Alexander Mortillite

Git Workflow Essay

Learning how to use git is essential for any programmer, not only is it extremely useful for version control, but it also allows for you to create backups of your projects to websites like github. Development teams that don’t use version control solutions will likely run into the same common errors. From problems keeping up to date with the “main” branch of code, to increased time in troubleshooting bugs that might(will) occur when different parts of the team try to merge their code with one another. There are multiple ways to configure version control, from command line solutions, built in integration with IDEs such as Eclipse or Intellij to applications built solely for pushing and pulling files to specific repositories such as Github desktop.

The most important topic you need to understand when learning about git workflow is what a repository is. It is the key that holds everything together; a Repository or Repo is a central location which data is stored to. One of the bright sides of version control is that each change can be easily reversed, hence the name “version control”. Another feature that repos have are branches, this system functions as a way to create off-shoots of your main source code, allowing you to have beta branches for future updates or a development branch if you wanted to.

Now that I have explained some aspects of version control and some of its capabilities, I can get into how the magic behind the scenes work. The best way to learn how to use git is to see each of the moving parts in action. The first of the parts is the local repository, it functions exactly how you would expect, it is a directory within your machine that is used to save your work locally. This is vital for users for several reasons, first of which is that not everything that you work on needs to be or should be pushed to the main project. Secondly the local repo allows each file to be edited individually by each person on the team, without getting in the way of each other. Step two in your code's journey is the commit feature, this functions as a staging area within your storage device. This section allows you to choose files and create packages that can be uploaded to a git repository such as github or a differing repository solution. This staging area allows you to also upload messages along with the packages that you make so that others can quickly interpret the purpose of the code you are about to push. Step three is called pushing, this is the step where the commits that you made in step two will be taken from your storage and “pushed” to the central branch. This process doesn’t modify your code within your local repository, it just copies and uploads it. The last step, although technically required for utilizing git is strongly recommended; is called pulling. Pulling allows you to easily update your local repository with a version of the code on the central branch. This will keep all aspects of your project updated ensuring that you are operating on the most recent version of your codebase. To prevent lossages in code, it is important to make sure that you commit and push code that you have worked on to the central repository before you make a pull request. If you do this process in the wrong order then you can run into situations where you overwrite code that you have been working on with no way of retrieving it once it has been overwritten assuming that you have never committed it before.

Getting started with version control is generally the hardest thing about it, preferably you first learn the process of what git workflow is and then you implement it into your personal programming workflow. This isn’t always the case though, some people will unfortunately implement git without understanding the whole process and will either form a litany of bad habits or worse, losing code which defeats the purpose of its initial implementation.

References

<https://www.atlassian.com/git/tutorials/what-is-version-control>