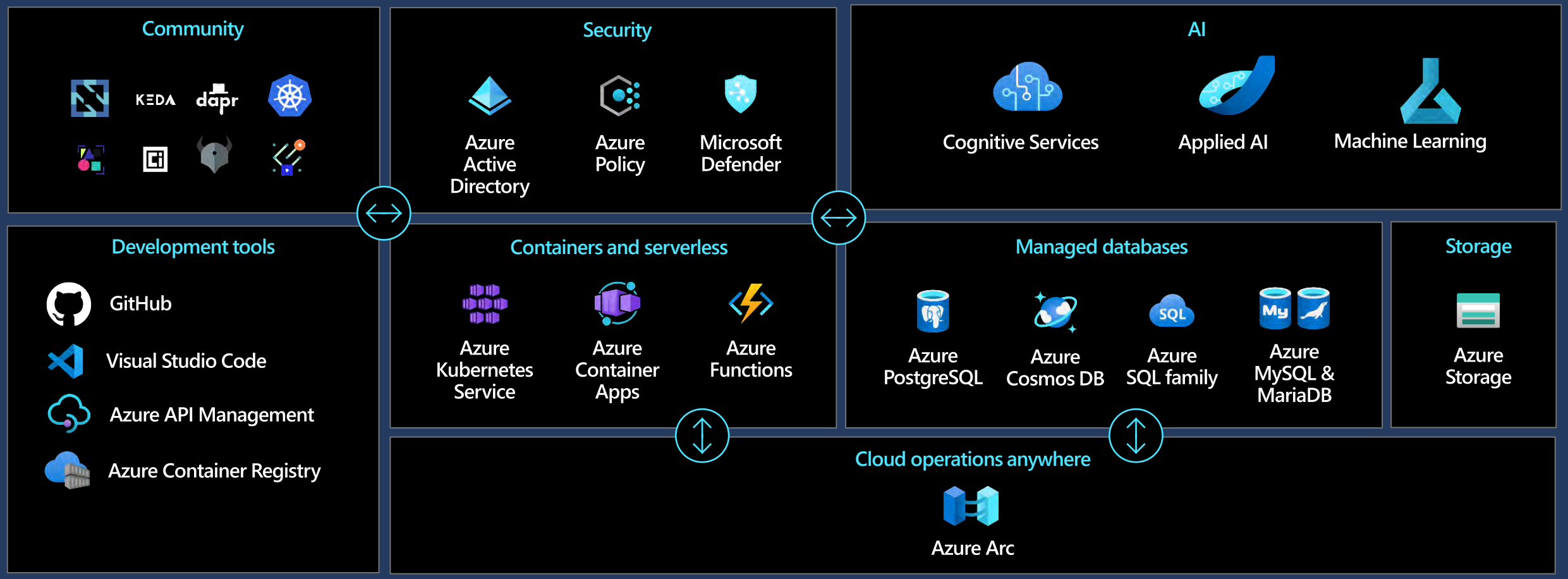
Microsoft

Azure Container Apps

Bojan Vrhovnik
Senior Cloud Solution Architect
bovrhovn@microsoft.com
@bvrhovnik

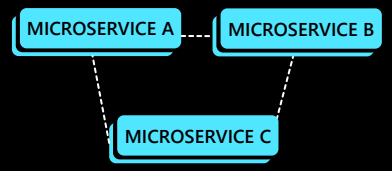
Kubernetes KEDA DAPR Envoy

Building cloud-native on Azure



What can you build with Azure Container Apps?

Microservices

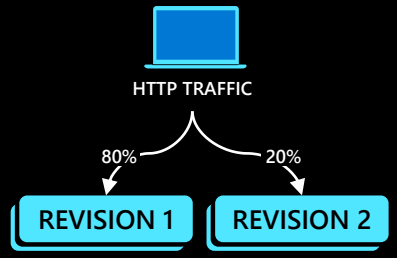


Microservices architecture with the option to integrate with Dapr

AUTO-SCALE CRITERIA

Individual microservices can scale independently using any KEDA scale triggers

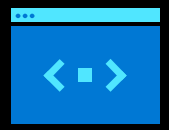
Public API endpoints



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

Scaling is determined by the number of concurrent HTTP requests

Web Apps



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

Scaling is determined by the number of concurrent HTTP requests

Event-driven processing

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

Scaling is determined by the number of messages in the queue

Background processing

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

Scaling is determined by the level of CPU or memory load

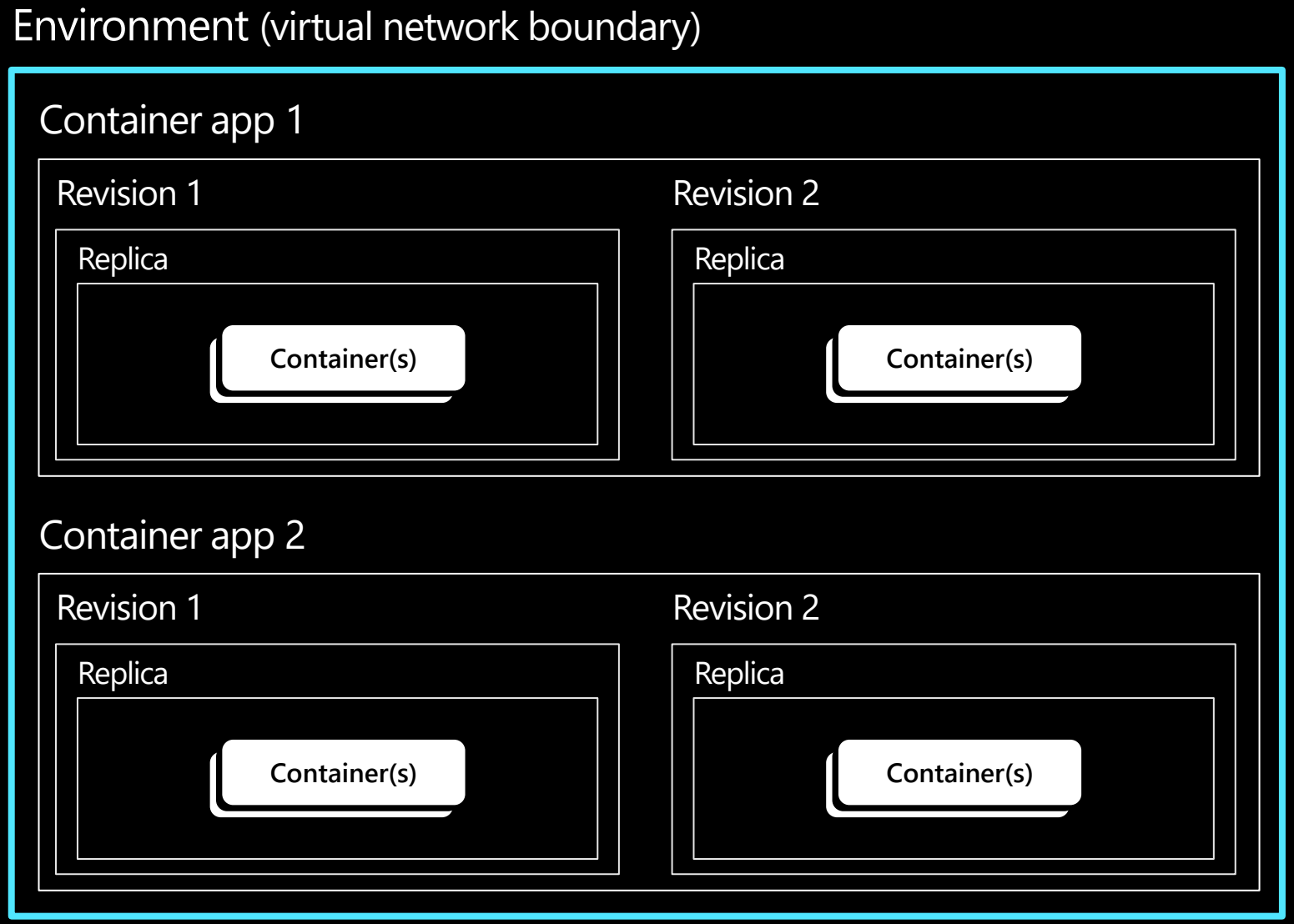
How does ACA compare to AKS?

	Azure Kubernetes Service (AKS) Infrastructure focus, higher flexibility	Azure Container Apps (ACA) Application focus, infrastructure abstraction
Core value proposition	Managed Kubernetes cluster in Azure with full access to the Kubernetes API server and high level of control over cluster configuration with a node-based pricing model	Fully-managed serverless abstraction on top of Kubernetes infrastructure, purpose built for managing and scaling event-driven microservices with a consumption-based pricing model
Optimized for	• Upstream feature parity with a managed control plane • Operations flexibility with advanced customization • Experienced Kubernetes operators	• Platform-as-a-Service experience with serverless scale • Developer productivity with low operations overhead • Linux-based, general-purpose stateless containers
Interaction model	• Operators deploy node-based AKS clusters using Azure Portal, CLI or Infrastructure-as-Code templates (IaC) • Developers deploy containers via Kubernetes deployment manifests or HELM charts to logically-isolated namespaces within the cluster	• Developers deploy containers as individual Container Apps using Azure Portal, CLI or IaC templates without any Kubernetes manifests required • Related container apps are deployed to a shared Container Apps environment comparable to a Kubernetes namespace
OSS Integration	• Provides a set of cluster extensions and add-ons for operators to enable OSS components in-cluster including Dapr, KEDA, Open Service Mesh, GitOps (Flux), Pod Identity, etc. • Supports manual installation via Kubernetes manifests	Includes opinionated platform capabilities powered by CNCF projects including Dapr, KEDA and Envoy which are fully platform-managed and supported • Envoy: managed ingress and traffic splitting • KEDA: managed, event-driven autoscale • Dapr: codified best practices for microservices

Demo
Getting started

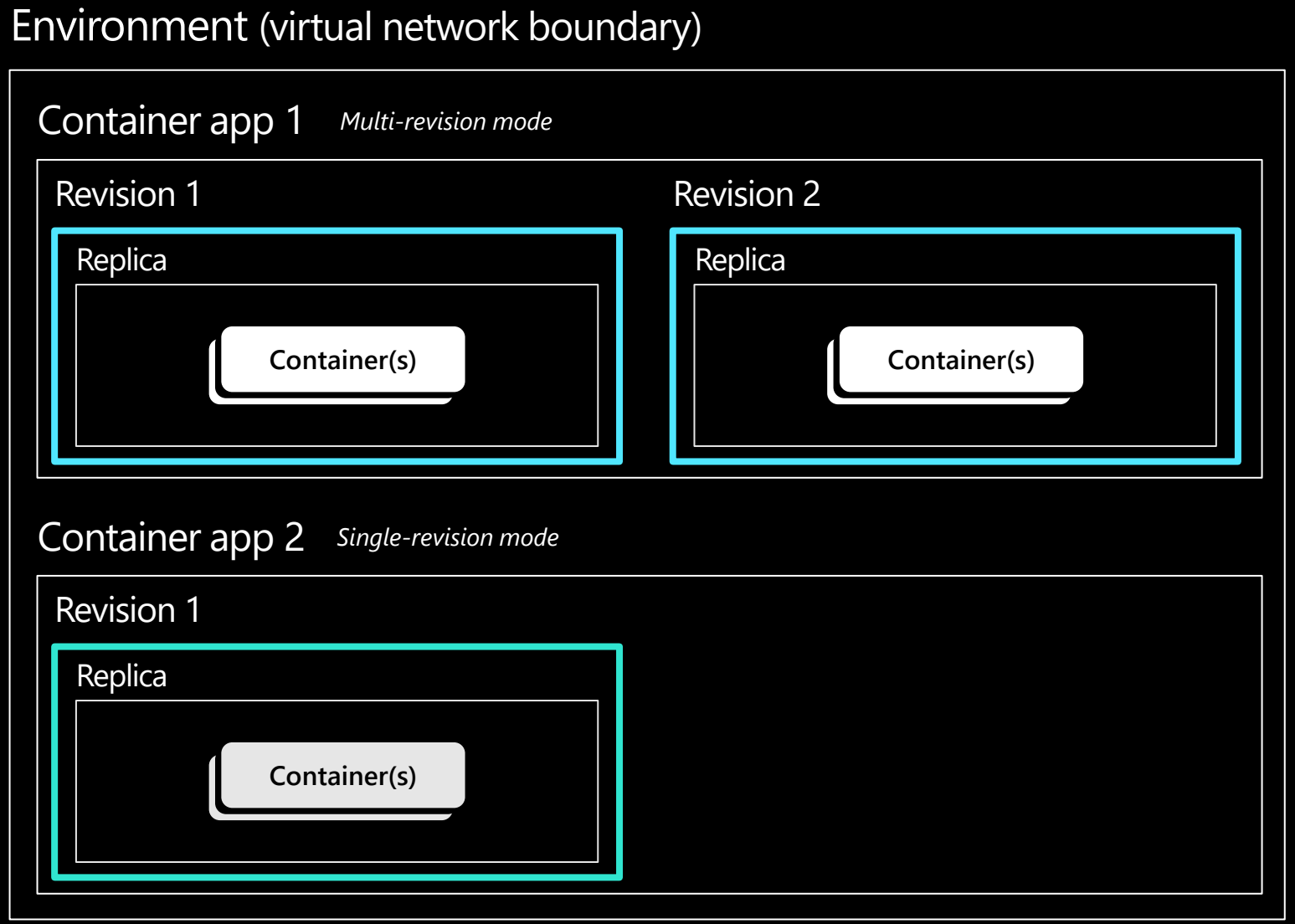
Environments

Environments define an isolation and observability boundary around a collection of container apps deployed in the same virtual network



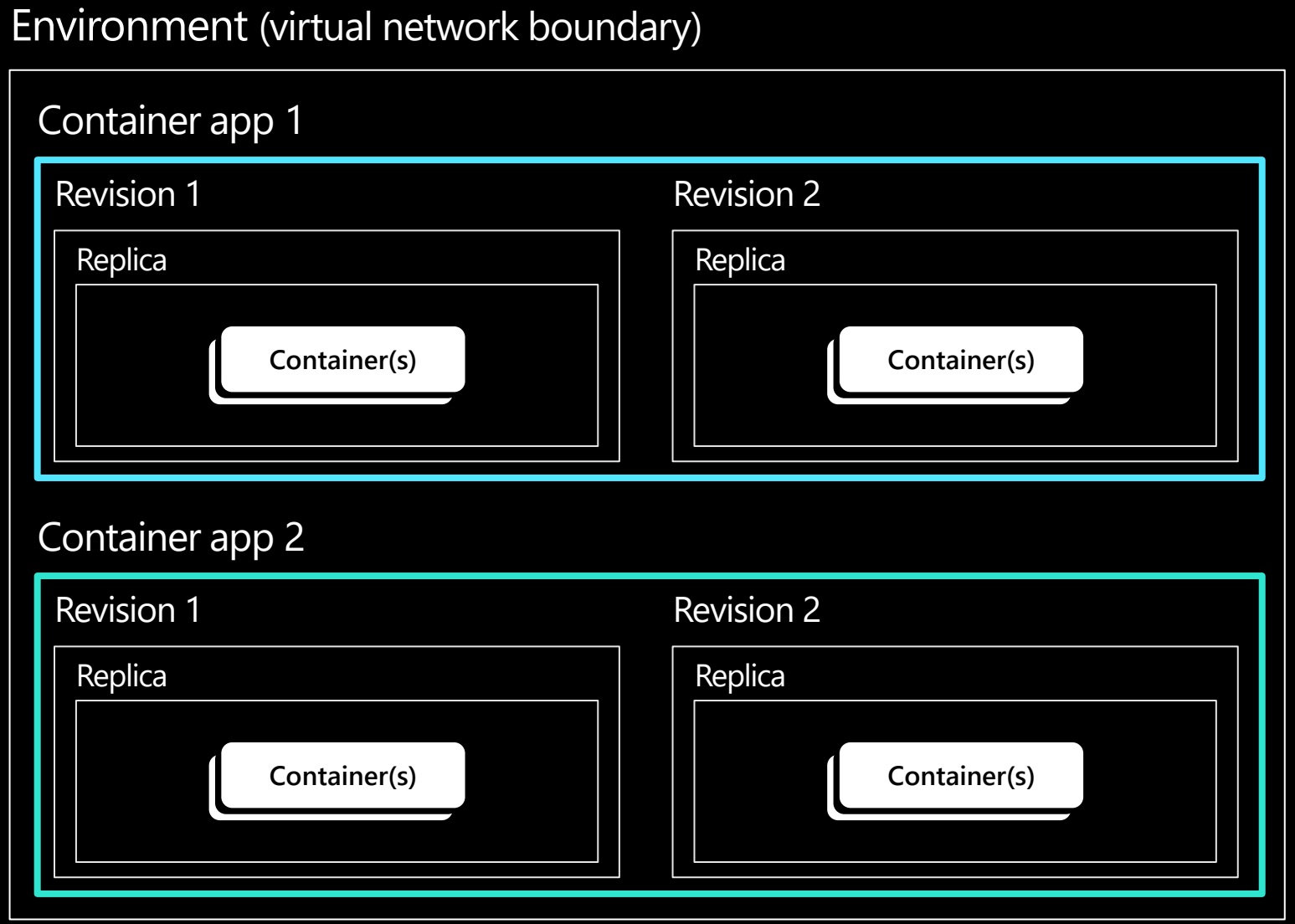
Revisions

Revisions are immutable version snapshots of a container app



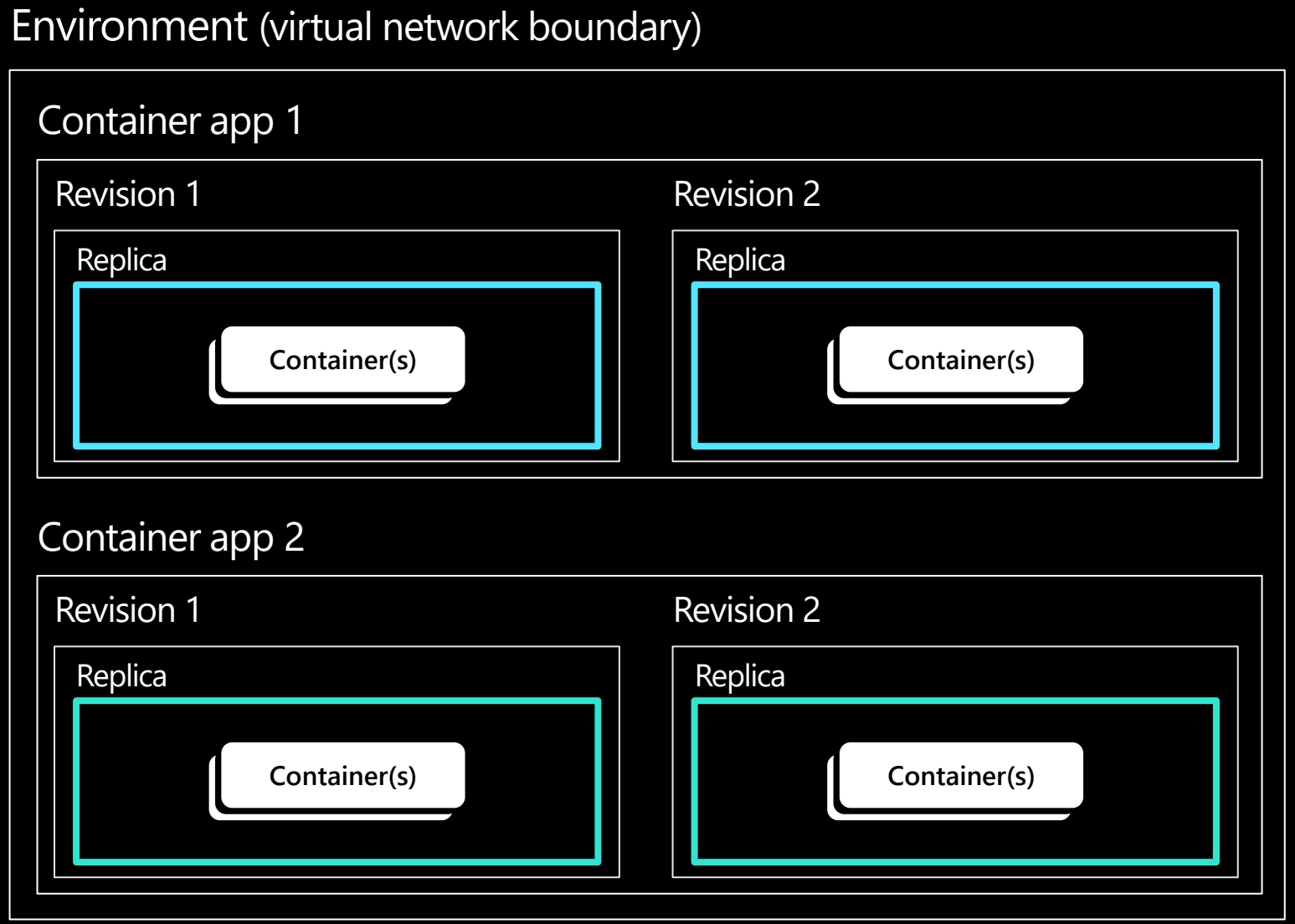
Container Apps

A Container App hosts a single, independent microservice and includes its desired state configuration



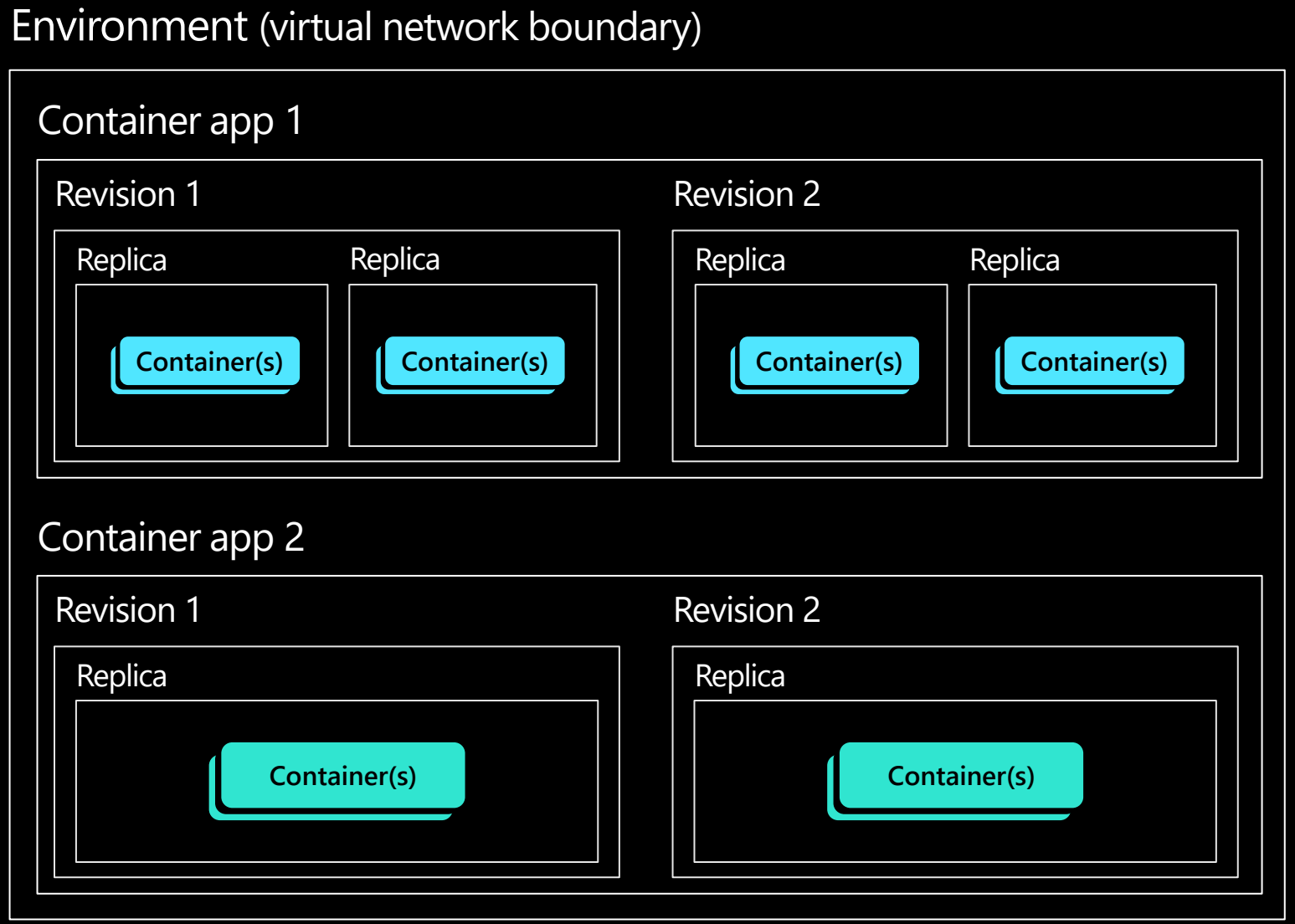
Replicas

Replicas are the unit of scale in container apps, with the default replica count being 0



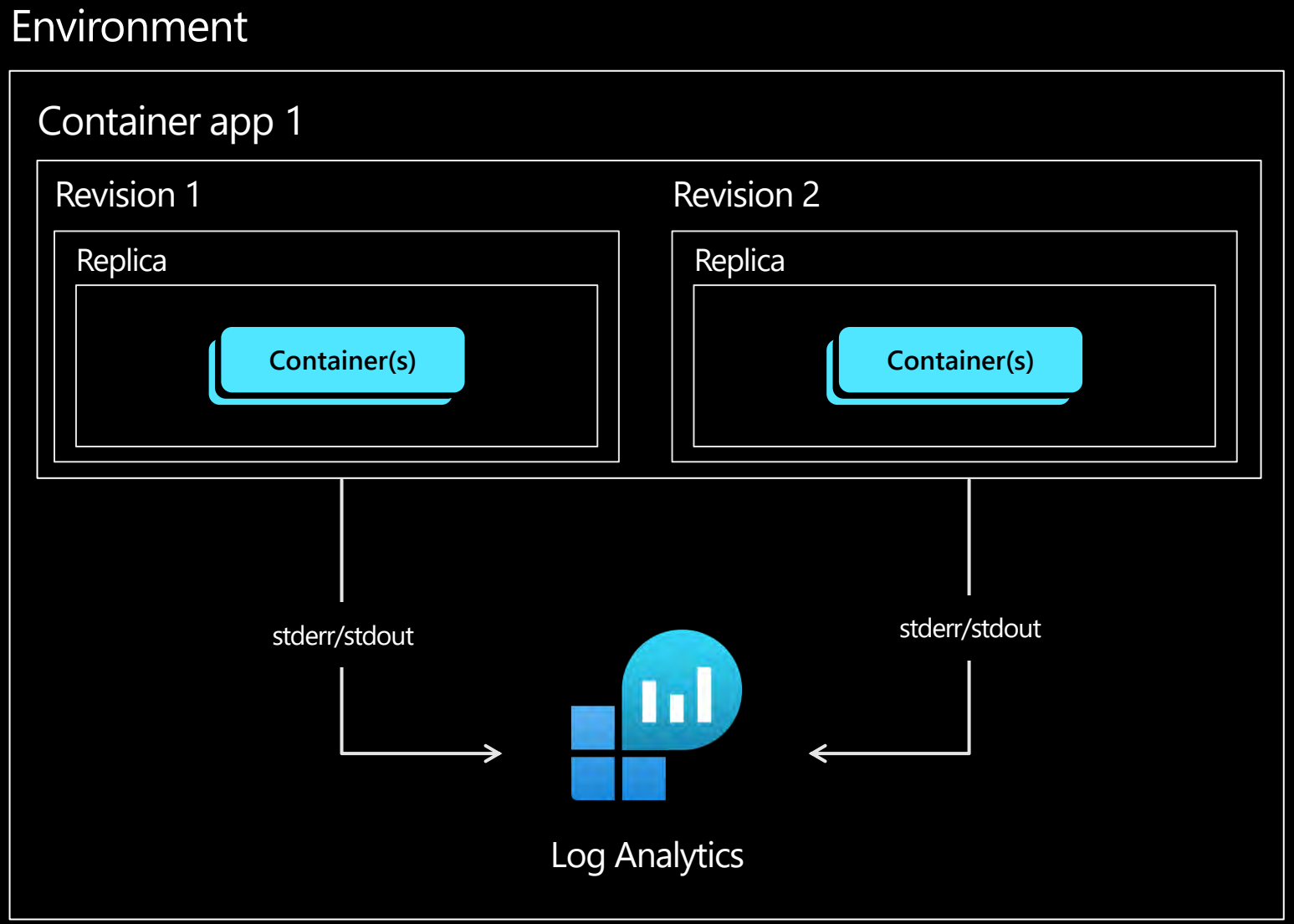
Containers

Containers in Azure Container Apps can use any development stack of your choice



Logging

Containers write logs to standard output or standard error streams surfaced via Log Analytics



Microsoft

Demo

Getting from local machine to the cloud

github.com/vrhovnik/conf24-2023-aca-demos

Microservice development challenges

- How do I **integrate with external systems** that my app has to react and respond to?
- How do I **create event driven apps** which reliably send events from one service to another?
- How do I create **long running, stateful services** that can recover from failures?
- How do I observe the calls and events between my services to **diagnose issues in production**?
- How do I **discover other services** and call methods on them?
- How do I **secure communication** between services?
- How do I **prevent committing to a technology** early and have the flexibility to swap out an alternative based on project or environment changes?

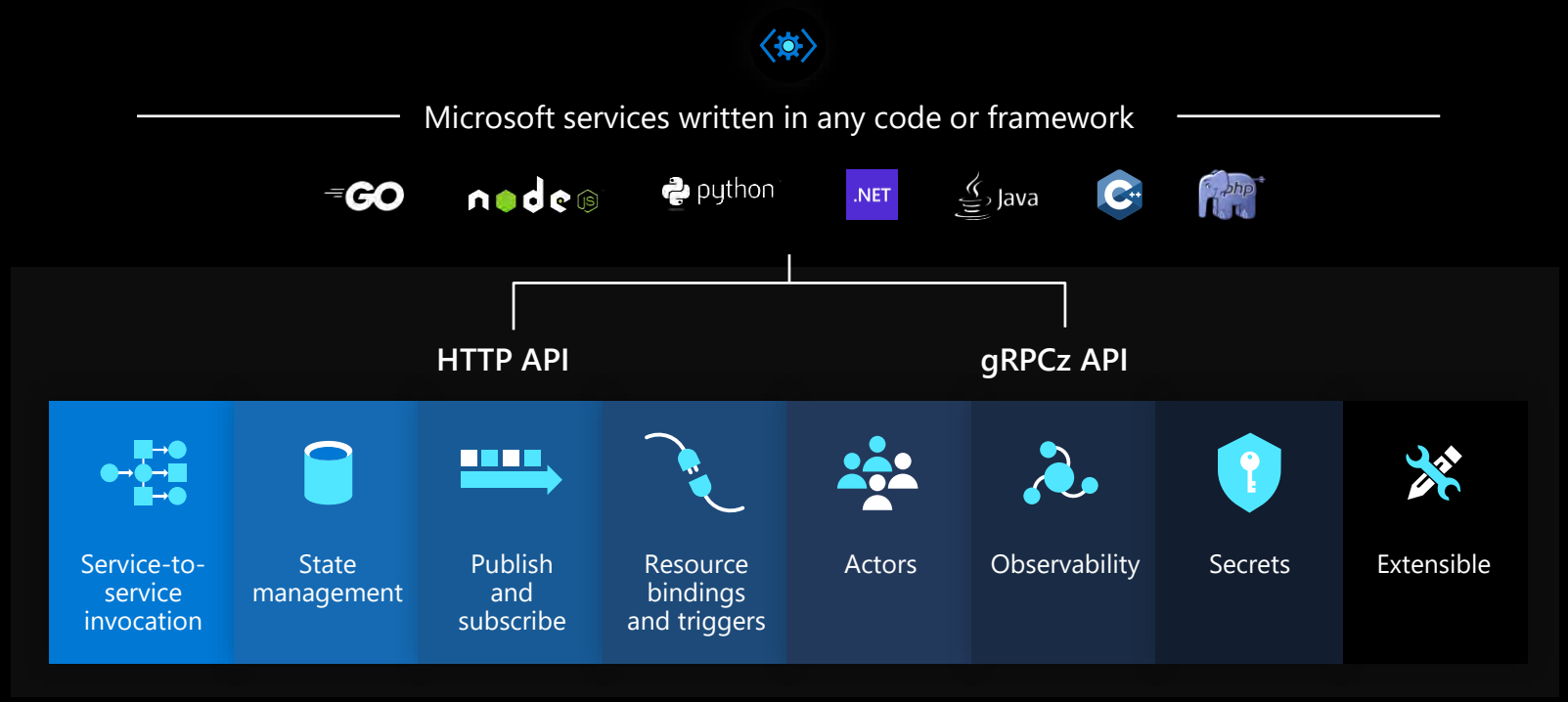
Microservices using any language or framework

Any cloud or edge infrastructure

dapr

Distributed Application Runtime

Portable, event-driven, runtime for building distributed applications across cloud and edge

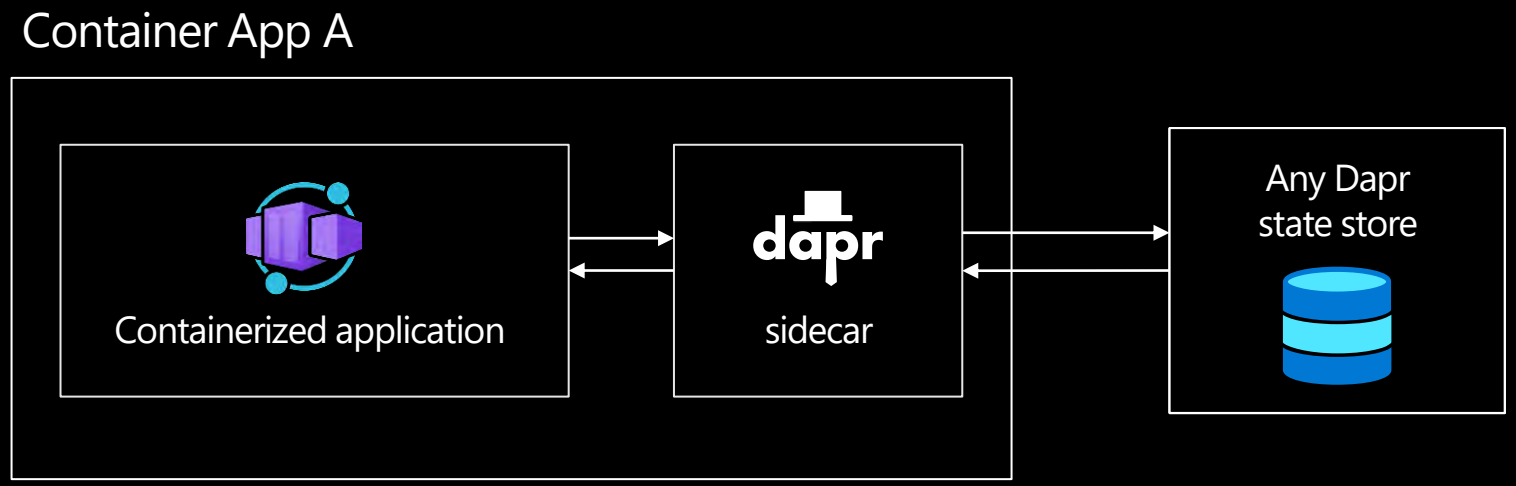


dapr.io

CLOUD NATIVE COMPUTING FOUNDATION

State management

Dapr provide apps with state management capabilities for CRUD operations, transactions and more




```
POST http://localhost:3500/v1.0/state/orders
```

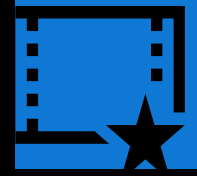
Demo
Dapr usage

Resources

Learn More about Azure Container Apps
aka.ms/containerapps

Deploy your first Container App
aka.ms/containerapps/deploy

Azure Container Apps documentation
aka.ms/containerapps/docs



Container Apps GitHub page
aka.ms/containerapps/github

Try for free
aka.ms/containerapps/tryfree