Basic Git Commands

1. **git init**

Initializes a new Git repository in the current directory.

Example:

```
mkdir my-project

cd my-project

git init
```

This command creates a git directory where Git stores all metadata for the repository.

2. **git clone <url>**

Clones a remote repository to your local machine.

Example:

```
git clone https://github.com/user/repo.git
```

This creates a copy of the repository on your local machine with all files and commit history.

3. **git status**

Displays the status of the working directory and staging area.

Example:

```
git status
```

Shows which files have been modified, staged for commit, or are untracked.

4. **git add <file>**

Stages changes in a specified file for the next commit.

Example:

```
git add file.txt
```

Adds file.txt to the staging area.

5. **git add .**

Stages all changes in the current directory and its subdirectories.

Example:

```
git add .
```

Stages all new, modified, and deleted files.

6. **git commit -m "<message>"**

Commits the staged changes with a descriptive message.

Example:

```
git commit -m "Fix bug in login feature"
```

Records the staged changes to the repository with the message "Fix bug in login feature".

7. **git log**

Shows the commit history of the current branch.

Example:

```
git log
```

Displays a list of commits with details like commit ID, author, date, and message.

8. **git diff**

Shows the differences between the working directory and the index or between different commits.

Example:

```
git diff
```

Displays changes in files that have not yet been staged.

9. **git reset <file>**

Unstages a file, keeping the changes in the working directory.

Example:

```
git reset file.txt
```

Removes file.txt from the staging area but retains changes in the working directory.

10. **git rm <file>**

Removes a file from the working directory and stages the removal.

Example:

```
git rm file.txt
```

Deletes file.txt and stages this change for commit.

Branching and Merging

11. **git branch**

Lists all local branches or creates a new branch if a name is provided.

Example:

```
git branch
```

```
git branch new-branch
```

Lists all local branches or creates a branch named new-branch.

12. **git checkout <branch>**

Switches to the specified branch.

Example:

```
git checkout new-branch
```

Changes the current working branch to new-branch.

13. **git switch <branch>**

Switches to the specified branch (alternative to git checkout).

Example:

```
git switch new-branch
```

Switches to new-branch.

14. **git checkout -b <branch>**

Creates a new branch and switches to it.

Example:

```
git checkout -b feature-x
```

Creates a new branch feature-x and switches to it.

15. **git merge <branch>**

Merges changes from the specified branch into the current branch.

Example:

```
git merge feature-x
```

Integrates changes from feature-x into the current branch.

16. **git rebase <branch>**

Re-applies commits from the current branch on top of the specified branch.

Example:

```
git rebase main
```

Applies commits from the current branch on top of main.

17. **git branch -d <branch>**

Deletes the specified branch (only if it has been fully merged).

Example:

```
git branch -d old-branch
```

Deletes old-branch if it has been merged.

18. **git branch -D <branch>**

Forcefully deletes the specified branch, even if it has unmerged changes.

Example:

```
git branch -D old-branch
```

Forcefully deletes old-branch.

Remote Repositories

19. **git remote -v**

Lists remote repositories associated with the local repository.

Example:

```
git remote -v
```

Shows URLs for fetching and pushing.

20. **git remote add <name> <url>**

Adds a new remote repository.

Example:

```
git remote add origin https://github.com/user/repo.git
```

Adds a remote named origin with the specified URL.

21. **git fetch <remote>**

Fetches changes from the specified remote repository but does not merge them.

Example:

```
git fetch origin
```

Downloads changes from origin without merging them.

22. **git pull <remote> <branch>**

Fetches and merges changes from the specified remote branch into the current branch.

Example:

```
git pull origin main
```

Fetches and merges changes from origin/main into the current branch.

23. **git push <remote> <branch>**

Pushes changes from the local branch to the specified remote branch.

Example:

```
git push origin main
```

Pushes the local main branch to origin.

24. **git push -u <remote> <branch>**

Pushes changes and sets the upstream branch for the current branch.

Example:

```
git push -u origin feature-x
```

Pushes feature-x to origin and sets it as the upstream branch.

25. **git push --force**

Forces a push to the remote repository, potentially overwriting changes.

Example:

```
git push --force origin main
```

Forcefully pushes to origin/main, overwriting remote changes if necessary.

Advanced Commands

26. **git stash**

Temporarily saves changes in the working directory that are not yet staged for commit.

Example:

```
git stash
```

Saves changes to a stash and reverts the working directory to the last commit.

27. **git stash pop**

Applies the most recent stash and removes it from the stash list.

Example:

```
git stash pop
```

Applies the latest stashed changes and removes the stash from the list.

28. **git stash list**

Lists all stashed changes.

Example:

```
git stash list
```

Shows a list of stashes with their identifiers.

29. **git tag**

Lists, creates, or deletes tags. Tags are used to mark specific points in history (like releases).

Example:

```
git tag

git tag v1.0
```

Lists tags or creates a tag named v1.0.

30. **git tag -d <tag>**

Deletes a tag.

Example:

```
git tag -d v1.0
```

Deletes the tag v1.0.

31. **git rebase -i <commit>**

Interactively rebases commits starting from a specific commit, allowing for editing, reordering, or squashing commits.

Example:

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

Opens an interactive editor to rebase the last 3 commits.

32. **git cherry-pick <commit>**

Applies the changes from a specific commit to the current branch.

Example:

```
git cherry-pick abc1234
```

Applies the commit with ID abc1234 to the current branch.

33. **git blame <file>**

Shows who last modified each line of a file.

Example:

```
git blame file.txt
```

Displays the commit and author information for each line in file.txt.

34. **git reflog**

Shows a log of all actions and branch movements, useful for recovering lost commits.

Example:

```
git reflog
```

Lists all the recent actions in the repository, including commits, checkouts, and more.

Summary

- **Basic Commands**: git init, git clone, git status, git add, git commit, git log, git diff, git reset, git rm.
- **Branching and Merging**: `git branch