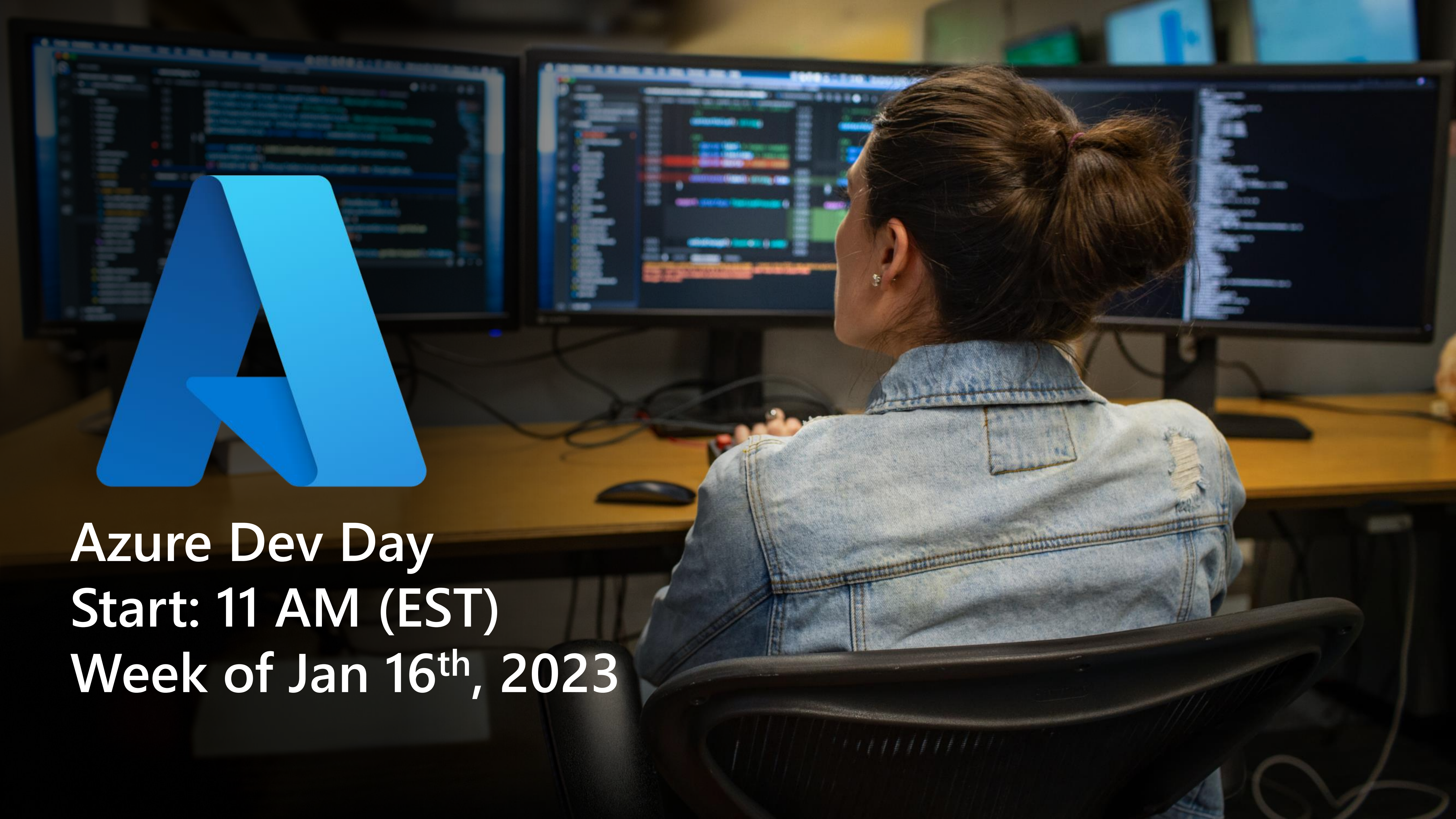


Azure Dev Day

Start: 11 AM (EST)

Week of Jan 16th, 2023

Azure Dev Day

Gary T. Ciampa

Cloud Solution Architect

✉ Gary.Ciampa@microsoft.com

in Gary @ Linkedin

Gary.Ciampa@github

A ❤ Devs

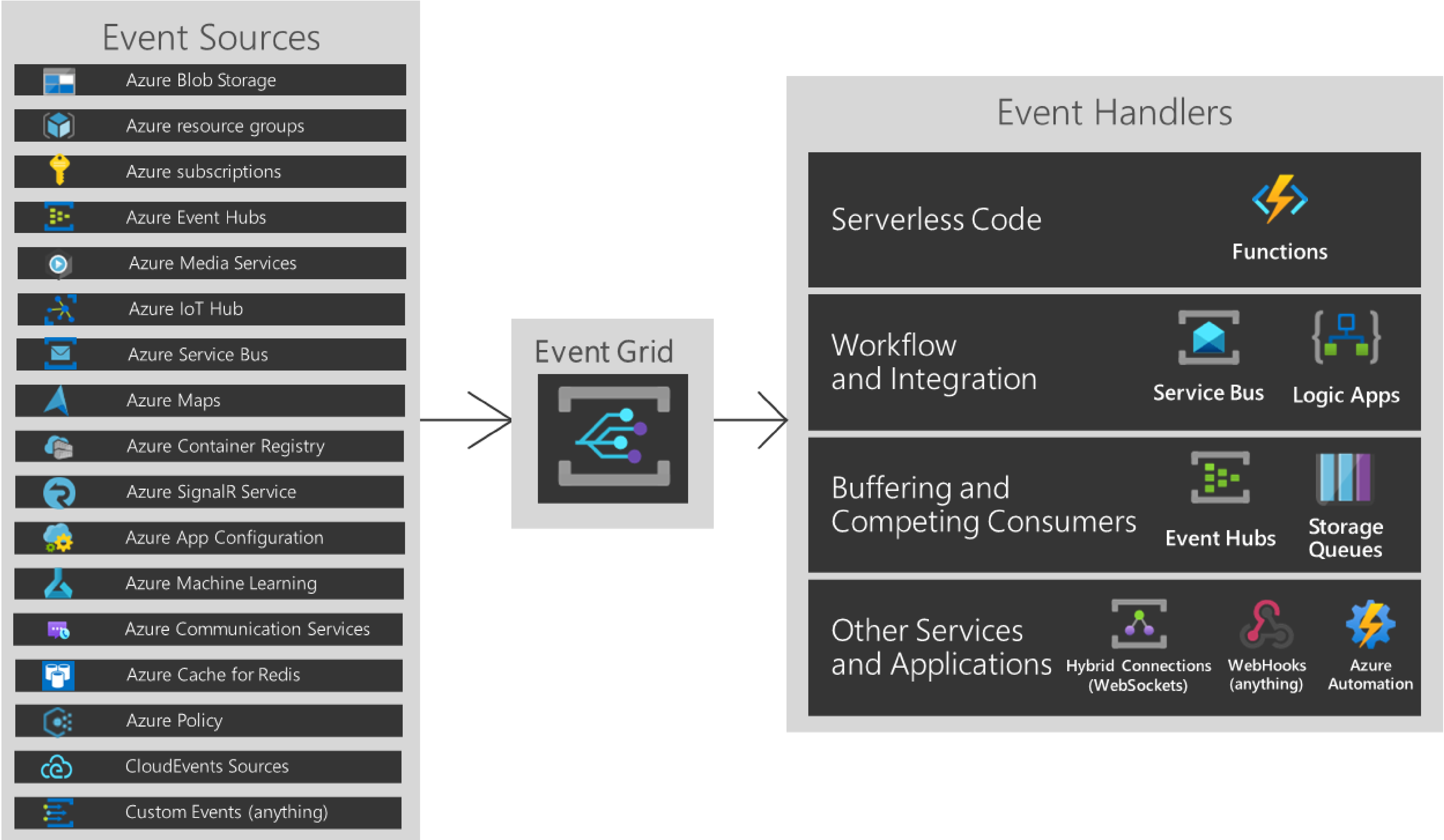
Agenda (11:00 AM EST)

Day 1: 16 JAN	Azure introduction & fundamentals
	Web-based solutions (Presentation & Lab)
Day 2: 17 JAN	Serverless, event-driven solutions (Presentation & Lab)
Day 3: 19 JAN	Azure Kubernetes Services (Presentation & Lab)
Day 4: 20 JAN	DevOps for deploying solutions
	Kahoot Trivia – Microsoft SWAG

Day 3:

Prologue

- Day 1: Azure App Service notes & discussion
- Day 2: Azure Serverless introduction, event grid, functions, cosmos db
- Event Grid sources & handlers
- Azure APIM workshop demo (aka.ms/apimlove)
- **Kahoot.it – For the win, MSFT store swag (Azure App Services & Serverless)**



Event Sources

- Azure Blob Storage
- Azure resource groups
- Azure subscriptions
- Azure Event Hubs
- Azure Media Services
- Azure IoT Hub
- Azure Service Bus
- Azure Maps
- Azure Container Registry
- Azure SignalR Service
- Azure App Configuration
- Azure Machine Learning
- Azure Communication Services
- Azure Cache for Redis
- Azure Policy
- CloudEvents Sources
- Custom Events (anything)

Event Grid

Event Handlers

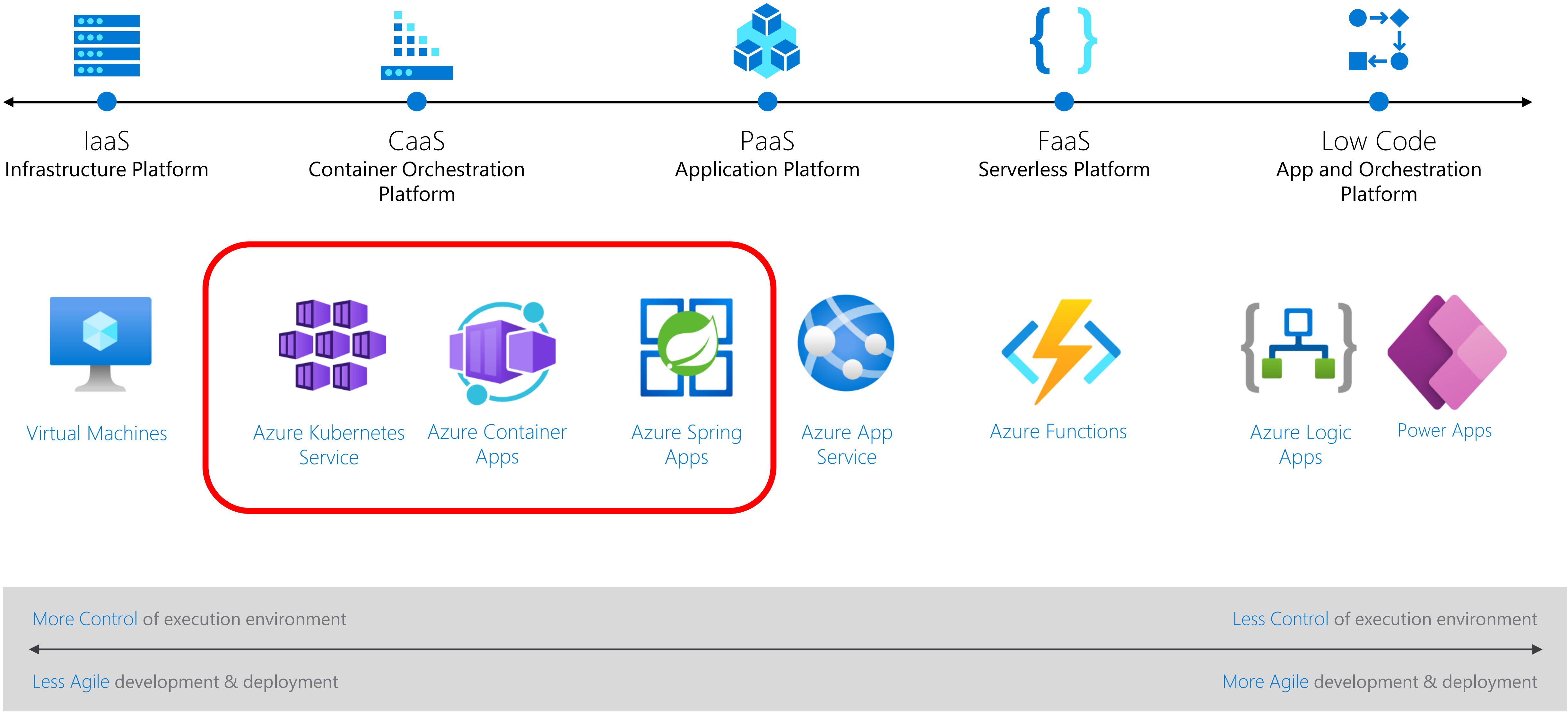
Serverless Code	Functions
Workflow and Integration	Service Bus Logic Apps
Buffering and Competing Consumers	Event Hubs Storage Queues
Other Services and Applications	Hybrid Connections (WebSockets) WebHooks (anything) Azure Automation

3

Microservice Solutions

Develop and deploy microservices using Azure Container Apps and Azure Container Registry

Application hosting continuum



IaaS
Infrastructure Platform

CaaS
Container Orchestration Platform

PaaS
Application Platform

FaaS
Serverless Platform

Low Code
App and Orchestration Platform

Virtual Machines

Azure Kubernetes Service

Azure Container Apps

Azure Spring Apps

Azure App Service

Azure Functions

Azure Logic Apps

Power Apps

More Control of execution environment

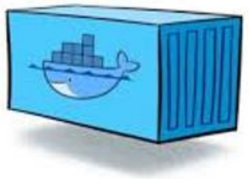
Less Control of execution environment

Less Agile development & deployment

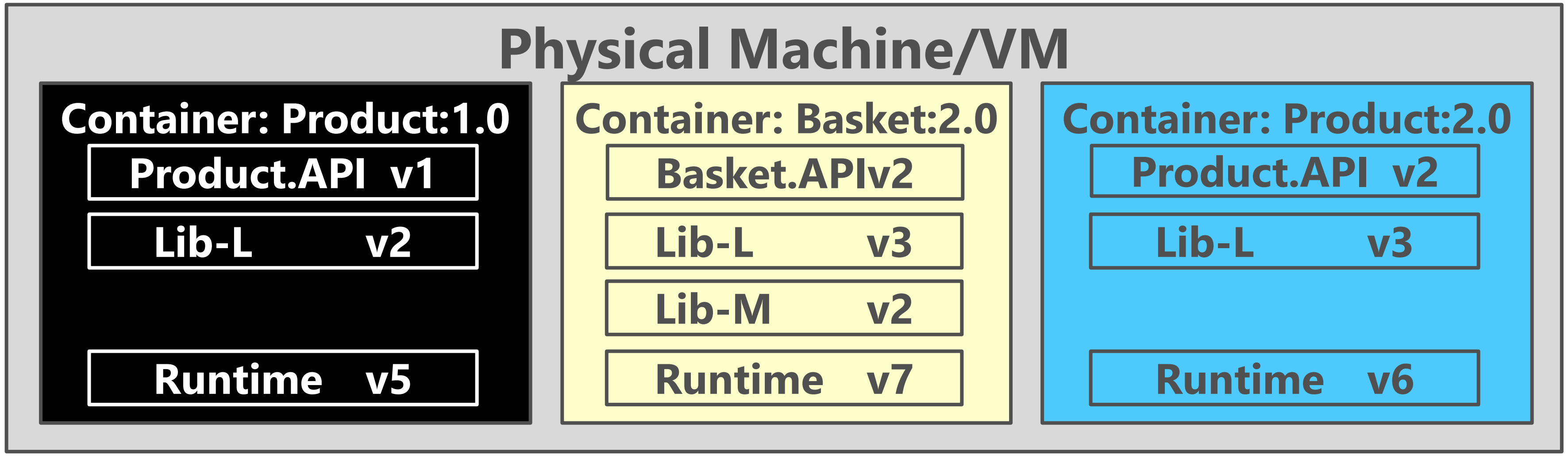
More Agile development & deployment

Overview of Containers

What is a Container?



- Portable unit of deployment
- Application code and dependencies compartmentalized
- Virtualization without the need of a VM overhead
- Best practice to organize one service/container



Physical Machine/VM

Container: Product:1.0
Product.API v1
Lib-L v2
Runtime v5

Container: Basket:2.0
Basket.APIv2
Lib-L v3
Lib-M v2
Runtime v7

Container: Product:2.0
Product.API v2
Lib-L v3
Runtime v6

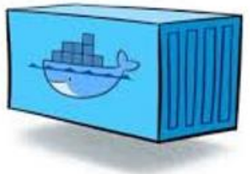
What Problems Do Containers Solve?

- Guarantees consistency across DEV, TEST and PROD
- Increases Productivity
- Isolation & Performance
- Smaller footprint than VMs

Containers are a great environment for deploying Microservices

Moving to Microservices

Defining Microservices?

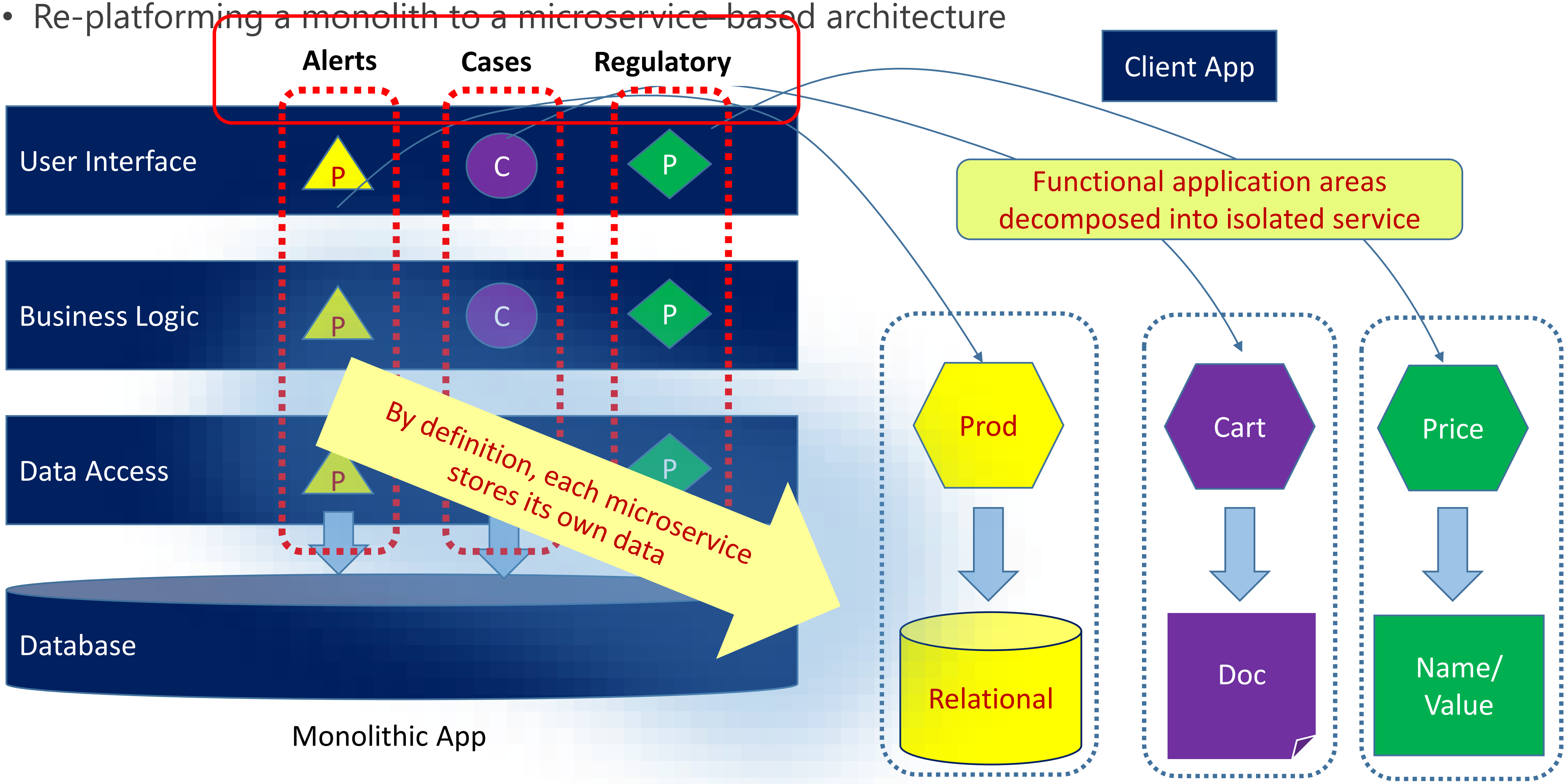


- An approach to application development in which a large system is built as a suite of modular services
- Each service supports a specific business goal (capability) – a single concern
- Each service is fully independent and self-contained, exposing a well-defined interface to communicate with other services
- Each encapsulates its own data and chooses its underlying data store
- Embracing cross-platform, each can be written leveraging a different programming platform
- Each can deploy frequently and evolve independently, composing with others to form an application

Microservice Architecture

Moving to Microservices

- Re-platforming a monolith to a microservice-based architecture

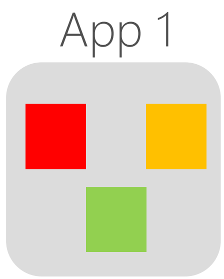


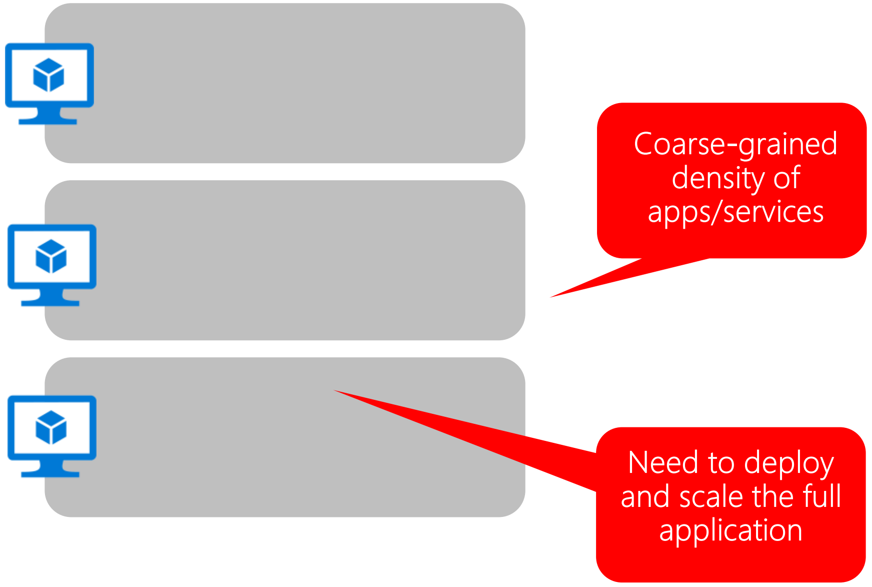
Functional application areas decomposed into isolated service

By definition, each microservice stores its own data

Traditional application approach

- A traditional application has most of its functionality within a few processes that are componentized with layers and libraries.
- Scales by cloning the app on multiple servers/VMs



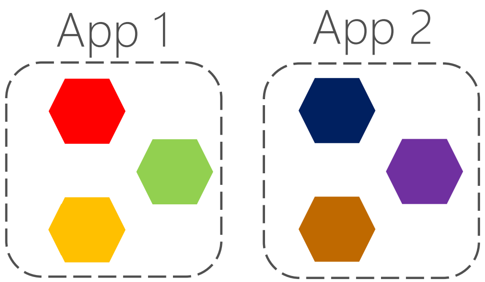


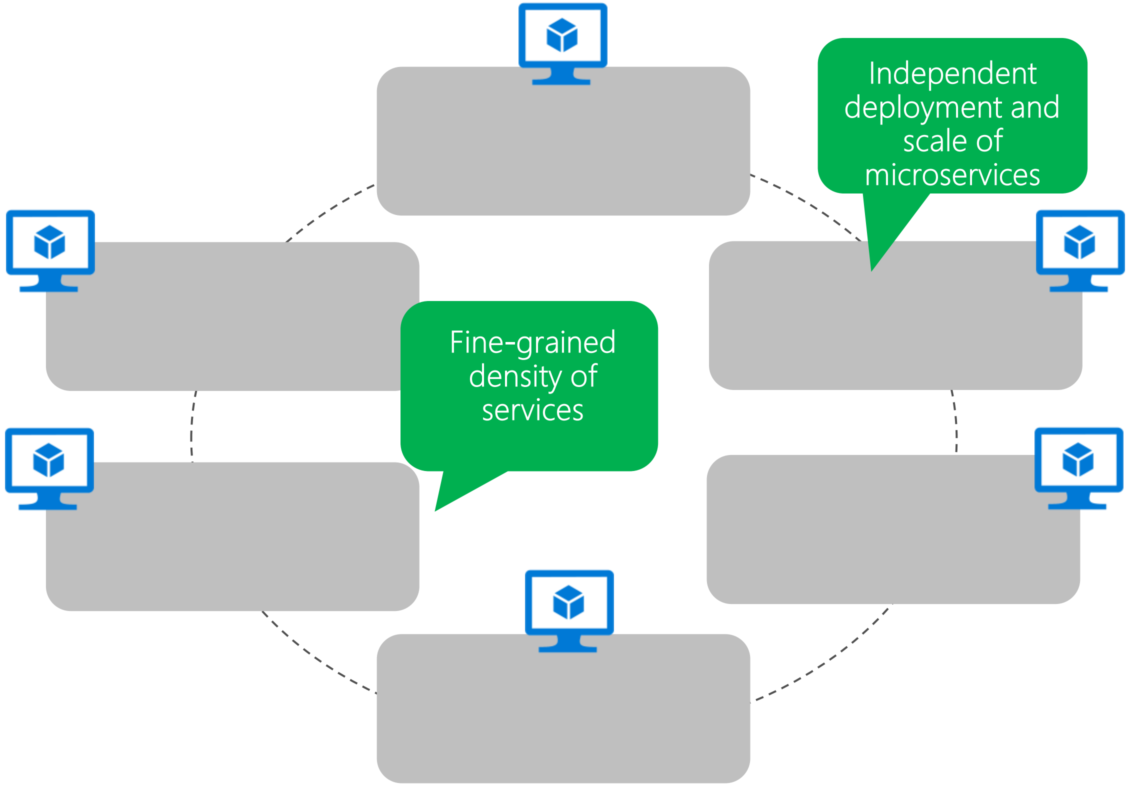
Coarse-grained density of apps/services

Need to deploy and scale the full application

Microservices application approach

- A microservice application segregates functionality into separate smaller services.
- Scales out by **deploying each service independently** with multiple instances across servers/VMs



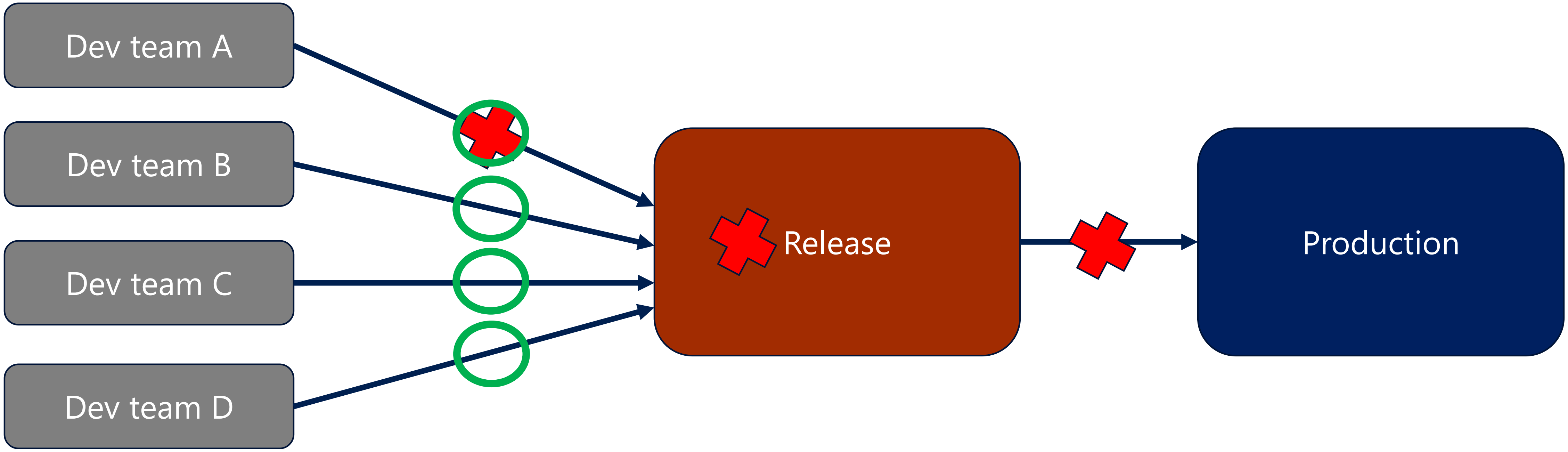


Independent deployment and scale of microservices

Fine-grained density of services

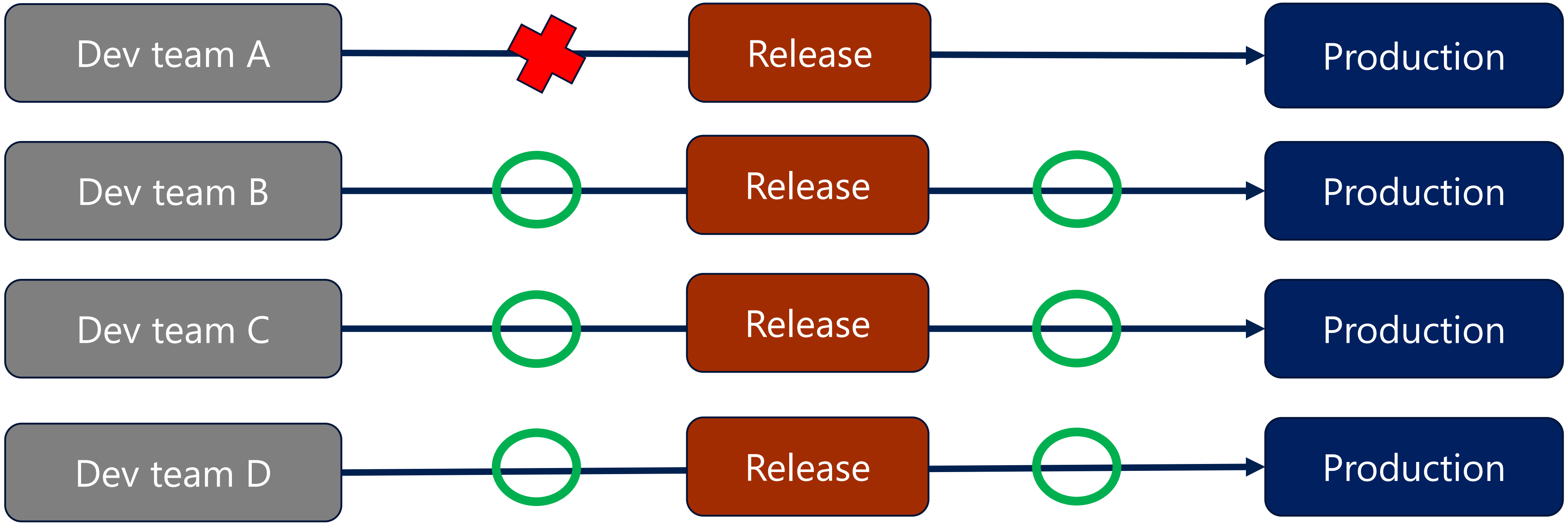
How Monoliths Diminish Agility

- Single codebase - singe release pipeline
  - All teams *share code base/dependencies* – tightly-coupled
  - All team *share same release cadence*
  - A defect in a dependency can block multiple teams and the release itself

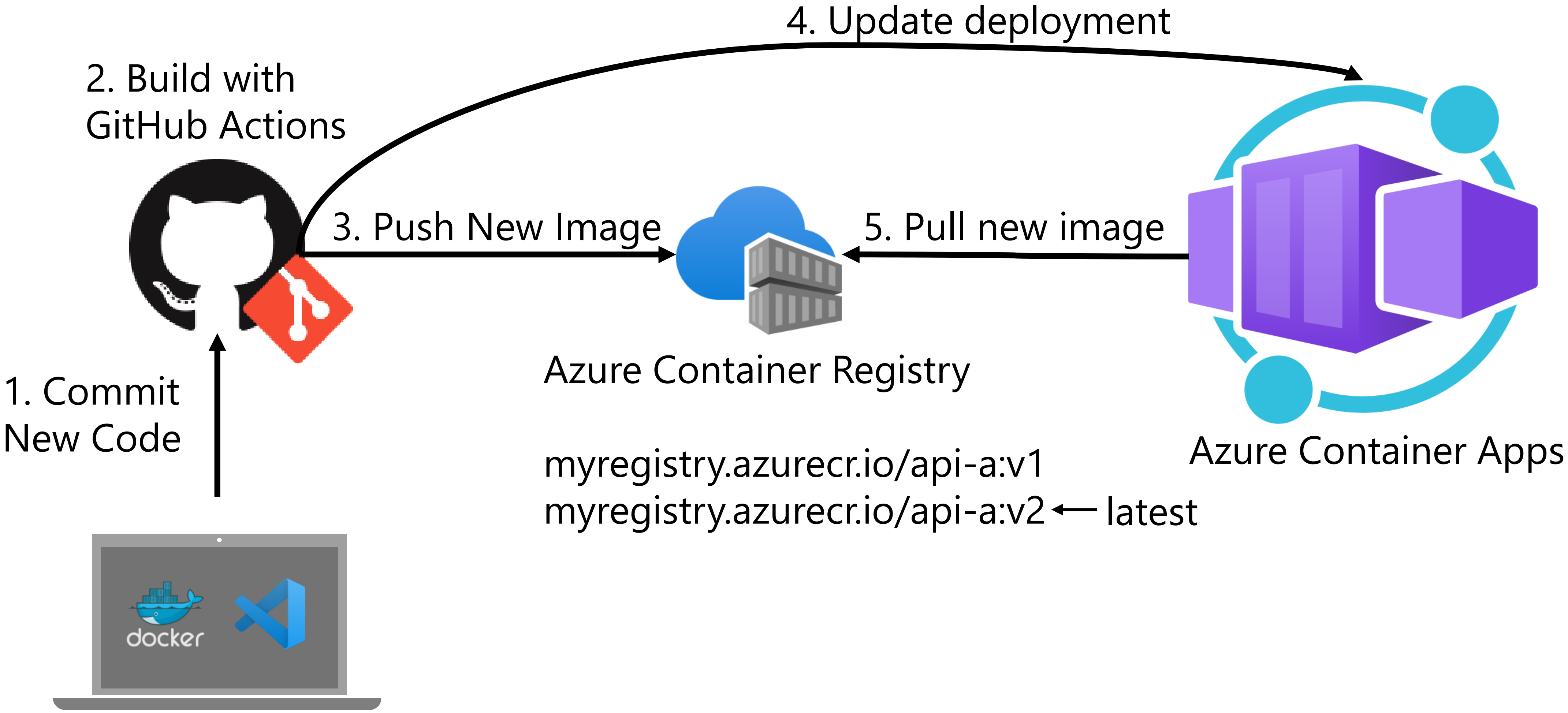


How Microservices Promote Agility

- Each team owns it own service and codebase…
  - Services are *isolated* and *do not directly share dependencies*
  - Each service has its *own release cadence*
  - Each *deploys independently*



Azure Container Registry



myregistry.azurecr.io/api-a:v1
myregistry.azurecr.io/api-a:v2 ⟵ latest

Kubernetes - Container Orchestrator

Challenges of a containerized world

As application development has moved towards a container-based approach, the need to orchestrate and manage the inter-connected resources becomes important

- Load Balancing
- Naming and Discovery
- Logging and Monitoring
- Debugging and Introspection
- Networking

The elements of orchestration

Scheduling

Affinity/anti-affinity

Health monitoring

Failover

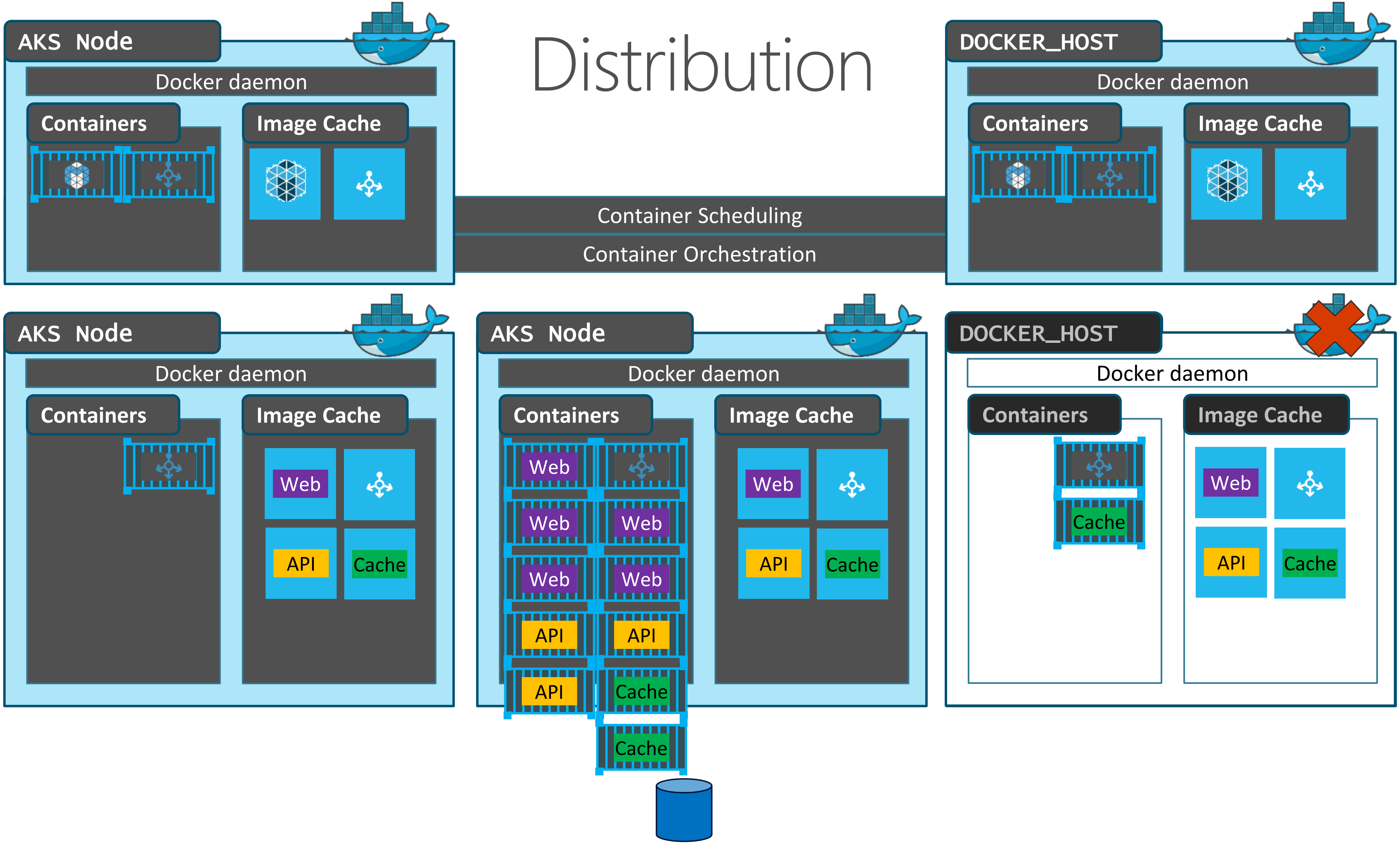
Scaling

Networking

Service discovery

Coordinated app upgrades

Distribution



Azure Kubernetes Service

Azure Kubernetes Service (AKS)

Fully-managed Kubernetes platform hosted in Azure as a PaaS service

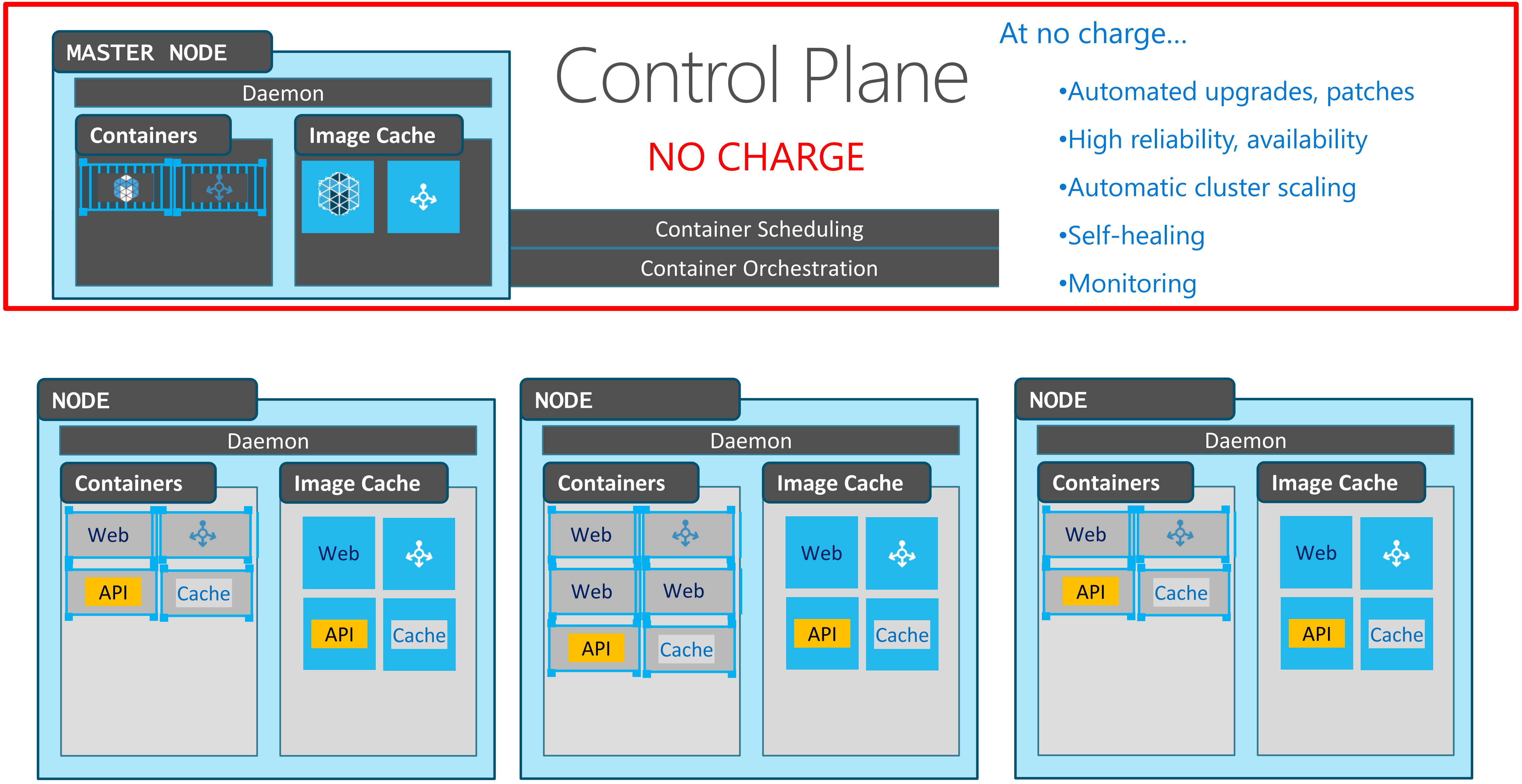
Deeply integrated with Azure dev tools and services

Abstracts the complexity and operational overhead of managing Kubernetes

- AKS implements K8S services, with a custom K8S config file optimized for Azure
- AKS is a K8s managed service w/in Azure

At no charge…

- Automated upgrades, patches
- High reliability, availability
- Automatic scaling
- Self-healing
- Monitoring



Control Plane

NO CHARGE

At no charge…

- Automated upgrades, patches
- High reliability, availability
- Automatic cluster scaling
- Self-healing
- Monitoring

MASTER NODE

Daemon

Containers

Image Cache

Container Scheduling

Container Orchestration

NODE

Daemon

Containers

Web

API

Cache

Image Cache

Web

API

Cache

NODE

Daemon

Containers

Web

Web

Web

API

Cache

Image Cache

Web

API

Cache

NODE

Daemon

Containers

Web

API

Cache

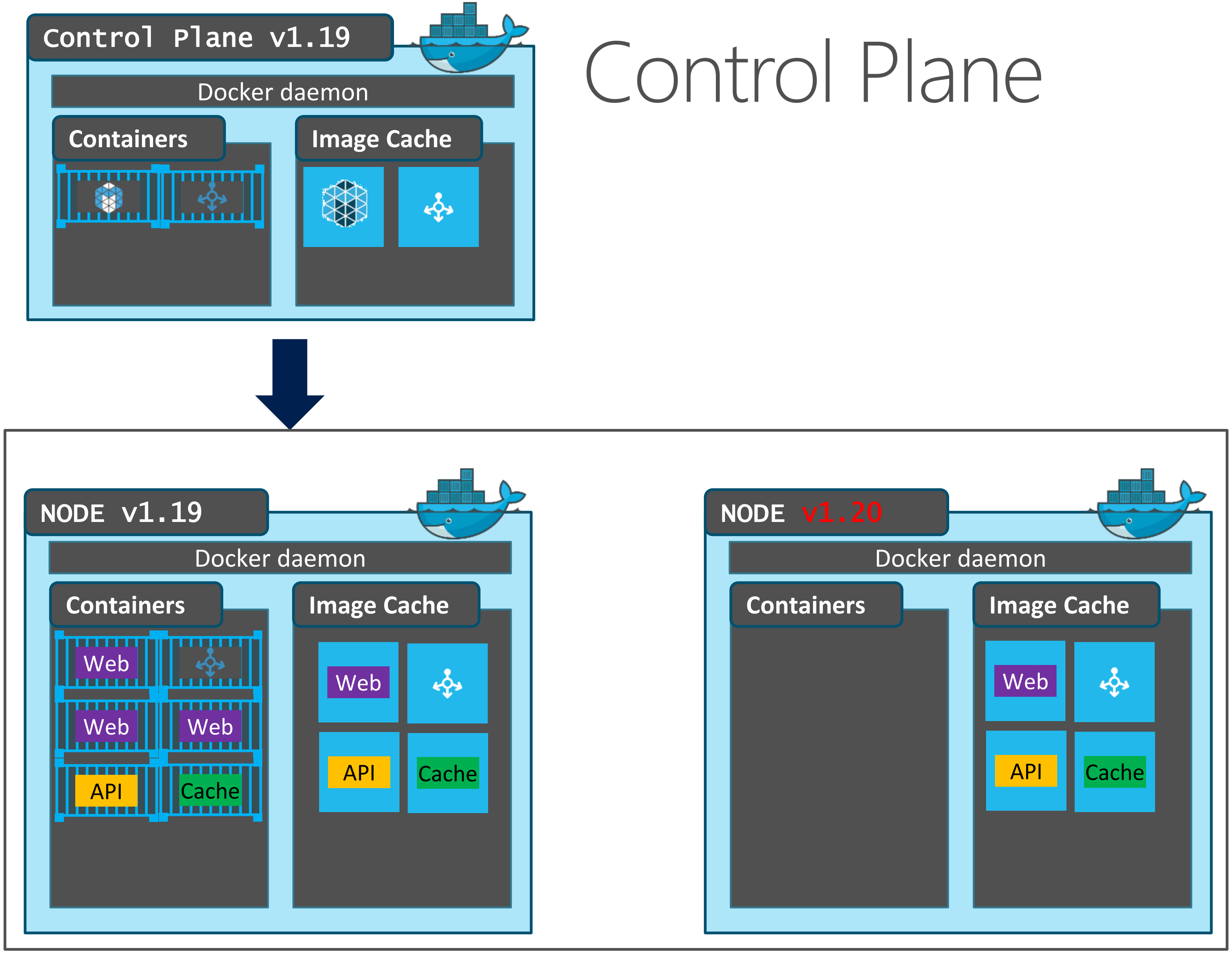
Image Cache

Web

API

Cache

Control Plane



At no charge…

- Automated upgrades, patches
- High reliability, availability
- Automatic cluster scaling
- Self-healing
- Monitoring

AKS Features

High Availability
High Reliability

Availability Zones
99.95% SLA
Self-Healing

Cluster Autoscaler

Node Autoscaler
Virtual Nodes

Security

Azure Key Vault
Azure Active Directory
Private Clusters

Monitoring

Azure Log analytics with Container Insights

AKS – References

Documentation, learn, best practices, industry use cases

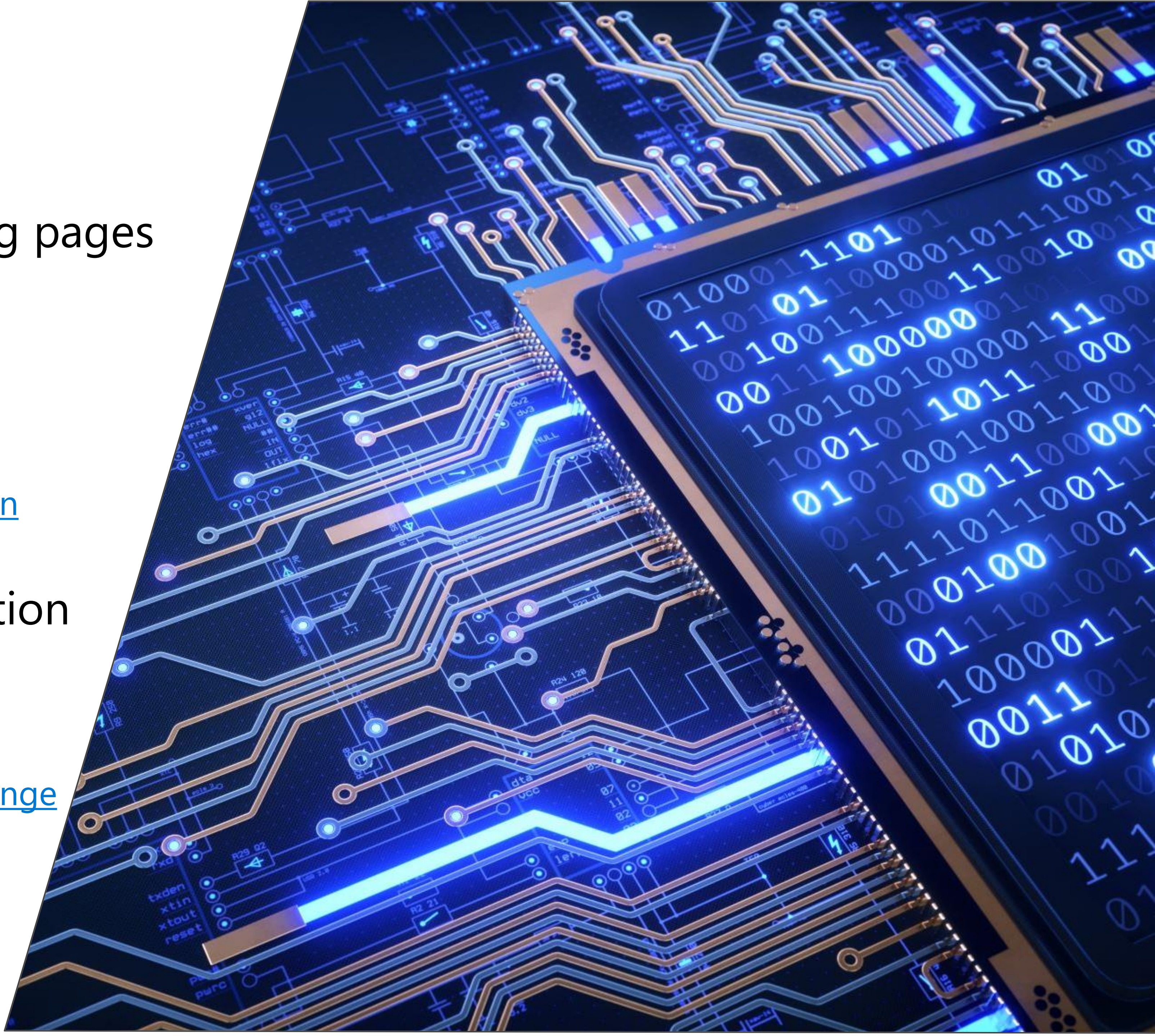
AKS References

Azure Kubernetes Service landing pages

- Azure Kubernetes Service portal
- Azure Kubernetes Service pricing
- Azure Kubernetes Service documentation

Azure Kubernetes Service education

- Azure Kubernetes Service-learning path
- Azure Kubernetes Service 50 days challenge
- Azure Developer Cloud Skills Challenge



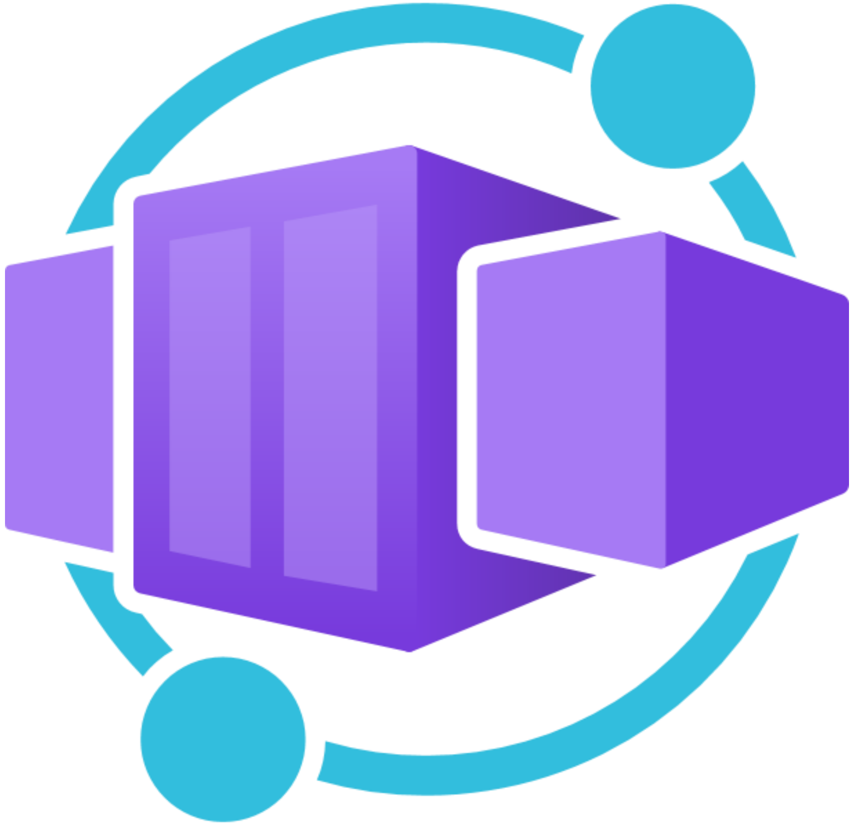
Azure Container Apps

Serverless Container Hosting

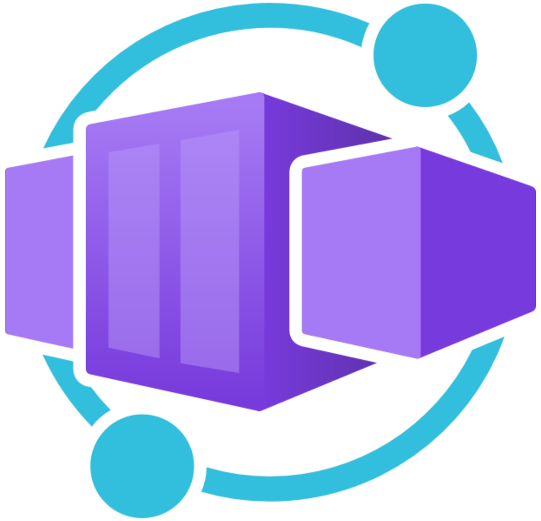
Robust scaling with KEDA scale triggers

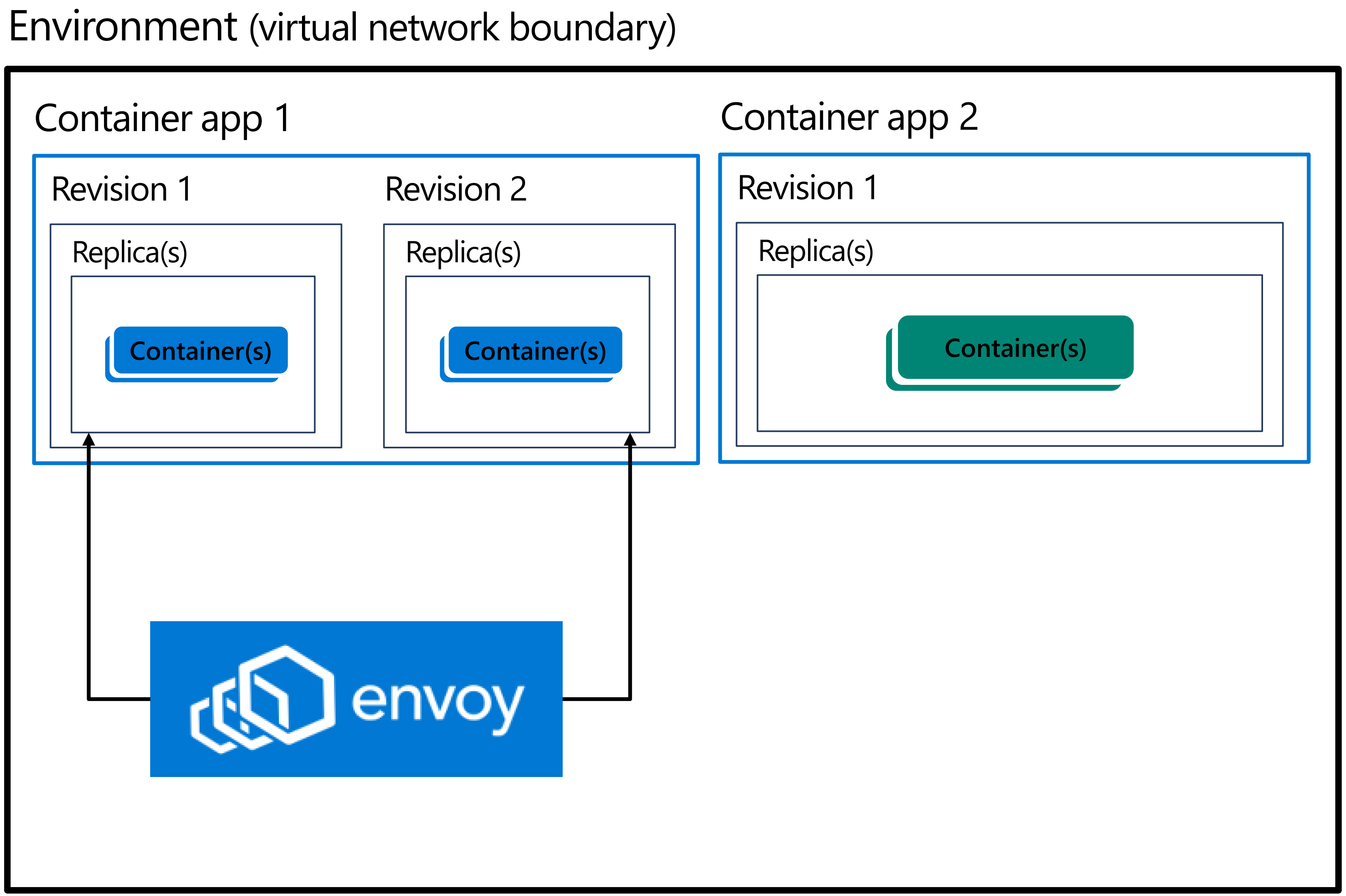
Built-in Dapr integration

Multiple revisions per app



ACA Concepts





Environment (virtual network boundary)

Container app 1

Revision 1

Replica(s)

Container(s)

Revision 2

Replica(s)

Container(s)

Container app 2

Revision 1

Replica(s)

Container(s)

Microservice Solutions Lab

Azure Dev Day