**git revert**

git revert <commit>: This command is used to undo the changes made by a specific commit. It creates a new commit that reverses the changes of the specified commit. This is a safe way to undo changes without rewriting the project history.

Example: git revert 1a2b3c4d (where 1a2b3c4d is the commit hash you want to revert)

git revert -n <commit>: The -n or --no-commit option reverts the changes but doesn't create a commit immediately. This allows you to make additional changes before committing.

Example: git revert -n 1a2b3c4d

**git reset**

git reset <commit>: This command resets the current branch to the specified commit, potentially altering the project history. By default, git reset will not alter your working directory or staged changes, it only moves the HEAD and the current branch reference.

Example: git reset 1a2b3c4d

git reset --soft <commit>: The --soft flag resets the branch to a previous commit but keeps your staged changes and working directory unchanged. This is useful for undoing a commit while keeping the changes for a new commit.

Example: git reset --soft 1a2b3c4d

git reset --mixed <commit>: This is the default mode for git reset. It resets the index but not the working tree. Changes are preserved but not staged.

Example: git reset --mixed 1a2b3c4d (or simply git reset 1a2b3c4d)

git reset --hard <commit>: The --hard flag resets both the staging area and the working directory to match the specified commit. Any local changes are lost. This is a powerful command that should be used with caution.

Example: git reset --hard 1a2b3c4d

git reset --keep <commit>: Similar to --hard, but if the working directory has modifications, the reset will be aborted. This is a safer option than --hard when you want to preserve local changes.

Example: git reset --keep 1a2b3c4d

Remember, git revert is safer for undoing changes in shared/public branches since it doesn't alter the existing history. git reset, especially with the --hard option, is more disruptive and is best used in local or private branches. Always be cautious when using git reset to avoid losing work.