Version Control Guidelines

Austin Moore

Bellevue

CSD380 - DevOps

Mr. Issa

August 31, 2024

Version Control Guidelines

I’m reviewing three sources that have version control guidelines. These sources include Atlassian, GitHub, and Google. The first source is Atlassian. They have created popular tools like Bitbucket and Jira and have some extensive guidelines when it comes to control. The first guideline is to use a branching strategy; Atlassian ensures there are well-defined branching models, Gitflow as an example, to be able to manage different features, fixes, or releases. The second guideline is to commit frequently, but with purpose; this guideline ensures developers are committing code changes often to avoid losing work, but that the commits are logical and meaningful. The third guideline is to write clear commit messages; this guideline ensures there is clear and descriptive commit message that helps track the history and reasoning around the commitment. The fourth and final guideline is avoid committing generated files; this ensures that files that can be generated by the build process should not be committed to the repository.

The second source is GitHub. GitHub is a major platform when it comes to version control, and has an emphasis on collaboration and clarity. The first guideline is using pull requests for code review; Git recommends using pull requests before merging code on the main branch. This ensures any potential issues are caught and that you are collaborating well before the commit. The second guideline is to follow a consistent naming convention; this ensures branches and commit messages have a standardized naming scheme which will allow for a better understanding when reviewing prior commits. The third guideline is protect the main branch; this ensures that the main branch is protected by restricting direct commits and requiring an extra review before the commit happens. The fourth and final guideline is document your process; Git encourages documentation in their version control practices, which helps make it easier for new people to get up to speed quickly when using this tool.

The third source is Google. Google has a variety of software they utilize internally, and their focus is around scalability and efficiency. The first guideline is use a single repository for related projects; Google believes that having a large repository with multiple projects will help manage projects better and make them easy to share. The second guideline is automate code quality checks; this includes having a tool that automatically enforces coding standards, helps run tests, and checks the code for vulnerabilities all before it’s merged. The third guideline is prioritize readability; this ensures there is an emphasis on writing code that is easy to maintain and ensures commit messages and coding tyles are consistent. The fourth and final guideline is minimize long-lived branches; any long-lived branch can lead to potential integration issues and should be avoided by merging changes frequently.

When comparing these, all three sources are aligned on the importance of commit messages and branching strategies, including the emphasis on the quality of these messages. Each source does slightly differ based on what they are looking to do; Google is focused on large scalability and multiple project management in their rules, whereas Git is focused on pull requests and branch protection, and Atlassian is primarily focused on more traditional branching strategies overall. Most of these guidelines are relevant even today, especially since the sources where this data was collected from is extremely popular and used frequently with today’s technology and software. However, some of the rules may need adjusted depending on the particular situation you are looking to solve. For example, Google's focus on large project management may not be ideal for a smaller company or team looking to manage only a handful of projects.

After researching different guidelines, here is my personal list of guidelines I’d have mapped out. The first guideline would be commit frequently with purpose; this will ensure you are not over-committing and creating lots of additional messaging for tracking, but ensures the commits you do make are substantial and show a real change. The second one would be write clear messages; this will help make sure commit messages are easily understood and clearly show what that change included. The third guideline is use pull requests for code review; this will help ensure the latest code is up to date, but also gives availability for a team to come together and collaborate on that code. The fourth guideline would be follow a branching strategy; a well-defined branching strategy is important because it shows consistency, is easy to follow, and helps with bug fixes and releases being managed efficiently. The fifth and final guideline would be automate code quality checks; this will help maintain quality code by catching any issues early on through this process.

References

Atlassian. (2024). *Version control software: An overview*. Bitbucket. Retrieved August 31, 2024, from <https://bitbucket.org/product/version-control-software>

Google. (2024). *Best practices for version control*. Google Cloud. Retrieved August 31, 2024, from <https://cloud.google.com/docs/terraform/best-practices/version-control>

Guthrie, G. (2023, August 18). *6 best practices for git version control*. Nulab. Retrieved August 31, 2024, from <https://nulab.com/learn/software-development/version-control-best-practices/>

Quora. (2024). *What version control system does google use, and why?* Retrieved August 31, 2024, from <https://www.quora.com/What-version-control-system-does-Google-use-and-why#:~:text=As%20other%20answers%20have%20noted,of%20the%20development%20toolchain%20begin.>