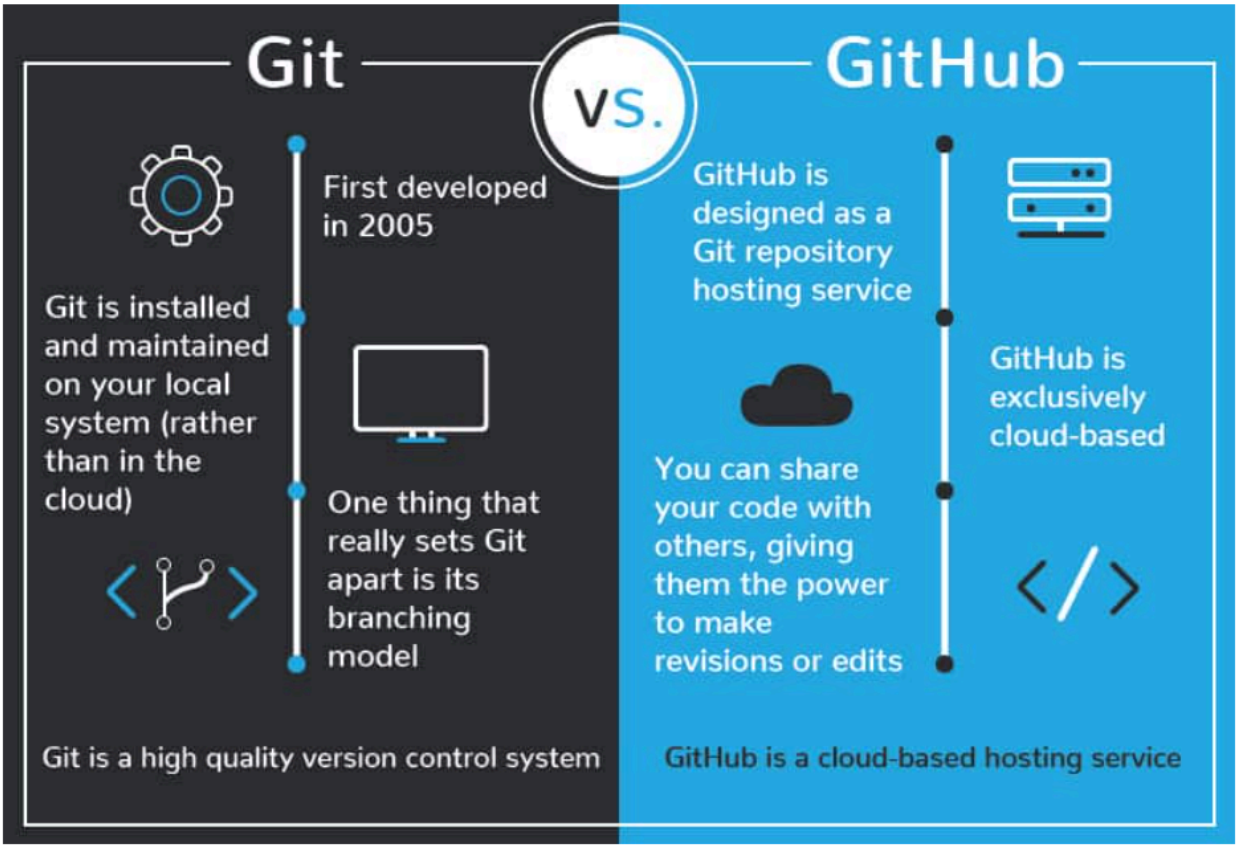
ALL ABOUT GIT

- What and Why Git?

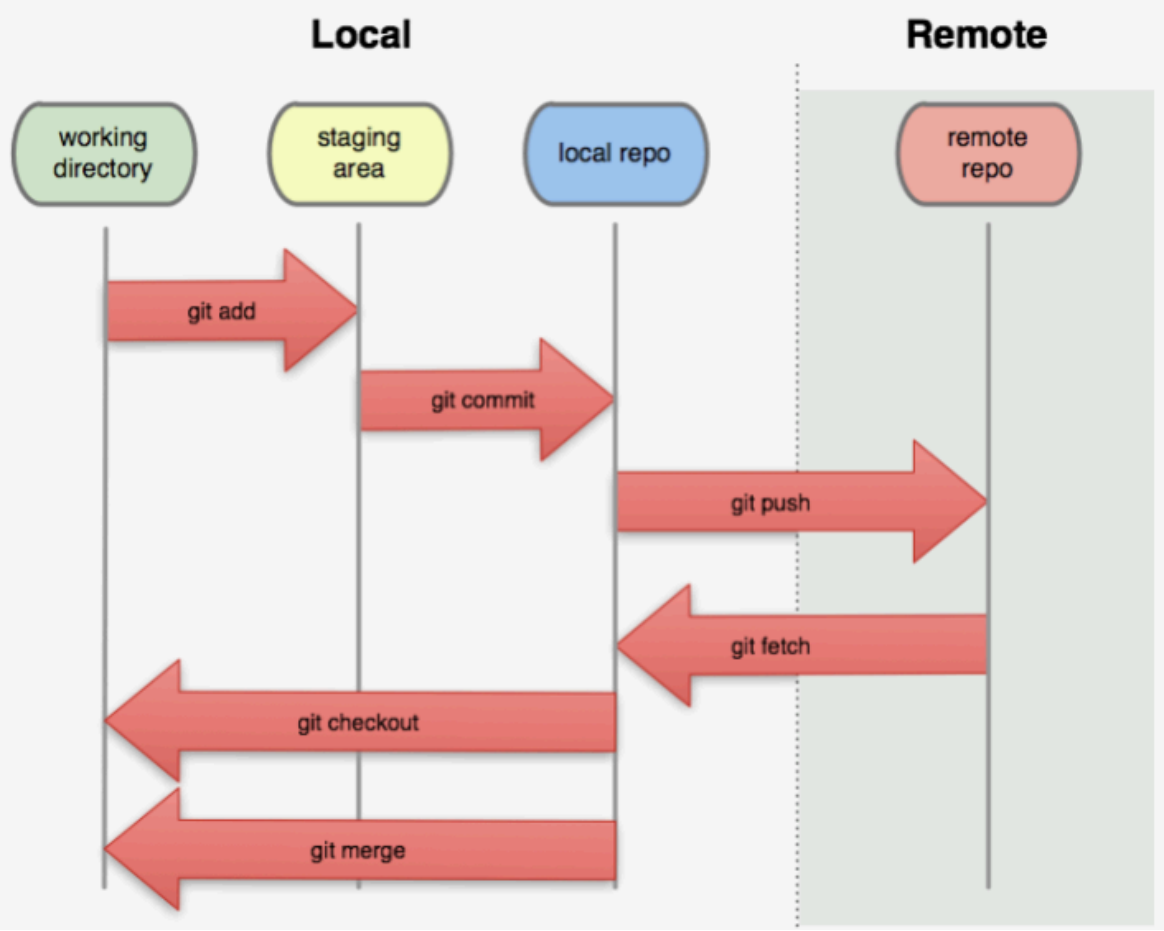
Git is a distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers who are collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.
Programming languages: Python, C, C++, Perl, Tcl, Shell script

- Git vs GitHub

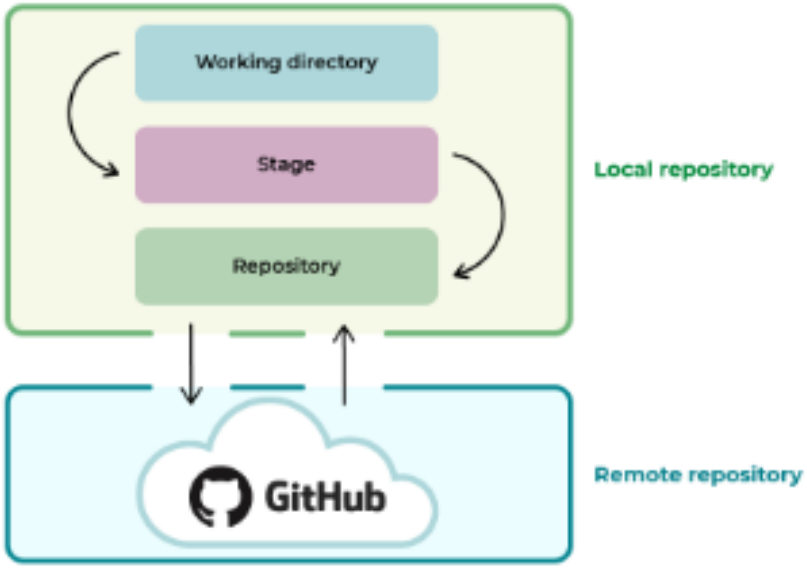


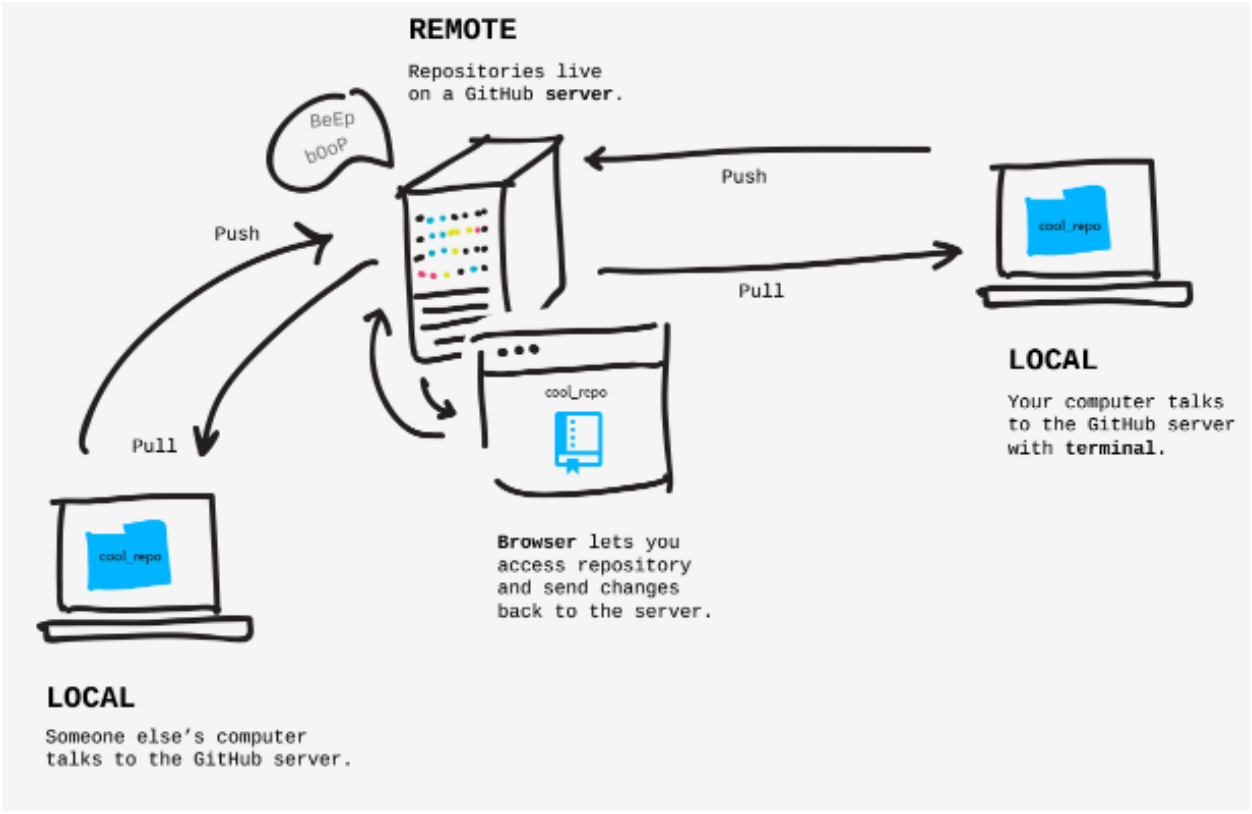
Made by Sejal Randive

- Git commands and architecture



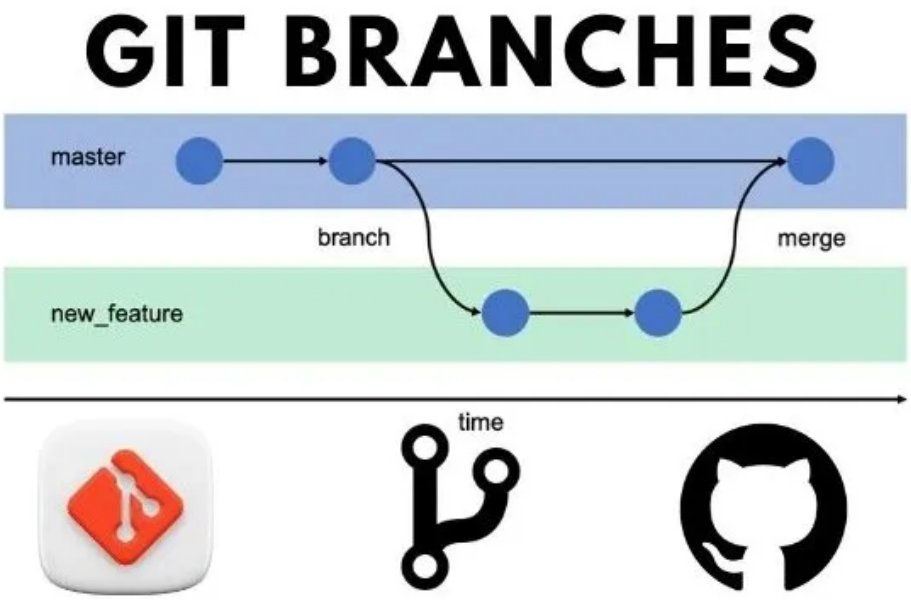
- Local and Remote Repository (git cloud)





Made by Sejal Randive

- Git Branches



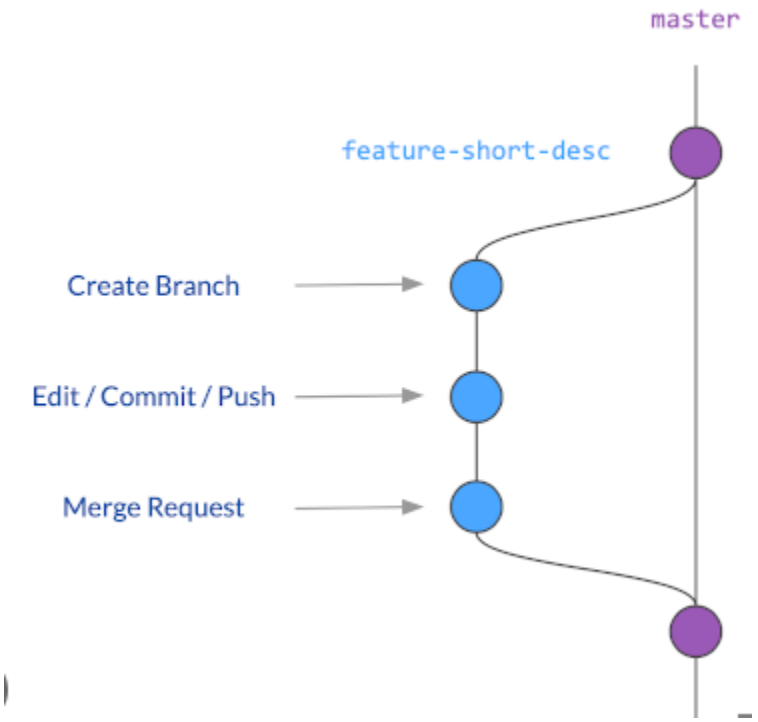
Git branches are like parallel universes where you can work on different aspects of your project simultaneously. They provide a way to separate your work into distinct lines of development, allowing you to make changes and experiment without affecting the main or stable version of your code.
It's a copy of the main code. It makes a replica, we work on it and then merge it back to the main code again.
At Delphi, we work on the branches of our own name only.

- Why Use Branches: The Benefits of Using Branches in Git

Using Git branches offers several significant advantages:

1. **Isolation**: Branches provide a safe space to work on specific tasks. You can make changes without worrying about affecting the main project until you're ready.
2. **Collaboration**: In a team, each member can work on their own branches simultaneously. This promotes collaboration without stepping on each other's toes.

3. **Experimentation**: You can create branches to experiment with new ideas. If an experiment doesn't work out, you can simply discard the branch without affecting the main project.
4. **Bug Fixing**: Branches are perfect for isolating and fixing bugs. You can create a branch to address a specific issue, test the fix, and then merge it back into the main project when it's verified.
5. **Versioning**: Branches enable you to maintain multiple versions of your project simultaneously. For example, you might have one branch for the current stable release and another for a future version with new features.



- Resolving Merge Conflicts:

  Handling Conflicts: Merge conflicts can happen during any merge scenario when Git cannot automatically reconcile changes between branches. Conflicts occur when the same part of a file has been modified differently in both branches. We can manually solve these conflicts by sitting together and working on that same file.