Git Hub Commands

Git is a modern and widely used **distributed version control** system in the world.

Version Control System

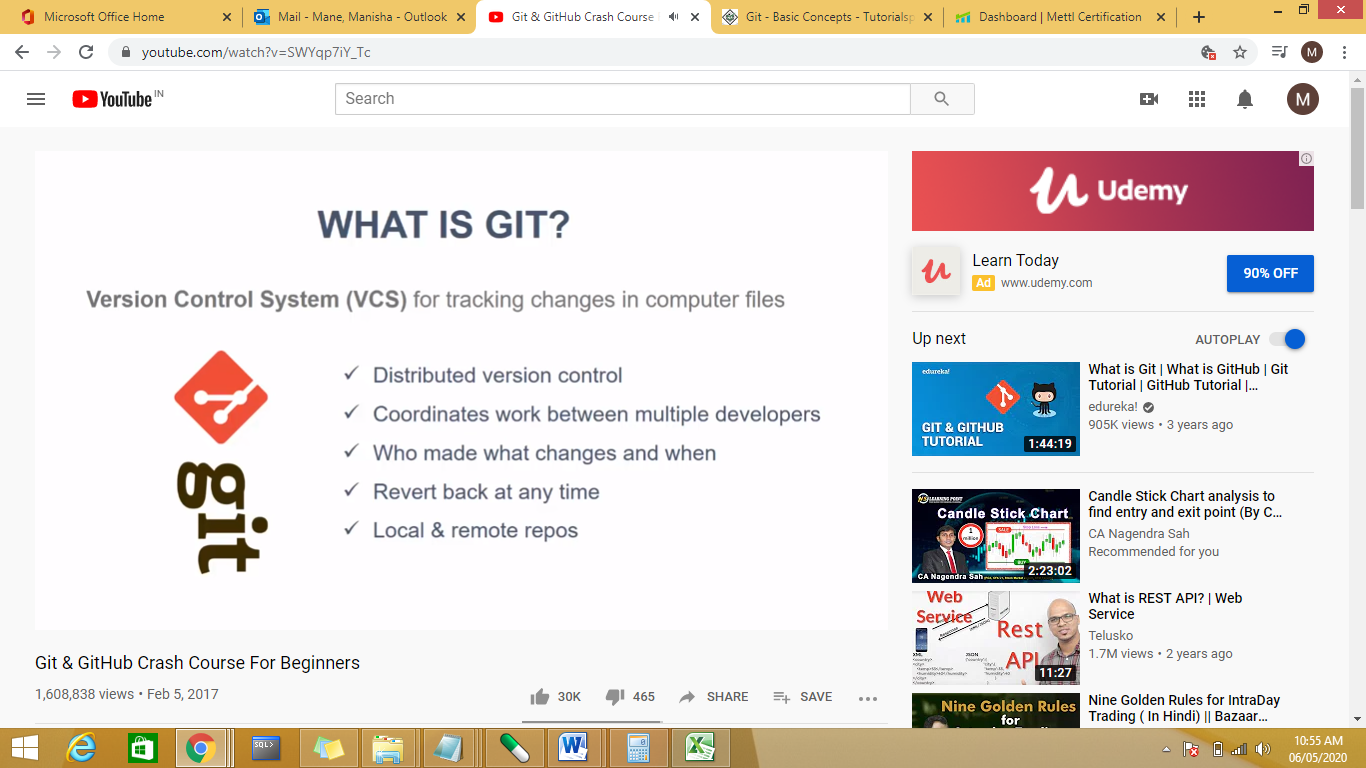
**Version Control System (VCS)** is a software that helps software developers to work together and maintain a complete history of their work.

Listed below are the functions of a VCS −

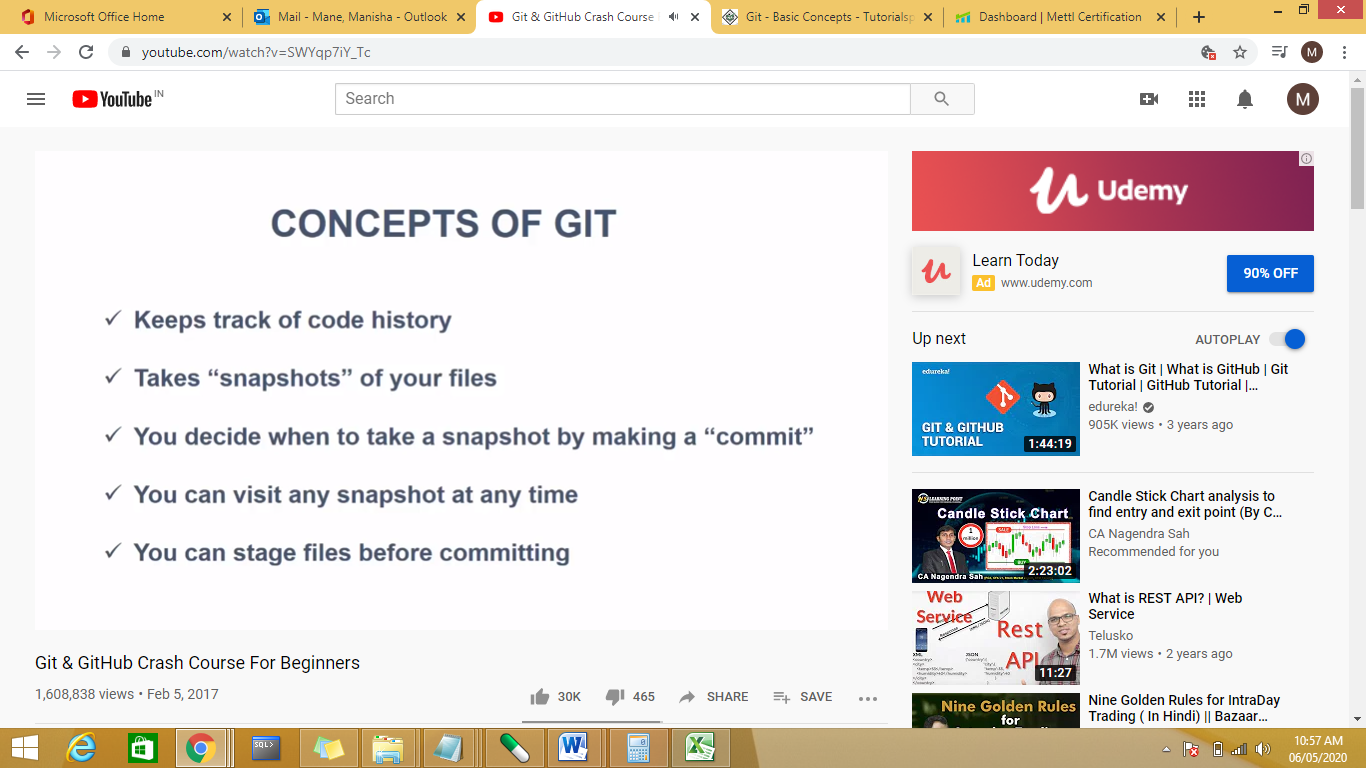
* Allows developers to work simultaneously.
* Does not allow overwriting each other’s changes.
* Maintains a history of every version.

Following are the types of VCS −

* Centralized version control system (CVCS).: Centralized version control system (CVCS) uses a central server to store all files and enables team collaboration. Unfortunately, if the central server goes down for an hour, then during that hour, no one can collaborate at all. And even in a worst case, if the disk of the central server gets corrupted and proper backup has not been taken, then you will lose the entire history of the project. Here, distributed version control system (DVCS) comes into picture.
* Distributed/Decentralized version control system (DVCS). DVCS clients not only check out the latest snapshot of the directory but they also fully mirror the repository. If the server goes down, then the repository from any client can be copied back to the server to restore it. Every checkout is a full backup of the repository.



Local repository:on the local machine when program does any changes the changes can be pushed to remote repository for example github.For local repository we don’t need internet but to push to remote we need to internet



## Advantages of Git

### Free and open source

### Fast and small

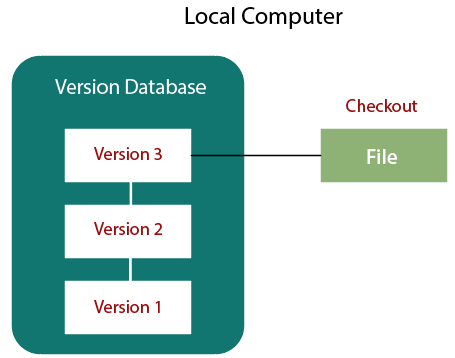
### Implicit backup

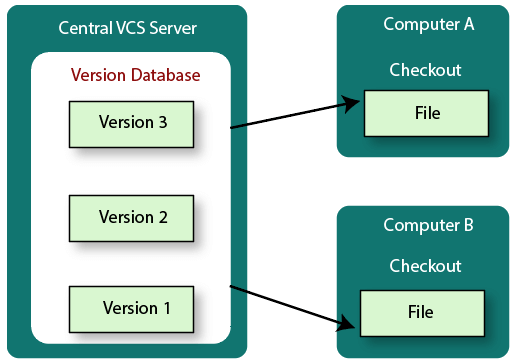
Security

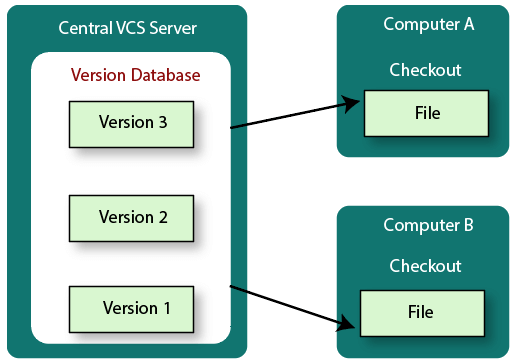
No need of powerful hardware

Easier branching









Every VCS tool provides a private workplace as a working copy. Developers make changes in their private workplace and after commit, these changes become a part of the repository.

**Working Directory and Staging Area or Index**

Let us see the basic workflow of Git.

**Step 1** − You modify a file from the working directory.

**Step 2** − You add these files to the staging area.

**Step 3** − You perform commit operation that moves the files from the staging area. After push operation, it stores the changes permanently to the Git repository.



Terminology:

### Blobs

Blob stands for **B**inary **L**arge **Ob**ject. Each version of a file is represented by blob.

### Trees

Tree is an object, which represents a directory. It holds blobs as well as other sub-directories.

### Branches

Branches are used to create another line of development. By default, Git has a master branch, which is same as trunk in Subversion. Usually, a branch is created to work on a new feature. Once the feature is completed, it is merged back with the master branch and we delete the branch

### Tags

Tag assigns a meaningful name with a specific version in the repository.

### Clone

Clone operation creates the instance of the repository. Clone operation not only checks out the working copy, but it also mirrors the complete repository

### Pull

Pull operation copies the changes from a remote repository instance to a local one.

### Push

Push operation copies changes from a local repository instance to a remote one.

### HEAD

HEAD is a pointer, which always points to the latest commit in the branch

### [Fetch](https://www.javatpoint.com/git-fetch)

It is used to fetch branches and tags from one or more other repositories

### [Index](https://www.javatpoint.com/git-index)

The Git index is a staging area between the working directory and repository.

### [Merge](https://www.javatpoint.com/git-merge)

Merging is a process to put a forked history back together.

### [Master](https://www.javatpoint.com/git-origin-master)

Master is a naming convention for Git branch. It's a default branch of Git.

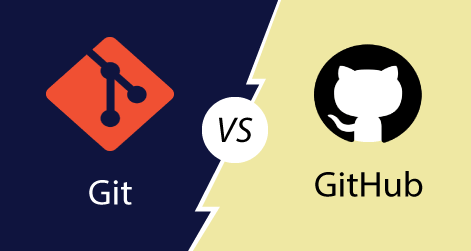
# What is GitHub?

GitHub is a Git repository hosting service. GitHub also facilitates with many of its features, such as access control and collaboration. It provides a Web-based graphical interface.

# Git vs GitHub

**Git is an open-source distributed version control system** which is available for everyone at zero cost. It is designed to handle minor to major projects with speed and efficiency. It is developed to co-ordinate the work among programmers. The version control allows you to track and work together with your team member at the same workspace.

:



While **GitHub is a Git repository hosting service**. It is a web-based service. GitHub facilitates with all of the features of distributed version control and source code management (SCM) functionality of Git. It also supports some of its characteristics in a single software to

|  |  |
| --- | --- |
| **Git** | **GitHub** |
| Git is a distributed version control tool that can manage a programmer's source code history. | GitHub is a cloud-based tool developed around the Git tool. |
| A developer installs Git tool locally. | GitHub is an online service to store code and push from the computer running the Git tool. |
| Git focused on version control and code sharing. | GitHub focused on centralized source code hosting. |
| It is a command-line tool. | It is administered through the web. |
| It facilitates with a desktop interface called Git Gui. | It also facilitates with a desktop interface called GitHub Gui. |
| Git does not provide any user management feature. | GitHub has a built-in user management feature. |
| It has minimal tool configuration feature. | It has a market place for tool configuration. |

Git, SVN Mercurial are other tools for VCS

Life Cycle of Git

## Stages in GIT Life Cycle

Files in a ***Git*** project have various stages like ***Creation***, ***Modification***, ***Refactoring***, and ***Deletion*** and so on.

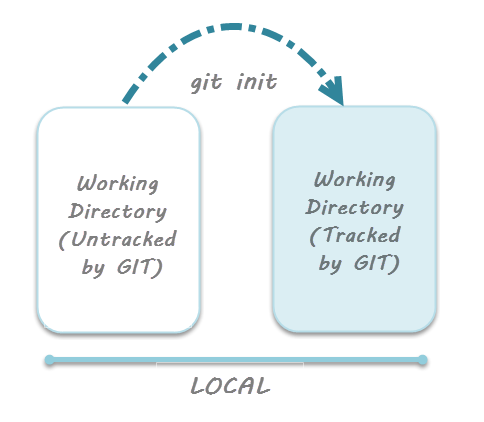
However when a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states:

* ***Working directory***
* ***Staging area***
* ***Git directory***

### *****Working Directory*****

Consider a project residing in your local system. This project may or may not be tracked by Git. In either case, this project directory is called your Working directory.

***Working directory is the directory containing hidden .git folder.***

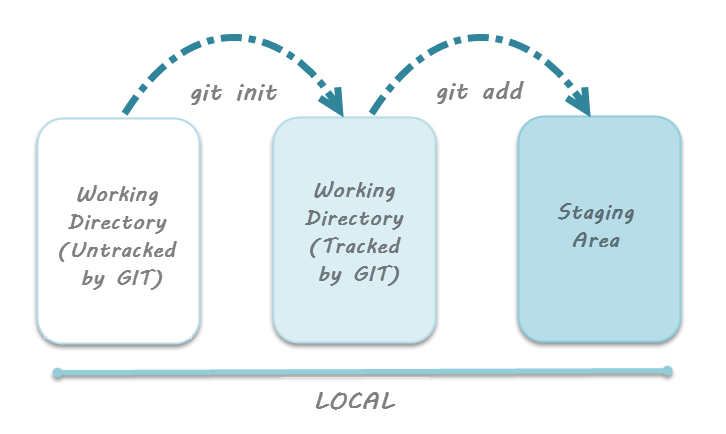


### *****Staging Area*****

While we’re in the working directory, we select the files that have to be tracked by Git. **Why do we need to this? Why don’t we track everything in the project?** That’s because some files in the project like**class files, log files, result files and temporary data files are dynamically generated.** It doesn’t make sense to track the versions of these files. **Whereas the source code files, data files, configuration files and other project artifacts contain the business logic of the application.** These files are to be tracked by Git are thus needs to be added to the staging area.

**In other words, staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.**

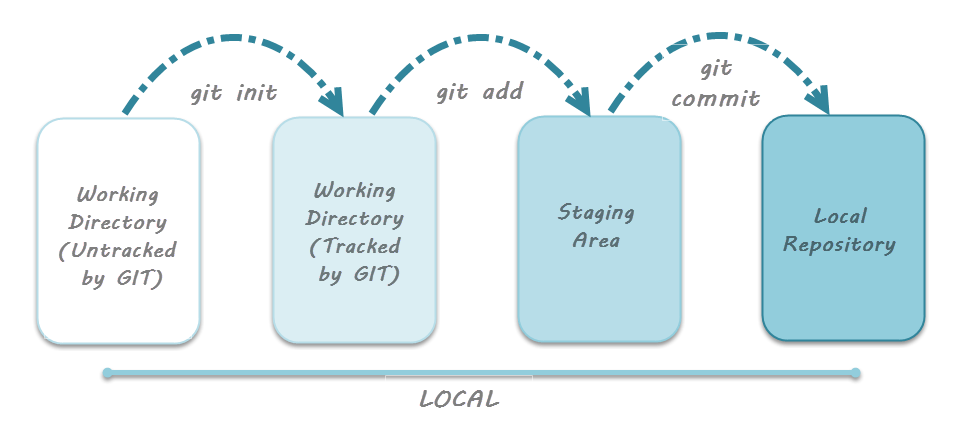
It’s important to make a quick note of the term called ***indexing*** here. **Indexing**is the process of adding files to the staging area. In other words, index constitutes of files added to the staging area**.**This term will be explained again in the coming tutorial on ***Git terminologies***.

***Note:*** ***git add*** – Command to add files to staging area.

### ****Git Directory****

Now that the files to be committed are grouped and ready in the staging area, we can commit these files. So, we commit this group of files along with a commit message explaining what is the commit about. Apart from commit message, this step also records the author and time of the commit. Now, a snapshot of the files in the commit is recorded by Git. The information related to this commit (names of files committed, date and time of commit, author of commit, commit message) is stored in the Git directory.

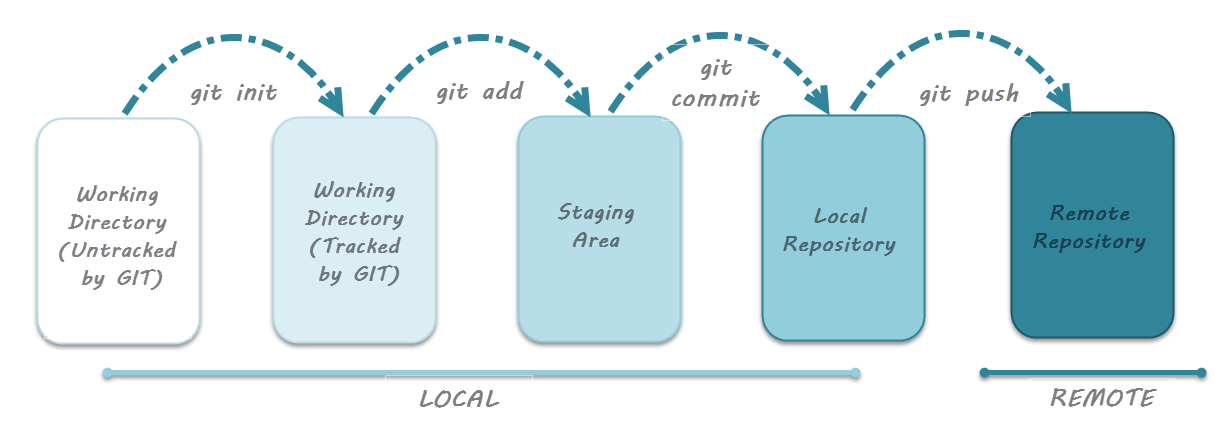
***Thus, Git directory is the database where metadata about project files’ history will be tracked.***

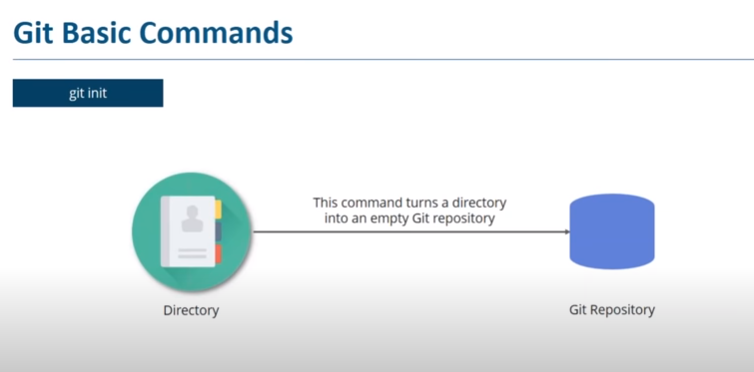
***Note:*** ***git commit -m”your message”*** – Command to commit files to Git repository with message.

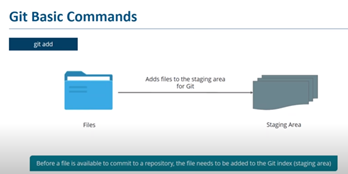
### ****Additional Lifecycle Stage with Github****

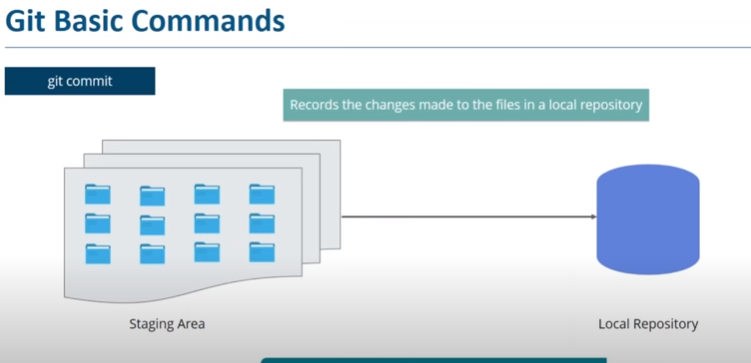
you can track versions of your files by using only Git. That is, ***Github is needed when you want to collaborate and publish your code*** to a team or community. Thus, it helps to remember that the Git cycle doesn’t conventionally involve Github.

However, we work in teams and collaborate with multiple people on a given project. This makes it imperative to understand the additional stage related to Github. While dealing with Github, there’s a concept of ***Remote repository*** and a related process called ***Pushing*** the files**.**

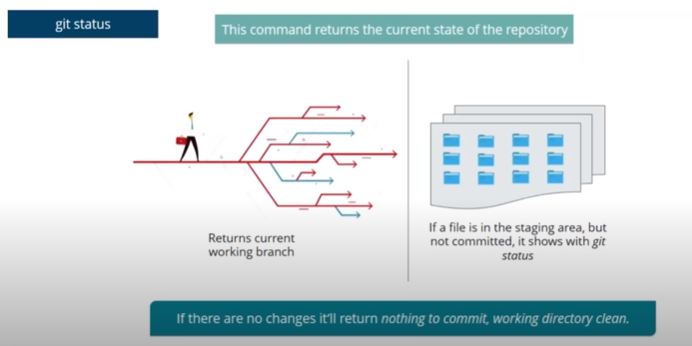


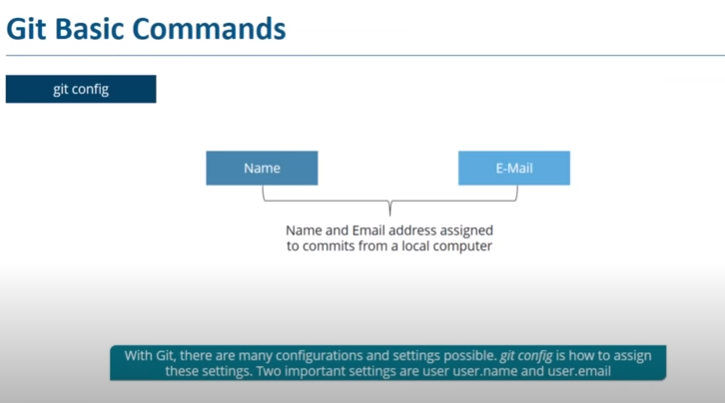






 Each commit has unique ID.



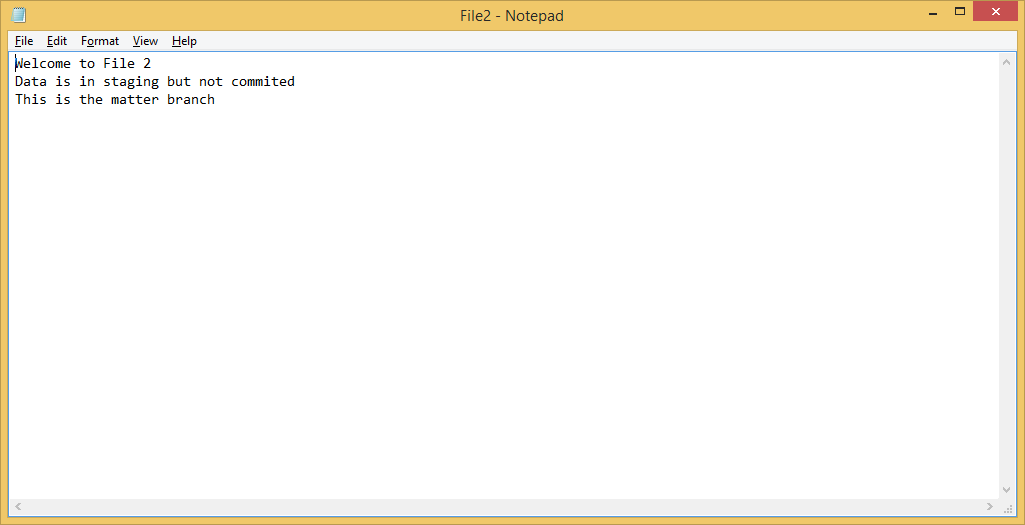


Branching:

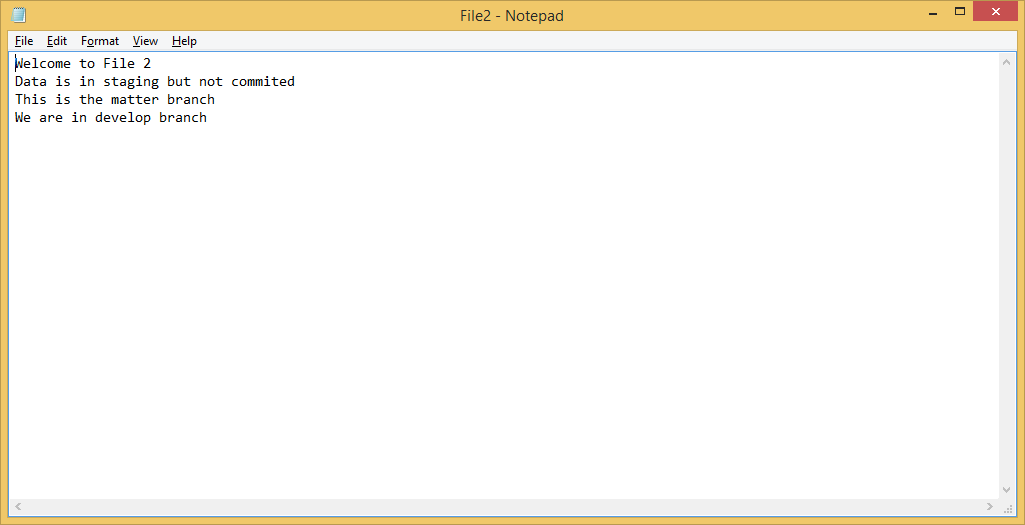
Branches are used to create another line of development. By default, Git has a master branch, which is same as trunk in Subversion. Usually, a branch is created to work on a new feature. Once the feature is completed, it is merged back with the master branch and we delete the branch

For Example I am having a working model and want to add new feature , To keep the master copy unchanged we will create a new branch (duplicate copy) of master and do the changes in the new branch Once the feature is completed, it is merged back with the master branch and we delete the branch

$git checkout master



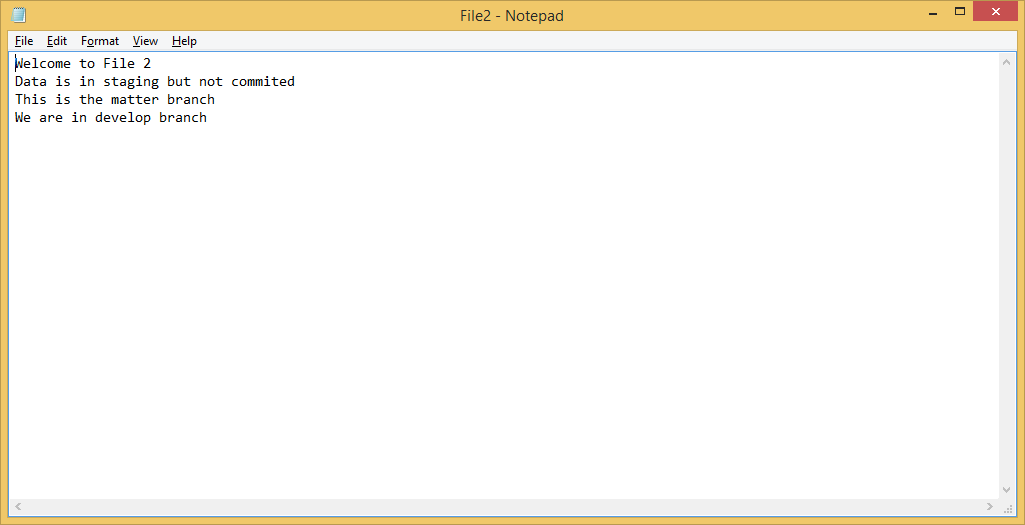
$git checkout develop



Create a new branch

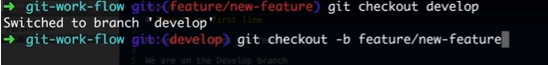
Dell@Admin MINGW64 /g/ManishaGit/Myprojct (develop)

$ git checkout -b feature/new-feature //-b is branch



Flow

Master🡪develop🡪feature/newfeature



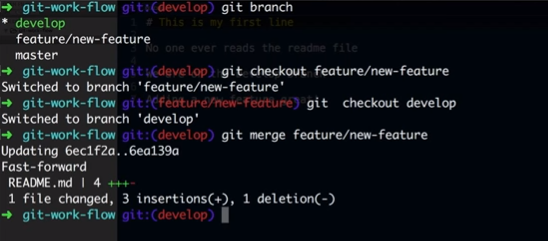
Master

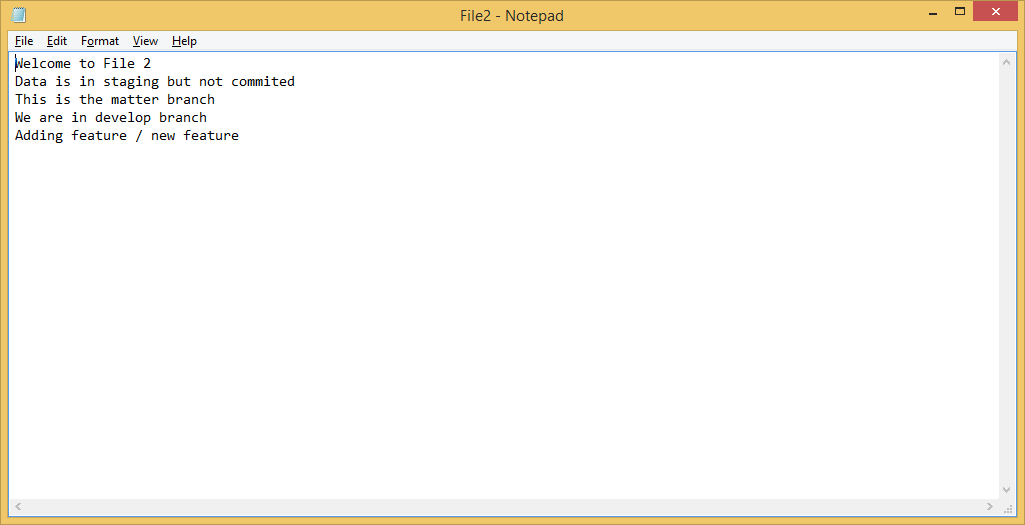
Master🡪develop assigned to programmer A

Master🡪develop🡪fetaure/new-feature assigned to programmer B

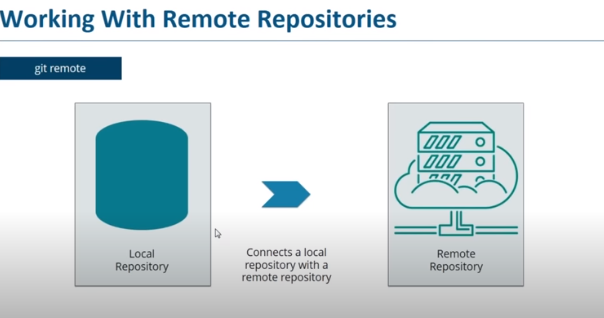
Now the code will be isolated from each other

Merging – for integration of all features developed by diff devlopers





Git with Remote repository



Clone

Local working copy of remote repository.

