

APL64 Cross-platform Component as an Azure Function Application

Contents

Overview	1
Download the Example Application	2
Create the Azure Function Application in Visual Studio	2
Create the APL64 CPC.....	14
Publish the APL64 CPC Nuget package	18
Set-up the Visual Studio Package Sources	19
Install the APL64 CPC Nuget package into the Project	20
Update the SquareRootCalc.cs Source Code	22
Add the APL64CpcSquareRoot Project Dependency.....	23
Update the AzureFunctions.cs Code File.....	24
Run the Application.....	29
Azure Function Deployment	33
Learn More	33

Overview

An APL64 Cross-platform component (CPC) can be the basis of a ‘server-less’ web application. APL64 programmer-defined functions support the algorithmic, calculation and other selected portions of the application, such as data persistence, output generation, etc.

The application is ‘server-less’ because the application will run only when there is a user request for it. The application does not require a ‘24-7’ dedicated web server running even when there are no users. The cost to expose a ‘server-less’ application is generally based on the actual resources used to run the application by an end user.

Such an application is automatically scalable from a no, a few, or many end users, without excess costs for ‘idle’ web servers.

The technology to create a ‘server-less’ application is available in [Microsoft Azure Function](#), [Google Cloud](#) and [AWS Lambda](#). This example illustrates an APL64 CPC exposed as a Microsoft Azure function. In these environments, the same APL64 CPC can be transparently used.

[Microsoft Azure](#) is a cloud-based application platform. An [Azure Function](#) is server-less application with minimal infrastructure and production costs dependent on actual client usage. In this example a Azure Function project is created with calculation functionality provided by an APL64 cross-platform component (CPC).

The event which causes the Azure function to run is a ‘trigger’. An Azure function can be [triggered](#) in many ways, including timer, http web request, mobile application, another web server, a request from a different Azure Function, a desktop application, etc.

In this example the application is triggered by a client’s web request. The web request is initiated by browsing to a url for the Azure function application. The web request triggers the first Azure function in this example to present an html page for application-specific user input. When the user submits the input, another Azure function calculates the desired results, and prepares the browser response for the user.

In this example the APL64 cross-platform component (CPC) exposes two APL64 programmer-developed, public functions which provide the application’s calculation functionality.

Developing an Azure Function application involves APL64 and Microsoft .Net technology. In this example Microsoft Visual Studio 2022 is used to develop the application. Microsoft [Visual Studio Community version](#) and [Visual Studio Code](#) are available at no cost.

Download the Example Application

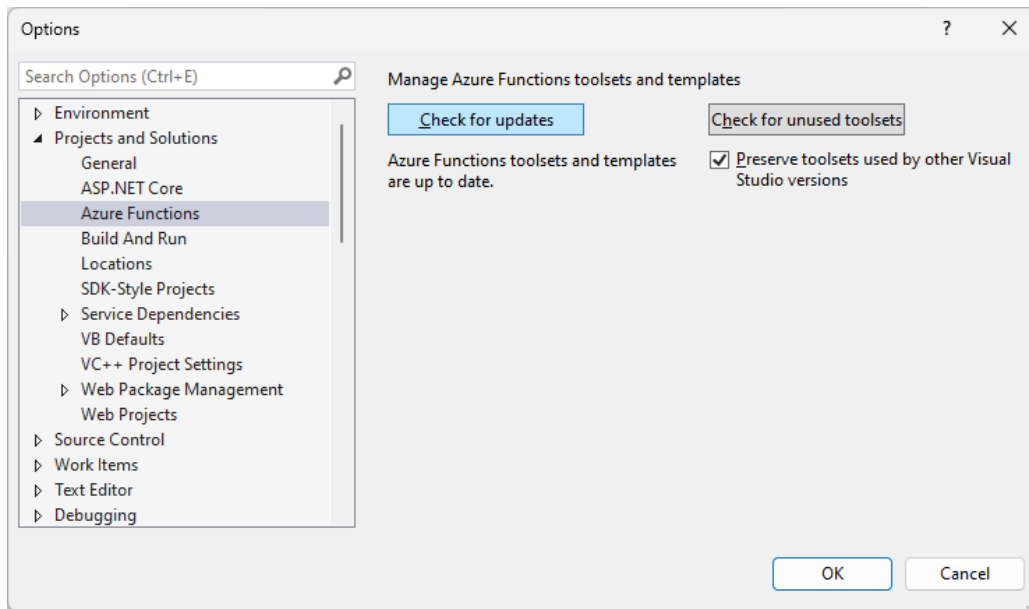
Download the example source code here:

<http://apl2000.com/APL64/UserDocumentation/APL64CpcAzureFunction.zip>

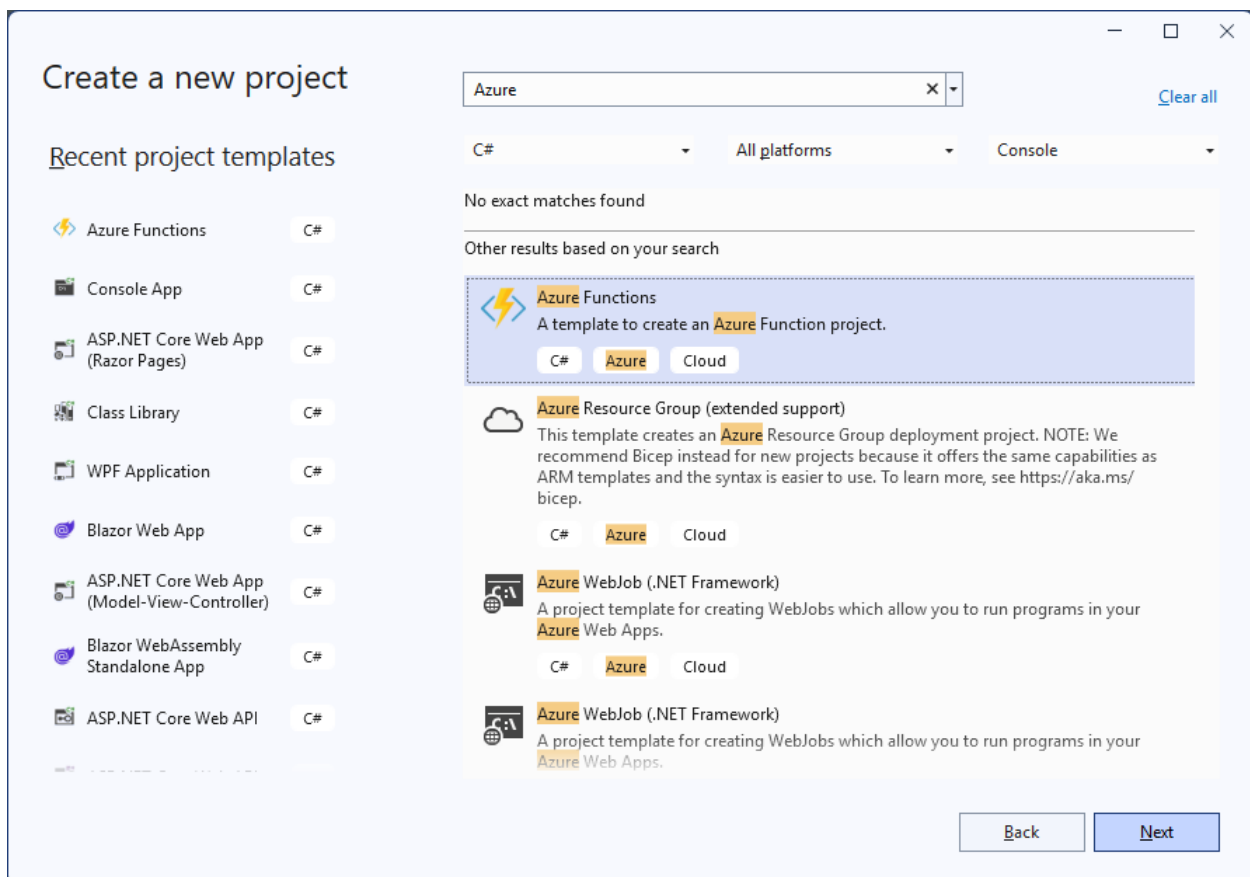
Expand the zip-format file to c:\, so that it is easy to follow this document. The compiled portions of the application are not included in the zip-format file. To be able to run the application follow the directions in each step of this document. Some steps have already been applied to the downloaded application. The document instructions provide the steps to a complete sample application without downloading the zip-format file.

Create the Azure Function Application in Visual Studio

Open Visual Studio, without creating a project, and check that the version is at least ‘Version 17.12.4’ and that the Visual Studio project templates are up to date:



In Visual Studio 2022, create a new project using the Azure Functions template:



Complete the configuration dialog:

Configure your new project

Azure Functions C# Azure Cloud

Project name

APL64CpcAzureFunction

Location

C:\

Solution name ⓘ

APL64CpcAzureFunction

☐ Place solution and project in the same directory

Project will be created in "C:\APL64CpcAzureFunction\APL64CpcAzureFunction\"

Back Next

Complete the additional information dialog. In this example an HTTP Trigger is illustrated. The 'Anonymous' authorization level is selected for design and debugging purposes. In a production environment a more secure authorization level is recommended.

—□×

Additional information

Azure Functions

C#AzureCloud

Functions worker ⓘ

.NET 8.0 Isolated (Long Term Support) ▾

Function ⓘ

Http trigger ▾

☐ Enlist in .NET Aspire orchestration (Preview) ⓘ

☒ Use Azurite for runtime storage account (AzureWebJobsStorage) ⓘ

☐ Enable container support ⓘ

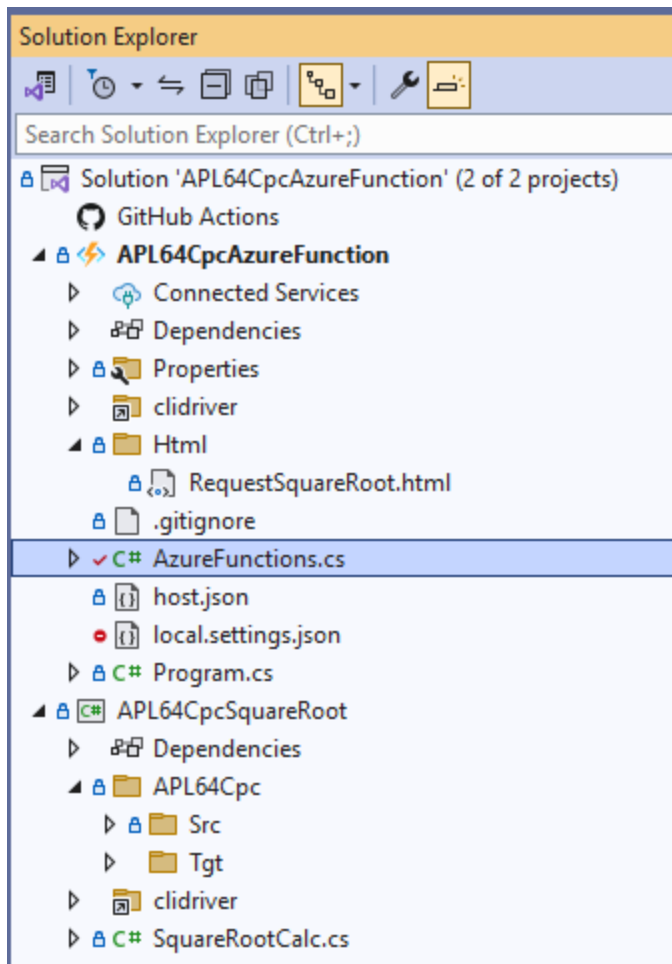
Authorization level ⓘ

Anonymous ▾

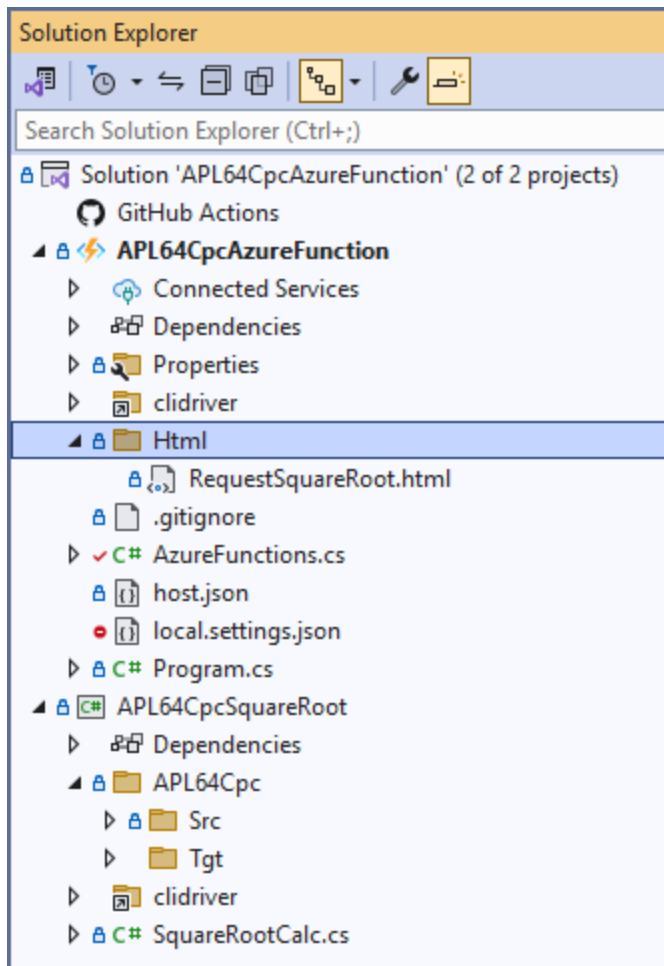
Back

Create

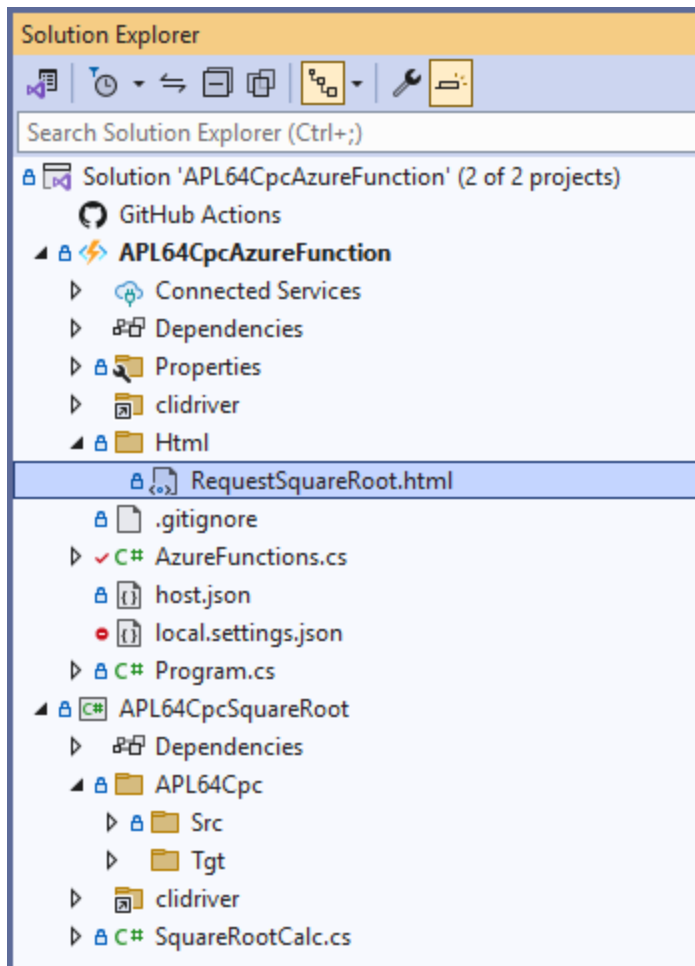
Rename the 'Function1.cs' and included class source code in the APL64CpcAzureFunction project to 'AzureFunctions.cs':



Add the Html folder to the APL64CpcAzureFunction project:



Add the 'RequestSquareRoot.html' web page into the Html folder of the APL64CpcAzureFunction project:

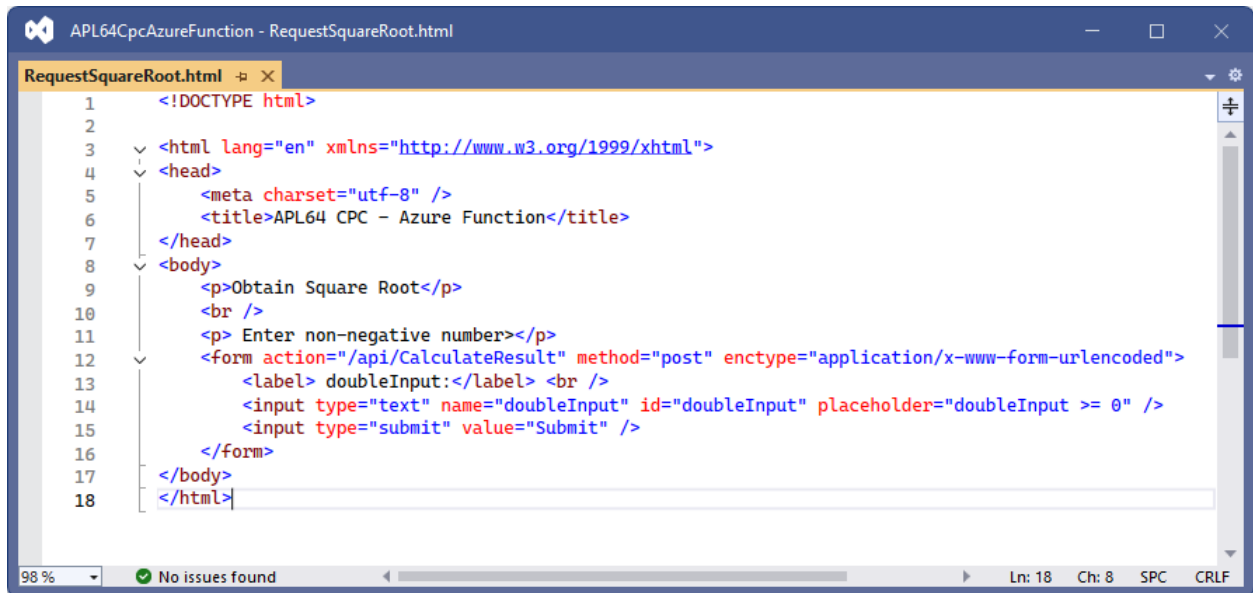


Replace the source code of the 'RequestSquareRoot.html' web page into the Html folder of the APL64CpcAzureFunction project with:

```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta charset="utf-8" />
  <title>APL64 CPC - Azure Function</title>
</head>
<body>
  <p>Obtain Square Root</p>
  <br />
  <p>Enter non-negative number</p>
  <form action="/api/CalculateResult" method="post" enctype="application/x-www-form-urlencoded">
    <label>doubleInput:</label> <br />
    <input type="text" name="doubleInput" id="doubleInput" placeholder="doubleInput >= 0" />
    <input type="submit" value="Submit" />
  </form>
```

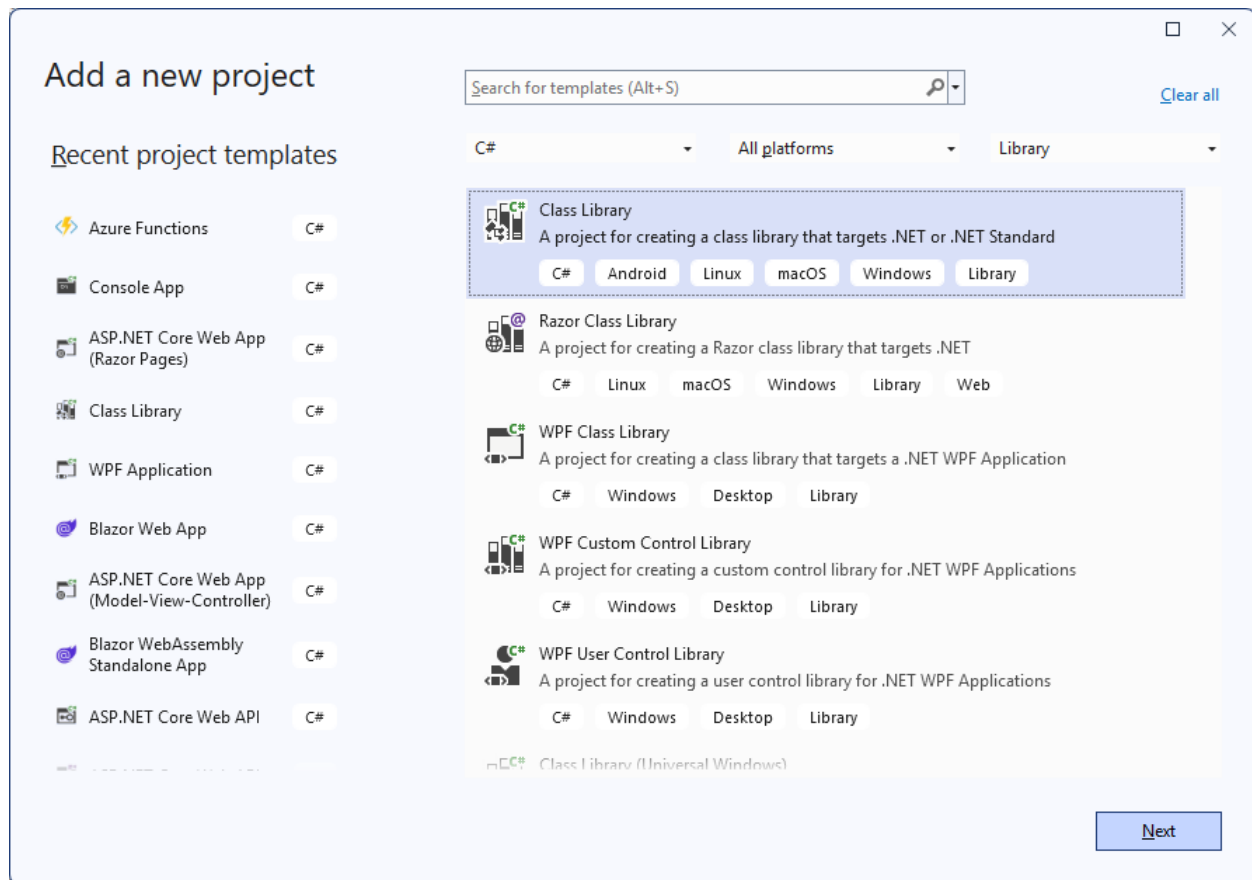
```
</body>
</html>
```



```
1 <!DOCTYPE html>
2
3 <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5   <meta charset="utf-8" />
6   <title>APL64 CPC - Azure Function</title>
7 </head>
8 <body>
9   <p>Obtain Square Root</p>
10  <br />
11  <p>Enter non-negative number</p>
12  <form action="/api/CalculateResult" method="post" enctype="application/x-www-form-urlencoded">
13    <label> doubleInput:</label> <br />
14    <input type="text" name="doubleInput" id="doubleInput" placeholder="doubleInput >= 0" />
15    <input type="submit" value="Submit" />
16  </form>
17 </body>
18 </html>
```

Set the Properties | Build Action of the 'RequestSquareRoot.html' file to 'Content' and its 'Copy to Output Directory' setting to 'Copy always'. These settings assure that when the Azure function is run, the 'RequestSquareRoot.html' page text can be served to the user as a 'WebResponse'.

Add a 'class library' project to the solution, to contain the APL64 CPC:



Configure the class library project:

□

×

Configure your new project

Class Library

C#AndroidLinuxmacOSWindowsLibrary

Project name

APL64CpcSquareRoot

Location

C:\APL64CpcAzureFunction

...

Project will be created in "C:\APL64CpcAzureFunction\APL64CpcSquareRoot\"

BackNext

Provide the additional information for the class library project:

□

×

Additional information

Class Library

C#AndroidLinuxmacOSWindowsLibrary

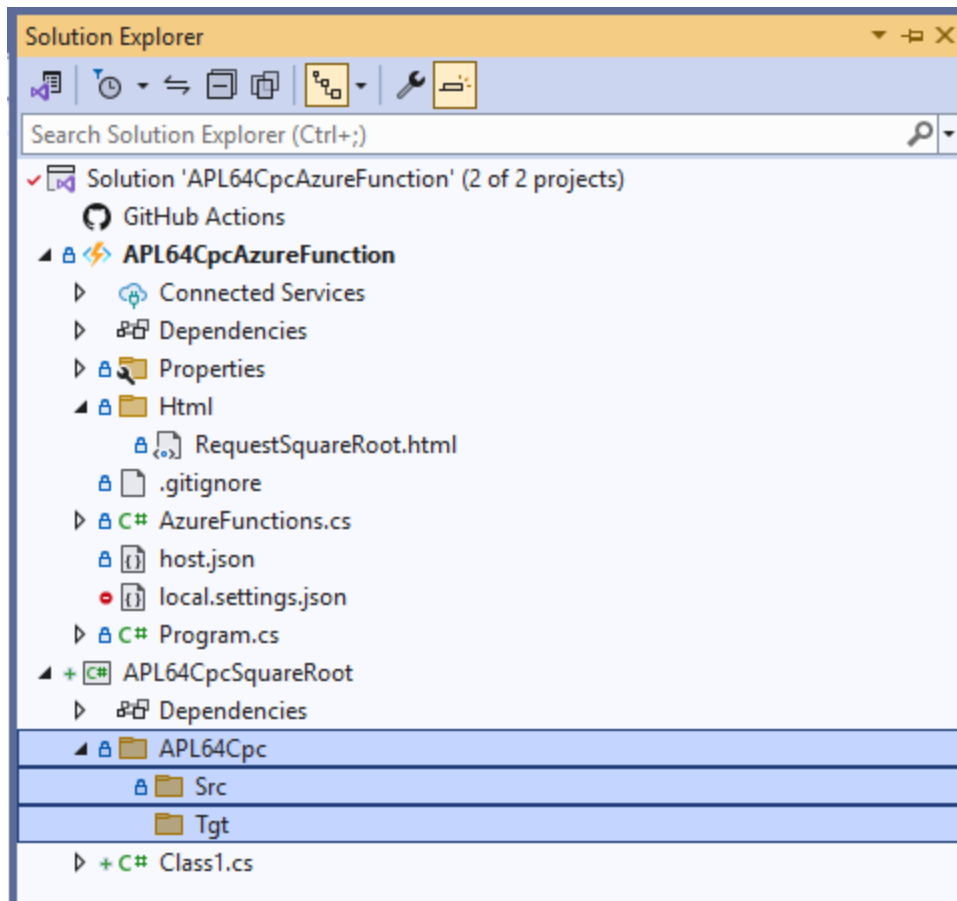
Framework ⓘ

.NET 8.0 (Long Term Support) ▾

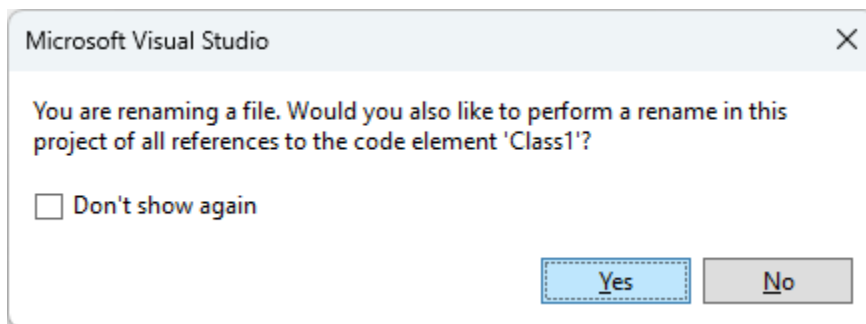
Back

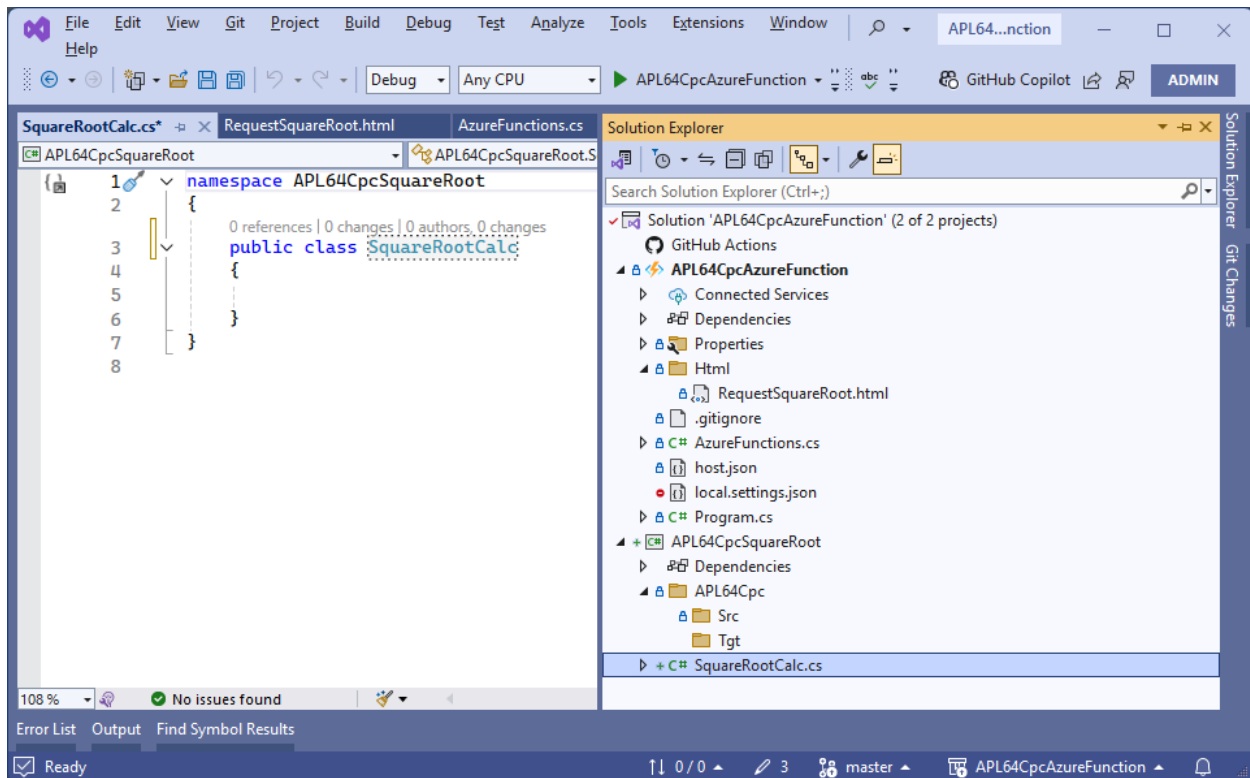
Create

Add the APL64Cpc folder and Src and Tgt folders to the APL64CpcSquareRoot project:



In the APL64CpcSquareRoot project, rename the Class1.cs code file and the contained class source code to SquareRootCalc.cs





Create the APL64 CPC

In an APL64 Developer version, create the APL64 public functions SquareRootCalculation and CubeRootCalculation functions.

.Net requires explicit data types for method arguments and results. In this example the Azure function provides and expects string data type values, so the APL64 programmer-defined, public function must use the enstring/destring (</>) functions appropriately.

```
:public (double@sqRt;bool@hasError;string@errMsg)←SquareRootCalculation double@input
hasError←0
:TRY
  errMsg←<"
  sqRt←input*0.5
:CATCH
  hasError←1
  errMsg←<EM
  sqRt←0
:ENDTRY
```

```
:public (double@cRt;bool@hasError;string@errMsg)←CubeRootCalculation double@input
hasError←0
```

```

:TRY
errMsg<<"
cRt<input*÷3
:CATCH
hasError<1
errMsg<<"EM
cRt<0
:ENDTRY

```

APL64: C:\APL64CpcAzureFunction\APL64CpcSquareRoot\APL64Cpc\Src\CPCRoots.ws64

File Edit Session Objects Tools Options Help

VSquareRo... X

```

0 :public (double@sqRt;bool@hasError;string@errMsg)+SquareRootCalculation double@input
1   hasError<0
2   :TRY
3     errMsg<<"
4     sqRt<input*0.5
5   :CATCH
6     hasError<1
7     errMsg<<"EM
8     sqRt<0
9   :ENDTRY

```

[9;7]

VCubeRoot... X

```

0 :public (double@cRt;bool@hasError;string@errMsg)+CubeRootCalculation double@input
1   hasError<0
2   :TRY
3     errMsg<<"
4     cRt<input*÷3
5   :CATCH
6     hasError<1
7     errMsg<<"EM
8     cRt<0
9   :ENDTRY

```

[4;13]

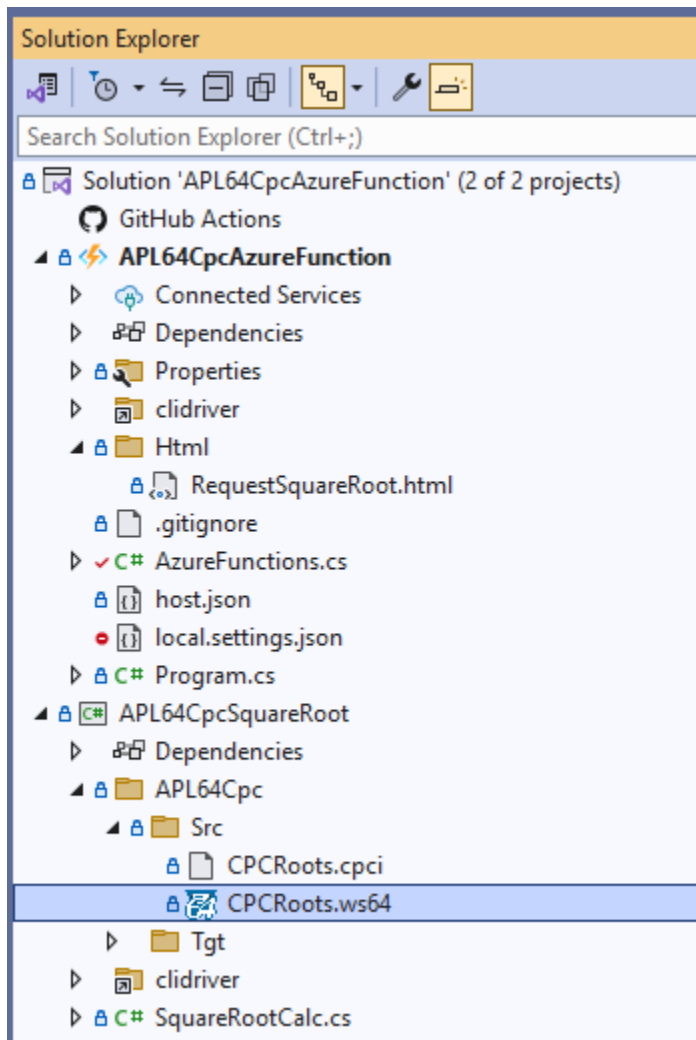
```

0 )fns
1 CubeRootCalculation SquareRootCalculation
2 )wsid
3 IS C:\APL64CpcAzureFunction\APL64CpcSquareRoot\APL64Cpc\Src\CPCRoots.ws64
4

```

Ready | Hist: Ln: 4 Col: 6 Ins | Classic | Num | EN_US

Save the workspace to the Src folder of the APL64CpcSquareRoot project:



Use the Options | Create Runtime Assembly | Create Cross-platform Component menu item to present the CPC creation dialog.

Complete the required entries of the CPC creation dialog.

Click the Create button to create the CPC. Some of the entries are case-sensitive since they are used in .Net.

Save the completed CPC information to the Src folder of the APL64CpcSquareRoot project

CPC Workspace Path *:

C:\APL64CpcAzureFunction\APL64CpcSquareRoot\APL64Cpc\Src\CPCRoots.ws64

Open Folder in File Explorer

CROSS PLATFORM COMPONENT CONTENT

Cross Platform Component Target Folder *:

C:\APL64CpcAzureFunction\APL64CpcSquareRoot\APL64Cpc\Tgt

Browse

Open Folder in File Explorer

Cross Platform Component Name *:

CPCRoots

☒ Same as CPC Workspace Name

Cross Platform Component Class Name *:

RootsClass

APL64 xml-format Configuration File:

Browse

☐ Include APL64 xml-format Configuration File

Additional Files Required for the Application

Source Path	Base Target Path	Target Path Suffix	Overwrite
Add One Required File Add Multiple Required Files Add Folder of Required Files Remove All Required Files			

ASSEMBLY META-DATA

Properties.Details: File Description:

Calculate roots

Properties.Details: Product Name *:

APLNext.RootCalcs

Properties.Details: Copyright *:

@APLNext LLC

Properties.Details: File Version *:

0.0.0.1

Properties.Details: Version:

0.0.0.1

Assembly.Info: Company Name *:

APLNext LLC

Assembly.Info: Application Description:

Assembly.Info: Version:

0.0.0.1

Assembly.Info: Neutral Language

Assembly.Info: Description:

Calculate roots

Application Icon File:

Browse

Nuget Package Guid *:

34e04381-347a-43b8-b13d-71d2dee98cc8

Create

Nuget Package Id*:

APLNext_LLC.CPCRoots

☒ Use CompanyName.ComponentName

Nuget Package Files

Create

Exit

New CPC Info

Load CPC Info

Save CPC Info

* Required Entries

APL64: Create Cross Platform Component

CPC created: CPCRoots.dll

Create CPC result: Publish procedure successful

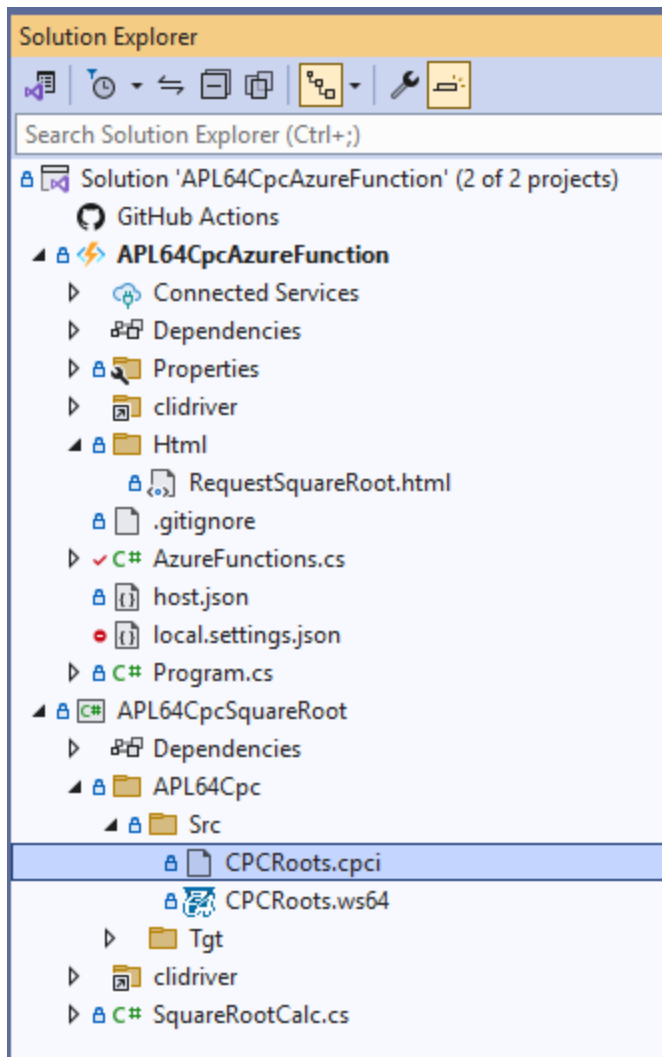
Publish output log file:

C:\APL64CpcAzureFunction\APL64CpcSquareRoot\APL64Cpc\Tgt\CPCRoots.Info.log

APL64 Cross-platform Component:

C:\APL64CpcAzureFunction\APL64CpcSquareRoot\APL64Cpc\Tgt\CPCRoots.dll

OK

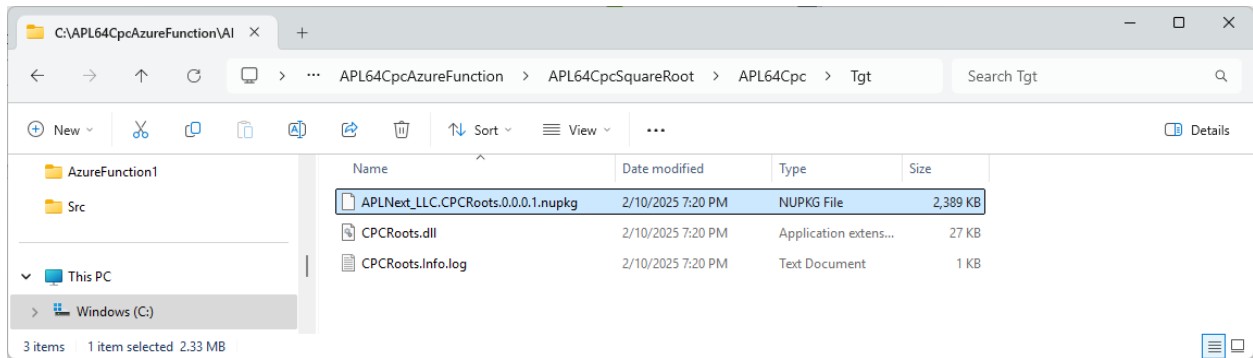


Publish the APL64 CPC Nuget package

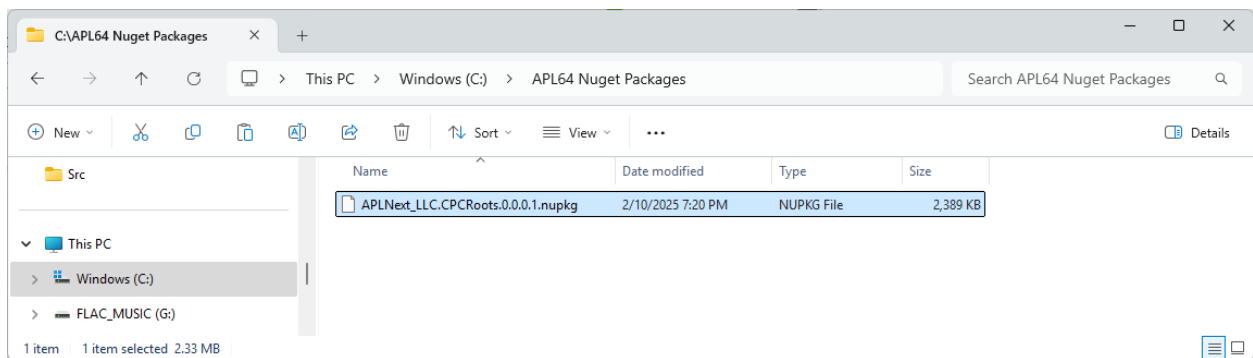
In a production environment the Nuget package may be published to a public or private Nuget repository. For testing purposes, the Nuget package must be copied to a location on the current workstation which is not in the Visual Studio solution folder to avoid including unnecessary components in the project.

Copy the APL64 CPC Nuget package from the Tgt folder of the APL64CpcSquareRoot project to the APL64 Nuget Package publishing folder:

From:

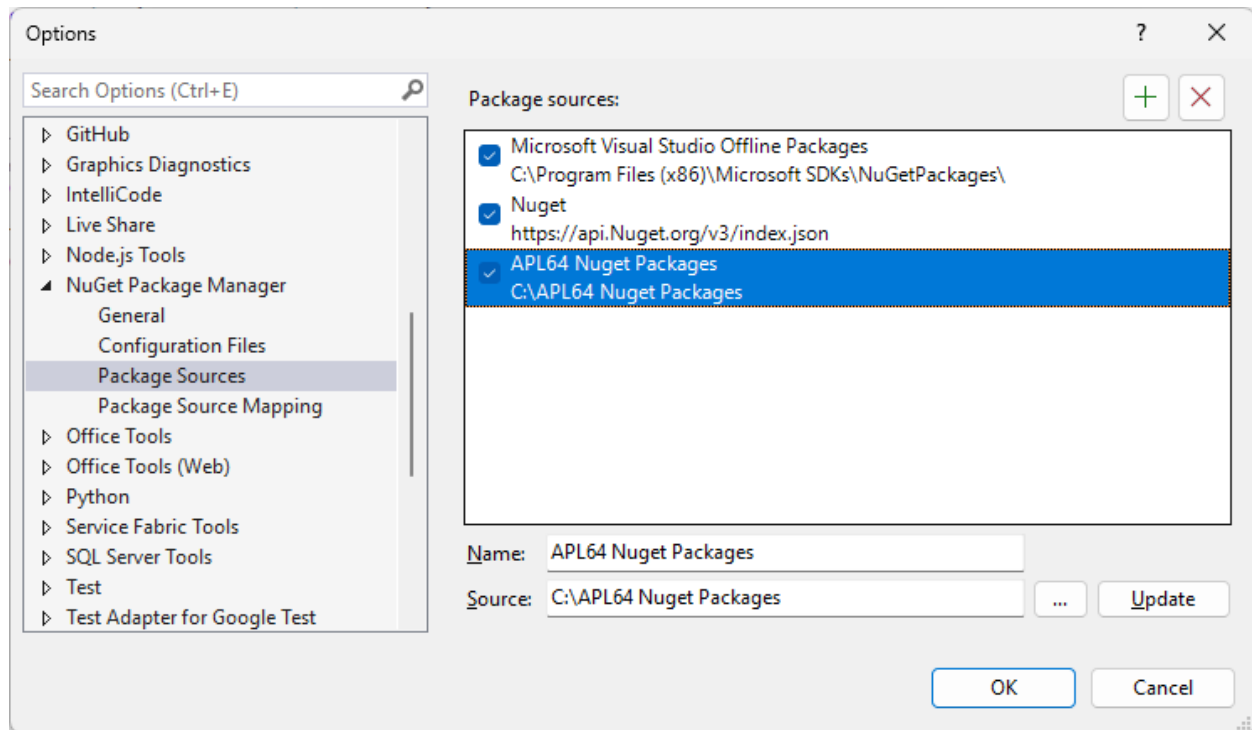


To:



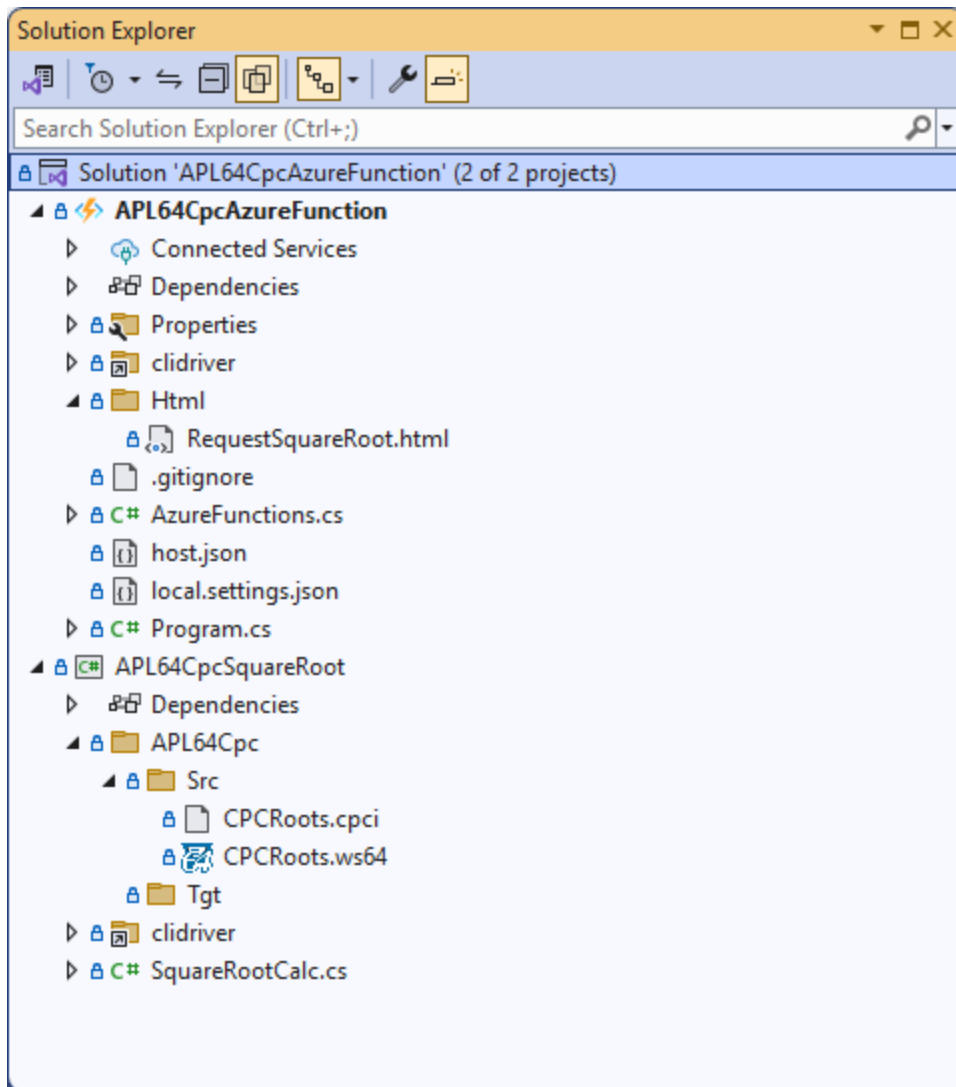
Set-up the Visual Studio Package Sources

The publish folder containing the APL64 Nuget package must be included in the Visual Studio 'package sources':



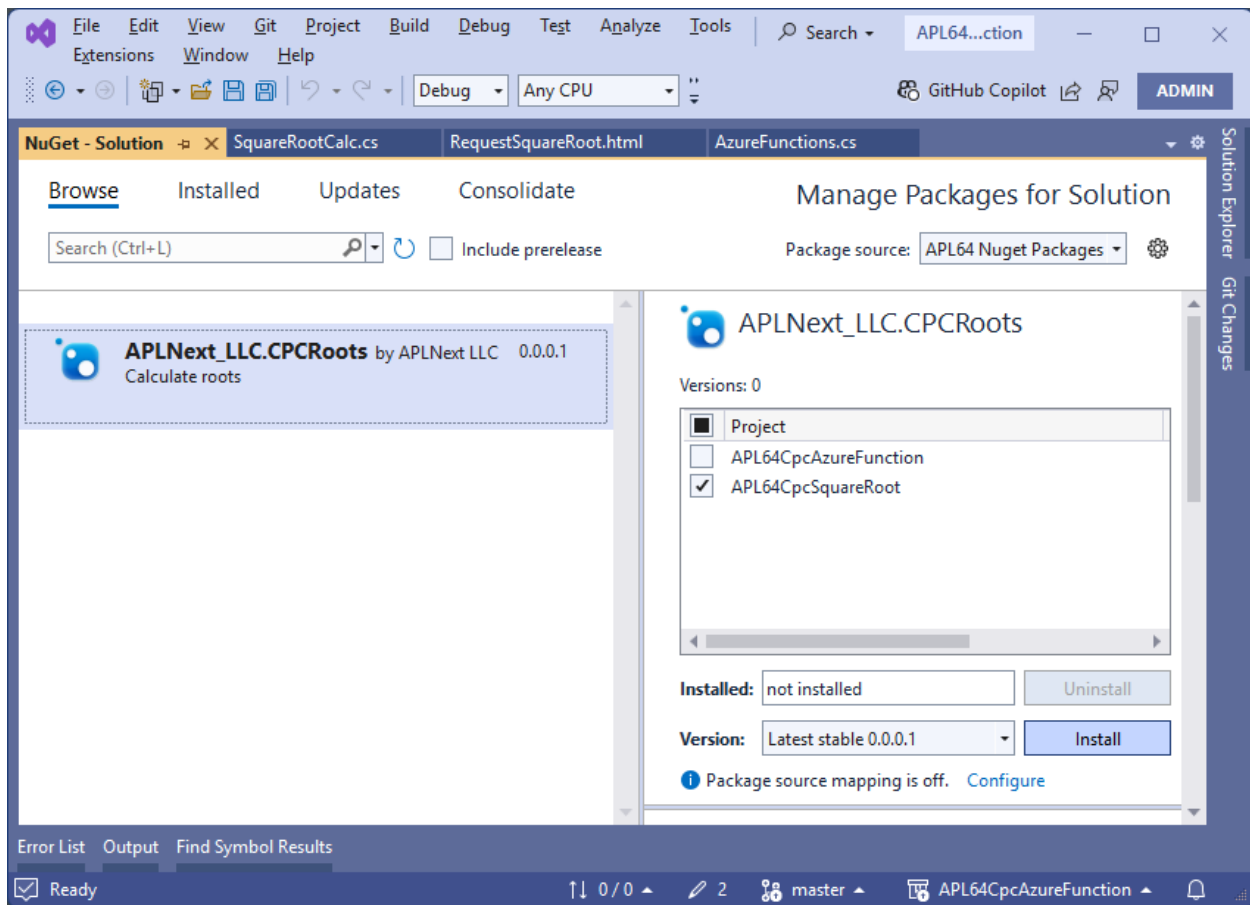
Install the APL64 CPC Nuget package into the Project

Select the Solution node of the Visual Studio Solution Explorer:



Use the **Tools | Nuget Package Manager | Manage Nuget Packages for Solution** menu item in Visual Studio to present the Manage Packages for Solution dialog. Select the APL64 CPC Nuget package, and install it into the APL64CpcSquareRoot project of the solution.

The APL64 CPC is used only in the APL64CpcSquareRootProject, so the APL64 CPC Nuget package needs to be installed only in that project.



Update the SquareRootCalc.cs Source Code

Replace the source code in the SquareRootCalc.cs code file in the APL64CpcSquareRoot project with:

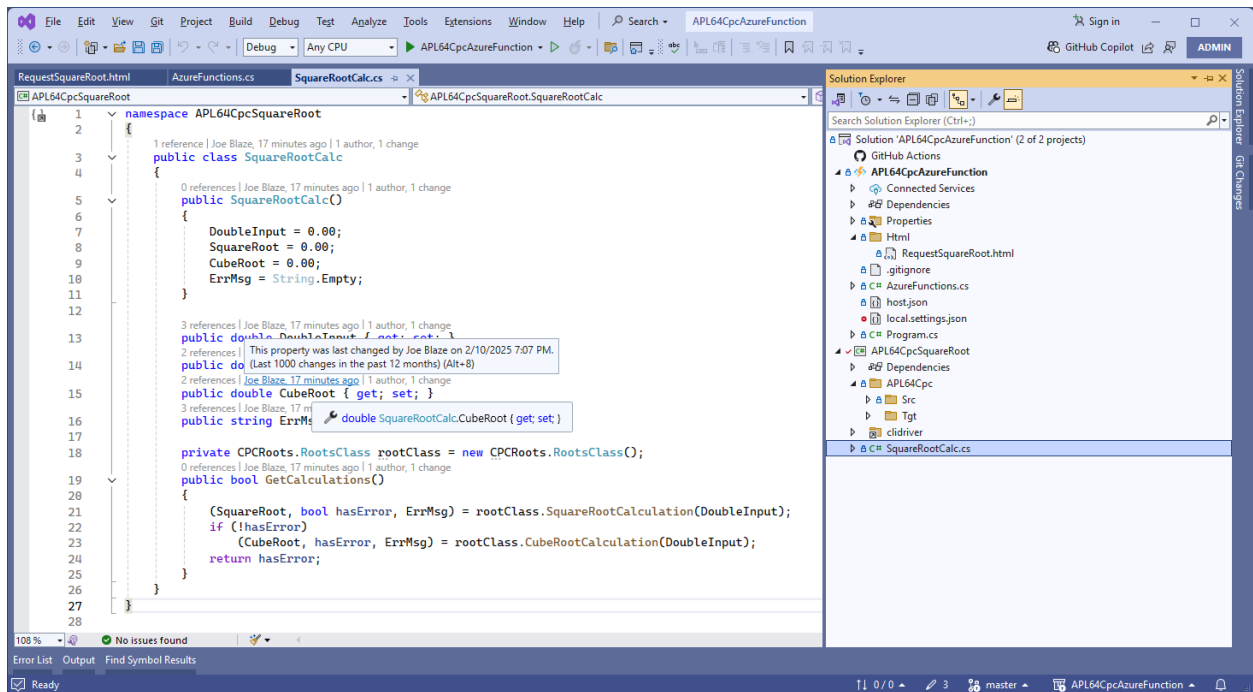
```
namespace APL64CpcSquareRoot
{
    public class SquareRootCalc
    {
        public SquareRootCalc()
        {
            DoubleInput = 0.00;
            SquareRoot = 0.00;
            CubeRoot = 0.00;
            ErrMsg = String.Empty;
        }

        public double DoubleInput { get; set; }
        public double SquareRoot { get; set; }
        public double CubeRoot { get; set; }
        public string ErrMsg { get; set; }
    }
}
```

```

private CPCRoots.RootsClass rootClass = new CPCRoots.RootsClass();
public bool GetCalculations()
{
    (SquareRoot, bool hasError, ErrMsg) = rootClass.SquareRootCalculation(DoubleInput);
    if (!hasError)
        (CubeRoot, hasError, ErrMsg) = rootClass.CubeRootCalculation(DoubleInput);
    return hasError;
}
}
}

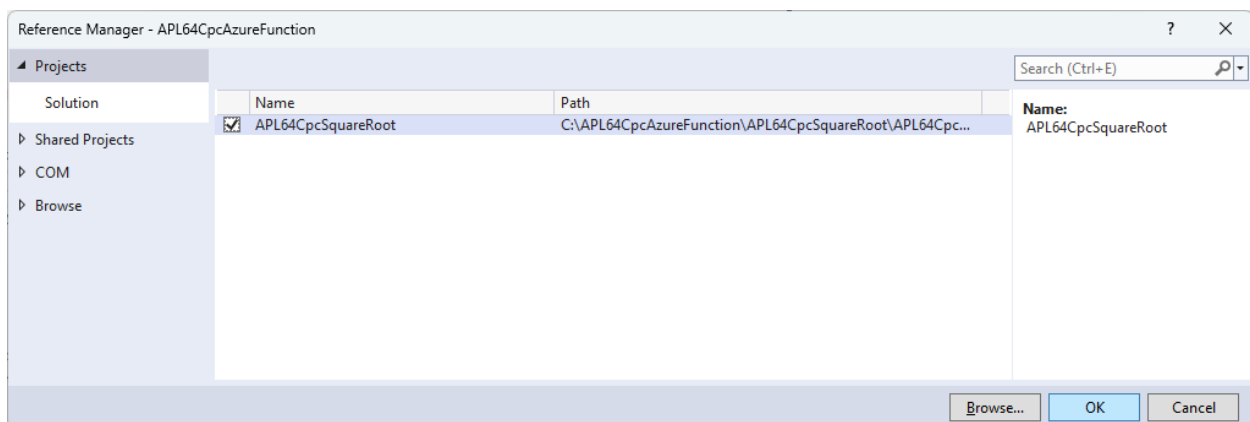
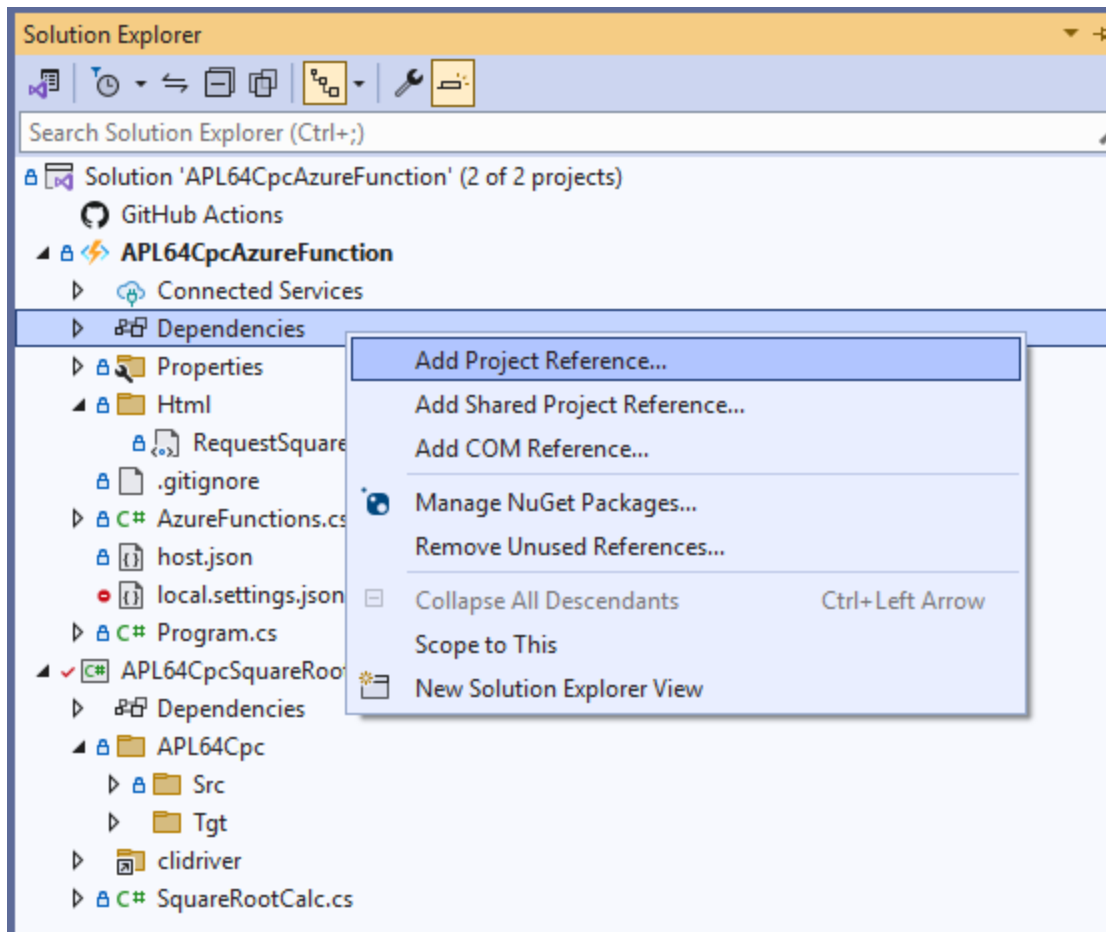
```



The 'rootClass' is an instance of the APL64 CPC CPCRoots.RootsClassm which exposes the APL64 SquareRootCalculation and CubeRootCalculation public functions as .Net methods. When the GetCalculations() method is run, the user input (DoubleInput) is passed to the APL64 public functions which set the SquareRoot class SquareRoot, CubeRoot and ErrMsg property values.

Add the APL64CpcSquareRoot Project Dependency

Add the APL64CpcSquareRoot project as a dependency of the APL64CpcAzureFunction project:



Update the AzureFunctions.cs Code File

Update (replace) the code in the AzureFunctions.cs code file with:

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
```

```

using Microsoft.Azure.Functions.Worker;
using Microsoft.Extensions.Logging;
using System.Reflection;
using System.Text;

namespace APL64CpcAzureFunction
{
    public class AzureFunctions
    {
        private readonly ILogger<AzureFunctions> _logger;

        public AzureFunctions(ILogger<AzureFunctions> logger)
        {
            _logger = logger;
        }

        [Function("GetInput")]
        public IActionResult RunGetInput([HttpTrigger(AuthorizationLevel.Anonymous, "get", "post")]
        HttpRequest req)
        {
            _logger.LogInformation("C# HTTP trigger function processed a request.");
            var asmPath = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
            var path = Path.Combine(asmPath, "Html/RequestSquareRoot.html");
            var htmlText = File.ReadAllText(path);
            var contentResponse = new ContentResult()
            {
                Content = htmlText,
                ContentType = "text/html",
                StatusCode = 200
            };
            return contentResponse;
        }

        [Function("CalculateResult")]
        public IActionResult RunCalculateResult([HttpTrigger(AuthorizationLevel.Anonymous, "post",
        Route = null)] HttpRequest req)
        {
            _logger.LogInformation("C# HTTP trigger function processed a request.");

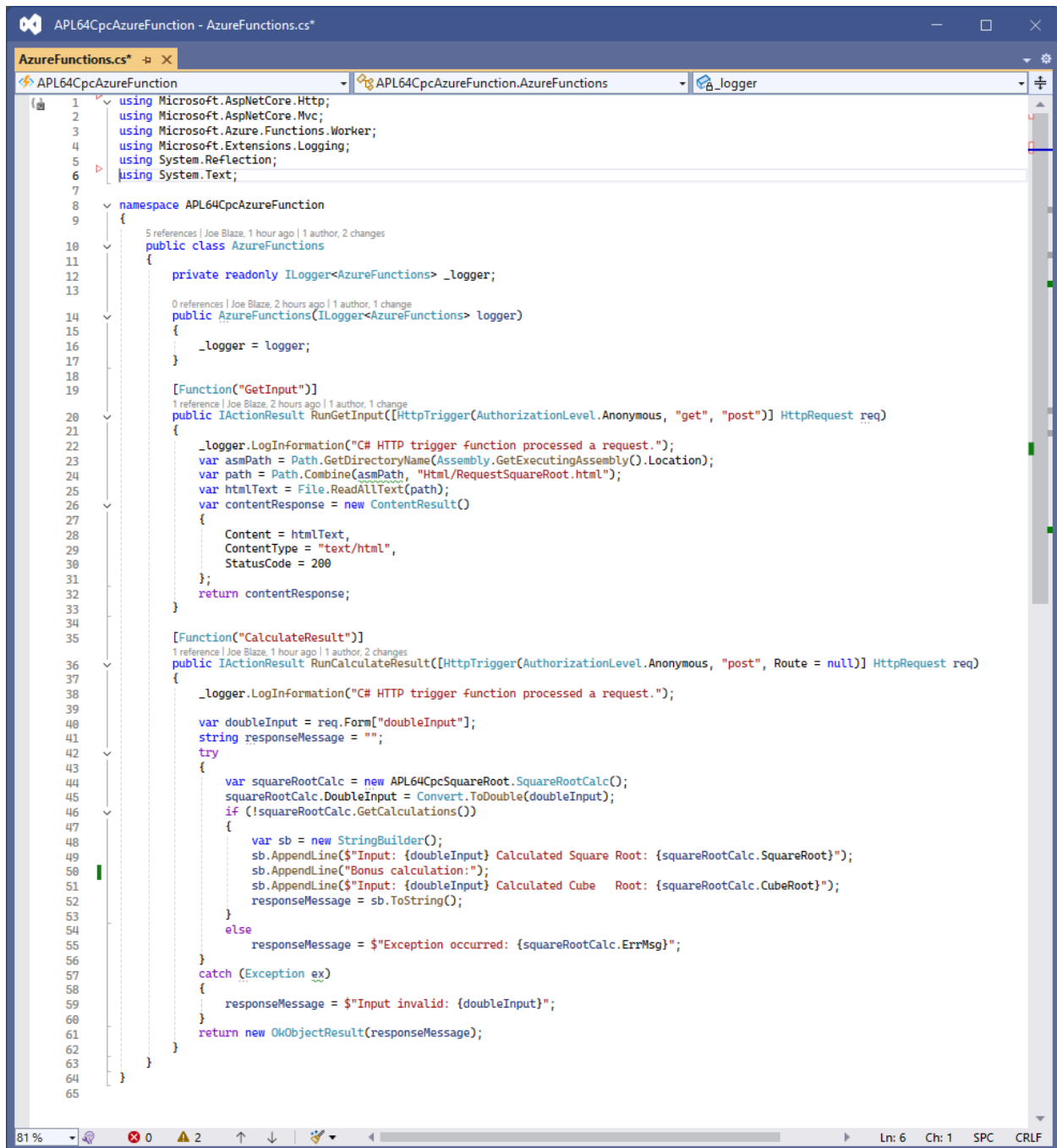
            var doubleInput = req.Form["doubleInput"];
            string responseMessage = "";
            try
            {
                var squareRootCalc = new APL64CpcSquareRoot.SquareRootCalc();
                squareRootCalc.DoubleInput = Convert.ToDouble(doubleInput);
                if (!squareRootCalc.GetCalculations())
                {
                    var sb = new StringBuilder();

```

```

        sb.AppendLine($"Input:      {doubleInput}      Calculated      Square      Root:
{squareRootCalc.SquareRoot}");
        sb.AppendLine("Bonus calculation:");
        sb.AppendLine($"Input:      {doubleInput}      Calculated      Cube      Root:
{squareRootCalc.CubeRoot}");
        responseMessage = sb.ToString();
    }
    else
        responseMessage = $"Exception occurred: {squareRootCalc.ErrMsg}";
    }
    catch (Exception ex)
    {
        responseMessage = $"Input invalid: {doubleInput}";
    }
    return new OkObjectResult(responseMessage);
}
}
}

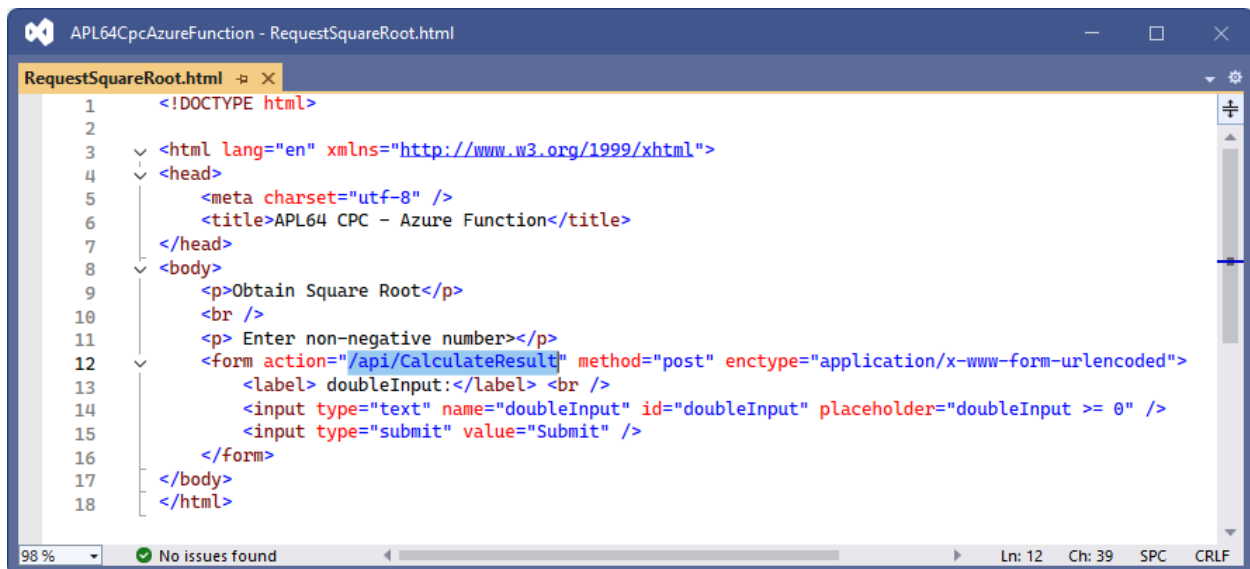
```



```
1 using Microsoft.AspNetCore.Http;
2 using Microsoft.AspNetCore.Mvc;
3 using Microsoft.Azure.Functions.Worker;
4 using Microsoft.Extensions.Logging;
5 using System.Reflection;
6 using System.Text;
7
8 namespace APL64CpcAzureFunction
9 {
10     5 references | Joe Blaze, 1 hour ago | 1 author, 2 changes
11     public class AzureFunctions
12     {
13         private readonly ILogger<AzureFunctions> _logger;
14
15         0 references | Joe Blaze, 2 hours ago | 1 author, 1 change
16         public AzureFunctions(ILogger<AzureFunctions> logger)
17         {
18             _logger = logger;
19         }
20
21         [Function("GetInput")]
22         1 reference | Joe Blaze, 2 hours ago | 1 author, 1 change
23         public IActionResult RunGetInput([HttpTrigger(AuthorizationLevel.Anonymous, "get", "post")] HttpRequest req)
24         {
25             _logger.LogInformation("C# HTTP trigger function processed a request.");
26             var asmPath = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
27             var path = Path.Combine(asmPath, "Html/RequestSquareRoot.html");
28             var htmlText = File.ReadAllText(path);
29             var contentResponse = new ContentResult()
30             {
31                 Content = htmlText,
32                 ContentType = "text/html",
33                 StatusCode = 200
34             };
35             return contentResponse;
36         }
37
38         [Function("CalculateResult")]
39         1 reference | Joe Blaze, 1 hour ago | 1 author, 2 changes
40         public IActionResult RunCalculateResult([HttpTrigger(AuthorizationLevel.Anonymous, "post", Route = null)] HttpRequest req)
41         {
42             _logger.LogInformation("C# HTTP trigger function processed a request.");
43
44             var doubleInput = req.Form["doubleInput"];
45             string responseMessage = "";
46             try
47             {
48                 var squareRootCalc = new APL64CpcSquareRoot.SquareRootCalc();
49                 squareRootCalc.DoubleInput = Convert.ToDouble(doubleInput);
50                 if (!squareRootCalc.GetCalculations())
51                 {
52                     var sb = new StringBuilder();
53                     sb.AppendLine($"Input: {doubleInput} Calculated Square Root: {squareRootCalc.SquareRoot}");
54                     sb.AppendLine("Bonus calculation:");
55                     sb.AppendLine($"Input: {doubleInput} Calculated Cube Root: {squareRootCalc.CubeRoot}");
56                     responseMessage = sb.ToString();
57                 }
58                 else
59                 {
60                     responseMessage = $"Exception occurred: {squareRootCalc.ErrMsg}";
61                 }
62             }
63             catch (Exception ex)
64             {
65                 responseMessage = $"Input invalid: {doubleInput}";
66             }
67             return new OkObjectResult(responseMessage);
68         }
69     }
70 }
```

The 'GetInput' Azure function reads the text of the 'RequestSquareRoot.html' file in the project and uses that text as the content response of the end user's request for a square root calculation. The 'RequestSquareRoot.html' file text is presented as a web page to the end user.

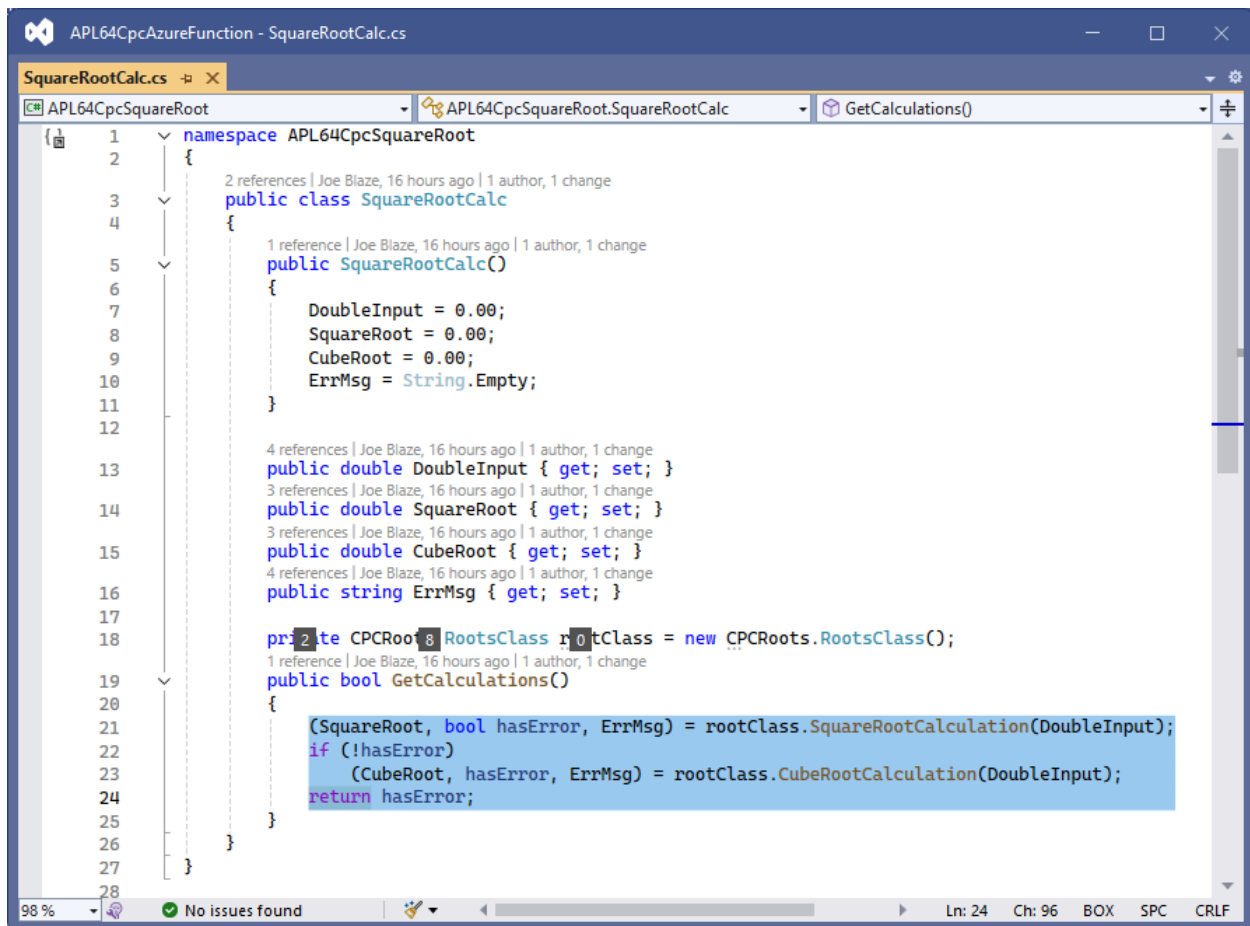
When the submit button is clicked, the 'RequestSquareRoot.html' page directs the end user's request for a square root calculation to the '/api/CalculateResult' url which is supported by the 'CalculateResult' Azure function:



```
1 <!DOCTYPE html>
2
3 <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5   <meta charset="utf-8" />
6   <title>APL64 CPC - Azure Function</title>
7 </head>
8 <body>
9   <p>Obtain Square Root</p>
10  <br />
11  <p>Enter non-negative number</p>
12  <form action="/api/CalculateResult" method="post" enctype="application/x-www-form-urlencoded">
13    <label> doubleInput:</label> <br />
14    <input type="text" name="doubleInput" id="doubleInput" placeholder="doubleInput >= 0" />
15    <input type="submit" value="Submit" />
16  </form>
17 </body>
18 </html>
```

When the end user provides the input and clicks the submit button, the input is provided to the 'CalculateResult' Azure function. After the calculations are completed, the 'CalculateResult' Azure function prepares the response and sends it to the end user's browser.

The 'CalculateResult' Azure function uses the APL64 public functions, exposed as .Net methods in the APL64 CPC Nuget package to obtain the calculated results. The 'CalculateResult' Azure function returns the result as an 'OkObjectResult' message to the client's browser:



```
1 namespace APL64CpcSquareRoot
2 {
3     2 references | Joe Blaze, 16 hours ago | 1 author, 1 change
4     public class SquareRootCalc
5     {
6         1 reference | Joe Blaze, 16 hours ago | 1 author, 1 change
7         public SquareRootCalc()
8         {
9             DoubleInput = 0.00;
10            SquareRoot = 0.00;
11            CubeRoot = 0.00;
12            ErrMsg = String.Empty;
13        }
14
15        4 references | Joe Blaze, 16 hours ago | 1 author, 1 change
16        public double DoubleInput { get; set; }
17
18        3 references | Joe Blaze, 16 hours ago | 1 author, 1 change
19        public double SquareRoot { get; set; }
20
21        3 references | Joe Blaze, 16 hours ago | 1 author, 1 change
22        public double CubeRoot { get; set; }
23
24        4 references | Joe Blaze, 16 hours ago | 1 author, 1 change
25        public string ErrMsg { get; set; }
26
27        private CPCRoots.RootsClass rootClass = new CPCRoots.RootsClass();
28        1 reference | Joe Blaze, 16 hours ago | 1 author, 1 change
29        public bool GetCalculations()
30        {
31            (SquareRoot, bool hasError, ErrMsg) = rootClass.SquareRootCalculation(DoubleInput);
32            if (!hasError)
33            {
34                (CubeRoot, hasError, ErrMsg) = rootClass.CubeRootCalculation(DoubleInput);
35            }
36            return hasError;
37        }
38    }
39 }
```

Run the Application

Click F5 in Visual Studio to run the application.

The Azurite Azure emulator creates the urls for the 'GetInput' and 'CalculateResults' Azure functions. In a production application the urls are published by Azure for (authorized) use by the clients of the application.

In Visual Studio, click F5 to run the Azurite simulator:

```
Select C:\Program Files\dotnet\dotnet.exe

Azure Functions Core Tools
Core Tools Version:          4.0.6821 Commit hash: N/A +c09a2033faa7ecf51b3773308283af0ca9a99f83 (64-bit)
Function Runtime Version: 4.1036.1.23224

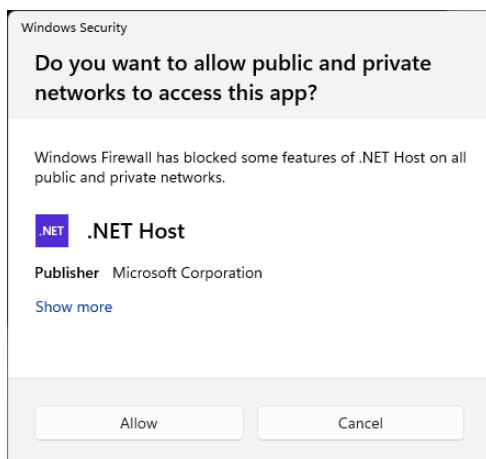
[2025-02-11T16:24:52.183Z] Found C:\APL64CpcAzureFunction\APL64CpcAzureFunction\APL64CpcAzureFunction.csproj. Using for
user secrets file configuration.
[2025-02-11T16:24:54.231Z] Azure Functions .NET Worker (PID: 30852) initialized in debug mode. Waiting for debugger to a
ttach...
[2025-02-11T16:24:54.266Z] Worker process started and initialized.

Functions:

    CalculateResult: [POST] http://localhost:7167/api/CalculateResult
    GetInput: [GET,POST] http://localhost:7167/api/GetInput

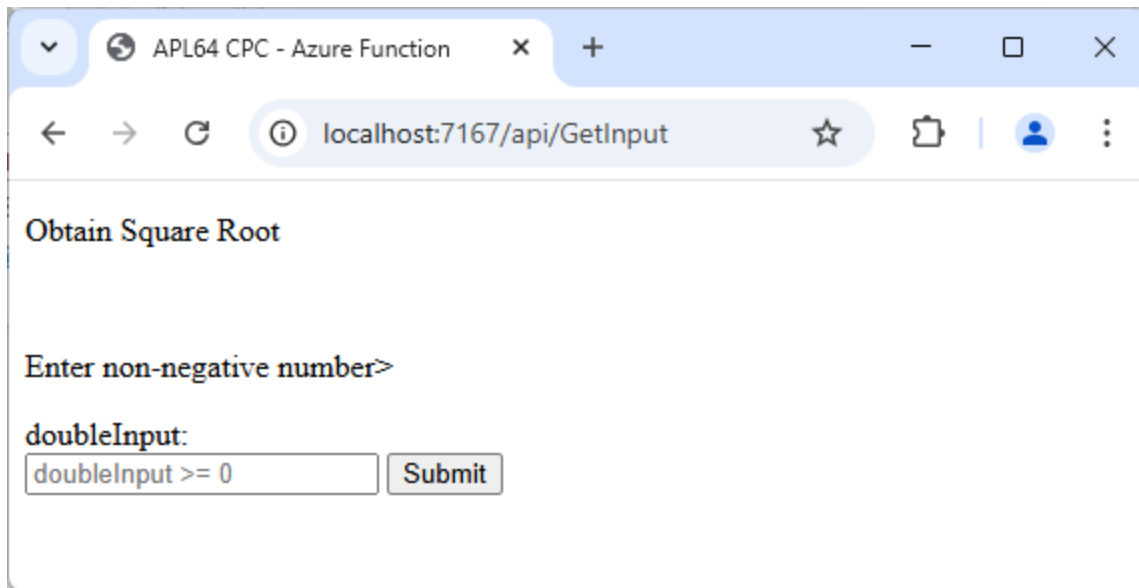
For detailed output, run func with --verbose flag.
```

Before the application starts, a Windows warning may be presented. Click the 'Allow' button to continue:

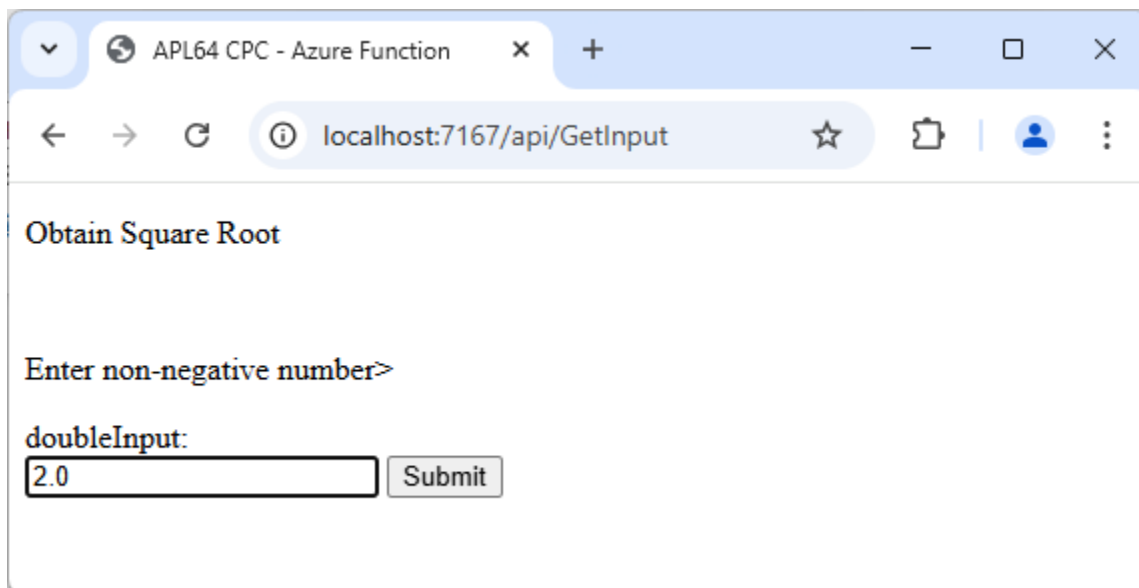


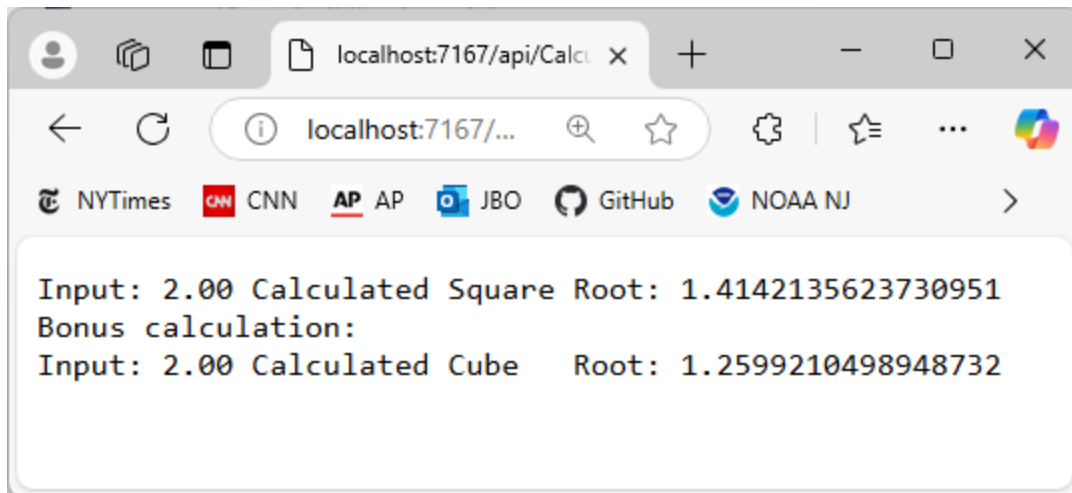
The developer of the application will provide the end user with the url of the application. To simulate the end user, copy the 'GetInput' link: <http://localhost:7167/api/GetInput>, and paste it into a web browser to run the application:

Paste it into a Web browser and browse to the link, to display the input html-format page:

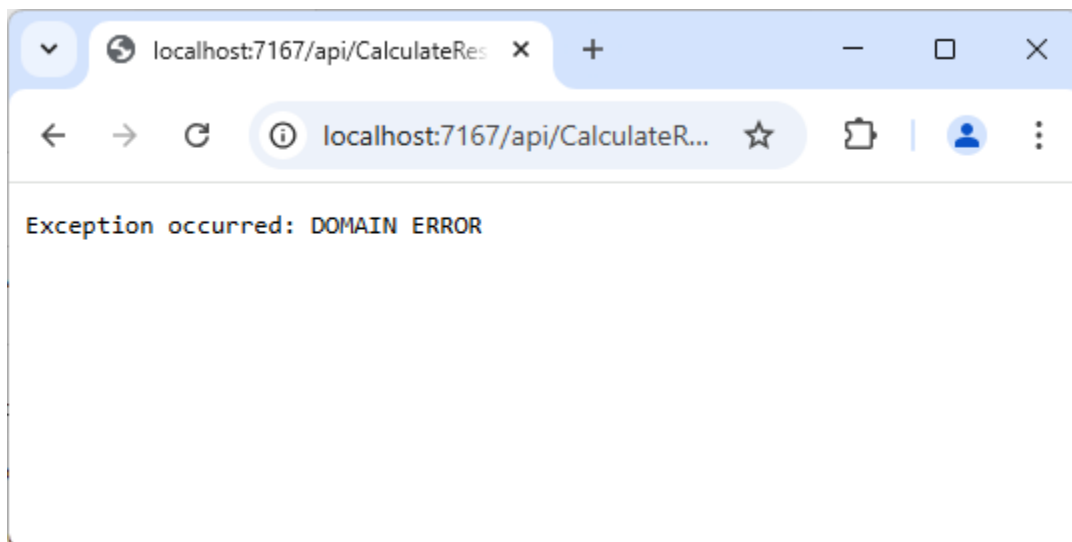
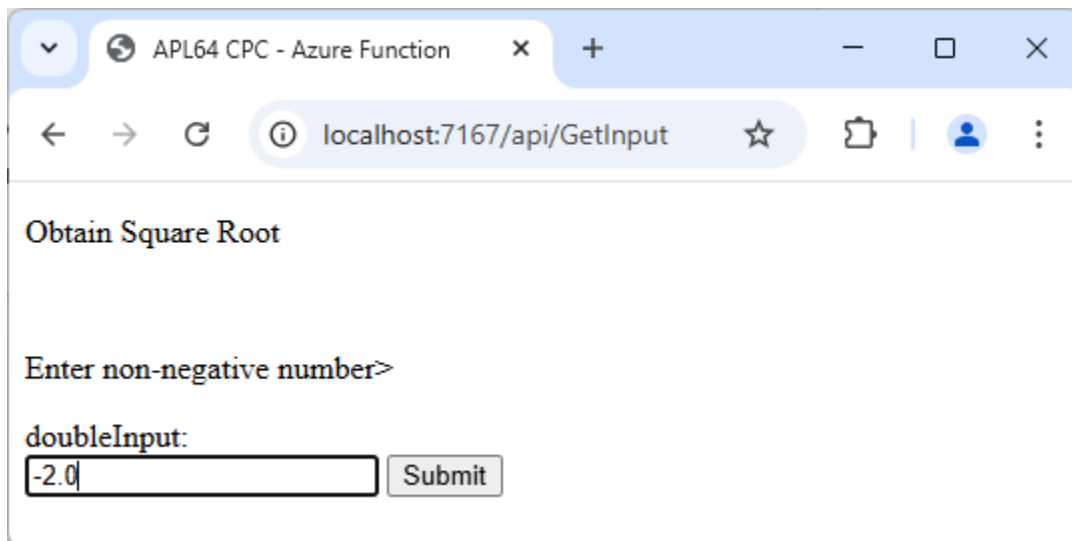


Enter a non-negative number into the Html input GUI control, and click the Html Submit button GUI control to initiate the request to calculate the square root of the input value:





Browse to the application's data input url, enter a negative number, and click the 'Submit' button:



Azure Function Deployment

This example is a demonstration project. Significant cosmetic enhancements, security and validation features need to be added for a production-ready project.

A no-cost [Azure subscription](#) is available to test an Azure Function application, and eventually make it available as a production application to users.

Publishing an Azure Function application is beyond the scope of this document. A basic workflow for the process of [publishing an Azure Function application](#) is available starting with the 'Publish the project to Azure' paragraph in the linked article.

Learn More

Contact support@apl2000.com for assistance using APL64 to create applications like an Azure Function application.