Rebase

Rebase as an Alternative to Merge

- Merging is definitely the easiest and most common way to integrate changes.
- But merging is not the only one: “Rebase” is an alternative means of integration.
- Rebasing is quite a bit more complex than merging

Rebase
– –
Understand merge first

Two possibilities

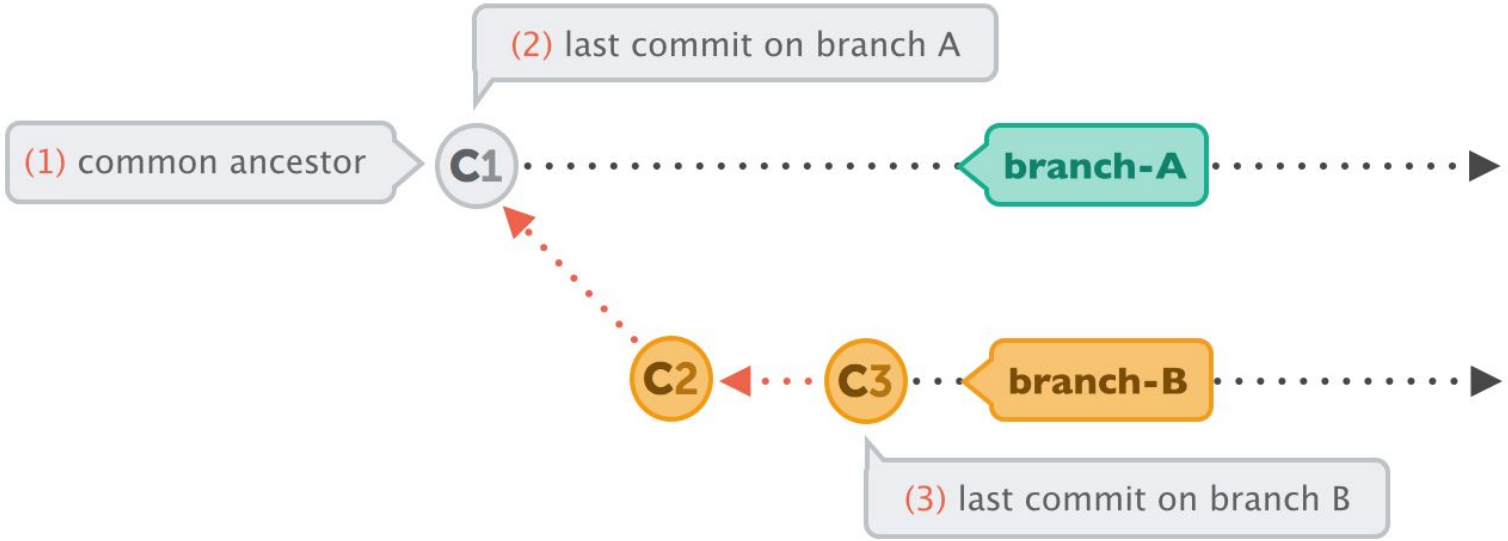
- Fast-Forward
- Merge Commit

Understand merge first : Fast-Forward

- In very simple cases, one of the two branches doesn‘t have any new commits since the branching happened - its latest commit is still the common ancestor.

Understand merge first : Fast-Forward

- Only one branch has new commits



Understand merge first : Fast-Forward

- In this case, performing the integration is dead simple
- Git can just add all the commits of the other branch on top of the common ancestor commit.
- In Git, this simplest form of integration is called a “fast-forward” merge. Both branches then share the exact same history.

Understand merge first : Fast-Forward

- Both branch have same history after fast-forward

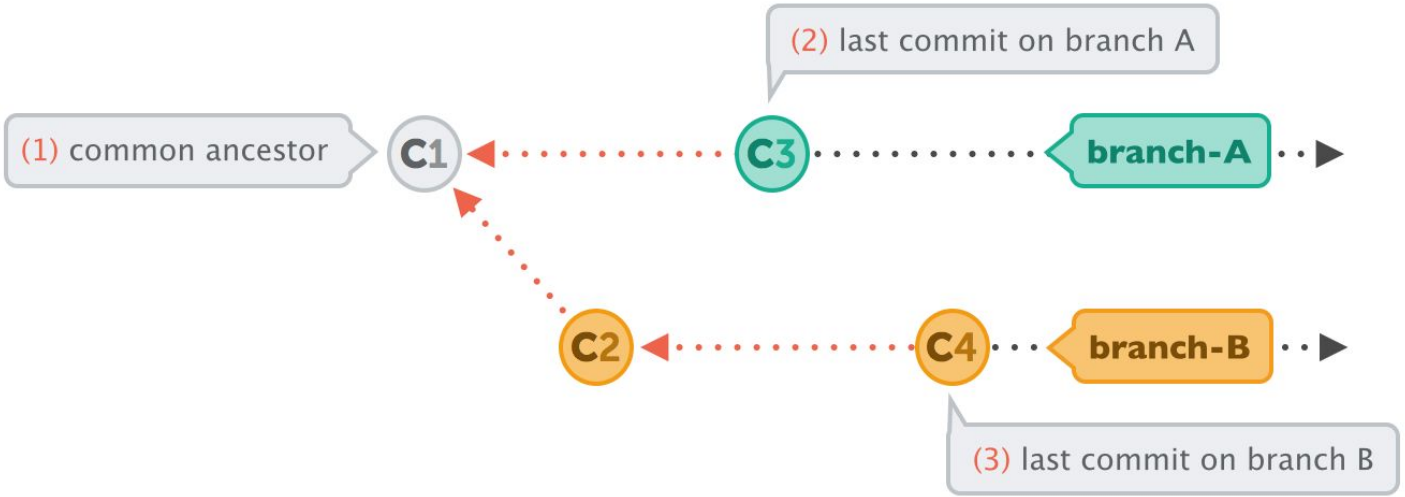
Demo

Understand merge first : Merge Commit

- In a lot of cases, however, both branches moved forward individually.
- And can have different commits

Understand merge first : Merge Commit

- Both branches have commits that are done after branch created

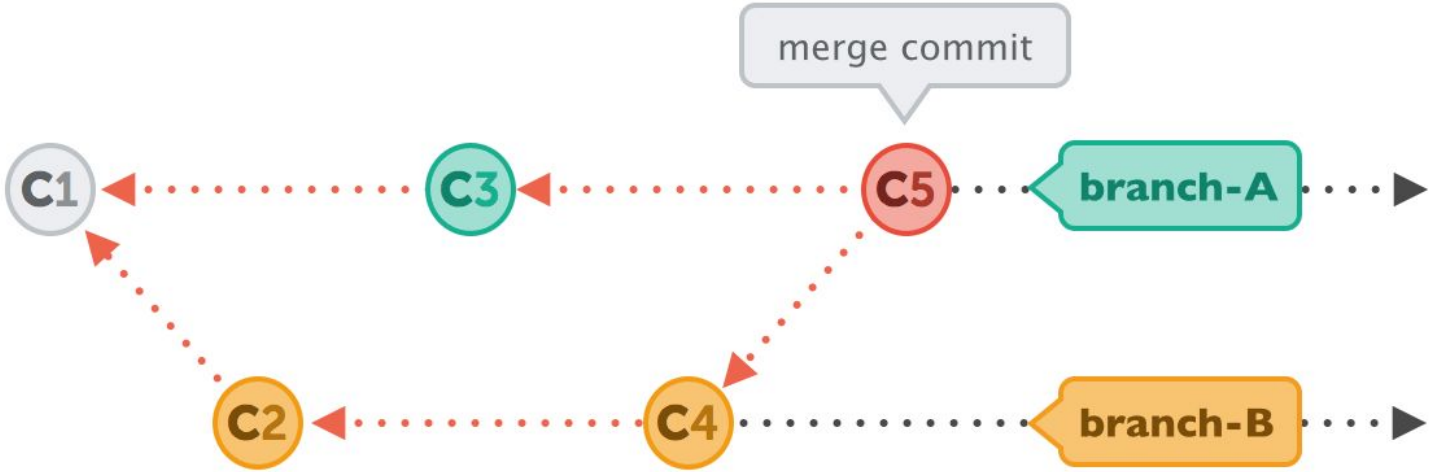


Understand merge first : Merge Commit

- To make an integration, Git will have to create a new commit that contains the differences between them - the merge commit.

Understand merge first : Merge Commit

- Git automatically created merge commit “C5”

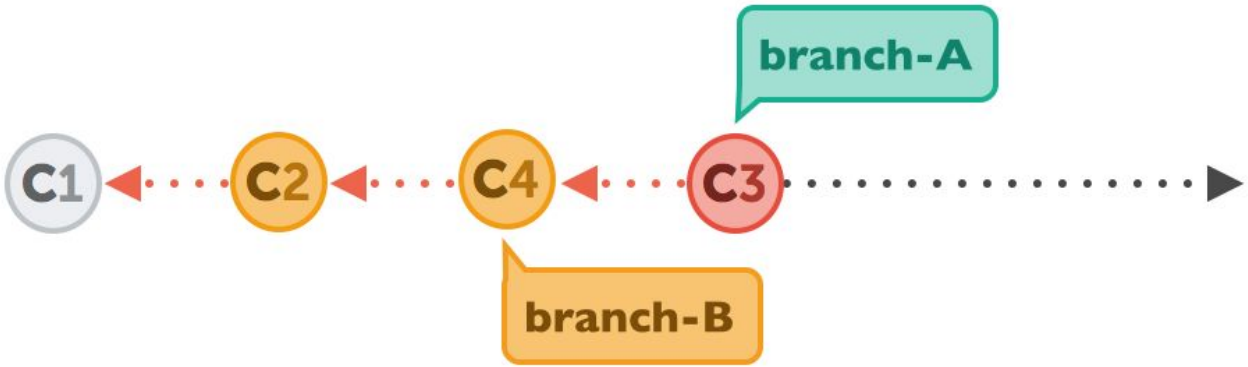


Demo

Rebase

- Sometimes we prefer to go without such automatic merge commits.
- We want the project's history to look as if it had evolved in a single, straight line.
- No indication remains that it had been split into multiple branches at some point.

Rebase

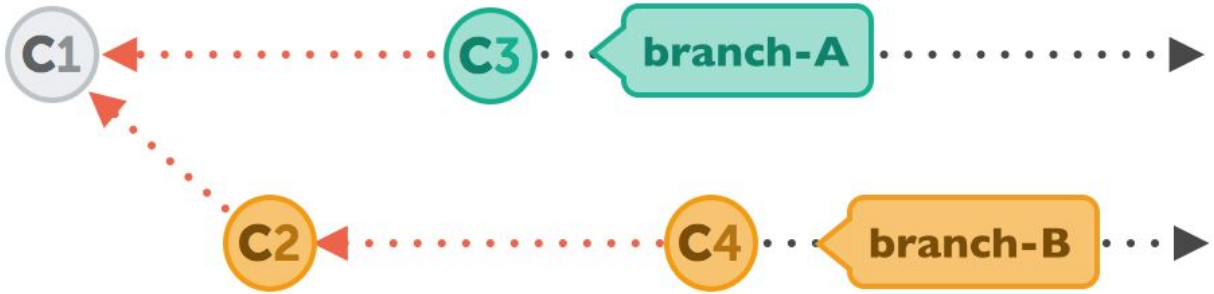


Rebase

- Let‘s walk through a rebase operation step by step.
- The scenario is the same as in the previous examples: we want to integrate the changes from branch-B into branch-A, but now by using rebase.

Rebase

- Same scenario as we did with Merge



Rebase Command

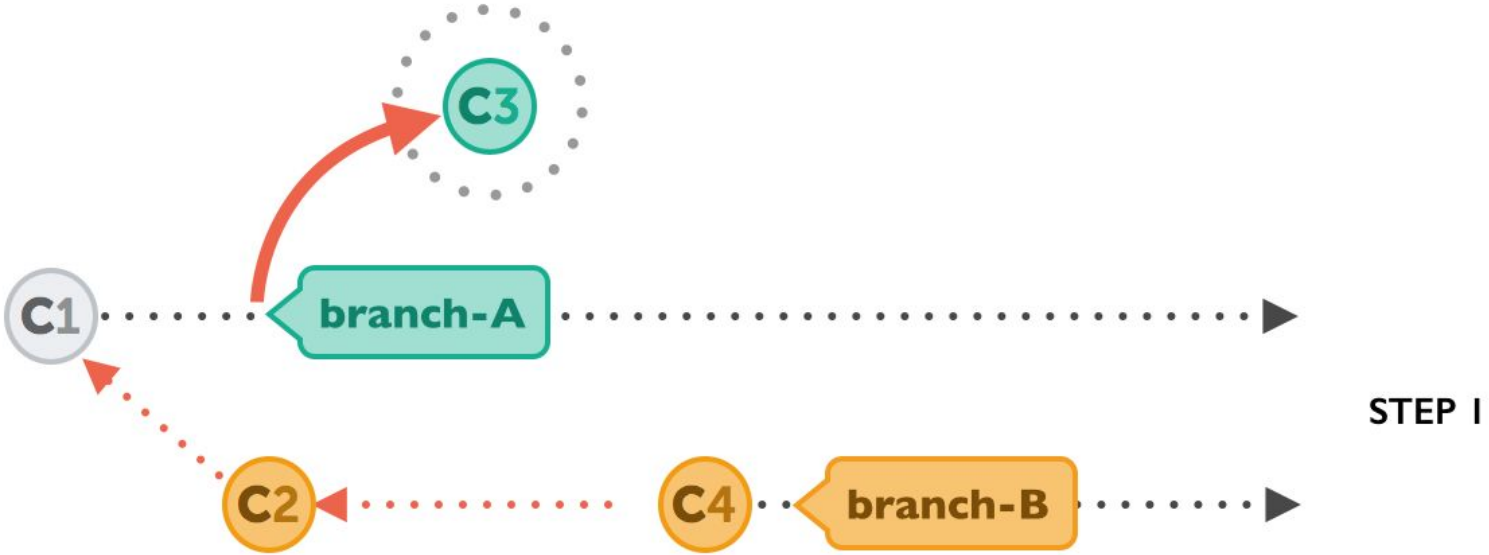
- git rebase <BranchName>
- git rebase branch-B

Rebase -- Step 1

- First, Git will “undo” all commits on branch-A that happened after the lines began to branch out (after the common ancestor commit).
- However, of course, it won’t discard them: instead you can think of those commits as being “saved away temporarily”.

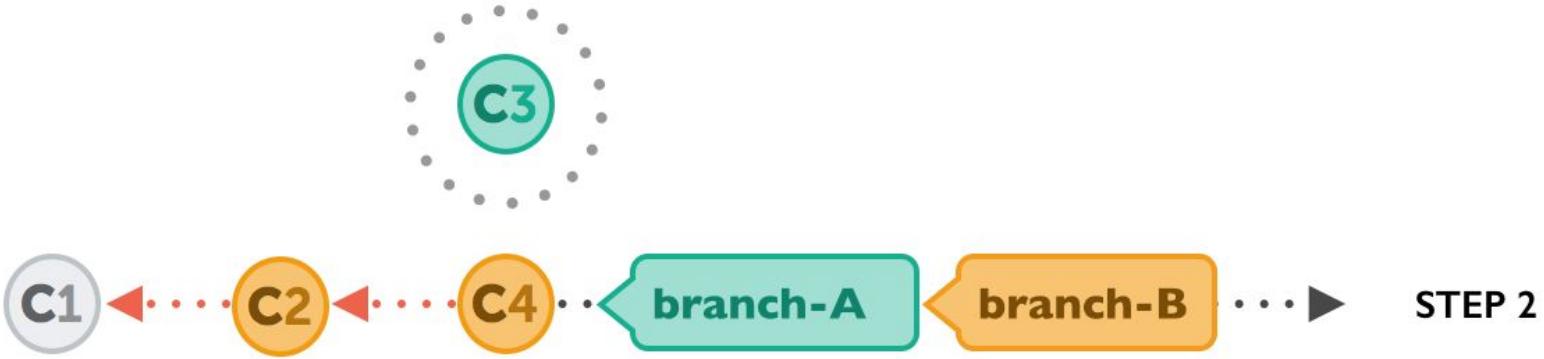
Rebase -- Step 1

- Undo all commits on branch-A after common ancestor



Rebase -- Step 2

- Next, it applies the commits from branch-B that we want to integrate. At this point, both branches look exactly the same.

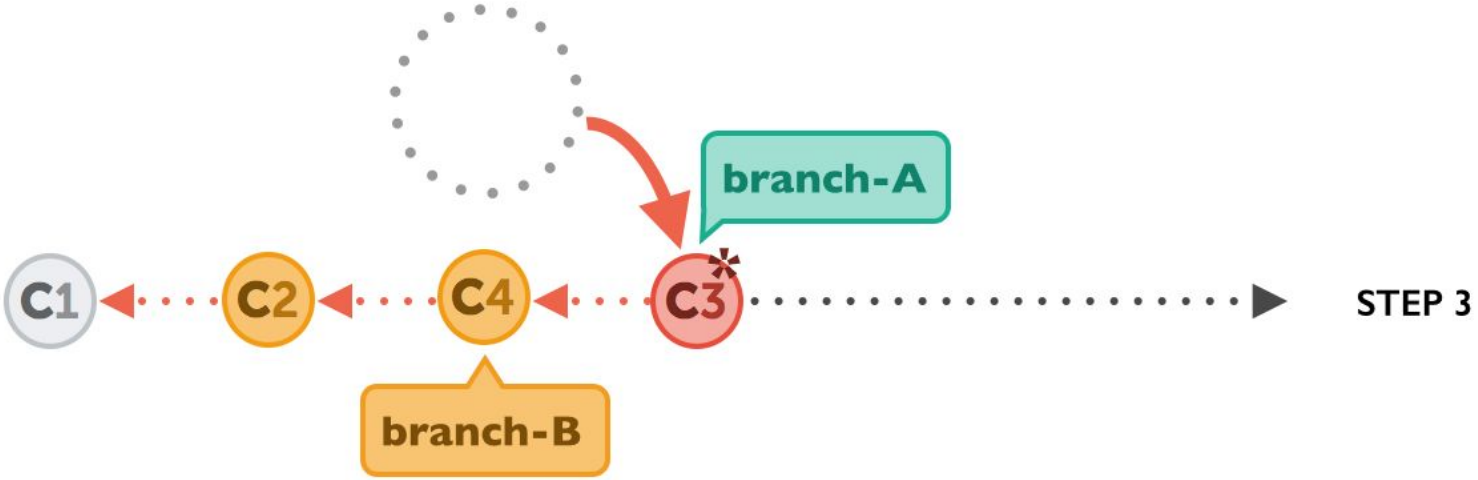


Rebase -- Step 3

- In the final step, the new commits on branch-A are now reapplied - but on a new position, on top of the integrated commits from branch-B (they are re-based).
- The result looks like development had happened in a straight line.
- Instead of a merge commit that contains all the combined changes, the original commit structure was preserved.
-

Rebase -- Step 3

- Applying Branch A commits in the end



The Pitfalls of Rebase

- Of course, using rebase isn‘t just sunshine and roses. You can easily shoot yourself in the foot if you don‘t mind an important fact: **rebase rewrites history**.
- As you might have noticed in the last diagram above, commit “C3*” has an asterisk symbol added.
- This is because, although it has the same contents as “C3”, it‘s effectively a different commit.

The Pitfalls of Rebase

- The reason for this is that it now has a new parent commit (C4, which it was rebased onto, compared to C1, when it was originally created).
- Rewriting history in such a way is unproblematic as long as it only affects commits that haven‘t been published, yet
- If it is published then, some other developer might have based his work on on original C3, this will make it more and more complex

The Pitfalls of Rebase

- Therefore, you should use rebase only for cleaning up your local work - but never to rebase commits that have already been published.

Demo