Using BASH for Git  
  
git config –global user.name ”Jill Moens”  
git config –global user.email [jill.moens@gmail.com](mailto:jill.moens@gmail.com)  
  
git config –list  
  
  
ls = list  
ls -la =all of the list  
git init = initialized empty git repository on your computer (first go the the desired map) (the master branch is then automatically created)  
git status = see in which status the files are: modified, staged or committed  
git add “filename” = status from modified to staged  
git add -A = all files status from modified to staged  
git reset “filename” = return status from staged to modified  
touch .gitignore = creating a .gitignore file. In this file you can add the filenames only which you do not want to change the status. (example in .gitignore file: index.html )  
git commit -m”” = committing the files with a message between “”.  
git log = see the unique number, the author and date.  
git clone “url” . = copy the url you want to clone. (First go to the desired (empty) folder. The dot is saying it must be placed in this folder)  
git remote -v = information about the repository  
git branch -a = shows the branches you have  
  
After commiting, you edited again. 🡪 Pushing changes  
Commit changes like previously: (so first locally)  
git diff (see the changes to the code)  
git status  
git add -A  
git commit -m””  
Then push:   
git pull origin **master** =(origin is name of repository: check with git remote -v, master is the branch. it will pull changes that have been made to that repository (made by others for example)  
git push originmaster = (pull the files to the repository)  
  
<https://www.youtube.com/watch?v=HVsySz-h9r4>

git checkout master OR  
git switch master = Switching to the branch master  
git checkout -b “new branch name” OR  
git switch -c “new-branch name” = creating a new branch and moves the HEAD to this branch name (-c means –create)  
git switch - = return to your previously checked out branch  
git branch “new branch name” = creating a new branch

! always pull from the master and push your new features to a new branch, so your colleagues and manager can give feedback first. Then if the code is good, then you ‘merge’ to add the code to the master branch. If new code is added to the master branch, everyone who has access to the software (for example), gets a notification that there is an update. Anything in the main branch is always deployable.

HEAD is a pointer to the local branch you’re currently on.  
git log –oneline –decorate = here you see where HEAD is pointing to  
git log –oneline –decorate –graph –all = history of commits  
git log –all = displays all the branches

(HEAD -> master, testing) (master is the branch you’re currently on)

The HEAD moves forward when a new commit is made.   
  
  
  
  
HEAD moves when you checkout

That command did two things. It moved the HEAD pointer back to point to the master branch, and it reverted the files in your working directory back to the snapshot that master points to. This also means the changes you make from this point forward will diverge from an older version of the project. It essentially rewinds the work you’ve done in your testing branch so you can go in a different direction.

New commit at master branch:   
Now you can merge!  
  
push: doorgeven   
pull: ophalen  
  
STEPS  
1. Maak op github een nieuwe repository.  
2. Zet er een bestand in.  
3. Clone de repository naar je computer.  
4. Maak een nieuwe branch aan en ga daar heen (head)  
5. Wijzig het bestand op je computer (voeg er code aan toe bijv).  
6. Commit   
7. Push naar de nieuwe branch  
8. Ga naar github en pull request  
9. Merge je branch met main  
10. Op je computer ga je weer naar main.  
11. Haal het aangepaste bestand op met pull.