**When a pull request (PR) contains code that is not properly formatted?**

**Leave a Comment**: Politely inform the contributor that the code needs to be formatted according to the project's coding standards

**Provide Resources**: Share links to any formatting guidelines or tools that the project uses, such as:

**When a pull request (PR) doesn't include test cases**

· **Politely Request Tests**: If you're the reviewer, ask the contributor to add tests before merging the PR. Clearly explain that adding tests is important for ensuring the quality and stability of the code.

· **Specify Which Tests Are Needed**: If possible, specify the types of tests required, such as:

* **Unit Tests**: To test individual functions or methods.
* **Integration Tests**: To ensure that different parts of the application work together.
* **End-to-End Tests**: If applicable, to verify the entire application flow.

If someone pushes directly to the **main** branch, bypassing the usual code review process (such as through a **Pull Request**

**Revert the Commit**: If the changes cause problems or are deemed inappropriate for the main branch, you can **revert the commit** or **reset the branch** to a previous stable state.

Difference between git rest and reverse

| **Feature** | git revert | git reset |
| --- | --- | --- |
| **Purpose** | Creates a new commit that undoes a previous commit | Moves HEAD to a specified commit and optionally modifies the index/staging area and working directory |
| **Affects History** | Does **not** modify commit history (preserves it) | **Modifies** history (rewrites it) |
| **New Commit Created?** | Yes, a new commit is created that undoes the changes | No, it moves the HEAD pointer to the given commit |
| **Safe for Collaboration?** | Yes, it's safe for collaborative environments | Not safe if commits have already been pushed to the remote (may require --force) |
| **Changes Working Directory?** | No, only affects the commit history | Yes, can affect staging area and working directory depending on options (--soft, --mixed, --hard) |
| **Use Case** | Undoing a commit in a public/shared branch without rewriting history | Undoing commits locally or cleaning up commit history (before pushing) |
| **Best for** | Reverting a commit that has already been pushed, preserving history | Removing commits before pushing or fixing local mistakes |
| **How to Use** | git revert <commit> | git reset <commit>  git reset --soft <commit>  git reset --mixed <commit>  git reset --hard <commit> |
| **Impact on Remote Repository** | Safe, no impact on remote history | Can be dangerous for remote repositories; may require git push --force to overwrite history |
| **Type of Undo** | Adds a new commit that undoes changes | Moves HEAD and changes history, can reset staging area or working directory |
| **Feature** | git fetch | git pull |
| **Purpose** | Retrieves updates from the remote repository (e.g., new commits, branches), but does **not** merge them into your local branch. | Retrieves updates from the remote repository **and** merges them into your local branch in one step. |
| **Changes Local Working Directory** | **No**, git fetch only updates the local reference of remote branches. | **Yes**, git pull fetches changes **and** merges them into the current branch, potentially changing the working directory. |
| **Updates Local Branch** | **No**, it does not modify your local branches or working directory. It just updates remote-tracking branches (e.g., origin/main). | **Yes**, it updates your current local branch by merging the fetched changes. |
| **Operation Steps** | Two steps: 1. Fetches changes (downloads new commits from the remote). 2. Remote branches are updated locally (e.g., origin/main). | One step: 1. Fetches changes (like git fetch) and 2. Automatically merges changes into your current local branch. |
| **Use Case** | When you want to see what changes are available on the remote repository **without affecting your local branch** or working directory. It allows you to review changes before merging them. | When you want to fetch the changes from the remote **and automatically merge** them into your local branch. |
| **Common Workflow** | git fetch is often used in combination with git merge to manually integrate remote changes into your local branch. | git pull is often used when you want to **automatically fetch and merge** in a single step. |
| **Safety** | **Safer** because it only updates references to remote branches and does not change your local branch or working directory. | **Riskier** because it will merge remote changes into your current working branch, which might cause merge conflicts or unexpected changes. |
| **Merge Behavior** | No merge occurs in git fetch. To merge fetched changes, you need to use git merge or other merging tools. | git pull merges the fetched changes automatically into your local branch. If there are conflicts, you'll need to resolve them. |
| **Example Command** | git fetch origin (fetches changes from the origin remote). | git pull origin main (fetches and merges changes from the main branch of the origin remote into the current branch). |

### 1. git init:

#### ****Purpose****:

* Initializes a new Git repository in a directory. It sets up the necessary .git directory that allows Git to track changes and version control in that folder.
* This command is typically used when you start a new project or want to turn an existing directory into a Git-managed repository.

### 2. git tag:

#### ****Purpose****:

* Used to create, list, or delete tags in Git. A **tag** is a reference to a specific commit, usually used to mark a particular point in history, such as a release or version.
* Tags are useful for marking important points in your project’s history (e.g., version releases, milestones).

### 3. git stash:

#### ****Purpose****:

* Temporarily saves (or stashes) your uncommitted changes (both staged and unstaged) so that you can work on something else, and then apply the changes back later.
* This is useful when you’re working on a task but need to switch contexts quickly (e.g., switching branches, pulling new changes), but don’t want to commit your incomplete work yet.