Git Repository

Github Repository:

<https://github.com/Boomstick64/Gitdemo>

Git merge commit:

<https://github.com/Boomstick64/Gitdemo/commit/0a60a0b7493fc2530ec239b7230c8538d91b9fa3>

Git Pull request:

<https://github.com/Boomstick64/Gitdemo/pull/2>

One of the biggest issues that plagues the programming profession is the fragility of projects created and worked on local machines. Projects interfacing with software components of computers are given unparallel access to its functionality, with that, comes a lack of security and stability. The ability to create and design with computer capabilities has been and always will be utilised endlessly, causing the need for a solution that includes preventative or corrective measures that at the very least allow for the salvaging of any completed work.

Programmers are often the ones at the end of the stick, creating large changes, providing new computer instructions, accessing and combing through the inner workings of the computer which has the high potential to touch something accidentally causing unforeseen problems to occur. Quite often, when these errors occur and there is a lack of safeguards in place to protect, isolate and research the problem, the issue leaves the local machine stuck in a specific state with the project, limiting the programmer’s ability to debug and solve the issue that has occurred.

For a solution to meet the requirements of stability and security of the project it must provide previously unavailable tools for programmers to use and access to help solve issues such as what was listed previously. A variety of capabilities may be provided to meet the requirements such as; the ability to track changes done, so that someone may locate and isolate where the problem area occurs, given a programmers intent, inform the programmer of potential mistakes in the changes they made compared to the changes they intended so that they may correct the errors, allow them to roll back changes after some sort of mistake occurs, or they were testing something and preferred it back the way it was, they are able to make multiple isolated development environments where they can create variations of the project with differences yet capable of merging changes together and most importantly, be able to store the project and backups to a cloud like service so that it is not reliant on hardware exclusively and local machines. Many of which, Git version control is capable of.

Git version control is often praised and used due it’s vast variety of capabilities and reliability. A list of its advantages goes as follows: Vast array of functioning, useful git clients, free to use, extremely high user control over projects, well suited software integration with most operating systems, easy compatibility with file types, highly documented in use and capabilities and safe and secure network for storing projects.

As with any service though, there are disadvantages to using Git version control. These often spout from git version control’s advantages having unfortunate side effects. These include; manually downloading and deleting files contained inside a git repository with the possibility of unintentionally deleting a file, creating changes on different machines but in the same file causing merge conflicts if a merge is attempted which may mean the deletion of one person’s work and without the use of a git client is not extremely user friendly.

With this all-in mind, it is extremely important to recognize the necessity for the use of git version control because despite it’s disadvantages it would be negligent to not use git in favour of nothing. In encourages unnecessary risk that can cost even the largest group of developers way too much.