1) **\`ls\`:** Lists files and directories in the current directory.
2) **\`git config --global user.name\`:** Sets the global Git username.
3) **\`git config --global user.email\`:** Sets the global Git email.
4) **\`history\`:** Displays the command history.
5) **\`code .\`:** Opens the current directory in Visual Studio Code.
6) **\`git init\`:** Initializes a new Git repository.
7) **\`ls -lart\`:** Lists files and directories with detailed information, including hidden files, sorted by modification time in reverse order.
8) **\`git status\`:** Displays the state of the working directory and the staging area.
9) **\`git add index.html\`**: Adds the file \`index.html\` to the staging area.
10) **\`git rm --cached index.html\`:** Removes the file \`index.html\` from the staging area.
11) **\`git commit -m "initial commit**"\`: Commits the staged changes with the message "initial commit".

12) **`git commit`:** Opens the default text editor to write a commit message for the staged changes.

13) **`touch contact.html`:** Creates an empty file named `contact.html`.

14) **`git add -A`:** Adds all changes (new, modified, and deleted files) to the staging area.

15) **`git checkout contact.html`:** Discards changes in the working directory for `contact.html`.

16) **`git checkout -f`:** Discards all local changes and resets the working directory to match the last commit.

17) **`git log`:** Displays the commit history.

18) **`git log -p -1`**: Shows the commit history with patch (diff) for the last commit.

19) **`git diff`**: Shows the changes between the working directory and the staging area.

20) **`git diff --staged`**: Shows the changes between the staging area and the last commit.

21) **`touch waste.html`**: Creates an empty file named `waste.html`.

22) **`clear`:** Clears the terminal screen.

23) **`git status -s`**: Displays the status of the working directory and the staging area in a short format.

24) **`touch mylog.log`**: Creates an empty file named `mylog.log`.

25) **`git branch`**: Lists all the branches in the repository.

26) **`git branch feature1`**: Creates a new branch named `feature1`.

27) **`git commit -m "index.html using feature1"`**: Commits the staged changes with the message "index.html using feature1".

28) **`git checkout master`**: Switches to the `master` branch.

29) **`git merge feature1`**: Merges the `feature1` branch into the current branch.

30) **`git checkout -b feature2`**: Creates a new branch named `feature2` and switches to it.

31) **`git checkout master`**: Switches to the `master` branch.

32) **`git merge feature2`**: Merges the `feature2` branch into the current branch.

33) **`git log`**: Displays the commit history.

34) **git push origin main –force** : If you're certain that you want to overwrite the remote branch with your local branch (be careful as this can cause loss of work for others), you can force the push: