**Pipeline as Code**

* This is expressing CI/CD pipeline in terms of some code/expressions/statements
* This is part of version control i.e. each change done to the steps will have history
* <https://www.jenkins.io/doc/book/pipeline/pipeline-as-code/> for official docs

Azure DevOps

# Starter pipeline

# Start with a minimal pipeline that you can customize to build and deploy your code.

# Add steps that build, run tests, deploy, and more:

# https://aka.ms/yaml

trigger:

- main

pool:

name: Default

steps:

- task: Maven@3

inputs:

mavenPOMFile: 'pom.xml'

goals: 'package'

publishJUnitResults: true

testResultsFiles: '\*\*/surefire-reports/TEST-\*.xml'

testRunTitle: 'unittests'

Jenkins

node("JDK-11-MVN") {

stage("get-code") {

sh " git clone https://github.com/spring-projects/spring-petclinic.git && cd spring-petclinic"

}

stage("build") {

sh "mvn package"

junit '\*\*/surefire-reports/TEST-\*.xml'

}

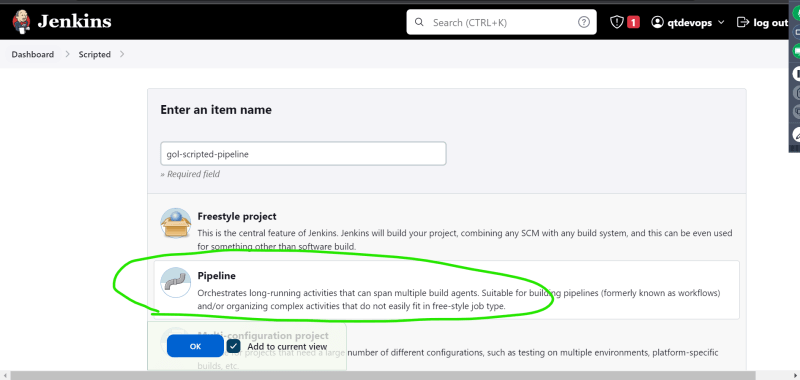
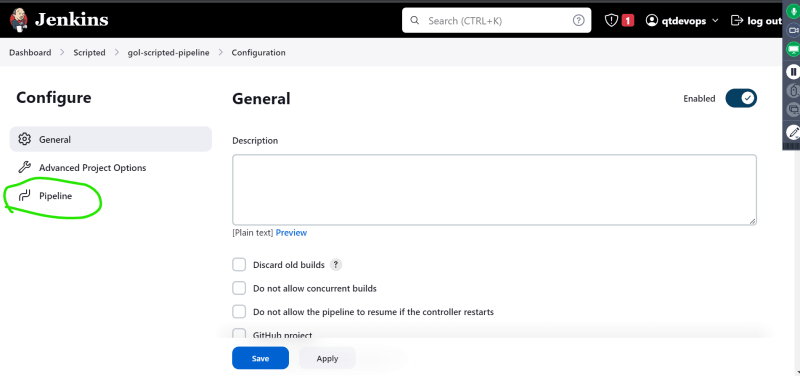
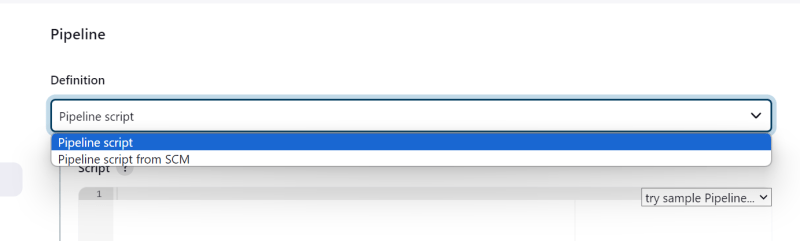
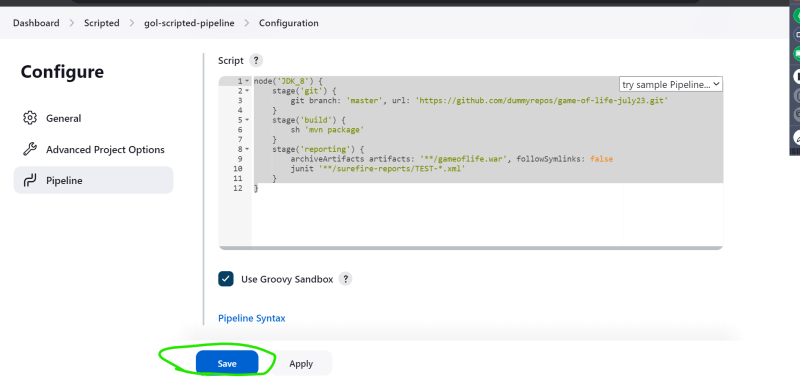
}

* The only manual work would be to go to jenkins and create a pipeline project and configure where your pipeline is

**Pipeline as Code in Jenkins**

* Jenkins has two flavours
  + Scripted Pipeline
    - This was developed where you can execute groovy language directly
  + Declarative Pipeline
    - Jenkins has created a DSL (Domain specific Language) which is mostly inspired from traditional jenkins

**Create a Scripted Pipeline**

* Create a game of life project  
    
  
* Pipeline can be written directly or can be chosen from source code management  
  
* Open pipeline syntax
* Created the following structure  
  

Sample pipeline

node('JDK\_8') {

stage('git') {

git branch: 'master', url: 'https://github.com/dummyrepos/game-of-life-july23.git'

}

stage('build') {

sh 'mvn package'

}

stage('reporting') {

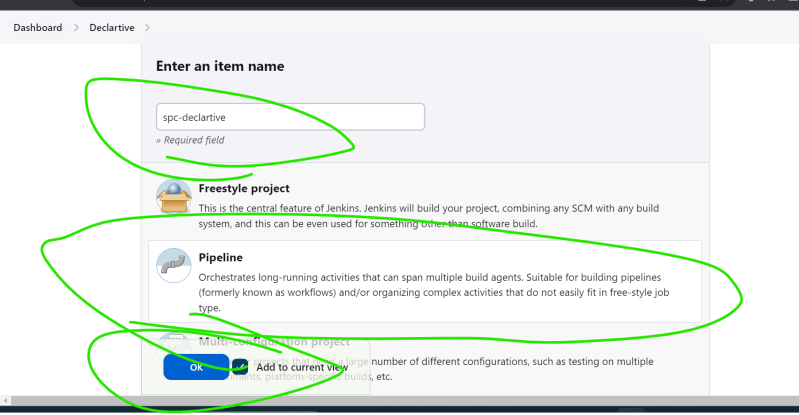
archiveArtifacts artifacts: '\*\*/gameoflife.war', followSymlinks: false

junit '\*\*/surefire-reports/TEST-\*.xml'

}

}

**Create a Declarative Pipeline**

* This has different structure in pipeline not for creating the project  
  
* We have written

pipeline {

agent { label 'JDK-17' }

stages {

stage('git') {

steps {

git branch: 'main', url: 'https://github.com/dummyrepos/spring-petclinic-1.git'

}

}

stage('build') {

steps {

sh 'mvn package'

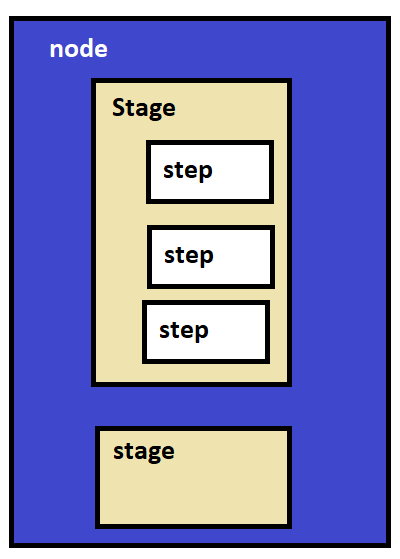
}

}

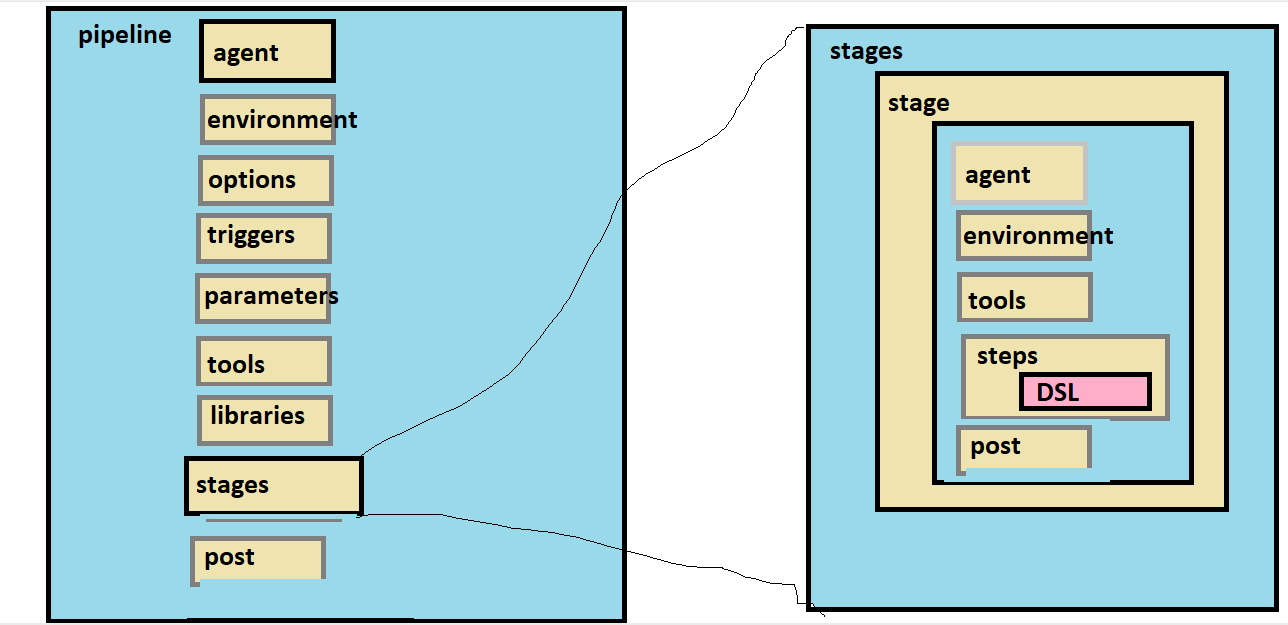
}

}

**Scripted Pipelines**

* Generally, we create a file called as Jenkinsfile
* Basic structure: <https://www.jenkins.io/doc/book/pipeline/syntax/#scripted-pipeline>   
  
* For all the steps <https://www.jenkins.io/doc/pipeline/steps/>
* In scripted and declarative pipelines when we install plugins, we get extra steps.

**Declarative Pipelines**

* Here also we create a file called as Jenkinsfile
* Basic structure: <https://www.jenkins.io/doc/book/pipeline/syntax/#declarative-pipeline> for official docs of jenkins  
  

**Let’s create a declarative pipeline by exploring most options in Declarative pipeline**

* <https://github.com/dummyrepos/spring-petclinic-1> for the repo
* Lets create a develop branch
* We have developed the basic skeleton

pipeline {

agent { label 'JDK-17' }

options {

timeout(time: 30, unit: 'MINUTES')

}

triggers {

pollSCM('\* \* \* \* \*')

}

tools {

jdk 'JDK\_17'

}

stages {

stage('vcs') {

steps {

}

}

stage('build and package') {

steps {

}

}

stage('reporting') {

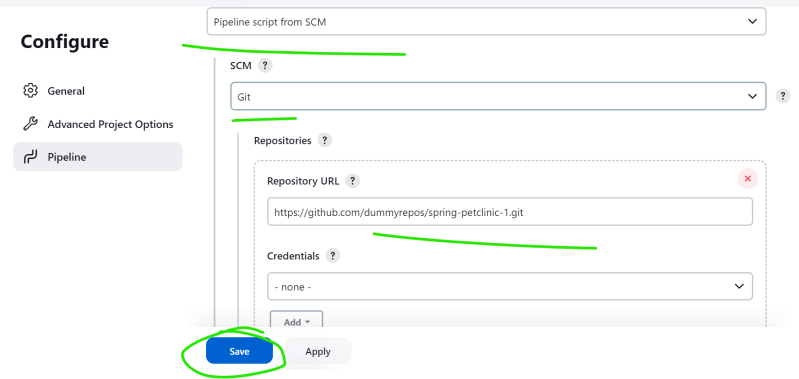
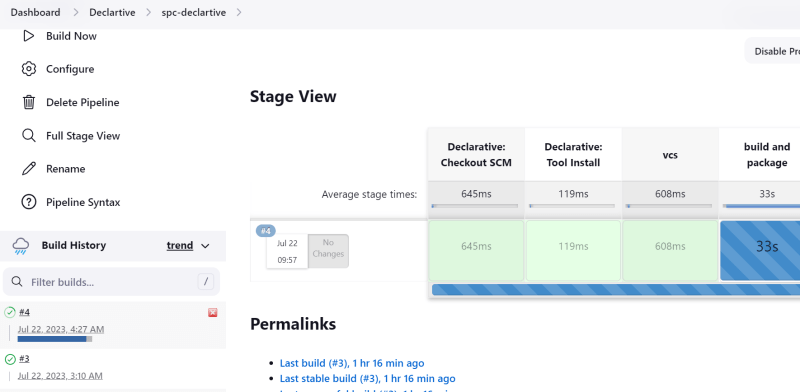
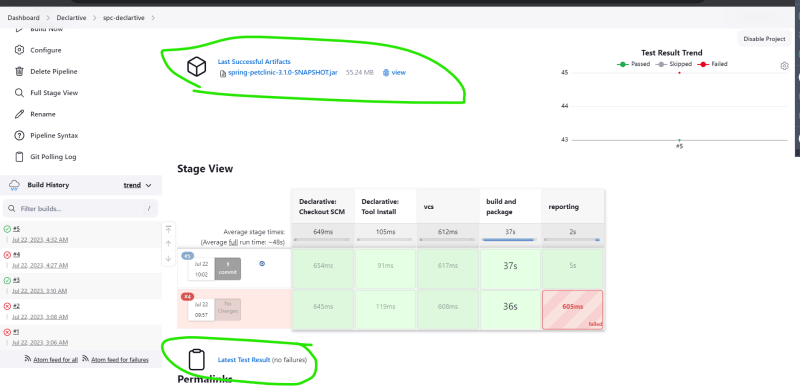
steps {

}

}

}

}

* Now using pipeline steps reference <https://www.jenkins.io/doc/pipeline/steps/> let’s do the build
* git <https://www.jenkins.io/doc/pipeline/steps/git/#git-git>
* and also, other steps as done in the class
* <https://github.com/dummyrepos/spring-petclinic-1/commit/303bcbbf79ca50d4e982dbf0fe017cef5af85101> for the changeset
* Create a project and build now  
    
  
* Build result  
  

**Note:**

* Create a free account in mailtrap <https://mailtrap.io/>
* Exercise: Create a declarative pipeline
  + for spring petclinic
  + for game of life