Version control system

A version control system records the changes in a file or set of files over a time and we can recall those specific versions later. So ideally, we can place file in the computer version control system. Version control system allows the users to revert the files to previous state, revert the project to previous state, identify the changes over a time, see who last modified something might causing problem, and you can recall the lost files.

Example: Git, SVN

Distributed version control system

A distributed version control system is a form of version control system complete code base.

Include Full history, mirrored in every developer’s computer.

Compared to centralized version control system, this enables automatic management branching and merging speeds of the operations.

It has the ability work offline; it does not depend single location for backup.

The most popular version control system git is distributed version control system.

In distributed version control system, the user has local copy of repository. So, the clients don’t just checkout the latest snapshot of the file even it fully mirrors the repository.

The files contain all the files and metadata of the file present in the main repository.

You can create branches, commit changes, view logs even you are in offline.

Example: Git

Centralized version control system

Centralized version control system uses central server to store all the files and enables team to collaboration.

Centralized version control system is based on idea that there is single central copy of your project somewhere else and programmers will commit changes in central copy.

Committing means record changes in central copy.

Those systems have single server that contain versioned files, and some clients check files from the central place

Example: SVN

Git

Git is a distributed version control system.

Git keep tracking changes on the files in the computer and coordinates the work on the files among multiple users.

Git does not necessarily rely on central serves to store the date of projects.

Usually, every user clone’s repository and have fully history of their project in their own hard disc.

It is a copy of original file which original stored in thirty party.

Git helps keep track changes you made in the code.

While working any errors occurs in code and don’t know what causing simple you can revert back to the previous state.

Recover the files.

Git features

Git compactable for all the operating system using these days.

Non-linear development: it allows all the users to work all over the world to perform operation on a project remotely.

branching

Speed

Reliable

Lightweight

Open source

Secure

Git workflow

Working directory staging area local repo remote repo

>> Add >>> commit >>> push

Merge fetch

pull

Create a file and add to staging area. After adding file to staging area commit it to local repos. After commit push it to the remote repo [master default branch]

when new employee joined into your project. he will clone it from remote repo

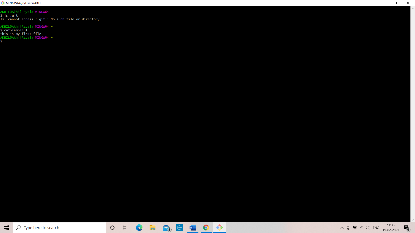
fork is for taking file from remote repo to our remote and clone it.

Git add: It is a command used add the file from working directory to the staging area.

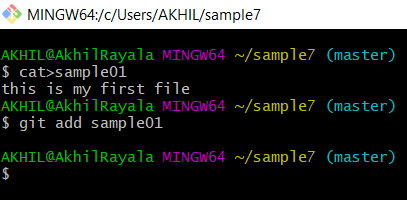
Suppose you created sample file

syntax

cat>sample file after creating the file add that file to the staging area.



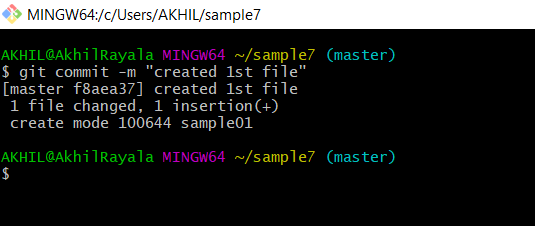
Using command git add [name of the file or directory]



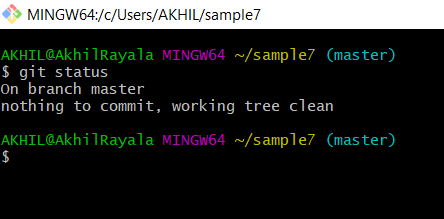
Git commit: It is a command used to add the file from staging area to the local repository.

Git commit used to add the file from staging area to the local repository.

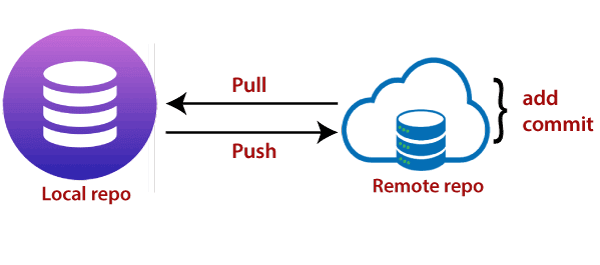
Git commit -m “1st file created”



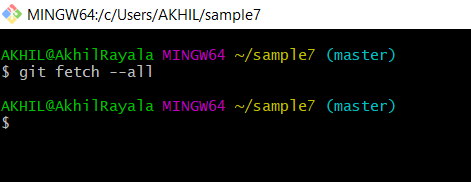
Git status: git status command used to check the status



Git push: It is used to push to file from local repository to the remote repository.



Git fetch: It is used to get the file from the remote repository to the local repository.



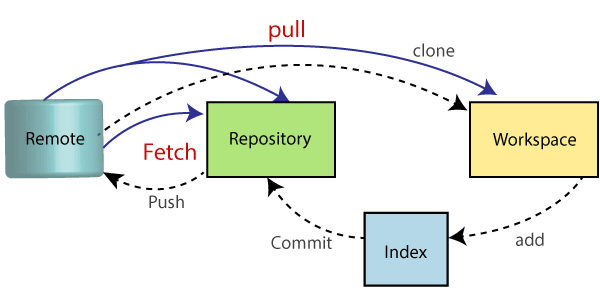
Git fetch [branch URL] [branch name] to fetch a branch

Git merge: It is used to get the file from the local repository to the working directory.

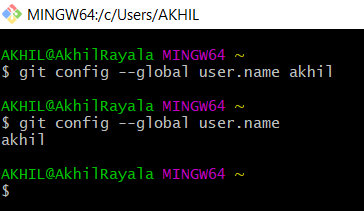
Copy the URL



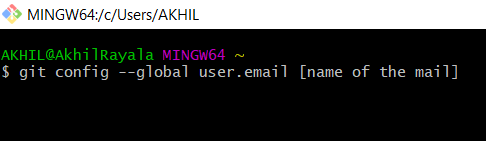
Git pull: It is used to get the file from the remote repository to the working directory. Pull equal to the fetch and merge.

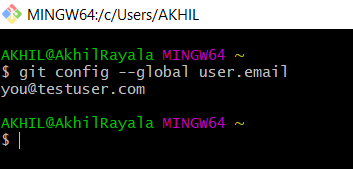


Git config - -global user.name [name] to add the user



Git config - -global user. Mail [mail] to the mail

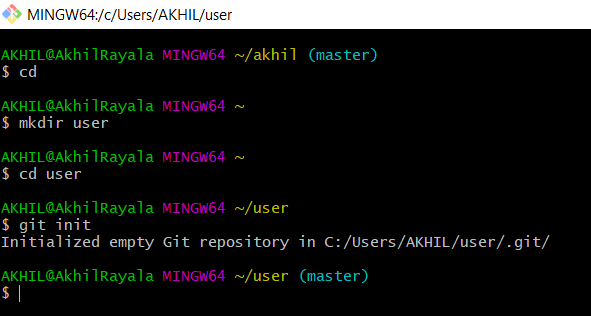




Git init: Git init is the command first we run in the git. Git init is used to create an empty or blank repository. A repository is collection of files or project related information.

The init creates. git subdirectory in the present working directory.

Init runs outside of the repository



Repository: it is the collection one or more files or collection of projects.

Merge: if you want to modify the files in the remote repository you will create a branch and modify or make changes what you want merge it with remote repository.

Merge is used to join the branches.

Git merge [branch name]



Staging area: in working directory what we had done are first added to staging area before it commits to the local repository before commit, we will add number of files to the staging area.

Clone: git clone takes the local copy of remote repository for example: a new person joined in your project; he wants your complete project details so he will clone the original repository copy from the remote repository.

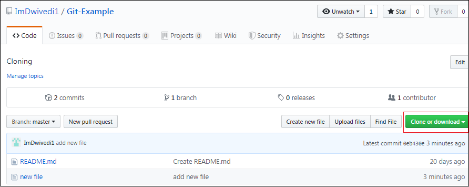
Git clone [ repository URL]

1st step

Open GitHub and navigate to the main page of the repository.

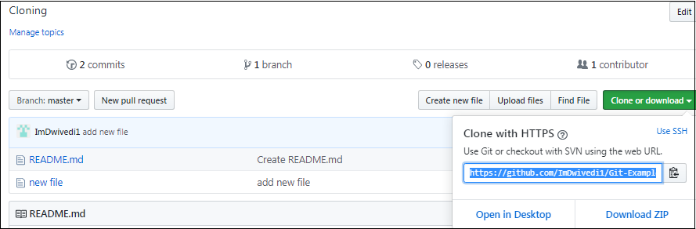
Step 2

Under the repository name, click on **Clone or download**.



Step 3

Select the **Clone with HTTPs section** and **copy the clone URL** for the repository. For the empty repository, you can copy the repository page URL from your browser and skip to next step.



Step 4:

Open Git Bash and change the current working directory to your desired location where you want to create the local copy of the repository.

Step 5

Use the git clone command with repository URL to make a copy of the remote repository. See the below command:

1. $ git clone https://github.com/ImDwivedi1/Git-Example.git

Stash: generally, stash meaning store the date in a hidden place in git stash means the same.

When we are working in a project half-work done in project but you want switch the branches but you don’t want to commit that work so stash allows you to do.

Git stash allows you to switch the branches without committing the current project.

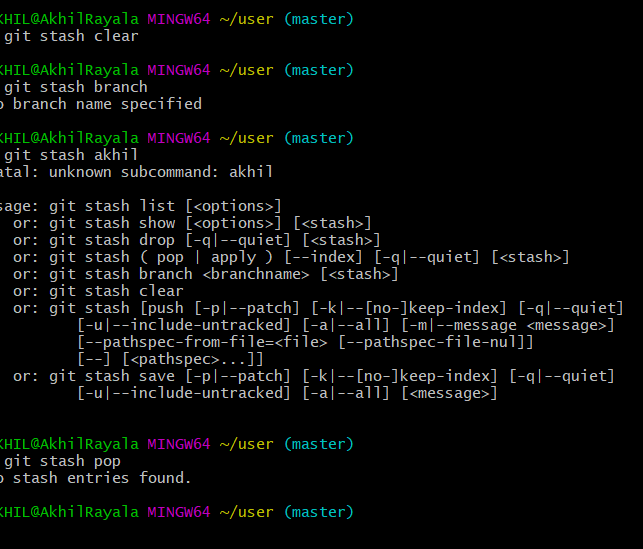
Git stash

Git stash list

Git stash branches

Git stash clear

Git stash pop



Fork: forking repository allows us experiment on the project without affecting the original project. Forking is used propose some modification on some one’s project

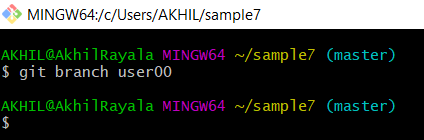
Forking is taking the existing project as a starting point.

Fork is used to take the file from the remote repository to our remote.

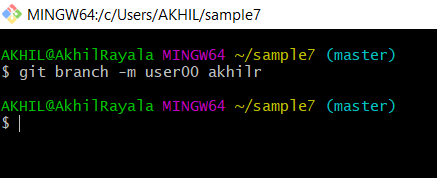
Git push origin master

Branch: branch a version repository which diverges from the main working repository. Using branch, we can rename, list, delete etc. master branch is the default branch in the git. If you want add new things create a branch add that and merge that branch with existing branch.

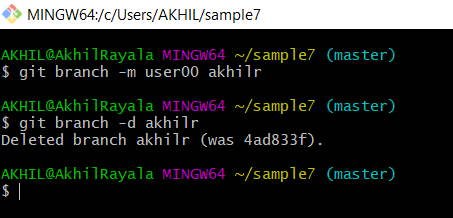
Git branch [branch name] to create a branch



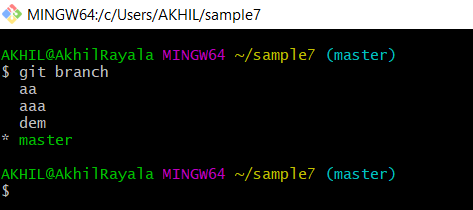
Git branch -m [old branch name] [new branch name] to change the branch name



Git branch -d [branch name] to delete a branch

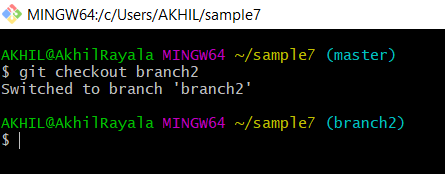


Git branch or git branch list --- to know the branches list

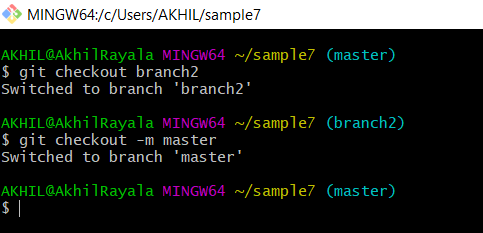


Git push origin -delete [branch] to delete the remote branch

Git checkout [branch name] to checkout between two branches without commit git checkout command is used

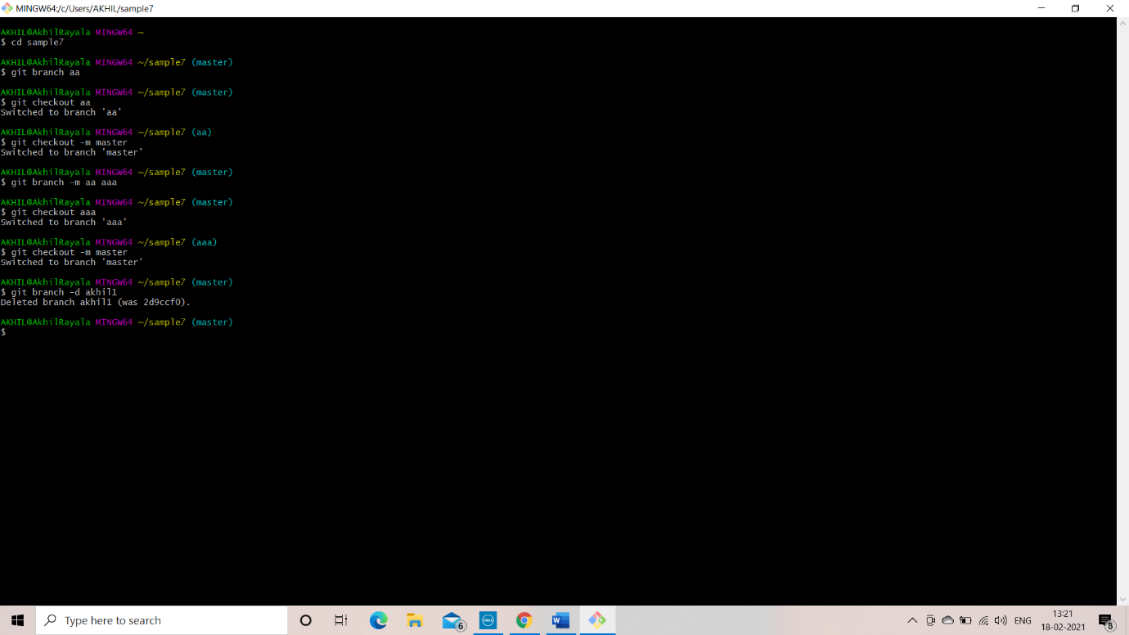


Git branch -m master --- switch to master branch from any other branch



Git merge [branch name] it allows to merge another branch with the active branch.

Ls .git



Ls .git

