Difficulties or challenges during development

The challenges are below -

1. Continuous Integration/Continuous Delivery.
2. Non-optimal developer productivity.
3. Distributed teams.
4. Managing build artifacts.
5. Scalability.

**Branches:**

A branch can be thought of as an internal clone of a repository. When you are working on a repo you will choose which branch to work with. If you have not created any branches, then you are automatically working with the default branch. At any time, you can create a new branch based on an existing branch and, then you change between branches depending on what you need to work on at any given time.

**Stages:** A staging step in git allows you to continue making changes to the working directory, and when you decide you want to interact with version control, it allows you to record changes in small commits

**Commits:**

Your commit messages are a form of communication. You are sending a message to your teammates or even to your future self! Be nice to your colleagues. Be nice to your future self. Make sure that your commit messages are meaningful.

Recall that update changes the working copy by applying any edits that appear in the repository but have not yet been applied to the working copy.

In a centralized version control system, you can update (for example, svn update) at any moment, even if you have locally uncommitted changes. The version control system merges your uncompleted changes in the working copy with the ones in the repository. This may force you to resolve conflicts. It also loses the exact set of edits you had made since afterwards you only have the combined version. The implicit merging that a centralized version control system performs when you update is a common source of confusion and mistakes.

**Merges**: 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.

**Push/pull:**

This is an extension to make small frequent commits for those that use distributed version control.

Push frequently to your master repository. Push every day where possible. The longer you wait to integrate your work with the team the more difficult merging will become and the more likely it will be that you will break the build.

**Commit log:** The git log command lists all the commits made to a repository. You can see the hash of each Git commit, the message associated with each commit, and more metadata. This command is useful for displaying the history of a repository.

**Forking or cloning your repository is an alternative to branching.**

Where a branch is an internal copy of a repository, a fork is an external copy: a completely separate repository. A forked repository is related to the original repository and code changes can easily be merged between them. Anything you can do with branching you can also do by forking. You can fork separate repositories for development and production, you can fork separate repositories for features, tasks and individuals.

Forking can be more flexible than branching. Once you have branched your repository, it remains in your repository and contributes to its size (bloating the repository). Forks of a repository exist externally, making it easier to safely dispose of them when they are no longer needed. If you find yourself creating short-lived or many branches, you might want to consider using forks instead.