

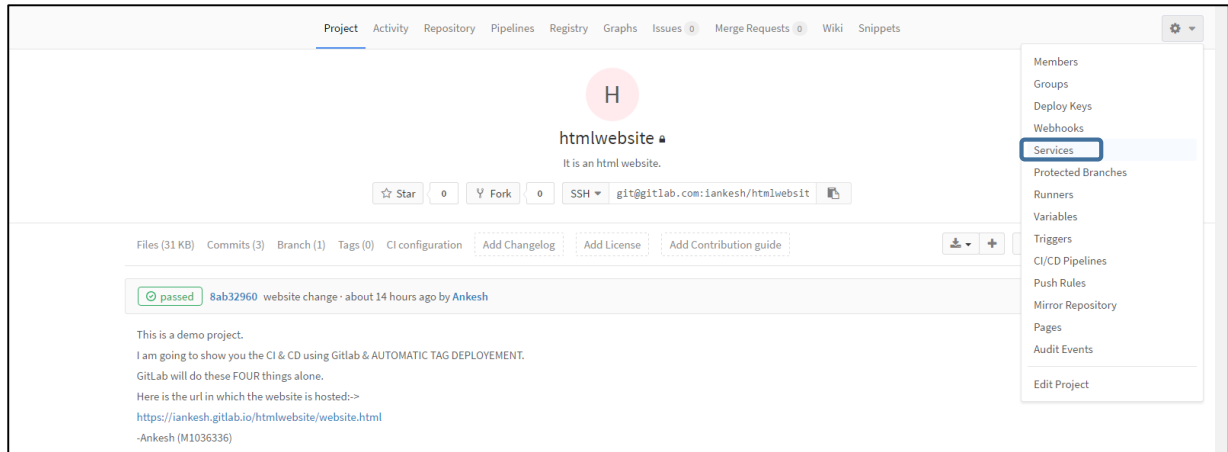
Tag based Deployment using GitLab

After knowing about the GitLab CE with CI, we have deployed a html website showing the CI & CD with GitLab. Now we will show the tag based deployment in GitLab using Jenkins.

Step 1:

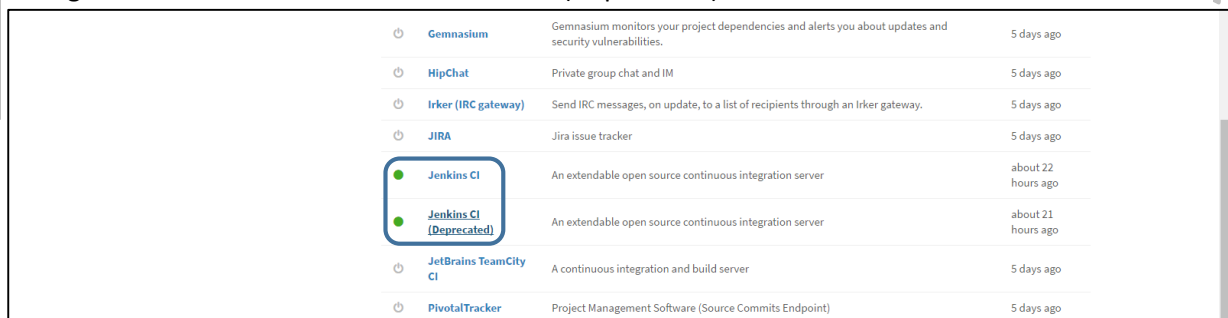
As Jenkins provides plugins to integrate with other services. Gitlab provides SERVICES.

Click on Settings > Services.



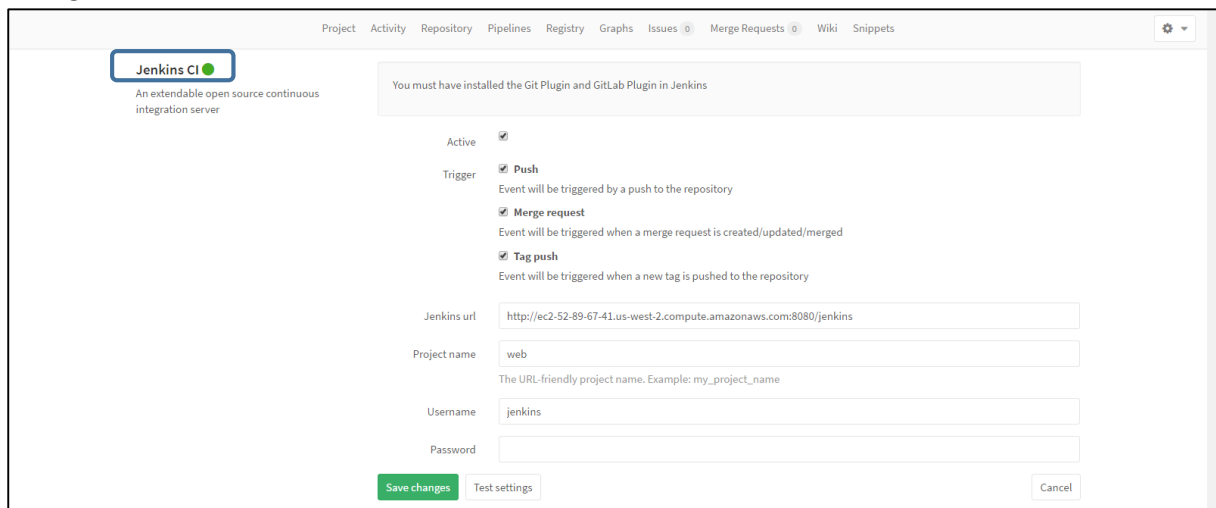
Step 2:

Configure two services: Jenkins CI & Jenkins CI (Deprecated).



Step 3:

Configure Jenkins CI as:



Step 4:

Configure Jenkins CI (Deprecated) as:

Project Activity Repository Pipelines Registry Graphs Issues Merge Requests Wiki Snippets

Jenkins CI (Deprecated)
An extendable open source continuous integration server

You must have installed GitLab Hook plugin into Jenkins. This service is deprecated. Use "Jenkins CI" service instead.

Active ☒

Trigger ☒ **Push**
Event will be triggered by a push to the repository

Project url

Multi-project setup enabled? ☒
Multi-project mode is configured in Jenkins GitLab Hook plugin.

Should unstable builds be treated as passing? ☒
Unstable builds will be treated as passing.

[Save changes](#) [Test settings](#) [Cancel](#)

Step 5:

Open your Jenkins Dashboard and create a job:

Jenkins

search

enable auto refresh

add description

S	W	Name	Last Success	Last Failure	Last Duration
		web	22 hr - #18	N/A	4 sec

Icon: S M L

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

Build Queue
No builds in the queue.

Build Executor Status
1 Idle
2 Idle

Step 6:

Go to Manage Jenkins > Configure System.

In GitLab Section, authenticate GitLab to Jenkins by assigning the API token in credentials.

GitLab

Enable authentication for '/project/' end-point ☒

GitLab connections

Connection name
A name for the connection

Gitlab host URL

Credentials **GitLab API token** [Add](#)
The complete URI to the Gitlab server (i.e. http://gitlab.org)
API Token for accessing Gitlab

[Add](#) [Advanced...](#) [Test Connection](#) [Delete](#)

Step 7:

Go inside the job and click on Configure.

Source Code Management

☐ None
☐ CVS
☐ CVS Projectset
☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

Repository browser

URL

Version

Additional Behaviours

☐ Subversion

Build Triggers

☐ Build after other projects are built

Step 8:

For tag based deployment, configure tag part in Git Publisher plugin and save it.

Post-build Actions

☒ Git Publisher

Push Only If Build Succeeds ☐

Merge Results ☐

If pre-build merging is configured, push the result back to the origin

Force Push ☐

Add force option to git push

Tags

Tag to push

Tag message

Create new tag ☒

Update new tag ☐

Target remote name

Tags to push to remote repositories

Branches to push to remote repositories

Notes to push to remote repositories

Step 9:

Now we will create a change in our local repository and push it to remote repository.
As soon as we push the changes, we can see GitLab starts building automatically.

The screenshot shows the GitLab CI/CD interface for a pipeline. At the top, it says "passed" and "Build #4467371 for commit 1c94720c from master by @iankesh less than a minute ago". The main area is a terminal window showing the build process:

```
Running with gitlab-ci-multi-runner 1.6.0 (01b3eal)
Using Shell executor...
Running on ip-172-31-16-128...
Fetching changes...
Removing public/
HEAD is now at b186580 asd
from https://gitlab.com/iankesh/web
b186580..1c94720 master -> origin/master
* [new tag] v1.0.18 -> v1.0.18
* [new tag] v111 -> v111
* [new tag] v222 -> v222
* [new tag] v333 -> v333
Checking out 1c94720c as master...
$ mkdir .public
$ cp -r * .public
$ mv .public public
Uploading artifacts...
public: found 3 matching files
Uploading artifacts to coordinator... ok id=4467371 responseStatus=201 Created token=ionuncER8
Build succeeded
```

On the right side, there are sections for "Build artifacts" (Download, Browse), "Build details" (Duration: 3 seconds, Finished: less than a minute ago, Runner: #28616, Raw, Erase), "Commit title" (checking tag creation), "Tags" (shell), "Stage" (build), and a status bar showing "pages" with a checkmark.

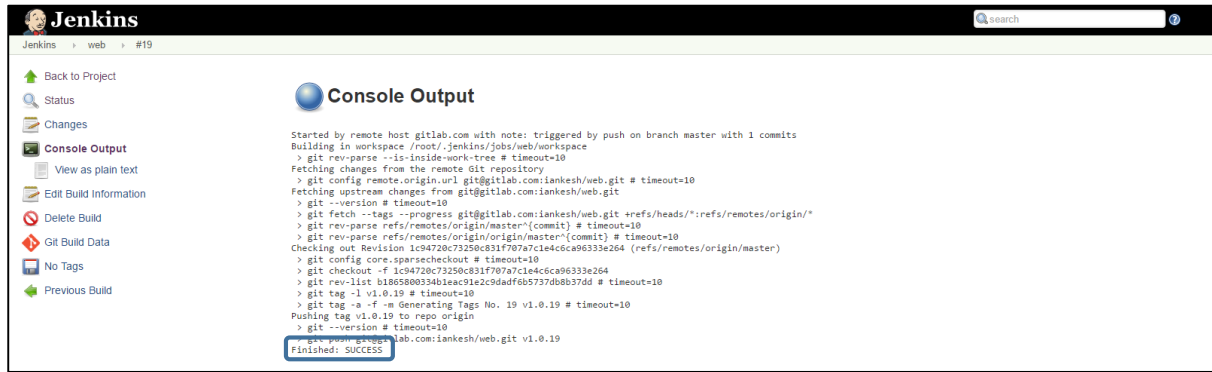
And when GitLab builds, Jenkins has already taken the changes from the GitLab repository, keep the build in pending and wait for Gitlab to complete its build.

The screenshot shows the Jenkins web interface. The "Build History" section on the left lists builds #13 through #19. Build #19 is highlighted with a blue box and labeled "pending—in the quiet period. Expires in 0.25 sec". The "Permalinks" section on the right lists links for the last build, last stable build, last successful build, and last completed build, all for build #19.

As soon as Gitlab finishes its work, Jenkins will start building the job.

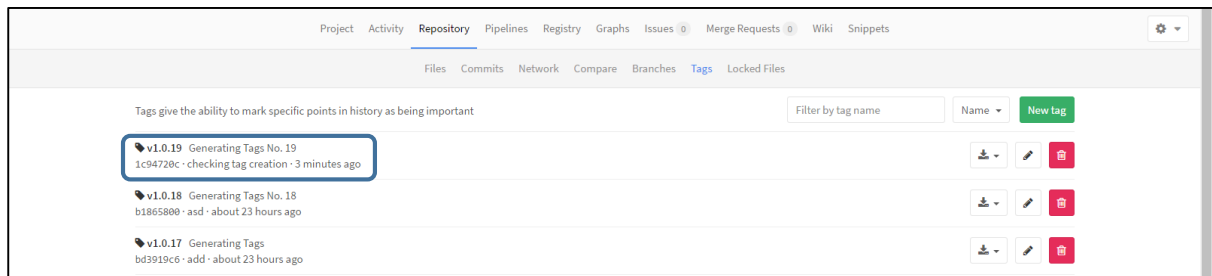
The screenshot shows the Jenkins web interface after the build has completed. The "Build History" section on the left lists builds #13 through #19. Build #19 is highlighted with a blue box and labeled "Successful". The "Permalinks" section on the right lists links for the last build, last stable build, last successful build, and last completed build, all for build #19.

If the build is succeeded, it generates a tag in GitLab.



Step 10:

After jenkins build got succeeded, you can see the tags generated in GitLab Tag section.



Mindtree\M1036336