Git Commands :

**Git Setup**

Create a **new Git repository** from an existing directory:

git init [directory]

**Clone** **a repository** (local or remote via HTTP/SSH):

git clone [repo / URL]

Clone a repository **into a specified folder** on your local machine:

git clone [repo / URL] [folder]

**Git Configuration**

Attach an **author name** to all commits that will appear in the version history:

git config --global user.name "[your\_name]"

Attach an**email address** to all commits by the current user:

git config --global user.email "[email\_address]"

Apply Git’s automatic **command line coloring** which helps you keep track and revise repository changes:

git config --global color.ui auto

Create a **shortcut (alias)** for a Git command:

git config --global alias.[alias\_name] [git\_command]

**Note:**Git requires you to type out the entire command to perform actions. Setting shortcuts for commonly used commands can speed up and simplify development. For example, you can use the alias **st** for the status command by typing the command: **git config --global alias.st status**

Set a **default text editor**:

git config --system core.editor [text\_editor]

Open Git’s **global configuration file**:

git config --global --edit

**Managing Files**

Show the **state of the current directory** (list staged, unstaged, and untracked files):

git status

List the **commit history** of the current branch:

git log

List **all commits from all branches**:

git log --all

**Compare two branches** by showing which commits from the first branch are missing from the second branch:

git log [branch1]..[branch2]

Examine the difference between the **working directory and the index**:

git diff

Explore the difference between the **last commit and the index**:

get diff --cached

See the difference between the **last commit and the working directory**:

get diff HEAD

Display the**content and metadata** of an object (blob, tree, tag or commit):

git show [object]

**Git Branches**

List **all branches** in the repository:

git branch

List all **remote branches**:

git branch -aa

**Create a new branch** under a specified name:

git branch [branch]

**Switch to a branch** under a specified name (if it doesn’t exist, a new one will be created):

git checkout [branch]

**Note**: For a more detailed tutorial on working with Git branches, you can refer to our article on [How to Create a New Branch](https://phoenixnap.com/kb/git-create-new-branch) or [How to Switch Branches in Git](https://phoenixnap.com/kb/how-to-switch-branches-git).

**Delete** a local branch:

git branch -d [branch]

[Rename a branch](https://phoenixnap.com/kb/how-to-rename-git-branch-local-remote) you are currently working in:

git branch -m [new\_branch\_name]

**Merge** the specified branch with the current branch:

git merge [branch]

**Making Changes**

**Stage changes** for the next commit:

git add [file/directory]

**Stage everything** in the directory for an initial commit:

git add .

**Commit staged snapshots** in the version history with a descriptive message included in the command:

git commit -m "[descriptive\_message]"

**Undoing Changes**

**Undo changes** in a file or directory and create a new commit with the [git revert](https://phoenixnap.com/kb/git-revert-last-commit) command:

git revert [file/directory]

[Unstage a file](https://phoenixnap.com/kb/git-unstage-files) without overwriting changes:

git reset [file]

Undo any changes introduced **after the specified commit**:

git reset [commit]

**Show untracked files** which will be removed when you run **git clean** (do a dry run):

git clean -n

**Remove** untracked files:

git clean -f

**Rewriting History**

**Replace the last commit** with a combination of the staged changes and the last commit combined:

git commit --amend

**Rebase the current branch** with the specified base (it can be a branch name, tag, reference to a HEAD, or a commit ID):

git rebase [base]

List **changes made to the HEAD** of the local repository:

git reflog

**Remote Repositories**

Create a new **connection to a remote repository** (give it a name to serve as a shortcut to the URL):

git remote add [name] [URL]

**Fetch a branch** from a remote repository:

git fetch [remote\_repo] [branch]

**Fetch a repository** and merge it with the local copy:

git pull [remote\_repo]

**Push a branch** to a remote repository with all its commits and objects:

git push [remote\_repo] [branch]