1. What is Git?

[Git](https://www.simplilearn.com/tutorials/git-tutorial/what-is-git) is a version control system for tracking changes in computer files and is used to help coordinate work among several people on a project while tracking progress over time. In other words, it’s a tool that facilitates source code management in software development.

Git favors both programmers and non-technical users by keeping track of their project files. It enables multiple users to work together and handles large projects efficiently.



2. What do you understand by the term ‘Version Control System’?

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.

This helps ensure that all team members are working on the latest version of the file



3. What is GitHub?

   To provide Internet hosting for version control and software development, GitHub makes use of Git.

4. Mention some popular Git hosting services.

* GitHub
* SourceForge
* GitLab
* Bitbucket

5. Different types of version control systems

* Local version control systems have a database that stores all file changes under revision control on disc in a special format.
* Centralized version control systems have a single repository, from which each user receives their working copy.
* Distributed version control systems contain multiple repositories, and different users can access each one with their working copy.

6. What benefits come with using GIT?

* Data replication and redundancy are both possible.
* It is a service with high availability.
* There can only be one Git directory per repository.
* Excellent network and disc performance are achieved.
* On any project, collaboration is very simple.

7. What’s the difference between [Git and GitHub](https://www.simplilearn.com/tutorials/git-tutorial/git-vs-github)?

|  |  |
| --- | --- |
| Git | GitHub |
| Git is a software | GitHub is a service |
| [Git can be installed](https://www.simplilearn.com/tutorials/git-tutorial/git-installation-on-windows) locally on the system | GitHub is hosted on the web |
| Provides a desktop interface called git GUI | Provides a desktop interface called GitHub Desktop. |
| It does not support user management features | Provides built-in user management |

8. What is a Git repository?

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?

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. How is Git different from Subversion (SVN)?

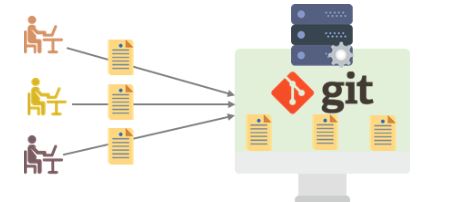
|  |  |
| --- | --- |
| GIT | SVN |
| Git is a distributed decentralized version control system | SVN is a centralized version control system. |
| Git stores content in the form of metadata. | SVN stored data in the form of files. |
| The master contains the latest stable release. | In SVN, the trunk directory has the latest stable release |
| The contents of Git are hashed using the SHA-1 hash algorithm. | SVN doesn’t support hashed contents. |

11. Name a few Git commands with their function.

* Git config - Configure the username and email address
* Git add - Add one or more files to the staging area
* Git diff - View the changes made to the file
* Git init - Initialize an empty Git repository
* Git commit - Commit changes to head but not to the remote repository

12. What are the advantages of using Git?

* Faster release cycles
* Easy team collaboration
* Widespread acceptance
* Maintains the integrity of source code
* [Pull requests](https://www.simplilearn.com/tutorials/git-tutorial/git-pull-request)



13. What language is used in Git?

Git is a fast and reliable version control system, and the language that makes this possible is ‘C.’

Using [C language](https://www.simplilearn.com/c-programming-article) reduces the overhead of run times, which are common in high-level languages.

### 14. What is the correct syntax to add a message to a commit?

 git commit -m "x files created"

### 15. Which command is used to create an empty Git repository?

git init - This [command](https://www.simplilearn.com/tutorials/git-tutorial/git-commands) helps to create an empty repository while working on a project.

### 16. What does git pull origin master do?

The git pull origin master fetches all the changes from the master branch onto the origin and integrates them into the local branch.

git pull = git fetch + git merge origin/ master

After having gone through the beginner level Git interview questions, let us now look at intermediate GIT interview questions and answers.

## **Intermediate Git Interview Questions**

### 17.  What does the git push command do?

The [Git push command](https://www.simplilearn.com/tutorials/git-tutorial/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.



### 18. Difference between git fetch and git pull.

|  |  |
| --- | --- |
| Git Fetch | Git Pull |
| The Git fetch command only downloads new data from a remote repository. | Git pull updates the current HEAD branch with the latest changes from the remote server. |
| It does not integrate any of these new data into your working files. | Downloads new data and integrate it with the current working files. |
| Command - git fetch origin  git fetch --all | Tries to merge remote changes with your local ones.  Command - git pull origin master |

### 19. GitHub, GitLab and Bitbucket are examples of git repository \_\_\_\_\_\_\_ function?

hosting. All the three are services for hosting Git repositories

### 20. What do you understand about the Git merge conflict?

A [Git merge conflict](https://www.simplilearn.com/tutorials/git-tutorial/merge-conflicts-in-git) 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.



21. How do you resolve conflicts in Git?

Here are the steps that will help you resolve conflicts in Git:

* Identify the files responsible for the conflicts.
* Implement the desired changes to the files
* Add the files using the git add command.
* The last step is to commit the changes in the file with the help of the git commit command.

22. What is the functionality of git ls-tree?

The git ls-tree command is used to list the contents of a tree object.

Or

This command returns a tree object representation of the current repository along with the mode and the name of each item and the SHA-1 value of the blob.

23. What is the process to revert a commit that has already been pushed and made public?

There are two processes through which you can revert a commit:

1. Remove or fix the bad file in a new commit and push it to the remote repository. Then commit it to the remote repository using:

git commit –m “commit message”

2. Create a new commit to undo all the changes that were made in the bad commit. Use the following command:

git revert <commit id>

24. How is a bare repository different from the standard way of initializing a Git repository?

|  |  |
| --- | --- |
| Standard way | Bare way |
| You create a working directory with the git init command. | Does not contain any working or checked out copy of source files. |
| A .git subfolder is created with all the git-related change history. | Bare repositories store git revision history in the root folder of your repository instead of the .git subfolder. |

25. What does git clone do?

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

26. What is Git stash?

Let’s say you're a developer and you want to switch branches to work on something else. The issue is you don’t want to make commits in uncompleted work, so you just want to get back to this point later. The solution here is the Git stash.

Git stash takes your modified tracked files and saves it on a stack of unfinished changes that you can reapply at any time. To go back to the work you can use the stash pop.

27. What does the git reset --mixed and git merge --abort commands do?

git reset --mixed is used to undo changes made in the working directory and staging area.

git merge --abort helps stop the merge process and return back to the state before the merging began.

28. What do you understand about the Staging area in Git?

The Staging Area in Git is when it starts to track and save the changes that occur in files. These saved changes reflect in the .git directory. Staging is an intermediate area that helps to format and review commits before their completion.

29. What is Git Bisect and how do you use it?

The Git Bisect command performs a binary search to detect the commit which introduced a bug or regression in the project’s history.

Syntax: git bisect <subcommand> <options>

30. How do you find a list of files that has been changed in a particular commit?

The command to get a list of files that has been changed in a particular commit is:

git diff-tree –r {commit hash}

* -r flag allows the command to list individual files
* commit hash lists all the files that were changed or added in the commit.

31. What is the use of the git config command?

The git config command is used to set git configuration values on a global or local level. It alters the configuration options in your git installation. It is generally used to set your Git email, editor, and any aliases you want to use with the git command.

32. What is the functionality of git clean command?

The git clean command removes the untracked files from the working directory.

33. What is SubGit and why is it used?

SubGit is a tool that is used to migrate SVN to Git. It transforms the SVN repositories to Git and allows you to work on both systems concurrently. It auto-syncs the SVN with Git.

34. If you recover a deleted branch, what work is restored?

The files that were stashed and saved in the stashed index can be recovered. The files that were untracked will be lost. Hence, it's always a good idea to stage and commit your work or stash them.

35. Explain these commands one by one– git status, git log, git diff, git revert <commit>,  git reset <file>.

* Git status - It shows the current status of the working directory and the staging area.
* Git revert<commit> -  It is used for undoing changes to a repository's commit history.
* Git log- It is a key tool for reviewing and reading the history of everything that happens to a repository.
* Git diff- It is a multi-purpose Git command that performs a diff function on Git data sources when executed.
* Git reset<file>- it is used to unstage a file.

36. What exactly is tagging in Git?

Tagging enables developers to mark all significant checkpoints as their projects progress.

37. What exactly is forking in Git?

It is a repository duplicate and forking allows one to experiment with changes without being concerned about the original project.

38. How to change any older commit messages?

You can change the most recent commit message with the git commit —amend command.

39. How to handle huge binary files in Git?

Git LFS is a Git extension for dealing with large and binary files in a separate Git repository.

40. Name a few GIT tools.

Git comes with a few built-in tools like Git Bash and Git GUI.

41. Will you make a new commit or amend an existing one?

The git commit —amend command allows you to easily modify the most recent commit.

42. What do you mean by branching strategy?

It is employed by a software development team while writing and managing code with a version control system.

43. Difference between head, working tree, and index.

They are all names for various branches. Even Though a single git repository can track an arbitrary number of branches, the working tree is only associated with one of them, and HEAD points to that branch.

44. Is there a git GUI client available for Linux?

Git includes built-in GUI tools for committing (git-gui) and browsing (gitk), but there are a number of third-party tools available for users seeking platform-specific experience.

45. What is the benefit of a version control system?

Version control enables software teams to maintain efficiency and agility while the team grows by adding more developers

### 46. What do you mean by git instaweb?

 It is a script used to set up a temporary instance of Gitweb.

### 47. What exactly is the forking workflow?

Forking is a git clone operation that is performed on a server copy of a project's repository.

### 48. Mention benefits of forking workflow.

Contributions can be integrated without everyone trying to push to a single central repository.

### 49. What is the Gitflow workflow?

The Gitflow Workflow specifies a strict branching model centered on the project release.

### 50. What does the commit object contain?

The commit object contains a tree of blob objects and other tree objects that represent the project revision's directory structure.

### 51. Write the syntax of rebasing in git.

Syntax is as follows: $git rebase <branch name>

### 52. What are Git Hooks?

They are scripts that are executed automatically whenever a specific event occurs in a Git repository.

### 53. What is Git stash vs Git stash pop?

Git stash pop removes the (topmost, by default) stash when applied, whereas git stash apply keeps it in the stash list for future use.

### 54. Explain git reflog This command is used by Git to record changes made to the branches' tips.

### 55. Role of the git annotate command.

In git, it is used to track each line of the file based on the commit information.

### 56. What is a git Directory?

It is the storage place of the metadata and object database of the project.

### 57. How can a conflict be settled in Git?

Edit the files to resolve any incompatible changes first, then use "git add" to add the corrected files and "git commit" to save the repaired merge.

### 58. What is the standard method for branching in GIT?

In GIT, the best way to create a branch is to have one'main' branch and then another branch for implementing the changes that we want to make.

### 59. How do you set up a Git repository?

If you want to add an empty repository to a directory in Git, use the git init command.

### 60. What is the proper syntax for appending a message to a commit?

Git commit -m "x files created" is the syntax.

### 61. Use of git instaweb.

It is used to launch a web browser and a webserver with an interface into a local repository automatically.

### 62. Describe git is-tree.

It represents a tree object with each item's mode and name included.

### 63. What exactly is git cherry-pick?

A command typically used to move specific commits from one branch of a repository to another.

### 64. State the difference between “git remote” and “got clone”?

“Git remote” allows you to create an entry in the git configuration which specify a URL.

“Git clone” lets you create a new git repository by letting you copy it from the  current URL.

### 65. Difference between “pull request” and “branch”?

“Pull request” is done when you feel like changing the developer’s change to another person's code branch. And “Branch” is just a separate version of code.

### 66. How might you recover a branch that has previously pushed changes in the main repository yet has been coincidentally erased from each team member's local machines?

We can easily recover this by seeing the latest commit of the branch in the reflog and then going through the new branch.

### 67. What is a detached head?

Detach head refers to that the currently checked repository is not in the local branch.

### 68. What command helps us to know the branches merged into master and which are not?

git branch  - -merged lets us get the lost of the branches which are currently merged into the current branch

git branch - - no- merged shows the branches which are not merged

### 69. Is LDAP Authentication Supported?

GitLab API only supports LDAP authentication since version 6.0 and higher.

## **Popular Git Interview Questions**

### 70. A simple definition of Git

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

### 71. Is the C++ language used in Git?

Yes, but C programming language is a widely used language in Git.

### 72. How does Git work?

Git works by tracking changes to files in a project and allowing developers to easily revert to previous versions if necessary. Git also makes it easy to collaborate on projects, as it allows multiple developers to work on the same codebase simultaneously.

### 73. What are some of the most popular Git commands?

Some of the most popular Git commands are,

* "git init" (which initializes a Git repository)
* "git add" (which adds files to a Git repository)
* And "git commit" (which saves changes to a Git repository).

### 74. List out the functions provided by Git repository hosting service

There are many functions that a Git repository hosting service can provide. Some of the most common and useful functions include:

* Providing a web interface for users to interact with the repositories
* Allowing users to clone repositories
* Allowing users to view and download statistics about the repositories
* Providing a way for users to push changes to the repositories
* Keeping track of changes made to the repositories

### 75. What is the difference between Git and other revision control systems?

* Git is a distributed revision control system, which means that it can be used without a central server. This allows for a great deal of flexibility in how projects are managed.
* On the other side, revision control systems are often centralized, which can limit the flexibility of how projects are managed.

### 76. How does Git work?

Git works by tracking changes to files in a repository. When a file is changed, Git calculates a unique identifier for the change, called a "commit hash". The commit hash allows Git to identify the change and track it over time.

### 77. How do I install Git?

Installing Git is simple. Just download the latest version from the Git website (<https://git-scm.com/>).

### 78. How do I use Git?

To use Git, a developer first creates a local repository on their computer. This repository contains all the files for a project and the history of all the changes made to those files.

Or just follow the instructions in the Git documentation (<https://git-scm.com/doc>).

### 79. What are some of the drawbacks of Git?

* One of the main drawbacks is that it can be difficult to learn and use, especially for those who are not familiar with version control systems.
* And Git is not always reliable and can sometimes be slow.

### 80. What are some of the most important commands in Git?

Some of the most important Git commands are "commit", "push", and "pull".

* The "commit" command is used to save changes to the local repository.
* And the "push" command is used to send changes to the remote repository.
* Then the "pull" command is used to retrieve changes from the remote repository.

### 81. What are some of the most important features of Git?

Some of the most important features of Git are its distributed nature, its ability to track changes, and its support for branches.

* The distributed nature of Git allows developers to work independently and offline.
* The ability to track changes helps developers to keep track of their work and revert to previous versions if necessary.
* The support for branches allows developers to experiment with new features without affecting the main codebase.

### 82. What is a branch in Git?

A branch is a way to isolate development work on a particular aspect of a project. When a branch is created, it diverges from the primary branch. It allows developers to work on a new feature or bug fix without affecting the main codebase.

### 83. What is a commit in Git?

A commit is a way to save changes to a branch. When a commit is made, a snapshot of the current state of the branch is created. This snapshot can be used to revert the branch to that state if necessary.

### 84. What is conflict in Git?

Conflict in Git occurs when two or more developers have made changes to the same part of a file, and those changes can't be automatically merged. When this happens, Git will mark the file as conflicted and leave it up to the developers to resolve the conflict.

Resolving a conflict can be done by manually editing the file to choose which changes should be kept, or by using a tool like Git's merge command to automatically merge the changes.

### 85. What does the git status command do?

The git status command is used to obtain the current state of a Git repository. This command can be used to determine whether the repository is clean or dirty, and to see which files have been modified. The git status command will also show which branch is currently checked out and whether there are any uncommitted changes.

### 86. Why is it considered to be easy to work on Git?

There are many reasons that Git is considered an easy tool to work with.

* It has a straightforward learning curve. Even those new to programming can easily learn how to use Git with just a few hours of practice.
* Git is highly flexible and can be easily customized to fit the needs of any project.
* And, Git is very stable and reliable, so users can trust that their work will be safe and sound.

### 87. What do you know about Git Stash?

Git stash is a powerful tool that allows you to save your changes and revert your working directory to a previous state. This is especially useful when switching branches or reverting to a previous commit.

And Git Stash takes a snapshot of your changes and stores them away for later use.

### 88. What differentiates between the commands git remote and git clone?

The main difference between the git remote and git clone commands is that the git remote adds a remote repository as a shortcut to your current repository, while the git clone creates an entirely new copy of a remote repository.

### 89. Tell me the difference between git pull and git fetch?

Both of these commands will fetch any new commits from the remote repository, but they differ in how they handle these commits.

Git pull will merge the remote commits into the current branch, while git fetch will simply retrieve the commits and store them in the local repository. This means that if you have any uncommitted changes, git pull may result in merge conflicts, while git fetch will not.

### 90. Is Git and GitHub the same thing?

No, Git and GitHub are two different things.

* Git is a version control system that lets you track changes to your code.
* GitHub is a hosting service for Git repositories. You can use GitHub to store your code remotely, or you can use it to collaborate with other developers on a project.

### 91. What about Git reflog?

Git reflog is a history of all the changes made to a git repository. It is a valuable tool for debugging and troubleshooting purposes.

And Git reflog can be used to view the history of a repository, see who made what changes, and when those changes were made.

### 92. What is a detached head?

A detached HEAD is a state where the HEAD pointer is not pointing to the current commit. This can happen if you check out a commit that is not the most recent, or if you reset your head to a previous commit.

### 93. How to avoid a detached head?

There are a few different ways to avoid a detached HEAD.

* The first is to simply commit your changes before switching branches. This will ensure that your changes are saved to a specific branch, and you won't have to worry about them being lost when you switch branches.
* Another way to avoid detached HEAD is to use the "git checkout" command with the "-b" option.

### 94. How will you resolve conflict in Git?

To resolve a conflict in Git, you will need to first identify the source of the conflict. Once you have identified the source of the conflict, you can use the "git pull" command. This will pull the latest changes from the remote repository and merge them with your local copy.

If the "git pull" command doesn't resolve the conflict, you can try the "git merge" command. This will merge the two versions of the code manually. You will need to resolve the conflicts manually and then commit the merged code.

### 95. What is Subgit and where do you use Subgit?

Subgit is a tool for managing Git repositories with Subversion history. It allows you to keep your existing subversion history while moving to Git, and it also provides a way to keep your Git history synchronized with subversion.

There are many reasons that you might want to use Subgit,

* For example, if you have a large subversion repository with a lot of history, moving to Git can be a huge undertaking; that's where Subgit can help transition smoother.
* And if you work in an environment where subversion is the primary version control system, using Subgit can help you keep your Git history in sync with the rest of the team.

## **Advanced Git Interview Questions**

### 96. Explain the different points when a merge can enter a conflicted stage.

There are two stages when a merge can enter a conflicted stage.

1. Starting the merge process

If there are changes in the working directory of the stage area in the current project, the merge will fail to start. In this case, conflicts happen due to pending changes that need to be stabilized using different Git commands.

2. During the merge process

The failure during the merge process indicates that there’s a conflict between the local branch and the branch being merged. In this case, Git resolves as much as possible, but some things have to be fixed manually in the conflicted files.

### 97. What has to be run to squash the last N commits into a single commit?

In Git, squashing commits means combining two or more commits into one.

Use the below command to write a new commit message from the beginning.

git reset -soft HEAD~N &&git commit

But, if you want to edit a new commit message and add the existing commit messages, then you must extract the messages and pass them to Git commit.

The below command will help you achieve this:

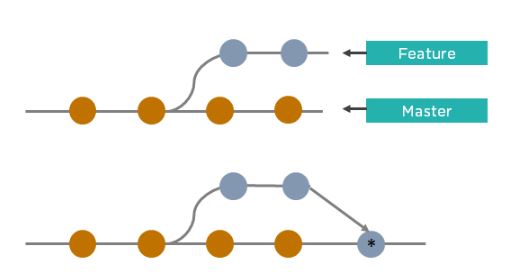
git reset -soft HEAD~N &&git commit -edit -m“$(git log -format=%B -reverse .HEAD@{N})”

### 98. What is the difference between fork, branch, and clone?

|  |  |  |
| --- | --- | --- |
| Fork | Branch | Clone |
| The fork is the process when a copy of the repository is made. It's usually experimentation in the project without affecting the original project. They’re used to advise changes or take inspiration from someone else’s project. | Git branches refer to individual projects within a git repository. If there are several branches in a repository, then each branch can have entirely different files and folders. | Git clone refers to creating a clone or a copy of an existing git repository in a new directory. Cloning automatically creates a connection that points back to the original repository, which makes it very easy to interact with the central repository. |

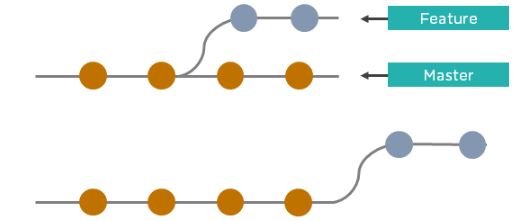
### 99. How is Git merge different from Git rebase?

Git merge is used to incorporate new commits into your feature branch.



* Git merge creates an extra merge commit every time you need to incorporate changes.
* It pollutes your feature branch history.

As an alternative to merging, you can rebase the feature branch into master.



* Git rebase Incorporates all the new commits in the master branch.
* It rewrites the project history by creating brand new commits for each commit in the original branch

### 100. What is the command used to fix a broken commit?

To fix a broken commit in Git, you may use the “git commit --amend” command, which helps you combine the staged changes with the previous commits instead of creating an entirely new commit.

### 101. How do you recover a deleted branch that was not merged?

To recover a deleted branch, first, you can use the git reflog command. It will list the local recorded logs for all the references. Then, you can identify the history stamp and recover it using the git checkout command.

102. What is git stash drop?

The Git stash drop command is used to remove a particular stash. If there’s a stash you're no longer using or you want to remove a specific item of stash from the list, you can use the stash commands.

Let’s say you want to delete an item named stash@{abc}; you can use the command:

git stash drop stash@{abc}.

103. What’s the difference between reverting and resetting?

|  |  |
| --- | --- |
| Reverting | Resetting |
| The revert command in Git is used to create a new commit that undoes the changes made in the previous commit. When you use this command, a new history is added to the project; the existing history is not modified. | Git reset is a command that is used to undo the local changes that have been made to a Git repository. Git reset operates on the following: commit history, the staging index, and the working directory. |

104. How can you discover if a branch has already been merged or not?

There are two commands to determine these two different things.

git branch --merged - Returns the list of branches that have been merged into the current branch.

git branch --no-merged - Returns the list of branches that have not been merged.

105. What is “git cherry-pick”?

The command git cherry-pick enables you to pick up commits from a branch within a repository and apply it to another branch. This command is useful to undo changes when any commit is accidentally made to the wrong branch. Then, you can switch to the correct branch and use this command to cherry-pick the commit.

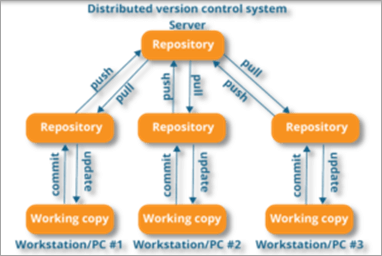
**Q #1) What is Git?**

**Answer:** Git is a Distributed Version Control tool. It is compatible with distributed non-linear workflows as it offers data assurance for building good quality software.

Git is free and open-source. It can be used for almost any kind of project, be it small or big in size. Git is known for its great speed and efficiency. Git repositories are very easy to find and access. Due to its certain features, Git is highly flexible, secure and compatible with your system.

**Q #2) What is a distributed Version Control System?**

**Answer:** A distributed VCS is a system that does not depend upon a central server to keep a project file and all its versions. In distributed VCS, each collaborator or developer gets a local copy of the main repository and this is called a clone.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/distributed-version-control-system.png)

*[image*[*source*](https://www.edureka.co/)*]*

As you can see in the above diagram, every collaborator maintains a local repository on their local machines. They can commit and update the local repositories without any issues.

Using a pull operation, a developer can update his local repository with the latest changes from the central server. Using the push operation, they can send their changes from the local repository to the central server.

**Q #3) Who created Git?**

**Answer:** Git was created by Linus Torvalds in 2005 on the road to develop Linux Kernel.

**Q #4) Which language is used in Git?**

**Answer:** C is the underlying programming language in which Git is written. C language makes Git fast by evading runtime overheads linked with other high-level programming languages.

**Q #5) What are the advantages/main features of Git?**

**Answer: Enlisted below are the various features of Git.**

**(i) Free & Open Source:**

Git is issued under GPL’s (General Public License) open source license. You need not pay anything to use Git.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Free-Open-Source.png)

It is absolutely free. As it is open-source, you can modify the source code according to your needs.

**(ii) Speed:**

As you are not required to connect to any network for executing all the actions, it performs all the tasks quickly. Obtaining version history from a locally stored repository can be one hundred times speedier than obtaining it from the remote server.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Speed.png)

Git is written in C, which is the underlying programming language that evades runtime overheads linked with other high-level languages.

**(iii) Scalable:**

Git is highly scalable. So, if the number of collaborators increases in the coming time, then Git can easily accommodate this change.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Scalable.png)

Despite the fact that Git represents an entire repository, the data kept on the client’s side is very small as Git compacts the entire vast data through a lossless compression technique.

**(iv) Reliable:**

As every collaborator has its own local repository, on the instances of a system crash, the lost data can be recuperated from any of the local repositories. At all times, you will have a backup of all your files.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Reliable-1.png)

**(v) Secure:**

Git utilizes the SHA1 (Secure Hash Function) to name and identify objects inside its repository. Each artifact and commit are check-summed and recovered through its checksum during checkout.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Secure.png)

The Git history is saved in a manner in which the ID of a specific version (a commit in terms of Git) relies on the total development history running up to that commit. Once a file version is pushed to Git, then there is no way to change it without being noticed.

**(vi) Economical:**

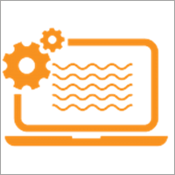
In the case of a centralized version control system, the central server must be strong enough to attend requests of the entire team. This is not a problem for smaller teams, however as the team expands, the hardware limitations of the server can be an impediment for performance.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Economical.png)

In the case of distributed version control systems like Git, the team members don’t require interaction with the server expect when they are required to push or pull changes. All the heavy lifting occurs at the client end, thus the server hardware can be kept quite simple certainly.

**(vii) Supports Non-linear Development:**

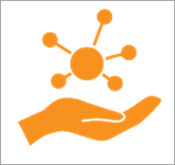
Git provides rapid branching & merging and contains particular tools for envisaging and traversing a non-linear development history. A basic notion in Git is that a change will be merged more frequently than it is written as it is sent across different reviewers.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Supports-non-linear-Development.png)

Git Branches are extremely lightweight. A branch in Git refers only to a single commit. The complete branch structure can be created, with the help of parent commits.

**(viii) Easy Branching:**

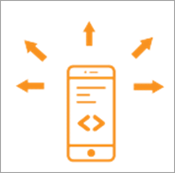
Branch management through Git is very straightforward and easy. It requires just a few jiffies to create, delete, and merge branches. Feature branches give an insulated environment to each change to your codebase.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Easy-Branching.png)

When a developer requires to begin working on something, irrespective of the size of work, they create a new branch. This makes sure that the master branch constantly holds a production-quality code.

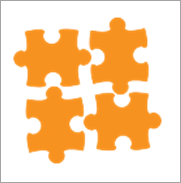
**(ix) Distributed Development:**

Git provides every developer a local copy of the whole development history, plus the changes get cloned from one such repository to another. These changes are introduced as added development branches and can be merged in the same manner as a locally developed branch.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Distributed-development.png)

**(x) Compatibility along with present Systems or Protocol:**

Repositories can be published through HTTP, FTP or a Git protocol on top of either a plain socket or ssh.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Compatibility-along-with-present-systems-or-protocol.png)

**Q #6) How do you create a Repository in Git?**

**Answer:** To create a repository, you need to create a directory for the project if it does not already exist, and then simply execute the command “**git init**”. By executing this command, a .git directory will be created inside the project directory i.e. now your project directory has turned into a Git repository.

**Q #7) What is a .git Directory?**

**Answer:** The moment you create a repository, you will find a .git directory present inside it. This .git directory contains all the metadata of the repository and maintains a track of all the changes made to the files in your repository, by keeping a commit history.

All the information regarding commits, hooks, refs, object databases, remote repository addresses, etc. are kept inside this folder. This is the most crucial part of Git. When you clone any Git repository on your local machine, this .git is the directory that actually gets copied.

**Q #8) What happens if the .git directory gets deleted?**

**Answer:** If the .git/ directory gets deleted, then you will lose track of your project’s history. The repository will no longer be under version control.

**Q #9) Which command is used for writing a Commit Message in Git?**

**Answer:** The command used for passing on a message to a git commit is *git commit -m “commit message”.*The flag *m*is used to pass a commit message.

**Q #10) What is the bare Git repository? How is it different from a standard/non-bare Git repository?**

**Answer:** Repositories that are created through *git init*command are the standard/non-bare Git repositories.

In the top-level folder of such repository, you will find two things:

1. A .git subdirectory keeping all metadata and track of the history of your repo.
2. A working tree.

The repositories which are created using *git init –bare*command are known as bare Git repositories. They are mainly used for sharing. They do not contain any working tree. They keep the git revision history of your repository in the root folder rather than having it inside the .git subfolder.

It just contains bare repository data. This is how a bare Git repository is different from a standard Git repository. Also, a bare repository does not have a default remote *origin*repository as it serves as an origin repository for multiple remote users.

Since a bare repository does not contain any workspace, the *git push*and *git pull* commands do not work over a bare repo. You are not required to commit any changes to a bare repo.

**Q #11) Mention some Git Repository Hosting Services.**

**Answer:**

* Github
* Pikacode
* Gitlab
* Microsoft VSTS
* BitBucket
* GitEnterprise
* SourceForge
* LaunchPad
* Perforce
* Beanstalk
* Assembla

**Q #12) Name some Basic Operations in Git.**

**Answer: Some basic operation in Git include:**

* Initialize
* Add
* Commit
* Push
* Pull

**Q #13) Name some Advanced Operations in Git.**

**Answer: Some advanced operations in Git are:**

* Branching
* Merging
* Rebasing

**Q #14) How will you distinguish between Git and SVN?**

**Answer:**Git is a distributed version control whereas SVN is centralized. This leads to many differences between the two in terms of their features and functionalities.

|  | **Git** | **SVN** |
| --- | --- | --- |
| **Server Architecture** | The computer on which your Git has installed acts as both client and server. Each developer has a local copy of the complete version history of the project on their individual computers. Git changes occur locally.  Hence, the developer is not required to be connected to the network at all times. Only for push and pull operations, developers would need internet connection to connect to remote server. | SVN has a separate client and server. It is not locally available. You will be required to be connected to the network to perform any action.  Also, in SVN, since everything is centralized, so in case the central server gets crashed or corrupted, it will result in entire data loss for the project. |
| **Branching** | Git is mostly preferred by developers due to its effective branching model. Git branches are lightweight but powerful.  They are only references to a particular commit. You can create, delete or modify a branch anytime with no impact on other commits. So, fork, branch and merge are easy with Git. | SVN has a complicated branching and merging model and its time-consuming to manage.  In SVN, branches are generated as directories within the repository. This directory structure is mainly problematic. When the branch is ready, you need to commit back to the trunk. Since you are not the only one who is merging the changes, so the version of the truck may not be regarded as developers’ branches. This can lead to conflicts, missing files and jumbled changes in your branch. |
| **Access Control** | Git presumes that all the contributors will be having the same permissions. | SVN permits you to define read/write access controls at each and directory level. |
| **Auditability** | In Git, the changes are tracked at the repository level. Git does not bother too much about maintaining the precise history of changes made in your repository. The distributed nature of Git lets any collaborator change any part of their local repo’s history. With Git, it’s difficult to figure a true history of changes in your codebase.  For example, you will lose history after rename in Git. | In SVN, the changes are tracked at the file level.  SVN maintains a pretty consistent and precise change history. You can recover exactly the same data as it was at any instant in the past.  SVN history is permanent and always definite. |
| **Storage Requirements** | Git and SVN store the data in the same manner. The disk space usage is equal for both of them. The only difference comes into picture in case of binary files. Git is not friendly to binary files.  It can’t handle the storage of large binary files. | SVN has an xDelta compression algorithm that works for both binary and text files.  So, SVN can handle storing large binary files in comparatively lesser space than Git. |
| **Usability** | Both Git and SVN use command line as primary UI.  Git is largely used by developers/technical users. | SVN is largely used by non-technical users as it's easier to learn. |
| **Content** | Cryptographic SHA-1 Hash. | No hashed content. |
| **Global Revision Number** | Not available | Available |

**Q #15) How will you differentiate between Git and GitHub?**

**Answer:**Git is a high-quality version control system. It is distributed in nature and is employed to track changes in source code throughout software development. It has a unique branching model that helps in synchronizing work among developers and tracking changes in any files.

The primary goals of Git are speed, data integrity, providing support to distributed, non-linear workflows. Git is installed and maintained on the local machine, instead of the cloud.

GitHub is a cloud-based Git repository hosting service that brings teams together. It gives you a web-based GUI as well as provides access control and many collaboration features, fundamental task management tools for each project.

Also, GitHub is an open-source i.e. code is kept on a centralized server and can be accessed by everyone.

**Q #16) What is a conflict in Git and how to resolve it?**

**Answer:**Git has an automatic merging feature that handles the merge commits on its own, provided the code changes have occurred on different lines and in different files.

But, in case of competing for commits where there are changes in the same lines of code of a file or a file has been deleted in one branch but exists and modified in another, Git is unable to automatically resolve differences and thus raises merge conflict.

In such cases, it requires your help to decide which code to include and which code to discard in the final merge.

A merge conflict can occur during merging a branch, rebasing a branch, or cherry-picking a commit. Once a conflict is detected, Git highlights the conflicted area and asks you to resolve it. Once the conflict is resolved, you can proceed with the merge.

**Follow the below steps to resolve a competing line change merge conflict:**

1. Open Git Bash (Git command line).
2. Use ***cd <repository-name>*** command to go to the local Git repository which is having the merge conflict.
3. Use the ***git status*** command to produce the list of files affected by the merge conflict.
4. Open the text editor that you use and traverse to the file that has merge conflicts.
5. To see the start of the merge conflict in your file, look the document for the conflict marker <<<<<<<. At the point when you open the file, you’ll observe the modifications from the HEAD or base branch after the line <<<<<<< HEAD. Then, you’ll observe =======, which partitions your modifications are from the modifications in the other branch, trailed by >>>>>>> BRANCH-NAME.
6. Choose in the event that you need to keep just your branch’s changes, just keep the other branch’s changes, or make a fresh change, that may include changes from the two branches. Erase the conflict markers <<<<<<<, =======, >>>>>>> and do the changes that you need in the final merge.
7. Use ***git adds.*** command to add or stage your changes.
8. Finally, use the ***git commit -m “message”*** command to commit your changes with a comment.

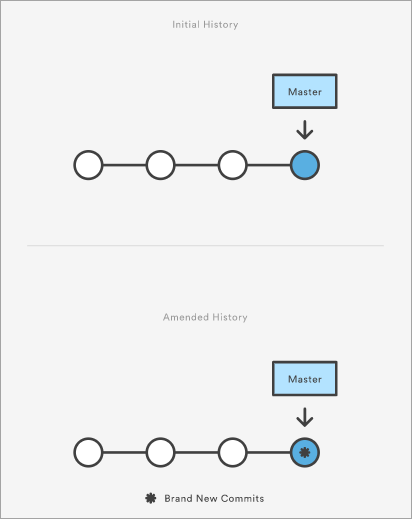
**To resolve the removed file merge conflict, you need to follow the below steps:**

1. Open Git Bash (Git command line).
2. Use ***cd <repository-name>*** command to go to the local Git repository that has the merge conflict.
3. Use the ***git status*** command to produce the list of files affected by the merge conflict.
4. Open the text editor that you use and traverse to the file that has merge conflicts.
5. Choose if you wish to keep the removed file. You can check the latest changes done in the removed file in your text editor.
6. Use ***git add <filename>*** command to add the removed file back to the repository. Or, Use ***git rm <filename>*** command to remove the file from your repository.
7. Finally, use the ***git commit -m “message”*** command to commit your changes with a comment.

**Q #17) How will you fix a Broken Commit?**

**Answer:** To fix a broken commit or to change the last commit, the most convenient method is to use the command “*git commit -amend’*.

It allows you to combine staged changes with the previous commit as an alternative for creating an entirely new commit. This replaces the most recent commit with the amended commit.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/broken-commit.png)

*[image*[*source*](https://www.atlassian.com/)*]*

Through this command, you can also edit the previous commit message without changing its snapshot.

**Q #18) What is the use of git instaweb?**

**Answer:** It is a script through which you can instantly browse your working Git repository in a web browser.

This script sets up gitweb and a webserver to browse the local repository. It automatically directs a web browser and runs a web server through an interface into your local repository.

**Q #19) What is git is-tree?**

**Answer:** *‘git is-tree’* signifies a tree object comprising the mode and the name of all items along with the SHA-1 value of the blob or the tree.

**Q #20) Is there a way to revert a git commit that’s already been pushed and made public?**

**Answer:** Yes, to fix or revert a bad commit, there are two approaches that can be used based upon the scenario.

**They are:**

1. The very obvious way is to make a fresh commit where you remove the bad file or fix the errors in it. Once done, you can push it to a remote repository.
2. Another approach is to create a new commit to undo all changes that were done in the previous bad commit. This can be done through git revert command – “*git revert <name of bad commit>”*

**Q #21) How will you differentiate between git pull and git fetch?**

**Answer: Git pull**command pulls all new commits from a specific branch in the central repository and makes the target branch in your local repository up-to-date.

**Git fetch** also aims at the same thing, however, its underlying functionality is a bit different. When you do a git fetch, all the new commits from a specific branch will be pulled in your central repository and these changes will be stored in a new branch in your local repository. This is called a fetched branch.

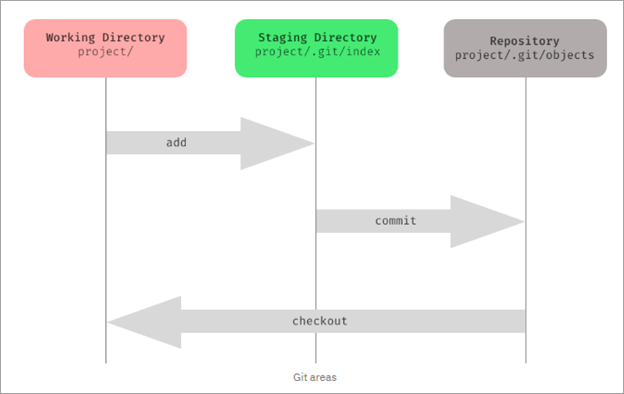
If you wish to see these changes in your target branch, then you need to perform a *git merge* after git fetch. The target branch will be updated with the latest changes only after merging it with the fetched branch.

So, a git pull brings the local branch up-to-date with its remote version, whereas a git fetch does not directly change your own local branch or working copy under *refs/heads.*Git fetch can be used to update your remote-tracking branches under **refs/remotes/<remote>/.**

In simple words, **git pull is equal to git fetch followed by a git merge**.

**Q #22) What is the use of Staging area or Indexing in Git?**

**Answer:** From Git’s perspective, there are three areas where the file changes can be kept i.e. working directory, staging area, and repository.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/of-staging-area-or-indexing-in-Git.png)

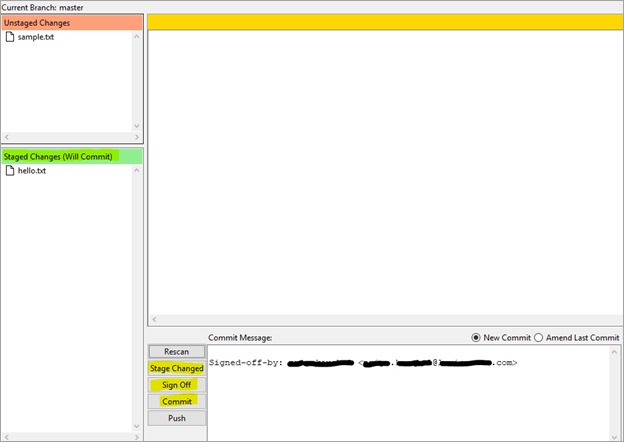
First, you make changes in your project’s working directory stored on your computer file system. All the changes remain here until you add them to an intermediate area called staging area.

You can stage the changes by executing *git add .*command. This staging area gives you a preview of your next commit and basically lets you fine-tune your commits. You can add or remove changes in the staging area until you are satisfied with the version you are going to commit.

Once you verify your changes and sign off the stage changed, then you can finally commit the changes. Upon commit, they go the local repository i.e. into .git/objects directory.

If you use Git GUI, then you will see the option to stage your changes. In the below screenshot, the file sample.txt is under unstaged changes area which means that it’s in your working directory.

You can select a file and click on ‘stage changed’, then it will be moved in the staging area. **For example**, the file hello.txt is present in stage changed (will commit) area. You can verify your changes and then do a sign-off, followed by a commit.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/Staging.png)

Staging is also referred to as indexing because git maintains an index file to keep track of your file changes across these three areas. The files which are staged are currently in your index.

When you add changes to the staging area, then the information in the index gets updated. When you do a commit, its actually what’s in the index that gets committed, and not what’s in the working directory. You can use the ***git status*** command to see what’s in the index.

**Q #23) What is Git Stash?**

**Answer:** GIT stash captures the current state of the working directory and index and keeps it on the stack for future use. It reverts the uncommitted changes (both staged and unstaged) from your working directory and returns you a clean working tree.

You can work on something else now, and when you come back, you can re-apply these changes. So, if you want to switch from one context to another without losing your current changes, then you can use stashing.

It is helpful in quick context switching, where you are in a mid-way of a code change that you don’t want to commit or undo it right now and you have got something else to work on. The command to use is git stash.

**Q #24) What is the Git Stash drop?**

**Answer:** When you no longer require a specific stash, you can remove it by executing **git stash drop <stash\_id> command**. If you want to remove all the stashes in one go from the repository then you can run **git stash clear command**.

**Q #25) What is Git stash apply? How is it different from Git stash pop?**

**Answer:** Both the commands are used to reapply your stashed changes and start working from where you had left.

In **git stash apply**command, the changes will be re-applied to your working copy and will also be kept in the stash. This command can be used when you want to apply the same stashed changes to multiple branches.

In **git stash pop** command, the changes are removed from the stash and are re-applied to the working copy.

**Q #26) What is the use of git clone command?**

**Answer:** The **git clone** command creates a copy of the existing central Git repository into your local machine.

**Q #27) When is the git config command used?**

**Answer:** The **git config**command is used to set configuration options for your Git installation.

**For example,** after you download Git, you need to use below the config commands to setup username and commit email address in Git respectively:

*$ git config –global user.name “<username>”*

*$ git config –global user.email “<email id>”*

So, mainly, things like the behavior of the repository, user information and preferences can be set up with the help of this command.

**Q #28) How will you identify if the branch is already merged into master?**

**Answer:**

**By executing the below commands, you can get to know the branch merge status:**

1. **git branch –merged master:** This will list out all the branches that have been renamed into master.
2. **git branch –merged:**This will list out all the branches that have been merged into HEAD.
3. **git branch –no-merged:**This will list out all the branches that are not yet merged.

By default, this command tells the merge status of local branches only. If you want to know about both local and remote branch merge status, then you can use *-a*flag. If you want to check only for remote branches, then you can use *-r*flag.

**Q #29) What are Hooks in Git?**

**Answer:** Git hooks are certain scripts that Git runs before or after an event like commit, push, update or receive. You will find the ‘hooks’ folder inside .git directory in your local repository. You will find the build-in scripts here pre-commit, post-commit, pre-push, post push.

These scripts get executed locally before or after the occurrence of an event. You can also modify these scripts according to your needs and Git will execute the script when that particular event occurs.

**Q #30) What is the use of git fork? How is forking different from cloning?**

**Answer:** To fork a project means to create a remote, server-side copy of the original repository. You can rename this copy, and start doing a new project around this without affecting the original project. The fork is not the core concept of Git.

The fork operation is used by Git workflow and this idea exists longer for free and open-source software like GitHub. Generally, once you have forked the project, you will rarely contribute to the parent project again.

**For example,** OpenBSD is a Unix-like open-source Operating system that was developed by forking NetBSD which is another Unix-like open-source OS.

However, in the fork, a direct connection exists between your forked copy and original repository. At any time, you can contribute back to the original project by using the pull requests.

In the forked copy, all the main data like codes and files get copied from the original repository, however, branches, pull requests and other features do not get copied. Forking is an ideal way for open source collaboration.

Cloning is essentially a Git concept. A clone is a local copy of any remote repository. When we clone a repository, the entire source repository along with its history and branches gets copied to our local machine.

Unlike forking, there is no direct connection between the cloned repository and the original remote repository. If you want to do pull requests and continue back to the original project, then you should get yourself added as a collaborator in the original repository.

Cloning is also a great way for creating a backup of the original repository as the cloned copy also has all the commit history.

**Q #31) How will you find out what all files have been changed in a particular Git commit?**

**Answer:** By using the hash value of the particular commit, you can execute the below command to get the list of files that have been changed in a particular commit:

**git diff-tree -r {hash}**

This will list down all the files that have been modified, and also the files that have been added. The -r flag is used to list individual files along with their path instead of collapsing them in their root directory names only.

**You can also use the below command:**

**git diff-tree –no-commit-id –name-only -r {hash}**

–no-commit-id will retrain the commit hash numbers to come in the output. Whereas, -name will exclude the file paths and only give the file names in the output.

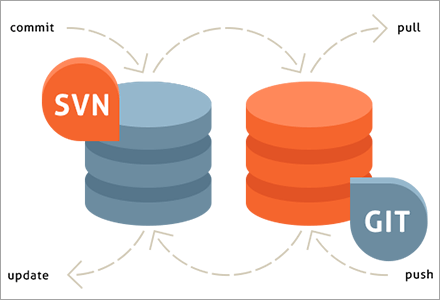
**Q #32) What is the difference between git checkout [branch name] and git checkout -b [branch name]?**

**Answer:** The command **git checkout [branch name]** will switch from one branch to another.

The command **git checkout -b [branch name]** will create a new branch and also switch to it.

**Q #33) What is SubGit?**

**Answer:** SubGit is a tool that is used for SVN to Git Migration. It is developed by a company called TMate. It converts the SVN repositories to Git and lets you do work on both the systems concurrently. It auto-syncs the SVN with Git.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/SubGit.png)

*[image*[*source*](https://subgit.com/)*]*

You can create an SVN||Git mirror using this tool. SubGit should be installed on your Git server. It will detect all the settings of your remote SVN repository, including SVN revisions, branches, and tags, and converts them into Git commits.

It also preserves the history including tracking merge data.

**Q #34) Can you recover a deleted branch in Git?**

**Answer:** Yes, you can. To recover a deleted branch, you should know the SHA off the top of your head. SHA or hash is a unique ID that Git creates with every operation.

When you delete a branch, you get the SHA displayed on the terminal:

**Deleted branch <your-branch-name> (was <sha>)**

You can use the below command to recover the deleted branch:

**git checkout -b <your-branch-name> <sha>**

If you don’t know the SHA for the commit at the tip of your branch then you can first use the **git reflog** command to know the SHA value and then apply the above checkout command to restore your branch.

**Q #35) What is** ***git diff*command? How is it different from *git status?***

**Answer:** **Git diff** is a multi-use command that can be executed to show the differences between two arbitrary commits, changes between the working tree & a commit, changes between working tree & an index, changes between two files, changes between index & a tree, etc.

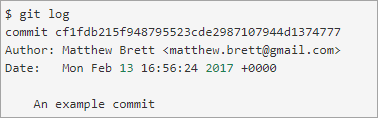
The **git status** command is used to inspect a repository. It shows the state of the working directory and staging area. It will list down the files that have been staged, which haven’t been staged and the files that are untracked.

**Q #36) What does a Commit object contain?**

**Answer:** The commit object contains the top-level tree object hash, parent commits hash(if any), author and committer information, commit date and commit message.

You can view this through the **git log** command.

**Example:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/commit-object.png)

*[image*[*source*](https://matthew-brett.github.io/)*]*

**Q #37) What is git cherry-pick? What are the scenarios in which git cherry-pick can be used?**

***Answer:*Git cherry-pick** is a powerful command to apply the changes introduced by one or more existing commits. It allows you to pick a commit from one branch and apply it to another.

***git cherry-pick commitSha*** is the command used for cherry-picking. commitSha is the commit reference.

This command can be used for undoing changes. For instance, if by mistake you have made a commit to a wrong branch, then you can check out the correct branch and cherry-pick the commit to where it should belong.

It can also be used in team collaboration. There can be scenarios where the same code needs to be shared between two components of the product. In this case, if one developer has already written that code, then the other one can cherry-pick the same.

Cherry-picking is also useful in bug hotfixes where a patch commit can be cherry-picked directly into the master branch to fix the issue as soon as possible.

**Q #38) What is ‘git reset’ is used for? What is the default mode of this command?**

***Answer:*Git reset** is a powerful command for undoing local changes to the state of a Git repo. This command resets the current HEAD to the specified stage.

It resets both the index and the working directory to the state of your last commit. Git reset has three modes i.e. soft, hard and mixed. The default mode of operation is mixed.

**Q #39) What is the difference between ‘HEAD’, ‘working tree’ and ‘index’?**

**Answer:** The working tree or workspace is the directory containing the source files that you are currently working on.

The index is the staging area in Git where the commits are prepared. It lies between the commit and your working tree. Git index is one large binary file that enlists all files in the current branch, their names, sha1 checksums, and timestamps.

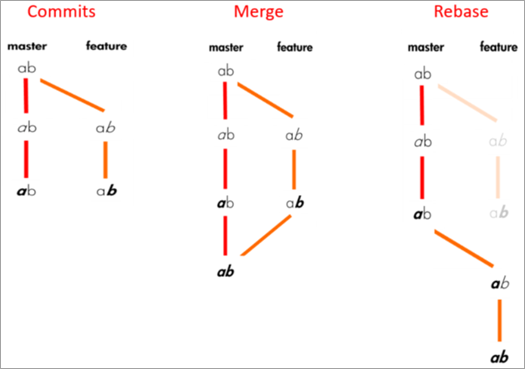
This file is present at <baseOfRepo>/.git/index. HEAD is the reference or pointer to the latest commit in the current checkout branch.

**Q #40) What’s the difference between rebase and merge? When should you rebase and when should you merge?**

**Answer:** Both rebase and merge commands are used to integrate changes from one branch to another but in a different manner.

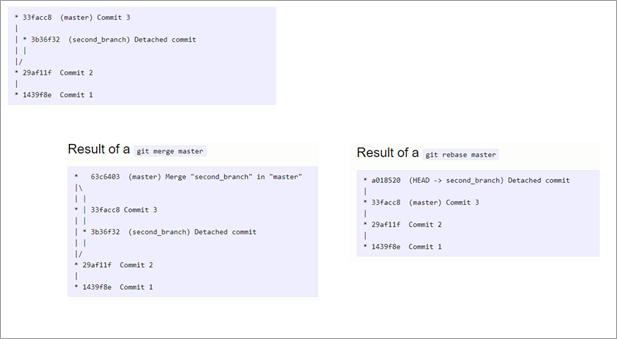
As seen in the below two images, suppose you have commits (this is before merge/rebase). After the merge, you will get the result as a combination of commits. It binds together the histories of both the branches and creates a new ‘merge commit’ in the feature branch.

On the other hand, rebase will move the whole feature branch to begin at the tip of the master branch.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/rebase-and-merge.png)

*[image*[*source*](https://medium.com/)*]*

**Commits will look like:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/01/commits.png)

Rebasing is not recommended for public branches as it creates inconsistent repositories. However, rebasing is a good option for private branches/individual developers. It is not very suitable for branch-per-feature mode. But if you have a branch-per-developer model, then rebasing is of no harm.

Also, rebase is a destructive operation, so your development team should be skilled enough to apply it correctly. Otherwise, committed work can be lost.

Furthermore, reverting a merge is easier than reverting a rebase. So, if you know that there can be possibilities for revert required, then you should use the merge.

Merge perseveres history as it is whereas rebase rewrites history. Thus, if you want to see the history completely as it occurred then you should use merge.

**Q #41) What is the syntax for rebasing?**

**Answer:** The syntax for rebase command is ***git rebase [new-commit]***

**Q #42) How will you remove a file from Git without actually removing it from your local filesystem?**

**Answer:** **You can use the ‘cached’ option for this:**

**git rm -rf –cached $FILES**

This command will remove the files from your repository without deleting them from your disk.

**Q #43) What is the common branching pattern in Git?**

**Answer:**The common branching pattern is based on the git-flow. It has two main branches i.e. master and development.

* The master branch contains the production code. All the development code is merged into the master branch at some point in time.
* The development branch contains the pre-production code. When the features are completed, they get merged to the master branch, generally through a CI/CD pipeline.

**This model also has some supporting branches that are utilized during the development cycle:**

* **Feature Branches/Topic Branches:** They are used to develop new features for upcoming releases. It may branch off from the develop branch and must be merged back into the develop branch. Generally, these branches exist only in developer repositories, and not in origin.
* **Hotfix Branches:** They are used for unplanned production release when there is a need to fix any critical bug immediately in the live prod version. They may branch off from master and must be merged back into develop and master.
* **Release Branches:** They are used for the preparation of new production release. The release branch lets you do minor bug fixes and prepare metadata for release. They may branch off from development and must be merged back into master and develop.

# **GIT Interview Questions**

### 1) What is GIT?

Git is an open source distributed version control system and source code management (SCM) system with an insistence to control small and large projects with speed and efficiency.

### 2) Which language is used in Git?

Git uses 'C' language. Git is quick, and 'C' language makes this possible by decreasing the overhead of run times contained with high-level languages.

### 3) What is a repository in Git?

A repository consists of a list named .git, where git holds all of its metadata for the catalog. The content of the .git file is private to Git.

### 4) What is 'bare repository' in Git?

A "bare" repository in Git includes the version control information and no working files (no tree), and it doesn?t include the special. git sub-directory. Instead, it consists of all the contents of the .git sub-directory directly in the main directory itself, whereas working list comprises of:

1. A .git subdirectory with all the Git associated revision history of your repo.
2. A working tree, or find out copies of your project files.

5) What is the purpose of GIT stash?

GIT stash takes the present state of the working file and index and puts in on the stack for next and gives you back a clean working file. So in case if you are in the middle of object and require to jump over to the other task, and at the same time you don't want to lose your current edits, you can use GIT stash.

6) What is GIT stash drop?

When you are done with the stashed element or want to delete it from the directory, run the git 'stash drop' command. It will delete the last added stash item by default, and it can also remove a specific topic if you include as an argument.

7) What are the advantages of using GIT?

Here are some of the essential advantages of Git:

* Data repetition and data replication is possible
* It is a much applicable service
* For one depository you can have only one directory of Git
* The network performance and disk application are excellent
* It is effortless to collaborate on any project
* You can work on any plan within the Git

8) What is the function of 'GIT PUSH' in GIT?

'GIT PUSH' updates remote refs along with related objects

9) Why do we require branching in GIT?

With the help of branching, you can keep your branch, and you can also jump between the different branches. You can go to your past work while at the same time keeping your recent work intact.

10) What is the purpose of 'git config'?

The 'Git config' is a great method to configure your choice for the Git installation. Using this command, you can describe the repository behavior, preferences, and user information.

11) What is the definition of "Index" or "Staging Area" in GIT?

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'.

12) What is a 'conflict' in git?

A 'conflict' appears when the commit that has to be combined has some change in one place, and the current act also has a change at the same place. Git will not be easy to predict which change should take precedence.

13) What is the difference between git pull and git fetch?

Git pull command pulls innovation or commits from a specific branch from your central repository and updates your object branch in your local repository.

Git fetch is also used for the same objective, but it works in a slightly different method. When you behave a git fetch, it pulls all new commits from the desired branch and saves it in a new branch in your local repository. If you need to reflect these changes in your 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**

14) How to resolve a conflict in Git?

If you need to resolve a conflict in Git, edit the list for fixing the different changes, and then you can run "git add" to add the resolved directory, and after that, you can run the 'git commit' for committing the repaired merge.

15) What is the purpose of the git clone?

The git clone command generates a copy of a current Git repository. To get the copy of a central repository, 'cloning' is the simplest way used by programmers.

16) What is git pull origin?

pull is a get and a consolidation. 'git pull origin master' brings submits from the master branch of the source remote (into the local origin/master branch), and then it combines origin/master into the branch you currently have looked out.

17) What does git commit a?

Git commits "records changes to the storehouse" while git push " updates remote refs along with contained objects" So the first one is used in a network with your local repository, while the latter one is used to communicate with a remote repository.

18) Why GIT better than Subversion?

GIT is an open source version control framework; it will enable you to run 'adaptations' of a task, which demonstrate the changes that were made to the code over time also it allows you keep the backtrack if vital and fix those changes. Multiple developers can check out, and transfer changes, and each change can then be attributed to a particular developer.

19) Explain what is commit message?

Commit message is a component of git which shows up when you submit a change. Git gives you a content tool where you can enter the adjustments made to a commit.

20) Why is it desirable to create an additional commit rather than amending an existing commit?

There are couples of reason

1. The correct activity will devastate the express that was recently saved in a commit. If only the commit message gets changed, that's not a problem. But if the contents are being modified, chances of excluding something important remains more.
2. Abusing "git commit- amends" can cause a small commit to increase and acquire inappropriate changes.

21) What does 'hooks' comprise of in Git?

This index comprises of Shell contents which are enacted after running the relating git commands. For instance, Git will attempt to execute the post-commit content after you run a commit.

22) What is the distinction between Git and Github?

A) Git is a correction control framework, a tool to deal with your source code history.

GitHub is a hosting function for Git storehouses.

GitHub is a website where you can transfer a duplicate of your Git archive. It is a Git repository hosting service, which offers the majority of the distributed update control and source code management (SCM) usefulness of Git just as including its features.

23) In Git, how would you return a commit that has just been pushed and made open?

There can be two answers to this question and ensure that you incorporate both because any of the below choices can be utilized relying upon the circumstance:

Remove or fix the bad document in another commit and push it to the remote repository. This is a unique approach to correct a mistake. Once you have necessary changes to the record, commit it to the remote repository for that I will utilize

**git submit - m "commit message."**

Make another commit that fixes all changes that were made in the terrible commit. to do this, I will utilize a command

**git revert <name of bad commit>**

24) What does the committed item contain?

Commit item contains the following parts; you should specify all the three present below:

A set of records, representing to the condition of a task at a given purpose of time

References to parent commit objects

An SHAI name, a 40 character string that uniquely distinguishes the commit object.

25) Describing branching systems you have utilized?

This question is a challenge to test your branching knowledge with Git along these lines, inform them regarding how you have utilized branching in your past activity and what reason does it serves, you can refer the below mention points:

**Feature Branching:**

A component branch model keeps the majority of the changes for a specific element within a branch. At the point when the item is throughout tested and approved by automated tests, the branch is then converged into master.

**Task Branching**

In this model, each assignment is actualized on its branch with the undertaking key included in the branch name. It is anything but difficult to see which code actualizes which task, search for the task key in the branch name.

**Release Branching**

Once the create branch has procured enough features for a discharge, you can clone that branch to frame a Release branch. Making this branch begins the following discharge cycle so that no new features can be included after this point, just bug fixes, documentation age, and other release oriented assignments ought to go in this branch. When it is prepared to deliver, the release gets converged into master and labeled with a form number. Likewise, it should be converged once again into creating a branch, which may have advanced since the release was started.

At last, disclose to them that branching methodologies fluctuate starting with one association then onto the next, so I realize essential branching activities like delete, merge, checking out a branch, etc.

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

The appropriate response is immediate.

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.

27) How might you fix a messed up submit?

To fix any messed up commit, you will utilize the order "git commit?correct." By running this direction, you can set the wrecked commit message in the editor.

28) Mention the various Git repository hosting functions.

The following are the Git repository hosting functions:

* Pikacode
* Visual Studio Online
* GitHub
* GitEnterprise
* SourceForge.net

29) Mention some of the best graphical GIT customers for LINUX?

Some of the best GIT customer for LINUX is

1. Git Cola
2. Smart git
3. Git-g
4. Git GUI
5. Giggle
6. qGit

30) What is Subgit? Why use it?

'Subgit' is a tool that migrates SVN to Git. It is a stable and stress-free migration. Subgit is one of the solutions for a company-wide migration from SVN to Git that is:

1. It is much superior to git-svn
2. No need to change the infrastructure that is already placed.
3. It allows using all git and all sub-version features.
4. It provides stress ?free migration experience.

## Q1. What is Git?

**Ans.** Git is an open-source project distributed version control system (DVCS). Many commercial projects rely on Git as every developer’s code copy is also treated as a repository, which contains all changes done in the past. Below is the detailed description of DVCS:

* **Control System:** Git is known for its features like a content tracker, and it stores content.
* **Version Control System:** It helps developers to store code at the same time and Git modifies as and when more codes are added. The version control system helps in maintaining and keeping records of all the changes. Further, it offers features like branches and merges.
* **Distributed Version Control System:** Git has a remote repository and a local repository, which are stored on servers and computers, respectively. This means that code is stored in both the central server and the developer’s computer. Hence it is termed a distributed version control system.

## Q2. Why is it said that Git is designed keeping in mind performance, security, and flexibility?

**Ans.** Git was developed in 2015 by Linus Torvalds for Linux kernel development. But in the last decade, it gained a lot of interest, and today, due to its flexibility, nearly every development environment uses Git and runs Git command-line tools on every major operating system.

Below are the reasons why Git is popular:

* **Performance:** Git has very powerful raw performance characteristics be it branching, merging, or comparing the past versions; it is robust and optimized. Git gives special attention to the content, and it uses a blend of delta encoding and compression. Further, it also clearly stores directory contents and metadata object versions.
* **Security:** Integrity is the topmost priority of Git. Its cryptography hashing algorithm named SHA1 safely stores all objects in the Git repository and maintains a true relationship between files and directories.
* **Flexibility:** From supporting nonlinear development workflow to adaptability with various systems and protocols, Git is exceptionally elastic. Git’s amazing tracking system offers features like treating branching and tagging as first-class citizens. Its ‘change history’ also features stores operations affecting branches and tags.

## Q3. What is the meaning of the commands – git status, git log, git diff, git revert <commit>,  git reset <file>?

**Ans.**

|  |  |
| --- | --- |
| **Command** | **Meaning** |
| git status | Gives a list of which files are staged, unstaged, and untracked |
| git log | Illustrates the entire commit history by using the default format |
| git diff | Displays the unstaged changes between index and working directory |
| git revert <commit> | Undoes all the changes made in <commit> and applies it to the current branch by creating a new commit |
| git reset <file> | Removes <file> from a staging area without overwriting any changes by keeping the working directory unchanged |

## Q4. What is Commit?

**Ans.** Git creates a commit as and when a developer saves any new work. Commit is a screenshot of all the files, and Git will use the previously used file if a file is not changed from one commit to another. One commit creates a chain to other commits and forms a development history graph. Unique cryptocurrency hash identifies commit in Git.

## Q5. What are Branches in Git?

**Ans.** As multiple developers work parallel on a program, they create their own local repository, and this creates multiple changes in a single commit. However, in Git, branches manage various separations, and once the work in a branch is finished, it is merged with the master branch.

## Q6. Are Git and GitHub the same thing?

**Ans.** Git and GitHub are connected as [Github](https://www.naukri.com/learning/what-is-github-st293-tg639) is a service to use Git, but they both have slight differences:

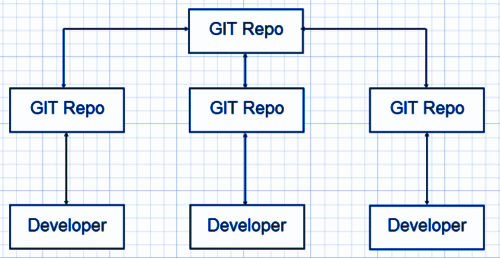
|  |  |
| --- | --- |
| **Git** | **GitHub** |
| Git is a software tool to use a version control system | GitHub is a hosting service for git repositories |
| Tool for projects that want to collaboratively develop software | Service for projects that use Git for version control |

## Q7. Which is a better version control system – Git or SVN?

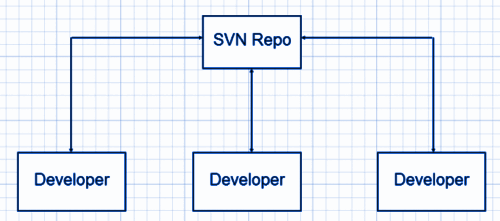
**Ans.** Both Git and SVN have their pros and cons, and below are the differences between Git and SVN:

|  |  |
| --- | --- |
| **Git** | **SVN** |
| Decentralized and distributed version control tool | Centralized version control tool |
| Clones all repositories on the local system | Stores version history on the server-side repository |
| Supports offline commits | Supports online commits only |
| Swift push/pull operations | Slow push/pull operations |
| Automatically shares work to commit | Doesn’t support automatic sharing |

**Git Workflow:**



**SVN Workflow**



## Q8. Name some Git repository hosting functions.

**Ans.** Below is the list:

* Github
* Gitlab
* Bitbucket
* SourceForge
* GitEnterprise

## Q9. State the difference between “git pull” and “git fetch.”

**Ans.** This is an important Git interview question. “git pull” and “git fetch” are used for downloading new data from a remote repository.

“git fetch – It downloads new data from the repository but does not support integrating this data into working files. It offers a fresh view of things that happened in the remote repository.

“git pull” – This command is used to update the current HEAD data branch with all the changes that occurred in the remote repository. Thus, it downloads the data and integrates it with existing working files.

## Q10. How do you edit or fix the last commit message in Git?

**Ans.** If you forget to add anything in the commit message or committed a typo error, you can rectify it by using the –amend flag command.

$ git commit –amend -m “Sorry I missed an important update”

Note: –amend flag will only help in editing or fixing the last commit message.

***Explore the***[***7 Most In-Demand Tech Skills***](https://www.naukri.com/learning/articles/trending-tech-skills-to-master/)

## Q11. How can you change any older commit messages?

Ans. To change an older commit the command is –

$ git rebase –interactive

## Q12. How to deal with huge binary files in Git?

**Ans.** Handling large binary files is a significant problem in git, and to handle this problem “Large File Storage” extension works well for Git. Simply install LFS on your local computer, and after this, your large files will not be stored in the local repository. Instead, it will be stored in a dedicated LFS cache and store.

**Also Read>>**[***Trending Tech Skills in: Cloud, Game development and DevOps***](https://www.naukri.com/learning/articles/trending-tech-skills-in-2020-cloud-game-development-and-devops/)

## Q13. How to resolve and solve merge conflicts?

**Ans.** It is very easy to resolve merge conflicts as Git allows you to go back to the previous state. Just use a “git merge –abort” command, and you will be able to undo the merge and start the task again.

## Q14. What do you mean by “git cherry-pick”?

**Ans.** If by mistake, you have committed a change into the wrong branch, you can use the “git cherry-pick” command. This command will allow you to apply commit from one branch to another branch.

$ git cherry-pick <commit id>

## Q15. In which scenario do you use the “git cherry-pick” command?

Ans. Git cherry-pick command can sometimes result in duplicate commits, and thus, it must be cautiously used. The below situations are apt if planning to use the git cherry-pick command:

* When you mistakenly make a commit in the wrong branch
* When you want to make changes that are proposed by other team members

## Q16. Name some of the Git tools that you use.

**Ans.** This is a commonly asked Git interview question. Below is the list of most popular Git tools:

|  |  |
| --- | --- |
| **Git Tools** | **License Type** |
| GitHub Desktop | MIT |
| GitKraken | Proprietary |
| SmartGit | Proprietary |
| Tower | Proprietary |
| Git Up | GNU GPL |

## Q17. What do you mean by the bare repository in Git?

**Ans.** While initializing a new Git repository, – run git init function is used, and this directory becomes a ‘Working Tree.’ Also, Git creates its own .git directory (which is hidden) where it tracks all the changes and stores the commit objects.

However, a bare repository, also called bare repos, works without creating a ‘Working Tree.’ This bare repository is utilized as a remote repository. It helps share it

with all the users where the developers will clone it and locally make the required changes.

## Q18. Why do developers use Git Clone?

**Ans.** Developers prefer cloning as it is the simplest way to get a copy of the central repository. The ‘git clone’ command helps in generating a copy of the current Git repository.

## Q19. Will you create an additional commit or amend an existing commit?

**Ans.** It is preferable to create an additional commit because:

* It might cause inappropriate changes
* A correct activity that was recently saved in a commit might ruin the express
* The chances are high that you miss including significant remains

***Check out the list of***[***Top Online IT Courses***](https://www.naukri.com/learning/articles/top-online-courses-for-it-professionals/)

## Q20. Which branching strategies have you used?

**Ans.** To answer this Git interview question, you can share all the branching strategies that you have used. You can frame your answer as follows:

I have used –

* **Feature Branching:** keeps all of the changes for a particular feature inside of a branch.
* **Task Branching:** In this, each task is implemented on its own branch with the task key included in the branch name
* **Release Branching**: After the develop branch has acquired enough features for a release, that branch can be cloned to form a Release branch. This starts the next release cycle. Now you can not add new features after this point. Only bug fixes, documentation generation, etc. can be added to this branch. When it is ready to ship, the release gets merged into the master and gets a version number.

## Q21. Name some of the most popular Git repository hosting functions.

**Ans.** Below is the list of Git repository hosting functions:

* Pikacode
* Assembla
* Visual Studio Online
* GitHub
* GitEnterprise
* net
* Beanstalk
* CloudForge
* GitLab
* Planio
* Perforce
* Fog Creek Kiln

## Q22. Why do you use Subgit?

**Ans.** The Subgit is a popular tool used for stress-free transferring of SVN to Git and it allows using various Git and sub-version features.

## Q23. State the difference between HEAD, working tree, and index.

**Ans.** The working tree, also known as the working directory or workspace, is the directory tree of source files. Whereas, the index, which is also known as the staging area, lists all the files in the current branch.

HEAD is known as the last commit, which was marked in the check-out branch.

## Q24. Name some Git GUI clients for Linux.

**Ans.** Below is the list of Git GUI clients for Linux:

|  |  |  |  |
| --- | --- | --- | --- |
| Git Clients for Linux | Clients for Windows | Clients for Mac | Cross-platform Git Clients |
| Git Force | Tortoise Git | GitX-dev | Aurees |
| Gitg | GitHub | GitBox | SmartGit |
| QGit | Sourcetree | GitUp | GitKraken |

## Q25. Explain the benefits of using the Version Control System (VCS)?

**Ans.** The benefits of the Version Control System (VCS) are as follows:

* All team members can work freely on any file at any time
* Allow us to compare files, identify differences, and merge the changes into a common version
* Keep a track of application builds by determining which version is currently in development, QA, and production
* Allows all team members to have a complete history of the project.

## Q26. What is a Git repository?

**Ans.** A Git repository is a place that has a collection of files of different versions of a Project. Git stores these files either on the local repository or the remote repository. There are two types of repositories:

* Bare Repository: contains the .git folder
* Non-bare Repository: contains both the git index and the checked-out copy of working files

## Q27. What is git instaweb? How is it used?

**Ans.** A git instaweb is a script that helps to set up a temporary instance of GitWeb on a web server for browsing local repositories. It requires a lightweight server such as Lighttpd or Webrick. It is used to automatically direct a web browser and run a webserver with an interface into the local repository.

## Q28. Explain Git stash.

**Ans.** A git stash is a place where you can temporarily stash (or store) changes made to the working copy so we can work on something else, and then come back and reapply them afterward. A git stash is separate from the staging area, the working directory, or the repository.

## Q29. What are the benefits of forking workflow?

**Ans.** The benefits of forking workflow are as follows:

* The contributions can be integrated without requiring everyone to push to a single central repository.
* Developers can push to their own server-side repositories while only the project maintainer can push to the official repository.
* A maintainer can accept commits from any developer without providing them write access to the official codebase.

## Q30. What do you mean by the Gitflow workflow?

**Ans.** This is an important Git interview question. The Gitflow Workflow specifies a branching model for Git. It provides a framework for managing large projects and is mostly used for projects that have a scheduled release cycle. Gitflow assigns very specific roles to different branches and defines how and when they should interact:

* **Master:** This branch is always ready to be released on LIVE. It releases when everything is fully tested and approved.
* **Develop:** All feature branches are merged into this branch and all tests also are performed here. When everything is thoroughly checked, it can be merged into the master.
* **Feature:** Each new feature should reside in its own branch, which can be pushed to develop branch as their parent branch.
* **Hotfix:** These branches are used to quickly patch production releases. They are based on master instead of develop.

## Q31. What is the difference between git remote and git clone?

**Ans.** With git remote, you can create, view, and delete connections to other repositories. It’s used to refer to a remote repository or a central repository.

|  |  |
| --- | --- |
| **git remote** | **git clone** |
| Allows you to create, view, and delete connections to other repositories. | Enables you to create a clone or copy of the target repository. |
| Targets a remote repository or a central repository. | It targets a different already existing repository. |

## Q32. How will you find out if a branch has already been merged or not?

**Ans.** We use the following commands to find out if a branch has already been merged or not:

* **git branch –merged master –**it will list all the branches that have been renamed into master.
* **git branch –merged** – it lists the branches that have been merged into the current branch (HEAD).
* **git branch –no-merged** – it lists the branches that have not been merged.

## Q33. What does the commit object contain?

**Ans.** The commit object contains the following:

1. A set of files that represents the state of a project at a certain time
2. Reference to parent commit objects
3. SHAI name – a 40 character string that uniquely identifies the commit object

## Q34. What is the syntax for Rebasing in git?

**Ans.** The syntax for Rebasing in Git is:

git rebase [new-commit]

***Check out the***[***10 Best Software Development Tools and Frameworks to Learn***](https://www.naukri.com/learning/articles/best-software-development-frameworks-and-tools-to-learn/)

## Q35. Why are the Git Stash Drop and Git Stash Clear commands used?

**Ans.** Git Stash drop command or <stash\_id> is used to remove a particular stash that is not required.

Git stash clear command is used when all the stashes are to be removed in one go from the repository.

## Q36. Explain Git Hooks.

**Ans.** Git hooks are simple scripts that run before or after certain actions, such as commit, push, update, or receive. They consist of of shell scripts that are activated when you run the corresponding [Git commands](https://www.naukri.com/learning/articles/git-commands-with-examples/). Git Hooks are useful in many tasks, such as client-side validation.

## Q37. What is the difference between revert and reset?

**Ans.** The differences between revert and reset are:

|  |  |
| --- | --- |
| **Revert** | **Reset** |
| It creates a new commit that undoes the changes made in the previous commit. | It undoes the local changes that have been made to a Git repository. |
| New history is added to the project and the existing history is not modified. | This command may alter existing history. The Reset command operates on the commit history, the staging index, and the working directory. |
| Command: git revert | Command: git reset |

## Q38. Explain the functions of the git reset –mixed and git merge –abort commands.

**Ans.** The **git reset –mixed** command undoes the changes made in the working directory and staging area.

The **git merge –abort** command stops the merge process and returns to the state before the merging began.

## Q39. What is the difference between Git stash apply and Git stash pop?

**Ans.** The ‘Git stash apply’ and ‘Git stash pop’ commands are used when you have to reapply the stashed changes and start working from where you left.

The difference between them is that while the ‘Git stash apply’ command keeps the changes in the stash list for later use, the ‘Git stash pop’ command removes the changes from the stash after applying it.

***Also explore:***

* Paid and [free online courses](https://www.naukri.com/learning/coursera-courses-certification-training-v475) by Coursera
* Popular online [Udemy courses](https://www.naukri.com/learning/udemy-courses-certification-training-v21)
* Top online [edX courses](https://www.naukri.com/learning/edx-courses-certification-training-v117)

## Q40. Explain the role of the git-add command.

**Ans.** The git-add command adds new or changed files in your working directory to the Git staging area. Running the git add command will not change any of your work in the Git repository. Changes are only made to your repository when you execute the git commit command.

In simple terms, when you change and save a file (or multiple files, then, before you commit, you must git add. The git add command selects that file, and moves it to the staging area, for inclusion in the next commit. You can select a specific file, all files, a directory, or specific parts of a file for staging and commit. We can perform the add command multiple times before a commit.

**Below is the syntax for the git add command:**

git add [filename]

## Q41. How to delete a branch in Git?

**Ans.** Deletion of Git branches is done after you have merged a branch into your codebase. To delete a branch we can use the command: **git branch -d branch\_name.**Below are the commands to delete a git branch locally or remotely:

**Deleting a branch Locally**

You can delete a git branch on your local machine using the command: **git branch -d <local\_branch\_name>**

**Deleting a branch Remotely**

You can delete a git branch remotely using the command: **git push origin –delete <remote\_branch\_name>**

## Q42. What is git reflog?

**Ans.** Reference logs like the commit information of when the branch was created, checked out, renamed, etc. are recorded by the reflog command. This command tracks the changes made in the repository references (branches or tags). It also records and manages the branches/tags log history that was either created locally or checked out.

Every action you perform inside of Git where data is stored can be found inside of the reflog. So, if if you think that merge, rebase, or some other action has destroyed your work, you can find it again using the reflog command.

**Explore**[**Free Online Courses with Certificates**](https://www.naukri.com/learning/free-online-courses-certificates)

## Q43. Explain the role of the git annotate command.

**Ans.** The git annotate command tracks each line of the file based on the commit information. It annotates each line within the given file with information from the commit which introduced that change. The annotate command can also annotate from a given revision.

**Below is the Syntax of the git annotate command:**

git annotate [<options>] <file> [<revision>]

## **Q44. Name the command for creating an empty Git repository.**

**Ans.** Below is the command that helps create an empty Git repository:

git init

## **Q45. Explain the function of git ls-tree.**

**Ans.** The main function of the git ls-tree command is to list the contents of a tree object.

## **Q46. What is the role of the git clean command?**

**Ans.** The main function of the git clean command is to remove the untracked files from the working directory.

## **Q47.** Explain the role of the git config command with examples.

**Ans.** The use of git config command is to get and set git configuration values on a global or local project level. It uses your username to associate commits with an identity. You can also change your Git configuration, including your username.

**For example:**

You can give a username and email id to associate a commit with an identity. This will help you know who has made that commit.

git config –global user.name “Your Name”: It will add a username.

git config –global user.email “Your E-mail Address”: It will add an email id.

## Q48. What is Git bisect? How does it help to determine the source of a (regression) bug?

**Ans.** Git bisect command uses a binary search algorithm to find the commit in your project’s history that introduced a bug in your code. The git bisect command divides the history of your project into the good and the bad commit range. It points your current project state to a mid-range commit snapshot. Now, this command moves through every commit id between this range while pausing at each snapshot to allow you to test the code. You declare the commit as bad if the bug exists.

The Syntax for the Git bisect command is:

git bisect <subcommand> <options>

## Q49. How to squash the last N commits into a single commit?

**Ans.** The following are the two ways to squash the last N commits into a single commit:

* Use the below command to write the new commit message from scratch

git reset –soft HEAD~N &&git commit

* To start editing the new commit message with a concatenation of the existing commit messages, you will have to get those messages and pass them to commit:

git reset –soft HEAD~N &&git commit –edit -m”$(git log –format=%B –reverse .HEAD@{N})”

## Q50. What is a .git Directory?

**Ans.** A .git directory consists of all the metadata of the repository. It also keeps a track of all the changes made to the files in your repository, by keeping a commit history. It keeps all information related to commits, hooks, refs, object databases, etc.

When you create a repository, you will find a .git directory inside it. If you clone any git repository on your local machine, the .git is the directory that gets copied.

**1. What is a clone in GitHub?**

Cloning a Git repository means we can create a local copy of the code provided by the developer. You can simply do it with a command line: git clone git://github.com/facebook/facebook-ios-sdk.git. and we can have the code in the facebook-ios-sdk directory.

|  |
| --- |
| **Explore**[**Git Tutorial**](https://mindmajix.com/git-tutorial)**for more information** |

**2. How much space do we get on GitHub?**

We get a space of 1 GB but if it exceeds 1GB, we receive a polite email from GitHub Support requesting to reduce the size of the repository and scale it down. In addition, here we placed a limit of files exceeding 100 MB in size.

**3. What do you know about GitHub and its repository?**

It is basically a source code management system that can be considered for both small as well as large-scale software development projects. Generally, it is widely preferred for error-free and reliable computer code. Although the users can keep up the pace with SCM, it is also possible for them to add features as per their preference. A Repository is basically the directory of Git where the metadata of the same is stored. The data might be shared or private depending on the project.

**4. How it is possible for you as a user of Git to define the information of a user, behavior of a repository as well as the information of preferences in the programming?**

This can simply be done with the help of a command named Git config. Although there are other methods getting the results through the command prompt always makes sure of originality and reliability.

**5. What exactly do you know about the GIT stash?**

It is used when there is a need of storing the current state of a project so that the users can continue with the same at a later stage. There is often a need to switch to another job when one is active and developers can simply keep up the pace with such a situation with the Stash. It simply enables the users not to lose their edits.

**6. Name the tool that can be deployed for Git migration?**

SubGit

**7. Can you tell us a few benefits of using GitHub over other platforms?**

There are a very large number of benefits that developers can easily make sure of with this approach. The very first one is the high availability of GitHub along with excellent support. In addition to this, all the users can simply make sure of the data replication, as well as the redundancy of the same. Moreover, the error-free outcomes are exactly what for which GitHub is widely appreciated.

It is a platform independent and users can easily get the results in the desired manner without compromising with anything.  Also, it is collaboration-friendly and users can simply use it in the way they are comfortable.

**8. What language is considered in Git and what is the benefit of same in this approach according to you?**

Git is purely based on the C and the same makes sure of imposing a limit on the overhead of runtimes which are generally associated with other platforms in its class. Also, c makes it compatible with all the other domains and the developer’s already existing work.

**9. Compare Git with SVN?**

When it comes to handling data with large size, Git is not widely preferred. However, for users, it is possible to handle more than one project with the latter provided they remain in the same repository. In multiple branches, Git fails to support the commits while the SVN can do so easily.

**10. When it comes to software development, what are the major factors the user should be careful about?**

The software should be developed by understanding the exact needs of the client or the task which it has to perform. It should be rich in terms of features and API. Moreover, it should be secure and reliable enough to be trusted by the organizations. There are other factors such as the length of the code and the factors that can influence the same which are also necessary to pay attention to.

**11. On what projects you have already worked on which are based on Github?**

This question is often asked in IT interviews. You need to give a short or a detailed overview of the projects you have handled, the problems you faced, the outcome of the project, the benefits organization and you as a developer derived from it, the scope of the project, and the time taken to complete it. Moreover, you should mention what sort of experience you derived from them.

**12. What is the upper limit on the heads in the Gits?**

There is no limit and users are free to involve any number of heads in the repository. It can be considered as a simple standard reference to a committed object. The commit object couldn’t be the same for all the heads.

**13. Tell us whatever you know about the Github development process?**

It is basically nothing but quite similar to that of a life cycle of any specific software. Thus, you should have knowledge about the life cycle of software and the factors that can directly influence the same. There are actually several activities that are a part of the process and they are:

1. Analysis of the requirement
2. Specifications of the project
3. The architecture of the software
4. Real time implementation
5. Testing of software
6. Documentation and reporting
7. Maintenance
8. Training and support available with the same

**14. Do you think GitHub is better as compared to Subversion of same? Why or why not?**

One of the best things about Github is it’s an open-source technology where the developers are free to run the versions of their projects without worrying about anything. This is exactly what enables the developers to have a quick review of all the changes made to the code over time. The users can also keep track of actual code before modification. With a subversion, nothing like this is possible. Therefore, Github is a better option.

**15. What was your biggest problem to date while working on a Github project?**

Generally, problems are quite common in any development process. Depending on the nature and type of tasks, you might have faced a lot of issues. You can mention them all here and can genuinely answer how they were actually sorted out. Problem solving is actually learning how to tackle the challenges.

**16. How many characters are there in the SHAI name?**

It is a 40 Character String that can vary in some special cases.

**17. Can you put the Computer software and computer program separate from one another by a simple comparison?**

Basically, a computer program is nothing but part of a programming code and the same is responsible for the successful execution of a task. On the other hand, computer software includes the code for programming including the guide on how to use it and concerned documents.

**18. How it is possible for a developer to simply update the remote references related to the different objects?**

This is possible through the Git PUSH. In fact, it is actually the prime function of the same.

**19. What do you know about the significance of software development?**

In the century we are living in, the overall time is taken to accomplish a task largely matters when it comes to the corporate level. The software is powerful in saving a lot of time. Also, they impose a strict upper limit on the dependency of a task on humans. The tasks are generally governed, controlled, monitored, and accomplished by the software in the current time. Thus, the scope is software development is booming and users should pay attention to using the best available technology of the same.

**20. Is it possible in Git to merge a branch into the master? How can you find the same?**

Yes, it is possible and the users can easily keep the pace up with the same anytime. The users can directly check the list under the branch section to know more about this.

**21. Are you familiar with the Git Clone?**

It is basically a command which is deployed when it comes to copying a Git repository that already exists. There are a lot of programmers who make use of this. The best thing is it really doesn’t matter whether the project is large or small, the same can easily be considered whenever the need for the same is felt.

**22. Name any two Git repository hosting services that are common**

These are Visual Studio Online and Git Enterprise

**23. What is SHAI name?**

It is basically a string character that is responsible for the identification of the committed objects. The fact is users are free to make the changes to the default commit objects and the same is used for knowing and locating the overall changes made with a track record of the same.

**24. Is it possible to create a repository in Git? What is the step that needs to be performed before doing the same?**

Yes, it is possible and the users have to first create a directory. The same defines the project under consideration and the information related to the same.

**25. What is branching it Git and what are its benefits?**

The users are free to make as many branches as they want.  A branch is nothing but a set of tasks that is created by the users. While performing any task, whenever the interrupt arrives, the user can switch to another branch on priority and can accomplish the same first. The users can easily switch to the previous branch without compromising with anything. It actually boosts up the speed and enables users to perform multiple tasks at the same time.  Branches are generally marinated as one in the Github.

**26. What is Gitlog and when you can use it?**

It is basically a command that can be executed when it comes to finding the history of a project according to the date, changes made, the developer who handled it, and the usefulness of the same.

**27. What is the conflict situation in Git and how it can be solved?**

Conflict is a situation when both the commits that need to be merged have changed at the same place. This often confuses the software which change should be taken into consideration and which one should be neglected. The best manner to solve this concern is to simply edit the files through the appropriate procedure.

**28. Can you name an alternative method for performing the merging task in Git?**

This is called Rebasing. Generally, the users don’t prefer this method and this is because it takes a lot of time for the accomplishment of a lot of tasks that matter a lot. Sometimes a lot of unexpected errors can declare their presence in case this task is not performed accurately. This is a method that is useful only for those who have great experience in Git technology.

**29. Name a few graphical Git clients for the Linux platforms?**

These are

* Git-g, Git cola
* Git GUI
* Giggle

These clients can easily be used in conjunction with each other or they can be deployed independently

**30. What is the significance of Git version control?**

When it comes to simply track the background of an array of files and changing their version, this approach is generally preferred. This actually works by capturing the snapshot of the moments and tasks associated. All the information remains present in the repository. However, if the users want, they can simply keep it at any desired location.

**31. What do you mean by the Commit message?**

It is basically a message which you can see on the screen while working on Git whenever a change is committed. It is possible to keep a record of all the changes made by the user in an editor. The history of changes needs to be explored lately for a specific task.

**32. How can the users enhance the functionality of a branch in Git?**

It is possible to do so by adding the desired feature in any of the branches. This is generally done through a command Git merge and the best part is there is no limit on adding the features in the branch. Any branch can have any number of features.

**33. What is the gist of Git?**

Gists are a great way to share the work of any developer. They can share parts of files, full applications, or single files. Anyone can access gists at https://gist.github.com. Each Gist is a Git repository, which means, it can be forked and cloned.

**34. How can we create a gist?**

Creating a gist requires a very simple process as depicted in the steps below: -

1. Sign in to GitHub.
2. We should navigate to the gist home page.
3. After this, we need to type an optional description and name for the gist.
4. Key in the text of your gist into the gist text box.
5. Following this, we should select either to create a public gist or to create a secret gist.

**35. What is gist programming?**

GitHub provides a hosting service that facilitates a web-based Git repository. It includes all the functionality of Git with additional features added in. The gist is an additional attribute added to GitHub, which facilitates the sharing of code snippets, notes, to do lists, and more. We can save our Gists as secret or public in the repository.

## **Basic Questions**

### ****1. What is the difference between Git and SVN?****

|  |  |
| --- | --- |
| **Git** | **SVN** |
| Git is a Decentralized Version Control tool | SVN is a  Centralized Version Control tool |
| It belongs to the 3rd generation of Version Control tools | It belongs to the 2nd generation of Version Control tools |
| Clients can clone entire repositories on their local systems | Version history is stored on a server-side repository |
| Commits are possible even if offline | Only online commits are allowed |
| Push/pull operations are faster | Push/pull operations are slower |
| Works are shared automatically by commit | Nothing is shared automatically |

### ****2. What is Git?****

I will suggest you attempt this question by first telling about the architecture of git as shown in the below diagram just try to explain the diagram by saying:

* Git is a Distributed Version Control system(DVCS). It lets you track changes made to a file and allows you to revert back to any particular change that you wish.
* It is a distributed architecture that provides many advantages over other Version Control Systems (VCS) like SVN. One of the major advantages is that it does not rely on a central server to store all the versions of a project’s files.
* Instead, every developer “clones” a copy of a repository I have shown in the diagram with “Local repository” and has the full history of the project available on his hard drive. So when there is a server outage all you need to do to recover is one of your teammate’s local Git repository.
* There is a central cloud repository where developers can commit changes and share them with other teammates.

### ****3. What is a distributed VCS?****

* These are the systems that don’t rely on a central server to store a project file and all its versions.
* In Distributed VCS, every contributor can get a local copy or “clone” of the main repository.
* As you can see in the above diagram, every programmer can maintain a local repository which is actually the copy or clone of the central repository which is present on their hard drive. They can commit and update their local repository without any hassles.
* With an operation called “pull”, they can update their local repositories with new data from the central server and “pull” operation affects changes to the main repository from their local repository.

### 

### ****4. What is the difference between Git and Github?****

[Git](https://bit.ly/31MeW9b) is a version control system of distributed nature that is used to track changes in source code during software development. It aids in coordinating work among programmers, but it can be used to track changes in any set of files. The main objectives of Git are speed, data integrity, and support for distributed, non-linear workflows.

[GitHub](https://bit.ly/2rVhL7Q) is a Git repository hosting service, plus it adds many of its own features. GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, basic task management tools for every project.

### ****5. What are the benefits of using Version Control System?****

* With the Version Control System(VCS), all the team members are allowed to work freely on any file at any time. VCS gives you the flexibility to merge all the changes into a common version.
* All the previous versions and variants are neatly packed up inside the VCS. You can request any version at any time as per your requirement and you’ll have a snapshot of the complete project right at hand.
* Whenever you save a new version of your project, your VCS requires you to provide a short description of the changes that you have made. Additionally, you can see what changes are made in the file’s content. This helps you to know what changes have been made in the project and by whom.
* A distributed VCS like Git allows all the team members to have a complete history of the project so if there is a breakdown in the central server you can use any of your teammate’s local Git repository.

### ****6. What language is used in Git?****

Instead of just telling the name of the language, you need to tell the reason for using it as well. I will suggest you to answer this by saying:

Git uses ‘C’ language. GIT is fast, and ‘C’ language makes this possible by reducing the overhead of run times associated with high-level languages.

### ****7. Mention the various Git repository hosting functions.****

* Github
* Gitlab
* Bitbucket
* SourceForge
* GitEnterprise

### ****8. What is a commit message?****

The command that is used to write a commit message is “**git commit -a**”.  
Now explain about -a flag by saying -a on the command line instructs git to commit the new content of all tracked files that have been modified. Also, mention you can use “**git add <file>**” before git commit -a if new files need to be committed for the first time.

### ****9. How can you fix a broken commit?****

In order to fix any broken commit, use the command “git commit --amend”. When you run this command, you can fix the broken commit message in the editor.

### ****10. What is a repository in Git?****

Repository in Git is a place where Git stores all the files. Git can store the files either on the local repository or on the remote repository.

### ****11. How can you create a repository in Git?****

This is probably the most frequently asked question and the answer to this is really simple.

To create a repository, create a directory for the project if it does not exist, then run the command “**git init**”. By running this command .git directory will be created in the project directory.

### ****12. What is ‘bare repository’ in Git?****

A “bare” repository in Git contains information about the version control and no working files (no tree) and it doesn’t contain the special .git sub-directory. Instead, it contains all the contents of the .git sub-directory directly in the main directory itself, whereas the working directory consists of :

1. A .git subdirectory with all the Git related revision history of your repository.
2. A working tree, or checked out copies of your project files.

### ****13. What is a ‘conflict’ in git?****

Git can handle on its own most merges by using its automatic merging features. There arises a conflict when two separate branches have made edits to the same line in a file, or when a file has been deleted in one branch but edited in the other. Conflicts are most likely to happen when working in a team environment.

### ****14. How is git instaweb used?****

‘git instaweb’ is used to automatically direct a web browser and run a webserver with an interface into your local repository.

### ****15. What is git is-tree?****

‘git is-tree’ represents a tree object including the mode and the name of each item and the SHA-1 value of the blob or the tree.

### ****16. Name a few Git commands and explain their usage.****

          Below are some basic Git commands:

|  |  |
| --- | --- |
| Command | Function |
| git rm [file] | deletes the file from your working directory and stages the deletion. |
| git log | list the version history for the current branch. |
| git show [commit] | shows the metadata and content changes of the specified commit. |
| git tag [commitID] | used to give tags to the specified commit. |
| git checkout [branch name]  git checkout -b [branch name] | used to switch from one branch to another.  creates a new branch and also switches to it. |

## **Intermediate level Questions**

### ****17. How to resolve a conflict in Git?****

The following steps will resolve conflict in Git-

1. Identify the files that have caused the conflict.
2. Make the necessary changes in the files so that conflict does not arise again.
3. Add these files by the command git add.
4. Finally to commit the changed file using the command git commit

### ****18. In Git how do you revert a commit that has already been pushed and made public?****

There can be two approaches to tackle this question and make sure that you include both because any of the below options can be used depending on the situation:

* Remove or fix the bad file in a new commit and then push it to the remote repository. This is the most obvious way to fix an error. Once you have made necessary changes to the file, then commit it to the remote repository using the command: git commit -m “commit message”
* Also, you can create a new commit that undoes all changes that were made in the bad commit. To do this use the command

git revert <name of bad commit>

### ****19. What is SubGit?****

SubGit is a tool for SVN to Git migration. It can create a writable Git mirror of a local or remote Subversion repository and use both Subversion and Git as long as you like.

Now you can also include some advantages like you can do a fast one-time import from Subversion to Git or use SubGit within Atlassian Bitbucket Server. We can use SubGit to create a bi-directional Git-SVN mirror of an existing Subversion repository. You can push to Git or commit to Subversion as per your convenience. Synchronization will be done by SubGit.

### ****20. What is the difference between git pull and git fetch?****

Git pull command pulls new changes or commits from a particular branch from your central repository and updates your target branch in your local repository.

Git fetch is also used for the same purpose but it works in a slightly different way. When you perform a git fetch, it pulls all new commits from the desired branch and stores it in a new branch in your local repository. If you want to reflect these changes in your target branch, git fetch must be followed with a git merge. Your target branch will only be updated after merging the target branch and fetched branch. Just to make it easy for you, remember the equation below:

Git pull = git fetch + git merge

### ****21. What is ‘staging area’ or ‘index’ in Git?****

That before completing the commits, it can be formatted and reviewed in an intermediate area known as ‘Staging Area’ or ‘Index’. From the diagram it is evident that every change is first verified in the staging area I have termed it as “stage file” and then that change is committed to the repository.

### Diagram Description automatically generated

### ****22. What work is restored when the deleted branch is recovered?****

The files which were stashed and saved in the stash index list will be recovered back. Any untracked files will be lost. Also, it is a good idea to always stage and commit your work or stash them.

If you want to fetch the log references of a particular branch or tag then run the command – “git reflog <ref\_name>”.

**23. What is git stash?**

Often, when you’ve been working on part of your project, things are in a messy state and you want to switch branches for some time to work on something else. The problem is, you don’t want to do a commit of half-done work just so you can get back to this point later. The answer to this issue is Git stash.

Stashing takes your working directory that is, your modified tracked files and staged changes and saves it on a stack of unfinished changes that you can reapply at any time.

**24. What is the function of ‘git stash apply’?**

If you want to continue working where you had left your work then ‘git stash apply‘ command is used to bring back the saved changes onto your current working directory.

**25. What is the difference between the ‘git diff ’and ‘git status’?**

‘git diff ’ depicts the changes between commits, commit and working tree, etc. whereas ‘git status’ shows you the difference between the working directory and the index, it is helpful in understanding a git more comprehensively. ‘git diff’ is similar to ‘git status’, the only difference is that it shows the differences between various commits and also between the working directory and index.

**26. What is the difference between ‘git remote’ and ‘git clone’?**

‘git remote add’ creates an entry in your git config that specifies a name for a particular URL whereas ‘git clone’ creates a new git repository by copying an existing one located at the URL

**27. What is git stash drop?**

Git ‘stash drop’ command is used to remove the stashed item. It will remove the last added stash item by default, and it can also remove a specific item if you include it as an argument.

Now give an example.

If you want to remove a particular stash item from the list of stashed items you can use the below commands:

**git stash list:**It will display the list of stashed items like:  
stash@{0}: WIP on master: 049d078 added the index file  
stash@{1}: WIP on master: c264051 Revert “added file\_size”  
stash@{2}: WIP on master: 21d80a5 added number to log

If you want to remove an item named stash@{0} use command **git stash drop stash@{0}**.

**28. How do you find a list of files that have changed in a particular commit?**

For this answer instead of just telling the command, explain what exactly this command will do.

To get a list file that has changed in a particular commit use the below command:

**git diff-tree -r {hash}**

Given the commit hash, this will list all the files that were changed or added in that commit. The -r flag makes the command list individual files, rather than collapsing them into root directory names only.

You can also include the below-mentioned point, although it is totally optional but will help in impressing the interviewer.

The output will also include some extra information, which can be easily suppressed by including two flags:

**git diff-tree --no-commit-id --name-only -r {hash}**

Here –no-commit-id will suppress the commit hashes from appearing in the output, and –name-only will only print the file names, instead of their paths.

**29. What is the function of ‘git config’?**

Git uses your username to associate commits with an identity. The git config command can be used to change your Git configuration, including your username.

Now explain with an example.

Suppose you want to give a username and email id to associate a commit with an identity so that you can know who has made a particular commit. For that I will use:

**git config –global user.name “Your Name”:**This command will add a username.  
**git config –global user.email “Your E-mail Address”:**This command will add an email id.

**30. What does a commit object contain?**

Commit object contains the following components, you should mention all the three points presented below:

* A set of files, representing the state of a project at a given point of time
* Reference to parent commit objects
* An SHA-1 name, a 40 character string that uniquely identifies the commit object

**31. Describe the branching strategies you have used.**

* **Feature branching** – A feature branch model keeps all of the changes for a particular feature inside of a branch. When the feature is fully tested and validated by automated tests, the branch is then merged into master.
* **Task branching** – In this model, each task is implemented on its own branch with the task key included in the branch name. It is easy to see which code implements which task, just look for the task key in the branch name.
* **Release branching** – Once the develop branch has acquired enough features for a release, you can clone that branch to form a Release branch. Creating this branch starts the next release cycle, so no new features can be added after this point, only bug fixes, documentation generation, and other release-oriented tasks should go in this branch. Once it is ready to ship, the release gets merged into master and tagged with a version number. In addition, it should be merged back into the develop branch, which may have progressed since the release was initiated.
* In the end tell them that branching strategies vary from one organization to another so I know basic branching operations like delete, merge, checking out a branch, etc.

**32. Explain the advantages of forking workflow**

* There is a fundamental difference between the forking workflow and other popular git workflows. Rather than using a single server-side to act as the “central” codebase, it gives every developer their own server-side repository. The Forking Workflow is commonly seen in public open-source projects.
* A crucial advantage of the Forking Workflow is that contributions can be integrated without even needing everybody to push to a single central repository that leads to clean project history. Developers can push to their own server-side repositories, but only the project maintainer can push to the official repository.
* If developers are ready to publish a local commit, then they push the commit to their own public repository and not the official one. After this, they go for a pull request with the main repository that lets the project maintainer know an update is ready to be integrated.

**33. How will you know in Git if a branch has already been merged into master?**

The answer is pretty direct.

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

**git branch --merged** – It lists the branches that have been merged into the current branch.  
**git branch --no-merged** – It lists the branches that have not been merged.

**34. Why is it desirable to create an additional commit rather than amending an existing commit?**

There are a couple of reasons for this –

1. The amend operation destroys the state that was previously saved in a commit. If there is just the commit message being changed then that’s not a problem.  But if the contents are being amended then chances of eliminating something important remains more.
2. Abusing “git commit- amend” can result in the growth of a small commit and acquire unrelated changes.

**35. What does ‘hooks’ comprise of in Git?**

This directory consists of shell scripts that are activated if you run the corresponding Git commands.  For example, git will try to execute the post-commit script after you have run a commit.

**36. In Git, how would you return a commit that has just been pushed and made open?**

One or more commits can be reverted through the use of git revert. This command, in a true sense, creates a new commit with patches that cancel out the changes introduced in specific commits. If in case the commit that needs to be reverted has already been published or changing the repository history is not an option then in such cases, git revert can be used to revert commits. If you run the following command then it will revert the last two commits:

git revert HEAD~2..HEAD

Alternatively, there is always an option to check out the state of a particular commit from the past and commit it anew.

### ****37. How to remove a file from git without removing it from your file system?****

One has to be careful during a git add, else you may end up adding files that you didn’t want to commit. However, git rm will remove it from both your staging area (index), as well as your file system (working tree), which may not be what you want.

Instead, use git reset:

git reset filename          # or

echo filename >> .gitingore # add it to .gitignore to avoid re-adding it

This means that git reset <paths> is exactly the opposite of git add <paths>.

### ****38. Can you explain the Gitflow workflow?****

To record the history of the project, Gitflow workflow employs two parallel long-running branches – master and develop:

* Master – this branch is always ready to be released on LIVE, with everything fully tested and approved (production-ready).
* Hotfix – these branches are used to quickly patch production releases. These branches are a lot like release branches and feature branches except they’re based on master instead of develop.
* Develop – this is the branch to which all feature branches are merged and where all tests are performed. Only when everything’s been thoroughly checked and fixed it can be merged to the master.
* Feature – each new feature should reside in its own branch, which can be pushed to the develop branch as their parent one.

### ****39. Tell me the difference between HEAD, working tree and index, in Git.****

* The working tree/working directory/workspace is the directory tree of (source) files that you are able to see and edit.
* The index/staging area is a single, large, binary file in <baseOfRepo>/.git/index, which lists all files in the current branch, their SHA-1 checksums, timestamps, and the file name – it is not another directory which contains a copy of files in it.
* HEAD is used to refer to the last commit in the currently checked-out branch.

### ****40. What is Git fork? What is the difference between fork, branch, and clone?****

* A fork is a copy of a repository. Normally you fork a repository so that you are able to freely experiment with changes without affecting the original project. Most commonly, forks are used to either propose changes to someone else’s project or to use someone else’s project as a starting point for your own idea.
* git cloning means pointing to an existing repository and make a copy of that repository in a new directory, at some other location. The original repository can be located on the local file system or on remote machine accessible supported protocols. The git clone command is used to create a copy of an existing Git repository.
* In very simple words, git branches are individual projects within a git repository. Different branches within a repository can have completely different files and folders, or it could have everything the same except for some lines of code in a file.

### ****41. What are the different ways you can refer to a commit?****

* In Git each commit has a unique hash. These hashes are used to identify the corresponding commits in various scenarios, for example, while trying to checkout a particular state of the code using the git checkout {hash} command.
* Along with this, Git maintains a number of aliases to certain commits, known as refs. Also, every tag that is created in the repository effectively becomes a ref and that is exactly why you can use tags instead of committing hashes in various git commands. Git also maintains a number of special aliases that are changed based on the state of the repository, such as HEAD, FETCH\_HEAD, MERGE\_HEAD, etc.
* In Git, commits are allowed to be referred to as relative to one another. In the case of merge commits, where the commit has two parents, ^ can be used to select one of the two parents, for example, HEAD^2 can be used to follow the second parent.
* And finally, refspecs are used to map local and remote branches together. However, these can also be used to refer to commits that reside on remote branches allowing one to control and manipulate them from a local git environment.

### ****42. What is the difference between rebasing and merge in Git?****

* In Git, the rebase command is used to integrate changes from one branch into another. It is an alternative to the “merge” command. The difference between rebasing and merge is that rebase rewrites the commit history in order to produce a straight, linear succession of commits.
* Merging is Git’s way of putting a forked history back together again. The git merge command helps you take the independent lines of development created by git branch and integrate them into a single branch.

### ****43. Explain the difference between reverting and resetting.****

* Git reset is a powerful command that is used to undo local changes to the state of a Git repository. Git reset operates on “The Three Trees of Git” which are, Commit History ( HEAD ), the Staging Index, and the Working Directory.
* Revert command in Git creates a new commit that undoes the changes from the previous commit. This command adds a new history to the project. It does not modify the existing history.

### ****44. What is git cherry-pick?****

The command git cherry-pick is normally used to introduce particular commits from one branch within a repository onto a different branch. Another common use is to forward- or back-port commits from a maintenance branch to a development branch. This is in contrast with other ways such as merge and rebase which normally apply many commits onto another branch.

Consider:

git cherry-pick <commit-hash>

### ****45. How do you find a list of files that have changed in a particular commit?****

git diff-tree -r {hash}

Given the commit hash, this will list all the files that were changed or added in that commit. The -r flag makes the command list individual files, rather than collapsing them into root directory names only.

The output will also include some extra information, which can be easily suppressed by including a couple of flags:

git diff-tree --no-commit-id --name-only -r {hash}

Here –no-commit-id will suppress the commit hashes from appearing in the output, and –name-only will only print the file names, instead of their paths.

## **Advanced level Questions**

### ****46. How do you squash the last N commits into a single commit?****

### ****There are two options to squash the last N commits into a single commit include both of the below-mentioned options in your answer****

If you want to write the new commit message from scratch use the following command  
**git reset –soft HEAD~N &&git commit**

If you want to start editing the new commit message with a concatenation of the existing commit messages then you need to extract those messages and pass them to Git commit for that I will use  
**git reset –soft HEAD~N &&git commit –edit -m”$(git log –format=%B –reverse**[**.HEAD@{N}**](mailto:HEAD..HEAD@%7b1%7d)**)”**

### ****47. What is Git bisect? How can you use it to determine the source of a (regression) bug?****

* Git bisect is used to find the commit that introduced a bug by using binary search. The command for Git bisect is  
  **git bisect <subcommand> <options>**
* Now since you have mentioned the command above explain to them what this command will do.
* This command uses a binary search algorithm to find which commit in your project’s history introduced a bug. You use it by first telling it a “bad” commit that is known to contain the bug, and a “good” commit that is known to be before the bug was introduced. Then Git bisect picks a commit between those two endpoints and asks you whether the selected commit is “good” or “bad”. It continues narrowing down the range until it finds the exact commit that introduced the change.

### ****48. How do you configure a Git repository to run code sanity checking tools right before making commits, and preventing them if the test fails?****

I will suggest you to first give a small introduction to sanity checking.

Sanity or smoke testdetermines whether it is possible and reasonable to continue testing.

Now explain how to achieve this.

This can be done with a simple script related to the pre-commit hook of the repository. The pre-commit hook is triggered right before a commit is made, even before you are required to enter a commit message. In this script, one can run other tools, such as linters and perform sanity checks on the changes being committed into the repository.

Finally, give an example, you can refer the below script:

**#!/bin/sh**  
**files=$(git diff –cached –name-only –diff-filter=ACM | grep ‘.go$’)**  
**if [ -z files ]; then**  
**exit 0**  
**fi**  
**unfmtd=$(gofmt -l $files)**  
**if [ -z unfmtd ]; then**  
**exit 0**  
**fi**  
**echo “Some .go files are not fmt’d”**  
**exit 1**

This script checks to see if any .go file that is about to be committed needs to be passed through the standard Go source code formatting tool gofmt. By exiting with a non-zero status, the script effectively prevents the commit from being applied to the repository.

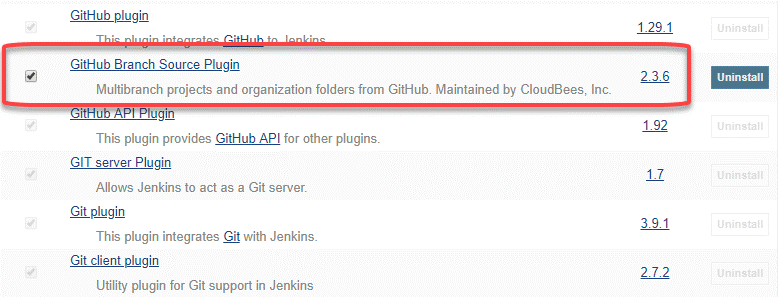
### ****49. How do you integrate Git with Jenkins?****

**Step 1**. Click on the manage jenkins button on your jenkins dashboard.

**Step 2**. Click on manage jenkins plugin.

**Step 3:**In the Plugins Page

1. Select the GIT Plugin
2. Click on **Install without restart.**The plugin will take a few moments to finish downloading depending on your internet connection, and will be installed automatically.
3. You can also select the option **Download now and Install after restart** In which plugin is installed after restart
4. You will be shown a “No updates available” message if you already have the Git plugin installed.

**Step 4**: Once the plugins have been installed, go to **Manage Jenkins** on your Jenkins dashboard. You will see your plugins listed among the rest.

**50. What is git reflog?**

The ‘reflog’ command keeps a **track of** **every single change made in the references**(branches or tags) of a repository and keeps a log history of the branches and tags that were either created locally or checked out. Reference logs such as the commit snapshot of when the branch was created or cloned, checked-out, renamed, or any commits made on the branch are maintained by [Git](https://www.edureka.co/blog/what-is-git/) and listed by the ‘reflog’ command.

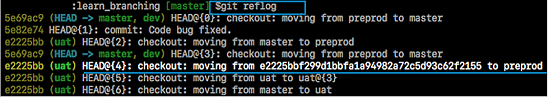
***Note:****The branch will be recoverable from your working directory only if the branch ever existed in your local repository i.e. the branch was either created locally or checked-out from a remote repository in your local repository for Git to store its reference history logs.*

This command must be executed in the repository that had the lost branch. If you consider the remote repository situation, then you have to execute the reflog command on the developer’s machine who had the branch.

**command:** git reflog

### ****51. How to recover a deleted branch using git reflog?****

**Step 1**: **History logs of all the references**

Get a list of all the local recorded history logs for all the references (‘master’, ‘uat’ and ‘prepod’) in this repository.

**Step 2: Identify the history stamp**

As you can see from the above snapshot, the highlighted commit id: e2225bb along with the HEAD pointer index:4 is the one when ‘preprod’ branch was created from the current HEAD pointer pointing to your latest work.

**Step 3**: **Recover**

If you want to recover back the ‘preprod‘ branch then use the command  ‘git checkout’ passing the HEAD pointer reference with the index id – 4. This is the pointer reference when ‘preprod’ branch was created long commit id highlighted in the output screenshot.