Unit 1 Module 2: Collaborating with git

Dylan Lane McDonald

CNM STEMulus Center
Web Development with PHP

August 10, 2014

Outline

The STEMulus Center
powered by CNM Ingenuity

Why Use Software Configuration Management?

Software Configuration Management (SCM) systems automate the process of synchronizing code and reconciling code changes. SCM systems also provide full history on changes in the codebase.

Why Use Software Configuration Management?

Software Configuration Management (SCM) systems automate the process of synchronizing code and reconciling code changes. SCM systems also provide full history on changes in the codebase.

SCM systems also allow for non linear development of the codebase by allowing the code to branch into independent development paths. These branches can later be merged into other branches to create a unified, fully integrated codebase.

Other Software Configuration Management Systems

Three major systems are deployed today:

- CVS: Concurrent Version System, this was the original SCM written in 1986 as a series of shell scripts to facilitate in-class collaboration.
- Subversion: Still very widely deployed, Subversion was written as a replacement for CVS.
- git: Written as an improved SCM for the Linux kernel project by Linus Torvalds, git is rapidly becoming the industry standard.

Linus Torvalds

Quote of the Day

"Subversion has been the most pointless project ever started... Subversion used to say CVS done right: with that slogan there is nowhere you can go. There is no way to do CVS right... if you like using CVS, you should be in some kind of mental institution or somewhere else."
$\approx$ *Linus Torvalds*

Linus Torvalds

Quote of the Day

"Subversion has been the most pointless project ever started...Subversion used to say CVS done right: with that slogan there is nowhere you can go. There is no way to do CVS right...if you like using CVS, you should be in some kind of mental institution or somewhere else."
$\approx$ *Linus Torvalds*

Vitæ

Linus Torvalds *is the founder of the Linux project, developed in 1990 when he couldn't afford to buy a SUN workstation. He remains active in both the Linux and `git` projects.*

The STEMulus Center
powered by CNM Ingenuity

git Workflow

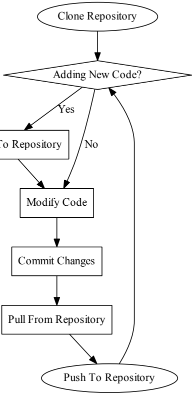

Figure 1: git Workflow

Working within the Workflow

The key to working within the workflow in Figure 1 is to **commit early, push often**. This facilitates the following two positive effects:

Working within the Workflow

The key to working within the workflow in Figure 1 is to **commit early, push often**. This facilitates the following two positive effects:

- Decreased chance of conflicts to be resolved in the codebase.
- Increased history in the codebase, which provides valuable information for your team to use while debugging.

Working within the Workflow

The key to working within the workflow in Figure 1 is to **commit early, push often**. This facilitates the following two positive effects:

- Decreased chance of conflicts to be resolved in the codebase.
- Increased history in the codebase, which provides valuable information for your team to use while debugging.

"Batch commits" are counter-productive to this process and are discouraged. It is better to make many small commits instead of one large one. And as, always, use a cheat sheet! [1, 2]

git Cheat Sheets

Zack Rusin.
git cheat sheet.
http://zrusin.blogspot.com/2007/09/git-cheat-sheet.html.

fournova Software GmbH.
git cheat sheet.
http://www.git-tower.com/blog/git-cheat-sheet/.