**Version Control Guidelines**

Jackie Scott Module 3.2 8/26/24

Version Control or source control is a practice utilized to track and manage changes to software code. There are many version control software tools that help with tracking and changes. These tools help software developer teams work more efficiently and smarter. DevOps teams utilize Version control software heavily since it helps reduce development time and ensures more successful deployments. This software tracks any modifications made to scripts. This makes rolling back to a working version easier if a mistake is made. Regardless of which version control is utilized or no matter how big or small a team may be, best practices should be used.

The article “8 Version Control Best Practices,” written by Brent Schiestl, explains those 8 best practices and why they should be used.

1. **Commit Changes Atomically:** Files should be committed altogether, not in pieces.
2. **Commit files with a single purpose – not as a backup:** Every commit should have a single purpose, even if it is a bug fix or adding a new feature. If a single change creates multiple independent changes, it can become difficult to read and review.
3. **Write good commit messages:** Every commit should clearly explain why, not necessarily how. This makes for easier review.
4. **Don’t break builds:** Avoid breaking builds by doing complete commits. Ensure every commit is usable by others without something breaking.
5. **Do reviews before committing to a shared repository:** Before committing to a shared repository, also review and review often. This can be done through a review system or a pull request.
6. **Make sure every commit is traceable:** A project should be able to build and pass test cases before and after a commit. Store the author of a change for security and auditing purposes. Also store any additional information like reviewer comments or if a commit is associated with a specific issue or request.
7. **Follow branch best practices:** this includes such as

* Keeping things simple
* Have well-defined code policies
* Giving code lines an owner
* Use branches for releases or milestones
* Protect the mainline
* Merge down and copy up

1. **Protect your assets:** Do this by incorporating the right security measures as your version control system stores valuable assets such as

* intellectual property
* Product designs
* Videos, graphics, and images
* Business documents

Adekola Olawale, a writer for freeCodeCamp, published an article entitled “Git Best Practices – A Guide to Version Control for Beginners” that focuses on many things, from how to use Git to best practices. Olawale states that for users “to get the most out of Git, it’s important to follow best practices when working with the tool.” The practices noted in the article are

1. **Keep commits small and focused:** Focus on specific changes or features. This, in return, will make it easier to understand what was done in each commit while also reducing the risk of any conflicts.
2. **Use clear and concise commit messages:** Messages should clearly describe what changes were made and why. A good message should give enough information to understand the context of a change.
3. **Branch frequently to isolate changes:** This makes it easier to manage changes and collaborate with other developers. This can also help reduce any risk of conflicts and errors.
4. **Use pull requests for code reviews:** These requests make tracking changes easier and ensuring that script changes are properly tested and documented before any merge occurs. PRs for code review enable collaboration and quality control.
5. **Keep the repository clean and up to date:** This is crucial for maintaining the health of a codebase. Avoid committing any unnecessary files like temporary files or build artifacts.

The third source I found is an article written by Ben Holt entitled "Source Code Management: Best Practices for Version Control,” which lists four best practices for version control.

1. **Create a branching policy:** This policy should be well-defined, keeping it simple to maximize team adoption and daily use.
2. **Ensure traceability:** The easiest way to ensure traceability is to use a version control tool and application lifecycle management tool. These applications make tracing each patch or code change to a specific request easier.
3. **Manage commits properly:** Keep commits small, only one at a time, and focus on a single change at a time. This will make rollbacking much easier in the case of a failure.
4. **Write thorough commit messages:** Explaining the intentions when committing to a repository will give the entire history and traceability.

While all three articles had slightly different best practices regarding version control, with one article focusing only on Git, all the sources had the same principles. From commit practices, commit messages, branching, and isolation of changes to traceability and review. In the article “8 Version Control Best Practices,” written by Brent Schiestl, I did find it very interesting that one of the best practices included was protecting your assets by incorporating the correct security measures. This point was one no other article mentioned. In the end, Schiestl is correct. An organization’s version control system stores so much information that it could essentially take down an organization if not correctly protected. Another difference, maybe due to the focus on using Git, was Olawale’s practice of maintaining repositories clean and free of any unnecessary files, such as temporary files.

One guideline that might be less relevant today is Schiestl’s recommendation to avoid breaking builds. With modern CI/CD pipelines, automated testing often detects issues before code is merged into the main branch, reducing the likelihood of broken builds reaching production. Similarly, Holt’s suggestion to implement complex branching policies may also be outdated. Agile methodologies prioritize simplicity and speed and are widely adopted in organizations today, making overly complex branching models unnecessary.

The guidelines I have learned to adapt in both my professional career and now in school are

1. **Commit Small and Often:** I commit every assignment separately so that I know exactly where the issue may lie if there is an error.
2. **Clear Commit Messages:** Adding a clear and concise message when committing on exactly what module and assignment number is added makes for easier future understanding in case any issues occur.
3. **Repository Cleanliness:** This is just who I am; I hate clutter and am insanely organized. Naturally, this translates to my repositories by making sure I only commit necessary documents, nothing extra.

I follow these simple guidelines because they help me maintain and uphold a high-quality standard in my repositories. I also feel that these guidelines keep me and train me to think in the DevOps practices.

Works Cited:

Adekola.olawale. (2023, May 16). *Git best practices – A guide to version control for Beginners*. freeCodeCamp.org. https://www.freecodecamp.org/news/how-to-use-git-best-practices-for-beginners/#best-practices-for-using-git

Atlassian. (n.d.). *What is version control: Atlassian Git Tutorial*. https://www.atlassian.com/git/tutorials/what-is-version-control

Holt, B. (2023, October 11). *Source code management: Best practices for version control*. Sterling PLM. https://sterlingplm.com/articles/source-code-management-best-practices-for-version-control/

Hovhannisyan, A. (2022, June 8). *Make Atomic Git commits: Aleksandr Hovhannisyan*. RSS. https://www.aleksandrhovhannisyan.com/blog/atomic-git-commits/

Schiestl, B. (n.d.). *8 version control best practices*. Perforce Software. https://www.perforce.com/blog/vcs/8-version-control-best-practices