Handwritten Notes on Git & GitHub

— by Aditya Jadhav

Git & Github notes

- by Aditya Jadhav

Defⁿ :- Git is free, open source version control system

* Version control system - VCS helps a software team manage changes to source code over time

- VCS keeps track of every modification to the code in special kind of database

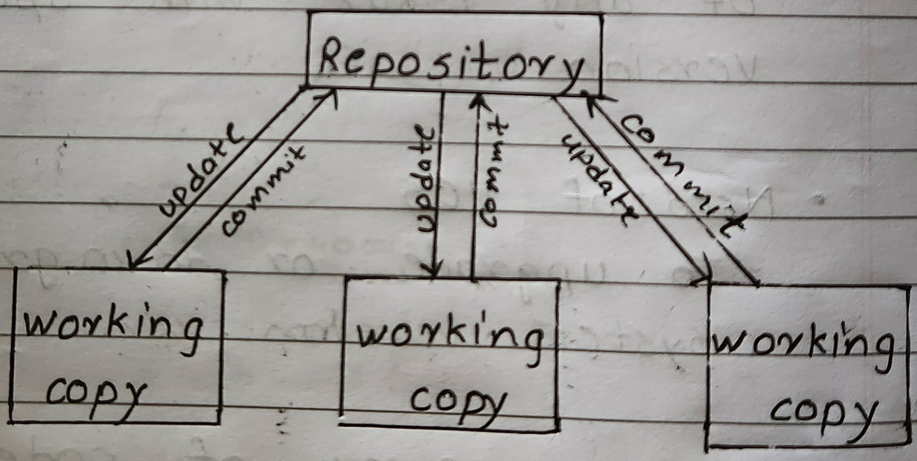
- VCS system helps team to roll back to previous version in case of any issue with specific version

- Need of VCS
  - To upgrade or down grade the system version
  - For recovery of code

- Types of VCS

1) Centralized Version Control System (CVCS)
2) Distributed Version Control System (DVCS)

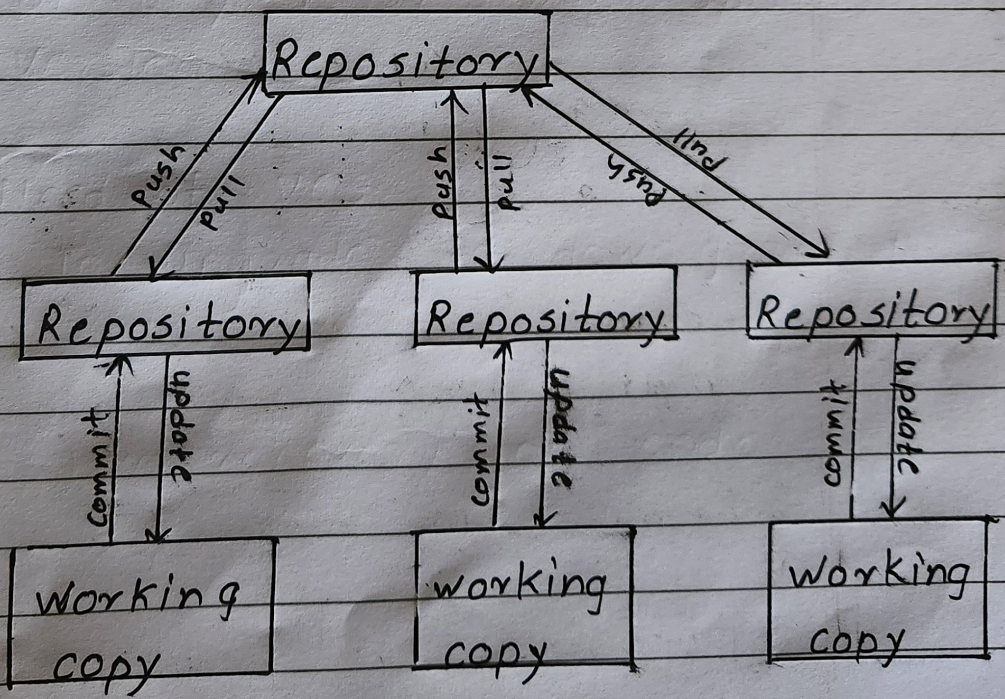
1) Centralized VCS
- CVCS uses central server to store all files and enables team collaboration
- CVCS workes on single repository to which users can directly access a central server



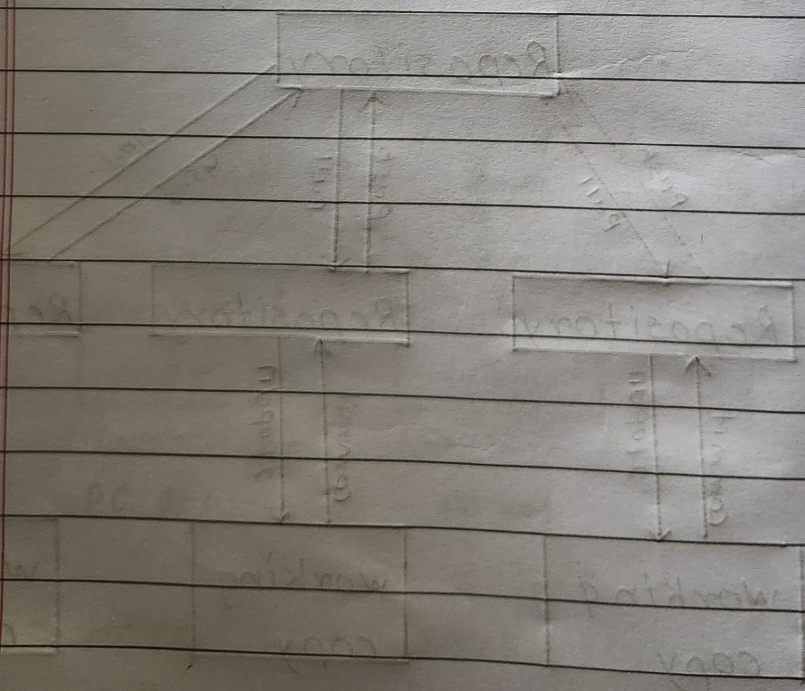
- Here each work station is connect with central code Repository

- Draw backs of CVCS
  i) It is not locally available
  ii) Crash of CVCS will result in losing the entire data of project.

2) Distributed VCS
- In Distributed VCS, every contributor has local copy or "clone" of the main repository
- User can change and commit local Repo without any interference
- User can update their local repo the central server



- Operations in DVCS are fast
- New changes can be done locally without manipulating the central data
- If the central ~~data~~ server gets crashed at any point of time, the lost data can be easily recovered from any one of the contributor's local repositories.



Install GIT

https://git-scm/downloads

Install GIT for mac OS

There are several options for installing git on mac OS.
Any non-source distribution are provided by third parties, and may not be up to date with latest source release

choose one of the following options for installing GIT on mac OS

i) Homebrew
Install homebrew if you don't already have it, then:-
$ brew install git

ii) Mac Ports
install MacPorts if you don't already have it, then
$ sudo port install git

iii) Xcode
Apple shipes a binary package of Git with Xcode

Install GIT for windows

i) Standalone Installer
- 32-bit Git for windows setup
- 64-bit Git for windows setup

ii) Portable ("thumbdrive edition")
- 32-bit Git for windows ~~setup~~ portable
- 64-bit Git for windows portable

iii) Using winget tool

winget install --id Git.Git -e --source ~~wid~~ winget

- Launch Git in windows

To launch Git Bash open windows start menu, type git bash and press Enter

You can execute git commant
'git --version'
To check the version

Configure User Information

suppose we are working on project with 100+ teammembers / Developer and they are pushing their code in remote repo so to identify which code commit is done by which user. That's why we need to configure user information

- config user

```
git config --global user.name "Aditya"

git config --global user.email "abc@xyz.com"
```

- Get user information

```
git config --list
```

- Update username

```
git config --global user.name "Aditya Jadhav"
```

classmate
Date
Page

* Create GIT repository

step 1 :- create empty project

step 2 :- navigato to project file
for eg cd project-name

step 3 :- use command
'git init'

This commands create hidden file .git

* GIT commit

- To add information about untracked file we use 'git commit'
- for eg.
we added a new file on our project ie file.txt
- and we check by command
'git status'

we will get
On branch master
No commits yet
Untracked files:
file.txt

- to add this file on staging are we use
  'git add file.txt'

No we check status by
  git status

we get
  No commits yet
  changes to be commited :-
    new file : file.txt
  Untracked files:

- to add all files we use
  'git add .'

Now for commit we use
  'git commit -m "message"'
  purpose of commit

Now file is being tracked
suppose any modification happan the we check status it will show
  modified : file.txt
To checkoute changes made we can use
  'git diff'
The we can add and commit again that file.

* GIT Tracking & GIT LOGs

- suppose we add new file named sfile.txt

- Then when we check status it shows

  tracked files:

  file.txt

  untracked files:-

  sfile.txt

- To keep track of second file we can use command

  'git add sfile.txt'

  or

  'git add .

- So using 'git status' we can keep track of working directory

'git log'

This command is used to show last 10 commits which contains commit id, author, date

If we want to see commit from specific author we can use command like

git log --author = "Aditya Jadhav"

author name or email id

'git log --online'

used to show all commits

~~'git log --online'~~

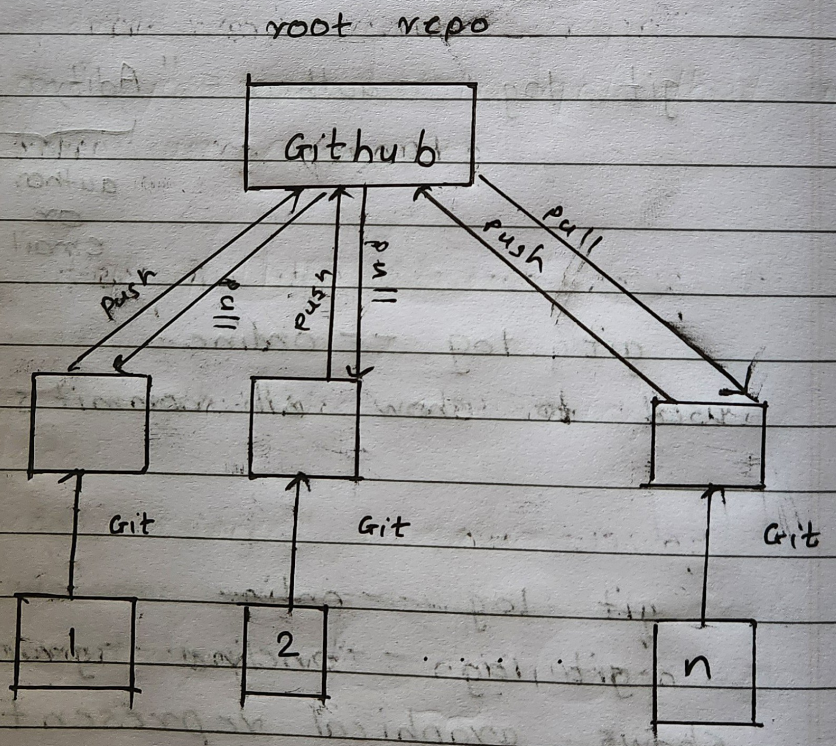
'git log --oncline --graph'

shows graphical representation

Introduction to GIT Hub

- Github is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code

- Github is remote repository for your code base



- website link

  https://github.com

- Create remote repo on github

  - Go on '+' icon select new repository
  - declare repository name
  - Add description (optional)
  - select a Public or Private
  - click on 'create Repositary'

- Some additional features

  - README file
    This is where you can write a long description for your project

  - .gitgnore
    choose which files not to track from a list of templates

Date
Page

- for cloning the git repo use command

'git clone -b main https://github.com/repo.git'

branch main

Github repo link

- Now when the new file has been updated on remote repo and you want to get that updaked file use command

'git pull'

- same way we can change file from local we can use comman

'git commit -am "message"'

combination of add command and commit command

now to upload the updates on remote repo use command

'git push'

It ask for username & password the it will push

* setup Authentication in Local & Github

Don't need to provide username and password

we can use 2 keys ie SSH and GPG keys

Here are steps to use SSH keys

step 1 :- Go to profile setting

step 2 :- select SSH and GPG keys

step 3 :- Use command for SSH key generation

'ssh-keygen -t ed25519 -C "email@xyz.com"'

step 4 :-

It will provide 2 keys

privet key & public key

step 5 :- cat /path

path of pubic key

It will provide complete public key path

step 6 :- Then add that path in SSH keys in github

step 7 :- To test SSH key use command
'ssh -T git@github.com'

step 8 :- Now ~~af~~ for first time while push it will ask for username and password. Then you don't need to enter that again.

* Branches in GIT

- Get to know all branches
'git brach'

- Create new branch

'git branch branch-name'

- Switch between branch

'git switch branch-name'

- After modification in new branch to push code ~~or~~ in new branch we can use

'git push origin branch-name'

- The file from one branch cannot be accessed from another branch

- Merge branch on github

- select pull request
- select new pull request
- select from which branch you wat to send pull request
- for eg

main ← new-branch

- selet create pull request
- select merge pull request.

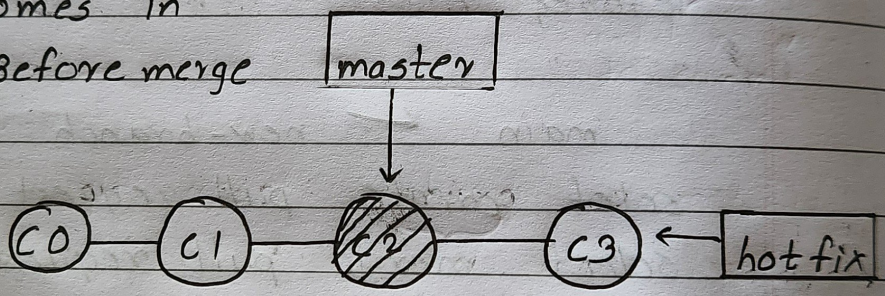
Date
Page

* merge branches in GIT

- Isolating features into different branches is crucial pratice for any serious developer

- At some point a piece of code will reach a state where you'll want to integrate it with the rest of project This is where the git merge command comes in

Before merge



After merge


```
git switch master
git merge hotfix
git commit
git push
```

classmate
Date
Page

* Revert in GIT

- Revert is all about to Undo the changes, you did in repo
- In Git this can be done via RESET and REVERT.

- **RESET** :-
  - Physically, user can think of it as a "roll back".
  - Reset points local environment back to previous commit.

- **REVERT** :-
  - Net effect of git revert command is similar to reset, but its approach is different.
  - Revert add a new commit at the end of chain to "cancel" changes.

- Revert or Reset ?
  - If user have already pushed commits to remote repo, a revert is nicer way to cancel out changes.
  - Git workflow works well for picking up additional commits at the end of a branch, but it can challenging if a set of commits is no longer seen in the chain when someone resets the branch

pointer back

- If commit in local then reset is good If commit is pushed then revert is good option

- Git reset

To use git reset we can use two commands

'git reset --soft commit-id'

--soft: Reset commit history

commit-id: until you want to commit

or

'git reset --hard commit-id'

--hard: Resets action as well as history

- Git revert

to use git revert we use command

'git revert HEAD'

* Comparison in GIT

i) Compare staging area & working Directory

when we do any changes in repo we need to compare them so we get to know what kind of changes/ commits has been made

For comparing working directory we use command

'git diff file_name'

we will get changes made on new commit
for eg.
previous changes
+ new changes

For comparing entire staging area we use

'git diff'

ii) compare commits in GIT

for comparison of two commits we use command

'git diff commit1-id commit2-id'

* Git stash *

Git stash temporarily saves 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.

Stashing is handy if you need to quickly switch context and work on something else, but you're mid-way through code change and aren't quite ready to commit

Commands for git stash

'git stash'

'git stash -a'

adds untracked file

'git stash pop' } remove from stash

'git stash apply'

* More command in GIT

- Help

  'git help'

- Move file

  'git mv file.txt path/path.txt'

- Remove file

  'git rm file.txt'

- View

  'git status'

- Blame (discover why and when updated)

  'git blame file.txt'

- Aborting

  'git checkout -- file.document.txt'

- Rebashe

  Rebasing is the rewiding of existing commits on branch with intent of moving the "branch start point" forword, then replying the ~~rewind~~ reward commits. This allows developers to test their branches change safely on their private branch just as they were made on top of mainline code, including any recent mainline fixes

  'git rebash <branch_name>'

- Tagging

  'git tag <tag-name>'

- Remote

  'git remote -v'