“Git Hub” And Collaboration

By Thomas Park

“Git Hub” is an internet site that has generally been used to host coding projects. It offers free remote “repositories” for code data. These repositories can have one or many collaborators in charge. Not only is code stored remotely, new versions can be branched, updated, and merged-- allowing many people to work together on one coding project.

When I first learned about Git Hub’s branching model, I felt that it would work in other capacities. The staff at the site agreed that projects such as text and music creation were possible-- changes in code and changes in media files work in pretty much the same way. The workflow would be perfect for these kinds of projects.

I believe that the branching model has more potential yet. It is a common issue with group projects that they lack a center-- a locus of documents, policies, data, and so forth-- that can be administered easily. Either one person controls everything, or various elements can be found strewn across a collection of drives and servers.

The Git Hub branching method allows an administrator to compile essential material, and then to give permission, where desired, to specific staff, for particular changes to be made. These changes, if proposed, can be analyzed, and, if accepted, “Merged” into the main branch of data.

The system resembles the current “Google Docs” model, but it does not rely on Google (with their notorious data-mining practices, among other things). It is also a better model for allowing administrators to restore data to previous conditions-- ie. changes are not lost in multiple edits. Branching prevents data loss. And, simply keeping a local repository, and comparing the local to the remote, allows administration to immediately detect if changes have been made, and to decide whether to permit them.

That being said, the branching model is not exclusively authoritarian in flavor. Staff can easily “Request A Pull”-- from their remote version of the directory, they can create branches, and send them for review. As stated, the project head(s) then can decide whether to implement these changes, not to, or to send them back for more development.

An additional benefit of Git Hub is that materials are “transparent”-- i.e. they can be publicly accessed. However, only certain people can make lasting changes to them, at the main hub. Therefore, projects of a public and inclusive nature easily offer core policies and information to interested parties, without violating the integrity of the material.

It may seem clear that Git Hub is not perfect for everything. In its free form, there is little security, so that data meant to be kept private should be stored elsewhere. There are paid platforms that allow for more privacy.

For those interested, I have initiated a collaborative text project. A bit like an “exquisite corpse” project, participants are invited to install Git Bash on their console(s), clone the repository to their local workspace, create a branch, make edits, and request a pull-- so that I can merge the changed files. This project illustrates a sample workflow for a public collaboration.