1. **What is Git?**
   1. Git is a distributed version control system used for tracking changes in source code during software development. It allows multiple developers to collaborate on projects efficiently by managing revisions, facilitating branching and merging, and providing a history of changes made to the codebase.
2. **What do you understand by the term ‘Version Control System’?**
   1. A version control system (VCS) records all the changes made to a file or set of data, so a specific version may be called later if needed.
3. **What is GitHub?**
   1. To provide Internet hosting for version control and software development, GitHub makes use of Git.
4. **Mention some popular Git hosting services.**
   1. GitHub, SourceForge, GitLab, Bitbucket
5. **Different types of version control systems.**
   1. Local version control systems, Centralized version control systems, Distributed version control systems
6. **What benefits come with using GIT?**
   1. Version control system
   2. data recovery if lost
   3. Git Branching
7. **What’s the difference between Git and GitHub?**
   1. **Git:** Software, Local installation, Git GUI interface
   2. **GitHub:** Service, Hosted on the web, GitHub Desktop interface, Built-in user management
8. **What is a Git repository?**
   1. Git repository refers to a place where all the Git files are stored. These files can either be stored on the local repository or on the remote repository.
9. **How can you initialize a repository in Git?**
   1. If you want to initialize an empty repository to a directory in Git, you need to enter the git init command. After this command, a hidden .git folder will appear.
10. **What language is used in Git?**
    1. Git is a fast and reliable version control system, and the language that makes this possible is ‘C.’
11. **What does git pull origin master do?**
    1. The **git pull origin master** fetches all the changes from the master branch onto the origin and integrates them into the local branch.
12. **What does the git push command do?**
    1. The Git push command is used to push the content in a local repository to a remote repository. After a local repository has been modified, a push is executed to share the modifications with remote team members.
13. **Difference between git fetch and git pull.**
    1. **Git Fetch:** Only downloads new data from a remote repository. Does not integrate any of these new data into your working files. **git fetch origin**
    2. **Git Pull:** Updates the current HEAD branch with the latest changes from the remote server. Downloads new data and integrates it with the current working files. **git pull origin master**
14. **What do you understand about the Git merge conflict?**
    1. A Git merge conflict is an event that occurs when Git is unable to resolve the differences in code between the two commits automatically. Git is capable of automatically merging the changes only if the commits are on different lines or branches.
15. **How do you resolve conflicts in Git?**
    1. Identify the files responsible for the conflicts.
    2. Implement the desired changes to the files.
    3. Add the files using the **git add** command.
    4. Commit the changes in the file with the help of the **git commit** command.
16. **What is the functionality of git ls-tree?**
    1. The **git ls-tree** command is used to list the contents of a tree object.
17. **What is the process to revert a commit that has already been pushed and made public?**
    * 1. Create a new commit to undo all the changes made in the bad commit. Use the following command: **git revert <commit id>**
18. **What does git clone do?**
    1. Git clone allows you to create a local copy of the remote GitHub repository. Once you clone a repo, you can make edits locally in your system rather than directly in the source files of the remote repo.
19. **What is Git stash?**
    1. Git stash takes your modified tracked files and saves them on a stack of unfinished changes that you can reapply at any time. It is used when you want to switch branches to work on something else without making commits in uncompleted work.
20. **What is Git rebase?**
    1. Git rebase is a command used to integrate changes from one branch into another. The primary advantage of rebasing is that it allows you to keep a linear project history.
21. **Explain the git cherry-pick command.**
    1. The **git cherry-pick** command is used to apply the changes introduced by some existing commits. It allows you to pick a specific commit from one branch and apply it onto another branch.
22. **What is the purpose of .gitignore in a Git repository?**
    1. The **.gitignore** file specifies intentionally untracked files that Git should ignore. It is useful to avoid committing unnecessary files and directories, such as temporary files and build artifacts, into the version control system.
23. **How do you check the differences between the working directory and the staging area in Git?**
    1. The **git diff** command can be used to check the differences between the working directory and the staging area in Git.
24. **What is the use of Git bisect?**
    1. Git bisect is a command that uses binary search to help find the commit that introduced a bug. It is a powerful tool for identifying the commit that introduced a regression in the code.
25. **How can you rewrite Git commit history?**
    1. Git commit history can be rewritten using commands like **git rebase** or **git commit --amend**. These commands allow you to modify and organize commits before pushing them to a remote repository.
26. **What is the difference between git rebase and git merge?**
    1. **Git Rebase:** It can lead to a cleaner and more linear history, but it rewrites commit history, which can cause issues if the feature branch has already been shared with others.
    2. **Git Merge:** Creates a new commit that combines the changes from the source branch (the branch you're merging from) into the target branch (the branch you're merging into).
27. **How can you move or rename a file in Git?**
    1. You can use the **git mv** command to move or rename a file in Git. It is equivalent to moving the file manually and then using **git add** on the new file and **git rm** on the old file.
28. **What is a detached HEAD in Git?**
    1. A detached HEAD state in Git occurs when you check out a commit directly, rather than a branch. This means you are no longer on any branch, and any commits you create in this state will be unreachable unless you create a branch to access them.
29. **How does Git store data?**
    1. Git stores data as snapshots of a mini filesystem, where each commit is a full snapshot of the source code at a given point in time. It also uses a system called object storage to keep track of changes and files.
30. **What is the role of git reflog?**
    1. **git reflog** is a reference log that helps you manage and restore local branches that may have been lost. It records changes in the state of your branch references, providing a safety net in case you accidentally delete or modify branches.
31. **How do you stop tracking a file in Git without deleting it?**
    1. To stop tracking a file without deleting it, you can use the **git rm --cached** command. This removes the file from the staging area but preserves it in the working directory.
32. **What is the use of the git clean command?**
    1. The **git clean** command is used to remove untracked files from your working directory. It is helpful when you want to clean up your local workspace by removing files that are not part of the Git repository.
33. **How do you view the commit history in Git?**
    1. The **git log** command is used to view the commit history in Git. It displays a list of commits along with their details, including author, date, and commit message.
34. **What is the purpose of the git tag command?**
    1. The **git tag** command is used to create, list, delete, or verify tags in the Git repository. Tags are references to specific points in Git history, often used to mark release points.
35. **How can you recover a deleted branch in Git?**
    1. To recover a deleted branch in Git, you can use the **git reflog** to find the commit where the branch was deleted and then recreate the branch at that commit using the branch command.
36. **What is the significance of the .gitattributes file?**
    1. The **.gitattributes** file allows you to specify attributes for pathnames in your repository. It is used to control the normalization of line endings, merge strategies for specific files, and other attributes.
37. **What is Git LFS?**
    1. Git LFS (Large File Storage) is an extension for Git that handles large files by replacing them with text pointers in your Git repository while storing the actual file contents on a separate server.
38. **What is a Git submodule?**
    1. A Git submodule is a Git repository embedded inside another Git repository. It allows you to embed one Git repository within another, providing a way to manage dependencies.
39. **How does Git store passwords?**
    1. Git can store passwords using credential helpers, which are external tools that securely store login information and provide it to Git on request. Examples include the Git Credential Manager and cache-based credential storage.
40. **What is the role of Git garbage collection?**
    1. Git garbage collection is the process of cleaning up unnecessary files and optimizing the local repository. It helps in reclaiming space and improving the performance of the repository.
41. **What is the purpose of the Git index?**
    1. The Git index, also known as the staging area, is a crucial component in Git. It is a binary file that stores information about what will go into the next commit. When you perform a git add, you are updating the index.
42. **How can you squash commits in Git?**
    1. To squash commits in Git, you can use an interactive rebase (**git rebase -i**) and mark the commits as "squash" or "fixup." This allows you to combine multiple commits into a single commit.
43. **What is the difference between git revert and git reset?**
    1. **Git Revert:** Introduces a new commit that undoes the changes of a previous commit. Maintains a linear project history.
    2. **Git Reset:** Moves the branch pointer to a specified commit, discarding all commits after that point. It can be used to rewrite history.
44. **Explain shallow cloning in Git.**
    1. Shallow cloning in Git involves only fetching a limited history of a repository, reducing the download size. It can be done using the **--depth** option with the **git clone** command.
45. **Explain what git fsck is used for.**
    1. **git fsck** is used to perform a file system check on the Git repository. It verifies the connectivity and integrity of the objects in the repository and reports any issues or inconsistencies.
46. **What is the purpose of the git pull --rebase command?**
    1. The **git pull --rebase** command fetches changes from a remote repository and rebases your local changes on top of the remote changes. It is an alternative to the regular pull, which merges the changes.
47. **What is the Git upstream branch?**
    1. The upstream branch in Git refers to the default remote branch that your local branch is associated with. It is the branch you set up to track when you create a new branch or clone a repository.
48. **What is the git push --force command used for?**
    1. The **git push --force** command is used to forcefully push local changes to a remote repository, overwriting the remote branch's history with the local branch's history. It should be used with caution.
49. **How do you revert a Git repository to a previous commit?**
    1. To revert a Git repository to a previous commit, you can use the **git reset** or **git revert** command. The choice depends on whether you want to preserve the commit history or create a new commit that undoes the changes.
50. **What is the purpose of the git describe command?**
    1. The **git describe** command provides a human-readable output describing the given commit relative to the closest tagged commit. It is useful for obtaining a meaningful version string.
51. **How do you create an annotated tag in Git?**
    1. To create an annotated tag in Git, you can use the **git tag -a <tag-name>** command, which opens the default text editor for you to add a tag message. Annotated tags store extra information such as the tagger's name, email, and date.
52. **What is the purpose of the .gitkeep file?**
    1. The **.gitkeep** file is used to include an otherwise empty directory in the version control system. Git ignores empty directories, so adding a **.gitkeep** file helps in keeping the directory tracked.
53. **How do you unstage changes in Git?**
    1. To unstage changes in Git, you can use the **git reset HEAD <file>** command. This removes the changes from the staging area while keeping them in the working directory.
54. **How do you amend the last commit in Git?**
    1. To amend the last commit in Git, you can use the **git commit --amend** command. This opens the default text editor, allowing you to modify the commit message or add additional changes.