Server & Tools Blogs > Developer Tools Blogs > .NET Web Development and Tools Blog

Sign in

.NET Web Development and Tools Blog

Your official information source from the .NET Web Development and Tools group at Microsoft.

Visual Studio

Visual Studio 2017 Tools for Azure Functions

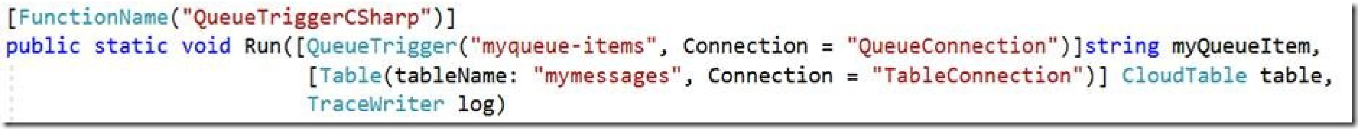
★★★★☆

May 10, 2017 by Andrew B Hall - MSFT // 84 Comments

Share 160 246 80

Today we're pleased to announce the release of the first preview of Visual Studio 2017 Tools for Azure Functions. This preview introduces some exciting changes over our previous release. Beyond bringing support for Visual Studio 2017, this release:

- Enables creating pre-compiled C# functions that bring better cold start performance than script based functions, and opens the entire eco-system of Visual Studio tools for class libraries including code analysis, unit testing, complete IntelliSense, 3rd party extensions, etc.
- Uses WebJobs attributes to declare function bindings directly in the C# code rather than the separate function.json file.



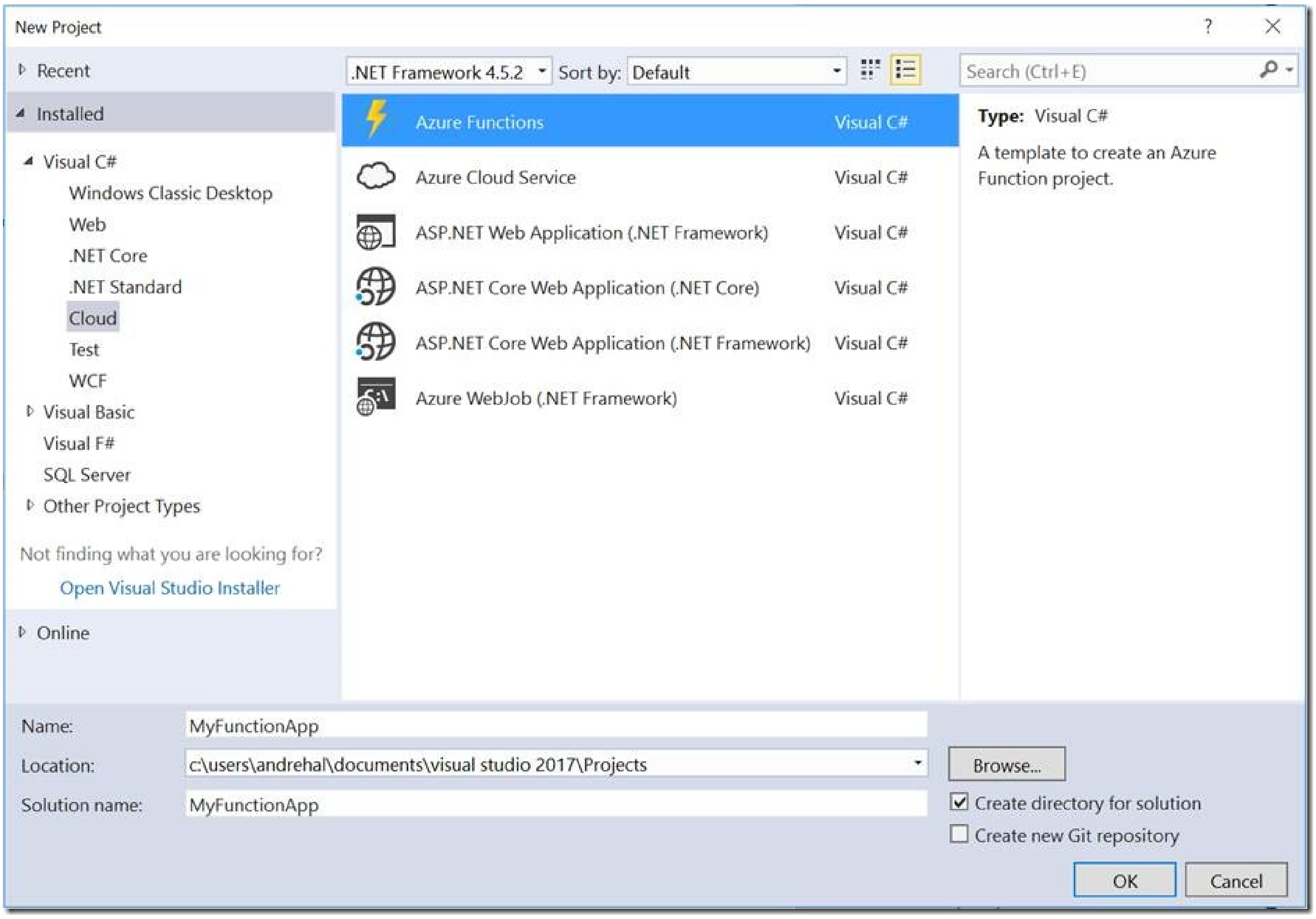
```
[FunctionName("QueueTriggerCSharp")]
public static void Run([QueueTrigger("myqueue-items", Connection = "QueueConnection")]string myQueueItem,
                       [Table(tableName: "mymessages", Connection = "TableConnection")] CloudTable table,
                       TraceWriter log)
```

Getting Started

To get started:

- You need to install Visual Studio 2017 (version 15.3) Preview (it will not work with any earlier versions of Visual Studio)
- You must have either the "ASP.NET and web development" or "Azure development" workload installed
- Download and install the Visual Studio 2017 Tools for Azure Functions extension

To create a new project, choose **File -> New Project**, and the **Azure Functions** project type


New Project
?
Recent
Installed
Visual C#
Windows Classic Desktop
Web
.NET Core
.NET Standard
Cloud
Test
WCF
Visual Basic
Visual F#
SQL Server
Other Project Types
Not finding what you are looking for?
Open Visual Studio Installer
Online
Search (Ctrl+E)
Azure Functions Visual C#
Azure Cloud Service Visual C#
ASP.NET Web Application (.NET Framework) Visual C#
ASP.NET Core Web Application (.NET Core) Visual C#
ASP.NET Core Web Application (.NET Framework) Visual C#
Azure WebJob (.NET Framework) Visual C#
Browse...
OK
Cancel

.NET Framework 4.5.2 Sort by: Default

Type: Visual C#

A template to create an Azure Function project.

Name: MyFunctionApp

Location: c:\users\andrehal\documents\visual studio 2017\Projects

Solution name: MyFunctionApp

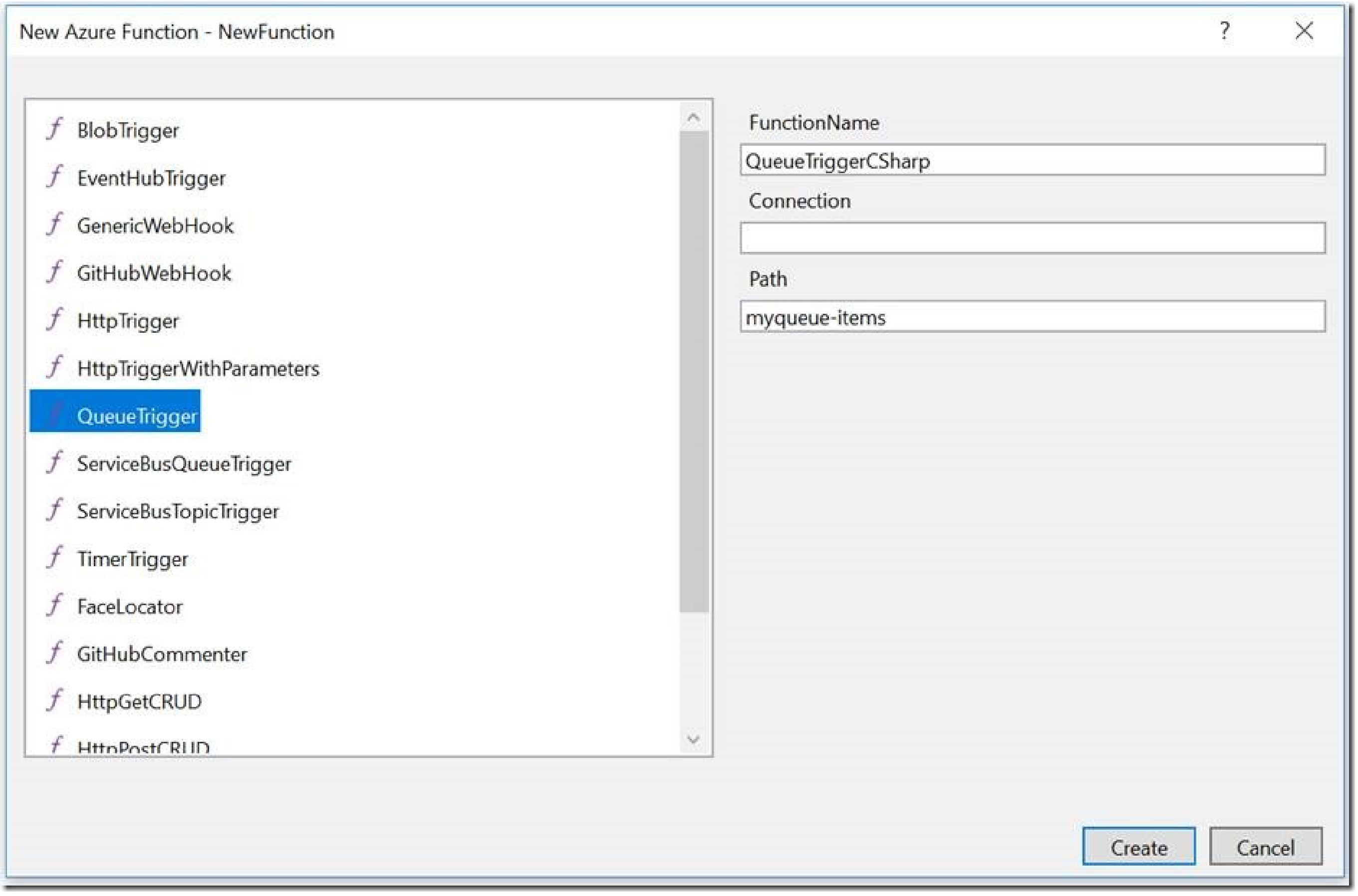
☑ Create directory for solution

□ Create new Git repository

This will create an empty project which contains the following files:

- **host.json** enables configuring the function host
- **local.settings.json** which stores setting information such as connection strings used for running the function on the development machine. *Note: For all trigger types except HTTP, you need to set the value of AzureWebJobsStorage to a valid Azure Storage account connection string.*

To add a function to the application right click the project and choose "Add Item", then choose the "Azure Function" item template. This will launch the Azure Function dialog that enables you to choose the type of function you want, and enter any relevant binding information. For example, in the dialog below, the queue trigger asks you for the name of the function, the name of the connection string to the storage queue, and the name of the queue (path).



FunctionName: QueueTriggerCSharp

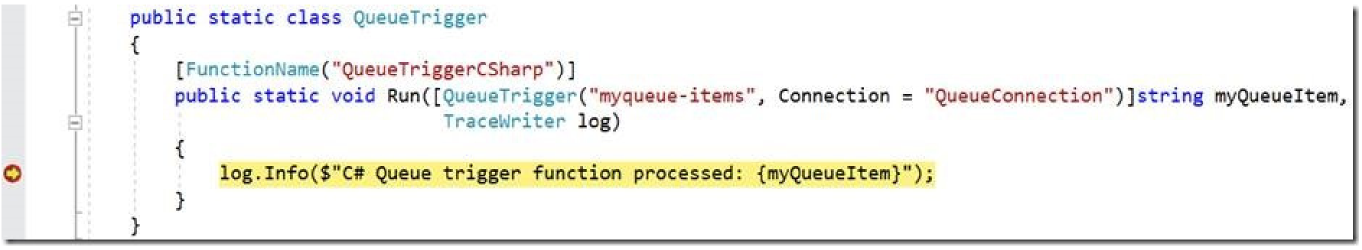
Connection:

Path: myqueue-items

This generates a new class that has the following elements:

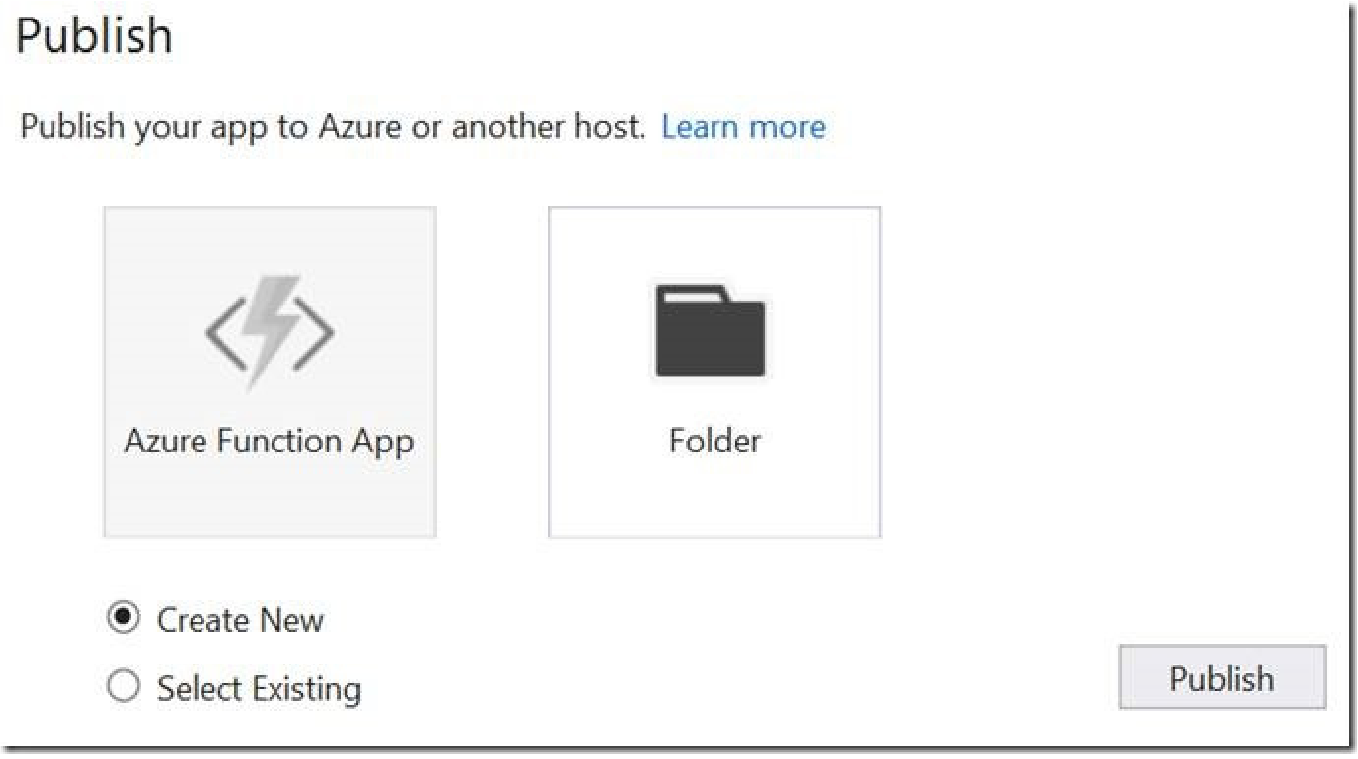
- A static Run method, that is attributed with [FunctionName] attribute. The [FunctionName] attribute indicates that the method is the entry for an Azure Function.
- The first parameter has a QueueTrigger attribute, this is what indicates is a queue trigger function (and takes the binding information as parameters to the attribute. In this case the name of the queue and the connection string's setting name)

Once you have a function, local development works like you would expect. You can run and debug it locally, add NuGet packages, create unit tests, and anything else you would do for a class library.



```
public static class QueueTrigger
{
    [FunctionName("QueueTriggerCSharp")]
    public static void Run([QueueTrigger("myqueue-items", Connection = "QueueConnection")]string myQueueItem,
                           TraceWriter log)
    {
        log.Info($"C# Queue trigger function processed: {myQueueItem}");
    }
}
```

To publish a Function project to Azure directly from Visual Studio, right click the project and choose "Publish". On the publish page, you can either create a new Function App in Azure or publish to an existing one. Note: even though the Folder option is currently appears, it's not intended for use with Azure Functions at this time.


Publish

Publish your app to Azure or another host. Learn more

● Create New

○ Select Existing

It's also possible to configure continuous delivery using Visual Studio Team Services.

Common questions

Here are answers to come common questions:

What's next for Azure Function tools? Our goal is to make these a part of Visual Studio 2017 without the need to install a separate extension.

How can I file issues or provide feedback on these preview tools? You can file issues or provide feedback on the Azure Functions GitHub repo and prefixing them with *[Visual Studio]*

Are these targeting .NET Standard 2.0 as outlined in the roadmap post? This first preview is producing .NET 4.6.1 class libraries. It's still our intent to default to .NET Standard 2.0 in the future, but given that .NET Standard is releasing its first preview at the same time as these tools, it isn't quite ready for use in the Azure Functions runtime yet.

I have existing functions written as .csx scripts, how do I port those to the new precompiled project type? To convert a .csx file into a new function you will need to move the Run method into a class, remove #load, and replace #r with assembly or project to project references (see complete steps).

What about support for Visual Basic and F#? It will be possible to create Azure Functions using Visual Basic and F# in a future update, but support is not currently included in this release.

What is the plan for the Visual Studio 2015 tools? The Visual Studio 2015 tooling was an initial preview that got us a lot of great feedback and we learned a lot from them. Given our pivot to pre-compiled functions with the intent to focus on .NET Standard 2.0, we have dependencies that will only exist in Visual Studio 2017 Update 3 and beyond, so there are no plans to release any future updates for Visual Studio 2015. Once we complete the .NET Standard 2.0 work, it will be possible to work with Azure Functions in Visual Studio Code as well as Visual Studio 2017 if you prefer or are unable to upgrade to Visual Studio 2017.

Conclusion

We know that it's been a while coming, but we hope that your try the Visual Studio 2017 Tools for Azure Functions and let us know how they work for you. You can do that below, in the Azure Functions GitHub repo, or via twitter at @AndrewBrianHall and @AzureFunctions.

Search MSDN with Bing

○ Search this blog ● Search all blogs

ASP.NET Resources

www.ASP.Net

ASP.NET Forums

Web Developer Checklist

ASP.NET User Voice

.NET Meetups

Related Blogs

Scott Guthrie's Blog

Scott Hanselman's Blog

Sayed Hashimi's Blog

Jon Galloway's blog

Mads Kristensen's blog

Entity Framework Team blog

Brady Gaster's Blog

NuGet Team Blog

Jeff Fritz's Blog

Vittorio Bertocci's Blog

Pages

We are Hiring

Share This Post