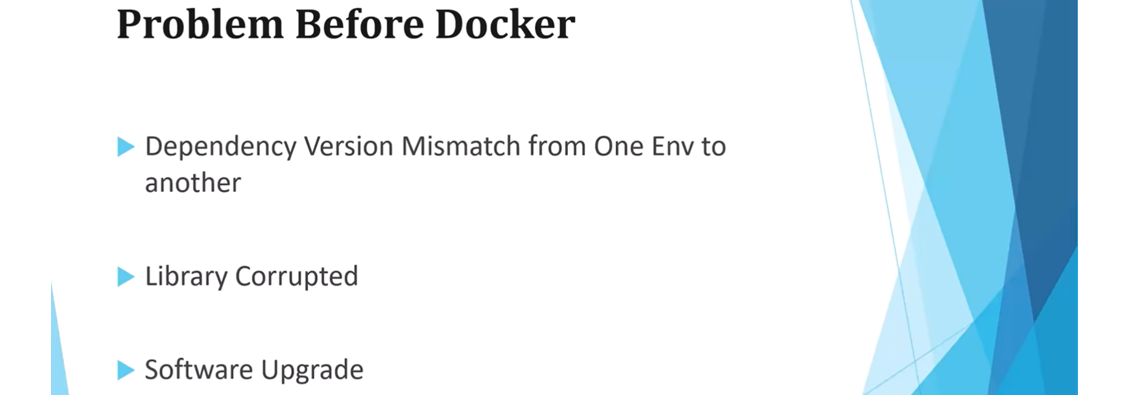
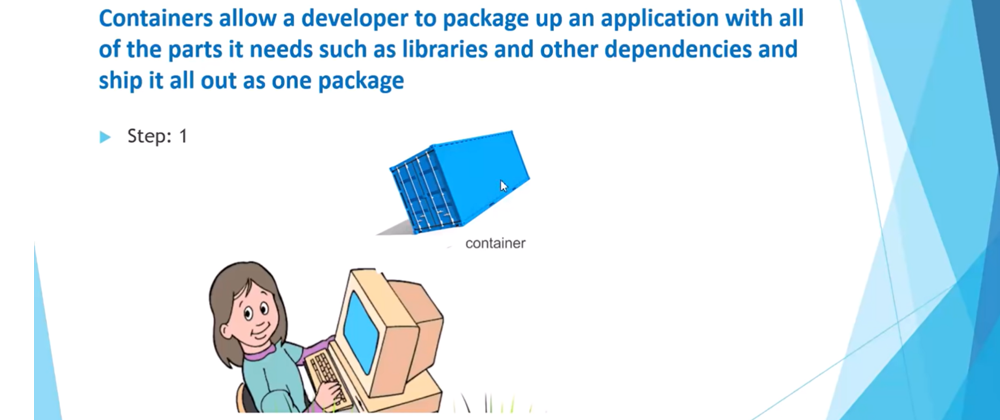
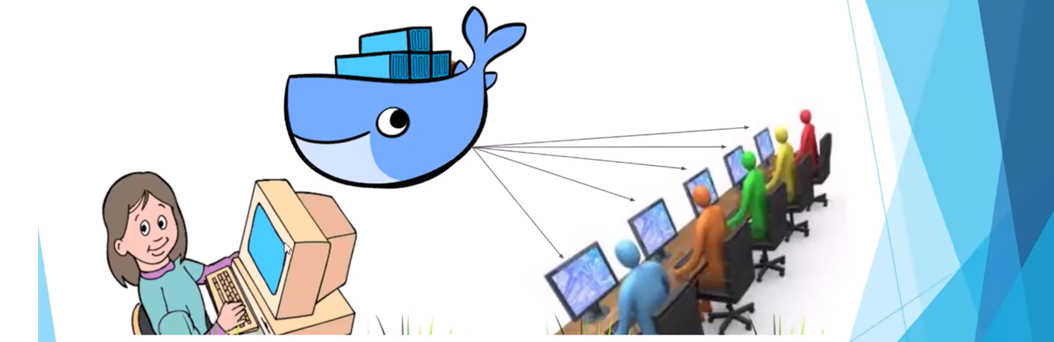
* **­­What is Docker & How to install it in Windows OS**



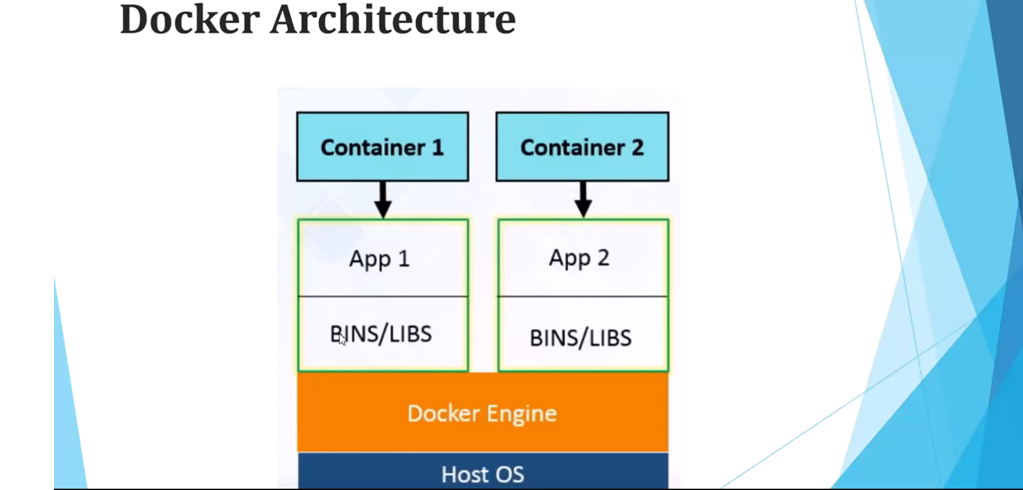


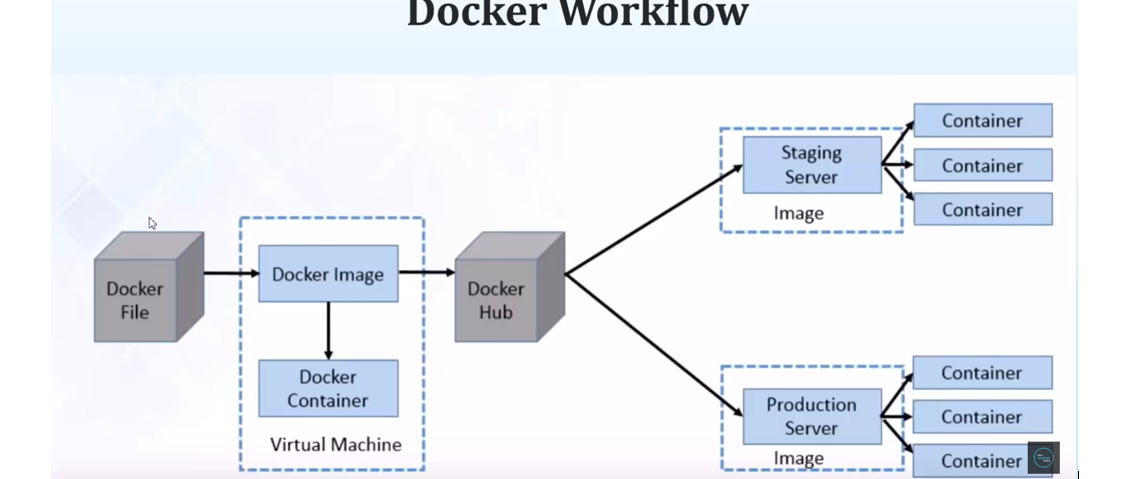












Docker hub is like a central repository of docker images.

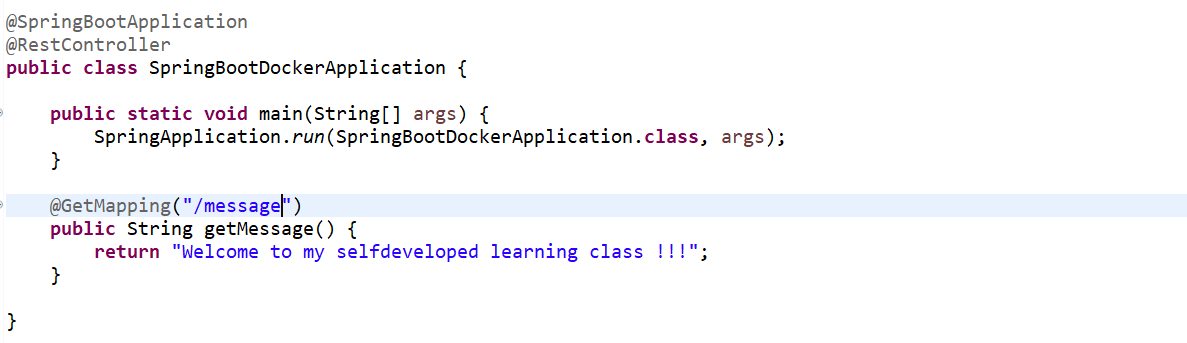
* **Docker - Dockerizing your Spring Boot Application**

We can dockerize any java application so let’s create a spring boot application and then we will create a Dockerfile and then we will create a docker image and we will dockerizing our application means creating a docker file creating a docker image and run or deploy that image on docker container the entire process called dockerizing our application.

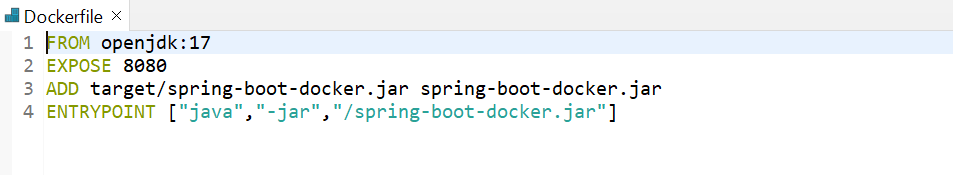
**Application**- **spring-boot-docker**

**Dependency** – **Spring Web**

Now let’s create one rest endpoint to verify whether our application is running over container or not.



Let’s create one docker file



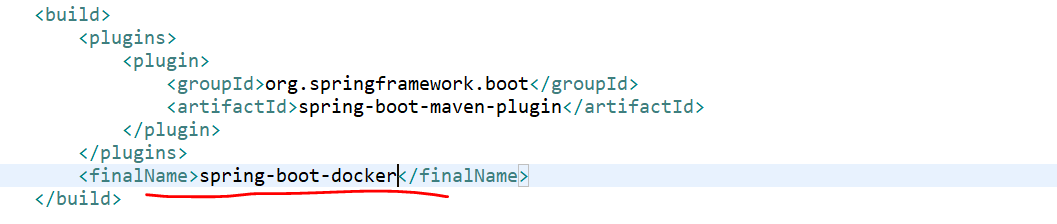
1st one is **FROM** means start with a base image containing with our java runtime, so here we are going to write a jdk related thing. This openjdk:17 image is available in our docker hub.

2nd one is **EXPOSE** I am giving 8080, so the mean of this expose make port 8080 available to outside this container.

3rd one is **ADD** so here we need to add our jar of this application (spring-boot-docker). So here let’s specify the path where we can get the jar **target/**spring-boot-docker.jar spring-boot-docker.jar

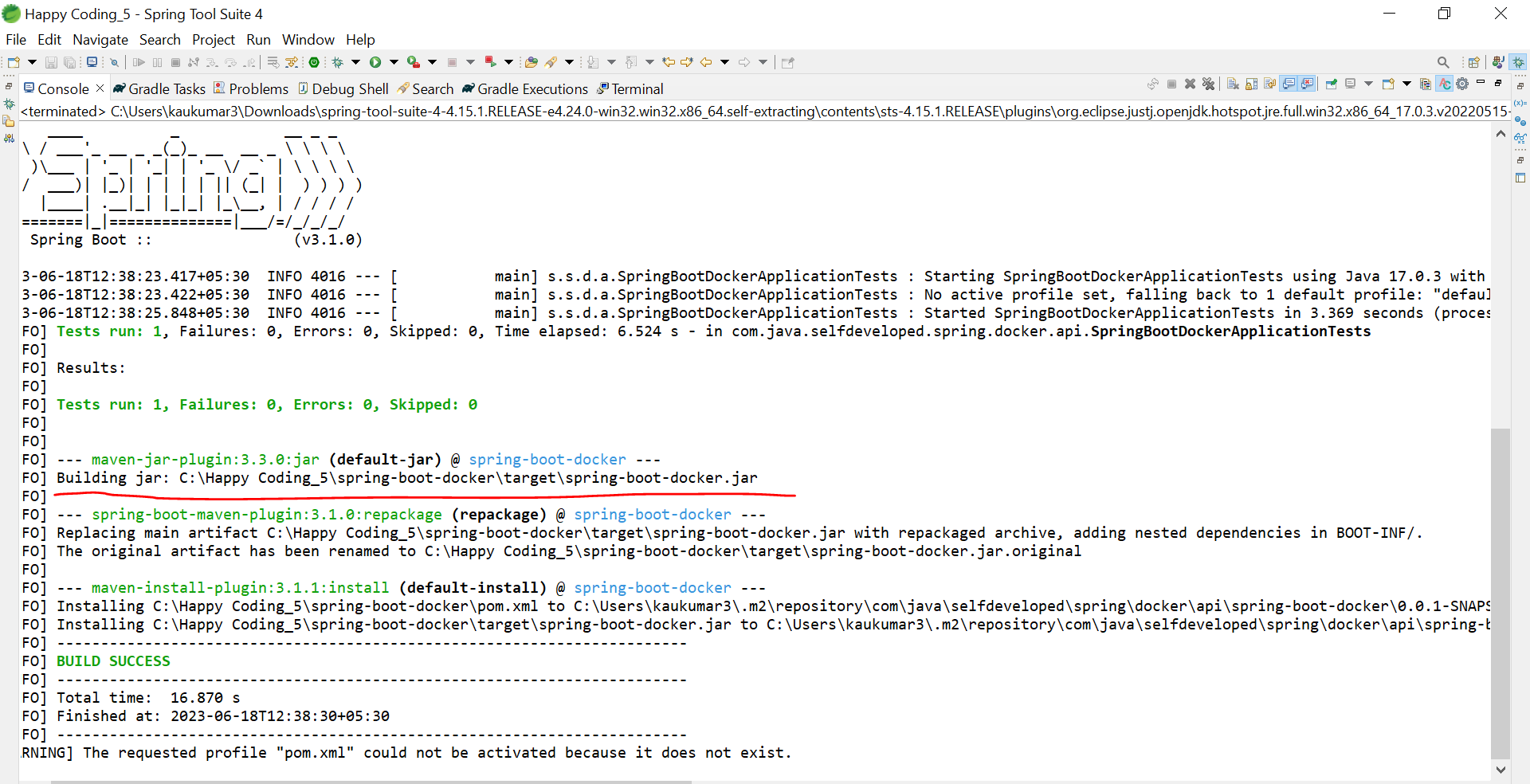
4th one is **ENTRYPOINT** so, here we need to specify the command to run the jar. ["java","-jar","/spring-boot-docker.jar"].

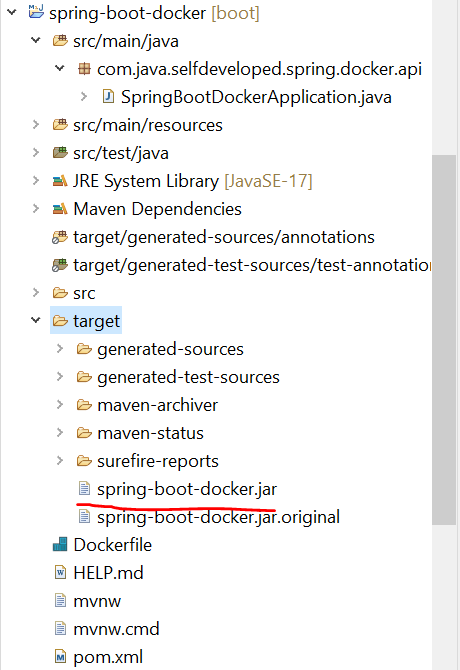
Till now we didn’t create the jar. So, now let’s create the jar first. So go to pom.xml file in <build> tag add one tag called <finalName> and specify jar name.



Now let’s generate our jar.

App->Run AS->Maven Install





Now finally we have created the jar.

So now we need to create a docker image of it.

Go to project directory and open the command prompt, as we already installed the docker desktop so we can run all command through cmd as well. Type one command.

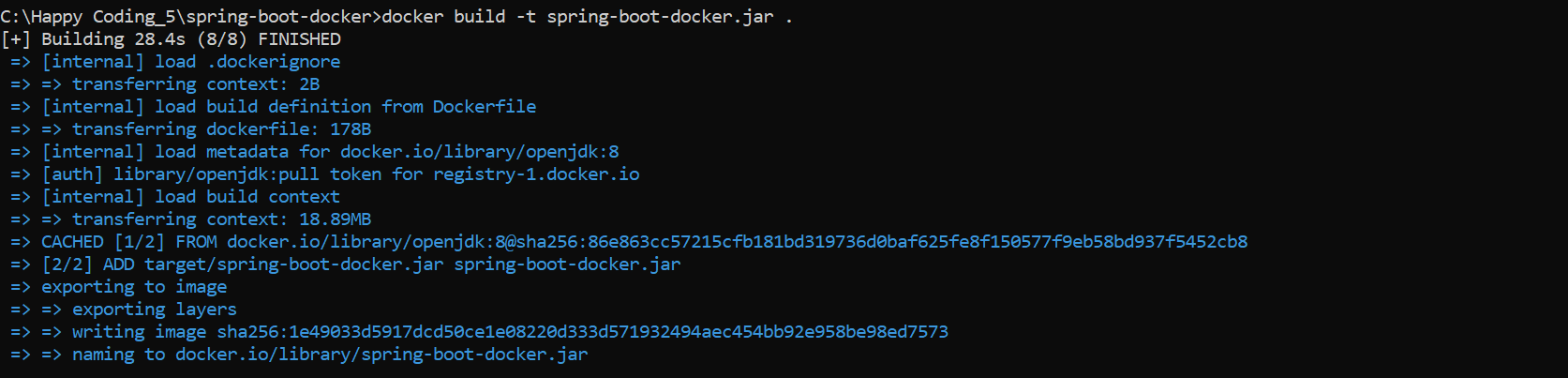
**++ docker build -t spring-boot-docker.jar .**

so as of now there is no images in our docker repository.

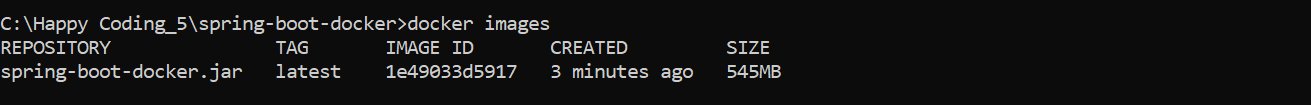
**++ docker logout**

**++ docker login**  //+dockerjimaihu +KrTum5h87@7

**++ docker build -t spring-boot-docker.jar .**



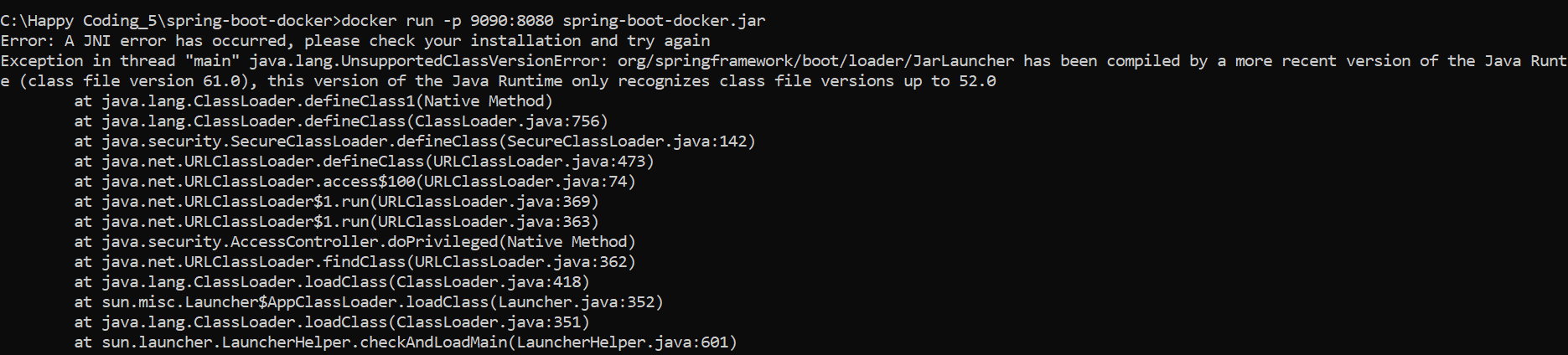
**++ docker images**



Now we need to run this docker images.

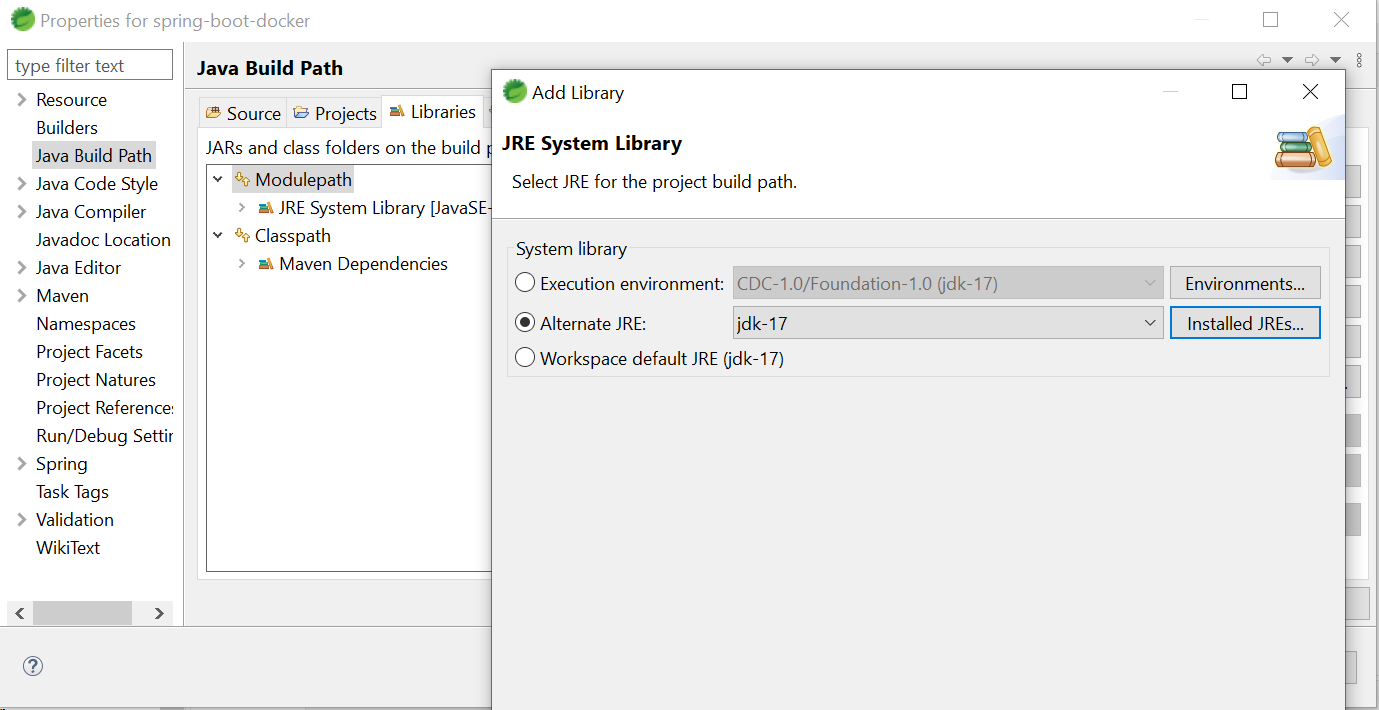
**++ docker run -p 9090:8080 spring-boot-docker.jar**

So, this 8080 is my local port what we specify in our docker file and this 9090 will be run on our container.



Getting exception so. Let’s fix this issue first.

Java build path-> Library->alternate JRE-> jdk 17 path

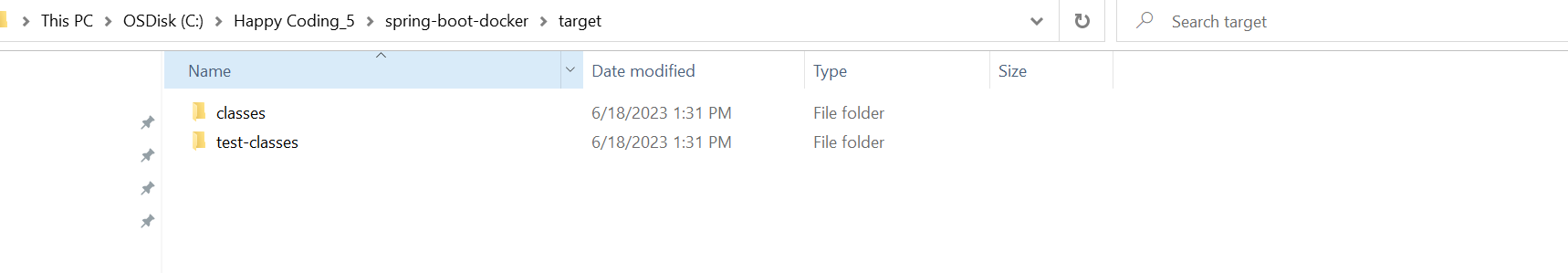


Java app->configuration->java compiler-> 1.8 selected( compiler version should be lower version of our jdk and jre earlier it was 17)



Now build the project again and create a docker image and then try to run it.

While do Maven clean our jar got remove

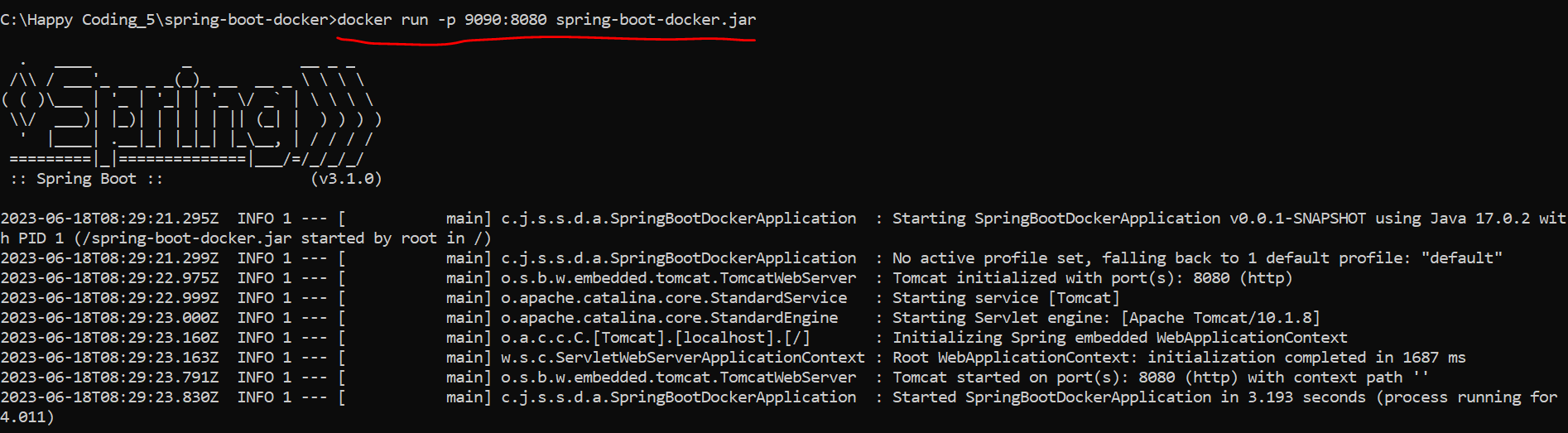


Now let’s create our jar using **Run AS** -> **maven install**

Now we created our jar again and now let’s create a docker image and let’s try to run it.

Ultimately, we found the issue the issue was we are running our application in local on jdk 17 and we specified the jdk version 8 on our Dockerfile.

So always keep in mind our java compiler version always should be lower than our jdk or jre version.

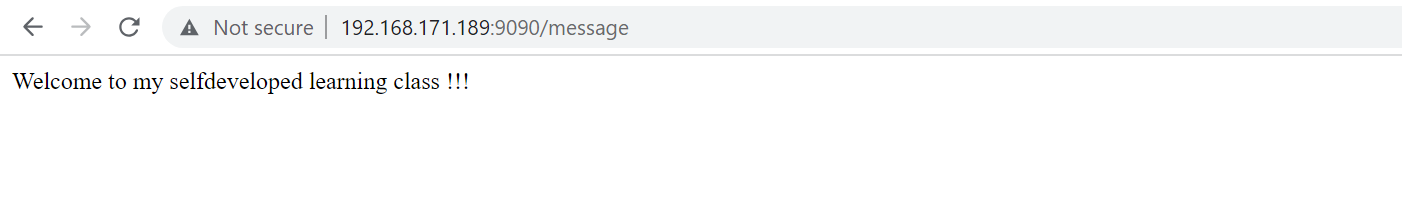


So, here 9090 is our container port and 8080 is our local port.

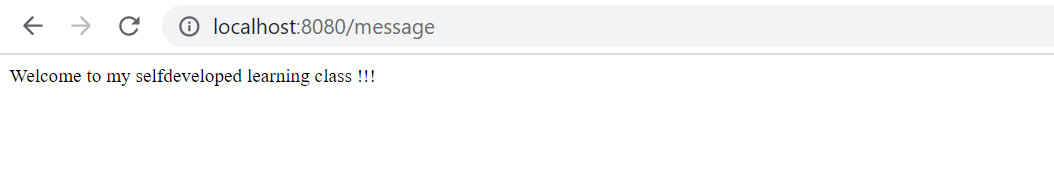
Now our application is deployed and running on docker container you can access on port 9090. Now once our docker image is deployed in our container you can be able to access it.

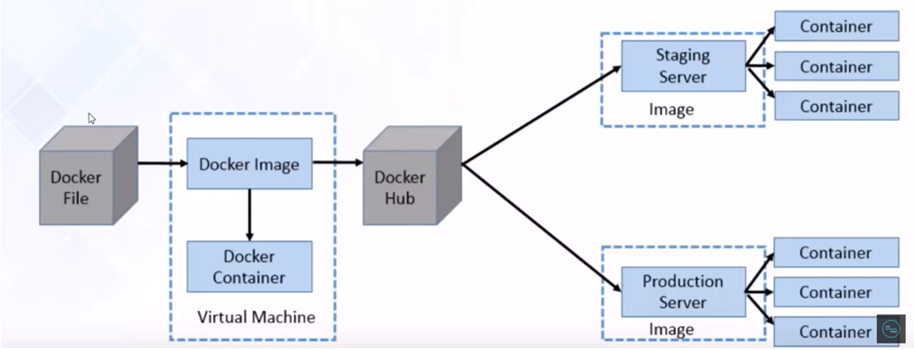
Now instead of our local host let’s use our System IP address.

<http://192.168.65.0:9090/message>



Similarly, we are exposing our local port 8080 as well so we can access on local host:8080 as well.





So, this is what our docker flow we have created the docker image and we up our instances in our docker container and we are able to access it our port 9090.

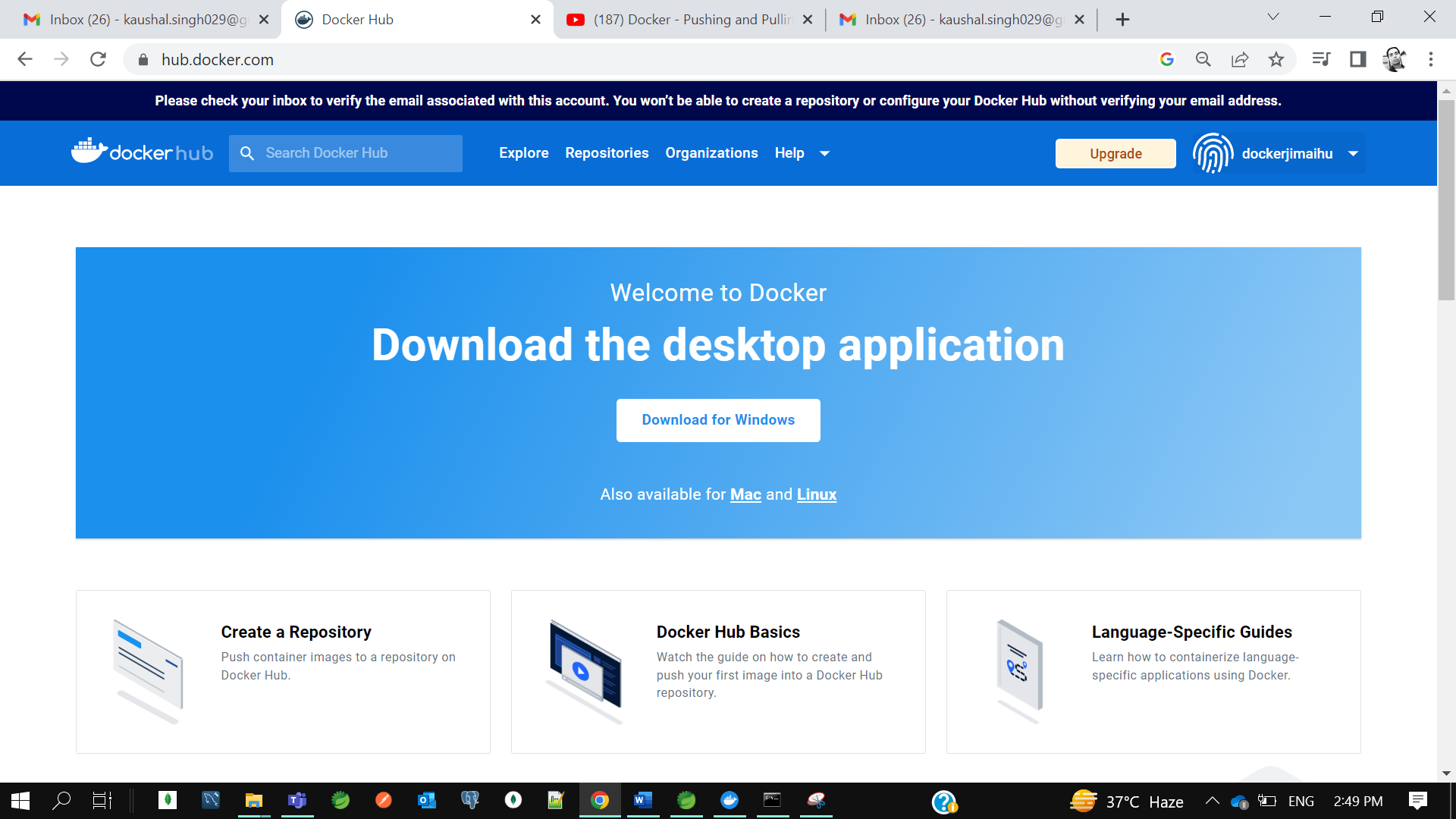
* **Docker - Pushing and Pulling Docker Image to Docker Hub**

Let’s see how to push our docker image to docker hub and how to pull it from docker hub.

What exactly this **docker hub**. Docker hub is a kind of repository. as we have a repository like Github where we can store our SourceCode similarly docker image we can store in our **docker hub**.

As this docker hub is a kind of repository so we need to create an account in our docker hub.

**Credentials**: dockerjimaihu/KrTum5h87@7

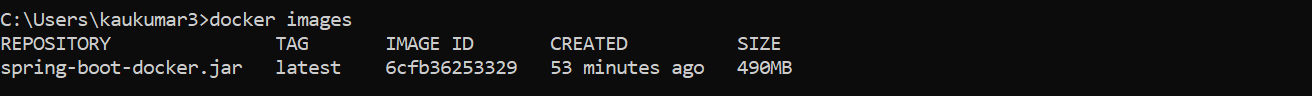


So, we logged in.

Now we need to login from the cmd.

**++ docker login**

**++ docker image ls**



So, we have created one docker image in previous tutorial so let’s push to our docker hub.

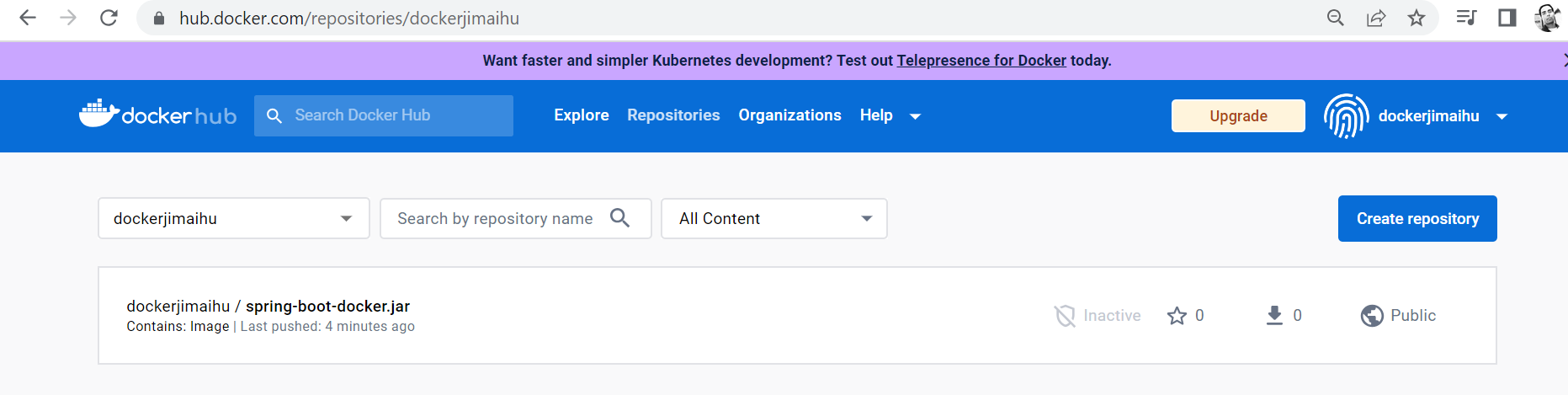
As docker hub is a public repository. so, we need to specify some tag to our docker image to identify that this docker image belongs to this **dockerjimaihu**. So, we need to specify a tag. Our tag should be same as our docker userid.

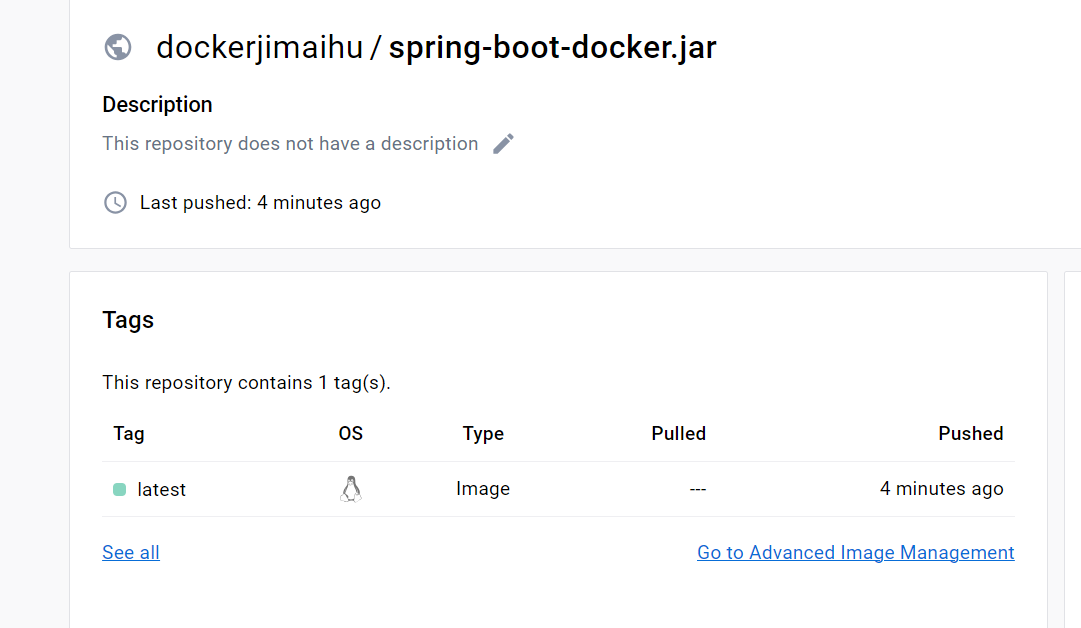
So, to specify a tag there is a command in docker

**++ docker tag spring-boot-docker.jar dockerjimaihu/spring-boot-docker.jar**

To push our docker image on docker hub

**++ docker push dockerjimaihu/spring-boot-docker.jar**

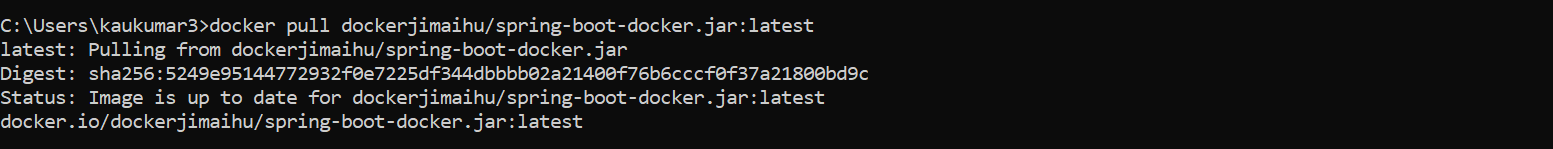




So, we can go to our docker hub repository, and we can see our **spring-boot-docker** image.

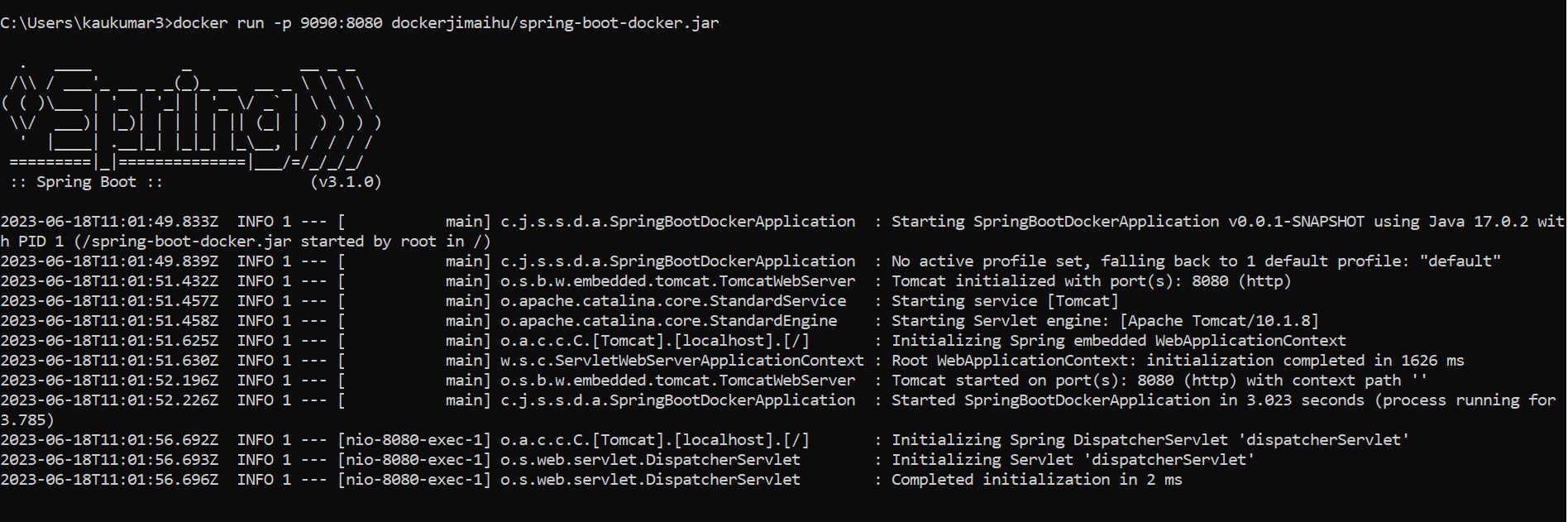
Now let’s pull this image and run the docker image

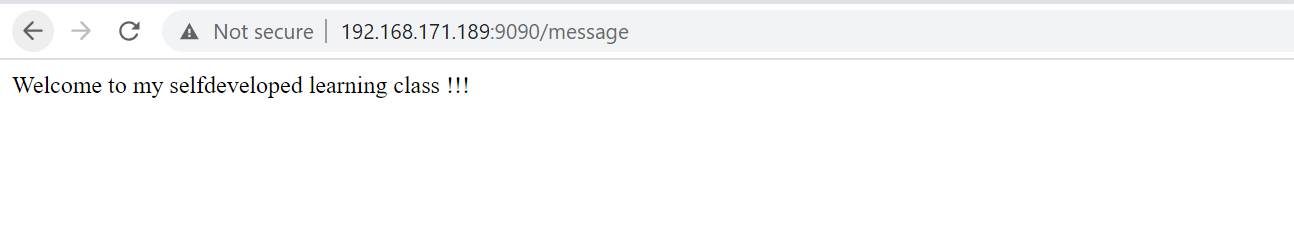
**++ docker pull dockerjimaihu/spring-boot-docker.jar:latest**



Now we pulled the image. Now let’s run…

**++ docker run -p 9090:8080 dockerjimaihu/spring-boot-docker.jar**





So instead of our localhost we need to use our Ip Address.  
so this is so easy developers no need to install anything in their local machine, they just need to pull the image from docker hub, and they need to run it on their local system.

* **Build and push Spring Boot Docker images using Maven plugin**

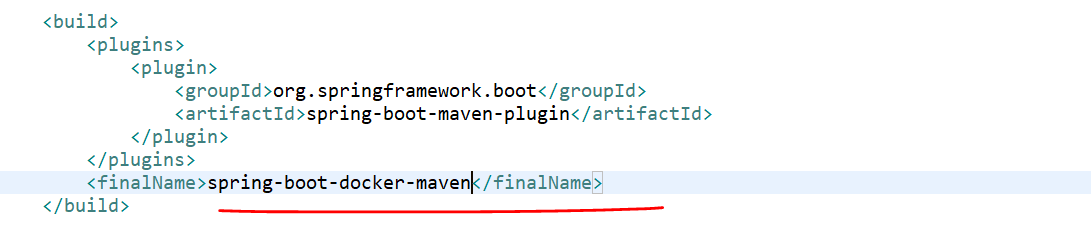
So earlier we were using docker file to create a docker image now let’s use maven plugin.

Now let’s create a springboot project and we will dockerize using maven plugin.

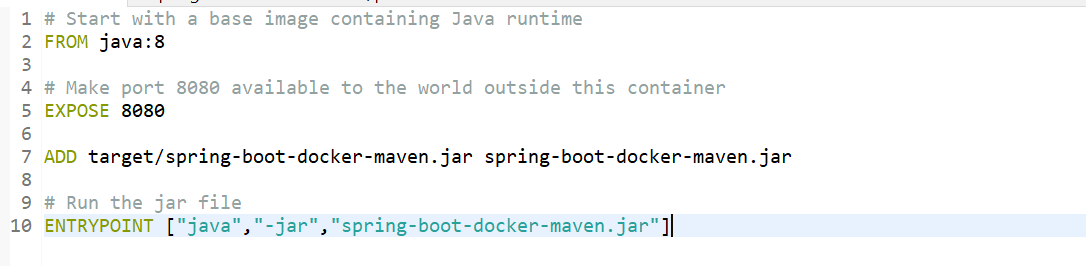
**App**- spring-boot-docker-maven

**Dependency**-Spring Web

Now go to pom.xml and let’s change the jar name here



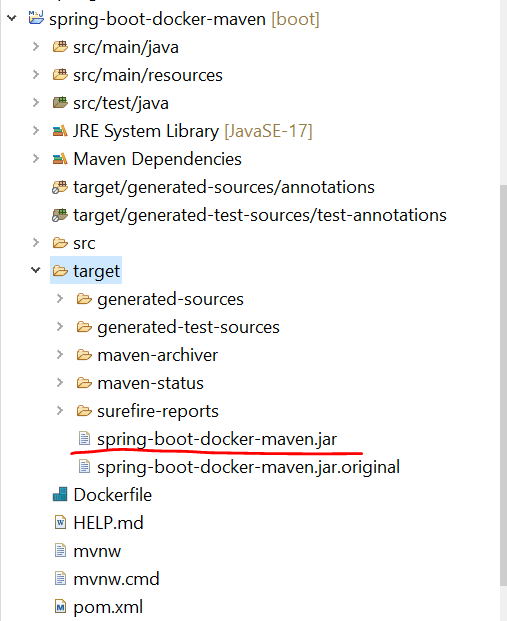
Now let’s create the Dockerfile.



Now we need to generate the jar.

Run AS -> **Maven Install**





>>> Now let’s add the maven plugin to build an image and push it to the docker hub.

so inside plugins let me add the plugin.



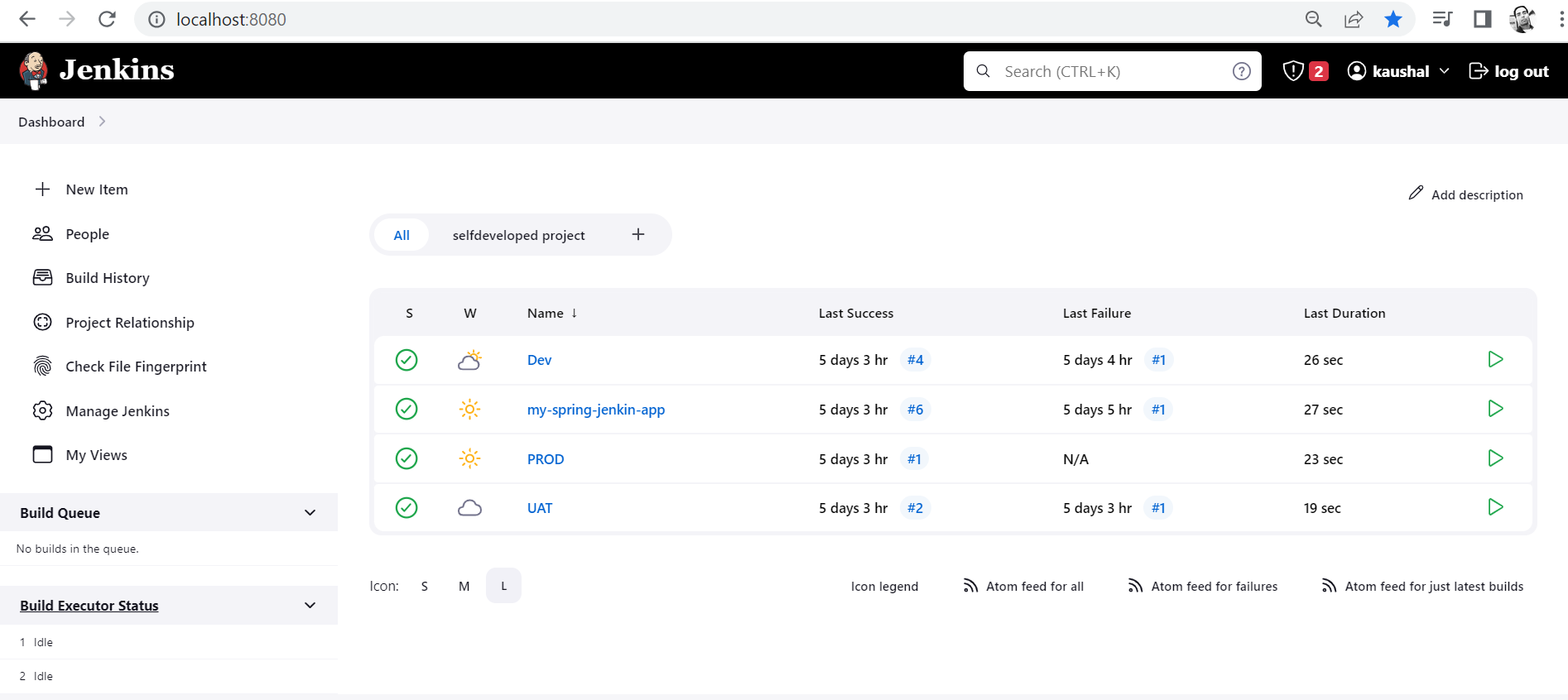
So, the plugin is **dockerfile-maven-plugin**. The version is 1.4.0. in the repository we need to specify the docker\_id here. docker\_id is nothing the username of your docker hub. So here the docker\_id is **dockerjimaihu** andthen the project artifact id which is **spring-boot-docker-maven** so, with the same name the image will be deployed to the docker hub. We specified the jar file here target/**project.build.finalN**ame which is **spring-boot-docker-maven.jar.** we set the goal as build and push inside the install.

So, once you run the docker install it will cover both build and push.so, it will build the image and it will push to the docker hub.

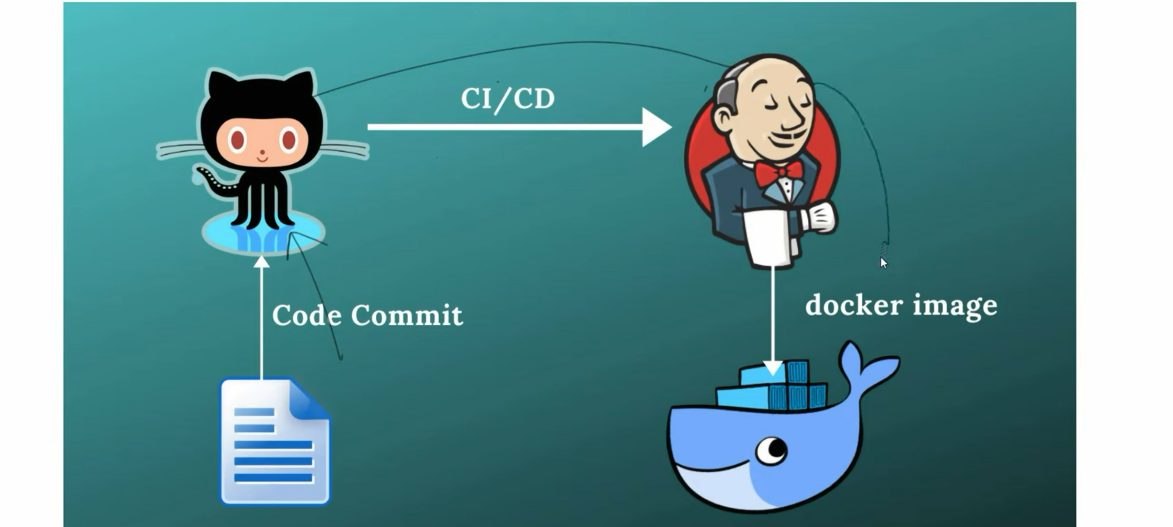
Before docker install let’s first login to the docker hub through cmd.

\*\*\* I got some exception here\*\*\*\*\*\*\*\*\*

* **Building Docker Images using Jenkins step by step | Devops Integration**



Let’s start with Devops Integration. So, you should have basic knowledge about docker and jenkins.



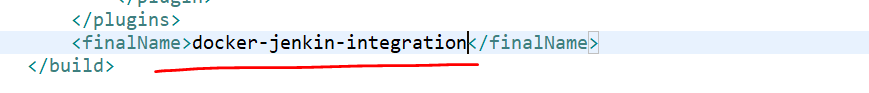
We will create one github repository. and that repo we will integrate with jenkins. So that each code commit to my github jenkins will process continuous Integration and deployment. Once the Jenkin build is succeeded, then jenkins will create a docker image for us and he will push the docker image to docker hub.

So, here we are not going to create a docker image manually and we are not going to push the docker image to docker hub. All the dockerizing stuff would be done by jenkins itself.

Let’s create a java project and that java project will sink with github and then github will link with Jenkin.

**App**- docker-jenkin-integration

Now we specify the jar name in our pom.xml



**Github:**

Create a github repository with name **docker-jenkin-integration** and commit our code to github.

We have committed and pushed our changes to github using below cmd-

Gilt Repo- <https://github.com/kaushal8707/docker-jenkin-integration.git>

* git init
* git status
* git add .
* git commit -m "first commit"
* git branch -M main
* git remote add origin https://github.com/kaushal8707/docker-jenkin-integration.git
* git push -u origin main

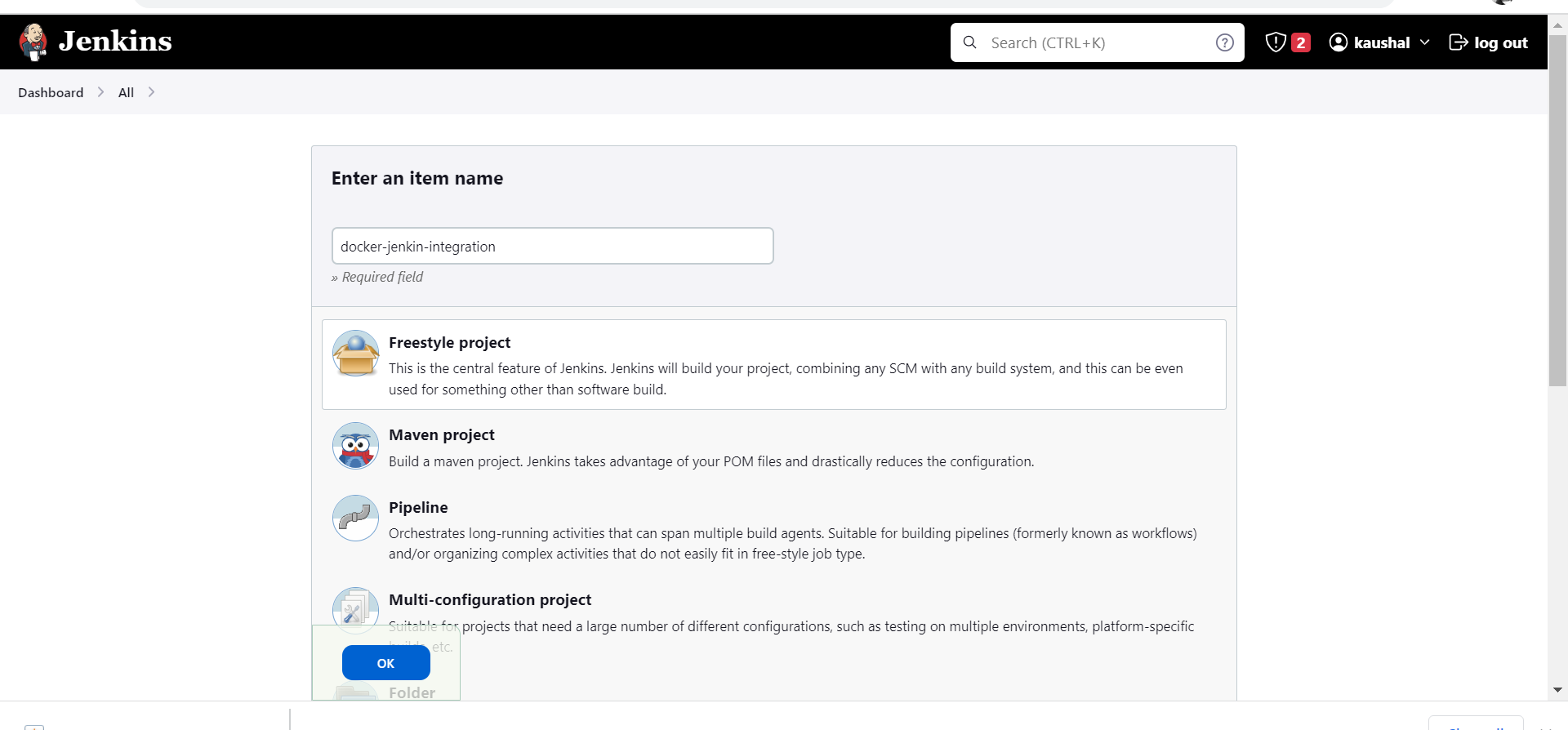
we have committed our code to main branch.

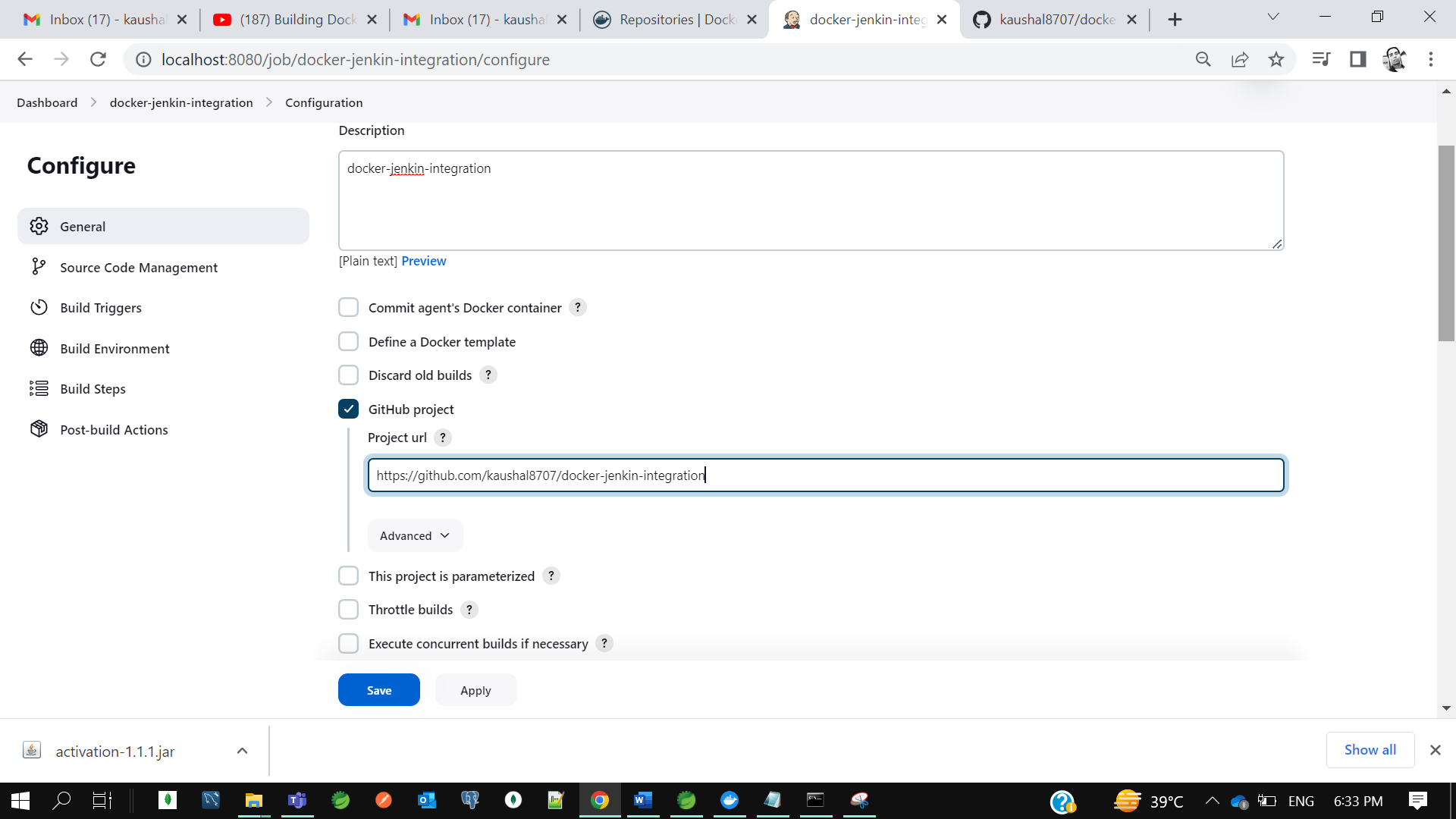
So, we have created code base and we sink with github.

Now this github need to sink with jenkins.

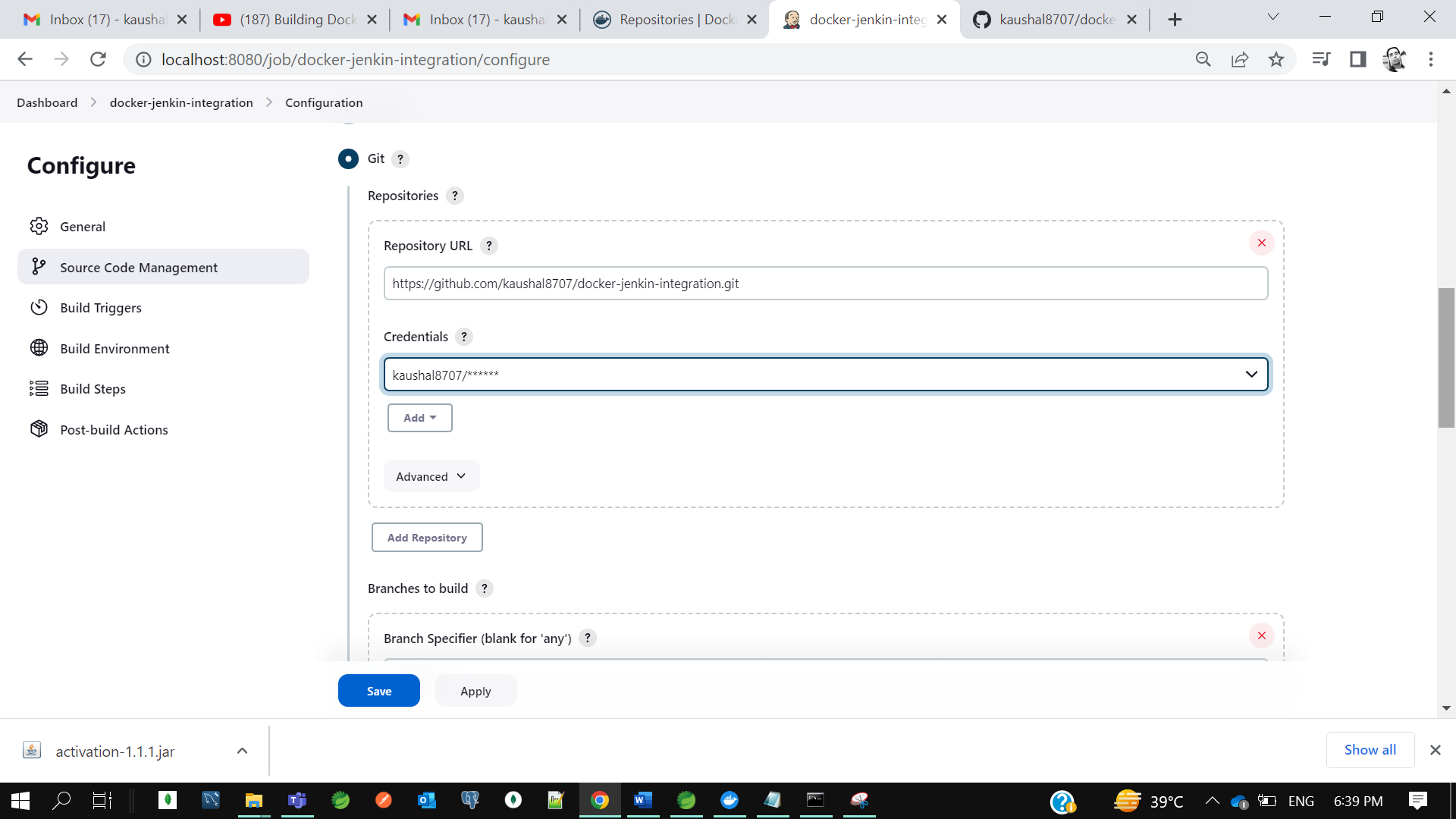
Now run the jenkins war our start jenkin which will run on port 8080

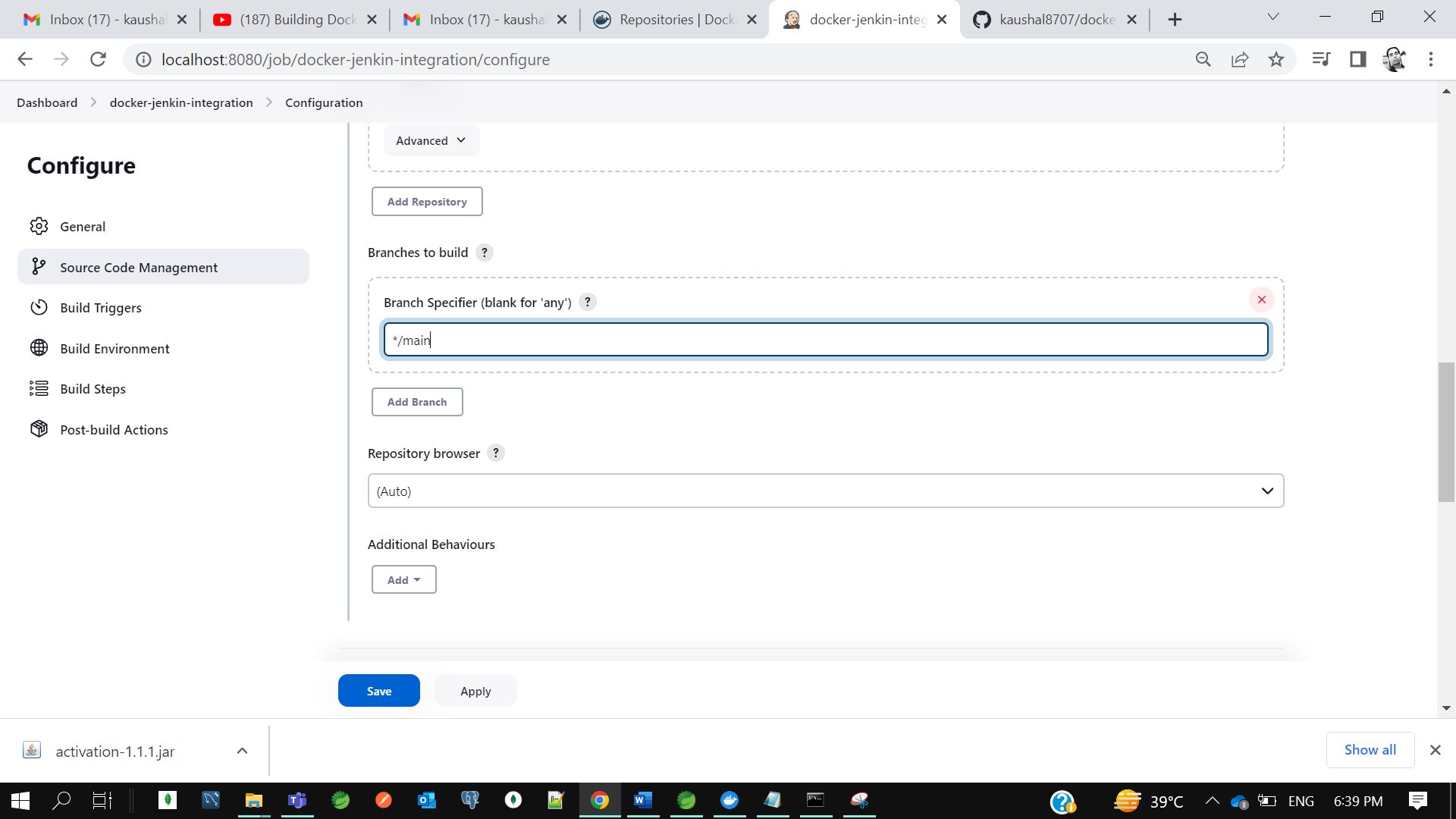
Now create a job and select free style project. And follow jenkins initial tutorial.

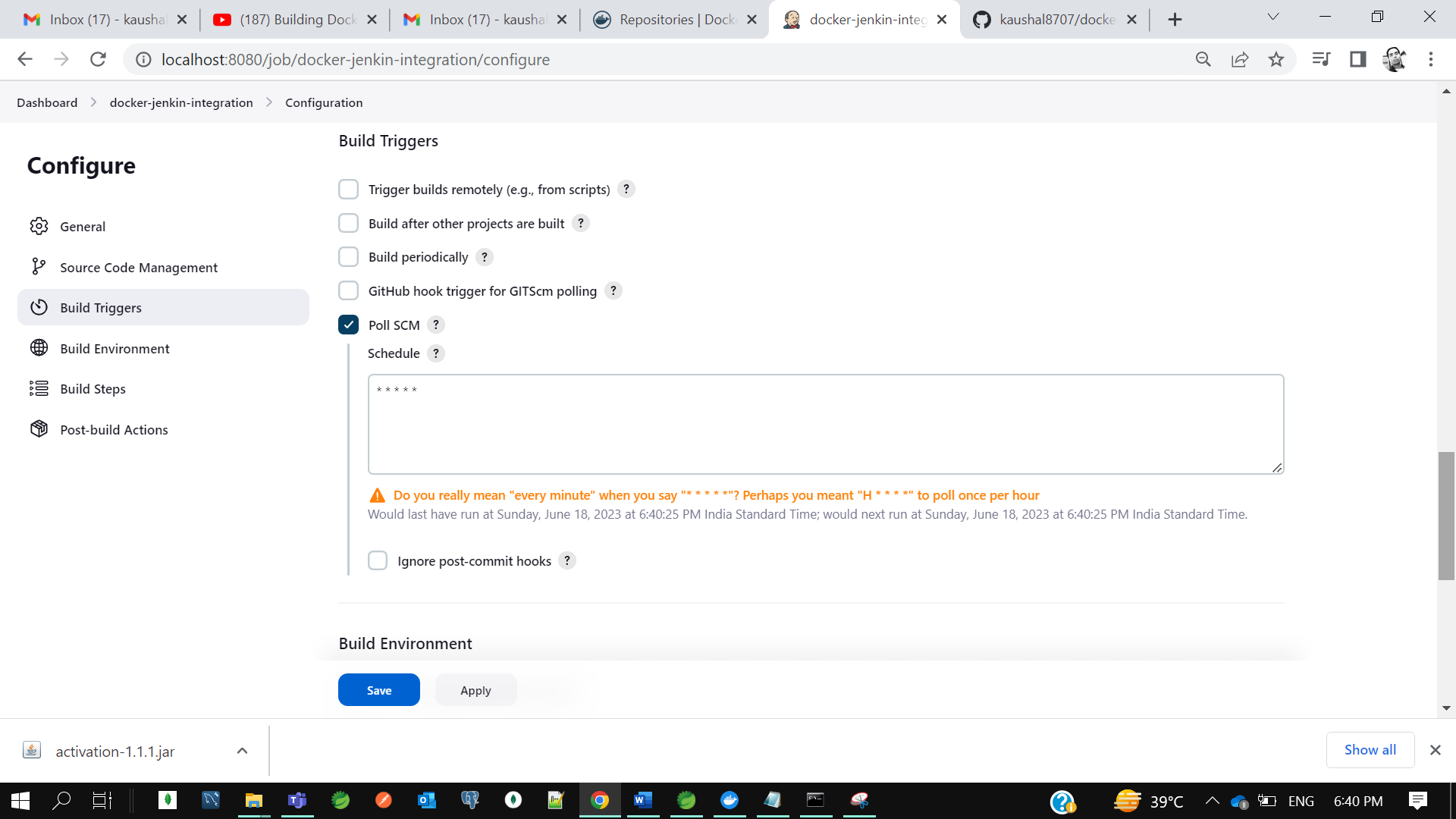


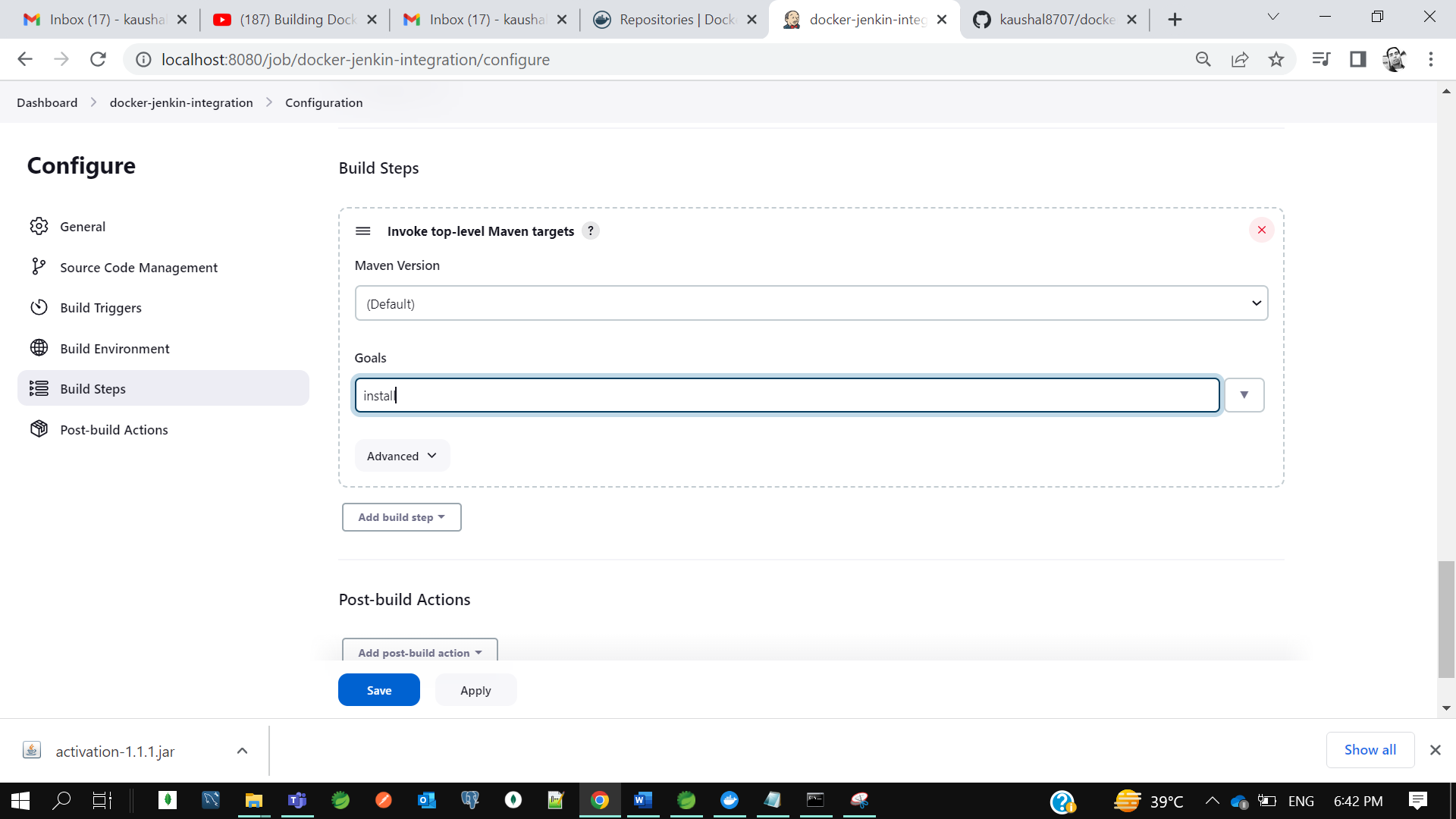


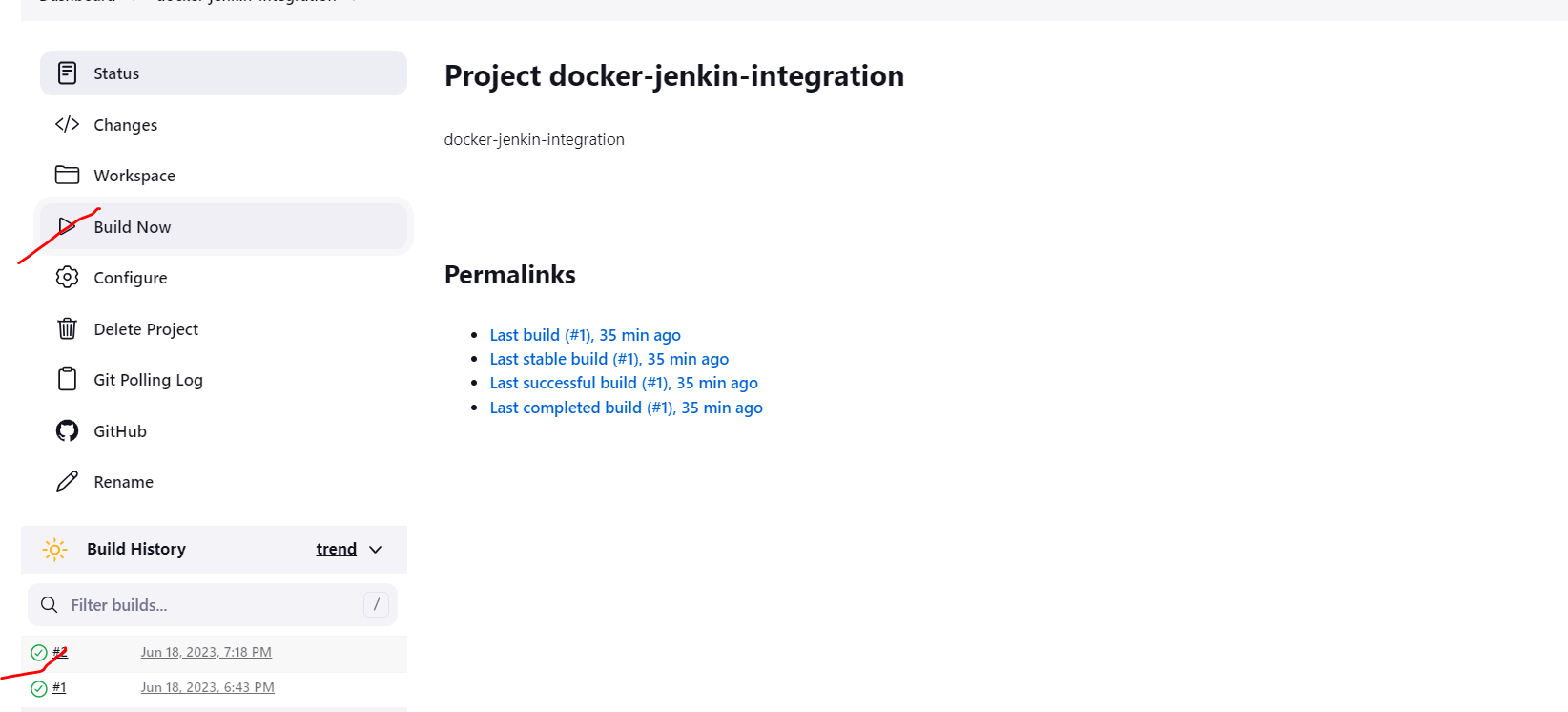
**Jenkin credential- kaushal8707/KrTum5h87@7**











Our build got successful.

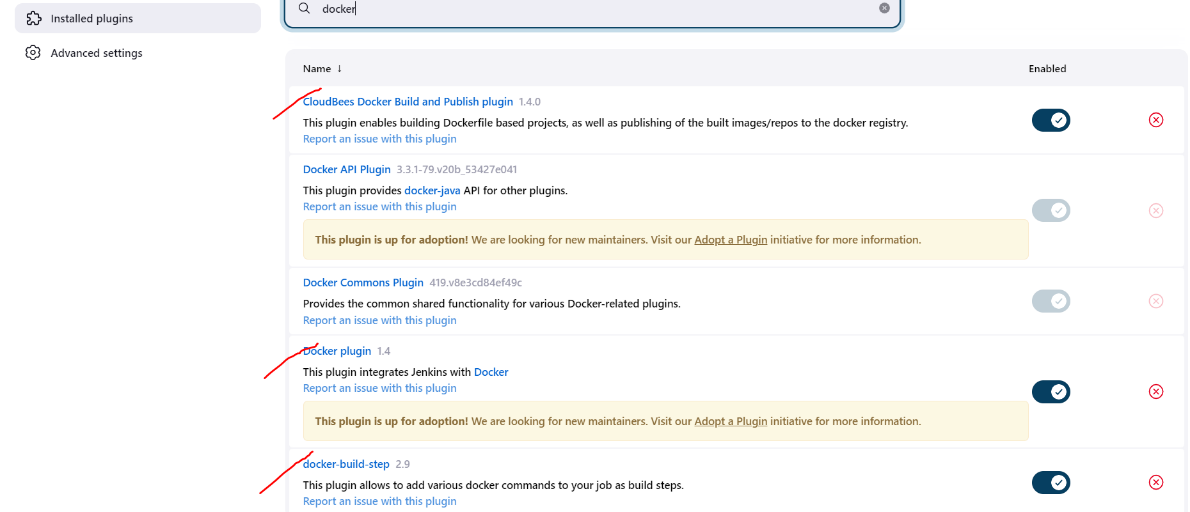
Now let’s verify the sink between github and jenkins we will change the code and commit in github the build should automatically start in jenkins. So, when we will commit jenkins will start the maven goal install which we configure to start the maven build.

So, we can see our build got succeeded.

Now we need to integrate jenkins with docker

So, go to jenkin we need to add one plugin. As I already installed docker plugin

Jenkin Home-> Manage jenkins->plugin-> installed plugin-> 3 docker plugin as marked



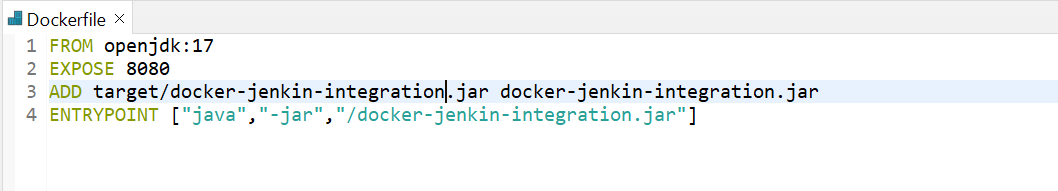
To verify either your jenkins plugin installed successful or not.

Go to jenkins-> project->configure-> add build steps.



\*\*\*Now how jenkins will understand your java code that we need to create a docker image.

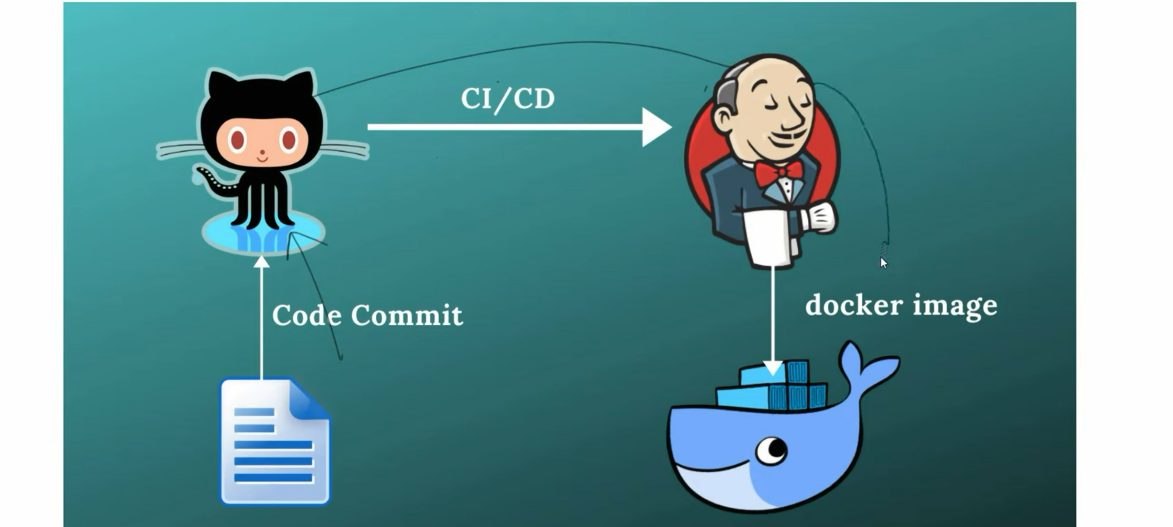
\*\*\*How we can instruct jenkin that take this code and build a docker image for us.so for that we need to create the Dockerfile in our code. In our java code we will add Dockerfile and that we will push to github. Now jenkin will load it from github and he will process the build once the build is succeeded jenkin will able to create the docker image and he will be able to push the docker image to docker hub.



Let’s commit this code to git.



So, build got successful.



So, in jenkin we added the plugin and in our code we already added the docker file.

How we can instruct jenkin that take this code and build a docker image for us.so for that we need to create the Dockerfile in our code. In our java code we will add Dockerfile and that we will push to github. Now jenkin will load it from github and he will process the build once the build is succeeded jenkin will be able to create the docker image and he will be able to push the docker image to docker hub.

So for that I need to give Docker Hub credential to Jenkins.

Project jenkin-> configure->add build step->docker build and publish->

In repo name of docker hub give **tag\_name/docker\_image\_name**

And along with it we need to add our docker hub credential



Credential – dockerjimaihu/KrTum5h87@7

Now start the build so it will create a docker image and push that image to docker hub.