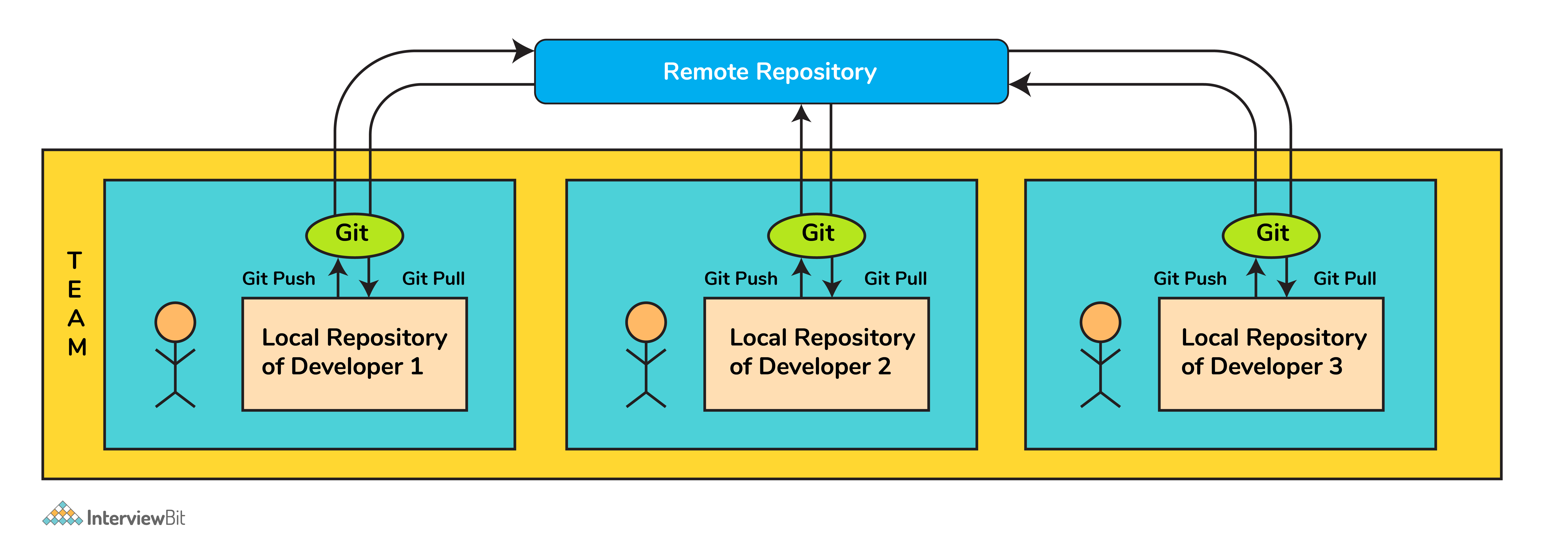
**What is Git and why is it used?**

* Git is the most popular, open-source, widely used, and an example of distributed version control system (DVCS) used for handling the development of small and large projects in a more efficient and neat manner.
* It is most suitable when there are multiple people working on projects as a team and is used for tracking the project changes and efficiently supports the collaboration of the development process.
* With the help of the versioning system, the developer can identify who has made what changes and then run tests and fix bugs if any and then do necessary feature implementation. In case of any unforeseen circumstances, the code can be reverted to any of the previously working versions thereby saving huge efforts.



[**GIT Cheat Sheet: Basic to Advanced Concepts**](https://www.interviewbit.com/git-cheat-sheet/)

**Scope of Git:**

* Due to a well-established version control system and the support for collaborative work, git has garnered wide popularity not just amongst the software developers, but also among the people who do other tasks like documentation or any other collaborative work. It can seem challenging at first, but once we get the hang of git, we find that it makes our lives much simpler.
* It has an amazing branching system that supports nonlinear development along with keeping the developers accountable for their code. This helps in making the development process efficient and faster.

### 1. What is a version control system (VCS)?

A VCS keeps track of the contributions of the developers working as a team on the projects. They maintain the history of code changes done and with project evolution, it gives an upper hand to the developers to introduce new code, fixes bugs, and run tests with confidence that their previously working copy could be restored at any moment in case things go wrong.

### 2. What is a git repository?

A repository is a file structure where git stores all the project-based files. Git can either stores the files on the local or the remote repository.

### 3. What does git clone do?

The command creates a copy (or clone) of an existing git repository. Generally, it is used to get a copy of the remote repository to the local repository.

**4. What does the command git config do?**

The git config command is a convenient way to set configuration options for defining the behavior of the repository, user information and preferences, git installation-based configurations, and many such things.   
  
For example:  
To set up your name and email address before using git commands, we can run the below commands:

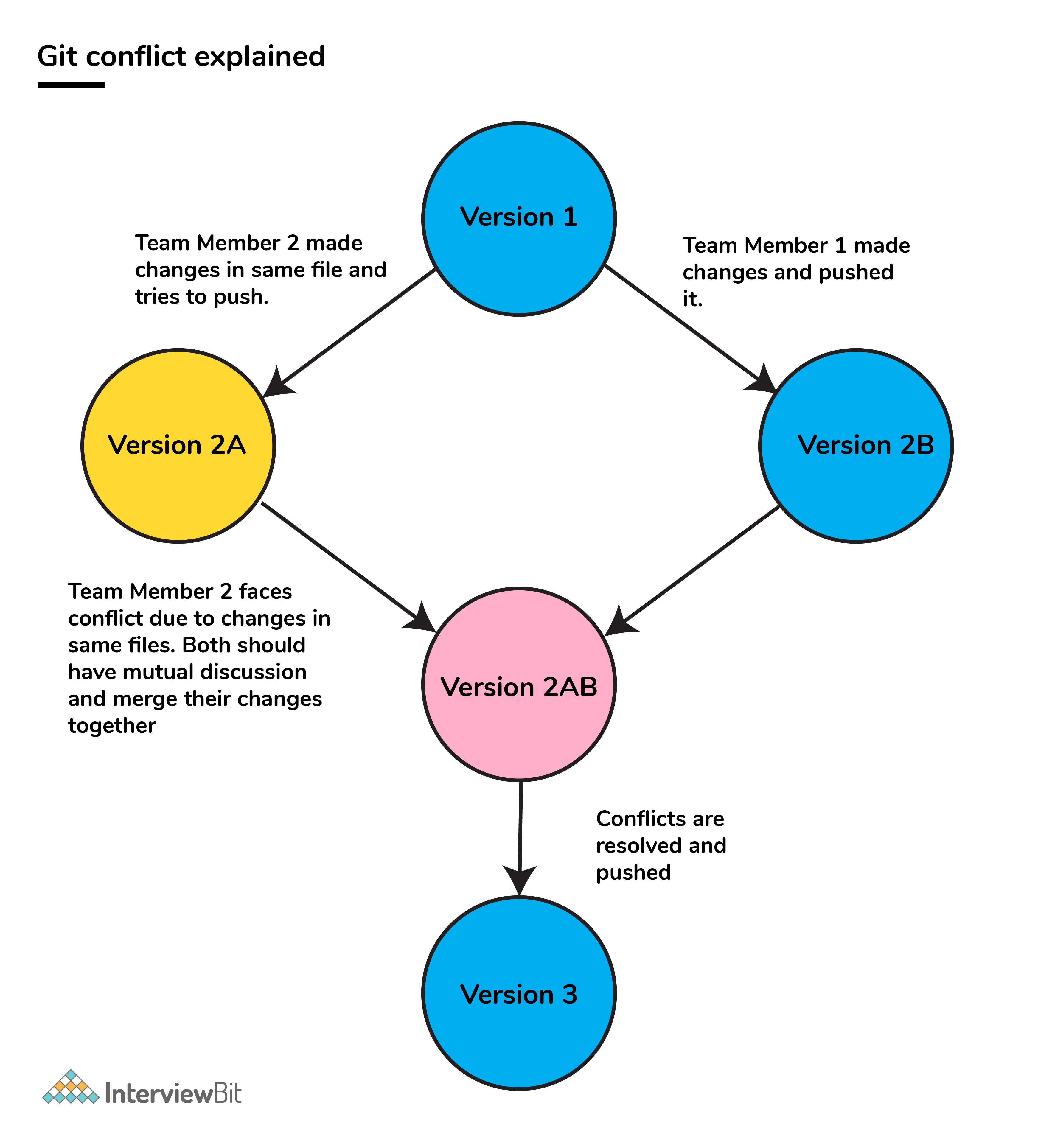
* git config --global  
  user.name  
  “<<your\_name>>”
* git config --global user.email “<<your\_email>>”

**5. Can you explain head in terms of git and also tell the number of heads that can be present in a repository?**

* A head is nothing but a reference to the last commit object of a branch.
* For every repository, there will always be a default head referred to as “master” or now “main” (as per GitHub) but there is no restriction to the count of heads available. In other words, it can have any number of heads.
* **Usages:**  
    
  - To go or checkout to 1 commit before the latest commit, we use git checkout HEAD~1  
    
  - To uncommit the last 3 commits without losing the changes, we first run git reset HEAD~3. Then we can see the changes made in the last 3 commits and then update it manually and commit it finally.  
    
  - In order to uncommit the last 3 commits and also remove the changes, we can run the command: git reset --hard HEAD~3. This command will completely remove all the changes.  
    
  - To look into the changes made in the last 3 commits, we can run git diff HEAD~3  
    
  - To make a new commit by reverting the last 3 commits, we can run the command: git revert --no-commit HEAD~3...HEAD

**6. What is a conflict?**

* Git usually handles feature merges automatically but sometimes while working in a team environment, there might be cases of conflicts such as:  
    
  1. When two separate branches have changes to the same line in a file  
  2. A file is deleted in one branch but has been modified in the other.
* These conflicts have to be solved manually after discussion with the team as git will not be able to predict what and whose changes have to be given precedence.



**7. What is the functionality of git ls-tree?**

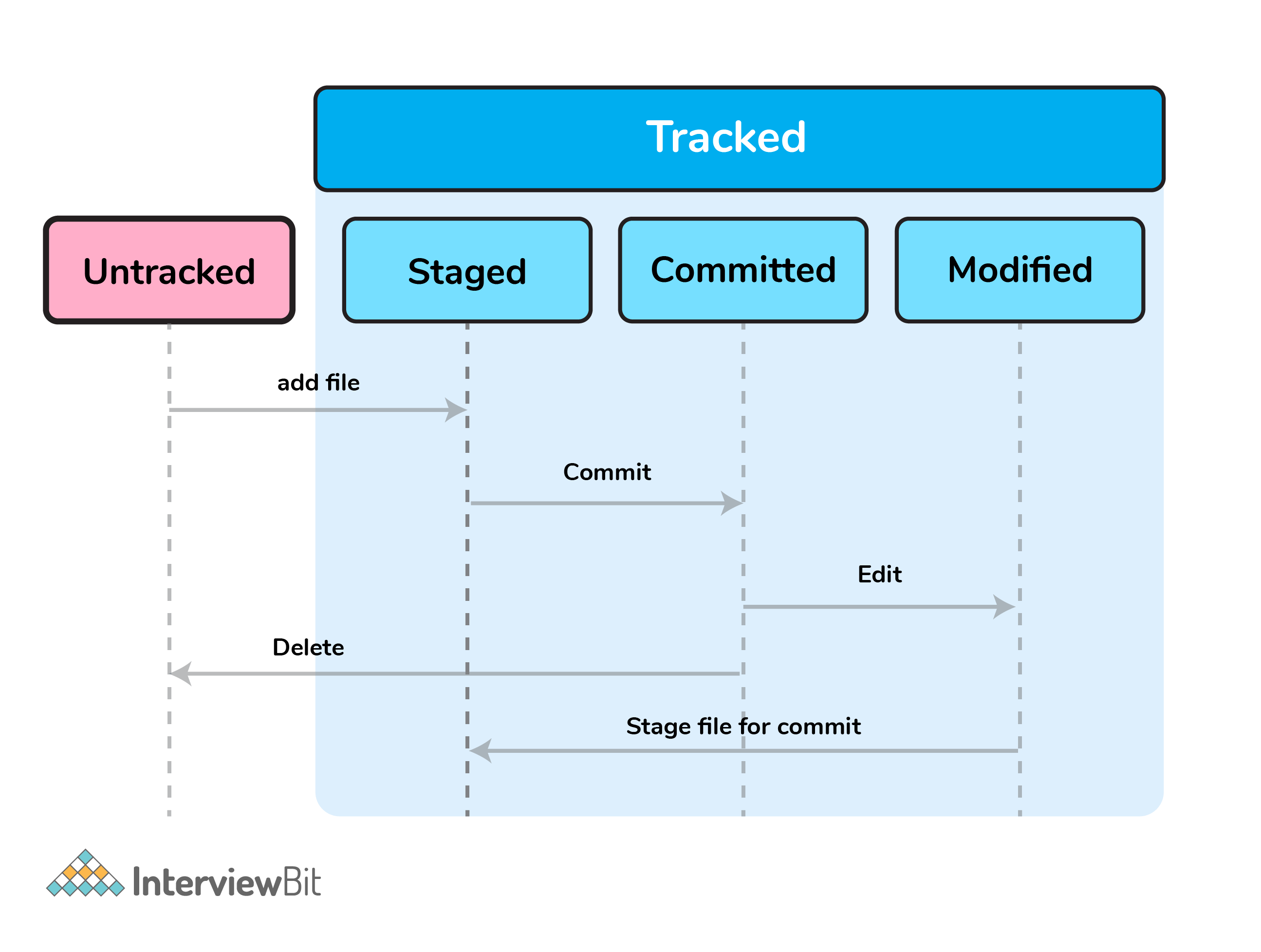
This command returns a tree object representation of the current repository along with the mode and the name of each item and the SHA-1 value of the blob.

**8. What does git status command do?**

git status command is used for showing the difference between the working directory and the index which is helpful for understanding git in-depth and also keep track of the tracked and non-tracked changes.

**9. Define “Index”.**

Before making commits to the changes done, the developer is given provision to format and review the files and make innovations to them. All these are done in the common area which is known as ‘Index’ or ‘Staging Area’.



In the above image, the “staged” status indicates the staging area and provides an opportunity for the people to evaluate changes before committing them.

**10. What does git add command do?**

* This command adds files and changes to the index of the existing directory.
* You can add all changes at once using git add . command.
* You can add files one by one specifically using git add <file\_name> command.
* You can add contents of a particular folder by using git add /<folder\_name>/ command.

**Intermediate GIT Interview Questions**

**11. Why is it considered to be easy to work on Git?**

With the help of git, developers have gained many advantages in terms of performing the development process faster and in a more efficient manner. Some of the main features of git which has made it easier to work are:

* **Branching Capabilities:**  
    
  - Due to its sophisticated branching capabilities, developers can easily work on multiple branches for the different features of the project.  
  - It also has an easier merge option along with an efficient work-flow feature diagram for tracking it.
* **Distributed manner of development:**  
    
  - Git is a distributed system and due to this nature, it became easier to trace and locate data if it's lost from the main server.  
  - In this system, the developer gets a repository file that is present on the server. Along with this file, a copy of this is also stored in the developer’s system which is called a local repository.  
  - Due to this, the scalability of the project gets drastically improved.
* **Pull requests feature:**  
    
  - This feature helps in easier interaction amongst the developers of a team to coordinate merge-operations.  
  - It keeps a proper track of the changes done by developers to the code.
* **Effective release cycle:**  
    
  - Due to the presence of a wide variety of features, git helps to increase the speed of the release cycle and helps to improve the project workflow in an efficient manner.

**12. How will you create a git repository?**

* Have git installed in your system.
* Then in order to create a git repository, create a folder for the project and then run git init.
* Doing this will create a .git file in the project folder which indicates that the repository has been created.

**13. Tell me something about git stash?**

Git stash can be used in cases where we need to switch in between branches and at the same time not wanting to lose edits in the current branch. Running the git stash command basically pushes the current working directory state and index to the stack for future use and thereby providing a clean working directory for other tasks.

**14. What is the command used to delete a branch?**

* To delete a branch we can simply use the command git branch –d [head].
* To delete a branch locally, we can simply run the command: git branch -d <local\_branch\_name>
* To delete a branch remotely, run the command: git push origin --delete <remote\_branch\_name>
* Deleting a branching scenario occurs for multiple reasons. One such reason is to get rid of the feature branches once it has been merged into the development branch.

**15. What differentiates between the commands git remote and git clone?**

git remote command creates an entry in  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.

**16. What does git stash apply command do?**

* git stash apply command is used for bringing the works back to the working directory from the stack where the changes were stashed using git stash command.
* This helps the developers to resume their work where they had last left their work before switching to other branches.

**17. Differentiate between git pull and git fetch.**

| **git pull** | **git fetch** |
| --- | --- |
| This command pulls new changes from the currently working branch located in the remote central repository. | This command is also used for a similar purpose but it follows a two step process:  1. Pulls all commits and changes from desired branch and stores them in a new branch of the local repository.  current 2. For changes to be reflected in the current / target branch, git fetch should be followed by git merge command. |

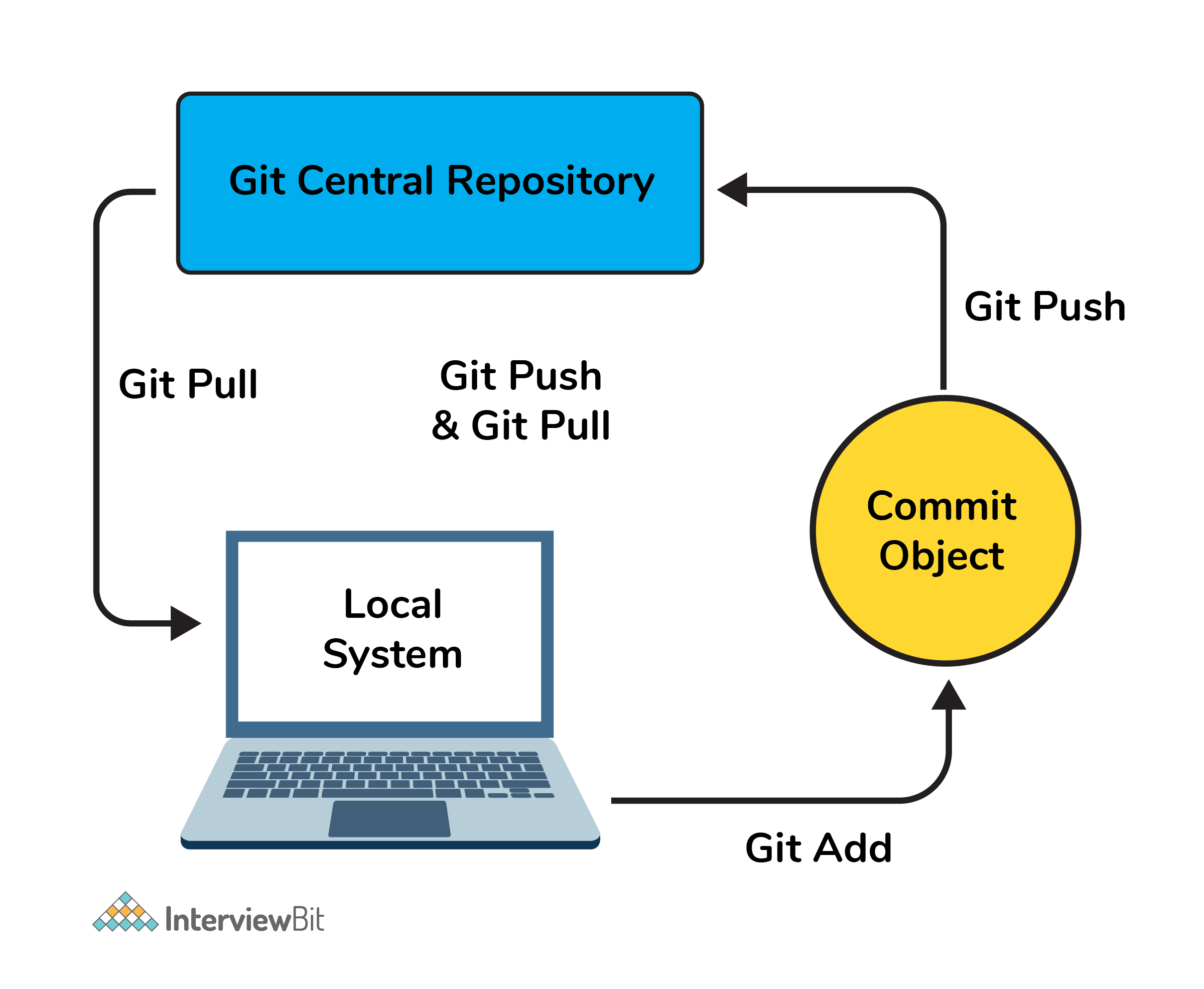
git pull = git fetch + git merge

**18. Can you give differences between “pull request” and “branch”?**

| **pull request** | **branch** |
| --- | --- |
| This process is done when there is a need to put a developer’s change into another person’s code branch. | A branch is nothing but a separate version of the code. |

**19. Why do we not call git “pull request” as “push request”?**

* “Push request” is termed so because it is done when the target repository requests us to push our changes to it.
* “Pull request” is named as such due to the fact that the repo requests the target repository to grab (or pull) the changes from it.



**20. Can you tell the difference between Git and GitHub?**

| **Git** | **GitHub** |
| --- | --- |
| This is a distributed version control system **installed on local machines** which allow developers to keep track of commit histories and supports collaborative work. | This is a **cloud-based source code repository** developed by using git. |
| This is maintained by “The Linux Foundation”. | This was acquired by “Microsoft” |
| SVN, Mercurial, etc are the competitors | GitLab, Atlassian BitBucket, etc are the competitors. |

* GitHub provides a variety of services like forking, user management, etc along with providing a central repository for collaborative work.

**21. What do the git diff and git status commands do?**

| **git diff** | **git status** |
| --- | --- |
| This shows the changes **between commits, working trees,** etc. | This shows the difference **between the working directory and index** that is essential in understanding git in depth. |

* git diff works in a similar fashion to git status with the only difference of showing the differences between commits and also between the working directory and index.

**22. What has to be run to squash multiple commits (last N) into a single commit?**

Squashing multiple commits to a single one overwrites the history which is why it is recommended to be done using full caution. This step can be done by running the command: git rebase -i HEAD~{{N}} where {{N}} represents the number of commits needed to be squashed.

**23. How would you recover a branch that has already pushed changes in the central repository but has been accidentally deleted from every team member’s local machines?**

We can recover this by checking out the latest commit of this branch in the reflog and then checking it out as a new branch.

**24. Can you tell something about git reflog?**

This command tracks every single change made in the repository references (that can be branches or tags) and also maintains the branches/tags log history that was 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 and listed by the ‘reflog’ command.

* This recovery of the branch is only possible when 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 should be executed in the repository that had the lost branch.

**25. What consists of a commit object?**

A commit object consists of the following components:

* A set of files that represents the state of a project at a given point in time.
* Reference to parent commit objects.
* A 40 character string termed as SHA-1 name uniquely identifies the commit object.

**26. Explain the levels in git config and how can you configure values using them?**

* In order to make git work, it uses a set of configurations that are pre-defined by default by means of configuration files (or config files). We can change the default behavior of git by just modifying these files which are basically text files. In order to do this, it is important to understand how git identifies these files. It does so by following the below steps:  
    
  - Firstly, git searches for the config values in the system-wide gitconfig file stored in <<installation\_path>>/etc/gitconfig file that has settings defined and applied to **every user** of the system and all their repos.  
       - In case you want git to search from this particular file and read/write on it, we can pass the option --system to git config command.  
    
  - Next, git searches for the ~/.gitconfig file or ~/.config/git/config that has the scope specific to the user.  
     - Git can be made to read/ write from this file specifically bypassing --global to the git config command.  
    
  - Lastly, git searches for the config values in the git directory of the local repository that we are currently working on.  
     - These config values are specific to that particular repository alone and can be accessed by passing --local to the git config command.This is the default config file that gets accessed and modified upon in case we do not specify any levels.

**27. What is a detached HEAD and what causes this and how to avoid this?**

Detached HEAD indicates that the currently checked-out repository is not a local branch. This can be caused by the following scenarios:

* When a branch is a read-only branch and we try to create a commit to that branch, then the commits can be termed as “free-floating” commits not connected to any branch. They would be in a detached state.
* When we checkout a tag or a specific commit and then we try to perform a new commit, then again the commits would not be connected to any branch. When we now try to checkout a branch, these new commits would be automatically placed at the top.  
    
  In order to ensure that detached state doesn't happen, =instead of checking out commit/tag, we can create a branch emanating from that commit and then we can switch to that newly created branch by using the command: git checkout -b <<new\_branch\_name>>. This ensures that a new branch is checkout out and not a commit/tag thereby ensuring that a detached state wouldn't happen.

**28. What does git annotate command do?**

* This command annotates each line within the given file with information from the commit which introduced that change. This command can also optionally annotate from a given revision.
* Syntax: git annotate [<options>] <file> [<revision>]
* You can get to learn more about this command from the official git documentation [here](https://git-scm.com/docs/git-annotate).

**29. What is the difference between git stash apply vs git stash pop command?**

* git stash pop command throws away the specified stash (topmost stash by default) after applying it.
* git stash apply command leaves the stash in the stash list for future reuse. In case we wanted to remove it from the list, we can use the git stash drop command.

git stash pop = git stash apply + git stash drop

**Advanced GIT Interview Questions**

**30. What command helps us know the list of branches merged to master?**

* git branch --merged helps to get the list of the branches that have been merged into the current branch.
* Note: git branch --no-merged lists the branches that have not been merged to the current branch.

**31. How will you resolve conflict in Git?**

* Conflicts occur whenever there are multiple people working on the same file across multiple branches. In such cases, git won't be able to resolve it automatically as it is not capable of deciding what changes has to get the precedence.
* Following are the steps are done in order to resolve git conflicts:  
  1. Identify the files that have conflicts.  
  2. Discuss with members who have worked on the file and ensure that the required changes are done in the file.  
  3. Add these files to the staged section by using the git add command.  
  4. Commit these changes using the git commit command.  
  5. Finally, push the changes to the branch using the git.

**32. What is best advisable step in cases of broken commit: Create an additional commit OR amend an existing commit?**

* It is always advisable to create an additional commit rather than amending the existing commit due to the following reasons:  
  - Doing the amend operation destroys the previously saved state of that commit. If only the commit message gets changes or destroyed, it's acceptable but there might be cases when the contents of the commits get amended. This results in the loss of important information associated with the commit.  
  - Over usage of git commit --amend can have severe repercussions as the small commit amend can continue to grow and gather unrelated changes over time.

**33. How to revert a bad commit which is already pushed?**

There can be cases where we want to revert from the pushed changes and go back to the previous version. To handle this, there are two possible approaches based on the situations:

* **Approach 1**: Fix the bad changes of the files and create a new commit and push to the remote repository. This step is the simplest and most recommended approach to fix bad changes. You can use the command: git commit -m "<message>"
* **Approach 2**: New commit can be created that reverts changes done in the bad commit. It can be done using git revert <name of bad commit>

**34. What is the functionality of “git cherry-pick” command?**

This command is used to introduce certain commits from one branch onto another branch within the repository. The most common use case is when we want to forward- or back-port commits from the maintenance branch to the development branch.

**35. Explain steps involved in removing a file from git index without removing from the local file system?**

* Sometimes we end up having certain files that are not needed in the git index when we are not being careful while using the git add command. Using the command git rm will remove the file from both the index and the local working tree which is not always desirable.
* Instead of using the git rm command we can use the git reset command for removing the file from the staged version and then adding that file to the .gitignore file to avoid repeating the same mistake again.

git reset <file\_name> # remove file from index

echo filename >> .gitingore # add file to .gitignore to avoid mistake repetition.

**36. What are the factors involved in considering which command to choose among: git merge and git rebase?**

Both these commands ensure that changes from one branch are integrated into another branch but in very different ways. Git rebasing can be thought of as saying to use another branch as a new base for the work.

* Whenever in doubt, it is always preferred to use the git merge command.  
    
  Following are some factors that tell when to use merge and rebase commands:
* In case our branch gets contributions from other developers outside the team as in open-source or public repositories, then rebase is not preferred.  
  - This is because rebase destroys the branch and it results in broken and inconsistent repositories unless the git pull --rebase command is used.
* Rebase is a very destructive operation. If not applied correctly, it results in loss of committed work which might result in breaking the consistency of other developer’s contribution to the repository.
* If the model of having branches per feature is followed, rebasing is not a good idea there because it keeps track of related commits done by the developers. But in case the team follows having branches per developer of the team, then the branch has no additional useful information to be conveyed. In this model, rebasing has no harm and can be used.
* If there is any chance where there might be a necessity to revert a commit to previous commits, then reverting a rebase would be almost impossible as the commit data would be destroyed. In such cases, the merge can be used.

**37. How do you find a commit which broke something after a merge operation?**

* This can be a time-consuming process if we are not sure what to look at exactly. Fortunately, git provides a great search facility that works on the principle of binary search as git-bisect command.
* The initial set up is as follows:

git bisect start # initiates bisecting session

git bisect bad # marks current revision as bad

git bisect good revision # marks last known commit as good revision

* Upon running the above commands, git checks out a revision that is labeled as halfway between “good” and “bad” versions. This step can be run again by marking the commit as “good” or “bad” and the process continues until the commit which has a bug is found.

**38. What are the functionalities of git reset --mixed and git merge --abort?**

* git reset --mixed command is used for undoing changes of the working directory and the git index.
* git merge --abort command is used for stopping the merge process and returning back to the state before the merging occurred.

**39. Can you tell the differences between git revert and git reset?**

| **git revert** | **git reset** |
| --- | --- |
| This command is used for creating a new commit that undoes the changes of the previous commit. | This command is used for undoing the local changes done in the git repository |
| Using this command adds a new history to the project without modifying the existing history | This command operates on the commit history, git index, and the working directory. |

**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 |

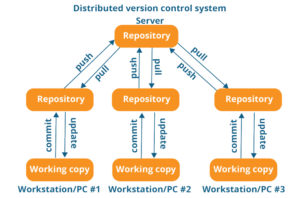
**2. What is Git?**

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?**

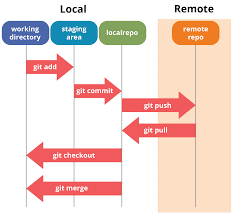
Git pull command pulls new changes or commits from a particular branch from your central repository and updates your target branch in your local repository.

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.

****

**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.

[[](https://www.edureka.co/devops-certification-training)](https://www.edureka.co/devops-certification-training" \t "_blank)

**[DevOps Certification Training Course](https://www.edureka.co/devops-certification-training" \t "_blank)**

[Explore Curriculum](https://www.edureka.co/devops-certification-training" \t "_blank)

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

**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.

**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 |

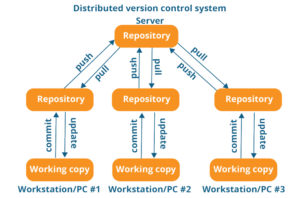
**2. What is Git?**

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?**

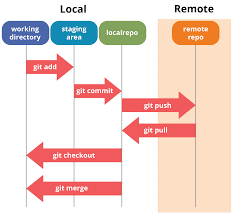
Git pull command pulls new changes or commits from a particular branch from your central repository and updates your target branch in your local repository.

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.

****

**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.

[[](https://www.edureka.co/devops-certification-training)](https://www.edureka.co/devops-certification-training" \t "_blank)

**[DevOps Certification Training Course](https://www.edureka.co/devops-certification-training" \t "_blank)**

[Explore Curriculum](https://www.edureka.co/devops-certification-training" \t "_blank)

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

### -------------------------------------------------------------------------------------

### 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’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 |

4. 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.



5. 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.



6. How is Git different from Subversion (SVN)?

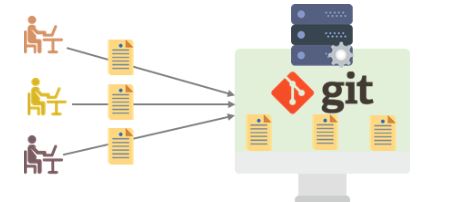
|  |  |
| --- | --- |
| GIT | SVN |
| Git is a distributed decentralized version control system | SVN is a centralized version control system. |
| Git stores content in the form of metadata. | SVN stored data in the form of files. |
| The master contains the latest stable release. | In SVN, the trunk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

7. 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

8. 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)



9. 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.

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

 git commit -m "x files created"

11. 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.

12. 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

13.  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.



14. 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 |

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

hosting. All the three are services for hosting Git repositories

16. 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.



Free Course: Introduction to DevOps Tools

Master the Fundamentals of DevOps Tools[ENROLL NOW](https://www.simplilearn.com/devops-tools-free-course-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-tutorial-skillup-course-promotion)



17. 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.

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

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

19. 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>

20. 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. |

21. 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

22. 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.

23. 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.

24. 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.

25. 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>

26. 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.

27. 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.

28. What is the functionality of git clean command?

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

29. 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.

30. 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.

Now let’s raise the level of difficulty with advanced Git interview questions and answers.

Interested to learn more about Git? Check out the DevOps Engineer Master's Program and get certified today.

Advanced Git Interview Questions

31. 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

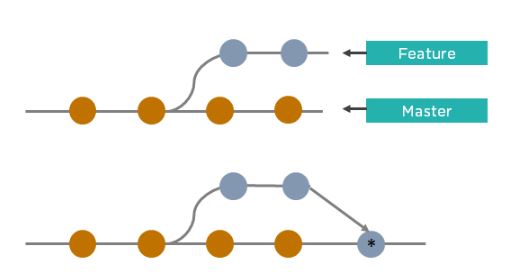
The failure during the merge process indicates that there’s a conflict between the local branch and the branch being merged. In this case, Git resolves as much as possible, but some things have to be fixed manually in the conflicted files.

33. 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. |

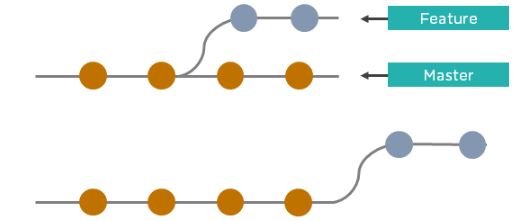
34. How is Git merge different from Git rebase?

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

35. 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.

36. 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.

### 37. 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}.

### 38. 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. |

### 39. 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.

### 40. 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.

-------------------------------------------------------------------------------------------------------------

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:

Skip Ad

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

**1. What are Git and GitHub?**

Git is an open-source and free distributed version control system developed to handle projects of all sizes quickly and efficiently.

GitHub uses Git to provide Internet hosting for version control and software development. It offers the functionality of distributed version control and source code management, which is found in Git, in addition to other unique features.

***Check out our blog to learn more about***[***Commands in Git***](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/)***!***

**2. What is the difference between Git and GitHub?**

Git is a version control system that is used in the management of the source code history. GitHub, on the other hand, is a cloud-based hosting service that is used in the management of Git repositories. GitHub is designed to help in the better management of open-source projects.

**3. What is a Git repository? Name some popular Git hosting services.**

Repositories contain a batch of files that are different versions of a project. These files are imported from the repositories into the local servers of users for further modifications and updates in the content.

A few popular Git hosting services are:

* GitHub
* GitLab
* Bitbucket
* SourceForge

***The***[***GIT Cheat Sheet***](https://intellipaat.com/blog/tutorial/devops-tutorial/git-cheat-sheet/)***by Intellipaat comes in handy while studying GIT.***

**4. What is a version control system? Mention its types.**

A version control system (VCS) is a software tool used to create different project versions and store them in a repository. All modifications to the code are recorded and tracked by the VCS.

**Types of version control systems:**

* **Local version control systems** have a database that maintains all the file changes on disk under revision control in a special format.
* **Centralized version control systems** contain one repository, and each user gets their own working copy.
* **Distributed version control systems** contain multiple repositories, each accessible to separate users with their own working copy.

**5. What are the main differences between Git and SVN?**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Git** | **SVN** |
| Type of version control | Distributed | Centralized |
| Access to networks | Not mandatory | Mandatory |
| Global revision number | Not available | Available |
| Content | Cryptographic SHA-1 Hash | No hashed content |

Here we list some of the most important differences between Git and SVN:

* When it comes to handling large files, Git is not preferred but SVN can handle multiple projects in the same repository
* Git does not have ‘commits’ across multiple branches but SVN lets you create the folders on any location in the repository layout
* You cannot commit changes in Git but SVN lets you create a tag as a branch and you can create multiple revisions under a root tag

**6. What are the advantages of using GIT?**

Here are some of the most important advantages of Git:

* Data redundancy and data replication is possible
* It is a highly available service
* For one repository we can have only one directory of Git
* The network performance and disk utilization are excellent
* It is very easy to collaborate on any project
* We can work on any sort of project within the Git

**7. What language is used in GIT?**

C is the programming language that is used for creating Git which ensures that the overheads are reduced.

**8. What are the advantages of Git over SVN?**

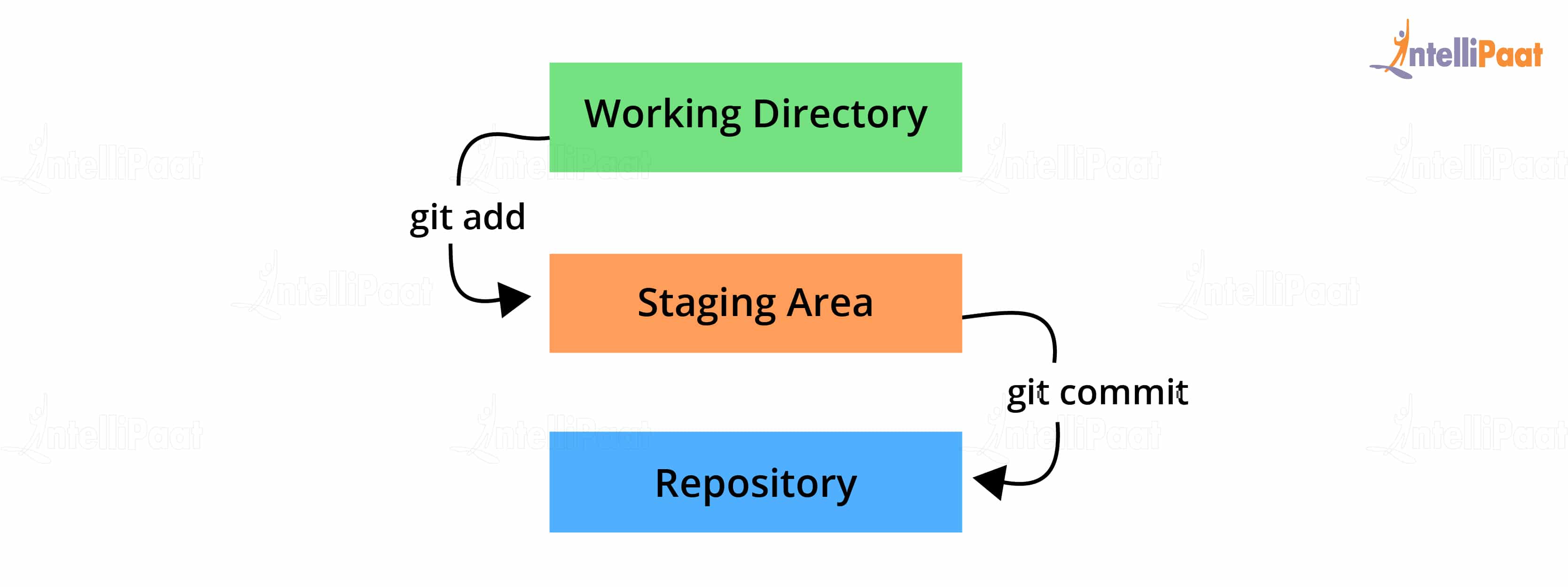
Since Git is an open source version control system it lets us run multiple versions of our project so that it shows the changes that are made to the code over time and if needed we can keep track of the changes that we have made. This means that a large number of developers can make their own changes and upload those changes so that the changes can be attributed to the particular developers.

**9. What is Git Bash?**

Git Bash is an application that installs Bash, Git, and a few Bash utilities that are commonly used on a Windows OS. In Git Bash, interaction is possible with Git elements and the repository through different commands.

**10. What is the meaning of “Index” or “Staging Area” in GIT?**

When we are making the commits, we can make changes to it, format it and review it in the intermediate area known as ‘Staging Area’ or ‘Index’.



### ****11. What is tagging in Git?****

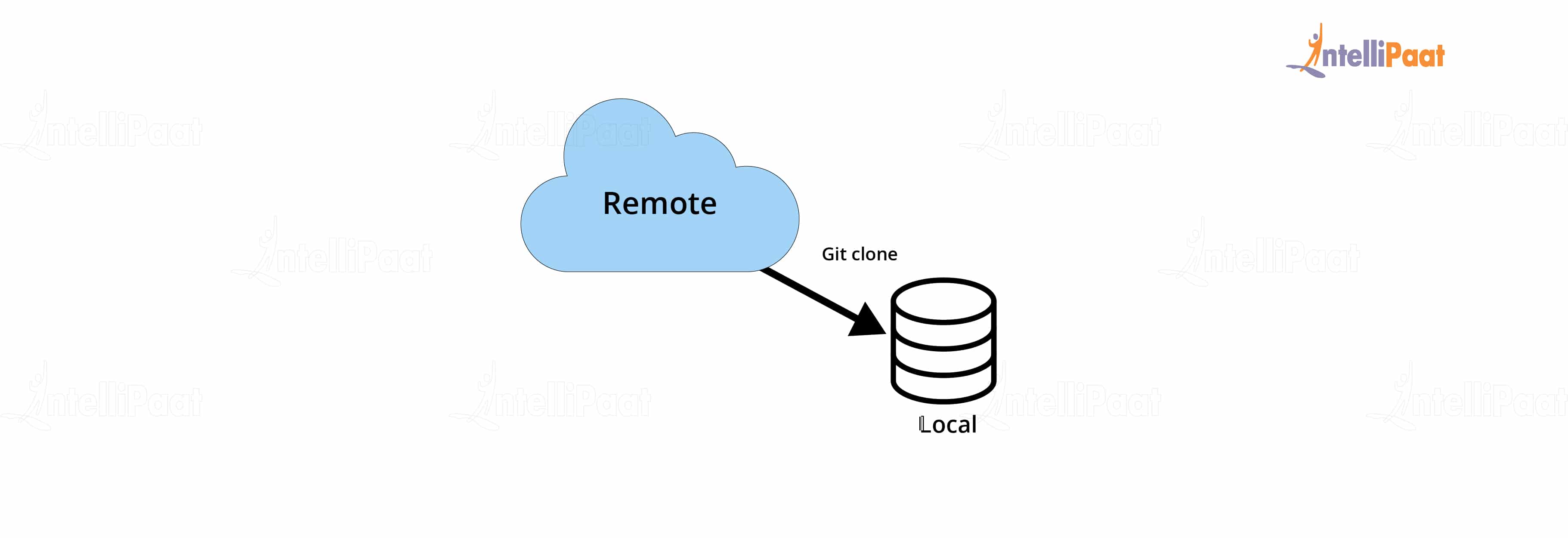
Tagging allows developers to mark all the important checkpoints through the course of their projects’ progress. Instead of commit IDs, tag names can be used while commits are checked out and pushed to a remote repo.

### ****12. What is forking in Git?****

A repository copy is called a fork. So, forking allows one to experiment with changes without worrying about the original project. This process is ideal for proposing changes to someone else’s projects.

### ****13. What is the use of a Git clone?****

The Git clone command lets us copy the existing Git repository. If we want to get a copy of the central repository then the best way to do it is using ‘cloning’.



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

The ‘Git config’ is a great way to configure our options for the Git installation. Using this command, we can define the repository behavior, preferences and user information.

### 15. What is the process for creating a repository in Git?

If we want to create a repository in Git, then we need to run the command “git init”. With this command .git repository, we can create a directory in the project directory.

**16. What is cherry-pick in Git?**

Git cherry-pick is a command that allows the picking of arbitrary Git commits by reference and adding them to the HEAD. Cherry-picking is the process of picking a commit from one branch and applying it to another. It helps in undoing changes.

**17. What is origin in Git?**

Origin refers to the remote repository that a project was originally cloned from and is used instead of the original repository’s URL. This allows for easier referencing.

**18. What is the git push command?**

The git push command is applied for uploading content to a remote repository from a local repository. Pushing can overwrite changes, so it should be used with caution.

**19. What is the git pull command?**

The git pull command is for fetching and downloading content from a remote repository and integrating it with a local repository.

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

Git fetch retrieves new data from a remote repository but does not integrate it into our working files. It helps in checking if any changes happened in the remote repository. It does not manipulate or destroy anything in the process.

Git pull, on the other hand, updates the HEAD with the latest changes from the remote server and directly integrates it into the working copy files. Using git pull can end in merge conflict as it tries to merge remote changes with the local ones.

**21. Explain git checkout in Git.**

Git checkout allows for the switching of the HEAD. It can be used to restore the historic versions of files as well. The command operates upon files, commits, and branches.

**22. What does git rebase do?**

Rebasing is the reapplying of commits on top of another base trip. A sequence of commits is applied from distinct branches into the final commit. It is a linear process of merging and an alternative to the git merge command. Rebasing makes it seem like one has created a branch from a different commit.

**23. What is the difference between git rebase and git merge?**

In git rebase, a feature branch is moved into a master. Git merge maintains the history by adding a new commit.

**24. What is revert in Git?**

The git revert command is a forward-moving undo operation. It is a safe way to undo changes as it will create a new commit that inverses the changes instead of deleting or orphaning commits in the commit history.

**25. What is the difference between resetting and reverting?**

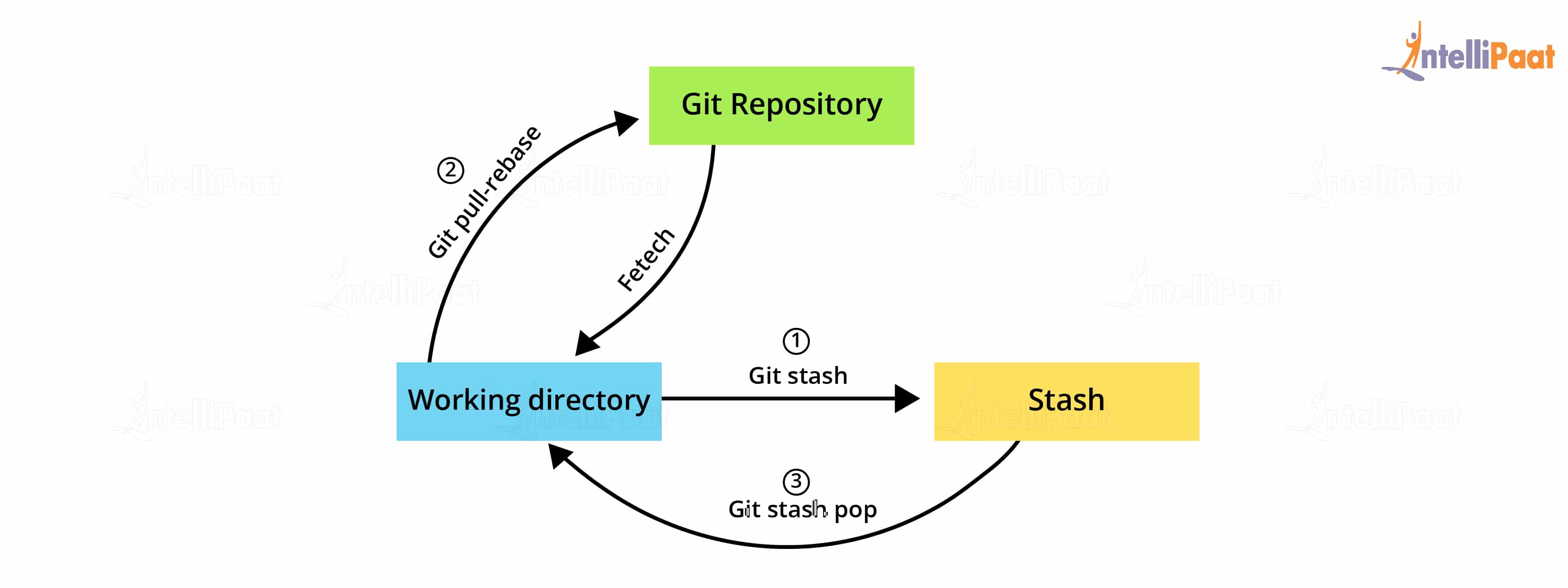
While git reset changes the state of the branch to a previous one by removing all of the states after the desired commit, git revert does it through the creation of new reverting commits and keeping the original one intact.

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

Git remote adds a reference to a remote repository for further tracking. Git clone, like its name, clones an existing remote repository and builds a new one.

**27. What is GIT stash?**

The Git stash will take the working directory in the current state and index it to put on the stack at a later stage so that what we get is a clean working directory.



This means that if we are in the middle of some task and need to get a clean working directory and simultaneously we want to keep all our current edits, then we can use the Git stash.

**28. How are fork, branch, and clone different from each other?**

Forking creates a copy of the original repository, and it remains in the GitHub account. Whereas, in cloning, the repository is copied to the local machine using Git. Forking is used to propose changes to the repository owners. In cloning, the changes are directly pushed to the original repository, provided the user has write access. A branch occurs within a repository and is a way to keep developing and modifying the software without affecting the main project.

**29. What is the difference between git reflog and log?**

The git log is a public record of the commit history for a branch. Reflog, on the other hand, is a private one of the repository’s local commits.

Unlike reflog, the git log is a part of the Git repository and is replicated after a push, fetch, or pull. A developer cannot access a local repository’s reflog without having access to the computer where it is located.

**30. What is GIT stash drop?**

When we are finished with working on the stashed item or want to remove the list, we can use the Git stash drop. This will ensure that the item that is last added by default or any particular item can be removed from the argument.

**31. How to identify if a certain branch has been merged into master?**

Git branch –merged master – shows all branches that are merged into master

Git branch – merged – shows all branches that are merged into the head

Git branch – no-merged –shows all the branches that are not merged

**32. Why do we need branching in GIT?**

With the help of branching, we can have our own branch and we can also jump between various branches. We can go to our previous work, at the same time keeping our recent work intact.

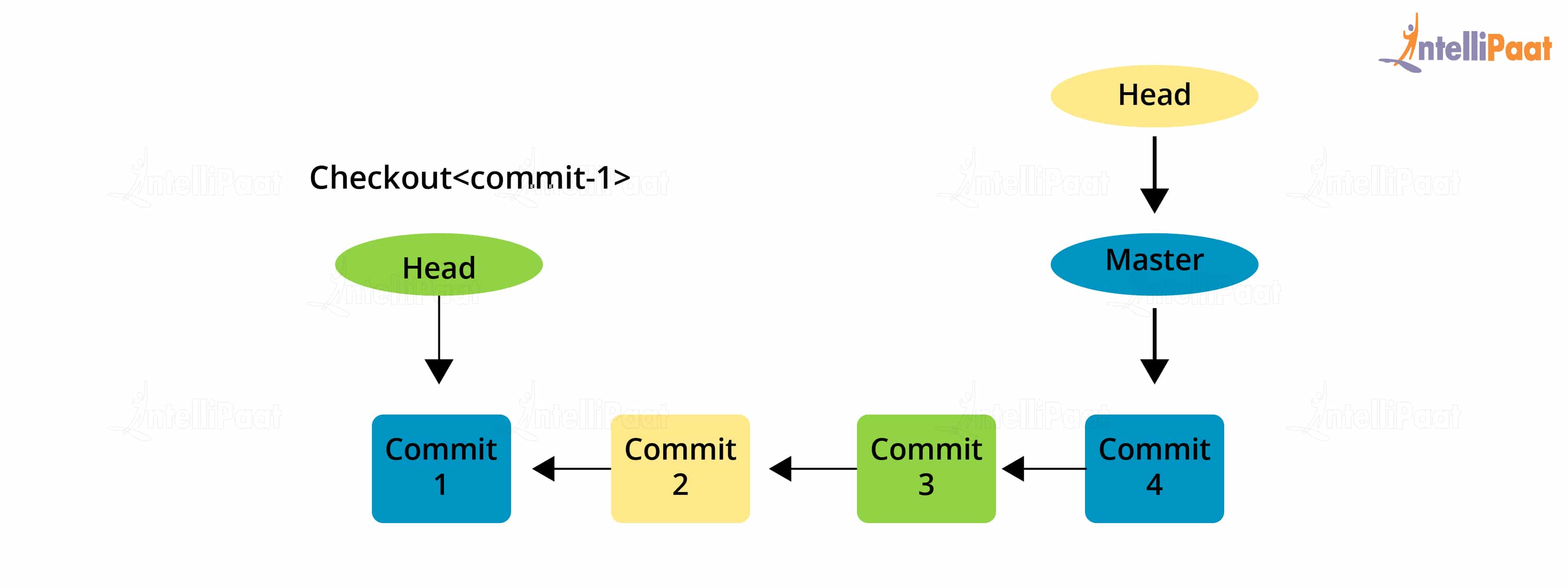
**Advanced Interview Questions**

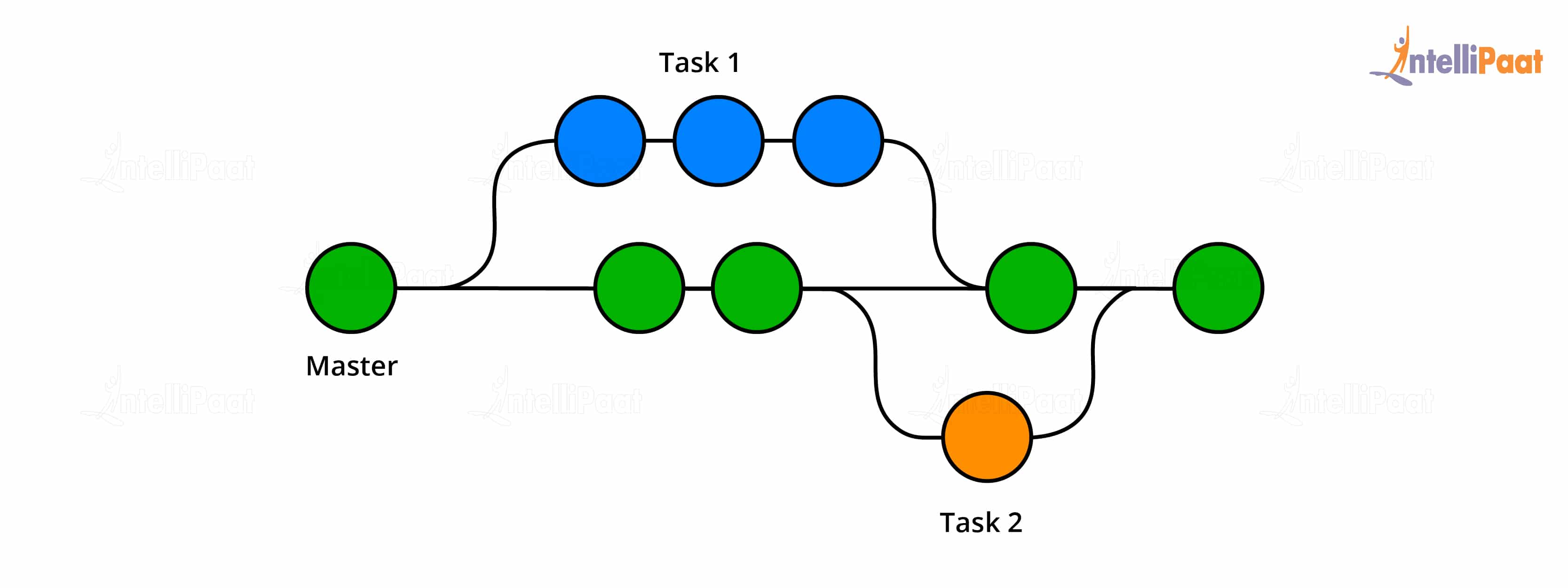
**33. What are the constituents of the commit object contain?**

* the state of a project at a given point of time is contained in a set of files
* Parent object commit references
* A 40-character string that uniquely identifies the commit object called a SHAI name

**34. What is HEAD in Git, and how many HEADs can be created in a repository?**

The reference to a commit object is called the HEAD. Every repository has a ‘Master’ which is the default head. There can be multiple heads in a repository.





**35. What is the regular way for branching in GIT?**

The best way to create a branch in GIT is to have one ‘main’ branch and then create another branch for implementing the changes that we want to make. This is extremely useful when there are a large number of developers working on a single project.

**36. State a way to create a new branch in Git.**

If we want to create a new feature into the main branch, then we can use the command ‘git merge’ or ‘git pull’.

**37. How do you define a ‘conflict’ in git?**

If we want to merge a commit there is a change in one place and the same change already exists then while merging the Git will not be able to predict which is the change that needs to be taken precedence.

**38. How to resolve a conflict in Git?**

If we want to resolve a conflict in Git, then we need to edit the files for fixing the conflicting changes and then we can run “git add” to add the resolved files and after that we can run the ‘git commit’ for committing the repaired merge.

[**Next**](https://intellipaat.com/blog/interview-question/aws-lambda-interview-questions/)

**Git Commands and Functions**

**git config:**It configures the username and email address.

**git add:**This command adds one or more files to the staging area.

**git add <file\_name>:**It specifically adds files one by one.

**git init:**This command initializes an empty Git repository.

**git commit:**It allows you to commit changes to the head but not to the remote repository.

**git stash:**It provides a clean working directory for other tasks while pushing the current working directory state and index to the stack for future use.

**git stash drop:** It deletes the stashed element (last-added stash item by default)  from the directory. Can also delete a specific topic included as an argument.

**git stash pop:** It discards the specified stash (topmost stash by default) after applying it.

**git pull:**It pulls innovation or commits from a specific branch of your central repository. It also updates your object branch in your local repository.

**git fetch:** It pulls all new commits from the desired branch. It then saves it in a new branch in your local repository.

**git merge:**It reflects the git fetch changes in your target branch. You can use git merge and specify the name of the other branch to bring it into the target branch.

**git branch - merged:** This command records the branches merged into the present branch.

**git reset --mixed:**You can use it for undoing changes of the working directory and the git index.

**git merge --abort:**It stops the merge process and returns back to the state before the merging occurs.

**git revert:**It creates a new commit to undo the changes of the previous commit. This command adds a new history without modifying the existing one.

**git reset:**You can use this command for undoing the local changes done in the git repository. It operates on the git index, commit history, and the working directory.

**git status:** It shows the difference between the working directory and the index.

**git clone:**It generates a copy of a current Git repository.

**git remote:**It helps you create, view, and delete remotes associated with the local repository.

**git branch –d [head]:** It simply deletes a branch.

**git branch -d <local\_branch\_name>:** You can use this command to delete a branch locally.

**git push origin --delete <remote\_branch\_name>:** You can use this command to delete a branch remotely.

**git reflog:** It tracks every single change made in the repository references and maintains the branches/tags log history, and thus gets the name reflog (reference + log).

**git annotate [<options>] <file> [<revision>]:** It annotates each line with information from the commit which introduced that change. It can also optionally annotate from a given revision.

**git cherry pick:**It introduces certain commits from one branch into another branch within the repository.

**git bisect:**It performs a binary search and detects the commit that introduced a bug or regression in the project’s history.

Here are some basic Git interview questions that you can practice for your upcoming interview:

1. How is Git different from SVN?
2. What are the advantages of using Git?
3. What are the limitations of Git?
4. Are Git and GitHub the same?
5. What is a Git repository?
6. How can you create a Git repository?
7. What is a bare repository?
8. Why do we require branching in Git?
9. What do you understand about the "Index" and "Staging Area" in GIT?
10. What is a conflict in Git? How will you resolve it?

*Learn about*[***how to become a DevOps engineer***](https://www.interviewkickstart.com/blog/how-to-become-a-devops-engineer)*here.*

**Advanced Git Interview Question and Answers**

Here are some advanced Git interview questions for senior developers to help you ace the challenging interview rounds.

**Q1. Why is it better to create an additional commit than amending an existing one?**

You must consider the following reasons while answering this Git interview question:

* The correct activity will devastate the recently saved commit. If you only change the commit message, it won't create a problem. However, if you modify the contents, there are more chances of excluding something important.
* If you abuse "git commit- amends," it can cause a small commit to increase and acquire inappropriate changes.

**Q2. Explain the different types of branching systems in Git.**

This Git interview question tests your branching knowledge with Git. You must elaborate on how you have utilized branching in your past activity. You can consider the following points while answering such Git interview questions:

**Feature branching:**A component branch model stores a majority of the changes for a specific element within a branch. The branch is converged into master when the item is tested and approved by automation.

**Task branching:**This model allows each assignment to be actualized on its branch with the undertaking key included in the branch name.

**Release branching:**You can clone the create branch that has procured enough features for a discharge to frame a release branch and begin the following discharge cycle. You cannot add new features after this point. It is the bug fixes, documentation age, and other release-oriented assignments that go in this branch.

**Q2. What is Subgit? Why would you use it?**

The tool that migrates SVN to Git is termed Subgit. It effectively detects the settings of your remote SVN repository and downloads SVN revisions. It then converts them to Git commits, thereby emerging as a stable solution for a company-wide migration from SVN to Git. The primary reasons for using Subgit are as follows:

* It is superior to git-svn.
* You need not change the infrastructure.
* It allows you to use all Git and all sub-version features.
* It provides a stress-free migration experience.

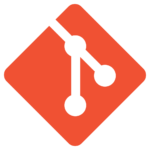
1. What are the advantages of forking workflow?

This is one of the most frequently asked Git interview questions. There are several advantages of the forking workflow over other popular git workflows, and you can compare them. You can highlight the following points while answering such Git interview questions.

* It gives you your own server-side repository rather than using a single server-side as the "central" codebase.
* It is ideal for public open-source projects.
* You can integrate contributions without requiring everybody to push to a single central repository that gives clean project history. You can push to your own server-side repositories as a developer, while the project maintainer can push to the official repository.

Given below are some of the advanced Git interview questions for your practice:

#### 1. What is Git?



Answer: Candidates would find this question among the most popular Git interview questions. [Git](https://www.whizlabs.com/blog/git-tutorial-understanding-git-basics/) is a distributed version control system (DVCS) and Source Code Management System (SCMS) that helps in tracking changes to a file. The functionality of Git allows developers to revert back to any specific change. Git can also help in the management of small and large projects with higher speed and efficiency. The distributed architecture of Git is the primary reason for its superiority in comparison to other version control systems. The most notable feature of Git is that it does not depend on a central server for storing all versions of the files in a project.

#### 2. Are Git and GitHub different?

Answer: Candidates can stumble upon this entry among basic Git interview questions. Git is a version control system for tracking modifications in source code over the course of software development. On the other hand, GitHub is the repository hosting service for Git. GitHub also provides additional features on its own. Some of the important features in GitHub include access control and collaboration features. In addition, GitHub also provides simple task management tools for different projects.

#### 3. What are the benefits of using Git?

Answer: Candidates should be prepared for this question as it is one of the most common Git interview questions. Here are the most noticeable benefits of Git.

* Any project could utilize Git without any restrictions
* Git supports collaboration
* Higher availability alongside functionality of data redundancy and replication
* Better disk utilization and network performance
* The facility of only one Git directory for every repository

#### 4. What language does Git use?

Answer: This is one of the trivial yet significant entry among top Git interview questions. However, candidates should make an effort to present the reason for the use of C language, rather than simply stating it! Git uses C language for the reduction of overheads with run times that are common in high-level languages.

#### 5. What is a repository in Git, and how can you create one?

Answer: Candidates will face a bot of technicality while addressing this type of basic Git interview questions. Repository in Git is the place that Git uses for storing all files. Git stores the files in the local repository or in the remote repository. The process to create a repository in Git is one of the most commonly asked questions. First of all, you have to create a directory for the project for creating a repository. Then, you can run the command “git init” and create a git repository in the project directory.

**Also Check:** [Top 25 OpenShift Interview Questions and Answers](https://www.whizlabs.com/blog/openshift-interview-questions-answers/)

#### 6. What is a bare repository?

Answer: Candidates can also come across this entry among top Git interview questions as a follow-up question. A bare repository contains information regarding version control. The bare repository does not contain any working files or the special.git sub-directory. On the contrary, the bare repository contains all information of the .git subdirectory in the main directory directly. The working directory contains the working tree, i.e., verified copies of project files and .git subdirectory with all Git related revision history for your repository.

#### 7. How is Git different from SVN?

Answer: The differences between Git and SVN are one of the notable entries among the best Git interview questions. First of all, Git does not have adequate functionality for managing excessively large files or frequently fluctuating binary files. On the other hand, SVN can manage multiple projects in the same repository.

The second point of difference is in the fact that Git lacks support for commits across multiple branches or tags. On the contrary, SVN or Subversion supports the creation of folders irrespective of the location in the repository layout. Modifications in Gits are impossible. However, Subversion provides flexibility to committers for treating a tag as a branch and creating multiple revisions in a tag root. However, the distributed Git is better than the centralized SVN.

#### 8. What is Git Commit, and what are the contents of the commit object?

Answer: Candidates should focus on this question as one of the important basic interview questions. The Git commit is a command executed during a project for recording the progress of the project. The commit object contains a set of files representing the state of a file at different instances of time. The commit object also consists of a reference to the parent commit. Finally, you can also find an SHAI name, a unique identifier with a 40-character string for the commit object.

#### 9. What is the different Git repository hosting functions?

Answer: This question is a prominent entry among the best Git interview questions. Apart from GitHub, the other prominent Git repository hosting functions include Gitlab, Bitbucket, GitEnterprise, and SourceForge.

#### 10. What is the conflict in Git?

Answer: Candidates could generally land up with a question on the conflict in Git in the basic Git interview questions. Git could handle the majority of merges on its own by leveraging its automatic features. Conflict is evident when two different branches make edits to the same line in a file. Another scenario for conflict in Git occurs with the deletion of a file on one branch and modification of the same file in another branch. Conflicts are general occurrences, especially when functioning in a team environment.

Preparing for a puppet interview? Go through these [top puppet interview questions](https://www.whizlabs.com/blog/top-puppet-interview-questions/) and answers and get ready to ace the interview.

#### 11. What is the process of resolving conflicts in Git?

Answer: This question is one of the prominent entrants among top Git interview questions. Candidates could expect this question alongside questions regarding definitions of conflict in git like in the previous question. Here are the steps for resolving conflicts in Git.

* Recognize the files responsible for the conflict
* Implement necessary changes to the files for avoiding any possibilities of conflict
* Add the files by using the command “git add.”
* The final step involves committing the changed file through the command “git commit.”

#### 12. What is branching, and how many types of branching are there in Git?

Answer: Candidates should prepare for this tricky entry among the most popular Git interview questions. Branching in Git is a helpful tool for supporting users in the creation of their personal branch and switch between the branches. Users can revert back to their old work without any compromises with the existing work. The different types of branching in Git are feature branching, task branching, and release branching. Feature branching involves keeping all changes of a particular feature in a branch.

After complete testing and validation of a feature, feature branching involves merging of the feature in the master. Task branching involves the inclusion of the new feature in the own branch along with including task key in the branch name. Release branching involves the formation of a clone of a branch in cases where a developed branch includes sufficient features for a release.

#### 13. What is SubGit?

Answer: Candidates can expect this question as one of the important Git interview questions at the starting level. SubGit is a tool for migrating from SVN to Git. SubGit can help in the creation of a writable Git mirror of a local or remote Subversion repository. It can use Subversion as well as Git for any duration the user needs. SubGit also provides faster one-time import from Subversion to Git. In addition, you can use SubGit within the Atlassian Bitbucket server. SubGit does not demand any changes in existing infrastructure. Furthermore, SubGit also offers flexibility for using all features of Git and Subversion.

#### 14. What are the uses of git instaweb?

Answer: The applications of Git commands are one of the prominent topics for the latest Git interview questions. The “git instaweb” command helps in automatically directing a web browser and running a web server with an interface to the local repository.

#### 15. What are the common Git commands and their uses?

Answer: Candidates could easily find this question among almost every discussion on Git interview questions. Here are some of the common Git commands, along with their functions.

“**git diff**” helps in showing changes between commits and those between commits and working tree.

“**git status**” helps in showing differences between the index and working directories

“**git stash applies**” is the command for bringing back saved changes on the working directory

“**git log**” helps in finding a specific commit in the history

“**git checkout**” is the command for updating directories of the working tree with directories from another branch without merging

“**git rm**” helps in removing files from staging area and files on the disk

“**git add**” helps in adding file modifications in the current directory to the index

“**git reset**” command helps in resetting the index. It also helps in resetting working directory to the state of the last commit

“**git is a tree**” is ideal for the representation of a tree object alongside the mode and name for each item

#### 16. What is the difference between clone, fork, and branch on Git?

Answer: This question is one of the unique entries among the best Git interview questions. A Git fork is a remote, server-side replica of a repository, different from the original. It is evident to note that a fork is not a Git concept and is a social paradigm. Clone in Git is a local copy of a particular remote repository. During the process of cloning, users copy the entire source repository information alongside all the branches and history. The branch is a process for the management of all changes in a single repository before merging them into the code. You can consider a branch as a thread of the development project that exists within a repository.

#### 17. How is a “branch” different from a “pull request”?

Answer: The difference between a branch and a pull request is one of the important DevOps interview questions for GitHub. The branch is merely a separate version of the code. A pull request becomes evident when an individual takes the repository, makes it their own branch, makes modifications, and then attempts merging that branch in another code repository.

#### 18. Do you know the use of “git cherry-pick”?

Answer: Commands will be a frequently occurring element in Git interview questions. The use of the “git cherry-pick” command is evident in the processes for the introduction of specific commits from a branch in a repository to another branch. The common application of “git cherry-pick” is evident in forward- or back-port of commits from a maintenance branch to a development branch. This approach is different from other approaches such as merge and rebase that generally apply various commits to another branch.

#### 19. What is the staging area in Git?

Answer: Candidates could find this entry among the majority of best Git interview questions. The staging area is an intermediate area that helps in formatting and reviewing commits before their completion. It is also known as an index. The staging area is the first place for verification of every change before committing to the repository.

#### 20. Is it reasonable to create an additional commit or modify an existing commit?

Answer: Such types of entries among Git interview questions can be confusing. You need to be clear that the creation of an additional commit is favorable as compared to modifying an existing commit. Modifications can destroy the previously saved state in the commit. Amendments in the content of the commit can lead to probabilities for the elimination of crucial data. Excessive use of “git commit- amend” can lead to the growth of a small commit and accumulation of unwanted changes.

**Check:** [Most Common Jenkins Interview Questions and Answers](https://www.whizlabs.com/blog/top-jenkins-interview-questions/)

### Intermediate Level Git Interview Questions

If you have a few years of experience working with Git, the interviewer may ask some command-based questions to the candidate for checking his knowledge. So, here we bring some commonly asked interview questions for the intermediate level Git professionals. Let’s go through these questions.

#### 21. What is the difference between the head, working tree, and index in Git?

Answer: Candidates should prepare for this entry among the latest Git interview questions for intermediate interview questions. The working tree is also known as the working directory or the workspace. It is the directory tree of source files that a user sees and edits. The index or the staging area is just a single, binary file in the <baseOfRepo>/.git/Index. The index contains a list of all files in the existing branch, the SHA1 checksums, file name, and time stamps. The “HEAD” implies a reference to the last commit in the existing checked-out branch.

#### 22. What is the use of “git config”?

Answer: You can find this question among the most popular Git interview questions. Git leverages your username for associating commits with a particular identity. The ‘git config’ command helps in changing Git configuration as well as your username. For example, if you want to have a username and email id for associating a commit with a particular identity, then you can use the following commands.

git config -global user. Name “Name” can add a username

git config -global user.email “E-mail Address” will add an email ID

#### 23. What is git stash?

Answer: This is one of the intermediate-level Git interview questions that appear in the majority of Git interviews. Working on a specific project involves a lot of mess and requirements for switching branches. Stashing on Git helps in taking your working directory with modified tracked files and staged modifications and storing them on a stack of unfinished changes. Users could reapply the changes at any time according to their preferences.

#### 24. What type of work can you restore with the recovery of a deleted branch?

Answer: Candidates should be prepared for unique Git interview questions like this one. Upon recovery of a deleted file, you can recover the stashed or saved files in the stashed index. Untracked files are not recoverable. The recommended best practice is to stage and commit your work in all cases or ensure their stashing.

#### 25. How is ‘git diff’ different from ‘git status’?

Answer: This entry is also one of the crucial DevOps interview questions for GitHub interviews. ‘git diff’ helps in representing the changes between commits and the changes between commits and working tree. On the other hand, ‘git status’ helps in finding the difference between the working directory and the index. As a result, it helps in understanding a particular git in detail. The prominent difference is that “git diff” shows differences between various commits, while “git status” does not.

#### 26. How can I know if a branch is already merged into master?

Answer: The answer to this type of Git interview questions is generally straightforward. The following commands can help in finding whether a branch has been merged in the master or not.

“git branch –merged” helps in listing out branches merged in the current branch.

“git branch –no -merged” helps in listing out branches not merged in the existing branch.

#### 27. What are the contents of ‘hooks’ in Git?

Answer: Candidates could find this entry among the latest Git interview questions. The “hooks” directory includes shell scripts that activate upon running the related Git commands. For instance, git would attempt the execution of a post-commit script after running a commit.

#### 28. How can I find a list of files changed in a specific commit?

Answer: Candidates would find practical questions like this among the most popular Git interview questions. The answer is quite simple! Just use the command ‘git diff-tree -r {hash}”. The ‘-r’ flag helps in listing the individual files. The output can contain some additional information. However, you can downplay the extra information with the help of two additional flags. The command with the additional flags would be “git diff-tree –no-commit-id –name-only -r {hash}.” The –no-commit-id helps in suppressing the commit hashes from the output. The “–name-only” flag helps in printing the file names rather than the paths.

#### 29. How can I remove a file from git without removal from the file system?

Answer: Candidates would have to prepare for this entry among difficult Git interview questions. “git rm” can help in removing files from staging area as well as the file system or working tree. However, the application of “git rm” cannot be valid in all cases. Therefore, you should use the “git reset” command with the syntax like “git reset filename.” You can also use the command “echo filename >> .gitignore” and add it to .gitignore library for avoiding re-additions.

#### 30. What are the differences between rebasing and merge in Git?

Answer: As you all know, commands would account for a major share of the latest Git interview questions. The rebase command in Git helps in the integration of changes from one branch to another. You can use it instead of the merge command. The merge command helps in taking independent lines of development in the git branch and integrating them in a single branch. The difference between the two commands is evident in the fact that rebases has to rewrite commit history for producing straight, linear order of commits.

It is required to be fully prepared before going for a Kubernetes interview. Prepare with these [top Kubernetes interview questions](https://www.whizlabs.com/blog/top-kubernetes-interview-questions/) to ace the interview!

#### 31. What is the difference between reverting and resetting?

Answer: Candidates could find this entry among new DevOps interview questions for GitHub interviews. The reset command in git helps in reverting local changes to the state of a Git repository. “git reset” works on commit history, the working directory, and the staging area. The revert command in git helps in the creation of a new commit that negates the changes from the previous commit. The revert command helps in adding a new history to the project without modifying existing history.

#### 32. How can you return a commit that has been pushed and made open?

Answer: “**git revert**” can help in reverting one or multiple commits. The command helps in the creation of a new commit that cancels out changes brought in the previous commits. The following command can help in reverting the two previous commits.

“git revert HEAD~2.HEAD”

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

Answer: Candidates could find this entry as one of the important Git interview questions. The first difference is that forming workflow does not use a single server-side as the “central” codebase. Every developer gets a personal server-side repository. Therefore, forking workflow is common in public open-source projects. Another advantage is the integration of contributions without the need for pushing to one particular central repository. Only the project manager can push to the official repository. Developers can let the project manager know that an update is ready for integration through a pull request.

#### 34. What is the difference between ‘git fetch’ and ‘git pull’?

Answer: Candidates should prepare for this entry among DevOps interview questions for GitHub interviews. ‘git fetches’ downloads only new data from remote repository. It does not ensure integration of downloaded data in your working files. ‘git pull’ is ideal for downloading as well as merging data from the remote repository in the local working files.

#### 35. What is the syntax for rebasing in Git?

Answer: The syntax for rebasing in Git is “**git rebase [new-commit]**”

#### 36. What is the use of ‘git bisect’?

Answer: The ‘git bisect’ is an important Git command for finding the command responsible for introducing a bug. The command uses a binary search algorithm for finding the commit in the project history that is responsible for introducing a bug.

#### 37. What is the Git stash drop?

Answer: Git stash drop is the command that helps in removing the list after completing work on the stashed item. As a result, it ensures the removal of any particular item or last added items from the argument.

#### 38. What are some of the best graphical Git client for LINUX?

Answer: Candidates could find this question commonly among Git interview questions. The best Git client for Linux is as follows –

* Git GUI
* Giggle
* Git Cola
* Smart Git
* qGit
* Git-g

#### 39. What is git pull origin?

Answer: The ‘git pull origin master’ command obtains commits from the master branch of the origin that could be the local origin or master branch. Then, it merges the origin or master into the currently checked out branch.

#### 40. What benefits do SCM tools offer with Git?

Answer: Users could avail exceptional benefits with SCM tools such as CVS, Subversion, ClearCase, and Perforce. The features such as convenient staging areas, multiple workflows, and cost-effective local branching validate the benefits of SCM tools.

Chef is one of the top DevOps tools. If you are going for a DevOps interview, don’t forget to check out these [top Chef interview questions](https://www.whizlabs.com/blog/top-chef-interview-questions/) and answers!

### Advanced Level Git Interview Questions

Being an advanced level Git professional, you may have advanced-level Git interview interviews in your interview. Let’s check out few advanced-level Git interview questions and prepare yourself for the interview.

#### 41. What is the process for squashing the last N commits to a single commit?

Answer: This entry is one of the advanced Git interview questions with two distinct responses depending on the context. In case of writing a new commit message from the start, you can utilize the following command.

“git reset -soft HEAD~N &&git commit.”

If you have to edit a new commit message with the addition of existing commit messages, then you should extract the messages and pass them to Git commit. The following command helps in achieving the above-mentioned function.

“git reset -soft HEAD~N &&git commit -edit -m“$(git log -format=%B -reverse [.HEAD@{N}](mailto:.HEAD@%257BN%257D))”

#### 42. How should I configure a Git repository for running code sanity checking tools?

Answer: Sanity checking helps in determining the possibility and feasibility of continuous testing. A sanity test is possible through a simple script that relates to the pre-commit hook of the concerned repository. The script also helps in running other tools such as linters and execute sanity checks for changes committed to the repository. Here is an example.

#!/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

The above-mentioned script evaluates the need for passing any .go file through the standard Go source code formatting tool. Exiting with a non-zero value helps the script prevent the application of the commit to the repository.

#### 43. What is git reflog?

Answer: “**git reflog**” command helps in tracking all the changes made in the references of a repository. It maintains a log history of locally created or checked out references to repository.

#### 44. How can I cherry-pick a merge commit?

Answer: Candidates could find this entry among advanced Git interview questions commonly. Cherry-pick uses a diff for finding out the difference between branches. With the merge commit for a different branch, it has two changesets and two parents. For instance, if you have merge commit ref 64cv89d, you have to specify -m and use parent 1 as a base as follows –

git checkout release-branch

git cherry-pick -m 1 64cv89d

#### 45. How can I copy a commit in one branch to another?

Answer: The ‘**cherry-pick**’ command is the best option in this case. It helps in finding the feasibility of reverting back an existing commit to the existing branch or location. Therefore, you have to switch to the target branch and then call the command “git cherry-pick {hash of that commit}”. As a result, you can find a new commit with a new hash because of the application of changes to a different destination.

#### 46. What if one of my teammates accidentally deleted a branch and pushed the changes to the central git repo and I want to recover the branch?

Answer: You need to review the latest commit to the particular branch with reflog. Then, you can check it out as a new branch.

#### 47. What is the Gitflow workflow?

Answer: Candidates could find this question among the most difficult Git interview questions. Gitflow workflow uses two long-running parallel branches known as master and develop. The components in Gitflow workflow are as follows.

The master branch is always ready for live release with everything production-ready.

The Hotflix branches help in quick patching of production releases.

The Develop branch helps in merging of all feature branches and also performs all the tests.

The Feature branch implies a unique branch for every new feature. The feature branch could be pushed to the development branch just like their parent branch.

#### 48. How is ‘git remote’ different from ‘git clone’?

Answer: ‘**git remote**’ helps in the creation of an entry in the git config while specifying a name for a specific URL. On the other hand, ‘git clone’ helps in the creation of a new git repository through copying an existing repository in the URL.

#### 49. What is the command for fixing a broken commit?

Answer: For fixing a broken commit, you can use the command “git commit –amend”. This command can help in fixing the broken commit message in the editor.

#### 50. What is the general branching pattern in Git?

Answer: The most general approach for the creation of branches in Git is the development of one “Main” branch. It also involves the creation of another branch for implementing new features. The pattern is helpful in cases where multiple developers work on a single project.

**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.

***Explore popular courses on Naukri Learning***

|  |  |
| --- | --- |
| [Popular Technology Course](https://www.naukri.com/learning/technology-courses-certification-training-ct139) | [Top Programming Courses](https://www.naukri.com/learning/programming-courses-certification-training-st619) |
| [Popular Cloud Computing Courses](https://www.naukri.com/learning/cloud-computing-courses-certification-training-st607-tg185) | [Popular Software Tools Courses](https://www.naukri.com/learning/software-tools-courses-certification-training-st293) |

**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:

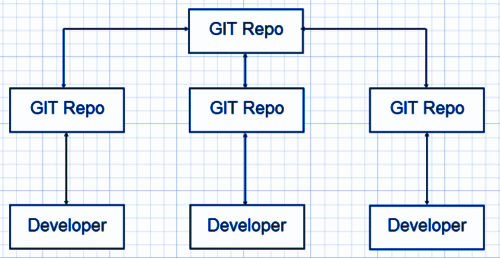
|  |  |
| --- | --- |
| **Git** | **GitHub** |
| Git is a software tool to use a version control system | GitHub is a hosting service for git repositories |
| Tool for projects that want to collaboratively develop software | Service for projects that use Git for version control |

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

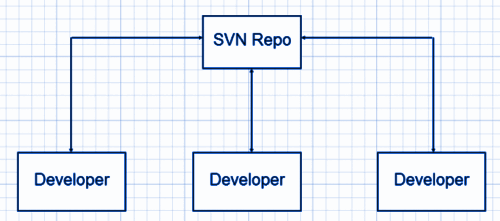
**Ans.** Both Git and SVN have their pros and cons, and below are the differences between Git and SVN:

|  |  |
| --- | --- |
| **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 GIT?**

GIT is a distributed version control system and source code management (SCM) system with an emphasis to handle small and large projects with speed and efficiency.

**2) What is a repository in GIT?**

A repository contains a directory named .git, where git keeps all of its metadata for the repository. The content of the .git directory are private to git.

**3) What is the command you can use to write a commit message?**

The command that is used to write a commit message is “git commit –a”.  The –a on the command line instructs git to commit the new content of all tracked files that have been modified. You can use “git add<file>” before git commit –a if new files need to be committed for the first time.

**4) What is the difference between GIT and SVN?**



The difference between GIT and SVN is

a) Git is less preferred for handling extremely large files or frequently changing binary files while SVN can handle multiple projects stored in the same repository.

b) GIT does not support ‘commits’ across multiple branches or tags.  Subversion allows the creation of folders at any location in the repository layout.

c) Gits are unchangeable, while Subversion allows committers to treat a tag as a branch and to create multiple revisions under a tag root.

**5) What are the advantages of using GIT?**

a) Data redundancy and replication

b) High availability

c) Only one.git directory per repository

d) Superior disk utilization and network performance

e) Collaboration friendly

f) Any sort of projects can use GIT

**6) What language is used in GIT?**

GIT is fast, and ‘C’ language makes this possible by reducing the overhead of runtimes associated with higher languages.

**7) What is the function of ‘GIT PUSH’ in GIT?**

‘GIT PUSH’ updates remote refs along with associated objects.

**8) Why GIT better than Subversion?**

GIT is an open source version control system; it will allow you to run ‘versions’ of a project, which show the changes that were made to the code overtime also it allows you keep the backtrack if necessary and undo those changes.  Multiple developers can checkout, and upload changes and each change can then be attributed to a specific developer.

**9) What is “Staging Area” or “Index” in GIT?**

Before completing the commits, it can be formatted and reviewed in an intermediate area known as ‘Staging Area’ or ‘Index’.

**10) What is GIT stash?**

GIT stash takes the current state of the working directory and index and puts in on the stack for later and gives you back a clean working directory.  So in case if you are in the middle of something and need to jump over to the other job, and at the same time you don’t want to lose your current edits then you can use GIT stash.

**11) What is GIT stash drop?**

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

**12) How will you know in GIT if a branch has been already merged into master?**

Git branch—merged lists the branches that have been merged into the current branch

Git branch—-no merged lists the branches that have not been merged

**13) What is the function of git clone?**

The git clone command creates a copy of an existing Git repository.  To get the copy of a central repository, ‘cloning’  is the most common way used by programmers.

**14) What is the function of ‘git config’?**

The ‘git config’ command is a convenient way to set configuration options for your Git installation.  Behaviour of a repository, user info, preferences etc. can be defined through this command.

**15) What does commit object contain?**

a)      A set of files, representing the state of a project at a given point of time

b)      Reference to parent commit objects

c)       An SHAI name, a 40 character string that uniquely identifies the commit object.

**16) How can you create a repository in Git?**

In Git, to create a repository, create a directory for the project if it does not exist, and then run command “git init”. By running this command .git directory will be created in the project directory, the directory does not need to be empty.

**17) What is ‘head’ in git and how many heads can be created in a repository?**

A ‘head’ is simply a reference to a commit object. In every repository, there is a default head referred as “Master”.  A repository can contain any number of heads.

**18)   What is the purpose of branching in GIT?**

The purpose of branching in GIT is that you can create your own branch and jump between those branches. It will allow you to go to your previous work keeping your recent work intact.

**19) What is the common branching pattern in GIT?**

The common way of creating branch in GIT is to maintain one as “Main“

branch and create another branch to implement new features. This pattern is particularly useful when there are multiple developers working on a single project.

**20) How can you bring a new feature in the main branch?**

To bring a new feature in the main branch, you can use a command “git merge” or “git pull command”.

**21) What is a ‘conflict’ in git?**

A ‘conflict’ arises when the commit that has to be merged has some change in one place, and the current commit also has a change at the same place. Git will not be able to predict which change should take precedence.

**22) How can conflict in git resolved?**

To resolve the conflict in git, edit the files to fix the conflicting changes and then add the resolved files by running “git add” after that to commit the repaired merge,  run “git commit”.  Git remembers that you are in the middle of a merger, so it sets the parents of the commit correctly.

**23) To delete a branch what is the command that is used?**

Once your development branch is merged into the main branch, you don’t need

development branch.  To delete a branch use, the command “git branch –d [head]”.

**24) What is another option for merging in git?**

“Rebasing” is an alternative to merging in git.

**25) What is the syntax for “Rebasing” in Git?**

The syntax used for rebase is “git rebase [new-commit] “

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

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

**27) What is GIT version control?**

With the help of GIT version control, you can track the history of a collection of files and includes the functionality to revert the collection of files to another version.  Each version captures a snapshot of the file system at a certain point of time. A collection of files and their complete history are stored in a repository.

**28) Mention some of the best graphical GIT client for LINUX?**

Some of the best GIT client for LINUX is

a) Git Cola

b) Git-g

c) Smart git

d) Giggle

e) Git GUI

f) qGit

**29) What is Subgit? Why to use Subgit?**

‘Subgit’ is a tool for a smooth, stress-free SVN to Git migration.  Subgit is a solution for a company -wide migration from SVN to Git that is:

a) It is much better than git-svn

b) No requirement to change the infrastructure that is already placed

c) Allows to use all git and all sub-version features

d) Provides genuine stress –free migration experience.

**30) What is the function of ‘git diff ’ in git?**

‘git diff ’ shows the changes between commits, commit and working tree etc.

**31) What is ‘git status’ is used for?**

As ‘Git Status’ shows you the difference between the working directory and the index, it is helpful in understanding a git more comprehensively.

**32) What is the difference between the ‘git diff ’and ‘git status’?**

‘git diff’ is similar to ‘git status’, but it shows the differences between various commits and also between the working directory and index.

**33) What is the function of ‘git checkout’ in git?**

A ‘git checkout’ command is used to update directories or specific files in your working tree with those from another branch without merging it in the whole branch.

**34) What is the function of ‘git rm’?**

To remove the file from the staging area and also off your disk ‘git rm’ is used.

**35) What is the function of ‘git stash apply’?**

When you want to continue working where you have left your work, ‘git stash apply’ command is used to bring back the saved changes onto the working directory.

**36) What is the use of ‘git log’?**

To find specific commits in your project history- by author, date, content or history ‘git log’ is used.

**37) What is ‘git add’ is used for?**

‘git add’ adds file changes in your existing directory to your index.

**38) What is the function of ‘git reset’?**

The function of ‘Git Reset’ is to reset your index as well as the working directory to the state of your last commit.

**39) 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.

**40) How git instaweb is used?**

‘Git Instaweb’ automatically directs a web browser and runs webserver with an interface into your local repository.

**41) What does ‘hooks’ consist of in git?**

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

**42) Explain what is commit message?**

Commit message is a feature of git which appears when you commit a change. Git provides you a text editor where you can enter the modifications made in commits.

**43) How can you fix a broken commit?**

To fix any broken commit, you will use the command “git commit—amend”. By running this command, you can fix the broken commit message in the editor.

**44) Why is it advisable to create an additional commit rather than amending an existing commit?**

There are couple of reason

a)  The amend operation will destroy the state that was previously saved in a commit.  If it’s just the commit message being changed then that’s not an issue.  But if the contents are being amended then chances of eliminating something important remains more.

b) Abusing “git commit- amend” can cause a small commit to grow and acquire unrelated changes.

**45) What is ‘bare repository’ in GIT?**

To co-ordinate with the distributed development and developers team, especially when you are working on a project from multiple computers ‘Bare Repository’ is used. A bare repository comprises of a version history of your code.

**46) Name a few Git repository hosting services**

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

--------------------------------------------------------------------------------------------------------------------