What is a version control system?

A version control system (VCS), also known as a source control system or revision control system, is a software tool or system that helps developers and teams manage changes to their code or other sets of files over time. The primary purpose of a version control system is to track and manage different versions of files, enabling multiple contributors to collaborate on projects efficiently.

Here are the key aspects and benefits of version control systems

History tracking

Branching

Merging

Merge conflicts resolution

Traceability

Backup and recovery of the previous version

Code review

A line of dots with a point in the middle

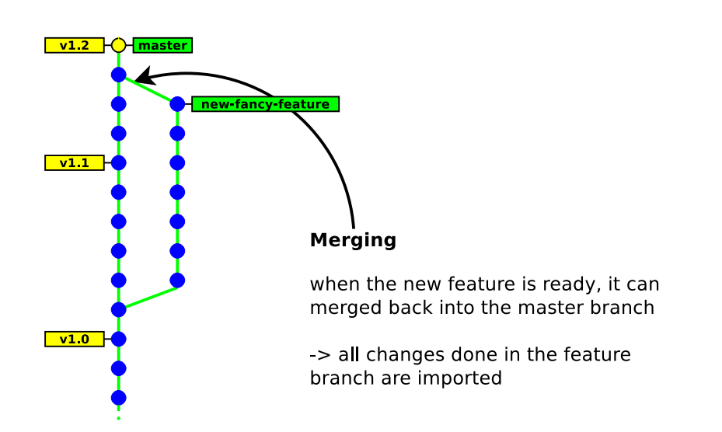
Description automatically generated with medium confidenceA diagram of a diagram of a diagram

Description automatically generatedA diagram of a tree

Description automatically generatedA diagram of a software

Description automatically generatedA diagram of a new fancy branch

Description automatically generated



What is git?

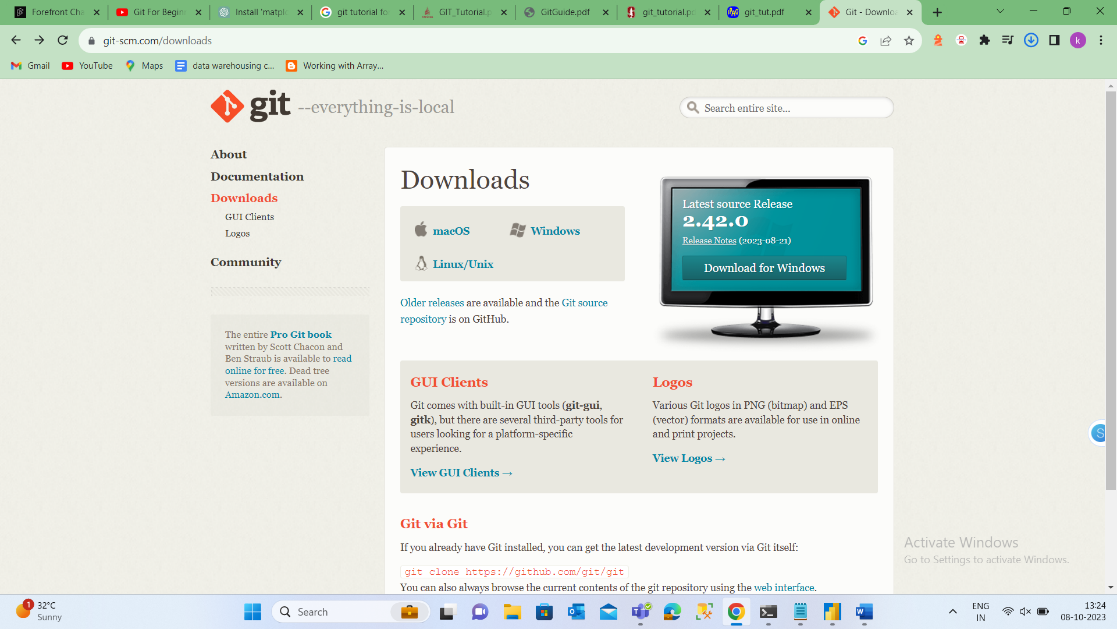
Git is a tool that programmers use to upload their code to servers online so they can write a program from many different computers, create different versions of their programs, revert their changes if they upload broken code, as well as collaborating with other team members and many other reasons.

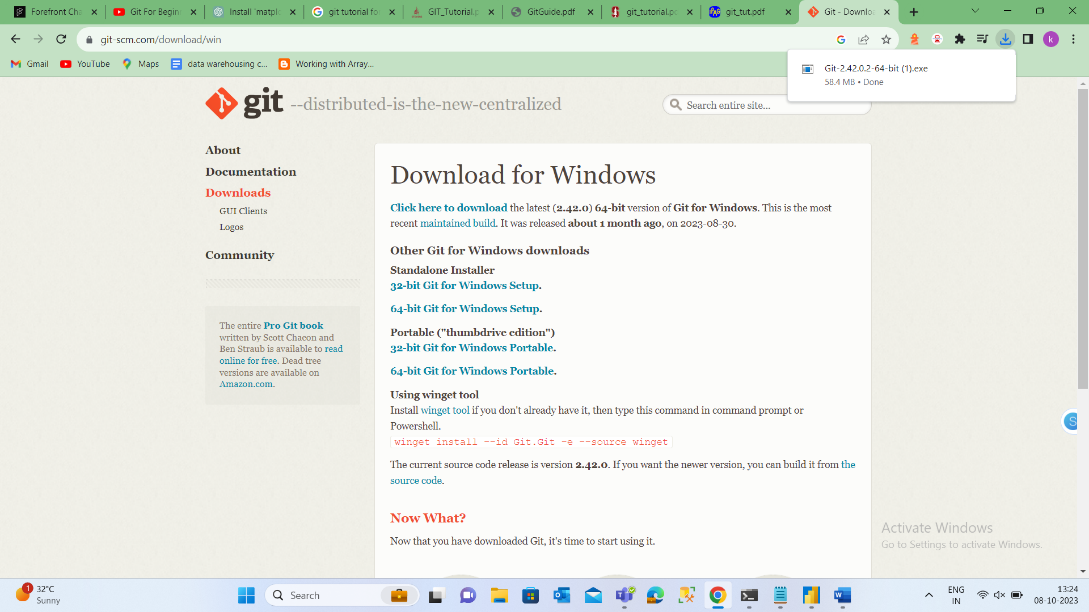
GitHub and GitLab are online servers and websites that are used to store code with Git online.

Code is uploaded to a place called a “repository,” or a repo Each repository can have many different “branches” that hold different versions of a program or, in our case, different assignments. A Git repository that you are working on may have many different branches, but you can only have one of those branches downloaded to your computer at a time.

Git setup

1. Go to the link http://git-scm.com/download/win and download the git package for windows





1. After downloading the git package run the git package
2. After installing the package you will have “Git GUI” and “Git Bash” applications on your computer.
3. To check the successful installation use the command git –version in CMD prompt.

Setting git configuration

Open the git bash and enter the following command to config the user

A black screen with white text

Description automatically generated

There are three stages in git

Local repository

Working directory

A screenshot of a computer screen

Description automatically generated

How to initialize git

1. Open git bash navigate to the required folder using   
   cd requiredpath
2. After navigating to the path enter git init
3. To check the installation use ls -a this will the give the list of files and directories and in that you will find a .git folder it indicates git is installed
4. Enter the command git status to find the list of tracked untracked and modified files in the working directory
5. Add the files present in the working directory to the staging are by using git add <file1> <file2> ...
6. After adding to staging area use git commit to push the file to the local repository.

These are the following uses in the git bash

|  |
| --- |
| Git cheat sheet |

|  |  |
| --- | --- |
| configuration | |
| git config --global user.name “[firstname lastname]” | set a name that is identifiable for credit when review version history |
| git config --global user.email “[valid-email]” | set an email address that will be associated with each history marker |
| git config --global color.ui auto | set automatic command line coloring for Git for easy reviewing |
|  |  |
| set and init | |
| git init | initialize an existing directory as a Git repository |
| git clone [url] | retrieve an entire repository from a hosted location via URL |
| git status | show modified files in working directory, staged for your next commit |
| git add [file] | add a file as it looks now to your next commit (stage) |
| git reset [file] | unstage a file while retaining the changes in working directory |
| git diff | diff of what is changed but not staged |
| git diff --staged | diff of what is staged but not yet commited |
| git commit -m “[descriptive message]” | commit your staged content as a new commit snapshot to the local repositary |
|  |  |
| branch and merge | |
| git branch | list your branches. a \* will appear next to the currently active branch |
| git branch (branch name ) | create a new branch at the current commit |
| git checkout branch name | switch to another branch and check it out into your working directory |
| git merge (branch) | merge the specified branch’s history into the current one |
| git log | show all commits in the current branch’s history |