Git init

Github

Install Git

<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

install and at the end launch git bash

This gives you git terminal (like cmd )

Use it as a terminal – not only for GIT but also ca run python

A screenshot of a computer

Description automatically generated

First create afolder and open Gitbash from the folder

First time – we have yo set name and email

CommandA black background with yellow and white text

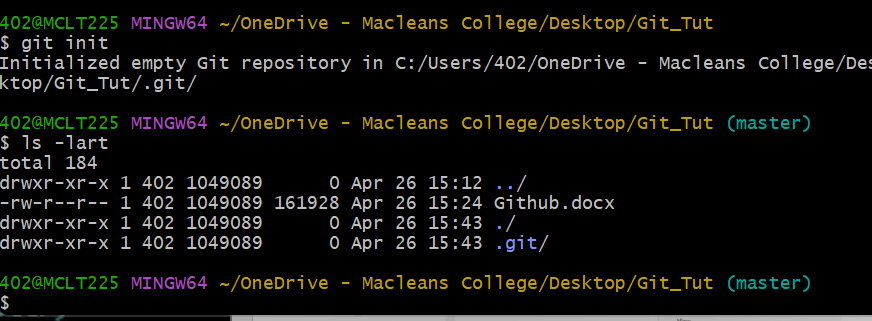
Description automatically generated

Check if usernames are configured correctly

Next create a git in the folder you creates using init command

A computer screen with text and numbers

Description automatically generated



A computer screen shot of a program

Description automatically generated

Now open the editor with the following command

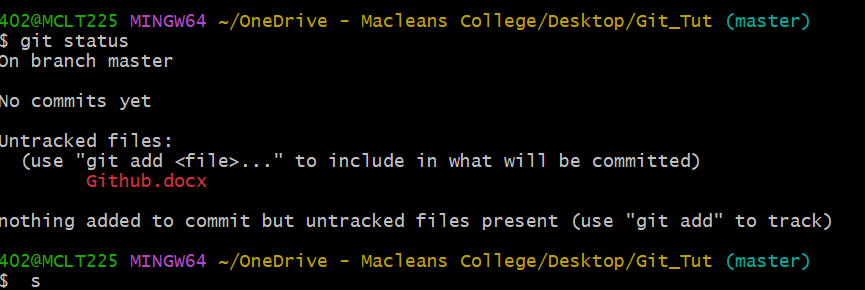


Any editor can be used

A computer screen with text and numbers

Description automatically generated

Now checking status thru command



Understand the following.

A screen shot of a computer

Description automatically generated

Untracked – you are not tracking your files with Git

File added to Git – and keep the changes recorded.

Commit – save

Staging area gets the committed file

Follow the arrows

$git add index.html

File has gone to staging area

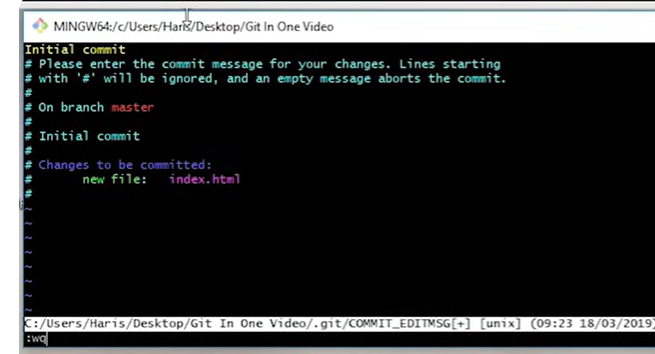
Git commit will track the file through Git

As u type git commit on bash it will take you to … ($git commit)

You will enter in Vim editor

Use I to activate and write Initial commit

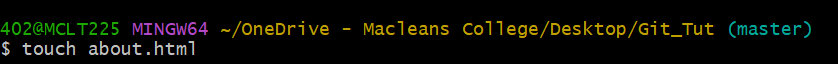
To quit :wq at the end



A computer screen shot of a computer code

Description automatically generated

Next touch command creates a new file



Three files created

A screen shot of a computer

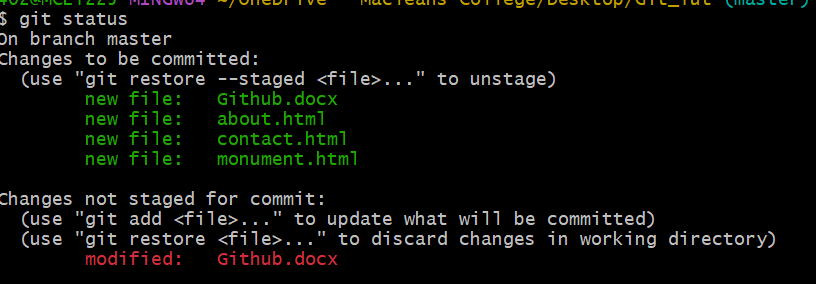
Description automatically generated

On adding these three files and checking git status A black screen with white text

Description automatically generated

To add all files to the staging area use 

Check the status



It tells that the files are waiting to be committed

Ready for commit now – last time we did using vim editor – now we are using a shortcut

Git commit -m “added more files”

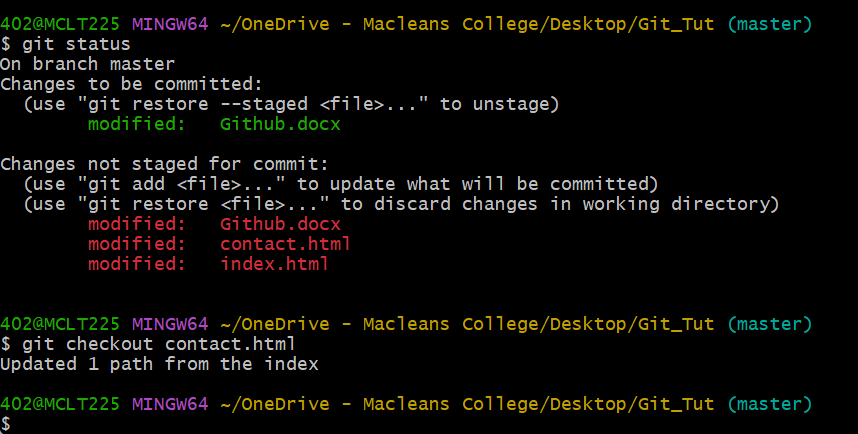
A screen shot of a computer code

Description automatically generated

Checkout command to recover – last matched commit

So no accidental changes savd

Git checkout filename

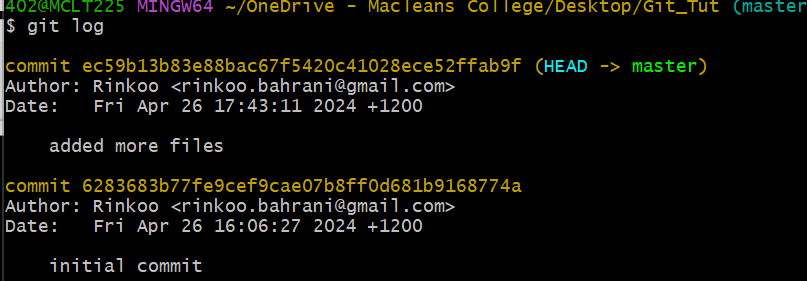


Git checkout -f

All files will math the previous commit

Next is git log

It tells you tells you all the commit messages for all the files



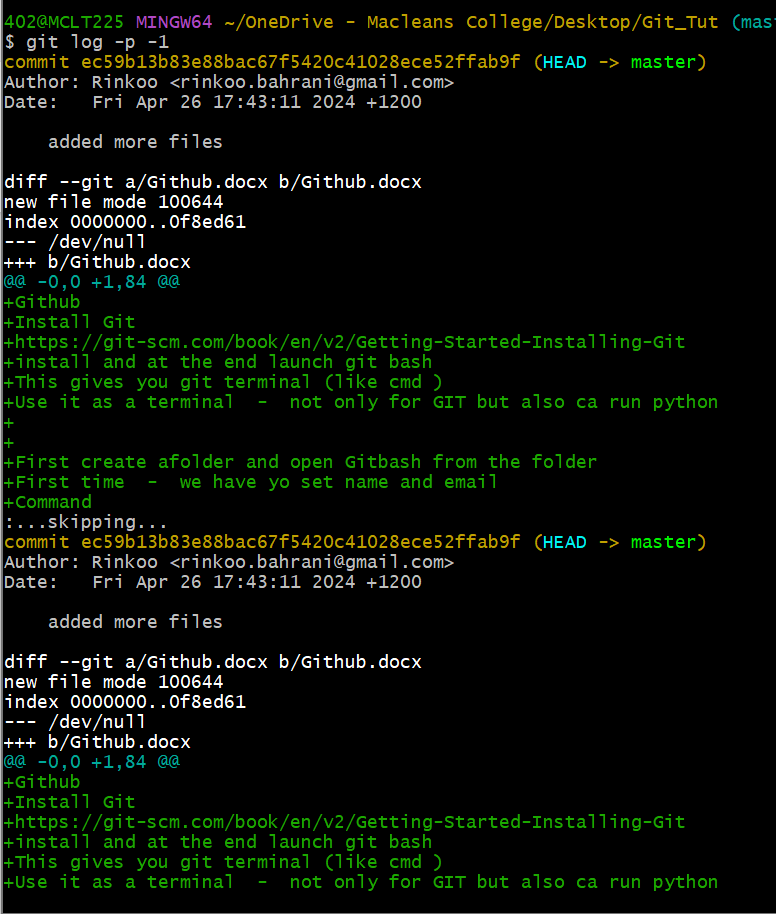
Next is if you want to filter the output of gitlog

Git log

Git log -p -1 (see one commit)

Use git log -p -1

Shows last one commit as done



‘q’ to quit out of this

This is telling what all changes are made to the files

Eg



A computer screen shot of a computer code

Description automatically generated

What is Git DIFF

Compares working tree with the staging area

As I make changes in index.html – it will (git status) will show the change and (git diff) will compare and show the changes made in the file



Next is to compare staging area to the last commit

Git diff –staged

Git status

Git checkout -f ------ matches all files from last commit

Git status

The gen process is to - add to staging area and then commit

You can avoid this by adding all files to commit

Git commit -a -m “skipped staging and fixed “

Clear is the command to clear the screen of command bash

Ls is to list all the files

Git commit -a -m “skipped staging area and fixed” will add directly to commit

Git rm filename will remove the file from staging area and commit

Git rm –cached filename will be removed only staging area but it will remain in memory.

Git add -A to add the file

Short status

Git status -s

Says files that are modified

Add file to staging area by git add contact.html

And then again run Git status -s

A black background with white text

Description automatically generated

Green M shows file is modified in staging area

Red M shows file is modified in working tree

A computer screen with white text

Description automatically generated shows file contact.html is modified in working tree and also in staging area.

Git ignore command

Touch .gitignore

Git ignore – there are some files that need not to be tracked- not to be shared with collaborators

Sy you create a folder logs mylogs file





Gitignore will store the names of the files that you do not want to track

Like mylogs.log

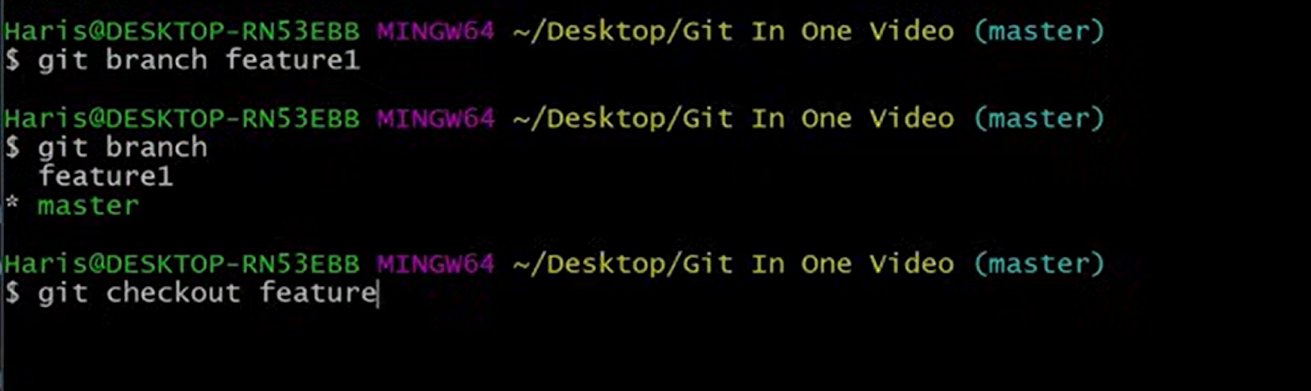
It is unnecessary to push and pull those files - and slow down the process

Branches

Create copies

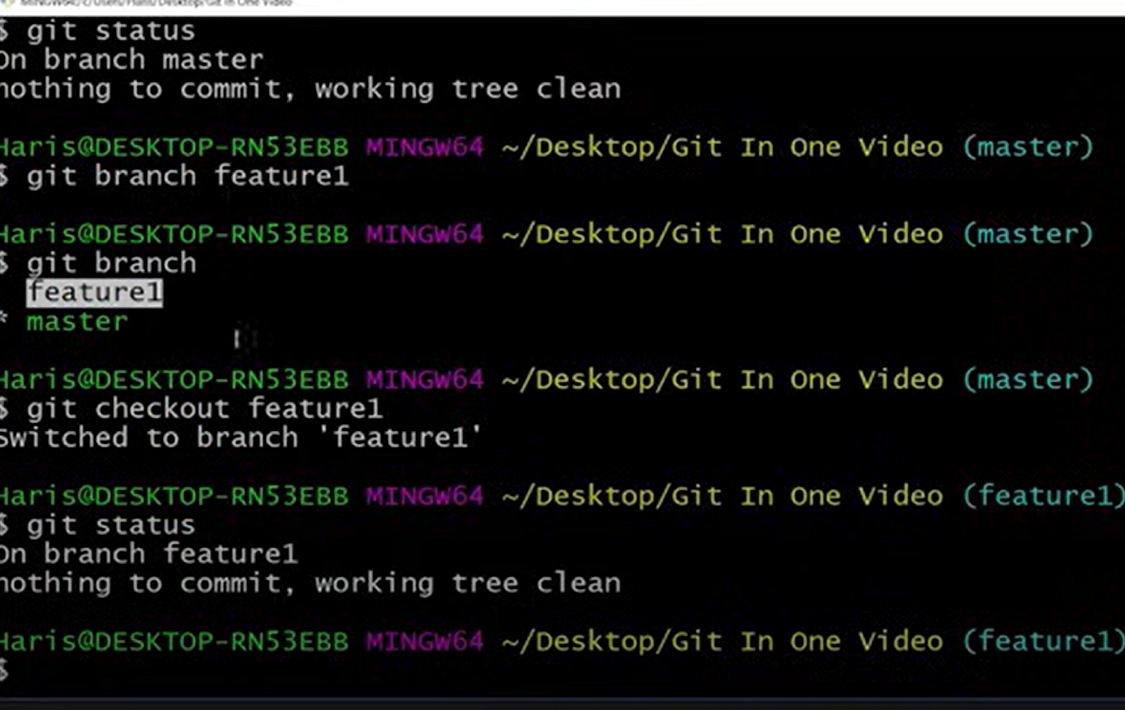
You create a new copy and keep working

Master is the main branch



We add features as feature 1 in our project and work in our copies

As we see it fits well and does not break our code – we merge features into our branch



To switch to feature 1 brach

Git checkout feature1

Git commit -a -m “added comment”

Git checkout master

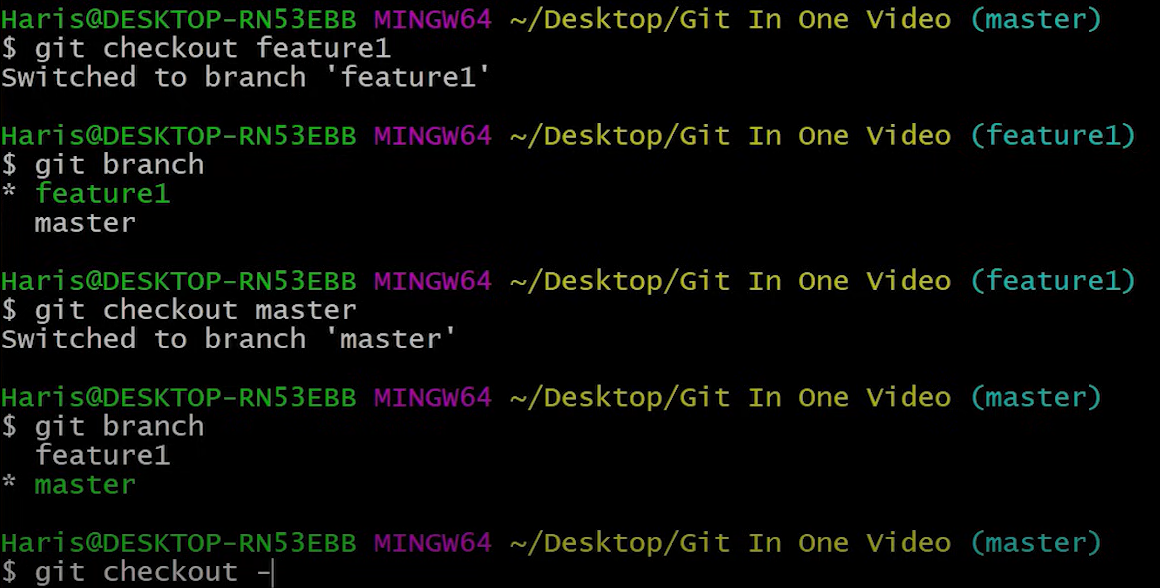
Merge

Come to master Git checkout master

Git merge feature1

Checking the status git log -p -2

How to switch between two branches feature1 and master and to check which branch are we in.



Command to create a new branch and enter in that branch

Git checkout -b flaskintegration

Means create and move/switch to flaskintegration branch

Create github account

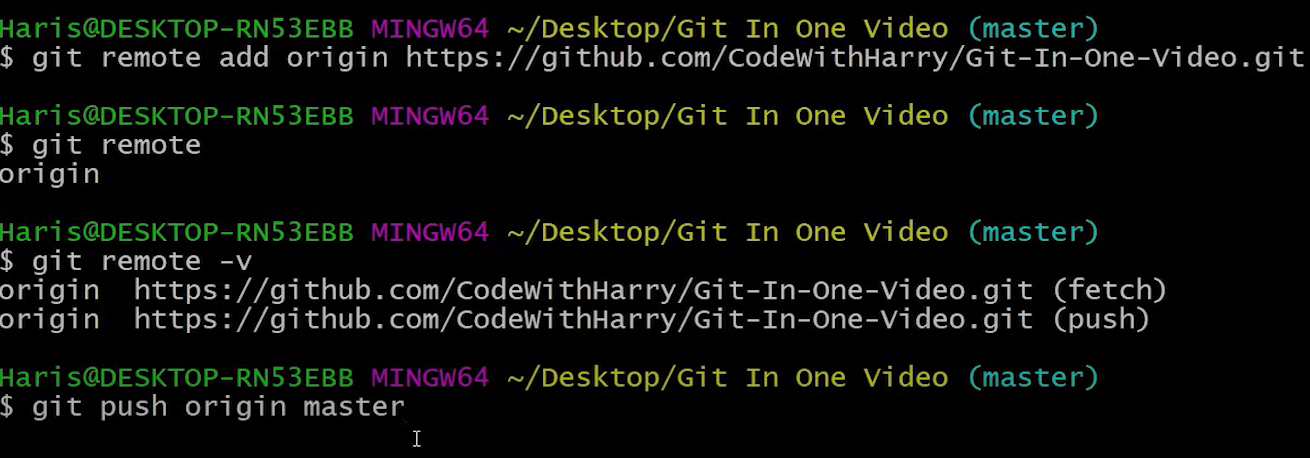
Create repo onguthub(remote)\

From your local git use the following command to link local git to remote(cloud) git



A screenshot of a computer

Description automatically generated



To give access from computer to cloud Git

