



# Advanced Serverless Workshop

*Advanced Serverless Workshop*

# Advanced Serverless Workshop



# Speakers



**Chad Green**

09:00 to 10:30



**Martine Dowden**

11:00 to 12:30



**Vadym Kazulkin**

13:30 to 15:00

# Panel Session



# Which cloud provider?



# Who has a serverless project currently running in production?

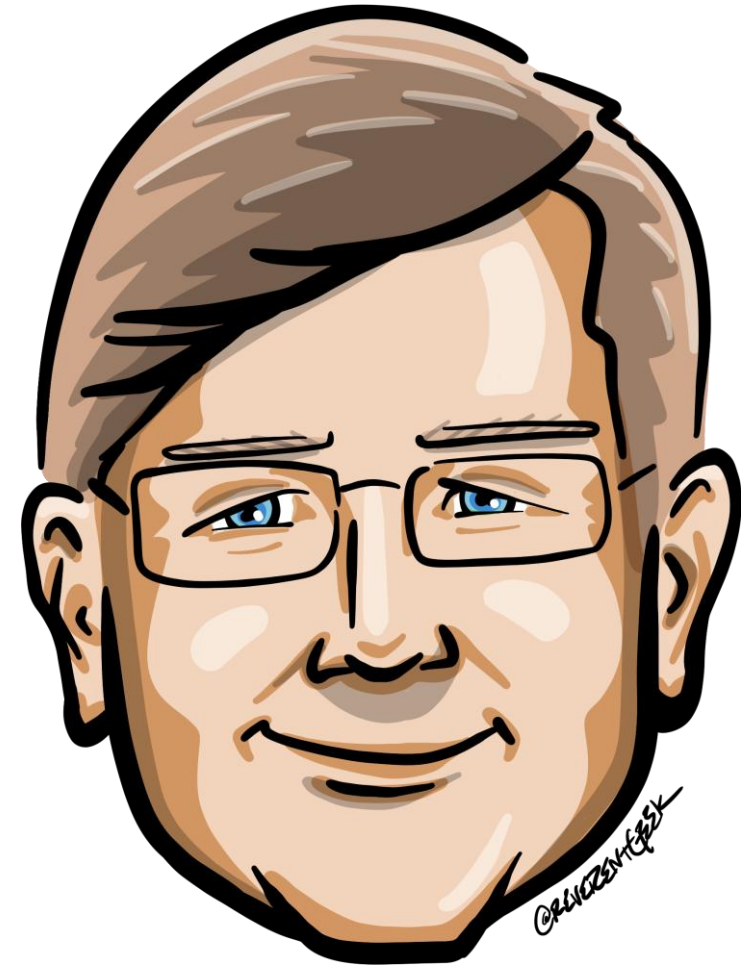


# Serverless Unleashed: From Design Patterns to Azure Solutions and Beyond

*Advanced Serverless Workshop*

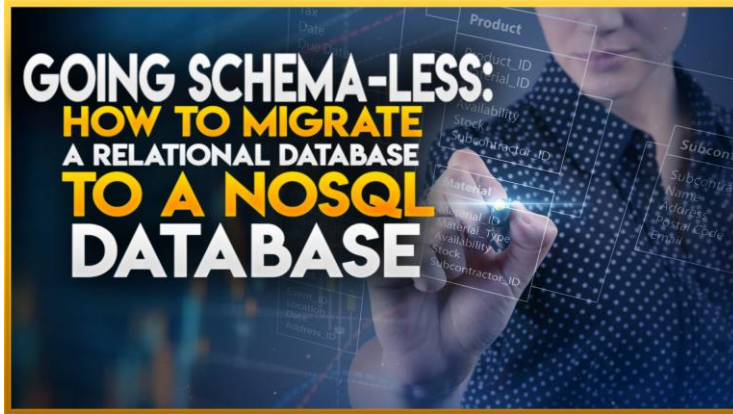


# Who is Chad Green?





# Busy Week



Tuesday @ 11:30



Tuesday @ 15:45



Wednesday @ 17:00

# Agenda

## Serverless Architecture

- Introduction to Serverless Computing
- Serverless Architecture Styles
- Cloud Design Patterns
- Software Design Patterns

# Agenda

## Serverless Architecture

## Azure Services Offerings

- Compute
- Workflow and Integration
- Data Processing and Analytics
- Messaging
- Data Storage

# Agenda

**Serverless  
Architecture**

**Azure Services  
Offerings**

**What's New  
and Coming**

- Recent updates to Azure Container Apps
- Upcoming features coming to Azure Functions



# Introduction to Serverless Computing

Serverless Unleashed: From Design Patterns to Azure Solutions and Beyond

# What is Serverless Computing

**On-Prem**

# What is Serverless Computing

IaaS

Infrastructure as a Service

PaaS

Platform as a Service

FaaS

Function as a Service



# What is Serverless Computing

IaaS

Infrastructure as a Service

PaaS

Platform as a Service

FaaS

Function as a Service

# Serverless Benefits

**Scalability**

# Serverless Benefits

**Scalability**

**Cost-Efficiency**

# Serverless Benefits

**Scalability**

**Cost-Efficiency**

**Developer  
Productivity**

# Serverless Benefits

**Scalability**

**Cost-Efficiency**

**Developer  
Productivity**

**High Availability**

# Serverless Benefits

**Scalability**

**Cost-Efficiency**

**Developer  
Productivity**

**High Availability**

**Rapid  
Development**

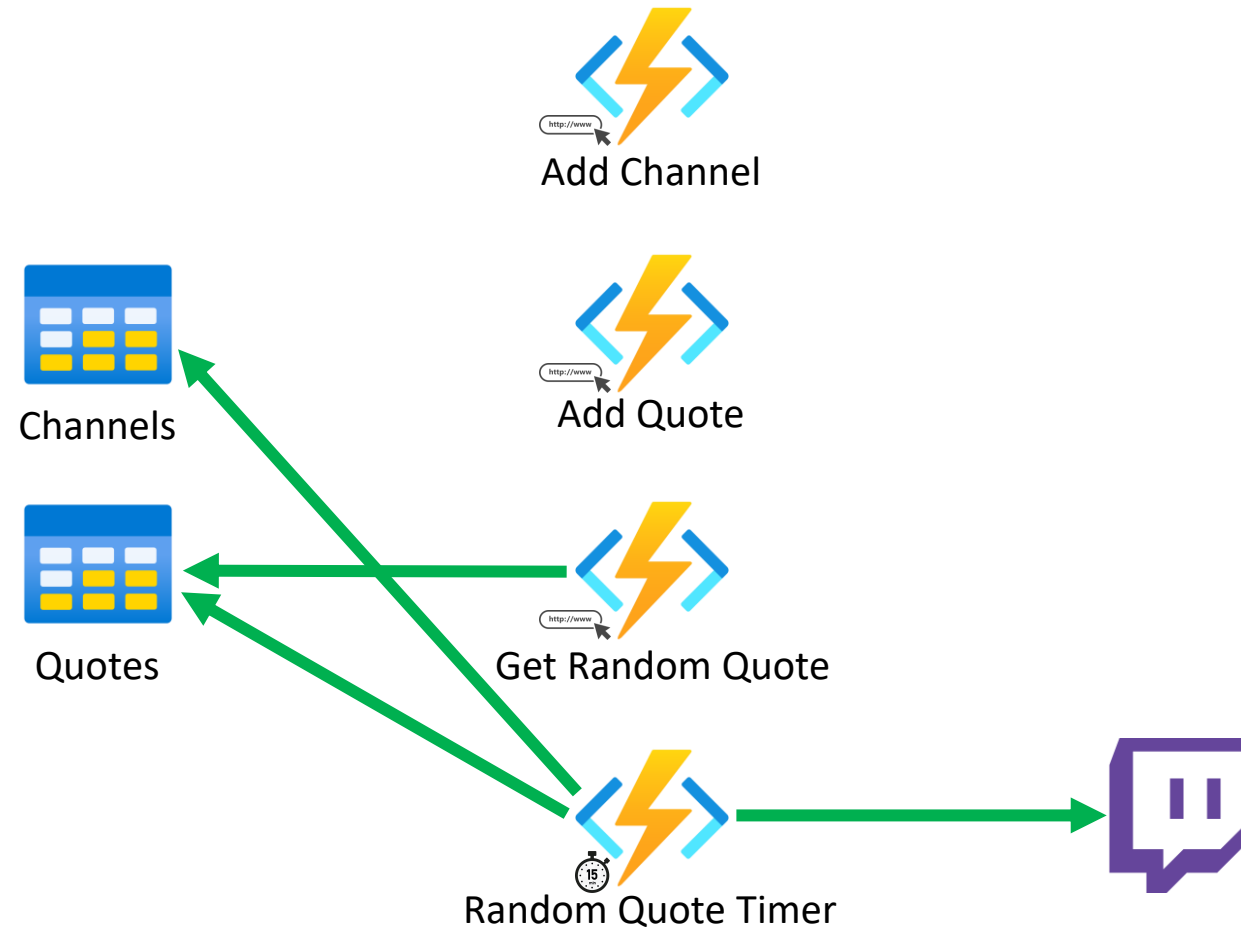


# Implemented Azure Solutions

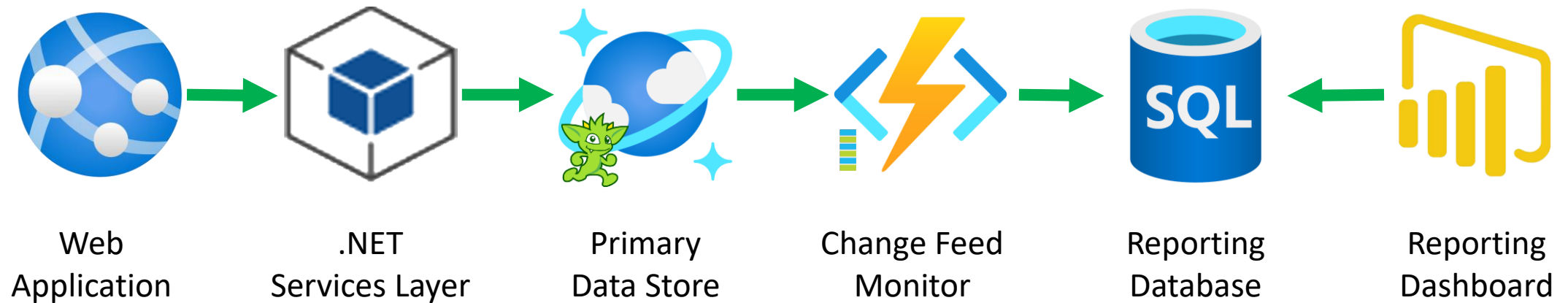
Serverless Unleashed: From Design Patterns to Azure Solutions and Beyond



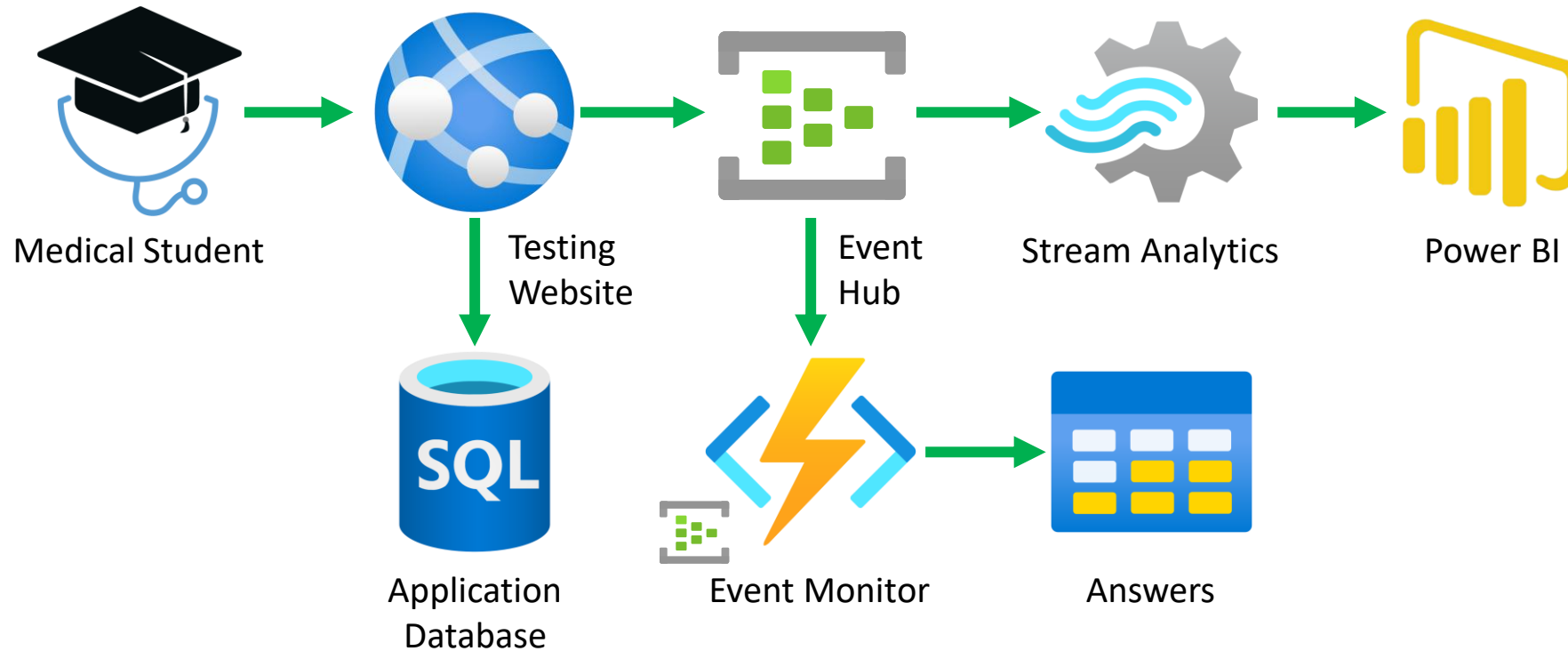
# Random Quote Generator



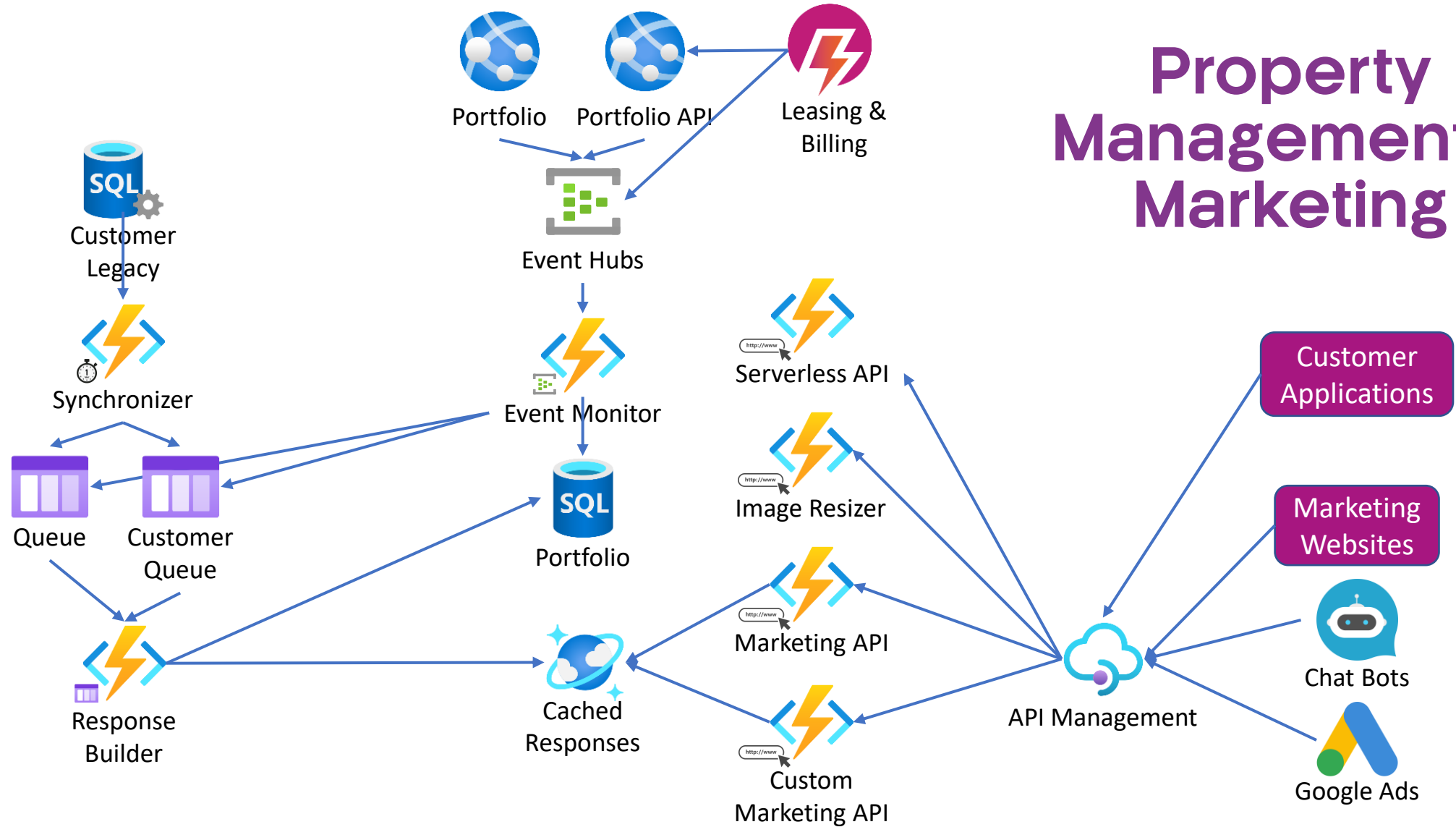
# Replicating Data



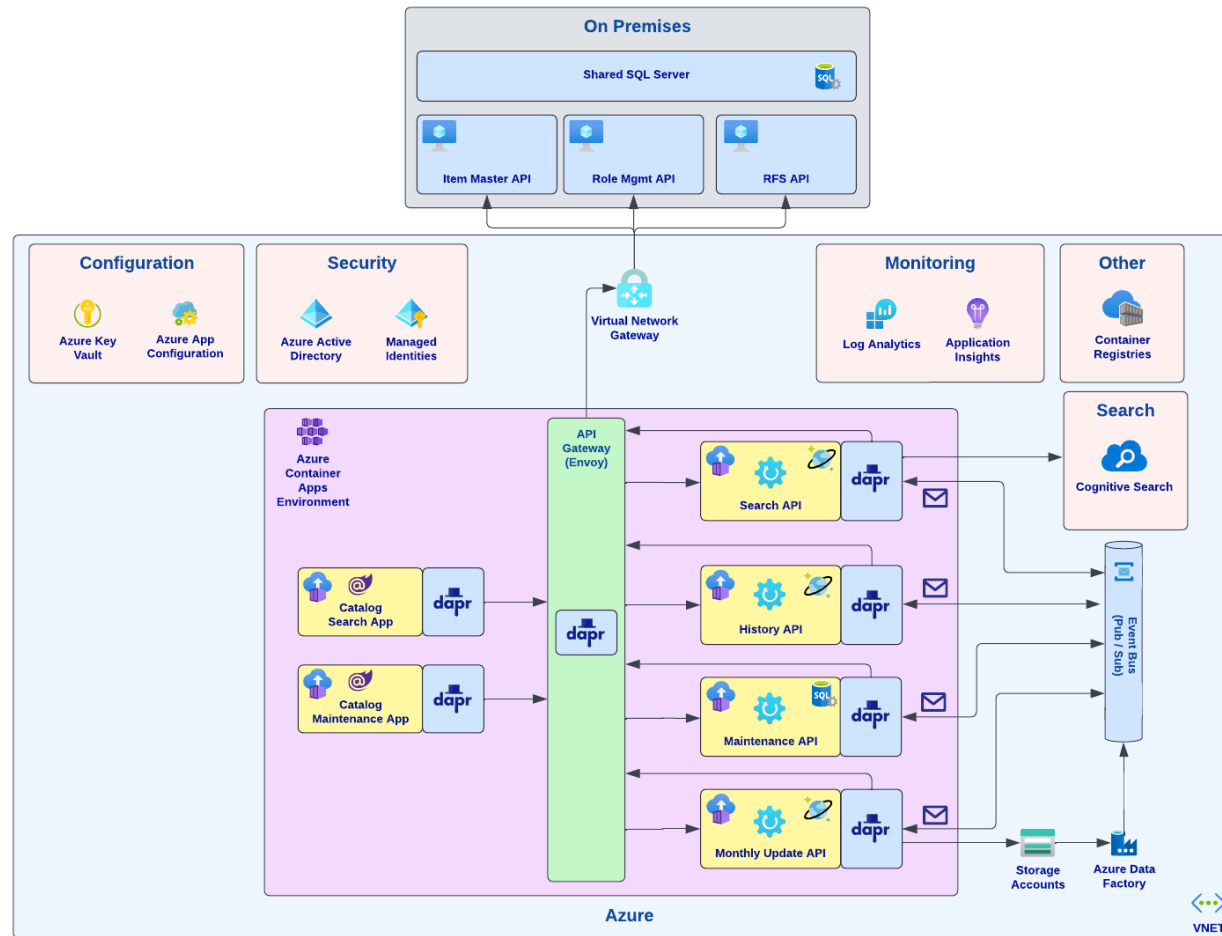
# Real-Time Reporting



# Property Management & Marketing



# Catalog Maintenance

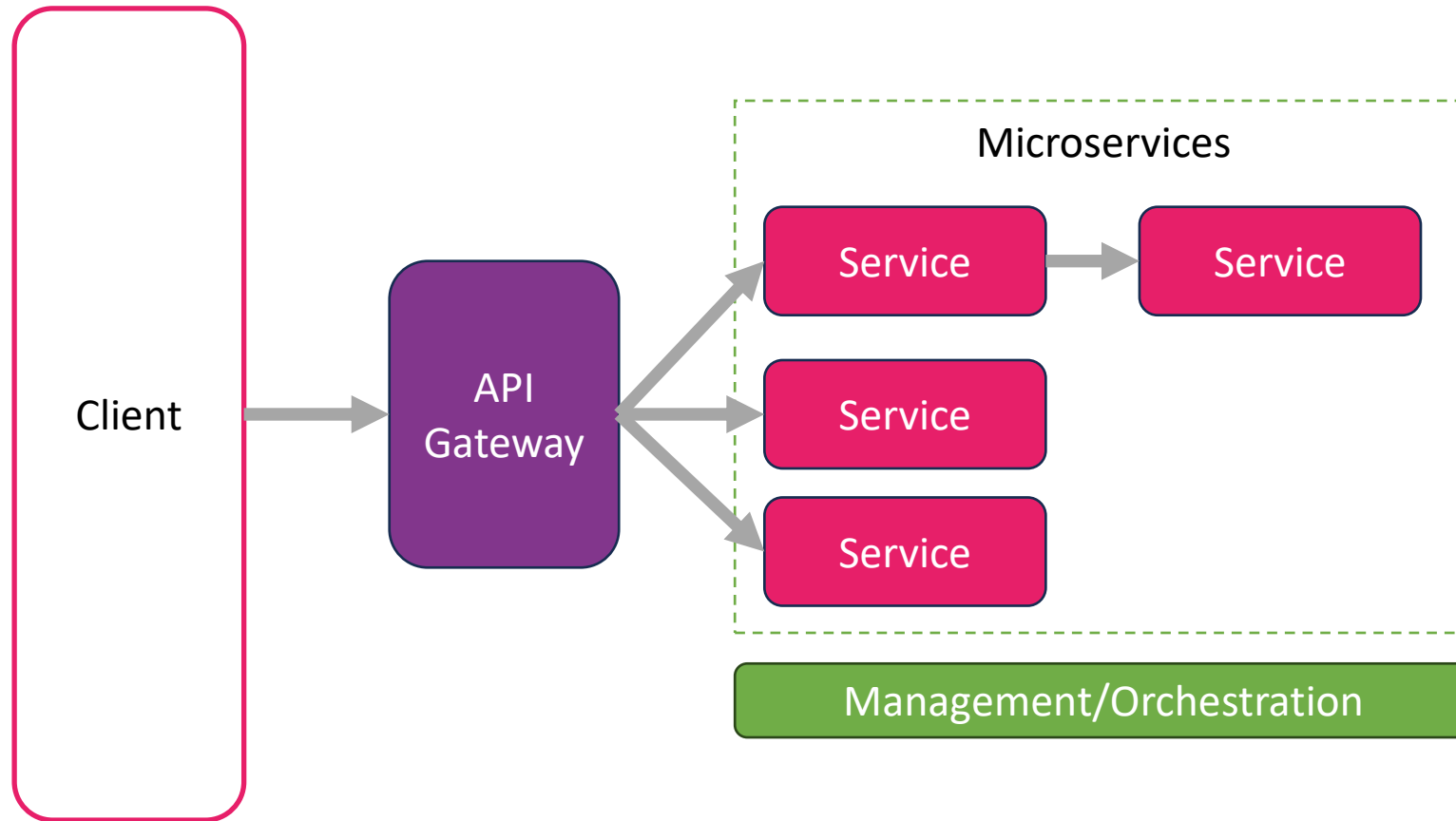




# Serverless Architecture Styles

Serverless Unleashed: From Design Patterns to Azure Solutions and Beyond

# Microservices





# Benefits of Microservices

**Flexibility**

# Benefits of Microservices

**Flexibility**

**Scalability**

# Benefits of Microservices

**Flexibility**

**Scalability**

**Resilience**

# Benefits of Microservices

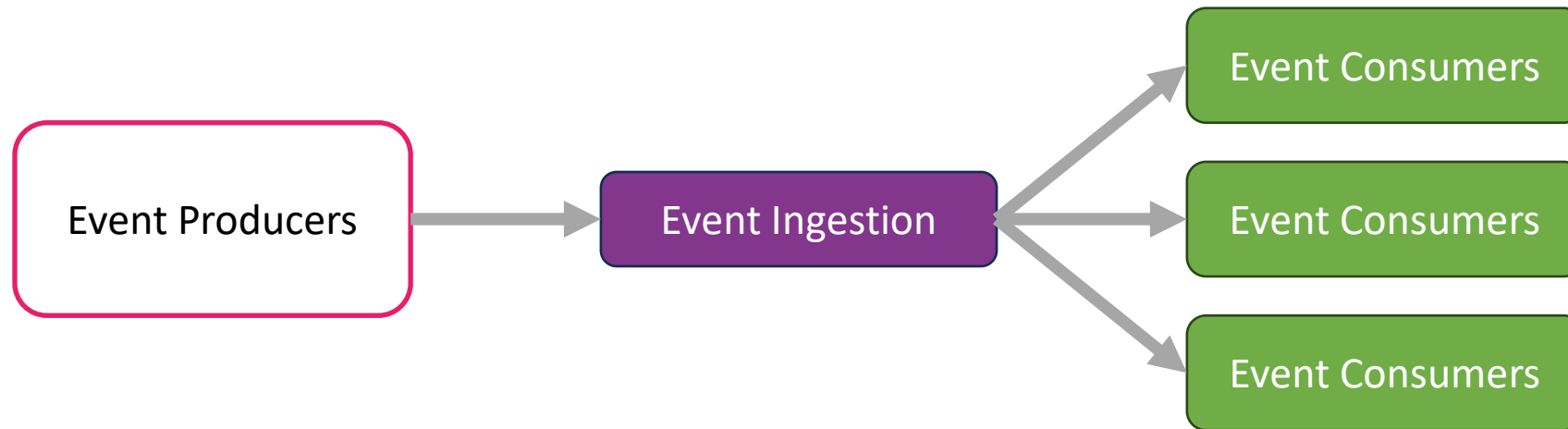
**Flexibility**

**Scalability**

**Resilience**

**Agility**

# Event-Driven Architecture



# Benefits of Event-Driven Architecture

**Scalability**

# Benefits of Event-Driven Architecture

**Scalability**

**Flexibility**



# Benefits of Event-Driven Architecture

**Scalability**

**Flexibility**

**Resilience**

# Benefits of Event-Driven Architecture

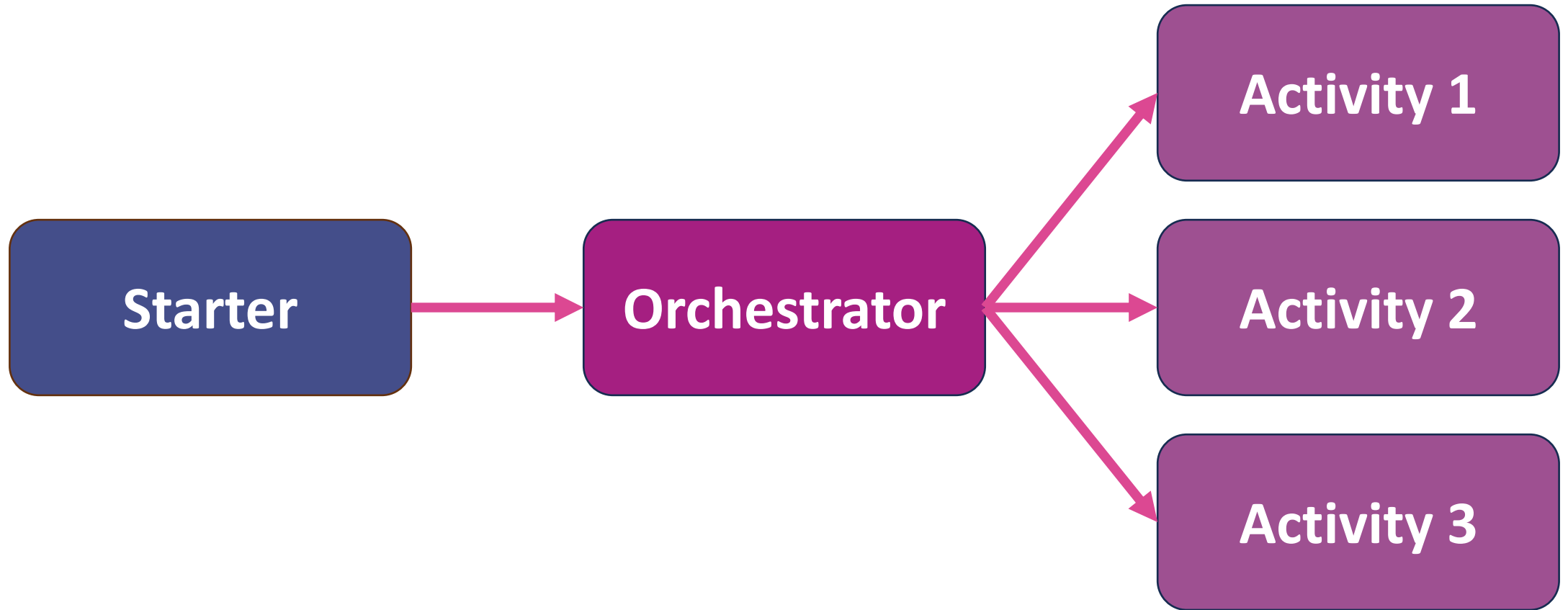
**Scalability**

**Flexibility**

**Resilience**

**Real-Time  
Processing**

# Workflow Orchestration



# Benefits of Workflow Orchestration

**Streamlined  
Business Processes**

# Benefits of Workflow Orchestration

**Streamlined  
Business Processes**

**Scalability**

# Benefits of Workflow Orchestration

**Streamlined  
Business Processes**

**Scalability**

**Flexibility**

# Benefits of Workflow Orchestration

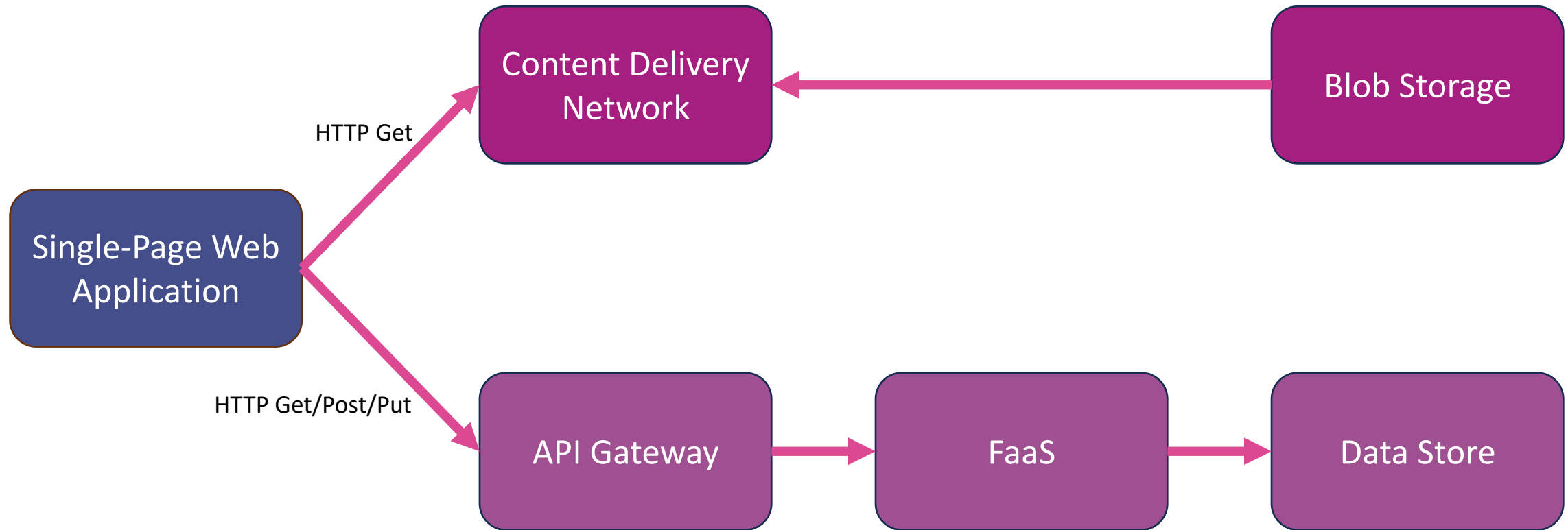
**Simplified Business Processes**

**Scalability**

**Flexibility**

**Reliability**

# Scalable Web Applications





# Benefits of Scalable Web Apps

**Automatic  
Scaling**

# Benefits of Scalable Web Apps

**Automatic  
Scaling**

**Global Reach**

# Benefits of Scalable Web Apps

**Automatic  
Scaling**

**Global Reach**

**Simplified  
Operation**

# Benefits of Scalable Web Apps

**Automatic  
Scaling**

**Global Reach**

**Simplified  
Operation**

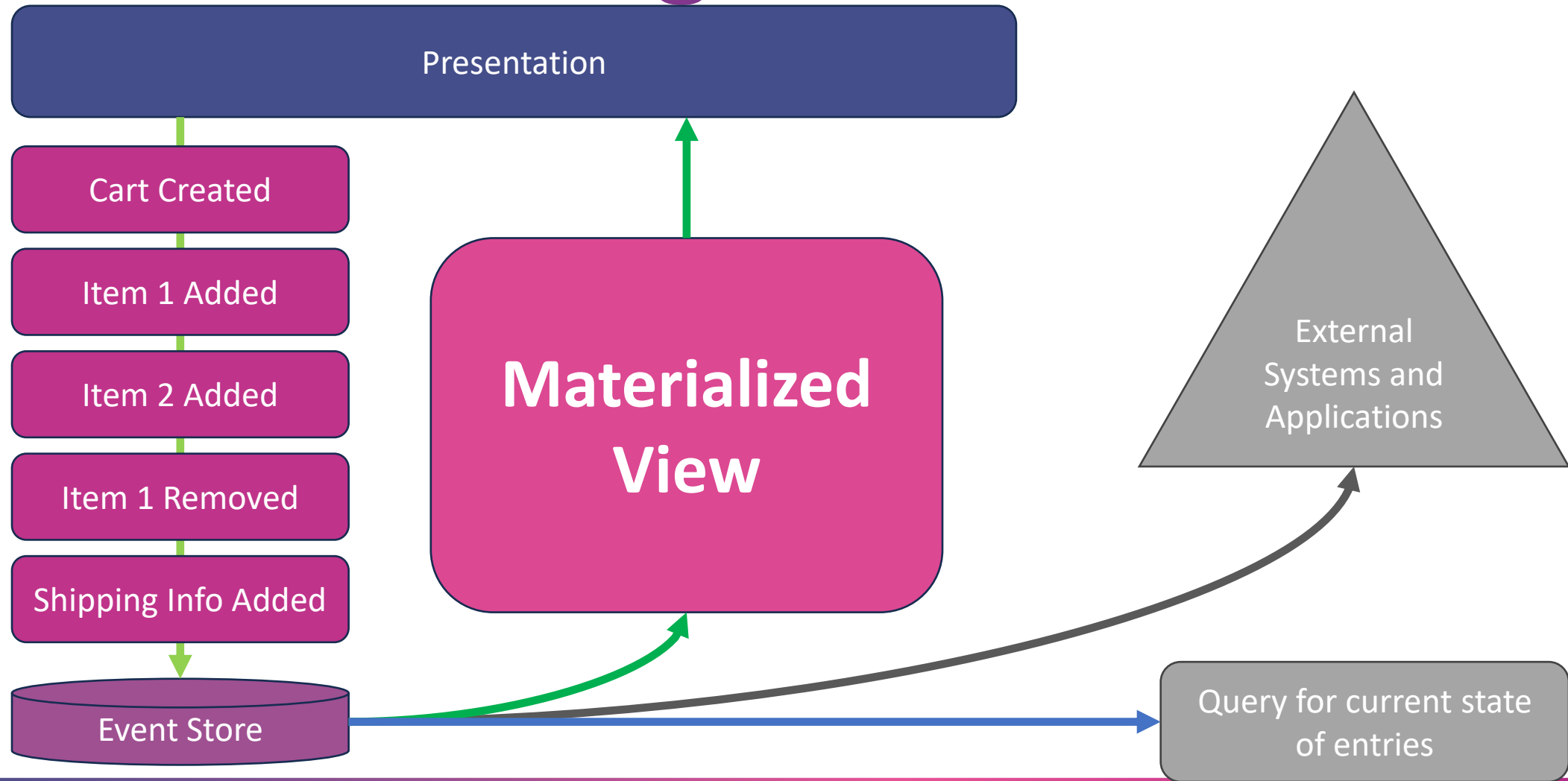
**Cost Efficiency**



# Cloud Design Patterns

Serverless Unleashed: From Design Patterns to Azure Solutions and Beyond

# Event Sourcing



# Benefits of Event Sourcing

**Immutable Audit  
Trail**

# Benefits of Event Sourcing

**Immutable Audit Trail**

**Temporal Queries**



# Benefits of Event Sourcing

**Immutable Audit Trail**

**Temporal Queries**

**Scalability and Reliability**

# Benefits of Event Sourcing

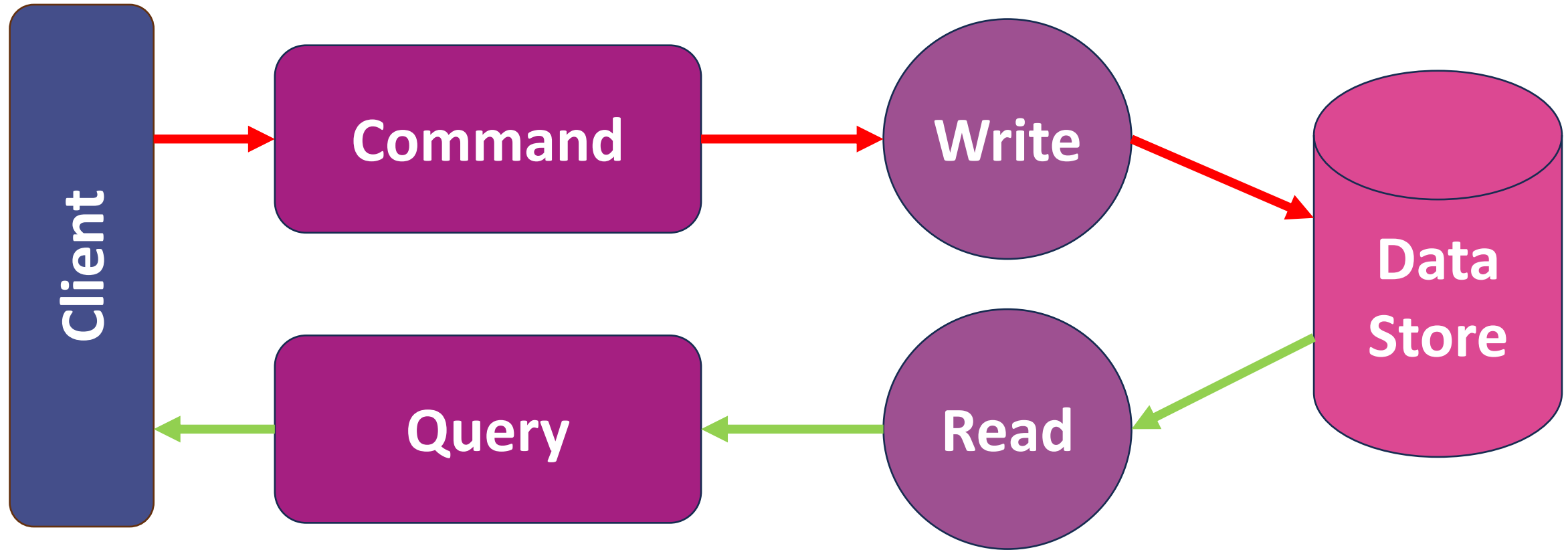
**Immutable Audit Trail**

**Temporal Queries**

**Scalability and Reliability**

**Flexibility and Extensibility**

# Command Query Responsibility Segregation (CQRS)



# Benefits of CQRS

**Scalability**

# Benefits of CQRS

**Scalability**

**Performance**

# Benefits of CQRS

**Scalability**

**Performance**

**Flexibility**

# Benefits of CQRS

**Scalability**

**Performance**

**Flexibility**

**Maintainability**

# Retry





# Benefits of Retry

**Improved Reliability**

# Benefits of Retry

**Improved Reliability**

**Fault Tolerance**

# Benefits of Retry

**Improved Reliability**

**Fault Tolerance**

**Reduced Operational  
Overhead**

# Benefits of Retry

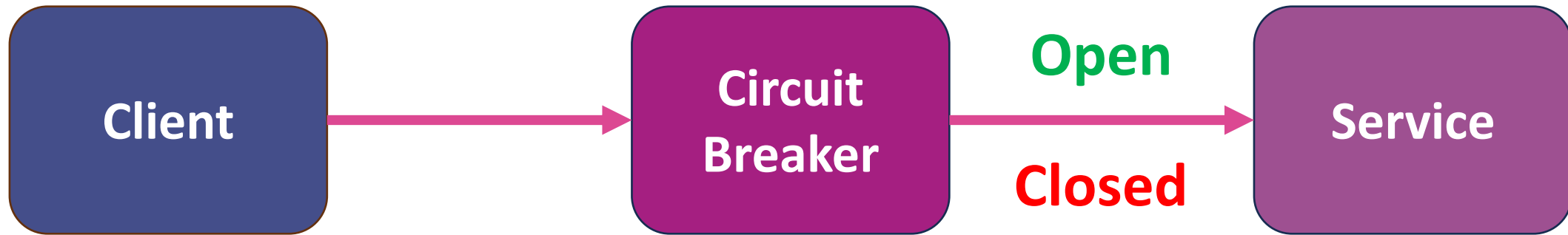
**Improved Reliability**

**Fault Tolerance**

**Reduced Operational  
Overhead**

**Enhanced User  
Experience**

# Circuit Breaker



# Benefits of Circuit Breaker

**Fault Tolerance**

# Benefits of Circuit Breaker

**Fault Tolerance**

**Load  
Management**

# Benefits of Circuit Breaker

**Fault Tolerance**

**Load  
Management**

**Improved User  
Experience**



# Benefits of Circuit Breaker

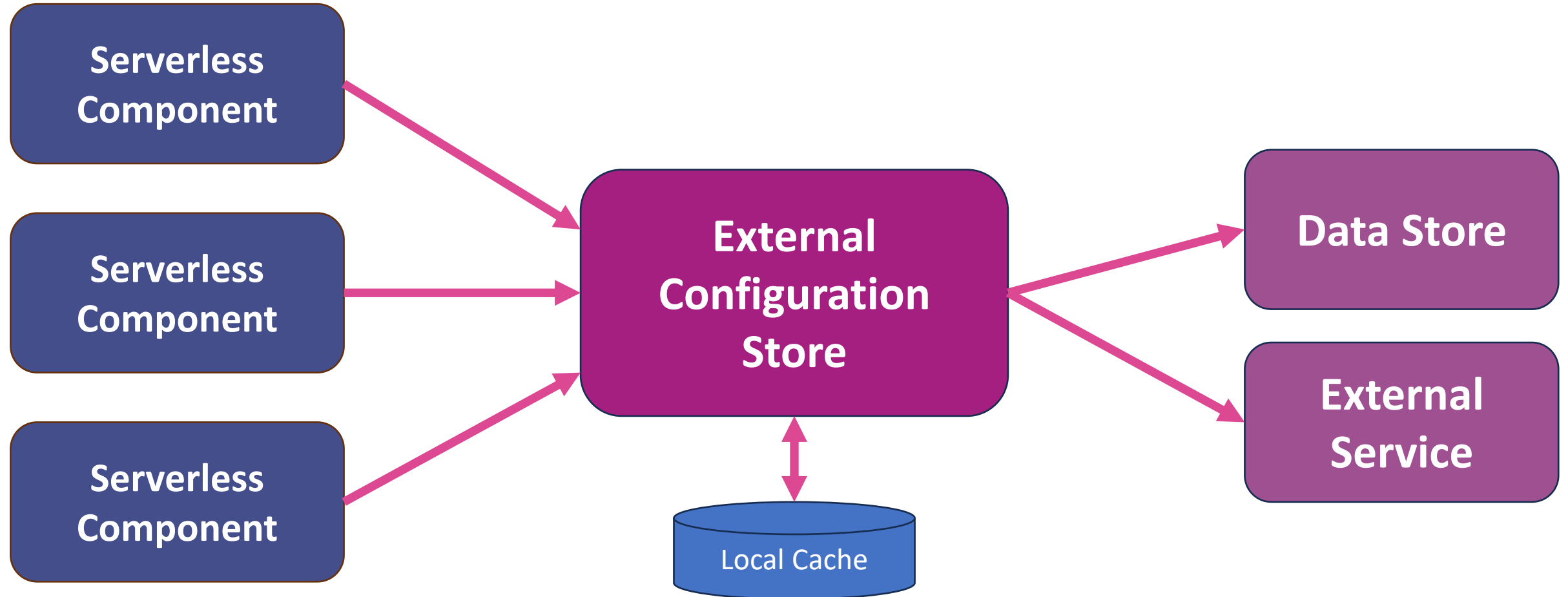
**Fault Tolerance**

**Load  
Management**

**Improved User  
Experience**

**Operational  
Insights**

# External Configuration



# Benefits of External Configuration

**Centralized  
Management**

# Benefits of External Configuration

**Centralized  
Management**

**Dynamic Configuration  
Updates**

# Benefits of External Configuration

**Centralized  
Management**

**Dynamic Configuration  
Updates**

**Enhanced Security**

# Benefits of External Configuration

**Centralized  
Management**

**Dynamic Configuration  
Updates**

**Enhanced Security**

**Scalability and  
Performance**

# Other Cloud Design Patterns

- Ambassador
- Anti-Corruption Layer
- Asynchronous Request-Retry
- Backends for Frontends
- Bulkhead
- Cache-Aside
- Choreography
- Claim Check
- Compensating Transactions
- Competing Consumers
- Compute Resource Consolidation
- Deployment Stamps
- Edge Workload Configuration
- Federated Identity
- Gatekeeper
- Gateway Aggregation
- Gateway Offloading
- Gateway Routing
- Geode
- Health Endpoint Monitoring
- Index Table
- Leader Execution
- Materialized View
- Pipes and Filters
- Priority Queue
- Publisher/Subscriber
- Queue-Based Load Balancing
- Rate Limiting
- Saga
- Scheduler Agent Supervisor
- Sequential Convoy
- Sharding
- Static Content Hosting
- Stranger Fig
- Throttling
- Valet Key

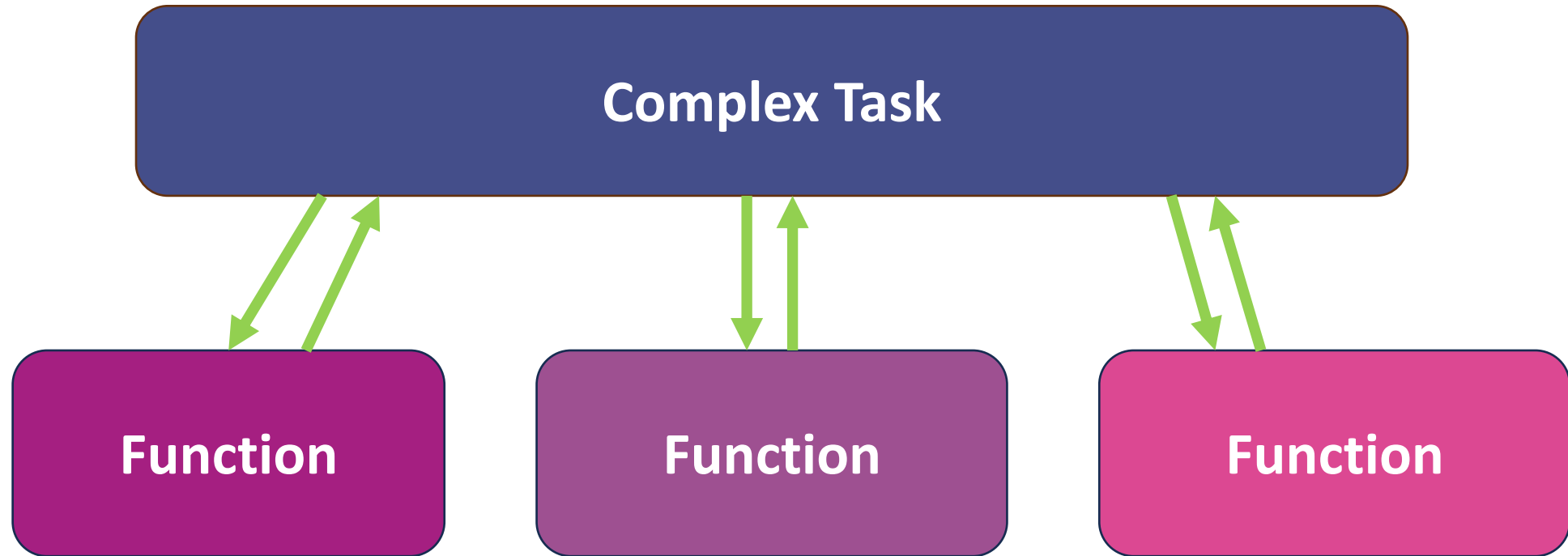


# Software Design Patterns

Serverless Unleashed: From Design Patterns to Azure Solutions and Beyond



# Function Composition



# Benefits of Function Composition

**Modularity**

# Benefits of Function Composition

**Modularity**

**Scalability**

# Benefits of Function Composition

**Modularity**

**Scalability**

**Flexibility**

# Benefits of Function Composition

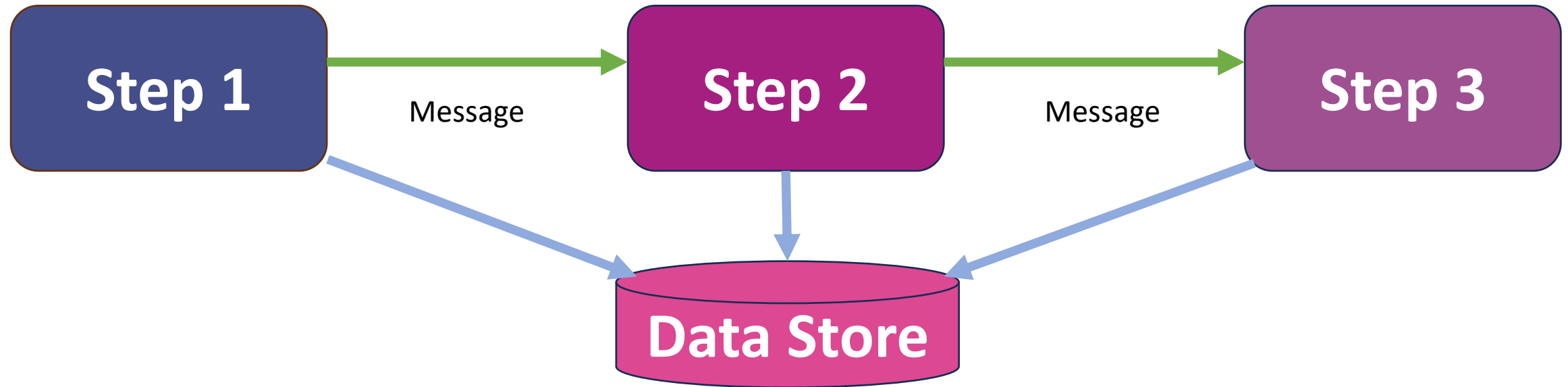
**Modularity**

**Scalability**

**Flexibility**

**Testability**

# Saga Pattern



# Benefits of Saga Pattern

**Transactional  
Integrity**

# Benefits of Saga Pattern

**Transactional  
Integrity**

**Fault Tolerance**



# Benefits of Saga Pattern

**Transactional  
Integrity**

**Fault Tolerance**

**Scalability**

# Benefits of Saga Pattern

**Transactional  
Integrity**

**Fault Tolerance**

**Scalability**

**Flexibility**

# Event Driven Pattern



# Benefits of Event-Driven Pattern

**Loose Coupling**

# Benefits of Event-Driven Pattern

**Loose Coupling**

**Scalability**

# Benefits of Event-Driven Pattern

**Loose Coupling**

**Scalability**

**Resilience**

# Benefits of Event-Driven Pattern

**Loose Coupling**

**Scalability**

**Resilience**

**Flexibility**

# Asynchronous Messaging





# Benefits of Asynchronous Messaging

**Loose Coupling**

# Benefits of Asynchronous Messaging

**Loose Coupling**

**Scalability**

# Benefits of Asynchronous Messaging

**Loose Coupling**

**Scalability**

**Reliability**

# Benefits of Asynchronous Messaging

**Loose Coupling**

**Scalability**

**Reliability**

**Flexibility**



# Microsoft Azure Serverless Services

Serverless Unleashed: From Design Patterns to Azure Solutions and Beyond

# Categories

**Compute**

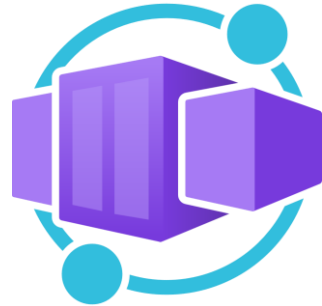
**Workflow and  
Integration**

**Data Processing and  
Analytics**

**Messaging**

**Data Storage**

# Compute

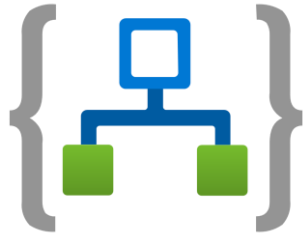


**Azure Container Apps**



**Azure Functions**

# Workflow and Integration



**Azure Logic Apps**



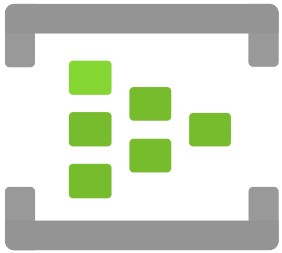
**API Management**



**Azure Event Grid**



# Data Processing and Analytics



**Azure Event Hubs**



**Azure Stream Analytics**



**Azure Synapse Analytics**

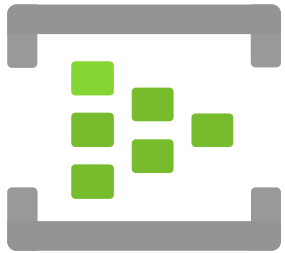


**Azure Data Lake Analytics**

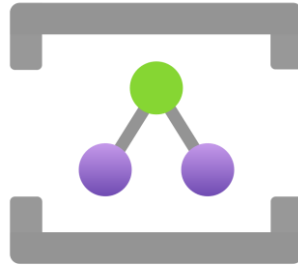
# Messaging



**Azure Service Bus**



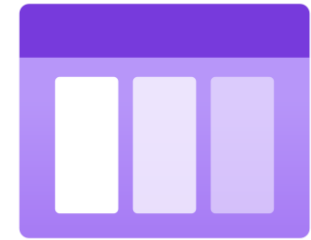
**Azure Event Hubs**



**Azure Relays**



**Azure Event Grid**



**Azure Storage  
Queues**

# Data Storage



**Azure SQL Database**



**Azure Cosmos DB**



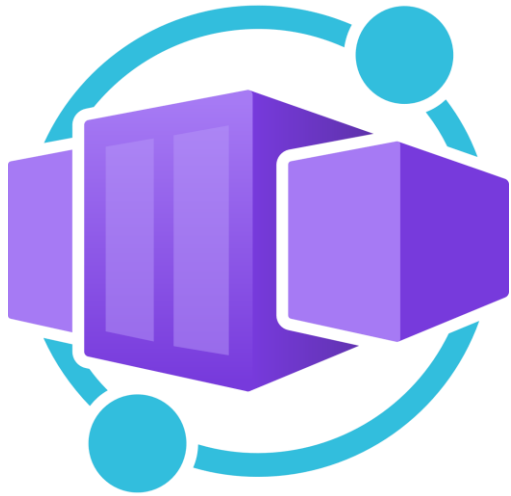
**Azure Storage**

# Azure Functions



- Asynchronous Request-Reply
- Backends for Frontends
- Compute Resource Consolidation
- Gatekeeper
- Messaging Bridge
- Pipes and Filters
- Publisher/Subscriber
- Queue-Based Load Balancing
- Retry
- Scheduler Agent Supervisor
- Throttling

# Azure Container Apps



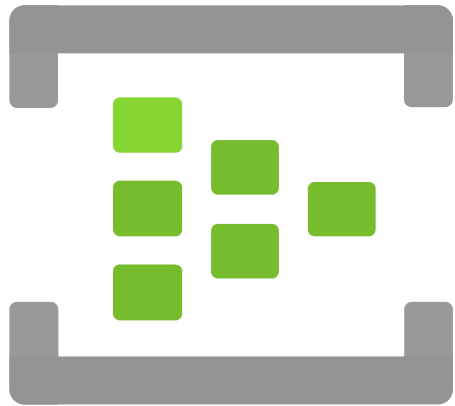
- Ambassador
- Backends for Frontends
- Bulkhead
- Circuit Breaker
- Gateway
- Gateway Aggregation
- Gateway Offloading
- Gateway Routing
- Pipes and Filters
- Sidecar
- Stranger Fig

# Azure Service Bus



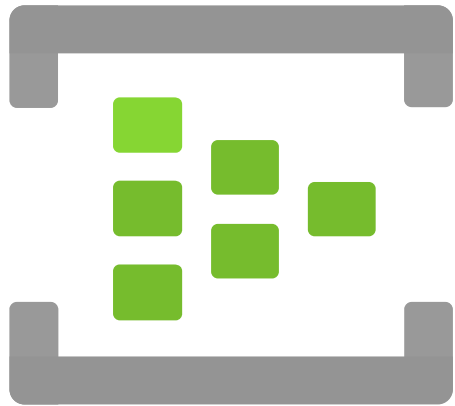
- Asynchronous Request-Reply
- Competing Consumers
- Event Sourcing
- Messaging Bridge
- Publisher/Subscriber
- Queue-Based Load Leveling
- Retry
- Saga
- Sequential Convoy

# Azure Event Hubs



- Event Sourcing
- Messaging Bridge
- Publisher/Subscriber
- Queue-Based Load Leveling

# Azure Event Grid



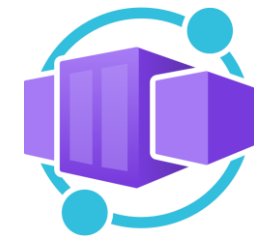
- Asynchronous Request-Reply
- Event Sourcing
- Messaging Bridge
- Publisher/Subscriber



# Azure API Management



- Ambassador
- Anti-Corruption Layer
- Backends for Frontends
- Gatekeeper
- Gateway Aggregation
- Gateway Offloading
- Gateway Routing
- Pipe and Filters
- Static Content Hosting
- Valet Key



# New and Upcoming Serverless Compute Enhancements

Serverless Unleashed: From Design Patterns to Azure Solutions and Beyond

# Azure Functions on Azure Container Apps



**Run Functions in Containers**

**Event-Driven Scaling**

**Open-Source Integration**

**Flexible Deployment**

**Monitoring and Networking**

# Why Azure Functions on Azure Container Apps?



**Integration with  
Kubernetes**

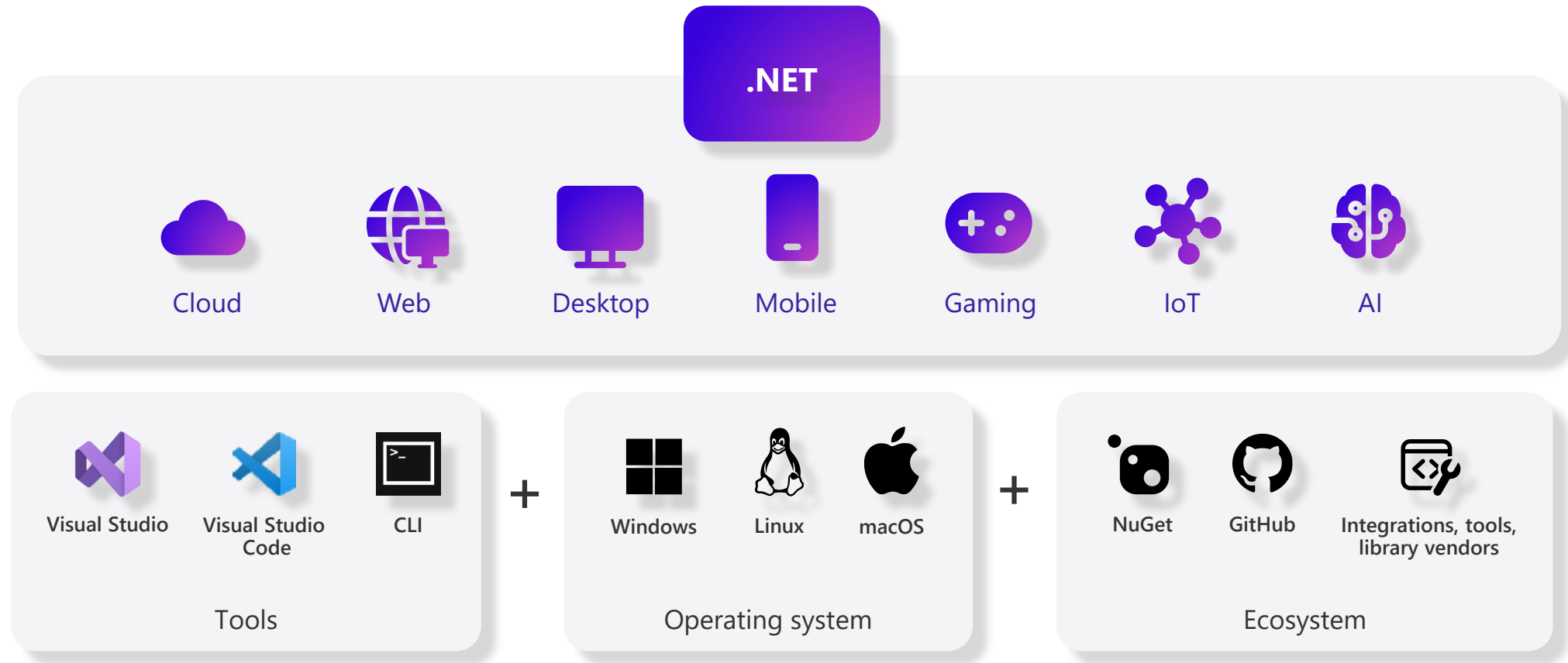
**Event-Driven  
Autoscaling**

**Open-Source  
Tools**

**Cost Efficiency**

**Flexible  
Deployment**

# Building anything with a unified platform



# What every app needs



## Every App Needs



Observability



Resiliency



Scalability



Manageability

# .NET 8 Includes



## Observability

Built in metrics with dimensions

DI integration for metrics

**Better Logging support**  
(faster, can object serialization)

Enrichment

Redaction

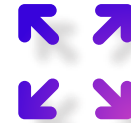
Testing fakes for Logging & Metrics



## Resiliency

New Polly based  
resiliency packages

SignalR Stateful Reconnect



## Scalability

**AOT**  
(increased density)

Performance

Chiseled Ubuntu



## Manageability

Certificate auto-rotation  
support in Kestrel

# It's still not easy



## It's Still Not Easy 🙄



Complex



Getting Started



Choices



Paved Path





# .NET Aspire

**A cloud ready stack for building observable,  
production ready, distributed applications**



# .NET Aspire

**A cloud ready stack for building observable,  
production ready, distributed applications**

Smart Defaults

Developer Dashboard

Orchestration

Service Discovery

Integrations

Deployment



# .NET Aspire

**A cloud ready stack for building observable,  
production ready, distributed applications**

**Smart Defaults**

Developer Dashboard

Orchestration

Service Discovery

Integrations

Deployment



## .NET Aspire Service Defaults



Observability



Resiliency



Health Checks



# .NET Aspire

**A cloud ready stack for building observable,  
production ready, distributed applications**

Smart Defaults

Developer Dashboard

Orchestration

Service Discovery

Integrations

Deployment



# .NET Aspire



## Developer Dashboard



Structured Logs



Metrics



Distributed Traces



Dependencies

The screenshot shows the .NET Aspire Developer Dashboard interface. The title bar reads 'AspireApp'. On the left is a sidebar with icons for Resources, Console, Structured, Traces, and Metrics. The main area is titled 'Resources' and contains a table with the following data:

| Type      | Name        | State   | Start time  | Source                     | Endpoints                 | Logs                 | Details              |
|-----------|-------------|---------|-------------|----------------------------|---------------------------|----------------------|----------------------|
| Container | cache       | Running | 12:48:23 PM | redis:7.2.4                | tcp://localhost:53683     | <a href="#">View</a> | <a href="#">View</a> |
| Project   | apiservice  | Running | 12:48:23 PM | AspireApp.ApiService.cs... | +2                        | <a href="#">View</a> | <a href="#">View</a> |
| Project   | webfrontend | Running | 12:48:23 PM | AspireApp.Web.csproj       | https://localhost:7234 +1 | <a href="#">View</a> | <a href="#">View</a> |



# .NET Aspire

**A cloud ready stack for building observable,  
production ready, distributed applications**

Smart Defaults

Developer Dashboard

**Orchestration**

Service Discovery

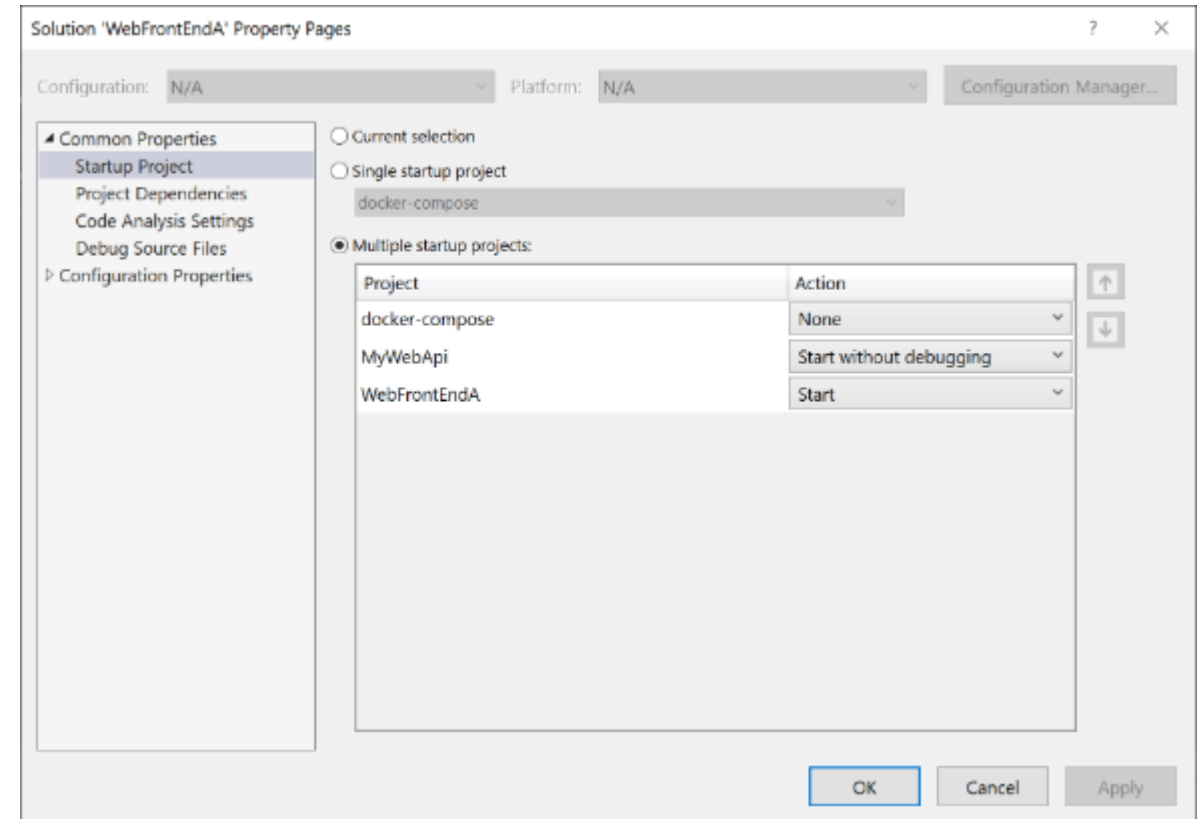
Integrations

Deployment

# Orchestration Before



```
{
  "version": "0.2.0",
  "compounds": [
    {
      "name": "Run all",
      "configurations": [
        "Run products",
        "Run store",
      ]
    }
  ],
  "configurations": [
    {
      "name": "Run products",
      "type": "dotnet",
      "request": "launch",
      "projectPath": "${workspaceFolder}\\Products\\Products.csproj"
    },
    {
      "name": "Run store",
      "type": "dotnet",
      "request": "launch",
      "projectPath": "${workspaceFolder}\\Store\\Store.csproj"
    }
  ]
}
```





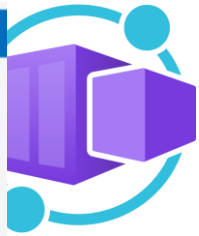
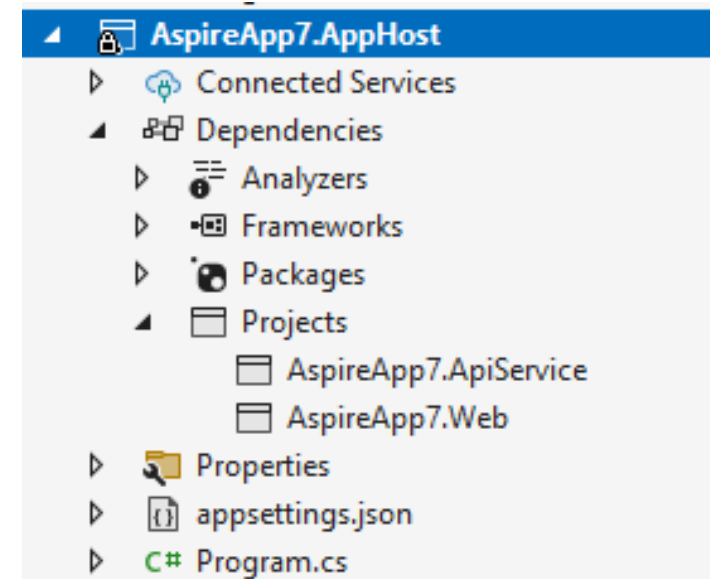
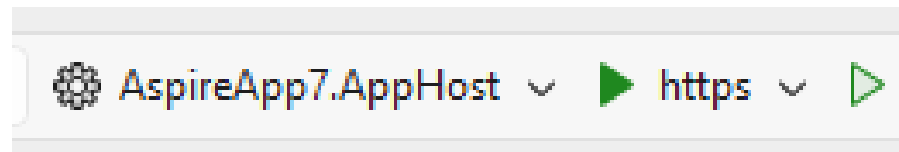
# Orchestration with .NET Aspire





## Orchestration


```
var builder = DistributedApplication.CreateBuilder(args);  
  
builder.AddProject<Projects.ApiService>("apiservice");  
  
builder.AddProject<Projects.Web>("webfrontend");  
  
builder.Build().Run();
```


# After with .NET Aspire





 AspireApp7

 Resources

 Console

 Structured

 Traces

 Metrics

## Resources

Filter...

| Type    | Name        | State   | Start time | Source                 | Endpoints                                                        | Logs | Details |
|---------|-------------|---------|------------|------------------------|------------------------------------------------------------------|------|---------|
| Project | apiservice  | Running | 2:34:57 PM | AspireApp7.ApiServi... | +2                                                               | View | View    |
| Project | webfrontend | Running | 2:34:57 PM | AspireApp7.Web.cs...   | <a href="https://localhost:7107">https://localhost:7107</a> , +1 | View | View    |



# .NET Aspire

**A cloud ready stack for building observable,  
production ready, distributed applications**

Smart Defaults

Developer Dashboard

Orchestration

**Service Discovery**

Integrations

Deployment

# Service Discovery Before



```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ProductEndpoint": "http://localhost:5228",
  "ProductEndpointHttps": "https://localhost:7130"
}
```

```
builder.Services.AddHttpClient<ProductService>(c =>
{
    var url = builder.Configuration["ProductEndpoint"] ?? throw new
        InvalidOperationException("ProductEndpoint is not set");

    c.BaseAddress = new(url);
});
```

## Service Discovery



```
var builder = DistributedApplication.CreateBuilder(args);

var apiService = builder.AddProject<Projects.ApiService>("apiservice");

builder.AddProject<Projects.Web>("webfrontend")
    .WithReference(apiService);

builder.Build().Run();
```

```
builder.Services.AddHttpClient<WeatherApiClient>(client =>
{
    client.BaseAddress = new("https+http://apiservice");
});
```



# .NET Aspire

**A cloud ready stack for building observable,  
production ready, distributed applications**

Smart Defaults

Developer Dashboard

Orchestration

Service Discovery

**Integrations**

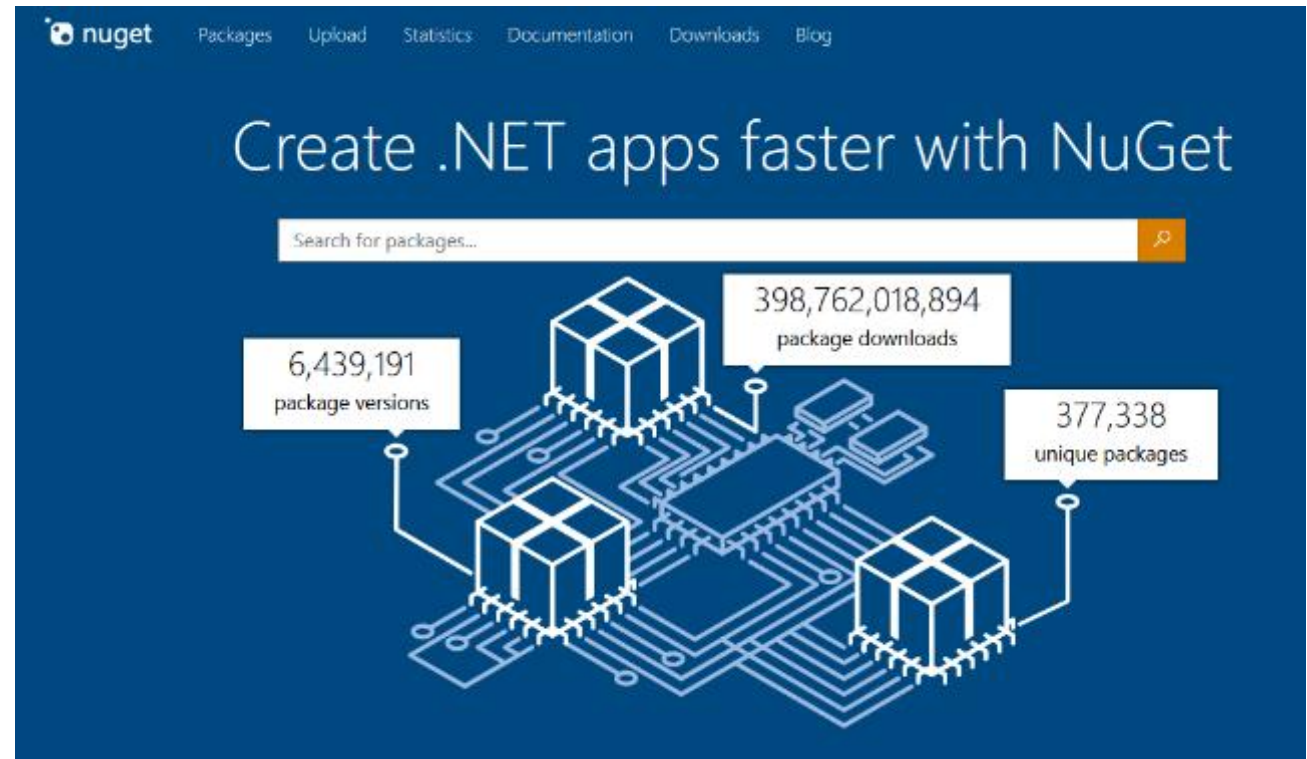
Deployment



# .NET Aspire



## Integrations





# .NET Aspire

Automatically will pull down container image and start it in Docker or Podman!

## Hosting Integration

```
var builder = DistributedApplication.CreateBuilder(args);  
  
var postgres = builder.AddPostgres("postgres").WithPgAdmin();  
var db = postgres.AddDatabase("db");  
  
var cache = builder.AddRedis("cache");  
  
var apiService = builder.AddProject<Projects.ApiService>("apiservice")  
    .WithReference(db);  
  
builder.AddProject<Projects.Web>("webfrontend")  
    .WithReference(apiService)  
    .WithReference(cache);  
  
builder.Build().Run();
```





# .NET Aspire



## Client Integration

```
var builder = WebApplication.CreateBuilder(args);
```

```
builder.AddServiceDefaults();  
builder.AddRedisOutputCache("cache");
```

```
builder.Services.AddRazorComponents()  
    .AddInteractiveServerComponents();
```

```
var app = builder.Build();
```

```
app.UseOutputCache();
```

```
app.MapRazorComponents<App>()  
    .AddInteractiveServerRenderMode();
```

```
app.MapDefaultEndpoints();
```

```
app.Run();
```

Configures service with  
retries, corresponding  
health checks, logging,  
and telemetry!!!!



# .NET Aspire

**A cloud ready stack for building observable,  
production ready, distributed applications**

Smart Defaults

Developer Dashboard

Orchestration

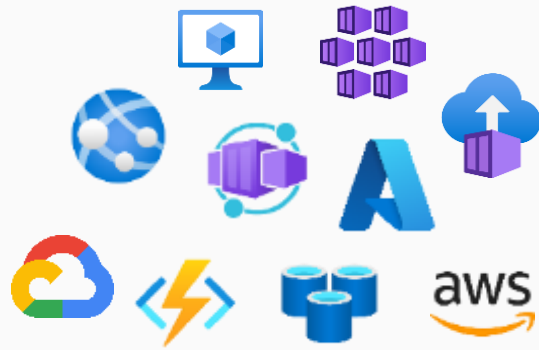
Service Discovery

Integrations

**Deployment**



## Flexible Integrations & Deployment



How You Do  
It Today!



AWS CDK



Azure Dev CLI



Visual Studio



# New and Upcoming Serverless Compute Enhancements

Serverless Unleashed: From Design Patterns to Azure Solutions and Beyond

# Many Hosting Options



## Consumption



Serverless

## App Service Environment



Network Isolation

## Premium



Sort of Serverless

## Azure Stack



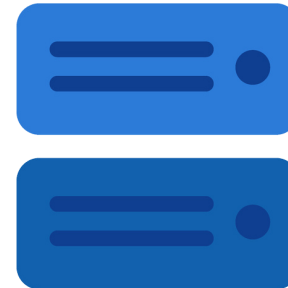
On Premises

## Dedicated



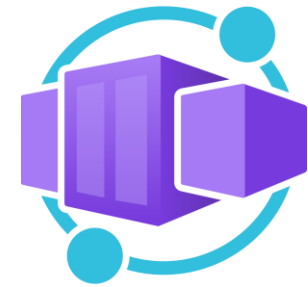
App Service Plan

## Functions Runtime



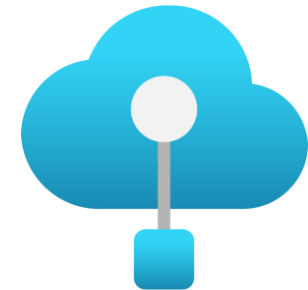
Your Server

## Container Apps



Consumption,  
Dedicated

## Azure IoT Edge



On Devices

# Many Hosting Options



Consumption

Premium

Dedicated

Container Apps

App Service Plan

Azure Stack

Functions Runtime

Azure IoT Edge



# #1 Issue in Serverless: Cold Starts





# #1 Issue in Serverless: Cold Starts





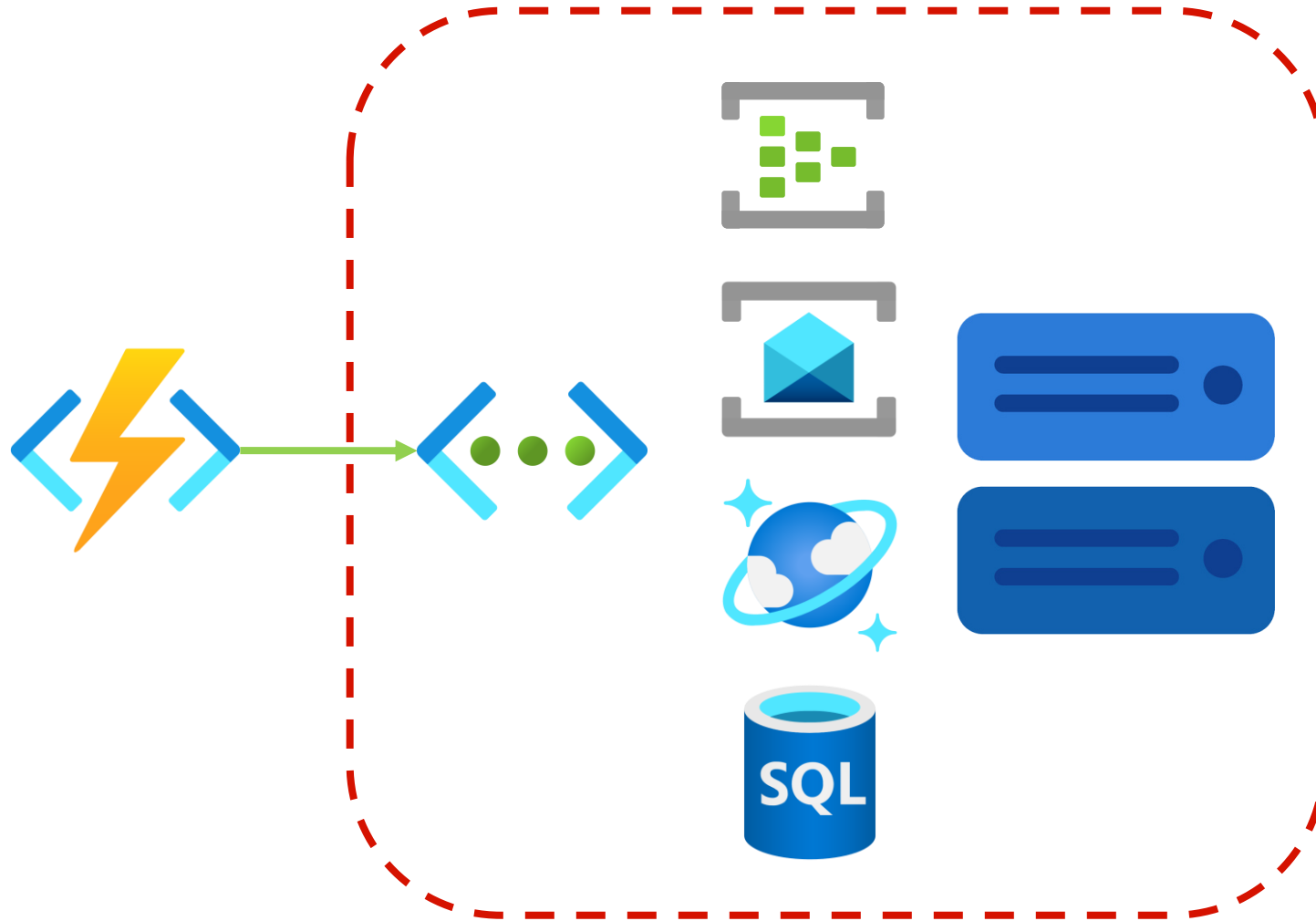


# Always Ready Instances

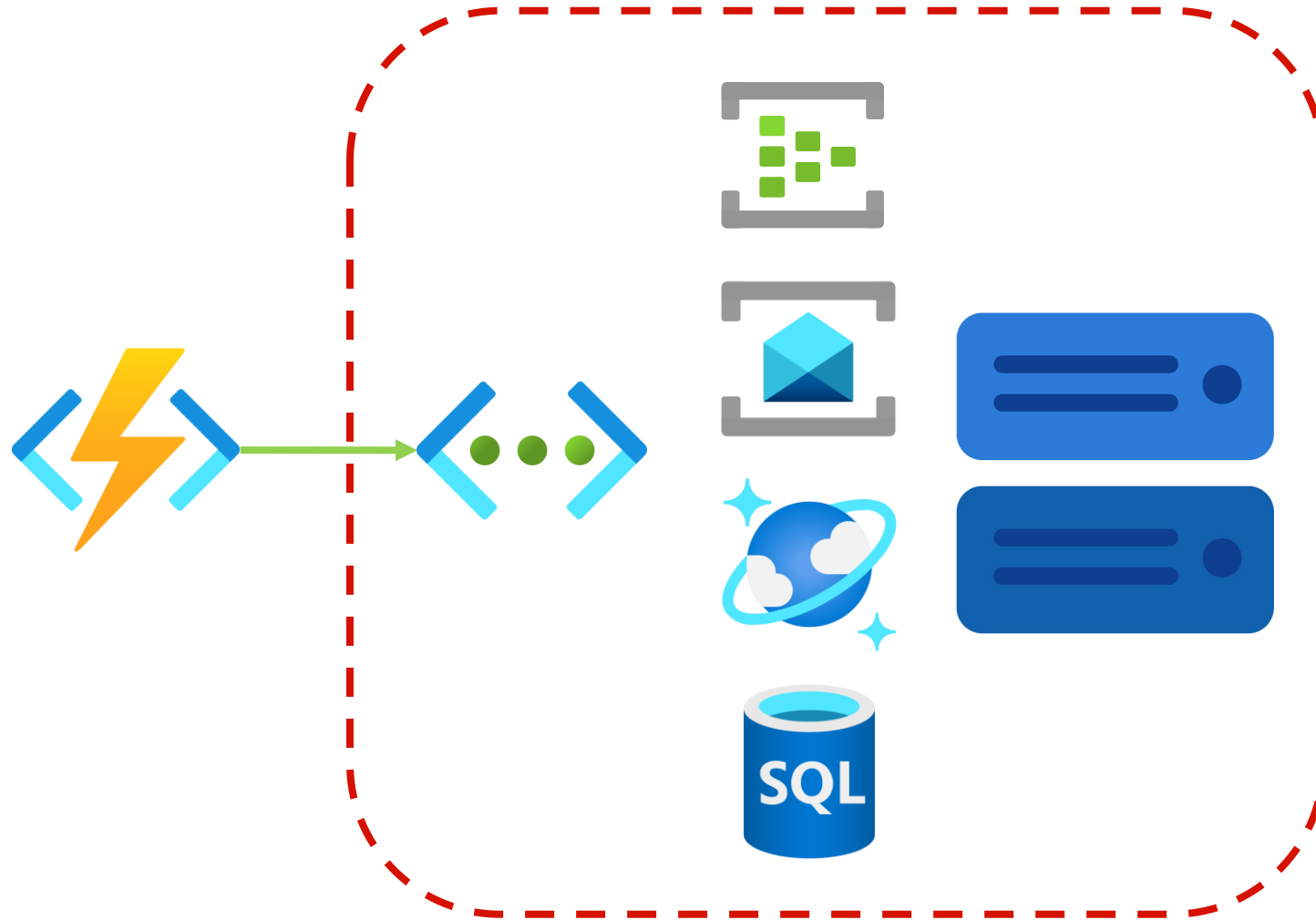
- Choose instances that are always running
- Set minimum number of instances always ready



# Virtual Network Support



# Virtual Network Support



- Inbound Private Endpoints
- Virtual Network Integration
- Virtual Network Triggers (non-HTTP)
- Hybrid connections (Windows only)

# Instance Memory Choice



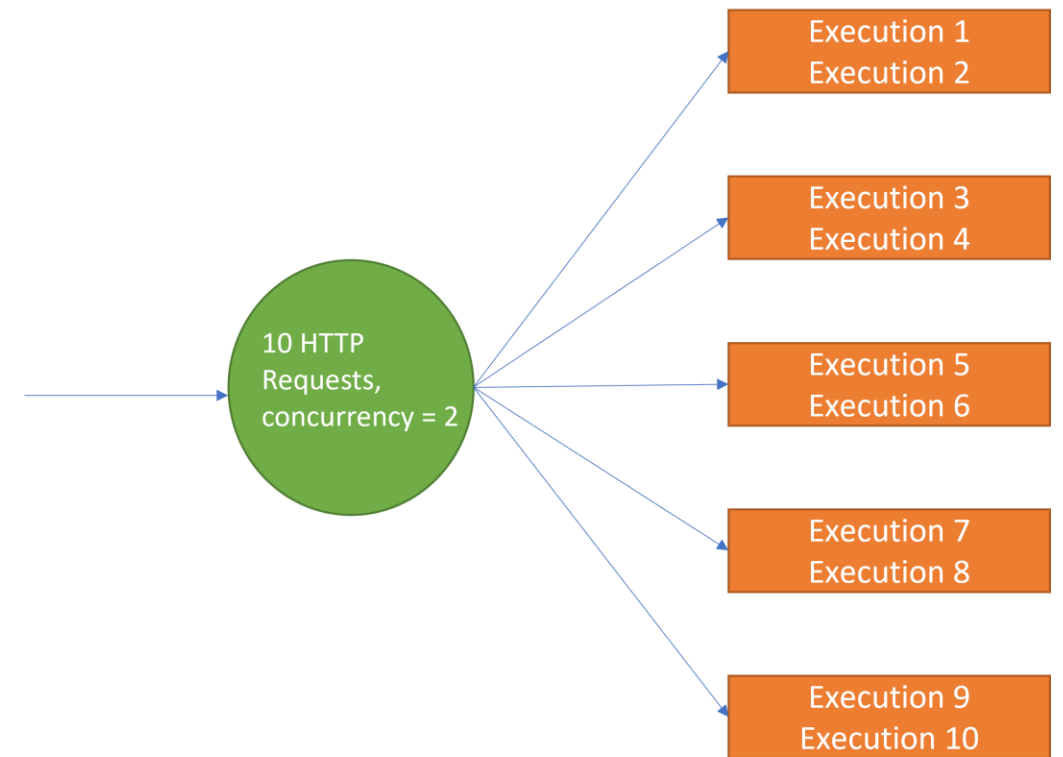
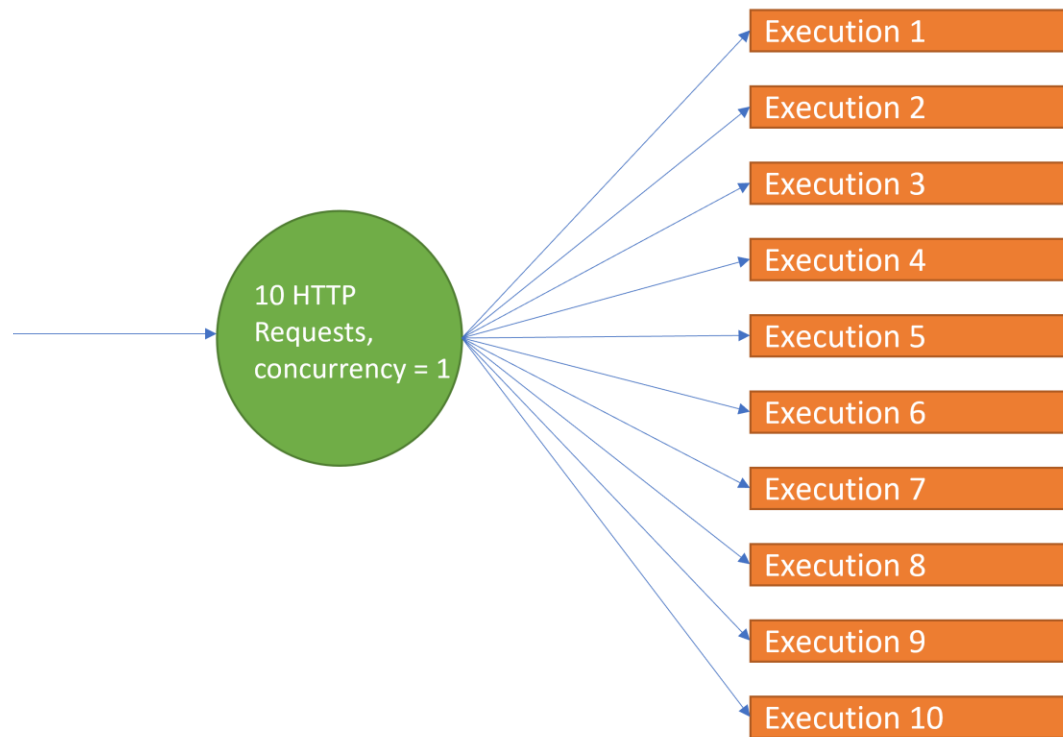
- 2048-Mb and 4096-Mb (more options coming)
- Default is 2048-Mb
- Change instance memory size at any time
- More memory means more can be done

# Per-Instance Concurrency



- Number of parallel executions
- Can set concurrency level per instance
- Has a direct effect on how your app scales

# Per-Instance Concurrency



# Per-Function Scaling

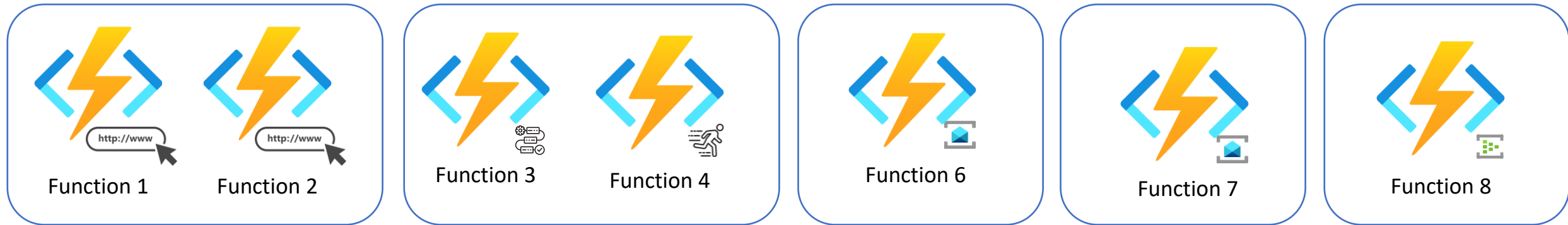


- Deterministic way of scaling your app on a per-function basis
- No code changes
- Special cases: HTTP, Blob (Event Grid), and Durable trigger

# Per-Function Scaling



- Deterministic way of scaling your app on a per-function basis
- No code changes
- Special cases: HTTP, Blob (Event Grid), and Durable trigger





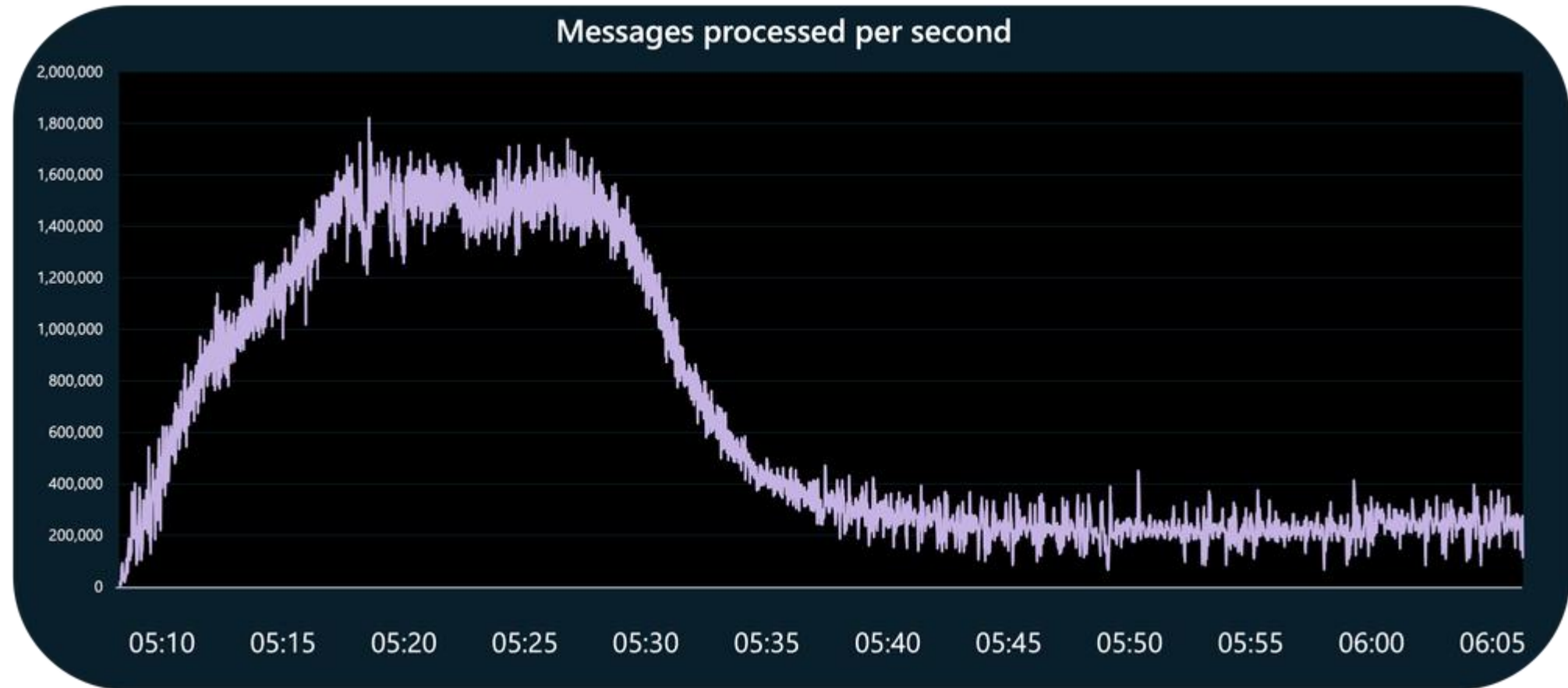
# Scale Out Further



- Default is 100 instances
- Highest is 1000 instances

**Region Subscription Memory Quota**  
512,000 MB per region per subscription

# Extreme Scalability



# Even More Benefits



- Azure Load Testing Integration
- Open Telemetry Opt-In
- Long Execution Times

# Demonstration of Extreme Scalability using Azure Functions Flex Consumption

# Flex Consumption Features and Benefits



- Always Ready
- Virtual Network Support
- Instance Memory Choice
- Per-Instance Concurrency
- Per-Function Scaling
- Scale-Out Further
- Extreme Scalability
- Azure Load Testing Integration
- Open Telemetry Opt-In
- Long Execution Times

## Pricing

# Pricing



**On-Demand**

**Always Ready**

| Meter                         | Free Grant (per month) | Pay as you go                 |
|-------------------------------|------------------------|-------------------------------|
| On Demand Execution Time      | 100,000 GB-s           | \$0.000016/GB-s               |
| On Demand Total Executions    | 250,000 executions     | \$0.20 per million executions |
| Always Ready Baseline         |                        | \$0.000004/GB-s               |
| Always Ready Execution Time   |                        | \$0.000009/GB-s               |
| Always Ready Total Executions |                        | \$0.20 per million executions |

# Pricing



## My Standard Scenario

3 million executions per month; each execution using 512 Mb and running for 1 second

Consumption: \$18.00

# Pricing



## Azure Function Scenario

3 million executions per month; each execution using 512 MB and running for 1 second

Consumption: \$18.00

Premium: \$155.27

Flex Consumption (w/o Always Ready): \$20.60



# Pricing



## **Flex Consumption with Always Ready Scenario**

3 million executions per month; each execution using 512 MB and running for 1 second; 1% of executions performed while idle

Consumption: \$18.00

Premium: \$155.27

Flex Consumption (w/o Always Ready): \$20.60

Flex Consumption (with Always Ready): \$33.70

# Pricing



## Actual Scenario

434k executions per month; each execution using 2048 MB and running for 400 milliseconds; VNet connectivity needed no Always Ready (cold starts acceptable)

Consumption: N/A

Premium: \$155.27

Flex Consumption: \$4.16



# Recap

Serverless Unleashed: From Design Patterns to Azure Solutions and Beyond

# Recap

## Serverless Architecture

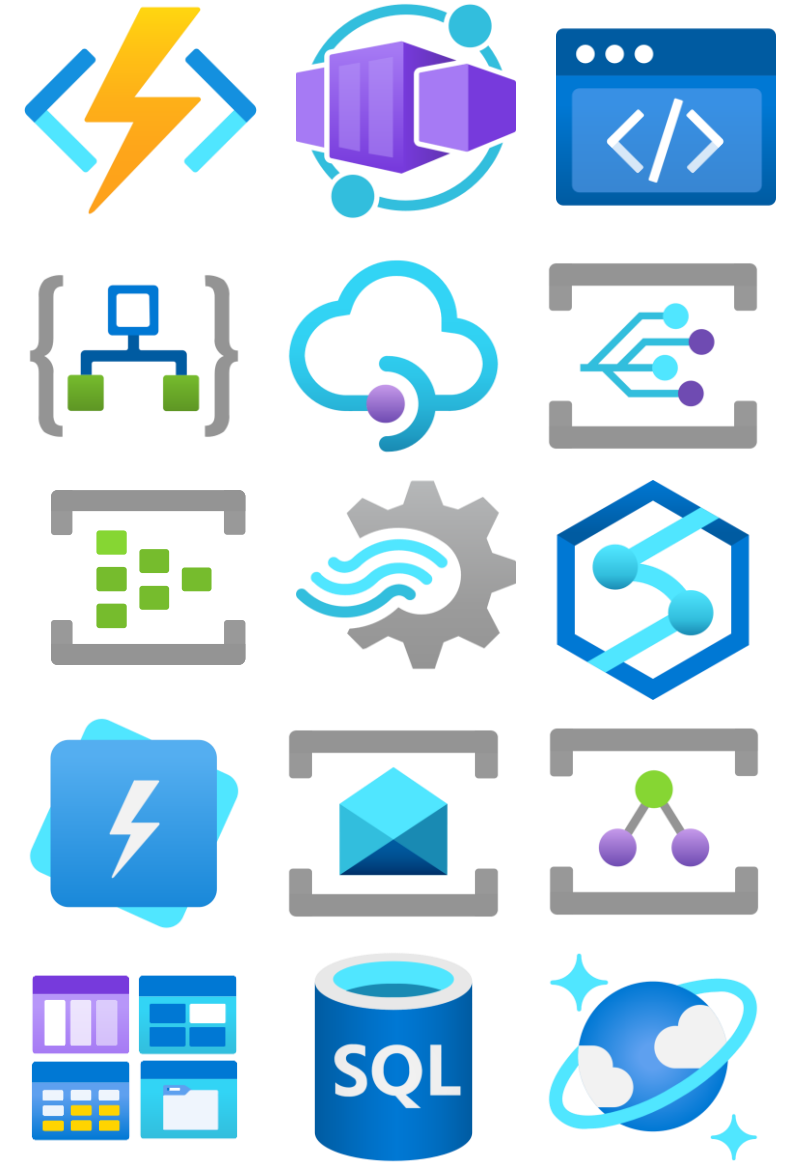
- Introduction to Serverless Computing
- Serverless Architecture Styles
- Cloud Design Patterns
- Software Design Patterns

# Agenda

## Serverless Architecture

## Azure Services Offerings

- Compute
- Workflow and Integration
- Data Processing and Analytics
- Messaging
- Data Storage



# Agenda

## Serverless Architecture

## Azure Services Offerings

## What's New and Coming

- Azure Functions on Azure Container Apps
- .NET Aspire
- Azure Functions Flex Consumption

# Thank You



chadgreen@chadgreen.com



TaleLearnCode



ChadGreen.com



ChadGreen



ChadwickEGreen

