# GIT ASSESSMENT-1

1.What is a git stash list?

Ans.

\*The Git stash list command will pull up a list of your repository’s stashes. Git

will display all of your stashes and a corresponding stash index.

git stash list

\*This the command for git stash list.

\*The Git stash list command will pull up a list of your repository’s stashes. Git

will display all of your stashes and a corresponding stash index.

\*Now, if you wish to view the contents of a specific stash, you can run the Git

stash show command followed by **stash@** and the desired index.

git stash show stash@<index>

\*we’re going to assume you’re working with your most recent stash, but you

can always utilize this syntax with other Git stash commands as needed.

2. How do you get a list of all the files that have been updated in a given

commit?

Ans.

To find out which files changed in a given commit, use the git log –

raw command. It's the fastest and simplest way to get insight into which

files a commit affects. The git log command is underutilized in general,

largely because it has so many formatting options, and many users get

overwhelmed by too many choices and, in some cases, unclear

documentation.

\*The log mechanism in Git is surprisingly flexible, though, and the –

raw option provides a log of commits in your current branch, plus a list

of each file that had changes made to it.

3. What is a Git merge conflict?

Ans.

Merge conflicts happen when you merge branches that have competing

commits, and Git needs your help to decide which changes to

incorporate in the final merge. Git can often resolve differences between

branches and merge them automatically. Usually, the changes are on

different lines, or even in different files, which makes the merge simple

for computers to understand. However, sometimes there are competing

changes that Git can't resolve without your help. Often, merge conflicts

happen when people make different changes to the same line of the

same file, or when one person edits a file and another person deletes

the same file. You must resolve all merge conflicts before you can merge

a pull request on GitHub. If you have a merge conflict between the

compare branch and base branch in your pull request, you can view a list

of the files with conflicting changes above the **Merge pull**

**request** button. The **Merge pull request** button is deactivated until

you've resolved all conflicts between the compare branch and base

branch.

4.How do you distinguish between git fetch and git pull? How do you

differentiate between Git Merge and Git Rebase?

Ans

Difference between Git fetch and pull

The key difference between git fetch and pull is that git pull copies

changes from a remote repository directly into your working directory,

while git fetch does not. The git fetch command only copies changes into

your local Git repo. The git pull command does both.

To really understand the difference between pull and fetch, you must

know how a Git installation is structured.

On a user’s workstation, a Git installation includes the following items:

* The local Git repository where the history of all commits across all
* branches are maintained.
* A working directory where a developer actively edits and updates
* files that Git tracks.

### Benefits of the git pull command

If a developer finds out that there are new, updated files on a remote

repository like GitHub, they will likely want to copy those changes from

GitHub to both their local repository and into their working directory.

This is what the *git pull* command does. The *git pull* command updates

### Benefits of the git fetch command

If you are actively working on files tracked by Git, but you still want to

update your local repository with the latest changes from a remote

repository, use the git fetch command.

A git fetch updates your local repo with all of the latest changes from a

remote repo but doesn’t make any changes to your local workspace.

The benefit of the git fetch vs git pull is that a fetch enables you to

continue editing files in your local working directory without having to

merge your code with updates from the remote repo.

With a git fetch, you can finish editing files locally, commit your files and

then do a git merge to synchronize your updates with the fetched files.

This brings you up to date with the updates the fetch pulled down from

the remote machine.

## Difference between **git merge and git rebase**

It is a most common puzzling question for the git user's that when to use

merge command and when to use rebase. Both commands are similar,

and both are used to merge the commits made by the different branches

of a repository.

Rebasing is not recommended in a shared branch because the rebasing

process will create inconsistent repositories. For individuals, rebasing can

be more useful than merging. If you want to see the complete history, you

should use the merge. Merge tracks the entire history of commits, while

rebase rewrites a new one.

Git rebase commands said as an alternative of git merge. However, they

have some key differences:

|  |  |
| --- | --- |
| **Git Merge** | **Git Rebase** |
| Merging creates a final commit at merging. | Git rebase does not create any commit at rebasing. |
| It merges all commits as a single commit. | It creates a linear track of commits. |
| It creates a graphical history that might be a bit complex to understand. | It creates a linear history that can be easily understood. |
| It is safe to merge two branches. | Git "rebase" deals with the severe operation. |
| Merging can be performed on both public and private branches. | It is the wrong choice to use rebasing on public branches. |
| Merging integrates the content of the feature branch with the master branch. So, the master branch is changed, and feature branch history remains consistence. | Rebasing of the master branch may affect the feature branch. |
| Merging preserves history. | Rebasing rewrites history. |
| Git merge presents all conflicts at once. | Git rebase presents conflicts one by one. |

6. What command uploads any GitHub repository to your computer using the git command?

Ans.

Files that you add to a repository via a browser are limited to 25 MB per

file. You can add larger files, up to 100 MB each, via the command line.

For more information, see "[Adding a file to a repository using the command line](https://docs.github.com/en/repositories/working-with-files/managing-files/adding-a-file-to-a-repository#adding-a-file-to-a-repository-using-the-command-line)."

To add files larger than 100 MB, you must use Git Large

File Storage. For more information, see "[About large files on GitHub](https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-large-files-on-github)."

1. On GitHub.com, navigate to the main page of the repository.

2.Above the list of files, select the **Add file** dropdown menu and

click **Upload files**. Alternatively, you can drag and drop files into your browser.

3.To select the files you want to upload, drag and drop the file or

folder, or click **choose your files**.

4.In the "Commit message" field, type a short, meaningful commit

message that describes the change you made to the file. You can

attribute the commit to more than one author in the commit message.

For more information, see "[Creating a commit with multiple authors](https://docs.github.com/en/pull-requests/committing-changes-to-your-project/creating-and-editing-commits/creating-a-commit-with-multiple-authors)."

5.Below the commit message fields, decide whether to add your commit

to the current branch or to a new branch. If your current branch is the

default branch, you should choose to create a new branch for your

commit and then create a pull request. For more information, see

"[Creating a pull request](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request)."

6. Click **Propose changes**.

7Q. How do you make a Git repository?

Ans.

You can store a variety of projects in GitHub repositories, including

open source projects. With open source projects, you can share code to

make better, more reliable software. You can use repositories to

collaborate with others and track your work. For more information, see

"[About repositories](https://docs.github.com/en/repositories/creating-and-managing-repositories/about-repositories)." To learn more about open source projects,

visit [OpenSource.org](https://opensource.org/about).

1.In the upper-right corner of any page, use the  drop-down menu, and

select **New repository**.

2.Type a short, memorable name for your repository. For example,

"hello-world".

3.Optionally, add a description of your repository. For example, "My first

repository on GitHub."

4. Choose a repository visibility. For more information, see "[About repositories](https://docs.github.com/en/repositories/creating-and-managing-repositories/about-repositories#about-repository-visibility)."

5.Select **Initialize this repository with a README**.

6.Click **Create repository**.

8Q. What is Git, and how does it work?

Ans.

Git is a version control system used for tracking changes in computer

files. It is generally used for source code management in software

development.

* Git is used to tracking changes in the source code
* The distributed version control tool is used for source code management
* It allows multiple developers to work together
* It supports non-linear development through its thousands of parallel branches

## **Features of Git**

* Tracks history
* Free and open source
* Supports non-linear development
* Creates backups
* Scalable
* Supports collaboration
* Branching is easier
* Distributed development