Los Alamos NATIONAL LABORATORY EST. 1943

memorandum

Computer, Computational and Statistical Sciences Division

To: CCS-2, MS D413
CCS-3, MS B256
CCS-6, MS F600
CCS-7, MS B287
From: Ed Dendy, CCS-DO, MS B297
Phone: 505-667-4399
Symbol: CCS-DO-344
Date: November, 18, 2020

Subject: Minimum required process for pushing code and documentation to external repository hosting sites, including Github and Bitbucket

Publishing LANL-developed open-source code on an external hosting platform, e.g. Github, is a release of information and must be considered carefully, and follow a documented review process. Below we give some examples of how this may be done as well as what to do and not to do should the process fail or be missed. While there are other avenues for intentional and unintentional release of source code, such as email, Git tools and repositories provide such flexibility that we see mistakes occur as part of what are considered normal daily operational use of these tools. Hence, we specifically focus on Git in this process. Some important facets pertaining to this memo and process include:

- This process is intended for existing Feynman Center approved open source codes and projects pushing to GitHub or other LANL approved external resource.
- All codes using GitHub should be approved by the Feynman Center.
- Modifications to this process are to be discussed with a relevant Responsible Line Manager (RLM).
- A project may resume its GitLab->GitHub operations once a project PI has notified their RLM that this process has been adopted.
- Project documentation should point to this memo and its extensions as the process used for moving open source code to GitHub.
- Staff that need to open source code should start with the Feynman Center here at codes.lanl.gov.
- For CCS users of Git, this process is mandatory and to be used as their process or as the minimum set of requirements for a foundational basis to their own process.
- For guidance regarding allowed resources for hosting open source code please see the link to Policy P226.pdf

Example 1: Code developed locally on Gitlab (e.g. gitlab.lanl.gov) and mirrored to github.com

1. Mirroring (pushing once a merge request has been approved via review) should only occur after a review by 2 or more approved reviewers

    a. Reviewers should carefully consider export-control concerns
    b. For incremental changes that do not introduce significant new functionality no additional DC review is required
    c. For changes that introduce significant new functionality or algorithms, particularly new physics additions, or new combinations of physics, an additional DC review is required
    d. Changes that add substantial new functionality not covered by the abstract submitted during open-sourcing require a new open-sourcing request
    e. What command line arguments should be built into the push to GitHub? It may be prudent to ensure "-squash" is included for example. That way anything caught at review time and removed will not be included in the push history.
2. Only well-reviewed (at least 2 reviewers) release and pre-release branches should be mirrored. No merge-request or work-in-progress branches should be mirrored. Limit the number of mirrored branches to an absolute minimum. A review includes all history and documentation in addition to the repository source code.
3. Mirroring, after review, can be performed
    a. Manually
    b. Automatically on successful merge of a reviewed merge request
4. All changes to the branches to be mirrored should be via merge requests to facilitate a documented review. For the merge request, use a review template that defines a checklist of steps for developers and reviewers to follow. This can be done via a repository-specific template for merge requests.

Example 2: Code developed directly on github.com

1. For officially established projects this development approach must have been sanctioned by the Feynman Center. See codes.lanl.gov. If issues related to timing occur iwht the open source process it should be discussed with a Responsible Line Manager.
2. That said, the implications of additions or changes being made should be well thought out in advance and written down. If a software project is being run appropriately this should be a natural part of the project process and not create additional overhead.
3. As in Example 1 changes that add substantial new functionality not covered by the abstract submitted during open-sourcing require an new open-sourcing request

Steps to follow should the inadvertent release of information occur off-site (e.g. Github.com), including code.

1. Contact the SIT and Responsible Line Manager (RLM)
    a. You may work with your Group Leader in contacting the SIT for an additional perspective.
    b. Please work with and be patient with the SIT. Should you feel that something is not being handled appropriately, discuss with your Group Leader.
4. Expect to be part of the process. Technical work does not take priority although issues regarding the impact on deliverables should be discussed with your group leader
5. DO NOT attempt to “fix” the situation on your own
6. DO NOT reach out to external repository owners (Github.com)
7. The SIT in conjunction with line management will provide instructions as to how to proceed and who will be responsible.
8. The SIT will work with involved individuals and subject matter experts to determine a path forward. Input and thought at this point is prudent as the SIT does not always understand the many technical aspects of what we do.