Melody Abapo
Github Assignment

Git is a powerful tool for managing and tracking code changes, making collaboration smoother and more efficient. Git has a repository, a database storing the complete history of a project. Every change is recorded as a commit. A commit is the state of the project at a specific moment. It includes the modified files, a timestamp, the author, and a short message explaining the changes. This detailed history is extremely useful for tracking progress, understanding the evolution of the code, and easily reverting to earlier versions if a bug is introduced or a feature needs to be rolled back.

Git workflows determine how developers interact with the repository and each other's code. Different workflows accomodate for different project sizes and team dynamics. For small projects with a few developers, a simple workflow where everyone works directly on the main branch would be sufficient. However, as projects grow, this approach can quickly become chaotic.

Feature Branch Workflows help developers to create separate branches for each new feature or bug fix. This isolates changes, allowing developers to work independently without affecting the main branch. Once a feature is complete and tested, it can be merged back into the main branch. Gitflow defines specific branch types for development, releases, and hotfixes. This provides a more structured approach for managing complex projects with multiple release cycles. Forking Workflows, often used in open-source projects, allow developers to create personal copies of the repository, make changes, and then submit those changes back to the original project maintainers through pull requests.

Pushing and pulling are how you keep your local copy of the repository synchronized with the remote repository. Pushing uploads your local commits to the remote repository, making your changes available to others. Pulling downloads changes from the remote repository to your local copy, keeping you up-to-date with everyone else's work.

Merging is the process of combining changes from one branch into another. Git can usually automatically merge changes without any issues. However, sometimes two developers might modify the same lines of code in different branches. This creates a merge conflict, which requires manual intervention. Developers will need to look at the conflicting code sections, decide which changes to keep or combine, and then mark the conflict as resolved before completing the merge. Merge conflicts are an important mechanism for preventing accidental code loss and ensuring that everyone's changes are integrated correctly.

Git workflow depends on the project's size and complexity. Simpler workflows work well for smaller projects. Larger, more complex projects work well with the structure and organization of Gitflow or Forking Workflows.

Learn Git workflow

- https://www.atlassian.com/git/tutorials/comparing-workflows
- Read the site, write a min. 1-page single space essay about
    - git, gitworkflow, commits, pushes, pulls, merges, merge conflicts, repositories