# **GIT Interview Questions**

1. ***What is GIT and why do we use it?***

**GIT - Global Information Tracker**

Git is the most commonly used version control system. Git tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to. Git also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

 It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development.

***2.What is VCS and what are different types of VCS? Compare their pros***

***and cons.***

A Version Control System (VCS) is a tool that helps software developers keep track of how their software development projects - desktop applications, mobile apps change over time.

It was created to allow developers a convenient way to create, manage, and share those versions. It allows them to have control over managing the versions of their code. Version control systems also enable collaboration within a team of software developers, without losing or overwriting anyone's work.

**The types of VCS are:**

**Local Version Control System**(A local version control system is a local database located on your local computer, in which every file change is stored as a patch. Every patch set contains only the changes made to the file since its last version)

**Centralized Version Control System**(A centralized version control system has a single server that contains all the file versions. This enables multiple clients to simultaneously access files on the server, pull them to their local computer or push them onto the server from their local computer)

**Distributed Version Control System(**With distributed version control systems, clients don’t just check out the latest snapshot of the files from the server, they fully mirror the repository, including its full history. ***)***

1. ***What are the advantages that GIT provides?***

Good distributed model as each developer gets a local repository with a full history of commits which makes git fast compared to other VCs.

Branching capabilities and merging are easy (as they are cheap), good data integrity.

They are free and open-source we can easily download the source code and performs changes to it. They can handle larger projects efficiently.

The push/pull operations are faster with a simple They save time and developers can fetch and create pull requests without switching.

Data redundancy and replications. Add ons can be written in many languages.

They have good and faster network performance and superior disk utilization and they think about its data like a sequence of snapshots.

The object model is very simple and minimizes push/pull data transfers.

***4.What do you know about GIT branching?***

 In Git, branches are a part of your everyday development process. Git branches are effectively a pointer to a snapshot of your changes. A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process. You can think of them as a way to request a brand new working directory, staging area, and project history. New commits are recorded in the history for the current branch, which results in a fork in the history of the project. The git branch command lets you create, list, rename, and delete branches. It doesn’t let you switch between branches or put a forked history back together again. For this reason, git branch is tightly integrated with the [git checkout](https://www.atlassian.com/git/tutorials/using-branches/git-checkout) and [git merge](https://www.atlassian.com/git/tutorials/using-branches/git-merge) commands.

1. ***What is merging in GIT?***

Merging is Git's way of putting a forked history back together again. The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch.In the most frequent use cases, git merge is used to combine two branches.

***6.What is GitHub/BitBucket/GitLab and what are their differences .compared to GIT?***

***What is git?***

What is **GITHUB**?

Github ls one of the first Git hosting platforms. The open source community started to use it for code sharing. (A lot of people are worried that Github will lose its open source roots so many are looking at alternatives like Gitlab.)

What is **GITLAB**?

Similar to Github, [Gitlab](https://about.gitlab.com/) is a git based repository hosting platform. From the beginning, Gitlab wanted to distinguish itself from Github, so it created a single product for the entire DevOps lifecycle. In Gitlab, tools like Issue trackers, continuous integration and continuous delivery are part of the product. Gitlab provides a single interface to the whole DevOps cycle.

What is **BITBUCKET**?

[BitBucket](https://bitbucket.org/product) is another online source code hosting service. During that time it worked only with Mercurial (a free distributed version control system), but it also has been using Git since October 2011 after being acquired by the Atlassian. It had its own advantages since Atlassian develops mainstream software tools like Jira, Trello and Confluence. Having great integration with such tools has been a great advantage for BitBucket.

If you are going to work only on open source projects, Github would be a good choice since one can upload their own projects and it also acts like an online portfolio. While GitLab can be used if you are an enterprise. It has free private and public repositories which doesn't restrict on the number of users. If you are someone who wants to host multiple repositories and work with many colleagues, then GitLab might be a good choice for you.

**BitBucke**t is the most popular among many organizations because of its user interface. It also has mercurial support. If you are an organization and you want to host your project, BitBucket would be a great choice.

1. ***What is SSH Key and why do we use it?***

An SSH key is **an access credential for the SSH (secure shell) network protocol**. This authenticated and encrypted secure network protocol is used for remote communication between machines on an unsecured open network. SSH is used for remote file transfer, network management, and remote operating system access.

***8.What is a repository in Git?***

A Git repository tracks and saves the history of all changes made to the files in a Git project. It saves this data in a directory called .git, also known as the repository folder.

Git uses a version control system to track all changes made to the project and save them in the repository. Users can then delete or copy existing repositories or create new ones for ongoing projects.

There are two types of Git repositories, based on user permissions:

**Bare Repositories**

Software development teams use bare repositories to share changes made by team members. Individual users aren't allowed to modify or create new versions of the repository.

**Non-Bare Repositories**

With non-bare repositories, users can modify the existing repository and create new versions. By default, the cloning process creates a non-bare repository.

**How to Get a Git Repository(2 ways)**

Turning an existing directory into a Git repository (initializing). **or**

Cloning a Git repository from an existing project.

1. ***How do you push codes from a new branch or from an existing branch?***

**from a new branch:**

git push -u origin [branch]: Useful when pushing a new branch, this creates an upstream tracking branch with a lasting relationship to your local branch

**from an existing branch:**

git push -> uploads all local branch commits to the corresponding remote branch. Updates the remote branch with local commits.

***10.What is Pull Request?***

Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

(also referred to as a merge request – is an event that takes place in software development when a contributor/developer is ready to begin the process of merging new code changes with the main project repository.)

How do pull requests work?

TLDR

Find a project you want to contribute to.

Fork it.

Clone it to your local system.

Make a new branch.

Make your changes.

Push it back to your repo.

Click the Compare & pull request button.

Click Create pull request to open a new pull request.

1. ***What is merge conflict and how to resolve it?***

A merge conflict is an event that takes place when Git is unable to automatically resolve differences in code between two commits. Git can merge the changes automatically only if the commits are on different lines or branches.

Let’s assume there are two developers: Developer A and Developer B. Both of them pull the same code file from the remote repository and try to make various amendments in that file. After making the changes, Developer A pushes the file back to the remote repository from his local repository. Now, when Developer B tries to push that file after making the changes from his end, he is unable to do so, as the file has already been changed in the remote repository.

To prevent such conflicts, developers work in separate isolated branches. The Git merge command combines separate branches and resolves any conflicting edits.

Question 11 I also remember Akin in class showed us , how to resolve Merge conflict on GITHUB, it was just by clicking on the

“resolve conflict” button on Git

A merge conflict is an event that takes place when Git is unable to automatically resolve differences in code between two commits. Git can merge the changes automatically only if the commits are on different lines or branches.

Conflict can be resolved by 2 ways for now:

First way, it is in GitHub we can click on "resolve conflict"

Second way; it is $ git pull on terminal on intellij and than $ git merge branchName

***12.What are some Linux commands you use mostly?***

1) pwd -> print working direstory. it tells you where you are

2) ls -> list items I have in the current directory

ls -a (means list all)shows all content including hodden folders

MAC USERS shortcuts to see your hidden folders in Finder;

command + shift + .

3) clear -> clear my working terminal

4) mkdir folderName ->(make directory) creating a new folder

5) rmdir (remove directory)

6) cd .. -> change directory moves one step back (goes to parent)

7) cd ~ ->

8) cd FolderName rmdrclear

9) cd / - moves to the root directory

10) touch file.extension -> creates an empty file

11) rm FileName removes file

12) echo TEXT >> file.extension (writes text in a file)

13) cat file.extention - showes whatever text is inside the file.extension

14) touch .notes.txt -> creates hidden file(notes.tt)

15) open file.txt - opens file.txt