What is version control? Why is it useful?

Version control is a tool that allows you to track progress, review and revert changes, and create backups in your codes. You can view the entire history of revisions to one or more files and merge revisions made by different people. If a mistake is made, developers can go back and compare earlier versions of the code to help fix the mistake. This also minimizes the possibility of losing all your files to some catastrophe.

Software developers usually work in teams, with one person working on one feature and another person working on a different feature. If these features need to be merged, version control helps by tracking a complete history to every change in every file by each contributor and helps discover and prevent conflicts.

What is a branch and why would you use one?

Another benefit of Version Control is branching. A branch is an independent line of development. Multiple streams of work can be kept separate from each and then can be merged back together at some time in the future. It is basically a brand new working directory, staging area, and project history.

By developing code in branches, you can work on both of them simultaneously, but also keep the master branch free of questionable code. If you want to add a feature or fix an issue, a new branch makes sure these changes do not affect the main code base and it gives you the opportunity to work out bugs before you actually commit the changes back to the main file.

What is a commit? What makes a good commit message?

When you commit a set of files and folders, you are making a save point for the project. A save point will never change unless you explicitly ask it to. The commit message describes this save point for future reference. A good commit message is detailed and states what was changed since the last commit.

What is a merge conflict?

A merge conflict occurs when you try to merge two branches that both have changes to the same part of the same file. If two people changed the same lines in that same file, or if one person decided to delete a section and the other person modified it, there will a conflict. Git won’t know which version to use and you have to resolve the conflict manually.

To resolve a merge conflict you should reach out to your teammate or colleague to figure out what happened. Once you figure out what should be the correct modification, you can use Git to resolve it. Git marks the conflicted area of the file with "<<<<<<< HEAD" and ">>>>>>> [other/branch/name]". The two conflicting changes are separated by “=======”.

Once you know what change should be kept, you can delete the wrong line and all the Git marks, making sure the file looks exactly how you want it. Then you can do a regular git add and commit the change with a new commit message.

[**phase-0-tracks**](https://github.com/jdubz1982/phase-0-tracks)/[version-control](https://github.com/jdubz1982/phase-0-tracks/tree/master/version-control)/**git-definitions.md**

#[Sea Life Centre](https://www2.visitsealife.com/)

### Paul the Octopus key stats:

\* \*\*Hatched: January 26, 2008\*\*

\* \*\*[Best Known for: Predicting results of 2010 World Cup Matches](http://www.ibtimes.com/octopus-made-better-world-cup-predictions-goldman-sachs-photos-1613882)\*\*

### Paul the Octopus current Halloween outfit:

\* \*\*[Cowboy Hat](http://www.allensboots.com/0475admy-adults-resistol-added-money-0475admy.html)\*\*

\* \*\*[Heart-Shaped Glasses](https://www.spencersonline.com/product/accessories/sunglasses/heart-and-flower-sunglasses/pc/2021/c/0/sc/2209/116719.uts?currentIndex=0&thumbnailIndex=18)\*\*