**4. Git Revert vs Git Reset**

**1. git revert**

* **Purpose:** Safely **undo a commit** by creating a **new commit** that reverses the changes of a specific commit.
* Keeps project history intact (no rewriting).
* Common in **team projects** because it doesn’t rewrite history.

**Examples:**

* Revert a single commit:
* git revert commit\_hash
* Revert multiple commits:
* git revert commit1\_hash commit2\_hash

**2. git reset**

* **Purpose:** Move the **HEAD** and branch pointer to a different commit, optionally changing the staging area and working directory.
* Can **rewrite history** (be careful when working with shared branches).

**Modes of reset:**

1. **Soft reset** (keeps changes staged):
2. git reset --soft commit\_hash
   * Moves HEAD, but keeps all changes staged for commit.
3. **Mixed reset** (default – keeps changes, unstaged):
4. git reset commit\_hash
   * Changes remain in working directory, but staging area is cleared.
5. **Hard reset** (dangerous – deletes changes):
6. git reset --hard commit\_hash
   * Resets everything to that commit (staged + working directory lost).

**✅ In short**

* git revert → Safe undo (makes a new commit to cancel changes).
* git reset → Dangerous undo (moves HEAD, can erase commits/changes).
* Use **revert** for **team work**, use **reset** for **local cleanups**.