Git :

Git is a distributed version control system that is open-source. It is made to manage small to large work quickly and effectively. It was created to organize the development team's work. We can keep track of each other's work and collaborate in the same workspace thanks to version control.

Features :

- Open-source
- Speed
- Secure
- Distributed

version control system :

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.

Pointes about git :

Git is a source code management system.

Github is the cloud version of Git.

It is used for source management in software development and can be used to keep track of changes in the files of a project.

Git has two types of environments:

1. Local - Available Only in Local.
2. Remote - Shared Repository.

We can push the code from the local repository to the remote repository.

We can pull the code from the remote repository to the local repository.

`$ git init` :

The git init command creates a new Git repository.

It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository.

Most other Git commands are not available outside of an initialized repository, so this is usually the first command you'll run in a new project.

Executing git init creates a .git subdirectory in the current working directory, which contains all of the necessary Git metadata for the new repository.

`$ git clone` :

By using git clone we can create local or remote repositories.

git clone is primarily used to point to an existing repository and make a clone or copy of that repository in a new directory, at another location.

The original repository can be located on the local filesystem or on remote machine accessible supported protocols.

The git clone command copies an existing Git repository.

`$ git status` :

It shows the current status of your local repository.

$ git add :

$ git add command adds a change in the working directory to the staging area.

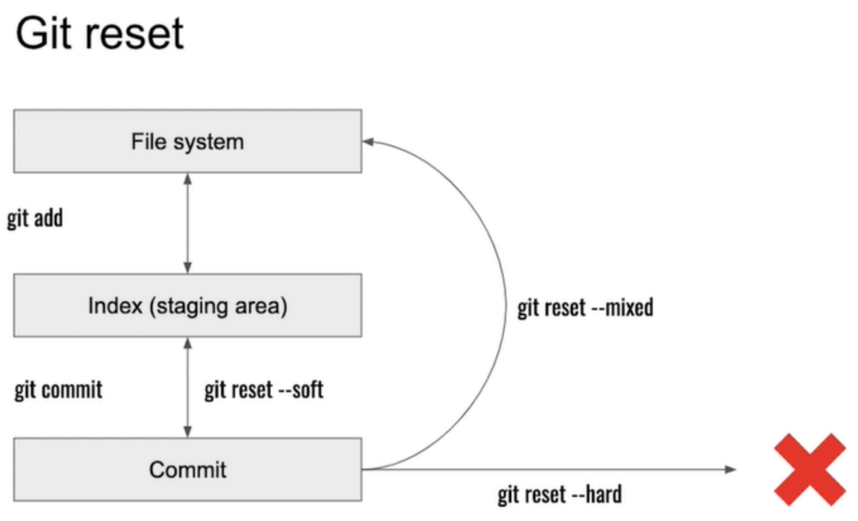
$ git add doesn't really affect the repository in any significant way—changes are not actually recorded until the user runs git commit .

1. $ git add <file name> - we can only add one file.
2. $ git add . - we can add multiple files.
3. $ git add -A & git add -all = these are the same.
4. $ git add*

	New Files	Modified Files	Deleted Files	
git add –A	✓	✓	✓	Stage All (new, modified, deleted) files
git add .	✓	✓	✗	Stage New and Modified files only
git add -u	✗	✓	✓	Stage Modified and Deleted files only

$ git reset

By using git reset rewind history without changing the contents of your local files.



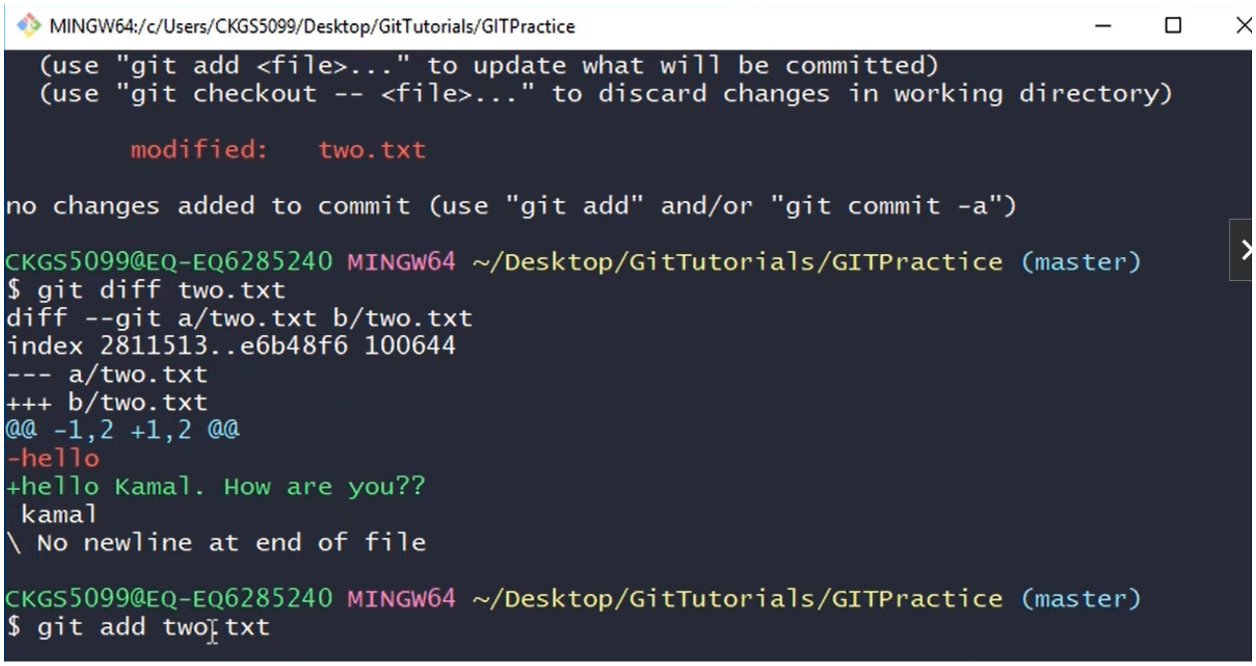
Staging to local repository

```
$ git commit -m ”Message here!”
```

By using the commit command we can update the code in the local repository.

Staging area :

The staging area is the middle ground between what you have done to your files (also known as the working directory) and what you had last committed (the HEAD commit). As the name implies, the staging area gives you space to prepare (stage) the changes that will be reflected on the next commit.

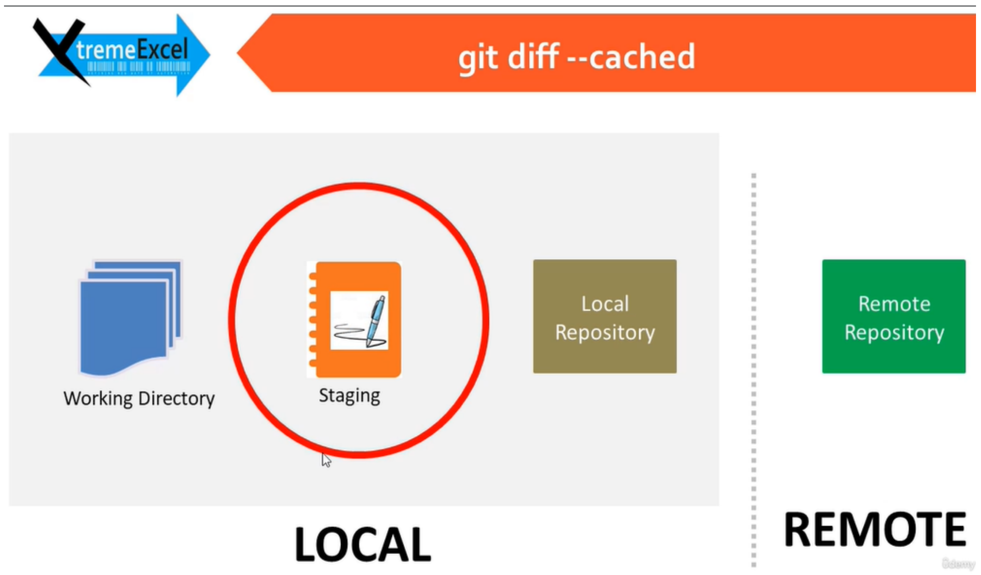


MINGW64:/c/Users/CKGS5099/Desktop/GitTutorials/GITPractice

```
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   two.txt

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

CKGS5099@EQ-EQ6285240 MINGW64 ~/Desktop/GitTutorials/GITPractice (master)
$ git diff two.txt
diff --git a/two.txt b/two.txt
index 2811513..e6b48f6 100644
--- a/two.txt
+++ b/two.txt
@@ -1,2 +1,2 @@
-hello
+hello Kamal. How are you??
 kamal
\ No newline at end of file

CKGS5099@EQ-EQ6285240 MINGW64 ~/Desktop/GitTutorials/GITPractice (master)
$ git add two.txt
```

If any changes in the file we can view those changes by using `$ git diff <file name>` .

- symbol shows the previous one.
+ Symbol shows the updated one or modified one.



Git reset --hard :

$ git reset --hard option resets the current branch tip, and also deletes any changes in the working directory and staging area. Therefore, it resets index entries to the state of the specific commit.

```
Before reset --hard :
     Two.txt

          --hello
          ++ hello mahesh
          How are you?

After reset --hard :
     Two.txt

          --hello
          How are you?
```

git rm :

$ git rm command can be used to remove individual files or a collection of files.

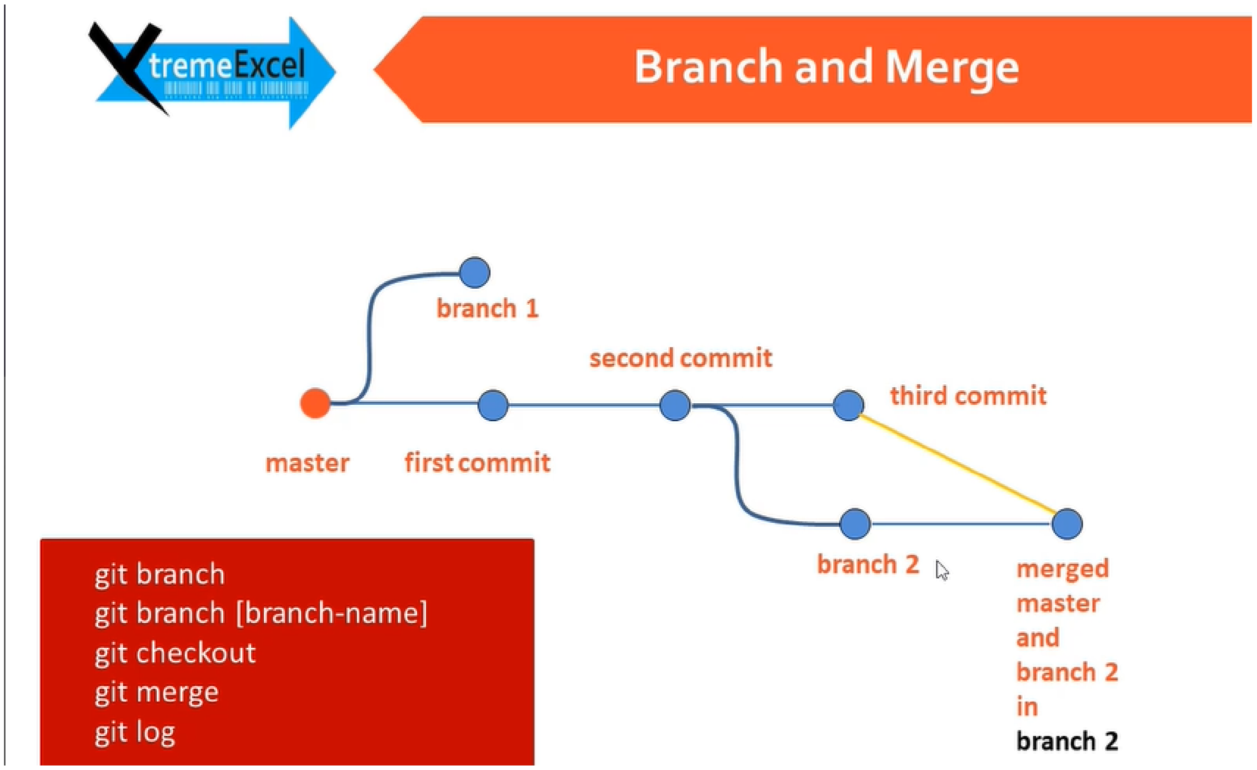
The primary function of git rm is to remove tracked files from the Git index.

git rm can be used to remove files from both the staging index and the working directory.

$ Git rm f <file name> - we can remove the file.
$ Git rm r <folder name> - we can remove the folder.

Git Branch and Merge :

By using this command we can create a new branch from the master.



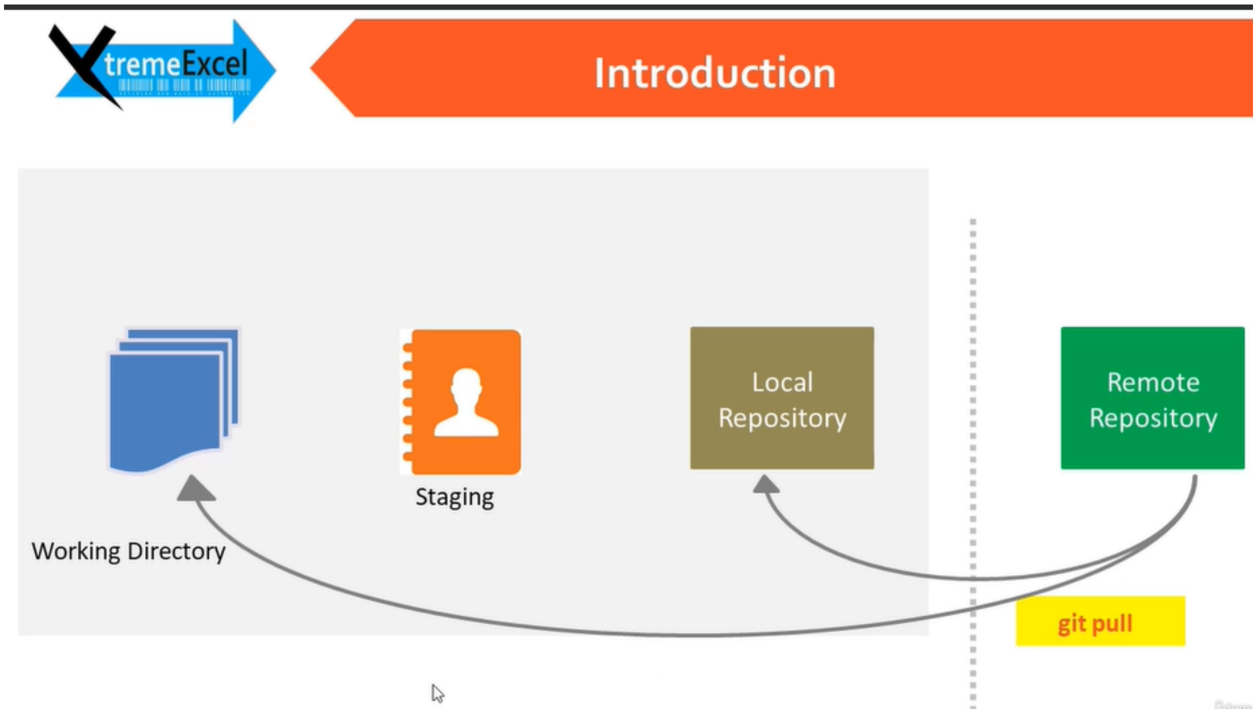
$ **git branch** - It returns the branches and also displays the current branch.

$ **git branch <branchname>** - We can create a new branch from master.

$ git checkout - By using this command we can switch over to another branch.

$ git log - Displays all of the commits in a repository's history.

$ git fetch - git fetch is a primary command used to download contents from a remote repository.



“ git fetch + git merge = git pull ”

How to delete the useless branch local repository?

Switch to master

$ git branch -D <Branch name>

Stash

Save the working directory

$ git stash - it takes users uncommitted changes(both staged and unstaged).

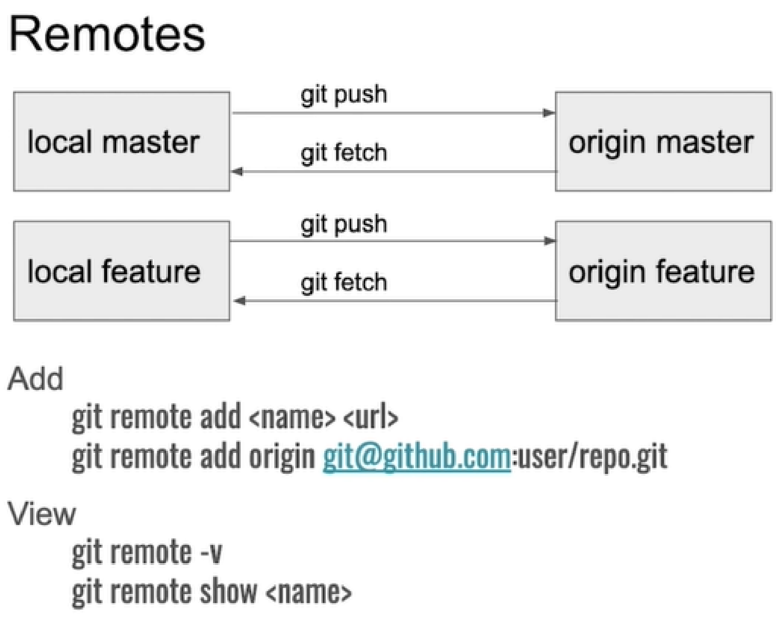
$ git stash pop - it can be used for reapplying previously stashed changes.Popping your stash removes the changes from your stash and reapplies them to your working copy.

$ git stash list - view the stashes.

$ git stash apply - leave in stash.

$ git stash drop - clear.

Remote:



Add

```
git remote add <name> <url>
git remote add origin git@github.com:user/repo.git
```

View

```
git remote -v
git remote show <name>
```