



Introduction to Azure Functions

Christopher MANEU
Startups Cloud Advocate @ Microsoft
@cmaneu

A quick intro to Azure & Serverless

Discover & create your 1st Azure Function

Advanced usages of Azure Functions

A quick intro to Azure & Serverless

Discover & create your 1st Azure Function

Advanced usages of Azure Functions

What is Azure?



Edge Devices

Azure Stack Hub Azure Stack Edge Azure Sphere Azure Kinect HoloLens



Serverless

Web	Databases
Mobile	Analytics
Mixed Reality	AI + Machine Learning
Containers	Internet of Things
Events + Integration	Media



Infrastructure

Compute Networking Storage Security Identity



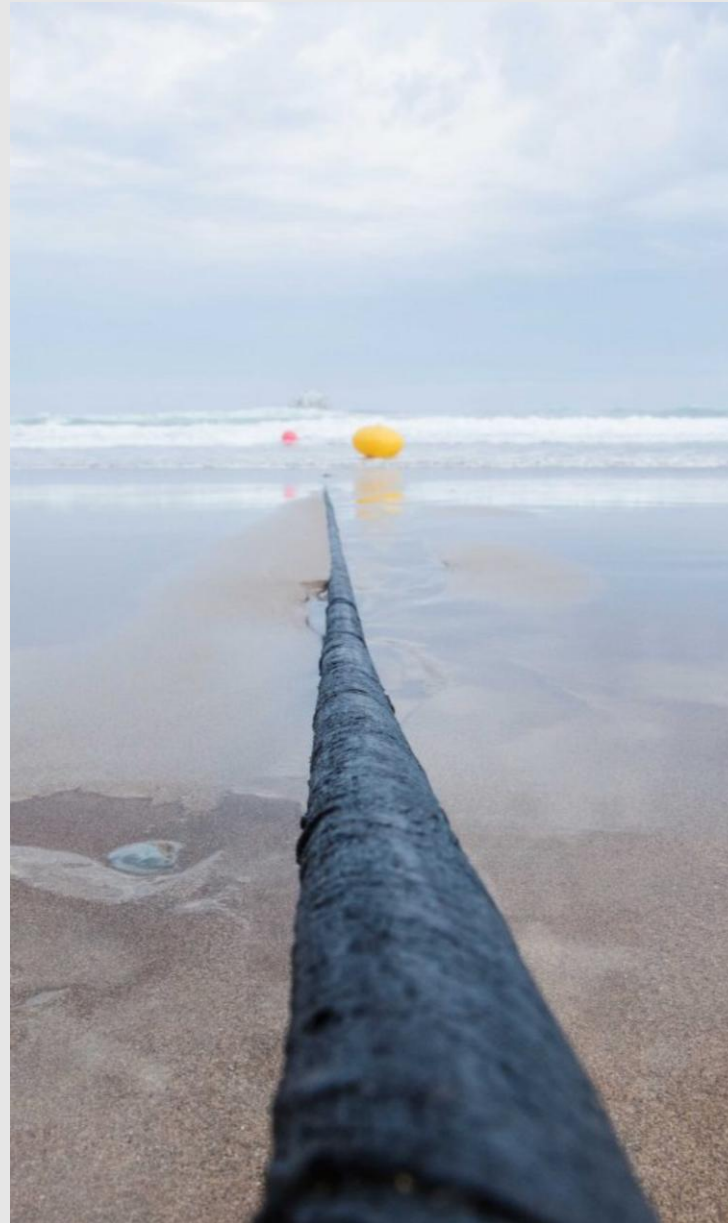
Tools

Visual Studio
GitHub
PowerApps
Power BI



58 Azure regions

Largest geographical footprint of any cloud provider with more than **58** Azure regions



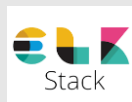
Marea sub-sea cable – 160Tbps – Virginia > Bilbao



Project Natic – Underwater datacenter

Azure works with what you use

~50% of Azure VMs
run Linux



Azure Hybrid

Innovation anywhere with Azure



Azure IoT

Any edge device



Azure Arc

Any datacenter, any cloud



Azure Stack

Integrated systems



Management

Security + Identity

App + Data Services

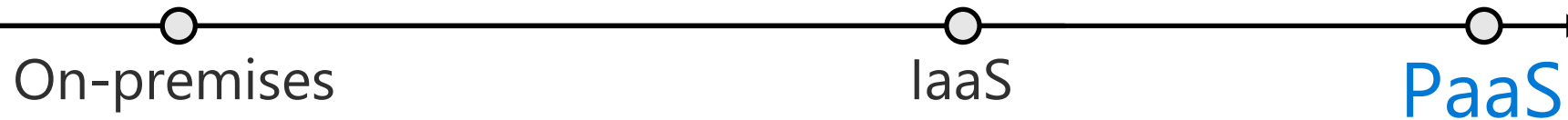
Dev Tools + DevOps

What is the right **size** of “**servers**” for my business needs?

How can I increase “**server**” utilization?

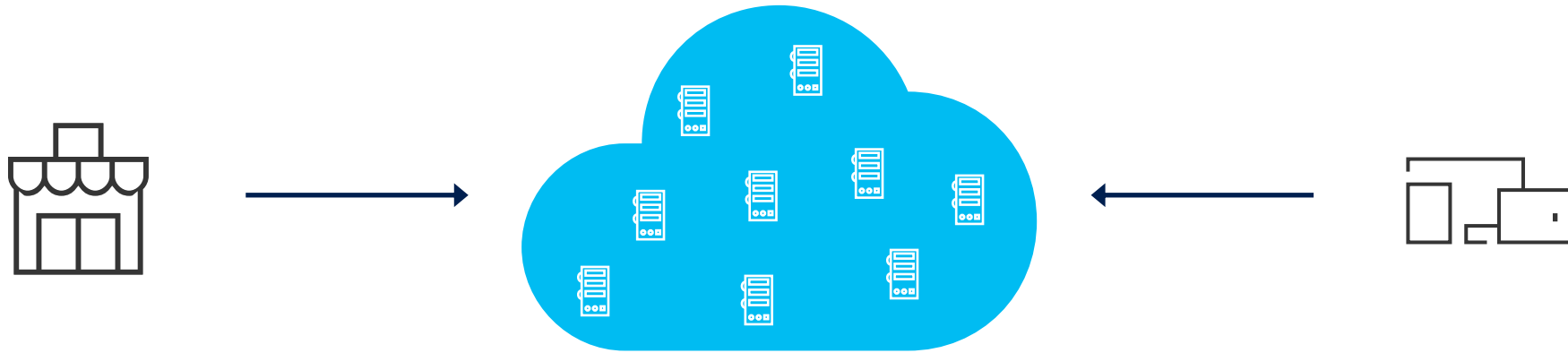
How many “**servers**” do I need?

How can I **scale** my app?

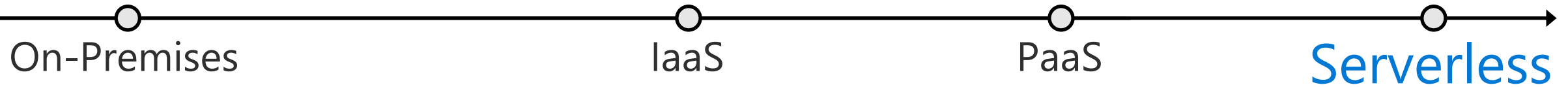


The “evolution” of application platforms

How do I **architect** my app?



Serverless, the platform for cloud native apps

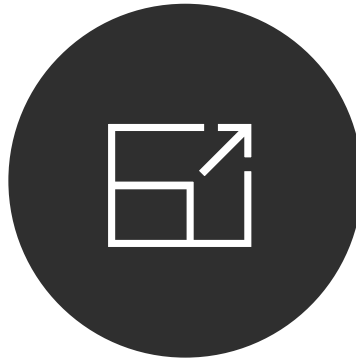


The “evolution” of application platforms

What is Serverless?



Abstraction
of servers

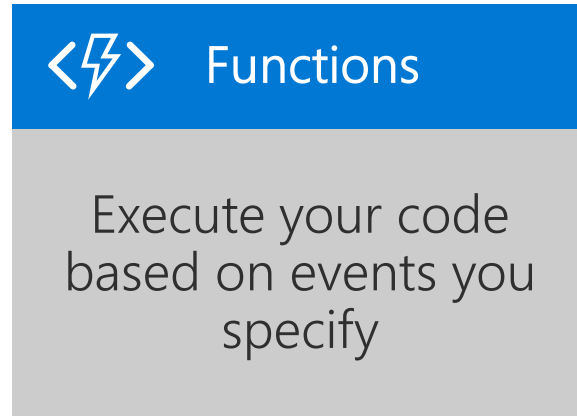


Event-driven/
instant scale



Micro-billing

Azure serverless platform components



Azure serverless platform components



Logic Apps

Design workflows
and orchestrate
processes



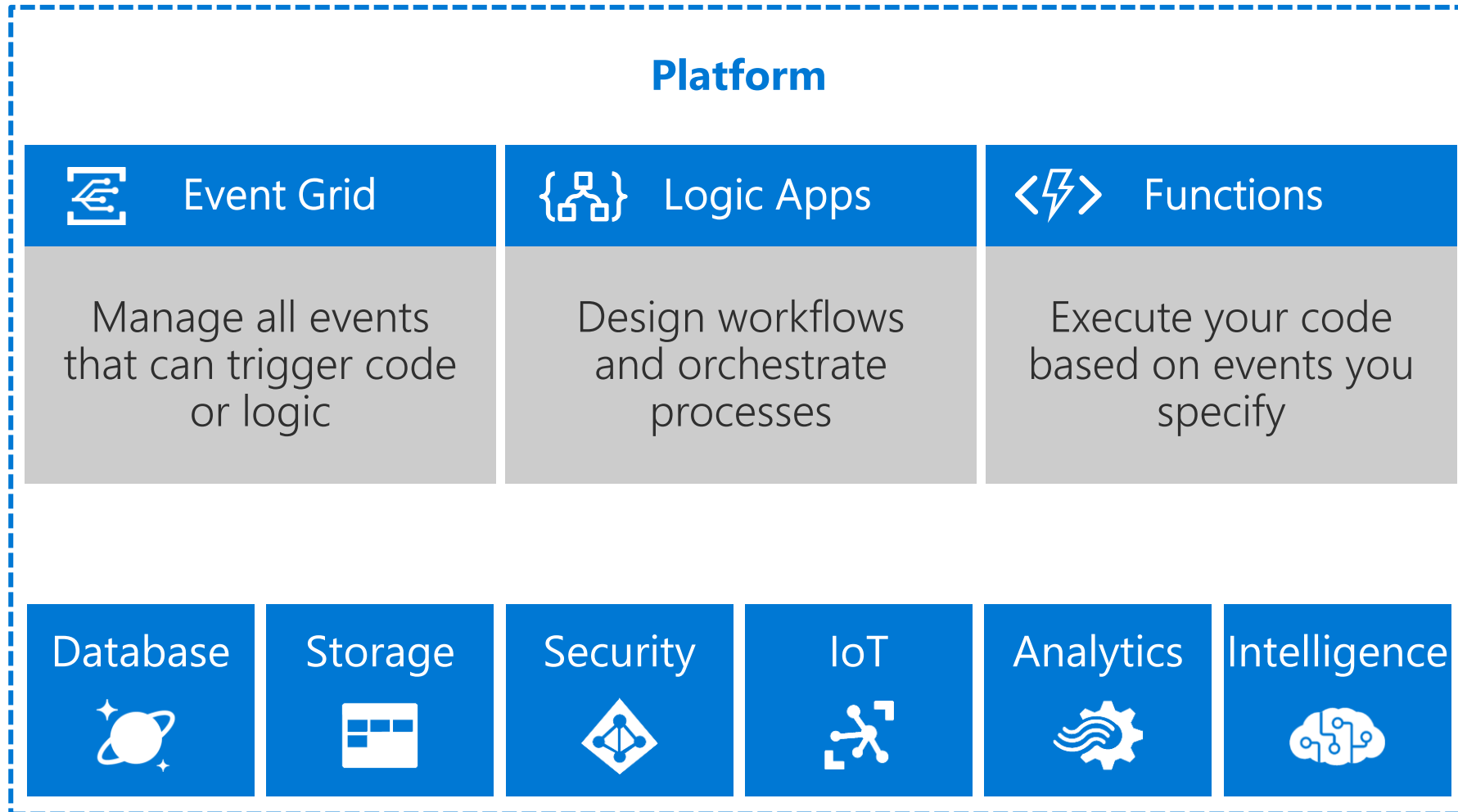
Functions

Execute your code
based on events you
specify

Azure serverless platform components



Azure serverless platform components



Azure serverless platform components

Development

 IDE support

 Integrated DevOps

 Local development

 Monitoring

 Visual debug history

Platform



Event Grid

Manage all events that can trigger code or logic



Logic Apps

Design workflows and orchestrate processes



Functions

Execute your code based on events you specify

Database



Storage



Security



IoT



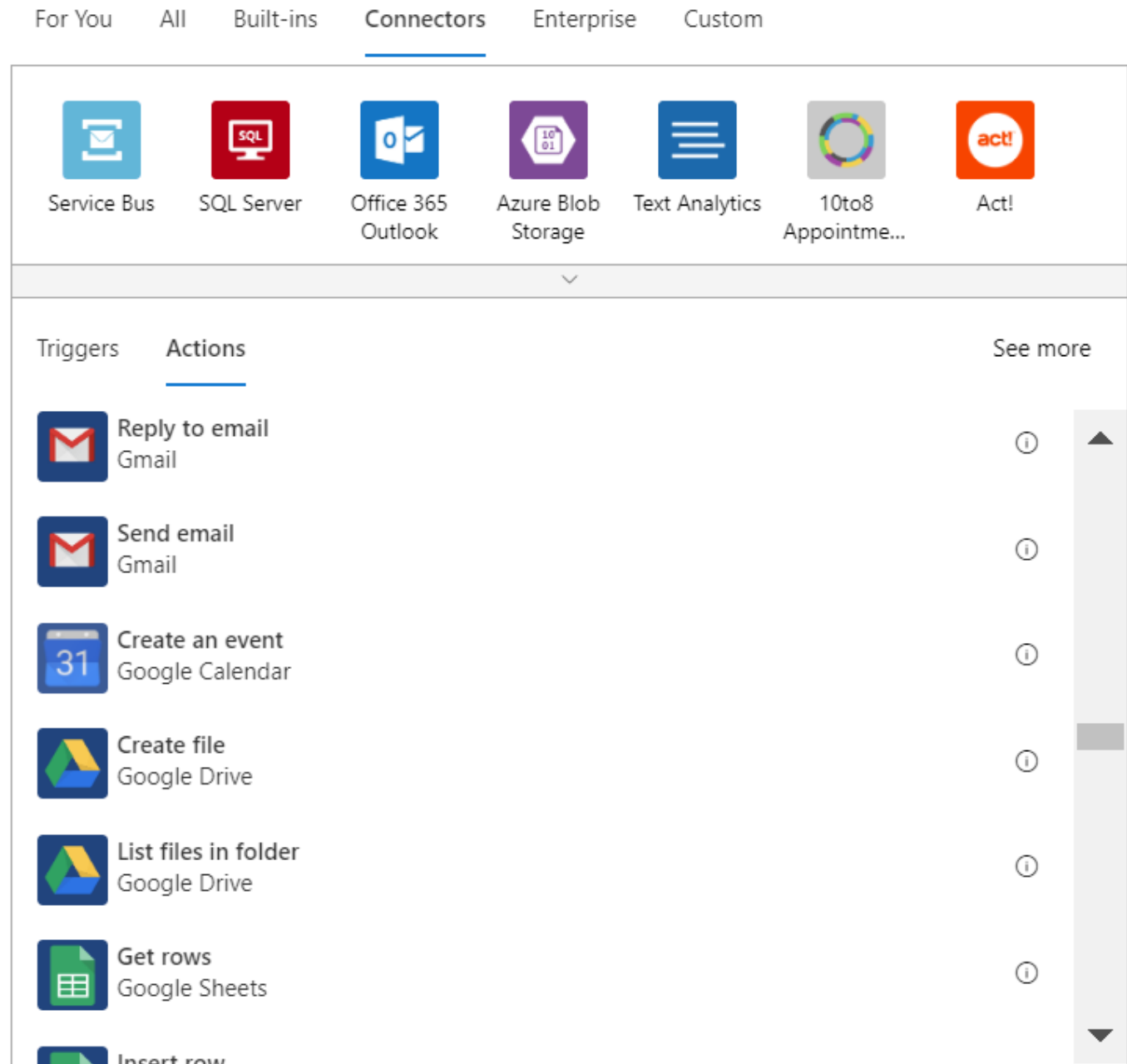
Analytics



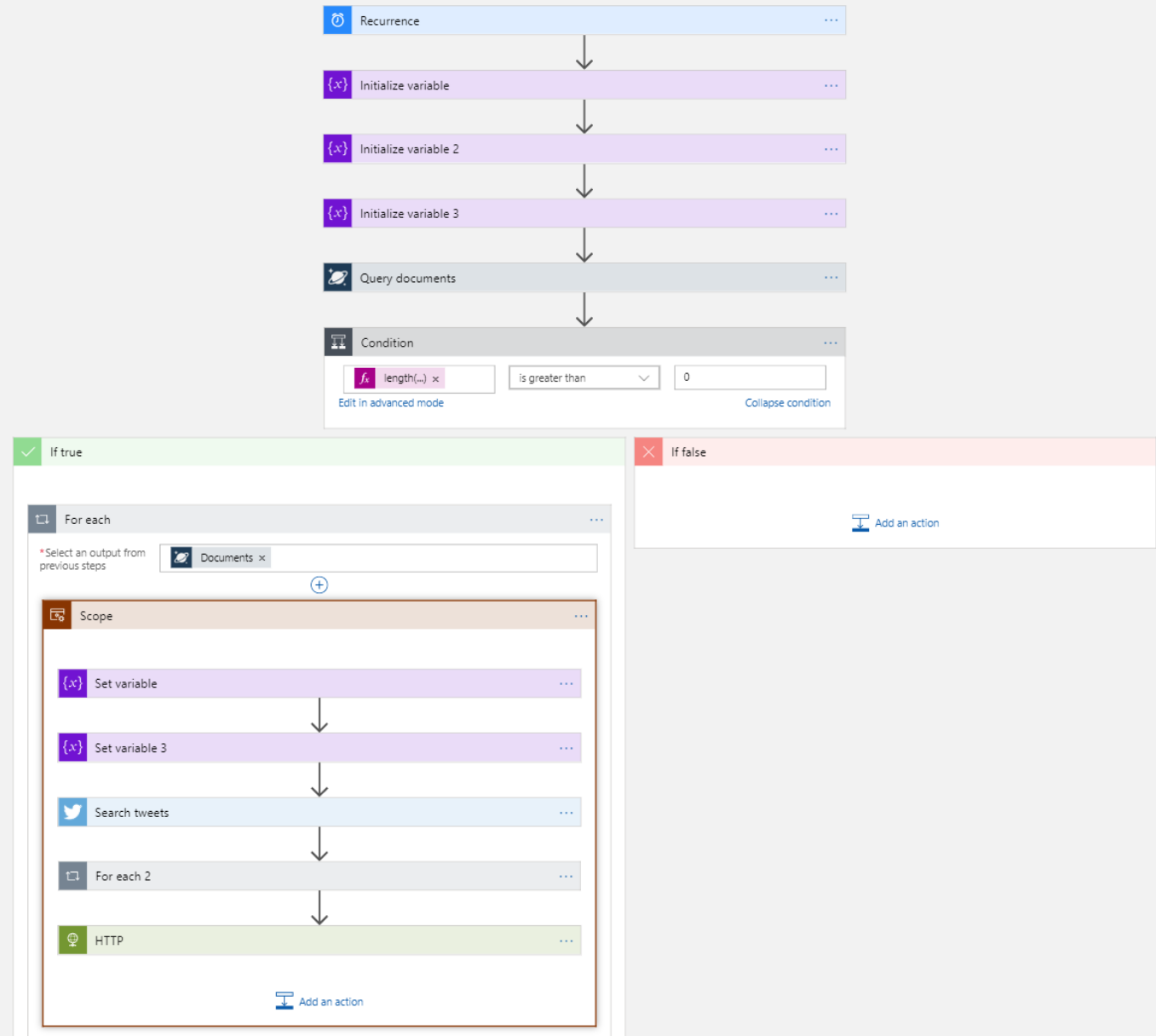
Intelligence



Hundreds of connectors for integrations



Visual workflow designer



A quick intro to Azure & Serverless

Discover & create your 1st Azure Function

Advanced usages of Azure Functions

Azure Functions

Events



React to timers, HTTP, or events from your favorite Azure services, with more on the way

Code



Author functions in C#, F#, Node.JS, **Java**, and more

Outputs



Send results to an ever-growing collection of services

Functions triggers and bindings

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	✓		✓
Queues and topics	Azure Service Bus	✓		✓
Storage tables	Azure Storage		✓	✓
SQL tables	Azure Mobile Apps		✓	✓
NoSQL DB	Azure Cosmos DB	✓	✓	✓
Push Notifications	Azure Notification Hubs			✓
Twilio SMS Text	Twilio			✓
SendGrid email	SendGrid			✓
Excel tables	Microsoft Graph		✓	✓
OneDrive files	Microsoft Graph		✓	✓
Outlook email	Microsoft Graph			✓
Microsoft Graph events	Microsoft Graph	✓	✓	✓
Auth tokens	Microsoft Graph		✓	

A quick intro to Azure & Serverless

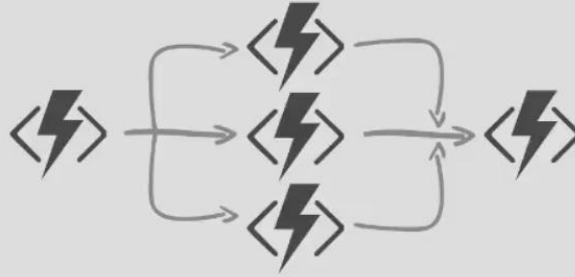
Discover & create your 1st Azure Function

Advanced usages of Azure Functions

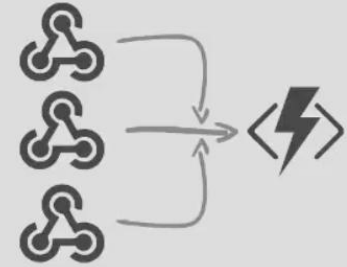
What's still hard?



Manageable Sequencing
+ Error Handling / Compensation



Fanning-out & Fanning-in



External Events Correlation



Flexible Automated Long-running
Process Monitoring



Http-based
Async Long-running APIs

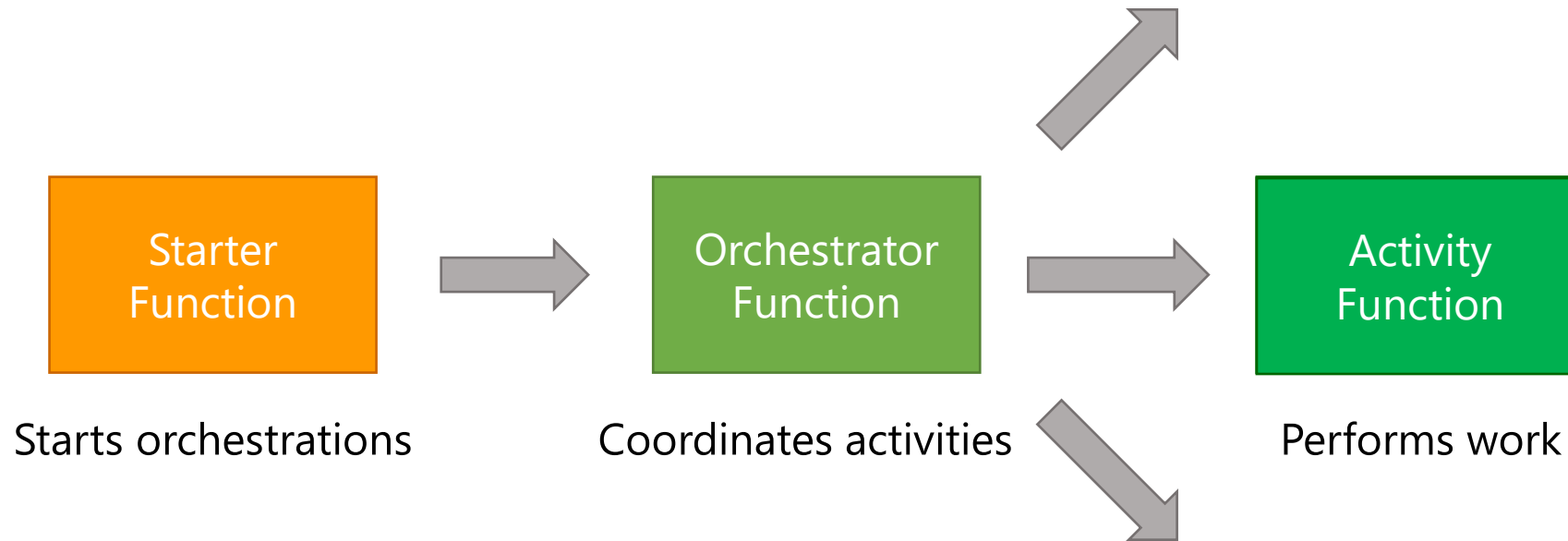


Human Interaction

Durable Functions

- Write **long-running orchestrations** as a **single function** while **maintaining local state**.
- **Simplify complex transactions and coordination (chaining, etc.)** Easily call a Function from another Function, synchronously or asynchronously.
- All of the above using code-only. No JSON schemas. No graphical designer.
- **C# and JavaScript** for now

Components



OrchestrationClient

OrchestrationContextTrigger

ActivityTrigger

What It Looks Like

// calls functions in sequence

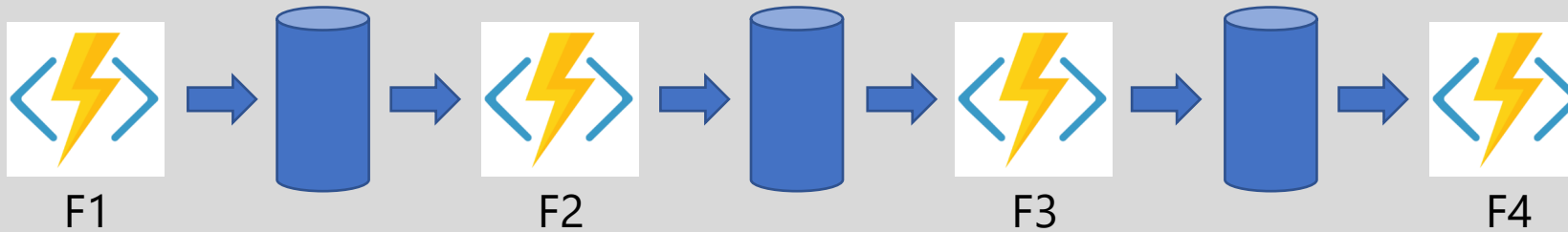
Orchestrator Function

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

Activity Functions

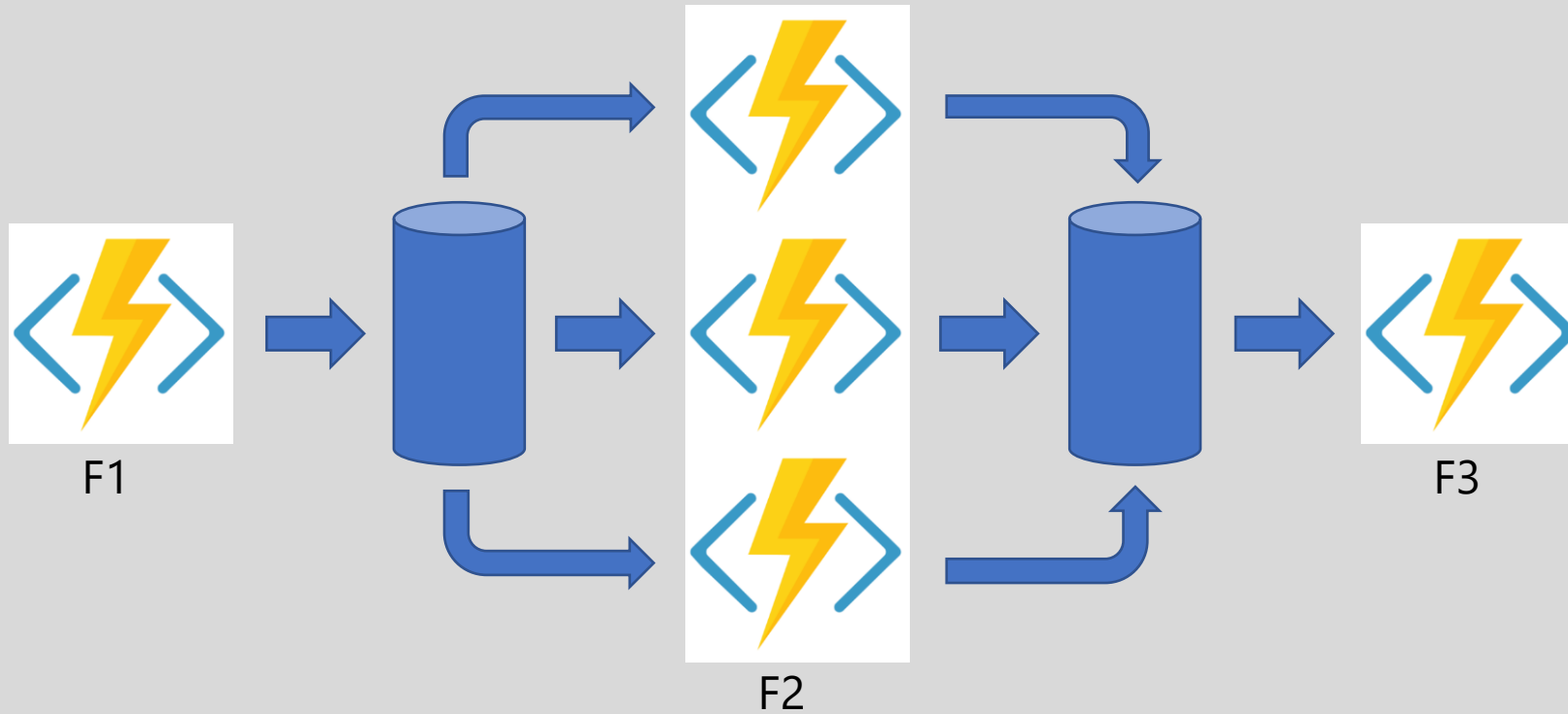
```
var x = await ctx.CallActivityAsync("F1");
var y = await ctx.CallActivityAsync("F2", x);
return await ctx.CallActivityAsync("F3", y);
```

Pattern #1: Function Chaining



```
// 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



```
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.CallActivityAsync<object[]>("F1");
    for (int i = 0; i < workBatch.Length; i++)
    {
        Task<int> task = ctx.CallActivityAsync<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.CallActivityAsync("F3", sum);
}
```

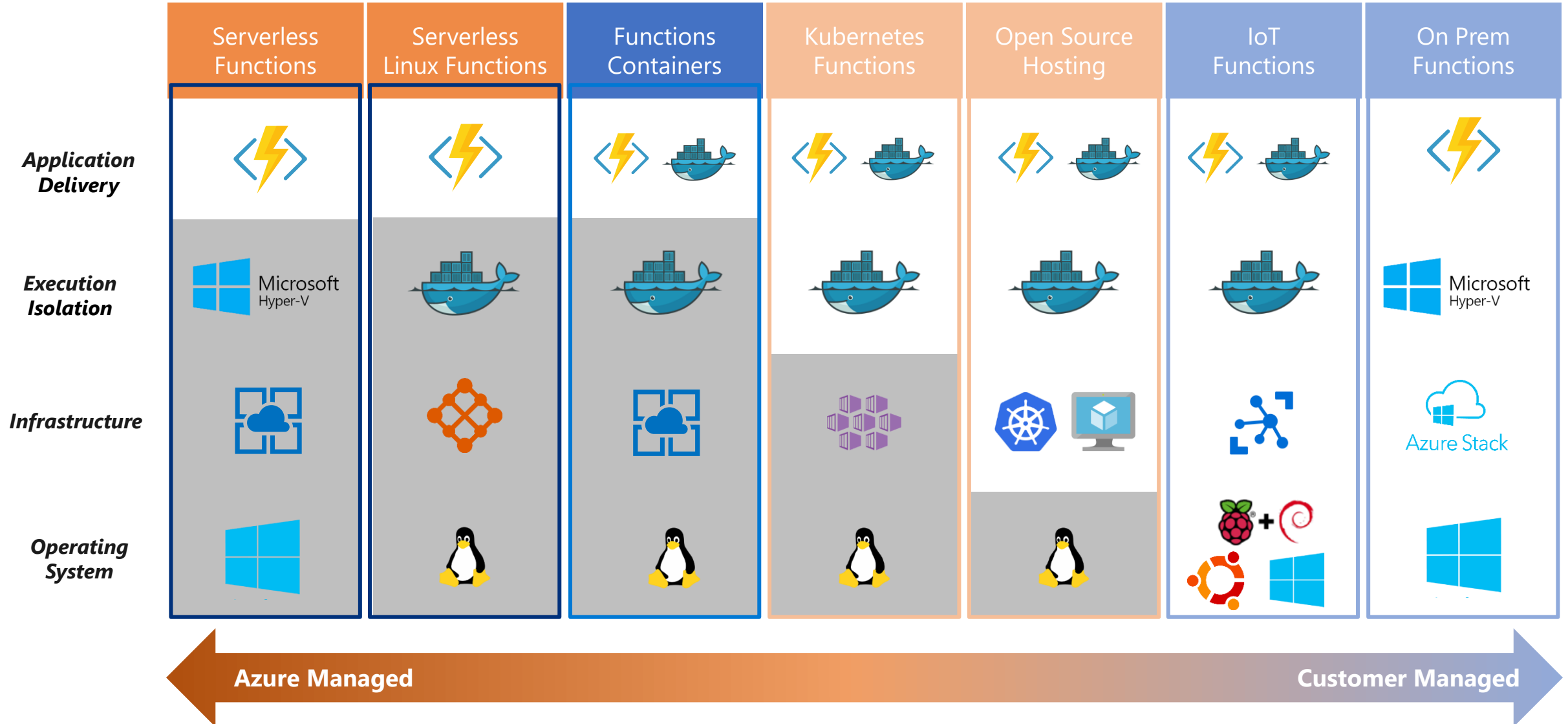
Pattern #3: Human Interaction w/Timeout



```
public static async Task Run(DurableOrchestrationContext ctx)
{
    await ctx.CallActivityAsync<object[]>("RequestApproval");
    using (var timeoutCts = new CancellationTokenSource())
    {
        DateTime dueTime = ctx.CurrentUtcDateTime.AddHours(72);
        Task durableTimeout = ctx.CreateTimer(dueTime, 0, cts.Token);

        Task<bool> approvalEvent = ctx.WaitForExternalEvent<bool>("ApprovalEvent");
        if (approvalEvent == await Task.WhenAny(approvalEvent, durableTimeout))
        {
            timeoutCts.Cancel();
            await ctx.CallActivityAsync("HandleApproval", approvalEvent.Result);
        }
        else
        {
            await ctx.CallActivityAsync("Escalate");
        }
    }
}
```

Functions Hosting Models





Kubernetes event driven autoscaling

- Monitors the rate of events to ***proactively*** scale any container
- Feeds the data into the Kubernetes system to drive scaling
- Allows containers to scale to and from zero
- Open source (MIT License) and onboarding to CNCF
- Extensible and pluggable scalers to grab metrics from any event source
 - Azure Service Bus
 - Azure Event Hubs
 - Kafka
 - RabbitMQ
 - AWS Simple Queue Service
 - GCP PubSub
 - More...

Azure Functions is an Open Source project

<https://github.com/Azure/Azure-Functions>

- [Azure Functions Host](#) - the Azure Functions runtime
- [Azure WebJobs SDK](#) - the "core" of the Azure Functions runtime and many bindings
- [Azure WebJobs SDK extensions](#) - the repositories of many bindings
- [Azure Functions Core Tools](#) - the command line tool for Azure Functions
- [Azure Functions Portal](#) - the UX for the Functions development portal
- [Azure Functions templates](#) - the templates which show up the Azure Functions portal
- [Azure Functions samples](#) - repository for some samples on how the runtime works

Image credits



Q&A

Christopher MANEU
Cloud Advocate @ Microsoft
@cmaneu



Thanks !

Christopher MANEU
Cloud Advocate @ Microsoft
@cmaneu