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

Q1. What is the difference between Git and SVN?

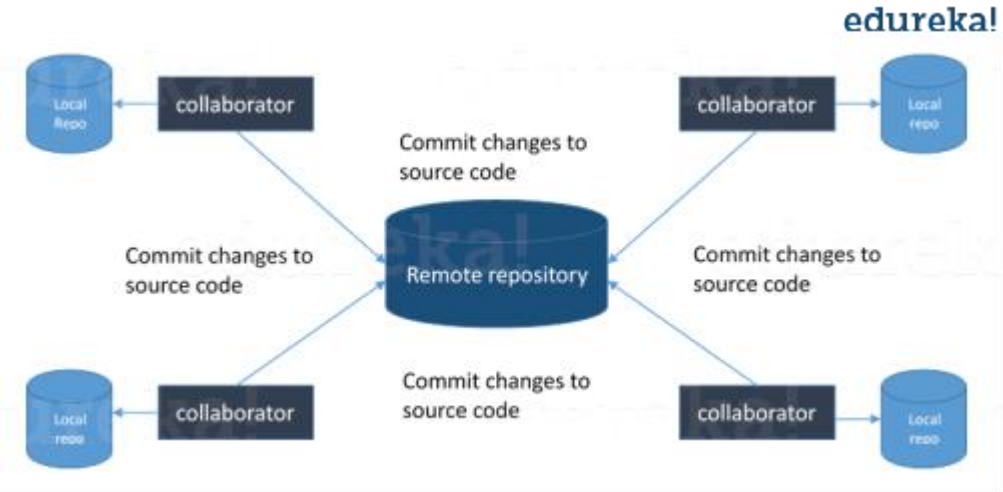
Git vs SVN	
Git	**SVN**
1. Git is a Decentralized Version Control tool	1. SVN is a Centralized Version Control tool
2. It belongs to the 3rd generation of Version Control tools	2. It belongs to the 2nd generation of Version Control tools
3. Clients can clone entire repositories on their local systems	3. Version history is stored on a server-side repository
4. Commits are possible even if offline	4. Only online commits are allowed
5. Push/pull operations are faster	5. Push/pull operations are slower
6. Works are shared automatically by commit	6. Nothing is shared automatically

Q2. What is Git?

I will suggest you to 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 can track changes to a file and allows you to revert back to any particular change.
- Its distributed architecture provides many advantages over other Version Control Systems (VCS) like SVN one major advantage 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 on his hard drive so when there is a server outage all you need for recovery is one of your teammate's local Git repository.

- There is a central cloud repository as well where developers can commit changes and share it with other teammates as you can see in the diagram where all collaborators are commiting changes “Remote repository”.



Now, the next set of Git interview questions will test your experience with Git:

Watch The Course Preview

Q3. What is the command to write a commit message in Git?

Answer to this is pretty straightforward.

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.

Q4. What is ‘bare repository’ in Git?

You are expected to tell the difference between a “working directory” and “bare repository”.

A “bare” repository in Git just contains the version control information 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, where as working directory consist of:

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

Q5. 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.

Q6. In Git how do you revert a commit that has already been pushed and made public?

There can be two answers to 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 push it to the remote repository. This is the most natural way to fix an error. Once you have made necessary changes to the file, commit it to the remote repository for that I will use **git commit -m “commit message”**
- Create a new commit that undoes all changes that were made in the bad commit.to do this I will use a command **git revert <name of bad commit>**

Q7. 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.

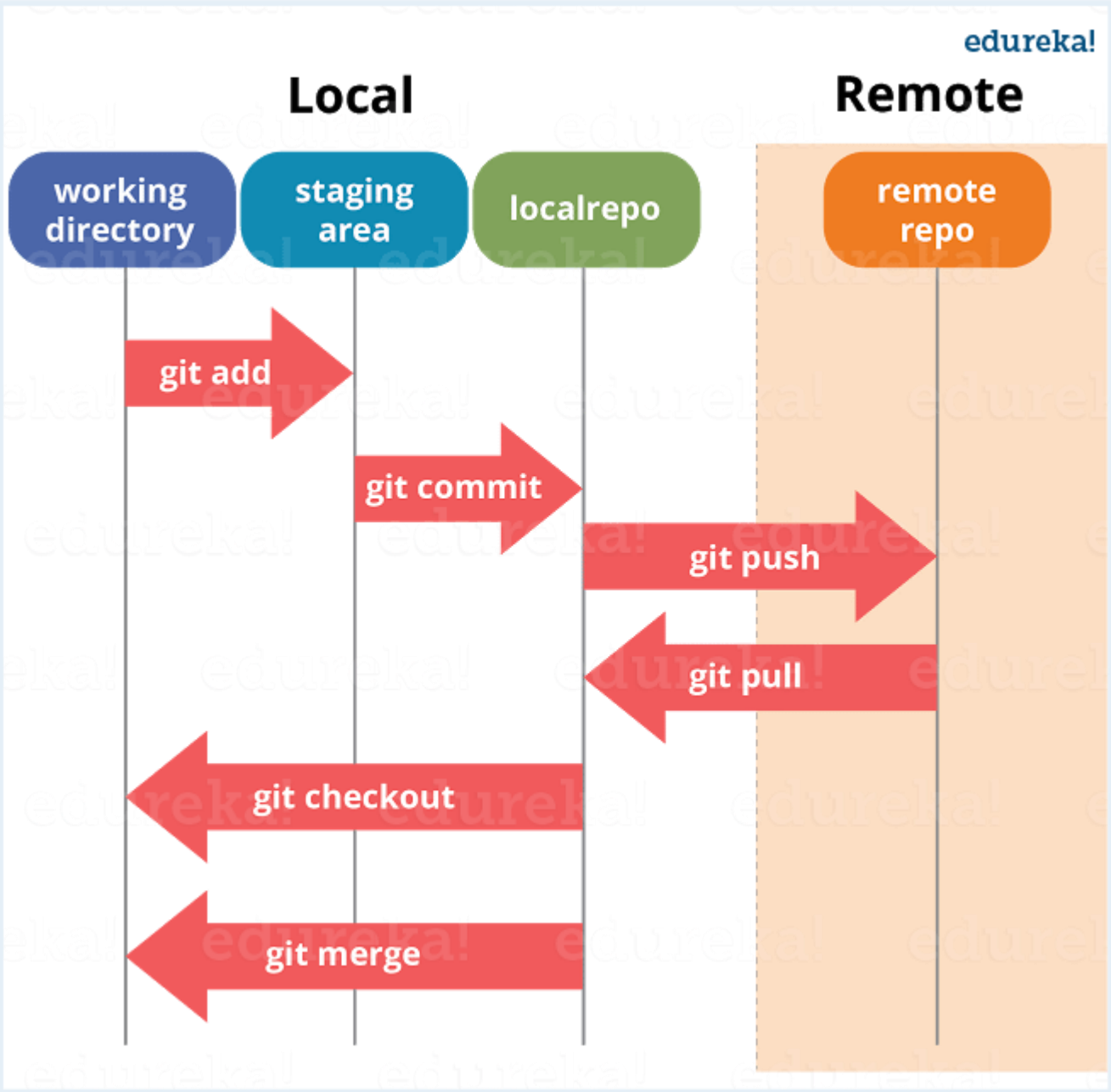
Git fetch is also used for the same purpose but it works in a slightly different way. When you perform a git fetch, it pulls all new commits from the desired branch and stores it in a new branch in your local repository. If you want to reflect these changes in your target branch, git fetch must be followed 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

Q8. What is ‘staging area’ or ‘index’ in Git?

For this answer try to explain the below diagram as you can see:

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.



If your interviewer has good knowledge on Git he/she will dig in deep, so the next set of Git interview questions will be more challenging.

Trending Courses in this category

DevOps Certification Training

Q9. What is Git stash?

According to me you should first explain the need for 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 sometime 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.

Now explain what 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.

Q10. What is Git stash drop?

Begin this answer by saying for what purpose we use 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}**.

Q11. How do you find a list of files that has 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 files 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.

Q12. What is the function of ‘git config’?

First tell why we need ‘**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 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 username.
git config –global user.email "Your E-mail Address": This command will add email id.

Q13. What does commit object contains?

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

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

Q14. How can you create a repository in Git?

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

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

Q15. How do you squash last N commits into a single commit?

There are two options to squash 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})"**

Q16. What is Git bisect? How can you use it to determine the source of a (regression) bug?

I will suggest you to first give a small definition of Git bisect.

Git bisect is used to find the commit that introduced a bug by using binary search. Command for Git bisect is **git bisect <subcommand> <options>**

Now since you have mentioned the command above explain 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.

Q17. How do you configure a Git repository to run code sanity checking tools right before making commits, and preventing them if the test fails?

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

A sanity or smoke test determines whether it is possible and reasonable to continue testing.

Now explain how to achieve this.

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

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

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

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

The Interviewer has not started asking questions on branching yet, so the next set of Git interview questions will be dealing with branching in Git.

Q18. Describe branching strategies you have used?

This question is asked to test your branching experience with Git so, tell them about how you have used branching in your previous job and what purpose does it serves, you can refer the below mention points:

- 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 develop branch, which may have progressed since the release was initiated.

In the end tell them that branching strategies varies from one organization to another so I know basic branching operations like delete, merge, checking out a branch etc..

Q19. 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.

Q20. What is Git rebase and how can it be used to resolve conflicts in a feature branch before merge?

According to me you should start by saying git rebase is a command which will merge another branch into the branch where you are currently working, and move all of the local commits that are ahead of the rebased branch to the top of the history on that branch.

Now, once you have defined Git rebase time for an example to show how it can be used to resolve conflicts in a feature branch before merge.

If a feature branch was created from the master, and since then the master branch has received new commits, Git rebase can be used to move the feature branch to the tip of master. The command effectively will replay the changes made in the feature branch at the tip of master, allowing conflicts to be resolved in the process. When done with care, this will allow the feature branch to be merged into master with relative ease and sometimes as a simple fast-forward operation.

You can also expect some off track questions, so the next question in this Git interview questions blog will be regarding SubGit.

Q21. What is SubGit?

Begin this answer by explaining what is SubGit used for.

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

Now you can 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 existing Subversion repository. You can push to Git or commit to Subversion at your convenience. Synchronization will be done by SubGit.

Define GIT?

Git is a Distributed Version Control system (DVCS). It can track changes to a file and allows you to revert back to any particular change.

Define 'bare repository' in Git?

A "bare" repository in Git it doesn't contain the special .git sub-directory, just contains the version control information and no working files (no tree). Instead, it contains all the contents of the git sub-directory directly in the main directory itself, where as working directory consist of:

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

What are the different ways you can refer to a commit?

In Git each commit is given a unique hash. These hashes can be used to identify the corresponding commits in various scenarios (such as while trying to checkout a particular state of the code using the git checkout {hash} command).

Additionally, Git also maintains a number of aliases to certain commits, known as refs. Also, every tag that you create in the repository effectively becomes a ref (and that is exactly why you can use tags instead of commit hashes in various git commands). Git also maintains a number of special aliases that change based on the state of the repository, such as HEAD, FETCH_HEAD, MERGE_HEAD, etc.

Git also allows commits to be referred as relative to one another. For example, HEAD~1 refers to the commit parent to HEAD, HEAD~2 refers to the grandparent of HEAD, and so on. In case of merge commits, where the commit has two parents, ^ can be used to select one of the two parents, e.g. HEAD^2 can be used to follow the second parent.

And finally, refspecs. These are used to map local and remote branches together. However, these can be used to refer to commits that reside on remote branches allowing one to control and manipulate them from a local Git environment.

What is a conflict in git and how can it be resolved?

A conflict arises when more than one commit that has to be merged has some change in the same place or same line of code. Git will not be able to predict which change should take precedence. This is a git conflict.

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 merge, so it sets the parents of the commit correctly.

What does commit object contains?

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

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

Reference to parent commit objects

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

What is SubGit?

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

What is the HEAD in GIT ?

AHEAD is a reference to the present looked at conferring.

It is a representative reference to the branch that we have looked at.

At any given time, one head is chosen as the 'present head' this head is otherwise called HEAD (dependably in capitalized).

What are Git Design objectives?

Distributed workflow (decentralised)

Easy merging (merge deemed more frequent than commit)

Integrity (protection against accidental/malicious corruptions)

Speed & scalability

What is Version Control with Git?

Version control is better than mailing files back and forth because:

- It's is not impossible to coincidentally overwrite or overlook someone's changes: whenever there's a conflict between one person's work and another's, the version control system automatically notifies users.
- If people are having some questions to ask, they will maintain the records what changes they have made.
- Nothing that is committed to version control is ever lost. it's always possible to go back in time to know exactly who wrote what on a particular day, or what version of a program was used to generate a particular set of results. This means it can be used like the undo feature in an editor, and since all previous versions of files are saved.

Git is one of many version control systems. It is more complex than some alternatives, but it is widely used, both because it's easy to set up and because of a hosting site called GitHub, which we will get to later.

What are the main benefits of GIT ?

Distributed System: GIT is a Distributed Version Control System (DVCS). So you can keep your private work in adaptation control yet totally escaped others. You can work disconnected too.

• Flexible Workflow:GIT enables you to make your own work process. You can utilize the procedure that is appropriate for your venture. You can go for brought together or ace slave or some other work process.

• Fast: GIT is quick when contrasted with other form control frameworks.

• Data Integrity: Since GIT utilizes SHA1, information isn't less demanding to degenerate.

• Free: It is free for individual utilize. Such huge numbers of beginners utilize it for their underlying activities. It likewise works exceptionally well with substantial size task.

• Collaboration: GIT is anything but difficult to use for ventures in which joint effort is required. Numerous prevalent open source programming over the globe utilize GIT