**+Git and Git Bash and Git Hub**

# Set-up directory (folder)

Open directory = cd (press tab button to let it autocomplete. EG:

linda@DESKTOP-J53OA8V MINGW64 /

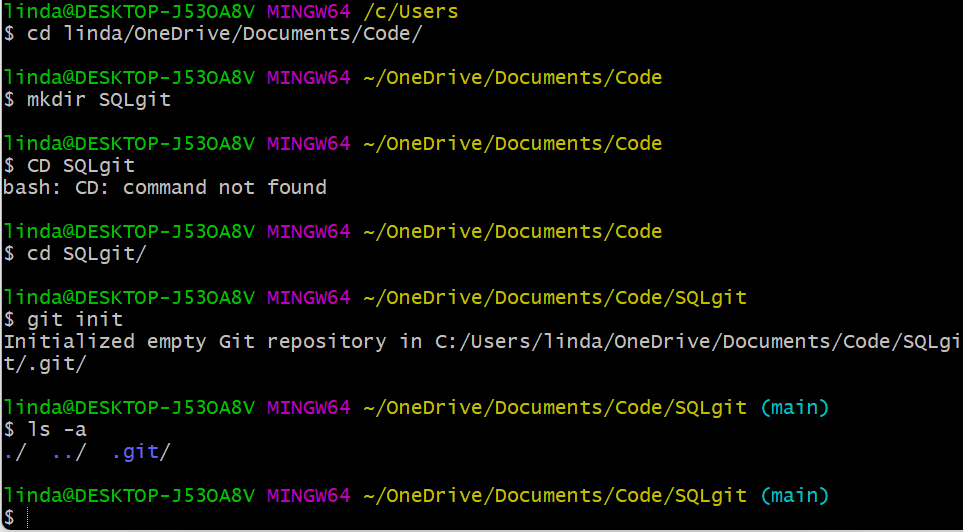
$ cd c:\U

Pressing Tab will autofill to

linda@DESKTOP-J53OA8V MINGW64 /

$ cd c:\Users

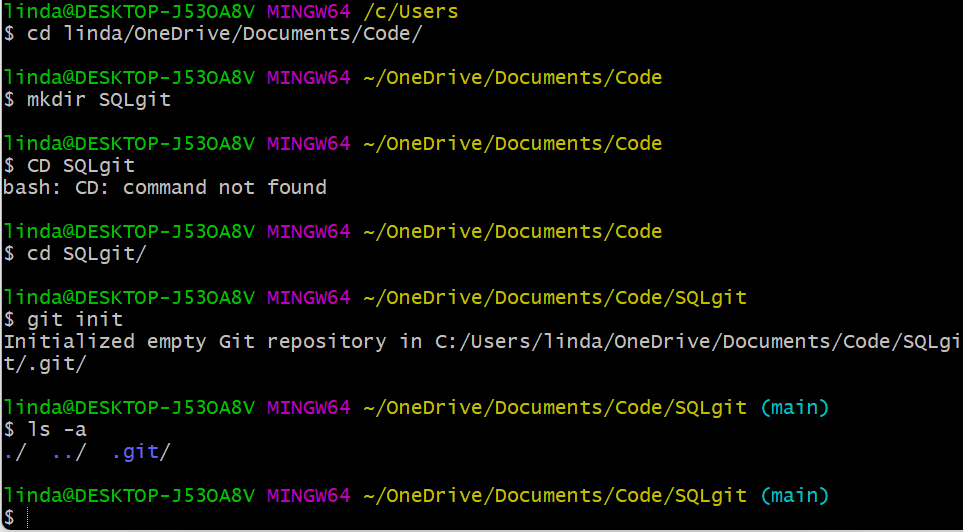
From there I’ve done cd to get to my code project folder and within that I’ve made (mkdir) an *SQLgit* folder:



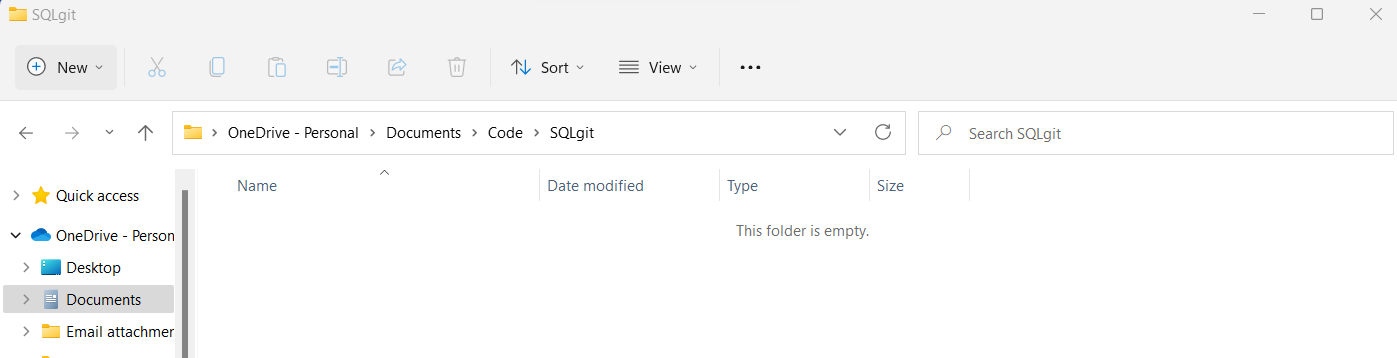
Then cd again to open the SQLgit folder we’ve just made (remember forward slash – I forgot below)

Then git init which prepares the folder for git (initialises it)

Optional: ls –a just allows us to check it worked and it should return: ./ ../ .git/



So far: one open folder that is initialised for git (you can check it physically for piece of mind! Reminder: git init is invisible so you’ll have a SQLgit folder that appears empty.



Use *touch* to create a readme.txt check it was successful by using list (*ls*)

*Mkdir* – creates a folder

*touch* – adds a time stamp to file but if the file doesn’t exist it creates an empty file.

linda@DESKTOP-J53OA8V MINGW64 ~/OneDrive/Documents/Code/SQLgit (main)

$ touch readme.txt

linda@DESKTOP-J53OA8V MINGW64 ~/OneDrive/Documents/Code/SQLgit (main)

$ ls

readme.txt

so far git status will show there is one readme file that is not being tracked by git because we need to add it, type *git status* to see this

linda@DESKTOP-J53OA8V MINGW64 ~/OneDrive/Documents/Code/SQLgit (main)

$ git status

On branch main

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

readme.txt

nothing added to commit but untracked files present (use "git add" to track)

So then if you add the readme file using *git add*

linda@DESKTOP-J53OA8V MINGW64 ~/OneDrive/Documents/Code/SQLgit (main)

$ git add readme.txt

linda@DESKTOP-J53OA8V MINGW64 ~/OneDrive/Documents/Code/SQLgit (main)

$ git status

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: readme.txt

it’s being tracked but has not been committed yet.

**Optional add a git ignore file:**

Touch .gitignore – creates a gitignore which means its in the folder but not tracked by git, you can write rules in your repo e.g: ignore all pdf files for example so that your documentation stays consistent and separate to your code.

If you add them on the same line it adds to gitignore – example below adds to text files (password and password2) to git ignore and shows their status compared to the read me. Note when running list/ls it just shows all the files, not their git status as it’

linda@DESKTOP-J53OA8V MINGW64 ~/OneDrive/Documents/Code/SQLgit (main)

$ touch .gitignore password.txt

linda@DESKTOP-J53OA8V MINGW64 ~/OneDrive/Documents/Code/SQLgit (main)

$ ls

password.txt readme.txt

linda@DESKTOP-J53OA8V MINGW64 ~/OneDrive/Documents/Code/SQLgit (main)

$ git status

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: readme.txt

Untracked files:

(use "git add <file>..." to include in what will be committed)

.gitignore

password.txt

linda@DESKTOP-J53OA8V MINGW64 ~/OneDrive/Documents/Code/SQLgit (main)

$ touch .gitignore password2.txt

linda@DESKTOP-J53OA8V MINGW64 ~/OneDrive/Documents/Code/SQLgit (main)

$ git status

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: readme.txt

Untracked files:

(use "git add <file>..." to include in what will be committed)

.gitignore

password.txt

password2.txt

# COMMIT, PULL AND PUSH

*git commit –m “type message here”*

Commits the file to the repo (so far add just means git is tracking the file locally. Remember git is the version control system, git hub is the repo online for others to use, they are completely different things).

In future you can also try git commit –a –m “type message here” which commits every tracked file in the folder (I assume this means as long as they’ve been added at one point then it will commit them but I’m not really sure) but made a note of it as when we’re more confident it might make more sense and be useful

Git pull after you’ve committed - if you have uncommitted changes, the merge part of the git pull command will fail and your local branch will be untouched, so always commit your changes in a branch before pulling new commits from a remote repository (according to freecodecamp and other sources online but may check with Martina).

*git push -u origin main*

or instead of main put whatever branch you are committing to. In future commits you won’t have to do this unless you want to change the branch