Jenkins

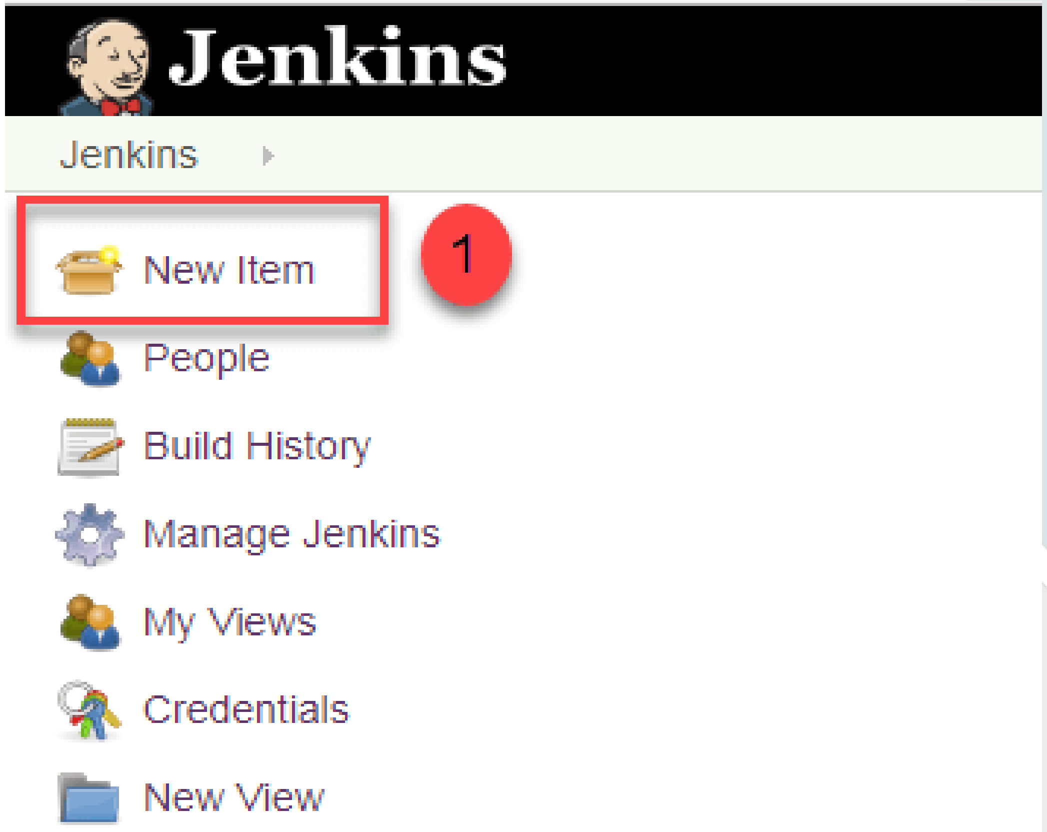
Jeeva S. Chelladhurai
CEO, Comorin Consulting Services
+91 97319 77222
jeeva@comorin.co

3. Jenkins Job Configuration

- Configuring Free-Style Job
- Pipeline Job
  - Continuous Delivery Pipeline
  - Jenkinsfile
  - Declarative vs Scripted Pipeline
  - Reasons to use Jenkins Pipeline
  - Steps to create Pipeline Job
  - Pipeline Job through GitHub Repo
- Multibranch Pipeline
- Folder Job
- GitHub Organization Job
- Multi-Configuration Job

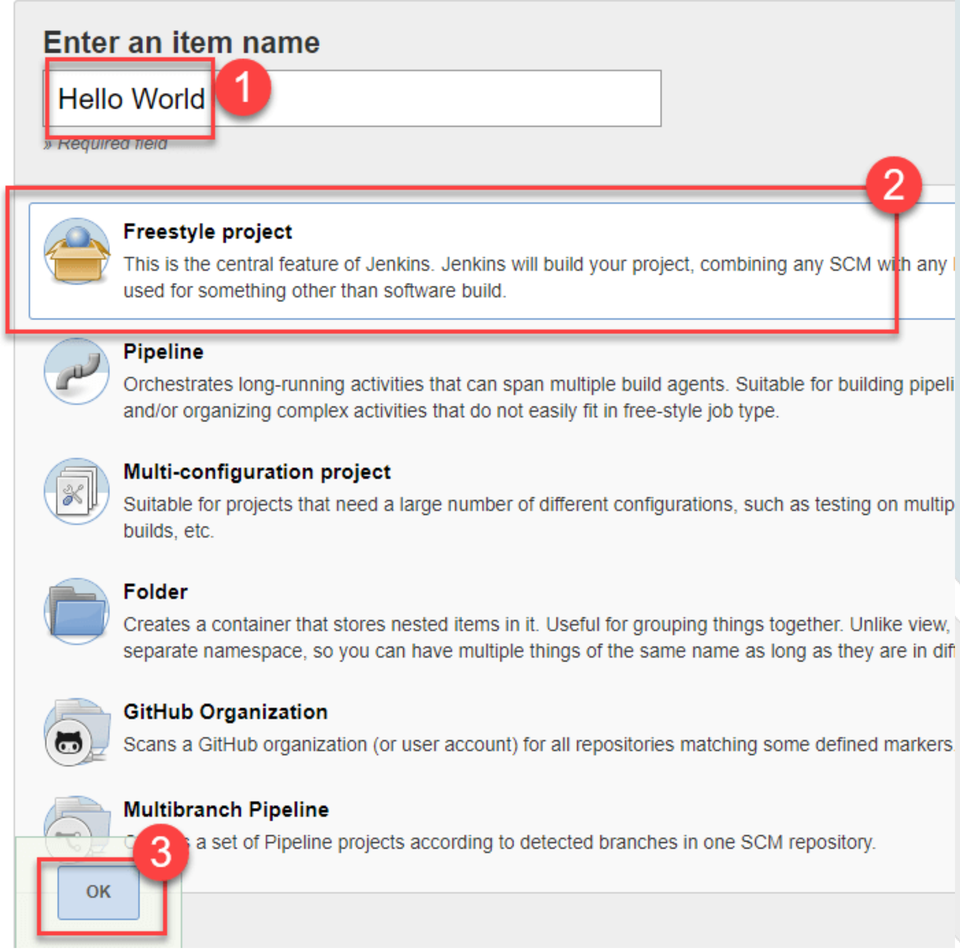
Configuring Free-Style Job

- **Step 1:**
  - Login to Jenkins
- **Step 2:**
  - Click on “New Item” at the top left-hand side of the dashboard



Configuring Free-Style Job Step 3

- **Step 3:**
  - Enter the name of the item that need to be created
  - Select free style job
  - Click okay



Enter an item name

Hello World

» Required field

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

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipeli
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 multip
builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view,
separate namespace, so you can have multiple things of the same name as long as they are in dif

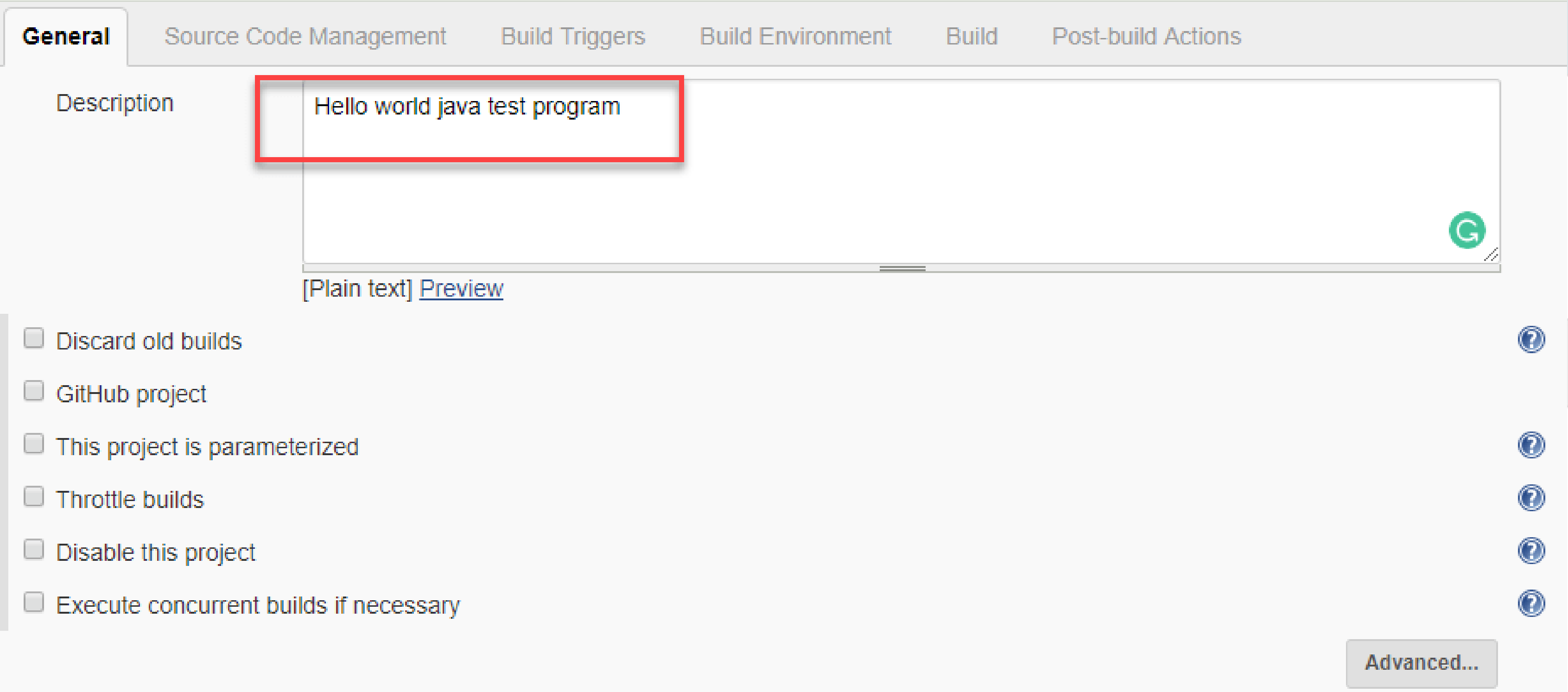
GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers

Multibranch Pipeline
a set of Pipeline projects according to detected branches in one SCM repository.

OK

Configuring Free-Style Job Step 4

- **Step 4:**
  - Enter the details of the project in description field



Description: Hello world java test program

□ Discard old builds

□ GitHub project

□ This project is parameterized

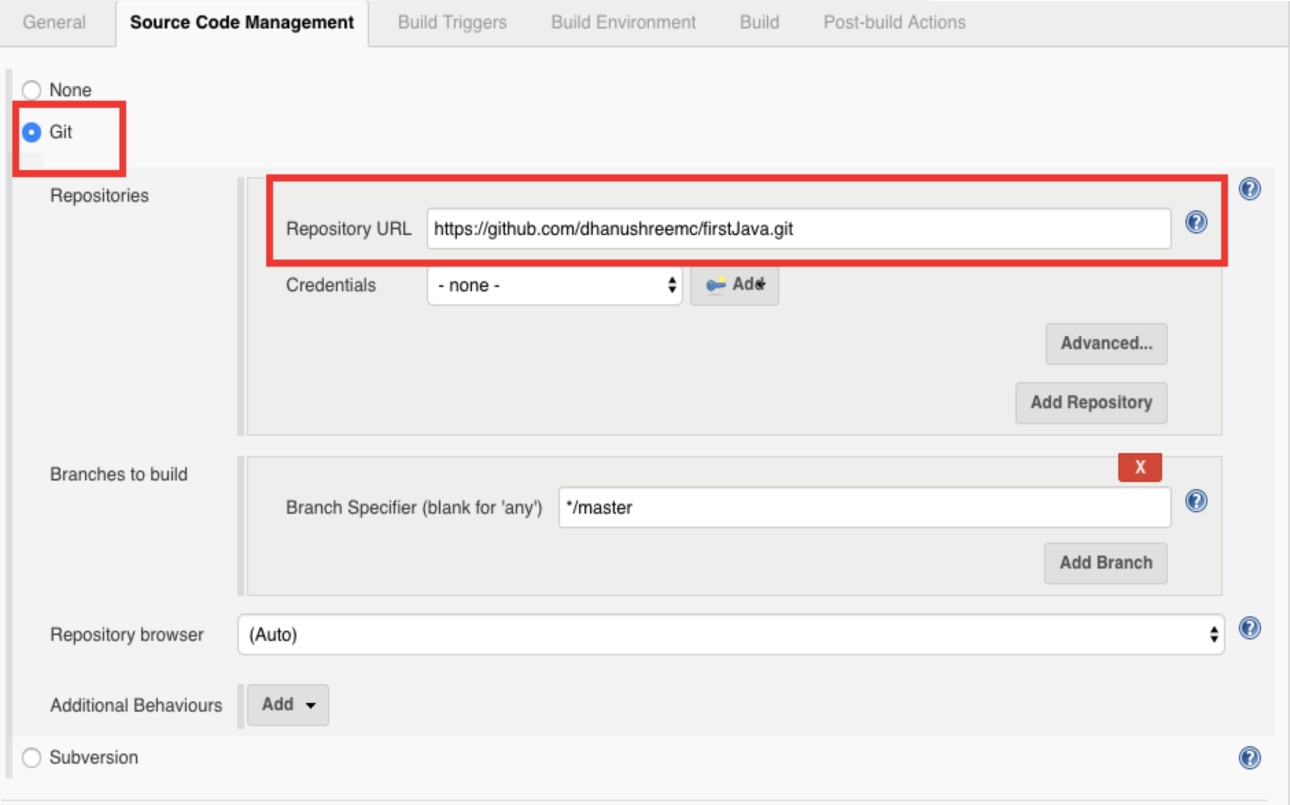
□ Throttle builds

□ Disable this project

□ Execute concurrent builds if necessary

Configuring Free-Style Job Step 5

- **Step 5:**
  - Under the source code management section enter the repository url. Here test repository located at https://github.com/dhanushreemc/firstJava.git



○ None
● Git

Repositories
Repository URL: https://github.com/dhanushreemc/firstJava.git
Credentials: - none -

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

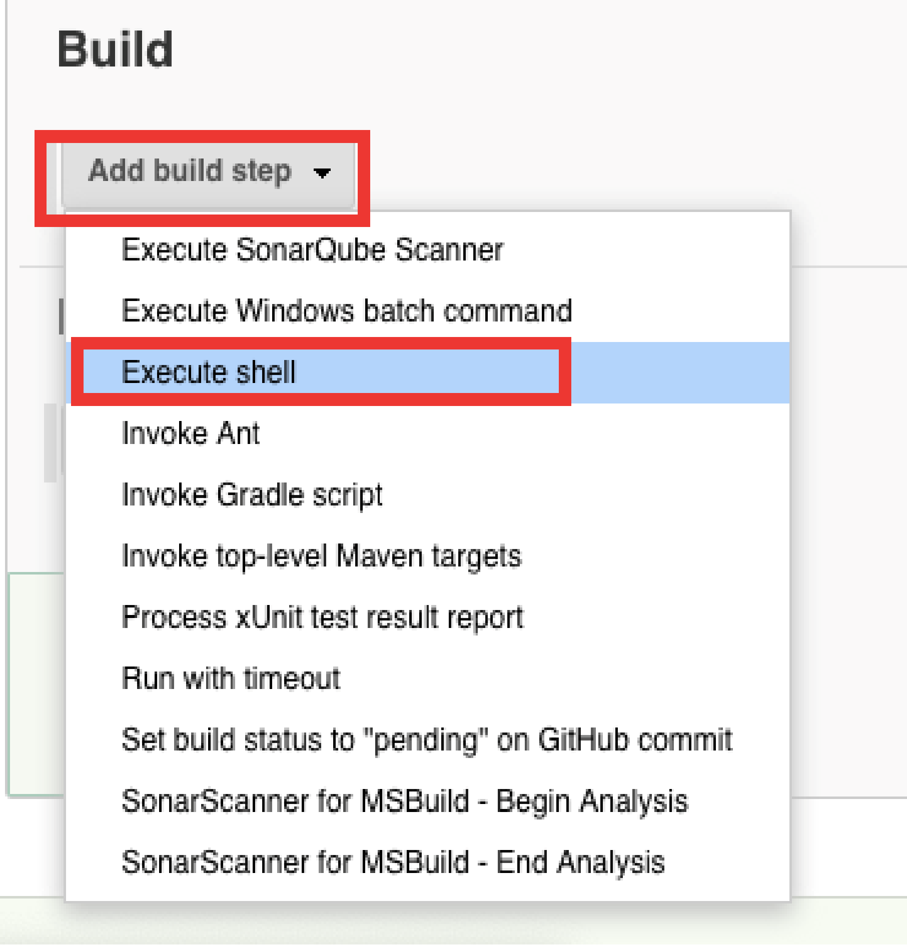
Repository browser: (Auto)

Additional Behaviours

○ Subversion

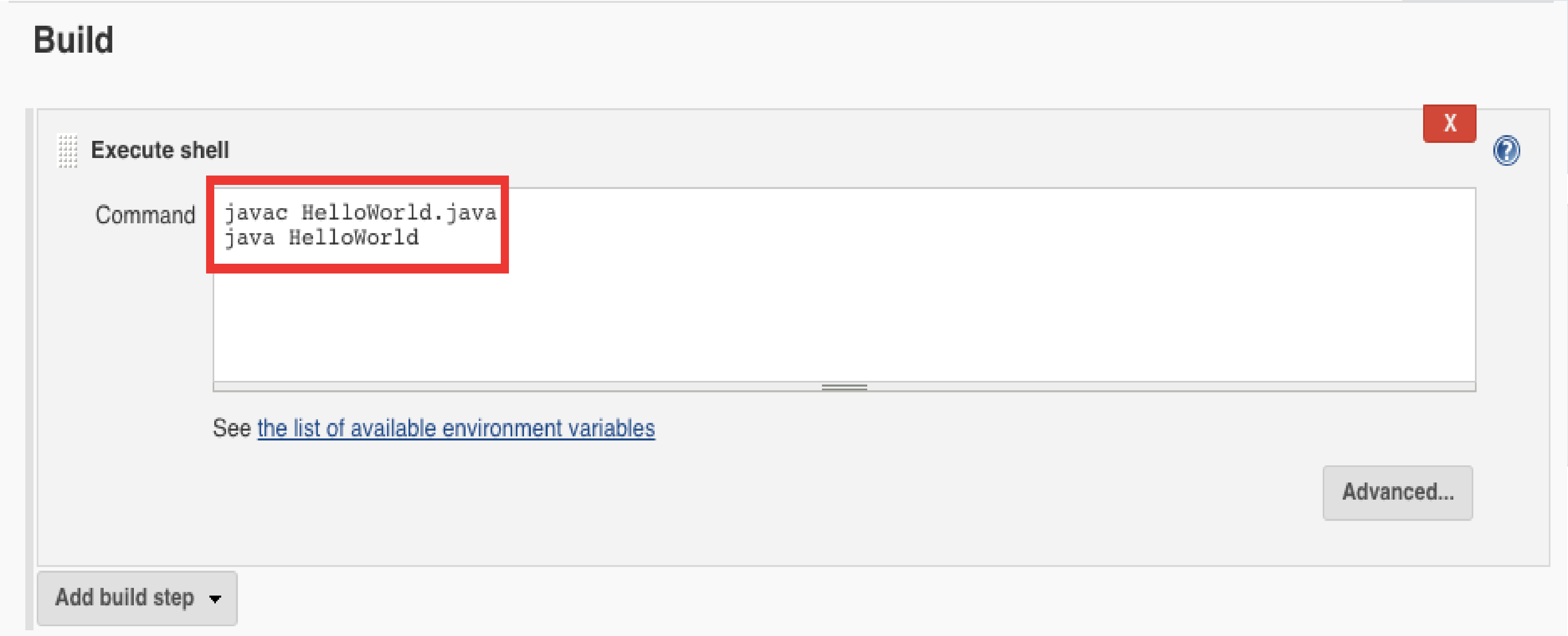
Configuring Free-Style Job Step 6

- **Step 6:**
  - Now click on “build” section to build the code at the time you want
  - The build can also be scheduled to run periodically
  - Click on “Add Build step”
  - Click & Select “execute shell”


Build
Add build step
Execute SonarQube Scanner
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Process xUnit test result report
Run with timeout
Set build status to "pending" on GitHub commit
SonarScanner for MSBuild - Begin Analysis
SonarScanner for MSBuild - End Analysis

Configuring Free-Style Job Step 7

- **Step 7:**
  - Add the commands to execute as like shown below



Build

Execute shell

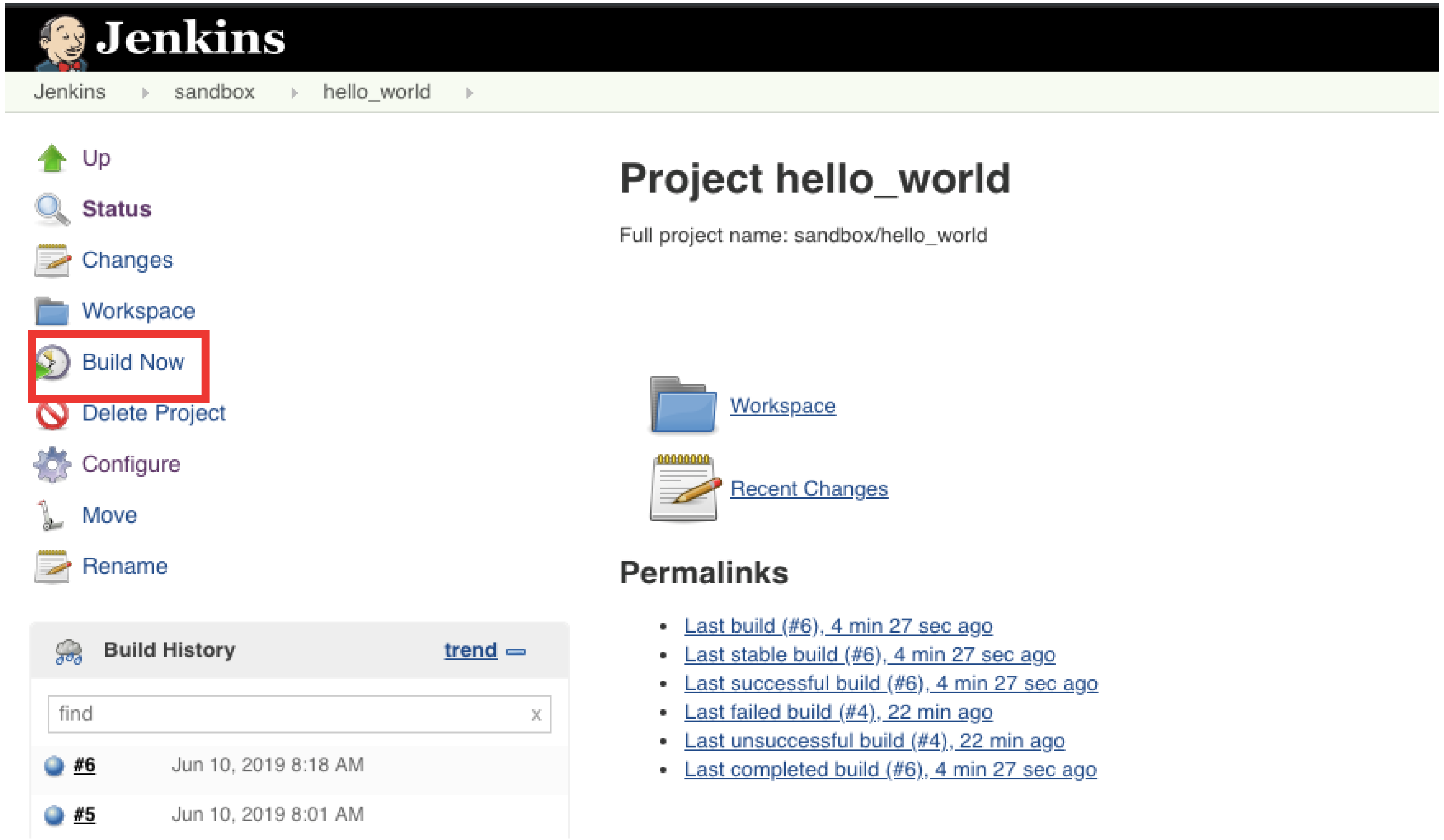
Command

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

See the list of available environment variables

Configuring Free-Style Job Step 8

- **Step 8:**
  - Now in the main screen, click “Build now” button as shown below



Jenkins

Jenkins › sandbox › hello_world ›

- Up
- Status
- Changes
- Workspace
- Build Now
- Delete Project
- Configure
- Move
- Rename

Build History trend —

find x

#6	Jun 10, 2019 8:18 AM
#5	Jun 10, 2019 8:01 AM

Project hello_world

Full project name: sandbox/hello_world

Workspace

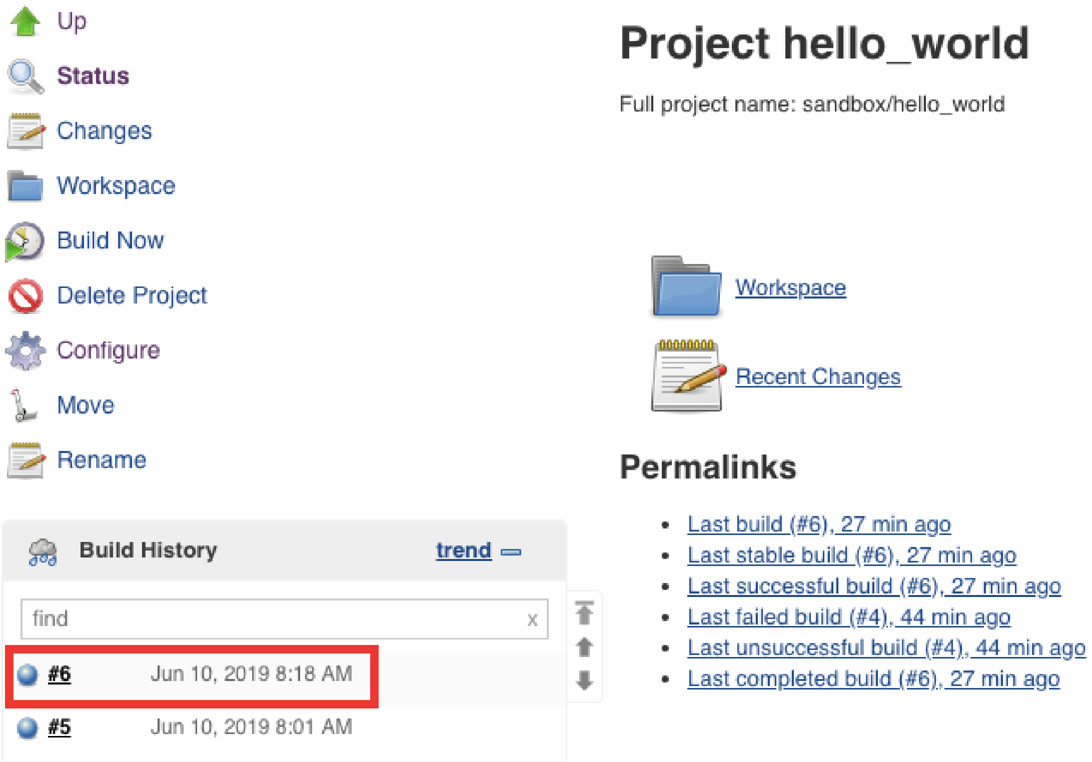
Recent Changes

Permalinks

- Last build (#6), 4 min 27 sec ago
- Last stable build (#6), 4 min 27 sec ago
- Last successful build (#6), 4 min 27 sec ago
- Last failed build (#4), 22 min ago
- Last unsuccessful build (#4), 22 min ago
- Last completed build (#6), 4 min 27 sec ago

Configuring Free-Style Job Step 9

- **Step 9:**
  - Click on “build number” and then click on “console output” and see the status of build which ran



Project hello_world

Full project name: sandbox/hello_world

Workspace

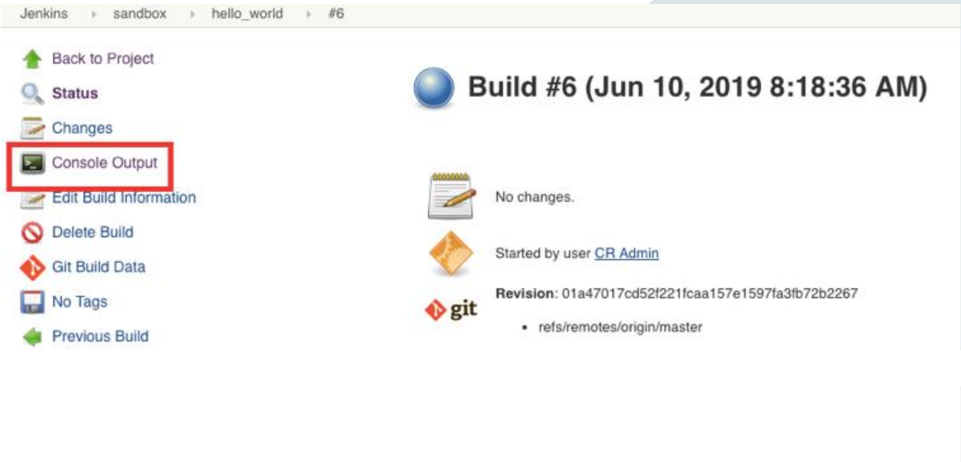
Recent Changes

Build History trend

#6	Jun 10, 2019 8:18 AM
#5	Jun 10, 2019 8:01 AM

Permalinks

- Last build (#6), 27 min ago
- Last stable build (#6), 27 min ago
- Last successful build (#6), 27 min ago
- Last failed build (#4), 44 min ago
- Last unsuccessful build (#4), 44 min ago
- Last completed build (#6), 27 min ago



Build #6 (Jun 10, 2019 8:18:36 AM)

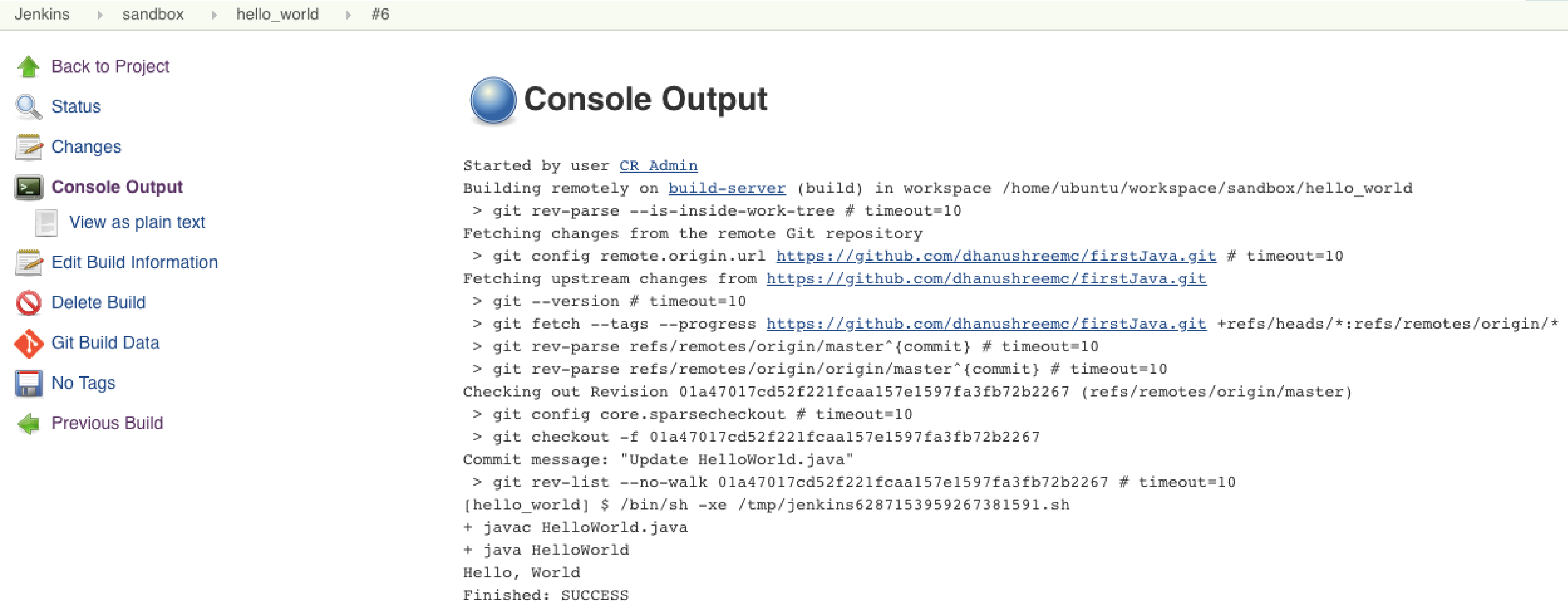
No changes.

Started by user CR Admin

Revision: 01a47017cd52f221fcaa157e1597fa3fb72b2267

- refs/remotes/origin/master

Console Output

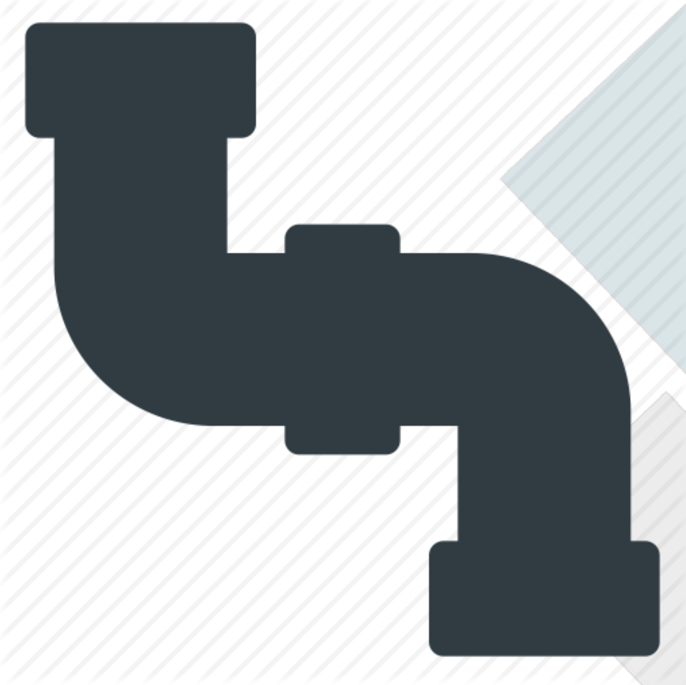


Console Output

```
Started by user CR Admin
Building remotely on build-server (build) in workspace /home/ubuntu/workspace/sandbox/hello_world
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/dhanushreemc/firstJava.git # timeout=10
Fetching upstream changes from https://github.com/dhanushreemc/firstJava.git
 > git --version # timeout=10
 > git fetch --tags --progress https://github.com/dhanushreemc/firstJava.git +refs/heads/*:refs/remotes/origin/*
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 01a47017cd52f221fcaa157e1597fa3fb72b2267 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 01a47017cd52f221fcaa157e1597fa3fb72b2267
Commit message: "Update HelloWorld.java"
 > git rev-list --no-walk 01a47017cd52f221fcaa157e1597fa3fb72b2267 # timeout=10
[hello_world] $ /bin/sh -xe /tmp/jenkins6287153959267381591.sh
+ javac HelloWorld.java
+ java HelloWorld
Hello, World
Finished: SUCCESS
```

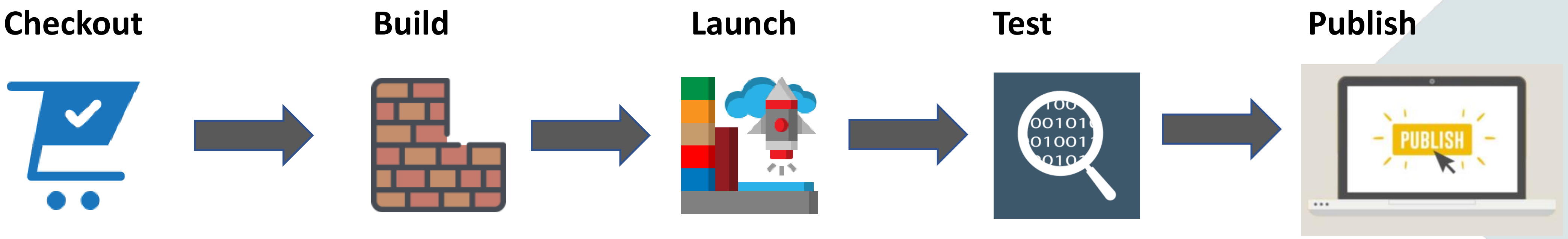
Pipeline Job

- Group of events or jobs interlinked with one another in sequence
- Combination of plugins, supports integration & implementation of continuous delivery pipelines
- Has an extensible automation server for creating simple or complex delivery pipelines
- Extensible automation server for creating simple or complex delivery pipelines “as code” via pipeline DSL (Domain Specific Language)



Continuous Delivery Pipelines

- Every job or event has some sort of dependency on at least one or more events

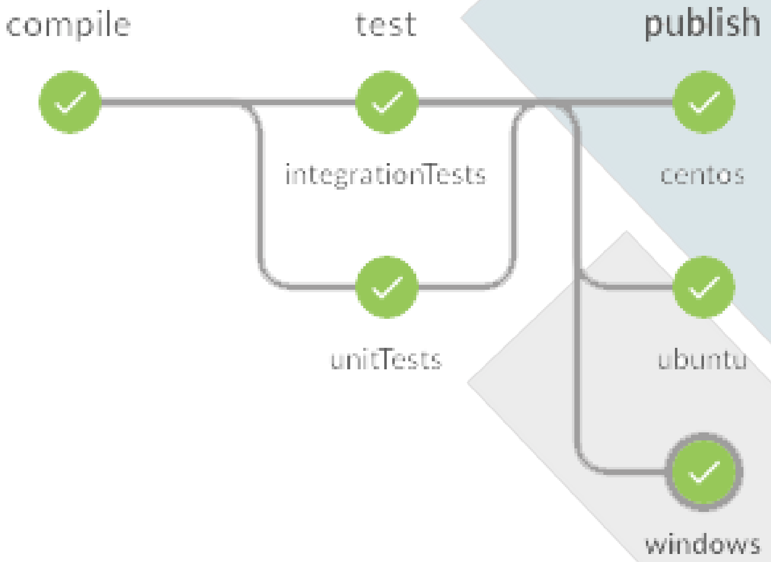


Checkout → Build → Launch → Test → Publish

- Figure represents a continuous delivery pipeline in Jenkins
- Every state has its events, which work in a sequence called a continuous delivery pipeline
- Automated expression to display the process for getting software for version control
- Every changes goes through a no. of complex process on its way to being released
- Involves developing a software in a reliable & repeatable manner
- Progression of build software through multiple stages of testing & deployment

Jenkinsfile

- Jenkins pipeline can be defined using a text file called “***Jenkinsfile***”
- Pipeline as code can be implemented using Jenkinsfile
- Can be defined by using domain specific(DSL)
- With Jenkinsfile the steps needed for running Jenkins pipeline can be written
- **Benefits:**
  - Can create pipelines automatically for all branches and execute pull requests with just one ***Jenkinsfile***
  - Can review code on the pipeline
  - Can audit Jenkins pipeline
  - Singular source for pipeline
  - Can be modified by multiple users



Declarative vs Scripted Pipeline Syntax

Declarative:

- Syntax offers an easy way to create pipelines
- Contains a predefined hierarchy to create Jenkins pipelines
- Gives you the ability to control all aspects of a pipeline execution in a simple, straight-forward manner

Scripted:

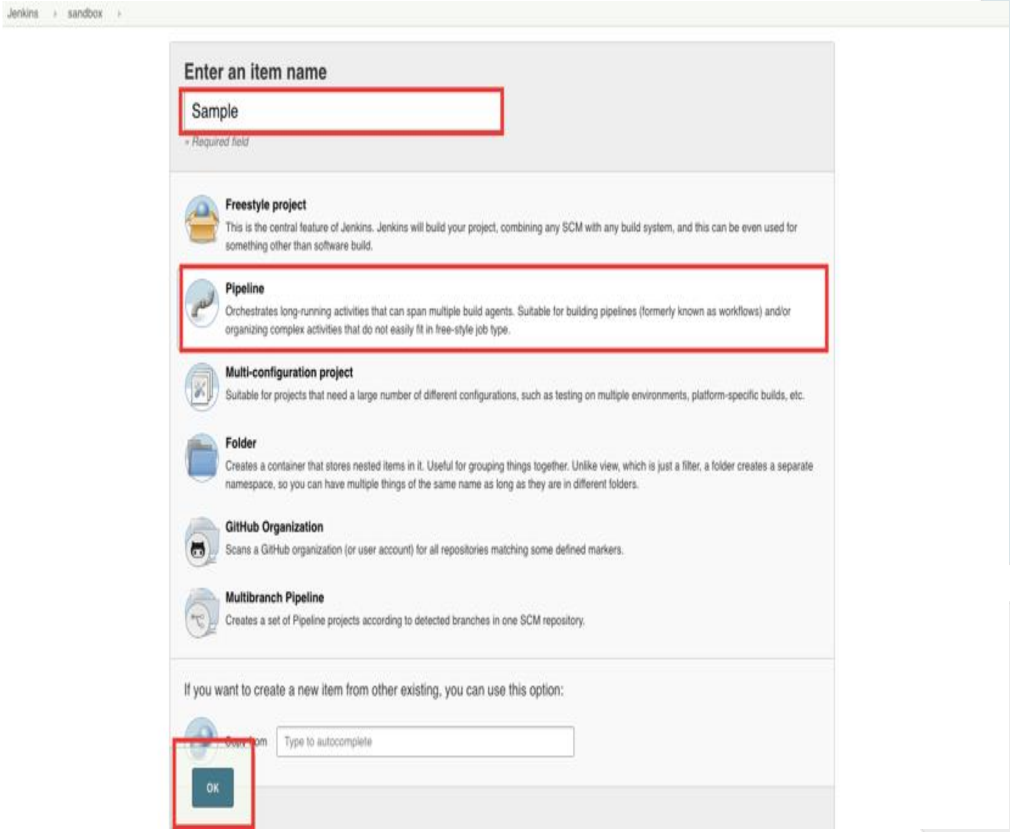
- Runs on the Jenkins master with the help of a lightweight executor
- Uses very few resources to translate the pipeline into atomic commands

Reasons to use Jenkins Pipeline

- Implemented as code, allows multiple users to edit & execute the process
- Pipelines are robust, if server restarts suddenly, will be automatically resumed
- Can pause the pipeline process & make it wait to resume until user input
- Support big projects
- Can run multiple jobs, & even use pipelines in a loop

Steps to create a Pipeline Job

- **Pre-requisites:** Install build-pipeline plugin
- Manage Jenkins->Manage plugins->go to available -> select pipeline and install
- **Step 1:**
  - Go to Jenkins dashboard
  - Click “new item” then select pipeline in job type
  - Enter the name of job
  - Click “Ok”



Jenkins › sandbox ›

Enter an item name

Sample

» 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.

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.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

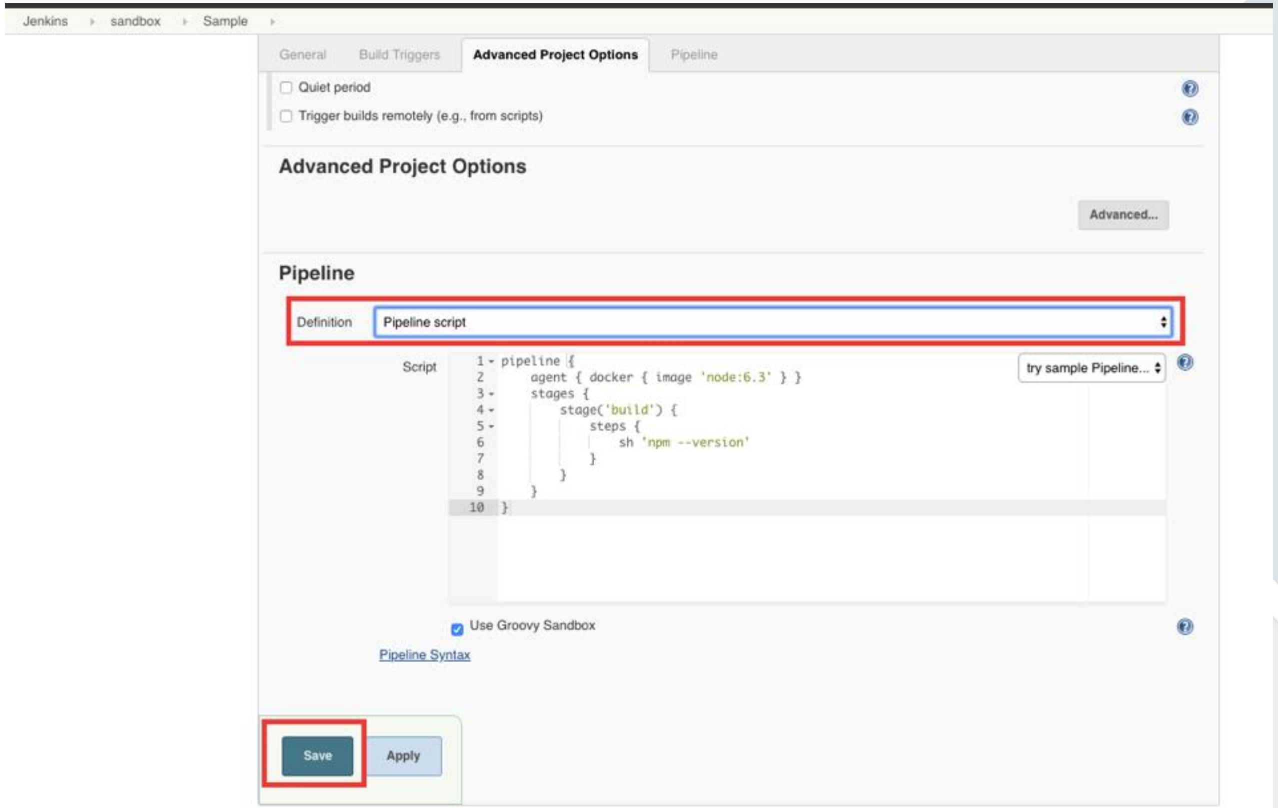
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:

Copy from Type to autocomplete

Steps to create a Pipeline Job Step 2

- **Step 2:**
  - Give description
  - Copy the given **jenkinsfile** code in to pipeline section by selecting “pipeline script” under definition



□ Quiet period

□ Trigger builds remotely (e.g., from scripts)

Advanced Project Options

Pipeline

Definition: Pipeline script

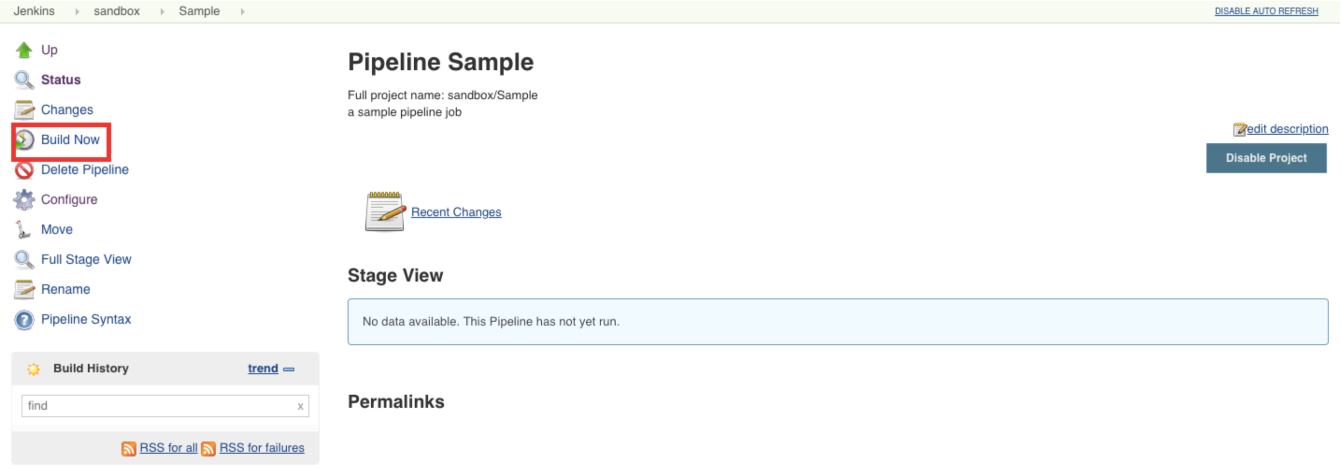
Script:

```
pipeline {
    agent { docker { image 'node:6.3' } }
    stages {
        stage('build') {
            steps {
                sh 'npm --version'
            }
        }
    }
}
```

☑ Use Groovy Sandbox

Steps to create a Pipeline Job Step 3

- **Step 3:**
  - Click on build now and got to console



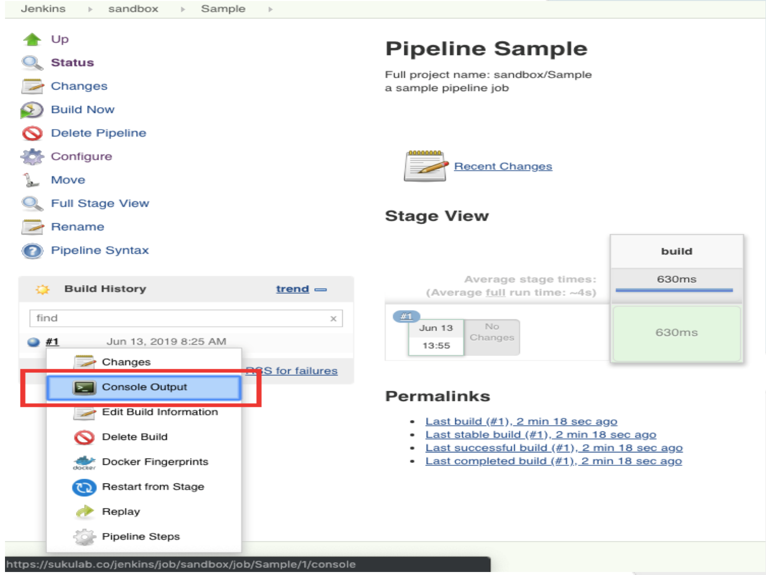
Pipeline Sample

Full project name: sandbox/Sample
a sample pipeline job

Stage View

No data available. This Pipeline has not yet run.

Permalinks



Pipeline Sample

Full project name: sandbox/Sample
a sample pipeline job

#1 Jun 13, 2019 8:25 AM

Stage View

Average stage times:
(Average full run time: ~4s)

build: 630ms

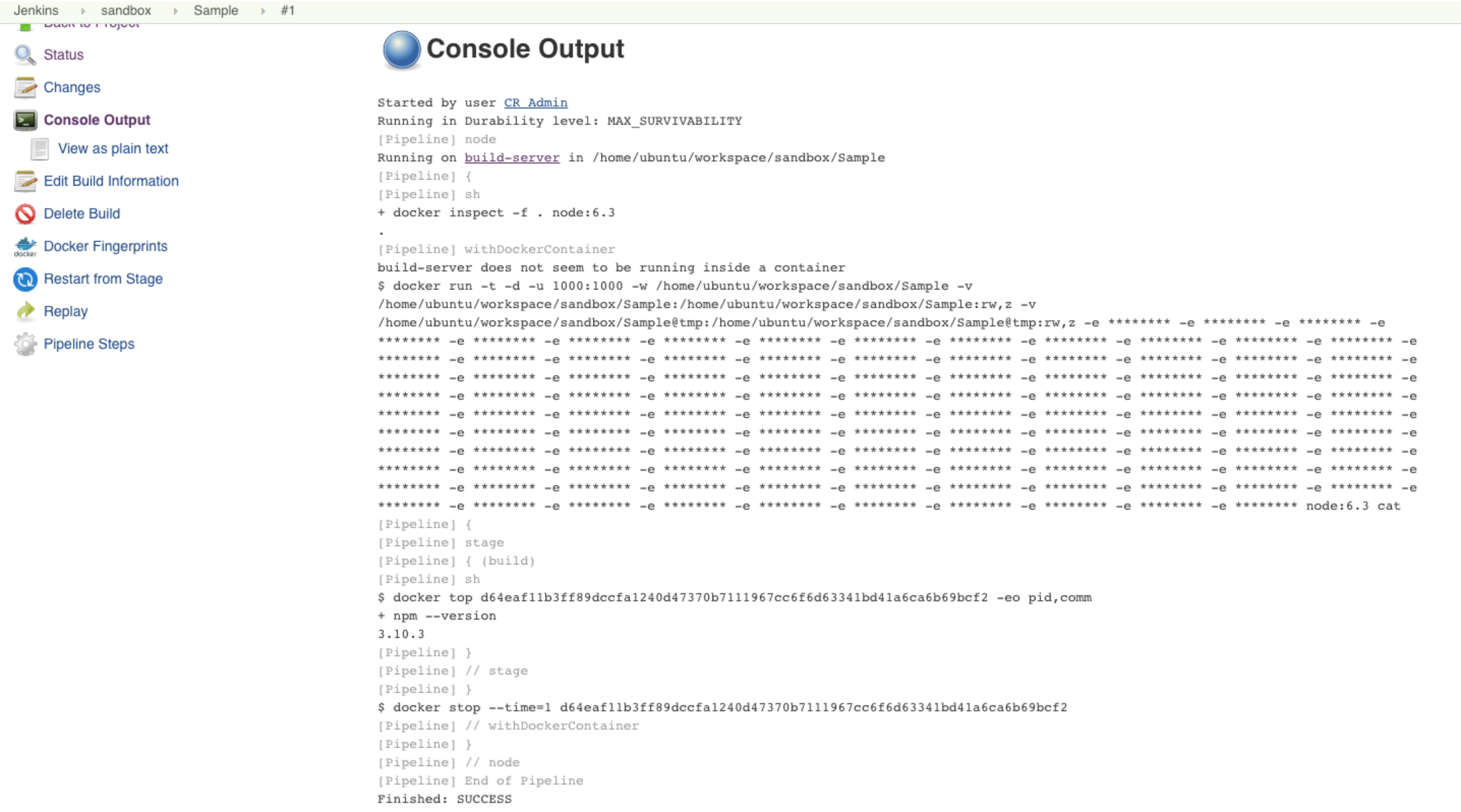
#1 Jun 13 13:55 — 630ms

Permalinks

- Last build (#1), 2 min 18 sec ago
- Last stable build (#1), 2 min 18 sec ago
- Last successful build (#1), 2 min 18 sec ago
- Last completed build (#1), 2 min 18 sec ago

https://sukulab.co/jenkins/job/sandbox/job/Sample/1/console

Console Output

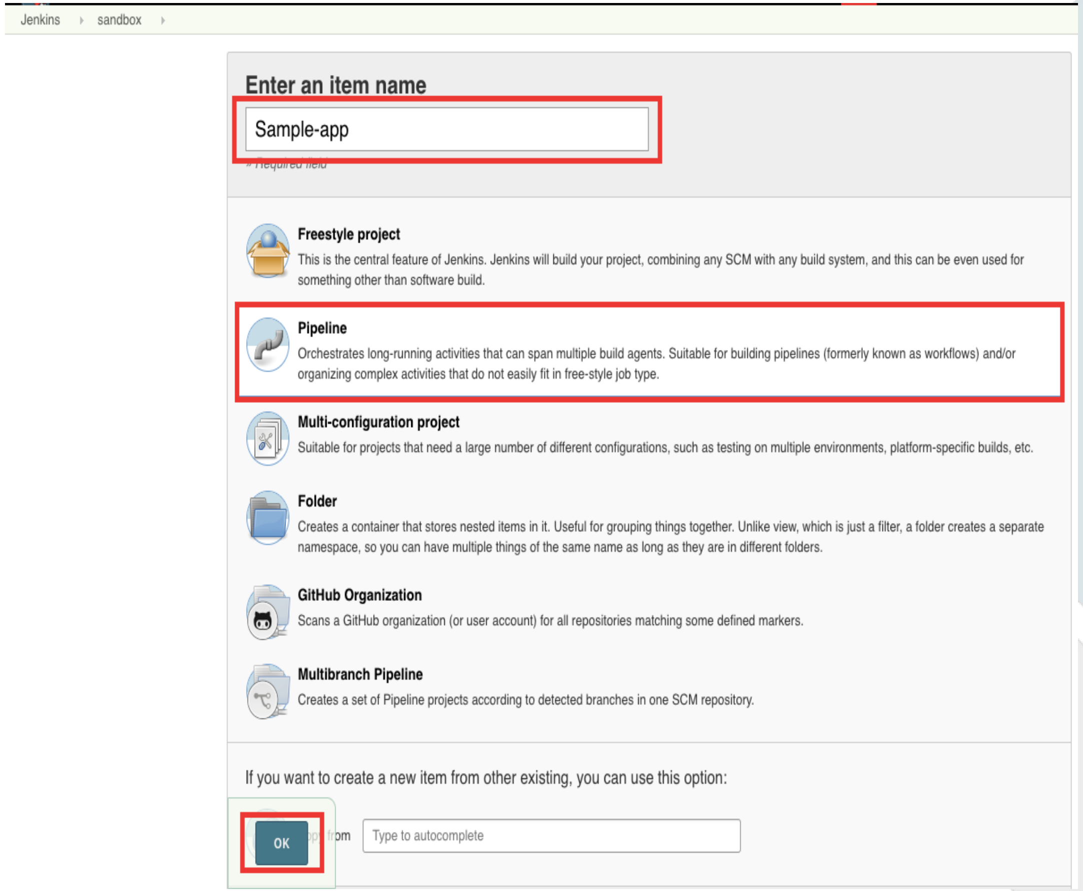


Console Output

```
Started by user CR Admin
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] node
Running on build-server in /home/ubuntu/workspace/sandbox/Sample
[Pipeline] {
[Pipeline] sh
+ docker inspect -f . node:6.3
.
[Pipeline] withDockerContainer
build-server does not seem to be running inside a container
$ docker run -t -d -u 1000:1000 -w /home/ubuntu/workspace/sandbox/Sample -v
/home/ubuntu/workspace/sandbox/Sample:/home/ubuntu/workspace/sandbox/Sample:rw,z -v
/home/ubuntu/workspace/sandbox/Sample@tmp:/home/ubuntu/workspace/sandbox/Sample@tmp:rw,z -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** node:6.3 cat
[Pipeline] {
[Pipeline] stage
[Pipeline] { (build)
[Pipeline] sh
$ docker top d64eaf11b3ff89dccfa1240d47370b7111967cc6f6d63341bd41a6ca6b69bcf2 -eo pid,comm
+ npm --version
3.10.3
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
$ docker stop --time=1 d64eaf11b3ff89dccfa1240d47370b7111967cc6f6d63341bd41a6ca6b69bcf2
[Pipeline] // withDockerContainer
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Pipeline Job through GitHub Repo Step 1

- **Pre-requisites:**
  - GitHub plugin need to be installed
  - Jenkins docker need to be installed on the VM where Jenkins job run
- Configure pipeline job:
- **Step 1:**
  - Go to Jenkins dashboard, click on “new item”
  - Give name for the job
  - Select pipeline as job type
  - Click on Ok



Enter an item name

Sample-app

» 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.

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.

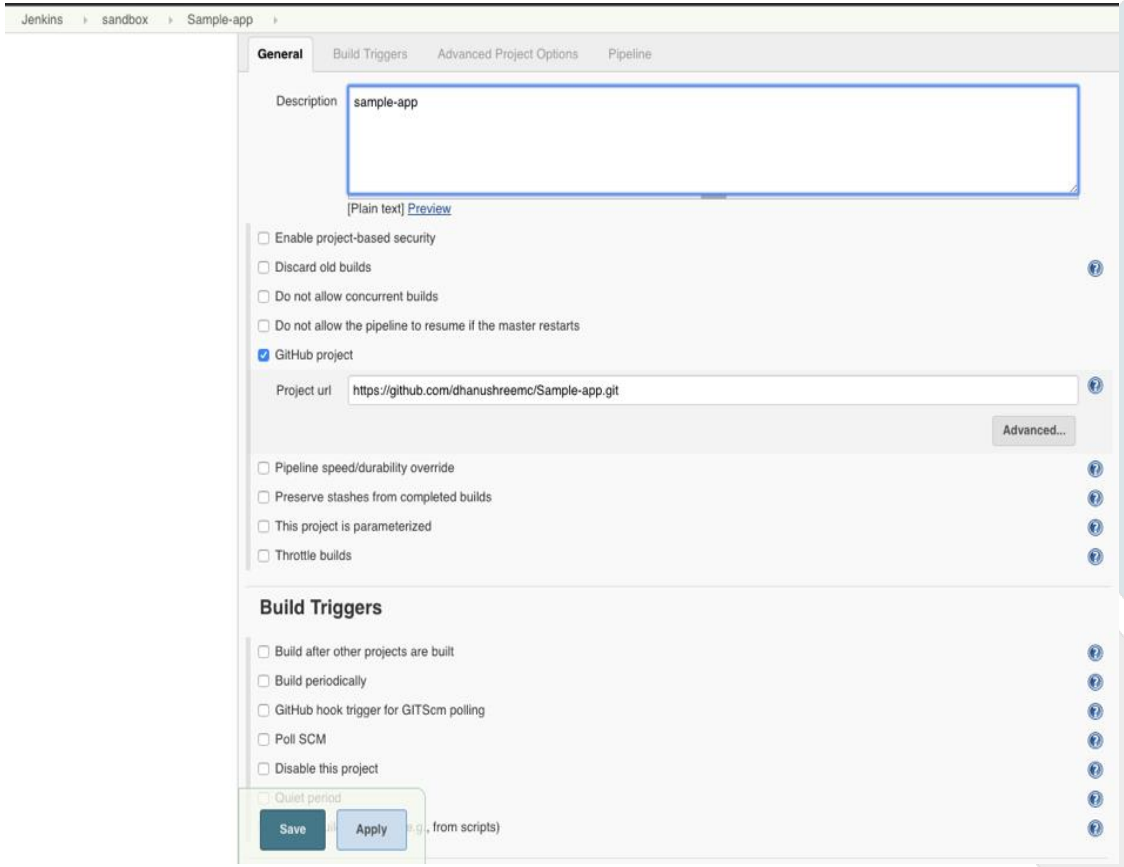
GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:

Pipeline Job through GitHub Repo Step 2

- **Step 2:**
  - Give description for Job,
  - Tick for GitHub project
  - Enter GitHub URL under General section



Description: sample-app

[Plain text] Preview

☐ Enable project-based security
☐ Discard old builds
☐ Do not allow concurrent builds
☐ Do not allow the pipeline to resume if the master restarts
☑ GitHub project

Project url: https://github.com/dhanushreemc/Sample-app.git

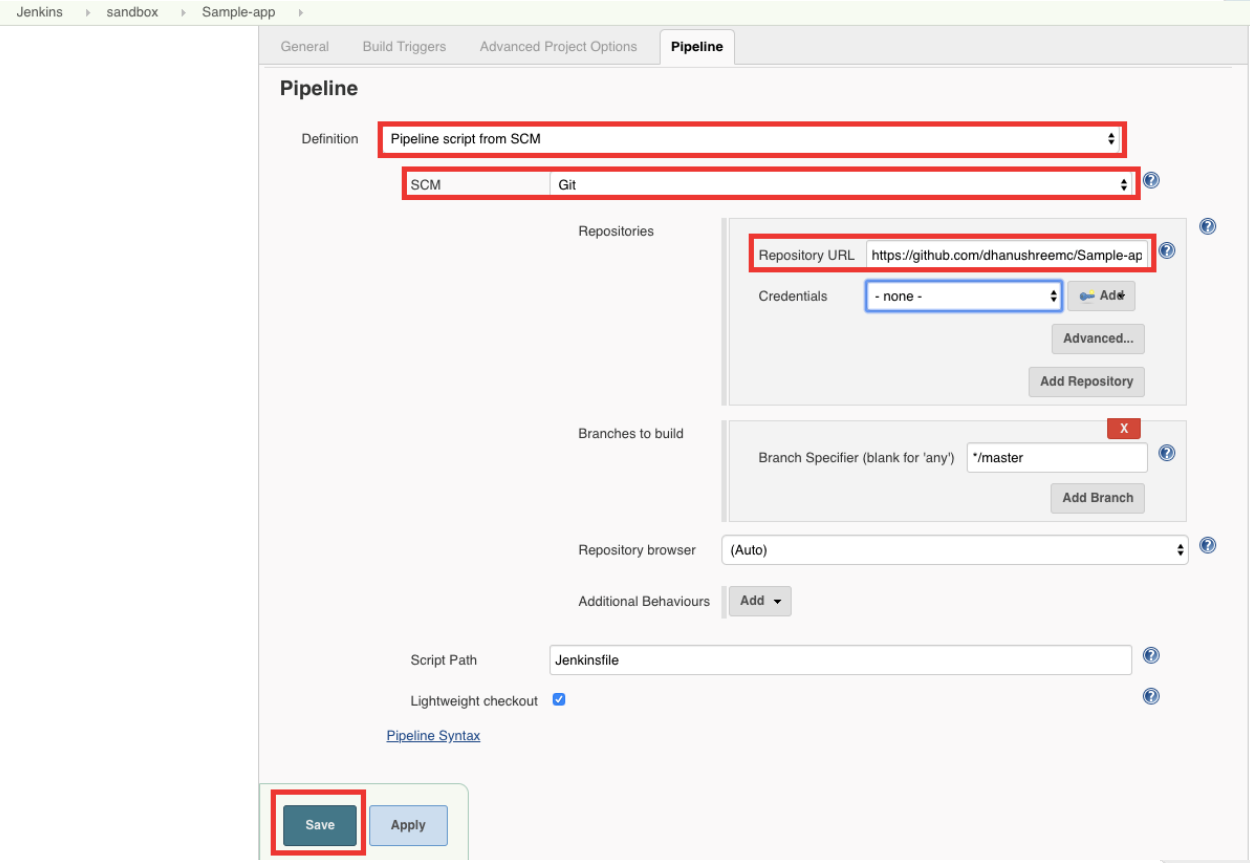
☐ Pipeline speed/durability override
☐ Preserve stashes from completed builds
☐ This project is parameterized
☐ Throttle builds

Build Triggers

☐ Build after other projects are built
☐ Build periodically
☐ GitHub hook trigger for GITScm polling
☐ Poll SCM
☐ Disable this project
☐ Quiet period
e.g., from scripts)

Pipeline Job through GitHub Repo Step 3

- **Step 3:**
  - Go to Pipeline section
  - Select Pipeline script from SCM under definition,
  - Select Git as SCM
  - Enter GitHub repository URL (need to provide credentials if the repo is private)
  - Click on save



Jenkins › sandbox › Sample-app ›

General | Build Triggers | Advanced Project Options | Pipeline

Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/dhanushreemc/Sample-ap

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

Script Path: Jenkinsfile

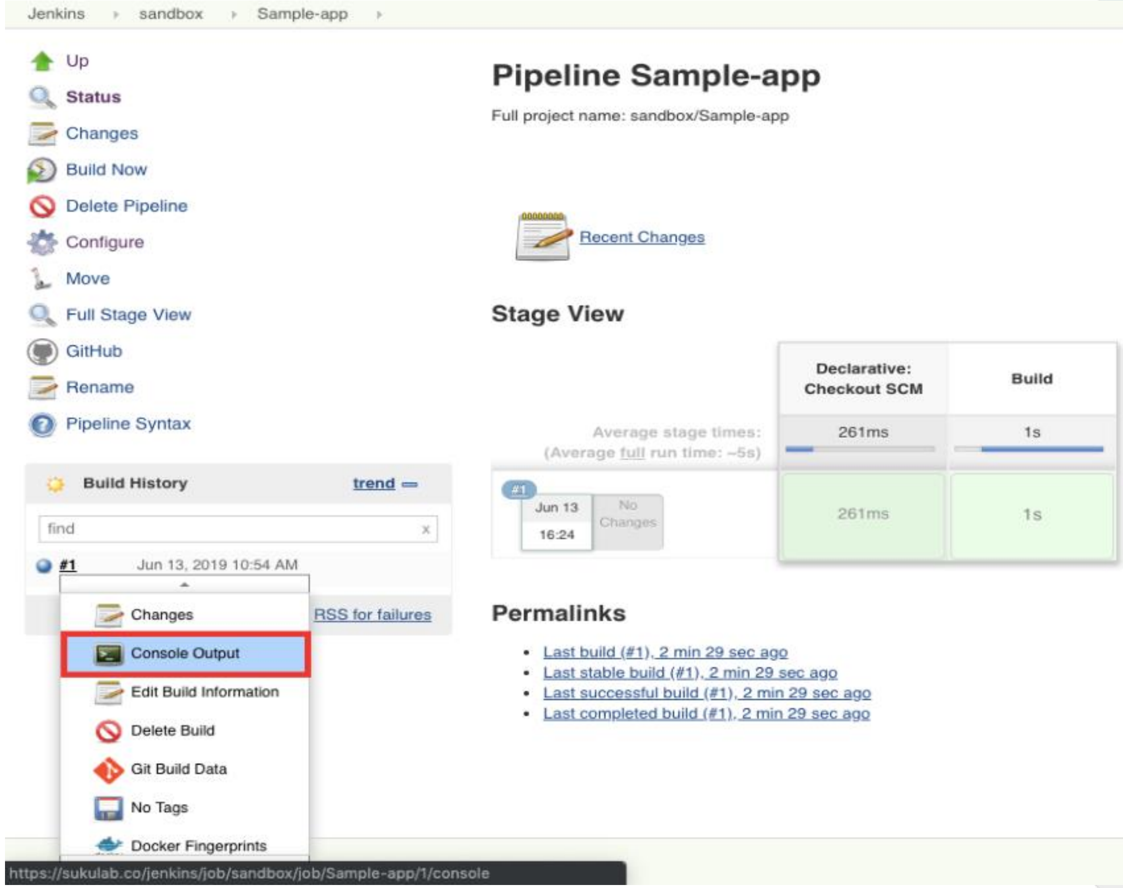
Lightweight checkout ☑

Pipeline Syntax

Pipeline Job through GitHub Repo Step 4

- **Step 4:**
  - Click on “Build now” to build the project
  - Click on build number
  - Select console output to see console log

  Note: Always Jenkinsfile should be in the root of the repository



#1 Jun 13, 2019 10:54 AM

Pipeline Sample-app

Full project name: sandbox/Sample-app

Stage View

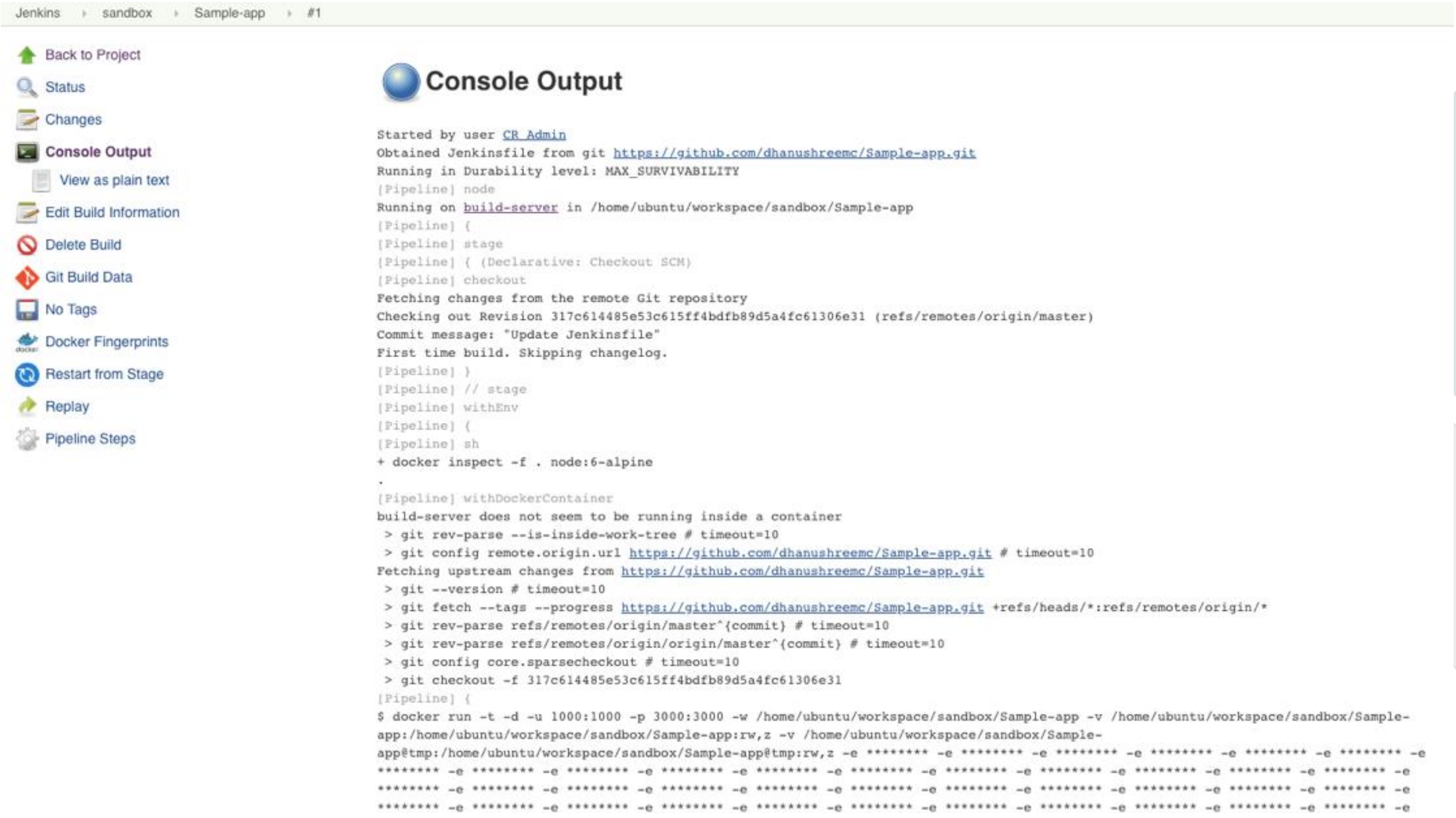
	Declarative: Checkout SCM	Build
Average stage times: (Average full run time: ~5s)	261ms	1s
#1 Jun 13 16:24 No Changes	261ms	1s

Permalinks

- Last build (#1), 2 min 29 sec ago
- Last stable build (#1), 2 min 29 sec ago
- Last successful build (#1), 2 min 29 sec ago
- Last completed build (#1), 2 min 29 sec ago

https://sukulab.co/jenkins/job/sandbox/job/Sample-app/1/console

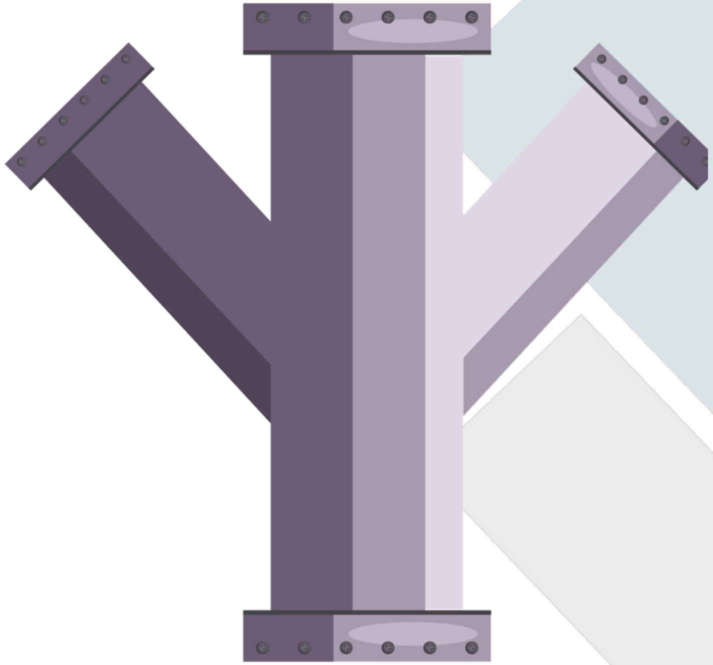
Console Output



```
Started by user CR Admin
Obtained Jenkinsfile from git https://github.com/dhanushreemc/Sample-app.git
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] node
Running on build-server in /home/ubuntu/workspace/sandbox/Sample-app
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Fetching changes from the remote Git repository
Checking out Revision 317c614485e53c615ff4bdfb89d5a4fc61306e31 (refs/remotes/origin/master)
Commit message: "Update Jenkinsfile"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ docker inspect -f . node:6-alpine
.
[Pipeline] withDockerContainer
build-server does not seem to be running inside a container
 > git rev-parse --is-inside-work-tree # timeout=10
 > git config remote.origin.url https://github.com/dhanushreemc/Sample-app.git # timeout=10
Fetching upstream changes from https://github.com/dhanushreemc/Sample-app.git
 > git --version # timeout=10
 > git fetch --tags --progress https://github.com/dhanushreemc/Sample-app.git +refs/heads/*:refs/remotes/origin/*
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 317c614485e53c615ff4bdfb89d5a4fc61306e31
[Pipeline] {
$ docker run -t -d -u 1000:1000 -p 3000:3000 -w /home/ubuntu/workspace/sandbox/Sample-app -v /home/ubuntu/workspace/sandbox/Sample-
app:/home/ubuntu/workspace/sandbox/Sample-app:rw,z -v /home/ubuntu/workspace/sandbox/Sample-
app@tmp:/home/ubuntu/workspace/sandbox/Sample-app@tmp:rw,z -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
```

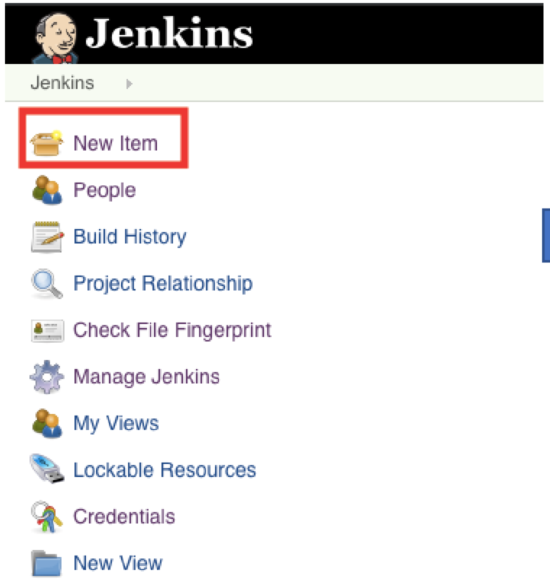
Multibranch Pipeline

- Enables you to implement different Jenkinsfile for different branches of the same project
- Jenkins automatically discovers, manages & executes pipelines for branches which contain a Jenkinsfile in source control
- This eliminates the need for manual Pipeline creation and management



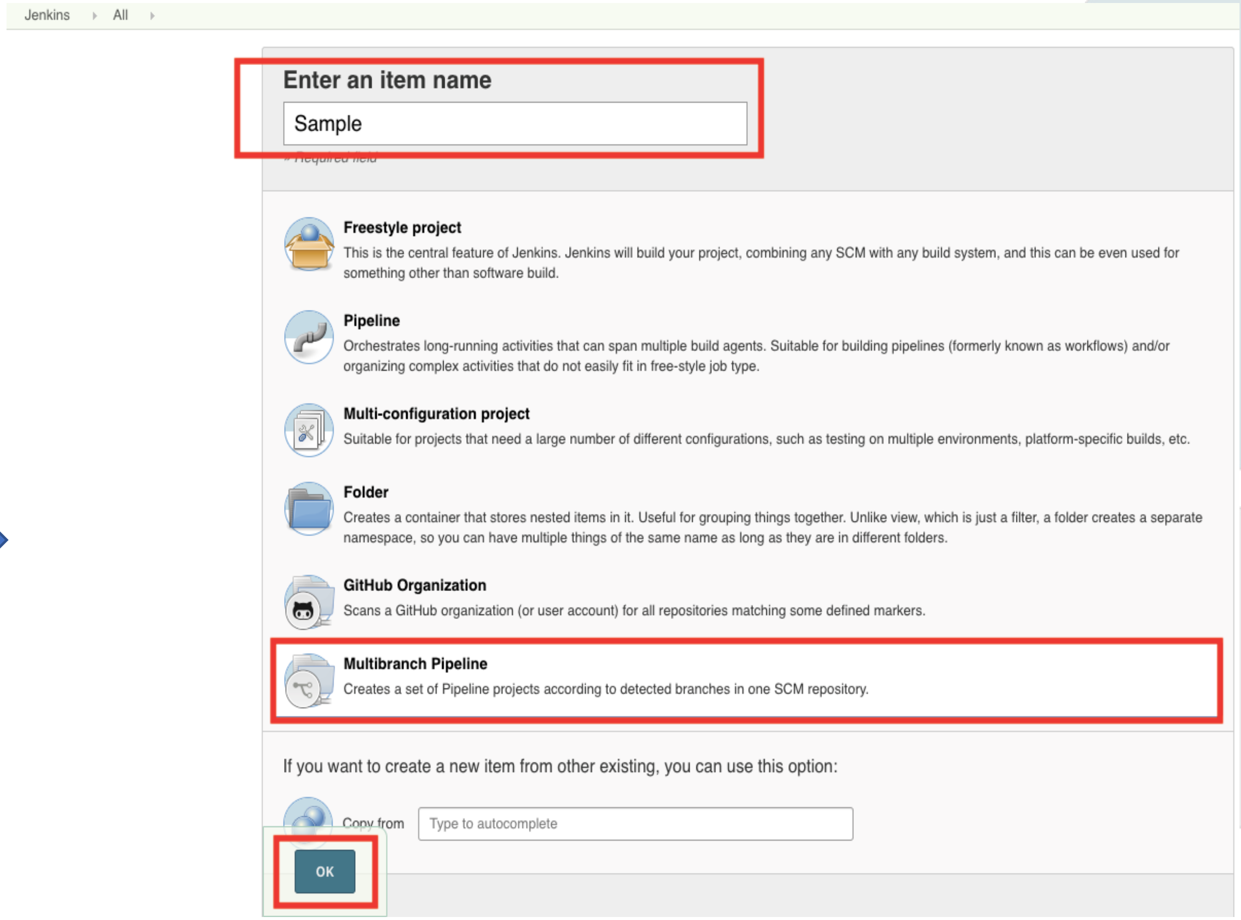
Steps to Create Multi-Branch Pipeline Step 1

- **Step 1:**
  - Go to Jenkins dash board
  - Click on new items
  - Give name for job
  - Select Multibranch Pipeline



Jenkins

Jenkins



Jenkins › All ›

Enter an item name

Sample

» 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.

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.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

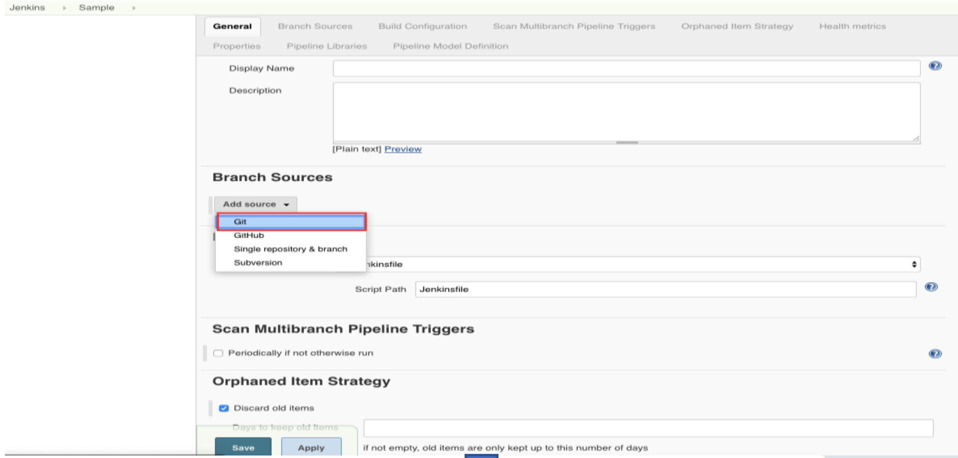
If you want to create a new item from other existing, you can use this option:

Copy from Type to autocomplete

Steps to Create Multi-Branch Pipeline Step 2

- **Step 2:**
  - Give Name, Description & select Git as Branch source as shown

Note: need to git credentials, if the repo is private



Display Name

Description

[Plain text] Preview

Branch Sources

…nkinsfile

Script Path: Jenkinsfile

Scan Multibranch Pipeline Triggers

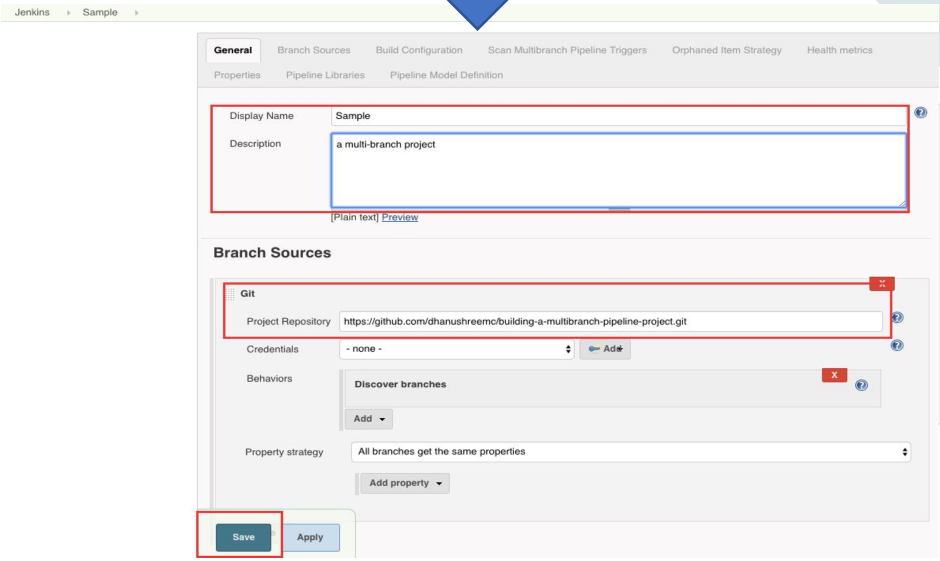
□ Periodically if not otherwise run

Orphaned Item Strategy

☑ Discard old items

Days to keep old items

if not empty, old items are only kept up to this number of days



Display Name: Sample

Description: a multi-branch project

[Plain text] Preview

Branch Sources

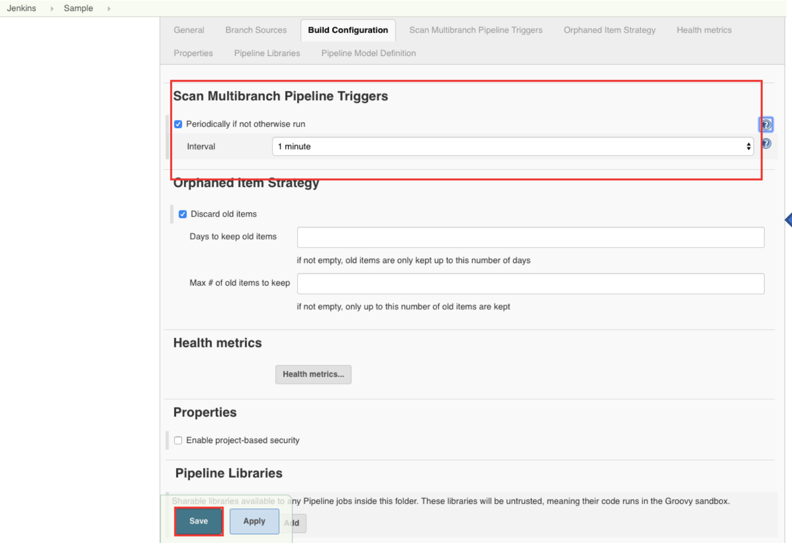
Git

Project Repository: https://github.com/dhanushreemc/building-a-multibranch-pipeline-project.git

Credentials: - none -

Behaviors: Discover branches

Property strategy: All branches get the same properties



Scan Multibranch Pipeline Triggers

☑ Periodically if not otherwise run

Interval: 1 minute

Orphaned Item Strategy

☑ Discard old items

Days to keep old items

if not empty, old items are only kept up to this number of days

Max # of old items to keep

if not empty, only up to this number of old items are kept

Health metrics

Properties

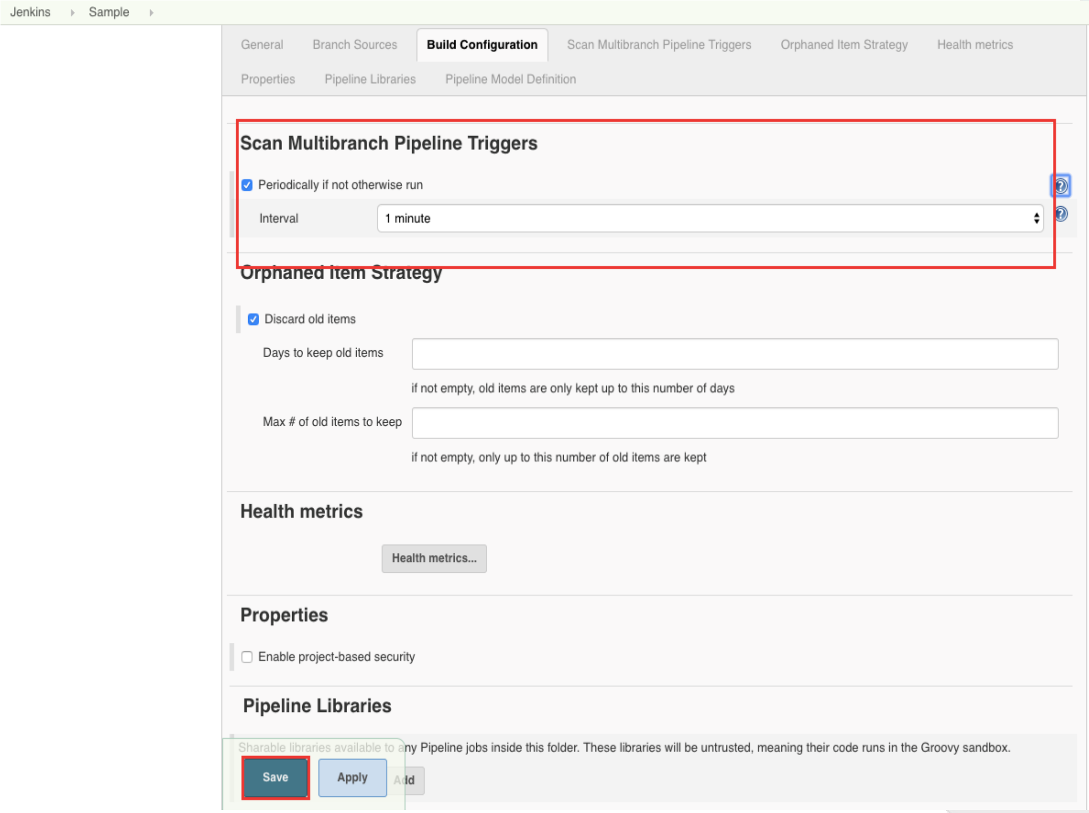
□ Enable project-based security

Pipeline Libraries

Sharable libraries available to any Pipeline jobs inside this folder. These libraries will be untrusted, meaning their code runs in the Groovy sandbox.

Steps to Create Multi-Branch Pipeline Step 3

- **Step 3:**
  - Select for 1 min from drop down for testing
  - Discard old builds by setting days to keep old items
  - Also Specify how many no. of old items to keep, remaining all other old builds will be cleaned
  - Enable project based security by selecting enable project based security to allow a group of users to use this job based on some restrictions
  - e.g.: only can read, build, view, update, etc.



Scan Multibranch Pipeline Triggers

☑ Periodically if not otherwise run

Interval: 1 minute

Orphaned Item Strategy

☑ Discard old items

Days to keep old items

if not empty, old items are only kept up to this number of days

Max # of old items to keep

if not empty, only up to this number of old items are kept

Health metrics

Properties

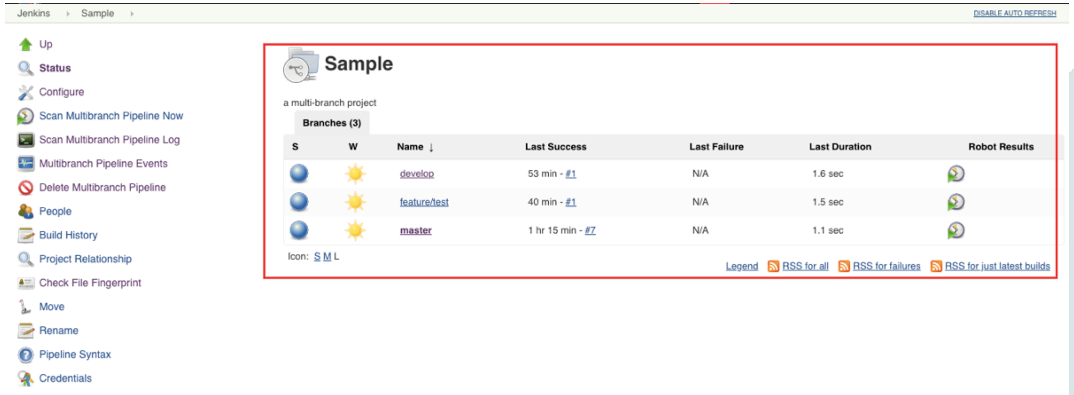
□ Enable project-based security

Pipeline Libraries

Sharable libraries available to any Pipeline jobs inside this folder. These libraries will be untrusted, meaning their code runs in the Groovy sandbox.

Cont’d…

- git clone https://github.com/dhanushreemc/building-a-multibranch-pipeline-project.git
- Get inside the repo and create branches
  - $cd building-a-multibranch-pipeline-project
  - $git checkout –b develop
  - git add .
  - $ git commit –m “creating new develop branch”
  - $ git push origin develop
- same way create feature/test branch



Sample

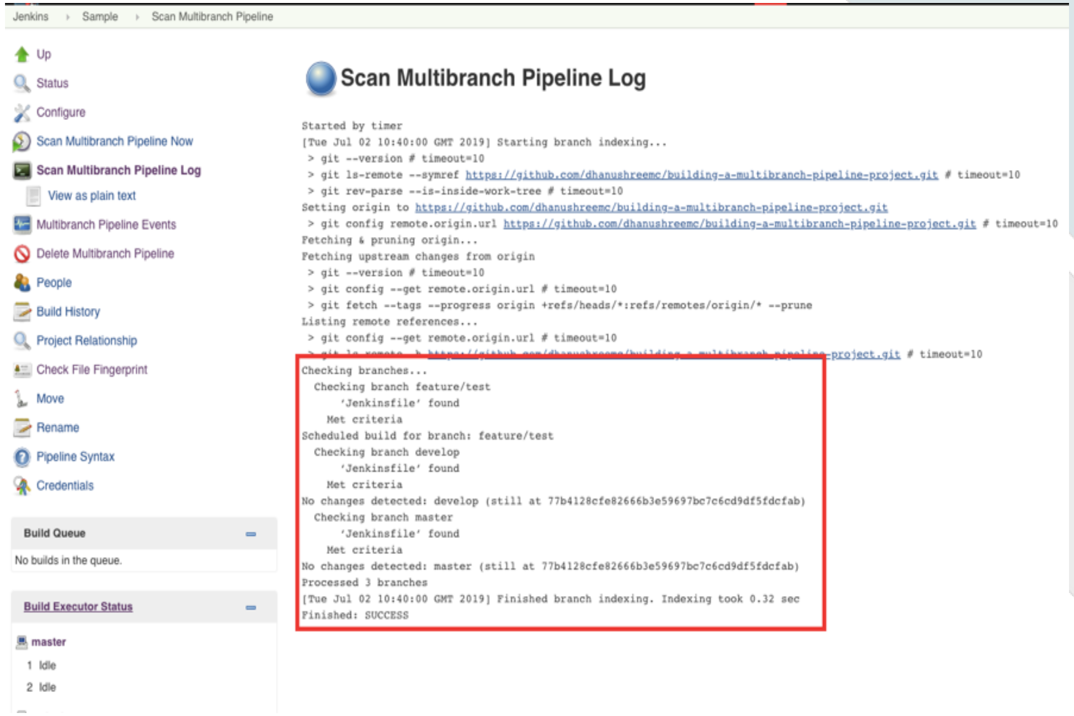
a multi-branch project

Branches (3)

S	W	Name ↓	Last Success	Last Failure	Last Duration	Robot Results
●	☀	develop	53 min - #1	N/A	1.6 sec	
●	☀	feature/test	40 min - #1	N/A	1.5 sec	
●	☀	master	1 hr 15 min - #7	N/A	1.1 sec	

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds



Scan Multibranch Pipeline Log

```
Started by timer
[Tue Jul 02 10:40:00 GMT 2019] Starting branch indexing...
 > git --version # timeout=10
 > git ls-remote --symref https://github.com/dhanushreemc/building-a-multibranch-pipeline-project.git # timeout=10
 > git rev-parse --is-inside-work-tree # timeout=10
Setting origin to https://github.com/dhanushreemc/building-a-multibranch-pipeline-project.git
 > git config remote.origin.url https://github.com/dhanushreemc/building-a-multibranch-pipeline-project.git # timeout=10
Fetching & pruning origin...
Fetching upstream changes from origin
 > git --version # timeout=10
 > git config --get remote.origin.url # timeout=10
 > git fetch --tags --progress origin +refs/heads/*:refs/remotes/origin/* --prune
Listing remote references...
 > git config --get remote.origin.url # timeout=10
 > git ls-remote -h https://github.com/dhanushreemc/building-a-multibranch-pipeline-project.git # timeout=10
Checking branches...
  Checking branch feature/test
      ‘Jenkinsfile’ found
    Met criteria
Scheduled build for branch: feature/test
  Checking branch develop
      ‘Jenkinsfile’ found
    Met criteria
No changes detected: develop (still at 77b4128cfe82666b3e59697bc7c6cd9df5fdcfab)
  Checking branch master
      ‘Jenkinsfile’ found
    Met criteria
No changes detected: master (still at 77b4128cfe82666b3e59697bc7c6cd9df5fdcfab)
Processed 3 branches
[Tue Jul 02 10:40:00 GMT 2019] Finished branch indexing. Indexing took 0.32 sec
Finished: SUCCESS
```

Cont'd…

Note: we can even create branches directly in GitHub repo

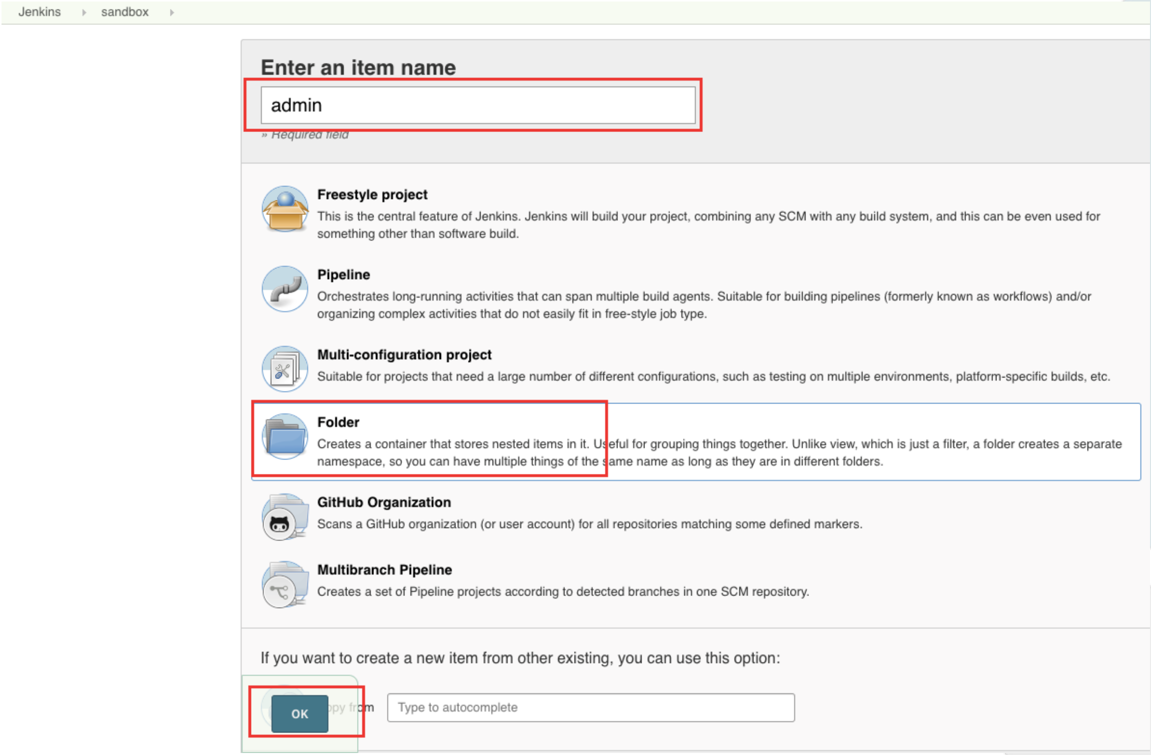
- After pushing branch successfully, according to branch indexing, Jenkins should build the develop branch
- Additional environment variable supported by Multibranch pipeline are:
  - **BRANCH_NAME**
    - Name of the branch for which this Pipeline is executing, for example master
  - **CHANGE_ID**
    - An identifier corresponding to some kind of change request, such as a pull request number

Folder Job

- Created in Jenkins to group jobs under one folder
- Can create multiple jobs of same name as long as they are in different folder

Steps to Create Folder Jobs Step 1

- **Step 1:**
  - Go to Jenkins Dashboard
  - Click on New item
  - Give proper name
  - Select folder
  - Click on Ok



Jenkins › sandbox ›

Enter an item name

admin

» 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.

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.

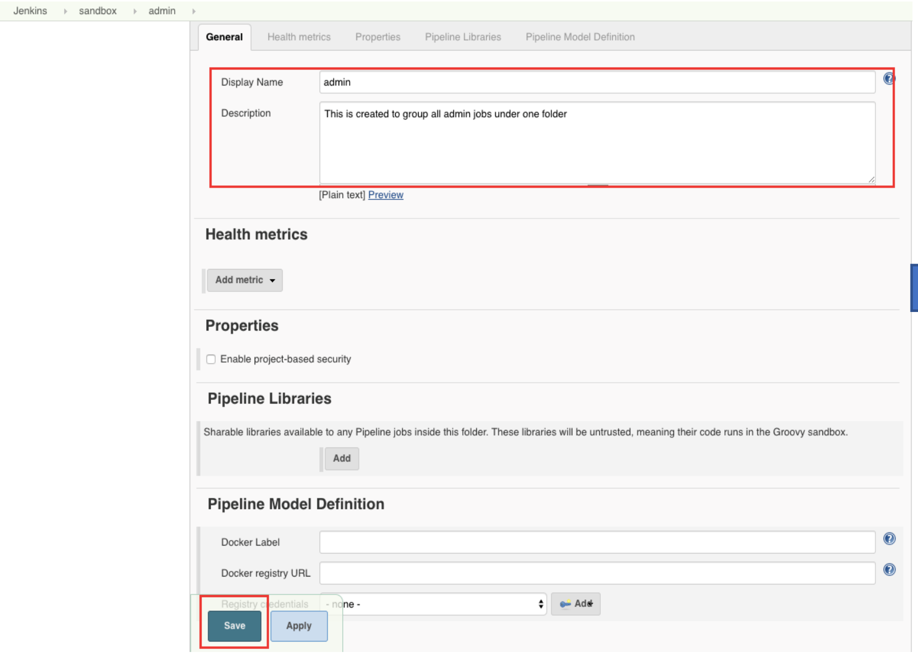
GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:

Steps to Create Folder Jobs Step 2

- **Step 2:**
  - Add Display name & description
  - Click on save

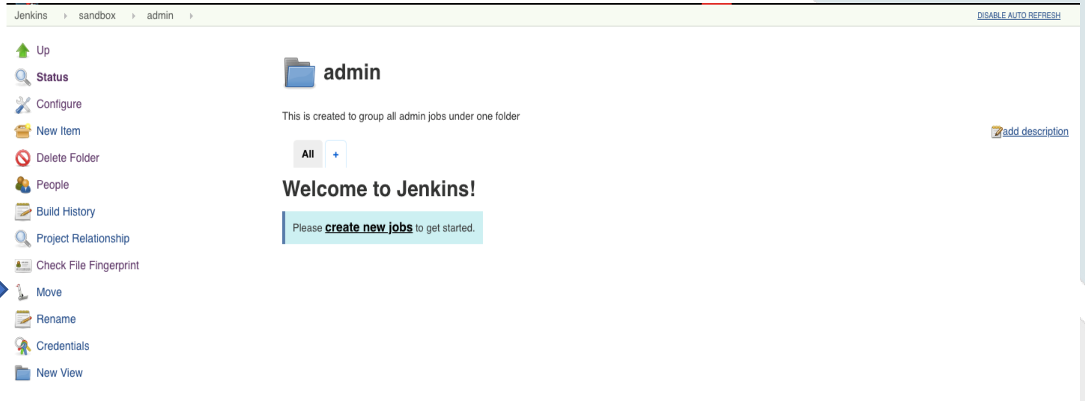


Display Name: admin

Description: This is created to group all admin jobs under one folder

□ Enable project-based security

Sharable libraries available to any Pipeline jobs inside this folder. These libraries will be untrusted, meaning their code runs in the Groovy sandbox.



admin

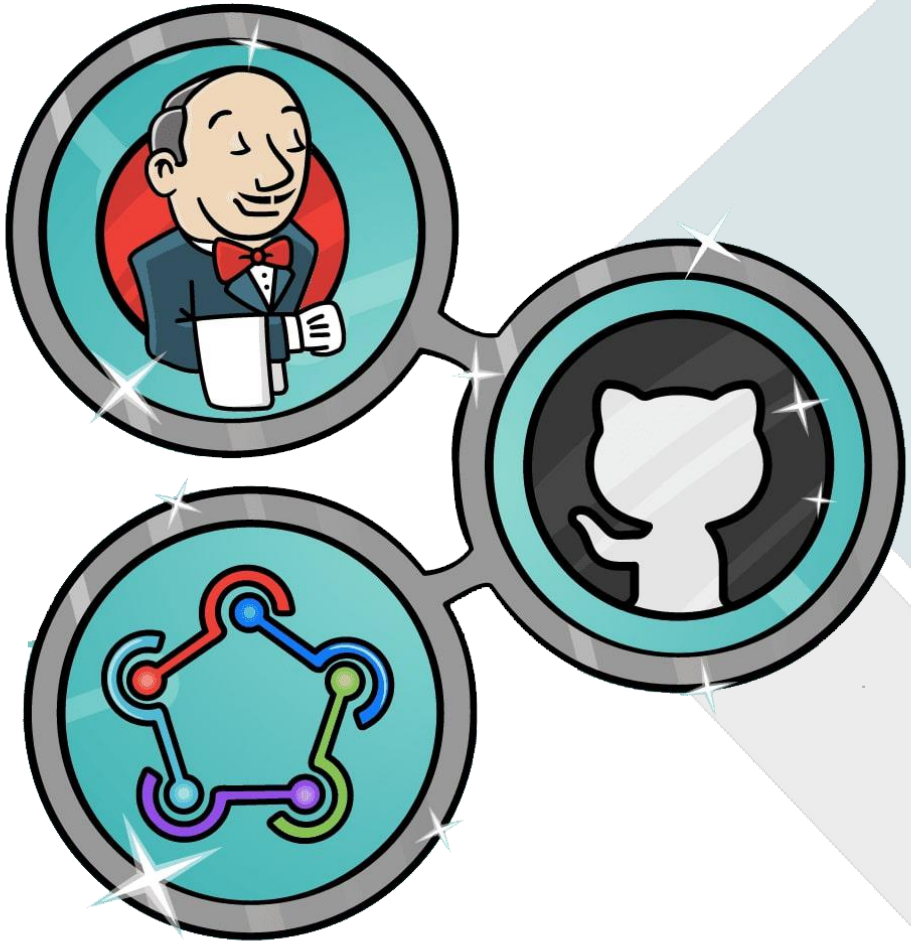
This is created to group all admin jobs under one folder

Welcome to Jenkins!

Please **create new jobs** to get started.

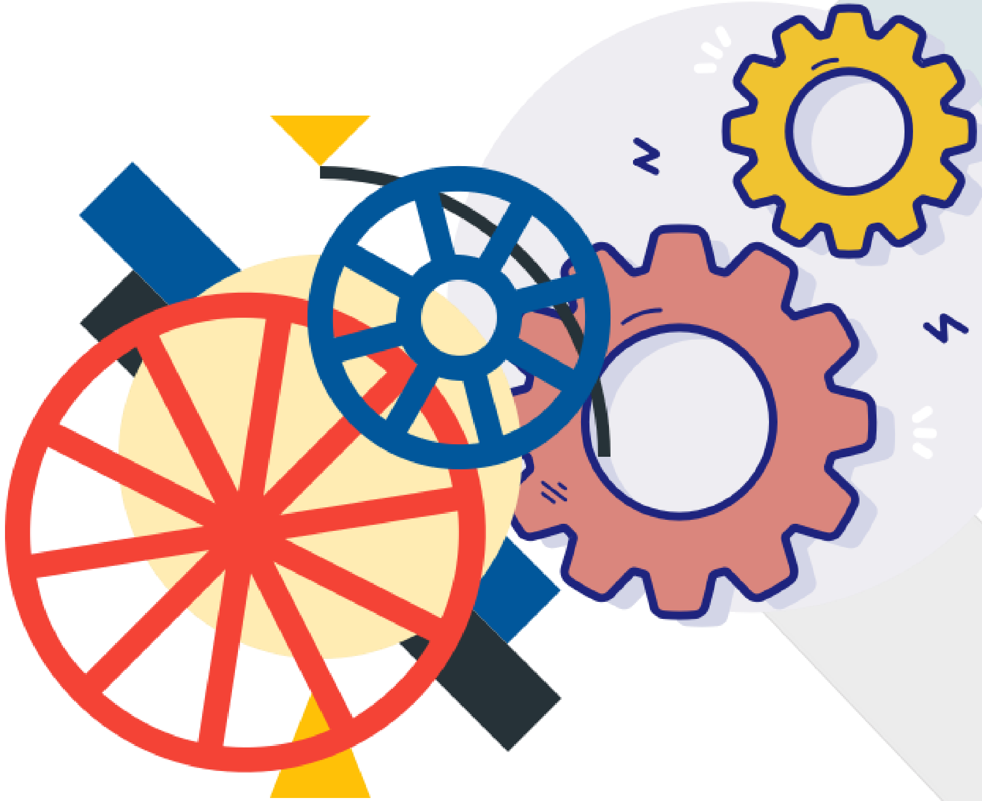
GitHub Organization Job

- Used to scan all repositories belonging to a Git Hub organization
- Build each branch of each repo of an organization



Multi-Configuration Job

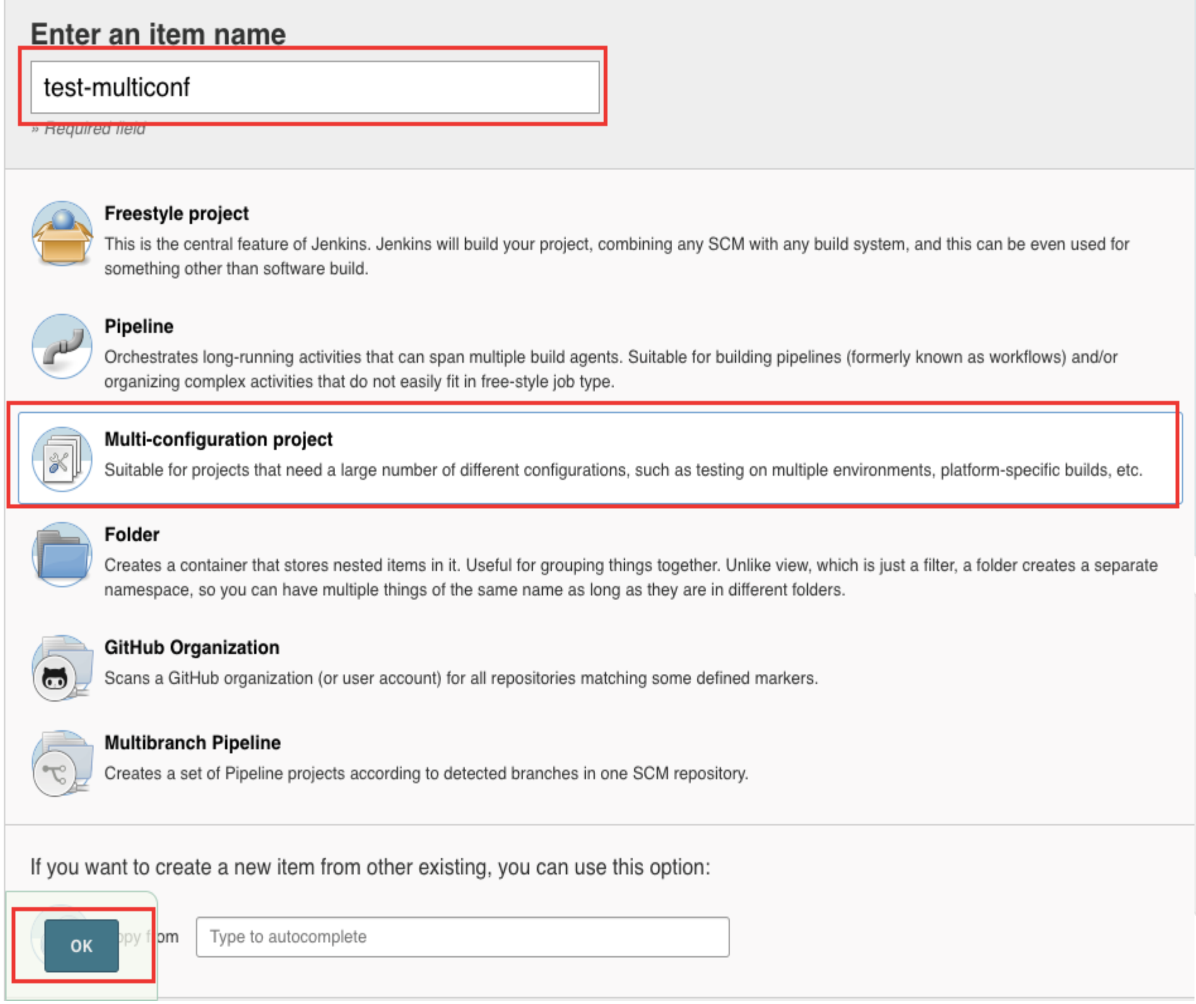
- Build jobs are an extremely powerful feature of Jenkins
- Like any other build job, but with a significant additional element: **the Configuration Matrix**
- This is where you define the different configurations that will be used to run your builds
- Can create only one job with many configurations
- Each configuration will be executed as a separate job



Steps to create multi-configuration Job Step 1

Step 1:

- Go to Jenkins dashboard
- Click on ‘New item’
- Enter job name
- Select “Multi-configuration project”
- Click on OK



Enter an item name

test-multiconf

» 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.

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.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

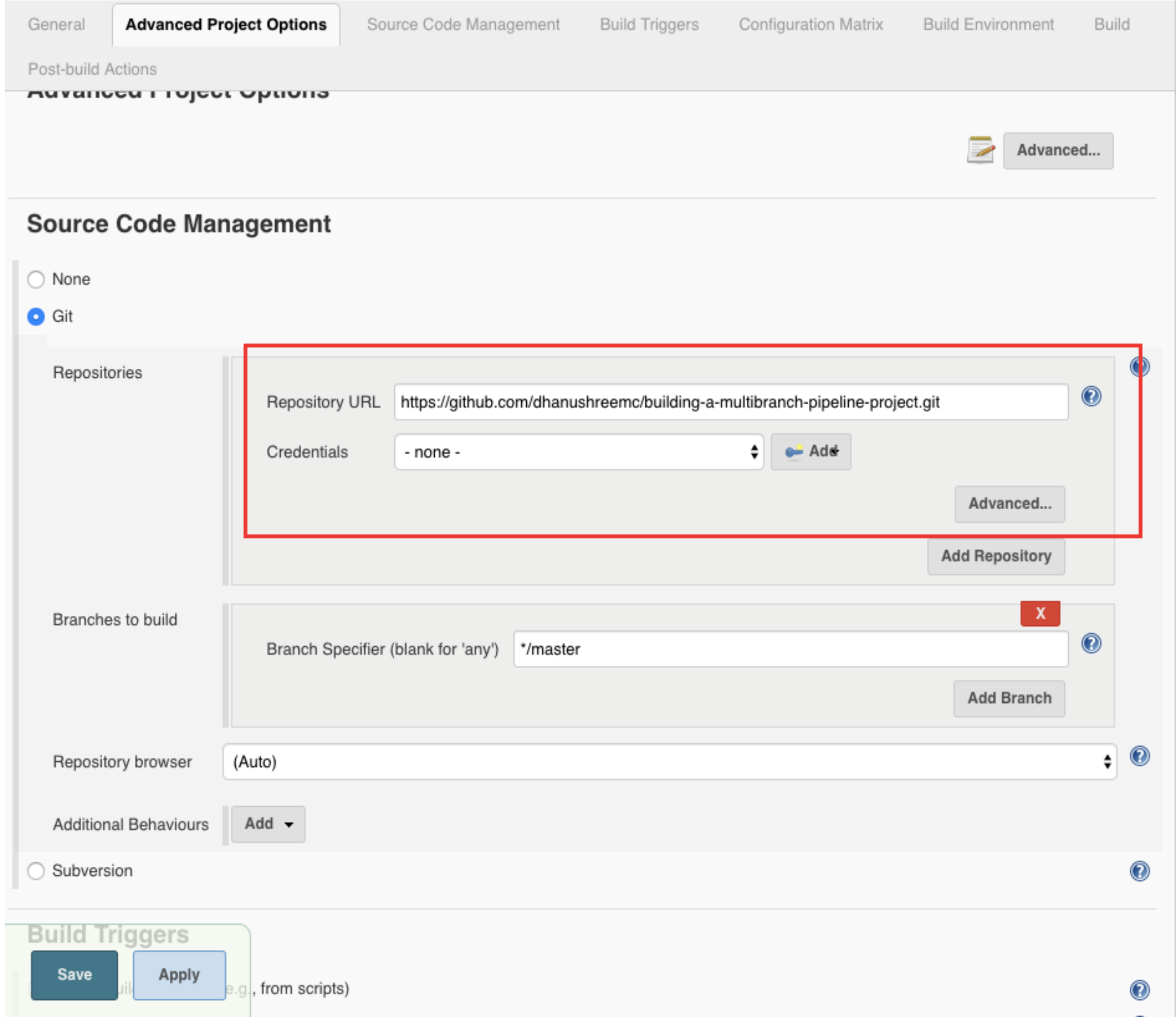
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:

Steps to create multi-configuration Job Step 2

Step 2:

- Give description
- Go to source code management section
- Select Git
- Enter GitHub URL, credentials(if the repo is private)



Source Code Management

○ None
● Git

Repositories

Repository URL: https://github.com/dhanushreemc/building-a-multibranch-pipeline-project.git

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

○ Subversion

Steps to create multi-configuration Job Step 3

Step 3:

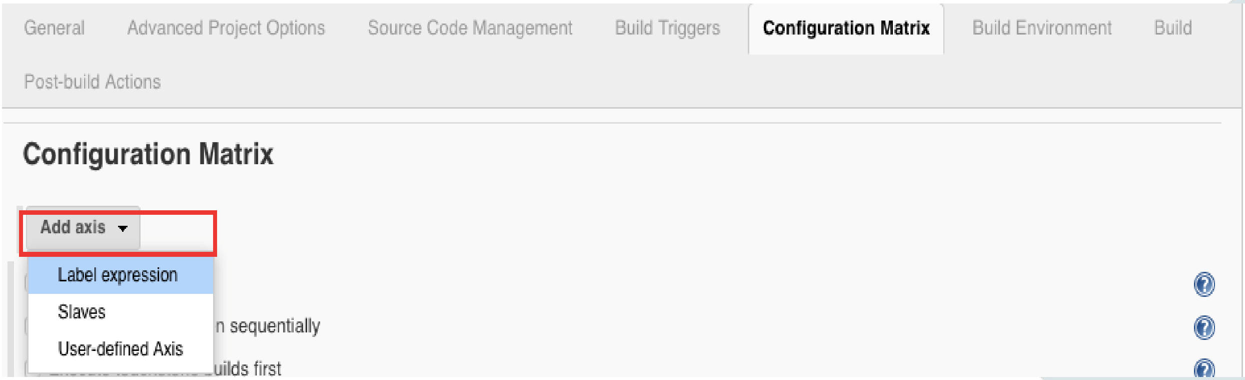
- Under Configuration matrix click on Add Axis, & select any one option

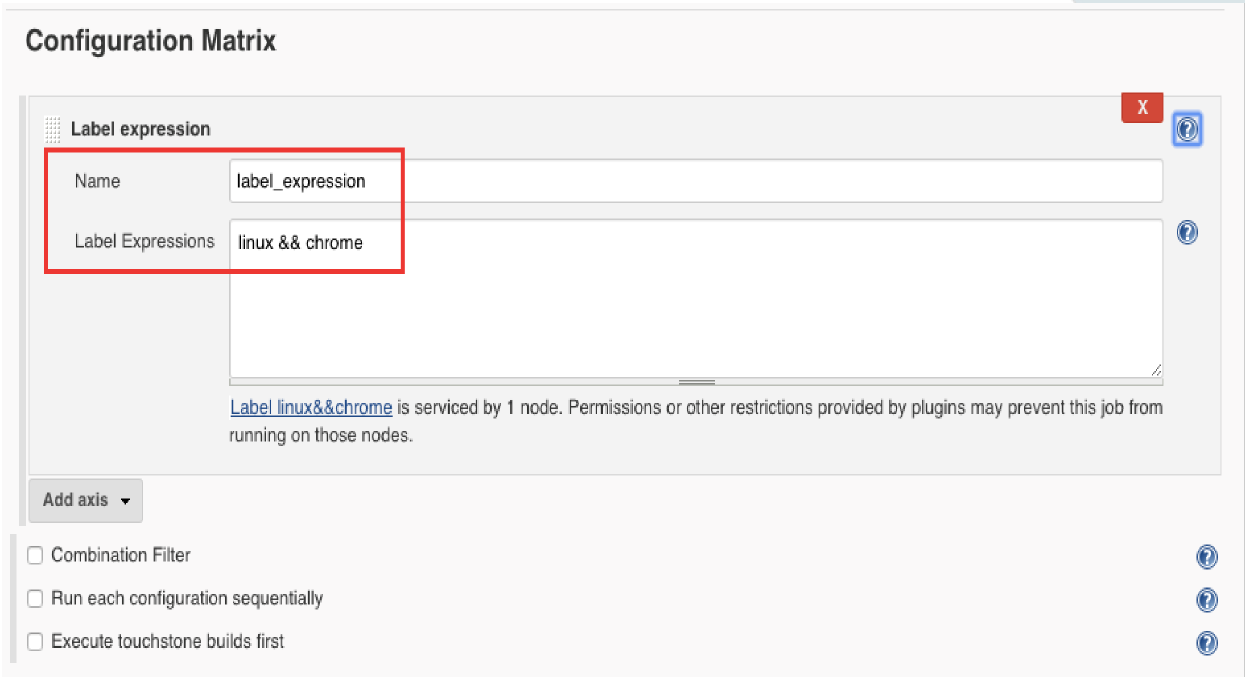
Label expression:

- This axis allows to run same job on multiple slave
- Supports to use Boolean expressions like foo && bar

Example: Linux && chrome

Note: Linux and chrome are the labels defined for Jenkins slaves





Name: label_expression

Label Expressions: linux && chrome

Label linux&&chrome is serviced by 1 node. Permissions or other restrictions provided by plugins may prevent this job from running on those nodes.

☐ Combination Filter

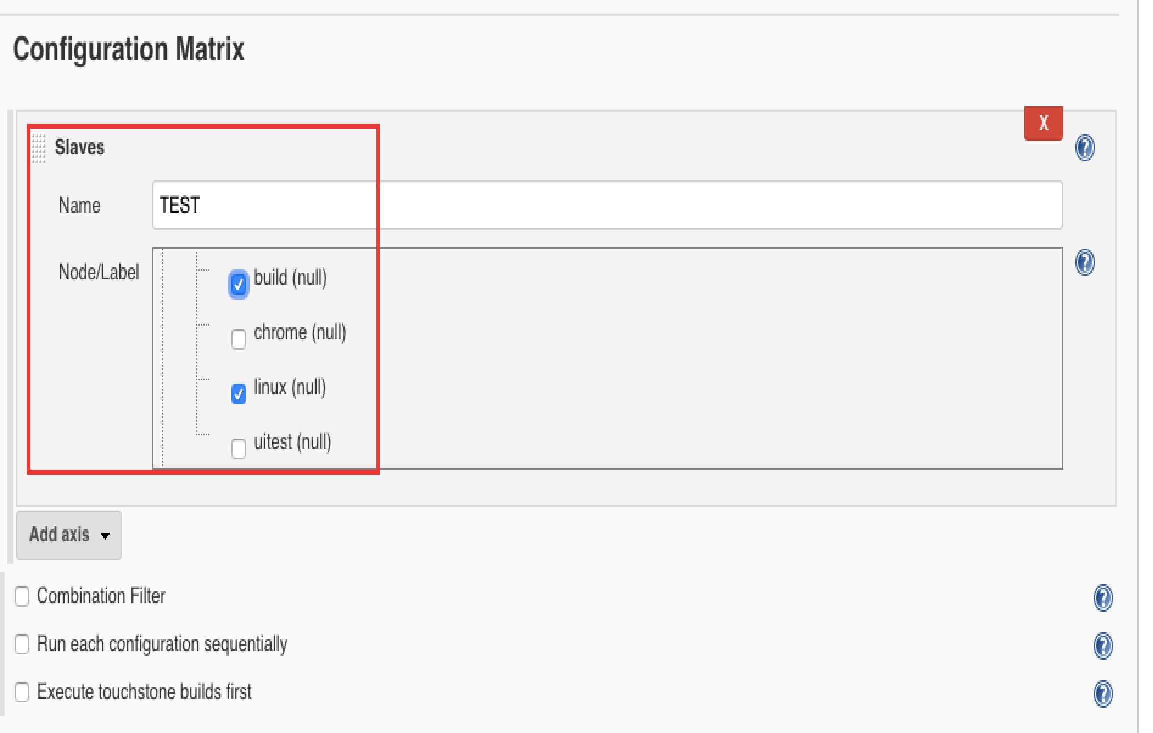
☐ Run each configuration sequentially

☐ Execute touchstone builds first

Cont'd…

Slaves:

- This axis allows to run the same build on multiple slaves
- Useful for example to cross-compile your C projects on multiple platforms, testing your Java projects on different OS
- This figure shows we have selected labels build & Linux to run jobs on these labelled nodes



Configuration Matrix

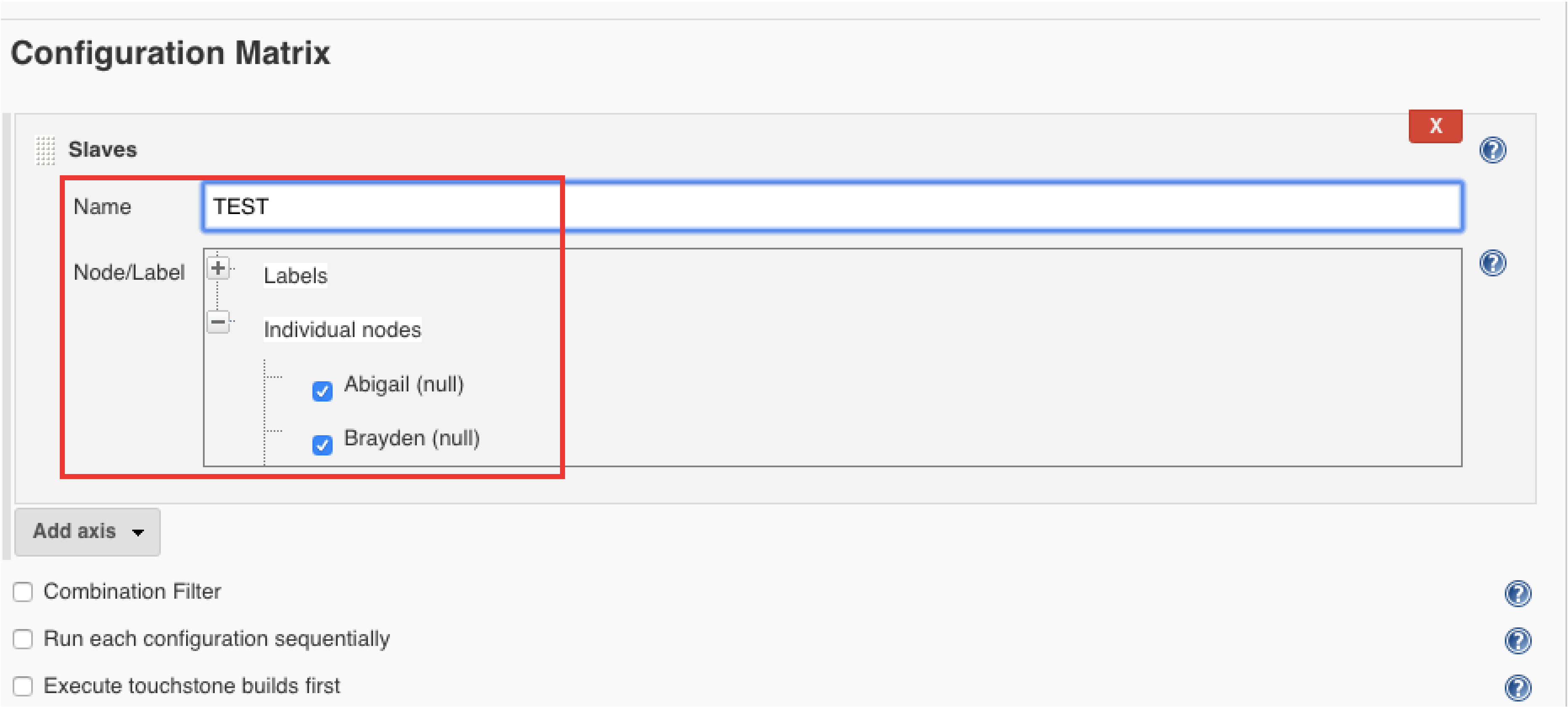
Slaves

Name: TEST

Node/Label:
☑ build (null)
☐ chrome (null)
☑ linux (null)
☐ uitest (null)

☐ Combination Filter
☐ Run each configuration sequentially
☐ Execute touchstone builds first

Cont’d…



Configuration Matrix

Slaves

Name: TEST

Node/Label:

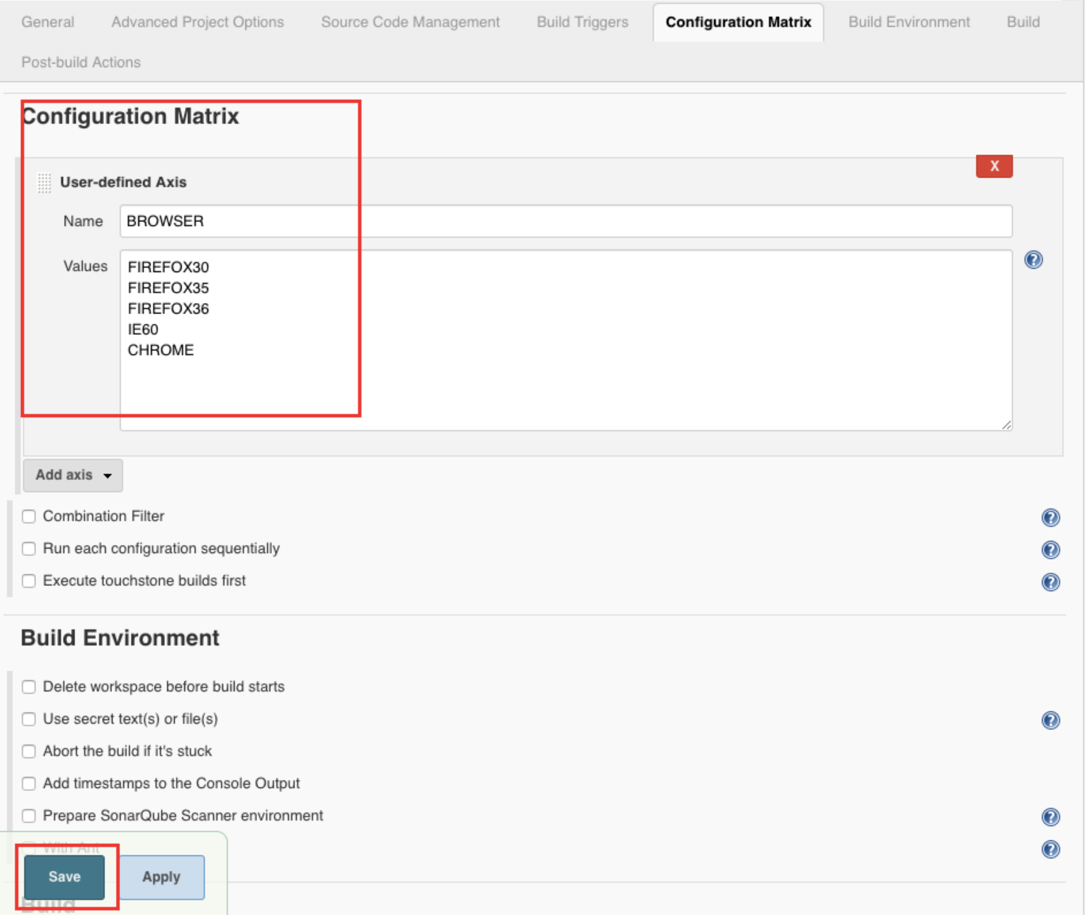
- Labels
- Individual nodes
  - ☑ Abigail (null)
  - ☑ Brayden (null)

- □ Combination Filter
- □ Run each configuration sequentially
- □ Execute touchstone builds first

Cont'd…

User-defined Axis:

- This is used to define Variable and values
- So that each configuration will run as different job as shown



Configuration Matrix

User-defined Axis

Name: BROWSER

Values:
FIREFOX30
FIREFOX35
FIREFOX36
IE60
CHROME

☐ Combination Filter

☐ Run each configuration sequentially

☐ Execute touchstone builds first

Build Environment

☐ Delete workspace before build starts

☐ Use secret text(s) or file(s)

☐ Abort the build if it's stuck

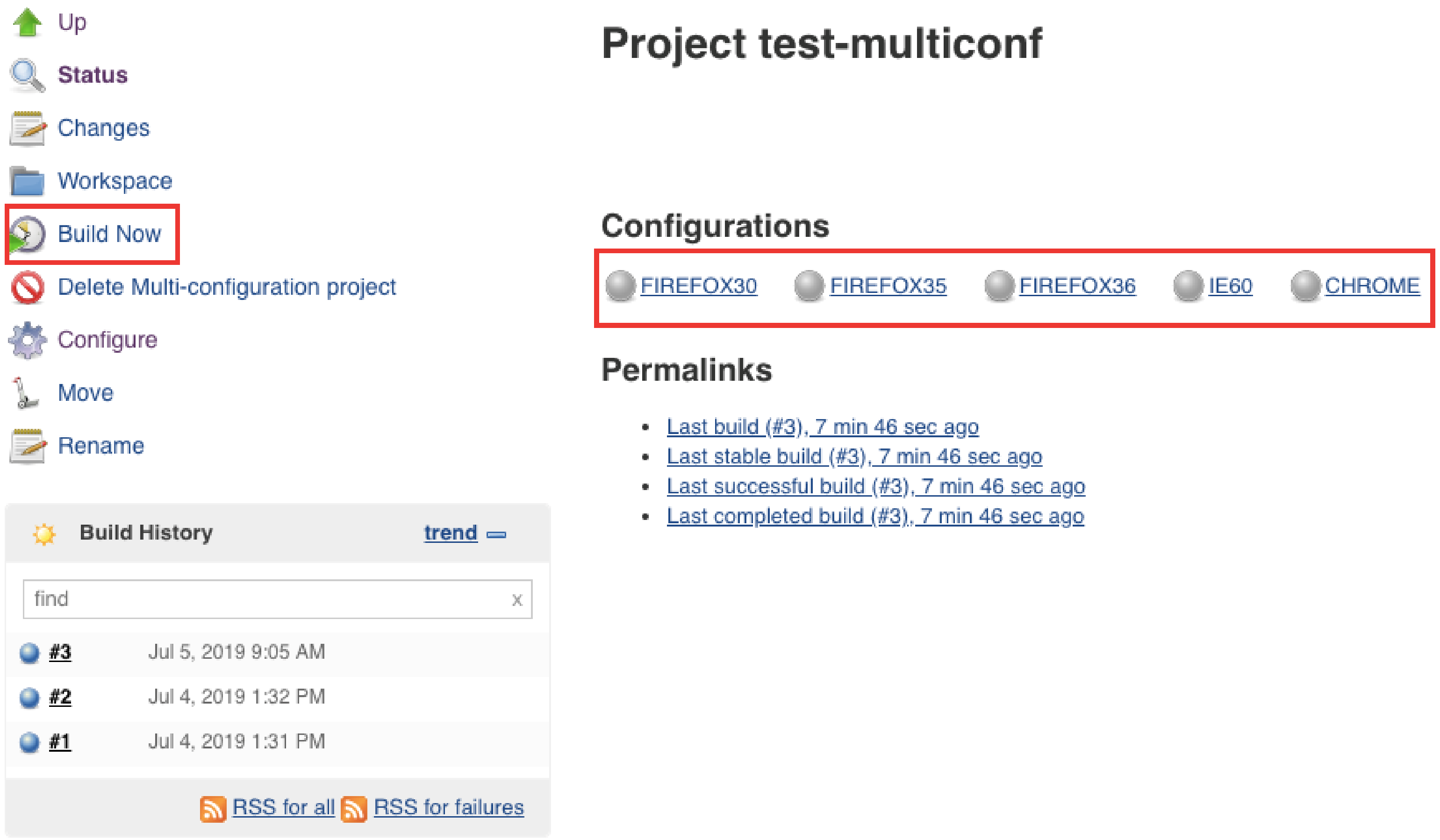
☐ Add timestamps to the Console Output

☐ Prepare SonarQube Scanner environment

☐ With Ant

Cont'd...

- After we select User defined axis
- Click on Ok



Project test-multiconf

Configurations

FIREFOX30 FIREFOX35 FIREFOX36 IE60 CHROME

Permalinks

- Last build (#3), 7 min 46 sec ago
- Last stable build (#3), 7 min 46 sec ago
- Last successful build (#3), 7 min 46 sec ago
- Last completed build (#3), 7 min 46 sec ago

Build History trend

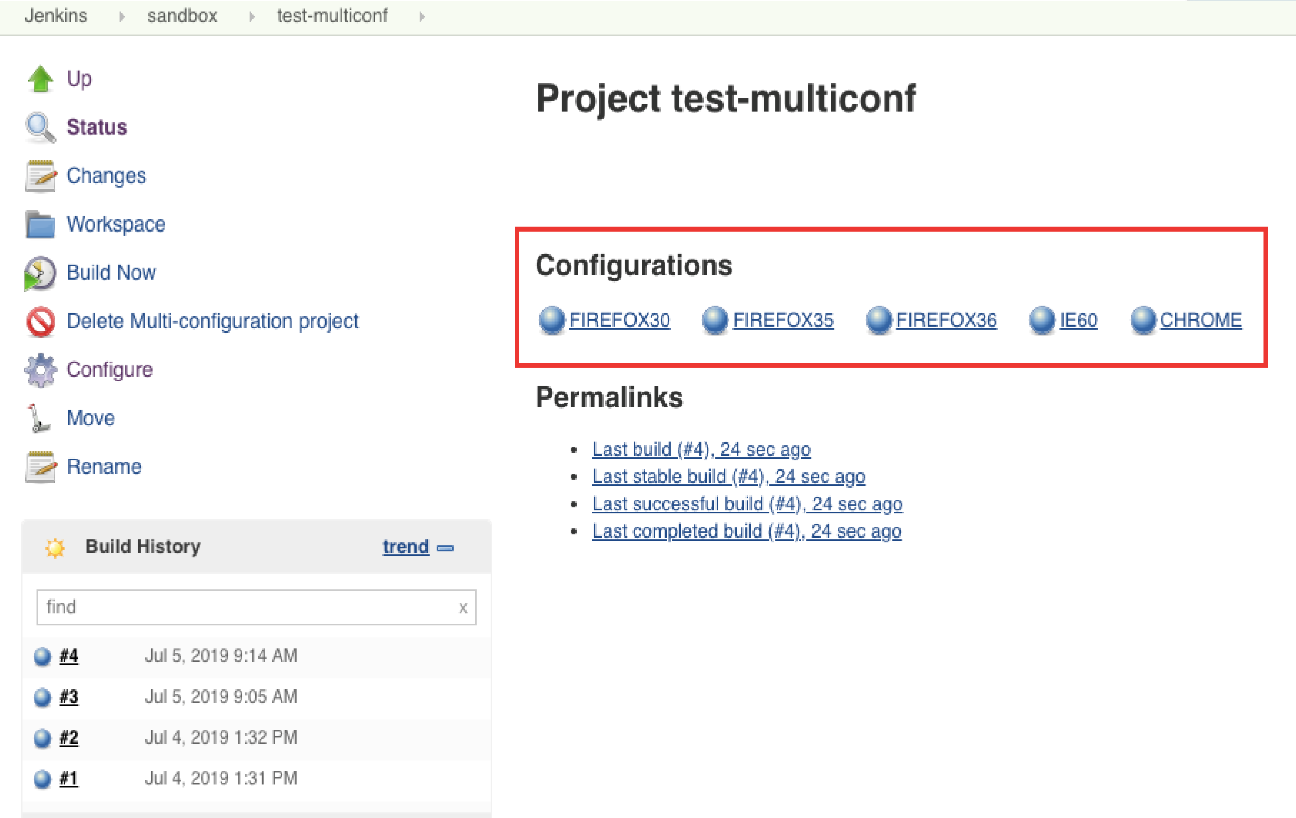
find x

#3	Jul 5, 2019 9:05 AM
#2	Jul 4, 2019 1:32 PM
#1	Jul 4, 2019 1:31 PM

RSS for all RSS for failures

Cont’d…

- After selecting user defined axis and entered Name and values as shown
- For each value Jenkins created a separate job once we build,
- By default the builds are parallel meaning jobs for each value defined will run parallel on specified slave
- After clicking Build now, we can se the jobs running as shown



Jenkins › sandbox › test-multiconf ›

- Up
- Status
- Changes
- Workspace
- Build Now
- Delete Multi-configuration project
- Configure
- Move
- Rename

Build History trend

find

#4	Jul 5, 2019 9:14 AM
#3	Jul 5, 2019 9:05 AM
#2	Jul 4, 2019 1:32 PM
#1	Jul 4, 2019 1:31 PM

Project test-multiconf

Configurations

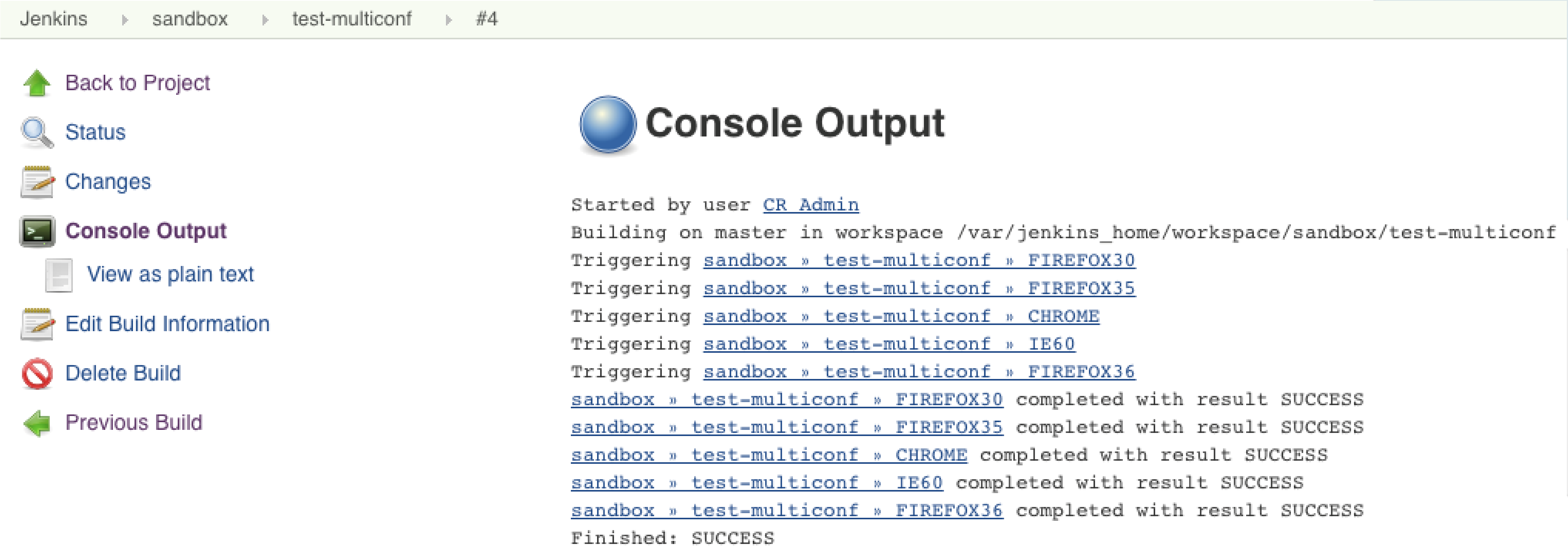
FIREFOX30 FIREFOX35 FIREFOX36 IE60 CHROME

Permalinks

- Last build (#4), 24 sec ago
- Last stable build (#4), 24 sec ago
- Last successful build (#4), 24 sec ago
- Last completed build (#4), 24 sec ago

Cont'd…

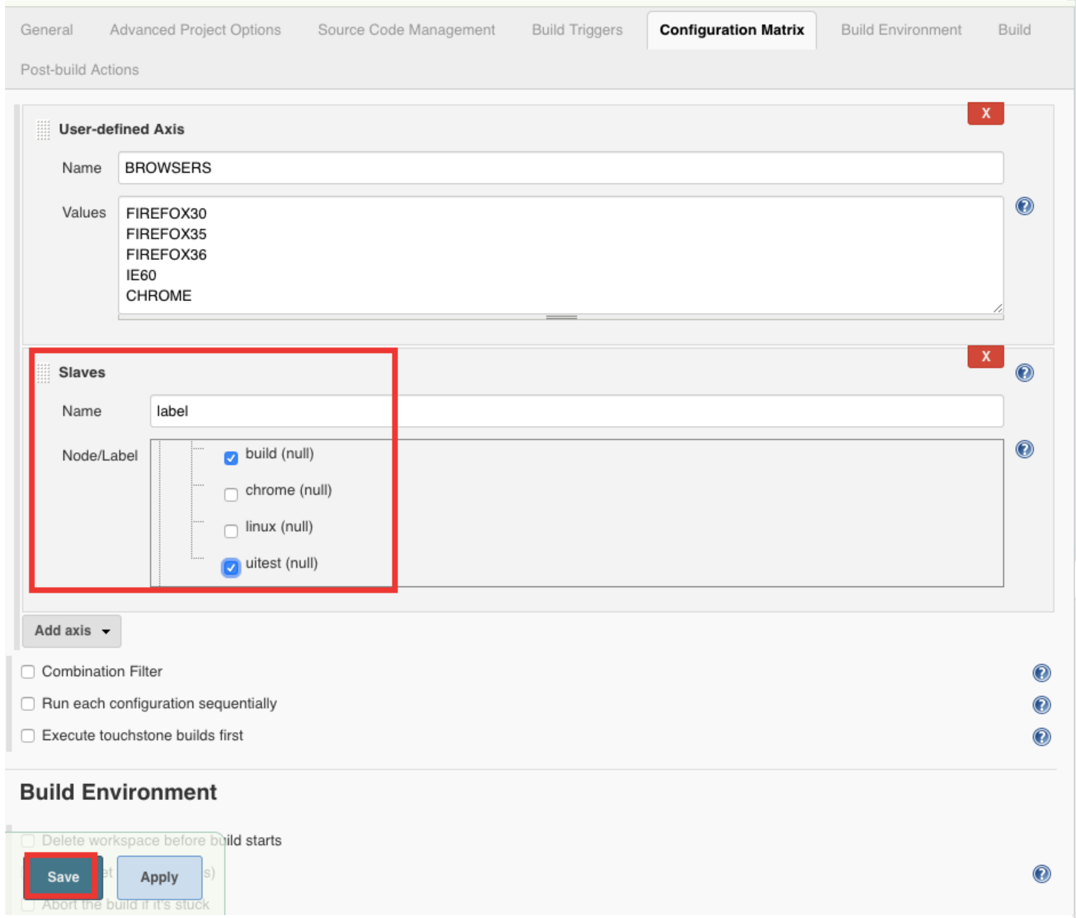
- If we see the console, we will get to know how the multi-configuration job triggers separate jobs parallel to run on different configurations



```
Started by user CR Admin
Building on master in workspace /var/jenkins_home/workspace/sandbox/test-multiconf
Triggering sandbox » test-multiconf » FIREFOX30
Triggering sandbox » test-multiconf » FIREFOX35
Triggering sandbox » test-multiconf » CHROME
Triggering sandbox » test-multiconf » IE60
Triggering sandbox » test-multiconf » FIREFOX36
sandbox » test-multiconf » FIREFOX30 completed with result SUCCESS
sandbox » test-multiconf » FIREFOX35 completed with result SUCCESS
sandbox » test-multiconf » CHROME completed with result SUCCESS
sandbox » test-multiconf » IE60 completed with result SUCCESS
sandbox » test-multiconf » FIREFOX36 completed with result SUCCESS
Finished: SUCCESS
```

Cont’d…

- Modify the job configuration so that we will add rule to run this job on only some slaves
- Click on save, we can see the matrix of jobs & slaves



User-defined Axis

Name: BROWSERS

Values:
FIREFOX30
FIREFOX35
FIREFOX36
IE60
CHROME

Slaves

Name: label

Node/Label:
- ☑ build (null)
- ☐ chrome (null)
- ☐ linux (null)
- ☑ uitest (null)

☐ Combination Filter

☐ Run each configuration sequentially

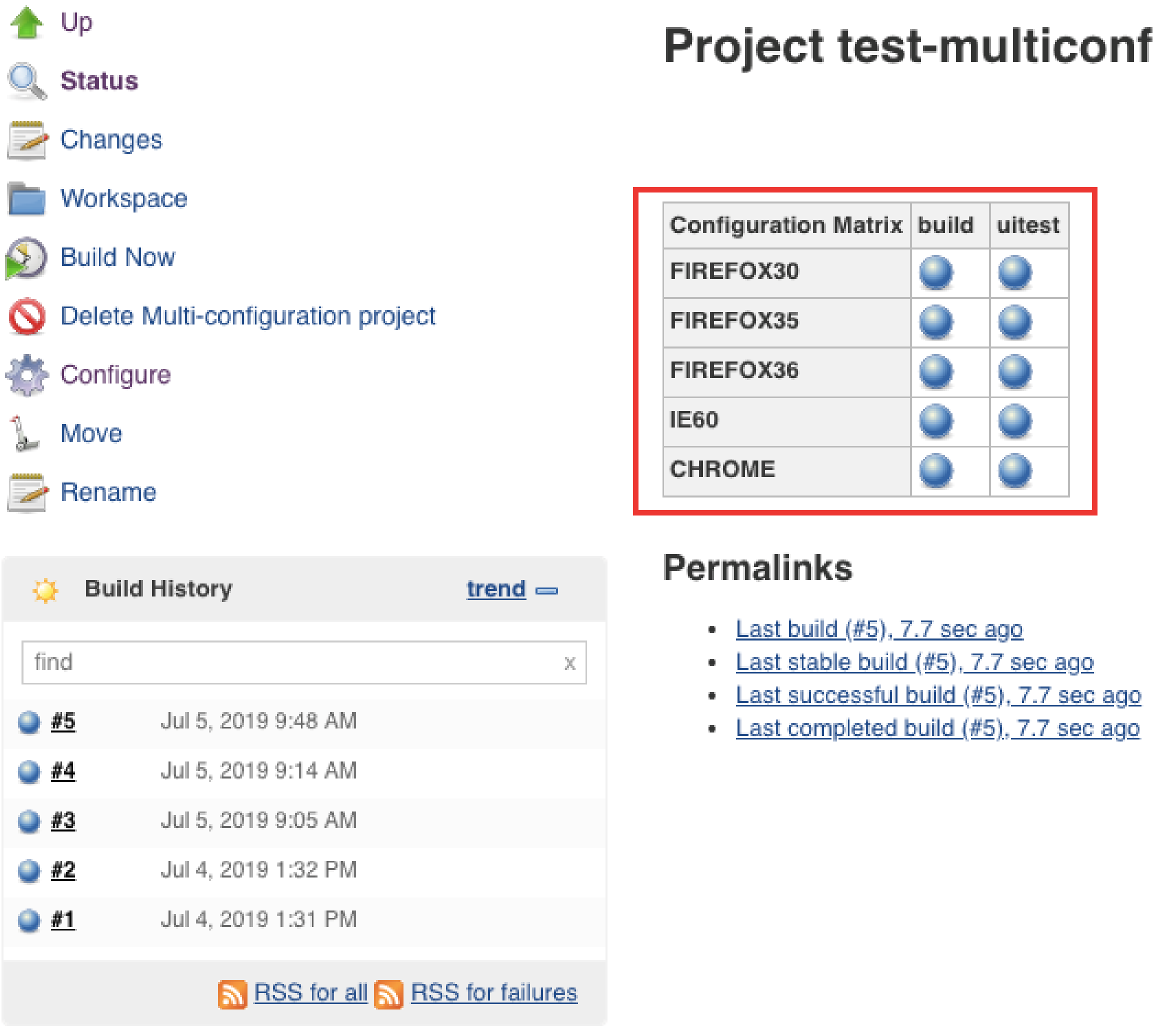
☐ Execute touchstone builds first

Build Environment

☐ Delete workspace before build starts

☐ Abort the build if it's stuck

Cont’d…



- Up
- Status
- Changes
- Workspace
- Build Now
- Delete Multi-configuration project
- Configure
- Move
- Rename

Project test-multiconf

Configuration Matrix	build	uitest
FIREFOX30	●	●
FIREFOX35	●	●
FIREFOX36	●	●
IE60	●	●
CHROME	●	●

Build History trend

find x

●	#5	Jul 5, 2019 9:48 AM
●	#4	Jul 5, 2019 9:14 AM
●	#3	Jul 5, 2019 9:05 AM
●	#2	Jul 4, 2019 1:32 PM
●	#1	Jul 4, 2019 1:31 PM

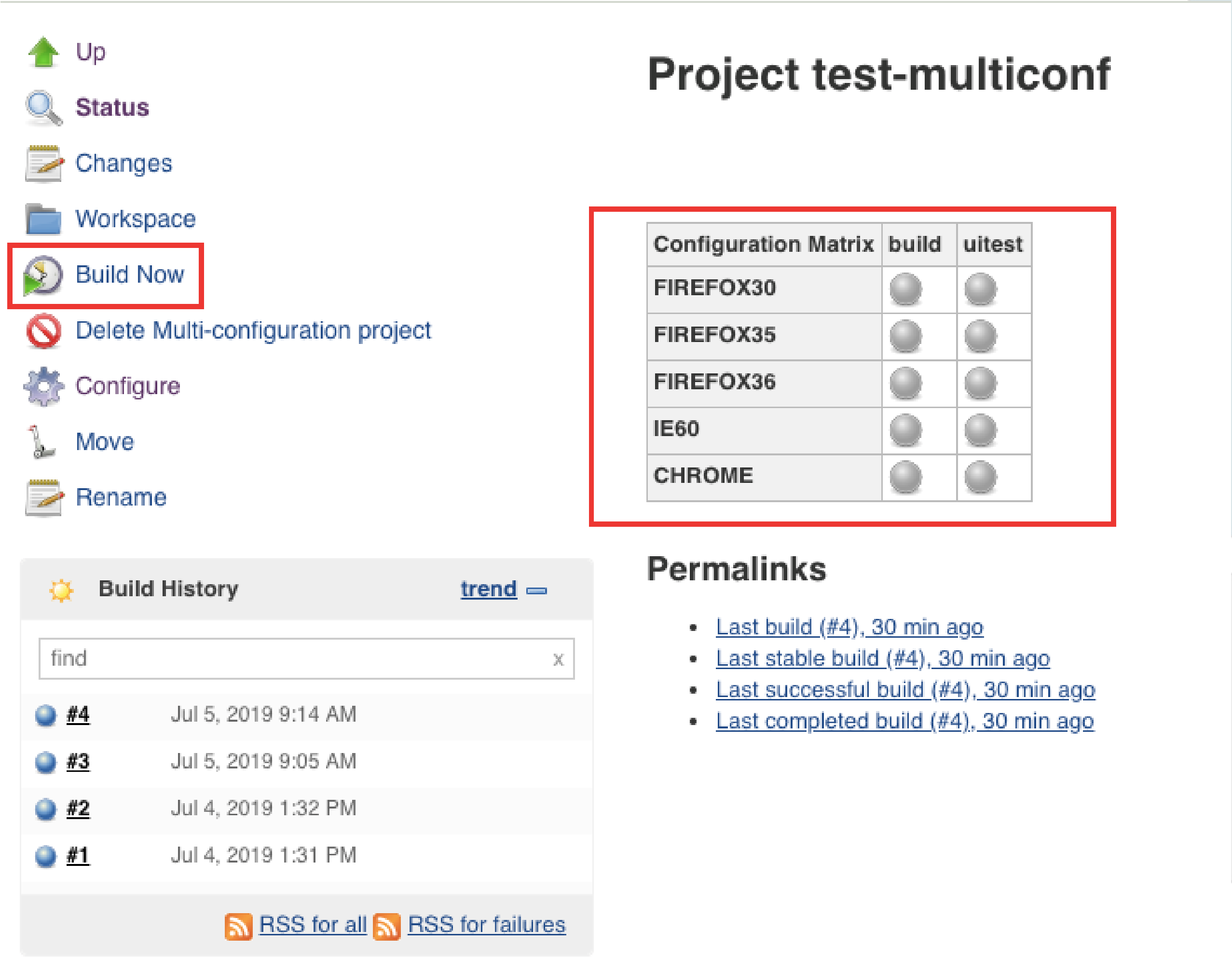
RSS for all RSS for failures

Permalinks

- Last build (#5), 7.7 sec ago
- Last stable build (#5), 7.7 sec ago
- Last successful build (#5), 7.7 sec ago
- Last completed build (#5), 7.7 sec ago

Cont'd…

- We can see here we have selected 2 labels(slaves) "build" and "uitest"
- The matrix configuration will run on the slaves which are labelled as build & "uitest"



Project test-multiconf

Configuration Matrix	build	uitest
FIREFOX30	●	●
FIREFOX35	●	●
FIREFOX36	●	●
IE60	●	●
CHROME	●	●

Permalinks

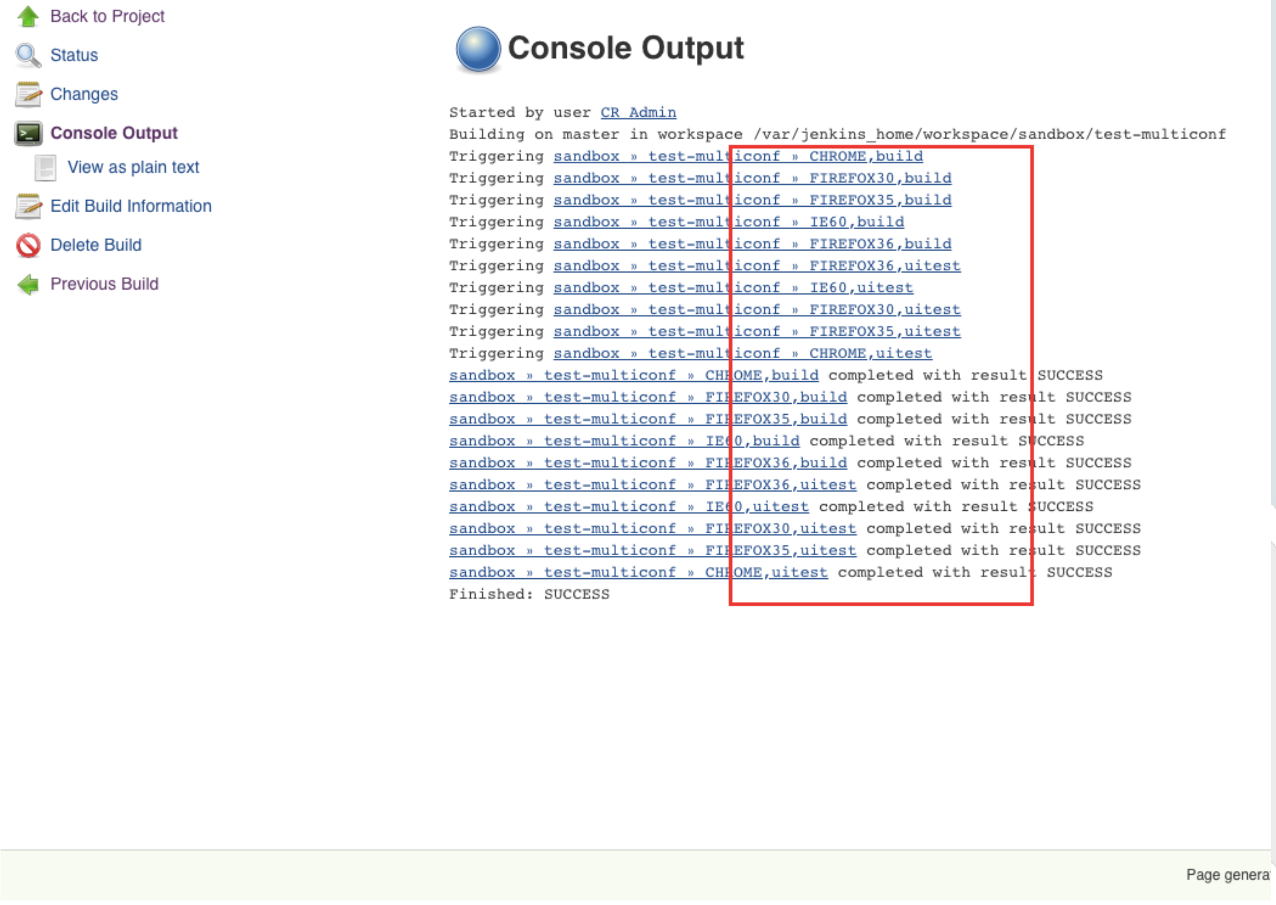
- Last build (#4), 30 min ago
- Last stable build (#4), 30 min ago
- Last successful build (#4), 30 min ago
- Last completed build (#4), 30 min ago

Build History trend

#4	Jul 5, 2019 9:14 AM
#3	Jul 5, 2019 9:05 AM
#2	Jul 4, 2019 1:32 PM
#1	Jul 4, 2019 1:31 PM

Cont’d…

- We can observe that how the jobs are running on different labels(slaves)

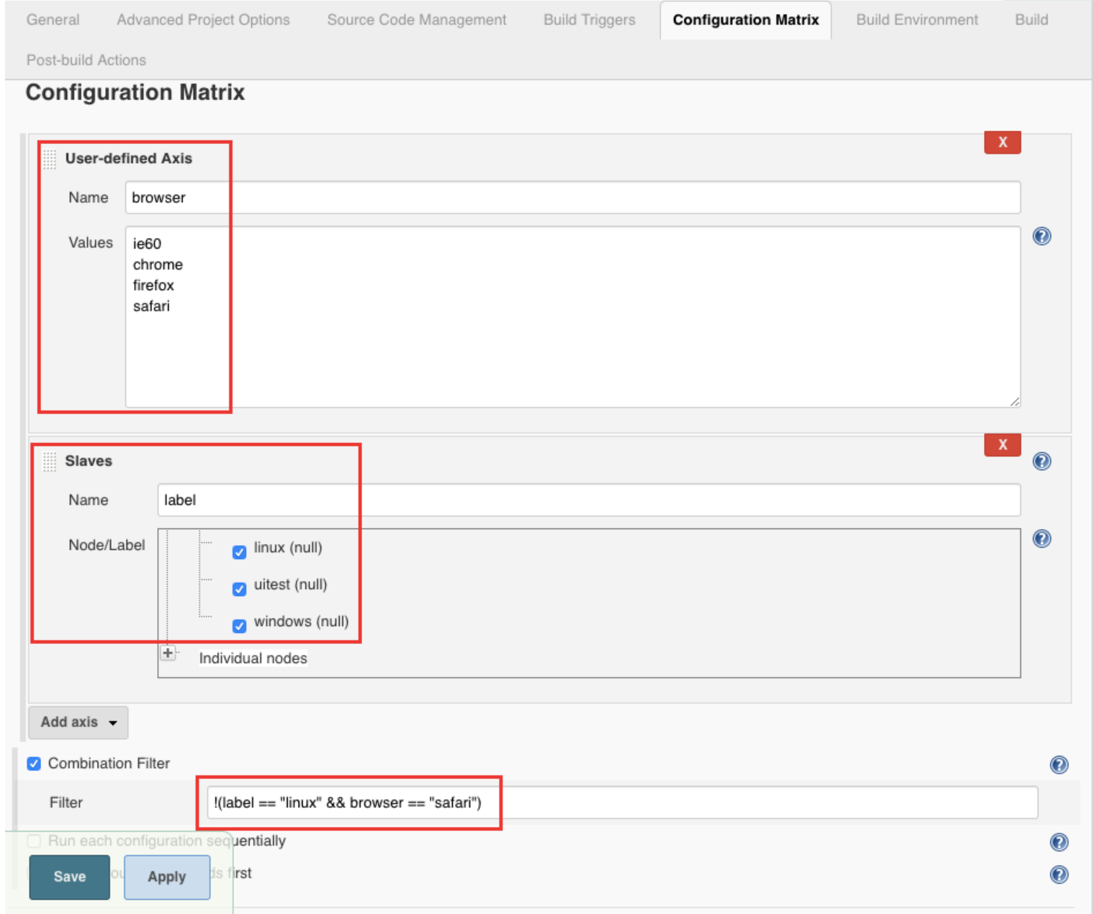


Console Output

```
Started by user CR Admin
Building on master in workspace /var/jenkins_home/workspace/sandbox/test-multiconf
Triggering sandbox » test-multiconf » CHROME,build
Triggering sandbox » test-multiconf » FIREFOX30,build
Triggering sandbox » test-multiconf » FIREFOX35,build
Triggering sandbox » test-multiconf » IE60,build
Triggering sandbox » test-multiconf » FIREFOX36,build
Triggering sandbox » test-multiconf » FIREFOX36,uitest
Triggering sandbox » test-multiconf » IE60,uitest
Triggering sandbox » test-multiconf » FIREFOX30,uitest
Triggering sandbox » test-multiconf » FIREFOX35,uitest
Triggering sandbox » test-multiconf » CHROME,uitest
sandbox » test-multiconf » CHROME,build completed with result SUCCESS
sandbox » test-multiconf » FIREFOX30,build completed with result SUCCESS
sandbox » test-multiconf » FIREFOX35,build completed with result SUCCESS
sandbox » test-multiconf » IE60,build completed with result SUCCESS
sandbox » test-multiconf » FIREFOX36,build completed with result SUCCESS
sandbox » test-multiconf » FIREFOX36,uitest completed with result SUCCESS
sandbox » test-multiconf » IE60,uitest completed with result SUCCESS
sandbox » test-multiconf » FIREFOX30,uitest completed with result SUCCESS
sandbox » test-multiconf » FIREFOX35,uitest completed with result SUCCESS
sandbox » test-multiconf » CHROME,uitest completed with result SUCCESS
Finished: SUCCESS
```

Cont'd…

- Edit configuration as shown & enable filter
- Define rules for filter
- One invalid config: !(label == "Linux" && browser == "Safari")
- That is when label is Linux & the browser is Safari are invalid combination
- Meaning execute job except this combination
- Click on apply, save & build now



Configuration Matrix

User-defined Axis

Name: browser

Values:
ie60
chrome
firefox
safari

Slaves

Name: label

Node/Label:
☑ linux (null)
☑ uitest (null)
☑ windows (null)
Individual nodes

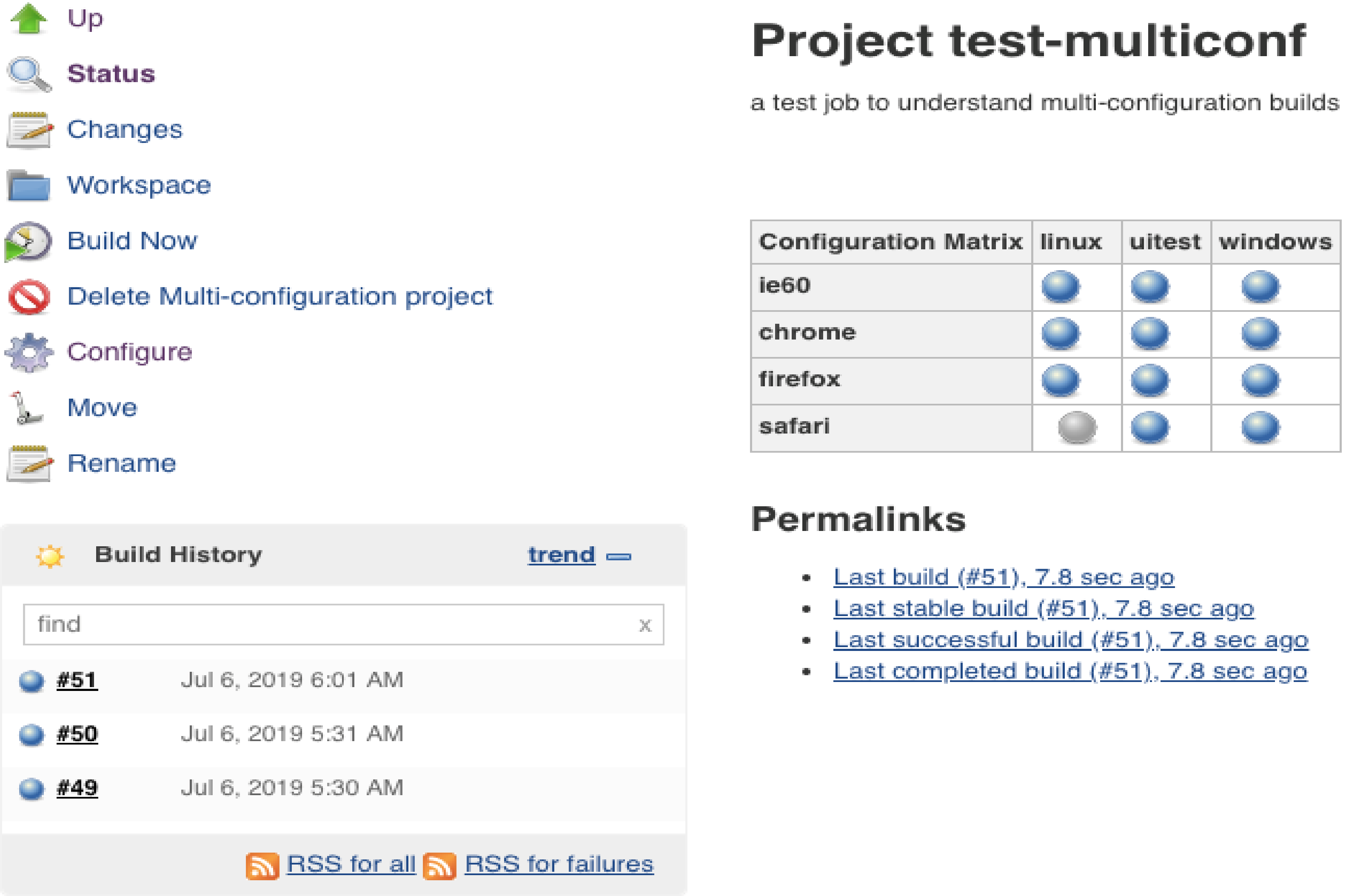
☑ Combination Filter

Filter: !(label == "linux" && browser == "safari")

□ Run each configuration sequentially

ou ds first

Cont’d…



- Up
- Status
- Changes
- Workspace
- Build Now
- Delete Multi-configuration project
- Configure
- Move
- Rename

Build History trend

find x

#51	Jul 6, 2019 6:01 AM
#50	Jul 6, 2019 5:31 AM
#49	Jul 6, 2019 5:30 AM

RSS for all RSS for failures

Project test-multiconf

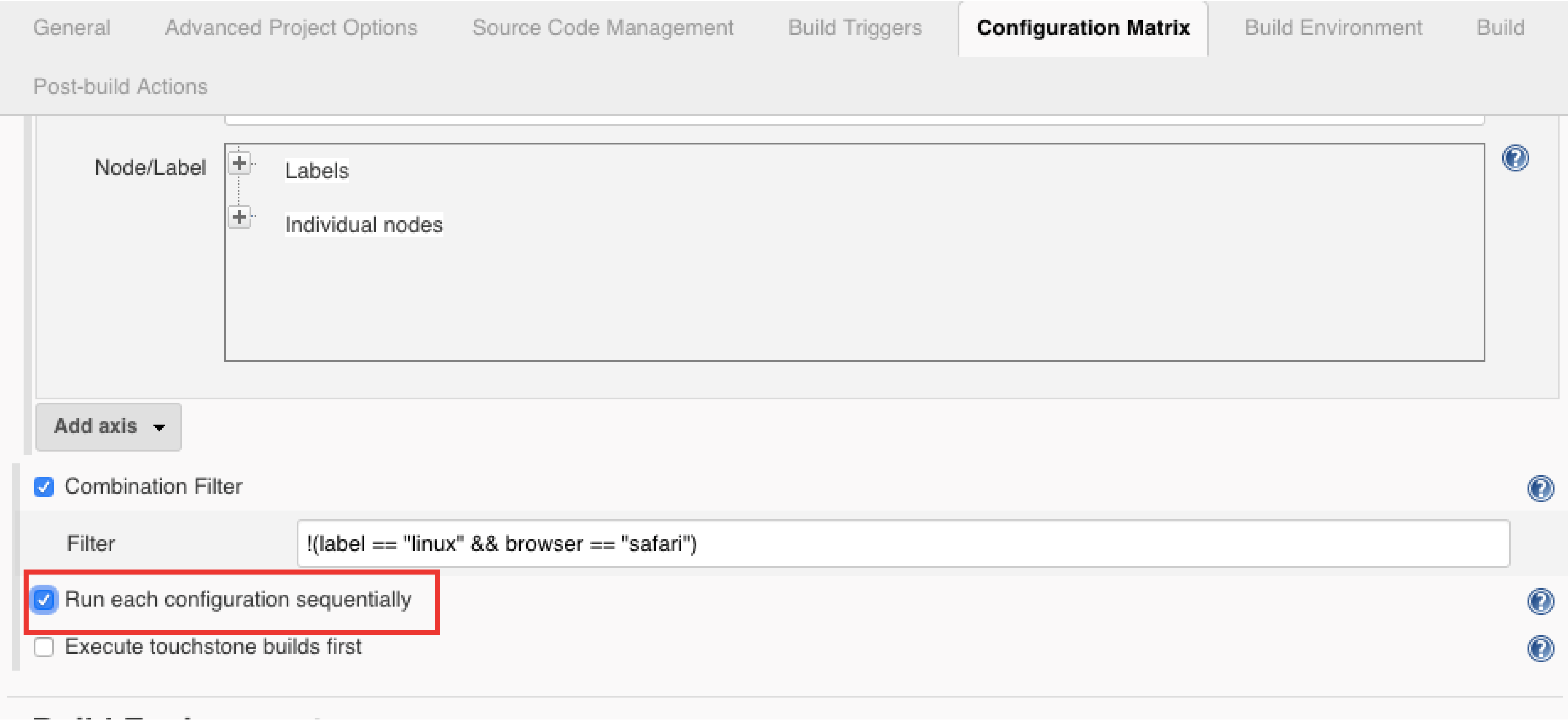
a test job to understand multi-configuration builds

Configuration Matrix	linux	uitest	windows
ie60	●	●	●
chrome	●	●	●
firefox	●	●	●
safari	○	●	●

Permalinks

- Last build (#51), 7.8 sec ago
- Last stable build (#51), 7.8 sec ago
- Last successful build (#51), 7.8 sec ago
- Last completed build (#51), 7.8 sec ago

Cont'd…



Node/Label

☑ Combination Filter

Filter: !(label == "linux" && browser == "safari")

☑ Run each configuration sequentially

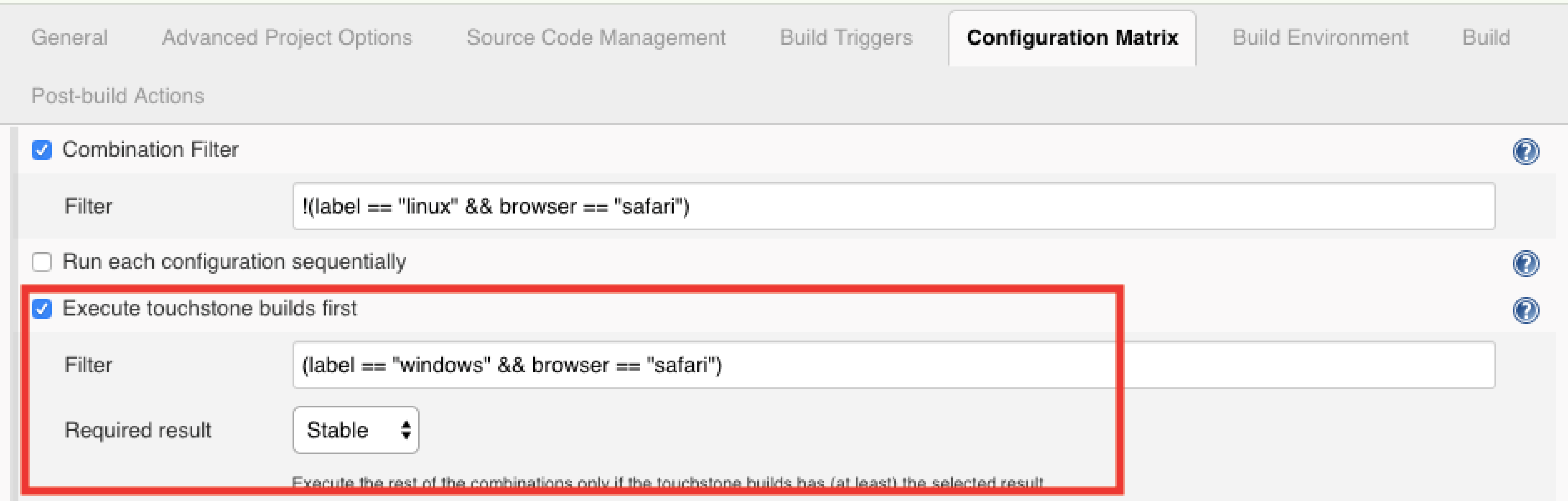
□ Execute touchstone builds first

Note:

- By default the jobs created by multi-configuration jobs are executed in parallel
- To execute sequentially, select run each configuration sequentially as shown

To execute a particular combination of build first

- Select Execute touchstone builds first
- Add value in filter, combination should run first example as shown below
- Specify the mentioned build status should also be stable so that the next jobs will trigger



☑ Combination Filter

Filter: !(label == "linux" && browser == "safari")

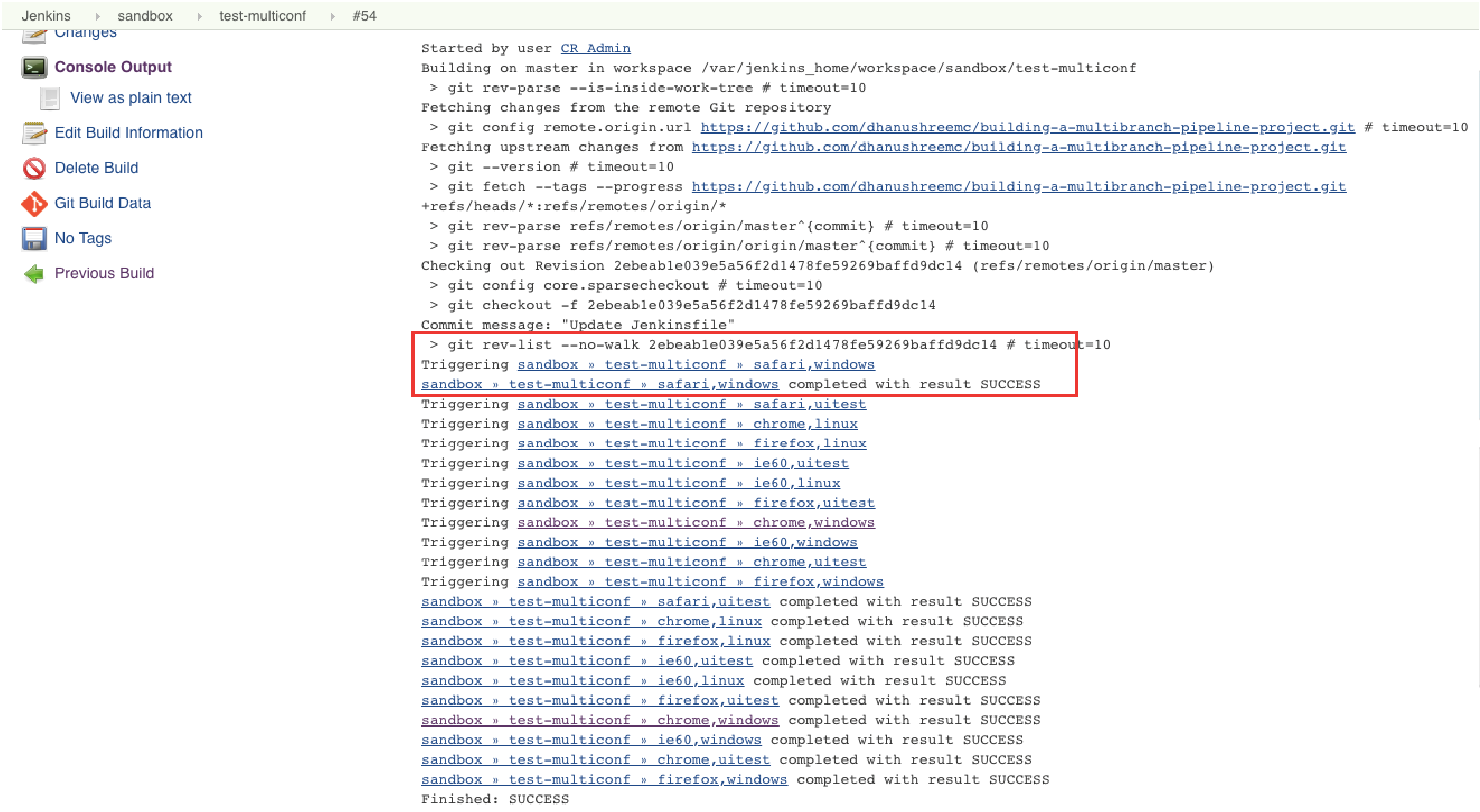
□ Run each configuration sequentially

☑ Execute touchstone builds first

Filter: (label == "windows" && browser == "safari")

Required result: Stable

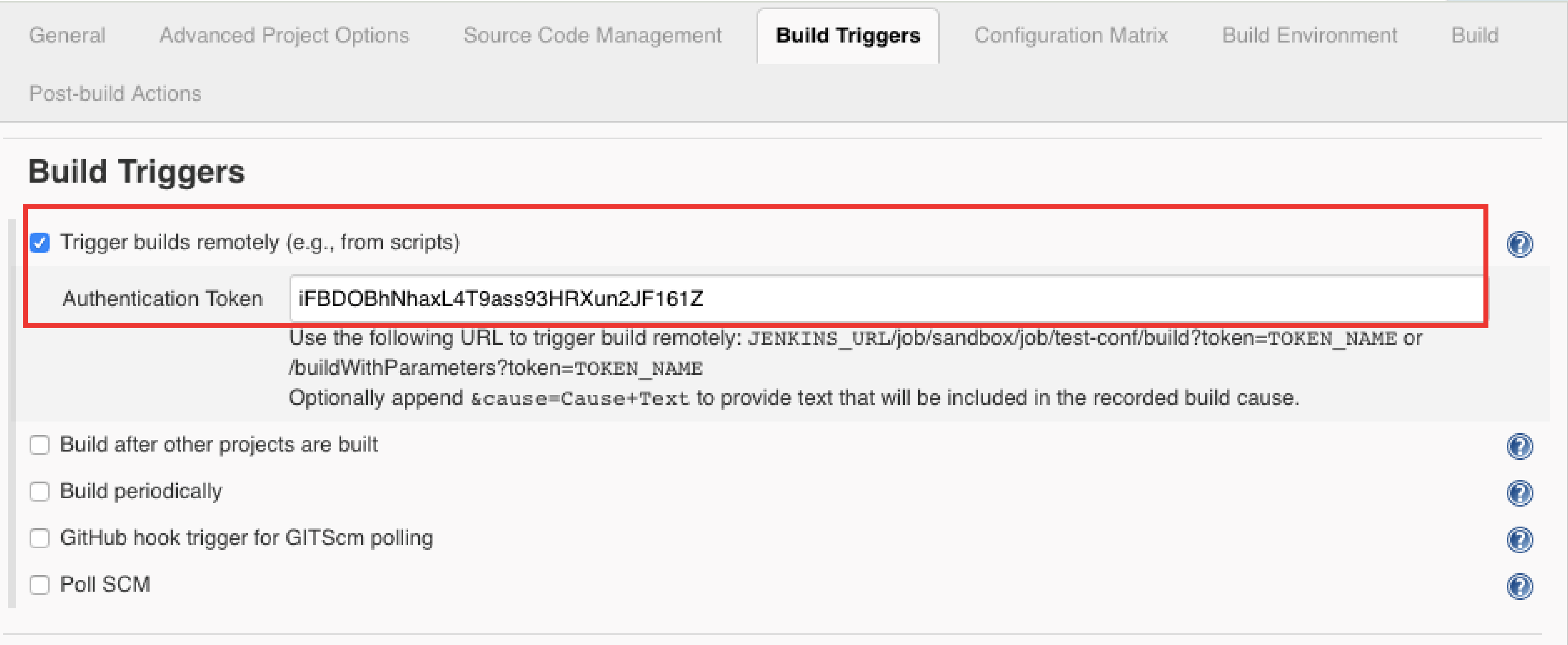
Execute the rest of the combinations only if the touchstone builds has (at least) the selected result



```
Started by user CR Admin
Building on master in workspace /var/jenkins_home/workspace/sandbox/test-multiconf
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/dhanushreemc/building-a-multibranch-pipeline-project.git # timeout=10
Fetching upstream changes from https://github.com/dhanushreemc/building-a-multibranch-pipeline-project.git
 > git --version # timeout=10
 > git fetch --tags --progress https://github.com/dhanushreemc/building-a-multibranch-pipeline-project.git
+refs/heads/*:refs/remotes/origin/*
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 2ebeable039e5a56f2d1478fe59269baffd9dc14 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 2ebeable039e5a56f2d1478fe59269baffd9dc14
Commit message: "Update Jenkinsfile"
 > git rev-list --no-walk 2ebeable039e5a56f2d1478fe59269baffd9dc14 # timeout=10
Triggering sandbox » test-multiconf » safari,windows
sandbox » test-multiconf » safari,windows completed with result SUCCESS
Triggering sandbox » test-multiconf » safari,uitest
Triggering sandbox » test-multiconf » chrome,linux
Triggering sandbox » test-multiconf » firefox,linux
Triggering sandbox » test-multiconf » ie60,uitest
Triggering sandbox » test-multiconf » ie60,linux
Triggering sandbox » test-multiconf » firefox,uitest
Triggering sandbox » test-multiconf » chrome,windows
Triggering sandbox » test-multiconf » ie60,windows
Triggering sandbox » test-multiconf » chrome,uitest
Triggering sandbox » test-multiconf » firefox,windows
sandbox » test-multiconf » safari,uitest completed with result SUCCESS
sandbox » test-multiconf » chrome,linux completed with result SUCCESS
sandbox » test-multiconf » firefox,linux completed with result SUCCESS
sandbox » test-multiconf » ie60,uitest completed with result SUCCESS
sandbox » test-multiconf » ie60,linux completed with result SUCCESS
sandbox » test-multiconf » firefox,uitest completed with result SUCCESS
sandbox » test-multiconf » chrome,windows completed with result SUCCESS
sandbox » test-multiconf » ie60,windows completed with result SUCCESS
sandbox » test-multiconf » chrome,uitest completed with result SUCCESS
sandbox » test-multiconf » firefox,windows completed with result SUCCESS
Finished: SUCCESS
```

To trigger a build remotely

- Go to job configurations-> Build Triggers->select Trigger builds remotely
- Enter authentication token- a randomly generated key
- Ex: iFBDOBhNhaxL4T9ass93HRXun2JF161Z as like below screenshot



Build Triggers

☑ Trigger builds remotely (e.g., from scripts)

Authentication Token: iFBDOBhNhaxL4T9ass93HRXun2JF161Z

Use the following URL to trigger build remotely: `JENKINS_URL`/job/sandbox/job/test-conf/build?token=`TOKEN_NAME` or /buildWithParameters?token=`TOKEN_NAME`
Optionally append `&cause=Cause+Text` to provide text that will be included in the recorded build cause.

☐ Build after other projects are built

☐ Build periodically

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

Cont'd…

- Go to another tab and enter Jenkins URL with job name
- This trigger builds remotely
  - url: JENKINS_URL/job/sandbox/job/testconf/build?token=iFBDOBhNhaxL4T9ass93HRXun2JF161Z
- Enter the Jenkins credentials & now Jenkins will build the job
- To Trigger Jenkins job remotely, from a script we need to create an API Token for the username and use that API Token in the URL as specified below
  - http://username:<api_token>@<jenkins_url>/job/jobname/build?token= iFBDOBhNhaxL4T9ass93HRXun2JF161Z

Jenkins API Token

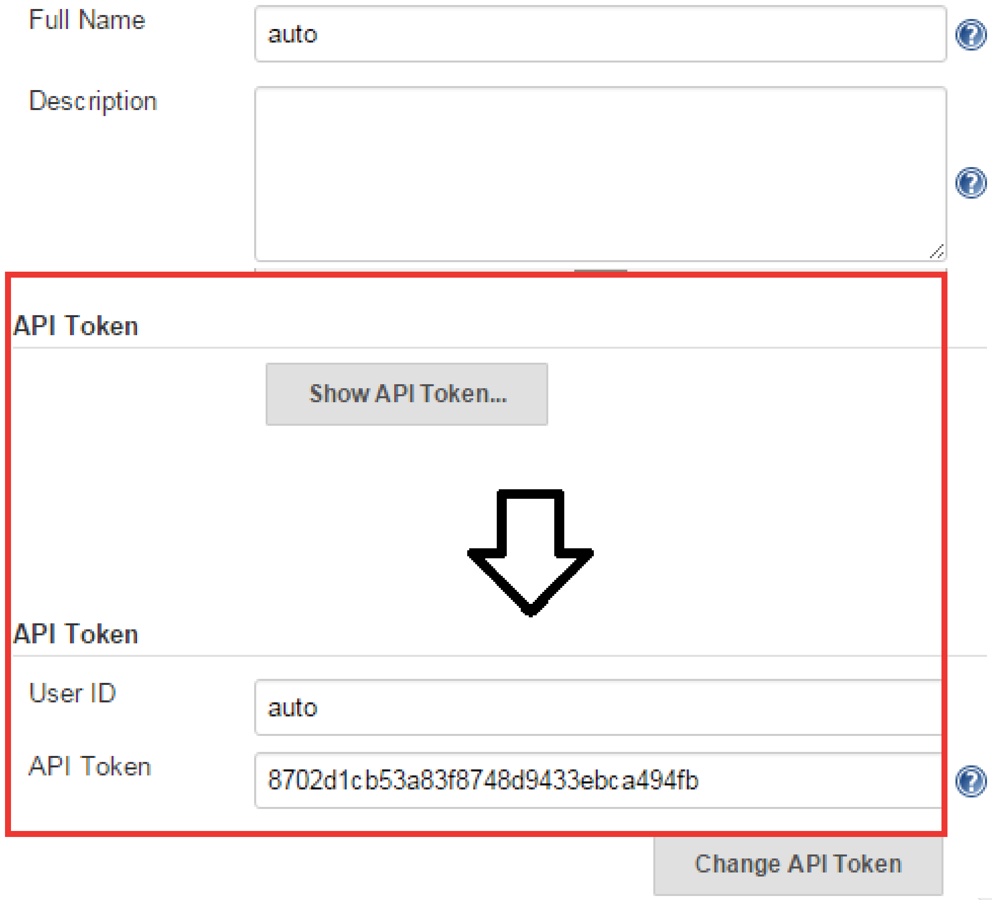
- Jenkins generated code
- Allow you to use HTTP BASIC authentication
- In order to make operations using CLI or REST calls to the Jenkins API

Steps to create API Token

Step 1:

- Go to Jenkins dashboard->Manage Jenkins->Manage Users->Create User
  - Enter Username –auto
  - Enter Password
  - Full name
  - Email address
- Then click on Create User
- Click on the use you created(auto) and click on configure
- Click on show API Token Copy API token

- Now the URL looks like below:
- http://auto:8702d1cb53a83f8748d9433ebca494fb@your-jenkins.com/job/JobName/build?token=iFBDOBhNhaxL4T9ass93HRXun2JF161Z
- Using the above URL now we can build the job through a script



Full Name: auto

Description:

API Token

API Token

User ID: auto

API Token: 8702d1cb53a83f8748d9433ebca494fb