Multi-branch pipeline

Multi-branch pipeline

A multi-branch pipeline is a Jenkins feature that allows you to automatically create Jenkins pipeline jobs for each branch in your source code repository

Install Required Plugins:
Install the GitLab Plugin in Jenkins.
Install the Pipeline Plugin if not already installed.

Use of Multi-branch

Automatic Branch Management: Creates Jenkins pipeline jobs for each branch in your repository automatically.
Branch Isolation: Ensures that changes in one branch do not interfere with builds in other branches.
Branch-specific Configuration: Allows customization of build, test, and deployment processes for each branch.
Automatic Triggering: Builds are triggered automatically whenever changes are pushed to a branch.

1.Configure GitLab Plugin

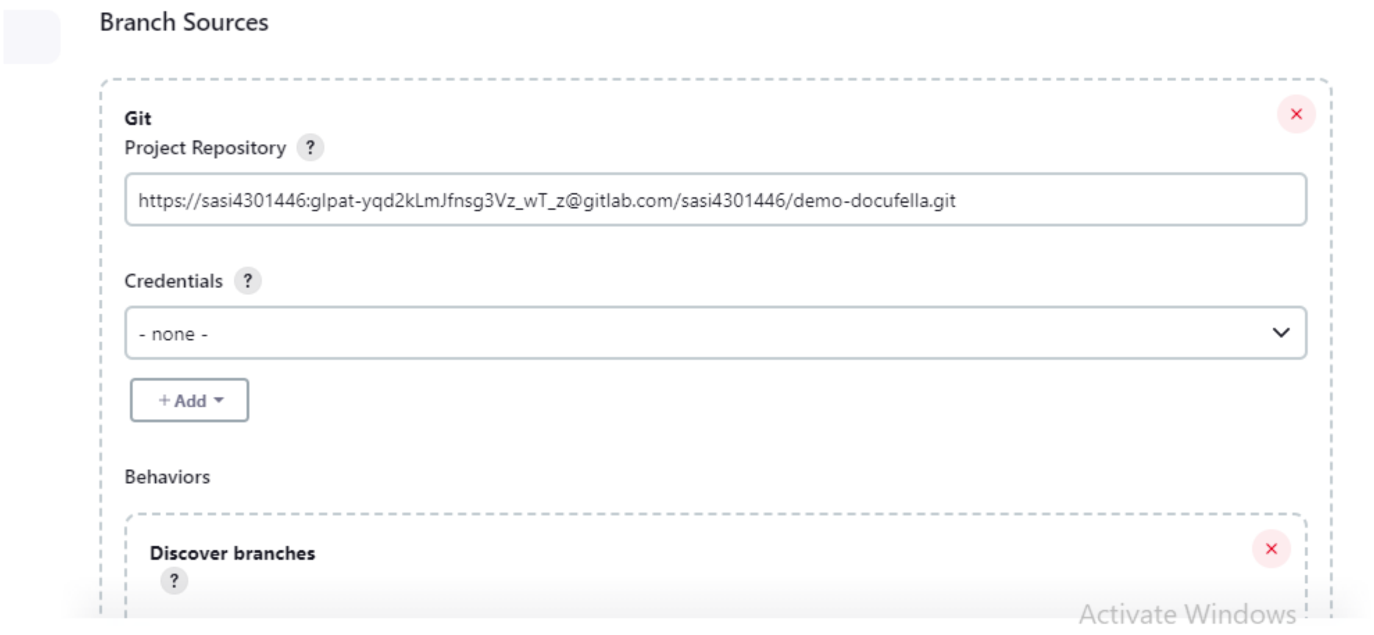
Go to Jenkins dashboard -> Manage Jenkins -> Configure System.
Scroll down to the "GitLab" section.
Add GitLab connection details (e.g., GitLab host URL, credentials).
Test the connection to ensure it's working

2.Set Up a Multi-Branch Pipeline Job in Jenkins:

Create a New Item:
Click on "New Item" in the Jenkins dashboard.
Enter a name for your job (e.g., "MyMultiBranchPipeline").
Select "Multibranch Pipeline" and click "OK".

3.Configure Branch Sources:

Under "Branch Sources", select "Git" and enter the repository URL with personal access token.
Optionally, you can specify credentials if your repository requires authentication.
Specify the branch sources, e.g., */master for the master branch.


Branch Sources

Git

Project Repository

https://sasi4301446:glpat-yqd2kLmJfnsg3Vz_wT_z@gitlab.com/sasi4301446/demo-docufella.git

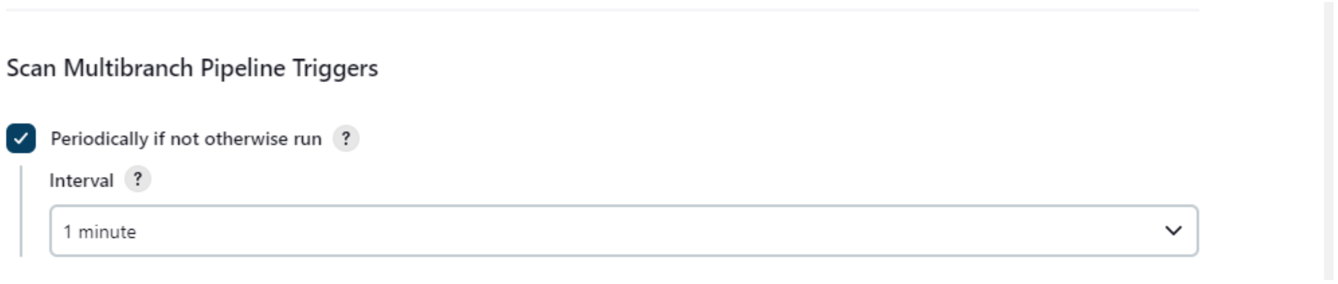
Credentials

- none -

Behaviors

Discover branches

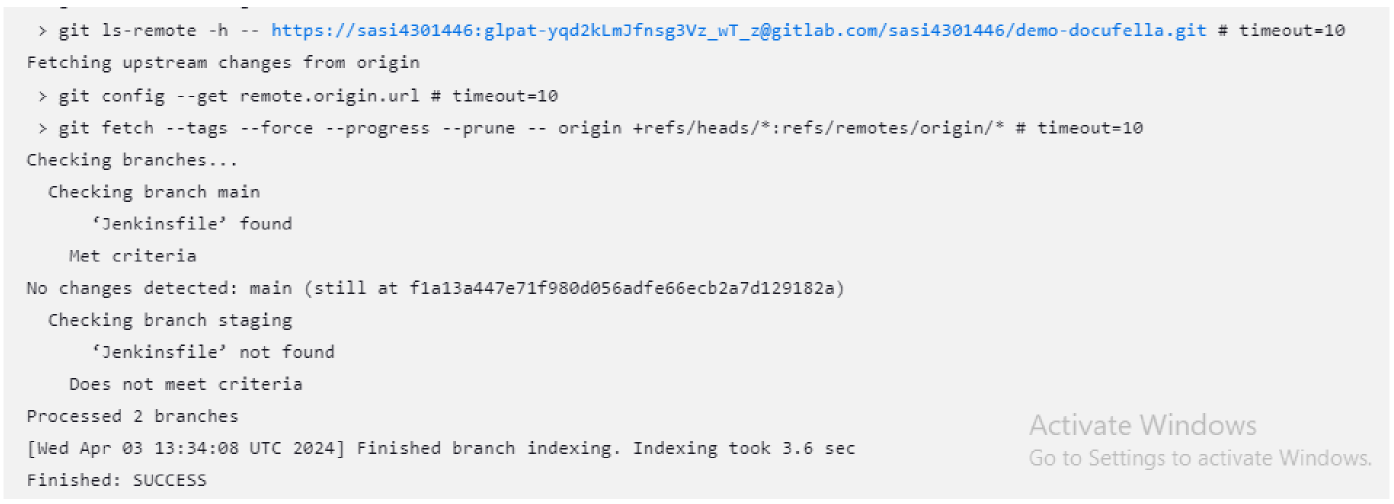
You need to set timer for Scan Multibranch pipeline trigger
Click "Save".


Scan Multibranch Pipeline Triggers

Periodically if not otherwise run: ✓

Interval: 1 minute

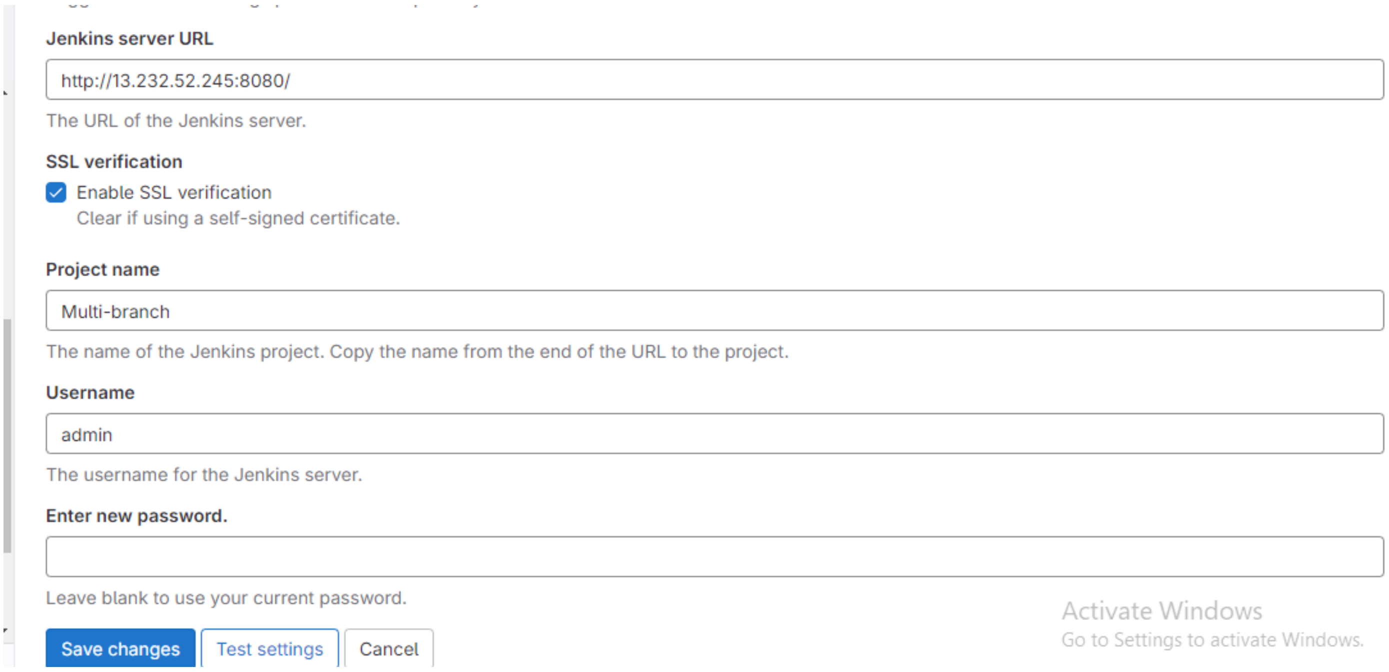
Once you save the pipeline, check the pipeline log


```
 > git ls-remote -h -- https://sasi4301446:glpat-yqd2kLmJfnsg3Vz_wT_z@gitlab.com/sasi4301446/demo-docufella.git # timeout=10
Fetching upstream changes from origin
 > git config --get remote.origin.url # timeout=10
 > git fetch --tags --force --progress --prune -- origin +refs/heads/*:refs/remotes/origin/* # timeout=10
Checking branches...
  Checking branch main
      'Jenkinsfile' found
    Met criteria
No changes detected: main (still at f1a13a447e71f980d056adfe66ecb2a7d129182a)
  Checking branch staging
      'Jenkinsfile' not found
    Does not meet criteria
Processed 2 branches
[Wed Apr 03 13:34:08 UTC 2024] Finished branch indexing. Indexing took 3.6 sec
Finished: SUCCESS
```

4. Configure Webhooks in GitLab:

Add Webhook:
Go to your GitLab project.
Navigate to Settings -> Integrations.
Add a new webhook with Jenkins URL , project name, username and password

Select the events you want to trigger the webhook (e.g., Push events, Merge requests events).



Jenkins server URL

http://13.232.52.245:8080/

The URL of the Jenkins server.

SSL verification

☑ Enable SSL verification
Clear if using a self-signed certificate.

Project name

Multi-branch

The name of the Jenkins project. Copy the name from the end of the URL to the project.

Username

admin

The username for the Jenkins server.

Enter new password.

Leave blank to use your current password.

5.Create a Jenkinsfile

Jenkinsfile in Repository:
Create a Jenkinsfile in the root directory of your GitLab repository.
Define the pipeline stages, including checkout, build, test, deploy, etc.

For example:
This script will go to directory and pull the changes

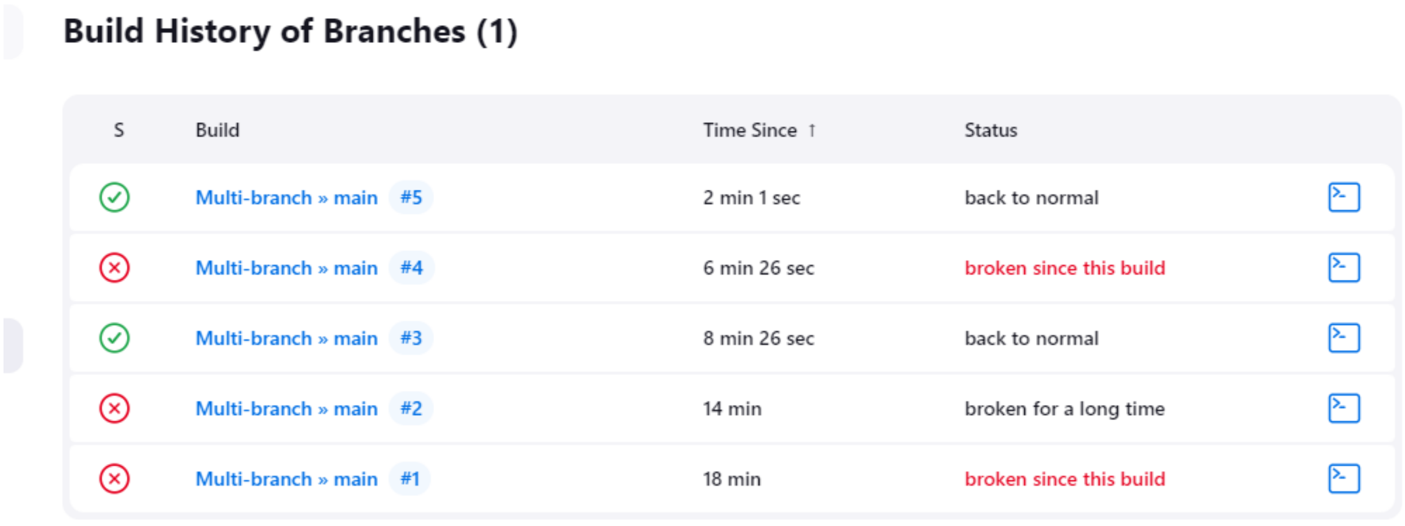
```
pipeline {
    agent any

    stages {
        stage('Checkout') {
            steps {
                script {
                    checkout scm
                }
            }
        }
        stage('Build') {
            steps {
                sh 'cd /mnt/demo-docufella'
            }
        }
        stage('Test') {
            steps {
                sh 'git status'
            }
        }
        stage('Deploy') {
            steps {
                sh 'git pull origin main'
            }
        }
    }
}
```

6. Run the Pipeline

Once everything is set up, Jenkins will automatically detect branches in your GitLab repository.
It will create Jenkins jobs for each branch defined in the Jenkinsfile.
Changes pushed to the repository will trigger Jenkins to run the pipeline for the respective branches.

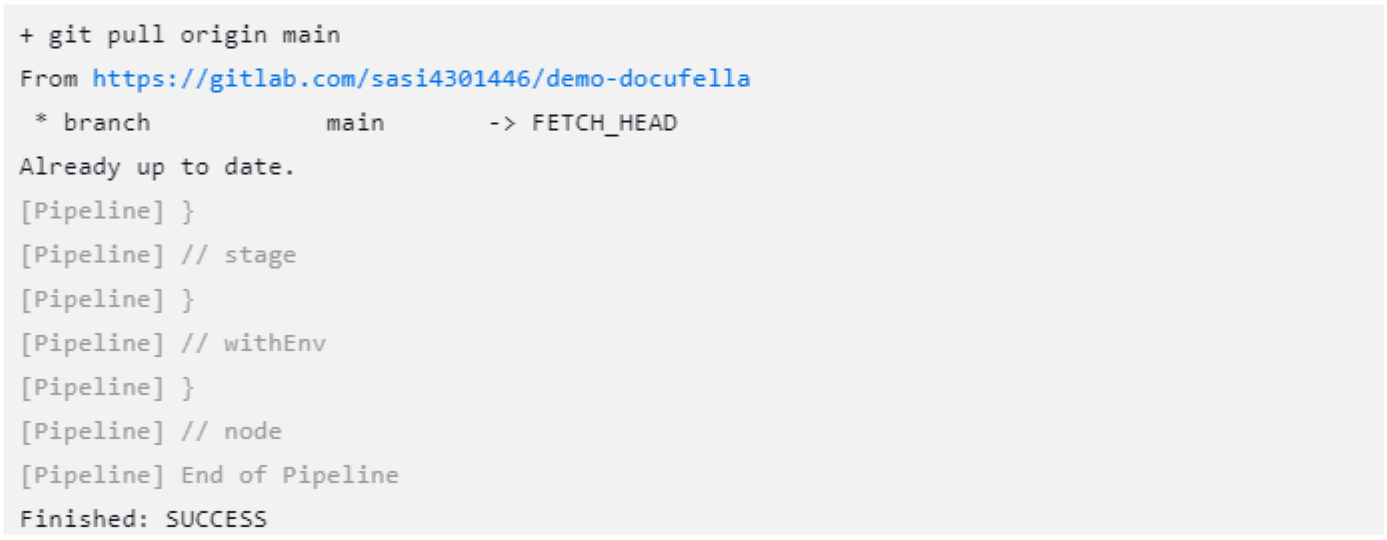
Check the build history



Build History of Branches (1)

S	Build	Time Since ↑	Status
✓	Multi-branch » main #5	2 min 1 sec	back to normal
✗	Multi-branch » main #4	6 min 26 sec	broken since this build
✓	Multi-branch » main #3	8 min 26 sec	back to normal
✗	Multi-branch » main #2	14 min	broken for a long time
✗	Multi-branch » main #1	18 min	broken since this build

Check the pipeline script log for success build



```
+ git pull origin main
From https://gitlab.com/sasi4301446/demo-docufella
 * branch            main       -> FETCH_HEAD
Already up to date.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Revision #2

Created Wed, Apr 3, 2024 1:32 PM by sasi

Updated Thu, Apr 4, 2024 5:03 AM by sasi