



TLV Cloud Workshops

Boost Yourself Up To The Cloud



Microservices – Development to Production with Azure

Avishay Balter

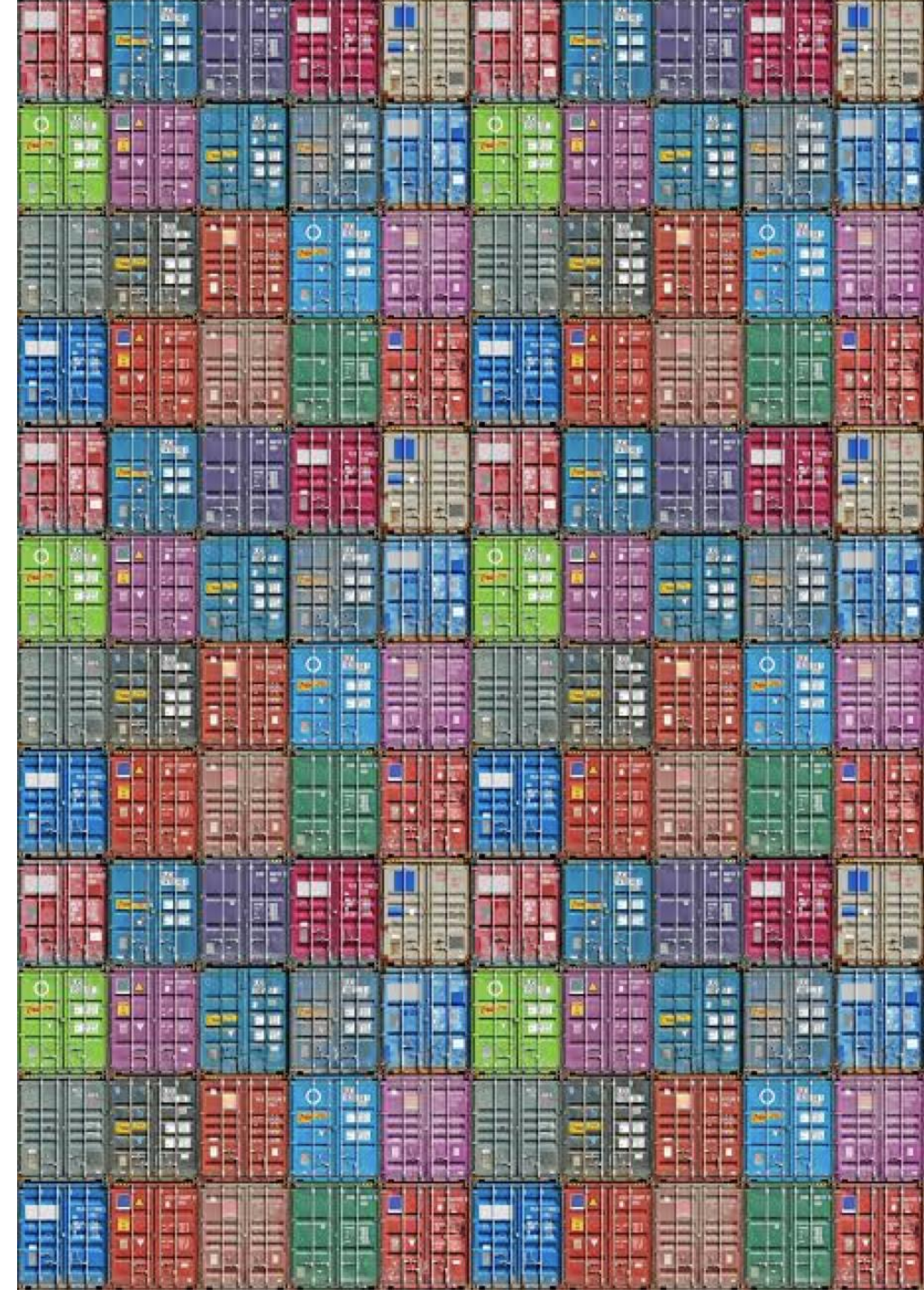
Azure Technical Solution Professional | Microsoft Israel

Isam Abu Fool

Azure Technical Solution Professional | Microsoft Israel

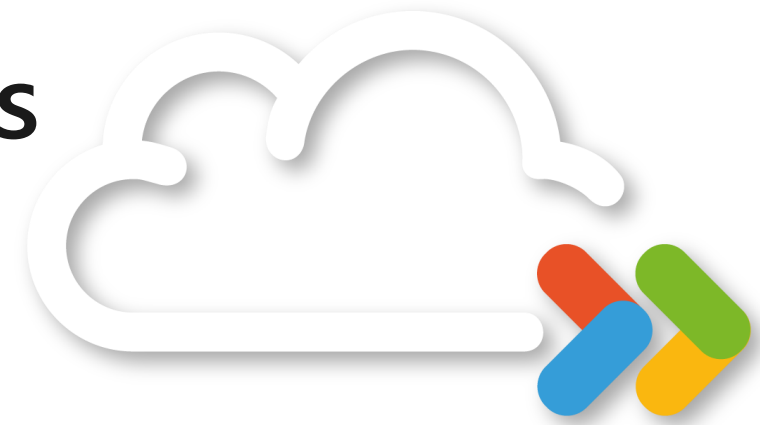
Agenda

- Introduction to microservices
- Traditional application vs microservices
- Container and Microservice Orchestration
- **DEMO:** Developing Microservices
- **DEMO:** DevOps-ing Microservices
- **DEMO:** Production and Scale of Microservices



Where did it come from?

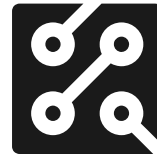
The cloud had changed expectations



Agility



Availability



Density



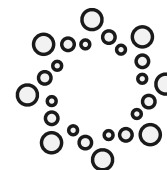
Hyper-scale



Immutability



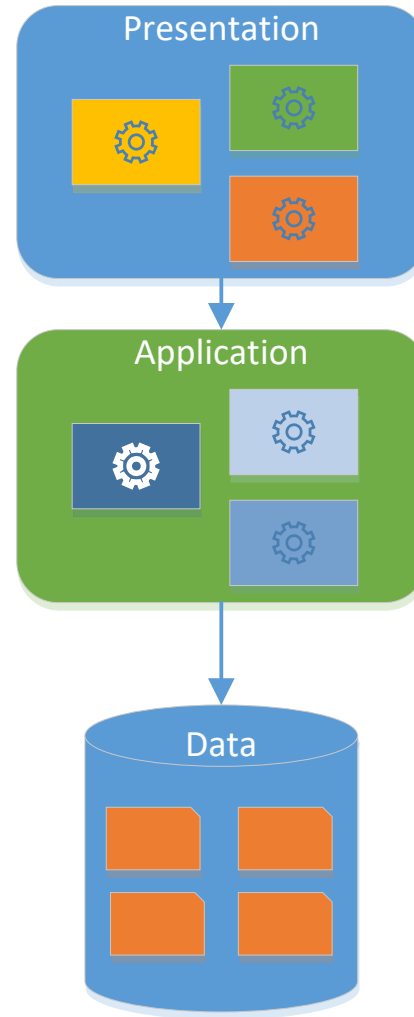
Elasticity



Portability

Most common problems in Apps today

Traditional 3 Tier Application



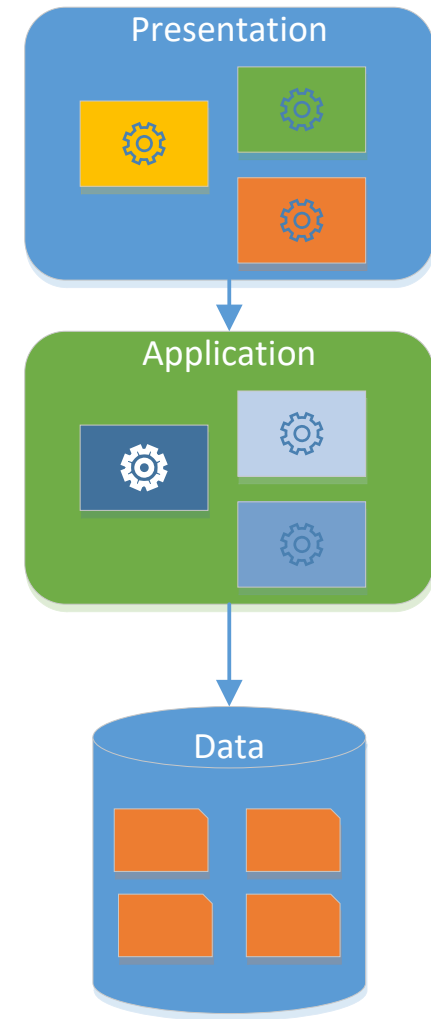
Most common problems in Apps today

- Code Complexity
- Hard to maintain/upgrade
- Reliability
- Hard to scale
- Difficult to use new/multiple development frameworks

Most common requirements today

- Continually evolving applications
- Faster delivery of features and capabilities
- Scalability
- Availability

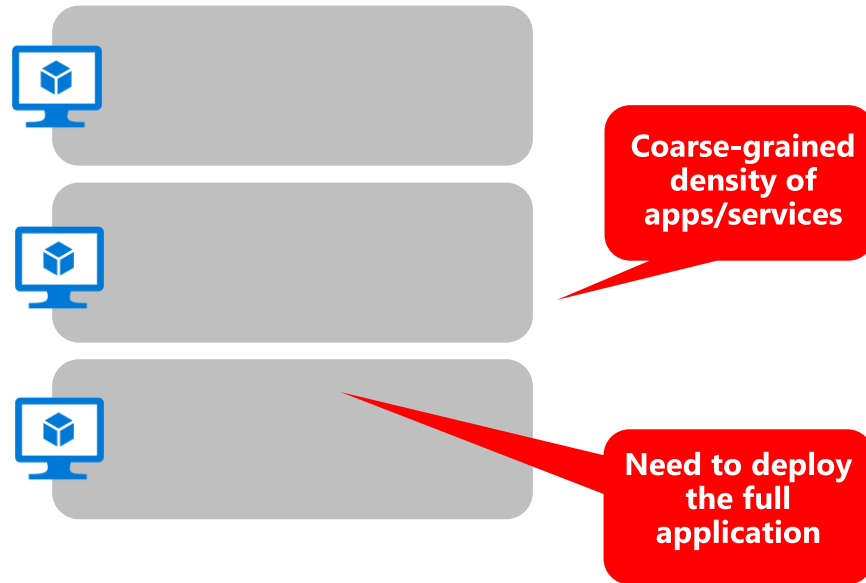
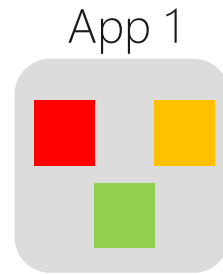
Traditional 3 Tier Application



What are Microservices?

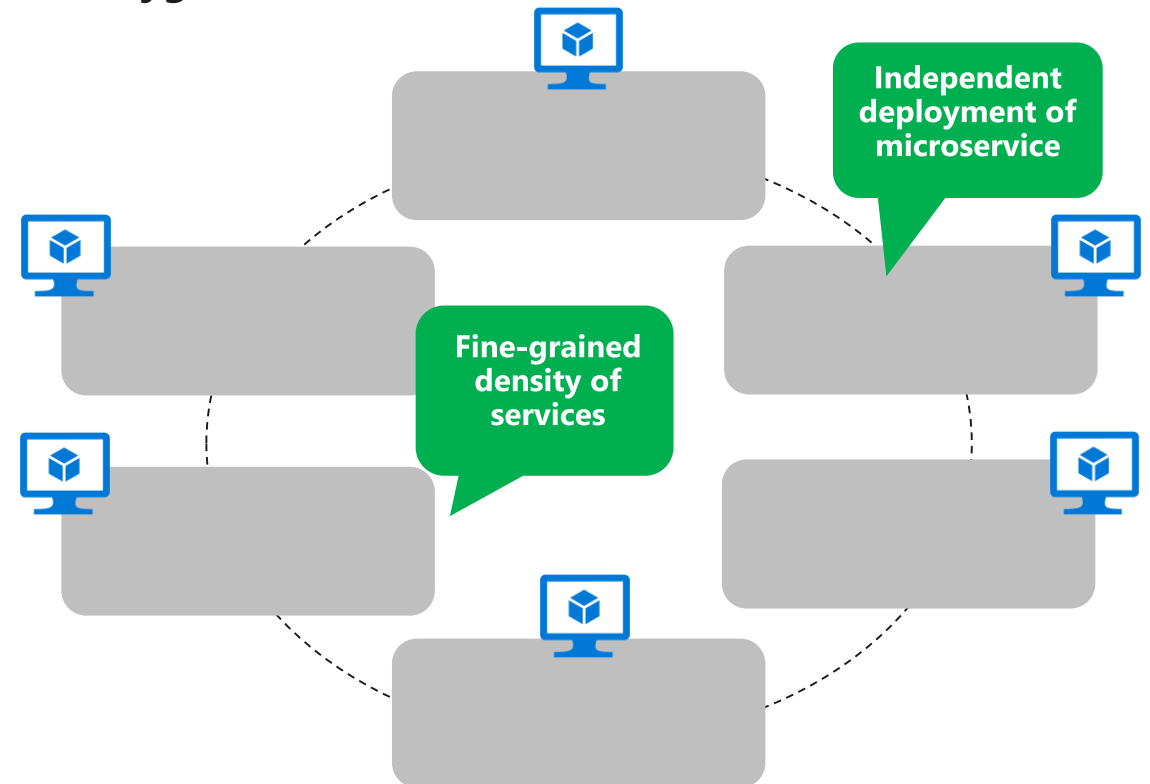
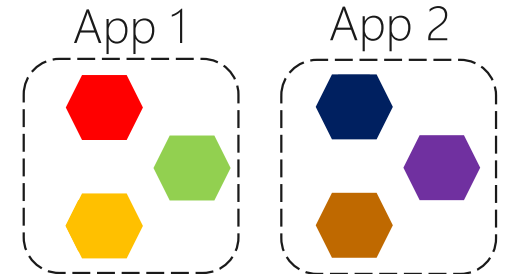
Traditional application approach

- Componentized with layers
- Scales by cloning servers/VMs
- Updated together



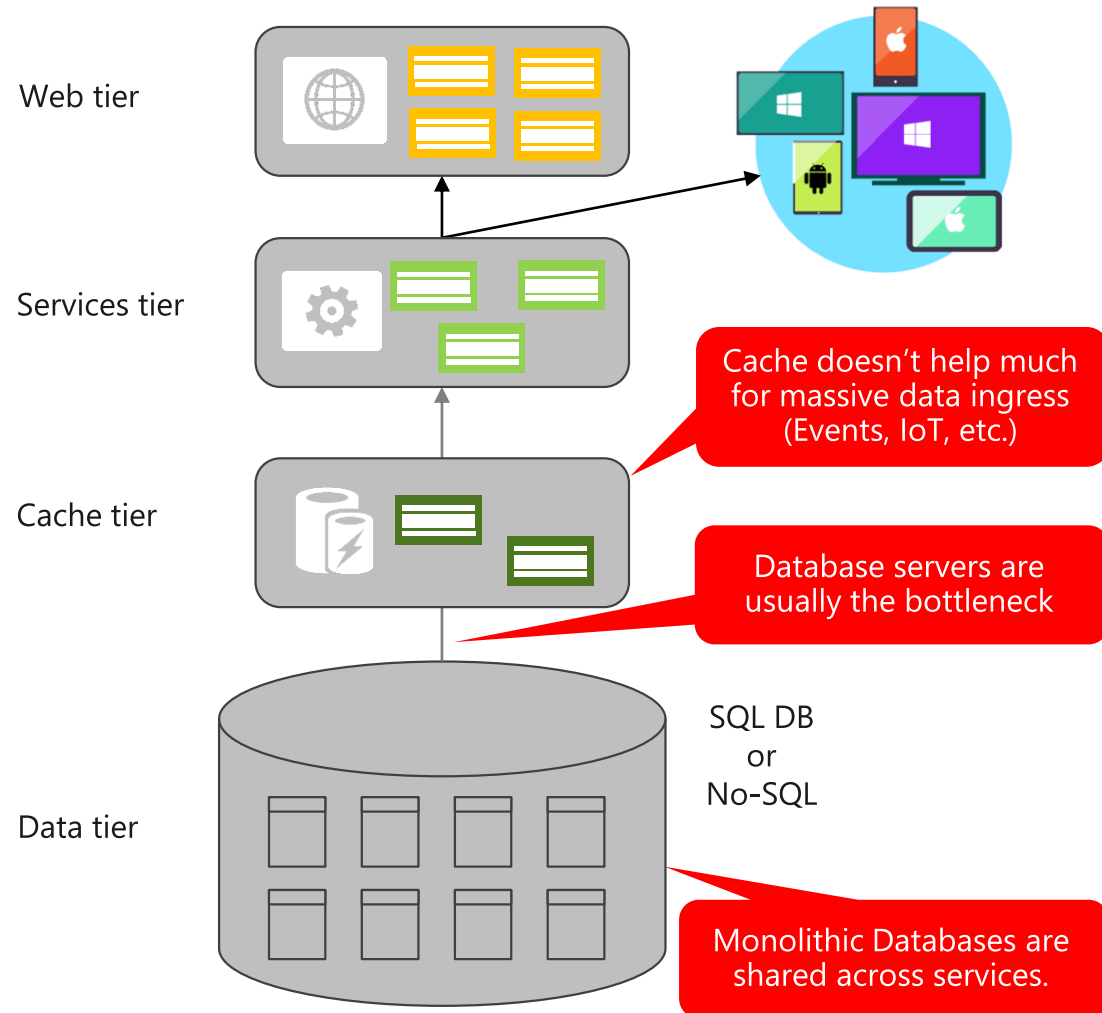
Microservices application approach

- Smaller services of functionality
- **Developed, deployed and updated independently**
- Scales out by **deploying each service independently**
- **Polyglot**



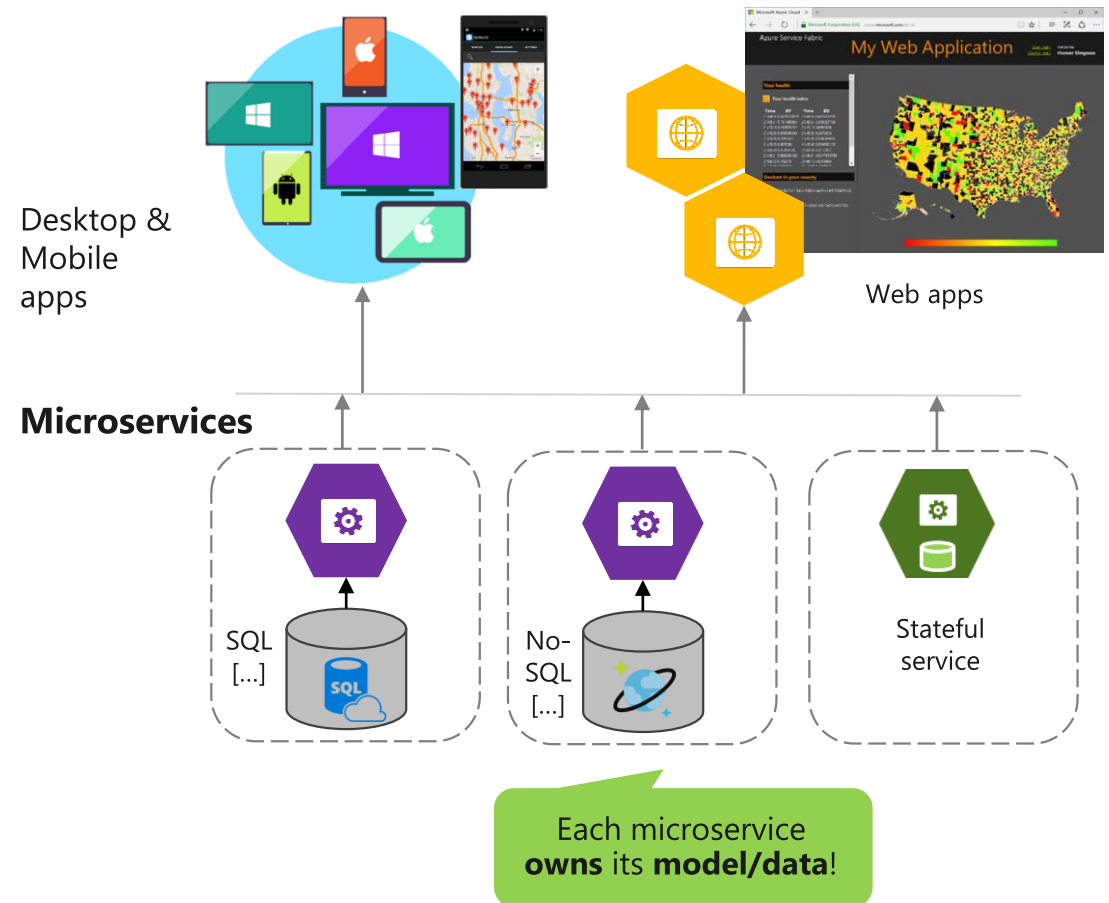
Traditional application approach

- Single monolithic database
- Tiers of specific technologies



Data in Microservices approach

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



12 Factor App – SOLID principles for cloud native

Codebase – One codebase tracked in revision

Dependencies – Explicitly declare and isolate dependencies

Configuration – Store config in the environment

Backing Service – Treat backing services as attached resource

Build, Release and Run – Strictly separate build and run stages

Process – Execute the app as one or more stateless processes

Port Binding – Export services via port binding

Concurrency – Scale out via the process model

Disposability – Maximize robustness with fast startup and graceful shutdown

Dev/prod Parity – Keep development, staging, and production as similar as possible

Logs – Treat logs as event streams

Admin Process – Run admin/management tasks as one-off processes

Cloud Native

<https://12factor.net/>

By Adam Wiggins (Heroku)

Microservices != Containers

But they are a great fit... 😊

Containers are NOT microservices

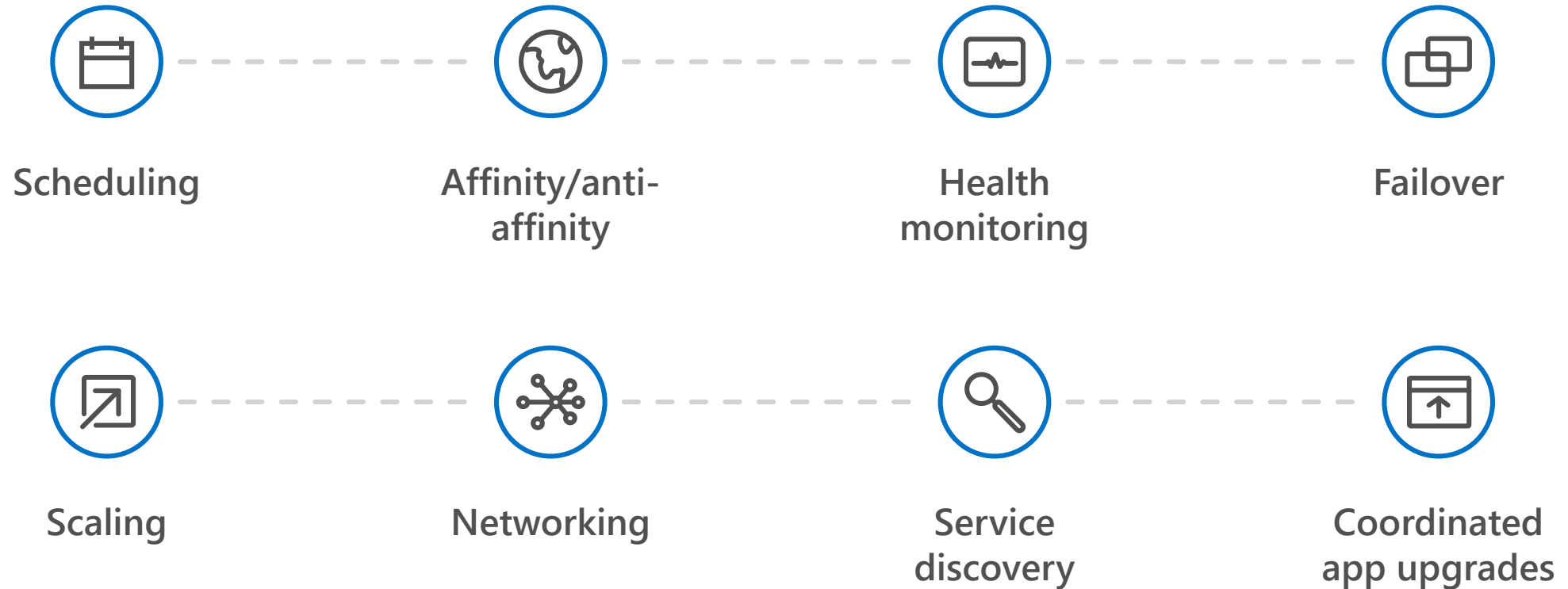
Well, you can still put a large monolithic application inside a container....

Microservices are an application design pattern:

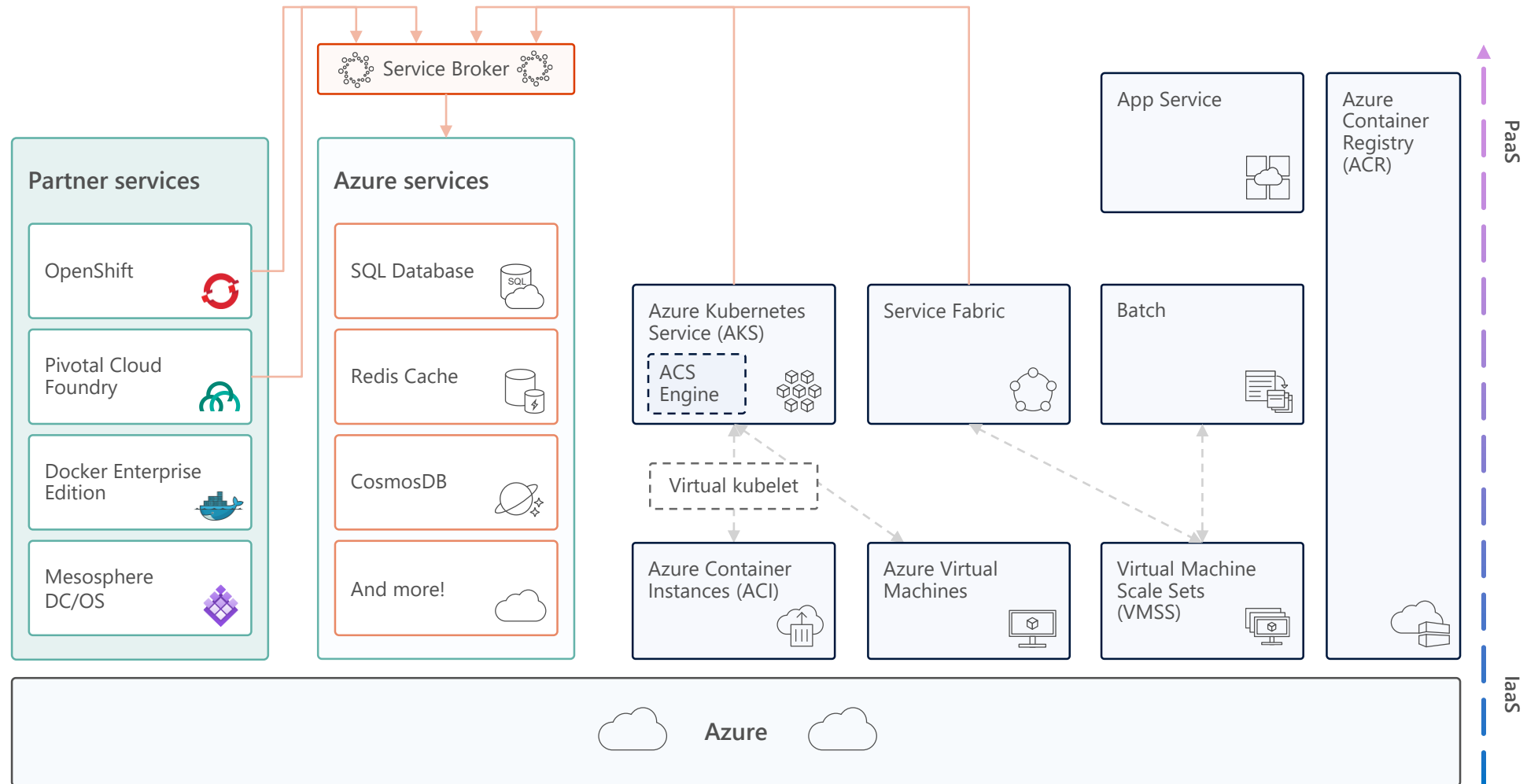
- Small units of responsibility
- Structured interfaces and communication
- Potentially different technology choices
- Generally horizontally scalable

Containers are OS Isolation\Encapsulation

The elements of **orchestration**



Azure container ecosystem



Kubernetes: the industry leading orchestrator



Portable

Public, private, hybrid,
multi-cloud

Extensible

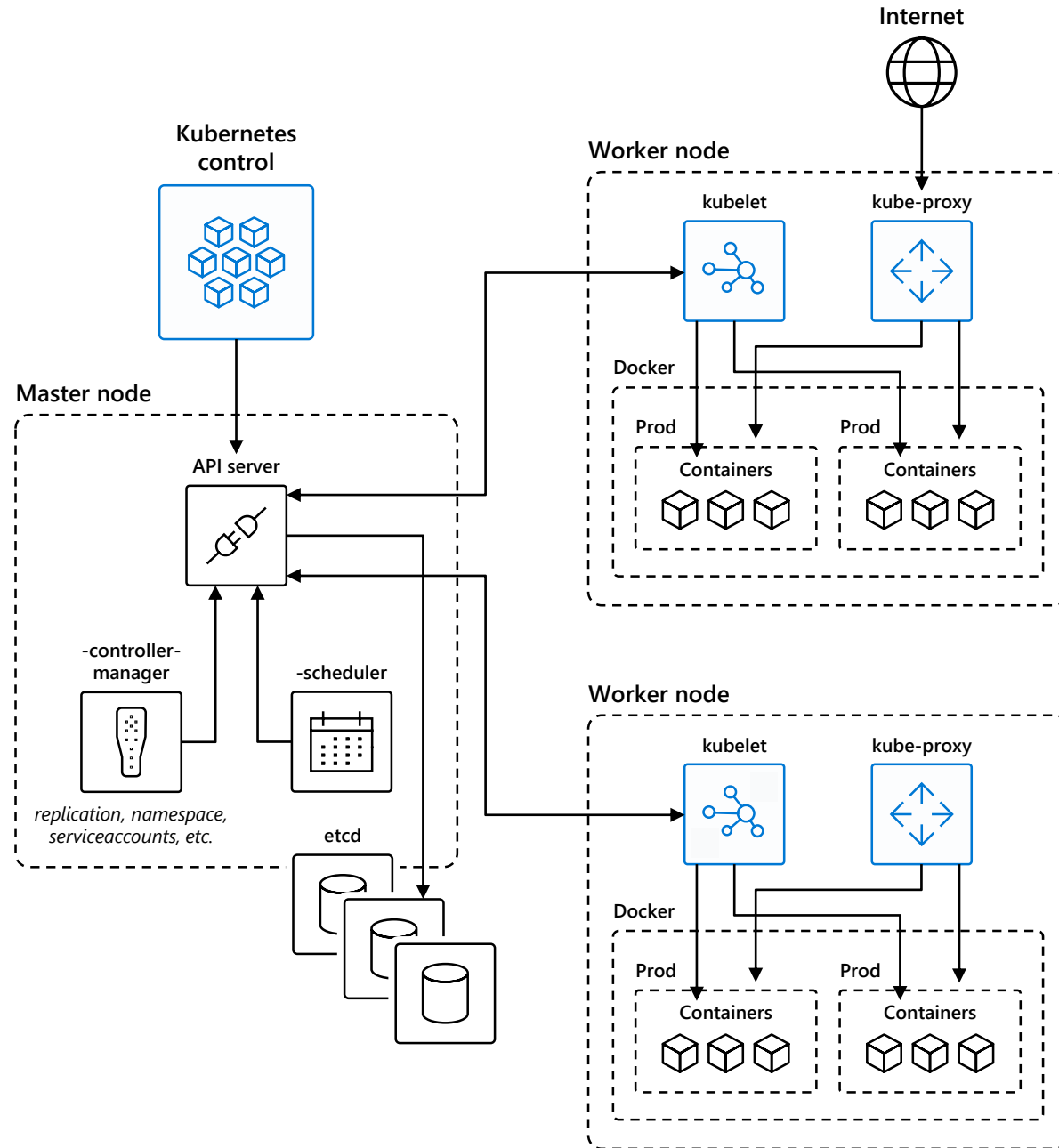
Modular, pluggable,
hookable, composable

Self-healing

Auto-placement, auto-restart,
auto-replication, auto-scaling

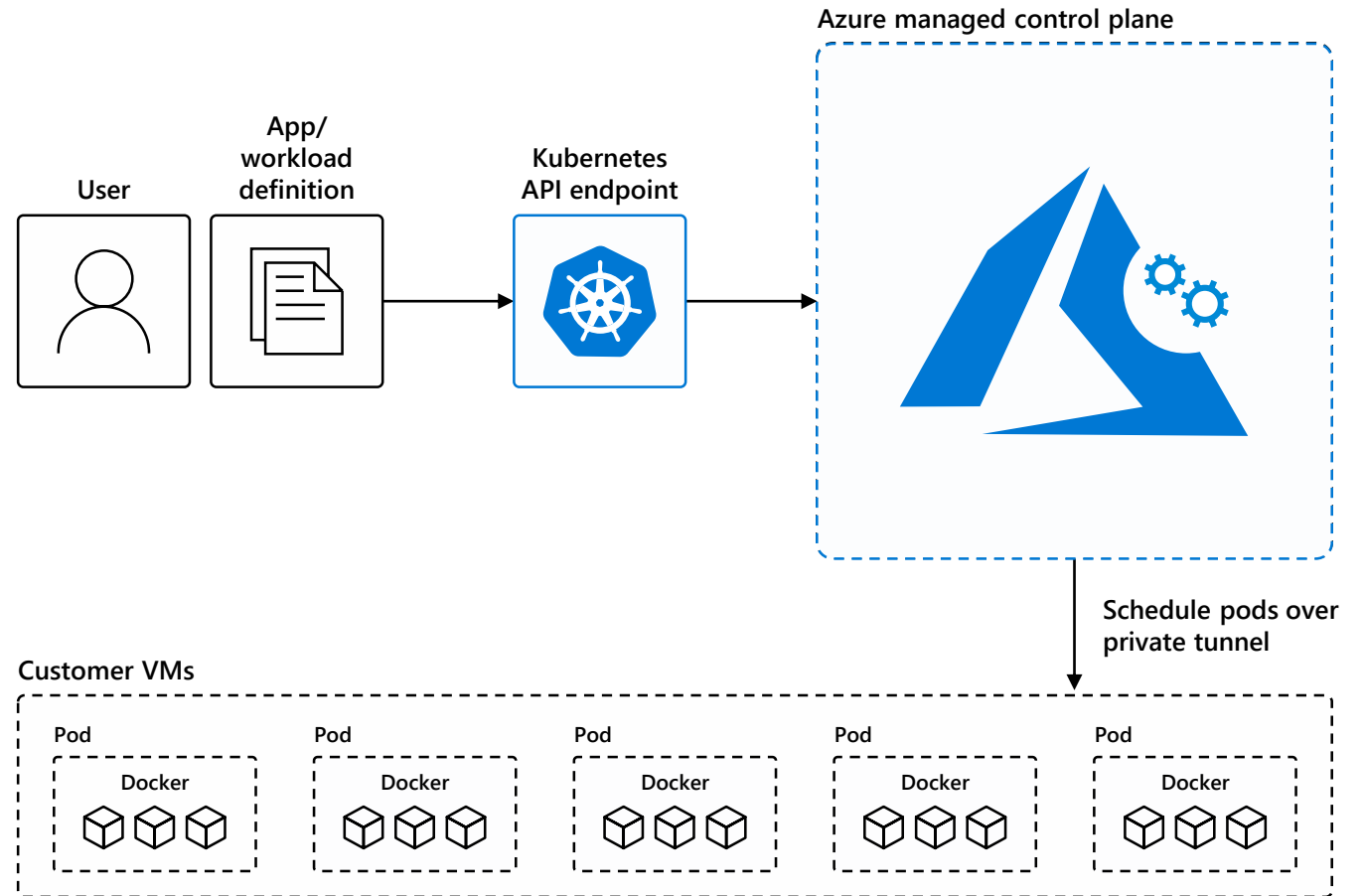
Kubernetes 101

1. Kubernetes users communicate with API server and apply desired state
2. Master nodes actively enforce desired state on worker nodes
3. Worker nodes support communication between containers
4. Worker nodes support communication from the Internet






How managed Kubernetes on Azure works

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



Azure makes Kubernetes easy

Deploy and manage Kubernetes with ease

 Task	 The old way	 With Azure
Create a cluster	Provision network and VMs Install dozens of system components including etcd Create and install certificates Register agent nodes with control plane	az aks create
Upgrade a cluster	Upgrade your master nodes Cordon/drain and upgrade worker nodes individually	az aks upgrade
Scale a cluster	Provision new VMs Install system components Register nodes with API server	az aks scale

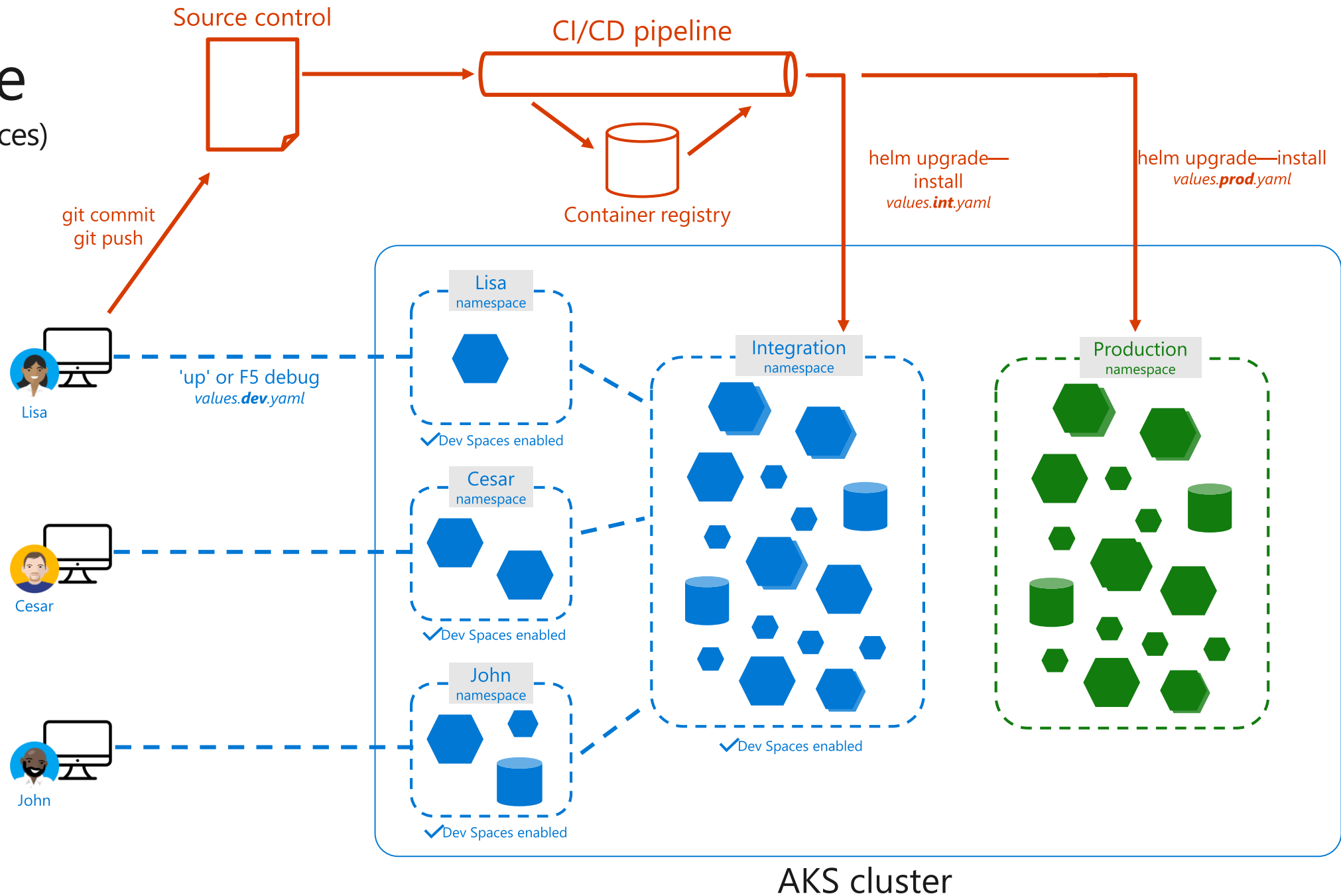
Demo 1

Collaborative development environments for microservices

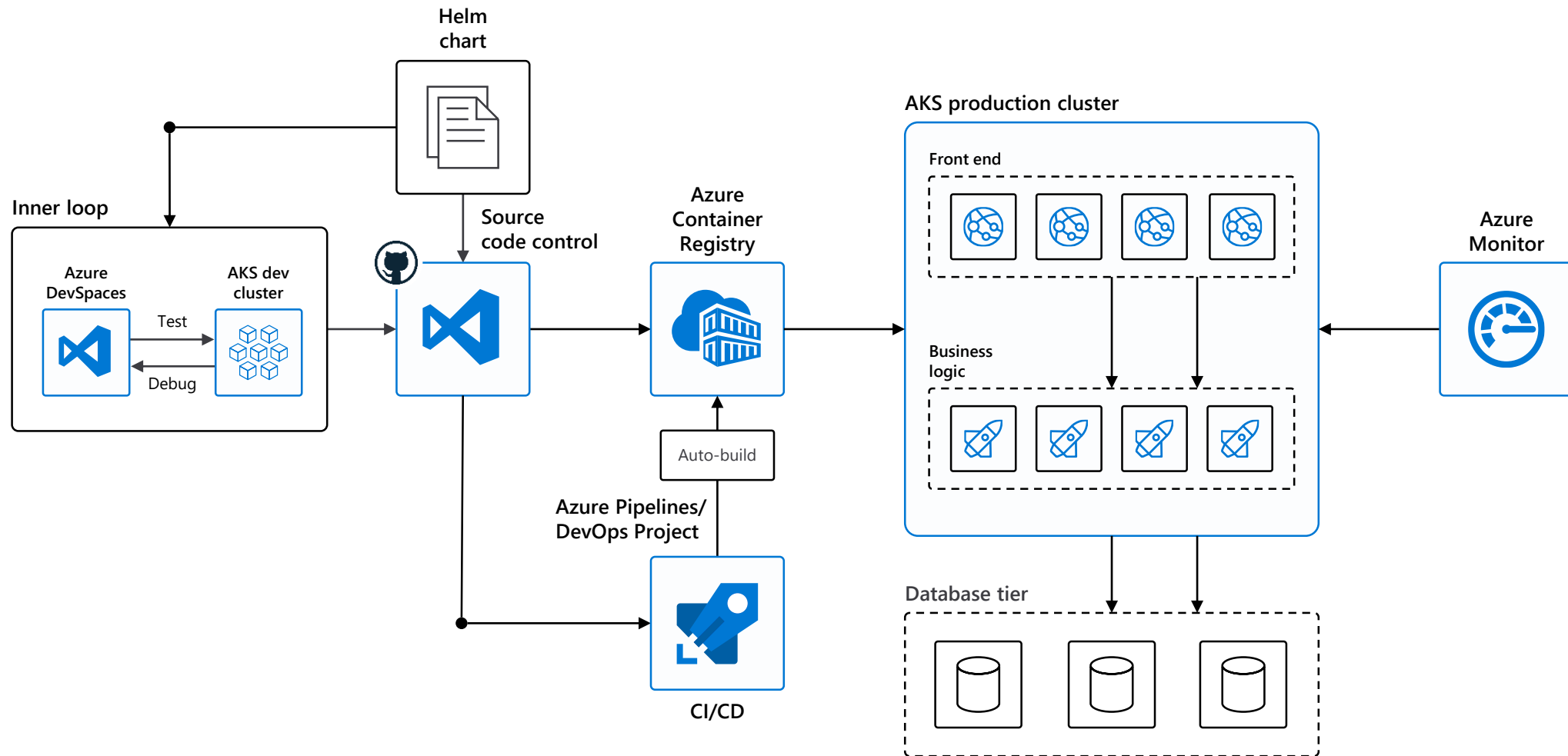


Lifecycle

(Azure Dev Spaces)



End to End Experience



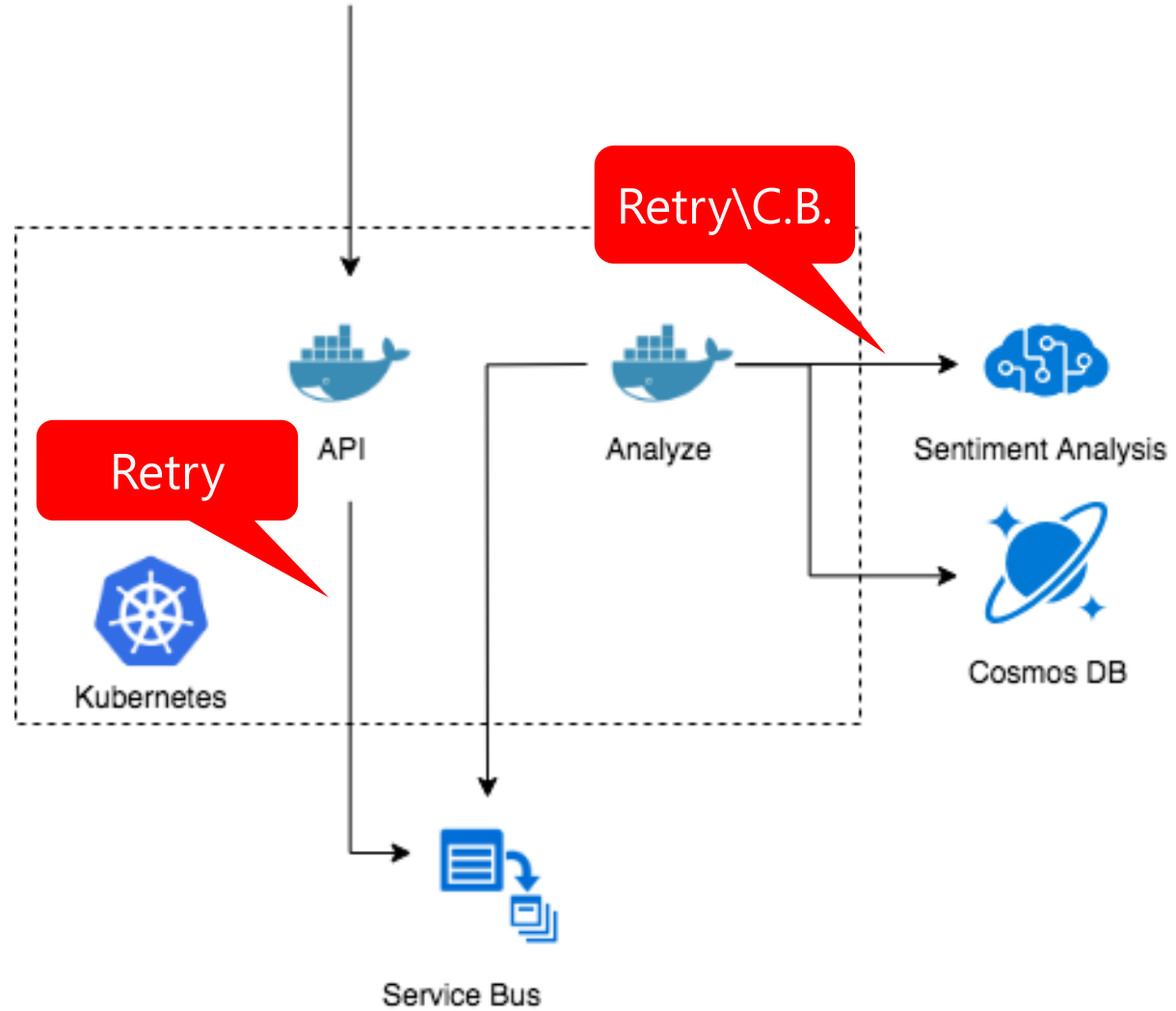
Demo 2

Microservices DevOps





Donald J. Trump
@realDonaldTrump



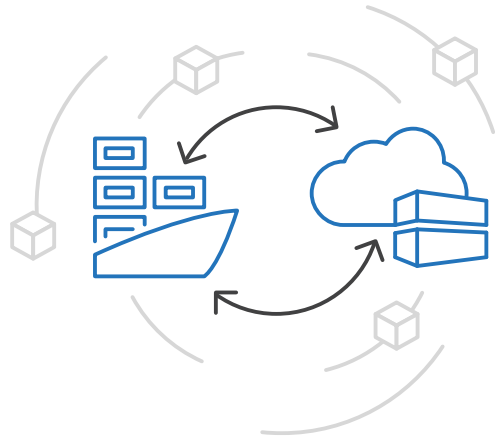
Demo 3

Microservices Scale out and Monitoring

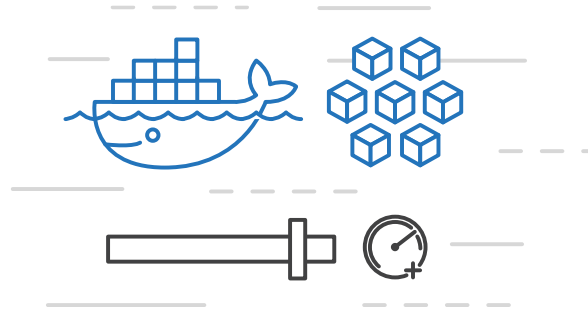


Azure Container Instances (ACI)

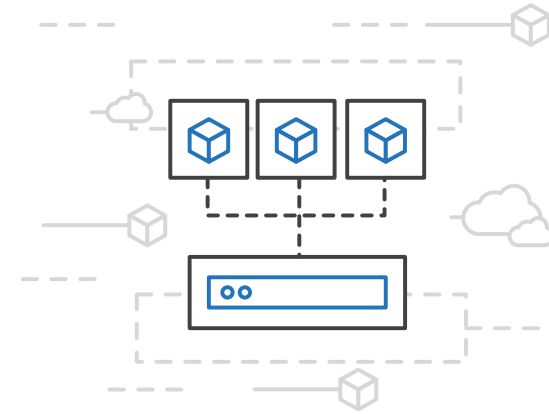
Easily run containers on Azure with a single command



Start using
containers right
away



Cloud-scale
container capacity

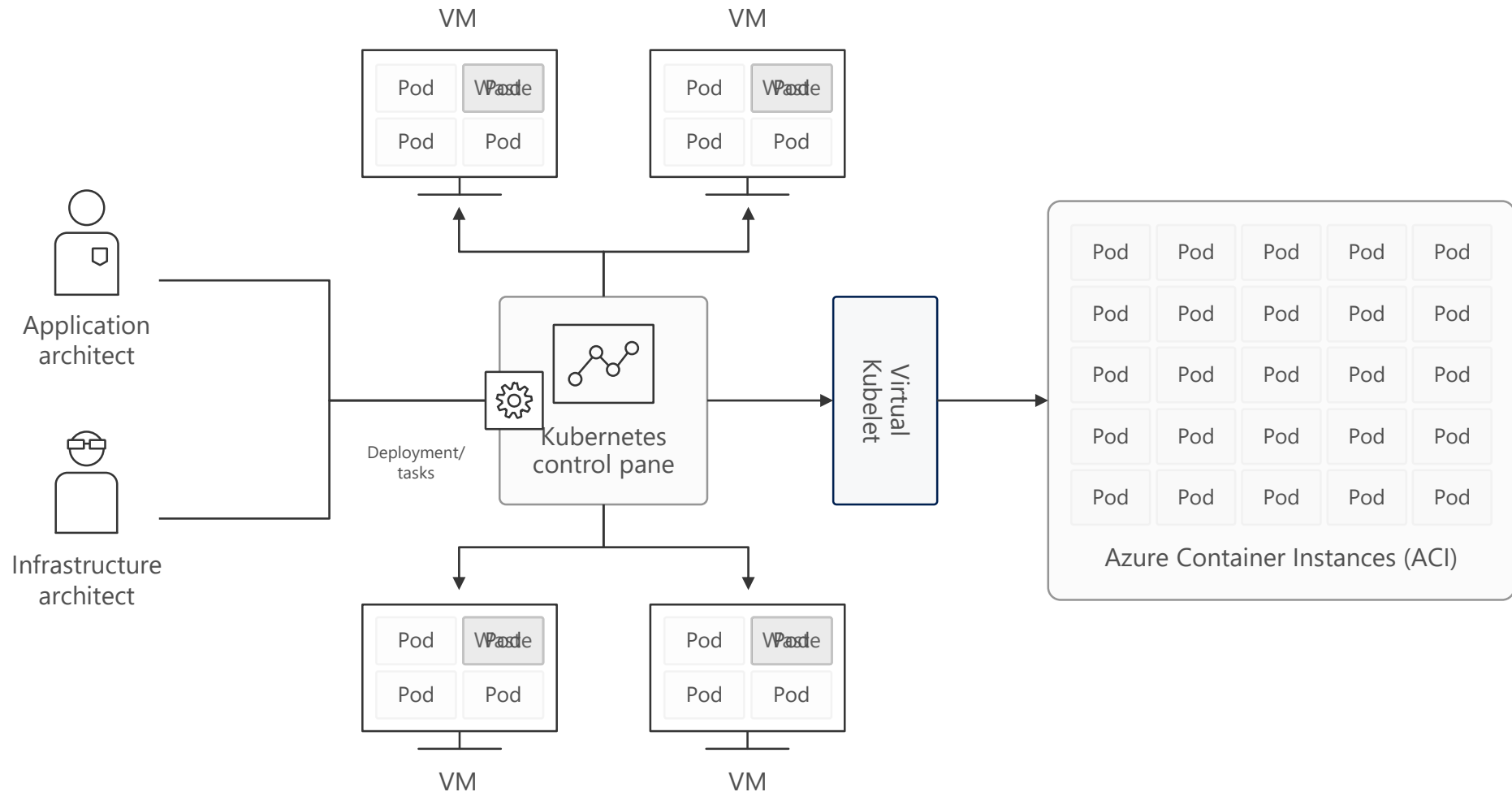


Hyper-visor
isolation



Bursting with the Virtual Kubelet

Azure Container Instances (ACI)



Guide/eBook and sample apps on microservices architecture

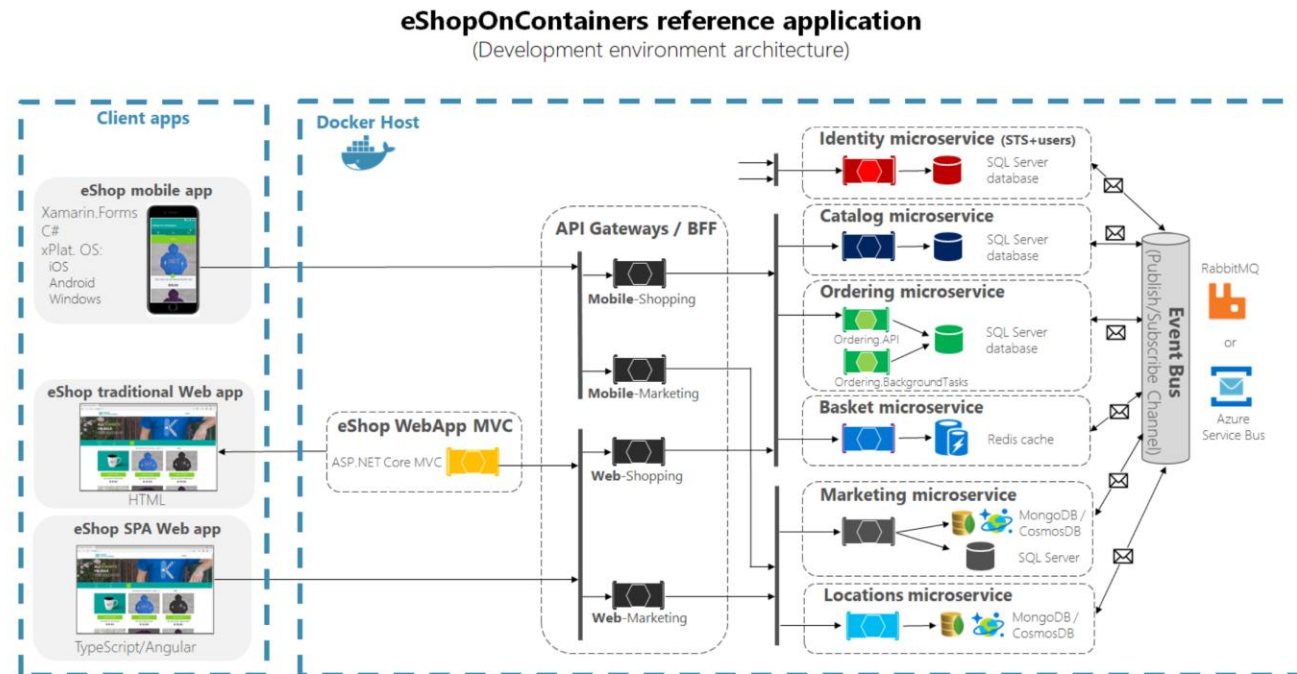
eBook/Guide



<https://aka.ms/microservicesebook>

eShopOnContainers: Reference microservices application

- Intended for .NET developers and solution architects
- Prescriptive guidance on Microservices implementation with .NET Core and Docker



<https://github.com/dotnet-architecture/eShopOnContainers>

Unwatch ▼

1,086

★ Star

7,141

Fork

2,711

