Building Gitlab Merge Requests With Jenkins

This guide is a step by step walkthrough of setting up Jenkins to build merge requests made against a Gitlab repository. This is incredibly useful because it can provide you with information about whether each merge request will merge smoothly into the project. As well as if any tests are broken in the merge request. Our team used a build job to build each merge request and used a second build job to continuously build the master branch. In order to do this we used Jenkins plugin that someone created to make this easy. This guide assumes that you have already installed Jenkins.

Installing Necessary Plugins

There are two plugins that are necessary for the Jenkins server to automatically build each merge request made into your projects Gitlab repository. The Gitlab Merge Request Builder plugin and the Git plugin.

1. Log into Jenkins
2. From the main page click the Manage Jenkins button on the right hand side of the screen.
3. Then select the Manage Plugins button from the next page.
4. Click the Available tab at the top of the list.
5. Find the Git Plugin and check the box.
6. Find the Gitlab Merge Request Builder and check the box.
7. Now click the Download and Install button.
8. Wait for the two plugins to be installed and restart Jenkins if it doesn’t auto restart.

Configuring the Gitlab Merge Request Builder Plugin

This plugin requires a little bit of configuration so that it functions properly.

1. Navigate back to the main page.
2. Click the Manage Jenkins button on the left hand side of the screen.
3. From the next page click Configure System.
4. Find the section labeled Gitlab Merge Request Builder.
5. In the Gitlab Host URL section add the url to the Gitlab server at metro. This is not the URL to your project on that server, but the url you would use to get to that server in the browser. At the time this guide was written the URL was https://gouda.msudenver.edu/gitlab/   This is different then a URL to a specific repository because that would have a repository name on it.
6. Next enter the username on Gitlab for the Jenkins user into the Jenkins Username text field. We just used one of our team members Gitlab account, and put their username here.
7. Then, add the API token for that user on Gitlab into the Jenkins API token text field. This is obtained through the Gitlab server.

8. On the Crontab line enter the following to have the Jenkins server and this plugin to poll the Gitlab server every five minutes for new merge requests. To learn more about Cron Job schedules you can click the help button next to this field in Jenkins. There is a space between each character except for H/5
   H/5 * * * *
9. Next, if you want the Jenkins server to post a comment to the merge request that it has started a build then check the box labeled Enable build triggered message.
10. Make sure that the Default Success, Default unstable and Default failure message fields are filled in with useful information.
11. Finally, checkmark the box Ignore SSL Certificate Errors.
12. Click the Save button

Creating The Build Job

Jenkins Jobs are what is used to build the application. We will need a specific job set up for the project.

1. Log into Jenkins.
2. From the main page click the new item button on the left hand side of the screen.
3. Now you can select Freestyle project
4. Give it a meaningful name in the Item name text field.
5. Click the OK button near the bottom of the page. This should take you to the configuration page for the job.

Configuring the Build Job

All the configuration for this job will occur on the configuration page. First, you must connect Jenkins to the Gitlab Repository and provide it access to the code. Since we are using the plugin and want to build all merge requests we have to use some variables and expressions here. Next, to automate the process it is very easy to set up Jenkins to use the Gitlab Merge Request Builder plugin. When merge requests are made then it will get the code from the project repository and the code from the merge request. It will merge the code together and run all tests. The results of this will be posted to the merge request on Gitlab with the Default Messages you set up in the last steps. Finally, you have to set a Environment Variable to be used during the build process.

Connecting to the Gitlab Repository

1. Find the Source Code Management section on the page and select Git.
2. For the Repository URL you have to enter the following

   $(gitlabSourceRepository)

3. Click the Add button next to the credentials drop down list. You need to enter in credentials for a user account on the Gitlab server that the Jenkins server should use.

   We used a username and password for Gitlab

4. Now select the credentials you just created from the drop down list labeled Credentials.
5. In the Branches to build enter the following to be able to access both the branch being merged into, and the branch that the merge is for.

   $(gitlabSourceName)/$(gitlabSourceBranch)

6. From the Repository browser drop down select gitlab.
   a. In the URL text field that appeared enter the full URL to your projects repository. Where the merges will be going too.
   b. For the version enter the version of Gitlab that the school is using. At the time of writing this guide in Spring of 2015 the Version was 3.1
7. Now from the Additional Behaviors dropdown select Merge before build.
   a. Put origin into the text field for Name of repository.
   b. Enter the following into the text field for Branch to merge to $(gitlabTargetBranch)

   c. Select default for the Merge strategy.

Using the Gitlab Merge Request Builder Plugin

1. Find the Build Triggers section right below Source Code Management section.
2. Click the checkbox for Gitlab Merge Requests Builder, and a few more configuration fields will appear.
3. For the Gitlab project enter the path to your project on Gitlab. This is not the full URL to gitlab and it is appended to the end of the URL you entered back when you configured the plugin. For our project this value was *falcon/cs-degree-advisor-and-planning-assistant*
4. Leave the Target Branch Regex text field blank.
5. The crontab line should be filled out already and should be the same as what you set in back when you configured the plugin.
6. Finally click the Use HTTP(s) URL checkbox.

Setting an Environment Variable

1. The final configuration for the Job is a environment variable that Jenkins will need to have in order to access the code. Find the Build Environment section.
2. Check the Inject environment variables to the build process checkbox.
3. In the Properties Content text area enter the following. GIT_SSL_NO_VERIFY=1

Create Build Script

The last step is to create a Build Script for this project. This Build Script is a shell script that when executed will perform all the steps necessary to build and execute the tests on this Rails application.

1. Find the Build section right underneath the Build Environment section.
2. Click the Add build step and select Execute shell.
3. In the Command box enter the following.
   ```
   echo "Started Build" $BUILD_NUMBER
   source ~/.bashrc
   rvm use --create ruby-2.2.0@my_app
   rvm --force gemset empty
   gem install bundler --no-rdoc --no-ri
   bundle install
   bundle exec rake db:drop:all
   bundle exec rake db:create:all
   bundle exec rake db:migrate
   bundle exec rake db:test:prepare
   bundle exec rake test
   ```

4. Finally click the Apply button at the bottom of the page.

Now every time a merge request is made against the project repository. Jenkins will take that merge request and check whether it will merge smoothly. It will also run all tests against the project. If any of the tests or the merge fails then it will comment on the merge request that it is currently failing. If everything completes successfully then it will comment on the merge request that everything was successful. This provides clear feedback very quickly about the state of each merge request.