

## Jenkins Scripted Pipeline (Groovy)

--> For scripted pipeline, we will be using Groovy language

--> Flexibility

--> Customization

--> Reusable components

Syntax:

```
node {  
    stage('git clone'){  
        echo 'git cloning'  
    }  
  
    stage('build'){  
        echo 'build...'  
    }  
}
```

```
ubuntu@ip-172-31-11-116:~/apache-tomcat-11.0.8/bin$ sh startup.sh  
Using CATALINA_BASE:   /home/ubuntu/apache-tomcat-11.0.8  
Using CATALINA_HOME:   /home/ubuntu/apache-tomcat-11.0.8  
Using CATALINA_TMPDIR: /home/ubuntu/apache-tomcat-11.0.8/temp  
Using JRE_HOME:        /usr  
Using CLASSPATH:       /home/ubuntu/apache-tomcat-11.0.8/bin/bootstrap.jar:/home/ubuntu/apache-tomcat-11.0.8/bin/tomcat-juli.jar  
Tomcat started.  
ubuntu@ip-172-31-11-116:~/apache-tomcat-11.0.8/bin$
```

## New Item

Enter an item name

demo\_scripted\_pipeline

Select an item type



### Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



### Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



### Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



### Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

## Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

### Definition

Pipeline script

Script ?

```
1 node {
2     def mvnHome
3     stage('Preparation') { // for display purposes
4         // Get some code from a GitHub repository
5         git 'https://github.com/jglick/simple-maven-project-with-tests.git'
6         // Get the Maven tool.
7         // ** NOTE: This 'M3' Maven tool must be configured
8         // ** in the global configuration.
9         mvnHome = tool 'M3'
10    }
11    stage('Build') {
12        // Run the maven build
13        withEnv(["MVN_HOME=$mvnHome"]) {
14            if (isUnix()) {
15                sh "$MVN_HOME/bin/mvn" -Dmaven.test.failure.ignore clean package
16            }
17        }
18    }
19 }
```

Scripted Pipeline

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

```
node {

    stage('Cloning') {
        echo 'git repo cloning'
    }
    stage('Build') {
        echo 'project build...'
    }
}
```

Scripted Pipeline

### Definition

Pipeline script

Script ?

```
1 node {
2
3     stage('Cloning') {
4         echo 'git repo cloning'
5     }
6     stage('Build') {
7         echo 'project build...'
8     }
9 }
10
```

Scripted Pipeline

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

Apply and Save

## ✓ Console Output

```
Started by user demo
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/demo_scripted_pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning)
[Pipeline] echo
git repo cloning
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
project build...
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Go back to Pipeline --> Configure

## Steps

### Sample Step

git: Git

git ?

Repository URL ?  

https://github.com/Haider7214/SpringApp.git

Branch ?  

main

Credentials ?  

- none -

+ Add

☒ Include in polling? ?

☒ Include in changelog? ?

Generate Pipeline Script

git branch: 'main', url: 'https://github.com/Haider7214/SpringApp.git'

For Declarative it is different, for Scripted it is totally different  
git branch: 'main', url: 'https://github.com/Haider7214/SpringApp.git'

```
node {
  def mvnPath
  stage('git clone') {
    git branch: 'main', url: 'https://github.com/Haider7214/SpringApp.git'
  }
  stage('Maven Build') {
    def mvnHome = tool name:'maven:3.9.10', type='maven';
    mvnPath = "${mvnHome}/bin/mvn";
    sh "${mvnPath} clean package";
    echo 'maven build success';
  }
}
```

Define your Pipeline using Groovy directly or pull it from source control.

#### Definition

Pipeline script

Script ?

```
1 node {  
2     def mvnPath  
3     stage('git clone') {  
4         git branch: 'main', url: 'https://github.com/Haider7214/SpringApp.git'  
5     }  
6     stage('Maven Build') {  
7         def mvnHome = tool name: 'maven:3.9.10', type: 'maven';  
8         mvnPath = "${mvnHome}/bin/mvn";  
9         sh "${mvnPath} clean package";  
10        echo 'maven build success';  
11    }  
12 }  
13
```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

Apply and Save

Manage Jenkins --> Tools

That's the name we have given for Maven

## Maven installations

Maven installations ^

[✎](#) Edited

Add Maven

### ≡ Maven

Name

maven-3.9.10

☒ Install automatically ?

### ≡ Install from Apache

Version

3.9.10

Add Installer ▾

Add Maven

## Failed

```
at PluginClassLoader for workflow-cps//com.cloudbees.groovy.cps.impl.FunctionCallBlock$ContinuationImpl.fixArg(FunctionCallBlock.java:103)
at java.base/jdk.internal.reflect.DirectMethodHandleAccessor.invoke(DirectMethodHandleAccessor.java:183)
at java.base/java.lang.reflect.Method.invoke(Method.java:580)
at PluginClassLoader for workflow-cps//com.cloudbees.groovy.cps.impl.ContinuationPtr$ContinuationImpl.receive(ContinuationPtr.java:21)
at PluginClassLoader for workflow-cps//com.cloudbees.groovy.cps.impl.ConstantBlock.eval(ConstantBlock.java:21)
at PluginClassLoader for workflow-cps//com.cloudbees.groovy.cps.Next.step(Next.java:83)
at PluginClassLoader for workflow-cps//com.cloudbees.groovy.cps.Continuable.run0(Continuable.java:147)
at PluginClassLoader for workflow-cps//org.jenkinsci.plugins.workflow.cps.SandboxContinuable.access$001(SandboxContinuable.java:18)
at PluginClassLoader for workflow-cps//org.jenkinsci.plugins.workflow.cps.SandboxContinuable.run0(SandboxContinuable.java:18)
at PluginClassLoader for workflow-cps//org.jenkinsci.plugins.workflow.cps.CpsThread.runNextChunk(CpsThread.java:180)
at PluginClassLoader for workflow-cps//org.jenkinsci.plugins.workflow.cps.CpsThreadGroup.run(CpsThreadGroup.java:419)
at PluginClassLoader for workflow-cps//org.jenkinsci.plugins.workflow.cps.CpsThreadGroup$2.call(CpsThreadGroup.java:327)
at PluginClassLoader for workflow-cps//org.jenkinsci.plugins.workflow.cps.CpsThreadGroup$2.call(CpsThreadGroup.java:292)
at PluginClassLoader for workflow-cps//org.jenkinsci.plugins.workflow.cps.CpsVmExecutorService.lambda$wrap$4(CpsVmExecutorService.java:51)
at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:317)
at hudson.remoting.SingleLaneExecutorService$1.run(SingleLaneExecutorService.java:139)
at jenkins.util.ContextResettingExecutorService$1.run(ContextResettingExecutorService.java:28)
at jenkins.security.ImpersonatingExecutorService$1.run(ImpersonatingExecutorService.java:68)
at jenkins.util.ErrorLoggingExecutorService.lambda$wrap$0(ErrorLoggingExecutorService.java:51)
at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:572)
at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:317)
at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1144)
at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:642)
at PluginClassLoader for workflow-cps//org.jenkinsci.plugins.workflow.cps.CpsVmExecutorService$1.call(CpsVmExecutorService.java:51)
at PluginClassLoader for workflow-cps//org.jenkinsci.plugins.workflow.cps.CpsVmExecutorService$1.call(CpsVmExecutorService.java:51)
at org.codehaus.groovy.runtime.GroovyCategorySupport$ThreadCategoryInfo.use(GroovyCategorySupport.java:136)
at org.codehaus.groovy.runtime.GroovyCategorySupport.use(GroovyCategorySupport.java:275)
at PluginClassLoader for workflow-cps//org.jenkinsci.plugins.workflow.cps.CpsVmExecutorService.lambda$categoryThreadFactory$0(CpsVmExecutorService.java:51)
at java.base/java.lang.Thread.run(Thread.java:1583)
```

Finished: FAILURE

I change the script

```
node {
  def mvnPath
  stage('git clone') {
    git branch: 'main', url: 'https://github.com/Haider7214/SpringBoot.git'
  }
  stage('Maven Build') {
    def mvnHome = tool name:'maven-3.9.10', type='maven';
    mvnPath = "${mvnHome}/bin/mvn";
    sh "${mvnPath} clean package";
    echo 'maven build success';
  }
}
```

Again Build failed

```
node {
  def mvnPath

  stage('git clone') {
    git branch: 'main', url: 'https://github.com/Haider7214/SpringBoot.git'
  }
  stage('Maven Build') {
    def mvnHome=tool name:'maven-3.9.10', type:'maven';
    mvnPath="${mvnHome}/bin/mvn";
    sh "${mvnPath} clean package";
    echo 'maven build success';
  }
}
```

This time build was successful

```

Progress (1): 357/465 kB
Progress (1): 374/465 kB
Progress (1): 390/465 kB
Progress (1): 406/465 kB
Progress (1): 423/465 kB
Progress (1): 439/465 kB
Progress (1): 456/465 kB
Progress (1): 465 kB

Downloaded from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-starter/2.7.0/spring-boot-starter-2.7.0.jar
[INFO] Replacing main artifact /var/lib/jenkins/workspace/demo_scripted_pipeline/target/demo_scripted_pipeline.jar with /var/lib/jenkins/workspace/demo_scripted_pipeline/target/demo_scripted_pipeline.jar
[INFO] The original artifact has been renamed to /var/lib/jenkins/workspace/demo_scripted_pipeline/target/demo_scripted_pipeline.jar.original
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 13.591 s
[INFO] Finished at: 2025-07-05T17:05:47Z
[INFO] -----
[Pipeline] echo
maven build success
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

### Jenkins Scripted (Groovy) Pipeline with Git + Maven

```

node {
    def mvnPath

    stage('git clone') {
        git branch: 'main', url: 'https://github.com/Haider7214/SpringBoot.git'
    }
    stage('Maven Build') {
        def mvnHome=tool name:'maven-3.9.10', type:'maven';
        mvnPath="${mvnHome}/bin/mvn";
        sh "${mvnPath} clean package";
        echo 'maven build success';
    }
}

```

### Multi Branch Pipeline in Jenkins

In Git repo, there will be multiple branches -> main branch, master, developer, release, feature

During git cloning, we got to specify which branch we are cloning from. Same Jenkins pipeline cannot work for all branches then if we are specifying the branch name explicitly

--> Creating different Jenkins pipeline for every branch would be difficult

--> We can create one common pipeline and build the code from multiple branches at a time with the help of "Multi-branch pipeline" concept

--> When we create Multi-branch pipeline, it will scan all the branches in given Github repo and execute pipeline for all the branches

Whichever branch only has the code change, only that branch will be built



Note: Whenever we run Multi-branch pipeline for second time, it will execute the pipeline for which code changes/commits have been done.