iTC User Guidance for GitHub

Version 0.2, 2019-06-24

Table of Contents

Background . . . 1
Intended Audience . . . 1
Resources . . . 1
1. Introduction . . . 1
1.1. Changes for GitHub . . . 2
1.2. Translating into GitHub . . . 2
2. Introduction to GitHub . . . 3
2.1. GitHub Account . . . 3
2.2. Overview of GitHub Sections . . . 4
2.2.1. Code Overview . . . 4
2.2.1.1. Changing the Branch . . . 5
2.2.2. Issues Overview . . . 6
2.2.3. Pull Requests Overview . . . 7
2.2.4. Wiki Overview . . . 9
3. Writing in GitHub . . . 10
3.1. Using Markdown/Asciidoctor . . . 10
3.2. Internal link references . . . 11
3.3. Referencing a Person . . . 12
4. Using Issues . . . 12
4.1. Reviewing Issues . . . 12
4.2. Creating New Issues . . . 12
4.2.1. Additional Fields . . . 13
4.2.1.1. Assignees . . . 13
4.2.1.2. Labels . . . 13
4.2.1.3. Milestone . . . 14
4.3. Link to Specific Document Locations in Issues . . . 15
4.3.1. Multi-line Permalinks . . . 16
5. Using Pull Requests . . . 17
5.1. Reviewing Pull Request Conversation . . . 17
5.2. Reviewing Pull Request Changes . . . 17
5.2.1. Commits View of a Pull Request . . . 18
5.2.2. Files Changed View of a Pull Request . . . 18
5.3. Pull Request Changes Display . . . 18
5.4. Adding Comments Directly into Pull Requests . . . 19
5.4.1. Resolving Comments in a Pull Request . . . 21
5.5. Creating a New Pull Request . . . 21
5.6. Working with Draft Pull Requests . . . 24
5.6.1. Creating a Draft Pull Request . . . 24

5.6.2. Continuing Work on a Draft Pull Request 24
5.6.3. Publishing a Draft Pull Request 25
5.7. Approving Pull Requests 25
6. File History 27

Acknowledgements

This document was created by the CCUF Team Tools Working Group for the benefit of everyone involved with Common Criteria.

If you want to participate in the maintenance of this or any other documents, send an email to ccuf-team-tools-wg@googlegroups.com.

Background

This document is intended as an introduction on how to use GitHub for the purposes of participating in an iTC. The CCUF Team Tools WG is providing a set of guidance and templates on using several free tools that can be used to create and maintain the documentation produced by an iTC.

Many of these tools are not necessarily familiar to the people who perform most of the tasks associated with an iTC, so several guides have been created to guide new users to these tools in how they will be used within the iTC.

For more information about the full set of tools and their guides, please see the CCUF Team Tools website.

Intended Audience

This guide is intended for anyone who is participating in an iTC using GitHub. While there are also guides for more advanced users, this guide will be useful for anyone participating in an iTC using GitHub for managing documentation.

This guide is specifically targeted to using GitHub through a web browser. There are additional tools that can be used with GitHub offline that are covered separately and build on the same functions that are described in this guide.

Resources

ADD REFERENCES HERE TO OTHER TTWG DOCS

1. Introduction

This guide will show how to use GitHub for making comments and editing the documents of an iTC. There are many reasons to do this, but among the most important are these:

- Concurrent editing - using GitHub it is possible for multiple people to work on the same document, at the same time, and have all edits tracked independently
- Comment tracking & history - using GitHub comments on documents (or any questions) can be

tracked, discussed, linked to actual edits and eventually closed while remaining available for later review (i.e. that time when you know something was discussed but can no longer remember why you came to a specific decision)

- Versioning/Release control - GitHub is built for the purpose of maintaining and releasing source code, so provides the tools needed for configuration and release management built-in

By taking advantage of the capabilities of GitHub, an iTC should be able to manage the process of ongoing creation and editing of their PP and SD without trying to email documents around with someone being stuck to integrating all the changes and comments. This more interactive process will hopefully better support ongoing efforts within the community.

1.1. Changes for GitHub

While there are many benefits, using GitHub does require some changes to how we would normally work on a document. Most of us are probably familiar with working on Word documents (in your editor of choice), with all the rich WYSIWYG formatting. The problem with these document formats is that they do not readily allow for multiple editors in their binary format.

The primary change for working in GitHub is the change in document format, from a WYSIWYG editor to working in plain text. The great advantage of plain text though is that it allows you to easily focus on the content, and this content can be easily indexed by GitHub to track multiple editors working at the same time. Changes as small as one character are readily tracked in GitHub and can be easily seen.

Of course we don't want plain text Protection Profiles, so we need to have some sort of syntax that allows us to specify things like headers, tables, bullets and such as we expect in our documents. While there are many formats that can be used for this, this guide points to the Asciidoctor processor to generate the styles we expect. The CCUF Team Tools WG has provided a template document titled ***Asciidoctor for iTC Syntax Reference*** for more information about the syntax that is used to create the styles that are most likely to be used in the iTC documents.

Another change is in how to comment on the documents. While today in a Word document you are most likely to just use the comment feature and highlight the place you are commenting on. Within GitHub this is still possible, but the feature is called an Issue (as opposed to a comment). The great thing about Issues though is that they do not need to be removed from the document when it is published, so the history of the comment and any discussion (and even a resolution) are maintained.

1.2. Translating into GitHub

As mentioned, there are some changes to how you would normally work in a Word document to how you will need to work in GitHub, and one of the biggest change is in the terminology, so here is a short description of the two main changes.

Comment → Issue

In a Word document you add Comments to the document. In GitHub you will instead create Issues. An Issue can be created that is tied to a specific point in a document or it can be a topic to discuss (i.e. not tied to a specific document or item within a document).

Change/Edit → Pull request (PR)

In a Word document when Track Changes is enabled, you are able to see the suggested edit and the replaced text. The equivalent in GitHub is to create a Pull request. This is how GitHub tracks changes made to documents and allows further discussions on the changes.

Version → Branch (sort of)

In GitHub active work is done on a branch. Generally you will work in a "develop" branch which is basically the working copy of the document until these are committed to the "Master" branch. At some point the "Master" is published and this will create say v1.0 of you document (or document set).

Repository

In GitHub a repository is the entirety of all files, Issues, Pull Requests, even the Wiki associated with a project. It is possible that the iTC may create several repositories under the iTC to separate work into different areas. Each repository has its own files, Issues and Pull Requests (though it is possible to link between them).

2. Introduction to GitHub

2.1. GitHub Account

The first step to using GitHub is to create an account. These are free (one of the reasons GitHub was chosen) and the sign up is found on the home page.

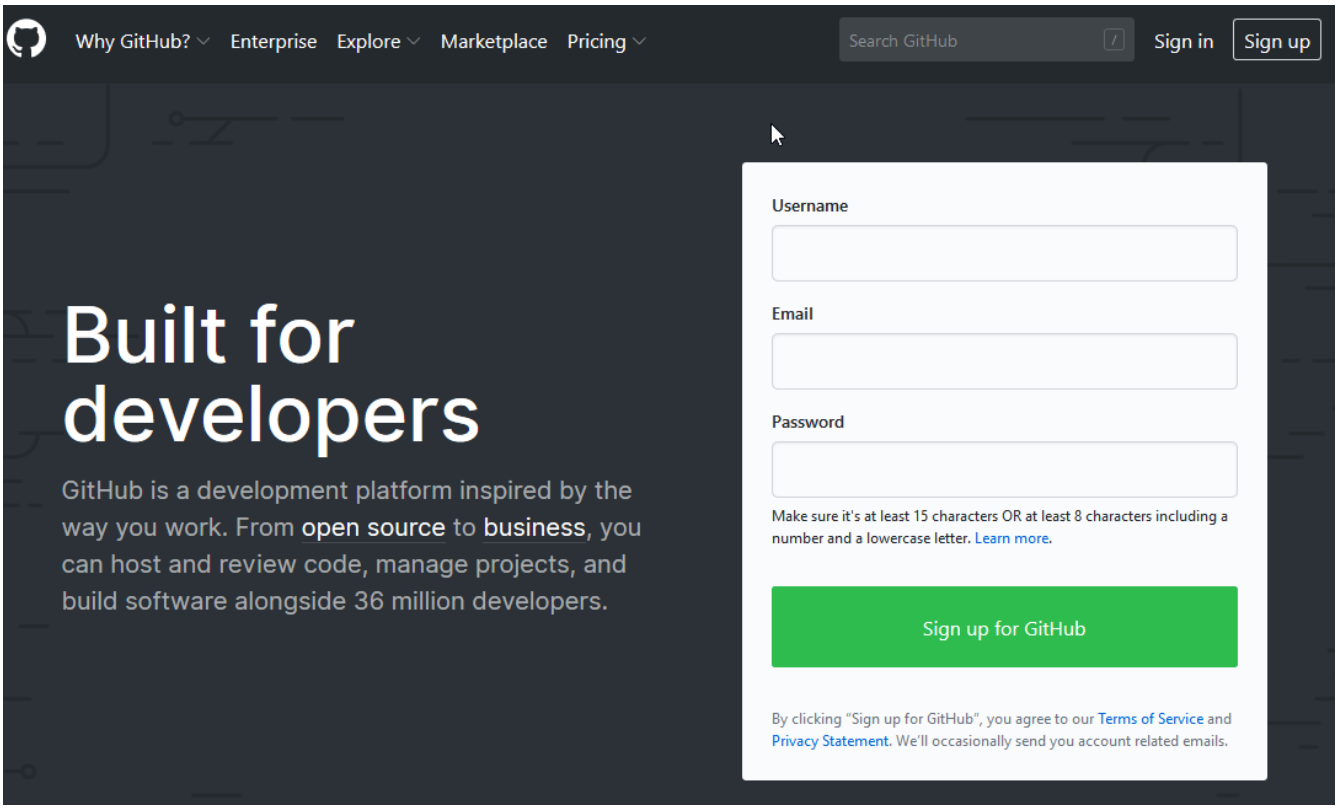


Figure 1. GitHub Sign up

Once you have created your account, you should provide the username to the iTC GitHub administrators. This is not required, but is recommended (and is required if you are to have

elevated privileges within the iTC).

Once you have created your account, sign in to interact with GitHub.

2.2. Overview of GitHub Sections

There are four primary areas within GitHub you will interact with.

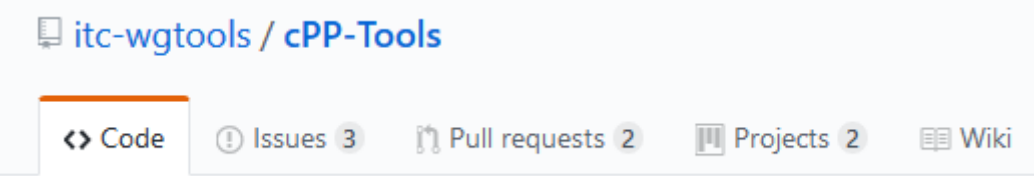


Figure 2. GitHub Sections

2.2.1. Code Overview

The Code section is like a folder of all the documents contained in the repository. There will be folders with documents inside like you would expect, though there will not be multiple versions of a single file (i.e. iterations of the document), only the one for the branch you are working on.

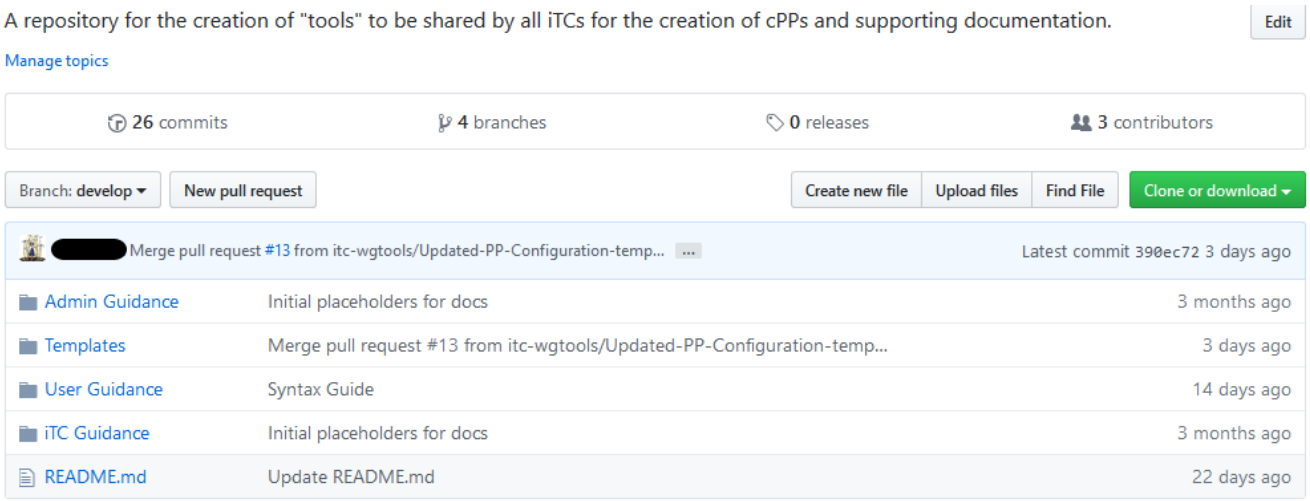


Figure 3. GitHub Code

Clicking on a folder will open that folder and show the files inside. The iTC administrator will likely have created folders to hold different, related documents.

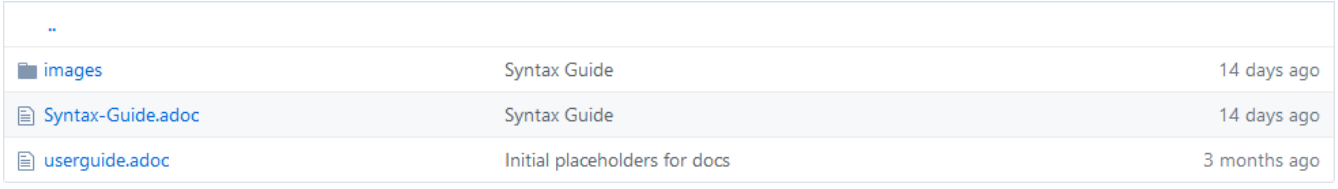


Figure 4. GitHub Code Subfolder

Since we are using Asciidoctor as the file format most of the files you see should end in ".adoc" (though you may also see PDF as output or images that were used in the documents).

Clicking on a file will open the file and display it (GitHub mostly parses the Asciidoctor files, so

while not exactly the final output, it will be pretty close).

2.2.1.1. Changing the Branch

As noted above, branches are used to show different versions (such as the target publishing branch, and any others that are being worked on in the meantime). There will always be two primary branches as noted, and generally work will be done in the develop branch. Additional branches will be created during the editing process. To switch between branches, click the Branch button and select the branch you want to work on.

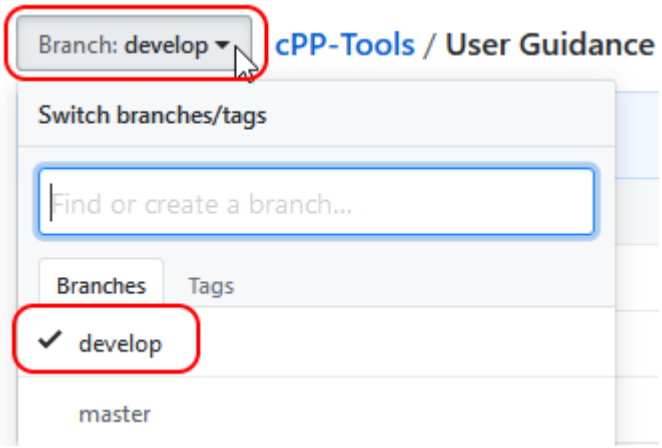


Figure 5. GitHub Change Branch

Changing the branch will show you the current state of the files stored within that branch. So for example if the develop branch has added a new image that isn't present in the existing Master (i.e. the current release), switching to the Master branch would not show that image while the develop branch will.

The administrator will set the default branch you should work on (usually develop), so you probably will not need to change branches often.

When you change the branch you are working on, the files shown in the Code view will change to the current branch. This could cause files to disappear or appear depending on the current status of each branch (for example if a file is being worked on in the develop branch that has not yet been committed to Master, then switching the branch will cause that file to appear/disappear in the list of files).

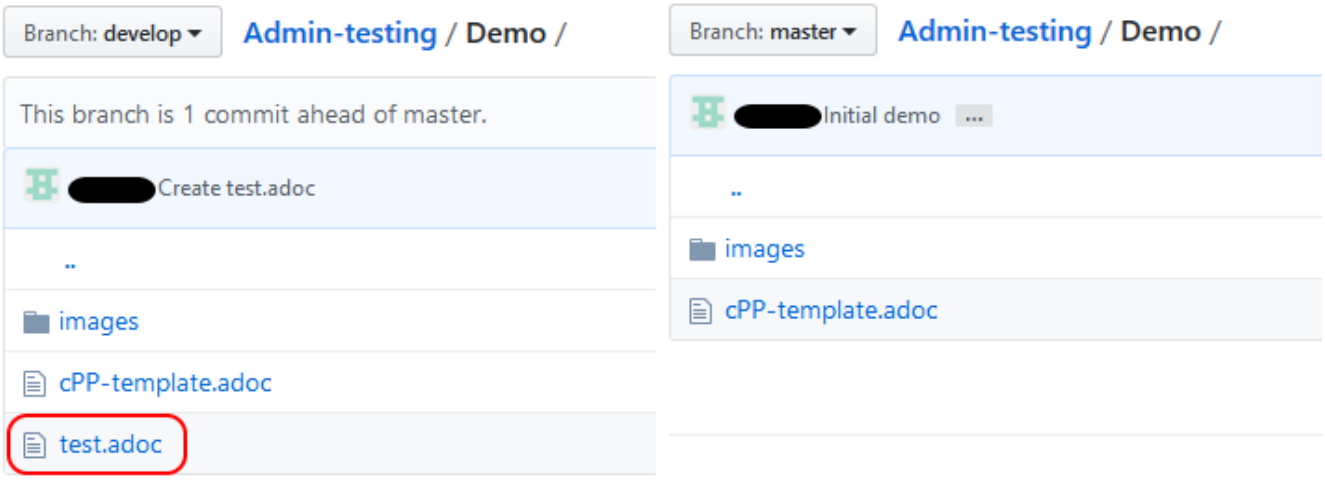


Figure 6. GitHub Files Changing in Different Branches

2.2.2. Issues Overview

The Issues area is basically the comments section. From here you can see open issues and directly create new ones.

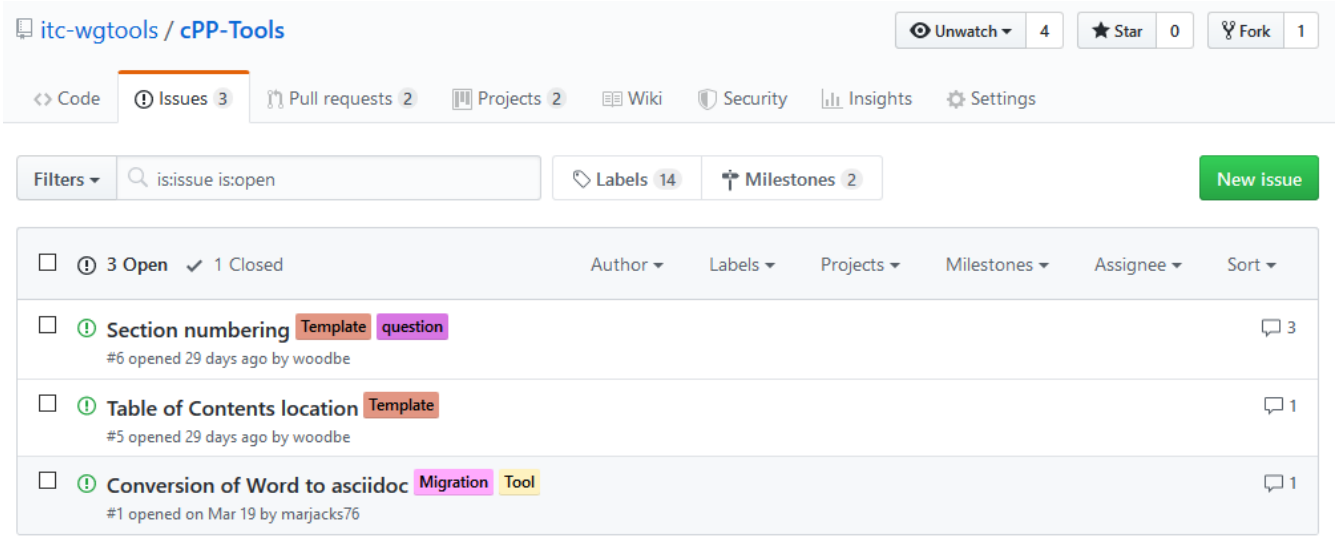


Figure 7. GitHub Issues

Clicking on an Issue title will open the Issue, showing the conversation in a style similar to a forum (each person's post in order of them being added from the first to the last at the bottom).

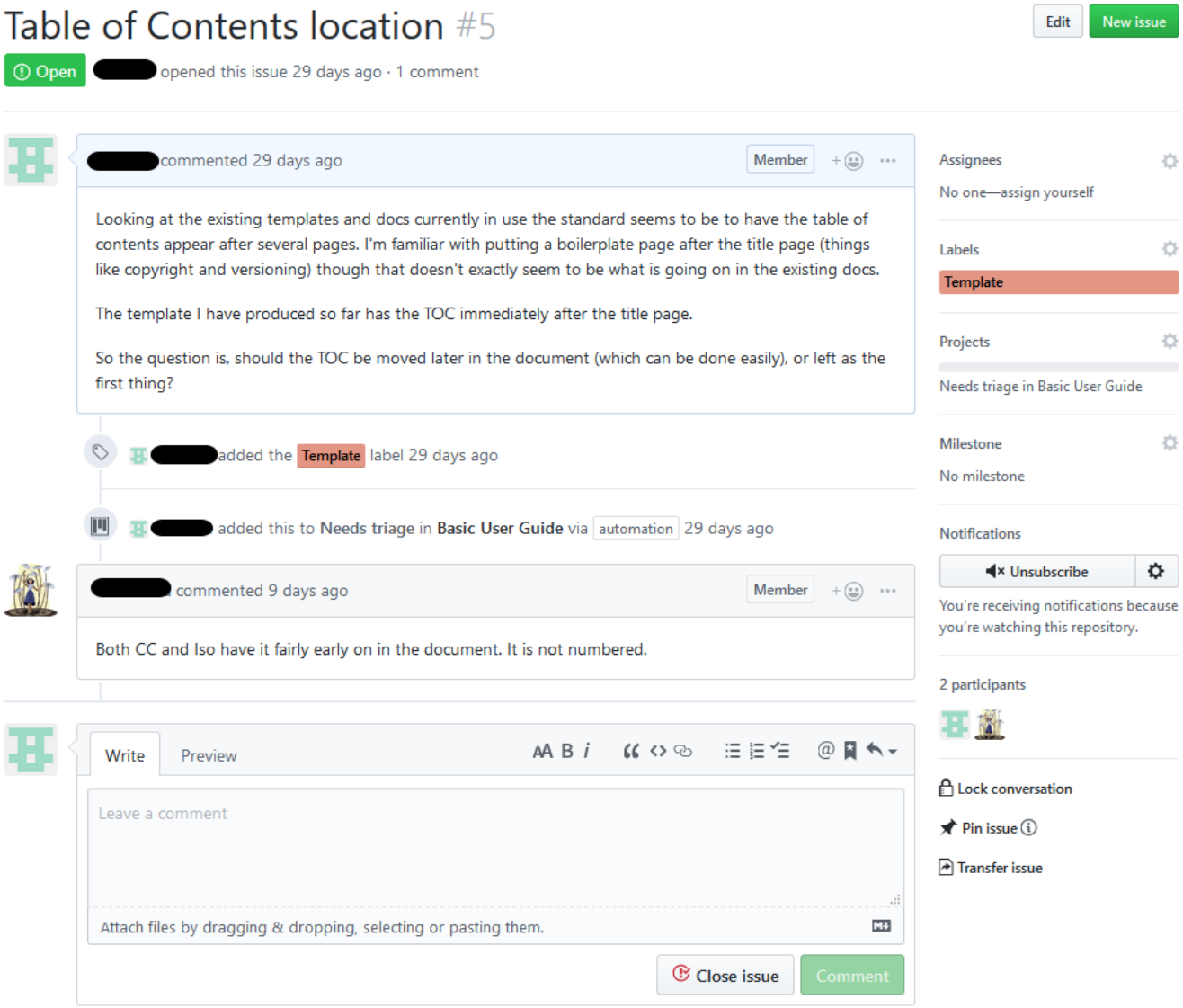


Figure 8. GitHub Issue View

Working with Issues will be described in the section Using Issues.

2.2.3. Pull Requests Overview

The Pull Requests area is the editing review section. From here you can see edits that have been made to documents that are waiting to be accepted and merged into the current branch.

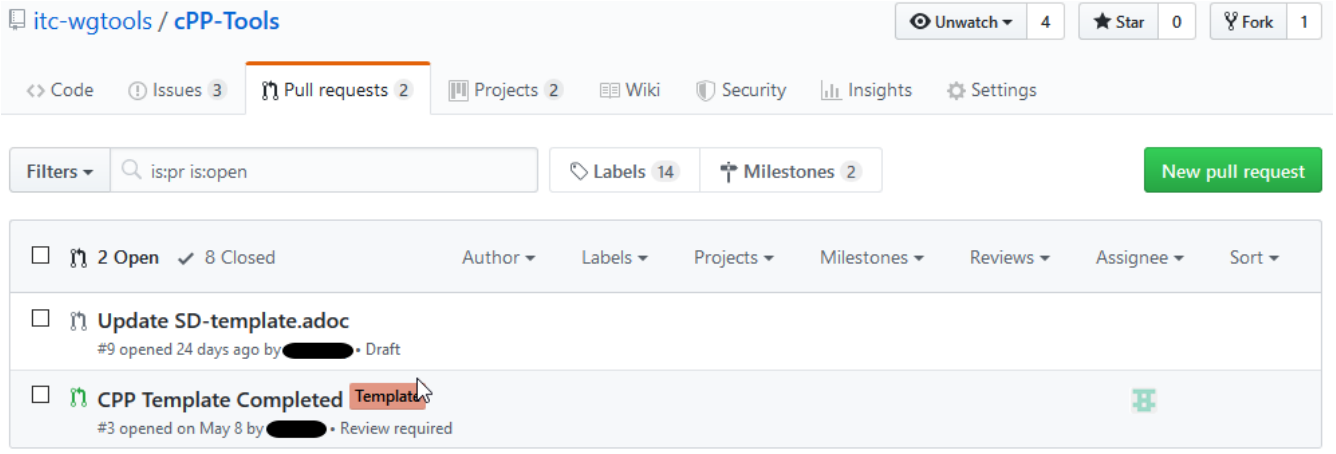


Figure 9. GitHub Pull requests

Clicking on a Pull request title will open the Pull request, showing the conversation about the Pull request as well as links to the changes that have been suggested. The view is similar to the Issues view.

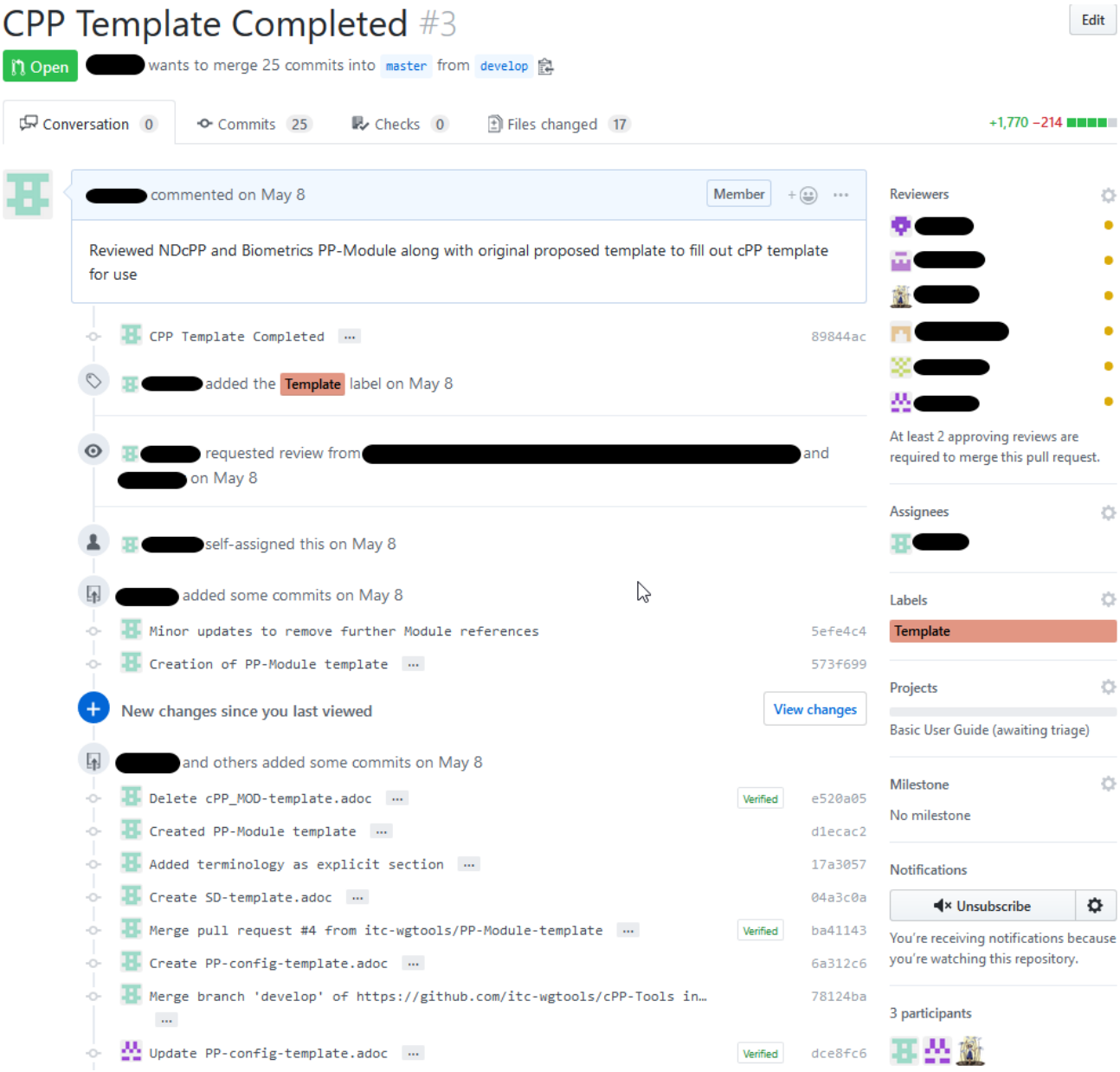


Figure 10. GitHub Pull request View

At the bottom of any Pull request you will see something like this.

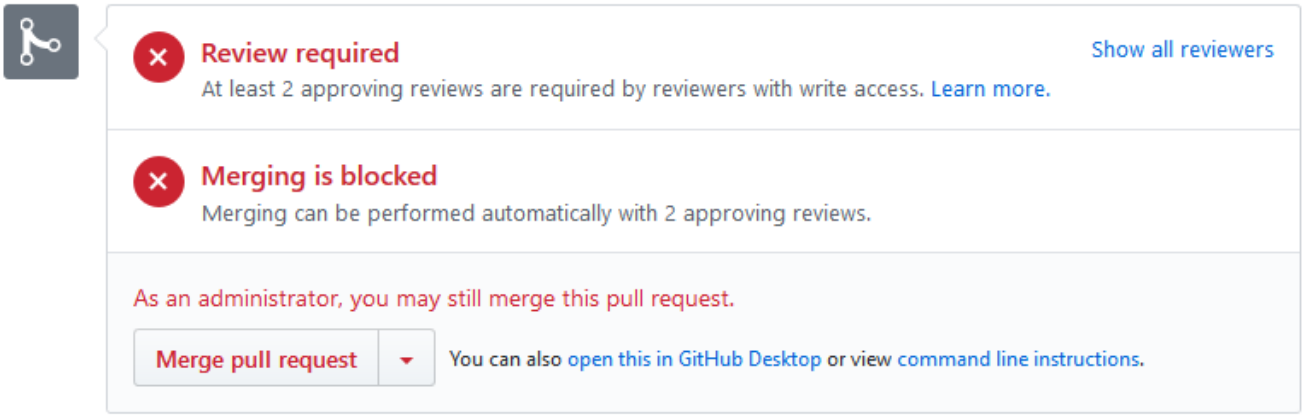


Figure 11. GitHub Pull requests

It may show different information (such as reviews have occurred and be green), but this shows the status of reviews on the Pull request and whether it is ready to be merged.

Merging is the process of accepting the proposed edit and making it part of the main working document (i.e. making it part of the branch).

Working with Pull requests will be described in the section [Using Pull requests].

2.2.4. Wiki Overview

The wiki is what you would expect, a wiki. You can create and edit pages here. This is useful for tracking things like meeting agenda/minutes and other useful information for everyone (like overviews of progress, direction, etc).

Live everything else in GitHub, every page change is fully tracked including who made the edits and when.

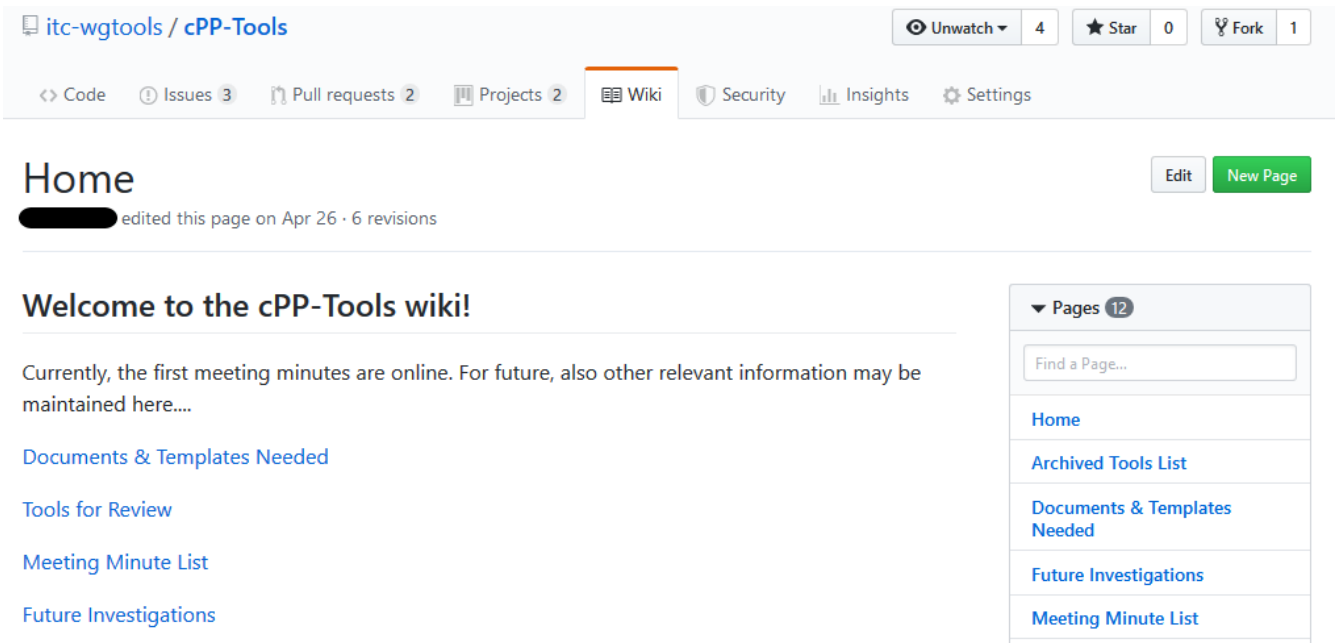


Figure 12. GitHub Wiki

As with any wiki, page content can be created to cover any topics that are needed.

3. Writing in GitHub

3.1. Using Markdown/Asciidoctor

When using GitHub, all the comments and documentation edits you make are in plain text. As noted in Changes for GitHub the documentation is all intended to be written in using the Asciidoctor syntax. But comments (or the wiki) in GitHub uses its own implementation of Markdown. These are similar but not quite the same.

For more information specifically about how to use the Asciidoctor syntax, review the document ***Asciidoctor for iTC Syntax Reference*** provided by the CCUF Team Tools WG. This document specifically provides examples of the syntax that is expected to be needed in the iTC documentation.

When editing comments or wiki entries though, the GitHub markdown needs to be used. The easiest way to use this is by using the highlighted icons at the top of the editor.

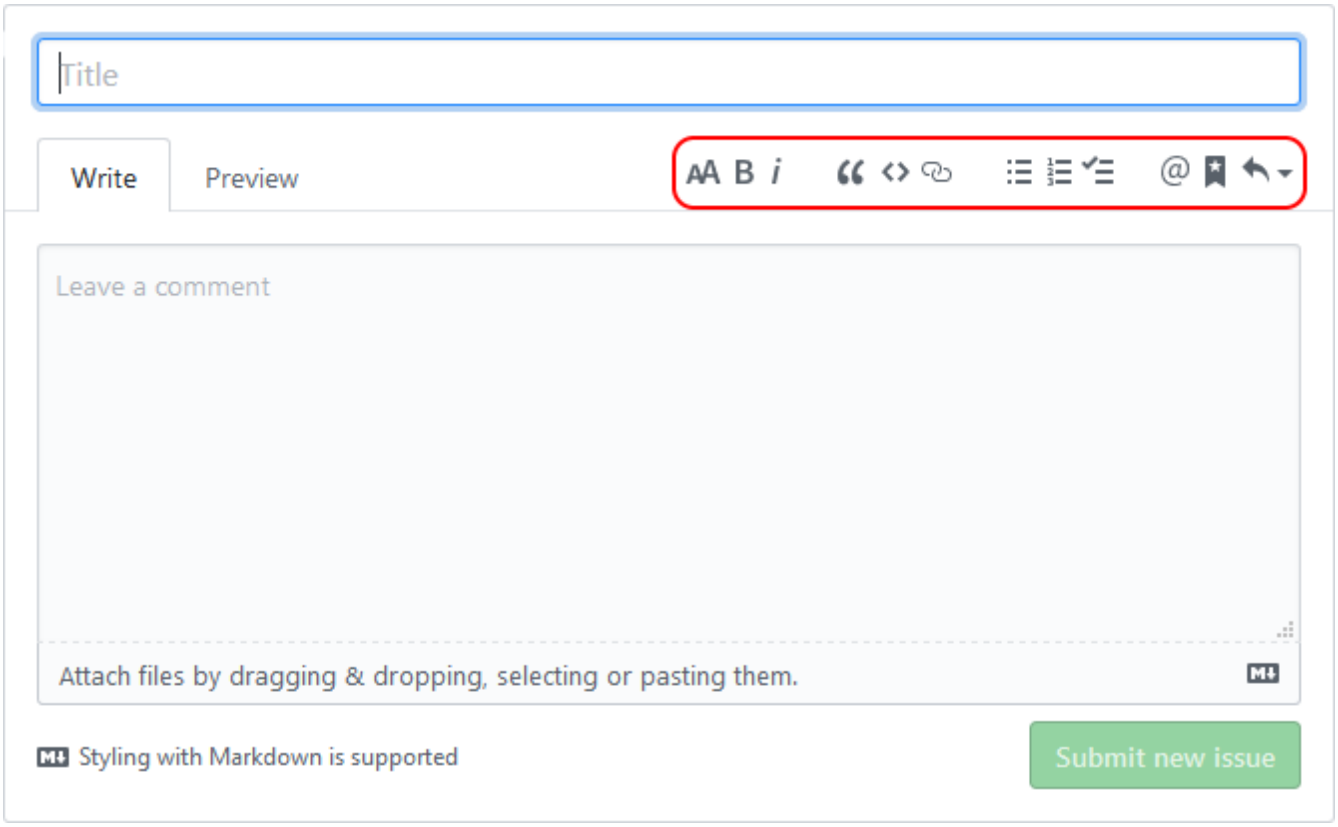


Figure 13. GitHub Markdown

These icons let you adjust the size, set bullets, make quotes, etc. These will automatically insert the proper markdown symbols for you. To see what the output will look like, click the Preview tab and the text will be rendered.

You will use the same comment box for all the text entry, whether for a comment or when editing a Pull request. The specific syntax you use, Asciidoctor or GitHub markdown is completely dependent on what you are doing.

Do not worry about making a mistake about which syntax to use though, as GitHub makes it easy to edit and make changes.

More information about GitHub markdown can be found here.

3.2. Internal link references

One of the more powerful features of the Issue and Pull request system is the ability to cross-link between related items. This is accomplished by starting with the number sign #. This will then bring up a menu of all the open Issues and Pull requests in the repository to select from. If you happen to know the number of the item you are trying to reference, you can start typing the number to narrow the choices (and if you just type the entire number the result is the same).

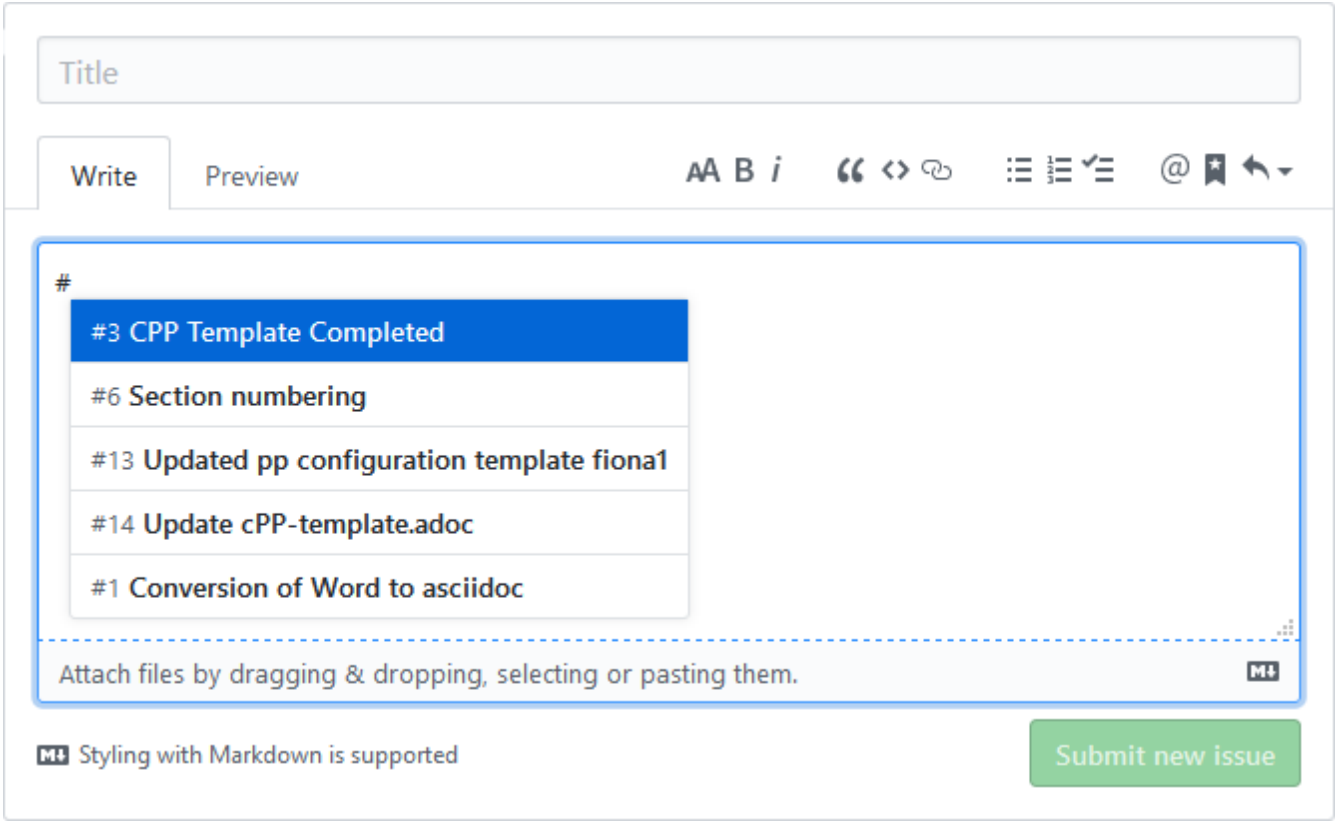


Figure 14. GitHub Internal Linking

This will automatically create a hyperlink to the other item in the text.

In the item that is referenced, there will be an added note to the conversation (which is linked to the referencing item).

Figure 15. GitHub Internal Cross Reference

3.3. Referencing a Person

In addition to being able to cross-link to other items, you may want to reference a specific person in a comment. This can be done using the @ symbol. When typing @ you will see a list of people (by their username) in the repository (or you can type the username if you know it).

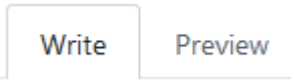
Referencing a person this way does two things. The first is it allows you to direct your comments to someone (such as replying to something said earlier when multiple are contributing). The second is that it specifically notifies that person they have been mentioned in the item so they know to check.

4. Using Issues

The Issues area is one of the two areas where you will probably spend most of your time in GitHub. As noted before, this is where conversations about your iTC will happen. In many cases, eventually this will lead to a Pull request, but the point of Issues is to talk about different aspects of the iTC work.

4.1. Reviewing Issues

Reviewing Issues is similar to commenting in any forum application. At the bottom of the Issue thread there will be a dialog box showing two tabs:



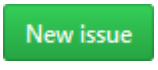
Any comments you want to make should be entered in the dialog box. Clicking the

button will add your contribution.

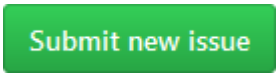
4.2. Creating New Issues

While reviewing existing issues is important, creating new Issues is a common task.

To create a new Issue, click the



button, provide a title and your description. Once you have entered your Issue, click the



button to create the Issue.

If you start to create an Issue and then move off to something else (another page) and then come back to create a new Issue, the previously entered content will still be shown in the window. This is a feature of the website.

4.2.1. Additional Fields

When creating (or reviewing) and Issue (or Pull request), there are several other fields that can be assigned. These fields can help assign specific people to review the Issue (they will get a notification about being assigned) as well as providing fields that can be used to filter the Issue.

Each of these fields can be configuring using the gear icon.

The Projects field may be used by the administrator but is not covered here.

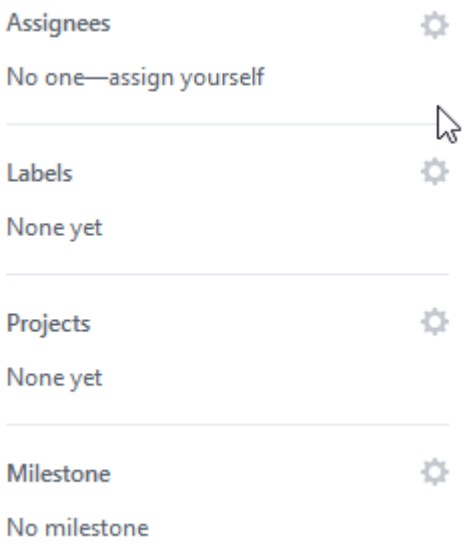


Figure 16. GitHub Additional Fields

These additional fields can be changed or assigned at any time, so submitting without them does not cause any problems, but as always, providing more information is better.

4.2.1.1. Assignees

This field allows you to assign other iTC members to review your Issue (or Pull request). There is no limit to the number that can be assigned though they must be selected individually.

4.2.1.2. Labels

The Labels field allows you to specify categories for the Issue (or Pull request). The specific Labels will be created by the administrator, but can be anything.

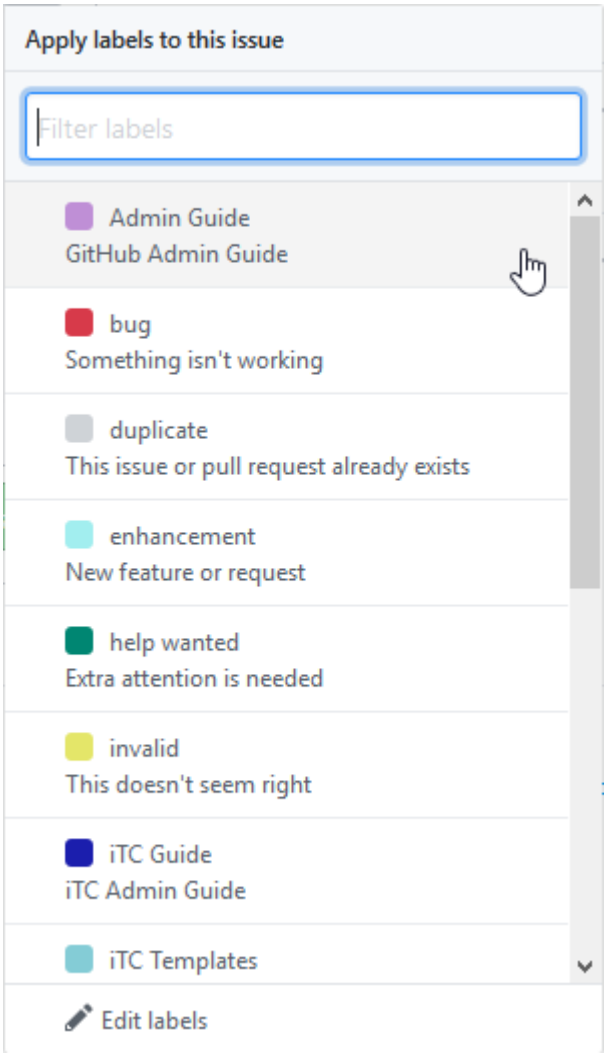


Figure 17. GitHub Labeling

As you can see in the example there are labels for specific topic areas as well as generic topics like bug or enhancement. If there are labels that will help categorize your Issue (or Pull requst) for others, you should select them from the available list. There is no limit to the number of Labels that can be assigned.

4.2.1.3. Milestone

The Milestone field allows you to specify a release target. Generally this would be some date for release, but may also be internal timelines for completion. If Milestones are being used, an appropriate Milestone should be selected.

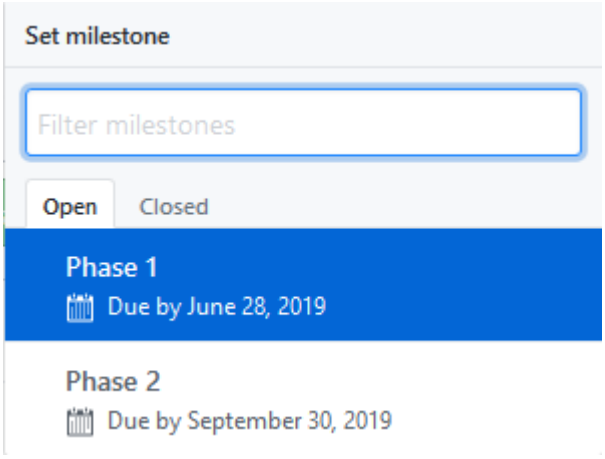


Figure 18. GitHub Milestones

Only one Milestone may be selected.

4.3. Link to Specific Document Locations in Issues

One of the most important type of links that can be created, especially in an Issue, is a permalink. A permalink is a direct reference to a location within a file and marks the location permanently (so it will be tracked to that location regardless of the changes that may occur over time. This is like highlighting text and adding a comment in a Word document. The benefit of a permalink is that it will be linked to the specific location in the document even as the document changes over time (so a year old link in an Issue will still point to the proper location in the document that may have changed many times).

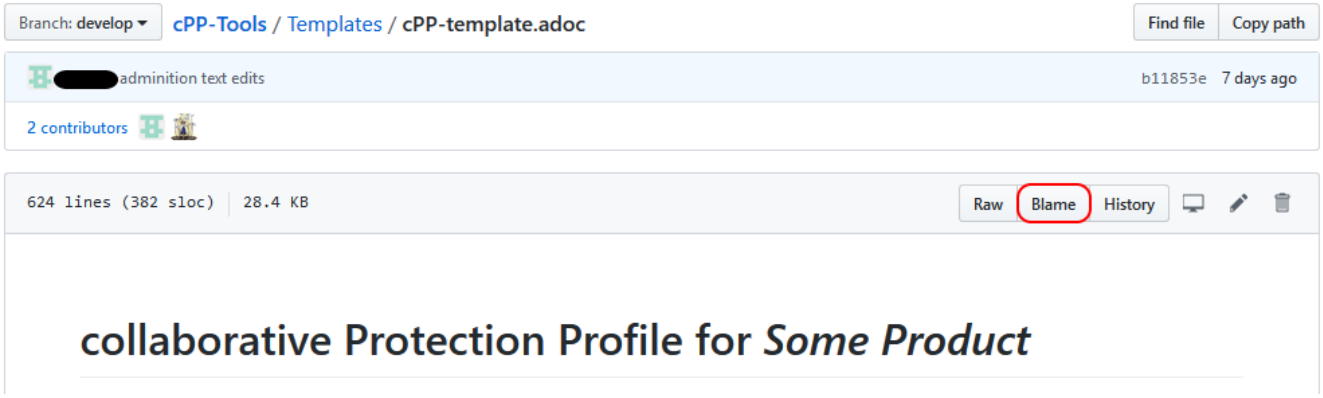
By using permalinks in your Issues, the reader can always find the correct location you are talking about.

When referencing a specific location within a document, you should always add a permalink to the line.

Because of the types of documents being used, the following is the process for adding a permalink.

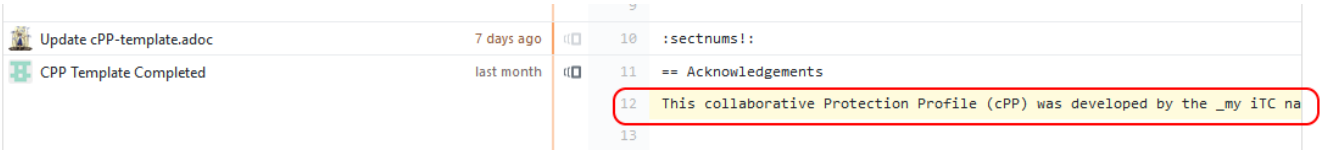
Open a second tab in the browser (so you can have the file and the Comment open at the same time).

1. In the Code area select the file you are making a comment on.
2. Click the Blame button



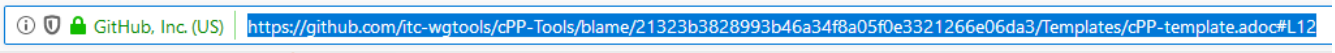
Permalink - Open File to Blame

3. Press the "y" key on your keyboard (this will change the URL to ensure you get the proper link)
4. Click the line number you are referencing (highlighted in yellow)



Permalink - Click the Line Number

5. Select the URL that is shown. It should end with **#Lxx** where **xx** is the line number you selected.



Permalink - Copy the URL

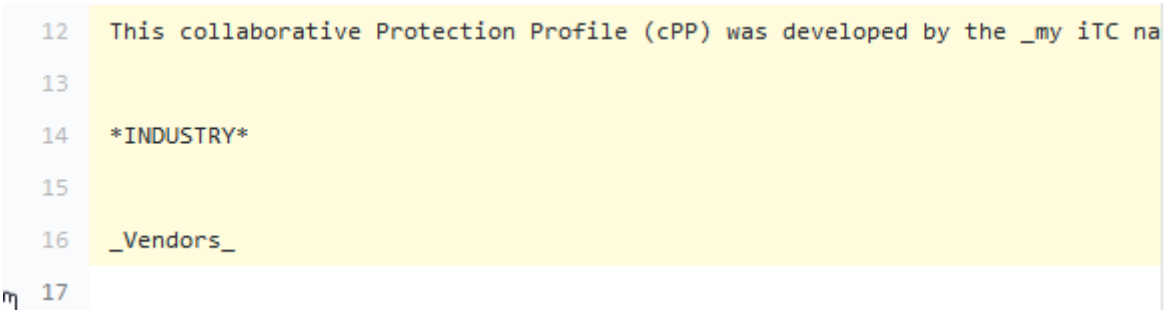
6. Paste the URL into your comment and add your comment.

4.3.1. Multi-line Permalinks

When a comment involves multiple lines, it is possible to link directly to the multiple lines as well, and not just picking one.

This can be done two ways (replace the above steps with these):

4. After selecting the line number, hold the Shift key and click the end line number



Permalink - Click Multiple Lines

Or this way:

6. After pasting the URL into the Comment, add **-Lxx** to the end of the line where **xx** is the last

line.

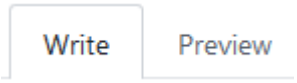
For a multi-line selection, the end of the URL should look like **#L12-L16** to select lines 12-16 in the document.

5. Using Pull Requests

The Pull requests area is where you will make suggested edits to the documents the iTC is working on. In addition to edits, Pull requests provide the ability to comment on the suggested changes in the same way as an Issue, allowing for discussions directly related to the changes to be housed in the same place.

5.1. Reviewing Pull Request Conversation

Reviewing Pull requests is similar to commenting in any forum application. At the bottom of the Pull request thread there will be a dialog box showing two tabs:



Any comments you want to make should be entered in the dialog box. Clicking the

button will add your contribution.

In addition to seeing comments, you will also see a list of all the changes that have been made in this Pull request. This can be small or large, depending on what the contributor has edited. See the figure GitHub Pull request View for an example of the additional information that is displayed.

5.2. Reviewing Pull Request Changes

When someone has made changes and created a Pull request, you can view them before they have been committed to the branch. This lets you comment on the changes or propose your own.

To view the changes, you should to look at either the Commits or the Files changed views.

Figure 19. GitHub Pull Request Files Changed

It is possible to view the files from the Conversation display, since it shows both comments and commits to the Pull request.

To view an individual change, click on the 6 character string (circled in the figure below). This string is a portion of the checksum that is calculated on the change and how GitHub tracks each change individually.

Figure 20. GitHub Pull Request Conversation Commits

5.2.1. Commits View of a Pull Request

In the Commits view you will see all the commits to the Pull request. Commits are the individual updates that have been made over time. For example the author of the Pull request may have made an initial change, and then someone else suggested a second change. Each of these individual changes are tracked by GitHub.

To view an individual change, click on the 6 character string (circled in the figure below).

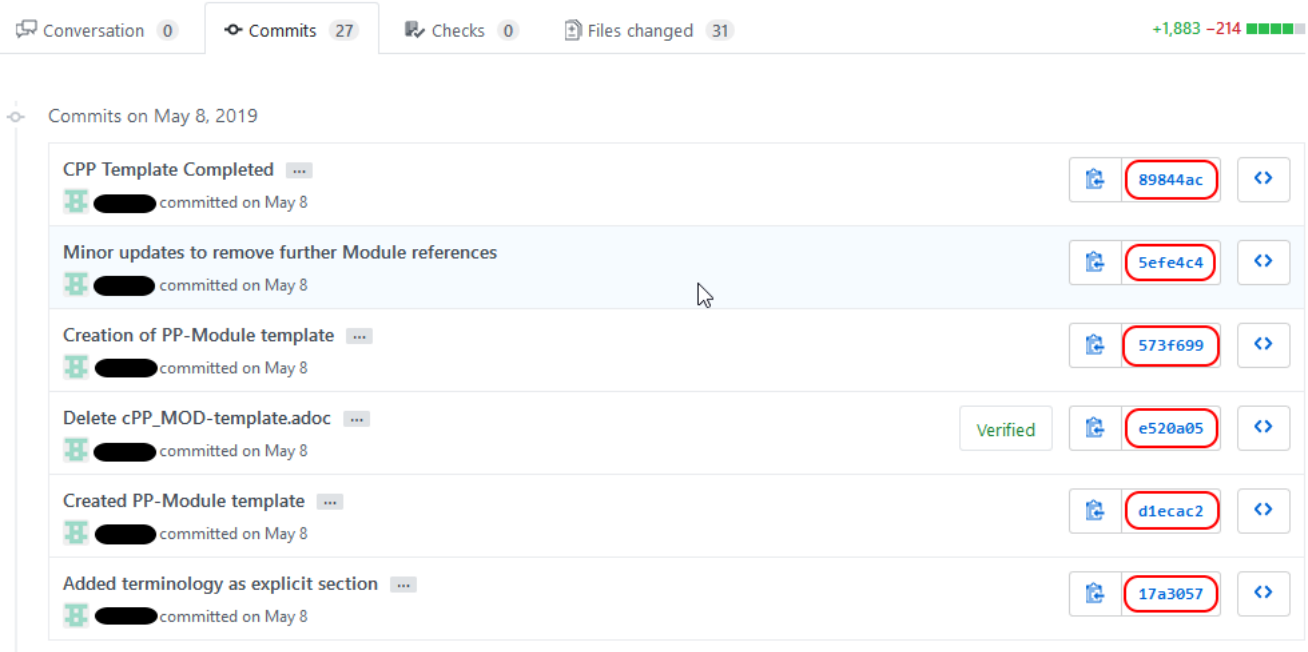


Figure 21. GitHub Pull Request Commits View

5.2.2. Files Changed View of a Pull Request

In the Files changed view you will see a list of all the files in the Pull request and all the changes in each of the files.

5.3. Pull Request Changes Display

Whether you access an individual commit via the Commits view or from the Files changed view, you will see the same basic display of changes.

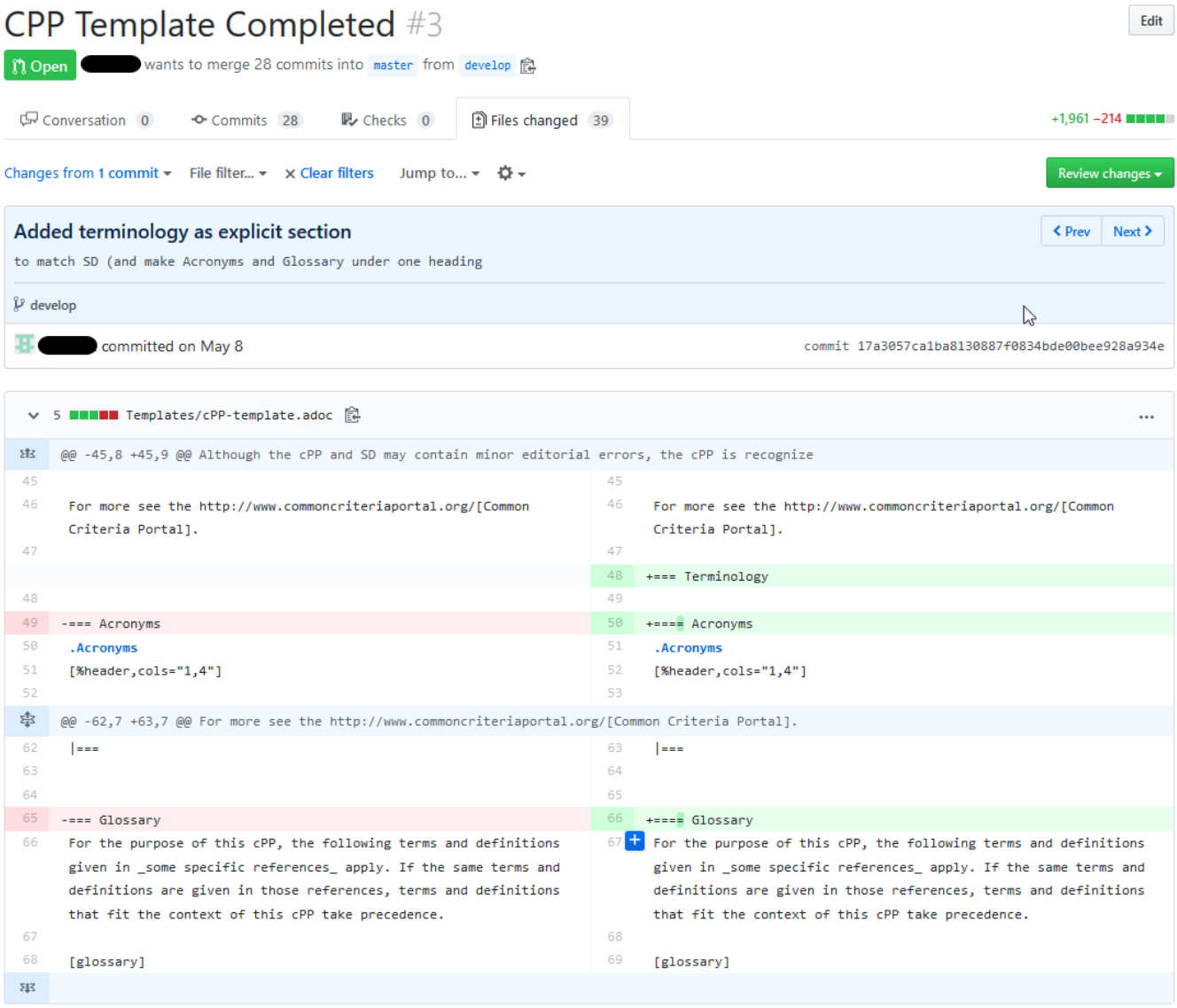


Figure 22. GitHub Pull Request Changes

On the left side of the display is the original file and the right contains the result proposed by the Pull request.

On the left you see lines with a "-" and highlighted in red. These are things from the original that are removed (or possibly just edited). On the right you see lines with a "+" and highlighted in green. These are things from the Pull request that are added. Note that in the case of line 49 that the original shows the line as deleted and the Pull request shows it added, but with the fourth "=" in darker green. This means that the change is actually that additional "=" (this is repeated on line 65). Also note how the Pull request lines are off in numbering due to the addition of "=== Terminology" on line 48, yet the rest of the file remains in sync.

By reviewing the changes side-by-side you can easily see how the Pull request will update the document.

5.4. Adding Comments Directly into Pull Requests

Sometimes, instead of commenting in general, you may prefer to enter a comment directly where a change is being requested (or where you would like to see a change). The comment is similar to any other comment in a Pull request or Issue, but instead of being shown within the full discussion it

will be seen inline to the document.

While displaying the changes, place the cursor over the line where you want to make the comment. As you move the cursor over each line, a blue + should show up next to the line number.

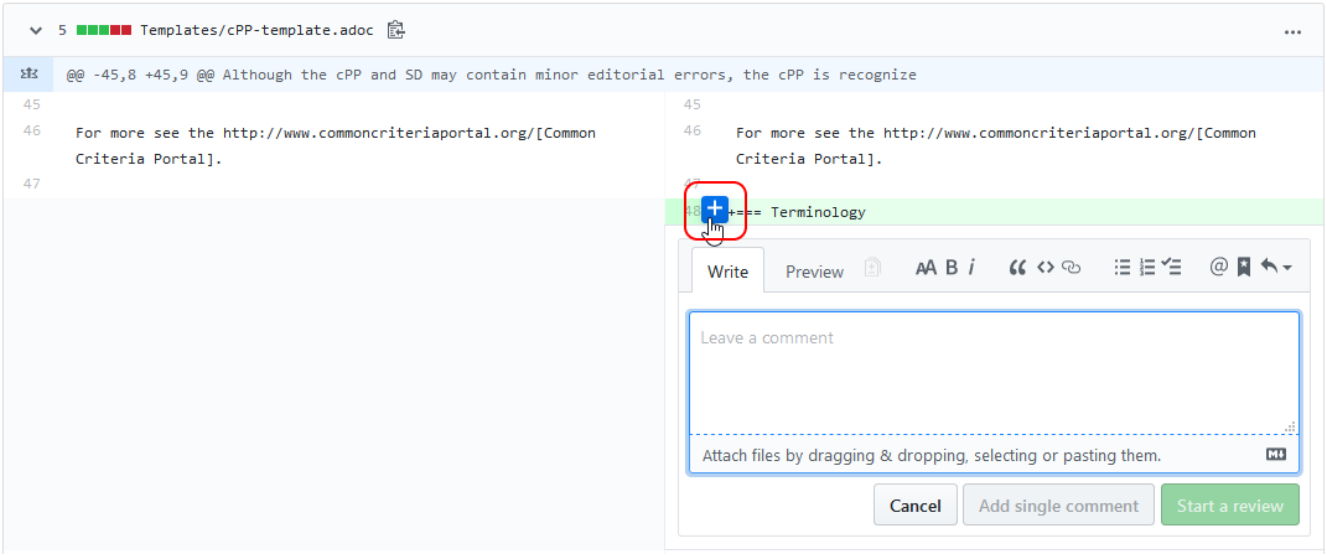


Figure 23. GitHub Pull Request Direct Comment

Clicking the + will open the comment dialog.

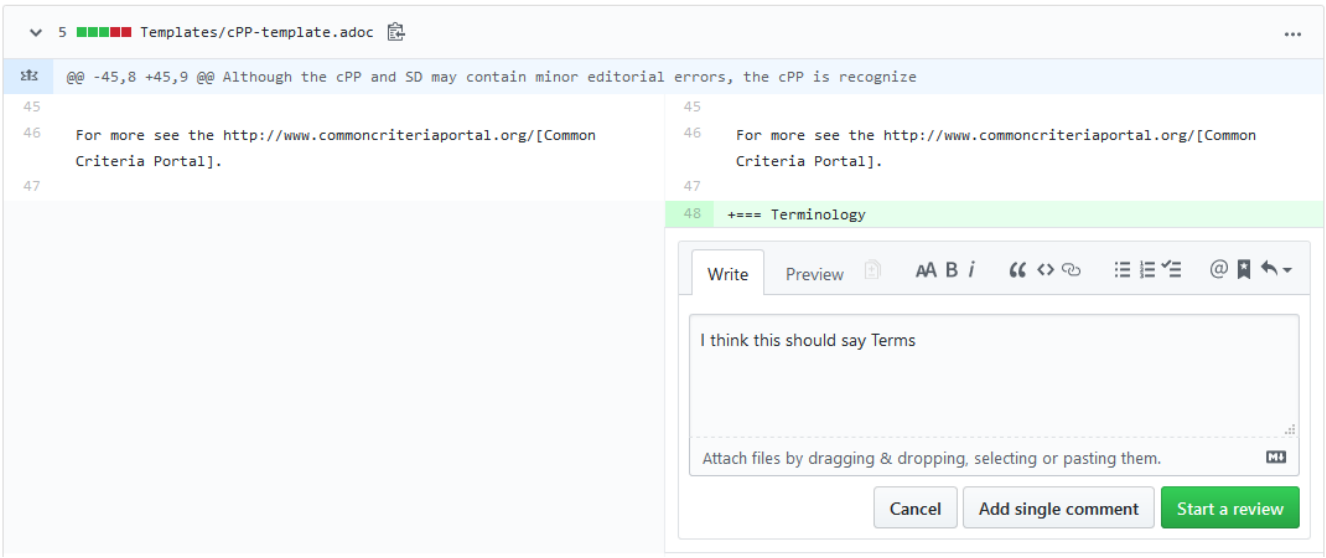
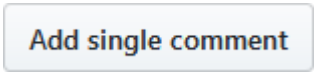
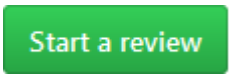


Figure 24. GitHub Pull Request Add single comment

To just add the comment, click:



The



button will be covered in Approving Pull Requests.

5.4.1. Resolving Comments in a Pull Request

When a comment is created directly in a Pull request, it will appear in the file view.

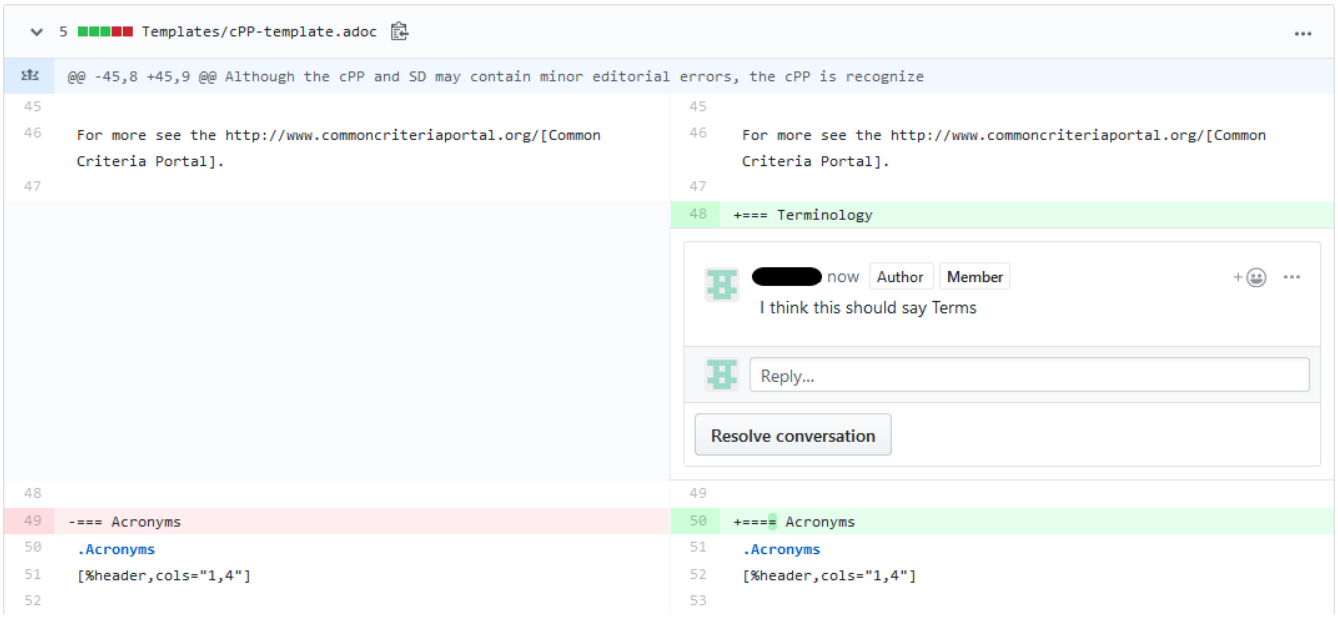
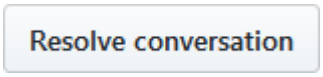


Figure 25. GitHub Pull Request Resolving a Comment

To close a comment you should reply and then click:



It is expected that any reply would have a reason for closing the comment (such as a Pull request to make an edit or a reason to not change anything).

5.5. Creating a New Pull Request

There are many ways to create a new Pull request. The instructions here is the simplest flow, especially for single changes.

If you need to make large changes to complicated documents, it may be best to perform the edits offline (this is covered in the iTC Advanced User Guide for GitHub).

The first step in creating a Pull request is to select the file you need to edit. Generally you will do this from the Code view to select the file from the repository.

The default branch is likely the develop branch (this is the recommended configuration). If you need to work in a different, check Changing the Branch to switch.

When you have opened a file in the Code view, you will see this bar at the top of the content.

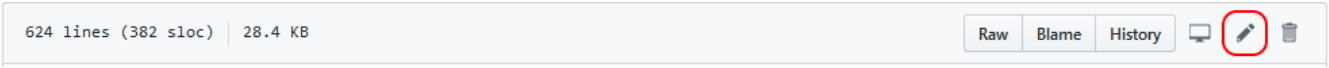


Figure 26. GitHub File Edit Bar

Click the pencil to edit the file. The window that opens is the editor for GitHub.

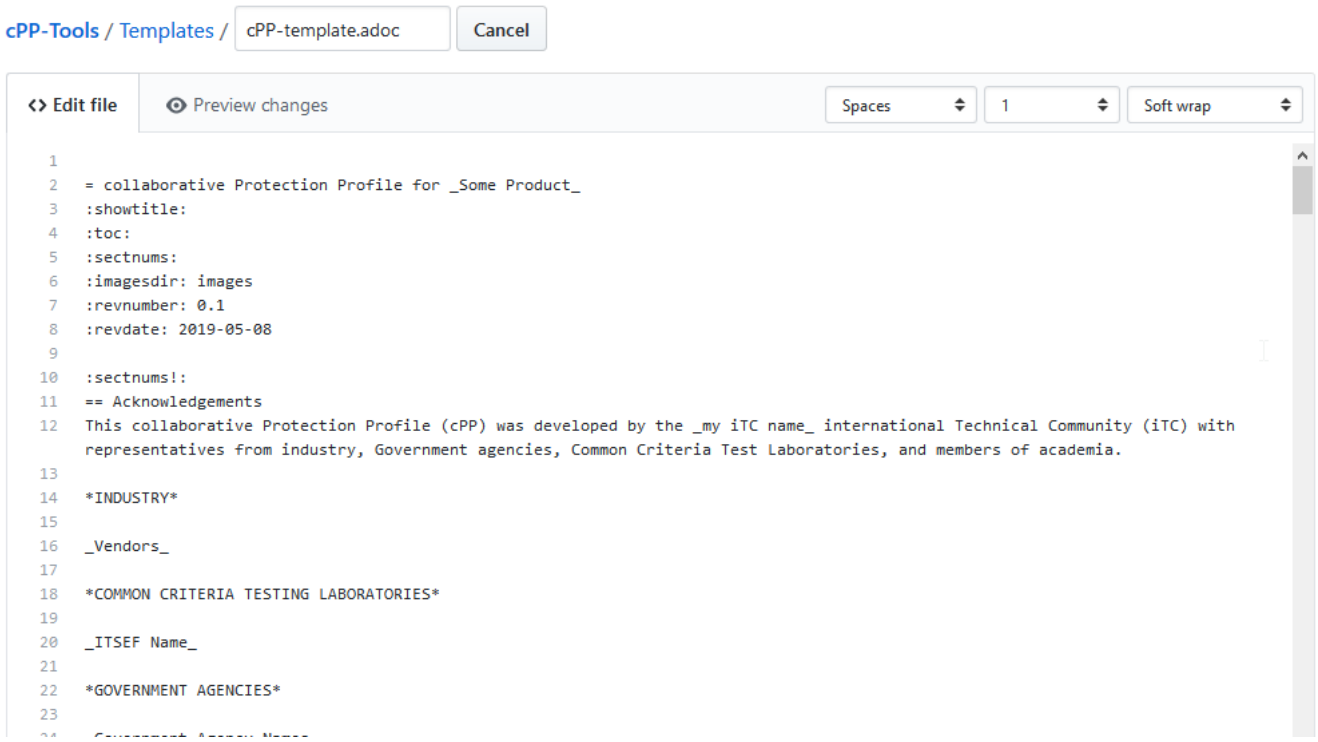


Figure 27. GitHub Editor

You can make any changes you need to at this point. It is also possible to search for content using **Ctrl-F** (if you have clicked inside the editor).

Once you have made your changes, you need to save them to a new branch. The new branch will form the basis of your Pull request.

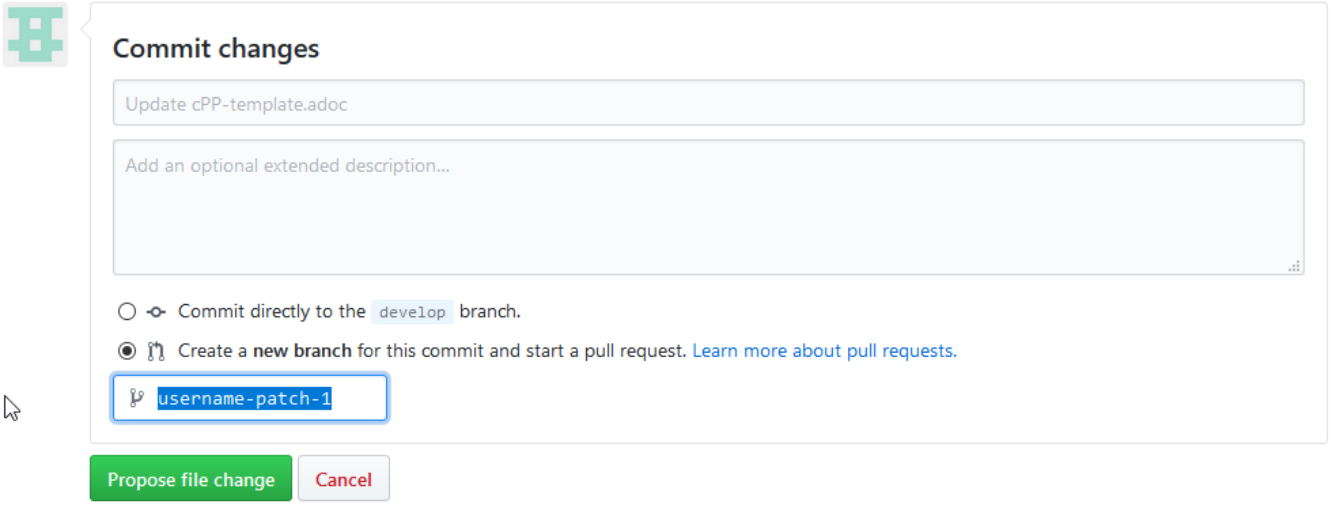


Figure 28. GitHub Create Branch

In the dialog box you should add a title (what is the point of the changes) and a description about them (maybe why the changes are being proposed).

GitHub will automatically propose a branch name (using your username and then "-patch-X" where

X is a number if there are other patches from the same person). You can rename this if you wish, but it does not matter.

Once the editing has been completed and you have added a description, click Propose file change.

This then brings you to the Open a pull request page.

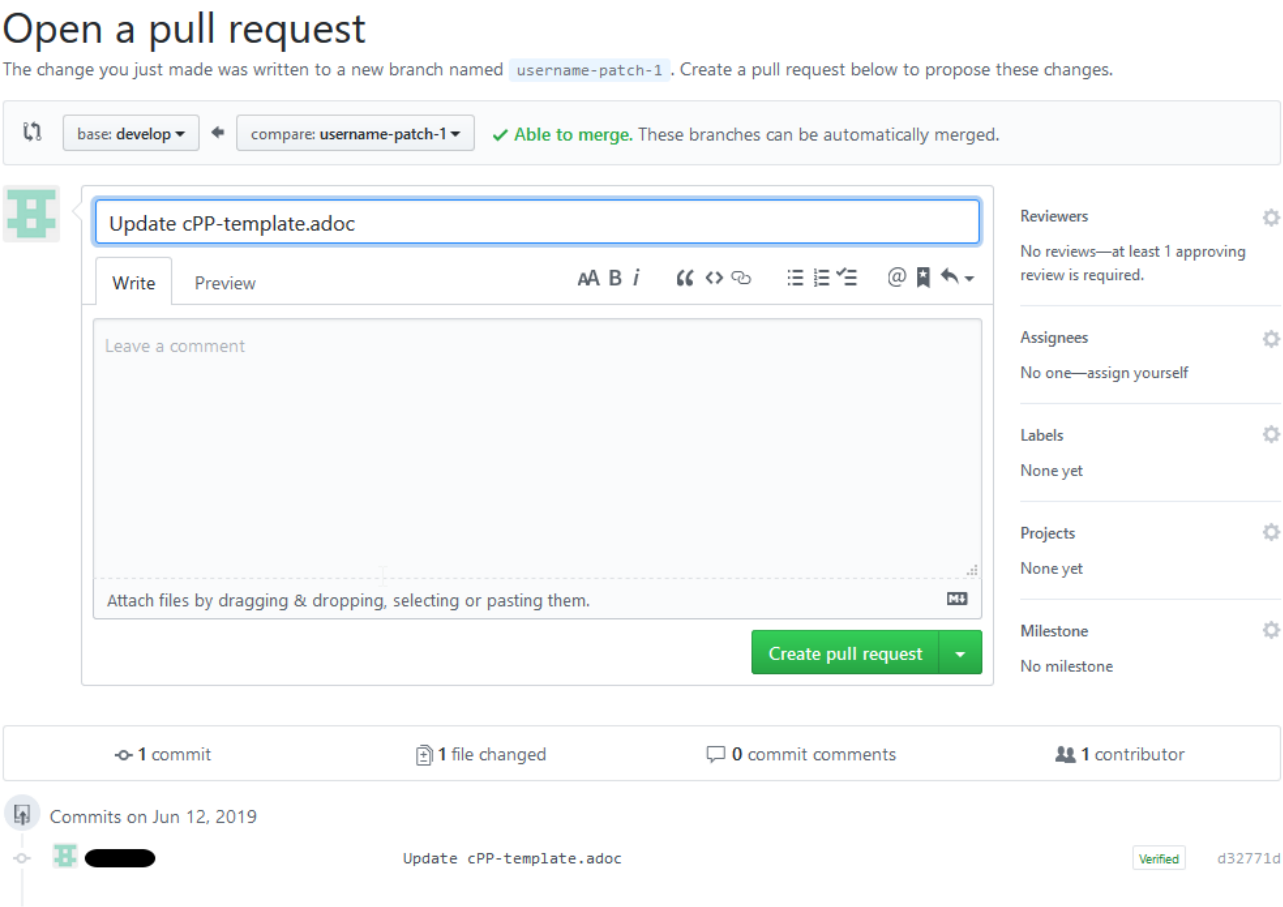


Figure 29. GitHub Open a Pull Request

The title and comments you provided on the editing page will be copied here. You can also add Additional Fields here, including requesting specific people to review your Pull request, by selecting "Reviewers" for the Pull request.

To create the Pull request, click:

While Pull requests can encompass very large changes, in many cases it is best for them to be small (or at least a single topic). This doesn't mean making individual spelling changes into individual Pull requests, but massive changes all over a document can be difficult for reviewers to track and fully accept.

By keeping changes to either a small number of edits or to a single topic of edits (say a change to an SFR and all its follow-on changes to other SFRs and related text), then the requests are more easily digestible for review and approval.

5.6. Working with Draft Pull Requests

While working on a Pull request you may need to stop and save your work so you can complete it later.

5.6.1. Creating a Draft Pull Request

To create a draft Pull request, follow the steps to create a Pull request, but on the Open a pull request page, click the dropdown on the

Once the Draft Pull Request has been selected, the button will change to show the draft status.

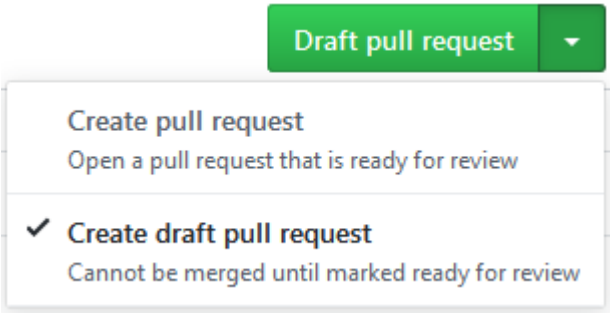


Figure 30. GitHub Create a Draft Pull Request

When the Pull request is created it will be marked as a draft.

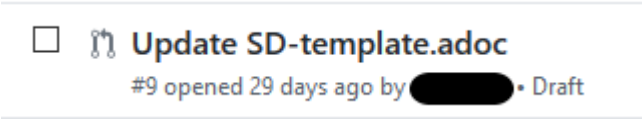


Figure 31. GitHub Draft Pull Request View

5.6.2. Continuing Work on a Draft Pull Request

A draft Pull request is a holding place for your commit, and you can edit it in the same manner as any other Pull request. The key difference is that only the owner can actually review and edit a draft Pull request.

To continue editing your Pull request, you can follow the steps in Reviewing Pull Request Changes. While you should always pick the last commit to start from, when you go to edit the file all the commits in this branch will show up, so it doesn't have a real impact as to which commit you use for your draft.

When you are done editing, you should see the option to commit the changes directly to your branch. This will allow you to continue adding to your draft Pull request. This will create a second commit (or third, etc.) to the Pull request.

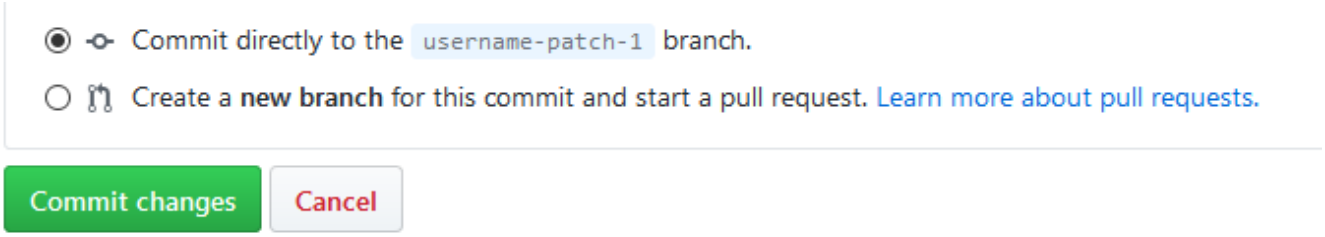


Figure 32. GitHub Draft Pull Request Commit

5.6.3. Publishing a Draft Pull Request

Once you have completed your edits and are ready to publish the draft Pull request, open the Pull request and on the Conversation view you should see **This pull request is still a work in progress**.

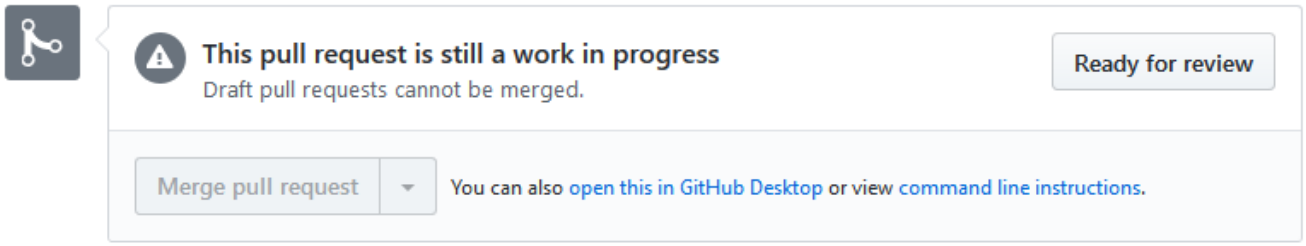
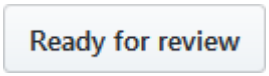


Figure 33. GitHub Draft Pull Request Ready for Review

To make the Pull request public click:



The Pull request will now be visible to everyone and can be merged into the document.

5.7. Approving Pull Requests

As with any document with multiple editors, at some point there needs to be an agreement about what to put in the document. GitHub provides the ability to require approvals of Pull requests before they can be accepted (merged) into the working (or final) document.

While specific people may be requested to approve the Pull request, any member of the repository is able to approve a Pull request. The person submitting the Pull request can specifically ask people to review it using the [Additional fields].

When people are assigned to review a Pull request they are given a notification of the request. This can be seen when the user views the Pull request.

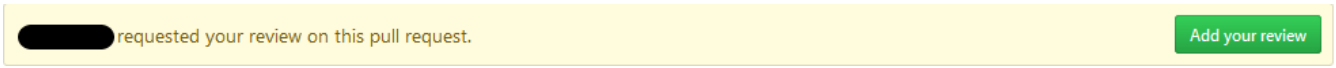
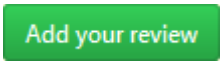


Figure 34. GitHub Pull Request Review Notification

To open the Review Changes dialog click:



Alternately, you can see this by going to the Files changed view and clicking:

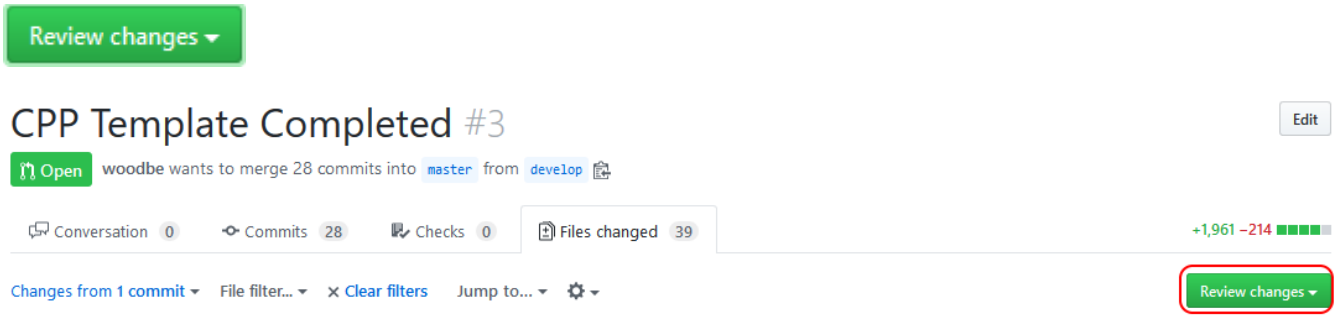


Figure 35. GitHub Pull Request Files changed Review

The result of either of these buttons is the Review dialog.

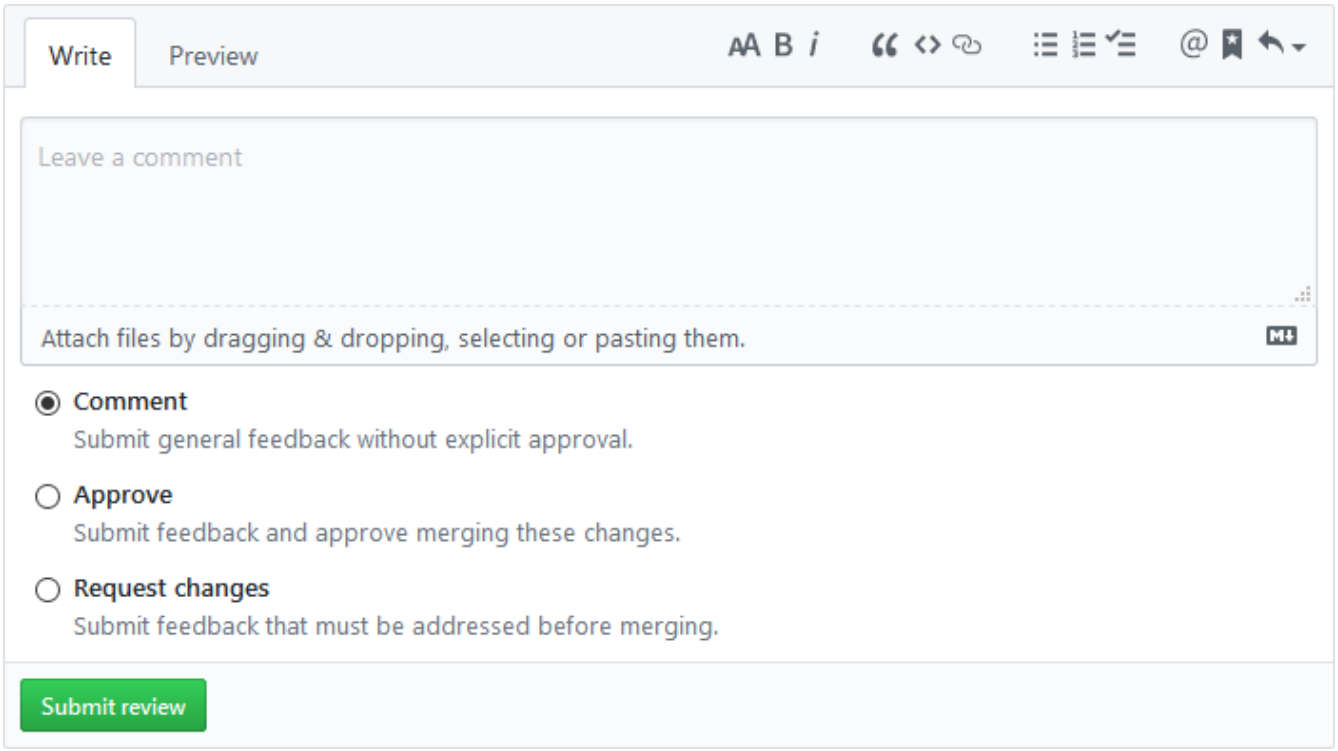


Figure 36. GitHub Pull Request Review

There is a dialog box to enter your thoughts/comments on the Pull request and then three options:

Comment

Just a comment on the Pull request, added to the conversation.

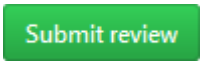
Approve

Your approval to merge this Pull request into the branch.

Request changes

This is basically a non-approval with a specific request to change something. To be useful the expected change should be specified in the dialog box. A change request must be resolved or rejected by an administrator before the Pull request can be merged.

Once you have made your comments and selected an approval status, click:



6. File History

As noted before, GitHub keeps track of every individual change (unless the documents are binary, like images, in which it just tracks different files as a whole). These can be seen at any time in the history of any file.

To access the file history, open the file from the Code view. From there, look for the History button.

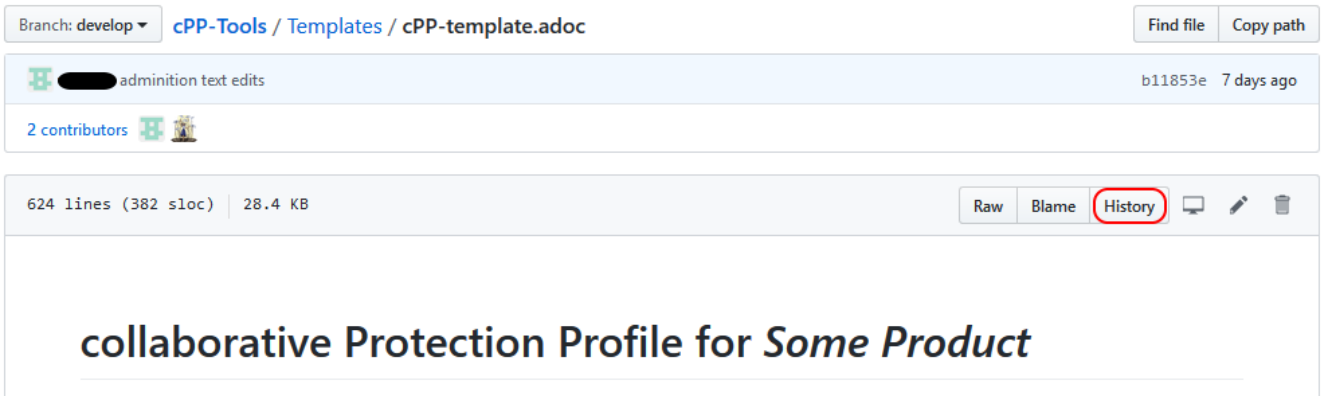


Figure 37. GitHub View File History

This will open the full list of changes that have been made to the file.

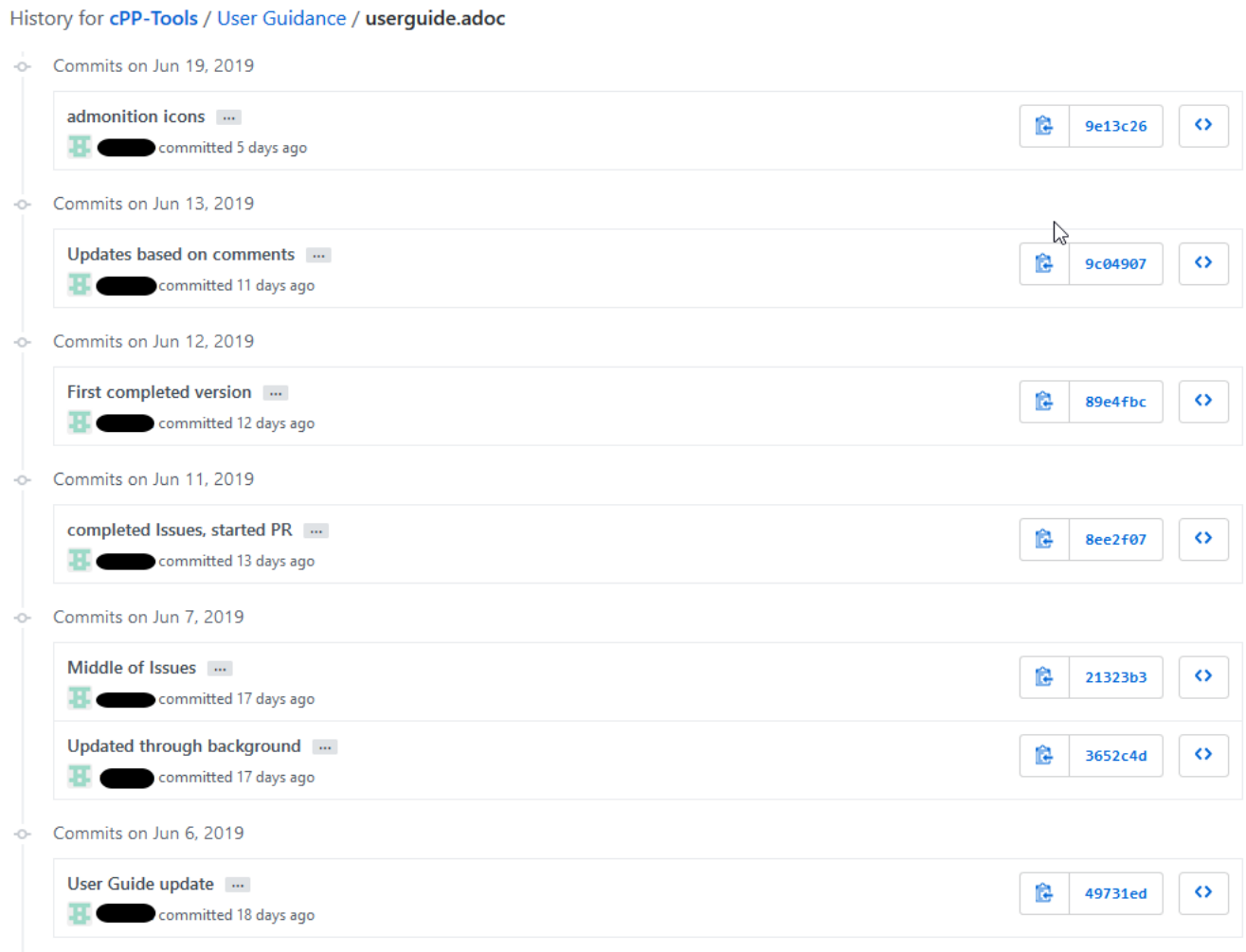


Figure 38. GitHub View File History List

The changes here can be viewed in the same manner as Reviewing Pull Request Changes, by clicking on any of the 6 character strings as shown on the right, to see each change that has been made to the file.

In addition, clicking:

will let you browse the repository (i.e. the Code view) at the time that change was made to the file. This can be helpful in providing reference to the changes (i.e. current status of any other documents at the time the change was made).