**Git**

Git has three main states that your files can reside in: committed, modified, and staged:

* Committed means that the data is safely stored in your local database.
* Modified means that you have changed the file but have not committed it to your database yet.
* Staged means that you have marked a modified file in its current version to go into your next commit snapshot.

**Basic Git Workflow**

* Create a new or checkout an existing repository.
* Modify/add/delete files.
* Either selectively stage just those changes you want to be part of your next commit or stage all.

- You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

- If working with a remote, you push local changes to the remote.

**Creating a repository**

**Create a remote repository via command line and/or in the Browser on github.com.**

**Remote repository:**

*curl -u 'funkerresch' https://api.github.com/user/repos -d '{"name":"realtimeaudioprogramming"}'*

**Local repository:**

Create a folder with the name of the repository (command line cd to directory, then mkdir DIRECTORYNAME)

Create a textfile *readme.md*, either manually or via command line, for example, with

*touch readme.md*

*cd TO/LOCAL/REPOSITORY*

*git init*

**Staging files**

**Stage everything according to the local directory structure including new, modified and deleted files):**

*git add –A*

**Stage everything without deleted files:**

*git add .*

**Stage modified and deleted, without new files:**

*git add -u*

**Stage single files and folders:**

*git add <filename>*

*git add <foldername>*

**Commit**

**Commit all staged files to local repository:**

*git commit –m “First Commit” (-m for commit message)*

**Add a remote to the local repository**

*git remote add origin https://github.com/funkerresch/stp\_seminar\_tuberlin*

**Push, pull and clone**

**Push to remote and link local repository with remote so git pull can be used without arguments:**

*git push –u origin master* (-*u links the local repository with the remote)*

**origin is an alias for the remote, only needed once, after that simply use:**

*git push*

**Download the head of the remote and merges it with your local repository:**

*git pull*

**Download and inits a new local repository from PATH/TO/REPOSITORY:**

*git clone PATH/TO/REPOSITORY*

**Git status information**

**To show aliases of your remote server:**

*git remote –v*

Instead of origin you could also use the url https://github.com/funkerresch/stp\_seminar\_tuberlin

**Get general status information:**

*git status*

**List the commit history:**

*git log*

**General information about HEAD:**

git show HEAD

**The easiest way to create local and remote repository**

Create a remote repository online including readme.md

Copy the link to the repository

*cd TO/LOCATION/WHERE/YOU/WANT/TO/SAVE/YOUR/LOCAL/REPOSITORY*

*clone https://github.com/LINKTOYOURREMOTEREPOSITORY*

**Gitignore**

Create .gitignore file with *touch .gitignore* (under linux and osx it will be invisible) in the

Root directory of your repository.

Use nano vi or emacs for editing : *nano .gitignore*

Add, for example, the line

*\*.html*

and save the file.

Now all html files will be excluded from staging

**Branches**

**To create a local branch:**

*git branch test*

**Switch to branch test:**

*git checkout test*

Edit something in your source with

*add -A*

*commit –m “Edit test branch”*

**To merge a branch with the master branch, switch to master:**

*git checkout master*

*git merge test*

**To delete local branch (only if merged and pushed to remote):**

git branch -d branch\_name

**Delete local branch (force):**

git branch -D branch\_name

**Delete the remote branch test:**

*git push origin --delete test*

**Branches II**

**To check out commit id and create a new branch of it:**

*git checkout -b <NEW BRANCH> <COMMITID>*

**To check out X commits before HEAD and create new branch of it:**

*git checkout –b <NEW BRANCH> HEAD~X*

**Get information about X commits before HEAD:**

git show HEAD~X

**Submodules**

**Add a submodule to your repository:**

*git submodule add PATH/TO/SUBMODULE*

**Remove submodule from your repository:**

*git submodule deinit <path\_to\_submodule>*

*git rm <path\_to\_submodule>*

*git commit-m "Removed submodule"*

**Remove submodule from directory:**

rm -rf .git/modules/<path\_to\_submodule>

A nice introduction to git:

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