Git Revert, Reset (Hard, Soft, Mixed), and Checkout

Youtube video

`Git Revert, Reset and Checkout` commands are used to discard or to undo changes in git.

How to discard (undo) changes in the working directory in git?

Try `git checkout --<file>` to discard uncommitted changes to a file in the working directory.

Discard changes for a particular file in the working directory

```
git checkout --<file>
```

Discard all changes in the working directory

```
git checkout .
```

How to discard (undo) changes in the staging directory in git?

Try `git checkout --<file>` to discard uncommitted changes to a file in the working directory. When we unstage a file, changes are kept in the working directory.

Discard changes for a particular file in the staging area

```
git resert HEAD --<file>
```

Discard all changes in the working directory

```
git resert HEAD *
```

Git rest command (It completely remove the commit from the commit history)

```
git reset --help
```

We have 3 types of reset in git:

1. **Hard reset:** change will be wiped out from the working directory, staging area, and local repository.

```
git reset --hard HEAD~1
```

1. **Soft reset:** change will be wiped out from the local repository and you will have a copy of your files in your working directory and staging area.

```
git reset --soft HEAD~1
```

1. **Mixed reset:** change will be wiped out from the local repository, staging area and you will have the copy of your files only in your working directory, or all files added as part of this commit are kept in the working area. **NB:** This is the default command if we type the git reset command without specifying any option or argument.

```
git reset --mixed HEAD~1
git reset HEAD~1
```

Git reset command

```
git reset --hard HEAD~1 : to remove the first commit
git reset --hard HEAD~2 : to remove the first 2 commits
git reset --hard HEAD~3 : to remove the first 3 commits
git reset --hard HEAD~n : to remove the first n commit
```

Git revert command (will require a new commit message)

- if we use `git reset` command when the code is already pushed and make public, the will have a conflict because it will only discard changes locally.
- The best option in this situation is `git revert`

```
git revert <commit Id>
git log --oneline
git revert 5fa9842
git log --oneline
git status
ls: changes will be gone also from the working area
```

What is the difference between git reset and git revert?

- **Git reset** completely takes out the commit from the history while git revert will require a new commit message.
- **Git reset** will revert the changes and delete all the commits in the log while git revert will revert the changes, required, and new commits and keep all the commits history in the log so that we can still revert to any commit as needed.
- We can get any change back with git revert because it going to create a new commit ID while `git reset` will not create any `commit ID`
- If your commit is not yet pushed to remote, you can use `git reset` to revert changes while git revert is used when the commit is already pushed and made public. This means `git reset` is used to undo only local commits.
- with `git revert`, you can do undo any commit like the fourth commit for instance while with `git reset`, you can only undo recent commits or the one commit, two commits, etc.
- **NB:** They is no way you can undo a particular commit with `git reset`. Using `git revert`, you can undo changes in any commit.
- **NB:** `Git reset hard` is a restricted command because changes will be going forever from the working directory, cache, and log file.