Azure Functions overview

Article • 05/24/2023

Azure Functions is a serverless solution that allows you to write less code, maintain less infrastructure, and save on costs. Instead of worrying about deploying and maintaining servers, the cloud infrastructure provides all the up-to-date resources needed to keep your applications running.

You focus on the code that matters most to you, in the most productive language for you, and Azure Functions handles the rest.

For the best experience with the Functions documentation, choose your preferred development language from the list of native Functions languages at the top of the article.

Scenarios

Functions provides a comprehensive set of event-driven triggers and bindings that connect your functions to other services without having to write extra code.

The following are a common, *but by no means exhaustive*, set of integrated scenarios that feature Functions.

If you want to...	**then...**
Process file uploads	Run code when a file is uploaded or changed in blob storage.
Process data in real time	Capture and transform data from event and IoT source streams on the way to storage.
Infer on data models	Pull text from a queue and present it to various AI services for analysis and classification.
Run scheduled task	Execute data clean-up code on pre-defined timed intervals.
Build a scalable web API	Implement a set of REST endpoints for your web applications using HTTP triggers.
Build a serverless workflow	Create an event-driven workflow from a series of functions using Durable Functions.
Respond to database changes	Run custom logic when a document is created or updated in Azure Cosmos DB.

If you want to...	then...
Create reliable message systems	Process message queues using Queue Storage, Service Bus, or Event Hubs.

These scenarios allow you to build event-driven systems using modern architectural patterns. For more information, see Azure Functions Scenarios.

Development lifecycle

With Functions, you write your function code in your preferred language using your favorite development tools and then deploy your code to the Azure cloud. Functions provides native support for developing in C#, Java, JavaScript, PowerShell, Python, plus the ability to use more languages, such as Rust and Go.

Functions integrates directly with Visual Studio, Visual Studio Code, Maven, and other popular development tools to enable seemless debugging and deployments.

Functions also integrates with Azure Monitor and Azure Application Insights to provide comprehensive runtime telemetry and analysis of your functions in the cloud.

Hosting options

Functions provides a variety hosting options for your business needs and application workload. Event-driven scaling hosting options range from fully serverless, where you only pay for execution time (Consumption plan), to always warm instances kept ready for fastest response times (Premium plan).

When you have excess App Service hosting resources, you can host your functions in an existing App Service plan. This kind of Dedicated hosting plan is also a good choice when you need predictable scaling behaviors and costs from your functions.

If you want complete control over your functions runtime environment and dependencies, you can even deploy your functions in containers that you can fully customize. Your custom containers can be hosted by Functions, deployed as part of a microservices architecture in Azure Container Apps, or even self-hosted in Kubernetes.

Next Steps

Azure Functions Scenarios

Get started through lessons, samples, and interactive tutorials

Azure Functions scenarios

Article • 05/24/2023

We often build systems to react to a series of critical events. Whether you're building a web API, responding to database changes, processing event streams or messages, Azure Functions can be used to implement them.

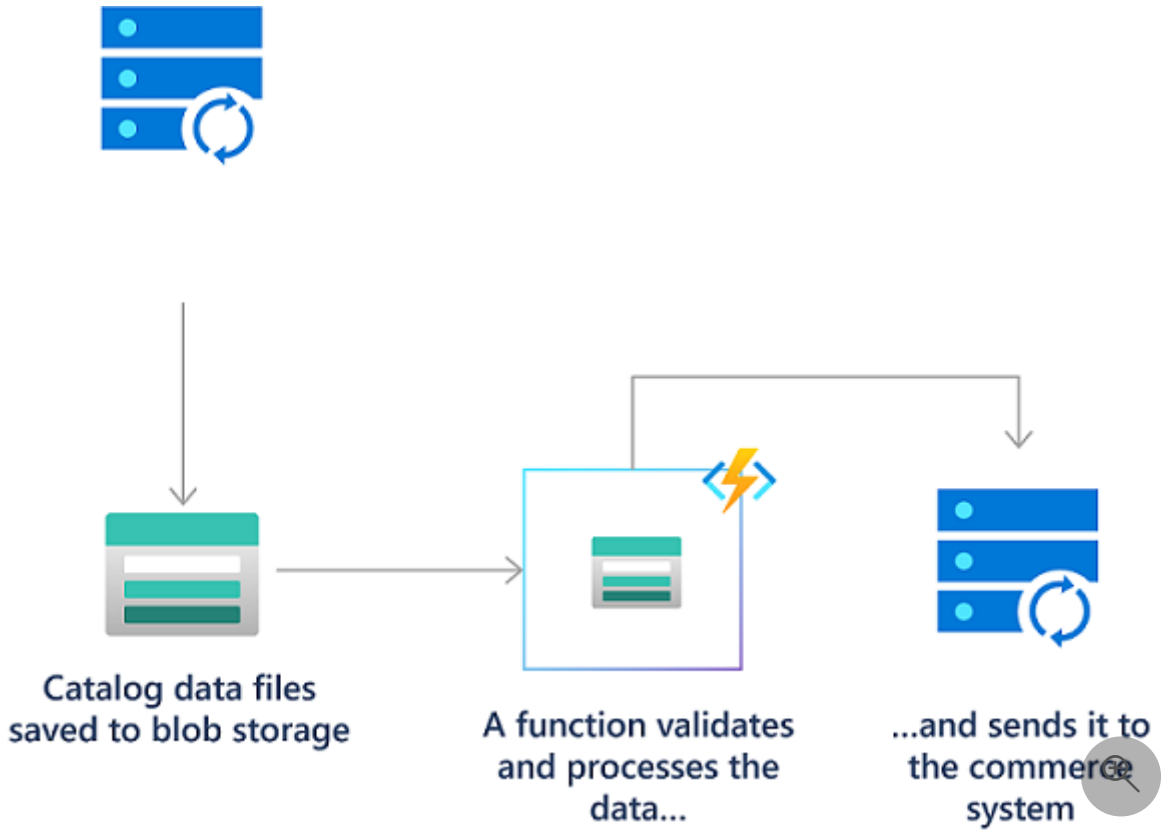
In many cases, a function integrates with an array of cloud services to provide feature-rich implementations. The following are a common (but by no means exhaustive) set of scenarios for Azure Functions.

Select your development language at the top of the article.

Process file uploads

There are several ways to use functions to process files into or out of a blob storage container. To learn more about options for triggering on a blob container, see Working with blobs in the best practices documentation.

For example, in a retail solution, a partner system can submit product catalog information as files into blob storage. You can use a blob triggered function to validate, transform, and process the files into the main system as they're uploaded.



The following tutorials use an Event Grid trigger to process files in a blob container:

For example, using the blob trigger with an event subscription on blob containers:

C#

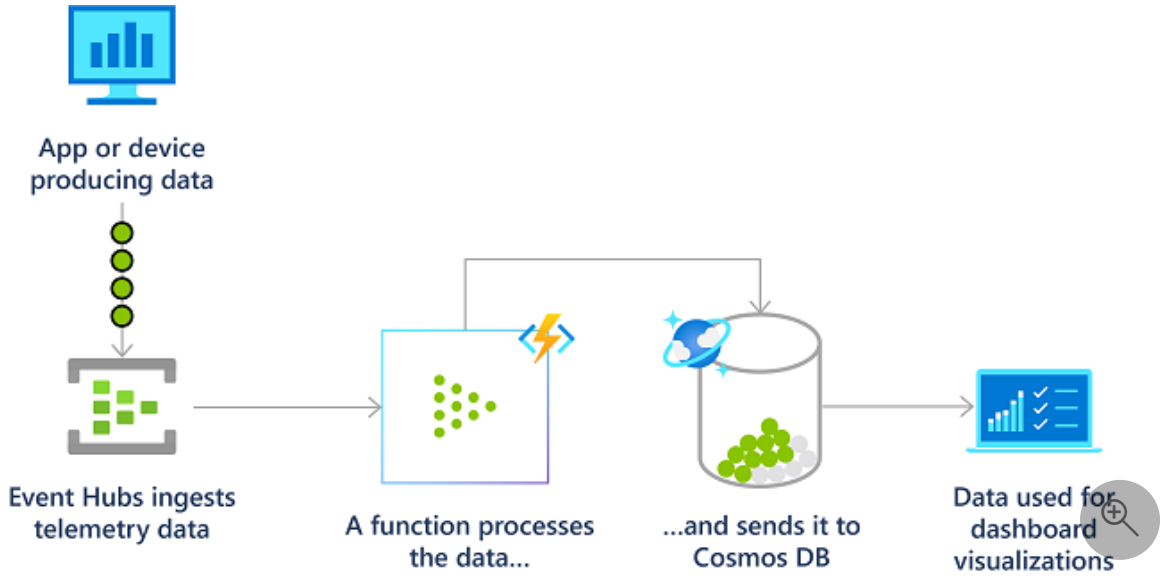
```
[FunctionName("ProcessCatalogData")]
public static async Task Run([BlobTrigger("catalog-uploads/{name}", Source =
BlobTriggerSource.EventGrid, Connection = "
<NAMED_STORAGE_CONNECTION>")]Stream myCatalogData, string name, ILogger log)
{
    log.LogInformation($"C# Blob trigger function Processed blob\n Name:
{name} \n Size: {myCatalogData.Length} Bytes");

    using (var reader = new StreamReader(myCatalogData))
    {
        var catalogEntry = await reader.ReadLineAsync();
        while(catalogEntry !=null)
        {
            // Process the catalog entry
            // ...

            catalogEntry = await reader.ReadLineAsync();
        }
    }
}
```

- Upload and analyze a file with Azure Functions and Blob Storage
- Automate resizing uploaded images using Event Grid
- Trigger Azure Functions on blob containers using an event subscription

Real-time stream and event processing

So much telemetry is generated and collected from cloud applications, IoT devices, and networking devices. Azure Functions can process that data in near real-time as the hot path, then store it in Cosmos DB for use in an analytics dashboard.

Your functions can also use low-latency event triggers, like Event Grid, and real-time outputs like SignalR to process data in near-real-time.



For example, using the event hubs trigger to read from an event hub and the output binding to write to an event hub after debatching and transforming the events:

C#

```
[FunctionName("ProcessorFunction")]
public static async Task Run(
    [EventHubTrigger(
        "%Input_EH_Name%",
        Connection = "InputEventHubConnectionString",
        ConsumerGroup = "%Input_EH_ConsumerGroup%")] EventData[]
inputMessages,
    [EventHub(
        "%Output_EH_Name%",
        Connection = "OutputEventHubConnectionString")]
IAsyncCollector<SensorDataRecord> outputMessages,
    PartitionContext partitionContext,
    ILogger log)
{
    var debatcher = new Debatcher(log);
    var debatchedMessages = await debatcher.Debatch(inputMessages,
partitionContext.PartitionId);

    var xformer = new Transformer(log);
    await xformer.Transform(debatchedMessages, partitionContext.PartitionId,
outputMessages);
}
```

- Streaming at scale with Azure Event Hubs, Functions and Azure SQL
- Streaming at scale with Azure Event Hubs, Functions and Cosmos DB
- Streaming at scale with Azure Event Hubs with Kafka producer, Functions with Kafka trigger and Cosmos DB
- Streaming at scale with Azure IoT Hub, Functions and Azure SQL
- Azure Event Hubs trigger for Azure Functions

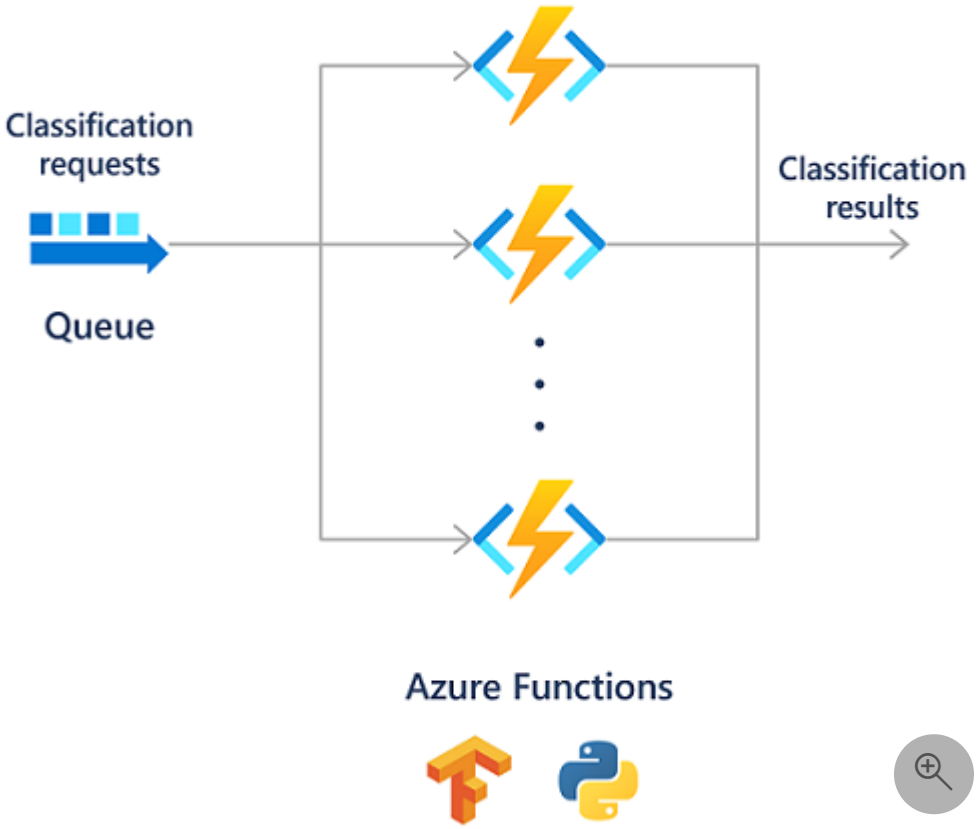
- Apache Kafka trigger for Azure Functions

Machine learning and AI

Besides data processing, Azure Functions can be used to infer on models.

For example, a function that calls a TensorFlow model or submits it to Azure AI services can process and classify a stream of images.

Functions can also connect to other services to help process data and perform other AI-related tasks, like text summarization.



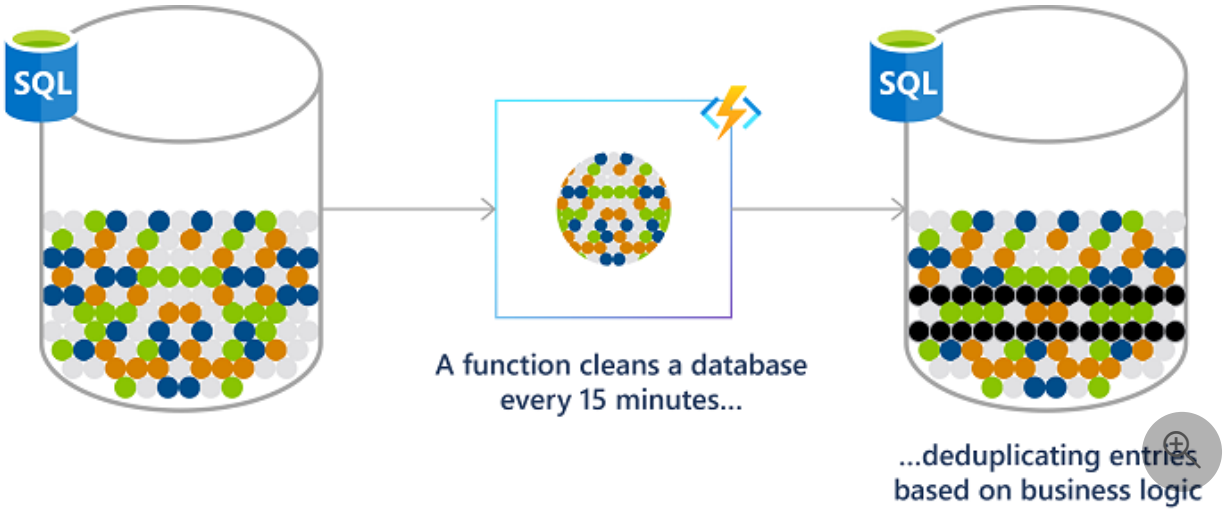
- Sample: Text summarization using AI Cognitive Language Service

Run scheduled tasks

Functions enables you to run your code based on a cron schedule that you define.

Check out how to Create a function in the Azure portal that runs on a schedule.

A financial services customer database, for example, might be analyzed for duplicate entries every 15 minutes to avoid multiple communications going out to the same customer.



C#

```
[FunctionName("TimerTriggerCSharp")]
public static void Run([TimerTrigger("0 */15 * * * *")]TimerInfo myTimer,
ILogger log)
{
    if (myTimer.IsPastDue)
    {
        log.LogInformation("Timer is running late!");
    }
    log.LogInformation($"C# Timer trigger function executed at:
{DateTime.Now}");

    // Perform the database deduplication
}
```

- Timer trigger for Azure Functions

Build a scalable web API

An HTTP triggered function defines an HTTP endpoint. These endpoints run function code that can connect to other services directly or by using binding extensions. You can compose the endpoints into a web-based API.

You can also use an HTTP triggered function endpoint as a webhook integration, such as GitHub webhooks. In this way, you can create functions that process data from GitHub events. To learn more, see Monitor GitHub events by using a webhook with Azure Functions.

For examples, see the following:

C#

```
[FunctionName("InsertName")]
public static async Task<IActionResult> Run(
    [HttpTrigger(AuthorizationLevel.Function, "post")] HttpRequest req,
    [CosmosDB(
        databaseName: "my-database",
        collectionName: "my-container",
        ConnectionStringSetting =
"CosmosDbConnectionString")]IAsyncCollector<dynamic> documentsOut,
    ILogger log)
{
    string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
    dynamic data = JsonConvert.DeserializeObject(requestBody);
    string name = data?.name;

    if (name == null)
    {
        return new BadRequestObjectResult("Please pass a name in the request
body json");
    }

    // Add a JSON document to the output container.
    await documentsOut.AddAsync(new
    {
        // create a random ID
        id = System.Guid.NewGuid().ToString(),
        name = name
    });

    return new OkResult();
}
```

- Article: Create serverless APIs in Visual Studio using Azure Functions and API Management integration
- Training: Expose multiple function apps as a consistent API by using Azure API Management
- Sample: Web application with a C# API and Azure SQL DB on Static Web Apps and Functions
- Azure Functions HTTP trigger