Version Control Guidelines

What is Version Control

Version Control is the practice of tracking and managing changes to software code. This helps teams track changes throughout the development process. Employing the right tools and guidelines will help the DevOps teams reduce workload and increase efficiency throughout a project. Across the development landscape, there are many different guidelines. It provides various benefits such as a long-term change history on each file, branching and merging, and traceability. In this paper, we will compare different guidelines that have been released.

gitlab.com lists 6 different standards.

1. Make incremental, small changes – Here you want to use the smallest amount of code possible. Once the issue or solution is found, break the updates into small batches that can be quickly tested by the end user. This decreases the likelihood of integration conflicts and allows the team to revert easily if merge conflicts happen.
2. Keep commits atomic – This again is keeping your work changes small and fixable in one or two tasks. Only commit when needed to provide a better context for the team
3. Develop using branches – Developing without affecting the main code line keeps your code clean from bugs and vulnerabilities so it is properly tested before it hits the main production branch
4. Write descriptive commit messages – Start with a verb in present tense to indicate the purpose of a commit. Each commit should only have one purpose.
5. Obtain feedback through code reviews – This ensures code quality and allows a different perspective to see elevate your code
6. Identify a branching strategy – A single branching strategy will decrease the chances of merging conflicts.

LinkedIn has an article specific to branching and merging with 7 different guidelines

1. Choose a branching strategy
2. Follow naming conventions – Keeps the branches organized and easy to identify
3. Keep branches short-lived
4. Merge frequently and carefully
5. Resolve conflicts promptly
6. Document and communicate your changes
7. Consider personal experiences

Finally, the university of Washington has an article specific to version control

1. Use a descriptive commit message
2. Make each commit a logical unit
3. Avoid indiscriminate commits – allows users to verify each change per file
4. Incorporate others’ changes frequently
5. Share your changes frequently
6. Coordinate with your co-workers
7. Remember that the tools are line-based
8. Don’t commit generated files – reduces file bloat. Users who want the generated file can do so using the program
9. Understand your merge tool
10. Obtaining your copy

Overall, all three articles do a great job at identifying guidelines for version control. If I had to rank them, I would put the Washington article at 1, GitLab at 2, and LinkedIn at 3. The Washington article goes in pretty good detail about the dos and don’ts of version control. The GitLab article is less detailed, but more concise for general purpose use. The LinkedIn article is also more for general purpose, but talks more about the risks if the standards aren’t followed.

Here is my top 5 based on all three articles

1. Choose a branching strategy – Having a plan in place will keep all developers on the same page when merging branches.
2. Document and communicate Changes – Communication is key in all aspects of life. There is no difference between commits and changes.
3. Make each commit a logical unit – Breaking the commits down to the smallest level is a great way to keep updates smooth and comfortable
4. Understand your merge tool – If the team understands the merge tool, it can increase efficiency for all involved. Also helps with training
5. Use descriptive commit messages – This will help others know what the commits are for. That way the user doesn’t have to waste time figuring out what the commit is for.

References

<https://about.gitlab.com/topics/version-control/version-control-best-practices/>

https://www.linkedin.com/advice/0/what-best-practices-branching-merging-version

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