Fontys - University of Applied Sciences - ICT

Git branching strategies

Instructable for the WKS

Ivan Georgiev Ovcharov
Student number: 4090993
Eindhoven, Thursday, May 6, 2021

Definitions, Acronyms and Abbreviations

Term	*Description*
GIT	Software used mainly by developers to collaborate and share their work.
GIT commit	A git commit is a command used to store the most recent changes on your project locally.
GIT branch	A branch in GIT is simply a lightweight movable pointer to one of the commits on your project
GIT repository	A git repository is a virtual “folder” where the user can store their committed work

Table of Contents

1 Introduction

- What is GIT?

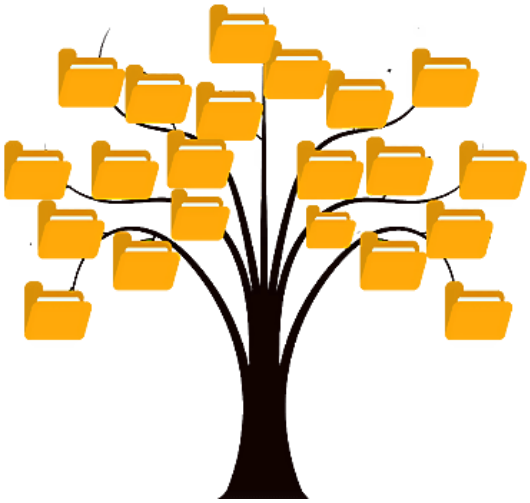
By far, the most widely used modern version control system in the world today is Git. GIT is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. Numerous projects and developers rely on GIT, as it allows them to work with a team, save and share their progress and have an overview of their previous work.

- Why is GIT so popular?

GIT is famous for its security and reliability, as it allows the user to closely manage the rights of users who want to view/commit changes onto an ongoing project. It is an extremely powerful tool, especially when used in a bigger scale, as within GIT there are commands that allow users to pinpoint different commits and previously submitted work.

Having said all of that, GIT repositories can become messy as the project grows in size and numbers of personnel. A solution for this is GIT branching, as it allows users to split their work and manage it more efficiently.

2. What are GIT branches?



A branch in GIT is similar to the branch of a tree. Analogically, a tree branch is attached to the central part of the tree called the trunk. While branches can generate and fall off, the trunk remains compact and is the only part by which we can say the tree is alive and standing. Similarly, a branch in GIT is a way to keep developing and coding a new feature or modification to the software and still not affecting the main part of the project. We can also say that branches create another line of development in the project. The primary or default branch in GIT is the master branch (similar to a trunk of the tree). As soon as the repository creates, so does the main branch (or the default branch).

Branches are generally named after the main focus on a certain section of a project (exmpl: we have a farm project, so a branch called “chicken coop” is created to manage that part of the farm).

GIT branching has inspired other companies to implement similar structures that allow the separation of files and different work and help with organizing.

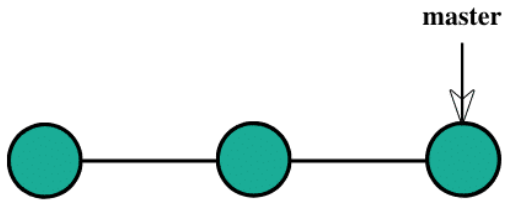
3. Why are branches so important?

GIT branches are essential for any project, no matter if it's on a large or small scale. Branches create another line of development that is entirely different or isolated from the main master branch (master branch refers to the default "main" branch upon creation of a GIT repository).

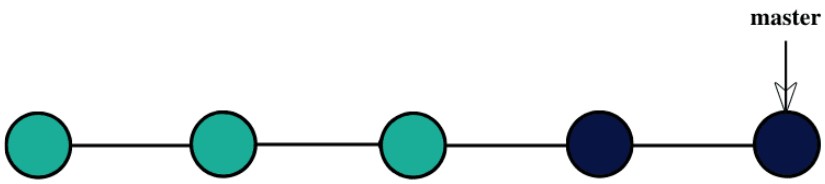
In the following example, two different types of management are presented (linear development and using branches):

➢ Linear development (and why it is inferior to GIT branches)

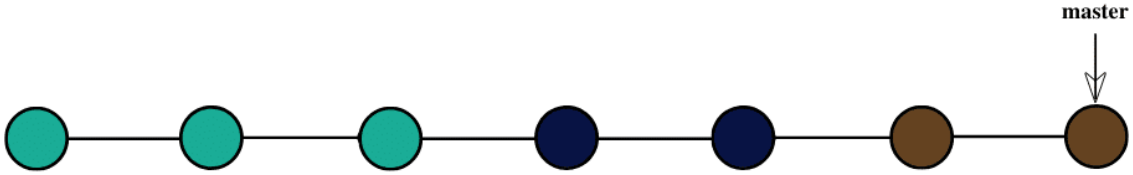
A group of developers has been working on a project for a client. So far, all of the implementations done by the team have covered the client's needs.



The team decides to develop a certain feature on the same code (marked by the blue colored dots).



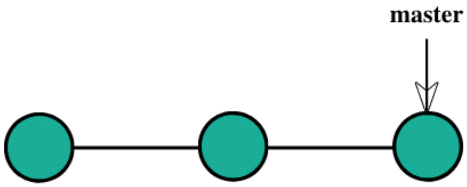
In the meantime, the group begins implementation of an additional feature (also covering the same code) and wait for the client's approval. (Brown dots note down the most recent feature).



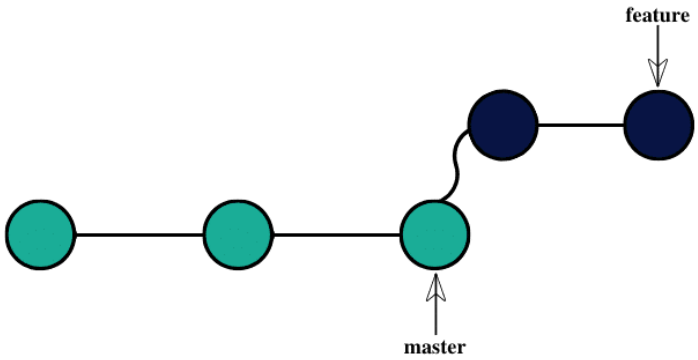
The client is satisfied with the newest features; however, he disapproves of the previous commits (blue dots). Since the team has been following the linear development, they now have to go back and delete everything, leaving lots of room for mistakes and errors in the code. This is where GIT branches come in handy.

➢ GIT branches

We have the same scenario: a team is working on a project (under a client's guidelines).

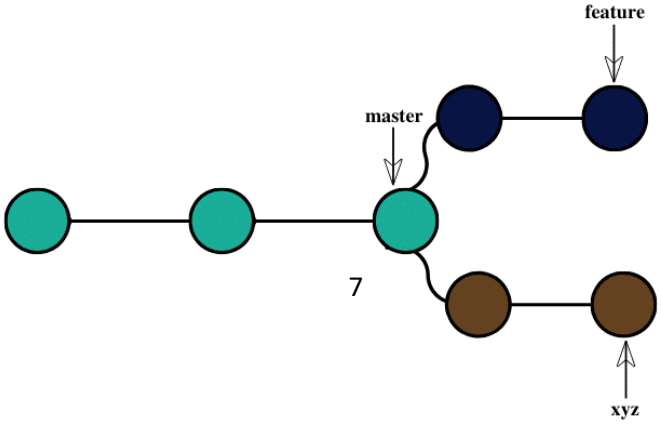


The team decides to develop a new feature and add it onto a new branch, where they can now separate their work.

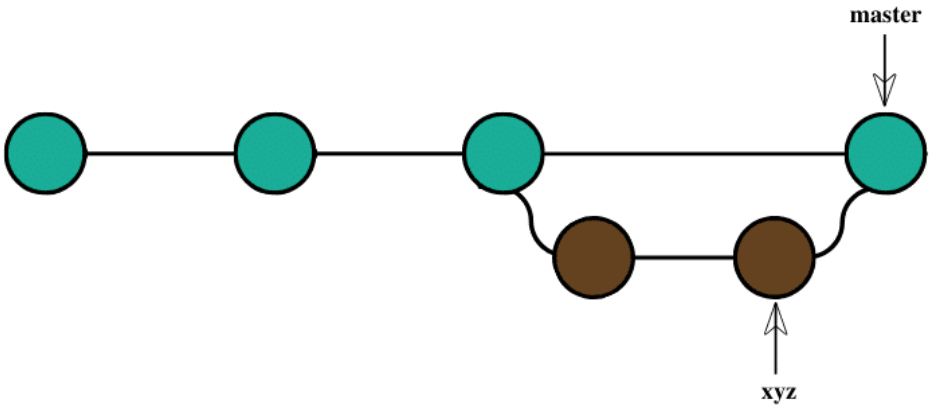


This is shown to the client.

In the meantime, the team begins working on a different implementation (brown dots) and wait for the client's approval (blue dots)



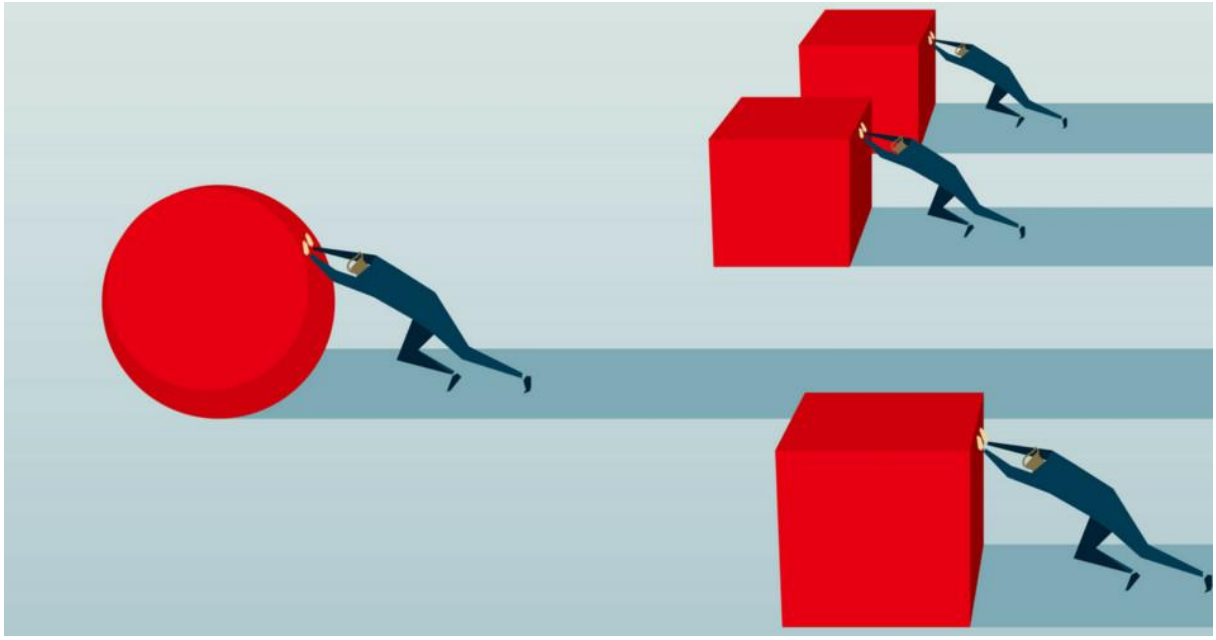
The client disapproves of the previous feature (blue dots), however, because of the convenience of GIT branches, the team simply has to remove that section, leaving almost no room for mistakes and saving a tremendous amount of time.



4. Working smarter, not harder

The idea behind GIT is to help any user manage their project, allowing them full control behind how the work is separated and the rights behind who can change/view it and follow their team's progress by checking the latest commits/statistics.

GIT branches are one of many additions to GIT that allow better management and allow a team to be versatile with their work. Whether it is a large-scaled company working for many clients or a smaller team, branching is always going to be at the heart of GIT. From the time of creating the repository until its deletion, everything involves branches. They can be moved, deleted or even presented, making them one of the most powerful features that GIT can offer.



References

1. https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell - GIT terms
2. https://www.atlassian.com/git/tutorials/what-is-git
3. https://www.toolsqa.com/git/branch-in-git/