

Beyond Hello World:



Getting Deeper into **Azure Functions**

Who is Chad Green?



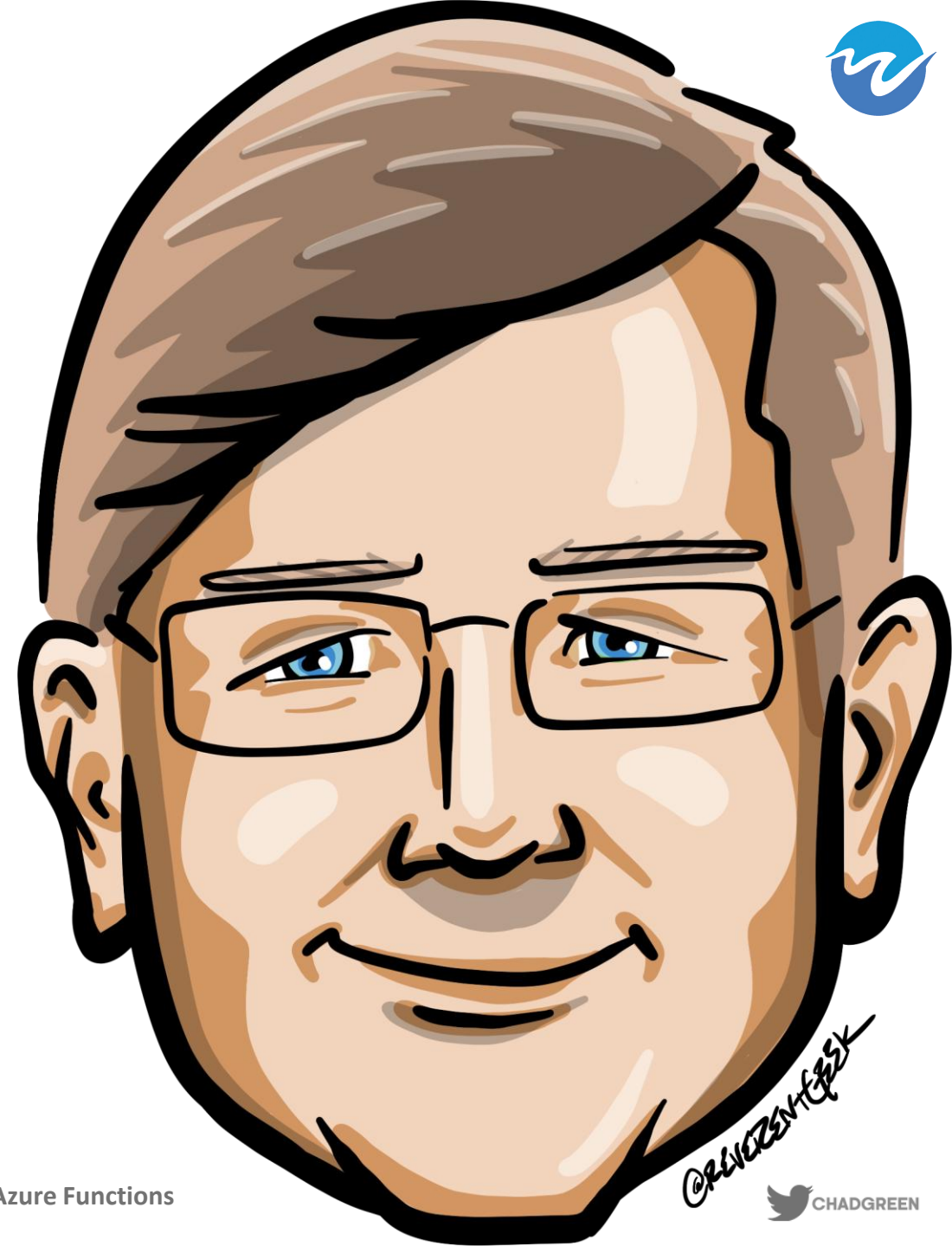
✉ chadgreen@chadgreen.com

💬 TaleLearnCode

🌐 ChadGreen.com

🐦 ChadGreen & TaleLearnCode

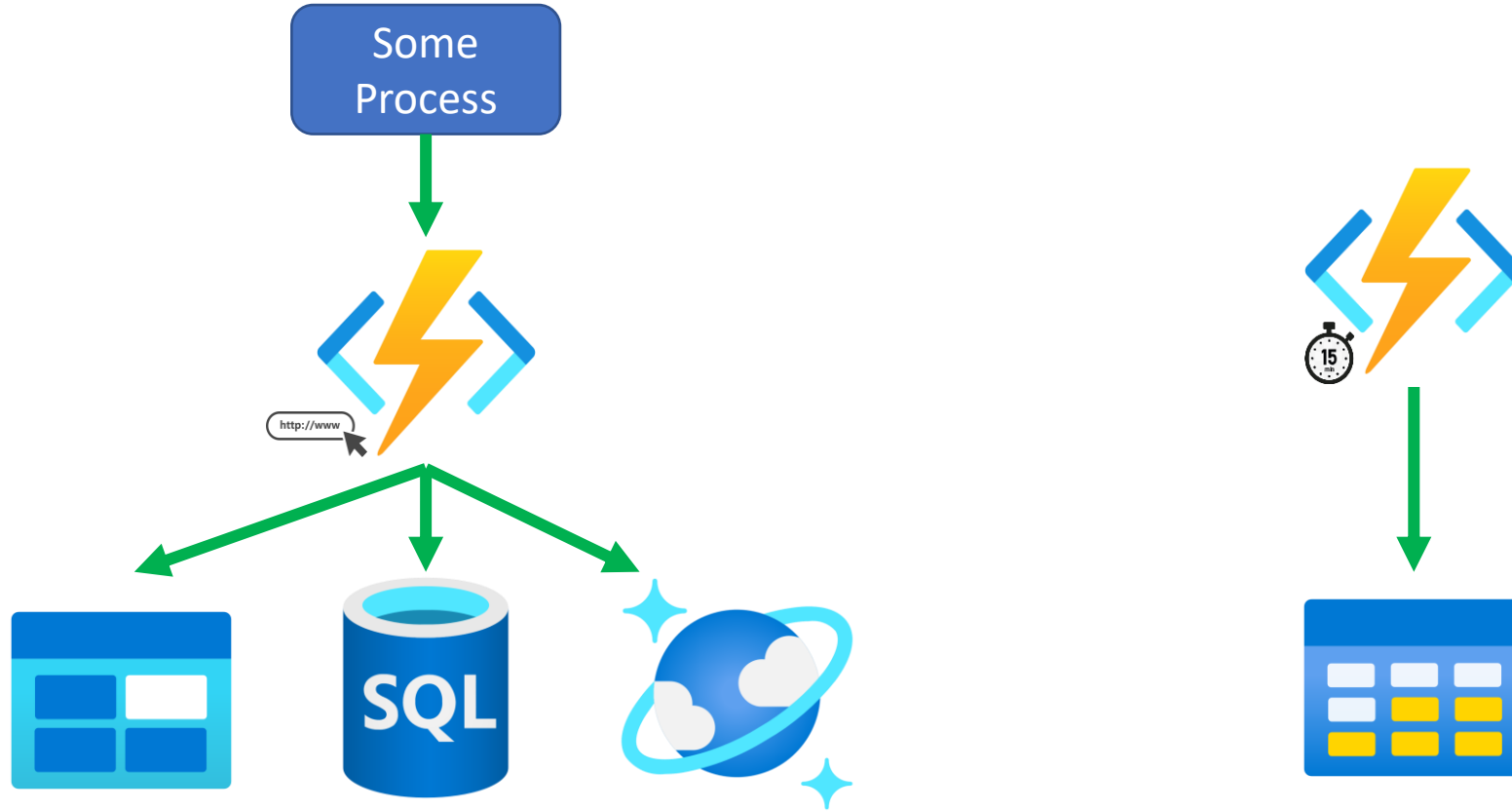
📌 ChadwickEGreen



@ChadGreen



Most Common Azure Function Patterns

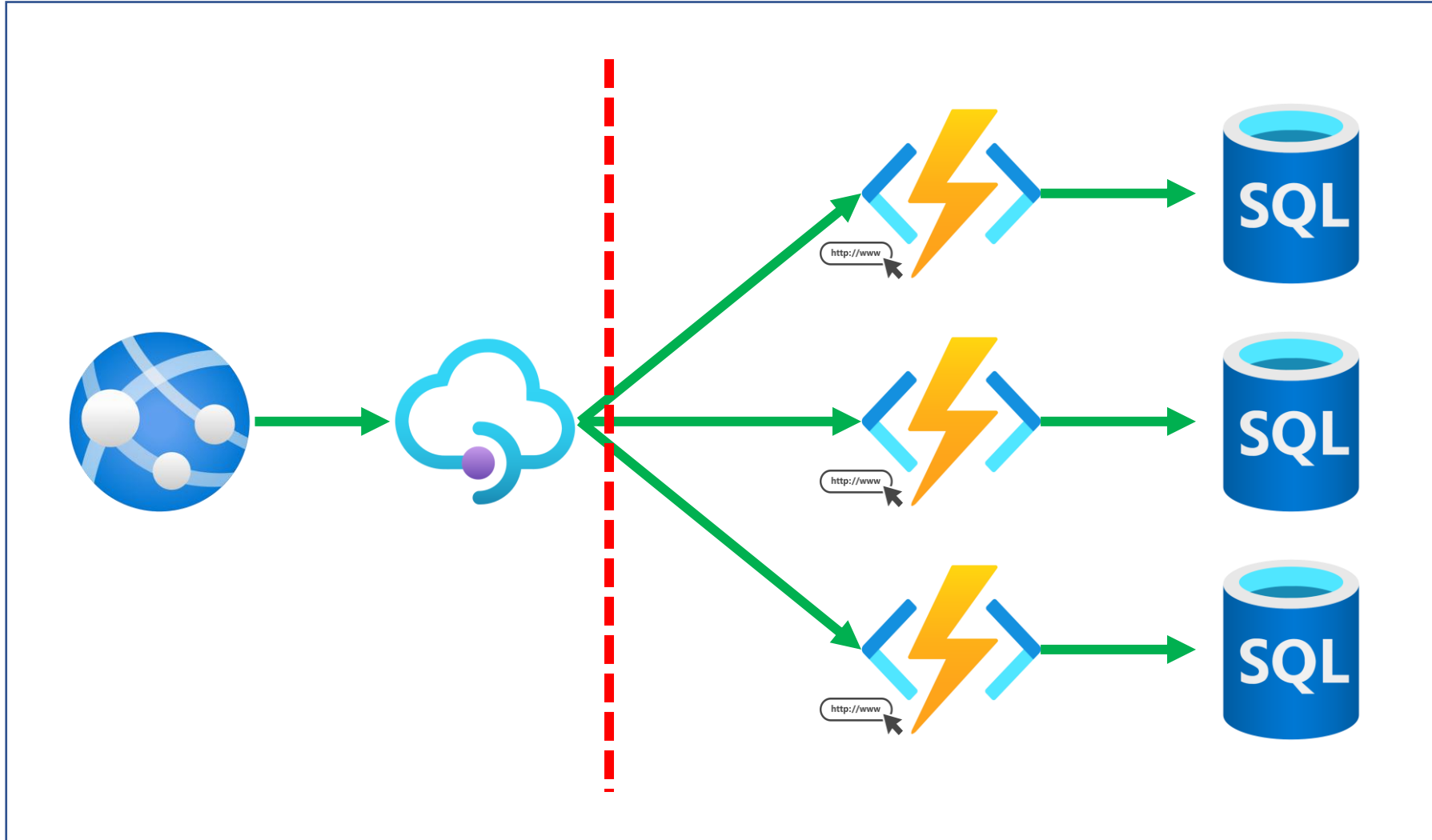




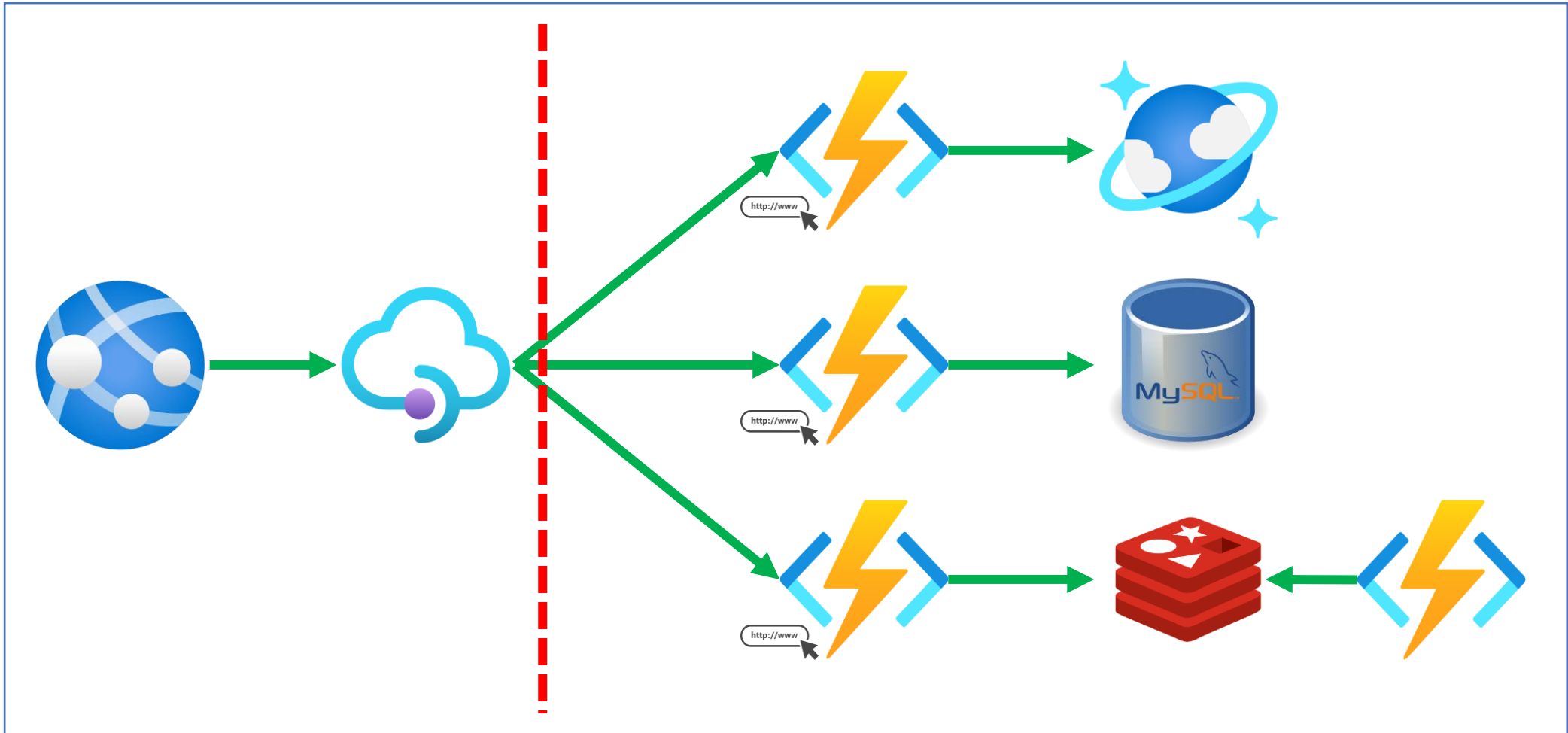
Real World Scenarios

Beyond Hello World: Getting Deeper into Azure Functions

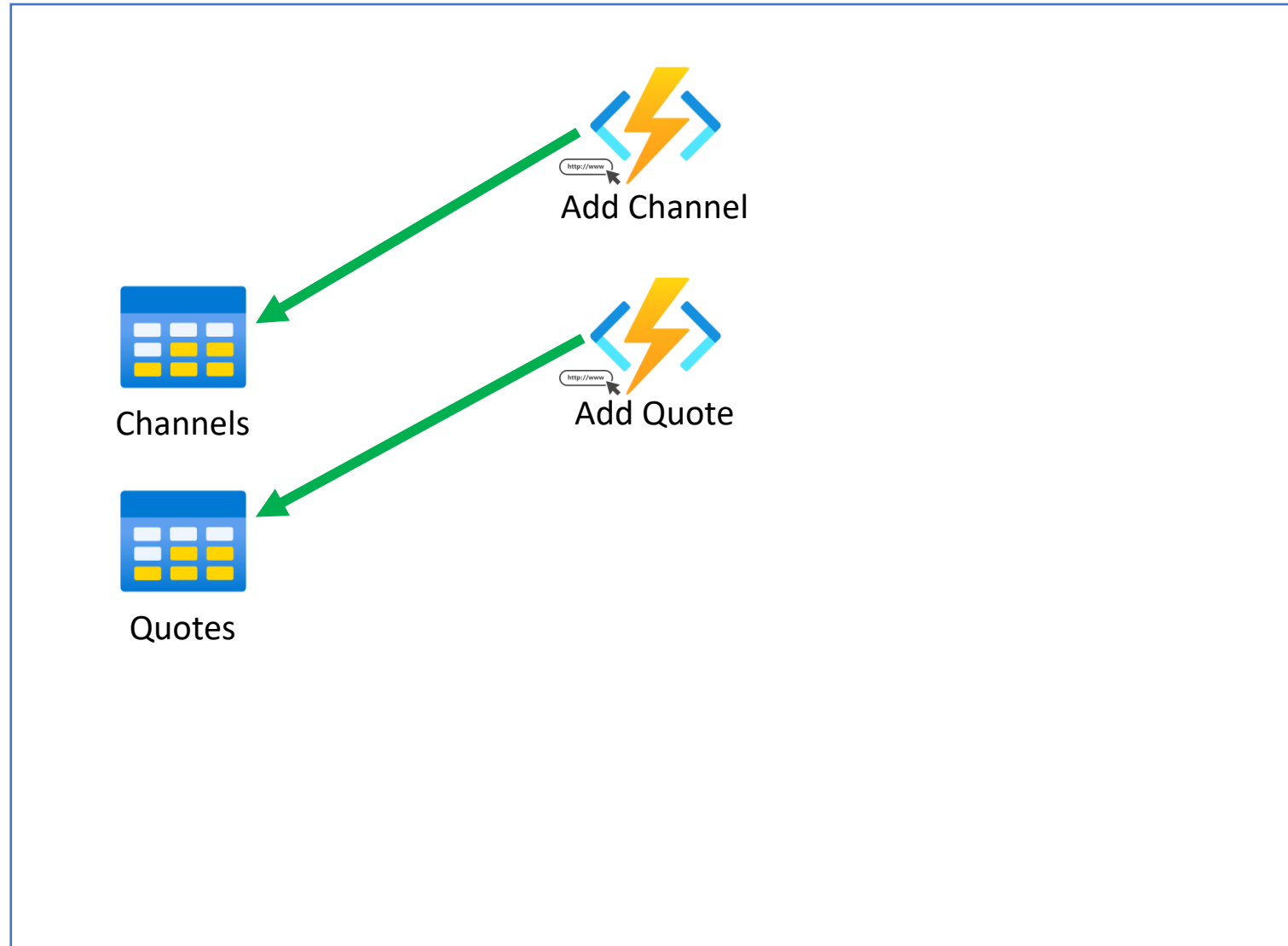
CRUD Microservice



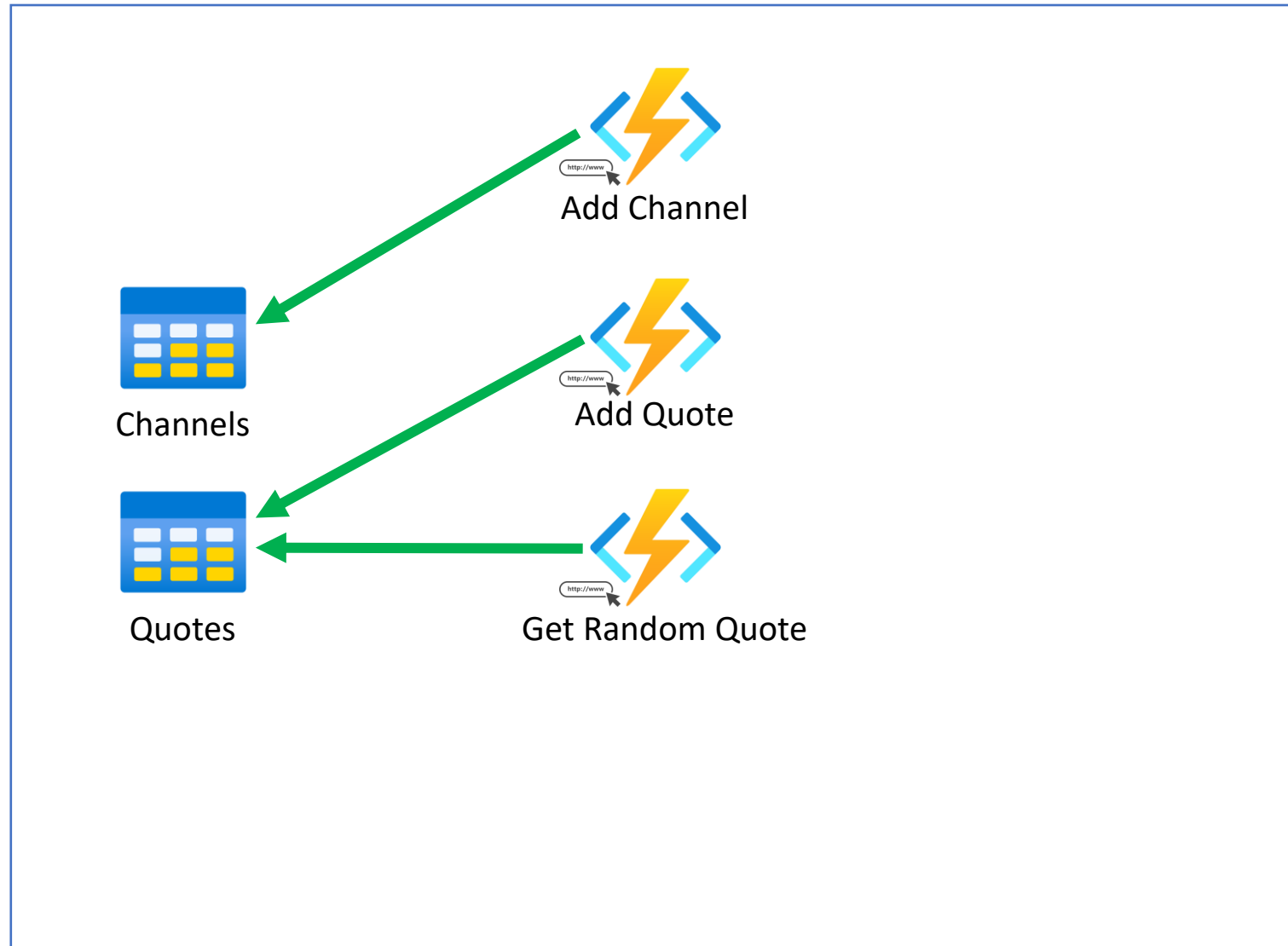
CRUD Microservice



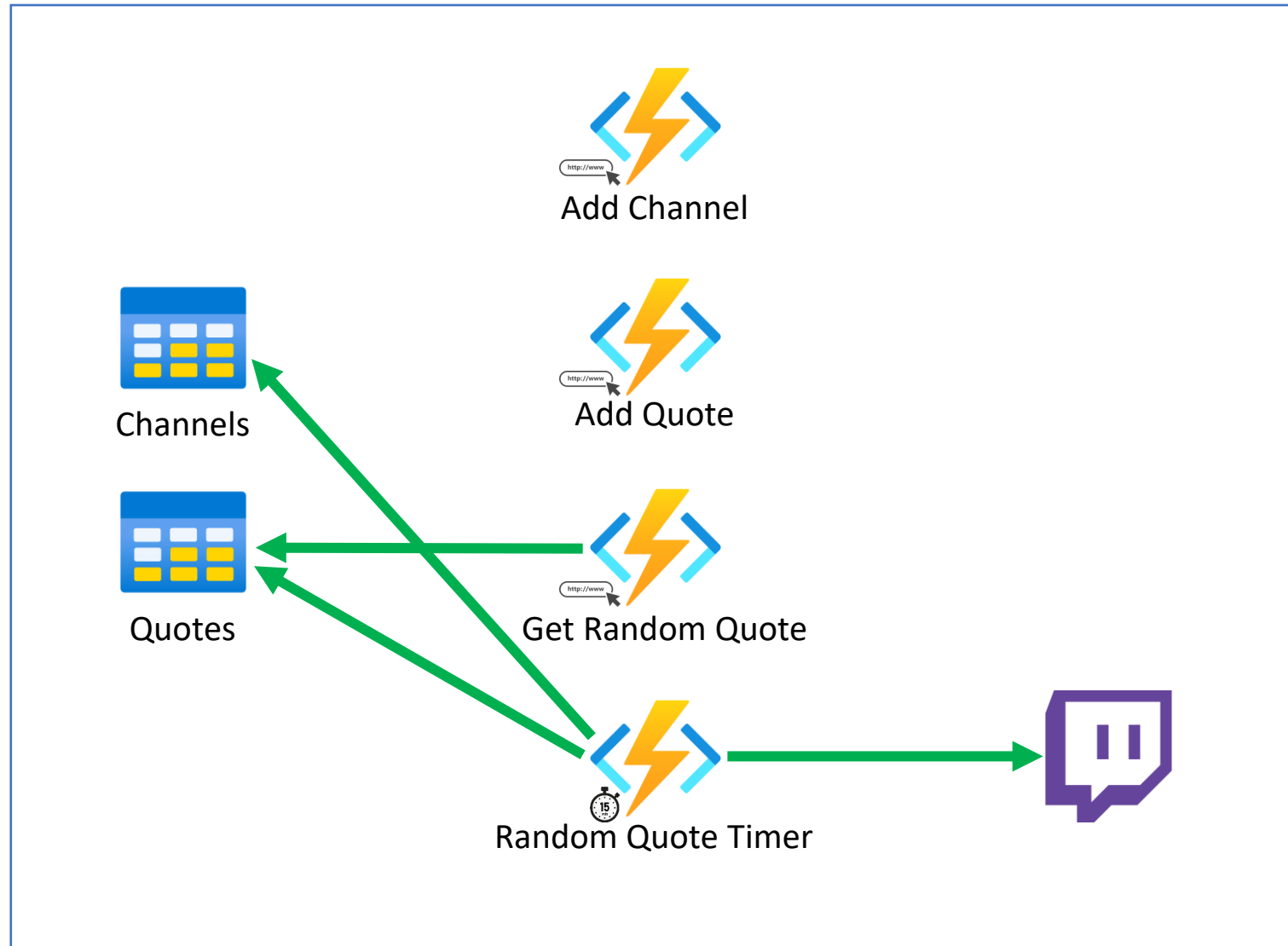
Random Quote Generator



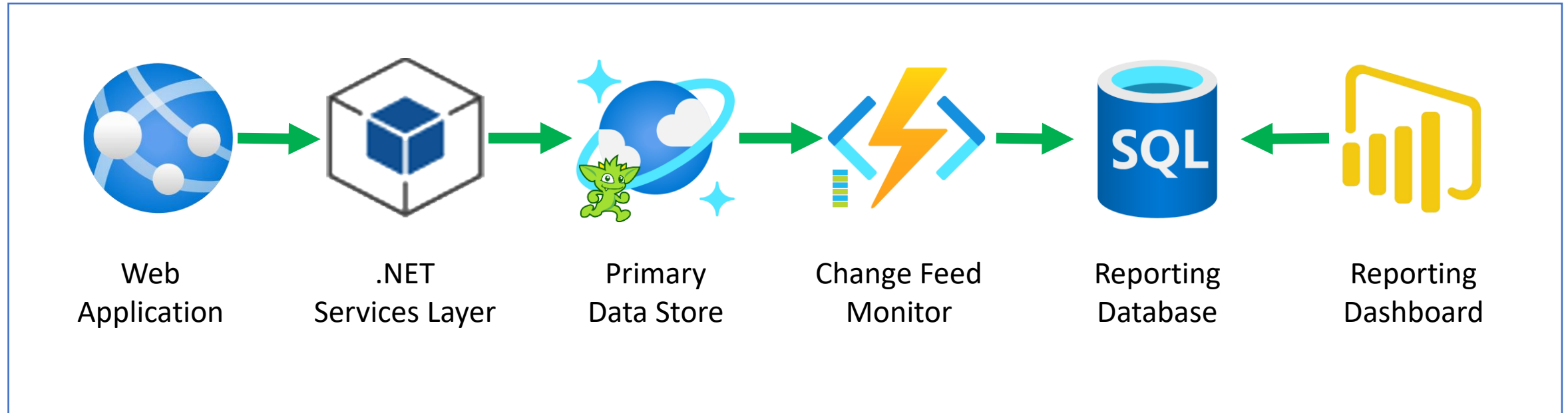
Random Quote Generator



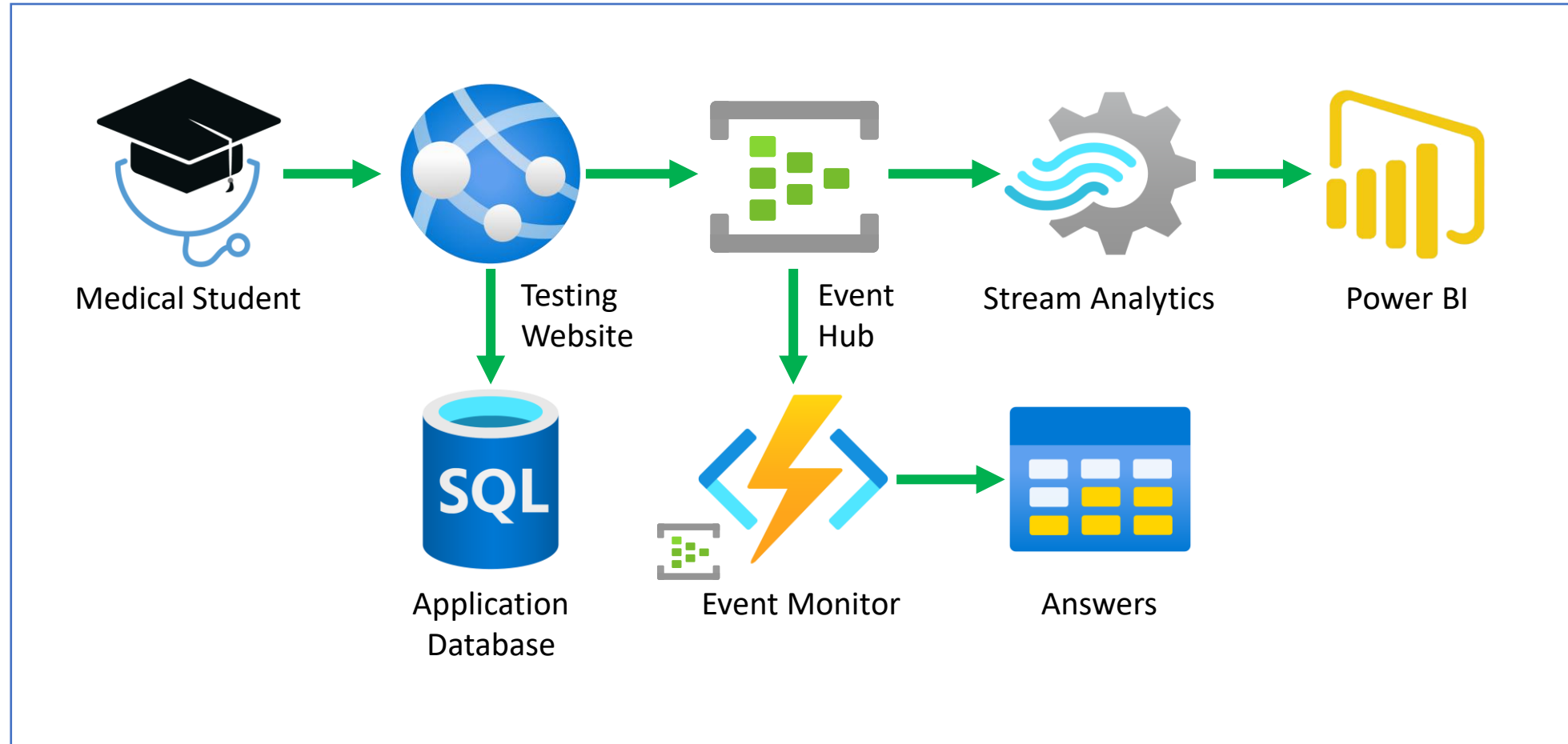
Random Quote Generator

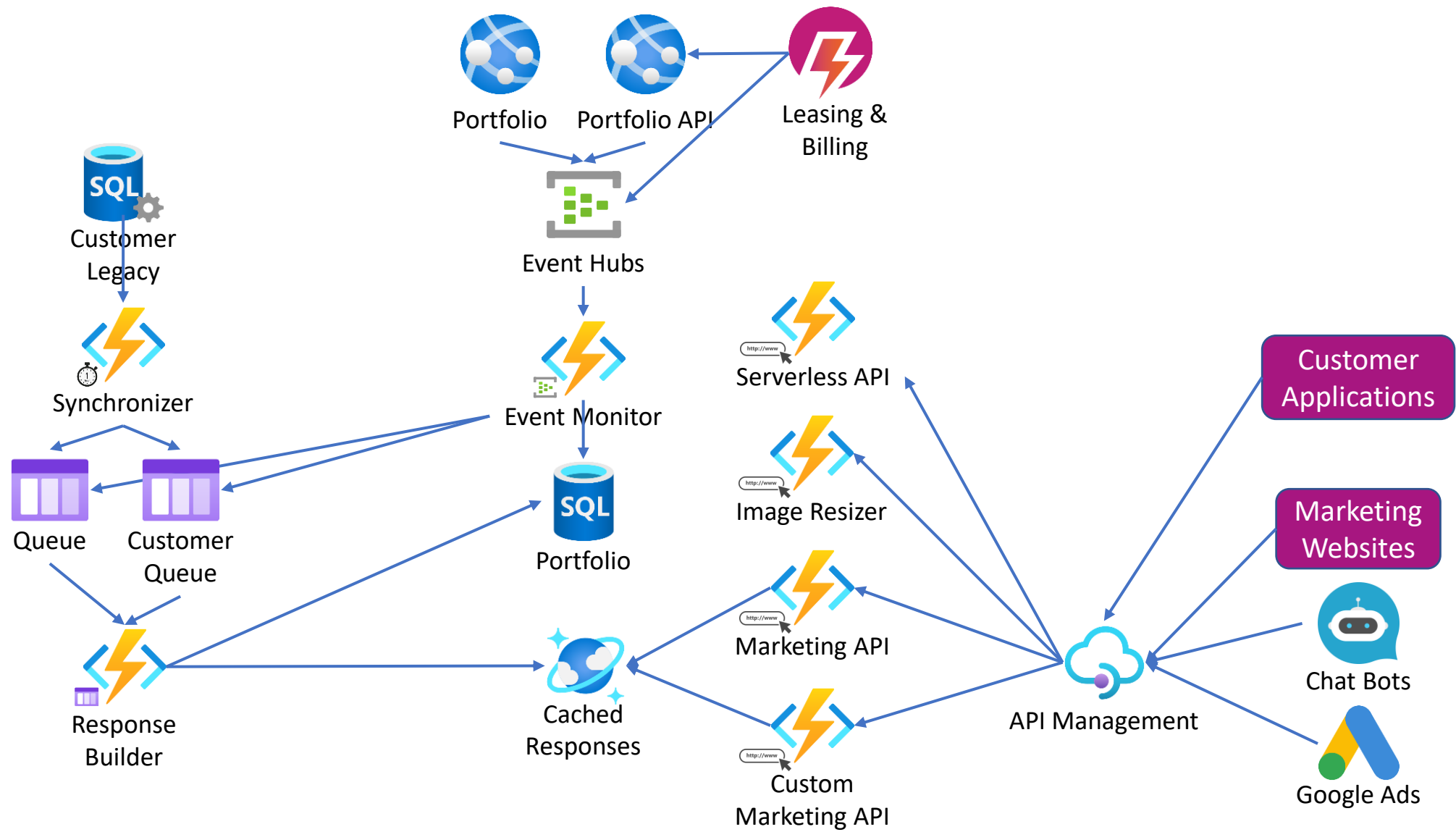


Replicating Data



Real-Time Reporting



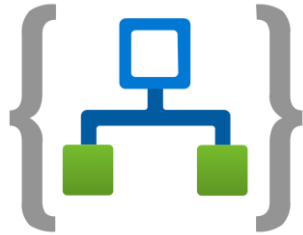




Microsoft Azure Serverless Offerings & Tools

Beyond Hello World: Getting Deeper into Azure Functions

Azure Serverless Offerings & Tools



Azure Logic Apps



API Management



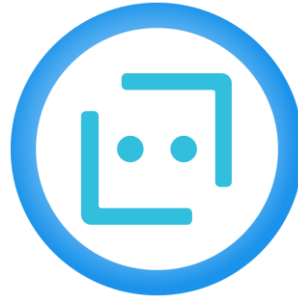
Azure Event Grid

Workflows and Integration

Azure Serverless Offerings & Tools



Azure Cognitive Services



Azure Bot Services



Azure Machine Learning

AI and Machine Learning

Azure Serverless Offerings & Tools



Azure SQL Database



Azure Cosmos DB

Database

Azure Serverless Offerings & Tools



Azure Blob Storage

Storage

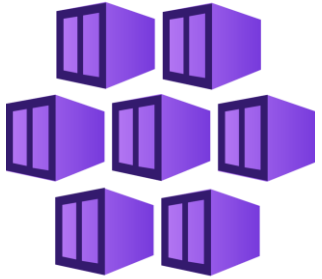
Azure Serverless Offerings & Tools



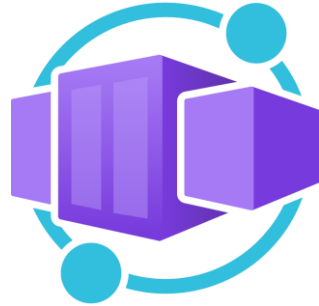
Azure Monitor

Monitoring

Azure Serverless Offerings & Tools



Azure Kubernetes Service



Azure Container Apps



Azure App Service



Azure Functions

Compute

Azure Serverless Offerings & Tools



Azure Stream Analytics

Analytics



Azure Functions Basics

Beyond Hello World: Getting Deeper into Azure Functions

Azure Functions



Code



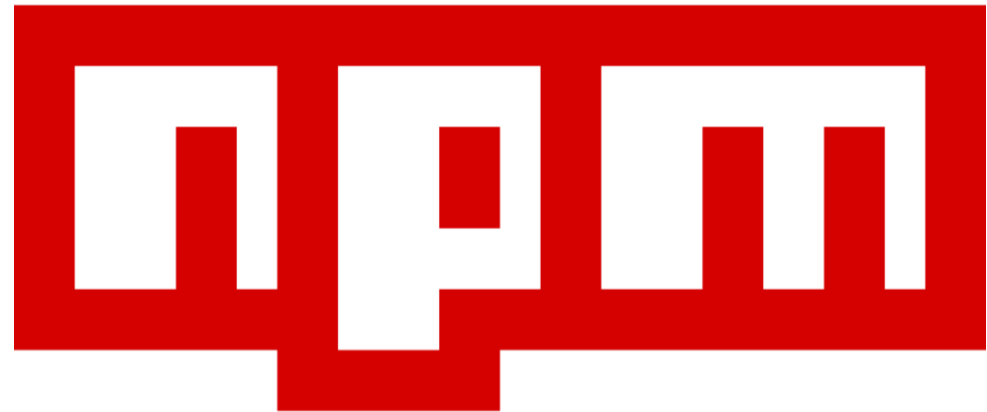
Events + data



Choice of Language



Bring your own dependencies



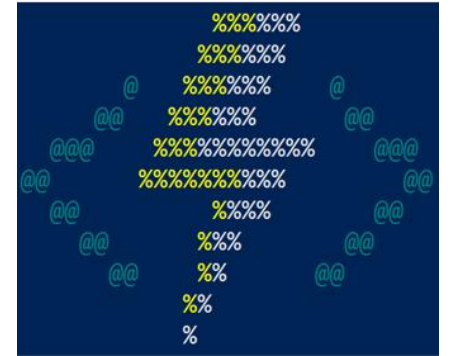
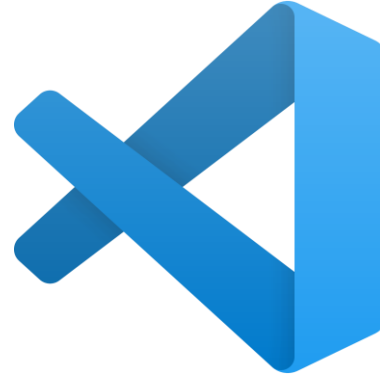
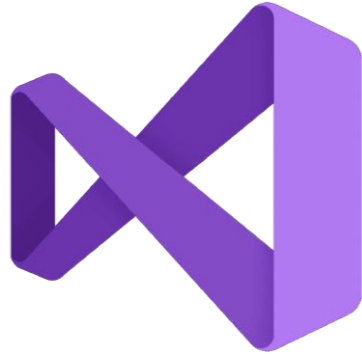
Simplified Integration



 RabbitMQ



Flexible Development



Maven™



Many Deployment Options



Consumption



Serverless

Many Deployment Options



Consumption



Serverless

App Service Plan



Free, Basic,
Standard,
Premium

Many Deployment Options



Consumption



Serverless

App Service Plan



Free, Basic,
Standard,
Premium

App Service Environment



Network
Isolation

Many Deployment Options



Consumption



Serverless

App Service Plan



Free, Basic,
Standard,
Premium

App Service Environment



Network
Isolation

Azure Stack



On Premises

Many Deployment Options



Consumption



Serverless

App Service Plan



Free, Basic,
Standard,
Premium

App Service Environment



Network
Isolation

Azure Stack



On Premises

Azure Functions Runtime



Functions on
Your Server

Many Deployment Options



Consumption



Serverless

App Service
Plan



Free, Basic,
Standard,
Premium

App Service
Environment



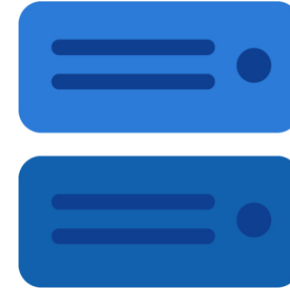
Network
Isolation

Azure Stack



On Premises

Azure
Functions
Runtime



Functions on
Your Server

Azure
IoT Edge



On Devices

Open Source



The screenshot shows the GitHub repository for Azure Functions. The repository is public and has 164 watches, 894 stars, and 150 forks. The main branch is 'main' with 3 branches and 0 tags. The repository contains several files and folders, including .github, Functions Fox, functions-premium-plan, schemas, .gitignore, VS-AzureTools-ReleaseNotes.md, and readme.md. The README file is expanded, showing the title 'Azure Functions' and a description: 'Azure Functions is an event driven, compute-on-demand experience that extends the existing Azure application platform with capabilities to implement code triggered by events occurring in virtually any Azure or 3rd party service as well as on-premises systems. Azure Functions allows developers to take action by connecting to data sources or messaging solutions, thus making it easy to process and react to events. Azure Functions scale based on demand and you pay only for the resources you consume. This repository acts as a directory for folks looking for the various resources we have for Azure Functions.' The 'Get Started with Azure Functions' section is also visible. On the right side, there are sections for 'About' (No description or website provided), 'Releases' (No releases published), 'Packages' (No packages published), and 'Contributors' (21 contributors).



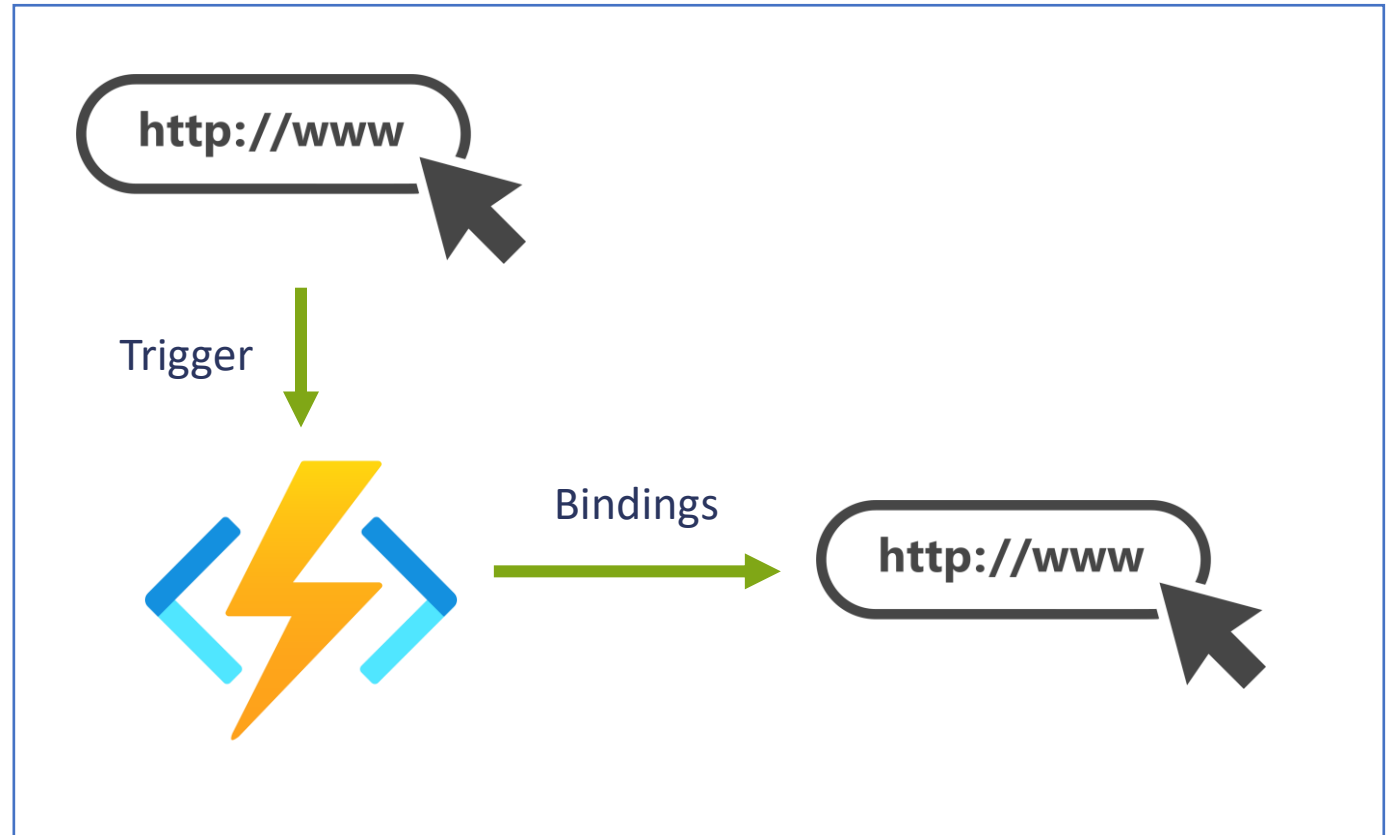
Supported Triggers and Bindings

Beyond Hello World: Getting Deeper into Azure Functions

HTTP



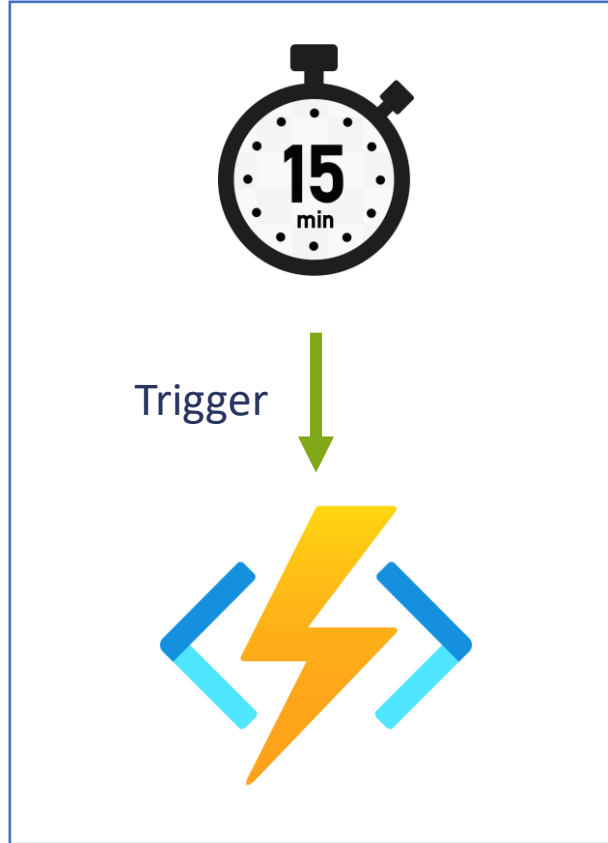
Tigger	✓
Input	✗
Output	✓
Consumption	✓
Premium	✓
Dedicated	✓
C# In-Process	✓
C# Isolated	✓
Java	✓
JavaScript	✓
PowerShell	✓
Python	✓



Timer



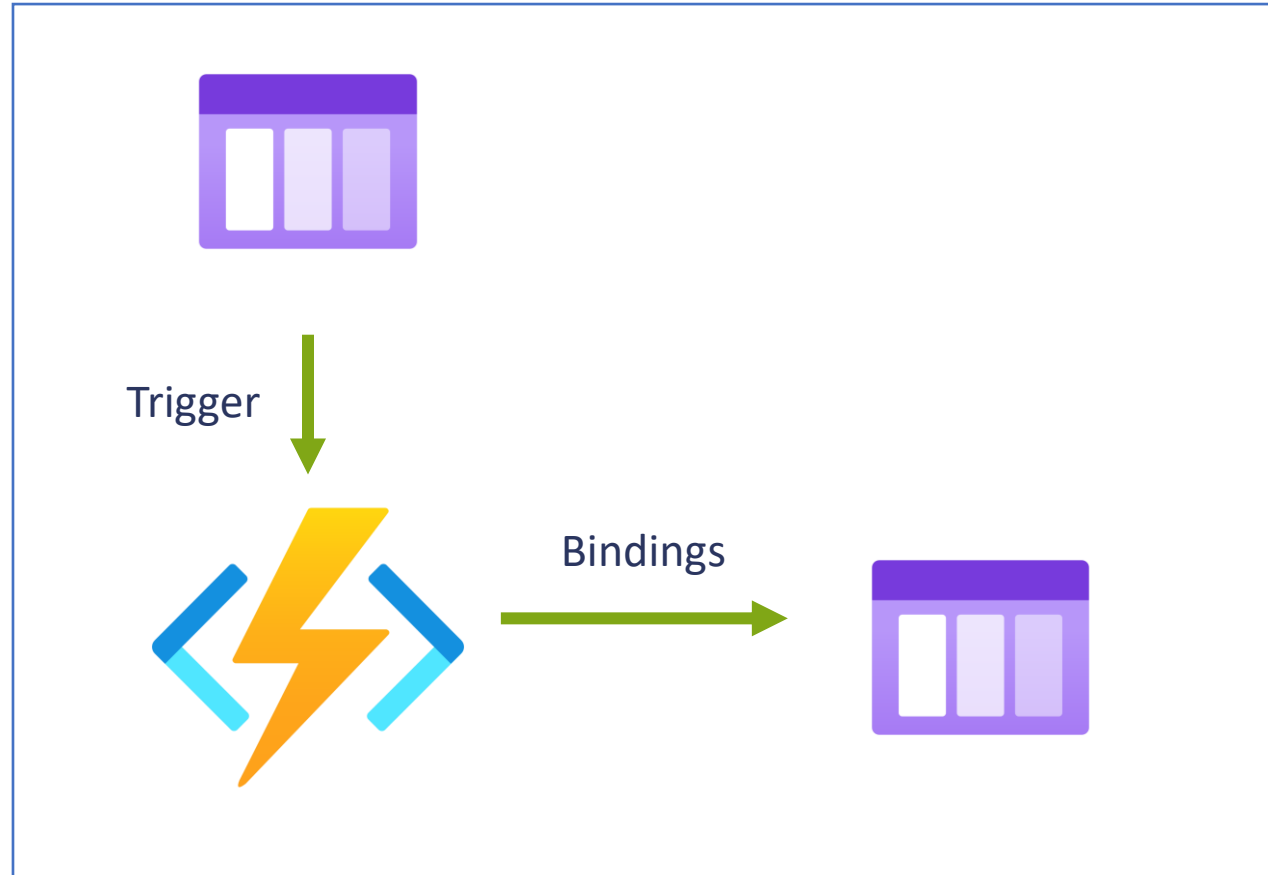
Tigger	✓
Input	✗
Output	✗
Consumption	✓
Premium	✓
Dedicated	✓



Queue Storage



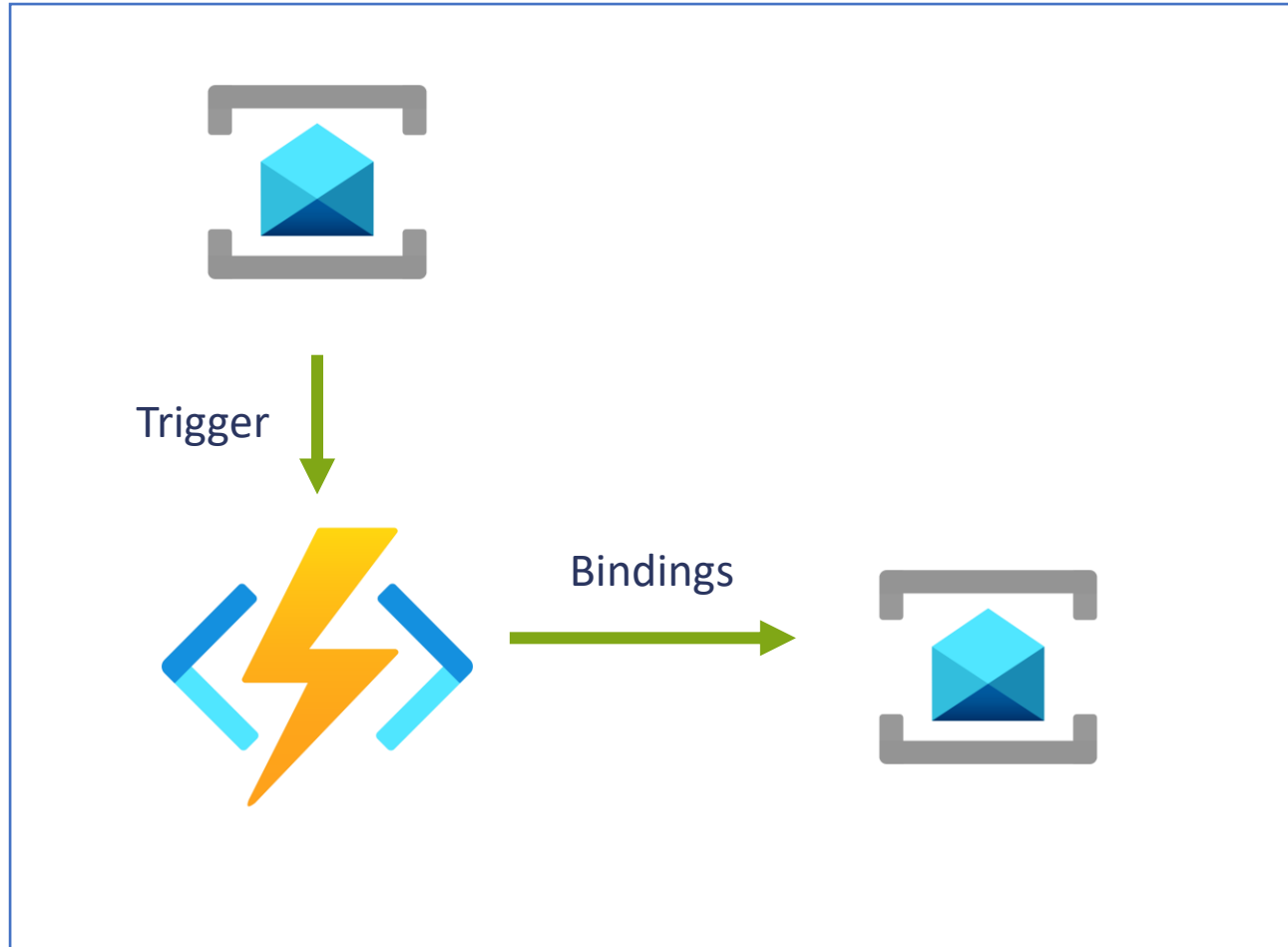
Tigger	✓
Input	✗
Output	✓
Consumption	✓
Premium	✓
Dedicated	✓



Service Bus



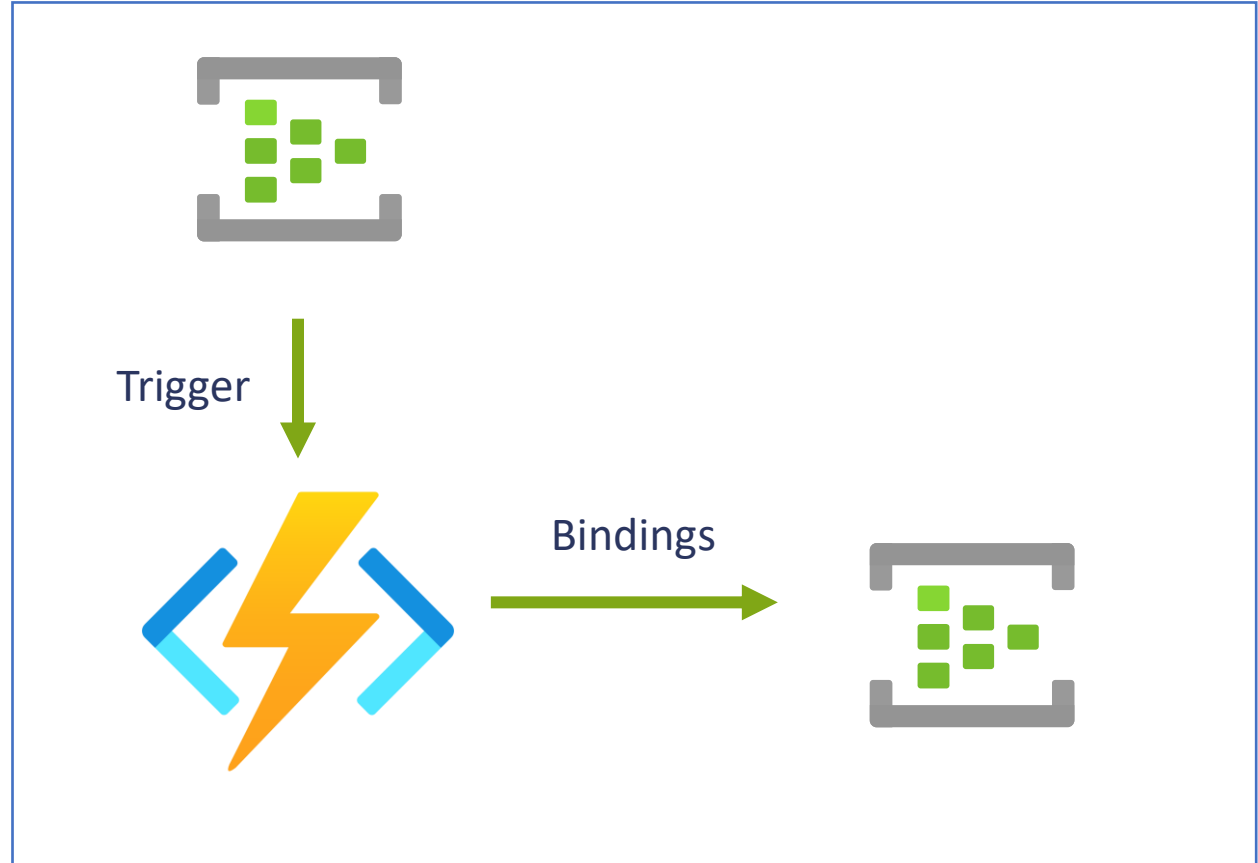
Tigger	✓
Input	✗
Output	✓
Consumption	✓
Premium	✓
Dedicated	✓



Event Hub



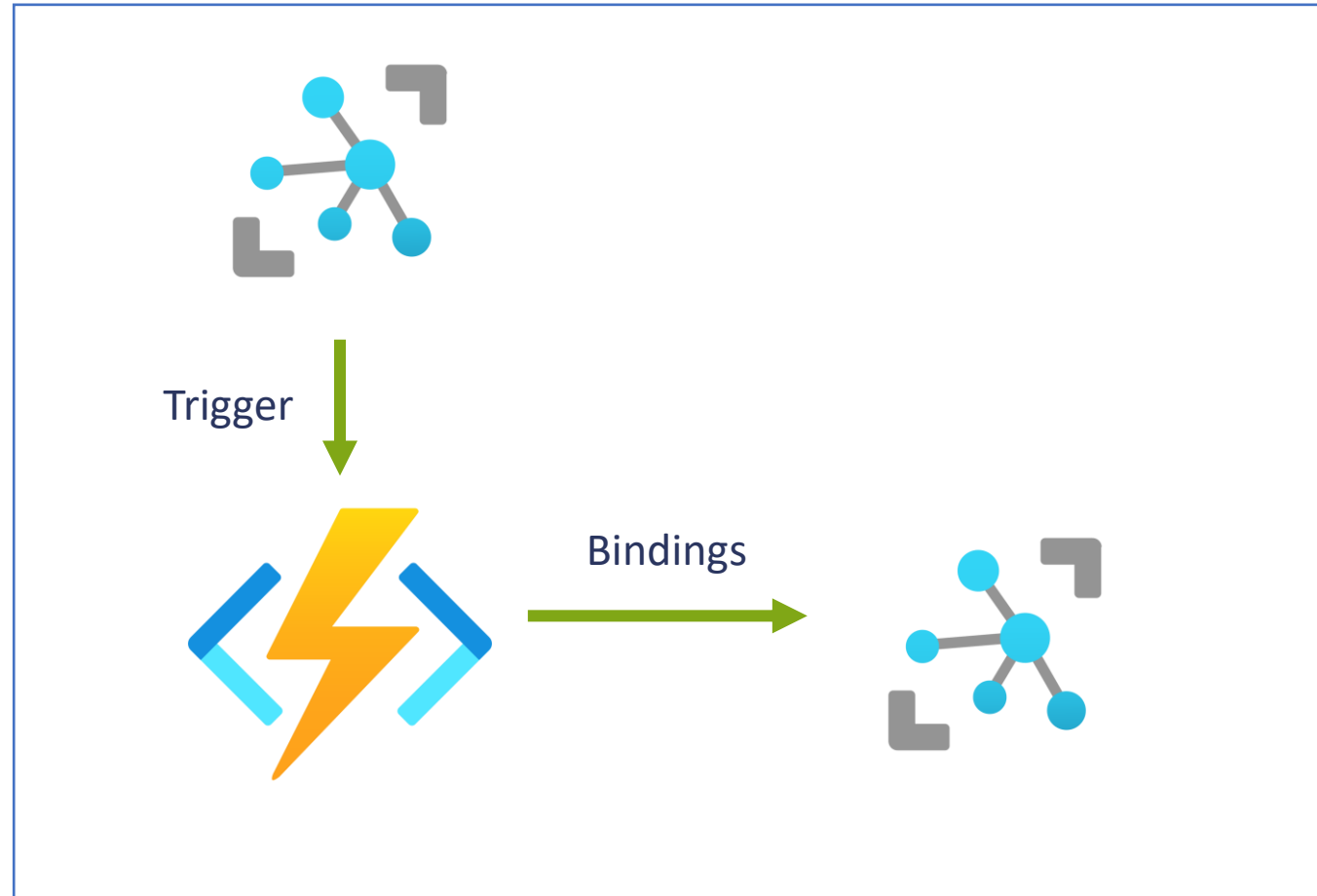
Tigger	✓
Input	✗
Output	✓
Consumption	✓
Premium	✓
Dedicated	✓



IoT Hub



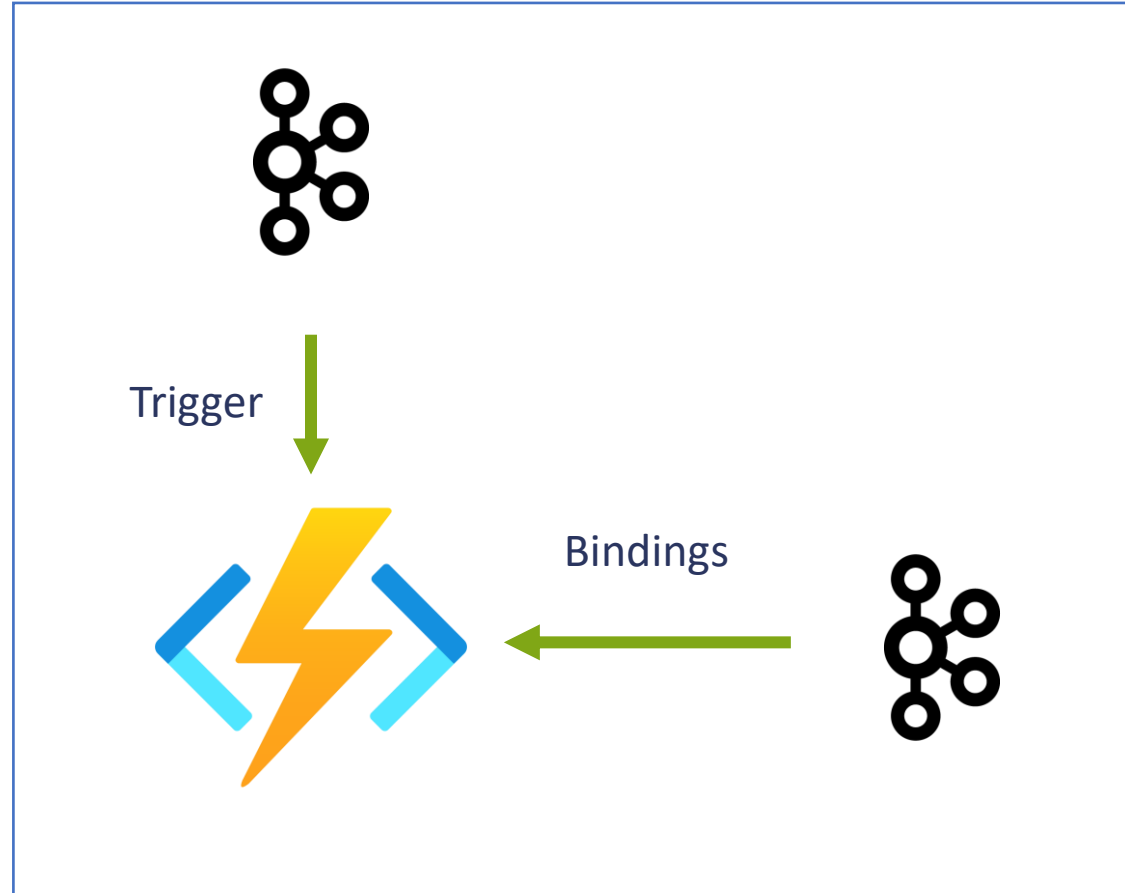
Tigger	✓
Input	✗
Output	✓
Consumption	✓
Premium	✓
Dedicated	✓



Kafka



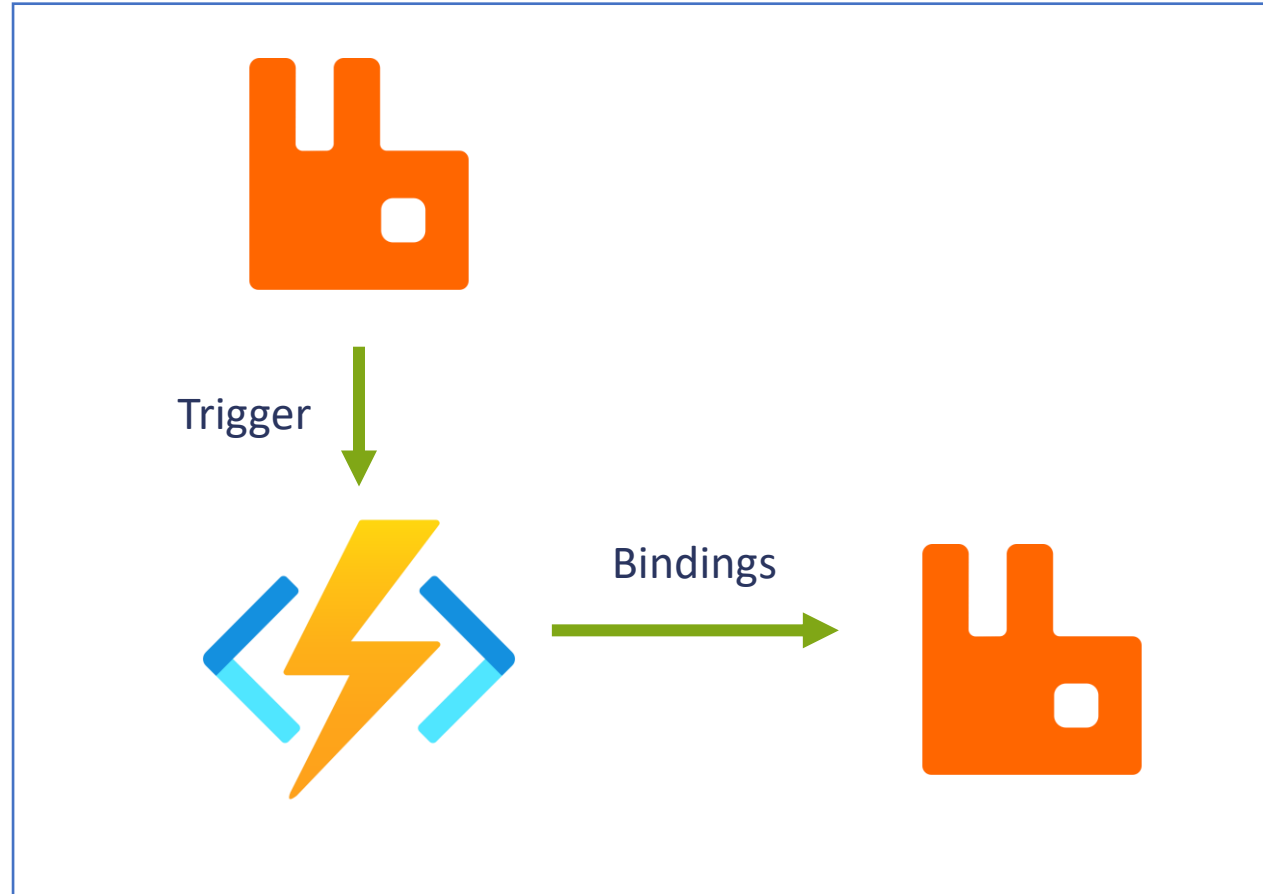
Tigger	✓
Input	✗
Output	✓
Consumption	✗
Premium	✓
Dedicated	✓



RabbitMQ



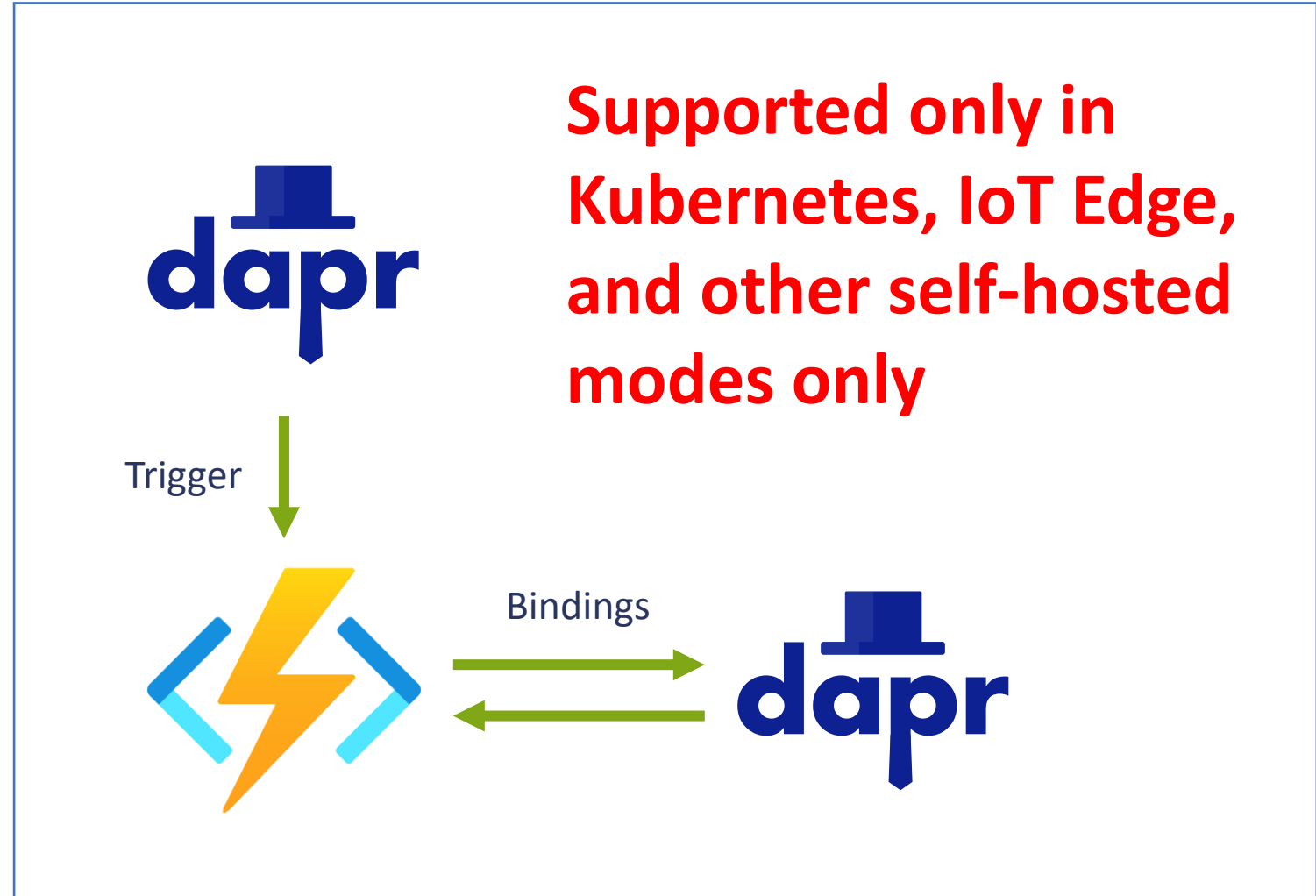
Tigger	✓
Input	✗
Output	✓
Consumption	✗
Premium	✓
Dedicated	✓



Dapr



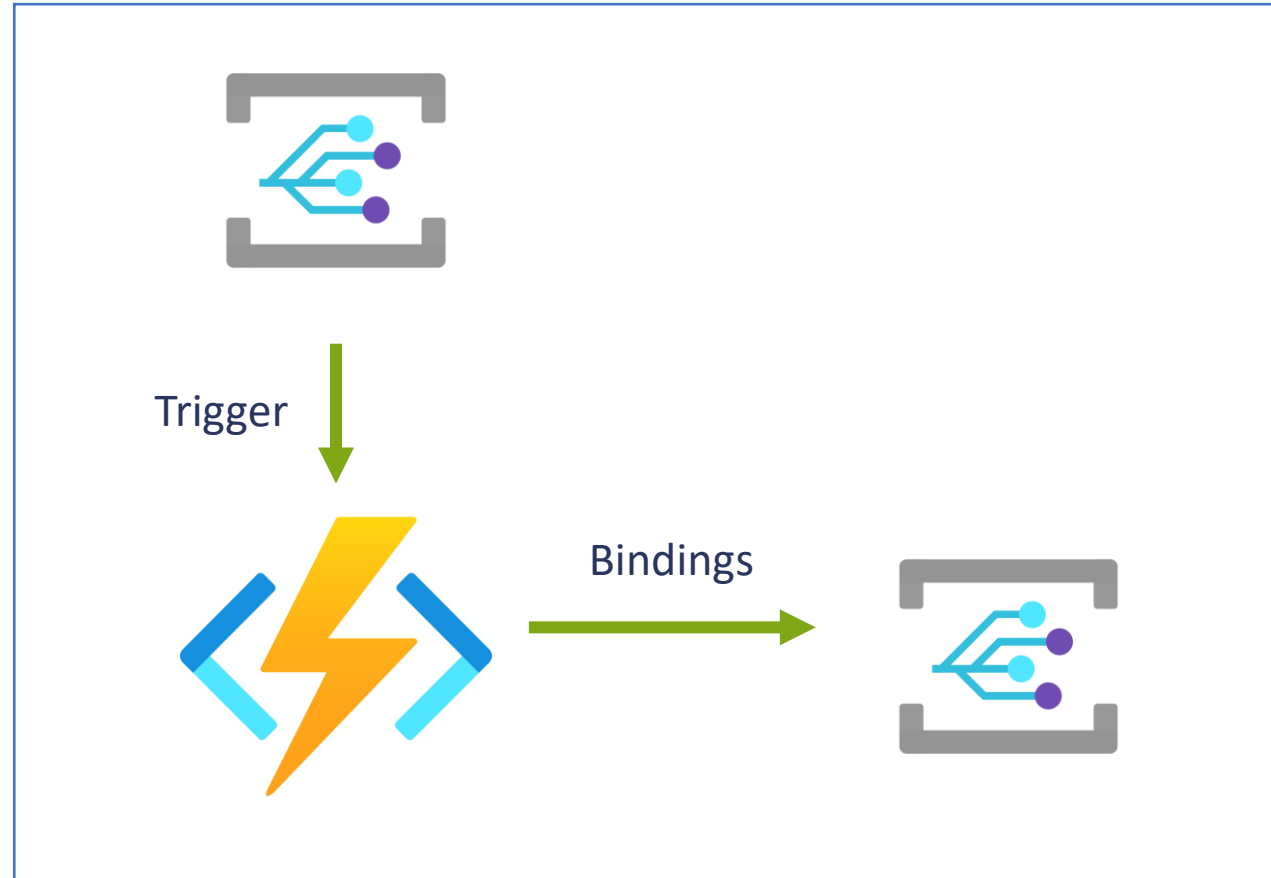
Tigger	✓
Input	✓
Output	✓
Consumption	✗
Premium	✗
Dedicated	✓



Event Grid



Tigger	✓
Input	✗
Output	✓
Consumption	✓
Premium	✓
Dedicated	✓



Blob



Tigger	✓
Input	✓
Output	✓
Consumption	✓
Premium	✓
Dedicated	✓

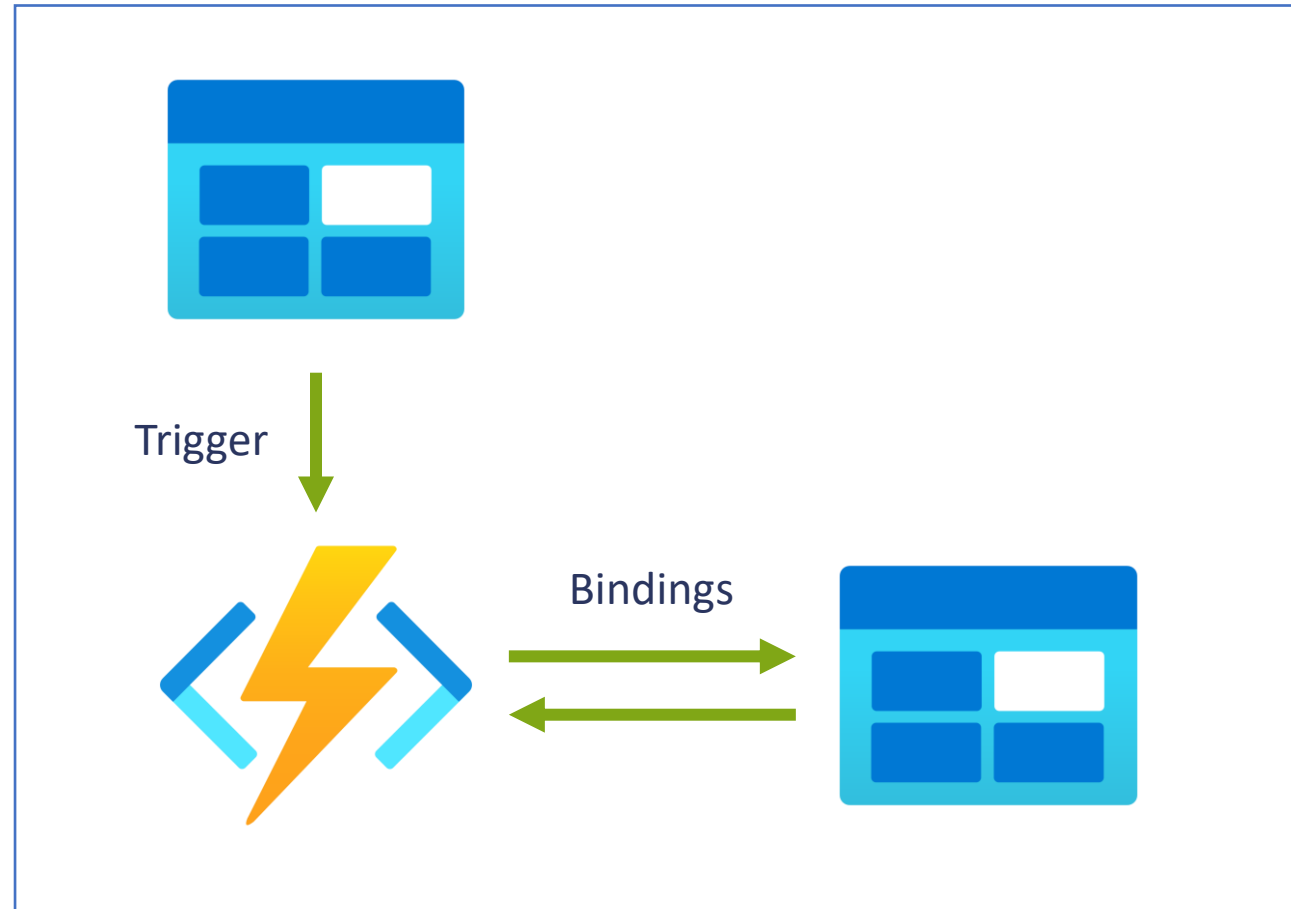
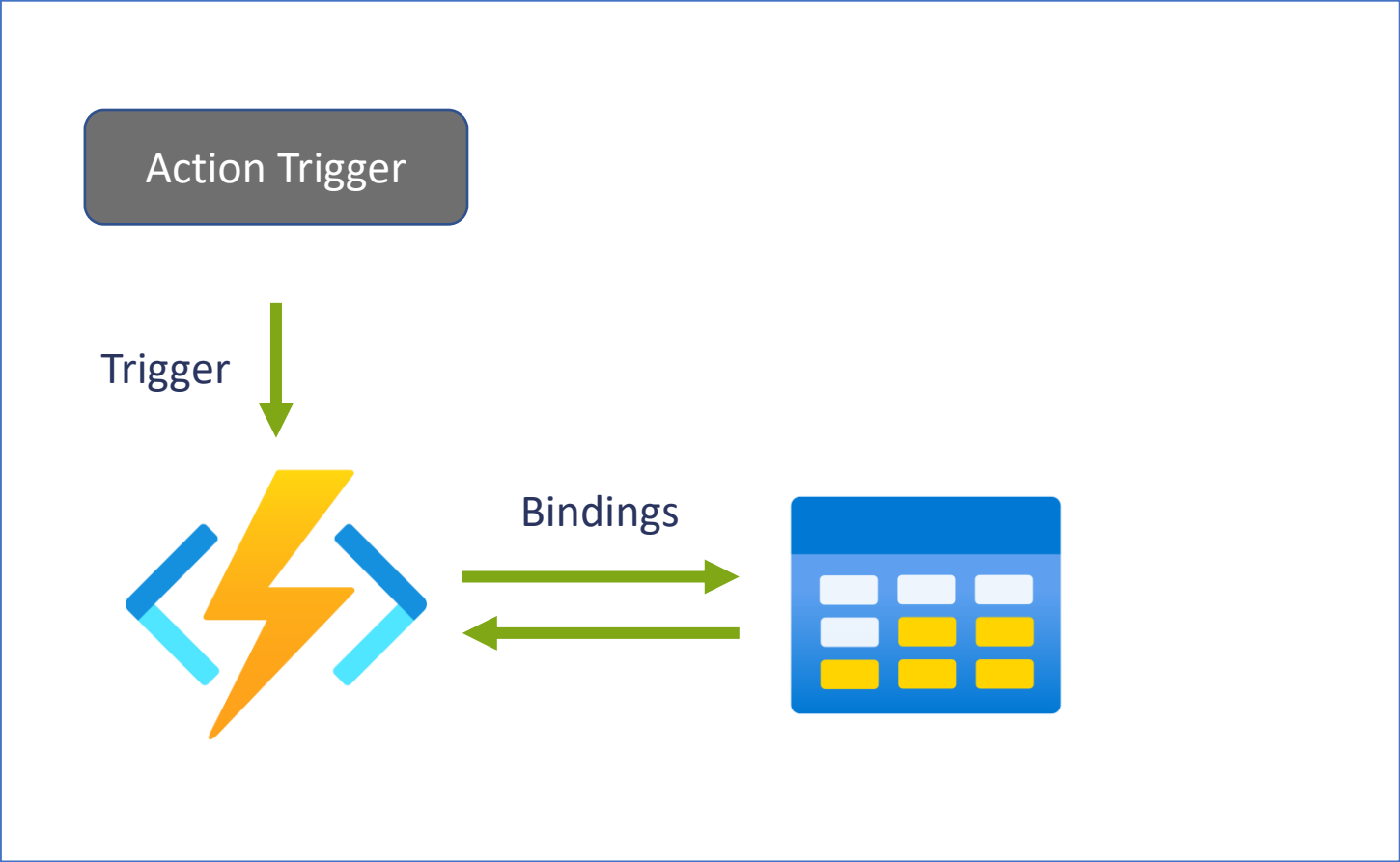


Table Storage



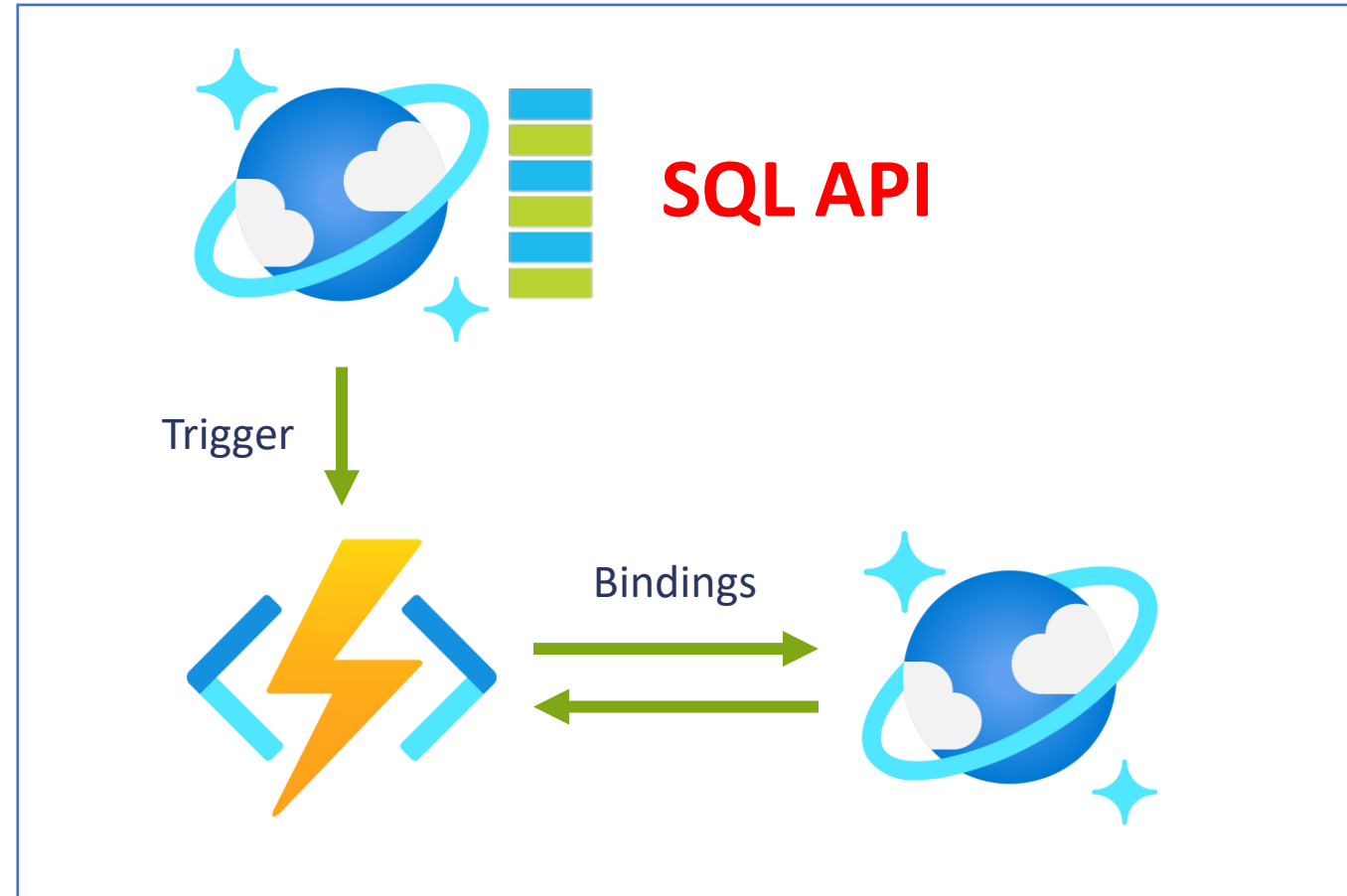
Tigger	✗
Input	✓
Output	✓
Consumption	✓
Premium	✓
Dedicated	✓



Cosmos DB



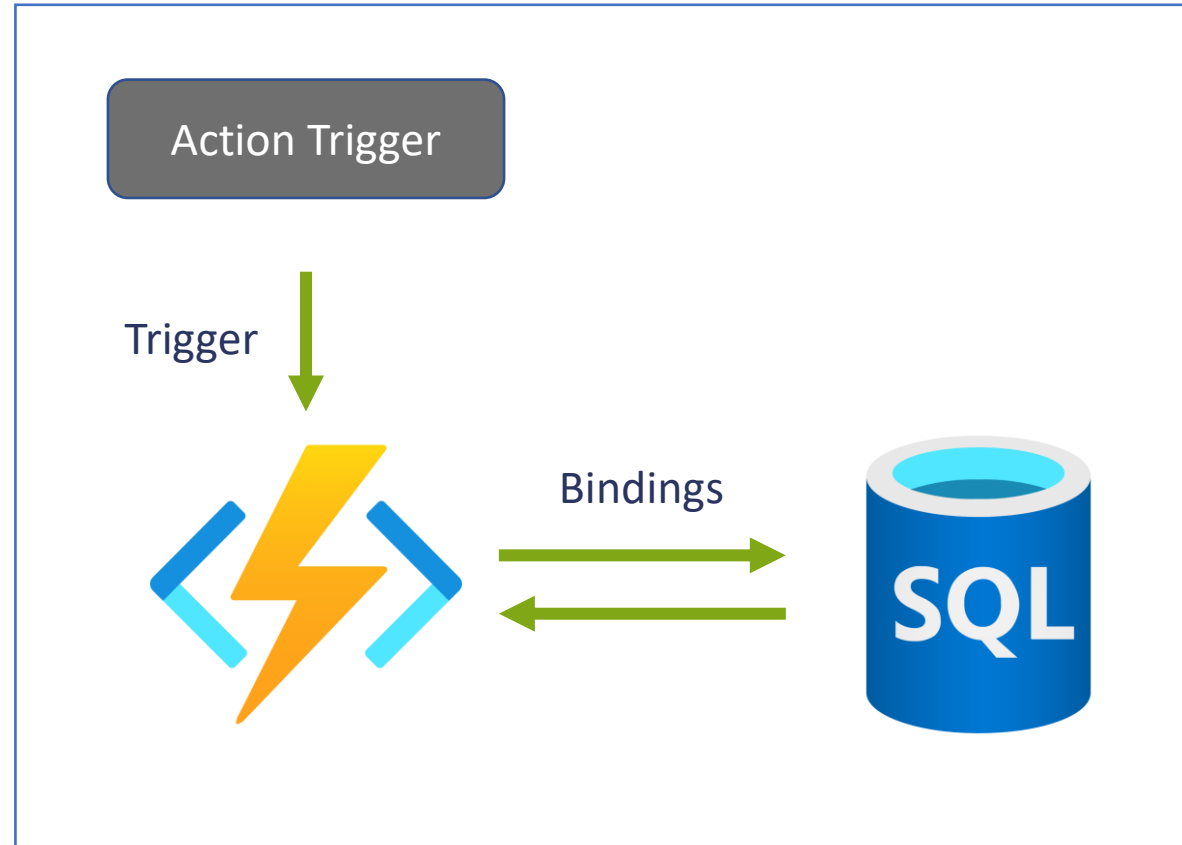
Tigger	✓
Input	✓
Output	✓
Consumption	✓
Premium	✓
Dedicated	✓



Azure SQL



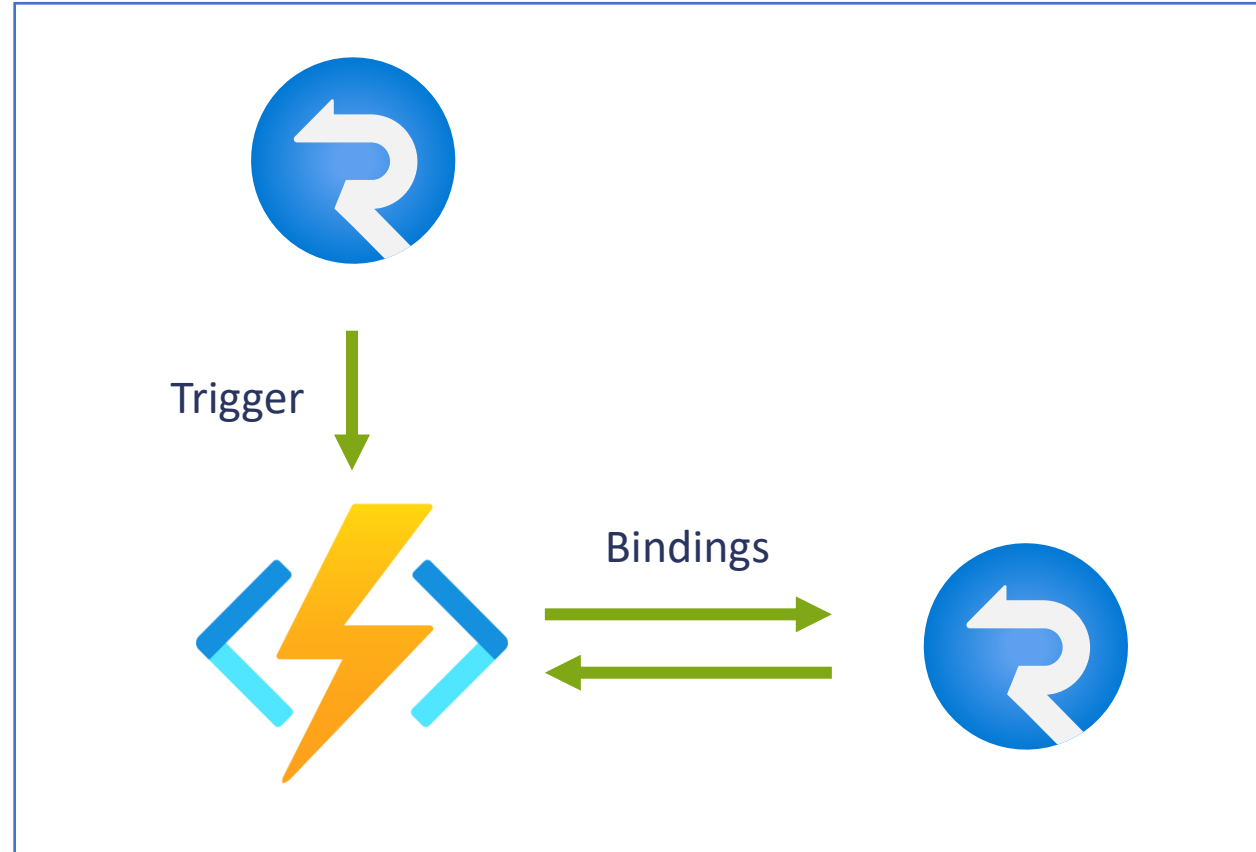
Tigger	✗
Input	✓
Output	✓
Consumption	✗
Premium	✓
Dedicated	✓



SignalR



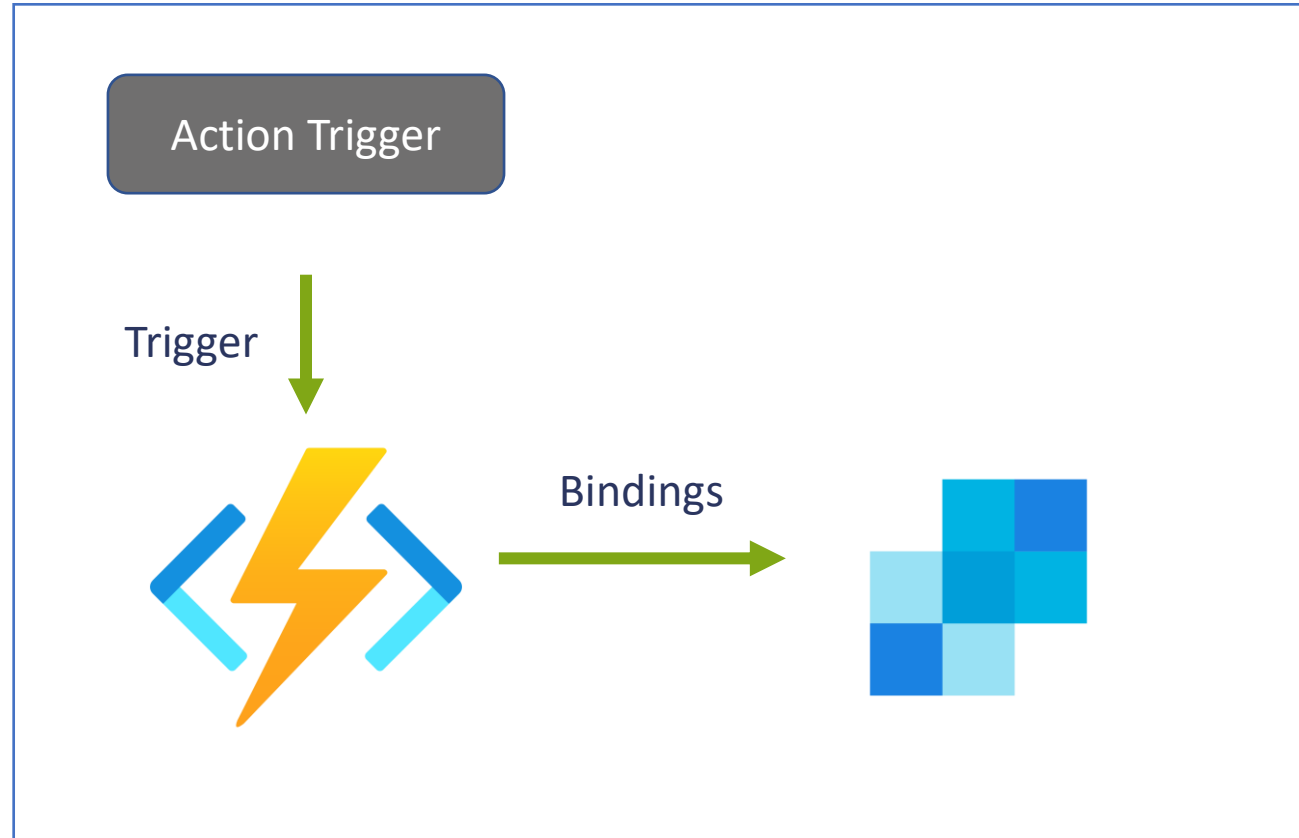
Tigger	✓
Input	✓
Output	✓
Consumption	✓
Premium	✓
Dedicated	✓



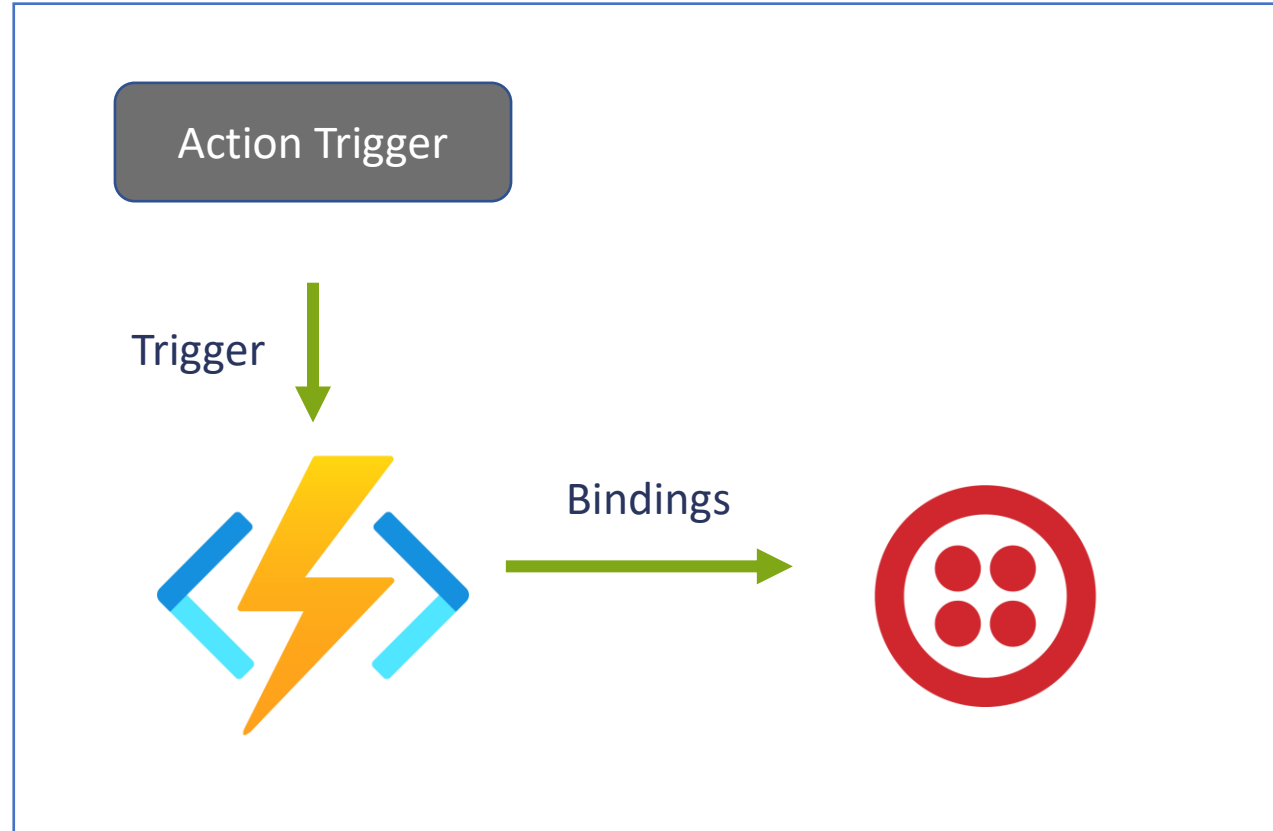
SendGrid



Tigger	✗
Input	✗
Output	✓
Consumption	✓
Premium	✓
Dedicated	✓



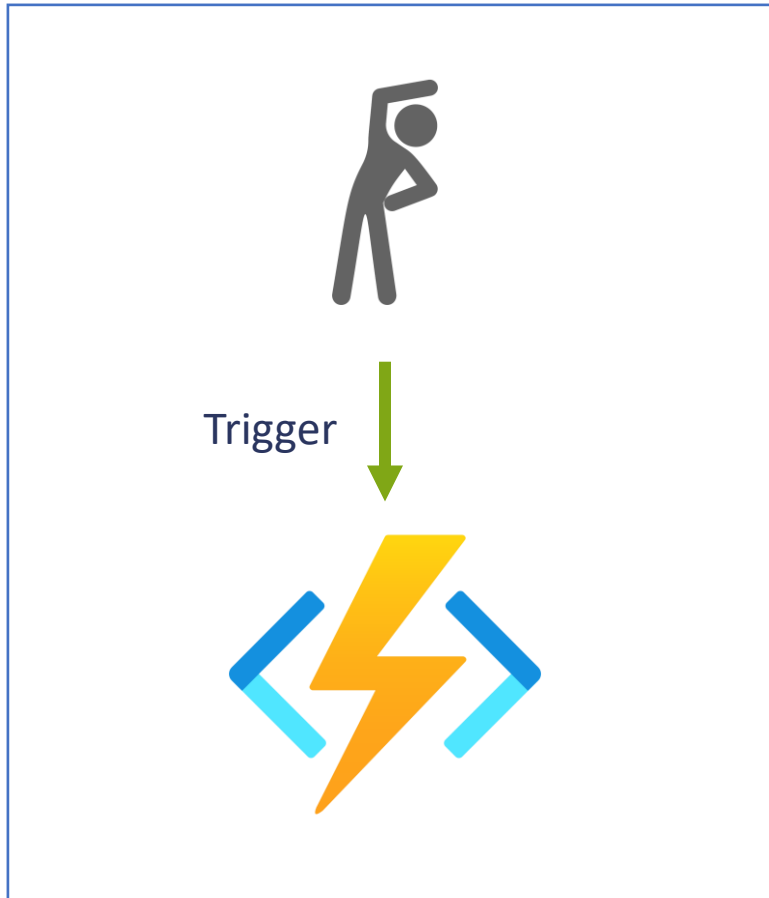
Tigger	✗
Input	✗
Output	✓
Consumption	✓
Premium	✓
Dedicated	✓



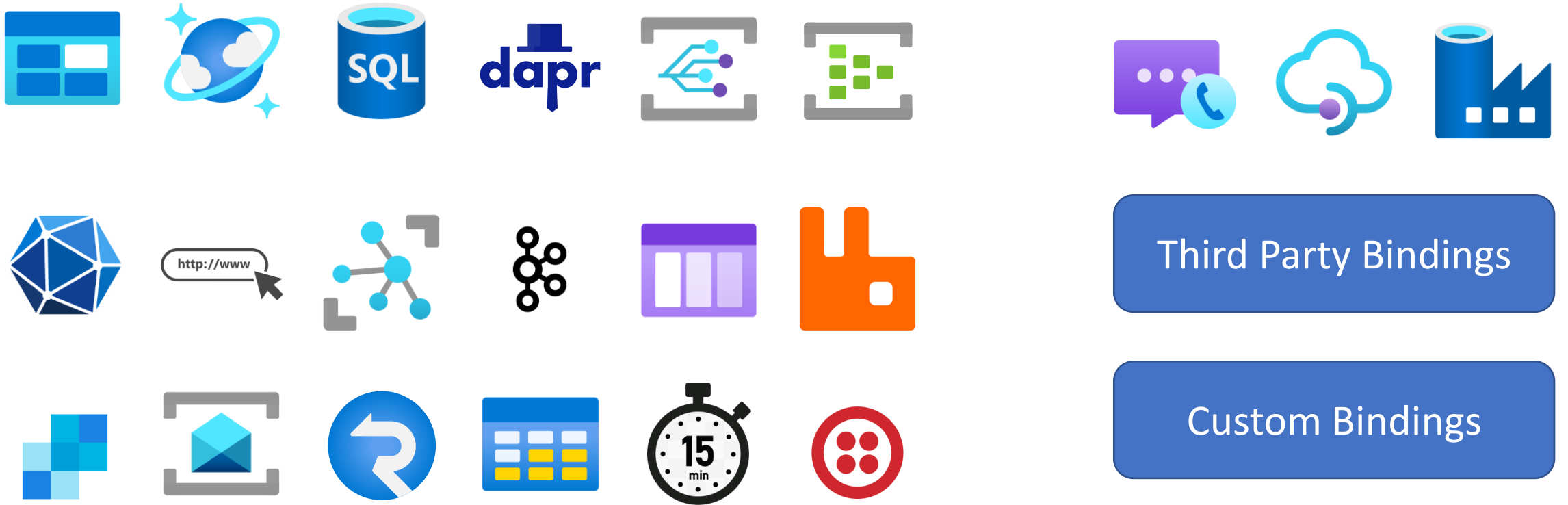
Warmup



Tigger	✓
Input	✗
Output	✗
Consumption	✗
Premium	✓
Dedicated	✓



Supported Bindings



Third Party Bindings

Custom Bindings



Durable Functions

Beyond Hello World: Getting Deeper into Azure Functions

Durable Functions

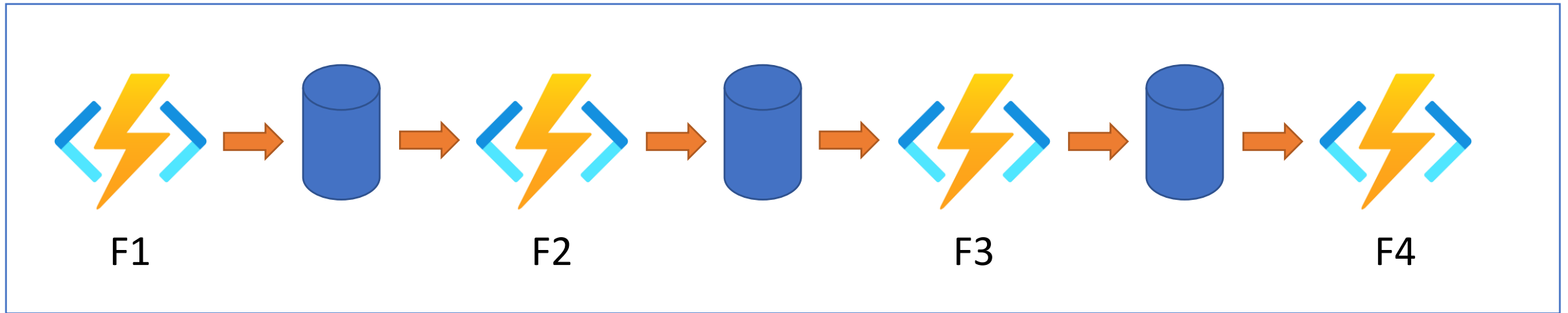


Stateful Functions?

Durable Functions



Patterns – Function Chaining





Durable Functions

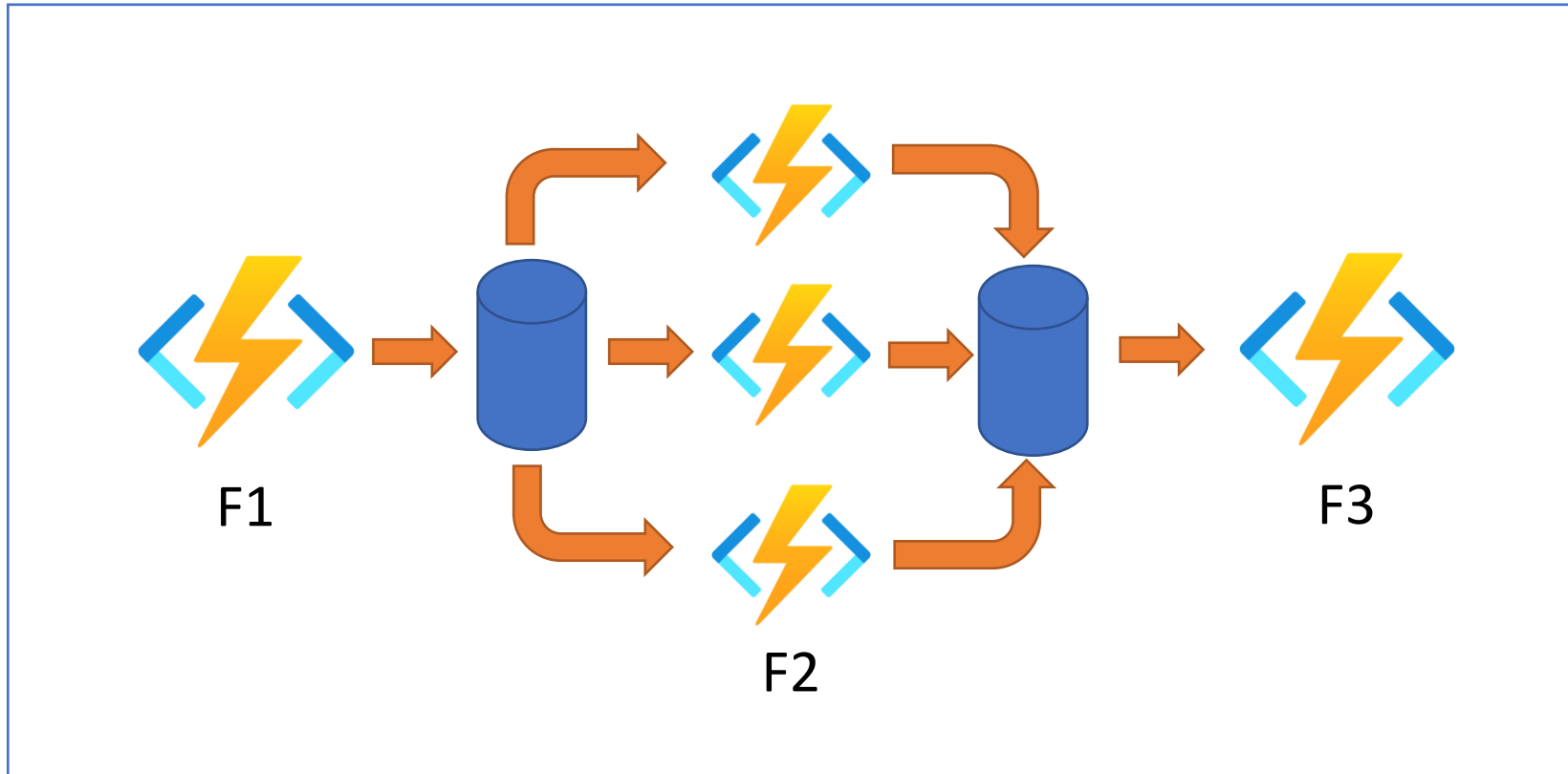
Patterns – Function Chaining

```
[FunctionName("Chaining")]
public static async Task<object> Run(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    try
    {
        var x = await context.CallActivityAsync<object>("F1", null);
        var y = await context.CallActivityAsync<object>("F2", x);
        var z = await context.CallActivityAsync<object>("F3", y);
        return await context.CallActivityAsync<object>("F4", z);
    }
    catch (Exception)
    {
        // Error handling or compensation goes here.
    }
}
```

Durable Functions



Patterns – Fan out/fan in





Durable Functions

Patterns – Fan out/fan in

```
[FunctionName("FanOutFanIn")]
public static async Task Run(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    var parallelTasks = new List<Task<int>>();

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

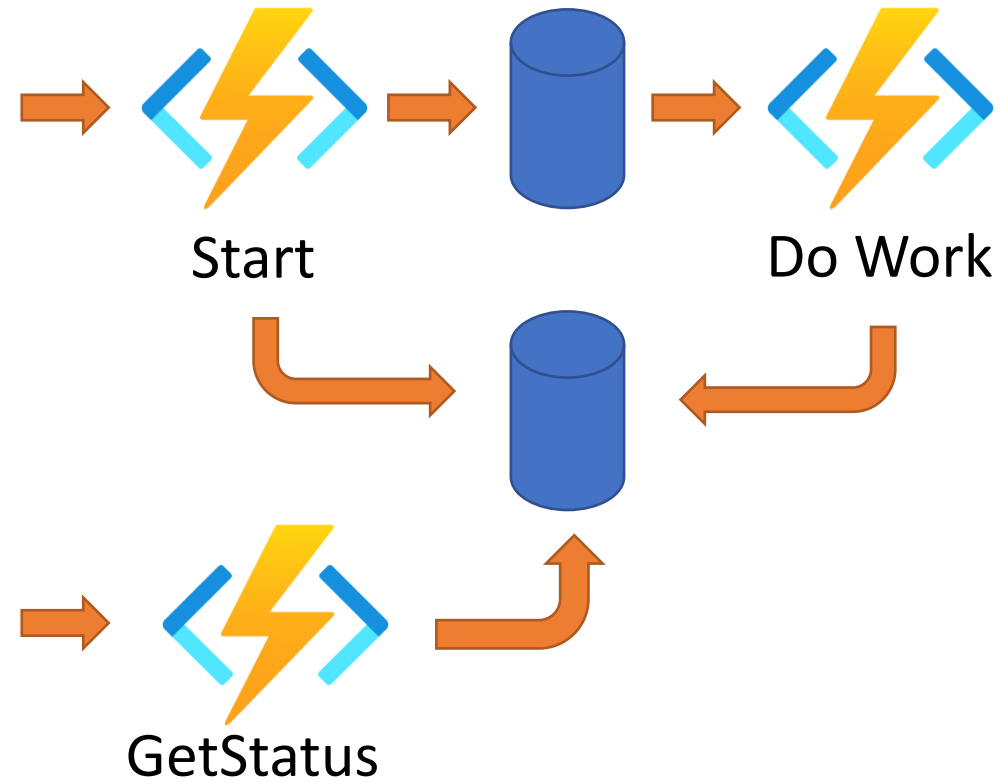
    await Task.WhenAll(parallelTasks);

    // Aggregate all N outputs and send the result to F3.
    int sum = parallelTasks.Sum(t => t.Result);
    await context.CallActivityAsync("F3", sum);
}
```



Durable Functions

Patterns – Async HTTP APIs





Durable Functions

Patterns – Async HTTP APIs

```
> curl -X POST https://myfunc.azurewebsites.net/api/orchestrators/DoWork -H "Content-Length: 0" -i
HTTP/1.1 202 Accepted
Content-Type: application/json
Location: https://myfunc.azurewebsites.net/runtime/webhooks/durabletask/instances/b79baf67f717453ca9e86c5da21e03ec

{"id":"b79baf67f717453ca9e86c5da21e03ec", ...}

> curl https://myfunc.azurewebsites.net/runtime/webhooks/durabletask/instances/b79baf67f717453ca9e86c5da21e03ec
HTTP/1.1 202 Accepted
Content-Type: application/json
Location: https://myfunc.azurewebsites.net/runtime/webhooks/durabletask/instances/b79baf67f717453ca9e86c5da21e03ec

{"runtimeStatus":"Running","lastUpdatedTime":"2019-03-16T21:20:47Z", ...}

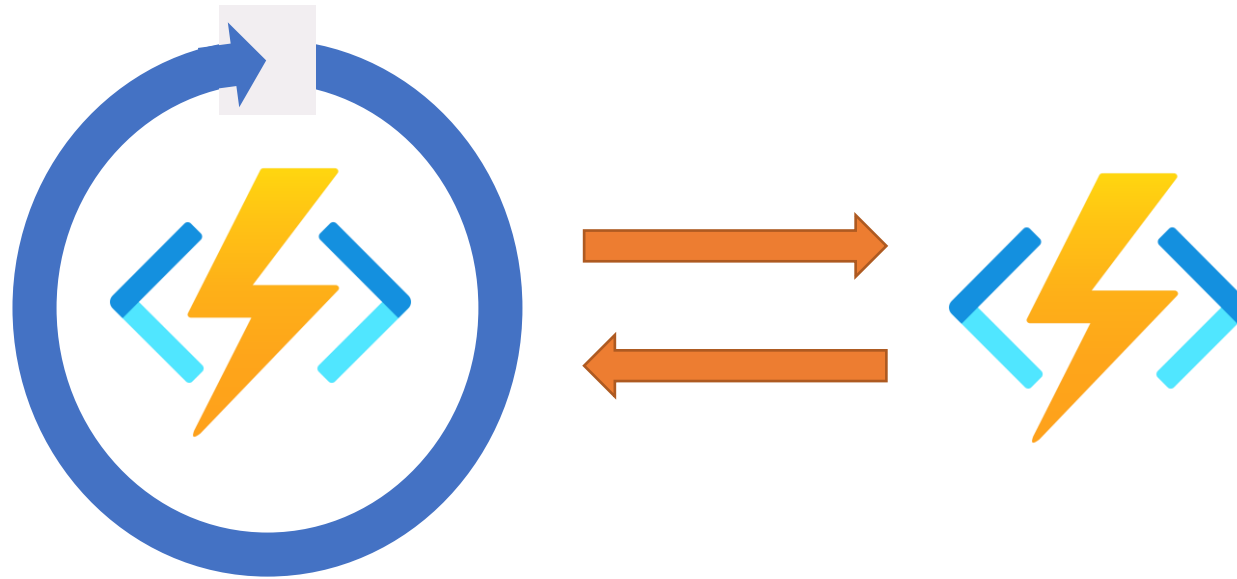
> curl https://myfunc.azurewebsites.net/runtime/webhooks/durabletask/instances/b79baf67f717453ca9e86c5da21e03ec
HTTP/1.1 200 OK
Content-Length: 175
Content-Type: application/json

{"runtimeStatus":"Completed","lastUpdatedTime":"2019-03-16T21:20:57Z", ...}
```

Durable Functions



Patterns – Monitor



Durable Functions



Patterns – Monitor

```
[FunctionName("MonitorJobStatus")]
public static async Task Run(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    int jobId = context.GetInput<int>();
    int pollingInterval = GetPollingInterval();
    DateTime expiryTime = GetExpiryTime();

    while (context.CurrentUtcDateTime < expiryTime)
    {
        var jobStatus = await context.CallActivityAsync<string>("GetJobStatus", jobId);
        if (jobStatus == "Completed")
        {
            // Perform an action when a condition is met.
            await context.CallActivityAsync("SendAlert", machineId);
            break;
        }

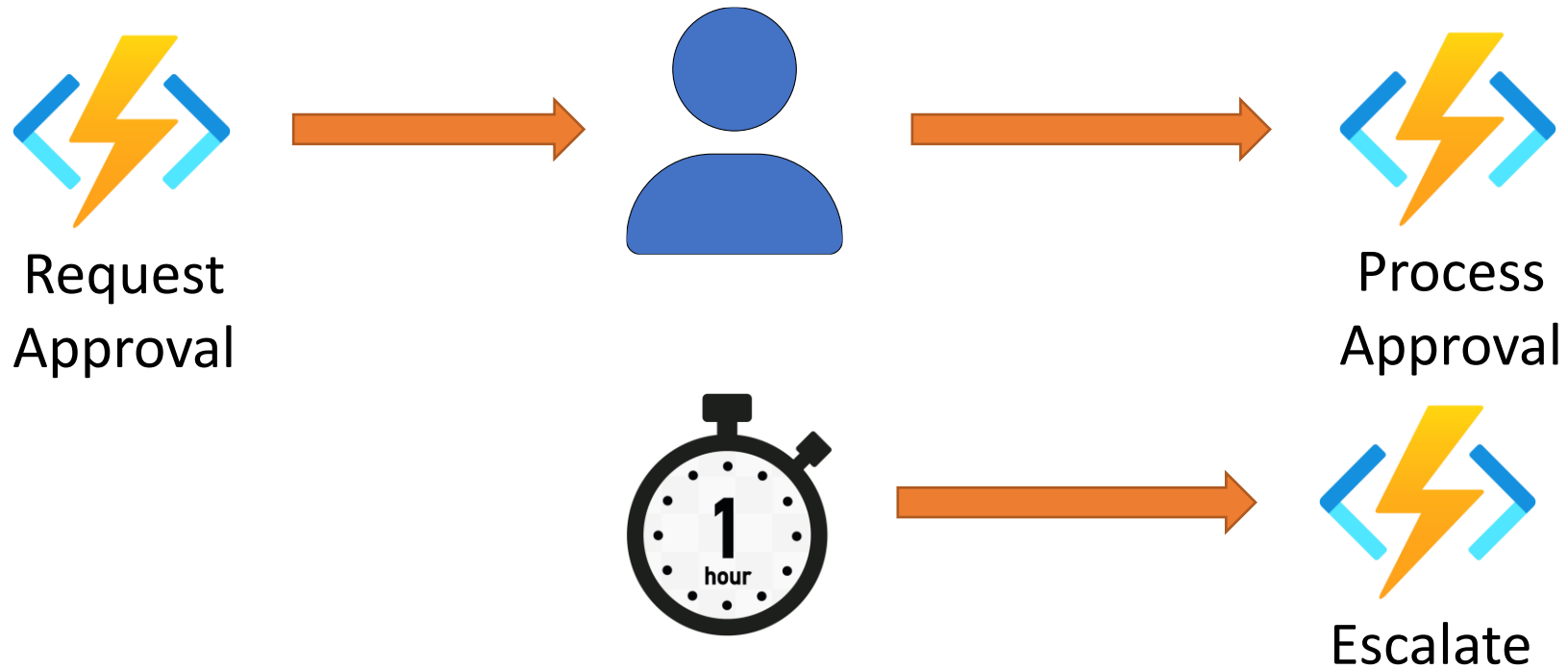
        // Orchestration sleeps until this time.
        var nextCheck = context.CurrentUtcDateTime.AddSeconds(pollingInterval);
        await context.CreateTimer(nextCheck, CancellationToken.None);
    }

    // Perform more work here, or let the orchestration end.
}
```

Durable Functions



Patterns – Human Interaction



Durable Functions



Patterns – Human Interaction

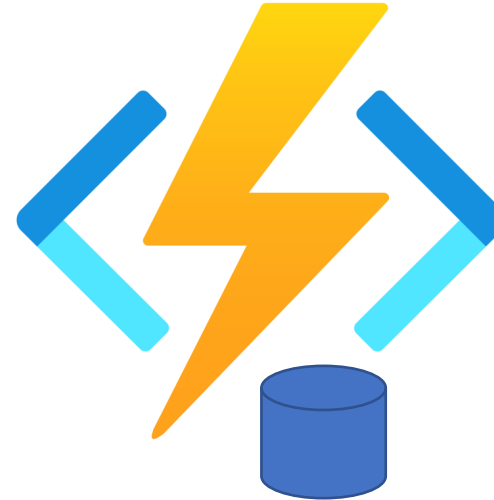
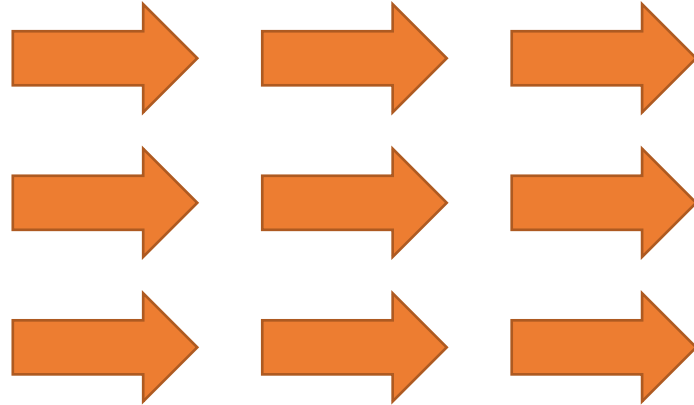
```
[FunctionName("ApprovalWorkflow")]
public static async Task Run(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    await context.CallActivityAsync("RequestApproval", null);
    using (var timeoutCts = new CancellationTokenSource())
    {
        DateTime dueTime = context.CurrentUtcDateTime.AddHours(72);
        Task durableTimeout = context.CreateTimer(dueTime, timeoutCts.Token);

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

Durable Functions



Patterns – Aggregator (stateful entities)





Durable Functions

Patterns – Aggregator (stateful entities)

```
[FunctionName("Counter")]
public static void Counter([EntityTrigger] IDurableEntityContext ctx)
{
    int currentValue = ctx.GetState<int>();
    switch (ctx.OperationName.ToLowerInvariant())
    {
        case "add":
            int amount = ctx.GetInput<int>();
            ctx.SetState(currentValue + amount);
            break;
        case "reset":
            ctx.SetState(0);
            break;
        case "get":
            ctx.Return(currentValue);
            break;
    }
}
```



Durable Functions

Patterns – Aggregator (stateful entities)

```
public class Counter
{
    [JsonProperty("value")]
    public int CurrentValue { get; set; }

    public void Add(int amount) => this.CurrentValue += amount;

    public void Reset() => this.CurrentValue = 0;

    public int Get() => this.CurrentValue;

    [FunctionName(nameof(Counter))]
    public static Task Run([EntityTrigger] IDurableEntityContext ctx)
        => ctx.DispatchAsync<Counter>();
}
```




Durable Functions

Patterns – Aggregator (stateful entities)

```
[FunctionName("EventHubTriggerCSharp")]
public static async Task Run(
    [EventHubTrigger("device-sensor-events")] EventData eventData,
    [DurableClient] IDurableEntityClient entityClient)
{
    var metricType = (string)eventData.Properties["metric"];
    var delta = BitConverter.ToInt32(eventData.Body, eventData.Body.Offset);

    // The "Counter/{metricType}" entity is created on-demand.
    var entityId = new EntityId("Counter", metricType);
    await entityClient.SignalEntityAsync(entityId, "add", delta);
}
```



Summary

- Real-World Scenarios
- Microsoft Serverless Offerings
- Azure Functions Triggers and Bindings
- Azure Durable Functions

Thank You!



- ✉ chadgreen@chadgreen.com
- 💬 TaleLearnCode
- 🌐 ChadGreen.com
- 🐦 ChadGreen & TaleLearnCode
- 🌐 ChadwickEGreen

