Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **DCSE**

Submitted By:
AarushiVashisht
2110990016
G1

Submitted To:
Monit Kapoor

Aim: Setting up of Git Client

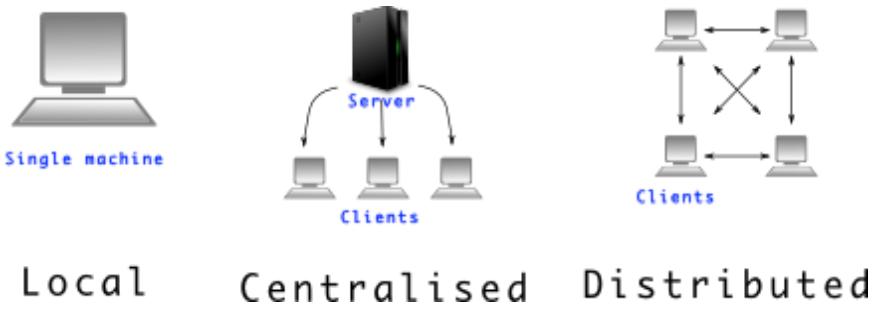
Theory:

What is Git?

Git is a software used for tracking changes in any set of files, usually used for coordinating work among members of a team.

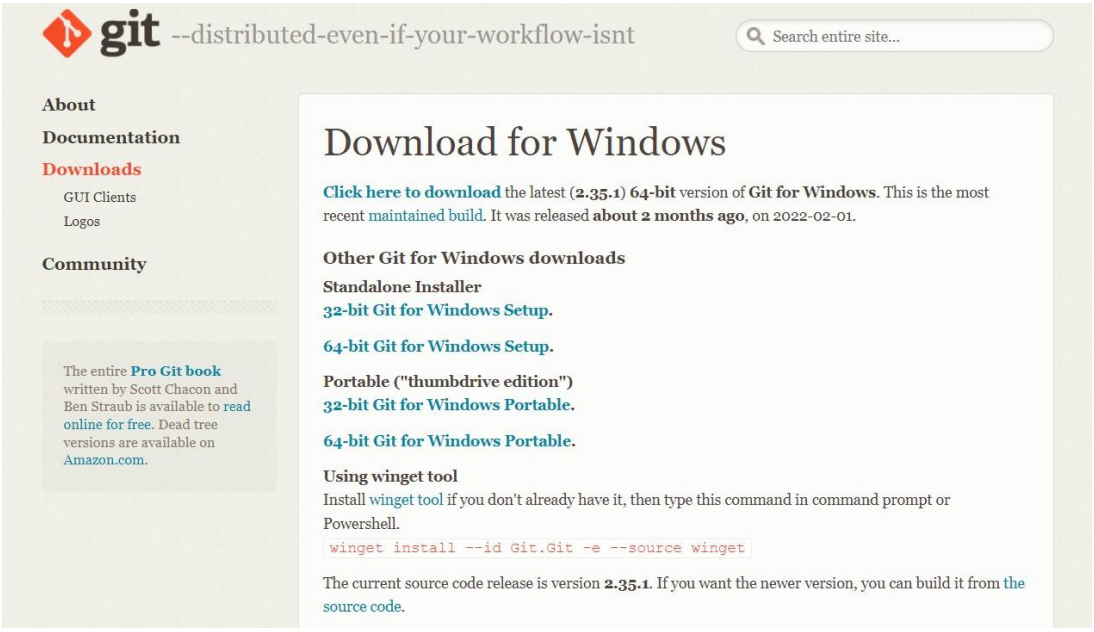
History of VCS:

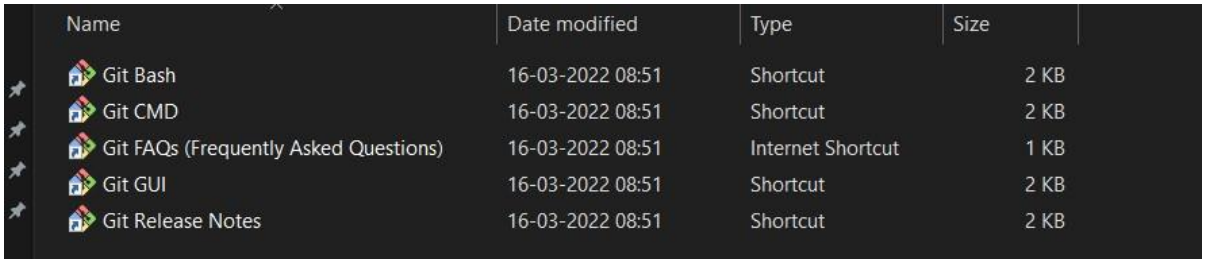
- **Local VCS:** No internet is needed because it uses a database to keep track of files.
- **Centralized VCS:** Centralized version control systems are based on the idea that there is a single "central" copy of your project somewhere (probably on a server), and programmers will "commit" their changes to this central copy. "Committing" a change simply means recording the change in the central system.
- **Distributed VCS:** A type of version control where the complete codebase including its full version history is mirrored on every developer's computer.



How to install GIT on Windows?

There are also a few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to https://git-scm.com/download/win and the download will start automatically. Note that this is a project called Git for Windows, which is separate from Git itself; for more information on it, go to https://gitforwindows.org.





Check version of git by using git –version command.

Aim: Setting up GitHub Account

Theory:

What is GitHub?

GitHub is a code hosting platform for version control and collaboration. In other words, it manages repositories.

Advantages:

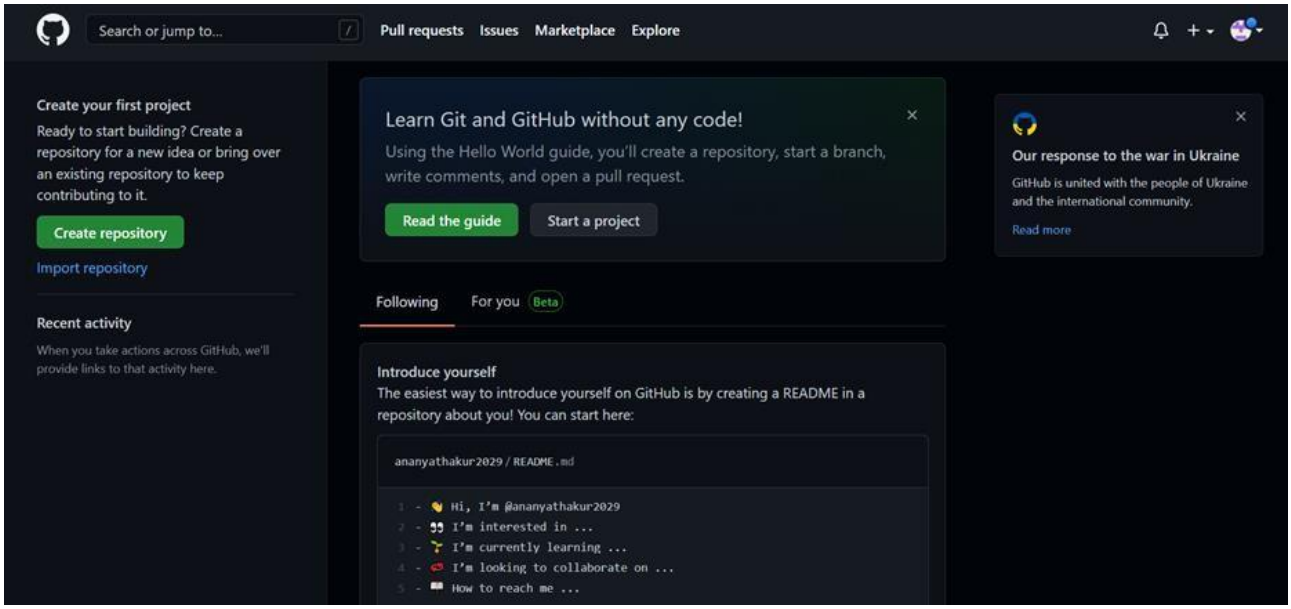
- It makes it easy to contribute to Open-Source projects.
- Track changes in your code across versions.

Procedure:

Search for GitHub in any search engine or https://github.com/signup

If you're a new user add your email and click on **Sign up for GitHub**. Otherwise click on **Sign In** at the top right corner

Signing into GitHub:



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

```
MINGW64:/c/Users/91935/Desktop/github

91935@Aarushii MINGW64 ~/Desktop/github (master)
$ git config --global user.name "aarushiv16"

91935@Aarushii MINGW64 ~/Desktop/github (master)
$ git config user.name
aarushiv16

91935@Aarushii MINGW64 ~/Desktop/github (master)
$ git config --global user.email "aarushi0016.be@chitkara.edu.in"

91935@Aarushii MINGW64 ~/Desktop/github (master)
$ git config user.email
aarushi0016.be@chitkara.edu.in

91935@Aarushii MINGW64 ~/Desktop/github (master)
$
```

Aim: Program to Generate log

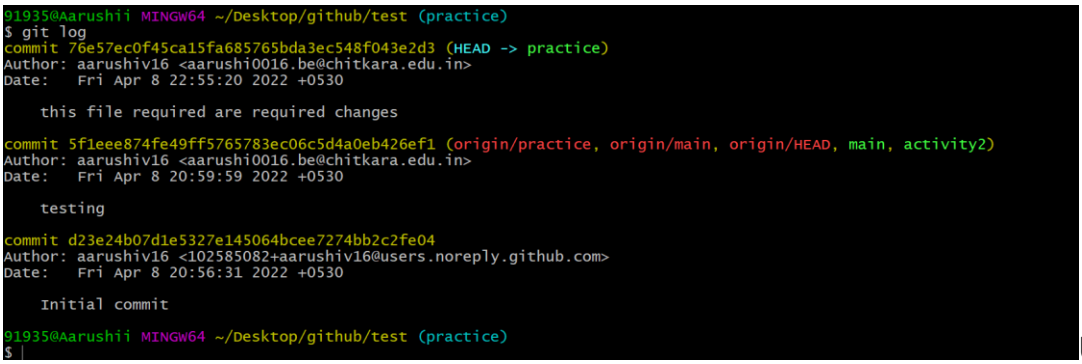
Theory:

Git Logs:

Logs are nothing but the history which we can see in Git by using the code Git log. It contains all the past commits, insertions and deletions which can be seen anytime.

Why do we need logs?

Logs help us to check the changes made in code or files and by whom. It also contains the details of insertions and deletions and also the time it was changed at.

- 

Use command git log **to access logs.**

Aim: Create and visualize branches

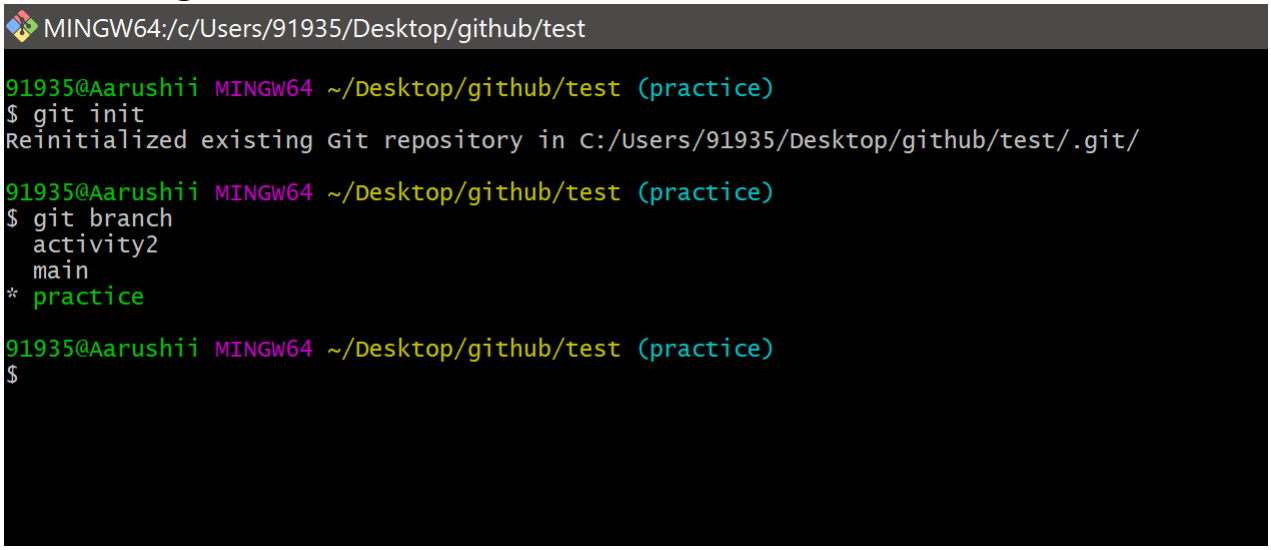
Theory:

How to create branches?

The main branch in git is called the master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the files which are created in branch are not shown in master branch. We can also merge both the parent (master) and child (other branches).

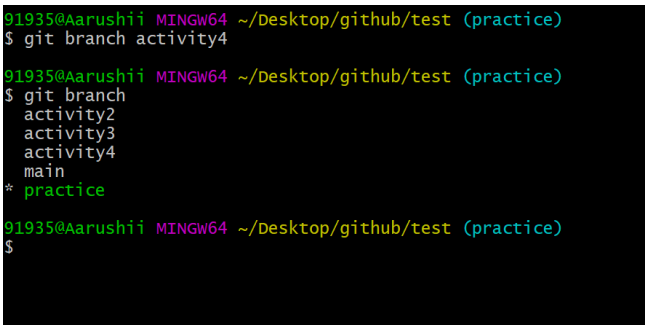
Follow these steps to create a separate branch in git.

1. **Firstly, you can check the present branches in the master branch with the help of this command.**
   $ git branch



2. **To create a branch, enter:**

   $git branch



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

$git checkout

```
91935@Aarushii MINGW64 ~/Desktop/github/test (practice)
$ git checkout activity3
Switched to branch 'activity3'

91935@Aarushii MINGW64 ~/Desktop/github/test (activity3)
$
```

All the data from the master branch has been transferred to this branch.

4.Staging the file

Create a file and check the status.

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

```
91935@Aarushii MINGW64 ~/Desktop/github/test (activity3)
$ touch aaru.txt

91935@Aarushii MINGW64 ~/Desktop/github/test (activity3)
$ git status
On branch activity3
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        aaru.txt

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

91935@Aarushii MINGW64 ~/Desktop/github/test (activity3)
$
```

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 --a

```
91935@Aarushii MINGW64 ~/Desktop/github/test (activity3)
$ git add --a

91935@Aarushii MINGW64 ~/Desktop/github/test (activity3)
$ git status
On branch activity3
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   aaru.txt


91935@Aarushii MINGW64 ~/Desktop/github/test (activity3)
$
```

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

5 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

```
91935@Aarushii MINGW64 ~/Desktop/github/test (activity3)
$ git commit -m"this is new file"
[activity3 292fb53] this is new file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 aaru.txt

91935@Aarushii MINGW64 ~/Desktop/github/test (activity3)
$
```

6 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.

```
91935@Aarushii MINGW64 ~/Desktop/github/test (activity3)
$ git status
On branch activity3
nothing to commit, working tree clean

91935@Aarushii MINGW64 ~/Desktop/github/test (activity3)
$
```

Now you can make any of the changes in the file but modifications wont be reflected in master branch.

Aim: Git lifecycle description

Theory:

Stages in GIT Life Cycle:

Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, 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:

- Working directory
- Staging area
- Git directory

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.

Staging Area:

Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.

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.

Remote Repository: means mirror or clone of the local Git repository in GitHub. And pushing means uploading the commits from local Git repository to remote repository hosted in GitHub.