For a long time now, GitHub/Gitlab/Bitbucket have been a very attractive option for software hosting and version control for developers to discover, share and build software. Individuals and companies are streamlining DevOps and CI/CD (**Continuous Integration and Continuous Delivery**) workflows, hosting and reviewing code, increasing efficiency for project management and so much more. Bitbucket and GitHub offer a free plan for private repositories but GitHub is increasingly becoming more attractive than Bitbucket because it offers a free plan for unlimited private repositories and unlimited collaborators compared to Bitbucket. GitHub offers issue tracking, project management boards, group milestone tracking, team discussions and organisation management in the free plan all in all. Gitlab is another alternative to GitHub and Bitbucket, which claims to offer a complete DevOps platform, delivered as a single application.

With developers opting to use GitHub more than Bitbucket because of the payment plan, many are resulting to migration of their private repositories from Bitbucket to GitHub. GitHub is only a code repository using a Git System. Git does the version control. Git system has a feature known as **Git Bash** which provides **BASH** emulation for running Git commands from a windows command line. Git Bash makes LINUX users feel comfortable when using windows. Once you install Git, it also automatically installs Git Bash and shell integration, which ensures that when you right-click in any folder, you will be able to open Git command line.

GitHub enables for the cloning of an existing repository in a new directory at another location. The command “***git repo”*** is used for the cloning process. When cloning a remote repository, a remote connection called ***“origin”*** is automatically created to connect back to the cloned repository. “***git remote***” command is used to access the remote repository.

GitHub also enables the tracking of more project files as the project being worked on keeps on growing. It has ***git add, git commit, git push*** and ***git pull*** features for adding, committing changes made in the project, pushing the changes to the online repo and also updating your local copy by pulling changes pushed by other team members respectively.

The Git system uses git branches. Branches are independent development routes, which can later be connected back to the main branch. They are used when adding a small or big feature; when testing some logic and do not want bad code to appear in the master branch; when separating releases and when fixing bugs or assigned issues. Master branch is created when a repository is created to represent the official project history. The command ***“git branch***” is used to check available git branches in the repository. Branches are merged once changes are made in any one of them so that other branches can contain changes made in one specific branch.

When working on big projects, which takes a long time and involve many developers, it is advisable to define a strict branching model or design centred around project release. Master branch holds the latest version of fully working code, which is the current code in production. The only merges to master branch, comes from release and hotfixes branches. Develop branch is created from the master branch and it complements the default master branch; it is the integration branch for features. Feature branch is created off the develop branch and once the feature branch is done, you merge it back to the develop branch. Release branch is created off the develop branch once all the features a completed and are ready for release. Because of releases, a suitable environment needs to be created for testing purposes. If any errors or bugs occur, they are taken care of and merged back to the develop branch. After the code is cleared as correct and working, it is merged back to develop branch and the develop branch is merged back to master branch with a release tag. The purpose of tagging is usually to take note of specific points in the repository history as being important.

Hotfixes branch is created off the master branch in case any errors/bugs are detected on the master branch. Once the errors are fixed then it is merged to both develop and master branches. Bugfix/Issue branch are the small branches used to fix errors detected in a release branch while being tested.