How Open Source Software Supports the Largest Computers on the Planet
Date: July 18, 2018
Presented by: Ian Lee (Lawrence Livermore National Laboratory) (lee1001@llnl.gov)

Slides:
https://speakerdeck.com/ianlee1521/how-open-source-software-supports-the-largest-computers-on-the-planet

Q: Since moving zfs to github, what portion of development comes from github users at large and what portion, other DOE labs vs. original LLNL manpower.

A: The best place to look for that is https://github.com/zfsonlinux/zfs/graphs/contributors which is the live graph of that information. You can see that the bulk of the work (1709 / 4637 commits) is done by Brian Behlendorf, the lead developer of ZFSonLinux, here at LLNL, but the next several leading contributors are from outside of LLNL.

Q: What was involved in obtaining pre-github history of projects?

A: Depending on the project, this was either to create a new git repository from scratch out of existing archives of the project releases or to migrate the repository from an older version control system (most often SVN). In the case of the former, we would take the archive files (e.g. tarballs or zip files) for versions 1, 2, 3, … and so forth, and check each one in as new “commits” in a new git repository. If the code was already in some form of version control, then we would migrate it forward. In particular, I found this blog post / write-up to be the most useful and have used it many (dozens) of times for various repositories: https://john.albin.net/git/convert-subversion-to-git

Q: What kind of security background checks are required for LLNL collaborators?

A: We don’t have any “security background checks” for collaborators per se. What we do instead is only add LLNL employees to the github.com/llnl organization. Additionally, the default permission in the organization is “Read Only” which means that users don’t get automatic “Write” permissions for being in the organization. For most collaborators, rather than trying to investigator the contributor, our projects take contributions and the maintainers of those projects approve the contributions (almost exclusively via pull

requests) themselves. In many cases, continuous integration (e.g. Travis CI or Appveyor) is used to automatically enforce certain checks, and more are being added.

Q: What are Ian's key 2 or 3 biggest advice/guidance to other labs in moving in these directions

A: Start small and build on early success. Don’t try to do everything at once, but take an incremental approach (much like agile software development) to making impactful changes).