GIT:

1) **Defintion:**
    1) VCS(Version Control System)
    2) Types:
        1) Local:
            1) RCS
        2) Centralised:
            1) CVS, Subversion
            2) One single repo
            3) Checkouts are done and code is pushed, no history of commits is kept
        3) Distributed:
            1) Git
            2) Github, AWS Codecommit, BitBucket, GitLab etc.
            3) Helps us in working in our local environment
    3) Why version control:
        1) So that there is no downtime and in-case any problems we can revert back to working version
2) **Basics:**
    1) If free and open-source
    2) Deigned by Linus Trovalds
    3) Not dependent on network access on a central server
    4) Goals:
        1) Speed
        2) Support for non-linear development
        3) fully distributed
        4) Able to handle large projects efficiently
3) **Local Git Areas:**
    1) Git Directory:
        1) When we clone the git project
    2) Working Directory:
        1) When we make any changes/ create a new file
        2) git status will show changed/created files in red-> not staged
    3) Staging Area:
        1) git add <<file-name>> -> the particular file
        2) git add . -> add all the files
        3) Git status -> show the changed/created file in green -> it has been added to staging area
    4) Repository:
        1) After adding files in staging area using git add
        2) git commit -m “commit-message”
    5) Remote(Github):
        1) git push origin master
        2) Generically -> git push origin <<branch-name>>

4) **Initial Git Config:**
   1) When we do git clone, git push , git pull-> it will prompt for git’s username and password
   2) Instead of mentioning it again and again
   3) We can:
      1) git config —global user.name=“<<value>>”
   4) To view all the configuration
   5) git config -list
5) **How to create a Git repo:**
   1) Locally:
      1) git init
      2) Add a .git file in your folder making it a repo
   2) Clone:
      1) git clone “url/dir”
6) **Important git commands:**
   1) git clone url/dir:
      1) This command will copy a Git repository so you can add/use it
   2) git add <<file-name>> OR git add **.**
      1) Add the changes to the staging area
   3) git commit
      1) Helps us to record a snapshot of the staging area and assign a commit message to it -> Repository
   4) git diff
      1) Show us the difference between what is staged and what is modified but unstaged
   5) git help <<command-name>>
      1) It will show us the info about the command-name passed
   6) git pull
      1) Fetch from a remote repo(GitHub repo) and try to merge it in our current local branch
      2) Git fetch + merge
   7) git fetch
      1) Import the commits, branches from the remote repo(GitHub) to the local repo
      2) It does not merge
   8) git push
      1) Push our new branches and date to a remote repo(GitHub)
   9) git revert
      1) Is for undoing shared public changes
   10) git reset
      1) Is for undoing local private changes
      2) git reset HEAD -filename
      3) Option:
         1) —soft: from repository -> staging
         2) —mixed: from repository -> working

        3) —hard: from repository -> bin

11) git log
    1) Viewing the commit history
12) git checkout:
    1) git checkout —filename : remove all the uncommitted changes in the file
    2) git checkout —. : undo all the uncommitted change
13) Branching:
    1) To create a new local branch:
        1) git branch <<branch-name>>
    2) To list all local branches:
        1) git branch
    3) To list all remote branches:
        1) git branch -r
    4) To list all the local and remote branches:
        1) git branch -a
    5) To switch to a local branch
        1) git checkout <<branch-name>>
    6) Create a new branch and checkout to that new branch:
        1) git checkout -b <<branch-name>>
    7) Branch deletion:
        1) git branch -d <<branch-name>>
        2) git branch -D <<branch-name>>
            1) It will delete branches which have not been pushed/merged
    8) Remote Branch deletion:
        1) git push <remote> —delete <branch>
        2) Shorten:
            1) git push <remote>:<branch>
14) Merge Conflicts:
    1) May occur when conflicting changes re made in the same file, or when a person is trying to edit an already deleted file
    2) The conflicted file will have line marked with -> <<< or >>> symbols
    3) We have to go line by line and check which incoming change is better
15) git rebase <<branch-name>>
16) git stash
17) git stash pop