Gitlab-Guide for Students

We are working together on a topic involving the production and usage of code and documentation. In order to have a version control, backup your work and to give the opportunity for interaction, the following Gitlab-Repositories have been created for you:

* Thesis-Repo: BA/MA Thesis <FirstName> <LastName>
* Project-Repo: <project\_repo>

Prerequisite for using gitlab is to create a free account via the RWTH [gitlab-ce server](https://git-ce.rwth-aachen.de/users/sign_in).  
Secondly, you should create an SSH-Key for each PC that you are working with.

# Create an SSH Key

Go to terminal and create key-pair with encryption ed25519

ssh-keygen -t ed25519 -C "Gitlab Key Pair for <Your name/PC name>"

Enter file name where to save the key. Recommended: Use standard location: just hit enter and remember the location.  
No passphrase (just hit enter)  
Read the key and copy to clipboard, something like (depends on location of file):

cat ~/.ssh/id\_ed25519.pub

Now copy the output to the Gitlab field TopBar first letter of your name -> Edit profile -> SSH Keys -> Add new key -> Paste and Hit Add Key

# 2a. Initialize repo for first time

Download the current repository into a new folder. If the result is not as you wanted, just delete the whole newly existent folder (together with its .git folder).

In gitlab go to the project and click Clone -> SSH Clone -> Copy link. Then in the terminal do:

git clone git@git-ce.rwth-aachen.de:teamdb/test-repo.git

Then go into the new folder “test-repo” and check the status of your new local git repository:

cd test-repo/  
git status

The status should tell you that you are on branch “main”. This is the branch we develop on.  
If you want to e.g. write a new GUI for your testbench, you can create a new branch “testbench” where you change only code dealing with that testbench until the GUI is finished. Later you then can merge the testbench-branch into the main-branch again to add the new functionality.

# 2b. Pull the (new/current) content of an online repository

If you already have a local repo of the project, and you only want to get it up to date, you can pull the changes from the online branch (of the same name):

git status probably shows last used branch on local repo  
git checkout main want to pull the online main branch into our local main branch  
git pull overwrite my main branch with the online main branch

# Create a new branch (optional)

Let’s assume the name of the new branch is “featurebranch”.   
First, get an overview over the branches with

git branch

Inside your local repo folder do:

git checkout –b featurebranch  
git status

Now you should see, that you are no longer on branch “main” but on branch “featurebranch”.

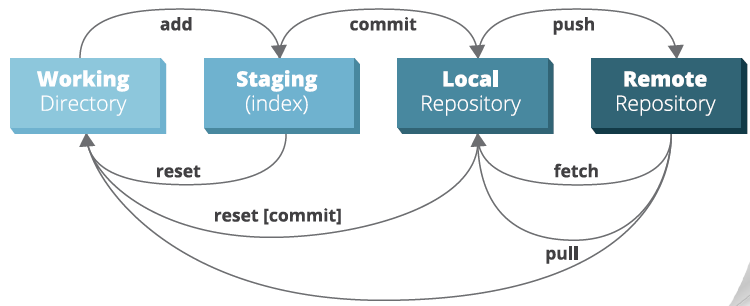
# Changing the content of your repository

Let’s assume you are back on the main branch and in your local repository folder (at the same level as the .git file) developing your code, arranging files into folders etc.  
Now that you have had access to the files inside your local repository and made all changes that you wanted to do, you need to update the online gitlab repository.

First, see which files have been changed and verify that you are on the correct branch.

git status

The following image shows the 3-level update process. If you do everything right, you should only need the commands marked with the red box.



1. Add changes to the repository

git add <file> or   
git add --all for adding whole repository’s changes or   
git add “foldername/\\*” for adding all changes inside a folder “foldername”  
git status shows all added changes in green now

The \* means all paths that have the part “foldername/” in common. Depending on whether you are on Windows or Linux, the \ / might change!

1. Commit the changes to the current branch together with a short message that comments on what you have changed.

git commit –m “New read function for the camera”

1. Push changes from the local branch to the online repository with name of the remote (default: origin) Optional: -u: to create the new branch (only for very first push to a new branch!).  
   Here: Let’s push the changes we comitted inside the local main branch (origin) to the online main branch (main):

git push origin main

# Merge a feature branch into the “main” branch

If you want to merge a branch into “main”, do the following steps on the gitlab webpage:

Merge stuff to the main (not production) branch in Gitlab by clicking “Create merge request” that will delete the source branch if the merge request is accepted (Always do this for feature branches only). Assign DB as reviewer.

Approval is optional, any Developer or Maintainer can approve.   
Can show changes and there on the right is the settings (inline vs side-by-side).  
I can comment on each line of code in the inline-differential view by hovering over it and drag-drop over the lines and clicking.

If you already initiated the merge request and still want to change something in the code to be merged, you can do so and push again. It will all be automatically considered to belong to your open merge request.   
After pushing the corrected final version, go to the review thread and click “Resolve Thread”. Then just **Approve** and **Merge.**

Of course, you can also merge using the terminal.

# Best practice

1. Pull
2. Do the changes
3. If a certain part is ready,
   1. Update the README.md
   2. Push your new version to the online gitlab account

There are many Git Cheat Sheets available online.   
I have attached one here.

