Workgroup: Network
Internet-Draft: draft-ietf-git-using-github-02
Published: 19 November 2019
Intended Status: Best Current Practice
Expires: 22 May 2020
Authors: M. Thomson   B. Stark
         Mozilla      AT&T

# Using GitHub at the IETF

## 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.

Source for this draft and an issue tracker can be found at https://github.com/ietf-gitwg/using-github.

## Status of This Memo

## Copyright Notice

## Table of Contents

## 1. Introduction

The IETF has an open and transparent process for developing standards. The use of GitHub or similar tools, when used as part of this process, can have several objectives. GitHub provides tools that can be helpful in editing documents. Use of this service has proven to reduce the time that Working Groups need to produce documents and to improve the quality of the final result.

The use of source control improves traceability and visibility of changes. Issue tracking can be used to manage open issues and provide a record of their resolution. Pull requests allow for better engagement on technical and editorial changes, and encourage contributions from a larger set of contributors. Using GitHub can also broaden the community of contributors for a specification.

The main purpose of this document is providing guidelines for how Working Groups might integrate the capabilities provided by GitHub into their processes for developing Internet-Drafts.

This document is meant as a supplement to existing Working Group practices. It provides guidance to Working Group chairs and participants on how they can best use GitHub within the framework established by RFC 2418 [RFC2418]. This document aims to establish norms that reduce the variation in usage patterns between different Working Groups and to avoid issues that have been encountered in the past.

A companion document, [GH-CONFIG], describes administrative processes that support the practices described in this document.

### 1.1. Distributed Version Control Systems

Version control systems are a critical component of software engineering and are quite useful also for document editing.

Git is a distributed version control system. 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.

## 1.2. GitHub

GitHub is a service operated at https://github.com/. GitHub provides centralized storage for git repositories. GitHub is freely accessible on the open Internet (see Section 10), albeit currently only via IPv4.

GitHub provides a simplified and integrated interface to not only git, but also provides basic user management, an issue tracker, associated wikis, project hosting, and other features.

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

## 1.3. Other Services

Git is not the only version control system available, nor is GitHub the only possible choice for hosting. There are other services that host revision control repositories and provide similar additional features to GitHub. For instance, BitBucket, or GitLab provide a similar feature set. In addition to a hosted service, software for custom installations exists.

This document concentrates primarily on GitHub as it has a large and active community of contributors. As a result, some content might not be applicable to other similar services. A Working Group that decides to adopt an alternative tool or service can still benefit from the general guidance in this document.

## 1.4. Document Goals

This document aims to describe how a Working Group might best apply GitHub to their work. The intent is to allow each Working Group considerable flexibility in how they use GitHub.

This document does require that policies for use of GitHub are agreed and clearly communicated within the Working Group (see Section 2). The remainder of the document contains guidelines and advice on how to construct a workable policy.

The requirements here apply to the case where Working Groups decide to use GitHub as a primary means of interaction. Individuals can set their own policies when using GitHub for managing their own drafts, or for managing drafts that they edit on behalf of a Working Group that has not explicitly adopted GitHub.

For both sets of users, this document aims to provide some amount of advice on practices that have proven to be effective.

This document only aims to address use of GitHub in developing documents. Working Groups could choose to use the tool to aid in managing their charter or session materials such as agendas, minutes, and presentations. Though the advice here might apply more broadly, using GitHub to manage other material is out of scope for this document.

### 1.5. Notational Conventions

The words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" are used in this document. It's not shouting; when they are capitalized, they have the special meaning defined in BCP 14 [RFC2119] [RFC8174].

## 2. Administrative Policies

The following administrative rules provide the necessary oversight and transparency.

### 2.1. Organizations

Organizations are a way of forming groups of contributors on GitHub. Each Working Group SHOULD create a new organization for the Working Group. A Working Group organization 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.

Each organization requires owners. The owner team for a Working Group repository MUST include responsible Area Directors. Area Directors MAY also designate a delegate that becomes an owner and Working Group chairs MAY also be owners.

A team with administrator access SHOULD be created for the Working Group Chairs and any Working Group 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.

Details about creating organizations adhering to these guidelines can be found in [GIT-CONFIG].

### 2.2. Communicating Policies

Each Working Group MAY set its own policy as to whether and how it uses GitHub. 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 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

Working Group Chairs are responsible for determining how to best accomplish the Charter in an open and transparent fashion. The Working Group Chairs are responsible for determining if there is interest in using GitHub and making a consensus call to determine if the proposed policy and use is acceptable.

Chairs MUST involve Area Directors in any decision to use GitHub for anything more than managing drafts.

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 repository MUST follow the guidelines in BCPs 78 and 79 ([COPYRIGHT], [IPR]). 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 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.

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.

Repositories MUST include a copy or reference to the policy that applies to managing any documents they contain. 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 Working Group policies available to casual contributors who might only interact with the GitHub repository.

GitHub prominently links to the CONTRIBUTING file 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.

The set of GitHub features (Section 4) that the Working Group relies upon need to be clearly documented in policies. This document provides some guidance on potential policies and how those might be applied.

Features that the Working Group does not rely upon SHOULD be made available to document editors. Editors are then able to use these features for their own purposes. For example, though the Working Group might not formally use issues to track items that require further discussion in order to reach consensus, keeping the issue tracker available to editors can be valuable.

Working Group policies need to be set with the goal of improving transparency, participation, and ultimately the quality of the consensus behind documents. At times, it might be appropriate to impose some limitations on what document editors are able to do in order to serve these goals. Chairs SHOULD periodically consult with document editors to ensure that policies are effective and not unjustifiably constraining progress.

## 3.2. 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 within the Working Group organization 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 overhead that negatively affects 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 (Section 3.3).

## 3.3. 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.

## 3.4. 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. Markdown-based formats have proven to be more accessible for new contributors, though ultimately decisions about format is left to document editors.

Formats that are not text-based SHOULD NOT be used, as these are ill-disposed to the sorts of interaction that revision control enables.

## 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. Issue Tracker

The GitHub issue tracker can be an effective way of managing the set of open issues on a document. 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.

When deciding to use GitHub, Working Group Chairs MUST decide how the GitHub issue tracker is 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.

A Working Group policy MAY require that all substantive changes be tracked using issues. Suggested policies for the use of the GitHub issue tracker are the primary subject of Section 5.

#### 4.1.1. Issue Labels

A system of labeling 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. Section 5.4 describes some basic strategies that might be adopted to manage decision-making processes.

#### 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 provide guidance on who is permitted to close an issue and under what conditions.

Restrictions on closing issues are generally not advisable until a document has reached a certain degree of maturity.

#### 4.1.3. Reopening Issues

Issues that have reached a resolution that has Working Group consensus MUST NOT be reopened unless new information is presented.

For long-running work items, new contributors often raise issues that have already been resolved. Chairs need to assess whether the arguments offered represent new information or not. This can require some discussion to determine accurately. Resolved issues MUST remain closed unless there is consensus to reopen an issue.

### 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 committing directly to the "master" branch of the repository. A pull request creates an artifact that records the reasons for changes and provides other contributors with an opportunity to review the change. Pull requests that address substantive issues SHOULD mention the issue they address in the opening comment.

**Note:** This document assumes that there is a unified effort on a document, all concentrated on a git "master" branch. More advanced usage of git is not in the scope of this document.

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

GitHub produces individualized email notifications of activity that each user can adjust to their preferences. In addition to these, some Working Groups have created read-only mailing lists that receive notifications about activity on Working Group 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.

An alternative is to rely on periodic email summaries of activity, such as those produced by a notification tool like github-notify-ml. This tool has been used effectively in several Working Groups, though it requires server infrastructure.

A Working Group that uses GitHub MAY provide either facility at the request of the chairs.

## 5. Typical Working Group Policies

Current experience with use of GitHub suggests a few different approaches to greater use of the tool in Working Groups.

This section describes some basic modes for interacting with GitHub, each progressively more involved. This starts with a very lightweight interaction where document management is the only feature that is formally used, then progressively more intensive use of the GitHub issue tracking capabilities are described. These approaches differ primarily in how discussion of substantive matters is managed. Most of the advice in this document applies equally to all models.

Working Groups can adjust these policies to suit their needs, but are advised to avoid gratuitous changes for the sake of consistency across the IETF as a whole.

## 5.1. Document Management Mode

In this mode of interaction, GitHub repositories are used to manage changes to documents, but the bulk of the work is conducted using email, face-to-face meetings, and other more traditional interactions. The intent of this policy is to enable document and issue management using GitHub while minimizing the complexity of the process.

In the version of this mode with the least interaction with GitHub, a repository is created for the purposes of document management by editors. Editors might maintain issues and pull requests for their own benefit, but these have no formal standing in the Working Group process.

## 5.2. Issue Tracking Mode

In addition to managing documents, the Working Group might choose to use GitHub for tracking outstanding issues. In this mode of interaction, all substantive technical discussions are tracked as issues in the issue tracker. However, discussion of any substantial matters is always conducted on mailing lists.

Under this mode, issues and pull requests can be opened by anyone, but anything deemed substantive MUST be resolved exclusively on the mailing list. Discussion on GitHub is kept to a minimum. Only editorial matters can be resolved using the issue tracker.

Chairs and editors are given discretion in determining what issues are substantive. As documents mature, it is generally prudent to err more toward consulting the mailing list where there is doubt. As with other Working Group decisions, chairs are the arbiters in case of dispute.

A recurrent problem with this mode of interaction is the tendency for discussions to spontaneously develop in the issue tracker. This requires a degree of discipline from chairs and editors to ensure that any substantive matters are taken to the mailing list.

As mailing lists remain the primary venue for discussion of substantive matters, this mode and the document management only modes remain those most compatible with existing work practices for Working Groups. Participants in a Working Group that operates under either model can reasonably be expected to receive all relevant communication about the work of the group from the mailing list.

Though the mailing list is used for making decisions, the issue tracker can still be a useful record of the state of issues. It is often useful if chairs or editors record details of decisions in issue comments when closing issues as resolved.

## 5.3. Issue Discussion Mode

This GitHub interaction mode differs from the other modes in that discussion relating to substantive technical matters is allowed to occur on GitHub issues. Though decisions are always subject to confirmation on the mailing list, participants are permitted to conduct substantive discussions on the issue tracker. In some cases, this can include making some decisions without involving the Working Group mailing list.

A Working Group mailing list remains a critical venue for decision making, even where issue discussion occurs elsewhere. Working Group mailing lists generally include a wider audience than those who follow issue discussion, so difficult issues always benefit from list discussion.

Decisions about Working Group consensus MUST always be confirmed using the Working Group mailing list. However, depending on the maturity of documents, this might be a more lightweight interaction, such as sending an email confirmation for a set of resolutions made using GitHub.

Using the mailing list to resolve difficult or controversial issues is strongly encouraged. In those cases, the issue tracker might be used to more fully develop an understanding of problems before initiating a discussion on the mailing list, along lines similar to the design team process (see Section 6.5 of [RFC2418]).

As a more involved process, adopting this mode can require changes in policies as documents become more mature. It is possible to use different processes for different documents in the Working Group.

Working Group chairs SHOULD confirm that the Working Group has consensus to adopt any process. In particular, the introduction of a more tightly-controlled process can have the effect of privileging positions already captured in documents, which might disadvantage alternative viewpoints.

### 5.3.1. Early Design Phases

During early phases of the design of a protocol, chairs MAY allow editors to manage all aspects of issues. Editors are permitted to make decisions about how to both identify and resolve technical issues, including making any changes that editors feel necessary.

Chairs need to explicitly decide that this sort of process is needed and announce the decision to the Working Group. In many cases, documents that are adopted by a Working Group are already sufficiently mature that a looser process is not beneficial. The primary reason to grant editors more discretionary power is to improve the speed with which changes can be made. The risk is that design changes might not always reflect the consensus of the Working Group.

Changes made by editors under this process do not completely lack oversight. GitHub and git provide tools for ensuring that changes are tracked and can be audited. Within the usual Working Group process it is expected that Internet-Drafts will receive regular review. Finally, process checkpoints like Working Group Last Call (WGLC; Section 7.4 of [RFC2418]) provides additional safeguards against abuse.

Working Groups are advised against allowing editors this degree of flexibility for the entirety of a document lifecycle. Once a document is more stable and mature, it is likely appropriate to move to a more tightly controlled process.

### 5.3.2. Managing Mature Documents

As a document matures, it becomes more important to understand not just that the document as a whole retains the support of the Working Group, but that changes are not made without wider consultation.

Chairs might choose to manage the process of deciding which issues are substantive. For instance, chairs might reserve the ability to use the design label to new issues (see Section 5.4.1) and to close issues marked as design. Chairs should always allow document editors to identify and address editorial issues as they see fit.

As documents mature further, explicit confirmation of technical decisions with the Working Group mailing list becomes more important.

Gaining Working Group consensus about the resolution of issues can be done in the abstract, with editors being permitted to capture the outcome of discussions as they see fit.

More mature documents require not only consensus, but consensus about specific text. All substantive changes to documents that have passed WGLC SHOULD be proposed as pull requests, and MUST be discussed on the mailing list, and MUST have chairs explicitly confirm consensus. Chairs MAY institute this stricter process prior to WGLC.

**Note:** It is generally sufficient to trust editors to manage adherence with these policies, aided by the transparency provided by the version control system. There are tools that can be used to more tightly control access to repositories, but they can be overly constraining.

## 5.4. Issue Labelling Schemes

Several schemes for use of issue labels in managing issues have been used successfully. This section outlines these strategies and how they might be applied.

A design/editorial split (see Section 5.4.1) is useful in all cases that the issue tracking capability is used. Working Groups that only use GitHub for issue tracking might find that distinction sufficient for their needs.

Working Groups or editors might use additional labels as they choose. Any label that is used as part of a process requires that the process be documented and announced by Working Group chairs. Editors SHOULD be permitted to use labels to manage issues without any formal process significance being attached to those issues.

### 5.4.1. Design/Editorial Labelling

The most important distinction about an issue is whether it is substantive. The labels of design and editorial are used to represent this distinction.

An issue labeled as design has or might have a substantive effect on a document. For protocol specifications, a design issue is one that might affect implementations or interoperability requirements. Addressing a design issue ultimately requires Working Group consensus, even if the resolution is to make no change.

An issue labeled as editorial has no substantive effect on a document, except to the extent that addressing the issue might make understanding the specification easier. Resolution of editorial issues can be left to the discretion of editors.

This distinction can be applied to all types of document. For instance, a design issue for an Informational document might be raised to discuss possible changes to important concepts in the document.

### 5.4.2. Decision Labelling

Labels can be used to manage processes. As documents mature and issues become more numerous, labels can be used to clearly mark the status of issues. In particular, labelling of issues can be used to help in managing Working Group decisions.

For documents that are less mature, issues with resolutions but no specific proposals for changes to text might be marked editor-ready as a way of signaling that there is consensus about an approach, but no specific proposal. Chairs might use this to signal that discussion is complete and that editors are to be given discretion in the construction of text.

In contrast, if specific text is a prerequisite for resolving issues, as might be the case for more mature documents, a proposal-ready label might be used by editors to mark issues that they believe to have acceptable resolutions.

For resolved issues, a has-consensus label might be used by chairs to mark issues for which formal Working Group decisions have been made (Section 6.1 of [RFC2418]).

A v2 or next-version label might be used to mark and thereby save issues for a future version of or extension to a protocol, particularly where a resolution is made to take no action.

### 5.4.3. Component Labelling

Repositories with multiple interrelated documents or a complex document with multiple logical components might benefit from labels that identify different aspects of the work. The choice of appropriate labels for components will depend on the structure of specific documents.

### 5.4.4. Other Labels

Other labels can be used depending on the needs of editors and Working Group processes. For example,

* An invalid label might be used for issues that were raised in error.

* A blocked label might indicate an issue is awaiting resolution of an external process or related issue.

* A parked label might be used to indicate issues that do not require immediate Working Group attention.

## 6. Internet-Draft Publication

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. Documents submitted as Internet-Drafts are not expected to address all open issues or merge outstanding pull requests.

Editors SHOULD create a new Internet-Draft submission two weeks prior to every session, which includes IETF meetings, other in-person meetings, and telephone or video conferences (see Section 7.1 of [RFC2418]). Though discussion could use the current version of a document from version control, participants in a session can't be expected to monitor changes to documents in real-time; a published Internet-Draft ensures that there is a common, stable state that is known to all participants.

Internet-Drafts that use a GitHub repository SHOULD include a notice that includes a reference to the repository. This notice might also include information about where to discuss the draft.

Revisions used to generate documents that are submitted as Internet-Drafts SHOULD be tagged in repositories to provide a record of submissions.

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 can 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. Other participants might not use GitHub at all. Chairs are reminded that assessing consensus based on GitHub content alone cannot be assumed to reach all interested participants.

Chairs MUST consider input from all discussion venues when assessing consensus including GitHub, mailing lists, and in-person meetings.

Each venue has different selection biases that might need to be considered.

A Working Group chair MUST 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.

If permitted, GitHub will be used for technical discussion and decisions, especially during early stages of development of a document. Any decisions are ultimately confirmed through review, and ultimately, through Working Group Last Call (see Section 7.4 of [RFC2418]).

The use of issues and labels has proven to be effective in managing contentious issues. Explicitly labeling closed issues to explicitly identify those with formal consensus means that there is no confusion about the status of issues.

## 8. Continuous Integration

Various third-party services offer the ability to run tests and other work 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. Including a prominent link to this version of the document (such as in the README) makes it easier for new contributors to find a readable copy of the most recent version of a draft. In addition, including links to differences between this generated version and any published document helps contributors identify recent changes.

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

For a document that use formal languages as part of the specification, 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. 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 (e.g., 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.

## 10. 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.

## 11. 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.

Widespread use of git reduces the exposure to a system failure because the primary repository is replicated in multiple locations.

This includes hosted web pages; the content of web pages is maintained as a branch 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.

The potential for malicious actions by compromised or malcontent editors, chairs and area directors is 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.

## 12. IANA Considerations

This document has no IANA actions.

## 13. References

### 13.1. Normative References

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

**[GIT-CONFIG]** Cooper, A. and P. Hoffman, "GitHub Configuration for IETF Working Groups", Work in Progress, Internet-Draft, draft-ietf-git-github-wg-configuration-03, 21 October 2019, <http://www.ietf.org/internet-drafts/draft-ietf-git-github-wg-configuration-03.txt>.

**[IPR]** Bradner, S. and J. Contreras, "Intellectual Property Rights in IETF Technology", BCP 79, RFC 8179, DOI 10.17487/RFC8179, May 2017, <https://www.rfc-editor.org/info/rfc8179>.

**[RFC2026]** Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://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, <https://www.rfc-editor.org/info/rfc2119>.

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

**[RFC8174]** Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

### 13.2. Informative References

**[GH-CONFIG]** Cooper, A. and P. Hoffman, "GitHub Configuration for IETF Working Groups", Work in Progress, Internet-Draft, draft-ietf-git-github-wg-configuration-03, 21 October 2019, <http://www.ietf.org/internet-drafts/draft-ietf-git-github-wg-configuration-03.txt>.

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

## Appendix A. Experiences from Working Groups

### A.1. CORE

The CoRE WG (Constrained RESTful Environments) has been actively using the Trac/SVN combination offered by the Tools Team for its older drafts.

Some newer drafts (including some drafts that are not yet WG drafts but could be considered candidates for that) are now being worked on in the core-wg GitHub organization.

These drafts generally use Martin Thomson's template, except where the build process (examples, grammars) is much more complicated than can easily be supported by this template.

For most repos, a CI (continuous integration) process is set up that generates a readable editor's copy (in HTML form) as well as a diff from the most recent submitted version (tools TXT diff), linked from the README; both have turned out to be very valuable. (Unfortunately, the travis-based CI process is somewhat brittle, so there is an appreciable failure rate.)

We try to keep discussion on the mailing list (as opposed to getting them entirely in the GitHub issues), but may not have been very successful in that; it definitely requires constant vigilance.

The WG Wiki says:

> With respect to the mode of operation of the repository, the CoRE WG follows the lead of the HTTPBIS WG. Specifically that means that GitHub issues are welcome to record editorial issues as well as technical ones; as are "pull requests" (forks of the repository with fixes for an issue). However, technical discussion should not happen in the forums implicitly created by the issues, but on the WG mailing list.

We currently do not have an active backup regime.

## A.2. QUIC

The QUIC WG was chartered in October 2016, and has been using GitHub very intensively.

We created a GitHub organization called "quicwg", which the WG chairs administer. Under that organization, we set up two teams, one for WG document editors and one for regular contributors. Membership in the former team is contingent on being chosen as an editor for a WG deliverable. The latter team is more open, and consists of people that the chairs and editors want to assign reviews or issues to. Obviously, anyone can raise issues, comment on them, submit pull requests, etc. The benefit of the "contributors" team really lies in allowing the assignment of tasks to individuals, which is otherwise not possible.

Underneath the "quicwg" organization, we created two repositories, one for WG materials and one for our base WG drafts. Only the chairs have commit permissions to the WG materials repo, which is mostly used to hold presentations and other materials from our various meetings. This repo is configured to not allow issues to be raised or have a wiki (we instead store Markdown files inside the repo.)

Our second repo, for "base drafts", is where most of the work occurs. The decision to use a common repo for several drafts was deliberate. QUIC is a complex protocol with a complex specification, text moves between different documents and issues can affect several. Maintaining each draft in a separate repo, while "cleaner" on first impression, actually complicates this workflow. When the WG adopts additional drafts, we will decide on a case-by-case basis whether they will be made part of the "base drafts" or if we create a new repo underneath the organization. Since Martin Thomson is an editor, we use his setup template to rapidly publish HTML editor copies of the specs.

The "base drafts" repo is configured to allow issues to be raised, and its wiki is enabled (but rarely used.) Editors (and chairs) have commit rights to this repo.

We use sets of labels to tag issues that are raised. One set simply indicates which draft(s) an issue applies to, or whether it is potentially of broad "design" impact, or "editorial" in nature so that an editor can use his or her own discretion to resolve it without WG consensus. A second set is used to track the WG consensus on each issue (with states that currently include "needs-discussion", "confirm-consensus", "notify-consensus" and "editor-ready"). Issues progress from "needs-discussion" to either "confirm-consensus" or "notify-consensus". The former is entered when consensus amongst the participants in the discussion has emerged, and the WG needs to confirm this consensus on the list. The latter is entered when a consensus call happened at a WG meeting, and the mailing list needs to confirm this consensus. (It is not clear if two separate labels actually make all that much sense here.) Once WG consensus has been established, an issue is labeled "editor-ready".

Within only a few months of being chartered, QUIG WG had ~250 issues raised, many of which attracted dozens of comments. Good issue topics and actively searching for prior issues before opening new ones is essential to manage the workflow.

In order to allow WG participants to follow the activity on GitHub without needing to check the GitHub web site, we have set up a separate "quic-issues" mailing list at the IETF. It was a deliberate decision to use a list other than the regular WG mailing list. First, because we are intensively using GitHub, a lot of notifications get generated (dozens per day), which would drown out other list traffic, Second, the issues list is configured as a read-only list, where all incoming email is rejected, except for some whitelisted senders. The intent is to keep all discussion on the regular WG mailing list, or on GitHub tickets. (While GitHub will correctly reflect email replies to issue notifications, they seem to lose sender information, which is useless.)

Getting GitHub notifications to go to this list was mildly painful, and involved creating a dummy "IETF QUIC WG" GitHub user account, whose subscription email address is the quic-issues list address. The dummy user was made a member of the QUIC GitHub organization, and will therefore by default "track" all repo activity. This will cause GitHub to create the desired stream of notification emails to an IETF list. One caveat here is that GitHub uses the email address associated with the user who is interacting with the web site as the sender address of notification emails, which requires regular whitelisting in mailman. It also means that these users are allowed to otherwise email the issues list; we trust they don't. This email integration is rather dissatisfyingly complex; we'd be interested to learn of a better way.

## A.3. HOMENET

After Martin Thomson's presentation on using GitHub (presented at the WG Chairs lunch), the homenet chairs (one of the chairs to be precise) set up https://github.com/ietf-homenet-wg. WG draft authors were asked if they wanted to use it. All draft authors agreed to try, so copies of the current drafts were converted to Markdown (which seemed to be the recommended format) and separate repositories created for each draft. GitHub teams (comprised of authors) were created for the drafts, as per available instructions.

The repositories were created using instructions from Martin Thomson's template. But since instructions for gh-pages proved too confusing, the gh-pages were created by manually uploading (using the GitHub UI) the .html and .txt files output by "make gh-pages" to the gh-pages branch. README.md was then edited to point to the uploaded .html file. However, one of the authors decided it was easier to use Google docs to receive comments and maintain an Editor's copy of the draft (in Markdown). He is able to give permission (on request) to other WG group members to comment directly on Google docs. Therefore the copy on GitHub is out-dated and has effectively been abandoned.

The main authors of the other two drafts decided not to use the repositories created for them, but wanted a new repository for one of the drafts. The authors have collaborated on GitHub in this new repository, but the WG has not been invited to look, comment, or contribute there. This is probably for the best, since the draft pointed to by the README.md is the version in the gh-pages of the old repository for the same draft.

In short, the use of GitHub has been highly experimental, and no formal WG activities have taken place there. The GitHub organization has effectively been a playground area for the chairs and authors to stumble around to figure out how to use this tool.

## A.4. BABEL

After Martin Thomson's presentation on using GitHub (presented at the WG Chairs lunch), one of the authors of a draft in the babel WG asked the chairs if they might consider setting up a GitHub organization. In the absence of a response (and the absence of follow-up to the absent response), the draft was put in a personal GitHub repository. The repository was created using instructions (as best as they could be understood) from Martin Thomson's template. The gh-pages, continuous integration (CI), and such proved beyond comprehension to someone who had only rudimentary (almost non-existent) experience with makefiles, CI, and the inner workings of any of the tools involved; but who, fortunately, did have a laptop running Linux, experience with vi, and had formed a very basic understanding of git (mostly in the context of Bitbucket) over the past year.

Carsten Bormann was kind enough to convert the existing XML to Markdown (since the recommendation was to use Markdown). This turned out to be an excellent recommendation. Writing drafts using Markdown is likely to be significantly easier for many people. The rules are easier and it's so much simpler to figure out what's wrong.

The repository was used by the two draft authors to have discussions using Issues. These were useful to track agreements between the authors and to create to-do lists.

The repository has also been used to ask WG members to look at the Editor's draft (manually pushed to gh-pages) and see if comments had been handled to their liking. WG members were not asked to use Issues to make comments. Comments from WG members were all discussed on the WG list.

Updates to the draft have been pushed directly to the master branch. There was one attempt to first create a separate branch and then a pull request to the master. This proved to be too much effort in the context of this draft.

## Appendix B.  Acknowledgments

This work wouldn't have been possible without the hard work of those people who have trialled use of GitHub at the IETF. Alia Atlas contributed significant text to an earlier version of this document.

The experiences of the CORE WG in Appendix A.1 were contributed by Carsten Bormann. The experiences of the QUIC WG in Appendix A.2 were contributed by Lars Eggert.

## Authors' Addresses

Martin Thomson
Mozilla

Email: mt@lowentropy.net

Barbara Stark
AT&T

Email: barbara.stark@att.com