A Git Workflow for Open Source Collaboration - Part I - Introduction

Posted At : September 13, 2010 10:59 AM | Posted By : Bob Silverberg
Related Categories: Git Workflow, Git, ValidateThis

As some of you may know, I'm the lead developer on an open source project called **ValidateThis**. I changed the version control software that I'm using for the project from Subversion to Git almost a year ago, and I've been very happy with Git ever since. There was a bit of a learning curve to Git, particularly as I'd never really used the command line much, and it's pretty much required with Git, but that was actually one of the reasons for the switch - to give me the impetus to really learn how to work with Git.

I am lucky enough to have several contributors to the project, and they made me aware recently that the fact that the source is housed in a Git repository is interfering with their ability to make contributions. They all have plenty of experience working with Subversion, but Git is quite new to most of them. Simply learning the syntax of Git is no problem, but what many people find difficult, me included, is figuring out how to change their workflow, as working with Git can be quite different from working with Subversion. So I decided to do some research and come up with a proposal for a workflow for all of the contributors. We're going to give it a try and see how well it goes. If changes need to be made, we'll make them. My hope is that the workflow we devise is one that can be used by other open source projects as well, if they choose to do so.

I am going to describe this workflow in detail through a series of posts on my blog, this post being the first installment. The series will likely contain the following posts:

- Introduction
- Setting Up Your Local Environment
- Developing Code
- Submitting Code to the Project

Credit Where Credit is Due

The overall workflow that I'll be describing is based on one published by **Vincent Driessen** on **his blog** in a post titled ***A successful Git branching model***. I read a number of other excellent posts about Git workflows as I was researching this, so if you'd like to educate yourself before going further here are some links:

- ***A Git Workflow for Agile Teams*** by **Rein Henrichs**
- ***My Common Git Workflow*** by **Yehuda Katz**
- ***My Git Workflow*** by **Oliver Steele**
- ***The Thing About Git*** by **Ryan Tomayko**
- ***Our Git Workflow*** by **Brandon Konkle**

Each of those posts addresses different issues, some of them having to do with collaboration, some having to do with how you work locally, and a lot of talk about rebasing. Some of the posts are a bit lengthy, but I do encourage you to read through them if you want to get a good idea of some of the issues that people have thought about when trying to come up with a Git workflow.

This Workflow is Based on Vincent Driessen's Workflow

Even if you choose not to read any of the posts that I've listed above, I'm afraid you're going to have to read through **Vincent Driessen's post** as it forms the basis of this workflow. I don't think it would be fair for me to simply repeat all of the information from Vincent's blog, so the remainder of this series is going to assume you're familiar with Vincent's workflow.

This Workflow Uses GitHub

There is no reason why you have to use **GitHub** for your remotes when implementing a similar workflow but, because I *am* using GitHub, the instructions that I describe will use terms and screen caps from GitHub, and will be based on some of GitHub's features, including the brand spanking new **pull request system**.

What's Next?

Part II in the series will describe how to set up your local environment to participate in the workflow. It will cover topics including:

- Forking an existing GitHub repo
- Cloning a repo to your machine
- Setting up a tracking branch
- Adding a remote for the main project repo