GIT

Friday, December 27, 2024 5:48 PM

Day 1: Basic git commands

Commands:

- **Initialize Repository:**
  - git init → Initializes a new Git repository.

Create Files:
- touch index.html file.txt ... → Creates new files (untracked initially).

Add Files to Staging Area:
- git add index.html → Adds specific file to the staging area.
- git add . → Adds all changes (new, modified, deleted files) to the staging area.

Commit Changes:
- git commit -m "message context" → Commits staged changes with a descriptive message.

Amend Last Commit:
- git commit --amend → Adds changes to the last commit and allows modifying its message.
- git commit --amend --no-edit → Modifies the last commit without changing its message.

View Repository Status and History:
- git status → Shows the current status of the working directory and staging area.
- git log → Displays commit history.
- git log --oneline → Displays commit history in a condensed, one-line format.
- git reflog → Shows a history of HEAD movements.

Unstage Changes:
- git restore --staged m3.txt → Unstages the file (moves it back to the working directory).

Remove Commits:
- git rm <file> → Deletes the file and stages the removal for the next commit.
- git rm --cached m3.txt → Unstage a file but keep it locally.

Reset Commits:
- git reset --soft HEAD~1 → Resets the last commit but keeps changes staged. (local)
- git reset --soft b38go29 → Resets all commits up to the specified commit ID (keeping changes staged).
- git reset --mixed HEAD~1 → Resets the last commit and unstages changes (default).
- git reset HEAD~1 → Equivalent to git reset --mixed HEAD~1.
- git reset --hard HEAD~1 → Resets the last commit, unstages, and discards changes.
- git reset --hard ORIG_HEAD → Resets to the original state before the last operation.

Revert Commit:
- git revert b38go29 → Reverts the changes introduced by the specified commit and creates a new commit with the reversed changes.

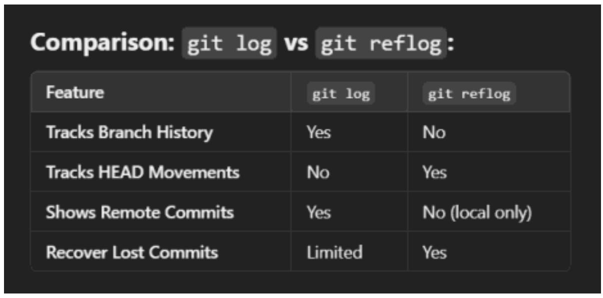
Branch Management:
- git branch → Lists branches.
- git branch *name* → Gets branch or creates it

Switch Branches:
- git checkout *name* → Switches to the specified branch.
- git checkout -b *name* → Creates a new branch and switches to it.
- git switch *name* → Switches to the specified branch and creates it if not created.
- git switch -C *name* → Creates and switches to a branch, overwriting if it already exists.
- git branch -M *name* → Switch master branch.

Stash Commits:
- git stash → Saves untracked changes in stash memory (like cache)
- git stash push -m "message" → Save stash with message.
- git stash list → Displays all saved stashes with their index and messages.
- git stash apply *ID* → Restores changes from a specific stash but keeps it in the list.
- git stash pop ID → Apply stash and remove from stash list
- git stash clear →clear stash list

Terminologies:
Head: reference to the **current branch** and the last commit on that branch.
Detached head: points directly to a specific commit **instead of a branch.**
~: reference commits relative to the current HEAD.
.gitinore: A file is used in Git to specify files or directories that should be **ignored by Git**.

NOTE:
- git reset --soft: keeps changes in the **staging area**.
- git reset --mixed: keeps changes in the **working directory** but unstages them.
- git reset --hard: removes changes entirely from both the working directory and staging area.


Comparison: `git log` vs `git reflog`:

Feature	`git log`	`git reflog`
Tracks Branch History	Yes	No
Tracks HEAD Movements	No	Yes
Shows Remote Commits	Yes	No (local only)
Recover Lost Commits	Limited	Yes

Day 2: Git & Github

Merge Branches:

- **Basic Merge:** git merge <branch> → Combines the specified branch into the current branch.
- **Recursive Merge:**
  - Fast-forward (default): Moves the branch pointer forward.
  - Recursive strategies: -ours, -octopus, -subtree.
  - git merge -s recursive → Uses the recursive strategy (similar to ort but slower).
- **Squash Merge:**
  - git merge --squash → Combines changes without committing, placing them in the staging area.
- **Force Non-Fast-Forward:**
  - git merge --no-ff → Ensures a recursive merge even if a fast-forward is possible.

Rebase:

- **Rebase Onto a Branch:**
  - git rebase master → Moves the current branch to the tip of master, rewriting commit history.
- **Interactive Rebase:**
  - git rebase -i HEAD~1 → Opens an interactive editor to modify commits (e.g., reword, edit, squash).

Delete Branches:

- **Hard Delete** *(Ignores Unmerged Changes):*
  - git branch -D <branch> → Deletes the branch even if there are unmerged updates.
- **Soft Delete** *(Prevents Deletion with Unmerged Changes):*
  - git branch -d <branch> → Deletes the branch only if all updates are merged.

GitHub Commands:

- **Set Up Remote Repository:**
  - git remote add origin <SSH_URL> → Connects the local repository to a remote repository.
- **Pull Changes from Remote:**
  - git pull <SSH_URL> → Fetches and integrates changes from the remote repository.
- **Push Changes to Remote:**
  - git push origin main → Pushes commits to the main branch of the remote repository.
- **Fetch Without Merging:**
  - git fetch origin main → Updates the local repository with changes from origin/main without merging.
- **View Differences Between Branches:**
  - git log main..origin/main → Shows commits in origin/main that are not in main.
  - git diff main origin/main → Displays differences between the local main branch and origin/main.

Note: a: refers to the old version, b: to the new version, and --- /dev/null indicates a new file.

Fast-Forward vs. Recursive Merge:

- Fast-forward (default): Moves the branch pointer forward.
- Recursive strategies: -ours, -octopus, -subtree.

Github termenologies:

- delta: Changes or differences between files/commits. sends differences
- reused: Files or commits that Git already knows about on the remote. skips
- pack-reused: A group of objects (like files, commits) that Git already has in the remote repository. reuses instead of sending