**ECF Next UI Code Review Process**

# **Overview**

The purpose of this document is to provide some guidance in doing code review for the ECF Next UI team and to help with code quality and eliminating obvious bugs (hopefully). Some of the bugs that are reported by QA are related to styling, minor issues and most may not be related to functional requirements.

The UI team intended purpose is not to eliminate defects found by QA – this can never happen; but, limit the number of obvious defects reported by QA.

Please feel free to modify or add comments and critique where concerned or strongly objected to some of the methodologies and practices provided in this document. Or, we could have another meeting, which is fine by me.

# **Code Review Process**

## Code Reviewee Tasks (**All underlined sections below are Hyperlinks**)

* Create a pull request – See:
  + [Working with Branches](#_Working_with_Branches)
  + [Preparing for Pull request](#_Preparing_for_Pull), and;
  + [Creating A pull request](#_Creating_A_pull)

## Code Reviewer Tasks

* [Initial Review](#_Initial_review_of) – done by Dale Anderson, John Holcomb or Christine Rudolph
* [Code Quality Review](#_Tasks_Performed_for) – done by Boaz Kallagunta and Tellek Liberty
* Approve and Merge reviewee’s branch to development branch – done by Boaz Kallagunta and Tellek Liberty

# Working with Branches

This section helps with expressing the importance of having a single branch with a PBI, defect or when there are many developers working on a PBI, creating a branch for specific tasks.

## Scenarios

### One Developer-One PBI/Defect-**Single Branch Per PBI/Defect**

**Overview:** this scenario consists of a single developer assigned to one PBI or Defect within a specific sprint.

Process:

* **Create a branch for the PBI or Defect using this naming convention.** This is the only difference between the two processes; the rest of the processes are the same. See table “Git and Git Flow scenarios compared”.
  + Branch Name: [application prefix name]-[pbi or defect]-[pbi or defect #]-[branch name]. Example of a branch name would be “ecf-defect-06300-tab-inconsistency” or “ecf-pbi-53944-person-case-list”. See sections “[Creating a Branch using Git](#_Creating_a_Branch)” and “Creating a Branch using Git Flow” for more details.
* **Publish the Branch to Azure Git Repository for ECF Next**. See sections “[Publishing branches using Git](#_Publishing_branches_using)” and “Publishing Branches using Git Flow”.
* **Git repo management**. See sections for “Git Repo Management in Visual Studio Code”.
* **Delete your local and remote branch**. See sections for “[Deleting local branch](#_Deleting_the_local)”, “[Deleting Remote Branch using Azure DevOps](#_Deleting_remote_branch)” and “Deleting local and remote branch using Git Flow”

## Many Developers-One PBI/Defect-**Single Branch Per Tasks**

Overview: This scenario occurs when you have multiple developers working on the same PBI or defect.

Process: The same process as above, but, the only difference is creating the branch. Create the branch using this naming convention – [application prefix name]-[pbi or defect]-[pbi or defect #]-[task name]-[branch name]. Example: “ecf-defect-06300-**fix-it**-tab-inconsistency” or “ecf-pbi-53944-**links-refactor**-person-case-list”. Follow same steps listed above.

# Preparing for Pull request

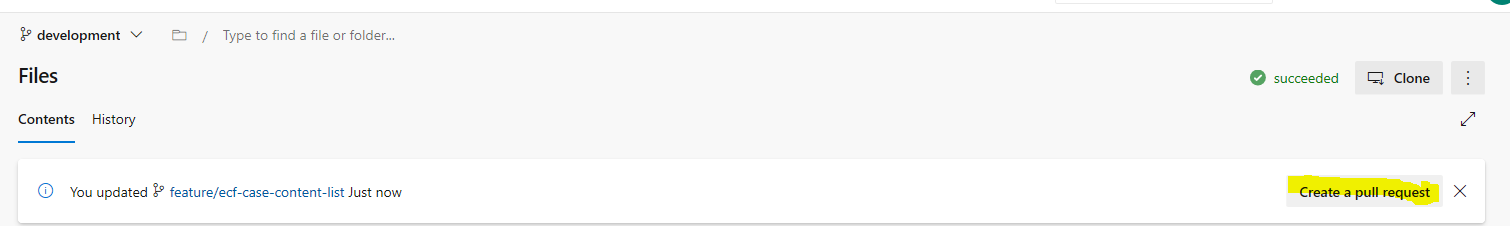
In order to create a pull request, you must have an existing branch. These are some recommendations for managing your branches. See section “[Working with Branches](#_Preparing_for_Pull)” to understand more details. These two prerequisites must be considered before creating a Pull Request.

1. Branch must not contain typescript or html lint errors
2. Branch should not contain compilation errors – may have runtime errors, though.
   1. Perform “npm run buildQa” – ensure that the branch you are going to merge to the development branch is compilable. It may contain runtime errors, but we must ensure that the code being merged should compile.

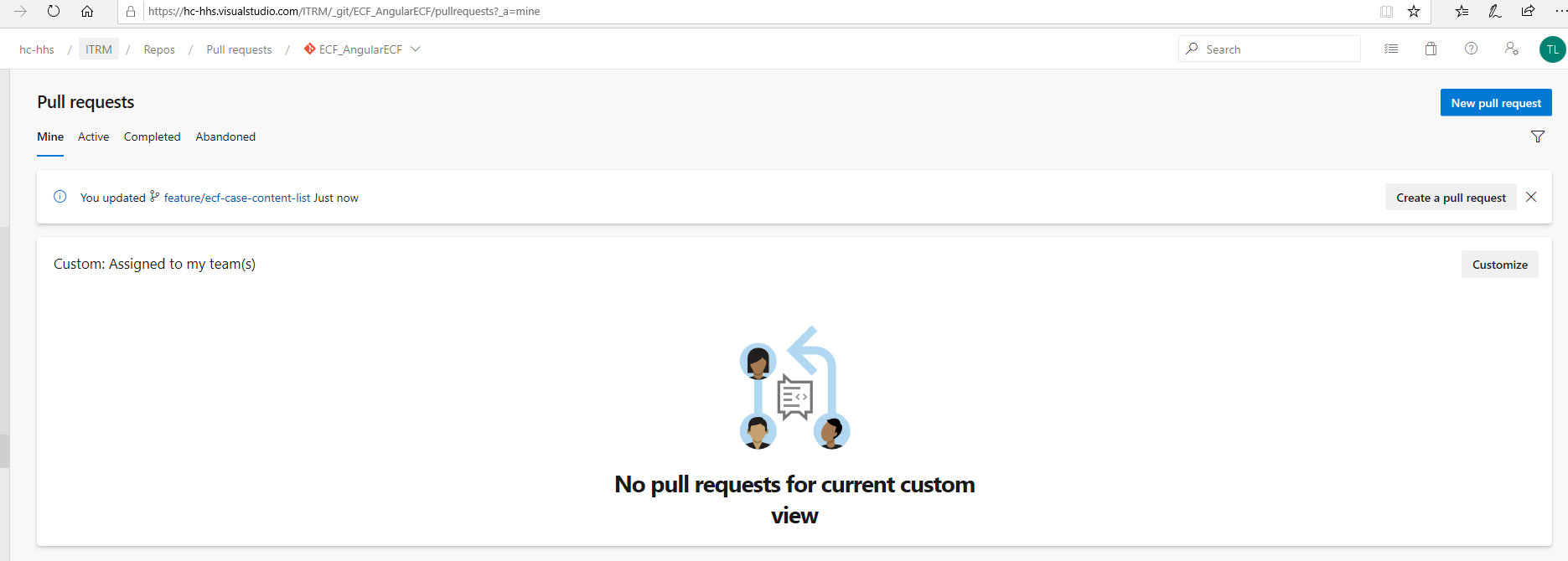
# Creating A pull request

The developer requesting a code review must create the pull request. In order to create a pull request, do the following after you have pushed your code from your local repository to the remote repository on Azure DevOps.

1. Navigate to ECF Next Azure DevOps and Click Create Pull Request using one of the following ways:

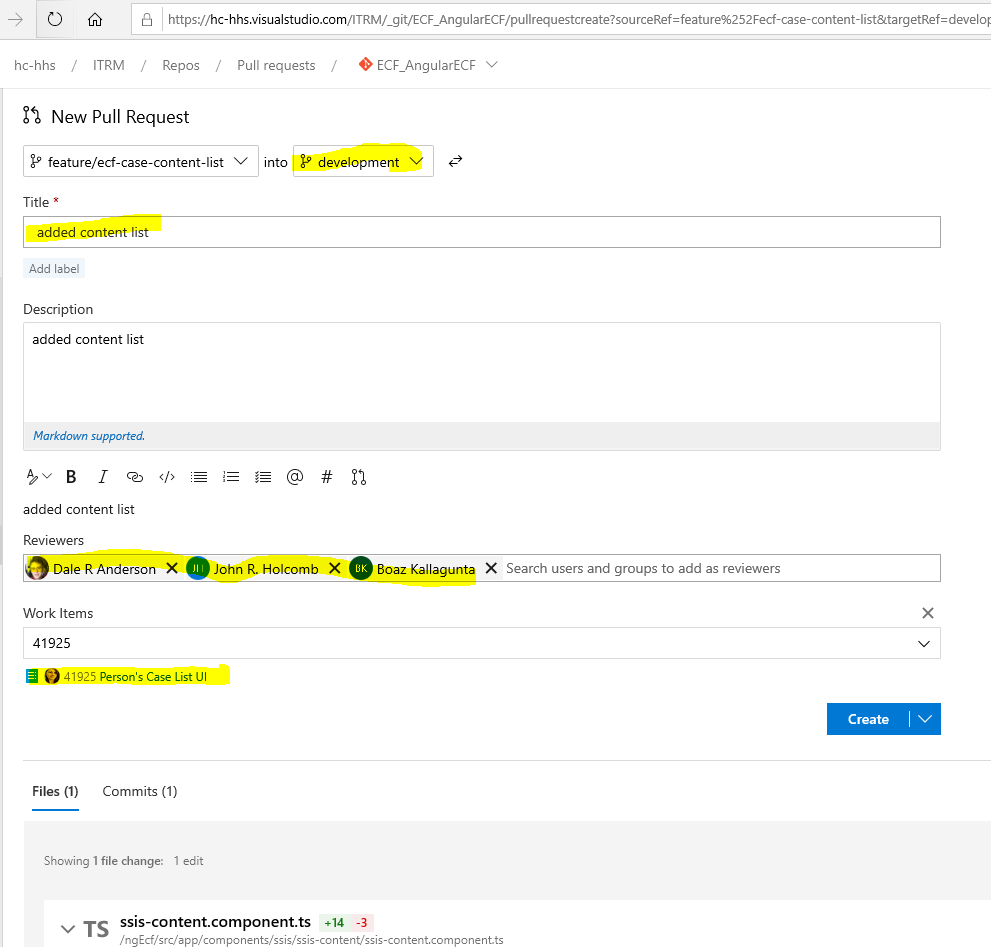


* Click on “Create a pull request” button highlighted above.



* Navigate to the “Pull requests” view in Azure DevOps.
* Click “Create a pull request” button shown to the right above.

1. Once you click the “Create a pull request” button, fill the form up, like below:



* Select “development” as the target branch
* Enter a title and description
* Enter all UI developers as reviewers. There’s a two-step process for our code review process. One, to test code and the second is for code quality.
* Enter an existing PBI number in the “Work Items” section
* Click “Create” to create the pull request and notify reviewers that a pull request has been created.

# Initial review of PBI

The purpose of the initial review is not to perform the same tasks as QA; but, to find bugs that are obvious. Examples, would be unexpected behavior of the UI, see unhandled errors in the browser’s console log, obvious misplacement of CSS Styles, etc.

## Steps Performed during the initial review

* Checkout the branch that is to be merged, example: “git checkout JHAngular”
* Perform this node command – “npm run buildQa”
* Launch the application – “npm start”
* Browse or navigate to the functionality that is to be merged and open the browser’s console to see logging, etc. As you click around or perform simple functionalities of the feature, look in the browser console for errors or unexpected behaviors.
* Report the issue to the reviewee and start the process over when code is pushed again

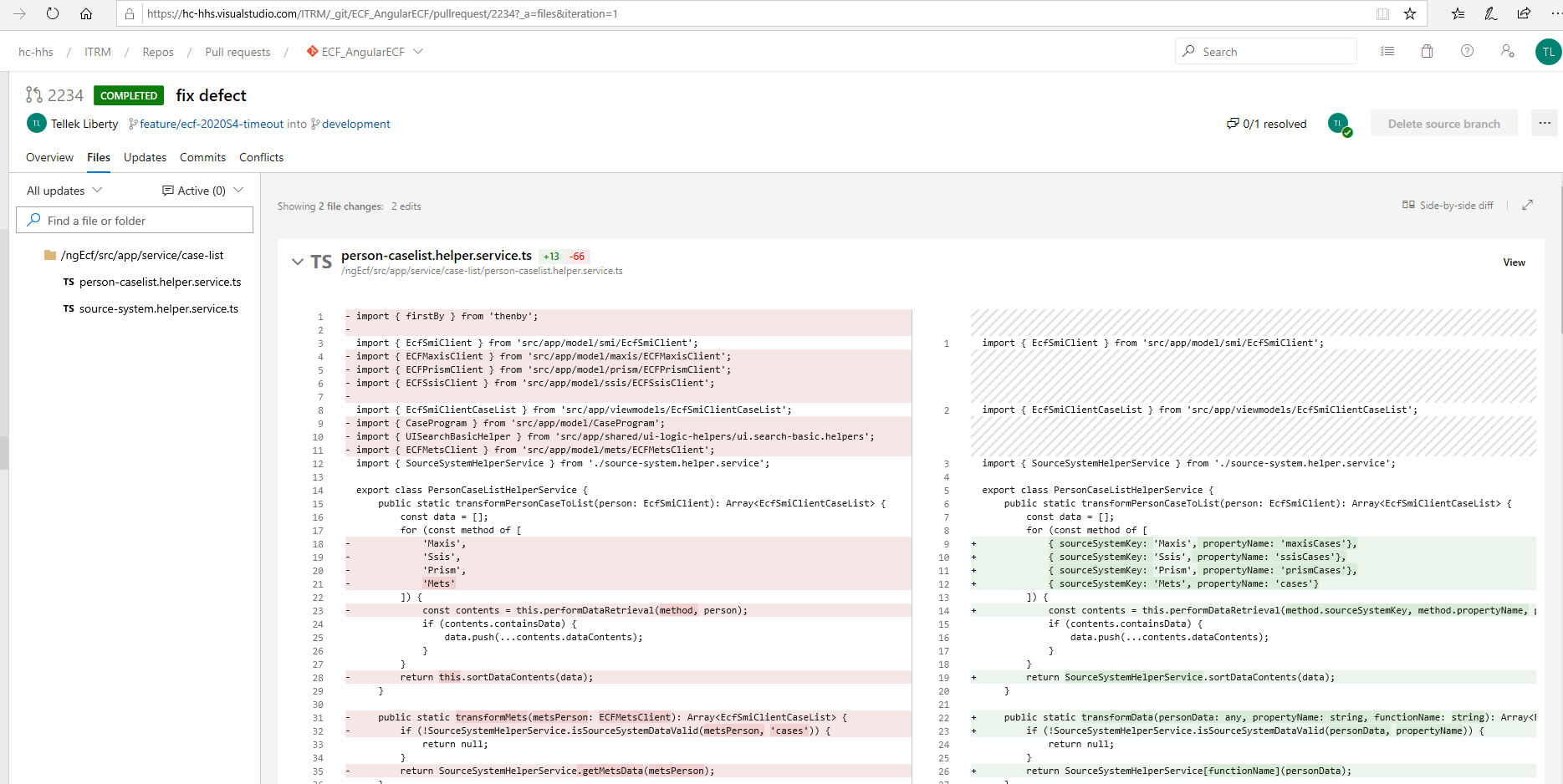
# Tasks Performed for Code Quality

The purposes of these tasks are:

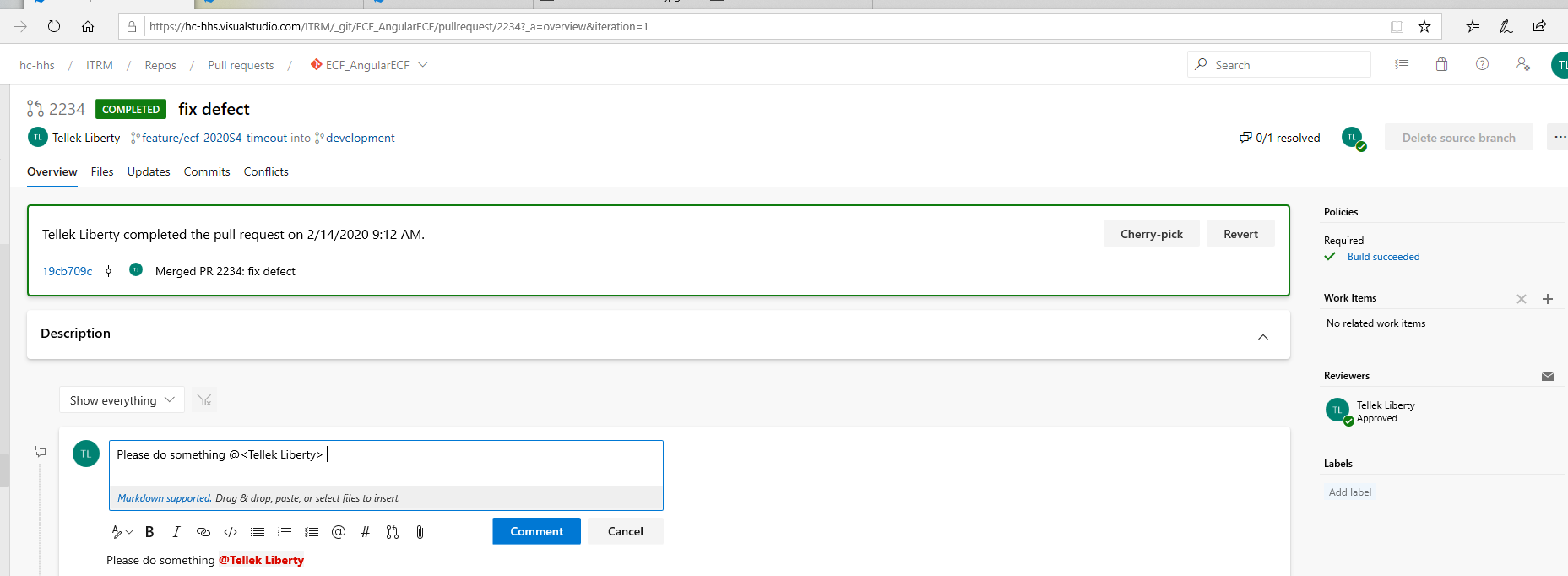
* Review for Code Quality
* Share Knowledge
* And keep code implementation within the technical architecture design selected by the Lead and Architect of ECF Next

### Steps

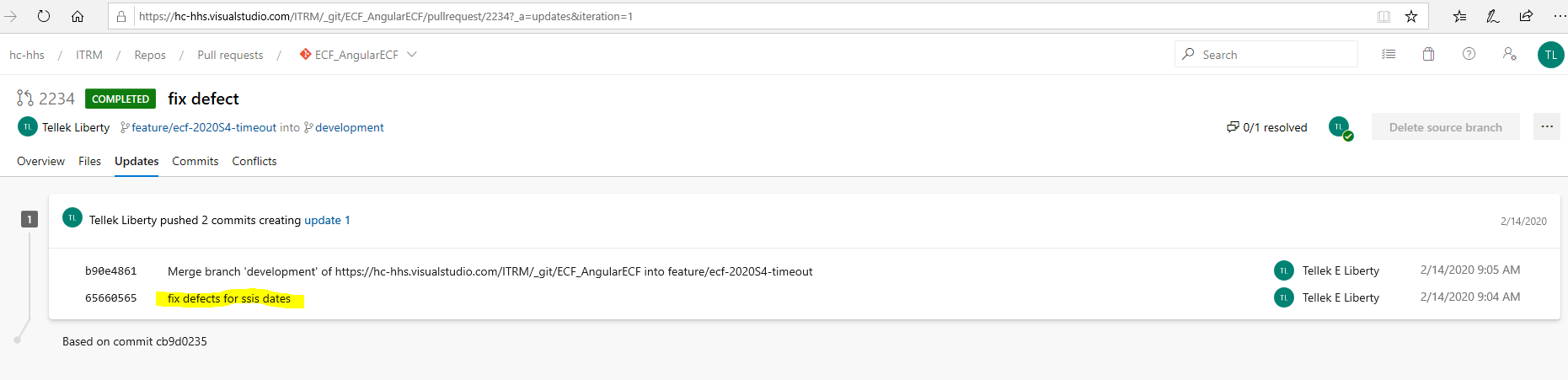
1. Review the code of the application from the “Files” Tab in the Pull Request view. See image below:



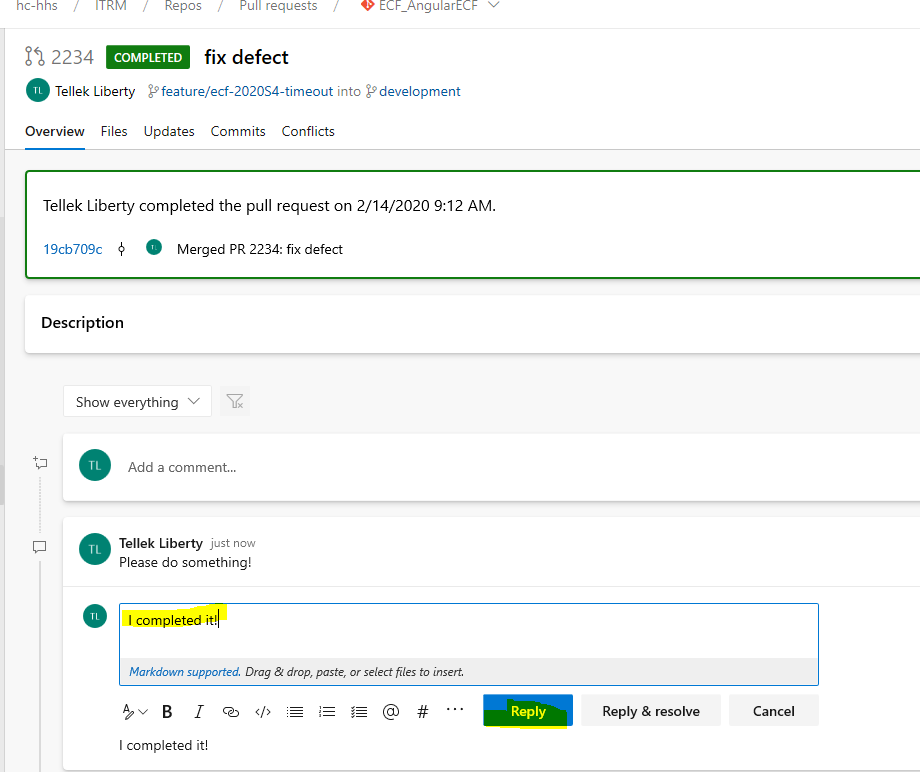
1. If there are any code anomalies, write a comment for the developer requesting a pull request. See image below:



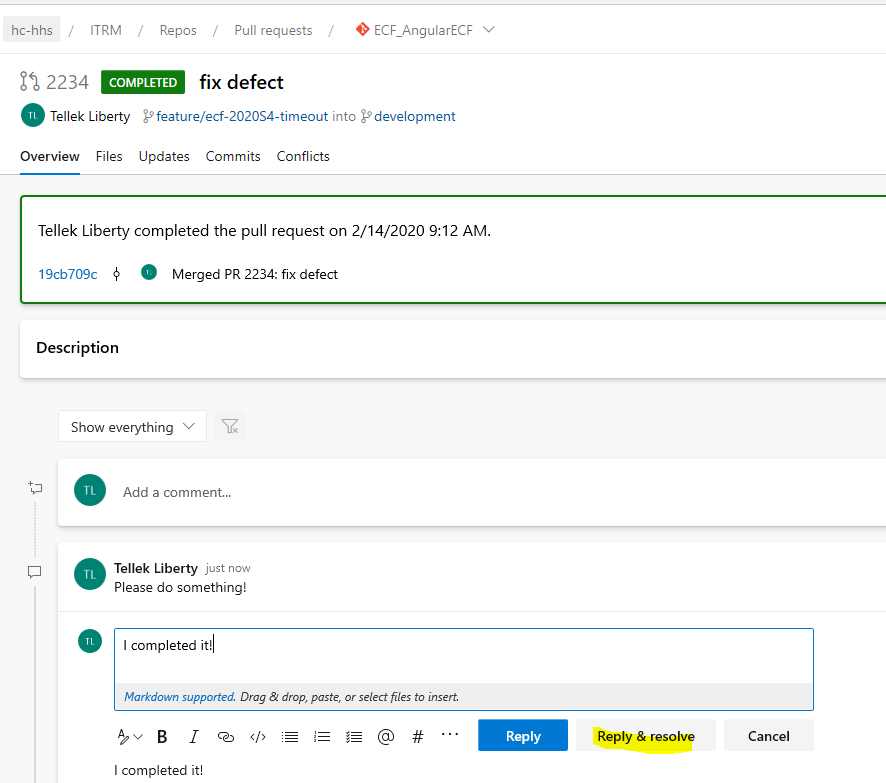
1. Once the reviewee sees the comment via email, he will make the change to his code and then commit and push to the branch under review. The changes will be reflected in the “Updates” tab. See image below:



1. The reviewee will then reply to the comment and mention the reviewer with the @mentioned keyword. Then click “Reply” to send an email to the reviewer. See image below:



1. Once resolved correctly by the reviewer. The reviewer will add a comment and click “Reply & resolve” to resolve the anomaly. See image below:

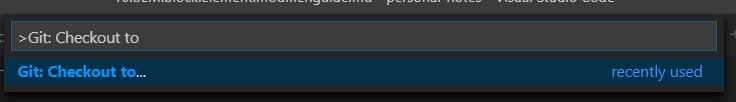


1. The reviewer will then click the “Approve” button and then perform the necessary steps to merge the branch to development.
2. Process Completed!

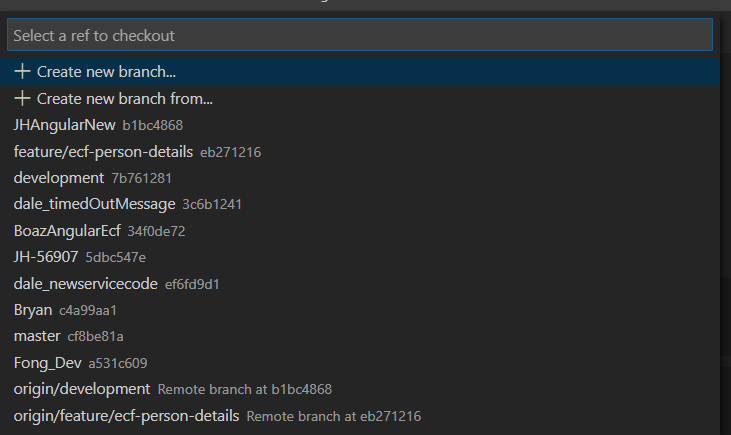
# Tutorials – GIT Source Control Extension (VS CODE)

## Creating a Branch using GIT Source Control

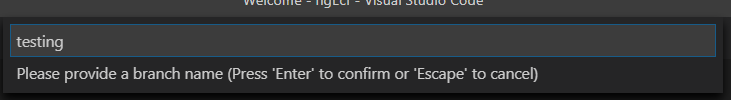
* In Visual Studio Code, press CTRL+SHIFT+P on the keyboard. Type “GIT: Checkout to” and hit enter



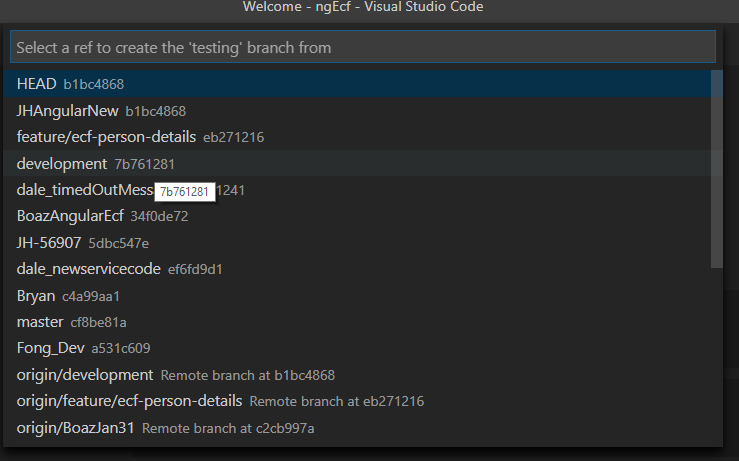
* The options below will display. Select or click “Create new branch from…” and hit enter



* Type the new branch name (in the image below, I deliberately typed testing) and hit enter



* Select “development” and hit enter (You may select any other branch you desire, I am choosing development because we will always use it to associate with PBI’s/defects/tasks).



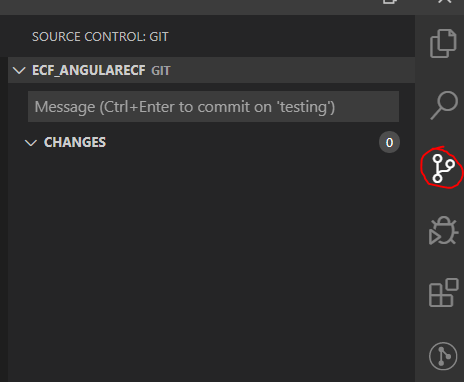
* Verify that your new branch is selected by looking at the lower left corner of Visual Studio Code Status Bar.



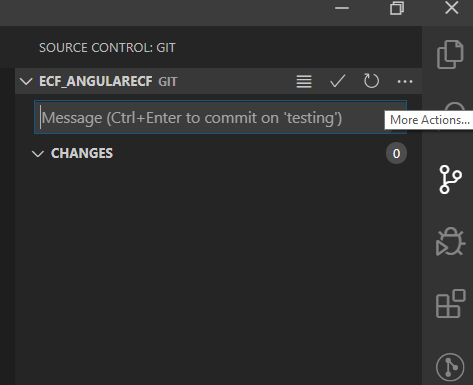
* Process is COMPLETED

## Publishing branches using Git

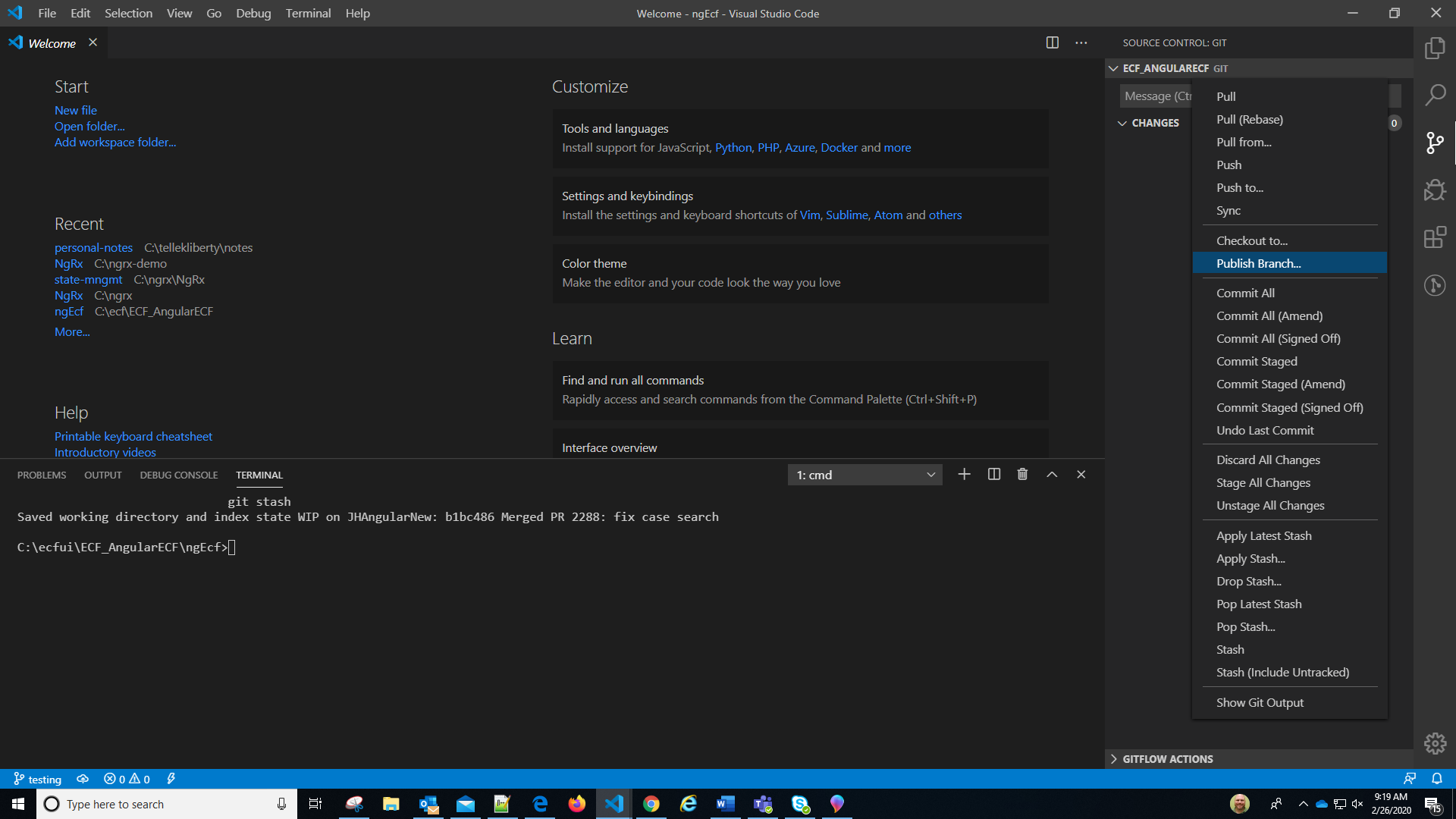
* Select the GIT Source Control Extension within Visual Studio Code



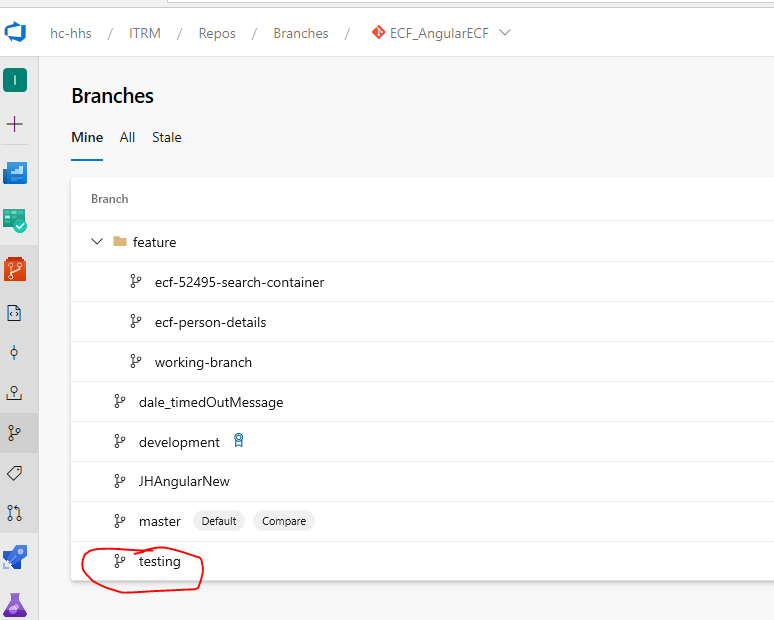
* Select or click the ellipsis on the GIT tab by hovering over the tab



* When the menu is shown, select “Publish Branch…”

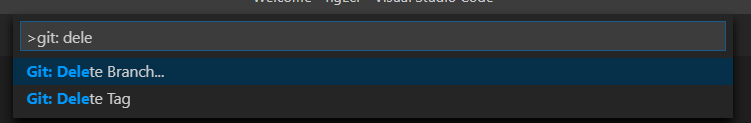


* Verify branch was publish in Azure DevOps

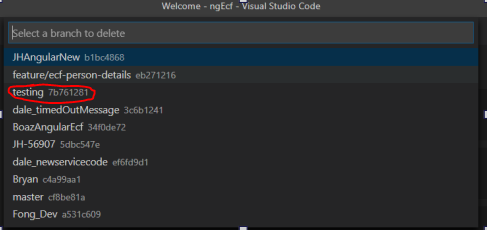


## Deleting the local branch

* Press CTRL+SHIFT+P to enter command – “GIT: Delete Branch…” and hit enter



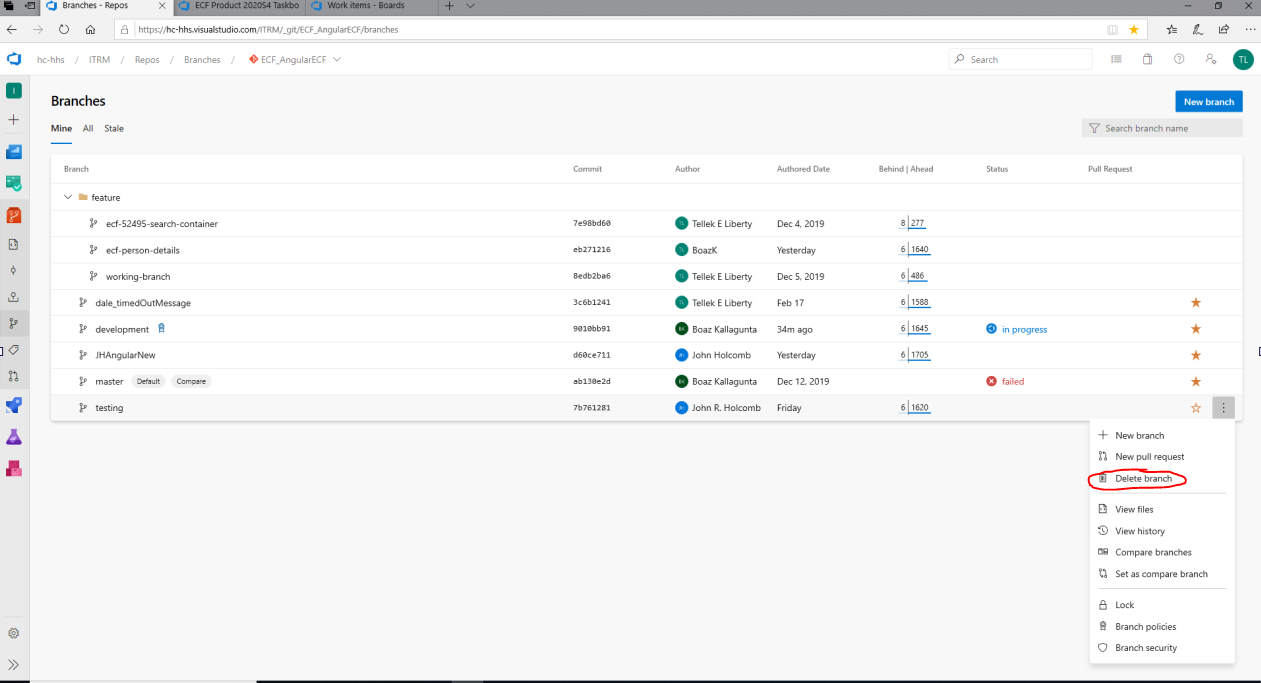
* Select your branch name – in this case, it is the “testing” branch.



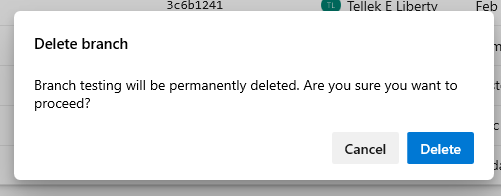
* Complete

## Deleting remote branch in Azure DevOps

* Navigate to ECF Next Azure DevOps Branch UI, click the ellipsis menu to the far right and click “Delete Branch”



* Click the “Delete” button in the confirmation prompt



* Complete

# Conclusion

If you have gotten here, thanks for reading. Please ensure to make some comment or recommendations to this process. Or, we can schedule another meeting. 😊

**Tellek E Liberty**