# Interview Questions

1. **What is Git**?
   1. Git is a distributed version control system (VCS) that is widely used in software development to track changes in source code during the development of a project. It was created by Linus Torvalds in 2005 and has since become the de facto standard for version control.
2. **What are key concepts and features of GIT**?
   1. Here are some key concepts and features of Git:
   2. **Version Control System (VCS)**: Git is a version control system that allows multiple developers to collaborate on a project. It keeps track of changes made to files over time, allowing you to revert to previous states, compare changes, and collaborate with others efficiently.
   3. **Distributed System**: Git is a distributed version control system, meaning that each developer has a complete copy of the entire repository, including its history. This allows developers to work offline and makes the system more robust.
   4. **Branching and Merging**: Git provides powerful branching and merging capabilities. Developers can create branches to work on new features or bug fixes independently. Changes made in one branch can be merged back into the main codebase when ready.
   5. **Commit**: A commit in Git represents a snapshot of the project at a specific point in time. Each commit has a unique identifier, a commit message describing the changes made, and references to the changes.
   6. **Repository**: A Git repository is a collection of files and directories, along with the version history and metadata. It can be local (on a developer's machine) or remote (hosted on a server, e.g., GitHub, GitLab, Bitbucket).
   7. **Remote Repository**: Git allows collaboration by enabling developers to work on a local copy of a repository and push their changes to a remote repository. This is often done using hosting services like GitHub, which provide a central location for collaboration.
   8. **Pull**: Pulling in Git refers to fetching changes from a remote repository and integrating them into the local repository. It is often used to update the local copy with changes made by others.
   9. **Push**: Pushing in Git involves sending local commits to a remote repository, making them available to other developers.
   10. **Merge Conflict**: A merge conflict occurs when Git is unable to automatically merge changes from different branches. Developers must resolve these conflicts manually.
   11. **GitHub and GitLab**: GitHub and GitLab are popular web-based platforms that host Git repositories. They provide additional collaboration features, such as issue tracking, pull requests, and code reviews.
       1. Git is an essential tool in modern software development, enabling efficient collaboration, version control, and project management. It is used by individual developers, small teams, and large organizations for a wide range of projects.
3. **What are the fundamental concepts of Git**?
   1. In the context of Git and GitHub, the terms "Working Directory," "Staging Area," "Local Repository," and "Remote Repository" are fundamental concepts that represent different stages in the Git workflow.
   2. **Working Directory**:
      1. The working directory is the directory on your local machine where you are currently working on your project. It contains the actual files of your project.
      2. These files may be in various states—some might be modified, others unmodified, and some may be newly created.
      3. Changes made to files in the working directory are not automatically tracked by Git until you explicitly tell Git to do so.
   3. **Staging Area (Index)**:
      1. The staging area, also known as the index, is an intermediate area where you can prepare and organize changes before committing them to the version control system.
      2. After making changes in the working directory, you use the git add command to move these changes to the staging area. This process is often referred to as staging changes.
      3. The staging area allows you to selectively choose which changes to include in the next commit.
   4. **Local Repository**:
      1. The local repository is a hidden directory within your project that contains the complete history and snapshots of your project at various points in time.
      2. When you commit changes using the git commit command, the changes in the staging area are saved as a new commit in the local repository.
      3. The local repository is where Git stores the full history of your project, and you can use Git commands to navigate through this history.
   5. **Remote Repository**:
      1. The remote repository is a repository that is hosted on a server, often on platforms like GitHub, GitLab, or Bitbucket.

|  |  |  |  |
| --- | --- | --- | --- |
| Working Directory | Staging Area | Local Repository | Remote Repository |
| Write your code here and save files. Git init to make normal folder as a git folder working directory |  |  |  |
| Check if the status of git using **git status**. To send your code files (untracked files) to staging area use **git add .** |  |  |  |
| From working directory to staging area you can send files, one by one or at a time git add <filename> git add . |  |  |  |

1. **What is git clone**?
   1. The ‘git clone’ command is used to create a copy of a remote Git repository on your local machine. This command initializes a new Git repository and fetches the entire history and files from the specified remote repository. The cloned repository on your local machine will have the same commit history and file structure as the original repository.
   2. The basic syntax for ‘**git clone**’ is as follows:
      1. git clone <repository\_url>
   3. When you clone a repository from a remote server, you create a local copy of the remote repository on your machine. You can also add a remote repository to an existing local repository using the git remote command.
      1. Example: git clone <https://github.com/example/repository.git>
   4. Here's a breakdown of what happens when you run git clone:
   5. **Initialization**: Git initializes a new repository in the specified destination directory on your local machine.
   6. **Remote Connection**: Git establishes a connection to the remote repository specified by the <repository\_url>.
   7. **Fetch**: Git fetches all the branches, commits, and files from the remote repository to your local machine.
   8. **Default Branch**: The default branch (often named "master" or "main") is checked out in your local repository.
   9. Here's a breakdown of what happens when you run git clone:
   10. After the “**git clone**” command completes, you have a complete copy of the remote repository on your local machine, and you can start working with the code. The connection to the remote repository is preserved, allowing you to fetch updates from the remote repository later.
   11. Additional options can be used with git clone to specify a different destination directory, clone a specific branch, or use other configuration options. For example:
       1. git clone -b branch\_name <repository\_url>
   12. This command clones the specified branch (branch\_name) from the remote repository.
   13. The git clone command is a fundamental step when starting to work on a project, especially when collaborating with others or when setting up a local development environment for an existing project hosted on platforms like GitHub, GitLab, or Bitbucket.
   14. The remote repository is where you can collaborate with other developers. You can push your local commits to the remote repository, and you can also fetch changes made by others.
2. In the context of GitHub:
   1. GitHub:
   2. GitHub is a web-based platform that provides hosting for Git repositories. It allows you to store your repositories remotely and provides additional collaboration features such as pull requests, issues, and project management tools.
   3. Origin:
   4. In Git, "origin" is a default name commonly used to refer to the remote repository from which the local repository was initially cloned. You typically push changes to and pull changes from the origin remote.
   5. In summary, the workflow typically involves making changes in the working directory, staging those changes in the staging area, committing them to the local repository, and then pushing those commits to a remote repository, such as one hosted on GitHub.
3. What is git init?
   1. **Git init**:
   2. The git init command is used to initialize a new Git repository. When you run this command in a directory, Git sets up the necessary data structures and files, creating a new repository from scratch.
4. What is ‘**git status**’?
   1. git status is a command used in the Git version control system to display the state of the working directory and the staging area. When you run git status in your terminal or command prompt while inside a Git repository, it provides information about:
   2. **Changes to be committed**: This section shows the changes that are staged (added to the index) and ready to be committed to the repository.
   3. **Changes not staged for commit**: These are the changes in the working directory that have not yet been staged (added to the index). In other words, these are modifications that you have made but haven't yet told Git that you want to include in the next commit.
   4. **Untracked files:** These are files in your working directory that are not tracked by Git. Git is unaware of these files, and they won't be included in commits unless you explicitly add them.
   5. The git status command is useful for understanding the current state of your repository and what changes are pending. It helps you decide what actions to take next, such as staging files for a commit or adding new files to the repository.
5. What is **‘git add’**?
   1. **git add** is a Git command used to add changes in the working directory to the staging area. The staging area, also known as the "index," is a place where changes are organized before they are committed to the repository. By adding changes to the staging area using git add, you are preparing them for the next commit.
   2. **Add Specific Files**: You can add specific files to the staging area by specifying their names: **git add filename.txt**
   3. **Add All Changes**: You can add all changes in the working directory to the staging area using: **git add .**
      1. The . indicates the current directory. This command stages all changes in the current directory and its subdirectories.
   4. **Add All Changes Interactively**: Git provides an interactive mode for adding changes, where you can selectively add changes from specific files or parts of files. To enter interactive mode, use: **git add -i**
      1. This command opens an interactive interface where you can choose which changes to stage.
      2. After using git add to stage changes, you can review the changes using git status to see which files are staged and which files have changes that are not yet staged. Once you have added all the changes you want to the staging area, you can commit them using git commit.