Gitlab webhook integration with Jenkins and Slack

Configuring notification channel for Gitlab updates:

1. Create a Slack app (if you don't have one already)

You won't get very far without doing this step, but luckily it's very simple, we even have a nice green button for you to click:

Create your Slack app

Pick a name, choose a work-space to associate your app with (bearing in mind that you'll probably be posting lots of test messages, so you might want to create a channel for sandbox use), and then click *Create App*. If you've already created one, you can use it too.

2. Enable Incoming Webhooks

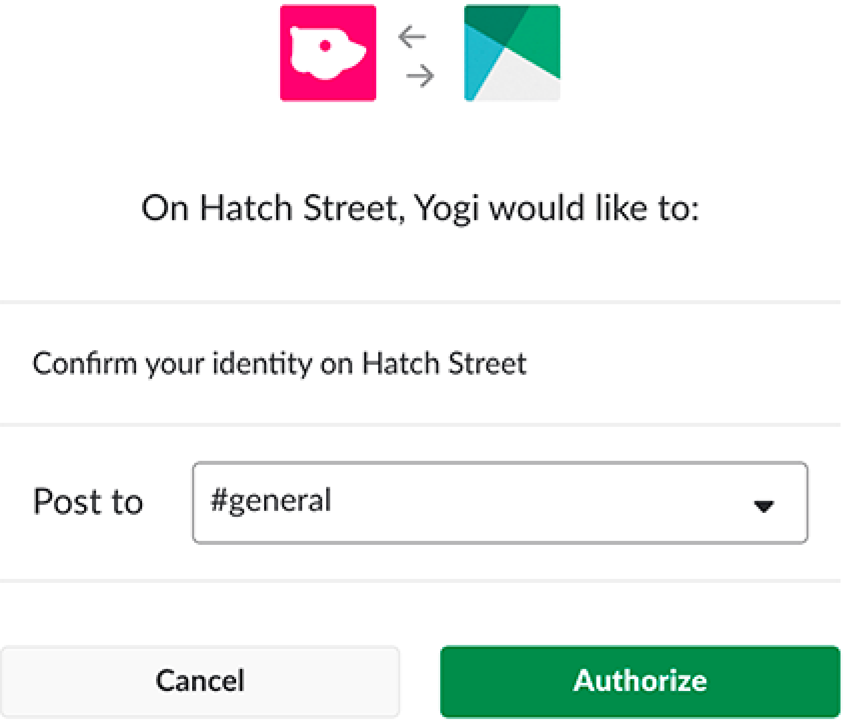
After creating, you'll be redirected to the settings page for your new app (if you are using an existing app, just load its settings via your app's management dashboard).

From here select the *Incoming Webhooks* feature, and click the *Activate Incoming Webhooks* toggle to switch it on. If you already have this activated.

3. Create an Incoming Webhook

Now that Incoming Webhooks are enabled, the settings page should refresh and some extra options will appear. One of those options will be a really helpful button marked *Add New Webhook to Workspace*, and you should click it.

What this button does is trigger a shortcut version of the installation flow for Slack apps, one that is completely self-contained so that you don't have to actually build any code to generate an Incoming Webhook URL. We'll show how you can generate webhooks programmatically later, but for now you'll see something like the following screen:



On Hatch Street, Yogi would like to:

Confirm your identity on Hatch Street

Post to #general

Go ahead and pick a channel that the app will post to, and then click to *Authorize* your app.

You'll be sent back to your app settings, and you should now see a new entry under the *Webhook URLs for Your Workspace* section, with a Webhook URL that'll look something like this:

```
https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXX
XXXXX
```

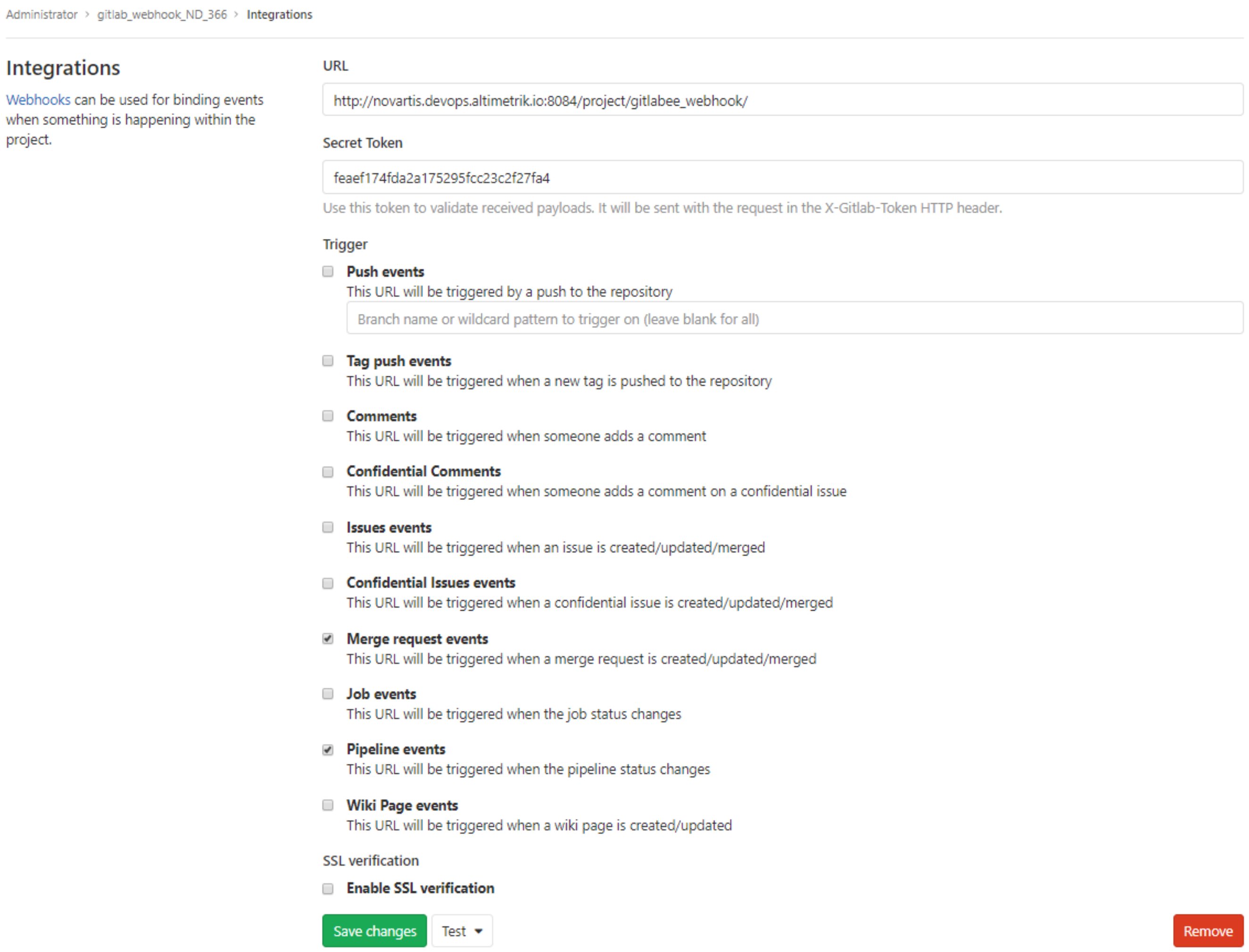
That URL is your shiny new Incoming Webhook, one that's specific to a single user, and a single channel. We've kind of run out of cookies, but nice work anyway!

Let's see how you can actually use that webhook to post a message.

Keep it secret, keep it safe. Your webhook URL contains a secret. Don't share it online, including via public version control repositories. **Slack actively searches out and revokes leaked secrets.**

Jenkins job to implement a use case with Gitlab webhook trigger :

Snapshot featuring Gitlab webhook:



Administrator › gitlab_webhook_ND_366 › Integrations

Integrations

Webhooks can be used for binding events when something is happening within the project.

URL

http://novartis.devops.altimetrik.io:8084/project/gitlabee_webhook/

Secret Token

feaef174fda2a175295fcc23c2f27fa4

Use this token to validate received payloads. It will be sent with the request in the X-Gitlab-Token HTTP header.

Trigger

□ **Push events**
This URL will be triggered by a push to the repository
Branch name or wildcard pattern to trigger on (leave blank for all)

□ **Tag push events**
This URL will be triggered when a new tag is pushed to the repository

□ **Comments**
This URL will be triggered when someone adds a comment

□ **Confidential Comments**
This URL will be triggered when someone adds a comment on a confidential issue

□ **Issues events**
This URL will be triggered when an issue is created/updated/merged

□ **Confidential Issues events**
This URL will be triggered when a confidential issue is created/updated/merged

☑ **Merge request events**
This URL will be triggered when a merge request is created/updated/merged

□ **Job events**
This URL will be triggered when the job status changes

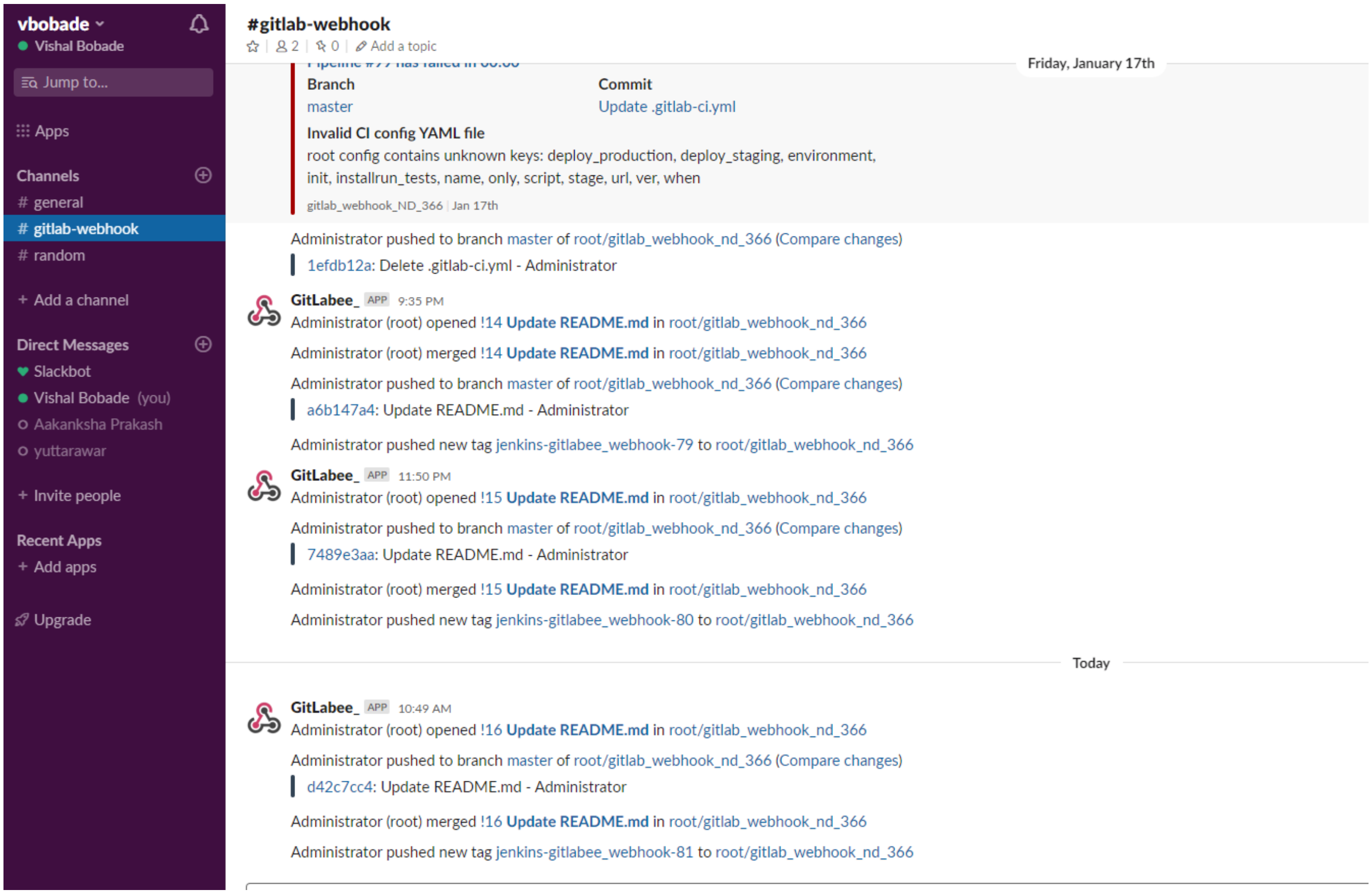
☑ **Pipeline events**
This URL will be triggered when the pipeline status changes

□ **Wiki Page events**
This URL will be triggered when a wiki page is created/updated

SSL verification

□ **Enable SSL verification**

Slack updates can be seen as below:



Friday, January 17th

Pipeline #77 has failed in 00:00

Branch	Commit
master	Update .gitlab-ci.yml

Invalid CI config YAML file

root config contains unknown keys: deploy_production, deploy_staging, environment, init, installrun_tests, name, only, script, stage, url, ver, when

gitlab_webhook_ND_366 | Jan 17th

Administrator pushed to branch master of root/gitlab_webhook_nd_366 (Compare changes)

1efdb12a: Delete .gitlab-ci.yml - Administrator

GitLabee_ APP 9:35 PM

Administrator (root) opened !14 **Update README.md** in root/gitlab_webhook_nd_366

Administrator (root) merged !14 **Update README.md** in root/gitlab_webhook_nd_366

Administrator pushed to branch master of root/gitlab_webhook_nd_366 (Compare changes)

a6b147a4: Update README.md - Administrator

Administrator pushed new tag jenkins-gitlabee_webhook-79 to root/gitlab_webhook_nd_366

GitLabee_ APP 11:50 PM

Administrator (root) opened !15 **Update README.md** in root/gitlab_webhook_nd_366

Administrator pushed to branch master of root/gitlab_webhook_nd_366 (Compare changes)

7489e3aa: Update README.md - Administrator

Administrator (root) merged !15 **Update README.md** in root/gitlab_webhook_nd_366

Administrator pushed new tag jenkins-gitlabee_webhook-80 to root/gitlab_webhook_nd_366

Today

GitLabee_ APP 10:49 AM

Administrator (root) opened !16 **Update README.md** in root/gitlab_webhook_nd_366

Administrator pushed to branch master of root/gitlab_webhook_nd_366 (Compare changes)

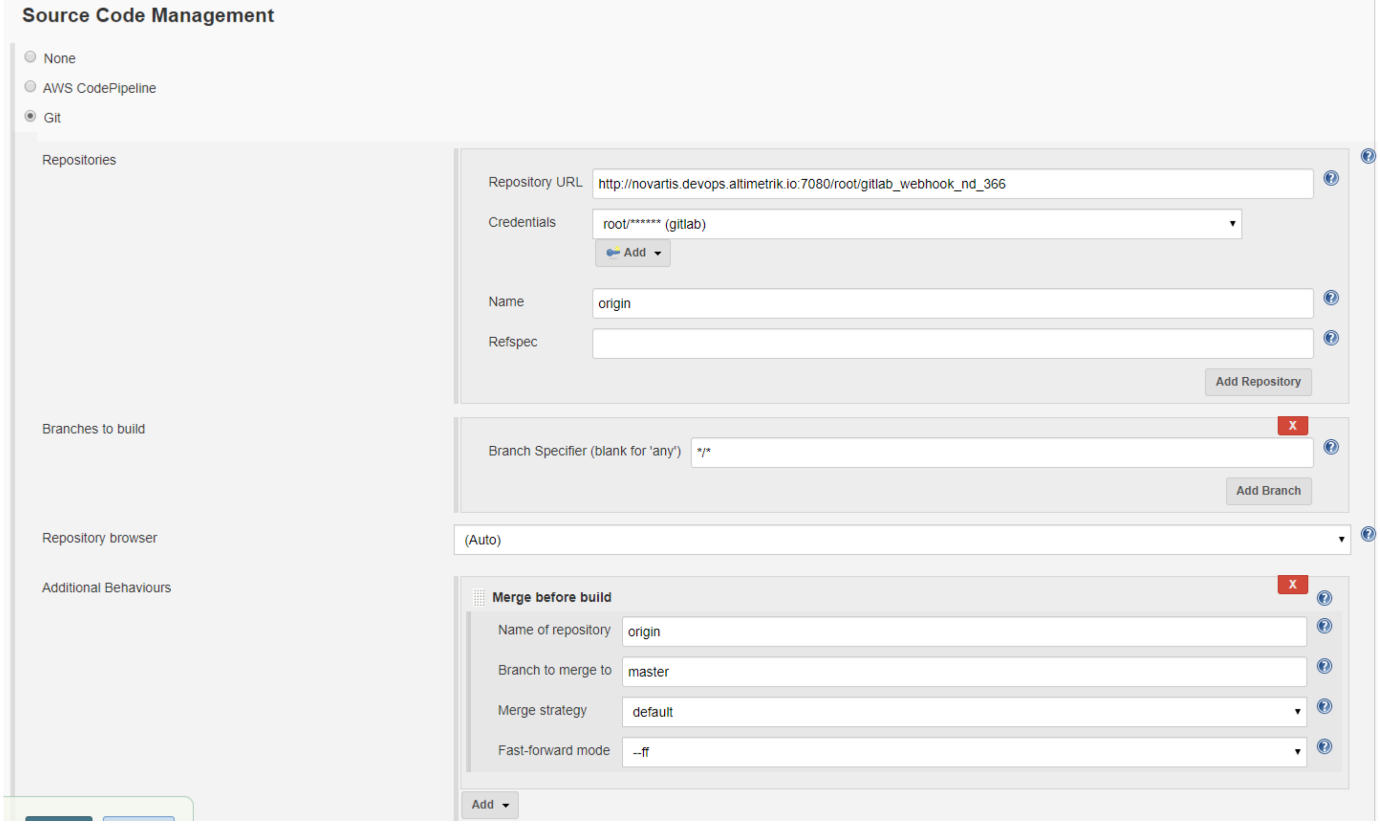
d42c7cc4: Update README.md - Administrator

Administrator (root) merged !16 **Update README.md** in root/gitlab_webhook_nd_366

Administrator pushed new tag jenkins-gitlabee_webhook-81 to root/gitlab_webhook_nd_366

Jenkins job configuration will look like below:

Step 1



Source Code Management

○ None
○ AWS CodePipeline
● Git

Repositories

Repository URL: http://novartis.devops.altimetrik.io:7080/root/gitlab_webhook_nd_366

Credentials: root/****** (gitlab)

Name: origin

Refspec:

Branches to build

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

Repository browser: (Auto)

Additional Behaviours

Merge before build

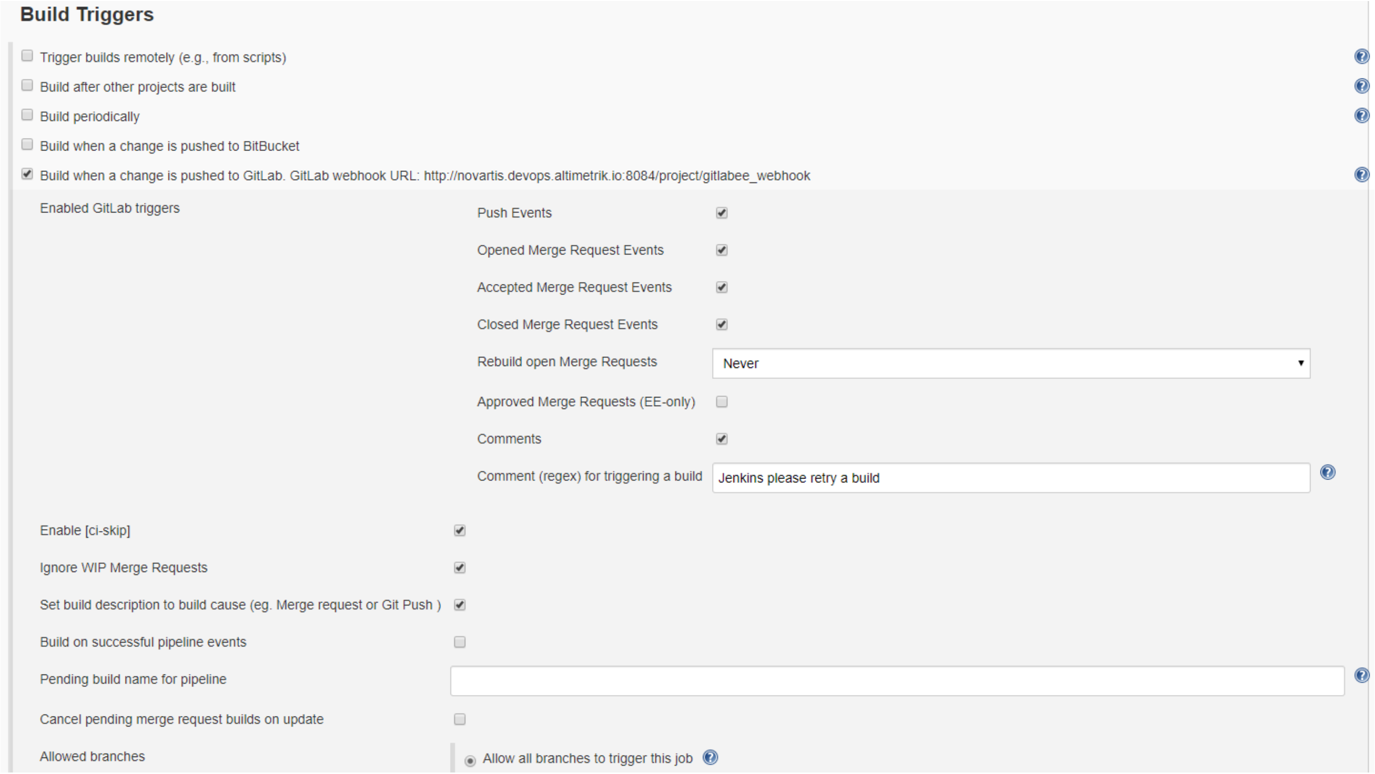
Name of repository: origin

Branch to merge to: master

Merge strategy: default

Fast-forward mode: --ff

Step 2



Build Triggers

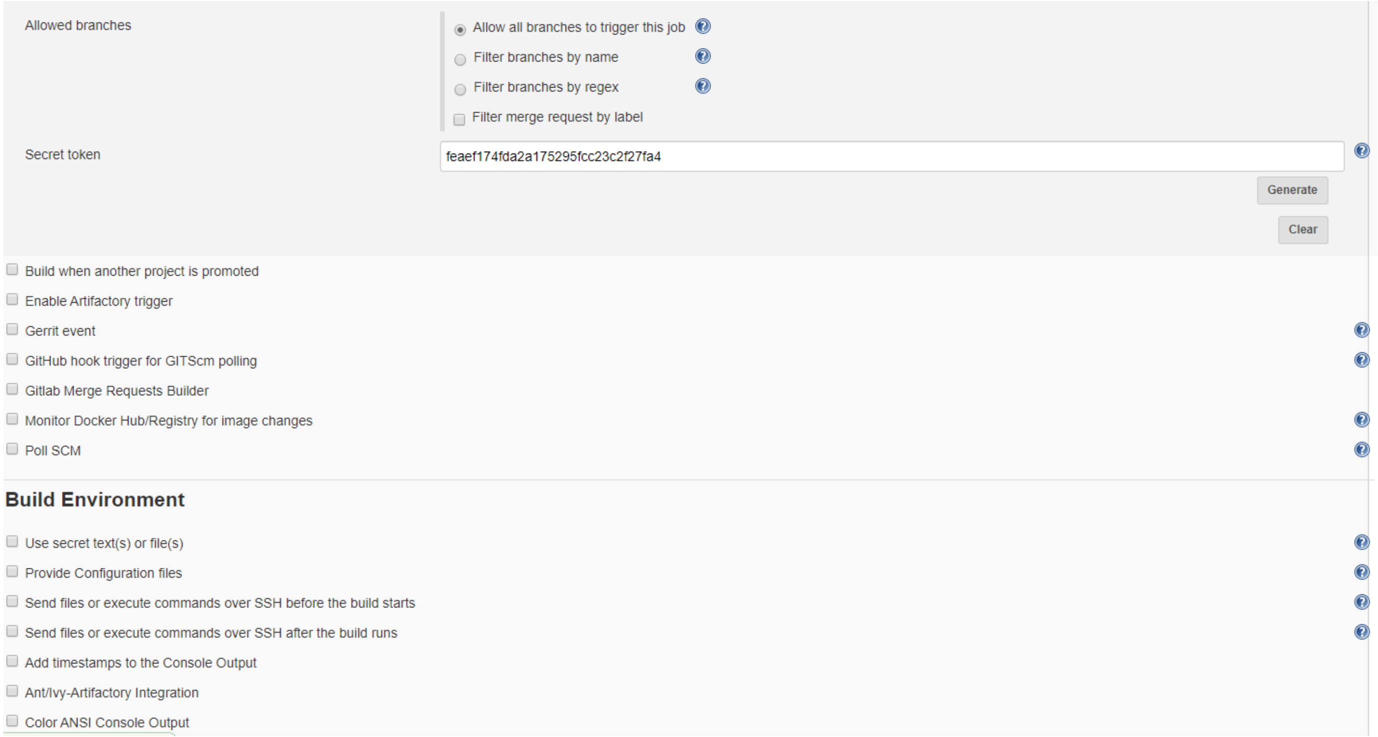
- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Build when a change is pushed to BitBucket
- ☑ Build when a change is pushed to GitLab. GitLab webhook URL: http://novartis.devops.altimetrik.io:8084/project/gitlabee_webhook

Enabled GitLab triggers

- Push Events ☑
- Opened Merge Request Events ☑
- Accepted Merge Request Events ☑
- Closed Merge Request Events ☑
- Rebuild open Merge Requests: Never
- Approved Merge Requests (EE-only) ☐
- Comments ☑
- Comment (regex) for triggering a build: Jenkins please retry a build

- Enable [ci-skip] ☑
- Ignore WIP Merge Requests ☑
- Set build description to build cause (eg. Merge request or Git Push) ☑
- Build on successful pipeline events ☐
- Pending build name for pipeline:
- Cancel pending merge request builds on update ☐
- Allowed branches: ● Allow all branches to trigger this job

Step 3



Allowed branches

- ● Allow all branches to trigger this job
- ○ Filter branches by name
- ○ Filter branches by regex
- ☐ Filter merge request by label

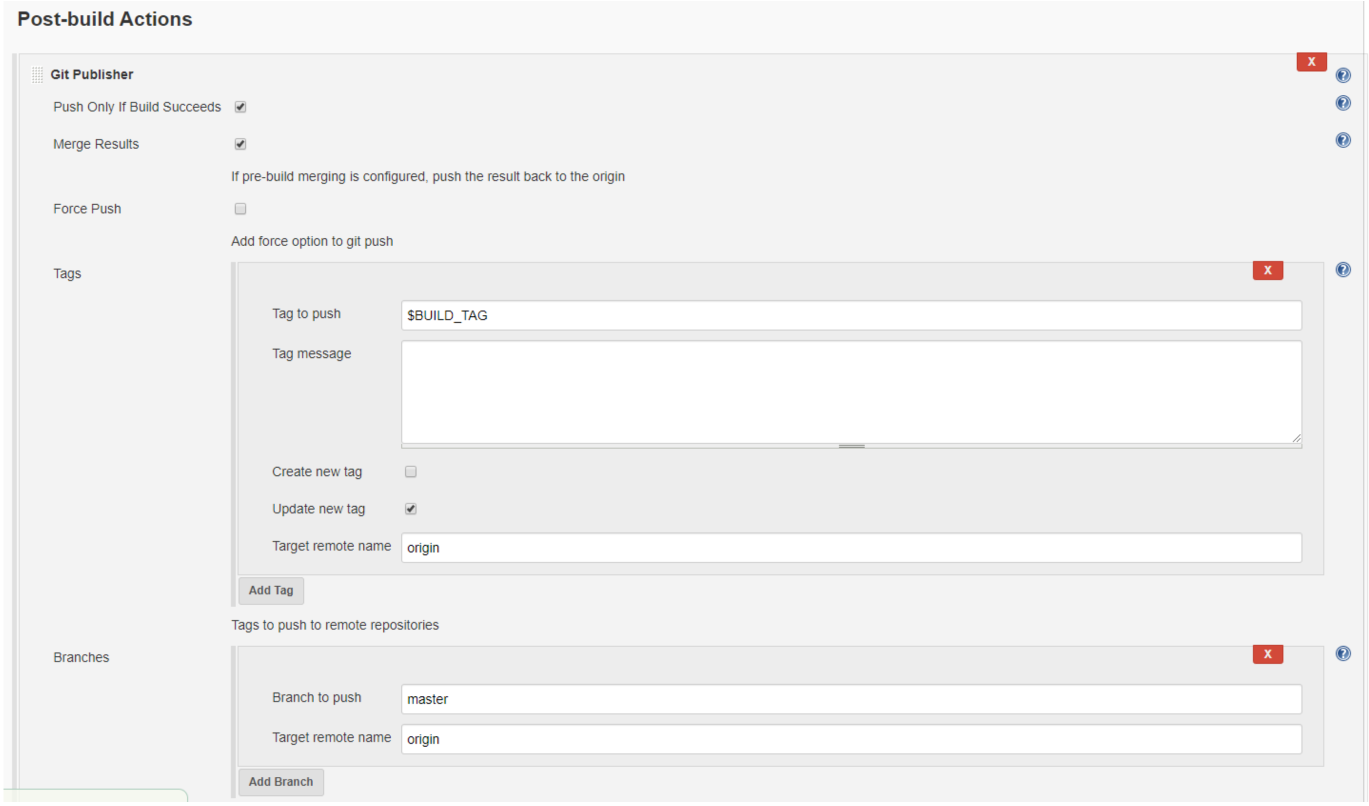
Secret token: feaef174fda2a175295fcc23c2f27fa4

- ☐ Build when another project is promoted
- ☐ Enable Artifactory trigger
- ☐ Gerrit event
- ☐ GitHub hook trigger for GITScm polling
- ☐ Gitlab Merge Requests Builder
- ☐ Monitor Docker Hub/Registry for image changes
- ☐ Poll SCM

Build Environment

- ☐ Use secret text(s) or file(s)
- ☐ Provide Configuration files
- ☐ Send files or execute commands over SSH before the build starts
- ☐ Send files or execute commands over SSH after the build runs
- ☐ Add timestamps to the Console Output
- ☐ Ant/Ivy-Artifactory Integration
- ☐ Color ANSI Console Output

Step 4



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: $BUILD_TAG

Tag message:

Create new tag □

Update new tag ☑

Target remote name: origin

Tags to push to remote repositories

Branches

Branch to push: master

Target remote name: origin

Pre-requisites required for Gitlab - Jenkins Integration.

Gitlab

Gitlab Login Credentials

Gitlab Url like http://novartis.devops.altimetrik.io:7080

Jenkins

Jenkins Login Credentials

Jenkins Url like http://novartis.devops.altimetrik.io:8084

Showcasing Jenkins Integrations within Git lab.

Step 1 : Add the corresponding repository within Git lab repository with in Jenkins for a free style job.

Step 2 : Build the corresponding jobs available using build jobs option in Jenkins.

Step 3: Use the option Publish build status to git lab which would reflect he status of Jenkins for the repository on Git lab.

The above sequence would help in creating Git lab integrations to reflect Jenkins build status in Git lab.

Application of Git lab hooks to trigger jobs on Jenkins.

Procedure to add Web hooks on Jenkins

Create a job which would have a default option of a generated web hook url on jenkins.

Create a web hook in the settings of repository

Project > "Project name" > settings > Integrations.

Add the list of features to be added

To test the feature

Click on the Test button provided above.

Once Web hook is working the below would be appearing on Jenkins.

Feature of Gitlab:

Merge request can be enabled only when Build is successful for the Branch.

The above feature can be enabled only by the below steps

Go to the settings > repository > Merge request

Check the below to enable the merge requests for successful merges of branches.

Successful pipelines of branches are appeared in the below way.

Once the build pipelines are successful the the two branches can be merged.

References:

https://docs.bitnami.com/aws/how-to/create-ci-pipeline/