In a programming context, a build is a version of a program that, as a rule, is a pre-release version and is identified by a build number rather than by a release number.

Simply put, a software build is a set of executable code that is ready for use by customers. The DevOps team compiles the source code, such as code in Java or C++, into binaries to make sure it's functional and test code quality before committing it.

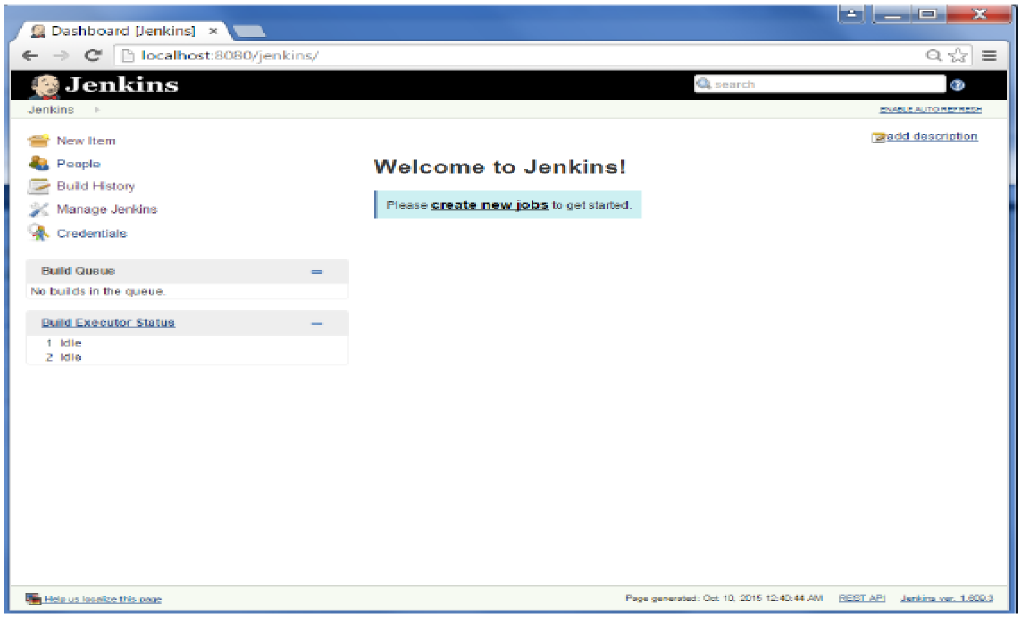
Software is updated regularly until the maker decides to discontinue supporting it. This may entail a series of builds and many software releases to the public.

Iterative (repeated) builds, otherwise known as continuous integration, are an essential part of an optimal development process where application components are collected and repeatedly compiled for testing purposes to ensure a reliable final product.

Additionally, build tools enable developers to automate some programming tasks to further streamline the process

For this exercise, we will create a job in Jenkins which picks up a simple HelloWorld application, builds and runs the java program.

Step 1 − Go to the Jenkins dashboard and Click on New Item



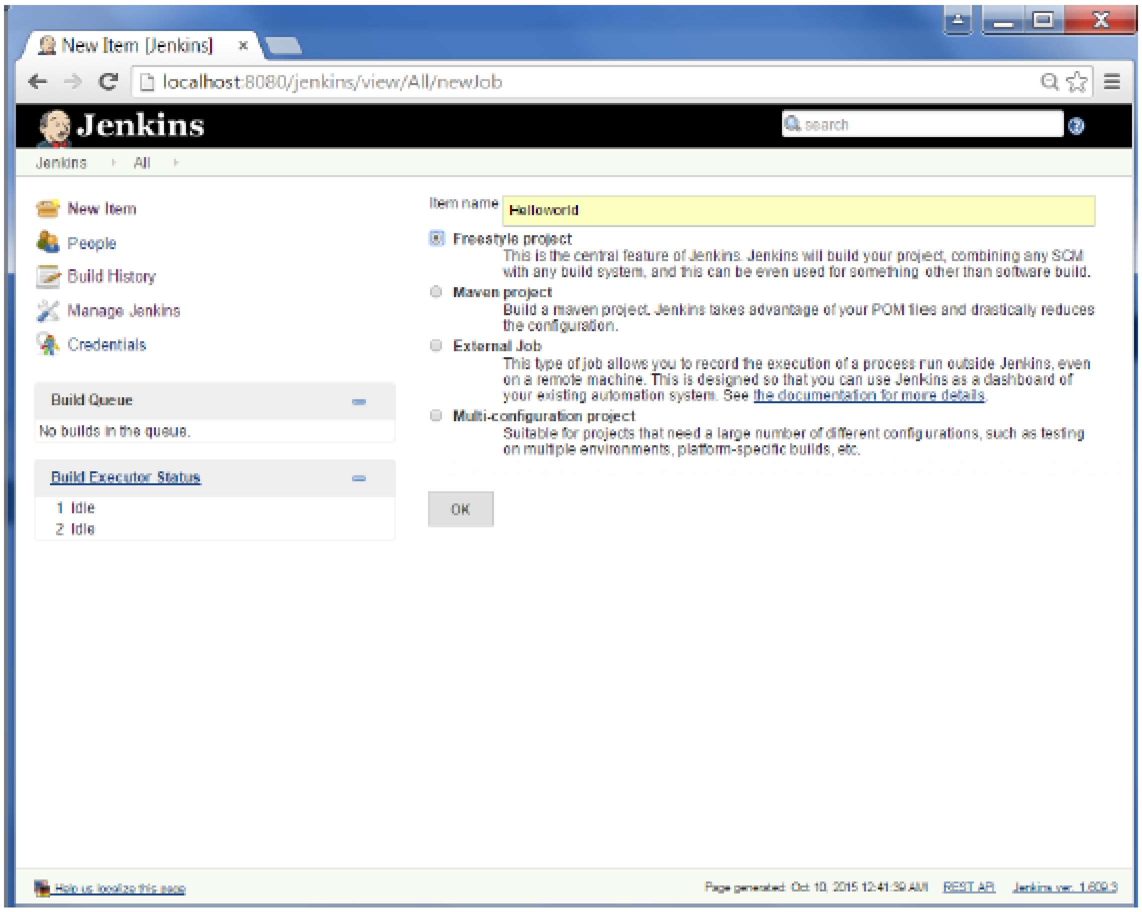
Welcome to Jenkins!

Please create new jobs to get started.

No builds in the queue.

Page generated: Oct 10, 2015 12:40:44 AM

Step 2 − In the next screen, enter the Item name, in this case we have named it Helloworld. Choose the ‘Freestyle project option’



No builds in the queue.

1 Idle
2 Idle

Item name: Helloworld

[✓ Freestyle project]
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

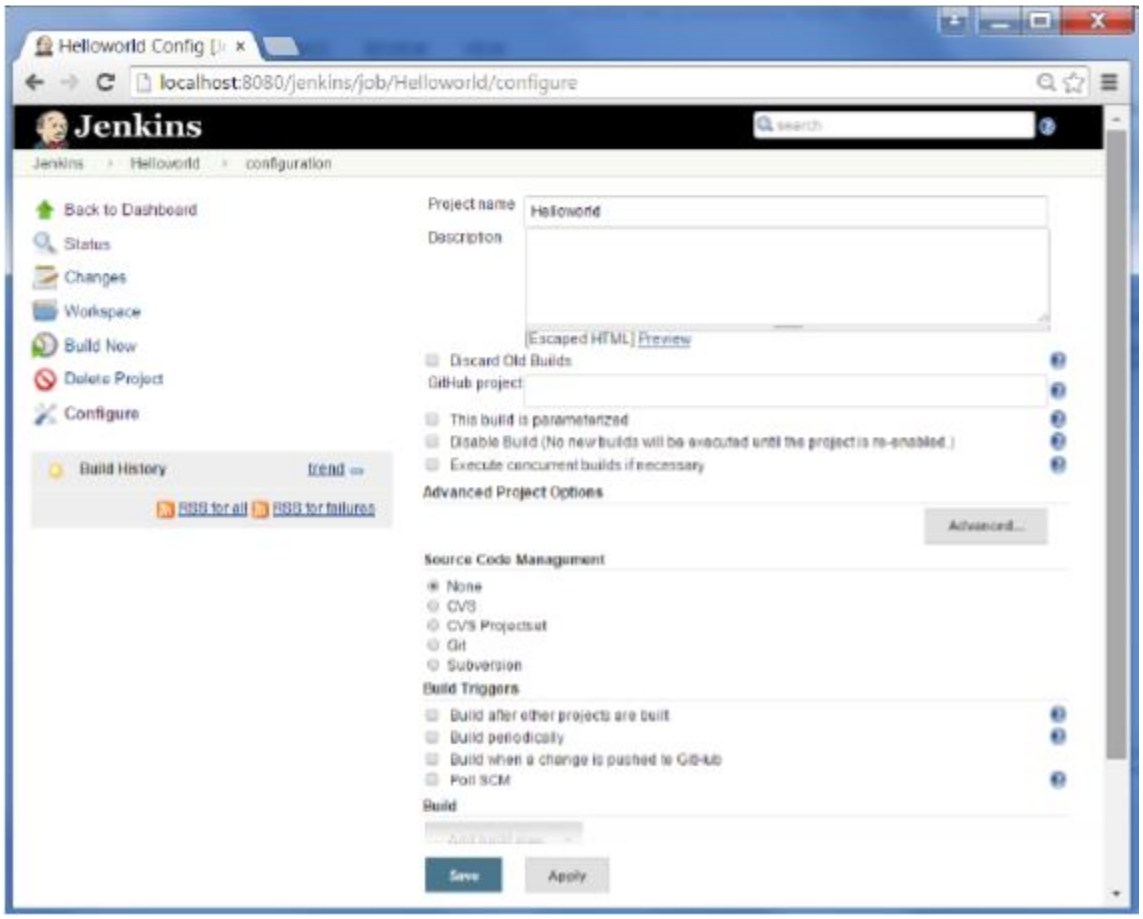
External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See the documentation for more details.

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

Help us localize this page

Page generated: Oct 10, 2015 12:41:59 AM REST API Jenkins ver. 1.609.3

Step 3 − The following screen will come up in which you can specify the details of the job.



Project name: Helloworld

Description

[ ] Discard Old Builds

GitHub project

[ ] This build is parameterized

[ ] Disable Build (No new builds will be executed until the project is re-enabled.)

[ ] Execute concurrent builds if necessary

Advanced Project Options

Source Code Management

[✓ None]
CVS
CVS Projectset
Git
Subversion

Build Triggers

[ ] Build after other projects are built

[ ] Build periodically

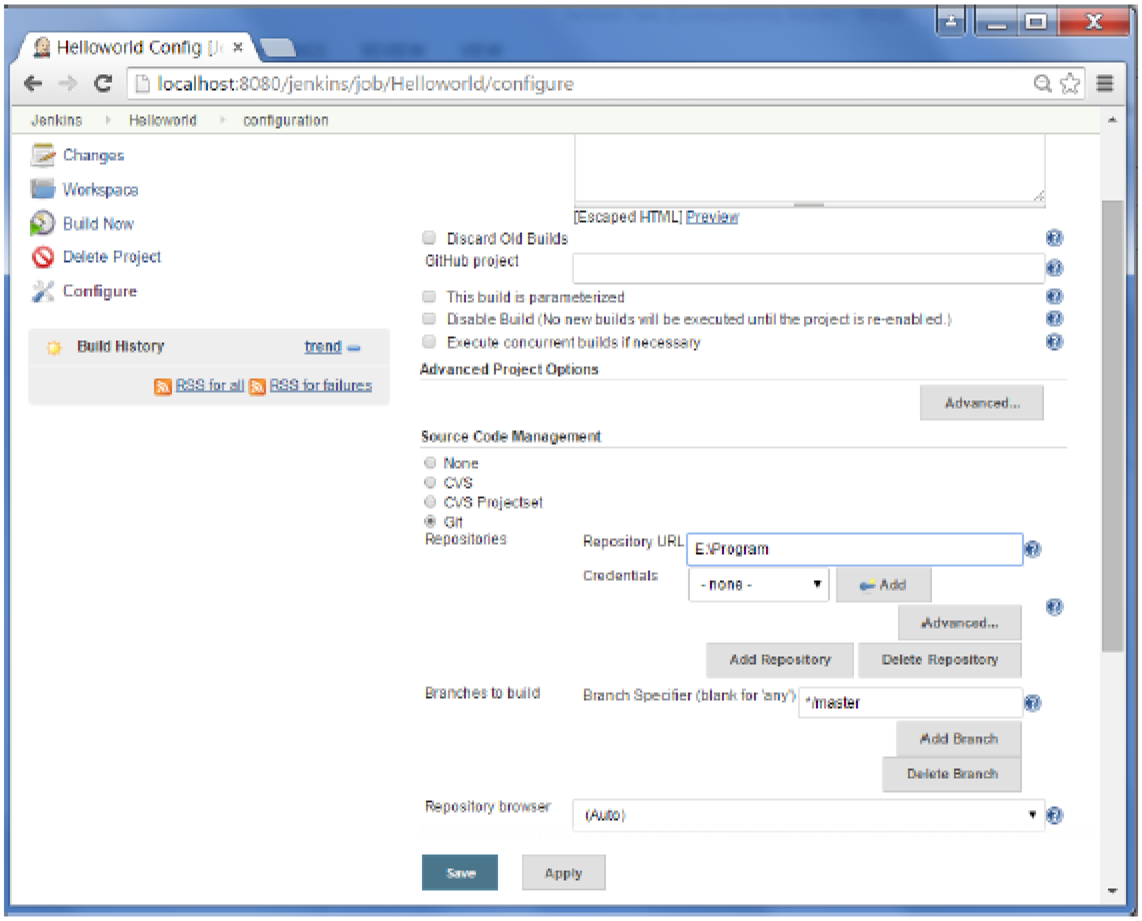
[ ] Build when a change is pushed to GitHub

[ ] Poll SCM

Build

Step 4 − We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a 'HelloWorld.java' file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

Note − If you repository if hosted on Github, you can also enter the url of that repository here. In addition to this, you would need to click on the Add button for the credentials to add a user name and password to the github repository so that the code can be picked up from the remote repository.



[Escaped HTML] Preview

Discard Old Builds

GitHub project

This build is parameterized

Disable Build (No new builds will be executed until the project is re-enabled.)

Execute concurrent builds if necessary

Advanced Project Options

Advanced...

Source Code Management

None

CVS

CVS Projectset

[✓ Git]

Repositories — Repository URL: E:\Program

Credentials: - none -   Add

Advanced...

Add Repository   Delete Repository

Branches to build — Branch Specifier (blank for 'any'): */master

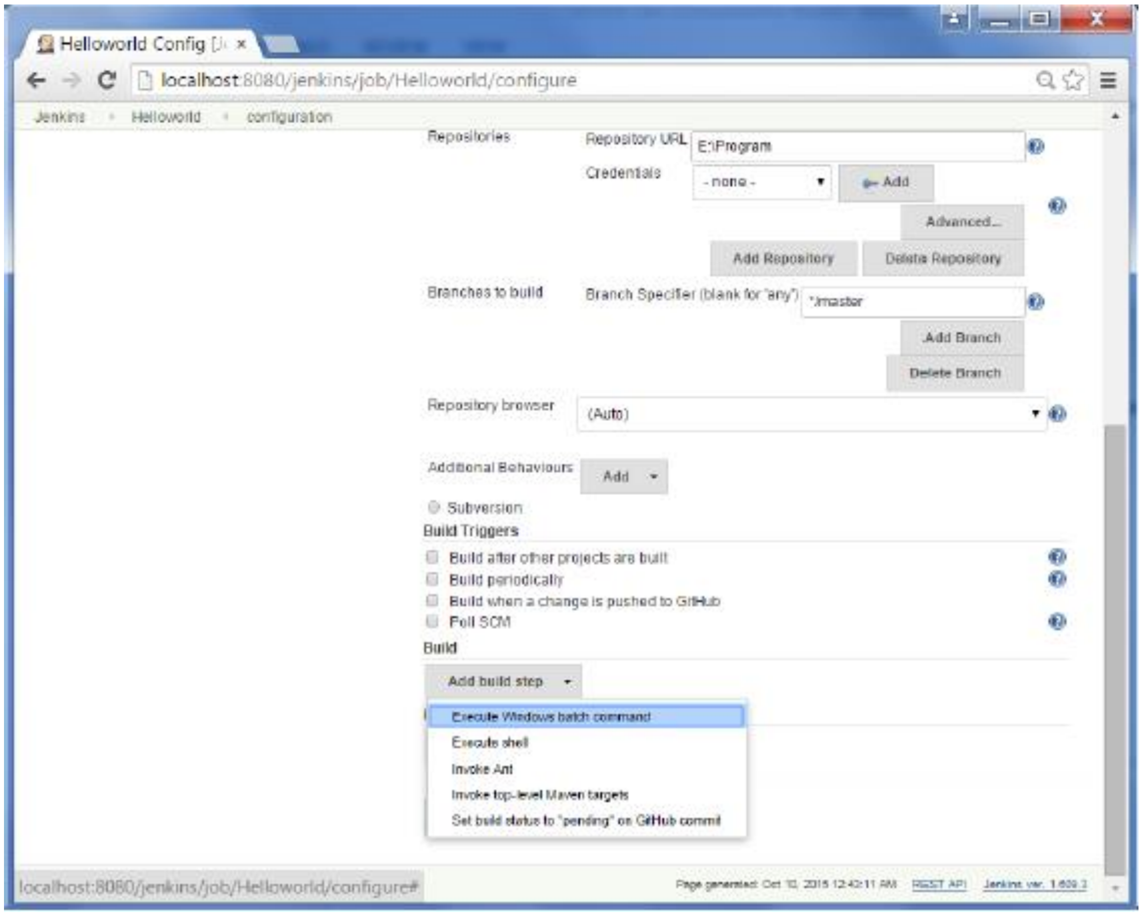
Add Branch

Delete Branch

Repository browser: (Auto)

Save   Apply

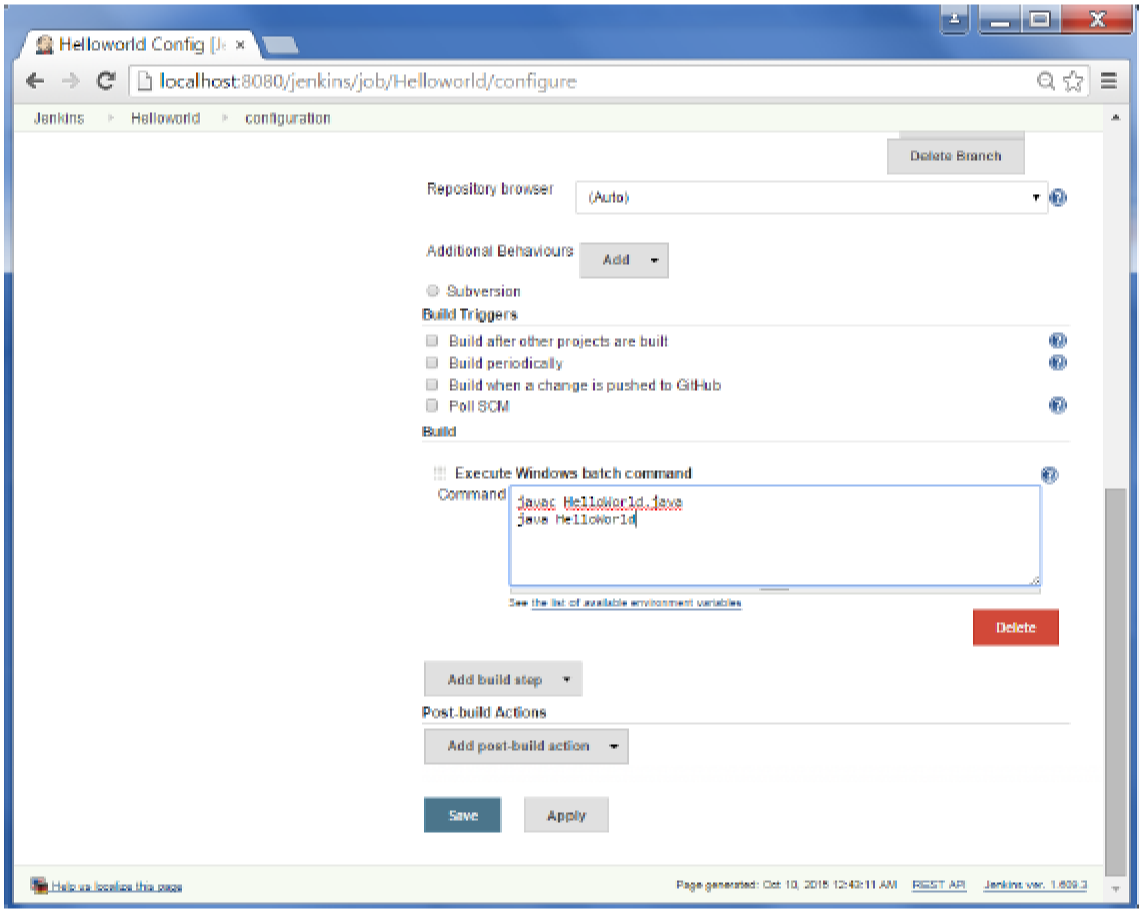
Step 5 – Now go to the Build section and click on Add build step → Execute Windows batch command


Helloworld Config [J
localhost:8080/jenkins/job/Helloworld/configure
Jenkins
Helloworld
configuration
Repositories
Add
Advanced...
Add Repository
Delete Repository
Branches to build
Add Branch
Delete Branch
Additional Behaviours
Add
Subversion
Build Triggers
Build after other projects are built
Build periodically
Build when a change is pushed to GitHub
Poll SCM
Build
Add build step
Execute Windows batch command
Execute shell
Invoke Ant
Invoke top-level Maven targets
Set build status to "pending" on GitHub commit
localhost:8080/jenkins/job/Helloworld/configure#
Page generated: Oct 10, 2015 12:42:11 AM
REST API
Jenkins ver. 1.609.2

Repository URL: E:\Program
Credentials: - none -
Branch Specifier (blank for 'any'): */master
Repository browser: (Auto)

Step 6 – In the command window, enter the following commands and then click on the Save button.

```
Javac HelloWorld.java
Java HelloWorld
```



Repository browser (Auto)

Additional Behaviours

Subversion

Build Triggers

- Build after other projects are built
- Build periodically
- Build when a change is pushed to GitHub
- Poll SCM

Build

Execute Windows batch command

Command

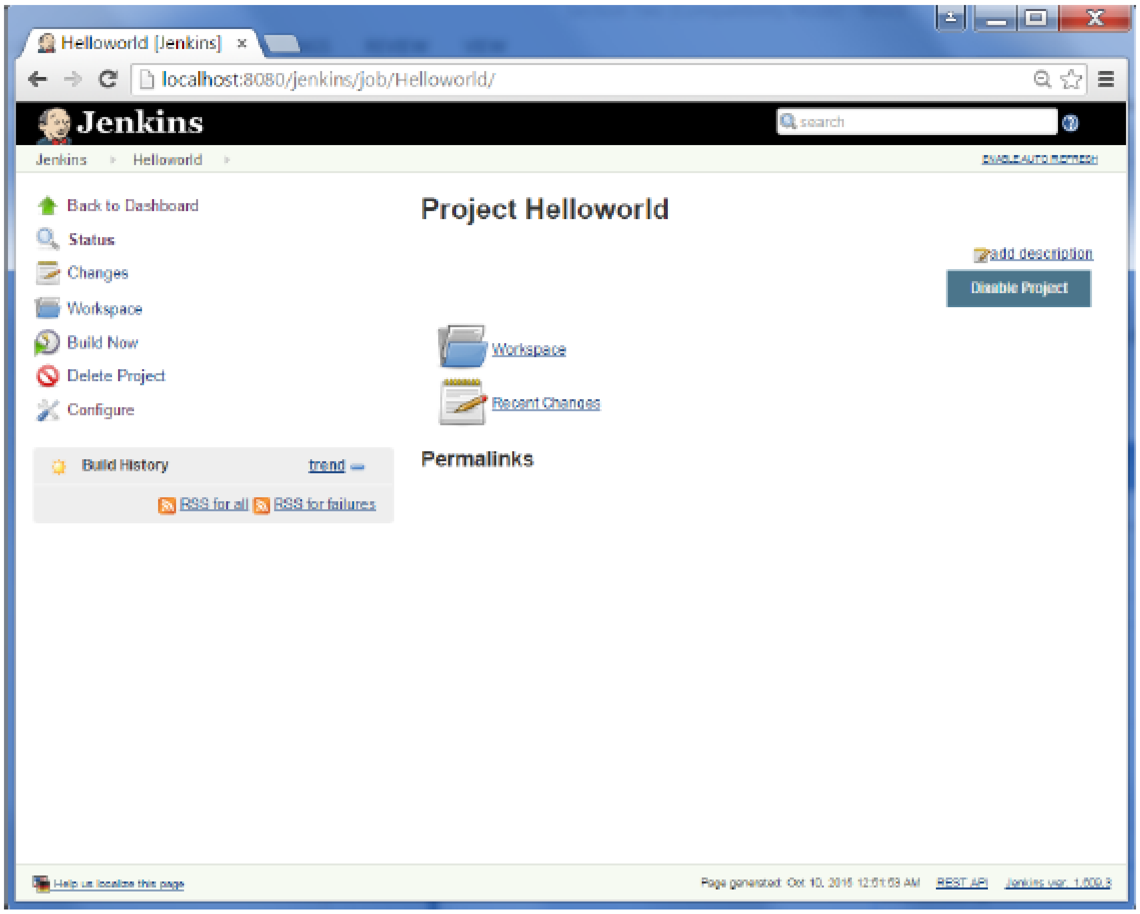
```
javac HelloWorld.java
java HelloWorld
```

See the list of available environment variables

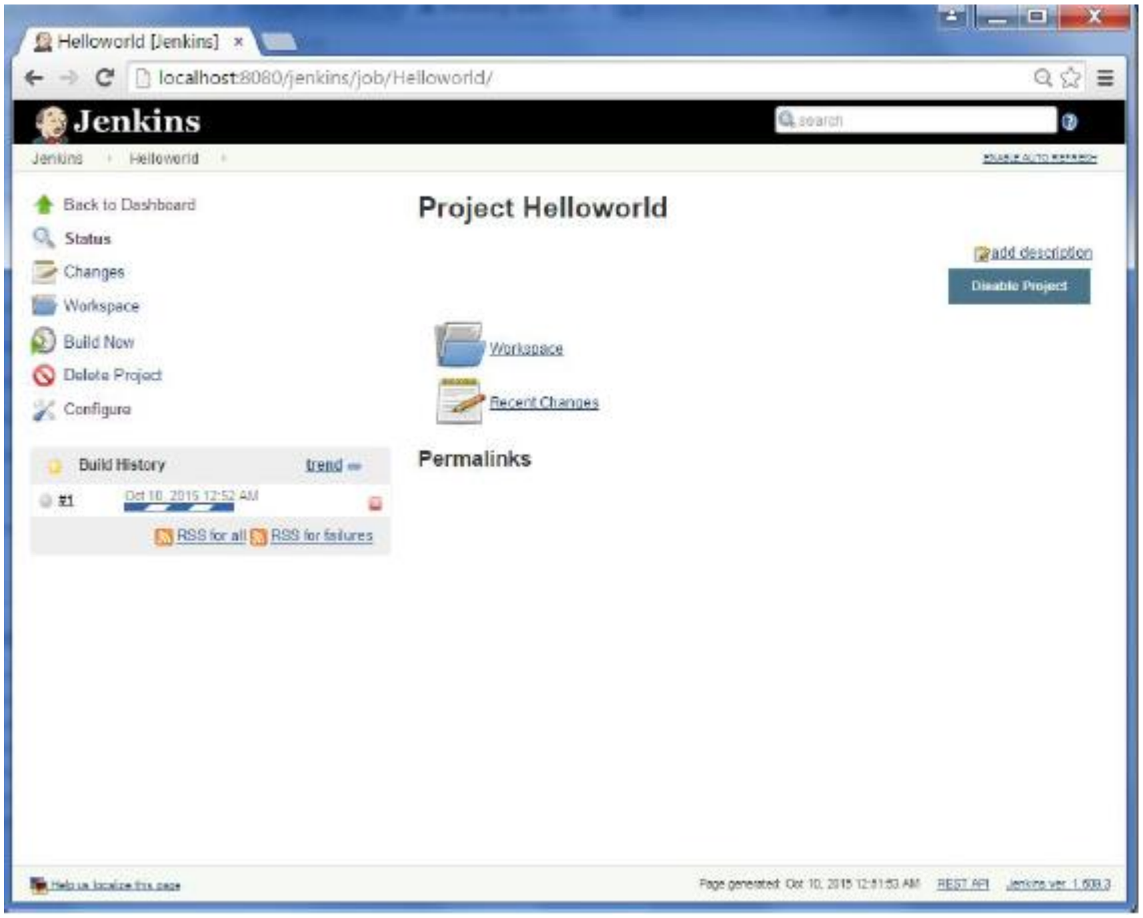
Post-build Actions

Page generated: Oct 10, 2015 12:43:11 AM

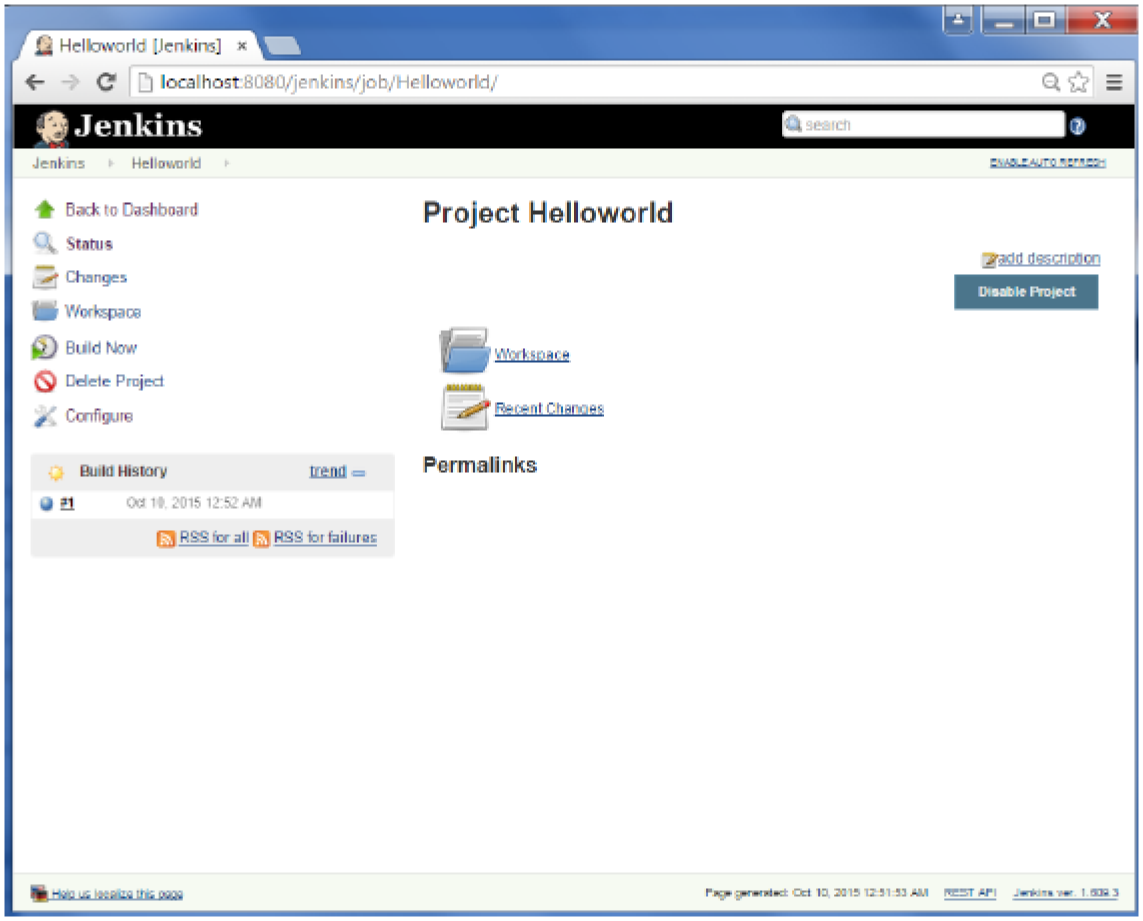
Step 7 − Once saved, you can click on the Build Now option to see if you have successfully defined the job.



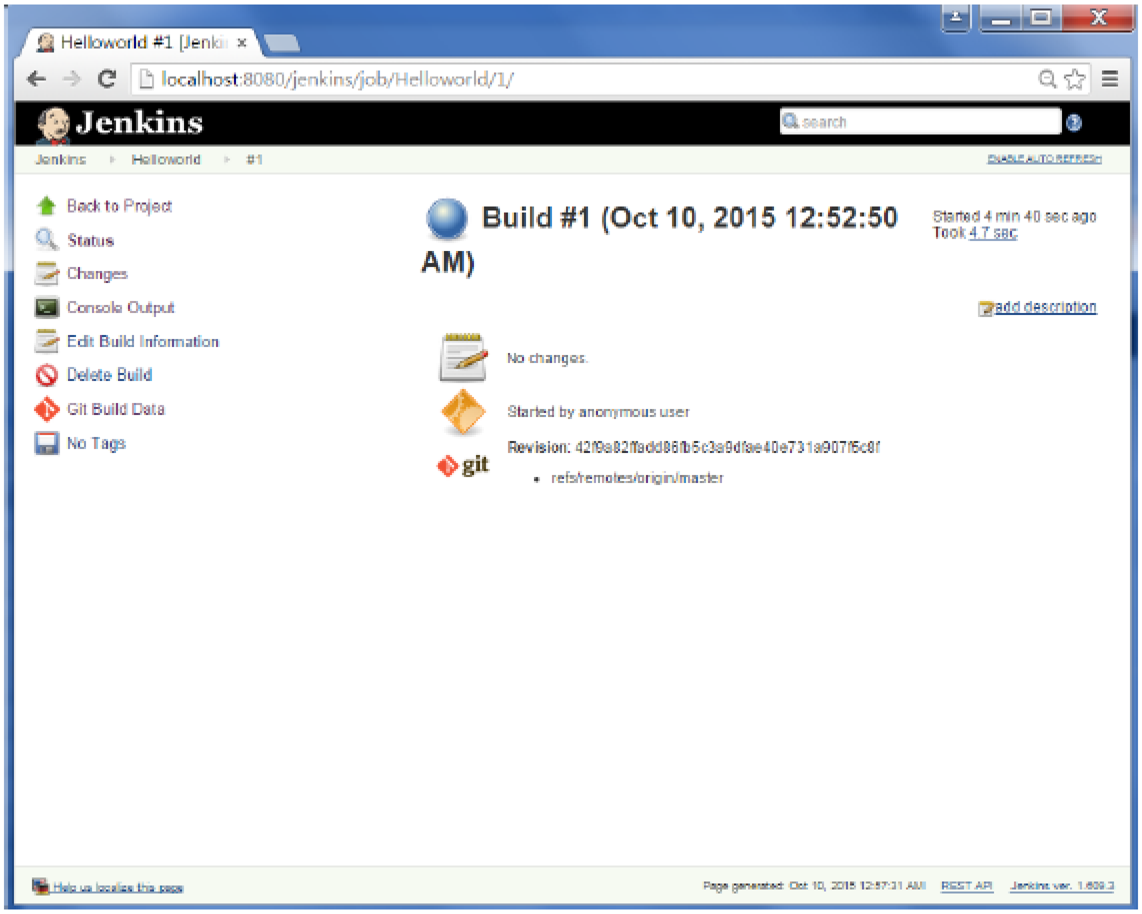
Step 8 − Once the build is scheduled, it will run. The following Build history section shows that a build is in progress.



Step 9 − Once the build is completed, a status of the build will show if the build was successful or not. In our case, the following build has been executed successfully. Click on the #1 in the Build history to bring up the details of the build.



Step 10 − Click on the Console Output link to see the details of the build



Build #1 (Oct 10, 2015 12:52:50 AM)

Started 4 min 40 sec ago
Took 4.7 sec

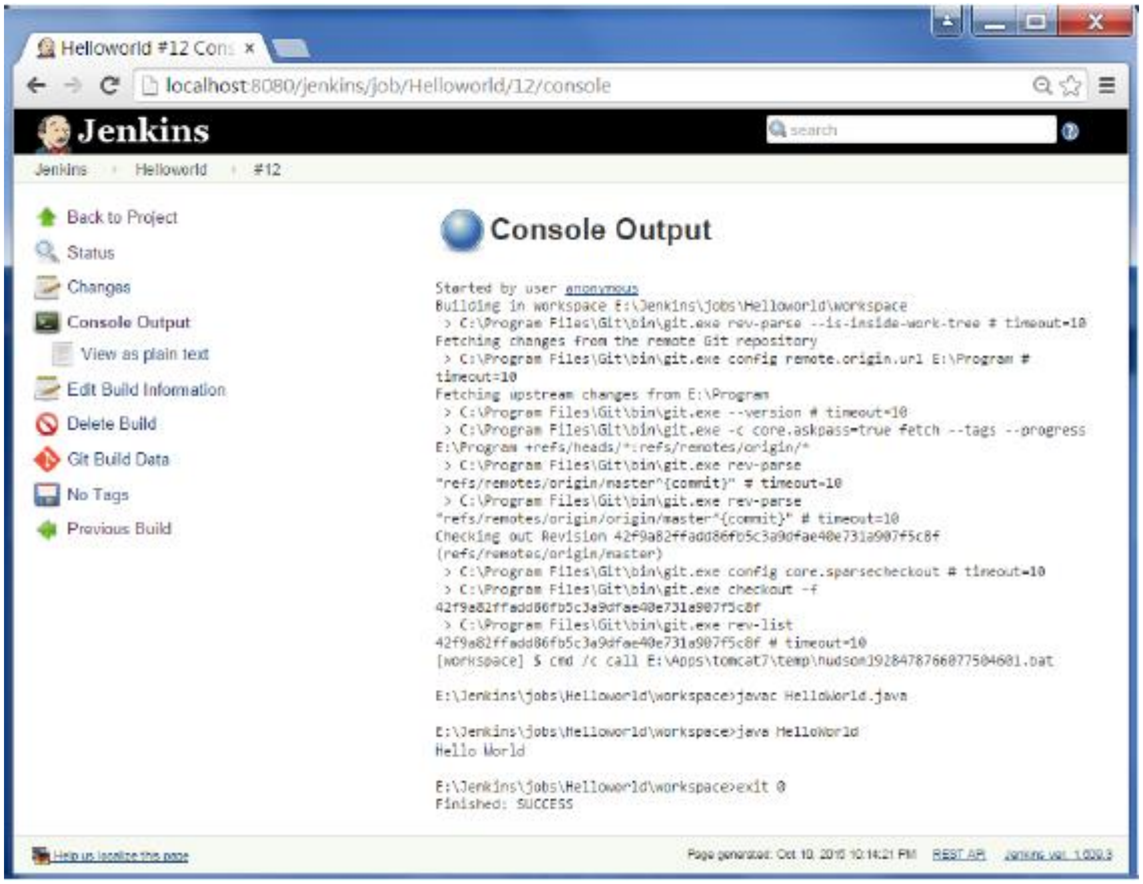
No changes.

Started by anonymous user

Revision: 42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f

- refs/remotes/origin/master

Page generated: Oct 10, 2015 12:57:31 AM



```
Started by user anonymous
Building in workspace E:\Jenkins\jobs\Helloworld\workspace
 > C:\Program Files\Git\bin\git.exe rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > C:\Program Files\Git\bin\git.exe config remote.origin.url E:\Program #
timeout=10
Fetching upstream changes from E:\Program
 > C:\Program Files\Git\bin\git.exe --version # timeout=10
 > C:\Program Files\Git\bin\git.exe -c core.askpass=true fetch --tags --progress
E:\Program +refs/heads/*:refs/remotes/origin/*
 > C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/master^{commit}" # timeout=10
 > C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/origin/master^{commit}" # timeout=10
Checking out Revision 42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f
(refs/remotes/origin/master)
 > C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
 > C:\Program Files\Git\bin\git.exe checkout -f
42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f
 > C:\Program Files\Git\bin\git.exe rev-list
42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f # timeout=10
[workspace] $ cmd /c call E:\Apps\tomcat7\temp\hudson1928478766077504601.bat

E:\Jenkins\jobs\Helloworld\workspace>javac HelloWorld.java

E:\Jenkins\jobs\Helloworld\workspace>java HelloWorld
Hello World

E:\Jenkins\jobs\Helloworld\workspace>exit 0
Finished: SUCCESS
```

Apart from the steps shown above there are just so many ways to create a build job, the options available are many, which what makes Jenkins such a fantastic continuous deployment tool.

Jenkins, the heart of DevOps, is a continuous integration tool that allows developers to integrate code into a shared repository at regular intervals. The usual practice is to trigger a build as soon as the code is committed to the repository.

If you're worried about Jenkins replacing Hudson, we can assure you that Hudson is still available as a managed project by Oracle. In fact, Jenkins was forked from the Hudson source code and later began to be supported by the open community.

The common process of Jenkins involves the following 3 steps:

- Developers make changes to the code and commit it to their repository
- Jenkins picks up the code and later runs builds and any tests that may be required
- As a post step, notifications can be triggered, logs can be checked, or any other cleaning job can take place

Any event mapped above, is taken care of by a Jenkins job. A Jenkins job is simply a process that runs on Jenkins server to provide the above mentioned functionality.

If you're new to Jenkins, we're going to help you understand the basics of the Jenkins pipeline, along with the functionality aspect of Jenkins.

What is Jenkins Pipeline?

Jenkins pipeline allows us to define a complete list of events that happen in the code lifecycle. Starting from the build, to testing and deployment.

We can use a set of plugins that help in the implementation of certain processes as a continuous delivery pipeline. Where pipelines are defined using code by using groovy language to define the processes that would run in the pipeline.

To implement pipeline as code, a Jenkins file needs to be present at the project's root repository.

Jenkins file supports two different syntax:

- Declarative
- Scripted

Declarative pipeline was a recent release and provides better syntax support over the scripted pipelines.

Why should you use Jenkins pipeline?

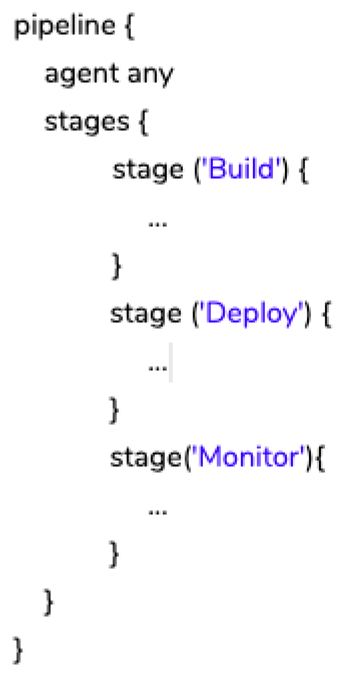
- Pipeline can be run in a loop.
- It supports larger projects that may involve a high CPU job, provided the Jenkins infrastructure is scalable enough to support it.
- Since Jenkins pipeline is written in code, any number of users can use it as a template, modify it and run customized tests and processes.

- Multiple jobs can be run in parallel.
- Jenkins Pipeline is robust. Pipeline can automatically be resumed from it might have stopped for any reason.

Understanding the pipeline

- PipelineA user-defined block, which contains all the processes such as build, test, deploy, etc. All the stages and steps are defined in this block.
- NodeThe node is a machine on which Jenkins runs. A node block is used in scripted pipeline syntax.
- StageThis block contains a series of steps in a pipeline. i.e., build, test, deploy processes on a stage.

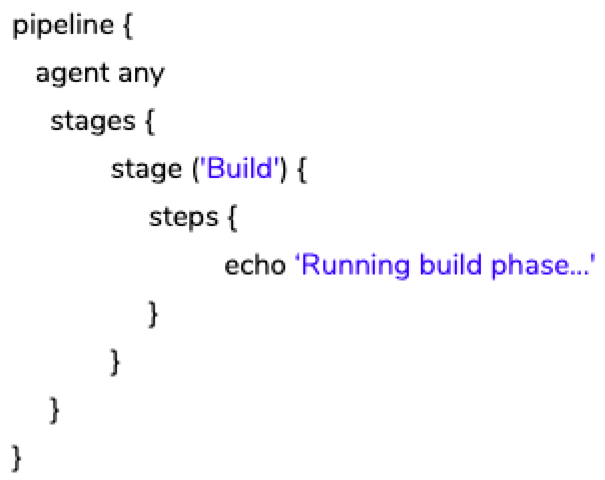
Let us see an example for multiple stages, where each stage performs a specific task:



```
pipeline {
   agent any
   stages {
          stage ('Build') {
              ...
          }
          stage ('Deploy') {
              ...
          }
          stage('Monitor'){
              ...
          }
   }
}
```

Multiple Stages in Jenkins pipeline

- StepA step is a single task that executes a particular process at a defined time. A pipeline involves steps defined within a stage block.



```
pipeline {
   agent any
     stages {
             stage ('Build') {
                  steps {
                            echo 'Running build phase...'
                  }
             }
     }
}
```

Single step in Jenkins pipeline

Defining a Jenkins pipeline

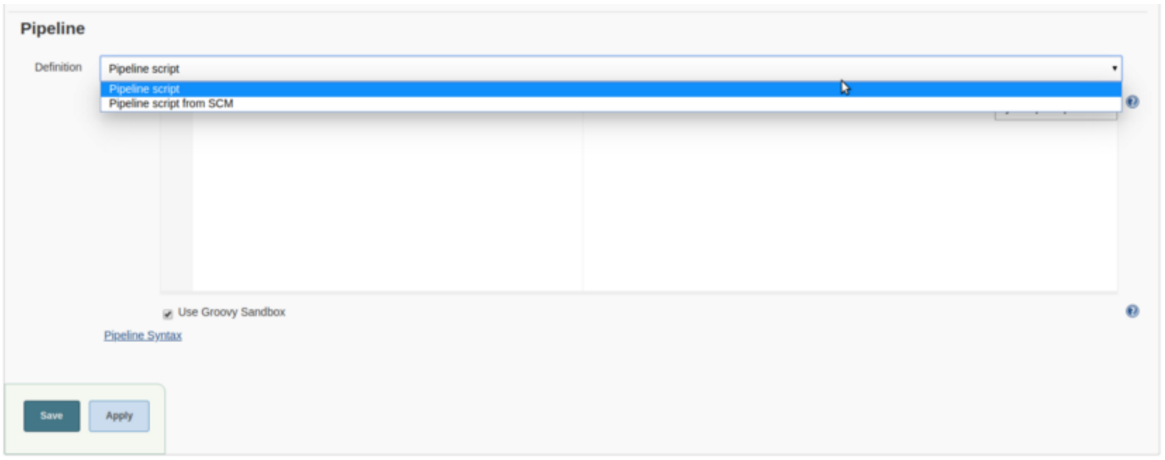
The following steps run smoothly once you have the initial configuration and installations complete for Jenkins.

For Jenkins Pipeline, install the Pipeline plugin
Go to Manage Jenkins > Manage Plugins > Available section > Search Pipeline.

If you are already installed Pipeline It will display in the Installed section.

Create a new project with the Pipeline.

Go to Jenkins Home > New Item > Create Project with Pipeline
After creating Pipeline Project the following window should appear:



Create project with Jenkins Pipeline

This is where we write pipeline code.

The two options seen in the image are as follows:

- Pipeline ScriptWrite the Pipeline directly on Jenkins.
- Pipeline Script from SCMWrite Jenkinsfile and upload it on any SCM and then use it from that repository.

Jenkins Scripted Pipeline

Jenkins pipelines are traditionally written as scripted pipelines. Ideally, the scripted pipeline is stored in the Jenkins webUI as a Jenkins file. The end-to-end scripted pipeline script is written in Groovy.

- ✓ It requires knowledge of Groovy programming as a prerequisite.
- ✓ The Jenkinsfile starts with the word **node**.
- ✓ Can contain standard programming constructs like if-else block, try-catch block, and so on.

Sample Scripted Pipeline

```
node
   stage('Stage 1')
      echo 'hello'
```

What Is Jenkins Declarative Pipeline?

The Declarative Pipeline subsystem in Jenkins Pipeline is relatively new, and provides a simplified, opinionated syntax on top of the Pipeline subsystems.

- The latest addition in Jenkins pipeline job creation technique.
- Jenkins declarative pipeline needs to use the predefined constructs to create pipelines. Hence, it is not flexible as a scripted pipeline.
- A Jenkinsfile starts with the word **pipeline**.
  Jenkins declarative pipelines should be the preferred way to create a Jenkins job, as they offer a rich set of features, come with less learning curve, and there are no prerequisites to learn a programming language like Groovy just for the sake of writing pipeline code.

  We can also validate the syntax of the declarative pipeline code before running the job. It helps to avoid a lot of runtime issues with the build script.

  Our First Declarative Pipeline

```
pipeline
    agent any
    stages
        stage('Welcome Step')
            steps
                echo 'Welcome to LambdaTest'
```

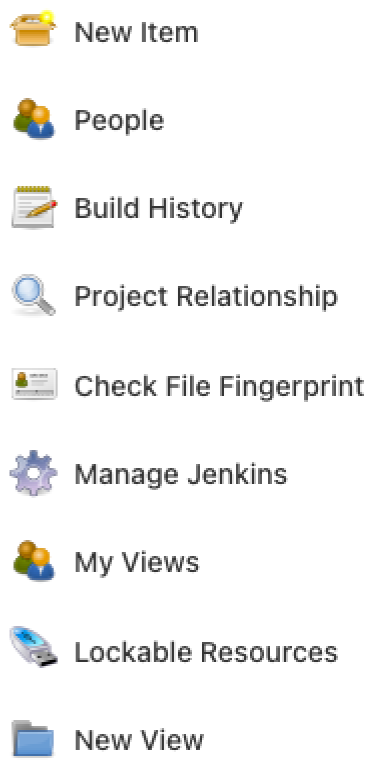
We recommend VS Code IDE for writing Jenkins pipeline scripts, especially when creating Jenkins Declarative pipeline examples.

Running Your First Declarative Pipeline

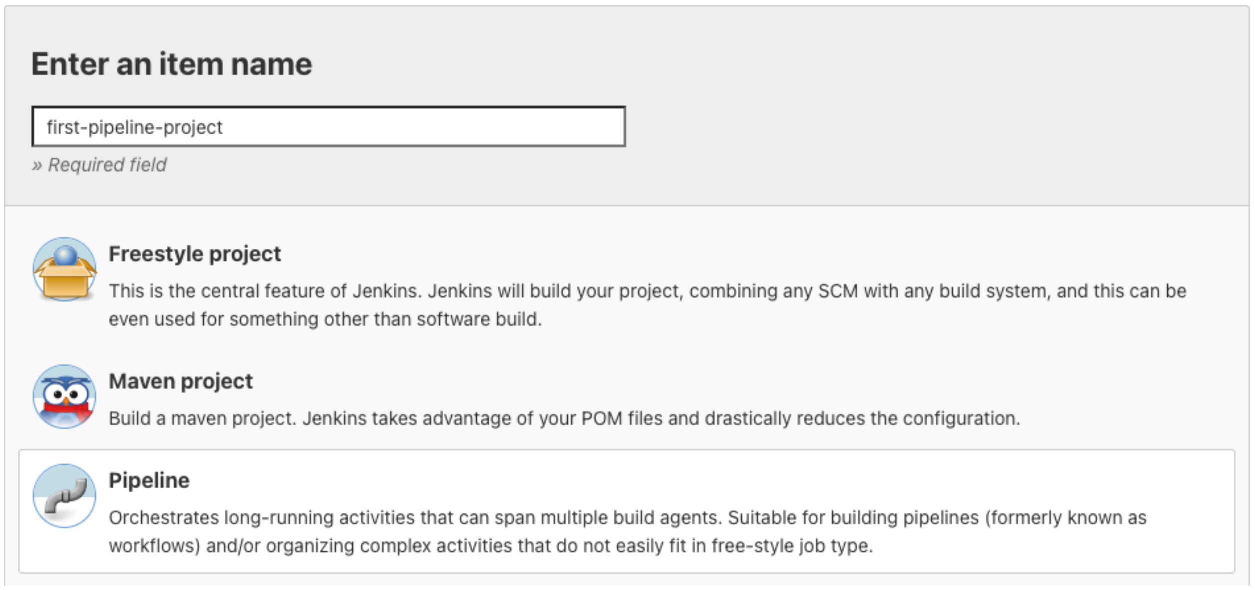
Now that you are well-acquainted with the Jenkins pipeline's basics, it's time to dive deeper. In this section, we will learn how to run a Jenkins declarative pipeline.

Let us run our Jenkins declarative pipeline step by step.

Step 1: Open Jenkins home page (http://localhost:8080 in local) & click on **New Item** from the left side menu.



Step 2: Enter **Jenkins job name,** choose the style as **Pipeline**, and click **OK**.


Enter an item name

first-pipeline-project

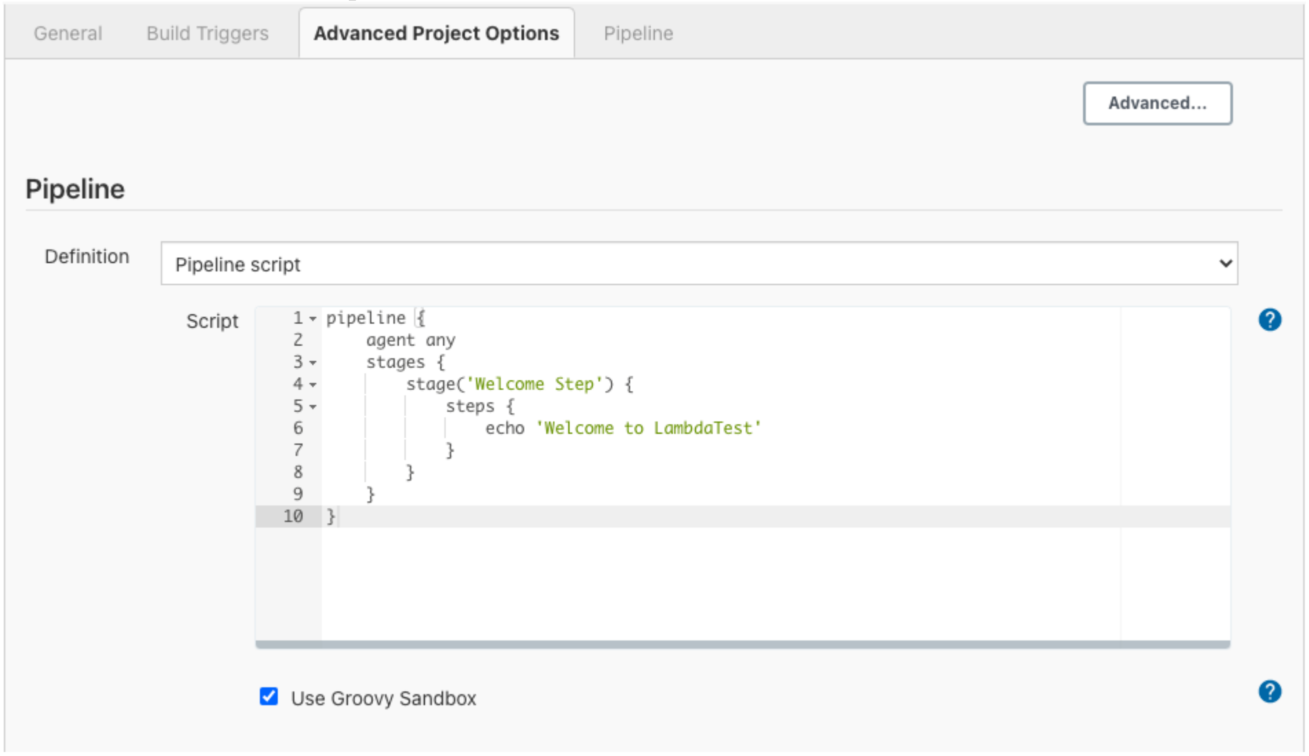
» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

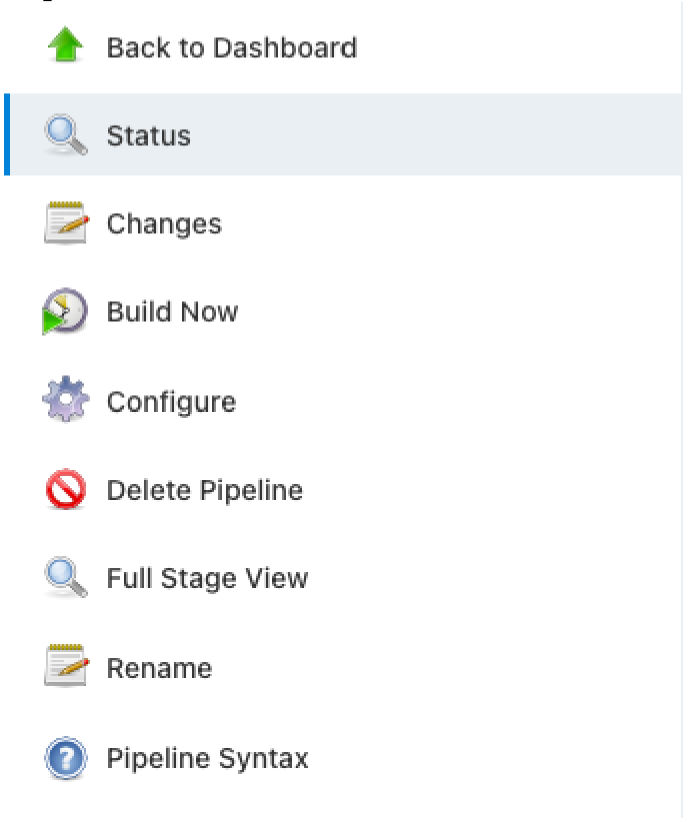
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Step 3: Scroll down to the **Pipeline** section and copy-paste your first Declarative style Pipeline code from below to the script textbox.

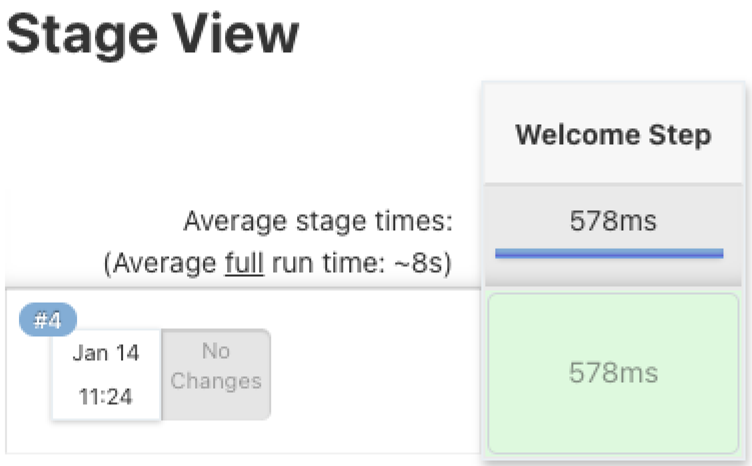


```
pipeline {
    agent any
    stages {
        stage('Welcome Step') {
            steps {
                echo 'Welcome to LambdaTest'
            }
        }
    }
}
```

Step 4: Click on the Save button and click on **Build Now** from the left side menu.



We can see the build running stage by stage in **Stage** View.



Step 5: To check logs from Build Job, click on any stage and click on the **check logs** button. Or you can use the **Console Output** from the left side menu to see the logs for the build.