10 Git Commands Every Beginner Developer Should Know

Starting with Git can feel scary. You may see a black terminal window and wonder where to begin. Git lets you track changes in your code and work with others. These ten commands will help you feel at home with Git. Each command has a clear explanation and an Example: you can try right away.

1. git init

git init turns any folder into a Git repository. A repository is where Git stores your project history.

What it does?
It creates a hidden folder called .git. This folder holds all the data Git needs.

How to use it

1. Open your terminal.
2. Change to your project folder with cd.
3. Run git init.

Example::

text

```
cd my_project_folder
git init
```

After this you can start tracking files in this folder.

2. git clone

git clone makes a copy of an existing repository. You often use it to get code from a remote server.

What it does
It copies the code and the full history. You can work on it locally.

How to use it

1. Find the repository URL you want to copy.
2. Run git clone followed by that URL.

Example:

text

```
git clone https://github.com/username/project.git
```

This creates a new folder named project with all the files.

3. git status

git status shows you the state of your working folder. It tells you which files are new, changed, or ready to be committed.

What it does
It lists tracked and untracked files. It also shows staged changes.

How to use it

1. Run git status in your repository folder.

Example:

text

```
git status
```

Look at the output to see what Git sees.

4. git add

git add stages files for the next commit. Staging means you prepare files to be saved in your project history.

What it does
It tells Git which changes you want to include in your next snapshot.

How to use it

1. Run git add followed by a file name or . to add all files.

Example:

text

```
git add index.html
git add .
```

The first line stages index.html. The second line stages all changed files.

5. git commit

git commit saves your staged changes in a snapshot. Each commit has a message that explains what you did.

What it does
It takes the staged changes and stores them in the repository history.

How to use it

1. Run git commit with a message flag and a message in quotes.

Example:

text

```
git commit -m "Add header section to homepage"
```

This creates a new commit with your staged changes.

6. git push

git push sends your commits to a remote repository. This lets others see your work.

What it does
It uploads your local commits to the server you cloned from or another you set up.

How to use it

1. Run git push along with the remote name and branch name.

Example:

text

```
git push origin main
```

origin is the default remote name. main is the default branch name in new repositories.

7. git pull

git pull fetches the latest changes from a remote repository and merges them into your local branch.

What it does
It updates your local code with new commits from others.

How to use it

1. Run git pull with the remote name and branch name.

Example:

text

```
git pull origin main
```

This gets any new commits from main on origin and applies them to your folder.

8. git branch

git branch lists and manages branches. Branches let you work on new features without affecting the main code.

What it does
Without arguments it shows all branches. With a name it creates a new branch.

How to use it

1. Run git branch to see all branches.
2. Run git branch branch_name to make a new branch.

Example:

text

```
git branch
git branch feature-login
```

The first line lists branches. The second line creates a branch called feature-login.

9. git checkout

git checkout switches your working folder to a different branch or commit.

What it does
It changes your files to match the branch or snapshot you choose.

How to use it

1. Run git checkout with a branch name.

Example:

text

```
git checkout feature-login
```

Now you work on feature-login. Your code reflects that branch.

10. git merge

git merge brings changes from one branch into another. You often merge a feature branch into main.

What it does
It combines commit histories and updates your current branch.

How to use it

1. Switch to the branch you want to merge into.
2. Run git merge with the branch you want to bring in.

Example:

text

```
git checkout main
git merge feature-login
```

This merges feature-login into main.

Bonus Tips and Common Mistakes to Avoid

- Always write clear commit messages. A short message like Fix typo helps you remember why you made the change.
- Avoid committing secrets like passwords. Use .gitignore to skip files you do not want to track.
- Do not forget to pull before you start working. This avoids merge conflicts with others.
- Clean up branches after you merge. Delete old branches with git branch -d branch_name.
- Practice in a test folder before you work on real projects.

You now know ten key Git commands. Try them step by step in your own project. With practice the command line will feel friendly. Git becomes a powerful tool in your workflow. Good luck and enjoy coding with Git!