Git & Github

- Git : (Git Book)

- A version control system (VCS) is used tracks the history of changes as people & teams collaborate on projects together.

- Git is a VCS. It is:
  - popular
  - free & Open Source
  - fast & Scalable

- Github :

- It is a website that allows developers to store and manage their code using git

- Github repository

- A repository, or Git Projects, encompasses the entire collection of files & folders associated with a project, along with each files revision history

- Git Command line

```
git --version
```

Git

(i) It is S/w
(ii) command-line tool
(iii) maintained by linux
(iv) launched 2005
(v) installed on the system

Github

(i) It is service
(ii) graphical user interface
(iii) Maintained by Microsoft
(iv) 2008.
(v) hosted on the web.

- Configuring Git (~ → root directory inside)

git config --global user.name "My Name"

git config --global user.email "--@gmail.com"

git config --list (कोणसे अकाउंट के अंदर change करना वो command)

- Clone & Status.
- Clone - Cloning repository on our local m/c.

git clone <--link-->

- Status - display the state of code

git status

- Cd .. - for exit directory
- Cd - change directory
- clear - clear terminal
- ls : list of
- ls -a : Folder all hidden files

- Repository status.
- untracked
new files that git doesn't yet track
- modified
changed
- staged
file is ready to be commited
- Unmodified
unchanged.

- Add & Commit
- Add - adds new or changed files in your working directory to Git staging area.

git add <--file name-->

- Commit - it is the record of change

git commit -m "some msg"

- Push Command

Upload local repo content to remote repo

git push origin main

• Init Command

- It is used to create new git repo
- initialize new git repo.

git init

Note [mkdir - used to make new directory].
eg. mkdir LocalRepo → name.

- git init
- git remote & origin <.link.>
- git remote -v (to verify remote)
- git branch (to check branch)
- git branch -M main (to rename branch)
- git push origin main

Note : git push -u origin main
↓
set upstream

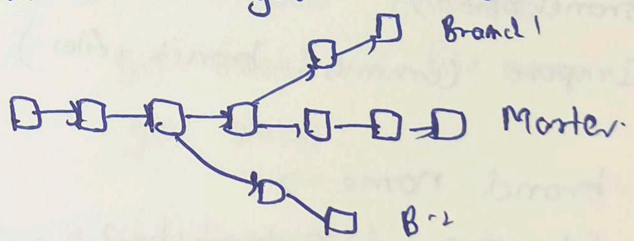
[.When we want work on some project on long time & we work on origin main then -u is used.)

• Workflow.

Github repository
↓
Clone
↓
Changes
↓
add
↓
Commit
↓
Push

• Git Branch :

- A branch is a version of repo. that diverges from the main working project.
- When you want to add new feature a fix. bug. you add new branch to summarize your changes.



- Master Branch - default branch. (only one master branch)
- When you make first commit. youre given a master branch to starting commit pt.

- When you start making a commit, then master branch pointer automatically moves forward.
- It is the branch in which all the changes eventually get merged back

git branch (to check branch)

git branch -M main (to rename branch)

git checkout <- branch name -> (to navigate)

git checkout -b <- new branch name ->
(to create new branch)

git branch -d <- branch name ->
(to delete branch).

• Merging Code

• Way 1

~~dfc git diff~~

git diff <- branch name ->
(to compare commits, branch, files)

git merge <- branch name ->
(to merge 2 branches)

• Way 2

Create a PR

• Pull Request.

- It lets you tell others about changes you've pushed to a branch in a repo. on GitHub.

• Pull Command

git pull origin main

- used to fetch & download content from a remote repo & immediately update the local repo. to match that content.

• Resolving Merge Contents.

- An event that takes place when git is unable to automatically resolve differences in code bt^n two commits.

• UNDOing Changes.

Case 1 : Staged changes (No commit)

git reset <- file name ->

git reset.

Case 2 : Commited changes (one commit)

git reset HEAD -1

Case 3: Committed changes
(For many commits)

git reset <- commit hash ->

git reset --hard <- commit head ->

- Fork.

- A Fork is a new repo. that shares code & visibility settings with the original "upstream" repo.
- Fork is a rough copy

- Git help.

- If you are having trouble remembering commands or options for commands, you can use Git help.

git command --help. (see all available options for specific command)

git help --all (see all possible commands)

ChatGPT

- Introduction.

- ChatGPT is an AI ~~content~~ chatbot developed by OpenAI & released in Nov. 2022

- It is used to generate content like

(i) generate code for ---

(ii) Translate Code to another lang.
translate this code to java.

(iii) Find Bug
Detect the error in this code

(iv) Generate Optimized Version of code
generate C++ code for swapping 2 no. with best time complexity.

(v) Explain Code

(vi) Ask Git Commands.

(vii) Write Documentation

(viii) Write Resume

(ix) Prepare for interviews.

① Code Debuging & Testing.

• limitations

- Accuracy Problem
- Sensitivity to phrasing
- Harmful instruction.
- Lack of creativity
- Lack of common sense
- Difficulty with specialized topics

• features

- Natural language Understanding
- Knowledge & Info.
- Creativity
- Suggestions
- Language Translation
- Continuous learning

• GPT-3 - Released in Nov. 2020.
less accurate & fluent.

• GPT-3.5 - released in June 2021
intermediate accurate

• GPT 4. released in Nov. 2022
more accurate than other

Basic GIT Interview Questions

1. 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.

2. 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.

3. 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>>"

4. 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`

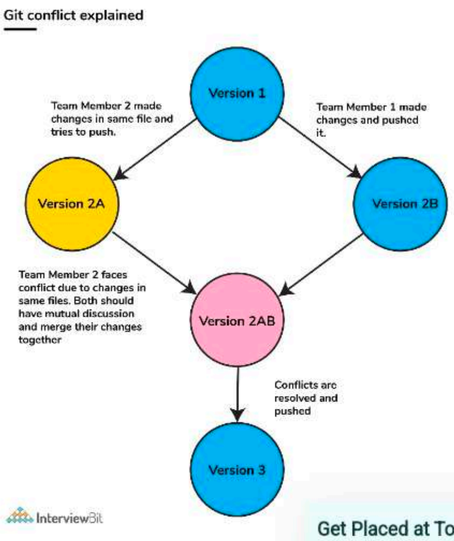
  - To make a new commit by reverting the last 3 commits, we can run the command: `git revert --no-commit HEAD~3...HEAD`

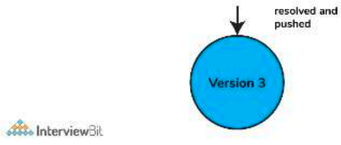
5. What is a conflict?

- 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.





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

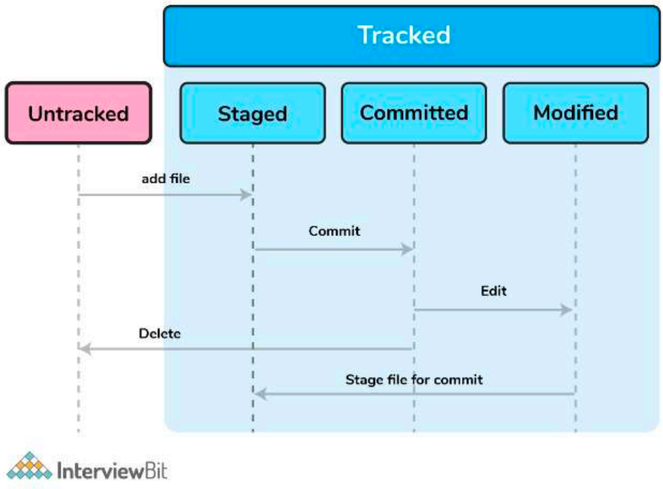
This command returns a tree object representation of the current repository along with the mode and the name of each item and the SHA-1 value of the blob.

7. 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.

8. 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.

9. 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.

10. 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.

Intermediate GIT Interview Questions

11. 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.

12. 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.

13. 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 b

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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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
```

19. 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.

20. 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 easie

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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
```

27. 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.

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

- “Push request” is termed so because it is done when the target reposit

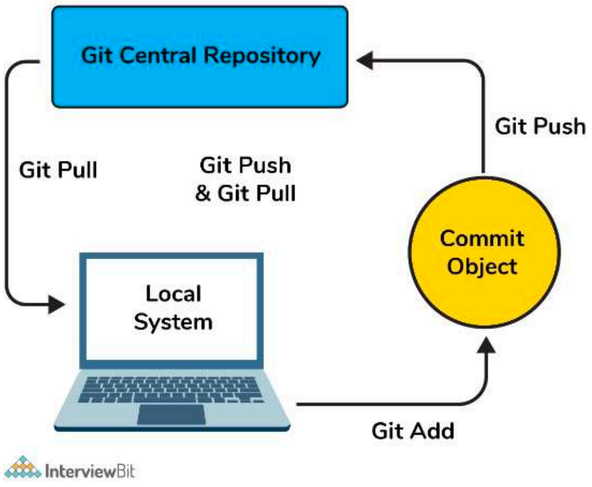
```
git pull = git fetch + git merge
```

27. 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.

28. 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.



29. 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.

Advanced GIT Interview Questions

30. 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.

31. 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.

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

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.

Conclusion

Basic Git Interview Questions for Freshers

1. What is Git?

Git is a distributed version control system (DVCS) that is used to track changes in source code during software development. It permits multiple developers to work on a project together without interrupting each other's changes. Git is especially popular for its speed, and ability to manage both small and large projects capably.

2. What is a repository in Git?

A Git repository (or repo) is like a file structure that stores all the files for a project. It continues track changes made to these files over time, helping teams work together evenly. Git can control both local repositories (on your own machine) and remote repositories (usually hosted on platforms like GitHub, GitLab, or Bitbucket), allowing teamwork and backup.

3. What is the difference between Git and GitHub?

Git	GitHub
Git is a version control system used to track changes in files over time	GitHub is a platform where Git repositories can be stored and shared
It runs locally on your computer	It is a cloud-based service
Git can be used offline, as it operates locally on your machine.	GitHub requires an internet connection because it is hosted on the web

4. What is origin in Git?

In Git, "origin" states to the default name offered to the remote repository from which local repository was cloned. Git origin is used as a reference to control fetches, pulls, and pushes.

5. What is the purpose of the .gitignore file?

The '.gitignore' file tells Git which files and folders to ignore when tracking changes. It is used to avoid attaching unneeded files (like logs, temporary files, or compiled code) to your repository. This saves repository clean and targeted on important files only.

6. What is a version control system (VCS)?

A version control system (VCS) records the work of developers coordinating on projects. It keeps the history of code changes, permitting developers to add new code, fix bugs, and run tests securely. If required, they can restore a past working version, verifying project security.

7. What is the git push command?

The 'git push' command is used to share local repository changes to a remote repository. It changes the remote repository with the recent commits from the fixed local branch.

8. What is the git pull command?

The 'git pull' command updates the current local branch with changes from a remote repository and combining it with a local repository.

9. What does git clone do?

The git clone forms a copy of a remote repository upon your local machine. Git clone downloads all files, branches, and history, enabling you to start working on the project or contribute to it. With git clone -b , you can download and work on an individual branch of a repository.

10. What are the advantages of using GIT?

Using Git provides multiple advantages:

- It assists teamwork by supporting multiple developers to work on the same project together.
- Each developer has a local copy of the repository, improving performance and enabling offline

clone -b , you can download and work on an individual branch of a repository.

10. What are the advantages of using GIT?

Using Git provides multiple advantages:

- It assists teamwork by supporting multiple developers to work on the same project together.
- Each developer has a local copy of the repository, improving performance and enabling offline work.
- Free and widely supported.
- Git supports work on various types of projects.
- Each repository has only one Git directory.

11. What is the difference between git init and git clone?

The git init develops a new, empty Git repository in the present directory, while 'git clone' copies an existing remote repository, containing all files and history, to a local directory.

12. What is git add?

The git add command marks changes in your project for the next commit. It tells Git which files to involve in the later update, making them ready to be saved in the repository. This is the early step in recording changes in the Git repository.

13. What is git status?

The 'git status' command shows the recent status of your Git repository. It tells you which files have changed, which ones are ready to be committed, and which ones are new and unobserved. This benefits you monitor your work's growth and see what changes want to be set up or committed.

14. What is a commit in Git?

A commit in Git denotes a snapshot of changes made to files in a repository. It grabs all the changes you have made to files—like additions, or deletions of files at a particular moment. Each commit has a unique message explaining what was done. This helps you track your project's history, undo changes if requisite, and work with others on the same project.

15. What is the purpose of the git clean command?

The git clean command is used to erase ignored files from the working directory of Git repository. Its motive is to clean up the workspace by deleting files that are not being saved by Git, checking a clean state with only observed files present.

16. What is a 'conflict' in git?

Git usually manages merges automatically, but conflicts occur when two branches edit the same line or when one branch deletes a file that another edits.

17. What is the meaning of 'Index' in GIT?

In Git, the 'Index' (also called as the "Staging Area") is a place where alterations are temporarily store before committing them to the repository. It permits you to select and prepare specific alterations from your working directory before properly saving them as part of the project's history.

18. How do you change the last commit in git?

To change the preceding commit in Git, use 'git commit --amend' after making changes, stage them with 'git add' , and save with the editor.

19. What is `git checkout`?

The 'git checkout' helps you switch between branches or return files to a previous state in Git. Now, it is suggested to use 'git switch' for changing branches and 'git restore' to return files. These commands are more intent on their particular tasks for better clearness and capability.

20. How do you switch branches in Git?

To switch branches in Git, use 'git checkout ' to move to a present branch. On the other hand, use git switch in newer Git versions for the same objective. This permits you to work on different versions of features of your project across separate branches.

20. How do you switch branches in Git?

To switch branches in Git, use 'git checkout ' to move to a present branch. On the other hand, use git switch in newer Git versions for the same objective. This permits you to work on different versions or features of your project stored in separate branches.

21. Name some popular Git hosting services?

- GitHub
- GitLab
- SourceForge.net
- Bitbucket
- Visual Studio Online

22. What are the different types of Git repositories?

Git has two types of repositories

- **Bare Repository:** A repository without a working directory, typically used for sharing projects remotely.
- **Non-Bare Repository:** A repository that includes a working directory where developers can modify and track files.

23. How does Git handle file deletion?

Git provides the git rm command to remove files from the working directory and the staging area. If you only want to remove a file from Git but keep it in the working directory, use:

```
git rm --cached <file_name>
```

24. How can you create an alias in Git?

Aliases can be created to simplify Git commands using:

```
git config --global alias.<alias_name> '<git_command>'
```

25. How do you rename a branch in Git?

To rename the current branch:

```
git branch -m <new_branch_name>
```

To rename a different branch:

```
git branch -m <old_branch_name> <new_branch_name>
```

Intermediate Git Interview Questions and Answers

26. What is the difference between git fetch and git pull?

Command	Description
git fetch	Retrieves updates from a remote repository but does not merge them into the local repository. This allows reviewing changes before integrating them.
git pull	Fetches updates from a remote repository and immediately merges them into the current local branch. Equivalent to git fetch followed by git merge.

27. Explain Git rebase and when do you use it?

The Git rebase is a process to combine alterations from one branch into another. It forms a linear history, avoiding merge commits. Use it to clean up commit history, keep a project history sequential, and make feature branches up-to-date before uniting.

28. How will you create a git repository?

- Download Git on your system if you have not already.
- Create a project folder in the location where you want your repository.
- Open Terminal or Command Prompt and visit your project folder.
- Run git init in the project folder. This will create a .git folder, showing your repository is set.

sequential, and make feature branches up-to-date before uniting.

28. How will you create a git repository?

- Download Git on your system if you have not already.
- Create a project folder in the location where you want your repository.
- Open Terminal or Command Prompt and guide to your project folder.
- Run 'git init' in the project folder. This will create a '.git' folder, showing your repository is set.

29. What differentiates between the commands git remote and git clone?

Command	Description
git remote	Manages connections to remote repositories. It allows users to add, remove, and view remote repositories linked to a local project. However, it does not download any files.
git clone	Creates a local copy of an existing remote repository, including all its files, branches, and commit history. This allows developers to start working on a project immediately.

30. What are the benefits of using a pull request in a project?

Teams can together work on distinct parts of the system and later combine their changes using pull requests. This way boosts team capability.

- **Code Review:** Pull requests allow team members to review code before merging, ensuring better code quality.
- **Collaboration:** Multiple developers can work on the same project and discuss changes within the pull request.
- **Version Control:** Helps track proposed changes and keeps the main branch stable.

31. What is a Git bundle?

A Git bundle is a collective file that wraps all data from Git repository, such as commits, branches, and tags. It acts as a handy approach for relocating a repository offline or sharing upgrades when network connection is not available. To form a git bundle, perform the following command:

```
git bundle create <bundle_file> <refs>
```

32. What are the advantages of Git over SVN?

Here the some advantages of Git over SVN:

- **Distributed Version Control:** Git allows every developer to have a complete copy of the repository, whereas SVN relies on a centralized model.
- **Faster Performance:** Git is generally faster due to local operations, while SVN requires network communication for many actions.
- **Branching and Merging:** Git offers lightweight and efficient branching, whereas SVN branches are heavier and more complex.
- **Offline Work:** Since Git is distributed, developers can work offline, committing changes locally before pushing them.
- **Better Conflict Resolution:** Git provides more advanced tools for resolving merge conflicts compared to SVN.

33. What is git stash?

The git stash is a command used to temporarily save changes that are not yet ready to be committed. It allows developers to switch branches or work on something else without losing their progress. Stashed changes can be reapplied later using git stash pop or git stash apply.

Key Git Stash Commands:

- **Save current changes:**

```
git stash
```

- **View stashes:**

```
git stash list
```

- **Reapply the most recent stash and remove it from stash history:**

```
git stash pop
```

- **Better Conflict Resolution:** Git provides more advanced tools for resolving merge conflicts compared to SVN.

33. What is git stash?

The git stash is a command used to temporarily save changes that are not yet ready to be committed. It allows developers to switch branches or work on something else without losing their progress. Stashed changes can be reapplied later using git stash pop or git stash apply.

Key Git Stash Commands:

- **Save current changes:**

```
git stash
```

- **View stashes:**

```
git stash list
```

- **Reapply the most recent stash and remove it from stash history:**

```
git stash pop
```

- **Reapply a stash without removing it from history:**

```
git stash apply
```

- **Delete a specific stash:**

```
git stash drop stash@{n}
```

34. How do you revert a commit that has already been pushed and made public?

To revert a commit that has been pushed and made public, follow these steps:

- Checkout the Branch: Switch to the branch where you want to revert the commit.

```
git checkout <branch-name>
```

- Find the Commit to Revert: Use 'git log' to find the commit hash of the commit you want to revert.

```
git log
```

- Revert the Commit: Use 'git revert' followed by the commit hash of the commit you want to revert.

```
git revert <commit-hash>
```

- Review Changes: Git will open your default text editor to confirm the revert message. Save and close the editor to proceed.
- Push the Revert: Finally, push the reverted commit to the remote repository.

```
git push origin <branch-name>
```

35. Explain the difference between reverting and resetting?

Operation	Description
git reset	Moves the HEAD pointer to a previous commit, removing changes from the commit history. It can modify staged and working directory changes.
git revert	Creates a new commit that undoes changes from a previous commit without modifying commit history. It is safer when working in a shared repository.

35. Explain the difference between reverting and resetting?

Operation	Description
git reset	Moves the HEAD pointer to a previous commit, removing changes from the commit history. It can modify staged and working directory changes.
git revert	Creates a new commit that undoes changes from a previous commit without modifying commit history. It is safer when working in a shared repository.

36. What is the difference between git reflog and log?

Feature	git reflog	git log
Purpose	Tracks movements of HEAD and branch changes	Displays commit history of a repository
Visibility	Shows all references, even those not in branch history	Only shows commits in the current branch history
Recoverability	Can help recover lost commits	Does not track changes outside commit history
Use Case	Used to restore lost branches or commits	Used to review past commits and changes

37. What is the HEAD in Git?

- HEAD in Git is a reference to the latest commit on the currently checked-out branch.
- It determines which commit and branch you are working on.
- When switching branches, HEAD updates to point to the latest commit of the new branch.
- If in a detached HEAD state, it means you are working on a specific commit instead of a branch.

38. What is the purpose of 'git tag -a'?

The git tag -a command is used to create an annotated tag in Git. Annotated tags store additional metadata such as the tagger's name, email, date, and a message describing the tag. These tags are useful for marking important points in a repository, such as software releases.

```
git tag -a v1.0 -m "Version 1.0 Release"
```

To push annotated tags to a remote repository:

```
git push origin v1.0
```

39. What is the difference between 'HEAD', 'working tree' and 'index' in Git?

Concept	Description	Role	Location
HEAD	A reference to the current commit or branch you are working on. It points to the tip of the current branch.	Represents the current branch or commit you are on	Located in .git/HEAD.
Working Tree	The directory where your project files reside. It reflects the actual state of files on your filesystem and includes changes made but not yet staged.	Contains the actual files, including any modifications or additions not yet staged.	The local directory in your project.
Index(Staging Area)	A file (also called the staging area) that acts as a buffer between the working tree and the repository. Files staged here are ready to be committed.	Holds changes that you want to commit to the repository.	Located in .git/index.

```
git push origin v1.0
```

39. What is the difference between 'HEAD', 'working tree' and 'index' in Git?

Concept	Description	Role	Location
HEAD	A reference to the current commit or branch you are working on. It points to the tip of the current branch.	Represents the current branch or commit you are on	Located in .git/HEAD.
Working Tree	The directory where your project files reside. It reflects the actual state of files on your filesystem and includes changes made but not yet staged.	Contains the actual files, including any modifications or additions not yet staged.	The local directory in your project.
Index(Staging Area)	A file (also called the staging area) that acts as a buffer between the working tree and the repository. Files staged here are ready to be committed.	Holds changes that you want to commit to the repository.	Located in .git/index.

40. How to resolve a conflict in Git?

To resolve a conflict in Git

- **Identify Conflict:** Git will alert you of a conflict during merge or rebase.
- **Open the Conflicted File:** You'll see conflict markers like <<<<<<< HEAD, =======, and >>>>>>>.
- **Resolve the Conflict:** Edit the file to keep your changes, the incoming changes, or a combination of both.
- **Stage the Resolved File:** Use git add <file> to mark the conflict as resolved.
- **Commit the Changes:** Run git commit to complete the merge.
- **Continue Rebase (if applicable):** Use git rebase --continue if you were rebasing.

41. Explain the difference between 'git merge' and 'git rebase' and when you would use each?

Feature	git merge	git rebase
What it does	Combines two branches and creates a merge commit.	Reapplies commits from one branch on top of another, creating a linear history.
Commit History	Keeps the commit history of both branches intact, including a merge commit.	Rewrites the commit history to make it linear.
Result	A new merge commit that preserves the branch structure.	A clean, straight history without merge commits.
When to use	When you want to preserve both branches' histories and structure.	When you want a clean, linear commit history without merge commits.
Best for	Collaborative environments and when integrating feature branches.	Personal branches, squashing, or cleaning up commit history before pushing.
Merge Commit	Yes, a merge commit is created.	No, history is rewritten without merge commits.

42. What language is used in GIT?

Git is mainly developed using the C programming language. The core features and commands of Git, containing its data structures and algorithms, are applied in C. This choice of language confirms productivity, speed, and portability across distinct operating systems and platforms.

43. How do you add a file to the staging area?

To add a file to the staging area in Git, use the command:

```
git add <file>
```

productivity, speed, and portability across distinct operating systems and platforms.

43. How do you add a file to the staging area?

To add a file to the staging area in Git, use the command:

```
git add <file>
```

This stages the file for the next commit. To stage all changes, use:

```
git add .
```

44. What is 'git diff?

`git diff` shows the differences between files or commits. It compares changes in the working directory, staging area, or between two commits.

- **Compare working directory with staging area**: `git diff`
- **Compare staged changes with last commit**: `git diff --staged`
- **Compare two commits**: `git diff <commit1> <commit2>`

45. What is a Git commit hash?

A Git commit hash is a unique identifier (SHA-1) for each commit. It helps to reference and track specific commits in the repository. You can use it to check out or reset to a particular commit.

46. What is a detached HEAD state?

A detached HEAD state occurs when you check out a specific commit rather than a branch. In this state, commits are not attached to any branch, meaning they may be lost if not properly managed.

47. How can you delete a remote Git branch?

To delete a remote branch:

```
git push origin --delete <branch-name>
```

This removes the branch from the remote repository.

48. What is the purpose of git cherry-pick?

Here's the purpose of git cherry-pick in points:

- **Selective Commit Application:** Applies changes from a specific commit to the current branch.
- **No Full Branch Merge:** Useful when you want to incorporate changes from a branch without merging the entire branch.
- **Clean Commit History:** Helps maintain a clean history by including only sel50.ected changes.
- **Bug Fixes:** Ideal for bringing specific bug fixes or features from one branch to another.
- **New Commit:** The changes are applied as a new commit on the current branch, preserving the original commit's content.

49. What does git ls-files do?

The command git ls-files lists all files that are currently tracked by Git in the repository. It displays the files in both the working directory and the staging area. This command helps identify which files are being tracked and are ready to be committed. It does not show untracked files or files ignored by .gitignore.

50. How do you fetch all remote branches?

Run:

```
git fetch --all
```

This command fetches all branches and their latest changes from all remotes without merging them.

Advanced Git Interview Questions for Experienced

51. What is the Git object model?

The Git object model consists of four major types of objects:

- **Blobs:** Store the file data (the contents of files
- Trees: Represent directory structures and references to blobs (files) and other trees

them.

Advanced Git Interview Questions for Experienced

51. What is the Git object model?

The Git object model consists of four major types of objects:

- **Blobs:** Store the file data (the contents of files).
- **Trees:** Represent directory structures and contain references to blobs (files) and other trees (subdirectories).
- **Commits:** Store snapshots of the repository at a particular point in time, along with metadata like author, timestamp, and parent commit references.
- **Tags:** Store references to specific commits, typically used for marking important points in the repository's history, such as releases.

52. Explain 'git rebase' and when you would use each?

git rebase moves or reapplies commits from one branch onto another, creating a linear commit history.

When to use:

- **Clean Commit History:** To avoid merge commits and keep a straight line of history.
- **Sync with Latest Changes:** To update your branch with the latest changes from another branch.
- **Squashing Commits:** To combine multiple commits into one before merging.

53. What is a git hook and how might you use it?

A Git hook is a script that runs automatically at certain points in the Git workflow, like before or after a commit, merge, or push.

Use cases:

- **Pre-commit:** Check code quality or run tests before committing.
- **Post-commit:** Automate tasks like notifications after a commit.
- **Pre-push:** Ensure tests pass before pushing changes.

54. Explain the difference between git reset, git revert, and git checkout?

- **git reset:** Moves HEAD to different commit, potentially changing history.
- **git revert:** Undoes exact commit by making new commit with inverse changes.
- **git checkout:** Switches branches or checks out files from commit, putting you in "detached HEAD" state for direct commits.

55. How do you handle large files with Git?

To handle large files with Git, use Git LFS (Large File Storage). It replaces large files in the repository with pointers, while the actual file content is stored externally.

Steps:

1. Install Git LFS:

```
git lfs install
```

2. Track large files:

```
git lfs track "*.psd"
```

3. Commit and push as usual. Git LFS will handle the large file storage automatically.

56. What is 'bare repository' in Git?

A bare repository in Git is a repository that doesn't have a working directory. It only contains the Git data, such as branches, tags, and commits, without the actual project files. Bare repositories are typically used as remote repositories where other developers push and pull changes.

- No working directory (no actual files).

Used for sharing changes between collaborators (remote).

- Stored typically in the .git folder format.

57. What is branching in Git?

Branching in Git permits forming separate lines of development. It allows users to work on features or fixes separately from the main codebase, facilitating parallel development and simpler integration.

- No working directory (no actual files).

Used for sharing changes between collaborators (remote).

- Stored typically in the .git folder format.

57. What is branching in Git?

Branching in Git permits forming separate lines of development. It allows users to work on features or fixes separately from the main codebase, helping parallel development and simpler integration of changes.

58. What is a Git submodule?

A Git submodule is a repository embedded inside another Git repository. It allows you to keep track of an external repository as part of your project. Submodules are useful when you want to include external libraries or dependencies while keeping their history separate.

59.How can you undo a commit that hasn't been pushed to the remote repository?

We can use git reset to undo a commit:

- To keep changes in the working directory:

```
git reset --soft HEAD~1
```

- To remove changes from both the commit and working directory:

```
git reset --hard HEAD~1
```

60. How do you squash multiple commits into one using Git rebase?

To squash multiple commits into one:

1. Start an interactive rebase:

```
git rebase -i HEAD~<number-of-commits>
```

2. In the editor, change pick to squash (or s) for the commits you want to combine.

3. Save and close the editor to squash the commits.

4. Git will open another editor to combine commit messages. Edit the message and save.

61. How do you reset a commit to a previous commit without losing changes in the working directory?

To reset to a previous commit without losing changes:

```
git reset --soft <commit-hash>
```

62. What is the difference between git reset --hard and git clean -fd?

- **git reset --hard:** Resets the commit history, the staging area, and the working directory to a specific commit, discarding any changes.
- **git clean -fd:** Removes untracked files and directories from the working directory but does not affect the commit history or staged changes.

63. How do you apply a patch from a remote Git repository?

To apply a patch:

1. Fetch the patch from the remote repository:

```
git fetch <remote> <branch>
```

2. Apply the patch:

```
git apply <patch-file>
```

64. How do you configure a Git repository to use a specific user for a particular project?

To set a specific user for a project:

```
git apply <patch-file>
```

64. How do you configure a Git repository to use a specific user for a particular project?

To set a specific user for a project:

```
git config user.name "Your Name"
git config user.email "your.email@example.com"
```

65. What is a Git reflog, and how is it useful?

The Git reflog tracks the history of changes to the HEAD and branches, including changes that are not part of the commit history (e.g., resets, rebase). It allows you to recover lost commits by providing references to previous states.

Use:

```
git reflog
```

66. How do you manage multiple remotes in a Git repository?

We can manage multiple remotes using:

- **Add a new remote:**

```
git remote add <remote-name> <remote-url>
```

- **View remotes:**

```
git remote -v
```

- **Remove a remote:**

```
git remote remove <remote-name>
```

67. How do you configure and use Git hooks for pre-commit checks?

Git hooks allow you to run custom scripts at various points in the Git workflow. For pre-commit checks:

- Go to the .git/hooks directory.
- Rename or copy pre-commit.sample to pre-commit.
- Write your script to check code style, run tests, etc.
- Make the hook script executable:

```
chmod +x .git/hooks/pre-commit
```

68. How do you manage large binary files in Git without using Git LFS?

One way to manage large binary files without using Git LFS is by using Git Large File Storage (LFS), but if you can't use LFS, you can store large files in a separate repository and reference them using Git submodules.

69.How do you fetch changes from multiple remotes in Git?

To fetch changes from all remotes:

```
git fetch --all
```

70. How do you perform a Git bisect to find the commit that introduced a bug?

To perform a Git bisect:

- **Start the bisect process:**

```
git bisect start
```

70. How do you perform a Git bisect to find the commit that introduced a bug?

To perform a Git bisect:

- **Start the bisect process:**

```
git bisect start
```

- **Mark the current commit as bad (where the bug is present):**

```
git bisect bad
```

- **Mark a known good commit (where the bug is not present):**

```
git bisect good <commit-hash>
```

- Git will check out a commit between the "good" and "bad" ones. Test the code to see if the bug is present.
- Repeat the process by marking the commit as bad or good based on whether the bug is present.
- Once the bad commit is found, stop the bisect process:

```
git bisect reset
```

Conclusion

In conclusion, **Git** is a crucial tool for developers, especially when it comes to managing projects and collaborating on code. If you're preparing for an interview, understanding **GitHub interview** questions and knowing your way around common **Git commands interview questions** can greatly improve your chances of success.

Top 70+ Git Interview Questions and Answers - 2025

Next Article
Top 70+ Git Interview Questions and Answers - 2025

Similar Reads

1. Top HR Interview Questions and Answers (2025)
2. DevOps Interview Questions and Answers 2025
3. Top 25 Maven Interview Questions and Answers for 2024
4. Top 25 UI/UX Design Interview Questions in 2025
5. Project Manager Interview Questions and Answers
6. Git Exercises, Practice Questions and Solutions
7. Top 25 Frequently Asked Interview Questions in Technical Rounds
8. GreyOrange Interview Questions
9. Technical Interview Questions
10. Top 10 DevOps Skills to Learn in 2025