Committer Criteria

Table of contents

1 Criteria for Committership

Committers are responsible for reviewing and integrating code changes. The PMC votes to make a contributor a committer based on an assessment of their contributions to the project. Contributions can be made in many ways, and there is no one route to committership. That said, here are the general criteria that the PMC looks for from all potential committers:

- **A history of sustained contribution to the project.** This is a way for a contributor to demonstrate their expertise in an area, and thus their ability to help review and commit contributions by others in that same area. Sustained contribution is also a way of demonstrating commitment to the project, essentially that someone will continue contributing even after they become a committer.
- **High-quality contributions.** As experienced contributors, committers set an example for others in the project, and help inculcate a culture of high-quality contributions. For code contributions, this means clean, documented code which includes unit tests if appropriate and passes precommit checks. For reviews, this means thorough, actionable feedback, and a +1 vote (even non-binding) only when there is a high degree of confidence in the change.
- **Community involvement.** Contributors are always expected to be polite, constructive, and respectful of others during community interactions. Disagreement can (and will) happen over technical issues, but the discussion should remain friendly and focused on technical merits. Committers also have the additional responsibility of mentoring newer contributors, as well as helping to grow the community through actions like responding to emails on the user and dev list.

2 Example Paths to Committership

Here are a few hypothetical example paths to a commit bit:

Han Meimei works at a big company operating a large Apache Hadoop cluster. While deploying the newest version of Hadoop to her staging cluster, she discovers a number of bugs and performance regressions. She files JIRAs for these issues, and is able to post patches for some of them. On the remaining JIRAs, she works with other community members by doing additional debugging on her cluster and testing and reviewing intermediate patches. Additionally, Han Meimei works with the release manager of the release line, *Lei Li*, to make sure critical issues are backported to the next maintenance release. Han Meimei also makes an effort to help fix and review other critical issues targeted at the next maintenance release, even though her cluster is not currently affected. She continues doing similar stabilization work for subsequent maintenance releases in this release line, demonstrating her commitment to shipping high-quality upstream releases.

Elise works at a commercial big data vendor as a developer. Elise ramps up by doing some newbie JIRAs, but then gets involved with a large cross-company development effort

happening on a feature branch. Elise helps to review the design of the feature, providing constructive feedback, and works with other community members to divide the work into subtasks. At this point, the PMC grants Elise branch committership to accelerate development. Elise submits good patches for a number of subtasks, and shows care and thoroughness when reviewing and committing code of others. After the feature branch is later merged to trunk, Elise continues to find and fix bugs related to the feature, as well as reviewing code contributions from others.

Raj works as a test engineer at a commercial big data vendor. Raj notices the lack of integration testing upstream, and contributes a test harness for fault-injection testing which is committed to the project. Raj coordinates with the PMC to get the test harness running on Apache infrastructure, and starts triaging the output and filing JIRAs for relevant bugs with reproductions. Raj helps with gathering logs and reproducing issues, and continues making improvements to the test harness.

Peter is a technical writer interested in improving Hadoop's documentation. He starts by fixing outdated information and incorrect examples, reaching out to subject matter experts and exploring the codebase to clarify technical details. As Peter becomes more familiar with the set of documentation, he realizes that a newly contributed feature didn't include sufficient documentation, and contributes a guide on how to setup and use the feature that also explains some of the design decisions that influenced API design. Peter continues to broaden his doc contributions to more and more areas, and also is active answering user questions on the mailing list. This demonstrates his knowledge of the project, attention to detail, and also his user-focus.