BASIS has Git Bug Control, Get Git too!

You have probably experienced this dilemma at least once in your career. Somewhere along the way, a well-intentioned person made a change to your product's code that introduced a nasty bug. At the time, you didn't discover it because the code compiled just fine and everything seemed to run okay. Then on a dark and dreary day, a customer crashed into this rather major bug and promptly reported it to you. The customer discovered the bug long after you had made extensive changes to the code. Rather than keeping those changes and trying to patch them, it would be ideal if there was some way to just back the bug out of the code.

But how do you back a bug out? Is there a simple way?

If your code is just a set of files stored in a single location on the server, you have no choice but to try to patch the bug. If you keep backups, you can restore the file from a previous point in time, as long as your backups go back far enough and you know when someone made the offending change...exactly why developers invented source control. Source control systems keep backups of your precious files with a record of when changes were made, what files were changed, how the file content was changed, and who changed them. This makes recovery from this scenario possible and in most cases, very easy.

Source code control has been around in some form since the 1960's. There are a lot of choices out on the market today including Subversion, Microsoft Team Foundation Server, Mercurial, and Git, to name a few. If you work with any kind of files containing valuable information that changes over time, you will find source control invaluable since it allows you to track your changes and revert if necessary. This article takes a close look at how Git benefits BASIS and what it can do for you.

By Shaun Haney
Quality Assurance Engineer

Git, a Distributed Source Control System

Linus Torvalds, the chief architect of the Linux kernel, developed Git out of necessity to maintain the Linux kernel in a source control system. The nature of the Linux kernel project differs from a lot of single-company commercial projects, as its source is developed by various highly distributed contributors all over the world simultaneously.

To accommodate this extreme development environment, Torvalds developed Git as a distributed version control system, able to quickly process and compress large volumes of code, with support for parallel code development. The way this works is that people who want to contribute source to the archive get their own full copy of the archive. The archive is relatively small compared to the amount of code it actually holds. Each version or "commit" in the archive is just a collection of differences or "deltas" from the previous version. Users can update their individual archives from another archive, usually the original archive, by performing a "pull" from that archive. Likewise, once users have changes they would like to contribute to another archive, they can "push" their changes to that archive.

If users are working on long-term changes, they can create their own "branch." Creating a separate branch allows a user to safely create and test changes without affecting the master branch. Once a user is sure that his changes are safe to incorporate into the master branch, he can merge in those changes.

Because of these features, Git has proven a valuable source control solution for the Linux kernel project. In the spirit of open source, Git is not only available to Linux kernel contributors, but to anyone who needs a distributed versioning system.

Git for Version Control

For BASIS, Git is an extremely valuable tool for source control for the AddonSoftware® project. This past June, BASIS converted the long-standing AddonSoftware Subversion (SVN) archive to Git because its development spans several companies with valuable contributions from multiple VARs. In addition, Git supports local version control and switching between branches, both of which satisfy critical needs in this environment. Developers get their own full copy of the repository and create a branch for the feature they add. As they continue to develop, they update regularly from BASIS' repository and push their feature back to BASIS' repository when it is ready. While SVN supports branching, Git puts the entire repository on the developer's local machine to allow quick and seamless switching between branches using the same working directory. This avoids having separate directories for each branch that require individual updating.

Git for Preserving Customizations Through the Upgrade Cycle

Git also serves as a valuable tool to help AddonSoftware partners preserve and update their vertical applications or customizations through the upgrade cycle. BASIS provides a read-only central Git repository that contains each major AddonSoftware release. When AddonSoftware partners want to upgrade their customers' installations with the new release, they clone the BASIS repository and roll it back to the same version that their customer has. Next, the partners add their verticals or customizations to the repository, then pull all the revisions, including the very latest release from the central BASIS archive. During the pull, a merging process takes place that updates the partner's verticals or customizations where needed. They can then port the upgraded vertical or customized version back to their customer's AddonSoftware installation and continue to reuse the cloned archive as the starting point for future upgrades. In this way, Git allows AddonSoftware partners to maintain their vertical or customizations through upgrades, making the overall upgrade process dramatically easier and, more importantly, affordable to the customer.

Imagine how much easier it would be to keep your vertical current with the annual upgrades from the original vendor if your source code control system could automatically manage the upgrades after your first integration. That is what Git does for you; you never have to struggle through an upgrade again and your customers can benefit from every new upgrade and security release with ease.

Git for You

In addition to Git's capabilities for large multi-party projects, Git certainly benefits smaller companies and individuals as well. Benefits include basic source control, speedy updates, local development, branching and merging support, and integrated support in the Eclipse IDE (links.basis.com/eclipse).

- **Basic source code control.** As mentioned in the introduction, any non-trivial project benefits highly from source code control. Source code control gives the developer the ability to back out changes or at least identify a specific change when product development has gone awry. The more frequently you check in your changes, the more you benefit. Most modern source code control systems offer branching and merging, allowing you to introduce drastic changes to your product that temporarily break everything for you without affecting anyone else so long as you're checking it into your own branch.
- **Speedy updates.** Git is a very fast, efficient system. Because of the way Git stores differences, you can often clone an entire repository in Git in as much time as it takes to check out a single revision in SVN. Even better, after you first clone the archive, any updates to or from another archive only consist of differences, making the transaction speedy.
- **Local development.** Unlike many other versioning systems, you do not need access to your central repository to check in changes to Git. You can continue to check in changes to your local repository and push those changes to the central repository once you have network access again.
- **Branching and merging.** Modern versioning systems support branching and merging, but Git excels at it. Since the entire repository is available to you locally, you can seamlessly jump between or create new branches as needed. Whenever you switch between branches, the checkout of the branch is always in your working directory. In centralized versioning systems like SVN, you need to check out the desired branches from the central repository, keeping each branch in its own directory. With Git, creating your own branch is easy and highly recommended if the change involves multiple files or blocks of code. Once you're ready to merge your changes in, Git remembers where you branched and tracks the changes accordingly. In fact, if you change a line of code in your branch, and someone else makes a different change in the same line in the source branch from which you created yours, Git notes the conflict and prevents these changes from overwriting each other.
- **Support in the Eclipse IDE.** BASIS has developed Eclipse plug-ins, introduced at TechCon2013 – the BBj Enterprise Manager and BASIS Development Tools. The Eclipse IDE also comes with EGit that provides Git integration with Eclipse for easy check-in or updating of the source within the IDE when working with BBj code.

Summary

Git is more than a versioning tool; it's a robust tool for anyone who needs to keep their own sets of code or to merge files. BASIS developers and AddonSoftware partners find Git to be a valuable multi-purpose tool with strong capabilities that dramatically improves their productivity.

Whether or not you use Git as your ultimate source code control tool, version control is an asset in all development environments, from the single developer to large, distributed networks. With all of the excellent choices available, everyone can find the tool that best suits their needs and begin reaping the benefits of version control. Once you make the transition, you will wonder how you ever lived without it! ■

Read *'Git'dy Up Developers!* at links.basis.com/12git

"Git" Started Now

Keeping your code under versioning control with Git sounds like a great idea. What do you need in order to get the process going?

- The Eclipse IDE comes with EGit, which has the tools you will need in order to work with Git repositories. Find information on how to get Eclipse and our plug-ins at links.basis.com/eclipse
- Learn how to use Git in the IDE, at wiki.eclipse.org/EGit/User_Guide
- Apress provides an excellent book on Git at git-scm.com/book

With these readily available resources, you can have a Git repository going and start preserving your precious code today!