Git Concepts: Cherry-pick, Rebase, Merge, and Git Clients

1. Cherry-Pick (Reuse specific commits)

Scenario:
You’re working in Sprint-03 and you realize a bug fix or a useful feature was committed in Sprint-02, and you want to apply that specific commit to your current branch.

Commands:

```
git checkout sprint-03
git cherry-pick <commit-hash>
```

To cherry-pick multiple commits:

```
git cherry-pick <commit1> <commit2> <commit3>
```

Use Case: Copying specific changes without merging the whole branch.

2. Rebase (Reuse entire branch with clean history)

Scenario:
You want to place all your commits from Sprint-03 on top of the latest changes from Sprint-02, keeping history linear.

Commands:

```
git checkout sprint-03
git rebase sprint-02
```

Before Rebase:

```
sprint-02: A - B - C - D
sprint-03:      \
           E - F
```

After Rebase:

```
sprint-02: A - B - C - D - E' - F'
```

Use Case: Keeping a clean, linear history while applying updates from another branch.

3. Merge (Combine branches with history)

Scenario:
You completed work in Sprint-03 and want to bring those changes into Sprint-02, keeping the complete commit history.

Commands:

```
git checkout sprint-02
git merge sprint-03
```

Use Case: Combining full branches, maintaining commit history and context.

4. Git Clients (for GUI-based operations)

Top Clients:

- GitHub Desktop

- GitKraken

- Sourcetree

- TortoiseGit

- SmartGit

- Git Cola

- Aurees

- Fork

- Magit

- GitForce

Scenario:
Developers/testers prefer GUI tools like GitKraken or GitHub Desktop for ease of use.
DevOps prefer CLI tools like Git Bash for scripting and automation.

Quick Tip – Rebase vs Cherry-pick

- Rebase is for applying a whole branch.
- Cherry-pick is for applying selected commits.