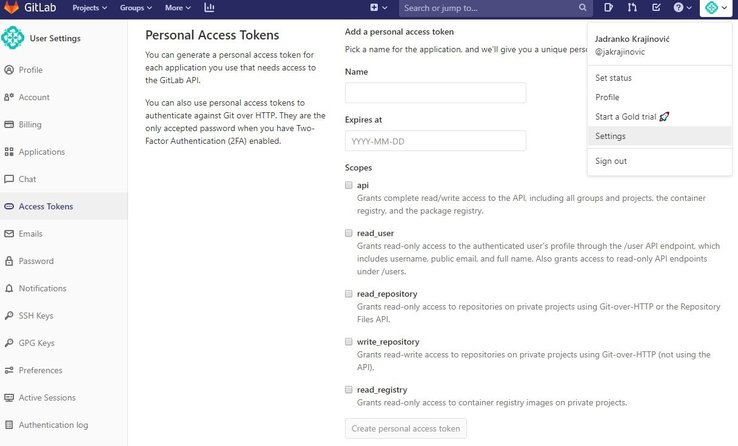
**Configuring Gitlab and Jenkins**

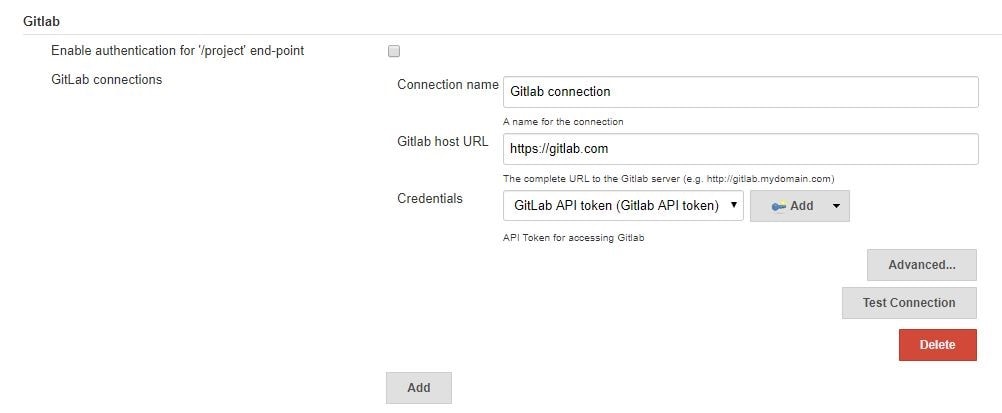
**Jenkins: Access Rights to GitLab**

In order to use GitLab with Jenkins, you’ll need to generate an access token in GitLab, which you can do in **User menu > Settings > Access tokens**

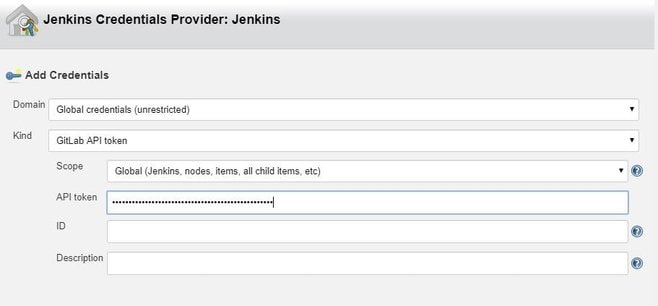


and configure GitLab Connection on Jenkins by adding the newly generated token.

In Jenkins, go to **Manage Jenkins > Configure system** and find the GitLab section.

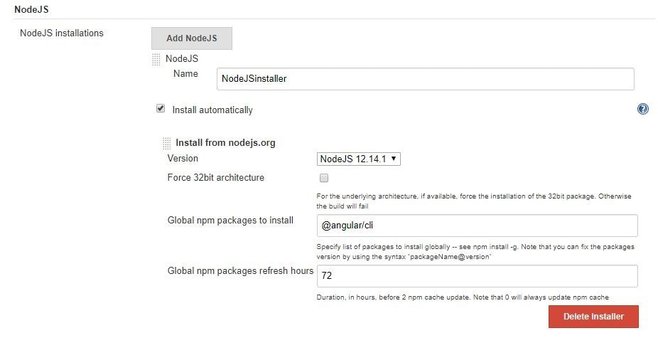


To add a token that you previously generated, click on **Add** by the Credentials input and choose Jenkins. In the credentials dialog, choose **GitLab API token** in the *Kind* input and paste your token from GitLab into the API token input field.



**Jenkins: Configure NodeJSInstaller**

In order to be able to run npm scripts, it is necessary to configure NodeJSInstaller. In Jenkins, go to **Manage Jenkins > Global Tool Configuration > NodeJS installations.**

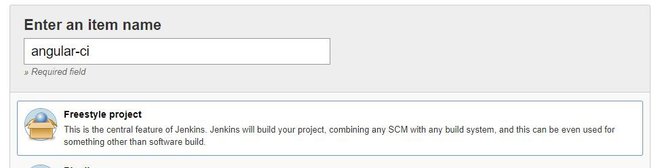


**Jenkins: Create CI build for Angular**

In order to be able to run Angular tests and check your code style in Jenkins on the created merge request in GitLab you’ll have to:

1. Click on the *New item* link in the Jenkins dashboard

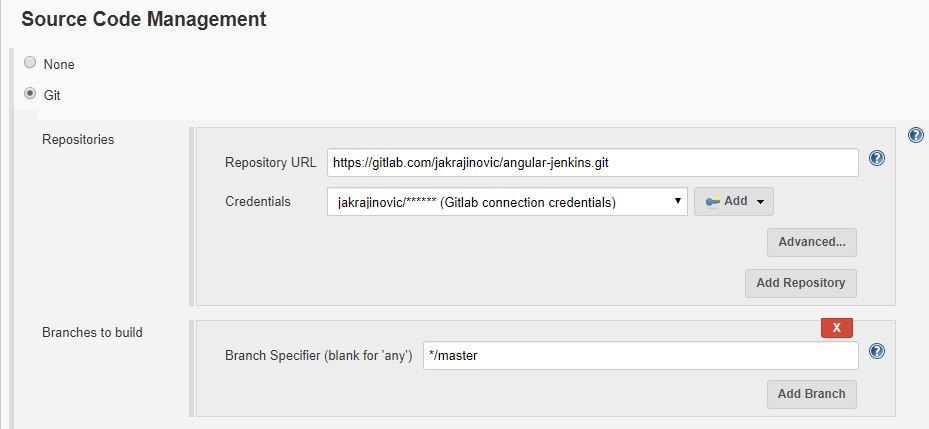
2. Enter a job name and choose *Freestyle project*



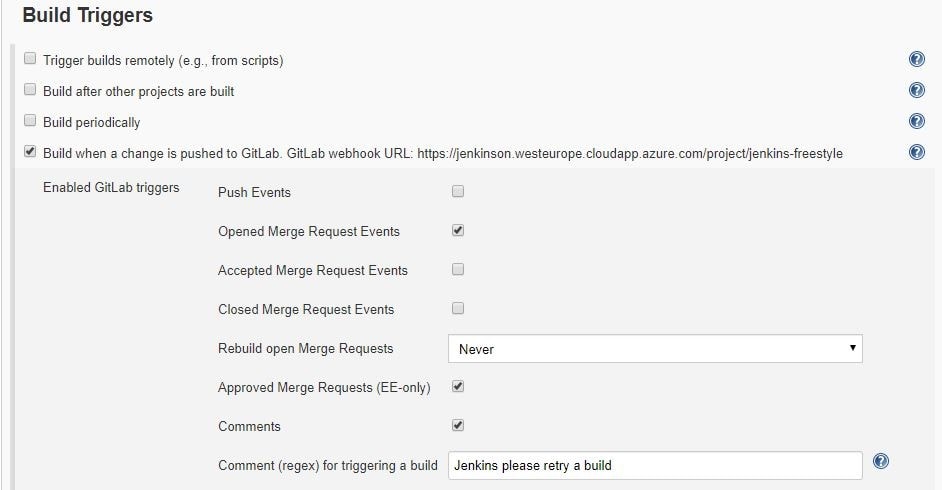
3. Choose the GitLab Connection that we’ve just created in the Gitlab Connection section.



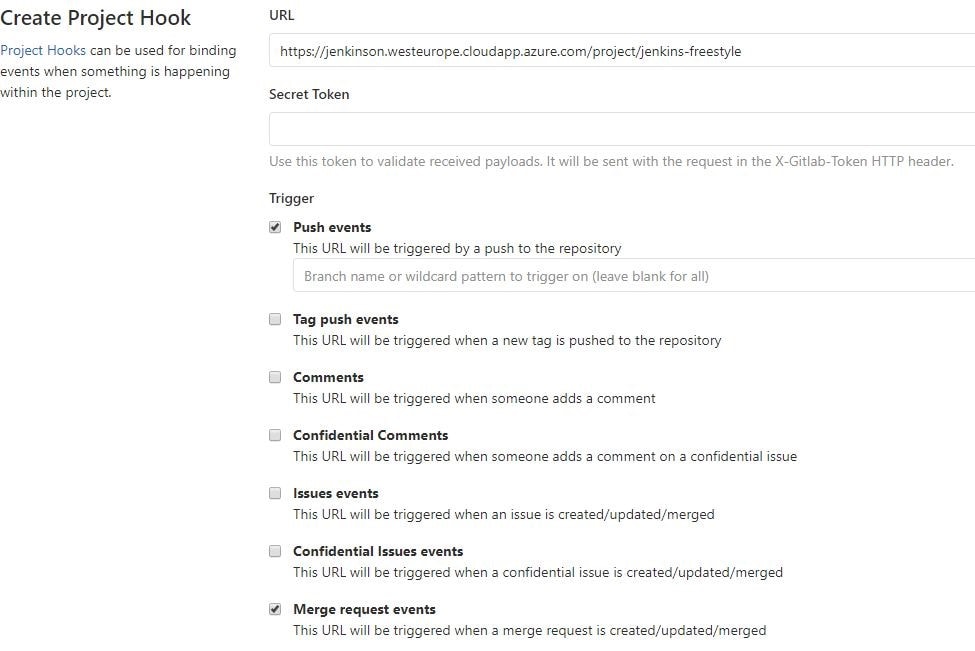
4. Choose Git as your source code management. Enter your repository URL. Create new credentials on Jenkins. These credentials are for cloning the project. You use them to log in to Gitlab.



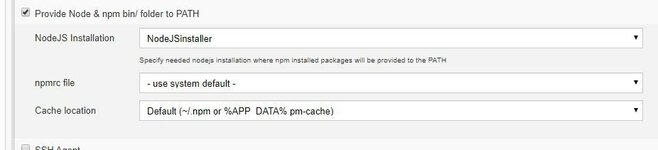
5. Next, configure build triggers, i.e. on which GitLab event to run a build. In this particular example, angular-ci-build is going to trigger when a new merge request gets created.



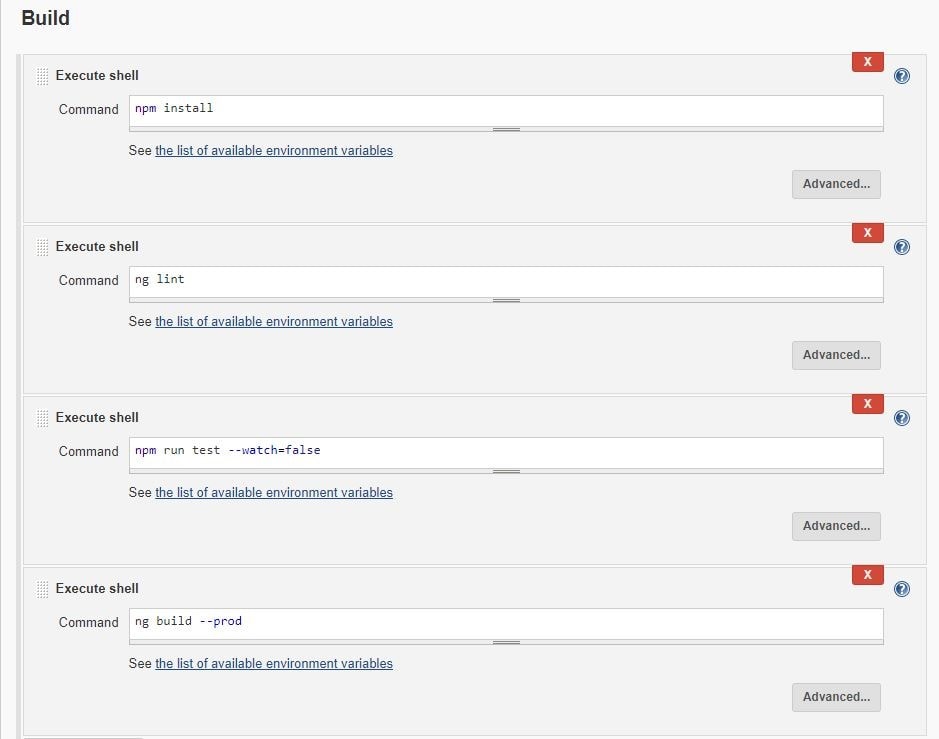
In this step, we need to go back to GitLab and create a hook that will trigger this build under **Settings > Integrations**. Copy the URL provided by Jenkins and paste it into the project hook form and finally click *Add webhook*.



6. Provide the configured NodeJsInstaller in the global configuration to be able to run npm commands.



7. And finally, in the Build section choose **Add build step > Execute shell.** Write shell scripts to test the Angular app code and run tests.



Click *Save* and we are good to go. At this point everything should work.

When you create a new merge request, GitLab should trigger angular-ci-build on Jenkins and you should see *status pending*on that particular merge request page.

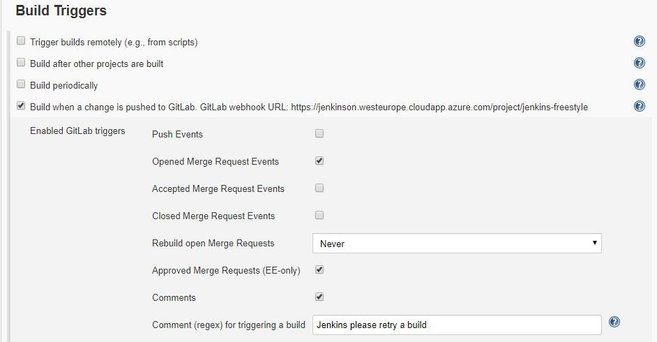
When Jenkins is done, the status on GitLab should automatically be updated. Depending on whether the build passed or not, the merge button will change color.

**Jenkins: Create CD Build for Angular**

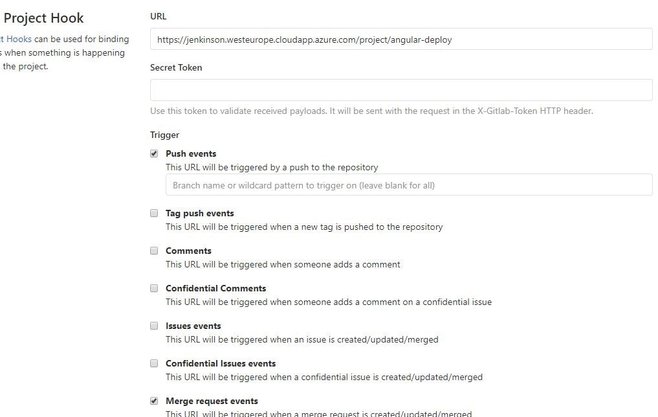
In order to be able to deploy Angular to another Linux machine, we need to:

Repeat steps 1–4 from *Jenkins: Create CI Build for Angular*, changing only the name of the build. This time, it can be angular-deploy.

5. For step five, we now choose a different configuration for deployment. We are going to run this build when a merge request gets accepted.

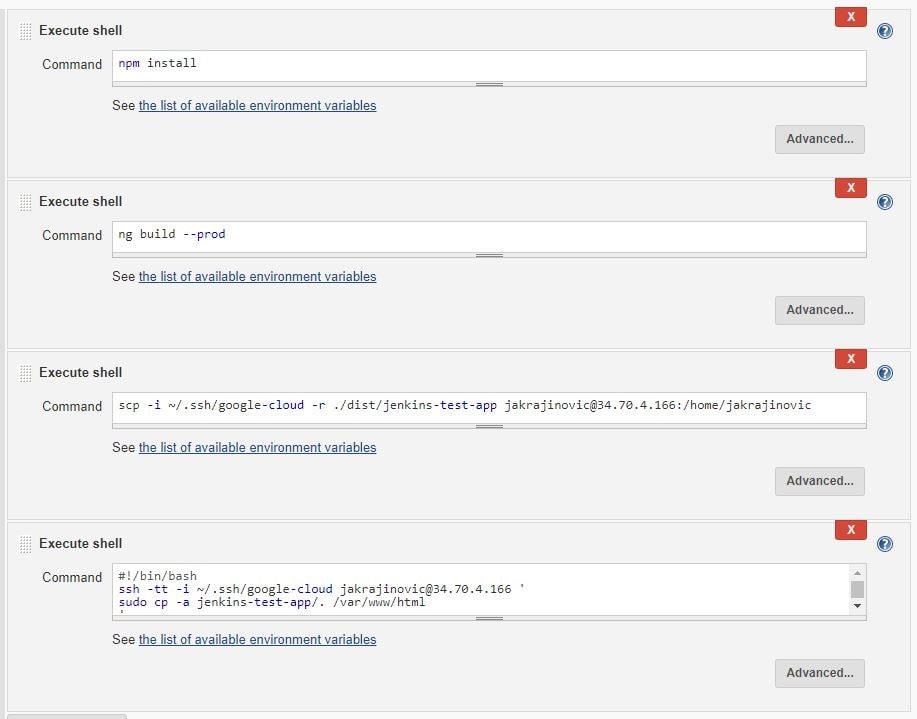


Just like for the CI build, we have to create a new GitLab hook that will hit the Jenkins build endpoint.



6. This step is also the same as in CI; we need to provide the NodeJSInstaller we already configured globally.

7. This step is different from CI; this time we don’t have to test and check linting, but only build the application and copy-paste it to another machine with ssh.

sv