GIT is version control system.

* Easily file recovery (if any file is lost).
* Who introduced an issue and when.
* Rollback to previous working state (timestamp is granularity).
* Captures **snapshot** of the changes. ‘.git’ folder is created that have the project history. We can fetch latest version or any time version from this and could be moved to Working Directory.
* Most of the **operation is local**. It means we can work on the local machine and push the changes to Centralized Server. Operation like Push/Pull is not local (remote).
* GIT has **integrity**. Files are safe.
* GIT generally only **adds data**.

History

* Local Version Control System: Storage locally.
* Centralized Version Control System: Storage in centralized server.
* Distributed Version Control System: Many Can pull/push full history from Centralized Server.

Github is the hosting service that hosts the git repository. It is the website that manages the Version Control System. In background it uses GIT.

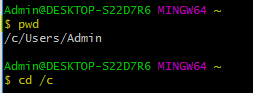
**GIT Installation:**

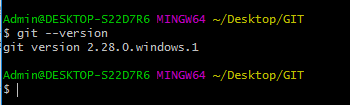
While installing GIT, we get below thing:

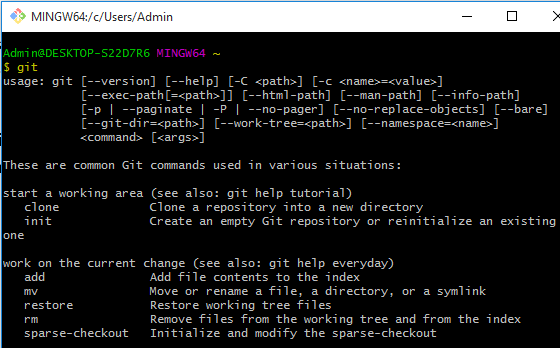
GIT Command Line Interface

GIT Bash (Terminal Program)

**Basic Command:**







Command git is used to fetch all the other commands that could be used by git.

**We have used GIT BASH as it is quite similar to LINUX prompt. If we fire same command in cmd also, we will get same output.**

**Also, we can perform same operations using GUI. That is also one of the way. It is same as WINScp.**

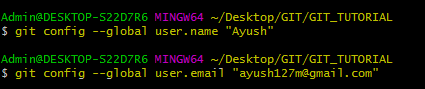
We can go to any folder in Windows OS. We can right click and then select ‘GIT Bash here’.



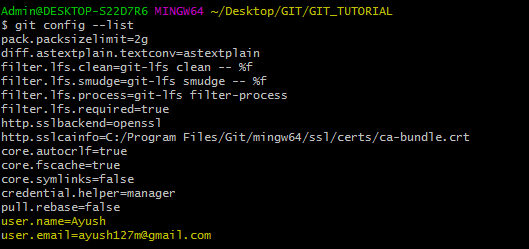
So currently, we are in GIT\_TUTORIAL directory.

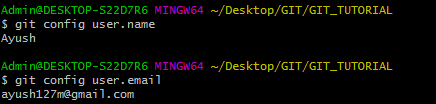
**Config User ID and Email**

This is required as we would be making commit in code. So other users must be able to know that changes had been done by whom.



On checking the configuration:





**GIT three stage architecture**

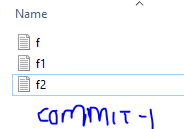


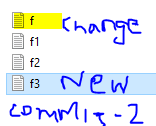
**Working Directory:** The folder location where we work. Like if we are working on local machine, we would be using Windows explorer, so the path that we would be using to work/ change and all, would be working directory.

**Staging Area:** This would be the files that we would want to commit to create new version.

**GIT Directory:** This is dot (.) file that would be keeping all the compressed file (all committed). It will fetch files as per user command. Like files commit 4 days ago. So it will fetch all files that had commit 4 days back. This will then show to the user.

Files could be moved from Working directory to GIT directory directly.

 The files will be in .git

 These will be added to Stage as f1 and f2 are not updated and we do not want to unnecessarily move.

The files will then move to .git.

**Working with first project**

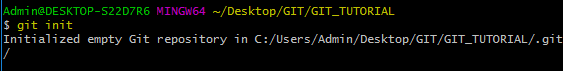
Created a folder in a location and then right clicked on the ‘Git Bash Here’ after getting into the folder. Then simply executed the command ‘git status’.



Once executed, we got the message saying that “Not a git repository”.

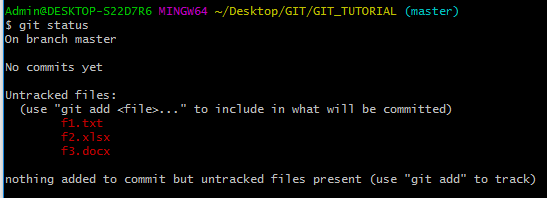
This command checks whether the directory is git repository or not.

To convert the repository into Git Repository, we need to use ‘init’ command.



Now the folder (repository) GIT\_TUTORIAL had been converted to Git repository.

On checking status again:



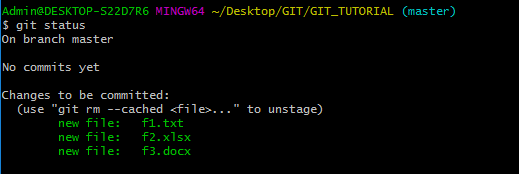
We can see that the message ‘No commits yet’ is displayed. Also the files present in Git location are shown over here.

To move the files in Staging Area, we need to write below command:



The files would be moved to Staging Area. ‘--a’ stand for All.

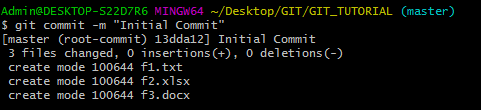
On checking the status again:



Here it can be seen that there are three files that are ready to be committed.

Previously all files were untracked files and then after running ‘--a’ (Staging), we have made them ready for commit. Currently files are in staging area.

To commit those files:

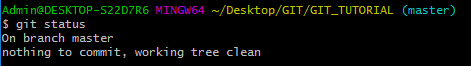


If we use only ‘git commit’, then a new terminal would be opened. And if we write as mentioned in above command then it would commit the files after displaying the required message

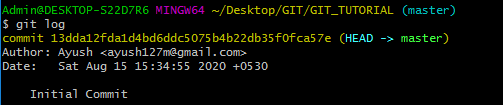
git commit –m “Message on Commit String”

So we can say that the snapshot of the repository had been taken.

On checking the status again:

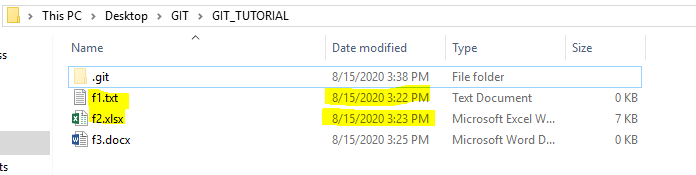


To know all the commits, we can use below command:

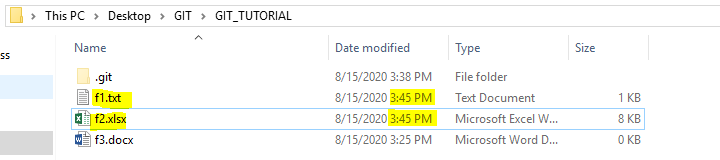


**Scenario where changing two files after commit but uploading only one for snapshot.**

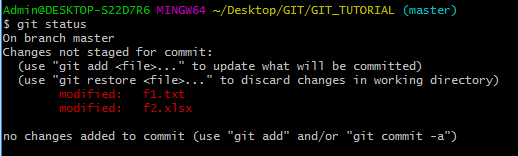
**Before:**



**After:**



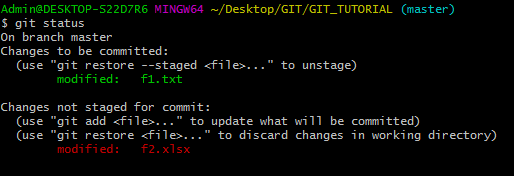
We have added data on f1 and f2 after first commit. Now let’s check the status:



Here both the files are shown in red color as they have been modified. Now suppose we only want to commit f1 file. We do not want to commit on f2 as still some more changes are required.

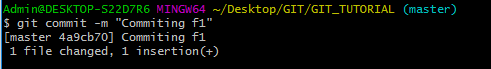


This command will add f1 into staging. It will not add f2. Now if we check the status again:

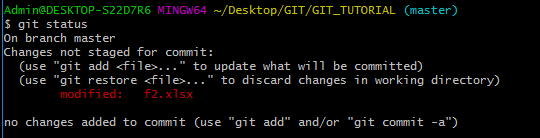


We can see that f1 is moved to staging (as shown in Green) and f2 is still shown in red as it also hd been modified. But we do not want to move it due to some reason.

Now we can commit f1. So command will be:

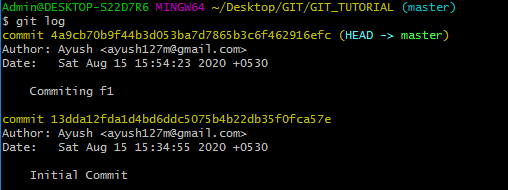


Now checking the git status:



Only f2 is shown as a modified. F1 is not shown as it already had been committed.

On viewing the log:



So the changes are moved to staging and the commit.

Also we rerun the command git init after our repository is initialized, then our repository will be reinitialized.