Lab requirements

- This lab requires **Microsoft Edge** or an Azure DevOps supported browser.
- **Set up an Azure DevOps organization:** If you don't already have an Azure DevOps organization that you can use for this lab, create one by following the instructions available at Create an organization or project collection.
- Visual Studio 2022 Community Edition available from Visual Studio Downloads page. Visual Studio 2022 installation should include **ASP.NET and web development**, **Azure development**, and **.NET Core cross-platform development** workloads.

Lab overview

Azure Artifacts facilitate discovery, installation, and publishing NuGet, npm, and Maven packages in Azure DevOps. It's deeply integrated with other Azure DevOps features such as Build, making package management a seamless part of your existing workflows.

Objectives

After you complete this lab, you will be able to:

- Create and connect to a feed.
- Create and publish a NuGet package.
- Import a NuGet package.
- Update a NuGet package.

Estimated timing: 40 minutes

Instructions

Exercise 0: Configure the lab prerequisites

In this exercise, you will set up the prerequisites for the lab, which include the preconfigured Parts Unlimited team project based on an Azure DevOps Demo Generator template and a Visual Studio configuration.

Task 1: Configure the team project

In this task, you will use Azure DevOps Demo Generator to generate a new project based on the **PartsUnlimited** template.

1. On your lab computer, start a web browser and navigate to Azure DevOps Demo Generator. This utility site will automate the process of creating a new Azure DevOps project within your account that is prepopulated with content (work items, repos, etc.) required for the lab.

2. **Note**: For more information on the site, see https://docs.microsoft.com/en-us/azure/devops/demo-gen.

3. Click **Sign in** and sign in using the Microsoft account associated with your Azure DevOps subscription.

4. If required, on the **Azure DevOps Demo Generator** page, click **Accept** to accept the permission requests for accessing your Azure DevOps subscription.
5. On the **Create New Project** page, in the **New Project Name** textbox, type **Package Management with Azure Artifacts**, in the **Select organization** dropdown list, select your Azure DevOps organization, and then click **Choose template**.
6. On the **Choose a template** page, in the list of templates, click the **PartsUnlimited** template, and then click **Select Template**.
7. Back on the **Create New Project** page, click **Create Project**
8. **Note**: Wait for the process to complete. This should take about 2 minutes. In case the process fails, navigate to your DevOps organization, delete the project, and try again.
9. On the **Create New Project** page, click **Navigate to project**.

Task 2: Configuring the Parts Unlimited solution in Visual Studio
In this task, you will configure Visual Studio to prepare for the lab.

10. Ensure that you are viewing the **Package Management with Azure Artifacts** team project on the Azure DevOps portal.
11. **Note**: You can access the project page directly by navigating to the https://dev.azure.com/<your-Azure-DevOps-account-name>/Package%20Management%20with%20Azure%20Artifacts URL, where the `<your-Azure-DevOps-account-name>` placeholder, represents your account name.
12. In the vertical menu on the left side of the **Package Management with Azure Artifacts** pane, click **Repos**.
13. On the **Files** pane, click **Clone**, select the drop-down arrow next to **Clone in VS Code**, and, in the dropdown menu, select **Visual Studio**.
14. If prompted whether to proceed, click **Open**.
15. If prompted, sign in with the user account you used to set up your Azure DevOps organization.
16. Within the Visual Studio interface, in the **Azure DevOps** pop-up window, accept the default local path and click **Clone**. This will automatically import the project into Visual Studio and open a new web browser tab displaying the Migration Report page.
17. **Note**: In the **Review Project and Solution Changes** dialog box, review the warnings about unsupported project types and click **OK**.
18. Close the web browser tab displaying the Migration Report page.
19. Leave Visual Studio window open for use in your lab.

Exercise 1: Working with Azure Artifacts
In this exercise, you will learn how to work with Azure Artifacts by using the following steps:

- create and connect to a feed.

- create and publish a NuGet package.
- import a NuGet package.
- update a NuGet package.

Task 1: Creating and connecting to a feed
In this task, you will create and connect to a feed.

20. In the web browser window displaying your project settings in the Azure DevOps portal, in the vertical navigational pane, select **Artifacts**.
21. With the **Artifacts** hub displayed, click **+ Create feed** at the top of the pane.
22. **Note**: This feed will be a collection of NuGet packages available to users within the organization and will sit alongside the public NuGet feed as a peer. The scenario in this lab will focus on the workflow for using Azure Artifacts, so the actual architectural and development decisions are purely illustrative. This feed will include common functionality that can be shared across projects in this organization.
23. On the **Create new feed** pane, in the **Name** textbox, type **PartsUnlimitedShared**, in the **Scope** section, select the **Organization** option, leave other settings with their default values, and click **Create**.
24. **Note**: Any user who wants to connect to this NuGet feed must configure their environment.
25. Back on the **Artifacts** hub, click **Connect to feed**.
26. On the **Connect to feed** pane, in the **NuGet** section, select **Visual Studio** and, on the **Visual Studio** pane, copy the **Source** url.
27. Switch back to the **Visual Studio** window.
28. In the Visual Studio window, click **Tools** menu header, in the dropdown menu, select **NuGet Package Manager** and, in the cascading menu, select **Package Manager Settings**.
29. In the **Options** dialog box, click **Package Sources** and click the plus sign to add a new package source.
30. At the bottom of the dialog box, in the **Name** textbox, replace **Package source** with **PartsUnlimitedShared** and, in the **Source** textbox, paste the URL you copied in the Azure DevOps portal.
31. Click **Update** and then click **OK** to finalize the addition.
32. **Note**: Visual Studio is now connected to the new feed.
33. Close and reopen the other Visual Studio instance you used for cloning the PartsUnlimited repository, to account for the artifact source update and open the **PartsUnlimited** solution. You will need it in the third task of this exercise.

Task 2: Creating and publishing a NuGet package
In this task, you will create and publish a NuGet package.

34. In the Visual Studio window you used to configure the new package source, in the main menu, click **File**, in the dropdown menu, click **New** and then, in the cascading menu, click **Project**.

35. **Note**: We will now create a shared assembly that will be published as a NuGet package so that other teams can integrate it and stay up to date without having to work directly with the project source.

36. On the **Recent project templates** page of the **Create a new project** pane, use the search textbox to locate the **Class Library (.NET Framework)** template, select the template for C#, and click **Next**.
37. On the **Class Library (.NET Framework)** page of the **Create a new project** pane, specify the following settings and click **Create**:

Setting	Value
Project name	**PartsUnlimited.Shared**
Location	accept the default value
Solution	**Create new solution**
Solution name	**PartsUnlimited.Shared**
Framework	**.NET Framework 4.5.1**

38. **Note**: Make sure not to select **.NET Standard**.

39. Within the Visual Studio interface, in the **Solution Explorer** pane, right-click **Class1.cs**, in the right-click menu, select **Delete**, and, when prompted for confirmation, click **OK**.
40. Within the Visual Studio interface, in the **Solution Explorer** pane, right-click the **PartsUnlimited.Shared** project node and select **Properties**.
41. Within the **PartsUnlimited.Shared** properties pane, verify that the **Target framework** is set to **.NET Framework 4.5.1**.
42. Press **Ctrl+Shift+B** to build the project.

43. **Note**: In the next task we'll use **NuGet.exe** to generate a NuGet package directly from the built project, but it requires the project to be built first.

44. Switch to the web browser displaying the Azure DevOps portal.
45. Navigate to the **Connect to feed** pane, in the **NuGet** section and select **NuGet.exe**. This will display the **NuGet.exe** pane.
46. On the **NuGet.exe** pane, click **Get the tools**.
47. On the **Get the tools** pane, click the **Download the latest NuGet** link. This will automatically open another browser tab displaying the **Available NuGet Distribution Versions** page.
48. On the **Available NuGet Distribution Versions** page, select nuget.exe version **v5.5.1** and download the executable to the local **Downloads** folder.

49. Switch to the **Visual Studio** window. In the **Solution Explorer** pane, right-click the **PartsUnlimited.Shared** project node and, in the right-click menu, select **Open Folder in File Explorer**.
50. Within the File Explorer window, move the downloaded **nuget.exe** file from the **Downloads** folder into the folder containing the **.csproj** file.
51. In the same File Explorer window, select the **File** menu header, in the dropdown menu, select **Open Windows PowerShell**, and, in the cascading menu, click **Open Windows PowerShell as administrator**.
52. In the **Administrator: Windows PowerShell** window, run the following to create a **.nupkg** file from the project.
53. **Note**: This is a shortcut to package the NuGet bits for deployment. NuGet is highly customizable. To learn more, refer to the NuGet package creation page.

54. CodeCopy

```
55.      ./nuget.exe pack ./PartsUnlimited.Shared.csproj
```

56. **Note**: Disregard any warnings displayed in the **Administrator: Windows PowerShell** window.
57. **Note**: NuGet builds a minimal package based on the information it is able to identify from the project. For example, note that the name is **PartsUnlimited.Shared.1.0.0.nupkg**. That version number was retrieved from the assembly.
58. Switch back to the **Visual Studio** window, in the **Solution Explorer** pane, expand the **PartsUnlimited.Shared\Properties** node, click **AssemblyInfo.cs** to open it in the central pane of the window, and review its content.
59. **Note**: The **AssemblyVersion** attribute specifies the version number to build into the assembly. Each NuGet release requires a unique version number, so if we continue to use this method for creating packages, we need to increment this before the build.
60. Switch to the **Administrator: Windows PowerShell** window and run the following to publish the package to the **PartsUnlimitedShared** feed:
61. **Note**: You need to provide an **API Key**, which can be any non-empty string. We're using **AzDO** here. When prompted, sign in to your Azure DevOps organization.

62. CodeCopy

```
63.      ./nuget.exe push -source "PartsUnlimitedShared" -ApiKey AzDO
   PartsUnlimited.Shared.1.0.0.nupkg
```

64. Switch to the web browser window displaying the Azure DevOps portal and, in the vertical navigational pane, select **Artifacts**.
65. On the **Artifacts** hub pane, click the dropdown list in the upper left corner and, in the list of feeds, select the **PartsUnlimitedShared** entry.
66. **Note**: The **PartsUnlimitedShared** feed should include the newly published NuGet package.
67. Click the NuGet package to display its details.

Task 3: Importing a NuGet package
In this task, you will import a NuGet package.

68. Switch to the **Visual Studio** window displaying the **Parts Unlimited** solution.
69. In the **Solution Explorer** pane, right-click the **References** node under the **PartsUnlimitedWebsite** project and, in the right-click menu, select **Manage NuGet Packages**. This will open the **NuGet: PartsUnlimitedWebsite** tab in the central pane of the window.
70. In the **NuGet: PartsUnlimitedWebsite** pane, click the **Browse** tab and, in the **Package source** drop-down list in the upper right corner of the pane, select **PartsUnlimitedShared**.
71. **Note**: The list of packages will consist only of the single package you just added.
72. Select the package and, in the **PartsUnlimited.Shared** pane, click **Install** to add it to the project.
73. When prompted, in the **Preview Changes** dialog box, click **OK**.
74. Press **Ctrl+Shift+B** to build the project and verify that the build completed successfully.
75. **Note**: The NuGet package doesn't add any value yet, but we managed to verify that the workflow works as intended.

Task 4: Updating a NuGet package
In this task, you will update a NuGet package.

76. Switch to the **Visual Studio** window that has the **PartsUnlimited.Shared** project open (containing the NuGet source project).
77. In the **Solution Explorer** pane, right-click the **PartsUnlimited.Shared** project node, in the right-click menu, select **Add** and, in the cascading menu, select **New Item**.
78. In the **Add New Item - PartsUnlimitedShared** dialog box, in the list of **Visual C# items**, ensure that the **Class** template is selected, in the **Name** textbox at the bottom of the dialog box, type **TaxService.cs**, and click **Add** to add the class.
79. **Note**: We will pretend that tax calculation will be consolidated into this shared class and managed centrally so that other teams can simply work with the NuGet package.

80. In the central pane, in the code of the **TaxService.cs** class, replace the existing definition of the class with the following code and save the file:

81. CodeCopy

```
82.      namespace PartsUnlimited.Shared
    {
        public class TaxService
        {
            static public decimal CalculateTax(decimal taxable,
   string postalCode)
            {
                return taxable * (decimal).1;
            }
        }
    }
```

83. **Note**: Since we're updating the assembly (and package), we need to update the assembly version.

84. In the Visual Studio window, in the central pane, click the **AssemblyInfo.cs** tab to display the content of the corresponding file.
85. In the **AssemblyInfo.cs** file, change the `[assembly: AssemblyVersion("1.0.0.0")]` to `[assembly: AssemblyVersion("1.1.0.0")]` and save the file.
86. Press **Ctrl+Shift+B** to build the project.
87. Switch to the **Administrator: Windows PowerShell** window and run the following command to repackage the NuGet package.

88. **Note**: The new package will have the updated version number.

89. CodeCopy

```
90.      ./nuget.exe pack PartsUnlimited.Shared.csproj
```

91. From the **Administrator: Windows PowerShell** window, run the following command to publish the updated package.

92. **Note**: The published artifact version number is changed to reflect the package version update.

93. CodeCopy

94. `./nuget.exe push -source "PartsUnlimitedShared" -ApiKey AzDO PartsUnlimited.Shared.1.1.0.nupkg`

95. Switch to the web browser window displaying the Azure DevOps portal with the **PartsUnlimitedShared 1.0.0** artifact pane.
96. On the **PartsUnlimitedShared 1.0.0** artifact pane, click the **Versions** tab and verify that it includes versions **1.0.0** and **1.1.0**.
97. Switch back to the **Visual Studio** window displaying the **PartsUnlimited** project.
98. In the **Solution Explorer** pane, navigate to and select **PartsUnlimitedWebsite\Utils\DefaultShippingTaxCalculator.cs**. This will automatically open the file in the central pane of the window.
99. In the code of the **DefaultShippingTaxCalculator.cs** file, locate the call to **CalculateTax** on line **20** and replace `tax = CalculateTax(subTotal + shipping, postalCode);` with `tax = PartsUnlimited.Shared.TaxService.CalculateTax(subTotal + shipping, postalCode);`
100. **Note**: The original code called a method internal to this class, so the code we're adding to the beginning of the line is redirecting it to code from our NuGet assembly. However, since this project hasn't updated the NuGet package yet, it's still referencing 1.0.0.0 and doesn't have these new changes available, so the code will not build properly.
101. In the **Solution Explorer** pane, right-click the **References** node and, in the right-click menu, select **Manage NuGet Packages**.
102. **Note**: NuGet is aware of our update, as indicated by the content of the **Updates** tab.
103. In the **NuGet: PartsUnlimitedWebsite** pane, click the **Updates** tab, in the search textbox, type **PartsUnlimited.Shared** and, on the right-hand side of the pane, next to the **Version: Latest stable 1.1.0** dropdown list, click **Update** to install the new version.
104. **Note**: There may be many NuGet updates available, but you should only need to update **PartsUnlimited.Shared**. Note that it may take a little while for the package to become completely available for updating. If you get an error, wait a moment and try again.
105. When prompted, in the **Preview Changes** dialog box, click **OK**.
106. Press the **F5** key to build and run the site. Verify that it works as expected.

Review

In this lab, you learned how to work with Azure Artifacts by using the following steps:

- Created and connect to a feed.
- Created and publish a NuGet package.
- Imported a NuGet package.
- Updated a NuGet package.