Basic Concepts

1. **What is the difference between Git and GitHub?**
    - Git is a version control system.
    - GitHub is a cloud platform for hosting Git repositories.
2. **What is the difference between a commit, push, pull, and fetch?**
    - `commit`: Saves changes locally.
    - `push`: Sends local commits to the remote repo.
    - `pull`: Fetch + merge changes from remote.
    - `fetch`: Gets changes from remote, but doesn't apply them.
3. **What is a branch? Why use it?**
    - A branch is a parallel version of your code. Used to develop features, fix bugs, etc., without affecting the main codebase.

Common Commands Confusion

4. **What's the difference between `merge` and `rebase`?**
    - `merge`: Combines histories, keeping all commits.
    - `rebase`: Rewrites commit history for a cleaner, linear timeline.
5. **When should I use `stash`?**
    - When you need to switch branches but don’t want to commit your current changes yet.
6. **What does `git reset` do?**

- It undoes commits:
  - `--soft`: Keep changes staged.
  - `--mixed`: Keep changes in working directory (unstaged).
  - `--hard`: Discard all changes.

Troubleshooting Issues

7. **How do I undo a commit?**
   - Use `git reset`, `git revert`, or `git checkout` depending on the situation.
8. **How do I resolve a merge conflict?**
   - Manually edit the conflicting files, mark them as resolved (`git add`), then commit.
9. **What does “detached HEAD” mean?**
   - You're not on a branch — you're viewing a specific commit. Any changes won’t be saved to a branch unless you create one.

Working with Remotes

10. **How do I sync my fork with the original repo?**
    - Add the original repo as a remote (often `upstream`) and pull/merge/rebase from it.
11. **How do I clone a specific branch?**
    - `git clone -b branch-name <repo-url>`
12. **Why is `git pull` not updating my files?**

    - Maybe you're not on the right branch, or your local branch has diverged from the remote.

Others

13. **How do I remove a file from the repo but keep it locally?**
    - `git rm --cached filename`
14. **What is `.gitignore` and how does it work?**
    - It tells Git which files/folders to ignore and not track.
15. **How do I recover a deleted branch or commit?**
    - Use `git reflog` to find the lost commit and restore it.

Branching & Workflow Confusion

16. **What is the difference between `origin/main` and `main`?**
    - `main` is your local branch.
    - `origin/main` is the remote-tracking branch.
17. **Should I use Git Flow, GitHub Flow, or trunk-based development?**
    - It depends on your team and release needs:
        - Git Flow: suited for release-heavy workflows.
        - GitHub Flow: ideal for continuous delivery.
        - Trunk-based: minimal branches, rapid iteration.

18. **How do I rename a branch (locally and remotely)?**

- Locally: `git branch -m old-name new-name`

  Remotely:

  ```
  git push origin new-name
  git push origin --delete old-name
  ```

Conflicts, Errors & Recovery

19. **How do I abort a merge or rebase?**
    - Merge: `git merge --abort`
    - Rebase: `git rebase --abort`
20. **What to do if I committed to the wrong branch?**
    - `git cherry-pick` or `git stash` + switch branch + `git stash pop`
21. **How do I fix a broken or messed-up merge?**
    - Use `git reset`, `git reflog`, or re-merge from a clean state.
22. **How do I resolve “non-fast-forward” errors?**
    - Use `git pull --rebase` or force push if appropriate: `git push -f`

Cleaning & Managing History

23. **How do I squash commits?**
    - Use interactive rebase: `git rebase -i HEAD~N`

24. **What’s the safest way to clean up old branches?**

    - `git branch -d branch-name` (only if merged)
    - `git branch -D` (force delete)

25. **How do I remove sensitive files from commit history?**

    - Use `git filter-branch`, `git rebase`, or tools like `BFG Repo-Cleaner`

26. **How can I make a shallow clone?**

    - `git clone --depth=1 <repo-url>`

Remote Access & Security

27. **Why am I being asked for my username/password or token repeatedly?**

    - You may need to use SSH or a credential helper.
    - GitHub has deprecated password authentication; use a **Personal Access Token**.

28. **How do I switch from HTTPS to SSH for remote URLs?**

    - `git remote set-url origin git@github.com:user/repo.git`

Collaboration Issues

29. **How do I prevent pushing to `main`?**

    - Use branch protection rules (e.g., on GitHub) or pre-push Git hooks.

30. **How do I force everyone to follow a PR/code review process?**

    - Enforce pull requests and enable branch protection in your Git hosting platform.

31. **How can I handle multiple people working on the same file?**

- Use feature branches and merge frequently to reduce conflicts.

Advanced Tools and Use Cases

32. **What is the `reflog` and when should I use it?**

- It's a history of all HEAD changes — great for recovering lost commits or branches.

33. **What's the difference between `git log`, `git reflog`, and `git show`?**

- `git log`: shows commit history.
- `git reflog`: shows all moves of HEAD (even deleted commits).
- `git show`: displays one commit's details.

34. **How do I use Git hooks?**

- Place executable scripts in `.git/hooks/` for events like pre-commit, pre-push, etc.

35. **How can I see who made changes to a specific line?**

- `git blame filename`

36. **What are submodules and how do they work?**

- They let you include other Git repos inside a repo.
- Use `git submodule add <url>` and manage updates separately.

Files & Tracking

37. **Why is Git not tracking my file?**

    - It may be in `.gitignore`, or you haven't added it with `git add`.

38. **How do I track changes to a file that was previously ignored?**

    Remove it from `.gitignore`, then use:
    `git add -f filename`

39. **How do I view the difference between staged and unstaged changes?**

    - Staged: `git diff --cached`

    - Unstaged: `git diff`

40. **What is the difference between `git clean`, `git reset`, and `git checkout` for removing changes?**

    - `git clean`: removes **untracked files**

    - `git reset`: unstages changes or moves commits

    - `git checkout -- file`: restores file to last committed version

History & Commit Manipulation

41. **How can I edit an older commit message?**
    - Use:
      `git rebase -i HEAD~N`
    - Then change `pick` to `reword` for the desired commit.

42. **How do I split a single commit into multiple commits?**

    - `git reset` the commit and recommit the changes in parts.

43. **How do I combine multiple repositories into one?**

- Use `git remote add` + `git fetch` + `git merge --allow-unrelated-histories`

Can I keep Git history but remove all files?

```
git rm -r *

git commit -m "Clean slate"
```

Staging Area & Partial Commits

45. **How can I stage only part of a file?**

- Use:
  `git add -p`

  This lets you choose hunks interactively.

46. **What happens if I commit without staging?**

- Nothing is committed. Git only commits what's been added with `git add`.

47. **How do I unstage a file but keep the changes?**

- Use:
  `git reset HEAD filename`

Config, Hooks, and Internals

48. **How do I change the default branch name?**

- After creating the repo:

```
git branch -m main
```

- `git push -u origin main`

49. **What's inside the `.git` folder?**

- All metadata: commits, branches, logs, config, objects.

50. **How can I set different user.name/email for different repos?**

- Use: `git config user.name "Your Name"`
- `git config user.email "you@example.com"`
- Run this inside the target repo for a local config.

51. **How can I create a template for commits (like JIRA ticket prefix)?**

- Use `commit.template` in Git config and pre-fill messages.

Remote Workflows & CI

52. **How do I check if my local branch is up-to-date with the remote?**

- Use:

```
git fetch
git status
```

53. **Why does Git say "everything up to date" but I don't see changes?**

- You're likely not on the correct branch, or you need to pull explicitly.

54. **How do I rollback a deployment (CI/CD) using Git?**

- Use:
```
git checkout <last-good-commit>
```
- `git push origin HEAD --force`

Conceptual Confusions

55. **What's the difference between a remote and a remote-tracking branch?**

- Remote: the actual hosted repo (`origin`)
- Remote-tracking branch: local snapshot of the remote branch (`origin/main`)

56. **What is a "fast-forward" merge vs "no-ff"?**

- Fast-forward: just moves the pointer forward (no extra commit)
- No-ff: always creates a merge commit, even if not strictly needed

57. **Why does Git sometimes refuse to delete a branch?**

- Because it's the current branch or unmerged. Use `-D` to force.

58. **Is it safe to force push?**

- Only if:
  - You know what you're doing.
  - You're the only one working on the branch.
  - Or the team agrees on using it (e.g., to rewrite history on PRs).