Source Code Management with Multiple QA environments using Git

Posted At : October 28, 2010 12:05 AM | Posted By : Tariq Ahmed
Related Categories: Tools of the Trade

Up until now we've been using Seapine Surround SCM, it works much like Microsoft's Visual Source Safe in that it's linear development. When we were a very small team of 3 people Surround worked great because our development life cycle was also linear in that we developed in a Development branch, promoted to a QA branch which had a related QA environment, and then promoted to a Staging/PreProduction branch prior to releasing.

Now being slightly larger (7 developers) the linear approach has become a hurdle. We have multiple projects in play, using the same code base, and only one QA environment. This resulted in various issues:

- With two+ projects in *THE* QA environment, you were never really sure if you were testing the project you think you're testing. Or if there's a bug, which project that bug originated from. It's really trying to do integration testing without the premise that all changes are meant to work with each other.
- You'd get a log jam of changes where projects in QA weren't done testing, while in Development the next round of changes on the same files couldn't move forward because they were tied up in QA.
- Hotfixes were problematic. As you'd have to make the change in front of what's in QA, and then make sure it retroactively gets merged back down stream.
- Sometimes a file that was in QA, would have an unrelated change in Development put in that is then tested in QA and then prematurely pushed forward into Staging.

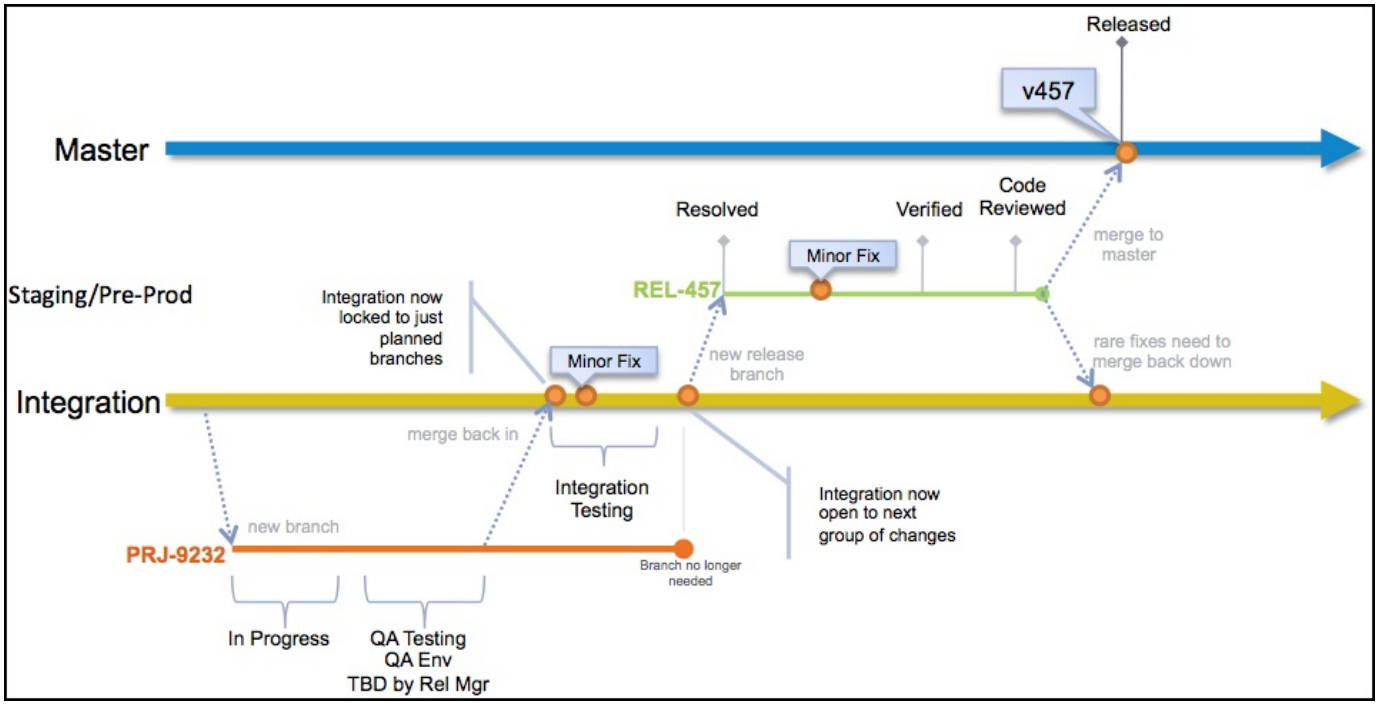
The next step is we needed to introduce multiple QA environments so that we can develop all these project independently of each other, test all these projects independently of each other, integrate them together, and support hotfixes with less grief.

Moving forward...

After doing an analysis of tooling and source code management (scm) processes we determined that Git was the scm tool of choice. It provides a lot of flexibility in that you can merge any which way, and branching is fast and inexpensive, thus making it easy to tailor a process that will work well for what we want to do.

Our process and branching model is completely inspired by this blog posting, so a big kudos to Vincent Driessen for putting that diagram together.

What we did to help train the development and project management staff is break it down into the various scenarios so that they could relate to it in comparison to how they were used to doing things.



We'll probably have to do a few more tweaks to get the kinks out, but if it helps I've made the presentation available for download below (I tried posting to SlideSix, SlideRocket, and SlideShare - but they didn't render it properly).

Download/View the scm process slides.