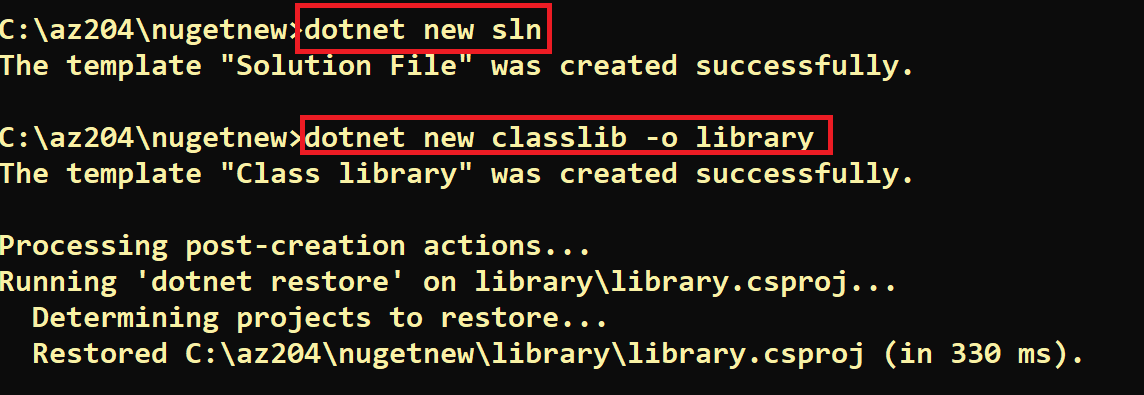
md nugettest

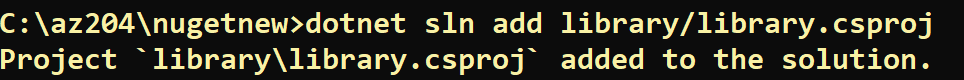
cd nugettest

dotnet new sln

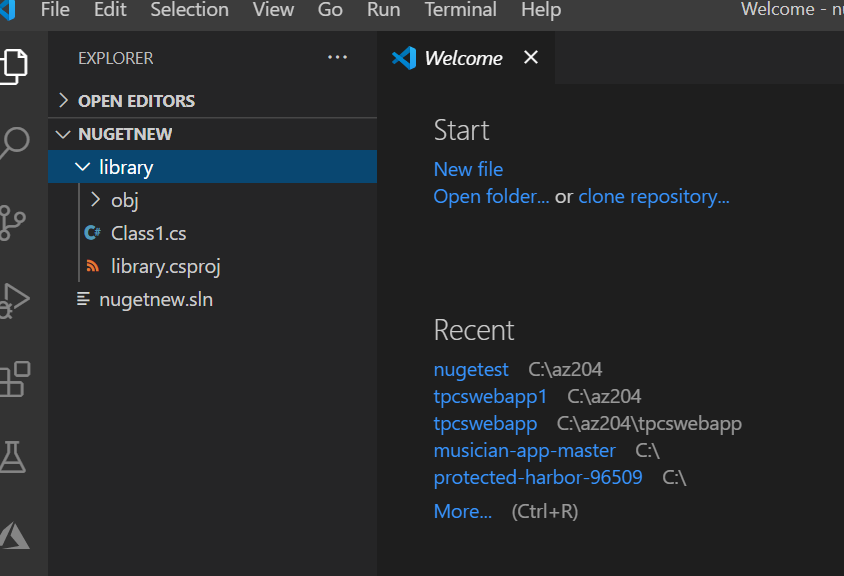
dotnet new classlib -o library



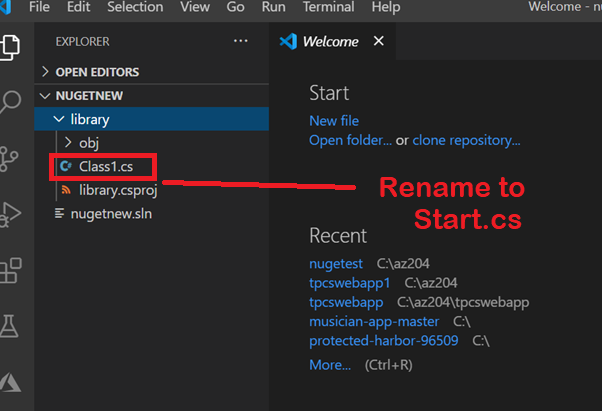
dotnet sln add library/library.csproj



code .



Rename our Class1.cs file to Start.cs.



Now let's replace whatever code is in there with this:

using System;

namespace library

{

  public class Start

  {

    public int Add(int a, int b)

    {

        return a + b;

    }

    public int Sub(int a, int b)

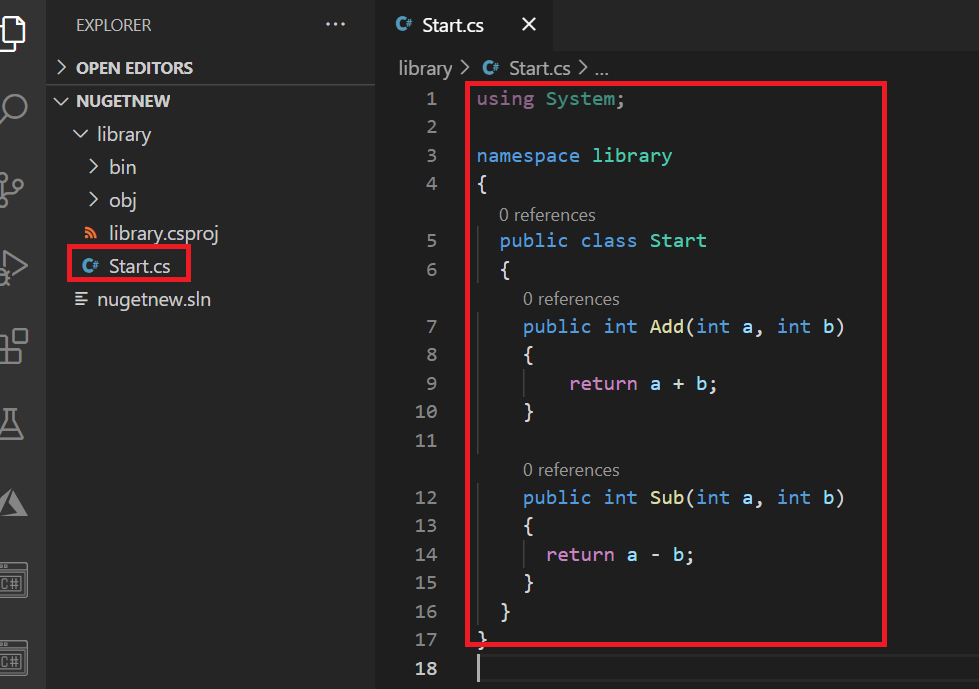
    {

      return a - b;

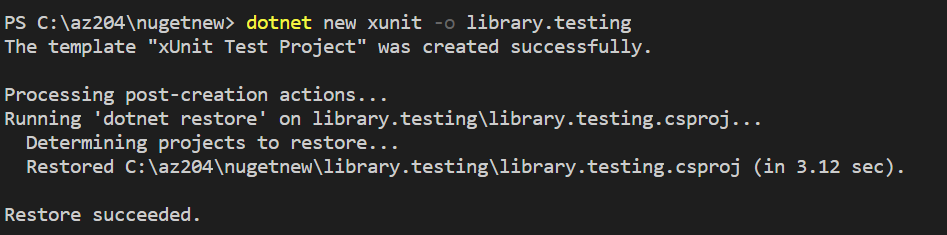
    }

  }

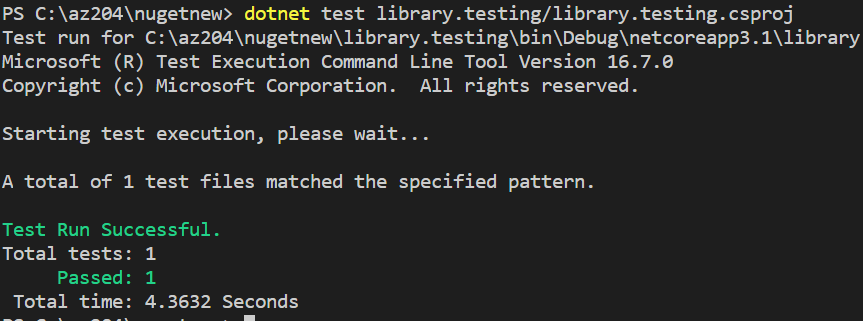
}



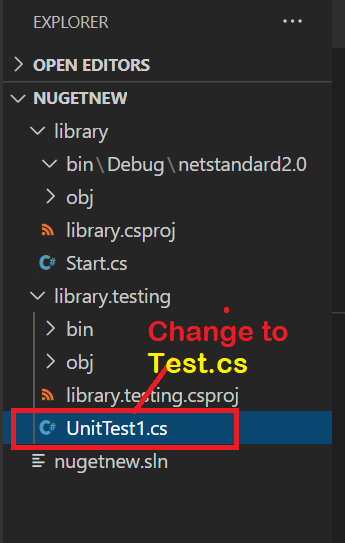
dotnet new xunit -o library.testing

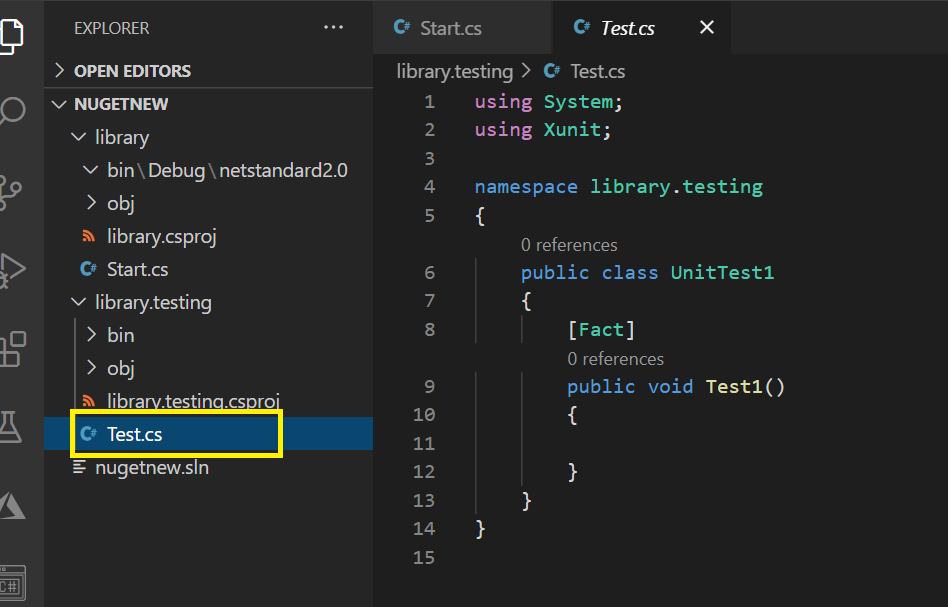


dotnet test library.testing/library.testing.csproj



Rename test file UnitTest1.cs to Test.cs. What you name doesn't matter, it's just nice if you give it a meaningful name.





Now Put This Code

using System;

using Xunit;

using library;

namespace library.testing

{

    public class Test

    {

        [Fact]

        public void TestingAdd()

        {

            var start = new Start();

            var actual = start.Add(1, 2);

            Assert.Equal(3, actual);

        }

        [Fact]

        public void TestingSub()

        {

            var start = new Start();

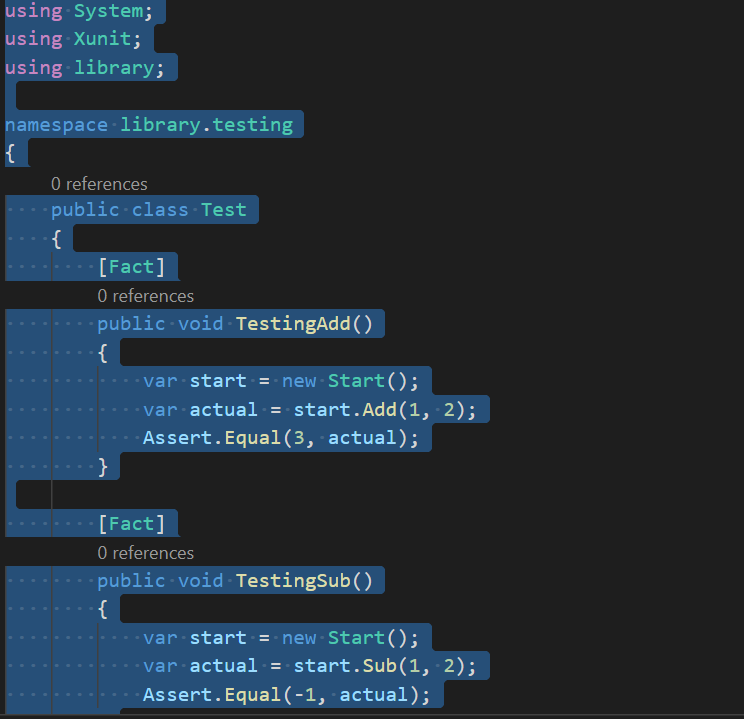
            var actual = start.Sub(1, 2);

            Assert.Equal(-1, actual);

        }

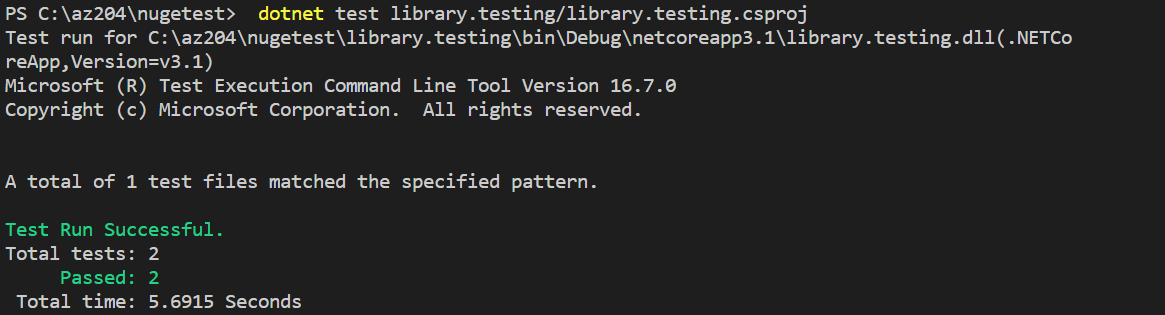
    }

}



Next head to the terminal and type:

dotnet test library.testing/library.testing.csproj



Good, we are ready for turning this into a NuGet package now that our code looks like its working.

Create a NuGet package

We need to do the following to create a NuGet package:

1. **Add** some meta information to the library project
2. **Invoke** the pack command, this will create the package

Meta information are things like name, author, company and so on. You will need to add this as entry in the library.csproj file under the XML tag PropertyGroup. It can look like this:

<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>

    <TargetFramework>netcoreapp3.0</TargetFramework>

    <PackageId>TPCS</PackageId>

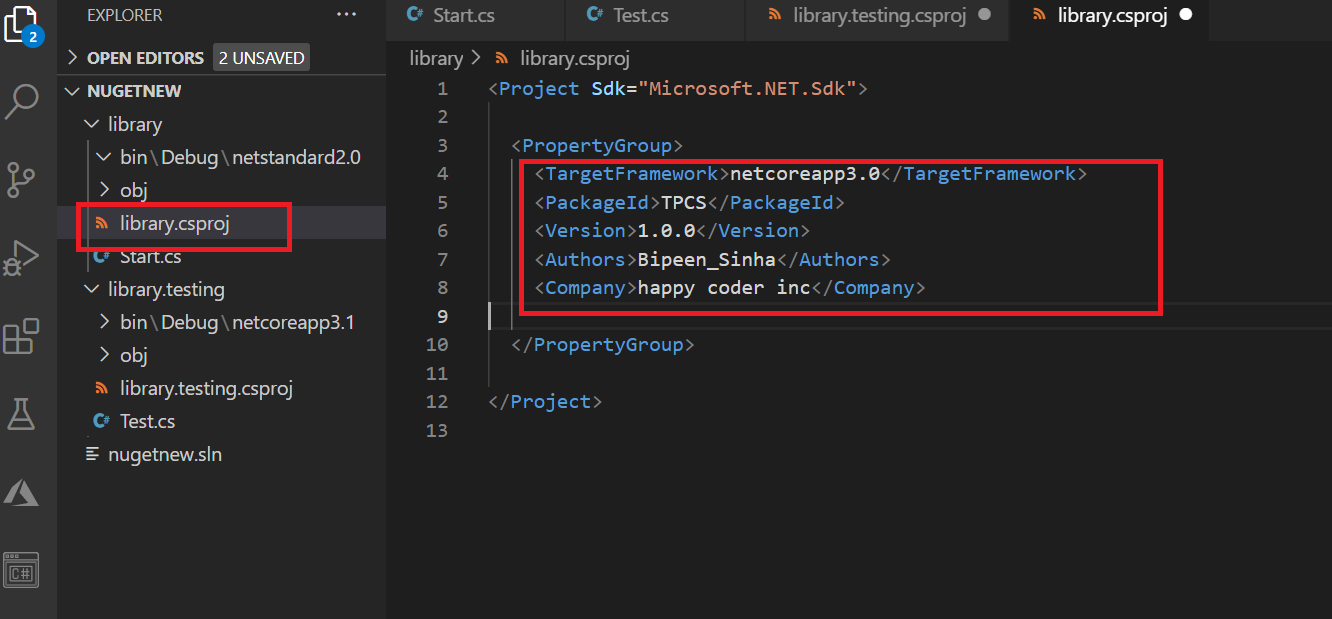
    <Version>1.0.0</Version>

    <Authors>Bipeen\_Sinha</Authors>

    <Company>happy coder inc</Company>

  </PropertyGroup>

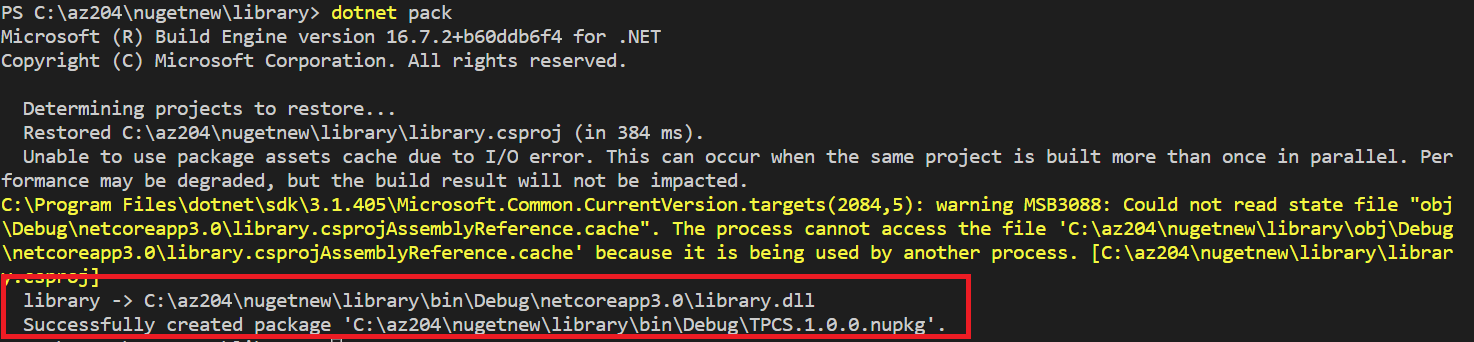
</Project>



Next let's create our package. Let's navigate to our library directory and type:

cd library

dotnet pack



We can see from the above image that we got a package created in the bin/Debug folder called TPCS.1.0.0.nupkg.

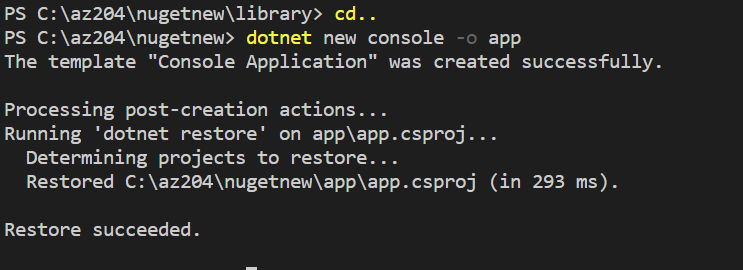
## Install your package locally

So how do we do this? Well, we need a project to test it on. Let's scaffold a console application

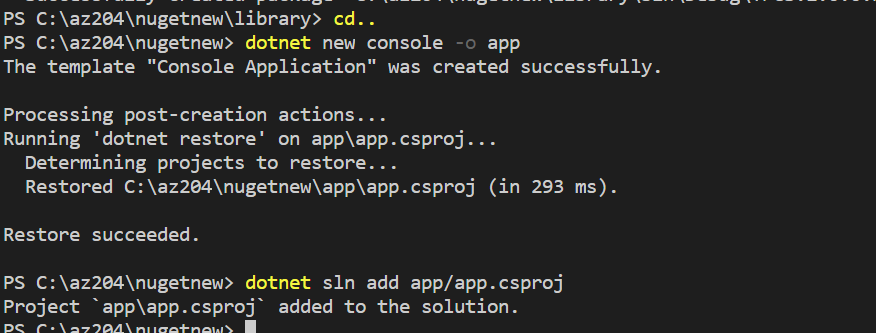
In vs terminal

cd..

dotnet new console -o app

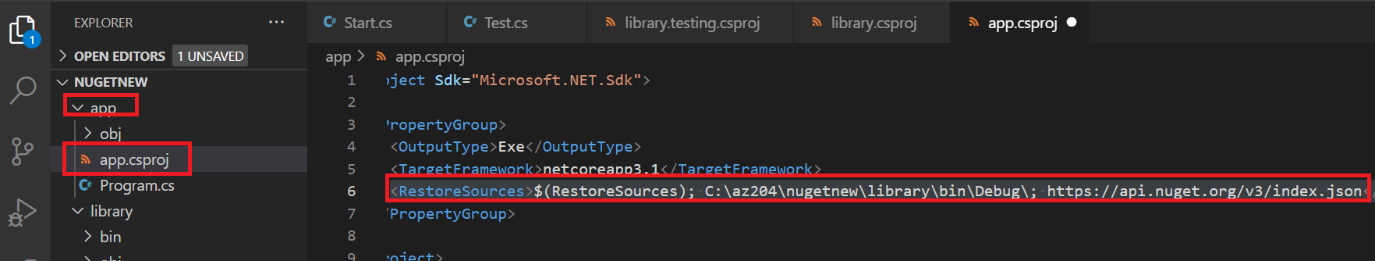


dotnet sln add app/app.csproj



We need to go into the target projects .csproj file. We want to test out our NuGet package on our console app called app so we open up app.csproj and under PropertyGroup we add a tag called RestoreSources. Here we point out both the path to our local NuGet package and the NuGet stream. It should look like so:

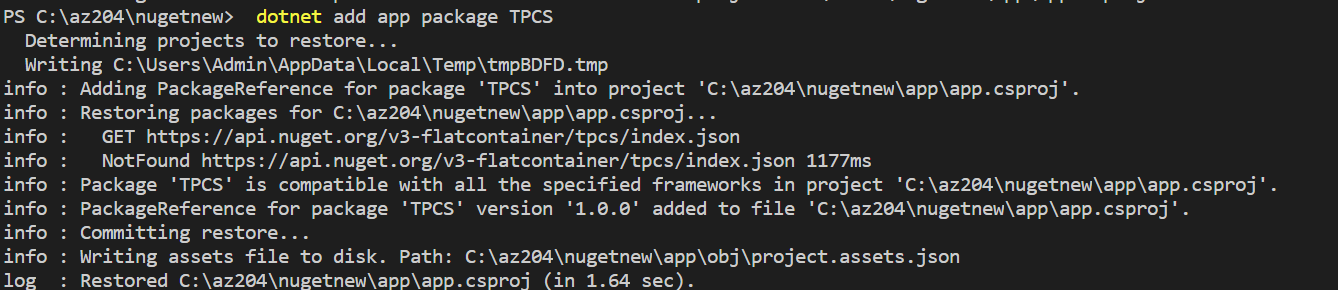
<RestoreSources>$(RestoreSources); C:\az204\nugetnew\library\bin\Debug\; https://api.nuget.org/v3/index.json</RestoreSources>



### **Add the package**

Now that we pointed out where NuGet can find our package we are ready to install it.

**dotnet add app package TPCS**



Now lets try out our installed package. Change Program.cs in your app project to the following code:

using System;

using library;

namespace app

{

class Program

{

static void Main(string[] args)

{

var start = new Start();

Console.WriteLine(string.Format("Hello World! {0}", start.Add(2, 2)));

}

}

}

using System;

using library;

namespace app

{

  class Program

  {

    static void Main(string[] args)

    {

        var start = new Start();

        Console.WriteLine(string.Format("Hello World! {0}", start.Add(2, 2)));

    }

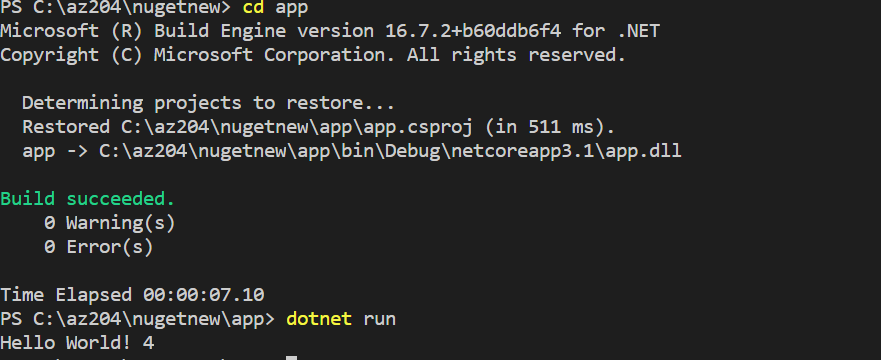
  }

}

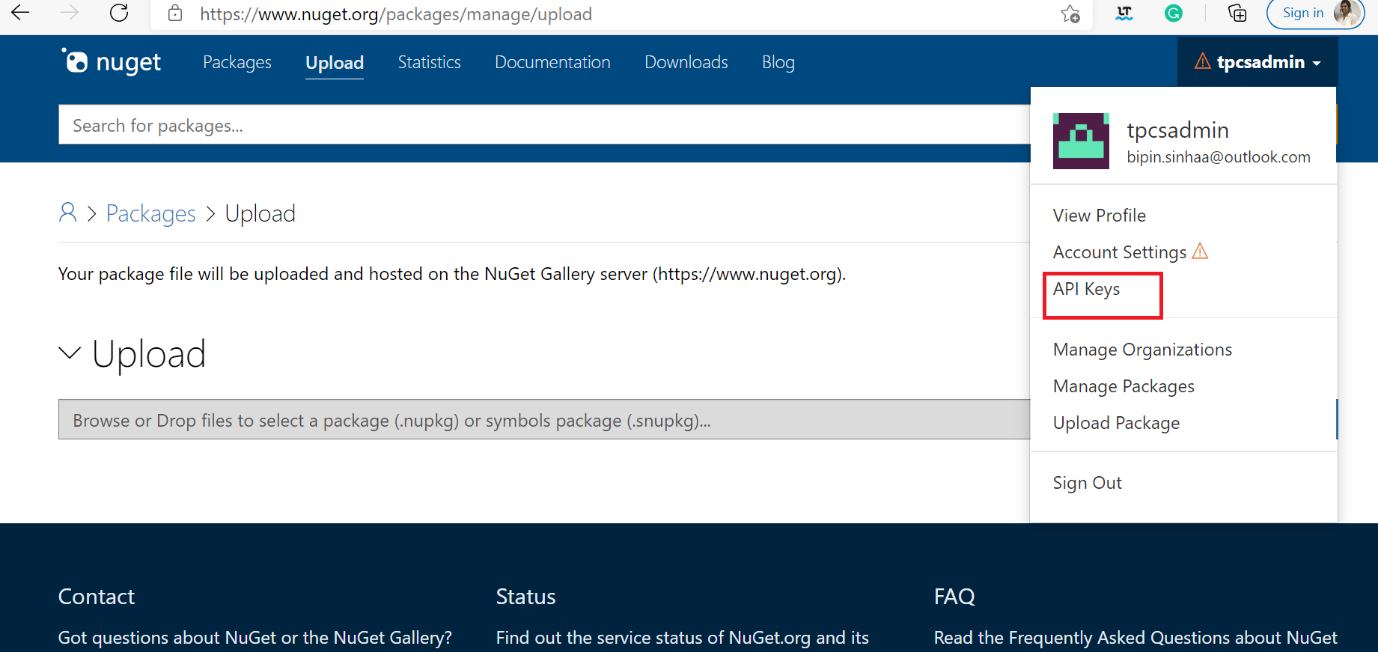
Cd app

Dotnet build

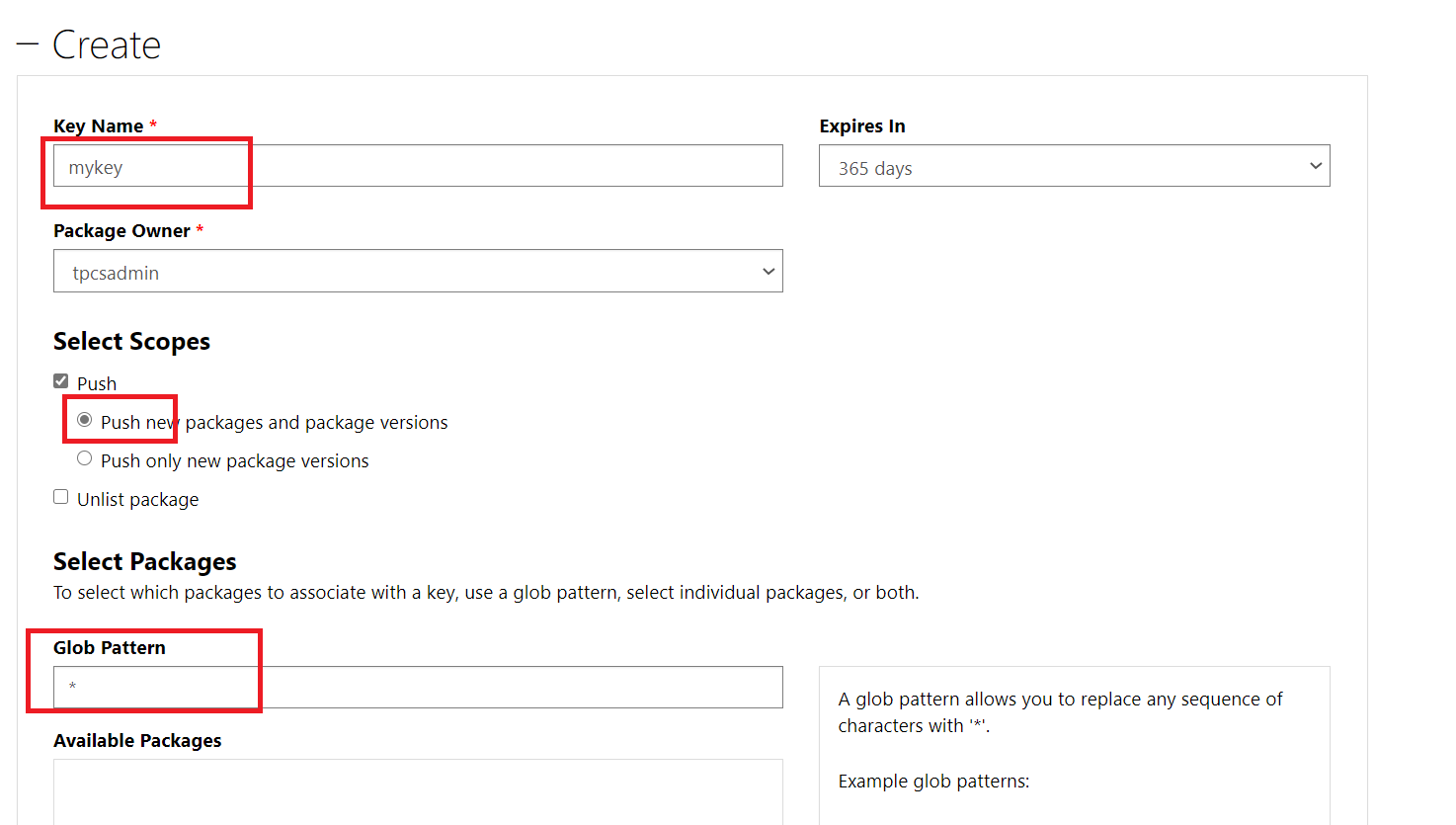
Dotnet run



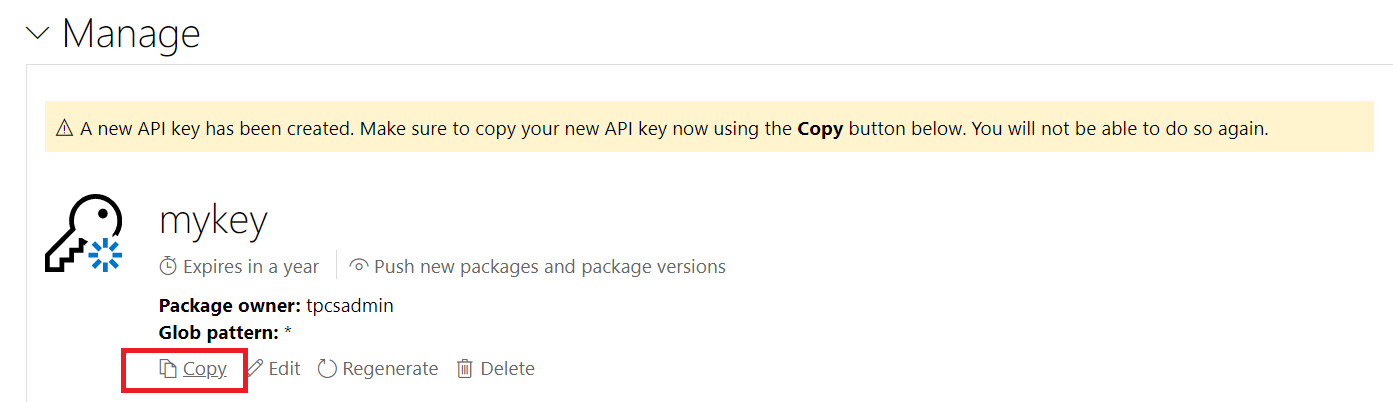
Login to nuget.org and create an API Key



On the page click the + Create and fill in Key Name and on Selected Scopes check Push. On the field Glob Pattern just enter a \*. The last step is clicking the blue Create button. Scroll down the manage now and find the Manage section. Your new key should exist here. Press the Copy button and your key should now be in your clipboard.

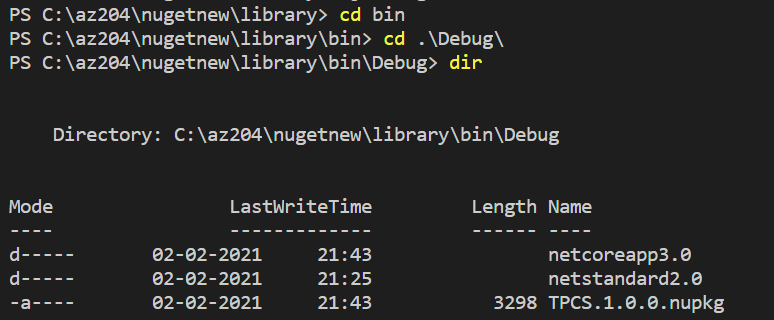


Copy the key



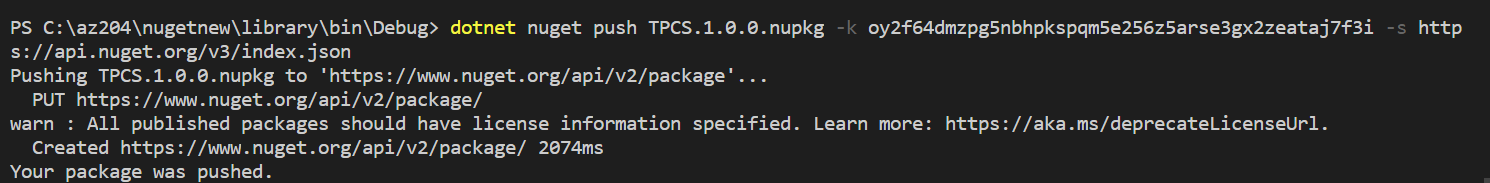
oy2f64dmzpg5nbhpkspqm5e256z5arse3gx2zeataj7f3i

Go to Library/bin/debug and type below command

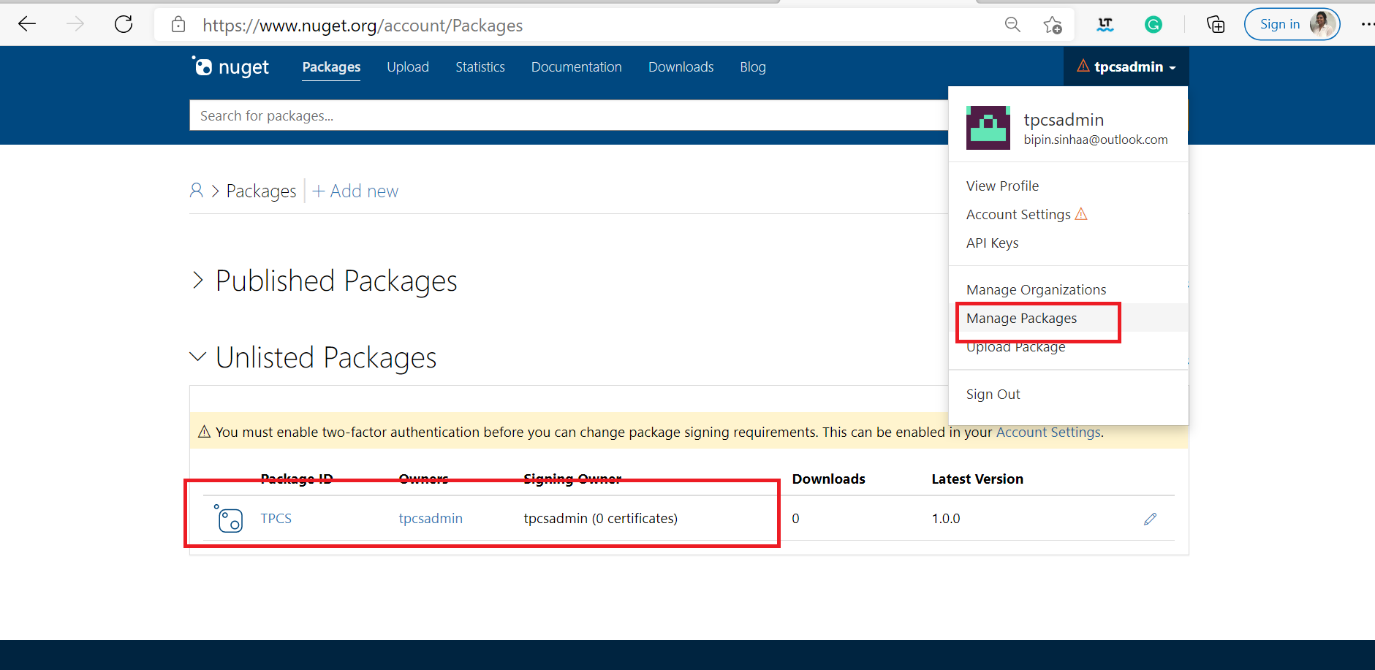


**dotnet nuget push TPCS.1.0.0.nupkg -k oy2f64dmzpg5nbhpkspqm5e256z5arse3gx2zeataj7f3i**

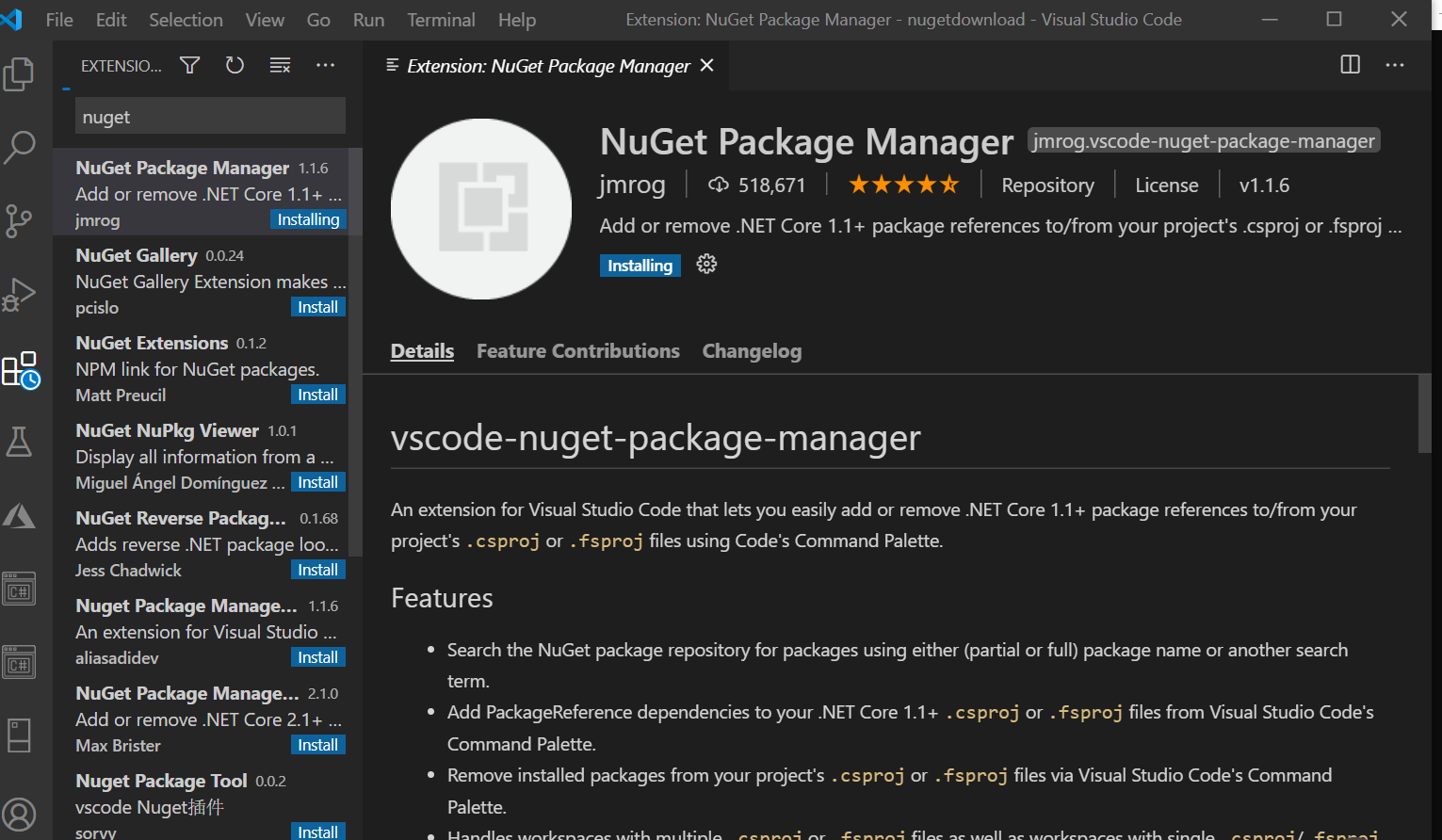
**-s https://api.nuget.org/v3/index.json**



<https://www.nuget.org/packages/TPCS/>

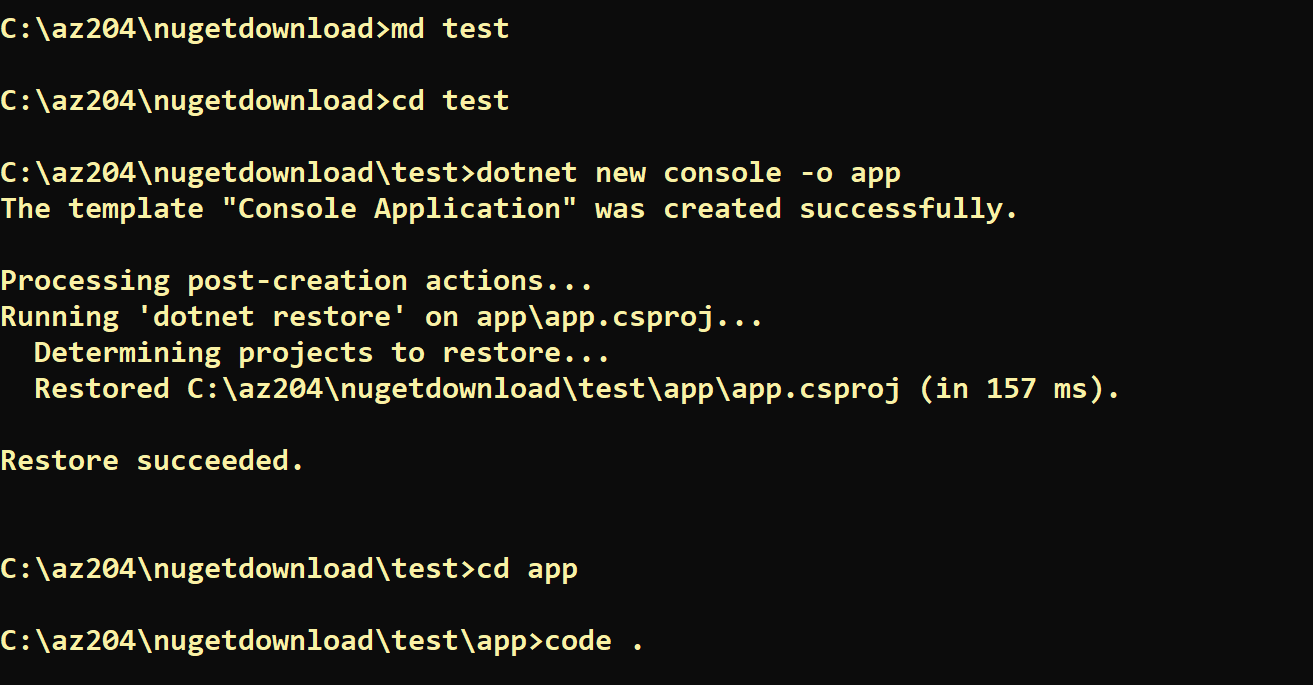


Todpownload



Once you install

Create a new app



And change the code in Program.cs as below

using System;

using library;

namespace app

{

  class Program

  {

    static void Main(string[] args)

    {

        var start = new Start();

        Console.WriteLine(string.Format("Hello World! {0}", start.Add(2, 2)));

    }

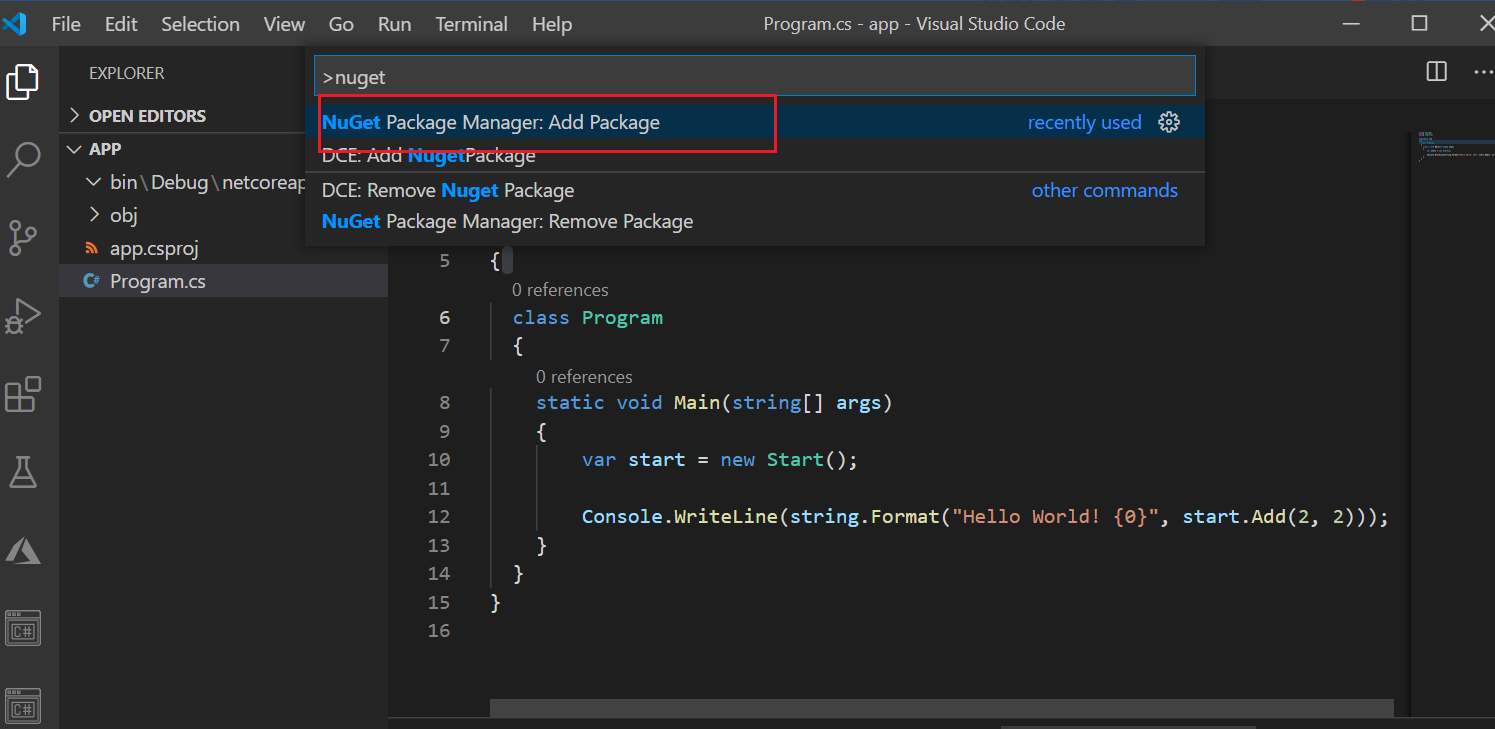
  }

}

It will show error in Library and Start()

Now CTRL+Shift+P

Type nuget and select Package Manager : Add Package



Type TPCS

It will prompt to restore say yes

The error gone

Npw run

Donet build

Dotnet run