GIT CLASS 3

GIT BRANCHES:

- A branch represents an independent line of development.
- The git branch command lets you create, list, rename, and delete branches.
- The default branch name in Git is master.
- allows you to work on different features or changes to your code independently, without affecting the main or other branches.
- It's a way to organise and manage your code changes, making it easier to collaborate and maintain your project.

COMMANDS:

git branch	used to see the list of branches
git branch branch-name	to create a branch
git checkout branch-name	to switch one branch to another
git checkout -b branch-name	used to create and switch a branch at a time
git branch -m old-branch new-branch	used to rename a branch
git branch -d branch-name	to delete a branch
git branch branch-name deleted-branch-id	Used to get deleted branch id
git branch -D branch-name	to delete a branch forcefully

The -d option will delete the branch only if it has already been pushed and merged with the remote branch. Use -D instead if you want to force the branch to be deleted, even if it hasn't been pushed or merged yet. The branch is now deleted locally.

Now all the things you have done is on your local system.

GIT MERGE:

Git merge is a command used in the Git version control system to combine changes from one branch.

To merge: git merge branch_name

GIT CHERRY-PICK:

Git cherry-pick is a command in Git that allows you to take a specific commit from one branch and apply it to another branch. It's like picking a cherry (commit) from one branch and adding it to another branch, allowing you to selectively copy individual commits without merging the entire branch.

Command: git cherry-pick commit_id

GIT MERGE CONFLICTS:

GIT makes merging super easy!

CONFLICTS generally araise when two people two people have changed the same lines in a file (or) if one developer deleted a file while another developer is working on the same file!

In this situation git cannot determine what is correct!

Lets understand in a simple way!

cat>file1 : hai all

add & commit

git checkout -b branch1

cat>file1 : 1234

add & commit

git checkout master

cat>>file1 : abcd

add & commit

git merge branch1 : remove it

git diff file1

vim file1

git add

git commit -m "final commits"

NOTE: Don't give file name on commit

Identify Merge Conflicts:

see the file in master branch then you will see both the data in a single file including branch names that are dividing with conflict messages

Resolve Conflicts:

open file in VIM EDITOR and delete all the conflict dividers and save it!

add git to that file and commit it with the command (git commit -m "merged and resolved the conflict issue in abc.txt")