git **init**

git **clone** <URL of the remote repository to clone from>

git **remote** add <name of the new remote> <url of the remote>: adds a remote repository

git **config** --global alias.<name of the alias> ‘<git command>’

git config --bool core.bare true : make a repository bare, without working directory

git **add** <file name/ “.”>: add all files to the stage

git **commit** [–m ‘<message>’]

git commit --amend: corrects the last commit. Changes the hash.

git **tag** -n6: enumerate all tags with 6 lines of the annotated message

git tag [-a, -d] <name> : -a: annotated, -d: delete

git tag <tag name> <tag name>^{} -f –a: change the message of a tag

git tag -f -a <tagname>: change the tag to the current commit.

git **push** [--follow-tags] <name of the bare remote repository> <name of a branch/ “--all">: update the remote repository from the local. --follow-tags: pushes the annotated tags, not if it is missed.

git push <name of the bare remote repository> <name of a branch> --force: update the branch in the remote repository (by deleting commits if neccesary)

git **fetch** <name of the remote repository> [-p]: downloads the new modifications from the remote repository. -p: remove the obsolete branches.

git **pull** <name of the remote repository> <name of a local branch>: combination of ‘fetch’ and ‘merge’.

git **mergetool**: while merging two branches with conflict.

git **status**: see the tracked files to commit.

git **diff** <commit> <commit>

git **difftool** <commit> <commit>

git difftool --dir-diff: open all different files together.

git log --oneline --all --graph --decorate --since=2015-10-30 --max-count=20

git **checkout** <name of the branch>: changes the branch

git checkout -b <name of the branch>: creates a new branch and change to it.

git checkout <name of the branch> [name of the file]: updates the working directory (or only the specified file) to the stage.

git checkout -- .: remove all modifications in current commit.

git **reset** HEAD file name: updates the stage of the specified file) to the last commit, without deleting commits.

git **reset** HEAD: updates the stage (or only the specified file) to the last commit. Delete if you are on the last commit of a branch

git reset --hard HEAD~3: updates the stage and the working repository (or only the specified file) to the fourth last commit, and delete the new ones.

git reset --soft HEAD~3: change the current commit to the fourth last commit and delete the new ones, while keeping the changes of the last commits (nothing will be lost as every past change is in the stage or remain untrackedly changed).

git **restore** <file name/ “.”>: discard changes in the file in the working directory that were made under a different commit.

git **restore** –staged <file name/ “.”>: unstaged the stages changes in the file that were made under a different commit.

git **branch**: Shows the available branches.

git branch [-d]<name of the branch>: creates (-d: deletes) a new branch.

git branch-m<old name of the branch> <new name of the branch>: changes the name of the branch.

git **cherry-pick** <name of a commit>: set the commit in the current branch.

git **merge** [--squash] <name of the branch>: merges the indicated branch with HEAD. (-squash: add into the stage all changes made in the indicated branch)

git merge --abort: resets the commit, where were to be resolved the conflict.

git **rebase** <name of the branch>: rebases (level the two branches by adding the HEAD-branch to the indicated branch, while changing the hashes of the HEAD-branch (the identifiers of the commits)).

git rebase -i <name of the branch>~5: opens an interface to edit the last 5 commits of the indicated branch (to edit the messages of the commits, to change the order or to squash some commits into one).

git **stash** [-m “…”]: save the changes in a stash-entry [-m: custom message (default: commit number)].

git stash list: lists the saved stash entries with name and the branch.

git stash apply [stash@{2}]: loads the last [or the specified] stash entry.

git stash pop [stash@{2}]: combination of apply and drop.

git stash drop [stash@{2}]: removes the last [or the specified] stash entry.

git stash clear: removes all stash entries.