

© Copyright Microsoft Corporation. All rights reserved.

FOR USE ONLY AS PART OF MICROSOFT VIRTUAL TRAINING DAYS PROGRAM. THESE MATERIALS ARE NOT AUTHORIZED  
FOR DISTRIBUTION, REPRODUCTION OR OTHER USE BY NON-MICROSOFT PARTIES.



# Microsoft Azure Virtual Training Day: Innovate and Scale with Cloud Native Apps



# Build optimized cloud applications

---

## Module objectives:

- 
- Describe the fundamental structure of a cloud-native app
  - Identify situations where you should build a cloud-native app

# Introduction




# Introduction

- Cloud-native apps represent a modern approach to app development, where software systems are designed with cloud technologies in mind
- Focus on **architectural modularity**, rather than monolithic, all-in-one applications

# What are cloud-native apps?



The background is a dark blue collage of six images. Top-left: A winding road through a desert landscape under a starry night sky. Top-middle: A modern stadium with a large, illuminated roof structure. Top-right: An astronaut floating in space, holding a tool. Bottom-left: A futuristic, sleek car with glowing blue accents. Bottom-middle: A hand holding a smartphone next to a small, white, cube-shaped device. Bottom-right: A close-up of a laptop keyboard with a glowing blue light.

# We're living in a future defined by accelerated innovation

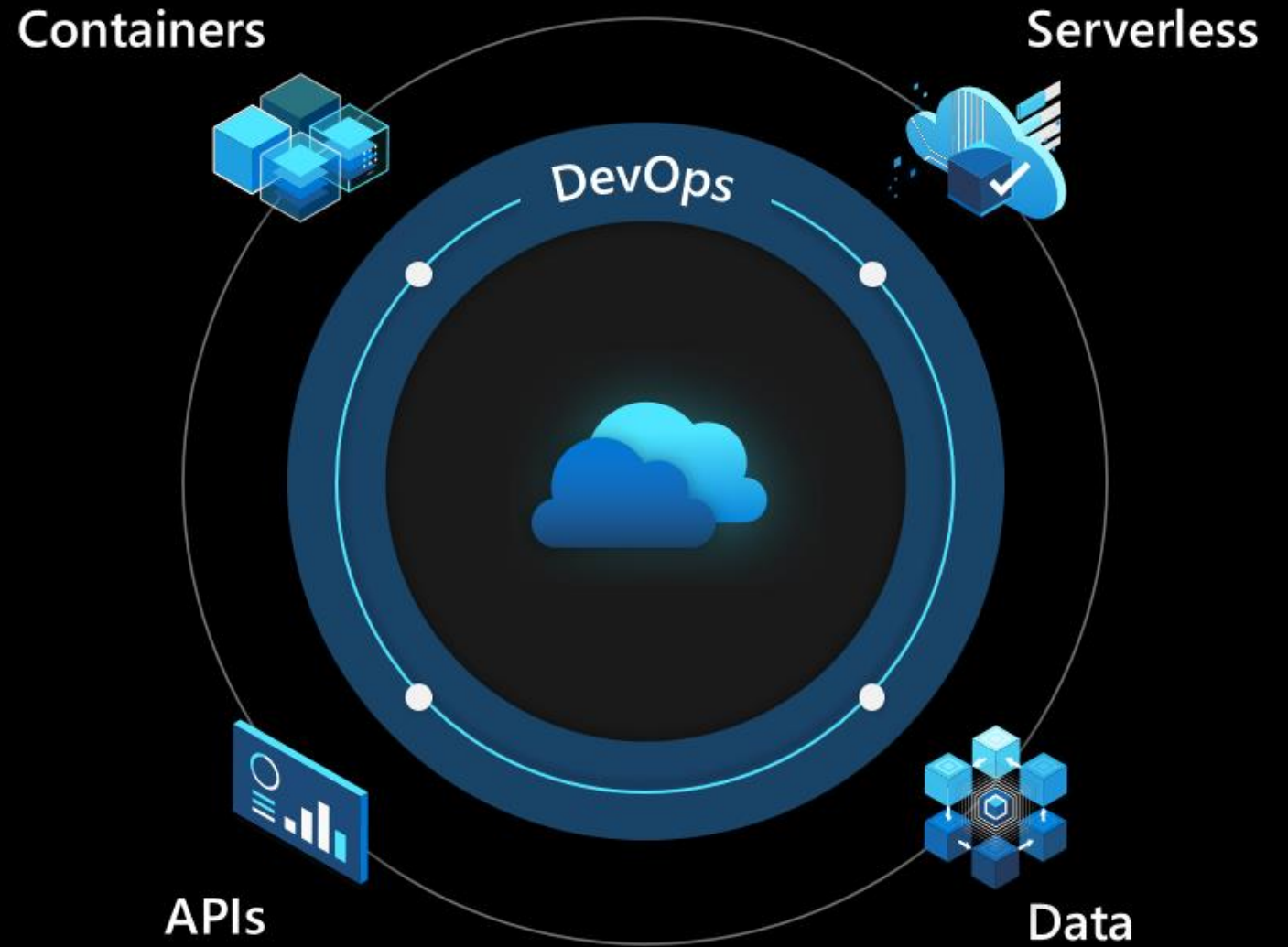
A new class of applications is being built



A realistic image of the Earth from space, showing continents and clouds. Overlaid on the globe are several thin, white, elliptical lines representing orbital paths. A network diagram is also overlaid, consisting of small white dots (nodes) connected by thin white lines, primarily concentrated in the upper half of the globe.

The app of the future is  
cloud native

# What is cloud-native?



# What is serverless?



# Serverless options in Azure

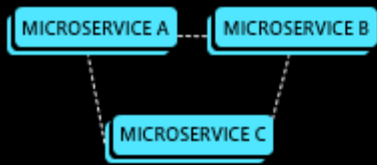
- Serverless Functions – a scale-on-demand event-driven compute experience
- Serverless Containerized Microservices – deploy containerized apps without managing complex infrastructure
- Serverless Kubernetes – provision and manage Kubernetes-based applications
- Serverless Application Environments - run and scale web, mobile, and API applications

# PaaS versus serverless

- Differences between PaaS and serverless
- Choosing between PaaS and serverless

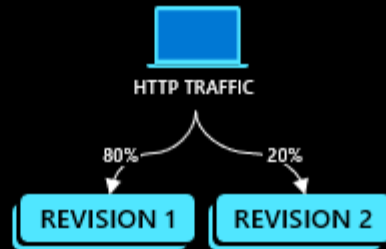
# What can you build with serverless in Cloud-native?

## Microservices



Microservices architecture with the option to integrate with [Dapr](#)

## Public API endpoints



E.g., API app with HTTP requests split between two revisions of the app

## Web Apps



E.g., Web app with custom domain, TLS certificates, and integrated authentication

## Event-driven processing



E.g., Queue reader app that processes messages as they arrive in a queue

## Background processing



E.g., Continuously running background process transforms data in a database

### AUTO-SCALE CRITERIA

Individual microservices can scale independently using any KEDA scale triggers

Scaling is determined by the number of concurrent HTTP requests

Scaling is determined by the number of concurrent HTTP requests

Scaling is determined by the number of messages in the queue

Scaling is determined by the level of CPU or memory load

# Using containers and microservices



# Using containers with cloud-native apps

- **Containers**, loosely isolated environments that can run software packages, are usually key components of cloud-native apps.
- Containers can scale out more cost effectively and nimbly than virtual machines.



# Manage containers easily with a Kubernetes service

- Kubernetes is a technology that manages multiple containers for you.
- Key benefits of Kubernetes: **self-healing, load balancing, and simplified security configuration management.**

# Designing a cloud-native app



# Designing a cloud-native app

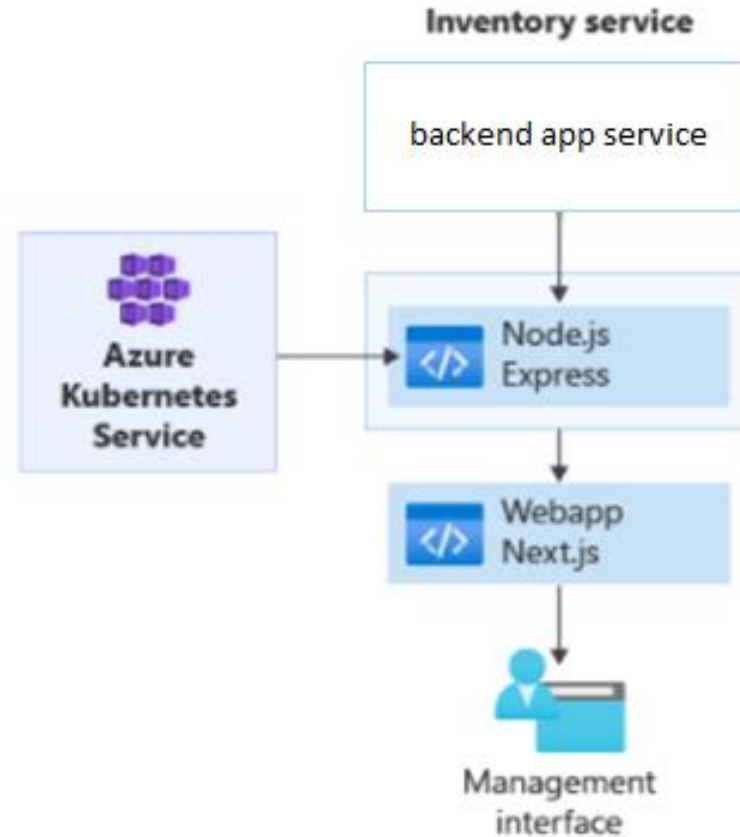
Because cloud-native apps are made up of the components of your choice, you can easily architect a solution that uses technologies you're comfortable with.

# Scenario: Online retail front and back-end apps

- Suppose you work Adatum, an online retail supplier, where you lead a small development team that's been tasked with building and maintaining the company's online presence
- Start by creating a small inventory management app
- Later, add functionality to the app
- Cloud-native apps are agile

# Scenario: Architecting a cloud-native solution for Adatum:

## Starting small



# Growing our application



# When to use cloud-native apps



# Challenges for modern developers





# Cloud native applications



## Agility

Strong developer experience

Most complete tool chain from source to production

Opensource Functions runtime

---



## Reliability

Fully managed database with >99.999-percent HA

Single-digit millisecond latencies on reads and writes

Available in more regions than any other cloud providers

---



## Security

\$1B investment every year in security

>90 compliance certifications

Out of the box integration with Azure Policy, Active Directory, and Security Center

# Summary



# Summary

In this module, you learned about cloud-native apps.

- Describe the fundamental structure of a Cloud Native App
- Identify situations where you should build a Cloud Native App



# Manage your application with Azure Container Apps

---

## Module objectives:

- Run containers on Azure Kubernetes Service
- Run containers on Azure Container Apps
- Implement Azure Functions

# Introduction



# Introduction

- When you're creating cloud-native applications, you can enjoy the many benefits of using containers, which are a great way to bundle and run applications.
- Many cloud-native architectures turn to Kubernetes to deploy and manage containers.

# Scenario: Processing orders at scale

Suppose you work for Adatum, an online retail supplier of seasonal products. You've been tasked with building an inventory management app.



Section name or number

# Running containers with Kubernetes



# Running containers with Kubernetes

- Containers are a virtualization technology.
- Containers don't have their own internal operating system.

# Using containers in the cloud

**Azure Container Registry** provides storage for container images in the cloud. Container Registry provides security benefits, such as:

- Building container images
- Authentication for who can see and use your images
- You can sign images to increase trust and reduce the chances of an image becoming accidentally-or intentionally-corrupted or otherwise infected
- All images stored in a container registry are encrypted at rest

# Azure Kubernetes Service (AKS) does the heavy lifting

AKS handles Kubernetes for you, by deploying, managing, and scaling Kubernetes clusters.

# Demo: Create an AKS cluster

# Developing with containers and AKS



# Open-source benefits

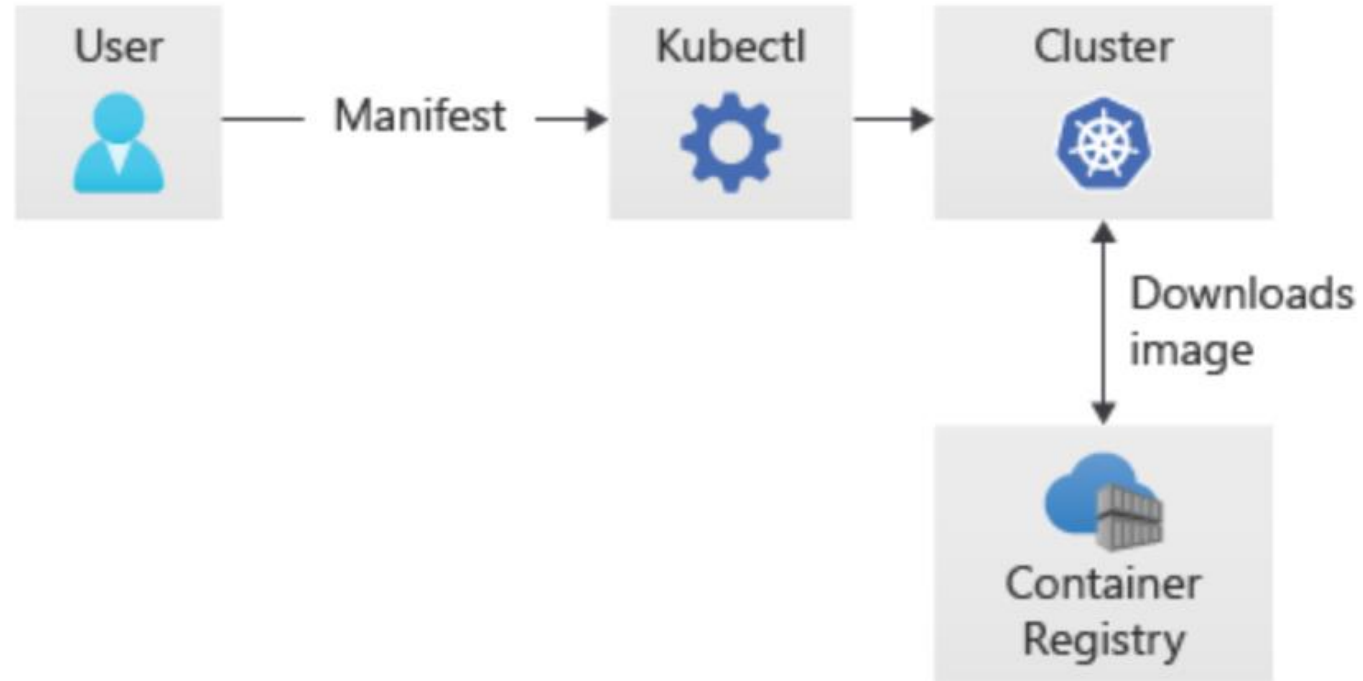
AKS can use **kubect**l, the standard Kubernetes command-line tool, or you can choose to leverage other open-source tools e.g., Argo CD.

# Deploying to a cluster

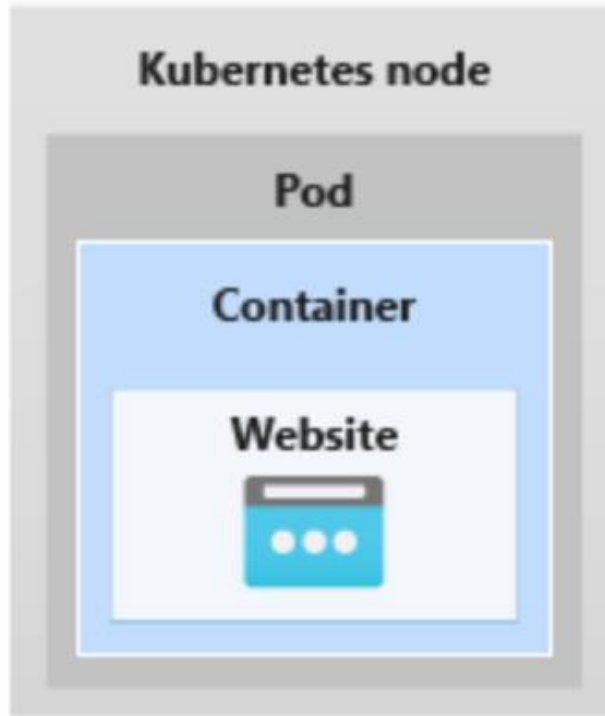
We can use `kubectl` to deploy a container from our container registry to the Kubernetes cluster.



# Creating a deployment manifest



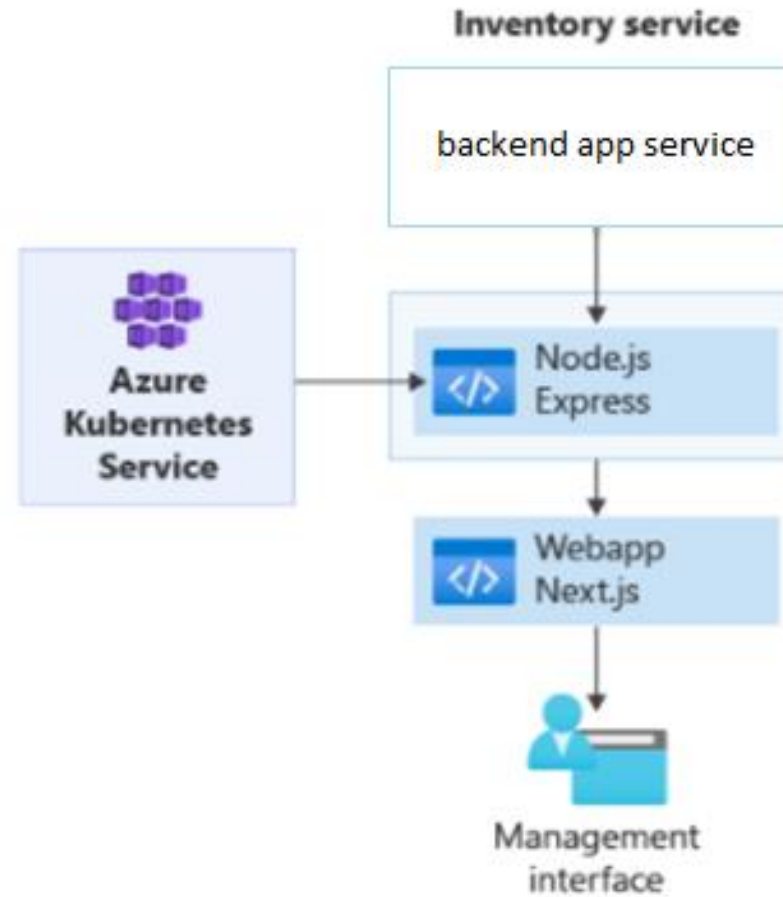
# Running a container image in Kubernetes



# Kubernetes health checks

One of the key benefits of Kubernetes is the ability to restore applications to the exact instance that had been tested and saved, otherwise known as **self-healing**.

# What our container will do



# Demo: Set up a development environment with AKS

# Connecting cloud-native components



# Connecting cloud-native components

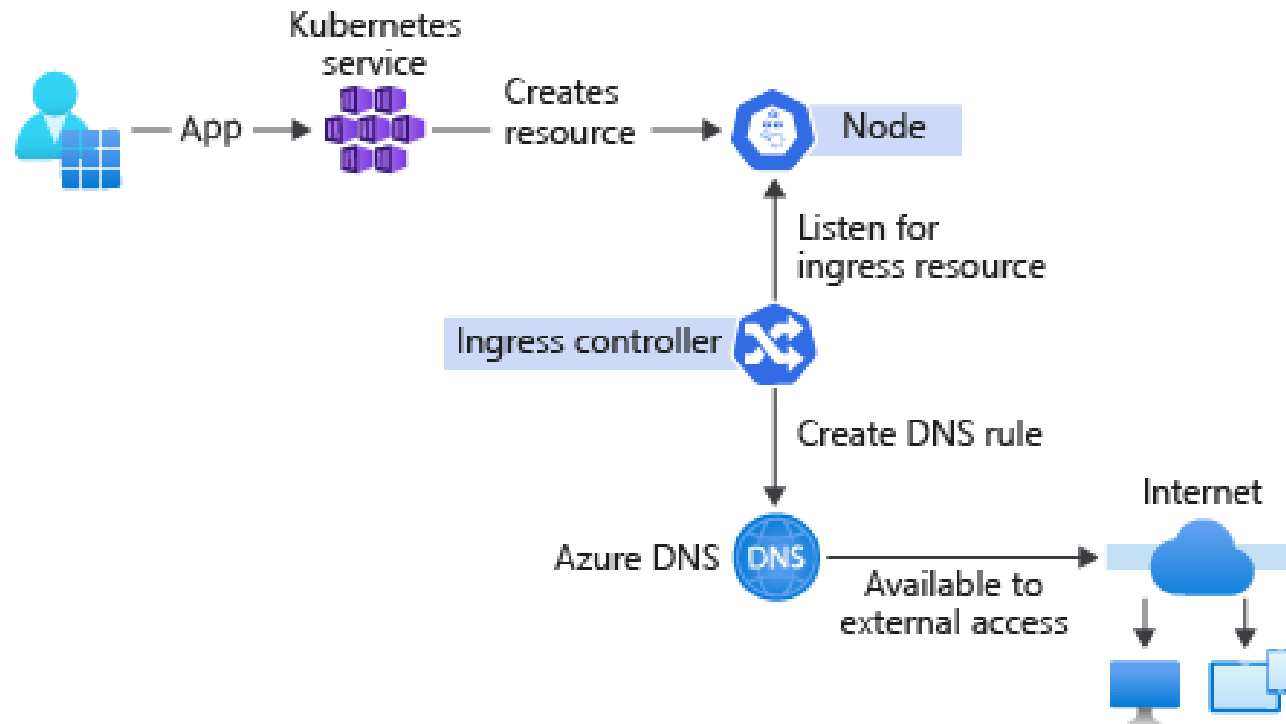
- Expose the application API
- Access applications on the cluster

# Connecting the pieces together

- An AKS cluster blocks all inbound traffic to the cluster to assure network security.
- To expose applications hosted on AKS to the world, you need to create a Kubernetes Service or an Ingress Controller.



# Ingress controllers



# Reliable rolling deployments

Once you have set up your manifest files, Kubernetes provides a rich feature set for deployment options, such as:

- Canary deployments
- Deploying services in parallel
- Taking only a specific amount of system capacity offline at a time
- Circuit Breakers if a deployment malfunctions

# Connecting the inventory management solution

In our scenario, we've hosted a Node container in AKS to process the order information that informs our inventory app. For a management webapp to receive information from the Node container, we need to enable HTTP application routing and create an ingress manifest file.

**Demo: Connect cloud-native components**

# Running containers in Azure Container Apps



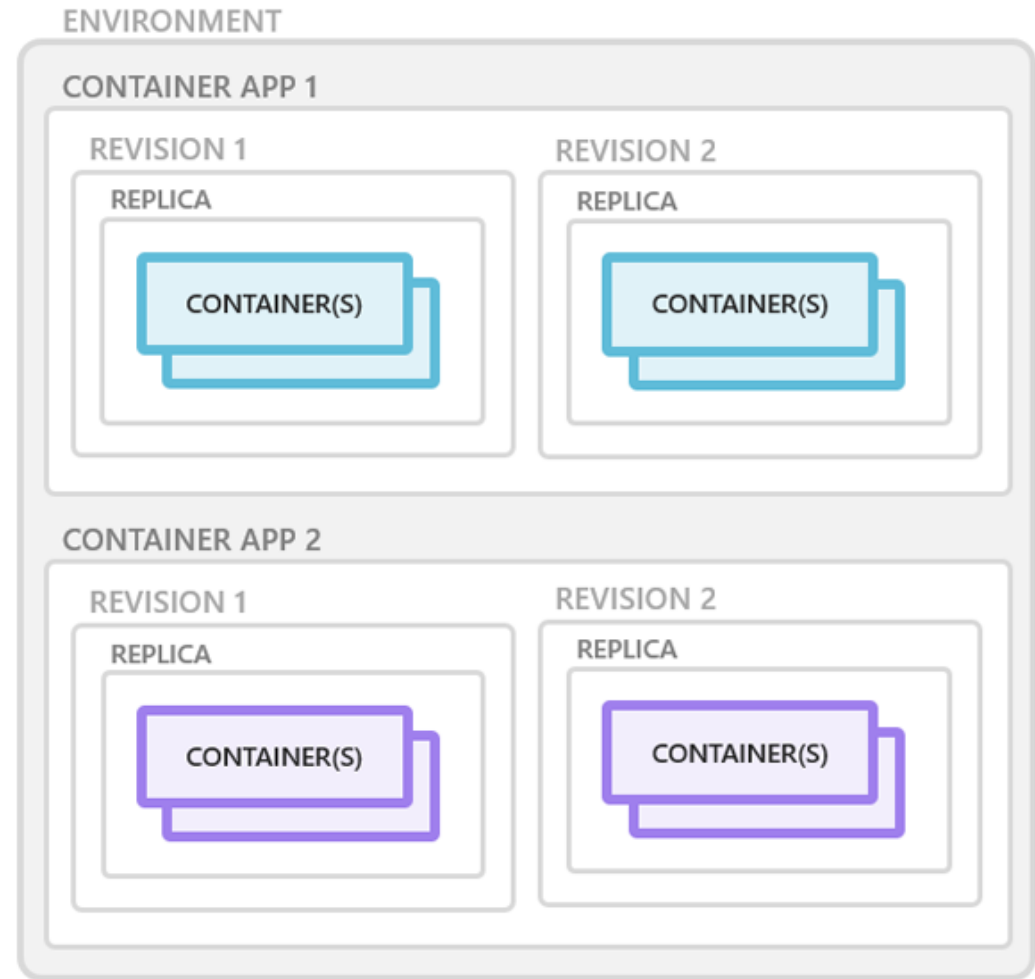
# What is Azure Container Apps?

- Runs microservices and containerized applications on a serverless platform
- Enables applications to scale dynamically
- Supports full application lifecycle

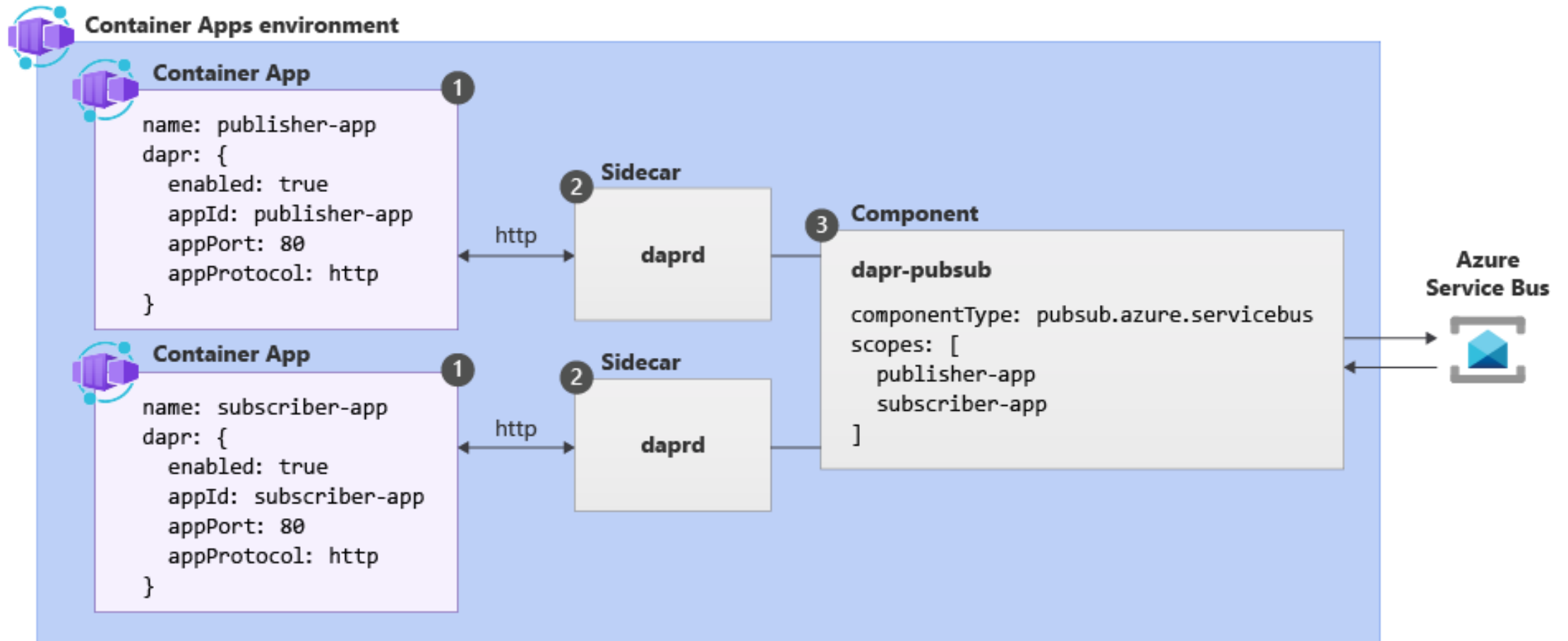
# Containers in Azure Container Apps



**Containers** for an Azure Container App are grouped together in pods inside revision snapshots.



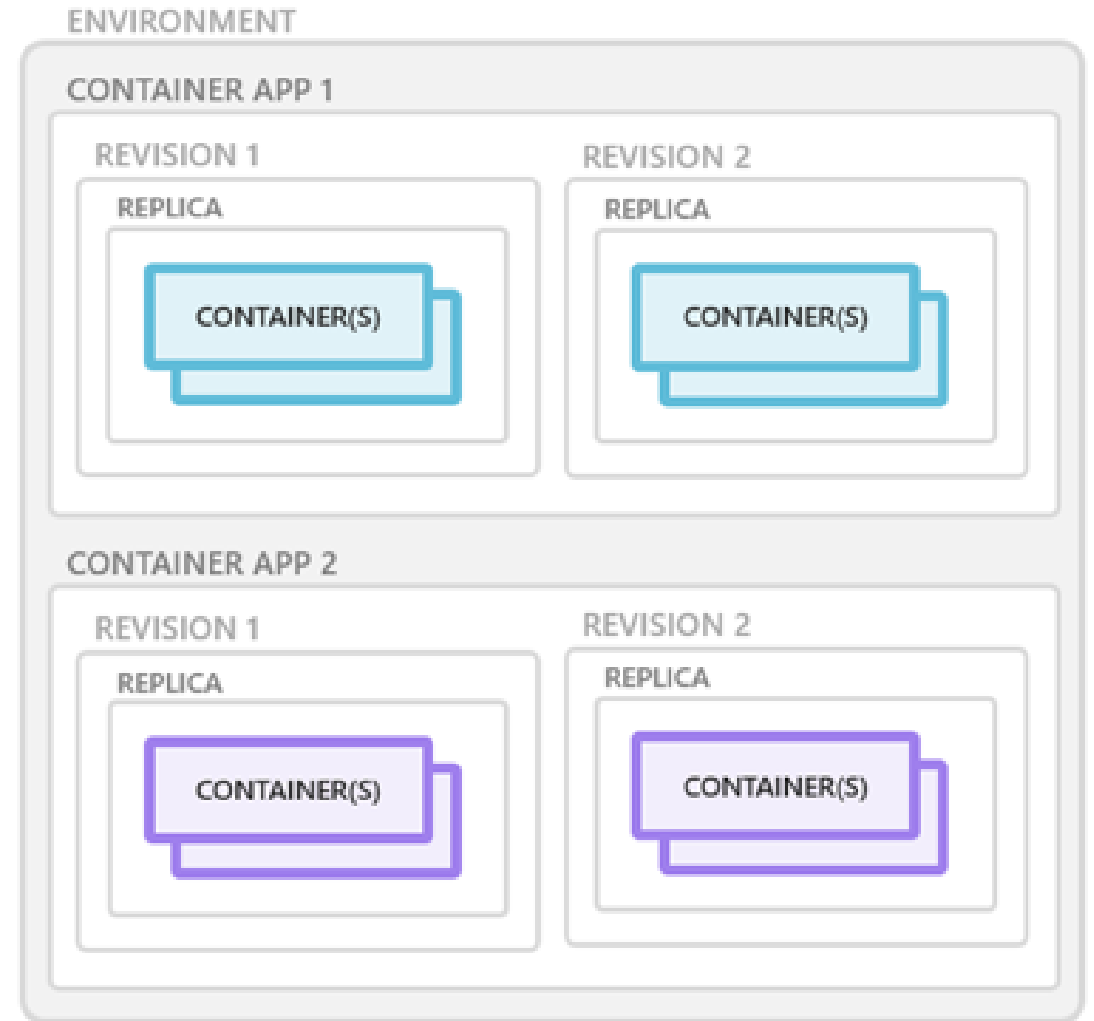
# Dapr integration with Azure Container Apps





# Automatic scaling with Azure Container Apps

- Scaling and replicas
- Scaling rules
- Events and KEDA support



# **Demo: Deploy a background processing application with Azure Container Apps**

# Implementing an event driven architecture with Azure Functions



# What are Azure Functions?

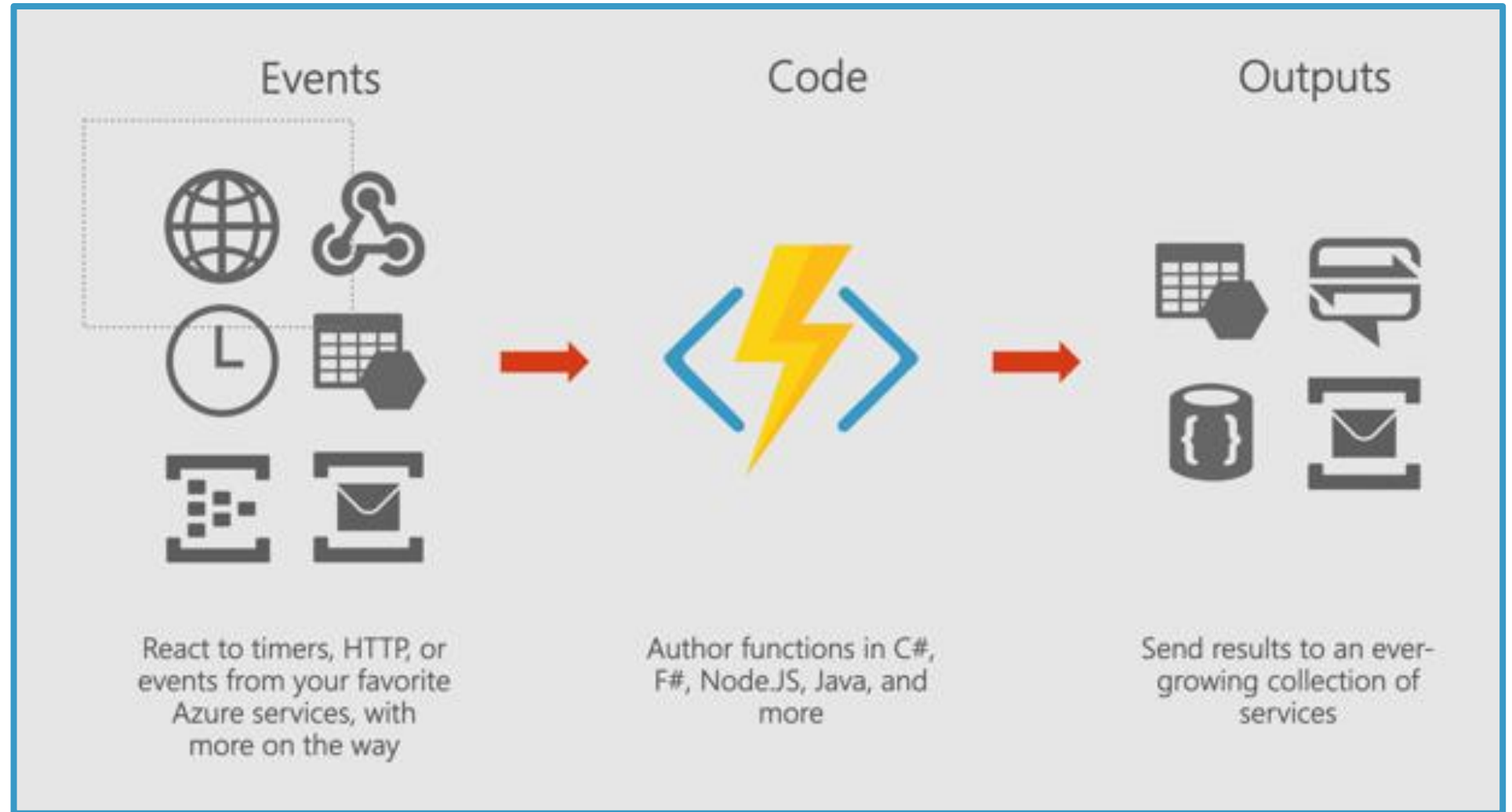
---

## Optimized for Event Driven Functions

- Event-driven programming model
- Elastic scale
- Extensive support for programming languages
- Open source and can run anywhere
- Industry leading tools



# How do Azure Functions work?



# Why Azure Functions in our deployment?

- Increase developer velocity
- Boost team performance
- Improve organizational impact

# Top use cases for Azure Functions

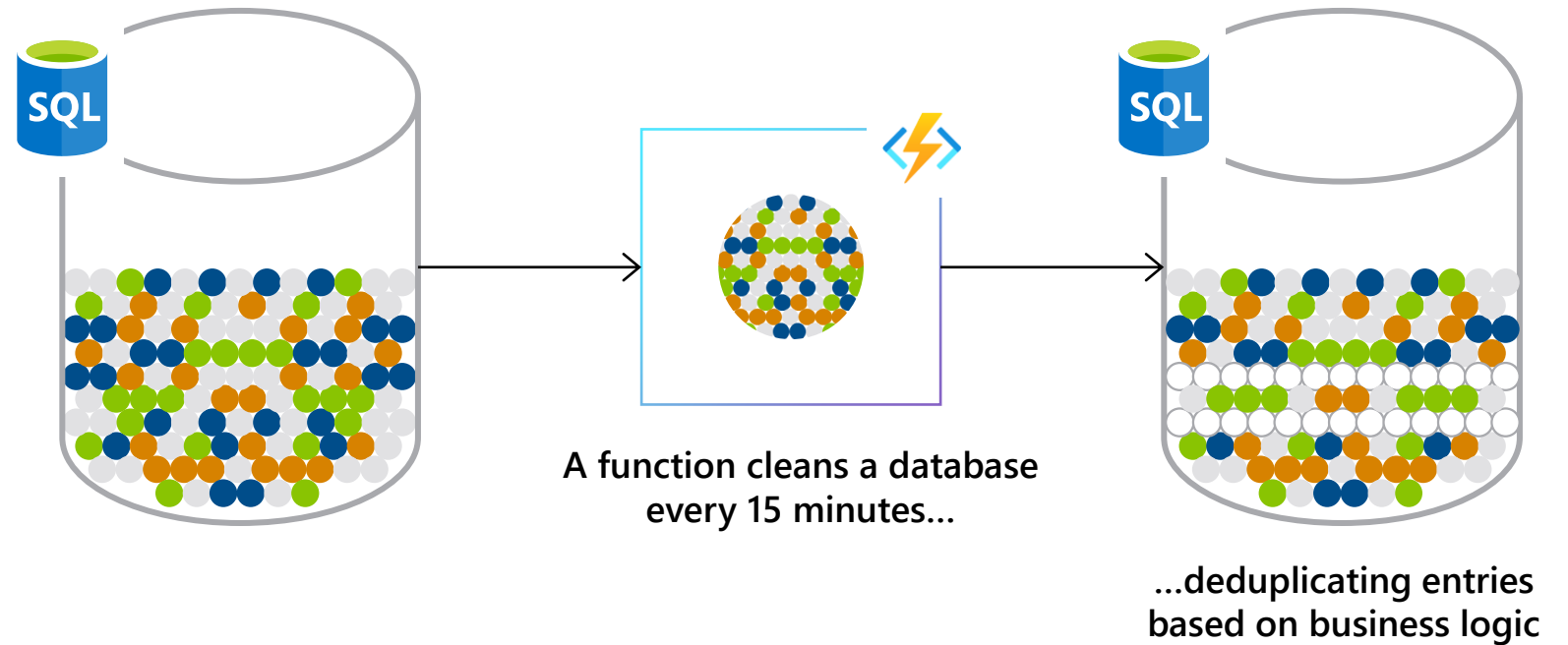
- Automation of scheduled tasks
- Real-time stream processing
- Backend for Web and mobile apps

# Automation of scheduled tasks

## SCENARIO EXAMPLE

### Financial services

A customer database is analyzed for duplicate entries every 15 minutes, to avoid multiple communications being sent out to same customers



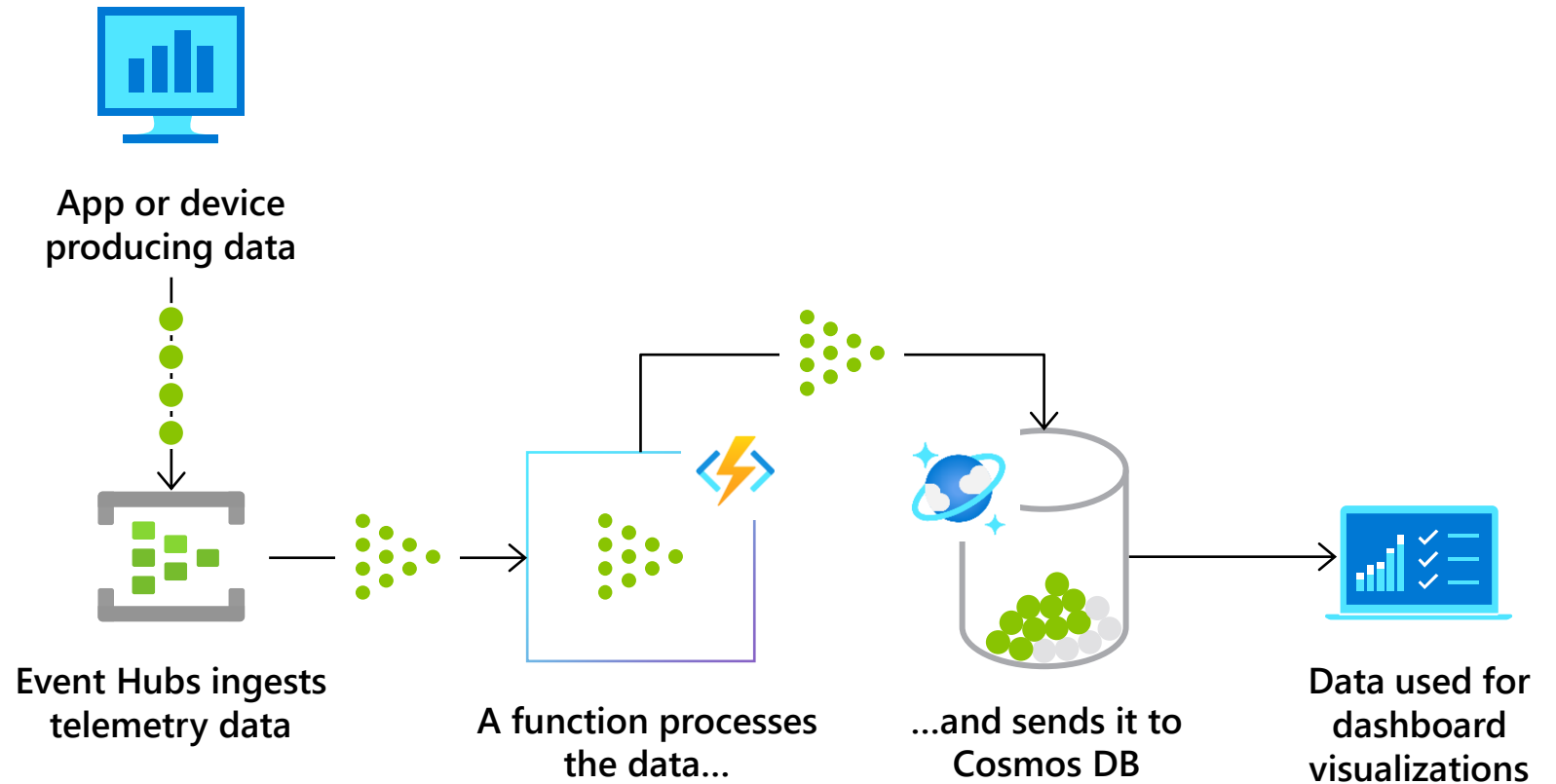


# Real-time stream processing

## SCENARIO EXAMPLE

### ISV

Huge amounts of telemetry data is collected from a massive cloud app. That data is processed in near real-time and stored in a DB for use in an analytics dashboard



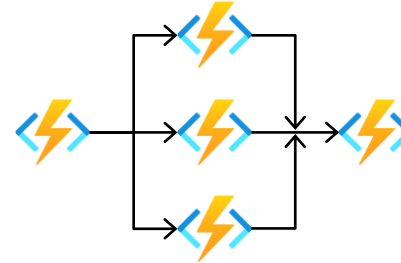
# Workflows and orchestration with Durable Functions

## PATTERNS/USE CASES

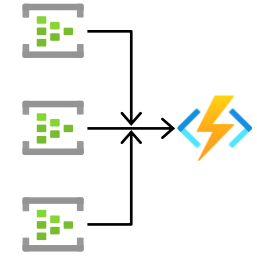
Durable Functions is an extension of Azure Functions that lets you write stateful functions in a serverless compute environment



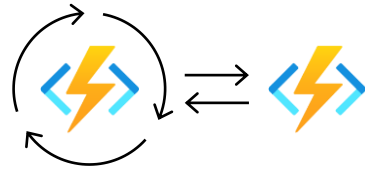
Manageable sequencing +  
error handling/compensation



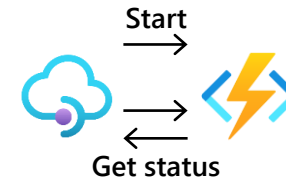
Fanning out and fanning in



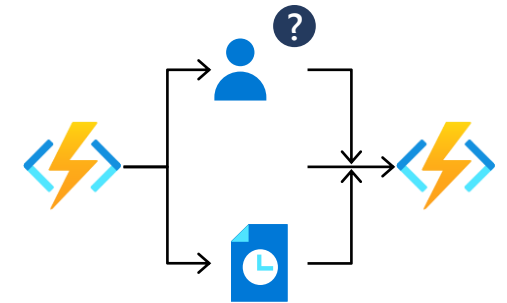
External events correlation



Flexible automated long-running  
process monitoring



Http-based async long-  
running APIs



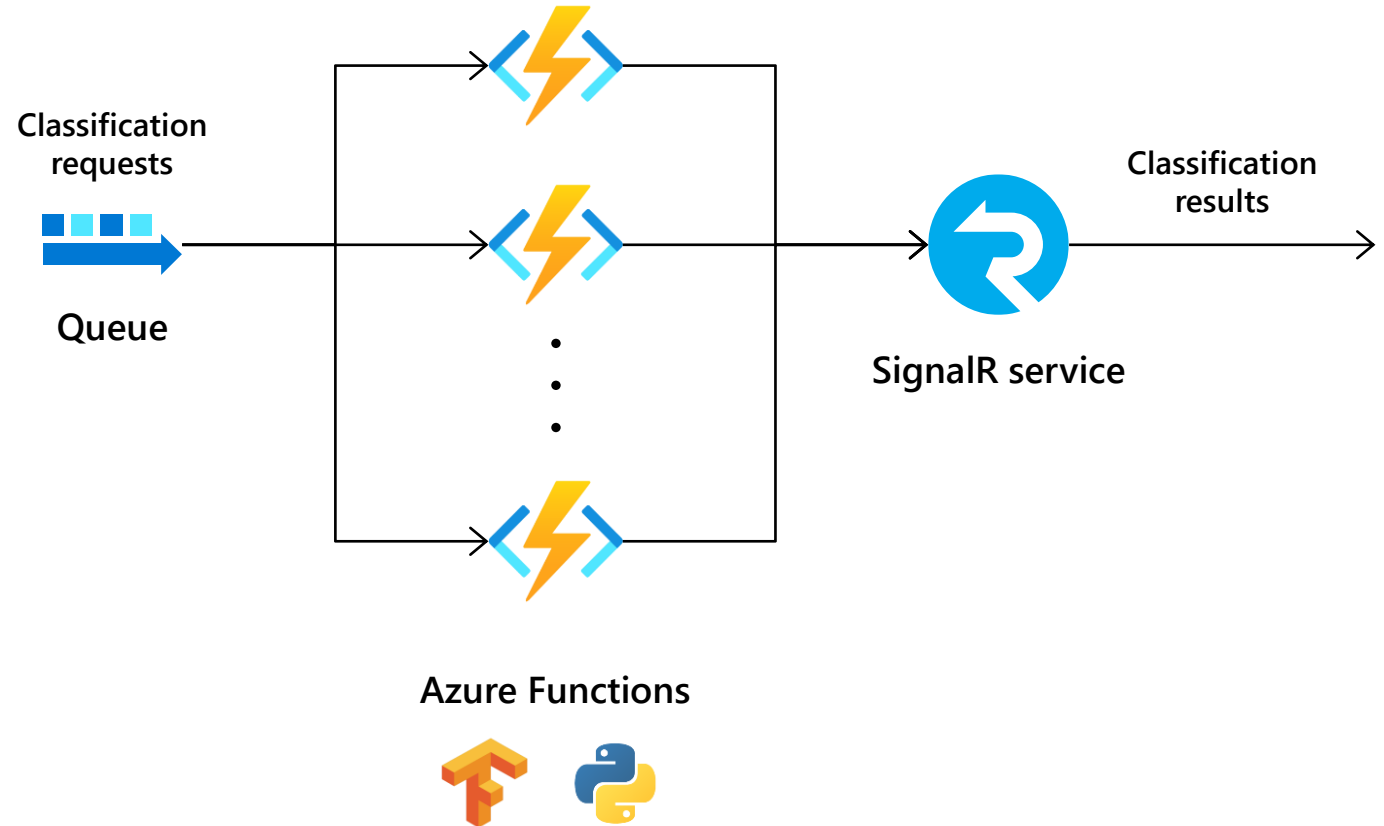
Human interaction

# Machine Learning

## SCENARIO EXAMPLE

### Image Classification Service

Images are ran through a TensorFlow model to perform classification on a stream of images

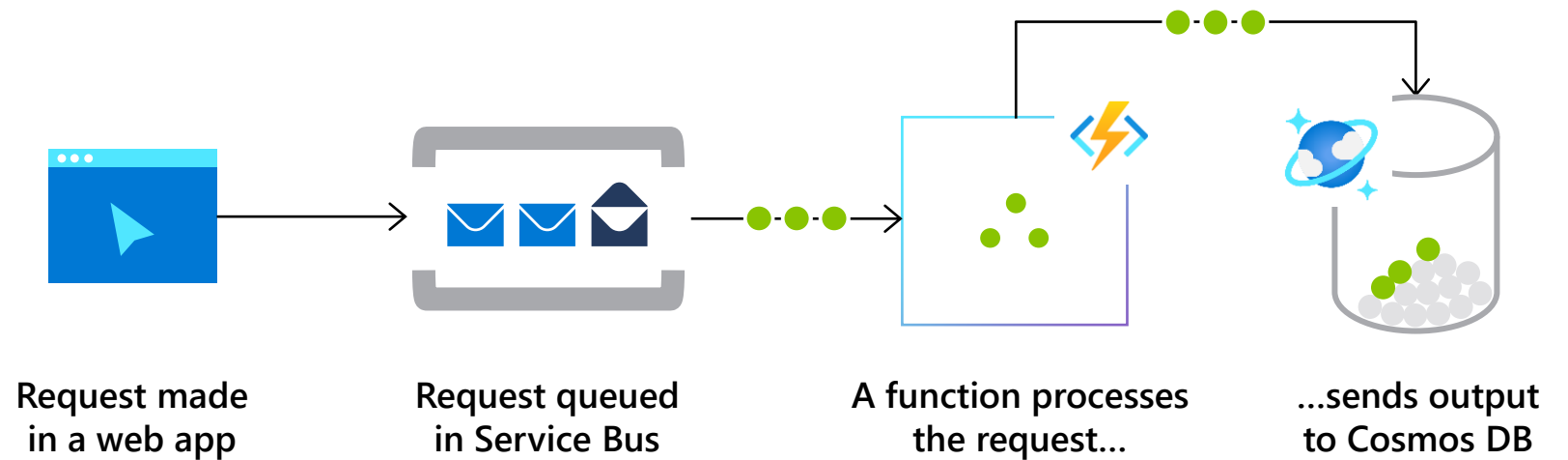


# Web application backends

## SCENARIO EXAMPLE

### Retail

Online orders are picked up from a queue, processed and the resulting data is stored in a database



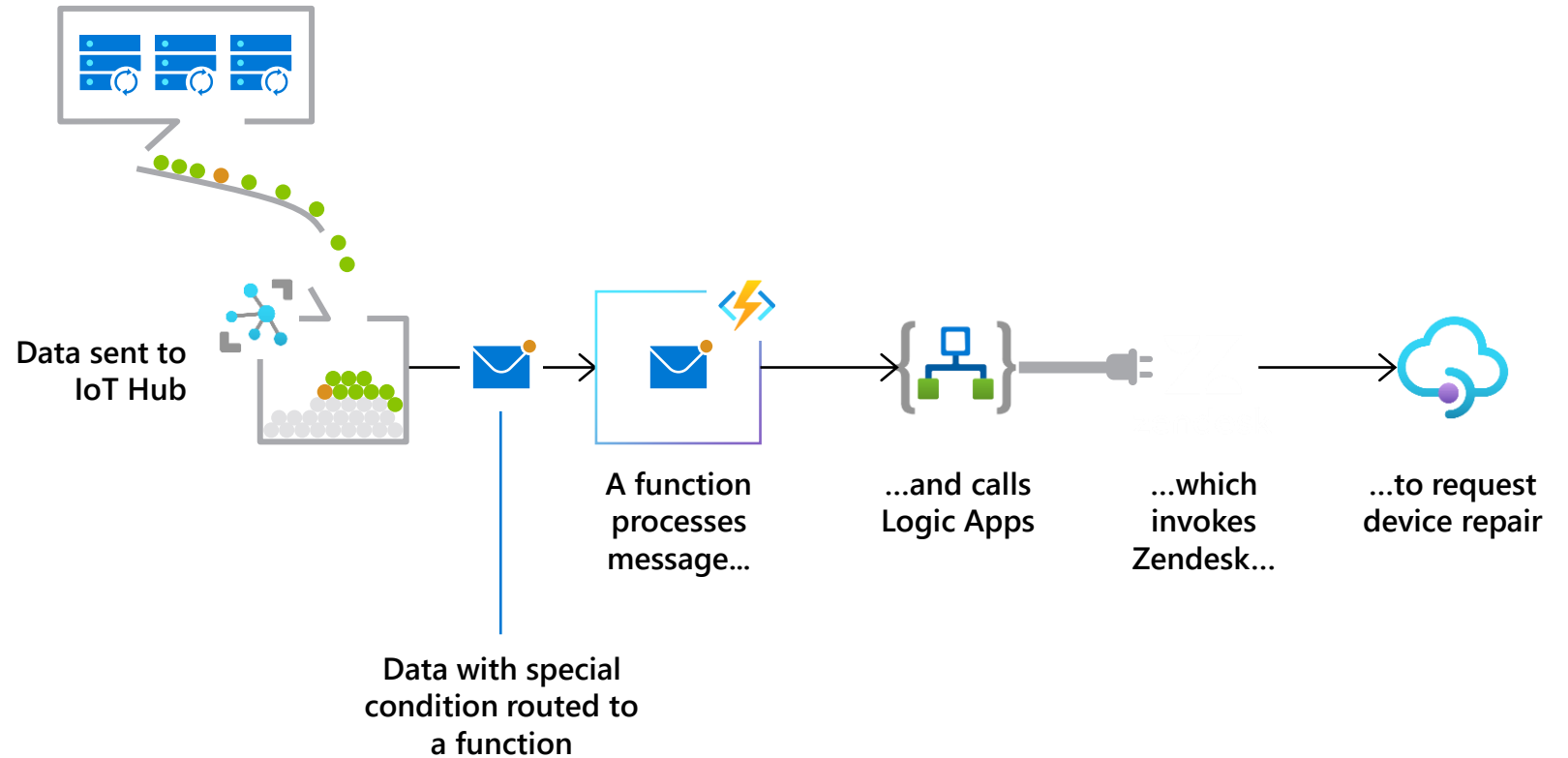
# IoT-connected backends

## SCENARIO EXAMPLE

### Manufacturing

A manufacturing company uses IoT to monitor its machines. Functions detect anomalous data and trigger a message to the Service department when repair is required.

Connected IoT devices  
producing data



# **Demo: Deploy an Azure Function project to Azure**

# Summary

# Summary

After completing this module, you know how to:

- Create a Kubernetes AKS cluster
- Run a container in Kubernetes
- Connect the container to a webapp
- Deploy an Azure Function





# Leverage managed databases for cloud-native applications

---

## Module objectives:

- Describe the characteristics and functionality of Azure Cosmos DB.
- Learn the concept of service in the context of cloud-native applications
- Set up a basic service
- Implement Azure Database for PostgreSQL

# Define the concept of services



# Define the concept of services

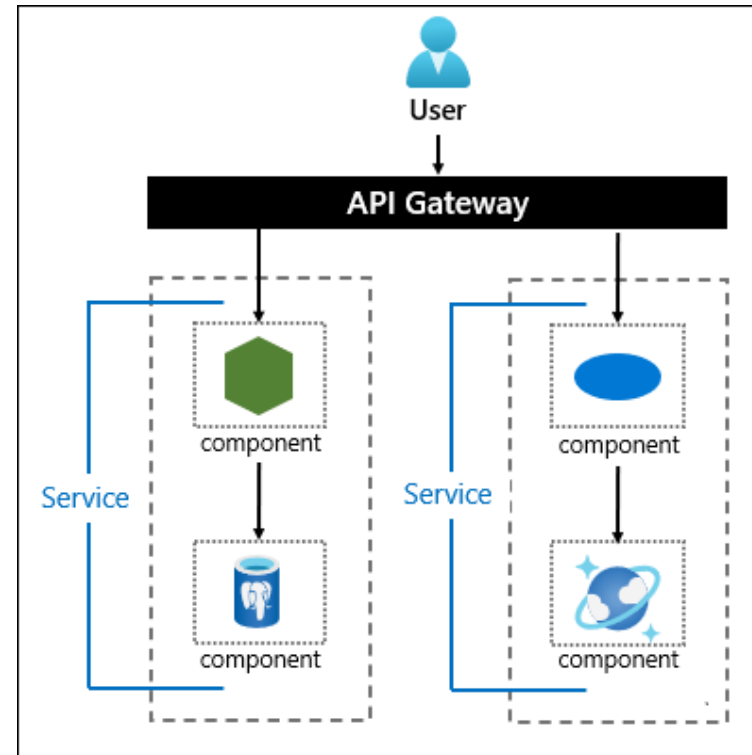
Typically, you transition from the traditional software programming model to cloud-native applications when you need to improve the agility, availability, and resiliency of your workloads.

# What is a service?

The term *service* represents a collection of components that collectively deliver specific, workload-oriented functionality to your cloud-native application.

# How can services use Azure capabilities?

In the context of cloud-native applications, you can optimize the use of services by using Azure capabilities.



# Describe Azure Cosmos DB



# What is Cosmos DB?

- Azure Cosmos DB is a fully managed, cloud-native NoSQL database.



# What are the advantages of Cosmos DB over relational databases?

- Support for different consistency models
- Built-in replication
- Multiple-region writes
- Configurable conflict resolution mechanism

# What is the Cosmos DB resource model?

- To implement Azure Cosmos DB, you need to first create an Azure Cosmos DB account in your Azure subscription.
- The number of resources available to process data within a database or its individual collections depends on the number of available Request Units (RU). Cosmos DB offers 3 modes of RU allocation:
  - Provisioned throughput mode
  - Autoscale mode
  - Serverless mode

# What are the benefits and use cases of Cosmos DB in streaming and other high-performance scenarios?

Azure Cosmos DB offers many capabilities that make it particularly suitable for streaming scenarios, including:

- Partitioning
- Time to Live (TTL)
- Change feed
- Performance and resiliency SLAs
- Schema-less databases
- Automatic indexing

Note: A logical partition can't exceed 20 GB in size.

# Demo: Set Up Azure Cosmos DB

# Describe Azure Database for PostgreSQL



# What is Azure Database for PostgreSQL?

- A Microsoft-managed implementation of the PostgreSQL Community Edition database engine.
- Available in 3 deployment modes:
  - Single Server
  - Flexible Server
  - Hyperscale

# **Demo: Set up Azure Database for PostgreSQL**

# Summary





# Summary

After completing this module, you know how to:

- Describe the characteristics and functionality of Azure Cosmos DB.
- Learn the concept of service in the context of cloud-native applications
- Set up a basic service
- Implement Azure Database for PostgreSQL
- Identify cloud-native applications scenarios in which Azure Database for PostgreSQL can provide meaningful benefits.



# Monitor your deployments with native Azure services

---

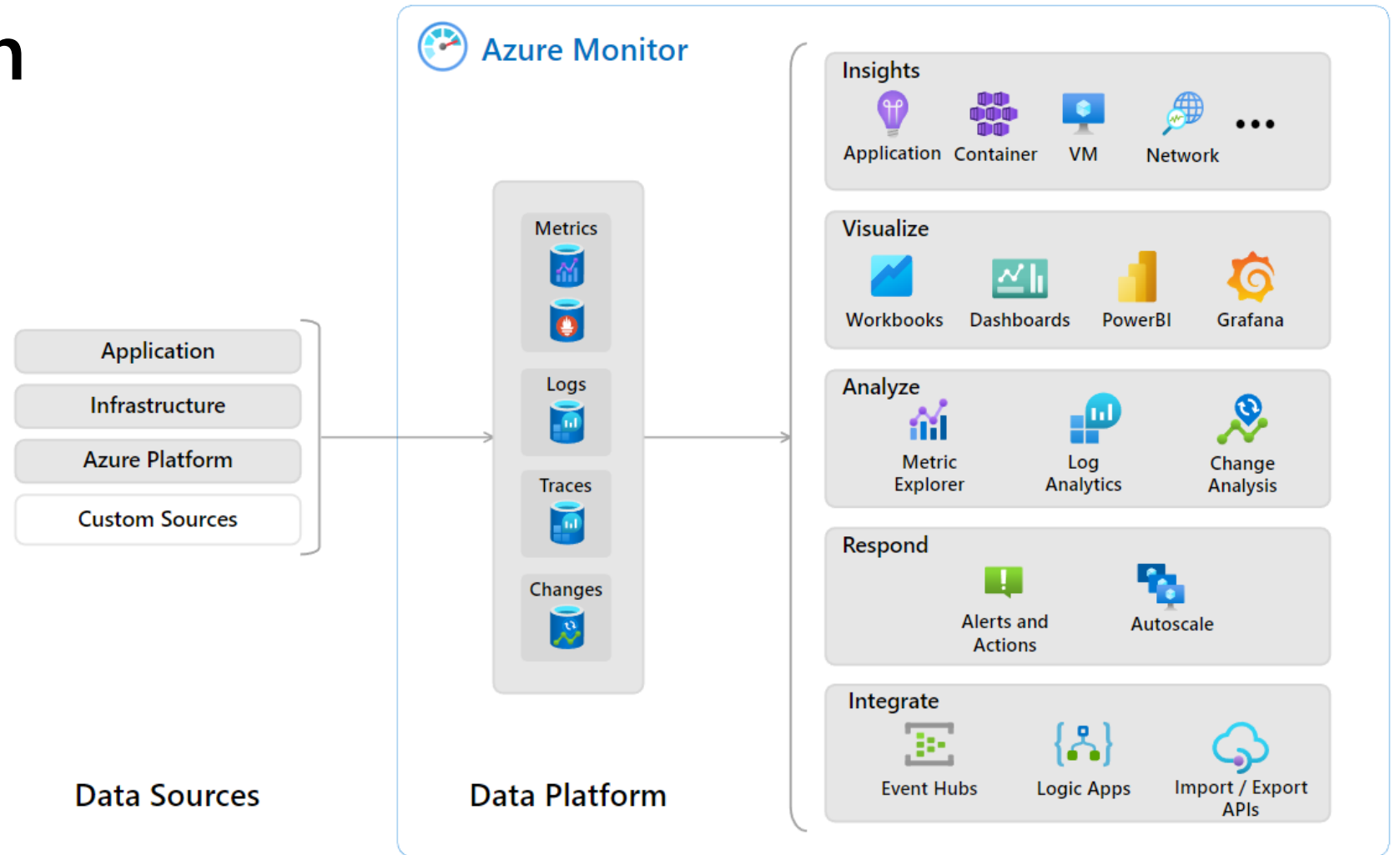
## Module objectives:

- 
- Monitor Container App instances.
  - Analyze log data with Log Analytics
  - Monitor AKS and Container Apps with Azure Monitor

# Monitor Container App instances



# Get started with Azure Monitor



# Monitoring Azure Container Apps

You can monitor Azure Container Apps throughout the development-to-production lifecycle:

- Develop and test
- Deployment
- Maintenance

# Analyze log data with Log Analytics



# Application logging in Azure Container Apps

managedEnvironment-rgcloudnativeap-bcc6 | Logging options

Container Apps Environment

Search

Overview

Access control (IAM)

Tags

Settings

Dapr components

Certificates

Custom DNS suffix

Locks

Apps

Monitoring

Logs

Logging options

Refresh Send us your feedback

Where do you want to store logs for this environment?

Logs Destination

☒ **Azure Log Analytics:** Search and analyze logs at any scale.

☐ **Azure Monitor (preview):** View, analyze, and act on log data. If selected, you'll route logs to one or more destinations in Diagnostic settings.

☐ **Don't save logs:** Stream logs without storing them.

Log Destination Configuration

Subscription \*WWL Technical Content Development

Resource group \*rg-cloudnativeapps

Log Analytics workspace \*workspacergcloudnativeapps815

[Create new](#)



# Using the container console and log streams

Troubleshooting your application inside a container

- Container console
- Log streams

# Monitor Container Apps with Azure Monitor



# Monitor Azure Container Apps metrics

- Available metrics
- Metric snapshots
- Metrics explorer

# Configure monitoring with Log Analytics

- System logs
- Console logs
- Query log with Log Analytics

# Configure Alerts

- Alert types
  - Metric alerts
  - Log alerts
- Alert rules
  - Condition
  - Logic

## Create an alert rule ...

ScopeConditionActionsDetailsTagsReview + create

### Project details

Select the subscription and resource group in which to save the alert rule.

Subscription \* ⓘVendor learning support subscription

Resource group \* ⓘmy-container-apps

Create new

### Alert rule details

Severity \* ⓘ3 - Informational

Alert rule name \* ⓘmaximum replica alert

Alert rule description ⓘAlert when there are more than 10 replicas

Advanced options

Review + create

Previous

Next: Tags >

# **Demo: Configure logging for Azure Container Apps**

# Summary



# Summary

After completing this module, you know how to:

- Describe the tools used to monitor Azure Container Apps
- Configure Log Analytics and the Azure Container Apps logs
- Configure monitoring for Azure Container Apps





# Deploy and maintain cloud-native apps with GitHub actions and Azure Pipelines

---

## Module objectives:

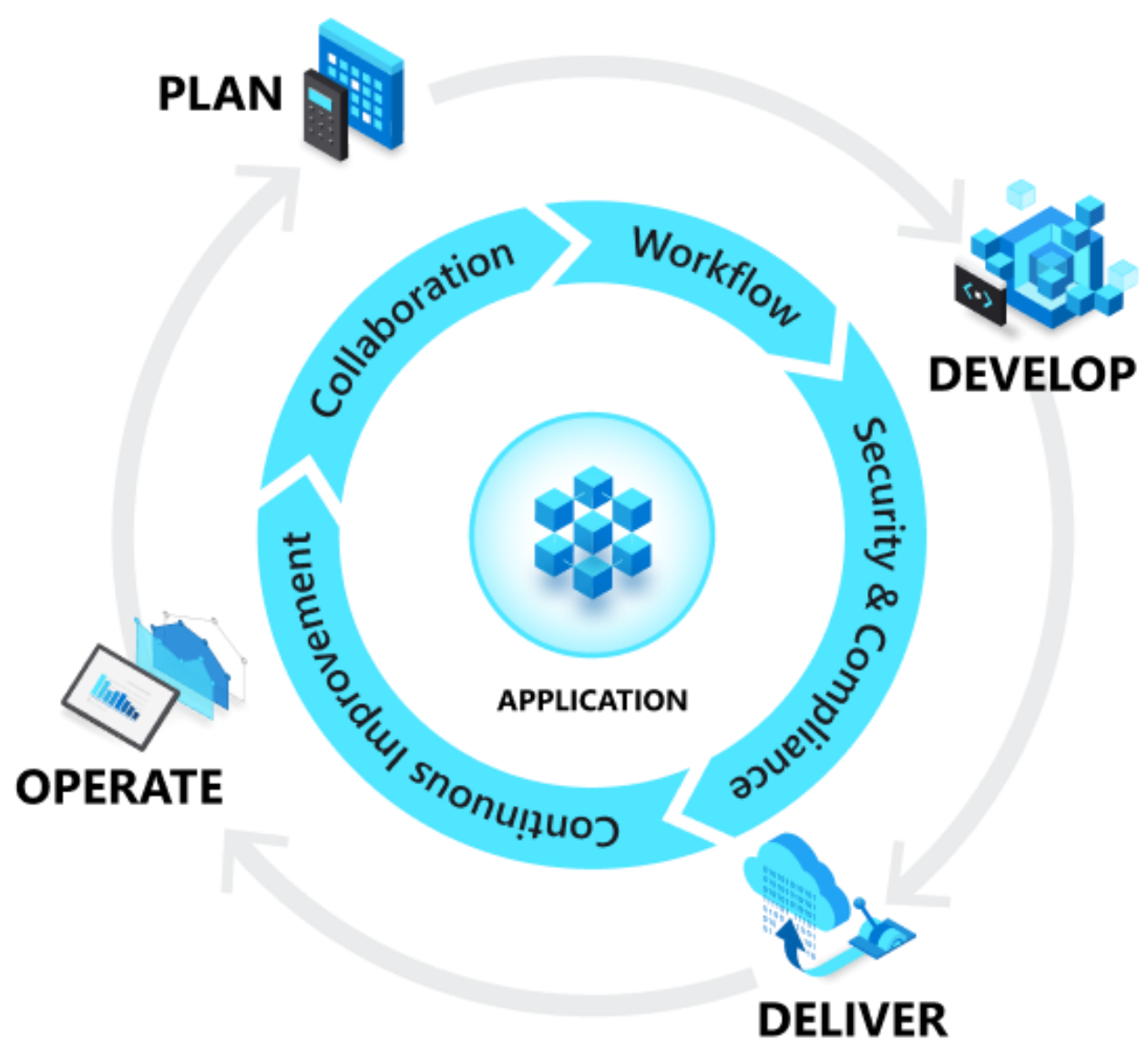
- 
- Describe the principles of DevOps and their implementation in cloud-native application scenarios.
  - Implement DevOps principles by using GitHub repositories, actions, and workflows, as well as Azure Pipelines and Azure Repos.
  - Build and deploy infrastructure and applications by using GitHub workflows and Azure.

# Define the principles and benefits of DevOps



# What is DevOps?

A compound of development (Dev) and operations (Ops), DevOps is the union of people, process, and technology to continually provide value to customers.



# What are the main DevOps components?

The main DevOps components that you want to focus on include:

- Source control
- Continuous integration (CI)
- Continuous delivery (CD)
- CI/CD pipelines
- Infrastructure as Code (IaC)

# What are CI/CD pipelines?

*CI* is the process of automating the build and testing of code every time an update is committed to the target repository in a version control system.

*CD* is the process of automating delivery of artifacts to the target environment that's providing resources for deploying and running the corresponding software.

# What is IaC?

Infrastructure as Code (IaC) applies DevOps principles to managing and maintaining services that traditionally are the responsibility of infrastructure and platform teams within an IT organization.

- Declarative code
- Imperative code

# What are the benefits of GitHub Actions?

GitHub Actions provide task automation and workflow functionality.

GitHub Actions consist of the following components:

- Workflow
- Runner
- Event
- Job
- Step
- Action



# Deploy and maintain cloud-native applications by using GitHub Actions



# How do cloud-native applications benefit from DevOps?

DevOps practices facilitate implementing cloud-native applications. They closely align with Twelve-Factor App guidelines that serve as the foundation for building cloud-native apps. In particular, they help address the following guidelines:

- **Code Base:** A single code base for each microservice that's stored in its own repository and tracked with version control.
- **Build, Release, Run:** Each release must enforce a strict separation across the build, release, and run stages.

# How to implement CI/CD by using GitHub Actions

Cloud-native applications are particularly suitable for containerization and container orchestration, which further enhances their agility.

GitHub Actions also include support for capabilities such as:

- Provisioning resources by using Azure Resource Manager templates.
- Running Azure CLI commands.
- Applying Azure Policy.

# Commonly-used actions applicable to Azure-based cloud-native applications

Scenarios include:

- `azure/login`
- `azure/arm-deploy`
- `azure/k8s-create-secret`
- `azure/k8s-bake`
- `azure/k8s-deploy`

# **Demo: Explore the use of GitHub Actions**

# Summary



# Summary

After completing this module, you know how to:

- Describe the principles of DevOps in cloud-native scenarios
- Implement DevOps principles using GitHub actions
- Build and deploy applications using GitHub workflows and Azure Pipelines