

Working with Pull Requests in Visual Studio Code and Azure DevOps

Overview

Pull requests let your team give feedback on changes in feature branches before merging the code into the master branch. Reviewers can step through the proposed changes, leave comments, and vote to approve or reject the code. Azure DevOps provides a rich experience for creating, reviewing, and approving pull requests.

It is required that you complete the Git lab prior to taking this lab.

Prerequisites

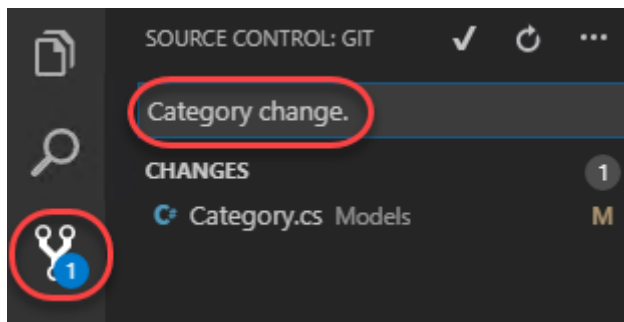
- Visual Studio Code with the C# extension installed.
- Azure Repos Extension for Visual Studio Code
- This lab requires you to complete task 1 from the prerequisite instructions.
- The Git lab is required to be completed as a prerequisite for this lab.

Exercise 1: Working with pull requests

When the Git lab ended, we had created a new branch and made a change to some of the code. Now we need to commit that change to the new branch and push it to the server. Once there, we can create a pull request so that the branch can be merged with the master.

Task 1: Creating a new pull request

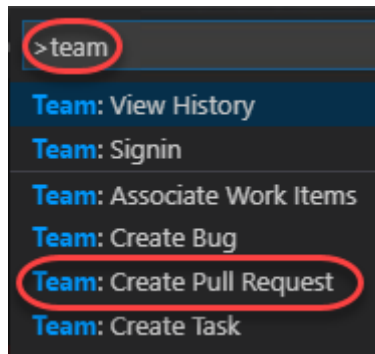
1. Return to **Visual Studio Code**.
2. Select the **Source Control** tab. It should recognize that you have uncommitted changes to **Category.cs**. Enter a comment of "**Category change**" and press **Ctrl+Enter** to commit to the local **release** branch.



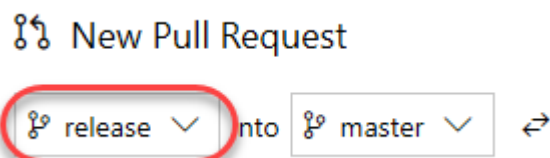
3. Click the **Synchronize Changes** button to push the commit to the server branch.



4. Press **Ctrl|Shift+P** to open the **Command Palette**.
5. Start typing "**Team**" and select **Team: Create Pull Request** when it becomes available. This will open a new pull request for the current branch in a new browser tab.




6. The **New Pull Request** form should already contain the key information reviewers will need, as well as who those reviewers should be (if any). If not, select **release** as the branch to merge into **master**.



7. You can customize any of this, and some of it may be required based on policy. Also note that the work item associated with the branch when created is referenced. Click **Create**.

New Pull Request

 release  into  master  

Title *

Category change

Add label

Description

Category change


Markdown supported.

Category change


Reviewers

Search users and groups to add as reviewers

Work Items 

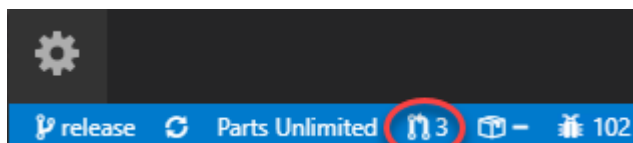
Search work items by ID or title 

  4133 Sel_IE_Navigate Failed in 2488

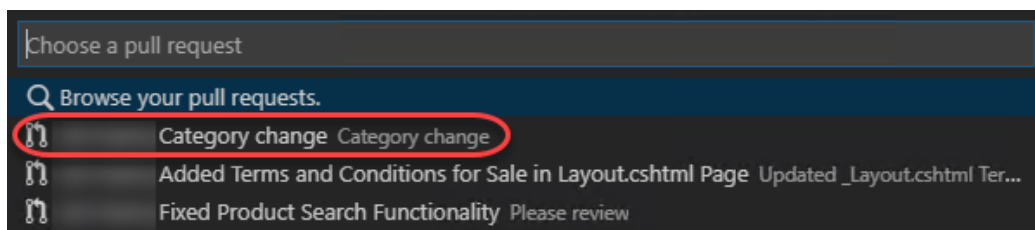
Create 

Task 2: Managing pull requests

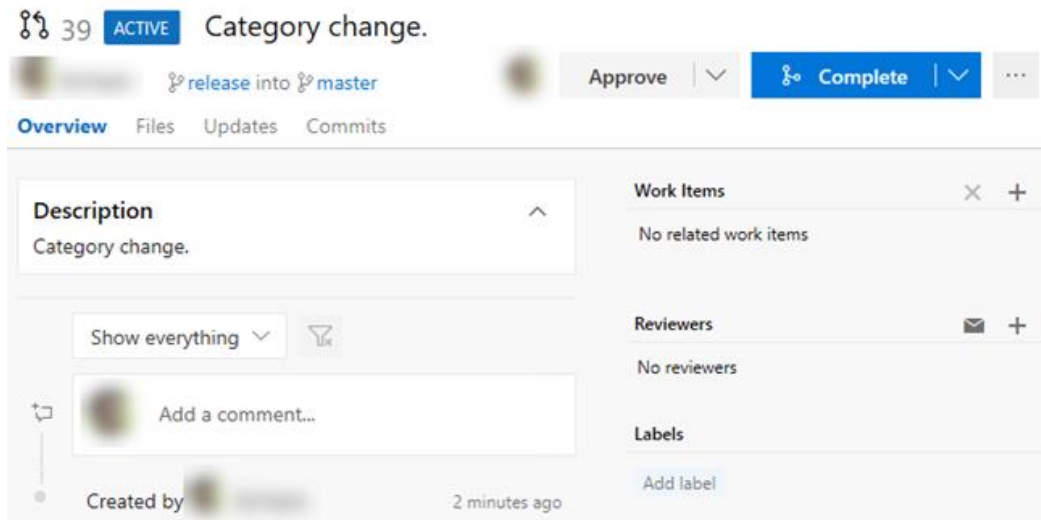
1. Return to **Visual Studio Code**.
2. Click the **Browse your pull requests** button at the bottom of the screen.



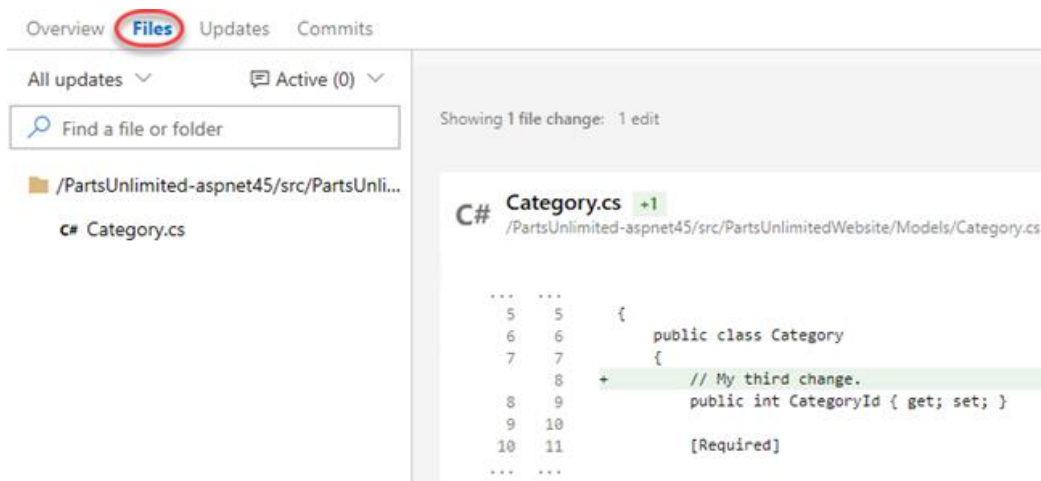
3. Select the pull request you just created. This will open it in a new browser tab.



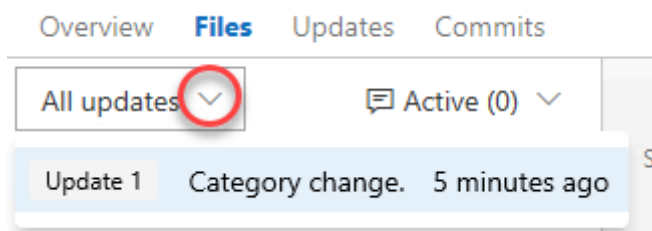
4. The **Overview** tab contains all of the key information specified in the creation form, as well as options to approve and complete the request.



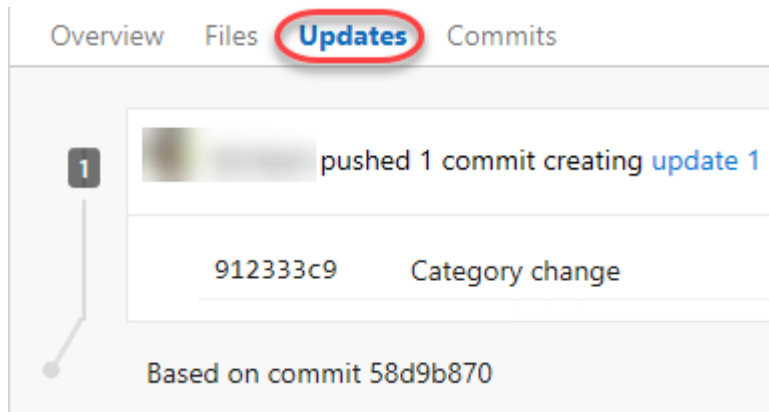
5. Select the **Files** tab to review the files involved in the commit.



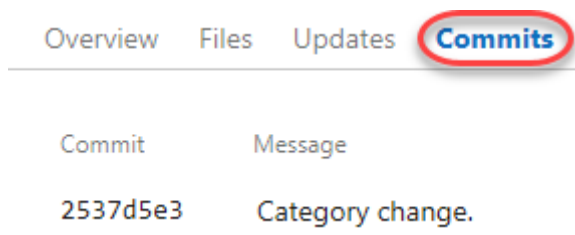
6. Note that you can select a specific update from the dropdown if you like.



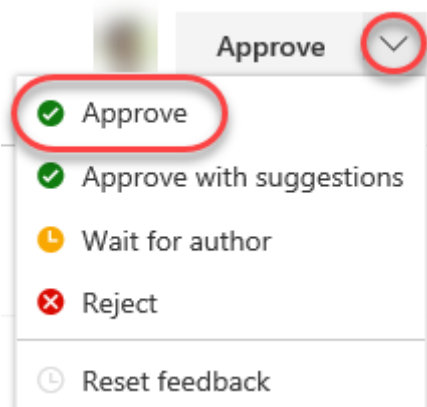
7. Click the **Add comment** button next to the source file. Enter a comment using markdown and click **Comment** to save it. Note that there is a live preview of your comment before you commit to it.



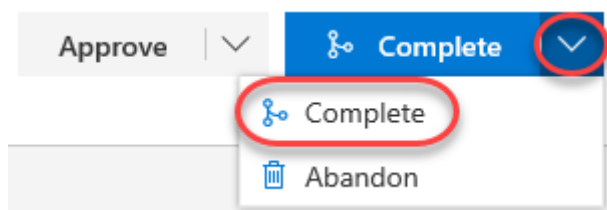
13. Select the **Commits** tab, where you can review the commits made to the branch.



14. Since everything seems to be in order, **Approve** the pull request.



15. Since the approvers have signed off, **Complete** the pull request.



16. You can accept the default messages in the pull request dialog. The first option is to complete the work items linked to the branch being merged. Note that you also have the option to delete the target branch after merging, as well as the ability to **squash changes** during the merge. Squash merging is a merge option that allows you to condense the Git history of topic branches when you complete a pull

request. Instead of each commit on the topic branch being added to the history of the default branch, a squash merge takes all the file changes and adds them to a single new commit on the default branch. Squash merging keeps your default branch histories clean and easy to follow without demanding any workflow changes on your team. Click **Complete merge**.

Complete pull request ×

Merged PR 48: Category change

Category change

Related work items: #4313

☒ Complete linked work items after merging

☒ Delete release after merging

☐ Squash changes when merging [Learn more](#)

Complete merge Cancel

17. When the merge completes, the pull request should be marked as **Completed**.

45 **COMPLETED** Category change


18. Return to the **Pull Requests** home.


/ Parts Unlimited / **Pull requests** / PartsUnlimited ▾

19. Select the **Completed** tab and click the pull request as though you were visiting it fresh.

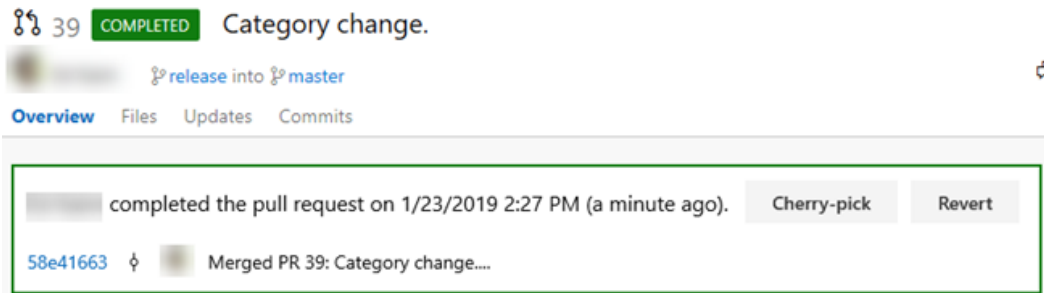
Pull Requests

Mine Active **Completed** Abandoned

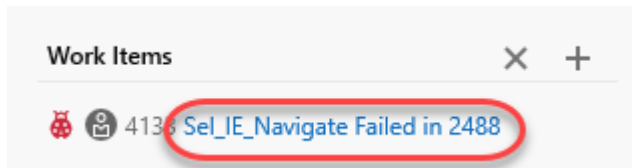
 **Category change**

requested #48 into  master

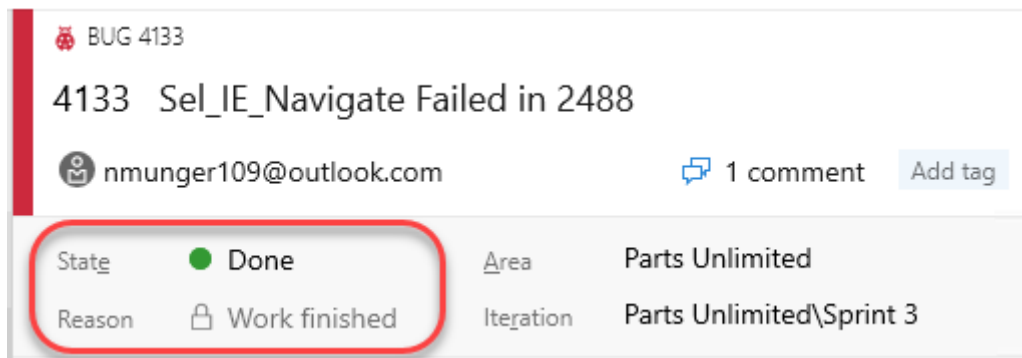
20. You can review the pull request, as well as **Cherry-pick** or **Revert** it if needed. Cherry-picking is the process of selecting specific commits from one branch to apply to another, conceptually similar to a copy/paste operation.



21. Under **Work Items**, click one of the linked work items.



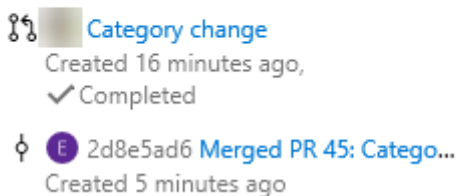
22. Note that the work item has now been marked as **Done**.



23. Under the **Development** tab, you can see the commit and pull request have been associated with the work item.

Development

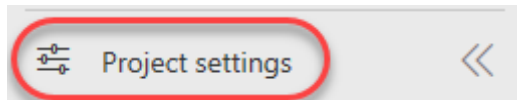
+ Add link



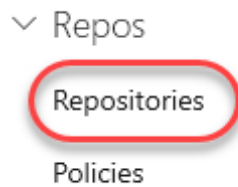
Task 3: Managing Git branch and pull request policies

As projects and teams scale in complexity, it becomes help to automate more of the processes put in place to ensure quality.

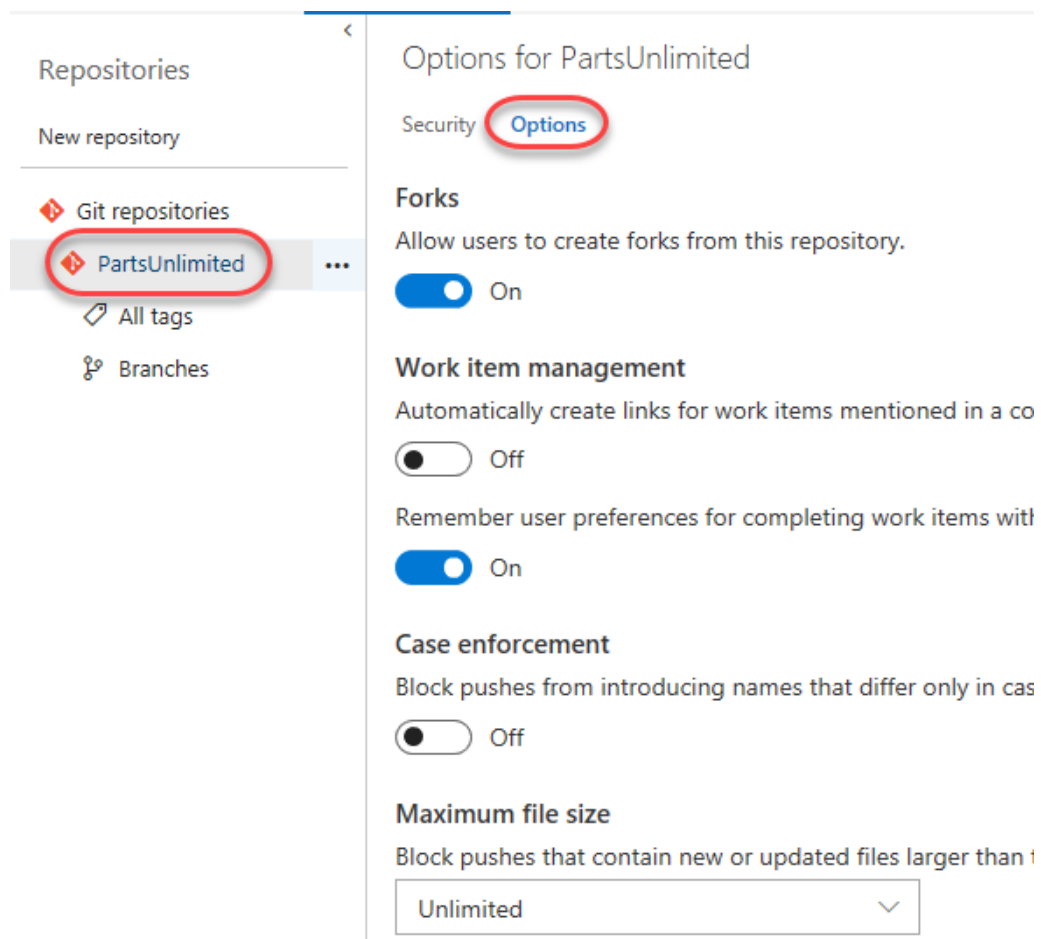
1. Open **Project settings**.



2. Select **Repositories** under **Repos**.



3. Select the **PartsUnlimited** repo. Like everything else in Azure DevOps, you can manage security to a great level of detail. Select the **Options** tab. This allows you to set some useful policies, such as whether or not you want to allow users to fork the repo, whether discussed work items are automatically linked, and so on.



4. Select the **master** branch. Like the repo, you have a great deal of control over its security settings. You can also define a wide variety of policies for the branch to enforce. Click **Branch Policies**.

The screenshot displays the Azure DevOps web interface. On the left, the 'Repositories' sidebar shows a tree view under 'PartsUnlimited' with 'Branches' expanded. The 'master' branch is highlighted with a red circle. On the right, the 'Security for master branch in PartsUnlimited' page is shown, with the 'Branch Policies' tab selected and circled in red. Below the tabs, a list of 'VSTS Groups' is visible, including 'Build Administrators', 'Contributors', 'Project Administrators', 'Readers', 'Project Collection Administrators', 'Project Collection Build Service Accounts', and 'Project Collection Service Accounts'. Under the 'Users' section, 'Microsoft.VisualStudio.Services.TFS' and 'Project Collection Build Service (sharplogic)' are listed.

5. Azure DevOps branch policies are very effective in enforcing a level of quality control in the repo. For example, you can control pull requests by requiring a minimum number of reviewers, checking for linked work items, requiring that all comments have been resolved, and more. You can even require validation through a successful build and configure external approval services. If there are certain sections of code that require their own approvers to be included, you can include them here as well.

Policies for: Parts Unlimited > PartsUnlimited > master

 Save changes  Discard changes

Protect this branch

- Setting a Required policy will enforce the use of pull requests when updating the branch
- Setting a Required policy will prevent branch deletion
- Manage permissions for this branch on the [Security page](#)

☐ Require a minimum number of reviewers

Require approval from a specified number of reviewers on pull requests.

☐ Check for linked work items

Encourage traceability by checking for linked work items on pull requests.

☐ Check for comment resolution

Check to see that all comments have been resolved on pull requests.

☐ Enforce a merge strategy

Require a specific type of merge when pull requests are completed.

Build validation

Validate code by pre-merging and building pull request changes

 Add build policy


Require approval from external services

Require third party services to post successful status to complete pull requests. [Learn more](#)

 Add status policy

Automatically include code reviewers

Include specific users or groups in the code review based on which files changed.

 Add automatic reviewers

6. Check **Require a minimum number of reviewers**. By default, this requires at least two reviewers to approve a pull request, and also requires (by default) that the original author is not one of them.



Require a minimum number of reviewers

Require approval from a specified number of reviewers on pull requests.

Minimum number of reviewers

2

☐

Allow users to approve their own changes.

☐

Allow completion even if some reviewers vote "Waiting" or "Reject".

☐

Reset code reviewer votes when there are new changes.

7. Click **Add automatic reviewers**.

Automatically include code reviewers

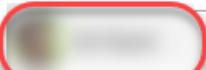
Include specific users or groups in the code review based on which files changed.

+ Add automatic reviewers

8. Add yourself as a **Reviewer** and set the **Path filter** to `"/PartsUnlimited-aspnet45/src/PartsUnlimitedWebsite/Controllers/*"`. Having the requirement of **Required** ensures that you will be required to sign off on any changes proposed to the controllers code in the web app. Click **Save**.

Add automatic reviewers

Reviewer(s) *



Search users and groups

Path filter (optional) ⓘ

/PartsUnlimited-aspnet45/src/PartsUnlimitedWebsite/Controllers/*



Policy requirement

☒

Required

All reviewers must approve to complete pull requests.

☐

Optional

Reviewers will be added automatically for configured paths, but approvals are not required to complete pull requests.



Custom message

Save

Cancel

9. Click **Save changes** to save the changes to the policy.

Policies for: Parts Unlimited > PartsUnlimited > master

 Save changes  Discard changes
