**INTRODUCTION TO GIT AND GIT HUB**

**Difference between Git and Git hub**

GIT:

* A distributed version control system (DVCS) primarily designed for managing source code and version history. It operates using various protocols to facilitate communication and data transfer between repositories.
* the Git protocol itself refers to a specific network protocol used for transferring Git-specific data between repositories.

GIT HUB:

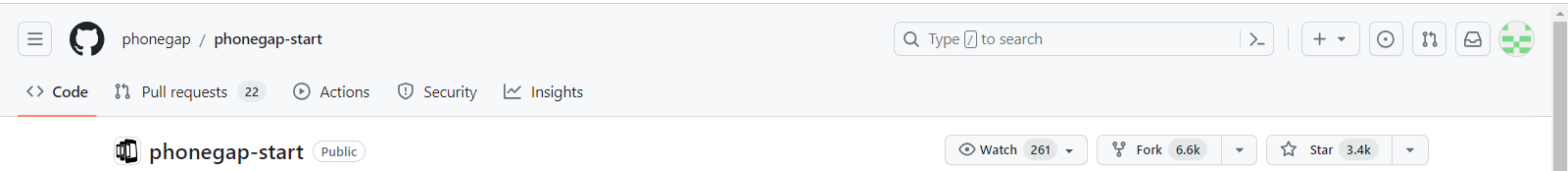
* Web-based platform that utilizes Git for version control, allowing users to host their Git repositories remotely and collaborate with others through various features and tools.

**GIT-HUB**

**Basic terms:**

Repository (Repo): A repository is a folder or storage space where your project and its entire history are stored. It can exist either locally on your computer or remotely on a hosting service like GitHub, GitLab, or Bitbucket.

Branch: A branch is a parallel version of the code. It allows developers to work on separate changes without affecting the main codebase.



1.Search bar: Type the term you want to search for into the search bar. You can search for repositories by their name, code snippets, specific file names, issues, usernames, or other relevant information.

2.Create new file:It is used to create a new repository,import repository and also to start a new project.

3.Issues:We can search about the issues in any repository.

4.Pull requests: a pull request is a request to merge changes from one branch (typically a feature branch) into another (like the main branch). It allows others to review the code before merging.

5.Notifications:To review about the incoming notifications.

* Open, Closed, and Assigning: Issues can be open, closed, or assigned. Open issues are actively being discussed or worked on, while closed issues are resolved.

4.Pull requests: Request to merge changes from one branch (typically a feature branch) into another (like the main branch). It allows others to review the code before merging.

6.Watch:Show the Github users who have asked to be notified of activity in a repository, but have not become collaborators.

7.Fork: Creating a personal copy of someone else's repository into your account. It allows you to freely experiment with changes without affecting the original repository.

8.Star:To rate or save a repository.

**Deleting a repository**

**Cloning**

Git will start cloning the repository from the remote location to your local machine. It will create a new directory with the repository name and download all the files and history.

**Git basic commands**

1. git config: Used to set up Git configurations, including user details, aliases, etc.
2. git init: Initialize a new Git repository in the current directory.
3. git clone [repository\_url]: Clone an existing repository into a new directory.
4. git add: Add file changes to the staging area.
5. git commit: Record staged changes with a commit message.
6. git status: Check the current status of the working directory and staging area.
7. git branch: List, create, or delete branches in the repository.
   * 1. git branch: List all branches.
     2. git branch <branch\_name>: Create a new branch.
     3. git branch -d <branch\_name>: Delete a branch.
8. git checkout: Switch between branches or checkout files from a specific commit.
9. git merge: Merge changes from one branch to another.
10. git push: Push local changes to a remote repository.
11. git pull: Fetch changes from a remote repository and merge them into the current branch.

**Concepts on GITHUB, GitLab And BitBucket**

GitHub, GitLab, and Bitbucket are popular web-based Git repository hosting services that offer version control and collaboration tools for managing Git repositories.

**GitHub:**

-Overview:GitHub is one of the largest and most widely used platforms for hosting Git repositories.

- Key Features:

- Collaboration: Offers robust collaboration tools, including issue tracking, pull requests, code review, project boards, and wikis.

- Community: GitHub has a vast community of developers, making it a hub for open source projects.

- Integration: Supports various integrations and has a vast marketplace of apps and services that complement its functionalities.

- Use Cases: Popular for open source projects, personal projects, team collaborations, and for showcasing coding skills via public repositories.

**GitLab**:

- Overview: GitLab is a comprehensive DevOps platform that includes features beyond version control, offering a wider set of tools for the software development lifecycle.

- Key Features:

- Integrated DevOps: Provides features for CI/CD pipelines, issue tracking, code review, container registry, and Kubernetes integration.

- Flexibility: Offers options for self-hosting (GitLab Community Edition) and a hosted SaaS version (GitLab.com).

- Open Core Model: Combines an open-source core with additional proprietary features in its enterprise versions.

- Use Cases:Suited for organizations looking for an all-in-one DevOps platform, especially those emphasizing CI/CD and automation.

**Bitbucket:**

- Overview: Bitbucket is developed by Atlassian and focuses on providing Git and Mercurial version control repositories.

- Key Features:

- Integration with Atlassian Tools: Offers seamless integration with Jira, Confluence, and other Atlassian products.

- Private Repositories: Provides free private repositories for small teams and integrates well within the Atlassian ecosystem.

- Lacks Extensive Community: Bitbucket may have fewer users than GitHub, but it's a strong choice for teams already using Atlassian products.

- Use Cases: Atlassian users looking for version control tightly integrated with other Atlassian tools, especially for private repositories and small teams.

**Industrial practices of using Git**

* Git version control best practices help software development teams meet the demands of rapid changes in the industry combined with increasing customer demand for new features.
* Write the smallest amount of code possible to solve a problem. After identifying a problem or enhancement, the best way to try something new and untested is to divide the update into small batches of value that can easily and rapidly be tested with the end user to prove the validity of the proposed solution and to roll back in case it doesn't work without deprecating the whole new functionality.
* Atomic Commits:

Making granular, atomic commits with descriptive messages to track changes efficiently and aid in future debugging.

* Rebasing and Merging:

Utilizing rebasing to maintain a cleaner commit history and avoid unnecessary merge commits.Choosing merge strategies (e.g., fast-forward, squash merging) based on project requirements.

* Automated Pipelines:

Implementing CI/CD pipelines triggered by version control events to automate build, test, and deployment processes.

* Testing and Quality Assurance:

Automated testing upon commits to prevent regressions and ensure code quality.

* Pull Requests (PRs) and Code Reviews:

Using PRs/merge requests for proposing changes and initiating code reviews.

Team members review code changes, suggest improvements, and ensure code quality before merging.

* Code Review Standards:

Following established guidelines for code reviews, including reviewing design, functionality, and adherence to coding standards.

* Write descriptive commit messages starting with a verb in present tense in imperative mood to indicate the purpose of each commit in a clear and concise manner.

**Cloning a repository to local**

**References**

1. <https://www.pluralsight.com/resources/blog/cloud/git-terms-explained>
2. <https://opensource.com/article/22/11/git-concepts>
3. <https://www.toolsqa.com/git/git-terminologies/>
4. <https://www.edureka.co/blog/git-commands-with-example/>
5. <https://www.atlassian.com/git/glossary>
6. <https://about.gitlab.com/topics/version-control/version-control-best-practices/>
7. <https://docs.github.com/articles/cloning-a-repository>
8. ChatGPT