R. KUMAR.

8/2/24 Git

Repository Basics: your project over time

.git/ in a project traces all changes made t

→ while cloning it creates local repository and maintai
conn. b/w Remote rep and local repository.

→ Browse → init create repository

→ Git clone: Create an exact copy of all files and changes
a git repo at the time of the clone.

→ Remote Repository: — A git repo hosted on the internet or ne

→ Local Repository: — A git repo hosted on your machine.

Commits:

→ Commit is a snapshot of your repository at one point in time.

→ git status and stored in "objects" folder with unique ID number or "SHA"

→ git add path

→ git commit -m "some message".

→ git commit --amend -m

→ git commit messages are used to explain the function of the commit

→ git commit workflow: 1) making changes
2) staging changes
3) commiting change.

→ git commit message title is limited to 72 characters.

→ Don't end commit message with punctuations.

→

Commit Type	Description.
Feat	→ Refers to feature
Fix	→ Refer to bug fix
Docs	→ Changes to documentation like Readme
style	→ Style or formatting changes
Perf	→ Improves code performance
test	→ Improves test a feature.

Diff

R. KUMAR

→ Working Directory: The file system where you can view and modify files.

→ Staging Directory: The index folder inside the '.git' folder that contains the changes added through staging

→ Diff: A change b/w two data sets.

Branching:

→ We can 1) Create
2) rename and
3) delete the branches.

→ Git checkout <name of branch>

→ Five different branch types in total.

1) Main } Primary
2) Develop
3) Feature } supporting
4) Release
5) Hotfix

→ main branch is commonly referred as "master"

→ " " contain production-ready code that can be released.

→ Develop Branch: It contains pre production code with newly developed features that are in the process of being tested

→ Feature Branch: It is used when adding new feature to your code.

→ Release Branch: It is used when preparing new production releases.

→ Hotfix Branch: It is used to quickly address necessary changes in your main branch.

→ Creating a local branch: → git branch feature-A
→ git branch
→ git checkout feature A
→ git ~~branch~~ checkout -b feature-B
→ git branch

→ Fuzzy Finder : cmd/ctrl + P

→ Rename a branch:

git branch --list

git branch -m newname (current branch name will change)

→ git branch -m bugA bugB (other branch name where can be changed i.e. bugA to bugB)

→ Delete a branch:

git branch -d branchname

-D

→ if we want to delete Remote repository we use

git push origin --delete update-log

→ To get list of remote branches give (git branch -r)

→ undo : Ctrl/cmd + Z

Merging and Rebasing:

→ Rebasing → cleaner history
→ more readable graph
→ tougher to resolve conflicts.

→ Merging → Preserves history
→ better for merge conflicts.
→ easy to undo.

→ merge conflicts : This will occur when git unable to automatically resolve the files in code b/w to commits.

→ after this it will ask → 1) Keep Modified version
2) Delete the file
3) Keep base version.

→ Rebasing - duting commits from one branch and adding them to another

→ Merge - add all changes from one branch to another branch

→ conflicts may trigger when 1) merging a branch
4) Rebasing a branch
3) cherry picking

Remotes, forks, and pull requests:

→ Git pull: The changes you will be getting from your remote repository and adding to your local copy.

→ Git fetch: If we want to see all the current branches and changes in your remote repository

14/4/24

→ Comparing Git pull vs fetch: Git fetch is safer alternative because it pulls in all the commits from your remote but doesn't make any changes to your local files.

→ Git pull is faster and advanced one.

→ Commands: git fetch / git pull

→ what is ssh? (Secure shell) → It is a network protocol that allows one computer to securly connect to another computer.

→ Fork: A clone of a git repository, typically hosted on a git hosting service.

→ Pull Request: An event where a contributer asks a repo maintainer to review code that they wish to merge into a project.

→ It is also known as merge request.

Situational Tools:

→ Stashing: Creating a stash in Git saves uncommitted changes so you can work on other things in your repository without losing your work.

→ When you're ready to reapply your changes, you will have the option to "apply" or "pop" your stash to your currently checked out branch.

→ Git stash apply: Git stash apply will take the changes you have stored in a stash and apply them to the working directory of your currently checked out branch and will also keep the stash intact (unchanged).

→ Git stash pop: It also do the same thing as apply do, but only difference is, It will delete the stash after the changes have been applied.

→ Cherry Pick: It allows you to apply a specific commit from one branch to another. It enables you to select and incorporate individual commits without merging an entire branch.

→ It can be useful in scenarios where you only want to bring in specific changes from one branch to another.

→ Squash: The process of combining multiple commits into a single commit.

→ You are essentially condensing several commits to into a single commit.

→ It creates a cleaner and more concise commit history.

→ Git tag: In Git, a tag is a reference to a specific commit in the repository. Unlike branches, tags are typically used to mark specific points in the commit history as being important.

→ Git Hooks: These are shell scripts that allows you to run at certain points in the git workflow.

→ These scripts can be customized to perform specific actions before or after events such as

committing, merging, pushing & receiving changes.

* → Git provides both client side and server-side hooks to tailor the workflows to the needs of your project.

→ Git Reset: It is used to reset the state of the current branch to a specific commit.

→ The Three primary modes of git reset are

(1) Soft Reset: A soft reset moves the branch pointer to a different commit but leave the changes staged in the working directory and staging area.

(2) Mixed Reset: It resets the branch pointer and the staging area, but it leaves the changes in your working directory as unstaged.

(3) Hard Reset: Undo all the changes between HEAD and the commit and discard all the changes.

→ Git LFS: Git LFS is a Git extension that allows users to save space by storing binary files in a different location.

→ Interactive Rebase: It is a powerful Git feature that allows you to interactively and iteratively modify the commit history of a branch.

→ Git patch: It is used to share the code with team members and project collaborators. It involves taking someone's work and adding it to local Git repository.

→ Git worktree: It allows you to checkout and work in multiple Git branches simultaneously.

(1) Git worktree Add
(2) Git worktree List
(3) Git worktree Remove