Rachel Theis

Module 3 Assignment

CSD 380

6.15.25

**Source #1: Version control concepts and best practices by Michael Ernst**

This document is self-described as a brief introduction to version control, but covers the topic quite extensively and is understandable for anyone, even at the beginner level. The best practices for version control, as outlined by the author, are below (Ernst, 2022):

1. Use a descriptive commit message
2. Make each commit a logical unit: This concept makes it easier to track bugs and other activity since pushing several activities at once (i.e. a bug fix, a new feature, and a typo fix) can make it difficult to home in on unexpected behaviors within the code.
3. Avoid indiscriminate commits: Developers should double check their changes before they commit using the following commands (Ernst, 2022):

# Lists all the modified files

git status

# Shows specific differences, helps me compose a commit message.

# If I am using the staging area:

git diff --staged

# Whether I am using the staging area or not:

git diff

# If I am using the staging area:

git commit -m "Descriptive commit message"

# If I am not using the staging area, commit just the files I want to:

git commit file1 file2 -m "Descriptive commit message"

1. Incorporate others’ changes frequently
2. Share your changes frequently
3. Coordinate with your co-workers: Avoid errors caused by same-line editing conflicts by communicating with coworkers when these conflicts may occur.
4. Remember that the tools are line-based
5. Don’t commit generated files
6. Understand your merge tool
7. Obtain your copy (git clone URL, hg clone URL, svn checkout URL)
8. Typical Git Workflow:
   1. On the main branch: git pull
   2. git branch *NEW-BRANCH-NAME*
   3. git checkout *NEW-BRANCH-NAME*
   4. As many times as desired:
      1. Make local edits.
      2. Examine the local edits:  git status and git diff
      3. git commit, or git add , then git commit
      4. git pull
   5. Ensure that tests pass.
   6. git push
   7. Make a pull request for branch *NEW-BRANCH-NAME*
9. Don’t rewrite history: Avoid using rebase, especially if you’re a beginner.
10. Cache your password
11. Enable email notifications for new pushes or commits.

I thought that this guide and list of best practices were extremely helpful. The guide felt thorough but not too “in the weeds”, and I feel this can be applied to any type of project I work on that uses version control. While much of this information wasn’t new to me, I did find it a helpful refresher, especially the reminder to always use commands to verify changes before pushing and to avoid rebase. Since this was updated in 2024, all of the guidelines felt relevant to today’s tech landscape.

URL: <https://homes.cs.washington.edu/~mernst/advice/version-control.html>

**Source #2: Source Code Management (Atlassian)**

This article is part of Atlassian’s larger Git tutorial and explains the importance and benefits of source code management. This article also outlines version control best practices, which I’ve summarized below (Atlassian, n.d.):

1. Commit often
2. Ensure you’re working from the latest version
3. Make detailed notes: The commit log should explain the “why” and “what” of the contents of the commit.
4. Review changes before committing
5. Use branches: Branches are essential for enabling parallel development on a project with multiple collaborators.
6. Agree on a workflow

I thought that this article provided a really helpful framework that outlines essential best practices for version control, like using branches and working from the latest version. Failure to follow these guidelines could lead to major project headaches and confusion, and the conciseness of the list makes it easy to follow and emphasizes the importance of the points that were selected for these guidelines. Interestingly, this article is not against using rebase and paints it as a way to mitigate issues that arise from incorrect commits. Since this article is geared toward professionals and the other article is geared toward students, perhaps the difference in opinion is related to the intended audience of each.

URL: <https://www.atlassian.com/git/tutorials/source-code-management>

**Source #3: Understanding Version Control: A Beginner’s Guide to Git and GitHub (Medium)**

This source is a community-sourced Medium article that stood out to me because it had more engagement than most articles that I see on Medium, indicating common attitudes and advice about version control and Git of software developers and other tech professionals. In addition to offering advice and guidelines for version control, this article also has a helpful how-to guide for setting up repositories and getting started with Git commands. Below is a summary of the author’s version control guidelines, all of which remain relevant today (ProDevOpsGuy Tech Community, 2024):

1. Use meaningful commit messages
2. Always work on feature branches
3. Regularly merge branches and resolve conflicts to avoid more complex conflicts from arising in the future
4. Protect your main branch by requiring pull requests and code reviews
5. Make sure to commit regularly
6. Always pull the latest changes before pushing your work
7. Always work on branches, not directly on main or master

URL: <https://dev.to/prodevopsguytech/understanding-version-control-a-beginners-guide-to-git-and-github-54i9>

**Summary:**

After reviewing these version control guidelines, I think that these guidelines are the most important:

* Commit often and descriptively
  + Why: committing often is a crucial version control practice in order to make changes and actions modular and easy to track. By committing changes often and separately from other changes, the version control is easy to follow, and debugging is easier when issues arise. Similarly, descriptively commenting on commits makes them easier to understand and follow, and makes debugging easier.
* Review changes before committing them
  + Why: By reviewing commits beforehand, developers can minimize errors and keep the version control as clean as possible. Additionally, this practice can ensure that unintended changes are not being pushed to the project, and multiple intended changes aren’t being pushed together.
* Use branches
  + Using branches is essential to parallel and collaborative development, and maintains project security and integrity by staying away from the main
* Communicate
  + In order to effectively use version control, a team must communicate well and establish processes and mutual understandings of version control procedures and practices. Developers can mitigate misunderstanding-related version control challenges by maintaining a clear and open line of communication.

**References**

Atlassian. (n.d.). *Source Code Management | Atlassian Git Tutorial*. Atlassian. https://www.atlassian.com/git/tutorials/source-code-management

Ernst, M. (2022, March 7). *Version control concepts and best practices*. Washington.edu. https://homes.cs.washington.edu/~mernst/advice/version-control.html

ProDevOpsGuy Tech Community. (2024, September 20). *Understanding Version Control: A Beginner’s Guide to Git and GitHub*. DEV Community. https://dev.to/prodevopsguytech/understanding-version-control-a-beginners-guide-to-git-and-github-54i9