What is Git

Git is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently.

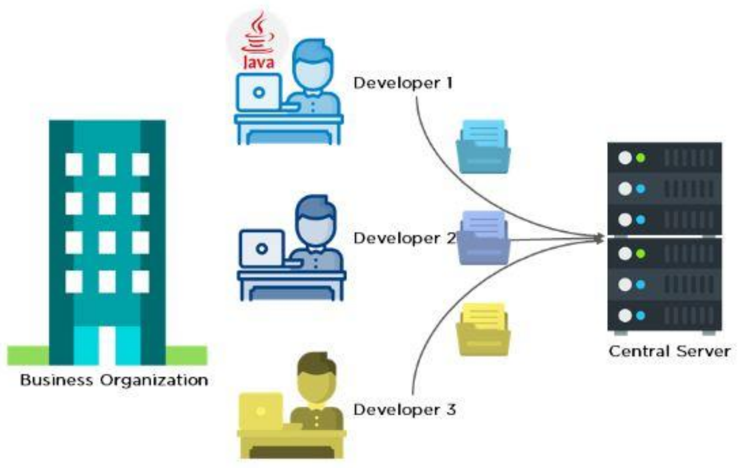
Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development.

Linus Torvalds created Git in 2005 for the development of the Linux kernel.

Before diving deep, let’s explain some scenario with and without git development:

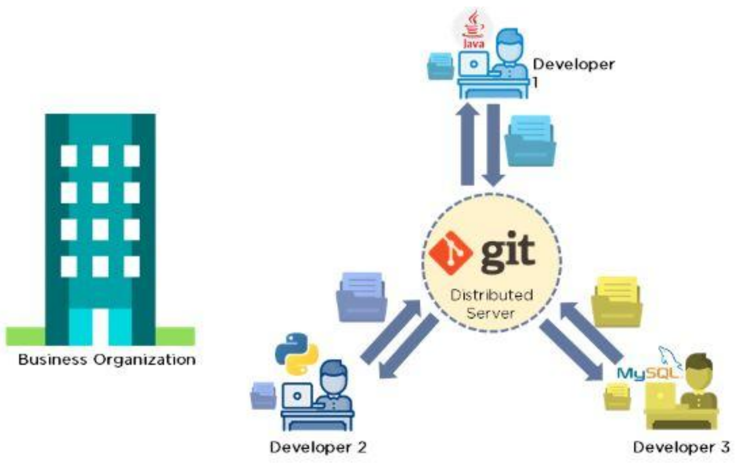
Without Git:

- Developers used to submit their codes to the central server without having copies of their own
- Any changes made to the source code were unknown to the other developers
- There was no communication between any of the developers

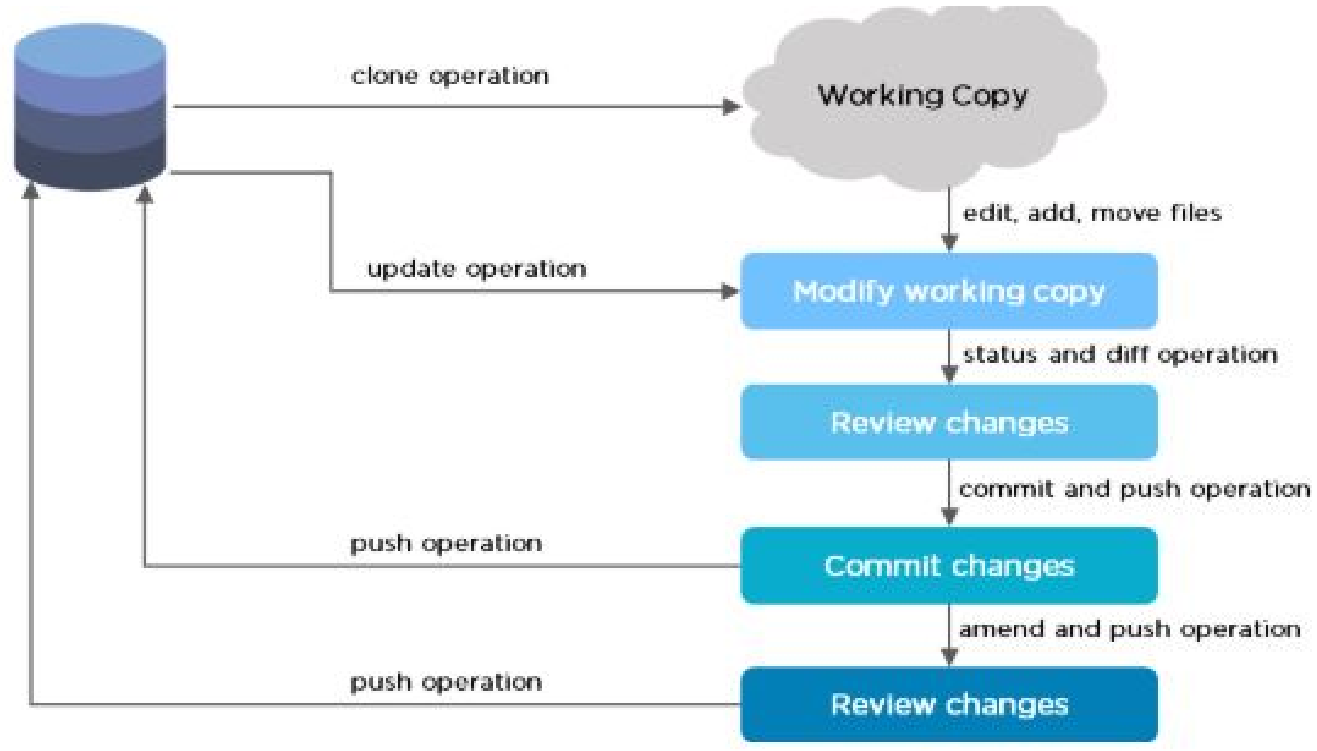


With Git:

- Every developer has an entire copy of the code on their local systems
- Any changes made to the source code can be tracked by others
- There is regular communication between the developers

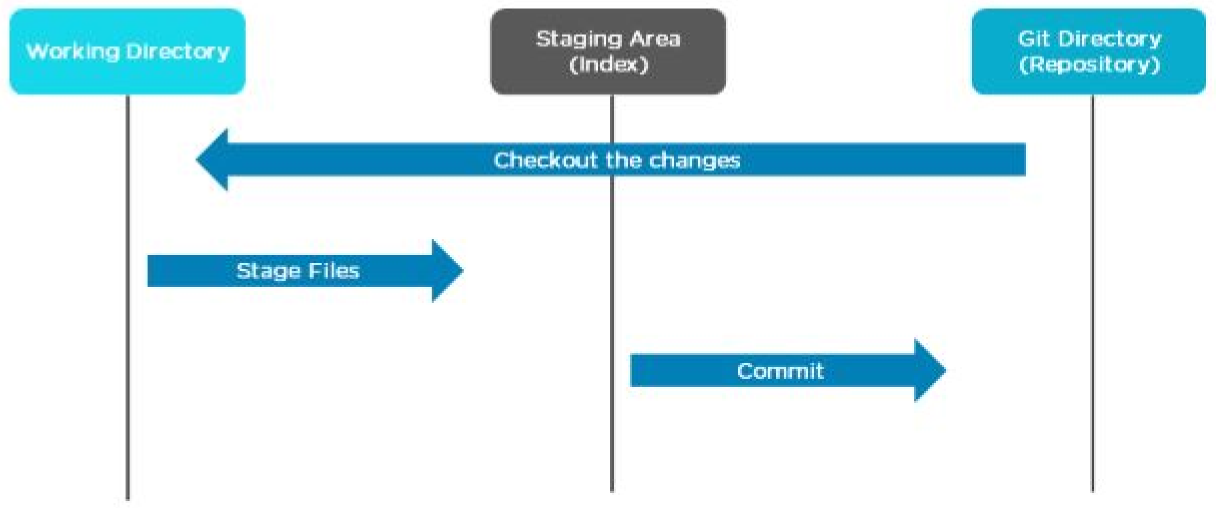


Git Workflow



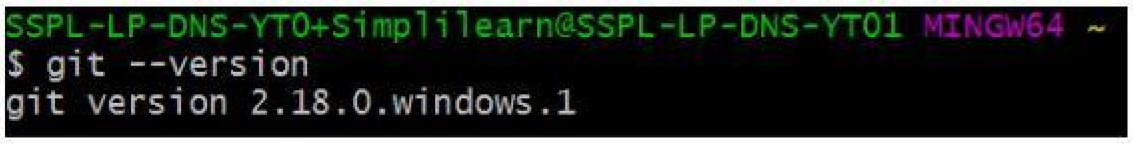
The Git workflow is divided into three states:

- **Working directory** - The working area, or working directory, consists of files that you are currently working on. In this area files are not handled by git. These files are also referred to as "untracked files."
- **Staging area (Index)** - Staging area is files that are going to be a part of the next commit, which lets git know what changes in the file are going to occur for the next commit.
- **Git directory (Repository)** - Perform a commit that stores the changes permanently to your Git directory.



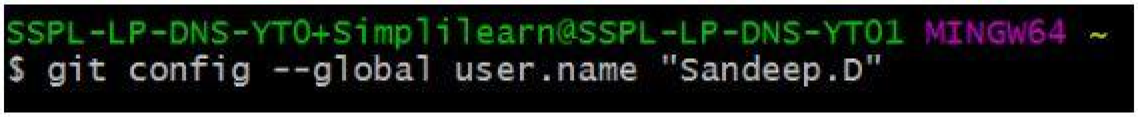
Command

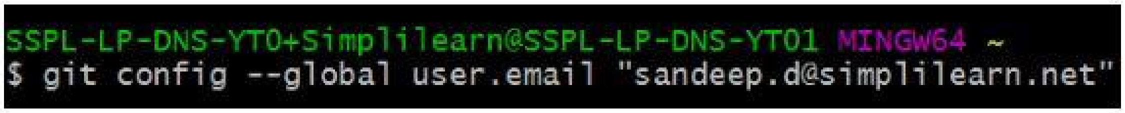
- Check the version of Git.

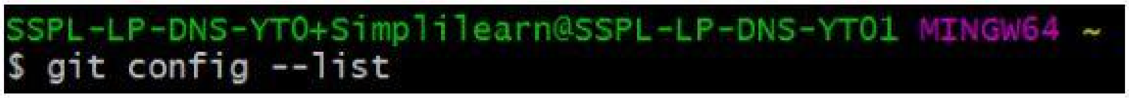


```
SSPL-LP-DNS-YTO+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ git --version
git version 2.18.0.windows.1
```

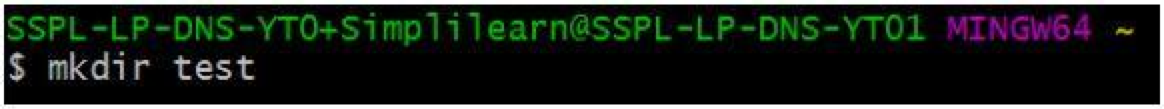
- Set up global config variables - If you are working with other developers, you need to know who is checking the code in and out, and to make the changes.


```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ git config --global user.name "Sandeep.D"
```

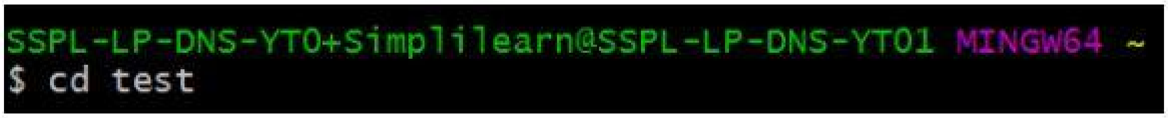

```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ git config --global user.email "sandeep.d@simplilearn.net"
```


```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ git config --list
```

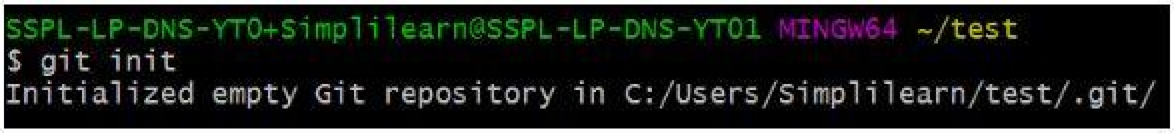
- Create a “test” repository in the local system.


```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ mkdir test
```

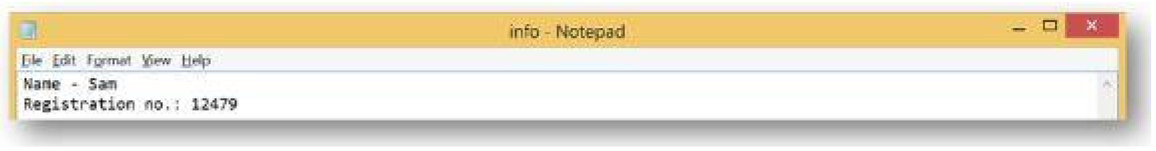
- Move to the test repository.


```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ cd test
```

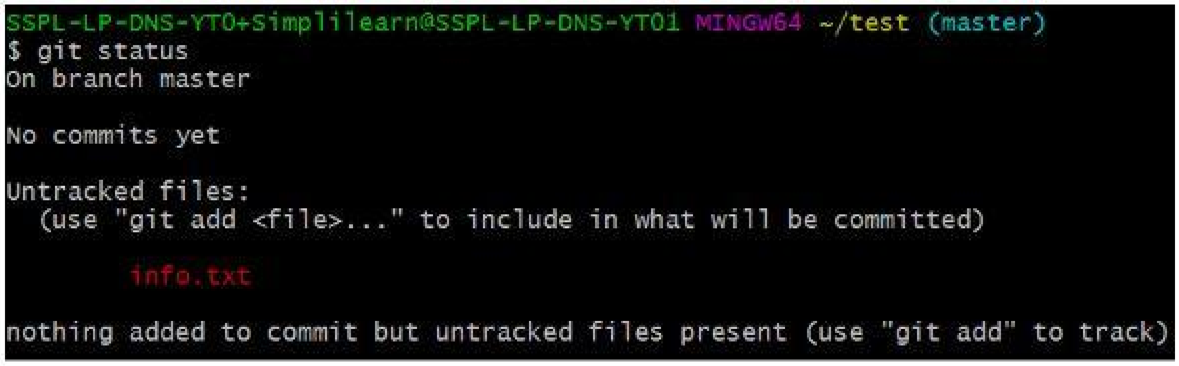
- Create a new git instance for a project.


```
SSPL-LP-DNS-YT0+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test
$ git init
Initialized empty Git repository in C:/Users/Simplilearn/test/.git/
```

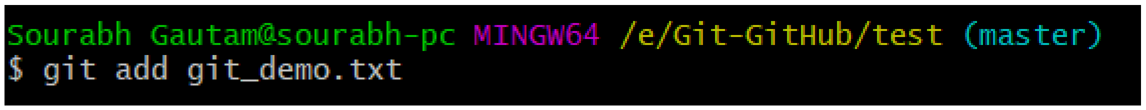
- Create a text file called info.txt in the test folder; write something and save it.


```
Name - Sam
Registration no.: 12479
```

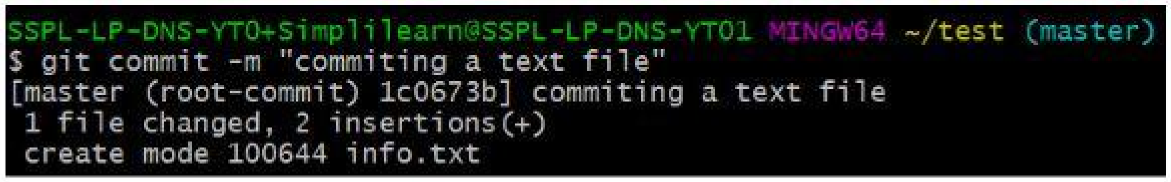
- Check the status of the repository.


```
SSPL-LP-DNS-YTO+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git status
On branch master

No commits yet

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

        info.txt

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

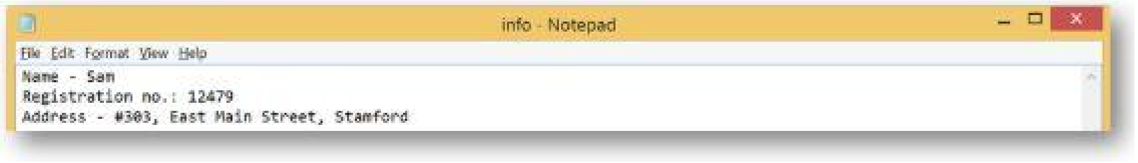
- Add the file you created to make a commit.


```
Sourabh Gautam@sourabh-pc MINGW64 /e/Git-GitHub/test (master)
$ git add git_demo.txt
```

- Commit those changes to the repository's history with a short message.

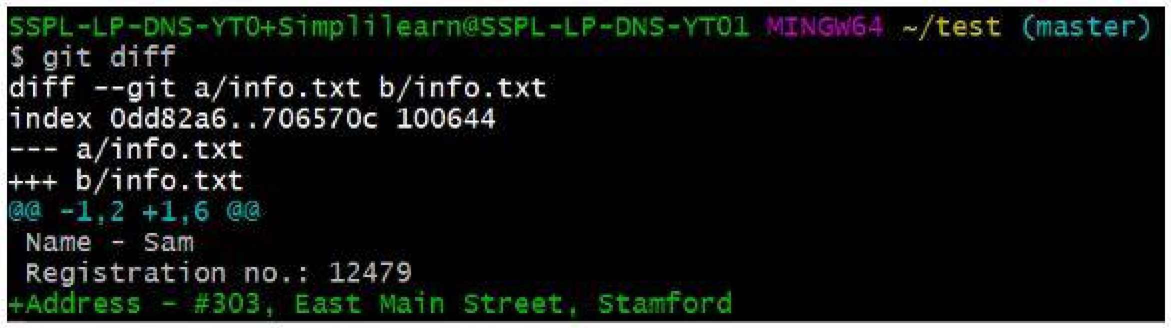

```
SSPL-LP-DNS-YTO+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git commit -m "commiting a text file"
[master (root-commit) 1c0673b] commiting a text file
 1 file changed, 2 insertions(+)
 create mode 100644 info.txt
```

- Make any necessary changes to the file and save.



```
Name - Sam
Registration no.: 12479
Address - #303, East Main Street, Stamford
```

- Now that you've made changes to the file, you can compare the differences since your last commit.



```
SSPL-LP-DNS-YTO+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~/test (master)
$ git diff
diff --git a/info.txt b/info.txt
index 0dd82a6..706570c 100644
--- a/info.txt
+++ b/info.txt
@@ -1,2 +1,6 @@
 Name - Sam
 Registration no.: 12479
+Address - #303, East Main Street, Stamford
```

Adding a local repository to GitHub using Git

1.) Create a new repository on GitHub.com. To avoid errors, do not initialize the new repository with README, license, or gitignore files. You can add these files after your project has been pushed to GitHub.
2.) Create New Repository drop-down
3.) Open Git Bash.
4.) Change the current working directory to your local project.
5.) Initialize the local directory as a Git repository.
$ git init -b main

6.) Add the files in your new local repository. This stages them for the first commit.
$ git add .

7.) Commit the files that you've staged in your local repository.
$ git commit -m "First commit"

8.) At the top of your repository on GitHub.com's Quick Setup page, click to copy the remote repository URL.
9.) Copy remote repository URL field

10.) In the Command prompt, add the URL for the remote repository where your local repository will be pushed.

```
$ git remote add origin  <REMOTE_URL>
$ git remote -v    #Verifies the new remote URL
```

# if origin is already exist go for remove the previous origin with following command

```
$ git remote rm origin
```

11.) Push the changes in your local repository to GitHub.com.

```
$ git push origin main
```

# make sure before push command all the changes has been committed else it will throw error.

Important questions

1.) How to move a file from staging area to working directory?

```
git checkout -- filename
```

2.) How to move a file from git repository to working directory?

```
git checkout commit_id
```

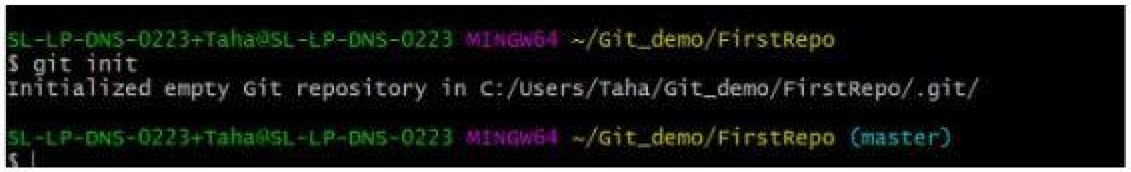
3.) How to add and commit in a single command?

```
git add . ; git commit -am "message"
```

Git Commands: Working With Local Repositories

git init

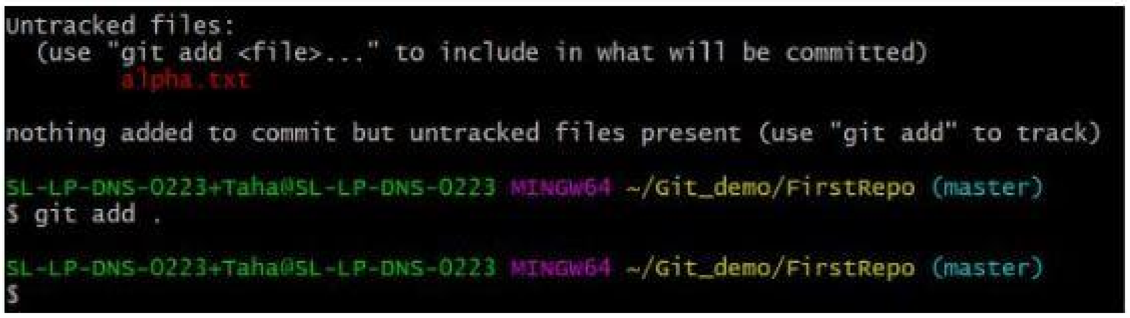
- The command git init is used to create an empty Git repository.
- After the git init command is used, a .git folder is created in the directory with some subdirectories. Once the repository is initialized, the process of creating other files begins.



```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo
$ git init
Initialized empty Git repository in C:/Users/Taha/Git_demo/FirstRepo/.git/

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ |
```

git add

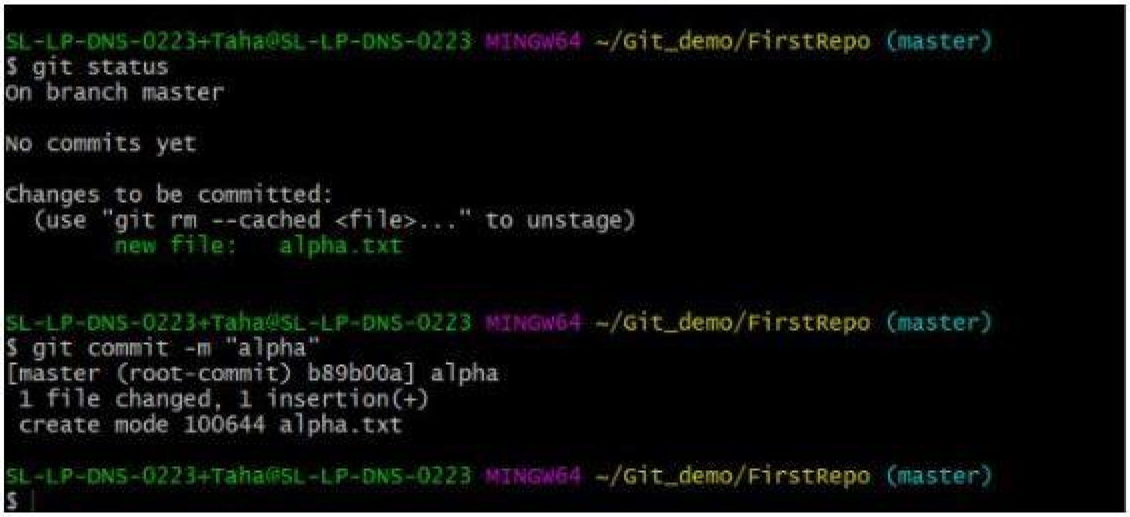
- Add command is used after checking the status of the files, to add those files to the staging area.
- Before running the commit command, "git add" is used to add any new or modified files.



```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        alpha.txt

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

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git add .

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$
```

git commit -m “message”

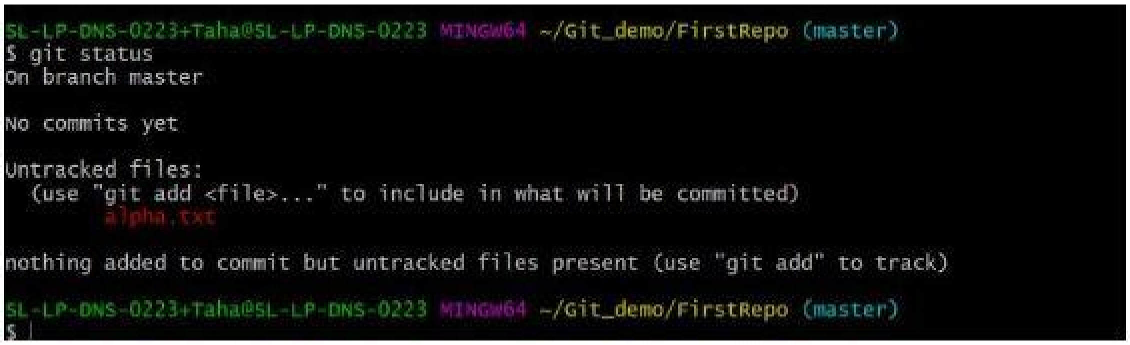
- The commit command makes sure that the changes are saved to the local repository.
- The command "git commit –m <message>" allows you to describe everyone and help them understand what has happened.



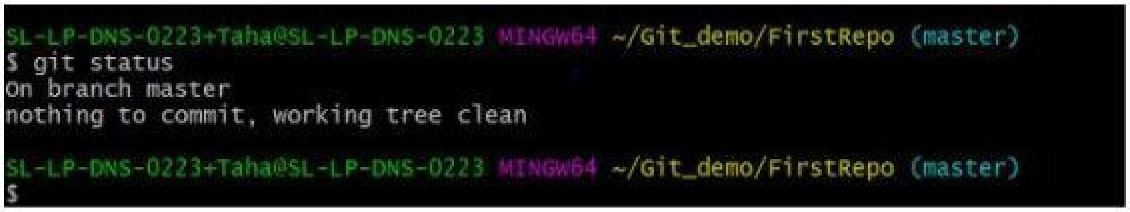
```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   alpha.txt


SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git commit -m "alpha"
[master (root-commit) b89b00a] alpha
 1 file changed, 1 insertion(+)
 create mode 100644 alpha.txt

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$
```

git status

- The git status command tells the current state of the repository.
- The command provides the current working branch. If the files are in the staging area, but not committed, it will be shown by the git status. Also, if there are no changes, it will show the message no changes to commit, working directory clean.



```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        alpha.txt

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

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$
```

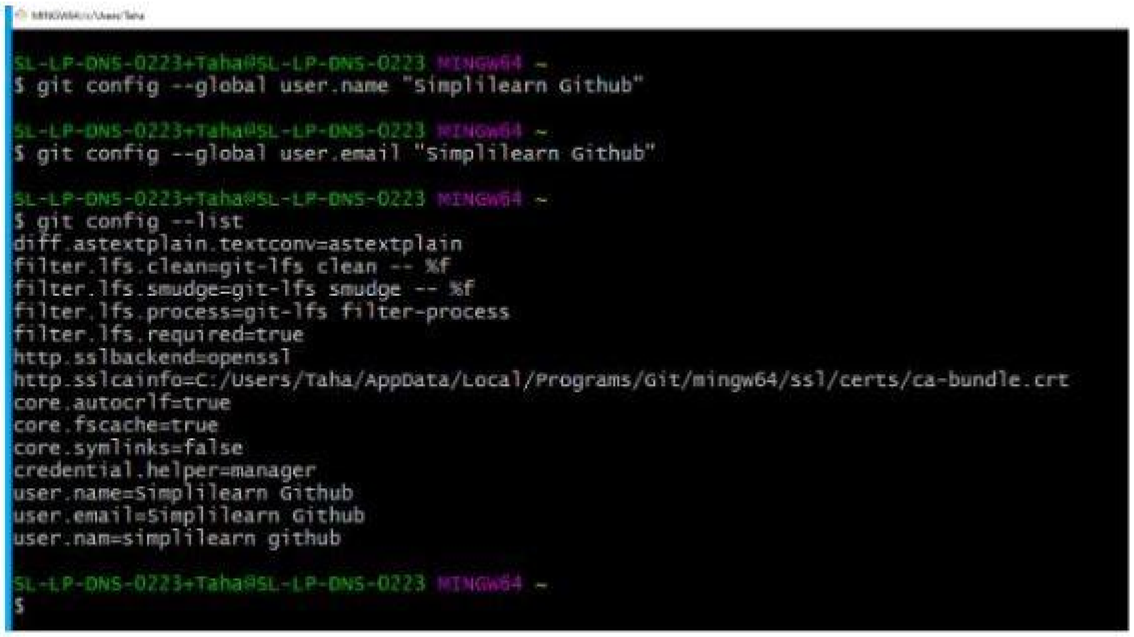


```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git status
On branch master
nothing to commit, working tree clean

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$
```

git config

- The git config command is used initially to configure the user.name and user.email. This specifies what email id and username will be used from a local repository.
- When git config is used with --global flag, it writes the settings to all repositories on the computer.

```
git config --global user.name "any user name"

git config --global user.email <email id>
```



```
MINGW64:/c/Users/Taha

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~
$ git config --global user.name "Simplilearn Github"

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~
$ git config --global user.email "Simplilearn Github"

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Users/Taha/AppData/Local/Programs/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
credential.helper=manager
user.name=Simplilearn Github
user.email=Simplilearn Github
user.nam=simplilearn github

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~
$
```

git branch

- The git branch command is used to determine what branch the local repository is on.
- The command enables adding and deleting a branch.

```
# Create a new branch

        git branch <branch_name>

# List all remote or local branches

        git branch –list

# Delete a branch

        git branch -d <branch_name>
```

git checkout

- The git checkout command is used to switch branches, whenever the work is to be started on a different branch.
- The command works on three separate entities: files, commits, and branches.

```
# Checkout an existing branch

        git checkout <branch_name>

#Checkout and create a new branch with that name

        git checkout -b <branch_name>
```

git merge

- The git merge command is used to integrate the branches together. The command combines the changes from one branch to another branch.

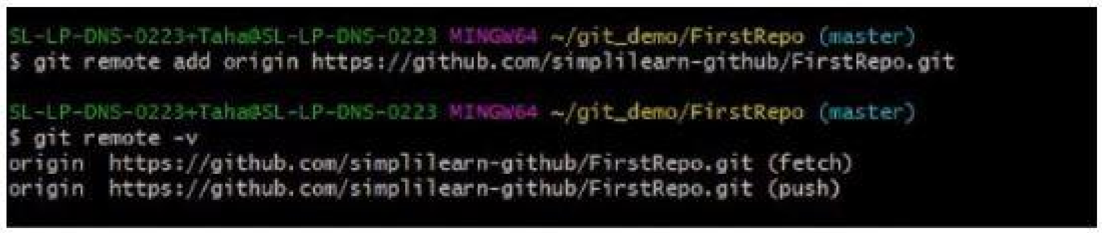
- It is used to merge the changes in the staging branch to the stable branch.

```
git merge <branch_name>
```

git remote

- The git remote command is used to create, view, and delete connections to other repositories.
- The connections here are not like direct links into other repositories, but as bookmarks that serve as convenient names to be used as a reference.

```
git remote add origin <address>
```



```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/git_demo/FirstRepo (master)
$ git remote add origin https://github.com/simplilearn-github/FirstRepo.git

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/git_demo/FirstRepo (master)
$ git remote -v
origin  https://github.com/simplilearn-github/FirstRepo.git (fetch)
origin  https://github.com/simplilearn-github/FirstRepo.git (push)
```

git clone

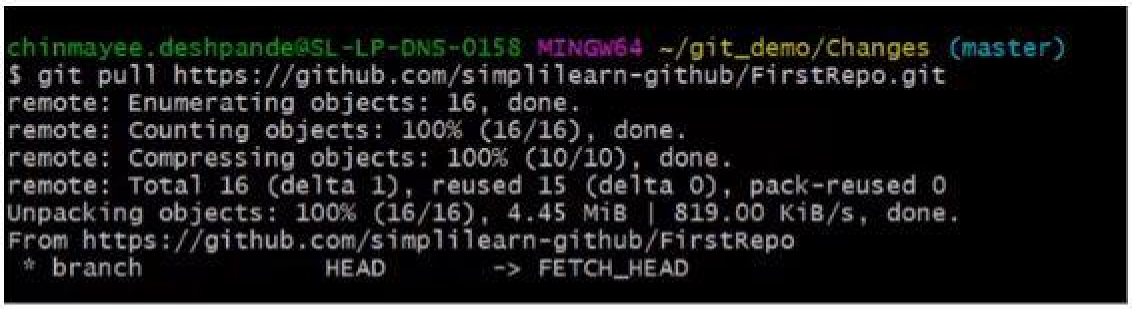
- The git clone command is used to create a local working copy of an existing remote repository.
- The command downloads the remote repository to the computer. It is equivalent to the Git init command when working with a remote repository.

```
git clone <remote_URL>
```

git pull

- The git pull command is used to fetch and merge changes from the remote repository to the local repository.
- The command "git pull origin master" copies all the files from the master branch of the remote repository to the local repository.

```
git pull <branch_name> <remote URL>
```

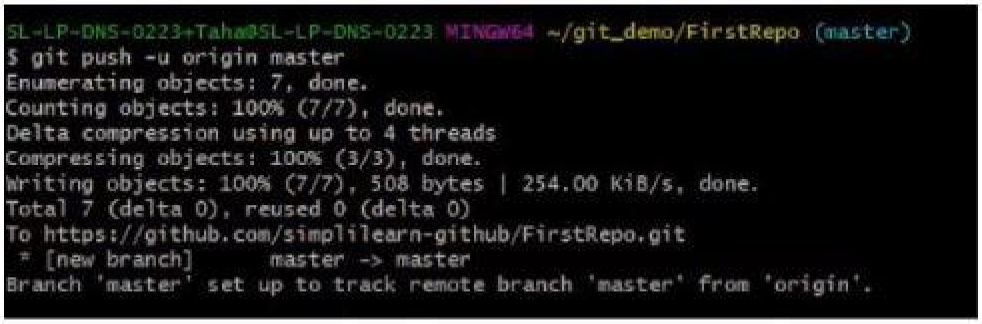


```
chinmayee.deshpande@SL-LP-DNS-0158 MINGW64 ~/git_demo/Changes (master)
$ git pull https://github.com/simplilearn-github/FirstRepo.git
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 16 (delta 1), reused 15 (delta 0), pack-reused 0
Unpacking objects: 100% (16/16), 4.45 MiB | 819.00 KiB/s, done.
From https://github.com/simplilearn-github/FirstRepo
 * branch            HEAD       -> FETCH_HEAD
```

git push

- The command git push is used to transfer the commits or pushing the content from the local repository to the remote repository.
- The command is used after a local repository has been modified, and the modifications are to be shared with the remote team members.

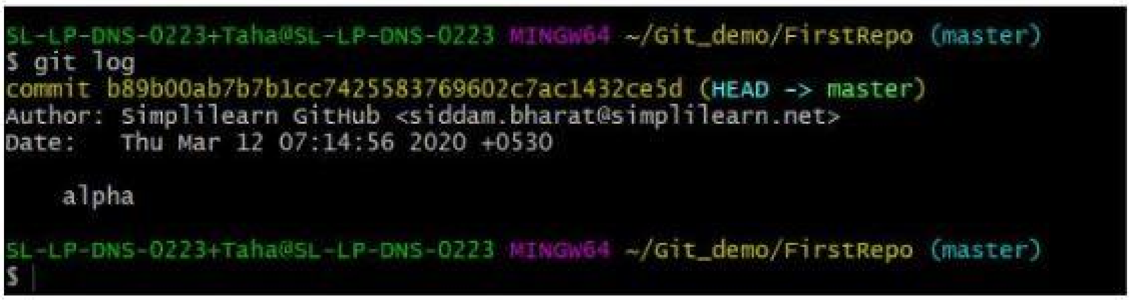
```
git push -u origin master
```



```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/git_demo/FirstRepo (master)
$ git push -u origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (7/7), 508 bytes | 254.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0)
To https://github.com/simplilearn-github/FirstRepo.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

git log

- The git log command shows the order of the commit history for a repository.
- The command helps in understanding the state of the current branch by showing the commits that lead to this state.



```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ git log
commit b89b00ab7b7b1cc7425583769602c7ac1432ce5d (HEAD -> master)
Author: Simplilearn GitHub <siddam.bharat@simplilearn.net>
Date:   Thu Mar 12 07:14:56 2020 +0530

    alpha

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/Git_demo/FirstRepo (master)
$ 
```