

## 1- Install role based authorization plugin.

The screenshot shows the Jenkins Plugin Manager interface. The top navigation bar includes links for Back to Dashboard, Manage Jenkins, and a search bar. Below the navigation is a toolbar with tabs: Updates, Available, Installed (which is selected), and Advanced. A search input field is also present. The main content area displays a table of installed plugins:

Enabled	Name	Version	Previously Installed version	Uninstall
<input checked="" type="checkbox"/>	Caffeine API Plugin Caffeine api plugin for use by other Jenkins plugins.	2.9.2- 29.v717aac953ff3		<button>Uninstall</button>
<input checked="" type="checkbox"/>	Matrix Authorization Strategy Plugin Offers matrix-based security authorization strategies (global and per-project).	2.6.8		<button>Uninstall</button>
<input checked="" type="checkbox"/>	Role-based Authorization Strategy Enables user authorization using a Role-Based strategy. Roles can be defined globally or for particular jobs or nodes selected by regular expressions.	3.2.0		<button>Uninstall</button>

## 2 - Create new user & create read role and assign it to that user.

The screenshot shows the Jenkins dashboard. The top navigation bar includes links for New Item, People, Build History, and My Views. The main content area features a "Welcome to Jenkins!" message with a sub-instruction: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." Below this is a "Start building your software project" section with a "Create a Job" button. On the left, there are two collapsed sections: "Build Queue" (No builds in the queue) and "Build Executor Status" (1 Idle, 2 Idle). At the bottom right, there are links for REST API and Jenkins 2.303.3.

The screenshot shows the Jenkins "Manage and Assign Roles" page. The top navigation bar includes links for New Item, People, Build History, Manage Jenkins, My Views, Lockable Resources, and New View. The main content area has a "Manage Roles" title. It includes sections for "Global roles" and "Item roles".

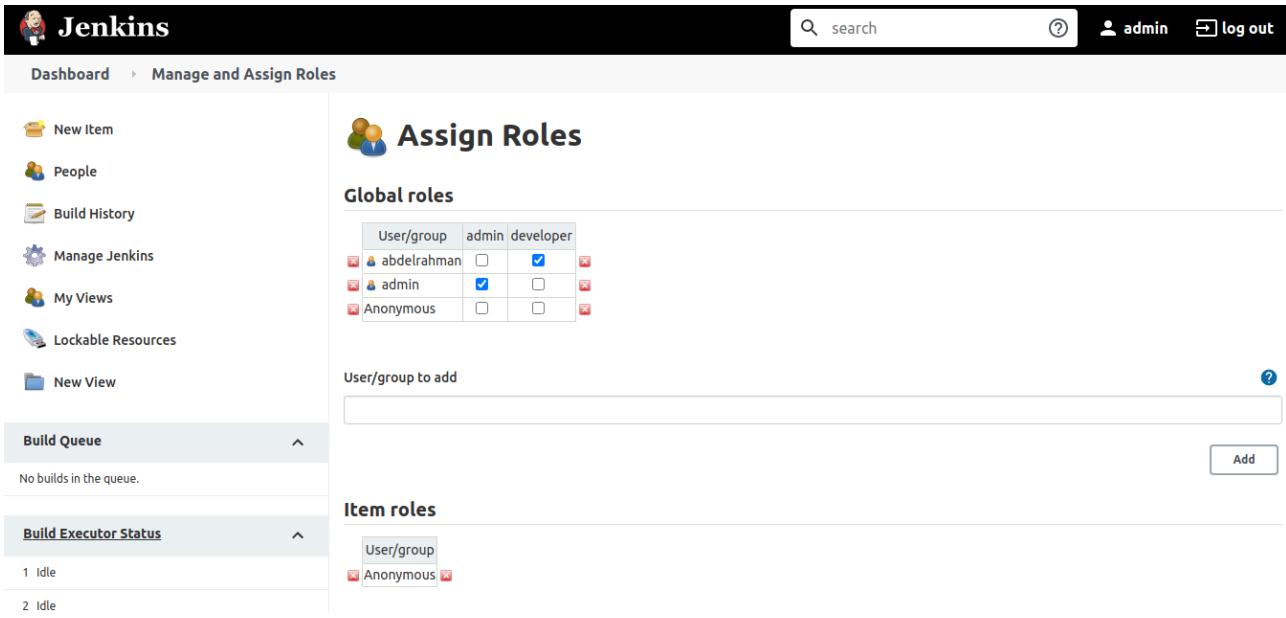
**Global roles**

Role	Overall	Credentials	Agent
admin	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
developer	<input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

**Role to add**

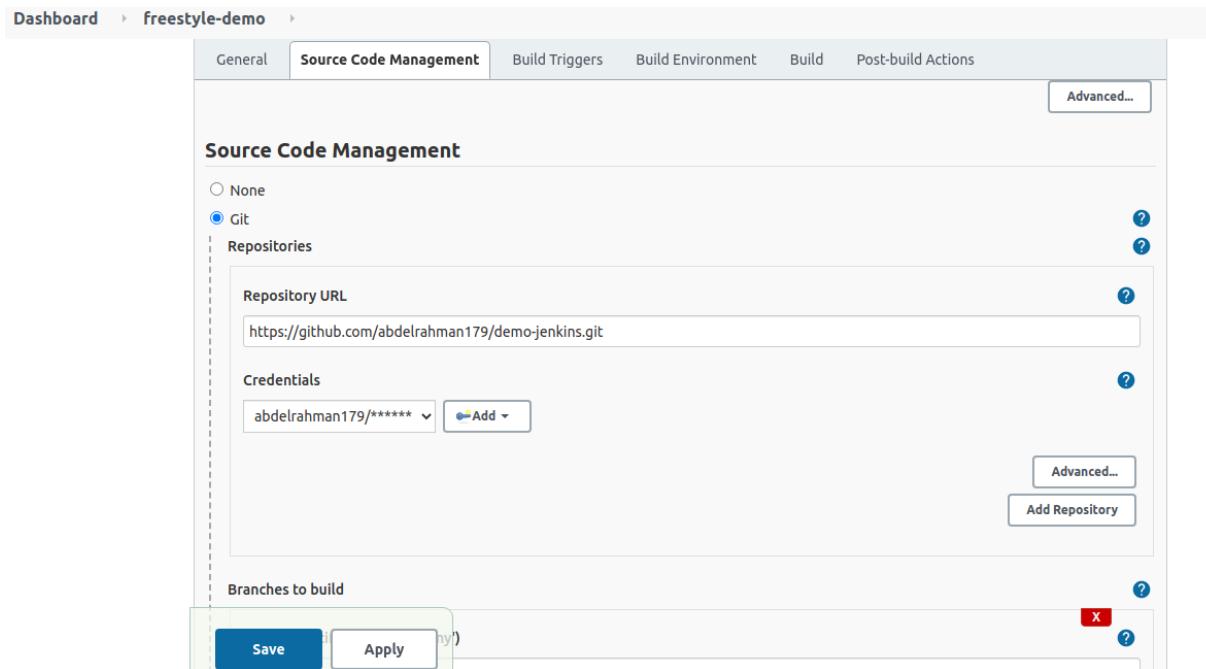
**Item roles**

Role	Pattern	Credentials	Job	Run	SCM
		Create Delete ManageDomains Update View Build Cancel Configure Create Delete Discover Move Read Workspace Delete Replay Update T...			



The screenshot shows the Jenkins 'Assign Roles' configuration page. At the top, there's a navigation bar with links for 'Dashboard', 'Manage and Assign Roles', and user information ('admin' and 'log out'). On the left sidebar, there are links for 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. Below these are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 idle). The main content area is titled 'Assign Roles' and contains two sections: 'Global roles' and 'Item roles'. In 'Global roles', a table lists users and their assigned roles: abdelrahman (admin, developer), admin (admin), and Anonymous (no roles). In 'Item roles', there's a table showing 'Anonymous' with a checked checkbox. A search bar and an 'Add' button are also present.

3- Create a free-style pipeline and link it to private git repository (Create directory and create file with "hello world" inside)



The screenshot shows the 'Source Code Management' configuration for a Jenkins job named 'freestyle-demo'. The 'Source Code Management' tab is selected. The 'General' tab is visible at the top. The 'Source Code Management' section includes tabs for 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. An 'Advanced...' button is located in the top right of the main configuration area. The 'Source Code Management' section has a 'None' radio button and a selected 'Git' radio button. Under 'Git', there's a 'Repositories' section with a 'Repository URL' input field containing 'https://github.com/abdelrahman179/demo-jenkins.git' and a 'Credentials' dropdown currently set to 'abdelrahman179\*\*\*\*\*'. Buttons for 'Advanced...', 'Save', and 'Apply' are at the bottom. A 'Branches to build' section is partially visible below, with a 'Save' button and a 'Cancel' button.

Dashboard > freestyle-demo >

General	Source Code Management	Build Triggers	Build Environment	Build	Post-build Actions
<input type="checkbox"/> Use secret text(s) or file(s)					
<input type="checkbox"/> Abort the build if it's stuck					
<input type="checkbox"/> Add timestamps to the Console Output					
<input type="checkbox"/> Inspect build log for published Gradle build scans					
<input type="checkbox"/> With Ant					

### Build

**Execute shell**

**Command**

```
ls
mkdir sp_bootcamp
cd sp_bootcamp
touch demo
echo "Hello World !!" >> demo
```

See the [list of available environment variables](#)

[Advanced...](#)

[Add build step ▾](#)

### Post-build Actions

**Save** **Apply**

Dashboard > freestyle-demo > #8

[Back to Project](#)

[Status](#)

[Changes](#)

**Console Output**

[View as plain text](#)

[Edit Build Information](#)

[Delete build '#8'](#)

[Git Build Data](#)

[Previous Build](#)

### Console Output

Started by user **admin**  
Running as **SYSTEM**  
Building in workspace /var/jenkins\_home/workspace/freestyle-demo  
[WS-CLEANUP] Deleting project workspace...  
[WS-CLEANUP] Deferred wipeout is used...  
[WS-CLEANUP] Done

The recommended git tool is: NONE  
using credential **github**  
Cloning the remote Git repository  
Cloning repository <https://github.com/abdelrahman179/demo-jenkins.git>  
> git init /var/jenkins\_home/workspace/freestyle-demo # timeout=10  
Fetching upstream changes from <https://github.com/abdelrahman179/demo-jenkins.git>  
> git --version # timeout=10  
> git --version # 'git' version 2.30.2'  
using GIT\_ASKPASS to set credentials  
> git fetch --tags --force --progress -- <https://github.com/abdelrahman179/demo-jenkins.git>  
+refs/heads/\*:refs/remotes/origin/\* # timeout=10  
> git config remote.origin.url <https://github.com/abdelrahman179/demo-jenkins.git> # timeout=10  
> git config --add remote.origin.fetch +refs/heads/\*:refs/remotes/origin/\* # timeout=10  
Avoid second fetch  
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10  
Checking out Revision 333f73cb204bb839291ba647ec926393451686fa (refs/remotes/origin/main)  
> git config core.sparsecheckout # timeout=10  
> git checkout -f 333f73cb204bb839291ba647ec926393451686fa # timeout=10  
Commit message: "Initial commit"  
> git rev-list --no-walk 333f73cb204bb839291ba647ec926393451686fa # timeout=10  
[freestyle-demo] \$ /bin/sh -xe /tmp/jenkins8827553752418027954.sh  
+ ls  
README.md  
+ mkdir sp\_bootcamp

**Jenkins**

Dashboard > [New Item](#) [People](#) [Build History](#) [Project Relationship](#) [Check File Fingerprint](#) [Manage Jenkins](#) [My Views](#) [Lockable Resources](#) [New View](#)

**Build Queue**  
No builds in the queue.

**Build Executor Status**

All	W	Name	Last Success	Last Failure	Last Duration
		Freestyle-demo	5 min 5 sec - #8	15 min - #6	1.1 sec

Icon: S M L

Legend: Atom Feed for all Atom Feed for Failures Atom Feed for Just latest builds

Dashboard > freestyle-demo > #8

[View as plain text](#)

[Edit Build Information](#)

[Delete build '#8'](#)

[Git Build Data](#)

[Previous Build](#)

```

The recommended git tool is: NONE
using credential github
Cloning the remote Git repository
Cloning repository https://github.com/abdelrahman179/demo-jenkins.git
> git init /var/jenkins_home/workspace/freestyle-demo # timeout=10
Fetching upstream changes from https://github.com/abdelrahman179/demo-jenkins.git
> git --version # timeout=10
> git --version # 'git version 2.30.2'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/abdelrahman179/demo-jenkins.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/abdelrahman179/demo-jenkins.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 333f73cb204bb839291ba647ec926393451686fa (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 333f73cb204bb839291ba647ec926393451686fa # timeout=10
Commit message: "Initial commit"
> git rev-list --no-walk 333f73cb204bb839291ba647ec926393451686fa # timeout=10
[freestyle-demo] $ /bin/sh -xe /tmp/jenkins8827553752418027954.sh
+ ls
README.md
+ mkdir sp_bootcamp
+ cd sp_bootcamp
+ touch demo
+ echo Hello World !!
Finished: SUCCESS

```

```

logs
jenkins@47472a1a1b2a:~$ cd workspace/
jenkins@47472a1a1b2a:~/workspace$ ls
freestyle-demo  freestyle-demo@tmp
jenkins@47472a1a1b2a:~/workspace$ cd freestyle-demo
jenkins@47472a1a1b2a:~/workspace/freestyle-demo$ ls
README.md  sp_bootcamp
jenkins@47472a1a1b2a:~/workspace/freestyle-demo$ cd sp_bootcamp/
jenkins@47472a1a1b2a:~/workspace/freestyle-demo/sp_bootcamp$ ls
demo
jenkins@47472a1a1b2a:~/workspace/freestyle-demo/sp_bootcamp$ cat demo
Hello World !!
jenkins@47472a1a1b2a:~/workspace/freestyle-demo/sp_bootcamp$ 

```

#### 4- Create declarative in Jenkins GUI pipeline for your own repository to do "ls"

Dashboard > pipeline-demo >

[General](#) [Build Triggers](#) [Advanced Project Options](#) [Pipeline](#)

[SCM](#)

**Git**

**Repositories**

**Repository URL**: https://github.com/abdelrahman179/demo-jenkins.git

**Credentials**: abdelrahman179/\*\*\*\*\*\*\*\* [Add](#)

[Advanced...](#) [Add Repository](#)

**Branches to build**

**Branch Specifier (blank for 'any')**: \*/main

[Save](#) [Apply](#) [Add Branch](#)

Dashboard > pipeline-demo >

General Build Triggers Advanced Project Options Pipeline

Add Branch

Repository browser (Auto)

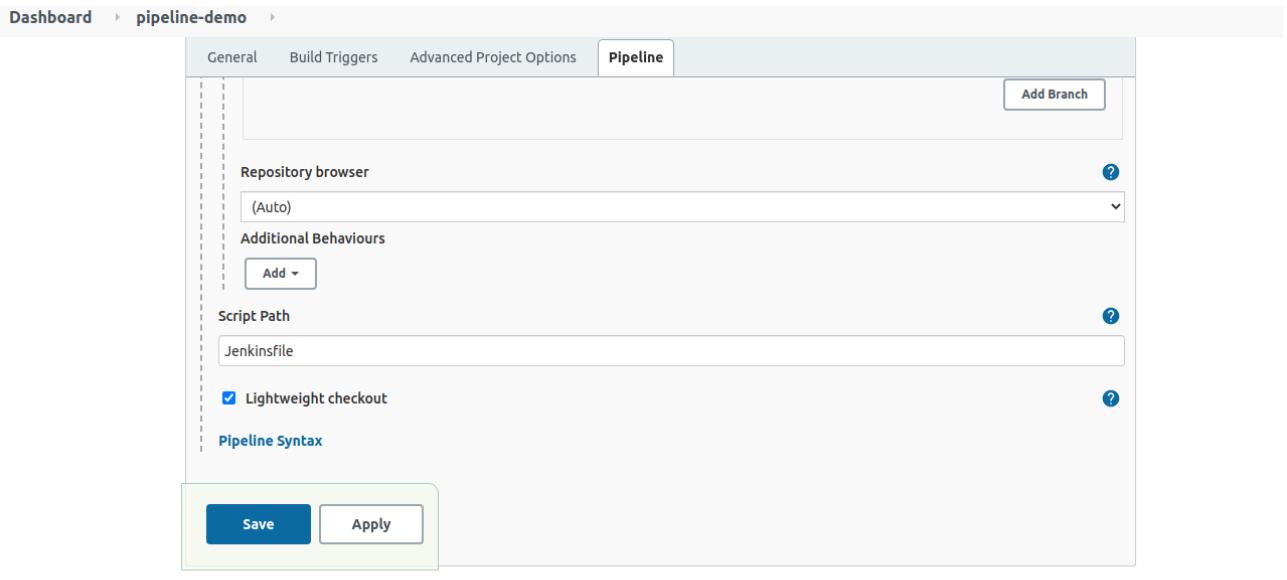
Additional Behaviours Add ▾

Script Path Jenkinsfile

Lightweight checkout

Pipeline Syntax

Save Apply



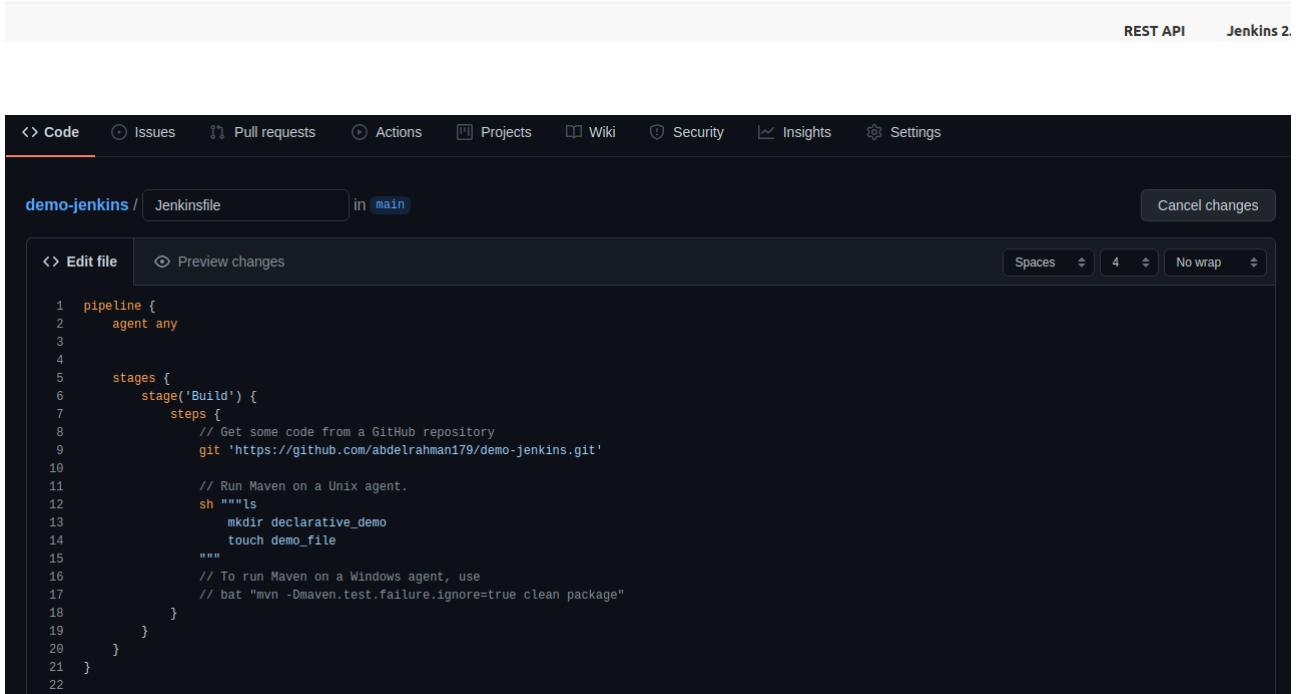
REST API Jenkins 2.

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

demo-jenkins / Jenkinsfile in main Cancel changes

<> Edit file Preview changes Spaces 4 No wrap

```
1 pipeline {
2     agent any
3
4     stages {
5         stage('Build') {
6             steps {
7                 // Get some code from a GitHub repository
8                 git 'https://github.com/abdelrahman179/demo-jenkins.git'
9
10                // Run Maven on a Unix agent.
11                sh """
12                    mkdir declarative_demo
13                    touch demo_file
14                """
15                // To run Maven on a Windows agent, use
16                // bat "mvn -Dmaven.test.failure.ignore=true clean package"
17
18            }
19        }
20    }
21 }
```



## 5- create scripted in Jenkins GUI pipeline for your own repository to do "ls"

The screenshot shows the Jenkins Pipeline configuration screen. At the top, there are tabs: General, Build Triggers, Advanced Project Options (which is selected), and Pipeline. Below the tabs is an 'Advanced...' button. The main area is titled 'Pipeline' and has a 'Definition' section. Under 'Definition', there is a dropdown set to 'Pipeline script'. A code editor displays the following Groovy script:

```
git https://github.com/jglick/simple-maven-project-with-tests.git
// Get the Maven tool.
// ** NOTE: This 'M3' Maven tool must be configured
// in the global configuration.
mvnHome = tool 'M3'

stage('Build') {
    // Run the maven build
    withEnv(["MVN_HOME=$mvnHome"]) {
        if (isUnix()) {
            sh """
                ls
                mkdir demo_scripted
                touch demo_scripted
            """
        } else {
            bat("%MVN_HOME%\bin\mvn" -Dmaven.test.failure.ignore clean package)
        }
    }
    stage('Results') {
```

Below the code editor, there is a checked checkbox labeled 'Use Groovy Sandbox'. At the bottom of the 'Definition' section are 'Save' and 'Apply' buttons. To the right of the 'Definition' section is a 'Scripted Pipeline' dropdown.

## 6- create the same with jenkinsfile in your branches as multi-branch pipeline.

The screenshot shows the Jenkins Multibranch Pipeline Log. The left sidebar contains navigation links: Up, Status, Configure, Scan Multibranch Pipeline Now, Scan Multibranch Pipeline Log (selected), View as plain text, Multibranch Pipeline Events, Delete Multibranch Pipeline, People, Build History, Project Relationship, Check File Fingerprint, Rename, and Pipeline Syntax. The main panel is titled 'Scan Multibranch Pipeline Log' and shows the log output for a scan. The log starts with 'Started [Wed Nov 10 23:05:22 UTC 2021] Starting branch indexing...' and continues through various git commands, credential handling, and branch checking. The log ends with 'Processed 2 branches'.

```
Started [Wed Nov 10 23:05:22 UTC 2021] Starting branch indexing...
> git --version # timeout=10
> git --version # 'git version 2.30.2'
using GIT_ASKPASS to set credentials
> git ls-remote --symref -- https://github.com/abdelrahman179/demo-jenkins.git # timeout=10
> git rev-parse --resolve-git-dir /var/jenkins_home/caches/git-b2513134f4ad8c1499366fb246a92152/.git # timeout=10
Setting origin to https://github.com/abdelrahman179/demo-jenkins.git
> git config remote.origin.url https://github.com/abdelrahman179/demo-jenkins.git # timeout=10
Fetching & pruning origin...
Fetching & pruning origin...
Listing remote references...
> git config --get remote.origin.url # timeout=10
> git --version # timeout=10
> git --version # 'git version 2.30.2'
using GIT_ASKPASS to set credentials
> git ls-remote -h -- https://github.com/abdelrahman179/demo-jenkins.git # timeout=10
Fetching upstream changes from origin
> git config --get remote.origin.url # timeout=10
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress --prune -- origin +refs/heads/*:refs/remotes/origin/* # timeout=10
Checking branches...
  Checking branch demo
    'Jenkinsfile' found
    Met criteria
Did not schedule build for branch: demo
  Checking branch main
    'Jenkinsfile' found
    Met criteria
Did not schedule build for branch: main
Processed 2 branches
```

Dashboard &gt;

 New Item People Build History Project Relationship Check File Fingerprint Manage Jenkins My Views Lockable Resources New ViewAll [+](#)

S

W

Name :

Last Success

Last Failure

Last Duration



freestyle-demo

53 min - #8

1 hr 3 min - #6

1.1 sec



multi-branch-demo

32 sec - log

N/A

2.7 sec

Icon: **S M L**

Legend

 Atom feed for all Atom feed for failures Atom feed for just latest builds**Build Queue**

No builds in the queue.

**Build Executor Status**