysql.git> >> this is our web source code



> we need to pack file so we choose maven and find maven(now gradle is more updated) from hub.docker.com

We select “[3.5.4-jdk-8-alpine](https://github.com/carlossg/docker-maven/blob/f581ea002e5d067deb6213c00a4d217297cad469/jdk-8-alpine/Dockerfile)”



You will see that this container will map our $(pwd) with target package and then set the working directory then call command line “mvn clean package” example

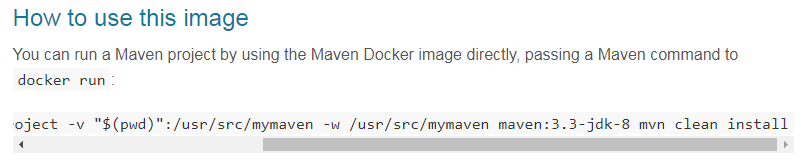
docker container run --rm -v $(pwd):/src -w /src maven:3.5.4-jdk-8-alpine mvn clean package

--rm will remove container after finished

-v $(pwd):/src will mount your working directory to /src

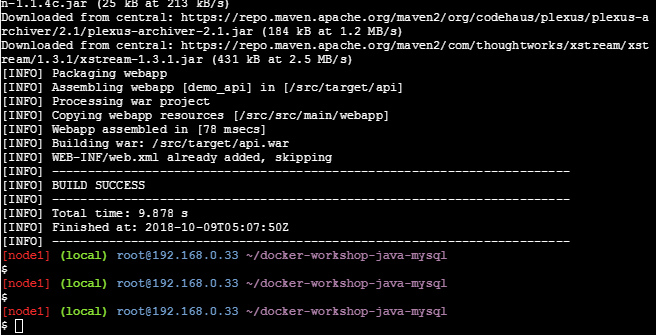
-w /src will use src as wokring directory

mvn clean package is the command that will execute



>> behind the scene, docker will download apline OS, Java, Maven

>> then it will call maven to buildfile



File is here hooley!!!!



Then we will try to create this image for later use

FROM maven:3.5.4-jdk-8-alpine

WORKDIR /src

COPY . /src

RUN mvn clean package

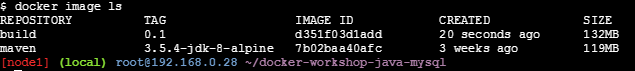
## We could repleace above RUN command with below two command which could be override

# ENTRYPOINT ["mvn"]

# CMD ["clean", "package"]

We could create image from command below

* docker image build -t build:0.1 -f Dockerfile\_build .

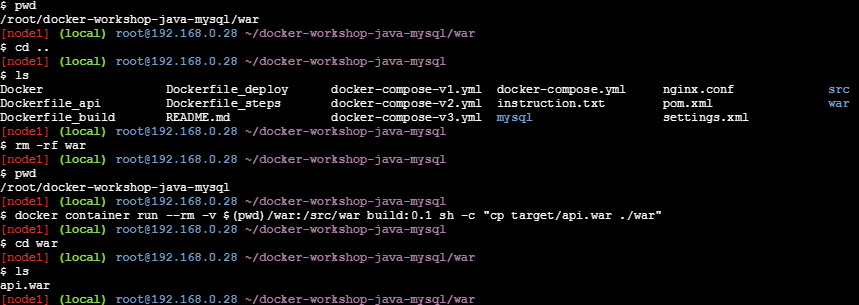


#docker container run build:0.1 sh -c "mvn clean package"

docker container run –rm -v $(pwd)/war:/src/war

docker container run –rm -v $(pwd)/war:/src/war build:0.1 sh -c “cp target/api.war ./war”

this will map our location “root/..sql” to /src/war/ then in the final we will use secure shell and execute command to copy war file from the target/api.war to host location which is ./war



1. We will deploy this war file to tomcat by using war file from previous exercise. we will have two method 1) command line 2) use docker file to create from scratech

19.1 command with -v:

docker container run -d -p 8080:8080 -v $(pwd)/war/api.war:/usr/local/tomcat/webapps/api.war tomcat:9.0.10-jre8-alpine

19.2 docker file: Dockerfile\_deploy

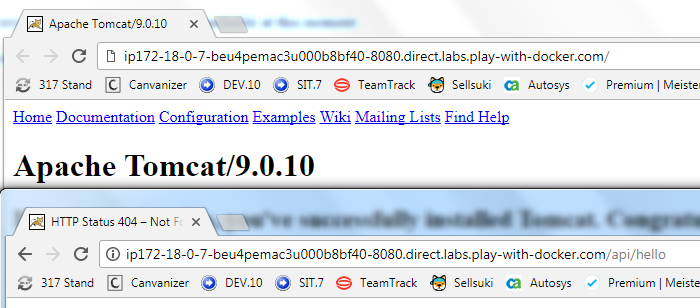
FROM tomcat:9.0.10-jre8-alpine

COPY ./war/api.war /usr/local/tomcat/webapps/api.war

# docker image build -t deploy:0.1 -f Dockerfile\_deploy .

# docker container run -p 8080:8080 deploy:0.1

# http://localhost:8080/api/hello



1. Multi step build as Dockerfile\_step

As [builder] and then in the second docker we could use –from=[builder]

For example

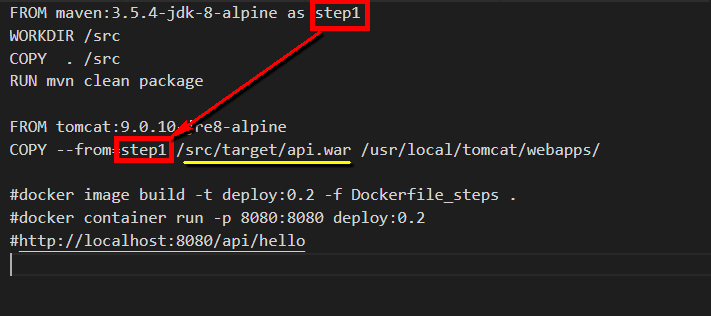
1) we will maven clean package and keep everything in step1

2) pass step1 to another step which is tomcat to use the same stage and pass api.war to the webapp path

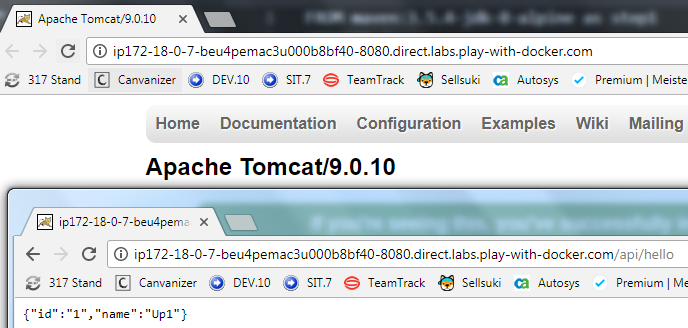
3) the 1) will be remove

4) boom web app will be start on your tomcat

If you want to get out from the tomcat need to add -d as a daemon

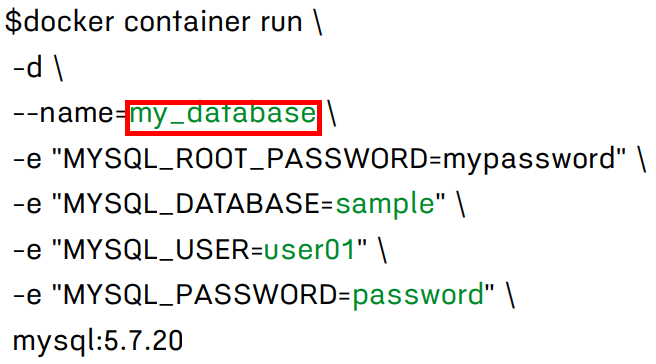


Boom here tomcat is comeup and you web is already up too

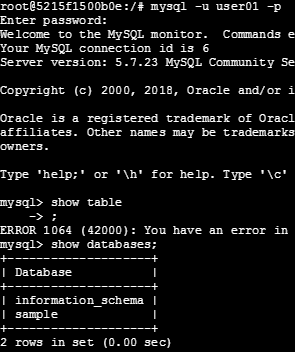


1. Database: Stateful image so we must map the volumn to outside. Default will have map with the container

Mysql have a lot of configuration file by using -e



Container name is very important cause jdbc in other container will use this name to create connection



1. Create Database with existing data

For mysql image if we open document we will found that if we put statement in the “docker-entrypoint-initdb.d” folder then it will auto import to database

FROM mysql:5.7.20 COPY import.sql /docker-entrypoint-initdb.d/

If in our code jdbc have miss match for the database host name then we could use –link to link them as below

$docker container run -p 8080:8080 --link db: my\_database docker-workshop-java-mysql\_api

db : our database image name

my\_database : database host that use in our code

docker-workshop-java-mysql\_api : our tomcat image

1. Docker compose come from fig.sh which already deprecate

We could define every command line in yml. One space also count

Docker-compose.yml -> could be convert to K8S (Kompose)

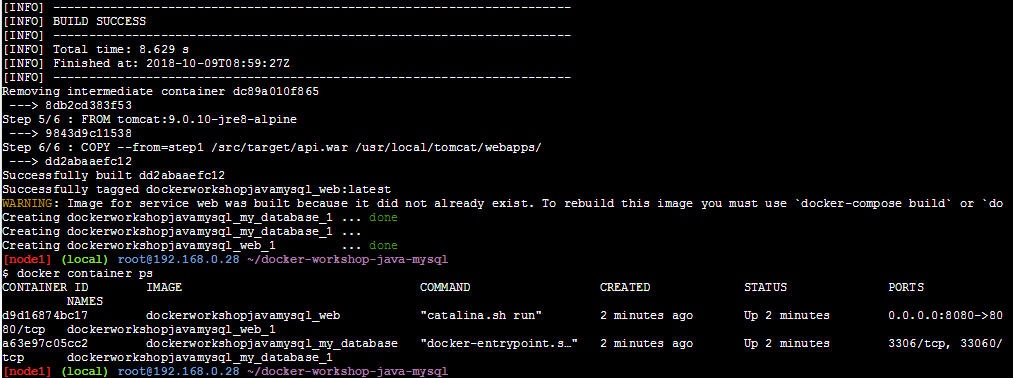
New docker could be deployed on K8S now!!!

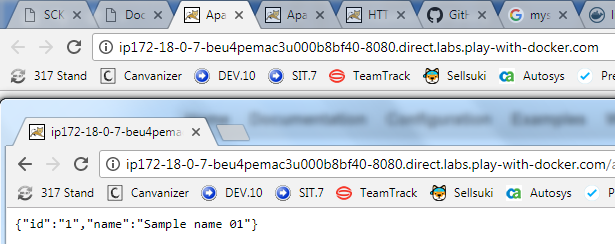
docker-compose up -d

docker-compose down

docker-compose top top >> to check service

docker-compose scale 5 >> must not expose port other wise it will have error





1. Owasp top 10

Check the tool that most used