Git Pull Request Workflow Guide

Objective
Learn how to properly create and merge pull requests (PRs) using Git and GitHub.

---

1. Branching Structure
Use a clear branching model:
- main → stable production branch
- dev → integration branch
- feature/* → for new work or updates

Example:
```
git checkout -b dev
git push -u origin dev
git checkout -b feature/readme-update
git push -u origin feature/readme-update
```

---

2. Making Changes in Feature Branch
Edit files while on the feature branch:
```
notepad README.md
```
or (if using VS Code):
```
code README.md
```

Make real edits (e.g., add author info):
Author
K. Manikanta – Practicing Git branching and PR workflow

Stage and commit:
```
git add README.md
git commit -m "Added author section to README"
git push origin feature/readme-update
```

---

3. Creating the Pull Request
Go to GitHub → your repository → Pull Requests tab → **New Pull Request**

Select:
- Base: main (or dev)
- Compare: feature/readme-update

If you see “This branch is 1 commit ahead,” that means changes are detected.

Click **Create Pull Request**, add a title/description, and **Merge Pull Request**.

---

4. Post-Merge Cleanup
After merge:
```
git checkout dev
git pull origin dev
git checkout main
git pull origin main
```

Optional (delete merged feature branch):

```
git branch -d feature/readme-update
git push origin --delete feature/readme-update
```

---

5. Common Problems
- “There isn’t anything to compare” → Branches are identical (no new commits)
- “Everything up-to-date” after commit → You didn’t actually change file content

---

6. Best Practices
- Always branch from dev, not main.
- Commit often with meaningful messages.
- Use PRs even in solo projects to practice collaboration workflow.
- Delete merged branches to keep repo clean.

Summary
A pull request is a formal request to merge changes between branches. It’s essential for code reviews, version control discipline, and clean integration workflows in DevOps environments.