Command Line

Graphicsl user interface – GUI

Comand prompt is an ulternative to gui’s

(time saver)

Helps to start server

Install and download tools

Run code

Execute file

Helps with git

Dir to view all folders in the “main file”

Cd command stands for change directory and .. means move one folder up

Absolute and relevant paths.

Relevant path means starting in one file and navigating to another

Absolute path starts at the beginning easier way to move

Relative paths are normally used at work

Creating and deleting folders

Creating a new one is-mkdir with a file name eg mkdir practice and then cd practice to change to that new file that has been created.

To create a file(text, word or excel)

File name (the one you are in already shows) >echo “what is going to be in the document” >test.txt or test.xls for excel

And to see what is in this file is file you use the command type with the file name eg type test.xls

Cls to clear the command panel

Deleting files eg del test.xls

Deleting folder eg rmdir and the file name

Copying and moving files.

To copy files eg copy test.xls

The cd to different file

To move a file or folder eg move test.xls

Command prompt basics

dc-print current path

dir-list items

cls-clears command prompt

cd .. -move one directory up

echo text>name(.type of the file)

mkdir filename- makes a new file with a name

del file -deletes a file

rmdir folder – deletes a full folder

copy file- copies a file

move file folder – moves a file or folder

How Git works

One main folder and the files/folders with that

That is our working directory

Saving the first version- you do that via a commit

It has to have a message

You can create a second commit once you have made changes

Git in on VSC

You can check your git version eg git –version and it will show you what version you have

Git status gives you info about your working directory

To innitiolize you need to use git init

Then git add .

Getting commit id- git log you can see the commit ID

It is also labeled the head, wich is the last commit

Branches

A unique set of code changes

Can save different commits to different branches

To see al branches eg git branch

New branch eg git branch second branch

To create and change to that branch git checkout -b second branch

Merging branches

Git merge you must be in the branch that everything must end up in

Git merge and then the other branches name.

Git merge second branch

Attached and detached head

Last commit is the head

Detached head- is a commit that is not part of a specific branch

Commands with branches

Git switch allows to switch and create new branches, same as git checkout -b branch2

Eg git switch – second branch

To create a new one eg git switch -c second branch

Deleting branches

Deleting a file from staging area

First check eg git ls-files

Git rm with the file name

Undo unstaged changes

git checkout revers unstated changes

git restore, will do the same

to delete a file, git clean the specify what must be cleaned

eg git clean -dn to check

then git clean -df to force the delete

undoing staged changes

git reset simply copies the latets copy of this file into the staging area

git restore helps restore a earlier verson of the project

eg git restore --staged file name

deleting commits with git reset

git ls-files

shows all files in the staging list

git reset—soft HEAD-1 to show how far it must go backa9the number)

default way

git restet HEAD-1

the commit gets deleted

different way

git reste –hard HEAD-1

Removes everything everywhere

Deleting Branches

To check branches use git branch

To delete git branch -d (to delete merged branches)

Git branch -D (to delete the branch it doesn’t matte if it was merged or not)

Add file name after the d’s

Delete multiple branches you just use the two branches names

Commiting a detached head

To view a specific commit use the commit id

Eg git branch branch name

Understanding .gitigore

You have to greate it for yourself and you have to stipulate what is not needed for the project

.gitignore- once you put a file name in there it will be ignored by git and with that will not be moved to the staging area

If you put \*.log all files that end with .log will be ignored

If you want a specific file that ends with log you need to specify it

There are 3 different type of files

Ignored,tracked and untracked

You can also let git ignore full folders

Summery of module

General commands

Git –version to see what version has been installed on the system

Git init lets you greate a empty repository

Git status check working directory

Git log display all commits for the branch you are in

Git ls-files list the data in the staging area

Git add .

Git add file name

Git commit -m “message”

Git checkout commitid you can see a specific commit

Git branchname greating a new branch

Git checkout branchname go to a specific branch

Git checkout -b branchname grate and go to the new branch

Git mergs otherbranch bring other branches changes to current branch

Understanding the stash

Stash is internal memory and you can save unsave or unstaged changes

Eg git stash, this will save and clears your progress that you have made and to apply these changes its git stash apply. You get the latest one back

Git stash list, shows all your stashes

To add a message, git stash push -m “message”

With this you can identify the stashes

Git stash pop will remove from the stash list and add to the project

To delete its git stash drop and the number of the item on the list

To clear the stash its git stash clear

Bringing lost data back with git reflog

Allows us to bring back deleted branches

Understanding merge types

Fast forward

Non fastforward-recursive

Octapus

Ours

Subtree

Ff merge- only work if you have no additional commits after you have created the new feauter branch

Git branch you see all your branches

Git merge to merge the branches

The head moves to the last commit

Git reset –HEAD-1 lets you reset the hea to a spefic branch

Git merge –squash lets you only merge the latest commit

Git recursive, git merge --no-ff means you want to merge but not use the fast forward merge

This merge will also become the head

You use this when ypu also have commits in the master branch after ypou have created commits in the master branch

Git rebase,

Does not move commits it greates new commits

Merging conflics

Durig the merge- once a merge happends vsc will give you a choice to fix the conflicts and with that you can then commit after that but only once you fix the conflict

Cherry picking copared to rebase and merge

Merging- creates a commit and is a new commit

Rebase- changes a single commit parent and creates a new commit id for everything that has been rebased

Cherry picking ads a spesifc commit to branch head and copies a new commit id

Merge vs cherry picking

Cherry picking- git cherry-pick then commit id

Then only that commit will be added

Additional commands

Tagging commands

Git tag with a description of the tag then the commit id

Git show with the description will show all info about the commit

You can use git checkhout show

To delete it use git tag-d

Annotated tag

Git tag -a then description and to add a message you add with -m “message”

Conclusion of section 4

Git stash means- temporary storage for unstaged and uncommited changes.

Git reflog – project log that logs everything even deleted data that can be braught back

Git merge - combining different branches

Git rebase- changed the base of commits in another branch

Git cherry pick- copies a commit including the changes and then adds it to the head of a different branch with a new commit id