

**Name: Aashima Mahajan**  
**Roll No: 2110990021**  
**Course:Source Code Management**  
**Department:CSE**

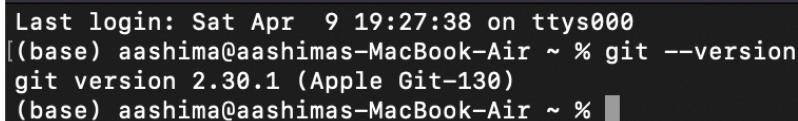


## **Aim : Setting up Git Client**

### **Theory**

A version control system, or VCS, tracks the history of changes as people and teams collaborate on projects together. As developers make changes to the project, any earlier version of the project can be recovered at any time.

### **Download Git for Mac**

A terminal window screenshot with a dark background. The text shows a login session on a Mac, followed by a command to check the git version. The output displays the git version as 2.30.1 (Apple Git-130).

```
Last login: Sat Apr 9 19:27:38 on ttys000
[(base) aashima@aashimas-MacBook-Air ~ % git --version
git version 2.30.1 (Apple Git-130)
(base) aashima@aashimas-MacBook-Air ~ %
```

- ❖ On Mac git is preinstalled
- ❖ Check by running git — version on terminal

## **Aim : Setting up GitHub Account**

### **Theory**

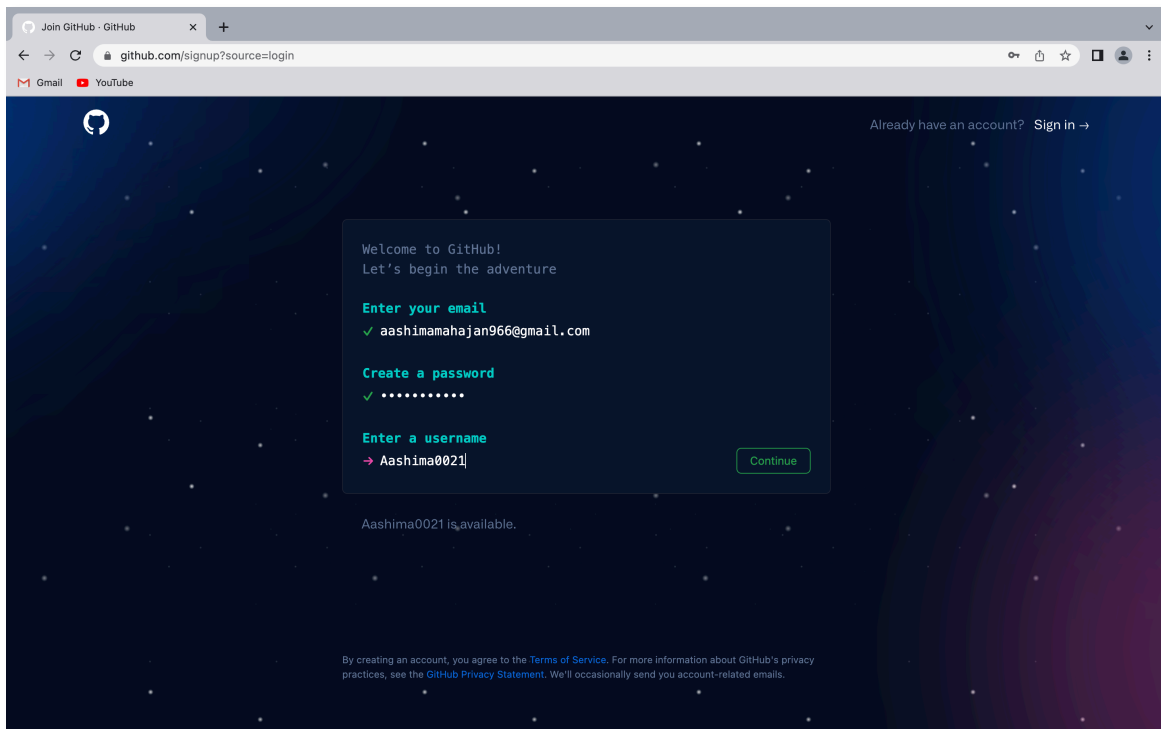
GitHub is a website and cloud-based service (client) that helps an individual or a developers to store and manage their code. We can also track as well as control changes to our or public code. GitHub offers user accounts for individuals and organisations for teams of people working together.

### **Procedure**

1. Google (any search engine)

Search for git-hub or (<https://github.com/signup>).

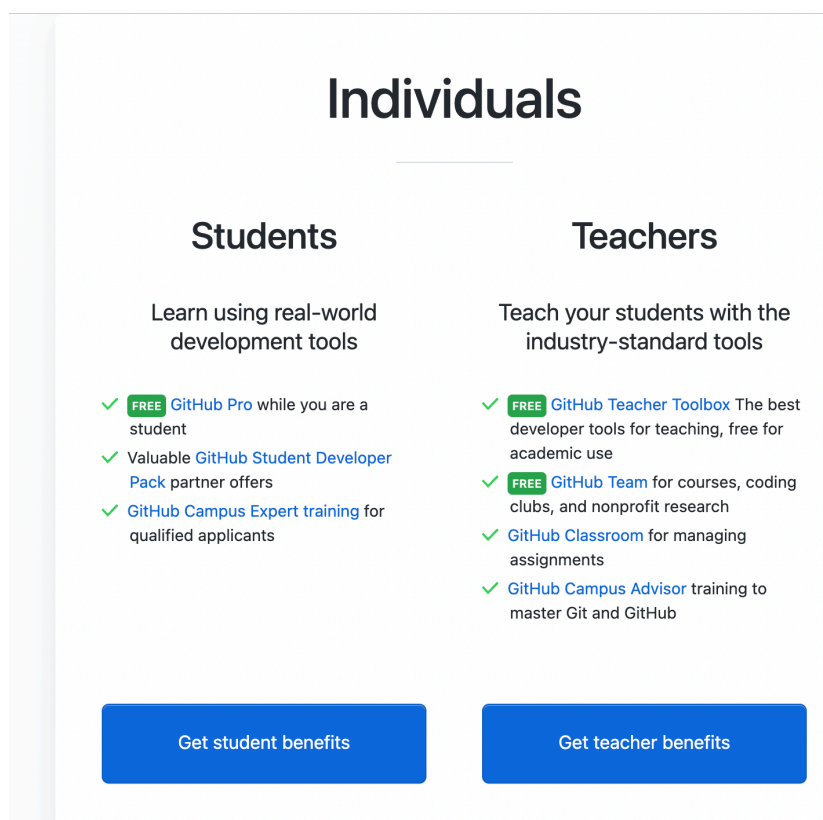
**2. Enter your personal details.** In addition to creating a username and entering an email address, you'll also have to create a password. Your password must be at least 15 characters in length *or* at least 8 characters with at least one number and lowercase letter.



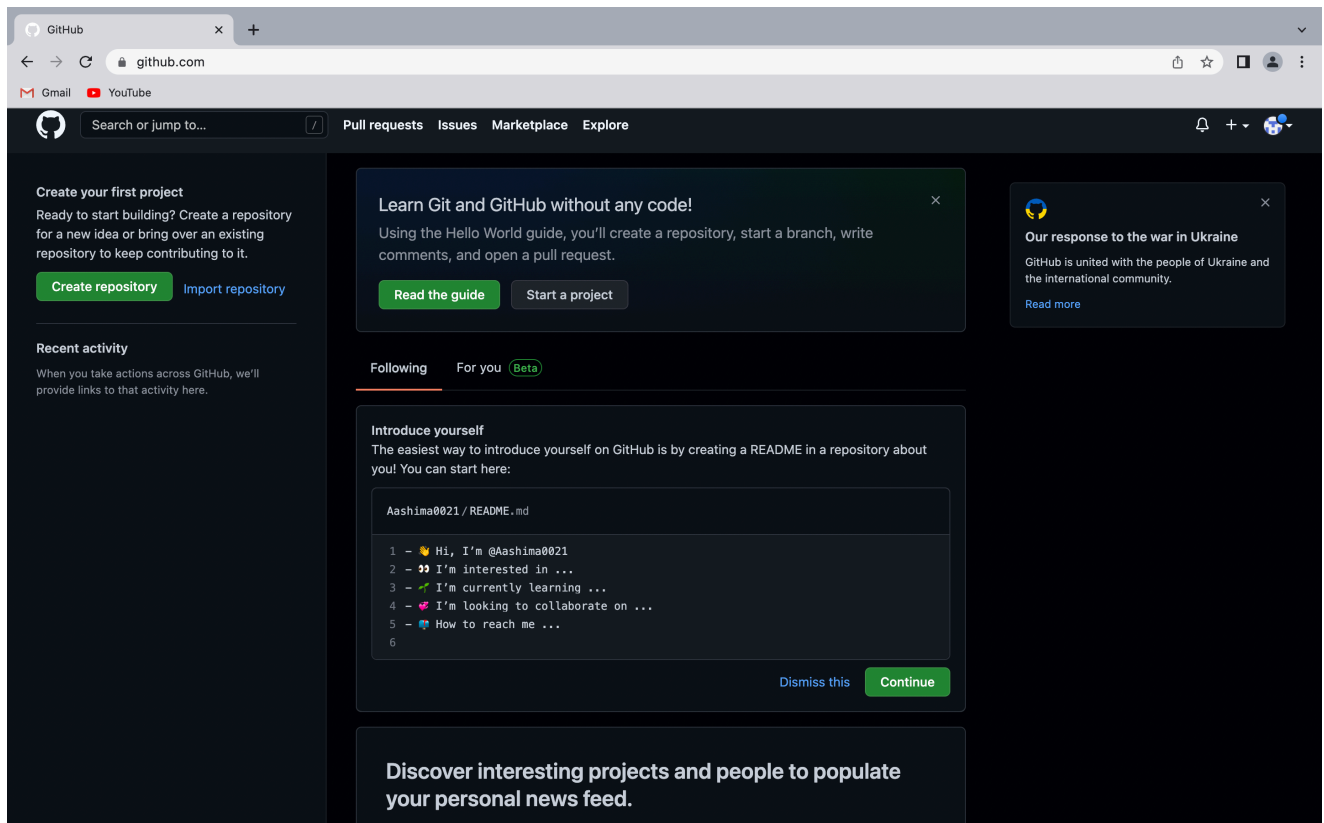
Click on the continue button to create an Account.

**3. Complete the CAPTCHA puzzle.** The instructions vary by puzzle, so just follow the on-screen instructions to confirm that you are a human.

**4. Choose the plan which you want to select and continue.**



## 5.Interface of GitHub



## Aim : Program to Generate Log

### Theory

Git log is a utility tool to review and read a history of everything that happens to a repository. Multiple options can be used with a git log to make history more specific. Generally, the git log is a record of commits.

### Procedure-

#### **git status**

The git status command displays the state of the working directory and the staging area.

```

Last login: Sat Apr  9 20:12:07 on ttys000
[(base) aashima@aashimas-MacBook-Air ~ % cd Desktop
[(base) aashima@aashimas-MacBook-Air Desktop % cd SCM
[(base) aashima@aashimas-MacBook-Air SCM % touch first.cpp
[(base) aashima@aashimas-MacBook-Air SCM % git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/aashima/Desktop/SCM/.git/
[(base) aashima@aashimas-MacBook-Air SCM % git status
On branch master

No commits yet

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

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

```

The **git add** command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit. However, **git add** doesn't really affect the repository in any significant way—changes are not actually recorded until you run [git commit](#).

## Git Commit

Adding commits keep track of our progress and changes as we work. Git considers each commit change point or "save point". It is a point in the project you can go back to if you find a bug, or want to make a change.



When we commit, we should **always** include a **message**.

By adding clear messages to each commit, it is easy for yourself (and others) to see what has changed and when.

```
[(base) aashima@aashimas-MacBook-Air SCM % git .add
git: '.add' is not a git command. See 'git --help'.

The most similar command is
  add
[(base) aashima@aashimas-MacBook-Air SCM % git add .
[(base) aashima@aashimas-MacBook-Air SCM % git status
On branch master

No commits yet

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

[(base) aashima@aashimas-MacBook-Air SCM % git commit -m "this file contains all the changes"
[master (root-commit) 77ee262] this file contains all the changes
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 first.cpp
[(base) aashima@aashimas-MacBook-Air SCM % git status
On branch master
nothing to commit, working tree clean
[(base) aashima@aashimas-MacBook-Air SCM % git log
commit 77ee2624bc6f2d7219ee7080aa1e1d09d2b1e038 (HEAD -> master)
Author: Aashima-123 <aashima0021.be21@chitkara.edu.in>
Date:   Sat Apr 9 21:23:52 2022 +0530

    this file contains all the changes
(base) aashima@aashimas-MacBook-Air SCM %
```

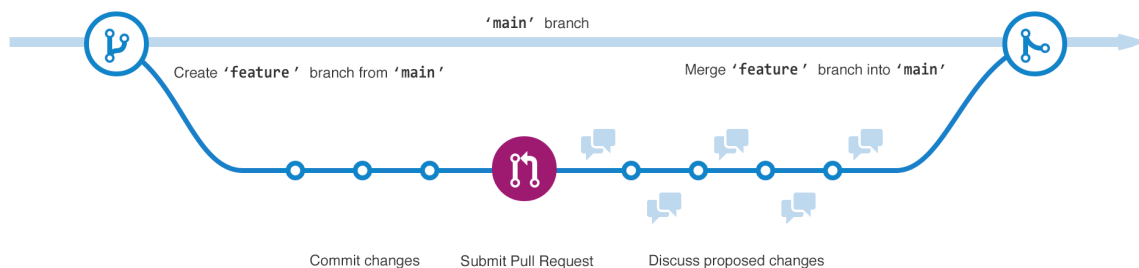
## The git log Command

The git log command shows a list of all the commits made to a repository.

# Aim: Create and Visualise Branches

## Theory

Branching lets you have different versions of a repository at one time. The main branch in git is called as 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 file which are created in branch are not shown in master branch. We can also merge both the parent (master) and child (other branches)



- By default, the branch that is created automatically is called *master* or *main*.

```
[(base) aashima@aashimas-MacBook-Air SCM % git branch
* master
```

## Procedure:



◆For creating a new branch -> git branch  
<branch name>

◆For checking all the branches -> git branch

```
[(base) aashima@aashimas-MacBook-Air SCM % git branch page
[(base) aashima@aashimas-MacBook-Air SCM % git branch
* master
page
```

◆To change the present working branch -> git  
checkout <branch name>

```
[(base) aashima@aashimas-MacBook-Air SCM % git checkout page
Switched to branch 'page'
[(base) aashima@aashimas-MacBook-Air SCM % git branch
master
* page
(base) aashima@aashimas-MacBook-Air SCM %
```

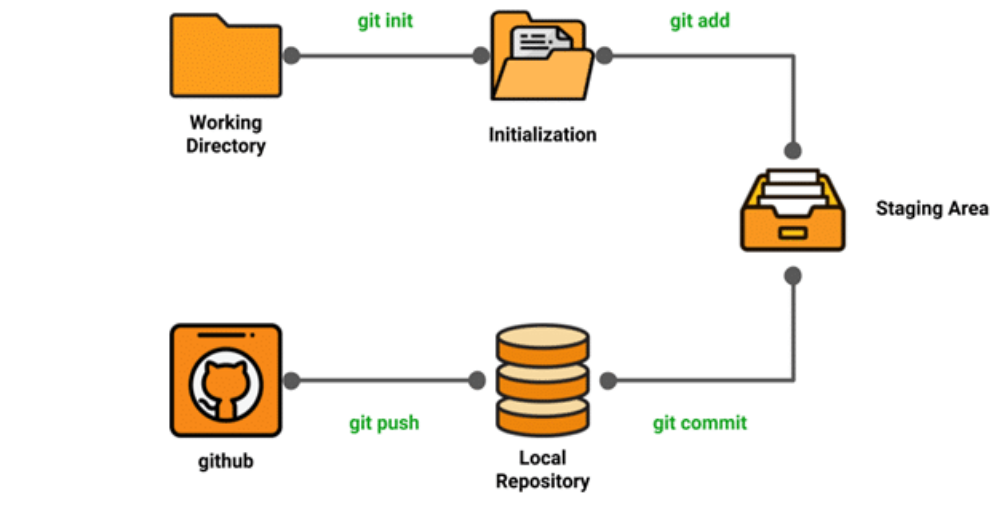
# Aim : Git life cycle description

## Theory

Git is used in our day-to-day work, we use git for keeping a track of our files, working in a collaboration with our team, to go back to our previous code versions if we face some error. Git helps us in many ways. Let us look at the Life Cycle that git has and understand more about its life cycle.

When a directory is made a git repository, there are mainly 3 states which make the essence of Git Version Control System. The three states are –

- ❖ Working Directory
- ❖ Staging Area
- ❖ Git Directory

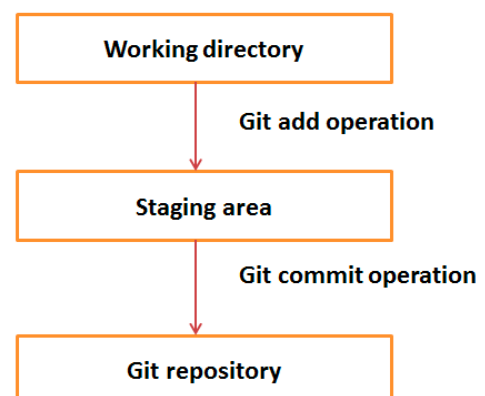


## 1. Working Directory

Whenever we want to initialise our local project directory to make it a git repository, we use the ***git init*** command. After this command, git becomes aware of the files in the project although it doesn't track the files yet. The files are further tracked in the staging area.

## 2. Staging Area

Now, to track the different versions of our files we use the command ***git add***. We can term a staging area as a place where different versions of our files are stored. ***git add*** command copies the version of your file from your working directory to the staging area. We can, however, choose which files we need to add to the staging area because in our working directory there are some files that we don't want to get tracked, examples include node modules, env files, temporary files, etc. Indexing in Git is the one that helps Git in understanding which files need to be added or sent. You can find your staging area in the ***.git*** folder inside the ***index*** file.



### **3. Git Directory**

Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the ***git commit*** command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about. Git preserves the information or the metadata of the files that were committed in a Git Directory which helps Git in tracking files and basically it preserves the photocopy of the committed files. Commit also stores the name of the author who did the commit, files that are committed, and the date at which they are committed along with the commit message.

