Day1

syllabus

0. Introduction Git,

- Introduction Git repository and git structure,
- Adding code to git,
- Introduction to GitHub,
- Creating pull requests,

1. VCS - Version Control System
2. Git - Distributed VCS .

- developed by Linus Torvalds
- it uses version method
- makes copies of files at every commit
- other VCS on commit saves the changes done in the file (difference in files versions called delta)

3. cryptography authentication history (SHA)
4. GIT
    - here , master branch managed by project manager

demo

1. `git config --global --list`
- `git config --global user.name`
- git config help

```
git-config - Get and set repository or global options

SYNOPSIS
       git config [<file-option>] [type] [--show-origin] [-z|--null] name
[value [value_regex]]
       git config [<file-option>] [type] --add name value
       git config [<file-option>] [type] --replace-all name value
[value_regex]
       git config [<file-option>] [type] [--show-origin] [-z|--null] --get
name [value_regex]
       git config [<file-option>] [type] [--show-origin] [-z|--null] --get-
all name [value_regex]
       git config [<file-option>] [type] [--show-origin] [-z|--null] [--
name-only] --get-regexp name_regex [value_regex]
       git config [<file-option>] [type] [-z|--null] --get-urlmatch name
URL
```

- `git status -s gives`
- 1. ?? (untracked)
- 2. A _ (Add to staging area)
- 3. M _ (file modified, added to staging area)[color:green]
- ◦ i.e add is done
- `git add ___.file`
- 4. _ M (file modified, but not staged)[color:red]
- 5. U U - files have conflict
- `git log`
  - ◦ check your activity
- `git log --oneline --graph --decorate`
  - ◦ to get graphical representation of files
- `git diff`
  - ◦ givers difference between two version of a file
  - ◦ modified content
- `git add . or git add -A`

Day2

2. demo on how to revert back /previous version of source folder and files in it / repo folder on PC

- works only when **changes not moved to staging area**

`git checkout . (for all) git checkout file.extension (for particular file )`

- eg.
- `git checkout index.html`

3. if changes moved to staging area then use command

- to reset to version before staging area

`git reset --hard`

4. git stash area

`git stash`

- move your data from your file to a temporary stack area, called stash
- it moves all modified content , after previous commit

`git stash list`

- list the changes as array element

> git stash pop git stash apply

- to bring back the modified content from stash to your file
- > git stash pop 0
- gives [0] entry in stash
- > git stash show -p
- it shows your content in stash
- > git stash clear
- to clear stash from all elements(containing files data)
- git stash commands

```
usage: git stash list [<options>]
   or: git stash show [<stash>]
   or: git stash drop [-q|--quiet] [<stash>]
   or: git stash ( pop | apply ) [--index] [-q|--quiet] [<stash>]
   or: git stash branch <branchname> [<stash>]
   or: git stash save [--patch] [-k|--[no-]keep-index] [-q|--quiet]
                      [-u|--include-untracked] [-a|--all] [<message>]
   or: git stash [push [--patch] [-k|--[no-]keep-index] [-q|--quiet]
                       [-u|--include-untracked] [-a|--all] [-m <message>]
                       [-- <pathspec>...]]
   or: git stash clear
```

- to get documentation on a command

> git stash --help

5. Branching

- 1. it is another line of development
  - master branch contains
  - 1. lastest changes /to code
  - 2. stable code/functionality
  - 3. non bugged code/compilable code
  - 4. non -crashing / no exception
  - other than master branch ,all branches are called **features branch**
  - only one master branch in one repo
  - separate commit for separate branches
  - directory specific features common for all branches, i.e stash,virtual directory,staging area
- 2. demos
  - 1. to show which branches are there in git repo

- git branch
- 2. to create a branch
- git branch branchName
- 3. to switch to a branch with name branchName, use command
- git checkout branchName
- 4. to delete a branch use
- git branch -d branchName
- 5. git branch doucmentation
- git branch --help

```
git-branch - List, create, or delete branches

SYNOPSIS
       git branch [--color[=<when>] | --no-color] [-r | -a]
               [--list] [-v [--abbrev=<length> | --no-abbrev]]
               [--column[=<options>] | --no-column] [--sort=<key>]
               [(--merged | --no-merged) [<commit>]]
               [--contains [<commit]] [--no-contains [<commit>]]
               [--points-at <object>] [--format=<format>] [<pattern>...]
       git branch [--track | --no-track] [-l] [-f] <branchname> [<start-
point>]
       git branch (--set-upstream-to=<upstream> | -u <upstream>)
[<branchname>]
       git branch --unset-upstream [<branchname>]
       git branch (-m | -M) [<oldbranch>] <newbranch>
       git branch (-c | -C) [<oldbranch>] <newbranch>
       git branch (-d | -D) [-r] <branchname>...
       git branch --edit-description [<branchname>]
```

6. Merging branches

- 1. merging two branches
- 2. demos to merge feature branch to master
- 1. go to master branch

  git checkout master;

- 2. use command

  git merge branchName; git merge courses;

- 3. deleteing branches after merge(optional)

> git branch -d branchName git branch -d courses

- 3. git merge doc

> git merge --help

```
SYNOPSIS
       git merge [-n] [--stat] [--no-commit] [--squash] [--[no-]edit]
               [-s <strategy>] [-X <strategy-option>] [-S[<keyid>]]
               [--[no-]allow-unrelated-histories]
               [--[no-]rerere-autoupdate] [-m <msg>] [<commit>...]
       git merge --abort
       git merge --continue
```

7. Branching advanced command

- 1. create a branch and switch to that branch use command

> git checkout -b branchName

- e.g

> git checkout -b about-us

8. Merge conflict

- two branches having change on same line
- when we try to merge these brnaches , confict part is shown for 2nd branch
- to resolve the conflict, developer has to make the decision,either
  - 1. either let it be
  - 2. or delete changes
  - then , we have to manually commit ,the master branch 3. > git add . 4. > git commit -m "brnach added"

9. A distributed version control system (DVCS)

- different DVCS, server for shared repo/remote repo
  - 1. GitHub
  - 2. GitLab
  - 3. BitBucket
- each git repository contain
- 1. code
- 2. meta data

10. GITHUB

- 1. create a repo on github
- 2. add it to local PC

- `git remote add origin https://github.com/porjesuraj/Sunbeam-website.git`
- 3. to get info about remote repository on local PC
- `git remote -v`
- 4. now if you create a branch , and push the changes to github remote repo from the feature branch ,
- `git add .`
- `git commit -m ""`
- `git push -u origin feature_branch`
- e.g
    - `git push -u origin attendance-report`
- after push, on github website,now we have created a merge request on github
    - now the person managing master can use this request
    - and merge it in origin reposity i.e remote repository
- 5. if you want to pull a feature this branch, i.e present in remote repository(github.lab) -we need to be using command

```
git pull origin feature_branch;
```

- 6. to check files content :
- `vim filename.extension`
- `cat filename.extension`

the remote repository like on github and gitlab is absolute clone of local repository

Questions

1. What is GIT?

- Git is an open source distributed version control system and source code management SCM.
- system with an insistence to control small and large projects with speed and efficiency.

2. Which language is used in Git?

- Git uses 'C' language. Git is quick, and 'C' language makes this possible by decreasing the overhead of run times contained with high-level languages.

3. What is a repository in Git?

- A repository consists of a list named .git, where git holds all of its metadata for the catalog. The content of the .git file is private to Git.

4. What is 'bare repository' in Git?

- A "bare" repository in Git includes the version control information and no working files (no tree., and it doesnt include the special git sub-directory.
- Instead, it consists of all the contents of the .git sub-directory directly in the main directory itself, whereas working list comprises of:
    - A .git subdirectory with all the Git associated revision history of your repo.
    - A working tree, or find out copies of your project files.

5. What is the purpose of GIT stash?

- GIT stash takes the present state of the working file and index and puts in on the stack for next and gives you back a clean working file.
- So in case if you are in the middle of object and require to jump over to the other task, and at the same time you don't want to lose your current edits, you can use GIT stash.

6. What is GIT stash drop?

- When you are done with the stashed element or want to delete it from the directory,

> run the git 'stash drop' command.

- It will delete the last added stash item by default, and it can also remove a specific topic if you include as an argument.

7. What are the advantages of using GIT?

- Here are some of the essential advantages of Git:
    - Data repetition and data replication is possible
    - It is a much applicable service
    - For one depository you can have only one directory of Git
    - The network performance and disk application are excellent
    - It is effortless to collaborate on any project
    - You can work on any plan within the Git

8. What is the function of 'GIT PUSH' in GIT?

- 'GIT PUSH' updates remote refs along with related objects

9. Why do we require branching in GIT?

- With the help of branching, you can keep your branch, and you can also jump between the different branches.
- You can go to your past work while at the same time keeping your recent work intact.

10. What is the purpose of 'git config'?

- The 'Git config' is a great method to configure your choice for the Git installation.
- Using this command, you can describe the repository behavior, preferences, and user information.

11. What is the definition of "Index" or "Staging Area" in GIT?

- When you are making the commits, you can make innovation to it, format it and review it in the common area known as 'Staging Area' or 'Index'.

12. What is a 'conflict' in git?

- A 'conflict' appears when the commit that has to be combined has some change in one place, and the current act also has a change at the same place.
- Git will not be easy to predict which change should take precedence.

13. What is the difference between git pull and git fetch?

- Git pull command pulls innovation or commits from a specific branch from your central repository and updates your object branch in your local repository.
- Git fetch is also used for the same objective, but it works in a slightly different method.
    - When you behave a git fetch, it pulls all new commits from the desired branch and saves it in a new branch in your local repository.
    - If you need to reflect these changes in your target branch,
    - git fetch should be followed with a git merge.
    - Your target branch will only be restored after combining the target branch and fetched branch. To make it simple for you, remember the equation below:

> Git pull = git fetch + git merge

14. How to resolve a conflict in Git?

- If you need to resolve a conflict in Git,
    - edit the list for fixing the different changes, and
    - then you can run "git add" to add the resolved directory,
    - and after that, you can run the 'git commit' for committing the repaired merge.

15. What is the purpose of the git clone?

- The git clone command generates a copy of a current Git repository.
- To get the copy of a central repository, 'cloning' is the simplest way used by programmers.

16. What is git pull origin?

- pull is a get and a consolidation.
- 'git pull origin master' brings submits from the master branch of the source remote (into the local origin/master branch), and then
- it combines origin/master into the branch you currently have looked out.

17. What does git commit a?

- Git commits "records changes to the storehouse"
- while git push " updates remote refs along with contained objects"
- So the first one is used in a network with your local repository,
- while the latter one is used to communicate with a remote repository.

18. Why GIT better than Subversion?

- GIT is an open source version control framework;
- it will enable you to run 'adaptations' of a task, which demonstrate the changes that were made to the code over time also it allows you keep the backtrack if vital and fix those changes.

- Multiple developers can check out, and transfer changes, and each change can then be attributed to a particular developer.

19. Explain what is commit message?

- Commit message is a component of git which shows up when you submit a change.
- Git gives you a content tool where you can enter the adjustments made to a commit.

20. Why is it desirable to create an additional commit rather than amending an existing commit?

- There are couples of reason
    - The correct activity will devastate the express that was recently saved in a commit.
    - If only the commit message gets changed, that's not a problem.
    - But if the contents are being modified, chances of excluding something important remains more.
    - Abusing "git commit- amends" can cause a small commit to increase and acquire inappropriate changes.

21. What does 'hooks' comprise of in Git?

- This index comprises of Shell contents which are enacted after running the relating git commands.
- For instance, Git will attempt to execute the post-commit content after you run a commit.

22. What is the distinction between Git and Github?

- 1. Git is a correction control framework, a tool to deal with your source code history.
- 2. GitHub is a hosting function for Git storehouses.
    - GitHub is a website where you can transfer a duplicate of your Git archive.
    - It is a Git repository hosting service, which offers the majority of the distributed update control and source code management (SCM) usefulness of Git just as including its features.

mcq links

Git Mcq