GIT

Git is a version control system that helps track changes in code and manage project versions. It's widely used in collaborative development and allows for effective tracking, branching, merging, and reverting of code.

It is used for:

- Tracking code changes
- Tracking who made changes
- Coding collaboration

GIT Commands

- To check if Git is properly installed:

  git --version
  git version 2.30.2.windows.1

- Configure Git - Set the name and email that will be attached to each commit:

  git config --global user.name "Your Name"
  git config --global user.email you@example.com

- To check the installation path of Git on your system

  git --exec-path

- To Opens the global Git configuration file in an editor. (Lets you view and edit settings that apply to all repositories for the current user)

  git config --global --edit

  For example, it might look like this:

  ```ini
  [user]
      name = Your Name
      email = your.email@example.com
  [core]
      editor = nano
  [color]
      ui = auto
  ```

- To make a new directory.

  mkdir myproject

- To change the current working directory.
  cd myproject

- To change the directory
  cd /c/Users/manisha.sharma/Desktop

- To display the files and directories in the current working directory
  ls

- To list all files and directories in the current directory, including hidden files
  ls –a
- To see all the available options for the specific command
  git *command* -help
- To see all possible commands
  git help --all
- To create new branch
  git branch dev
- To switch from one branch to another branch
  git checkout master
- To merge the branch to another branch
  git merge dev

Here are the basic steps to commit a project to Git

1. cd path/to/your/project # Navigate to the project directory
2. git init # Initialize the Git repository (if needed)
3. git add . # Stage all files for commit
4. git commit -m "Commit message" # Commit with a message
5. git remote add origin <URL> # Add remote repository (if pushing to remote)
6. git push -u origin main # Push to the remote repository

Stages in Git Workflow

1. **Working Directory (Untracked/Modified)**

    a. This is the initial stage, where files exist on your computer but are either **untracked** (new files Git doesn't know about yet) or **modified** (files that have been changed since the last commit).
    b. When you edit files in the working directory, these changes are not yet part of the Git version history.
2. **Staging Area (Indexed)**
    a. When you run git add <filename> (or git add . to add all changes), Git moves the selected files from the working directory to the staging area.
    b. The staging area stores a snapshot of the changes, which allows you to prepare a precise set of modifications for the next commit. You can add or remove files from this area as needed before finalizing a commit.
3. **Repository (Committed)**
    a. When you run git commit -m "message", Git takes the files in the staging area and saves them as a **new commit** in the local repository.
    b. Once committed, these changes become part of Git history and are saved in your repository, where they are safe from further modifications until you create new changes.

Example of the Flow Through Stages

1. **Working Directory**: Edit files in your project.
2. **Staging Area**: Stage files with git add <filename> or git add.(to stage all files), preparing them for commit.
3. **Repository**: Commit staged files to the repository with git commit -m "message".

Master (or Main) Branch

- The master branch is like the **main version of the app everyone uses**—the official release. It’s stable, complete, and only includes fully tested features.

Dev (Development) Branch

- The dev branch is a **working version where new features and updates are added and tested**. It’s separate from master so that changes won’t affect the main app until they’re fully ready.

Branch

- Branch is a new/separate version of the main repository.

Example: Imagine Instagram. The master branch is what all users see in the app store—it's the official version that's safe to use. When Instagram developers are confident a feature is perfect, they add it to master, making it part of the official app everyone uses.

Suppose Instagram wants to add a "Dark Mode." They create and test "Dark Mode" in dev, trying different styles, fixing bugs, and ensuring it works perfectly. Multiple developers can work on dev together, testing and improving without affecting the official master app. Once "Dark Mode" is polished, it's moved from dev to master, so it becomes part of the official app users can enjoy.