

Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **DCSE**

CHITKARA
UNIVERSITY



Submitted By:

Hardik Sharma

2110990547

G1

Submitted To:

Monit Kapoor

AIM 1: SETTING UP OF GIT CLIENT.**THEORY:****What is GIT?**

Git is a free and open-source version control system used to handle small to very large projects efficiently.

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

How To Install GIT On Windows**PROCEDURE:**

STEP1: We can install Git on Windows, using the most official build which is available for download on the GIT's official website or by just typing (scmgit) on any search engine.

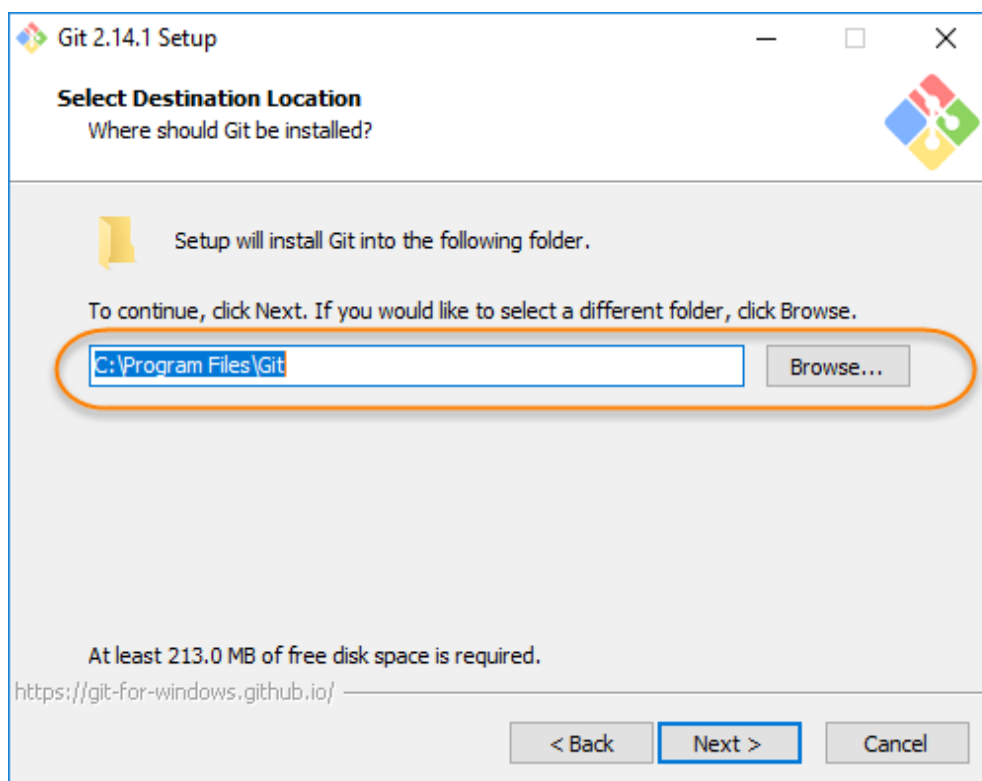


STEP2: Click on download for windows. Go to the folder where new downloads get store. Double click on the installer. The installer gets save on the machine as per the Windows OS configuration.

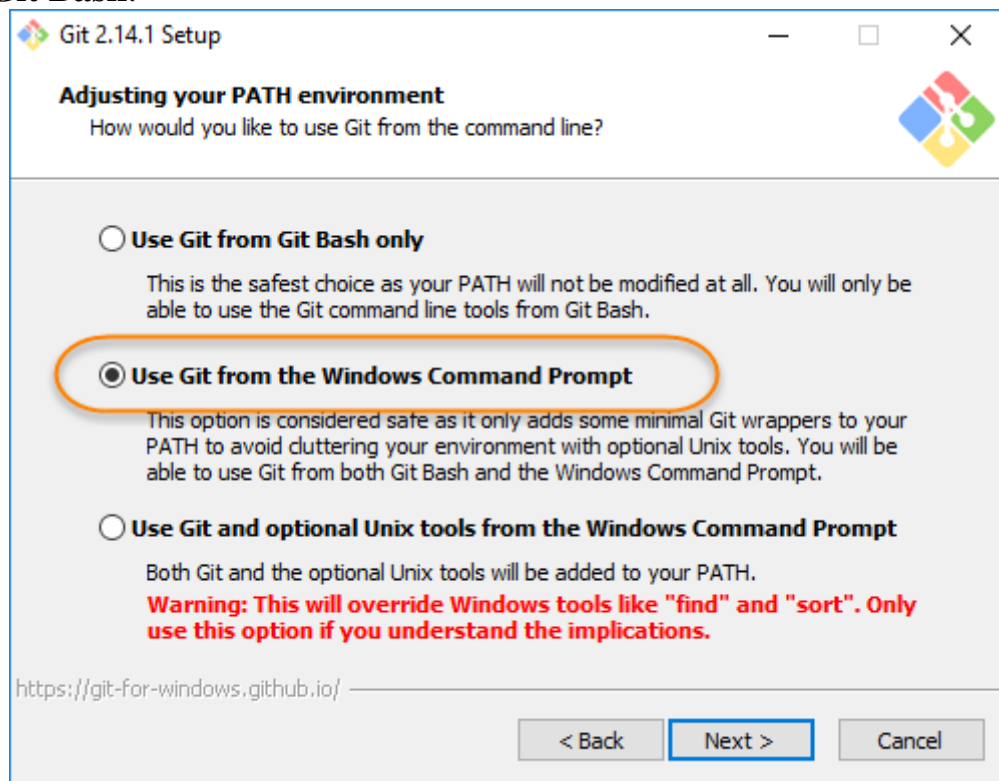


STEP 3: When you've successfully started the installer, you should see the Git Setup wizard screen. Follow the Next and Finish prompts to complete the installation.

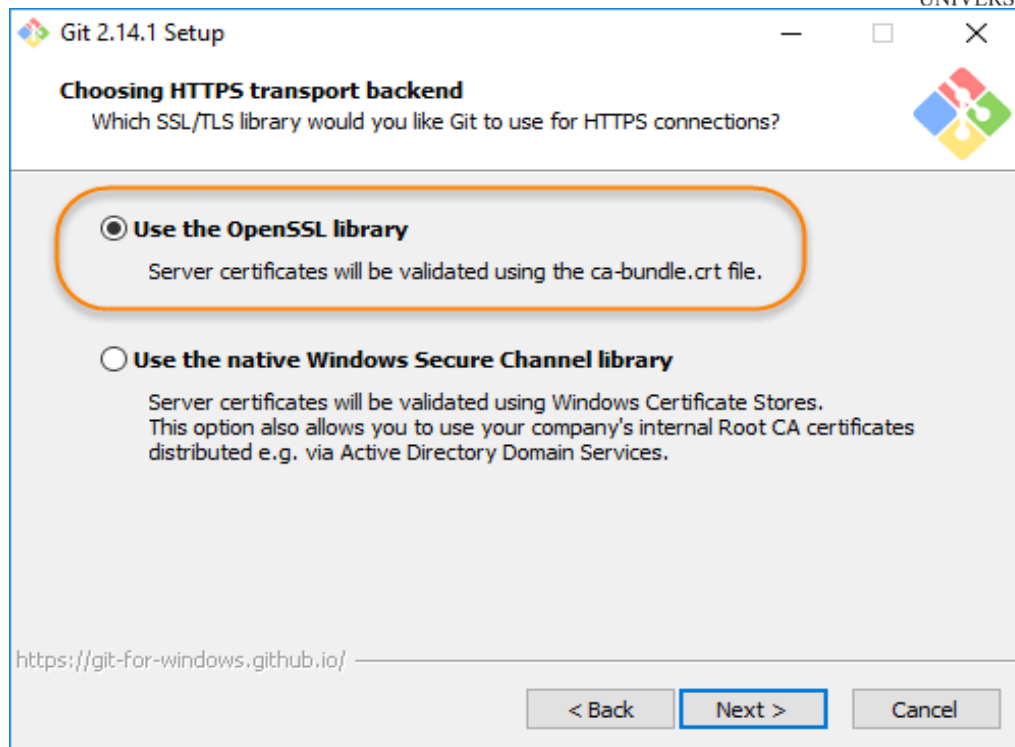
STEP 4: You may like to keep the installation to another folder, so you can set the path of destination location using browse option or either let it be as shown and continue.



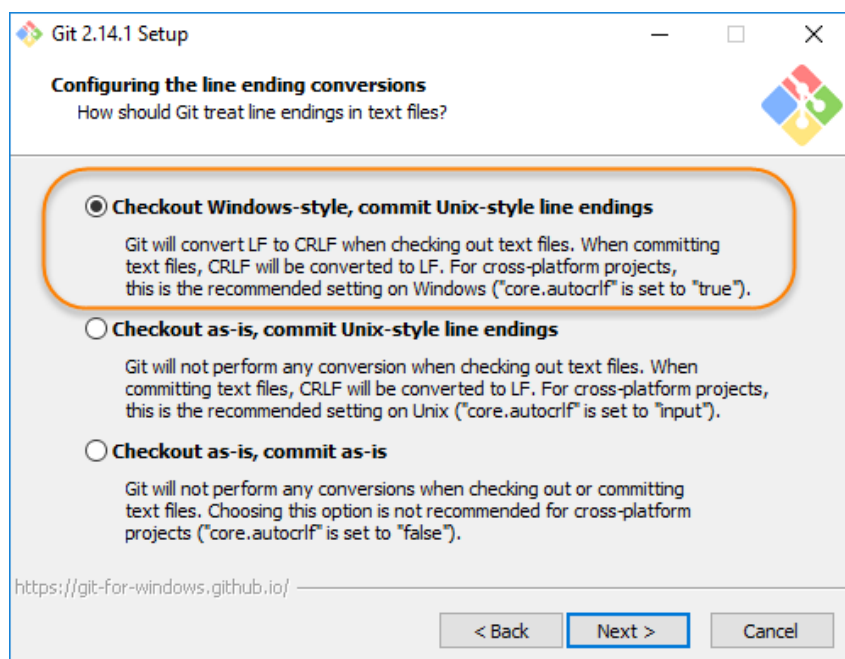
STEP 5: This is asking your choice that whether you like to Git from the **Windows Command Prompt** or you like to use some other program like **Git Bash**.



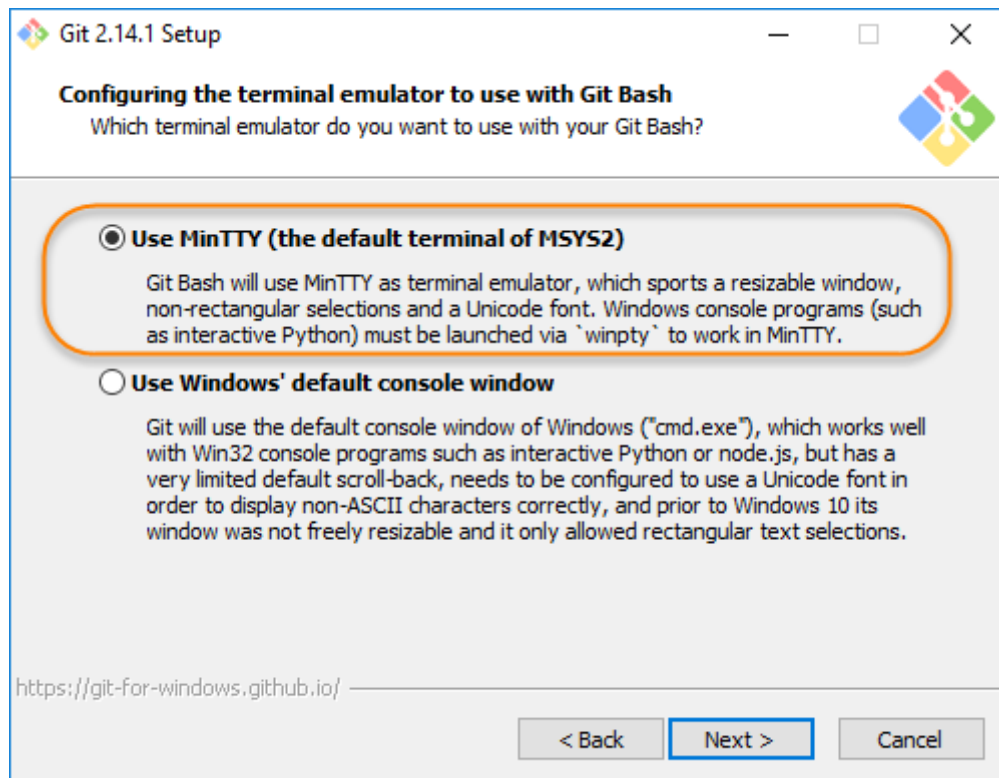
STEP 6: Select use the openSSL library



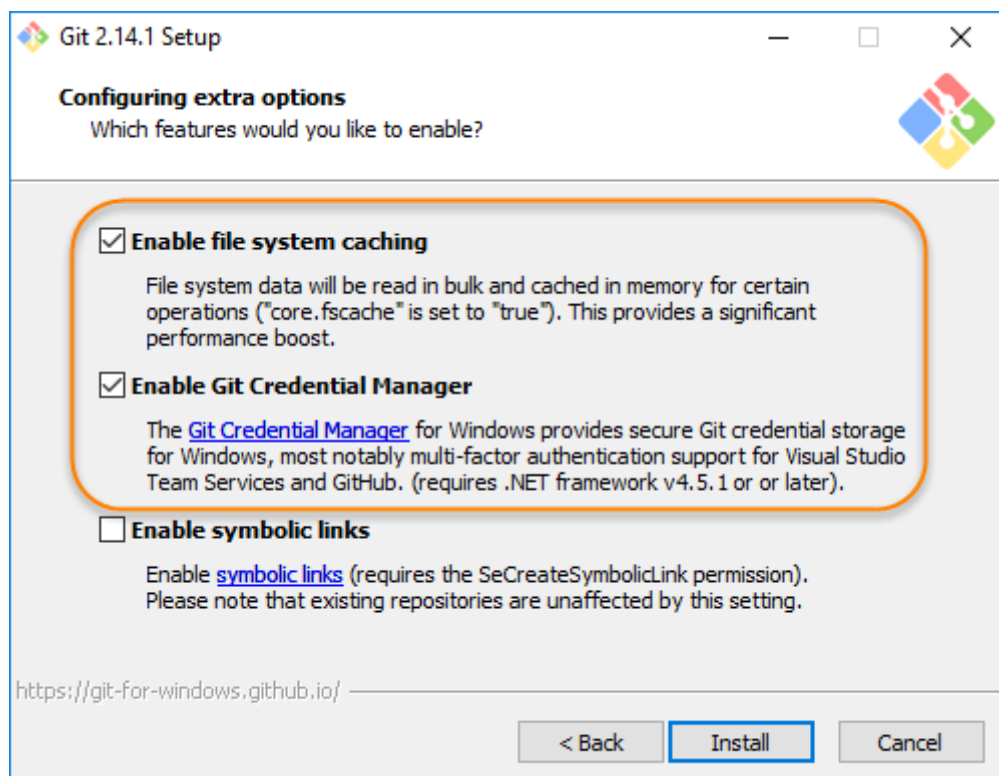
STEP 7: select the option and click on next.



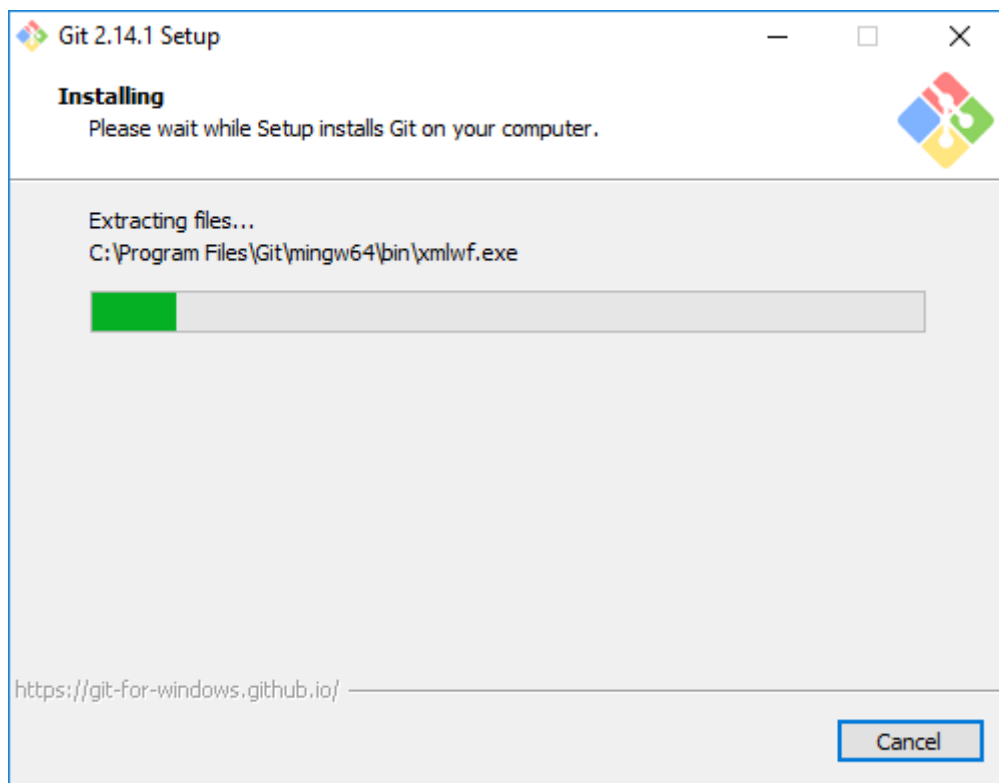
STEP 8: select the option and click on next.



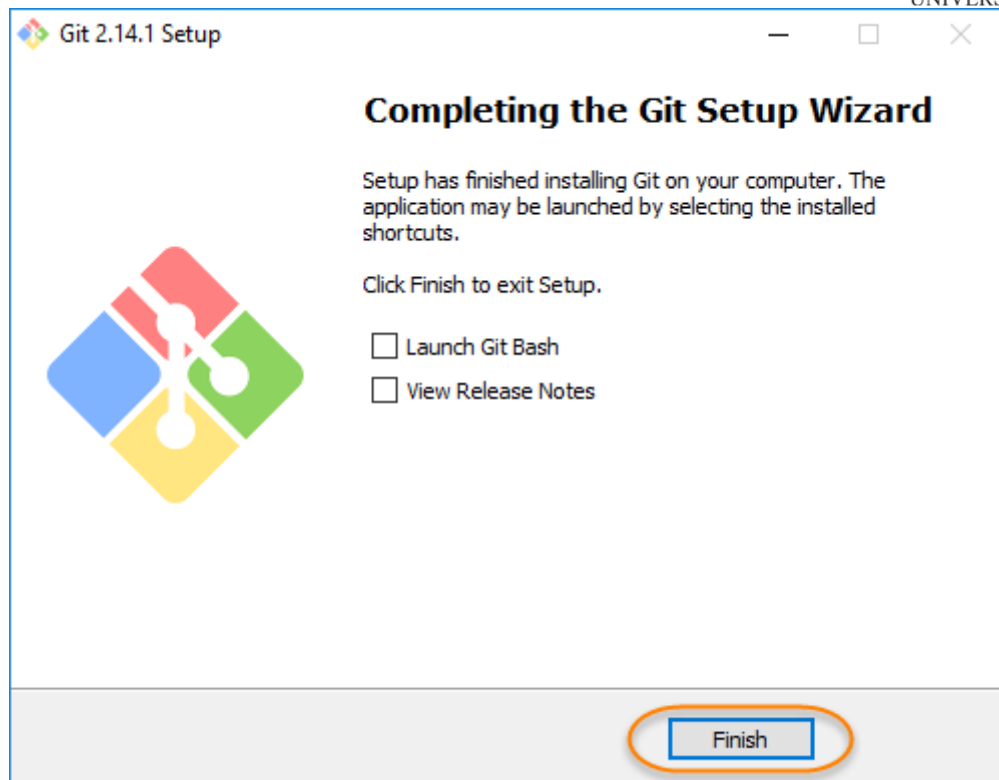
STEP 9: Just go with the default selections and click on install



STEP 10: Wait for minutes for the installation to complete.



STEP 11: Once done just click on finish button and the GitHub is installed on your systems.



AIM : TO SETUP GITHUB ACCOUNT

THEORY:

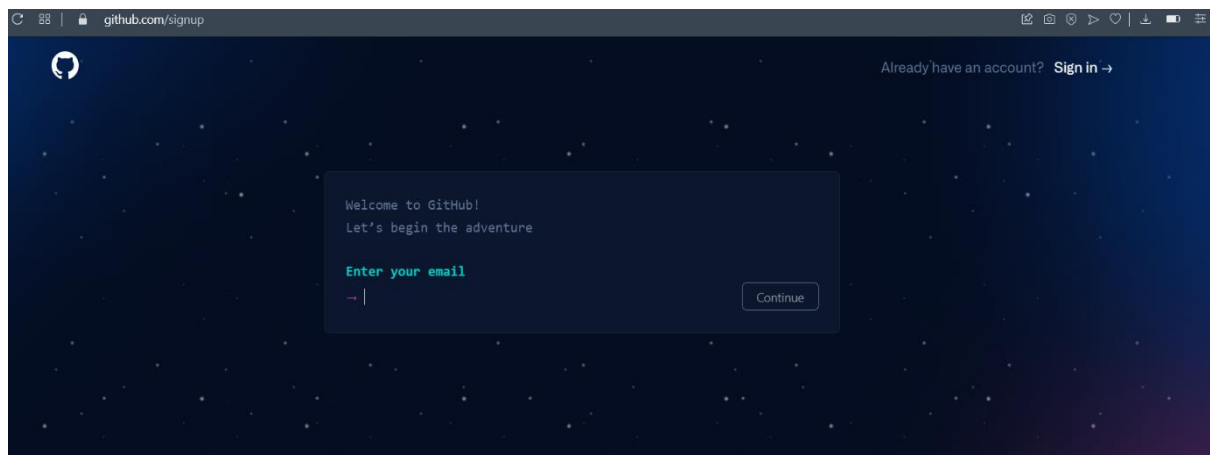
GitHub is a website and cloud-based service (client) that helps an individual or a developer to store and manage their code. We can also track as well as control changes to our or public code. GitHub's has a user-friendly interface and is easy to use .

We can connect the git-hub and git but using some commands shown below in figure 001. Without GitHub we cannot use Git because it generally requires a host and if we are working for a project, we need to share it with our team members, which can only be done by making a repository . Additionally , anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects

PROCEDURE:

STEP1: On any search engine like google, Microsoft edge, opera, etc. , search for git-hub.

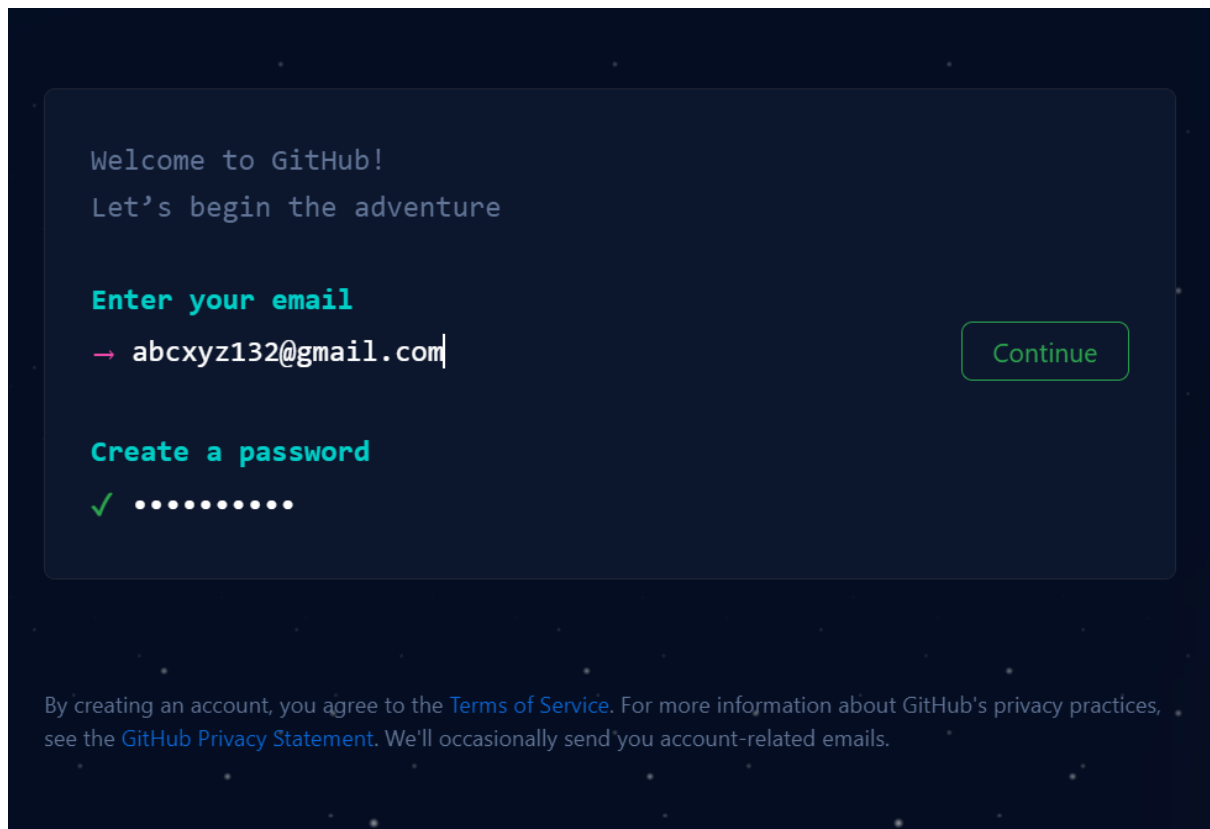
A dialogue box would appear welcoming you to the git-hub.



STEP2: If you already have an account, then on the top right corner there's an option of signing up. Click on "SIGN IN ".

STEP 3: But if you don't have any account, then enter your email and continue

STEP 4: Then create a strong password for your GitHub account



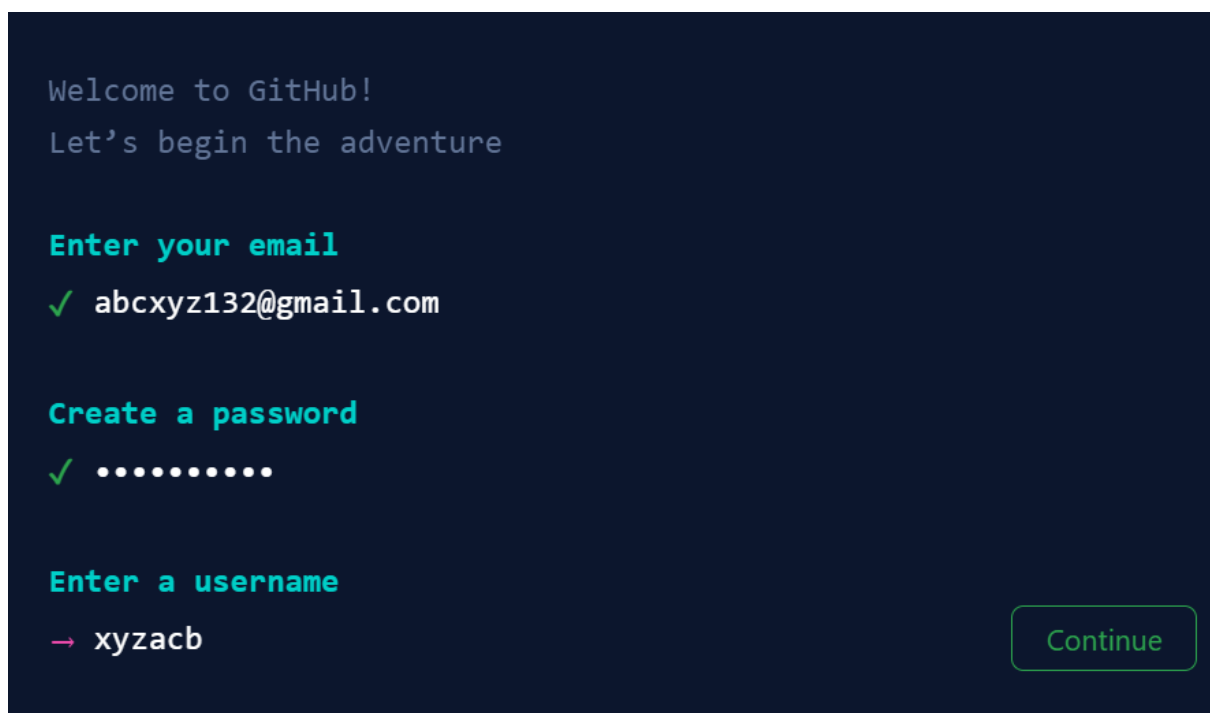
Welcome to GitHub!
Let's begin the adventure

Enter your email
→ abcxyz132@gmail.com

Create a password
✓

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

STEP 4: Create a username



Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ abcxyz132@gmail.com

Create a password
✓

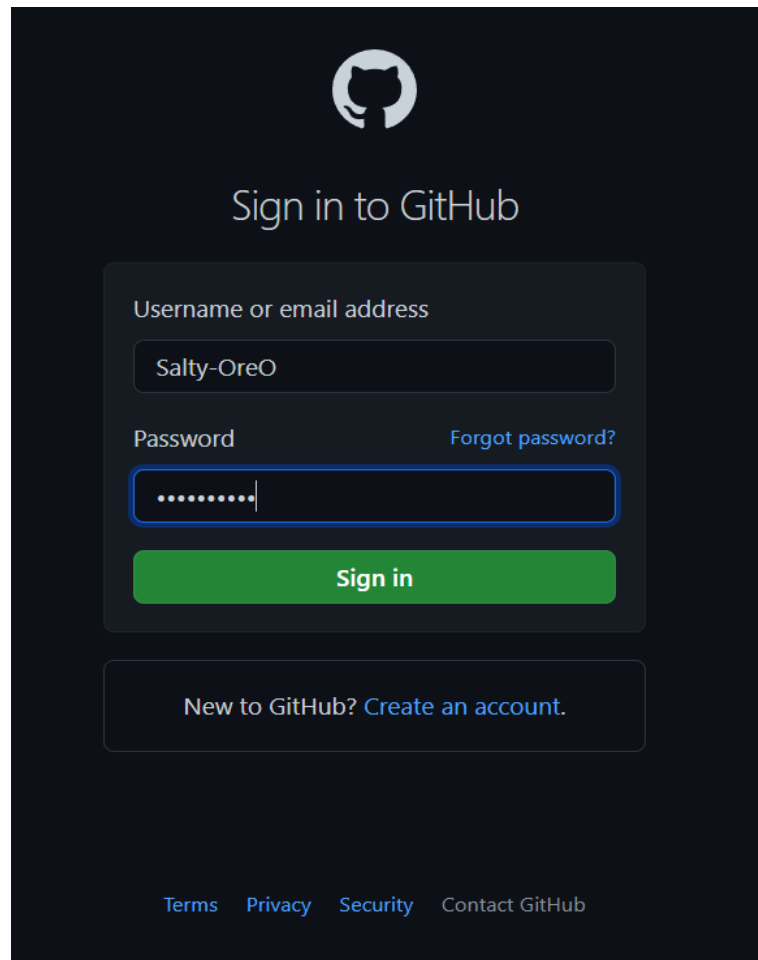
Enter a username
→ xyzacb

STEP 5: fill in all the details that are required and click on create account and your GitHub account would be created

BUT IF YOU ALREADY HAVE AN ACCOUNT, THEN FOLLOW THESE STEPS:

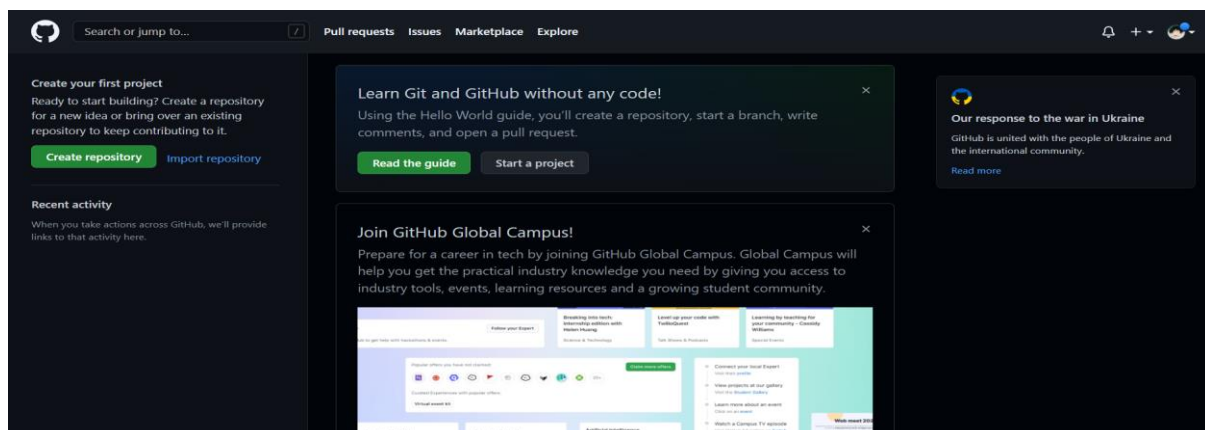
STEP 1: Sign In

Enter your email-address or username , then the password and continue



The image shows the GitHub sign-in page. At the top is the GitHub logo. Below it, the text "Sign in to GitHub" is centered. There are two input fields: "Username or email address" with the value "Salty-OreO" and "Password" with masked characters. A "Forgot password?" link is next to the password field. Below the fields is a green "Sign in" button. At the bottom, there is a link "New to GitHub? Create an account." and a footer with links for "Terms", "Privacy", "Security", and "Contact GitHub".

STEP 2: you have logged in in your account. Now you can create and edit any project



Linking GitHub account with Git Bash:

Username:

git config --global user.name "username in github"

Email:

git config --global user.email "your email in github"

Check Username & Email:

git config user.name

git config user.email

```
de11@DESKTOP-U9LGIF3 MINGW64 ~ (master)
$ git config --global user.name "HardikSharma"

de11@DESKTOP-U9LGIF3 MINGW64 ~ (master)
$ git config --global user.email "hardik0547.be21@chitkara.edu.in"

de11@DESKTOP-U9LGIF3 MINGW64 ~ (master)
$ git config user.name
HardikSharma

de11@DESKTOP-U9LGIF3 MINGW64 ~ (master)
$ git config user.email
hardik0547.be21@chitkara.edu.in
```

EXPERIMENT 3

AIM : PROGRAM TO GENERATE LOGS

THEORY:

Git Logs:

Logs are nothing but the history which we can see in git . It contains all the past commits, insertions and deletions in it which we can see any time.

Why do we need logs:

Logs helps to check that what were the changes in the code or any other file and by whom. It also contains the number of insertions and deletions including at which time it was changed.

PROCEDURE:

The command used to generate logs in git is :

STEP1: Create a file in the folder.

```
dell@DESKTOP-U9LGIF3 MINGW64 ~ (master)
$ mkdir project1

dell@DESKTOP-U9LGIF3 MINGW64 ~ (master)
$ cd project1

dell@DESKTOP-U9LGIF3 MINGW64 ~/project1 (master)
$ git init
Initialized empty Git repository in C:/Users/dell/project1/.git/

dell@DESKTOP-U9LGIF3 MINGW64 ~/project1 (master)
$ ls
first_prog.cpp
```

STEP 2: Check status

It will show the file name in red colour. This means that the file is untracked.

```
dell@DESKTOP-U9LGIF3 MINGW64 ~/project1 (master)
$ git status
On branch master

No commits yet

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

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

STEP 3: Staging of file

```

dell@DESKTOP-U9LGIF3 MINGW64 ~/project1 (master)
$ git add first_prog.cpp

dell@DESKTOP-U9LGIF3 MINGW64 ~/project1 (master)
$ git status
On branch master

No commits yet

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

```

STEP 4: Commit the file and check the status

```

dell@DESKTOP-U9LGIF3 MINGW64 ~/project1 (master)
$ git commit -m "This is a new file"
[master (root-commit) aedbddd] This is a new file
1 file changed, 8 insertions(+)
create mode 100644 first_prog.cpp

```

STEP 5: Now run the command \$ git log to generate the commits

```

create mode 100644 first_prog.cpp

dell@DESKTOP-U9LGIF3 MINGW64 ~/project1 (master)
$ git log
commit aedbddd84fc63a022648aa0e149e0e12d078889 (HEAD -> master)
Author: HardikSharma <hardik0547.be21@chitkara.edu.in>
Date:   Sat Apr 9 22:24:12 2022 +0530

    This is a new file

```

- ✓ As it can be observed, on using this command, the system displays all the changes made in the file or list of all the commits in the history along with the information of the user.
- ✓ **This commands clearly defines the git as the 'version-controlled system' as it allows us to rollback to any of the previous working states and keeps track of all the versions.**

AIM : TO CREATE AND VISUALIZE BRANCHES.

THEORY:

In Git, a branch is a new/separate version of the main repository.

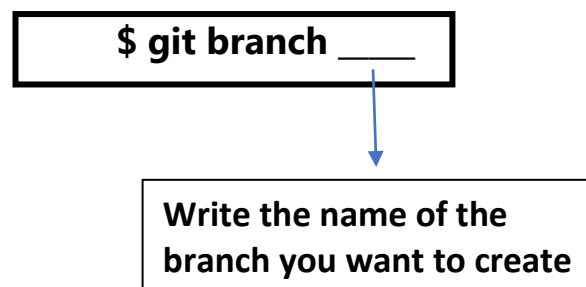
Branches allow you to work on different parts of a project without impacting the main branch.

When the work is complete, a branch can be merged with the main project.

You can even switch between branches and work on different projects without them interfering with each other.

FOLLOW THESE STEPS TO CREATE A NEW BRANCH IN GIT.

1.To create a branch, enter:

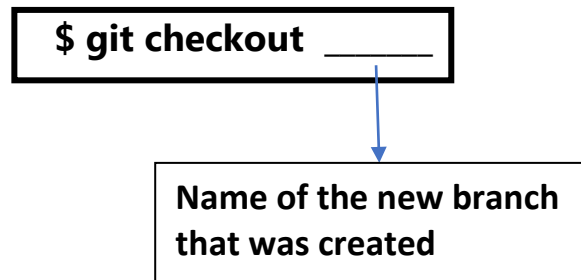


```
dell@DESKTOP-U9LGIF3 MINGW64 ~/project (master)
$ git branch activity1

dell@DESKTOP-U9LGIF3 MINGW64 ~/project (master)
$ git branch
  activity1
* master
```

The branch has been created, the * beside master specifies that we are currently on that branch.

2.The next step is to transfer the data from the master branch to the new branch. For this we use:



```

de11@DESKTOP-U9LGIF3 MINGW64 ~/project1 (master)
$ git checkout file1
Switched to branch 'file1'
  
```

checkout is the command used to check out a branch. Moving us from the current branch, to the one specified at the end of the command:

3.Staging the file.

We have made changes to a file.

Now check the status of the current branch:

You would observe the name of the file is in red colour with a notation “untracked “.

```

nothing to commit, working tree clean
de11@DESKTOP-U9LGIF3 MINGW64 ~/project1 (file1)
$ git status
On branch file1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   first_prog.cpp

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

This means that only the data has been transferred to the file but we cannot make changes in the same as the current working directory is the master branch.

To overcome this, we stage the file by using the command:

\$ git add --all

```
de11@DESKTOP-U9LGIF3 MINGW64 ~/project1 (file1)
$ git add --all

de11@DESKTOP-U9LGIF3 MINGW64 ~/project1 (file1)
$ git status
On branch file1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   first_prog.cpp
```

Check if the file is staged or not by using \$ git status command.

- ✓ As you can observe the file name is green in colour now which denotes the file is staged.

4.Commit

After staging we need to commit. This assures the system that the directory has been shifted to the new file. For this, the command used is:

\$ git commit-m "message"

```
de11@DESKTOP-U9LGIF3 MINGW64 ~/project1 (file1)
$ git commit -m "Added a new line"
[file1 67f8862] Added a new line
1 file changed, 4 insertions(+), 2 deletions(-)
```

5.Check the status

On checking the status, a message will be displayed as ‘ working tree clean’. Which means all the files inside that directory are tracked.

```
de11@DESKTOP-U9LGIF3 MINGW64 ~/project1 (file1)
$ git commit -m "Added a new line"
[file1 67f8862] Added a new line
1 file changed, 4 insertions(+), 2 deletions(-)

de11@DESKTOP-U9LGIF3 MINGW64 ~/project1 (file1)
$ git status
On branch file1
nothing to commit, working tree clean

de11@DESKTOP-U9LGIF3 MINGW64 ~/project1 (file1)
$ |
```

Now you can make any of the changes in the file but the modifications won't be reflected in the master branch.

EXPERIMENT 5

AIM: TO EXPLAIN GIT LIFECYCLE

THEORY: 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:

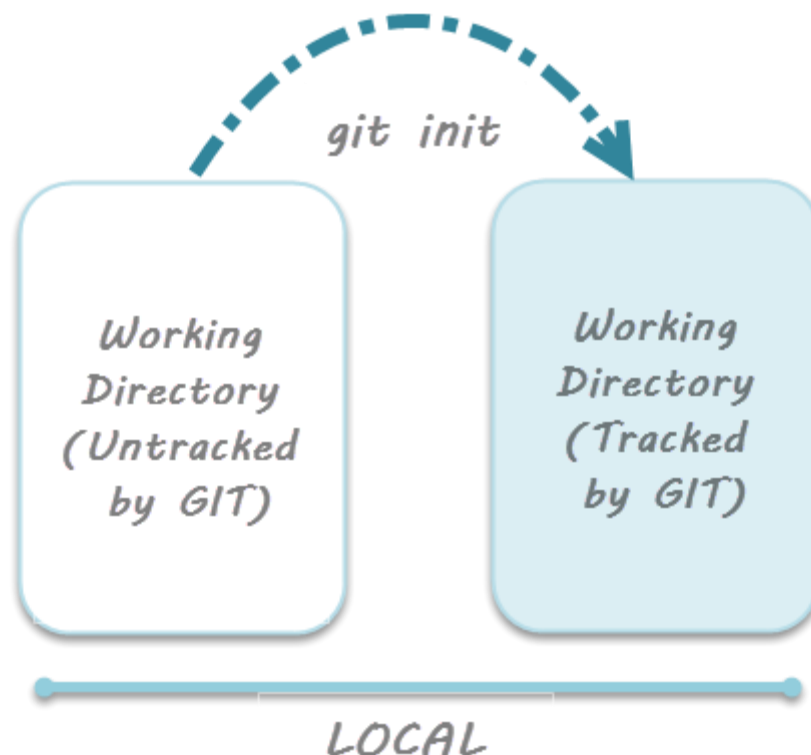
1. **Working directory**
2. **Staging area**
3. **Git directory**

1. 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.

Working directory is the directory containing hidden .git folder.

git init - Command to initialize a Git repository



reference for picture: www.toolsqa.com/git/git-life-cycle/

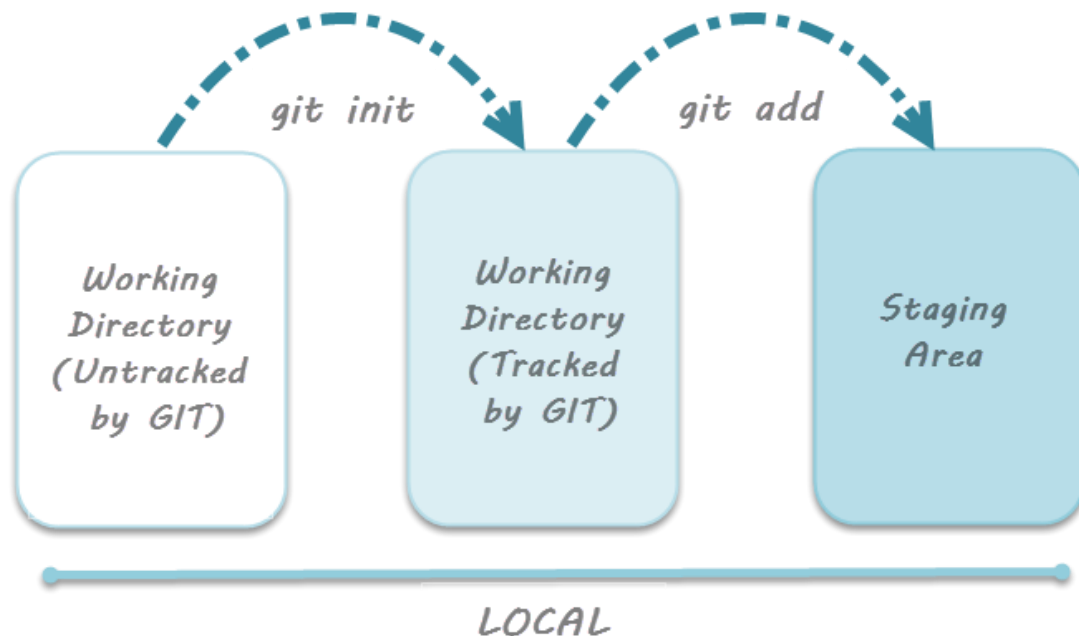
2. Staging area

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.

git add - Command to add files to staging area.



{reference for picture: www.toolsqa.com/git/git-life-cycle/ }

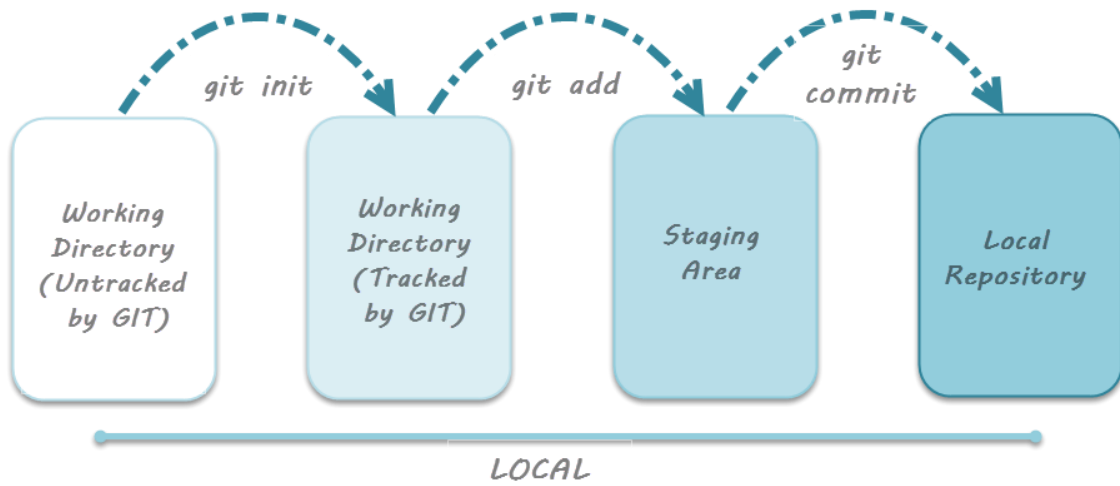
3. Git Directory

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 is stored in the Git directory.

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

git commit -m"your message" - Command to commit files to Git repository with message.



{reference for picture: www.toolsqa.com/git/git-life-cycle/ }