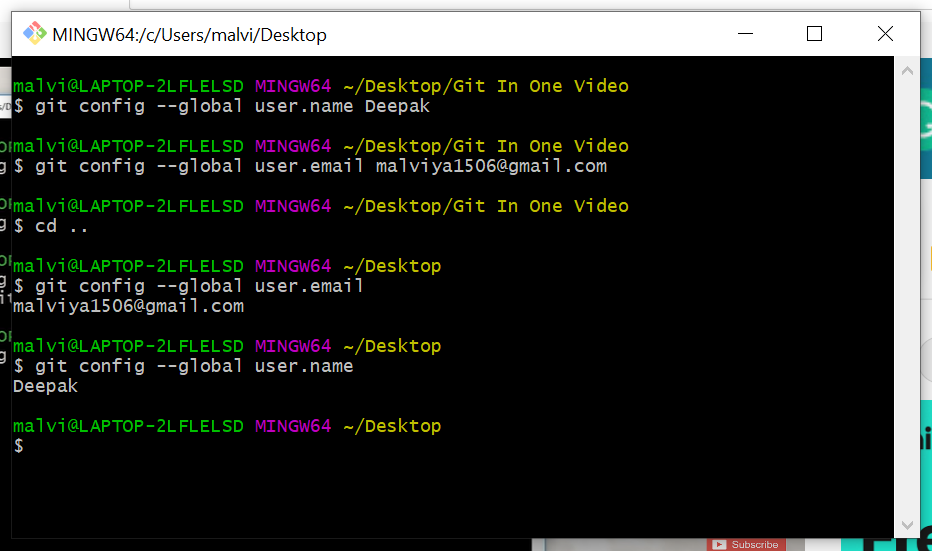
**Git and Github**

To set the configurations of the person working:

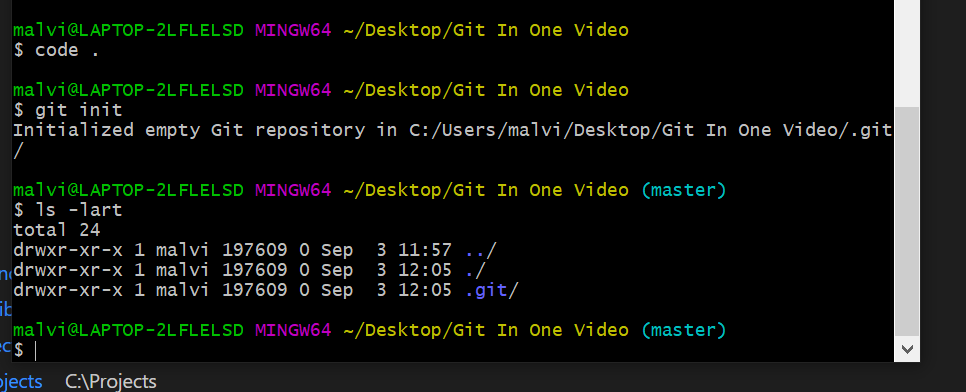
****

Open Visual Studio code by given command and initialize the git.

Check the folders and files present in the folder by the command

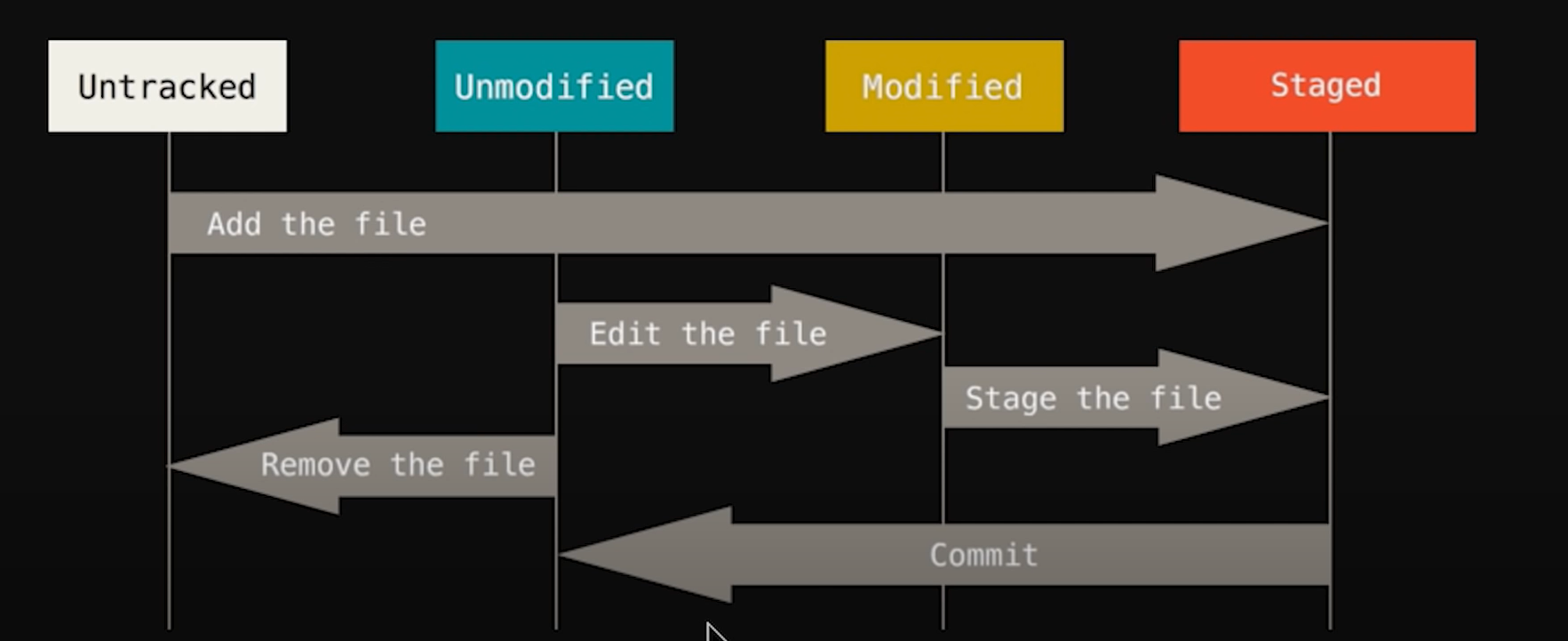
ls -lart

It will create a folder by name “git” where all the files will be available regarding git.

****

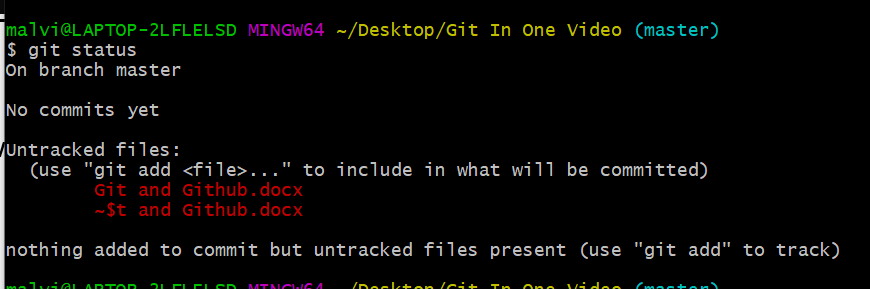
Here are the different status of Git files:-

1. **Untracked** – When Git not at all care about your file. You need to add this file to transfer it to Staged status.
2. **Staged** – When your file is in Git now, and ready to commit.
3. **Unmodified** – When your file is updated and absolutely fine. From here, we can either edit or remove the file.
4. **Modified** – When your file has been modified and want to get Staged and Commit to save the changes. From here, we should Staged and Commit the file. We can pass the file to Staged, but avoid commit if we want some time before commit.

****

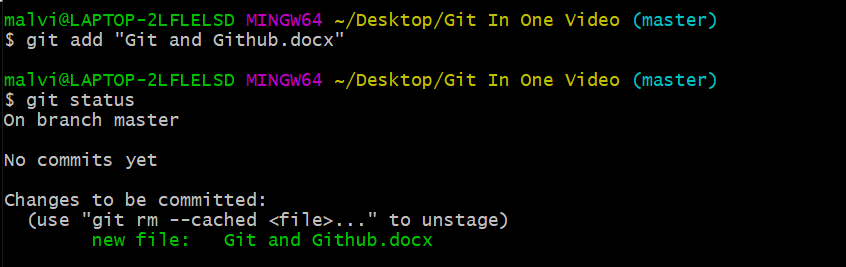
Just now, I have created a new file in a folder,

It is showing the status as “Untracked files”

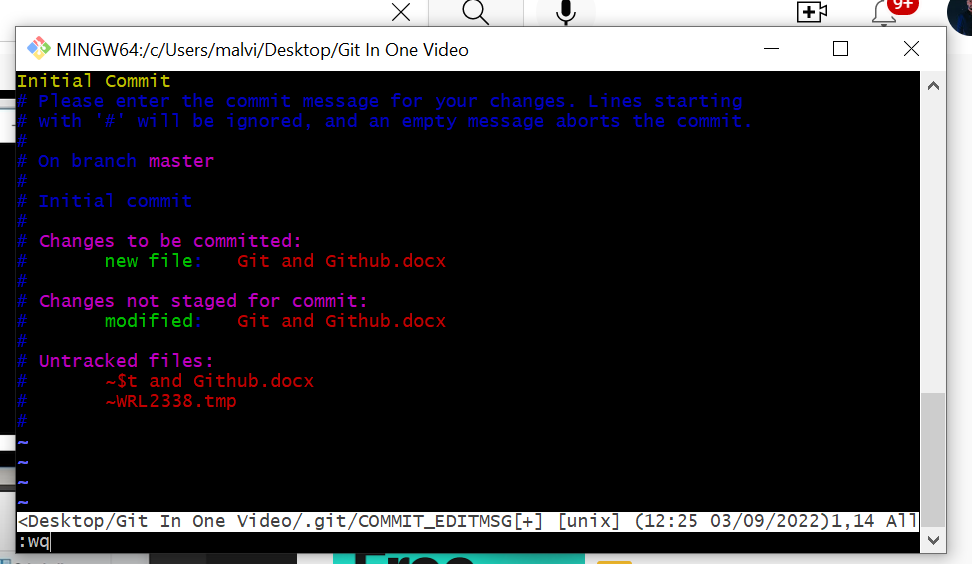


To Add it to Stage status, use below command,

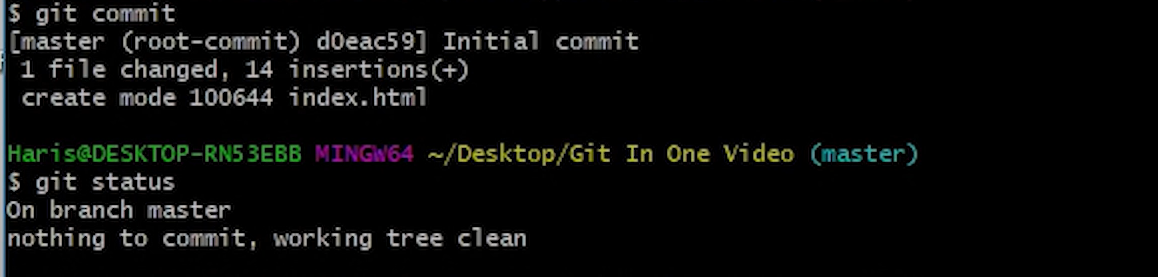
Now, the Status is showing as “Changes to be committed”

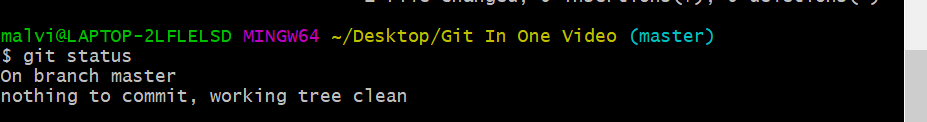


Always we should have an initial commit to save the structure. Click “i” to insert, and write “Initial Commit”



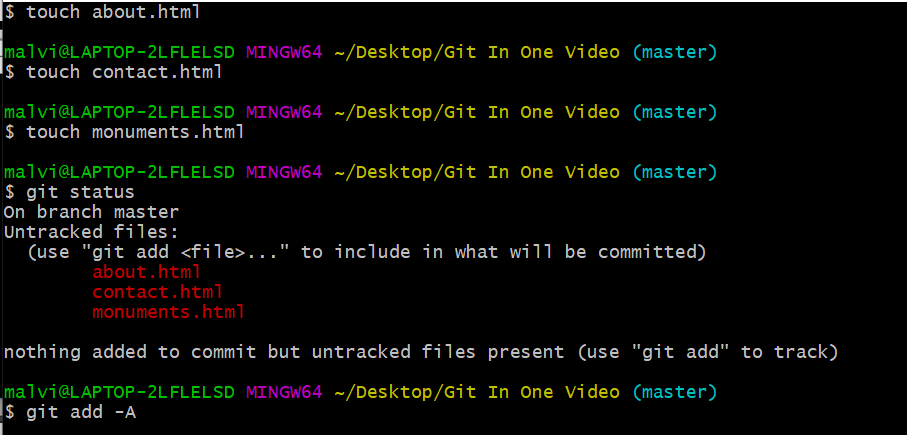
Esc+ “:wq” to exit





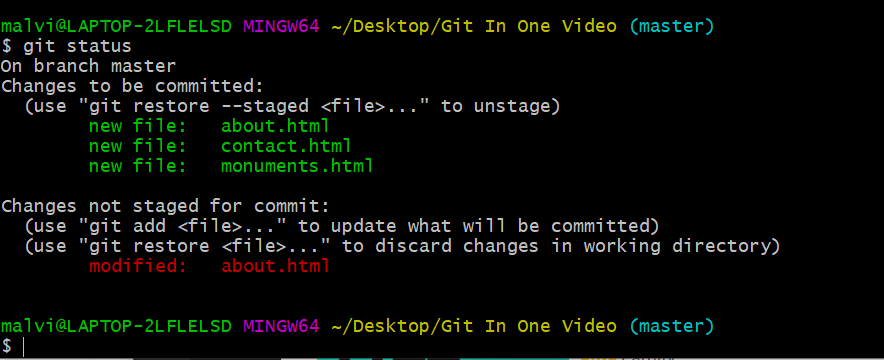
Working tree clean, ie. Everything is absolutely fine till now. Everything is ready.

To commit multiple files together,

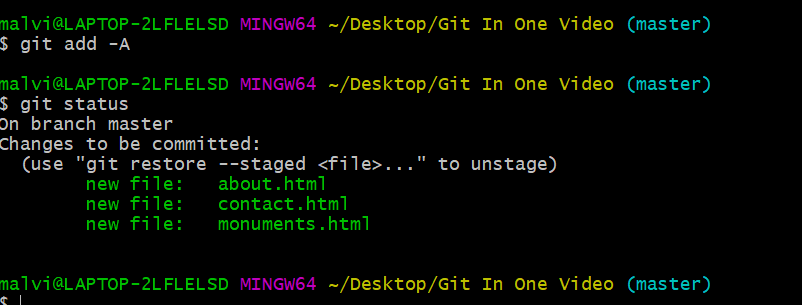
****

We Changed the file about.html after this,

And ran status command,



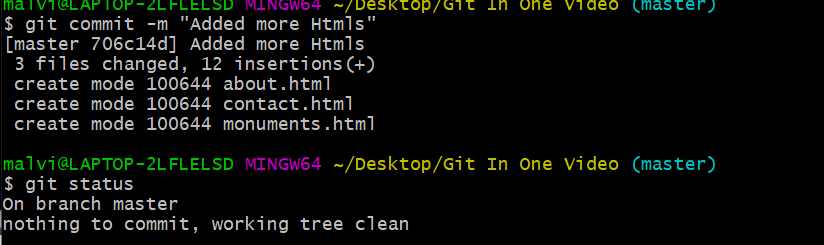
Now, we can see, about.html is present in both “Staging” area and “Modified” area, but we should note one thing here is that both these files are having different version, the one in Staging is not the updated one.



Now again all files are added.

Remember, previously when we have performed a commit, the process was very long, (Insert+ Text+Esc + “:wq”)

To avoid this, we give below command to commit,



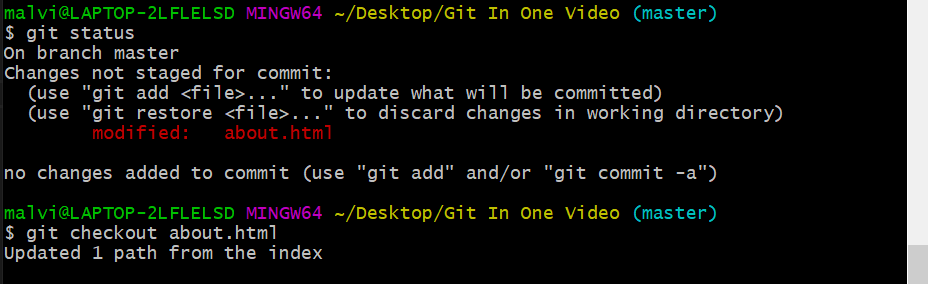
Here, “m” stands for message and we can give a message with the commit.

Now suppose, someone came and remove your whole code and wrote some vulgar comments on page and saved it.

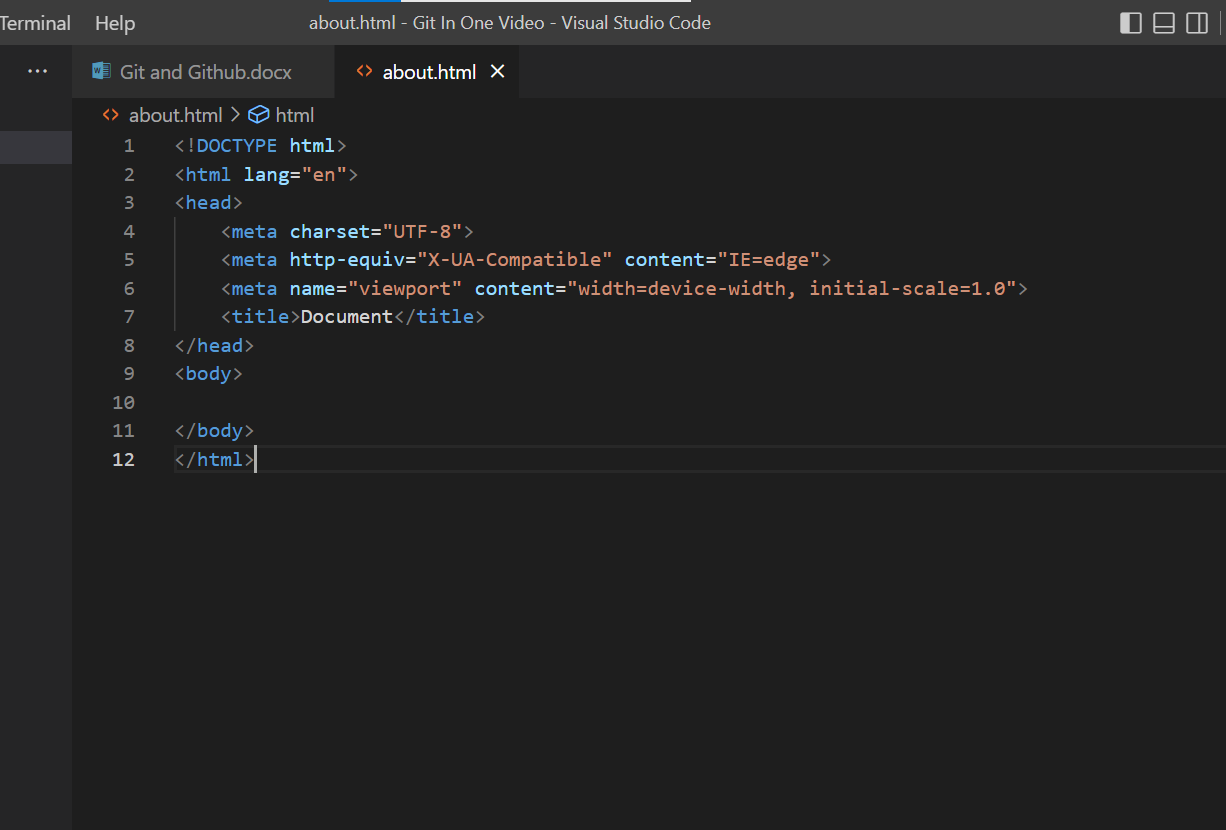
Now, what will you do? Your code has been gone?

* No, you can actually get it back using git. Lets see how,

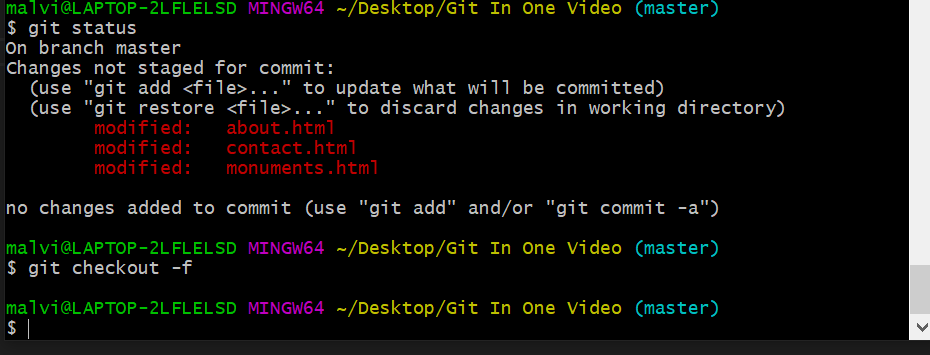
We can use checkout command to get back the last committed code,



You will be able to see your code back in file.



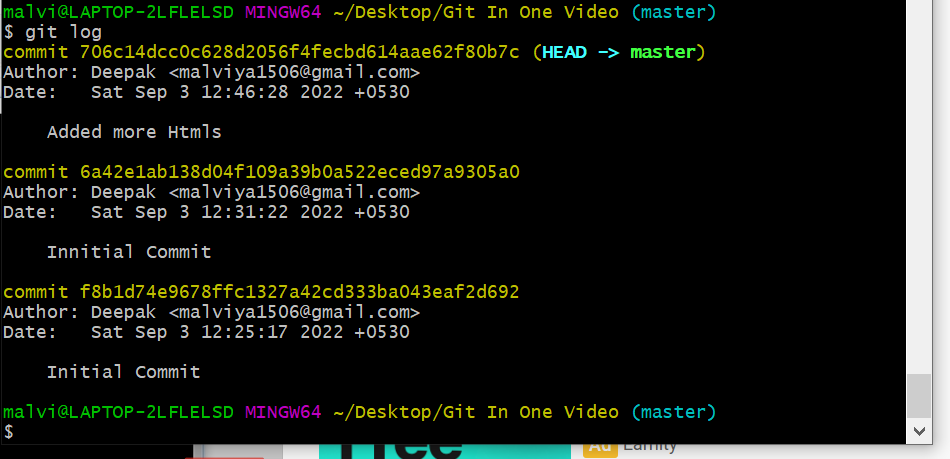
In case of the multiple files changed by someone else, and we want to get our file back,



It will match all the files to last commit.

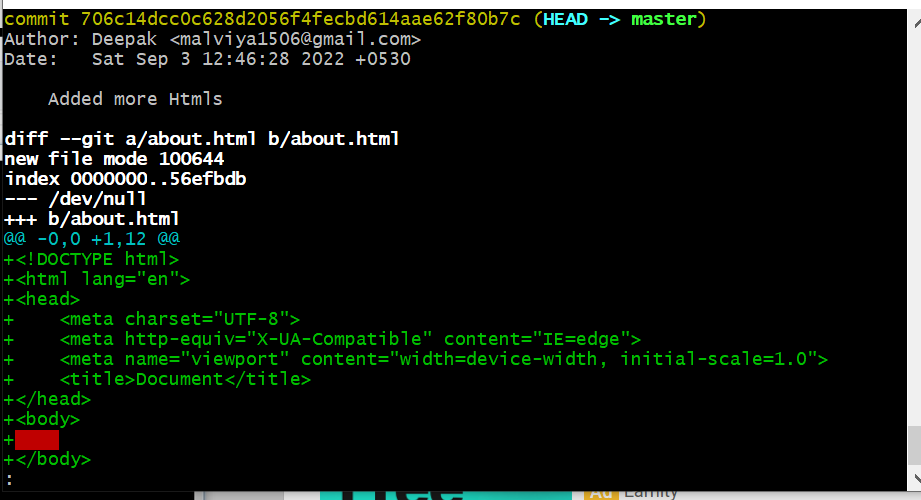
**Git log command:**

This is used to get all the commit informations.



Now suppose, there are 1000+ commits, and we want to filter the commit.

Use command- git log -p -1 (no. of commits you want)

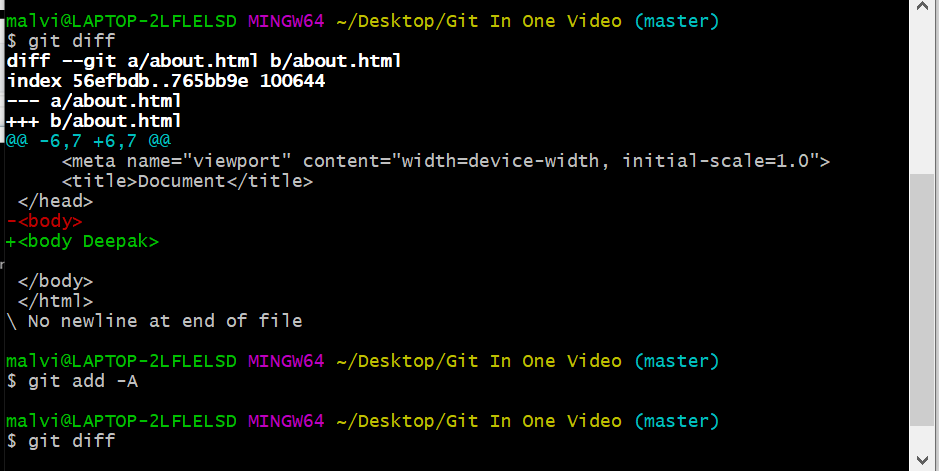


Enter Q to quit.

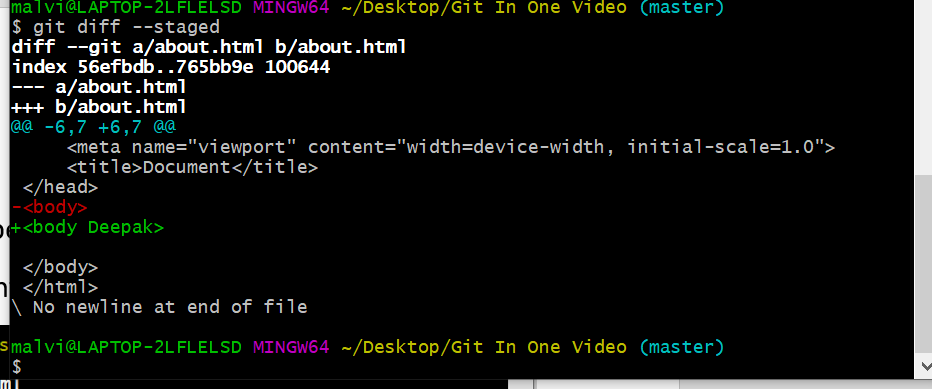
**git diff command**

It will tell us the difference between the current file and Staged file.

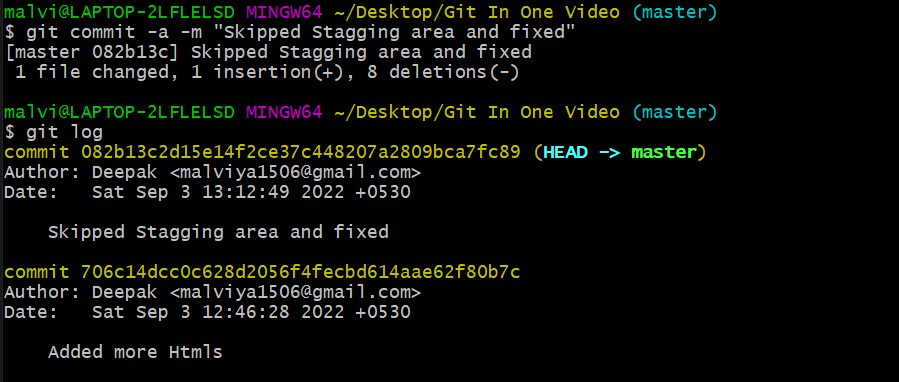
We have changed the current file.



Now, if want to compare the Staging area file with last commit file, use command as,

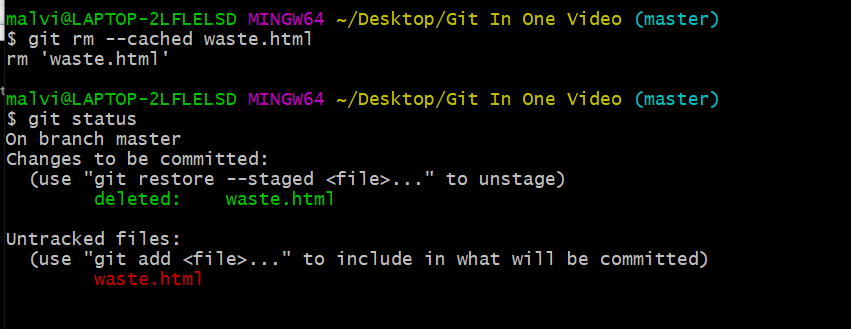


Now, suppose we want to skip the staging area and commit directly.

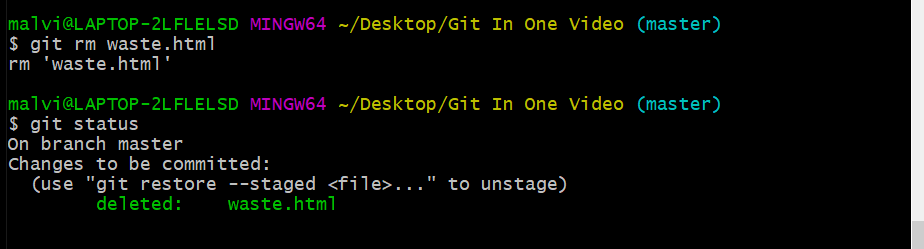


**Removing a file:**

To remove the file from “Staged” area, use below command



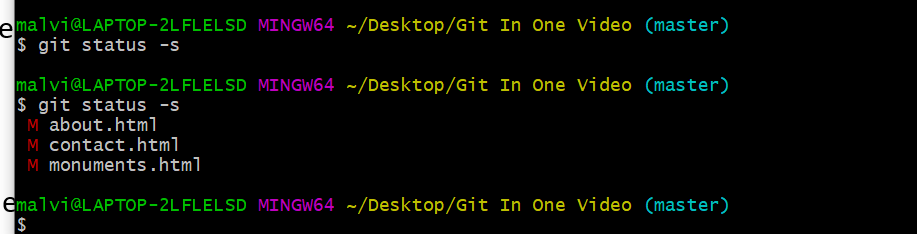
To remove the whole file, use below command,



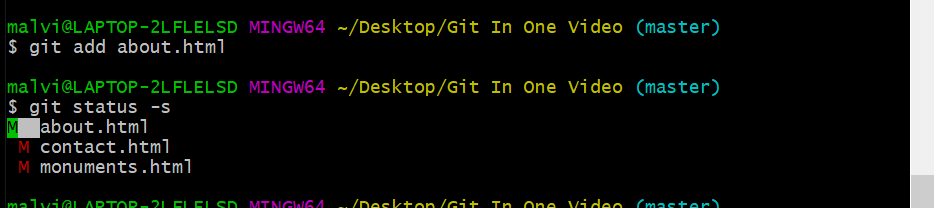
It will be permanently deleted.

Now, if we want to check the git status in short manner,

It will show that these files are modified.



Now, added a file and ran the command again



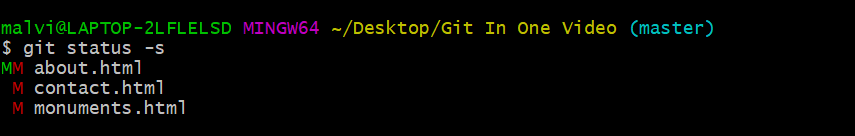
The M of about.html got shifted to its left.

We can see two boxes there.

Left: Staging Area(By Add)

Right: Working Tree(Current)

Now, if I change the file again, now, it is showing M at both the locations ie. Working tree and Staged Area

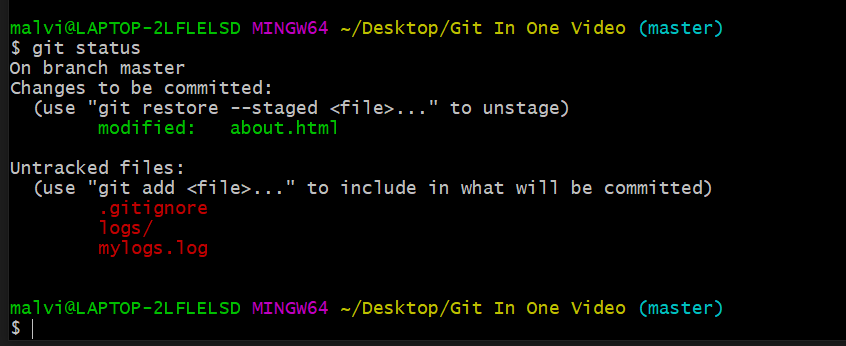


**Working with git ignore :-**

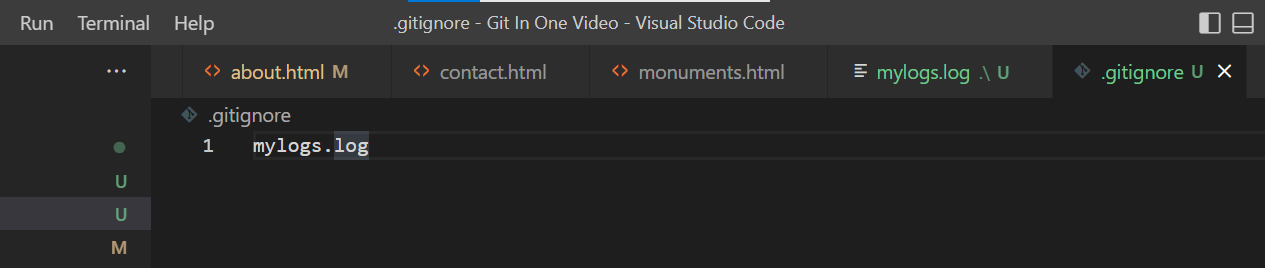
Now, suppose we have a file which is not important, and we can ignore that file, and we don’t want to commit it, like logs and automatic generated files.

(The files which are not required)

I have two files mylogs.log and different locations.



Now, if I will add these files in .gitignore file, there files will be ignored.



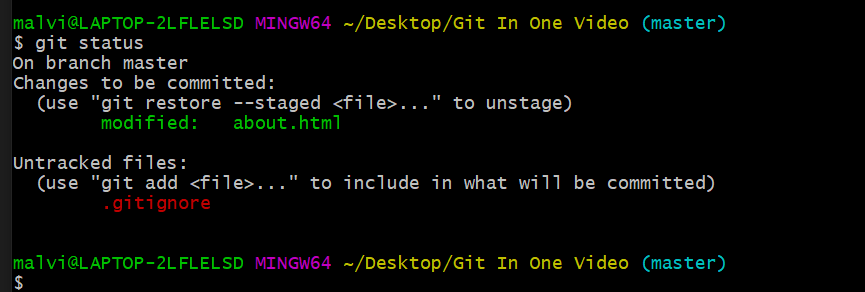
/mylogs.log (if we want the file of gitignore location)

\*.cpp

\*.log

Ignore/ (it is a folder)

Only gitignore we are able to see here now,

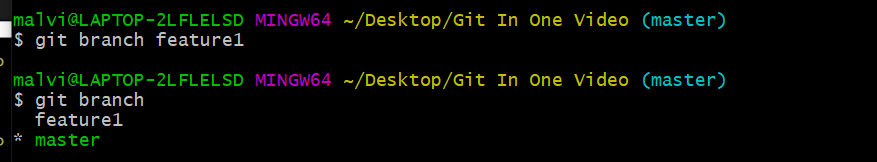


Add gitignore file and commit it.

**Working with Branches:**

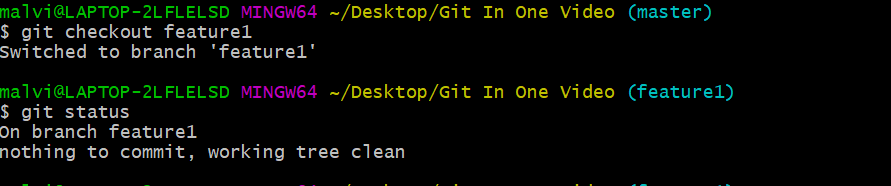
When you are working at some risky environment where if you will make any changes, you might lose your job because of any mistake.

So, in that case you prefer to work in branches.



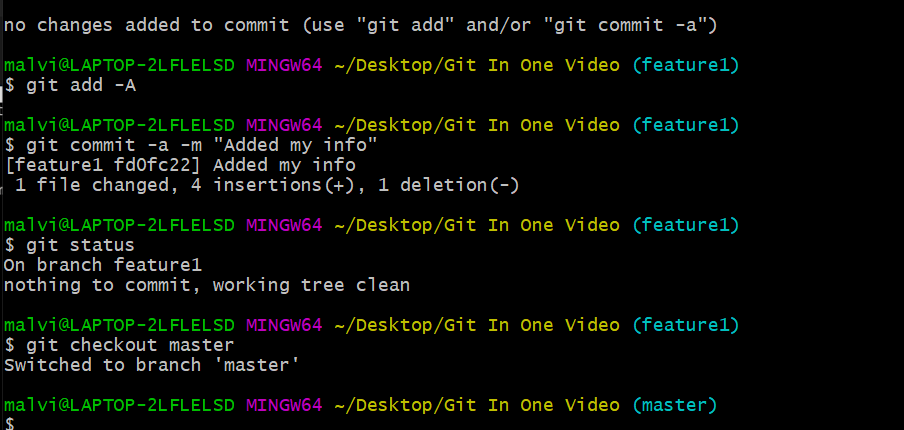
We have created a branch “feature1”, currently you are at “master” branch as that is green.

Changing the branch

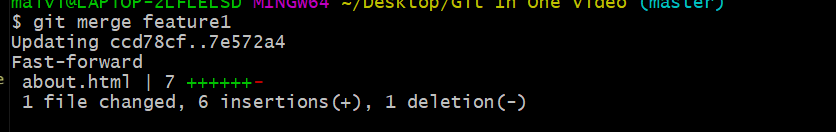


Now, we can work on feature1 branch, and we will merge it to master after successful execution.

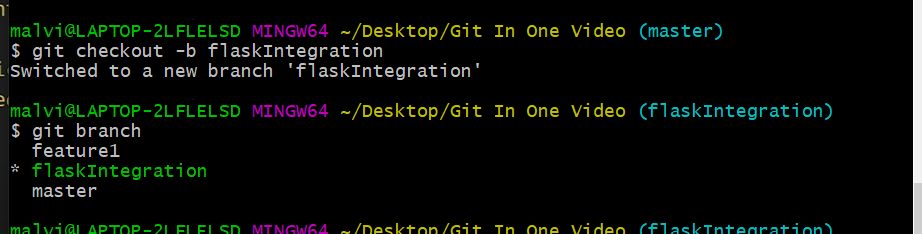
Now, you have made changes which are not required, so just checkout the same.



Now, we want to merge the master branch with feature1 branch,

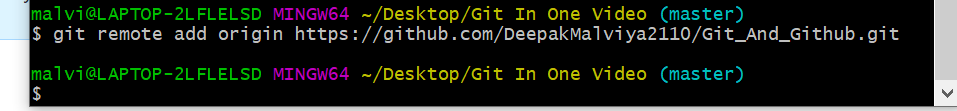


To create and go to that branch

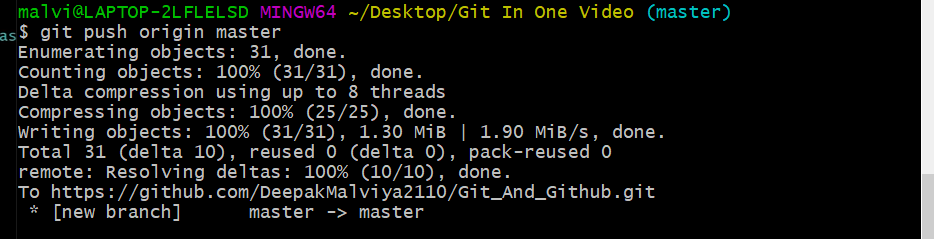


**Github:**

Create a github repository and add that repository in our project by using below command,



Now, we want to push our origin to the master repository.



Or use git push -u origin master

In case of private repository, some extra steps are required.