

Activate Azure with DevOps

Module 06: End-to-End DevOps - Lab 6.2 - Develop and Test

Student Lab Manual

Conditions and Terms of Use

Microsoft Confidential - For Internal Use Only

This training package is proprietary and confidential and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Copyright and Trademarks

© 2020 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see **Use of Microsoft Copyrighted Content** at <http://www.microsoft.com/about/legal/permissions/>

Microsoft®, Internet Explorer®, and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Contents

Introduction

Prerequisites

Exercise 1: Develop and Test

Task 1: Configuring Visual Studio Code

Task 2: Clone a remote repository

Task 3: Create a topic branch

Task 4: Installing the Azure Repos extension for Visual Studio Code

Task 5: Push your code changes to the topic branch

Task 6: Merge changes using pull requests

Task 7: Create a build definition

Task 8: Setup a CI build for topic branches

Task 9: Protect the master branch

Task 10: Approve a Pull Request

Task 11: Monitor Code and Build KPIs

Lab 6.2: End-to-End DevOps - Develop and Test

Introduction

In this lab, you will add an existing Visual Studio solution to your code repository. You will then setup a build definition to build your code and run your unit tests.

You'll learn:

- Understand the basic features of Git as a source control management system.
- Envision a simple source control management workflow to support the business needs in a DevOps focused team.

Prerequisites

The following is required to complete this hands-on lab:

- Microsoft [Visual Studio Code](#) with the [C# extension](#) and [Azure Repos](#) extension installed
- Git local source control management is installed. If it is not currently installed, you can install it from the following URL: <https://git-scm.com/download/>
- Complete **Task 1** to generate PartsUnlimited project from the demo generator: <https://azuredevopslabs.com/labs/azuredevops/prereq/> OR Complete **Agile Planning and Portfolio Management with Azure Boards** lab from earlier: <https://azuredevopslabs.com/labs/azuredevops/agile/>

Estimated Time to Complete This Lab

90 minutes

Exercise 1: Develop and Test

Task 1: Configuring Visual Studio Code

1. Open **Visual Studio Code**. In this task, you will configure a Git credential helper to securely store the Git credentials used to communicate with Azure DevOps. If you have already configured a credential helper and Git identity, you can skip to the next task.
2. From the main menu, select **Terminal | New Terminal** to open a terminal window.
3. Execute the command below to configure a credential helper.

```
git config --global credential.helper wincred
```

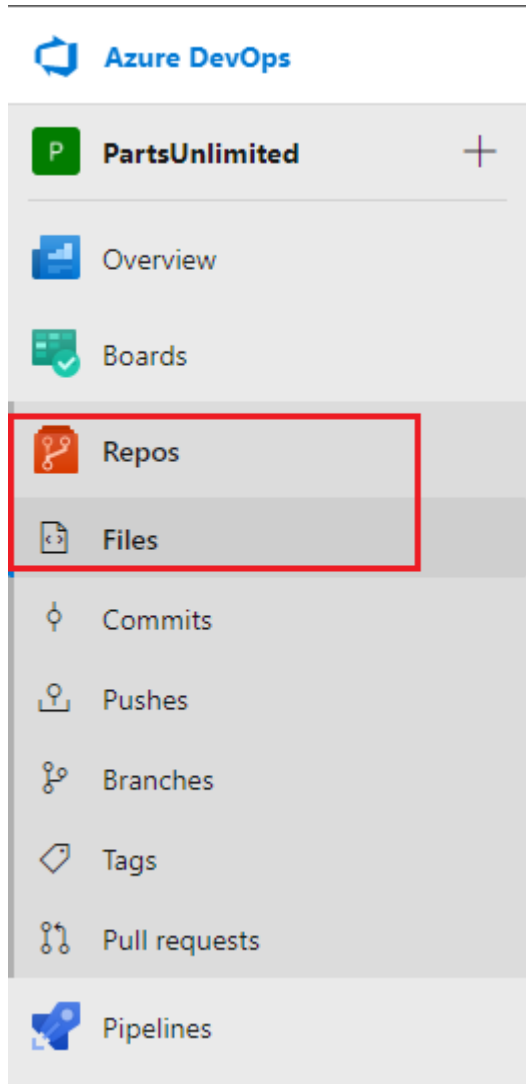
4. The commands below will configure your username and email for Git commits. Replace the parameters with your preferred username and email and execute them.

```
git config --global user.name "John Doe"  
git config --global user.email johndoe@example.com
```

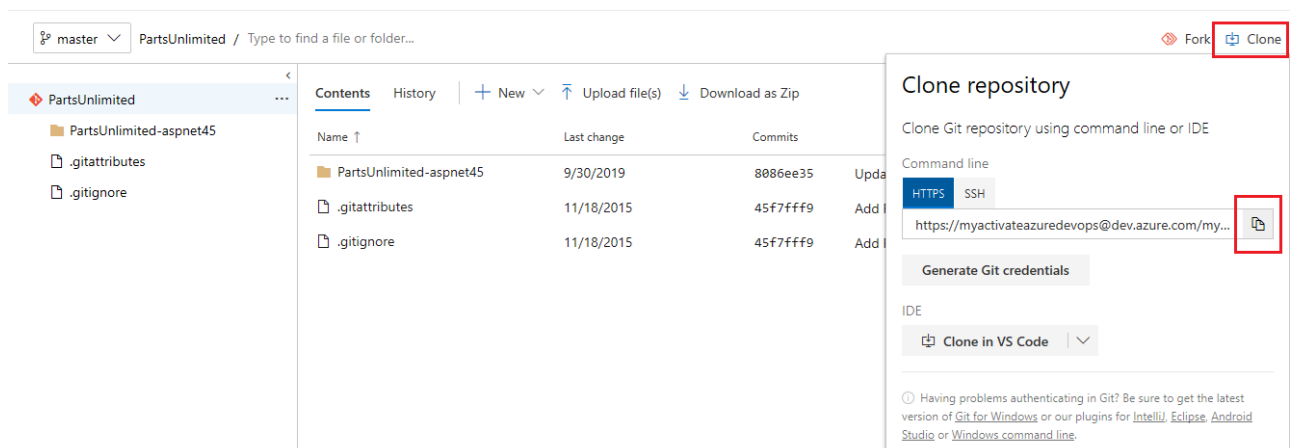
Task 2: Clone a remote repository

By default, the project creation adds a git repository on the server.

1. Create a "Repos" folder on your local machine (ex: "C:\Repos").
2. Back in Azure DevOps, go to the Repos>Files menu:



3. Clone the repository using the HTTPS URL:



4. Open Git Bash on your computer and type the following (Replacing the organization name and project with the name you chose):

```
cd C:/Repos
```

```
git clone  
https://myactivateazuredevops@dev.azure.com/myactivateazuredevops/PartsUnlimited/_git/PartsUnlimited
```

```
cd partsunlimited
```

Note: Above command and all other Git commands below can be performed from VS Code as well by opening a **New Terminal** and navigating to C:\Repos

5. To see the URLs that Git has stored for the short name (origin) for reading and writing to the remote repository, type:

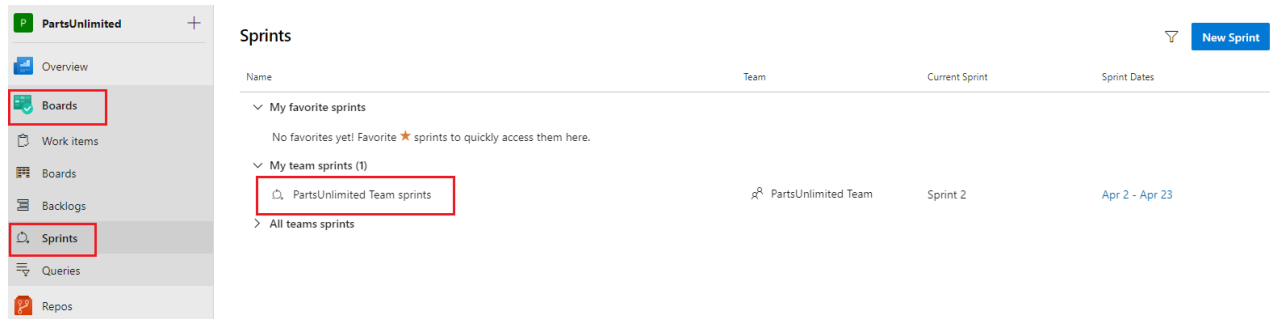
```
git remote -v
```

```
$ git remote -v  
origin https://myactivateazuredevops@dev.azure.com/myactivateazuredevops/hol/_git/PartsUnlimited (fetch)  
origin https://myactivateazuredevops@dev.azure.com/myactivateazuredevops/hol/_git/PartsUnlimited (push)
```

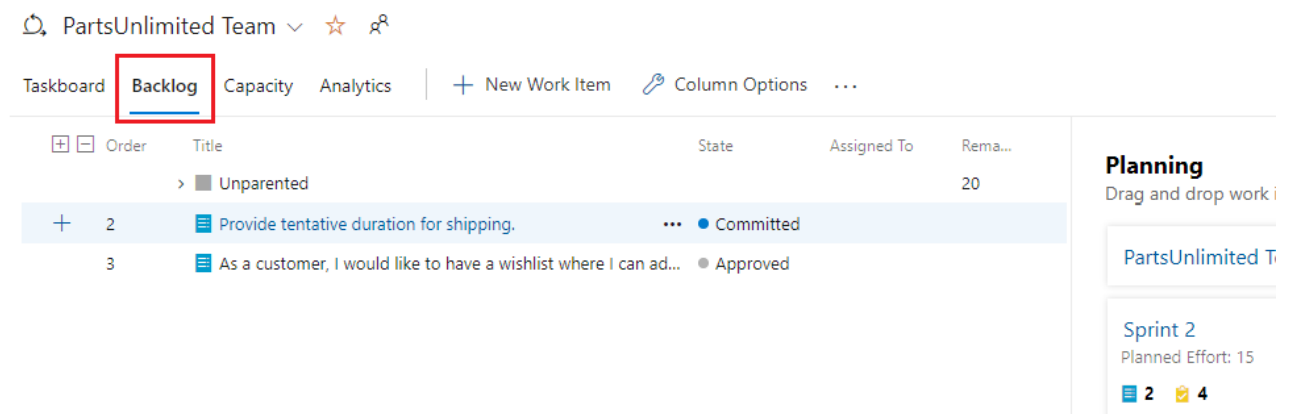
Note: Now that we have cloned the remote repository locally, in the remaining exercise we will set up a strategy that supports parallel development, a practice highly needed in Agile teams.

Task 3: Create a topic branch

1. Navigate to **Boards** -> **Sprint** and select **PartsUnlimited Team sprints**. Note: PartsUnlimited is the name of the team project in the screenshots, this may differ depending on the name you chose for your project.

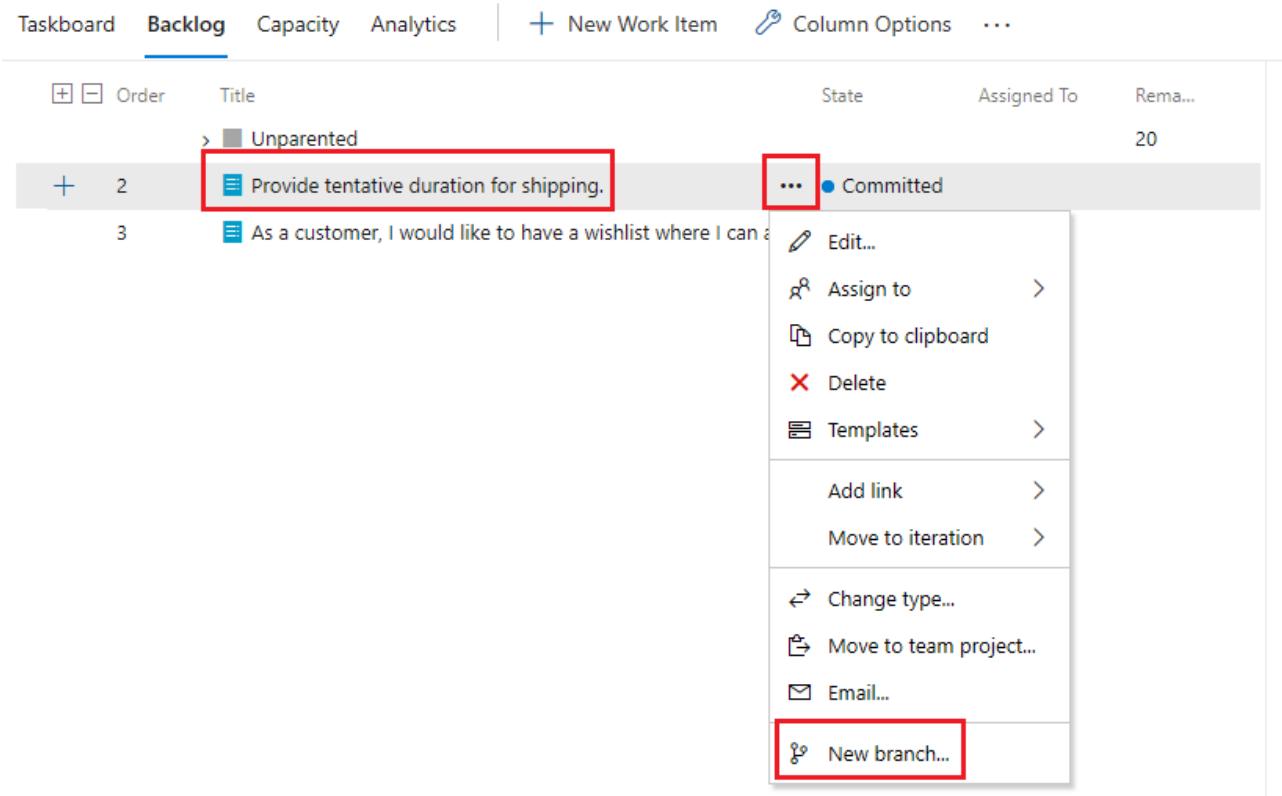


2. Select **Backlog** view for the current Sprint and switch to **Sprint 2**



3. Use the ellipses menu to the right-side for the work item **Provide tentative duration for shipping.** and create a topic branch for this work item:

Note: If you don't see this work item, create a new work item and create a topic branch for it.



4. Name the branch "topics/**tb-WID**" where WID is the work item Id (ex. topics/tb-36). This will place the tb-36 branch under the folder topics:

✕

Create a branch

Name

topics/tb-36

Based on

◆ PartsUnlimited ▼

🔗 master ▼

Work items to link

Search work items by ID or title ▼

☰ 👤 36 Provide tentative duration for shipping.
 Updated 24 minutes ago, ● Committed

Create branch
Cancel

5. Click **Create branch** to finish the branch creation.

Note: By following this approach, all the commits made against this branch will be automatically traced to the work item, regardless of the tool you choose to do your commits.

6. Click on **Branches** to view all your branches and the topics folder you placed them under.

H
hol
+

Overview

Boards

Repos

Files

Commits

Pushes

Branches

Tags

Branches

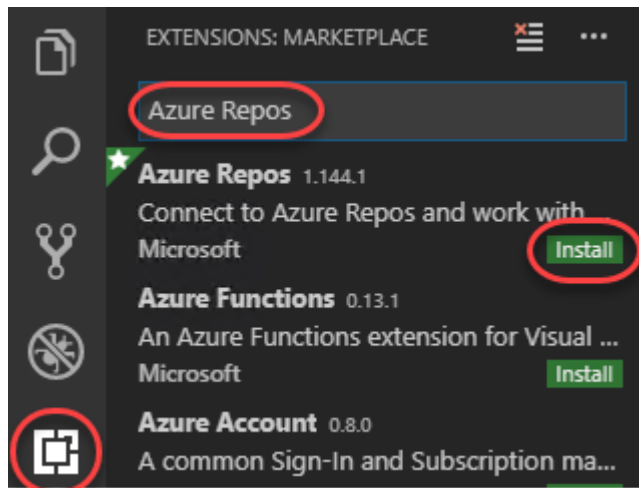
Mine
All
Stale

Branch	Commit
<div style="display: flex; align-items: center;"> ▼ topics </div>	
<div style="display: flex; align-items: center;"> 🔗 tb-36 </div>	🗑 8086ee35
<div style="display: flex; align-items: center;"> 🔗 AddTerms&Conditions </div>	🗑 9da57edd
<div style="display: flex; align-items: center;"> 🔗 CodedUITest </div>	🗑 96135fd6
<div style="display: flex; align-items: center;"> 🔗 demo-start </div>	🗑 816d9487
<div style="display: flex; align-items: center;"> 🔗 DependencyValidation </div>	🗑 1ca13fb2
<div style="display: flex; align-items: center;"> 🔗 e2e-complete </div>	🗑 a24bb398

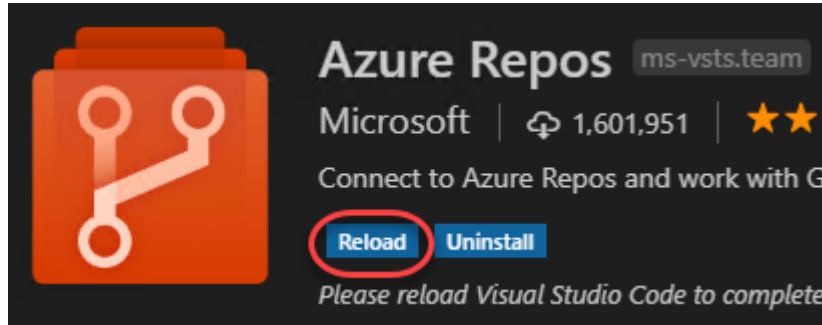
Task 4: Installing the Azure Repos extension for Visual Studio Code

Note: If you have Azure Repos extension already installed then move on to Step 4.

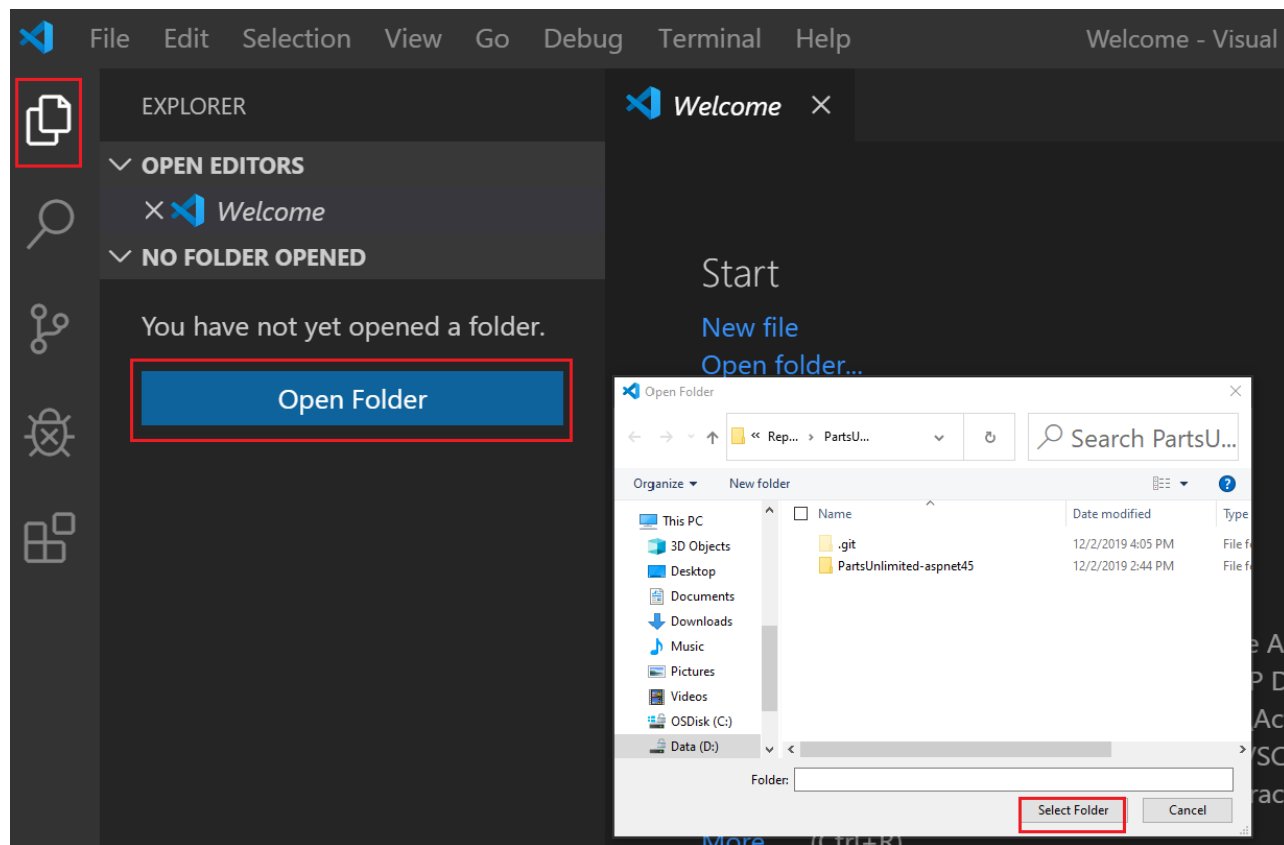
1. Open Visual Studio Code.
2. The Azure Repos extension provides convenient access to many features of Azure DevOps. From the **Extensions** tab, search for "**Azure Repos**" and click **Install** to install it.



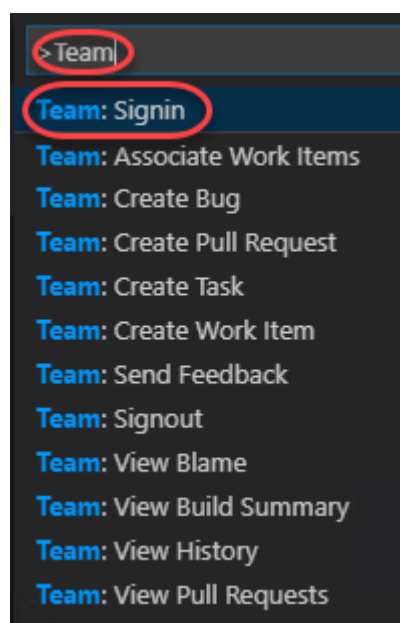
3. Click **Reload** once the extension has finished installing. If this option is not available, reopen Visual Studio Code.



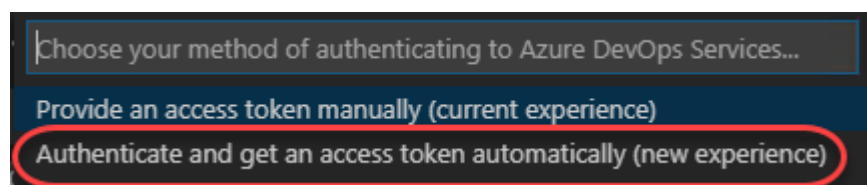
4. Press **Ctrl+Shift+E** and open the folder for the project you have been working on such as C:\Repos\PartsUnlimited



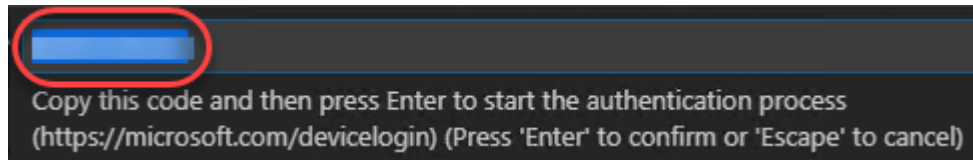
5. Press **Ctrl+Shift+P** to show the **Command Palette**.
6. Search for **"Team"** to see all the new commands that are now available for working with Azure Repos.
Select **Team: Signin**.



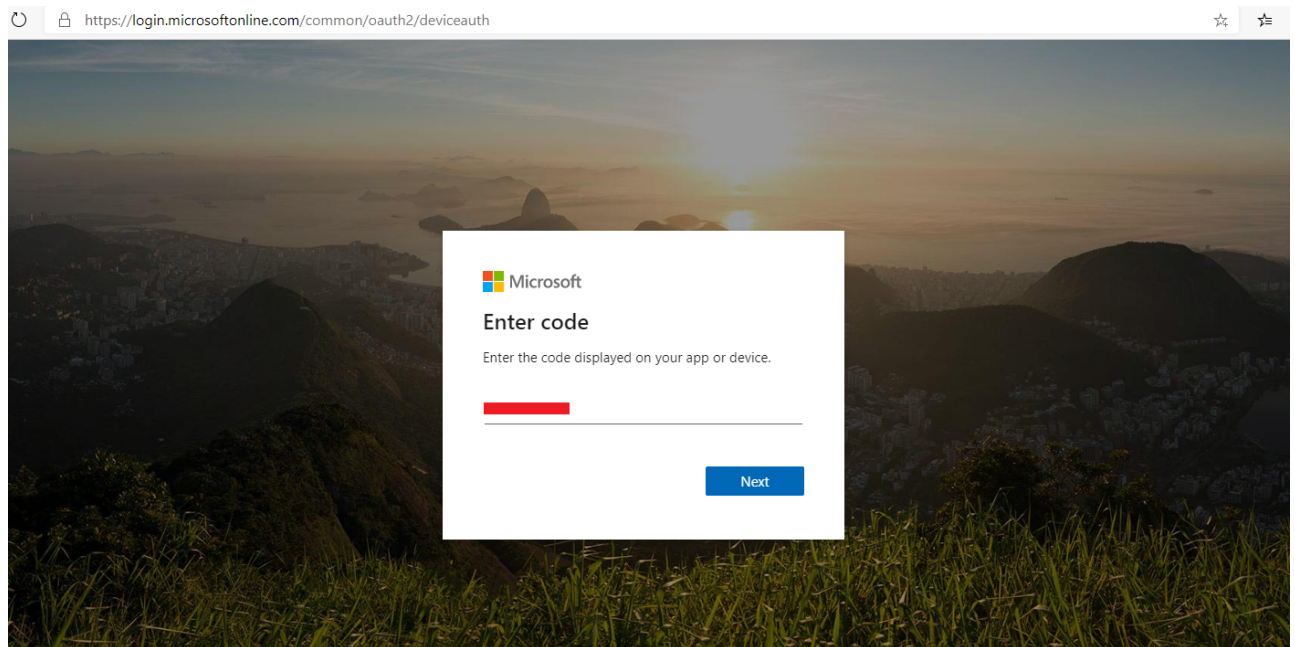
7. Select **Authenticate and get an access token automatically**. Note that you could alternatively provide the token created earlier if following the manual path.



8. Copy the provided token and press **Enter** to launch a browser tab.



9. Paste the code into the login box and click **Continue**.



10. Select the Microsoft account associated with your Azure DevOps account.
11. When the process has complete, close the browser tab.

Task 5: Push your code changes to the topic branch

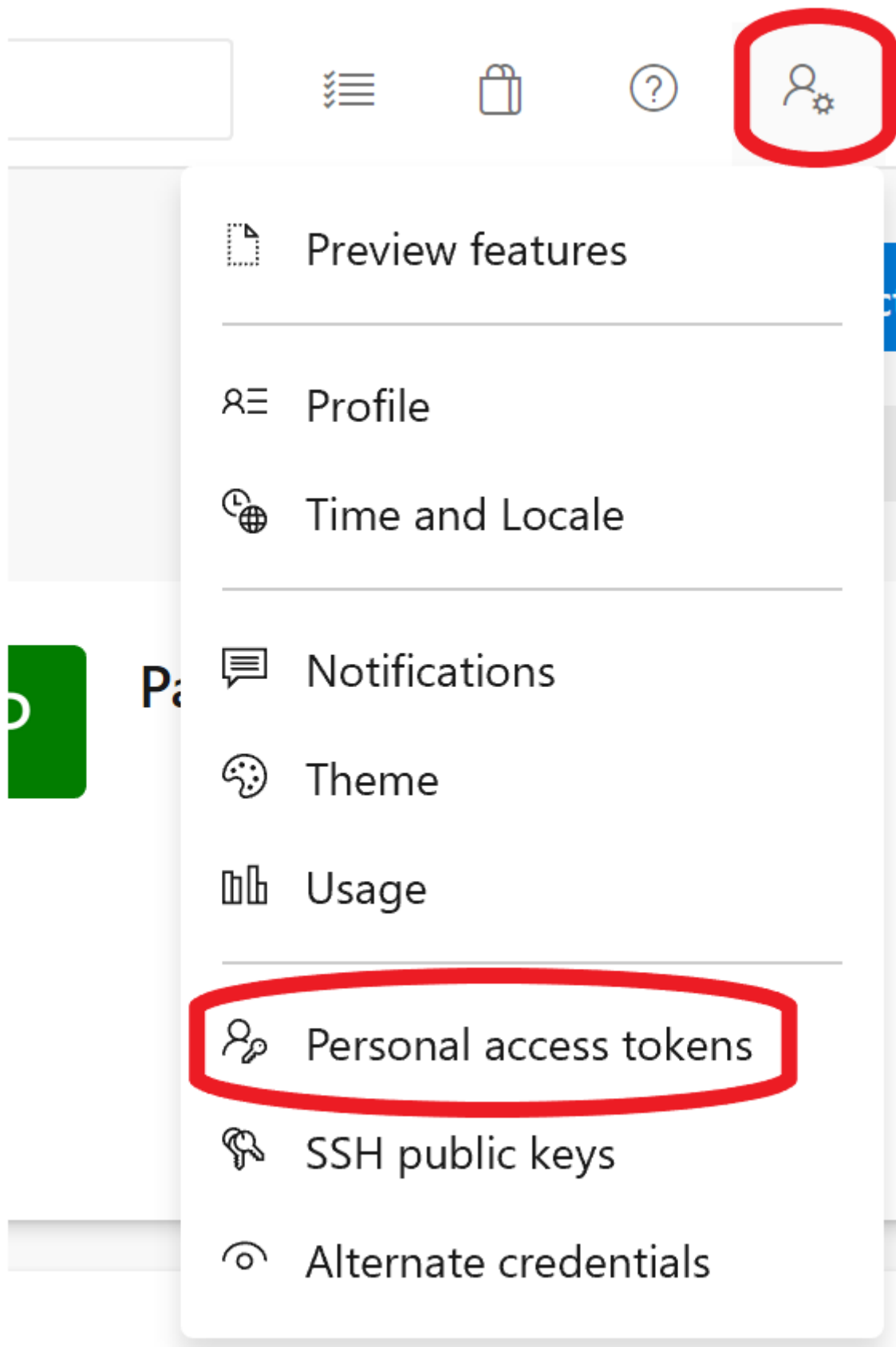
In this task, you will use Visual Studio Code to push your changes to the topic branch.

From your development workstation:

1. Continue working with the instance of Visual Studio Code opened in the previous task.
2. We are going to make a change and associate it with the topics branch. We will first need to sync your local repository with the remote one so that you can see the remote branch.
3. Click on the **Sync** button.



4. If VS Code doesn't ask for a password, Skip to step 7. If prompted to enter a password, go to dev.azure.com and create a personal access token.



5. Select **"New Token"**, Name the token, change the scope to full access, and click **"Create"**. Be sure to copy the code because you will not be able to access it again.

Create a new personal access token



Name

VSCode PAT

Organization

Expiration (UTC)

30 days

5/2/2020



Scopes

Authorize the scope of access associated with this token

Scopes

☒ Full access

☐ Custom defined

Create

Cancel

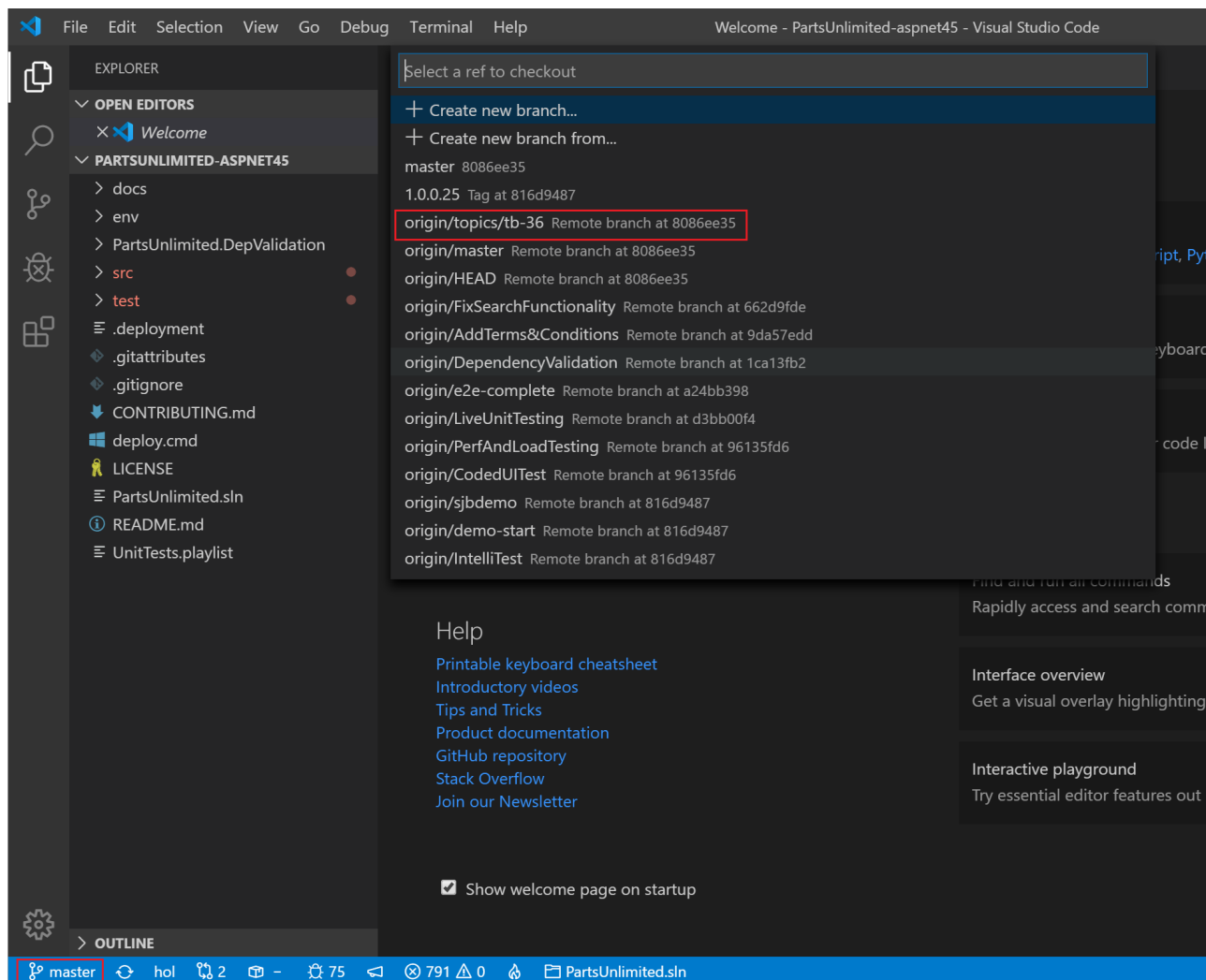
6. Paste the PAT code in VSCode and the sync will begin.

7. Click on the **master** branch, you will need to switch to "**Topics/tb-WID**" that you created on the remote server earlier.

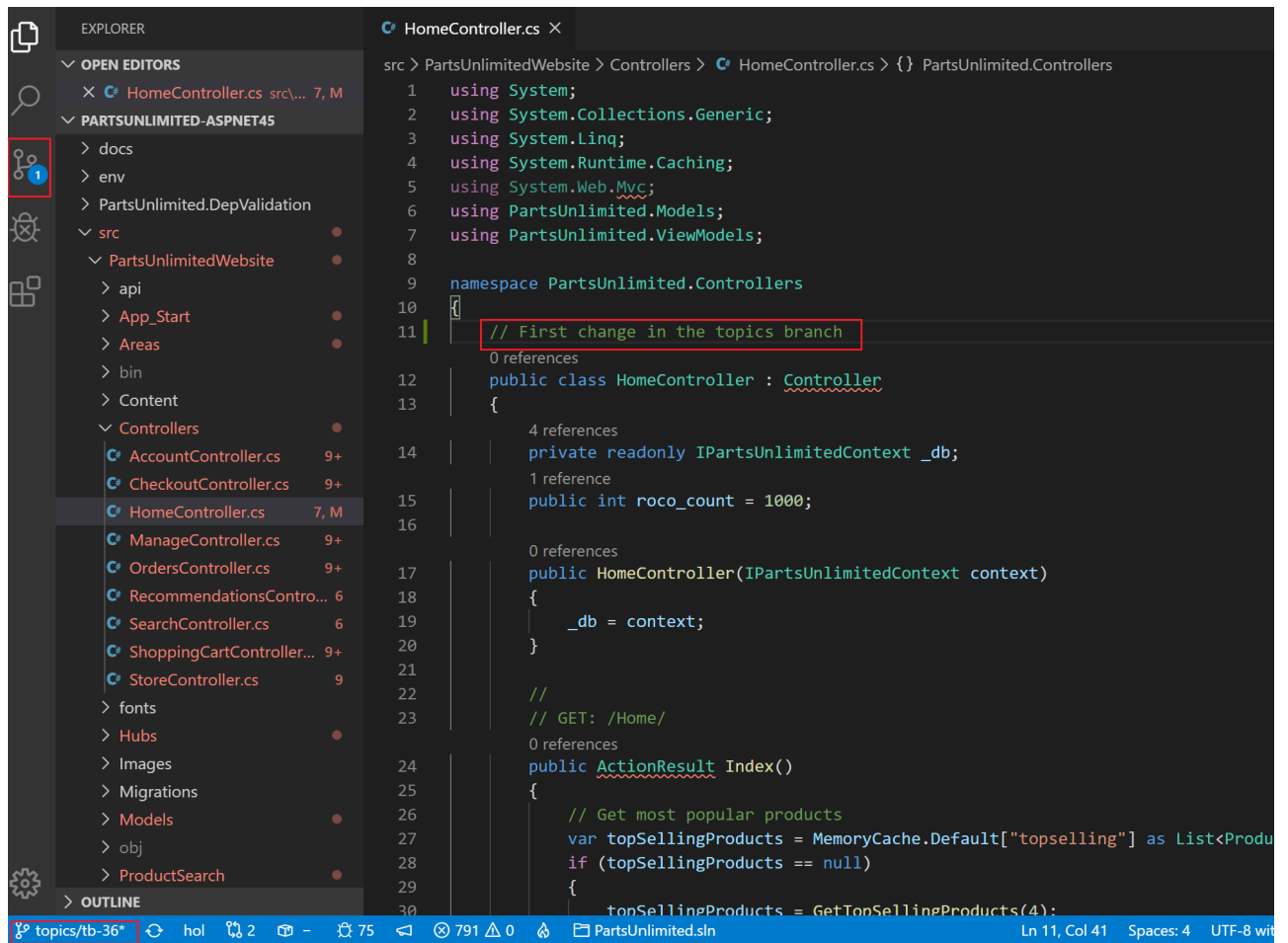


8. You will see that the topics/tb-WID branch doesn't exist locally yet, select "**origin/topics/tb-WID**" and notice the branch changes from master to "**Topics/tb-WID**". This will position you in the topic branch.

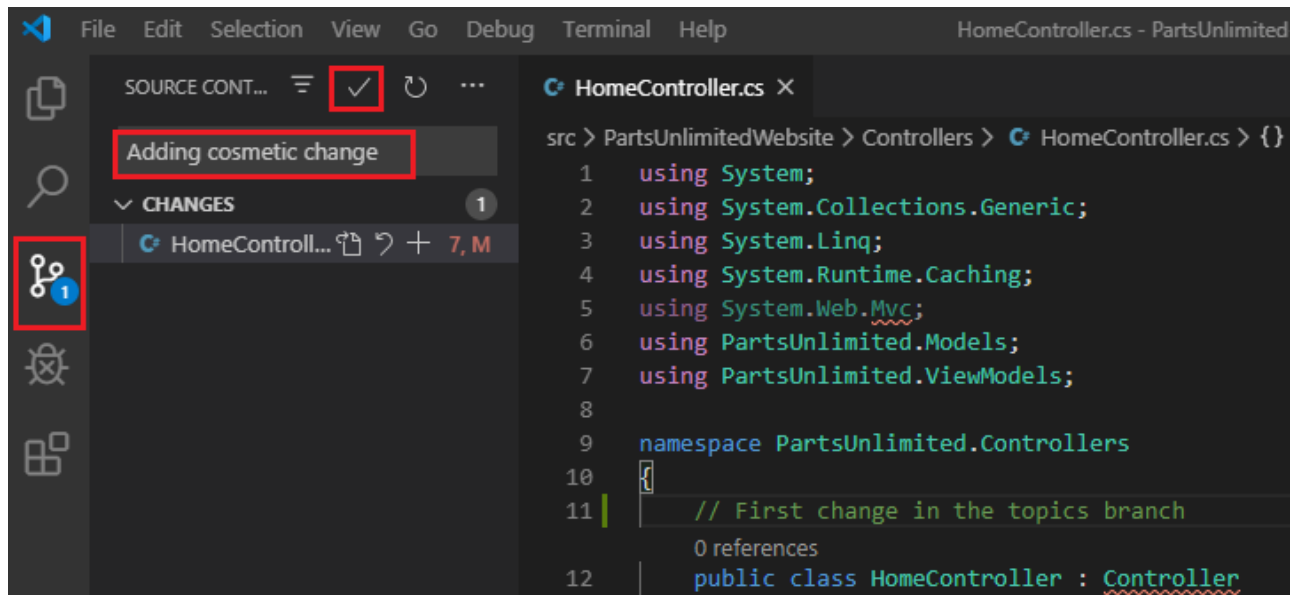
Note: If you do not see the remote branch, you will need to open the terminal in VS Code and run a **git pull**. You might also need to run the command: **git checkout topics/tb-WID**



- Open the **src -> PartsUnlimitedWebsite -> Controllers -> HomeController.cs** file and add a cosmetic change and save the file (**Ctrl + S**). Notice that Visual Studio Code indicates there is one uncommitted change on the task bar.



10. Click the source control icon and add the following comment: "Adding cosmetic change." Click the **check mark** to commit the comment locally. A pop up may appear asking "Would you like to automatically stage all your changes and commit them directly?", Select **Yes**.



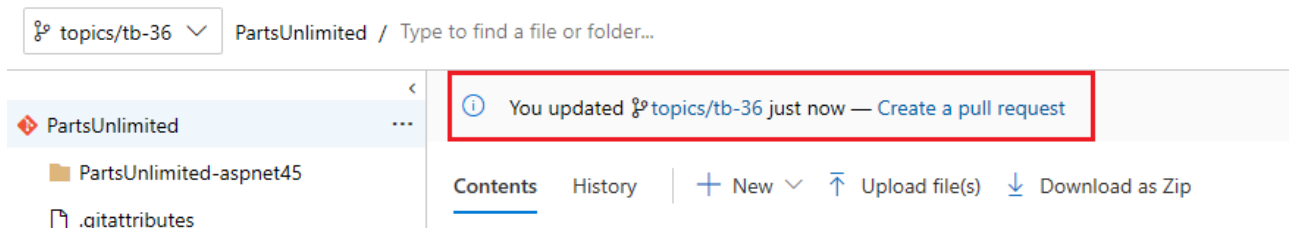
11. Now we need to push the local changes to the remote topics branch, notice the synchronization circle changed to a number with an arrow next to it. Click the icon so the remote branch and local branch stay in sync.



Task 6: Merge changes using pull requests

There are several ways of merging branches. We will choose one that embraces collaboration and add rigor to a DevOps team.

1. Go to the **Repos** -> **Files** section of your Azure DevOps project.
2. The portal indicates that you can create a Pull Request to merge your changes from your topic branches to the master branch after they go through a review.



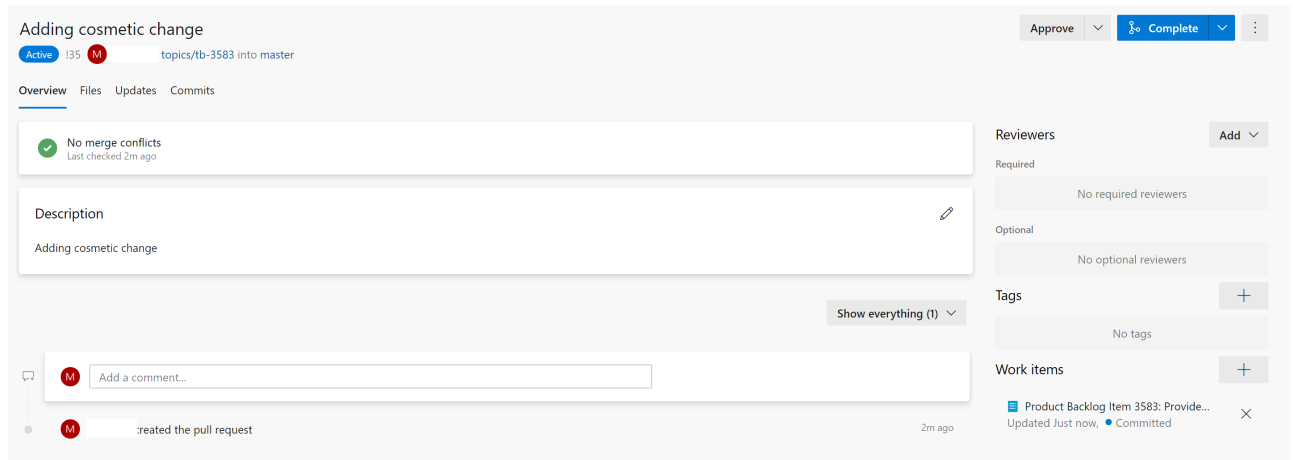
3. Click on **"Create a pull request"** from the info message towards the middle of the screen.
4. The "New Pull Request" form should now be visible.

 A screenshot of the 'New Pull Request' form. At the top, it says 'New Pull Request'. Below that, there are dropdowns for the source branch ('topics/tb-36') and the target branch ('master'), with an 'into' label and a refresh icon. The 'Title' field contains 'Adding cosmetic change'. There is an 'Add label' button. The 'Description' field also contains 'Adding cosmetic change', with a note 'Markdown supported.' below it. A rich text editor toolbar is visible. Below the description, there is a 'Reviewers' section with a search box 'Search users and groups to add as reviewers'. At the bottom, there is a 'Work Items' section with a search box 'Search work items by ID or title'. A blue 'Create' button is at the bottom right.

5. Click on **Create**

Note: This workflow is a very important step in correctly implementing traceability throughout the application lifecycle. The pull request should look like this:

Note: The screenshots of Pull Request are taken with the New Repos pull request experience Preview feature enabled. If this Preview Feature is disabled for you, the UI will be slightly different.



Note: In a normal situation, someone else should be approving the pull request, but for this lab, you will approve your own pull request. We will review this section later in the workshop.

6. Click on **Approve**.
7. And finally click on **Complete**. You will be presented with a dialog asking you to delete the topic branch. This is a normal behavior, and generally we should do so since the work item we were working on is now complete. In the case of this lab, however, we will keep the branch.
8. Uncheck the **Delete topics/tb-WID after merging** checkbox.
9. Click **Complete merge** to complete the pull request.

Complete pull request



Merge type

Merge (no fast forward)



Post-completion options

☒ Complete associated work items after merging

☐ Delete topics/tb-3583 after merging

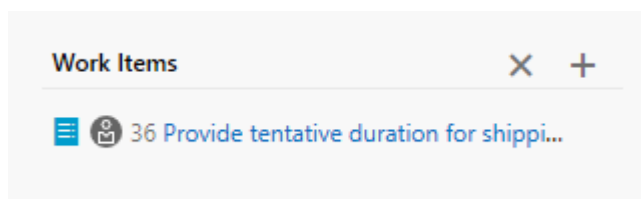
☐ Customize merge commit message

Cancel

Complete merge

Let's view the work item status.




10. Click the Work Item link in the pull request window:







11. Notice how all the previous actions are traced to the work item:

Development

[+ Add link](#)

  [Adding cosmetic change](#)
Created 20 minutes ago,  Completed

  [topics/tb-36](#)
Latest commit 2 hours ago
[Create a pull request](#)

  [c56837a1 Merged PR 3: Adding cosmetic ch...](#)
Created 4 minutes ago

Related Work

Task 7: Create a build definition

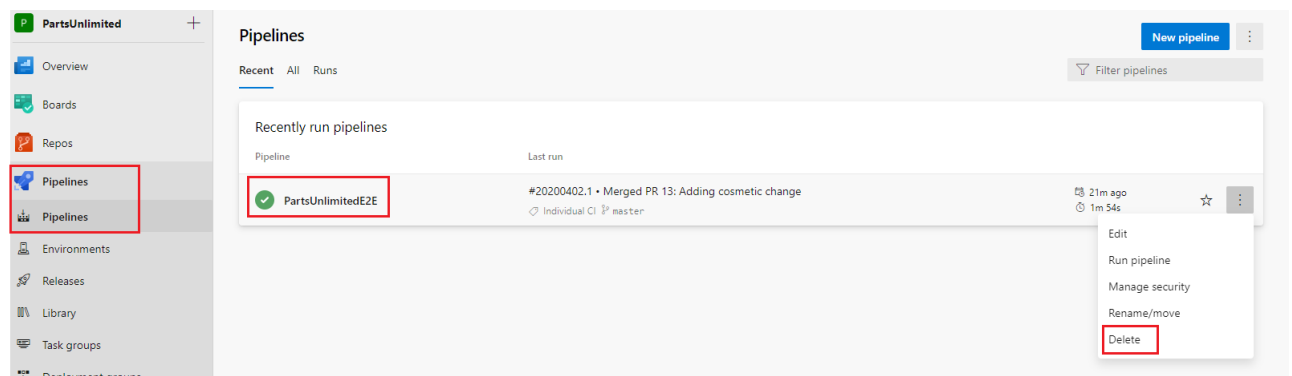
The PartsUnlimited project has currently one main master branch in its only Git repository. In the previous lab, we discussed how we will be creating topic branches for each User Story.

Our goal is to create a Continuous Integration build that will run any time code is pushed into the master or the topic branch. We will then enforce policies that prevent code from being merged into master without an approved pull request.

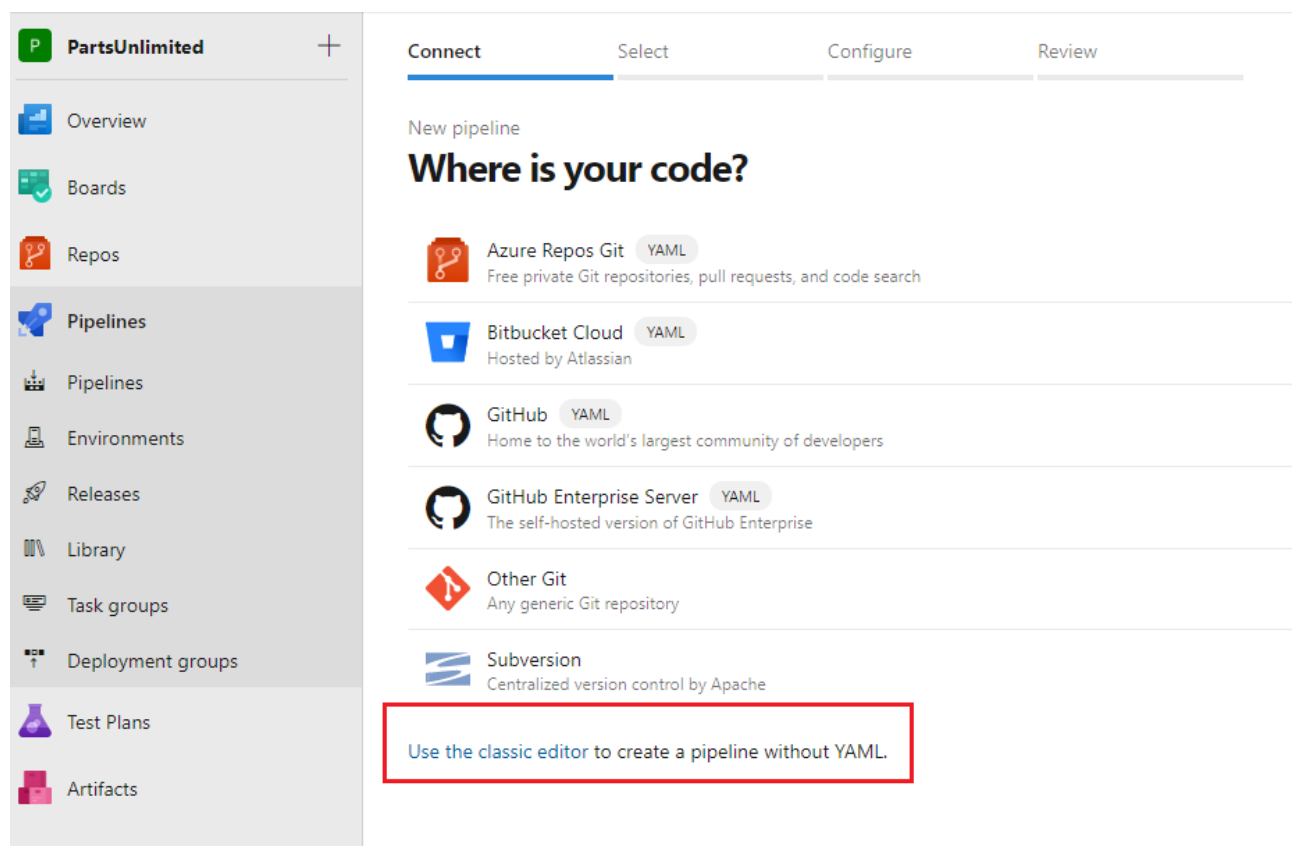
1. First, select **Pipelines** and delete the existing build pipeline

Note: If you have turned off the **Preview Feature** for **Multi-stage pipelines** then you will see Builds option under Pipelines. Although we are working on the classic pipeline in this lab, we will continue with Multi-stage pipeline turned on.

Note: You might have to first delete the Release pipeline from **Releases** to be able to delete the pipeline.









2. Click the "New Pipeline" button to create a new build definition and click **Use the classic editor**.




3. Choose **Azure Repos Git** as the source and pick the Parts Unlimited repository and the master branch.
Click **Continue**


Select a source

 Azure Repos Git	 GitHub	 GitHub Enterprise Server	 Subversion	 Bitbucket Cloud	 Other Git
--	---	---	---	--	--


Team project

 PartsUnlimited ▼

Repository

 PartsUnlimited ▼

Default branch for manual and scheduled builds

 master ▼


Continue

4. On the next screen, pick the **ASP.NET** template and click **Apply**.
Note: we could have started from an empty template, but the ASP.NET template has some tasks that will be needed anyway and simplifies the initial setup.)


Select a template Search


Or start with an [Empty job](#)


Configuration as code


 **YAML**
Looking for a better experience to configure your pipelines using YAML files?
Try the new YAML pipeline creation experience. [Learn more](#)


Featured

 **.NET Desktop**
Build and test a .NET or Windows classic desktop solution.

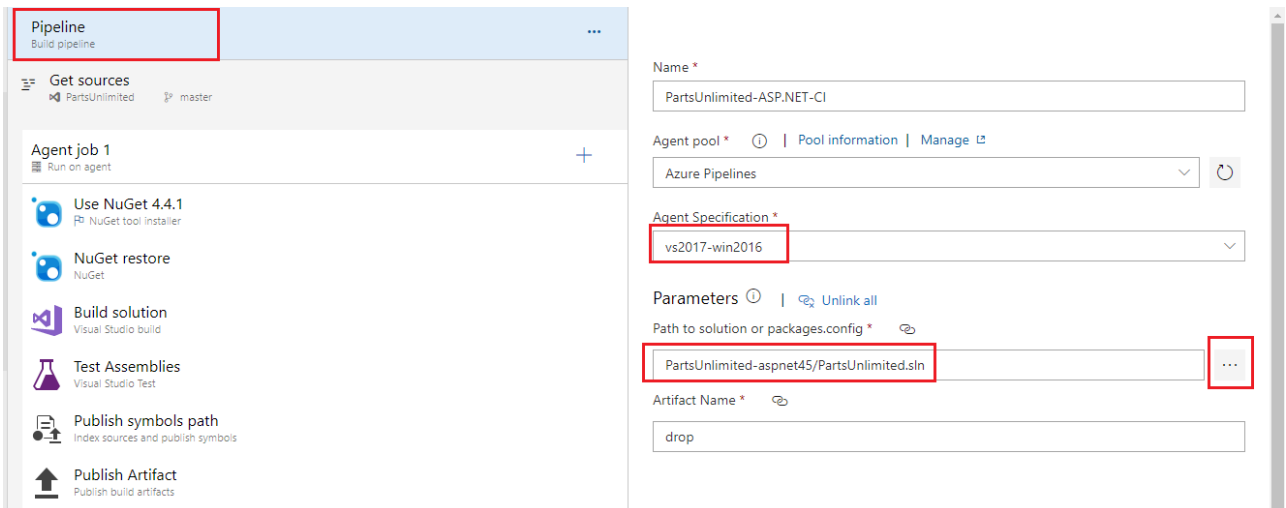
 **Android**
Build, test, sign, and align an Android APK.

 **ASP.NET**
Build and test an ASP.NET web application. Apply

 **Azure Web App for ASP.NET**
Build, package, test, and deploy an ASP.NET Azure Web App.

 **Docker container**
Build a Docker image and push it to a container registry.

5. Under the Pipeline tab, make sure **vs2017-win2016** is selected for the **Agent Specification** and set **Path to solution or packages.config** to **PartsUnlimited-aspnet45/PartsUnlimited.sln**.



The screenshot shows the 'Build pipeline' configuration page. The 'Pipeline' tab is selected. The 'Agent job 1' section shows a list of tasks: 'Get sources', 'Use NuGet 4.4.1', 'NuGet restore', 'Build solution', 'Test Assemblies', 'Publish symbols path', and 'Publish Artifact'. The 'Agent Specification' dropdown is set to 'vs2017-win2016'. The 'Path to solution or packages.config' field is set to 'PartsUnlimited-aspnet45/PartsUnlimited.sln'. The 'Artifact Name' field is set to 'drop'.

6. Looking over the build pipeline, we can see that the template we chose has added some predefined tasks. Remove the **Publish Symbols Path** (right click>remove) since we will not need it for this lab.
7. Next, click the **Triggers** Hub and enable the **Continuous Integration** switch.

Tasks Variables **Triggers** Options Retention History | Save & queue ▾ ...

Continuous integration

hol
Enabled

☒ Enable continuous integration

☐ Batch changes while a build is in progress

Scheduled + Add

8. Back under Tasks, review all the tasks and notice that for **NuGet restore** and **Build solution** tasks, **Solution** field is pointing to the .sln file we configured in the earlier step.

Pipeline
Build pipeline

Get sources
PartsUnlimited master

Agent job 1
Run on agent

Use NuGet 4.4.1
NuGet tool installer

NuGet restore
NuGet

Build solution
Visual Studio build

Test Assemblies
Visual Studio Test

Publish Artifact
Publish build artifacts

Visual Studio build ⓘ Link settings View

Task version 1.* ▾

Display name *
Build solution

Solution * ⓘ
PartsUnlimited-aspnet45/PartsUnlimited.sln

Visual Studio Version ⓘ
Latest

MSBuild Arguments ⓘ
/p:DeployOnBuild=true /p:WebPublishMethod=Package /p:PackageAsS
/p:SkipInvalidConfigurations=true /p:PackageLocation="\$(build.artifact

Platform ⓘ
\$(BuildPlatform)

Configuration ⓘ

9. Next, add a new task called **Copy Files** and move it below Visual Studio Test task.

Tasks Variables Triggers Options Retention History | Save & queue ▾ Discard Summary Queue ...

Pipeline
Build pipeline

Get sources
PartsUnlimited master

Agent job 1
Run on agent

Use NuGet 4.4.1
NuGet tool installer

NuGet restore
NuGet

Build solution
Visual Studio build

Test Assemblies
Visual Studio Test

Publish Artifact
Publish build artifacts

Copy files
Copy files from a source folder to a target folder using patterns matching file paths (not folder paths)

by Microsoft Corporation Learn more

Azure file copy
Copy files to Azure Blob Storage or virtual machines

Add tasks Refresh

copy

Add

10. Modify the Copy Files task using the below text:

Display name: Copy json files

Source Folder: \$(Build.SourcesDirectory)

Contents: **/*.json **Target Folder:** \$(Build.ArtifactStagingDirectory)

The screenshot shows the 'Copy json files' task configuration in the Azure DevOps Pipeline editor. The task is highlighted in the task list on the left. The configuration panel on the right shows the following settings:

- Display name ***: Copy json files
- Source Folder ***: \$(Build.SourcesDirectory)
- Contents ***: **/*.*.json
- Target Folder ***: \$(Build.ArtifactStagingDirectory)

11. Review the remaining **Publish Build Artifacts** task. This task publishes the build artifacts and makes them available to be used later (e.g. in a release pipeline)

12. Change the pipeline name to PartsUnlimited.master.ci.

The screenshot shows the pipeline name 'PartsUnlimited.master.ci' being edited in the Azure DevOps Pipeline editor. The 'Save & queue' button is visible at the top right.

13. Let's test the build pipeline by queueing a new build. Click **Save & queue** near the top.

14. On the screen that appears, leave all the default options and click **Save and run**.

15. Verify that the build is successful by monitoring the build logs by clicking on Agent Job 1

#20200402.1 Merged PR 13: Adding cosmetic change
on PartsUnlimited.master.ci

Cancel

Summary

Manually run by

View 107 changes

Repository and version

Time started and elapsed

Related

Tests and coverage

PartsUnlimited

Just now

1 work items

Get started

master dc4d94d

1m 17s

0 artifacts

Warnings 1

Build solution

PartsUnlimited-aspnet45\src\PartsUnlimitedWebsite\Controllers\ManageController.cs(135,41): Warning CS1998: This async method lacks 'await' operators and will run synchronously. Consider usi...

Jobs

Name	Status	Duration
Agent job 1	Running	1m 16s

16. Here we can view the build log output as our project is being built on a hosted agent.

← Jobs in run #20200402.1
PartsUnlimited.master.ci

Jobs

✓	Agent job 1	1m 27s
✓	Initialize job	6s
✓	Checkout PartsUnlimite...	6s
✓	Use NuGet 4.4.1	1s
✓	NuGet restore	23s
✓	Build solution	26s
✓	Test Assemblies	13s
✓	Copy json files	<1s
✓	Publish Artifact	8s
✓	Post-job: Checkout Pa...	<1s
✓	Finalize Job	<1s
✓	Report build status	<1s

✓ Agent job 1

1 Pool: [Azure Pipelines](#)

2 Image: [vs2017-win2016](#)

3 Agent: [Hosted Agent](#)

4 Started: Today at 4:43 PM

5 Duration: [1m 27s](#)

6

7 ▶ Job preparation parameters

8 ▶ [fx](#) 3 queue time variables used

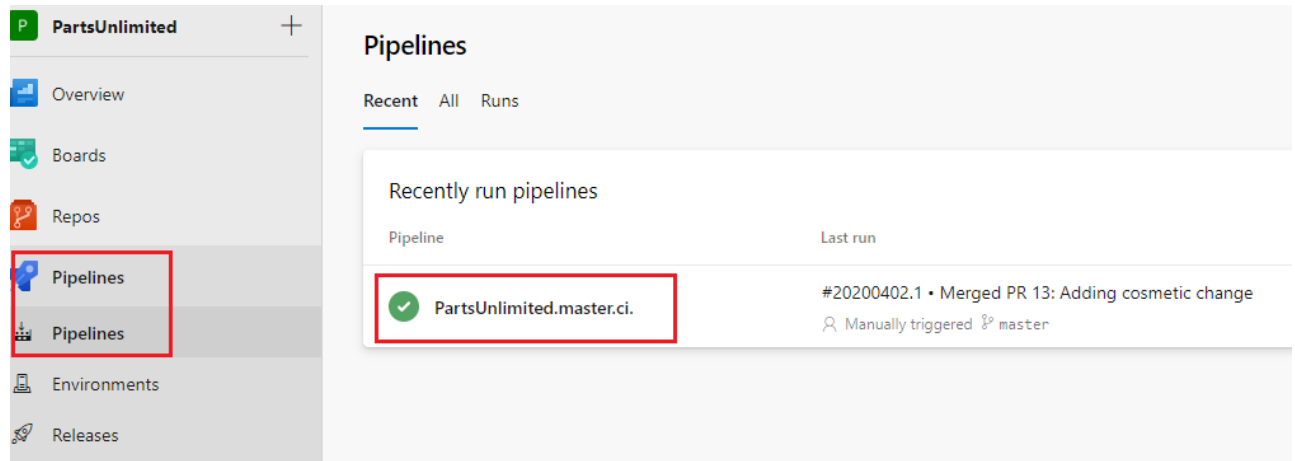
9 [1 artifact](#) produced

10 [87.5% tests](#) passed

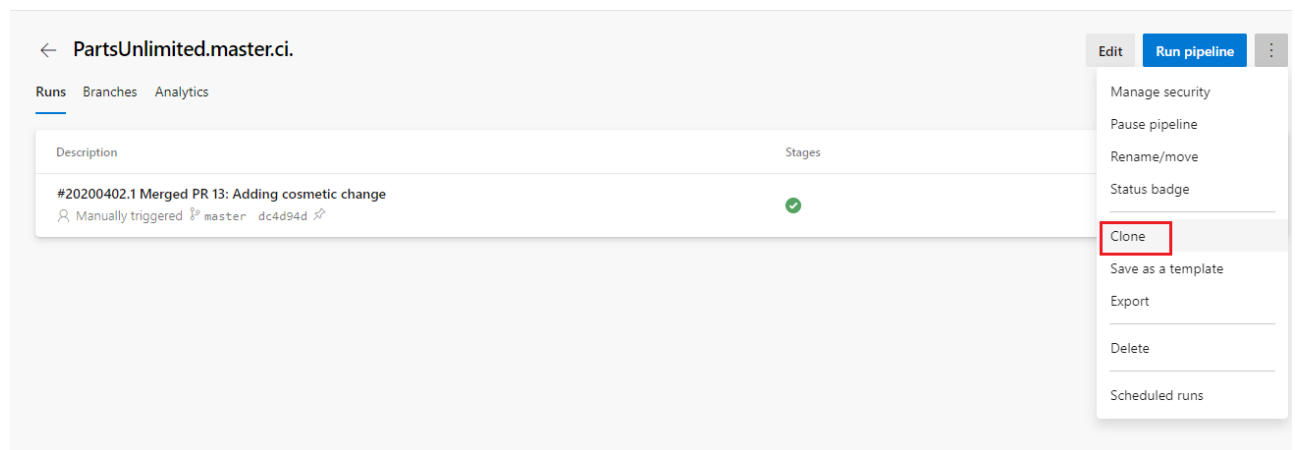
Task 8: Setup a CI build for topic branches

So far only our master branch is setup to build continuously. However, topic branches should also be building continuously.

1. Go to the **Pipelines** hub in your PartsUnlimited project (Pipelines > Pipelines) and select **PartsUnlimited.master.ci**.




2. Click on the menu button (...) towards the top-right near the "Run Pipeline" button. Click **Clone**.



3. In the cloned build pipeline, click on the **Triggers** tab:

Tasks Variables **Triggers** Options Retention History

4. Check the **Enable Continuous Integration** checkbox and change the branch filters to include "topics/*" by typing it into the search box and clicking enter (you may need to press enter twice).

 **PartsUnlimited**

☒ Enable continuous integration

☐ Batch changes while a build is in progress

Branch filters

Type

Include

+ Add

Path filters

+ Add


Branch specification


🔗 master

Mine All branches

topics/*

Press Enter to search in 'All branches'



 **PartsUnlimited**

☒ Enable continuous integration

☐ Batch changes while a build is in progress

Branch filters

Type

Include


+ Add

Path filters

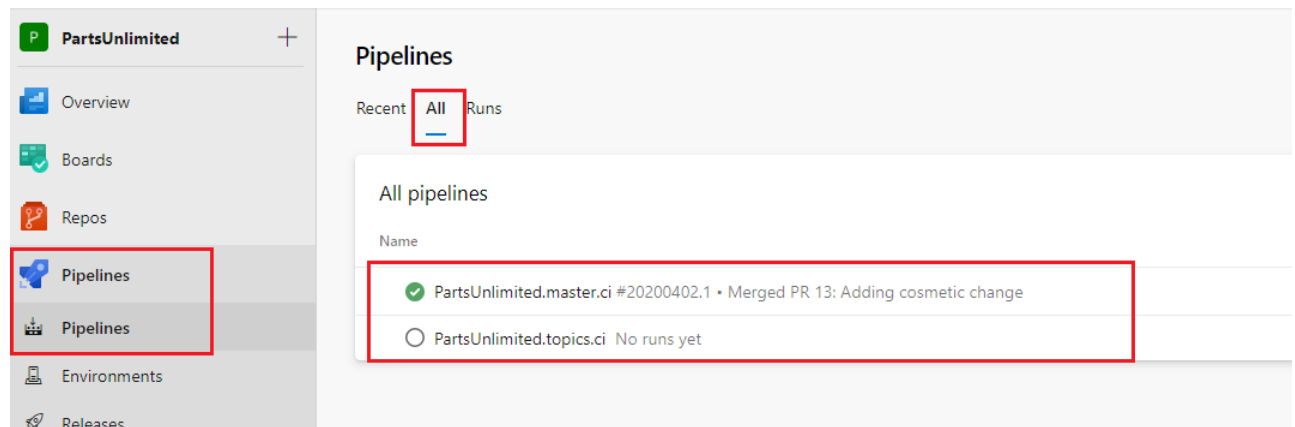
+ Add

Branch specification

🔗 topics/*



5. Save the build pipeline as "PartsUnlimited.topics.ci" by clicking on **Save & queue > Save**. No need to "Save and Queue" at this time. We will run this pipeline in the later task.
6. Navigate back to **Pipelines > Pipelines** and select **All** pipelines. Here you will see both the pipelines.



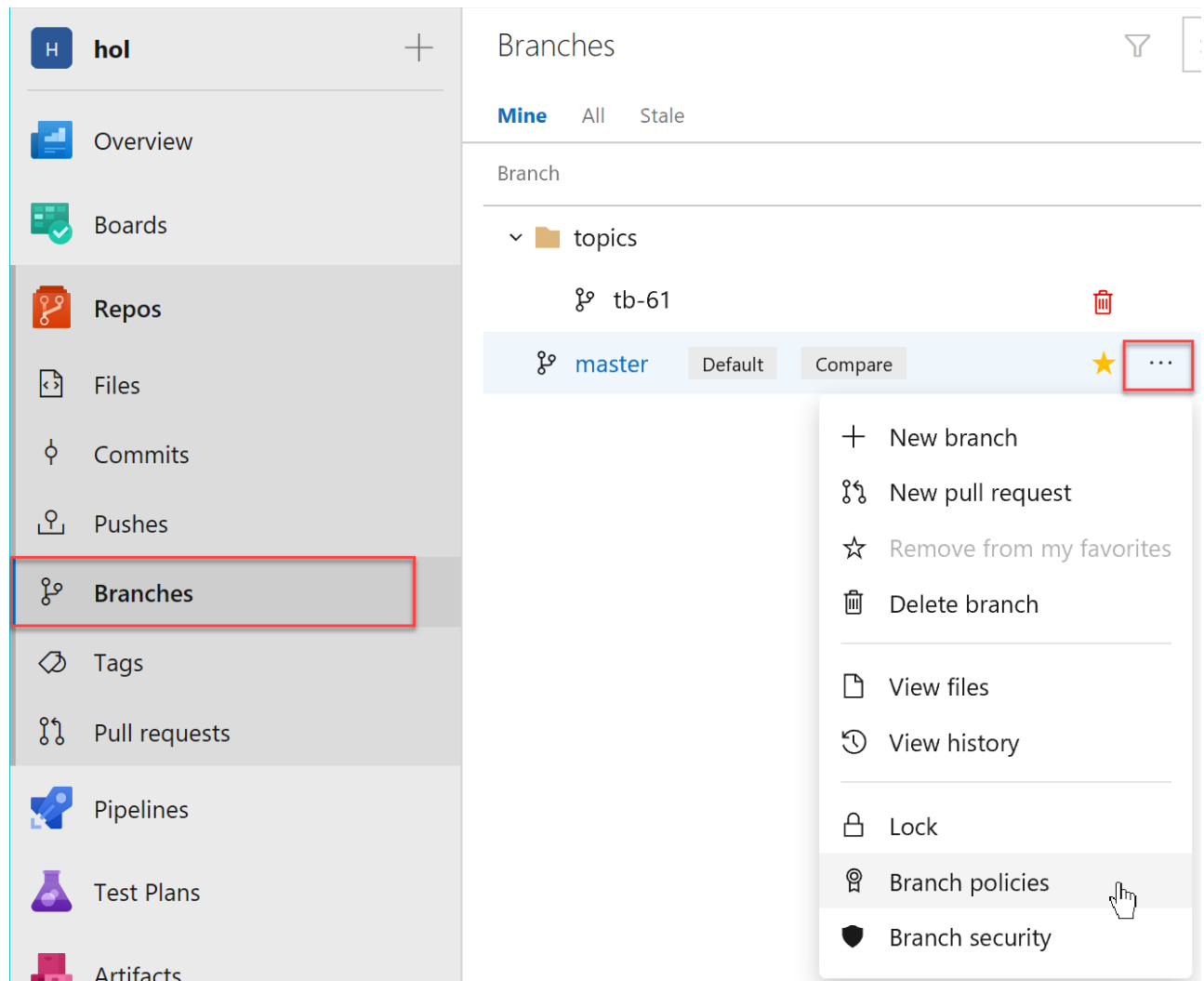
Note: Now each developer working on this project can work locally on their own branches, push those to Azure Repos and before merging those with the master branch with the Pull Request (PR), the topics pipeline will run to ensure that the changes made in that topics branch can build successfully on Azure DevOps.

Task 9: Protect the master branch

We have created a process for developers to work in their local repository, commit and push the changes to Azure Repos where the topics branch will first build and then after the successful build, we can merge changes to the master branch with a Pull Request.

Next, we want to prevent users from pushing code directly to the master branch without a pull request.

1. To implement such a policy, go to **Repos -> Branches**:



2. Click on the menu button (...) for the master branch.
3. Click **Branch policies**.
4. Select the following settings:

Branch policies for master



Save changes



Discard changes

**Require a minimum number of reviewers**

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

Minimum number of reviewers



Allow requestors to approve their own changes



Allow completion even if some reviewers vote to wait or reject



Reset code reviewer votes when there are new changes

**Check for linked work items**

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

Policy requirement



Required

Block pull requests from being completed unless they have at least one linked work item.



Optional

Warn if there are no linked work items, but allow pull requests to be completed.

**Check for comment resolution**

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

**Limit merge types**

Control branch history by limiting the available types of merge when pull requests are completed.

Build validation

Validate code by pre-merging and building pull request changes



Add build policy

Add build policy



Build pipeline *

PartsUnlimited.master.ci



Path filter (optional) 

No filter set

Trigger

☒ Automatic (whenever the source branch is updated)


☐ Manual

Policy requirement

☒ Required
Build must succeed in order to complete pull requests.

☐ Optional
Build failure will not block completion of pull requests.

Build expiration

☒ Immediately when  master is updated

☐ After hours if  master has been updated

☐ Never

Display name

Save

Cancel

5. Click **Save changes**.

Task 10: Approve a Pull Request

The next few steps are purposefully added without many screenshots to allow you to navigate through the user interface with what you've learned so far.

1. Go back to Visual Studio Code.
2. Switch to the master branch by click on bottom-left topics/tb-WID and selecting **master**.
3. Sync all the remote commits.
4. Create a new topics branch from VS Code following the naming convention "topics/tb-[WID]" (use any number as WID for this step).
5. Make sure you are positioned in the new topic branch.
Note: You can tell which branch you are in by looking at the bottom left corner in Visual Studio Code.
6. Modify the HomeController.cs file under **src -> PartsUnlimitedWebsite -> Controllers** and make another cosmetic change and save it.

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.Caching;
5  using System.Web.Mvc;
6  using PartsUnlimited.Models;
7  using PartsUnlimited.ViewModels;
8
9  namespace PartsUnlimited.Controllers
10 {
11     // Second change in the topics branch
12     public class HomeController : Controller
13     {
14         private readonly IPartsUnlimitedContext _db;
15         public int roco_count = 1000;
16
17         public HomeController(IPartsUnlimitedContext context)
18         {
19             _db = context;
20         }
21
22         // GET: /Home/
23         public ActionResult Index()
24         {
25             // Get most popular products
26             var topSellingProducts = MemoryCache.Default.Get("topSellingProducts");
27             if (topSellingProducts == null)
28             {
29                 topSellingProducts = GetTopSellingProducts();
30             }
31             return Json(topSellingProducts);
32         }
33     }
34 }

```

7. Commit and Push this change to the remote repository.



8. Navigate back to Azure DevOps and head to the **Pipelines** hub. Notice how a build is already started for the topics branch.

Azure DevOps myactivateazuredevops / PartsUnlimited / Pipelines



Pipelines

Recent All Runs

Recently run pipelines

Pipeline	Last run
 PartsUnlimited.topics.ci	#20200403.5 • s Individual CI
 PartsUnlimited.master.ci	#20200403.3 • N Individual CI

9. Next, go to the Repos\Files hub in Azure DevOps.

 You updated  **topics/tb-61** just now — [Create a pull request](#)

10. **Create** a pull request to merge the latest commit with the master branch.
11. A new build is now in progress for the master branch. You can view the build by switching back to the Pipelines hub.
Note: This is because of the **Build Policy** we configured in the earlier task.
12. Once the build completes, switch back to the Active Pull Request under **Repos -> Pull Requests -> Active** and select the PR you are working on.
13. Notice the Policies we configured in the earlier task are getting enforced.

Minor cosmetic change

Active 136 M topics/tb-36 into master

Overview Files Updates Commits

1 required check failed
1 optional check not yet run

Work items must be linked

At least 1 reviewer must approve

No merge conflicts
Last checked Just now

Description

Minor cosmetic change

Show everything (1)

Add a comment...

created the pull request Just now

Reviewers

Required

No required reviewers

Optional

No optional reviewers

Tags

No tags

Work items

No work items

14. Try to **Complete** the PR after you have approved it. You will not be able to complete it until you associate a Work Item with this PR. This traceability is a goal for a DevOps team.

Complete pull request

0 of 1 reviewers approved, No work items linked

Merge type

Merge (no fast forward)



Post-completion options

- ☒ Complete associated work items after merging
- ☐ Delete topics/tb-36 after merging
- ☐ Customize merge commit message

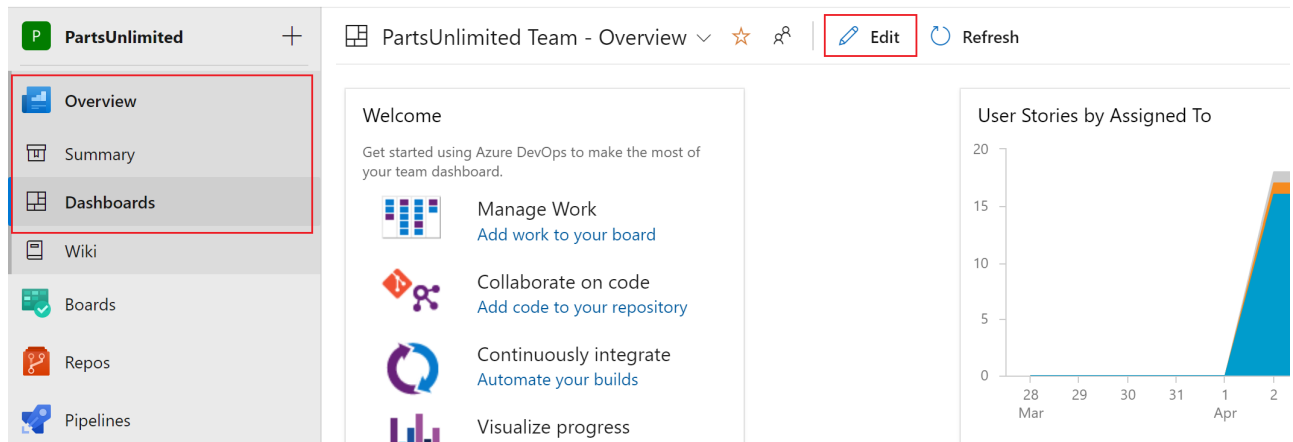
Cancel

Complete merge

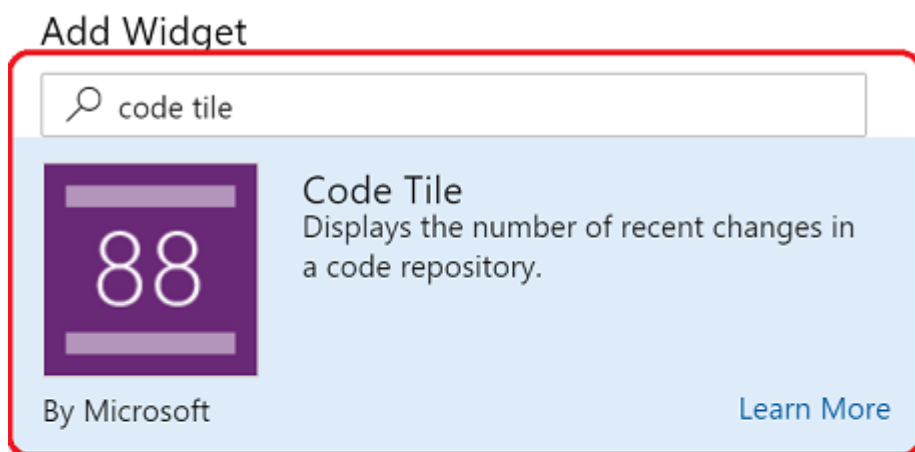
15. Link some work item to this PR and complete the PR. This should trigger another build for the master branch.

Task 11: Monitor Code and Build KPIs

1. Next, navigate to the **PartsUnlimited > Overview > Dashboard > PartsUnlimited Team - Overview dashboard** and click **Edit**.



2. Search for the **Code Tile** widget, click on **Add**.



Don't see a widget? Explore the [Extension Gallery](#)


Add

3. Configure the widget to point to the master branch:


Configuration ✕

Title

Repository

 PartsUnlimited ▼

Branch

 master ▼

Path

▼

4. Add the **Build History** widget and point it to the PartsUnlimited.master.ci build

Configuration ✕

Title

Pipeline

PartsUnlimited.master.ci ▼

Branch

☒ All branches

5. Select **Done Editing**. The team dashboard should now look as follows:

