

Cloud Function – Running Locally

Cloud Functions in GCP (Google Cloud Platform) are a way to run small pieces of code (called functions) in the cloud without needing to manage servers. This is known as **serverless computing**, which means Google takes care of all the server infrastructure for you. You just write your function, deploy it, and Google makes sure it runs when it's supposed to.

What Are Cloud Functions?

Cloud Functions are event-driven. That means they are designed to respond to specific events — for example, a file being uploaded to Google Cloud Storage, a message arriving in Pub/Sub, or an HTTP request being made to an endpoint.

You write your function in a supported programming language (like Python, Node.js, Go, etc.), and then deploy it to GCP. It runs only when triggered, and you are billed only for the time your function runs — not for idle time.

Key Benefits

1. **No server management**

You don't need to provision or manage any servers — Google handles that for you.

2. **Automatic scaling**

Your function can automatically scale up to handle many requests at once, and scale down to zero when not in use.

3. **Pay-as-you-go**

You only pay for the compute time your function uses, down to the nearest 100 milliseconds.

4. **Quick development**

Great for prototyping or building small, focused services — you can go from writing code to deploying in minutes.

5. **Tight integration with other GCP services**

Cloud Functions easily integrate with Cloud Storage, Pub/Sub, Firestore, BigQuery, and other GCP services.

Common Use Cases

1. **Data Processing**

For example, when someone uploads a file to Cloud Storage, a Cloud Function can automatically process or transform the file (like resizing an image or converting a format).

2. API Backend

You can expose a function via HTTP and use it as the backend for a web app or mobile app. It can handle user requests, fetch data, and send responses.

3. Event-driven automation

Automatically respond to events like changes in a database, new messages in a queue, or alerts from monitoring tools.

4. Real-time File Transformation

Modify or analyze files in real time. For example, if a user uploads a resume, a function could extract text and analyze it for keywords.

5. IoT and Real-time Streaming

Process sensor data from IoT devices as it comes in through Pub/Sub topics.

6. Security and Operations Automation

Automatically react to security events — for example, revoke access if suspicious behavior is detected, or log and alert based on cloud activity logs.

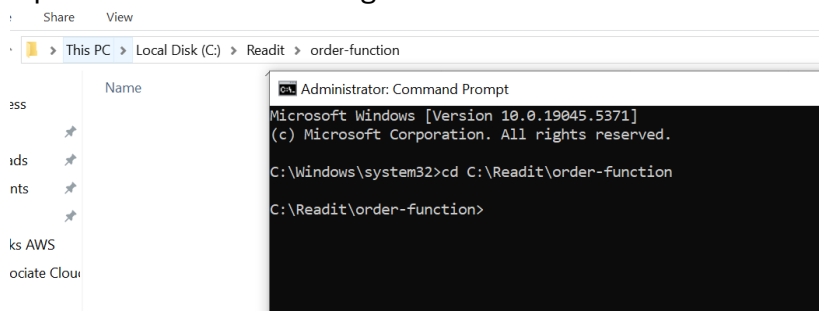
In short, **Cloud Functions** let you run small code tasks in response to cloud events without worrying about infrastructure. They are lightweight, scalable, and ideal for building microservices or automating cloud workflows.

In this lab first we will test our code locally which we later on going to deploy at GCP Functions.

In a real Environment as well, we follow the same process, first we test our code locally and only then we push it to the cloud function. This function will be based on .NET code.

To begin with the Lab

- 1) Go to the ReadIt app folder or any folder of your choice in your local machine and create a new folder named order-function.
- 2) Open CMD as admin and go to this folder in CMD.



- 3) First, we will install the tool that will be required to create a Google Cloud Function project in VS Code.

4) So, we will run our first command to install the tool.

dotnet new -i Google.Cloud.Functions.Templates

Then run this command shown below.

dotnet new gcf-http

(if the above command does not work or gives you an error, then you can use this command)

dotnet new gcf-http --force

```
C:\Windows\system32>cd C:\Readit\order-function
C:\Readit\order-function>dotnet new -i Google.Cloud.Functions.Templates
Warning: use of 'dotnet new --install' is deprecated. Use 'dotnet new install' instead.
For more information, run:
  dotnet new install -h

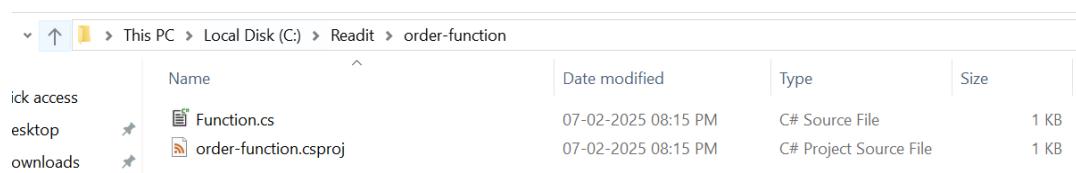
The following template packages will be installed:
  Google.Cloud.Functions.Templates

Success: Google.Cloud.Functions.Templates::2.2.1 installed the following templates:
Template Name                                           Short Name      Language      Tags
-----
Google Cloud Functions CloudEvent Function            gcf-event       [C#],F#,VB    Google/Cloud
Google Cloud Functions CloudEvent Function (Untyped)  gcf-untyped-event [C#],F#,VB    Google/Cloud
Google Cloud Functions HttpFunction                   gcf-http        [C#],F#,VB    Google/Cloud

C:\Readit\order-function>dotnet new gcf-http
The template "Google Cloud Functions HttpFunction" was created successfully.

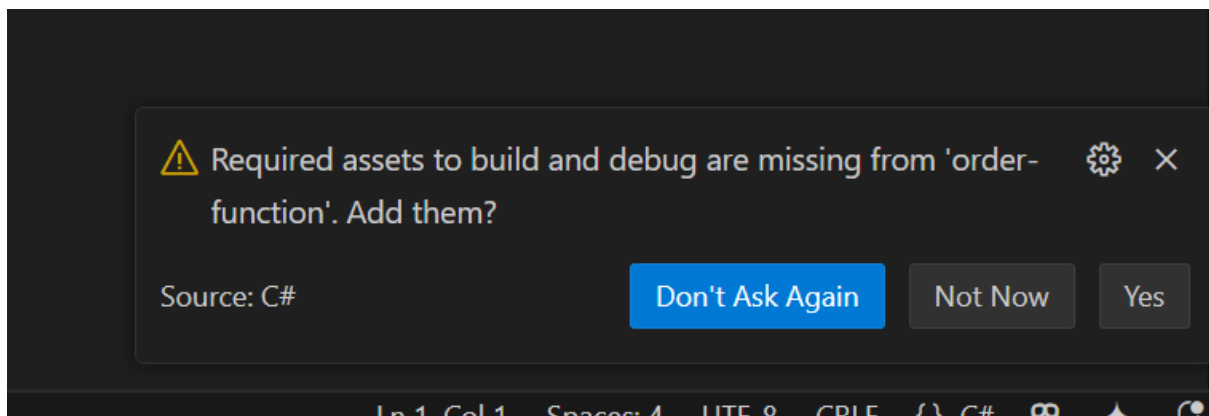
C:\Readit\order-function>
```

5) Now, you will notice two files have been created in the order-function folder we created.



This PC > Local Disk (C:) > Readit > order-function					
	Name	Date modified	Type	Size	
ick access	Function.cs	07-02-2025 08:15 PM	C# Source File	1 KB	
esktop	order-function.csproj	07-02-2025 08:15 PM	C# Project Source File	1 KB	
ownloads					

6) We will open the **order-function** folder in VS Code and make sure to click Yes if you see a message to add missing assets.



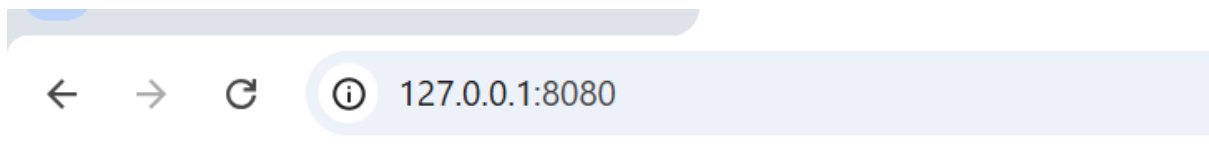
- 7) As of now, it is just a simple Hello function framework, but we will still test this if it runs locally.
- 8) So hit F5. As we have seen in previous labs, generally when we build an app locally, it opens a web URL, but not in this case.
- 9) So, we will open it manually. Copy the URL as shown below, which would look like something <http://127.0.0.1:8080>

```
9      /// <summary>
10     /// Logic for your function goes here.
11     /// </summary>
12     /// <param name="context">The HTTP context, containing the request and the response.</param>
13     /// <returns>A task representing the asynchronous operation.</returns>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Filter (e.g. text, \exclude, \escape)

```
order-function.dll (9056): Loaded 'C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App\6.0.36\Microsoft.Net.Http.Headers.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.
order-function.dll (9056): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.36\System.Net.Security.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.
order-function.dll (9056): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.36\System.Security.Cryptography.X509Certificates.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.
order-function.dll (9056): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.36\System.Security.Cryptography.Primitives.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.
2025-02-07T14:53:13.413Z [Microsoft.Hosting.Lifetime] [info] Now listening on: http://127.0.0.1:8080
2025-02-07T14:53:13.413Z [Microsoft.Hosting.Lifetime] [info] Application started. Press Ctrl+C to shut down.
2025-02-07T14:53:13.413Z [Microsoft.Hosting.Lifetime] [info] Hosting environment: Production
2025-02-07T14:53:13.414Z [Microsoft.Hosting.Lifetime] [info] Content root path: C:\Readit\order-function
```

- 10) Open the URL in the browser.



Hello, Functions Framework.

11) As we are able to access the webpage, it shows that the function is working

12) So now we will use our own code. For that, you need to download the order-baseline zip folder from GitHub and then unzip this folder.

13) You will find two files in this folder but you need to copy only the Function.cs file and paste it in the order-function folder, where you just created a function.

14) Now, if we go through this new file, we can see some quite simple things.

15) Here first log says “order function activated” will be created when the function is triggered, then we are going to read the body of the function.

```
10 public class Function : IHttpFunction
11 {
12     /// <summary>
13     /// Logic for your function goes here.
14     /// </summary>
15     /// <param name="context">The HTTP context, containing the request and the response.</param>
16     /// <returns>A task representing the asynchronous operation.</returns>
17     public async Task HandleAsync(HttpContext context)
18     {
19         Console.WriteLine("Order function activated...");
20         var body = new StreamReader(context.Request.Body).ReadToEndAsync().Result;
21         Console.WriteLine($"Order received: {body}");
22
23         await context.Response.WriteAsync("Order received successfully.");
24         Console.WriteLine($"Order function completed");
25     }
26 }
27
```

16) Now, again hit F5 to build this new code, and it will again generate the same URL as before. But this time, we cannot test it in the browser, as the browser cannot set the body of a request. So, we will use a tool named Postman

17) So first we need to download Postman, for this we can use link below

<https://www.postman.com/downloads/>

Right-click on the downloaded file and run as administrator
If asked, click continue without account

Working with APIs simplified with Postman

Your work email makes it easy for you to collaborate with your teammates.

Create Free Account

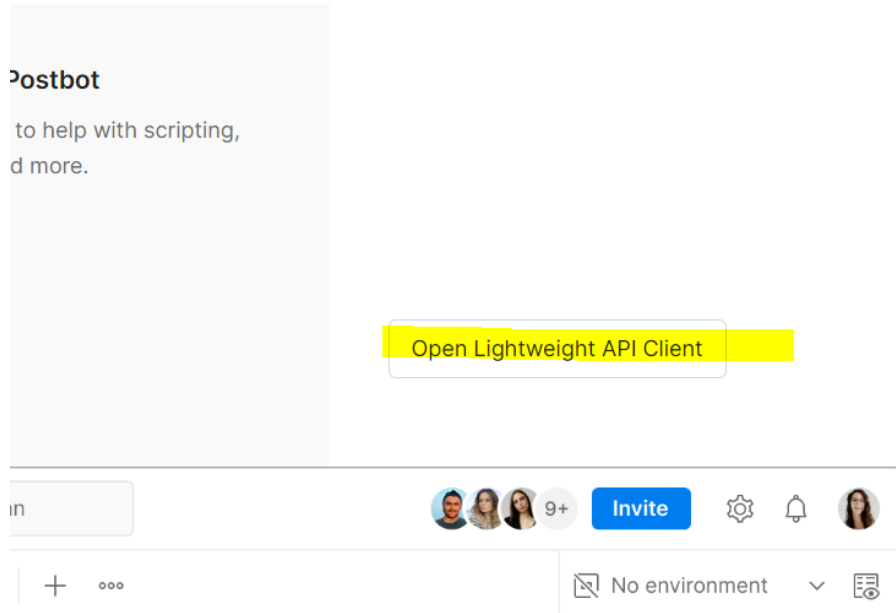
Already have an account? [Log In](#)

Continue with Google

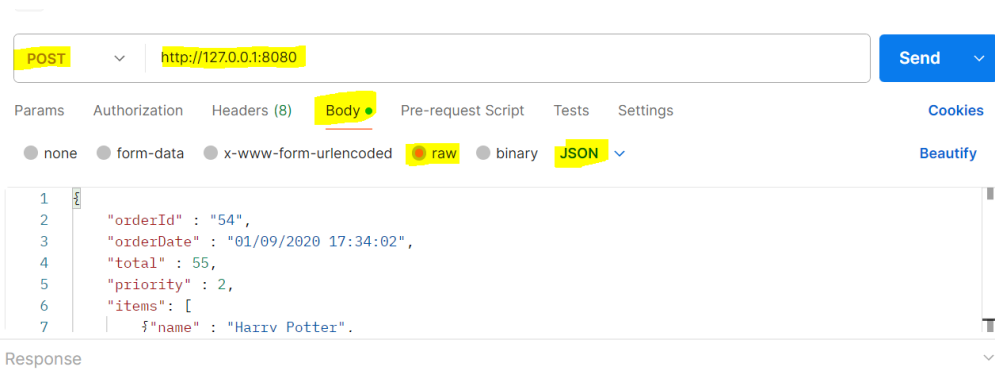
Single Sign On (SSO)

Continue without an account

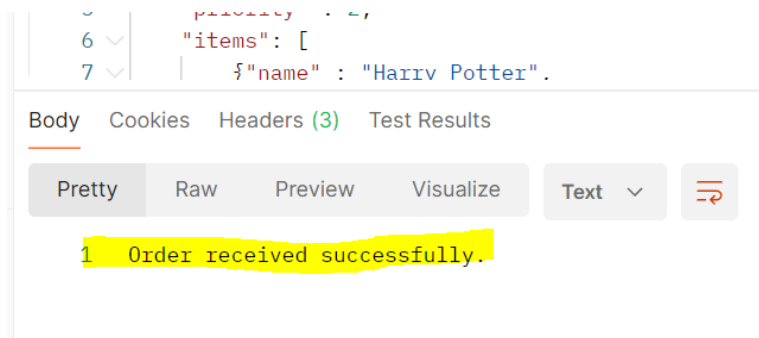
18) Then click open the lightweight API client



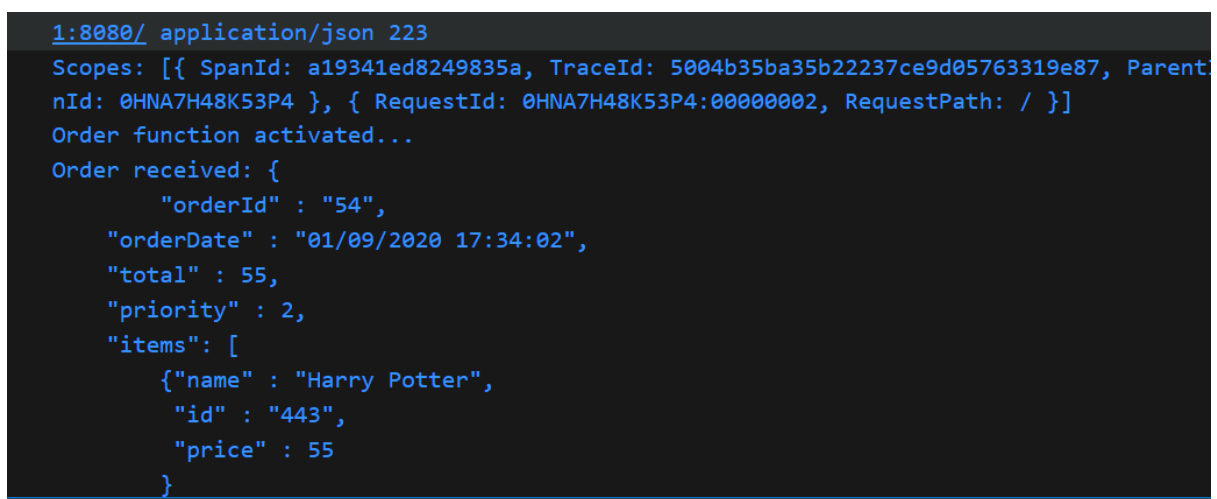
19) From the drop-down menu, select Post, paste the URL, go to the body and select raw. From the drop-down menu, select JSON
Now, open the second file (ordersample.json) from the zip folder and paste its data in Postman and click send



20) We should get a response that the order was received successfully



21) Now go back to VS Code and scroll down, you should be able to see the order details that we put in the body



22) Now the above activity shows that our function or code is running, and is also capable of receiving the requests.