William Luttmann

Mark Boady

CS 571 Advanced Programming Techniques

August 7, 2020

Research Report 2: GIT

Telepatech, the new company that has been gaining popularity in the tech world, is looking for additional input on how to improve programming collaboration and management as they begin to develop some of their own intellectual property. You may remember this company from when they were looking for suggestions for a standard text editor. Now, I am suggesting that they investigate version control software (VCS) to aid them in collaborative work. VCS is a great way to have a scalable team of programmers working on the same project while maintaining a master or main, up to date, version of the project they are working on. Let us take a deep dive into GIT, one of the most popular VCS to date.

GIT is an open source project developed in 2005 by Linus Torvalds, known as the creator of the Linux kernel. The open source project is in a mature stage as the majority of experienced programmers have used GIT for collaboration or code hosting at some point in their career. “At its most basic, Git is a ‘framework’ for collaborating on a codebase.” (Forest) Git is distributed in nature. Each programmer can have a full working copy of the code that is referred to as a repository. The local repository can contain the full history of changes to the code project or just the specific user’s work on the project. The remote repository is usually hosted on a remote server, like GitHub.com, but can also be set up internally on a local network. We will cover server hosting in detail later on. Git provides the user with some very important functionality

such as cloning, pulling, pushing, and committing to the remote repository. Cloning essentially creates an exact copy of the remote repository on the user's local system. Pulling is the term for importing the most up to date version from the remote repository to the local repository. Committing stages the local repository's changes while pushing, the opposite of pulling, exports the user's committed code to the remote repository. Now that we have covered the basics of GIT, let us explore why implementing it at Telepatech will improve the nature of code collaboration.

When multiple programmers are working as a team to deliver a product, it can be very messy in terms of creating patches and bug fixes and having everyone have the same file in the end when not using some kind of VCS. Without it, programmers will need to send their code files back and forth to each other and make sure that they have up to date copies of everyone else's changes while they send their own. This approach is not the best and is not scalable when teams can grow high in numbers. Telepatech should be implementing the modern form of development best practices which would be using software such as GIT. "It's easy to have a local copy of the codebase fall behind the global copy. Make sure to git pull or fetch the latest code before making updates. This will help avoid conflicts at merge time." (Atlassian 2) When using pull with git, programmers can be sure that the version of code they are working on is the latest copy and they do not have to send code back and forth since they are pulling from a remote repository that is a global source of truth. Another good use of GIT is implementing GitHub collaborators. Adding collaborators "…will give them 'push' access, which means they have both read and write access to the project and Git repository." (6.3 GitHub - Maintaining a Project) This feature can control who is able to update the code such as the program lead.

By controlling who has read and write access, it can greatly improve project management. "Gitflow Workflow is a Git workflow design that was first published and made

popular by Vincent Driessen at nvie. The Gitflow Workflow defines a strict branching model designed around the project release. This provides a robust framework for managing larger projects." (Atlassian 1) The idea of a GIT workflow is essential when planning for project management which introduces the feature of GIT branching. A repository can have a master branch and many other branches used for development. The master branch is usually the main branch for production releases. Other branches may include development work, additional features or releases, and hotfixes. By electing collaborators on each branch, Telepatech can have control over who is able to make changes to which branch. When reviewed, the work done on other branches can then be merged into the master branch. Additionally, changes made through committing to repository branches can be rolled back to some previous commits. This means that if something were to go wrong or if some feature gets scrapped, the code can be readily moved back to a previous state. Project management is made simple with the branching and rollback features of GIT.

Now let us look at the different types of GIT systems and see which one is the best fit for Telepatech. There are many ways GIT can be implemented at the company, but we will be looking at three main implementations. First, Telepatech can set up a GIT repository on their local network. To start, the company will have to make a main directory where the repository will be hosted. Then, "initialize a bare repository which has no working tree. This is important for making sure our initial push and pull between remotes works as intended." (Boardman) After that, each local machine can then clone the bare remote repository. After each machine has their own working copy of the remote master branch, they can then create their development branches and start committing and pushing and pulling as needed. The good thing is that Telepatech does not have to pay for the implementation, but they do have to worry about security and access

controls internally. “Any IT department must fulfill strict security policies. If the policies in your organization require to keep your code in-house, you of course have to deal with hosting your code behind your own firewall.” (Günther) Since Telepatech is a new company that is just starting out, they might not have the security expertise and may want to look at a different implementation.

The second way GIT can be implemented is having a GitLab server set up. GitLab is similar to GitHub as they both provide web hosting for remote repositories, however as Thomas Peham explains, “GitLab realized the need for better and deeper integrations between development and DevOps toolchains. With the latest release of 10.0, GitLab rethinks the scope of tooling for both developers and operation teams.” GitLab has bridged the gap between the programmers and business side of project releases. Peham also explains that GitLab can set role-based permissions instead of just read and write permissions. Issue tracking is also made easier with GitLab which has widget addons for employees that are a little less technical. “Both (GitLab and GitHub) are great issue trackers, especially when connected with a visual bug tracker like Usersnap. While your developers still enjoy the great issue tracking interface of GitLab and GitHub, your testers, colleagues, and clients can simply report bugs through the Usersnap widget.” (Peham) Again, while GitLab’s features are great, since Telepatech is just starting out they may not be able to use all the features intended for downstream users that may or may not be completely useful.

The last and what I believe to be the best implementation of GIT for Telepatech is the GitHub Enterprise setup. From the Enterprise GitHub FAQ, “GitHub Enterprise is the on-premises version of GitHub.com. It makes collaborative coding possible and enjoyable for large-scale enterprise software development teams.” GitHub Enterprise has the same features of

GitHub.com but is packaged to run on the company's local network through a virtual machine. GitHub Enterprise comes with access integration that is already set up at the company so that the already existing access control can be used like LDAP, SAML, or CAS which means that your programmers can use the same login to GitHub Enterprise that they use to log in to their machines. Since this is delivered as a virtual appliance, it can save time as you do not need to have a server rack installed and plugged into the network. Another thing that GitHub Enterprise has to offer is clustering which is great for scaling. Clustering provides the scalability through distributing the load across multiple nodes which may be preferable if Telepatech grows over time.

Now that we have seen how much VCS is needed for collaborating, there is no reason why Telepatech should not use GIT. With its pushing and pulling capabilities, programmers can easily have a full, up to date version of the code they are working on which is shared by all collaborators. The branching feature of GIT improves program management by using the different branches to control the production version and separate development, features, and hotfixes. While there are many ways to implement GIT as VCS in Telepatech, they must decide what setup will be most effective for their environment. Hosting their own internal GIT server is cost effective, but the company may not have the resources to maintain and keep up to date with security and other platform patches. GitLab is a great tool especially with its support for plugins that may help the less technical employees, but Telepatech may not have a need for this support at this time. GitHub Enterprise is a great fit as it is delivered as a virtual appliance which can be patched easily and integrates with the existing authentication system through LDAP, SAML, and CAS. Additionally, many programmers are already familiar with GitHub and will have a minimal learning curve to this system. To all at Telepatech, happy coding!

Works Cited

"6.3 GitHub - Maintaining a Project." *Git*, git-scm.com/book/en/v2/GitHub-Maintaining-a-Project.

Atlassian 1. "Gitflow Workflow: Atlassian Git Tutorial." *Atlassian*, www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow.

Atlassian 2. "Source Code Management: Atlassian Git Tutorial." *Atlassian*, www.atlassian.com/git/tutorials/source-code-management.

Boardman, Simon. "Setting up a Git Repository Across a Local Network." *Simon Boardman*, 12 Jan. 2016, s-boardman.github.io/bioinformatics/local-git/.

"FAQ." *GitHub Enterprise*, enterprise.github.com/faq.

Forest, Danny. "A Beginner's Guide on How to Improve Your Programming Skills." *Medium*, SkillUp Ed, 15 Oct. 2020, medium.com/skilluped/a-beginners-guide-on-how-to-improve-your-programming-skills-769b92fef2f2.

Günther, Tobias. "On-Premise Source Code Management - 7 Git Hosting Platforms Compared." *RSS*, July 2020, www.git-tower.com/blog/on-premise-git-code-hosting/.

Peham, Thomas. "GitLab vs GitHub: What Are The Key Differences?" *Usersnap*, 2 Sept. 2019, usersnap.com/blog/gitlab-github/.