Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Zeta**

Department: **DCSE**

Submitted By:

khushi

2110992029

G27 - A

Submitted To:

Dr. Anuj Jain

List of Tasks

Task 1

Installing and Configuring the Git client

The following sections list the steps required to properly install and configure the Git clients - Git Bash and Git GUI - on a Windows 7 computer.
Git is also available for Linux and Mac. The remaining instructions here, however, are specific to the Windows installation.
Be sure to carefully follow all of the steps in the first five sections. The last section, 6, is optional.

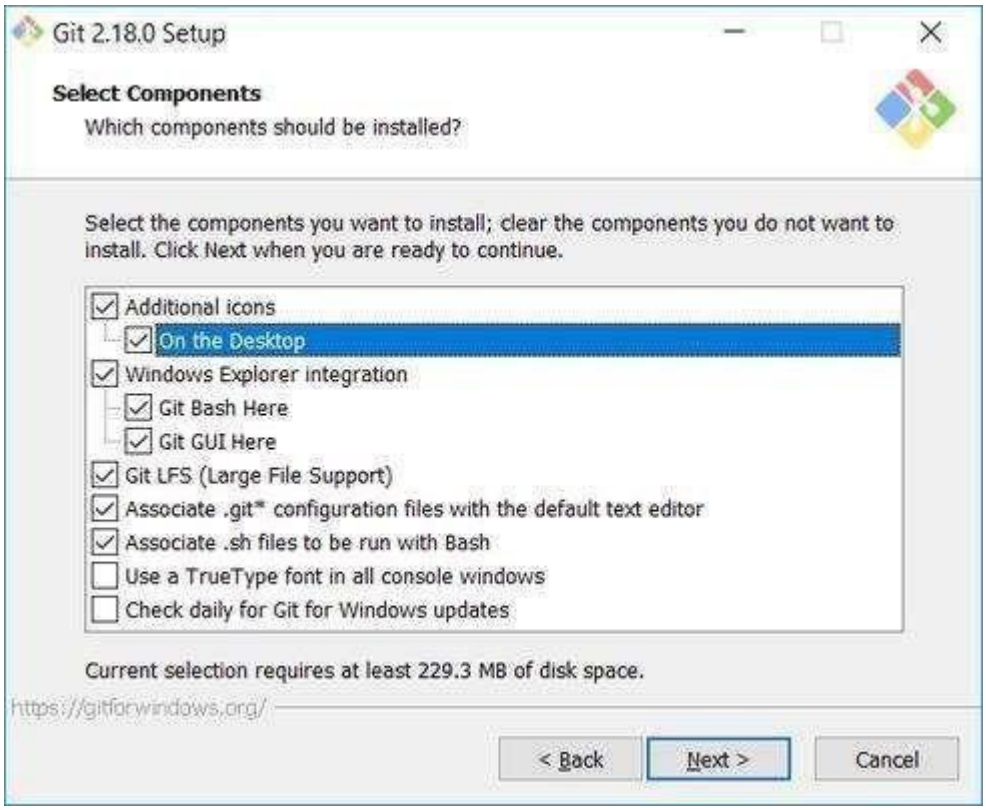
There is also a section on common problems and possible fixes at the bottom of the document.

1. Git installation

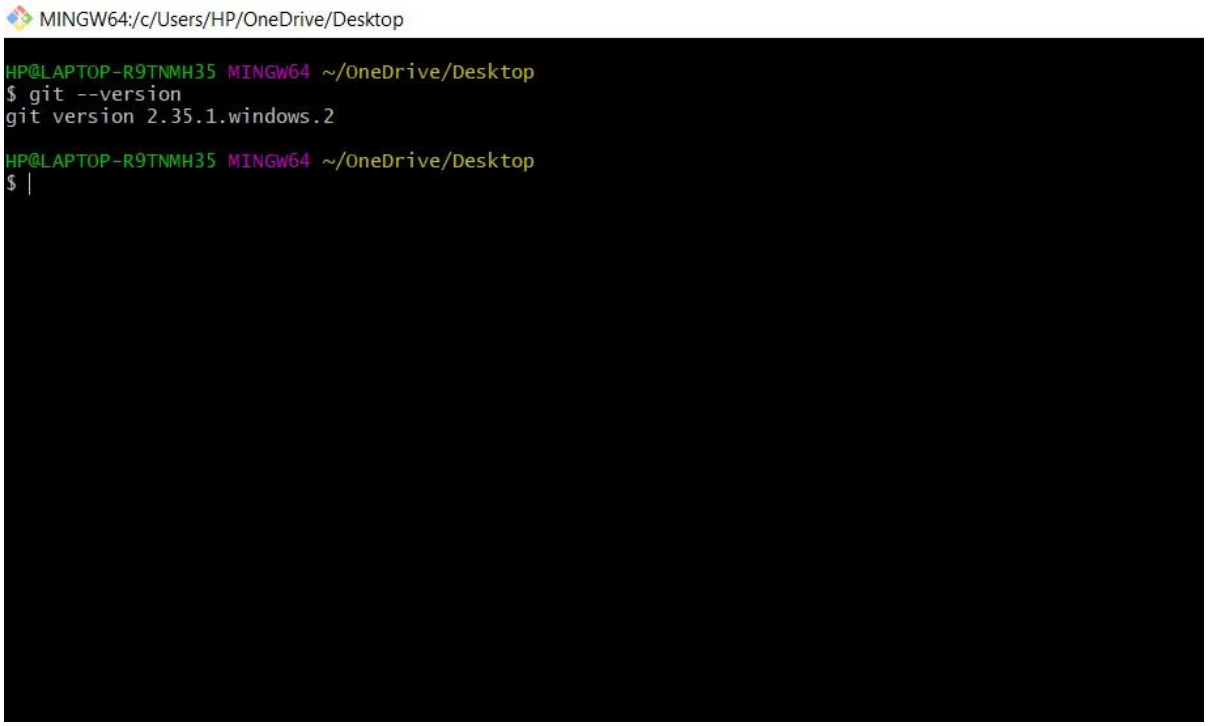
Download the Git installation program (Windows, Mac, or Linux) from http://git-scm.com/downloads.

When running the installer, various screens appear (Windows screens shown). Generally, you can accept the default selections, ***except in the screens below where you do NOT want the default selections:***

In the **Select Components** screen, make sure **Windows Explorer Integration** is selected as shown:


Git 2.18.0 Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Additional icons
On the Desktop
Windows Explorer integration
Git Bash Here
Git GUI Here
Git LFS (Large File Support)
Associate .git* configuration files with the default text editor
Associate .sh files to be run with Bash
Use a TrueType font in all console windows
Check daily for Git for Windows updates
Current selection requires at least 229.3 MB of disk space.
https://gitforwindows.org/
< Back
Next >
Cancel

Check for version :



Task 2

Setting up Github account

1. Creating an account

To sign up for an account on GitHub.com, navigate to https://github.com/ and follow the prompts. To keep your GitHub account secure you should use a strong and unique password. For more information, see "Creating a strong password."

2. Choosing your GitHub product You can choose GitHub Free or GitHub Pro to get access to different features for your personal account. You can upgrade at any time if you are unsure at first which product you want. For more information on all of GitHub's plans, see "GitHub's products."

3. Verifying your email address

To ensure you can use all the features in your GitHub plan, verify your email address after signing up for a new account. For more information, see "Verifying your email address."

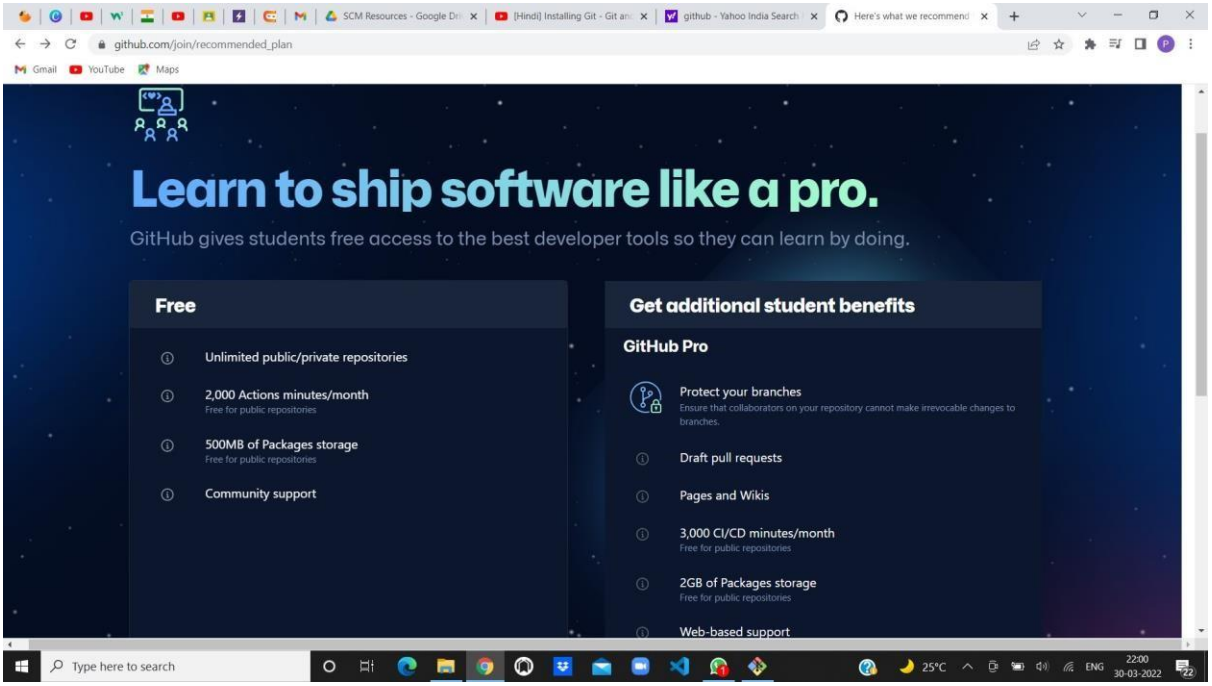
4. Configuring two-factor authentication

Two-factor authentication, or 2FA, is an extra layer of security used when logging into websites or apps. We strongly urge you to configure 2FA for the safety of your account. For more information, see "About two-factor authentication."

5. Viewing your GitHub profile and contribution graph

Your GitHub profile tells people the story of your work through the repositories and gists you've pinned, the organization memberships you've chosen to publicize, the contributions you've made, and the projects you've created. For more information, see "About your profile" and "Viewing contributions on your profile."

Screenshot :


github.com/join/recommended_plan
Gmail
YouTube
Maps
Learn to ship software like a pro.
GitHub gives students free access to the best developer tools so they can learn by doing.
Free
Unlimited public/private repositories
2,000 Actions minutes/month
Free for public repositories
500MB of Packages storage
Free for public repositories
Community support
Get additional student benefits
GitHub Pro
Protect your branches
Ensure that collaborators on your repository cannot make irrevocable changes to branches.
Draft pull requests
Pages and Wikis
3,000 CI/CD minutes/month
Free for public repositories
2GB of Packages storage
Free for public repositories
Web-based support
Type here to search
25°C
ENG
22:00
30-03-2022

Task 3

Generate logs

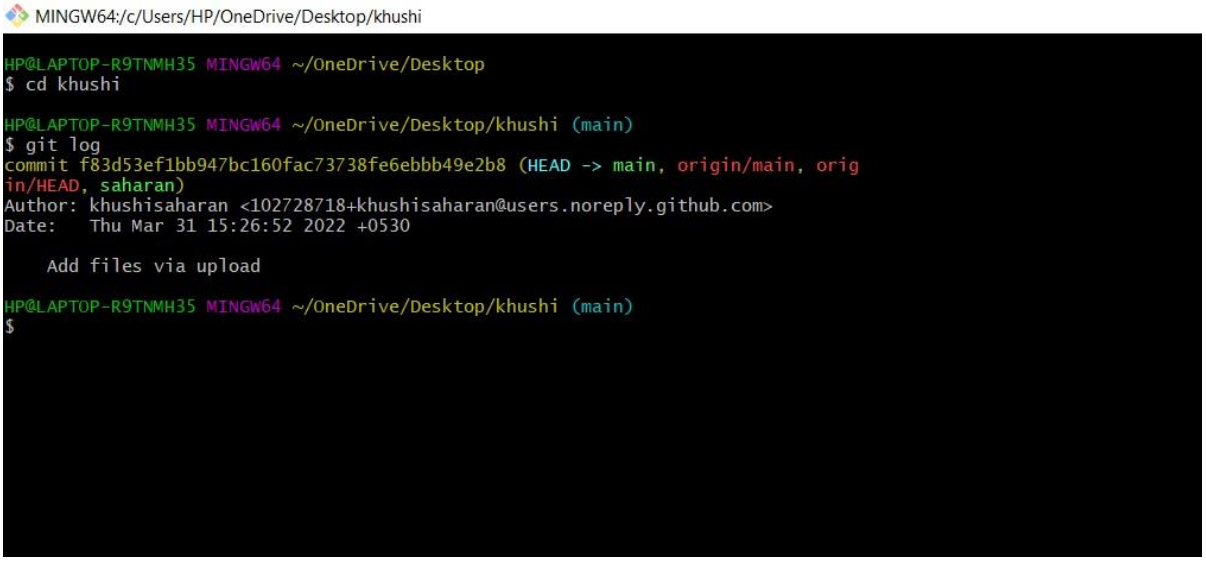
Theory:

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.



Task 4

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).

1. For creating a new branch: git branch “name of the branch”

2. To check how many branches we have: git branch

MINGW64:/c/Users/HP/OneDrive/Desktop/GIT1/khushi

```
HP@LAPTOP-R9TNMH35 MINGW64 ~/OneDrive/Desktop/GIT1
$ git clone https://github.com/khushisaharan/khushi.git
Cloning into 'khushi'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), 858.87 KiB | 661.00 KiB/s, done.

HP@LAPTOP-R9TNMH35 MINGW64 ~/OneDrive/Desktop/GIT1
$ cd khushi

HP@LAPTOP-R9TNMH35 MINGW64 ~/OneDrive/Desktop/GIT1/khushi (main)
$ git log
commit f83d53ef1bb947bc160fac73738fe6ebbb49e2b8 (HEAD -> main, origin/main, orig
in/HEAD)
Author: khushisaharan <102728718+khushisaharan@users.noreply.github.com>
Date:   Thu Mar 31 15:26:52 2022 +0530

    Add files via upload

HP@LAPTOP-R9TNMH35 MINGW64 ~/OneDrive/Desktop/GIT1/khushi (main)
$ git branch saharan

HP@LAPTOP-R9TNMH35 MINGW64 ~/OneDrive/Desktop/GIT1/khushi (main)
$ saharan
bash: saharan: command not found

HP@LAPTOP-R9TNMH35 MINGW64 ~/OneDrive/Desktop/GIT1/khushi (main)
$ git branch saharan
fatal: a branch named 'saharan' already exists

HP@LAPTOP-R9TNMH35 MINGW64 ~/OneDrive/Desktop/GIT1/khushi (main)
$
```

Task 5

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.