Greih Murray

Assignment 3.2

August 27, 2024

The sources chosen for this paper were GitLab, RadixWeb, and PEGA Documentation, each of which has its own list of guidelines or best practices as relating to version control within the DevOps space. GitLab recommends making small, incremental changes, keeping commits atomic, developing using branches, writing descriptive commit messages, obtaining feedback through code reviews, and identifying a branching strategy. Most of these mean exactly what they say, though for the sake of clarity, to keep commits atomic means to keep commits to a single unit of work, typically a single task or fix, and not to bundle multiple tasks into one commit.

RadixWeb’s list of guidelines is as follows: write clear and concise commit messages, address and resolve conflicts promptly, use pull or merge requests in a structured way to review and discuss changes, keep feature branches up to date, create tags or labels to mark important milestones, versions, or releases, and consider using dedicated tools for binary assets. As you can see, while some of these are the same as Gitlab’s the majority of the included guidelines are unique namely keeping feature branches up to date, creating tags, resoling conflicts promptly, and considering dedicated tools for binary file storage.

PEGA Documentation also has their own list which includes working in branches, considering major versions when the server or database updates to a new major version, incrementing the patch and minor version during each merge, merging into the next patch version, assigning specific roles to specific users, and maintaining an application record capped at major and minor versions. As can be seen, this list seems to focus more on versioning than version control, though it does still share a similarity with both of the previous lists, that being the recommendation to work in branches.

Personally, I believe that the most important guidelines for version control would be to work in branches, keep feature branches up to date, keep changes atomic, have good commit messages, have meaningful code reviews and act on the findings/discussions, and assigning specific roles to specific users. I believe that failing to work in branches, keeping those branches up to date, or failing to keep commits atomic, all have the potential to reduce the benefits from version control, as work would simply get messy and hard to track. The commit messages I chose as they can help quickly identify what a commit did and why it was included, code reviews allow for a fresh set of eyes to inspect the changes and possibly catch any mistakes that were made, and assigning specific roles could prevent issues where users are attempting to manipulate aspects they have no reason to interact with. For example, a front end developer does not need the same access as a DevOps specialist, and could potentially cause chaos if granted such access.