

## 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

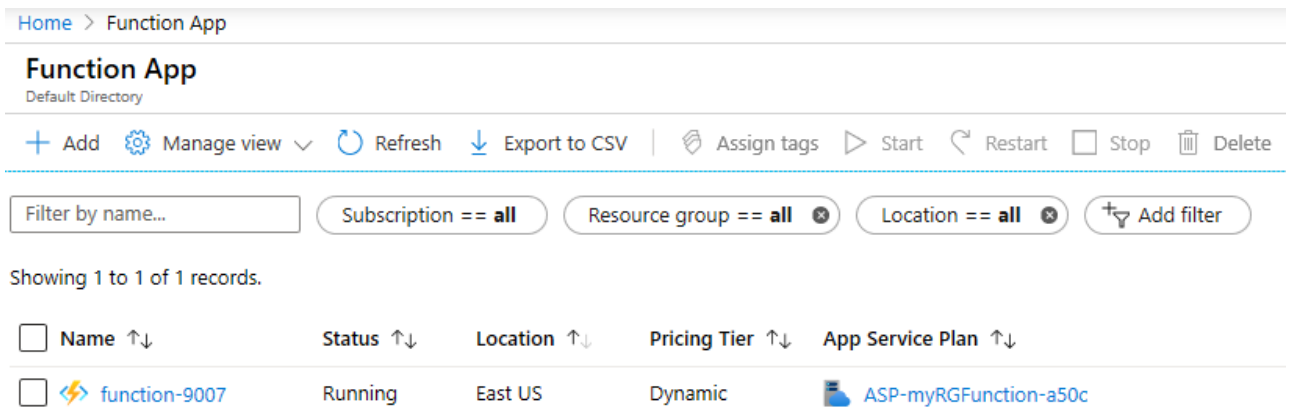
In this task, we will create a Function app.

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, clickGo to resourcefrom 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.



Home > Function App



### Function App

Default Directory

+ Add Manage view Refresh Export to CSV Assign tags Start Restart Stop Delete

Filter by name... Subscription == all Resource group == all Location == all Add filter

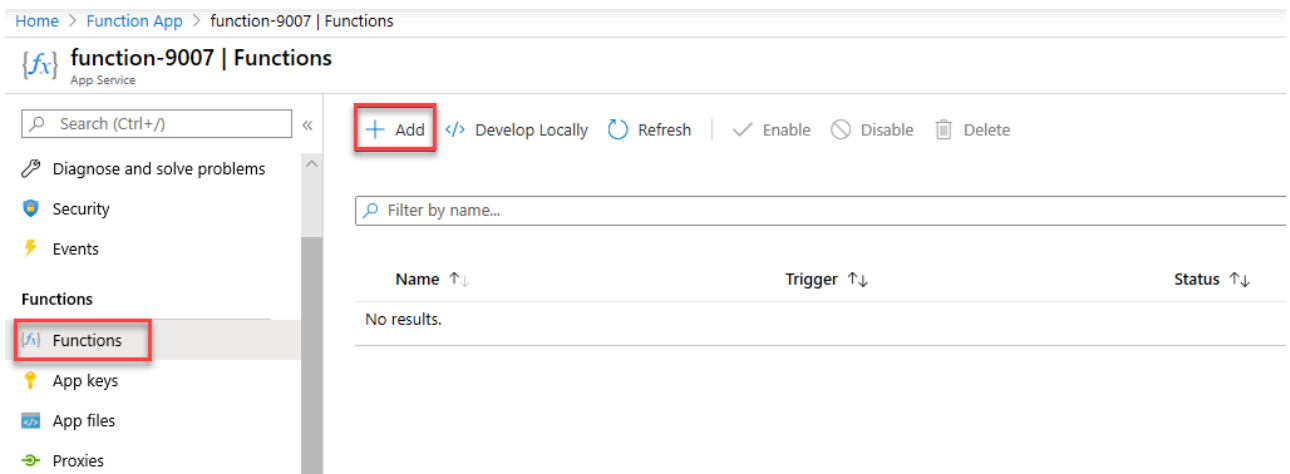
Showing 1 to 1 of 1 records.

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**.



Home > Function App > function-9007 | Functions

### function-9007 | Functions

App Service

Search (Ctrl+/) << + Add </> Develop Locally Refresh Enable Disable Delete

Diagnose and solve problems

Security

Events

Functions

Functions

App keys

App files

Proxies

Filter by name...

Name ↑↓	Trigger ↑↓	Status ↑↓
No results.		

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

## 08 - Implement Azure Functions

The screenshot shows the Azure Functions portal for a function app named 'function-6321'. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Security, Events (preview), Functions, App keys, App files, Proxies, Deployment, Deployment slots, Deployment Center (Classic), Deployment Center, Settings, Configuration, Authentication / Authorization, Authentication (preview), Application Insights, Identity, Backups, Custom domains, TLS/SSL settings, and Networking. The 'Functions' link is highlighted. The main area shows a table with columns 'Name' and 'Trigger', currently displaying 'No results.' The '+ Add' button is highlighted. The 'Add function' dialog is open on the right, showing the 'Select development environment' section with 'Develop in portal' selected. The 'Select a template' section shows a table of templates, with 'HTTP trigger' highlighted. The 'Template details' section shows the 'New Function' field set to 'HttpTrigger1' and the 'Authorization level' set to 'Function'.

**function-6321 | Functions**

Search (Ctrl+/) << + Add Refresh Delete

Filter by name...

Name ↑↓ Trigger ↑↓

No results.

**Add function**

Select development environment

Instructions will vary based on your development environment. [Learn more](#)

Development environment

Select a template

Use a template to create a function. Triggers describe the type of events that invoke your functions. [Learn more](#)

Filter

Template	Description
HTTP trigger	A function that will be run whenever it receives an HTTP request, responding based on data in the body or query string
Timer trigger	A function that will be run on a specified schedule
Azure Queue Storage trigger	A function that will be run whenever a message is added to a specified Azure Storage queue
Azure Service Bus Queue trigger	A function that will be run whenever a message is added to a specified Service Bus queue
Azure Service Bus Topic trigger	A function that will be run whenever a message is added to the specified Service Bus topic
Azure Blob Storage trigger	A function that will be run whenever a blob is added to a specified container

Template details

We need more information to create the HTTP trigger function. [Learn more](#)

New Function \*

HttpTrigger1

Authorization level \*

Function

Add Cancel

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

The screenshot shows the 'HttpTrigger1 | Code + Test' blade in the Azure Functions portal. The left sidebar contains navigation links for Overview, Code + Test, Integration, Monitor, and Function Keys. The 'Code + Test' link is highlighted. The main area shows the code editor for 'function-9007 \ HttpTrigger1 \ run.csx'. The code is as follows:

```
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 Run(HttpRequest req, ILogger log)
9 {
10     log.LogInformation("C# HTTP trigger function processed a request.");
11
12     string name = req.Query["name"];
13
14     string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
15     dynamic data = JsonConvert.DeserializeObject(requestBody);
16     name = name ?? data?.name;
17
18     return name != null
19         ? (ActionResult)new OkObjectResult($"Hello, {name}")
20         : new BadRequestObjectResult("Please pass a name on the query string or in the request body");
21 }
22
```

The 'Run' method signature and the return statement are highlighted with red boxes. The 'Get function URL' button is also highlighted.

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

**HttpTrigger1 | Code + Test**

Function

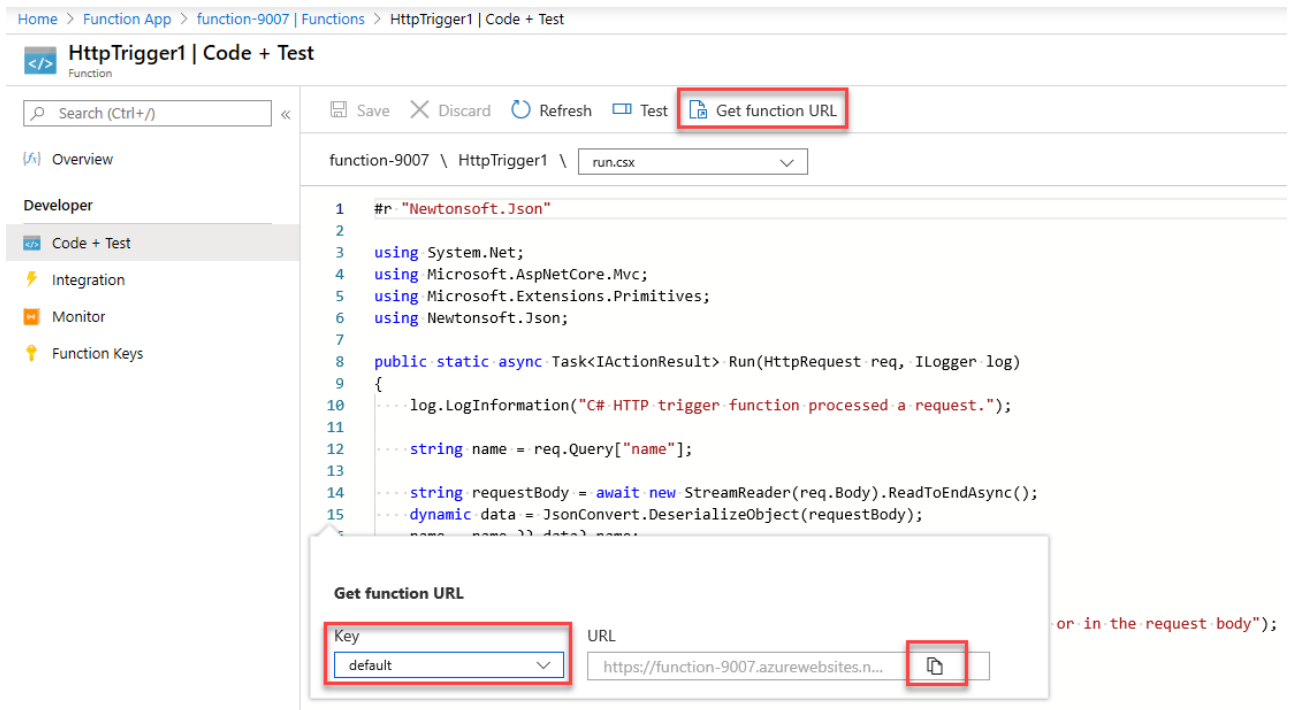
Search (Ctrl+/) << Save Discard Refresh Test Get function URL

function-9007 \ HttpTrigger1 \ run.csx

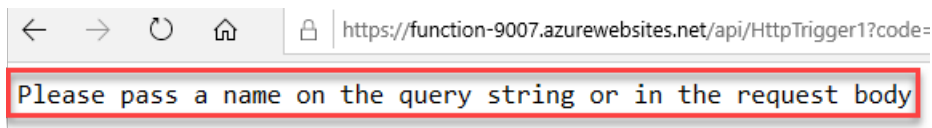
```
1 #r "Newtonsoft.Json"
2
3 using System.Net;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.Extensions.Primitives;
6 using Newtonsoft.Json;
7
8 public static async Task Run(HttpRequest req, ILogger log)
9 {
10     log.LogInformation("C# HTTP trigger function processed a request.");
11
12     string name = req.Query["name"];
13
14     string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
15     dynamic data = JsonConvert.DeserializeObject(requestBody);
16     name = name ?? data?.name;
17
18     return name != null
19         ? (ActionResult)new OkObjectResult($"Hello, {name}")
20         : new BadRequestObjectResult("Please pass a name on the query string or in the request body");
21 }
22
```

- Click **Get function URL** from the top section of function editor.
- Ensure that the value in the **Key** drop-down list is set to **default** and click **Copy** to copy the function URL.

## 08 - Implement Azure Functions



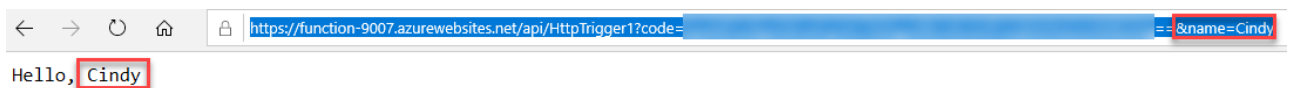
- Open a new browser tab and paste the copied function URL into your web browser's address bar. When the page is requested the function will run. Notice the returned message stating that the function requires a name in the request body.



- 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=X9xx9999xXXXXX9x9xxxXX==&name=cindy
```



- 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**.

## 08 - Implement Azure Functions

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

### HttpTrigger1 | Monitor

Function

Search (Ctrl+/)

Overview

Developer

Code + Test

Integration

**Monitor**

Function Keys

**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
<a href="#">2020-05-15 01:38:54.716</a>	✓ Success	200	33	9a05b4de7403af448662232ab9df808e
<a href="#">2020-05-15 01:36:18.615</a>	✓ 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.