# Git Revert - Detailed Explanation

`git revert` is used to undo a commit by creating a new commit that reverses the changes of a previous one. Unlike `git reset`, which can change history, `revert` is safe to use on shared/remote branches.

## Explanation of Your Notes

### 1. We do on remote also

This means `git revert` can be used on commits that have already been pushed to a remote repository (e.g., GitHub). It is safe because it doesn't alter the commit history. Instead, it adds a new commit that negates the changes from the target commit.

Example:

***git revert abc1234  
git push origin main***

### 2. We can revert any commit

You can revert any commit, whether it's the latest one or a commit in the middle of the history.

Example:

***git log  
git revert <commit-id>***

To revert multiple commits:

***git revert HEAD~3..HEAD***

### 3. It maintains history also

Git does not delete the original commit when you use revert. It creates a new commit that applies the inverse of the original commit's changes. This helps in maintaining a complete and auditable history.

## How `git revert` Works (Under the Hood)

When you run:

***git revert abc1234***

Git finds the changes introduced in abc1234, applies the opposite of those changes in a new commit, and retains abc1234 in the history.

## Use Case Example

Suppose you accidentally pushed a bug in `myfile.txt` on the `main` branch:

***vi myfile.txt  
# You notice the mistake  
git revert <buggy-commit-id>  
git add .  
git commit -m "Revert buggy commit"  
git push***

## Visual Representation

***Original history:  
A --- B --- C (buggy) --- D (latest)  
  
After revert:  
A --- B --- C --- D --- E (revert C)***

The history stays intact, but the effect of C is undone in E.

## Summary

| Feature | git revert |
| --- | --- |
| Affects history? | No |
| Safe for remote use? | Yes |
| Creates new commit? | Yes |
| Deletes old commit? | No |
| Use case | Undoing specific commits safely |

# 1. Git Basics and Repo Workflow

## Creating and Working with a Repo

git clone <repo-url>: Clones a repo (usually HTTPS or SSH).

git add -A: Stages all changes (new, modified, deleted).

git commit -m "message": Commits staged changes with a message.

git push: Pushes commits to the remote repo.

git log: Shows commit history.

git checkout -b <branch>: Creates and switches to a new branch.

git push -u origin <branch>: Pushes the new branch to remote and sets it to track upstream.

# 2. Git Reset vs Revert

## Reset

git reset --hard HEAD~1: Moves HEAD and branch pointer back by one commit and deletes that commit from history.

Use it with caution, mostly in local branches.

## Revert

Safe for remote/shared branches.

Creates a new commit that undoes changes from a specific previous commit.

Maintains the commit history intact.

Example:

***git revert <commit-id>***

You can also manually revert file changes:

***vi myjavafiletwo # revert changes manually  
git add -A  
git commit -m "reverted changes"  
git push***

# 3. Merge Conflicts

## Workflow

Modify the same file (e.g., myjavafile) in two branches (e.g., main and branchone).

Merge with:

***git checkout main  
git merge branchone***

If there are conflicts, Git will highlight them like:

***<<<<<<< HEAD  
content from main  
=======  
content from branchone  
>>>>>>> branchone***

## Steps to Resolve Conflict

Fix the conflicted file manually.

Run:

***git add <file>  
git commit -m "resolved merge conflict"  
git push***

# 4. Git Hooks

Hooks are custom scripts that automate tasks at different points in the Git workflow.

Location: .git/hooks/

Types:

- Client-side hooks: Run on user operations like commit, merge (e.g., pre-commit, commit-msg).

- Server-side hooks: Run on remote repo actions (e.g., pre-receive, post-receive).

# 5. Java Setup on Windows

## Install Steps

1. Download JDK 8 and install it.

2. Set environment variables:

- Add JAVA\_HOME and PATH in System Properties > Environment Variables.

- Example path: C:\Program Files\Java\jdk1.8.0\_271\bin

## Test Installation

Run the following commands to test your Java setup:

***java -version  
javac HelloWorld.java***

Generates a HelloWorld.class file that can be executed with:

***java HelloWorld***