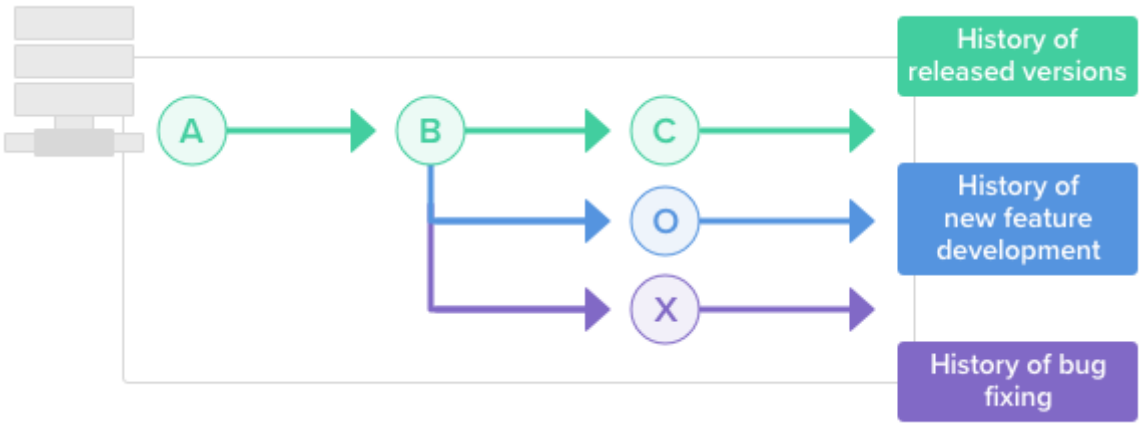
Using branches

In a collaborative environment, it is common for several developers to share and work on the same source code. While some developers will be fixing bugs, others will be implementing new features, etc. With so much going on, there needs to be a system in place for managing different versions of the same code base.

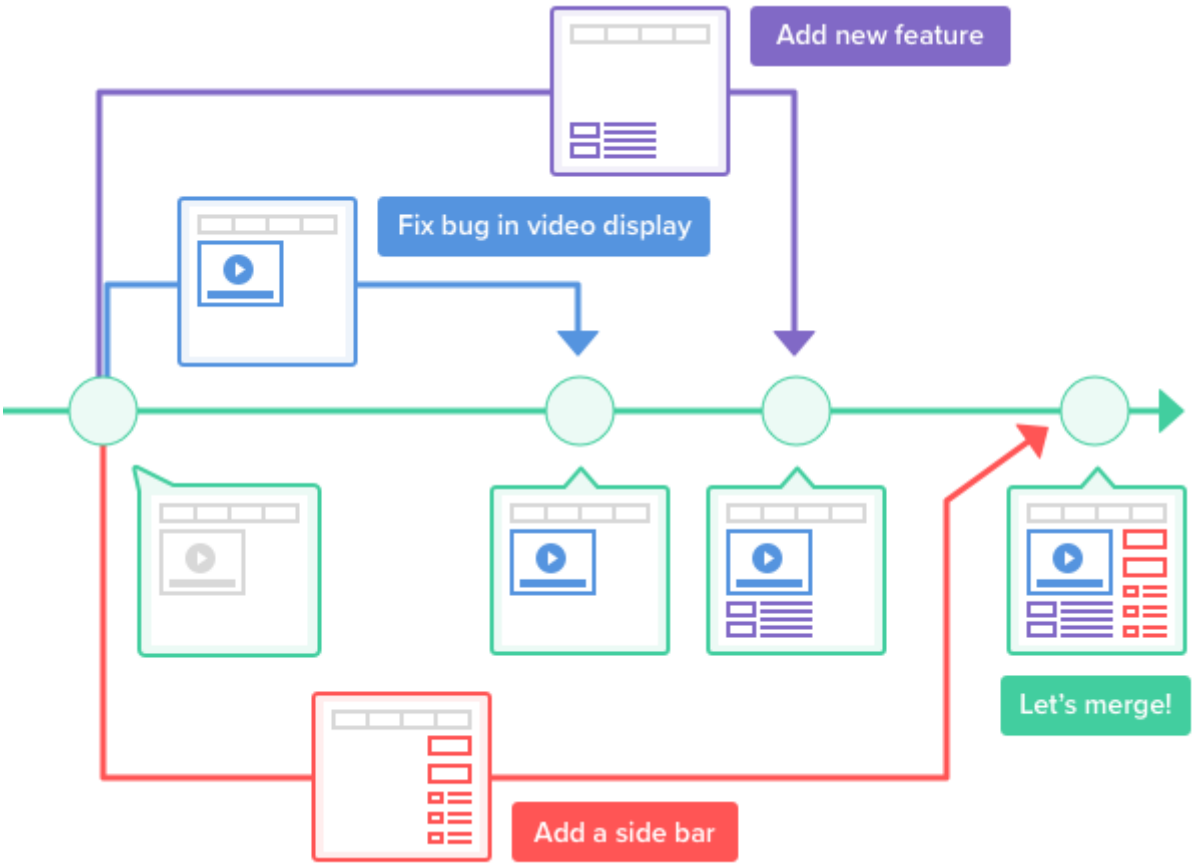
Branching allows each developer to branch out from the original code base and isolate their work from others. It also helps Git to easily merge versions later on.

What is a Git branch?

A Git branch is essentially an independent line of development. You can take advantage of branching when working on new features or bug fixes because it isolates your work from that of other team members.



The diagram below illustrates how development can take place in parallel using branches.



Multiple development projects taking place using the same source code.

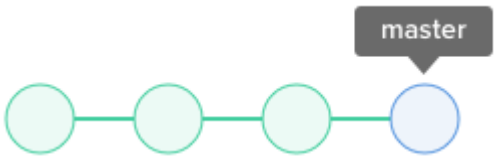
Branching enables you to isolate your work from others. Changes in the primary branch or other branches will not affect your branch, unless you decide to pull the latest changes from those branches.

It is a common practice to create a new branch for each task (i.e., a branch for bug fixing, a branch for new features, etc.). This method allows others to easily identify what changes to expect and also makes backtracking simple.

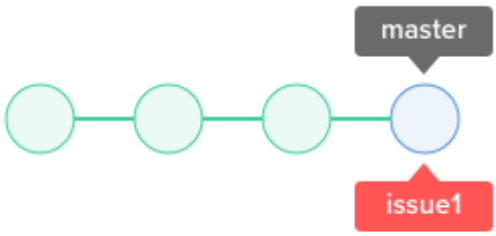
Create a branch

Creating a new branch does not change the repository; it simply points out the commit

The illustration below provides a visual on what happens when the branch is created. The repository is the same, but a new pointer is added to the current commit.



Before new branch is created



After new branch issue1 is created

Prev Next