VCS:

------

🡪 Version Control System is helps a software team manage changes to source code over time.

* It keep tracks the every modification to the code in a special kind of DB.
* If a mistake is made, developer can easily turn back the clock and compare the earlier version of the code.

Basic Terminologies:

* Repository (repo): DB storing the files.
* Server: The computer storing the repos.
* Client: The computer connecting to repos.
* Working set/copy : Your local directory of files, where you make changes.
* Trunk/Main: The primary location for code in the repo. Trunk is the main line in a tree structure code.

GIT:

GIT is distributed version control system, it manages source code. It helps S/W developer to work together and maintain complete history of their work.

It is a mature, actively maintained open source project originally developed by Linus Torvalds.

Step-1:

----------

* Register/Signup in github.

Step-2:

* Create a public repository.

Step3:

* Install Git:

1. Yum install git
2. <https://github.com/git-for-windows/git/releases/download/v2.19.1.windows.1/Git-2.19.1-32-bit.exe>
3. Git init 🡪 Create repo locally.
4. Git clone <repo name> 🡪 Clone the existing repo to local machine
5. Git config –list
6. Git config user.name “name” 🡪 Define the author name
7. Git config user.email “email Id” 🡪 Author email
8. Git config –global alias.<alias name> <git\_command>
9. Git add <file\_name)> 🡪 Add files to repo
10. Git commit –m “Commit message” 🡪 commit change

Git configure:

-----------------

* #git config –global user.name “name” 🡺 Set global user name
* #git config –global user.email “email” 🡪 Set global email
* #git config –list/#git config –global –list 🡪 show the configuration
* #git config –global color.ui auto 🡪 Set colour for git

Create repo:

---------------

* #mkdir “git\_practice”
* #cd git\_practice
* #git init <proj\_name> 🡪 You must initialize before doing git project/new repository
* #git clone <url> 🡪 Clone new repository

Change:

----------

* #git status 🡪 check the status of git list all new/modified files to be committed.
* #git diff 🡪 shows file difference
* #git add <file> 🡪 Add a file to git (It add file to staging state only)
* #git commit –m “Desc\_message” 🡪 (Add files staging area to local repos)

Remote repository:

---------------------------

* #git remote add origin <git\_url> 🡺 Add remote repository
* #git remote –v 🡪 check the repository
* #git push origin master 🡪 Push the committed code to github
* #git pull origin master 🡪 Pull latest code from remote repos.

Git Conflict:

---------------

If two developers are working in same code , 1st developer add/edit a codes, committed and push the code to remote repository at the same time 2nd developer is also add/edit code, committed and trying to push the code to remote repository that time git conflict happens.

To overcome git conflict 1st need to pull the latest code and remove the conflict line (git status) then add, commit and push the code.

Git Branch:

--------------

It is essentially an independent line of development. You can take advantage of branching when working on new features.

#git branch 🡪 List out the branches

#git branch –a 🡪 List out all branches

#git branch –list 🡪 List branches

#git branch <New\_Branch\_Name> 🡪 Create a new branch

#git checkout –b <New\_Branch\_Name> 🡪 Create a new branch

#git checkout <Switch Branch\_Name> 🡪 Switched to new branch

#git merge <Merging New\_Branc\_Name> 🡪 Suppose merge master branch to new\_branch

#git branch –d <Branch\_Name> 🡪 Delete a local branch

#git push origin –d <Remote\_Branch\_Name> 🡪 Delete remote branches

#git fetch 🡪 Fetch the newly created remote branch to local repos.

#

Git Tag:

Point to specific point in git history, tags are generally used to capture a point in history that is used to mark version.

* To mark release point of your code/data

1. Check out the branch where you want to create a tag:

* #git checkout “branch\_name”

1. Create a tag:

* #git tag “tag\_name”
* #git tag –a v1.0 –m “ver 1 of ..” (create annotated tag)

1. Display/show the tags

* #git tag
* #git show <version>
* #git tag –l “v1.\*”

1. Push tags to remote

* #git push origin v1.0
* #git push origin –tags
* #git push --tags

