GIT

In this tutorial you will learn about GIT, what it is good for, how it is used and how to get the ini-python-course repository.

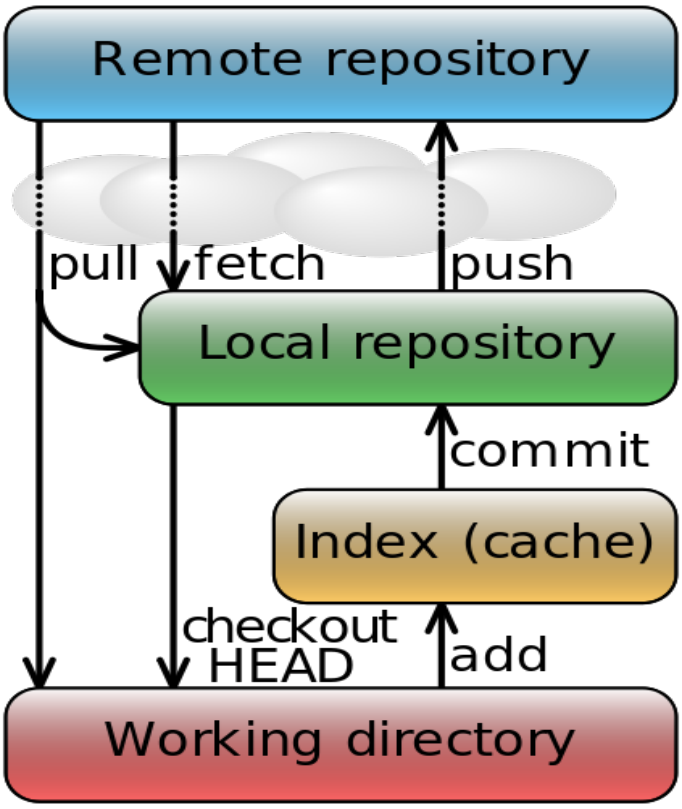
Version control systems

First of all GIT is a version control system (VCS) that is open source and free of charge.
A VCS has two main functions:

- It is a backup tool for you code/project
- It allows different persons to work on a project at the same time without interfering the others changes.

Due to the first point it also make sense to use a VCS if you are working alone on a project. Furthermore, it allows you to synchronize the work across difference machines (PC at home, PC at work, laptop, …)

The following graph illustrates the basic concept of a VCS.

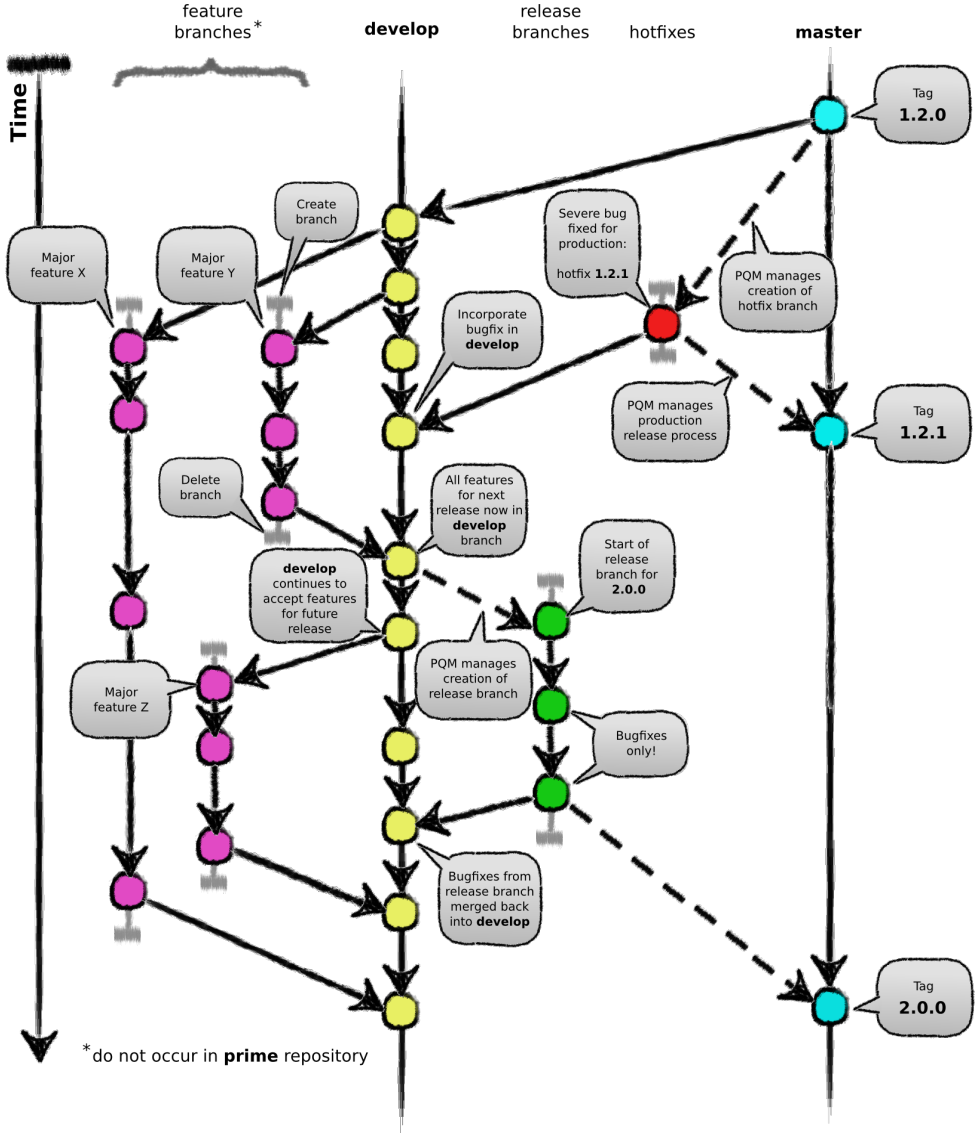


A classic VCS has a *remote repository,* which is hosted usually somewhere in the Internet, it contains the full repository with the full history allowing you to restore any backup at any time. Now to work on the content of the repository you have to create a *local repository,* which usually only contains the recent repository data i. e. not the history for example. *Remote* and *local repository are more like a database,* which contains the data in a compressed form with meta data like restore points etc.. A normal view on the data is provided by the *working directory* showing the

current state of the repository in normal file and folder structure as you expect. Thus files will be modified in the working directory. Once done with the work, a part or simply when it make sense to backup the changes will be committed to the local repository with the *commit* command. If you added files, which were not part of the repository before they have to be added/indexed before by using the *add* command. Now the changes are in the *local repository* and can be send to the *remote repository* by the *push* command. Now everybody else can get the changes by pulling from the remote repository. (For example you, when you want to continue on your on your machine at home.)

Using branches for software-management

A very important concept in VCSs I want to emphasize here are branches. Branches can be seen as development spin offs from the original repository that are meant to the merged back later on. Consider the following example: You developed a successful game on android. To keep the customers playing the game you want to integrate new levels, features while at the same time you have to create bugfixes. A corresponding exemplary GIT flow is shown in the following figure.



You have a master branch (cyan), from which a develop branch (yellow) has been create. All change are done on the develop branch but for bigger features such as new levels one would create a new branch (pink) from the develop branch. These branches will be merged/integrated back to

develop after they are done. If enough changes are done a new release is planned, for which a release branch is created for testing and integrating the code. After that the release is integrated into the master branch by merging the two branches. For small changes a bugfix branch (red) exists, which do not need extra branches and merges directly back to the master branch. Thus a branch keeps new stuff apart from the functioning core until it is ready for release, which helps to not clutter your project. Furthermore, when you just want to try a new feature create a branch! If the feature will be integrated later on merge back if it turned out to not work, just delete the branch.

GIT specific

Most version control systems like SVN for example are centralized such that the local repository is only a working copy, which does not have full history for example. Thus new users can only get the repository from the *remote repository* and restoring of an old version also needs the *remote repository.* GIT is a little bit different than that since it is a distributed version control system, meaning that every copy is a full working repository. This has obviously the advantage that a failure of the remote repository will cause no big problem. However, since a new user can pull from any repository GIT does allow non-linear flows and dependencies (Although possibel I would not recommened to do this)

Material

Now that you have a vague feeling for VCSs and GIT here is some material for deeper understanding and using GIT.

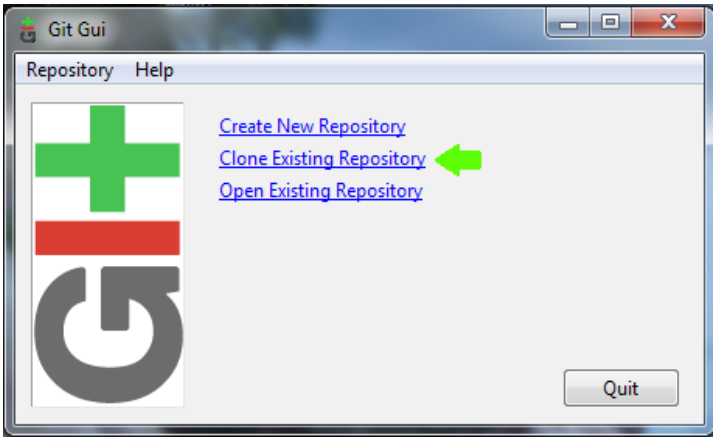
GIT commands overview
https://rogerdudler.github.io/git-guide/index.de.html
https://rogerdudler.github.io/git-guide/index.html

GIT tutorial
http://git-scm.com/docs/gittutorial
https://www.atlassian.com/git/tutorials

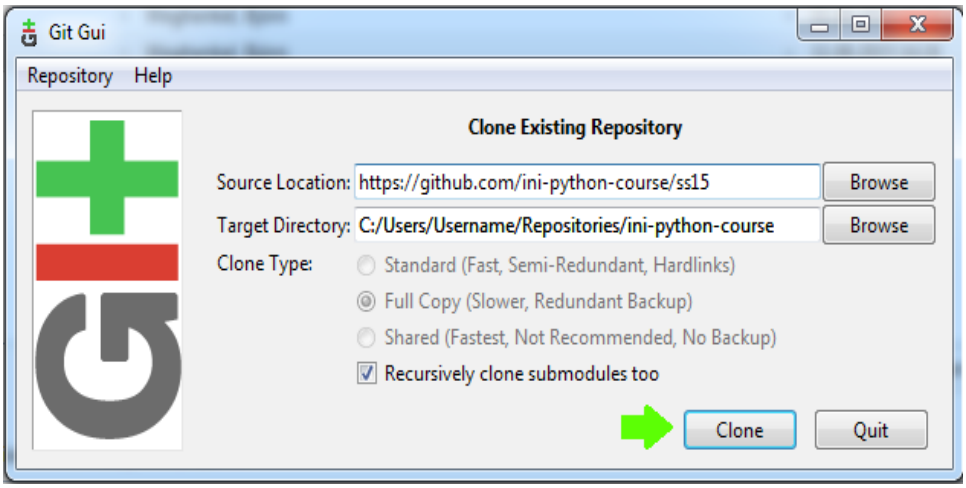
GIT sandbox playground (online GIT demo)
https://try.github.io/levels/1/challenges/1

Getting the ini-python-course tutorial

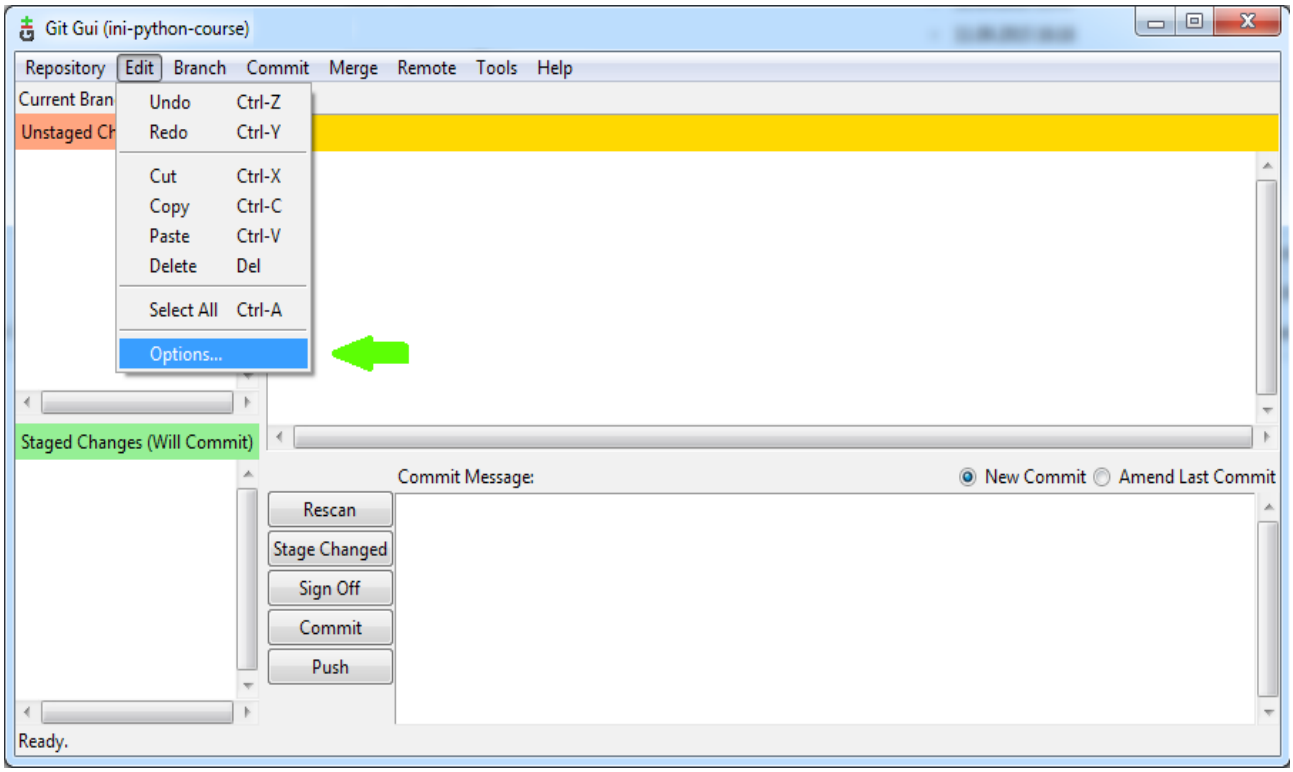
For the first time with GIT here are screen shots of how to get the ini-python-course repository.



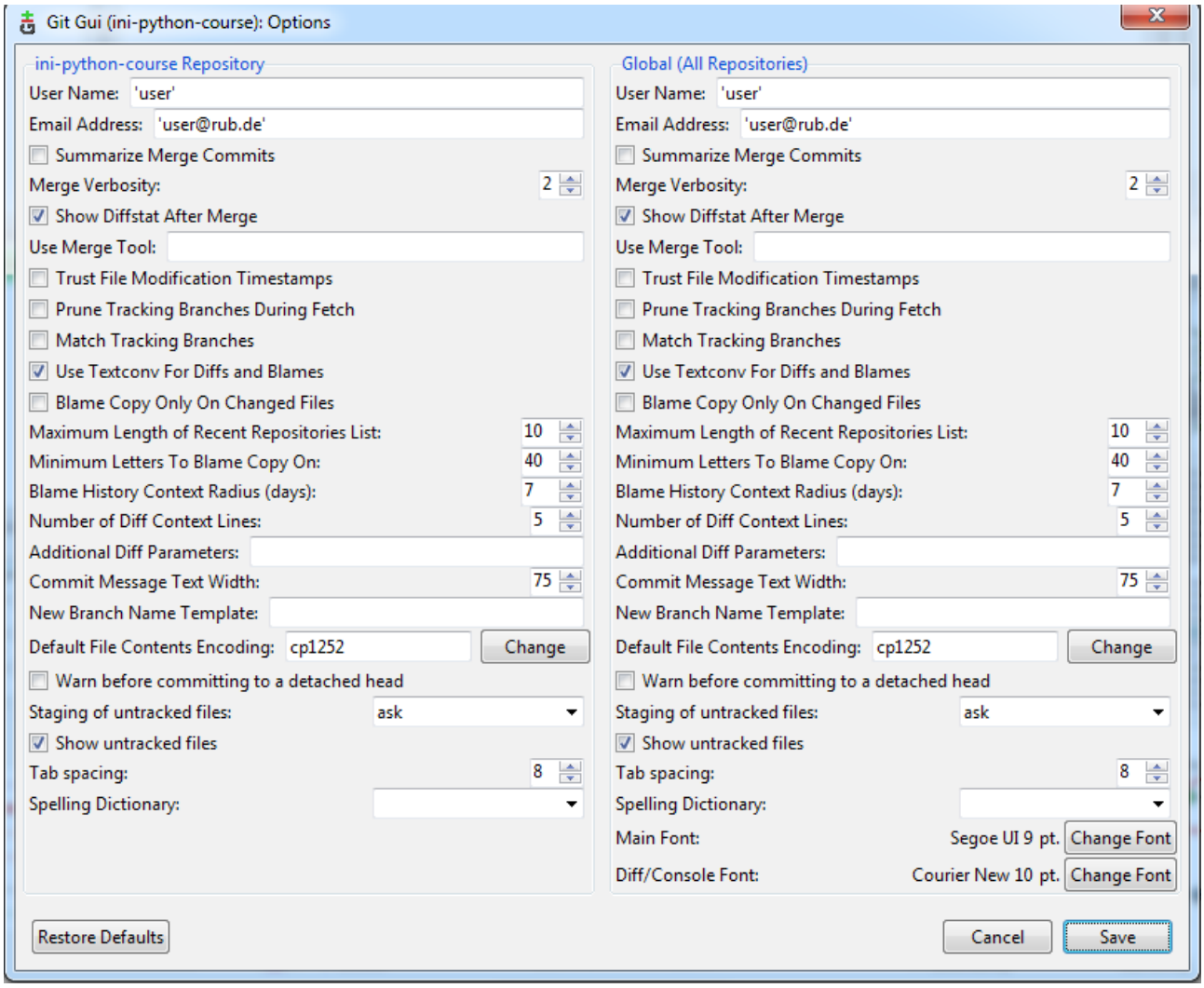
→ Start the GIT-gui and the start screen will appear or if you prefer the command line have a look (in parallel) to the end of this document.
→ Choose *Clone existing Repository*, enter the following source by replacing ss15 by the current semester and enter a destination directory



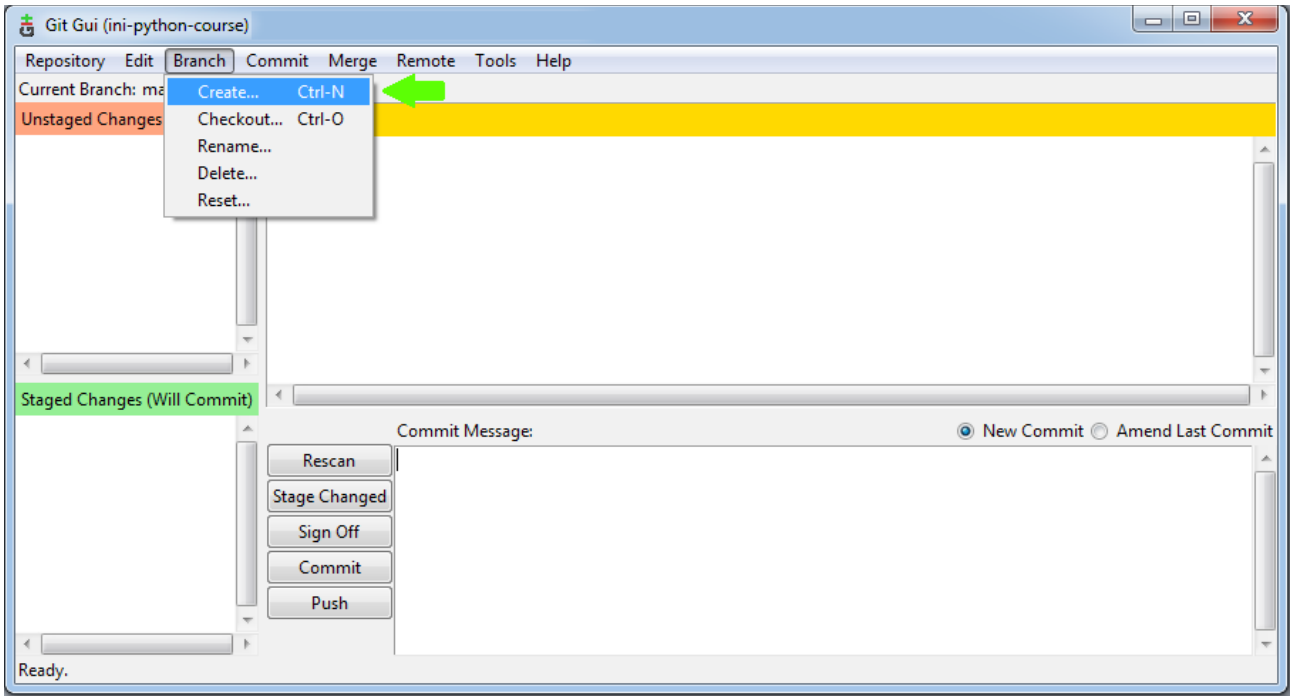
→ The project will be cloned, meaning that the local repository will be created. The default dialog appears.

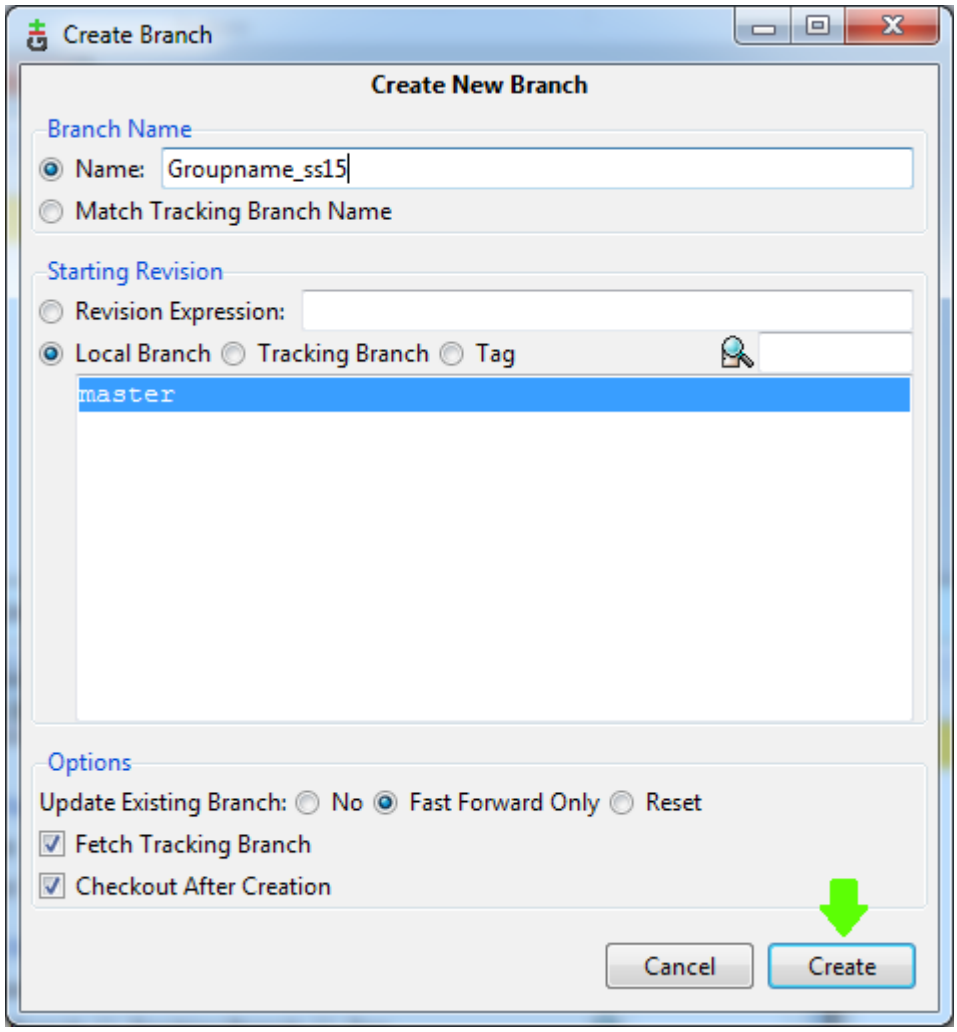


→ The next thing you have to do is to tell GIT you you are, therefore choose *Edit* → *Options* and enter your name/nick and your email. This information will be connected to your changes and commits such that one knows who has done what. You can set the info for the current repository or global for all repositories.

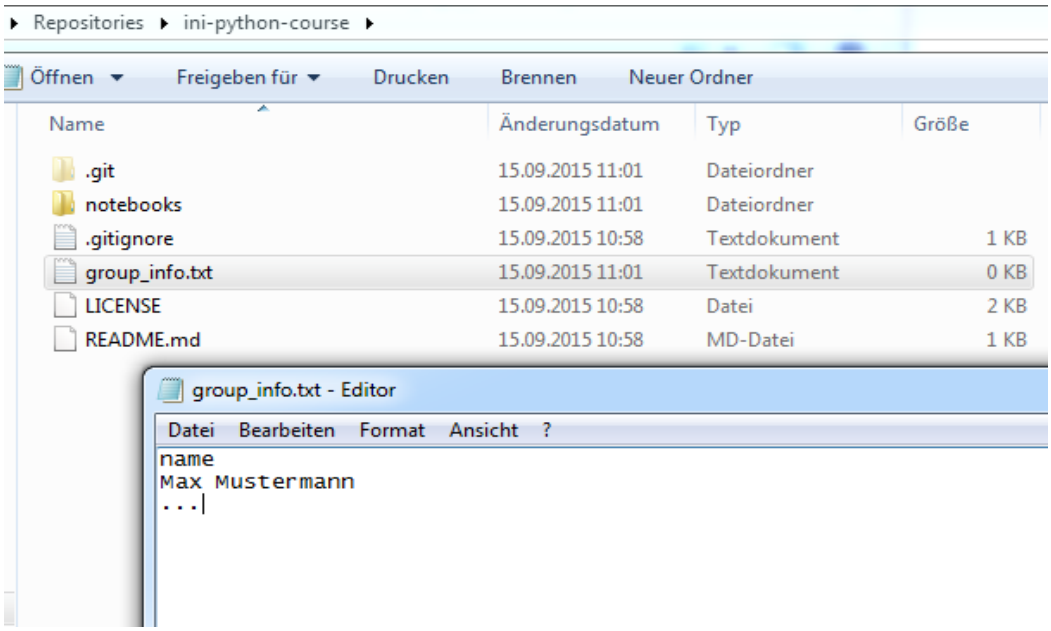


→ You are currently on the master branch such that we have to create a branch for your group and switch to that. Therefore, choose *Branch* → *Create* .



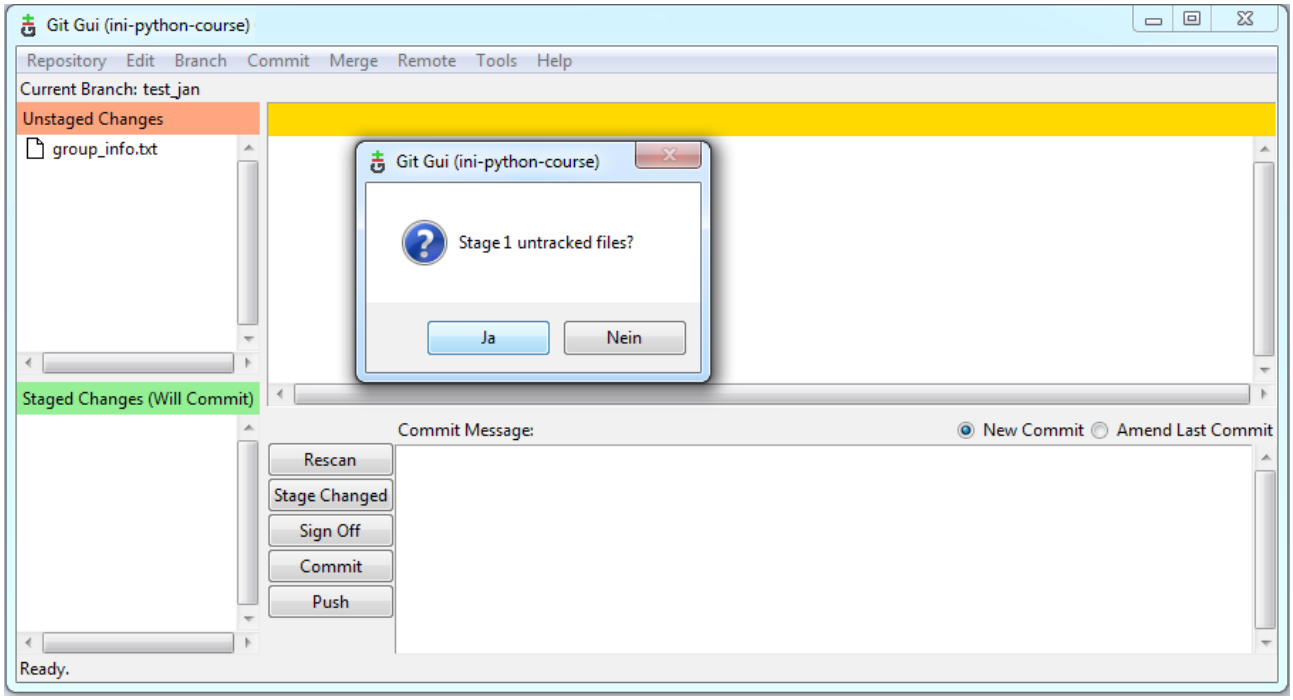


→ The name for the branch should be your *group name* followed by a *underscore* and the *semester* as shown in the figure above. Use the default settings are also shown in the figure above.
→ With the *checkout* command you can switch the working directory between the branches. Since we hooked *Checkout After Creation* we will be on the new branch automatically. You could go back to the master branch by Branch → Checkout and then selecting the master branch.

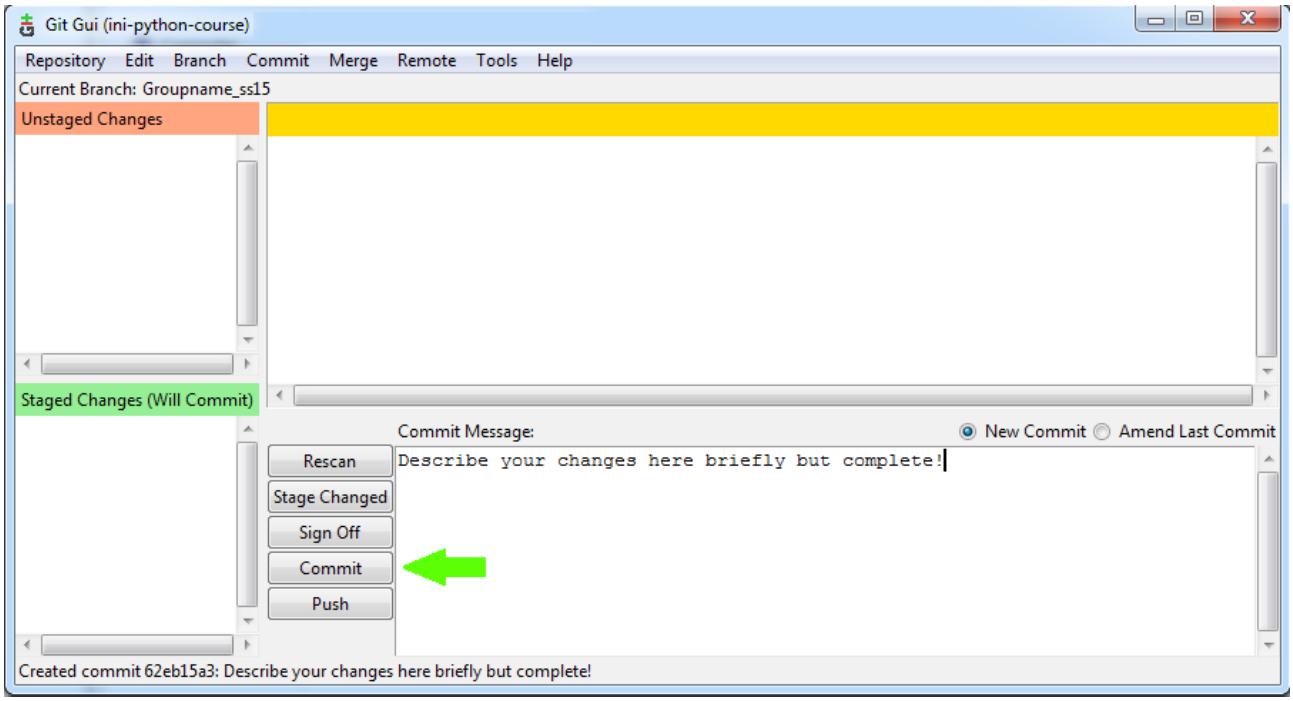


→ Now you can change the files in the working directory, which is the target directory you have been chosen in the beginning when cloning the repository.
→ Let's add a file named group_info.txt which contains the names of the group members to the repository as shown in the figure above.

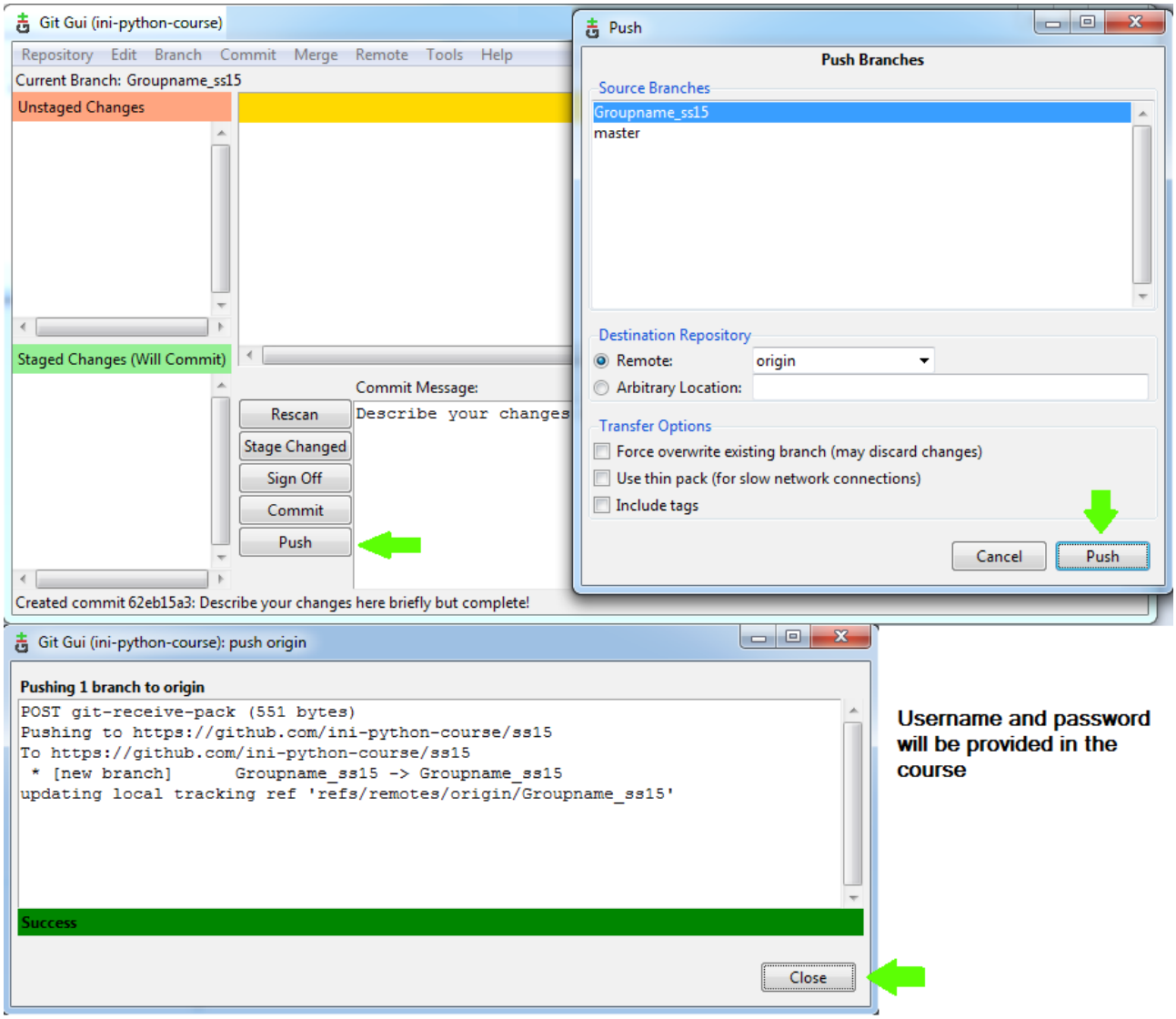
→ Whenever you add files you have to tell git such that it can index the new files, therefore press the rescan button. So far the new file is unstaged meaning that it will not be commited with the next commit. To change this press *stage changed*.



→ Now you are ready to commit, therefore enter a description of you changes, which should be short but complete and press the *commit button*.



→ Now push the changes by pressing the *push* button, select you branch as source and press *push*.
→ The remote repository host , Github will ask for a username and password that will be given to you in the course.



→ If the green line with success appears you are done!

→ The same can be done with the command line. The following figure shows the same steps in the same order we just did with the GUI.

```
C:\Users\Jan>cd Repositories

C:\Users\Jan\Repositories>git config --global user.name 'user'

C:\Users\Jan\Repositories>git config --global user.email 'user@rub.de'

C:\Users\Jan\Repositories>git clone https://github.com/ini-python-course/ss15 in
i-python-course
Cloning into 'ini-python-course'...
remote: Counting objects: 51, done.
remote: Compressing objects: 100% (37/37), done.
remote: Total 51 (delta 22), reused 38 (delta 13), pack-reused 0
Unpacking objects: 100% (51/51), done.
Checking connectivity... done.

C:\Users\Jan\Repositories>cd ini-python-course

C:\Users\Jan\Repositories\ini-python-course>git branch groupname

C:\Users\Jan\Repositories\ini-python-course>git checkout groupname
Switched to branch 'groupname'

C:\Users\Jan\Repositories\ini-python-course>git add .

C:\Users\Jan\Repositories\ini-python-course>git commit -m"Describe your changes
here briefly but complete"
[groupname ed4aa8c] Describe your changes here briefly but complete
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 group_info..txt

C:\Users\Jan\Repositories\ini-python-course>git push --set-upstream origin group
name
Username for 'https://github.com': ini-python-course
Password for 'https://ini-python-course@github.com':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 295 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To https://github.com/ini-python-course/ss15
 * [new branch]      groupname -> groupname
Branch groupname set up to track remote branch groupname from origin.

C:\Users\Jan\Repositories\ini-python-course>git push
warning: push.default is unset; its implicit value has changed in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the traditional behavior, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

When push.default is set to 'matching', git will push local branches
to the remote branches that already exist with the same name.

Since Git 2.0, Git defaults to the more conservative 'simple'
behavior, which only pushes the current branch to the corresponding
remote branch that 'git pull' uses to update the current branch.

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Username for 'https://github.com': ini-python-course
Password for 'https://ini-python-course@github.com':
Everything up-to-date

C:\Users\Jan\Repositories\ini-python-course>git pull
Already up-to-date.

C:\Users\Jan\Repositories\ini-python-course>git status
On branch groupname
Your branch is up-to-date with 'origin/groupname'.
nothing to commit, working directory clean

C:\Users\Jan\Repositories\ini-python-course>_
```

EGIT

Later we will use a plugin for eclipse named egit which allows you to pull, commit, push, merge, … directly from the eclipse IDE, which is pretty straigth forward and will be explain on the fly.

http://www.eclipse.org/egit/