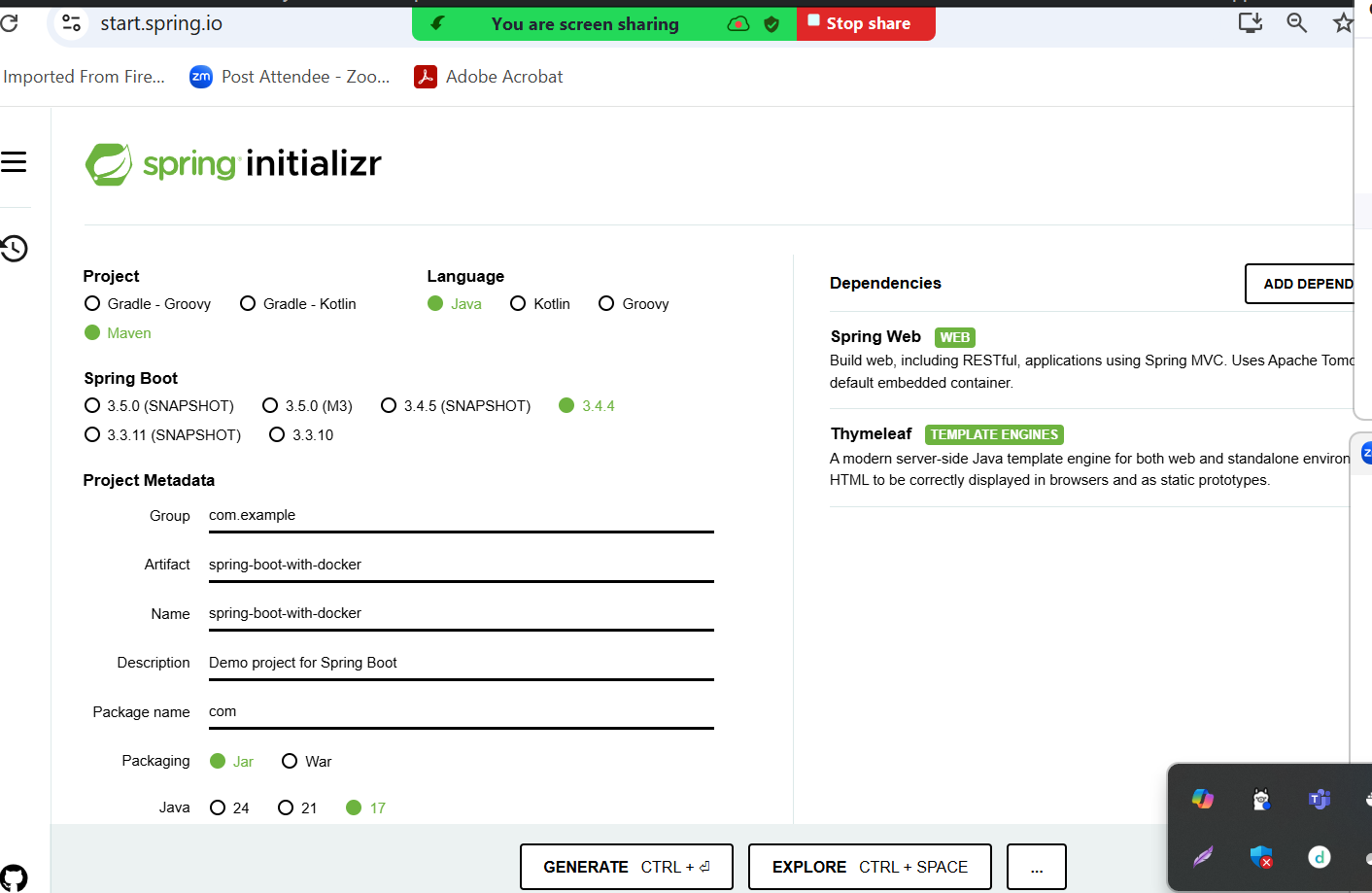
1. Creating docker image to run spring boot application

Create spring boot project with starter (depending upon your requirement)



**index.html** inside template folder

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<h2>Welcome to Spring boot with Docker</h2>

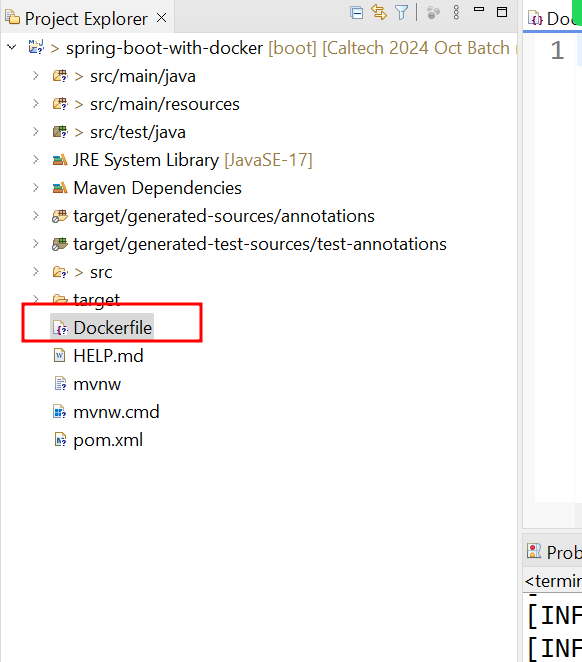
</body>

</html>

After development you need to create the jar file.

Using eclipse IDE.

Creating Dockerfile in Eclipse IDE.



**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"]

Now need to create the image

Open the terminal or command prompt in location where Dockerfile present.

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

**After image created. If image is responsible to run the web application. We need to run the command as**

**docker run -d -p 8080:8080 imagename**

**-d : detached mode**

**-p public port number yellow colour port number. Actual application port number**

**Red colour port number is expose port number. It can be same or different.**

**After run to verify container running or not. You need to execute the command as**

**docker container ls it display all running containere**

**docker contains ls -a it display all container (stopped as well as running)**

**open the browser**

[**http://localhost:8080**](http://localhost:8080)

**container docker command**

docker stop containerId/containerName

docker start containerId/containerName

docker rm containerId/containerName

docker rm containerId/containerName -f

**remove image**

docker rmi imageName/imageId

docker rmi imageName/imageId -f

**to remove all container, images, cache memory, network**

docker system prune -a

**running spring boot image with custom name**

docker run --name=spring-container -d -p 8080:8080 my-spring-boot

before publish the

1. We need to create tag for that image. Tag is unique identity with new update in same image

docker tag imageName dockerHubAccountId/ImageName:version

version can be any number with alphabets

**docker tag my-spring-boot akashkale/my-spring-boot:v1**

1. After created the tag we can push the image

docker push **akashkale/my-spring-boot:v1**

**if you get any error**

1. docker login

provide your accountId and password

docker pull akashkale/my-spring-boot:v1

docker run -d -p 8181:**8080** akashkale/my-spring-boot:v1

open the browser

<http://localhost:8181>

**running mysql image**

this command pull mysql 8 version database and run with set password as **root**

**docker run --name mysql-container -e MYSQL\_ROOT\_PASSWORD=root -d -p 3307:3306 mysql:8**

-e : Environment details

**docker container ls** this command to check mysql container running or not

Open/Connect mysql container OS

**docker exec -it mysql-container bash**

it open mysql OS Image

to connect mysql database

**mysql -u root -p**

password : **root**

spring-application database

spring boot image mysql pre defined image

**openjdk** once we run the image container start

network

OS Image OS image

mysql database not present

in application.properties file we need to write mysql container name, port number, username, and password.

both container run on different OS.

Base Machine

Window

**Docker compose:** Docker compose is a toolkit which help to run more than one containers. Those container are running independently or they can communicate with each other to share the data.

Docker compose use **docker-compose.yml** file. Inside this file we need to write all images as well as container details which help to run more than one containers.