What is a Git branch, and how should a team use branches?

3 November 2026 · 4 min read · blogs.iictp.org/git-branching-for-beginners

A Git branch is a movable pointer to a line of commits. It lets you work on a change in isolation without disturbing the main line of code, then merge it back when it is ready. Branches are cheap in Git, so teams use them for almost every change.

TL;DR The short version

- A Git branch is a movable pointer to a line of commits, letting you work on a change without disturbing the main line.
- A simple workflow is: update main, create a branch named for the change, commit, and merge it back when ready.
- Keep branches short-lived, ideally hours or a few days, to keep merge conflicts small.

Read online, listen to it, or change the text size: https://blogs.iictp.org/git-branching-for-beginners

Why use a branch at all?

Without branches everyone edits the same line of history, and one person’s half-finished work can break everyone else. A branch gives each change its own space, so it can be built, tested and reviewed before it reaches the shared code.

What does a simple workflow look like?

1. Update your local copy of the main branch.
2. Create a branch named for the change, for example a short description of the fix.
3. Commit small, meaningful steps with clear messages.
4. Push the branch and open a pull request.
5. After review and passing checks, merge it and delete the branch.

What is the difference between merge and rebase?

A merge joins two lines of history and keeps a record of when they diverged. A rebase replays your commits on top of the latest main branch, giving a straight history. Both are valid. The usual rule is not to rebase commits that other people have already pulled.

What are merge conflicts?

A conflict happens when two branches changed the same lines and Git cannot decide which version to keep. You resolve it by editing the file, choosing or combining the changes, and committing the result. Short-lived branches that are merged often produce far fewer conflicts.

How long should a branch live?

As short as practical, ideally hours or a few days. Long-lived branches drift away from the main line and make merging painful, which is why many teams prefer small changes that merge frequently. Trunk-based development takes that idea furthest.

> Related credential: Associate Technical Practitioner. Explore the exam facts and syllabus.

Sources and further reading

1. Chacon, S. and Straub, B. Pro Git (2nd ed.), chapter 3, “Git Branching”. git-scm.com/book.

Published by IICTP, an independent technical credential body, on 3 November 2026. The current version of this post is at https://blogs.iictp.org/git-branching-for-beginners. More guides: blogs.iictp.org.