GIT AND GITHUB

GIT:

- Git is a version control system.
- Version Control System is a tool that helps to track changes in code.

GIT-HUB:

- Website that allows developers to store and manage their code using Git

1.Concepts of GIT

1. Repository (Repo):

- A repository is a storage space where your project's history and files are kept.

- There are two types: local (on your machine) and remote (on a server).

2. Clone:

- Cloning is the process of creating a copy of a remote repository on your local machine.

3. Commit:

- A commit represents a snapshot of your project at a specific point in time.

- It includes changes made to files, a commit message explaining the changes, and a unique identifier (hash).

4. Branch:

- A branch is a separate line of development that diverges from the main codebase.

- It allows you to work on features or fixes without affecting the main code until you're ready to merge.

5. Merge:

- Merging is the process of combining changes from one branch into another.
- It's typically used to bring changes from a feature branch into the main branch.

6. Pull Request (PR):

- In Git repositories hosted on platforms like GitHub, GitLab, or Bitbucket, a pull request is a way to propose changes.
- It includes the changes you want to merge, and others can review, comment, and eventually merge the changes.

7. Fetch:

- Fetch retrieves changes from a remote repository but does not merge them into your local branch.
- It updates your remote tracking branches.

8. Pull:

- Pull is a combination of fetch and merge. It fetches changes from a remote repository and automatically merges them into your current branch.

9. Push:

- Push is the process of sending your local commits to a remote repository.
- It updates the remote repository with your changes.

10. Remote:

- A remote is a reference to a repository on the internet or another network.
- It allows you to interact with repositories hosted on platforms like GitHub, GitLab, or Bitbucket.

11. HEAD:

- HEAD is a reference to the latest commit in the branch you are currently working on.

- It points to the tip of the current branch.

12. Conflict:

- A conflict occurs when Git cannot automatically merge changes between branches.

- Manual intervention is required to resolve conflicts before the merge can be completed.

13.Fork:

-A fork is a new repository that shares code and visibility settings of the original "upstream" repository.

2.Basic Commands of GIT

Certainly! Here are some basic Git commands that you might find useful:

1. Initialize a Repository:

   - `***git init***`: Initializes a new Git repository in the current directory.

2. Clone a Repository:

   - `***git clone [repository_url]***`: Creates a copy of a remote repository on your local machine.

3. Configure Git:

   - `***git config --global user.name "[Your Name]"***`: Sets your name for Git commits.

   - `***git config --global user.email "[your_email@example.com]***"`: Sets your email for Git commits.

4. Check Repository Status:

   - `***git status***`: Shows the status of changes as untracked, modified, or staged.

5. Stage Changes:

- ***`git add [file]`***: Adds a file to the staging area.
- ***`git add .` or `git add --all`***: Adds all changes to the staging area.

6. Commit Changes:

- ***`git commit -m "[Descriptive Message]"`***: Commits the staged changes with a descriptive message.

7. Create a Branch:

- ***`git branch [branch_name]`***: Creates a new branch.
- ***`git checkout -b [branch_name]`***: Creates and switches to a new branch in one command.

8. Switch Between Branches:

- ***`git checkout [branch_name]`***: Switches to the specified branch.

9.Changes:

- ***`git merge [branch_name]`***: Merges changes from the specified branch into the current branch.

10. Pull Changes from Remote:

- ***`git pull origin [branch_name]`***: Fetches changes from the remote repository and merges them into the current branch.

11. Push Changes to Remote:

- ***`git push origin [branch_name]`***: Pushes local commits to the remote repository.

13. Undo Changes:

- ***`git reset [file]`***: Unstages changes for a file.
- ***`git reset --hard HEAD`***: Discards all changes in the working directory.

3.Concepts of GITHUB , GITLAB , GITBUCKET.

GitHub:

GitHub is a widely-used platform for hosting and collaborating on Git repositories. It serves as a hub for open-source projects, providing features like issues for tracking bugs and enhancements, pull requests for proposing and discussing changes, and actions for automating workflows. GitHub is known for its user-friendly interface and extensive community support, making it a go-to choice for many developers and organizations involved in collaborative software development.

GitLab:

GitLab is a comprehensive platform that goes beyond repository hosting. It includes built-in CI/CD tools for automating testing and deployment, as well as features like merge requests for collaboration, epics and issues for project management, and a wiki for documentation. GitLab's all-in-one approach makes it a popular choice for teams looking for an integrated solution that covers the entire development lifecycle, from code creation to deployment.

Bitbucket:

Bitbucket is a versatile platform supporting both Git and Mercurial repositories. It provides standard version control features like pull requests and branch permissions, along with additional tools such as pipelines for CI/CD and integration with Jira for project management. Bitbucket is available in two versions: Bitbucket Cloud, a hosted service, and Bitbucket Server, a self-hosted solution, offering flexibility to teams with different hosting preferences.

4.Industrial Practices of using GIT.

1. Collaborative Development:

- Git is widely used for collaborative coding efforts in industrial settings.
- Multiple developers can work on the same codebase simultaneously.

2. Code Review:

- Pull requests or merge requests are employed for thorough code reviews.
- Discussions, feedback, and approvals ensure code quality before merging.

3. Branching Strategies:

- Gitflow or GitHub Flow are common branching strategies.
- Defined rules for creating and merging branches streamline the development process.

4. Continuous Integration (CI) and Continuous Deployment (CD):

- Git integrates seamlessly with CI/CD systems.
- Automated testing, building, and deployment pipelines ensure code quality and quick releases.

5. Versioning and Tagging:

- Git supports precise versioning through tags.
- Tags mark releases, enabling easy rollback to specific versions.

6. Git Hooks:

- Git hooks, like pre-commit checks, are scripts triggered at various workflow points.
- They enforce code quality and adherence to standards.

7. Codebase Auditing and Monitoring:

- Git logs and history provide a detailed audit trail of code changes.
- Monitoring tools analyze Git history for project insights.

8. Issue and Project Management Integration:

- Git platforms integrate seamlessly with issue tracking and project management tools.
- This linkage ensures that code changes are associated with specific issues, fostering traceability.

9. Training and Documentation:

- Developers are trained on Git usage and best practices.
- Documentation, including guidelines and workflows, supports consistent usage and onboarding.

10. Git Flow Automation:

- Automation tools are sometimes employed to streamline Git Flow processes.

- These tools may automate the creation of feature branches, handle releases, and manage merges according to the chosen workflow.

5.Cloning a Repo to Local.

To clone a repository to your local machine using Git, follow these steps:

1. Open a Terminal or Command Prompt:

- Open the terminal or command prompt on your local machine. This depends on your operating system (e.g., Command Prompt or PowerShell on Windows, Terminal on macOS, or any terminal emulator on Linux).

2. Navigate to the Directory Where You Want to Clone the Repository:

- Use the `cd` command to navigate to the directory where you want to store the cloned repository.

```bash
cd path/to/your/directory
```

3. Clone the Repository:

- Use the `git clone` command followed by the URL of the repository you want to clone. The URL can be obtained from the repository's web page.

```bash
git clone https://github.com/example/repo.git
```

4. Provide Credentials (if required):

- If the repository is private and requires authentication, Git may prompt you to enter your username and password or a personal access token.

```bash
Username for 'https://github.com': your_username
```

5. Verify the Cloned Repository:

- After the cloning process is complete, navigate into the cloned directory using `cd repo` (replace "repo" with the actual name of the repository).

```bash
cd repo
```

You should now have a local copy of the repository on your machine.

6. Resources Used

- Chatgpt
- Apna College (YOUTUBE)
- Google