# 🔥All-in-One GIT Cheatsheet

| Command | Use Case | Syntax | Scenario |
|---|---|---|---|
| `git init` | Initialize a new Git repository in a directory. | `git init` | Starting version control for a new project. |
| `git clone` | Clone an existing repository to your local machine. | `git clone <repo-url>` | Downloading a GitHub repository to work on it locally. |
| `git add` | Stage changes for the next commit. | `git add <file>` | Adding modified files to prepare for a commit. |
| `git commit` | Save staged changes as a snapshot in the repository. | `git commit -m "<message>"` | Saving progress with a meaningful commit message like "Added user login feature." |
| `git status` | Check the status of the repository (e.g., staged, unstaged changes). | `git status` | Checking what files are modified before committing. |
| `git log` | View the commit history of the repository. | `git log` | Reviewing all past changes and commit messages. |
| `git branch` | List branches or create a new branch. | `git branch` / `git branch <name>` | Listing available branches or creating a feature branch like `feature-login`. |
| `git checkout` | Switch to another branch or restore files. | `git checkout <branch>` | Switching to `feature-login` branch to work on a specific feature. |
| `git merge` | Combine changes from one branch into another. | `git merge <branch>` | Merging `feature-login` branch into the `main` branch. |
| `git pull` | Fetch and merge changes from the remote repository to the local branch. | `git pull <remote> <branch>` | Pulling the latest updates from the remote repository (e.g., `origin/main`). |

| | | | |
|---|---|---|---|
| `git push` | Upload local commits to the remote repository. | `git push <remote> <branch>` | Pushing commits to GitHub or any remote platform for sharing code with others. |
| `git remote` | Manage remote connections. | `git remote add <name> <url>` | Adding a new remote repository to push and pull changes. |
| `git fetch` | Download changes from a remote repository without merging them. | `git fetch <remote>` | Fetching updates to preview before merging with local changes. |
| `git stash` | Temporarily save changes you don't want to commit yet. | `git stash` | Saving your current changes when switching branches for a quick fix. |
| `git stash pop` | Apply the stashed changes back to your working directory. | `git stash pop` | Restoring stashed changes after fixing a bug in another branch. |
| `git diff` | Compare changes between files or commits. | `git diff` | Viewing what has been modified before staging the changes. |
| `git rebase` | Reapply commits on top of another branch. | `git rebase <branch>` | Updating `feature-login` branch with the latest changes from `main`. |
| `git reset` | Undo changes by resetting the staging area or commits. | `git reset <mode> <file>` | Unstaging a file mistakenly added to the staging area with `git reset HEAD <file>`. |
| `git revert` | Undo a specific commit by creating a new commit. | `git revert <commit-hash>` | Reverting a previous commit that introduced a bug. |
| `git rm` | Remove files from the repository. | `git rm <file>` | Deleting a file from the repository and staging the deletion. |
| `git tag` | Mark a specific commit with a tag (e.g., release version). | `git tag <tag-name>` | Tagging a commit as `v1.0` for a project release. |

| | | | |
|---|---|---|---|
| `git blame` | Show who made the last change to each line of a file. | `git blame <file>` | Finding out who last edited a specific line in `app.py`. |
| `git show` | Display details of a specific commit, tag, or object. | `git show <commit-hash>` | Viewing changes introduced in a specific commit. |
| `git cherry-pick` | Apply specific commits from one branch to another. | `git cherry-pick <commit>` | Picking a critical bug fix commit to apply it directly to `main`. |
| `git config` | Set Git configuration options. | `git config --global <key> <value>` | Configuring your username and email for commits (`git config --global user.name "Anurag Srivastava"`). |
| `git clean` | Remove untracked files from the working directory. | `git clean -f` | Cleaning up unnecessary files before committing changes. |
| `git archive` | Create an archive (e.g., zip or tar) of the repository. | `git archive --format=<fmt> HEAD > archive-name` | Exporting the current state of the repository as a zip file. |
| `git bisect` | Perform a binary search to identify a bug-inducing commit. | `git bisect start` | Locating which commit caused a bug by iterating between good and bad states. |