**Question no 1:**

**What is Git?**

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. A Git repository is a virtual storage of your project. It allows you to save versions of your code, which you can access when needed.

Git helps you in following

* Creating/initializing new git repository
* Versioning project with new git repository
* Cloning existing git repository
* Saving changes to the repository

**Question no 2:**

**What are git workflows?**

Some common Git workflow are following

* Centralized workflow
* Feature Branching
* Gitflow workflow
* Forking workflow

**1)Centralized Workflow**

There is one central repository. Each developer clones the repo, works locally on the code, creates a commit with changes, and pushes it to the central repository for other developers to pull and use in their work.Centralized Workflow uses a central repository to serve as the single point-of-entry for all changes to the project.

**How it works?**

Developers start by cloning the central repository. In their own local copies of the project, they edit files and commit changes, however, these new commits are stored locally. To publish changes to the official project, developers "push" their local repository to the central repository.

The central repository represents the official project, so its commit history should be treated as sacred and immutable. If a developer’s local commits diverge from the central repository, Git will refuse to push their changes because this would overwrite official commits.

**2)Feature Branching**

Feature Branch Workflow creates dedicated branch instead of the main branch. This encapsulation makes it easy for multiple developers to work on a particular feature without disturbing the main codebase. It also means the main branch will never contain broken code, which is a huge advantage for continuous integration environments.

* Create new branch from main branch
* Update, add, commit, and push changes
* Push feature branch to remote

**How it works?**

Instead of committing directly on their local main branch, developers create a new branch every time they start work on a new feature. Feature branches should have descriptive names, like animated-menu-items or issue-#1061. The idea is to give a clear, highly-focused purpose to each branch. Git makes no technical distinction between the main branch and feature branches, so developers can edit, stage, and commit changes to a feature branch.