***SCM***

***Project 1***



By-Adarsh Bhardwaj

Roll No.-2110990074

***topics:***

* Setting up of Git Client.
* Setting up a GitHub account.
* Generate logs.
* Branches.
* Git lifecycle description.

**Introduction:**

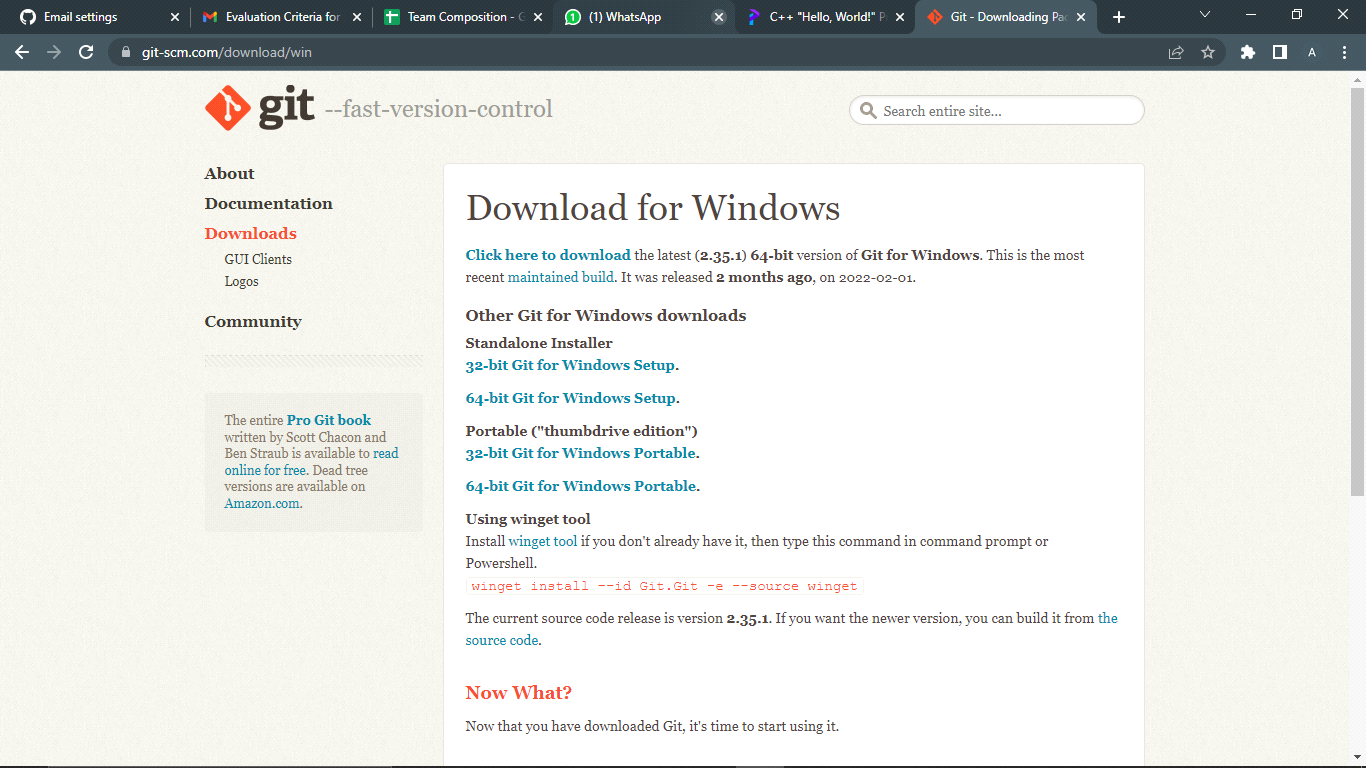
Git is a source code management technology used by DevOps. Git is a piece of software that allows you to track changes in any group of files. It is a free and open-source version control system that may be used to efficiently manage small to big projects.

Git is a version control system that allows numerous developers to collaborate on non-linear development projects.

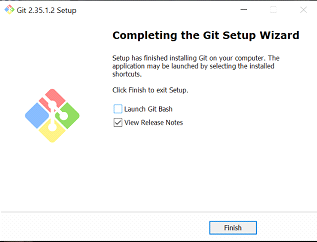
Git is an example of a distributed version control system (DVCS) (hence Distributed Version Control System).

**Installing git:**

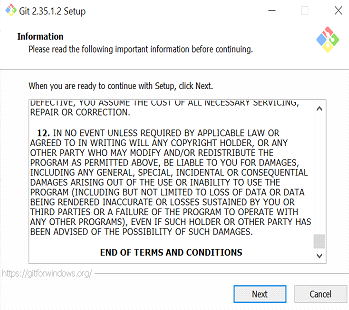
The most official build is available for download on the Git website. Just go to <https://git-scm.com/download/win> and the download will start automatically. Follow the steps listed further to proceed with the installation.



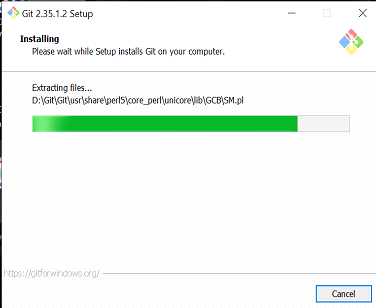
next step:



next step:



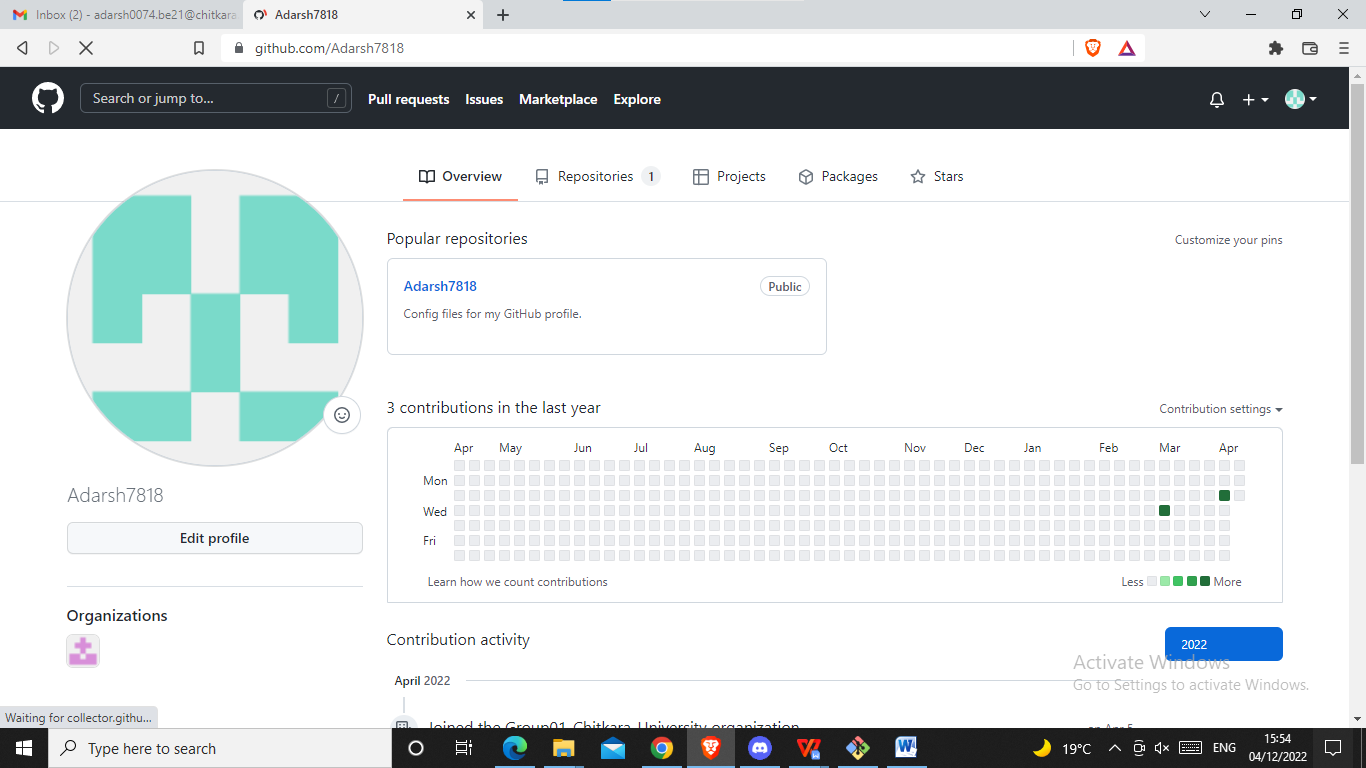
next step:

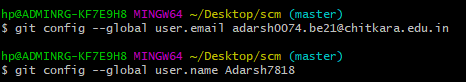


next step:

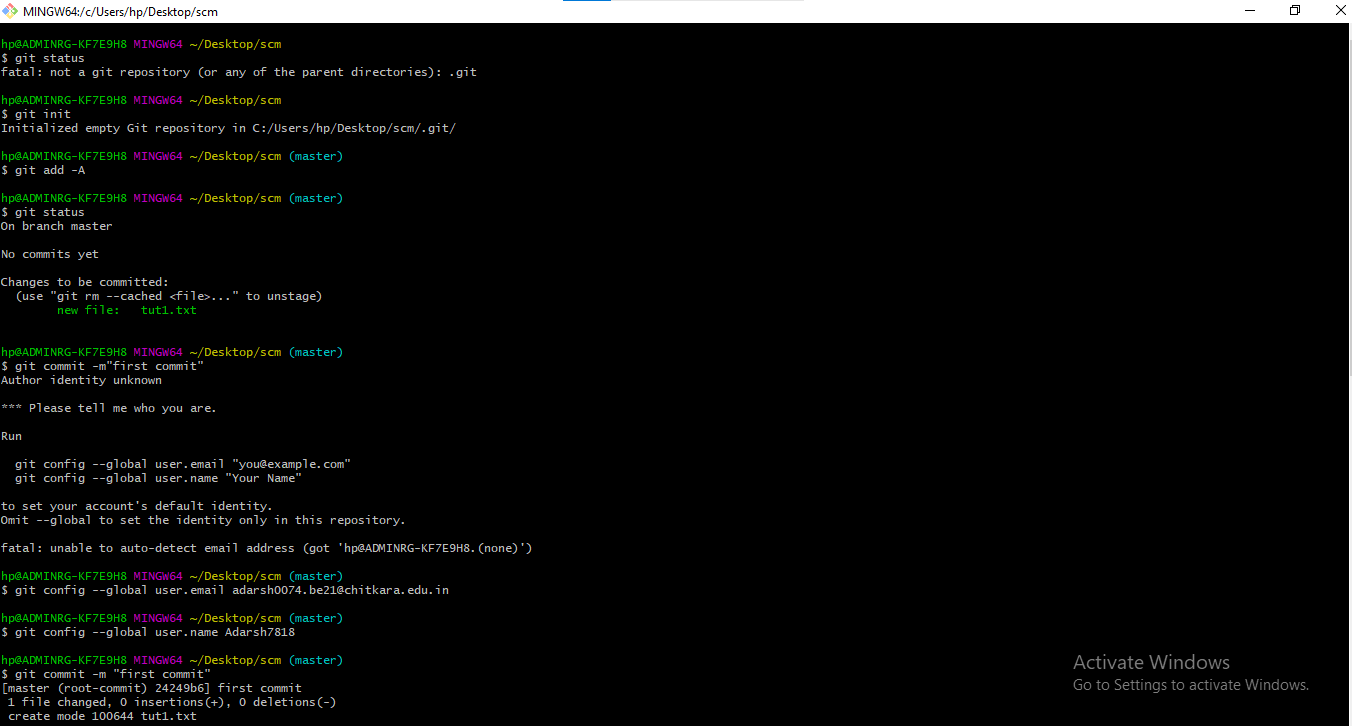
**Setting up of github account:**

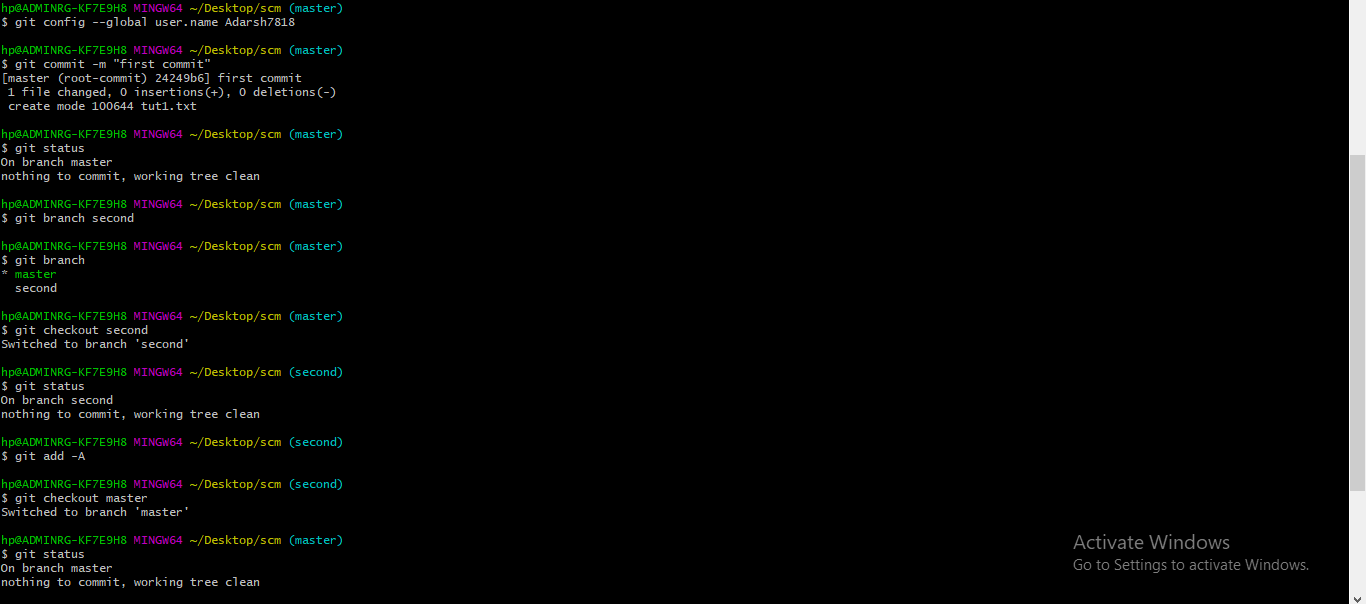
You can set up a GitHub account by going onto the website <https://github.com/> and choosing the ‘Sign Up” option. Enter the required details and you are done with creating account on GitHub.



**Setting user name and email:** 

**Various Commands:**





**mkdir**-- It is used for creating a directory using Git Bash

**cd command**-- Both Bash and Windows console host have a cd command. cd is an acronym for 'Change Directory'. cd is invoked with an appended directory name. Executing cd will change the terminal sessions current working directory to the passed directory argument.

**pwd**--The Bash command pwd is used to print the 'present working directory'. This is the folder or path that the current Bash session resides in.

**ls**--The Bash command ls is used to 'list' contents of the current working directory.

**git status**--The main tool you use to determine which files are in which state is the git status command. the command tells you which branch you’re on and informs you that it has not diverged from the same branch on the server.

**git add**-- In order to begin tracking a new file, you use the command git add. git add is a multipurpose command — you use it to begin tracking new files, to stage files, and to do other things like marking merge-conflicted files as resolved. It may be helpful to think of it more as “add precisely this content to the next commit” rather than “add this file to the project”.

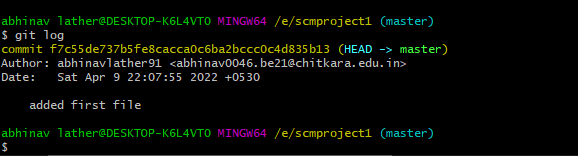
**git commit**-- The git commit command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as “safe” versions of a project—Git will never change them unless you explicitly ask it to. Prior to the execution of git commit, The git add command is used to promote or 'stage' changes to the project that will be stored in a commit. These two commands git commit and git add are two of the most frequently used.

**git push**-- When you have your project at a point that you want to share, you have to push it upstream. The command for this is simple: git push . If you want to push your master branch to your origin server (again, cloning generally sets up both of those names for you automatically), then you can run this to push any commits you’ve done back up to the server: $

git push origin master

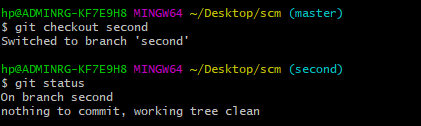
**Generating Logs:**

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.



**Git Branching:**

Branches allow you to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository. You always create a branch from an existing branch. Typically, you might create a new branch from the default branch of your repository.



**Git Lifecycle Description:**

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:

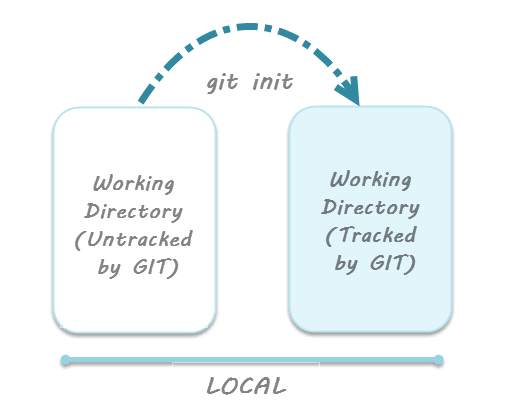
* Working directory
* Staging area
* Git directory

These stages are the essence of Git. You get great flexibility in tracking the files due to these stages that files can reside in under Git. Let's understand each of these states one by one.

**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.



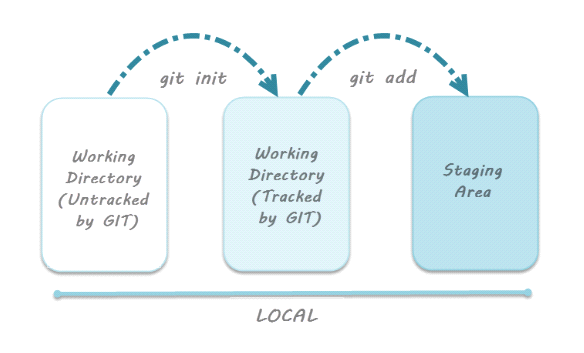
For sake of further discussion, let's assume that this directory is now tracked by Git. That is we've created a Git repository in this existing project directory. So, as discussed in the tutorial on Creation of Git Repository, a hidden .git folder is initialized therein. In this state, Git is just aware of the files in the project. It doesn't track the files yet. To track the files, we've to commit these files by first adding the files to the staging area. This brings us to the next state in Git life-cycle.

**Staging Area**

While we're in the working directory, we select the files that have to be tracked by Git. Why do we need to this? Why don't we track everything in the project? That's because 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.

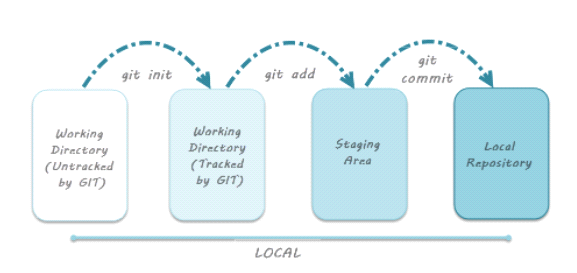
It's important to make a quick note of the term called indexing here. Indexing is the process of adding files to the staging area. In other words, index constitutes of files added to the staging area. This term will be explained again in the coming tutorial on Git terminologies.



**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 (names of files committed, date and time of commit, author of commit, commit message) is stored in the Git directory.

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



**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.

