**Build a Docker Jenkins Pipeline to Implement CI/CD Workflow**

DESCRIPTION

Demonstrate the continuous integration and delivery by building a Docker Jenkins Pipeline.

**Problem Statement Scenario:**

You are a DevOps consultant in AchiStar Technologies. The company decided to implement DevOps to develop and deliver their products. Since it is an Agile organization, it follows Scrum methodology to develop the projects incrementally. You are working with multiple DevOps Engineers to build a Docker Jenkins Pipeline. During the sprint planning, you agreed to take the lead on this project and plan on the requirements, system configurations, and track the efficiency. The tasks you are responsible for:

* Availability of the application and its versions in the GitHub
  + Track their versions every time a code is committed to the repository
* Create a Docker Jenkins Pipeline that will create a Docker image from the Dockerfile and host it on Docker Hub
* It should also pull the Docker image and run it as a Docker container
* Build the Docker Jenkins Pipeline to demonstrate the continuous integration and continuous delivery workflow

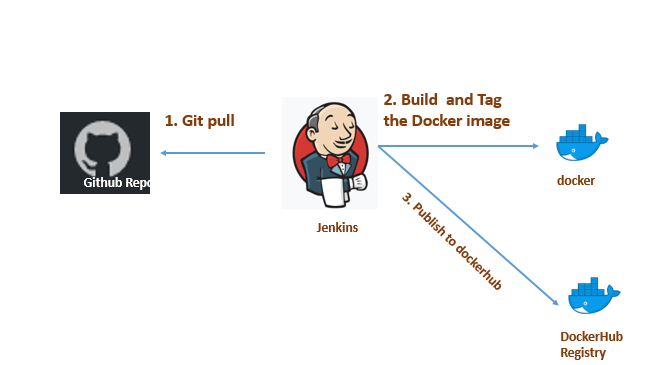
Company goal is to deliver the product frequently to the production with high-end quality.

**You must use the following tools:**

* Docker: To build the application from a Dockerfile and push it to Docker Hub
* Docker Hub: To store the Docker image
* GitHub: To store the application code and track its revisions
* Git: To connect and push files from the local system to GitHub
* Linux (Ubuntu): As a base operating system to start and execute the project
* Jenkins: To automate the deployment process during continuous integration

**Following requirements should be met:**

* Document the step-by-step process from the initial installation to the final stage
* Track the versions of the code in the GitHub repository
* Availability of the application in the Docker Hub
* Track the build status of Jenkins for every increment of the project



In this project, we will build the docker image and publish it on the Dockerhub using Jenkins Pipeline. We will create an automated CI/CD pipeline which will build the docker image, publish it and deploy it on the server.

**Required Tools on server**

**Few of the tools which we will require during this demonstration.**

* GitHub
* Git
* Jenkins
* Docker

We will be using **Amazon Linux Ec2** instance to create and test our solution for the given problem statement.

### **Download and install Jenkins**

To download and install Jenkins:

* To ensure that your software packages are up to date on your instance, use the following command to perform a quick software update:

[ec2-user ~]$ **sudo yum update –y**

* Add the Jenkins repo using the following command:

[ec2-user ~]$ **sudo wget -O /etc/yum.repos.d/jenkins.repo \**

**https://pkg.jenkins.io/redhat-stable/jenkins.repo**

* Import a key file from Jenkins-CI to enable installation from the package:

[ec2-user ~]$ **sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key**

[ec2-user ~]$ **sudo yum upgrade**

* Install Jenkins:

[ec2-user ~]$ **sudo yum install jenkins java-1.8.0-openjdk-devel -y**

[ec2-user ~]$ **sudo systemctl daemon-reload**

* Start Jenkins as a service:

[ec2-user ~]$ **sudo systemctl start jenkins**

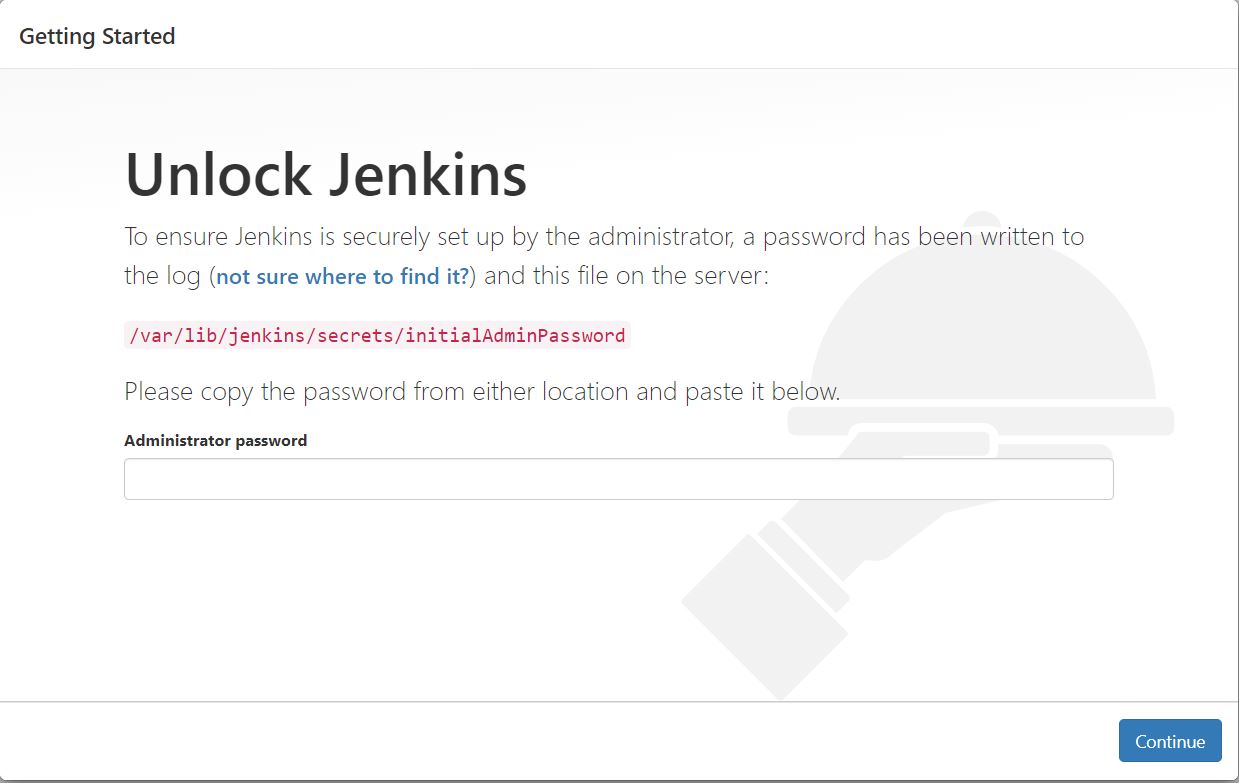
You can check the status of the Jenkins service using the command:

[ec2-user ~]$ **sudo systemctl status Jenkins**

### **Configure Jenkins**

Jenkins is now installed and running on your EC2 instance. To configure Jenkins:

* Connect to http://<your\_server\_public\_DNS>:8080 from your favorite browser. You will be able to access Jenkins through its management interface:



* As prompted, enter the password found in **/var/lib/jenkins/secrets/initialAdminPassword**.

Use the following command to display this password:

[ec2-user ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

* The Jenkins installation script directs you to the **Customize Jenkins page**. Click **Install suggested plugins**.
* Once the installation is complete, **Create First Admin User**, click **Save and Continue**.



### **Download and install Docker**

To download and install Docker:

* To ensure that your software packages are up to date on your instance, use the following command to perform a quick software update:

[ec2-user ~]$ **sudo yum update –y**

* Install the most recent Docker Engine package Amazon Linux 2:

[ec2-user ~]$ **sudo amazon-linux-extras install docker**

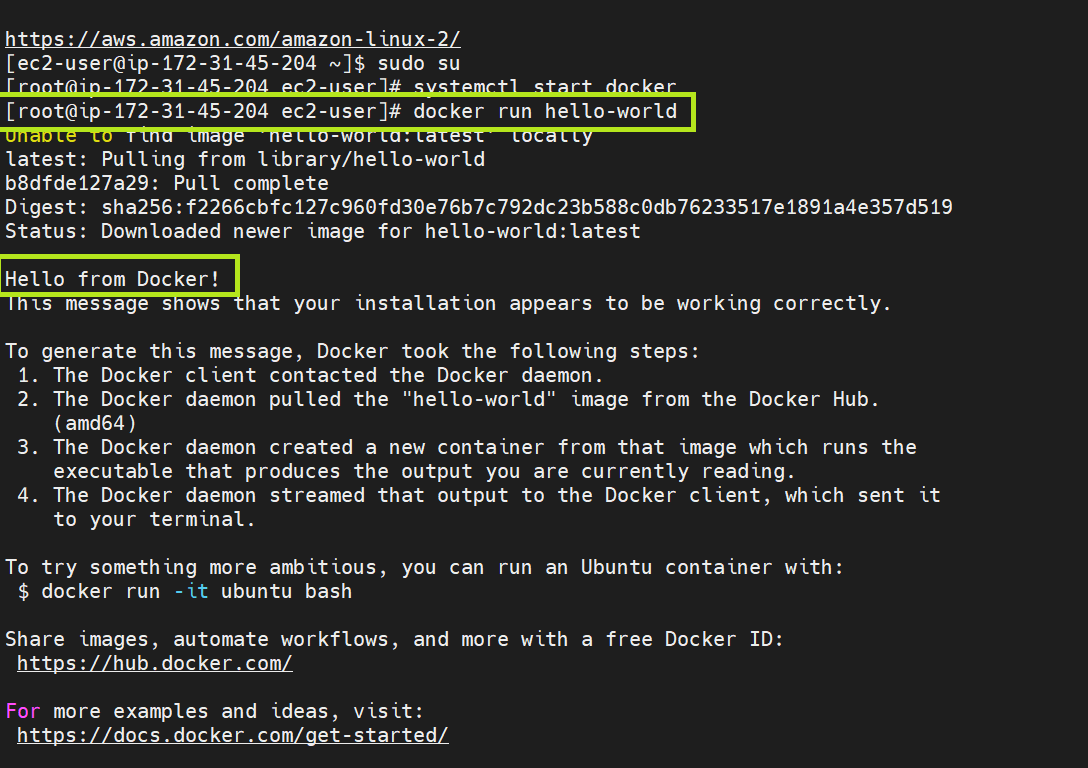
* Start the Docker service:

[ec2-user ~]$ **sudo service docker start**

* To test the installation run the command:

[ec2-user ~]$ **docker run hello-world**

After running the command if you get **Hello from Docker!** Then your installation was successful.



### **Download and install Git**

To download and install Git:

* To ensure that your software packages are up to date on your instance, use the following command to perform a quick software update:

[ec2-user ~]$ **sudo yum update –y**

* Install the most recent Git:

[ec2-user ~]$ **sudo yum install git -y**

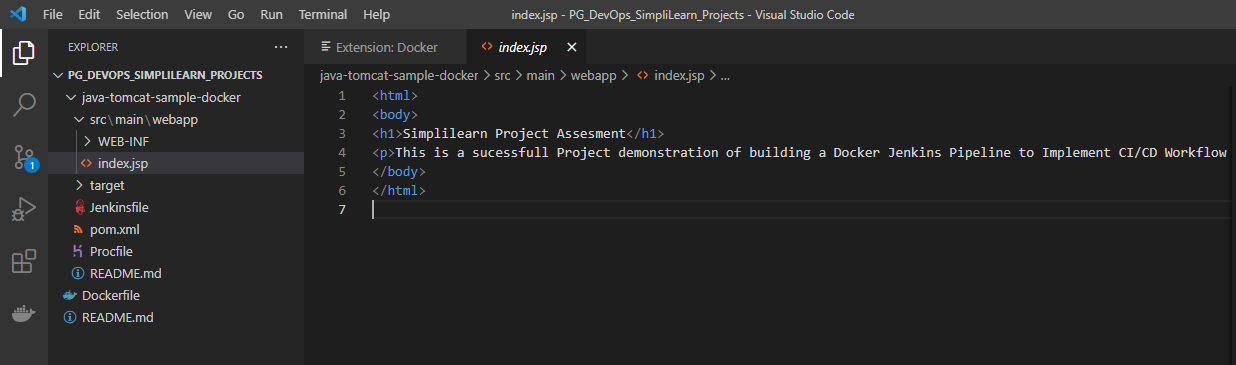
* Verify the installation:

[ec2-user ~]$ **git --version**

### **Create a sample Java Webapp project for deploying to tomcat on Docker Image**

**GitHub Project Link:**

https://github.com/bipin115/PG\_DevOps\_SimpliLearn\_Projects.git



### **Create Tomcat Image from a Dockerfile**

Create a file with name **Dockerfile**, the file will have no extension

Write these commands inside the Dockerfile

FROM tomcat:8.0

# Maintainer

MAINTAINER "Bipin Singh"

# COPY path-to-your-application-war path-to-webapps-in-docker-tomcat

COPY ./java-tomcat-sample-docker/target/java-tomcat-maven-example.war /usr/local/tomcat/webapps

EXPOSE 8080

CMD ["catalina.sh", "run"]

### **Write Jenkins file to create a CI-CD pipeline with VCS control**

Create a file with name **Jenkinsfile**, the file will have no extension

pipeline {

    agent any

    stages {

        stage('Initialize Git POLL SCM Feature') {

            steps {

                //enable remote triggers

                script {

                    properties([[$class: 'GithubProjectProperty', displayName: '', projectUrlStr: 'https://github.com/bipin115/PG\_DevOps\_SimpliLearn\_Projects.git/'], pipelineTriggers([pollSCM('\* \* \* \* \*')])])

                }

            }

        }

    stage('Git project checkout') {

           steps {

                git branch: 'main', url: 'https://github.com/bipin115/PG\_DevOps\_SimpliLearn\_Projects.git'

          }

        }

        stage('Build Application') {

            steps {

                sh 'mvn -f java-tomcat-sample-docker/pom.xml clean package'

            }

            post {

                success {

                    echo "Now Archiving the Artifacts...."

                    archiveArtifacts artifacts: '\*\*/\*.war'

                }

            }

        }

        stage('Docker Build and Tag') {

           steps {

                sh 'docker build -t samplewebapp:latest .'

                sh 'docker tag samplewebapp bipin115/samplewebapp:latest'

          }

        }

        stage('Publish image to Docker Hub') {

            steps {

                    withDockerRegistry([ credentialsId: "dockerHub", url: "" ]) {

                    sh  'docker push bipin115/samplewebapp:latest'

                    }

            }

        }

      stage('Run Docker container') {

            steps {

                sh "docker run -d -p 9090:8080 bipin115/samplewebapp"

            }

        }

    }

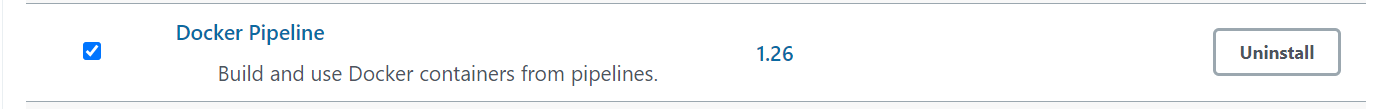
}

Job the above pipeline command will do:

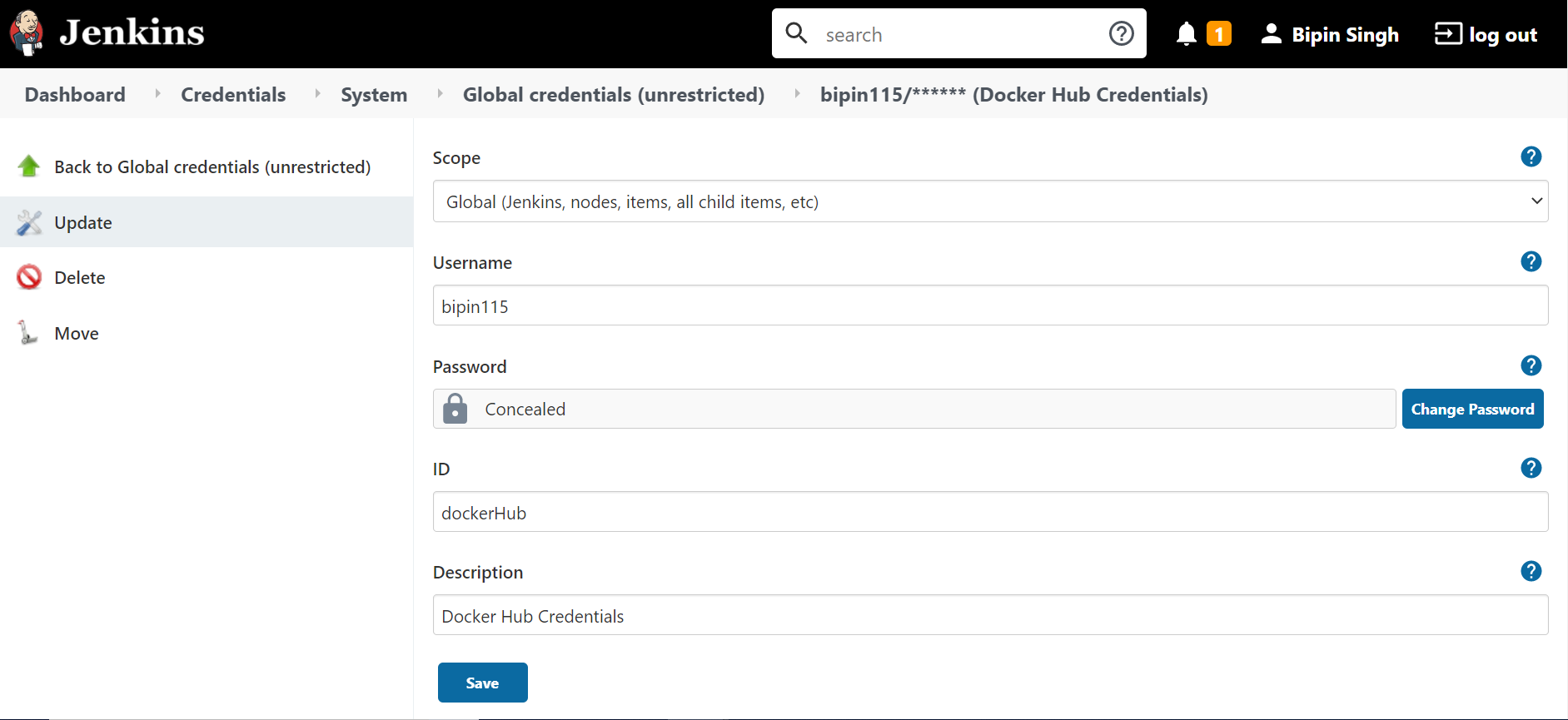
* Enabling Git poll SCM feature for every minute
* Project checkout from GitHub
* Generate the artefacts from the checked out project
* Build Docker Image from Dockerfile and the Tag the Image
* Publish the new Image over Docker Hub
* Pull the Image from Docker Hub and run as container over port 9090

### **Setting up Jenkins to build Pipeline Job for CI-CD**

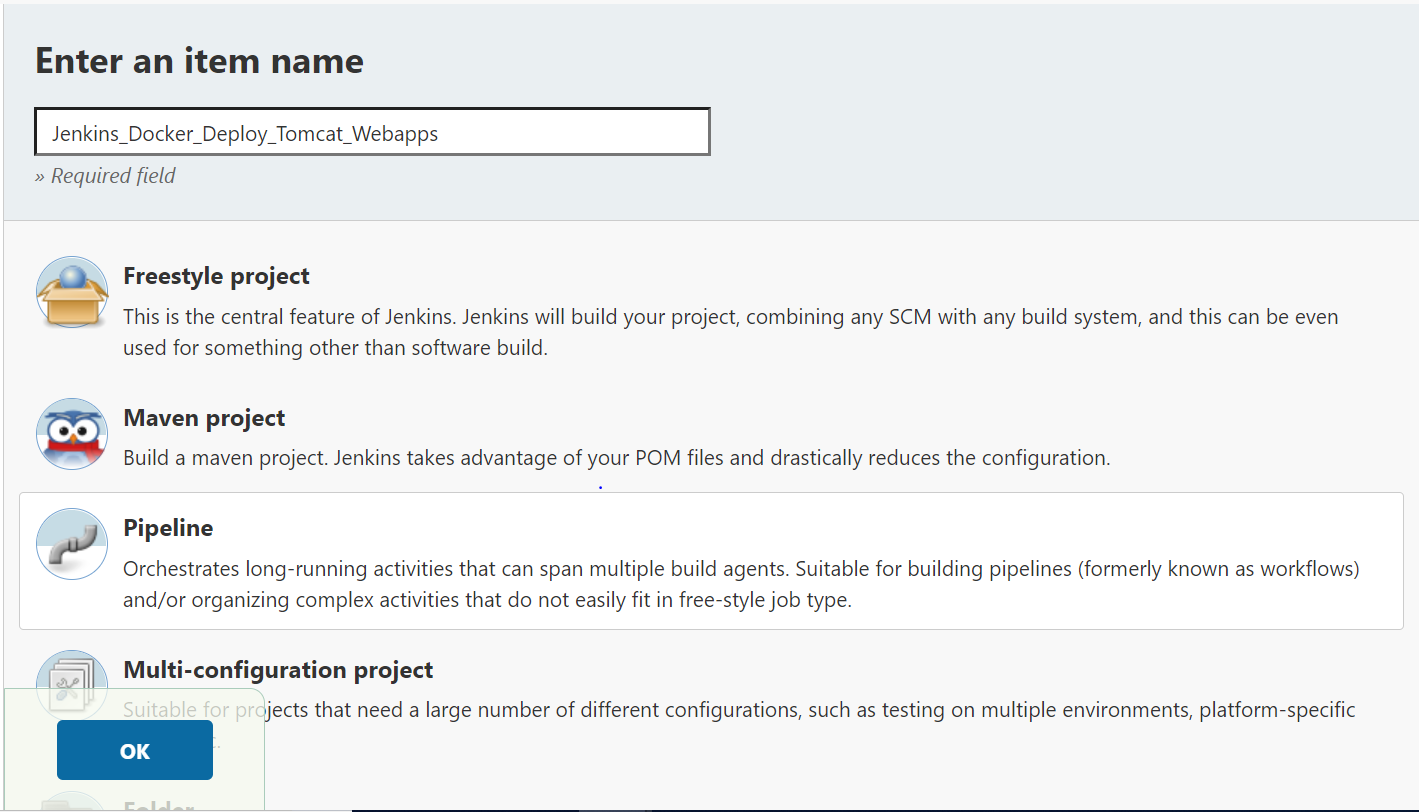
* Installing Docker Pipeline Plugin



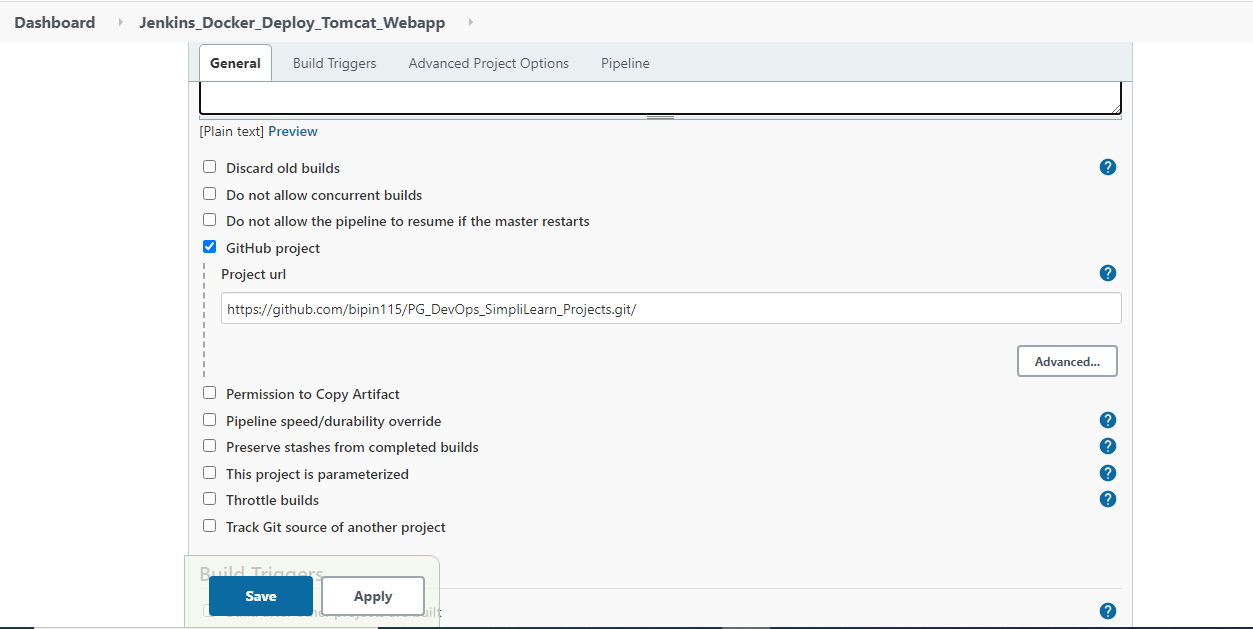
* Add Docker Hub credentials to the credentials manager



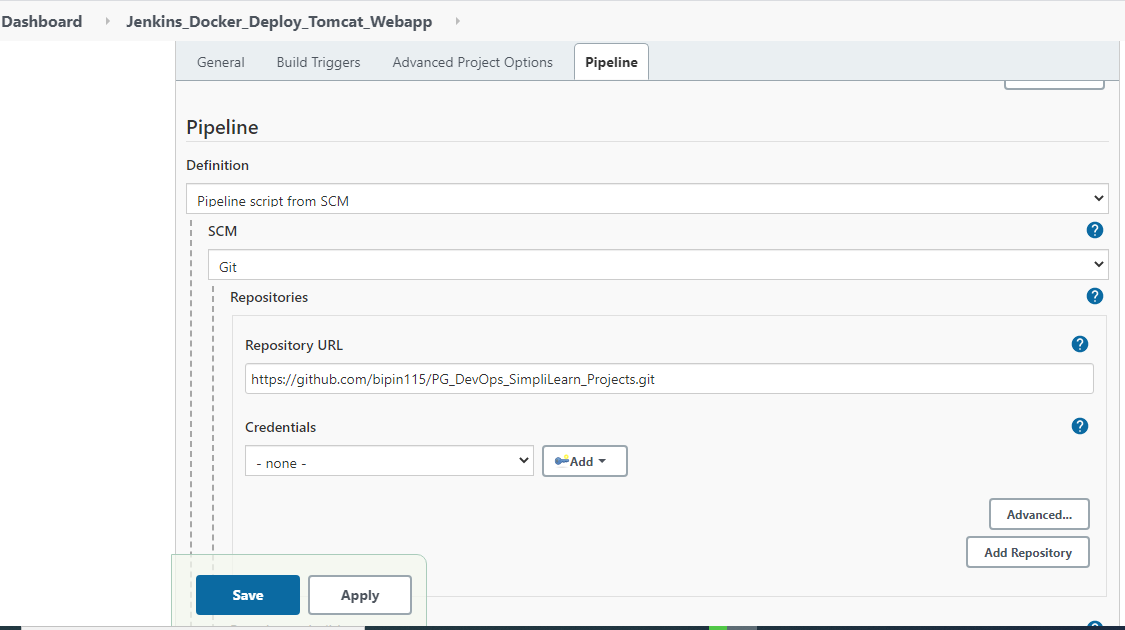
### **Creating Jenkins Job for CI-CD**

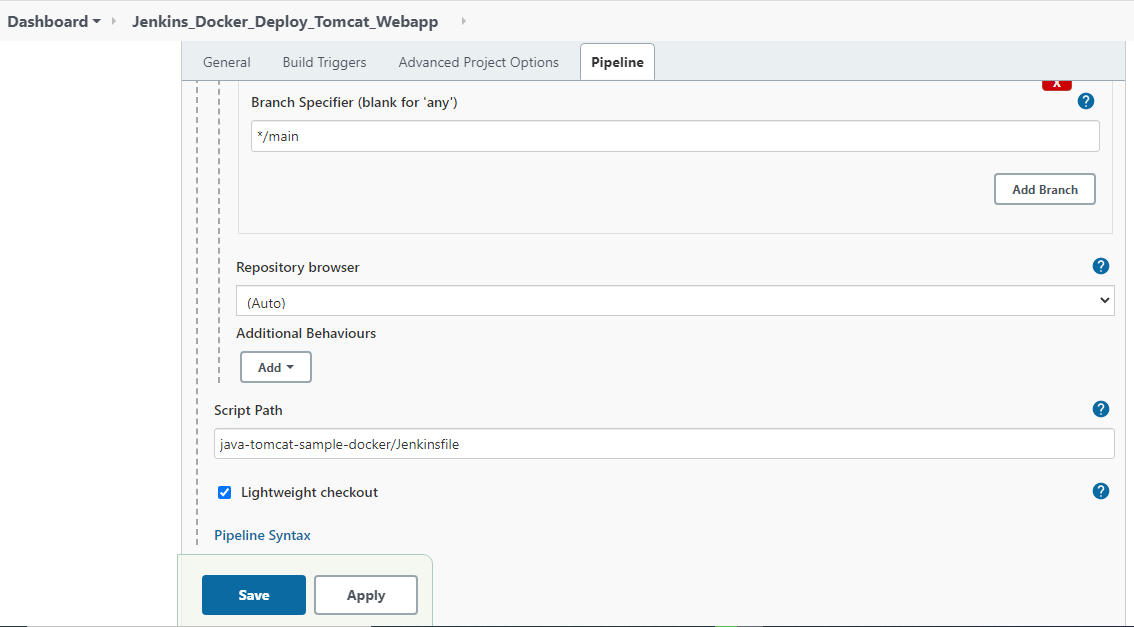


Adding Git hub project URL.

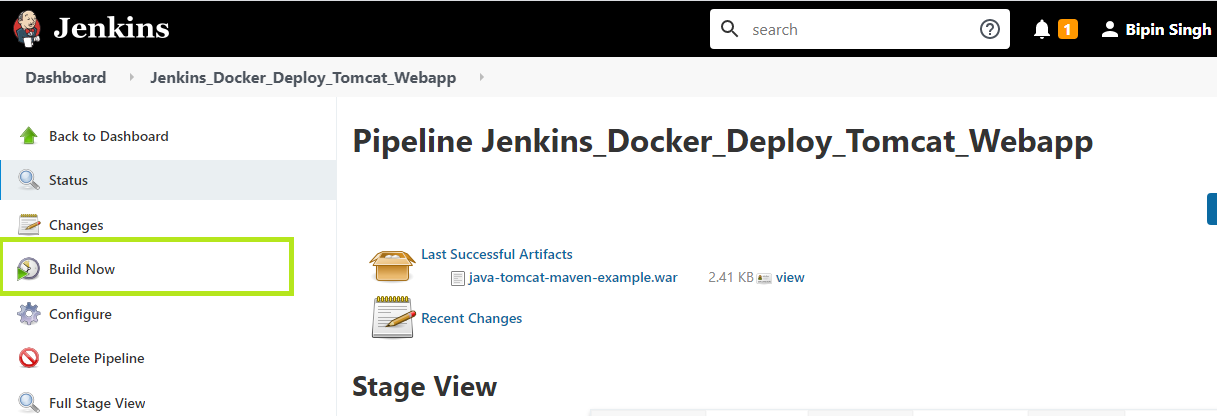


Set Pipeline using a Jenkins file path

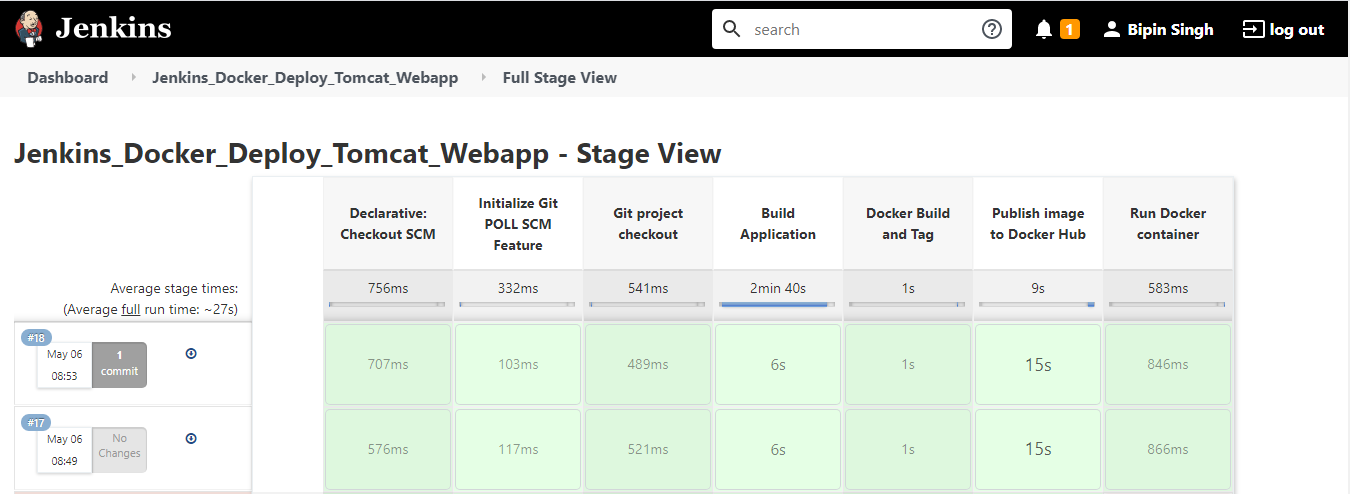




Once details are set save the job. Once the Job is saved click on Build now option.

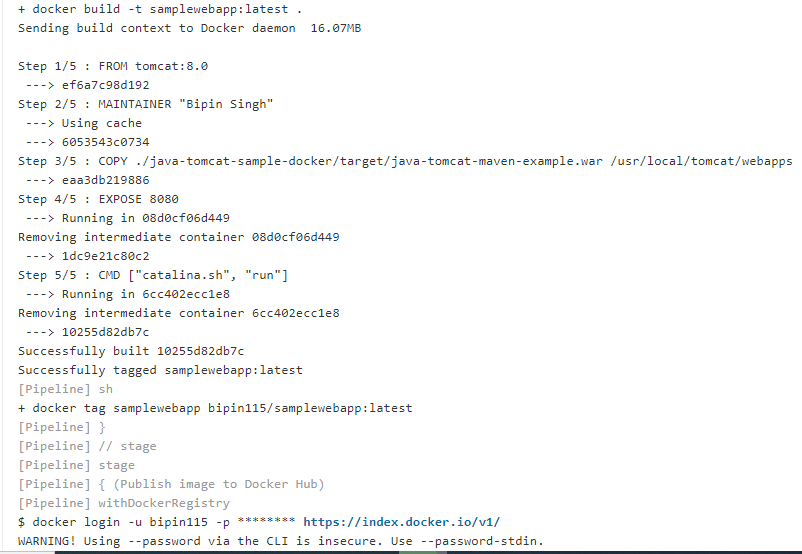


Pipeline job execution build output

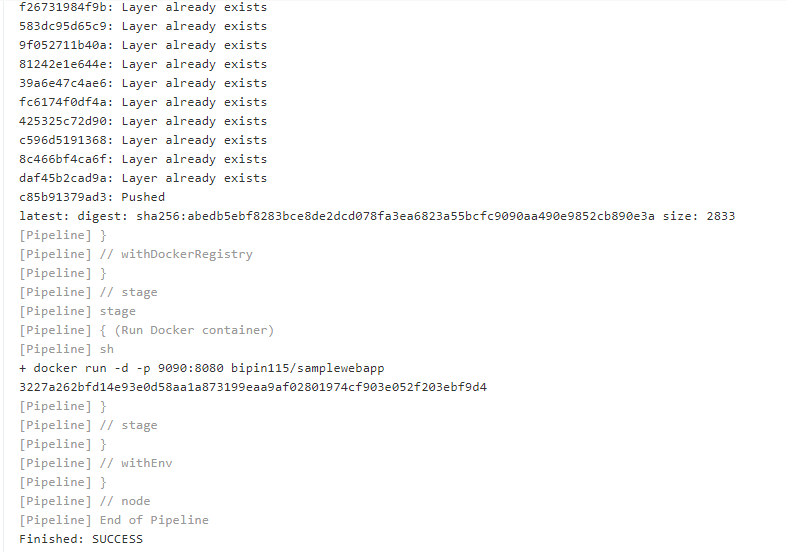




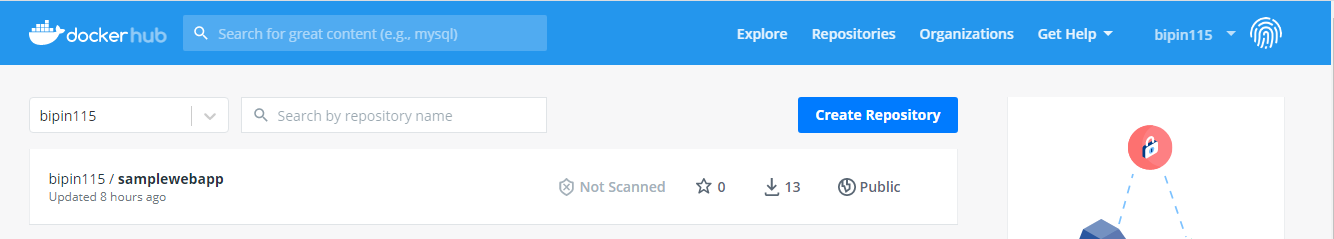








Uploaded Image on the Docker Hub.



Checking the running container on the console



Checking output in the browser

URL: <PUBLIC\_IP\_Address>:9090/ <War file Name> /

My app url: http://13.233.108.115:9090/java-tomcat-maven-example/

