Fetch

"Fetch" is a Git command that is used to retrieve the latest changes from a remote repository without automatically merging them into the local branch. It allows you to see the changes made by others without modifying your current branch.

Steps to perform a "fetch" in Git:

1. Open a terminal or command prompt and navigate to your local repository.

2. Run the following command to fetch the latest changes from the remote repository:

```
git fetch
```

This command will fetch all the branches and their respective commits from the remote repository.

3. After fetching the changes, you can view the updated branches and their commits by running:

```
git branch -r
```

This will show you the remote branches and their commit references.

4. If you want to merge the fetched changes into your current branch, you can use the *git merge* command. For example:

```
git merge origin/master
```

This will merge the changes from the remote branch *origin/master* into your current branch.

Pull

"Pull" is a Git command that is used to retrieve the latest changes from a remote repository and automatically merge them into the current branch. It combines the "fetch" and "merge" operations into a single command.

Steps to perform a "pull" in Git:

1. Open a terminal or command prompt and navigate to your local repository.

2. Run the following command to pull the latest changes from the remote repository and merge them into your current branch:

```
git pull
```

This command will automatically fetch the latest changes from the remote repository and merge them into your current branch.

3. If there are no conflicts between your local changes and the remote changes, Git will automatically merge them. If conflicts occur, you will need to manually resolve them.

4. After the pull operation is complete, you can use the *git log* command to view the commit history, including the merged changes.

Fetch V/S Pull

	Fetch	**Pull**
Definition	Retrieves the latest changes from a remote repository without merging them into the current branch.	Retrieves the latest changes from a remote repository and automatically merges them into the current branch.
Impact on local branch	Does not modify the local branch.	Automatically merges the retrieved changes into the local branch.
Remote tracking branches	*Updates the remote tracking branches (e.g., origin/master*) with the latest changes.	Updates the remote tracking branches *(e.g., origin/master)* and merges the changes into the local branch.
Conflicts	Does not cause any conflicts as it does not modify the local branch.	May cause merge conflicts if the retrieved changes conflict with the local branch's modifications.
Additional steps	Requires an additional step to explicitly merge the fetched changes into the local branch.	Merges the retrieved changes automatically into the local branch.
Usage scenario	Useful when you want to see the latest changes from a remote repository without merging them immediately.	Suitable when you want to retrieve and merge the latest changes from a remote repository into the local branch.

	Fetch	**Pull**
Command	*git fetch <remote>*	*git pull <remote>*

Merge

"Merge" is a Git command used to combine two or more branches together. It integrates changes from one branch into another branch, creating a new commit that incorporates the changes from both branches.

Steps to perform a "merge" in Git:

1. Open a terminal or command prompt and navigate to your local repository.

2. Ensure that you are on the branch where you want to merge the changes. Use the following command to switch to the target branch:

```
git checkout <target-branch>
```

Replace *<target-branch>* with the name of the branch where you want to merge the changes.

3. Run the following command to merge the changes from another branch into the current branch:

```
phpCopy codegit merge <source-branch>
```

Replace *<source-branch>* with the name of the branch that contains the changes you want to merge.

4. Git will attempt to automatically merge the changes. If there are no conflicts, the merge will be successful, and Git will create a new commit that combines the changes from both branches.

5. If conflicts occur during the merge, Git will pause the process and indicate the conflicting files. You will need to manually resolve the conflicts by editing the files and choosing which changes to keep.

6. After resolving the conflicts, stage the modified files using the *git add* command.

7. Once all conflicts are resolved, complete the merge by running the following command:

```
git commit
```

This will create a new commit with the merged changes.

8. After the merge is complete, you can use the *git log* command to view the commit history and verify that the changes have been merged.

Merge Conflict

A *"merge conflict"* occurs when Git is unable to automatically merge changes from different branches due to conflicting modifications in the same file or lines of code. It arises when two or more branches have made conflicting changes to the same part of a file, and Git is unable to determine which changes to keep.

Steps to resolve a merge conflict in Git:

1. When a merge conflict occurs, Git will notify you about the conflicting files. You can identify them by the presence of conflict markers *(<<<<<<<, =======, and >>>>>>>)* in the affected files.

2. Open the conflicting file(s) in a text editor and locate the conflict markers. They indicate the conflicting sections where changes from different branches are in conflict.

3. Review the conflicting changes and decide which changes to keep. Edit the file to manually resolve the conflict by modifying the code to include the desired changes.

4. Remove the conflict markers *(<<<<<<<, =======, and >>>>>>>)* once the conflict is resolved. Ensure that the final version of the code reflects the desired changes.

5. Save the changes in the file and stage it using the *git add* command.

6. After resolving all conflicts in the file(s), continue the merge process by running the command:

```
git commit
```

This will create a new commit that finalizes the merge with the resolved conflicts.

7.Git will automatically complete the merge process once the commit is made, and the conflicting changes will be successfully merged