How to Function

This is a short guide for creating an Azure Function. This is a high-level concept in order to be able to make the preparations correctly and completely so that no unwanted surprises occur later during implementation.

Prerequisites

Defined goal

A defined goal is necessary that we can check if our end product is doing what we are expecting. This can be as simple as one phrase.

Example:

My Azure Function should receive logs via HTTP and stores them into one log analytics. The response should indicate if the process was executed successfully.

Defined input

To know how to start the code, we first need to define the input(s) we need to get with the request to reach our goal. This can be static values or multiple dynamic values.

Example:

```
{
"logtype":"<logtypetoseperate>",
  "logbody":{
    "dynamictag1":"dynamicvalue1"
      ...
    }
}
```

In this case the object member "logtype" is static and only the value can be filled in on this specific field. On the other hand, "logbody" is dynamic. It should be possible to have one or one hundred members in the sub object "logbody".

Defined output

The output defines if the Function needs to return a value or which HTTP status code is expected.

Example:

Response: 200 OK

```
{
"language":"<languagecode>"
}
```

For the Function to work as intended, it should respond with the HTTP status code 200 and return this JSON structure.

Trigger method

The method is important that we know what oder which object(s) are triggering this function.

Example:

This function should be called from every Windows device via PowerShell. The trigger should be over HTTP.

Logic model

The logic model defines the logic of the code in a very high-level draft.

Example:

1. Get input
2. Check input values
3. Parse data for Log Analytics API
4. Send data to Log Analytics API
5. Get response from Log Analytics API
6. Send response

Decisions

Coding language

Azure Function Apps can execute different code with different languages. This is defined per Azure Function App and cannot be changed. Azure Functions offers these languages for code execution:

- PowerShell
- JavaScript

- Java
- C#
- Python

Billing plan

The billing plan defines if your function runs consumed based billing or premium plan billing. The benefits of each solution can be found here: [Pricing - Functions | Microsoft Azure]
For most use cases the consumption plan will be more than enough.

Procedure

Create Function App

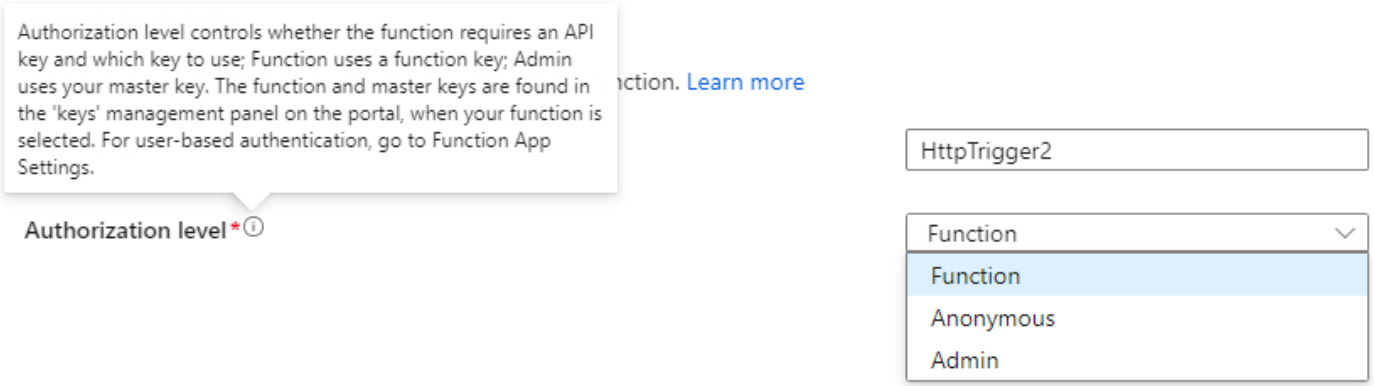
First you need an Azure Function App. On this you have to define the parts defined under "Descisions" and you assign the function into an Azure resource group.

Create Function

The Function itself holds the code which gets executed. It also is the unique identifier in the API URL to call if you use HTTP as a trigger. While creating a Function you can set the name and the trigger. For triggers you can choose between these ones:

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
Azure Event Hub trigger	A function that will be run whenever an event hub receives a new event

After the trigger and the name of the function is set you can choose the authorization level of the function.



For testing purposes, the Anonymous option is probably the easiest. For production functions it is recommended to use the Function level of authorization.

Code

When created you can copy your code into the web editor of the Function or start to develop the code directly in the Azure Portal.

Testing

For testing you can use the built in Test/Run option. If this behaves as it should, you can get the function URL with the push of a button. For testing the built-in default key is enough.

Create Function Key

The best practice is to use Function Keys as authorization on the Function level for production ready APIs. This code is then provided in the URL Query. For each workload which uses the same function it is best practice to use a separate Function Key so the access can be revoked per consumer.

Things to consider

Maximal runtime

The maximal runtime of one Azure Function is 230 seconds. After this time the function code gets cancelled and returns a 503 error.

Revision #7
Created 2022-12-05 16:37:07 UTC
Updated 2026-08-04 16:19:40 UTC by Luca Noah Caprez