How To Go About With Git

Things to remember:

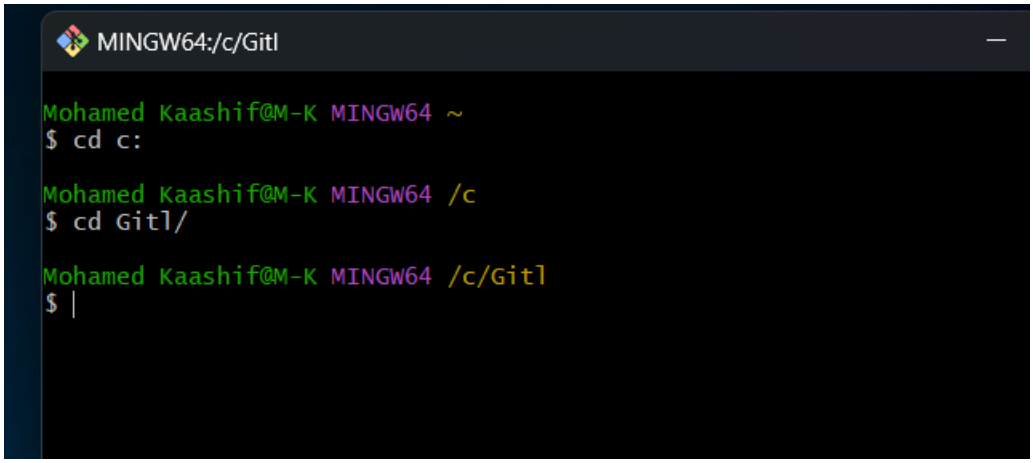
- Git is an open source version control system to track code changes
- **Repository:** a storage location used to track and manage the changes made in a file.

Things to do before starting any program:

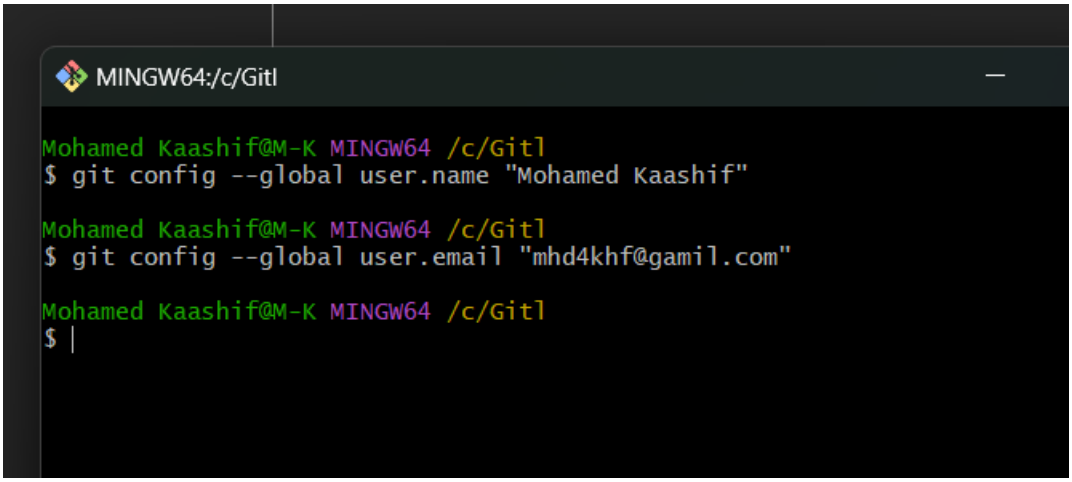
- Create a folder on the desktop or in any preferred location on your system.

(Ive created a “Gitl” folder located in my ‘C:’ drive)

- Open Git Bash Terminal and move into that folder with the help of **cd** command.



- Here configure your name and email so that git can track who has made the changes, use the command
    a. **git config –global user.name “Your Github account name”**
    b. git config –global user.email “Your email”

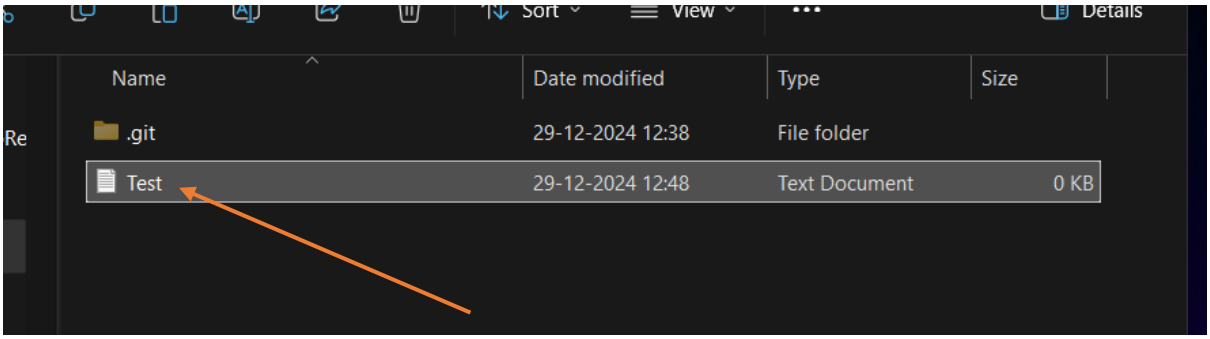


- Initialize git int this folder with the command,
    a. **git init**

Program 1

FOLLOW THE ABOVE STEPS AND THEN:

- Add a file in the folder you've created ("Gitl") from the file manager



- Get back on git bash and use the command
    a. git add . (or) git add Test.txt

(git add . adds all the files present in the folder)

- This adds the Test.txt to the staging area.

```
Mohamed Kaashif@M-K MINGW64 /c/Gitl (master)
$ git add .

Mohamed Kaashif@M-K MINGW64 /c/Gitl (master)
$
```

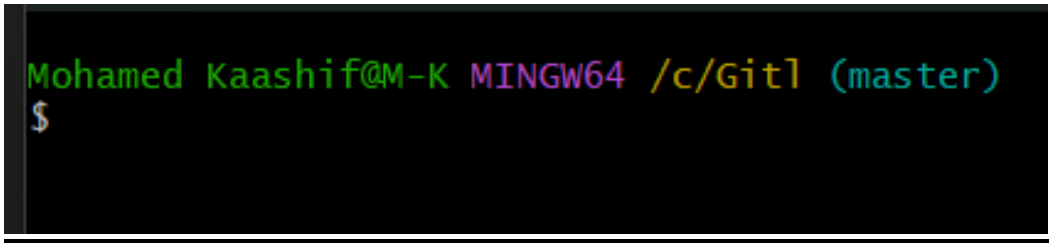
- Finally commit the change with the command
    b. Git commit -m "Your message"

```
Mohamed Kaashif@M-K MINGW64 /c/Gitl (master)
$ git commit -m "Added a file named Test.txt"
[master (root-commit) 6dafe56] Added a file named Test.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Test.txt

Mohamed Kaashif@M-K MINGW64 /c/Gitl (master)
$
```

Program 2

AFTER INITIALIZING GIT IN THE FOLDER,

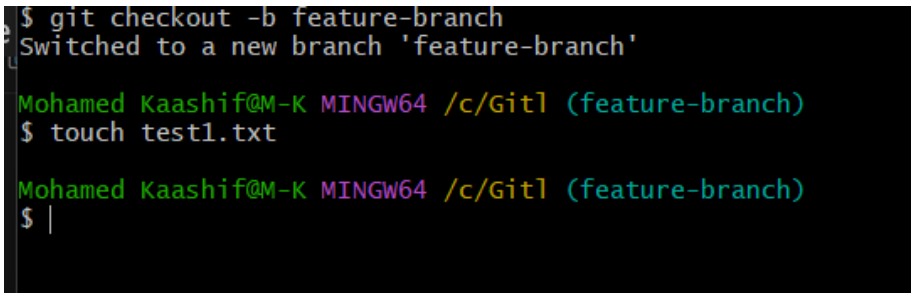


- If you terminal doesn't show the "(master)" at the end, simply use the command
    1. **git checkout master**

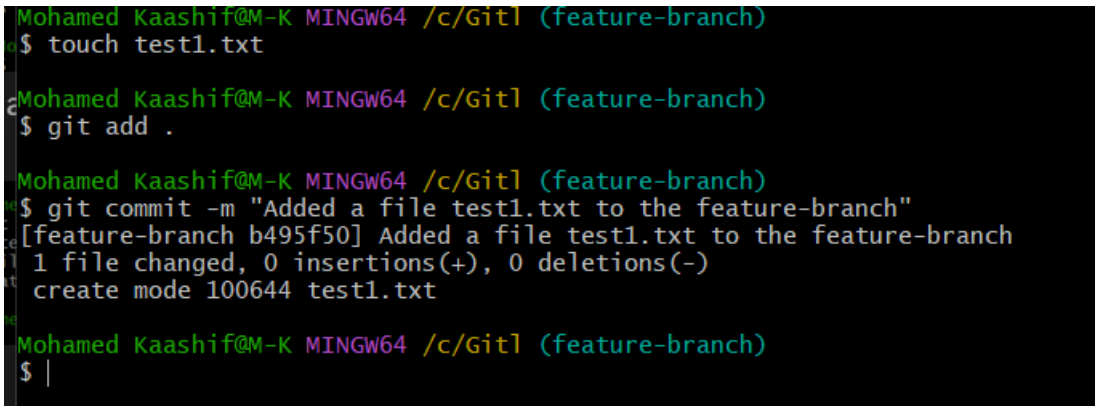
    By doing this, you'll get the same as above.

- Create a branch named feature-branch (any name is fine) and move into it with the help of a single command
    2. git checkout -b feature-branch

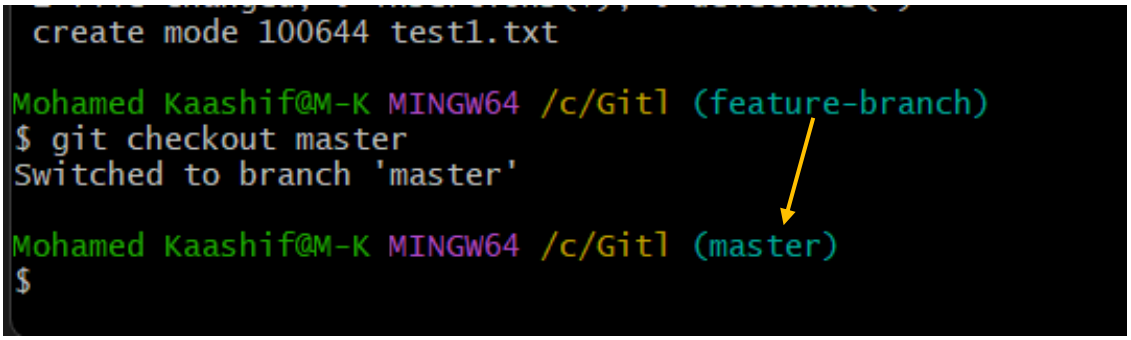
- Once your in the feature branch, add a file in it by using the command
    3. touch test1.txt



- Add this test1.txt to the staging area and commit it with a message.(This file is only available to the feature-branch)

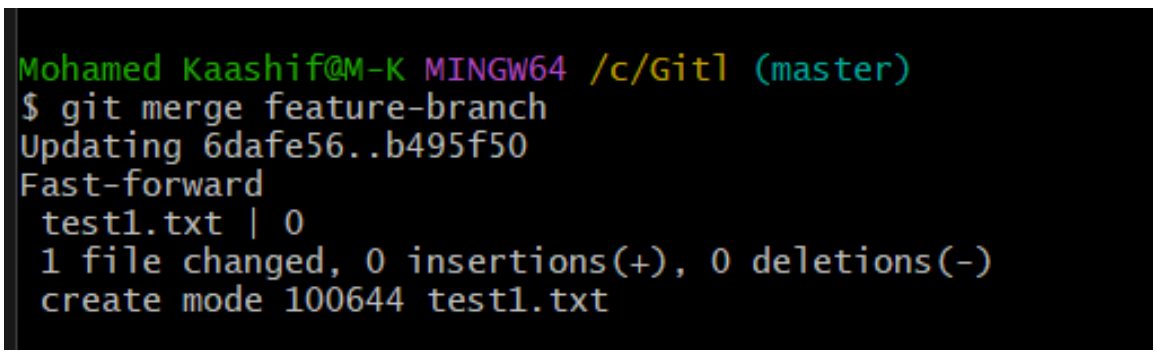


- Next checkout back to the “master” branch



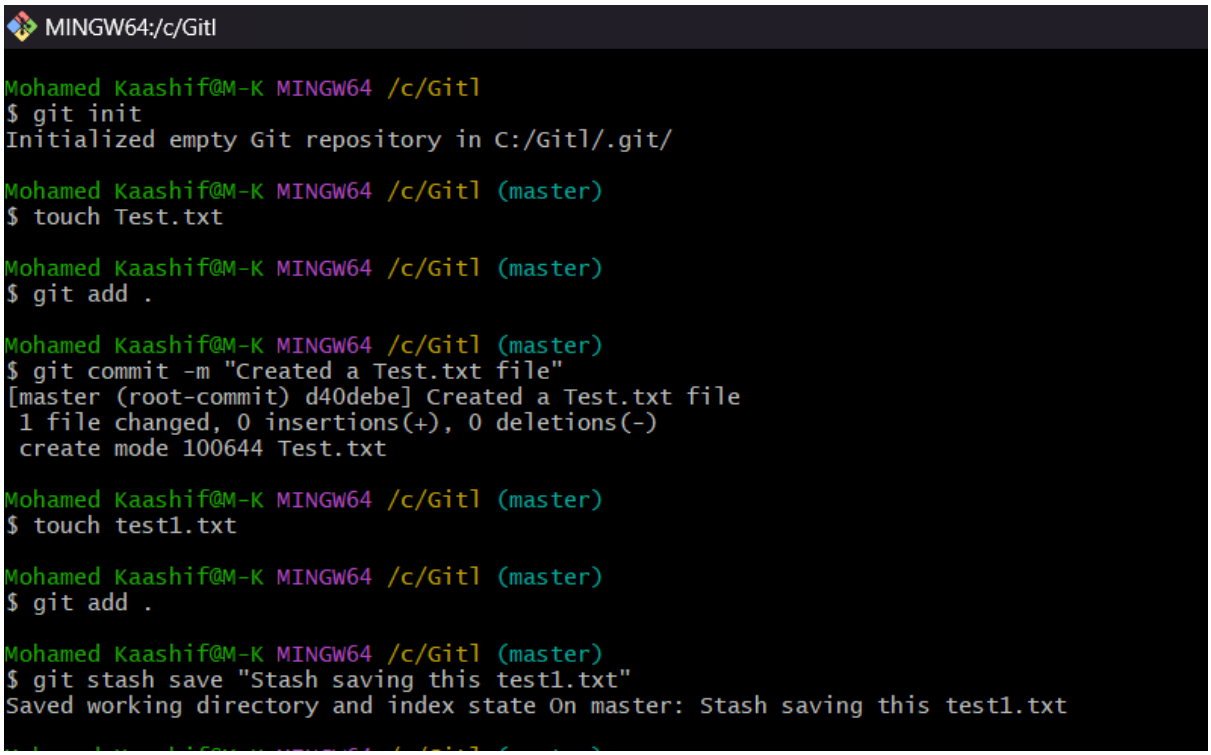
- Finally merge the feature branch to the master branch by using the command
    4. git merge feature-branch

(This will now add the test1.txt which was only available to the feature branch to the master branch)

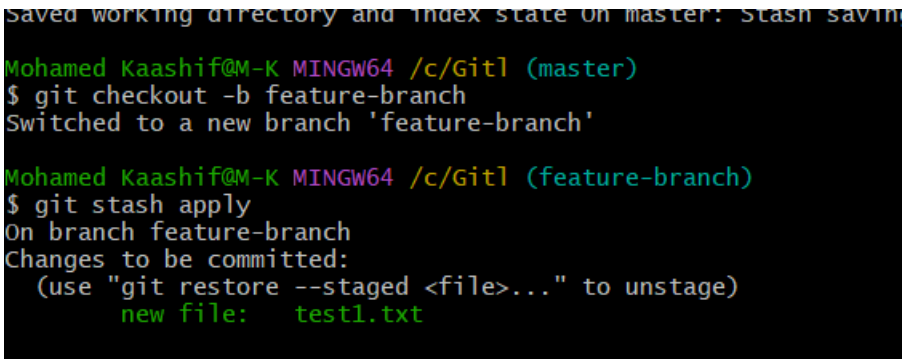


Program 3

- Here we will make a file Test.txt and a file test1.txt on the master branch. Next we will commit the Test.txt and stash save the test1.txt by adding it to the staging area.



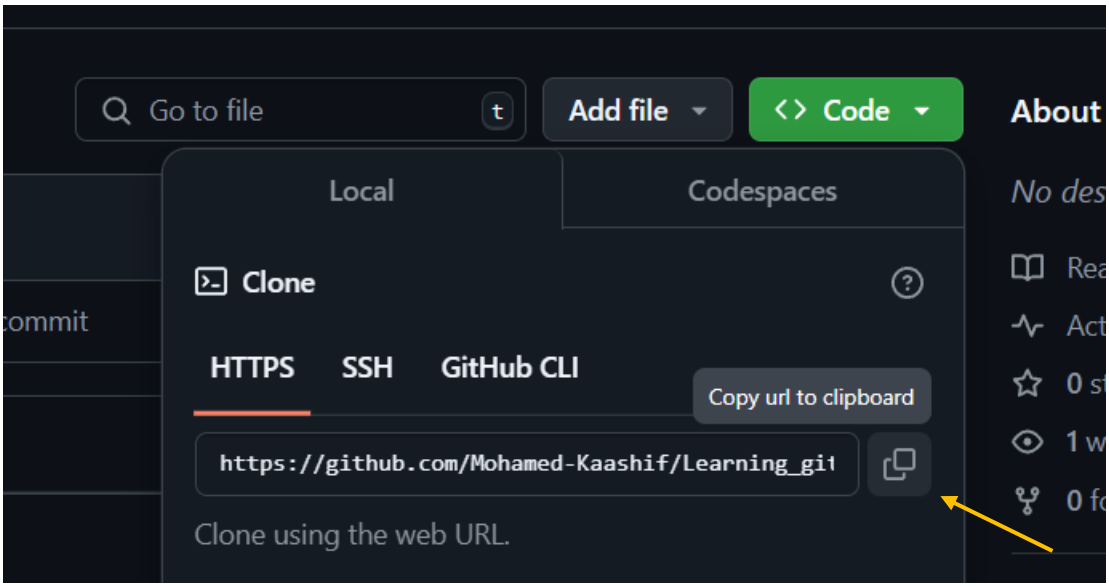
- Then we'll create a new branch called feature-branch and move into it and apply the stash changes



Program 4

FOR THIS ONE PROGRAM WE CAN SKIP THE PROCESS OF INITIALIZING GIT.

- Go to the github's website and copy the url of any project you'd like to enhance or work on.



- Next head to git bash and use the command
    1. git clone *url*

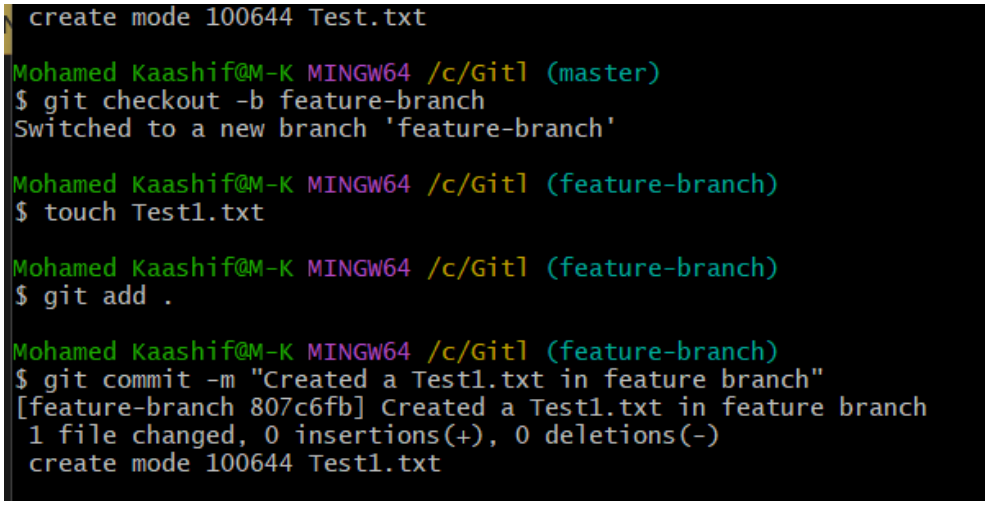
```
Mohamed Kaashif@M-K MINGW64 /c/Gitl (master)
$ git clone https://github.com/Mohamed-Kaashif/Learning_git.git
Cloning into 'Learning_git'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

Mohamed Kaashif@M-K MINGW64 /c/Gitl (master)
$
```

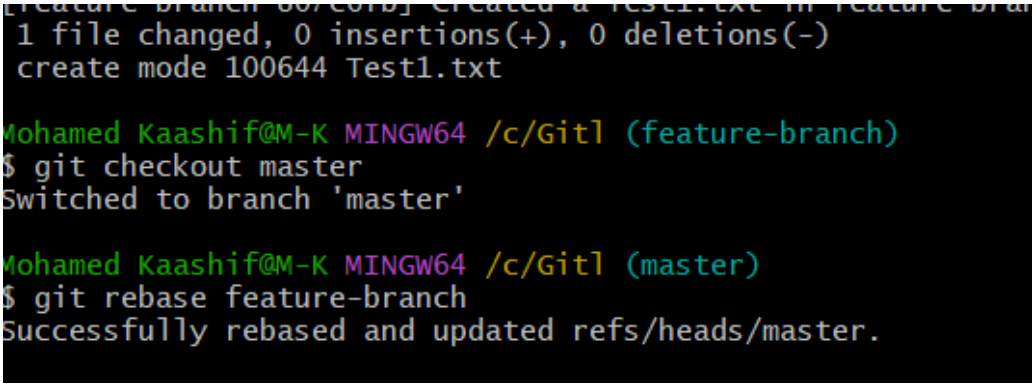
Program 5

- Here we create a file Test.txt in the master branch and commit it.

- Next we create and move into a feature-branch and add a file Test1.txt. commit this Test1.txt as well.



- Next we will checkout to the master branch and use the command **git rebase feature-branch**



Program 6

SAME AS PROGRAM 2

- Here the only added feature is that we are adding a message wile merging the feature-branch's file.

```
Mohamed Kaashif@M-K MINGW64 /c/Gitl
$ git init
Initialized empty Git repository in C:/Gitl/.git/

Mohamed Kaashif@M-K MINGW64 /c/Gitl (master)
$ git add .

Mohamed Kaashif@M-K MINGW64 /c/Gitl (master)
$ git commit -m "Added a Test.txt file"
[master (root-commit) 8fd7605] Added a Test.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Test.txt.txt

Mohamed Kaashif@M-K MINGW64 /c/Gitl (master)
$ git checkout -b Feature-branch
Switched to a new branch 'Feature-branch'

Mohamed Kaashif@M-K MINGW64 /c/Gitl (Feature-branch)
$ touch test1.txt

Mohamed Kaashif@M-K MINGW64 /c/Gitl (Feature-branch)
$ git add .

Mohamed Kaashif@M-K MINGW64 /c/Gitl (Feature-branch)
$ git commit -m "Added a file test1.txt to the feature-branch"
[Feature-branch 829b71d] Added a file test1.txt to the feature-branch
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test1.txt

Mohamed Kaashif@M-K MINGW64 /c/Gitl (Feature-branch)
$ git checkout master
Switched to branch 'master'
```

- Once we've got here, then we simply need to use the same merge command with a flag (i.e -m to add a message while merging)

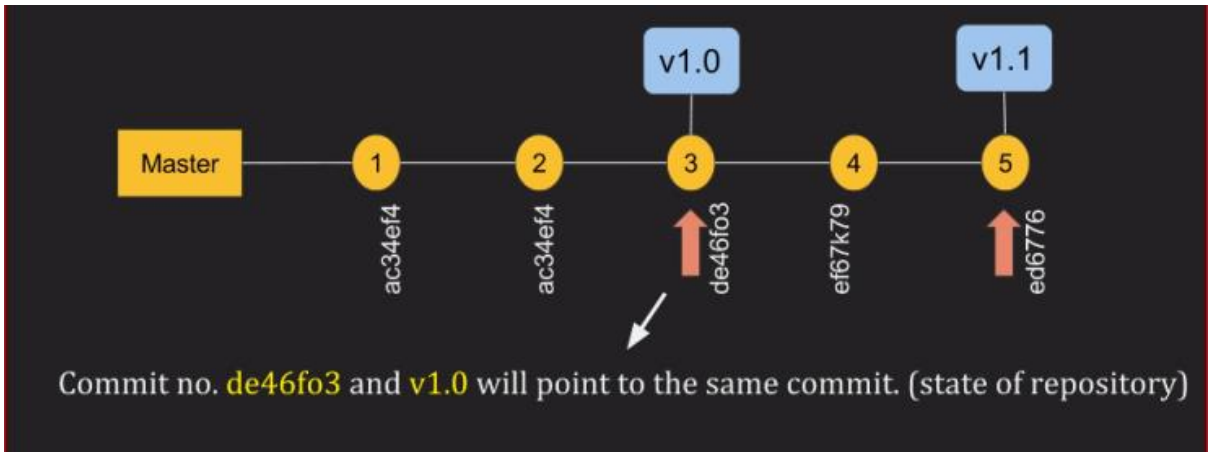
```
Switched to branch 'master'

Mohamed Kaashif@M-K MINGW64 /c/Gitl (master)
$ git merge feature-branch -m"Adding the test1.txt to the mastger branch"
Updating 8fd7605..829b71d
Fast-forward (no commit created; -m option ignored)
 test1.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test1.txt
```

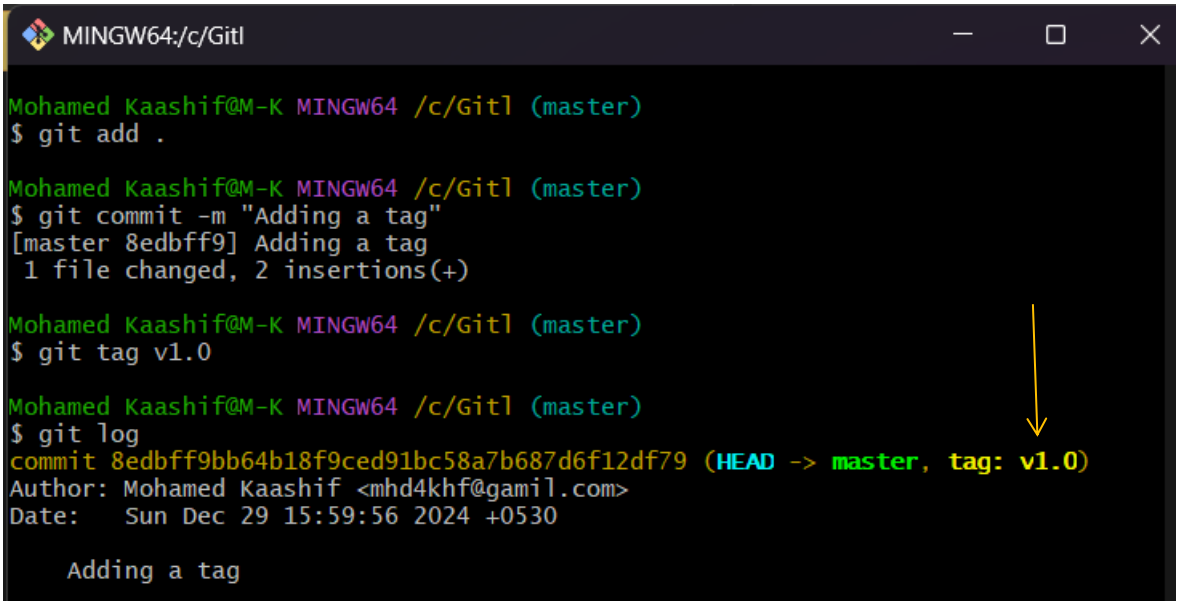
Program 7

INITIALIZE GIT IN A FOLDER AND MAKE A COMMIT (AS DONE IN PROGRAM 1)

- Tags are added to a specific commit something like a mile stone.



- To add a tag, simply use the command
    1. git tag *version*

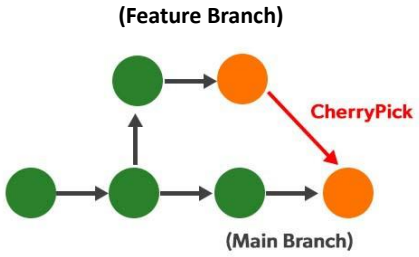


(the “git log” command shows all the commits made by the user)

Program 8

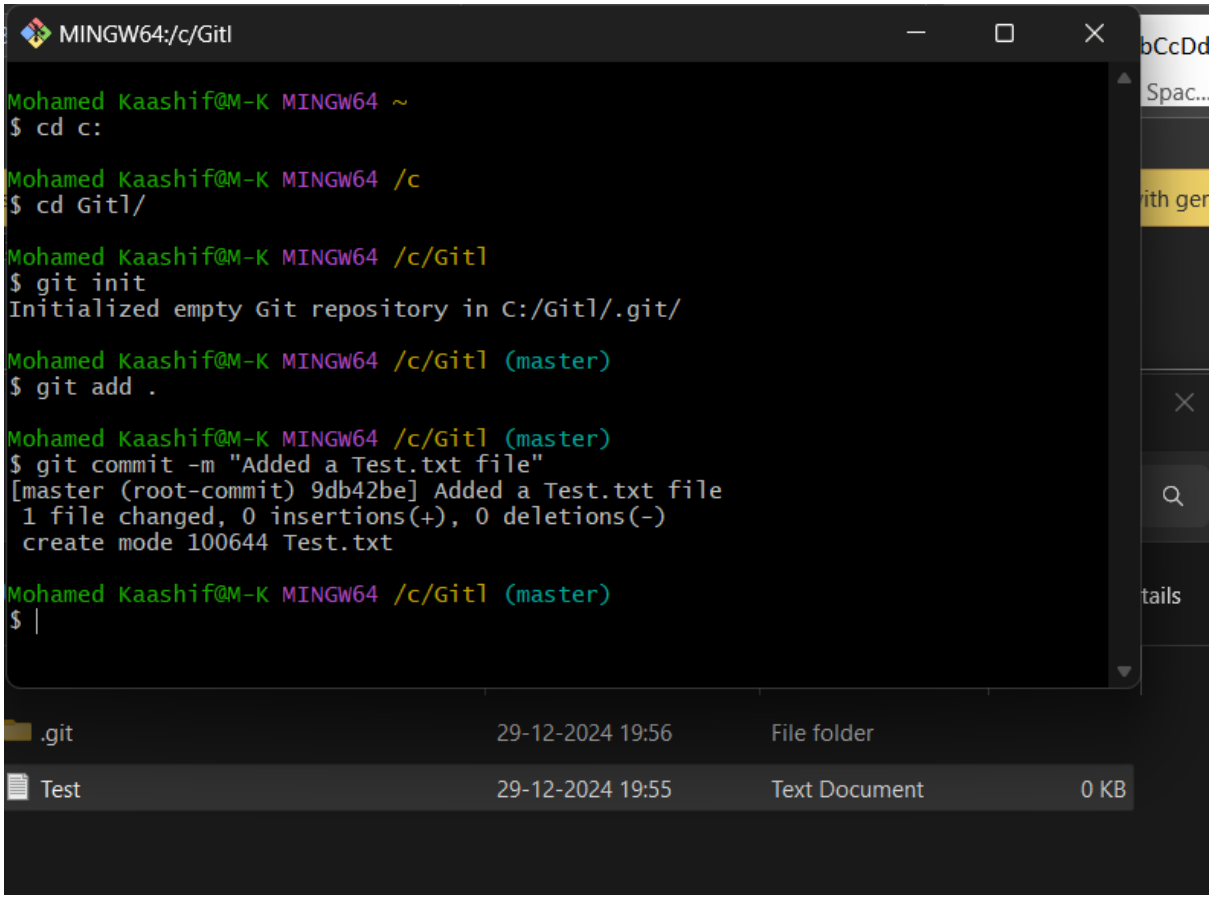
CHERRY-PICK

- Here, we create and switch from the main branch to a feature branch, add in a file and commit the change.
- Once that's done, we switch back to the main branch and use the cherry-pick command.



To begin,

- We first repeat the process as program 1 (initialize, create a Test.txt and commit the changes)



- Next create and move into the “feature-branch” with the command, **git checkout -b feature-branch**

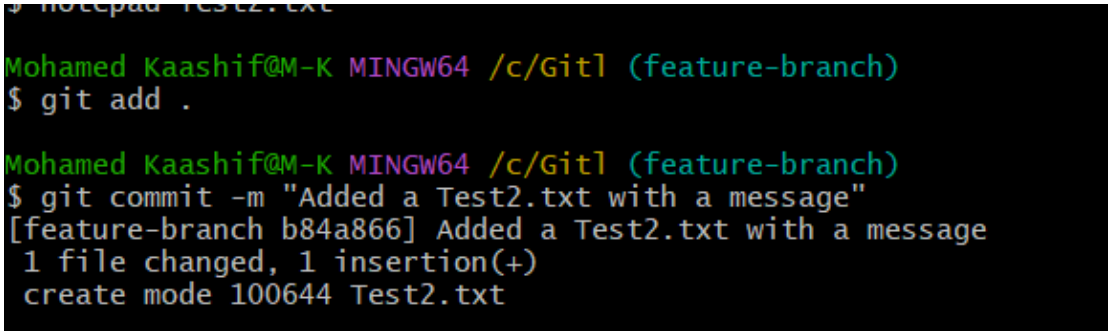
```
Mohamed Kaashif@M-K MINGW64 /c/Gitl (master)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
```

- Create a new file here called “Test2.txt” with the “touch” command.

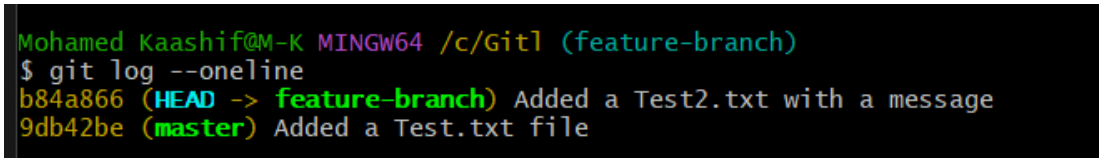
```
Mohamed Kaashif@M-K MINGW64 /c/Gitl (feature-branch)
$ touch Test2.txt
```

- Open the “Test2.txt” file with the help of the notepad command

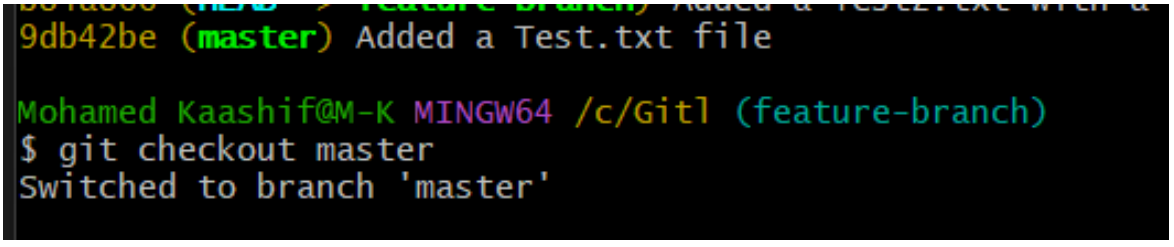
- After making some change in the Test2.txt file, save the it and close the notepad window to get back on git's terminal.
- Add it to the staging area and commit the changes.



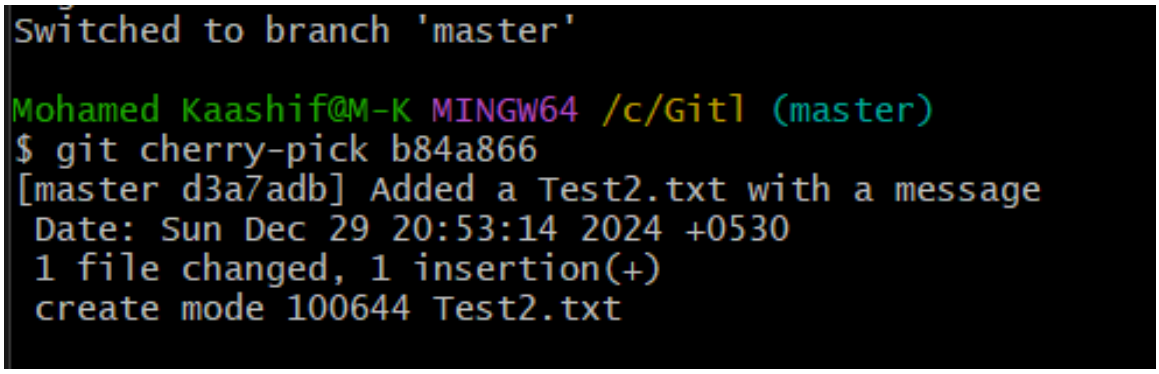
- Next, use the command git log –oneline, to get the commit ids of the feature branch



- Now checkout to the "master" branch



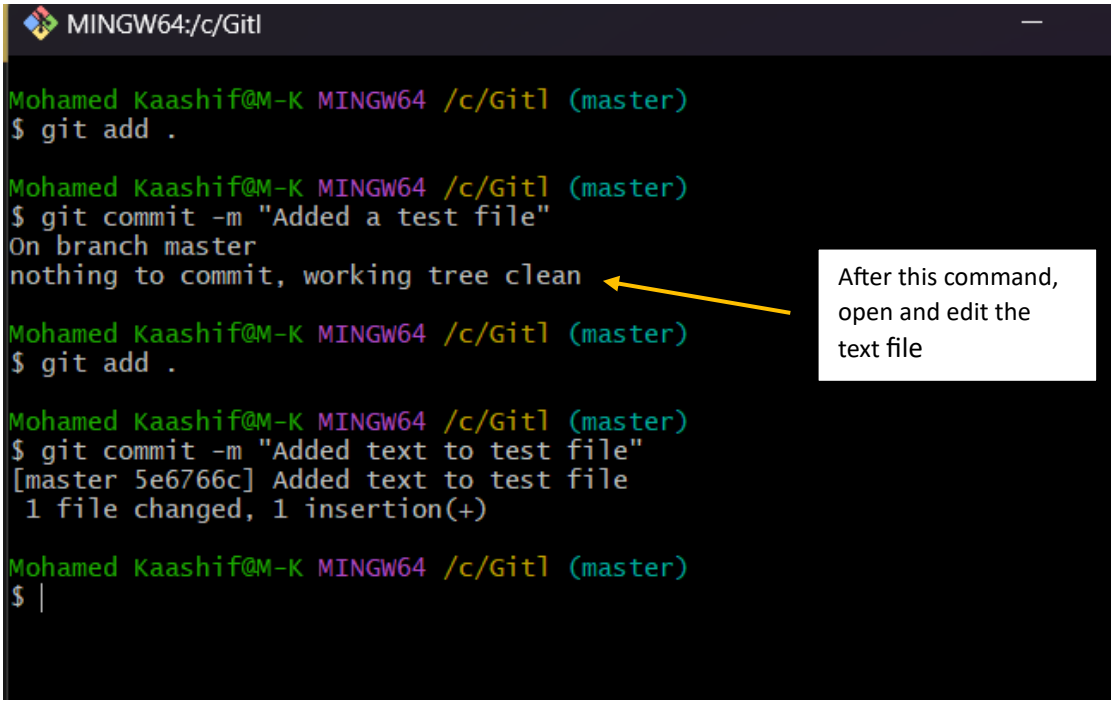
- Copy a commit id from the feature branch (that we got on using the command git log --online)
- And use it with the command, **git cherry-pick *commit_id***



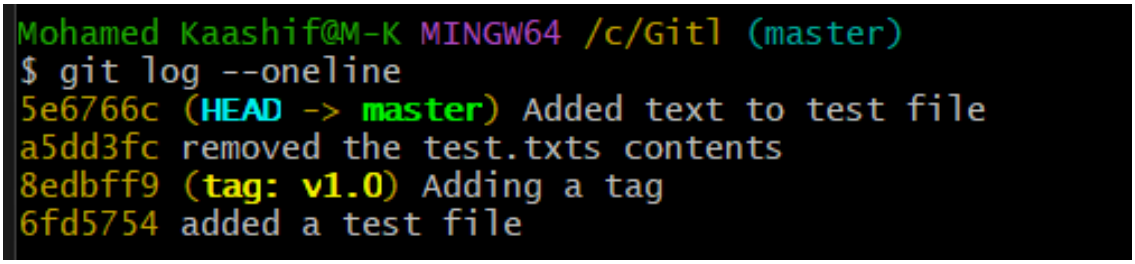
Program 9

DO PROGRAM 1, AND THEN

- Open the Test.txt from your file explorer and add any text. Save the file and on git bash add it to the staging area and commit it.

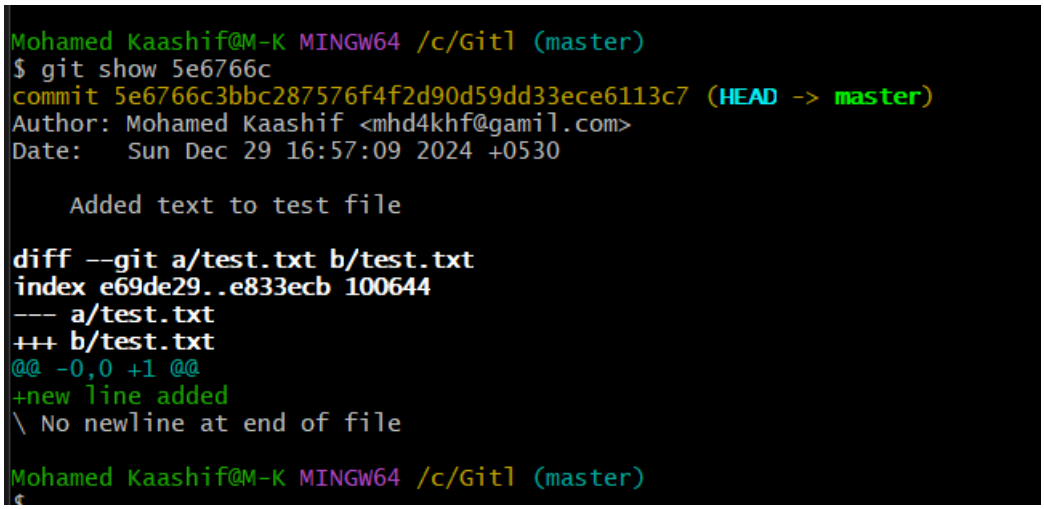


- To view the commit id's use the command
    1. git log --oneline



- Copy any of the commit id and use it with the command

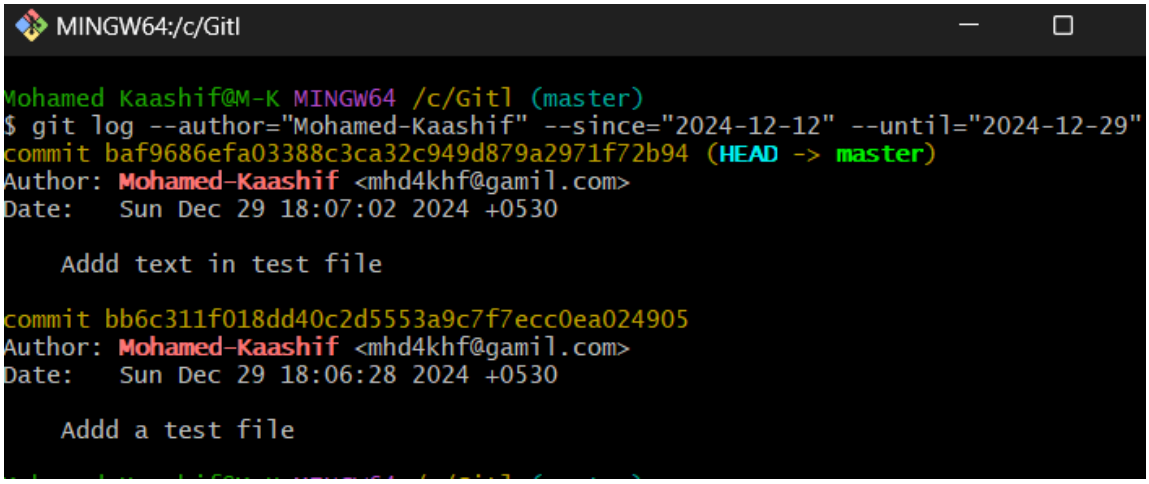
    2. git show *commit_id*



Program 10

USING THE COMMAND ‘GIT LOG’

- Perform at least 1 commit in you repository.
- Next use the command
  1. git log --author=”*the_user_name_you_gave_while_confi guring*” –since=”*any_date*” –until=”*current_date*”

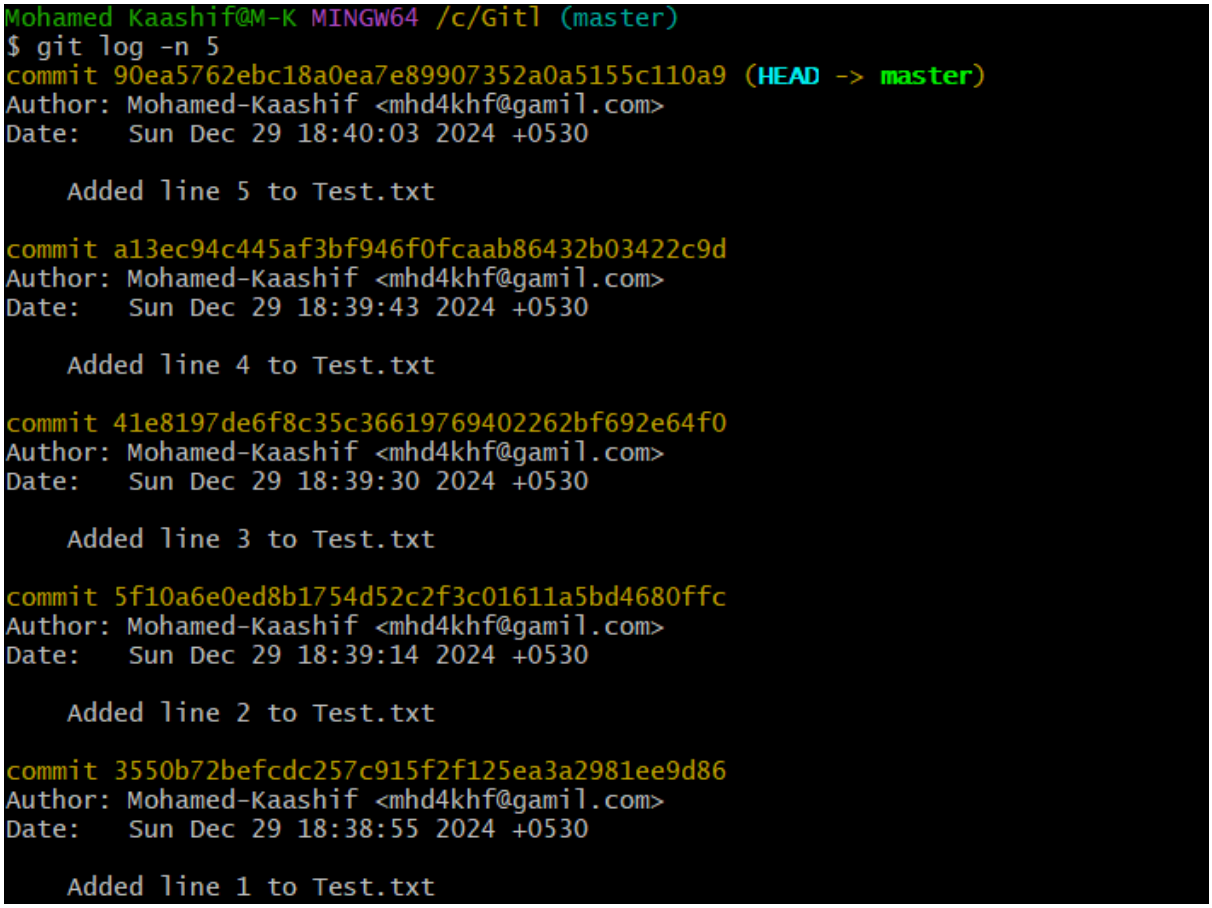


Program 11

MAKE 5 COMMITS IN A REPOSITORY

- To make 5 commits, you can add in one line or keep making a change in the “Test.txt”.

- Then use the command
  1. git log -n 5

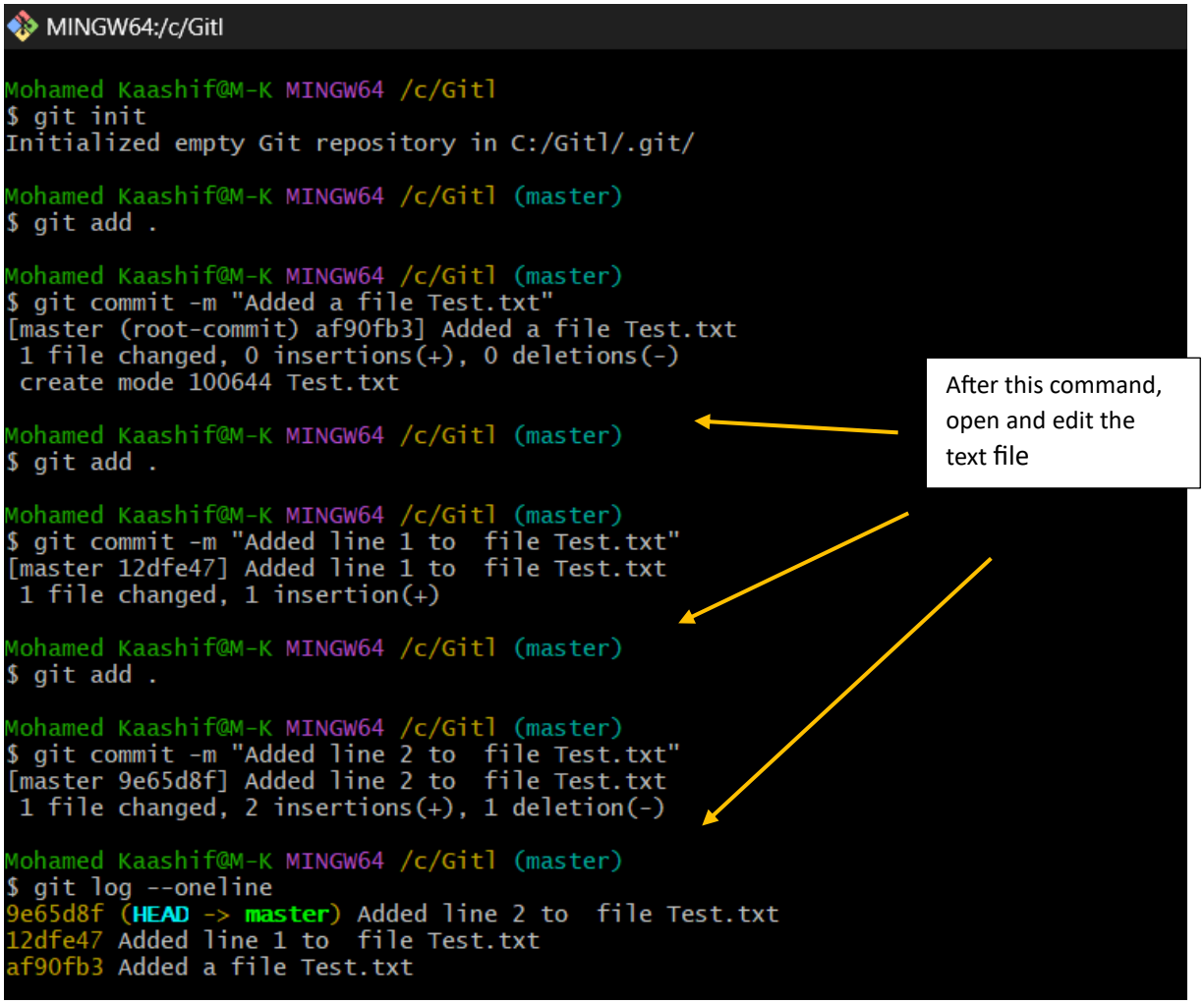


Program 12

MAKE ATLEAST 1 COMMIT TO REVERT

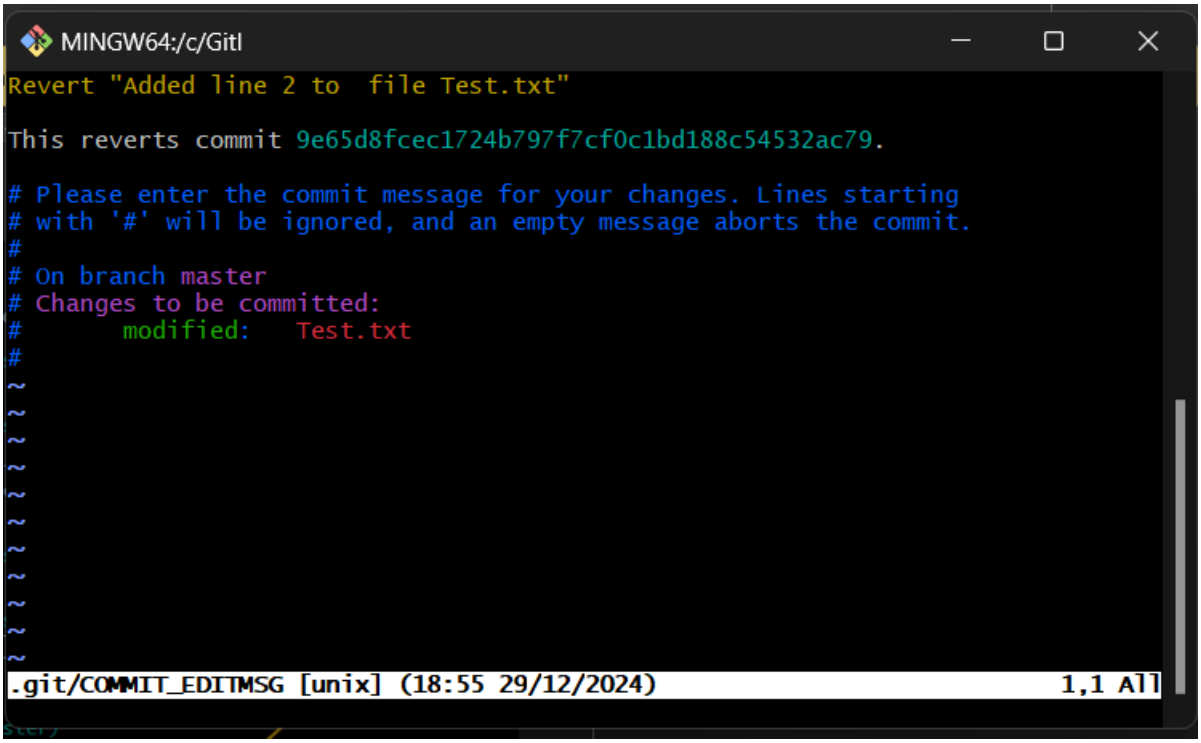
- After making a commit, use the command

  1. git log --oneline

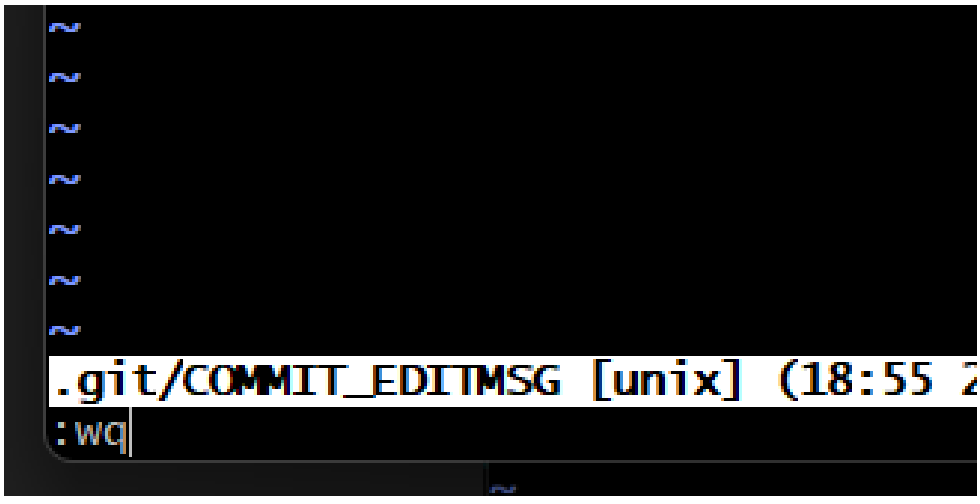


- Copy any of the above commit ids
- And use the command,

  2. git revert *commit_id*



- On getting this screen just enter ":wq" and press the enter key



- And you'll get this as the output