**GitHub**

What is Git?

Git is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to track changes in the source code, enabling multiple developers to work together.

What is GitHub?

GitHub is a Git repository hosting service that provides a web-based graphical interface. It is the world's largest coding community. Putting a code or a project into GitHub brings it increased, widespread exposure.

Benefits of GitHub and Git:

1. Version Control: Git allows for efficient version control, enabling you to track changes, collaborate with others, and easily revert to previous versions.

2. Collaboration: GitHub provides a platform for collaborative development, allowing multiple developers to work on the same project simultaneously, merge their changes, and resolve conflicts.

3. Code Management: GitHub serves as a centralised repository for storing and managing your code, making it easily accessible to yourself and others. It provides a structured organisation of files and directories.

4. Code Review: GitHub offers a built-in code review system, enabling developers to provide feedback, suggest improvements, and ensure the quality and integrity of the codebase.

5. Issue Tracking: GitHub provides a robust issue tracking system, allowing you to create, assign, and track issues or bugs in your project. It facilitates effective project management and prioritisation.

6. Community and Open Source: GitHub fosters a vibrant community of developers, making it a hub for open-source projects. It encourages collaboration, knowledge sharing, and contribution to open-source software.

7. Continuous Integration: GitHub supports seamless integration with various continuous integration tools, enabling automated testing, code quality checks, and deployment processes.

8. Documentation: GitHub provides a platform for documenting your projects using Markdown, making it easy to create and maintain project documentation alongside your code.

9. Project Showcase: GitHub serves as a professional portfolio for developers, allowing them to showcase their work, contribute to popular projects, and demonstrate their skills to potential employers or clients.

10. Integration and Extensibility: GitHub integrates with numerous development tools and services, providing a seamless workflow and enhancing productivity.

In summary, GitHub and Git offer efficient version control, collaboration, code management, code review, issue tracking, and integration capabilities, fostering effective and streamlined software development processes.

**100 Git Day to Day use commands:**

git init: Initializes a new Git repository.

git clone [repository URL]: Creates a local copy of a remote repository.

git add [file]: Adds a file to the staging area.

git commit -m "[message]": Commits changes to the repository with a descriptive message.

git status: Displays the status of the repository.

git diff: Shows the differences between the working directory and the staging area.

git diff --cached: Shows the differences between the staging area and the last commit.

git branch: Lists all branches in the repository.

git branch [branch name]: Creates a new branch.

git checkout [branch name]: Switches to the specified branch.

git merge [branch name]: Merges changes from the specified branch into the current branch.

git remote add [remote name] [remote URL]: Adds a remote repository.

git remote -v: Lists all remote repositories.

git pull [remote name] [branch name]: Fetches changes from a remote repository and merges them into the current branch.

git push [remote name] [branch name]: Pushes changes to a remote repository.

git fetch: Fetches changes from a remote repository.

git log: Displays the commit history.

git show [commit hash]: Shows the details of a specific commit.

git reset [commit hash]: Resets the repository to a specific commit.

git revert [commit hash]: Creates a new commit that undoes the changes made in a specific commit.

git rm [file]: Removes a file from the repository.

git mv [old file name] [new file name]: Renames a file.

git stash: Temporarily saves changes that are not ready to be committed.

git stash apply: Applies the most recent stash.

git stash list: Lists all stashes.

git stash drop: Deletes the most recent stash.

git tag: Lists all tags in the repository.

git tag [tag name]: Creates a new tag.

git tag -a [tag name] -m "[message]": Creates an annotated tag with a message.

git show-ref --tags: Shows the commit hash for a specific tag.

git push --tags: Pushes all tags to a remote repository.

git branch -d [branch name]: Deletes a branch.

git push origin :[branch name]: Deletes a remote branch.

git blame [file]: Shows the author and last modified date of each line in a file.

git config --global user.name "[name]": Sets the author name for commits.

git config --global user.email "[email]": Sets the author email for commits.

git checkout -- [file]: Discards changes made to a file.

git clean -n: Shows the untracked files that will be deleted.

git clean -f: Deletes all untracked files.

git remote prune [remote name]: Removes remote branches that have been deleted.

git cherry-pick [commit hash]: Applies the changes made in a specific commit to the current branch.

git rebase [branch name]: Incorporates changes from another branch onto the current branch.

git rebase -i [commit hash]: Interactive rebase, allows squashing, rearranging, or editing commits.

git reflog: Lists the history of branch references, including deleted commits.

git config --global alias.[alias name] "[git command]": Creates a custom alias for a Git command.

git bisect start: Starts a binary search to find the commit that introduced a bug.

git bisect good: Marks the current commit as good (bug-free).

git bisect bad: Marks the current commit as bad (contains a bug).

git bisect reset: Ends the binary search and returns to the original state.

git submodule init: Initializes submodules in the repository.

git submodule update: Updates submodules to the latest commit.

git submodule foreach [command]: Runs a specified command in each submodule.

git log --author="[author name]": Shows the commit history for a specific author.

git log --grep="[pattern]": Shows the commit history that matches a specific pattern.

git log --oneline: Shows a concise, one-line commit history.

git log --stat: Shows the commit history with statistical information.

git log --since="[time period]": Shows the commit history since a specific time period.

git clean -fd: Deletes untracked files and directories forcefully.

git config --global core.autocrlf [true/false/input]: Configures line-ending handling.

git blame -L [start],[end] [file]: Shows the author and last modified date for a specific range of lines in a file.

git cherry-pick [start commit hash]..[end commit hash]: Applies a range of commits to the current branch.

git push [remote name] --delete [branch name]: Deletes a remote branch.

git reset [file]: Unstages a file, removing it from the staging area.

git reset --hard: Resets the repository to the last commit, discarding all changes.

git rebase --continue: Continues a rebase operation after resolving conflicts.

git rebase --abort: Aborts a rebase operation, returning to the original state.

git stash save "[message]": Saves changes to a stash with a descriptive message.

git stash apply [stash]: Applies a specific stash.

git stash drop [stash]: Deletes a specific stash.

git tag -d [tag name]: Deletes a tag.

git tag -l "[pattern]": Lists tags that match a specific pattern.

git push origin --delete [tag name]: Deletes a remote tag.

git mv [old directory] [new directory]: Moves or renames a directory.

git diff [commit hash] [commit hash]: Shows the differences between two specific commits.

git blame -C [file]: Shows the author and last modified date of each line in a file, even for copied or renamed lines.

git config --global core.editor [editor]: Configures the default editor for Git.

git add -p [file]: Interactively selects changes to add from a file.

git commit --amend: Modifies the last commit with additional changes or a new message.

git commit --author="[author name]": Creates a new commit with a specified author.

git reset --soft [commit hash]: Resets the repository to a specific commit, keeping the changes in the staging area.

git revert -n [commit hash]: Creates a new commit that undoes the changes made in a specific commit, without creating a new commit.

git stash drop [stash]: Deletes a specific stash.

git stash pop [stash]: Applies and deletes a specific stash.

git fetch --prune: Fetches changes from remote repositories and removes references to deleted branches.

git pull --rebase: Fetches changes from a remote repository and rebases the current branch on top of the fetched changes.

git cherry-pick --abort: Aborts the cherry-pick operation, returning to the original state.

git log --graph: Shows the commit history in a graph format.

git log --before="[date]": Shows the commit history before a specific date.

git revert --no-commit [commit hash]: Undoes the changes made in a specific commit, preparing them for a new commit.

git commit --fixup [commit hash]: Creates a fixup commit that references a specific commit for later squashing.

git commit --squash [commit hash]: Squashes a specific commit into the previous commit during an interactive rebase.

git stash drop [stash]: Deletes a specific stash.

git push --force: Forces a push to a remote repository, overwriting existing history.

git push --follow-tags: Pushes commits and also pushes annotated tags.

git config --global core.ignorecase [true/false]: Configures Git to be case-sensitive or case-insensitive.

git commit --no-verify: Skips pre-commit and commit-msg hooks during a commit.

git commit --signoff: Adds a "Signed-off-by" line to a commit.

git commit --allow-empty: Creates an empty commit.

git branch -m [new branch name]: Renames the current branch.

git show-branch: Displays the relationship between branches.