# Running Microservices in Azure Kubernetes Service

VAIBHAV GUJRAL
CLOUD ARCHITECT | MICROSOFT AZURE MVP

# About me

Cloud & DevOps Architect with over 15 years of experience

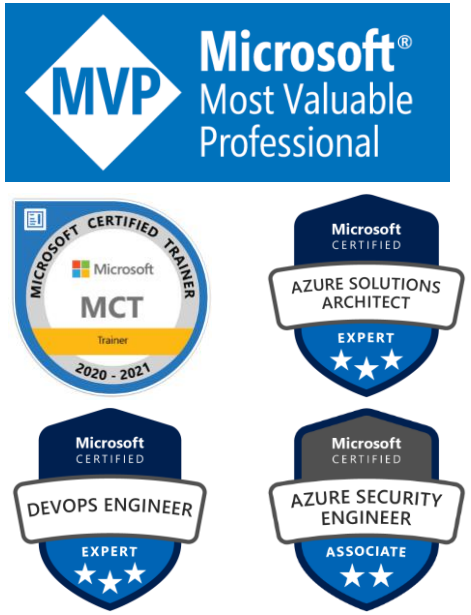
Born and brought up in India and moved to Omaha, NE in 2016

Microsoft Azure MVP and Microsoft Certified Trainer

Leader, Omaha Azure User Group([https://omahaazure.org](https://omahaazure.org))

Microsoft Certified Azure Solutions Architect Expert & others

AWS Certified Cloud Practitioner



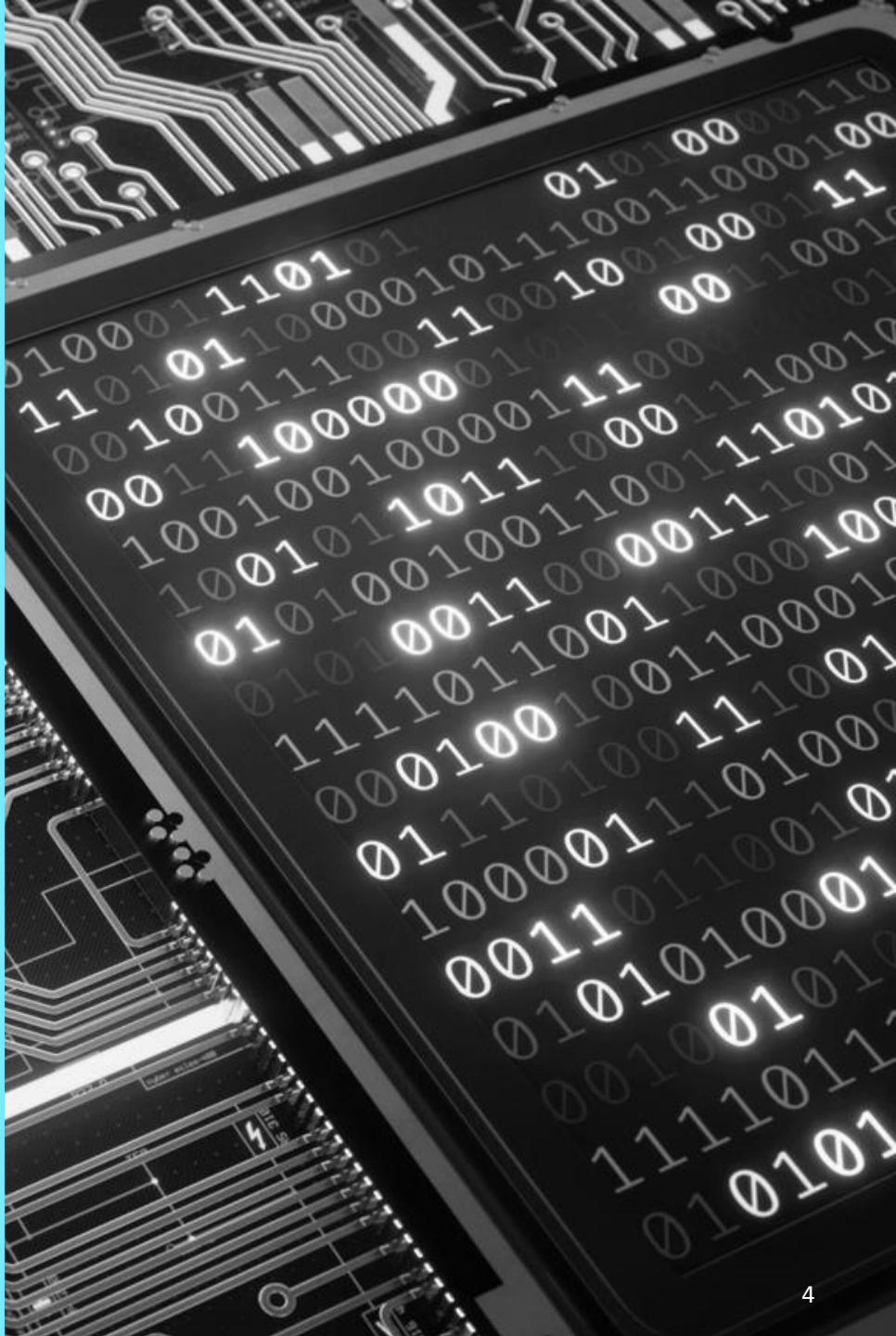
# Technology Evolution

| | Development Process | Application Architecture | Deployment Model | Infrastructure |
|---|---|---|---|---|
| Now | DevOps | Microservices | Containers | Cloud |
| Late 2000's | Agile | N-Tier | Virtual Machines | Hosted |
| 1990's and early 2000s | Waterfall | Monolithic | Physical Servers | Data Center |

# What are microservices?

*The microservice architectural style is an approach to developing a single application as a **suite of small services, each running in its own process** and communicating with lightweight mechanisms, often an HTTP resource API. These services are **built around business capabilities** and **independently deployable** by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be **written in different programming languages** and **use different data storage technologies**.*

—James Lewis & Martin Fowler



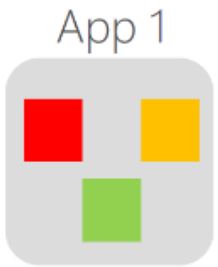
# What are microservices?

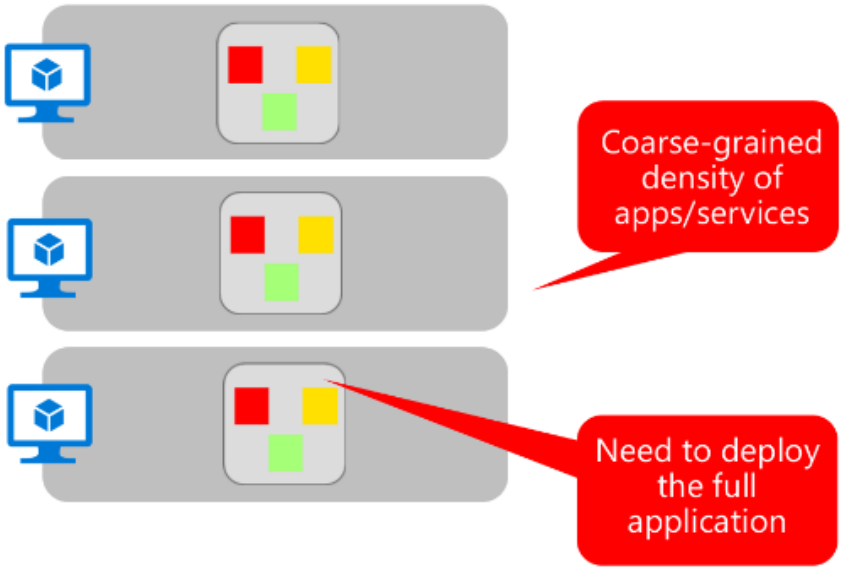
*Microservices are –*

1. *Suite of small services, each running in its own process*
2. *Built around business capabilities*
3. *Are independently deployable*
4. *May be written in different programming languages*
5. *Use different data storage technologies*

# Monolithic deployment approach

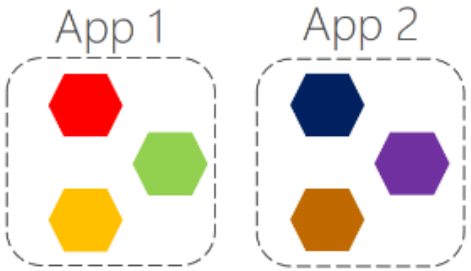
- A traditional application has most of its functionality within a few processes that are componentized with layers and libraries.
- Scales by cloning the app on multiple servers/VMs

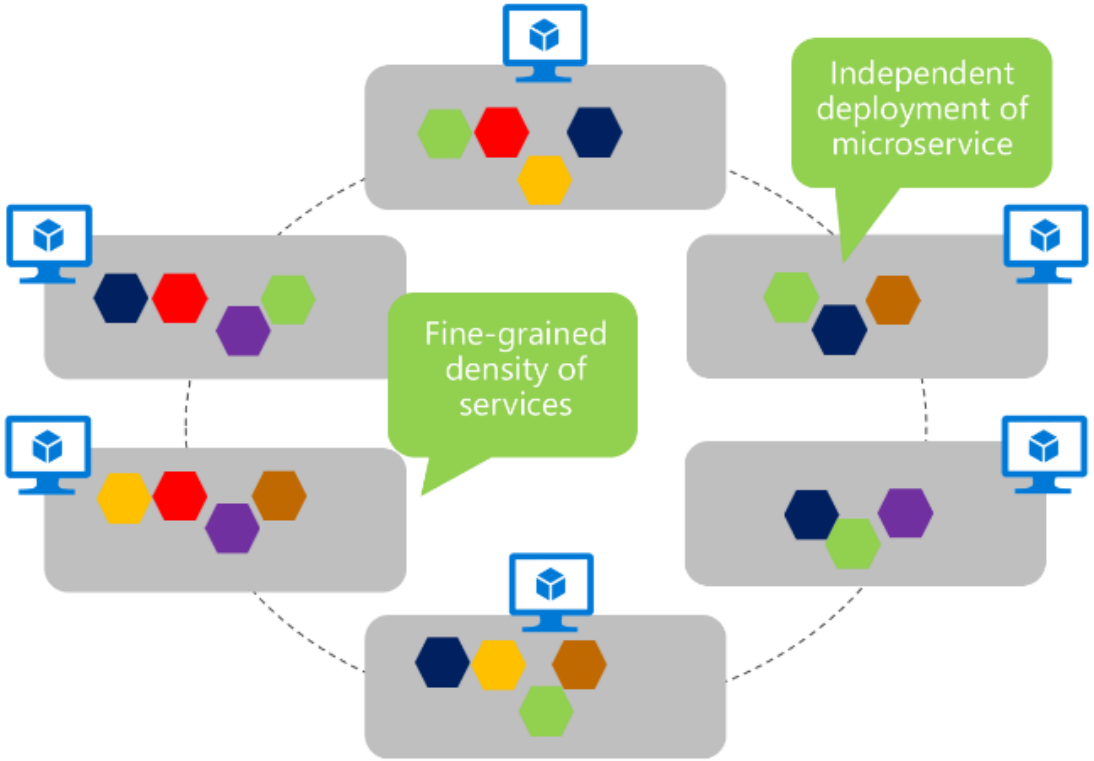



# Microservices application approach

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



# Maintainability
# vs
# Changeability (Replaceability)

# Defining Service Boundaries

General Rule - Service should do "**one thing**“

Design around business capabilities, not horizontal layers such as data access or messaging

Services should support loose coupling and high functional cohesion

- Microservices are loosely coupled if you can change one service without requiring other services to be updated at the same time
- A microservice is cohesive if it has a single, well-defined purpose

A service should encapsulate domain knowledge and abstract that knowledge from clients

# Before finalizing, ensure….

Each service has a **single responsibility**

There are **no chatty calls between services**. If splitting functionality into two services causes them to be overly chatty, it may be a symptom that these functions belong in the same service

Each service is small enough that it can be built by a **small team** working independently

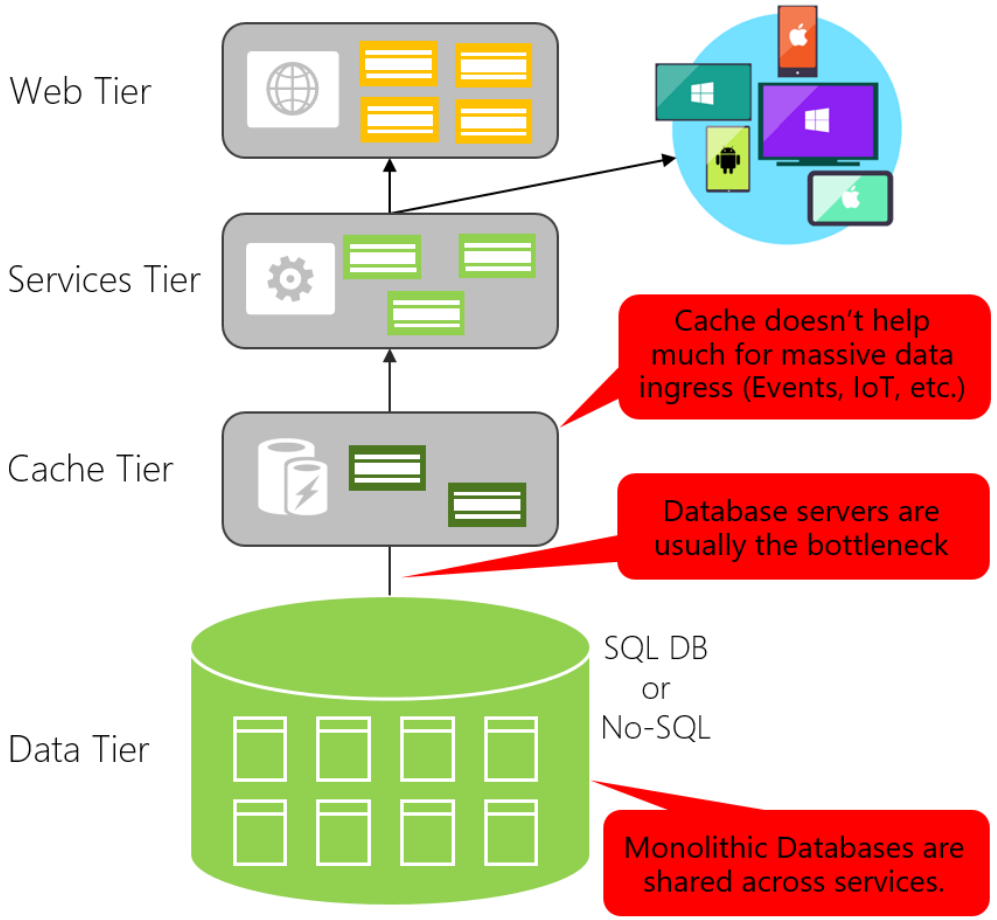
There are **no inter-dependencies**. It should always be possible to deploy a service without redeploying any other services

Services are **not tightly coupled**, and can evolve independently

Your service boundaries will not create problems with **data consistency or integrity**

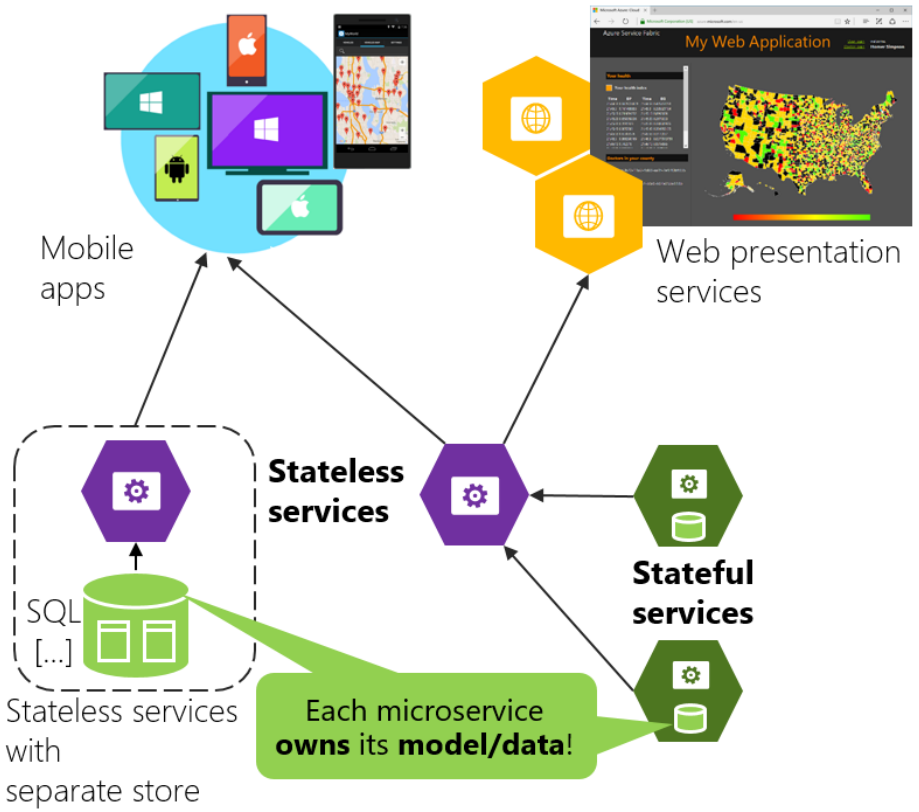
# Data in Traditional approach

- Single monolithic database
- Tiers of specific technologies




# Data in Microservices approach

- Graph of interconnected microservices
- State typically scoped to the microservice
- Remote Storage for cold data



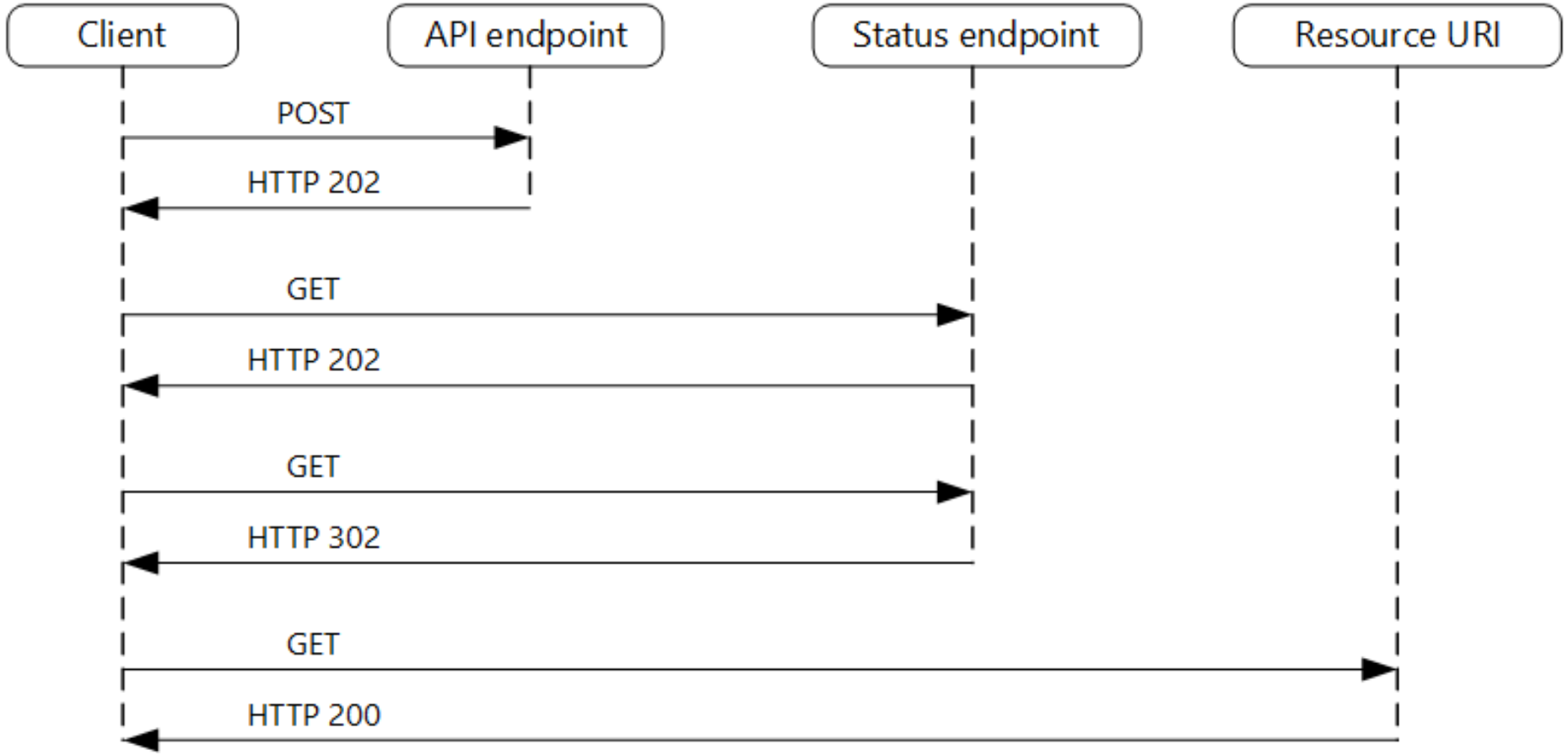
# Inter-service communication

**Synchronous**

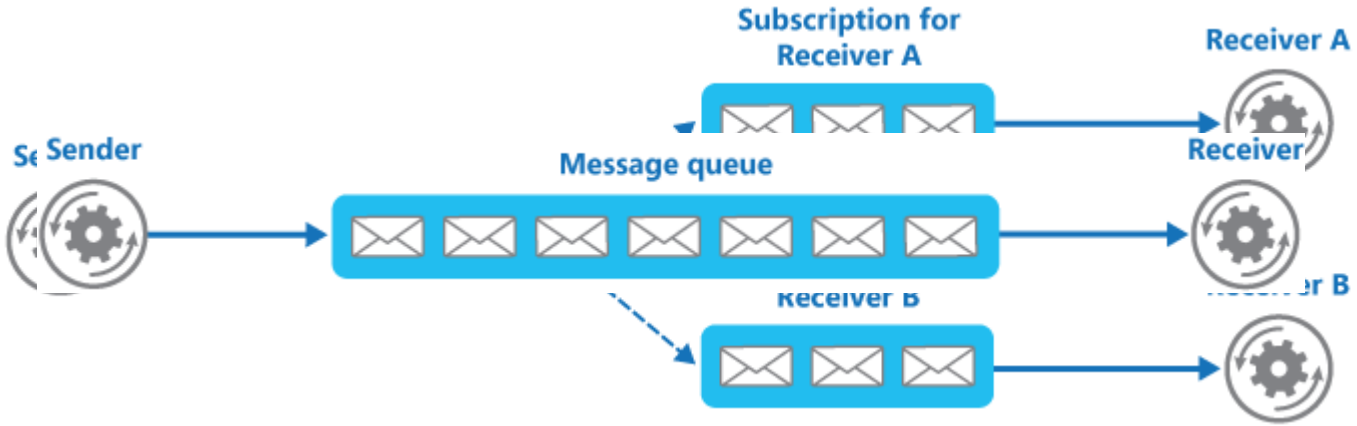
**vs**

**Asynchronous**

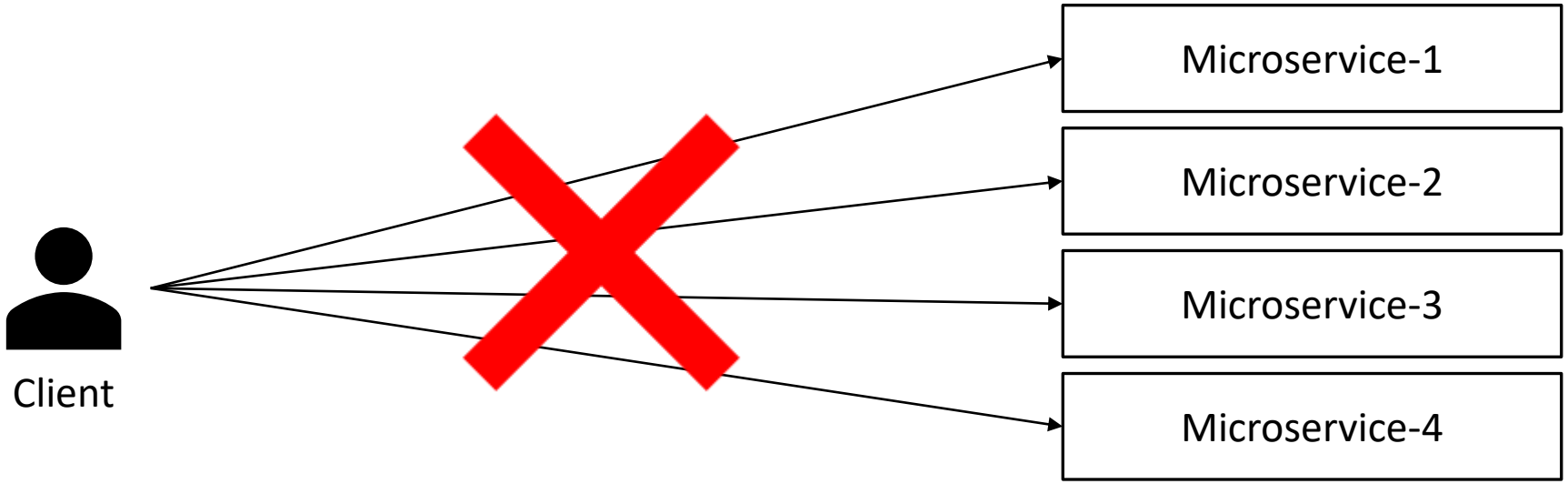
# Asynchronous Request-Reply pattern



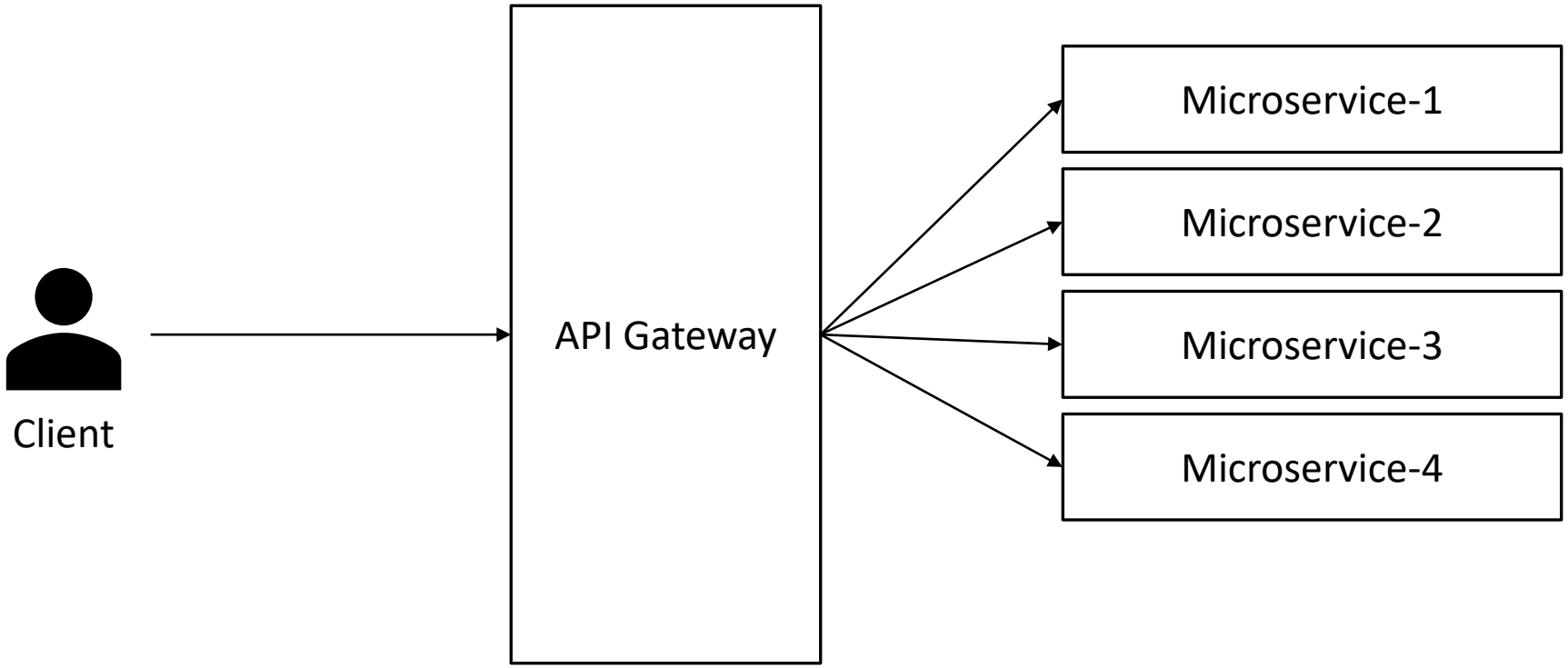
# Asynchronous Messaging Patterns



# Client Requests



# API Gateways



# Logging and Monitoring

Critical for tracking what's happening across services

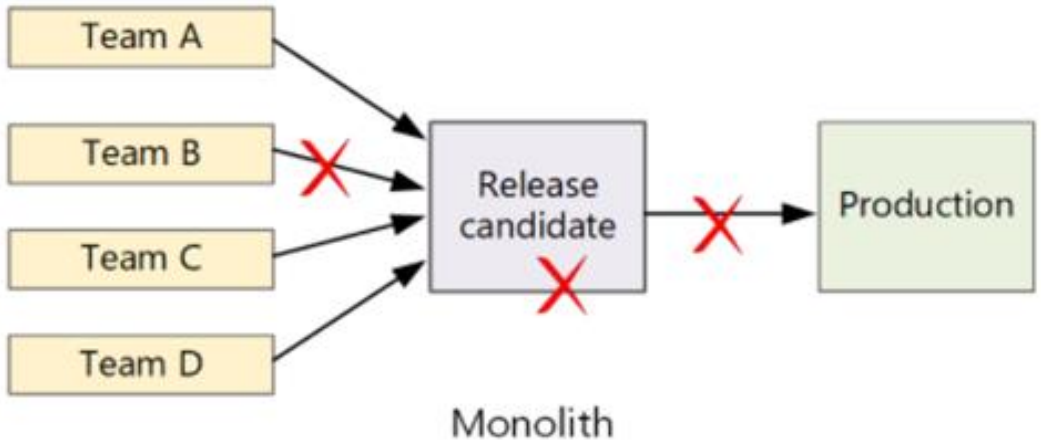
**Distributed Tracing** - understanding the flow of events across services

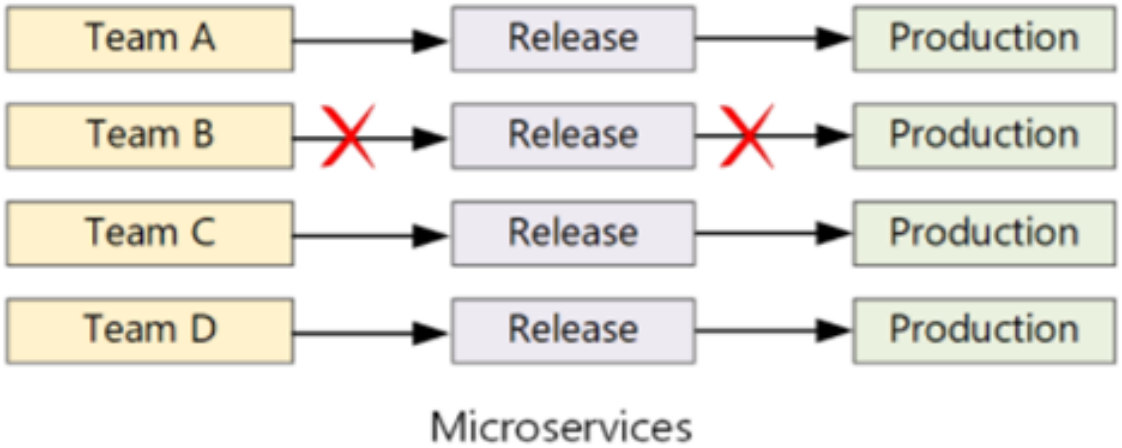
- A single operation or transaction may involve calls to multiple services.
- To reconstruct the entire sequence of steps, each service should propagate a correlation ID that acts as a unique identifier for that operation.

Available options in Azure

- Application Insights - managed service in Azure that ingests and stores telemetry data, and provides tools for analyzing and searching the data

# Release Management



# Benefits of Microservices

**Agility** - easier to manage bug fixes and feature releases

**Small Code, Small Teams**- "two-pizza rule"

**Mix of technologies**

**Resiliency**- If an individual microservice becomes unavailable, it won't disrupt the entire application, as long as any upstream microservices are designed to handle faults correctly

**Scalability**- allows each microservice to be scaled independently of the others

**Data Isolation**- It is much easier to perform schema updates, because only a single microservice is impacted.

# Early Adopters of Microservices

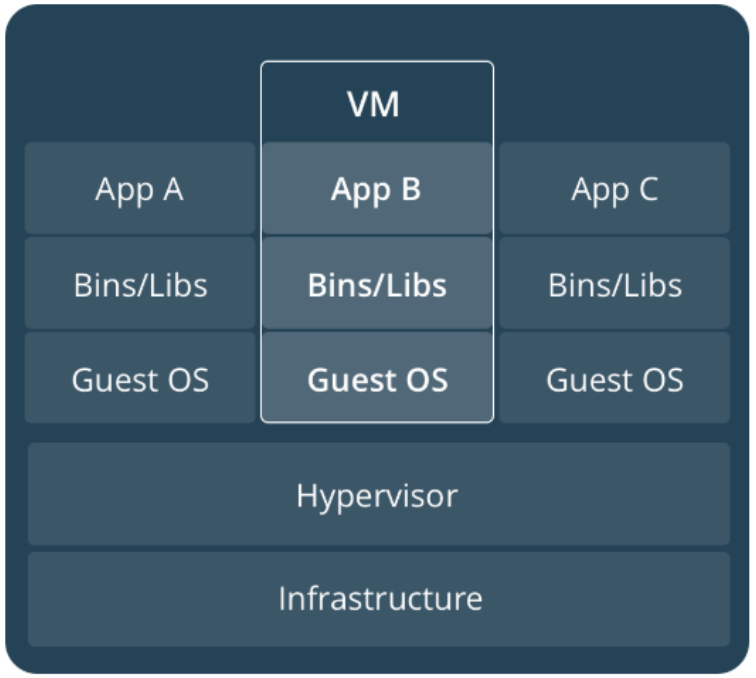




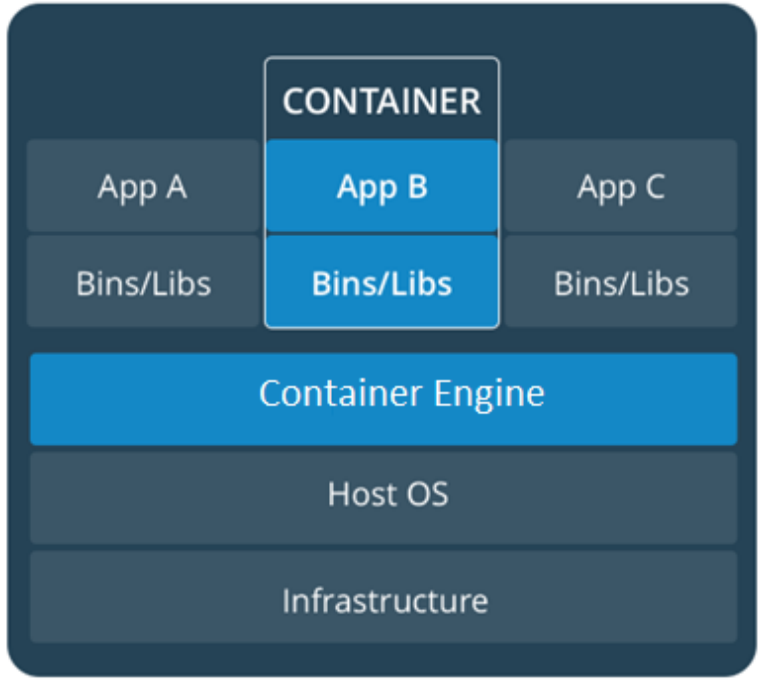


# Virtual Machines vs Containers

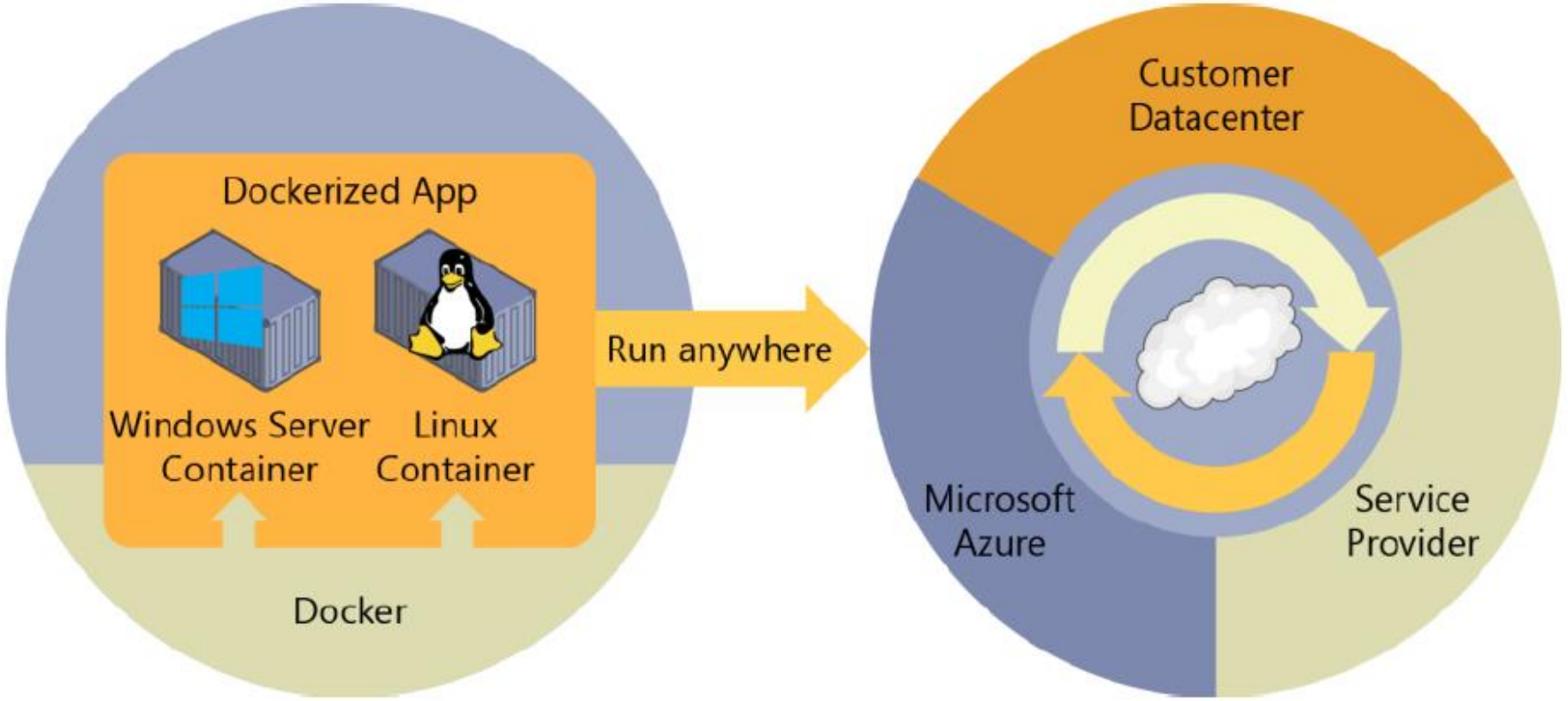
## VIRTUAL MACHINES




## CONTAINERS



# Build Once, Run Anywhere



# Benefits of Containers

Agility

+

Ship apps faster

Portability

+

Easily move workloads

Density

+

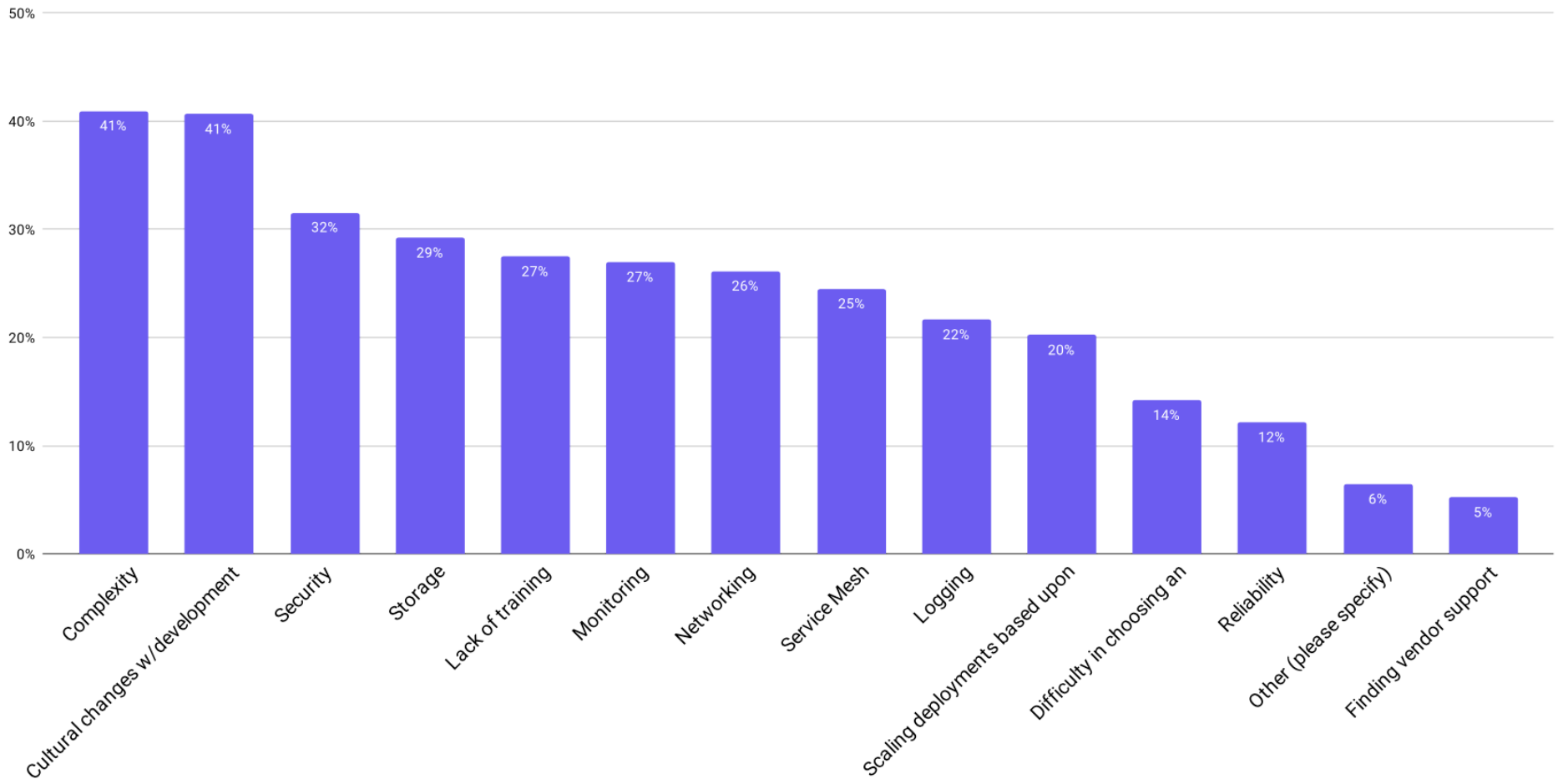
Achieve resource efficiency

Rapid scale

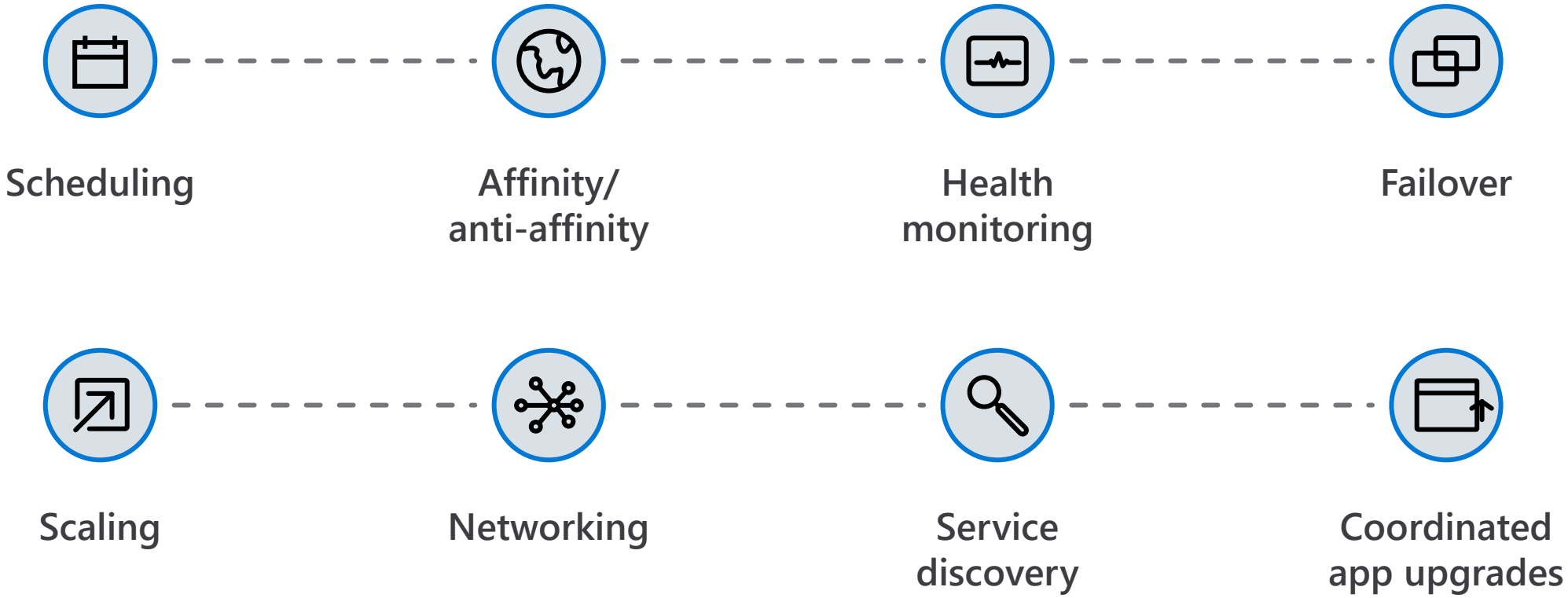
+

Scale easily to meet demand

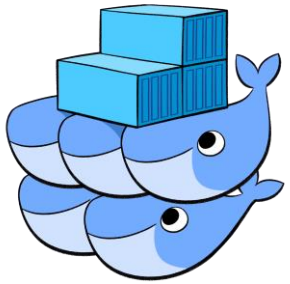
# Challenges with Container Management



# Container Orchestrators



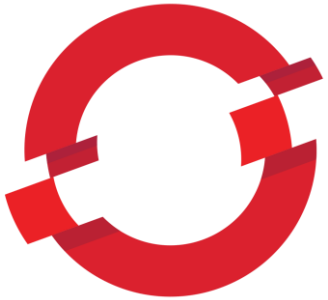
# Container Orchestrators



Docker Swarm

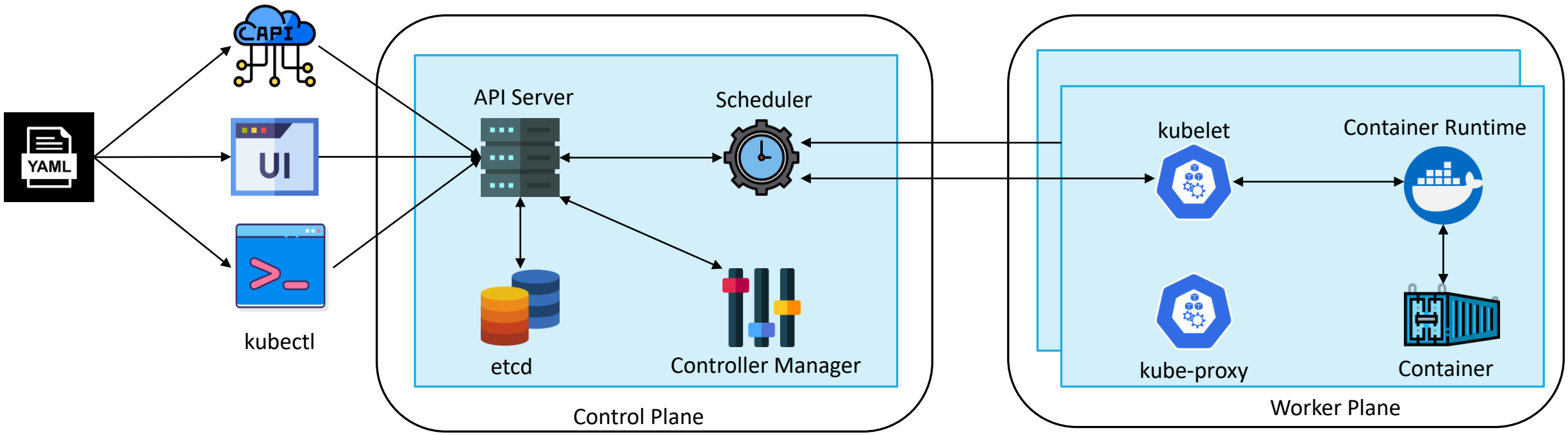
Kubernetes

HashiCorp Nomad

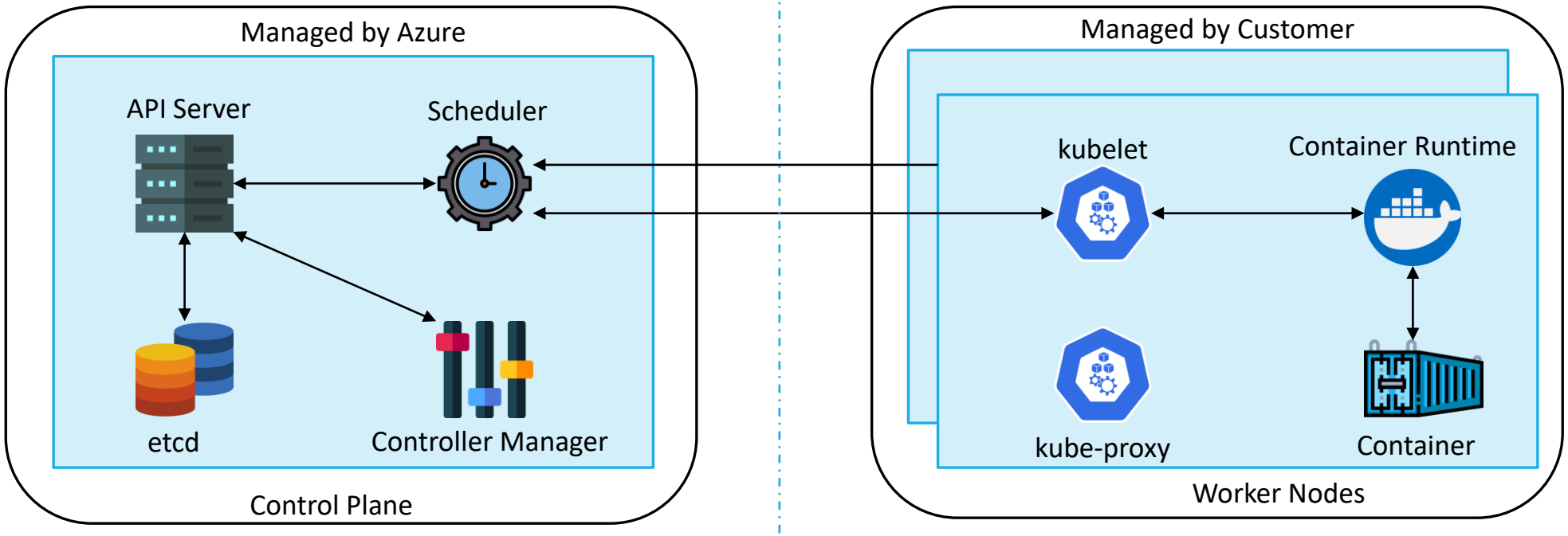


RedHat OpenShift

# Kubernetes Architecture

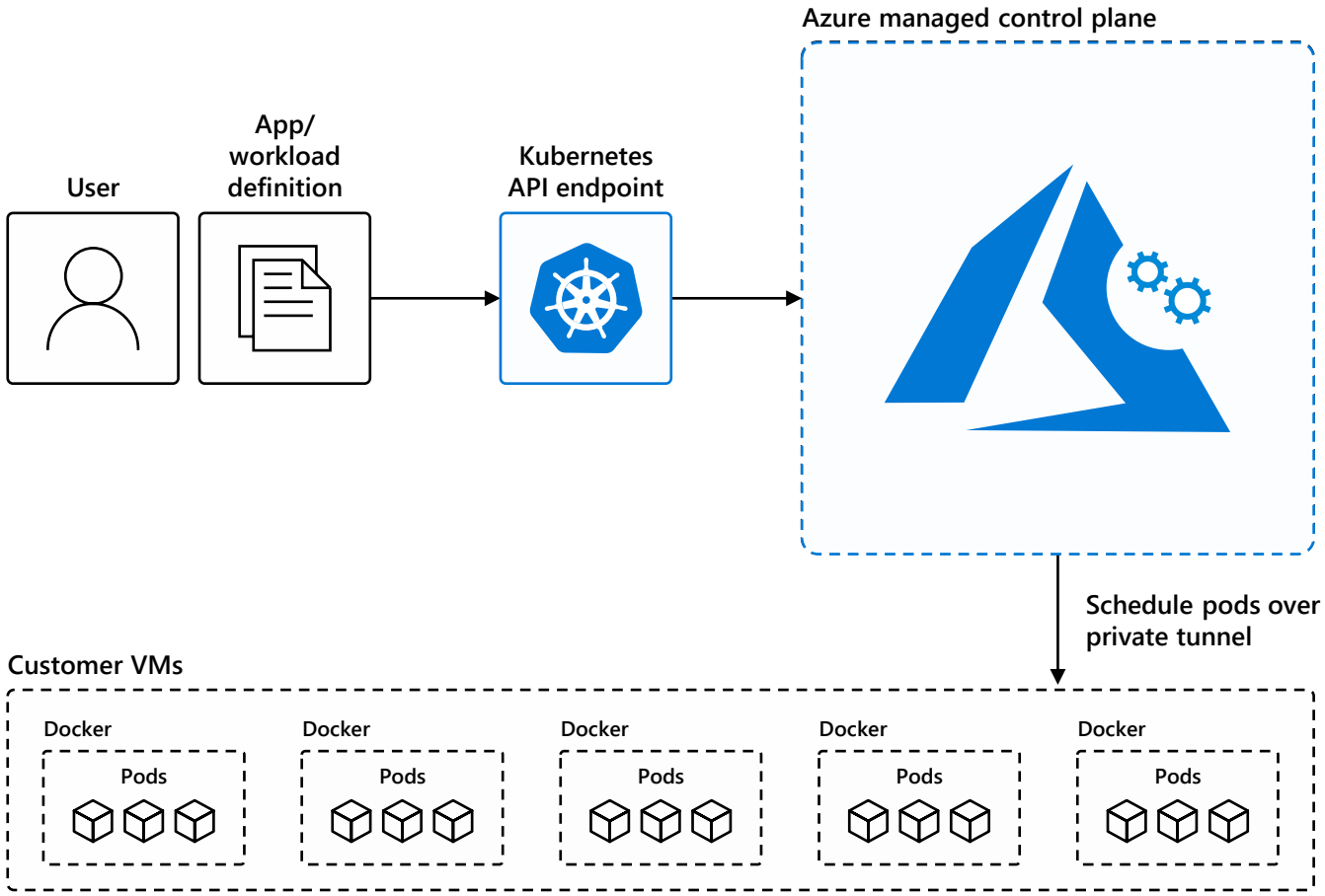


# Azure Kubernetes Service

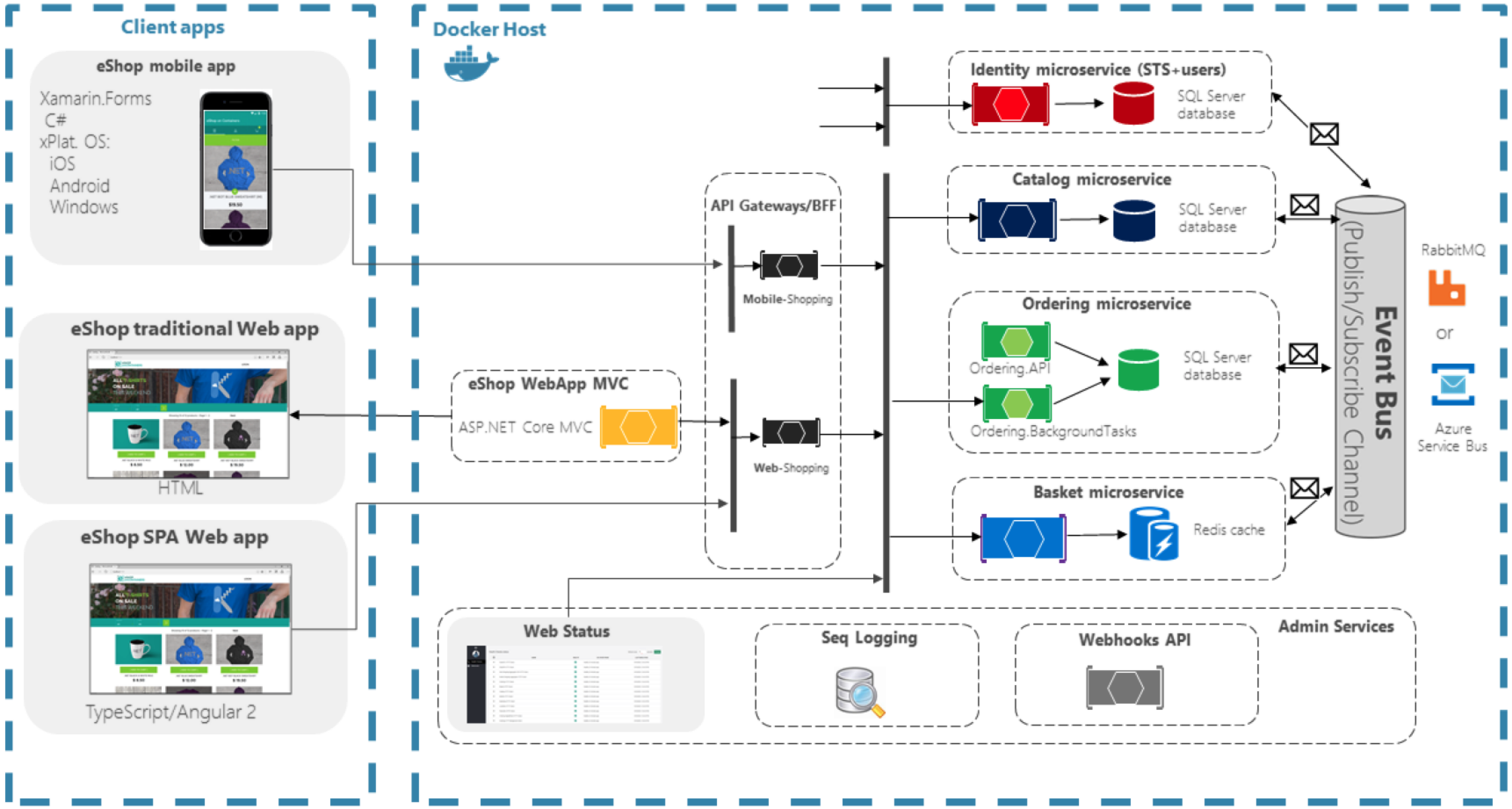


# Benefits of AKS

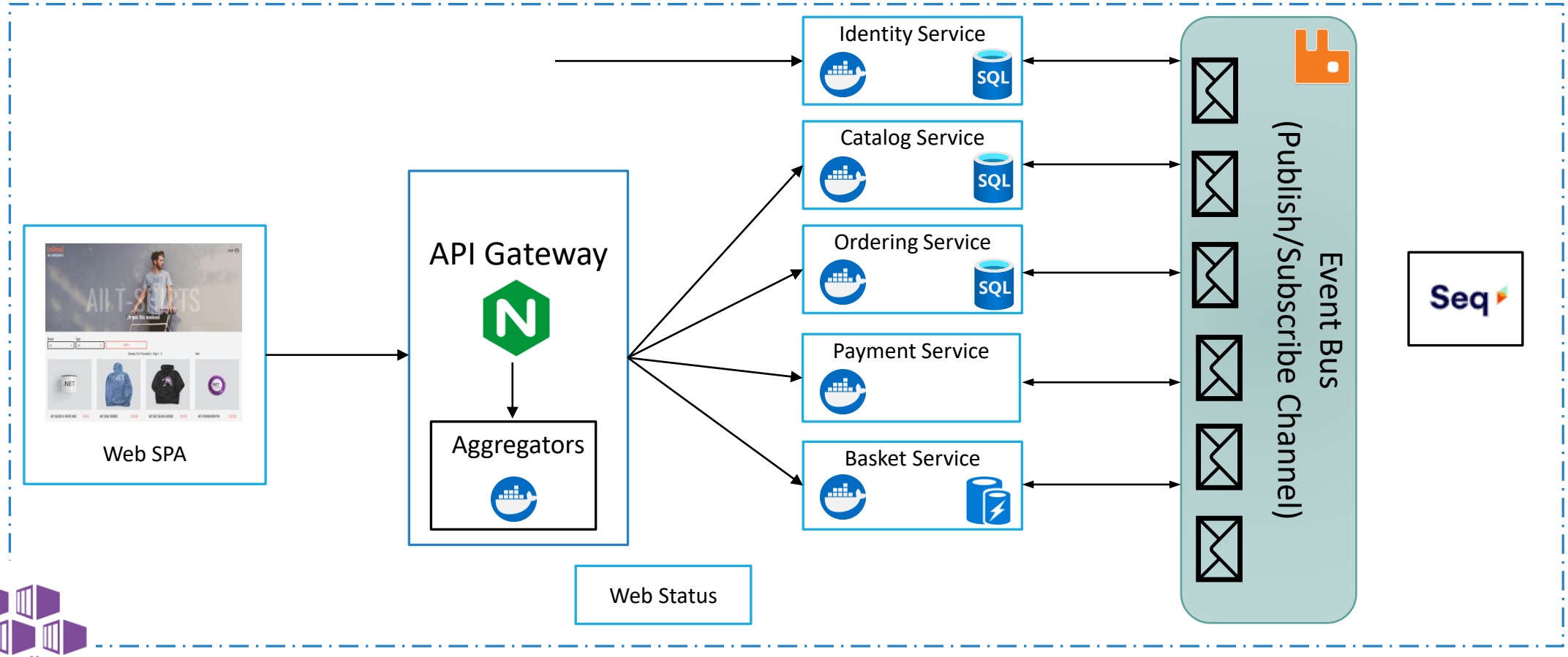
- Automated upgrades, patches
- High reliability, availability
- Easy, secure cluster scaling
- Self-healing
- API server monitoring
- At no charge



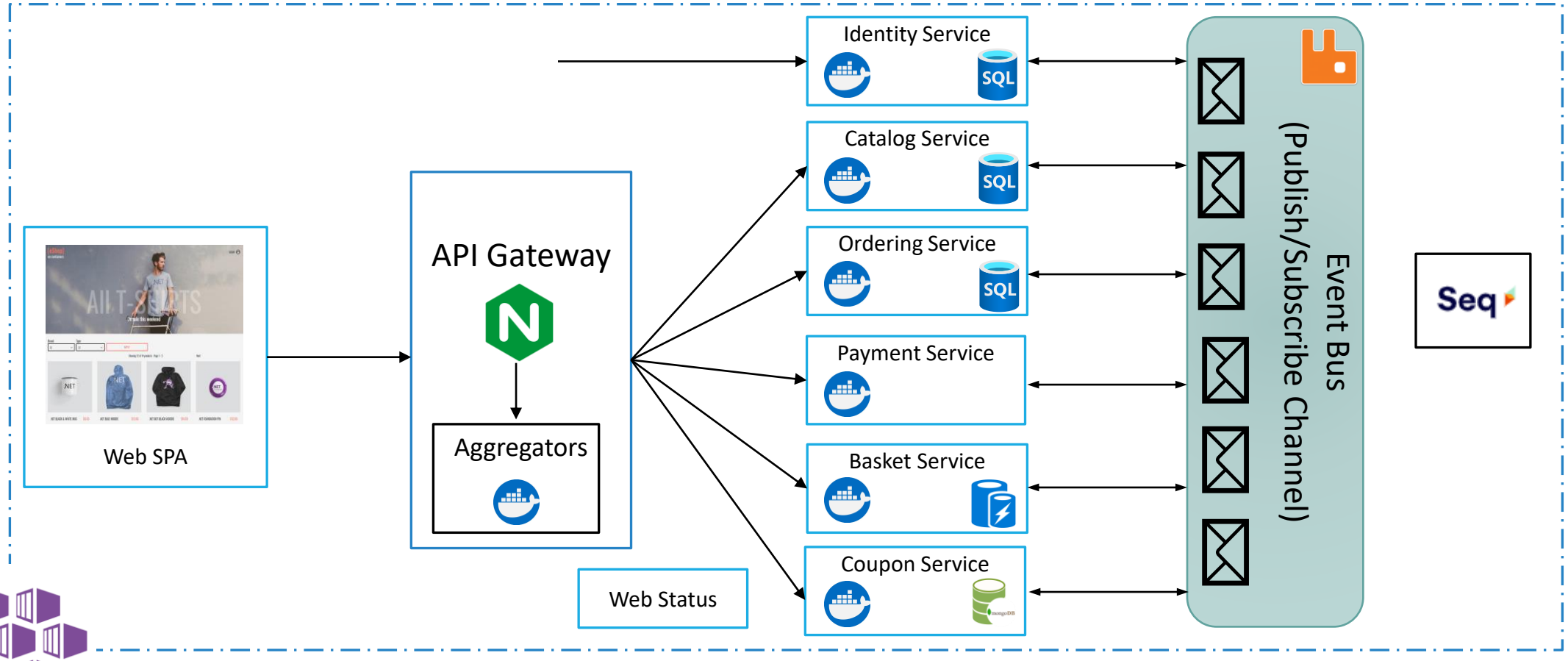
# Application Reference Architecture

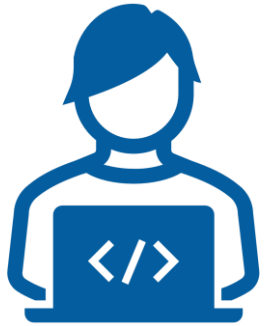


# Reference Architecture - Implementation



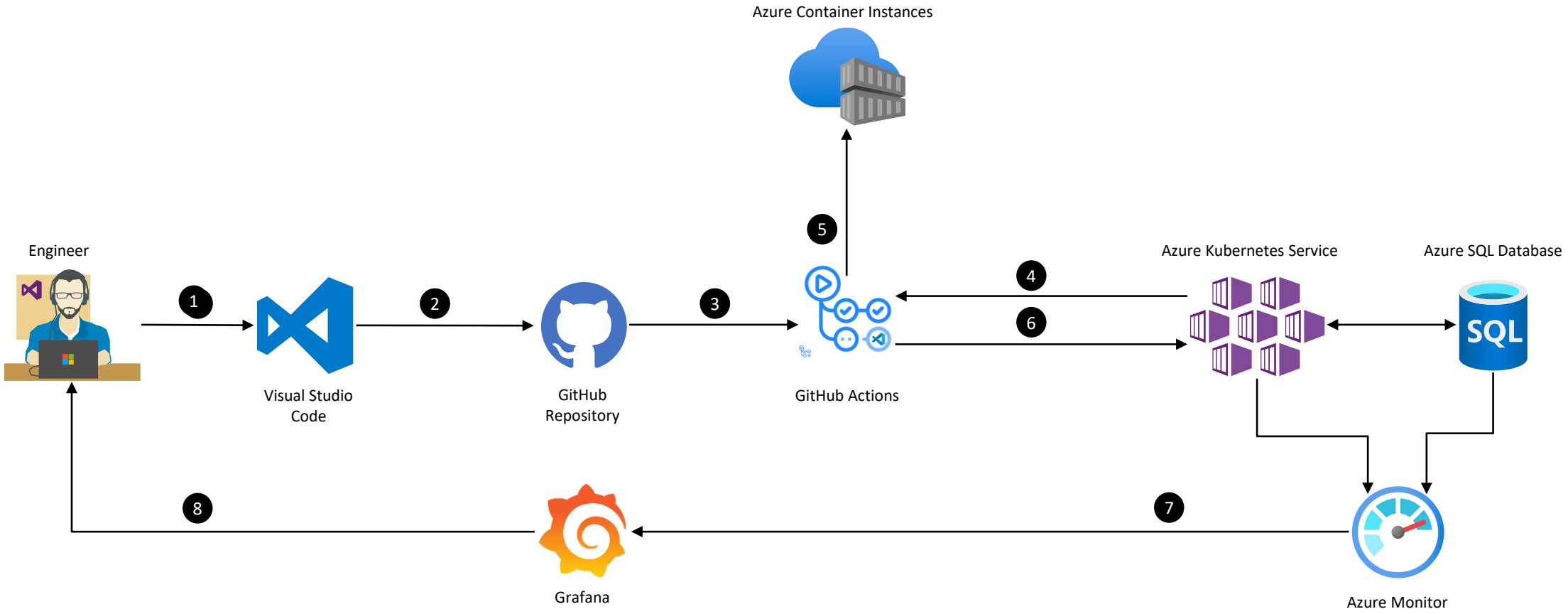
# Reference Architecture - Implementation



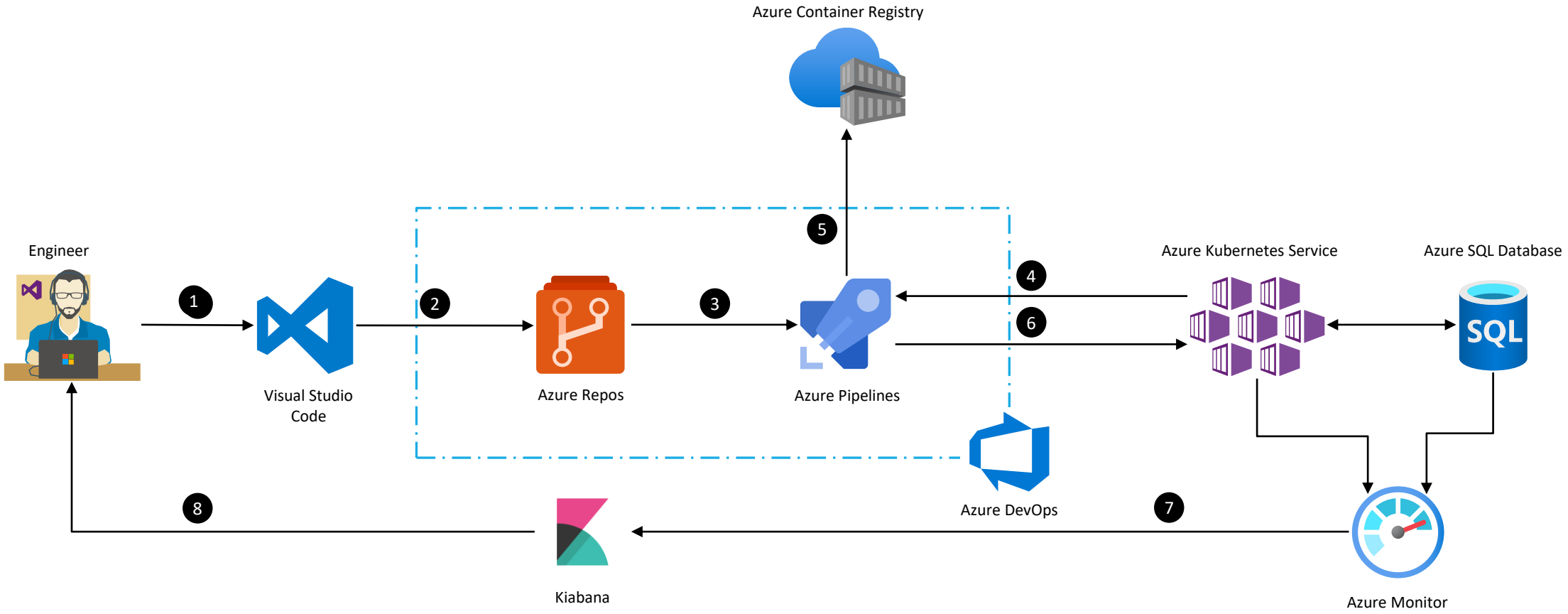



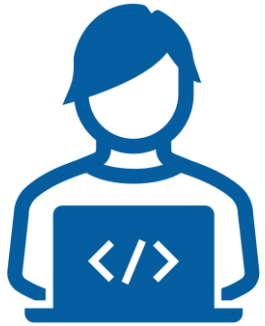
# DEPLOYING TO AZURE KUBERNETES SERVICE

# CI/CD using GitHub Actions



# CI/CD using Azure DevOps



# DEPLOYING THROUGH GITHUB ACTIONS

Seq | Personal | Events | Dashboards | Ingestion | Settings | Support

Applying coupon DISC-5

Text

2020-09-14 16:38:00 to 2020-09-14 16:40:30

14 Sep 2020 16:38:50.568 Applying coupon DISC-5

Event | Level (Information) | Type (0xDAABB1D1) | Retain | Download raw JSON

| | |
|---|---|
| ActionId | fabe90ae-3d54-4655-9b5e-53aa345ed2e2 |
| ActionName | Coupon.API.Controllers.CouponController.GetCouponByCodeAsync (Coupon.API) |
| ApplicationContext | Coupon.API |
| ConnectionId | 0HM2OUE115VHO |
| CouponCode | DISC-5 |
| ParentId | 0000000000000000 |
| RequestId | 0HM2OUE115VHO:00000002 |
| RequestPath | /api/v1/coupon/DISC-5 |
| SourceContext | Coupon.API.Controllers.CouponController |
| SpanId | 92638d66d6825741 |
| TraceId | 8cb35b57ddcbe64dac131cfd14862387 |

*124 events scanned to today at 4:38 PM… complete.*

Find signals and queries

SIGNALS

- *None*
- @Level
  - Errors
  - Warnings
- Exceptions

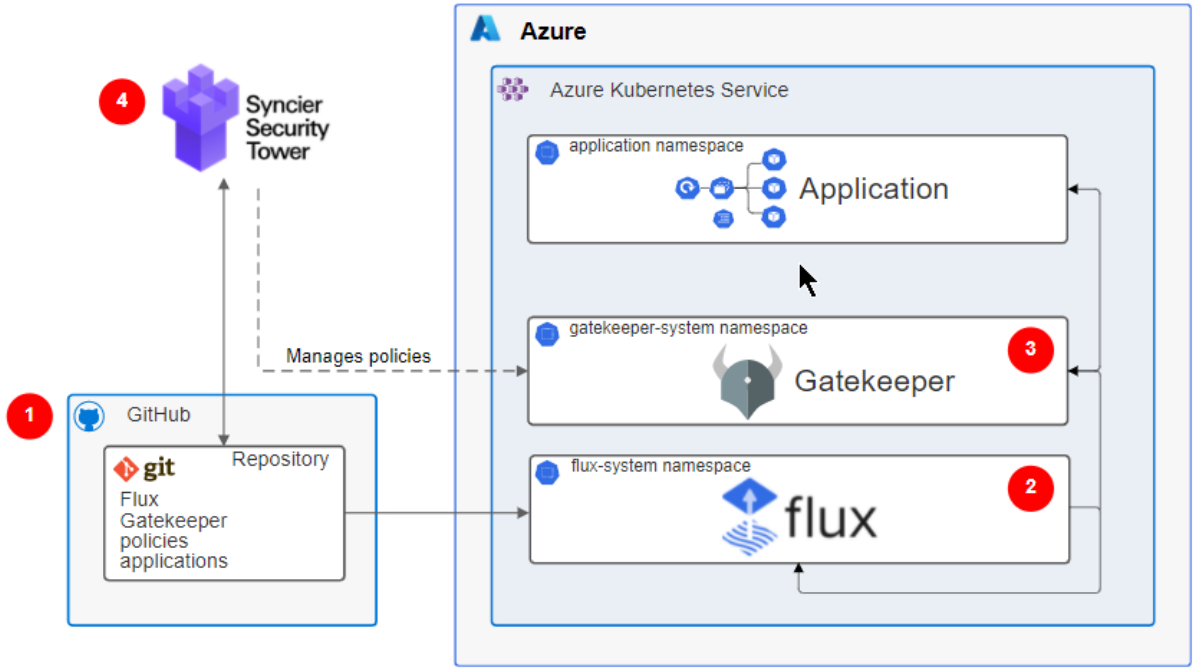
QUERIES

- Available Properties
- Count by Hour
- Latest as Table
- Space by Event Type

2020.2 Single-User License

# What is GitOps?

GitOps is an operational framework that takes DevOps best practices used for application development such as version control, collaboration, compliance, and CI/CD, and applies them to infrastructure automation.



# Further Learning

Microservices Guide - https://www.martinfowler.com/microservices/

Kubernetes Documentation - https://kubernetes.io/docs/home/

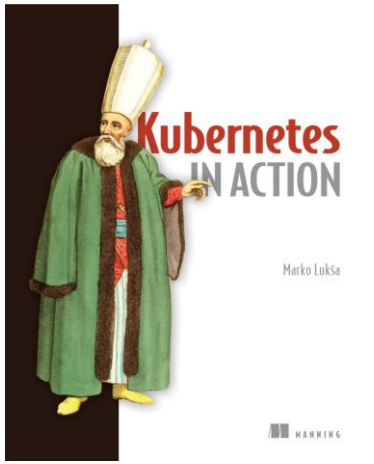
Kubernetes on Azure - https://bit.ly/2Y8CMzM

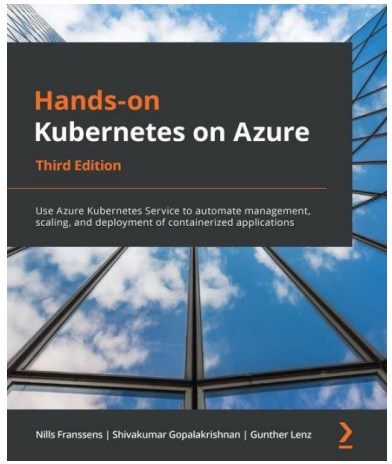
Microservices on AKS reference architecture - https://bit.ly/3pYT75N

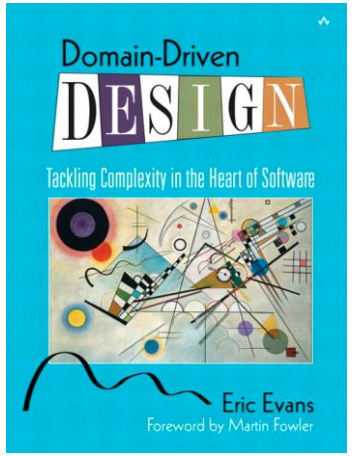
eShopOnContainers repo - https://bit.ly/3GM2VFZ

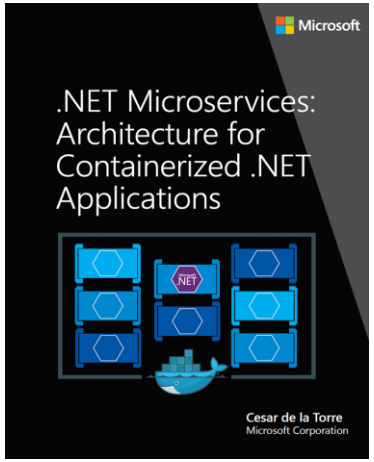
Demo 1 - https://bit.ly/3wadu0Q

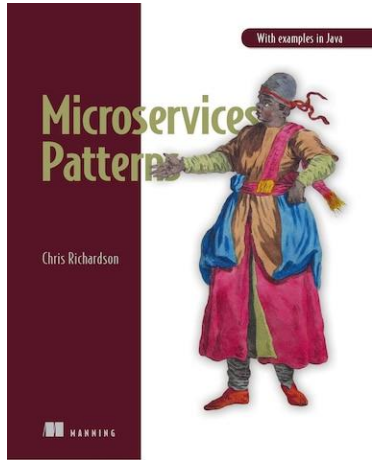
Demo 2 - https://bit.ly/3mEWFYF



# Contact Information

https://vaibhavgujral.com

@vaibhavgujral_

https://www.linkedin.com/in/vaibhavgujral/

https://www.youtube.com/c/VaibhavGujral

vaibhav@vaibhavgujral.com

LinkedIn

Twitter

Email

# Links

Session Feedback

Slides


Thank
You!