**GITHUB DOCUMENTATION**

**1.what is github?**

GitHub is a web-based platform that allows users to store, share, and collaborate on code, web pages, and other content.

**2.what is git?**

Git is a free, open-source version control system (VCS) that helps developers track and manage changes to their code and projects. It's the most widely used VCS in the world and is considered the modern standard for software development.

Here are some features of Git;

**. Branching**

**. Merging**

**. Snapshots**

**. Pull requests**

**3.stages of git?**

Files in a Git repository go through three stages before being under version control

* **Untracked:** The file exists, but is not part of Git's version control.
* **Staged:**The file has been added to Git's version control but changes have not been committed.
* **Committed:**The change has been committed

.

Some Git commands that can be used to interact with these stages include

* **git add:** Stages individual files or all files in a project directory. For example, git add -A stages all files, including new, modified, and deleted files.
* **git commit:** Creates a snapshot of the staged changes in a Git project's history.
* **git init:** Creates the initial .git directory in a new or existing project.
* **git status:** Provides information about the status of branches, modifications in the working directory, and the stage.

The staging area, also called the "index", is a file in the Git directory that contains information about what changes will go to the next commit.

**4.alternatives for github?**

There are several alternatives to GitHub, including open source options, community-driven platforms, and cloud-based platforms:

* **GitLab**

A self-hosted platform with tools for project management and collaboration. It has features like issue trackers, group milestones, and configurable issue boards.

* **Bitbucket**

A code hosting platform with unlimited public and private repositories. It's designed for professional teams and supports third-party integrations.

* **Azure DevOps**

A cloud-based platform from Microsoft with tools for version control, agile planning, and team collaboration.

* **Launchpad**

A collaborative development platform for Ubuntu and other open-source projects.

* **Radicle**

A sovereign code infrastructure for secure collaboration over a peer-to-peer network.

* **Gitea**

A lightweight code hosting solution written in Go.

* **HGKeeper**

A Mercurial repository hosting tool that supports HTTP pulling and SSH pushing.

* **Amazon CodeCommit**

A fully-managed source control service for hosting private Git repositories.

* **Google Cloud Source Repositories**

A cloud code platform that allows users to connect to repositories hosted on GitHub or Bitbucket.

.

**5.git config commands**

The git config command is a convenience function that is used to set Git configuration values on a global or local project level.

**1.Set Username**

git config --global user.name "Your Name"

**2.Set Email**

git config --global user.email [you@example.com](mailto:you@example.com)

**3.Check global configuration**

git config --global –list

**4.Check local (repository-specific) configuration**

git config --local –list

**5.Check system configuration**

git config --system –list

**6.Set editor globally**

git config --global core.editor "code --wait" # for VS Code

git config --global core.editor "nano" # for Nano

git config --global core.editor "vim" # for Vim

**7.Set merge tool globally**

git config --global merge.tool "meld"

**8.Create an alias for 'git status'**

git config --global alias.st status

**9.Create an alias for 'git checkout'**

git config --global alias.co checkout

**10.Create an alias for 'git commit'**

git config --global alias.ci commit

**11.Create an alias for 'git log --oneline'**

git config --global alias.lg "log --oneline"

**13.Convert line endings to LF on commit and to CRLF on checkout (Windows)**

git config --global core.autocrlf true

**14.Ensure line endings are LF across all platforms**

git config --global core.autocrlf input

**15.Configure pull to rebase by default**

git config --global pull.rebase true

**16.Unset global configuration**

git config --global --unset user.name

**17.Unset local configuration**

git config --local --unset core.editor

**18.Open global config file for editing**

git config --global –edit

**19.Open local config file for editing**

git config --local --edit

**6.git remote commands**

The git remote command lets you create, view, and delete connections to other repositories. Remote connections are more like bookmarks rather than direct links into other repositories.

**1.Adding a Remote**

git remote add <name> <url>

ex - git remote add origin <https://github.com/username/repo.git>

**2.Viewing Remotes**

git remote -v

**3.Removing a Remote**

git remote remove <name>

exe - git remote remove origin

**4.Renaming a Remote**

git remote rename <old-name> <new-name>

ex - git remote rename origin upstream

**5.Fetching from a Remote**

git fetch <name>

ex - git fetch origin

**6.Pushing to a Remote**

git push <name> <branch>

ex - git push origin main

**7.Pulling from a Remote**

git pull <name> <branch>

ex - git pull origin main

**8.Setting the Default Remote**

git push -u <name> <branch>

**9.Viewing Remote Branches**

git branch -r

**10.Deleting a Remote Branch**

git push <name> --delete <branch>

**11.Cloning a Repository**

git clone <url>

**12.Getting Remote Information**

git remote show <name>

**7.git add commands**

The Git add command adds a change in the working directory to the staging area.

Stage a specific file

git add <file>

ex - git add file.txt

Stage multiple specific files

git add <file1> <file2> <file3>

ex - git add file1.txt file2.txt file3.txt

Stage all changes in the current directory

git add .

Stage all changes in the entire repository

git add -A

Stage changes interactively

git add -i

Stage all changes in tracked files (modified and deleted)

git add -u

Stage specific patterns of files

git add <pattern>

ex - git add \*.txt

Stage changes in a directory

git add <directory>

ex - git add src/

Stage parts of a file interactively (patch mode)

git add -p <file>

ex - git add -p file.txt

Dry run to see what would be added

git add -n <file>

ex - git add -n file.txt

**8.git branch command**

The git branch command lets you create, list, rename, and delete branches. It doesn't let you switch between branches or put a forked history back together again. For this reason, git branch is tightly integrated with the git checkout and git merge commands.

List all branches

git branch

git branch -a

Create a new branch

git branch <branch-name>

ex - git branch feature-branch

Switch to a branch

git switch <branch-name>

git checkout <branch-name>

ex - git checkout feature-branch

Create and switch to a new branch in one command

git checkout -b <branch-name>

ex - git checkout -b feature-branch

git switch -c <branch-name>

Delete a branch

git branch -d <branch-name>

ex - git branch -d feature-branch

git branch -D <branch-name>

ex - git branch -D feature-branch

Rename a branch

git branch -m <new-branch-name>

ex - git branch -m new-branch-name

git branch -m <old-branch-name> <new-branch-name>

git branch -m old-branch-name new-branch-name

Compare branches

git diff <branch1> <branch2>

ex - git diff master feature-branch

View the last commit on each branch

git branch -v

View merged branches

git branch --merged

View unmerged branches

git branch --no-merged

Set upstream for a branch (tracking a remote branch)

git branch --set-upstream-to=<remote>/<branch> <local-branch>

ex - git branch --set-upstream-to=origin/feature-branch feature-branch

Show branch details (verbose mode)

git branch -vv

**9. git pull commands**

The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows.

Basic Git Pull Command

git pull

Specify a Remote and Branch

git pull <remote> <branch>

ex - git pull origin main

Git Pull with Rebase

git pull –rebase

Git Pull with Fast-Forward Only

git pull --ff-only

Git Pull and Squash Commits

git pull --squash

Git Pull and Set Upstream

git pull --set-upstream <remote> <branch>

**10. git push commands**

The git push command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo.

Basic Git Push Command

git push

Push to a Specific Remote and Branch

git push <remote> <branch>

ex - git push origin main

Set Upstream and Push (First Time Push)

git push --set-upstream <remote> <branch>

ex - git push --set-upstream origin main

Push All Branches

git push --all <remote>

ex - git push --all origin

Force Push (Overwrite Remote Changes)

git push --force

Force Push with Lease (Safer)

git push --force-with-lease

Push Tags

git push --tags

Push a Specific Tag

git push origin <tag>

ex - git push origin v1.0.0

Delete a Remote Branch

git push <remote> --delete <branch>

ex - git push origin --delete feature-branch

Push to a Specific Branch but Keep Local Branch Intact

git push origin <local-branch>:<remote-branch>

ex - git push origin feature-branch:main

**11. git init commands**

The git init command creates a new Git repository. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository. Most other Git commands are not available outside of an initialized repository, so this is usually the first command you'll run in a new project.

Basic Git Init Command

git init

Git Init in a Specific Directory

git init <directory>

Example: git init my-project

Git Init with Bare Repository

git init --bare

Git Init with Shared Permissions

git init --shared[=<permissions>]

Example: git init --shared=group

**12. upstrems commands**

In Git, the term "upstream" refers to the main branch from which a given branch was branched off of and to which it may eventually push changes to. By default, Git uses "origin" as the remote name for the upstream repository, but you can configure this to any name.

Set Upstream for a Local Branch (First Time Push)

git push --set-upstream <remote> <branch>

Example: git push --set-upstream origin main

Set Upstream for an Existing Local Branch

git branch --set-upstream-to=<remote>/<branch> <local-branch>

Example: git branch --set-upstream-to=origin/main feature-branch

Check Upstream Branch for the Current Branch

git rev-parse --abbrev-ref --symbolic-full-name @{u}

View All Tracking Branches

git branch -vv

Change Upstream Branch for a Local Branch

git branch --set-upstream-to=<remote>/<branch> <local-branch>

Remove Upstream Branch Tracking

git branch --unset-upstream

Pull from Upstream

git pull

Push to Upstream

git push

Example Workflow

git push --set-upstream origin main

git pull

git push

**13. shortcuts**

**General Keyboard Shortcuts**

s: Focus the search bar

t: Search files in a repository

l: Jump to a specific line in a file

w: Switch between branch/tag menus

y: Convert the URL to a permalink (locks URL to current commit)

.: Open GitHub repository in web-based VS Code editor

**Navigation**

**g + n**: Go to Notifications

**g + p**: Go to Pull Requests

**g + i**: Go to Issues

**g + m**: Go to Marketplace

**g + d**: Go to Discussions

**Code Review**

**u**: Collapse all outdated comments in a Pull Request

**i**: Show or hide the repository sidebar (files navigation)

**Issues/PRs**

**a**: Assign yourself to an Issue or PR

**l**: Label an Issue or PR

**m**: Add a milestone to an Issue or PR

**c**: Create a new Issue

**Code Navigation**

**f**: Open find in file while viewing a file

**b**: Open the blame view for the current file

**h**: Show file history

**Commenting**

**ctrl + enter**: Submit a comment in a Pull Request or Issue

**cmd/ctrl + shift + k**: Add a code block in a comment

**r**: Reply to a comment in an Issue or PR review