**Jenkins file:**

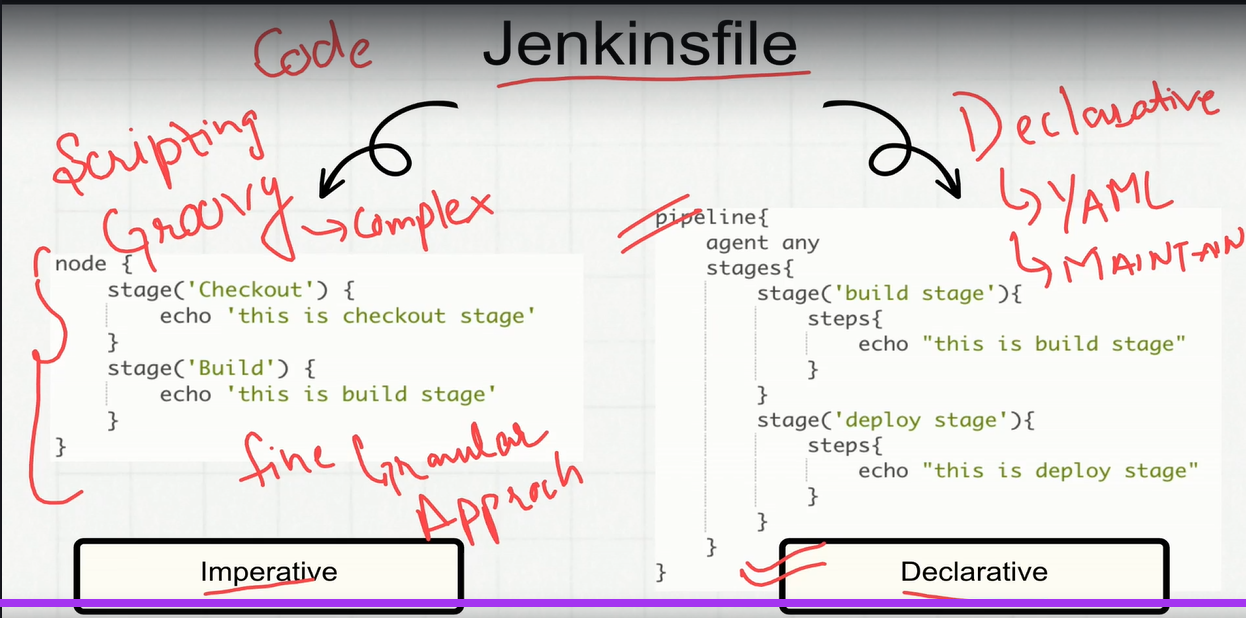
**Imperative**: Scripted (Groovey) – Complex code, Fine-grained approach

**Declarative:**

Simple syntax, easy to understand

Yaml language

Easy to maintain and rectify errors.



============ **1- Jenkinsfile - Basic pipeline** ===================

pipeline{

agent any

stages{

stage('Build stage'){

steps{

echo "this is build stage"

}

}

stage('Test stage'){

steps{

echo "this is test stage"

}

}

stage('Deploy stage'){

steps{

echo "this is deploy stage"

}

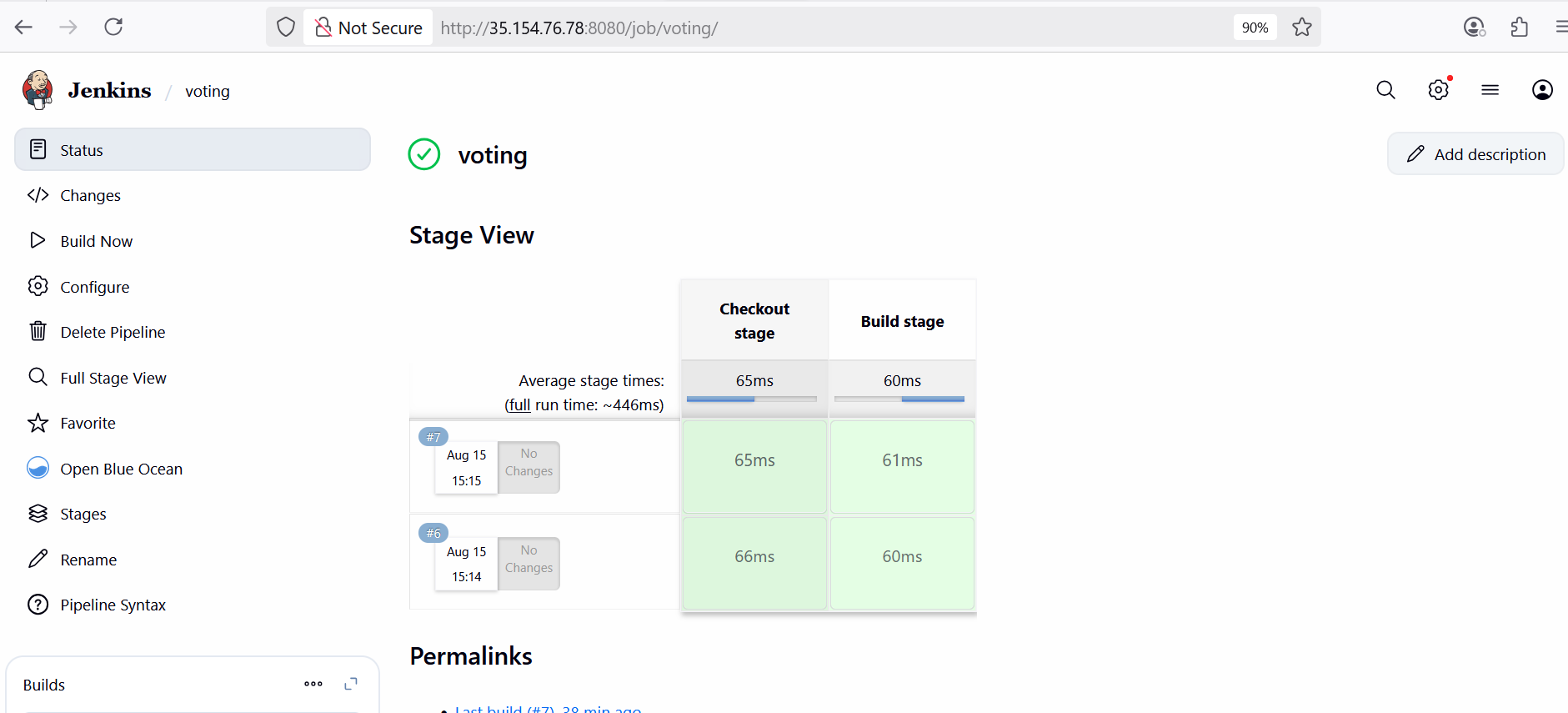
}

}

}

**Plugin**

* **Pipeline stage view**

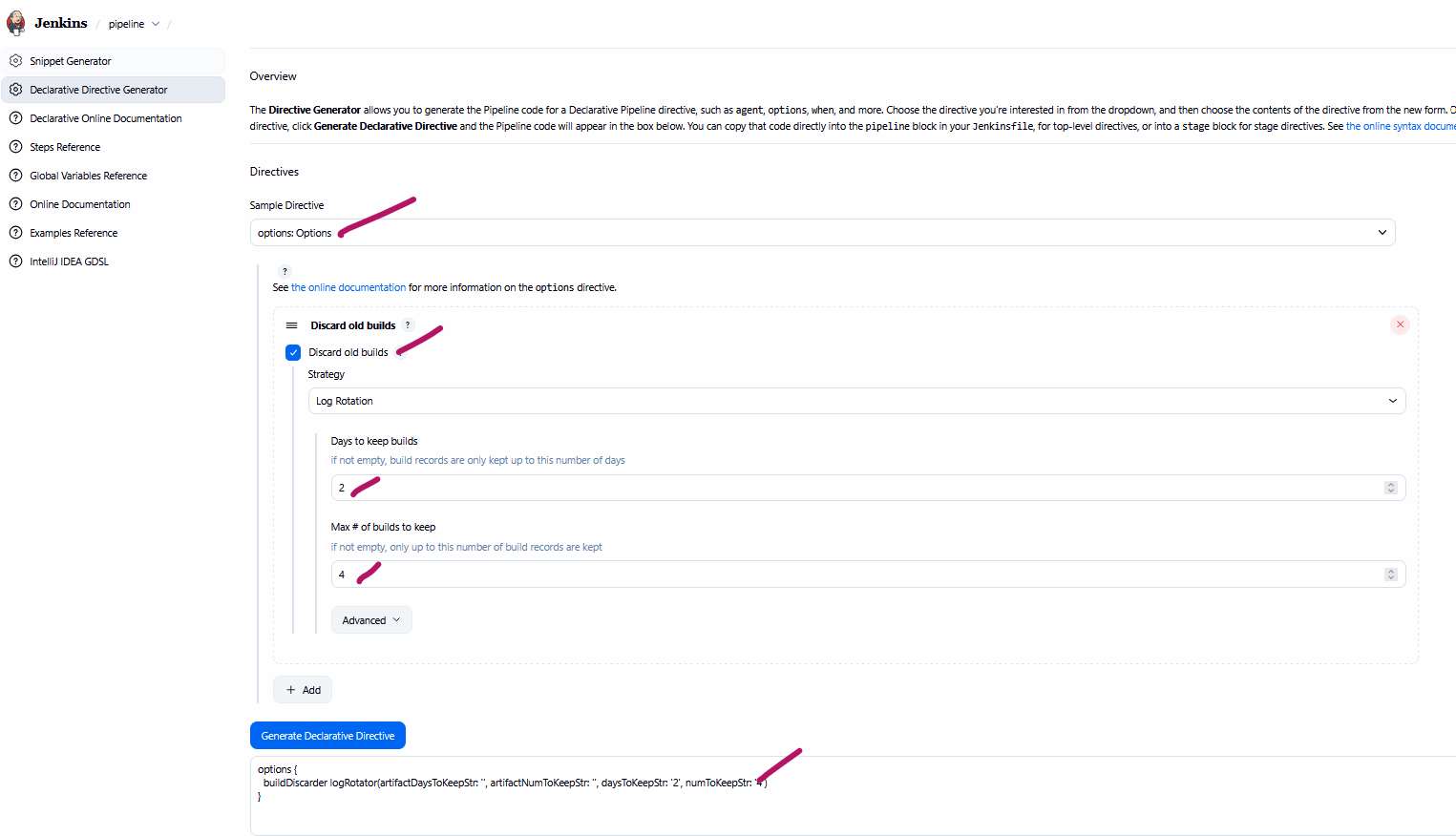
****

**=============== 2-Jenkinsfile: Using Options – Log rotation ===============**

**Use built-in pipeline syntax to create a Jenkinsfile, so no need to write everything manually. Just add details, it will make the details of that stage.**

**Create pipeline :**

**Click on pipeline syntax (left end) ->Declarative directive generator**

****

This will hold only 4 builds for 2 days, and the remaining builds will clean up automatically

pipeline{

agent any

**options** {

**buildDiscarder logRotator(artifactDaysToKeepStr: '', artifactNumToKeepStr: '', daysToKeepStr: '2', numToKeepStr: '4')**

}

stages{

stage('Build stage'){

steps{

echo "This is build stage"

}

}

}

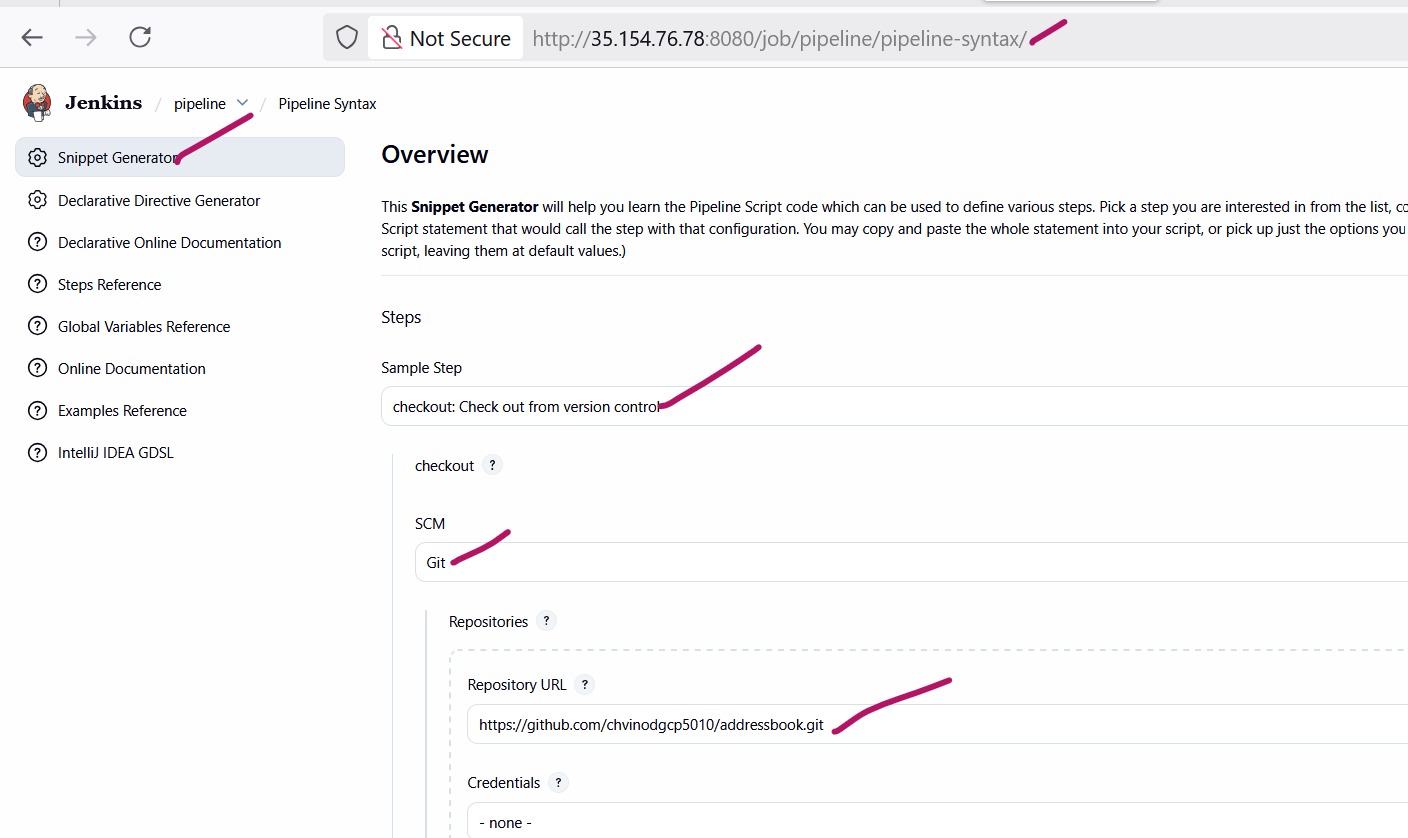
}

**Note:**

**It keeps a maximum of 4 builds only for 2days**

**=============== 3- Jenkinsfile with Checkout stage ===================**

<https://github.com/chvinodgcp5010/addressbook.git>



**Click on Pipeline syntax -> snippet generator ->Checkout : from version control**

Once done, click generate pipeline script

**checkout scmGit(branches: [[name: '\*/master']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/chvinodgcp5010/addressbook.git']])**

pipeline{

agent any

stages{

stage('Checkout git'){

steps{

**checkout scmGit(branches: [[name: '\*/master']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/chvinodgcp5010/addressbook.git']])**

}

}

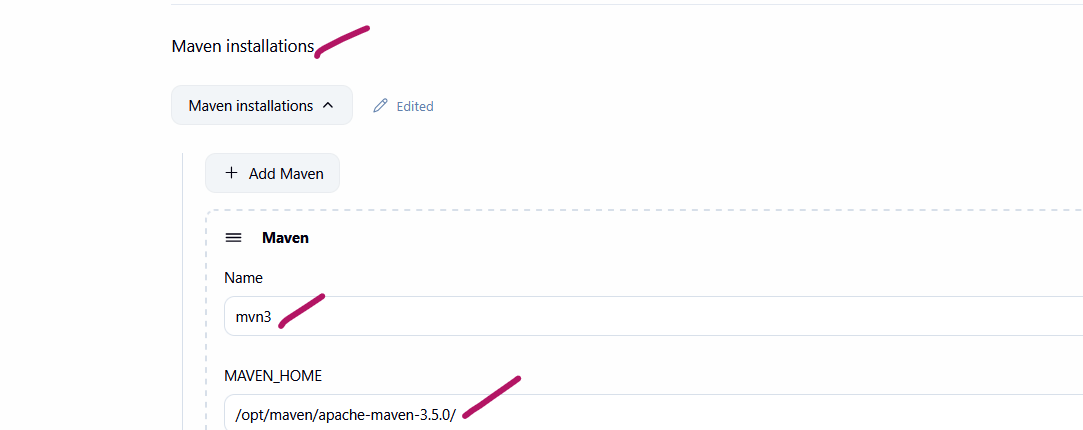
}

}

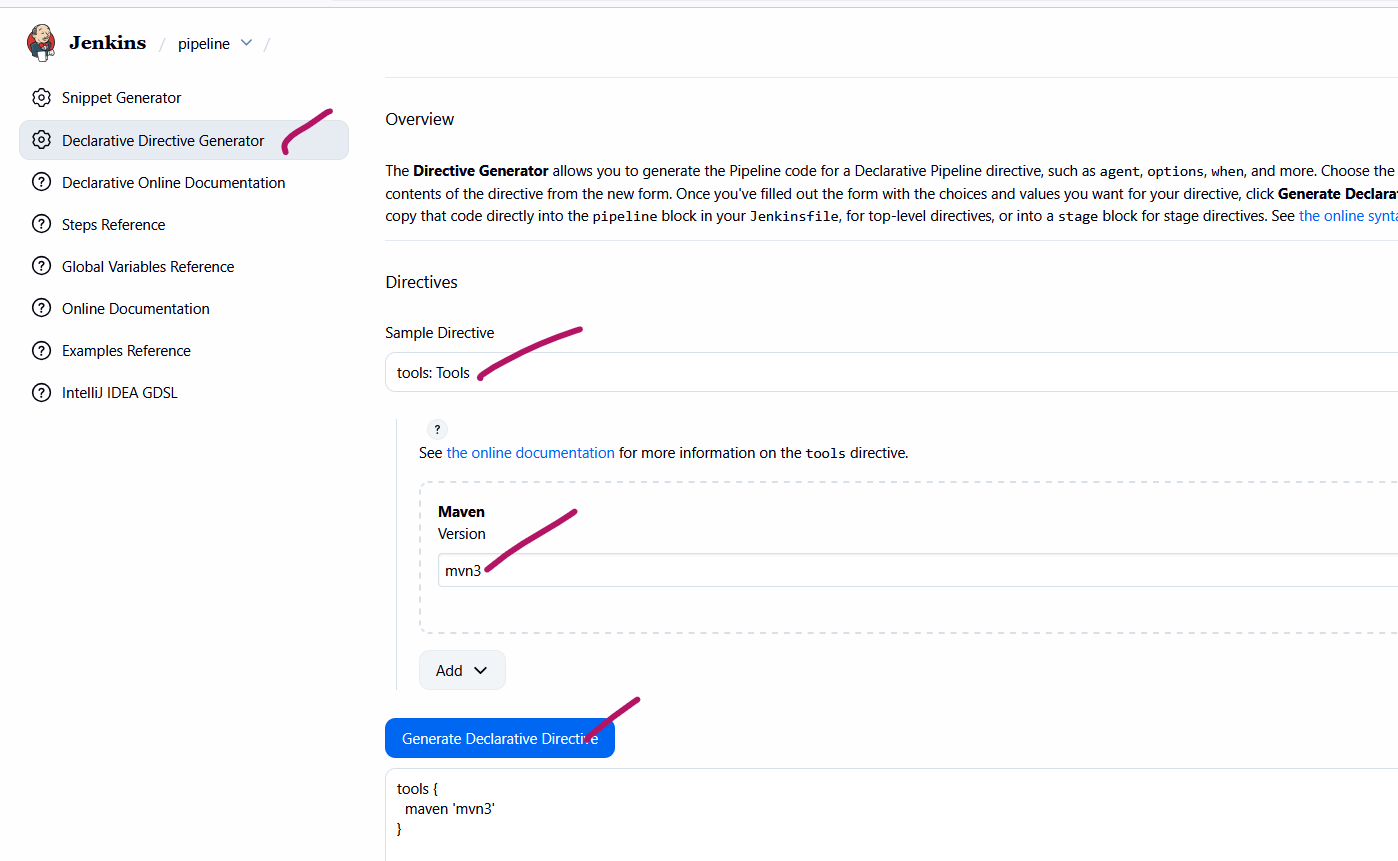
**================ 4- Jenkinsfile - Using tools =========================**

First, configure tools ( here, Maven)

Manage Jenkins -> Go to tools and configure Maven



Go to pipeline syntax – Declarative directive generator and choose tools



pipeline{

agent any

**tools {**

**maven 'mvn3'**

**}**

stages{

stage('source checkout'){

steps{

checkout scmGit(branches: [[name: '\*/master']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/chvinodgcp5010/addressbook.git']])

}

}

stage('Build code'){

steps{

**sh 'mvn clean package'**

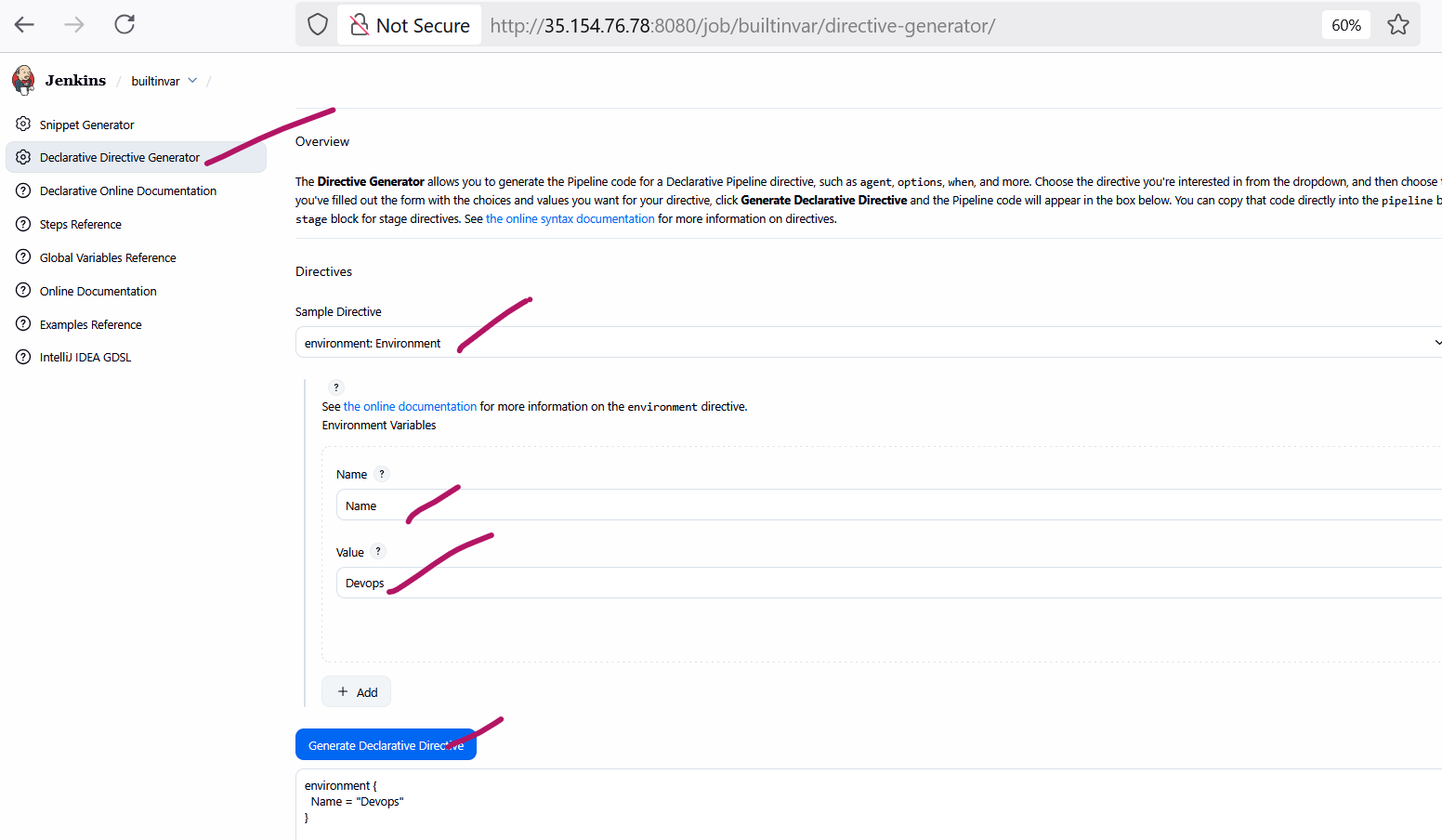
}

}

}

}

**==================== 5- Jenkinsfile – Env variable ===============**



pipeline{

agent any

environment {

Name = "Devops"

}

stages{

stage('env stage'){

steps{

echo "I am doing ${Name}"

}

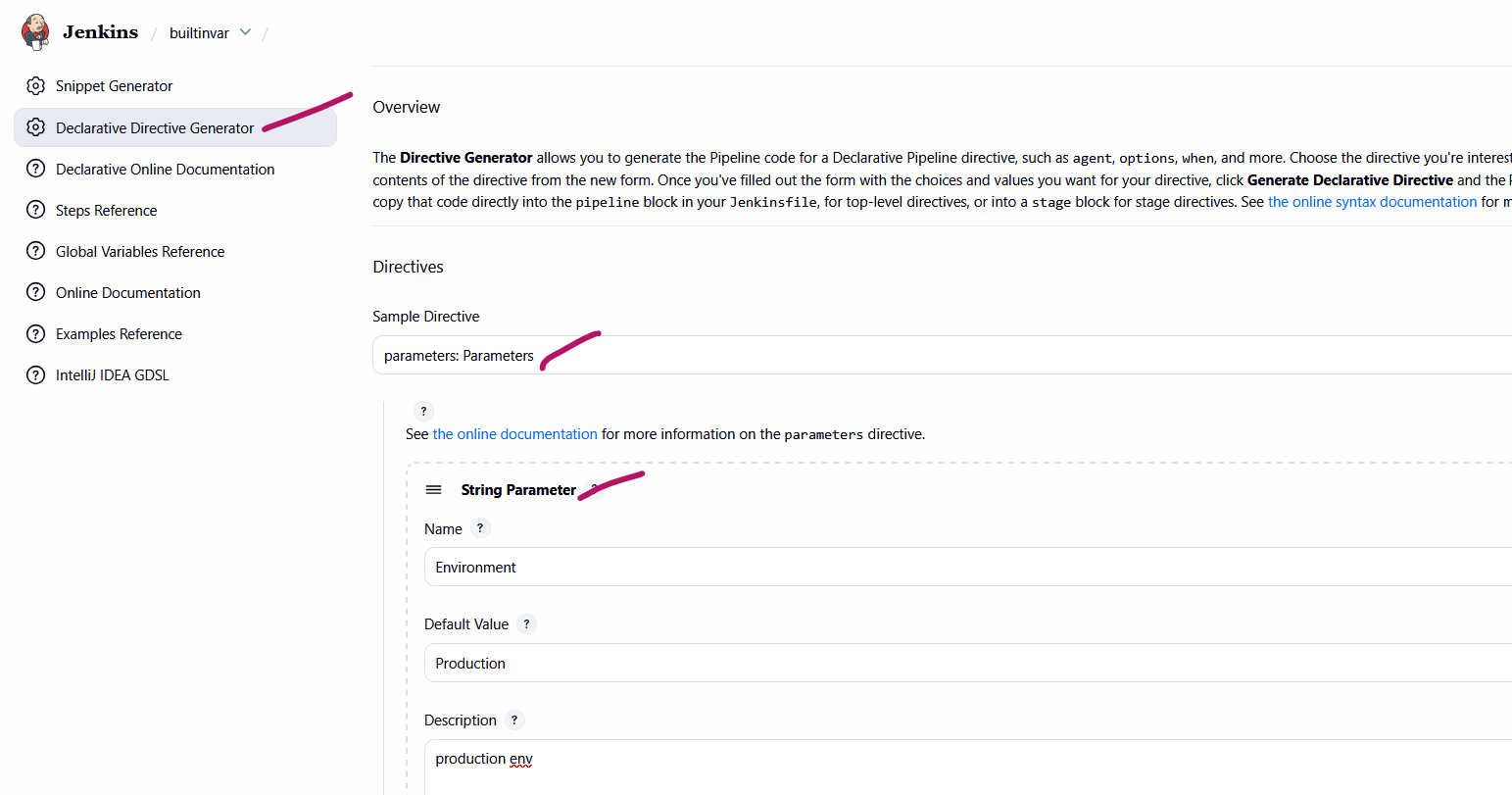
}

}

}

**================= 6 – Jenkinsfile – with parameters ======================**

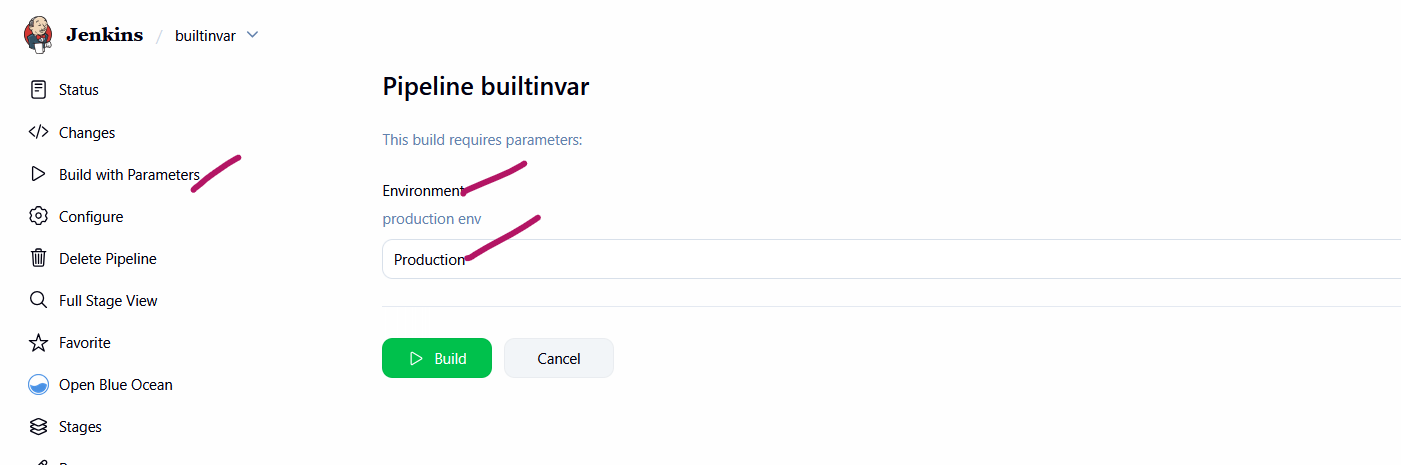
**Once you add, only then will you get an option of “Build with parameters” option on Jenkins**



parameters {

string defaultValue: 'Production', description: 'production env', name: 'Environment'

}



Here default value of the environment is “Production,” but we can update anything

pipeline{

agent any

**parameters {**

**string defaultValue: 'Production', description: 'production env', name: 'Environment'**

**}**

stages{

stage('using parameter values'){

steps{

**echo "Using $params.Environment"**

}

}

}

}

O/P:’



============**7 & 8 Jenkinsfile – Upstream and downstream job** ================

First, create 2 jobs: 1. Upstream 2. Downstream ( name: **'down-vnd-job')**

**Upstream Jenkinsfile**

pipeline{

agent any

stages{

stage('Upstream job'){

steps{

echo "This is upstream"

}

}

stage('Downstream job'){

steps{

**build job: 'down-vnd-job'**

}

}

}

}

Once Upstream is completed, it should trigger automatically a downstream job named” **'down-vnd-job'**

Downstream job – job name**: down-vnd-job**

pipeline{

agent any

stages{

stage('Downstream Job'){

steps{

echo "Successfully triggered downstream job using upstream"

}

}

}

}

**Note:**

**Here, if you trigger the upstream job, it will automatically trigger the downstream job as well.**

=============== **9 – Jenkinsfile: Using Environment variables** ==================

<https://www.jenkins.io/doc/book/pipeline/jenkinsfile/#using-environment-variables>

pipeline{

agent any

stages{

stage('built in variables'){

steps{

echo "Job number: ${env.**BUILD\_NUMBER**}"

// echo "Job number : ${ **BUILD\_NUMBER**}"

echo "Job Name: ${env.**JOB\_NAME**}"

// echo "Job Name: ${**JOB\_NAME**}"

}

}

}

}

======================**10- Jenkinsfile – using script** =========================

pipeline{

agent any

stages{

stage('script usage'){

steps{

script{

def age = 5

def result = age \* 2

echo "Hello, my age is ${result}"

}

}

}

}

}



================ **11- Jenkinsfile: using filescript** =======================

<https://github.com/chvinodgcp5010/addressbook.git>

**helloworld.sh**

#!/bin/bash

echo "Hello world"

pipeline{

agent any

stages{

stage('Clone repo'){

steps{

checkout scmGit(branches: [[name: '\*/master']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/chvinodgcp5010/addressbook.git']])

}

}

stage('shell script'){

steps{

**sh '''**

**chmod +x script.sh**

**sh script.sh**

**'''**

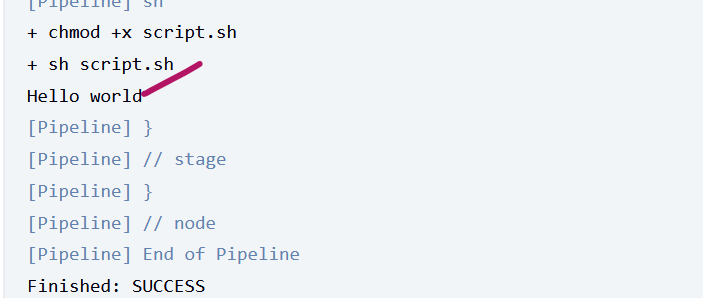
}

}

}

}

O/P:



============== **12- Jenkinsfile – Using** **loops** ================

pipeline{

agent any

stages{

stage('loop usage'){

steps{

script{

echo "Table of 5"

for (int i = 1; i <= 10; i++) {

echo "5 X ${i} = ${5 \* i}"

}

}

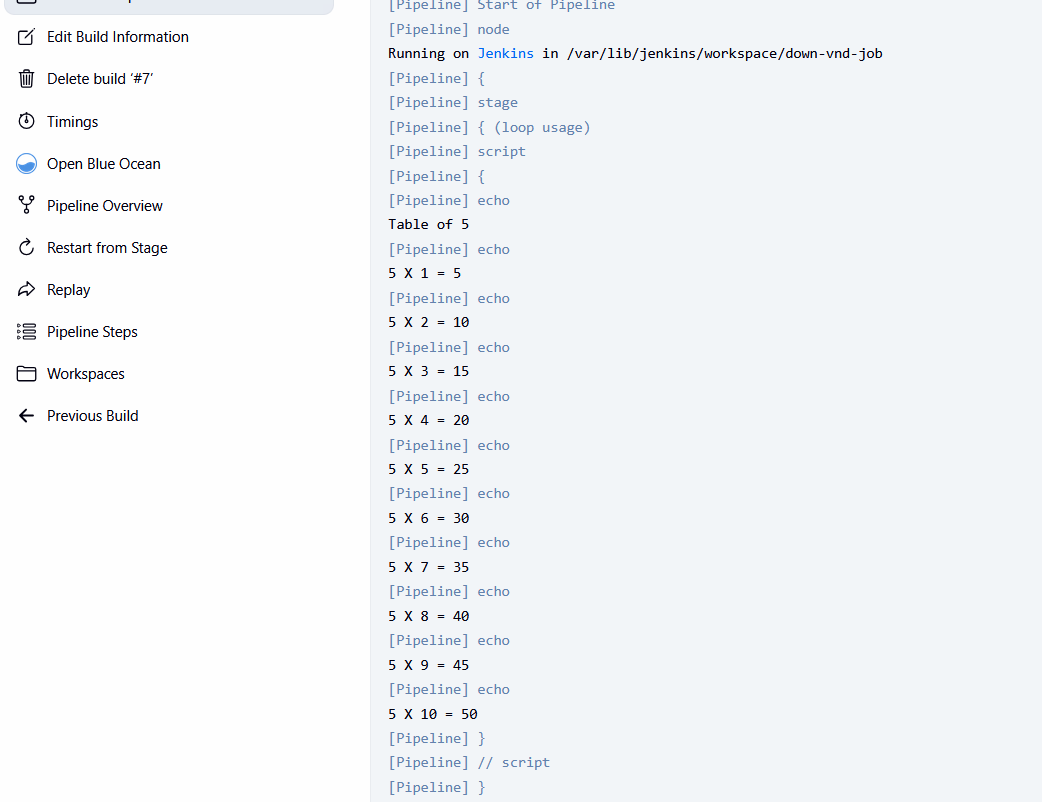
}

}

}

}

O/P:



================== **13- Jenkinsfile : Loops and script** ==================

<https://github.com/chvinodgcp5010/addressbook.git>

#!/bin/bash

env=$1

echo "Hello, I am in $env stage"

**--- Pipeline --**

pipeline{

agent any

stages{

stage('Checkout'){

steps{

checkout scmGit(branches: [[name: '\*/main']], extensions: [], userRemoteConfigs: [[url: 'https://https://github.com/chvinodgcp5010/addressbook.git]])

}

}

stage('script usage'){

steps{

script{

def values = ['uat', 'dev' , 'prod']

sh 'chmod +x helloworld.sh'

for (name in values) {

sh "sh helloworld.sh ${name}"

}

}

}

}

}

}

O/P:



**=============15- Jenkinsfile – conditions (if – else ) ===================**

pipeline{

agent any

stages{

stage('vote eligibility'){

steps{

script{

def age = 40

if (age >= 18) {

echo "Congrats you are eligible"

}

else{

echo "You are not eligible"

}

}

}

}

}

}

**============== ========= Docker ==============**

Is an Open-source platform for developing, shipping, and running applications in a standardized way using containers.

->contains all dependencies of appn

**Commands**

Docker ps

Docker run [image] [command]

Docker stop [containerid]

Docker rm [containerid]

Docker images

Docker rmi [imageid/name]

Docker - Installation

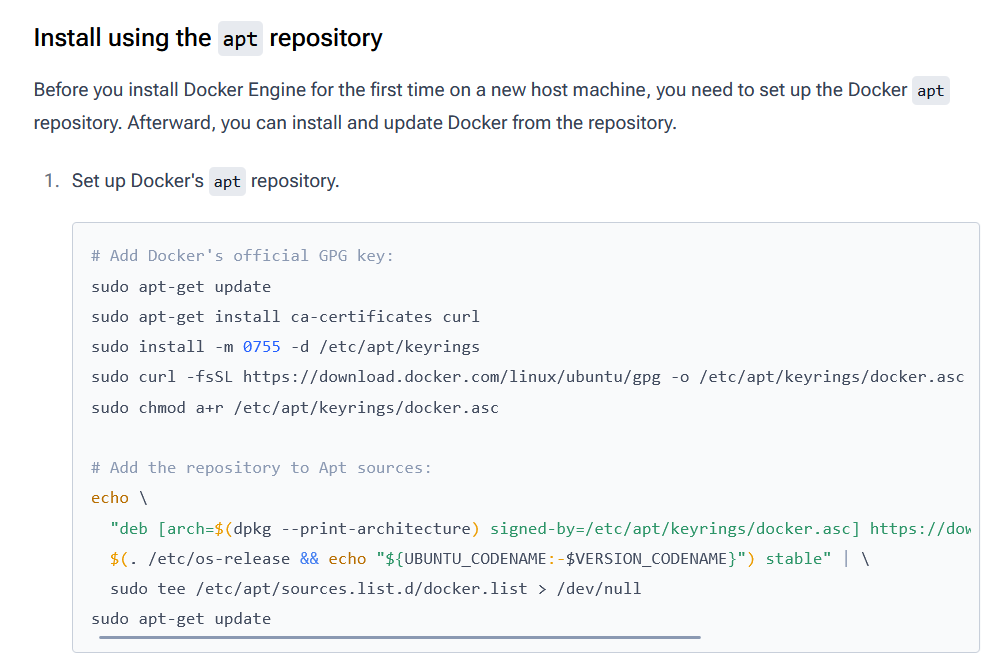
Before installation, install Docker **plugins**

**Docker** - This plugin integrates Jenkins with [Docker](http://docker.io/)

**Docker pipeline** - Build and use Docker containers from pipelines.

<https://docs.docker.com/engine/install/ubuntu/>

On Ubuntu



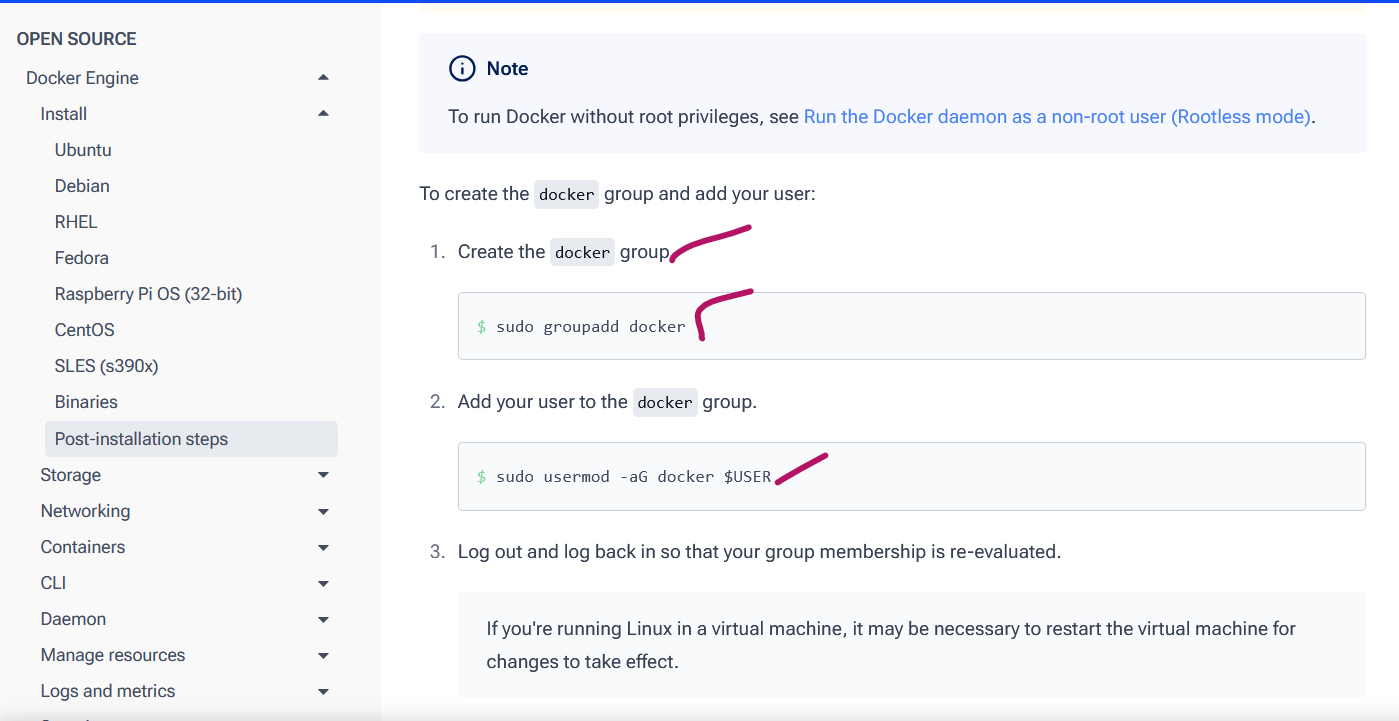
**2.Install docker packages**

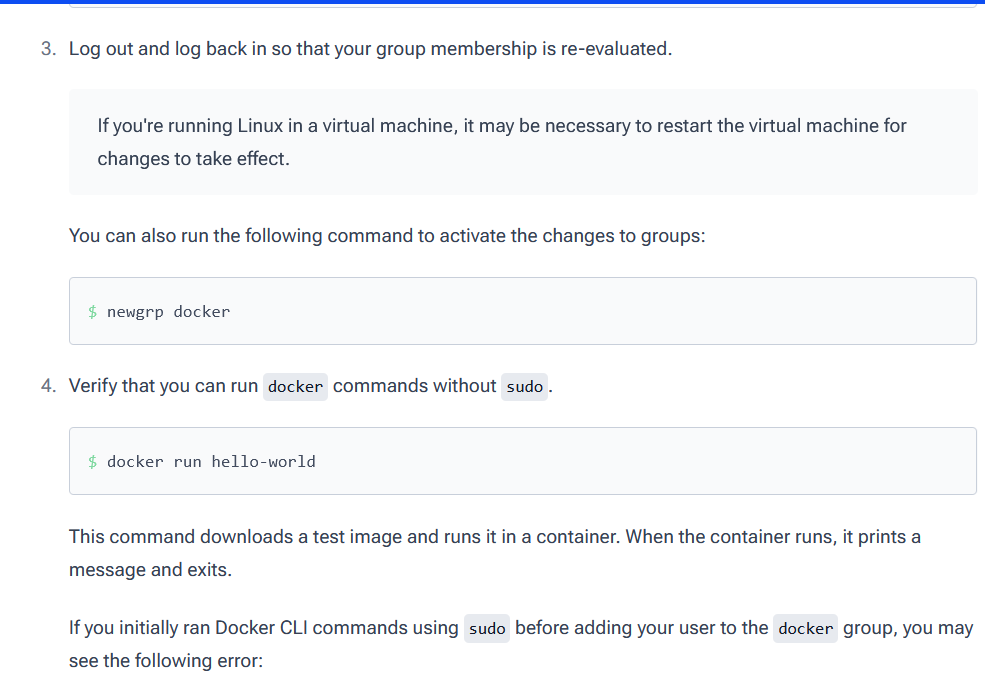
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

**3. that the installation is successful by running the hello-world image:**

sudo docker run hello-world

<https://docs.docker.com/engine/install/linux-postinstall/>





[**Configure Docker to start on boot with systemd**](https://docs.docker.com/engine/install/linux-postinstall/#configure-docker-to-start-on-boot-with-systemd)

sudo systemctl enable docker.service

sudo systemctl enable containerd.service

**Update docker path on Jenkins:**

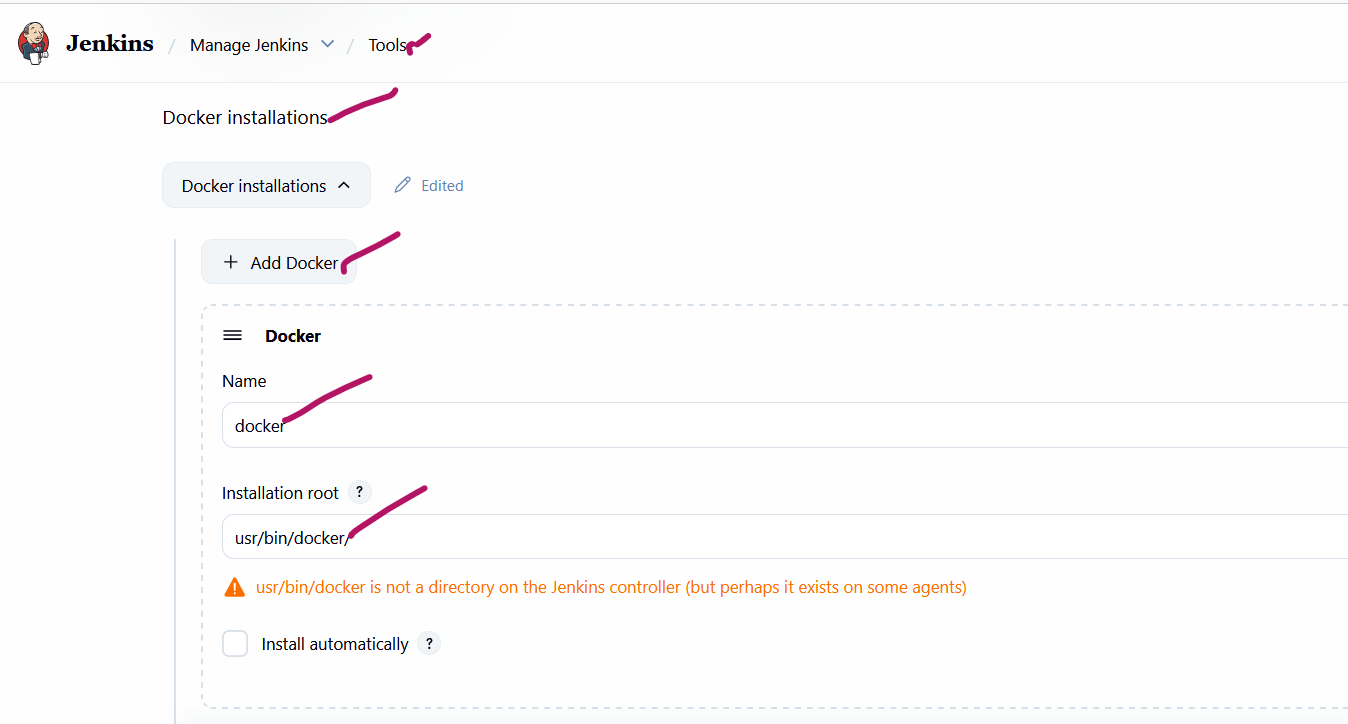
ubuntu@ip-172-31-39-248:~$ **which docker**

**/usr/bin/docker**

ubuntu@ip-172-31-39-248:~$ **docker --version**

Docker version **28.3.3**, build 980b856

**Manage Jenkins -> tools ->**

****

**Jenkinsfile**

pipeline{

agent any

stages{

stage('docker'){

steps{

echo "docker version"

sh 'docker --version'

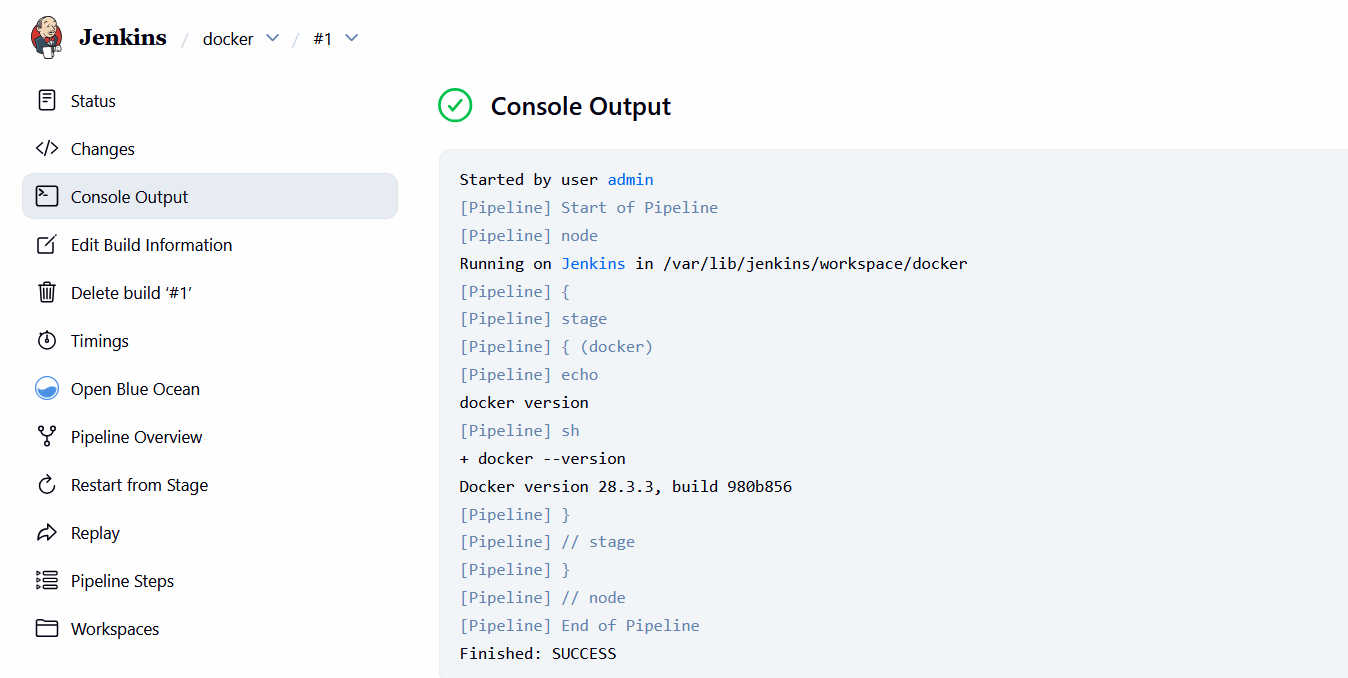
}

}

}

}

**O/P**:



pipeline{

agent any

stages{

stage('docker'){

steps{

**echo "docker version"**

**sh 'docker --version'**

**sh 'docker run -d nginx'**

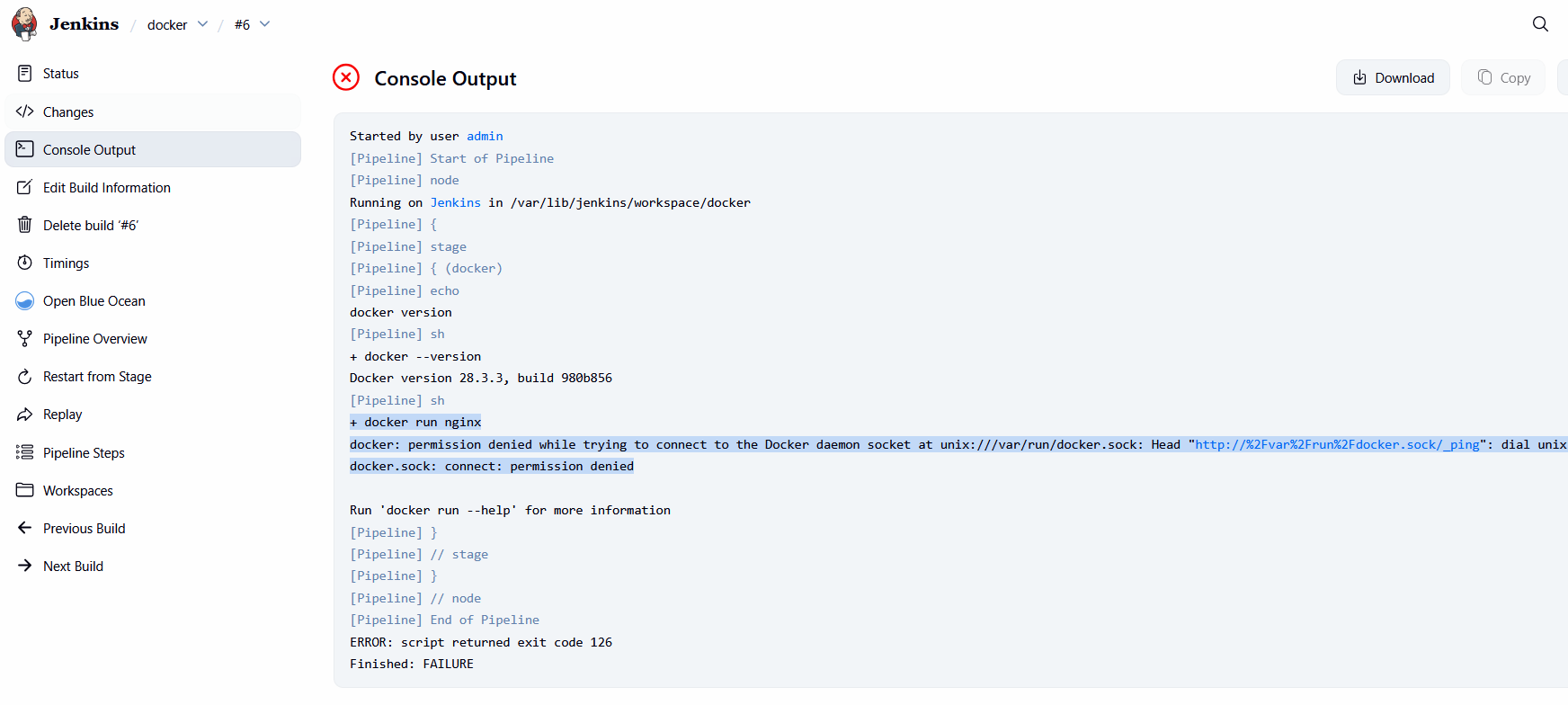
**sh 'docker ps'**

}

}

}

}

****

**To fix the issue, follow the process below**

[**https://linuxhandyman.com/docker\_daemon\_not\_running/?gad\_source=1&gad\_campaignid=22877399015&gclid=CjwKCAjwtfvEBhAmEiwA-DsKjnvOadTVkHGklrA8cTrgQGxkQUSbdmObxuTczTqk-7P0hX0i19ELPRoCZW4QAvD\_BwE**](https://linuxhandyman.com/docker_daemon_not_running/?gad_source=1&gad_campaignid=22877399015&gclid=CjwKCAjwtfvEBhAmEiwA-DsKjnvOadTVkHGklrA8cTrgQGxkQUSbdmObxuTczTqk-7P0hX0i19ELPRoCZW4QAvD_BwE)

**3. Fix Docker Socket Permissions**

**Check current socket permissions:**

ls -la /var/run/docker.sock

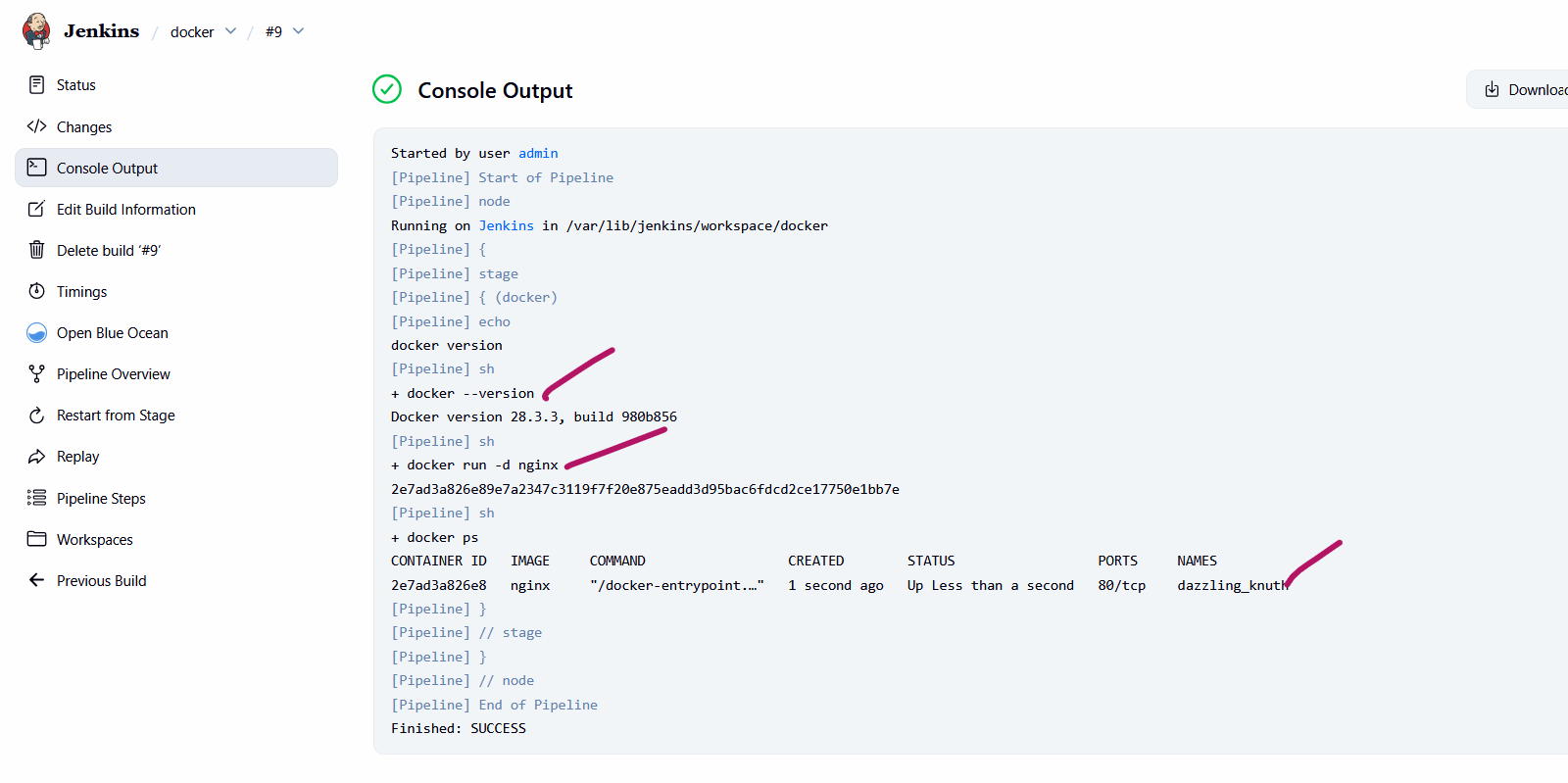
**Fix socket permissions (if needed):**

sudo chmod 666 /var/run/docker.sock

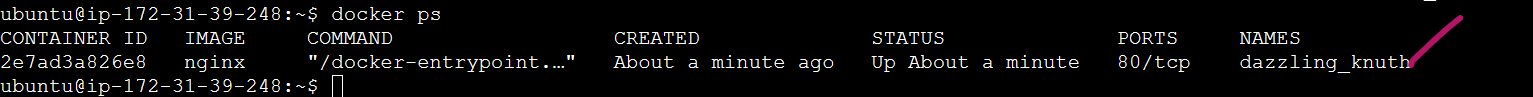
**4. Restart Docker Service (Advanced)**

sudo systemctl restart docker

**5.Try to execute the above build, and it will work**

****

**Check the Docker container created on the server**

****

**=============16- Jenkinsfile with Docker ===================**

pipeline{

agent any

tools {

maven 'mvn3'

}

stages{

stage(**'Source checkout'**){

steps{

checkout scmGit(branches: [[name: '\*/master']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/chvinodgcp5010/addressbook.git']])

}

}

stage("**Build code using maven**"){

steps{

sh 'mvn clean package'

}

}

stage("**package & Deploy code using Docker**"){

steps{

sh 'docker build -t addressbook .'

sh 'docker run -d addressbook'

}

}

stage("**Lisk docker images & running containers**"){

steps{

sh 'docker images'

sh 'docker ps'

}

}

}

}

O/P:



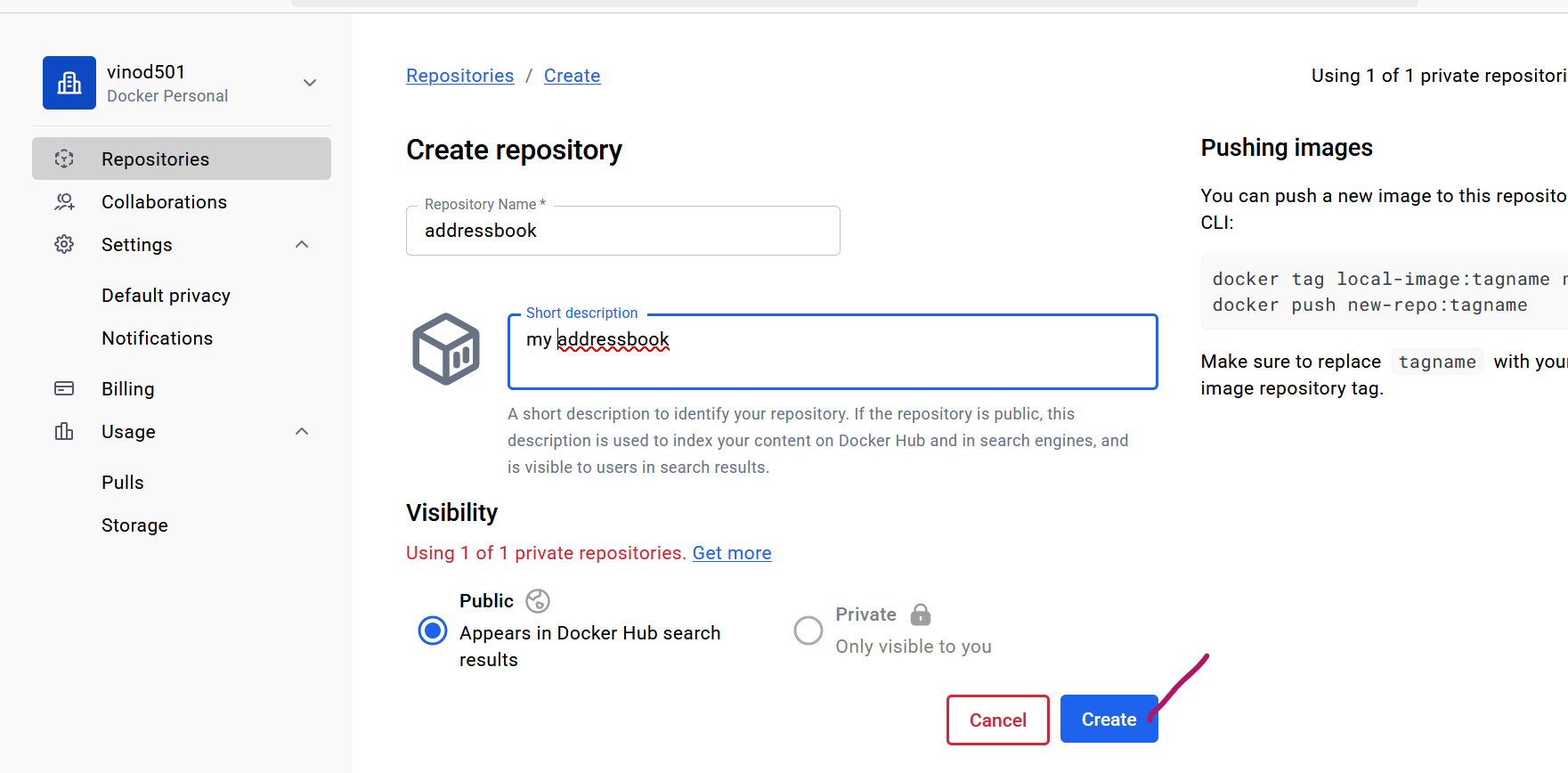
**============= 16.Jenkinsfile – Push image to Docker Hub to store image =============**

To save images, we required a repository. For this, we required a Docker login

We have to give credentials to save (securely save username and password, and secret info in Jenkins)

<https://hub.docker.com/repositories/vinod501>

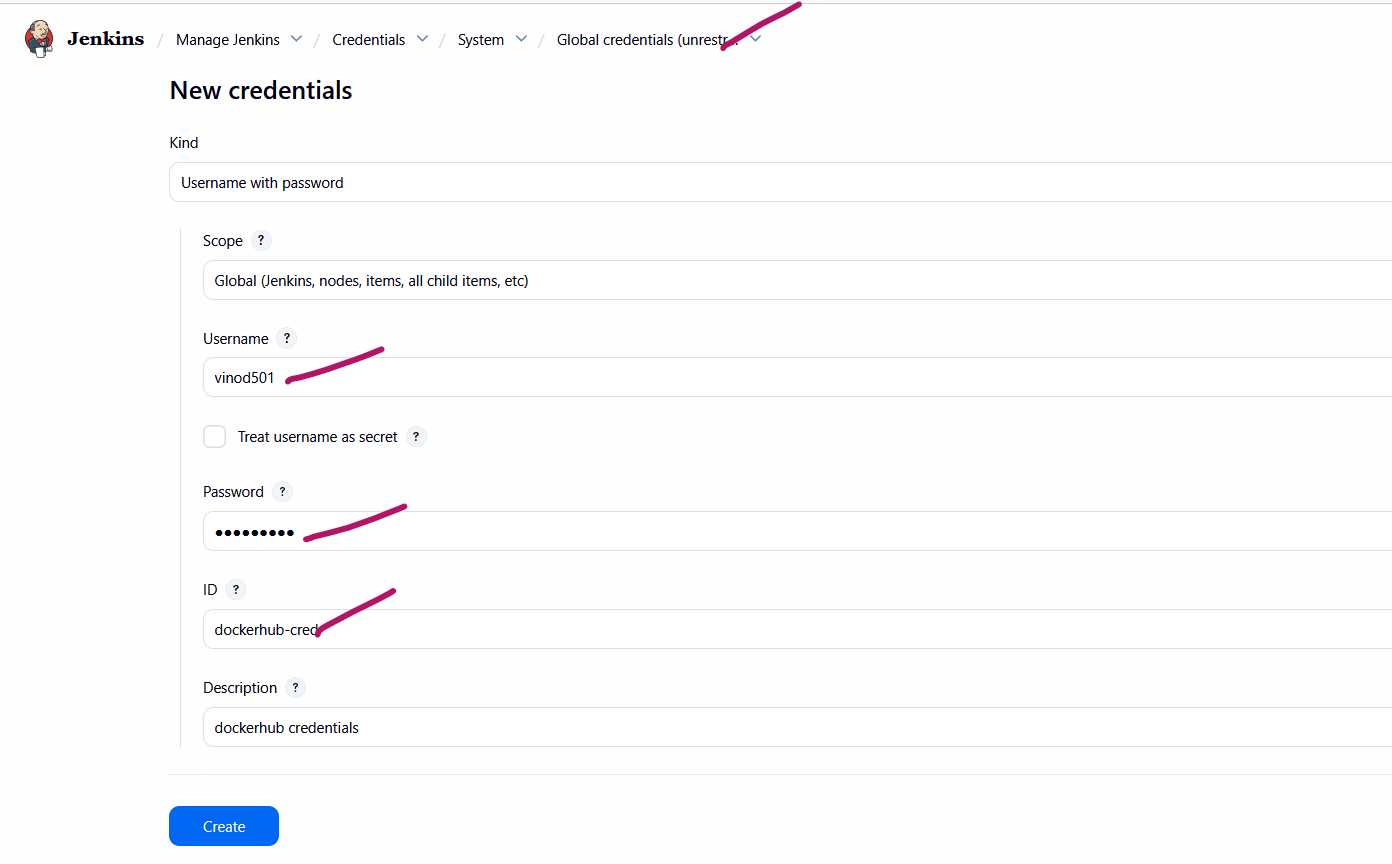
1. first, log in to Docker Hub and create a repository named (address book)and create

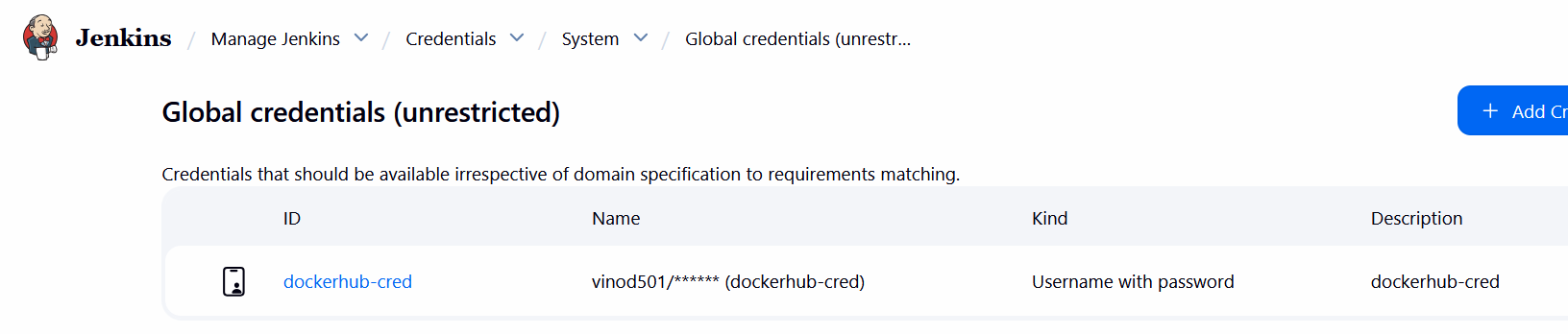


**2. Save Docker credentials on Jenkins**

Go to Jenkins -> Manage Jenkins -> Credentials -> System -> Global credentials

* Add credentials -> User name with password





pipeline{

agent any

environment {

**DOCKERHUB\_CREDENTIALS= credentials('dockerhub-cred')**

}

tools { //use syntax generator

**maven 'mvn3'**

}

stages{

stage('git checkouit'){

steps{

**checkout scmGit(branches: [[name: '\*/master']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/chvinodgcp5010/addressbook.git']])**

}

}

stage('Maven build'){

steps{

**sh 'mvn -v'**

**sh 'mvn clean package'**

}

}

stage('Docker build and push images'){

steps{

script{

withCredentials([usernamePassword(credentialsId: **'dockerhub-cred'**, passwordVariable: 'DOCKERHUB\_CREDENTIALS\_PSW', usernameVariable: 'DOCKERHUB\_CREDENTIALS\_USR')]) {

sh 'docker login -u $DOCKERHUB\_CREDENTIALS\_USR -p ${DOCKERHUB\_CREDENTIALS\_PSW}'

echo 'Login Completed'

sh 'docker build -t vinod501/aperture-2025 .'

sh 'docker push vinod501/aperture-2025'

}

}

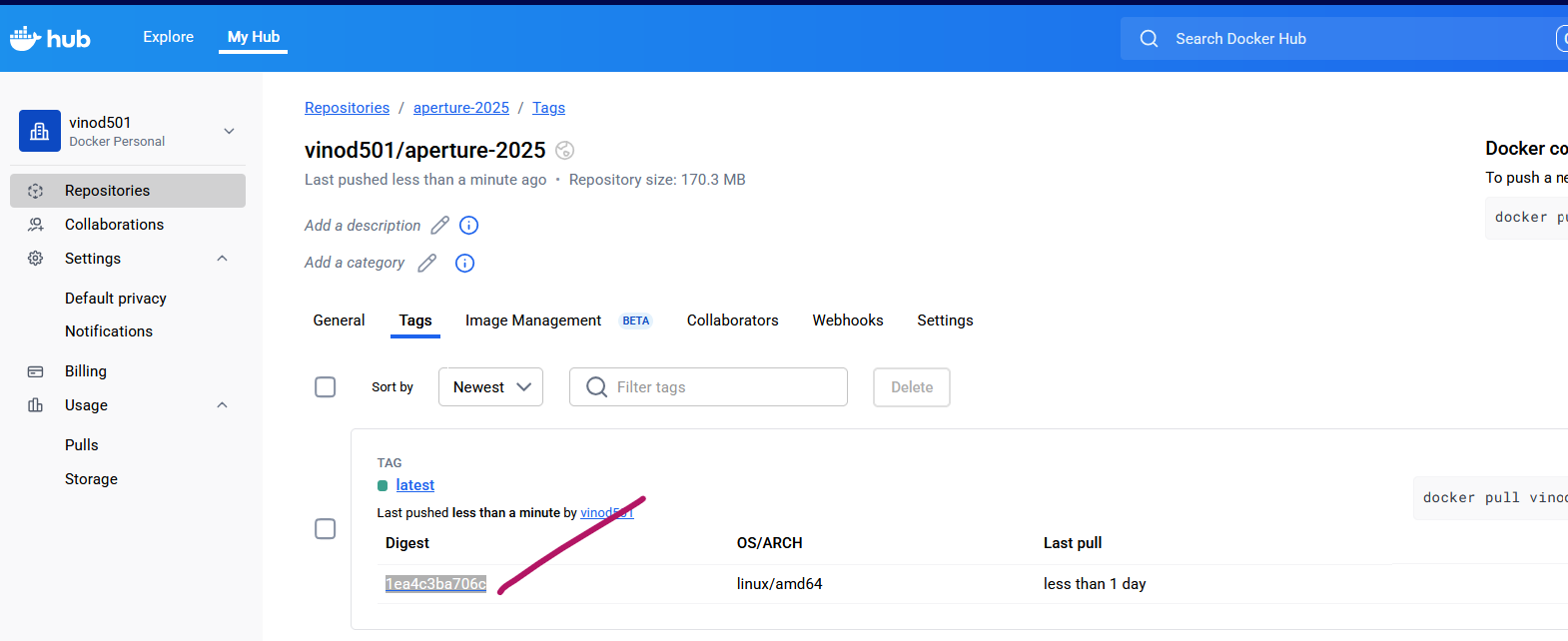
}

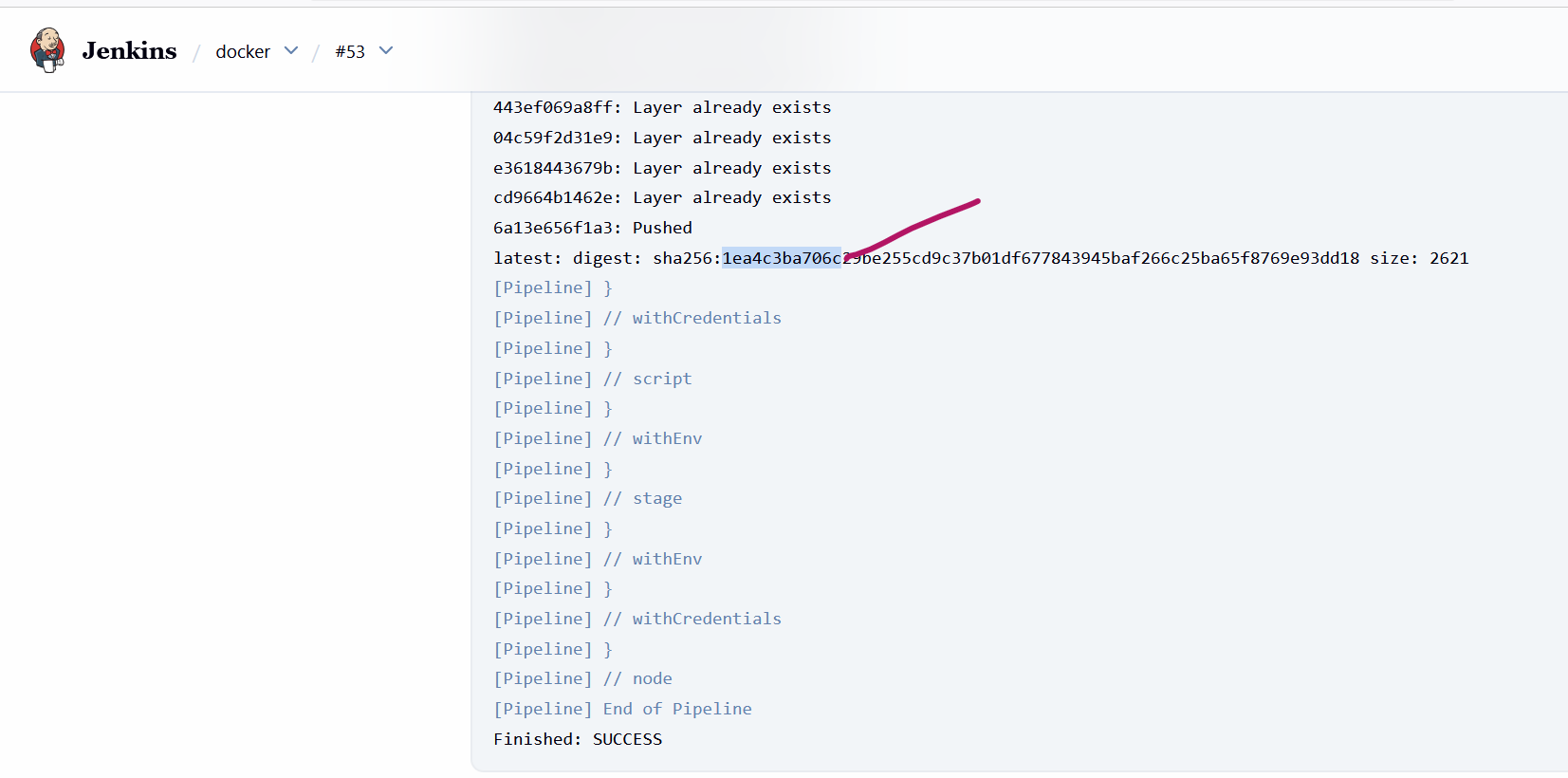
}

}

}

O/P:





================ Realtime project =============

pipeline{

agent any

tools{

maven 'mvn-vnd'

}

options {

**buildDiscarder logRotator(artifactDaysToKeepStr: '', artifactNumToKeepStr: '', daysToKeepStr: '30', numToKeepStr: '1')**

}

environment{

**dockerhub\_cred = credentials('dockerhub-cred')**

}

stages{

stage('Checkout stage'){

steps{

**checkout scmGit(branches: [[name: '\*/master']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/chvinodgcp5010/addressbook.git']])**

}

}

stage('Maven Build'){

steps{

**sh 'mvn package'**

}

}

stage('docker build'){

steps{

**sh "docker build -t vinod501/project-docker:${BUILD\_NUMBER} ."**

}

}

stage('dockerhub push'){

steps{

**sh "echo $dockerhub\_cred\_PSW | docker login -u $dockerhub\_cred\_USR --password-stdin"**

**sh "docker push vinod501/project-docker:${BUILD\_NUMBER}"**

}

}

stage('Docker run'){

steps{

**sh "docker run -d -p 80:8080 --name addressbook vinod501/project-docker:${BUILD\_NUMBER}"**

}

}

}

post {

always {

echo "job is completed"

}

success {

echo "It is a success"

}

failure {

echo "It has failed"

}

}

}