Question: **What is Git?**

Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people.
It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.

As a distributed revision control system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.
By far, the most widely used modern version control system in the world today is Git. Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvald.

Git is an example of a Distributed Version Control System, In Git, every developer's working copy of the code is also a repository that can contain the full history of
all changes.

Question: **What Are Benefits Of GIT?**

Here are some of the advantages of using Git

- Ease of use
- Data redundancy and replication
- High availability
- Superior disk utilization and network performance
- Only one .git directory per repository
- Collaboration friendly
- Any kind of projects from large to small scale can use GIT

Question: **What Is Repository In GIT?**

The purpose of Git is to manage a project, or a set of files, as they change over time. Git stores this information in a data structure called a repository. A git **repository** contains, among other things, the following:

- A set of **commit objects**.
- A set of references to commit objects, called **heads**.
- The Git repository is stored in the same directory as the project itself, in a subdirectory called .git.

Note differences from central-repository systems like CVS or Subversion:

There is only one .git directory, in the root directory of the project.
The repository is stored in files alongside the project. There is no central server repository.

Question: **What Is Staging Area In GIT?**

Staging is a step before the commit process in git. That is, a commit in git is performed in two steps:

-Staging and
-Actual commit
As long as a change set is in the staging area, git allows you to edit it as you like (replace staged files with other versions of staged files, remove changes from staging, etc.)

Question: **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 a bit 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 the git stash command. Stashing takes the dirty state of 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.

Question: **How To Revert Commit In GIT?**

Given one or more existing commits, revert the changes that the related patches introduce,and record some new commits that record them. This requires your working tree to be clean (no modifications from the HEAD commit).

git-revert - Revert some existing commits
SYNOPSIS
git revert [--[no-]edit] [-n] [-m parent-number] [-s] [-S[<keyid>]] <commit>…
git revert --continue
git revert --quit

git revert –abort

Question: **How To Delete Remote Repository In GIT?**

Use the git remote rm command to remove a remote URL from your repository.
The git remote rm command takes one argument:
A remote name, for example, destination

Question: **What Is GIT Stash Drop?**

In case we do not need a specific stash, we use git stash drop command to remove it from the list of stashes.
By default, this command removes to latest added stash
To remove a specific stash we specify as argument in the git stash drop <stashname>command.

Question: **What Is Difference Between GIT and Subversion?**

Here is a summary of Differences between GIT and Subversion
Git is a distributed VCS; SVN is a non-distributed VCS.
Git has a centralized server and repository; SVN does not have a centralized server or repository.
The content in Git is stored as metadata; SVN stores files of content.
Git branches are easier to work with than SVN branches.
Git does not have the global revision number feature like SVN has.
Git has better content protection than SVN.
Git was developed for Linux kernel by Linus Torvalds; SVN was developed by CollabNet, Inc.
Git is distributed under GNU, and its maintenance overseen by Junio Hamano; Apache Subversion, or SVN, is distributed under the open source license.

Question: **What Is Difference Between GIT Fetch & GIT Pull?**

GIT fetch – It downloads only the new data from the remote repository and does not integrate any of the downloaded data into your working files. Providing a view of the data is all it does.
GIT pull – It downloads as well as merges the data from the remote repository into the local working files.
This may also lead to merging conflicts if the user's local changes are not yet committed.
Using the "GIT stash" command hides the local changes.

Question: **What is Git fork? How to create tag?**

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.
A fork is really a Github (not Git) construct to store a clone of the repo in your user account. As a clone, it will contain all the branches in the main repo at the time you made the fork.

Create Tag:
Click the releases link on our repository page.
Click on Create a new release or Draft a new release.
Fill out the form fields, then click Publish release at the bottom.
After you create your tag on GitHub, you might want to fetch it into your local repository too: git fetch.

Question: **What is difference between fork and branch?**

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.
A fork is really a Github (not Git) construct to store a clone of the repo in your user account.
As a clone, it will contain all the branches in the main repo at the time you made the fork.

Question: **What Is Cherry Picking In GIT?**

Cherry picking in git means to choose a commit from one branch and apply it onto another.
This is in contrast with other ways such as merge and rebase which normally applies many commits onto a another branch.

Make sure you are on the branch you want apply the commit to. git checkout master Execute the following:
git cherry-pick <commit-hash>

Question: **What Language GIT Is Written In?**

Much of Git is written in C, along with some BASH scripts for UI wrappers and other bits.

Question: **How To Rebase Master In GIT?**

Rebasing is the process of moving a branch to a new base commit. The golden rule of git rebase is to never use it on public branches.
The only way to synchronize the two master branches is to merge them back together,resulting in an extra merge commit and two sets of commits that contain the same changes.

Question: **What is 'head' in git and how many heads can be created in a repository?**

There can be any number of heads in a GIT repository. By default there is one head known as HEAD in each repository in GIT.
HEAD is a ref (reference) to the currently checked out commit. In normal states, it's actually a symbolic ref to the branch user has checked out.
if you look at the contents of .**git**/**HEAD** you'll see something like "ref: refs/**heads**/master".
The branch itself is a reference to the commit at the tip of the branch.

Question: **Name some GIT commands and also explain their functions?**

Here are some most important GIT commands
GIT diff – It shows the changes between commits, commits and working tree.
GIT status – It shows the difference between working directories and index.
GIT stash applies – It is used to bring back the saved changes on the working directory.
GIT rm – It removes the files from the staging area and also of the disk.
GIT log – It is used to find the specific commit in the history.

GIT add – It adds file changes in the existing directory to the index.
GIT reset – It is used to reset the index and as well as the working directory to the state of the last commit.
GIT checkout – It is used to update the directories of the working tree with those from another branch without merging.
GIT Is tree – It represents a tree object including the mode and the name of each item.
GIT instaweb – It automatically directs a web browser and runs the web server with an interface into your local repository.

Question: **What is a "conflict" in GIT and how is it resolved?**

When a commit that has to be merged has some changes in one place, which also has the changes of current commit, then the conflict arises.

The GIT will not be able to predict which change will take the precedence. In order to resolve the conflict in GIT: we have to edit the files to fix the conflicting changes and then add the resolved files by running the "GIT add" command; later on, to commit the repaired merge run the "GIT commit" command. GIT identifies the position and sets the parents of the commit correctly.

Question: **How To Migrate From Subversion To GIT?**

SubGIT is a tool for smooth and stress-free subversion to GIT migration and also a solution for a company-wide subversion to GIT migration that is:

It allows to make use of all GIT and subversion features.
It provides genuine stress-free migration experience.
It doesn't require any change in the infrastructure that is already placed.
It is considered to be much better than GIT-SVN

Question: **What Is Index In GIT?**

The index is a single, large, binary file in under .git folder, which lists all files in the current branch, their sha1 checksums, time stamps and the file name.

Before completing the commits, it is formatted and reviewed in an intermediate area known as Index also known as the staging area.

Question: **What is a bare Git repository?**

A bare Git repository is a repository that is created without a Working Tree.
git init –bare.

Question: **How do you revert a commit that has already been pushed and made public??**

One or more commits can be reverted through the use of *git revert*. This command, in essence, creates a new commit with patches that cancel out the changes introduced in specific commits.

In case the commit that needs to be reverted has already been published or changing the repository history is not an option, *git revert* can be used to revert commits. Running the following command will revert the last two commits:
git revert HEAD~2..HEAD

Alternatively, one can always checkout the state of a particular commit from the past, and commit it.

Question: **How do you squash last N commits into a single commit?**

Squashing multiple commits into a single commit will overwrite history, and should be done with caution. However, this is useful when working in feature branches.
To squash the last N commits of the current branch, run the following command (with {N} replaced with the number of commits that you want to squash):
git rebase -i HEAD~{N}
Upon running this command, an editor will open with a list of these N commit messages,one per line.
Each of these lines will begin with the word “pick”. Replacing “pick” with “squash” or “s” will tell Git to combine the commit with the commit before it.
To combine all N commits into one, set every commit in the list to be squash except the first one.

Upon exiting the editor, and if no conflict arises, *git rebase* will allow you to create a new commit message for the new combined commit.

Question: **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.

Question: **How To Setup A Script To Run Every Time a Repository Receives New Commits Through Push?**

To configure a script to run every time a repository receives new commits through push,one needs to define either a pre-receive, update, or a post-receive hook depending on when exactly the script needs to be triggered.
Pre-receive hook in the destination repository is invoked when commits are pushed to it.
Any script bound to this hook will be executed before any references are updated.
This is a useful hook to run scripts that help enforce development policies.
Update hook works in a similar manner to pre-receive hook, and is also triggered before any updates are actually made.
However, the update hook is called once for every commit that has been pushed to the destination repository.
Finally, post-receive hook in the repository is invoked after the updates have been accepted into the destination repository.
This is an ideal place to configure simple deployment scripts, invoke some continuous integration systems, dispatch notification emails to repository maintainers, etc.
Hooks are local to every Git repository and are not versioned. Scripts can either be created within the hooks directory inside the ".git" directory, or

they can be created elsewhere and links to those scripts can be placed within the directory.

Question: **What Is Commit Hash?**

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.

Question: **What Is Conflict In GIT?**

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.

Question: **What are git hooks?**

Git hooks are scripts that can run automatically on the occurrence of an event in a Git repository. These are used for automation of workflow in GIT.

Git hooks also help in customizing the internal behavior of GIT. These are generally used for enforcing a GIT commit policy.

Question: **What Are Disadvantages Of GIT?**

GIT has very few disadvantages. These are the scenarios when GIT is difficult to use.
Some of these are:

Binary Files: If we have a lot binary files (non-text) in our project, then GIT becomes very slow. E.g. Projects with a lot of images or Word documents.
Steep Learning Curve: It takes some time for a newcomer to learn GIT.

Some of the GIT commands are non-intuitive to a fresher.
Slow remote speed: Sometimes the use of remote repositories in slow due to network latency. Still GIT is better than other VCS in speed.

Question: **What is stored inside a commit object in GIT?**

GIT commit object contains following information:

SHA1 name: A 40 character string to identify a commit
Files: List of files that represent the state of a project at a specific point of time
Reference: Any reference to parent commit objects

Question: **What Is GIT reset command?**

Git reset command is used to reset current HEAD to a specific state. By default it reverses the action of git add command. So we use git reset command to undo the changes of git add command. Reference: Any reference to parent commit objects.

Question: **How GIT protects the code in a repository**?

GIT is made very secure since it contains the source code of an organization. All the objects in a GIT repository are encrypted with a hashing algorithm called SHA1.
This algorithm is quite strong and fast. It protects source code and other contents of repository against the possible malicious attacks.
This algorithm also maintains the integrity of GIT repository by protecting the change history against accidental changes.