**Q.What isGIT?**  
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.

**Q. 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.

**Q. 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” before git commit –a if new files need to be committed for the first time.

**Q.What is the difference between GIT and SVN?**  
The difference between GIT and SVN is  
1. 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.  
2. GIT does not support ‘commits’ across multiple branches or tags.  Subversion allows the creation of folders at any location in the repository layout.  
3. Gits are unchangeable, while Subversion allows committers to treat a tag as a branch and to create multiple revisions under a tag root.

**Q. What are the advantages of using GIT?**  
1. Data redundancy and replication  
2. High availability  
3. Only one.git directory per repository  
4. Superior disk utilization and network performance  
5. Collaboration friendly  
6. Any sort of projects can use GIT

**Q. 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.

**Q. What is the function of ‘GIT PUSH’ in GIT?**  
‘GIT PUSH’ updates remote refs along with associated objects.

**Q. 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.

**Q. 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’.

**Q. 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.

**Q. 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.

**Q. 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

**Q. 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.

**Q. What is the function of ‘git config’?**  
The ‘git config’ command is a convenient way to set configuration options for your [Git installation](https://mindmajix.com/installation-of-git" \t "_blank).  Behaviour of a repository, user info, preferences etc. can be defined through this command.

**Q. What does commit object contain?**  
1. A set of files, representing the state of a project at a given point of time  
2. Reference to parent commit objects  
3. An SHAI name, a 40 character string that uniquely identifies the commit object.

**Q. 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.

**Q. 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.

**Q. 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.

**Q. 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.

**Q. 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”.

**Q. 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.

**Q. 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.

**Q. 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]”.

**Q. What is another option for merging in git?**  
“Rebasing” is an alternative to merging in git.

**Q. What is the syntax for “Rebasing” in Git?**  
The syntax used for rebase is “[git rebase](https://mindmajix.com/about-git-rebase" \t "_blank) [new-commit] “

**Q. 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.

**Q. 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.

**Q. Mention some of the best graphical GIT client for LINUX?**  
Some of the best GIT client for LINUX is  
1. Git Cola  
2. Git-g  
3. Smart git  
4. Giggle  
5. Git GUI  
6. qGit

**Q. 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.

**Q. What is the function of ‘git diff ’ in git?**  
‘git diff ’ shows the changes between commits, commit and working tree etc.

**Q. 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.

**Q. 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.

**Q. 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.

**Q. What is the function of ‘git rm’?**  
To remove the file from the staging area and also off your disk ‘git rm’ is used.

**Q. 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.

**Q. What is the use of ‘git log’?**  
To find specific commits in your project history- by author, date, content or [history ‘git log’](https://mindmajix.com/git-rewriting-history) is used.

**Q.What is ‘git add’ is used for?**  
‘git add’ adds file changes in your existing directory to your index.

**Q. 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.

**Q. 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.

**Q. How git instaweb is used?**  
‘Git Instaweb’ automatically directs a web browser and runs webserver with an interface into your local repository.

**Q. 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.

**Q. 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.

**Q. 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.

**Q. Why is it advisable to create an additional commit rather than amending an existing commit?**  
There are couple of reason  
1. 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.  
2. Abusing “git commit- amend” can cause a small commit to grow and acquire unrelated changes.

**Q. 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.

**Q. 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 anew.

**Q. 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.

**Q. How do you find a list of files that has changed in a particular commit?**  
git diff-tree -r {hash}  
Given the commit hash, this will list all the files that were changed or added in that commit. The -r flag makes the command list individual files, rather than collapsing them into root directory names only.  
The output will also include some extra information, which can be easily suppressed by including a couple of flags:  
git diff-tree –no-commit-id –name-only -r {hash}  
Here –no-commit-id will supress the commit hashes from appearing in the output, and –name-only will only print the file names, instead of their paths.

**Q. How do you 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.

**Q. What is git bisect? How can you use it to determine the source of a (regression) bug?**  
Git provides a rather efficient mechanism to find bad commits. Instead of making the user try out every single commit to find out the first one that introduced some particular issue into the code, git bisect allows the user to perform a sort of binary search on the entire history of a repository.  
By issuing the command git bisect start, the repository enters bisect mode. After this, all you have to do is identify a bad and a good commit:  
git bisect bad # marks the current version as bad  
git bisect good {hash or tag} # marks the given hash or tag as good, ideally of some earlier commit  
Once this is done, Git will then have a range of commits that it needs to explore. At every step, it will checkout a certain commit from this range, and require you to identify it as good or bad. After which the range will be effectively halved, and the whole search will require a lot less number of steps than the actual number of commits involved in the range. Once the first bad commit has been found, or the bisect mode needs to be ended, the following command can be used to exit the mode and reset the bisection state:  
git bisect reset

**Q. 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 plumbing commands](https://mindmajix.com/git-plumbing-commands" \t "_blank)). 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.

**Q. What is git rebase and how can it be used to resolve conflicts in a feature branch before merge?**  
In simple words, git rebase allows one to move the first commit of a branch to a new starting location. For example, if a feature branch was created from 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.

**Q. How do you configure a Git repository to run code sanity checking tools right before making commits, and preventing them if the test fails?**  
This can be done with a simple script bound 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. For example, the following 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  
… checks to see if any .go file that is about to be commited 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.

**Q. 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.

**Q. 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>**

**Q. 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.

**Q. 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>**

**Q. 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.

* 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.

**Q.   Name a few Git repository hosting services**

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

**Q.   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.