Subject Name: Source Code Management

Subject Code: CS181

Cluster: Beta

Department: DCSE

SUBMITTED BY:

AASHISH BHARDWAJ

2110990022

SUBMITTED TO:

DR MONIT KAPOOR

AIM :-Setting up of Git Client

Theory:

GIT –> It is basically used for pushing and pulling of code. We can use git and git-hub parallelly to work with multiple members or individually. We can make, edit, recreate, copy or download any code on git hub using git.

Advantages of GIT –>

1. Git performs very strongly and reliably when compared to other version control systems

2. Git is the kind of flexibility it offers to support several kinds of nonlinear development workflows and its efficiency in handling both small scale and large scale projects as well as protocols

3. Git offers the type of performance, functionality, security, and flexibility that most developers and teams need to develop their projects

4. Git is a widely supported open source project

Procedure:

On Mac running latest OS Git is preinstalled

You can check it by running git —version in terminal

AIM:-Setting Up GitHub Account

Theory:

What is GitHub **->** GitHub is a website and cloud-based service (client) that helps an individual or a developers to store and manage their code. We can also track as well as control changes to our or public code

Advantages of GITHUB –>

1. GitHub's has a user-friendly interface and is easy to use .
2. We can connect the git-hub and git but using some commands shown below in figure 001.
3. 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 will our team members, which can only be done by making a repository.
4. Additionally , anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects.

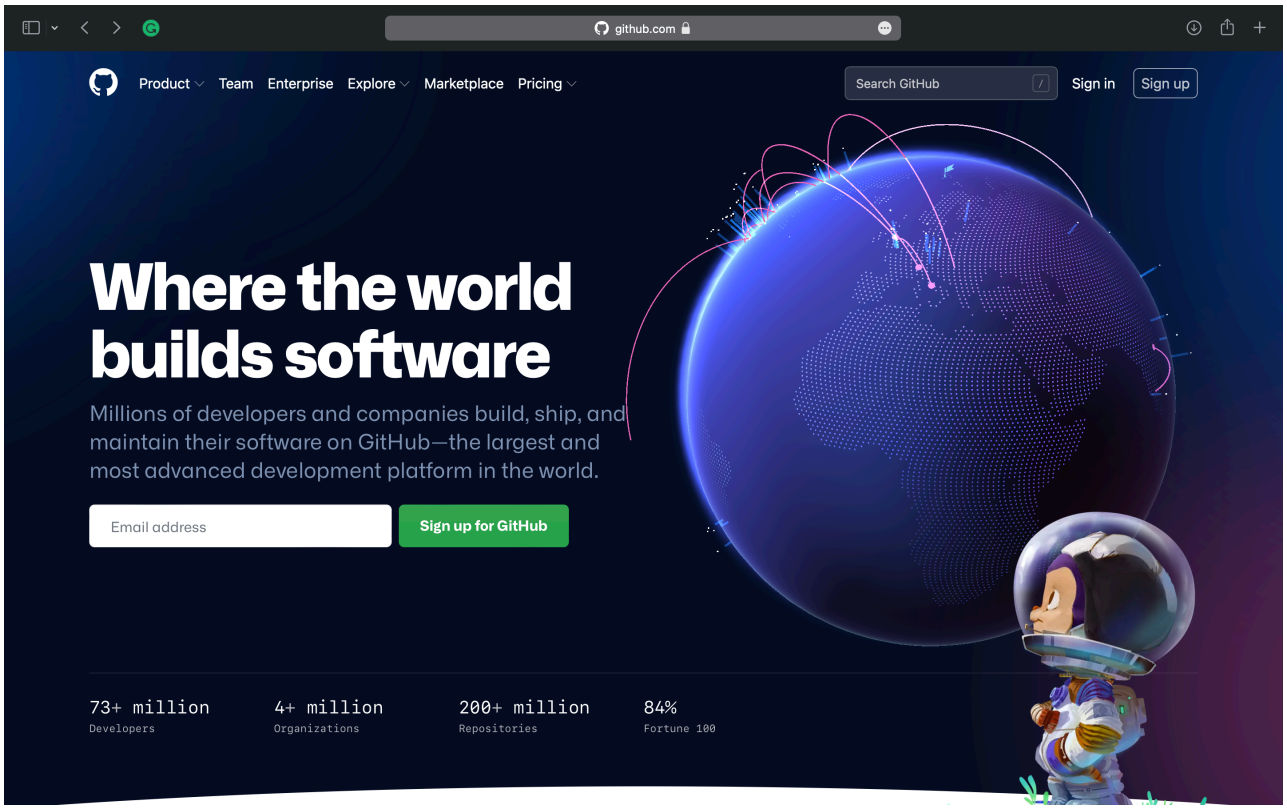
Procedure:-

Step1 :-

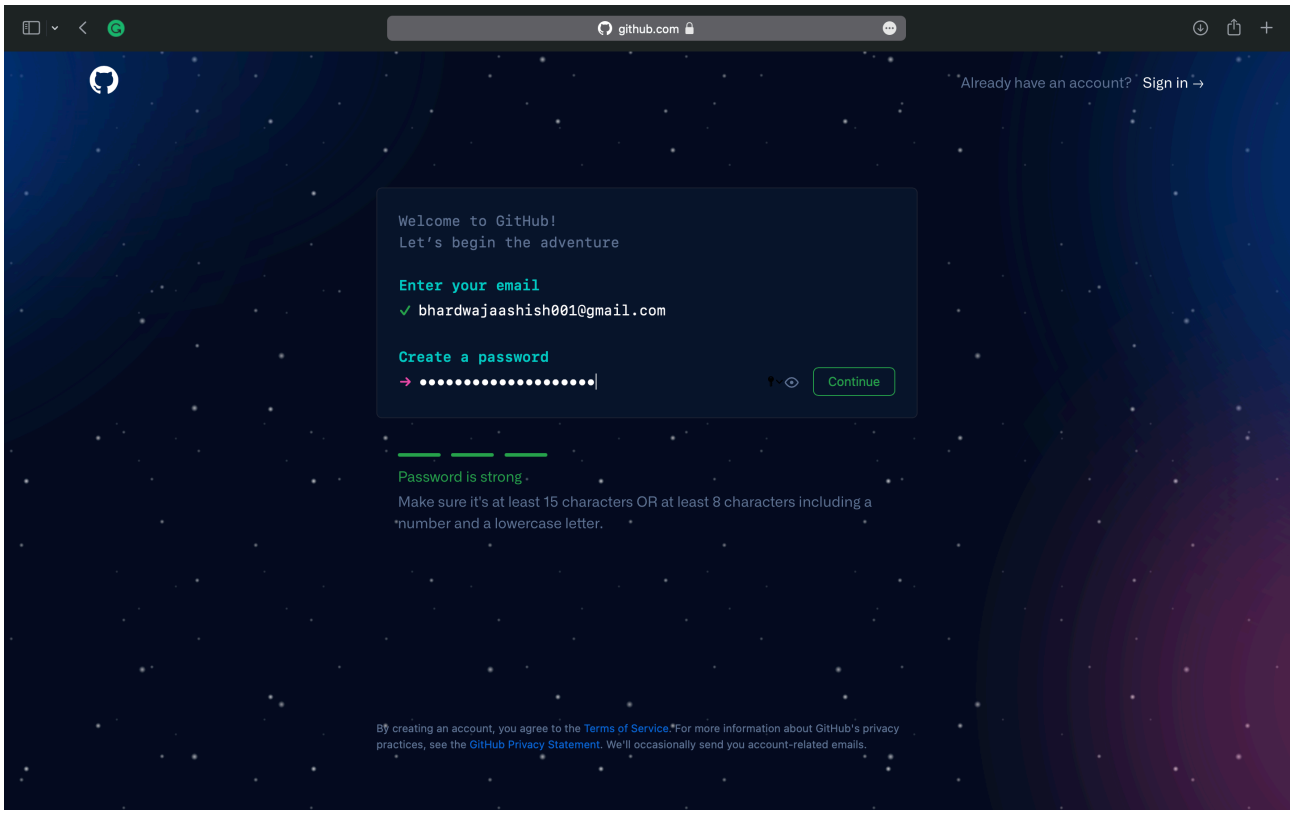
Google (any search engine)
Search for git-hub or (https://github.com/signup).

Step2 :-

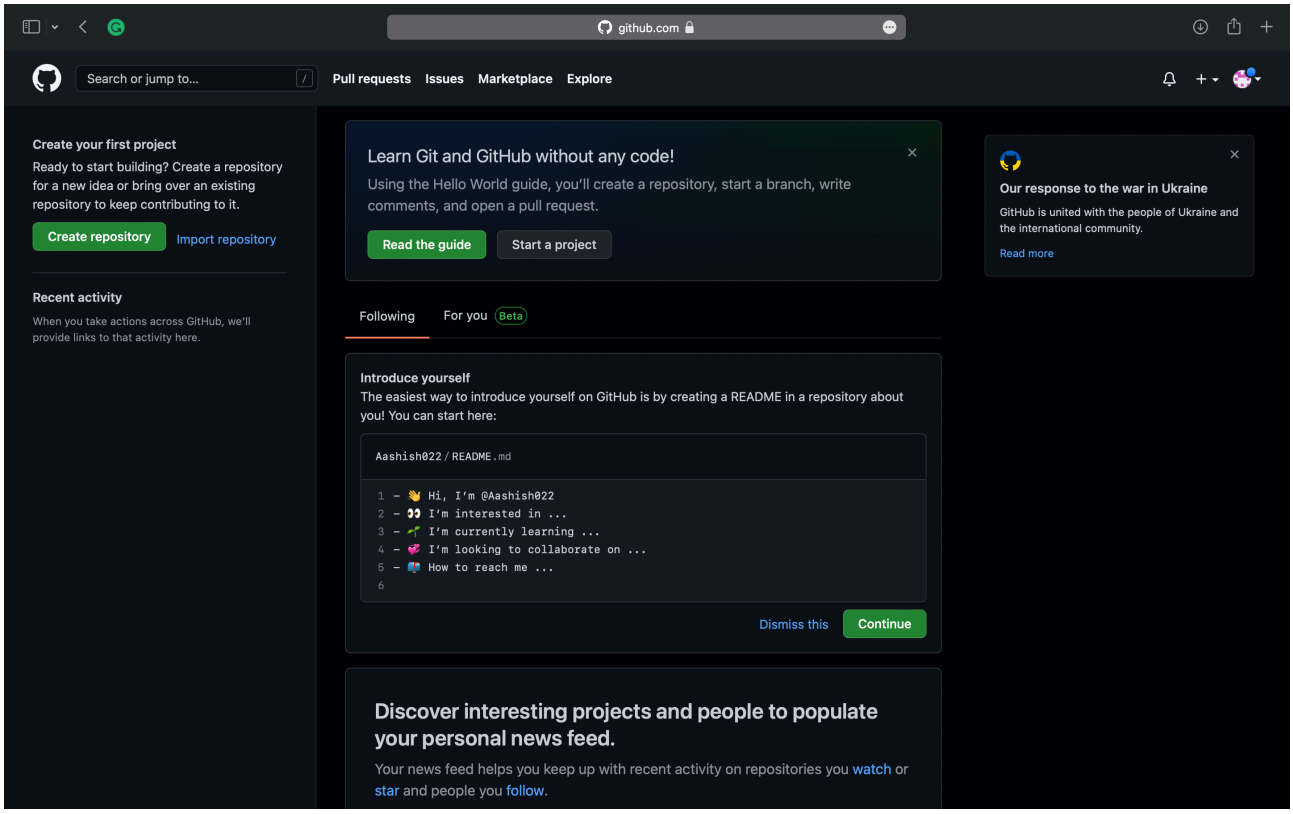
After visiting the link this type of interface will appear, if you already have account you can sign in and if not you can create.



Sign in into GitHub :-



Interface of GitHub :-



To link GitHub account with Git bash –

For username:-

git config --global user.name “username in git-hub”

For user email:-

git config --global user.email “your email in git-hub”

To verify:-

git config user.name
git config user.email

AIM:-How to use Git Log

Theory:

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.

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:-

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

```
(base) aashishbhardwaj@Aashishs-MacBook-Air github % git status
On branch master

No commits yet

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

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

Staging of file

```
(base) aashishbhardwaj@Aashishs-MacBook-Air github % git add .
(base) aashishbhardwaj@Aashishs-MacBook-Air github % git status
On branch master

No commits yet

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

Now run the command $ git log to generate all the commits in the repository

```
(base) aashishbhardwaj@Aashishs-MacBook-Air github % git log
commit df4c386fc2d6c908d00f76a3df92218cfa70e62d (HEAD -> master)
Author: "Aashish0022" <aashish0022.be21@chitkara.edu.in>
Date:   Sat Apr 9 20:52:58 2022 +0530

    this file contains all the changes
```

AIM:-Create and Visualize branch

Theory:

A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process. You can think of them as a way to request a brand new working directory, staging area, and project history. New commits are recorded in the history for the current branch, which results in a fork in the history of the project

Procedure:-

Creating a Branch

```
(base) aashishbhardwaj@Aashishs-MacBook-Air github % git branch
* master
(base) aashishbhardwaj@Aashishs-MacBook-Air github % git branch xyz
(base) aashishbhardwaj@Aashishs-MacBook-Air github % git branch
* master
  xyz
```

Changing Branch

```
(base) aashishbhardwaj@Aashishs-MacBook-Air github % git checkout xyz
Switched to branch 'xyz'
```

Staging the File

```
(base) aashishbhardwaj@Aashishs-MacBook-Air github % git status
On branch xyz
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .txt
        exp

nothing added to commit but untracked files present (use "git add" to track)
(base) aashishbhardwaj@Aashishs-MacBook-Air github % git add --a
(base) aashishbhardwaj@Aashishs-MacBook-Air github % git status
On branch xyz
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .txt
        new file:   exp

(base) aashishbhardwaj@Aashishs-MacBook-Air github % git commit -m "this is new file"
[xyz 11a3c5c] this is new file
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 .txt
 create mode 100644 exp
(base) aashishbhardwaj@Aashishs-MacBook-Air github % git status
On branch xyz
nothing to commit, working tree clean
```

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

AIM:- Git Life Cycle Description

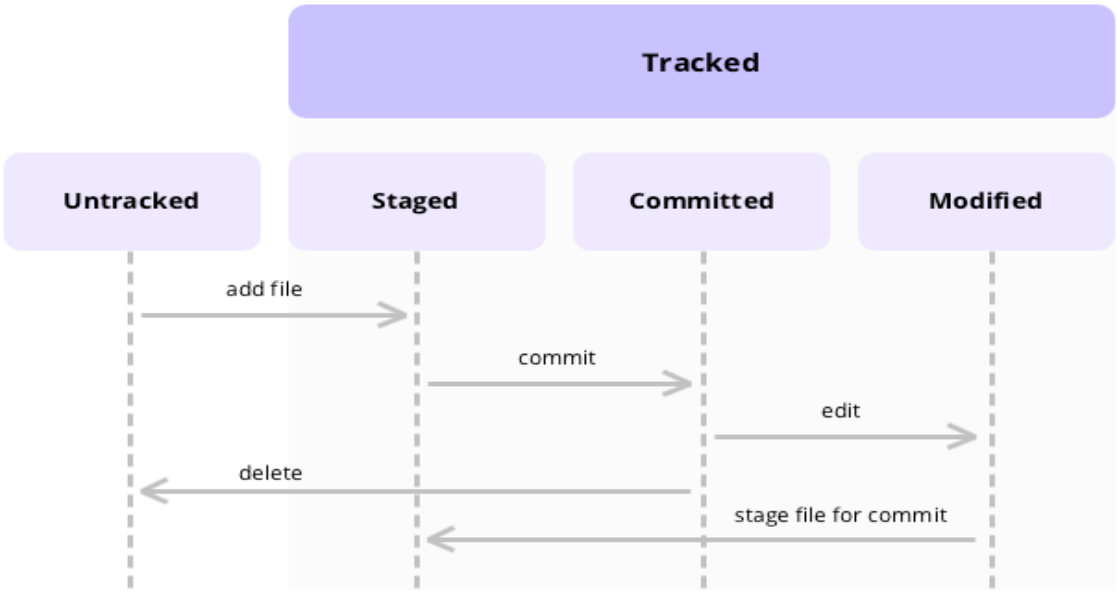
Introduction to Git Life Cycle

Git is one of the premier distributed version control systems available for programmers and corporates. In this article, we will see details about how a project that is being tracked by git proceeds with workflow i.e Git Life Cycle. As the name suggests is regarding different stages involved after cloning the file from the repository. It covers the git central commands or main commands that are required for this particular version control system

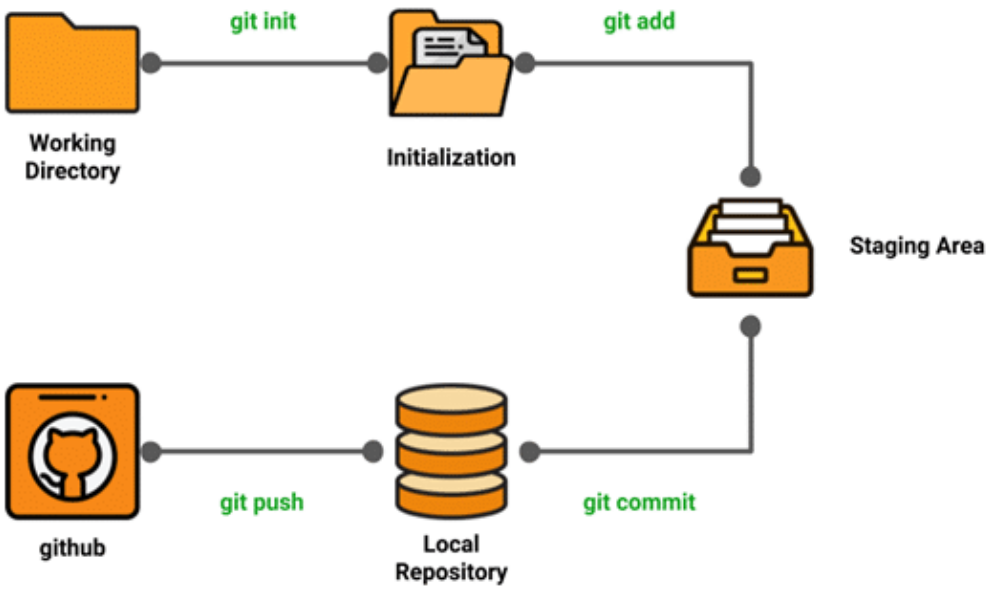
Workflow of Git Life Cycle

The workflow of the Git as follows:

- We will create a branch on which we can work on and later we will merge it with master
- Clone: First, when we have code present in the remote repository, we clone to local to form something called a local repository.
- Modifications/Adding Files: we perform several developments on the existing files or may as well add new files. Git will monitor all these activities and will log them.

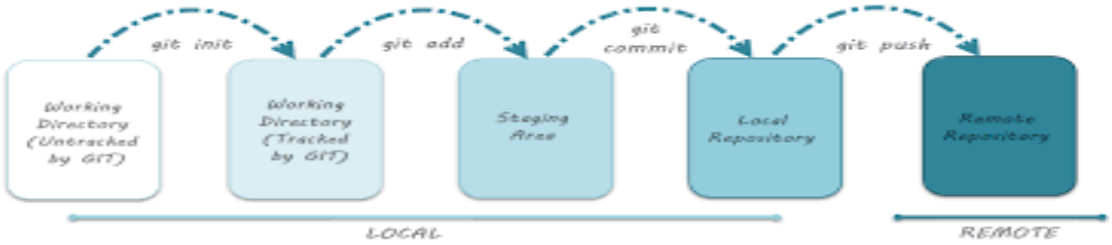


- We need to move the content that we require to transform to the master to the staging area by using git commands and the snapshot of staged files will be saved in the git staging area.
- We need to perform commits on the files that are staged and the recorded snapshot from the above steps will be permanently saved on the local repo and this particular is recorded by commit message for future referrals.



- Once we commit the code is available on the local repo but to send it to the master repo we need to perform PUSH operation
- If someone else is working on the same branch then there will be a possibility that he might have added his changes to the master by push. So we need to perform PULL operation before the PUSH operation if multiple people are working on the same branch and this workflow as shown below.
- Once the target branch is updated we need to get all the required approvals so that merge operation with the master is allowed.

This is the basic workflow of git was lots of intermediate commands like git add, git status, git commit, git push origin, git rebase, git merge, git diff, etc will be used depending upon the requirement of the user.



Stages of Git Life Cycle

So we have seen the workflow of the git life cycle above. But we need to know that we have a project linked with git then that project can reside in there of the following areas. Below mentioned areas are ingredients to the recipe of Git and having an idea of them will help you a lot to track the files that you are working on.

The stages are as discussed:

- Working Directory
- Staging Area
- Git Directory

These Three Stages are explained below:

1. Working Directory

- If you have your project residing on to your local machines then basically it is called even though it is linked to git or not. In either case, it will be called as the working directory. But when the available project is linked with git then basically there will be .git folder hidden in the so-called working directory. So the presence of the .git folder is enough to say that the folder is working copy on the machine and it is tracked by the git.
- At this stage, git knows what are the files and folders that it's tracking that's it. No other info will be available regarding this. To make sure that the newly added files get tracked in the working copy we need to make sure that those files are staged and this is our second residence for the files.

2. Staging Area

- When we make changes to the existing files in the working repo or if we add any folder of files and if we want these changes to need to be tracked and also need to be moved to the local repo for tracking then we need to move these changed files or newly added folder of file to the staging area. Git add is the basic command which will be used to move the modified files to the staged area.

- It’s ticked that been give to modified files or newly added folder of file to travel to the local repo for further traction. Those files that don’t have that ticket will be tracked by the git but they won’t be able to move to the target easily. Here index plays a critical role. GIT Index is something that comes in between local repo and working directory and it is the one that decides what needs to be sent to the local repo and in fact, it decides what needs to be sent to the central repo.

3. GIT Directory

- When we have done the modifications or addition of files or folder and want them to be part of the repository they first we do is to move them to the staging area and they will commit ready. When we commit then provide the appropriate commit message and files will be committed and get updated in the working directory.
- Now git tracks the commits and commit messages and preserves the snapshot of commit files and this is done in the Git specific directory called Git Directory. Information related to all the files that were committed and their commit messages will be stored in this directory. We can say that this git directory stores the metadata of the files that were committed.
-