Tabari Harvey, Module-3.2 Assignment, csd-380

Version Control Guidelines

Version control allows a lot of people to work on a project all at the same time. Each different person who is working on the project will, work on their portion of it. Being unique to them and they can share their portion of the project work. Allowing the edit of one or multiple people to not halt any other persons work.

The 8 best version control practices from Brent Schiestl on Perforce.com. Commit changes automatically which is all files that are in a commit become committed together or not. Committing files with a single purpose which means if you add too much the changes can become difficult to overlook. Writing good commit messages, and that means each commit needs to inform what it is with a good description. Don’t break builds, if the build is not a complete commit it could break for a team members work.

Reviewing commits before sharing can make sure code improves or gets good fixes in general. Every commit must be traceable so that it becomes easier to track down any code changes, all so that the code does not receive complications.

Following branch best practices helps when a branch experiences some changes. In order for it to follow into another branch the best practice is to be default and follow the following practices: Keep things simple, well-defined code branching policies, code lines need an owner, use branches for releases, protect your mainline, and merge down and copy up. Protect your assets which means that it would be a good practice to have a security plan to protect your assets. (Schiestl, 2020)

From the GitLab topics on Git version control best practices. Starts with the importance of making small changes. In order to try something new it’s best to break it down into small increments to quickly test the code.

Keeping commits atomic are in relation to making small changes like stated above. Atomic commits are work that is just one task at a time. When all of those commits are changed, they are merged together after.

Develop using branches is when teams use make commits atomic, by not affecting the main code line. With everything tracked in the branch the code is merged once it is ready.

When making commits there must be a good description of the commits. This is to ensure workers understand what the purpose of each commit is.

Obtain feedback through code reviews is a good way for the team to communicate and work effectively. This is to make sure the code runs smoothly and that everyone is on the same page.

Identify a branching strategy is important because the teams that work on these projects are very full of experience but those experiences are different. That diversity can meld together if the team follows a singular branch strategy. (What are git version control best practices?)

For these good practices from Michael Ernst, he gives advice on two version controls in centralized and decentralized. “The main difference between centralized and distributed version control is the number of repositories. In centralized version control, there is just one repository, and in distributed version control, there are multiple repositories.” (Ernst, 2024)

Starting with the centralized version control practices it is important to have a good description for commit messages. If that does not happen then miscommunication will take place, and the code might not flow well.

Each commit on the main branch needs to have a single objective. The purpose of this is to make small quick changes and to eventually merge all the small commits.

When you are making commits it can be easy to make more changes to the commits than originally planned. There are commands that one should follow before each commit is made. When working on these projects it is best to work with the most up-to-date version.

When you make changes, it is best to share the changes made. Coordinate with co-workers to have good cohesion to have a project run smoothly. It is important to remember that version control tools record changes on each line of code.

Don’t commit generated files because, version control is supposed to be files that are edited by people. It is best to understand the merge tools and always have a copy of the project that is being worked on.

For distributed version control practices, it starts with understanding the typical work flow. It would be best to not rewrite the history due to time conflicts and, those merges can create confusion on the project. Finally, when making changes, you have to commit and merge before you try and update the file. (Ernst, 2024)

Compare and contrast the guidelines among the three, did you find any guidelines that are not relevant today? The similarities between these different guidelines starts with that all of them agree that commits should be atomic, and have a single purpose. Commit messages should be clear and descriptive. All three sources emphasize the importance of having good branch practices. Finally, it is important to review commits and feedback before it is shared.

Perforce is focused on protection and security of the assets, also being traceable. It also adds that it’s super important to understand the rules of branch management. GitLab is all on team collaboration through atomic commits, the way branches are used, descriptions on commits, feedback from users on the project, and everyone having a branch strategy. Michael Ernst talks about centralized and decentralized systems. Centralized version control is on not committing to generated files, the commits need to be small, and it’s important to work with modern versions. The distributed/decentralized version control says to not rewrite history and the importance of understanding workflows.

Out of all of these guidelines I believe these are the most important. Writing good commit messages is important because, each commit is informative to team members. Breaking down the work into small increments will help the code get tested quickly and effectively. All so this code is merged into one to complete the whole project. Atomic changes follow the small increments style but that means every task is done one at a time. It is important to look through feedback and code reviews. It helps the less experienced understand the code and, keeps the workflow steady with important information. When making commits it is important to follow specific commands especially when they are up-to-date. It keeps the commits readable and flowing nicely. It is important to understand typical workflow. If you don’t know the workflow it could be consequential for the experienced and less-experienced.

I feel that these are the most important because it puts an emphasis on being organized and focused. This also shows the importance of that organization leading to good cooperation with the rest of the team. I believe the most important guidelines are understanding commit commands, otherwise if you are just making commits with no command understanding there is a lot of unnecessary confusion that will take place, and following the workflow otherwise, the lack of organization and focus does not equate a good project at completion.

Works Cited

Schiestl, B. (2020a, May 20). 8 version control best practices. Perforce Software. <https://www.perforce.com/blog/vcs/8-version-control-best-practices>

What are git version control best practices? about.gitlab.com. (n.d.). <https://about.gitlab.com/topics/version-control/version-control-best-practices/>

Ernst, M. (2024, April 8). Introduction to Version Control. Version control concepts and best practices. <https://homes.cs.washington.edu/~mernst/advice/version-control.html#Introduction_to_version_control>