

Git Commands

SETUP AND CONFIGURATION

1. **git**: git is a distributed version control system for code management.
Options: -v, -h, -P, -p
Usage: git add [file names]
git clone [git repository URL]
2. **config**: Helps in setting up the repository and global options.
Options: -replace-all, -get, -add
Usage: git config -global user.name [username]
git config -list
3. **help**: Provides help information about Git.
Options: -a, -c, -g
Usage: git help -all
git status -help

GETTING AND CREATING PROJECTS

1. **init**: Initialize an empty git repository or reinitialize an existing one.
Options: -q, -bare
Usage: git init
2. **clone**: Get the remote repository into the directory
Options: -l, -s
Usage: git clone [git repository URL]

BASIC SNAPSHOTTING

1. **add**: To stage changes.
Options: -f, -v
Usage: git add [file name], . [all changes]
2. **status**: Know the changes between commit, commits, working tree, etc.
Options: -s, -v, -long, -b
Usage: git status
3. **diff**: Display differences between commits, the working tree, or branches.

Options: -color, -[commit1] [commit2], -cached

Usage: git diff, git diff [commit1] [commit2], git diff -color

4. **commit**: Record changes to the repository.
Options: -m, -a, -v
Usage: git commit -m "[commit message]", git commit -am "[commit message]"
5. **reset**: Reset current HEAD to the specified state.
Options: -soft, -mixed, -hard
Usage: git reset -soft [commit], git reset -hard HEAD^

BRANCHING AND MERGING

1. **branch**: List, create, or delete branches.
Options: -r, -d, -m
Usage: git branch
2. **checkout**: Switch branches or restore working tree files.
Options: -b, -B, -force
Usage: git checkout [branch-name], git checkout -b [branch-name]
3. **merge**: Join two or more development histories together.
Options: -squash, -abort, -commit
Usage: git merge [branch-name] - merge a branch into current branch
4. **log**: Display commit logs.
Options: -oneline, -graph
Usage: git log, git log -oneline, git log -graph
5. **stash**: Stash changes in a dirty working directory away.
Options: save, list, pop, apply
Usage: git stash save, git stash list, git stash pop, git stash apply
6. **worktree**: Manage multiple working trees associated with a single Git repository.
Options: list, prune
Usage: git worktree list, git worktree prune

SHARING AND UPDATING

1. fetch: Fetch command is used to retrieve changes from a remote repository without merging them into your local branch.
Options: `-all`, `-a`, `-force`
Usage: `git fetch`, `git fetch origin`, `git fetch -all`, `git fetch -force`
2. pull: Used to fetch and merge changes from a remote repository into the current branch.
Options: `-rebase`, `-squash`
Usage: `git pull`, `git pull origin main`, `git pull -rebase`
3. push: Command is used to upload local repository content to a remote repository. It transfers commits, branches, and tags from your local repository to the remote repository.
Options: `-force`, `-u`, `-all`
Usage: `git push`, `git push origin main`, `git push -force`, `git push -all`
4. remote: To manage connections to remote repositories. It allows you to view, add, rename, and remove remote repositories.
Options: `show`, `rename [old] [new]`, `add [name] [url]`
Usage: `git remote add origin [url]`, `git remote remove origin`, `git remote show`

INSPECTION AND COMPARISON

1. show: shows one or more things [commits, tags. etc]
Options: `-format=[oneline — short — medium — full, -pretty]`
Usage: `git show -oneline`
2. log: provide commit info
Options: `-source`, `-full-diff`
Usage: `git log`

PATCHING

1. apply: To apply changes from a patch file to your working directory or index without committing them.
Options: `-check`, `-index`, `-reverse`
Usage: `git apply [pathname.patch]`, `git apply -check [pathname.patch]`
2. cherry-pick: To apply the changes introduced by existing commits to the current branch.

Options: `-e`, `-s`, `-x`

Usage: `git cherry-pick [commit-hash]`, `git cherry-pick -e [commit-hash]`

3. rebase: Command is used to reapply commits from one branch onto another branch.

Options: `-i`, `-x`, `-p`

Usage: `git rebase [branch]`, `git rebase -i [branch]`

4. revert: To reverse the changes introduced by a specific commit or a range of commits.

Options: `-n`, `-e`, `-s`

Usage: `git revert [commit-hash]`, `git revert -n [commit-hash]`

DEBUGGING

1. grep: Find matching pattern

Options: `-a`, `-i`

Usage: `git grep -i [text]`

GUIDES

1. gitignore: Intentionally untrack some files

Usage: `*.exe` [`.gitignore`]

EMAIL

1. request-pull: Get pending changes summary.

Options: `-p`

Usage: `git request-pull [version number] [URL] [branch name]`

EXTERNAL SYSTEMS

1. svn: Operate between Subversion repository and git.

Options: `-s`, `-no-metadata`, `-parent`

Usage: `git svn rebase`

ADMINISTRATION

1. clean
2. filter-branch
3. archive
4. bundle

SERVER ADMIN

1. daemon: A git repository server.
Options: `-export-all`, `-base-path`
Usage: `git daemon -export-all -base-path=.`
2. update-server-info: To help dumb server update auxiliary info file.
Options: `-f`
Usage: `git update-server-info`

PLUMBING COMMANDS

1. `commit-tree`
2. `show-ref`
3. `update-index`
4. `revert`