5-Days Workshop on Git and GitHub

Day 5 – Git Extras

25th February 2022

By,
Microsoft Learn Student Ambassador Community

Chat Issue Solved

- Go to this link (https://stin.to/67o1o) and we can chat online without any hurdles.
- We can also chat via Discord after this session.

Undoing Changes

- You can use **`git commit --amend –m "<new commit message>"`** to change the existing commit message,
- You can use **`git commit --amend --no-edit`** to amend the new staged changes without a new commit message.
- You can unstage a staged file with the command **`git restore --staged <filename>`**.
- You can unmodify a modified file with command **`git restore <filename>`**.
- You can use **`git restore --source <version> <filename>`** to restore a file of any version.

Git Aliases

- If you don’t want to type the entire text of each of the Git commands, you can easily set up an alias for each command.
- This can be done by: `git config --global alias.<short cmd> "<full command"`
- For example: We can create a shortcut to show last commit (`git log -1 --oneline`) by using command; `git config --global alias.last "log -1 --oneline"`,
- So, if we use command git last it will automatically run command `git log -1 --oneline`

Rebasing

- To integrate changes of one branch to another, we can use merge or rebase.
- Rebasing helps to create a clean look in the project.
- It maintains the linear structure of the commit history.
- Rebasing is just like replaying changes made in one branch in another branch.
- To rebase a branch to master use command;
  - `git checkout <branch>`
  - `git rebase master`
  - `git checkout master`
  - `git merge <branch>`

Rebase vs Merge

- To rebase means to change the entire history of your repository.
- To merge means to join the two branches together.
- The strong point to merge is that commit history is a historical document, valuable in its own right, and shouldn’t be tampered with.
- Similarly, merging makes things messier and rebase makes things look clean.
- **Rebase local changes before pushing to clean up your work, but never rebase anything that you’ve pushed somewhere.**

Stashing

- Git stash saves the uncommitted changes locally, allowing you to make changes, switch branches, and perform other Git operations.
- To create a stash, use command `git stash save "<stash message>"`
- To list all the stashes, use command `git stash list`.
- To retrieve stash changes, `git stash apply stash{@n}`
- To delete all the stash, use command `git stash clear`
- To delete only one stash, use command `git stash drop <stash_id>`

Squashing Commits

- The act of "squashing" your commits means that you combine multiple existing commits into a single one.
- To squash, use command `git rebase -i HEAD~n`
- And change pick to squash.

Git Reset

- To remove the commit, without removing the changes use `git reset --soft <hash>`.
- To remove the commit, by removing the staged file to working directory use `git reset --mixed <hash>`
- To remove the commit, by removing entire changes use command `git reset --hard <hash>`

Thank you!

Please refer to the chat section on our Microsoft Teams for resources and feel free to ask any queries about this session in our discord channel **#discussions**.