1) What is GIT?

Ans: GIT is a distributed version control system used in software development. It allows multiple developers to work on code simultaneously, tracks change over time and provide features like branching and merging. With GIT, developers can collaborate, manage code versions, and easily incorporate changes into the main codebase. It is widely used due to its efficiency and powerful capabilities.

2) what do you understand by the term 'version control system'?

Ans: A version control system (VCS) is a software tool that enables developers to track and manage changes to a project's source code or files over time. It provides a systematic approach to organising, preserving, and controlling different versions or revisions of the codebase. Version control systems allow multiple developers to work on the same project simultaneously, facilitate collaboration, enable easy rollback to previous versions, and provide a complete history of changes made to the code. They are essential for efficient software development, ensuring code integrity, and enabling team coordination.

3) What is GitHub?

Ans: GitHub is a web-based platform and service that hosts Git repositories. It provides a centralized location for developers to store, collaborate on, and manage their Git repositories. GitHub offers a user-friendly interface for working with Git, making it easier to create and manage repositories, track changes, and collaborate with other developers. It supports features like pull requests, issue tracking, code reviews, and project management tools, which enhance collaboration and streamline the software development process. GitHub is widely used by individuals, teams, and open-source communities to host and share their code, contribute to other projects, and showcase their work to the broader developer community.

4) Mention some popular Git hosting services.

Ans: -

GitLab: Offers cloud-hosted and self-hosted options, and provides comprehensive features including code hosting, issue tracking, and CI/CD.

Bitbucket: Provides free and paid plans, supports code collaboration, pull requests, and integrates with Atlassian tools like Jira and Confluence.

Azure DevOps: Microsoft's development toolset with Git version control, project management, CI/CD pipelines, and seamless integration with other Microsoft tools.

GitKraken: Git client with hosting services, features include Gitflow support, code reviews, and integration with GitHub, GitLab, and Bitbucket.

AWS CodeCommit: AWS-managed Git-based source control service, offers secure and scalable repositories with seamless integration with other AWS services.

5) Different types of version control systems.

Ans:

There are primarily three types of version control systems (VCS):

1. Centralized Version Control Systems (CVCS): These VCSs have a central server that stores the repository and manages version control. Developers check out files from the central server, make changes locally, and then commit them back to the central repository. Examples of CVCS include Apache Subversion (SVN) and Perforce.

2. Distributed Version Control Systems (DVCS): DVCSs do not rely on a central server for version control. Each developer has a complete copy of the repository, including its full history, on their local machine. This allows for greater flexibility, offline work, and faster operations. Git, Mercurial, and Bazaar are examples of DVCS.

3. Hybrid Version Control Systems: Hybrid VCSs combine elements of both centralized and distributed systems. They typically have a central server, but each developer also has a local copy of the repository. Changes are committed to the local repository first and then pushed to the central server. This approach combines the benefits of both centralized and distributed systems. Plastic SCM is an example of a hybrid VCS.

6) What benefits come with using GIT?

Ans:

GIT offers several benefits as follows :

1. Collaboration and Teamwork: GIT facilitates collaboration among developers by allowing multiple individuals to work on the same codebase simultaneously. It enables seamless integration of changes made by different team members, making it easier to coordinate and merge contributions.

2. Version Control: GIT tracks changes to files and directories, providing a complete history of modifications. It allows developers to review, compare, and revert to previous versions when needed. This ensures code integrity, helps identify and fix bugs, and provides a safety net for experimentation.

3. Branching and Merging: GIT's branching and merging capabilities enable developers to work on separate features or bug fixes independently. Branches provide an isolated space for development, allowing for experimentation without affecting the main codebase. Merging combines changes from different branches, facilitating the integration of new features or bug fixes.

4. Flexibility and Speed: GIT is designed for speed and efficiency, allowing developers to perform operations quickly, even with large codebases. It supports offline work, as each developer has a complete copy of the repository on their local machine. This flexibility enables productivity in diverse development environments.

5. Distributed Architecture: GIT's distributed nature provides redundancy and reliability. Each developer has a full copy of the repository, reducing the risk of data loss. It also enables work to continue even in the absence of a centralized server, promoting collaboration in distributed teams.

6. Community and Ecosystem: GIT has a large and vibrant community with extensive support, documentation, and resources available. Developers can leverage open-source projects, share their code, and benefit from collaboration within the GIT ecosystem.

7. Integration and Tooling: GIT integrates well with various tools and services used in the software development lifecycle. It seamlessly integrates with code editors, continuous integration and deployment (CI/CD) pipelines, project management tools, and hosting services, enhancing the overall development workflow.

7) What is a GIT repository?

Ans:
1) A GIT repository is a data structure that stores files and directories along with their complete version history.

2) It serves as a central hub for managing and tracking changes to source code and project files.

3) The repository contains all the commits made by developers, branching information, and metadata associated with the project.

4) It can be hosted on a remote server or stored locally on a developer's machine.

5) The repository maintains the entire history of changes, allowing developers to review, revert, and merge code changes.

6) Each commit within the repository has a unique identifier and can include descriptive messages.

7) It supports collaboration among developers, enabling multiple individuals to work on the same project simultaneously.

8) Branching and merging functionality allows developers to work on different features or isolate bug fixes without affecting the main codebase.

9) Repositories can be created from scratch or cloned from existing repositories.

10) GIT repositories can be hosted on platforms like GitHub, GitLab, or self-hosted servers.

11) GIT provides commands and tools for interacting with repositories, including committing changes, branching, merging, and pushing changes to remote repositories.

8) How can you initialize a repository in Git?

Ans:
To initialize a GIT repository, we can follow these steps:

1. Open a terminal or command prompt in the desired directory where you want to create the repository.

2. Use the `git init` command. Simply type `git init` and press Enter. This command initializes an empty GIT repository in the current directory.

3. Once executed, you will see a message indicating that the repository has been initialized. It will create a hidden `.git` folder, which contains all the necessary files and metadata for version control.

After initialization, you can start adding files, making commits, and managing your codebase using GIT commands.