Introduction to gitlab.dkrz.de

Tutorial, Hamburg 2017

Hendryk Bockelmann

DKRZ

Part I

GitLab Basics

Lab 1 gitlab.dkrz.de server

Goal: **Get to know the DKRZ GitLab server.**

What is GitLab?

- GitLab provides a web user interface to using the core Git software
- A user can share his/her software repository created and maintained by Git through GitLab
- gitlab.dkrz.de is the GitLab platform provided by DKRZ to manage your git-repositories

A general workflow for a user to use DKRZ's GitLab is as follows

- Log into the GitLab at DKRZ (`https://gitlab.dkrz.de`) using one's DKRZ account (LDAP)
- Create a project (or projects) in the GitLab
- *Pull* the project from GitLab to user's Laptop or on mistral
- Develop the software and commit the revisions using the commit command in Git
- *Push* the project revisions to the GitLab

Lab 2 A sample GitLab project

Goal: Creating a first GitLab project.

01 Create an ssh key, if you haven't done so.

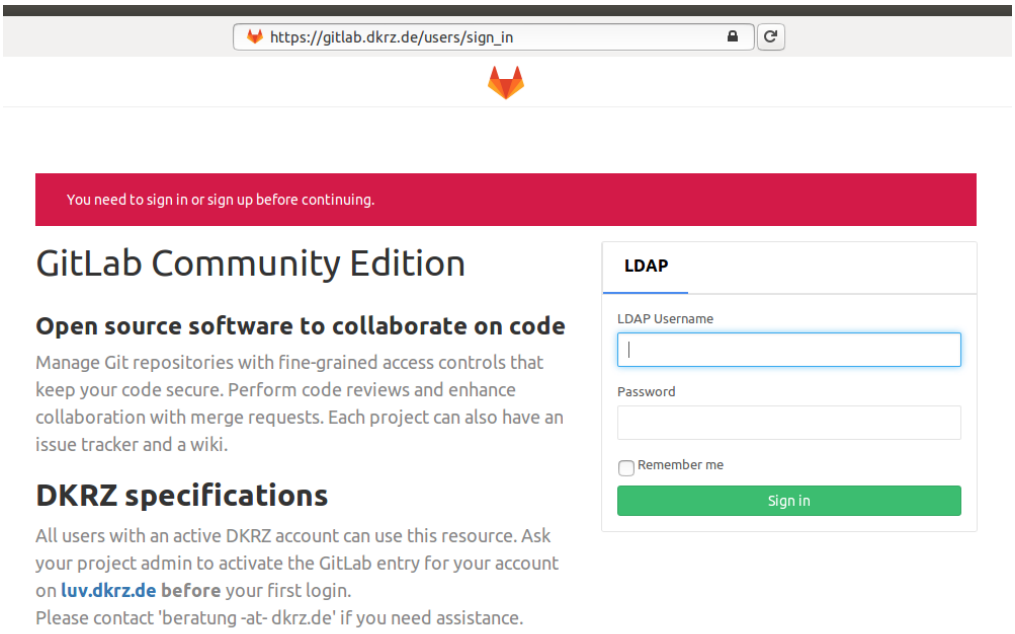
Execute:

```
ssh-keygen -t rsa -b 4096
ls ~/.ssh
```

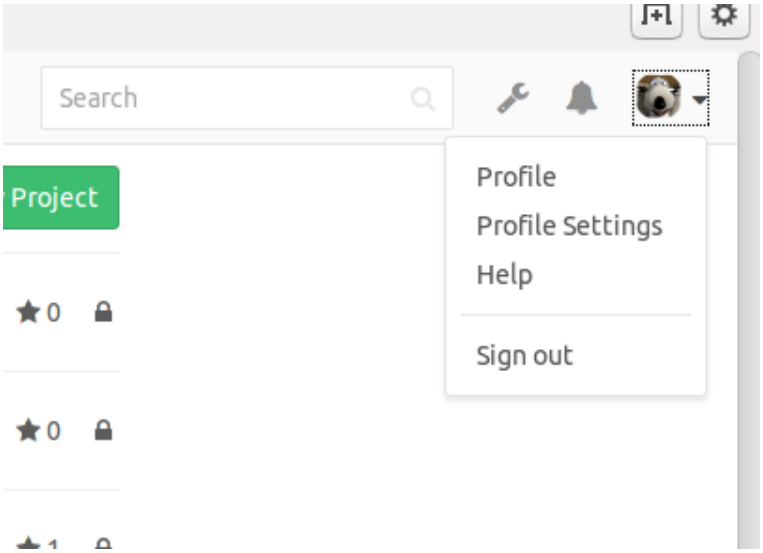
Output:

```
authorized_keys  config  id_rsa  id_rsa.pub  known_hosts
```

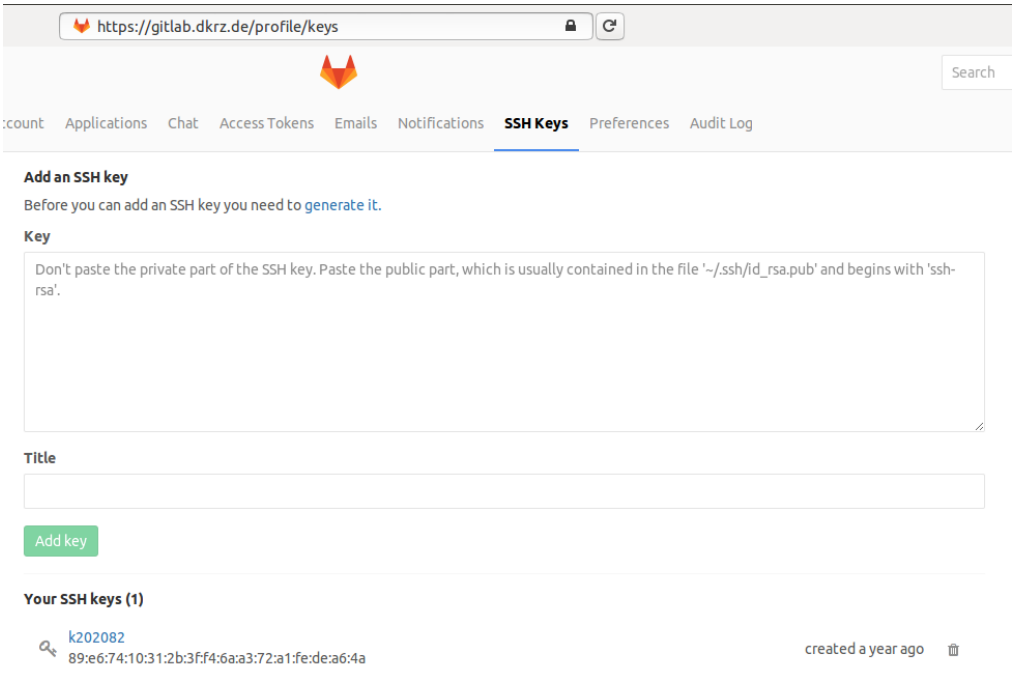
02 Open a web browser and log into DKRZ GitLab



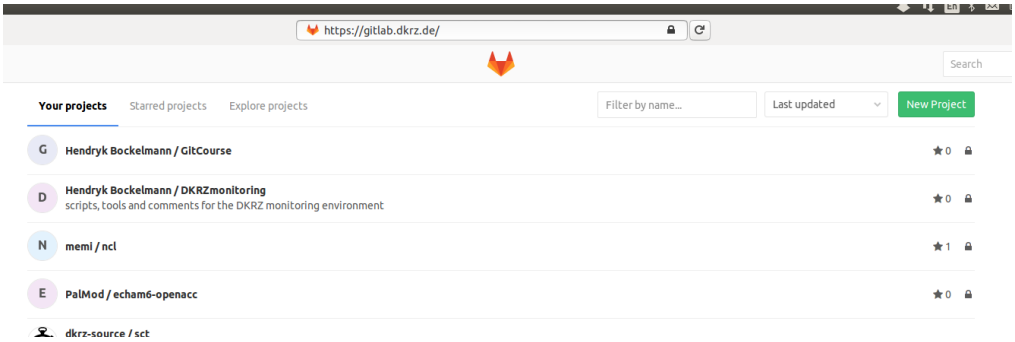
03 Click the person icon at the top menu bar to open up your 'Profile settings' section



04 **Add your public SSH key via copy'n'paste**

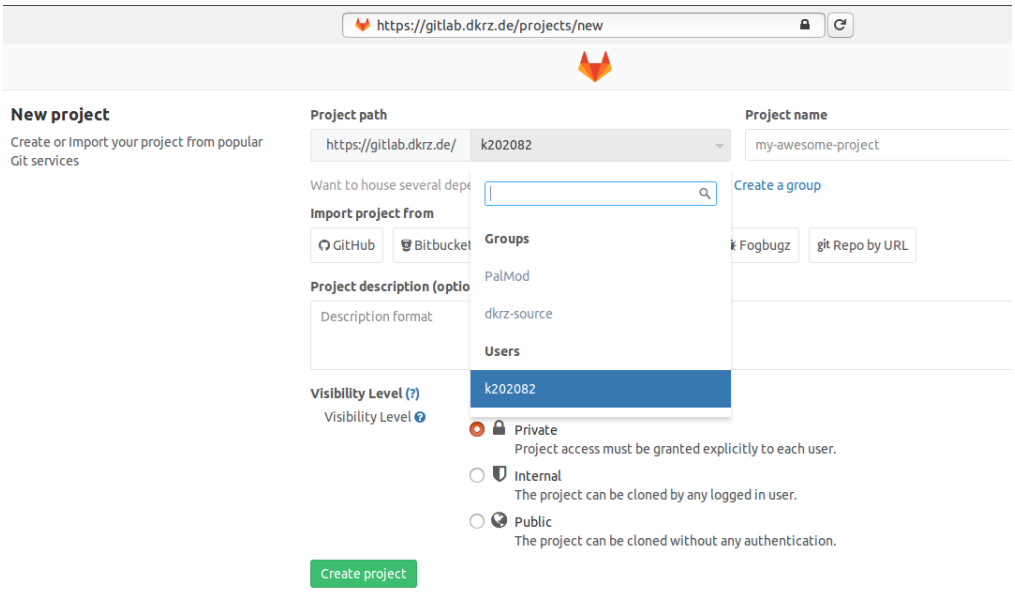


05 **Go back to the dashboard by clicking on the raccoon**

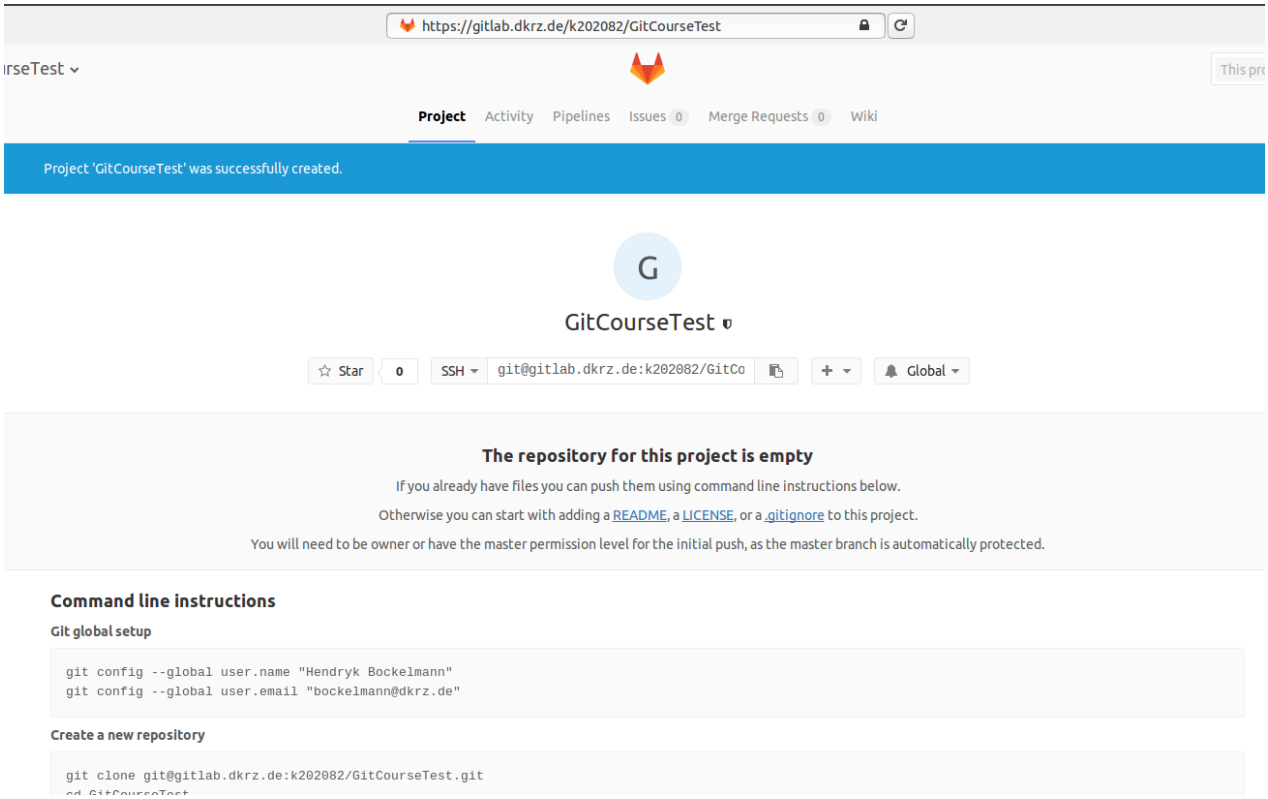


By default all users are given the rights to create 5 projects. If you require more than this please contact DKRZ to set a personal project limit.

06 Create a new project



Give your project a name and an optional description stating the purpose of the project. You can also select a visibility level for the project. By default the project is private meaning it is only visible to you. You can add members to the project at a later stage to allow them commit to your repository. It is advisable to make your project private.



Lab 3 A sample GitLab project, part 2

Goal: **Work with the remote repository on GitLab and a local repository.**

01 **Inspect the new project** The Gitlab project creation screen shows a set of commands you might use to create a local repository on your system (or mistral).

Execute these commands on your system first before taking further actions on Gitlab.

These steps are basically what you learn in any Git tutorial.

02 **Clone the new empty repository**

Execute:

```
git clone git@gitlab.dkrz.de:k202082/GitCourseTest.git
cd GitCourseTest
echo "This is a test" > README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Output:

```
Cloning into 'GitCourseTest'...
warning: You appear to have cloned an empty repository.
Checking connectivity... done.

[master (root-commit) b6795c1] add README
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

Counting objects: 3, done.
Writing objects: 100% (3/3), 230 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@gitlab.dkrz.de:k202082/GitCourseTest.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

03 **Check the remote**

Execute:

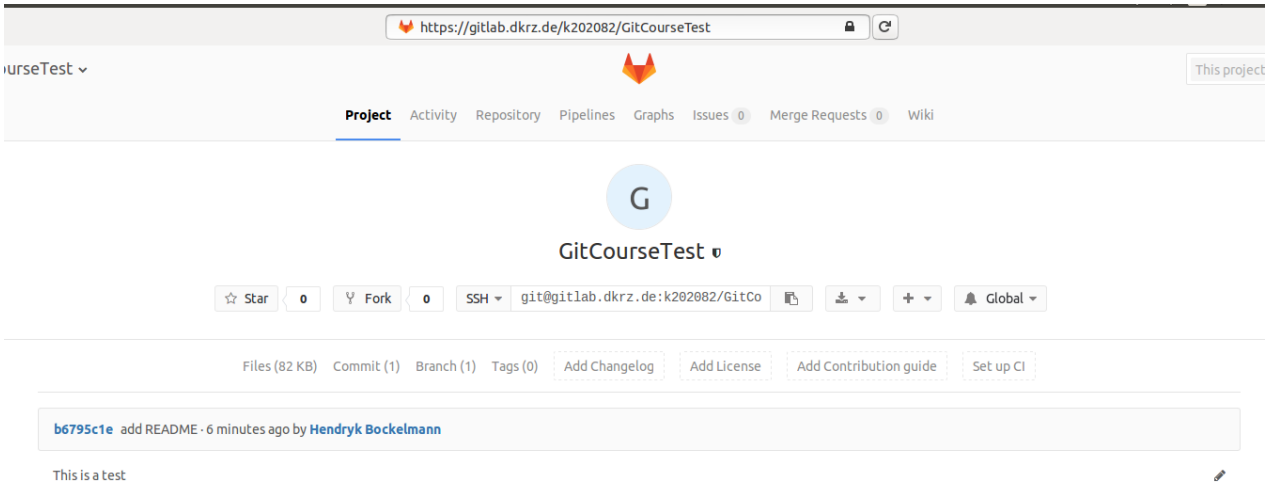
```
git remote show origin
```

Remark: the output URL obviously varies since you are using your personal/group project

Output:

```
* remote origin
  Fetch URL: git@gitlab.dkrz.de:k202082/GitCourseTest.git
  Push  URL: git@gitlab.dkrz.de:k202082/GitCourseTest.git
  HEAD branch: master
  Remote branch:
    master tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)
```

04 Verify the status in GitLab (reload web browser)



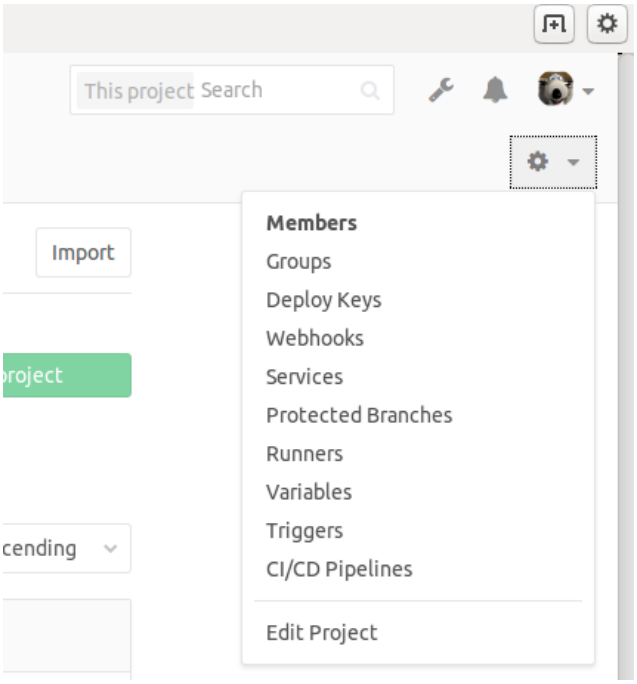
Lab 4 Edit the GitLab project

Goal: **Modify settings of GitLab projects, learn about GitLab features.**

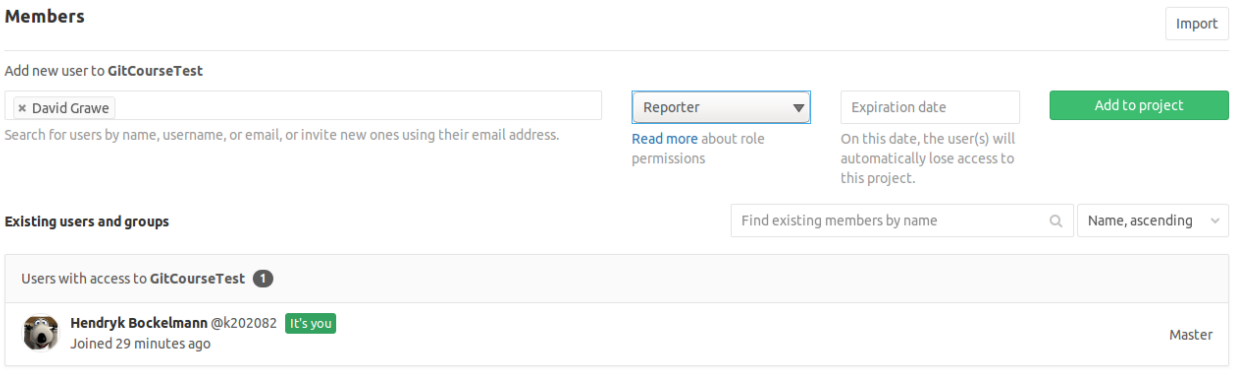
Assuming that you already created a private project, it is now time to invite other users

01 Share the project

On the web interface click the gear icon and 'Members'



02 Find a colleague to add



Some remarks on project visibility

Public projects

- can be cloned without any authentication of the user
- will be listed on the public access directory
- any logged in user will have Guest permissions on the repository (create new issues, leave comments, view wiki pages)

Internal projects

- can be cloned by any logged-in user
- will be listed on the public access directory for logged in users
- any logged in user will have Guest permissions on the repository

Private projects

- can be cloned only with explicitly granted access
- will not be listed on the public access directory
- each user can have dedicated access rights

Lab 5 Import existing repository

Goal: **Import an already existing local repository to GitLab.**

If you already have a local Git repository, you might add this to a GitLab project.

01 **Find your local repository**

Execute:

```
cd hello
git hist
git status
```

Output:

```
* 7d50c43 2016-12-27 | Changed README in original repo (HEAD, shared/master, ➥
    ➥ master) [Hendryk Bockelmann]
* 5c3f9b1 2016-12-27 | Add shared comment to README. [Hendryk Bockelmann]
* f068ab3 2016-12-27 | Updated Makefile. (greet) [Hendryk Bockelmann]
* 73dda04 2016-12-27 | Hello uses Greeter. [Hendryk Bockelmann]
* c1c85a2 2016-12-27 | Added type/greeter class. [Hendryk Bockelmann]
* f515b26 2016-12-27 | Made interactive [Hendryk Bockelmann]
* 17acd42 2016-12-27 | Added README. [Hendryk Bockelmann]
* 90d87c3 2016-12-27 | Added .gitignore. [Hendryk Bockelmann]
* d8e5a4f 2016-12-27 | Added a Makefile. [Hendryk Bockelmann]
...

# On branch master
nothing to commit (working directory clean)
```

02 **Check all existing remote repositories and eventually remove them**

Execute:

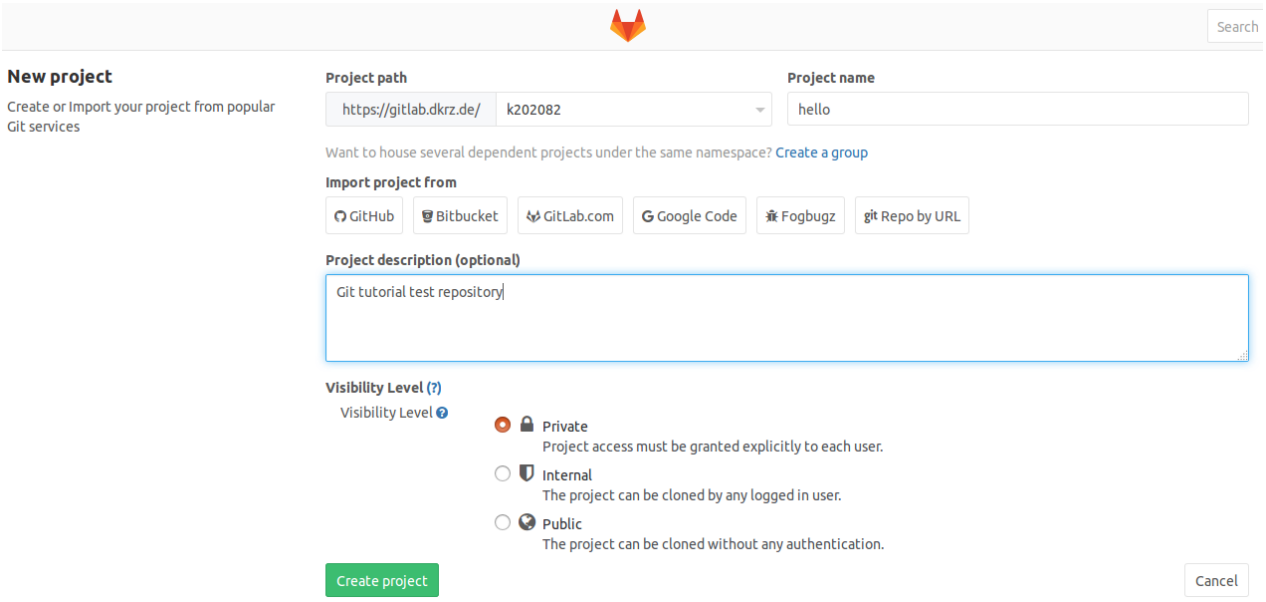
```
git remote -v
git remote rm shared
git branch -a
```

Output:

```
shared  ../hello.git (fetch)
shared  ../hello.git (push)

  greet
* master
```

03 **Create a new GitLab project**



04 Add new remote to your local repository

Execute:

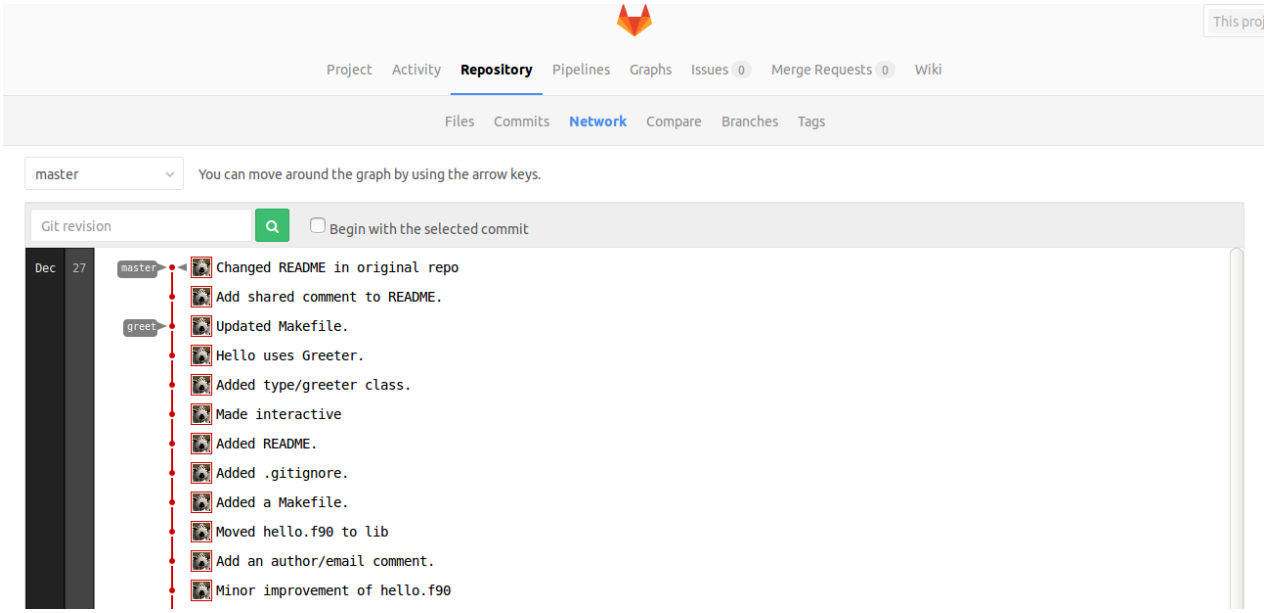
```
git remote add origin git@gitlab.dkrz.de:k202082/hello.git
git status
git push --all -u origin
```

Output:

```
# On branch master
nothing to commit (working directory clean)

Counting objects: 54, done.
Delta compression using up to 48 threads.
Compressing objects: 100% (43/43), done.
Writing objects: 100% (54/54), 5.25 KiB, done.
Total 54 (delta 14), reused 0 (delta 0)
remote:
remote:To create a merge request for greet, visit:
remote:https://gitlab.dkrz.de/k202082/hello/merge_requests/new?merge_request
remote:
To git@gitlab.dkrz.de:k202082/hello.git
 * [new branch]      greet -> greet
 * [new branch]      master -> master
Branch greet set up to track remote branch greet from origin.
Branch master set up to track remote branch master from origin.
```

05 Check the GitLab project



Now we are already able to check out some nice features of GitLab, we can take a look at the 'Repository' section with:

- Files: a file browser for your repository
- Commits: a list of the commits of the chosen branch
- Network: a graph visualizing your repository
- Branches: a list of all your branches
- Tags: all tagged points

By default, the git push command doesn't transfer tags to remote servers. You will have to explicitly push tags to a shared server after you have created them.

06 Add tags to GitLab

Execute:

```
git push --tags origin
```

Output:

```
Total 0 (delta 0), reused 0 (delta 0)
To git@gitlab.dkrz.de:k202082/hello.git
 * [new tag]         v1 -> v1
 * [new tag]         v1-beta -> v1-beta
```

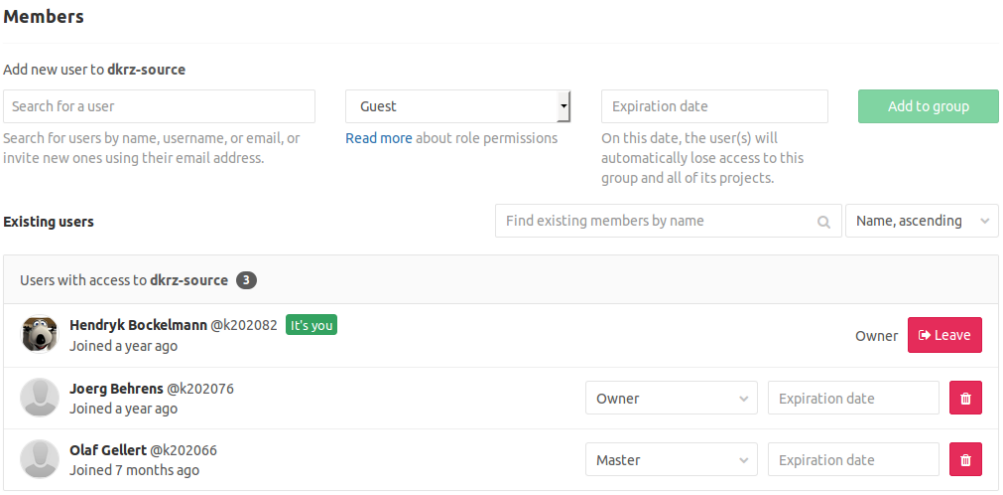
Part II

GitLab Additions

Groups ...

- can be created by any logged-in user
- are public visible by default
- serve as 'container' for individual users
- provide ability to grant role permissions per user

Group's projects inherit the permissions for all the users.



Roles

Users have different abilities depending on the access level they have in a particular group or project. If a user is both in a group's project and the project itself, the highest permission level is used.

- Guest: create issues, leave comments, view wiki
- Reporter: + pull code, manage issue tracker
- Developer: + manage/accept/create merge requests, create new branches, write wiki
- Master: + edit project, add team members
- Owner: + switch visibility level, remove project

Other Stuff

There are many more features in GitLab that might be used

- Wiki
- Issue tracking
- Continuous integration (pipelines)
- Merge requests

The last topic might be of interest and will be explained further.

Merge requests

Merge requests are useful to integrate separate changes that were made to a project, on different branches.

There are two main ways to have a merge request workflow:

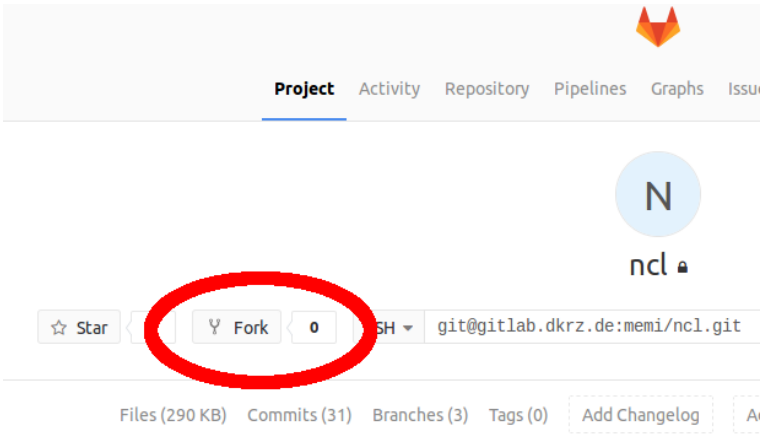
1. Protected main branches in a single GitLab project
2. Forking of the main project

Protected main branches in a single GitLab project

- everybody works within the same GitLab project
- project maintainers get Master access and the regular developers get Developer access
- maintainers mark the main branches as 'Protected'
- developers push feature branches to the project and create merge requests to have their feature branches reviewed and merged into one of the protected branches
- by default, only users with Master access can merge changes into a protected branch

Forking of the main project

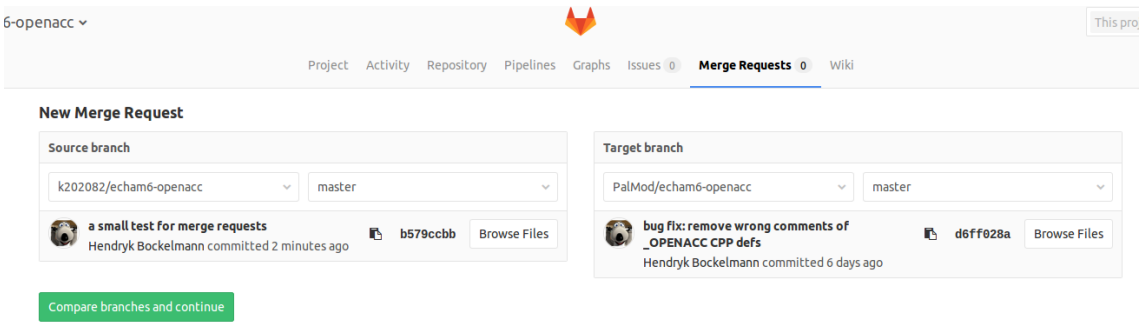
- maintainers get Master access, regular developers get Reporter access to main repository, which prohibits them from pushing any changes to it
- developers create forks of the main project and push their feature branches to their own forks
- to get their changes into main they need to create a merge request across forks



After a developer finalized and pushed his changes to his own (forked) project or to his own branch, a Merge request can be issued.

- go to the project from where you'd like to merge your changes and click on the Merge Requests tab
- click on New Merge Request

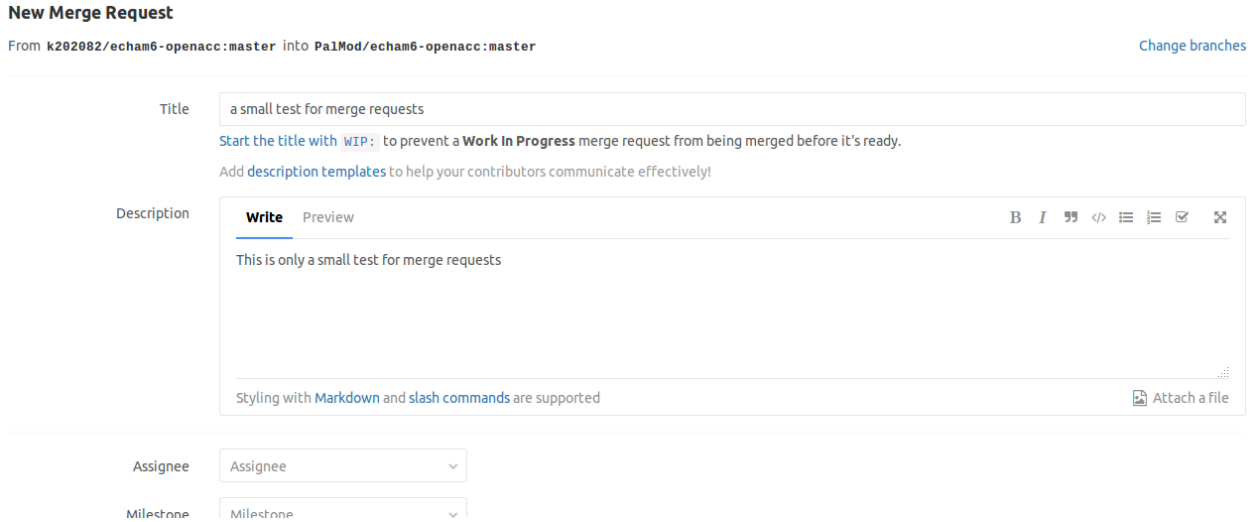
- select a source branch and click on the Compare branches and continue button



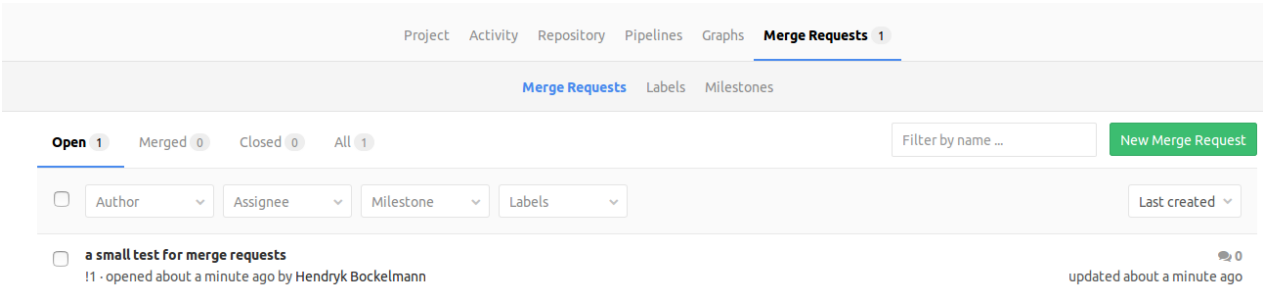
Add a title and a description to your merge request.

Optionally, select a user to review your merge request and to accept or close it.

Finally, click on the Submit merge request button.



The main project now has one open Merge Request.



The maintainer (having master access to main project) and other reporter can now discuss this request and finally accept or close it.