AIM:

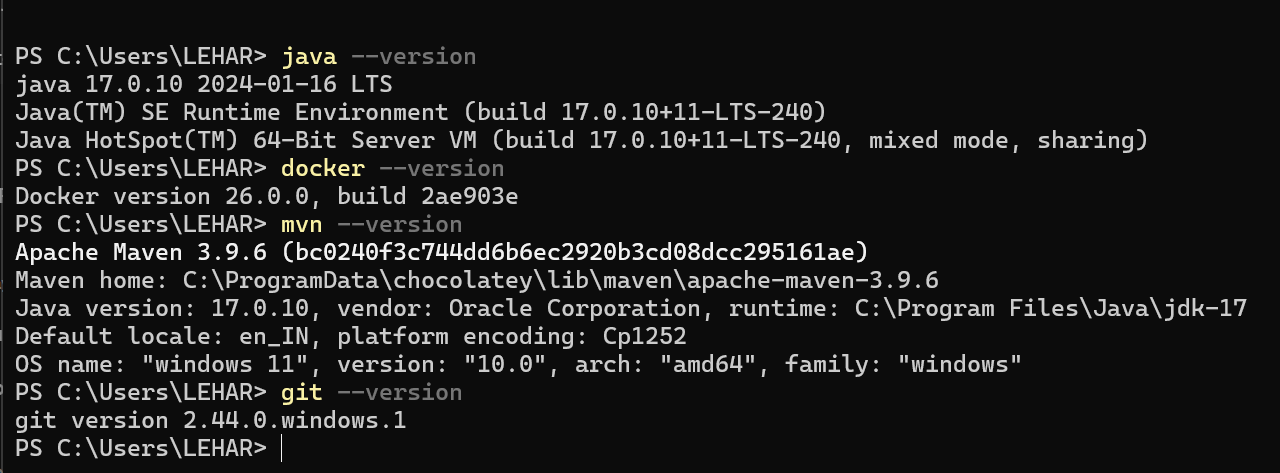
PUSH YOU DOCKER BUILD IMAGE IN NEXUS-REPOSITORY.

REQUIREMENTS:

* sonatypenexus
* jenkins
* maven
* git
* openjdk-17

(on your local system)

PRE-REQUISITE CHECK:



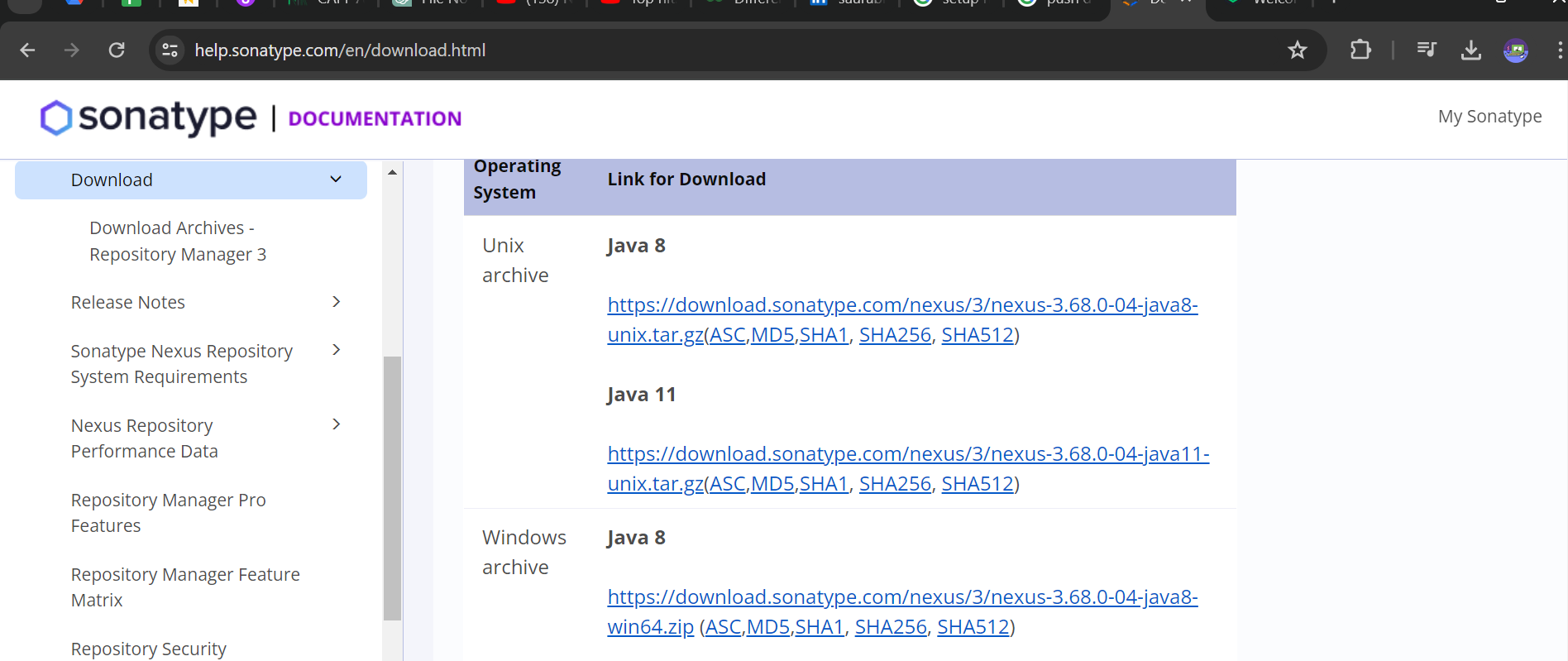
INSTALLATIONS:

SONATYPE-NEXUS

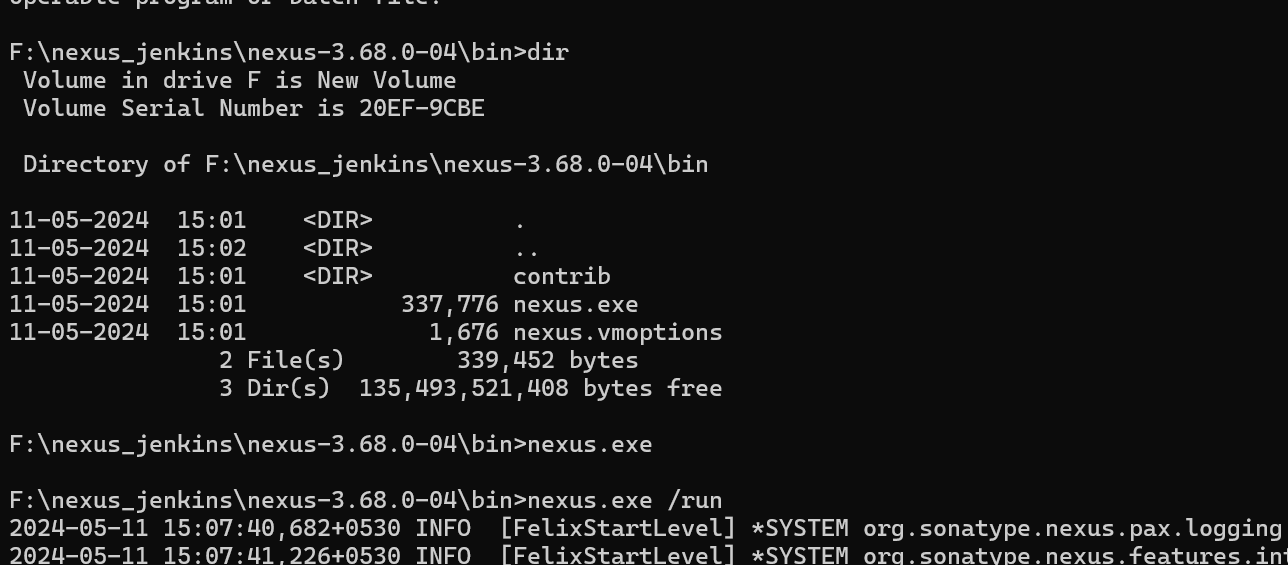
Download nexus in windows

<https://www.google.com/search?sca_esv=c45adb40d50f9228&sca_upv=1&sxsrf=ADLYWIJYmETyYaNYQ9eGzecnhxoP_TMI-g:1715418886477&q=setup+nexus+and+jenkins+on+windows&tbm=vid&source=lnms&prmd=visnbmtz&sa=X&ved=2ahUKEwjs-_mqoYWGAxVW1zgGHdfqDaYQ0pQJegQIDhAB&biw=1280&bih=585&dpr=1.5#fpstate=ive&vld=cid:3aba42e3,vid:TRI-GfkCNeE,st:0>  
  
according to this

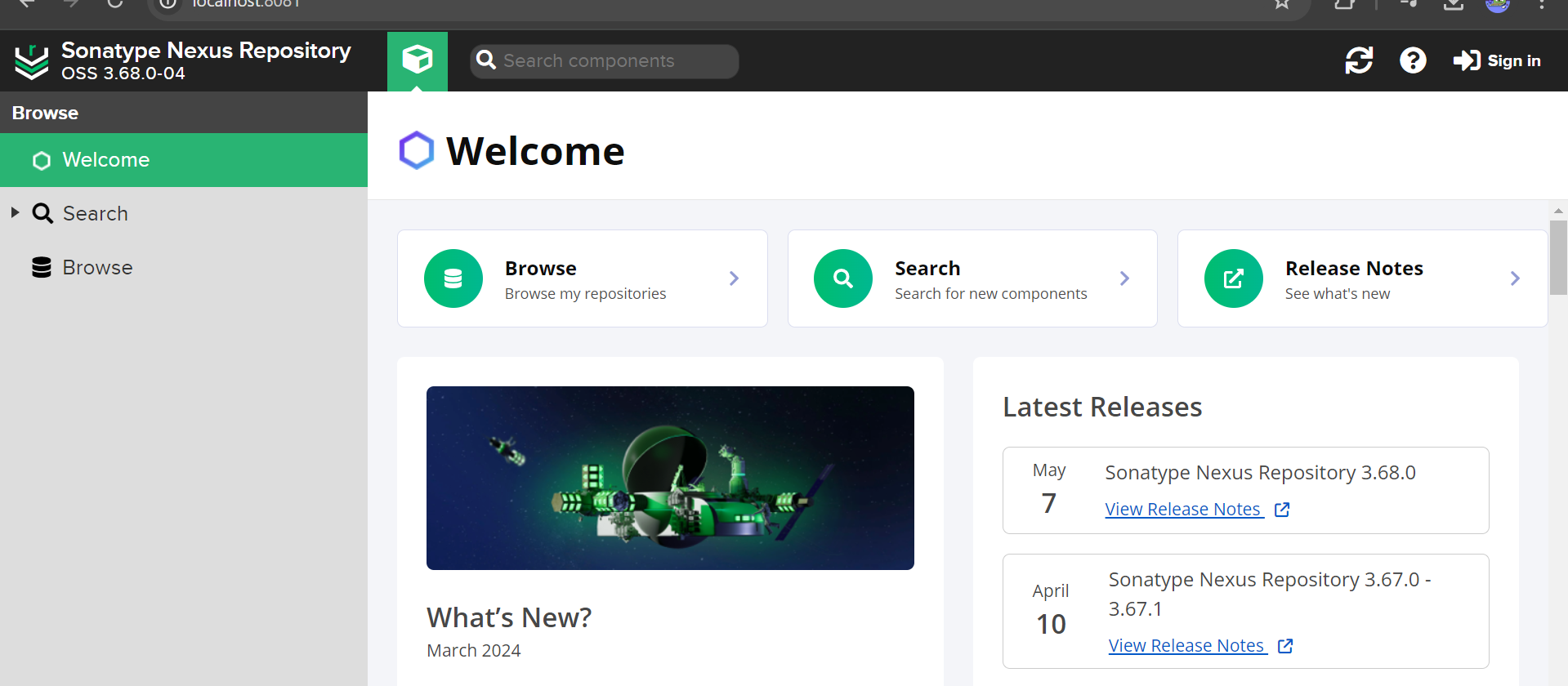
<https://help.sonatype.com/en/download.html>



Run in cmd



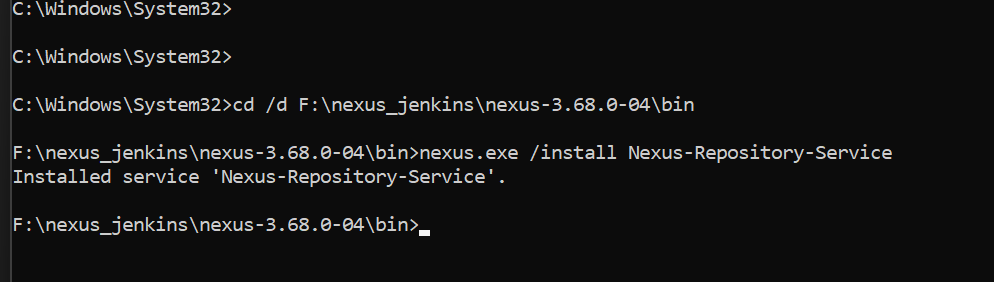
Then check on local host



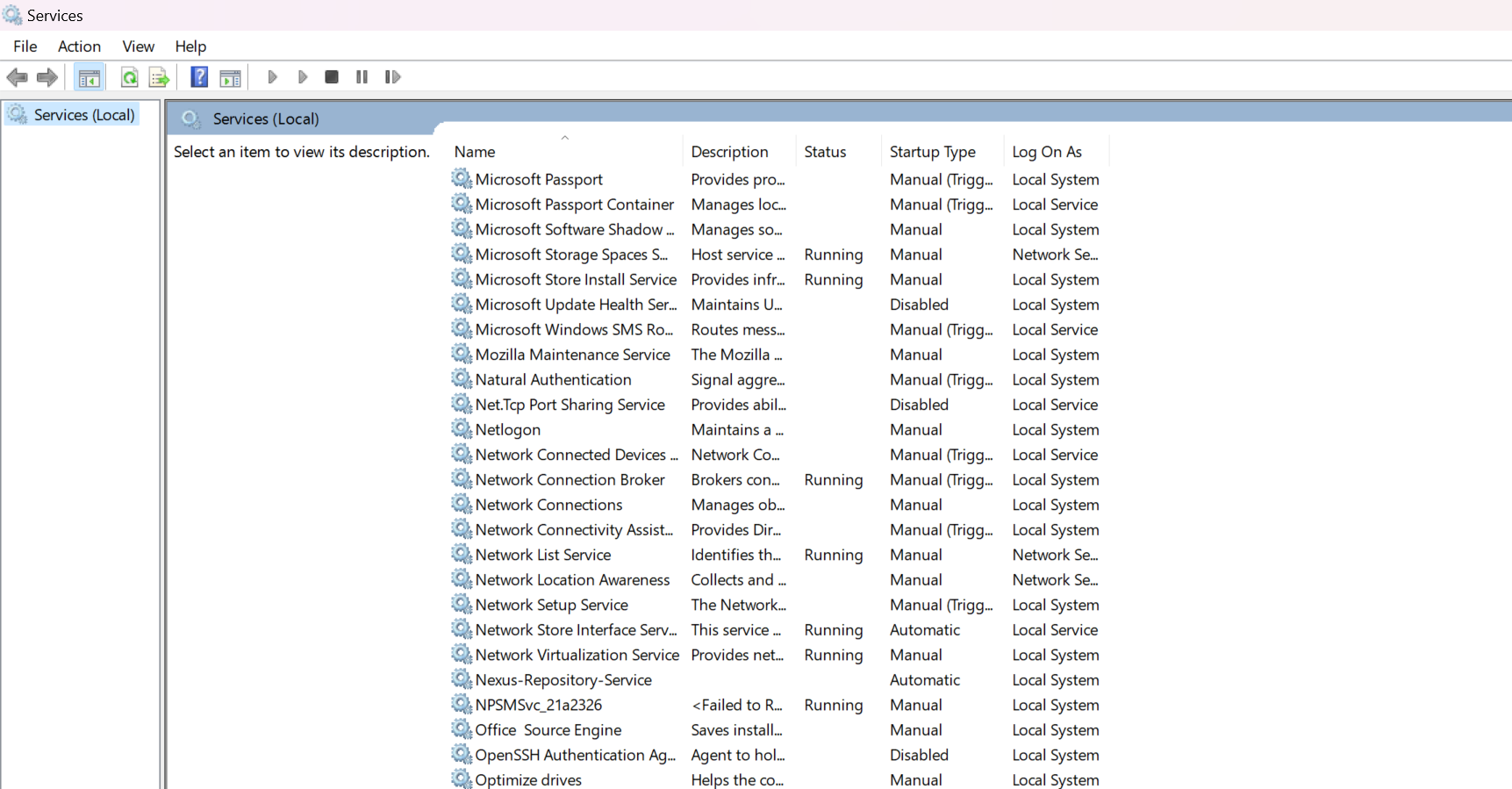
To install It in f drive us this cmds

FOR RUN ADMINISTRATOR CMD TO CHANGE DIRECTORY

cd /d F:\nexus\_jenkins\nexus-3.68.0-04\bin



Check nexus serivces in service YOU CAN RESTART JENKINS AND NEXUS SERVICES IN IT.

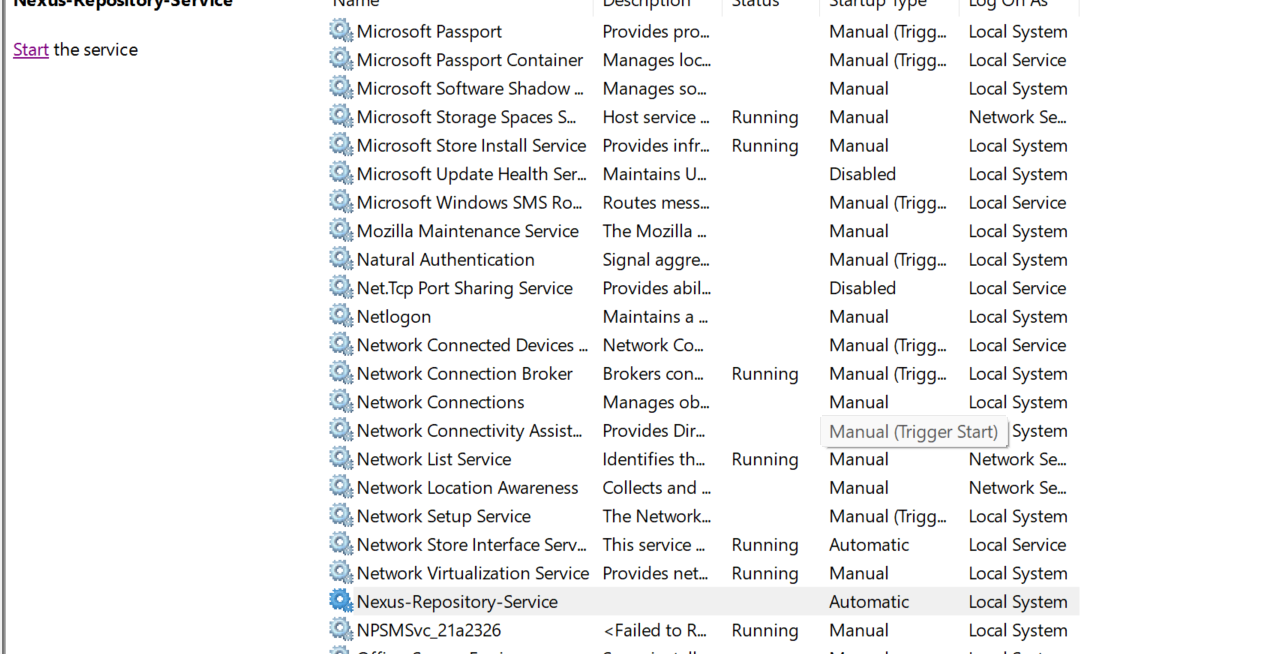


F:\nexus\_jenkins\nexus-3.68.0-04\bin>nexus.exe /install Nexus-Repository-Service

Installed service 'Nexus-Repository-Service'.

F:\nexus\_jenkins\nexus-3.68.0-04\bin>nexus.exe /start Nexus-Repository-Service

Starting service 'Nexus-Repository-Service'.



ABOVE PART DONE FOR THE ONCE YOU INSTALL THEN GO IN SERVICES AND RESTART STOP WITH IT

Sonatype password 680c4af3-cab9-4328-820e-a0828b3e61e5

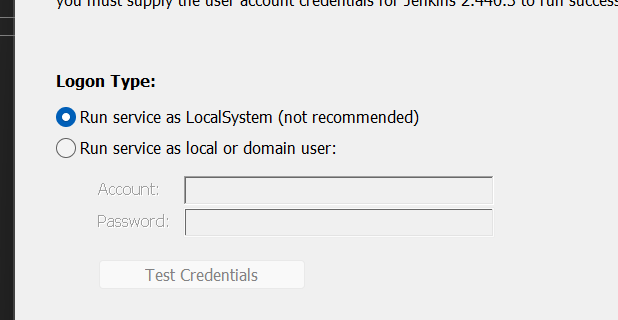
JENKINS

Download from windows jenkins

<https://www.jenkins.io/download/>

youtube video

<https://www.google.com/search?q=jenkins+in+windows+download&sca_esv=6e87d5184cf45ead&sca_upv=1&sxsrf=ADLYWIIUS9A_XUvwNvj3xV2sarc2HAyqCA%3A1715422243810&ei=I0Q_Zu2RMaWxseMPwY-YyA4&ved=0ahUKEwitr-3rrYWGAxWlWGwGHcEHBukQ4dUDCBA&uact=5&oq=jenkins+in+windows+download&gs_lp=Egxnd3Mtd2l6LXNlcnAiG2plbmtpbnMgaW4gd2luZG93cyBkb3dubG9hZDIGEAAYFhgeMgYQABgWGB4yBhAAGBYYHjILEAAYgAQYhgMYigUyCxAAGIAEGIYDGIoFMgsQABiABBiGAxiKBUjgOFCqDFi5OHABeACQAQCYAcwBoAGLF6oBBjAuMTQuMrgBA8gBAPgBAZgCC6AC_A7CAgoQABiwAxjWBBhHwgINEAAYgAQYsAMYQxiKBcICCBAAGKIEGIkFwgIHEAAYgAQYDcICCBAAGA0YHhgPwgIIEAAYBRgNGB7CAggQABgIGA0YHsICCBAAGBYYHhgPwgIIEAAYgAQYogSYAwDiAwUSATEgQIgGAZAGCpIHBTEuNy4zoAeFaQ&sclient=gws-wiz-serp#fpstate=ive&vld=cid:6d36aeb6,vid:Zdxko2bPAAw,st:0>



Using jenkins on docker

Used this commands

docker pull jenkins/jenkins:lts

to expose jenkins port

docker run -d -p 8080:8080 -p 50000:50000 --name jenkins --restart unless-stopped -v jenkins-data:/var/jenkins\_home jenkins/jenkins:lts

to get admin password

PS F:\nexus\_jenkins> docker exec $(docker ps -qf "name=jenkins") cat /var/jenkins\_home/secrets/initialAdminPassword

password

7c87a6c81f9447e7a595d446ef84e09e

BEFORE I HAVE SET JENKINS IN DOCKER

NOW BELOW JENKINS IS INSTALL ON MY SYSTEM BECAUSE IT SUPPORTS ONLY 17,21 VERSION AND IN MY SYSTEM THERE IS 22 JDK THEN I REINSTALL THE JAVA-17 TO SUPPORT JENKINS

JAVA

Java version if you want to download

<https://www.oracle.com/in/java/technologies/downloads/#jdk21-windows>

I have create a spring boot application

Its server port is 8082

And its maven build is done and jar file is create now create its image

Create docker image by creating Docekr file

<https://www.youtube.com/watch?v=RVIbMuNs1aw&t=720s>

FROM openjdk:11

ADD target/spring-boot-calling-external-api-0.0.1-SNAPSHOT.jar springboot-docker-demo.jar

ENTRYPOINT ["java","-jar","springboot-docker-demo.jar"]

4 docker build -t springboot-docker-demo:latest .

5 docker images

6 docker run -p 8082:8082 springboot-docker-demo

You have done maven bulit

Right click on pom file and maven build

Spring boot code

restApi Controller

package com.example.springbootdockernexus.controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class RestApiController {

@RequestMapping("/hello")

public String hello() {

return "Hello World";

}

@RequestMapping("/hell")

public String hell() {

return "Hello World";

}

}

Application.properties

spring.application.name=springboot-docker-nexus

server.port=8086

Dockerfile

# Use a base image with Java 17

FROM openjdk:17-jdk

# Set the working directory

WORKDIR /app

# Copy the compiled application JAR file into the container

COPY target/springboot-docker-nexus-0.0.1-SNAPSHOT.jar my-spring-boot-app.jar

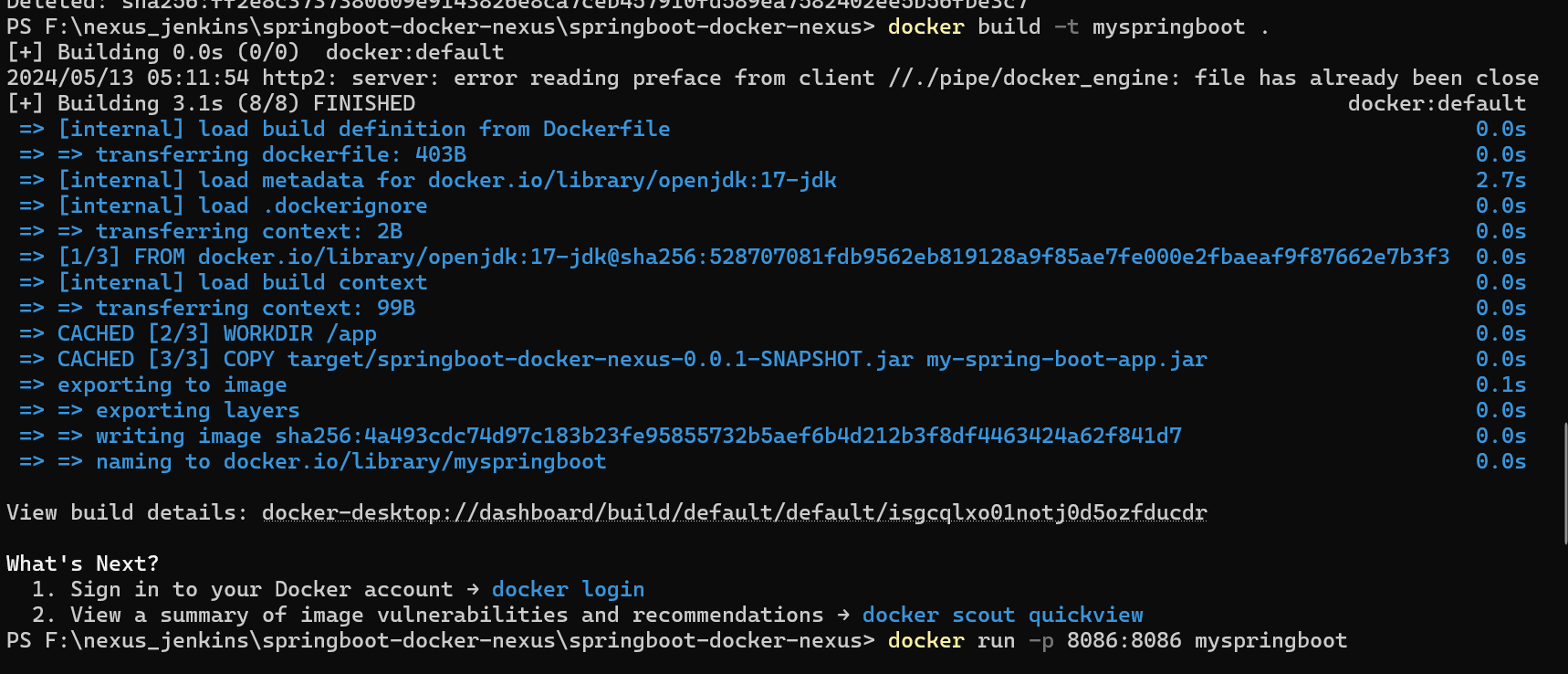
EXPOSE 8086

# Specify the command to run the Spring Boot application

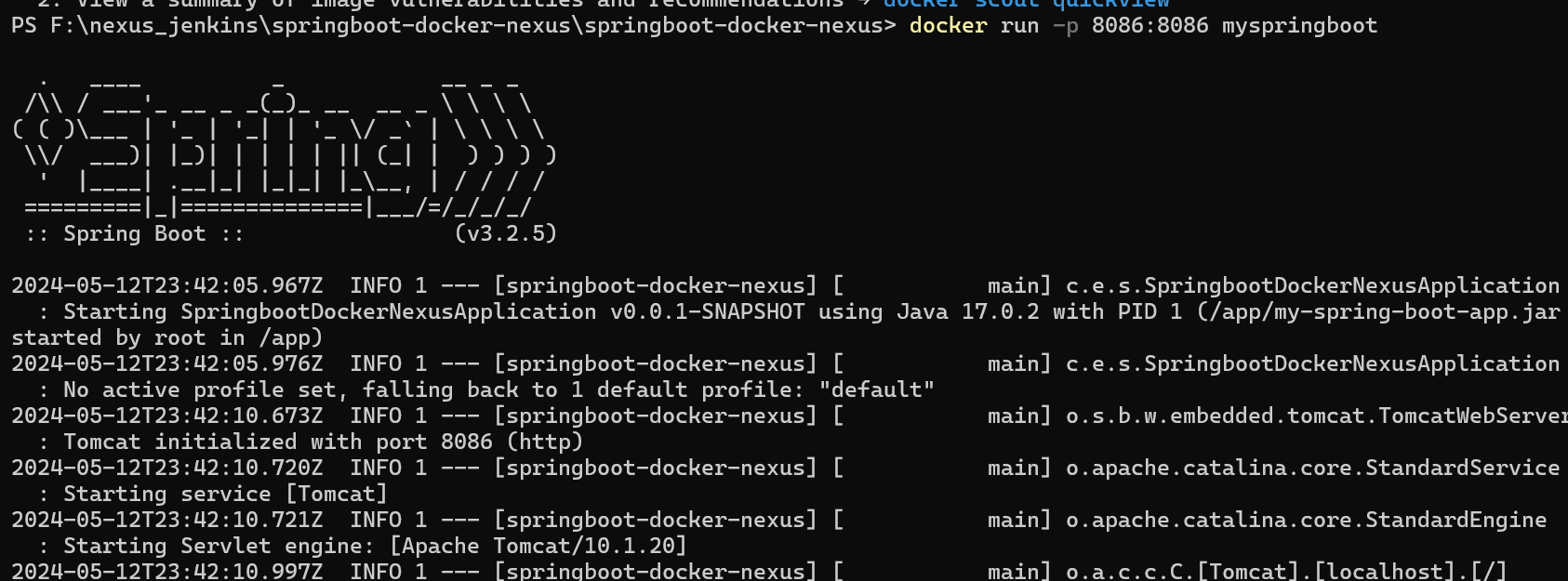
CMD ["java", "-jar", "my-spring-boot-app.jar"]

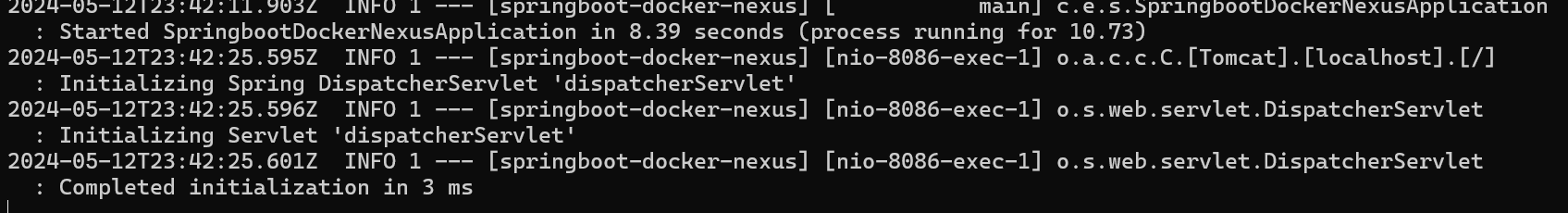
Image build

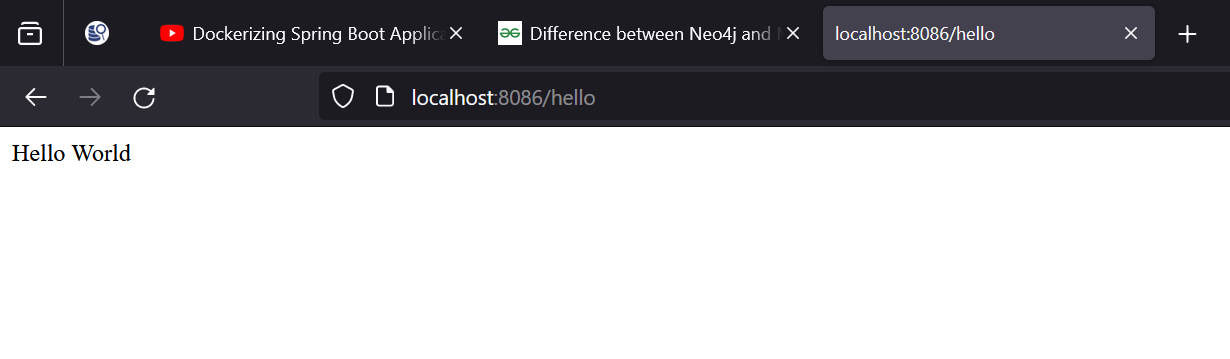
docker bulit -t myspringboot .



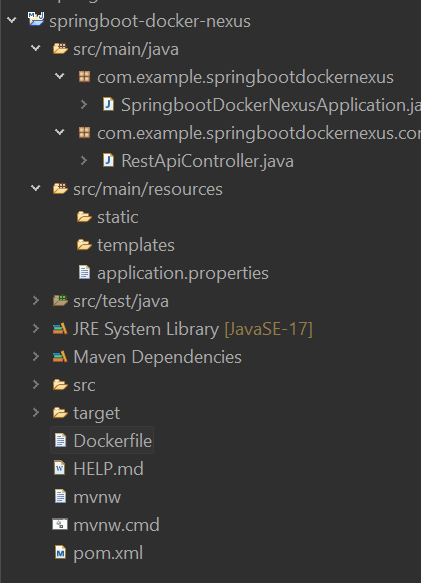
Run the container with exposing port 8086







It run on localhost:8096/hello

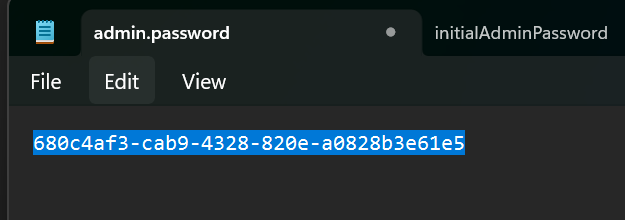
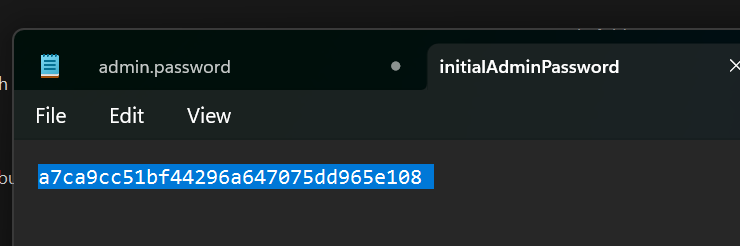


After setting up jenkins and sonatype AND SPRING BOOT APPLICATION RUN ON LOCAL

Jenkins admin pasword istalled on local a7ca9cc51bf44296a647075dd965e108

YOU CAN FILND BOTH SONATYPE AND JENKINS PASSWORD IN THE DIRECTORY WITH THIS NAME ALSO CHECK HIDDENFOLDER YOU CAN FIND PROGRAM DATA IN WHICH IT IS PRESENT

jenkins(login)

sonatype jenkins

now on sonatype

<https://gcore.com/learning/publishing-artifacts-to-nexus-using-jenkins-pipelines/>

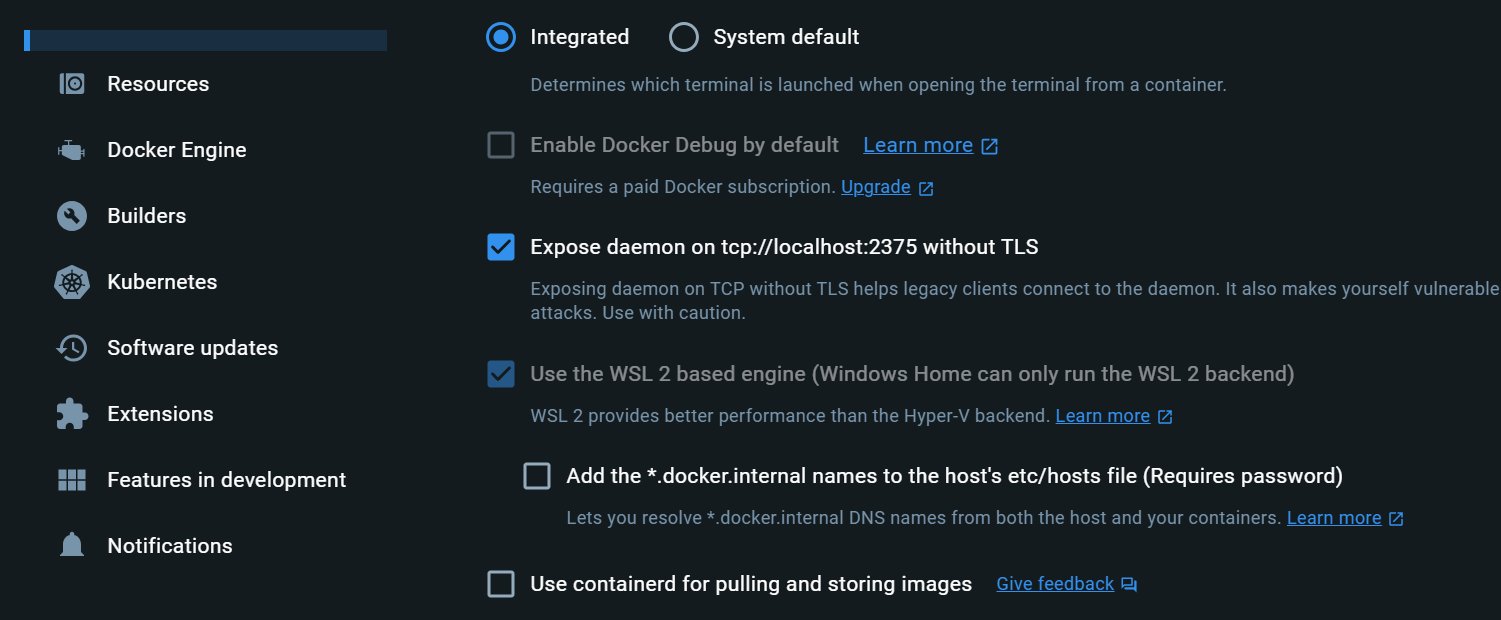
you can use this link

you create repository docker-hosted name

then you don’t need to create use these is already present admin and the password is admin respectively

add it to jenkins credentitials nexus-user-credentials

Create a repo docker hosted and enable it in docker desktop



Docker info before enable and after enabling

PS C:\Users\LEHAR> docker info

Client:

Version: 26.0.0

Context: default

Debug Mode: false

Path: C:\Program Files\Docker\cli-plugins\docker-sbom.exe

scout: Docker Scout (Docker Inc.)

Version: v1.6.3

Path: C:\Program Files\Docker\cli-plugins\docker-scout.exe

Debug Mode: false

HTTP Proxy: http.docker.internal:3128

HTTPS Proxy: http.docker.internal:3128

No Proxy: hubproxy.docker.internal

PS C:\Users\LEHAR> docker info

Client:

Version: 26.0.0

Context: default

Debug Mode: false

Plugins:

buildx: Docker Buildx (Docker Inc.)

Version: v0.13.1-desktop.1

Path: C:\Program Files\Docker\cli-plugins\docker-buildx.exe

compose: Docker Compose (Docker Inc.)

Version: v2.26.1-desktop.1

Path: C:\Program Files\Docker\cli-plugins\docker-compose.exe

debug: Get a shell into any image or container. (Docker Inc.)

Version: 0.0.27

PS C:\Users\LEHAR> docker info

Client:

Version: 26.0.0

Context: default

Debug Mode: false

Server:

Containers: 110

Running: 2

CPUs: 8

Total Memory: 3.759GiB

Name: docker-desktop

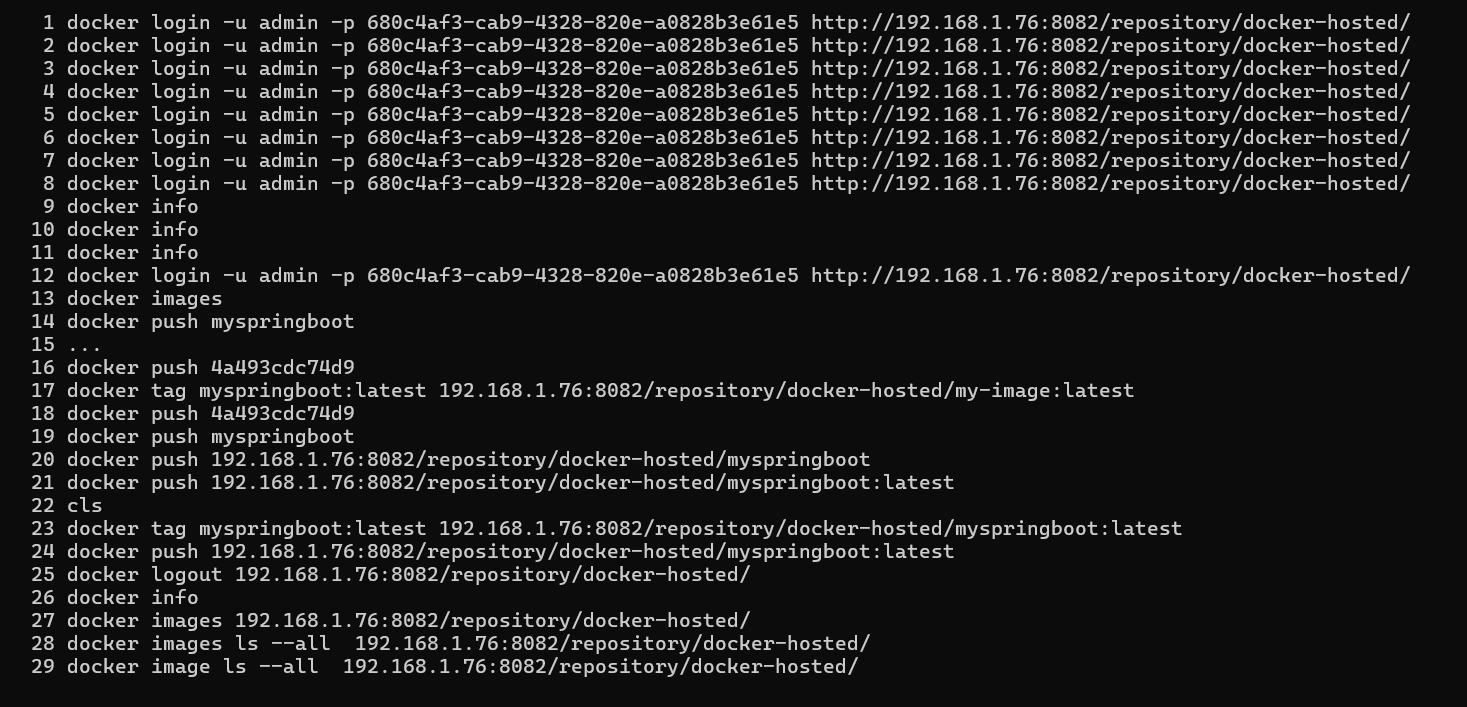
ID: 15eb90c5-a274-46c2-9777-a4cdb8b0d989

MANUALLY PUSH IN DOCKER HOSTED REPO IN SONATYPE

PS C:\Users\LEHAR> docker login -u admin -p 680c4af3-cab9-4328-820e-a0828b3e61e5 http://192.168.1.76:8082/repository/docker-hosted/

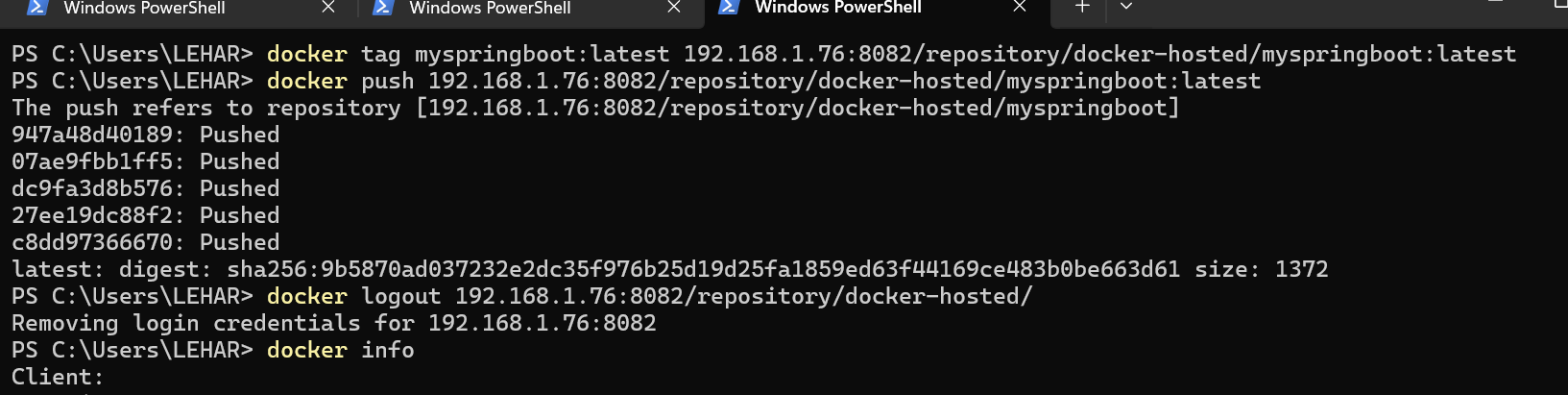
WARNING! Using --password via the CLI is insecure. Use --password-stdin.

Login Succeeded

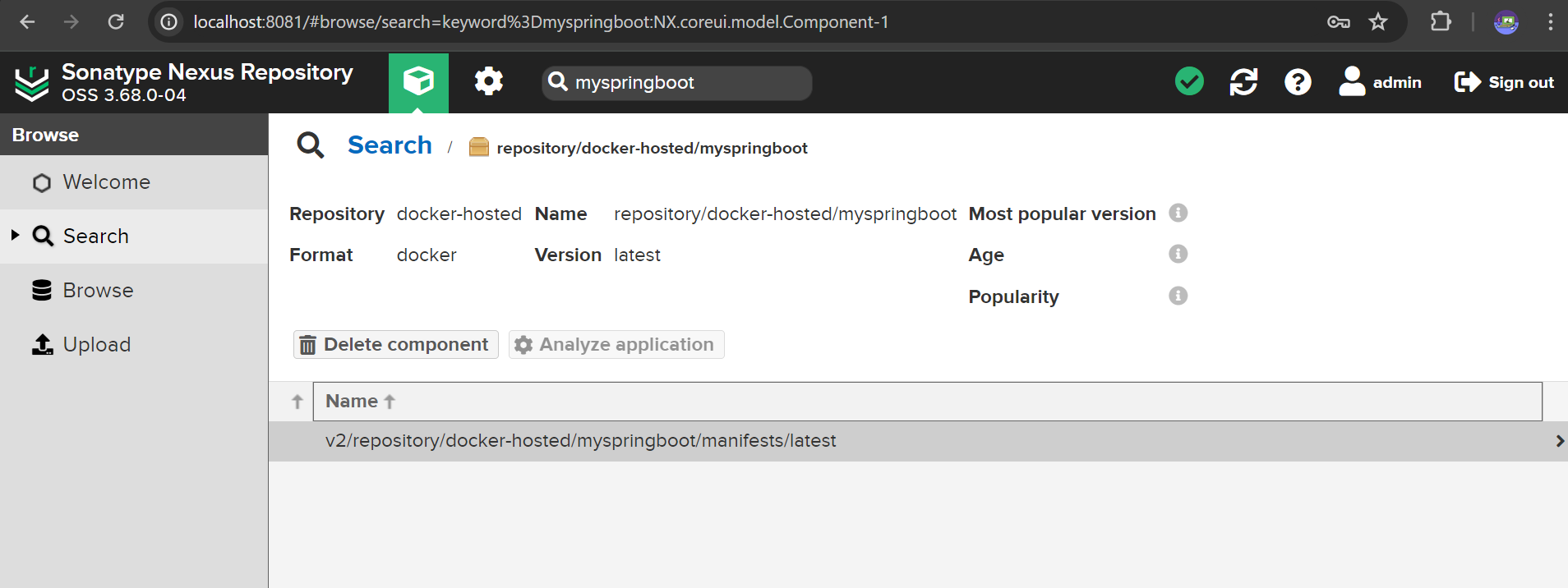


cmds

Then add tag then push image get push add:latest or what you like is required



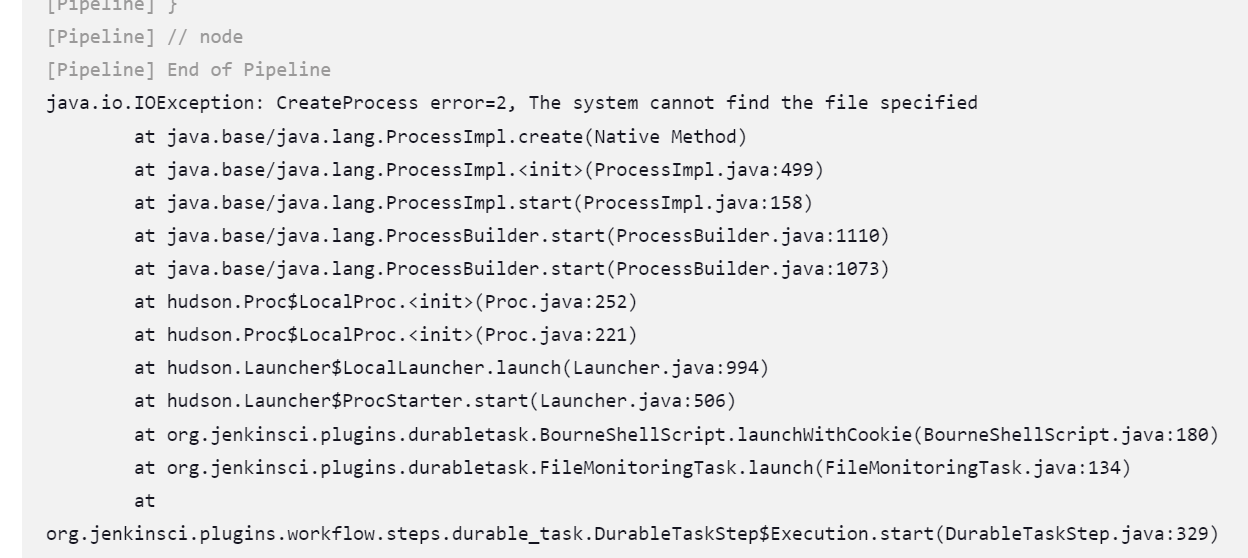
YOU CAN GO TO DOCKER-HOSTED THEIRE IMAGE IS PRESENT



Now add git commands on powershell

Add maven build in powershell

When your maven is not installed error occurred



Install maven on command prompt with run as administrator

////

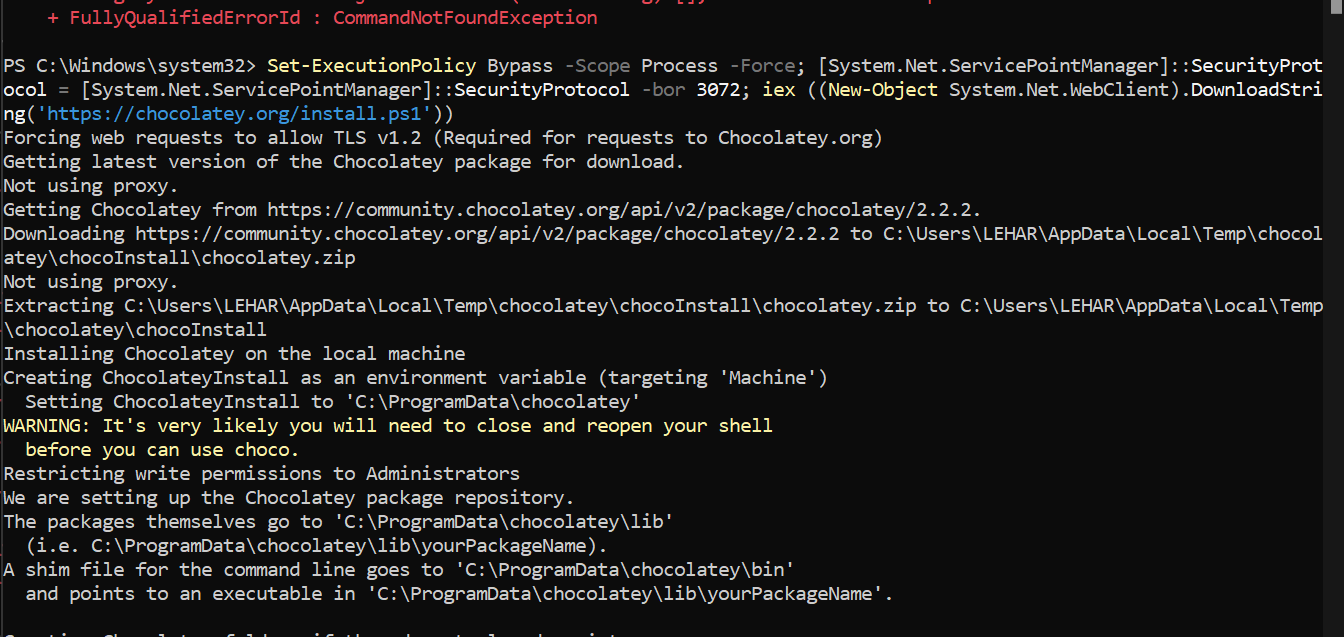
PS C:\Windows\system32> Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1')) Forcing web requests to allow TLS v1.2 (Required for requests to Chocolatey.org) Getting latest version of the Chocolatey package for download. Not using proxy. Getting Chocolatey from https://community.chocolatey.org/api/v2/package/chocolatey/2.2.2. Downloading https://community.chocolatey.org/api/v2/package/chocolatey/2.2.2 to

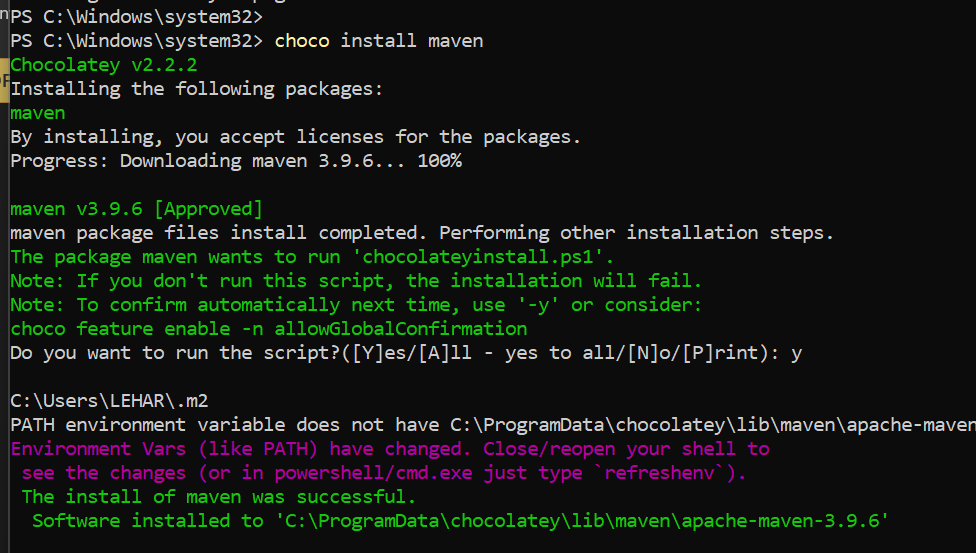
C:\Users\LEHAR\AppData\Local\Temp\chocolatey\chocoInstall\chocolatey.zip Not using proxy. Extracting C:\Users\LEHAR\AppData\Local\Temp\chocolatey\chocoInstall\chocolatey.zip to C:\Users\LEHAR\AppData\Local\Temp\chocolatey\chocoInstall Installing Chocolatey on the local machine Creating ChocolateyInstall as an environment variable (targeting 'Machine') Setting ChocolateyInstall to 'C:\ProgramData\chocolatey' WARNING: It's very likely you will need to close and reopen your shell before you can use choco. Restricting write permissions to Administrators We are setting up the Chocolatey package repository. The packages themselves go to 'C:\ProgramData\chocolatey\lib' (i.e. C:\ProgramData\chocolatey\lib\yourPackageName). A shim file for the command line goes to 'C:\ProgramData\chocolatey\bin' and points to an executable in 'C:\ProgramData\chocolatey\lib\yourPackageName'. Creating Chocolatey folders if they do not already exist. chocolatey.nupkg file not installed in lib. Attempting to locate it from bootstrapper. PATH environment variable does not have C:\ProgramData\chocolatey\bin in it. Adding... Adding Chocolatey to the profile. This will provide tab completion, refreshenv, etc. WARNING: Chocolatey profile installed. Reload your profile - type . $profile Chocolatey (choco.exe) is now ready. You can call choco from anywhere, command line or powershell by typing choco. Run choco /? for a list of functions. You may need to shut down and restart powershell and/or consoles first prior to using choco. Ensuring Chocolatey commands are on the path Ensuring chocolatey.nupkg is in the lib folder

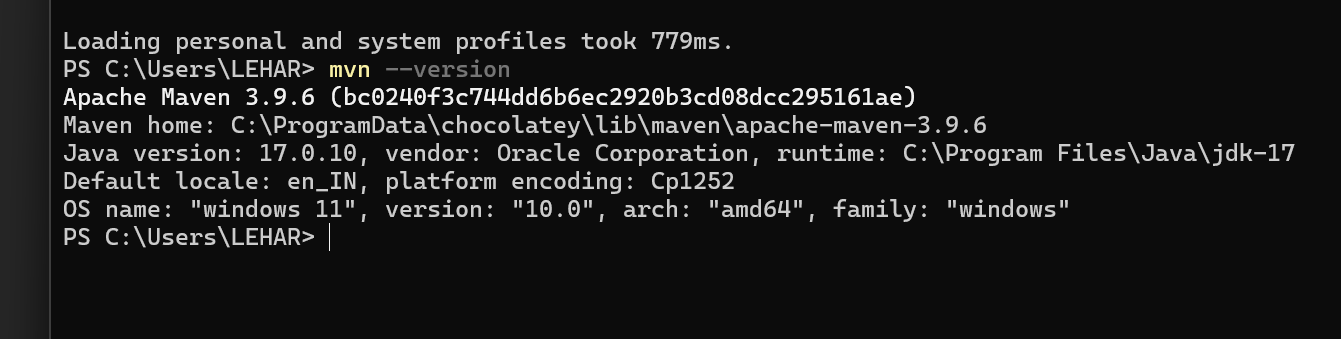
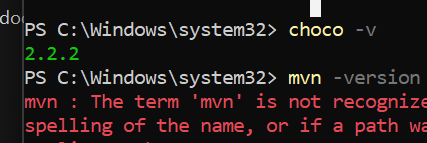
PS C:\Windows\system32> choco install maven Chocolatey v2.2.2 Installing the following packages: maven By installing, you accept licenses for the packages. Progress: Downloading maven 3.9.6... 100% maven v3.9.6 [Approved] maven package files install completed. Performing other installation steps. The package maven wants to run 'chocolateyinstall.ps1'. Note: If you don't run this script, the installation will fail. Note: To confirm automatically next time, use '-y' or consider: choco feature enable -n allowGlobalConfirmation Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): y C:\Users\LEHAR\.m2 PATH environment variable does not have C:\ProgramData\chocolatey\lib\maven\apache-maven-3.9.6\bin in it. Adding... Environment Vars (like PATH) have changed. Close/reopen your shell to see the changes (or in powershell/cmd.exe just type `refreshenv`). The install of maven was successful. Software installed to 'C:\ProgramData\chocolatey\lib\maven\apache-maven-3.9.6' Chocolatey installed 1/1 packages. See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log). PS C:\Windows\system32> choco -v 2.2.2

AFTER INSTALLING OPEN POWERSHELL THEN USE THIS COMMAND PS C:\Windows\system32> maven --version

//////////





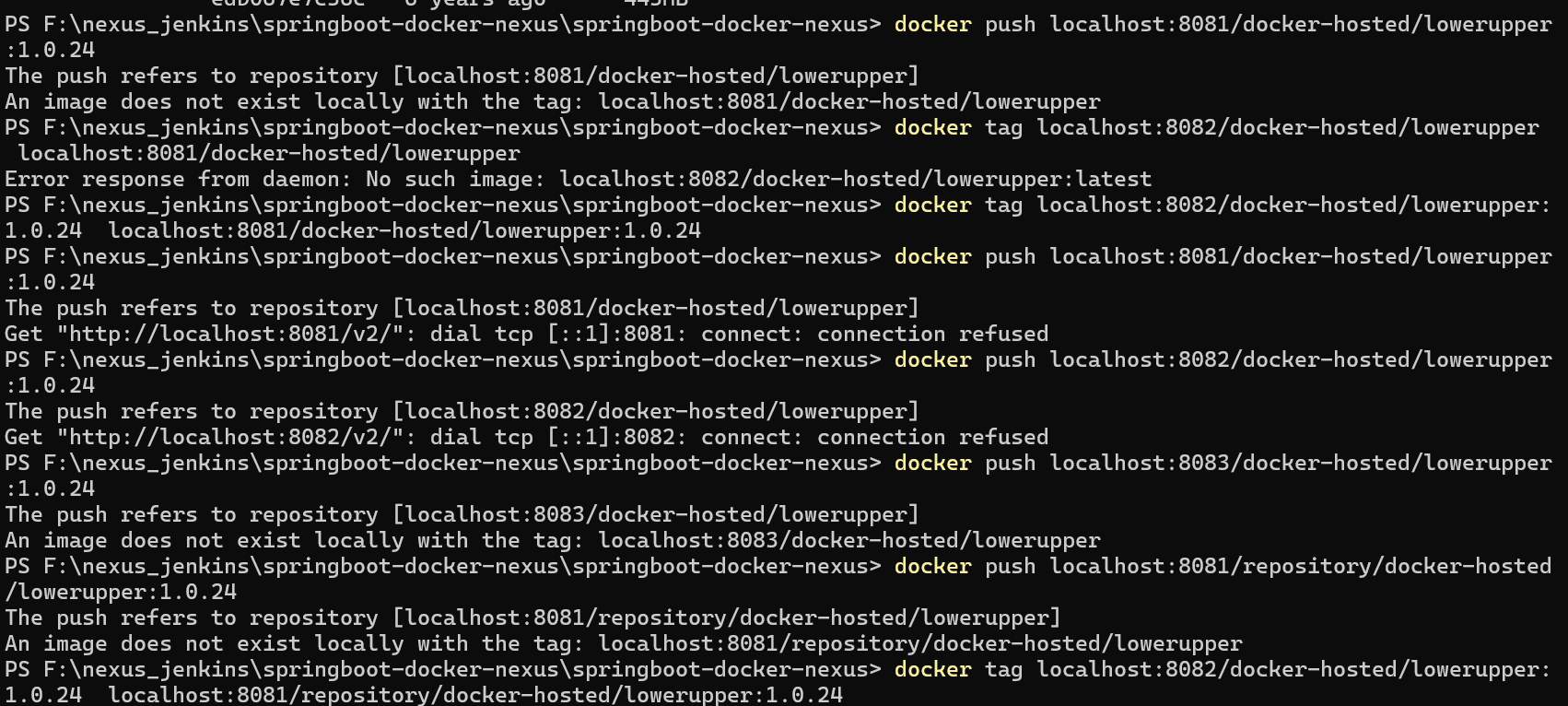


After maven build

After docker build image

commands

docker login -u admin -p 680c4af3-cab9-4328-820e-a0828b3e61e5 <http://192.168.1.76:8082/repository/docker-hosted/>



Using below commands

ERROR IS USED TO BE CONNECTION REFUSED BEAUSE IT CAN’T BE CONNECTED TO localhost USE YOUR OWN IP ADDRESS

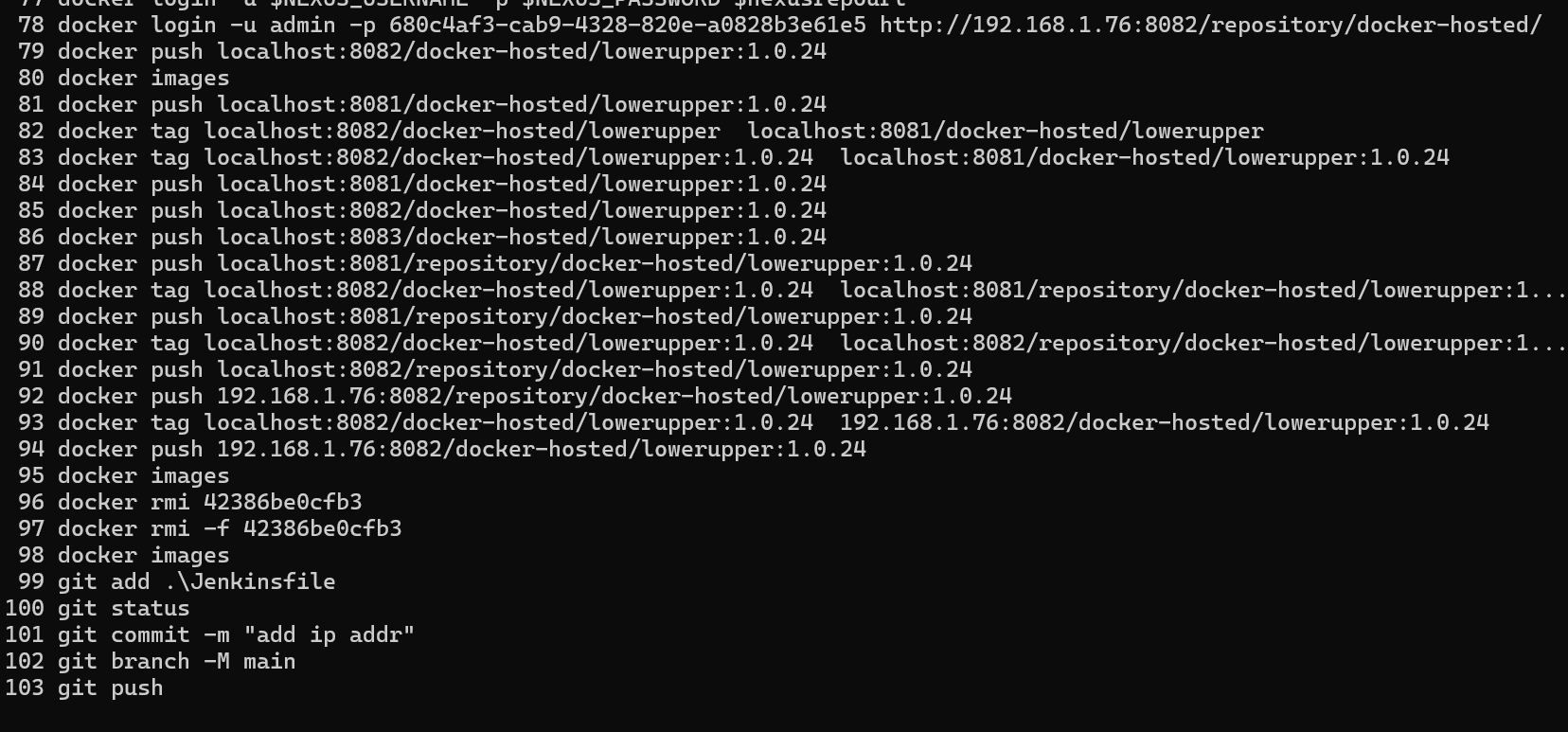
MANUALLY TRY EVERY COMMAND RUNNING ON POWESHELL OR NOT

add tag and push in sonatype url

92 docker push 192.168.1.76:8082/repository/docker-hosted/lowerupper:1.0.24

93 docker tag localhost:8082/docker-hosted/lowerupper:1.0.24 192.168.1.76:8082/docker-hosted/lowerupper:1.0.24

94 docker push 192.168.1.76:8082/docker-hosted/lowerupper:1.0.24



AFTER MANUALLY UPLOAD ON GIT AND BUILD IT WITH JENKINS PIPELINE

Git\_jenkins(jenkins pipline name)

JENKINS CODE WITH USING GIT REPOSITORY POLLING WITH SCM

pipeline {

agent any

environment {

imageName = "lowerupper"

imageTag = "1.0.${env.BUILD\_NUMBER}"

nexusUrl = "192.168.1.76:8082"

nexusRepository = "docker-hosted"

nexusrepourl = "http://192.168.1.76:8082/repository/docker-hosted/"

}

stages {

stage('Get Code') {

steps {

git branch: 'main', url: 'https://github.com/Lehar1107/nexus\_jenkins\_docker\_springboot.git'

}

}

stage('Build') {

steps {

bat 'mvn clean install -DskipTests=true'

}

}

stage('Build Docker Image') {

steps {

script {

def dockerfile = """

FROM openjdk:17-jdk

COPY target/springboot-docker-nexus-0.0.1-SNAPSHOT.jar /app/my-spring-boot-app.jar

WORKDIR /app

ENTRYPOINT ["java", "-jar", "my-spring-boot-app.jar"]

"""

writeFile file: 'Dockerfile', text: dockerfile

bat "docker build -t ${imageName}:${imageTag} ."

bat "docker image ls ${imageName}:${imageTag}"

}

}

}

stage('Push Docker Image to Nexus') {

environment {

NEXUS\_CREDENTIALS = credentials('nexus-user-credentials')

}

steps {

script {

def dockerImage = "${imageName}:${imageTag}"

def nexusImage = "${nexusUrl}/${nexusRepository}/${imageName}:${imageTag}"

//def nexusUsername = NEXUS\_CREDENTIALS?.username

//def nexusPassword = NEXUS\_CREDENTIALS?.password

//echo "Nexus Username: ${nexusUsername}"

//echo "Nexus Password: ${nexusPassword}"

withCredentials([usernamePassword(credentialsId: 'nexus-user-credentials', usernameVariable: 'NEXUS\_USERNAME', passwordVariable: 'NEXUS\_PASSWORD')]) {

withEnv(["DOCKER\_LOGIN=\${NEXUS\_USERNAME}", "DOCKER\_PASSWORD=\${NEXUS\_PASSWORD}"]) {

// echo "DOCKER\_USERNAME: ${env.NEXUS\_USERNAME}"

// echo "DOCKER\_PASSWORD: ${env.NEXUS\_PASSWORD}"

// sh 'docker login -u ${env.NEXUS\_USERNAME} -p ${env.NEXUS\_PASSWORD} ${nexusUrl}'

bat "docker login -u $NEXUS\_USERNAME -p $NEXUS\_PASSWORD $nexusrepourl"

bat "docker tag ${dockerImage} ${nexusImage}"

bat "docker push ${nexusImage}"

bat "docker logout $nexusrepourl"

}

}

}

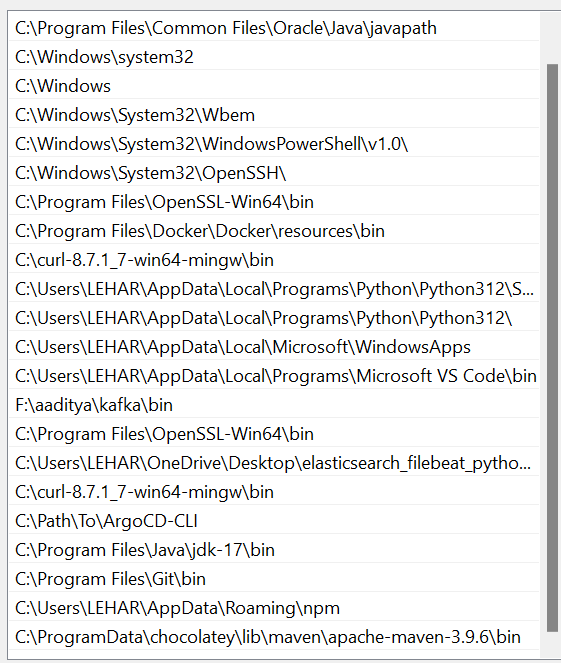
}

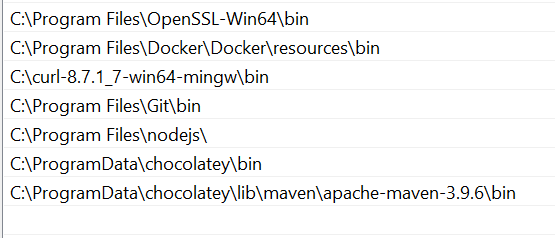
}

}

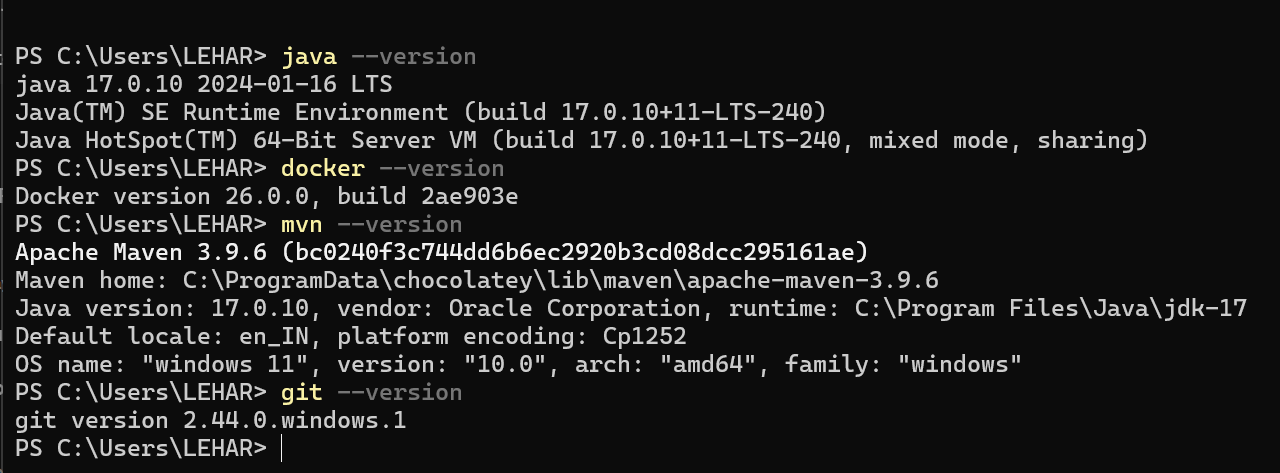
}

Ensure these paths git,maven,docker,jdk



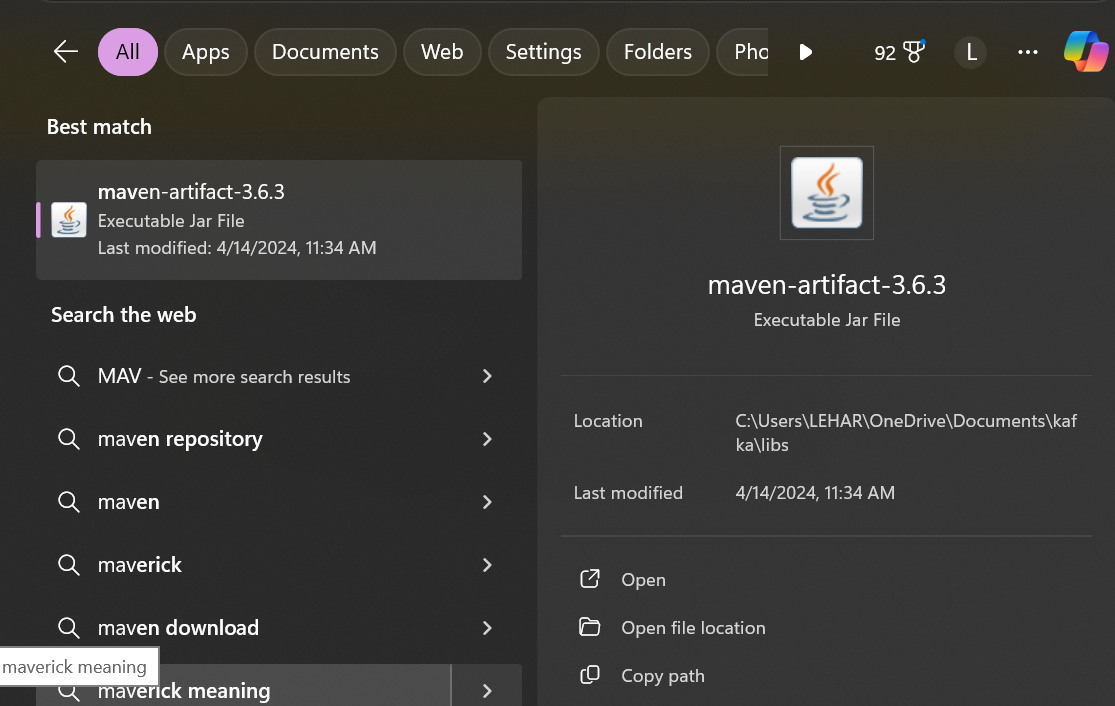


Make sure to check these versions running on powershell

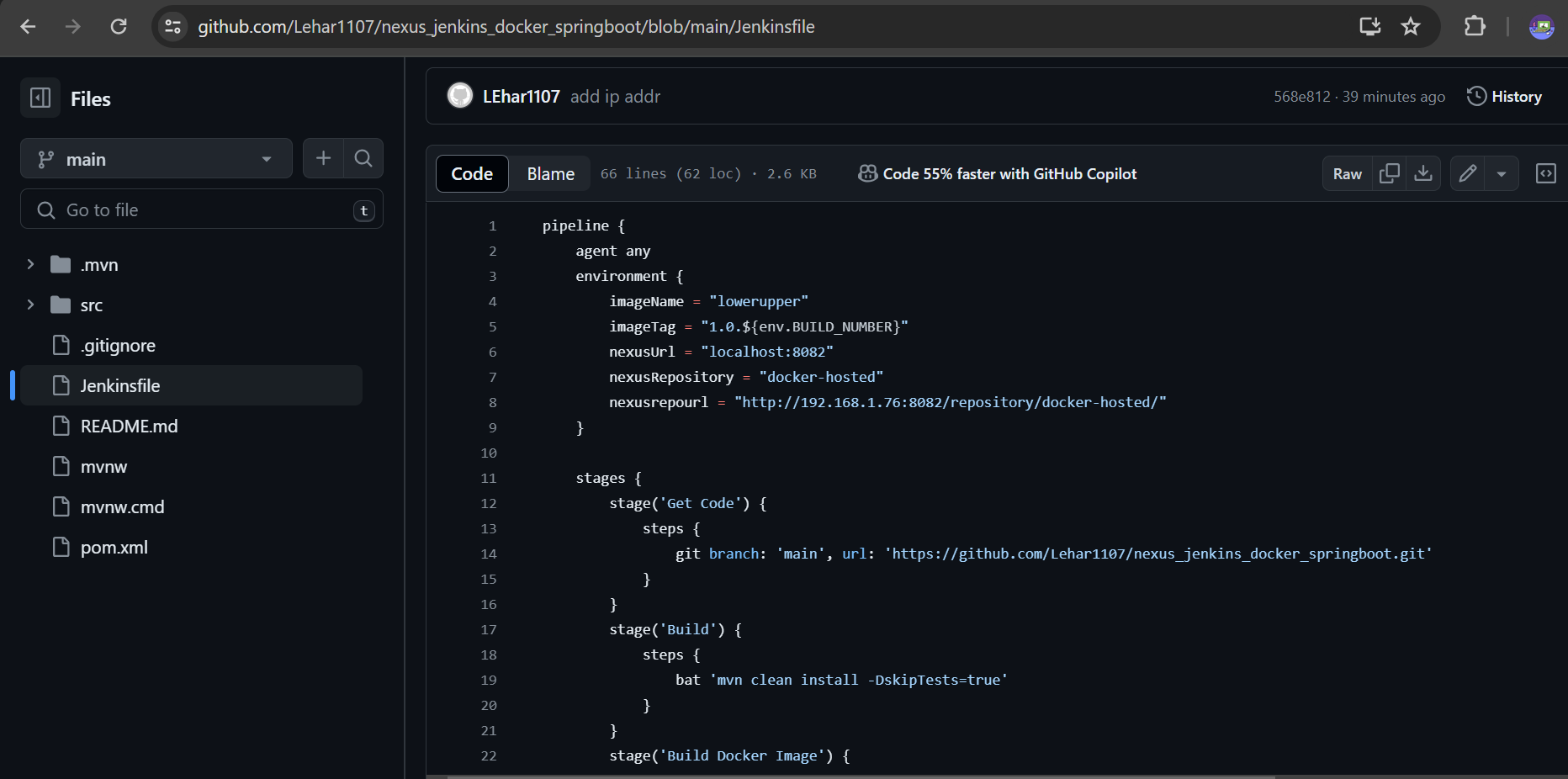


EVERY SETUP IS ON WINDOWS ENSURE THAT YOUR COMMADS WILL RUN ON POWESHELL

AS YOU R USINH POWESHELL SH WILL NOT WORK USE BAT AS YOU CAN SEE IN THE ABOVE CODE

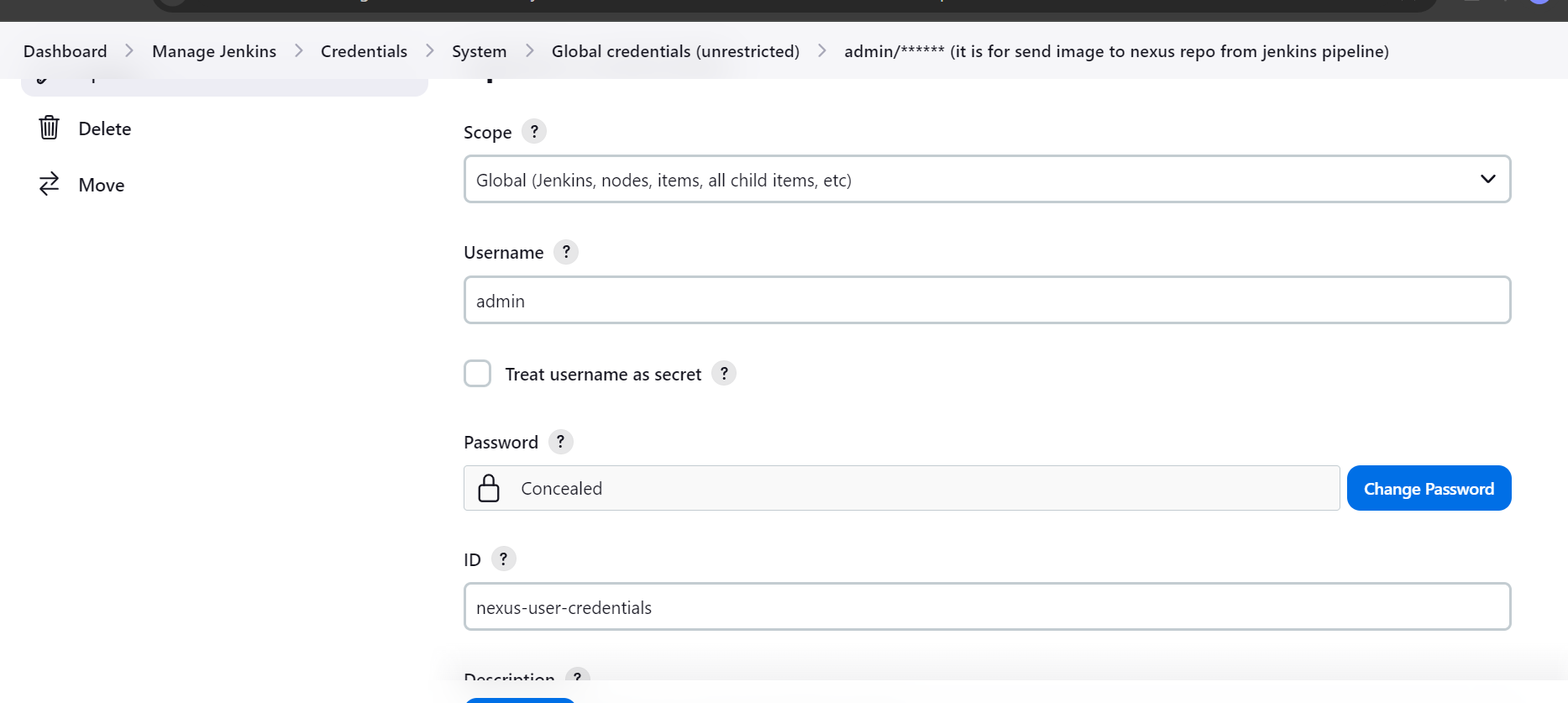


GIT HUB REPO

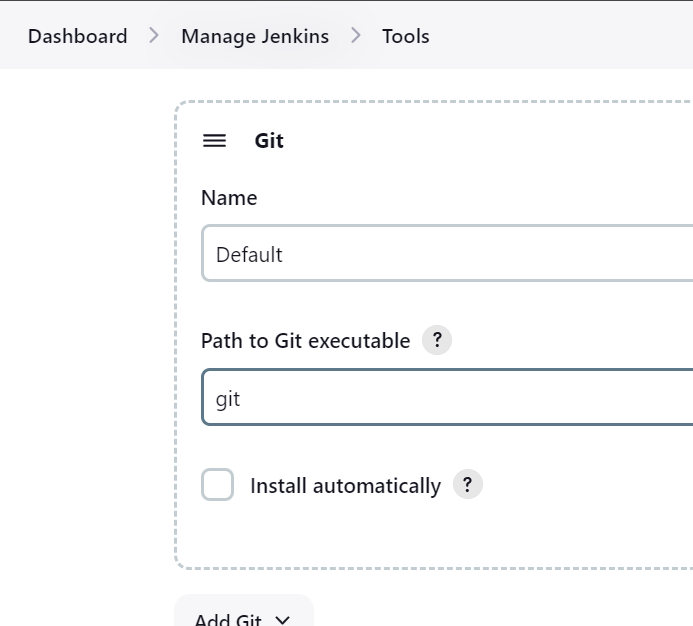


JENKINS FILE AND SPRING BOOT APPLICATION PRESENT ON YOU GIT REPO

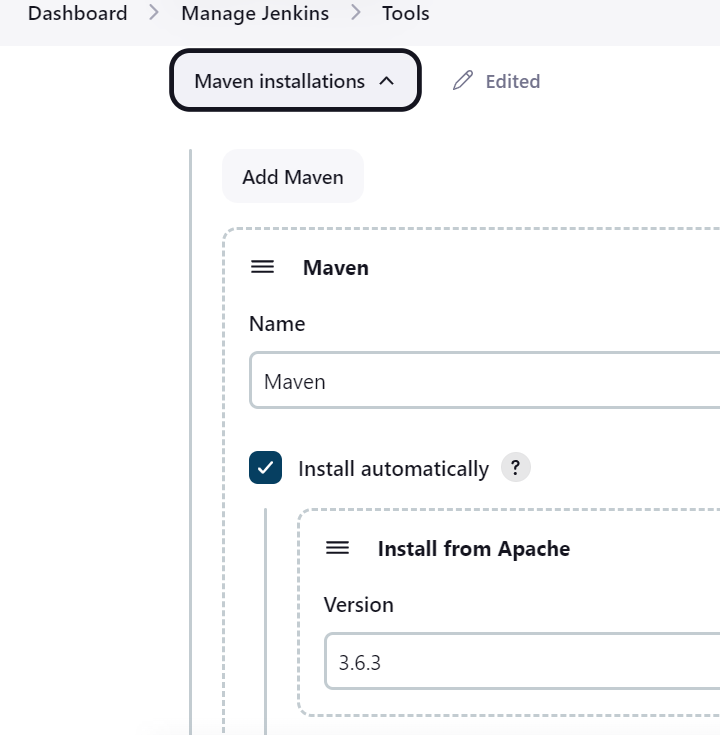
SONATYPE CREDENTIALS ADD ON JENKINS

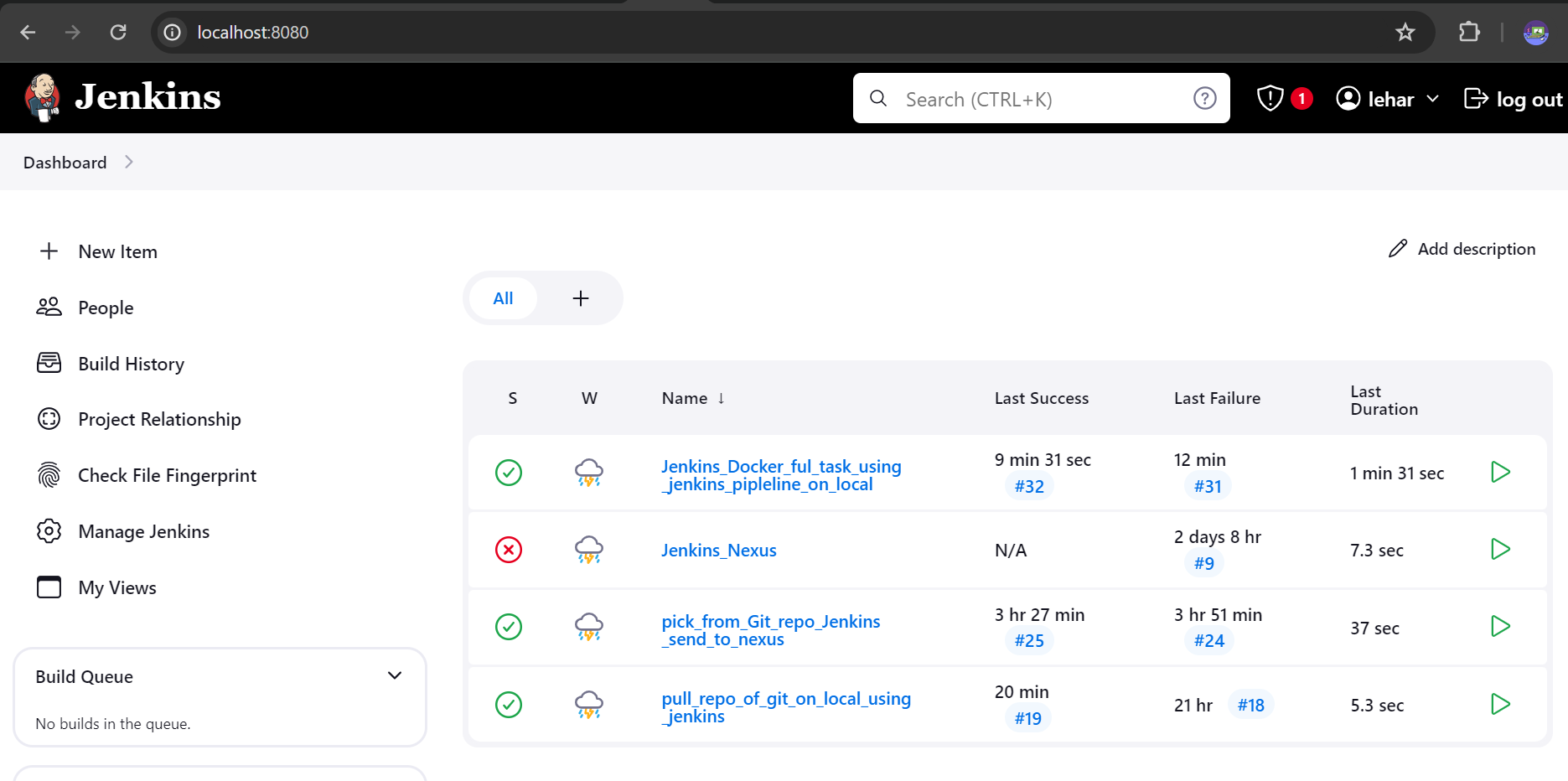


NO CHANGES DO IN GLOB TOOL CONFIGURATION

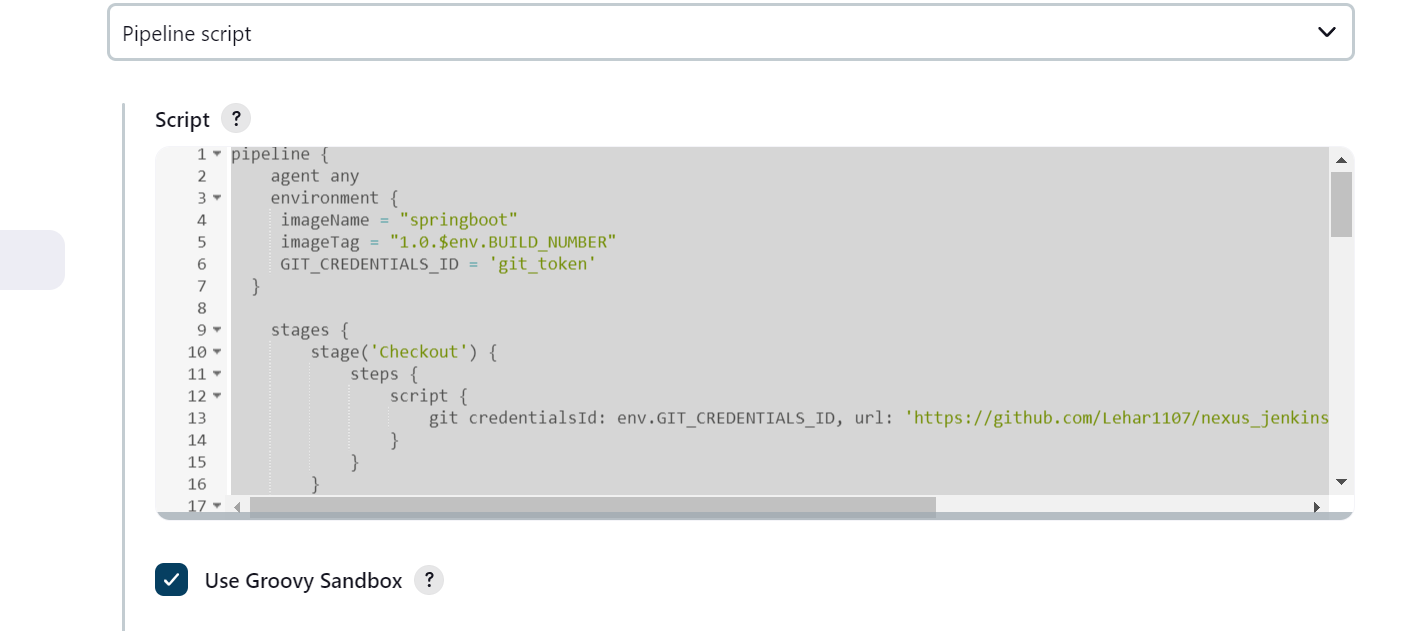


I HAVE SET THE MAVEN PLUGIN BY DEFAULT IS IS ALSO PRESENT





INSTEAD OF GIT REPO USE JENKINGS CONFIG



PREQUESITES WAS THE ABOVE

pipeline {

agent any

environment {

imageName = "springboot"

imageTag = "1.0.$env.BUILD\_NUMBER"

GIT\_CREDENTIALS\_ID = 'git\_token'

}

stages {

stage('Checkout') {

steps {

script {

git credentialsId: env.GIT\_CREDENTIALS\_ID, url: 'https://github.com/Lehar1107/nexus\_jenkins\_docker\_springboot.git', branch: 'main'

}

}

}

stage('Build') {

steps {

bat 'mvn clean install -DskipTests=true'

}

}

stage('Build Docker Image') {

steps {

script {

// def imageName = "lowerupper"

// def imageTag = "latest"

def dockerfile = """

FROM openjdk:17-jdk

COPY target/springboot-docker-nexus-0.0.1-SNAPSHOT.jar springboot.jar

WORKDIR /app

CMD ["java", "-jar", "springboot.jar"]

"""

writeFile file: 'Dockerfile', text: dockerfile

bat "docker build -t ${imageName}:${imageTag} ."

bat "docker image ls ${imageName}:${imageTag}"

}

}

}

stage('Push Docker Image to Nexus') {

environment {

NEXUS\_CREDENTIALS = credentials('nexus-user-credentials')

}

steps {

script {

def nexusUrl = "192.168.1.76:8082"

def nexusRepository = "docker-hosted"

def nexusrepourl = "http://192.168.1.76:8082/repository/docker-hosted/"

def dockerImage = "${imageName}:${imageTag}"

def nexusImage = "${nexusUrl}/${nexusRepository}/${imageName}:${imageTag}"

withCredentials([usernamePassword(credentialsId: 'nexus-user-credentials', usernameVariable: 'NEXUS\_USERNAME', passwordVariable: 'NEXUS\_PASSWORD')]) {

withEnv(["DOCKER\_LOGIN=\${NEXUS\_USERNAME}", "DOCKER\_PASSWORD=\${NEXUS\_PASSWORD}"]) {

// echo "DOCKER\_USERNAME: ${env.NEXUS\_USERNAME}"

// echo "DOCKER\_PASSWORD: ${env.NEXUS\_PASSWORD}"

// sh 'docker login -u ${env.NEXUS\_USERNAME} -p ${env.NEXUS\_PASSWORD} ${nexusUrl}'

bat "docker login -u $NEXUS\_USERNAME -p $NEXUS\_PASSWORD $nexusrepourl"

bat "docker tag ${dockerImage} ${nexusImage}"

bat "docker push ${nexusImage}"

bat "docker logout $nexusrepourl"

}

}

}

}

}

}

}

