–

**Azure Function with Queue**

# **Microsoft Azure + Storage Account + Azure Function + MVV**

# **Microsoft Azure + Storage Account + Azure Function + MVV**

**Azure Function Listening to a Queue**

**Azure Function Listening to a Queue**

PREPARED BY:

**Bipeen Sinha**Cloud Architect

PREPARED FOR:

**Simply Learn**Bangalore

**Version:** 1.0  
**Department:** Architect Department

Student Project Work

Student Project Work

Table of Contents

[**1.** **The Project Summary** 3](#_Toc54353829)

[**2.** **The Proposed Solutions (Logical Architecture)** 3](#_Toc54353830)

[**3.** **The Proposed Test-bed Scenario and High Level Steps:** 3](#_Toc54353832)

[**4.** **The Technical Execution Steps** 4](#_Toc54353833)

[Step1: Azure Portal Sign-in and Create Required Resources 4](#_Toc54353834)

[**Task 1: Sign in to the Azure Portal** 4](#_Toc54353835)

[**Task 2: Create the Resource Group** 4](#_Toc54353836)

[**Task 3: Create the Storage Account** 5](#_Toc54353837)

[**Task 4: Create a queue for input messages** 6](#_Toc54353838)

[Step 2: Create Azure Function using visual studio 7](#_Toc54353839)

[**Task 1: Create Azure Function App** 7](#_Toc54353840)

[**Task 2: Create Azure Function with queue trigger** 9](#_Toc54353841)

[**Task 3: Publish the function to Azure** 11](#_Toc54353842)

[**Task 4: Verify the functionality** 16](#_Toc54353843)

# **The Project Summary**

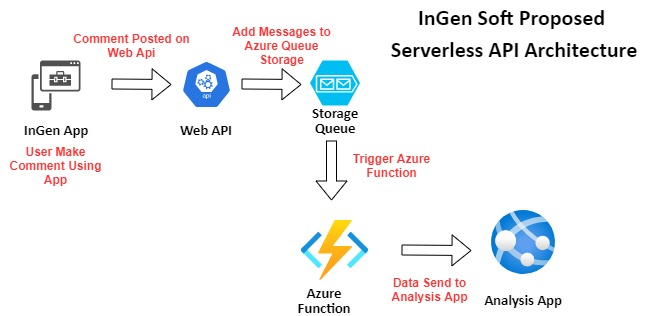
**Project Title** :

Azure Function Listening to a Queue

**Business Scenario** :

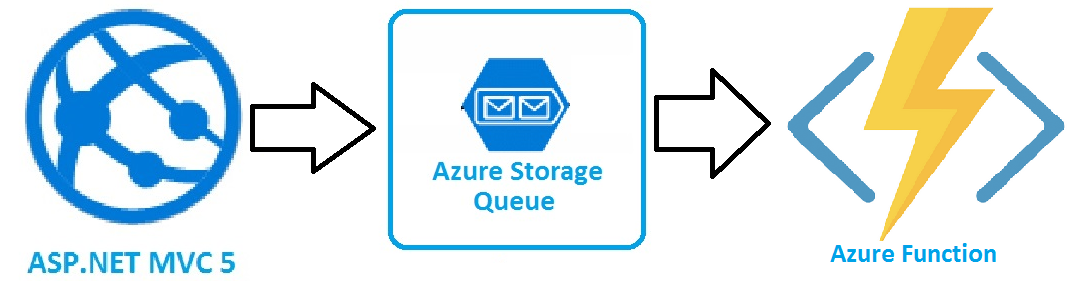
The **InGen Soft** company has a app for its customer , The app include an features which capture the Comment of Users and send it to Analysis App which is hosted on Azure App services . Company is facing a challenge in terms of efficiency while receiving the comment messages as it sometime add lags. So they decided to use the **Azure Queue** to store the message and use **Azure functions** to send the data to Analysis Service. The idea behind this is that the appearance of the message inside a Queue shall trigger the Function execution.

# **The Proposed Solutions (Logical Architecture)**

****

# **The Proposed Test-bed Scenario and High Level Steps:**

**The Test-bed Architecture:**



**The High Level Execution Steps**

For validation, this task has been assigned to a developer to test this functionality using Visual Studio 2017. Develop can perform Test with following High Level Steps

1. Log in to the Azure Portal with Admin Privilege
2. Create Following resources
   1. Resource Group
   2. Storage Account
   3. Input Queue services
3. Create an Azure function using Visual Studio which also includes Queue Binding and Deployment from visual studio.

# **The Technical Execution Steps**

## Step1: Azure Portal Sign-in and Create Required Resources

### **Task 1: Sign in to the Azure Portal**

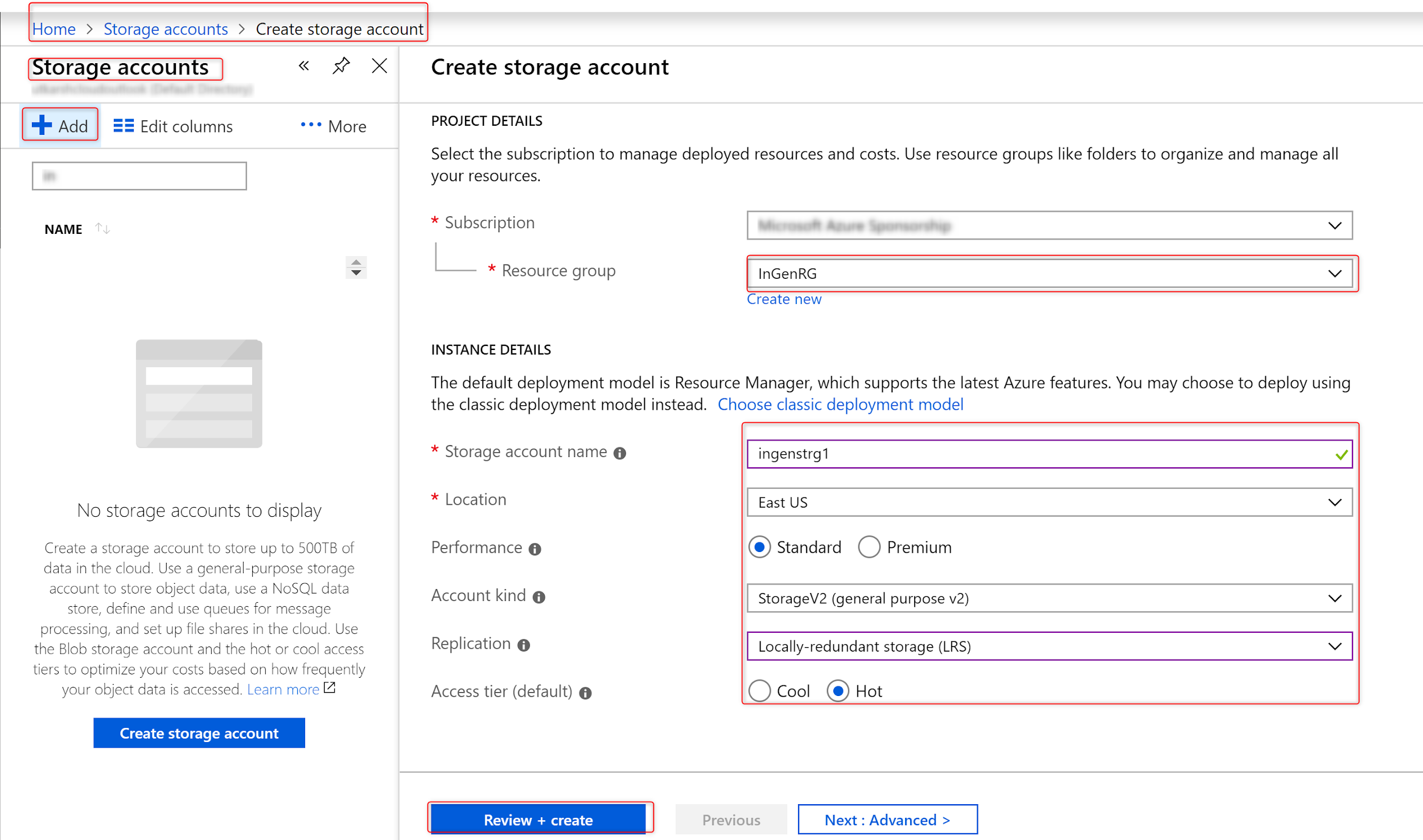
1. On the Start screen, click the **Internet Explorer** tile.
2. Go to *(*[*https://portal.azure.com*](https://portal.azure.com)*)*.
3. Enter the email address of your Microsoft account. Click **Next**.
4. Enter the password for your Microsoft account.
5. Click **Sign In**.

### **Task 2: Create the Resource Group**

1. In the navigation pane on the left side of the Azure Portal, click **All services**.
2. In the **All services** blade that displays, click **Resource groups**.
3. In the **Resource groups** blade that displays, view your list of resource groups.
4. At the top of the **Resource groups** blade, click the **Add** button.
5. In the **Resource group** blade, perform the following steps:
   1. In the **Resource group name** dialog box, provide the value **InGenRG**.
   2. In the **Resource group location** list, select **East US**.
6. In the **Resource group** blade, click **Create**.

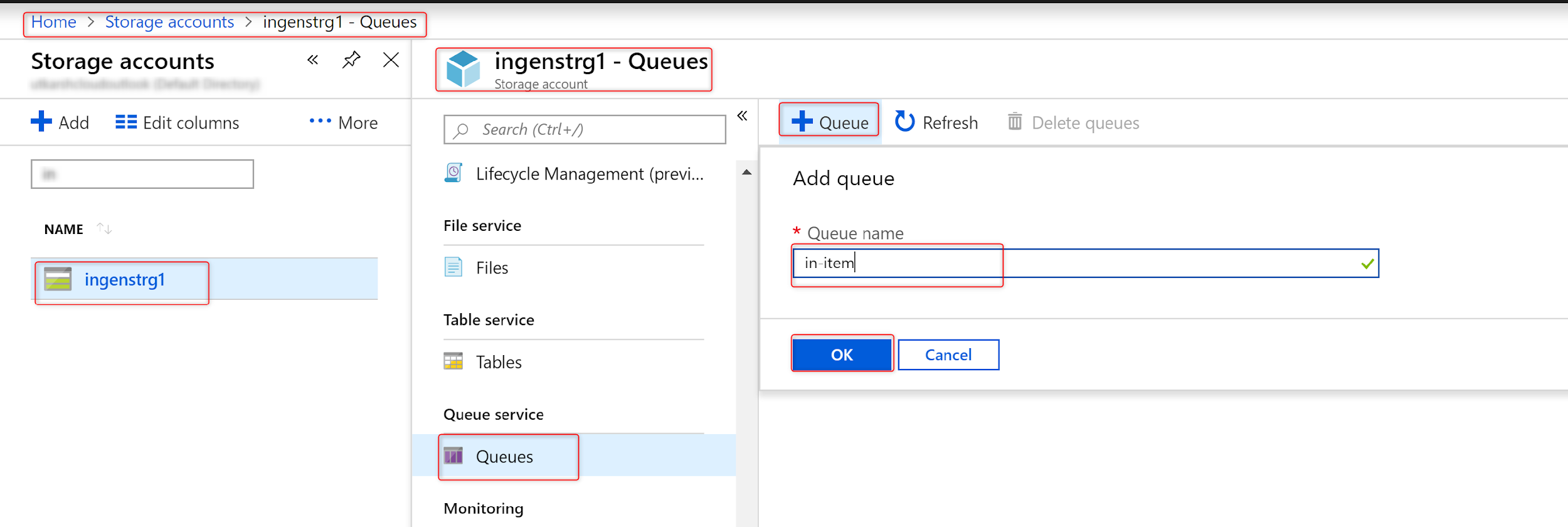
### **Task 3: Create the Storage Account**

1. In the navigation pane on the left side of the Azure Portal, click **All services**.
2. In the **All services** blade that displays, click **storage account**.
3. In the **storage account** blade that displays, view your list of storage accounts.
4. At the top of the **storage account** blade, click the **Add** button.
5. In the **storage account** blade, perform the following steps:
   * 1. **Resource group:** InGenRG
     2. **Storage account name**: ingenstrg1
     3. **Location**: East US
     4. **Performance:** standard
     5. **Account kind:** storage v2
     6. **Replication**: Locally-redundant storage
     7. **Access tier**: Hot
6. In the **storage account** blade, click **Create**.



### **Task 4: Create a queue for input messages**

1. Select the **ingenstrg1** storage account and click on queues
2. Add a new queue named: in-item



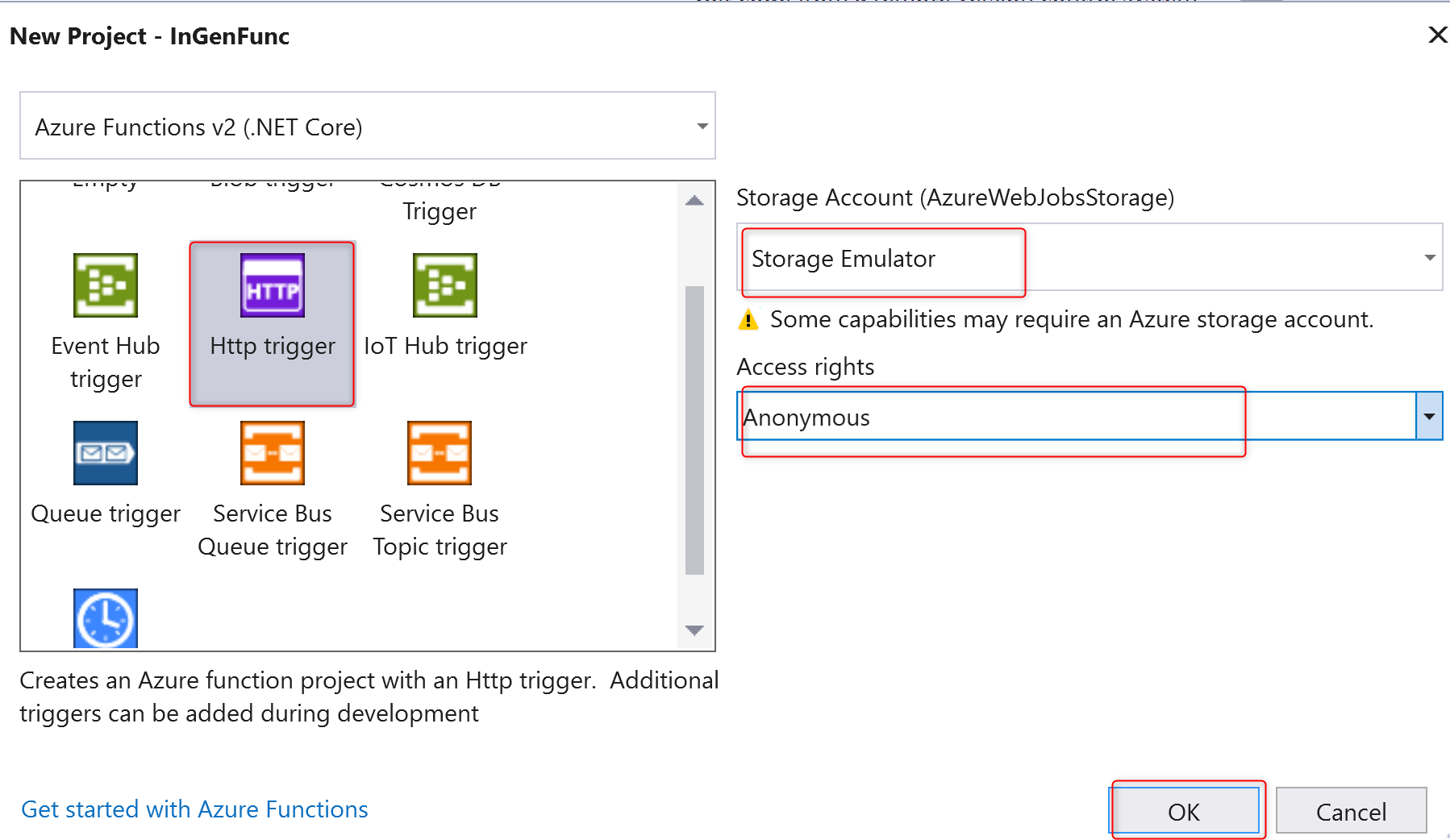
## Step 2: Create Azure Function using visual studio

### **Task 1: Create Azure Function App**

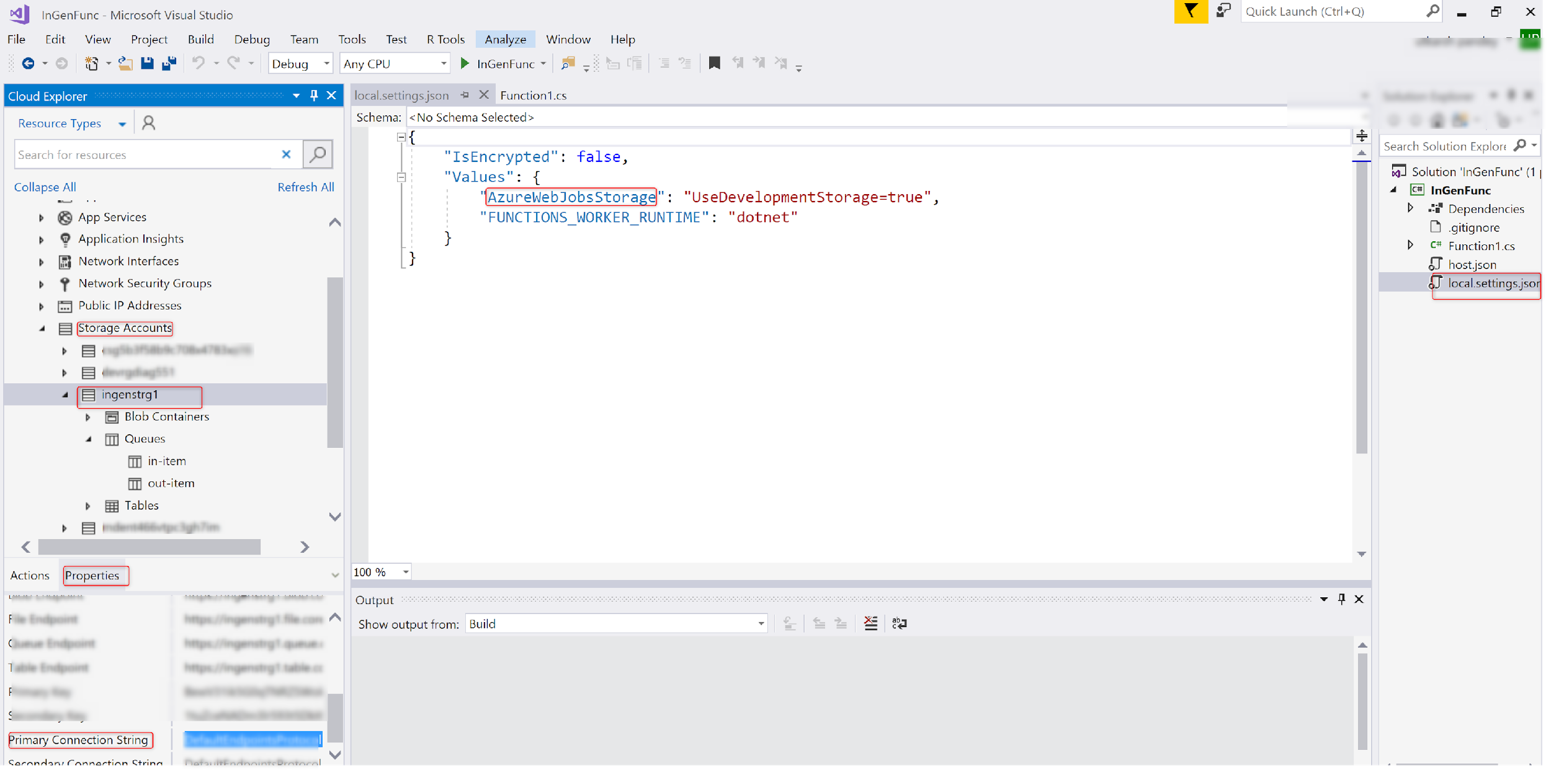
1. Open Visual Studio. The shortcut should be available on the desktop.
2. In the **New Project** dialog, select **Installed**, expand **Visual C#** > **Cloud**, select **Azure Functions**, type a **Name: ingenfunc** for your project, and click **OK**.



* + Select Http Trigger as template.

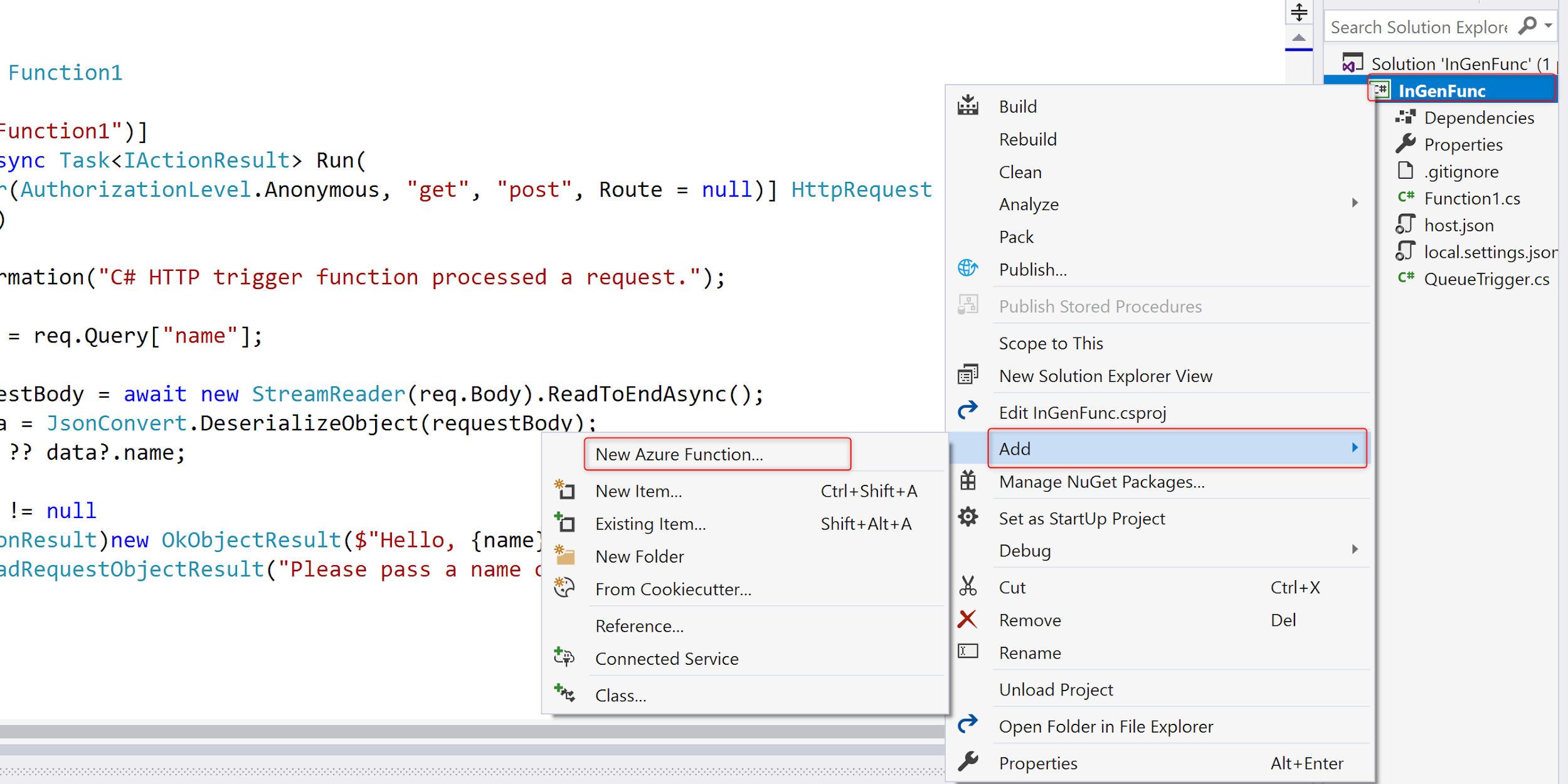


1. In Visual Studio, open **Cloud Explorer**, expand **Storage Account** > **Your Storage Account**, then select **Properties** and copy the **Primary Connection String** value.
2. In your project, open the **local.settings.json** file and set the value of the  **AzureWebJobsStorage** key to the connection string you copied.

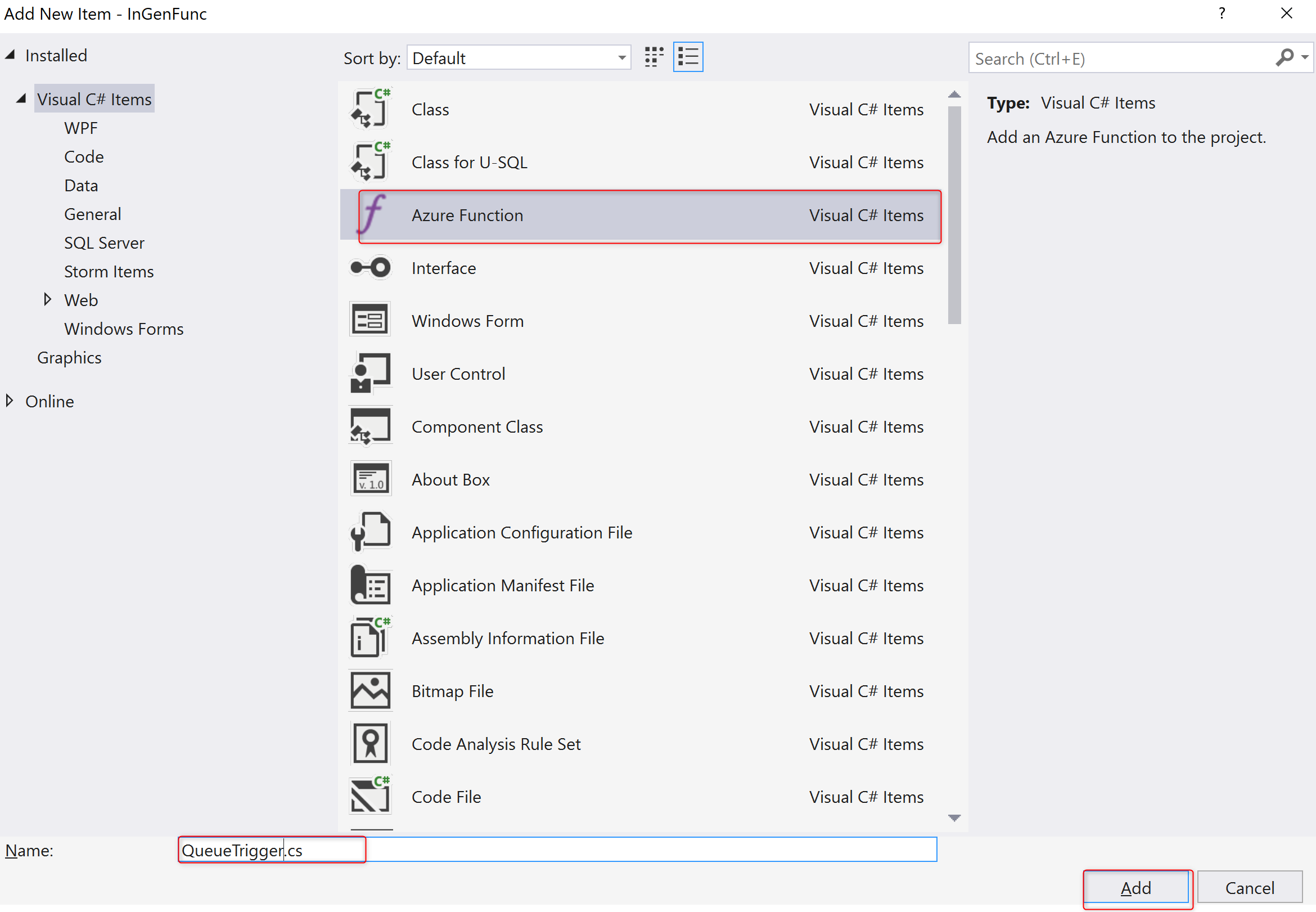


### **Task 2: Create Azure Function with queue trigger**

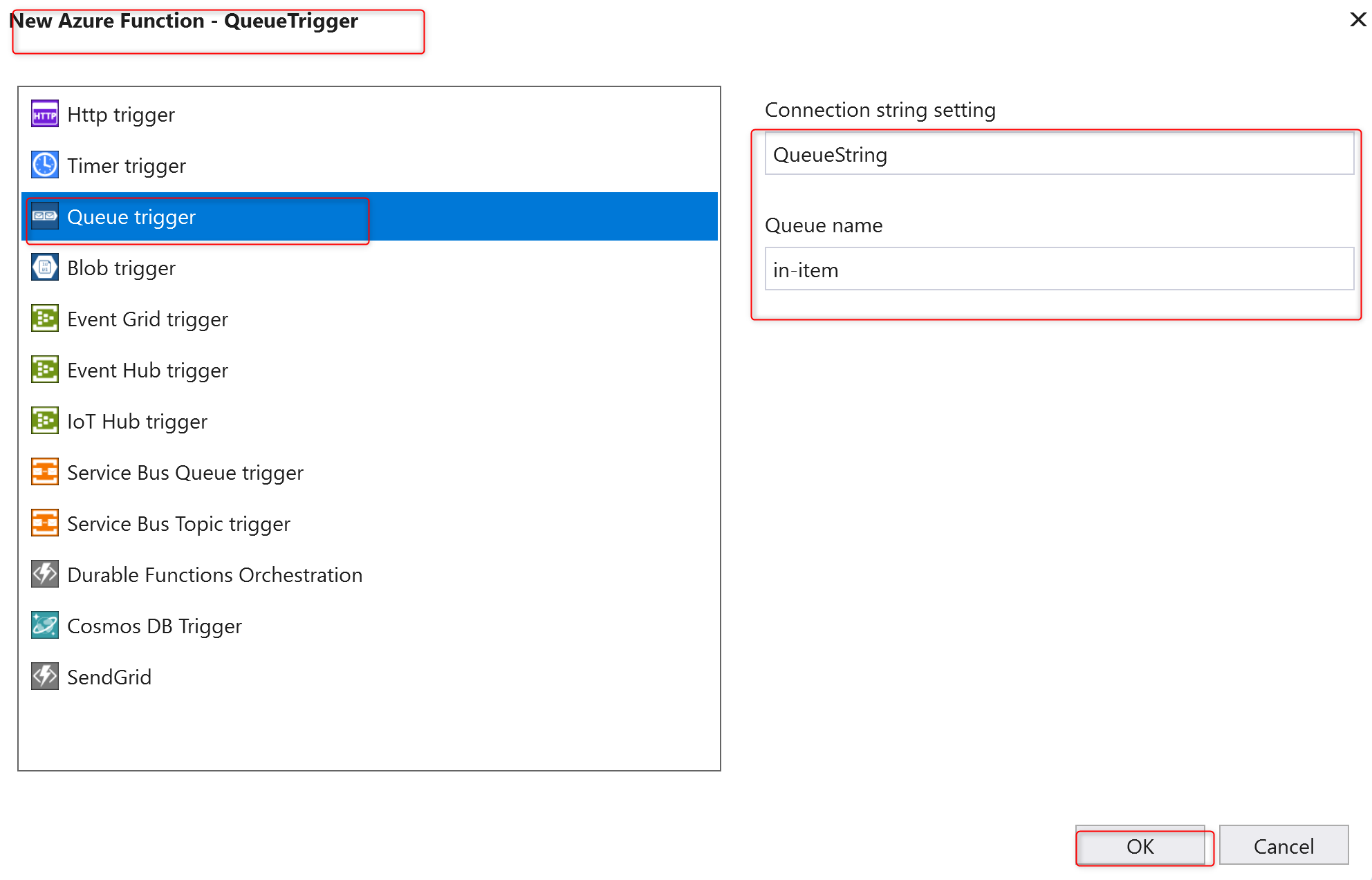
1. Right click on Project and add new Azure function.



1. Name the file as **QueueTrigger.cs.**

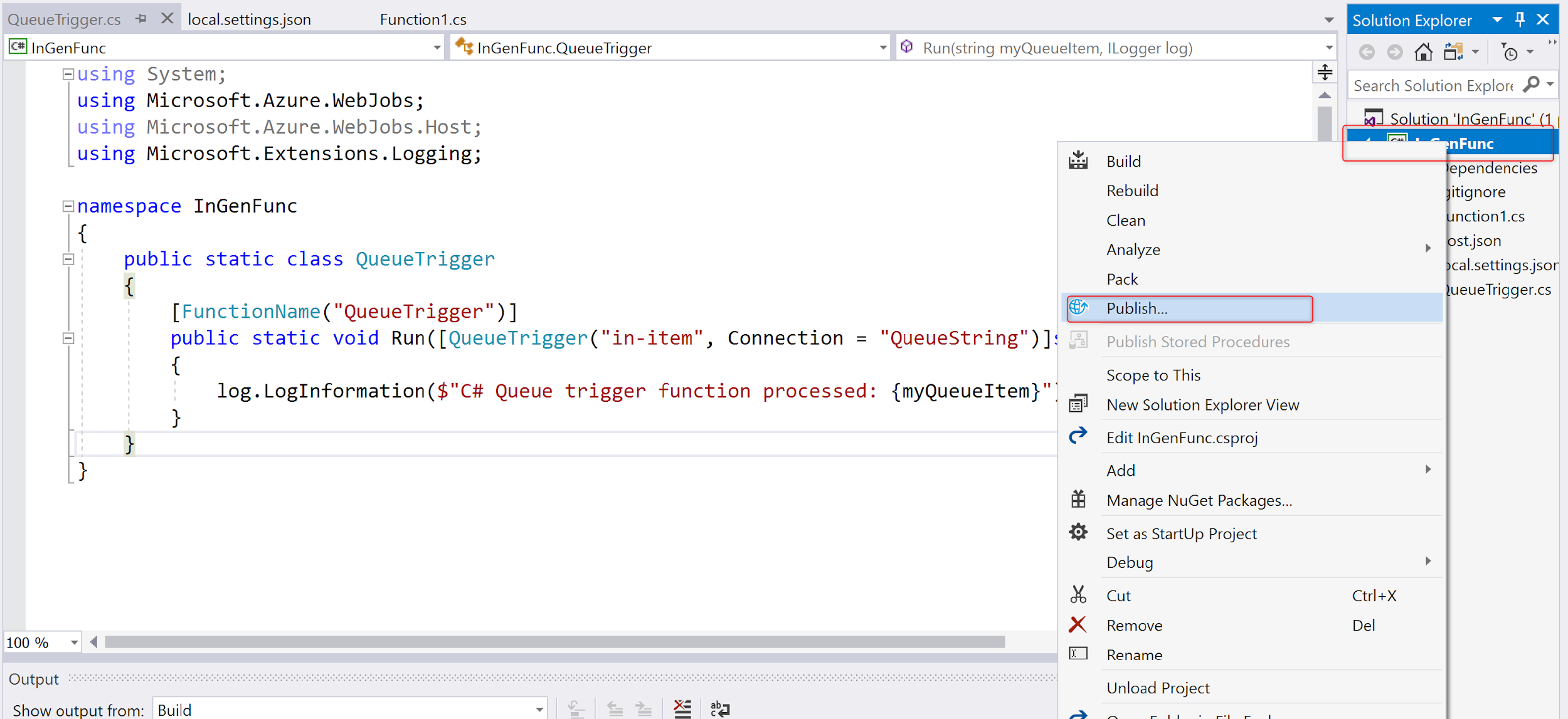


1. Select Queue Trigger from the template:
   1. **Connection String Setting**: QueueString
   2. **Queue Name**: in-item
2. Click OK.

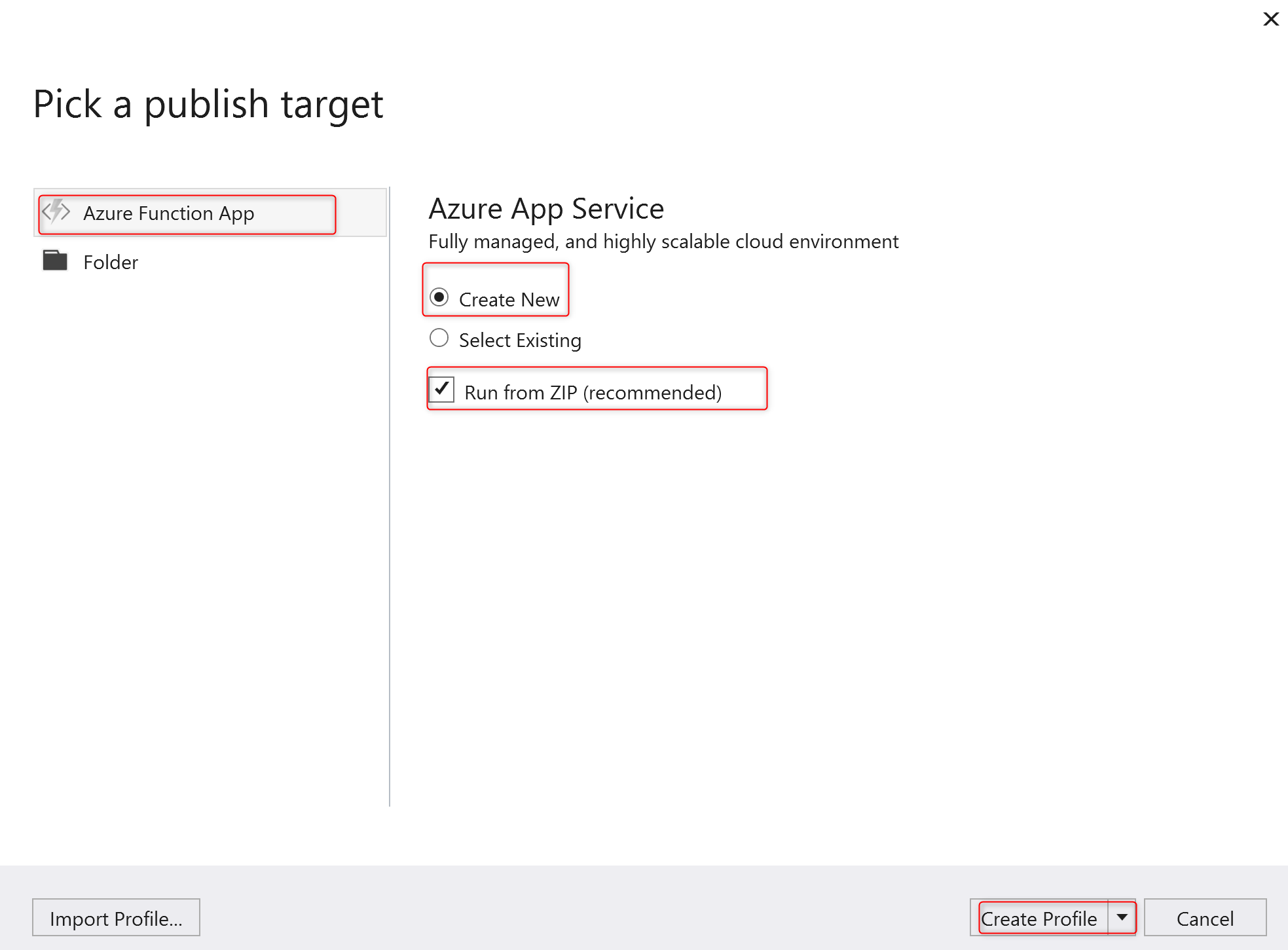


### **Task 3: Publish the function to Azure**

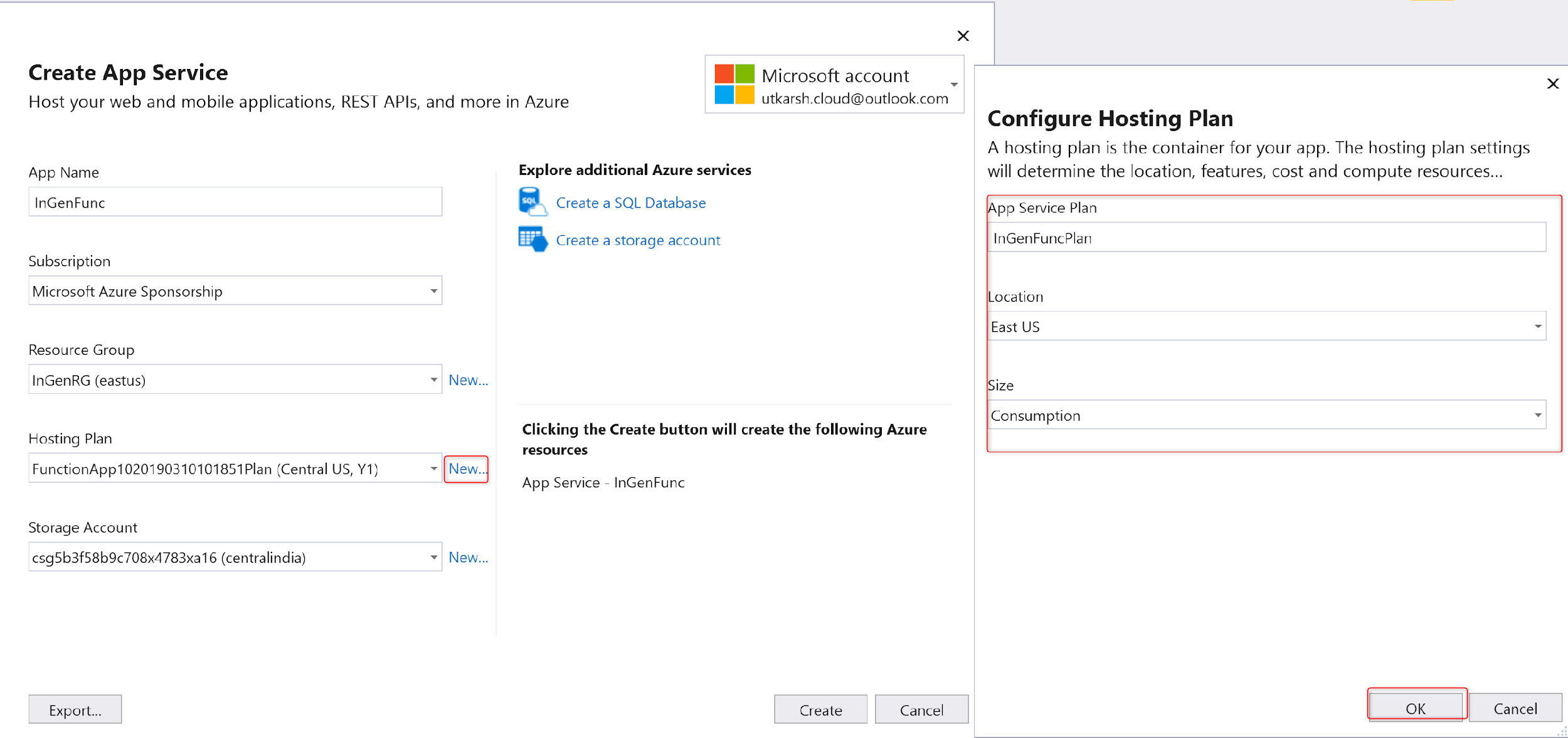
1. In VS **Solution Explorer**, right-click the project and select **Publish**.



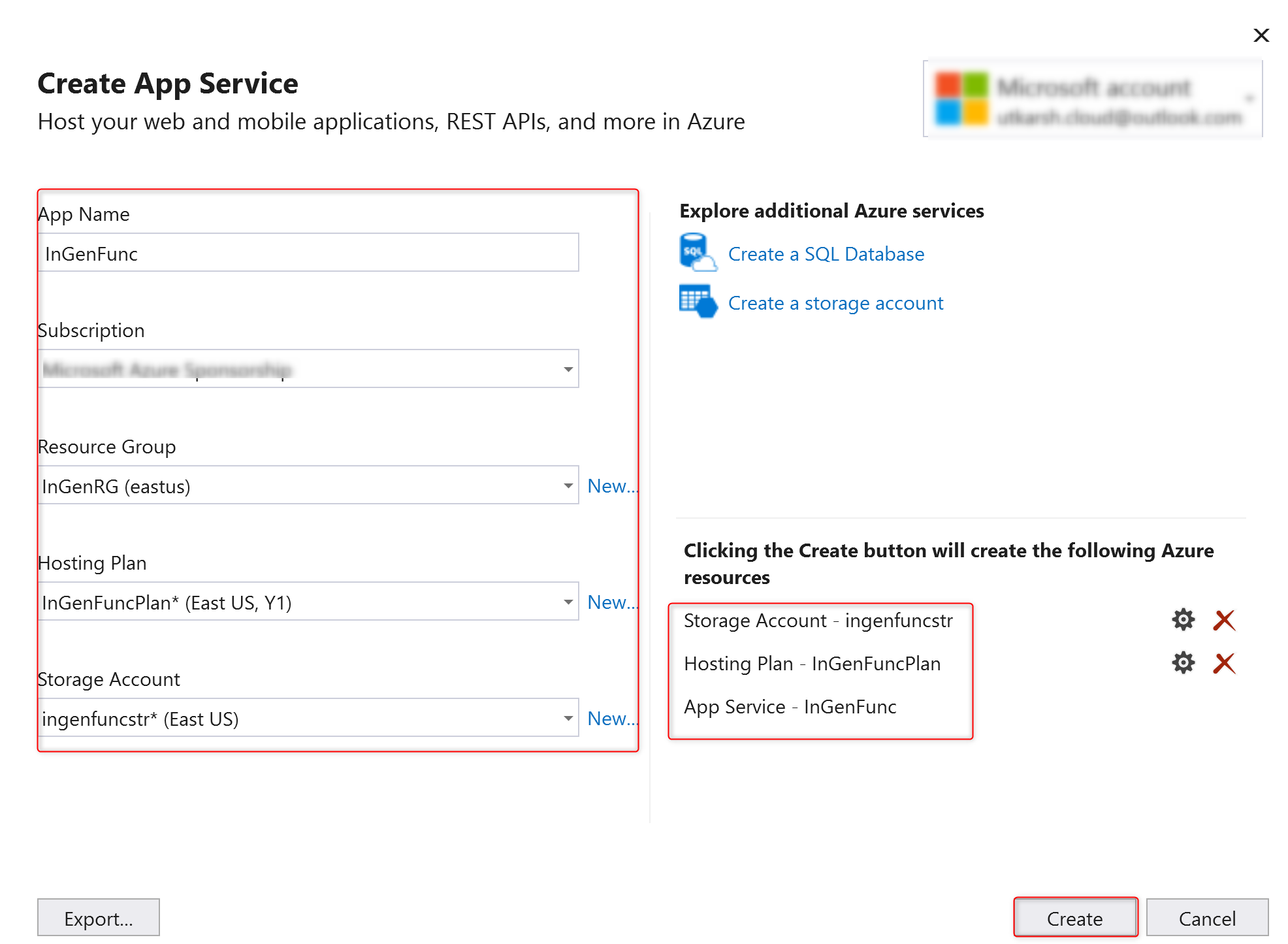
1. Select **Azure Function App**, choose **Create New**, and then select **Publish**. Ensure that selection to Run from Zip is marked.
2. Click on create Profile.



1. Create a new Hosting plan:
   1. **App Service plan**: InGenFuncPlan
   2. **Location**: East US
   3. **Size**: Consumption
   4. Click OK



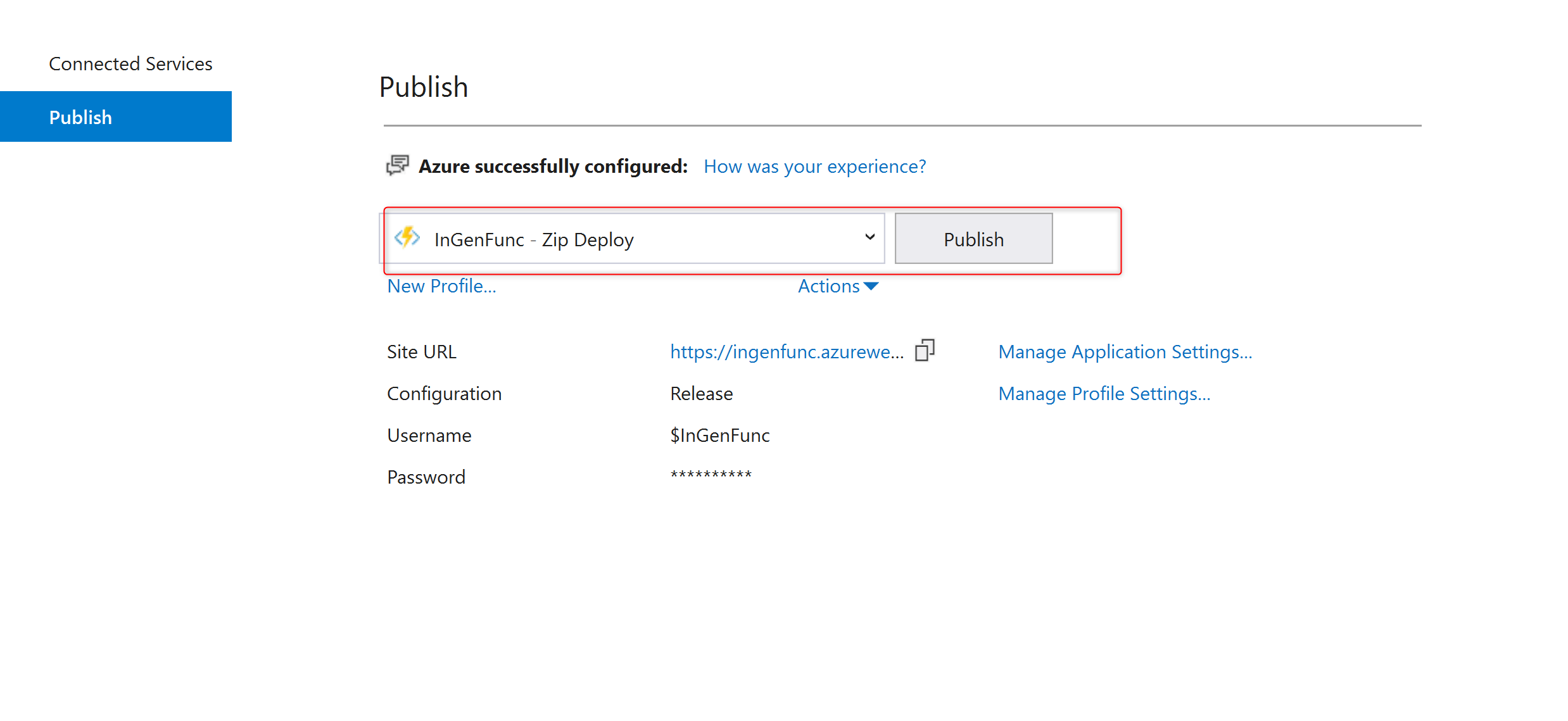
1. In the **Create App Service dialog**, use the **Hosting** settings as specified.



1. Click **Create** to create a function app and related resources in Azure with these settings and deploy the function project code.
2. After the deployment is complete, make a note of the **Site URL** value, which is the address of your function app in Azure.
3. Any settings we have added in the **local.settings.json** must also be added to the function app. These settings are not uploaded automatically when we publish the project. We can update this by clicking on manage Application Setting.
   1. Copy the value from Local to remote (under AzureWebJobStorage)



1. Click publish

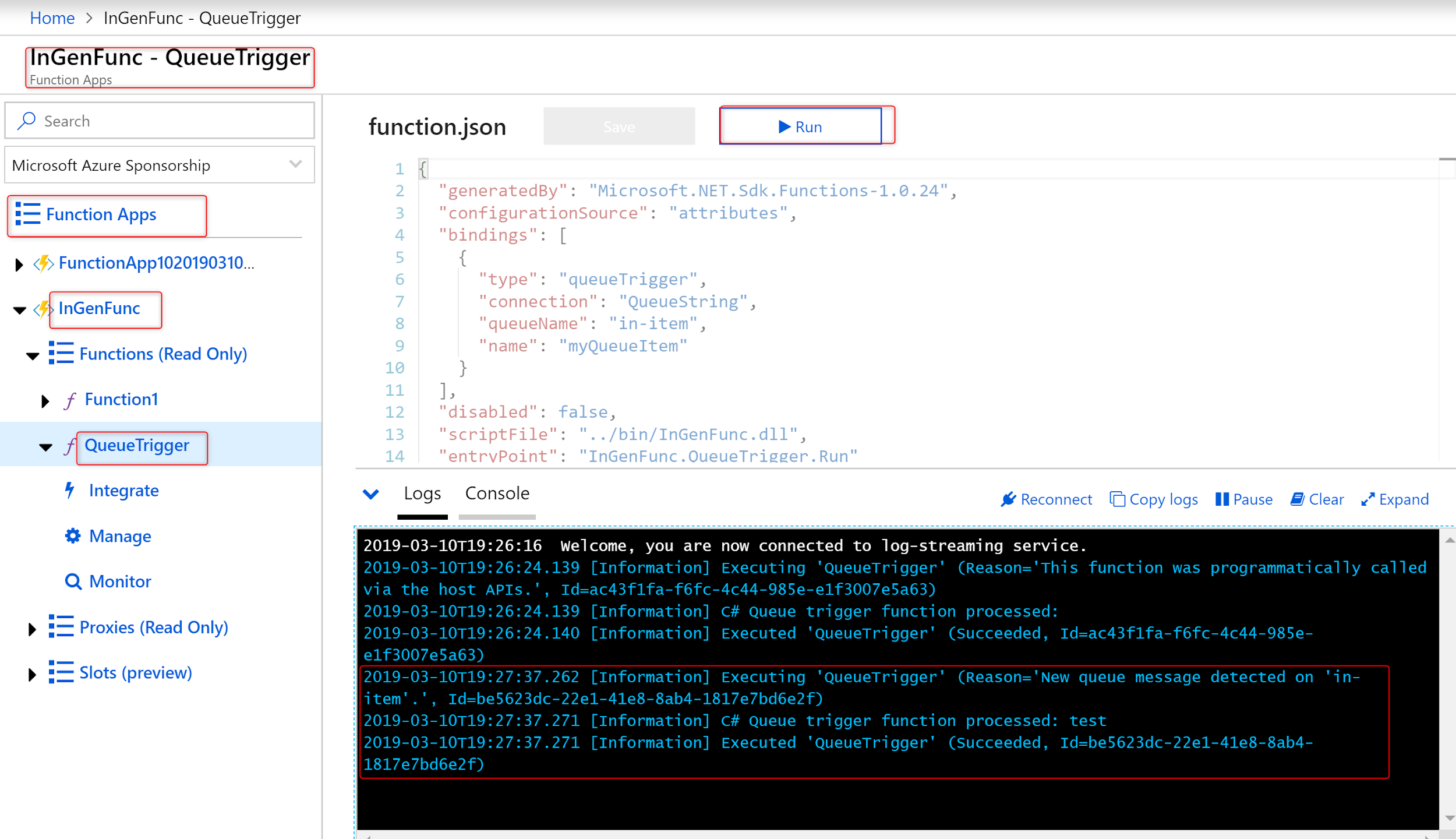


### **Task 4: Verify the functionality**

1. In Storage Explorer, expand your storage account: **ingeststrg1** ->**Queues** ->**in-item**, then click **Add message**.



* 1. Back in the Azure portal, browse to your function of **InGenFunc-QueueTrigger**, expand the **Logs** at the bottom of the page, and make sure that log streaming isn't paused.



* 1. Back in Storage Explorer, click **Refresh** and verify that the message has been processed and is no longer in the queue.

