Meeting of the Technical Steering Committee (TSC) Board

Tuesday, February 6th 2018
11:00am ET

Meeting Logistics

- https://www.uberconference.com/jeff_ef

- United States : +1 (510) 224-9559 (No PIN needed).

Antitrust Policy Notice

- Linux Foundation meetings involve participation by industry competitors, and it is the intention of the Linux Foundation to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

- Examples of types of actions that are prohibited at Linux Foundation meetings and in connection with Linux Foundation activities are described in the Linux Foundation Antitrust Policy available at http://www.linuxfoundation.org/antitrust-policy. If you have questions about these matters, please contact your company counsel, or if you are a member of the Linux Foundation, feel free to contact Andrew Updegrove of the firm of Gesmer Updegrove LLP, which provides legal counsel to the Linux Foundation.

THE LINUX FOUNDATION

Agenda

- Updated email for Karl: karl@ices.utexas.edu
- Two new ARM servers hosted at TACC:
  - thanks Eric!
- Component Review #5 results posted
- ISC tutorial
  - proposals are due next week, February 13
  - Tutorials are on Sunday, June 24 (9am-1pm for half day)
- Developer Certificate of Origin
- Python 2/3 discussion
- Component deprecation discussion

Developer Certificate of Origin (follow up)

OpenMPI-derived template from last time:

By making a contribution to this project, I certify that:

1. The contribution was created in whole or in part by me and I have the right to submit it under the OpenHPC open source license; or
2. The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the OpenHPC open source license (unless I am permitted to submit under a different license); or
3. The contribution was provided directly to me by some other person who certified (1) or (2) and I have not modified it.
4. I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project and the open source license(s) involved."

DCO v1.1 from Linux Foundation

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I have the right to submit it under the open source license indicated in the file; or

(b) The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license), as indicated in the file; or

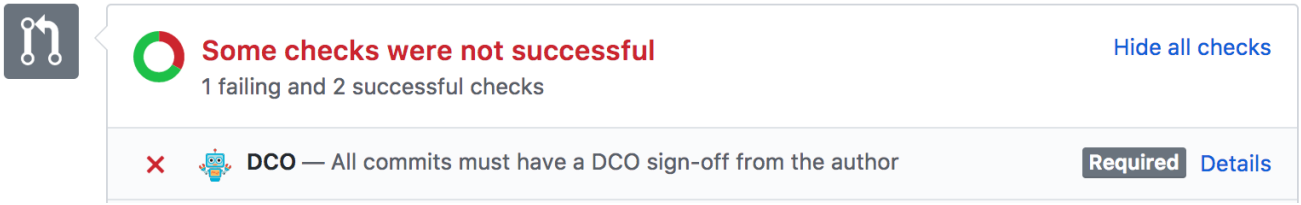
(c) The contribution was provided directly to me by some other person who certified (a), (b) or (c) and I have not modified it.

(d) I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved.

https://probot.github.io/apps/dco/

Developer Contribution Agreement

- In addition to the webhook the OpenMPI folks published for requiring signed PRs (from last time), there look to be other public tools for github, e.g.:
  - https://probot.github.io/apps/dco/



- Propose we try to leverage existing tool first, if that isn't satisfactory, we can host a light-weight web hook on one of our servers

Python discussion: Is it time for python 3?

- The Python core team plans to stop supporting Python 2 in 2020.
- Numpy is dropping support for python 2.7 next year
  - 2018 releases will include support for both 2.7 and 3.4-3.6
- Option 1: OHPC packages only python 3 versions of numpy, scipy, and mpi4py starting with 1.3.4
  - Previous versions still available in older repositories
- Option 2: Add python 3 versions along side updated python 2.7 versions for all OHPC releases through 2018
  - Then see Option 1

- Both options will require a compatibility module to allow users to run either environment

Component deprecation

- We have now completed 5 iterations of the component review process for adding new components!
- As of yet, have not identified any policy around the possible deprecation of components
- High level question: *do we want to devise a companion policy on component deprecation?*

- Assuming the answer is "maybe", let's start thinking about some of the issues:
  - reasons we might want to deprecate a component
  - likely problems associated with deprecation
  - data we might have at our disposal to help support a decision

Component deprecation (cont.)

- Some potential reasons we might want to deprecate a component or particular build configuration (OS, architecture, interconnect, etc)
  - lack of upstream development (although, some established utilities may not require significant ongoing development)
  - inability to land generic patches to upstream development needed for working ohpc builds
  - component becomes available in useable binary RPM form elsewhere (e.g. EPEL)
  - component functionality superseded by newer development project
  - security/correctness considerations
  - broken functionality with other component version upgrades
  - incompatible license change
  - lack of usage/adoption

Component deprecation (cont.)

- Potential problems associated with deprecation:
  - hit on community good will? no one likely happy to see their component dropped
    - would clearly need published policy
    - is a rebuttal process necessary?
  - when would visibility of a deprecated component go away?
    - e.g. would we forcibly remove a previous build from being visible in [base] or [updates] repository on a given branch?
    - or, would we just not update it any more (and stop exercising during integration testing)?
  - introduces a recurring additional policy item for TSC to manage
- Potential advantages:
  - one small way to help control build and CI growth by focusing our resources on components we think provide the most value to the community
  - can perhaps be more willing to take risks on initial submission if there is a downstream pathway to deprecation
  - "threat of deprecation" might potentially o improve upstream development interaction(s)
- Other items for consideration:
  - would there be a minimum amount of time to carry a package before it might be eligible for deprecation consideration?

Component deprecation (cont.)

- Data at our disposal: quantifying usage/adoption seems tricky:
  - we certainly can track downloads on a per component basis (across OS's and architectures)
    - of course, that doesn't necessarily mean they are being used locally
  - some inference based on questions/discussions that come up on the mailing list
  - probably can't ascertain much with folks who mirror entire repository (via repo tarball or rsync)
  - admins who leverage the convenience of our meta-packages may install packages their users don't necessarily access
  - there are projects that could be leveraged for sites to monitor which development "modules" their users load
    - would require an opt-in to monitor
    - we would presumably need a periodic phone home capability to receive the data if we wanted to try and monitor usage at this level

- General thoughts, comments on how to proceed?