**Git commands:**

### **1. git init**

* **Usage**: Initializes a new Git repository.

**Syntax**: git init

**Example**: git init my-project

### **2. git clone**

* **Usage**: Clones an existing Git repository to your local machine.

**Syntax**: git clone <repository-url>

**Example**: git clone <https://github.com/user/repo.git>

### **3. git status**

* **Usage**: Shows the status of your working directory and staging area.

**Syntax**: git status

**Example**: git status

### **4. git add**

* **Usage**: Stages files for commit.

**Syntax**: git add <file(s)>

**Example**: git add index.html

git add .

**5. git commit**

* **Usage**: Commits the staged changes to the repository.

**Syntax**: git commit -m "commit message"

**Example**: git commit -m "Add new feature"

### **6. git log**

* **Usage**: Shows the commit history of the current branch.

**Syntax**: git log

**Example**: git log --oneline

### **7. git diff**

* **Usage**: Displays the differences between your working directory and the index (staging area).

**Syntax**: git diff

### **8. git diff --staged**

* **Usage**: Shows the changes that have been staged but not yet committed.

**Syntax**: git diff --staged

### **9. git reset**

* **Usage**: Unstaged files or reverts the repository to a previous state.

**Syntax**: git reset <file>

**Example**: git reset index.html

### **10. git reset --hard**

* **Usage**: Resets the working directory and index to the last commit, discarding changes.

**Syntax**: git reset --hard

**11. git branch**

* **Usage**: Lists all the branches in the repository.

**Syntax**: git branch

**12. git branch <branch-name>**

* **Usage**: Creates a new branch.

**Syntax**: git branch <branch-name>

**Example**: git branch feature-xyz

### **13. git checkout <branch-name>**

* **Usage**: Switches to the specified branch.

**Syntax**: git checkout <branch-name>

**Example**: git checkout feature-xyz

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

* **Usage**: Creates and switches to a new branch in a single command.

**Syntax**: git checkout -b <branch-name>

**Example**: git checkout -b feature-xyz

### **15. git merge <branch-name>**

* **Usage**: Merges the specified branch into the current branch.

**Syntax**: git merge <branch-name>

**Example**: git merge feature-xyz

### **16. git rebase <branch-name>**

* **Usage**: Re-applies commits from one branch onto another.

**Syntax**: git rebase <branch-name>

**Example**: git rebase master

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

* **Usage**: Deletes a local branch.

**Syntax**: git branch -d <branch-name>

**Example**: git branch -d feature-xyz

### **18. git push origin <branch-name>**

* **Usage**: Pushes the local branch to the remote repository.

**Syntax**: git push origin <branch-name>

**Example**: git push origin feature-xyz

### **19. git pull origin <branch-name>**

* **Usage**: Fetches changes from the remote repository and merges them into the local branch.

**Syntax**: git pull origin <branch-name>

**Example**: git pull origin master

### **20. git fetch**

* **Usage**: Fetches changes from the remote repository without merging them.

**Syntax**: git fetch

**Example**: git fetch

### **21. git remote add <name> <url>**

* **Usage**: Adds a new remote repository.

**Syntax**: git remote add <name> <url>

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

### **22. git remote -v**

* **Usage**: Lists the remote repositories associated with the current repository.

**Syntax**: git remote -v

**Example**: git remote -v

### **23. git push**

* **Usage**: Pushes changes to the remote repository.

**Syntax**: git push

**Example**: git push

### **24. git pull**

* **Usage**: Fetches changes from the remote and merges them into your current branch.

**Syntax**: git pull

**Example**: git pull

### **25. git remote remove <name>**

* **Usage**: Removes a remote repository from the configuration.

**Syntax**: git remote remove <name>

**Example**: git remote remove origin

### **26. git remote rename <old-name> <new-name>**

* **Usage**: Renames an existing remote repository.

**Syntax**: git remote rename <old-name> <new-name>

**Example**: git remote rename origin upstream

### **27. git stash**

* **Usage**: Stashes the changes in the working directory.

**Syntax**: git stash

**Example**: git stash

### **28. git stash pop**

* **Usage**: Applies the most recent stash and removes it from the stash list.

**Syntax**: git stash pop

**Example**: git stash pop

### **29. git stash list**

* **Usage**: Lists all stashed changes.

**Syntax**: git stash list

**Example**: git stash list

### **30. git stash drop**

* **Usage**: Removes a specific stash from the list.

**Syntax**: git stash drop <stash-id>

* **Example**: git stash drop stash@{0}

### **31. git stash apply**

* **Usage**: Applies a specific stash without removing it from the list.

**Syntax**: git stash apply <stash-id>

**Example**: git stash apply stash@{0}

### **32. git revert**

* **Usage**: Reverts a commit by creating a new commit that undoes the changes.

**Syntax**: git revert <commit-id>

**Example**: git revert abc123

### **33. git reset**

* **Usage**: Resets the current branch to a specific commit.

**Syntax**: git reset <commit-id>

**Example**: git reset abc123

### **34. git checkout -- <file>**

* **Usage**: Discards changes in the working directory and reverts the file to the last commit.

**Syntax**: git checkout -- <file>

**Example**: git checkout -- index.html

### **35. git clean -f**

* **Usage**: Removes untracked files.

**Syntax**: git clean -f

**Example**: git clean -f

### **36. git clean -fd**

* **Usage**: Removes untracked files and directories.

**Syntax**: git clean -fd

**Example**: git clean -fd

### **37. git tag**

* **Usage**: Lists all tags.

**Syntax**: git tag

**Example**: git tag

### **38. git tag <tag-name>**

* **Usage**: Creates a lightweight tag at the current commit.

**Syntax**: git tag <tag-name>

**Example**: git tag v1.0.0

### **39. git tag -a <tag-name> -m "<message>"**

* **Usage**: Creates an annotated tag with a message.

**Syntax**: git tag -a <tag-name> -m "<message>"

**Example**: git tag -a v1.0.0 -m "Release version 1.0"

### **40. git push origin <tag-name>**

* **Usage**: Pushes a specific tag to the remote repository.

**Syntax**: git push origin <tag-name>

**Example**: git push origin v1.0.0

### **41. git push --tags**

* **Usage**: Pushes all local tags to the remote repository.

**Syntax**: git push --tags

**Example**: git push --tags

### **42. git tag -d <tag-name>**

* **Usage**: Deletes a tag locally.

**Syntax**: git tag -d <tag-name>

**Example**: git tag -d v1.0.0

### **43. git config --global user.name "<name>"**

* **Usage**: Sets the global username for Git.

**Syntax**: git config --global user.name "<name>"

**Example**: git config --global user.name "John Doe"

### **44. git config --global user.email "<email>"**

* **Usage**: Sets the global email for Git.

**Syntax**: git config --global user.email "<email>"

**Example**: git config --global user.email "you@example.com"

### **45. git config --list**

* **Usage**: Lists all Git configuration settings.

**Syntax**: git config --list

**Example**: git config --list

### **46. git config --global core.editor <editor>**

* **Usage**: Sets the default text editor for Git commit messages.

**Syntax**: git config --global core.editor <editor>

**Example**: git config --global core.editor "vim"

### **47. git version**

* **Usage**: Displays the current version of Git.

**Syntax**: git version

**Example**: git version

### **48. git help <command>**

* **Usage**: Displays help information for a specific Git command.

**Syntax**: git help <command>

**Example**: git help commit

### **49. git cherry-pick <commit>**

* **Usage**: Applies the changes from a specific commit onto the current branch.

**Syntax**: git cherry-pick <commit-id>

**Example**: git cherry-pick abc123

### **50. git bisect**

* **Usage**: Helps you find which commit introduced a bug (binary search).

**Syntax**: git bisect start

git bisect bad

git bisect good <commit-id>

**Example**: git bisect start

git bisect bad

git bisect good abc123