**Azure Functions**

Traditional application development demands a consideration of the underlying IT infrastructure. For cloud computing, an IT team must create, monitor and pay for cloud computing instances -- regardless of how much work that instance actually does for the business.

Serverless compute can be thought of as a function as a service (FaaS), or a microservice that is hosted on a cloud platform. Your business logic runs as functions and you don't have to manually provision or scale infrastructure. The cloud provider manages infrastructure. Your app is automatically scaled out or down depending on load. Azure has several ways to build this sort of architecture. The two most common approaches are Azure Logic Apps and Azure Functions.

The idea behind serverless computing, also known as function as a service, is to eliminate those infrastructure considerations for the user. With serverless, a user can simply create and upload code, and then define the triggers or events that will execute the code. Triggers can come from a wide range of sources, including another user's application or other cloud services, such as databases and event and notification hubs.

Once a trigger or event occurs, it is the cloud provider's responsibility to load the code into a suitable execution environment, run the code and then release the compute resources. There are still servers involved, but the user no longer needs to provision or manage compute instances. In addition, rather than pay for those compute instances and other associated resources each month, users pay for serverless computing based on the amount of time a function runs in a given billing cycle.

Azure Functions competes with other serverless computing offerings, including Amazon Web Services Lambda and Google Cloud Functions.

**Benefits of a serverless compute solution**

Serverless compute is a great option for hosting business logic code in the cloud. With serverless offerings such as Azure Functions, you can write your business logic in the language of your choice. You get automatic scaling, you have no servers to manage, and you are charged based on what is used — not on reserved time. Here are some additional characteristics of a serverless solution for you to consider.

* **Avoids over-allocation of infrastructure**

Suppose you've provisioned VM servers and configured them with enough resources to handle your peak load times. When the load is light, you are potentially paying for infrastructure you're not using. Serverless computing helps solve the allocation problem by scaling up or down automatically, and you're only billed when your function is processing work.

* **Stateless logic**

Stateless functions are great candidates for serverless compute; function instances are created and destroyed on demand. If state is required, it can be stored in an associated storage service.

* **Event driven**

Functions are event driven. This means they run only in response to an event (called a "trigger"), such as receiving an HTTP request, or a message being added to a queue. You configure a trigger as part of the function definition. This approach simplifies your code by allowing you to declare where the data comes from (trigger/input binding) and where it goes (output binding). You don't need to write code to watch queues, blobs, hubs, etc. You can focus purely on the business logic.

* **Functions can be used in traditional compute environments**

Functions are a key component of serverless computing, but they are also a general compute platform for executing any type of code. Should the needs of your app change, you can take your project and deploy it in a non-serverless environment, which gives you the flexibility to manage scaling, run on virtual networks, and even completely isolate your functions.

**Drawbacks of a serverless compute solution**

Serverless compute will not always be the appropriate solution to hosting your business logic. Here are a few characteristics of functions that may affect your decision to host your services in serverless compute.

* **Execution time**

By default, functions have a timeout of 5 minutes. This timeout is configurable to a maximum of 10 minutes. If your function requires more than 10 minutes to execute, you can host it on a VM. Additionally, if your service is initiated through an HTTP request and you expect that value as an HTTP response, the timeout is further restricted to 2.5 minutes. Finally, there's also an option called Durable Functions that allows you to orchestrate the executions of multiple functions without any timeout.

* **Execution frequency**

The second characteristic is execution frequency. If you expect your function to be executed continuously by multiple clients, it would be prudent to estimate the usage and calculate the cost of using functions accordingly. It might be cheaper to host your service on a VM.

While scaling, only one function app instance can be created every 10 seconds, for up to 200 total instances. Keep in mind, each instance can service multiple concurrent executions, so there is no set limit on how much traffic a single instance can handle. Different types of triggers have different scaling requirements, so research your choice of trigger and investigate its limits.

**Choosing a service plan**

Function apps may use one of two types of service plans.

The first service plan is the **Consumption service plan**. This is the plan that you choose when using the Azure serverless application platform. The Consumption service plan provides automatic scaling and bills you when your functions are running. The Consumption plan comes with a configurable timeout period for the execution of a function. By default, it is 5 minutes, but may be configured to have a timeout as long as 10 minutes.

The second plan is called the **Azure App Service plan**. This plan allows you to avoid timeout periods by having your function run continuously on a VM that you define. When using an App Service plan, you are responsible for managing the app resources the function runs on, so this is technically not a serverless plan. However, it may be a better choice if your functions are used continuously or if your functions require more processing power or execution time than the Consumption plan can provide.

**Azure Functions usage**

Serverless computing functions are typically not complete, full-featured applications. Instead, functions handle specific, short-lived tasks. Most functions involve some form of data processing, such as image or order processing, as well as file maintenance or data collection from internet of things (IoT) devices.

**Developing Azure Functions**

Azure Functions supports functions developed in C#, F#, Node.js, Python, PHP, batch, bash and any executable file format. Azure Functions also supports the NuGet open source package manager and the Node Package Manager for JavaScript, allowing developers to use popular libraries.

Developers can code Azure Functions directly within the Azure portal, but can also manage continuous integration (CI) and deploy code through GitHub, Microsoft Visual Studio Team Services and other development tools, such as Xcode, Eclipse and IntelliJ IDEA

**Pricing Plans for Functions**

Businesses can currently select between two pricing plans for Azure Functions: **consumption** and **Azure App Service**. In the consumption pricing plan, users pay for the time that each function actually runs. The other plan is for users that already employ Azure App Service to run web, mobile and API-based apps. These users can run Azure Functions at no added cost.

**Size and time limit**

Azure Functions has no limits on payload size, deployment package size and code or dependency size. However, each Azure Functions host instance is limited to 1.5 GB of memory, and the default execution time limit is 5 minutes -- although, users can increase this to 10 minutes.

Azure Functions does require an Azure Storage account that must support Azure Blob, Queue and Table storage services. Azure Functions uses Azure Storage to manage triggers and log function execution. The storage demands for Azure Functions may present an added cost to users.

**Useful links:** <https://docs.microsoft.com/en-us/learn/modules/create-serverless-logic-with-azure-functions/>

**What is a trigger?**

A trigger is an object that defines how an Azure function is invoked. For example, if you want a function to execute every 10 minutes, you could use a timer trigger. Every function must have exactly one trigger associated with it. If you want to execute a piece of logic that runs under multiple conditions, you need to create multiple functions that share the same core function code.

**Types of triggers**

Azure Functions support a wide range of trigger types. Here are some of the most common types:

| **Type** | **Purpose** |
| --- | --- |
| Timer | Execute a function at a set interval. |
| HTTP | Execute a function when an HTTP request is received. |
| Blob | Execute a function when a file is uploaded or updated in Azure Blob storage. |
| Queue | Execute a function when a message is added to an Azure Storage queue. |
| Cosmos DB | Execute a function when a document changes in a collection. |
| Event Hub | Execute a function when an event hub receives a new event. |

**What is a binding?**

A binding is a connection to data within your function. Bindings are optional and come in the form of input and output bindings. An input binding is the data that your function receives. An output binding is the data that your function sends. Unlike a trigger, a function can have multiple input and output bindings.

A trigger is a special type of input binding that has the additional capability of initiating execution.

**Types of supported bindings**

The type of binding defines where we are reading or sending data. There is a binding to respond to web requests and a large selection of bindings to interact directly with various Azure services as well as third-party services.

A binding type can be used as an input, an output or both. For example, a function can write to Azure Blob Storage output binding, but a blob storage update could trigger another function.

Some common binding types are listed below:

* Blob Storage
* Azure Service Bus Queues
* Azure Cosmos DB
* Azure Event Hubs
* External Files
* External Tables
* HTTP endpoints

**Binding properties**

Three properties are required in all bindings. You may have to supply additional properties based on the type of binding and storage you are using.

* **Name** Defines the function parameter through which you access the data. For example, in a queue input binding, this is the name of the function parameter that receives the queue message content.
* **Type** Identifies the type of binding, i.e., the type of data or service we want to interact with.
* **Direction** Indicates the direction data is flowing, i.e., is it an input or output binding?

Additionally, most binding types also need a fourth property:

* **Connection** Provides the name of an app setting key that contains the connection string. Bindings use connection strings stored in app settings to keep secrets out of the function code. This makes your code more configurable and secure.

**Sample binding:**

{

"bindings": [

{

"name": "order",

"type": "queueTrigger",

"direction": "in",

"queueName": "myqueue-items",

"connection": "MY\_STORAGE\_ACCT\_APP\_SETTING"

},

{

"name": "$return",

"type": "table",

"direction": "out",

"tableName": "outTable",

"connection": "MY\_TABLE\_STORAGE\_ACCT\_APP\_SETTING"

}

]

}

**Example Scenario for Timer trigger**

1. It's common to execute a piece of logic at a set interval. Imagine you're a blog owner and you notice that your subscribers aren't reading your most recent posts. You decide that the best action is to send an email once a week to remind them to check your blog. You implement this logic using an Azure function app with a timer trigger to invoke your function weekly.
2. Imagine a scenario where a busy hair salon has a recurring problem: their customers commonly miss their appointments. The appointments are reserved time slots. If a customer misses an appointment, the salon loses money. To fix this problem, the salon reaches out to you, a software developer. To improve the problem, you decide to send reminder text messages. These could be sent as soon as the appointment is scheduled or changed, and every morning, you'll send a text message to every customer with an appointment that day.
3. Suppose you run a social networking site for professionals. You're allowing your users to upload their headshot images to be posted on their profile. To reduce the workload on the web server, you want to create a serverless backend using Azure Functions to process this data. You want to create an image thumbnail and then save it off to permanent storage.

**What is a timer trigger?**

A timer trigger is a trigger that executes a function at a consistent interval. To create a timer trigger, you need to supply two pieces of information.

1. A Timestamp parameter name, which is simply an identifier to access the trigger in code.
2. A Schedule, which is a CRON expression that sets the interval for the timer.

**What is a CRON expression?**

A CRON expression is a string that consists of six fields that represent a set of times.

The order of the six fields in Azure is: *{second} {minute} {hour} {day} {month} {day of the week}*.

For example, a CRON expression to create a trigger that executes every five minutes looks like:

0 \*/5 \* \* \* \*

To build a CRON expression, you need to have a basic understanding of some of the special characters.

| **Special character** | **Meaning** | **Example** |
| --- | --- | --- |
| \* | Selects every value in a field | An asterisk "\*" in the day of the week field means *every* day. |
| , | Separates items in a list | A comma "1,3" in the day of the week field means just Mondays (day 1) and Wednesdays (day 3). |
| - | Specifies a range | A hyphen "10-12" in the hour field means a range that includes the hours 10, 11, and 12. |
| / | Specifies an increment | A slash "\*/10" in the minutes field means an increment of every 10 minutes. |

Now we'll go back to the original CRON expression example. Let’s try to understand it better by breaking it down field by field.

0 \*/5 \* \* \* \*

The first field represents seconds. This field supports the values 0-59. Because the field contains a zero, it selects the first possible value, which is one second.

The second field represents minutes. The value "\*/5" contains two special characters. First, the asterisk (\*) means "select every value within the field." Because this field represents minutes, the possible values are 0-59. The second special character is the slash (/), which represents an increment. When you combine these characters together, it means for all values 0-59, select every fifth value. An easier way to say that is simply "every five minutes."

The remaining four fields represent the hour, day, month, and weekday of the week. An asterisk for these fields means to select every possible value. In this example, we select "every hour of every day of every month."

**What is Http Trigger?**

An HTTP request is a common operation on most platforms and devices. Whether it's a request to look up a word in a dictionary or to get the local weather, we send HTTP requests all the time. Azure Functions allows us to quickly create a piece of logic to execute when an HTTP request is received.

An HTTP trigger is a trigger that executes a function when it receives an HTTP request. HTTP triggers have many capabilities and customizations, including:

* Provide authorized access by supplying keys.
* Restrict which HTTP verbs are supported.
* Return data back to the caller.
* Receive data through query string parameters or through the request body.
* Support URL route templates to modify the function URL.

When you create an HTTP trigger, select a programming language, provide a trigger name, and select an Authorization level.

**What is an HTTP trigger Authorization level?**

An HTTP trigger Authorization level is a flag that indicates if an incoming HTTP request needs an API key for authentication reasons.

There are three Authorization levels:

* Function
* Admin
* Anonymous

The Function and Admin levels are "key" based. To send an HTTP request, you must supply a key for authentication. There are two types of keys: **function** and **host**. The difference between the two keys is their scope. Function keys are specific to a function. Host keys apply to all functions inside the function app. If your Authorization level is set to Function, you can use either a function or a host key. If your Authorization level is set to Admin, you must supply a host key. The Anonymous level means that there's no authentication required.

**What is a blob trigger?**

A blob trigger is a trigger that executes a function when a file is uploaded or updated in Azure Blob storage. To create a blob trigger, you create an Azure Storage account and provide a location that the trigger monitors.