

# Azure Functions

---

CLIVE CIAPPARA



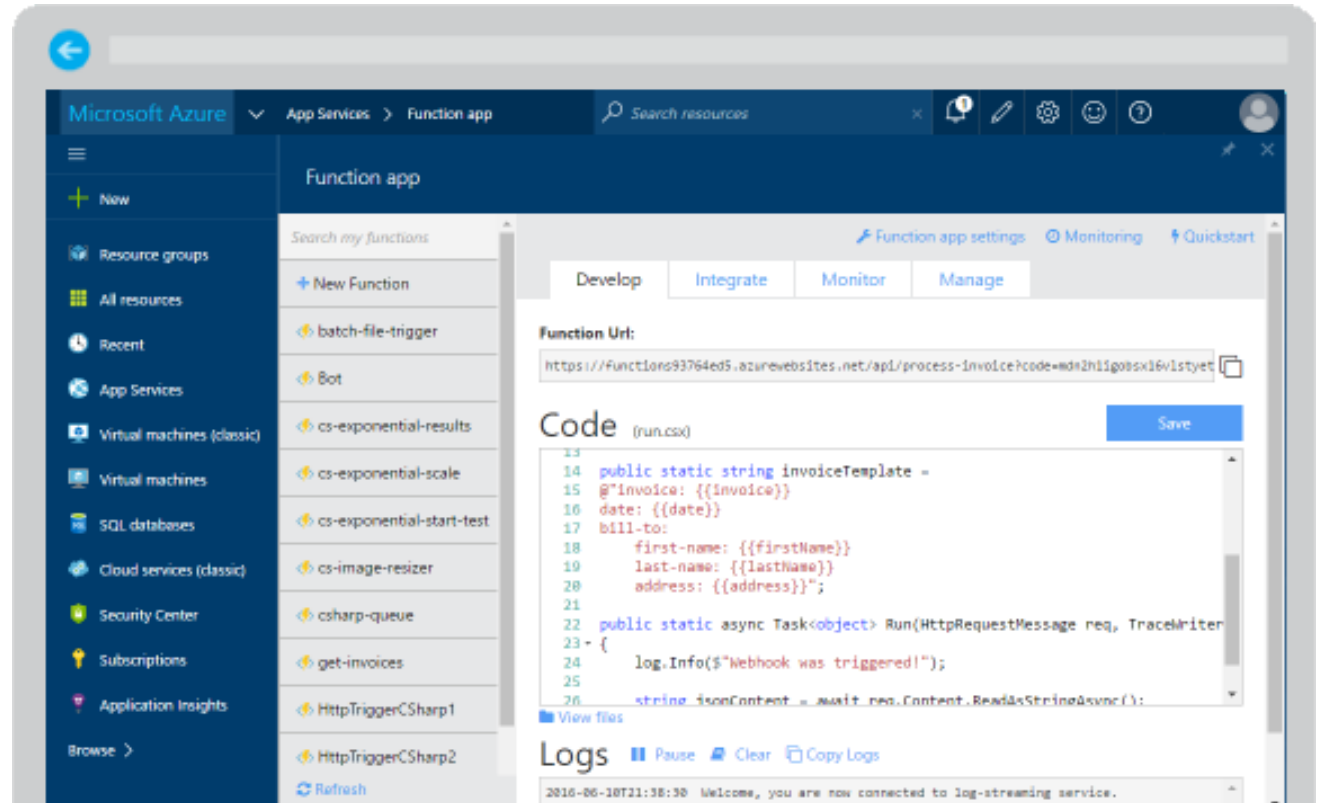
[ciappara.com](http://ciappara.com)



[@cliveciappara](https://twitter.com/cliveciappara)

# Azure Functions

Create a “serverless” event-driven experience that extends the existing Azure App Service platform by building “nanoservices” that can scale based on demand



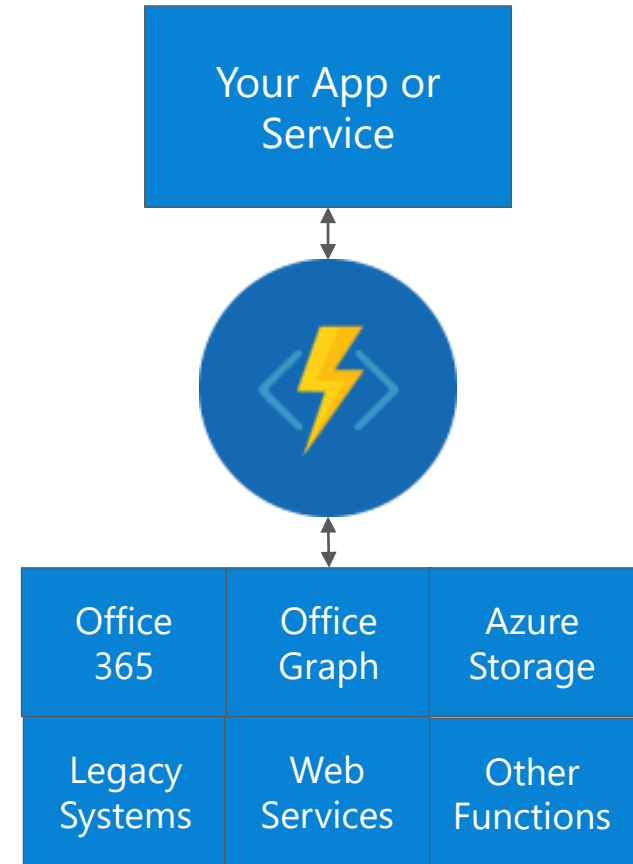
# Supported Languages and Tools

Create functions in JavaScript, C#, Python, and PHP, as well as scripting options such as Bash, Batch, and PowerShell, that can be triggered by virtually any event in Azure, 3rd party services, or on premise systems

The screenshot displays the Azure Functions configuration interface. On the left is a sidebar with settings: Runtime version: Latest (~0.4), Memory Size (with a slider), Features, Continuous Integration (Deploy your function code from Git), Authentication/Authorization (For functions that use the HTTP trigger, you can require calls to be authenticated.), CORS (Allow your HTTP-triggered functions to be called from within a web browser.), and API definition (Allow clients to more easily consume your HTTP-triggered functions.). The main panel features a blue banner titled "The faster way to functions" with the text: "Write any function in minutes - whether to run a simple job that cleans up a database or to build a more complex architecture. Creating functions is easier than ever before, whatever your chosen OS, platform, or development method. No install required." Below the banner, it says "Get started quickly with a premade function" and "1) Choose a scenario:". Three scenario icons are shown: "Timer" (clock icon), "Data processing" (database icon), and "Webhook + API" (code icon). To the right of the scenarios are three buttons: "Authentication", "Configure CORS", and "Configure API metadata".

# Common Scenarios

- Timer-based processing
- Azure service event processing
- SaaS event processing
- Serverless web application architectures
- Serverless mobile backends
- Real-time stream processing
- Real-time bot messaging



# Function App Templates

Function App templates are categorized into general areas of Timer, Data Processing, and Webhook & API

Choose a template

Language:  Scenario:

<b>BlobTrigger - C#</b> A C# function that will be run whenever a blob is added to a specified container	<b>BlobTrigger - Node</b> A Node.js function that will be run whenever a blob is added to a specified container	<b>Empty - C#</b> An empty C# function without triggers, inputs, or outputs	<b>Empty - Node</b> An empty Node.js function without triggers, inputs, or outputs
<b>EventHubTrigger - Node</b> A Node.js function that will be run whenever an event hub receives a new event	<b>Generic Webhook - C#</b> A C# function that will be run whenever it receives a webhook request	<b>Generic Webhook - Node</b> A Node.js function that will be run whenever it receives a webhook request	<b>GitHub WebHook - C#</b> A C# function that will be run whenever it receives a GitHub webhook request

- BlobTrigger
- EventHubTrigger
- Generic webhook
- GitHub webhook
- HTTPTrigger
- QueueTrigger
- ServiceBusQueueTrigger
- ServiceBusTopicTrigger
- TimerTrigger
- Blank & Experimental

# Timer Function Apps

- Run at explicitly specified intervals, like every day at 2:00 am using CRON expressions, like "0 \*/5 \* \* \* \*" (every 5 minutes)
- Can send information to other systems, but typically don't "return" information, only write to logs
- Great for redundant cleanup and data management
- Great for checking state of services
- Can be combined with other functions

# Data Processing Function Apps

- Run when triggered by a data event, such as an item being added to a queue or container
- Typically have in and out parameters
- Great for responding to CRUD events
- Great for performing CRUD events
- Great for moving content
- Access data across services



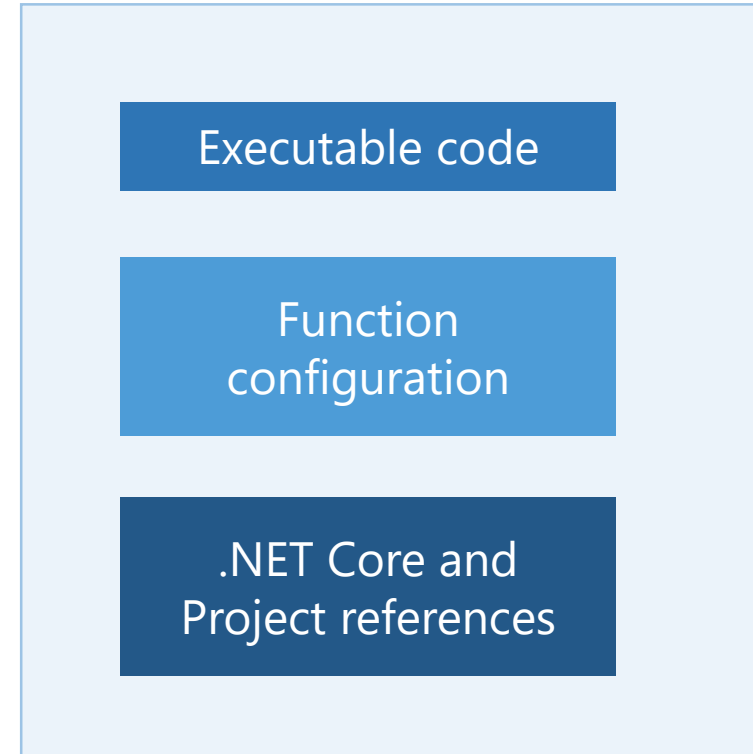
# Webhook & API Function Apps

- Triggered by events in other services, like GitHub, Team Foundation Services, Office 365, OneDrive, Microsoft PowerApps
- Takes in a request and sends back a response
- Often mimic Web API and legacy web services flows
- Typically need CORS settings managed
- Best for exposing functionality to other apps and services
- Great for building Logic Apps



# Anatomy of a Function

- A "Run" file that containing the function code
- A "Function" file containing all service and trigger bindings and parameters
- A "Project" file containing project assembly and NuGet package references
- App Service settings, such as connection strings and API keys



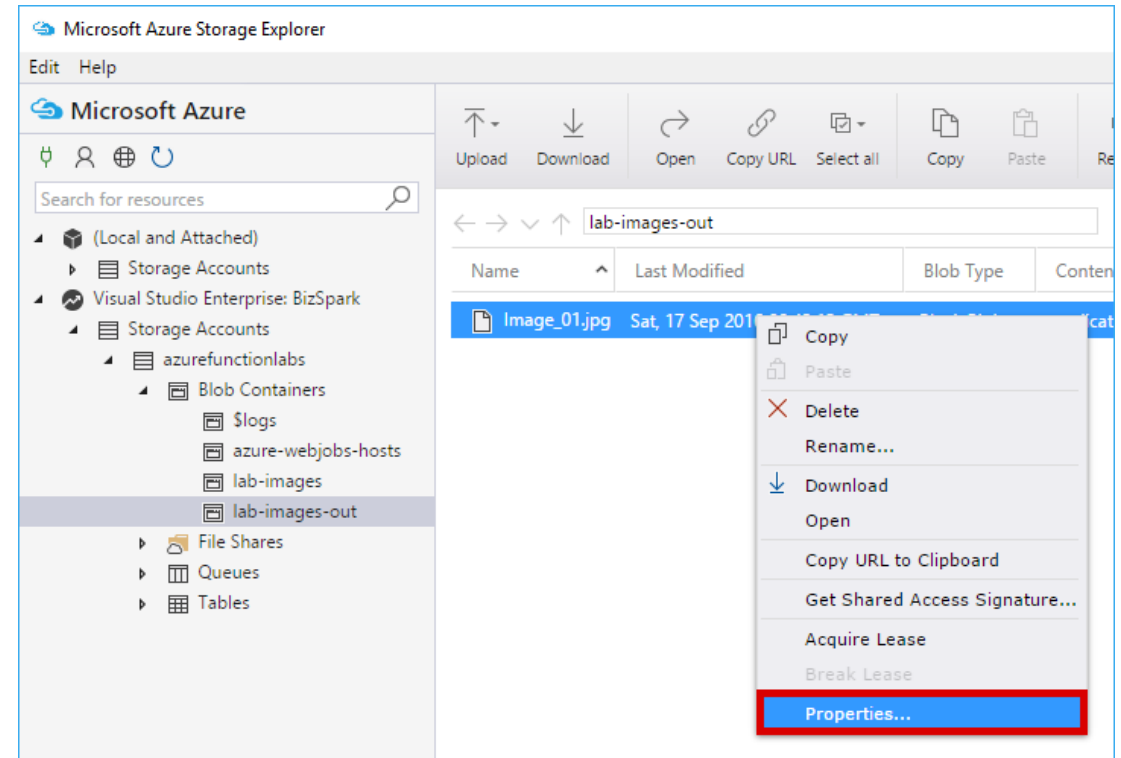
# Function Bindings

Bindings serve as the basis for all connections to and from a function. Many bindings can be “bi-directional” as well.

Type	Service	Trigger	Input	Output
Schedule	Azure Functions	✓		
HTTP (REST or webhook)	Azure Functions	✓		✓*
Blob Storage	Azure Storage	✓	✓	✓
Events	Azure Event Hubs	✓		✓
Queues	Azure Storage	✓		✓
Tables	Azure Storage		✓	✓
Tables	Azure Mobile Apps		✓	✓
No-SQL DB	Azure DocumentDB		✓	✓
Push Notifications	Azure Notification Hubs			✓

# Testing Functions

- Command-line tools
- 3<sup>rd</sup> party products such as Postman and Swagger
- Direct web calls via cURL
- Nested functions
- Microsoft Azure Storage Explorer
- Visual Studio Cloud Explorer





© 2016 Microsoft Corporation. All rights reserved. Microsoft, Windows, Windows Vista and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.