1. why use docker

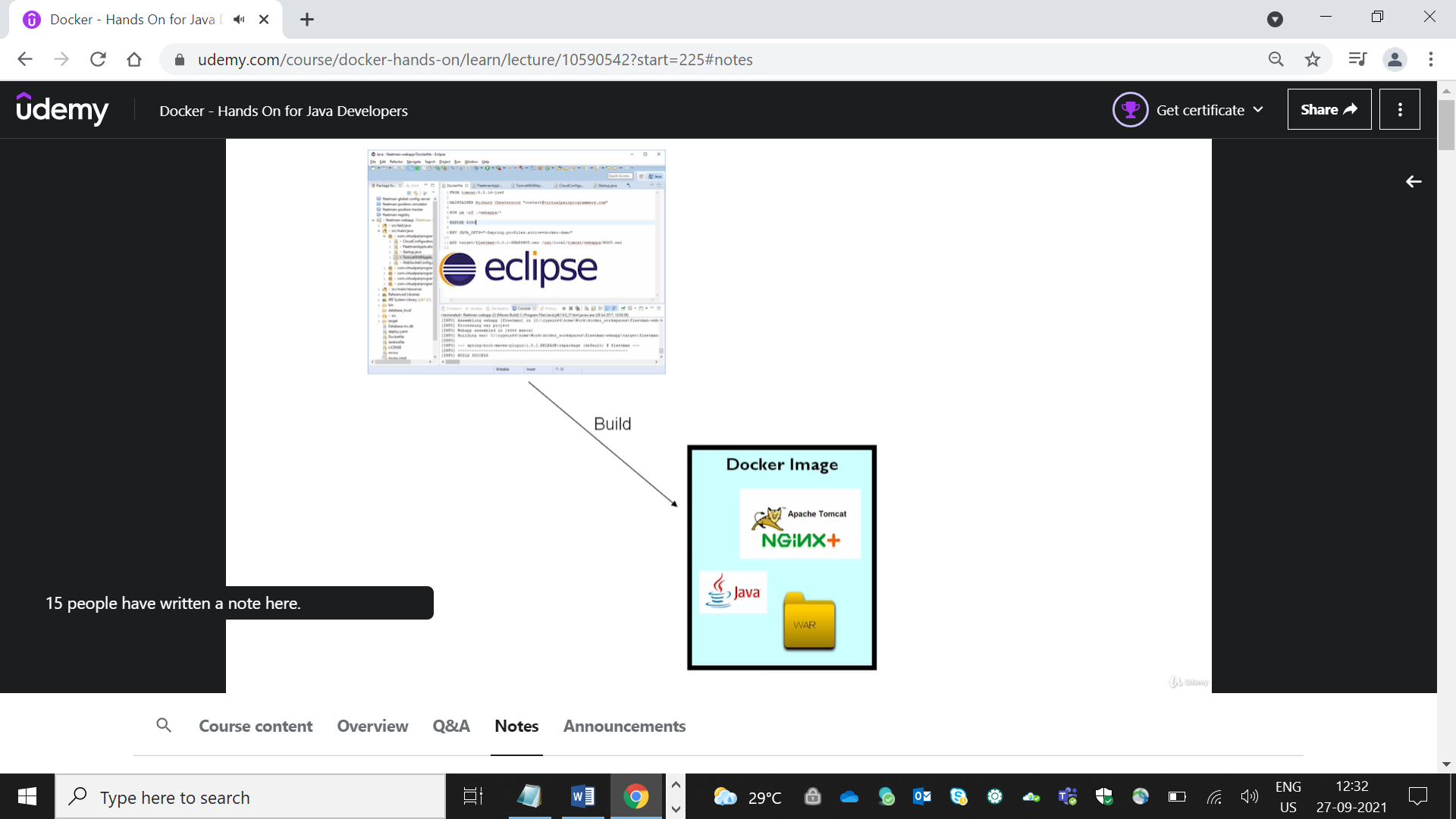
suppose as developer created war file it is working in development evioument but

it is not working in testing enviourment because they have not set the path.

But using docker it will not happend because we give all the steps in docker file

2.Using docker we can deliver final deliverable deployment. Just we have to the run docker image which contains the definition of complete enviourment which include the enviourment variable, Jar/War file and supported software.

When run you run the image then it will become a container.

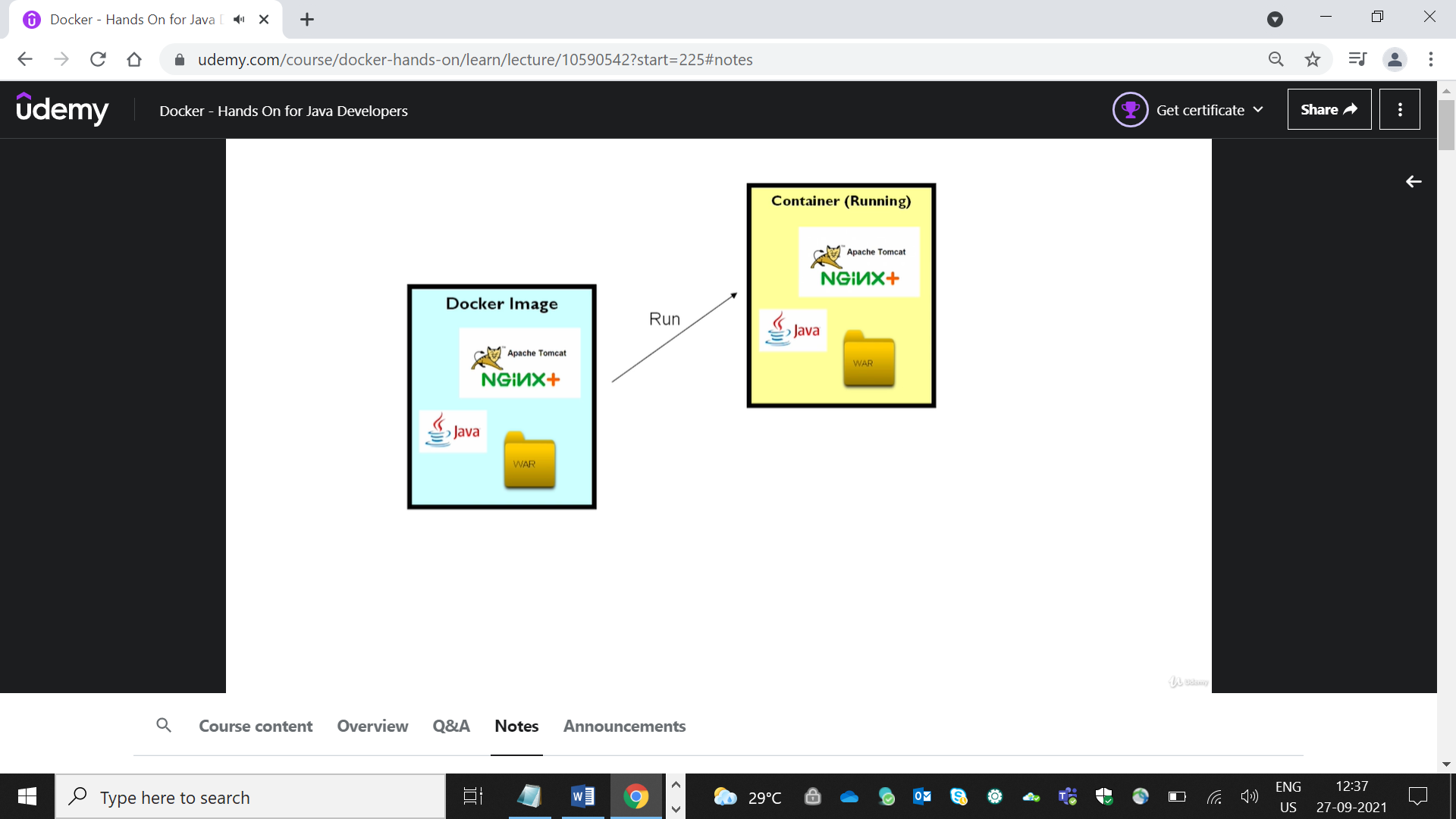


3. what is container

1. Container is an instance of Image

2. Create an instance of image by simply run of Docker image

3. Image is an entity which we build and when we run image then it will be container at runtime



4.what is container and virtual machine

1.Container does not contain a full operating system.

2. it much lighter and efficient than virtual machine

3.Kernel is handling low level processes such as memory manager and etc.

4. container is just a process running over the Kernel

5.container share the same linux kernel. and virtual machine have its own kernel.

5 .Managing the Container

1. List the images

     1.docker image ls

     2. docker ps

2. start container

docker container start containerid

3.docker container stop containerid

1.Run  docker command

   docker container run -p 80:8080 virtualpairprogrammers/fleetman-webapp

2. Log check

   docker container logs  idgenratedafterrundockerimage

3. Run shell command inside the container

  docker container exec -it ee bash

Dockerfile

-Commands

   docker container run -it ubuntu

   apt-get update

   apt-cache search jdk

-Install Jdk

   apt-get install -y openjdk-8-jdk

   javac

**5. Docker Hub**

Commit changes to docker hub

docker container commit -a "Richard chesterwood contact@virtualprogrammers.com" 5f myjdkimage

Push the Images in Dockerhub

To push docker image to docker hub

1. do login

    docker login

    enter username

    enter password

2.apply tag to image

docker image tag image\_id   dockerid/imagename

push docker image

3. run image of mysql

docker container run -e MYSQL\_ROOT\_PASSWORD=password -e MYSQL\_DATABASE=fleetman -d mysql:5

  check the log of container

  docker container logs -f dockerid

**Networking**

**When we have different container and we have to perform the communication between the network then we use the network**

create network command

docker network create my-network

add container to network

docker container run --network my-network --name database  -e MYSQL\_ROOT\_PASSWORD=password -e MYSQL\_DATABASE=fleetman

-d mysql:5

8. **Docker compose**

**1.** we create file which contains the all configuration which we used in command line

2. compose file contains all the configuration required of all the containers

9. Docker swarn

Dockers swarn used when there multiple container and we need to increase the scalability of container.

Docker Swarm we can scale the services horizontally by deploying the Docker containers across multiple hot machines.

These host machine are called as nodes of a swarm.

start docker swarm

docker swarm init

Create network to interact multiple docker instance

Cmd

Docker network create –driver overlay producer-consumer (producer-consumer is name of network)

Create service in docker swarm

Docker service create --network producer-consumer –name producer –p 8080:8080 javainsue/employee-producer

We join the other node by

Docker swarm join command

Difference Docker-compose and Docker-stack

Docker-compose used we need to deploy application in single host

Whereas Docker-stack we will use when we need to deploy services n different node.

Docker-compose and Docker-stack will be similar but only one difference is that pull the images from Docker hub in Docker-stack.

Below command use to create stack

Vim docker-compose.yml

Deploy the Docker stack

Docker stack deploy –c docker-compose.yaml testStack

Kubernate

Container ochestration

It is process of forming a clustor of container so that container will have all this non functional requirements(fault-tolerance,on-demand scalability,perforance,auto discovery,public acess,auto update and rollback)

Ge cluster details

Kubectl cluster-info

Get node details

Kubectl get node

Create pod

Kubectl run firstpod –image=nginx

Get pods details

Kubectl get pods

Inside the pod

Kubectl exec –it firstpod --/bin/bash

Install curl in ubtu

1 .apt-get update

2. apt-get install curl

Curl localhost

Delete pod

Kubctl delete pod firstpod

Create pod using yaml

Kubectl create –f firstnode.yml

**Show all the label**

Kubectl get all –show-labels

**Show all the label of perticular**

Kubectl get all –selector=’app=fb’

The main advantage of kubernaties is that scale up the pod if request increses or scale down the pods if request decreases

Deploy application on Kubernates

deploy application kubernatis

step to deployed application on kubernatis

-create the spring boot application

-create docker image

-create deployement object using yml configuration to tell kubernatis to take this docker image and deploy in kubernatis pod.

-create service object using yml configuration to tell kubernatis to expose my application outside kubernatis cluster so user can able to acess it.

database

pull the image form dockerhub

create deployemnt object to deploy object in the pod

create service object to acess ouside the kubernatis cluster

Need to synch docker and minikube

use below command

eval $(minikube docker-env)

kind- PersistanceValumnClain

which help to allocate space for database

create db.deployement.yaml

create app.deployment.yml

mentioned the replica information in and enviourment information in yml file and also mntion nodeport information

run yml

command

kubectl apply -f app.deployment.yml

mentioned

deployement details

service details

and space valumn details

to store sentive details need to use configmap and secrete to store data in encrypted format

configmap- plain text eg. Hostname,

secrete - encrypted text eg. Usename and password of database

we can encrypt the text using kubernatis command

echo -n 'root' | base64

deployment object -

service object-

deploy the object in kubernatis command

kubectl apply -f db-deployemnt.yaml