Git and GitLab Quick Start Tutorial

leifengblog.net/blog/git-and-gitlab-quick-start-tutorial

15 Feb 2021 | Tutorial | Git • GitLab • GitHub

This post is a quick start manual for beginners to Git or GitLab. Although this manual aims for the GitLab Enterprise edition on Windows, the instructions apply to general Git and GitLab/GitHub practices. This post assumes that you have already set up a GitLab web account where you can restore and manage your source code.

Download and Install Git for Windows

The following instructions are for installing Git on Windows. If you need to install Git for other operating systems, please follow instructions Installing Git.

Go to https://git-scm.com/download/win, and the download will start automatically. If the download doesn't start automatically, you can manually click the download links.

When the download finishes, you will get an `.exe` file, and you can double-click it to install. You can keep all default settings during installation.

Configure Git for First Time

After installation of Git on your Windows system, you can start `Git Bash` from the **Start Menu**:

You can also start `Git Bash` from the **Context Menu** through right-clicking from any local directory:

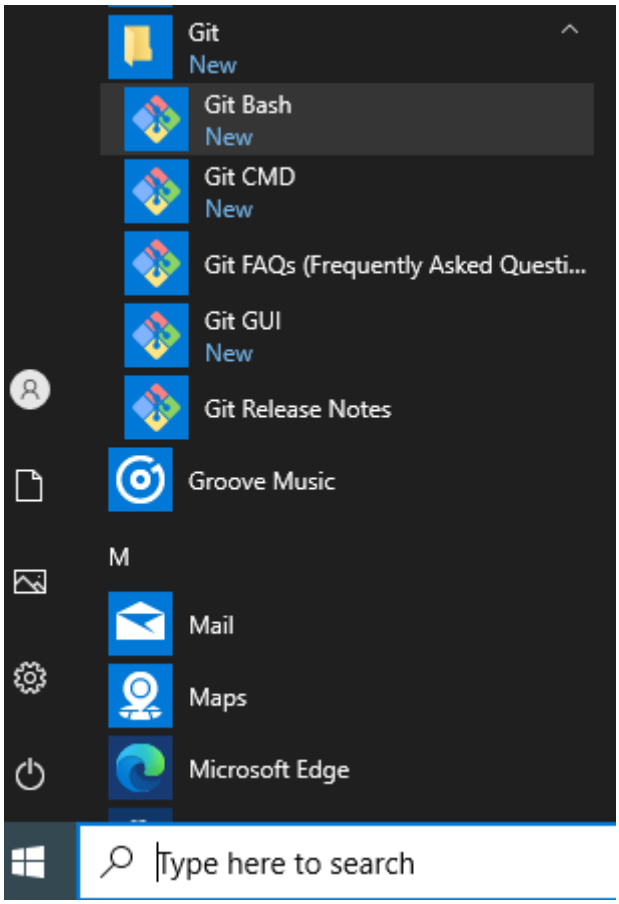
Once you start your `Git Bash` for the first time, it is a good idea to customize your Git environment by running the following commands. Note, the setting commands only need to do one time.

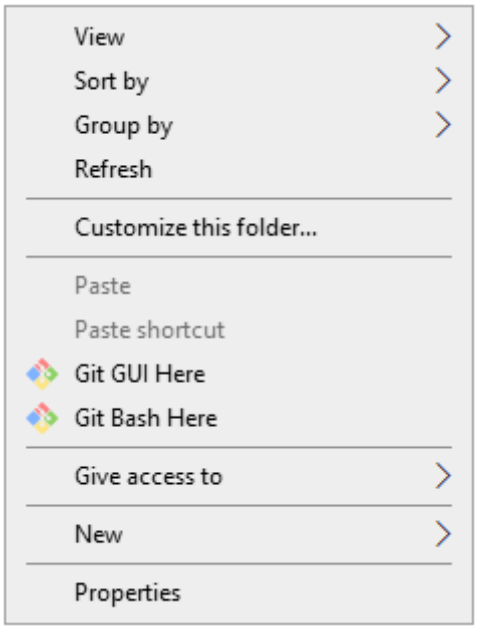
```
git config --global user.email
"you@example.com"
git config --global user.name "Your
Name"
```

You can check all your settings through running:

```
git config --list
```

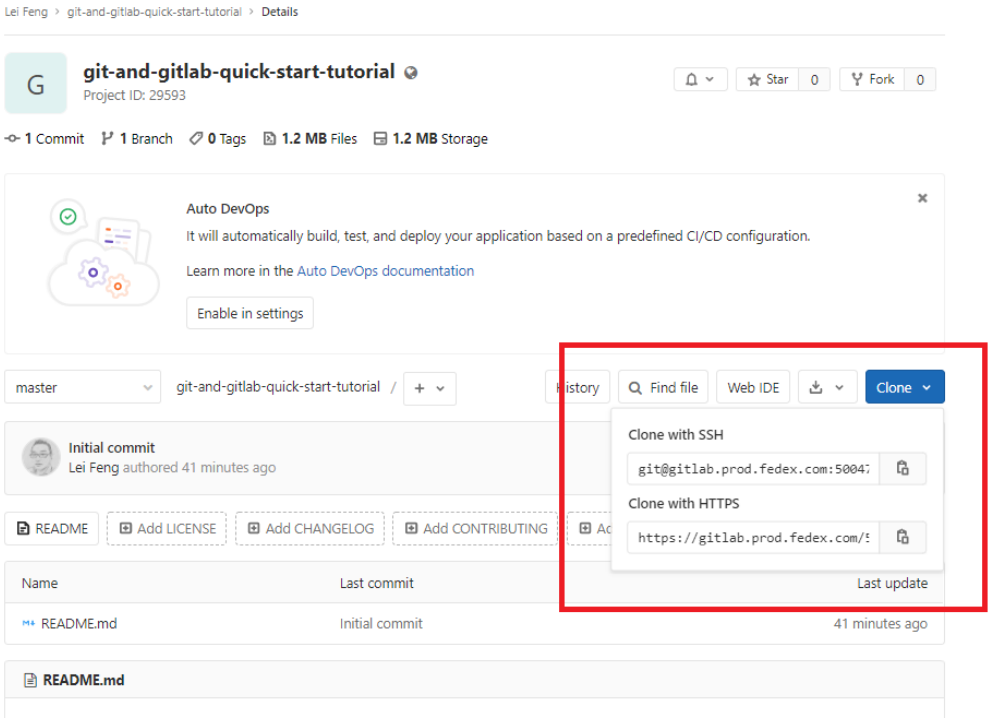
On Windows systems, this command looks for the file `.gitconfig` from your $HOME directory, which usually is `C:\Users\$USER`.





Clone a GitLab Repository

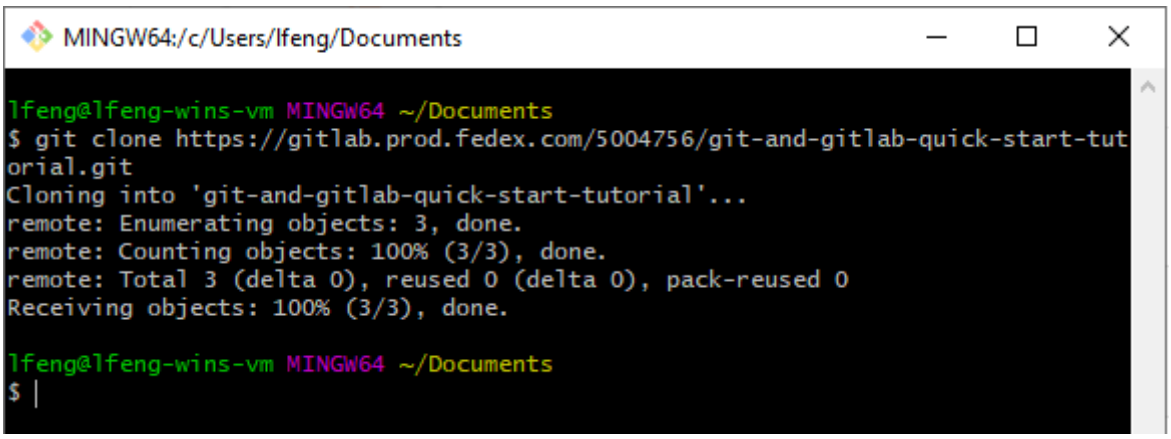
You can get a copy of a remote repository on GitLab through the `git clone` command. As shown in the following picture, there are two ways to connect to the remote repository: **HTTPS** and **SSH**.



Clone with HTTPS

You can copy the URL to the HTTPS and run the following command to clone a remote repository to your local directory. The command would prompt a message asking for the username and password to the remote GitLab project.

```
git clone <URL_HTTPS>
```



Clone with SSH

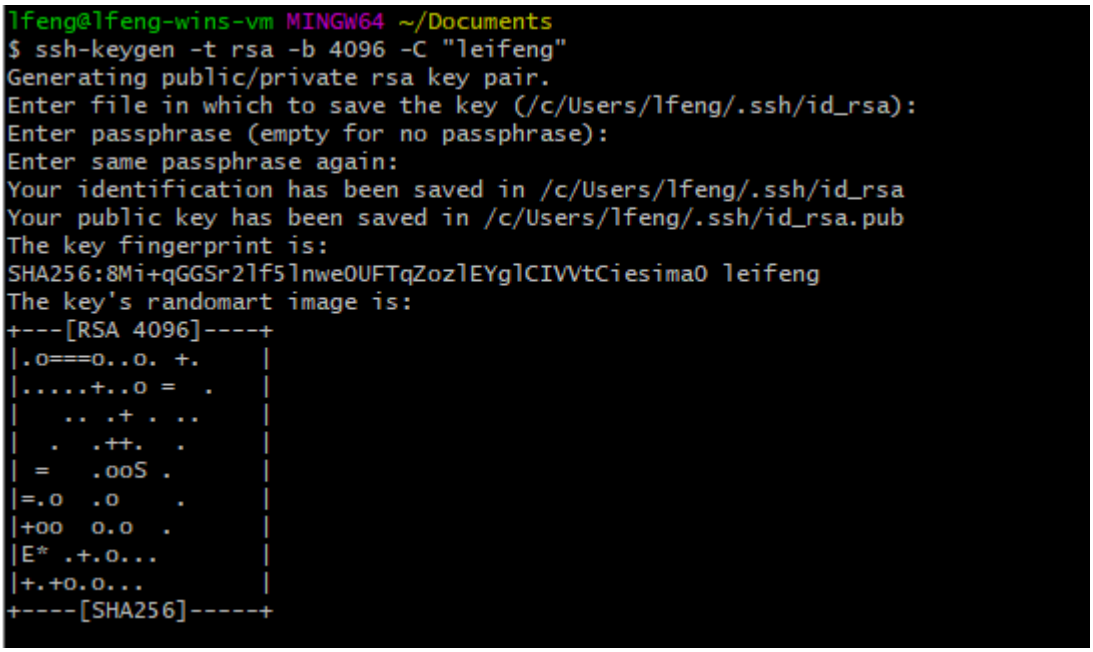
Cloning with SSH is securer and is the recommended way. To clone with an SSH connection, you need to follow the instructions below to set up SSH Key in your GitLab profile.

Generate SSH Key Pair

You can create a new SSH key pair by running the following command in the `Git Bash` or `PowerShell`. You can use the default key file name and empty passphrase. This command generates two files with names of `id_rsa` and `id_rsa.pub` under the path of

`C:\Users\$USER\.ssh\`.

```
ssh-keygen -t rsa -b 4096 -C "<USERNAME>" -f $HOME/.ssh/gitlab_rsa
```



Configure SSH in SSH Agent

Run the following commands

```
eval `ssh-agent -s`
ssh-add ~/.ssh/gitlab_rsa
```

Add the following settings into `~/.ssh/config` file:

```
# my company gitlab production environment
Host gitlab.prod.mycompany.com
  PreferredAuthentications publickey
  IdentityFile ~/.ssh/gitlab_rsa
```

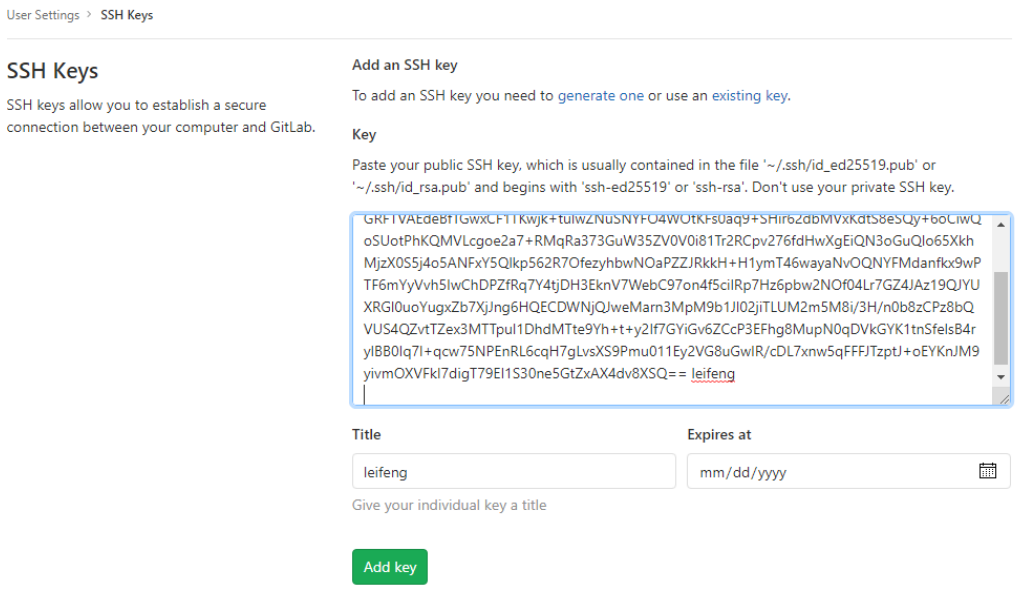
Generate a new one if it doesn't exist yet.

Set up SSH Keys in GitLab Profile

Once you get the SSH key pair, set up the SSH key through the following steps:

- Log into GitLab, click your profile picture in the upper-right corner of the page
- Select **Settings** from the dropdown list
- Select **SSH Keys** from the **User Settings** on the left menu
- Copy the content from generated `id_rsa.pub` and paste it in the **Key** box and set up the expiration date if needed

- Click **Add key**, you will get an email notifying you added a key to your profile.



Clone with SSH Command

Once you set up your SSH key in your GitLab profile, you can run the following command with the corresponding SSH URL to clone the remote repository without inputting your username and password.

```
git clone <URL_SSH>
```

```
lfeng@lfeng-wins-vm MINGW64 ~/Documents
$ git clone git@gitlab.prod.fedex.com:5004756/git-and-gitlab-quick-start-tutorial.git
Cloning into 'git-and-gitlab-quick-start-tutorial'...
The authenticity of host 'gitlab.prod.fedex.com (204.135.87.224)' can't be established.
ECDSA key fingerprint is SHA256:JMUhsOPcC5zCOqLMKvobe7ROtGeewAk7OUsUi7Tmqkw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'gitlab.prod.fedex.com,204.135.87.224' (ECDSA) to the list of known hosts.
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
Receiving objects: 100% (3/3), 251 bytes | 251.00 KiB/s, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
```

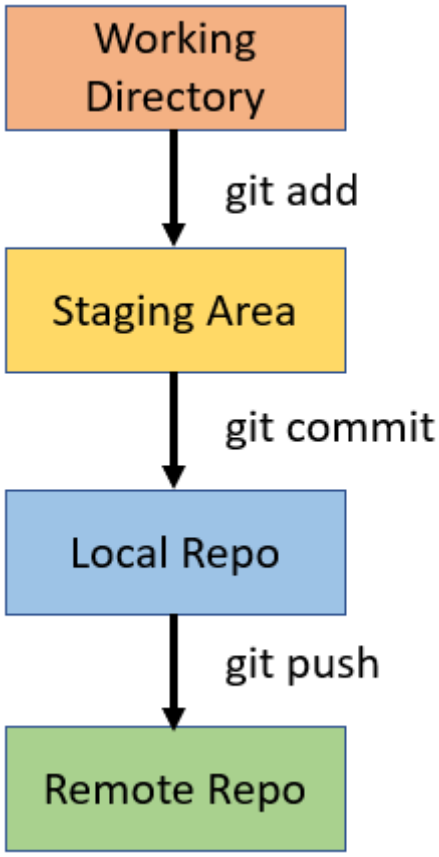
Once you clone the remote repository to local successfully, you can navigate to the directory through `cd <REPOSITORY_NAME>`:

```
lfeng@lfeng-wins-vm MINGW64 ~/Documents
$ cd git-and-gitlab-quick-start-tutorial/

lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (master)
$
```

Now, you are inside your working directory, and you can make any changes you like. Once you are happy with the changes, you can `add` and `commit` them to the local repository and then `push` to the remote repository so that your fellow team members can see the changes you made.

Please note that you are in the default **master** branch and all the changes you made are within this branch. And when you push the changes to the remote repo, it will update the **master** branch there. Usually, this is not the best practice.



Best Practices for Code Review

Excellent code depends on rigorous review. The following flow might be the best practices for code review and team collaboration:

- Once checks out a repository, a developer creates a new feature **branch**, makes changes in this feature branch, and tests it.
- When happy on the changes, the developer **push**es the changes in the **new branch**, and make a **merge request**.
- The developer assigns the **merge request** to a **reviewer**, who looks at it and makes comments as appropriate. When the reviewer finish, he/she can assign it back to the author.
- The author **addresses the comments**. This stage can go around for a while, but once both reviewer and author are happy, they can approve a merge or assign it to a final reviewer who can do the merge.
- The final reviewer follows the same reviewing process again. The author again addresses any comments. Once the final reviewer is happy and the build is green, then the new branch will be **merged** with the target branch.

Pull from Remote

Before you make any changes, it is always a good idea to run the command `git pull` to make sure your current local repository is up to date.

```
lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (master)
$ git pull
Already up to date.
```

Create a New Branch

Branches let you work on new features or bug fixes of the main project code in the `master` branch. You can use `git branch <BRANCH_NAME>` to create a new branch and `git branch` to check your branches.

```
lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (master)
$ git branch lei_test

lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (master)
$ git branch
  lei_test
* master
```

Note the asterisk next to the `master`, which indicates that you are currently in the `master` branch. To switch to your new branch, you need to run `git checkout <BRANCH_NAME>`.

```
lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (master)
$ git checkout lei_test
Switched to branch 'lei_test'

lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (lei_test)
$ git branch
* lei_test
  master

lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (lei_test)
$
```

Commit Local Changes

Now, you are in the new feature branch and ready to change your codes.

git status

Once you finish your changes, you can run `git staus` to check the changes you have made.

```
lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (lei_test)
$ git status
On branch lei_test
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        git-and-gitlab-quick-start-tutorial.md
        images/

no changes added to commit (use "git add" and/or "git commit -a")
```

git add

You can run the command `git add <FILE_NAME>` (add the specific file) or `git add .` (add all files) to add your changes to the staging area. A staging area is an intermediate place between your working directory and local Git repo where any changes that you've made can be reviewed before you actually commit them to the repo.

```
lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (lei_test)
$ git add .

lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (lei_test)
$ git status
On branch lei_test
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
        new file:   git-and-gitlab-quick-start-tutorial.md
        new file:   images/gitlab01.png
        new file:   images/gitlab02.png
        new file:   images/gitlab03.png
        new file:   images/gitlab04.png
        new file:   images/gitlab05.png
        new file:   images/gitlab06.png
        new file:   images/gitlab07.png
        new file:   images/gitlab08.png
        new file:   images/gitlab09.png
        new file:   images/gitlab10.png
        new file:   images/gitlab11.png
```

git commit

Next, you can commit the changes to your local repo through running `git commit -m "MESSAGE_TEXT"`.

```
lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (lei_test)
$ git commit -m "updated the tutorials"
[lei_test 960ee6b] updated the tutorials
 13 files changed, 76 insertions(+), 1 deletion(-)
 create mode 100644 git-and-gitlab-quick-start-tutorial.md
 create mode 100644 images/gitlab01.png
 create mode 100644 images/gitlab02.png
 create mode 100644 images/gitlab03.png
 create mode 100644 images/gitlab04.png
 create mode 100644 images/gitlab05.png
 create mode 100644 images/gitlab06.png
 create mode 100644 images/gitlab07.png
 create mode 100644 images/gitlab08.png
 create mode 100644 images/gitlab09.png
 create mode 100644 images/gitlab10.png
 create mode 100644 images/gitlab11.png
```

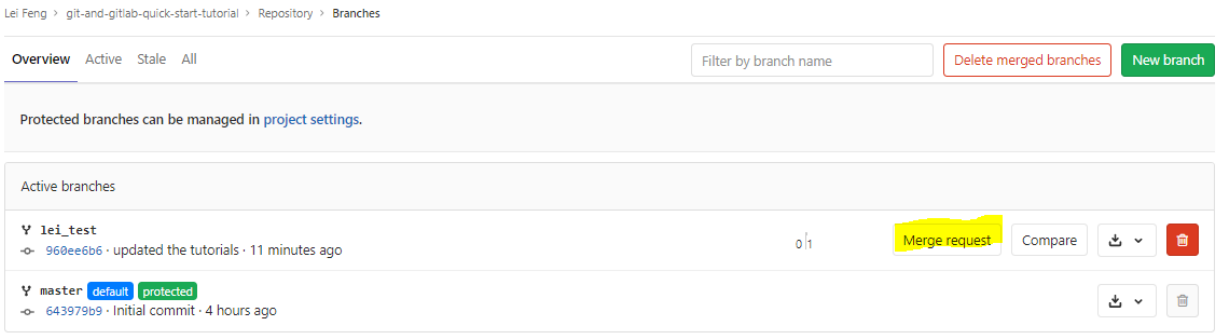
git push

After committing to the local repo, you can push the commits from the local to the new branch in the remote repo through running `git push -u origin <BRANCH_NAME>`.

```
lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (lei_test)
$ git push -u origin lei_test
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 2 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (16/16), 392.17 KiB | 5.30 MiB/s, done.
Total 16 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: To create a merge request for lei_test, visit:
remote:   https://gitlab.prod.fedex.com/5004756/git-and-gitlab-quick-start-tutorial/-/merge_requests/new?merge_r
equest%5Bsource_branch%5D=lei_test
remote:
To gitlab.prod.fedex.com:5004756/git-and-gitlab-quick-start-tutorial.git
 * [new branch]      lei_test -> lei_test
Branch 'lei_test' set up to track remote branch 'lei_test' from 'origin'.
```

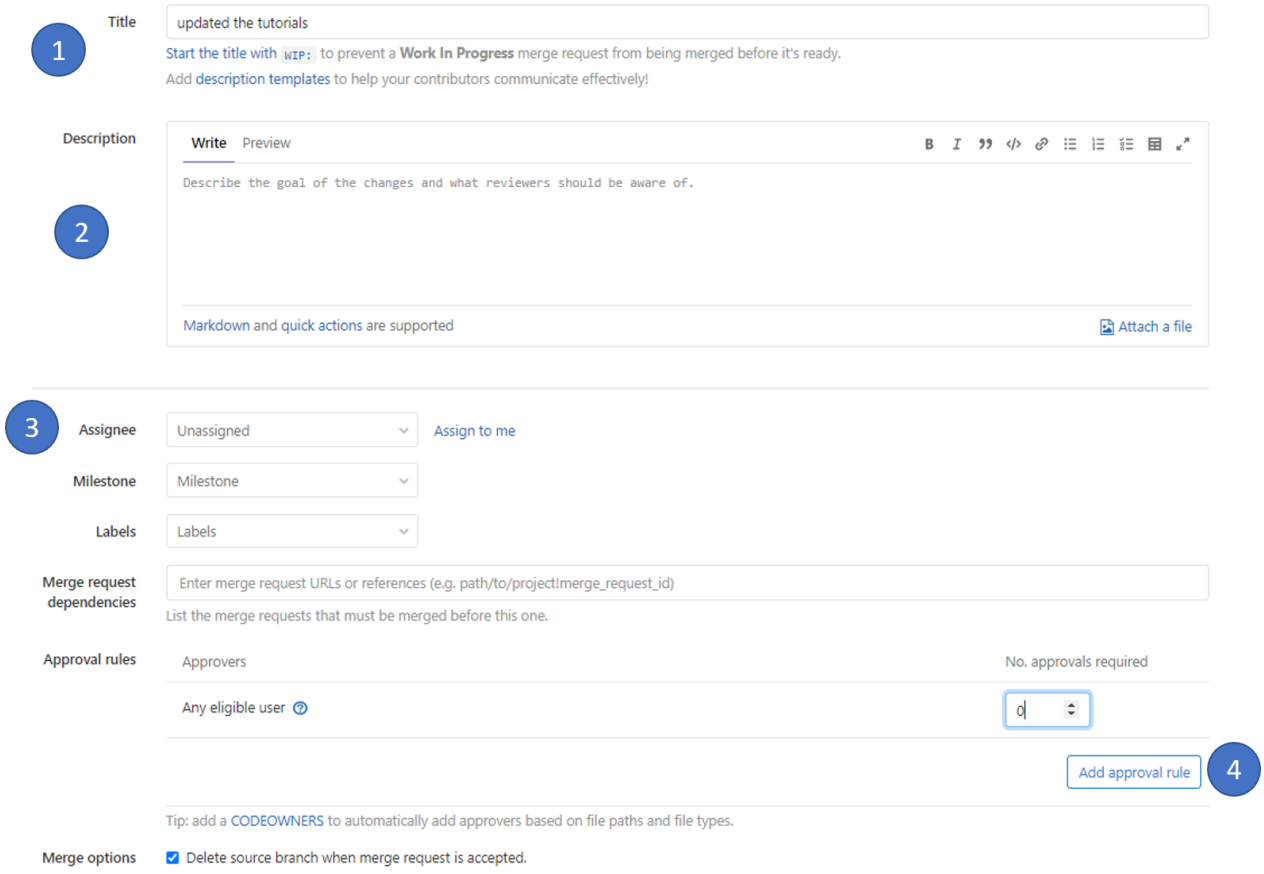
merge request

Once you push the local to the new branch in the remote repository, you can submit a **merger request** through the URL in the above command or from **Project** –>**Branches** on the GitLab page.



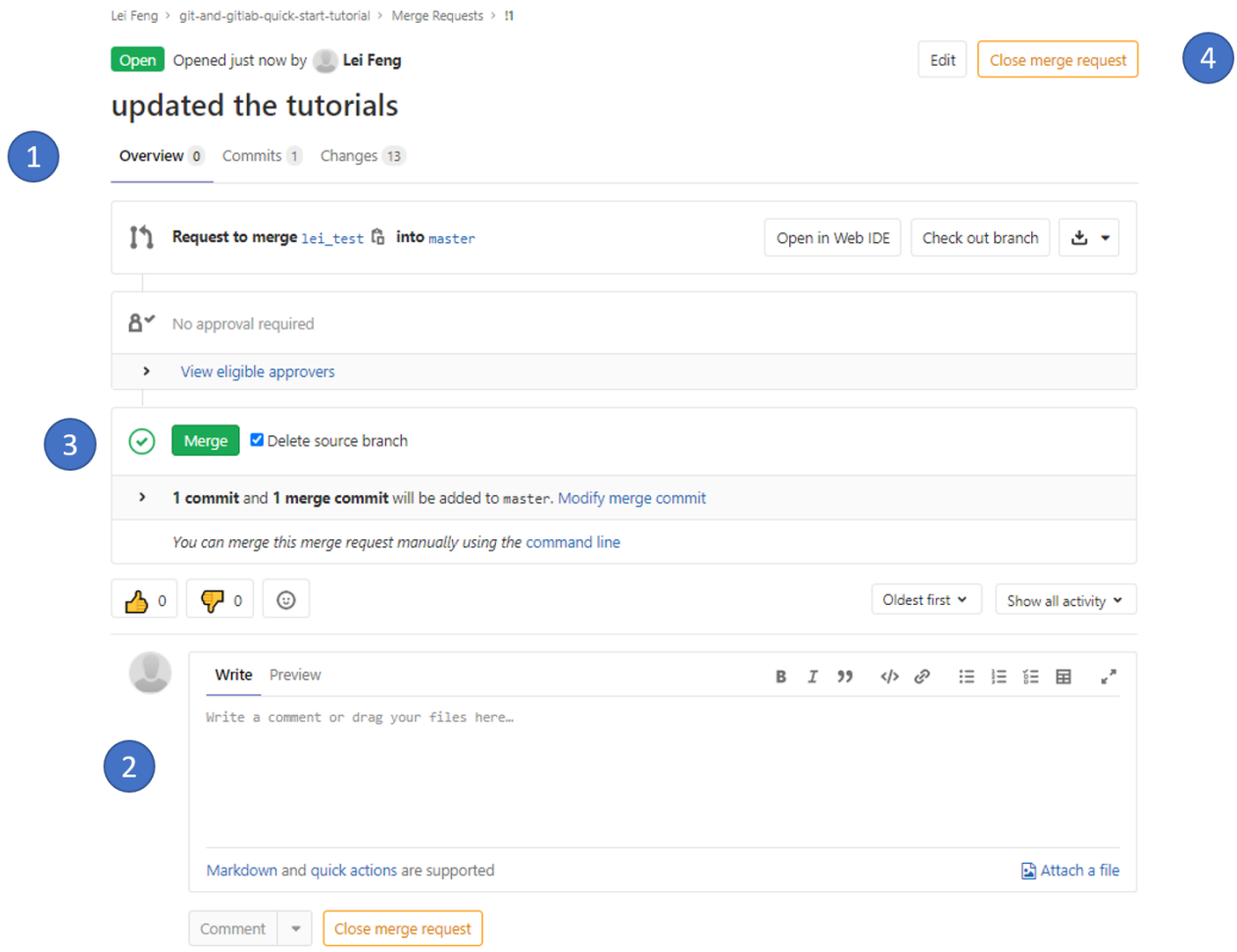
The **Merge Request** button will pop up the **New Merge Request** page, where you can:

1. fill the title of the request;
2. write a brief description;
3. assign reviewer(s), Milestone and Labels;
4. set up approval rules.



Once the merge request has been submitted, the reviewer(s) will get notification and they can go the request page to do:

1. check the commits and changes;
2. write comments;
3. approve the merge (if applicable);
4. close the merge request (if applicable).



delete local branch

Once your new branch has been merged to the target branch in the remote repository, you can delete your local branch by running `git branch delete -d <BRANCH_NAME>`. Note, you have to get out of the branch before you run this command deleting it.

```
lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (lei_test)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

lfeng@lfeng-wins-vm MINGW64 ~/Documents/git-and-gitlab-quick-start-tutorial (master)
$ git branch -d lei_test
warning: deleting branch 'lei_test' that has been merged to
         'refs/remotes/origin/lei_test', but not yet merged to HEAD.
Deleted branch lei_test (was 960ee6b).
```

pull from remote

Now, it will be a good practice to run `git pull` again to synchronize your remote repository to your local.

I hope this quick-start tutorial is helpful to you. Please feel free to leave any comments and advice. Thanks for reading.

References

- https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html
- https://about.gitlab.com/images/press/git-cheat-sheet.pdf
- https://about.gitlab.com/blog/2017/03/17/demo-mastering-code-review-with-gitlab/
- GitLab Get Started: https://about.gitlab.com/get-started/
- GitLab Learn: https://about.gitlab.com/learn/
- https://git-scm.com/book/en/v2
