Advanced Git Workflows

1. Rebase
2. Cherry-pick
3. Conflict resolution

Interactive Rebase

A Tool for Optimizing & Cleaning Up git Commit History.

- change a commit's message
- delete commits
- reorder commits
- combine multiple commits into one
- edit / split an existing commit into multiple new ones

Do NOT use Interactive Rebase on commits that you've already pushed/shared on a remote repository!

Instead, use it for cleaning up your local commit history before merging it into a shared team branch.

Step by Step

Step 1:

How far back do you want to go?
What should be the "base" commit?
To find this, use the command:

```
git log --oneline
```

This will show the commit messages.

Step 2: Use the command:

```
git rebase -i HEAD~3
```

An editing window will pop up. After that, we can manipulate these commits. However, do not change the commit message in this window. Instead, choose the type of manipulation you want to perform, and use the action keywords (pick, exec, merge, etc.). Then, save and close the editor.

After this, an editor window will appear again. You can now finally change the commit message and save it. You will see that the commit message has been updated.

Risky if used on shared/public branches (changes commit history)

Cherry-pick

In some special situations, We might only want a specific commit not all the commits from another branch. And this is where it gets cherry picking tool comes in handy.

Merge and rebase were built exactly for this job but it is not a replacement for marge and rebase.

Example :
You are working on a feature-branch, and there is one specific commit that you want to apply to the main branch.

Use this command Step by step

Action	Command
See commit history	*git log --oneline*
Checkout target branch	*git checkout main*
Apply specific commit	*git cherry-pick <commit_hash>*
Continue after conflict	*git cherry-pick --continue*
Cancel cherry-pick	git cherry-pick --abort

Git Conflict Resolution

A conflict happens when two branches change the same part of a file. Git asks you to fix it manually.

Steps to Resolve a Conflict:

1. Git will show the conflicted files.
2. Open the file. You will see markers like:

```
<<<<<<< HEAD
Your version
=======
Other version
>>>>>>> branch-name
```

3. Choose which changes to keep or combine both.
4. Delete the conflict markers.
5. Save the file.

After Fixing:

- Add the file:

```
git add filename
```

- If merging, run:

```
git commit
```

- If rebasing, run:

```
git rebase --continue
```

To Cancel:

- For merge:

```
git merge --abort
```

- For rebase:

git rebase --abort

Summary:

- Conflicts are normal. Fix them by editing files.
- Use add to mark them as resolved.
- Use abort to cancel if needed.