**1. Git Configuration**

**These commands help set up and configure Git for your system.**

* **git config --global user.name "Your Name"**  
  Sets the global username for Git commits.
* **git config --global user.email "your\_email@example.com"**  
  Sets the global email for Git commits.
* **git config --list**  
  Displays the current Git configuration settings.
* **git config --global core.editor "code --wait"**  
  Sets the default editor (e.g., VS Code) for Git operations.

**2. Git Repository Initialization & Cloning**

**These commands create or clone a repository.**

* **git init**  
  Initializes a new Git repository in the current directory.
* **git clone <repository\_url>**  
  Creates a copy (clone) of a remote repository to your local machine.
* **git clone <repository\_url> <directory>**  
  Clones a repository into a specified directory.

**3. Git Basic Commands**

**These commands help track and manage files.**

* **git status**  
  Shows the current state of the working directory and staging area.
* **git add <file>**  
  Adds a file to the staging area.
* **git add .**  
  Adds all changed and new files to the staging area.
* **git commit -m "Commit message"**  
  Commits staged changes with a descriptive message.
* **git commit -am "Commit message"**  
  Stages and commits all tracked files in one command.
* **git rm <file>**  
  Removes a file from both the working directory and staging area.
* **git mv <old\_name> <new\_name>**  
  Renames a file and stages the change.

**4. Git Branching & Merging**

**These commands manage different versions of your code.**

* **git branch**  
  Lists all branches in the repository.
* **git branch <branch\_name>**  
  Creates a new branch.
* **git checkout <branch\_name>** *(Legacy Command)*  
  Switches to the specified branch.
* **git switch <branch\_name>**  
  Recommended way to switch branches.
* **git switch -c <new\_branch>**  
  Creates and switches to a new branch.
* **git merge <branch\_name>**  
  Merges the specified branch into the current branch.
* **git branch -d <branch\_name>**  
  Deletes a local branch.
* **git branch -D <branch\_name>**  
  Forcibly deletes a local branch.
* **git log --oneline --decorate --graph --all**  
  Shows a graphical representation of commits across branches.

**5. Git Remote Commands**

**These commands help manage remote repositories.**

* **git remote -v**  
  Lists remote repositories connected to the local repo.
* **git remote add origin <repository\_url>**  
  Connects a remote repository (named **origin**) to your local repo.
* **git remote remove <remote\_name>**  
  Removes a remote repository.
* **git fetch origin**  
  Retrieves the latest changes from a remote repository but does not merge them.
* **git pull origin <branch>**  
  Fetches and merges the latest changes from a remote branch.
* **git push origin <branch>**  
  Pushes local commits to a remote repository.
* **git push -u origin <branch>**  
  Pushes and sets the upstream branch (tracks remote changes).
* **git push --force**  
  Forcefully pushes changes, overwriting remote history (use with caution).

**6. Git Stashing**

**These commands temporarily save changes without committing them.**

* **git stash**  
  Saves all uncommitted changes temporarily.
* **git stash list**  
  Lists all stashed changes.
* **git stash apply**  
  Applies the last stashed changes without deleting them.
* **git stash pop**  
  Applies and removes the last stash.
* **git stash drop**  
  Deletes the last stashed change.

**7. Git Reset & Revert**

**These commands undo changes.**

* **git reset <file>**  
  Unstages a file but keeps changes in the working directory.
* **git reset --soft HEAD~1**  
  Moves the last commit to the staging area (does not delete changes).
* **git reset --mixed HEAD~1**  
  Moves the last commit to the working directory (removes commit but keeps changes).
* **git reset --hard HEAD~1**  
  Deletes the last commit and all changes permanently.
* **git revert <commit\_hash>**  
  Creates a new commit that undoes changes from a specified commit.

**8. Git Log & History**

**These commands view commit history.**

* **git log**  
  Displays the commit history.
* **git log --oneline**  
  Shows a condensed one-line version of the commit history.
* **git log --author="Name"**  
  Shows commits made by a specific author.
* **git log --grep="keyword"**  
  Searches commit messages for a keyword.
* **git diff**  
  Shows the differences between the working directory and the last commit.
* **git diff --staged**  
  Shows changes staged for the next commit.

**9. Git Tagging**

**These commands create and manage tags.**

* **git tag**  
  Lists all tags in the repository.
* **git tag -a v1.0 -m "Version 1.0"**  
  Creates an annotated tag.
* **git push origin --tags**  
  Pushes all local tags to the remote repository.
* **git tag -d v1.0**  
  Deletes a local tag.

**10. GitHub Collaboration**

**These commands help interact with GitHub.**

* **git fork** *(Done via GitHub UI)*  
  Creates a copy of someone else’s repository on GitHub.
* **git clone <forked\_repository\_url>**  
  Clones the forked repository.
* **git remote add upstream <original\_repository\_url>**  
  Adds the original repository as an upstream remote.
* **git fetch upstream**  
  Retrieves updates from the original repository.
* **git merge upstream/main**  
  Merges upstream changes into your local branch.
* **git push origin <branch>**  
  Pushes your changes to your forked repository.
* **git pull-request** *(Done via GitHub UI)*  
  Requests to merge changes from your fork into the original repository.

**11. Git Cleanup & Maintenance**

**These commands keep repositories clean.**

* **git gc**  
  Cleans up unnecessary files in the repository.
* **git prune**  
  Removes unreachable objects.
* **git fsck**  
  Verifies the integrity of the repository.

**12. Git Aliases**

**These commands create shortcuts for Git commands.**

* **git config --global alias.co checkout**  
  Creates a shortcut for git checkout.
* **git config --global alias.st status**  
  Creates a shortcut for git status.
* **git config --global alias.cm "commit -m"**  
  Creates a shortcut for git commit -m.

**Final Notes**

* Git is a distributed version control system.
* GitHub is a remote hosting service for Git repositories.
* Use git help <command> to get detailed help on any Git command.