Paul Romer

CSD380

Module 3

Version Control Guidelines

Version control systems are essential tools for developers. They let you keep track of code changes, work with others, and avoid losing work. Git is the most common version control system today. It's used in personal projects and huge company codebases.

Microsoft's guide is a beginner friendly intro to version control. It explains how version control fits into DevOps and teamwork, and why it's so important for collaboration. Atlassian focuses on how teams should work with Git, like using feature branches, pull requests, and workflows. The Pro Git book goes deep into the details of Git: how it works, how to commit properly, and how to fix mistakes. The DevOps Handbook talks about how version control helps with fast and safe software delivery.

They all agree on:

- **Making small, focused commits**: Each change should be one clear thing. Don't mix multiple changes in one commit.
- **Writing clear commit messages**: Use a short summary line, and sometimes a longer explanation. Say what changed and why.
- **Using branches**: Work on features or fixes in their own branches. Merge them back when they're done and tested.
- **Reviewing code before merging**: Use pull requests or code reviews to catch bugs and improve quality.
- **Automating testing**: Set up continuous integration to run tests on every commit or merge.

What Feels Outdated?

Some older branching strategies like Git Flow, which involves lots of long-lived branches which can lead to complicated merges and can slow things down. Current advice is to use shorter branches and merge often into the main branch. Also, tagging every release by hand isn't always needed anymore because CI/CD tools can do that automatically.

My Version Control Rules

Based on everything I read and learned so far, these are the version control practices I think are most useful:

1. **Commit often:**
   Don’t wait too long to commit and make sure each commit makes sense on its own.
2. **Explain your changes:**
   Good commit messages help everyone understand the history of a project.
3. **Use short-lived branches:**
   Keep branches focused and merge them regularly. This avoids messy conflicts.
4. **Always review and test code:**
   Make sure someone else looks at your code, and that tests pass before merging into the main branch.
5. **Automate everything you can:**
   Use CI tools to check your code, run tests, and even handle deployments.
6. **Keep the main branch stable:**
   Never push broken code to the main branch. It should always be in a working state.

These rules help keep a project healthy and easy to work on. They also make it easier to find bugs, understand what changed, and keep everyone on the same page.

Sources

1. https://learn.microsoft.com/en-us/devops/develop/git/what-is-version-control
2. https://www.atlassian.com/git/tutorials
3. https://git-scm.com/book/en/v2
4. The DevOps Handbook