Network
Internet-Draft
Intended status: Best Current Practice
Expires: August 22, 2017

M. Thomson, Ed.
Mozilla
A. Atlas, Ed.
Juniper Networks

February 18, 2017

Using GitHub at the IETF
draft-thomson-github-bcp-00

Abstract

This document describes best practices for working groups that use GitHub for their work.

Note to Readers

Discussion of this document takes place on the GitHub@ietf mailing list (ietf-and-github@ietf.org), which is archived at https://mailarchive.ietf.org/arch/search/?email_list=ietf-and-GitHub .

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 22, 2017.

Copyright Notice

Table of Contents

1. Introduction

The IETF has an open and transparent process for developing standards; the use of GitHub, when used as part of this process as appropriate, can have several objectives. For some technology areas, it can broaden the community that is reviewing and improving the specifications. GitHub provides useful tools to speed up and manage a rapid iteration process for managing changes and tracking issues. Using tools that reduce the friction in rapidly improving documents and getting more relevant reviews can help improve the speed at which a Working Group completes its specifications.

This document describes how the IETF uses GitHub through the development of Internet-Drafts. This concentrates on the work that occurs within IETF working groups. Recommendations for working groups and their chairs are made for integrating these tools with their processes.

This document is meant as a companion to RFC 2418 [RFC2418]. It provides guidance to working group chairs and participants on how they can best use GitHub. The small number of rules in this document are there to ensure common usage patterns between working groups and to avoid issues that have been encountered in the past.

1.1. Integrated Tools: GitLab and Git

Different version control systems are a critical component of software engineering and are quite useful also for document editing. The IETF datatracker can currently provide an subversion repository for each Working Group for its version control system, but git is also possible.

Git is a distributed version control system and both GitLab and GitHub are based around git. Each instance of a repository contains a number of revisions. Each revision stores the complete state of a set of files. Users are able to create new revisions in their copy of a repository and share revisions between copies of repositories.

GitLab provides a simplified and integrated interface to not only git, but also provides basic user management, an issue tracker, associated wiki, project hosting, and more. GitLab is a commercial integrated software product that can be hosted and run by different organizations; a community version is also available.

1.2. GitHub: Community Outreach

GitHub is a service operated at https://GitHub.com/ . GitHub provides a centralized store for git repositories. GitHub is freely accessible on the open Internet, albeit currently only via IPv4.

There are a large number of projects at GitHub and associated a very large community of contributors. One way in which some IETF Working Groups have seen benefit is in the increased reviews and associated issues and improvements that come from broader participation by facilitating those in this community to participate.

This document contains some content that is quite specific to GitHub. A working group that decides to adopt one of the several different alternative services can still benefit from the general guidance in this document.

1.3. Notational Conventions

The words "MUST", "MUST NOT", "SHOULD", and "MAY" are used in this document. It's not shouting; when they are capitalized, they have the special meaning defined in [RFC2119].

2. IETF Administrative Policies

The following administrative rules provide the necessary oversight and transparency. They apply whether GitHub or a publicly-available GitLab instance is used by the Working Group. Working Groups that do not decide to use GitHub or a publicly-available GitLab instance are not impacted.

2.1. Naming and Ownership of Organizations

Each Working Group SHOULD create a new organization for the working group. It SHOULD be named consistently so that it can be found. For instance, the name could be ietf-<wgname> or ietf-<wgname>-wg. A single organization SHOULD NOT be used for all IETF activity, or all activity within an area. Large organizations create too much overhead for general management tasks, particularly when there is a need to maintain membership.

Since an organization must have some owners, that should be done via a team that is given owner privileges. This team MUST include the Area Directors and/or delegates of the Area Directors. This team SHOULD include the Working Group Chairs. A team with administrator access SHOULD be created and MAY include the Working Group Chairs and WG Secretary. Administrator access is preferable, since this does

not also include the ability to push to all repositories and ownership does not grant any other significant privileges.

When an Area Director changes, the outgoing Area Director MUST be removed from the organization's ownership team. This can be done by the continuing AD, the outgoing AD or the WG Chairs. The incoming Area Director and/or delegate MUST be added to the organization's ownership team. When a WG Chair changes, the responsible Area Director or a delegate MUST remove the previous WG Chair from the organization's ownersip or administrative team and SHOULD add the new WG Chair to that team.

When a Working Group is closed, the responsible Area Director is responsible for removing existing members from teams in the organization. Repositories MUST be updated along to indicate that they are no longer under development.

2.2. Backup and Archiving of Working Group's Organization and Repositories

When an IETF Working Group is closed and even when the associated mailing lists are closed, the associated mail archives and datatracker information are backed up and accessible. If a working group has used GitHub or GitLab, any repositories including issues and discussion SHOULD be backed up on IETF resources. It is desirable for those to be accessible via the Working Group's data-tracker page. For example, this might be via URLs listed in the More Info section on the WG Charter page.

The IETF MAY decide to backup information associated with a Working Group's organization periodically. This decision can be made differently per Working Group in consultation with the responsible Area Director.

2.3. Communicating IETF Policies in GitHub or GitLab

One important policy is the IETF IPR policy (see [RFC5378], [RFC3979], and [RFC4879]). Part of this policy requires making contributors aware of the policy.

The wording and details of how to do so are specified at https://trustee.ietf.org/license-for-open-source-repositories.html. The details are copied below, but the IETF web-site is authorative.

The IETF Trust license file for open source repositories [3] MUST be included prominently in any document repository.

Including this information in the CONTRIBUTING file is sufficient.

In addition to the above boilerplate text there is a benefit to including pointers to other working group materials, the IETF datatracker, specific drafts, or websites. Adding such text is at the discretion of the working group chairs.

2.4. Communicating GitHub or GitLab Use inside IETF

Each Working Group MAY set its own policy as to whether and how it uses GitHub or GitLab. It is important that occasional participants in the WG and others accustomed to IETF tools be able to determine this and easily find the policy and GitHub or GitLab organization.

A simple example of how to do this is to include a link to the GitHub organization on the WG Charter page in the Datatracker under More Info. Similarly, if there are multiple mailing list options, links to those mailing lists should be given. An example of this is at https://datatracker.ietf.org/wg/quic/charter/.

3. Deciding to Use GitHub

A Working Group Chairs are responsible for determining how to best accomplish the Charter in an open and transparent fashion. The WG Chairs are responsible for determining if there is interest in using GitHub or GitLab and making a consensus call to determine if a the proposed policy and use is acceptable. Chairs SHOULD involve area directors in this decision if they intend to use GitHub for anything more than managing of edits.

While a document editor can still use GitHub independently for documents that they edit, even if the working group does not expressly choose to use GitHub, any such public respository MUST follow the guidelines in Section 2.3. This recognizes that editors have traditionally chosen their own methods for managing the documents they edit but preserves the need for transparent contributions with awareness of IPR considerations.

3.1. What to Use GitHub For

Working group chairs have to decide what GitHub features the working group will rely upon. Section 4 contains a more thorough discussion on the different features that can be used.

Once a document is published in a repository on GitHub, many features like pull requests, issue tracking or the wiki can be individually disabled. If specific features are not used by the working group in the development of the document, disabling those features avoids creating confusion in the wider community about what can be used.

3.2. Working Group Policies

Working group chairs that decide to use GitHub MUST inform their working groups of their decision on the working group mailing list. An email detailing how the working group intends to use GitHub is sufficient, though it might be helpful to occasionally remind new contributors of these guidelines.

Working group chairs are responsible for ensuring that any policy they adopt is enforced and maintained.

Updating the README or CONTRIBUTING file in the repository with details of the process ensures that the process is recorded in a stable location other than the mailing list archive. This also makes any working group policies available to casual contributors who might only interact with the GitHub repository.

GitHub prominently links to the CONTRIBUTING on certain pages. This file SHOULD be used in preference to the README for information that new contributors need. A link to the CONTRIBUTING file from the README is advised.

3.3. Repositories

New repositories can be created within the working group organization at the discretion of the chairs. Chairs could decide to only create new repositories for adopted working group items, or they might create repositories for individual documents on request.

All repositories for working group documents MUST be public. Repositories for private documents MAY be kept private, but only where there is a specific reason for doing so. For instance, a document that details a security vulnerability might be kept private prior to its initial publication as an Internet-Draft. Once an Internet-Draft is published, repositories SHOULD be made public.

The adoption status of any document MUST be clear from the contents of the repository. This can be achieved by having the name of the document reflect status (that is, draft-ietf-<wg>-... indicates that the document was adopted), or through a prominent notice (such as in the README).

Experience has shown that maintaining separate repositories for independent documents is most manageable. This allows the work in that repository to be focused on a single item.

Closely related documents, such as those that together address a single milestone, might be placed in a single repository. This

allows editors to more easily manage changes and issues that affect multiple documents.

Maintaining multiple documents in the same repository can add overheads that negatively affect individual documents. For instance, issues might require additional markings to identify the document that they affect. Also, because editors all have write access to the repository, managing the set of people with write access to a larger repository is more difficult.

3.4. Editors and Contributors

Working group chairs MUST give document editors write access to document repositories. This can be done by creating teams with write access and allocating editors to those teams, or by making editors collaborators on the repository.

Working group chairs MAY also grant other individuals write access for other reasons, such as maintaining supporting code or build configurations. Working group chairs, as administrators or owners of the organization might also have write access to repositories. Users other than document editors, including chairs, SHOULD NOT write to working group documents unless with prior coordination with document editors.

Working groups MAY create a team for regular contributors that is only given read access to a repository. This does not confer additional privileges on these contributors, it instead allows for issues and pull requests to be assigned to those people. This can be used to manage the assignment of editorial or review tasks to individuals outside of the editor team.

3.5. Document Formats

In addition to the canonical XML format [RFC7991], document editors might choose to use a different input form for editing documents, such as markdown. The choice of input format is left to document editors.

4. Contribution Methods

Contributions to documents come in many forms. GitHub provides a range of options in addition to email. Input on GitHub can take the form of new issues and pull requests, comments on issues and pull requests, and comments on commits.

4.1. Issues

The GitHub issue tracker can be an effective way of managing the set of open issues on a document. The record of issues - both open and closed - can be a useful way of recording decisions made by a working group.

Issues can be given arbitrary labels, assigned to contributors, and assembled into milestones. The issue tracker is integrated into the repository; an issue can be closed using a special marker in a commit message.

Working group chairs MUST decide how the GitHub issue tracker are used. Use of the issue tracker could be limited to recording the existence of issues, or it might be used as the venue for substantial technical discussion between contributors.

4.1.1. Issue Labelling

A system of labelling issues can be effective in managing issues. For instance, marking substantive issues separately from editorial can be helpful at guiding discussion. Using labels can also be helpful in identifying issues for which consensus has been achieved, but that require editors to integrate the changes into a document.

Labels can be used to identify particular categories of issues or to mark specific issues for discussion at an upcoming session.

If labels are a core part of working group process, chairs MUST communicate any process to the working group. This includes the semantics of labels, and who can apply and remove these labels.

4.1.2. Closing Issues

Editors have write access to repositories, which also allows them to close issues. The user that opens an issue is also able to close the issue. Chairs MUST determine who is permitted to close an issue and under what conditions.

4.2. Pull Requests

Pull requests are the GitHub feature that allow users to request changes to a repository. A user does not need to have write access to a repository to create a pull request. A user can create a "fork", or copy, of any public repository. The user has write access to their own fork, allowing them to make local changes. A pull request asks the owner of a repository to merge a specific set of changes from a fork (or any branch) into their copy.

Editors SHOULD make pull requests for all substantial changes rather than commiting directly to the "master" branch of the repository.

Pull requests have many of the same properties as issues, including the ability to host discussion and bear labels. Critically, using pull requests creates a record of actions taken.

For significant changes, leaving a pull request open until discussion of the issue within the working group concludes allows the pull request to track the discussion and properly capture the outcome of discussions.

Groups of editors could adopt a practice of having one editor create a pull request and another merge it. This ensures that changes are reviewed by editors. Editors are given discretion in how they manage changes.

4.2.1. Discussion on Pull Requests

In addition to the features that pull requests share with issues, users can also review the changes in a pull request. This is a valuable feature, but it has some issues.

Comments in a review other than a summary are attached to specific lines of the proposed change. Such comments can be hard or impossible to find if changes are subsequently made to the pull request. This is problematic for contributors who don't track discussion closely.

For this reason, working group chairs SHOULD discourage the use of inline comments for substantial technical discussion of issues.

4.2.2. Merging Pull Requests

Working groups MUST determine who is permitted to merge pull requests. Document editors SHOULD be permitted to merge pull requests at their discretion. This requires that editors exercise some judgment. Working group chairs MAY occasionally identify a pull request and request that editors withhold merging until working group consensus has been assessed.

Note that the copy of a document that is maintained on GitHub does not need to be a perfect reflection of working group consensus at every point in time. Document editors need some flexibility in how they manage a document.

4.3. Monitoring Activity

Several working groups have created read-only mailing lists that subscribe to activity notifications on repositories. The volume of information on these lists can be too high to monitor actively, but access to an archive of actions can be useful.

A working group that uses GitHub SHOULD provide this facility. However, setting up this mailing list can be onerous and better solutions are still being sought.

5. Advice to Editors

Document editors are primarily responsible for maintaining documents. Taking on a few additional tasks can greatly improve the process for the working group.

Using GitHub means that it is more likely that a contribution is made by users who aren't very familiar with the work. If a duplicate issue is raised, point the user to the existing issue before closing the issue. If a contributor seems rude in a comment, be courteous in response.

Pull requests from new contributors can contain errors or omissions. Some contributors won't natively speak English, so changes might have grammatical errors. If a change is generally sound, rather than rejecting the pull request or requesting changes, accept the change and then make any minor corrections yourself.

Never close a pull request or issue without first understanding why it was made and then explaining why you aren't accepting it. If you are uncertain, ask a chair for guidance.

If a contributor makes a comment that raises what you believe to be a new issue, create an issue for them. If the issue has an obvious solution, consider creating a pull request. It doesn't matter what venue the issue was raised in, email, issue discussion, a pull request review, capturing issues quickly ensures that problems become visible and can be tracked.

This takes a little more effort, but these simple steps can help encourage contributions, which will ultimately improve the quality of your document.

6. Internet-Drafts

During the development of a document, individual revisions of a document can be built and formally submitted as an Internet-Draft. This creates a stable snapshot and makes the content of the in-progress document available to a wider audience.

Editors SHOULD endeavour to create a new Internet-Draft submission two weeks prior to every session (see Section 7.1 of [RFC2418]). Participants in a session can't be expected to monitor changes to documents in real-time; an Internet-Draft ensures that there is a common, stable state that is known to all participants.

Working group chairs MAY request the creation of an Internet-Draft at any time, in consultation with document editors.

7. Assessing Consensus

The work that occurs on GitHub could be part of the consensus process, but the ultimate decision on consensus regarding a document is made by the chairs [RFC2026].

Monitoring activity on GitHub could require a greater time commitment than following a mailing list. This is because there is an increased volume of activity to follow. Participants who wish to limit this time commitment might follow GitHub activity selectively, either by following only specific issues or by occasionally reviewing the state of the document. Chairs are reminded that assessing consensus based on GitHub content alone MUST NOT be assumed to reach all interested participants.

A working group chair SHOULD consult the working group mailing list for any issue that is potentially contentious. Relying on input provided through GitHub alone might result in gaining input from a narrower set of participants. This includes important milestones like working group last-call, where review from the widest possible audience ensures a higher quality document. Managing input from multiple sources in assessing consensus is similar to what is needed when balancing mailing list discussion versus in-person meeting discussion.

8. Continuous Integration

Various third-party services offer the ability to run tests and other computation when changes are made to a document.

One common practice is to use these continuous integration services to build a text or HTML version of a document. This is then

published to GitHub Pages, which allows users to view a version of the most recent revision of a document.

Continuous integration can also validate pull requests and other changes for errors. The most basic check is whether the source file can be transformed successful into a valid Internet-Draft. For example, this might include checking that XML source is syntactically correct.

For documents that use formal languages a part of specifications, such as schema or source code, a continuous integration system might also be used to validate any formal language that the document contains. Tests for any source code that the document contains might be run, or examples might be checked for correctness.

9. GitHub Limitations

At the time of writing, GitHub.com is not reachable using IPv6. This is an affront to all that the IETF stands for and a slap in the face to all the people who worked so hard to design and deploy the latest version of the Internet Protocol. While we can collectively be ashamed and disappointed that this is the situation, that doesn't necessarily make the service any less useful.

10. Security Considerations

Continuity of operations is always a consideration when taking a dependency on an external service. If GitHub were to fail in some way, anyone relying upon its services would be seriously affected.

Consistent use of git reduces the exposure to a system failure because the primary repository is replicated in multiple locations. This extends to web pages that are hosted because the content of the page is saved in the main repository. Maintaining a mirror of a repository that is hosted on GitHub is relatively simple and might be considered as a way to provide a backup for the primary repository.

However, other information maintained on GitHub is more vulnerable to loss. This includes issues and discussion on those issues, discussion and reviews of commits and pull requests, and any content hosted on the wiki. Tools exist for extracting this information for backup.

Malicious actions by compromised or malcontent editors, chairs and area directors are relevant in maintaining the integrity of the content that GitHub hosts. Backups allow for recovery of content, and regular submissions as Internet-Drafts ensure that work is not lost completely.

11. IANA Considerations

This document has no IANA actions.

12. References

12.1. Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <http://www.rfc-editor.org/info/rfc2026>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3979] Bradner, S., Ed., "Intellectual Property Rights in IETF Technology", BCP 79, RFC 3979, DOI 10.17487/RFC3979, March 2005, <http://www.rfc-editor.org/info/rfc3979>.

[RFC4879] Narten, T., "Clarification of the Third Party Disclosure Procedure in RFC 3979", BCP 79, RFC 4879, DOI 10.17487/RFC4879, April 2007, <http://www.rfc-editor.org/info/rfc4879>.

[RFC5378] Bradner, S., Ed. and J. Contreras, Ed., "Rights Contributors Provide to the IETF Trust", BCP 78, RFC 5378, DOI 10.17487/RFC5378, November 2008, <http://www.rfc-editor.org/info/rfc5378>.

12.2. Informative References

[RFC2418] Bradner, S., "IETF Working Group Guidelines and Procedures", BCP 25, RFC 2418, DOI 10.17487/RFC2418, September 1998, <http://www.rfc-editor.org/info/rfc2418>.

[RFC7991] Hoffman, P., "The "xml2rfc" Version 3 Vocabulary", RFC 7991, DOI 10.17487/RFC7991, December 2016, <http://www.rfc-editor.org/info/rfc7991>.

12.3. URIs

[2] https://trustee.ietf.org/license-for-open-source-repositories.html

Appendix A. Acknowledgments

This work wouldn't have been possible without the hard work of those people who have trialed use of GitHub at the IETF.

Authors' Addresses

Martin Thomson (editor)
Mozilla

Email: martin.thomson@gmail.com

Alia Atlas (editor)
Juniper Networks

Email: akatlas@gmail.com