Git is a distributed version control system that allows developers to manage and track changes to their codebase. There are several key actions and commands in Git that developers commonly use:

1. **git init**: Initializes a new Git repository in the current directory, creating a hidden **.git** folder that stores the repository's configuration and history.
2. **git clone**: Creates a copy of a remote Git repository on your local machine. It's often used to start working on an existing project.
3. **git add**: Adds changes or new files to the staging area, preparing them to be committed in the next snapshot.
4. **git commit**: Records a snapshot of the changes in the staging area and creates a new commit with a unique identifier. Developers often include a commit message to describe the changes.
5. **git status**: Displays the current status of your working directory, including any untracked, modified, or staged files.
6. **git log**: Shows a chronological history of commits in the current branch, including commit messages, authors, and timestamps.
7. **git branch**: Lists all branches in the repository. The currently checked-out branch is indicated with an asterisk.
8. **git checkout**: Switches between branches or commits. You can use it to create new branches, switch branches, or move between different commit snapshots.
9. **git merge**: Combines changes from one branch into another. It's commonly used to integrate feature branches into the main development branch.
10. **git pull**: Fetches changes from a remote repository and merges them into the current branch. It's often used to update your local branch with changes from a remote repository.
11. **git push**: Sends your local commits to a remote repository, updating the remote branch with your changes.
12. **git fetch**: Downloads changes from a remote repository without automatically merging them into your current branch. This is useful for reviewing changes before merging.
13. **git remote**: Lists and manages remote repositories that your local repository interacts with, such as GitHub or GitLab.
14. **git reset**: Allows you to unstage or unmodify changes in your working directory. It can also be used to move the branch pointer to a different commit.
15. **git rebase**: Reorganizes the commit history by applying your changes on top of another branch or commit, creating a linear history.
16. **git stash**: Temporarily saves changes that are not ready to be committed so that you can switch branches or perform other actions without losing your work.
17. **git tag**: Creates lightweight or annotated tags to mark specific commits as important milestones, such as releases or significant changes.
18. **git blame**: Shows the author and revision history of each line in a file, helping to identify who made specific changes.
19. **git cherry-pick**: Selectively applies a single commit from one branch to another, allowing you to pick and choose specific changes.
20. **git diff**: Displays the differences between different commits, branches, or the working directory.

These are some of the most common Git actions and commands used by developers to manage and collaborate on code in a version-controlled environment. Git offers a wide range of features and flexibility for tracking changes, branching, and collaborating with others on software projects.