GIT Commands:

Basic Commands:

1. **git init:** Initializes a new Git repository in the current directory.
2. **git clone [url]:** Clones an existing Git repository from a remote URL.
3. **git status:** Shows the current state of the working directory and staging area.
4. **git add [file/directory]:** Adds changes to the staging area.
5. **git commit -m "message":** Commits changes to the local repository with a message.
6. **git log:** Shows the commit history.
7. **git diff:** Shows the difference between commits, files, or branches.
8. **git show [commit]:** Shows the details of a specific commit.
9. **git reset --hard HEAD~1:** Resets the current HEAD to the previous commit.

Branching and Merging:

10. **git branch:** Lists all local branches.
11. **git branch [branch-name]:** Creates a new branch.
12. **git checkout [branch-name]:** Switches to a different branch.
13. **git checkout -b [branch-name]:** Creates a new branch and switches to it.
14. **git merge [branch-name]:** Merges the specified branch into the current branch.
15. **git rebase [branch-name]:** Rebases the current branch onto the specified branch.
16. **git merge --no-ff [branch-name]:** Merges a branch with a merge commit, even if a fast-forward merge is possible.

Remote Repositories:

17. **git remote add [alias] [url]:** Adds a remote repository.
18. **git remote -v:** Lists all remote repositories.
19. **git fetch [remote]:** Fetches changes from a remote repository.
20. **git pull [remote] [branch]:** Fetches and merges changes from a remote branch.
21. **git push [remote] [branch]:** Pushes local changes to a remote branch.
22. **git push --force [remote] [branch]:** Forces a push to a remote branch, overwriting existing commits.

Working with the Staging Area:

23. **git add -p:** Adds changes to the staging area interactively.
24. **git add .:** Adds all changes in the working directory to the staging area.
25. **git reset [file]:** Removes a file from the staging area.
26. **git reset HEAD [file]:** Removes a file from the staging area and the working directory.
27. **git stash:** Temporarily saves changes to the working directory.
28. **git stash pop:** Restores the most recent stashed changes.
29. **git stash list:** Lists all stashed changes.
30. **git stash clear:** Clears all stashed changes.

Inspecting History:

31. **git log --oneline:** Shows a compact log format.
32. **git log --graph:** Shows a graphical representation of the commit history.
33. **git log --follow [file]:** Shows the history of a file, including renames.
34. **git log --since="2 weeks":** Shows commits from the past two weeks.
35. **git log --author="John Doe":** Shows commits by John Doe.
36. **git log --grep="bugfix":** Shows commits containing the word "bugfix".
37. **git reflog:** Shows a log of all changes to the HEAD pointer.
38. **git show [commit-hash]:** Shows the details of a specific commit.

Manipulating History:

39. **git revert [commit-hash]:** Reverts a specific commit.
40. **git cherry-pick [commit-hash]:** Applies a specific commit to the current branch.
41. **git rebase -i [base-commit]:** Interactively rebase a series of commits.
42. **git reset --soft HEAD~1:** Unstages the last commit.
43. **git reset --mixed HEAD~1:** Unstages and resets the last commit.
44. **git reset --hard HEAD~1:** Unstages, resets, and discards the last commit.

Tagging:

45. **git tag [tag-name]:** Creates a tag at the current commit.
46. **git tag -a [tag-name] -m "message":** Creates an annotated tag with a message.
47. **git tag:** Lists all tags.
48. **git show [tag-name]:** Shows the details of a specific tag.
49. **git push origin --tags:** Pushes local tags to the remote repository.

Configuration:

50. **git config --global user.name "[name]":** Sets your global user name.
51. **git config --global user.email "[email]":** Sets your global user email.
52. **git config --global core.editor "vim":** Sets your preferred editor.
53. **git config --global color.ui true:** Enables colorized output.
54. **git config --global push.default simple:** Sets the default push behavior.

Advanced Topics:

55. **git bisect:** Helps find the commit that introduced a bug.
56. **git blame [file]:** Shows who last modified each line of a file.
57. **git filter-branch:** Rewrites the history of a repository.
58. **git submodule:** Manages submodules within a repository.
59. **git worktree:** Manages multiple working directories for a single repository.
60. **git archive:** Creates an archive of a repository or a specific commit.
61. **git shortlog:** Summarizes the commit history by author.
62. **git rev-list:** Lists commits based on various criteria.

63. **git diff --name-only:** Shows a list of files that have changed.
64. **git diff --stat:** Shows a summary of changes.
65. **git diff --patch:** Shows a detailed patch format of changes.
66. **git diff --word-diff:** Highlights word-level changes.
67. **git log -S "keyword":** Shows commits that introduce or remove a keyword.
68. **git log --date-order:** Sorts commits by date.
69. **git log --topo-order:** Sorts commits topologically.
70. **git log --pretty=format:"%h %an %s":** Customizes the log output format.
71. **git log --decorate:** Shows references (tags, branches) associated with commits.
72. **git log --follow --name-only:** Shows a list of files that have been renamed or copied.
73. **git log --since="yesterday" --until="today":** Shows commits between yesterday and today.
74. **git log --no-merges:** Excludes merge commits from the log.
75. **git log --first-parent:** Shows only the first parent of each merge commit.
76. **git log --no-commit-id:** Hides commit IDs from the log output.
77. **git log --abbrev-commit:** Shortens commit IDs.
78. **git log --date=short:** Shows the date in short format.
79. **git log --date=relative:** Shows the date relative to the current time.
80. **git log --date=local:** Shows the date in local time zone.
81. **git log --date=iso:** Shows the date in ISO 8601 format.
82. **git log --date=rfc2822:** Shows the date in RFC 2822 format.
83. **git log --date=format:%Y-%m-%d:** Customizes the date format.
84. **git log --patch-with-stat:** Shows a detailed patch format with statistics.
85. **git log --numstat:** Shows numerical statistics of changes.
86. **git log --shortstat:** Shows a summary of changes.
87. **git log --name-status:** Shows the status of changed files.
88. **git log --diff-filter=A:** Shows only added files.
89. **git log --diff-filter=D:** Shows only deleted files.
90. **git log --diff-filter=M:** Shows only modified files.
91. **git log --diff-filter=R:** Shows only renamed files.
92. **git log --diff-filter=C:** Shows only copied files.
93. **git log --diff-filter=ACMRT:** Shows all types of changes.
94. **git log --decorate --oneline --graph --all:** Shows a comprehensive log with decorations.
95. **git log --pretty=format:"%h %s (%an)" --graph --decorate --oneline --all:** Customizes the log output with a concise format.
96. **git log --follow --name-only --pretty=format:"