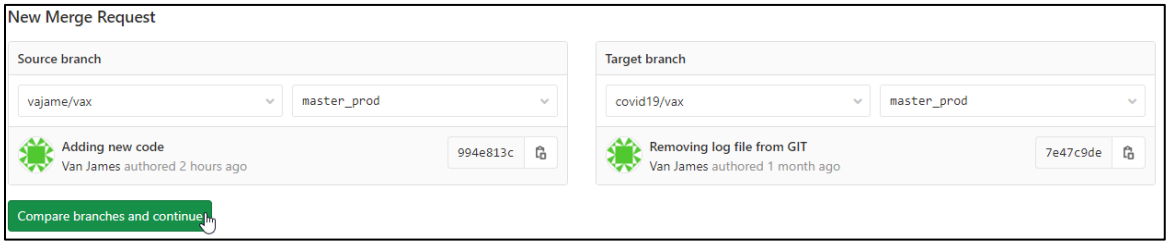
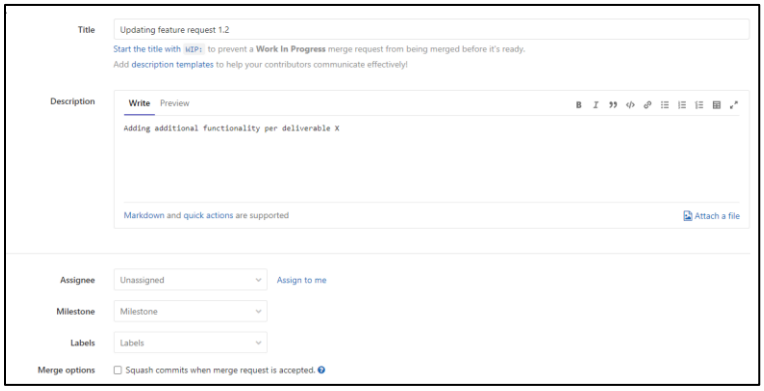
4. Identify the branch in the repository that should be merged back to the master branch, as depicted in Figure 14.

Figure 14: Map Personal Repository to Main Repository



5. Document the changes, to provide context to the maintainers of the master branch (Figure 15). Make sure to fill in a descriptive **Title** and document the changes in the **Description** box. A good description makes it easier for the maintainers to peer review the changes. In the **Description**, be sure to include information on the changes that were made and why, what changes were attempted (but were not fruitful), and the feature or bug fix that the update is addressing.

Figure 15: Dialog Form for Merge Request



6. Assign the merge to a maintainer for peer review and merge approval (Figure 16), then click the **Submit merge request** button (Figure 17).

Figure 16: Assignment Selection for Merge Request

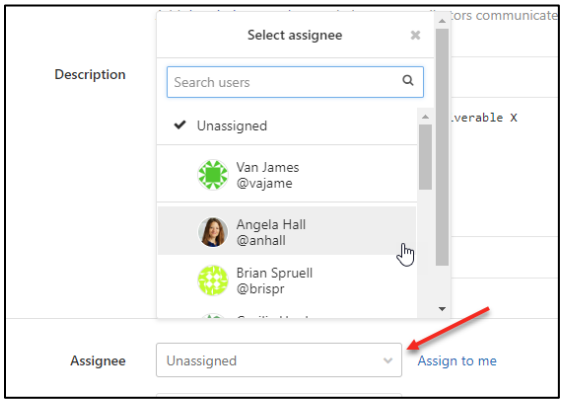


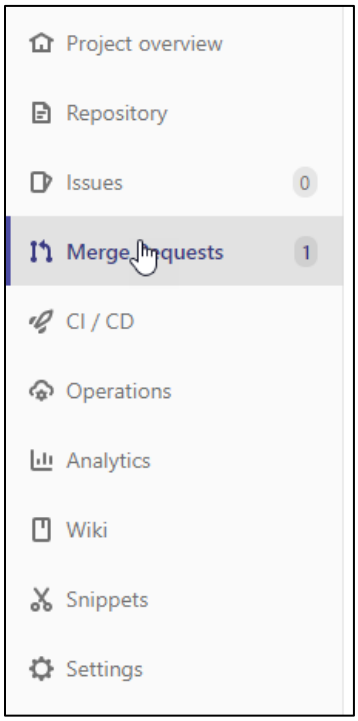
Figure 17: Merge Request Submission Button

4.7 Merge Request - Maintainer Perspective

As the maintainer of a repository, you may be called on to peer review code and approve merging updated code back into the master branch. To identify pending merge requests, use the following steps:

1. Use your web browser to navigate to the repository location.
2. On the left panel, the **Merge Request** field indicates whether you have any merge requests waiting for review (Figure 18).

Figure 18: Merge Request Awaiting Action

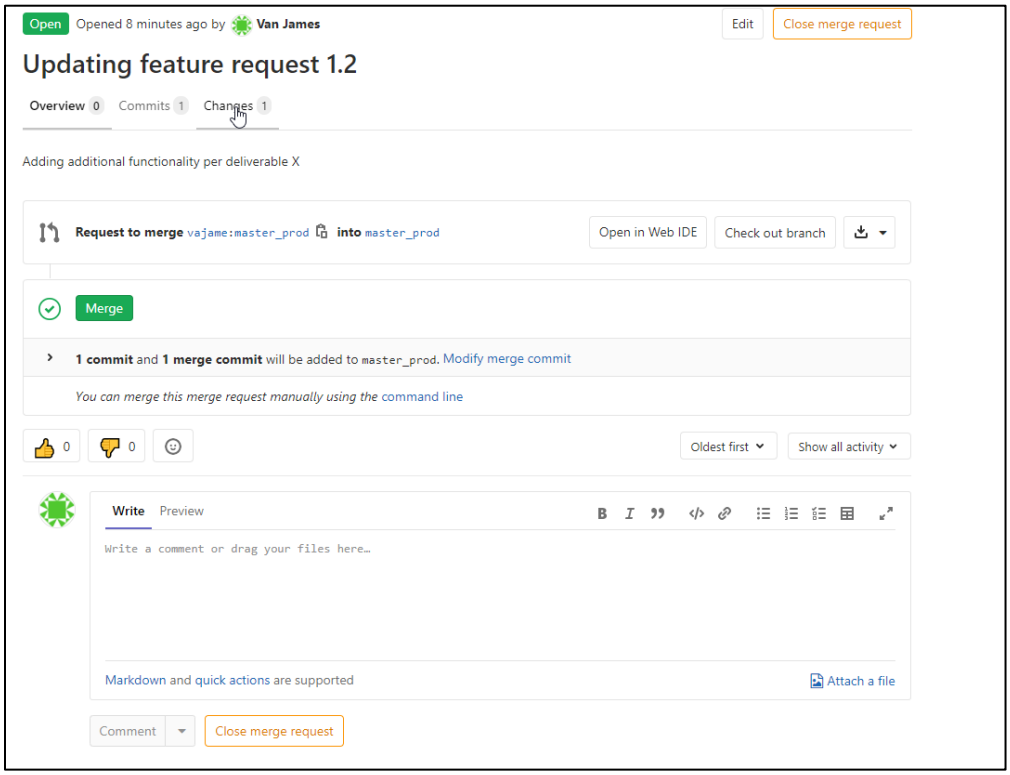


3. If a merge request needs to be reviewed, first select the **Merge Request** field on the left navigation panel, then select the merge that you wish to review. In the example shown in Figure 19, you are reviewing the update to feature request 1.2, as submitted by Van James.

Figure 19: Select the Desired Merge Request to Review

4. After you select the merge request for review, a dialog form opens in which you can communicate with the developer and review the changes to the code. From this form, you can review the changes by selecting the **Changes** tab (Figure 20).

Figure 20: Merge Request Dialog Form



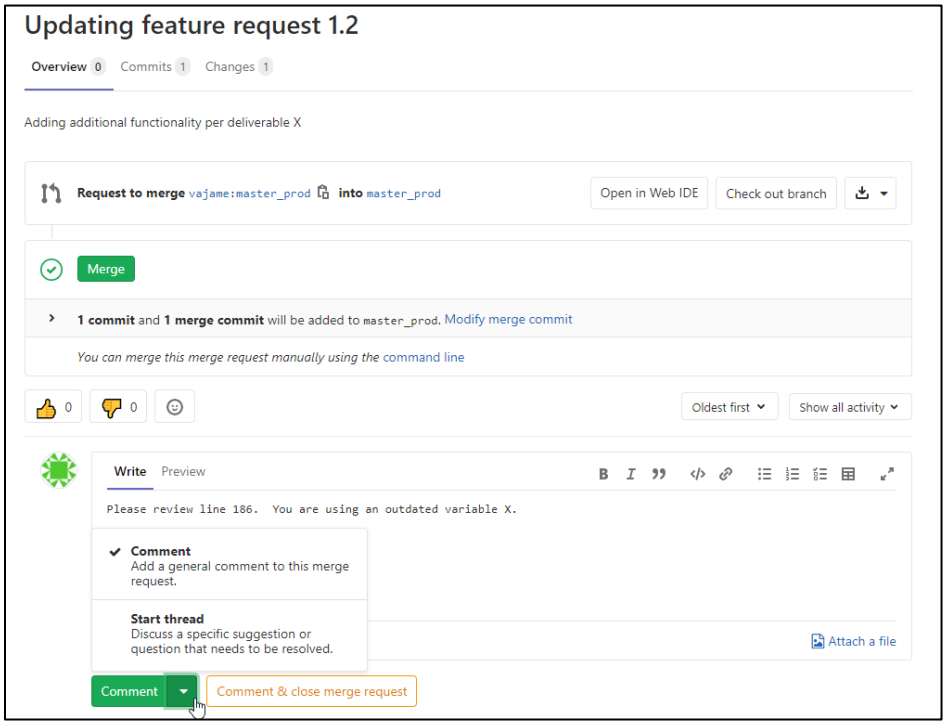
5. Selecting the **Changes** tab displays a form that identifies all changes that have been made and need review (Figure 21). Changes to be added to the code base are highlighted in green, while changes to be removed from the code base are highlighted in red.

   Note: It is necessary to review the code to ensure that any new code in green is sound in its logic and will not introduce new bugs into the main code base, in addition to reviewing the code highlighted in red to ensure that its removal will not adversely impact downstream processes.

Figure 21: Merge Request Code Review Panel

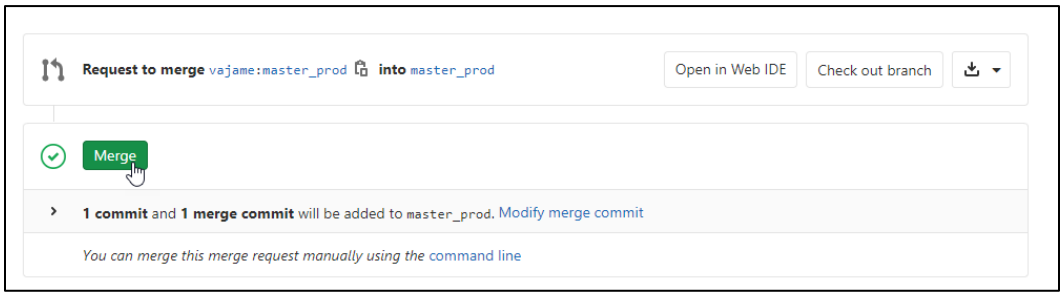
6. If questions arise during your code review, use the **Overview** panel to open a dialog with the developer (Figure 22). You can add comments to the merge request to document considerations that were taken when reviewing the merge. You can also open a thread with the developer to inquire about specific code elements.

Figure 22: Merge Request Overview Panel



7. After the code passes review, merge it into the main code base by selecting the **Merge** button (Figure 23).

Figure 23: Merge Request Acceptance Button



5 Common Use Cases

This section contains example workflows for some of the most common tasks encountered in application development by USPS resources. It is impossible to cover all possible scenarios that may arise, but the following scenarios provide a foundation that can be built upon to address most of the challenges that occur during the development life cycle.

5.1 New Code Development

New code should be developed in the master branch (Figure 24). There is no need to create additional branching for new code.

Example Workflow

To set up your playpen, follow the instructions in Section 3.4.

Always perform a git pull before developing new code or updating existing code to ensure you have the most recent version of the code.

Example Workflow

git pull

Develop new code …

git add <new code>

git commit -m "<description>" <new code>

git push <new code>

Figure 24: New Code Workflow

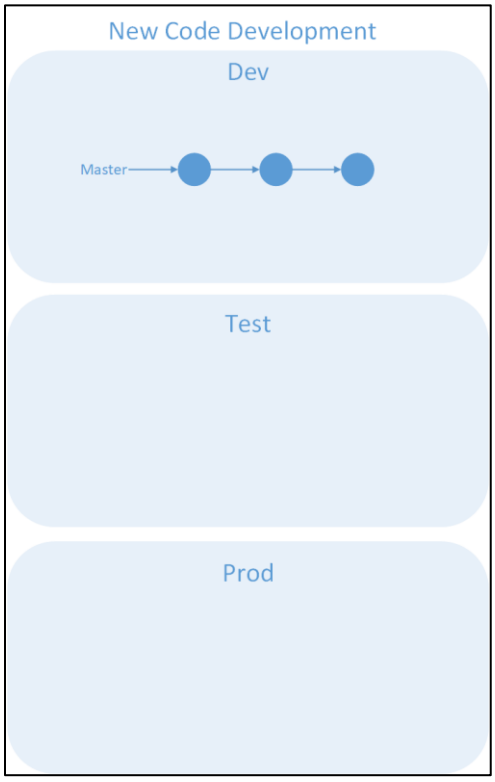
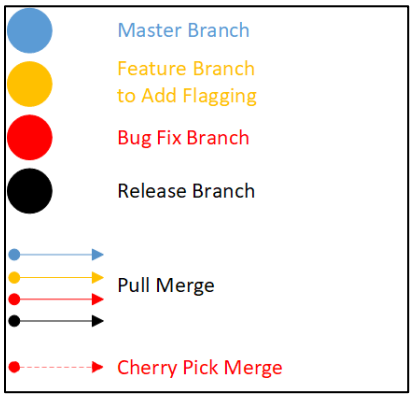


Figure 25: Quick Reference Legend



5.2 Bug Fix

To fix a bug in the master branch, first create a branch off of the master (Figure 26). Within that branch, update the code to fix the bug, then create a merge request to merge the bug fix back into the master branch.

Example Workflow

git pull

Create branch

git checkout -b <branch Name>

Update code

git commit -m "<description>" <new code>

Push new branch to the remote repository

git push -u origin <branch name>

Switch back to the master branch

git checkout master

Merge the bug fix back into the master repository.

git merge <branch name>

Push the changes up to the remote repository

git push