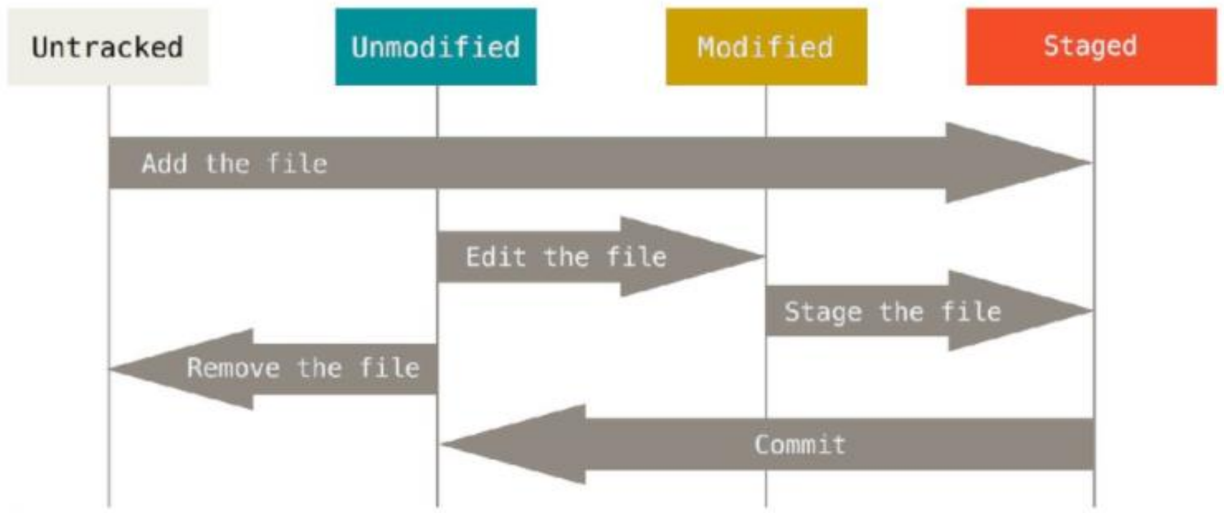
Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.
Below you can see the Lifecycle of git and how it works.

Lifecycle



Status

```
git status
```

This command is used to find out in which states you repository files are.

Now we will move on to practical

1. Install Git:

first, we download and install the git link is below
https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

2. Set username and email

After successfully installing the git we open the git bash terminal and set the username and email

```
$ git config –global user.name Amir
$ git config –global user.email amirali.cheema@outlook.com
```

3. Git Cloning

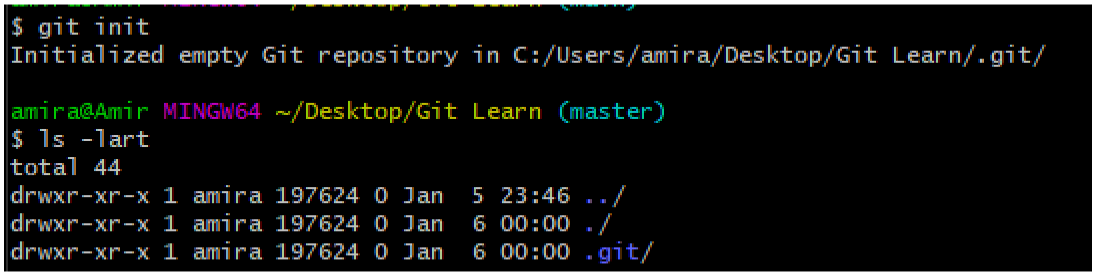
Through cloning, we get all code from GitHub to here in local
For example, I clone some code below is an example of how to get

```
$ git clone address here folder name
```

4. Git Initialize

If are doing from start in local computer, then first will initialize the folder by using the command

```
$ git init
$ ls -lart    # show all the hidden files
```



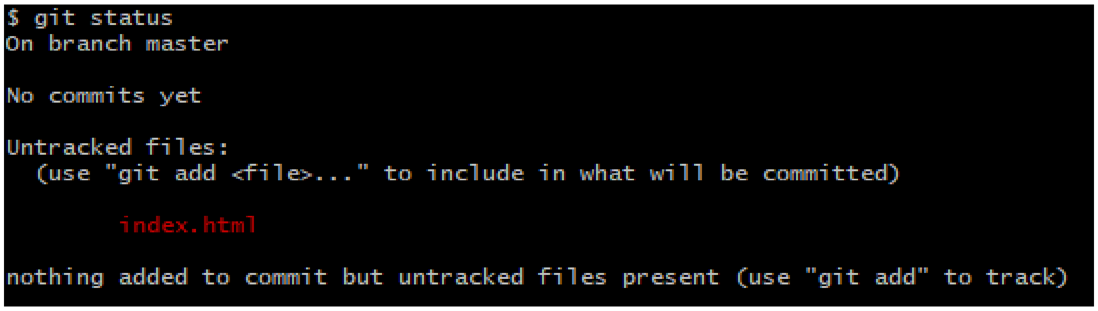
```
$ git init
Initialized empty Git repository in C:/Users/amira/Desktop/Git Learn/.git/

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ ls -lart
total 44
drwxr-xr-x 1 amira 197624 0 Jan  5 23:46 ../
drwxr-xr-x 1 amira 197624 0 Jan  6 00:00 ./
drwxr-xr-x 1 amira 197624 0 Jan  6 00:00 .git/
```

after creating some files and changes in code (In my case it's index.html but didn't add it)

5. Git Status

Status basically the current status after the operation for example I create a file index.html in my folder and check the status and he gave the info that file is untracked

```
git status    # tell the file is not committed yet
```



```
$ git status
On branch master

No commits yet

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

        index.html

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

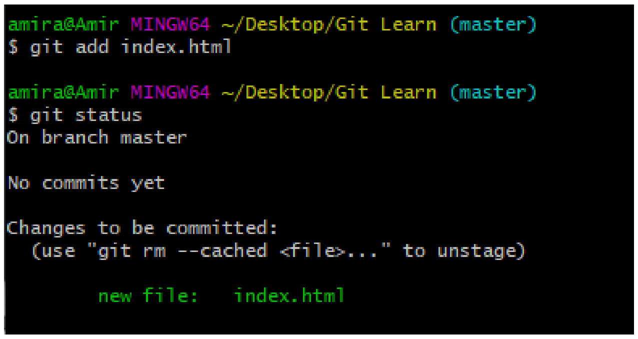
6. Git Add

After that, I add the index.html file by using the command

$ git add index.html # single file add

If I want to add multiple files at the same time, then I will use the following command

 $ git add -A   # multiple files add



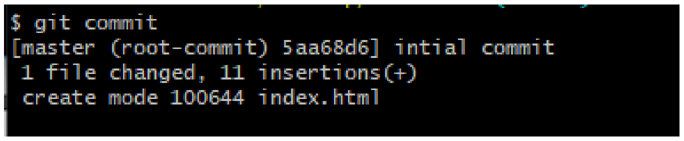
```
amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git add index.html

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git status
On branch master

No commits yet

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

        new file:   index.html
```

7. Git Commit

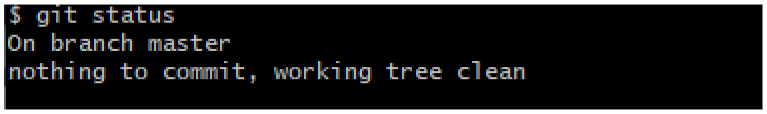
Through commit we can push file from stagged to unmodified

$ git commit

After that it will open vim editor to type press <i> then type initial commit this is the message that I gave to after the change in the file) then from the exit from vim editor by pressing ESC:wq



```
$ git commit
[master (root-commit) 5aa68d6] intial commit
 1 file changed, 11 insertions(+)
 create mode 100644 index.html
```



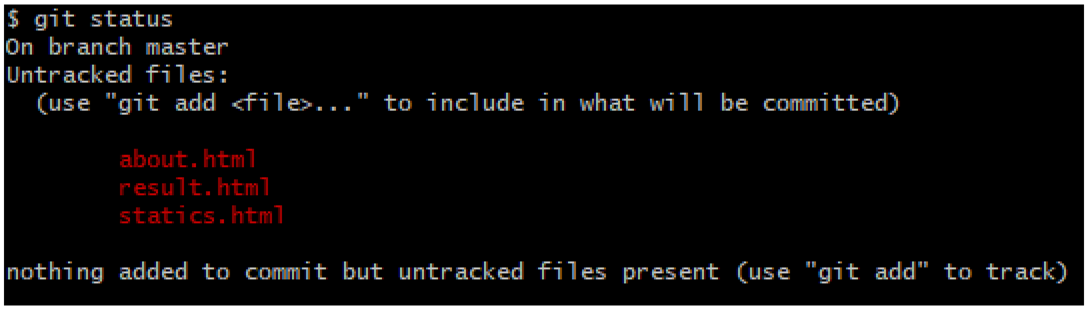
```
$ git status
On branch master
nothing to commit, working tree clean
```

Or I can the same thing with the different command which is easy

 git commit -m "initial commit"

Now I create some file

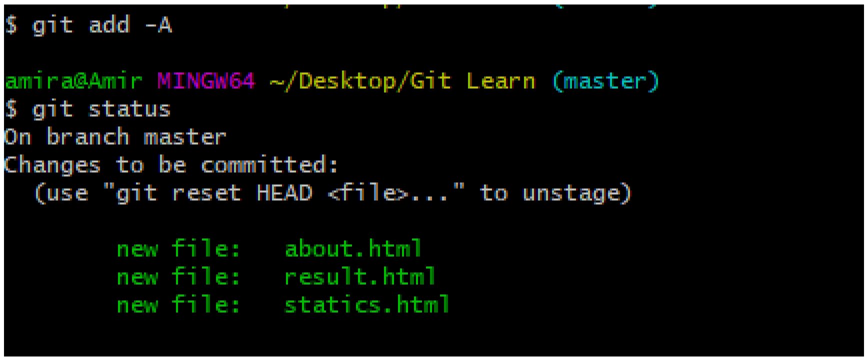
$ git status



```
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        about.html
        result.html
        statics.html

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

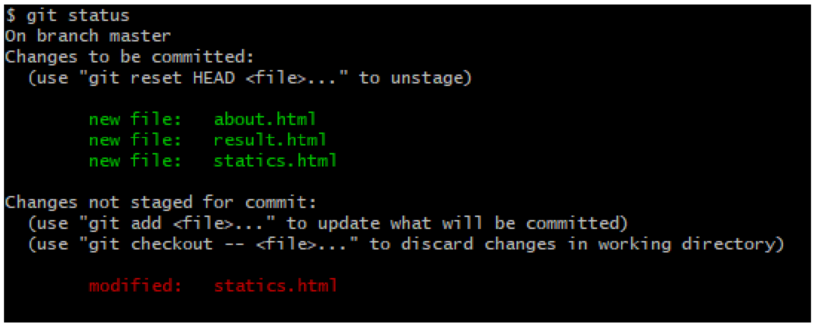
Now I will add all files (that I create) that i using the command

git add -A



```
$ git add -A

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   about.html
        new file:   result.html
        new file:   statics.html
```

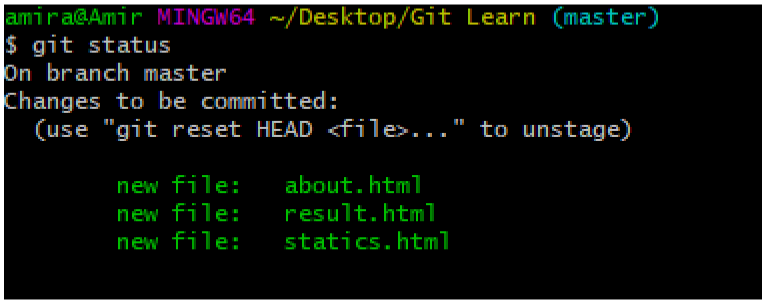
Now the file are ready to commit. Now I have some changes in statics.html And see the status



```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   about.html
        new file:   result.html
        new file:   statics.html

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   statics.html
```

Now I add the statics file again

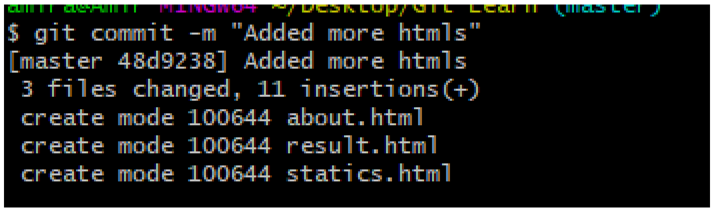
$ git add statics.html



```
amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   about.html
        new file:   result.html
        new file:   statics.html
```

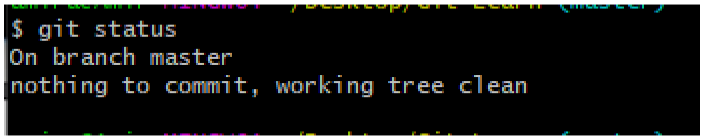
Now I commit all the files

$ git commit -m "Added more HTML"   # shortcut way to commit instead of using vim editor



```
$ git commit -m "Added more htmls"
[master 48d9238] Added more htmls
 3 files changed, 11 insertions(+)
 create mode 100644 about.html
 create mode 100644 result.html
 create mode 100644 statics.html
```

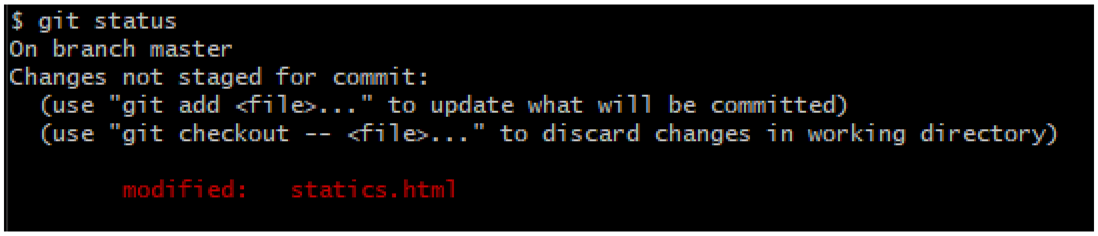
Now check the file status



```
$ git status
On branch master
nothing to commit, working tree clean
```

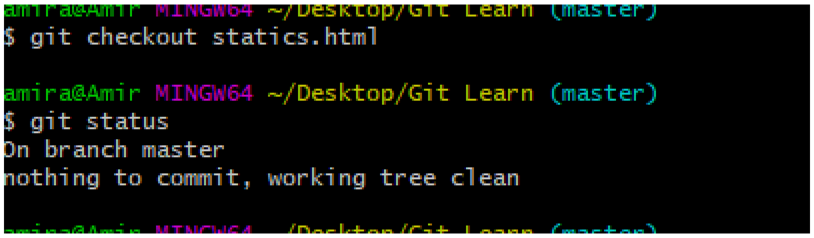
8. Git Checkout

For example, someone change code in file which was wrong. So, to resolve this problem we use
Like in my case file is a change like statistics.html which was the wrong code.



```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   statics.html
```

And I want the original code which I commit last time.
$ git checkout statics.html   # single file checkout

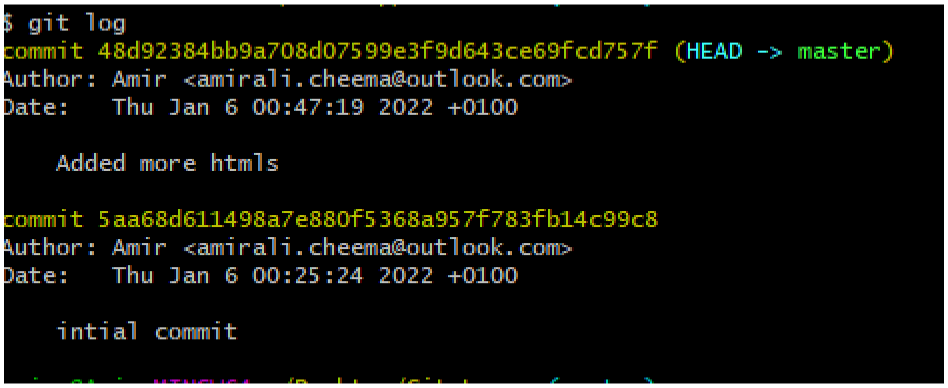


```
amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git checkout statics.html

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git status
On branch master
nothing to commit, working tree clean
```

If all file changes with the wrong code, then return them back in the original form
$ git checkout -f   # multiple file checkouts
Now I want all the detail of what commit who commit etc

9. Git log

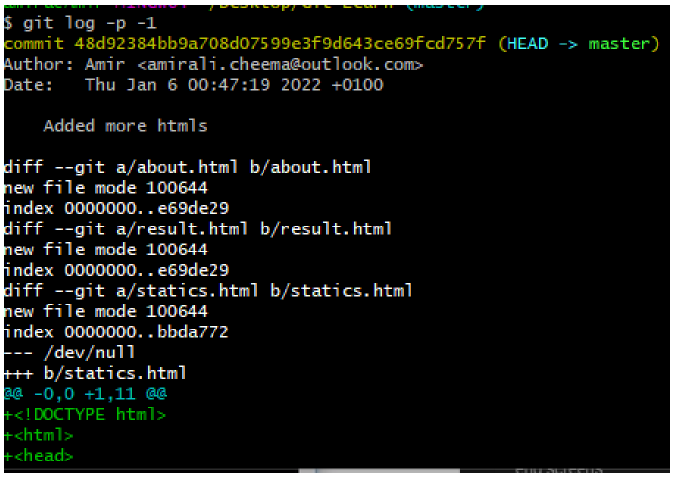
I want log is basically gave the summary of your commit that you did
$ git log



```
$ git log
commit 48d92384bb9a708d07599e3f9d643ce69fcd757f (HEAD -> master)
Author: Amir <amirali.cheema@outlook.com>
Date:   Thu Jan 6 00:47:19 2022 +0100

    Added more htmls

commit 5aa68d611498a7e880f5368a957f783fb14c99c8
Author: Amir <amirali.cheema@outlook.com>
Date:   Thu Jan 6 00:25:24 2022 +0100

    intial commit
```

you can also filter you commit for example I want to see only 1 commit log summary then I will use the following command

$ git log -p -1

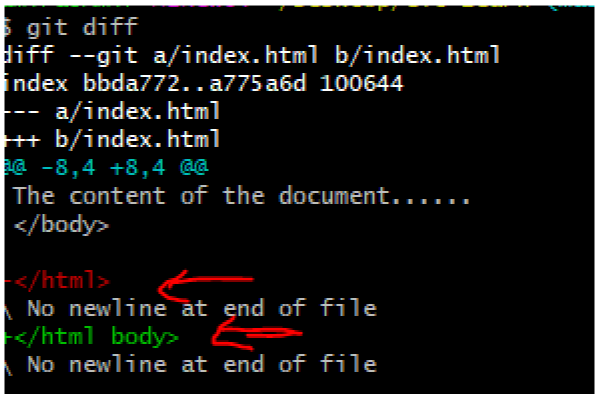


```
$ git log -p -1
commit 48d92384bb9a708d07599e3f9d643ce69fcd757f (HEAD -> master)
Author: Amir <amirali.cheema@outlook.com>
Date:   Thu Jan 6 00:47:19 2022 +0100

    Added more htmls

diff --git a/about.html b/about.html
new file mode 100644
index 0000000..e69de29
diff --git a/result.html b/result.html
new file mode 100644
index 0000000..e69de29
diff --git a/statics.html b/statics.html
new file mode 100644
index 0000000..bbda772
--- /dev/null
+++ b/statics.html
@@ -0,0 +1,11 @@
+<!DOCTYPE html>
+<html>
+<head>
```

10. Git diff

Git diff basically tell the difference between your last and current commit. In my case, I have some changes in the file of the index.html

$ git diff   # it compares the new changes with the last one

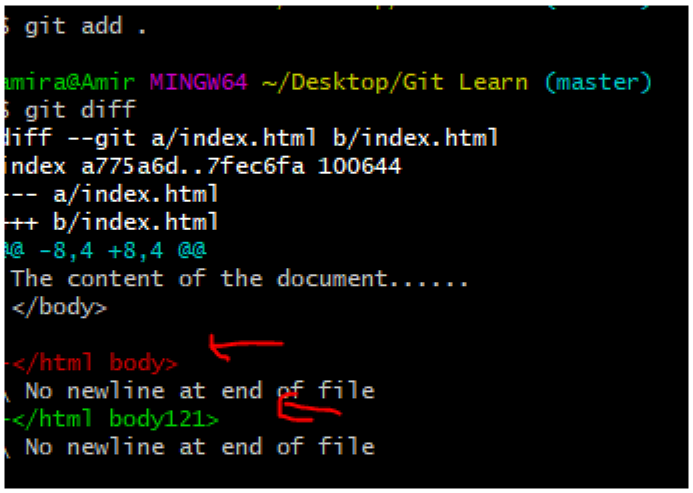


```
$ git diff
diff --git a/index.html b/index.html
index bbda772..a775a6d 100644
--- a/index.html
+++ b/index.html
@@ -8,4 +8,4 @@
 The content of the document......
 </body>

-</html>
\ No newline at end of file
+</html body>
\ No newline at end of file
```

$ git add - A

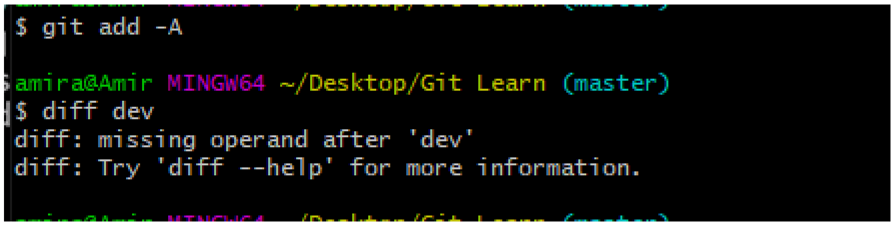
After adding, I again change

$ git diff

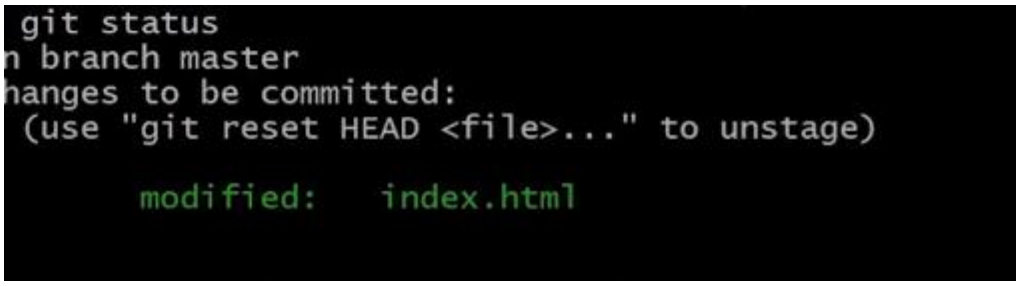


```
$ git add .

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git diff
diff --git a/index.html b/index.html
index a775a6d..7fec6fa 100644
--- a/index.html
+++ b/index.html
@@ -8,4 +8,4 @@
 The content of the document......
 </body>

-</html body>
\ No newline at end of file
+</html body121>
\ No newline at end of file
```

After adding changes files I check again



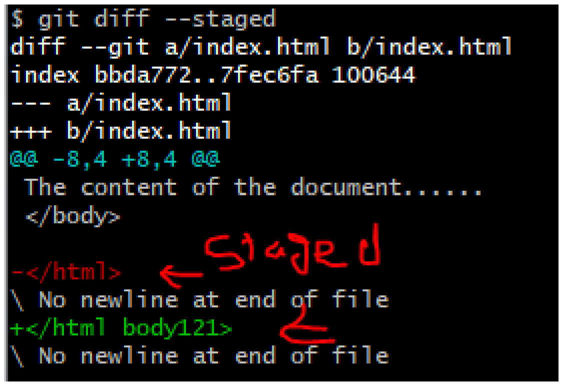
```
$ git add -A

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ diff dev
diff: missing operand after 'dev'
diff: Try 'diff --help' for more information.
```



```
 git status
n branch master
hanges to be committed:
 (use "git reset HEAD <file>..." to unstage)

        modified:   index.html
```

Now I want same code back (which was in stagged area) then I will use following command

$ git diff – staged

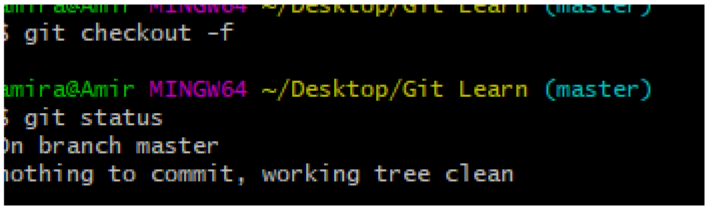


```
$ git diff --staged
diff --git a/index.html b/index.html
index bbda772..7fec6fa 100644
--- a/index.html
+++ b/index.html
@@ -8,4 +8,4 @@
 The content of the document......
 </body>

-</html>
\ No newline at end of file
+</html body121>
\ No newline at end of file
```

Now back to same code

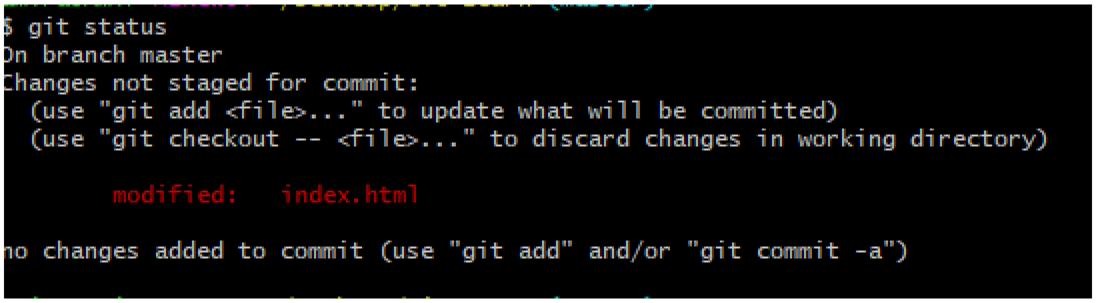
$ git checkout -f

$ git status



```
$ git checkout -f

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git status
On branch master
nothing to commit, working tree clean
```

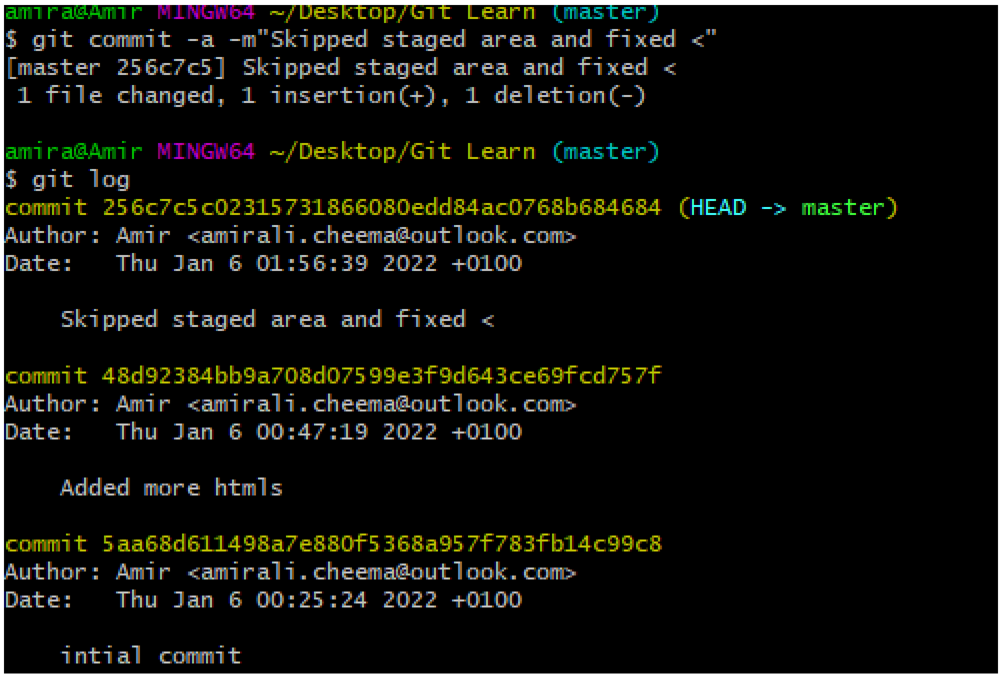
After that, I change some in an index.html file



```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
```

Now I use a shortcut way to add and commit in a single commit

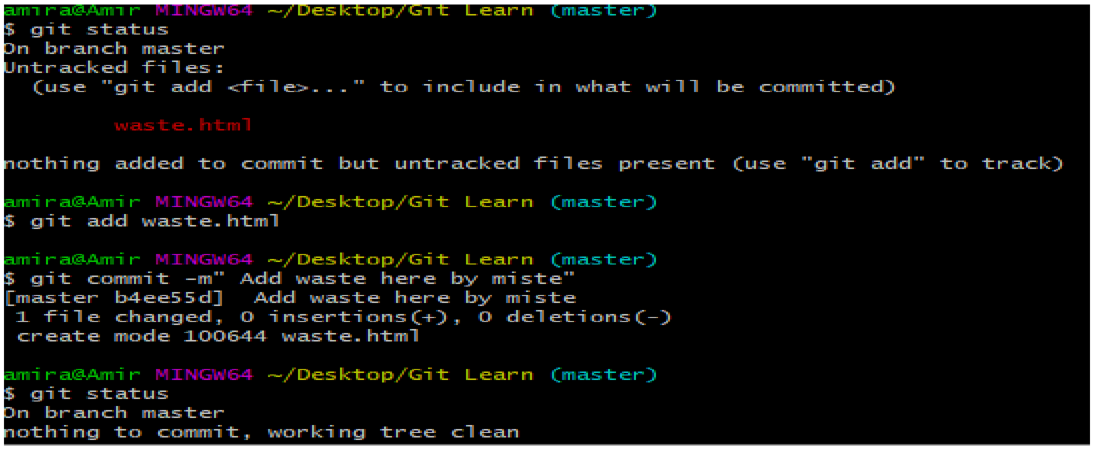
$ git commit -a -m ”Skipped staged area and fixed <”



```
amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git commit -a -m"Skipped staged area and fixed <"
[master 256c7c5] Skipped staged area and fixed <
 1 file changed, 1 insertion(+), 1 deletion(-)

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git log
commit 256c7c5c02315731866080edd84ac0768b684684 (HEAD -> master)
Author: Amir <amirali.cheema@outlook.com>
Date:   Thu Jan 6 01:56:39 2022 +0100

    Skipped staged area and fixed <

commit 48d92384bb9a708d07599e3f9d643ce69fcd757f
Author: Amir <amirali.cheema@outlook.com>
Date:   Thu Jan 6 00:47:19 2022 +0100

    Added more htmls

commit 5aa68d611498a7e880f5368a957f783fb14c99c8
Author: Amir <amirali.cheema@outlook.com>
Date:   Thu Jan 6 00:25:24 2022 +0100

    intial commit
```

11. Remove File

For example, I add and commit wrong then I remove in my case my file is waste.html

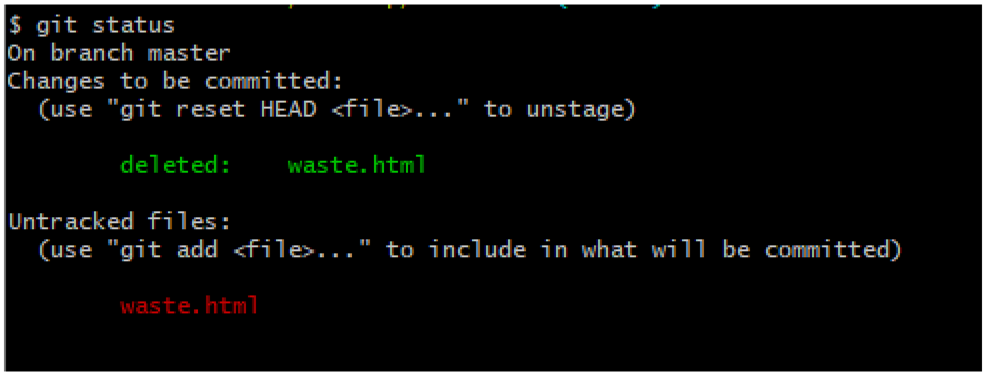


```
amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        waste.html

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

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git add waste.html

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git commit -m" Add waste here by miste"
[master b4ee55d]  Add waste here by miste
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 waste.html

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git status
On branch master
nothing to commit, working tree clean
```

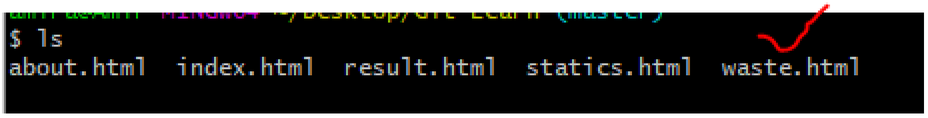
Here two ways to remove 1st way is to remove only from a stagged area, not a hard drive 2nd remove completely

1st way

$ git rm –cached waste.html   # remove from stagged only



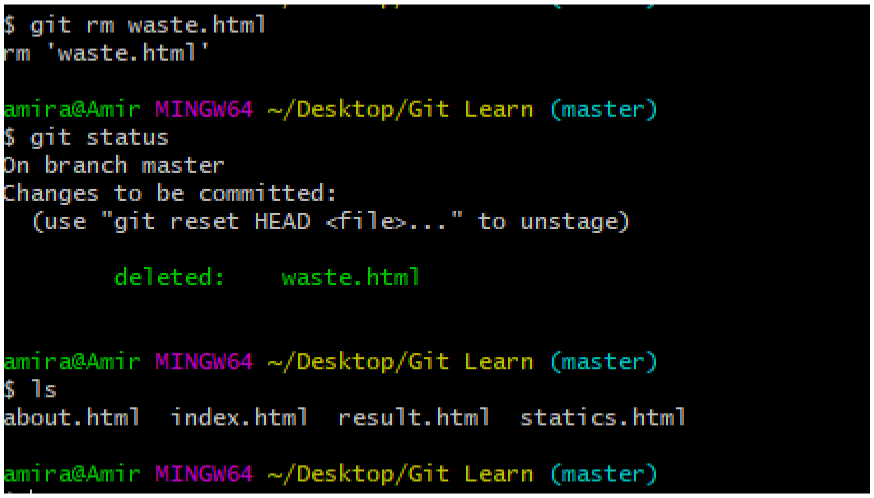
```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    waste.html

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

        waste.html
```



```
$ ls
about.html  index.html  result.html  statics.html  waste.html
```

2nd way

git rm waste.html   # remove from both stagged area and local

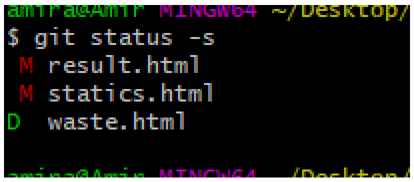


```
$ git rm waste.html
rm 'waste.html'

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    waste.html


amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ ls
about.html  index.html  result.html  statics.html

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
```

Another way to check the status

git status -s # provide a short summary of opr

For example, I change some code in 2 file

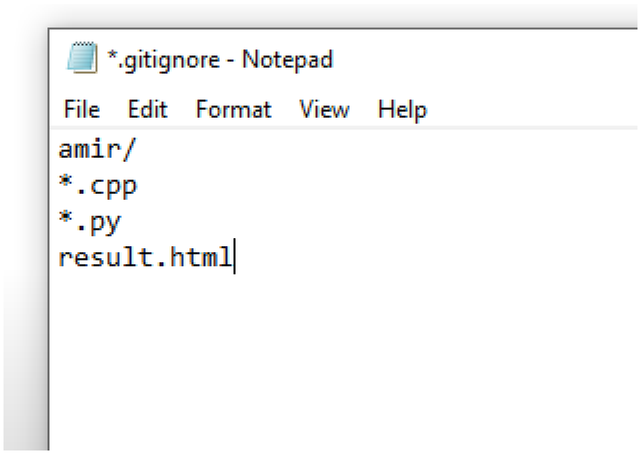


```
amira@Amir MINGW64 ~/Desktop/
$ git status -s
 M result.html
 M statics.html
D  waste.html
```

12. Ignore File

First, we make create a file with the name .gitignore And inside whatever I gave the name of a file with the extension it will ignore

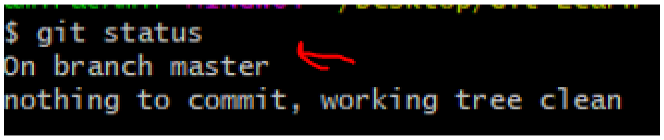
For example, I want to ignore folder amir then inside I write amir/ and he ignores all files inside a file Or I want to ignore a specific extension file like *.cpp or *.py he ignores these files as well just write inside the .gitignore file



```
amir/
*.cpp
*.py
result.html
```

12. Branches

git branches is a killer tool in git. Our main branch is master we make another branch and work on it and will not affect the master branch as well and if we merge the command then we merge our new branch with master

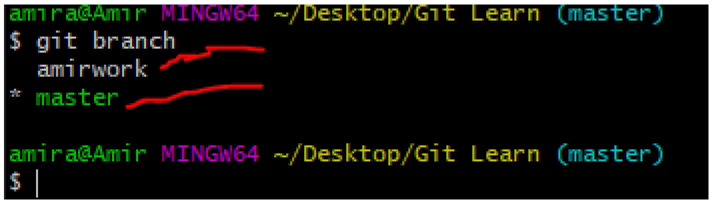


```
$ git status
On branch master
nothing to commit, working tree clean
```

Currently, we are in branch master so will create another branch with name amirwork

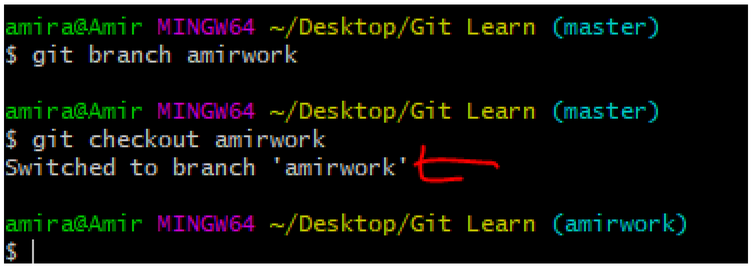
$ git branch amirwork # create an amirwork branch

$ git branch # check the branches



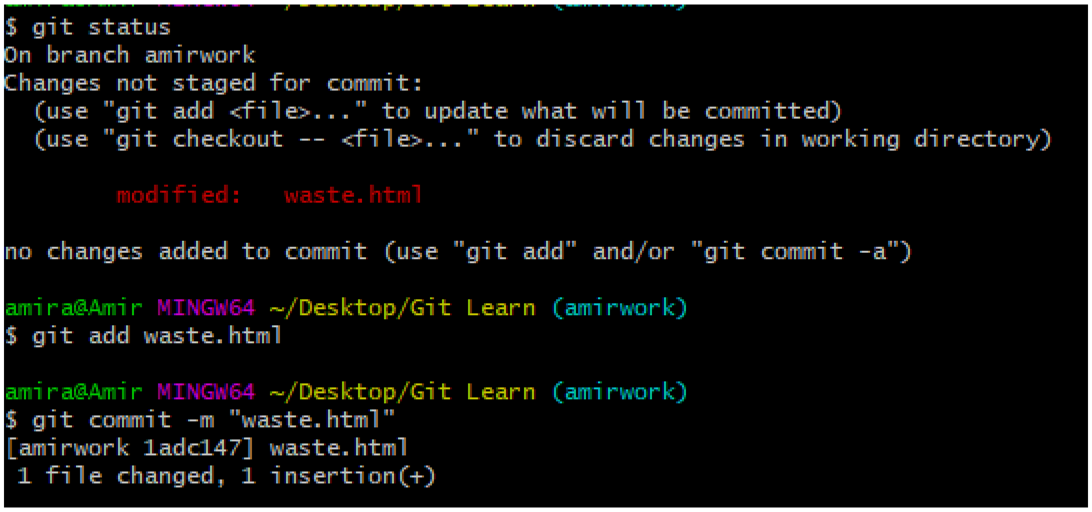
```
amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git branch
  amirwork
* master

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$
```

git checkout amirwork   # it will move from master to amirwork branch



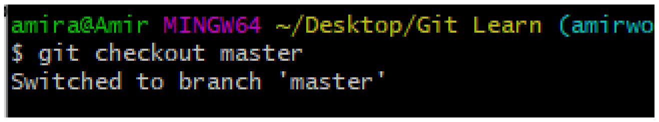
```
amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git branch amirwork

amira@Amir MINGW64 ~/Desktop/Git Learn (master)
$ git checkout amirwork
Switched to branch 'amirwork'

amira@Amir MINGW64 ~/Desktop/Git Learn (amirwork)
$ |
```

Here I work some on waste files through the amirwork branch then go back on a master



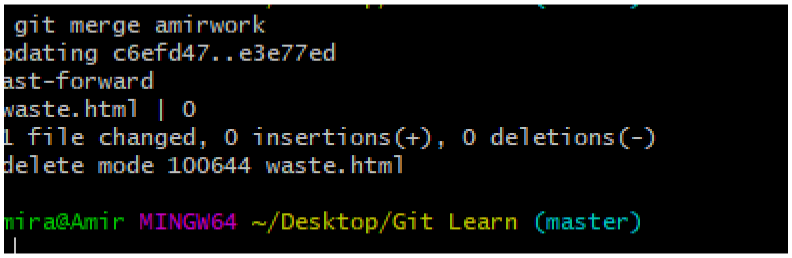
```
$ git status
On branch amirwork
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   waste.html

no changes added to commit (use "git add" and/or "git commit -a")

amira@Amir MINGW64 ~/Desktop/Git Learn (amirwork)
$ git add waste.html

amira@Amir MINGW64 ~/Desktop/Git Learn (amirwork)
$ git commit -m "waste.html"
[amirwork 1adc147] waste.html
 1 file changed, 1 insertion(+)
```

After finishing work on waste.html I checkout to master



```
amira@Amir MINGW64 ~/Desktop/Git Learn (amirwo
$ git checkout master
Switched to branch 'master'
```

For example, my waste.html work is confirmed then I merge amirwork with the master branch

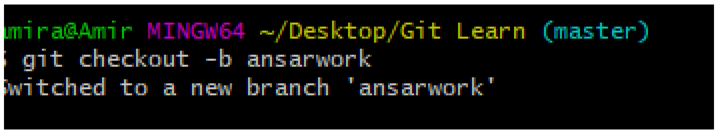


```
 git merge amirwork
pdating c6efd47..e3e77ed
ast-forward
waste.html | 0
1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 waste.html

mira@Amir MINGW64 ~/Desktop/Git Learn (master)
|
```

So, in this way branches work. It's a convenient method to do that if work is okay then you can merge in master.

We can also create multiple branches as well

$ git checkout -b ansarwork   # here I did two things first creates a branch and 2nd switch this branch as well



```
mira@Amir MINGW64 ~/Desktop/Git Learn (master)
 git checkout -b ansarwork
witched to a new branch 'ansarwork'
```

And that's all for git.

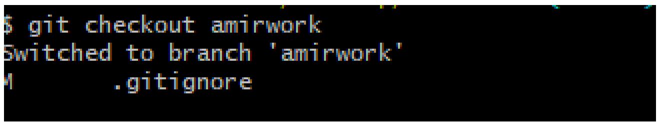
13. Github

Github is basically a web hosting service we push our code from local to the server.
In git we did everything so, it's time to push our code. First, I create a repository on GitHub then using the below command I push my commands.
$ git remote add origin https://github.com/AmirAli5/Git_Complete.git

If you want to change the address type $ git remote set-url then address below is example
$ git remote set-url origin git@github.com:AmirAli5/Git_Complete.git

Now I will push using this command
$ git push -u origin master

Here I push using the origin master branch which is the default. Now I change the branch and push the code same repo



```
$ git checkout amirwork
Switched to branch 'amirwork'
M       .gitignore
```

$ git push -u origin amirwork

Here I push code from the second branch. We can push multiple branches. We can see the code of each branch code. And commit as well. That's all.

Happy Learning😉