1. What is the purpose of branching in GIT?

Ans :

Git is a fundamental tool which used to maintain the source code and track its changes.

Maintain and mange the code for all team members.

Creating Branch : Git Branches allow us to create separate contexts where we can try new things or even

work on multiple ideas in parallel without risking the codebase.

And make use further for adding new changes/modification/new feature without distracting default branch.

Teammates can work on individual branch on there need and push the changes to master branch once get approved.

Hence branch provide alternate line for developing the project which isolated form Main branch.

And easy to maintain and manage your source code without risking the source code.

2.

1. What is a ‘conflict’ in git?

Conflict merge usually arises when your current branch merge and the branch you want to merge with your current branch

Have diverged , that is you have commits in your current branch which are not the there in the other branch which you looking to merge vice versa.

1. How can you bring a new feature in the main branch?

Ans : Here is the example to bring a new feature in the main branch

* *Git checkout -b new-branch* (create new branch with Branch name new-branch for an example)
* Do the changes on the file or do any modification /add new file etc.. once done the implementation
* Git add filename (add the changes from work repository to stage )
* Git commit -m “commit text” (commit the code from Stage to .git local repository )
* Git push origin new-branch
* Create Pull request : pull request from current branch to main branch
* Once pull request review : if feedback provided than do the changes and commit and push.
* Before Merging the branch : rebase the your branch on the top of the latest changes in main branch
* Rebase branch :
* Make sure you are on your branch by running the command git checkout new-feature.
* Run the command git fetch to fetch the latest changes from the remote repository.
* Run the command git rebase main. This will apply your changes on top of the latest changes in the main branch. If there are any conflicts, Git will prompt you to resolve them.
* After resolving any conflicts, run the command git add <filename> to stage the changes, and then run the command git rebase --continue to continue the rebase.
* Repeat steps 3 and 4 until the rebase is complete.
* Run the command git push --force to update your branch on the remote repository with the rebased changes
* Merge the branch: Once the changes have been reviewed and approved, merge your branch into the main branch using the repository's web interface.

1. What is another option for merging in git?

Ans : another option for combining changes from one branch to another in Git is called "rebase".

While merging combines changes from two branches by creating a new commit with the combined changes, rebasing applies the changes from one branch on top of another branch's commit history. In other words, when you rebase, you take the changes from one branch and move them to a new base commit in another branch, effectively rewriting the commit history of the rebased branch.

This can be useful in situations where you want to keep a linear commit history, or if you have a long-running feature branch that has diverged significantly from the main branch and you want to integrate the latest changes from the main branch into your feature branch.

To rebase a branch, you can use the following steps:

1. Make sure you are on the branch that you want to rebase. For example, git checkout my-feature-branch.
2. Run the command git fetch to make sure you have the latest changes from the remote repository.
3. Run the command git rebase <base-branch> to rebase your branch on top of the specified base branch. For example, git rebase main.
4. Git will apply the changes from your branch on top of the latest commit in the base branch. If there are any conflicts, Git will prompt you to resolve them.
5. After resolving any conflicts, run the command git add <filename> to stage the changes, and then run the command git rebase --continue to continue the rebase.
6. Repeat steps 4 and 5 until the rebase is complete.
7. Once the rebase is complete, you can use the command git push --force to update your branch on the remote repository with the rebased changes.

It's important to note that rebasing changes the commit history of your branch, so you should only rebase if you are working on a personal branch or a branch that hasn't been shared with other developers. If you have already shared your branch with others, rebasing can cause conflicts and make it difficult to merge changes later on.

In summary, while merging is the default option for combining changes from one branch to another in Git, rebasing can be a useful alternative in certain situations where you want to keep a linear commit history or integrate changes from one branch into another in a more streamlined way.

1. What is the difference between ‘git remote’ and ‘git clone’?

Ans : git remote and git clone are two different Git commands that serve different purposes.

git remote is used to manage the set of remote repositories that Git knows about. A remote repository is a repository on a different server that contains a copy of your project's code. When you clone a repository, Git automatically sets up a remote named "origin" that points to the repository you cloned from. You can use the git remote command to view, add, or remove other remotes for your project.

Here are some examples of git remote commands:

*git remote*: Lists the names of the remote repositories that Git knows about.

*git remote -v*: Lists the names and URLs of the remote repositories that Git knows about.

*git remote add <name> <url>* : Adds a new remote repository with the given name and URL to your Git project.

*git remote remove <name>* : Removes the remote repository with the given name from your Git project.

On the other hand, git clone is used to create a local copy of a remote repository. When you run git clone, Git copies all the files and history from the remote repository to your local machine, and sets up a local Git repository that tracks changes to the files.

Here's an example of how to use git clone:

*git clone <url>* : Clones the remote repository at the given URL to your local machine.

So, the main difference between git remote and git clone is that git remote manages the set of remote repositories that Git knows about, while git clone creates a local copy of a remote repository on your machine.

Top of Form

1. What is the function of ‘git diff ’ in git?

Ans :