Basic Git Commands

(1) `git init` – Initialize a new Git repository

(2) `git clone [url]` – Clone a repository from a remote URL

(3) `git add [file]` – Add a file to the staging area

(4) `git add .` – Add all files to the staging area

(5) `git commit -m "message"` – Commit changes with a message

(6) `git status` – Show the working tree status

(7) `git log` – Show commit history

(8) `git show [commit]` – Show information about a specific commit

(9) `git diff` – Show unstaged changes

(10) `git diff --staged` – Show staged changes

Branching and Merging

(11) `git branch` – List all branches

(12) `git branch [branch-name]` – Create a new branch

(13) `git checkout [branch-name]` – Switch to a specific branch

(14) `git switch [branch-name]` – Another way to switch branches

(15) `git checkout -b [branch-name]` – Create and switch to a new branch

(16) `git merge [branch-name]` – Merge a branch into the current branch

(17) `git branch -d [branch-name]` – Delete a branch

(18) `git branch -D [branch-name]` – Force delete a branch

(19) `git rebase [branch-name]` – Reapply commits on top of another base

(20) `git merge --abort` – Abort a merge if there are conflicts

Remote Repository Commands

(21) `git remote -v` – List remote repositories

(22) `git remote add [name] [url]` – Add a new remote repository

(23) `git remote remove [name]` – Remove a remote repository

(24) `git push origin [branch-name]` – Push changes to a remote branch

(25) `git push -u origin [branch-name]` – Push and set upstream branch

(26) `git pull origin [branch-name]` – Fetch and merge remote changes

(27) `git fetch origin` – Download objects and refs from remote

(28) `git remote set-url origin [url]` – Change the remote repository URL

(29) `git remote rename [old-name] [new-name]` – Rename a remote

(30) `git push --force` – Force push changes (use with caution)

Undoing Changes

(31) `git reset [file]` – Unstage a file

(32) `git reset --soft HEAD~1` – Undo the last commit but keep changes

(33) `git reset --hard HEAD~1` – Undo the last commit and discard changes

(34) `git revert [commit]` – Create a new commit that undoes a previous commit

(35) `git checkout -- [file]` – Discard changes in a file

(36) `git restore [file]` – Restore a file to its last committed state

(37) `git restore --staged [file]` – Unstage a file

(38) `git clean -f` – Remove untracked files

(39) `git clean -fd` – Remove untracked files and directories

(40) `git reflog` – Show a log of all actions performed

Stashing Changes

(41) `git stash` – Temporarily save changes

(42) `git stash list` – Show list of stashes

(43) `git stash apply` – Apply the latest stash

(44) `git stash apply stash@{n}` – Apply a specific stash

(45) `git stash drop` – Delete the latest stash

(46) `git stash clear` – Remove all stashes

(47) `git stash pop` – Apply and delete the latest stash

(48) `git stash branch [branch-name]` – Create a branch from a stash

(49) `git stash save "message"` – Save stash with a message

(50) `git stash show -p` – Show stash details

Tagging

(51) `git tag` – List all tags

(52) `git tag [tag-name]` – Create a new tag

(53) `git tag -a [tag-name] -m "message"` – Create an annotated tag

(54) `git show [tag-name]` – Show tag details

(55) `git push origin [tag-name]` – Push a tag to the remote repository

(56) `git push --tags` – Push all tags to remote

(57) `git tag -d [tag-name]` – Delete a local tag

(58) `git push origin --delete [tag-name]` – Delete a remote tag

(59) `git checkout [tag-name]` – Switch to a tag

(60) `git describe --tags` – Show the latest tag reachable from HEAD

Git Configuration

(61) `git config --global user.name "Your Name"` – Set global username

(62) `git config --global user.email "your@email.com"` – Set global email

(63) `git config --global core.editor "vim"` – Set default editor

(64) `git config --global color.ui true` – Enable colored output

(65) `git config --global alias.st status` – Create an alias for a command

(66) `git config --list` – Show current configuration

(67) `git config --global --unset user.name` – Remove a configuration

(68) `git config --global credential.helper cache` – Cache credentials

(69) `git config --global init.defaultBranch main` – Set default branch name

(70) `git help [command]` – Show help for a command

Advanced Commands

(71) `git bisect start` – Start a bisect session

(72) `git bisect bad` – Mark the current commit as bad

(73) `git bisect good [commit]` – Mark a commit as good

(74) `git bisect reset` – End a bisect session

(75) `git blame [file]` – Show who changed each line

(76) `git grep "text"` – Search for text in files

(77) `git cherry-pick [commit]` – Apply a commit to another branch

(78) `git submodule add [url] [path]` – Add a submodule

(79) `git submodule update --init --recursive` – Update submodules

(80) `git fsck` – Check for corruption

Git Hooks

(81) `git hook list` – Show available hooks

(82) `git hook add pre-commit` – Add a pre-commit hook

(83) `git hook add post-merge` – Add a post-merge hook

(84) `git hook remove pre-commit` – Remove a hook

(85) `git commit --no-verify` – Bypass hooks

Git Worktree

(86) `git worktree add [path] [branch]` – Add a new worktree

(87) `git worktree list` – List all worktrees

(88) `git worktree remove [path]` – Remove a worktree

(89) `git worktree prune` – Clean up stale worktrees

(90) `git worktree move [old-path] [new-path]` – Move a worktree

Miscellaneous Commands

(91) `git archive --format=zip -o repo.zip HEAD` – Create a zip archive

(92) `git shortlog` – Summarize commits by author

(93) `git count-objects -v` – Show repository size details

(94) `git show-branch` – Display branch information

(95) `git verify-commit [commit]` – Verify commit integrity

(96) `git verify-tag [tag]` – Verify tag integrity

(97) `git cherry -v` – Show commits not yet merged

(98) `git name-rev [commit]` – Find a branch name for a commit

(99) `git notes add -m "message"` – Add notes to a commit

(100) `git notes show [commit]` – Show notes for a commit