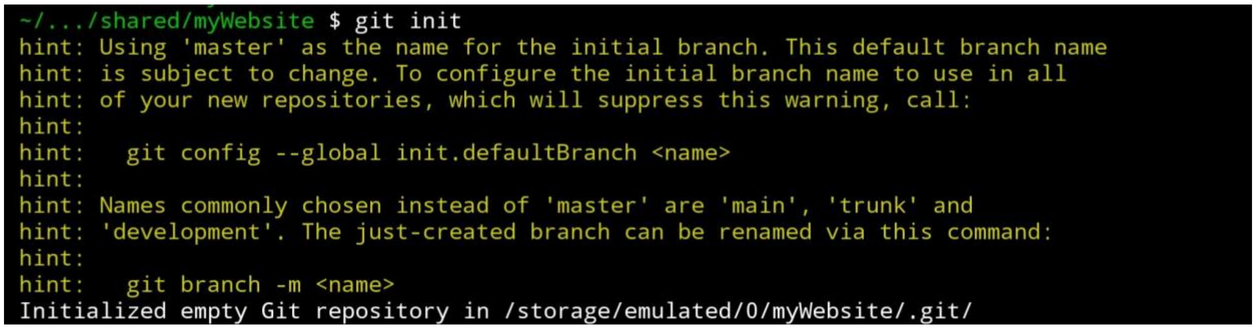
# ***Git Notes for Beginners***

1) Installing git in your system
   - If you're on windows/mac download git in your system from here.
   - If you're on android
     - you need to download an app called **termux** from here and then type the following command
     - pkg install git    //this will install git in your system
2) Telling who we are
   After installation its time to introduce ourself to the system
   git config –global user.name "firstname lastname"
   git config –global user.email youremail@xyz.com

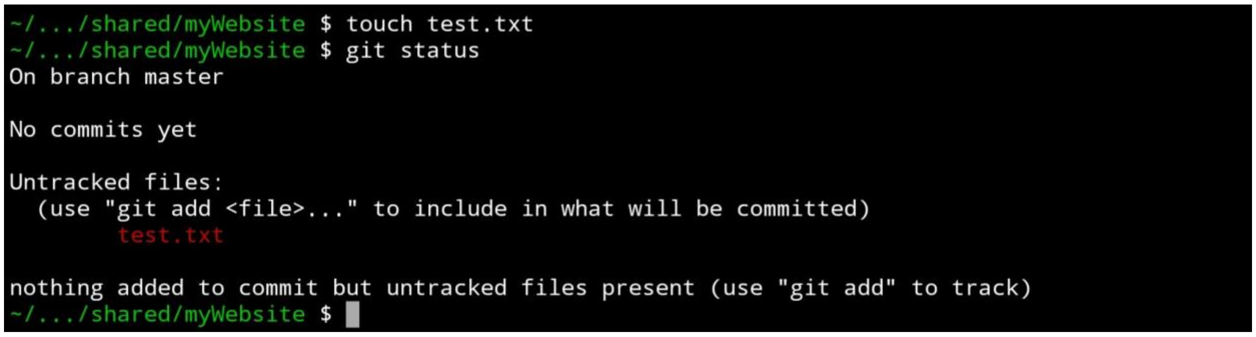
   now if you want to check, if your provided details are saved or not, type the following command
   git config user.name or user.email    //this will display the user name or email which you provided earlier
3) Initialising an empty repository
   Navigate to the **folder** or in other words **repository** which you want to initialise as a repository and then type
   git init    //after running the command it will look something like this



```
~/.../shared/myWebsite $ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /storage/emulated/0/myWebsite/.git/
```

   *here **myWebsite** is a folder created previously by the user which will be called as our repository
4) Creating files
   In order to create a new file run touch filename command and write some content in it   //touch is used to create a file
   e.g. touch test.txt
5) Tracking files
   Since we created a new file the git should know about it. So how do we check it?

git status //checks the status of all the files in the current repository



```
~/.../shared/myWebsite $ touch test.txt
~/.../shared/myWebsite $ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.txt

nothing added to commit but untracked files present (use "git add" to track)
~/.../shared/myWebsite $
```

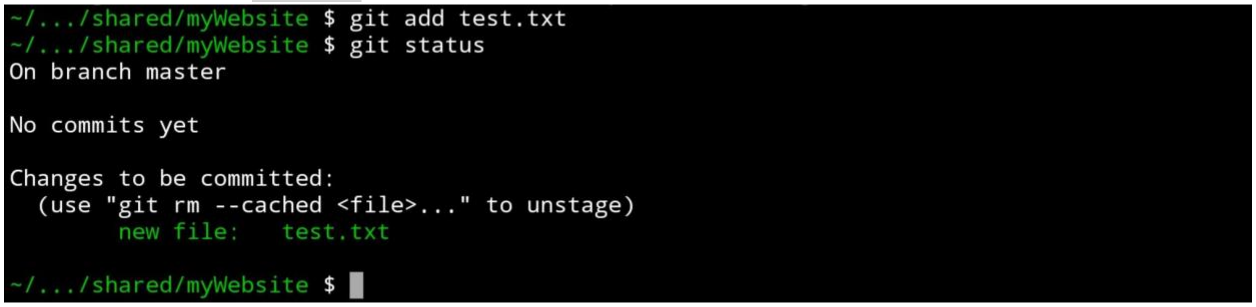
it is showing our newly created file under untracked files, it basically means files that have been created within our repo's working directory but have not yet been added to the repository's tracking index using the git add command.

-**Start tracking**

by running git add filename the git will add the mentioned file to its tracking index

e.g. git add test.txt

now run the git status command again and it will look something like this

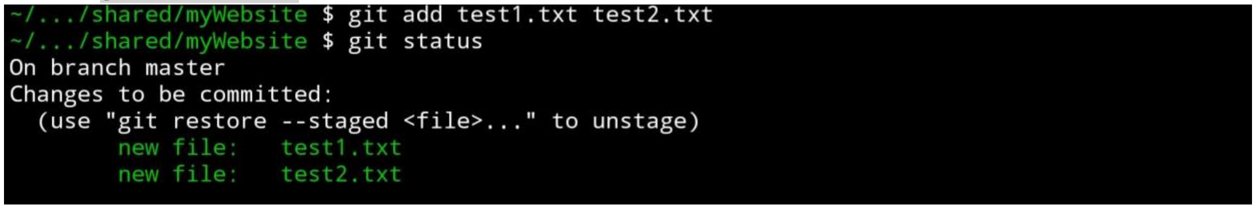


```
~/.../shared/myWebsite $ git add test.txt
~/.../shared/myWebsite $ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   test.txt

~/.../shared/myWebsite $
```

this is called the staging area which means that any changes that have been made to this particular file are ready to be committed in the next commit.

You can also add multiple files like this

git add file1 file2



```
~/.../shared/myWebsite $ git add test1.txt test2.txt
~/.../shared/myWebsite $ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   test1.txt
        new file:   test2.txt
```

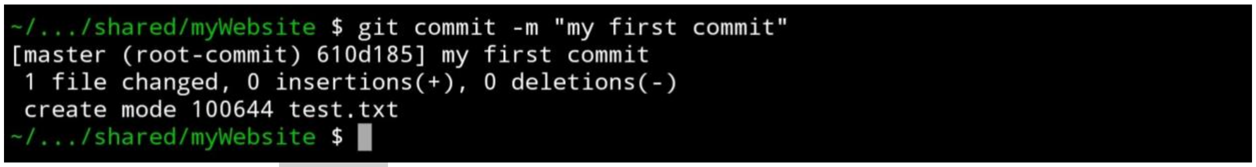
if you want to add all the files in one go run this command

git add . //this will add all the files in the staging area

6) Committing changes

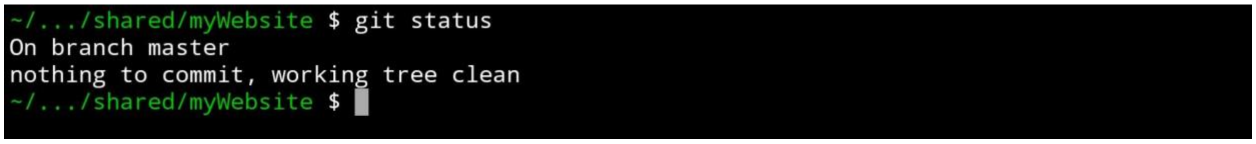
To commit/save a file changes simply run this command

git commit -m “your commit message” //-m means message and inside the quotation write your commit message



```
~/.../shared/myWebsite $ git commit -m "my first commit"
[master (root-commit) 610d185] my first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.txt
~/.../shared/myWebsite $
```

now if you run git status command again it will look something like this

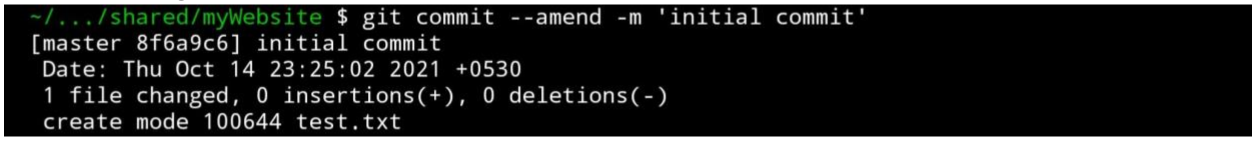


```
~/.../shared/myWebsite $ git status
On branch master
nothing to commit, working tree clean
~/.../shared/myWebsite $
```

because all the files have been committed.

If you want to change the commit message of your previous commit run

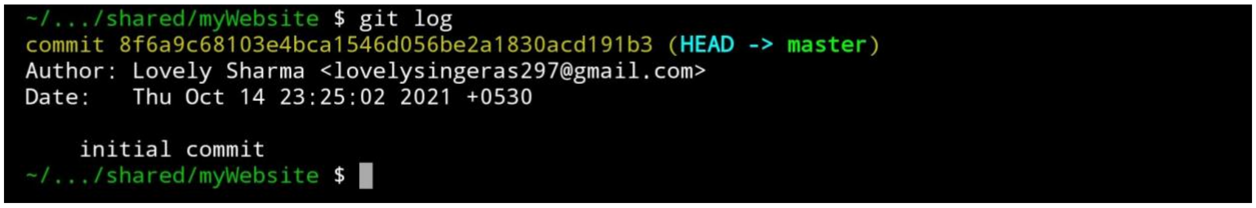
git commit –amend -m “new commit message” //this will update the previous commit message



```
~/.../shared/myWebsite $ git commit --amend -m 'initial commit'
[master 8f6a9c6] initial commit
 Date: Thu Oct 14 23:25:02 2021 +0530
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.txt
```

If you want to see all your previous commits just type

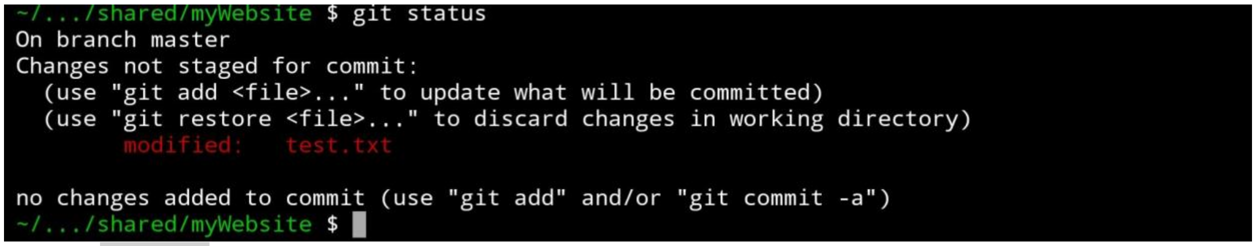
git log // this will display list of all the commits that have been made in the past



```
~/.../shared/myWebsite $ git log
commit 8f6a9c68103e4bca1546d056be2a1830acd191b3 (HEAD -> master)
Author: Lovely Sharma <lovelysingeras297@gmail.com>
Date:   Thu Oct 14 23:25:02 2021 +0530

    initial commit
~/.../shared/myWebsite $
```

7) Making changes in our repo

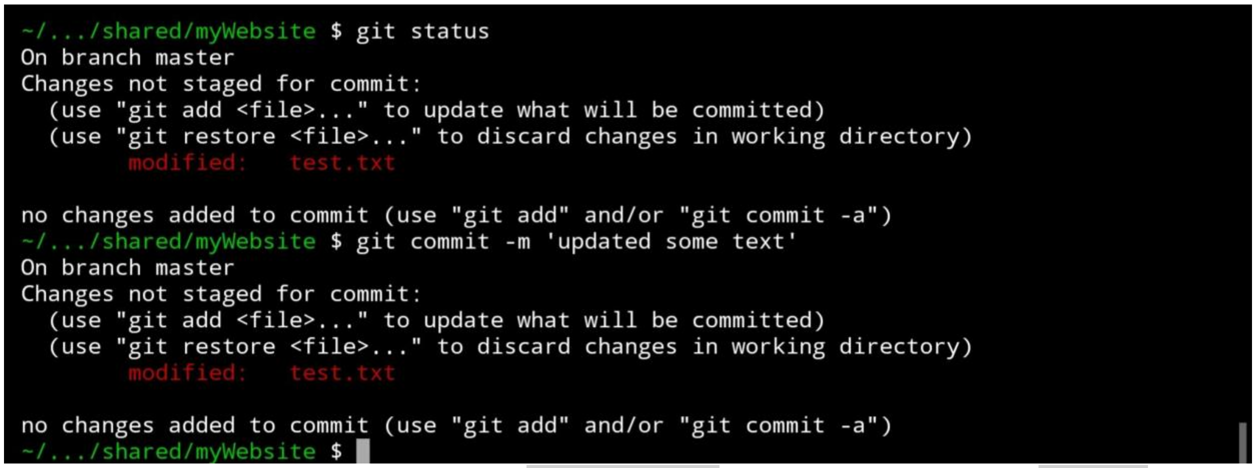
If we update a file the git will know that some changes have been made to a file you can use



```
~/.../shared/myWebsite $ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")
~/.../shared/myWebsite $
```

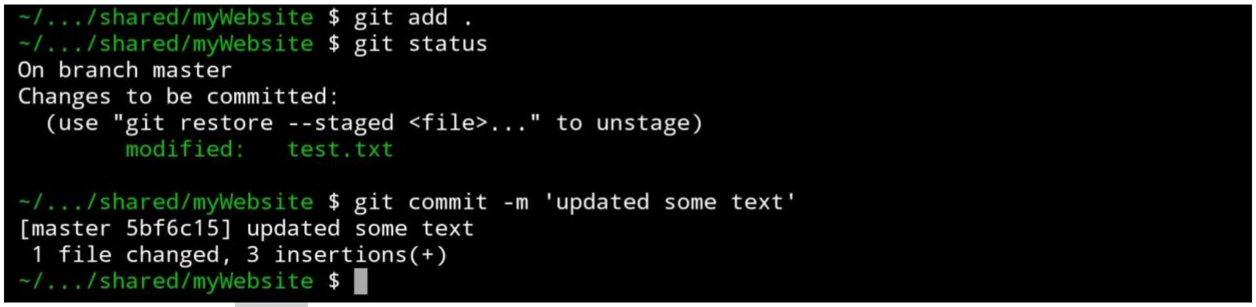
git status to see current status of all files here we have only one file in our repo so its showing status of one file only.

Now you can commit the files again but first you need to add file in the staging area then commit the file because if you directly try to commit there won't by any change.



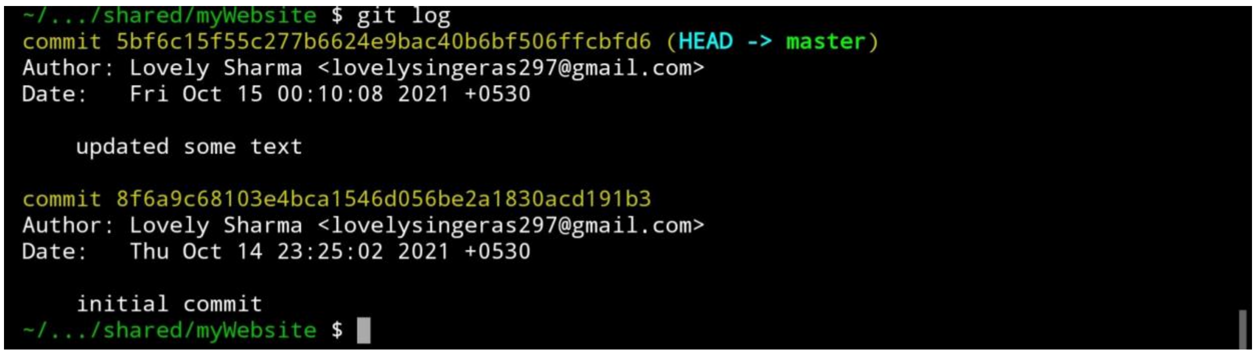
```
~/.../shared/myWebsite $ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")
~/.../shared/myWebsite $ git commit -m 'updated some text'
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")
~/.../shared/myWebsite $
```

So add the file to staging area  with git add filename and check the status with git status and finally commit your changes with git commit -m "new commit message"



```
~/.../shared/myWebsite $ git add .
~/.../shared/myWebsite $ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.txt

~/.../shared/myWebsite $ git commit -m 'updated some text'
[master 5bf6c15] updated some text
 1 file changed, 3 insertions(+)
~/.../shared/myWebsite $
```

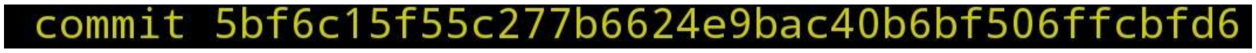
Now if you git log it will list all your previous commits.



```
~/.../shared/myWebsite $ git log
commit 5bf6c15f55c277b6624e9bac40b6bf506ffcbfd6 (HEAD -> master)
Author: Lovely Sharma <lovelysingeras297@gmail.com>
Date:   Fri Oct 15 00:10:08 2021 +0530

    updated some text

commit 8f6a9c68103e4bca1546d056be2a1830acd191b3
Author: Lovely Sharma <lovelysingeras297@gmail.com>
Date:   Thu Oct 14 23:25:02 2021 +0530

    initial commit
~/.../shared/myWebsite $
```

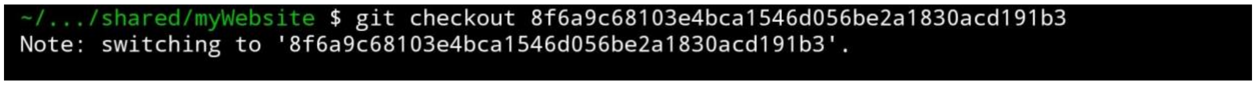
8) Travelling back in time
Consider a scenario in which you made some changes in a file which you were not supposed to and now you want to go back in time with those changes. In this case git helps us to do so.

In order to go at a certain instance just copy the commit hashcode/id which looks something like this



```
commit 5bf6c15f55c277b6624e9bac40b6bf506ffcbfd6
```

and run the following command
git checkout hashcode



```
~/.../shared/myWebsite $ git checkout 8f6a9c68103e4bca1546d056be2a1830acd191b3
Note: switching to '8f6a9c68103e4bca1546d056be2a1830acd191b3'.
```

this will  take us in the previous state of our file, if you could see this is the hashcode of our 1st commit when we only committed the file without any insertion.

The result of the above command will erase all the insertions made by us.
Similarly you can move between states using checkout command.
*try doing it on your own it will help you understand it better.