Git

1. What is Source Code Management?

It is a process through which we can store and manage any code. Developers write code, Testers write test cases and DevOps engineers write scripts. This code, we can store and manage in Source Code Management. Different teams can store code simultaneously. It save all changes separately. We can retrieve this code at any point of time.

----

2. What are the Advantages of Source Code Management?

. Helps in Achieving team work
. Can work on different features simultaneously
. Acts like pipeline b/w off shore & on shore teams
. Track changes (Minute level)
. Different people from same team as well as different teams can store code simultaneously (Save all changes separately)

----

3. Available Source Code Management tools in the market?

There are so many Source Code Management tools available in the market. Those are

. Git
. SVN
. Perforce
. Clear case

Out of all these tools, Git is the most advanced tool in the market where we are getting so many advantages compare to other Source Code Management tools.

----

4. What is Git?

Git is one of the Source Code Management tool where we can store any type of code. Git is the most advanced tool in the market now. We also call Git is version control system because every update stored as new version. At any point of time, we can get any previous version. We can go back to previous versions. Every version will have its unique number. That number we call commit-ID. By using this commit ID, we can track each change i.e. who did what at what time. For every version, it takes incremental backup instead of taking whole backup. That's why Git occupies less space. Since it is occupying less space, it is very fast.

----

5. What are the advantages of Git?

.Speed:-

Git stores every update in the form of versions. For every version, it takes incremental backup instead of taking whole backup. Since it is taking less space, Git is very fast. That incremental backup we call "Snapshot"

.Parallel branching:-

We can create any number of branches as per our requirement. No need to take prior permission from any one unlike other Source Code Management tools. Branching is for parallel development. Git branches allow us to work simultaneously on multiple features.

.Fully Distributed:-

Backup copy is available in multiple locations in each and every one's server instead of keeping in one central location unlike other Source Code Management tools. So even if we lose data from one server, we can recover it easily. That’s why we call GIT as DVCS (Distributed Version Control System)

----

6. What are the stages in Git?

There are total 4 stages in Git

1. Work space:-

It is the place where we can create files physically and modify. Being a Git user, we work in this work space.

2. Staging area/Indexing area:-

In this area, Git takes snapshot for every version. It is a buffer zone between workspace and local repository. We can't see this region because it is virtual.

3. Local repository:-

It is the place where Git stores all commits locally. It is a hidden directory so that no one can delete it accidentally. Every commit will have unique commit ID.

4. Central repository:-

It is the place where Git stores all commits centrally. It belongs to everyone who are working in your project. Git Hub is one of the central repositories. Used for storing the code and sharing the code to others in the team.

----

7. What is the common branching strategy in Git?

.Product is same, so one repo. But different features.
.Each feature has one separate branch
.Finally merge (code) all branches
.For Parallel development
.Can create any no of branches
.Can create one branch on the basis of another branch
.Changes are personal to that particular branch
.Can put files only in branches (not in repo directly)
.Default branch is “Master”
.Files created in workspace will be visible in any of the branch workspace until you commit. Once you commit, then that file belongs to that particular branch.

----

8. How many types of repositories available in Git?

There are two types of repositories available in Git

.Bare Repositories (Central)
These repositories are only for Storing & Sharing the code
All central repositories are bare repositories

.Non – Bare Repositories (Local)
In these repositories we can modify the files
All local /user repositories are Bare Repositories

----

9. Can you elaborate commit in Git?

.Storing file permanently in local repository we call commit.
.For very commit we get one commit ID
.It contains 40 long Alpha-numeric characters
.It uses the concept "Check some" (It's a tool in Linux, generates binary value equal to the data present in file)
.Even if you change one dot, Commit-ID will get changed
.Helps in tracking the changes

----

10. What do you mean by "Snapshot" in Git?

.It is a backup copy for each version git stores in repository.
.Snapshot is an incremental backup copy (only backup for new changes)
.Snapshot represents some data of particular time so that, we can get data of particular time by taking that particular snapshot
.This snapshot will be taken in Staging area in Git which is present between Git workspace and Git local repository.

----

11. What is GitHub?

Git hub is central git repository where we can store code centrally. Git hub belongs to Microsoft Company. We can create any number of repositories in Git hub. All public repositories are free and can be accessible by everyone. Private repositories are not free and can restrict public access for security. We can copy repository from one account to other account also. This process we call as "Fork". In this repository also we can create branches. Default branch is "Master"

----

12. What is Git merge?

By default, we get one branch in git local repository called "Master". We can create any no of branches for parallel development. We write code for each feature in each branch so that development happens separately. Finally, we merge code off all branches in to Master and push to central repository. We can merge code to any other branch as well. But merging code into master is standard practice that being followed widely. Sometimes, while merging, conflict occurs. When same file is in different branches with different code, when try to merge those branches, conflict occurs. We need to resolve that conflict manually by re-arranging the code.

----

13. What is Git stash?

We create multiple branches to work simultaneously on multiple features. But to work on multiple tasks simultaneously in one branch (i.e. on one feature), we use git stash. Stash is a temporary repository where we can store our content and bring it back whenever we want to continue with our work with that stored content. It removes content inside file from working directory and puts in stashing store and gives clean working directory so that we can start new work freshly. Later on you can bring back that stashed items to working directory and can resume your work on that file. Git stash applicable to modified files. Not new files. Once we finish our work, we can remove all stashed items form stash repository.

----

14. What is Git Reset?

Git Reset command is used to remove changes form staging area. This is bringing back file form staging area to work directory. We use this command before commit. Often we go with git add accidentally. In this case if we commit, that file will be committed. Once you commit, commit ID will be generated and it will be in the knowledge of everyone. So to avoid this one, we use Git reset.

If you add "--hard" flag to git reset command, in one go, file will be removed from staging area as well as working directory. We generally go with this one if we fell that something wrong in the file itself.

----

15. List some importnat Git commands?

git add (to movfile to staging area)
git coomit (to savefile into local repo)
git push (to push file to central repo)

git pull (to pull file to central repo)
git rm (to remove untracked file)
git reset (to remove changes from staging area)
git revert (to revert changes from local repo)

----

16. What is Git Revert?

Git Revert command is used to remove changes from all 3 stages (work directory, staging area and local repository). We use this command after commit. Sometimes, we commit accidentally and later on we realize that we shouldn't have done that. For this we use Git revert. This operation will generate new commit ID with some meaningful message to ignore previous commit where mistake is there. But, here we can’t completely eliminate the commit where mistake is there. Because Git tracks each and every change.

----

17. Difference between Git pull and Git clone?

We use these two command to get changes from central repository. For the first time if you want whole central repository in your local server, we use git clone. It brings entire repository to your local server. Next time onwards you might want only changes instead of whole repository. In this case, we use Git pull.
Git clone is to get whole copy of central repository
Git pull is to get only new changes from central repository (Incremental data)

----

18. What is the difference between Git pull and Fetch?

We use Git pull command to get changes from central repository. In this operation, internally two commands will get executed. One is Git fetch and another one is Git merge.
Git fetch means, only bringing changes from central repo to local repo. But these changes will not be integrated to local repo which is there in your server.
Git merge means, merging changes to your local repository which is there in your server. Then only you can see these changes.
So Git pull is the combination of Git pull and Git merge.
----
19. What is the difference between Git merge and rebase?

We often use these commands to merge code in multiple branches. Both are almost same but few differences. When you run Git merge, one new merge commit will be generated which is having the history of both development branches. It preserves the history of both branches. By seeing this merge commit, everyone will come to know that we merged two branches. If you do Git rebase, commits in new branch will be applied on top of base branch tip. There won't be any merge commit here. It appears that you started working in one single branch form the beginning. This operation will not preserves the history of new branch.
----
20. What is Git Bisect?

Git Bisect we use to pick bad commit out of all good commits. Often developers do some mistakes. For them it is very difficult to pick that commit where mistake is there. They go with building all commits one by one to pick bad commit. But Git bisect made their lives easy. Git bisect divides all commits equally in to two parts (bisecting equally). Now instead of building each commit, they go with building both parts. Where ever bad commit is there, that part build will be failed. We do operation many times till we get bad commit. So Git bisect allows you to find a bad commit out of good commits. You don't have to trace down the bad commit by hand; git-bisect will do that for you.

----

21. What is Git squash?

To move multiple commits into its parent so that you end up with one commit. If you repeat this process multiple times, you can reduce "n" number of commits to a single one. Finally we will end up with only one parent commit. We use this operation just to reduce number of commits.

----

22. What is Git hooks?

We often call this as web hooks as well. By default we get some configuration files when you install git. These files we use to set some permissions and notification purpose. We have different types of hooks (pre commit hooks & post commit hooks)

Pre-commit hooks:- Sometimes you would want every member in your team to follow certain pattern while giving commit message. Then only it should allow them to commit. These type of restrictions we call pre-commit hooks.

Post-commit hooks:- Sometimes, being a manager you would want an email notification regarding every commit occurs in central repository. These kind of things we call post commit hooks.

In simple terms, hooks are nothing but scripts to put some restrictions.

----

23. What is Git cherry-pick?

When you go with git merge, all commits which are there in new development branch will be merged into current branch where you are. But sometimes, requirement will be in such that you would want to get only one commit form development branch instead of merging all commits. In this case we go with git cherry-pick. Git cherry-pick will pick only one commit whatever you select and merges with commits which are there in your current branch. So picking particular commit and merging into your current branch we call git cherry-pick.

----

24. What is the difference between Git and SVN?

SVN:-

It is centralized version control system (CVCS) where back up copy will be placed in only one central repository.
There is no branching strategy in SVN. You can't create branches. So no parallel development.
There is no local repository. So can't save anything locally. Every time after writing code you need to push that code to central repository immediately to save changes.

Git:-
It is Distributed version control system where back up copy is available in every one's machine's local repository as well as central repository.
We can create any no of branches as we want. So we can go with parallel development simultaneously.
Every Git repository will have its own local repository. So we can save changes locally. At the end of our work finally we can push code to central repository.
----
25. What is commit message in Git?

Every time we commit, while committing, we have to give commit message just to identify each commit. We can't remember commit numbers because they contain 40 long alpha numeric characters. So, to remember commits easily, we give commit message. The format of commit message differs from company to company and individual to individual.
We have one more way to identify commits. That is giving "Tags". Tag is a kind of meaningful name to particular commit.

Instead of referring to commit ID, we can refer to tags. Internally tag will refer to respective commit ID.
These are the ways to get particular commit easily.
