GIT

1. What is git?

Git is a distributed version-control system for tracking changes done in source code during software development.

2. What is difference between git and GitHub and GitLab?

**Git** is an open-source tool developers install locally to manage source code,

while **GitHub** is an online web service to which developers who use **Git** can connect and upload or download resources.

**Git repository** is just a file location where we are storing all the files related to your project.

**GitLab** is an open source code repository and collaborative development platform.

**GitLab** offers a location for online code storage and collaborative development of massive software projects.

3. Any other version control tools other than git?

SVN, CVS etc

4. What is difference between svn and git?

**Git** is a distributed version control system, whereas **SVN** is a centralized version control system.

 Git has a centralized server and repository; SVN does not have a centralized server or repository.

 Git branches are easier to work with, than SVN branches.

 Git has better content protection than SVN.  
  
5. What is merge conflicts have you faced ever in you experience? if you face how you resolve?

A **merge conflict happens** when two branches are modified at the same region of a file and are subsequently **merged**.

6. What is git stash?

Sometimes you want to switch the branches, but you are working on an incomplete part of your current project. You don't want to make a commit of half-done work. Git stashing allows you to do so. **The** **git stash command** **enables you to switch branches without committing the current branch.**

7. What is git HEAD?

The **HEAD** points out the last commit in the current checkout branch. It is like a pointer to any reference. The HEAD can be understood as the "**current branch**." When you switch branches with 'checkout,' the HEAD is transferred to the new branch.

8. What is staging area in git?

A staging step in git allows you to continue making changes to the working directory, and when you decide you wanna interact with version control, it allows you to record changes in small commits.

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'.

**9. What is difference between git fetch and pull and clone?**

Git **pull** command pulls a specific branch from the central repository and updates our target branch in the local repository.

Git **fetch**, it pulls all new commits from the desired branch and saves it in a new branch in our local repository. If we need to reflect these changes in our 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**

$ git pull <option> [<repository URL><refspec>...]

In Git, **cloning** is the act of making a copy of any target repository. The target repository can be remote or local. You can clone your repository from the remote repository to create a local copy on your system.

Syntax: $ git clone <repository URL>

10. What is difference between git merge, merge conflict and rebase?

The **merging** is a procedure to connect the forked history. It joins two or more development history together. The git merge command facilitates you to take the data created by git branch and integrate them into a single branch**. Generally,** **git merge is used to combine two branches. It is a non-linear process of merging. Git merge will associate a series of commits into one unified history.**

$ git merge <commit>

When two branches are trying to merge, and both are edited at the same time and in the same file, Git won't be able to identify which version is to take for changes. Such a situation is called merge conflict. If such a situation occurs, it stops just before the merge commit so that you can resolve the conflicts manually.

In Git, the term **rebase** is referred to as the process of moving or combining a sequence of commits to a new base commit. **It is a linear process of merging.**

Suppose you have made three commits in your master branch and three in your other branch. If you merge this, then it will merge all commits at a time. But if you rebase it, then it will be merged in a linear manner.

$git rebase <branch name>

11. What is git ignore file? What is the purpose of the file?

Git ignore files is a file that can be any file or a folder that contains all the files that we want to ignore. The developers ignore files that are not necessary to execute the project. Git itself creates many system-generated ignored files. Usually, these files are hidden files. There are several ways to specify the ignore files. The ignored files can be tracked on a **.gitignore** file that is placed on the root folder of the repository. No explicit command is used to ignore the file.

12. What is git branch? What is master branch? Have you worked on branches?

A **branch** is a version of the repository that diverges from the main working project. When we want to add a new feature or fix a bug, we create a new branch to summarize the changes.

The **master branch** is a default branch in Git. It is instantiated when first commit made on the project. When you make the first commit, you're given a master branch to the starting commit point. When you start making a commit, then master branch pointer automatically moves forward. A repository can have only one master branch.

Operations:

Create ,Delete, List, Rename, Switch and Merge etc..

13. How to merge branches?

14. What is cherry-pick in git?

Cherry-picking in Git stands for applying some commit from one branch into another branch. In case you made a mistake and committed a change into the wrong branch, but do not want to merge the whole branch. You can revert the commit and apply it on another branch.

The main motive of a cherry-pick is to apply the changes introduced by some existing commit. A cherry-pick looks at a previous commit in the repository history and update the changes that were part of that last commit to the current working tree.

15. How to create a git project?

.init

16. How to check difference in between two files?

Git diff

17. How edit the committed message?

git **commit** --amend -m "New **commit message**.

18. How to check the last few commits?

Git log

19. How to delete remote branch and local branch?

Remote branch:

1. $ git push origin -delete <branch name>

Local branch:

1. $ git branch -D <branch name>

20. How to create local branch?

git branch  <branch name>

$ git checkout -b <BRANCH\_NAME> (local)

21. What is tag? How many types in tag? How we will create tags?

**Tags** make a point as a specific point in Git history. Tags are used to mark a commit stage as relevant. We can tag a commit for future reference.

There are two types of tags.

* Annotated tag
* Light-weighted tag

We can create a tag by using the git tag command.

1. $ git tag <tag name>

22. How to roll back committed code. (**Revert**)

In Git, the term revert is used to revert some changes. The git revert command is used to apply revert operation. It is an undo type command. However, it is not a traditional undo alternative. It does not delete any data in this process; instead, it will create a new change with the opposite effect and thereby undo the specified commit.

1. $ git revert -e <commit-id>

23. Git branch strategy

24. What is git clone

In Git, **cloning** is the act of making a copy of any target repository. The target repository can be remote or local. You can clone your repository from the remote repository to create a local copy on your system.

Syntax: $ git clone <repository URL>

25. What is Fork

A **fork** is a rough copy of a repository. Forking a repository allows you to freely test and debug with changes without affecting the original project. One of the excessive use of forking is to propose changes for bug fixing.

26. By what method will you know in Git if a branch has just been combined into master?

To know whether a branch has been merged into master or not you can utilize the below commands:

**git branch - merged** It records the branches that have been merged into the present branch.

**git branch - no merged** It records the branches that have not been merged.

26. Difference between git reset and git revert?

The biggest difference between **git reset** and **git revert** is that

**git reset** will reset the state of the branch to a previous state by dropping all the changes post the desired commit

while **git revert** will reset to a previous state by creating new reverting commits and keep the original commits.

It's recommended to use git revert instead of git reset in enterprise environment.

27. what is squash

**Squash** is technique in which you bundle up some less important commits into a single one.

28. Command to add and commit

git add-commit -m 'My commit message'