Git Advanced Commands

Most of the time, you need to compare two git files or branches before you commit or push. Here is a handy command to do that.

Usage

i) to compare the working directory with the local repo:

$ git diff HEAD <filename>

git diff

command shows the file differences which are not yet staged

step1: modify any file before moving into commit the changes into local area

check the diff

India@LAPTOP-VODHAM22 MINGW64 ~/desktop/myclass (master)

$ git diff --base test1.txt

---

git reset

The purpose of the “git reset” command is to move the current HEAD to the commit specified (

git reset [file] . This is the purpose of the “–hard” option. In order to undo the last commit and discard all changes in the working directory and index, **execute the “git reset” command with the “–hard” option and specify the commit before HEAD (“HEAD~1”**

reset is the command we use when we want to move the repository back to a previous commit, discarding any changes made after that commit. This command unstages the file, but it preserves the file contents.

India@LAPTOP-VODHAM22 MINGW64 ~/desktop/myclass (master)

$ cat >>test1.txt

second commit changes

$ git add .

$ git commit -m "second commit"

$ git log

This command undoes all the commits after the specified commit and preserves the changes locally.

This command discards all history and goes back to the specified commit.

$git reset --hard <commitid>

$ git reset --hard 7f7aa6ee815305114699a9661f125c7fd6df1877

---

git rm

syntax: git rm <filename>

This command deletes the file from your working directory and stages the deletion.

India@LAPTOP-VODHAM22 MINGW64 ~/desktop/myclass (master)

$ git rm showfile

to get that file back to working directory

$ git reset --hard <commitid>

---

git show

Usage: git show [commit-id]

This command shows the metadata and content changes of the specified commit.

$ git show 30b2bf47055d462d458c5d9f45784fe0fd96408d

Show the list of commits in one line format:

git log --oneline

$git add *

or

$git add -A

or

$git add .

or

$git add --a

This command adds one or more to the staging area.

---

git branch -r

#git fetch

When downloading content from a remote repo, git pull and git fetch commands are available to accomplish the task. You can consider git fetch the 'safe' version of the two commands. It will download the remote content but not update your local repo's working state, leaving your current work intact.

git pull is the more aggressive alternative; it will download the remote content for the active local branch and immediately execute git merge to create a merge commit for the new remote content.

If you want to retrieve code without changing your current working copy of a repo, you should use the git fetch command.

git fetch only downloads new data from a remote repository - but it doesn't integrate any of this new data into your working files.

If you want to see changes from the master branch to the origin before you actually merge all the changes to your required repository, so you can use this command:

- In Github repo, make few changes in any file(services.txt)

(GitBash)

$ cat services.txt

Now fetch the latest changes from github repos

$git fetch

Now check the services.txt file, it will remain the same but new changes will not get reflect in it unless you merge the new changes into local repos

$git merge

Now, need to check the services.txt file, where new changes will reflect after merge the changes

check the file

$cat services.txt

#git fetch

Now you are able to see all the changes whatever done with your repository, are all possible using just by check the outing branch:

#git checkout origin/main

#git pull <repo_url>

git pull will change the code you have stored on your local machine. The pull command may overwrite changes you have made to the local copy of a repo.

once git pull is run, a merge operation is initiated. This merges operation makes your local working copy identical to the code git pull has retrieved.

to update your current HEAD branch with the latest changes from the remote server.

$ git pull https://github.com/cubeiplKumar/AIT-classroomrepo.git

---

git stash

- temporarily stores all the modified tracked files.
- git stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on.

1. Modify any existing file

$ git stash

Saved working directory and index state WIP on main: 2d87a75 newstory modified

2. git stash list

3. Again modify the file

git stash -m "Cloud updates"

4. git stash list

stash@{0}: On main: Cloud updates

stash@{1}: WIP on main: 2d87a75 newstory modified

5.git stash apply stash@{1}

5. add the file and commit

git add <filename>

git commit -m "applied new changes"

Modifies any file in the repository

#cat >sample.txt

save

#git add sample.txt

#git stash save

#git stash list

need to save the temp.changes into the original file then

#git stash pop

add the file and commit

#git add sample.txt

#git commit -m "new changes merged into the original file by dev3"

#git push -u origin main