**Industrial Grade Project 1 – Java Project**

**MD Irfan Hussain**

1. **Business Challenge / Requirement**

ABC Technologies is a leading online retail store, and it has recently acquired a large retail offline business store. The business store has a large number of stores across the globe but is following the conventional pattern of development and deployment. As a result, it has landed at a great loss and is facing the following challenges.

* Low available
* Low scalable
* Low performance
* Hard to built and maintain
* Developing and deploying are time consuming.

1. **Goal of the Project**

Implement CI/CD such that ABC Company is able to be –

* Highly available
* Highly scalable
* Highly performant
* Easily built and maintained
* Developed and deployed quickly

1. **Problem statements / Tasks**

We need to develop a CI/CD pipeline to automate the software development, testing, packaging, and deployment, reducing the time to market the app and ensuring good quality service is experienced by end users. In this project, we need to—

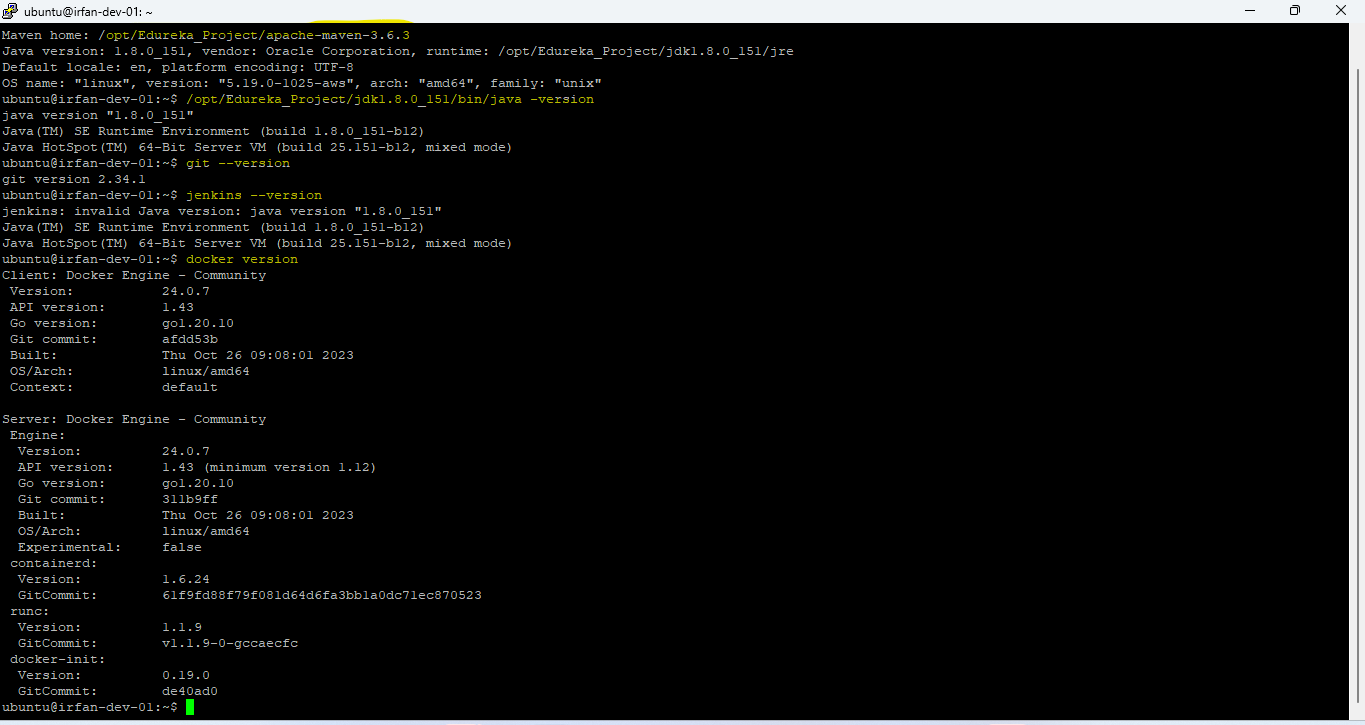
* push the code to our GitHub repository
* create a continuous integration pipeline using Jenkins to compile, test and package the code present in Github.
* Write Dockerfile to push the war file to the Tomcat server.
* Integrate Docker with Ansible and write the playbook.
* Deploy artifacts to the Kubernetes cluster
* Monitor resources using Grafana.

1. **Pre-requisites**

To start the project, we need some pre-requisites which must be installed on the system.

* Java
* Maven
* Git
* Jenkins
* Docker
* Ansible
* Kubernetes
* Grafana

Approach- Installed all the listed software on the server. Please refer to the below screenshot.



A black screen with white text

Description automatically generated

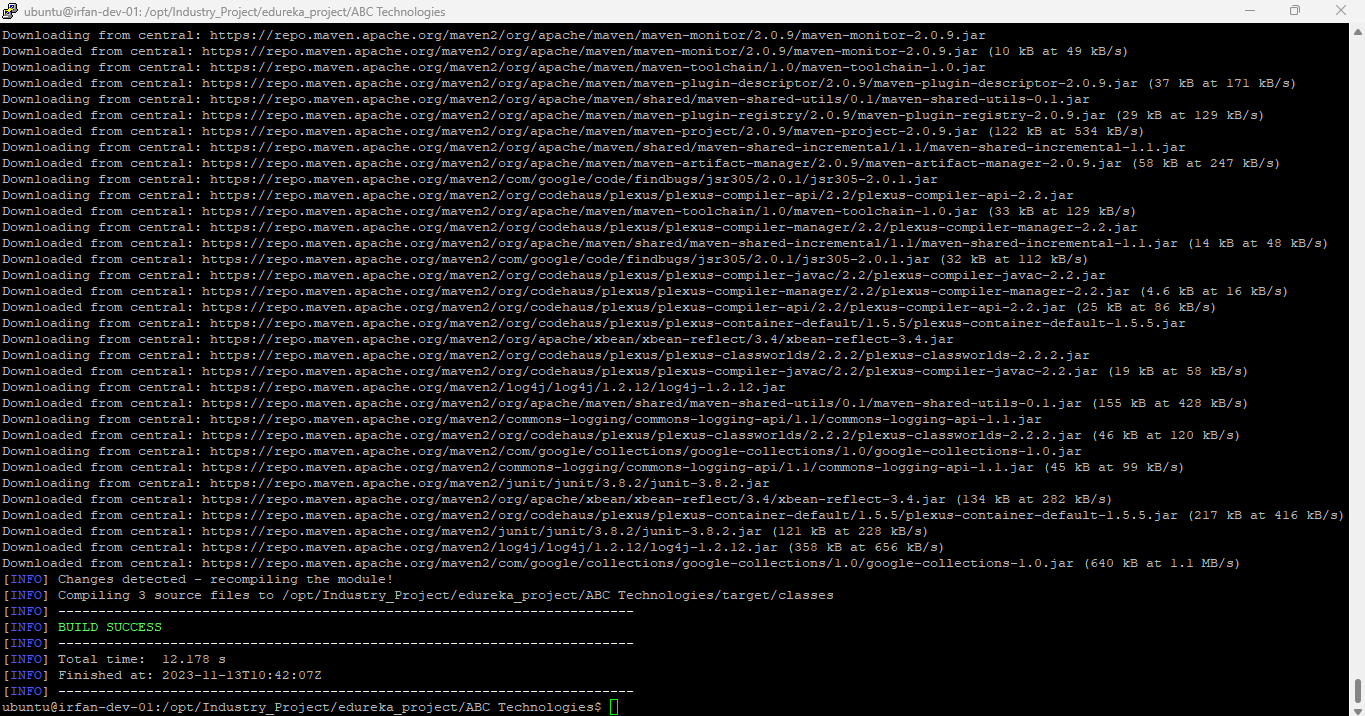
1. **Approach to solve.**

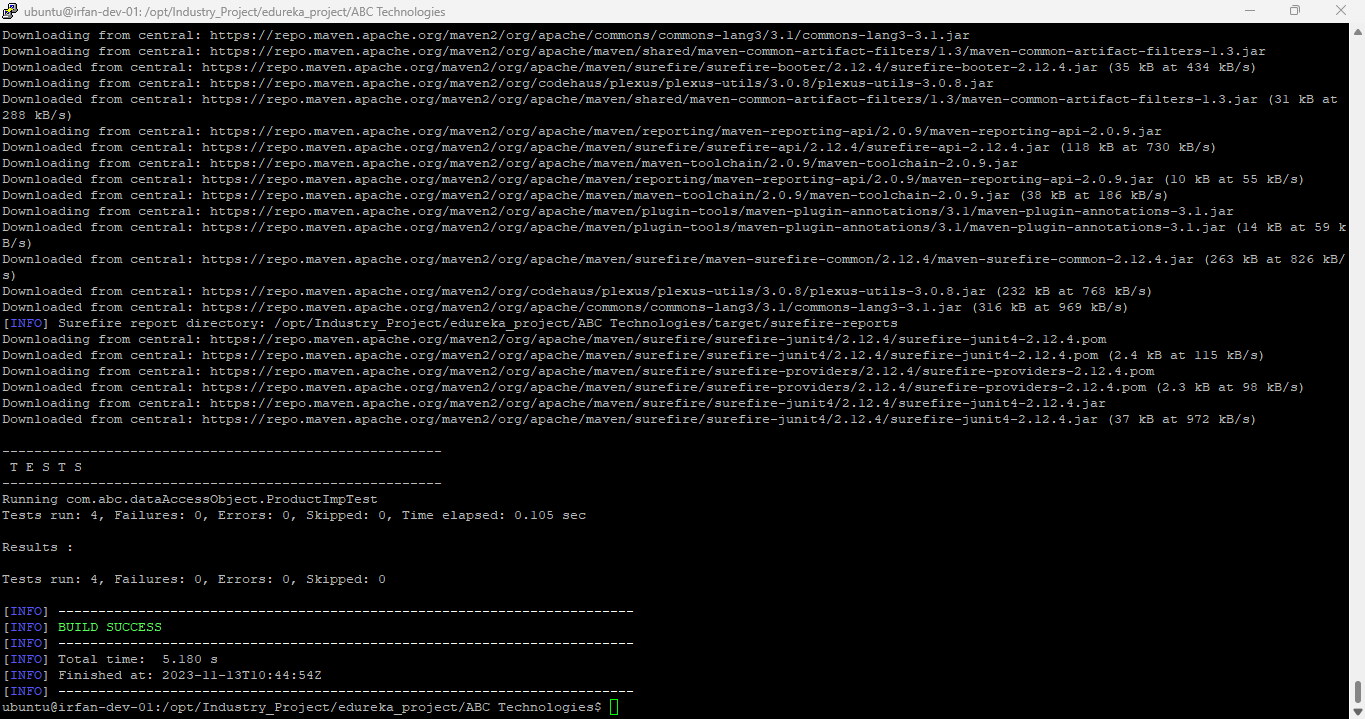
**TASK 1: Clone the project from the GitHub link shared in resources to your local machine. Build the code using Maven commands.**

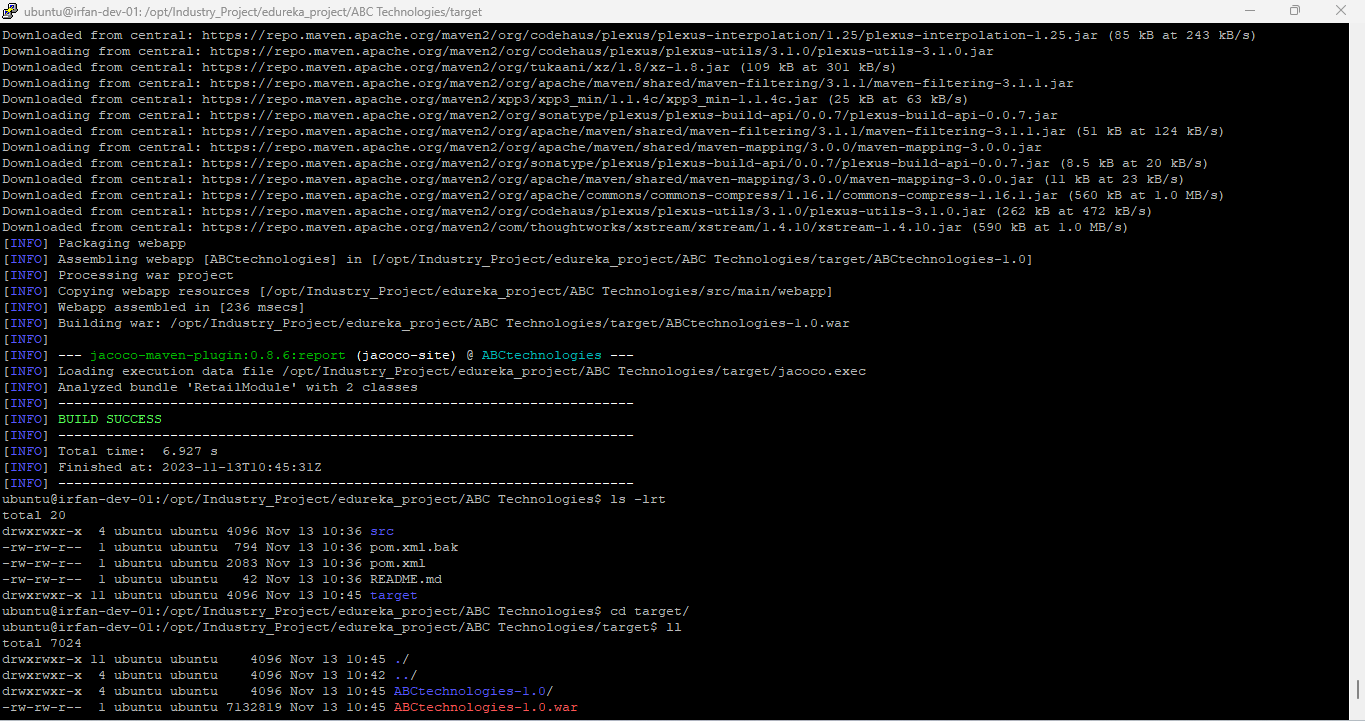
**Approach:** As project is based on java so using maven build tool to build the code as mentioned in the Task1. For this, I have installed Maven too and java on local environment (Ubuntu Distribution).

executed below tasks of maven in local environment and attached screenshot in order.

1. /opt/Industry\_Project/apache-maven-3.6.3/bin/mvn compile
2. /opt/Industry\_Project/apache-maven-3.6.3/bin/mvn test
3. /opt/Industry\_Project/apache-maven-3.6.3/bin/mvn package
4. /opt/Industry\_Project/apache-maven-3.6.3/bin/mvn clean install







**TASK 2: Set up the Git repository and push the source code. Then, log in to Jenkins.**

1. Create a build pipeline containing a job for each

* One for compiling source code
* Second for testing source code
* Third for packing the code

1. Execute the CI/CD pipeline to execute the jobs created in step 1
2. Set up a master-slave node to distribute the tasks in the pipeline.

**Approach:** Installed Jenkins in local environment with master and slave config. And created 3 jobs for compile, test and package the code then created pipelines to execute all 3 jobs and shared load on master and slave agent.

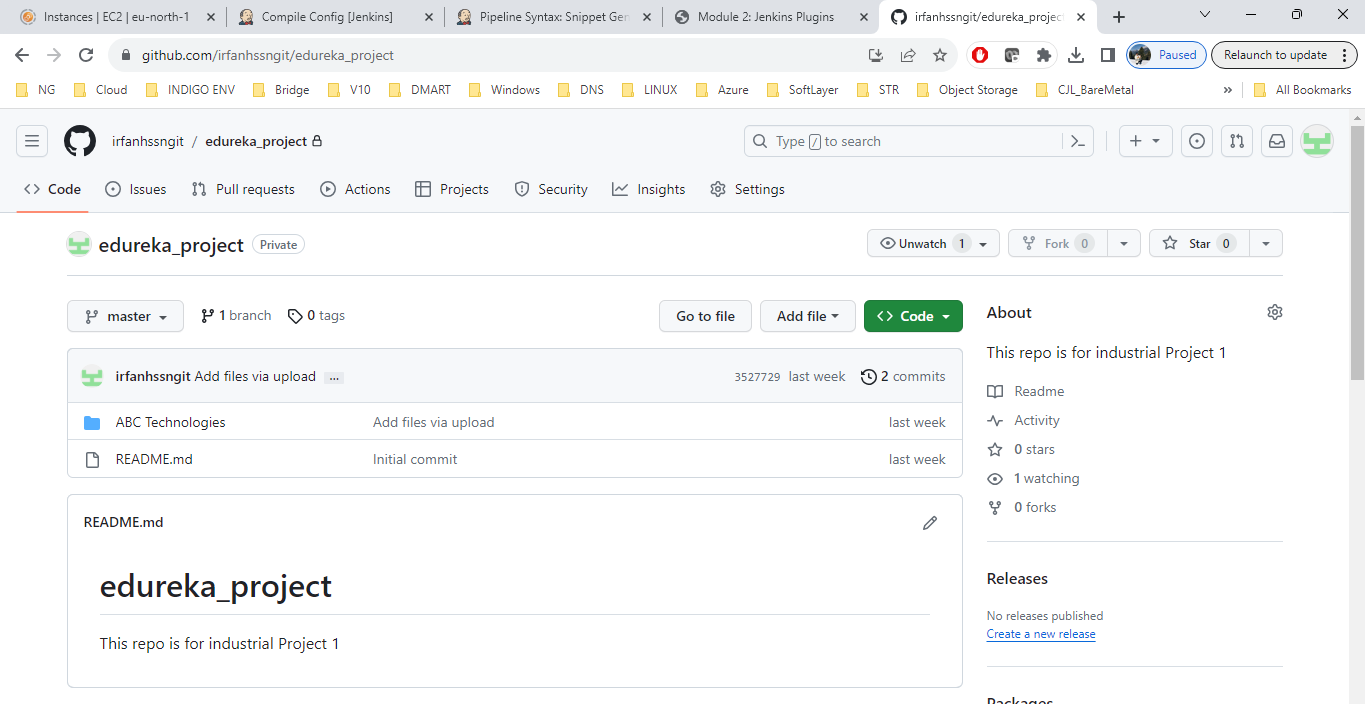
**Tool’s location:**

/opt/Industry\_Project/apache-maven-3.6.3

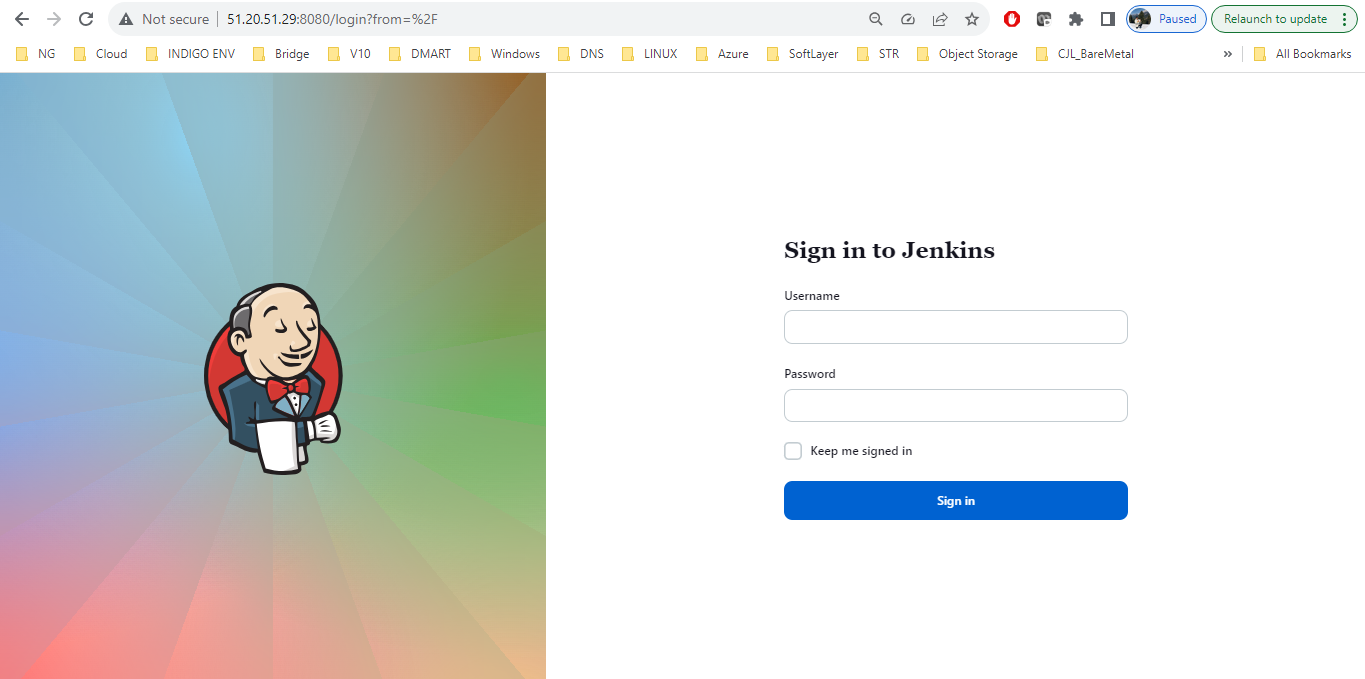
/opt/Industry\_Project/jdk1.8.0\_151

**Please find the screenshots of the Task 2:**

**Git Hub Repo and pushed source code to GitHub-**

****

**Jenkins Login-**



**Java, Maven and Git tools config-**

**A screenshot of a computer

Description automatically generated**

**Maven:**

A close-up of a line

Description automatically generated

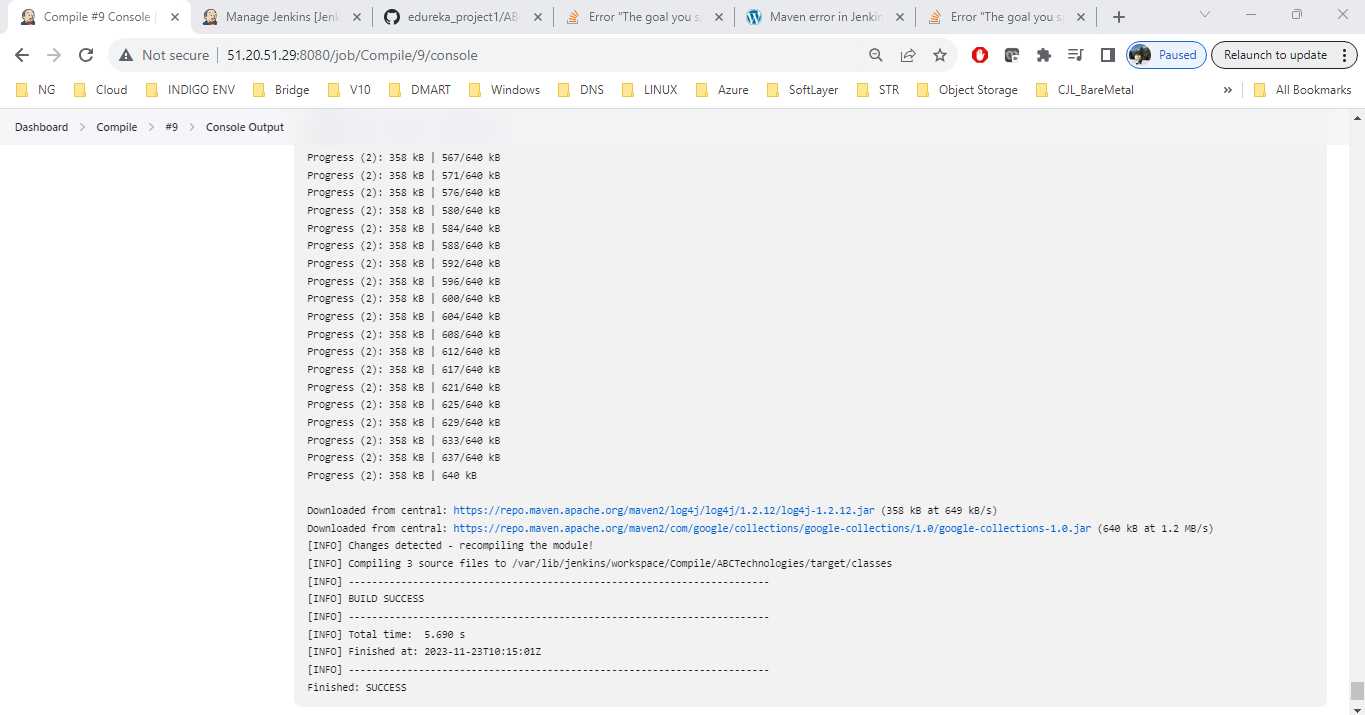
**Git:**

A close-up of a login

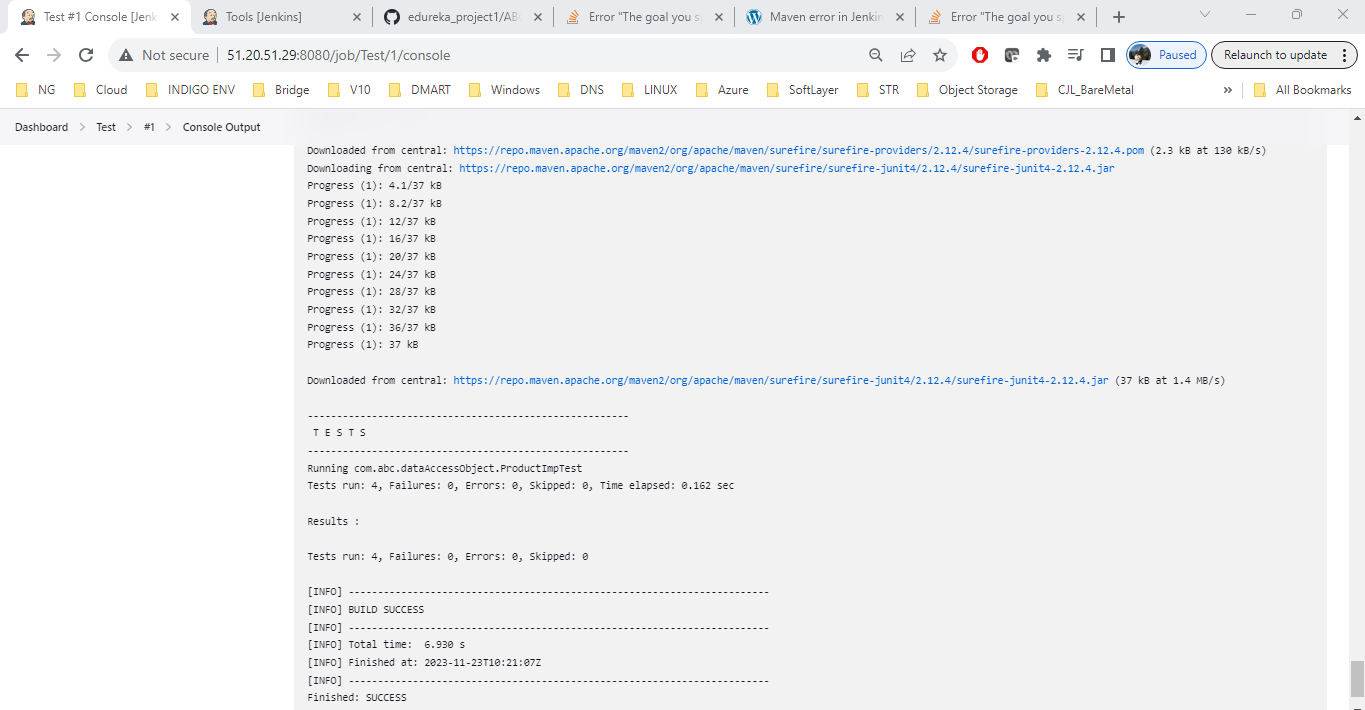
Description automatically generated

**STEP 1-**

**Compile Job result in Jenkins-**

****

**Test Job result in Jenkins-**

****

**Package Job result in Jenkins-**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**STEP 2- Execute the CI/CD pipeline to execute the jobs created in step 1.**

**A screenshot of a computer

Description automatically generated**

**STEP 3- Set up a master-slave node to distribute the tasks in the pipeline.**

Distributed load by assigning Compile task on master node and other two tasks (Test and Package) on slave node.

A screenshot of a computer

Description automatically generated

**TASK 3: Write a Docket file. Create an Image and container on the Docker host. Integrate docker host with Jenkins. Create CI/CD job on Jenkins to build and deploy on a container.**

1. **Enhance the package job created in step 1 of task 2 to create a docker image.**
2. **In the Docker image, add code to move the war file to the Tomcat server and build the image.**

**APPROACH:** Jenkins and docker setup are already there. Now will be creating a docket file to perform the following things.

1. Will create an image to copy the .war to tomcat server and deploy the image on a container.
2. Will be using package job created earlier to integrate the docker and perform build image and deploy the image on container.

**STEP 1: Will create an image to copy the .war to tomcat server and deploy the image on a container.**

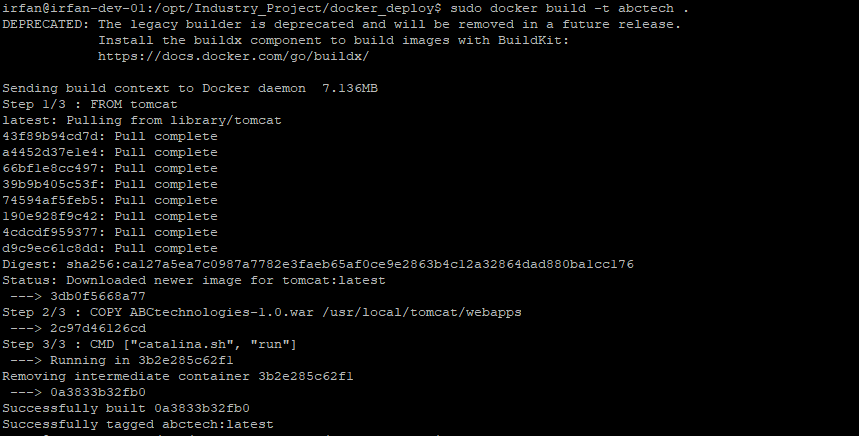
Dockerfile-

FROM tomcat

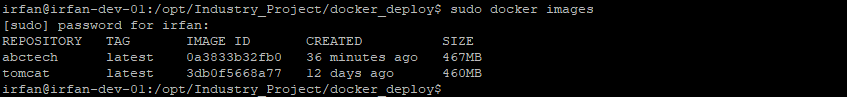
COPY ABCtechnologies-1.0.war /usr/local/tomcat/webapps

CMD ["catalina.sh", "run"]

Building an Image-



Deploying an image on a container-





****

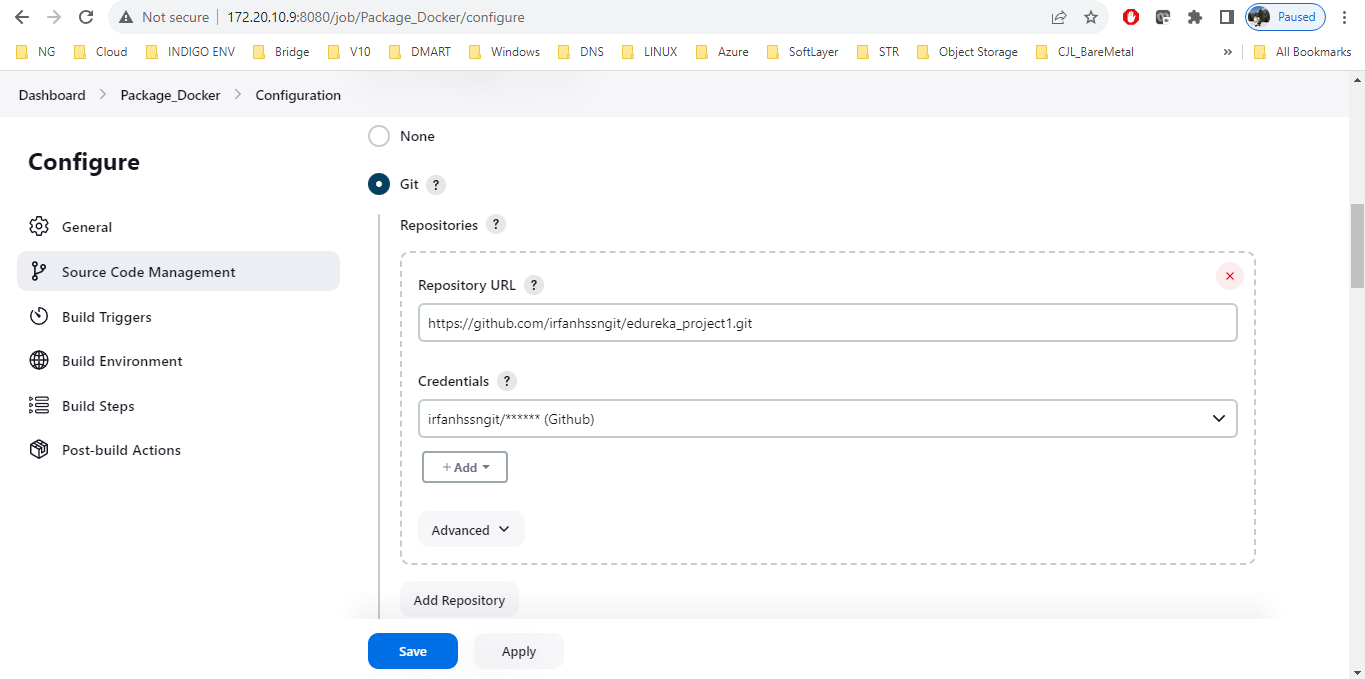
**A screenshot of a computer

Description automatically generated**

**STEP 2:** Will be using package job created earlier to integrate the docker and perform build image and deploy the image on container.

Below screenshot shows the Jenkins job which is pulling the code from Github then packaging the code into \*.war file and then using a docket file to deploy that war using tomcat on docker container post building an image and pushing an image to docker hub.

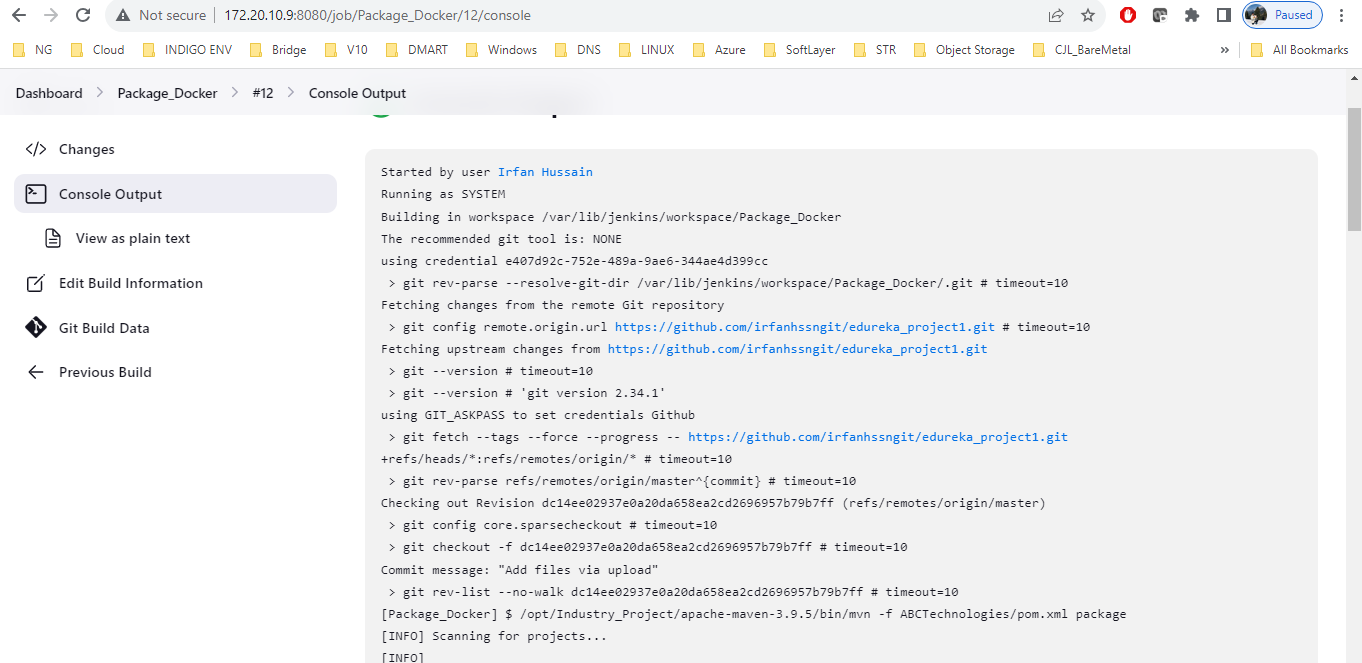
Configuration-



A screenshot of a computer

Description automatically generated

Git pulling -



Packaging Job-

A screenshot of a computer

Description automatically generated

Building an image, pushing to docker hub and deploying an image to container-

A screenshot of a computer program

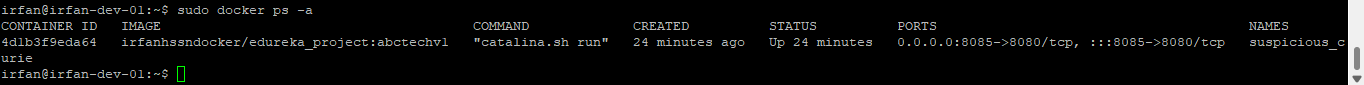
Description automatically generated

Pushed image to docker hub-

A screenshot of a computer

Description automatically generated

Validation of deployed image on container-



A screenshot of a computer

Description automatically generated