10 July – 2024

Day 2

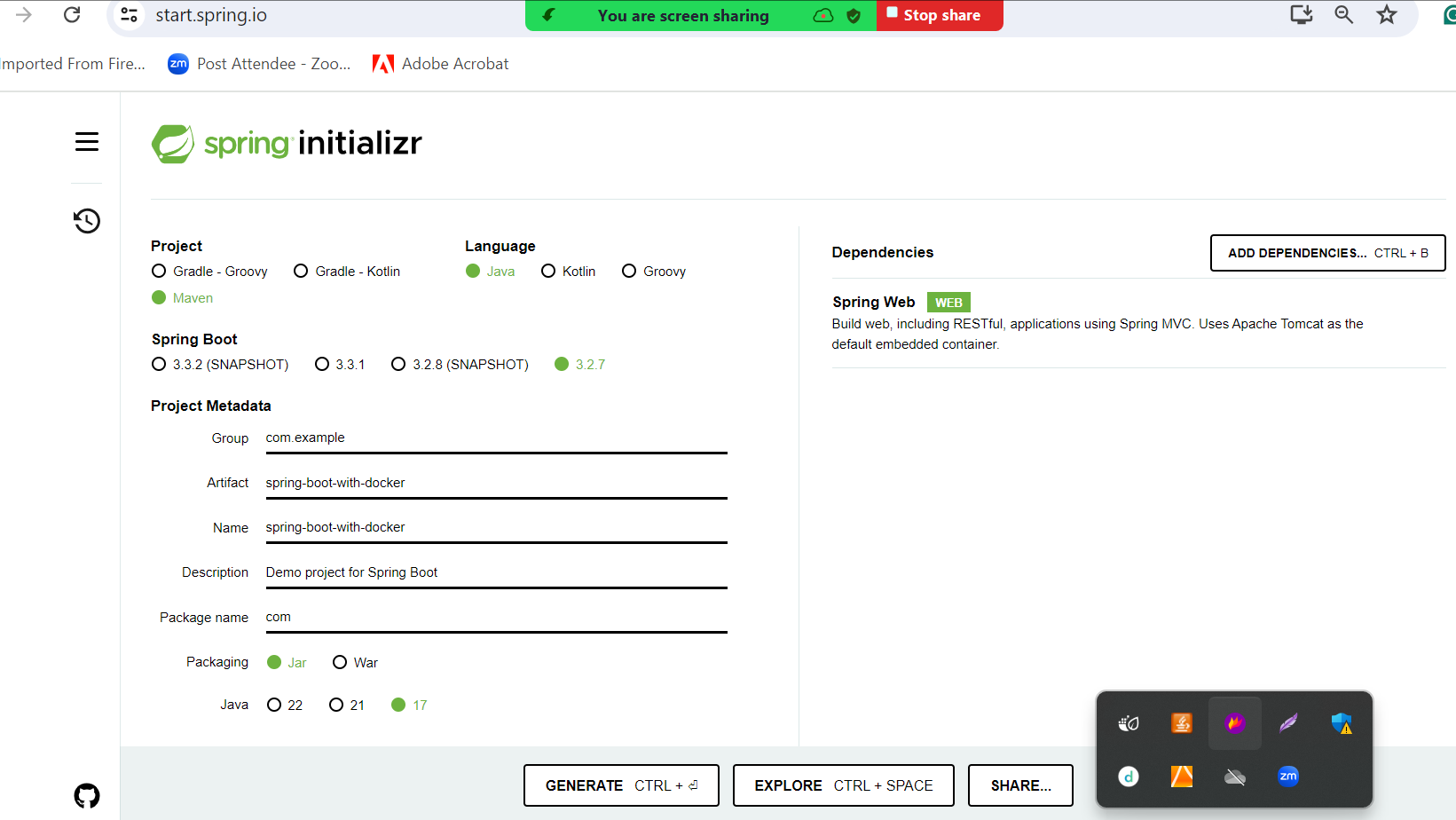
Integration and deployment

Docker

Docker compose

1. Creating image to run spring boot application

Using spring initializer we create simple spring boot application with web starter.



Please create more than one end points

Then create jar file using eclipse IDE with help of maven run 🡪 maven install

Then create Dockerfile

FROM openjdk:17

COPY ./target/spring-boot-with-docker-0.0.1-SNAPSHOT.jar .

CMD ["java","-jar","spring-boot-with-docker-0.0.1-SNAPSHOT.jar"]

**docker build -t my-spring-boot . -f Dockerfile**

if image is responsible to run the web application we need run the command as

docker run -d -p 9090:9090 imageName

-d : detached mode or background

-p publish port number

Right side or red colour port number is actual application port number is spring boot running port number.

Left side or green colour port number is expose port number it can be same or other number

docker run -d -p 9090:9090 my-spring-boot

docker run -d -p 9091:9090 my-spring-boot

docker run -d -p 9092:9090 my-spring-boot

then check on browser using port number as

90, 91 and 92

<http://localhost:9090>

<http://locahost:9091>

<http://locahost:9092>

if we want to see all running container we need to run the command as below. The below command display only running container.

docker ps

or

docker container ls

docker ps -a it display all container present in our local machine doesn’t matter they are running or stop.

To check specific contains logs details

docker logs containerId/containerName

stop the container

docker stop containerId/containerName

start the container

docker start containerId/containerName

remove container

docker rm containerId/containerName make sure container not running

or

docker rm containerId/containerName -f

command to remove the image

docker rmi imageName/imageId

or

docker rmi imageName/imageId -f

to remove all stopped container, images, network and cache memory.

docker system prune -a

mysql images

docker hub internally provide mysql image

docker run -e MYSQL\_ROOT\_PASSWORD=root -d -p 3307:3306 mysql:8.0

-e :ENVIRONMENT DETAILS like MYSQL\_ROOT\_PASSWORD as root

-d : detached mode

-p publish port number

docker ps

to connect MYSQL image OS.

docker exec -it containerId bash

docker exec -it c137e7443c48 bash

then after connected or container bash please login to mysql using below command as

mysql -u root -p

root

exit : first exit to come out from mysql to container os

exit : second exits to come out from container os to base os.

Creating image for react js ie frontend application

node --version : first verify in VM node js must be installed

First create the folder and inside that folder create react js project using below command as

npm install -g create-react-app

create-react-app react-docker

or

npx create-react-app react-docker

once project creation done we need to move inside a project folder using command as

cd react-docker

then open the project in VS code.

Edit in App.js file

To run the react js project we need to run the command as

npm start testing the project running or not.

After developed the code we need to build the project.

First stop the project using command as cnt + C or cnt + D

npm run build this command is use to create the build file.

After this command it create one build folder which contains all build file which we need to deploy on server like tomcat, ngnix or web logic etc.

React js project internally run on default port number 3000 with default web server.

Nginx open source sever. So docker provide nginx server image which help to create the image for UI technologies like HTML/CSS/JS or ReactJS or Angular JS.

Nginx default port number 80.

Now we create Docker file to create react js project image

Dockerfile

FROM nginx

COPY ./build/ /usr/share/nginx/html

docker build -t my-reactjs . -f Dockerfile

please verify image created or not using command as

docker images

to run the react js project we need to run the below command as

docker run -d -p 80:80 imageName

docker run -d -p 80:80 my-reactjs

docker run -d -p 81:80 my-reactjs

docker run -d -p 82:80 my-reactjs

using docker ps check all running container

then check the application on below url

<http://localhost:80>

<http://localhost:81>

<http://localhost:82>

we publish or push react js image in docker hub account.

docker login

it may ask dockerhubaccountid and password.

this command is use to connect local machine docker engine with Docker hub repository.

Once login done successfully. Before push we need to provide tag or identity for our user defined images

docker tag imageName dockerhubaccountid/imageName:tag

it can be version like v1,v2,v3 or number 1.0,2.0,3.0

**docker tag my-reactjs akashkale/my-reactjs:a1**

after tag created now we can publish the image in docker hub account

**docker push akashkale/my-reactjs:a1**

docker run -d -p 91:80 akashkale/my-reactjs:a1

docker run -d -p 99:80 akashkale/my-reactjs:a2

after run successfully using docker ps check the running container

then

<http://localhost:91>

docker run -d -p 92:80 dragosmn/my-reactjs:a1

docker run -d -p 93:80 paulodm/my-reactjsgood:v1

docker run -d -p 94:80 paulodm/my-reactjsgood:v2

releasing new version of images

1. First do some changes in project.
2. Re – build the project using command as npm run build
3. Re – create the image using command as docker build -t my-reactjs . -f Dockerfile
4. Re create the tag with different version
5. Creating new tag with version as a2 docker tag my-reactjs akashkale/my-reactjs:a2
6. Then push the new version docker push akashkale/my-reactjs:a2