

08 - Implement Azure Functions (5 min)

In this walkthrough, we will create a Function App to display a Hello message when there is an HTTP request.

Task 1: Create a Function app

In this task, we will create a Function app.

1. Sign in to the [Azure portal](#).
2. In the **Search** bar at the top of the portal, search for and select **Function App** and then, from the **Function App** blade, click **+ Add**, **+ Create**, **+ New**.
3. On the **Basic** tab of the **Function App** blade, specify the following settings (replace **xxxx** in the name of the function with letters and digits such that the name is globally unique and leave all other settings with their default values):

| Settings | Value |
|-------------------|---------------------------|
| Subscription | Keep default supplied |
| Resource group | Create new resource group |
| Function App name | function-xxxx |
| Publish | Code |
| Runtime stack | .NET |
| Version | 3.1 |
| Region | East US |

Note - Remember to change the **xxxx** so that it makes a unique **Function App name**

4. Click **Review + Create** and, after successful validation, click **Create** to begin provisioning and deploying your new Azure Function App.
5. Wait for the notification that the resource has been created.
6. When the deployment has completed, click **Go** to resource from the deployment blade. Alternatively, navigate back to the **Function App** blade, click **Refresh** and verify that the newly created function app has the **Running** status.

The screenshot shows the Azure portal interface for a Function App. At the top, there's a breadcrumb 'Home > Function App'. Below it, the title 'Function App' is displayed with 'Default Directory' underneath. A toolbar contains several icons: a plus sign for 'Add', a gear for 'Manage view', a circular arrow for 'Refresh', a download icon for 'Export to CSV', a tag icon for 'Assign tags', a play button for 'Start', a circular arrow with a slash for 'Restart', a square for 'Stop', and a trash can for 'Delete'. Below the toolbar is a filter bar with a text input 'Filter by name...', three filter buttons 'Subscription == all', 'Resource group == all', and 'Location == all', and an 'Add filter' button. Below the filter bar, it says 'Showing 1 to 1 of 1 records.' A table lists the function app with columns: Name, Status, Location, Pricing Tier, and App Service Plan. The table contains one row for 'function-9007' which is 'Running' in 'East US' on a 'Dynamic' pricing tier with the app service plan 'ASP-myRGFunction-a50c'.

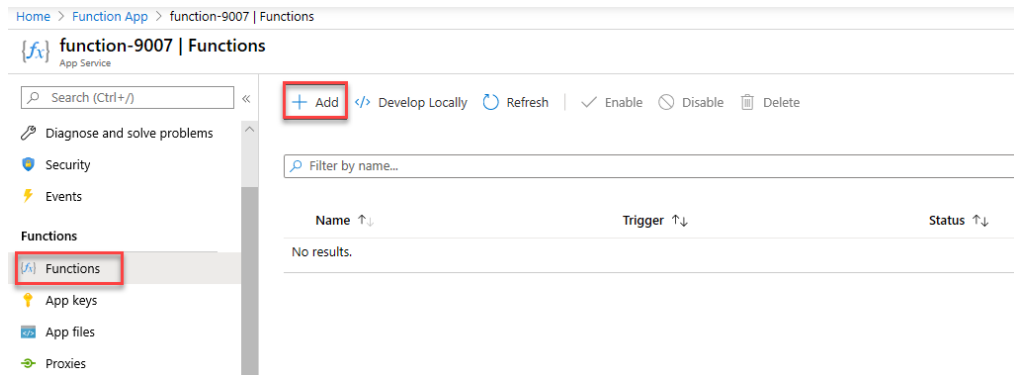
| Name | Status | Location | Pricing Tier | App Service Plan |
|---------------|---------|----------|--------------|-----------------------|
| function-9007 | Running | East US | Dynamic | ASP-myRGFunction-a50c |

Task 2: Create a HTTP triggered function and test

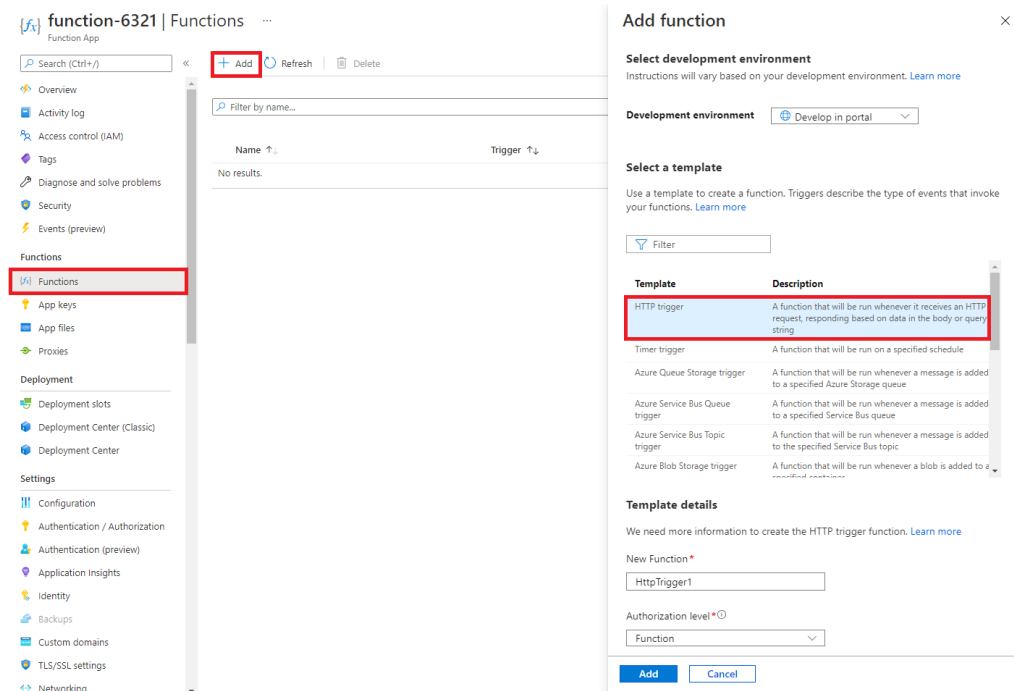
In this task, we will use the Webhook + API function to display a message when there is an HTTP request.

1. On the **Function App** blade, click the newly created function app.

2. On the function app blade, in the **Functions** section, click **Functions** and then click **+ Add, + Create, + New**.

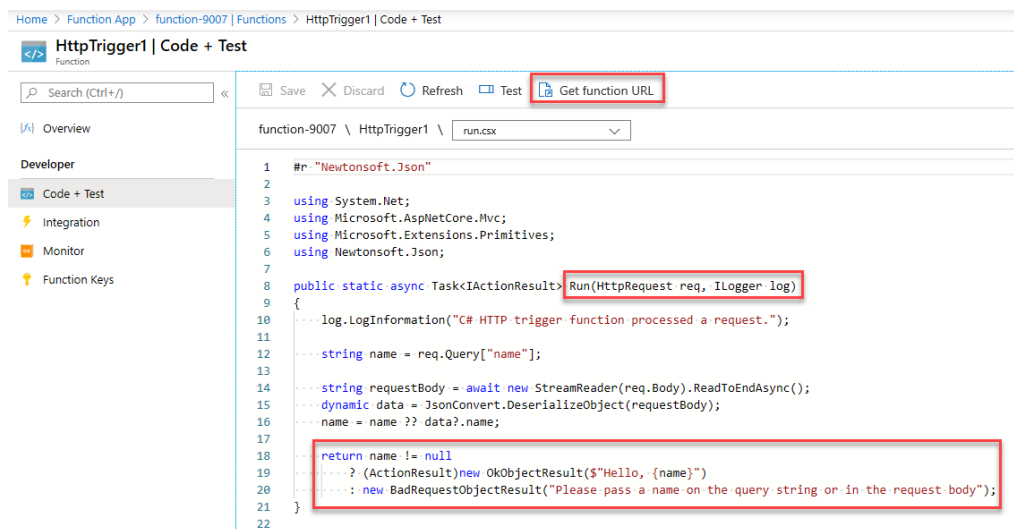


3. An **Add function** pop-up window will appear on the right. In the **Select a template** section click **HTTP trigger**. Click **Add**



4. On the **HttpTrigger1** blade, in the **Developer** section, click **Code + Test**.

5. On the **Code + Test** blade, review the auto-generated code and note that the code is designed to run an HTTP request and log information. Also, notice the function returns a Hello message with a name.



6. Click **Get function URL** from the top section of function editor.

7. Ensure that the value in the **Key** drop-down list is set to **default** and click **Copy** to copy the function URL.

Home > Function App > function-9007 | Functions > HttpTrigger1 | Code + Test

HttpTrigger1 | Code + Test

function-9007 \ HttpTrigger1 \ run.csx

```
1 #r "Newtonsoft.Json"
2
3 using System.Net;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.Extensions.Primitives;
6 using Newtonsoft.Json;
7
8 public static async Task

Get function URL



Key: default



URL: https://function-9007.azurewebsites.net/api/HttpTrigger1?code=...



or in the request body");


```

8. Open a new browser tab and paste the copied function URL into your web browser's address bar. When the page is requested the function will run. Notice the returned message stating that the function requires a name in the request body.

https://function-9007.azurewebsites.net/api/HttpTrigger1?code=...

Please pass a name on the query string or in the request body

9. Append **&name=yourname** to the end of the URL.

Note: For example, if your name is Cindy, the final URL will resemble the following:

https://azfuncxxx.azurewebsites.net/api/HttpTrigger1?code=X9xx9999xxxxX9xxXX=&name=cindy

https://function-9007.azurewebsites.net/api/HttpTrigger1?code=...&name=Cindy

Hello, Cindy

10. When you hit enter, your function runs and every invocation is traced. To view the traces, return to the Portal **HttpTrigger1 | Code + Test** blade and click **Monitor**. You can **configure** Application Insights by selecting the timestamp and click **Run query in Application Insights**.

Home > Function App > function-9007 | Functions > HttpTrigger1 | Monitor

HttpTrigger1 | Monitor

Invocations Logs

Success Count: 2 (Last 30 Days)

Error Count: 0 (Last 30 Days)

Invocation Traces

The twenty most recent function invocation traces. For more advanced analysis, run the query in Application Insights.

Run query in Application Insights Refresh

Filter invocations

| Date (UTC) | Success | Result Code | Duration (ms) | Operation Id |
|-------------------------|---------|-------------|---------------|----------------------------------|
| 2020-05-15 01:38:54.716 | Success | 200 | 33 | 9a05b4de7403a448662232ab9d808e |
| 2020-05-15 01:36:18.615 | Success | 400 | 167 | 205f016ac879f54bbf5f5e0700915225 |

Congratulations! You have created a Function App to display a Hello message when there is an HTTP request.

Note: To avoid additional costs, you can optionally remove this resource group. Search for resource groups, click your resource group, and then click **Delete resource group**. Verify the name of the resource group and then click **Delete**. Monitor the **Notifications** to see how the delete is proceeding.

