FUNCTIONS AS A SERVICE (FAAS): A BEGINNER'S GUIDE



Computers are pretty simple. With them, you can store information, move information, and do things to that information. Services have risen to help people manage all three activities via:

- Data centers
- Network security
- Software applications

Functions as a Service (FaaS) are the boutique clothing shops of the software world. Each function performs a very specific operation. Here's what you need to know about FaaS—and how you can use them.

What are functions?

First, let's understand functions. A function defines a procedure on how to change one element into another. The function remains static, while the variables that pass through it can vary.

The advent of cloud services has enabled Functions as a Service—one of many as-a-service offerings the cloud has made possible.

What are functions as a service?

Functions as a Service offer you (and your organization) the ability to:

- Create high-grade, quality functions that people can't write themselves.
- Turn that function, or set of functions, into a service for people to use.

Functions as a Service places a function onto a cloud instance, making it available for use by developers, applications, and other functions. When you write a function to a cloud instance, that function can then be served to any computer *without* relying on the user's system hardware or software for their service to work.

It's like ordering takeout from a restaurant: they don't have to rely on the tools in your kitchen to prepare a delicious meal.

Cloud functions allow:

- The function to work on the cloud's resources—not an individual's computer
- The code to be written in any language
- The service to be accessed anywhere with a network connection

These functions are becoming more common in machine learning, where ML modeling time, development resources, and talent are all very valuable and the product of their efforts results in a function that's both hard to reproduce and useful for people. An example of an ML FaaS is Open-AI's GPT-3—you can access it via a REST API.

How FaaS works

Functions can be accessed through one of several triggers that you define when you create the function. For use in an app, functions can occur on events that happen in the database. For example, a function can be triggered when a new item is written to a database, changed, or deleted from the database.

An "on-event" function might send an email to a user when their account is created. A function could also be written to send a notification to a user, or set of users, in a chatroom when a new message has been written to the room (equivalent to a new write on the database).

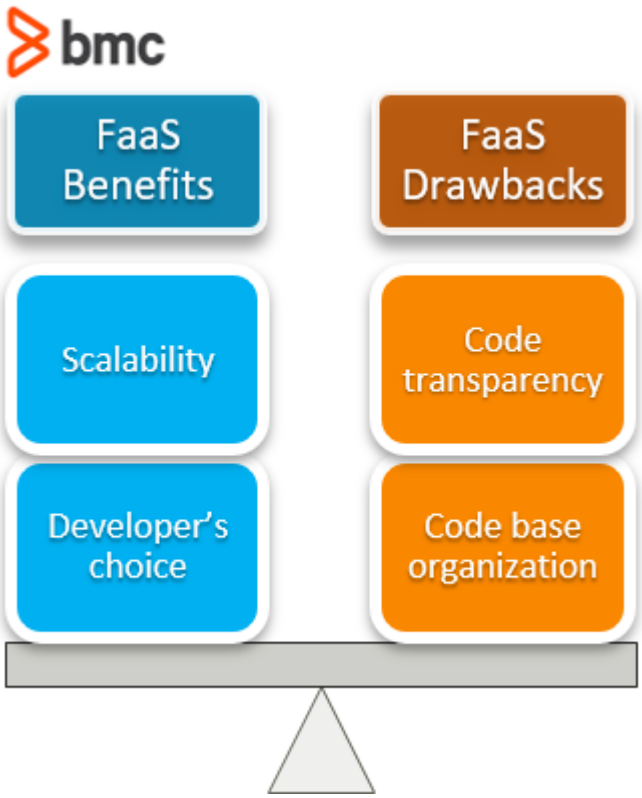
A function takes some inputs, and returns an output, so to use the function, you will have to know:

1. The data schema of the input so the function ran properly process the input data.
2. The data schema of the output so you know what to look for from the data.

Functions will vary entirely from use case to use case. Here are some examples:

- A function could take an input of a picture and output a label of cat or dog.
- A function could take input of a topic and output a sequence of words.
- A function could take in a YouTube video URL and output a JSON of all the video's stats.
- A function could take as input a word document and output a formatted PDF.

The larger cloud providers will have software development kits (SDKs) built to interact with their functions. If you wanted to write a function and allow other companies to use it, you can expose the function through an endpoint, commonly a REST API or, simply, a URL.



Advantages of FaaS

The two biggest FaaS benefits are overall scalability and the flexibility for the developer.

Scalability

A function on the cloud (especially with Kubernetes architecture) can scale to suit as many users as it needs. It also scales independently of other services.

If a search function suddenly becomes more popular than an alphabetizer function, the search function can scale separately from the alphabetizer, enabling the service to be offered to clients, while saving costs by only paying for the resources of the search function itself.

This scalability means:

- You won't pay for idle resources.
- There are fewer logistics when cloud resources manage servers.

Developer's choice

FaaS allow developers to work in whichever coding language they are familiar. Because of the container framework most devs operate in, you do not have to determine the primary operating system of your target customer and fit your dev skills to meet them.

This also opens your service to many more people because communication isn't being translated from OS to OS.

Disadvantages of FaaS

Drawbacks of functions as a service are lack of coding transparency and the organization of the code base.

Coding transparency

Testing code on the cloud still isn't totally developed. Bugs can be harder to spot than writing code and running it locally.

In most cloud systems today, there is not a good way to test the operability of a function until it is deployed. This requires either:

- More testing time
- More skilled developers to spot errors before deployment

Code base organization

Setting up Functions as a Service creates a new need: to organize functions in a modular way. It can be hard to keep track of the functions when you've deployed many.

Google recently released the Firebase Local Emulator Suite to help developers test their apps locally. Others have likely done so, too. It just shows they are trying to improve the interfaces in which developers interact with the cloud computing framework.

How to create & manage FaaS

Functions can be deployed on all the big cloud service providers. Because people will need to ping the function, the service provider can:

- Create access restrictions to the function
- Charge for its use
- Meter a user's activity (i.e.; 20 uses per day)
- Have a good idea of the function's costs

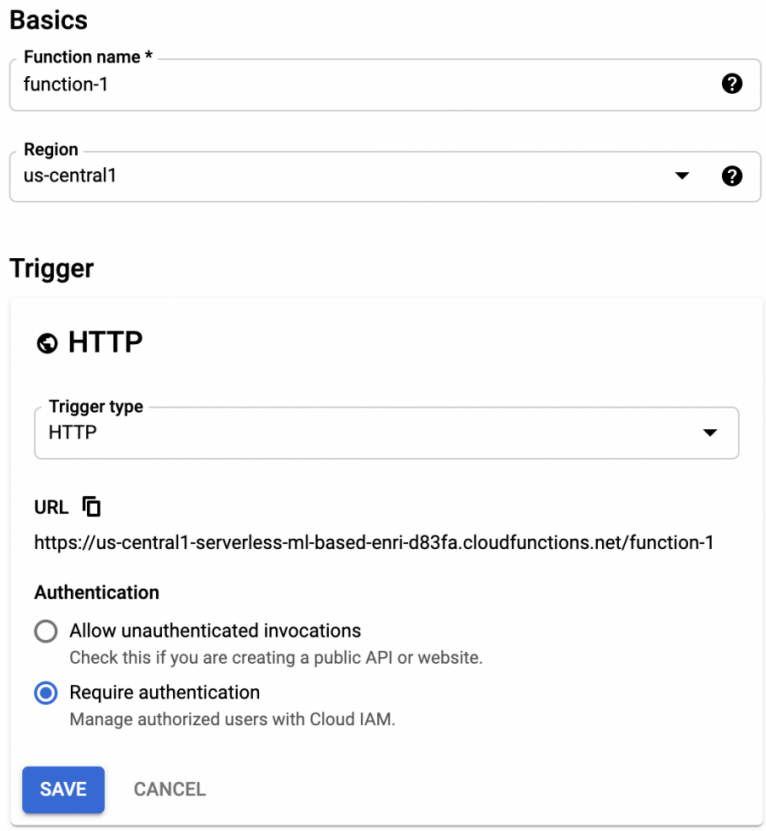
Creating a function

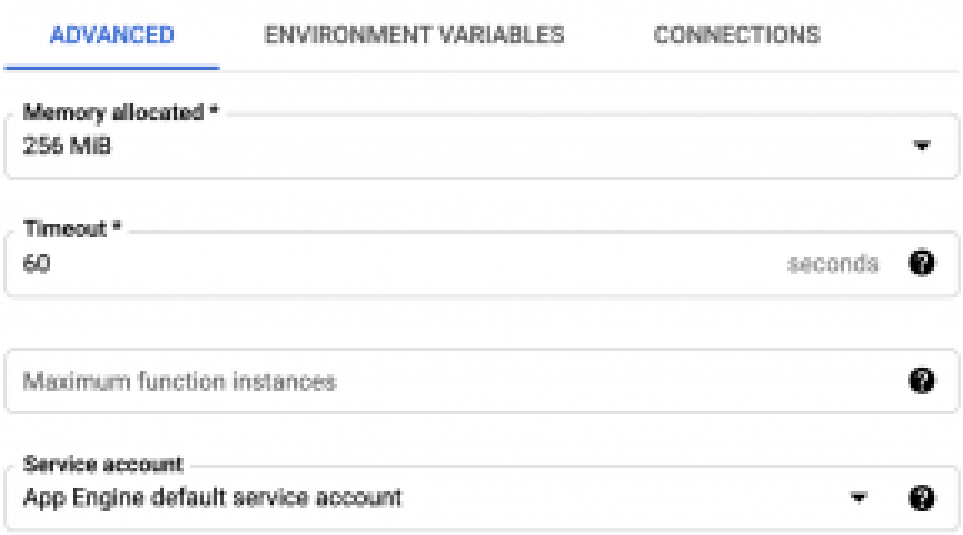
You can create a function from the GUI or deploy one from the Command Line. Each of the major cloud providers is improving their GUI interfaces to enable access for non-developers to engage with the process.

When creating a function, you generally have five options to choose and define:

1. Function Name
2. Region
3. Trigger Type
4. Memory Allocation
5. Timeout Value

Then the instance just needs the code. Here it is in GCP:





Managing a function

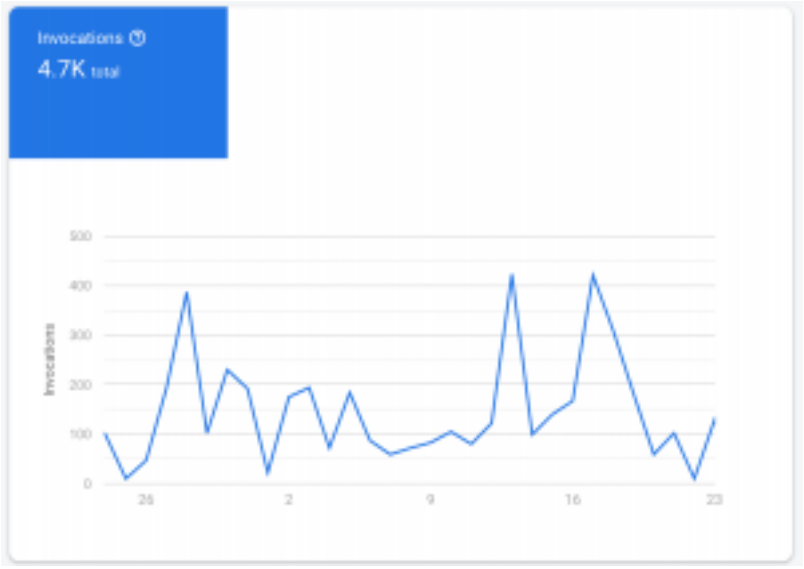
After it's deployed, you can manage a function from a list like this:

	Name ↑	Region	Trigger	Runtime	Memory allocated	Executed function	Last deployed
✓	analyze_sentiment	us-central1	HTTP	Python 3.7	256 MiB	sentiment_ts_document	Jun 9, 2019, 1:52:54 PM

Once it is deployed, Google's monitoring tools expose a lot of information on the function's container:

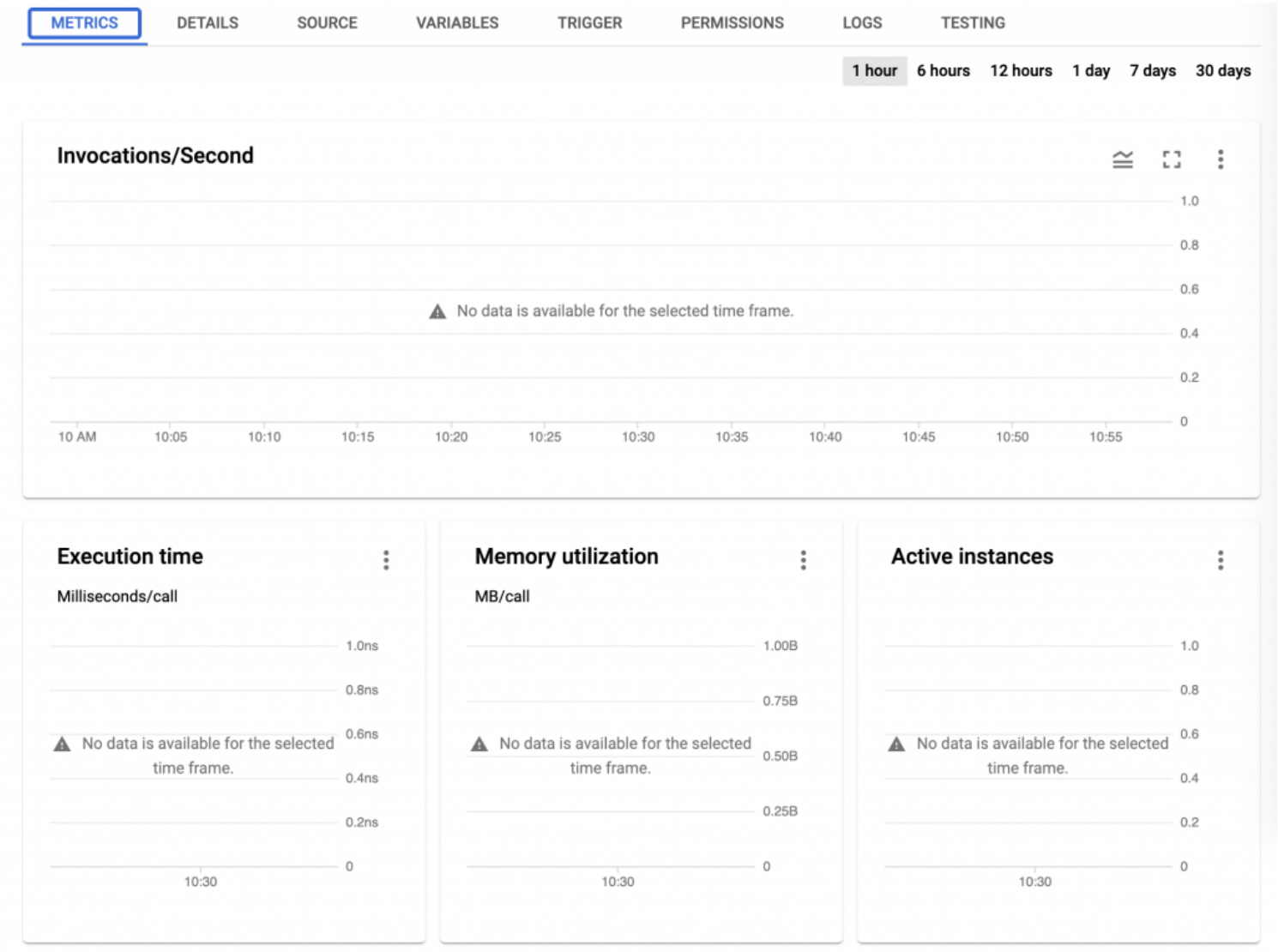
- Number of invocations
- Execution time
- Memory utilization

- Active instances
- Errors
- Logs



(A function invocations chart in GCP)

And here's what the functions dashboard looks like in GCP:



Related reading

- BMC DevOps Blog
- Google Cloud Functions: An Introduction
- FaaS vs Serverless: What's the Difference?
- Bring Kubernetes to the Serverless Party
- User Defined Functions (UDFs) in Snowflake, part of our Snowflake Guide
- API/Developer Portals: How To Create Great API Portals