Reading: Git Commands

In this reading, you will summarize and describe additional Git commands that you may use while working on your projects. You will also look at the syntax for each command.

Git is a widely used version control system that offers numerous benefits to developers and teams working on software development projects.

Let's look at some useful Git commands and understand them:

1. **git add**
    - *Description*: It adds changes to the staging area. This command stages the changes made to the files and prepares them for the next commit.
    - *Syntax*:
        - **git add <filename.txt>** (to add a specific file)
        - **git add .** (to add all the files that are new or changed in the current directory)
        - **git add -A** (to add all changes in the entire working tree, from the root of the repository, regardless of where you are in the directory structure)
2. **git reset**
    - *Description*: It resets changes in the working directory. When used with –hard HEAD, this command discards all changes made to the working directory and staging area and resets the repository to the last commit (HEAD).
    - *Syntax*:
        - **git reset**
        - **git reset –hard HEAD**
3. **git branch**
    - *Description*: It lists, creates, or deletes branches in a repository. To delete the branch, first check out the branch using **git checkout** and then run the command to delete the branch locally.
    - *Syntax*:
        - **git branch** (to list branches)
        - **git branch <new-branch>** (to create a new branch)
        - **git branch -d <branch-name>** (to delete a branch)
4. **git checkout main**
    - *Description*: It switches to the "main" branch. This will switch your current branch to "main."
    - *Syntax*: **git checkout main**
5. **git clone**
    - *Description*: It copies a repository from a remote source to your local machine. This will create a copy of the repository in your current working directory.
    - *Syntax*: **git clone <repository URL>**
6. **git pull**
    - *Description*: It fetches changes from a remote repository and merges them into your local branch. First, switch to the branch that you want to merge changes into by running the **git checkout** command. Then, run the **git pull** command, which will fetch the changes from the main branch of the origin remote repository and merge them into your current branch.
    - *Syntax*: **git pull origin main**
7. **git push**
    - *Description*: It uploads local repository content to a remote repository. Make sure you are on the branch that you want to push by running the **git checkout** command first, then push the branch to the remote repository.
    - *Syntax*: **git push origin <branch-name>**
8. **git version**
    - *Description*: It displays the current Git version installed on your system.
    - *Syntax*: **git version**
9. **git diff**
    - *Description*: It shows changes between commits, commit and working tree, etc. It also compares the branches.
    - *Syntax*:
        - **git diff** (shows the difference between the working directory and the last commit)
        - **git diff HEAD~1 HEAD** (shows the difference between the last and second-last commits)
        - **git diff <branch-1> <branch-2>** (compares the specified branches)

10. **git revert**
    - *Description*: It reverts a commit by applying a new commit. This will create a new commit that undoes the changes made by the last commit.
    - *Syntax*: **git revert HEAD**
11. **git config –global user.email <Your GitHub Email>**
    - *Description*: It sets a global email configuration for Git. This needs to be executed before doing a commit to authenticate the user's email ID.
    - *Syntax*: **git config –global user.email <Your GitHub Email>**
12. **git config –global user.name <Your GitHub Username>**
    - *Description*: It sets a global username configuration for Git. This needs to be executed before doing a commit to authenticate users' username.
    - *Syntax*: **git config –global user.name <Your GitHub Username>**
13. **git remote**
    - *Description*: It lists the names of all remote repositories associated with your local repository.
    - *Syntax*: **git remote**
14. **git remote -v**
    - *Description*: It lists all remote repositories that your local Git repository is connected to, along with the URLs associated with those remote repositories.
    - *Syntax*: **git remote -v**
15. **git remote add origin <URL>**
    - *Description*: It adds a remote repository named "origin" with the specified URL.
    - *Syntax*: **git remote add origin <URL>**
16. **git remote rename**
    - *Description*: The git remote rename command is followed by the name of the remote repository (origin) you want to rename and the new name (upstream) you want to give it. This will rename the "origin" remote repository to "upstream."
    - *Syntax*: **git remote rename origin upstream**
17. **git remote rm <name>**
    - *Description*: It removes a remote repository with the specified name.
    - *Syntax*: **git remote** *rm origin*
18. **git format-patch**
    - *Description*: It generates patches for email submission. These patches can be used for submitting changes via email or for sharing them with others.
    - *Syntax*: **git format-patch HEAD~3** (creates patches for the last three commits)
19. **git request-pull**
    - *Description*: It generates a summary of pending changes for an email request. It helps communicate the changes made in a branch or fork to the upstream repository maintainer.
    - *Syntax*: **git request-pull origin/main <myfork or branch_name>**
20. **git send-email**
    - *Description*: It sends a collection of patches as emails. It allows you to send multiple patch files to recipients via email. Please make sure to set the email address and name using the **git config** command so that the email client knows the sender's information when sending the emails.
    - *Syntax*: **git send-email *.patch**
21. **git am**
    - *Description*: It applies patches to the repository. It takes a patch file as input and applies the changes specified in the patch file to the repository.
    - *Syntax*: **git am <patchfile.patch>**
22. **git daemon**
    - *Description*: It exposes repositories via the Git:// protocol. The Git protocol is a lightweight protocol designed for efficient communication between Git clients and servers.
    - *Syntax*: **git daemon –base-path=/path/to/repositories**
23. **git instaweb**
    - *Description*: It instantly launches a web server to browse repositories. It provides a simplified way to view repository contents through a web interface without the need for configuring a full web server.

- *Syntax*: **git instaweb –httpd=webrick**

24. **git rerere**

    - *Description*: It reuses recorded resolution of previously resolved merge conflicts. Please note that rerere.enabled configuration option needs to be set to "true" (**git config –global rerere.enabled true**) for git rerere to work. Additionally, note that git rerere only applies to conflicts that have been resolved using the same branch and commit.

    - *Syntax*: **git rerere**

Conclusion

Git can be a highly useful tool in any development project. If your team utilizes Git for version control, you will likely use most of these commands regularly.

Author

Anamika Agarwal

Other Contributor

Lavanya T S