**Core Git Concepts**

1. Explain the difference between Git and GitHub.

2. Describe Git's architecture (working directory, staging area, repository).

3. What is a `commit` in Git? What information does it contain?

4. Explain the difference between `git fetch` and `git pull`.

5. What is a `.gitignore` file? Give examples of when you'd use it.

**Basic Commands**

6. How do you initialize a new Git repository?

7. Explain: `git add`, `git commit`, `git push`, `git clone`.

8. How do you check the status of your working directory?

9. How do you view your commit history?

10. How do you discard changes in your working directory?

**Branching & Merging**

11. How do you create, list, and delete branches?

12. Explain the difference between `git merge` and `git rebase`. When would you use each?

13. What is a \*\*fast-forward merge\*\*?

14. How do you resolve a merge conflict?

15. What is `git cherry-pick`?

**Undoing Changes**

16. How do you amend the most recent commit?

17. Explain the differences between:

- `git reset --soft`, `--mixed`, and `--hard`

- `git revert` vs. `git reset`

18. How do you recover a deleted branch?

**Advanced Git**

19. What is \*\*Git Stash\*\*? When would you use it?

20. Explain \*\*detached HEAD state\*\*. How do you fix it?

21. What is \*\*reflog\*\*? How can it save you from mistakes?

22. Describe \*\*submodules\*\* and alternatives (like subtrees).

**GitHub-Specific**

23. Explain the \*\*GitHub Flow\*\* workflow.

24. What is a \*\*Pull Request (PR)\*\*? How is it different from a \*\*Merge Request (MR)\*\*?

25. How do you \*\*squash commits\*\* in a PR?

26. What are \*\*GitHub Actions\*\*? Give a use case.

27. How do you \*\*fork\*\* a repository? Why would you do this?

**Collaboration & Best Practices**

28. What is \*\*semantic commit messaging\*\*? Why is it useful?

29. How would you review a colleague’s PR effectively?

30. Describe strategies for keeping long-running branches up-to-date.

31. Why should you avoid `git push --force` on shared branches?

**Scenario-Based Questions**

32. You accidentally committed sensitive data (e.g., API keys). How would you remove it?

33. Your `git log` shows 5 commits, but you want to combine them into one. How?

34. How would you revert a commit that was pushed to a shared branch?

35. `git pull` results in a merge conflict. How do you resolve it?

**DevOps Integration**

36. How would you integrate Git with CI/CD pipelines?

37. What are \*\*protected branches\*\* in GitHub? Why use them?

38. Explain \*\*branch protection rules\*\*.

**Bonus (Advanced)**

39. How does Git store data internally (blobs, trees, commits)?

40. What is the \*\*Git object hash\*\* (SHA-1) derived from?