Topics covered so far:

* Microservices
* SVC - **GIT** / Subversion / Mercurial
* SCM - **GitHub** - Shared remote repository / Bitbucket / GitLab
* Build tool - **Maven** - Java build tool / Gradle / Ant
* Continuous integration - **Jenkins** - Automation Server+Build server / Travis CI / Bamboo / Teamcity
* Before Devops, the waterfall model was being followed, devops brings agility. Monolithic architecture was being used, devops brings microservices.
* Phases of Devops lifecycle:
* Continuous integration → Code uploaded by developers to Git is compiled together by Jenkins
* Continuous testing
* Continuous delivery
* Continuous deployment
* Continuous monitoring
* .jar vs .war - Java Archive vs Java web application Archive
* Interviews always start with technicalities on Linux. Study –
* Port numbers
* Https and https protocols
* Dhcp protocols
* Https error / http status code
* 3 way tcp hand shaking
* OSI layer
* IPv4 and IPv5
* Class A and class b IP address
* CIDR
* Language programming methodologies used in various tools functions, eg Jenkins is build in Java
* EMC is a hardware company taken by DELL. Symmetric is a storage device. Clarion is a storage device. Netapp is a hardware company too. Switches, routers by Juniper used in Google data center. Switches, routers by Cisco in AWS data center.
* Types of Jobs in Jenkins –
* Freestyle - Not used at production level basic job
* Maven
* Pipeline
* Multibranch
* Parameterized

Enabling webhook for Jenkins interaction

Goto Github >> Repository >> Settings >> Add webhook >> URL=”<http://publicip-ec2:8080/github-webhook/>”

Steps to work with maven webhook in Jenkins to store artefacts on Jenkins server

To **understand** more about any item like freestyle project in Jenkins, if maven is downloaded it will be visible. Ensuring plugin download enables future pointers to compatible projects

Manage jenkins >> Plugins >> Search Maven integration, Config file provider, Pipeline maven integration, Pipeline maven plugin API, Cobertura >> Install >> Restart Jenkins

Then

1. Goto Github >> check repository project is existing =”makemytrip”

Then

Goto Jenkins >> New item=”makemytrip-maven” >> type=maven project >> OK >> Configure >> General >> Description=”This is a maven project job” >> Source code management >> GIT >> Repo URL=”[http://github.com/pophale-viraj/makemytripApr262025.git](http://github.com/pophale-viraj/makemytrip.git)” >> Credentials=None >> Branches to build=dev >>Triggers >> Github hook trigger for GITScm polling >> Build >> root POM=pom.xml, goals and options=clean package >> SAVE >> Verify webhook is active on Github

Then

Goto IntelliJ IDEA:

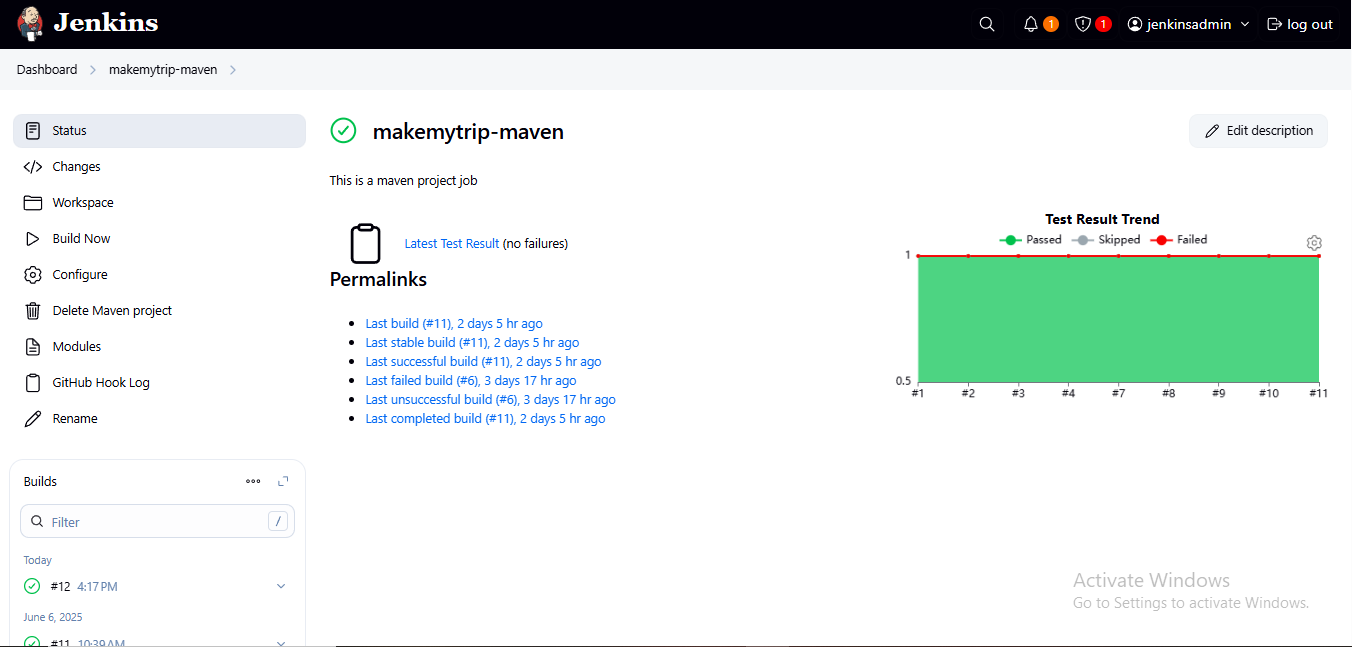
* Git branch
* Git pull origin dev
* Git checkout -b feature/cexchange
* Add java.class=”cexchange.java”
* Mvn clean compile, package
* Mvn spring-boot:run
* <http://localhost>
* Git add –all
* Git commit -m “adding new feature cexchange”
* Git push origin feature/cexchange

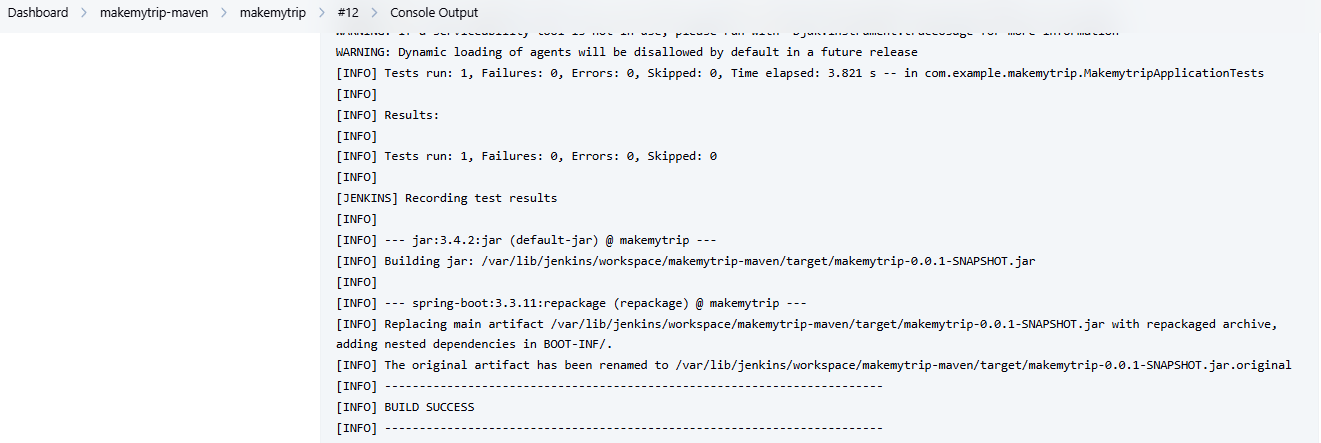
Then

Goto github >> create PR to dev >> Confirm merge >> webhook triggers jenkins job under makemytrip-maven ……artefact gets stored under makemytrip-maven, earlier artefacts were storing in local machine

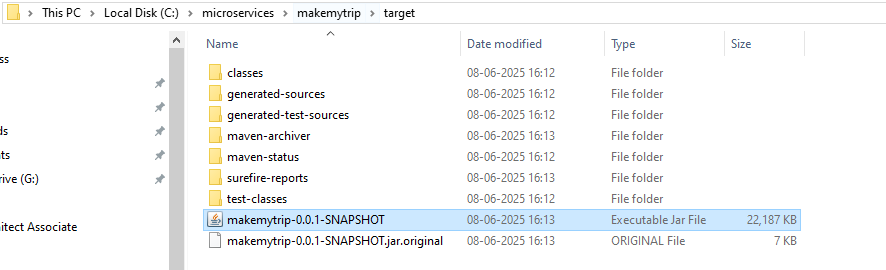
**This is a jenkins artefact storing a code pushed by all developers** ……verify by making changes to xml file and pushing to dev

>> Goto EC2, Github and verify console output of Jenkins

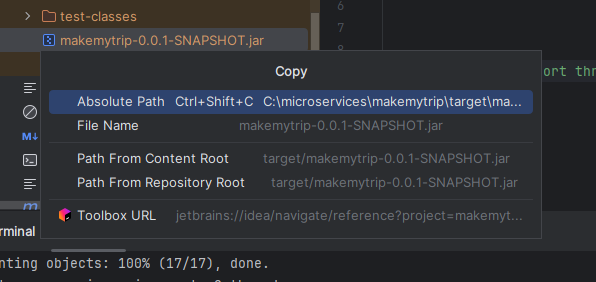
 Webhook trigger



Jenkins artefacts path on jenkins



C drive path of artefacts



Right click on jar file(makemytrip-0.0.1-SNAPSHOT.jar) and copy path/reference

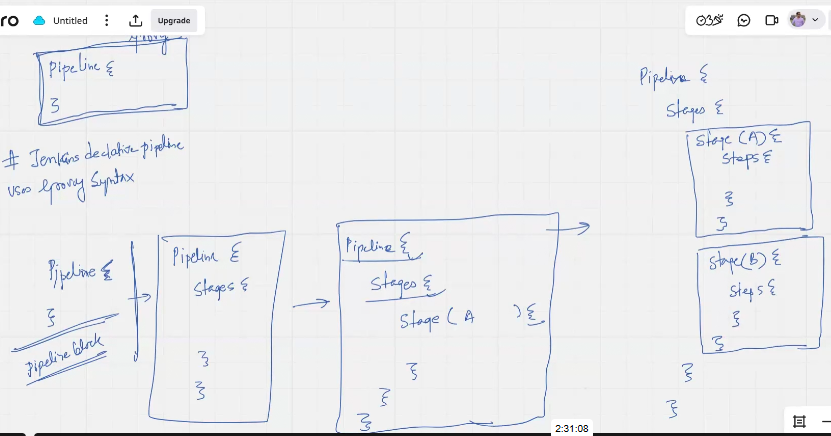
Steps to work with jenkinsfile and pipeline from IntelliJ

Goto IntelliJ

* Git pull origin dev
* Git checkout –b feature/jenkinsfile
* Goto MMT on left pane >> New >> file >> add pipeline

1. Goto Jenkins >> new item=”makemytrip-pipeline” >> Pipeline >> OK >> General >> This is our first pipeline >> Trigger=Github hook trigger for GITScm polling >> Pipeline=pipeline script from SCM >> SCM=git >> Repo URL=”<http://github.com/pophale-viraj/makemytrip.git>” >> Branches to build=dev >> Script path=Jenkinsfile >> SAVE

This is the main CI-CD pipeline file which configs what happens to our code



To note:

# entire pipeline will be written in pipeline block.

# In jenkins we create the pipeline using Jenkins declarative method.

# Jenkins declarative pipeline will use groovy syntax

# the concept of writing the pipeline via code is called as pipeline-as-code

# there will be multiple stage in pipeline, the number of stage depend on project requirement, there is no fixed no of stages.

# if first stage will fail then the pipeline will get abort and it will not run the second stage.

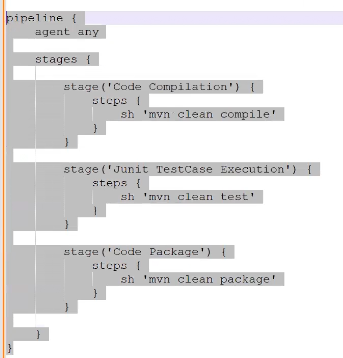
# steps block inside the stage block contains the command to be executed.

# a pipeline is a linear sequence of stages, where every stage will do some work and pass the control to another stage.

Jenkins declarative pipeline uses GROOVY syntax –Highly critical to understand kubernetes/ terraform/ ansible : <https://www.jenkins.io/doc/book/pipeline/#pipeline-1> ……Critical information

<https://github.com/satyam88/greentea/blob/master/may-10> < point C

Basic code to verify in chatgpt:



C:

pipeline {

agent any

options {

buildDiscarder(logRotator(numToKeepStr: '3', artifactNumToKeepStr: '3'))

}

tools {

maven 'mvn\_3.9.9' // Make sure this Maven tool is defined in Jenkins global tools

}

stages {

stage('Code Compilation') {

steps {

echo 'Starting Code Compilation...'

sh 'mvn clean compile'

echo 'Code Compilation Completed Successfully!'

}

}

stage('Code QA Execution') {

steps {

echo 'Running JUnit Test Cases...'

sh 'mvn clean test'

echo 'JUnit Test Cases Completed Successfully!'

}

}

stage('Code Package') {

steps {

echo 'Creating WAR Artifact...'

sh 'mvn clean package'

echo 'WAR Artifact Created Successfully!'

}

}

}

post {

success {

echo '✅ Build completed successfully.'

}

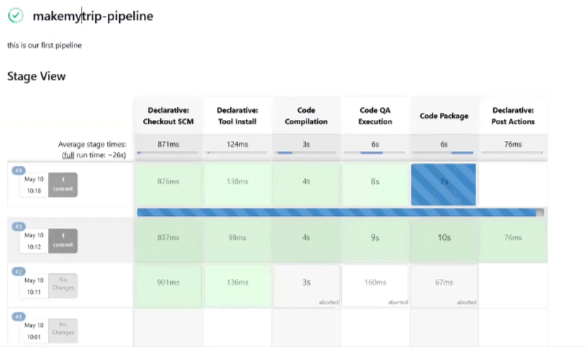
failure {

echo '❌ Build failed.'

}

}

}



Plugin to install

stage

maven

aws

terraform

docker

kubernetes

sonarqube

nexus

ecr

ansible

blueocean