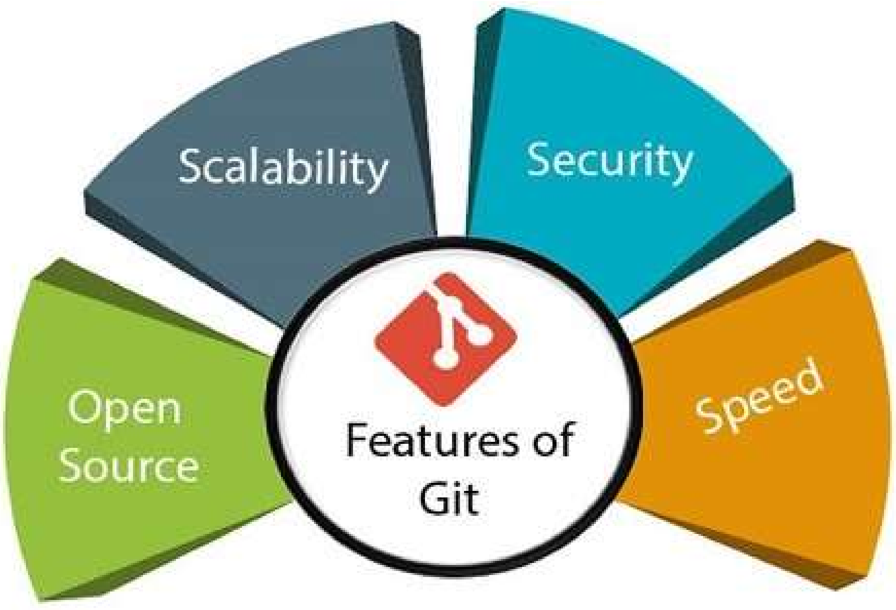
GIT

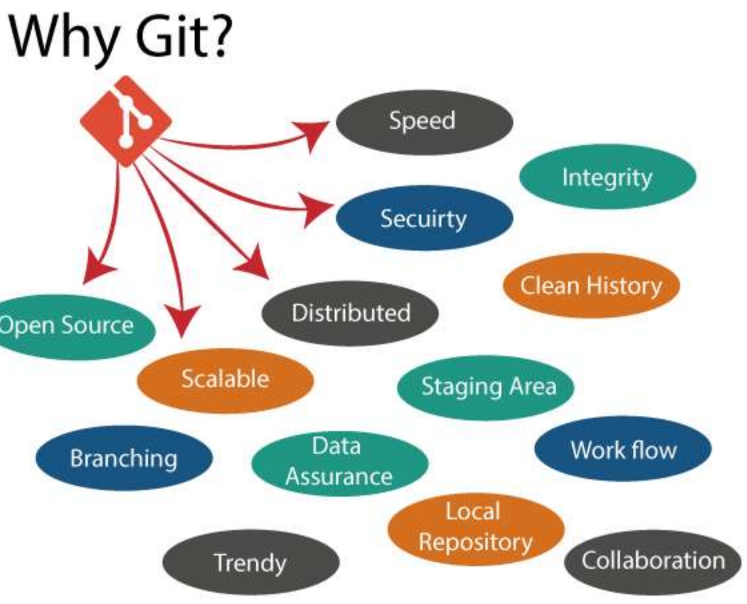
Introduction and Importance

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

What does Git do?

- Manage projects with Repositories
- Clone a project to work on a local copy
- Control and track changes with Staging and Committing
- Branch and Merge to allow for work on different parts and versions of a project
- Pull the latest version of the project to a local copy
- Push local updates to the main project





What is GitHub?

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

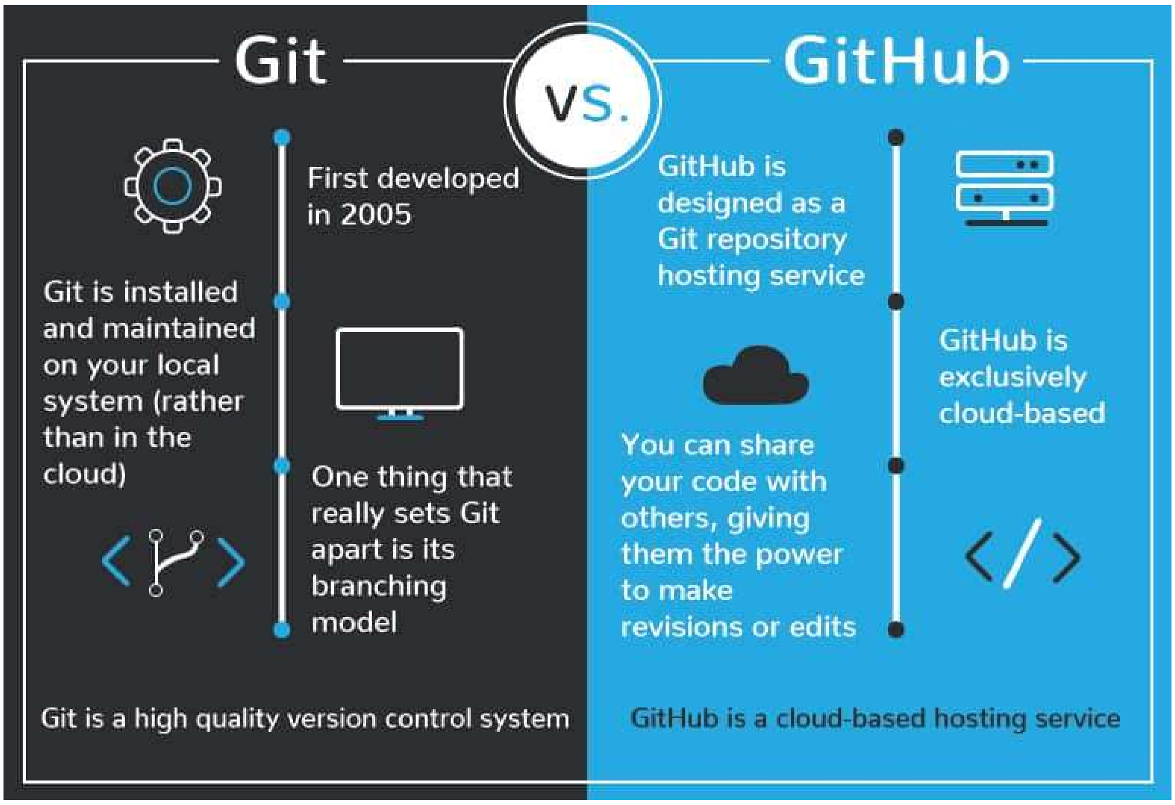
GitHub is an American company. It hosts source code of your project in the form of different programming languages and keeps track of the various changes made by programmers.

It offers both distributed version control and source code management (SCM) functionality of Git. It also facilitates some collaboration features such as bug tracking, feature requests, task management for every project.

GitHub
Collaboration
Bug Tracking
Branches
Git Repositories
Project Management
Team Management
Code Hosting
Track and assign tasks

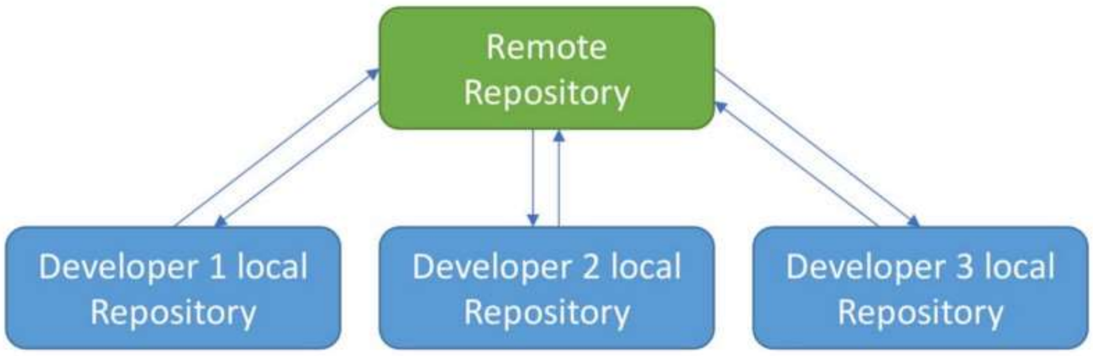
Differences between git and github

Git	Github
Software	Service
Local tool on system	Remote tool hosted on web
Focussed on version control and code sharing	Centralized source code hosting
Open source	Free and paid both
Git commands like push, pull, commit, add are used in git bash. Also provided a GUI based interface also.	It is a GUI.


Git
VS.
GitHub
First developed in 2005
Git is installed and maintained on your local system (rather than in the cloud)
One thing that really sets Git apart is its branching model
GitHub is designed as a Git repository hosting service
GitHub is exclusively cloud-based
You can share your code with others, giving them the power to make revisions or edits
Git is a high quality version control system
GitHub is a cloud-based hosting service

Use Case Scenario

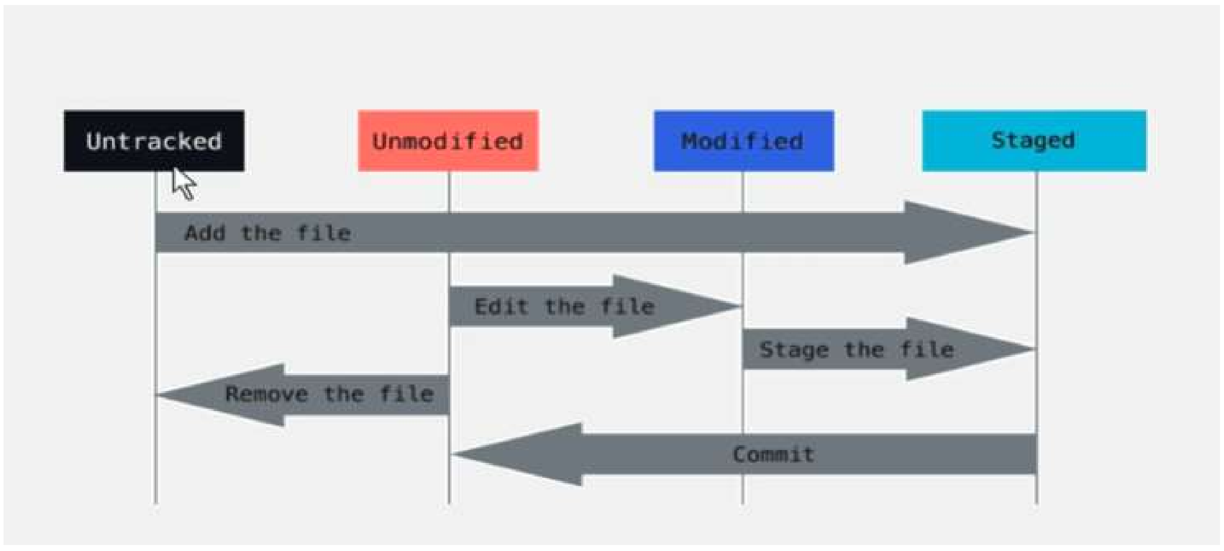
Each developer in a project will work in their local repository but eventually, they will push the code into a remote repository. Once the code is in the remote repository, other developers can see and modify that code. This makes the work easy, fast, and updating.


Remote Repository
Developer 1 local Repository
Developer 2 local Repository
Developer 3 local Repository

Concept of a Repository

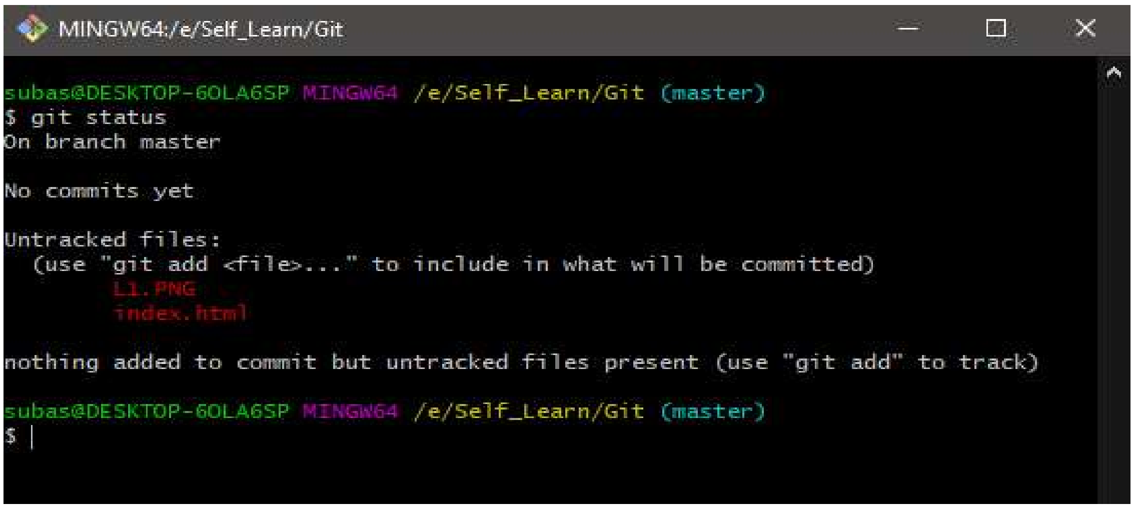
A repository is a central place in which an aggregation of data is kept and maintained in an organized way in a github account server or local git in your computer system. It is just a folder which is centralized. We can take an example of a github repository to well understand it. These repositories contain all your code and whatever you push in it.

Stages:



Untracked:

- We are not tracking files with the help of git.
- We add the file to say git to record the changes in our files.
- We put the files in stages to record changes.

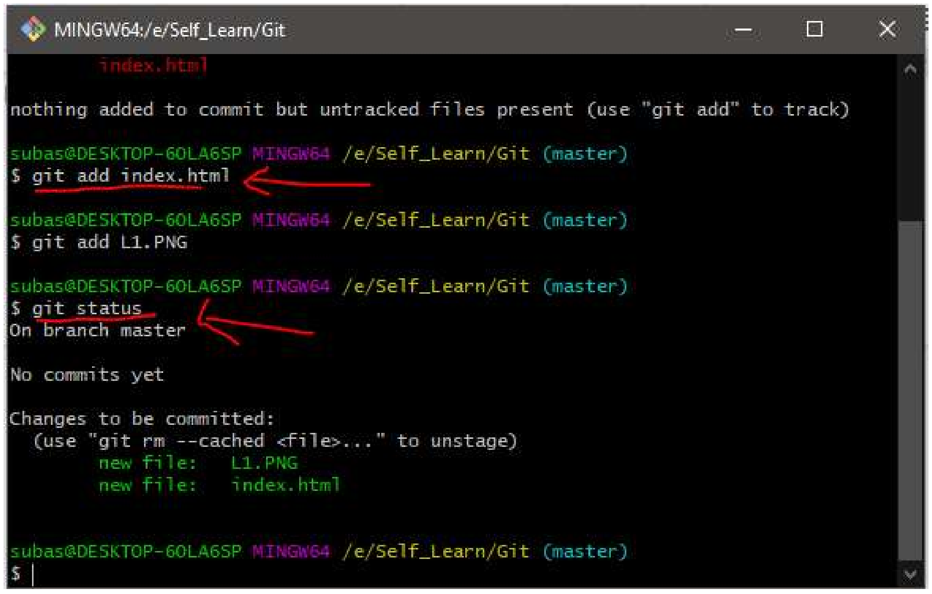


```
subas@DESKTOP-6OLA6SP MINGW64 /e/Self_Learn/Git (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        L1.PNG
        index.html

nothing added to commit but untracked files present (use "git add" to track)

subas@DESKTOP-6OLA6SP MINGW64 /e/Self_Learn/Git (master)
$ |
```

Staged:

- We keep those files in staging area that we want to commit
- We commit the files once we get it on staging area
- The files committed are gone to the Unmodified area then.

Using “ git add ‘filename’ " command to add the files. i.e we are moving the files to the staging area



```
MINGW64:/e/Self_Learn/Git
        index.html

nothing added to commit but untracked files present (use "git add" to track)

subas@DESKTOP-6OLA6SP MINGW64 /e/Self_Learn/Git (master)
$ git add index.html

subas@DESKTOP-6OLA6SP MINGW64 /e/Self_Learn/Git (master)
$ git add L1.PNG

subas@DESKTOP-6OLA6SP MINGW64 /e/Self_Learn/Git (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   L1.PNG
        new file:   index.html


subas@DESKTOP-6OLA6SP MINGW64 /e/Self_Learn/Git (master)
$ |
```

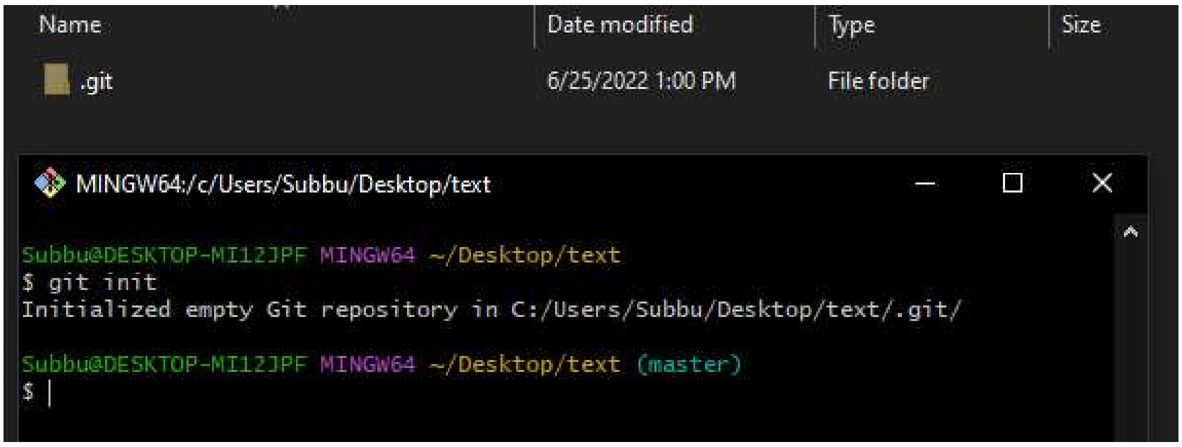
INIT

The git init command is the first command that you will run on Git. The git init command is used to create a new blank repository. It is used to make an existing project as a Git project. Several Git commands run inside the repository, but init commands can be run outside of the repository.
The git init command creates a .git subdirectory in the current working directory. This newly created subdirectory contains all of the necessary metadata. These metadata can be categorized into objects, refs, and temp files. It also initializes a HEAD pointer for the master branch of the repository.

Command

$ git init

Practically



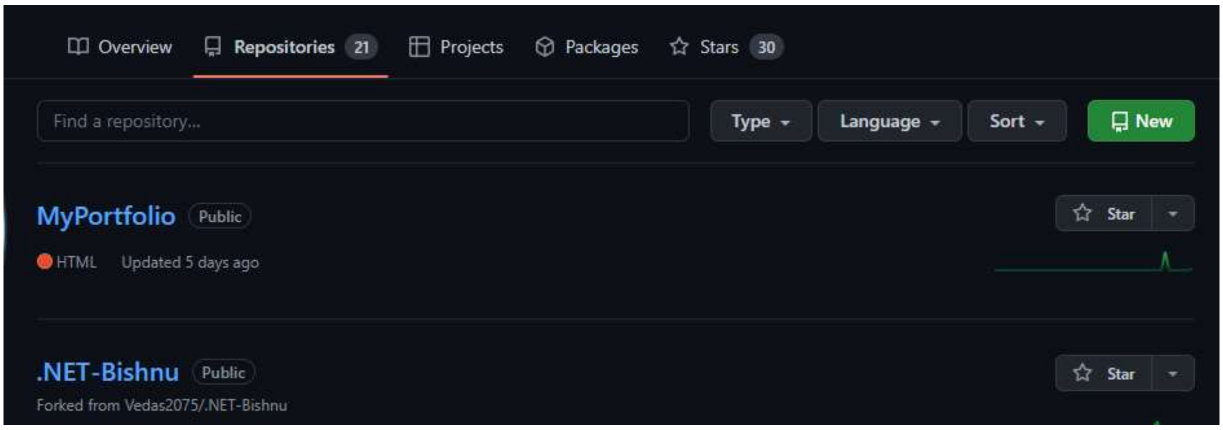
Name	Date modified	Type	Size
.git	6/25/2022 1:00 PM	File folder	

MINGW64:/c/Users/Subbu/Desktop/text

```
Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text
$ git init
Initialized empty Git repository in C:/Users/Subbu/Desktop/text/.git/

Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ |
```

PULL

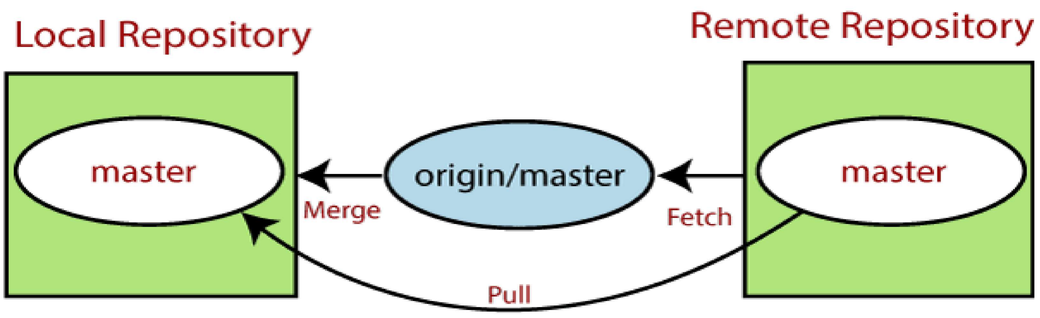
The pull command is used to access the changes (commits)from a remote repository to the local repository. It updates the local branches with the remote-tracking branches.



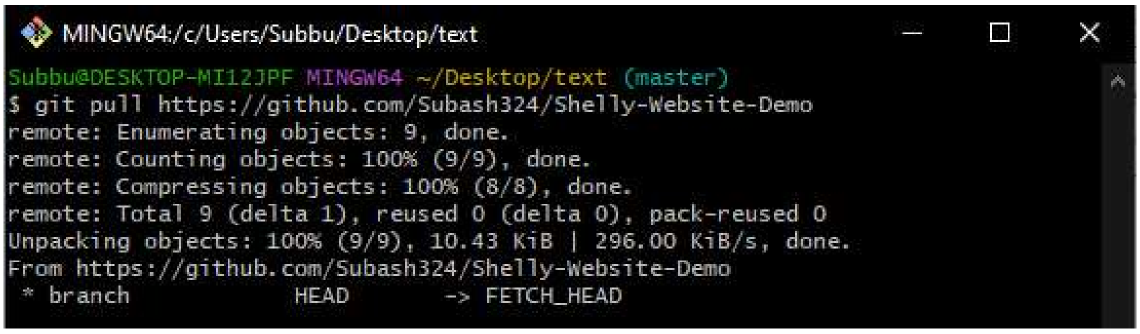
Remote tracking branches are branches that have been set up to push and pull from the remote repository. Generally, it is a collection of the fetch and merge commands. First, it fetches the changes from remote and combines them with the local repository.

Command

$ git pull <option> [<repository URL><refspec>...]



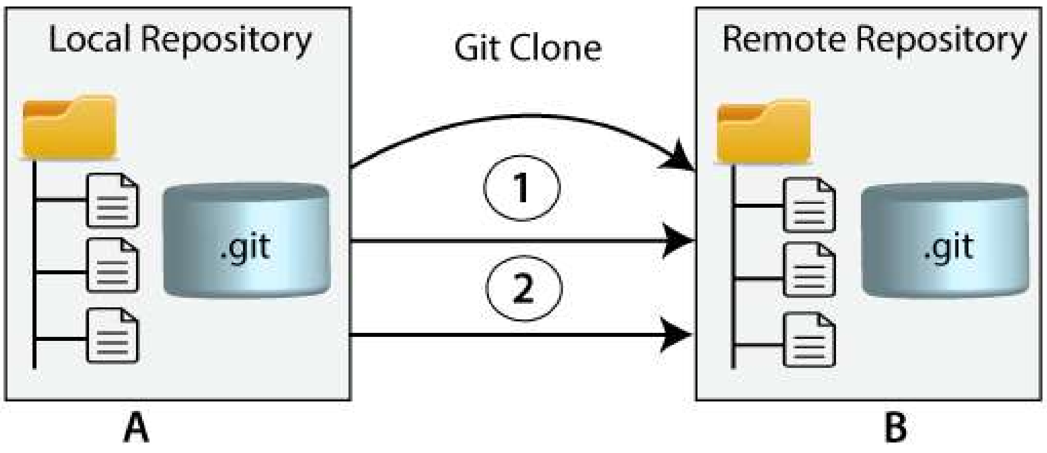
Practically



```
MINGW64:/c/Users/Subbu/Desktop/text
Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git pull https://github.com/Subash324/Shelly-Website-Demo
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), 10.43 KiB | 296.00 KiB/s, done.
From https://github.com/Subash324/Shelly-Website-Demo
 * branch            HEAD       -> FETCH_HEAD
```

CLONE

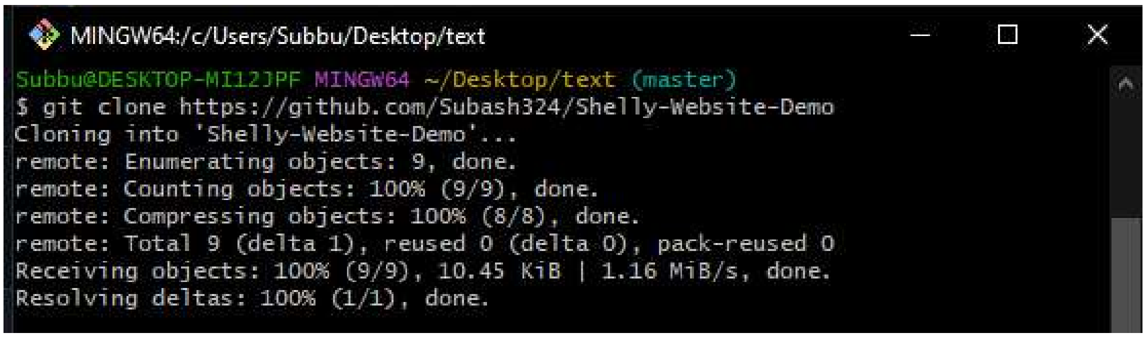
In Git, cloning is the act of making a copy of any target repository. The target repository can be remote or local. You can clone your repository from the remote repository to create a local copy on your system. Also, you can sync between the two locations.
The git clone is a command-line utility which is used to make a local copy of a remote repository. It accesses the repository through a remote URL.
Usually, the original repository is located on a remote server, often from a Git service like GitHub, Bitbucket, or GitLab. The remote repository URL is referred to the origin.



Command

$ git clone <repository URL>

Practically



```
Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git clone https://github.com/Subash324/Shelly-Website-Demo
Cloning into 'Shelly-Website-Demo'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (9/9), 10.45 KiB | 1.16 MiB/s, done.
Resolving deltas: 100% (1/1), done.
```

ADD

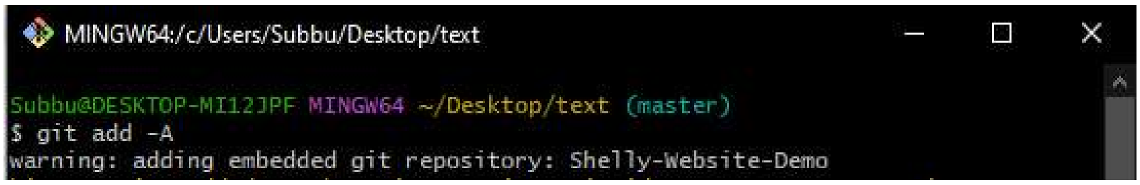
The git add command is used to add file contents to the Index (Staging Area).This command updates the current content of the working tree to the staging area. It also prepares the staged content for the next commit. Every time we add or update any file in our project, it is required to forward updates to the staging area.
The git add command is a core part of Git technology. It typically adds one file at a time, but there are some options available that can add more than one file at once.Git add

command is a straight forward command. It adds files to the staging area. We can add single or multiple files at once in the staging area.

Command

$ git add <File name>

Practically



```
MINGW64:/c/Users/Subbu/Desktop/text

Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git add -A
warning: adding embedded git repository: Shelly-Website-Demo
```

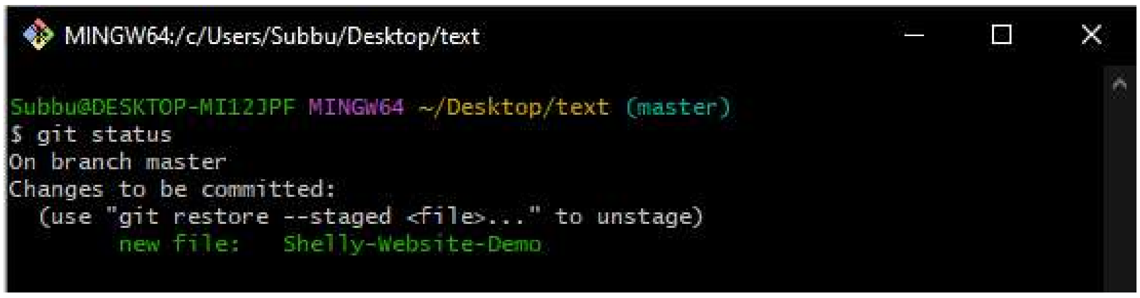
COMMIT

It is used to record the changes in the repository. It is the next command after the git add. Every commit contains the index data and the commit message. Every commit forms a parent-child relationship. When we add a file in Git, it will take place in the staging area. A commit command is used to fetch updates from the staging area to the repository.The staging and committing are co-related to each other. Staging allows us to continue making changes to the repository, and when we want to share these changes to the version control system, committing allows us to record these changes.
The commit command will commit the changes and generate a commit-id. The commit command without any argument will open the default text editor and ask for the commit message. We can specify our commit message in this text editor.

Command

$ git commit

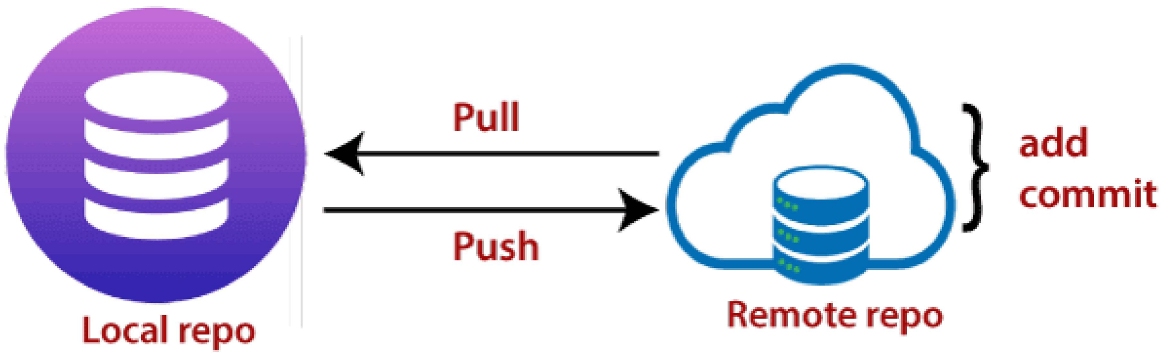
Practically



```
MINGW64:/c/Users/Subbu/Desktop/text

Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Shelly-Website-Demo
```

PUSH

The push term refers to uploading local repository content to a remote repository. Pushing is an act of transfer commits from your local repository to a remote repository. Pushing is capable of overwriting changes; caution should be taken when pushing.
The "git push" command is used to push into the repository. The push command can be considered as a tool to transfer commits between local and remote repositories.

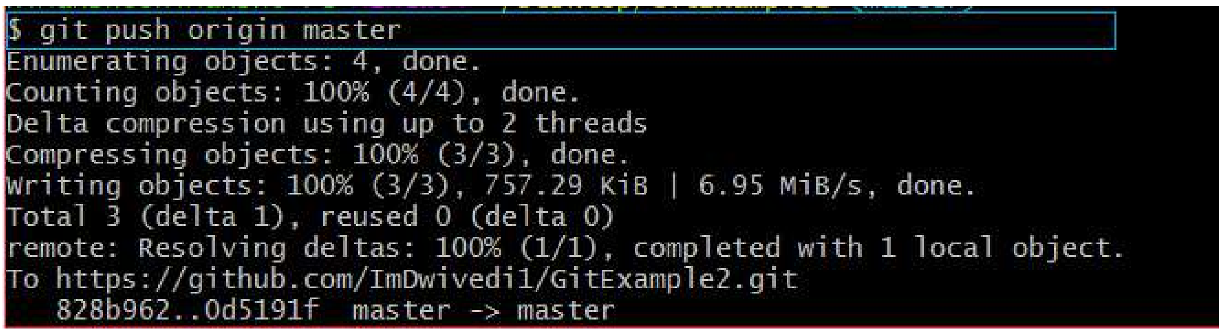


Git push origin master is a special command-line utility that specifies the remote branch and directory. When you have multiple branches and directory, then this command assists you in determining your main branch and repository.
Generally, the term origin stands for the remote repository, and master is considered as the main branch. So, the entire statement "git push origin master" pushed the local content on the master branch of the remote location.

Command

$ git push origin master

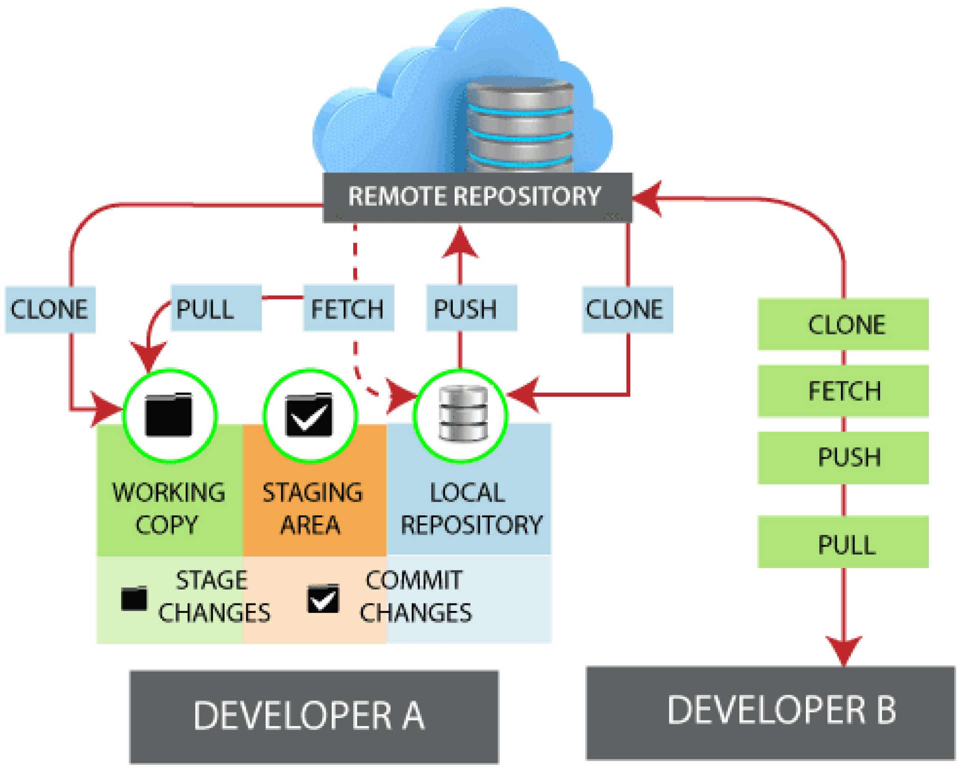
Practically



```
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 757.29 KiB | 6.95 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ImDwivedi1/GitExample2.git
   828b962..0d5191f  master -> master
```

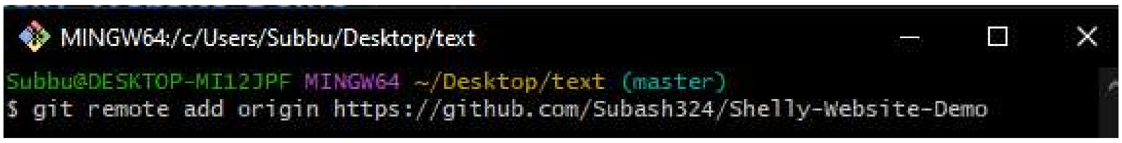
REMOTE

In Git, the term remote is concerned with the remote repository. It is a shared repository that all team members use to exchange their changes. A remote repository is stored on a code hosting service like an internal server, GitHub, Subversion, and more. In the case of a local repository, a remote typically does not provide a file tree of the project's current state; as an alternative, it only consists of the .git versioning data.The developers can perform many operations with the remote server. These operations can be a clone, fetch, push, pull, and more.



Git add remote

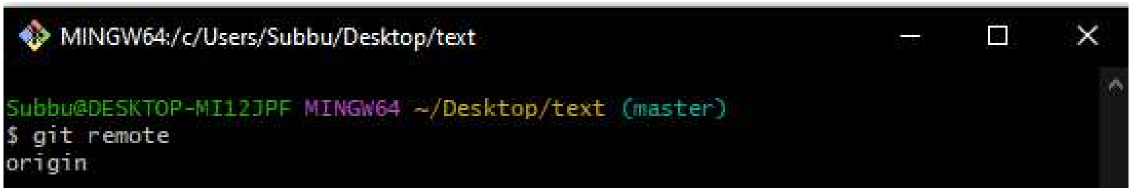
When we fetch a repository implicitly, git adds a remote for the repository. Also, we can explicitly add a remote for a repository. We can add a remote as a short nickname or short name.
It can be used as:
$ git remote add <short name><remote URL>



```
Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git remote add origin https://github.com/Subash324/Shelly-Website-Demo
```

Check Remote

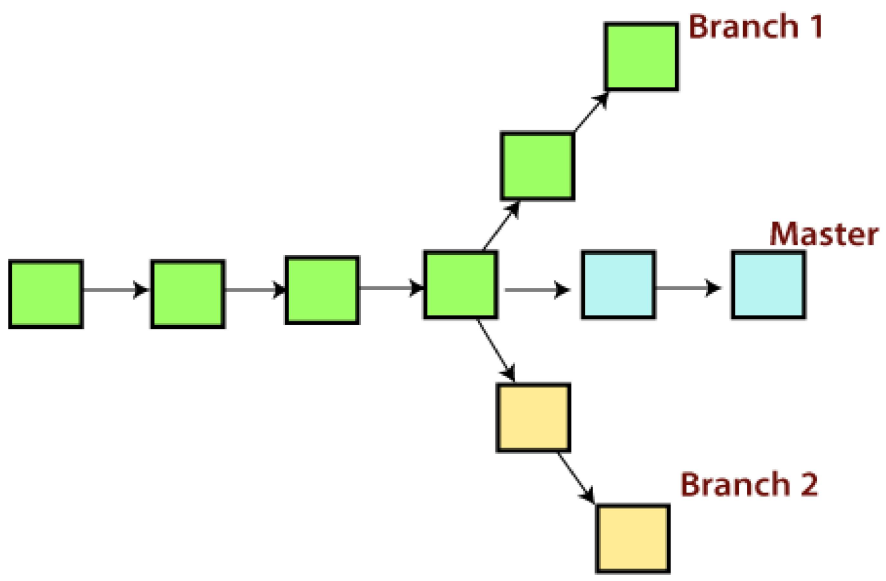
To check the configuration of the remote server, run the git remote command. The git remote command allows accessing the connection between remote and local. If you want to see the original existence of your cloned repository, use the git remote command.
It can be used as:
$ git remote



```
Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git remote
origin
```

GIT BRANCH

A branch is a version of the repository that diverges from the main working project. It is a feature available in most modern version control systems. A Git project can have more than one branch. These branches are a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug, you spawn a new branch to summarize your changes. So, it is complex to merge the unstable code with the main code base and also facilitates you to clean up your future history before merging with the main branch.



Git Master Branch

The master branch is a default branch in Git. It is instantiated when the first commit is made on the project. When you make the first commit, you're given a master branch to the starting commit point. When you start making a commit, then the master branch pointer automatically moves forward. A repository can have only one master branch.

Master branch is the branch in which all the changes eventually get merged back. It can be called an official working version of your project.

Operations on Branches

We can perform various operations on Git branches. The git branch command allows you to create, list, rename and delete branches. Many operations on branches are applied by git checkout and git merge command. So, the git branch is tightly integrated with the git checkout and git merge commands.

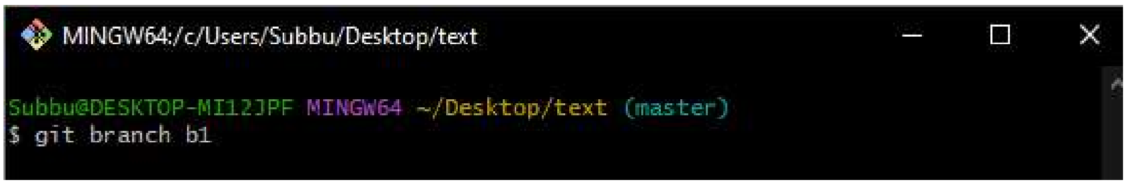
The Operations that can be performed on a branch:

Create Branch

We can create a new branch with the help of the git branch command. This command will be used as:

Syntax:

$ git branch <branch name>



MINGW64:/c/Users/Subbu/Desktop/text

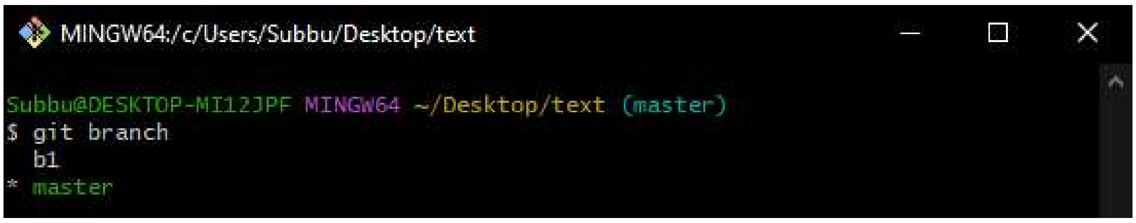
```
Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git branch b1
```

List Branch

We can List all of the available branches in your repository by using the following command.

Either we can use git branch - list or git branch command to list the available branches in the repository.

Syntax:



MINGW64:/c/Users/Subbu/Desktop/text

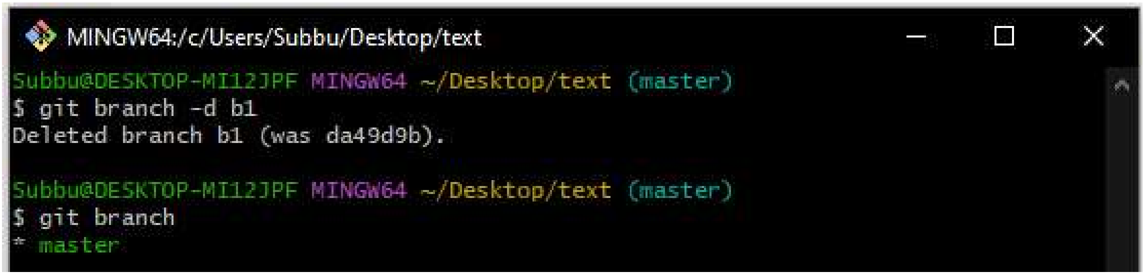
```
Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git branch
  b1
* master
```

Delete Branch

You can delete the specified branch. It is a safe operation. In this command, Git prevents you from deleting the branch if it has unmerged changes. Below is the command to do this.

Syntax:

$ git branch -d<branch name>



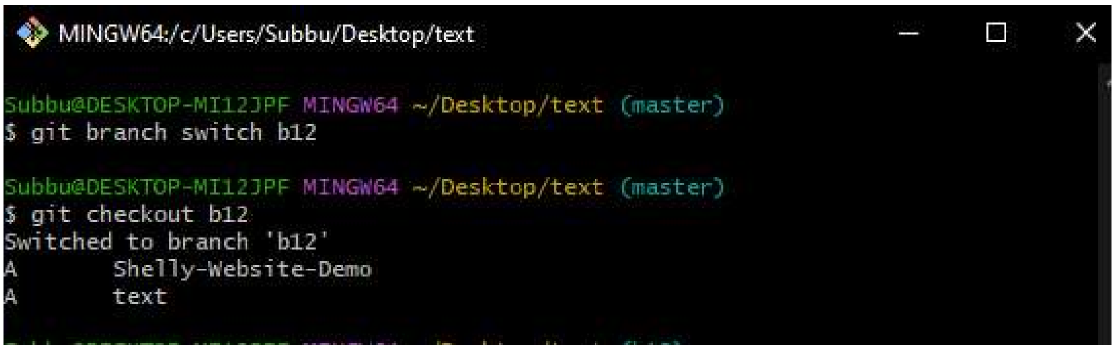
```
Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git branch -d b1
Deleted branch b1 (was da49d9b).

Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git branch
* master
```

Switch Branch

Git allows you to switch between the branches without making a commit. You can switch between two branches with the git checkout command.

Syntax:

git checkout<branch name>



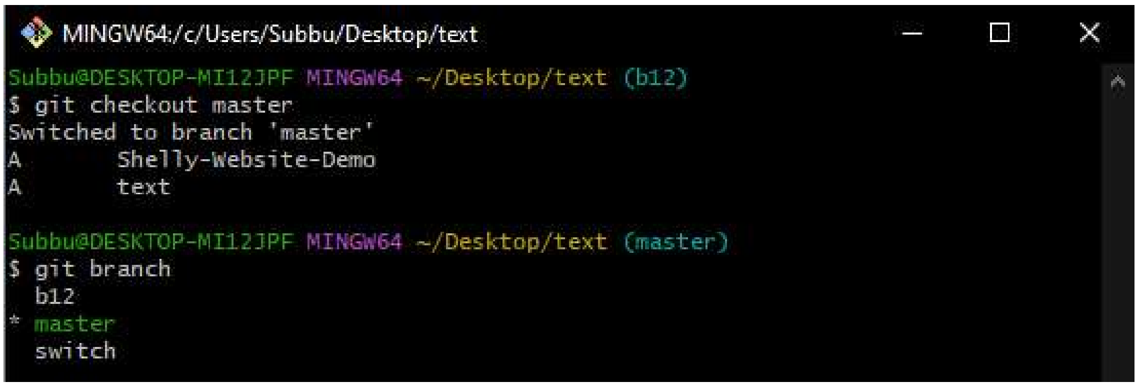
```
Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git branch switch b12

Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git checkout b12
Switched to branch 'b12'
A       Shelly-Website-Demo
A       text
```

Switch to master Branch

We can switch to the master branch from any other branch with the help of below command.

Syntax:

$ git branch -m master



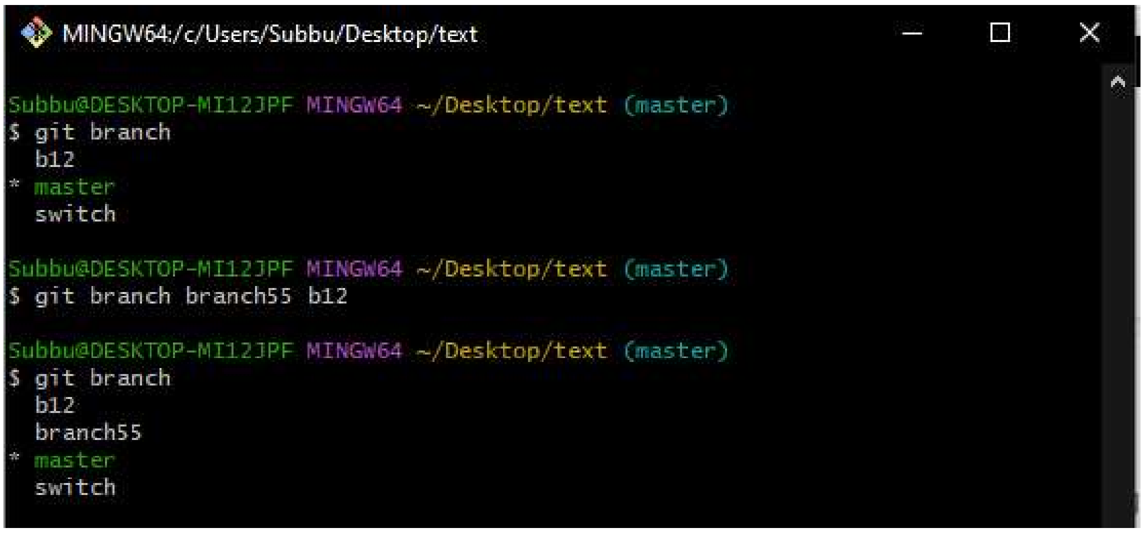
```
MINGW64:/c/Users/Subbu/Desktop/text
Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (b12)
$ git checkout master
Switched to branch 'master'
A       Shelly-Website-Demo
A       text

Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git branch
  b12
* master
  switch
```

Rename Branch

We can rename the branch with the help of the git branch command. To rename a branch, use the below command:

Syntax:

$ git branch -m <old branch name><new branch name>



```
MINGW64:/c/Users/Subbu/Desktop/text

Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git branch
  b12
* master
  switch

Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git branch branch55 b12

Subbu@DESKTOP-MI12JPF MINGW64 ~/Desktop/text (master)
$ git branch
  b12
  branch55
* master
  switch
```

Git Cheat Sheet

1. Git configuration

- Git config

  Get and set configuration variables that control all facets of how Git looks and operates.

  Set the name:

  $ git config --global user.name "User name"

  Set the email:

  $ git config --global user.email "himanshudubey481@gmail.com"

  Set the default editor:

  $ git config --global core.editor Vim

  Check the setting:

  $ git config -list

- Git alias

  Set up an alias for each command:

  $ git config --global alias.co checkout

  $ git config --global alias.br branch

  $ git config --global alias.ci commit

  $ git config --global alias.st status

2. Starting a project

- Git init

  Create a local repository:

  $ git init

- Git clone

  Make a local copy of the server repository.

  $ git clone

3. Local changes

- Git add

  Add a file to staging (Index) area:

  $ git add Filename

  Add all files of a repo to staging (Index) area:

  $ git add*

- Git commit

  Record or snapshots the file permanently in the version history with a message.

  $ git commit -m " Commit Message"

4. Track changes

- Git diff

  Track the changes that have not been staged: $ git diff

  Track the changes that have staged but not committed:

  $ git diff --staged

  Track the changes after committing a file:

  $ git diff HEAD

  Track the changes between two commits:

  $ git diff Git Diff Branches:

  $ git diff < branch 2>

- Git status

  Display the state of the working directory and the staging area.

  $ git status

- Git show Shows objects:

  $ git show

5. Commit History

- Git log

  Display the most recent commits and the status of the head:

$ git log

Display the output as one commit per line:

$ git log -oneline

Displays the files that have been modified:

$ git log -stat

Display the modified files with location:

$ git log -p

- Git blame

  Display the modification on each line of a file:

  $ git blame <file name>

6. Branching

- Git branch Create branch:

  $ git branch List Branch:

  $ git branch --list Delete a Branch:

  $ git branch -d Delete a remote Branch:

  $ git push origin -delete Rename Branch:

  $ git branch -m

- Git checkout

  Switch between branches in a repository.

  Switch to a particular branch:

  $ git checkout

  Create a new branch and switch to it:

  $ git checkout -b Checkout a Remote branch:

  $ git checkout

7. Merging

- Git merge

  Merge the branches:

  $ git merge

  Merge the specified commit to currently active branch:

  $ git merge

- Git rebase

  Apply a sequence of commits from distinct branches into a final commit.

  $ git rebase

  Continue the rebasing process:

  $ git rebase -continue Abort the rebasing process:

  $ git rebase --skip

- Git interactive rebase

  Allow various operations like edit, rewrite, reorder, and more on existing commits.

  $ git rebase -i

8. Remote

- Git remote

  Check the configuration of the remote server:

  $ git remote -v

  Add a remote for the repository:

  $ git remote add Fetch the data from the remote server:

  $ git fetch

  Remove a remote connection from the repository:

  $ git remote rm

  Rename remote server:

  $ git remote rename

  Show additional information about a particular remote:

  $ git remote show

Change remote:

$ git remote set-url

- Git origin master

  Push data to the remote server:

  $ git push origin master Pull data from remote server:

  $ git pull origin master

9. Pushing Updates

- Git push

  Transfer the commits from your local repository to a remote server. Push data to the remote server:

  $ git push origin master Force push data:

  $ git push -f

  Delete a remote branch by push command:

  $ git push origin -delete edited

10. Pulling updates

- Git pull

  Pull the data from the server:

  $ git pull origin master

  Pull a remote branch:

  $ git pull

- Git fetch

  Download branches and tags from one or more repositories. Fetch the remote repository:

  $ git fetch< repository Url> Fetch a specific branch:

  $ git fetch

  Fetch all the branches simultaneously:

$ git fetch -all

Synchronize the local repository:

$ git fetch origin

11. Undo changes

- Git revert

  Undo the changes:

  $ git revert

  Revert a particular commit:

  $ git revert

- Git reset

  Reset the changes:

  $ git reset -hard

  $ git reset -soft:

  $ git reset --mixed

12. Removing files

- Git rm

  Remove the files from the working tree and from the index:

  $ git rm <file Name>

  Remove files from the Git But keep the files in your local repository:

  $ git rm --cached

Git CheatSheet:
https://education.github.com/git-cheat-sheet-education.pdf