GitLab Jenkins Integration

Configure GitLab API Access:

Create a personal access token to authorize Jenkins' access to GitLab.

Step 1: Sign in to GitLab as the same user to be used with Jenkins.

Step 2: In the top-right corner, select your avatar.

Step 3: Select the "***Edit profile***" option

Step 4: On the left navigation pane, select the "***Access Tokens***" option.

Step 5: Check the "***API Scope***" checkbox and create a personal access token.

Step 6: Record the personal access token's value, because it's required in configuring the Jenkins server.

Jenkins-gitlab: oh7SgX2LiWuzk8YsiDfh

jen-git: glpat-es2Srm1jR8GsqtCyQeE5

webhook : glpat-7hzw-srY5ojT_UrREStM

4437b80789e0d2c368688c9883a721fc

Configure the Jenkins Server

Install and configure the required Jenkins plugins. The plugin must be installed and configured to authorize the connection to GitLab.

Step 1: On the Jenkins server, Select the "***Manage Jenkins"*** option in the left navigation pane. Then click the ***"Manage Plugins***" option.

Step 2: Install the **Jenkins GitLab Plugin** and **Jenkins Git Plugin**

Step 3: Now select the ***"Manage Jenkins"*** option. Then select the ***"Configure System"*** option.

Step 4: In the "***GitLab***" section, check the "***Enable authentication for '/project' end-point***" checkbox.

Step 5: Under the above checkbox, provide the Connection name.

Step 6: Enter the GitLab server's URL in the "***GitLab host URL***" field.

Step 7: Click the "***Add***" button, then choose "**Jenkins Credential Provider**".

Step 8: Choose the "***GitLab API token***" as the token type.

Step 9: Enter the GitLab personal access token's value in the "***API Token***" field and click the "***Add***" button.

Step 10: Click on the "***Test Connection***" button.

Step 11: On receiving a success message, you can be ensured that the connection is successful.

Step 12: Click the "***Save***" button

Configure the Jenkins project

Before implementing GitLab Jenkins Integration, you need to set up the Jenkins project where you want to execute your build.

Step 1: **Go to the Jenkins Dashboard.**

Step 2: In the left navigation pane, select the "***New item***" option.

Step 3: Assign a project's name according to your choice.

Step 4: Choose between "***Freestyle***" or "***Pipeline***" projects. Then click on "***OK***". (*Because the Jenkins plugin changes the build status on GitLab, it is recommended to select a Freestyle project. You must configure a script in a Pipeline project to update the status on GitLab.*)

Step 5: Write the description of your project as per your requirements.

Step 6: Choose your GitLab connection from the drop down.

Step 7: Scroll down and in the "***Source Code Management***" section, check the "Git" checkbox.

Step 8: Choose the repository branch if you have any.

Step 9: Copy the repository link of your project from GitLab and paste it in the Repository URL box in the Repository section under the "***Git***" checkbox.

Step 10: Scroll down to the "***Build Triggers***" section.

Step 11: Check the "***Build when a change is pushed to GitLab***" checkbox.

Step 12: Check the following checkboxes:

Accepted Merge Request Events

Closed Merge Request Events

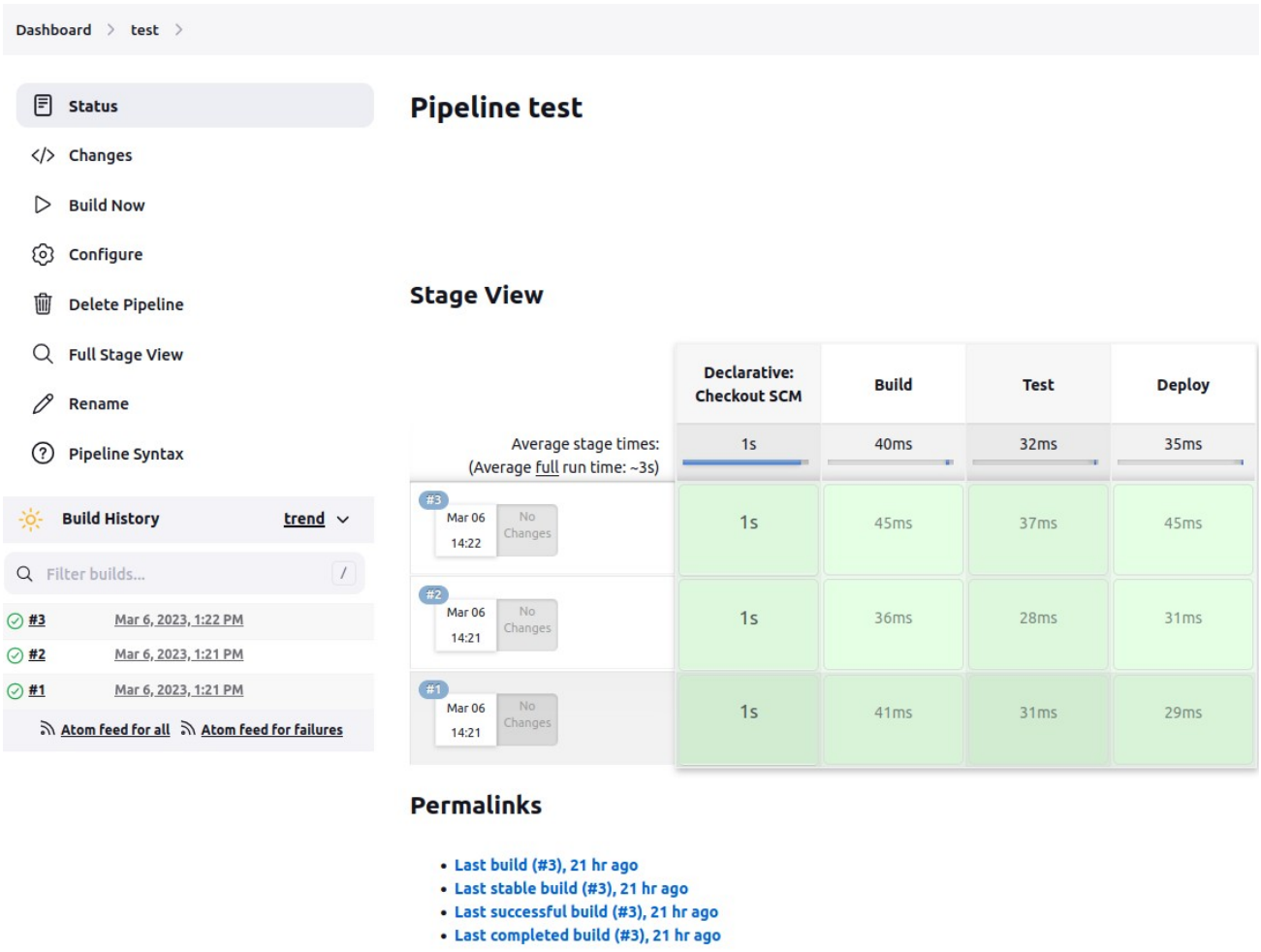
Step 13: Specify how to build status is reported to GitLab: If you created a **Freestyle** project, in the "***Post-build Actions***" section, choose "***Publish build status to GitLab***".

Integration Jenkins & Gitlab with Jenkinsfile contain stage : checkout SCM :

configure the jenkins project :

- Now in Jenkins create a new Job and in the section of *New Item* and select the same configuration for pipeline.

- Under the **Pipeline** section, select **Pipeline script from SCM** from the **Definition**
- select **Git** from the **SCM**
- Enter the repository URL (same as what you use for git clone
- Next to **Credentials**, click **Add** → **Jenkins** and fill in your GitLab username/password and click **Add**
- Select your credential from the **Credentials** If it doesn't, ensure you have the right username/password and that the user has access rights to the repository.
- Select build now



Jenkinsfile in brach main/test1:

```
pipeline {
    agent any

   stages {
      stage('Build') {
         steps {
            echo 'Building..'
         }
      }
      stage('Test') {
         steps {
            echo 'Testing..'
         }
      }
      stage('Deploy') {
         steps {
            echo 'Deploying....'
         }
      }
   }
}
```

- This is a Jenkins Pipeline script written in Groovy language.
- The script defines a simple Pipeline with three stages: "Build", "Test", and "Deploy". The agent directive is used to specify that the Pipeline can be run on any available agent.
- In the "Build" stage, the echo step is used to print the message "Building.." to the console output.

- In the "Test" stage, the echo step is used to print the message "Testing.." to the console output.
- In the "Deploy" stage, the echo step is used to print the message "Deploying...." to the console output.