echo "Some String" | git hash-object --stdin	Accepts the string in the stdin and creates a hash for it.
echo "Some String " | git hash-object --stdin -w	Same as above but, additionally writes it to the repository. If not inside a repo, it will give an error.
git branch -m main	Change the name of the current branch to 'main'.
git cat-file <hash> -t	Check the type of the repo object associated with the hash
git cat-file <hash> -p	Pretty print the data in the repo object.
git init	Initialise a folder as a git repository
git status	Check the status of all contents in the repository, modified, uncommitted, conflicts, etc.
git commit -m "<commit message>"	Commits the files to the repository.
git cat-file -p <commit hash>	Commit message that is stored in the repo and the hash of the text is created used as the key.
git count-objects	Count the no. of objects presently in the git database. This includes blobs (files) and trees (folders), both.
git branch	Lists all the branches, local and otherwise.
git branch <branch-name>	Create a new branch, named <branch-name>, from this point, without switching to it right away.
git merge <subject-branch>	Merge the subject-branch into the current branch. Current branch is the merge destination. This is where the new commit will be made.
git rebase <subject-branch>	Changes the base of the current branch from where it is to the HEAD of the subject-branch.
git tag <tag-name>	Creates a simple tag on the current commit i.e. without a tag object. This is very similar to a branch reference, other than that it is fixed to the commit and doesn't move to the latest commit later.
git tag <tag-name> -a -m "<annotation>"	Creates an annotated tag. It is a combination of a tag reference and a tag type database object, which contains the annotations.
git clone <addr-of-git-repo> [<destination-folder>]	Creates a local copy of the repo found at <addr-of-git-repo>. Automatically sets the relationship between the local and the remote repos, for later syncs.
git remote add origin <remote-repo>	Add the link as an origin for the local repo
git branch -M main	If default branch is still named as 'master', change it to 'main'.
git push --all origin	Push all branches to the remote repo
git push --tags origin	Push all the tags to the remote repo
git branch	shows only the locally available branch.
git branch -a	Shows all branches, local and remote.
git branch -r	Shows only remote branches.
git checkout --track origin/ideas	Pull down a branch; Repeat for each branch

git pull -all	ensure all tags are cloned as well.
git show-ref main	Shows all the references to the branch 'main'
git fetch	Fetches the latest changes on the remote repo .
git pull	'git fetch' + 'git merge'
git switch <branchname>	Switch to the branch <branchname>, and update Work Area accordingly. Limited to switching between commit objects referenced by branch-heads.
git checkout <location>	Similar to 'git switch' but in addition to branch-heads, checkout can checkout the code for commits directly, by using branch references, HEAD reference or even commit hash, anywhere in the repo.
git checkout -b <branchname>	Create a new branch name <branchname> and switch to it in a single go.
git log [--oneline] [--graph] [--decorate]	Show the history of current branch in a diagrammatic form on the command line.
git diff	Compare WA and Index
git diff --cached	Compare Index and repository
git rm <filename>	Remove file from WA and/or Index; No change to Repo -f : remove from both WA and Index --cached : remove only from Index; equivalent to unstage
git mv <fname1> <fname2>	Rename/Move a file and stage the change in a single go.
git reset <commit-hash>	Moves the branch from one commit to another directly. Always changes the Repo state. --soft: no change in Index or WA --mixed: Change Index but not WA --hard: Change Index and WA;
git stash --include-untracked	Stash all changes (staged and unstaged, even untracked) to stash.
git stash push	Similar to above
git stash list	List the changesets stashed currently
git stash pop	Apply the top changes in stash to my WA+Index, and remove it from the stash.
git stash apply <stash-name>	Apply the stash specified by <stash-name> to WA+Index, but do not remove it from the stash yet.
git stash drop <stash-name>	Remove the changeset specified by <stash-name>, without applying on WA & Index.
git <command> --patch	Can be used with add, checkout, stash, reset to effect the changes in hunk level granularity, rather than file level.
git cherry-pick <commit-hash>	Create a copy of the changes in the commit <commit-hash>, and apply on current branch to create a new commit here. Note that

	his brings in ONLY the changes in the specified commit and nothing before or after it.
git bisect …	Cuts down the git history in halves to help identify a bad commit start: begin the program good <good-commit-hash>: Latest known good commit bad <bad-commit-hash>: Earliest known bad commit good : Inform the prog. that current commit is a good one bad: Informs the prog. that current commit is a bad one reset: Stop the prog. and return state to what it was before executing the bisect.
git config --global alias.co checkout and similar…	Allows to create a short aliases for commands regularly used. "https://git-scm.com/book/en/v2/Git-Basics-Git-Aliases"
git show <commit-ref>	detailed info about the commit
git blame <file-name>	Shows the latest commit history for each line in the file.
git rebase -i	Start interactive rebase.
git filter-repo --path <file-name>	remove everything from repo history, excluding the history of <file-name>. use '--invert-paths' to reverse the selection.
git revert <commit-hash>	Create a new commit reversing/nullifying the changes in the specified commit.