Version control in professional software development

Version Control plays a crucial part in software development. As a developer, you'll work with other developers on projects to deliver software to customers. Depending on the role, you could be working with a small team of 2 or 3 developers in a single project or a large team spanning multiple projects. In either scenario, Version Control will be a crucial tool to help your team succeed.

However, Version Control must be complemented by other tools and procedures to ensure quality and efficiency throughout the software development process. In this lesson, we'll explore some of the common tools and strategies developers use in conjunction with Version Control.

Workflow

Using Version Control without a proper workflow is like building a city without traffic lights; without appropriate management, everything will turn into chaos.

For example, let's say you're working on a big project and editing a file. Another developer also starts editing a file. Both of you submit the file to the VCS at the same time. Now there's a conflict! How should the conflict be resolved? A good workflow will have a process for resolving conflicts.

Another example is when a new junior developer is joining your team. If the project code is used for a critical system, it is risky to allow them to submit code changes directly. To solve this, many developers use a peer review system where another developer must review code before it can be merged in.

Workflows are essential to ensure code is managed correctly and reduce mistakes from happening. Different projects will have different workflows. In this course, you'll learn some common workflows using the Git Version Control System.

Continuous Integration

Continuous Integration, or CI, is used to automate the integration of code changes from multiple developers into a single main stream. Using a workflow whereby small changes are merged frequently, often many times per day, will reduce the number of merge conflicts.

This process is widespread in test-driven software development strategies. CI is often used to automatically compile the project and run tests on every code change to ensure that the build remains stable and prevent regressions in functionality.

Continuous Delivery

Continuous Delivery is an extension of Continuous Integration. Once the changes have been merged into the main stream, a Continuous Delivery system automatically packages the application and prepares it for deployment. This helps avoid human error when packaging the application.

Continuous Deployment

Continuous Deployment is an extension of Continuous Delivery. The goal of Continuous Deployment is to deploy and release software to customers frequently and safely. The strategy commonly involves automatically deploying to a test (also known as staging) environment first to validate the deployment package and software changes. Once validated, it can automatically deploy to the live (also known as production) environment for customers.

Conclusion

With these tools and procedures, it is possible to understand how software starts from a developer writing code to being deployed live for customers to use. Of course, there is much more to running a live software service, but that is a lesson for another day.