1 . What are the advantages of using Git ?

Answer :

1. Performance :

Git performs very strongly and reliably when compared to other version control systems. New code changes can be easily commited, version branches can be effortlessly compared and merged, and code can also be optimized to perform better.

1. Security:

Git is designed specially to maintain the integrity of source code. File contents as well as the relationship between file and directories, tags, commits, versions etc. are secured cryptographically using an algorithm called SHA1 which protects the code and change history against accidental as well as malicious damage. You can be sure to have an authentic content history for your soruce code with Git.

1. Flexibility:

A key design objective of Git is the kind of flexibility it offers to support several kinds of nonlinear development workflows and its efficiency in handling both small scale and large scale projects as well as protocols. It is uniquely designed to support tagging and branching operations and store each and every activity carried out by the user as an integral part of "change" history. Not all VCSs support this feature.

1. Wide acceptance:

Git offers the type of performance, functionality, security and flexibility that most developers and teams need to develop their projects. When compared to other VCS Git is most widely accepted system owing to its universally accepted usability and performance standards.

1. Quality Open Source Project:

Git is a widely supported open source project with over ten years of operational history. People maintaining the project are very well matured and possess a long term vision to meet the long term needs of users by releasing staged upgrades at regular intervals of time to improve functionality as well as usability.

2. What language is used in Git ?

Answer:

Written in C, TCL, PERL, PYTHON

3. What is the meaning of index or staging Area in Git ?

Answer:

While working on a project, if we make any changes then we are dealing with our project’s working directory. This project directory is on our computer’s file system. All the changes we make remain in the working directory until we add them to the staging area via git add command. The staging area is best described as a preview of our next commit. Meaning when we do a git commit, git will take the changes that are in staging area and make a new commit out of those changes.

One practical use of the staging area is that it allows us to fine-tune our commits. We can add and remove changes from staging area until we are satisfied with how our next commit will look like, at which point we can do git commit. And after we commit our changes they go into .git/objects directory where they are saved as commit, blob and tree objects.

4. What is the process of creating a repository in Git ?

Answer:

1. In the upper-right corner of any page, click , and then click **New repository**.
2. Type a short, memorable name for your repository. For example, "Parth Repo".
3. Optionally, add a description of your repository. For example, "My repository on GitHub."
4. Choose to make the repository either public or private.
5. Select **Initialize this repository with a README**.
6. Click **Create repository**.

5. What is head in github and how many heads can be created in github?

Answer:

Head is the current branch. It is a symbolic reference to a branch. HEAD is always there and it’s pointing to one of those these other pointers, to one of the branches that we are currently working on. It is basically the parent of our next commit and it is what should be what was last checked-out into our working directory.

6. Why do we need branching in Git ?

Answer:

Using branches help you organize the workflow more efficiently and rather effortlessly. Let’s say you are building a software for a company in team. It would be a good idea to have a branch for each developer because usually they work on different things. And because of the way Git works, you can keep working on your branch regardless of the work that is happening in other branches.

7. Write a way to create a new branch in Git.

Answer:

$ git checkout -b mybranch

$ git push origin mybranch

To see all the branches we can use :

$ git branch -a

8. How do you define a conflict in Git ?

Answer:

A merge conflict usually occurs when your current branch and the branch you want to merge into the current branch have diverged. That is, you have commits in your current branch which are not in the other branch, and vice versa. Now, when Git merges the other branch into your current branch, it looks at the differences between the base commit and the current revision, and at the differences between the base commit and the other branch's latest commit. When there are unambiguous differences (i.e. only one side changed a certain piece of code), the changes are applied.

The merge conflicts occur when there are disagreeing changes. In that case, your conflicted file will have so-called conflict markers

9.