<https://github.com/zakiyanjum/ABC-1611>

# ABC-1611

question-04

1.Create a GitHub repo named ABC-1611

Go to https://github.com and log in.

Click the ➕ New repository button (top-right corner or under your profile icon).

Fill in the fields:

Repository name: ABC-1611

Public or Private: Choose based on your needs.

Initialize this repository with:

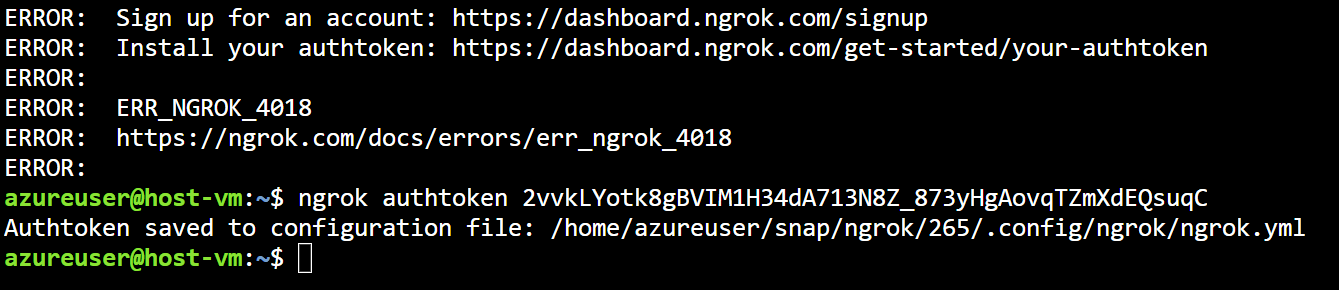
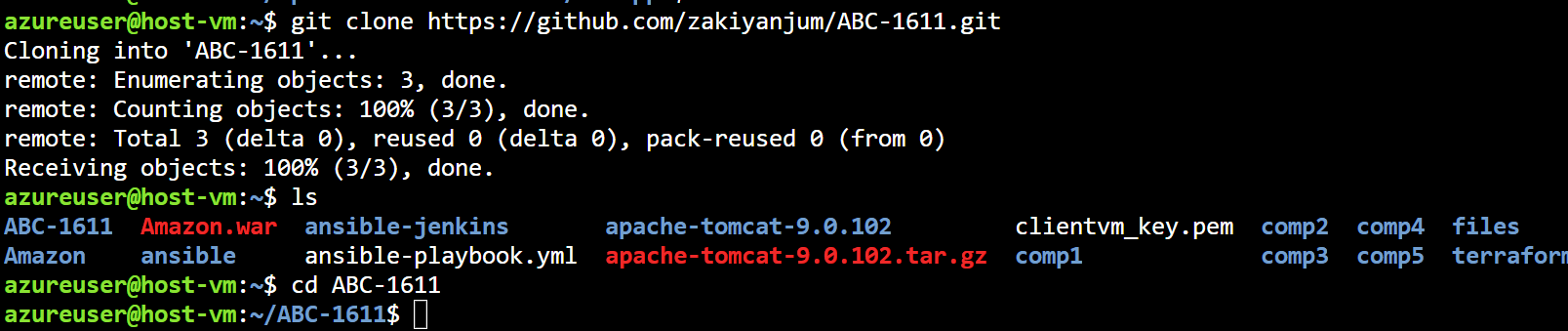
✅ Add a README file (optional)

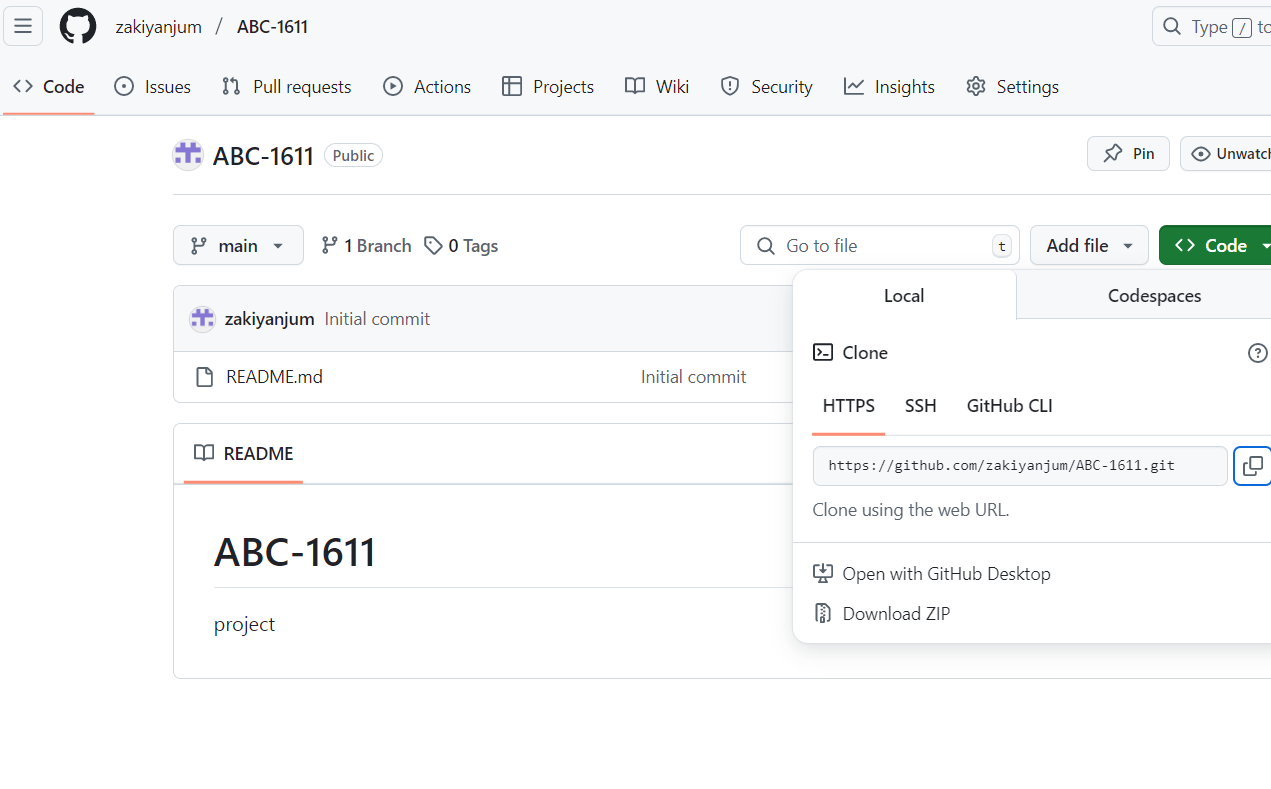
Click Create repository.

2.Clone it locally

git clone https://github.com/zakiyanjum/ABC-1611.git

cd ABC-1611

Git task1 and task2



**3.Use Maven archetype to create a basic web app**

run this in bash

mvn archetype:generate \

-DgroupId=com.example.webapp \

-DartifactId=my-webapp \

-DarchetypeArtifactId=maven-archetype-webapp \

-DinteractiveMode=false

add this in pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>

...

<repositories>

<repository>

<id>adikarthikgupta</id>

<url>https://pkgs.dev.azure.com/adikarthikgupta/\_packaging/adikarthikgupta/maven/v1</url>

<releases>

<enabled>true</enabled>

</releases>

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

</repositories>

...

</project>

setup credentials in settings.xml

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 https://maven.apache.org/xsd/settings-1.0.0.xsd">

<servers>

<server>

<id>adikarthikgupta</id>

<username>AzureDevOps</username>

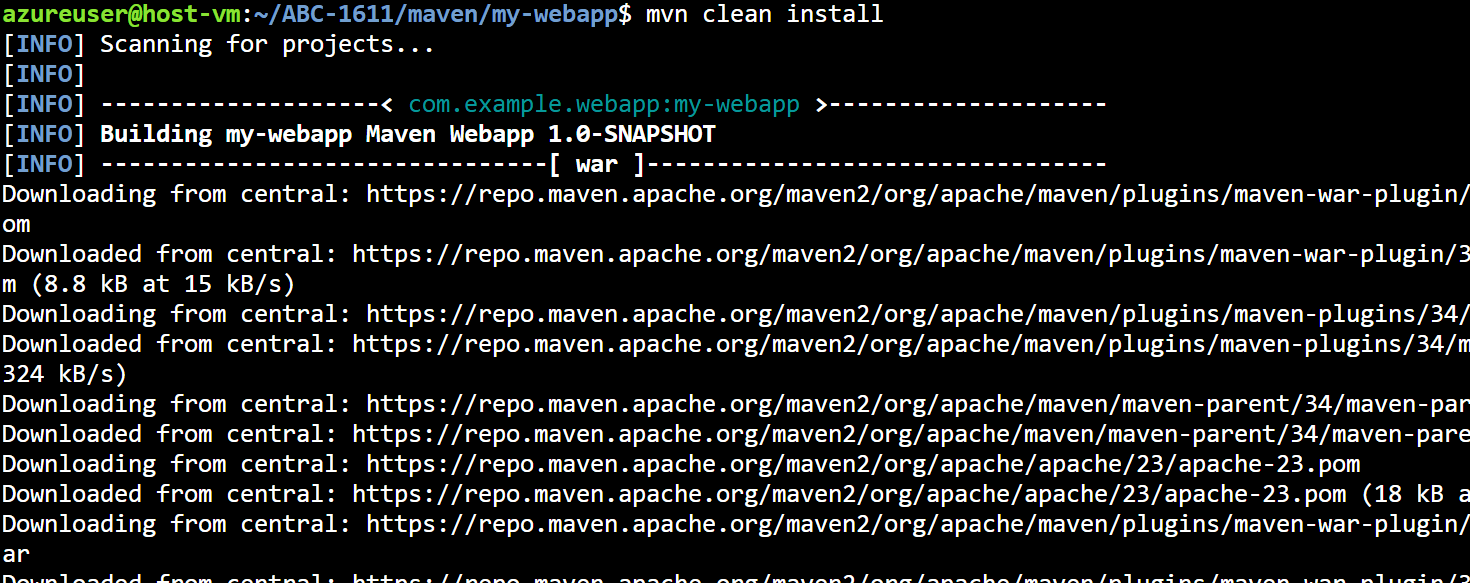
<password>{your-personal-access-token}</password>

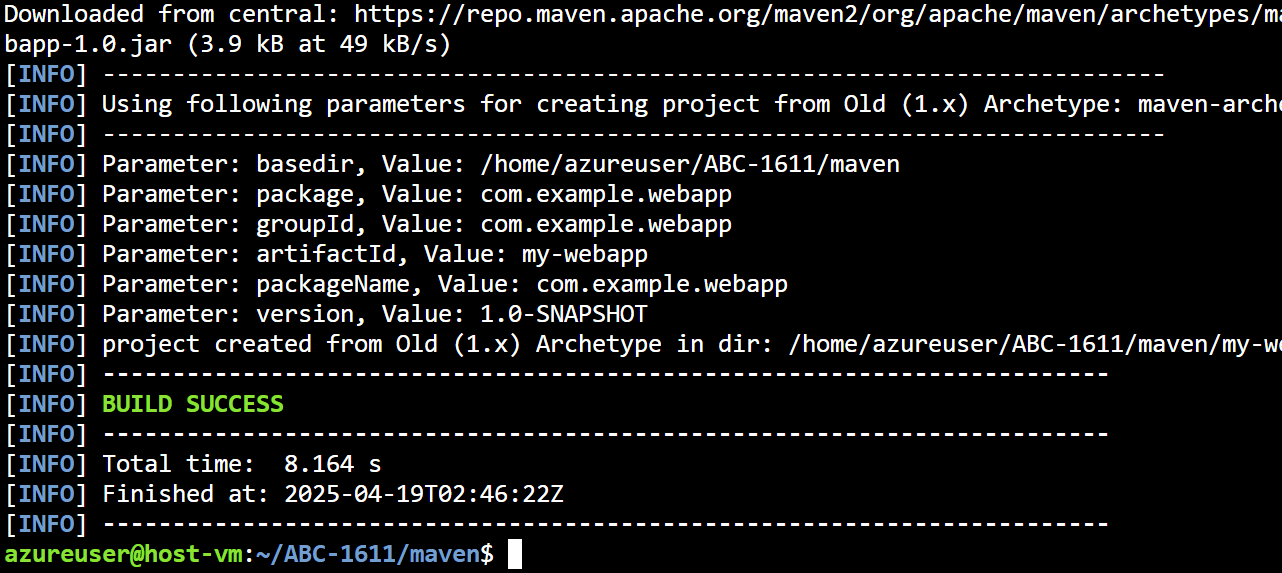
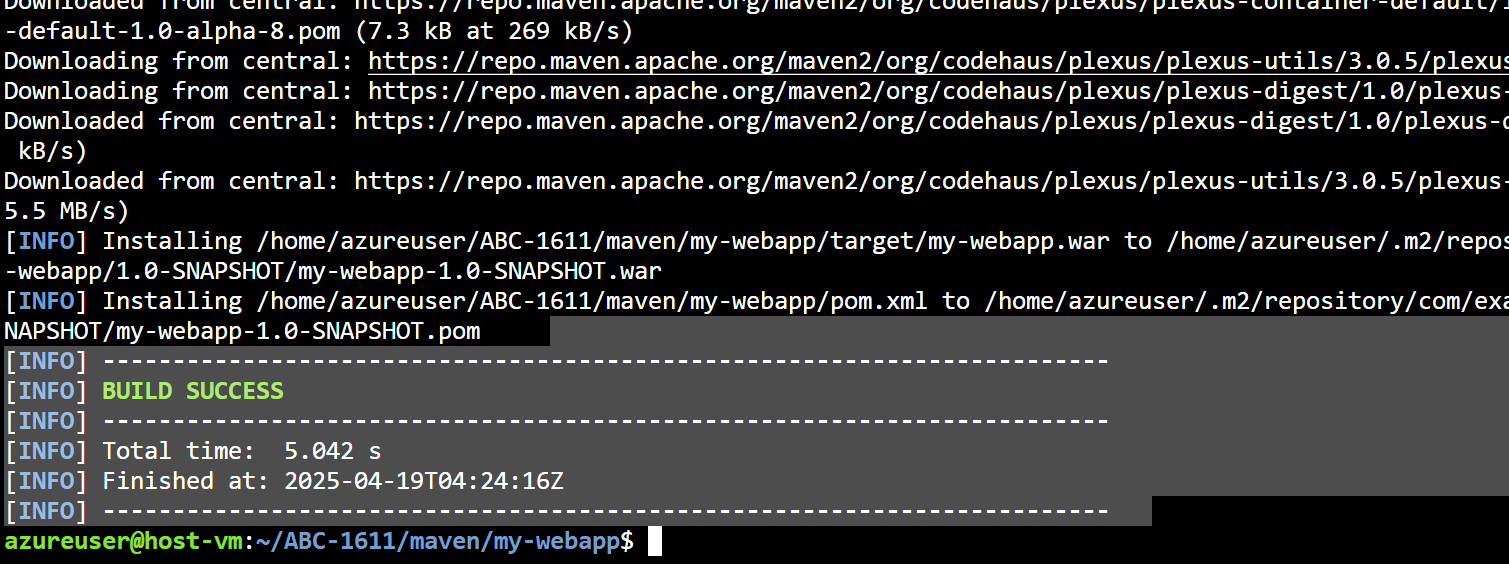
</server>

</servers>

</settings>

and finally do mvn clean install



4.Jenkins pipeline with 3 stages:

1: Build the Maven project

2: Push .war artifact to a remote server

3: Deploy the .war to Tomcat

pipeline {

agent any

environment {

TOMCAT\_URL = 'http://172.184.200.167:8081/' // Tomcat URL

TOMCAT\_USER = 'zakiya' // Tomcat server username

TOMCAT\_PASSWORD = 'zakiya' // Tomcat server password

ARTIFACT\_NAME = 'Amazon.war' // Artifact name

TOMCAT\_HOME = '/home/azureuser/apache-tomcat-9.0.104' // Tomcat home directory

}

stages {

// Stage 1: Clone the project

stage('Clone Project') {

steps {

git branch: 'master', url: 'https://github.com/zakiyanjum/Amazon'

}

}

// Stage 2: Check Project Structure

stage('Check Project Structure') {

steps {

sh '''

echo "Checking directory contents..."

ls -al

if [ ! -f Amazon/Amazon-Web/pom.xml ]; then

echo "pom.xml not found! Build cannot continue."

exit 1

fi

'''

}

}

// Stage 3: Clean the project

stage('Clean') {

steps {

dir('Amazon/Amazon-Web') { // Corrected path

sh 'mvn clean'

}

}

}

// Stage 4: Compile the project

stage('Compile') {

steps {

dir('Amazon/Amazon-Web') { // Corrected path

sh 'mvn compile'

}

}

}

// Stage 5: Run Tests

stage('Test') {

steps {

dir('Amazon/Amazon-Web') { // Corrected path

sh 'mvn test'

}

}

}

// Stage 6: Build the project

stage('Build') {

steps {

dir('Amazon/Amazon-Web') { // Corrected path

sh 'mvn install'

sh 'echo "Listing target directory:"'

sh 'ls -al target || true'

}

}

}

// Stage 7: Verify Artifact

stage('Verify Artifact') {

steps {

dir('Amazon/Amazon-Web') { // Corrected path

script {

echo 'Verifying if the artifact exists...'

sh '''

if [ ! -f target/${ARTIFACT\_NAME} ]; then

echo "Artifact ${ARTIFACT\_NAME} not found. Build failed."

exit 1

fi

'''

}

}

}

}

// Stage 8: Deploy to Tomcat

stage('Deploy') {

steps {

dir('Amazon/Amazon-Web') { // Corrected path

script {

echo 'Deploying to Tomcat...'

sh """

sudo cp target/${ARTIFACT\_NAME} ${TOMCAT\_HOME}/webapps/

"""

echo 'Restarting Tomcat...'

sh """

curl -u ${TOMCAT\_USER}:${TOMCAT\_PASSWORD} -X POST ${TOMCAT\_URL}/shutdown

sleep 5

curl -u ${TOMCAT\_USER}:${TOMCAT\_PASSWORD} -X POST ${TOMCAT\_URL}/start

"""

}

}

}

}

}

post {

always {

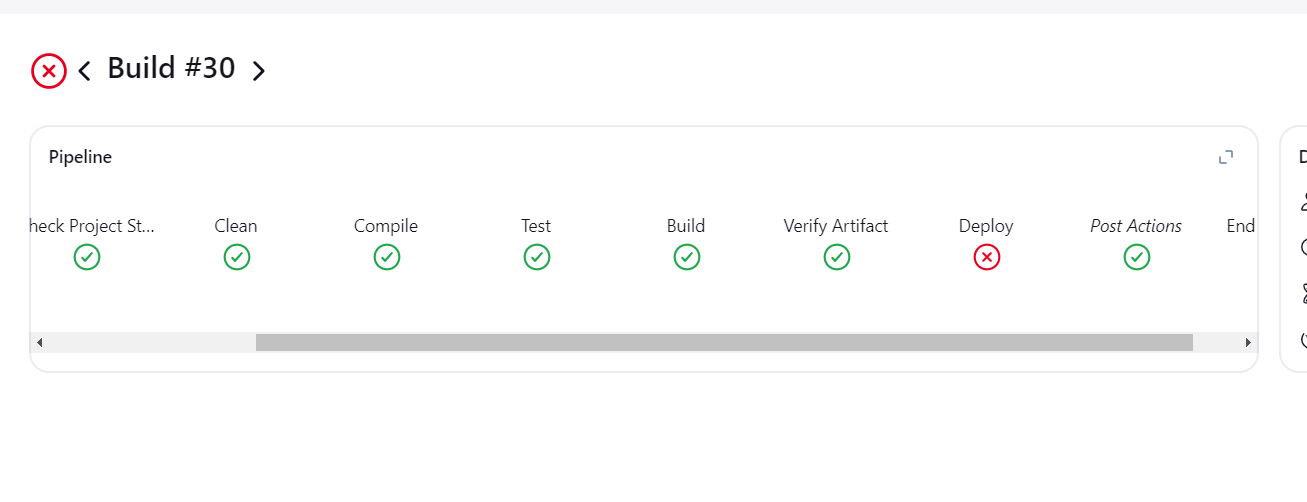
echo 'Pipeline execution complete.'

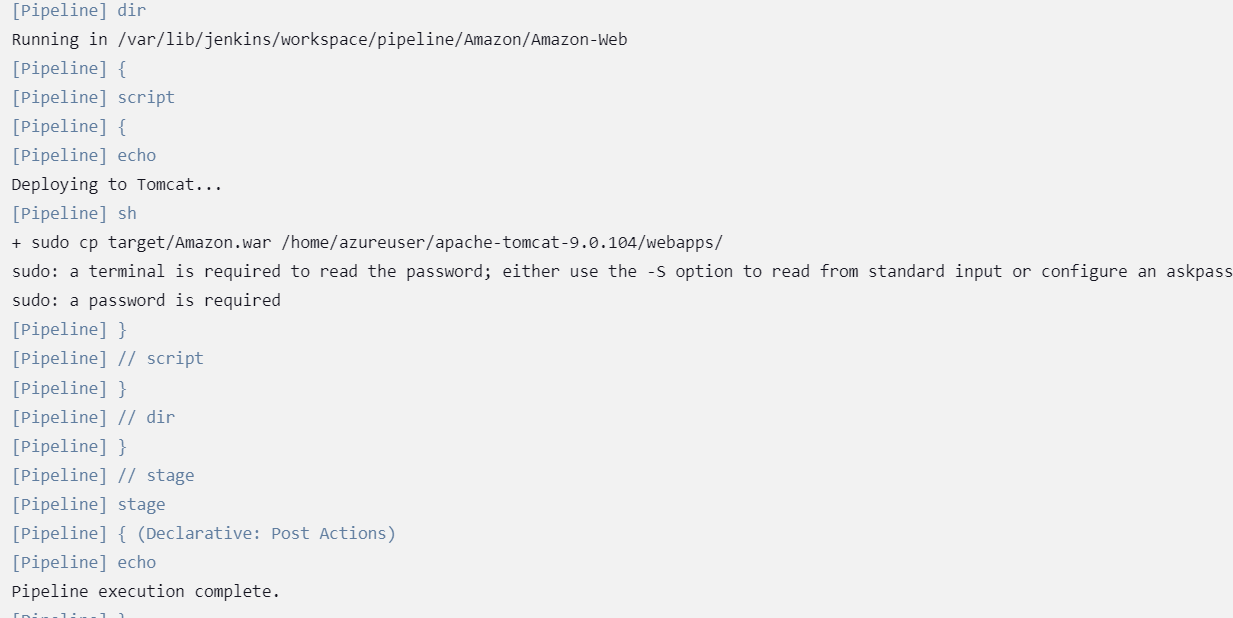
}

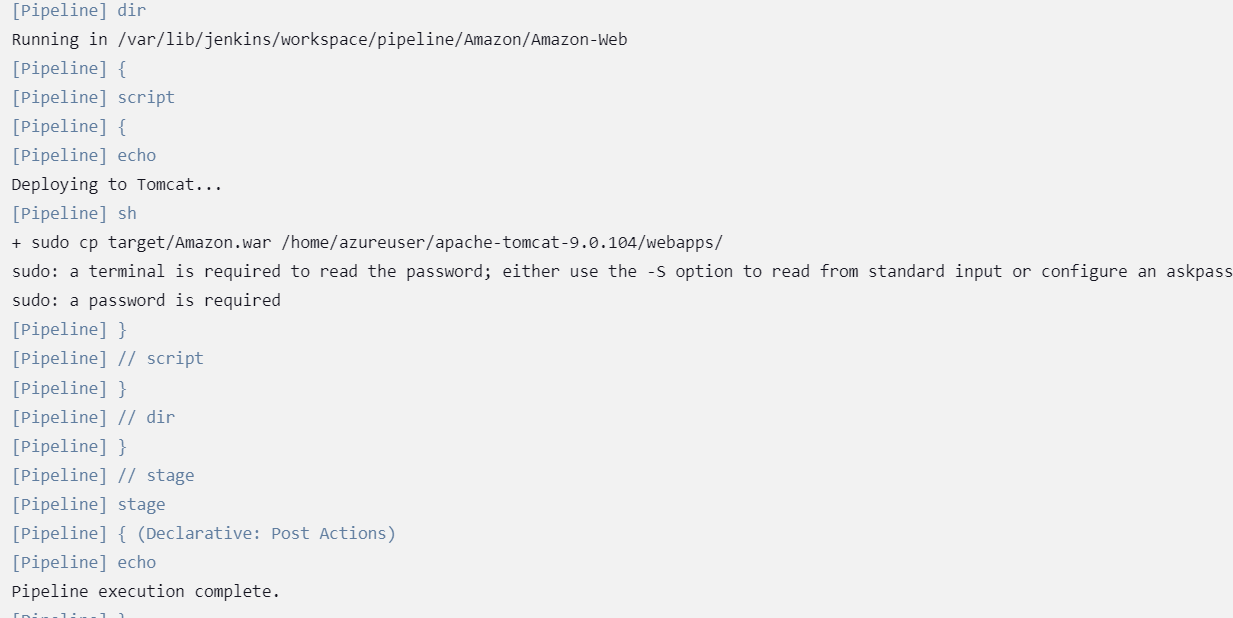
}

}









5.setting up webhook

Steps

5.Setup a GitHub Webhook to trigger Jenkins builds on code changes

1. Prerequisites

Jenkins is accessible from the internet or your GitHub repo (use a tunnel like ngrok if local).

GitHub repository set up.

Jenkins project (Freestyle or Pipeline).

Jenkins has the GitHub plugin installed.

2. Configure Jenkins Job

Option A: For Freestyle Project

Open Jenkins → Your Job → Configure

Under Build Triggers, check ✅ "GitHub hook trigger for GITScm polling"

Option B: For Pipeline Job

Open Jenkins → Your Pipeline Job → Configure

Under Build Triggers, check ✅ "GitHub hook trigger for GITScm polling"

Also make sure:

You're using GitHub in your git SCM section.

If Jenkins is private, expose it with a tunnel (e.g. ngrok http 8080).

🔗 3. Set Up Webhook in GitHub

Go to your GitHub repo → Settings → Webhooks

Click "Add webhook"

Fill in the webhook details:

Payload URL: http://<jenkins-server>/github-webhook/

(e.g., http://jenkins.example.com/github-webhook/ or http://<ngrok-domain>/github-webhook/)

Content type: application/json

Secret: (optional, for security)

Events to trigger: Select “Just the push event” or others like PRs.

Click Add Webhook

