Table of Contents

Steps to Integrate Gitlab with Jenkins 1
Jenkins Server configuration 1
Jenkins-to-Gitlab Clone AUthentication 2

Steps to Integrate Gitlab with Jenkins

Jenkins Server configuration

Install Plugins

- Goto `Manage Jenkins` → `Manage Plguins` → `Available Tab` and install following plugins.
  - Violation Comments to GitLab
  - GitLab
  - Blue Ocean
  - Common API for Blue Ocean
  - Config API for Blue Ocean
  - Dashboard for Blue Ocean
  - Events API for Blue Ocean
  - Git Pipeline for Blue Ocean
  - JWT for Blue Ocean
  - Personalization for Blue Ocean
  - Pipeline implementation for Blue Ocean
  - REST API for Blue Ocean
  - REST Implementation for Blue Ocean
  - Web for Blue Ocean
  - Blue Ocean Pipeline Editor
  - Blue Ocean Executor Info
  - Pipeline SCM API for Blue Ocean
  - docker
  - HTML Publisher
  - Pipeline Utility Steps

Create User for Gitlab

We need to create a user in Jenkins, such that when a JOBs executed by this user, when JOB is triggered by Webhooks from Gitlab.

- Goto `Manager Jenkins` → `Manage Users` → `Create User` and add user details.

Jenkins-to-GitLab authentication

This auth configuration is only used for accessing the GitLab API for sending build status to GitLab. It is not used for cloning git repos. The credentials for cloning (usually SSH credentials) should be configured separately, in the git plugin. This plugin can be configured to send build status messages to GitLab, which show up in the GitLab Merge Request UI. To enable this functionality:

- Create a new user in GitLab
- Give this user 'Developer' permissions on each repo you want Jenkins to send build status to
- Log in or 'Impersonate' that user in GitLab, click the user's icon/avatar and choose Settings
- Click on 'Access Tokens'
- Create a token named e.g. 'jenkins' with 'api' scope; expiration is optional
- Copy the token immediately, it cannot be accessed after you leave this page
- On the Global Configuration page in Jenkins, in the GitLab configuration section, supply the GitLab host URL, e.g. http://your.gitlab.server
- Click the 'Add' button to add a credential, choose 'GitLab API token' as the kind of credential, and paste your GitLab user's API key into the 'API token' field
- Click the 'Test Connection' button; it should succeed.

Jenkins-to-Gitlab Clone AUthentication

We need to Add SSH key to enable Jenkins to clone repositories.

- Generate SSH public/private key using below command

```
$ ssh-keygen -t rsa
```

- Goto `Manage Jenkins` → `Configure Credentials` → `Credentials` → ` System` → `Global Credentials` → `Add Credentials`
- Select Kind as `ssh username with private key`
- Set the username generate in previous section.
- Set `ID`
- Select option `private key enter directly` and add key from `cat ~/.ssh/id_rsa`
- Provide pass phrase if you've added
- And click ok
- Now login to Gitlab with user created in previous section.
- On Top right corner click on User Avatar → Settings and then on 'SSH Keys'.
- Add the public key from output of `cat ~/.ssh/id_rsa/pub`

Create Project and configure project settings in Gitlab.

- Now create a sample Project with Jenkinsfile in Gitlab.

```
pipeline {
    agent any

    stages {
        stage('Hello') {

            steps {
                echo "Hello from Jenkins pipeline"
            }

        }
    }
}
```

- Add Build User to the project with role `Developer`
- Goto Jenkins Server and create new multibranch project say `k-sample-ci-project`
- In Branch Sources section select `Git`
- Provide the git clone URL of the Gitlab project. `git@localhost:k-projects/sample-ci-project.git`
- Select Credentials as Git SSH User added in previous section.
- Click Save

Adding Webhook.

- Log in as that user (this is required even if you are a Jenkins admin user), then click on the user's name in the top right corner of the page
- Click `Configure` → `API Token` → `Add new Token` then 'generate', and note/copy the User ID and API Token
- Now goto Gitlab Project → <code>settings</code> → <code>Integration</code> add the webhook <code><a href="http://localhost:8182/project/<jenkins_project_name&gt" class="bare">http://localhost:8182/project/<jenkins_project_name&gt;</code>, and api token from previous step. click the 'Test' button, and it should succeed and trigger a JOB.

To prevent this type of exploitation from happening, starting with GitLab 10.6, all Webhook requests to the current GitLab instance server address and/or in a private network will be forbidden by default. That means that all requests made to 127.0.0.1, ::1 and 0.0.0.0, as well as IPv4 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 and IPv6 site-local (ffc0::/10) addresses won't be allowed.
This behavior can be overridden by enabling the option "Allow requests to the local network from hooks and services" in the "Outbound requests" section inside the Admin area under Settings (`/admin/application_settings/network`)