step-by-step beginner-friendly guide to using Git to:

1. ✅ Create a new branch
2. ✅ Commit your current code to that branch
3. ✅ Push it to the remote repository
4. ✅ Create a **Pull Request (PR)** to merge into the main remote branch (e.g., `main` or `master`)

> 📝 **Assumptions**: - You already have Git installed and initialized in your project (`git init` done). - Your remote (e.g., on GitHub/GitLab/Bitbucket) is set up and named `origin` . - The default/main branch is called `main` (adjust if yours is `master`). - You have write permissions (or forked the repo if contributing to someone else's project).

🔧 Step-by-Step Guide

1️⃣ Check your current status & stage changes {#️⃣-check-your-current-status–stage-changes }

```
# See modified/new files
git status

# Add (stage) all current changes
git add .

# Or selectively add files:
# git add filename1.py filename2.js
```

2️⃣ Create and switch to a new branch {#️⃣-create-and-switch-to-a-new-branch }

💡 **Best practice**: Name branches descriptively (e.g., `feat/login-page` , `fix/header-bug` , `docs/update-readme`).

```
# Create and switch to a new branch in one command
git checkout -b your-branch-name

# Example:
git checkout -b feat/user-profile
```

> ✅ This creates a new branch *based on your current commit* (usually `main` HEAD) and switches to it.

3️⃣ Commit your code to the new branch {#️⃣-commit-your-code-to-the-new-branch }

```
# Commit the staged changes with a meaningful message
git commit -m "Add user profile page with basic styling"
```

✅ Good commit messages are: **concise**, **imperative**, and **descriptive** (e.g., `"fix: prevent null pointer in auth middleware"` or `"docs: update installation instructions"`)

4️⃣ Push the branch to the remote repository {#️⃣-push-the-branch-to-the-remote-repository }

```
# Push your new branch to origin (GitHub/GitLab/etc.)
git push -u origin your-branch-name

# Example:
git push -u origin feat/user-profile
```

> 🔔 `-u` (or `--set-upstream`) links your local branch to the remote one — future pushes can just use `git push` .

5️⃣ Create a Pull Request (PR) {#️⃣-create-a-pull-request-pr }

🔹 On GitHub:

1. Go to your repo on **github.com**.
2. You'll often see a banner: > *"Your recently pushed branches: `feat/user-profile` — [Compare & pull request]"* Click **Compare & pull request**.
   If not, go to the **Pull requests** tab → **New pull request** → choose:
   - **base:** `main` (target branch you want to merge *into*)
   - **compare:** `feat/user-profile` (your branch)
3. Fill in the PR details:
   - **Title**: Clear summary (e.g., *"feat: add user profile page"*)
   - **Description**: Explain *what*, *why*, and *how*. Include screenshots, issue links (e.g., `Closes #123`), or testing steps.
   - Assign reviewers if needed.
4. Click **Create pull request** ✅

🔹 On GitLab:

Same idea: **Repository > Merge Requests > New merge request** → choose source & target branches.

🔹 On Bitbucket:

Pull requests > Create pull request

6️⃣ (Optional but recommended) Get feedback & update {#️⃣-optional-but-recommended-get-feedback–update }

- Teammates may request changes.
- To update your PR, just make more commits on the same branch and push:

```
git add .
git commit -m "fix: address review comments on profile form"
git push  # (no need for -u again — upstream is set)
```

✅ The PR will update automatically.

7️⃣ Merge the PR ✅ {#️⃣-merge-the-pr- }

Once approved, click **Merge pull request** (GitHub) or **Merge** (GitLab/Bitbucket). You can choose to:

- ✅ **Squash and merge** (recommended for clean history),
- 🔄 **Rebase and merge**,
- or ➕ **Create a merge commit**.

> 🗑️ After merging, you may delete the branch (GitHub offers this option).

8️⃣ Sync your local `main` with the updated remote {#️⃣-sync-your-local-main-with-the-updated-remote }

```
# Switch back to main
git checkout main

# Pull the latest (including your merged changes)
git pull origin main

# Optional: delete local feature branch
git branch -d feat/user-profile
```

🎯 Bonus Tips

Tip	Command / Note
🔄 See all branches	`git branch -a`
📜 Check commit history	`git log --oneline`
🚫 Undo last commit (keep changes)	`git reset --soft HEAD~1`
🧹 Clean up stale remote branches	`git remote prune origin`