Azure Serverless Computing

HELLO!

I am Abhishek Gupta

PRINCIPAL CONSULTANT

Xpirit (Part of Xebia Group)

Microsoft Certified Trainer

CloudAndMobileBlog.com

@AbhiForTweeting

Agenda

What is Serverless

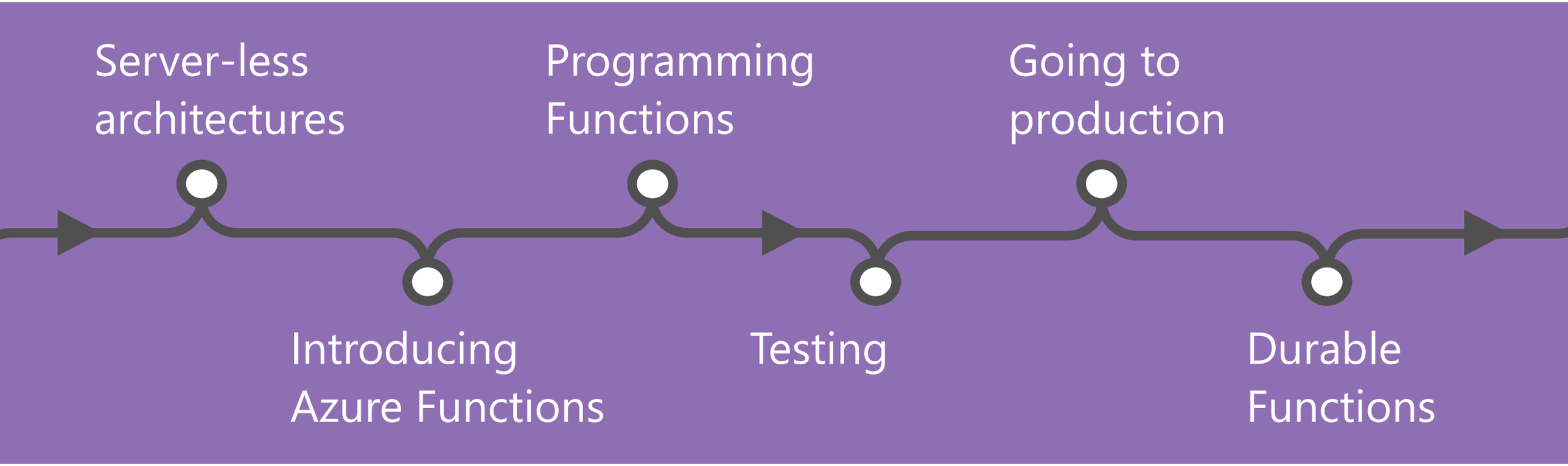
Why Serverless

Azure Functions

Cosmos DB with Azure Function

Azure Durable Functions

Presentation agenda



Server-less architectures and Azure Functions

Clear skies shouldn't cost much

Based on @Dougward
during Global Azure Bootcamp 2018

Pay only for the lightning bolts

Handle a lot of lightning bolts

Back to sunshine after the storm

Transitioning to server-less

There are no servers…

When you do not need them

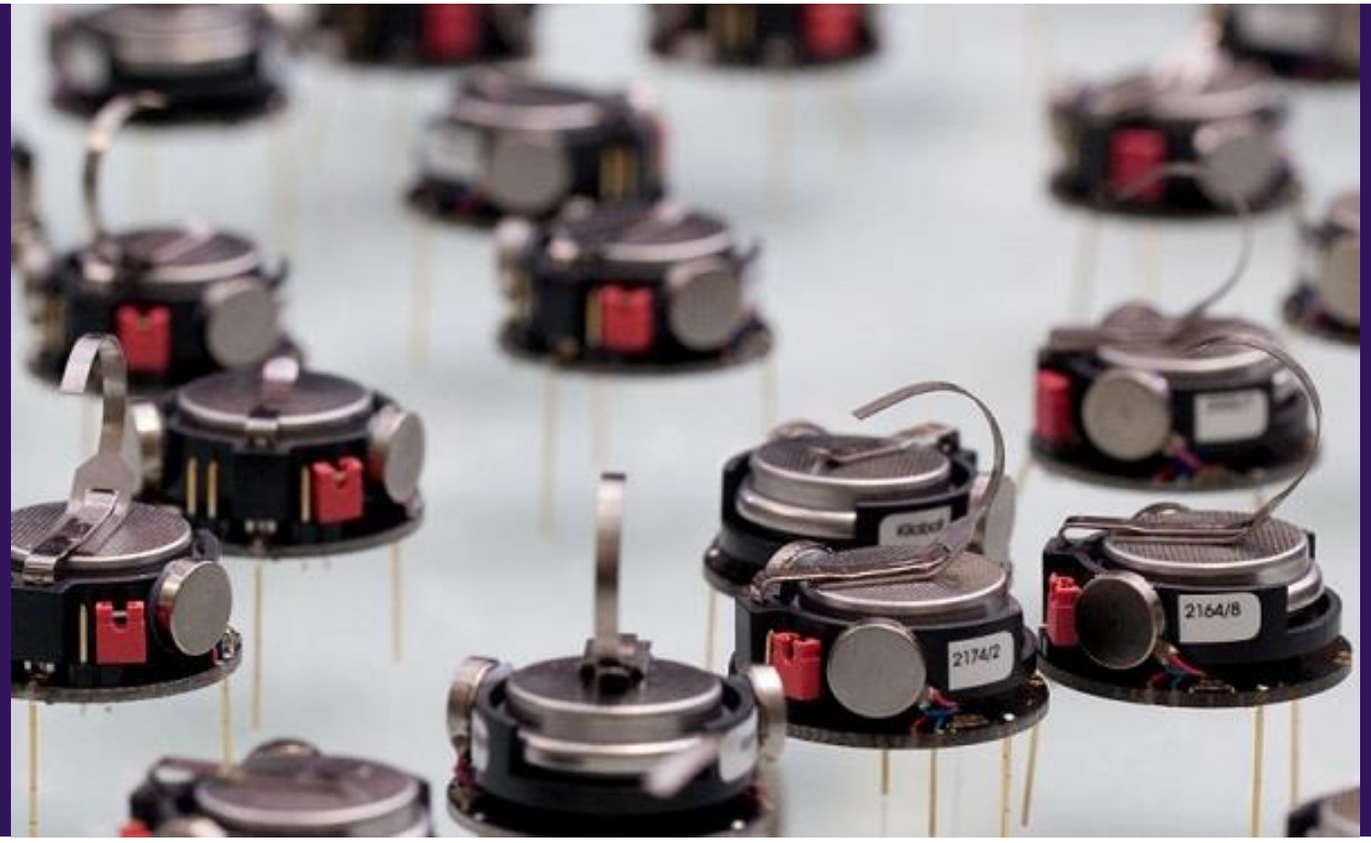
Transitioning to server-less

Available when there is work to do

Without you having to manage or control it

Functions as a Service (FaaS)

Small pieces of self-contained server-side logic



Event-driven

Responds to external triggers

Instant scaling

Abstraction of server infrastructure

Scales when needed

Pay by consumption

Charged by GB-s and # of executions

Server-less platform providers

Major cloud provider offer FaaS

New competitors enter competition

Amazon Lambda
(since 2014)

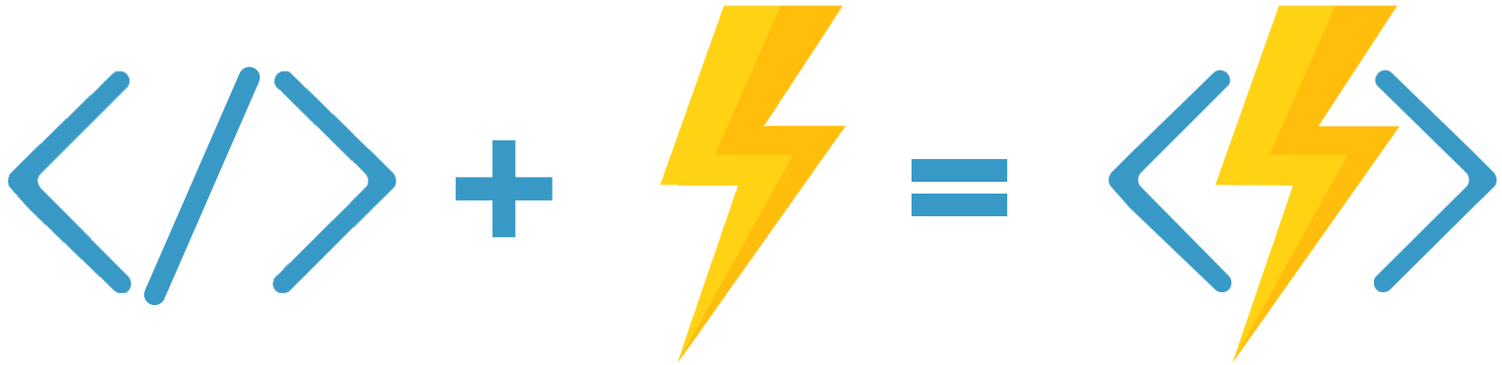
Google Cloud Functions
(since 2016)

Azure Functions
(since 2016)

Auth0 Webtask.IO
(since 2016)

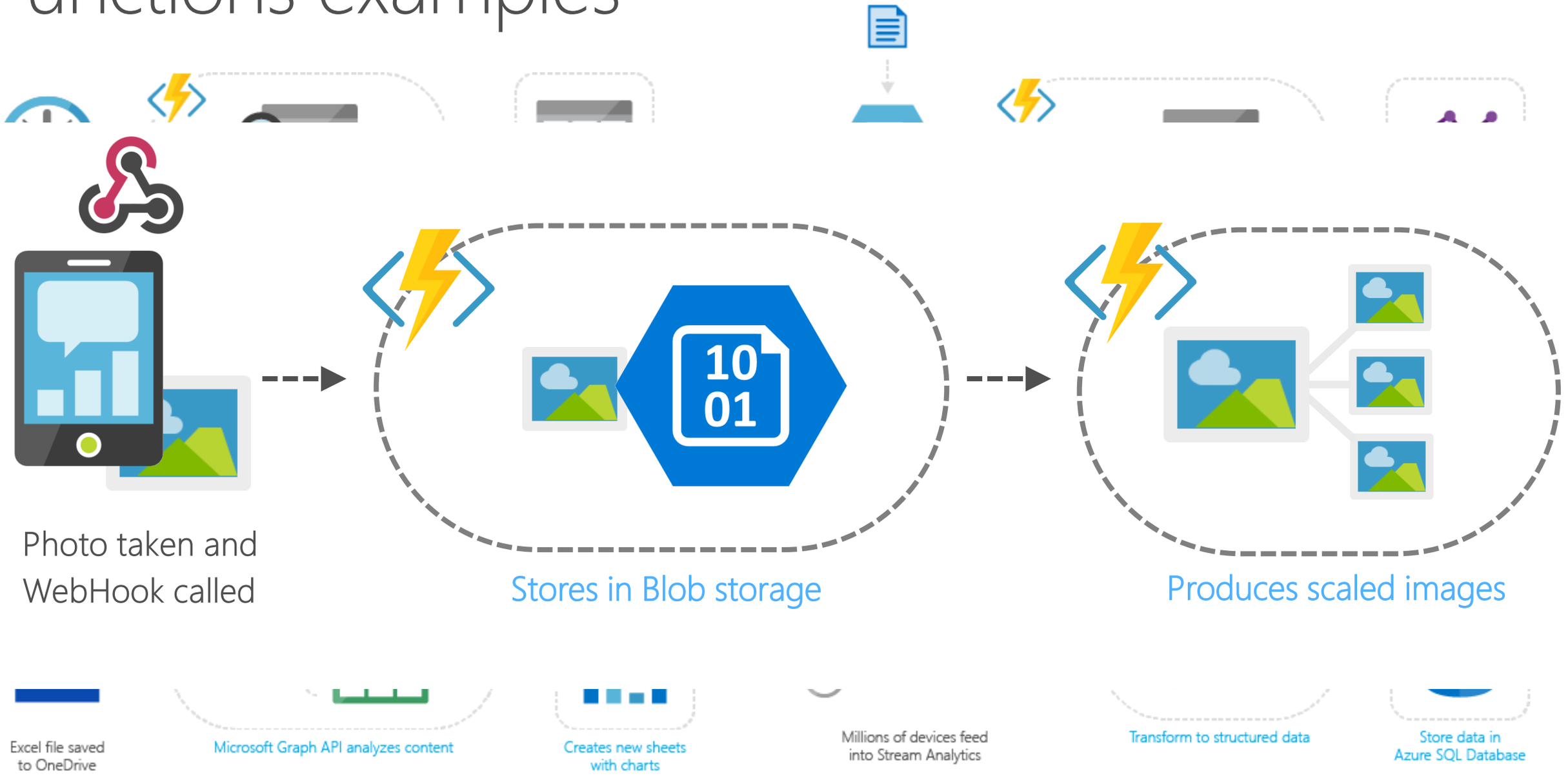
Focusing on Azure Functions

Code + Events + data = Azure Functions



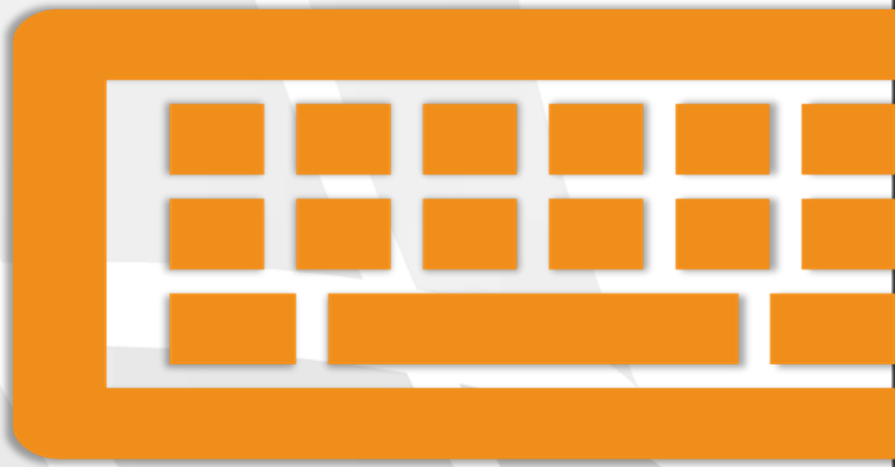
Process events with server-less code

Functions examples

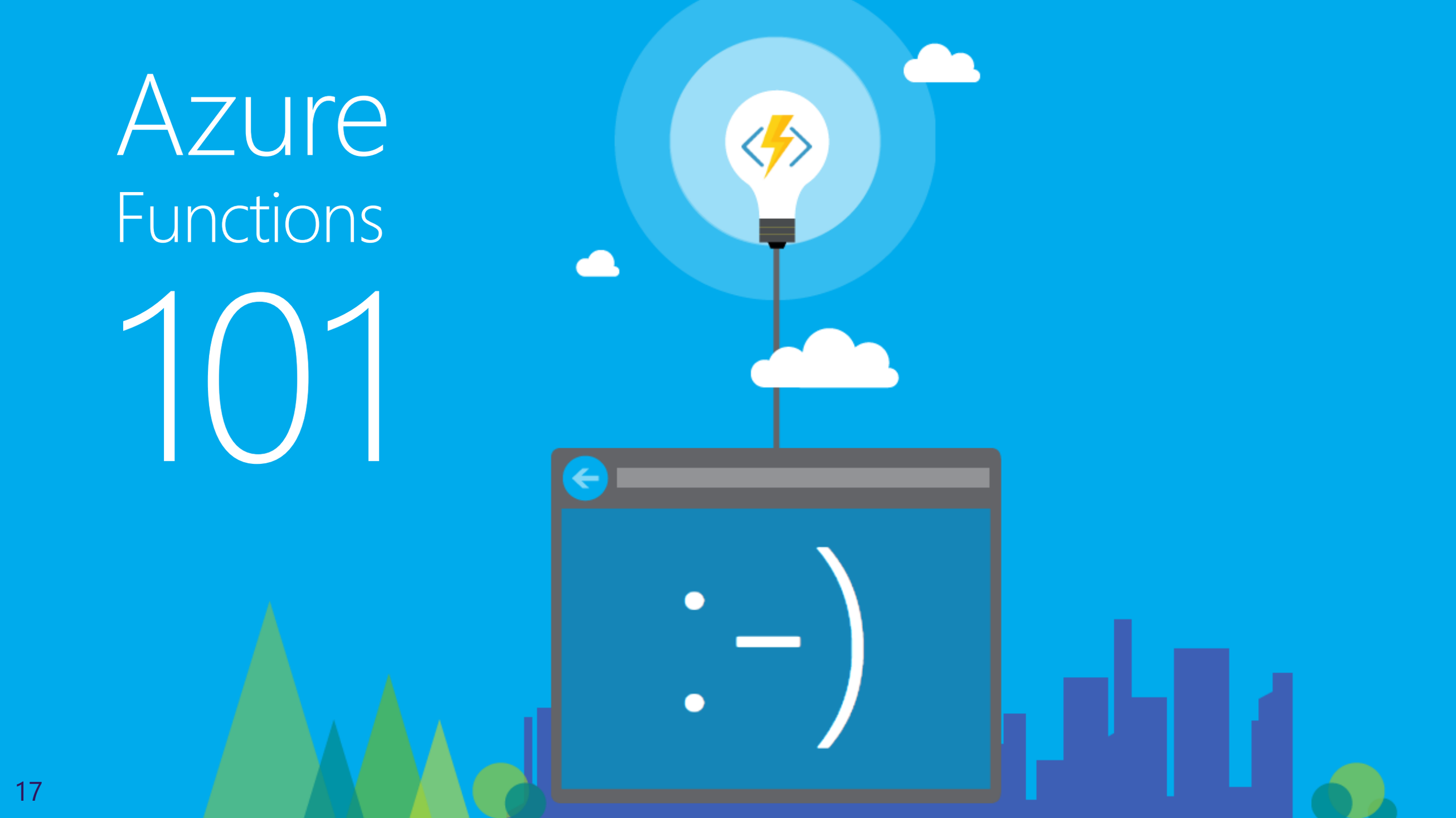


Lab 1 – Azure Functions 101

Lab



Azure Functions 101



Anatomy of an Azure Function App

Function apps

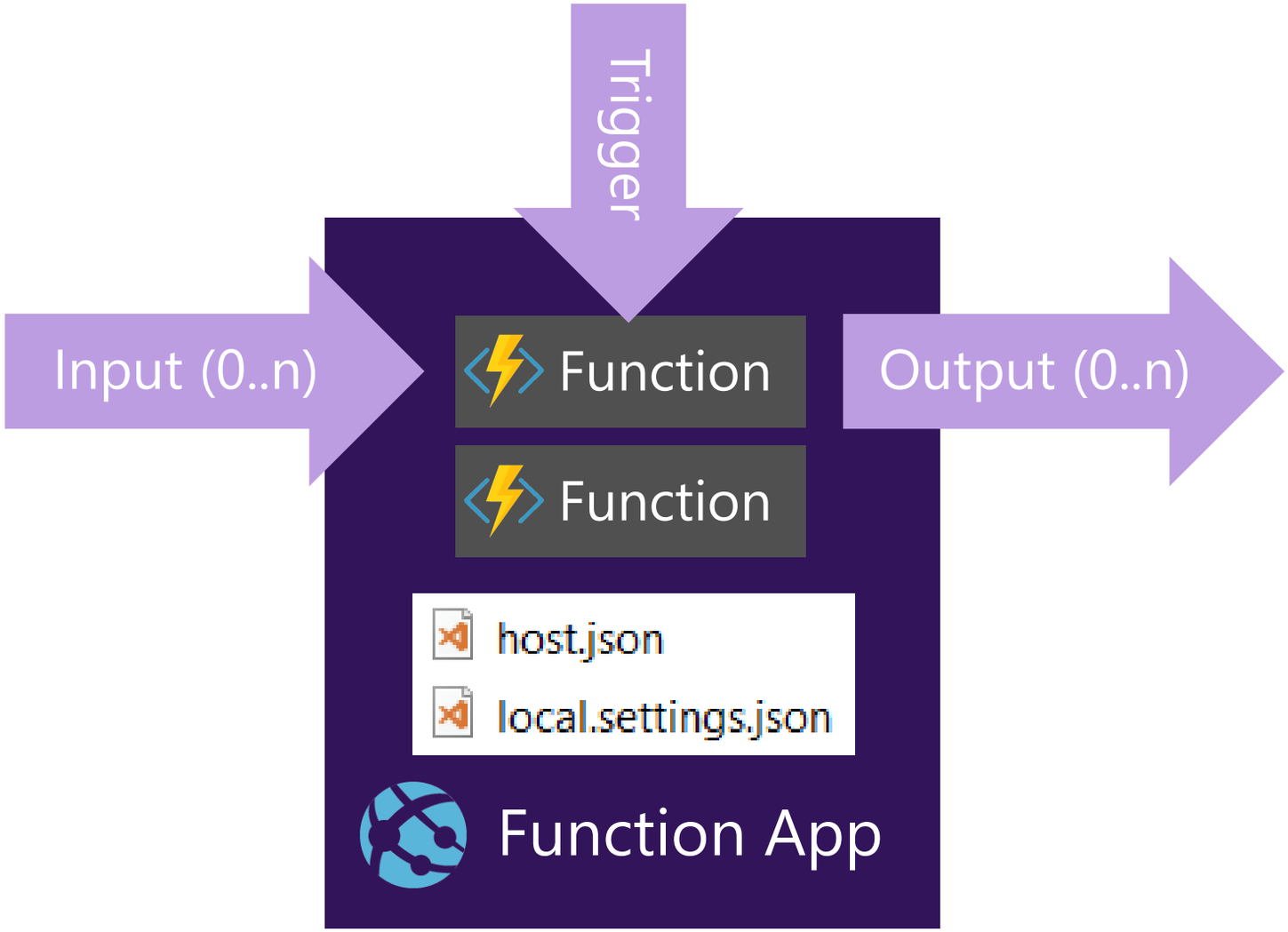
- Hosted as Azure App Service
- JSON based configuration
- Running (multiple) functions

Trigger starts execution

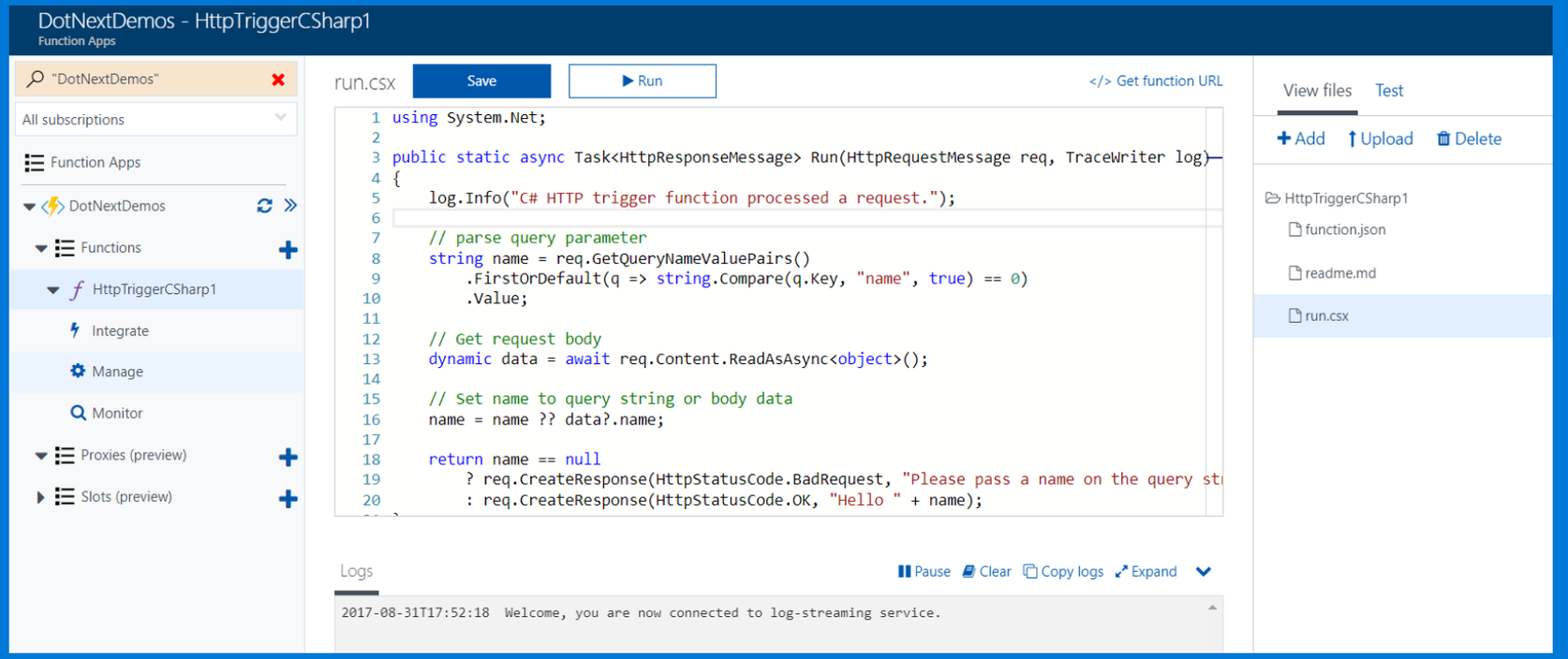
Bindings for input and output

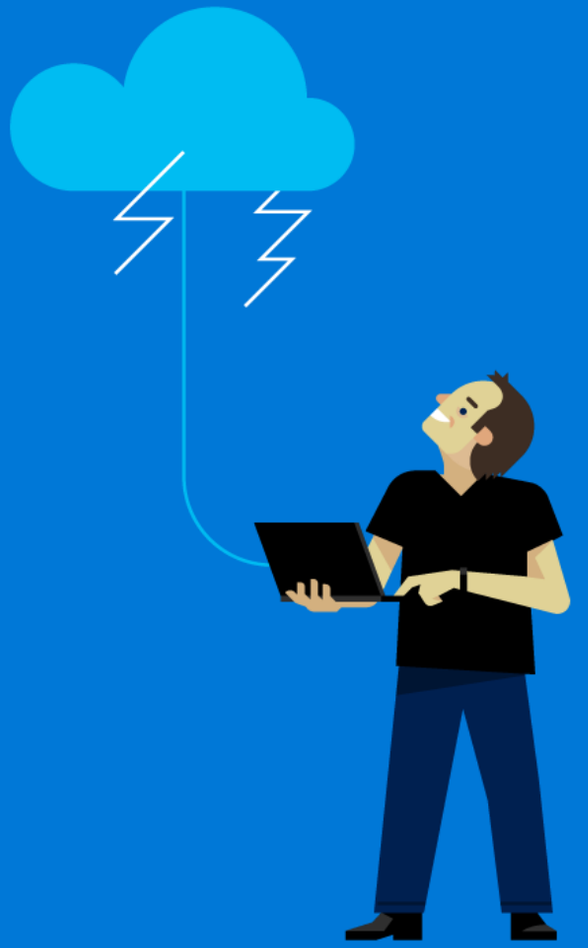
Zero or more possible

Triggers and bindings can vary per function

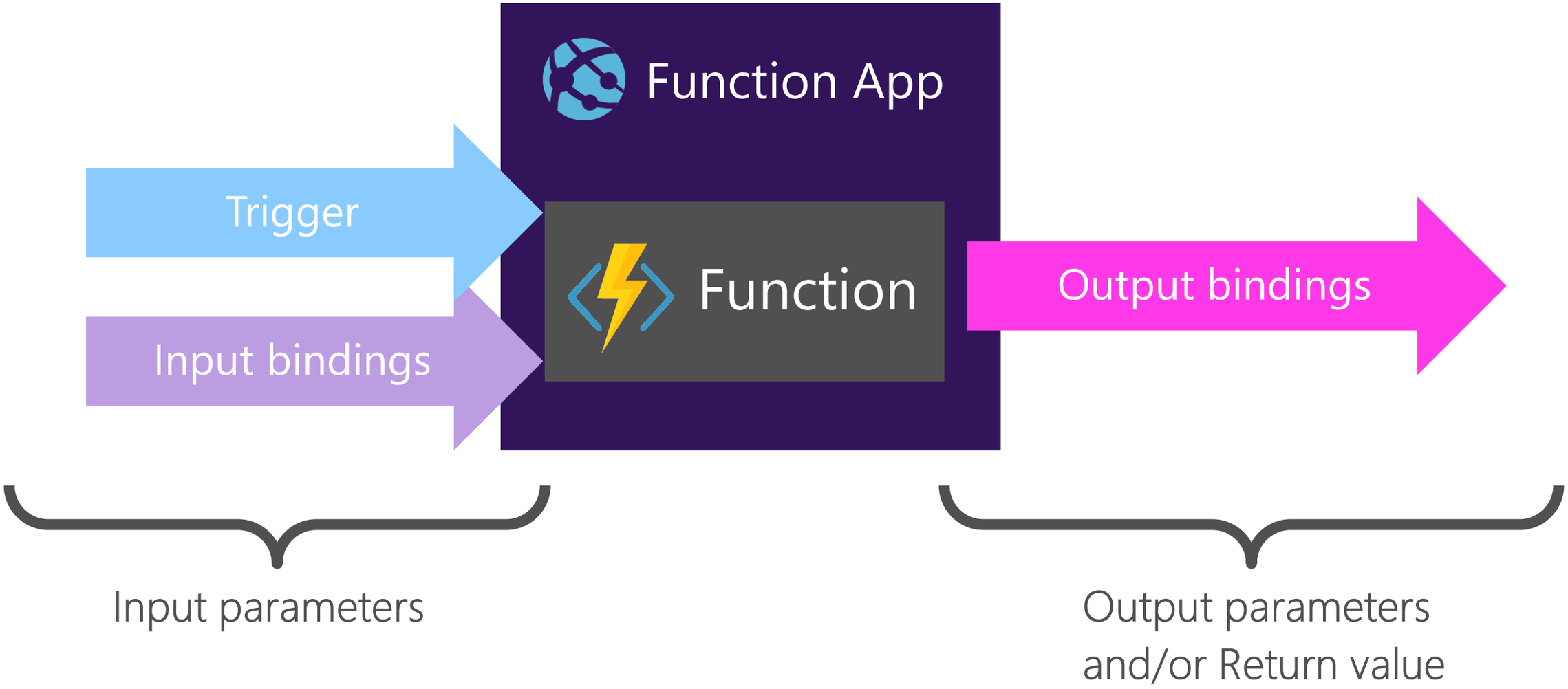


Demo
Creating and using Azure Functions

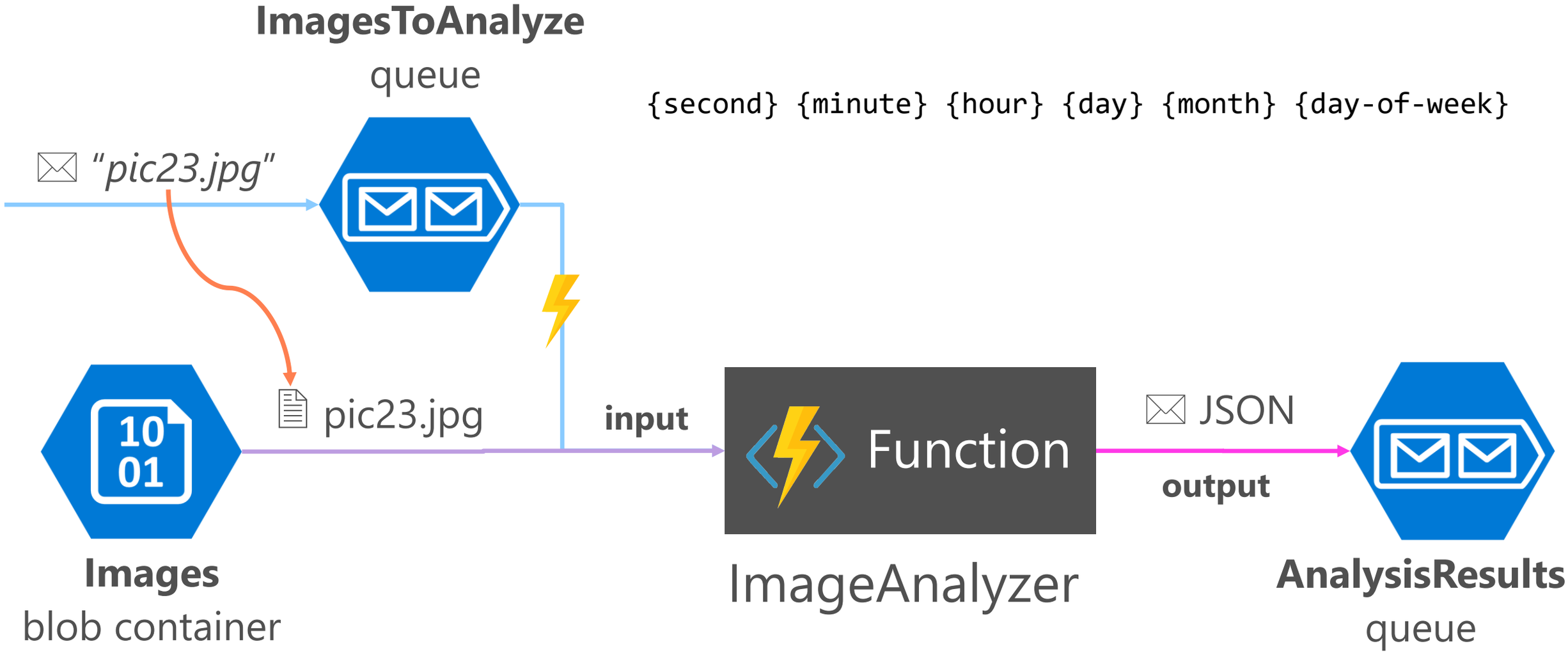




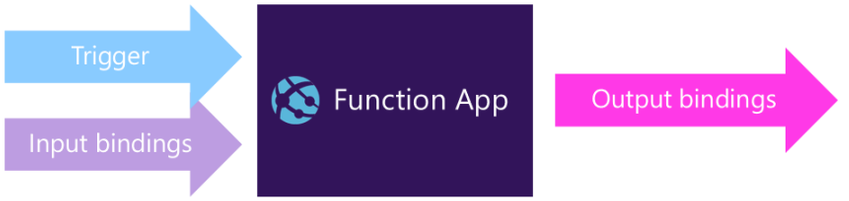
Programming model: It's a function



Triggers, input and output scenario



Abstractions over bindings



Define function.json metadata with attributes

```
[FunctionName("ImageAnalyzer")]
[return: Queue("AnalysisResults", Connection="…")]
public static AnalysisScore Run(
 [QueueTrigger("ImagesToAnalyze", Connection="…")] string imageName,
 [Blob("images/{queueTrigger}",
  FileAccess.Read,
  Connection = "…")] Stream blob, TraceWriter log) {
  return ...;
}
```

Return value
Single output binding can be done with return type

Trigger source
Connection property refers to value from application settings

Additional inputs (and outputs)
Attribute values can refer to trigger metadata with { }

Moving into production

Hosting options

Azure

Consumption-based

Scaling as needed

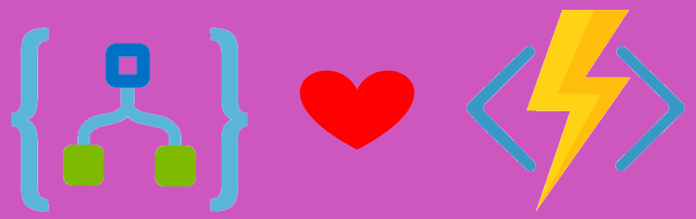
Might require time to provide compute instances

App Service plan

Available scale

Not so much server-less

Easy to combine with PaaS



On-premises

Azure Functions Runtime

Local installation of hosts

Connected via SQL Server database

Infrastructure operations

Not so much server-less

Durable Functions

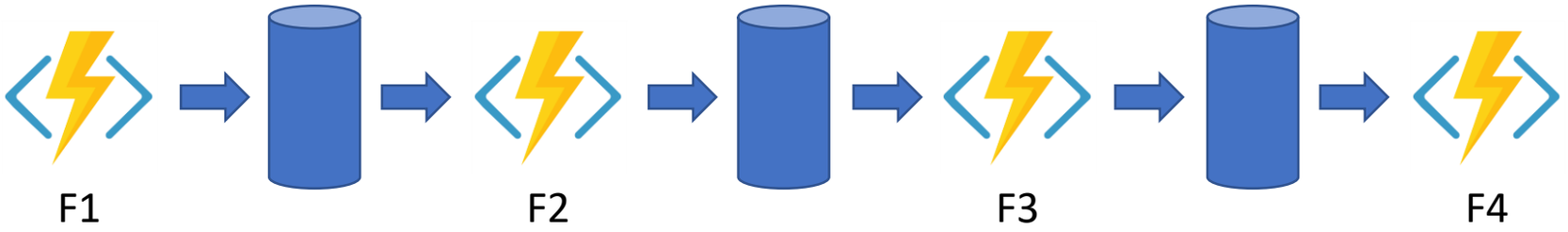
Why do we need “Durable Functions”

- Enable “long running” functions while maintaining local state.
- Simplify *complex* Function coordination (chaining, etc.)
- Easily call a Function from another Function
- All of the above using code-only

What is Durable Functions?

- Advanced feature for writing long-running orchestrations as a single C# function. No JSON schemas. No designer.
- New **orchestrator functions** can synchronously or asynchronously call other functions.
- Automatic **checkpointing**, enabling “long running” functions.
- Solves a variety of complex, transactional coding problems in serverless apps.
- Built on the open source Durable Task Framework.

Pattern #1: Function chaining - Today



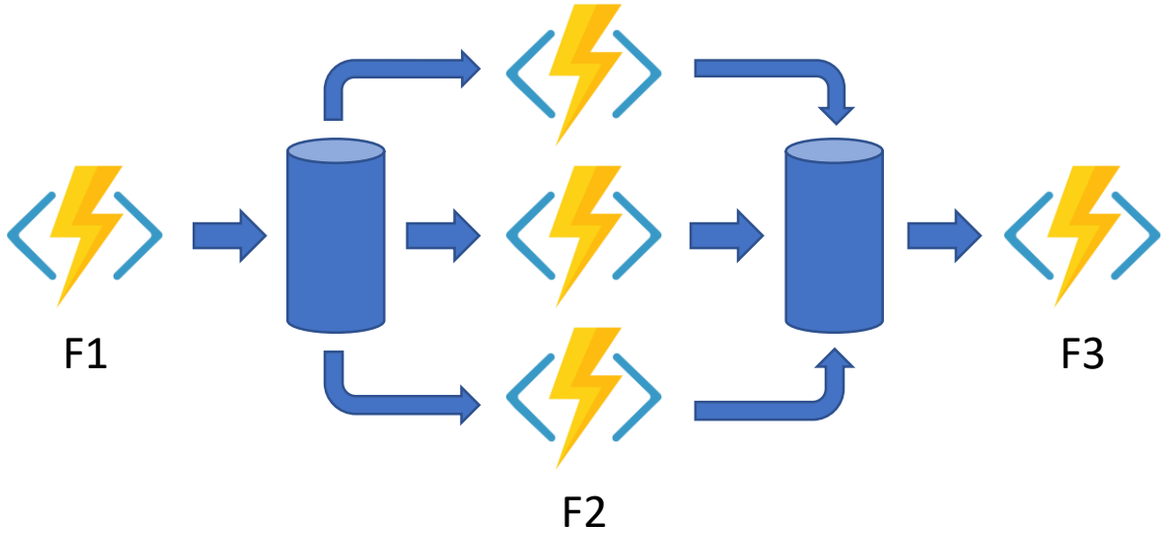
Problems:

- No visualization to show relationship between functions and queues.
- Middle queues are an implementation detail – conceptual overhead.
- Error handling adds a lot more complexity.

Pattern #1: Function chaining - Better

```
// calls functions in sequence
public static async Task<object> Run(DurableOrchestrationContext ctx)
{
    try
    {
        var x = await ctx.CallFunctionAsync("F1");
        var y = await ctx.CallFunctionAsync("F2", x);
        var z = await ctx.CallFunctionAsync("F3", y);
        return  await ctx.CallFunctionAsync("F4", z);
    }
    catch (Exception)
    {
        // global error handling/compensation goes here
    }
}
```

Pattern #2: Fan-out/Fan-in - Today



Problems:

- Fanning-out is easy, but fanning-in is significantly more complicated
- Functions offers no help with this scenario today
- All the same problems of the previous pattern

Pattern #2: Fan-out/Fan-in - Easy

```
public static async Task Run(DurableOrchestrationContext ctx)
{
    var parallelTasks = new List<Task<int>>();

    // get a list of N work items to process in parallel
    object[] workBatch = await ctx.CallFunctionAsync<object[]>("F1");
    for (int i = 0; i < workBatch.Length; i++)
    {
        Task<int> task = ctx.CallFunctionAsync<int>("F2", workBatch[i]);
        parallelTasks.Add(task);
    }

    await Task.WhenAll(parallelTasks);

    // aggregate all N outputs and send result to F3
    int sum = parallelTasks.Sum(t => t.Result);
    await ctx.CallFunctionAsync("F3", sum);
}
```

SERVERLESS360

- Technologies on Microsoft Azure for Serverless (INTEGRATION Focused) are on Silos
- The power comes once they are put together to solve a *REAL* business problems
- But, there is a challenge in Managing & Monitoring of Serverless Integration Applications

Serverless360 is a platform tool to manage and monitor Serverless applications effortlessly


Event Grid


Functions


Logic Apps


API Management


Queues


Topics


Relay


Event Hub

Dashboard

SERVERLESS360 | Composite Applications

License Plan: SB360_Partner_Gold | mohan.nagara...

Composite Applications

Refresh in 13 seconds

01 ServiceBus360 PROD

Queues	0
Topics	6
Logic Apps	4
Function Apps	0
Relays	0
Event Hubs	0
APIs	1

Alarm: 3

MANAGE

02 Document360 PROD

Queues	1
Topics	0
Logic Apps	0
Function Apps	0
Relays	0
Event Hubs	0
APIs	0

Alarm: 1

MANAGE

03 Order Processing

Queues	3
Topics	2
Logic Apps	3
Function Apps	4
Relays	0
Event Hubs	1
APIs	0

Alarm: 2

MANAGE

04 Invoice Processing

Queues	2
Topics	1
Logic Apps	2
Function Apps	4
Relays	2
Event Hubs	2
APIs	0

Alarm: 3

MANAGE

05 ECommerce

Queues	2
Topics	1
Logic Apps	1
Function Apps	1
Relays	0
Event Hubs	1
APIs	0

Alarm: 1

MANAGE

Specialty Outlet Sydney

Queues	1
Topics	4
Logic Apps	3
Function Apps	4
Relays	0
Event Hubs	1
APIs	0

Alarm: 5

MANAGE

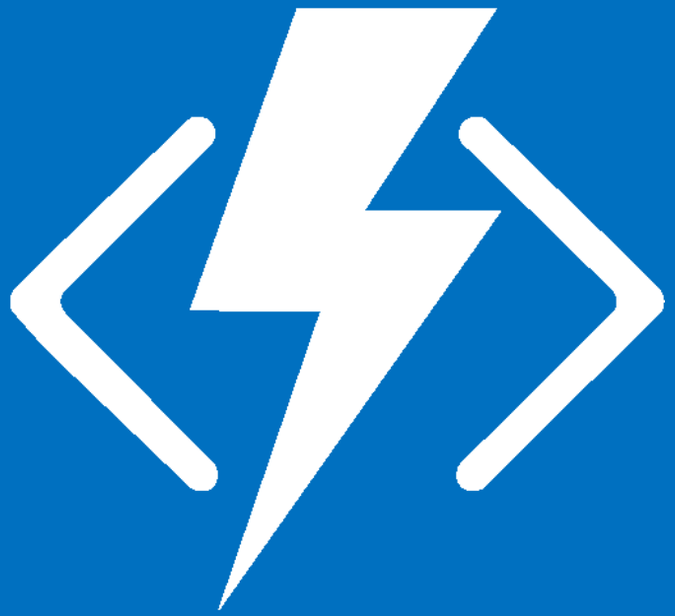
Retail Outlet Redmond

Queues	1
Topics	4
Logic Apps	4
Function Apps	0
Relays	0
Event Hubs	0
APIs	1

Alarm: 1

MANAGE

SERVERLESS360



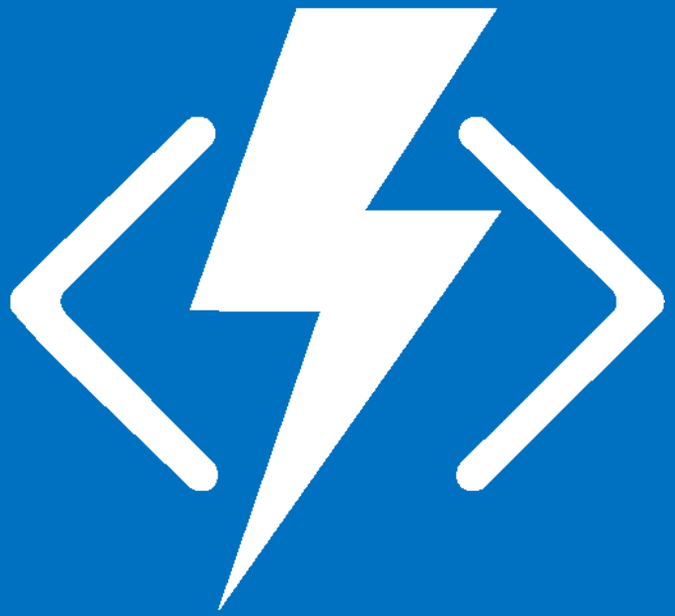
Management Capabilities in Serverless360

- Manage Azure Function apps from **various subscriptions at one place** in Serverless360.

- List of actions that you can perform on Azure Function apps,

    - Start
    - Stop
    - Restart
    - Delete
    - View properties of the Functions with in Function apps
    - Access Invocation Logs of the Functions by associating the Azure Function app to a Serverless360 Composite Application

SERVERLESS360



Monitoring Capabilities in Serverless360

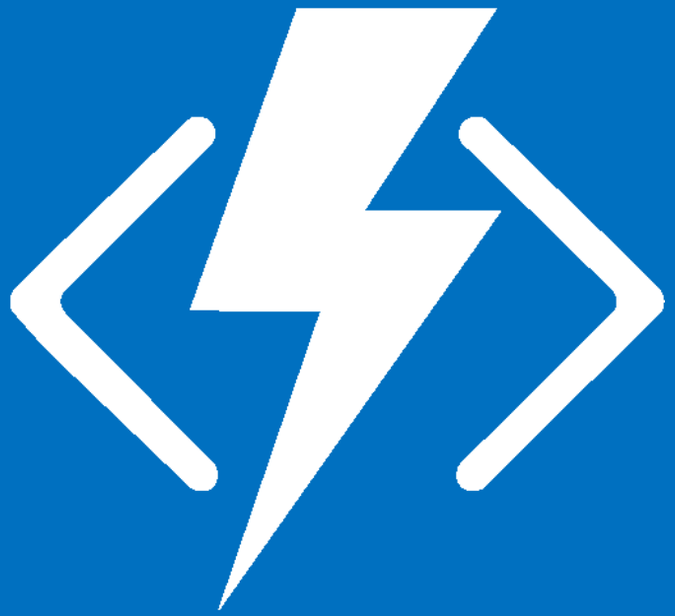
By associating Azure Function Apps to Serverless360 Composite Application, you gain access to multiple monitoring options like;

1. State based monitoring:
Monitor your Azure Function app based on the current '**state**'. Choose appropriate alarm from the below list based on the business need;

- **Threshold alarm** – Alerts you when a violation persists against an expected configuration for a specified period of time
- **Health Check alarm**– Checks the status of entities against the expected configuration at defined time and notify you on the status

2. Watch:
Get notified on Azure Function failure in near real time**.**

SERVERLESS360



Monitoring Capabilities in Serverless360

3. Data Monitoring: Monitor Azure Function Apps based on extensive set of metrics listed

Data In (Bytes)
Data Out (Bytes)
Http Server Errors (Count)
Memory working set (Bytes)
Average memory working set (Bytes)
Function execution Units (Count)
Function execution Count (Count)
Connections (Count)
Handle Count (Count)
Thread Count (Count)
Private Bytes (Bytes)
IO Read Bytes Per Second (BytesPerSecond)
IO Write Bytes Per Second (BytesPerSecond)
IO Other Bytes Per Second (BytesPerSecond)
IO Read Operations Per Second (BytesPerSecond)
IO Write Operations Per Second (BytesPerSecond)
IO Other Operations Per Second (BytesPerSecond)
Requests In Application Queue (Count)
Current Assemblies (Count)
Total App Domains (Count)
Total App Domains Unloaded (Count)
Gen 0 Garbage Collections (Count)
Gen 1 Garbage Collections (Count)
Gen 2 Garbage Collections (Count)

Resources

Read

https://cloudandmobileblog.com/2018/03/15/how-to-trigger-azure-function-by-azure-cosmos-db/

https://aka.ms/tryfunctions
https://functions.azure.com
https://docs.microsoft.com/en-us/azure/azure-functions/
https://github.com/Azure/Azure-Functions

Contact @AzureFunctions

Watch Function Junction videos on Channel9