GIT TUTORIAL FOR IAG0582

Updated: 09.2017

Disclaimer

This chapter will point out the parts which may change in time and are essentially variables whenever they occur in text / images. These variables will be surrounded by brackets ***[*** and ***]***.
Gitlab version is 9.4.5 at the last edit of this document.
NB! The sample values used in this document should be replaced in your case.

Variable	**Description**
Uni-ID	This is username which is used by multiple services in TTU e.g. gitlab. Earlier versions of this were in the following format: ***firstname.surname***. Newer version of Uni-ID has fixed length of 6 e.g. ***xxyyyy*** where ***xx*** is 2 first characters from first name and ***yyyy*** first 4 characters from surname. Some characters may be replaced e.g. Estonian ‘ü’ -> ‘u’. In this document Uni-ID will have a value of ***josmit*** (Derived from John Smith).
repository	Repository in gitlab is ‘Project’ name. This may vary from course to course meaning that it will be specified by the lecturers. For example in the Programming I course the name should be course code all in lowercase letters e.g. ***iag0581*** (for the older courses - 2016 and older) or ***iax0583*** (for 2017 and newer). In this document project name ***test*** is used.

Installing git

This chapter will describe how to set up your system for git usage.

Windows

Windows installation will ask you multiple things, just leave the default settings as is and click “next” until you can click “finish”. Then click finish and you’re done.
For a video tutorial one can follow this : https://www.youtube.com/watch?v=albr1o7Z1nw
Aforementioned tutorial also shows how to do basic git configuration (this will be covered in next chapters).

Other platforms

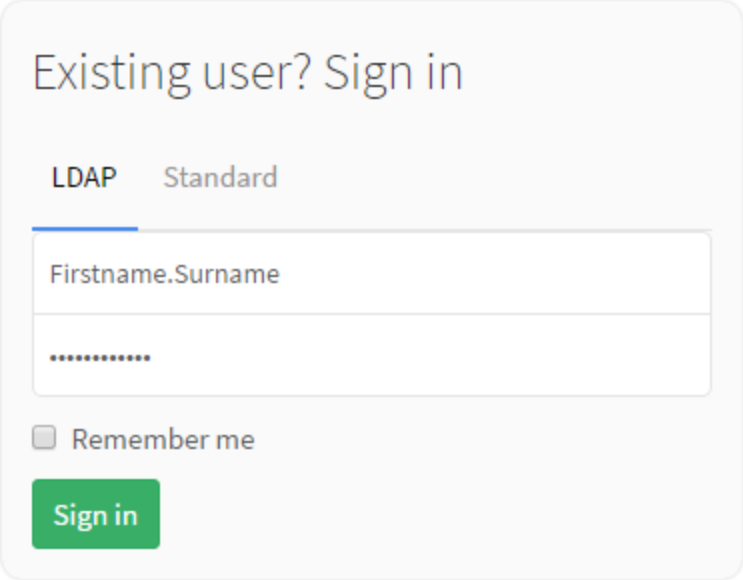
In linux and mac you can verify if you have git by typing “git --version” into terminal.

```
root@kali:~# git --version
git version 2.9.3
```

If you don’t have git then follow the tutorials given here:
https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Gitlab login

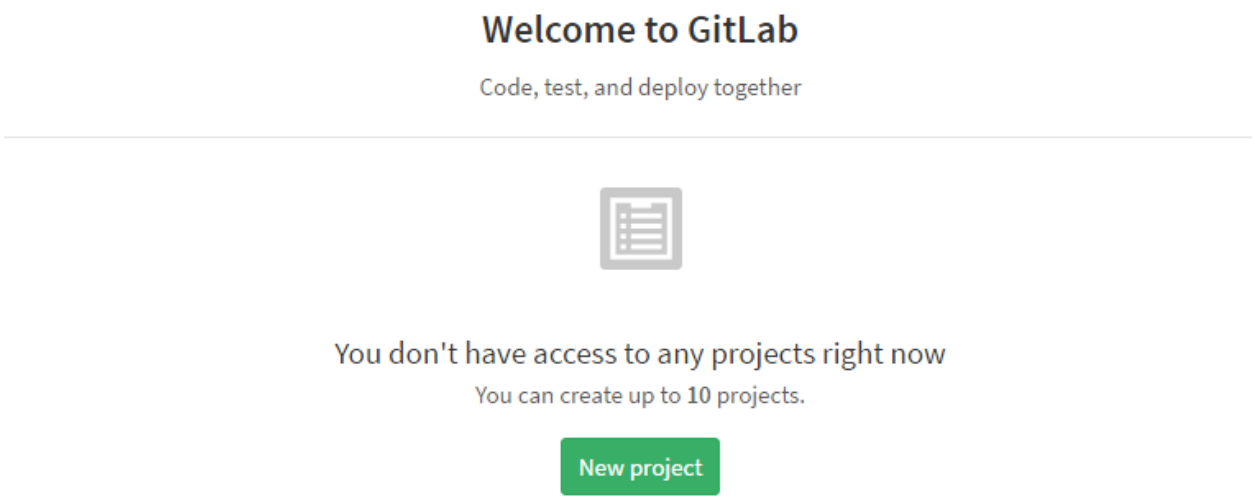
For you to create git project you need to log into gitlab (http://gitlab.ati.ttu.ee/). To log in with your **Uni-ID** make sure you have **LDAP** selected.



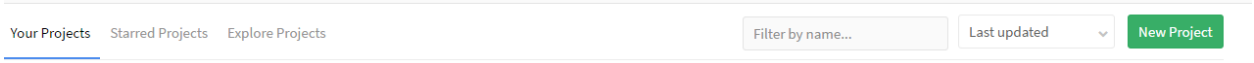
If you don’t have **Uni-ID** log into https://pass.ttu.ee/ with your ID card and set up your Uni-ID.

Create project

To create a new project one should navigate to http://gitlab.ati.ttu.ee/dashboard/projects and click green “New project” button. If you don’t have any previous projects you should see this selection in the center of the page as shown on the figure.

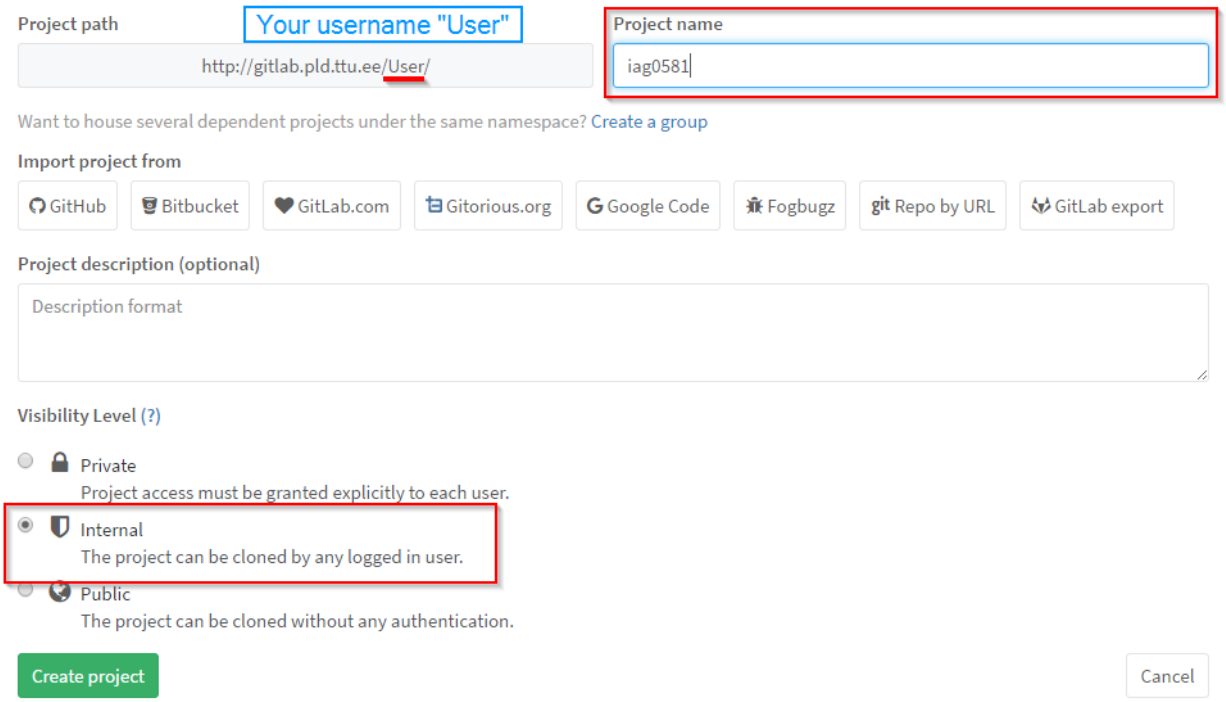


If you already have created a project you find the button on the right side of the page as seen on the figure.

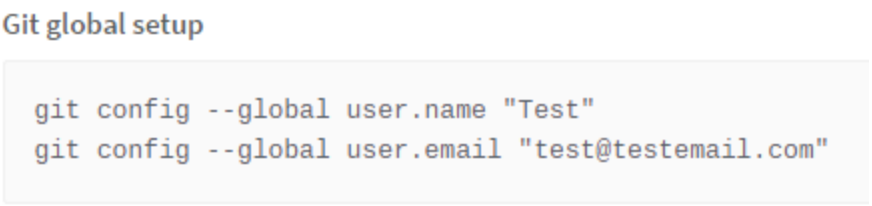


When creating new project make sure you :

- Name it as ***[repository]***.
- Set visibility as **Internal**

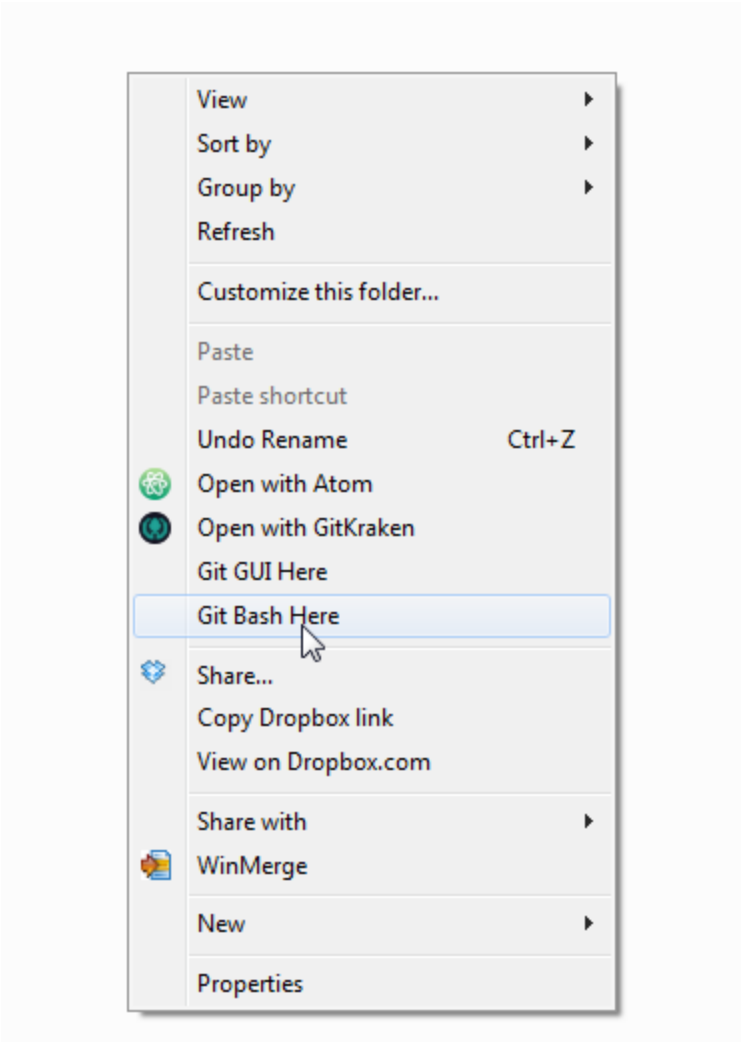


Next you shall be forwarded to the front page of your project. This is the url you should give us (Your project URL), in this example it is http://gitlab.ati.ttu.ee/josmit/test . In your project page under the "**Git global setup**" section you find the necessary commands for git configuration. These are the commands you should enter in the next chapter.



Configure git

To configure git open terminal window in the directory you wish to put your project in. In computer class this should be on your **P drive**!
Windows users should go to directory you wish to save your project in and right click in the folder and click "Git Bash Here"



For basic configuration type in the commands shown in your Git global setup. For user.name you should enter your full name and for user.email the email you used to create an account. If you used LDAP for login, your @ttu.ee email is used. **It is very important that this info matches**. To verify your data go to your profile settings http://gitlab.ati.ttu.ee/profile .

```
root@kali:~# git config --global user.name "Test User"
root@kali:~# git config --global user.email "test@testemail.com"
```

Create SSH key (optional)

Using SSH key is recommended because it offers you a way to push / pull code without entering your credentials every time. One can think of it as a fingerprint.
To create an ssh key you must type the following (NB! Check that your spaces are correct!):

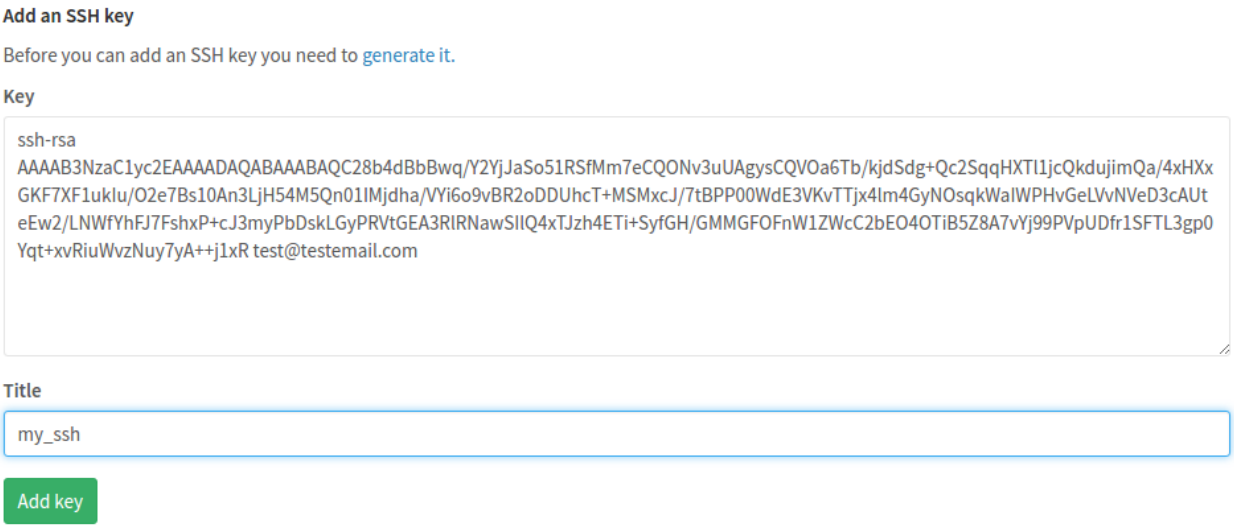
- ssh-keygen -t rsa -C “josmit@ttu.ee”
    - **Email should be the same as on the user.email setting!**

- * press enter *
    - Then the key will be saved in your home directory (~/.ssh/id_rsa.pub)
- * press enter *
    - If you type password here, you will always be prompted for password.
- * press enter *
- cat ~/.ssh/id_rsa.pub
    - This command will print the ssh public key on terminal.

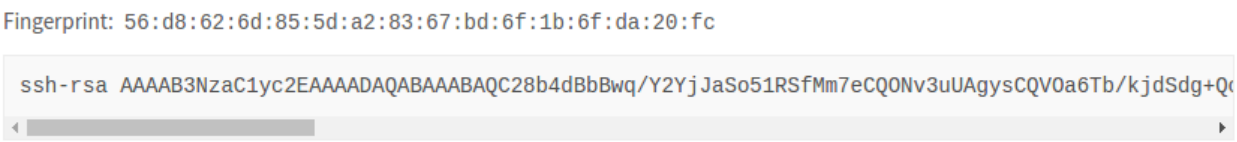
```
root@kali:~# ssh-keygen -t rsa -C "test@testemail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:LXVg+e0LoOiamzuC2hHMMVTPCHK7TJ1zs10KHwWU1HM test@testemail.com
The key's randomart image is:
+---[RSA 2048]----+
|. +..  o+++.     |
| + + =  .o+.E    |
|  = = * . o+..   |
| = + o * *... .  |
|  *   ..S... .   |
|   .  . ..  . .  |
| ..  .       . . |
|.....o.       .  |
|o ..*=           |
+----[SHA256]-----+
```

```
root@kali:~# cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC28b4dBbBwq/Y2YjJaSo51RSfMm7eCQONv3uUAgysC
QVOa6Tb/kjdSdg+Qc2SqqHXTl1jcQkdujimQa/4xHXxGKF7XF1ukIu/O2e7Bs10An3LjH54M5Qn01IMj
dha/VYi6o9vBR2oDDUhcT+MSMxcJ/7tBPP00WdE3VKvTTjx4lm4GyNOsqkWaIWPHvGeLVvNVeD3cAUte
Ew2/LNWfYhFJ7FshxP+cJ3myPbDskLGyPRVtGEA3RlRNawSIlQ4xTJzh4ETi+SyfGH/GMMGFOFnW1ZWc
C2bE040TiB5Z8A7vYj99PVpUDfr1SFTL3gp0Yqt+xvRiuWvzNuy7yA++j1xR test@testemail.com
```

Now you should add this key to your SSH keys in gitlab (http://gitlab.ati.ttu.ee/profile/keys).



Make sure that there is **no newline after the email address!** Then press “Add key”
You have now successfully added SSH key!

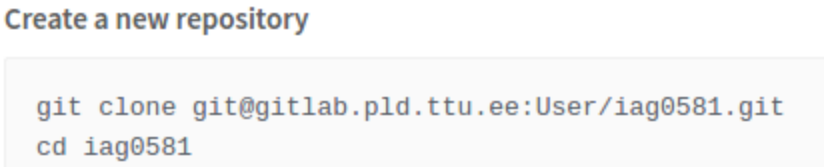


Cloning project

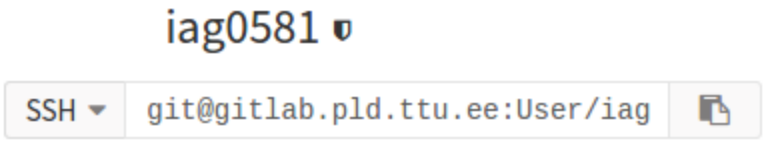
This chapter will describe how to clone using different methods. Using SSH you need to do the SSH key setup in previous chapter.

Cloning using SSH (preferred)

The first 2 commands used for this step are shown on the Create a new repository section in your project directory.



Then you should use the git clone command, in our example it would be “git clone git@gitlab.ati.ttu.ee:**[Uni-ID]**/***[repository]***.git”. You can get this URL in your project directory:



You shall be prompted with a question whether you wish to continue connecting to the server, you should type “yes” and then press enter.

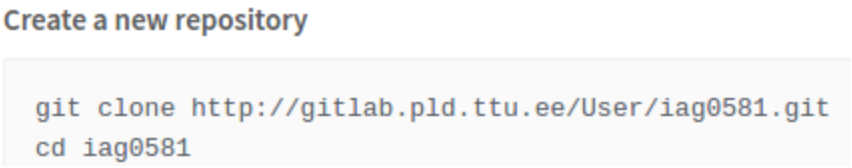
```
root@kali:~/Documents# git clone git@gitlab.pld.ttu.ee:User/iag0581.git
Cloning into 'iag0581'...
The authenticity of host 'gitlab.pld.ttu.ee (193.40.246.19)' can't be establishe
d.
ECDSA key fingerprint is SHA256:tGlLdy01Jl2jmdAIHv6kO+t0cW4Qqo8BnH/kDi6RYC8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'gitlab.pld.ttu.ee,193.40.246.19' (ECDSA) to the list
 of known hosts.
warning: You appear to have cloned an empty repository.
Checking connectivity... done.
```

After successfully cloning the repository you must go to that directory by typing “cd iag0581”

```
root@kali:~/Documents# cd iag0581/
root@kali:~/Documents/iag0581#
```

Cloning using HTTP

The first 2 commands used for this step are shown on the Create a new repository section in your project directory.



Then you should use the git clone command, in our example it would be “git clone http://gitlab.ati.ttu.ee/User/iag0581.git”. You can get this URL in your project directory:

You shall be prompted with Username and password. Your username could be found in the URL, in our example it is “User”. When you type your password **in linux password fields will not be filled with asterisks (*)!**

```
root@kali:~/Documents# git clone http://gitlab.pld.ttu.ee/User/iag0581.git
Cloning into 'iag0581'...
Username for 'http://gitlab.pld.ttu.ee': User
Password for 'http://User@gitlab.pld.ttu.ee':
warning: You appear to have cloned an empty repository.
Checking connectivity... done.
```

After successfully cloning the repository you must go to that directory by typing “cd iag0581”

```
root@kali:~/Documents# cd iag0581/
root@kali:~/Documents/iag0581#
```

Working process with git

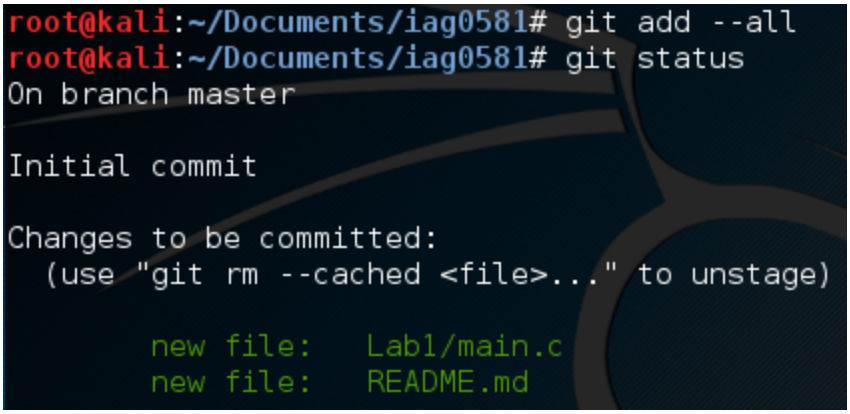
This chapter will describe the workflow with git. To use git in windows you must open the directory where your project is and right click in there and then press “Git Bash Here”.
To use these commands your working directory must be the project directory.

View new / modified / deleted files

To view changed files (modified / new / deleted) type “git status”.
New / untracked / modified files will be shown with red font.

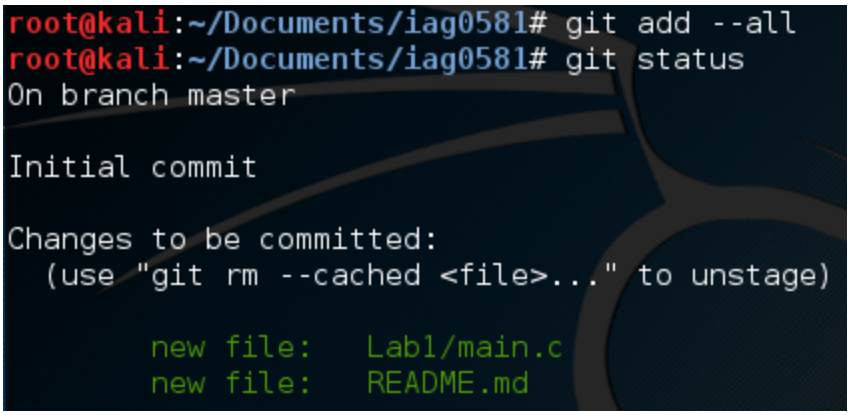
```
root@kali:~/Documents/iag0581# git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Lab1/
        README.md

nothing added to commit but untracked files present (use "git add" to track)
```

Staged files will be shown with green font.



Adding files to staging area

To add all new / modified / deleted files into staging area type “**git add --all**”. To add files to staging area means to save these files at current state. This enables users to view these files as they were in future.



Commiting staged files

To confirm the saving user must commit these changes by typing “**git commit -m “message”**”, where -m will indicate that message parameter will be added with this commit. By executing this command user will commit all staged files with the same message. User can also commit files seperately with different messages by specifying file: “**git commit README.md -m “Add README”**”
After typing the command the output should be something similar:

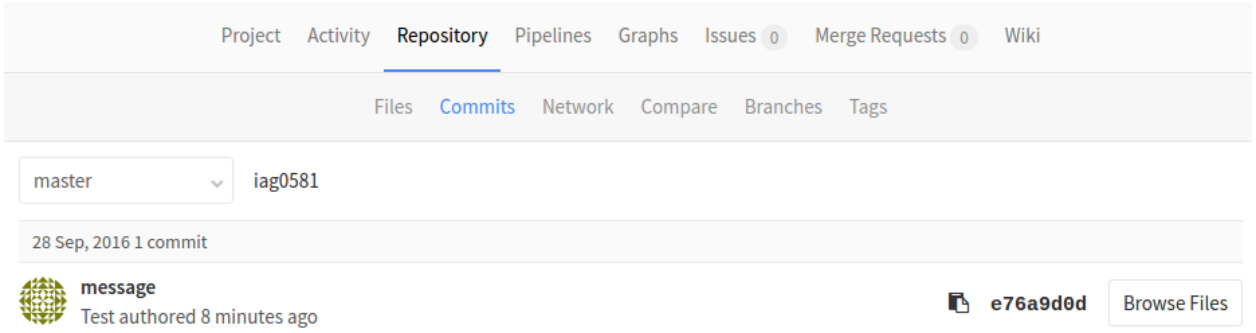
```
root@kali:~/Documents/iag0581# git commit -m "message"
[master (root-commit) e76a9d0] message
 2 files changed, 1 insertion(+)
 create mode 100644 Lab1/main.c
 create mode 100644 README.md
```

Pushing commits to gitlab server

This action is required to have the latest version of your files accessible in server. It is not necessary to do this after every commit however it is recommended to do after finishing your programming session.
To push changes into gitlab server type “**git push**”

```
root@kali:~/Documents/iag0581# git push
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 333 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To gitlab.pld.ttu.ee:/User/iag0581.git
 * [new branch]      master -> master
```

Now your files are pushed into server and the lecturers can access them and grade your homework / lab work.
All of the commits can be seen in your commits url at gitlab.ati.ttu.ee in our example the url is (http://gitlab.ati.ttu.ee/josmit/test/commits/master)



View commit history

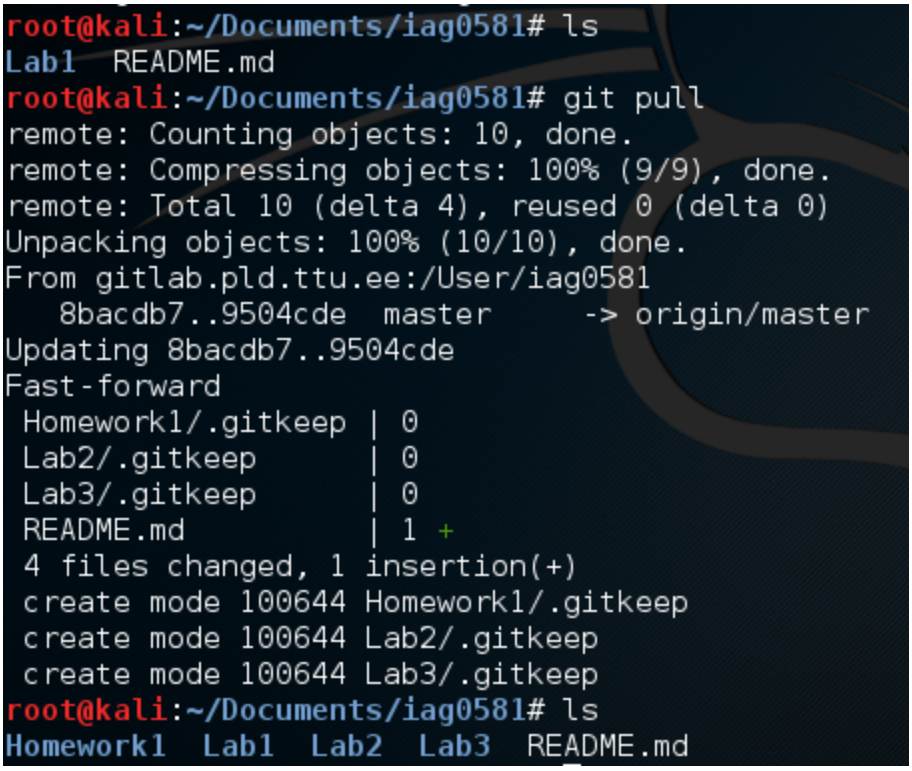
To see the commits made in this project one can type: “**git log**” or “**git log --pretty=oneline**”

```
root@kali:~/Documents/iag0581# git log
commit 8bacdb72ad36b0b2af4112b4cdfb6f6526b31eeb
Author: Test User <test@testemail.com>
Date:   Wed Sep 28 08:02:58 2016 -0400

    Modify last line, delete second line

commit 5fb960969d027e594149aba26bfca9cb839122e4
Author: Test User <test@testemail.com>
Date:   Wed Sep 28 08:02:08 2016 -0400

    Add two lines

commit e76a9d0d3c8c60173a05a4703a54bf3df4e45035
Author: Test User <test@testemail.com>
Date:   Wed Sep 28 07:52:52 2016 -0400

    message
```

```
root@kali:~/Documents/iag0581# git log --pretty=oneline
8bacdb72ad36b0b2af4112b4cdfb6f6526b31eeb Modify last line, delete second line
5fb960969d027e594149aba26bfca9cb839122e4 Add two lines
e76a9d0d3c8c60173a05a4703a54bf3df4e45035 message
```

Getting latest version from server

To get the latest version of files from server type “**git pull**”.



Note that ls shows the contents of current folder.

Special files

README.md

This file is usually added to the root directory of the repository. This file is a markdown file which is essentially a text file that supports styling. Project page will display the README.md file contents. See markdown cheatsheet and online markdown editor .

.gitignore

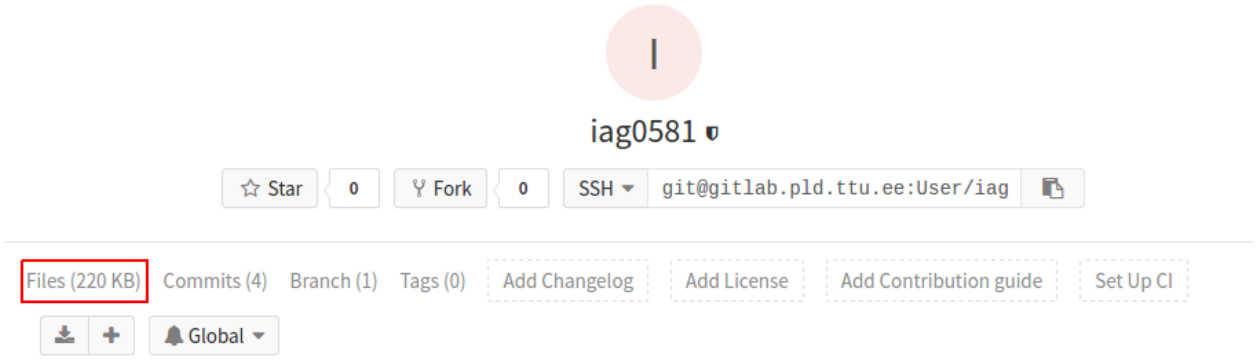
This file is added to the root directory of the repository. The contents of this file will dictate which files won't be tracked. It is a desired usage for binary files such as .exe etc.

Working process with Web interface

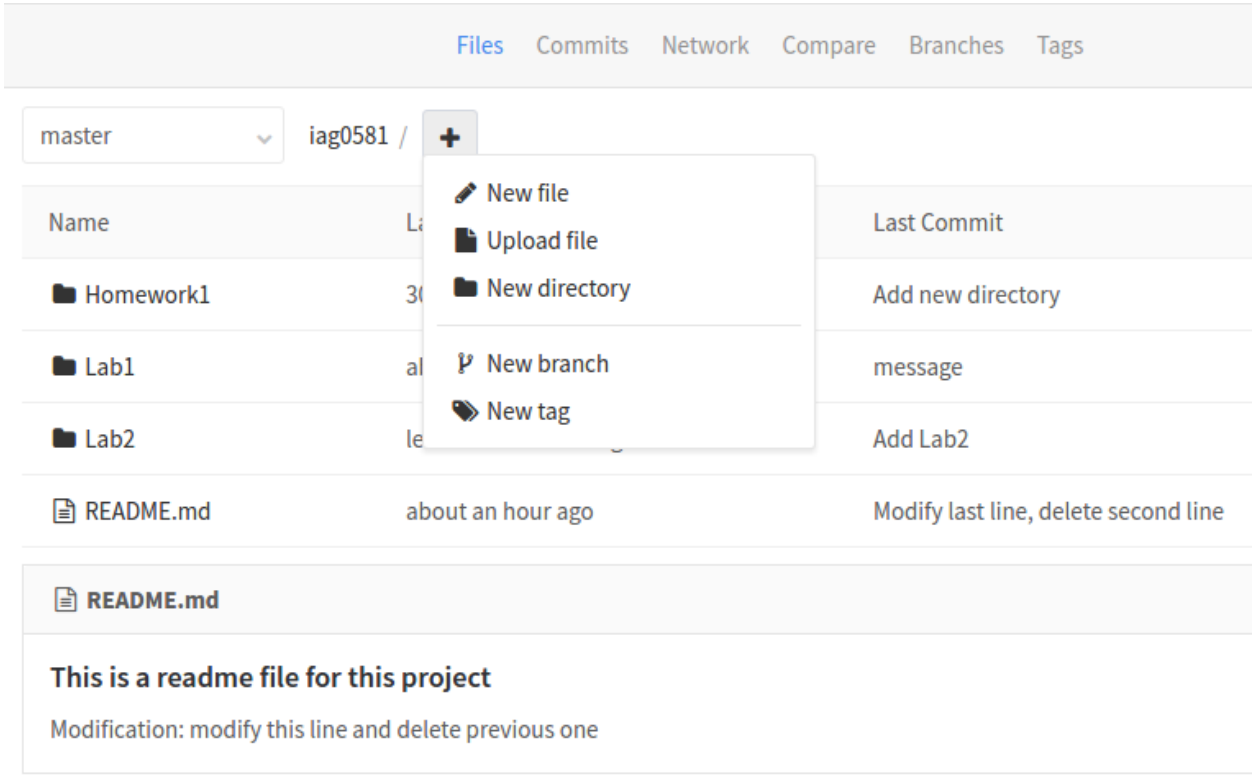
To operate with web interface one must first log in at http://gitlab.ati.ttu.ee/ .

Adding new files / directories

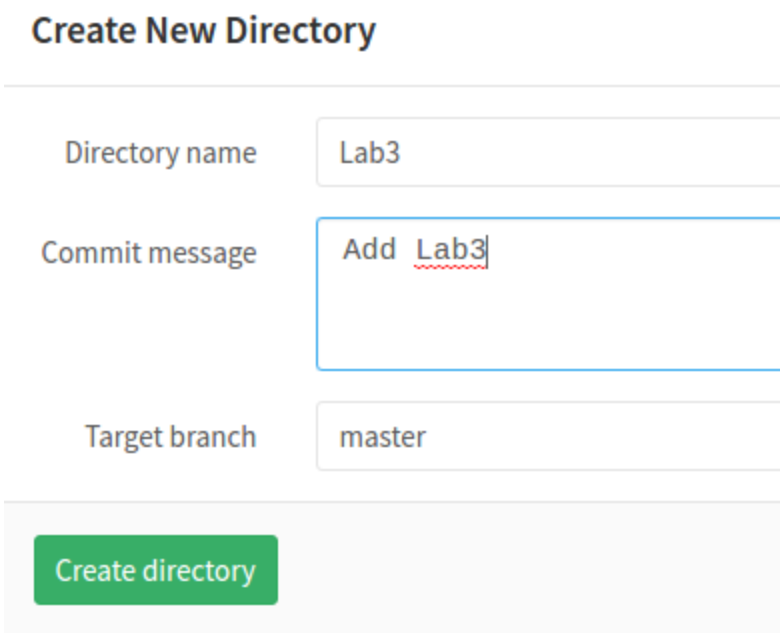
To add new files first navigate to your project and then press “Files”.



Then we can add files by pressing “+” button



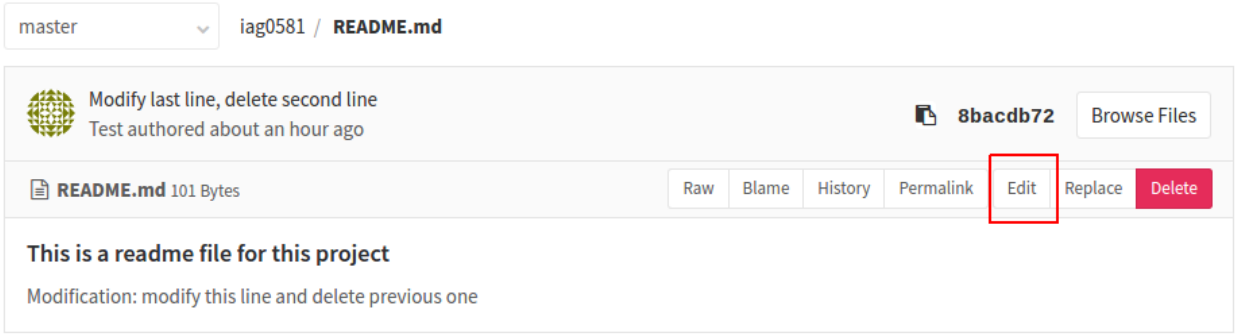
In this example we are adding a new directory “Lab3”



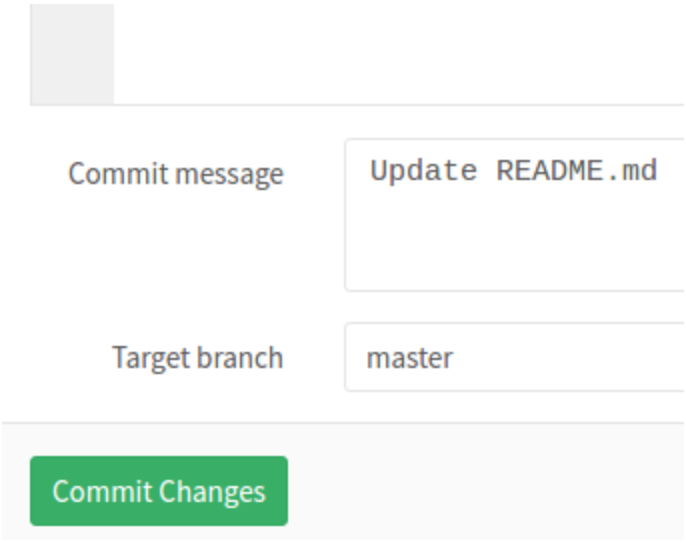
After pressing “Create directory” you will have added new directory with a commit message “Add Lab3”.

Modifying files

To modify file, from “Files” menu navigate to the file that needs modifications, then press “Edit” button.



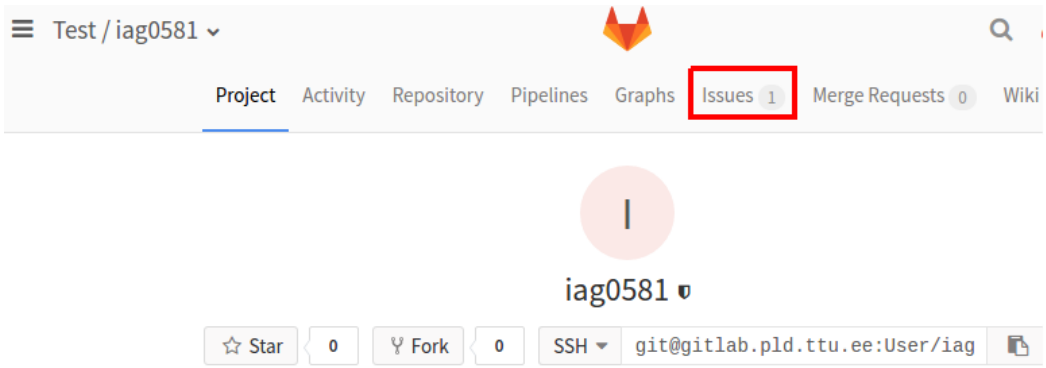
After making your modifications press “Commit Changes”.



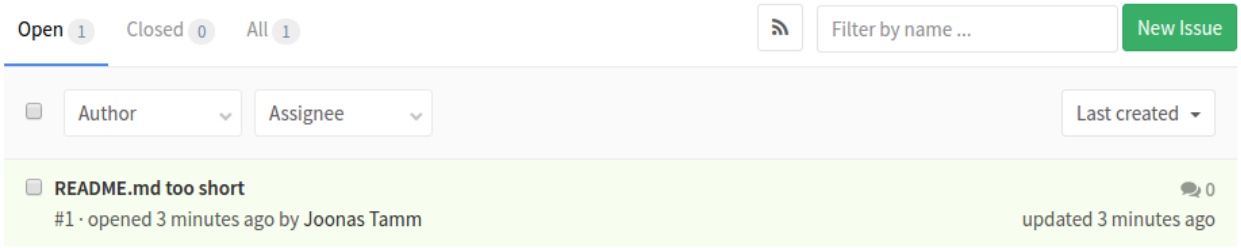
Now you have successfully changed your file and there is a history of it.

Correcting issues

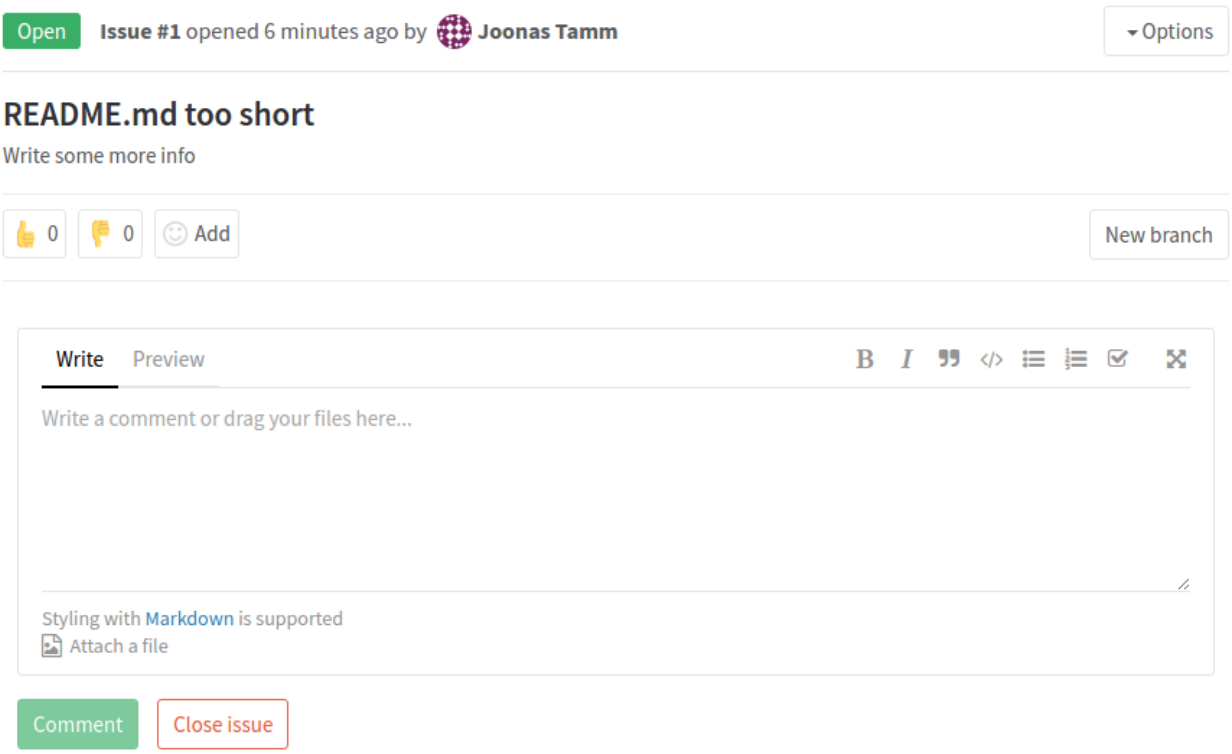
We shall be giving feedback by creating issues, your task is to fix these issues in order to get points.
You can find issues in your project directory:



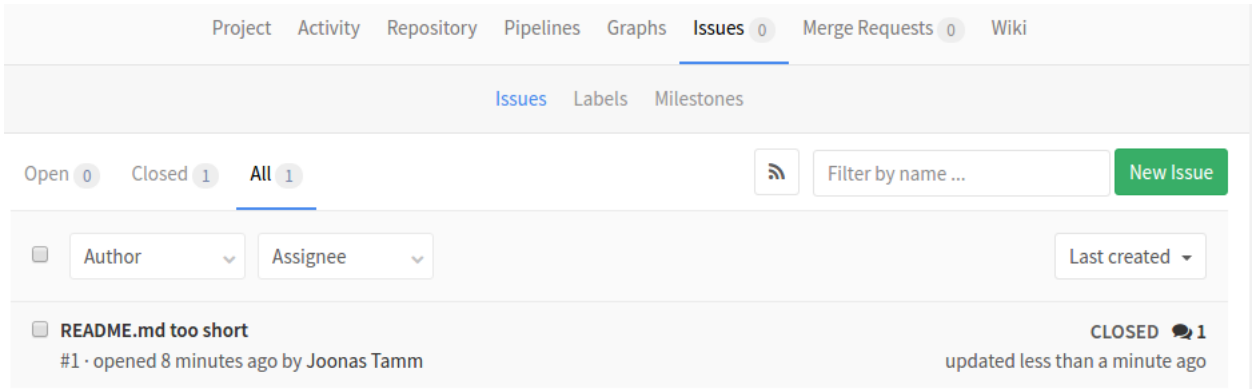
To view issues click on “Issues” and you should see something like this:



After clicking on the title (bold text) of the issue you should see more details.



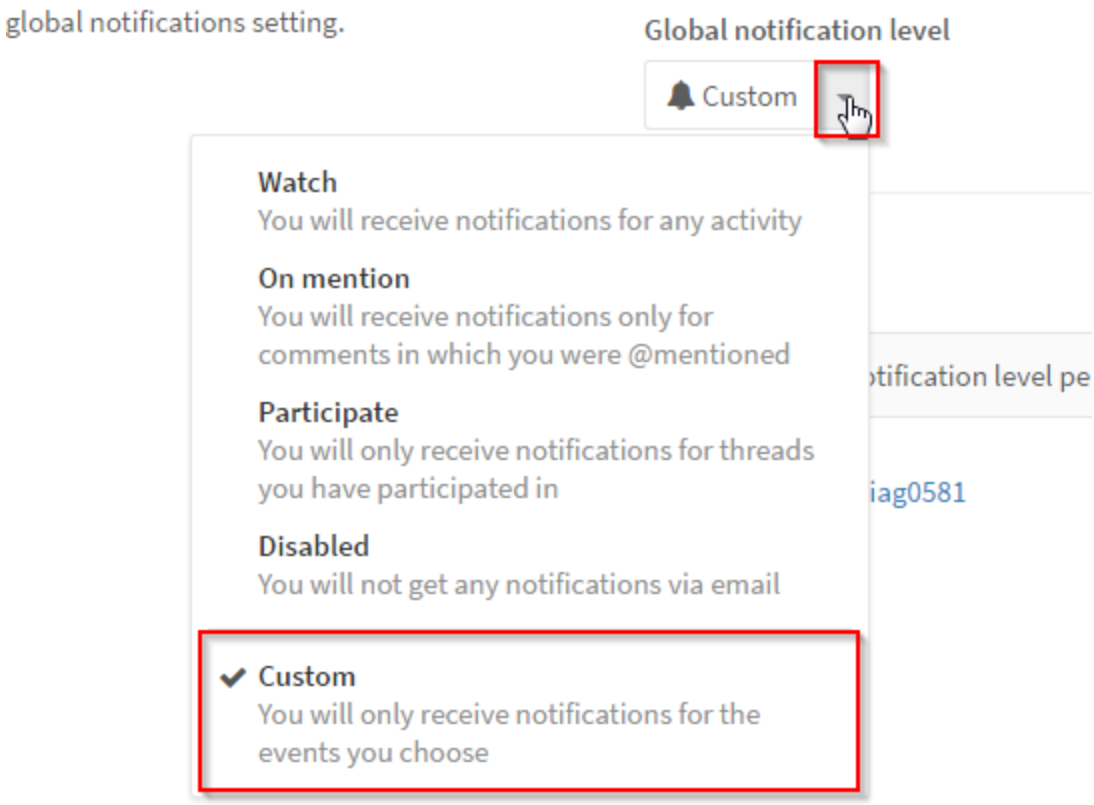
Once you have corrected the mistakes pointed out in this issue return to this page, write what you've done and mark this issue as complete by clicking on "Close issue" In your commits it would be wise to use format "Fix issue [issue number]" e.g. "Fix issue #1"
After closing the issue you can see these issues under "All" tab or "Closed" tab.



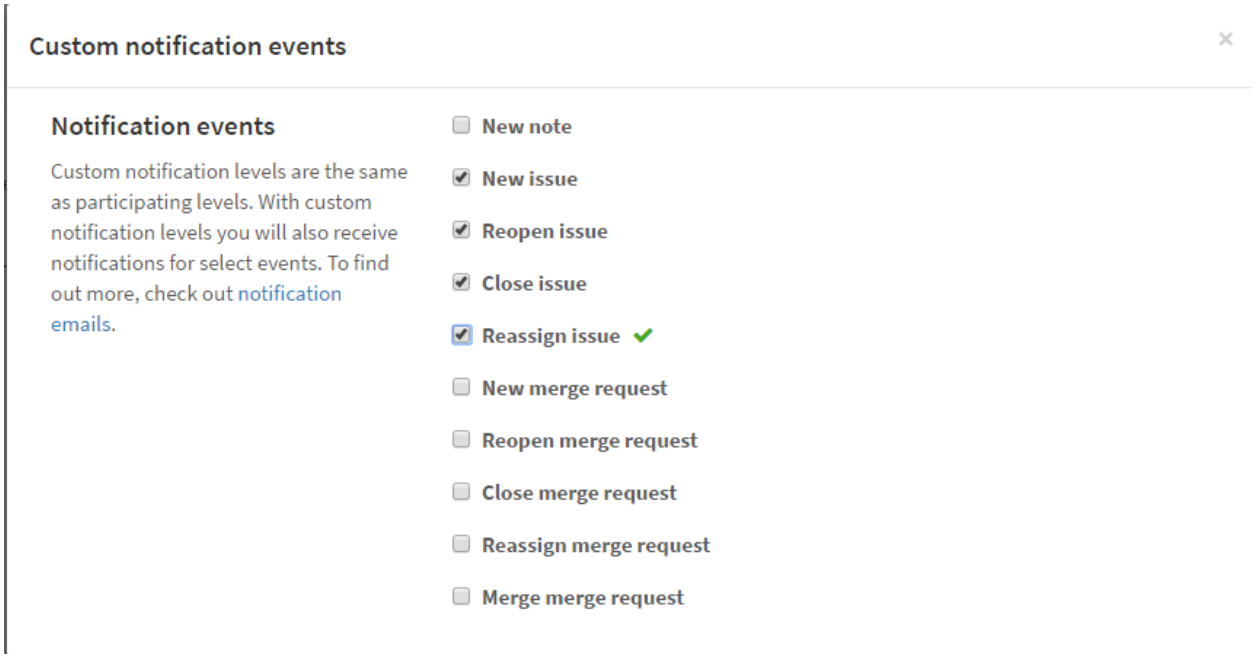
Now you should let us know that you have fixed issue and we will recheck your work.

Enable issue notifications

To receive email notifications about opened issues you must first enable them. Navigate to your notification settings page http://gitlab.ati.ttu.ee/profile/notifications .
Click on the arrow button and then on "Custom"



In custom issue menu you should enable the following notifications:



Email notifications will be sent on your email specified in Notification email tab.

Same settings should be applied to Project notifications. This is located on the same page but on the bottom right corner.

Errors

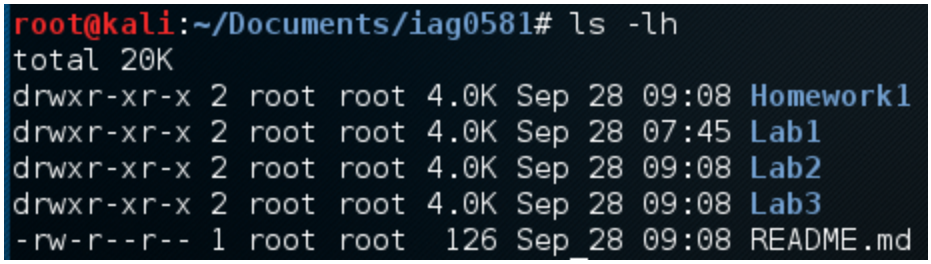
Server certificate verification failed

RULES

Rules described in this chapter must be followed in order to pass this course.

Project structure

Project name must be ***[repository]***. In project directory you should have each lab in different sub-directory. Preferred naming convention would be as follows : ***lab#*** where # is number.
For example by the end of week 3 your project structure could look something like this:



Grading / points

When giving points the following rules apply:

- All of your homeworks and lab works must be uploaded to gitlab.
    - Failing to do so will get you 0 points for homework / lab work.
- Code uploaded to gitlab must compile without errors (and preferably without warnings) and produce expected output.
    - For compiling use c11 **(-std=c11)** standard and enable all warnings **(-Wall)**. Check debug101.pdf for more info.
    - Failing to do so will result with 0 points. **You can correct / improve your code (See Appealing / Correcting)**
- Code uploaded to gitlab must be formatted correctly

    - Failing to do so will result in **0** points. **You can correct / improve your code (See Appealing / Correcting)**
    - Formatting guide by Risto H. : http://blue.pri.ee/ttu/coding-style/
- Missing homework deadline will result in given points divided by 2 (applies only for homework / lab work part).

Appealing / Correcting

In regard to appealing your results and correcting your work the following rules apply:

- You have 1 weeks after deadline to appeal your concerns. After that no changes will be made
    - If you have presented your work within deadline you have 1 weeks to correct your work and still receive maximum points
- You have 1 weeks after deadline to correct / improve your work. After that you will not receive more points for corrected / improved work.
- By default tasks will be graded once after the task **deadline** has passed and once after the **1 week correction deadline** has passed.
    - If you want your work checked after improving your work let one of us know in slack.