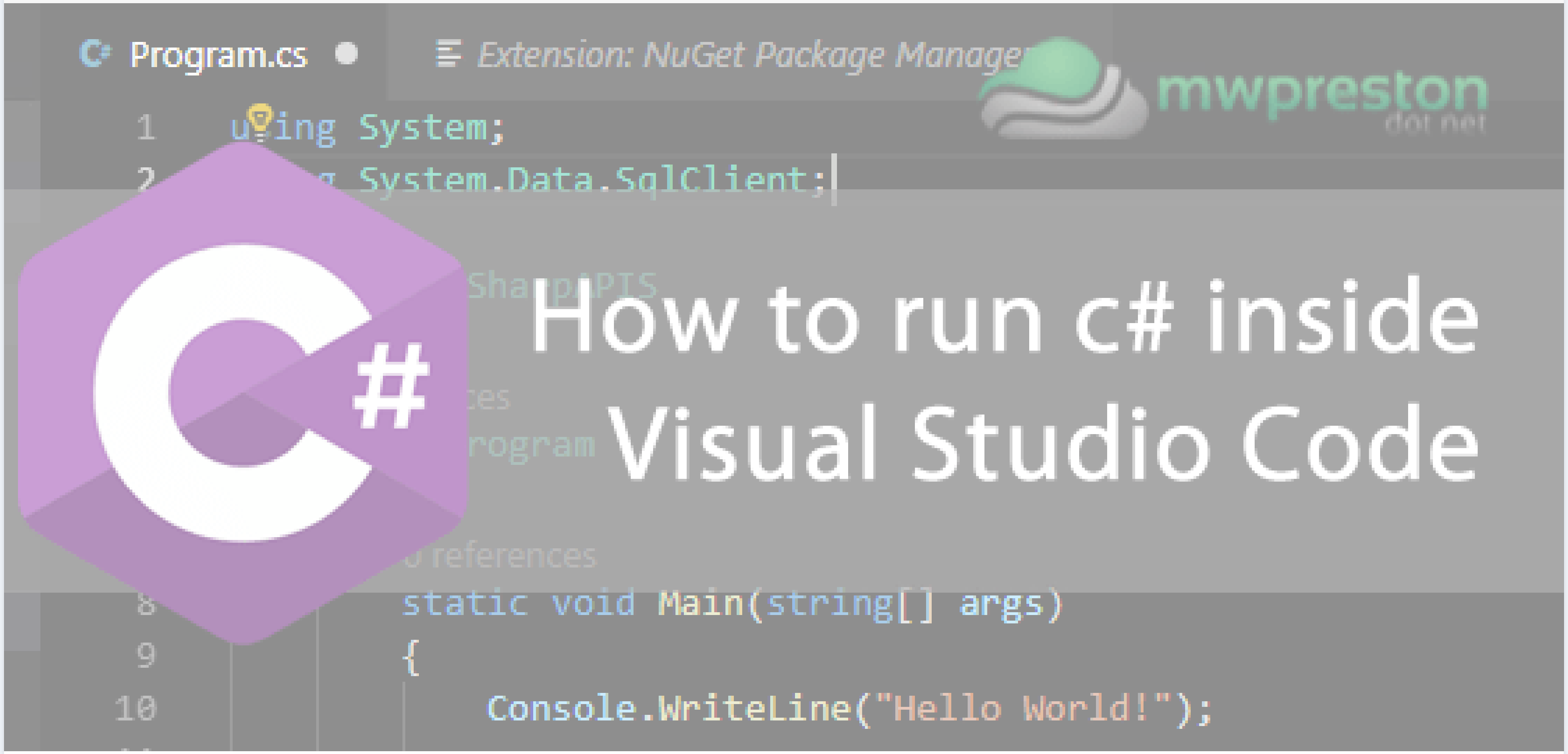
HOW TO RUN C# CODE WITHIN VISUAL STUDIO CODE

Posted by mwpreston | Sep 24, 2018 | All Articles, Automation & Scripting, Posts, September 2018 | 0 |


Program.cs
Extension: NuGet Package Manager
mwpreston dot net
How to run c# inside Visual Studio Code
0 references

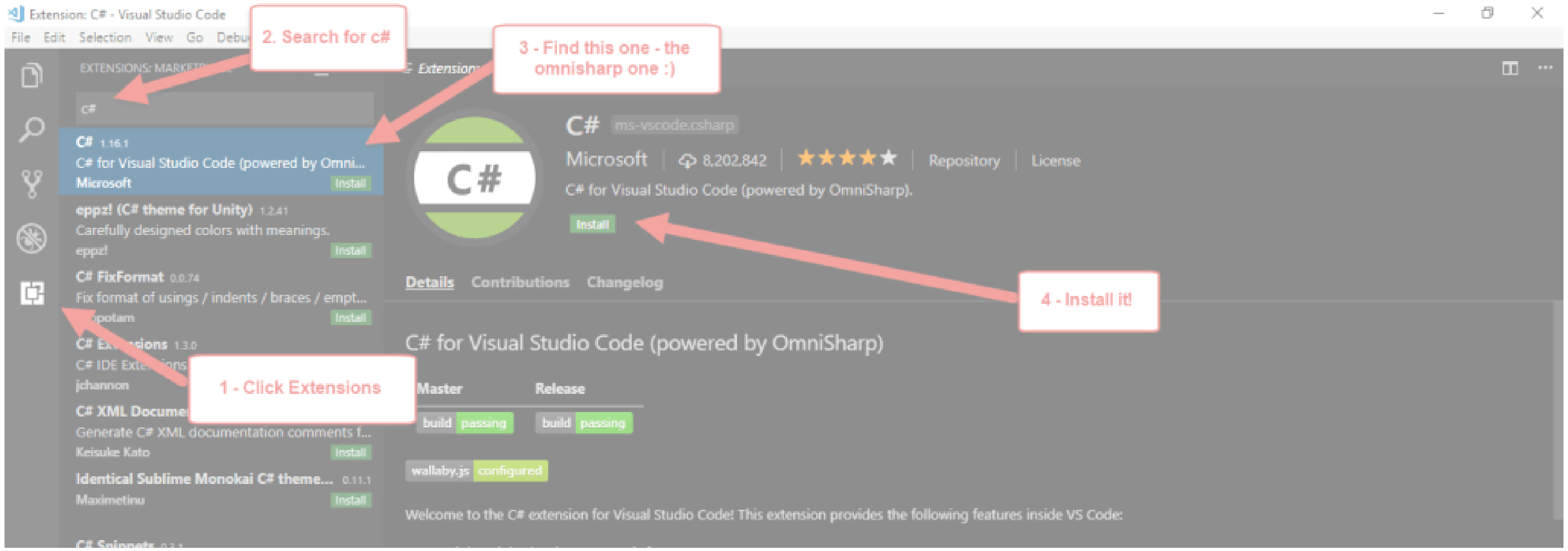
```
using System;
using System.Data.SqlClient;
static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
```

Ever since VS Code was released I've been a big fan! It's a great way for windows junkies like myself to have a terminal like programming scripting environment. With integration into Git, along with thousands of extensions to help me keep my code organized and formatted I began using it as my main editor for a lot of things, even PowerShell scripts.

Currently, I'm on somewhat of a personal mission to learn and explore new languages, solely for the purpose of illustrating how language independent RESTful APIs are. c# is next on my list, and while it's not a new language for me, utilizing it within Visual Studio Code is. During this "mission" I realized that there are quite a few things you need to do in order to run c# within Visual Studio Code. Initially, I was going to place these within the "How to Consume APIs with c#" blog, but realized that this probably works better standing on its own! So, let's take a look at how to prep VS Code to run c#!

First up, some leg work..

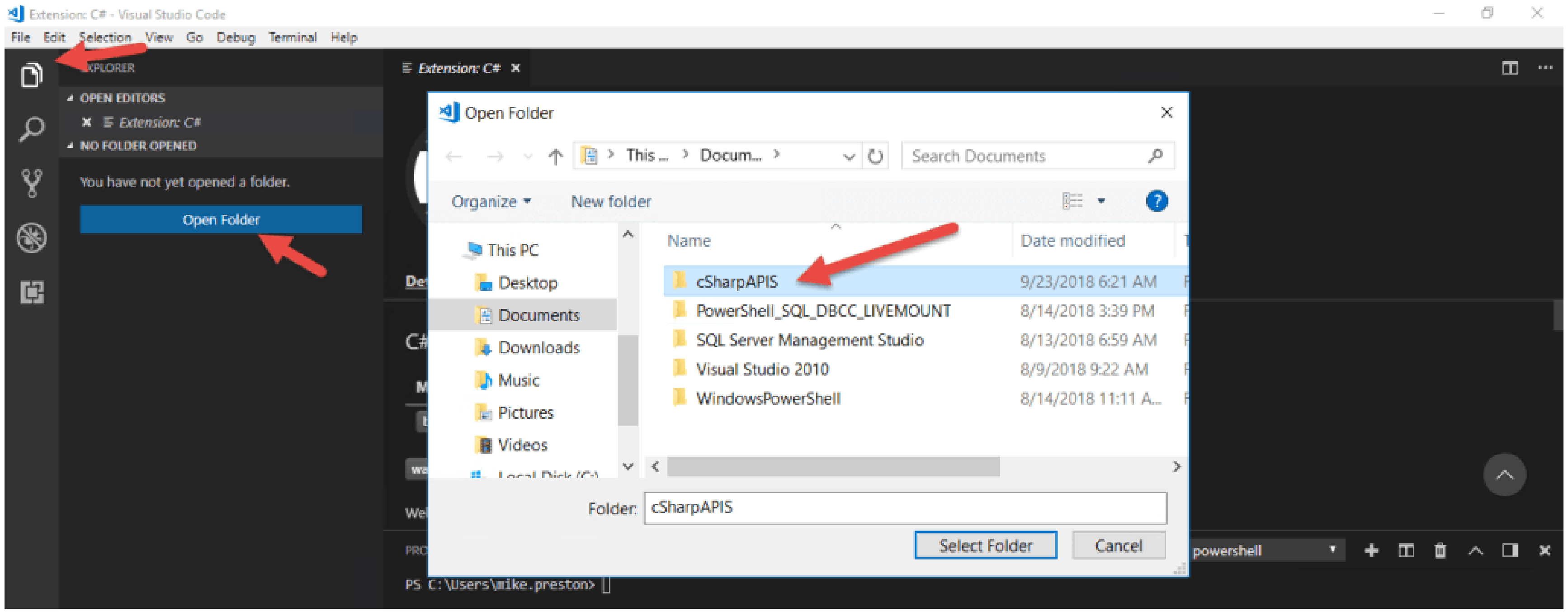
First up, obviously install Visual Studio Code – also, if you haven't already, download and install the . Net Core SDK. Finally, once those two are installed we can move on to getting the c# extension installed within VS Code. Note: there are a lot of extensions available to us within VS Code, so be sure we grab the right one. To do so, select 'Extensions' and search for 'c#'. In the list, select the extension that contains 'Omnisharp' and click 'Install' (See below).



Extension: C# - Visual Studio Code
2. Search for c#
3 - Find this one - the omnisharp one :)
4 - Install it!
1 - Click Extensions
EXTENSIONS: MARKETPLACE
c#
C# 1.16.1
C# for Visual Studio Code (powered by Omni...
Microsoft
Install
C#
ms-vscode.csharp
Microsoft
8,202,842
Repository
License
C# for Visual Studio Code (powered by OmniSharp).
Details
Contributions
Changelog
C# for Visual Studio Code (powered by OmniSharp)
Master
Release
build passing
wallaby.js configured
Welcome to the C# extension for Visual Studio Code! This extension provides the following features inside VS Code:

Initialize the Project

Alright, time to start actually building out the shell for our project – To do this we simply need to open a folder and run a few commands. From the explorer tab, click 'Open Folder' and point to an empty folder you would like to create your project in.



Name	Date modified
cSharpAPIS	9/23/2018 6:21 AM
PowerShell_SQL_DBCC_LIVEMOUNT	8/14/2018 3:39 PM
SQL Server Management Studio	8/13/2018 6:59 AM
Visual Studio 2010	8/9/2018 9:22 AM
WindowsPowerShell	8/14/2018 11:11 A...

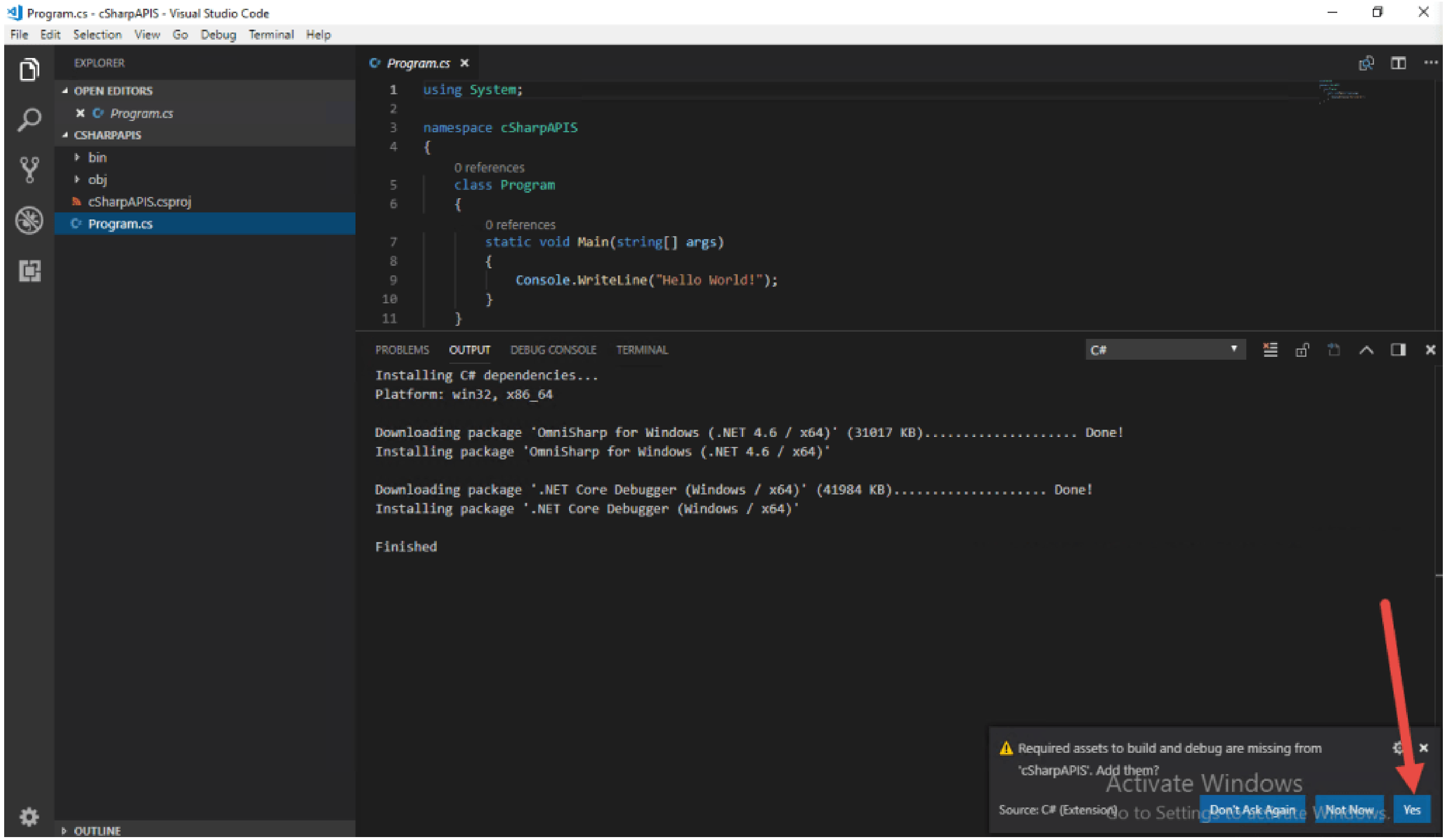
Folder: cSharpAPIS

```
PS C:\Users\mike.preston>
```

Once the folder loads we need to initialize our .net skeleton within it. To do so we will use dotnet command from the terminal. If you don’t see a terminal at the bottom of VS Code, go ahead and select ‘View->Terminal’ to display it and run the following command

```
dotnet new console
```

This will create a couple of files within our folder structure – the project definition file (.csproj) as well as the main file (.cs) to house our code. By default, we will see the standard “Hello World” example. Also, if prompted to download any other assets, go ahead and select ‘Yes’.



```
using System;

namespace cSharpAPIS
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
```

```
Installing C# dependencies...
Platform: win32, x86_64

Downloading package 'OmniSharp for Windows (.NET 4.6 / x64)' (31017 KB).................... Done!
Installing package 'OmniSharp for Windows (.NET 4.6 / x64)'

Downloading package '.NET Core Debugger (Windows / x64)' (41984 KB).................... Done!
Installing package '.NET Core Debugger (Windows / x64)'

Finished
```

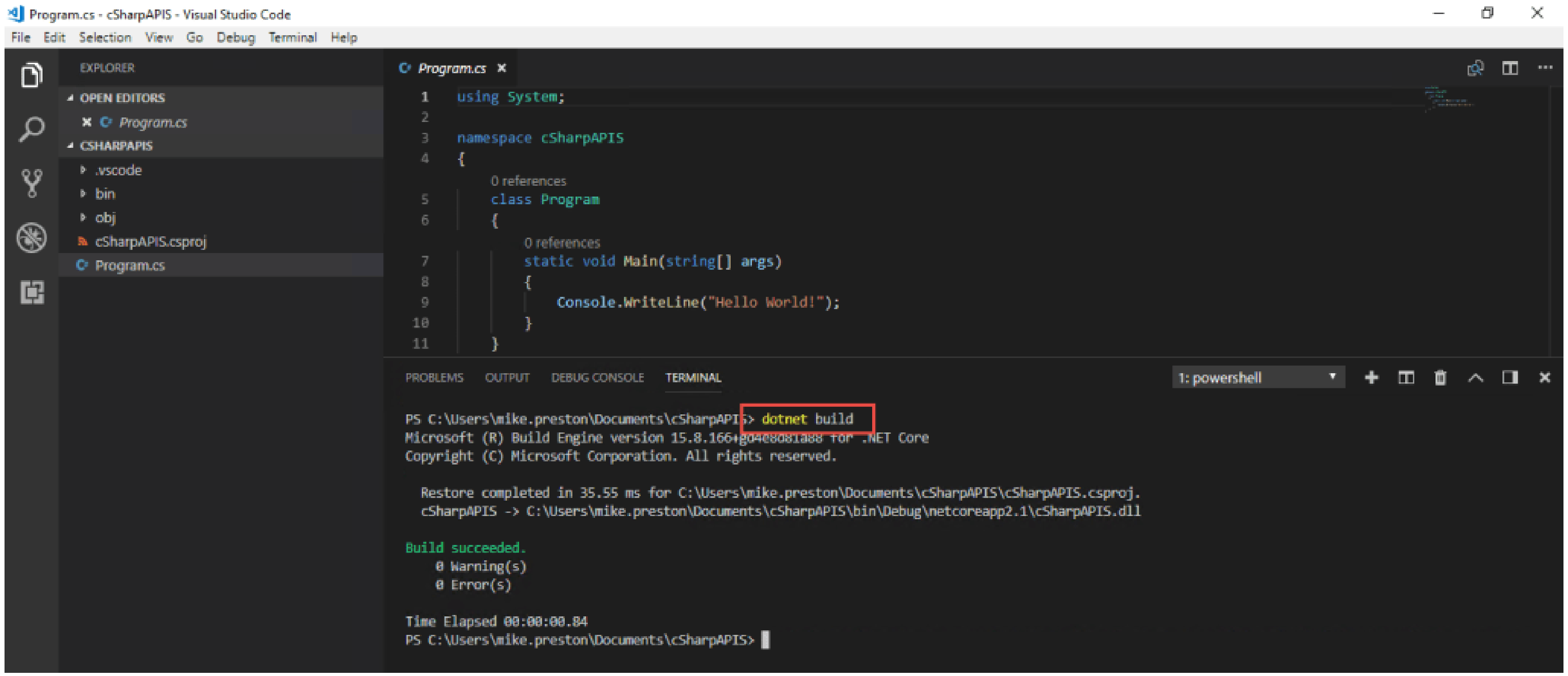
Required assets to build and debug are missing from 'cSharpAPIS'. Add them?

Source: C# (Extension)

If you have reached this point then a big congrats is in order – you have successfully setup your VS Code instance to run c# code!

Building and Running

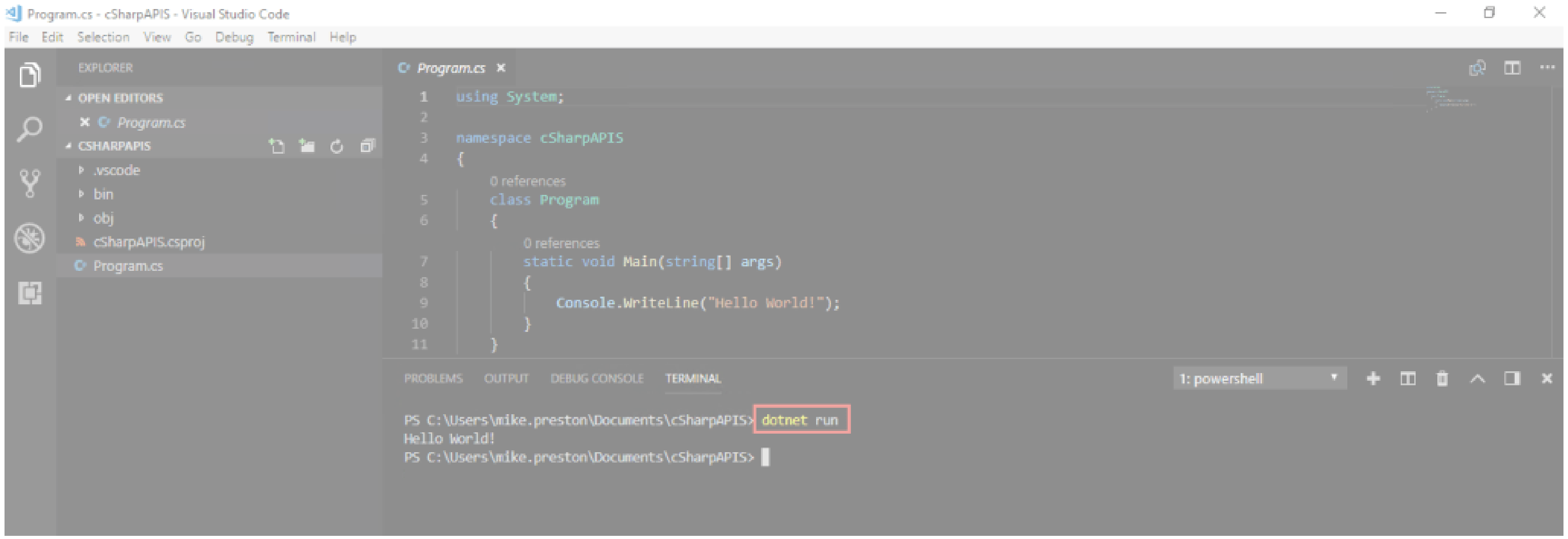
To build our test project let's go ahead and run 'dotnet ***build***' within our terminal.


Program.cs - cSharpAPIS - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
EXPLORER
OPEN EDITORS
Program.cs
CSHARPAPIS
.vscode
bin
obj
cSharpAPIS.csproj
Program.cs
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: powershell

```csharp
using System;

namespace cSharpAPIS
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
```

```
PS C:\Users\mike.preston\Documents\cSharpAPIS> dotnet build
Microsoft (R) Build Engine version 15.8.166+gd4e8d81a88 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 35.55 ms for C:\Users\mike.preston\Documents\cSharpAPIS\cSharpAPIS.csproj.
  cSharpAPIS -> C:\Users\mike.preston\Documents\cSharpAPIS\bin\Debug\netcoreapp2.1\cSharpAPIS.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:00.84
PS C:\Users\mike.preston\Documents\cSharpAPIS>
```

And likewise, to run the project code, simply execute 'dotnet ***run***' within the terminal.


Program.cs - cSharpAPIS - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
EXPLORER
OPEN EDITORS
Program.cs
CSHARPAPIS
.vscode
bin
obj
cSharpAPIS.csproj
Program.cs
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: powershell

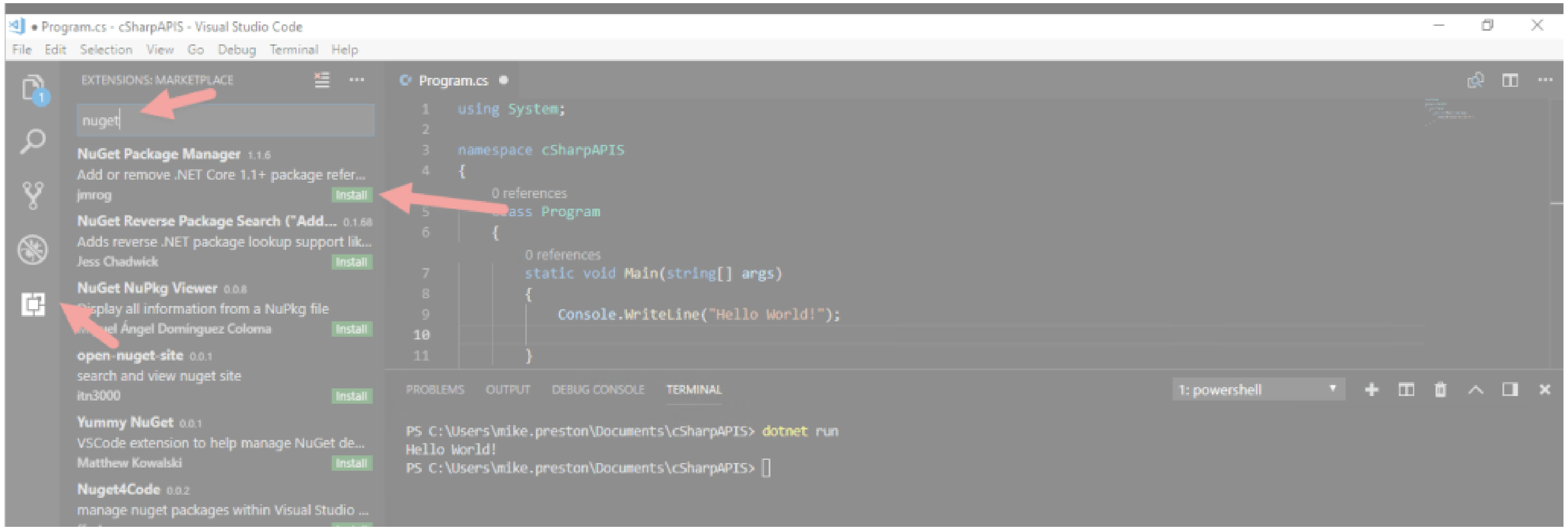
```csharp
using System;

namespace cSharpAPIS
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
```

```
PS C:\Users\mike.preston\Documents\cSharpAPIS> dotnet run
Hello World!
PS C:\Users\mike.preston\Documents\cSharpAPIS>
```

Where's my NuGet?!?!

Simply printing Hello World to the console is great and all but we all know that anything we write is much, much more complex. NuGet is functionality that those using the full-blown version of Visual Studio have come to love and depend on. NuGet is a nifty little package manager that allows us to maintain what packages and assembly references we have attached to our projects, maintaining and limiting versions and conflicts! So, it's kind of a must that we have it within VS Code, otherwise, I'll simply just go back to the original Visual Studio for my c# projects. Thankfully, we have a way of getting it – the nifty Visual Studio Code extensions!

Select the Extensions tab, search for Nu-Get and install NuGet Package Manager – Easy Peasy!


● Program.cs - cSharpAPIS - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
EXTENSIONS: MARKETPLACE
nuget
Program.cs
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: powershell

NuGet Package Manager 1.1.6
Add or remove .NET Core 1.1+ package refer...
jmrog Install

NuGet Reverse Package Search ("Add... 0.1.68
Adds reverse .NET package lookup support lik...
Jess Chadwick Install

NuGet NuPkg Viewer 0.0.8
Display all information from a NuPkg file
Miguel Ángel Domínguez Coloma Install

open-nuget-site 0.0.1
search and view nuget site
itn3000 Install

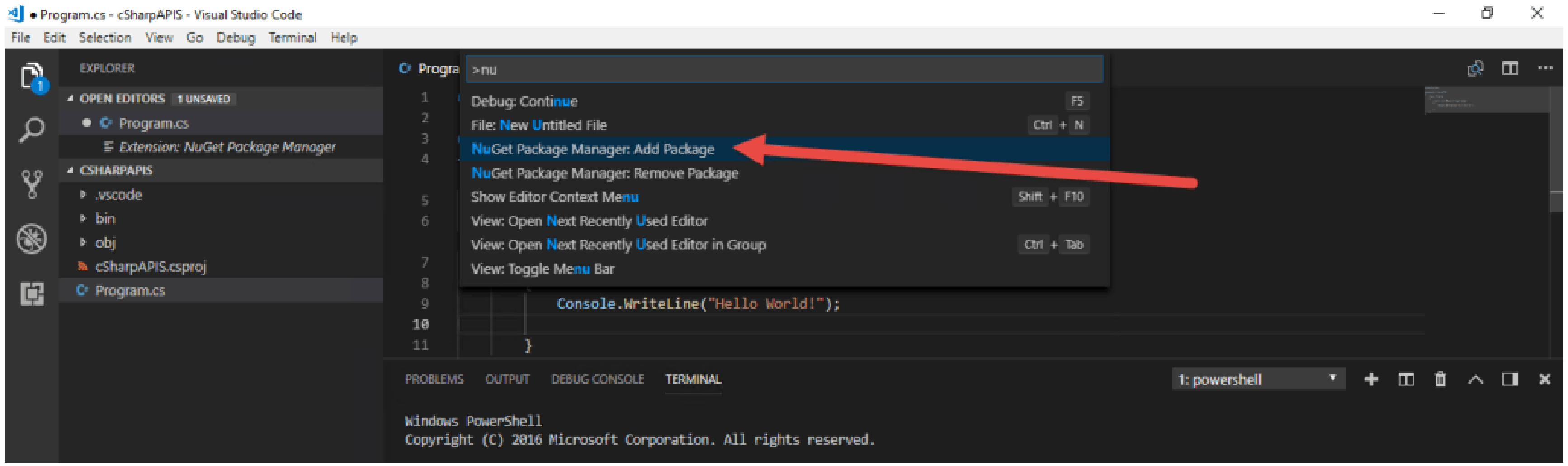
Yummy NuGet 0.0.1
VSCode extension to help manage NuGet de...
Matthew Kowalski Install

Nuget4Code 0.0.2
manage nuget packages within Visual Studio ...

```
using System;

namespace cSharpAPIS
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");

        }
```

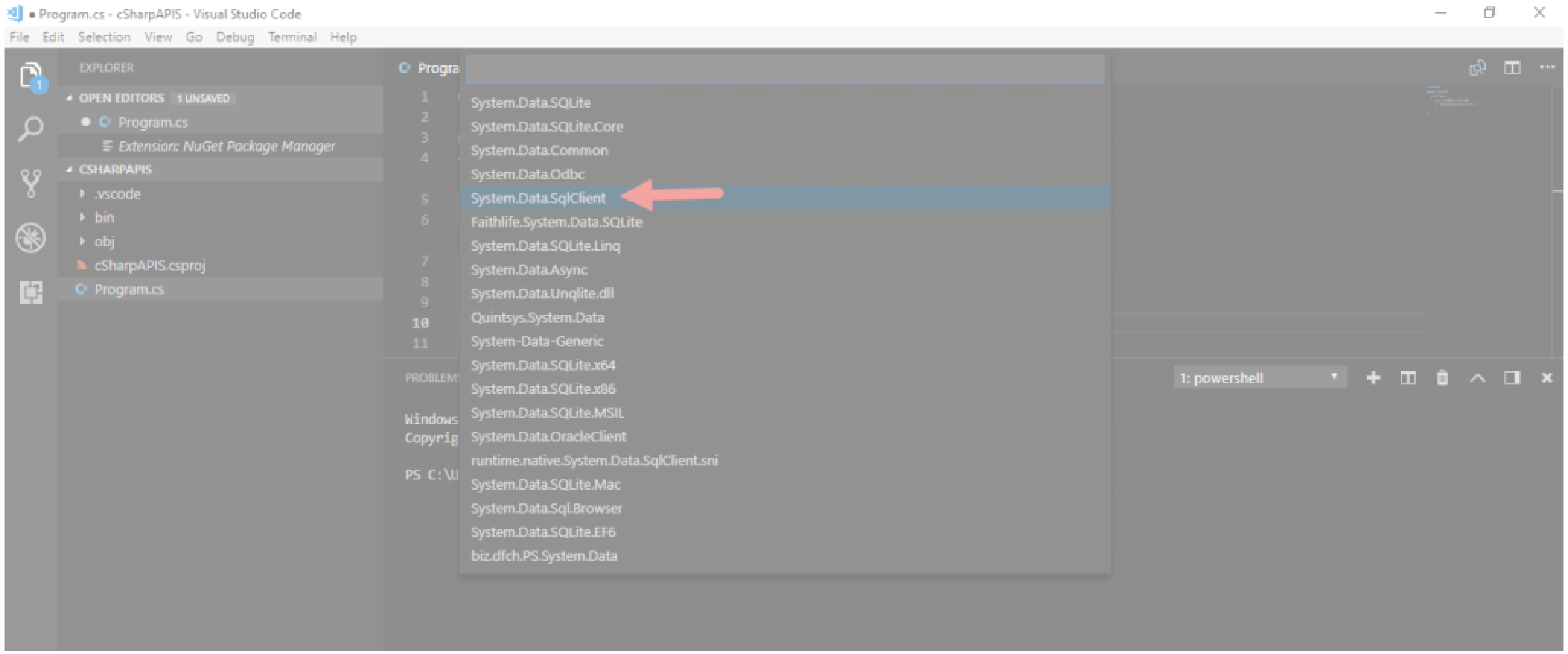
```
PS C:\Users\mike.preston\Documents\cSharpAPIS> dotnet run
Hello World!
PS C:\Users\mike.preston\Documents\cSharpAPIS> 
```

Now that we have NuGet installed let's have a quick look at how we can manage our packages – the easiest way that I have found is to simply hit the 'F1' key to bring up the VS Command pallet – a nifty little way to run single commands within our project. From there, simply start typing the work 'nuget', you should see the prompts displaying available options – the one we want, is the 'Add Package' command.



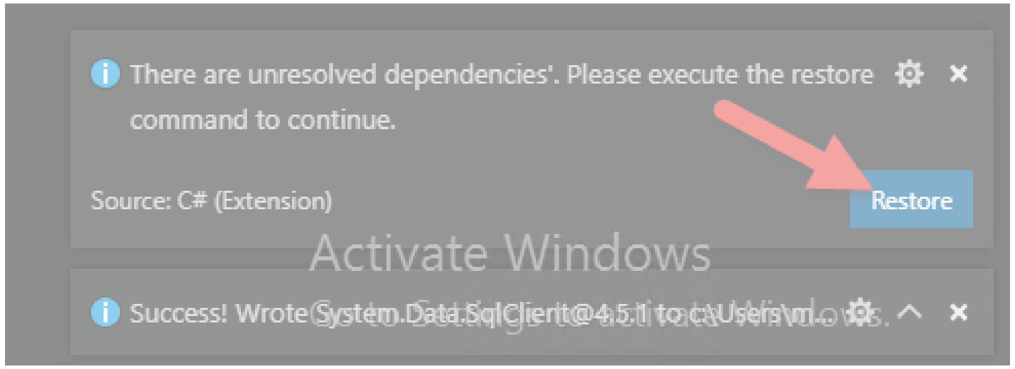
```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.
```

For this example let's add the System.Data.SQLClient package to our project. In the first prompt, you can simply input 'System.Data' and hit enter – this will bring up a list of all the system.data packages. From the list displayed, select your desired package, in our case SQLClient, select the desired version and away we go.



```
System.Data.SQLite
System.Data.SQLite.Core
System.Data.Common
System.Data.Odbc
System.Data.SqlClient
Faithlife.System.Data.SQLite
System.Data.SQLite.Linq
System.Data.Async
System.Data.Unqlite.dll
Quintsys.System.Data
System-Data-Generic
System.Data.SQLite.x64
System.Data.SQLite.x86
System.Data.SQLite.MSIL
System.Data.OracleClient
runtime.native.System.Data.SqlClient.sni
System.Data.SQLite.Mac
System.Data.Sql.Browser
System.Data.SQLite.EF6
biz.dfch.PS.System.Data
```

In the event that any dependencies are invalid, nuget and VS Code gives you the option to automatically resolve them. If you happen to see a message about unresolved dependencies go ahead and click 'Restore'


There are unresolved dependencies'. Please execute the restore command to continue.

Source: C# (Extension)

Activate Windows

Success! Wrote System.Data.SqlClient@4.5.1 to c:\Users\m...

And there you go – nuget successfully setup and installing packages for you! At this point, you can go ahead and through your 'using System.Data.SqlClient' command at the beginning of your cs files and start utilizing some of the SQL Client functionality!

As you can see there is a little bit of legwork in getting Visual Studio Code prepped and ready to run our c# code – but thankfully this is a one-time thing. Once you have it set up and configured you can use VS Code for all of your c# needs! Honestly, I gathered this information from a lot of different MS pages and blogs and hopefully getting this all on one article will help get you on the right track to getting your VS Code setup! Thanks for reading!

You Might Also Enjoy These Recent Posts...

1. **Friday Shorts – Visual Studio 2010 and BIDS, AJAX Toolkit Combo Boxes and the Great One** No support for BIDS development in Visual Studio 2010 Ahh Man! I ended up rebuilding one of my work machines...
2. **Quick to the point – VMware Tools Installation throwing vix error code 21012 – yeah, now what?!?** Ah, the subtleties of upgrading a vSphere environment – tackle the vCenter, tackle the hosts, and then take care of the...
3. **Dumping SharePoint Integrated SSRS reports to pdf using c#** Anytime I'm working on any sort of coding project Google is really my savior – there is absolutely no way...
4. **VMware View on the $99 HP Touchpad** Last month, being the quintessential geek that I am, I spent the greater part of one of my weekends installing...
5. **Installing offline bundles in ESXi 5** So I ran into a situation where I needed to get the HP CIM providers onto ESXi 5 in order...

SHARE: RATE:

< PREVIOUS

Consuming RESTful APIs with Ruby and RestClient

>NEXT

Consuming RESTful APIs with c#

ABOUT THE AUTHOR

mwpreston

Tech Enthusiast from Canada eh! vExpert, Author of Troubleshooting vSphere Storage and Toronto VMUG Leader! That's enough for now :)

RELATED POSTS

Free NFR from Altaro

August 3, 2016

The number of virtual devices exceeds the maximum for a given controller!!

November 18, 2011

A few updates to the Veeam silent install!

May 26, 2016

Free Course – VMware vCenter Operations Manager Fundamentals

June 7, 2012

Search …

Advertise Here

TOP POSTS & PAGES

How to run c# code within Visual Studio Code

Quick Fix - Making your inactive NFS datastore active again!

Learning 3PAR - Part 1 - Chunklets, Logical Disk, CPGs, and Virtual Volumes

Quick to the point - A specified parameter was not correct spec.vmProfile in Veeam

Expanding a Linux disk with gparted (and getting swap out of the way)

A newbies guide to ELK - Part 2 - Forwarding logs

Consuming RESTful APIs with Python

No vMotion for you! - A general system error occurred: vim.faultNotFound

Manually updating the Veeam Proxy Transport and Mount services

VMware View - Reset, Refresh, Rebalance, Recompose, Rehuh?

DISCLAIMER