Git & Github

17 November 2021 12:59

© Rajat Kumar
https://www.linkedin.com/in/imRajat/
https://github.com/im-Rajat

cd projects/my-project
git status
git init
git add .
git commit -m "Initial commit"
git commit -m "update"
git commit --allow-empty-message --no-edit

git remote add origin <remote repository URL>
Instead of above line use :
git remote set-url origin https://<githubtoken>@github.com/<username>/<repositoryname>.git

git push origin main
git push origin master
git push -f origin master

Commit without message : git commit -a --allow-empty-message -m ''

Empty commit : git commit --allow-empty -m "Updates"
git push origin <branch_name>

References :
https://www.git-tower.com/learn/git/faq/push-to-github
https://stackoverflow.com/questions/68775869/support-for-password-authentication-was-removed-please-use-a-personal-access-to
https://dev.to/github/how-to-undo-pushed-commits-with-git-2pe6

https://learngitbranching.js.org/

Check git configuration : git config --list
Commit any change : git commit
Create new branch : git branch [branchname]
Checkout branch : git checkout [branchname]
Create a new branch AND check it out at the same time : git checkout -b [branchname]
Merge branch : git merge [branchname]
Deataching head : git checkout [commitID]

Check all branch : git branch
Check current branch : git branch --show-current / git rev-parse --abbrev-ref HEAD

Edit a file : nano [filename] / gedit [filename]
Stage all changes/Add all files : add .
Commit changes : git commit -m "message"
Push changes : git push origin [branchname]
To pull only main branch : git pull origin main

If we see a yellow hint about "Pulling without specifying how to reconcile divergent branches" you can simply use one of the suggested commands to get rid of it the next time : (e.g. git config pull.rebase false).

Instead of using the git add . command to stage the file before committing we can commit it directly with the --all or -a option : git commit -a -m "message"

If git init is done from local, need to connect to github repository : git remote add origin [git url]

Check log : git log
Check log in 1 line : git log --oneline

https://profy.dev/project/github-minesweeper/cheatsheet
https://github.com/profydev/github-minesweeper-im-Rajat

Git Patch :

- To Create a patch :
  - git diff > patch_name.patch
- To use the patch and apply it :
  - Git apply patch_name.patch

Git Stash :

Do this can we have conflict while doing git pull

Git Commands for new User setup on terminal, and for branch change:

- `git config --list`
- `git config --global user.name <username>`
- `git config --global user.email <email>`
- `git clone https://github.com/im-Rajat/Git.git`

This will prompt for the password, please provide your password and it will clone your git repository there.

Git Commands for changing the branch (initally it would be in master branch):

- `git checkout <branch_name>`

After this you can make changes to the directory.

Git Commands for adding , commiting and pushing changes to your repository (or branch):

- `git add <folder_or_files_for_tracking>`
- `git commit –m "<description_for_your_commit>"`
- `git push –u origin <branch_name>`

If your push request is rejected then you can execute the below command to fetch the updates from remote repository and then run the push command.

- `git pull origin <branch_name>`

How to delete Directory/Multiples files:

- In the command-line, navigate to your local repository.
- Ensure you are in the default branch:
  git checkout <branch_name>
- The rm -r command will recursively remove your folder:
  git rm -r folder-name
- Commit the change:
  git commit -m "Remove duplicated directory"
- Push the change to your remote repository:
  git push origin <branch_name>

Git Cheat Sheet for commands:

SWTM-208
8_Atlassia...

Git Config

git config command
git config --global user.name "Your Name"
git config --global user.email "Your Email"

Staging Files

git add FILENAME
git add --all
git add -A
git add .
git commit -m "First Commit"

Git Environments

- Working
- Staging
- Commit

Restoring Files

git restore README.md
git restore .
git checkout .

- If we have added a file to git using add README.md, and we want to restore it from staging

Git Stash :

Do this can we have conflict while doing git pull

- Git stash
- Git pull
- Git stash apply

- Change in source call as per requirement
  (<<<<<<<< search this)

To stash
- git stash

To save a stash with a message:
- git stash -m "my_stash_name"

To list stashes:
- git stash list

To apply a stash:
- git stash apply
- git stash apply "stash@{n}"
- git stash apply n

To drop top hash, stash@{0}
- git stash drop

To drop specific stash - see git stash list
- git stash drop stash@{n}

Git Reset :

- git reset --hard

Git Merge

- Go to your local branch
- Git merge <aws_master> // name of merge you want to merge with

Update Forked Branch

If a new branch has been added to the original repository, it will not automatically appear in your forked repository. You will need to fetch the changes from the original repository and merge them into your forked repository.

To do this, you can follow these steps:

- Open your forked repository in a terminal or command prompt.
- Add the original repository as a remote by running the command: git remote add upstream <original_repository_url>. Replace <original_repository_url> with the URL of the original repository.
- Fetch the changes from the original repository by running the command: git fetch upstream.
- Merge the changes into your forked repository by running the command: git merge upstream/<branch_name>. Replace <branch_name> with the name of the branch that was added to the original repository.

After following these steps, the new branch should now be available in your forked repository.

```
git checkout <commit_hash>
git checkout -b <new_branch_name>  // old version
git switch -c <new_branch_name>  // new version
```

Git Repository details:
https://git.nagarro.com/root/NAGISVPR

Git Branch details:
Embedded_Networking

https://git.nagarro.com/root/NAGISVPR/tree/Embedded_Networking

Note: Please make sure VPN is connected otherwise git account is inaccessible.

```
# git restore .
# git checkout .
```

- If we have added a file to git using add README.md, and we want to restore it from staging then run : git restore --staged README.md.

- If we want to restore the changes back to original in README.md : git restore README.md

- Git restore only works with changes in a file, not with new created file, it won't delete it. Git isn't doing anything with untracked files. We need to delete the new file manually. But we deleted a new file and then restore it, it will work and restore the deleted file.

Logging
- Check log : git log
- Check log in 1 line : git log --oneline

Why Ignore Files?
- Sensitive info
- Personal notes
- System files

Use .gitignore
- Add file name here
- Add folder, folder must end with slash / in end
- Can add matters as well

Global ignore File
- git config - -global core.excludes file [file] (whenever we create a new project, it will pick up this file automatically)

Clearing the cache
- git rm r --cached .

Deleting File
- We can delete file using explorer of vs editor, after that we need to do staging (git add .) and git commit
- If we run git rm file_name.txt, it will delete the file as well to the staging. Will make things ready to commit
  - To restore above change :
    - We can run git restore -S .
    - Or we can run git restore --stage .
    - -S is same as --stage
    - After that we need to run git restore .
  - Git restore -S . is more restore staging
  - Git restore . Is for restoring the change/file.

Renaming File
- We can rename file using explorer of vs editor, after if we check git status it will show as new file created and old file deleted.
  - Just like linux system do renaming, create a copy with new name, and delete old one.
- If we restore and we need to delete the other file manually.
- Or we can use git command, git mv old_name.txt new_name.txt, this step also do the staging (git add .) itself.

Git always looking what we are doing based on last commit.

Differences
- Git diff

Changing History
Amending (use to modify changes without new commit, add change into last commit)
- command we use to add an item to a previous commit
- git commit --amend (redirect to editor to update the commit message if we want to)
- git commit -am "New commit message" (update the commit message if we want to in terminal itself)
- git commit -amend --no-edit (no edit means, leave the message same as what we did in last commit)
- git reset <commit-id> (unstage commit)
- git log --oneline (to check commit-id, will remove all commit from log after doing reset on a particular commit-id
- git reset --hard <commit-id> (dangerous command - this command will delete any commits, before the commit-id and also it will change all the files)

Rebasing (another way of changing history, and designed to take the commits from one branch and apply them to another)
- git rebase <branch>/<commit>
- git rebase --interactive <branch>/<commit>
- git rebase -i HEAD-2 (to back 2 step, redirect to editor)
- git rebase -i --root (redirect to editor)

Creating Branches
Looking at Branches
- git branch

Copying a branch
- git switch -c <NEW_BRNACH_NAME>
- git checkout -b NAME (slightly older version)

Merging
Will merge changes from one branch to current branch
- Git merge <branch>
- Git switch master (similar to git checkout master)

Deleting Branches

- git branch --delete NAME
- git branch -d NAME (as use this option as long as the branches are free of conflicts)
- git branch -D NAME (forces git to ignore things and just delete)

Git Flow

- Feature/fix branch
- Make changes
- Merge to master
- Delete old branch

Merge Conflict

- Need to choose which one of the branches we want to use or which one of the changes we want to use, and then accept everywhere in the project.

Git Stash & Git Clean

Stashing Code

- Git stash
- git stash list
- git stash apply
- git stash pop
- We can keep doing stash, it goes top of each other, like stack, latest on top
- git stash apply 0 (where 0 is stash number which we can get from command - git stash list, whichever stash number we want to restore)
- git restore . (restore changes to original)

Git Clean

- Git clean command lets us remove all un track files and directories from our branch super quickly.
- To remove old files that we don't need anymore.
- Remove untrack files
- Git clean -n # dry run (will show what will delete, good to run this first)
- Git clean -d # directories and subdirectories
- Git clean -f # force (to delete use this, when combine d and n)

Why GitHub?

- Cloud Repository
- Collaborative Development
- Project Management

Working with GitHub

- Set up remote
- Push
- Fetch/Pull

Create a new repository on the command line

- echo "# temp" >> README.md
- git init
- git add README.md
- git commit -m "first commit"
- git branch -M main
- git remote add origin https://github.com/im-Rajat/Git.git
- git push -u origin main

Push an existing repository from the command line

- git remote add origin https://github.com/im-Rajat/Git.git
- git branch -M main
- git push -u origin main

Remotes

- git remote add NAME URL (Very common name for the remote that we are using is origin)
- git remote remove NAME
- git rename OLDNAME NEWNAME
- git remote -v (-v means verbose, it will list all the remotes with a bunch of additional information for them)

Git Push

- Moves files from local to github
- git push REMOTE BRANCH
- git push --set-upstream-to origin main (--set-upstream-to is the longer version of -u)
- git push -u origin main # --set-upstream
- git push --all (push all local branches to github)
- git branch --set-upstream-to <origin/remote-branch>
- Git push origin master
- Git push -u origin master

Managing Projects with Github

- Contributors
- Issues
- Labels
- Miletones
- Projects

Syncing Github

- Clone
  - Git clone will take the copy of the GitHub repository, and place it on our local hard drive.
- Fetch
  - If we already have a copy of the repository on our local hard drive, to sync the information from GitHub to local, we can issue a git fetch command.
  - Fetch download information and brings that information to our local repo
  - It actually do anything with the information, we need to run git merge after running git fetch command.
- Pull
  - Git pull is a combination of doing a fetch, but also merging the data that is in our remote

with our local version.

References :
https://www.linkedin.com/learning/learning-git-and-github-14213624/https://github.com/LinkedInLearning/learning-git-github-2421501