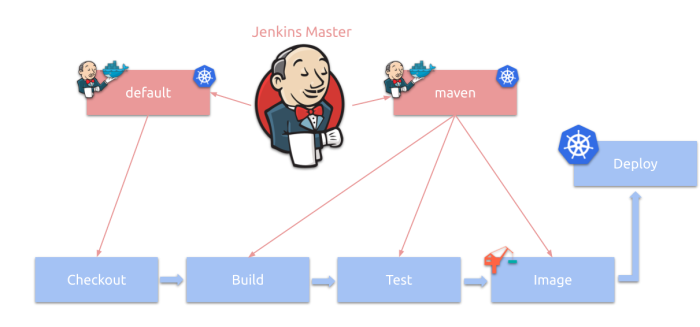
Jenkins and Docker



Installation Process jenkins:-

Download jenkins using this official side <https://www.jenkins.io/download/>

Installing process flow this link

<https://www.jenkins.io/doc/book/installing/windows/>

Default port :8080

Installation Process Docker :-

Download Docker exe file in the official Link

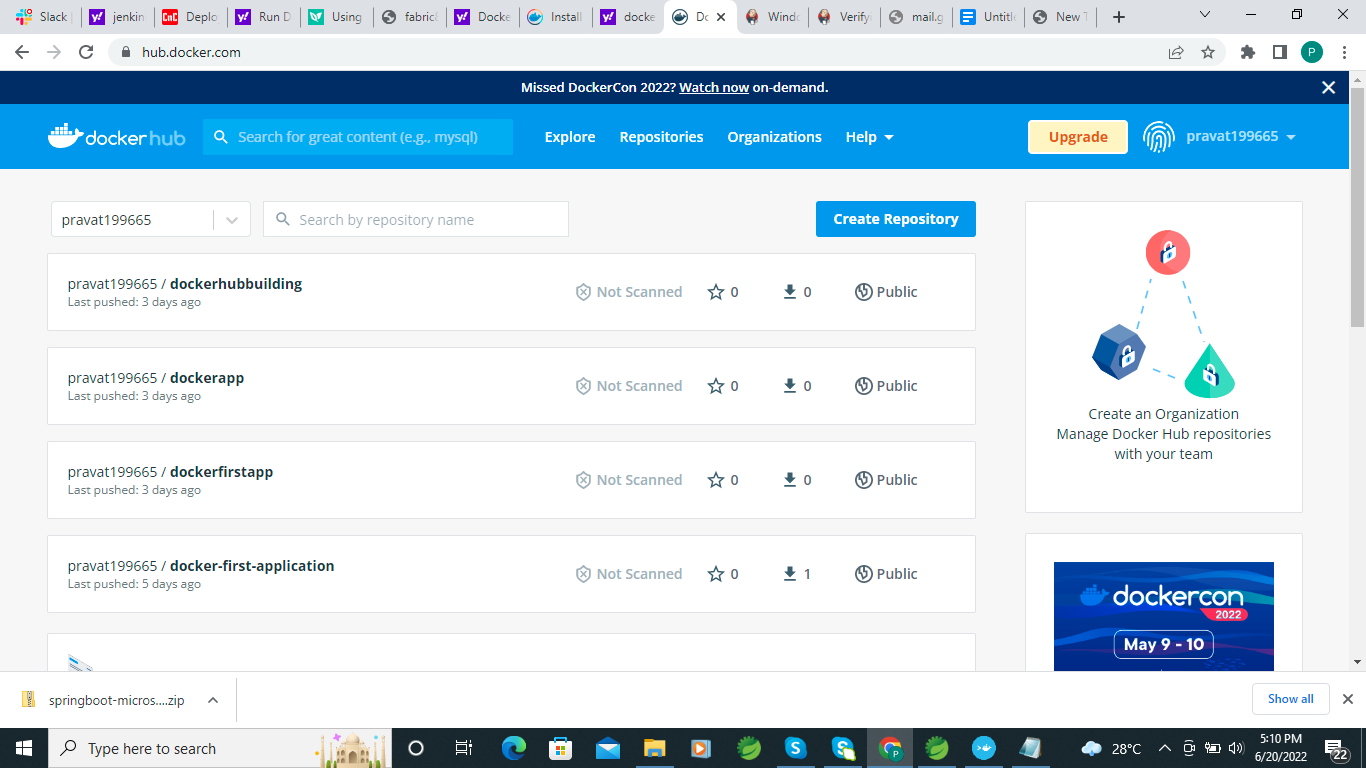
<https://docs.docker.com/desktop/windows/install/>

**Docker Hub Account Creation:-**

* Goto docker hub and create one account

<https://hub.docker.com/>

* Register and Login
* Create repository



Click on Button 'Create Repository'

Enter Project name : ex: dockertest

Choose Public Option

Click on create button

Repository Name

<username>/<repositoryName> => pravat199665/dockertest

* Create Spring boot application Ex: DockerTest
* Create Controller One Test API
* Run Application
* Test :http:localhost:8989/view
* Build the project

Mvn clean

Mvn install

* Application jar file created :ex : DockerTest.jar
* Create Dockerfile

#FROM <image>

FROM openjdk:11

# Provide Port Information

EXPOSE 8080

#ARG is used to define variable which can be passed as argument at run time to build docker image

ARG APP\_NAME=”DockerTest”

ARG APP\_VERSION=”1.0”

ARG JAR\_FILE=”/target/${APP\_NAME}-${APP\_VERSION}.jar”

#Copy jar file to app.jar

COPY ${JAR\_FILE} app.jar

#Jar Execution command

ENTRYPOINT ["java","-jar","app.jar"]

OR Direct pass

#FROM <image>

FROM openjdk:11

# Temp Run location

VOLUME /tmp

# Provide Port Information

EXPOSE 8080

#Jar Location to mapping name

ADD target/DockerTest-1.0.jar DockerTest-1.0.jar

#Jar Execution command

ENTRYPOINT ["java","-jar","/DockerTest-1.0.jar"]

Open Docker Terminal using administrative mode

It take time to open docker terminal

Check docker image list

Cmd >docker image ls

Check docker container

Cmd > docker container ls

Build docker image

Change Location to Project folder

$ cd C:/Pravat/Project/DokcerTest

Cmd >docker build -f Dokcerfile -t <Tagname> .

Docker build -f dockerfile -t dockertest

Run Image

Cmd > docker image ls

Cmd >docker run -p 8989:8989 dockertest

[http://192.168.99.100:8989/](http://192.168.99.100:8989/show)view

Docker Hub push/pull

Login docker hub account

Cmd >docker login

Username

Password

2. Create a tag , that link our local file and Remote Repository

Cmd >docker tag <image> <username>/<repository>:<tagName>

Docker tag dockertest pravat199665/dockertest:latest

docker push pravat199665/dockertest:latest

docker pull pravat199665/dockertest:latest

Docker logout

Jenkins and Create pipeline job :-

After install complete check jenkins service running or not

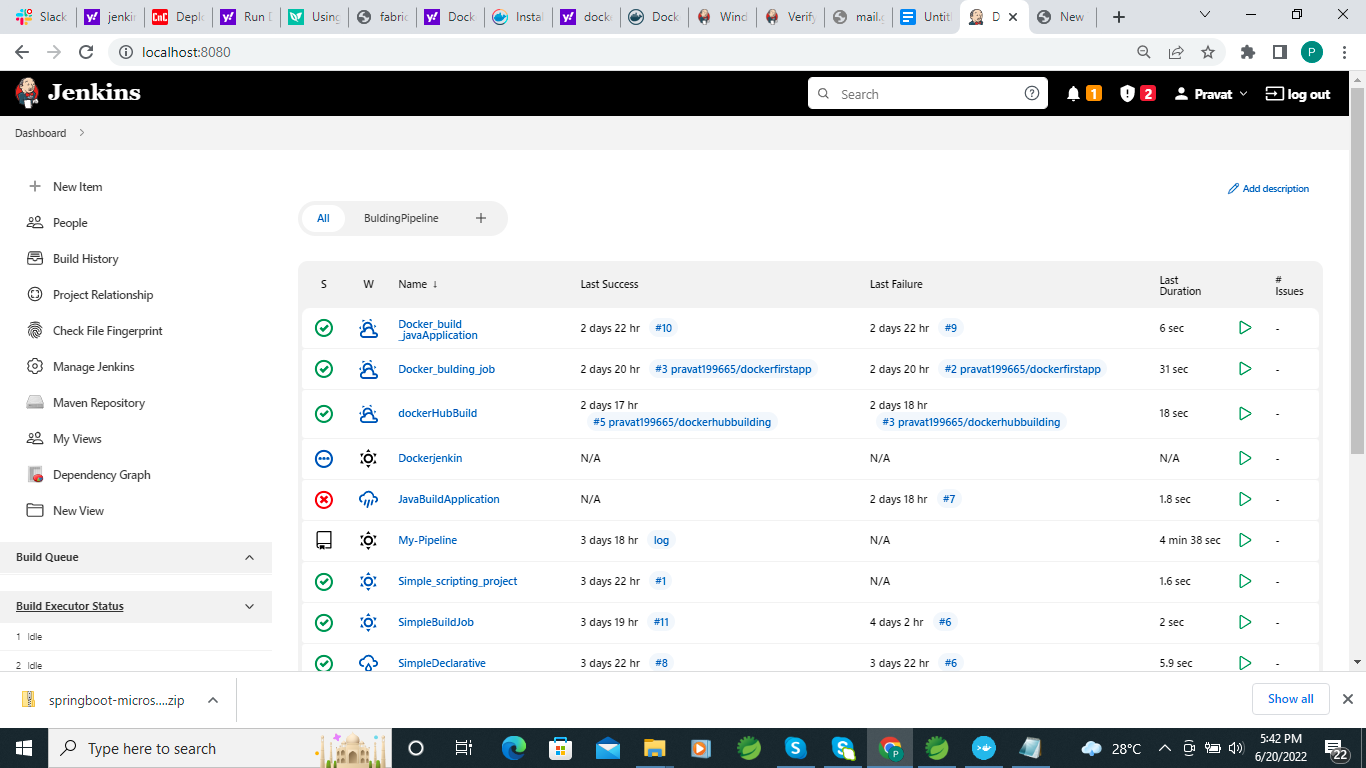
Goto windows search >service -> check jenkins service status

If stop then start jenkins service

Open browser

Type http://localhost :8080

Login your credenation



First create job without plugin

New item ->ItemName → Freestyleproject → Ok→

First i have check build without source code management

Non

Goto Build section

Select windows batch command

echo Current time is %mydate%:%mytime%

echo "Build Process sucessfully......."

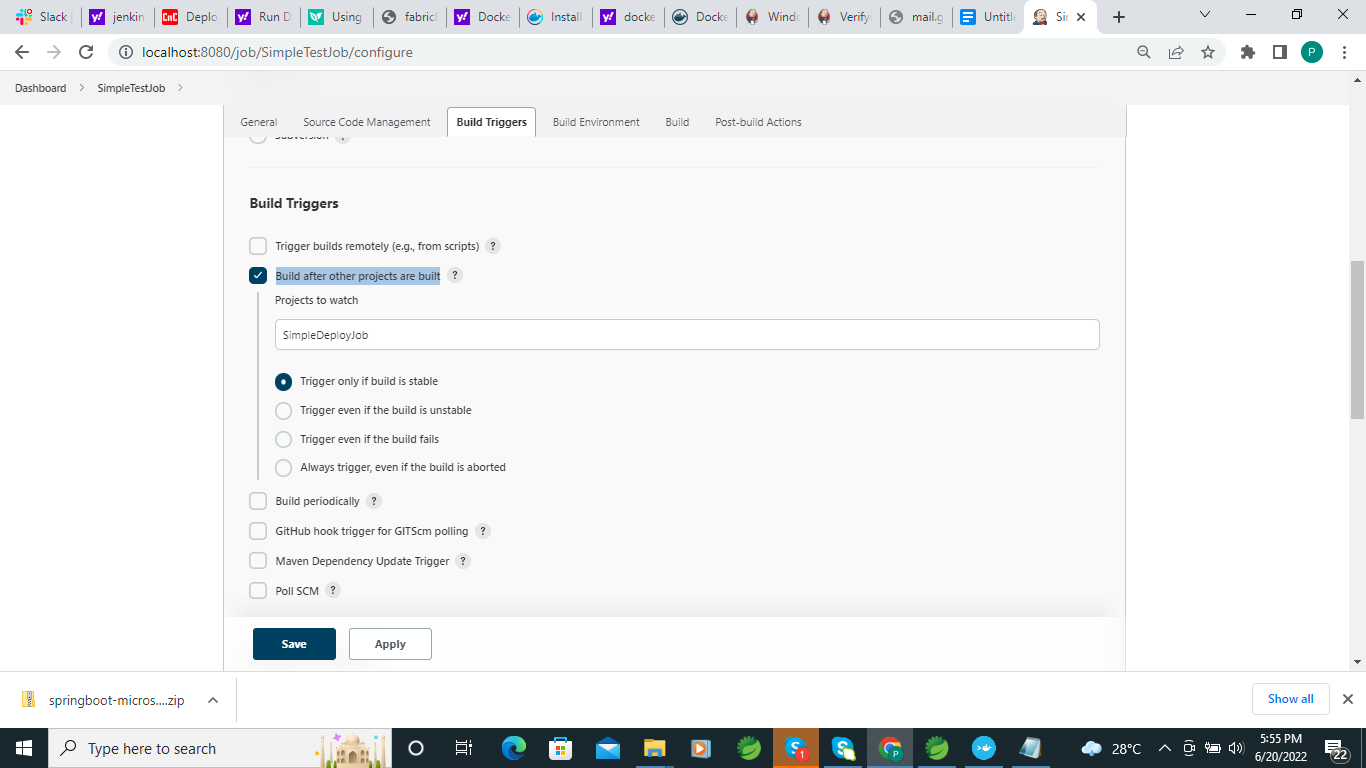
Apply and ok

I have created 3 other items

Build->Test->Deploy→Release

Then We have select sequence order to execute the jobs

Goto Test→Configuration—>**Build Triggers→Build after other projects are built and add which Job after execute**

****

**All 2 Jobs also selected**

**Then Build The job first Which is Build**

**Then all 4 jobs are executed one after the other.**

**We have to use Building Pipeline Plugin to execute Jobs**

**Install Building Pipeline plugin in manage jenkins —>manage plugin**

**Search building pipeline**

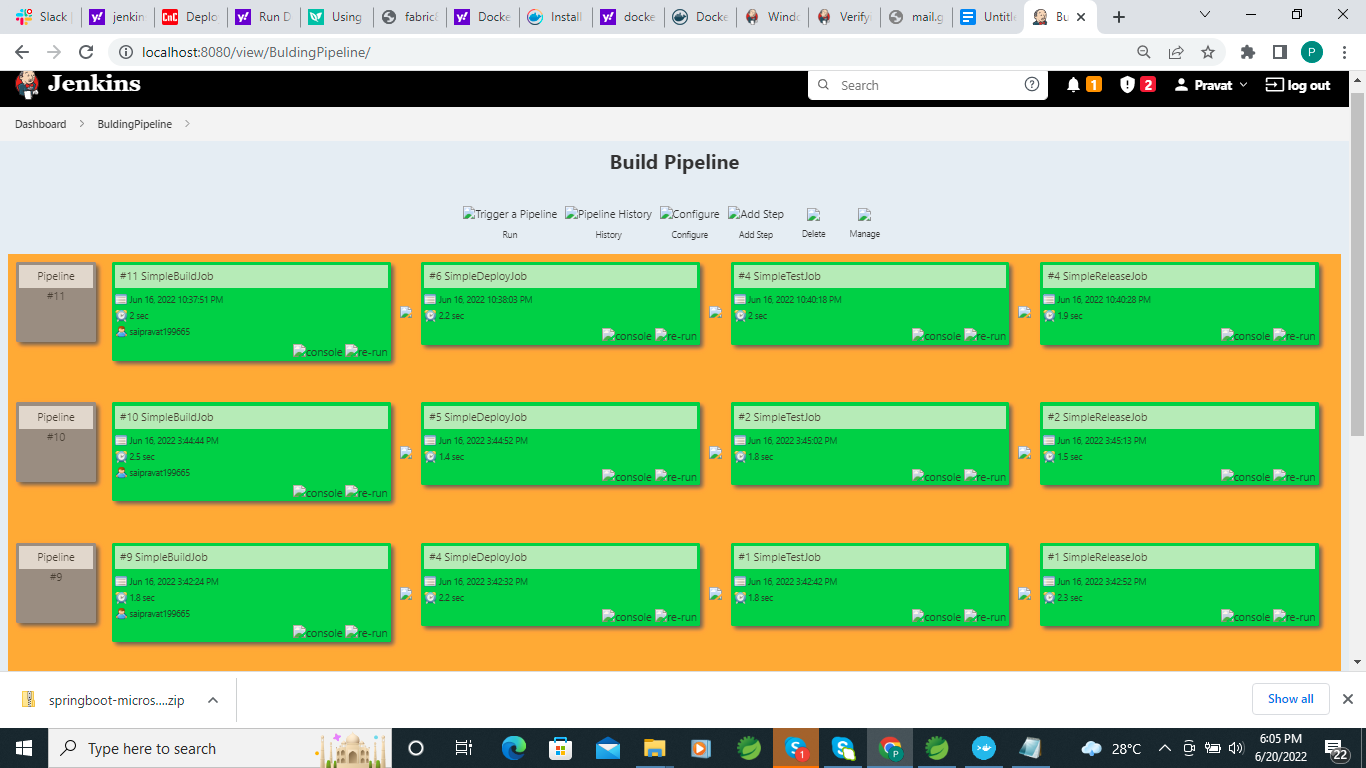
**And install**

**Create Building Pipeline and select**

**Upstream / downstream config Jobs**

**Apply and ok**

**Then run the build plugins job**

****

**Create Pipeline Job using jenkinsfile**

**====================**

**Create Spring boot application and added a jenkins file**

**Add file**

**pipeline{**

**agent any**

**stages{**

**stage('Build'){**

**steps{**

**echo "Building the project......."**

**}**

**}**

**stage('Test'){**

**steps{**

**echo "Test the project.....a.."**

**}**

**}**

**stage('Deploy'){**

**steps{**

**echo "deploy the project......."**

**}**

**}**

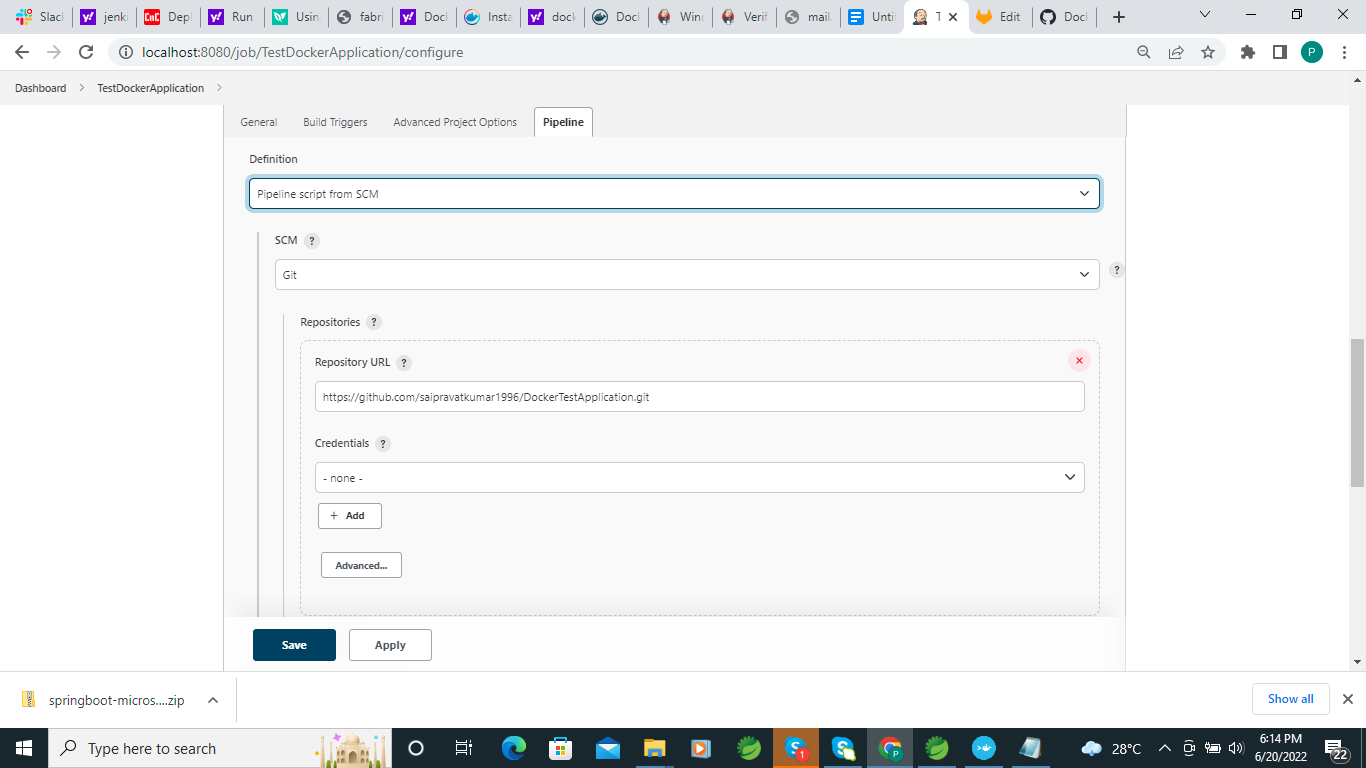
**}**

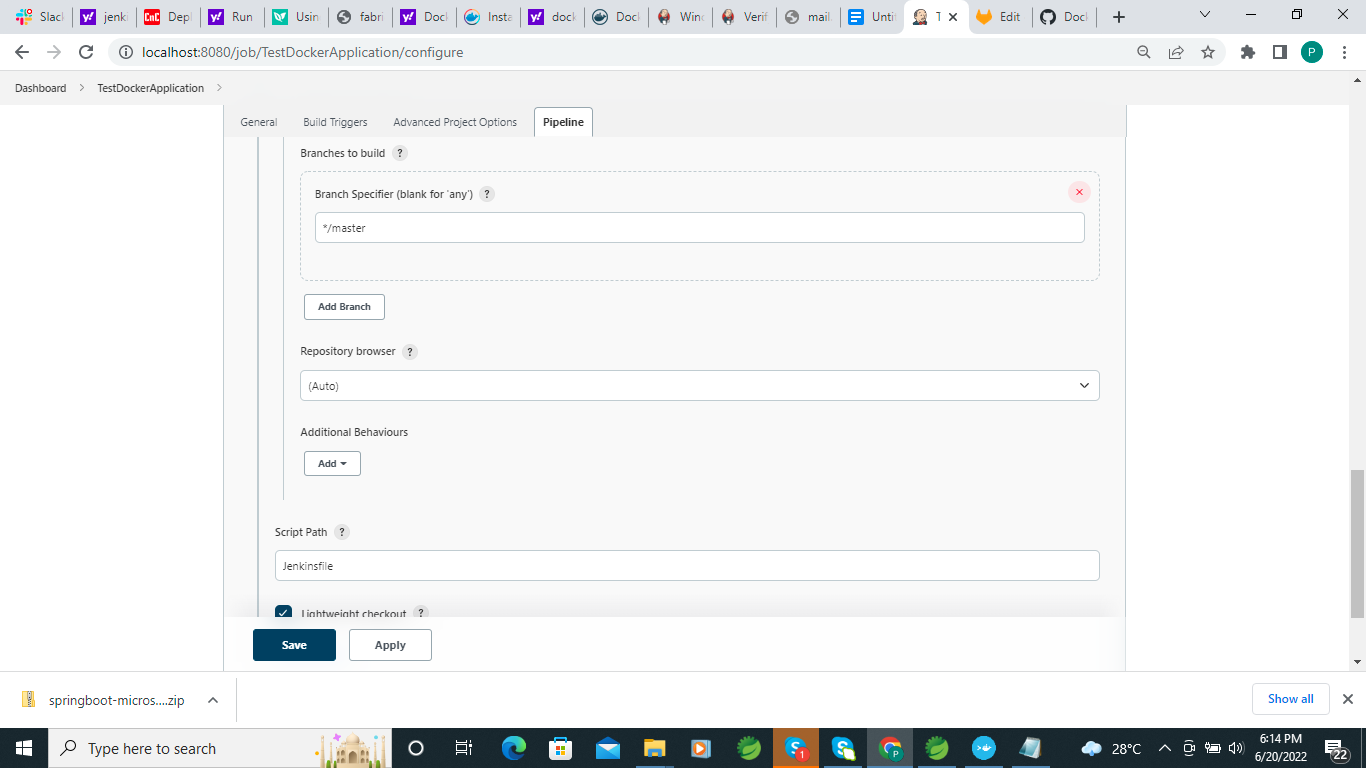
**}**

**Then Create Item in jenkin**

**Itename -> Pipeline ->Pipeline→ If we use script in github then use pipeline script in csm other wise we have to write script**

**I have select csm i have used git repository jenkinsfile**

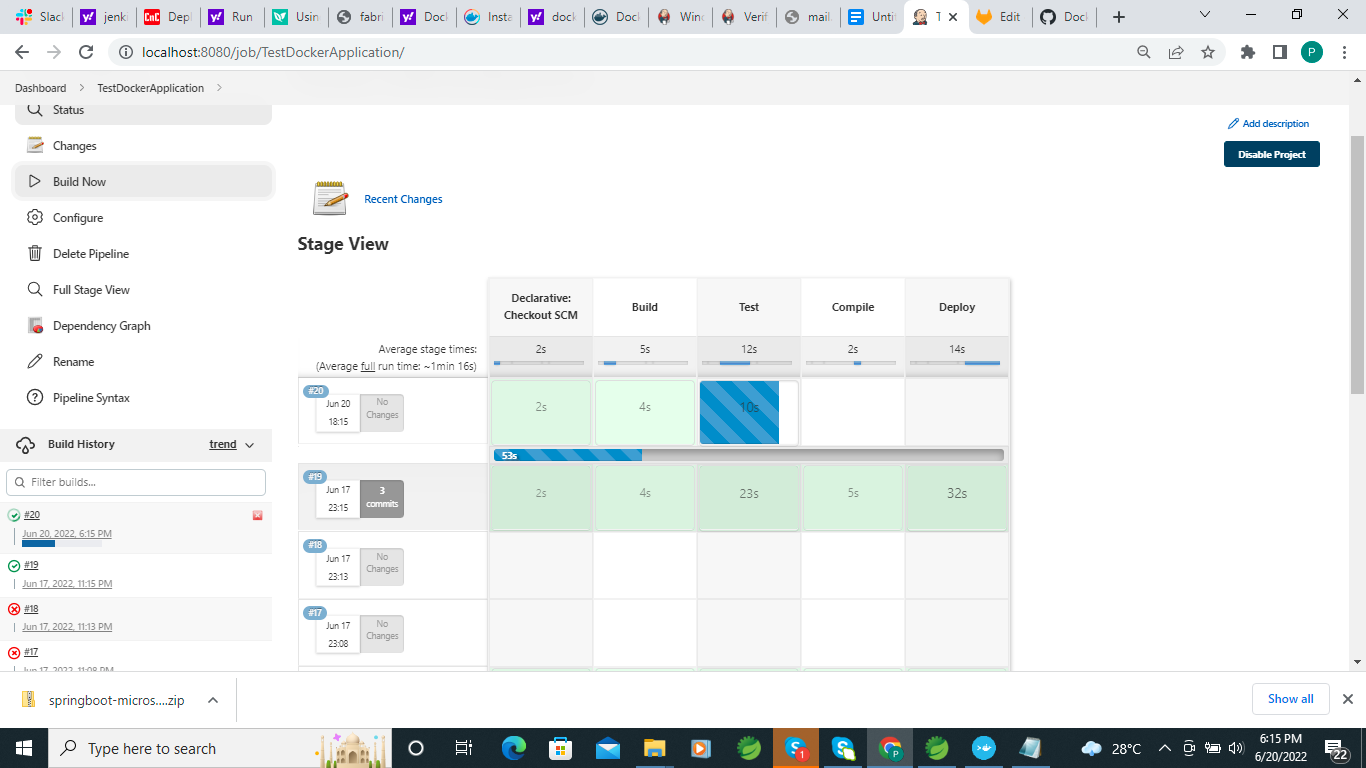
****

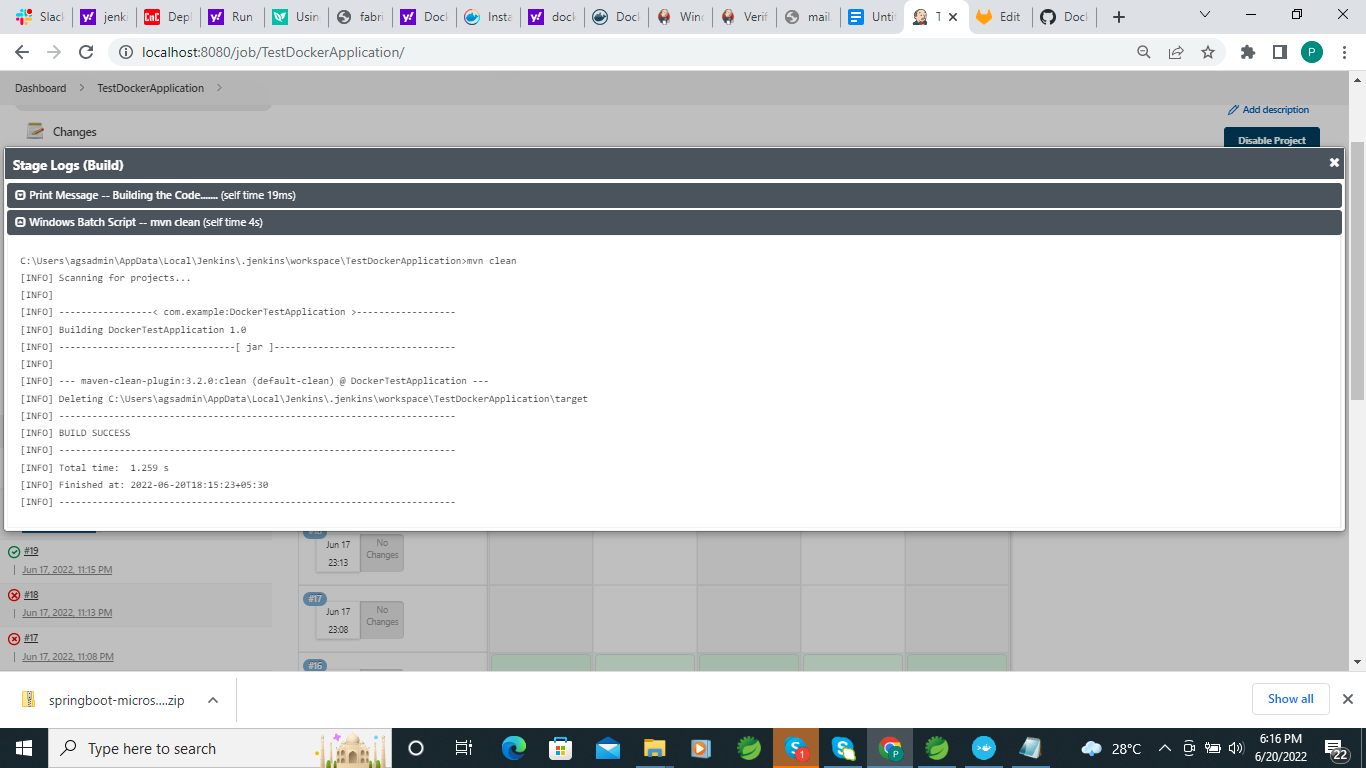
****

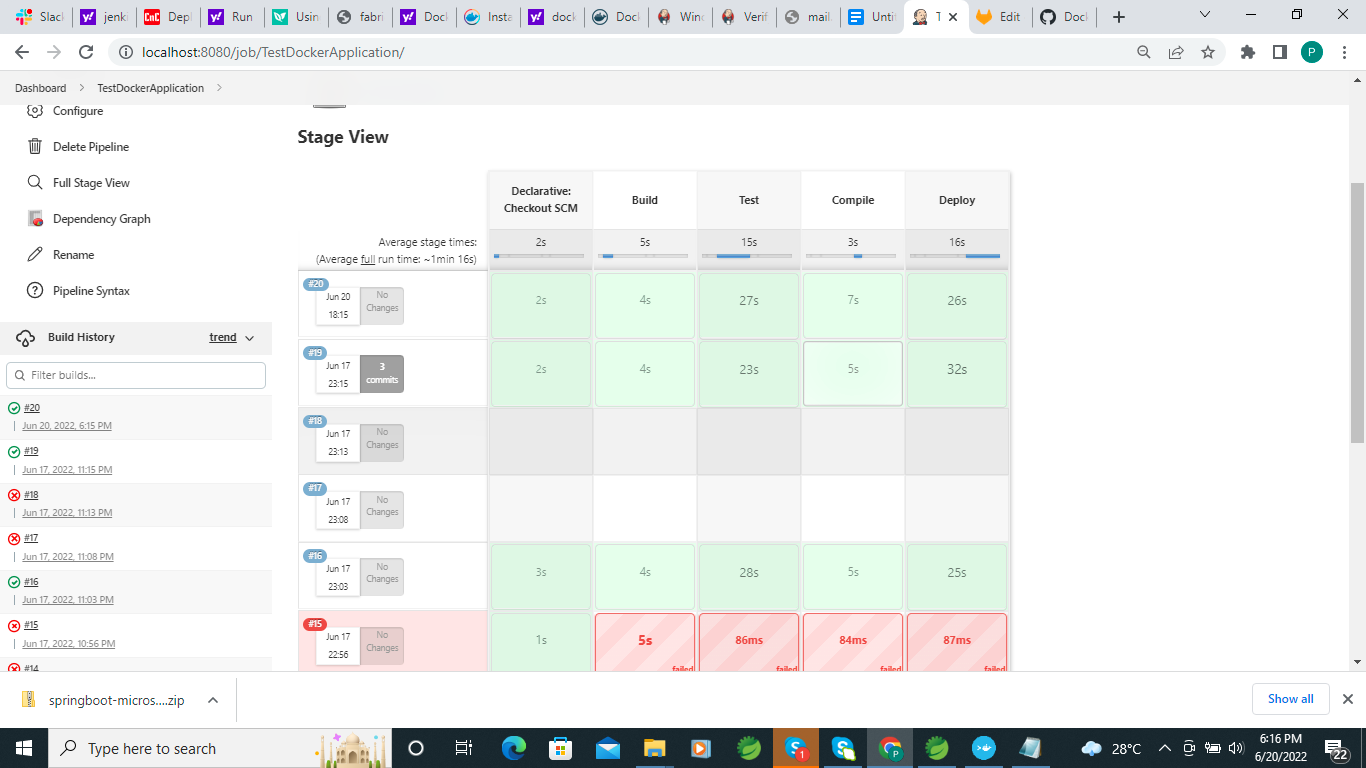
**Which filename define in github same name mention in script path**

**Apply and save**

**Build the job then it execute one by one**

****

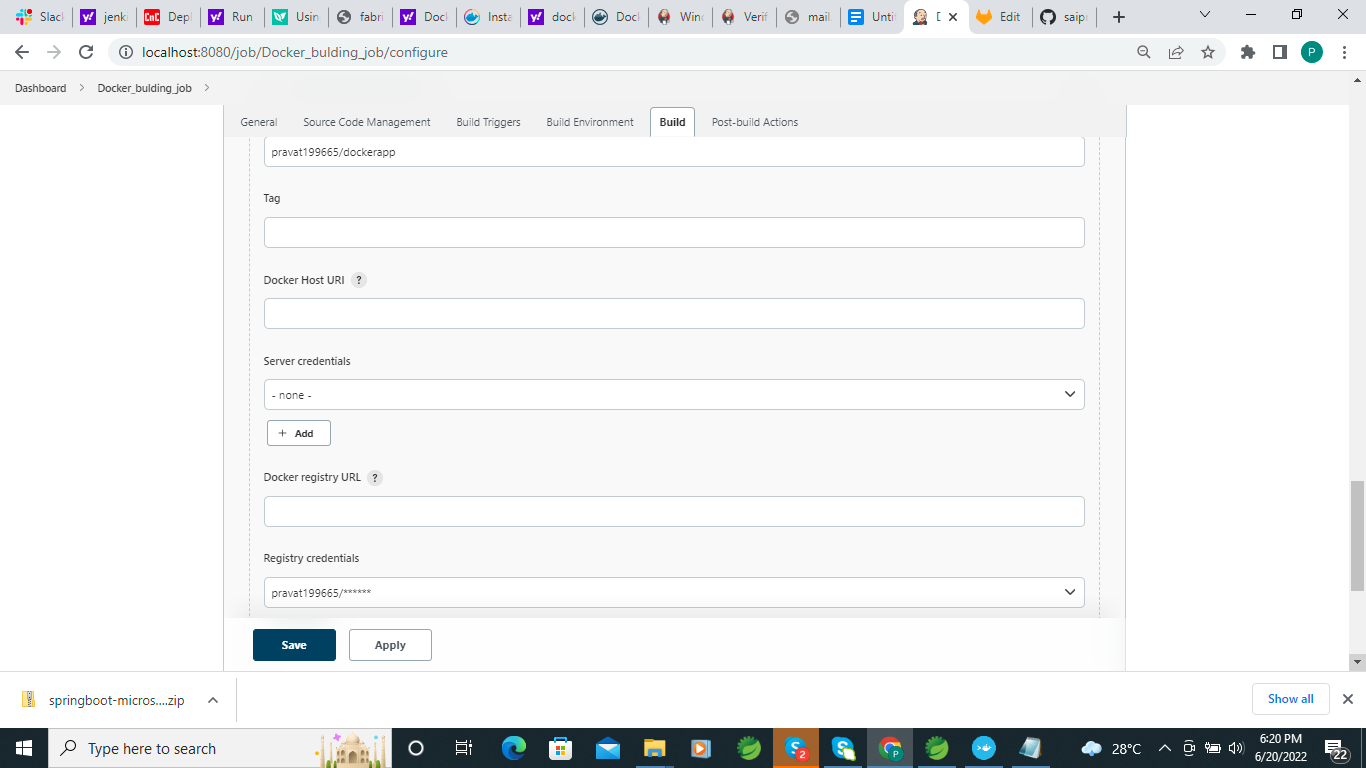
****

****

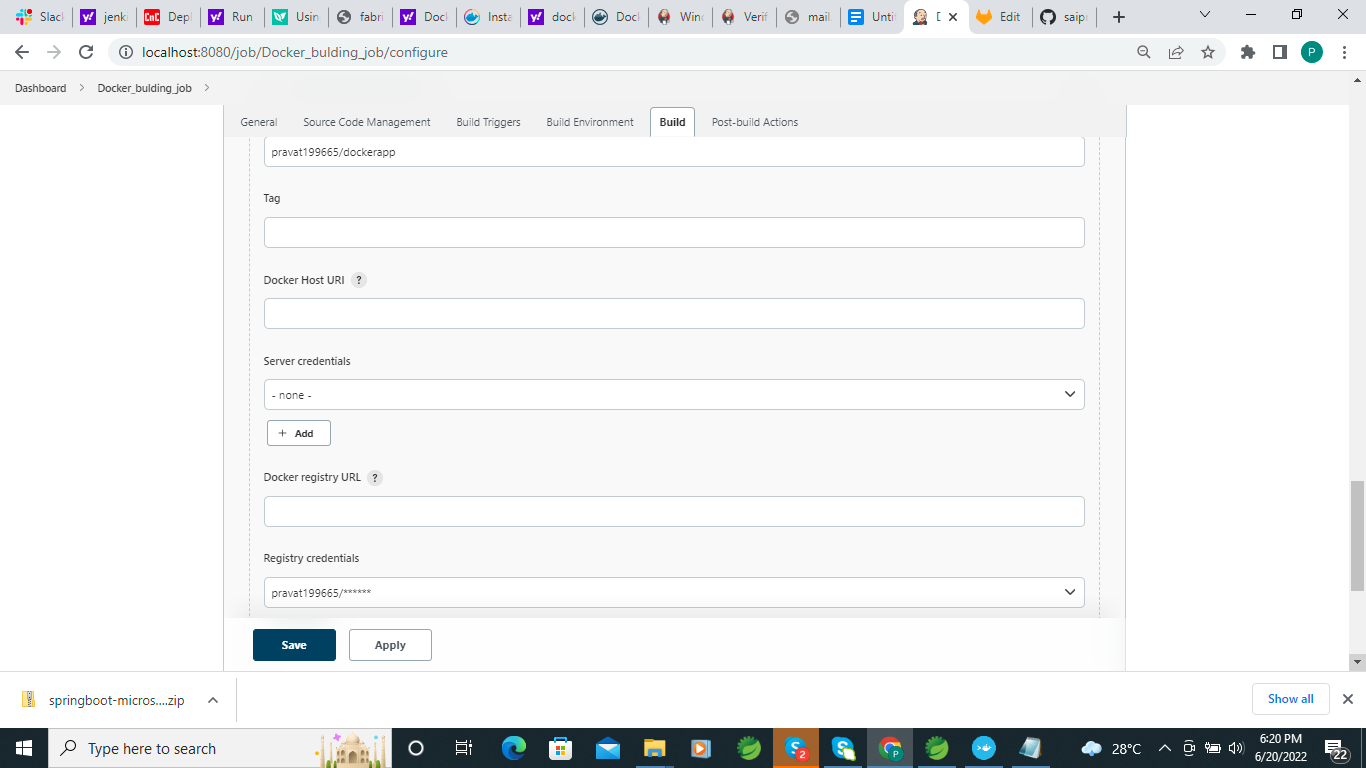
**If We build docker Build using jenkins in build section we have to choose Docker Build and Publish but Docker Build and Publish is plugin we have to install first**

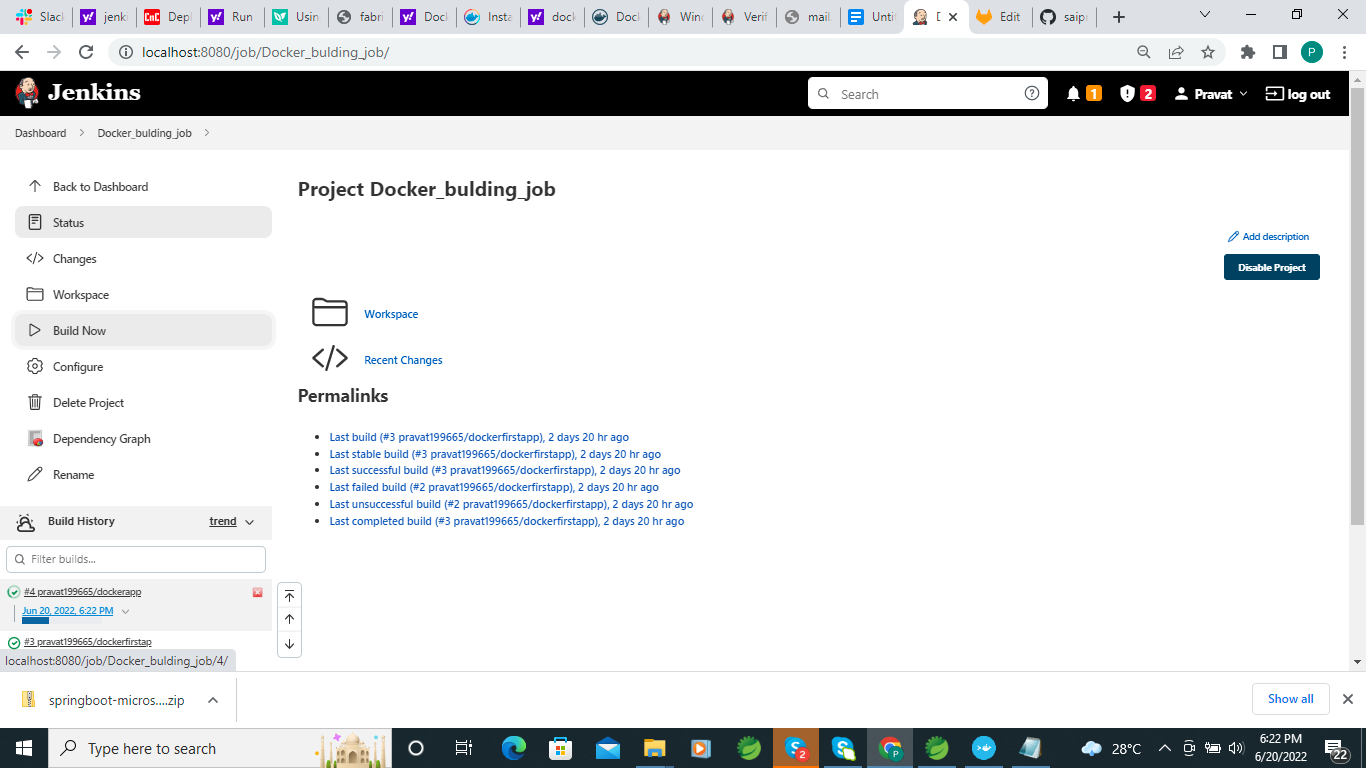
**Use your docker hub username/repostoryname**

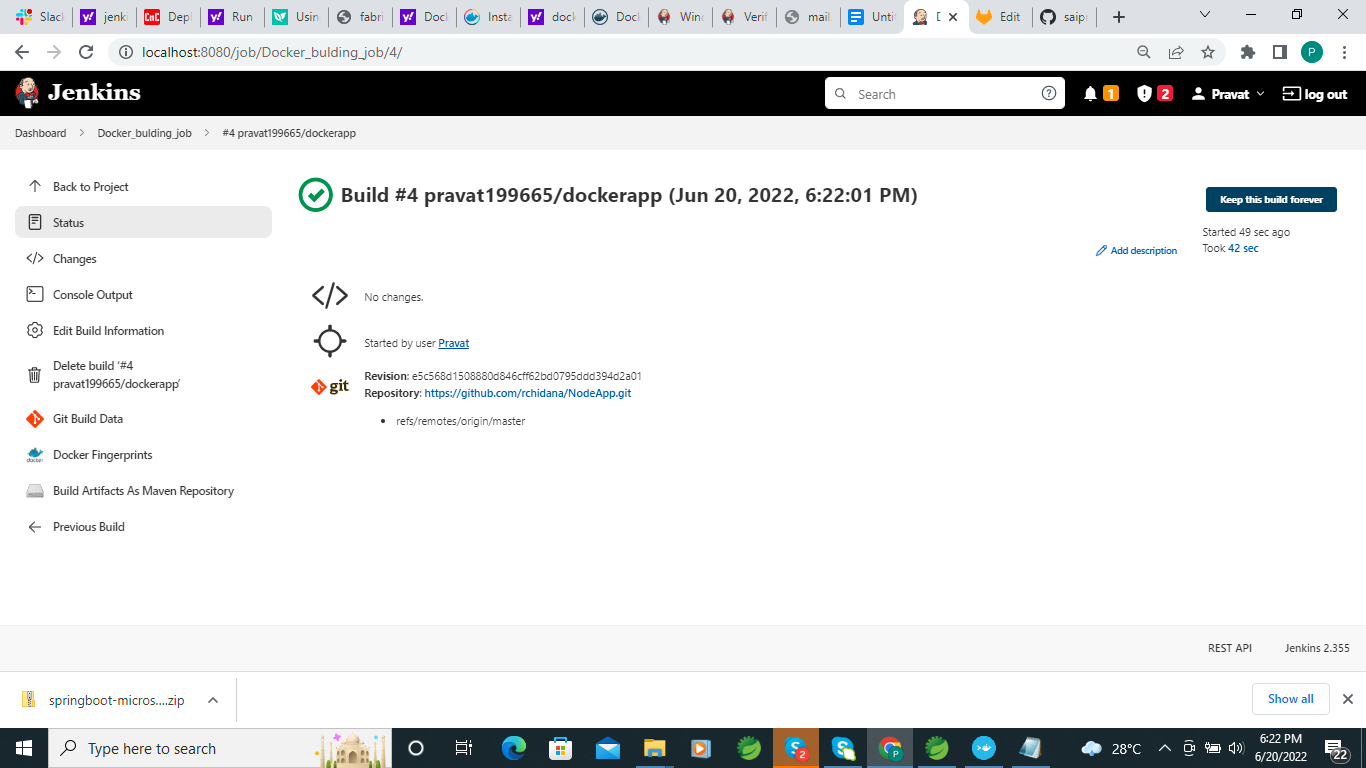
**In the registry we have to use docker hub credential**

****

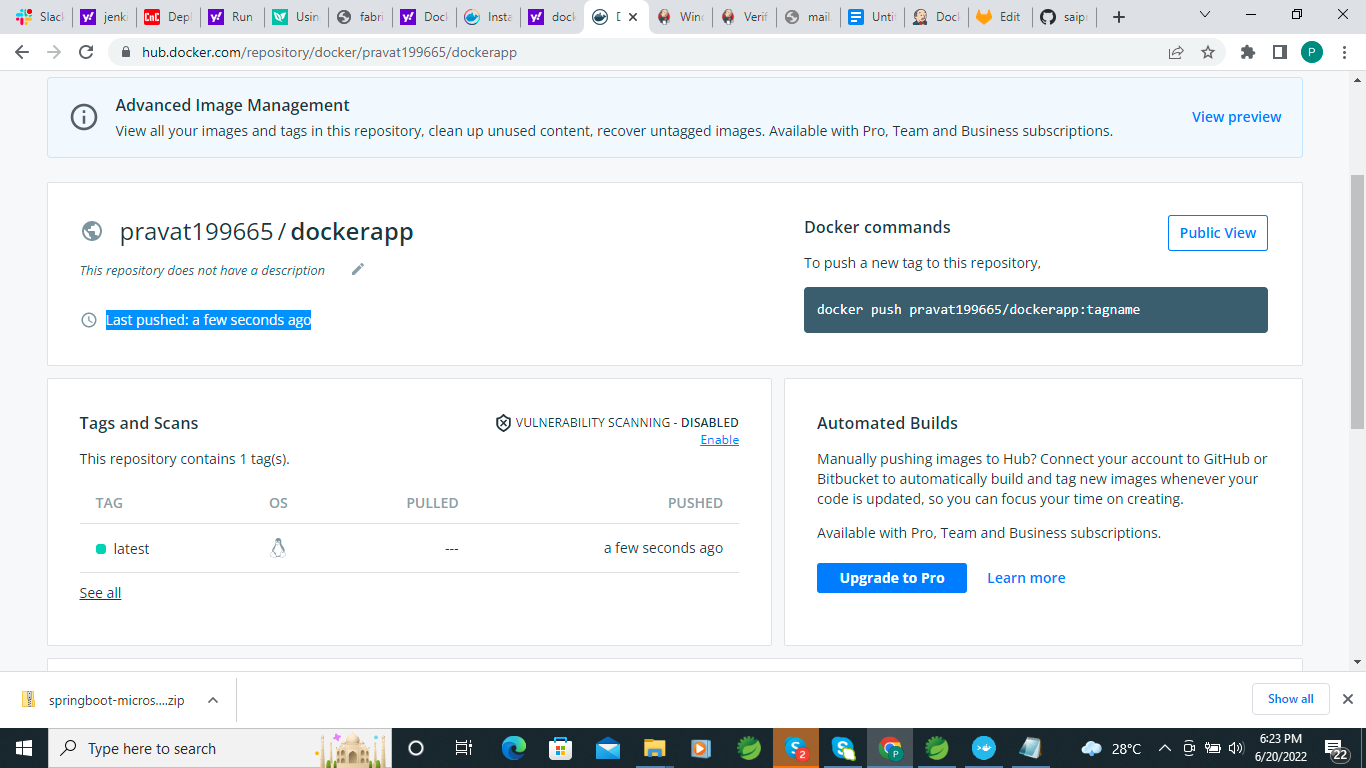
**Apply and save then build application**

****





Check Docker hub and docker image you get docker image



I have faced issue after creating image through jenkins the i can’t run application and DB also not connected in Docker.

I am working now after creating docker image through jenkins how to run.

Issue Solved :-

no main manifest attribute

Adding pom.xml file to your main class name

<plugin>

<!-- Build an executable JAR -->

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<configuration>

<archive>

<manifest>

<mainClass>com.example.demo.JenkinApplication</mainClass>

</manifest>

</archive>

</configuration>

</plugin>

When execute Jenkin script it automatically created docker image but manually run the docker image.

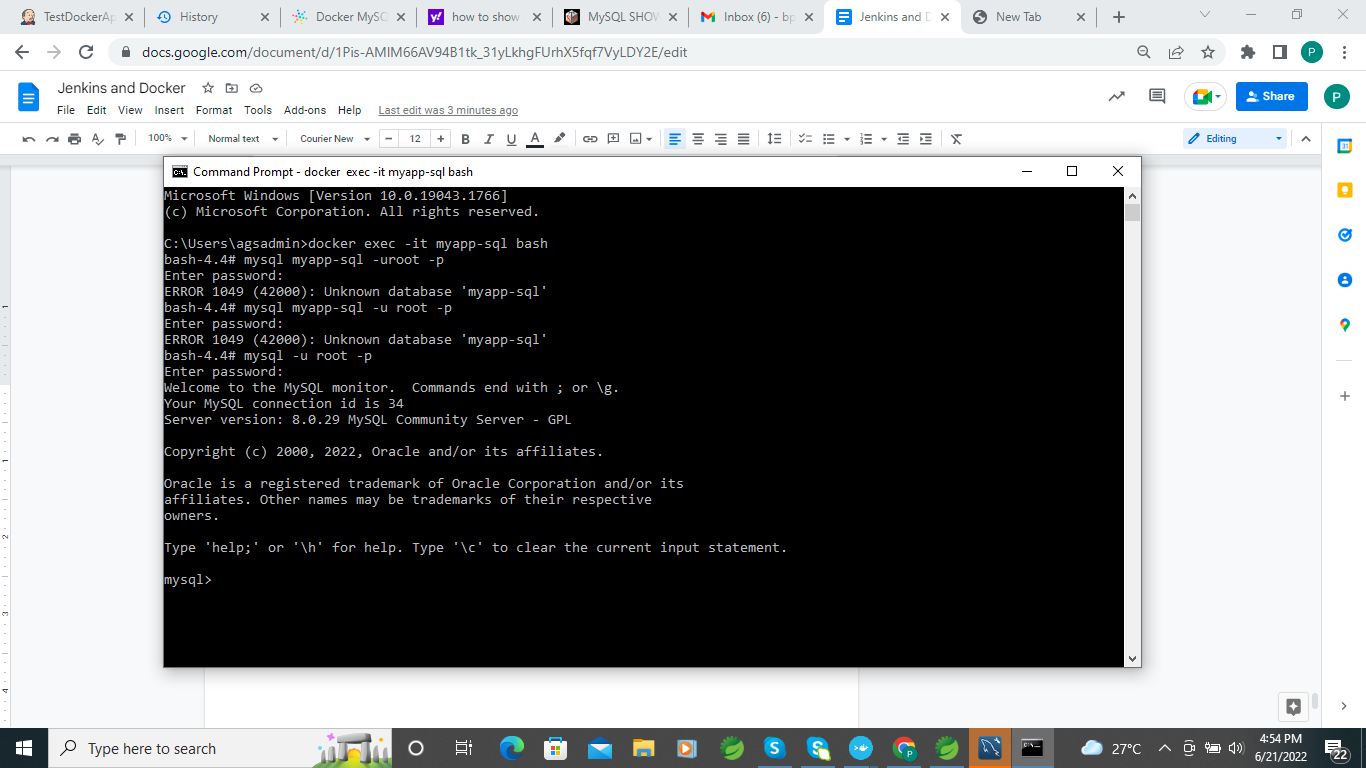
Run Mysql in Docker:-

docker run --name myapp-mysql -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=mydb -p 3306:3306 -d mysql:latest

sudo docker exec -it [container\_name] bash

* For the container you created as an example, you can run this command:

sudo docker -it mysql\_docker bash



**Database connection docker with Spring boot application :-**

**https://berativ.com/docker-with-spring-boot-and-mysql/**

Create spring boot application with db connection

Build maven clean and install to generated jar file

Inside Docker :-

Create Mysql db image

$ Docker pull mysql

Run Mysql image

$ **docker run -d -p 3306:3306 --name=docker-mysql --env="MYSQL\_ROOT\_PASSWORD=root" --env="MYSQL\_PASSWORD=root" --env="MYSQL\_DATABASE=demo" mysql**

Create docker application docker image

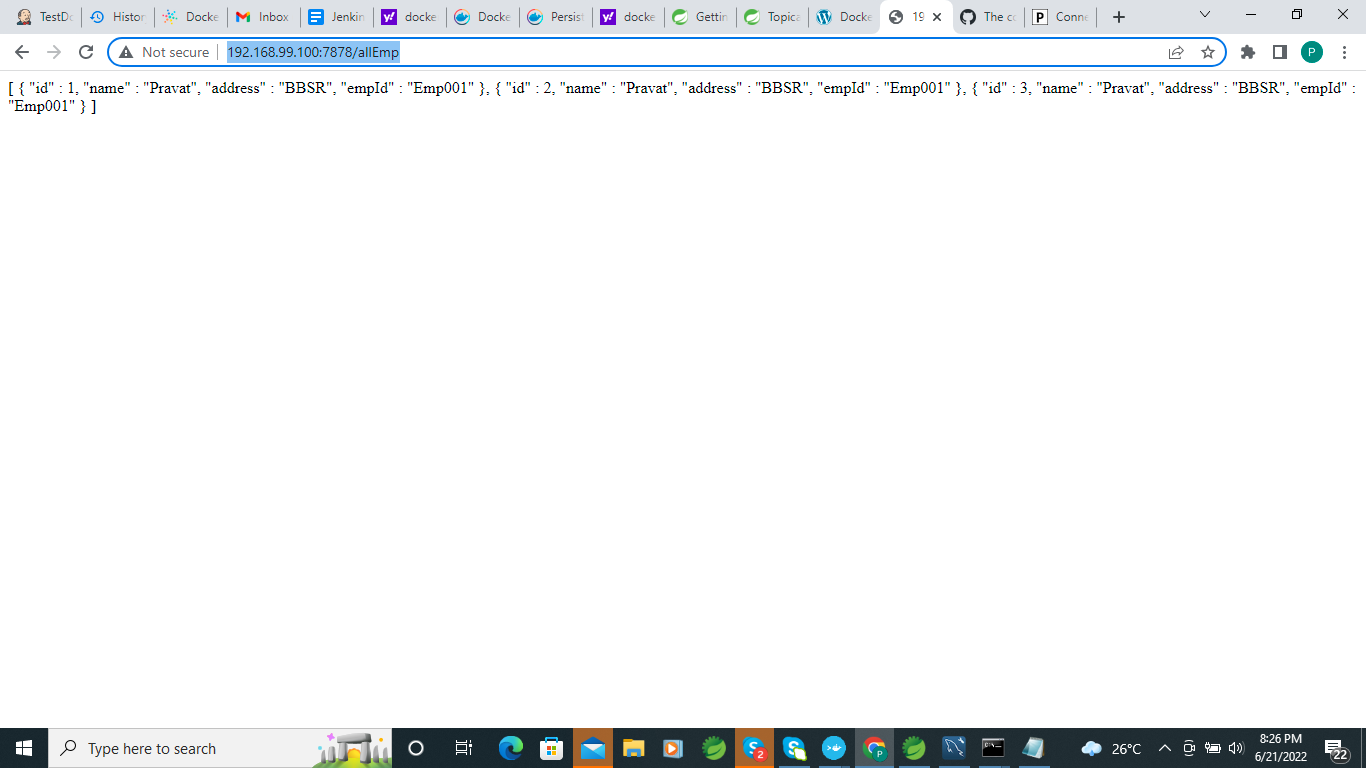
$ **docker image build -f Dockerfile -t <tagname> .**

$ **docker image build -Dockerfile -t testapp .**

**Run Image with docker environment db connection.**

**$ docker run -t --link docker-mysql:mysql -p 7878:7878 --name demo --env "SPRING\_DATASOURCE\_URL=jdbc:mysql://docker-mysql:3306/demo" demo**

[**http://192.168.99.100:7878/allEmp**](http://192.168.99.100:7878/allEmp)

****

**ocker run -e "SPRING\_PROFILES\_ACTIVE=dev" -p 8080:8080 -t springio/gs-spring-boot-docker**

**In Docker File We can use this**

**FROM maven:3.6.3-jdk-11-slim AS MAVEN\_BUILD**

**2**

**#FROM maven:3.5.2-jdk-8-alpine AS MAVEN\_BUILD FOR JAVA 8**

**3**

**4**

**ARG SPRING\_ACTIVE\_PROFILE**

**5**

**6**

**MAINTAINER Jasmin**

**7**

**COPY pom.xml /build/**

**8**

**COPY src /build/src/**

**9**

**WORKDIR /build/**

**10**

**RUN mvn clean install -Dspring.profiles.active=$SPRING\_ACTIVE\_PROFILE && mvn package -B -e -Dspring.profiles.active=$SPRING\_ACTIVE\_PROFILE**

**11**

**FROM openjdk:11-slim**

**12**

**#FROM openjdk:8-alpine FOR JAVA 8**

**13**

**WORKDIR /app**

**14**

**15**

**COPY --from=MAVEN\_BUILD /build/target/appdemo-\*.jar /app/appdemo.jar**

**16**

**ENTRYPOINT ["java", "-jar", "appdemo.jar"]**

**https://dzone.com/articles/automate-spring-boot-app-deployment-with-gitlab-ci**

==================================================

What is Jenkins ?

Jenkins is one of the most powerful tools for Continuous Integration and Continuous Deployment (CI/CD).

What is Jenkins Pipeline ?

Jenkins Pipeline is a collection of plugins that launch “Build ,test and deploy ” processes in a predefined sequence .

The Jenkins Pipeline Consolidates a set of instructions that are essential for building and testing applications in a repeatable manner. The idea is to extend the CD practices capabilities in a repeatable manner.The idea is to extend the CD practice capabilities,Which aims to bring the software from version control to a user-facing release.

**Different Type of Jenkins CI/CD Pipeline**

**The definition of a Jenkins Pipeline is constituted in the text file called Jenkinsfile. This file defines the implementation of Jenkins Pipeline in the source control repository. Moreover, all the aforementioned pipeline benefits accrue from this definition.**

**The Jenkinsfile supports two syntaxes – Declarative and Scripted. While Scripted syntax and the associated Scripted pipelines have been around for some time, Declarative syntax and its associated Declarative pipelines are relatively new.**

**Scripted Pipeline :-**

**A scripted pipeline uses the Groovy (JVM-based) language to create a pipeline as code. Some software developers prefer this pipeline because the use of Groovy code paves the way for making numerous APIs available. As a result, developers can orchestrate rather complex CD pipelines.**

**For creating scripted pipelines in Jenkins, developers must include a pipeline code within the UI and save it. Here’s the general syntax of a scripted pipeline:**

**Jenkinsfile (Scripted Pipeline)**

**node {**

**stage(‘Build’) {**

**…….**

**}**

**stage(‘Test’) {**

**…….**

**}**

**stage(‘Deploy’) {**

**…….**

**}**

**}**

### **Problems with scripted pipeline:**

* **The imbalance between developers’ experience in Groovy and Jenkins creates an impedance mismatch that is not easy to remove.**
* **Although scripted syntax is powerful and expedites the creation of complex pipelines, it concurrently makes the pipeline hard to read and hard to manage.**

**Declarative Pipeline :-**

**The declarative pipeline follows a relatively newer syntax, which narrows the scope of the pipeline structure through a pre-defined “minimum necessary structure.” On the technical front, the declarative syntax follows a conditional statement logic that generates a series of stages, executed sequentially throughout the pipeline’s lifecycle. Here’s the general syntax of a declarative pipeline:**

**Jenkinsfile (Declarative Pipeline)**

**pipeline {**

**agent any**

**stages {**

**stage(‘Build’) {**

**steps {**

**…….**

**}**

**}**

**stage(‘Test’) {**

**steps {**

**…….**

**}**

**}**

**stage(‘Deploy’) {**

**steps {**

**…….**

**}**

**}**

**}**

**}**

### Problems with declarative pipeline:

* **Since declarative pipelines don’t make use of Groovy code, the declarative syntax doesn’t access valuable APIs. But, at the end of the day, the declarative syntax is uncomplicated to read and comparatively easier to manage.**

**Jenkins Pipeline Palm Insurance ID**

**Git bpravat**

**Saibaba@96**

**Palm\_Insurance**

**Issue maven Build in Jenkins**

**[ERROR] The goal you specified requires a project to execute but there is no POM in this directory (C:\Users\agsadmin\AppData\Local\Jenkins\.jenkins\workspace\PalmInsurance). Please verify you invoked Maven from the correct directory.**

**To solve this problem**

**echo "mvn -f PalmInsurance/pom.xml clean"**

**echo "mvn -f PalmInsurance/pom.xml install"**

**If for some reason, we need to run the code from a different directory, the resolution is to provide the path for pom.xml while executing the maven goals. We can achieve this by using** *-f* **or —***file* **option. So instead of mentioning the goal as**

***mvn compile***

**we need to mention it as**

***mvn – f < pom.xml path > compile***

**The – f option will enable Jenkins to use the pom.xml mentioned next to it for executing the maven goal. Moreover, if we are running the code from Jenkins agent in windows, we will need to escape backslashes in the file path by using double backslashes as shown in code excerpt.**

[**https://thetechdarts.com/maven-error-in-jenkins-there-is-no-pom-in-this-directory/**](https://thetechdarts.com/maven-error-in-jenkins-there-is-no-pom-in-this-directory/)

**Declarative Pipeline Jenkinsfile :-**

**pipeline{**

**agent any**

**stages{**

**stage('Build'){**

**steps{**

**echo "Building the Code......."**

**bat "mvn -f PalmInsurance/pom.xml clean"**

**bat "mvn -f PalmInsurance/pom.xml compile"**

**bat "mvn -f PalmInsurance/pom.xml install"**

**}**

**}**

**stage('Test'){**

**steps{**

**echo "Test the project......."**

**echo "mvn test"**

**bat "mvn -f PalmInsurance/pom.xml test"**

**}**

**}**

**stage('Create Docker Image'){**

**steps{**

**echo "Build Docker Image........"**

**bat "docker container stop alterauto"**

**bat "docker container rm -f alterauto"**

**bat "docker build -f PalmInsurance/alertauto/Dockerfile -t alterauto ."**

**}**

**}**

**stage('Deploy'){**

**steps{**

**echo "deploy the project......."**

**bat "docker run -d --name alterauto -p 8083:8083 alterauto | tee alterauto.log"**

**}**

**}**

**}**

**}**

**WEBHOOKS With Jenkins and GitHub:-**

**In GitLab or Github**

**Goto→ Project Repository —---> Settings —--> WebHooks —-> Add newWebHooks — Payload (Jenkins URL)-->Choose json —-> Use security key (Optional) →check event →ok save**

**In Jenkins**

**Goto →Jobs —>Configure —>Build trigger choose (GitHub hook trigger for GITScm polling) —> save**

**Install Docker In Ubuntu :-**

[**https://test.docker.com**](https://test.docker.com) **Locally download Documents**

[**https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository**](https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository)

**https://docs.docker.com/engine/install/ubuntu/**

**sudo apt-get update**

**sudo apt-get install \**

**ca-certificates \**

**curl \**

**gnupg \**

**Lsb-release**

**sudo mkdir -p /etc/apt/keyrings**

**$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg**

**echo \**

**"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \**

**$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null**

**sudo apt-get update**

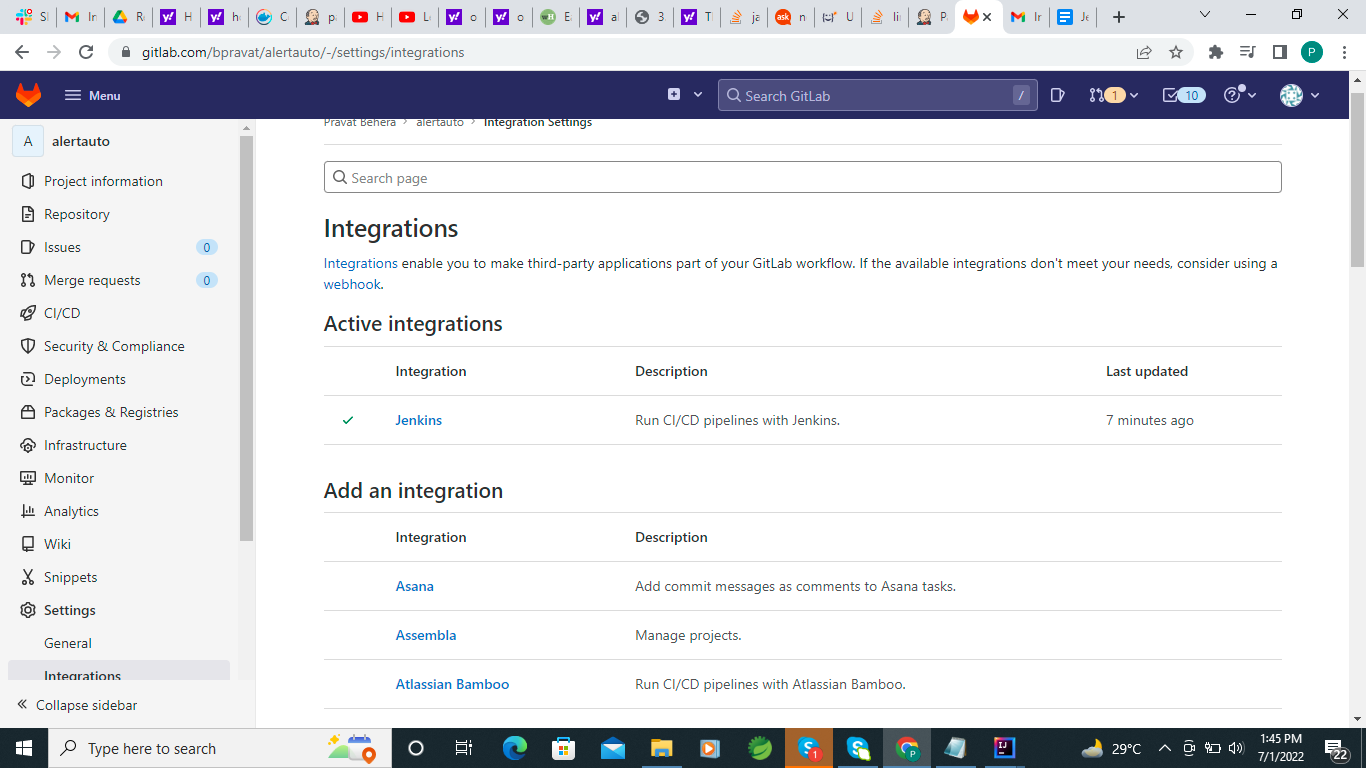
**$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin**

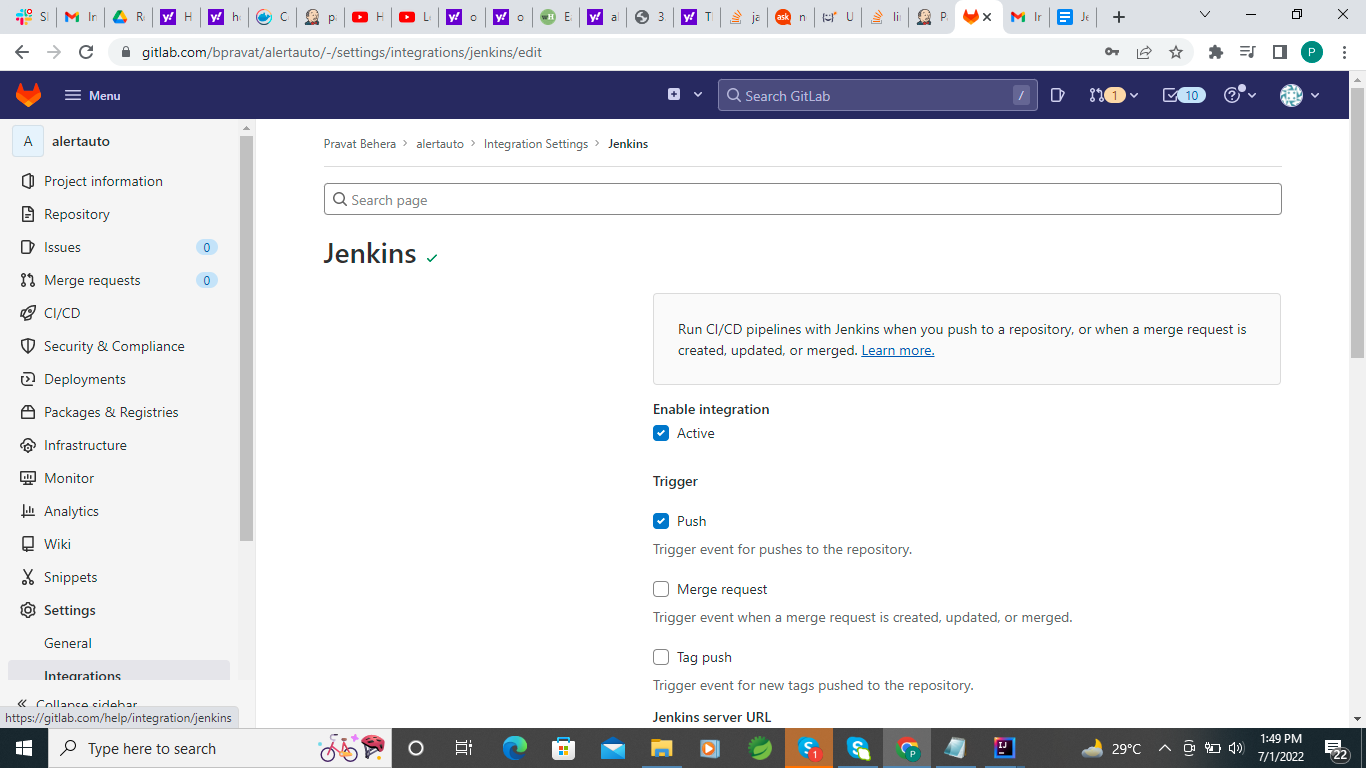
**apt-cache madison docker-ce**

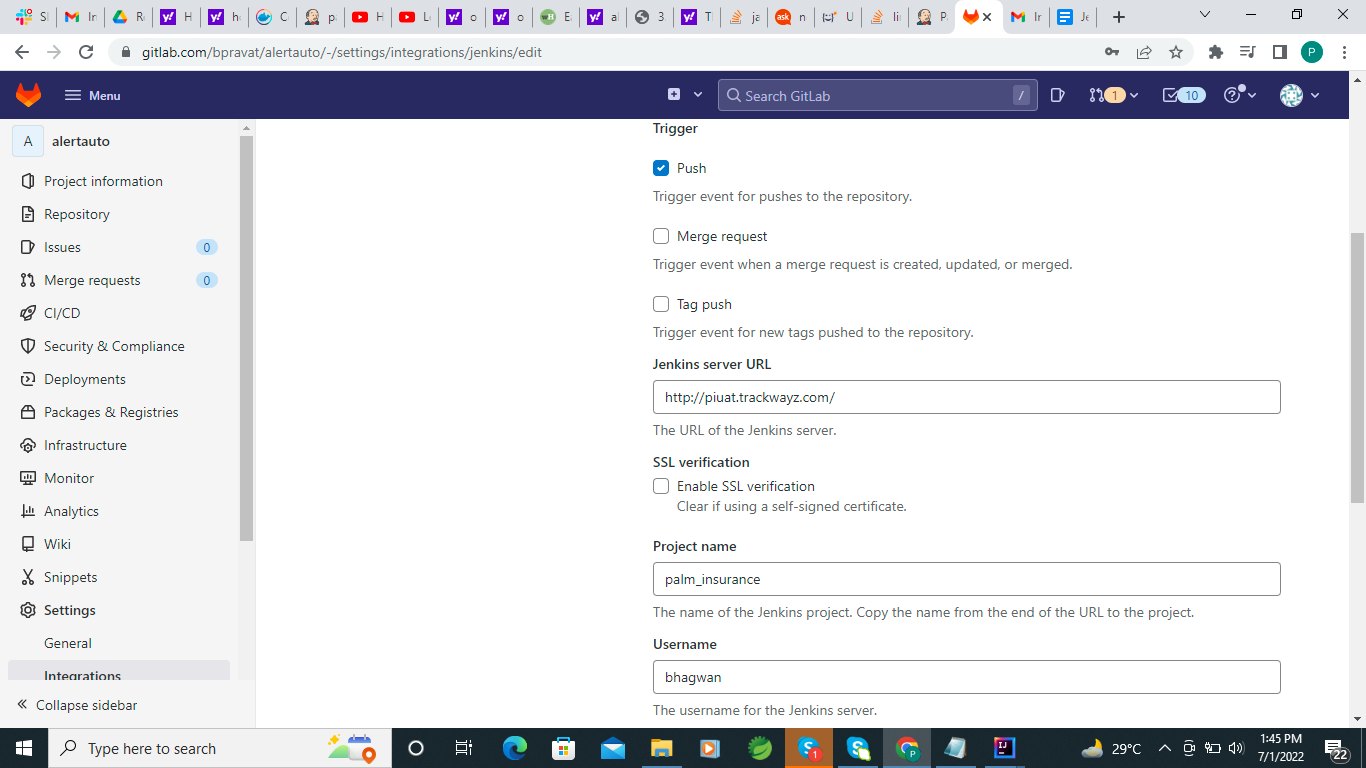
[**https://drive.google.com/file/d/1WeYsF5PtwMbhIY926Ck4veaNjeen-LiL/view?usp=sharing**](https://drive.google.com/file/d/1WeYsF5PtwMbhIY926Ck4veaNjeen-LiL/view?usp=sharing)

**WEB HOOK Integration using Gitlab and jenkins**

**Login Gitlab —>Open Project Repository —>Settings→ Integration →Project event select Jenkins —?**

****

****

****

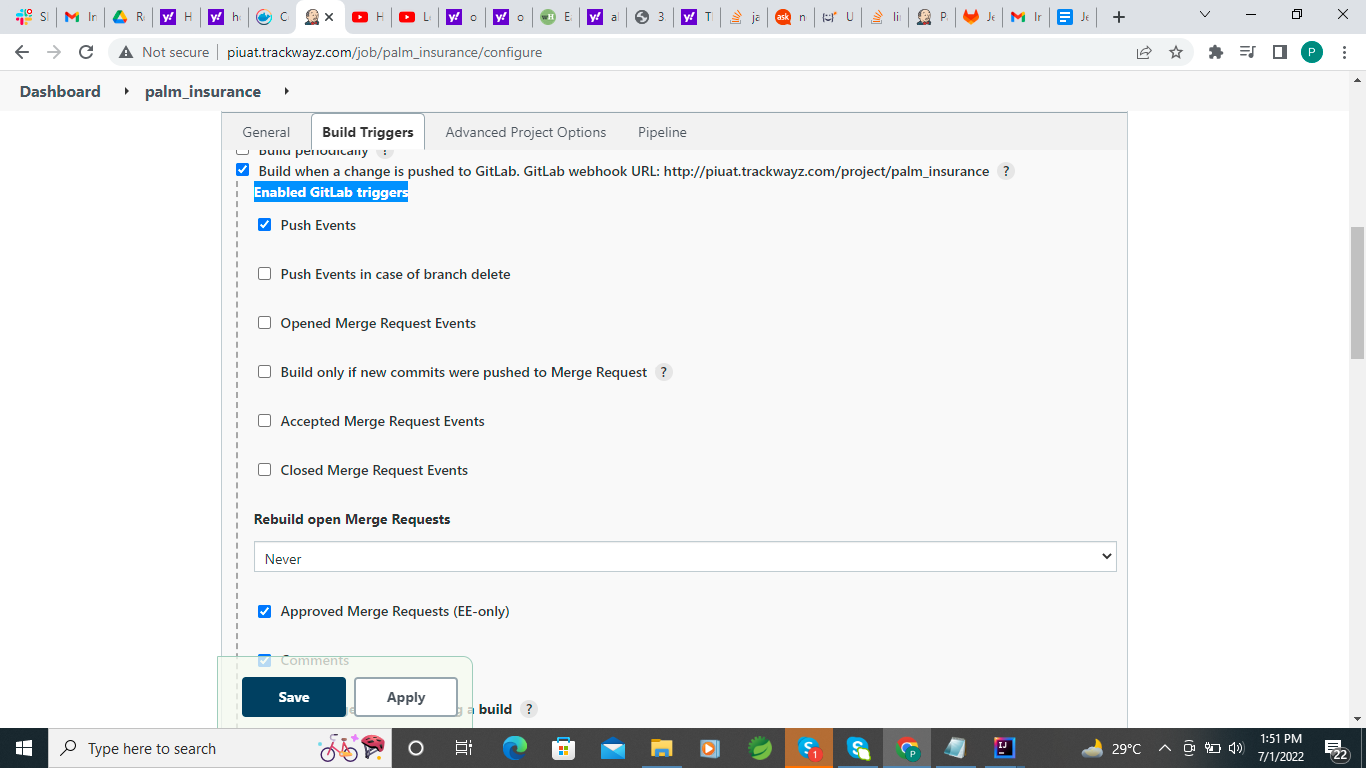
**Use jenkins URl Ex.**[**http://piuat.trackwayz.com/**](http://piuat.trackwayz.com/)

**Enter Jenkins Project Name ex :Palm\_insurance**

**Enter user and password**

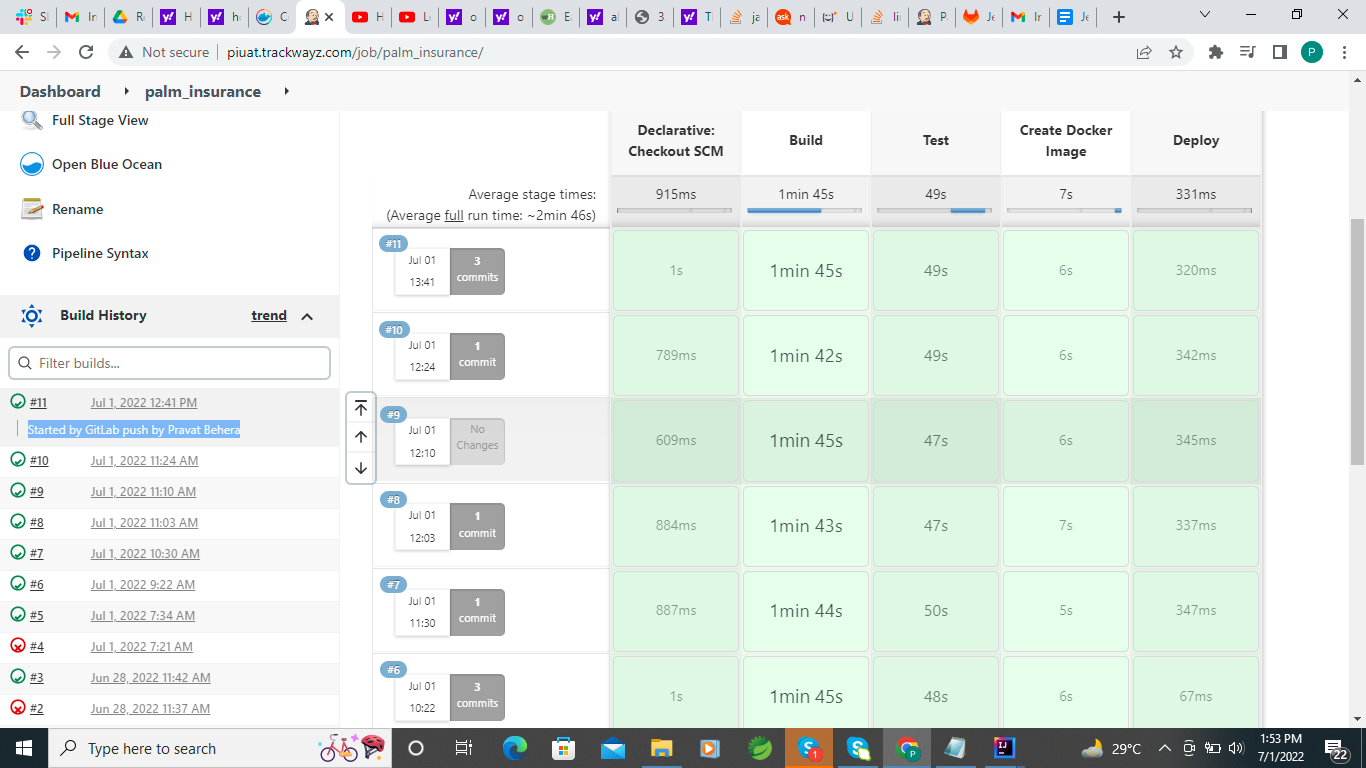
**Jenkins in active state**

**Then got to jenkins and Login jenkins Goto Project →open configure —>**

**Check **

**Apply and save**

**Now push with new change jenkins run automatically,**

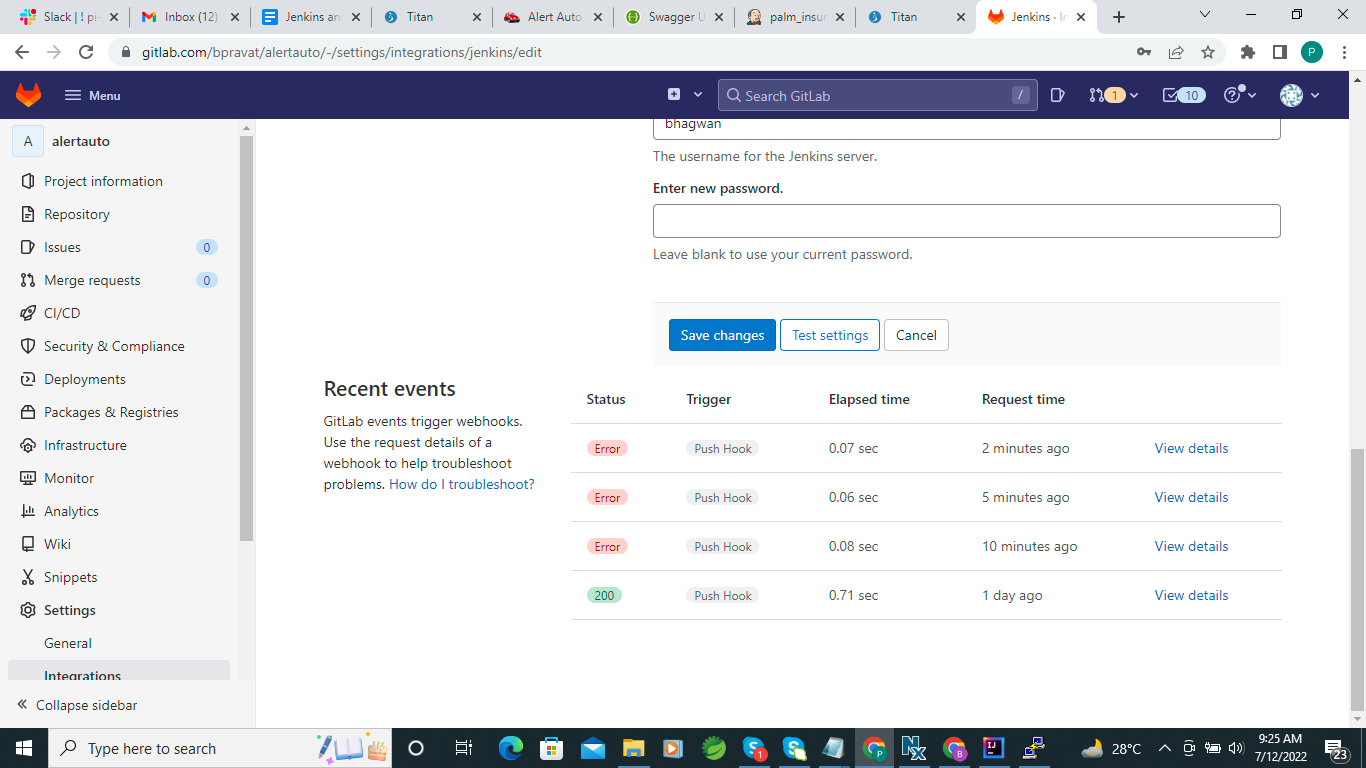
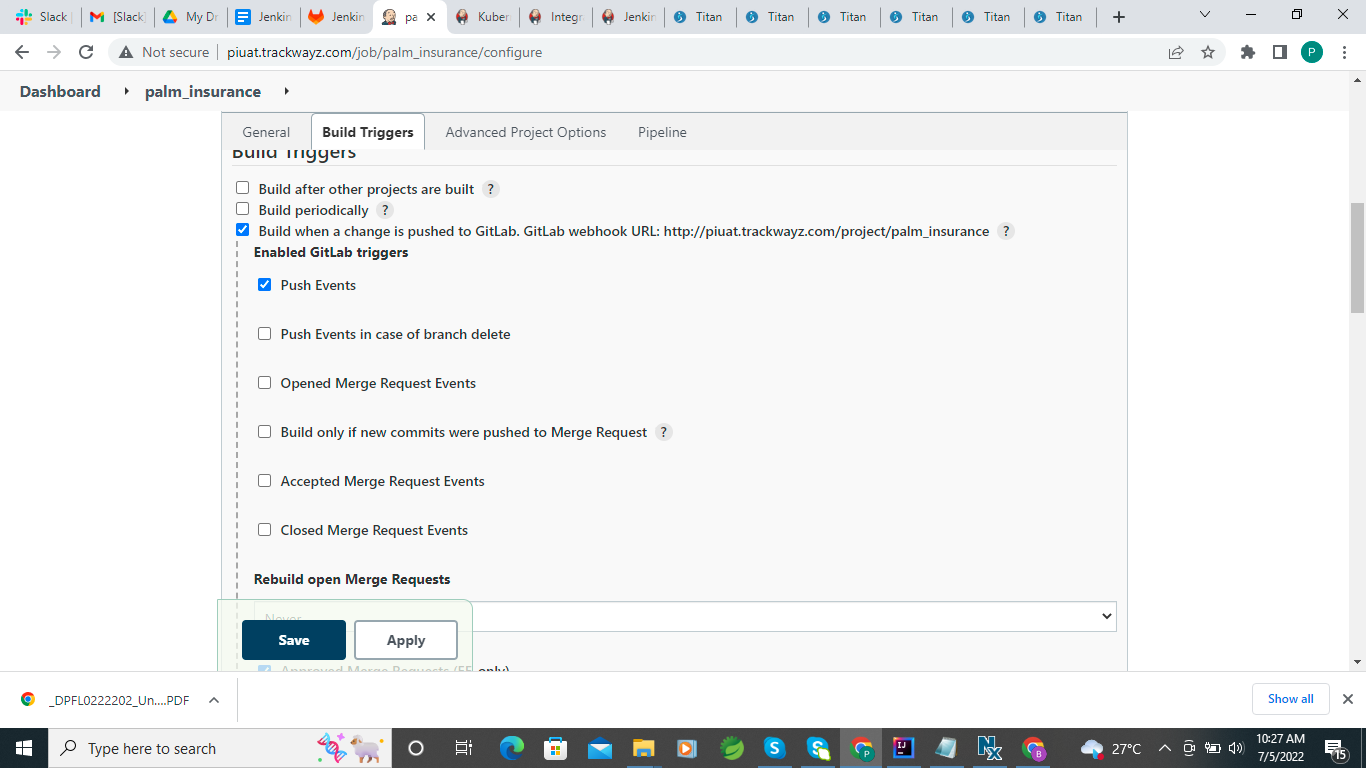
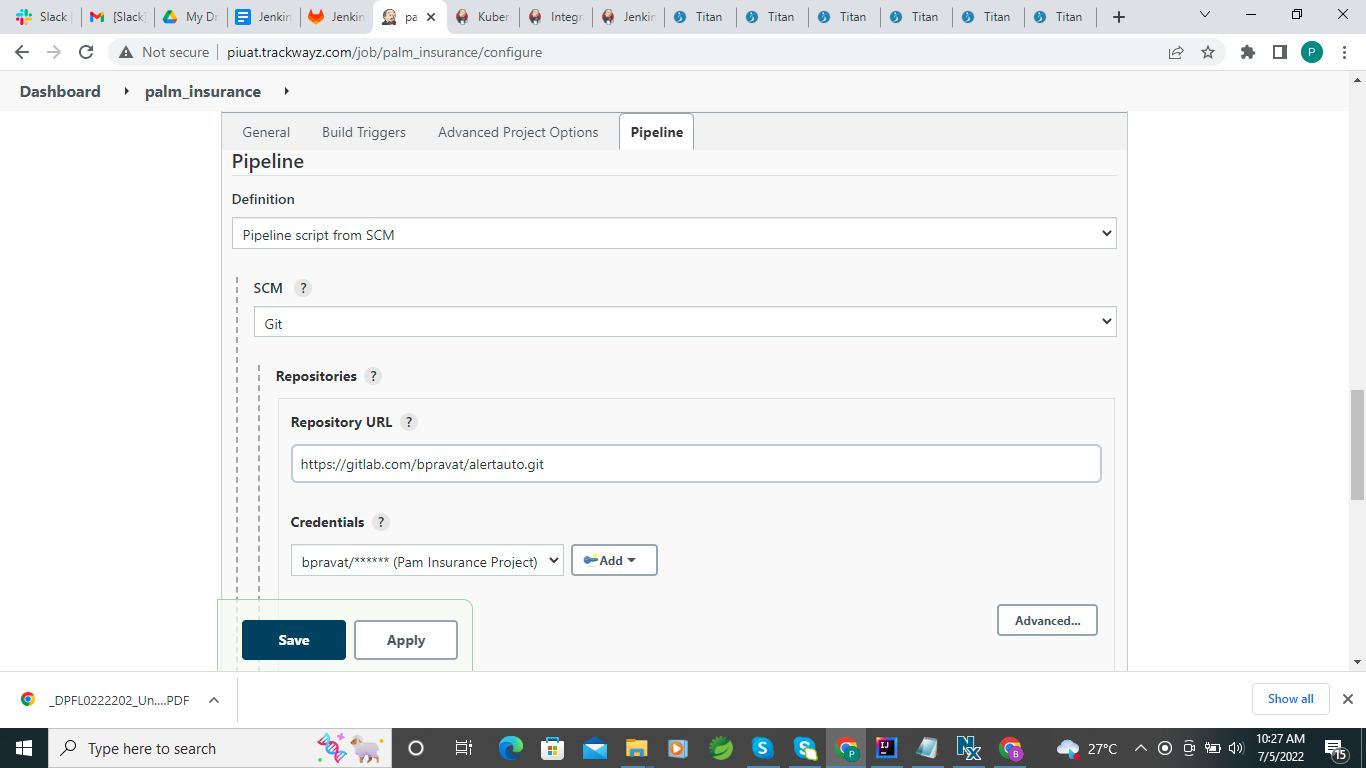
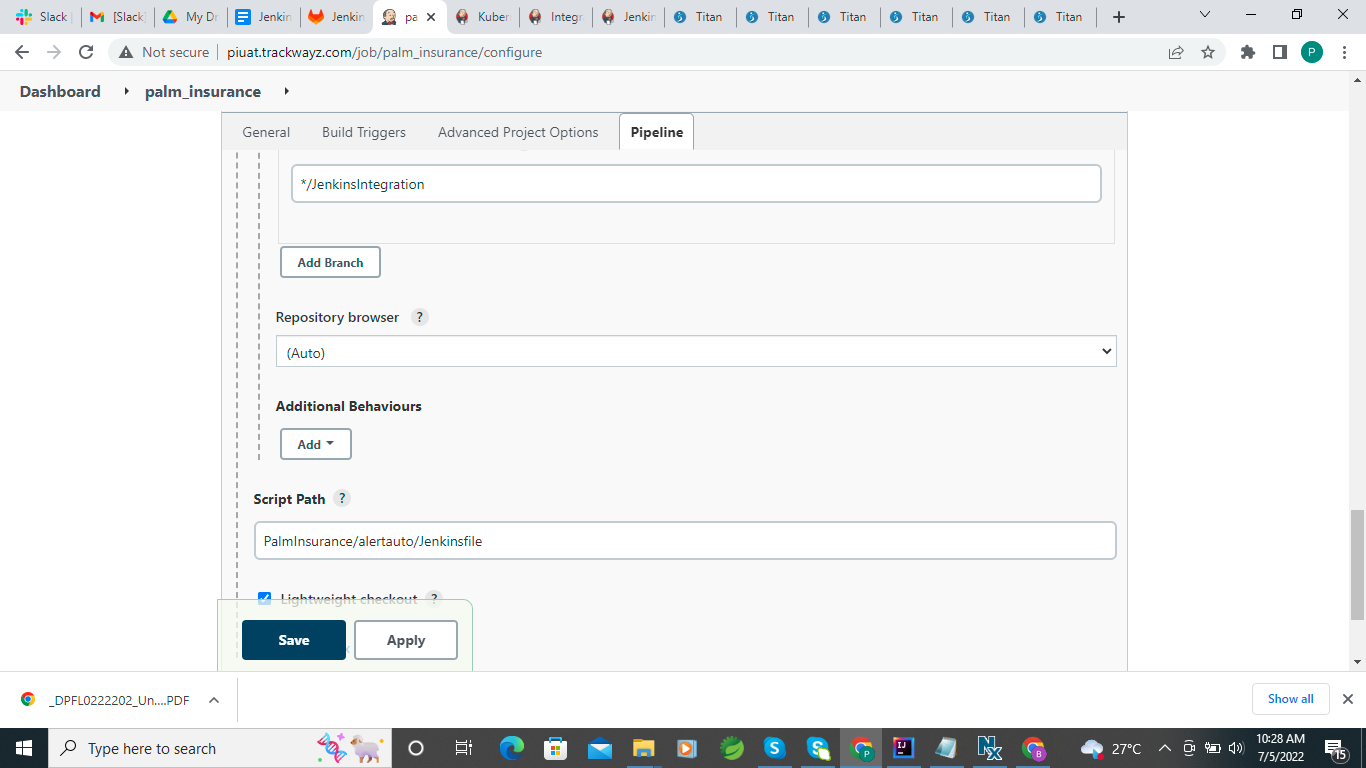
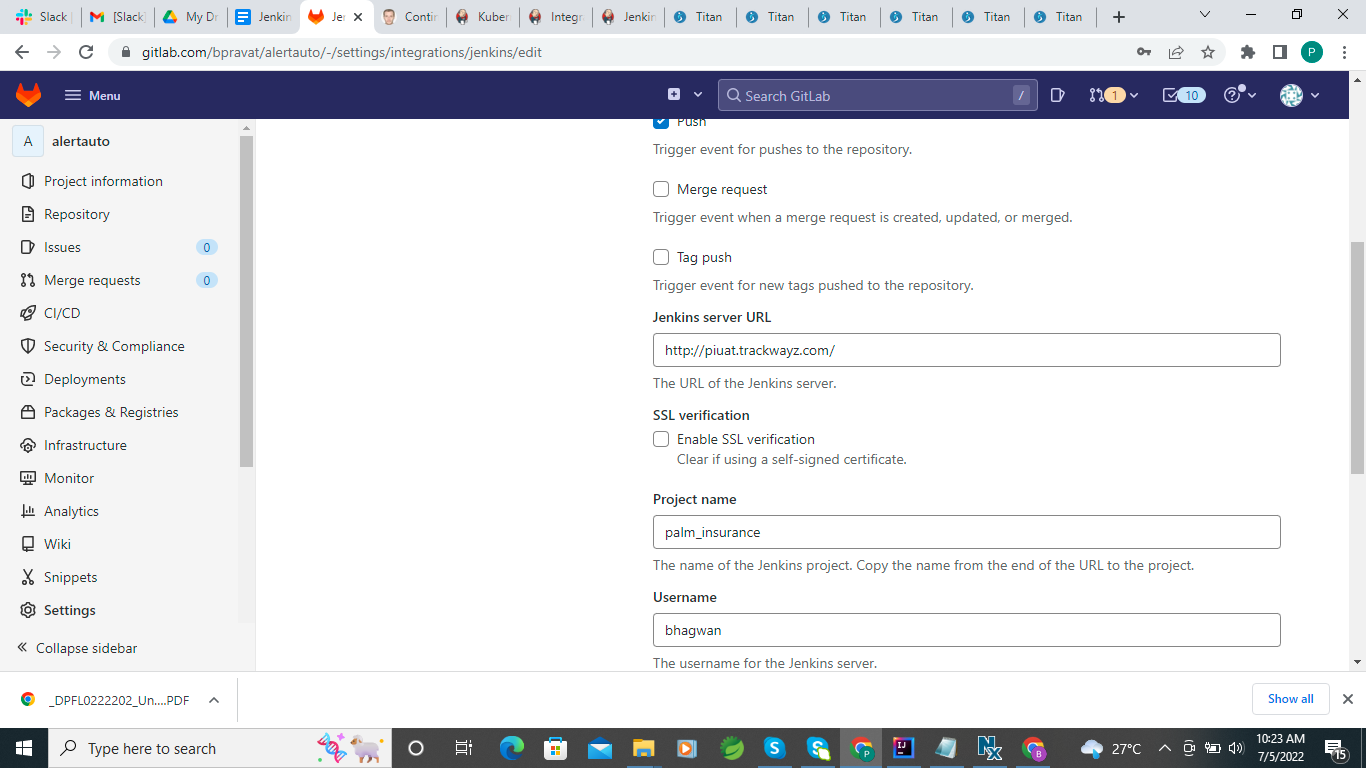
****

**Jenkins and Docker CI/CD Pipeline final flow :-**

[**https://drive.google.com/file/d/1do5Ex9oJgHwBbS2dGtQB2ryCi6Z2Sa8Q/view?usp=sharing**](https://drive.google.com/file/d/1do5Ex9oJgHwBbS2dGtQB2ryCi6Z2Sa8Q/view?usp=sharing)**.**

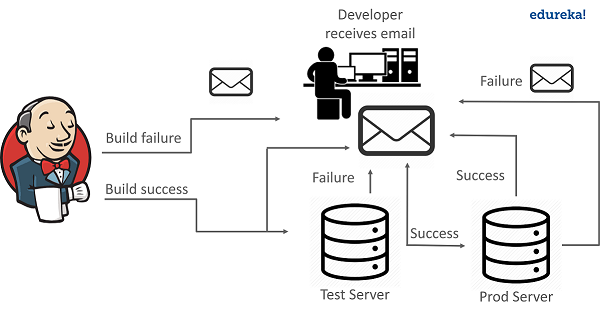
* **I was crating two file inside the module**

**JenkinsFile and DockerFile**

* **JenkinsFile used for writing Jenkins Pipeline script Build—->Testing—–>Creating Image —-->deploy.**
* **DockerFile used for creating docker images for deployment.**
* **When any changes in a particular branch then push to the git.**
* **Inside git i have used webhooks ,When push any code then webhook running jenkins jobs.**
* **Got to the Project repository —---->Settings —--------> Integration —-->Select jenkins CI/CD —---->check active and select push event and enter Jenkins URL and Jenkins Credential.**
* **Then Login Jenkins Create New Item and select Pipeline —>Ok —->Build trigger select git web hook trigger —>Pipeline select CSM script —>Use git repository URL —----> Use Git Credential —--> use git branch which branch use for deploy application —>Use Jenkins file path —->apply —>OK .**
* ****
* ****
* ****
* ****
* **Git Lab webHook**
* ****

**check one thing how we can keep one copy of every success build jar one one directory and on build failed send the mail to the manager ?**

**Email Notification For Failure Job :-**

****

* **This problem was solved by an automation tool called** [**Jenkins**](https://jenkins.io/)**. Jenkins has a service of Email Notifications to handle such situations.**
* **If the build is not successful then the team of developers is notified about the status of the build. This can be done with the help of an Email plugin in Jenkins. Plugins are the primary means of enhancing the functionality of a Jenkins environment to suit organization- or user-specific needs.**
* **Using the email plugin, you configure the email details of the concerned person who should be notified in case of build failure.**
* **Once the developer is notified about the error, he then fixes it and again commits the code to the GitHub. After this Jenkins again pulls the code from GitHub and prepares a fresh build.**
* **Similarly, Jenkins can solve the problem of application going down after the release, by notifying the concerned team, via email.**

## How To Send Email Notification In Jenkins?

**There are basically two ways to configure email notifications in Jenkins.**

1. **Using Email Extension Plugin – This** [**plugin**](https://plugins.jenkins.io/) **lets you configure every aspect of email notifications. You can customize things such as when to send the email, who receives it, and what the email says.**
2. **Using Default Email Notifier – This comes with Jenkins by default. It has a default message consisting of a build number and status.**

**VIDEO**

[**https://www.youtube.com/watch?v=MFgbp00hbVI**](https://www.youtube.com/watch?v=MFgbp00hbVI)

**DOCUMENTS**

**https://www.edureka.co/blog/email-notification-in-jenkins/**

[**https://stackoverflow.com/questions/6273668/how-can-i-backup-file-using-pom-xml#:~:text=You%20can%20run%20ANT%20tasks%20to%20rename%20and,target%2C%20rename%20and%20backup%20the%20current%20packaged%20war**](https://stackoverflow.com/questions/6273668/how-can-i-backup-file-using-pom-xml#:~:text=You%20can%20run%20ANT%20tasks%20to%20rename%20and,target%2C%20rename%20and%20backup%20the%20current%20packaged%20war)**.**

**AutoRestart service in Ubuntu Server :-**

1. **I was creating a script shell file alterrun.sh file**

**Inside the file i have written script code**

**Creating script shell file**

**vim alterrun.sh**

**–Write application running command**

**To save Type Esc then :wq**

**For permission use this command**

**chmod +x pamInsuarnce.sh**

**Run the script from the command line. Type ./pamInsuarnce.sh**

**Runjava.sh**

[**https://fedingo.com/how-to-run-scripts-on-start-up-in-ubuntu/#:~:text=There%20are%20two%20ways%20to%20run%20scripts%20on,following%20command%20to%20make%20rc.local%20file%20an%20executable**](https://fedingo.com/how-to-run-scripts-on-start-up-in-ubuntu/#:~:text=There%20are%20two%20ways%20to%20run%20scripts%20on,following%20command%20to%20make%20rc.local%20file%20an%20executable)**.**

**Auto Start When Restart the server :-**

**https://www.linuxbabe.com/linux-server/how-to-enable-etcrc-local-with-systemd**

* First We have to check the rc.local service available or not if not their then Create rc.local service in the below location in Ubuntu server.
* sudo vim /etc/systemd/system/rc.local.service
* Inside rc.local.service we have to use this code

[Unit]

Description=/etc/rc.local Compatibility

ConditionPathExists=/etc/rc.local

[Service]

Type=forking

ExecStart=/etc/rc.local start

TimeoutSec=0

StandardOutput=tty

RemainAfterExit=yes

SysVStartPriority=99

[Install]

WantedBy=multi-user.target

* Create sudo vim /etc/rc.local

#!/bin/sh

//Write execute command

Java -jar <jar root path>.jar

exit 0

# For permission the file

sudo chmod +x /etc/rc.local

#Enable service

sudo sysetmctl enable /etc/rc.local

Then put your execute code in the rc.local file



When rebooting the server then automatically runs the application.

<https://dev.to/raghwendrasonu/trigger-a-job-with-email-in-jenkins-1554>

Poll Mailbox Trigger