



# Azure Kubernetes Hackfest

*Delivering modern cloud native applications with  
open source technologies on Azure*



Kevin Harris  
Open Source Solutions Lead  
Azure Global Black Belt  
Microsoft Canada

Kevin.Harris@Microsoft.com  
@KevBHar



Ray Kao  
Open Source Solutions Lead  
Azure Global Black Belt  
Microsoft Canada



Ray.Kao@Microsoft.com  
@RayKao

Mathieu Benoit  
Cloud Solution Architect  
Microsoft Canada

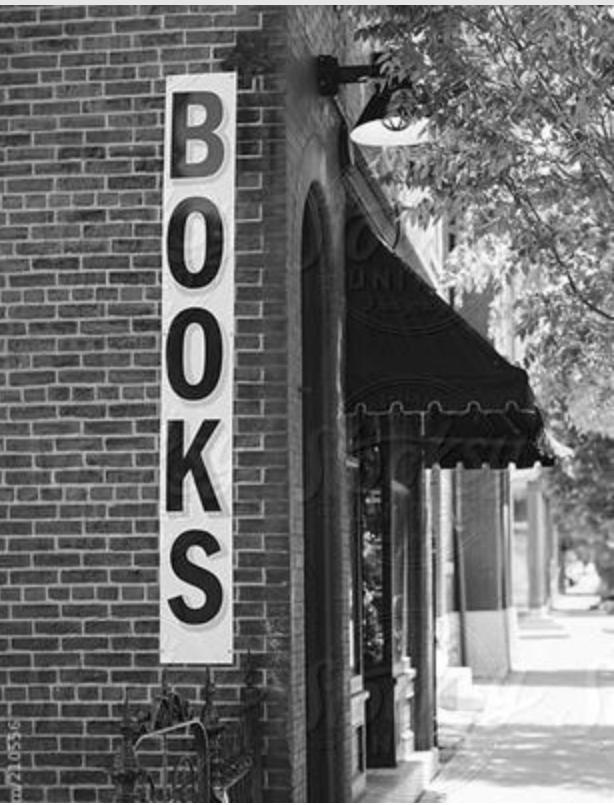


Mathieu.Benoit@Microsoft.com  
<https://aka.ms/mabenoit>

# Agenda

Time	Topic
8:30 – 9:00	Breakfast and Getting Settled
9:00 – 10:00	Cloud Native Applications, Containers, and Docker
10:00 – 11:15	<b>HANDS-ON: Labs 1, 2</b>
11:15 – 12:00	Kubernetes Overview, Azure Kubernetes Service
12:00	<b>Lunch</b>
1:00 – 2:00	<b>HANDS-ON: Labs 3, 4, 5</b>
2:00 – 2:45	Kubernetes Concepts, ACI, ACR, Operations / Management Concepts, Scaling
2:45 – 3:15	<b>HANDS-ON: Labs 6, 7</b>
3:15 – 3:30	Managing Storage and State, Online Service Broker, Upgrading
3:30 - 4:30	<b>HANDS-ON: Labs 8, 9, 10</b>
4:30 – 5:00	Release Tools, Q&A, Wrap-Up

# Why Modernize Apps?



NETFLIX

U B E R

skype™

amazon.com®

# Digital is disrupting industry fundamentals

## Products and technologies

Connected cars



Uconnect

Autonomous cars

DELPHI

Google



Electric vehicles



## Customer expectations

New experiences



waze

brightbox



Sales and service relationships



TRUECar

who can fix my car .com



## Business models

Car sharing services



CAR  
2GO



zipcar



U  
BER

Pay-as-you-go insurance



metromile

Fleet management services

Omnitracs



## Smart ecosystems

Smart homes



TurboCord™

EV Charging Made Easy.

Smart infrastructure



Plug  
Share



ABB



Smart cities

CITYWORKS  
+ a UI LABS Collaboration

Why should customers care about  
containers and microservices?

In reality, they shouldn't...

# They do care about cloud native applications



"Unlimited" Scale

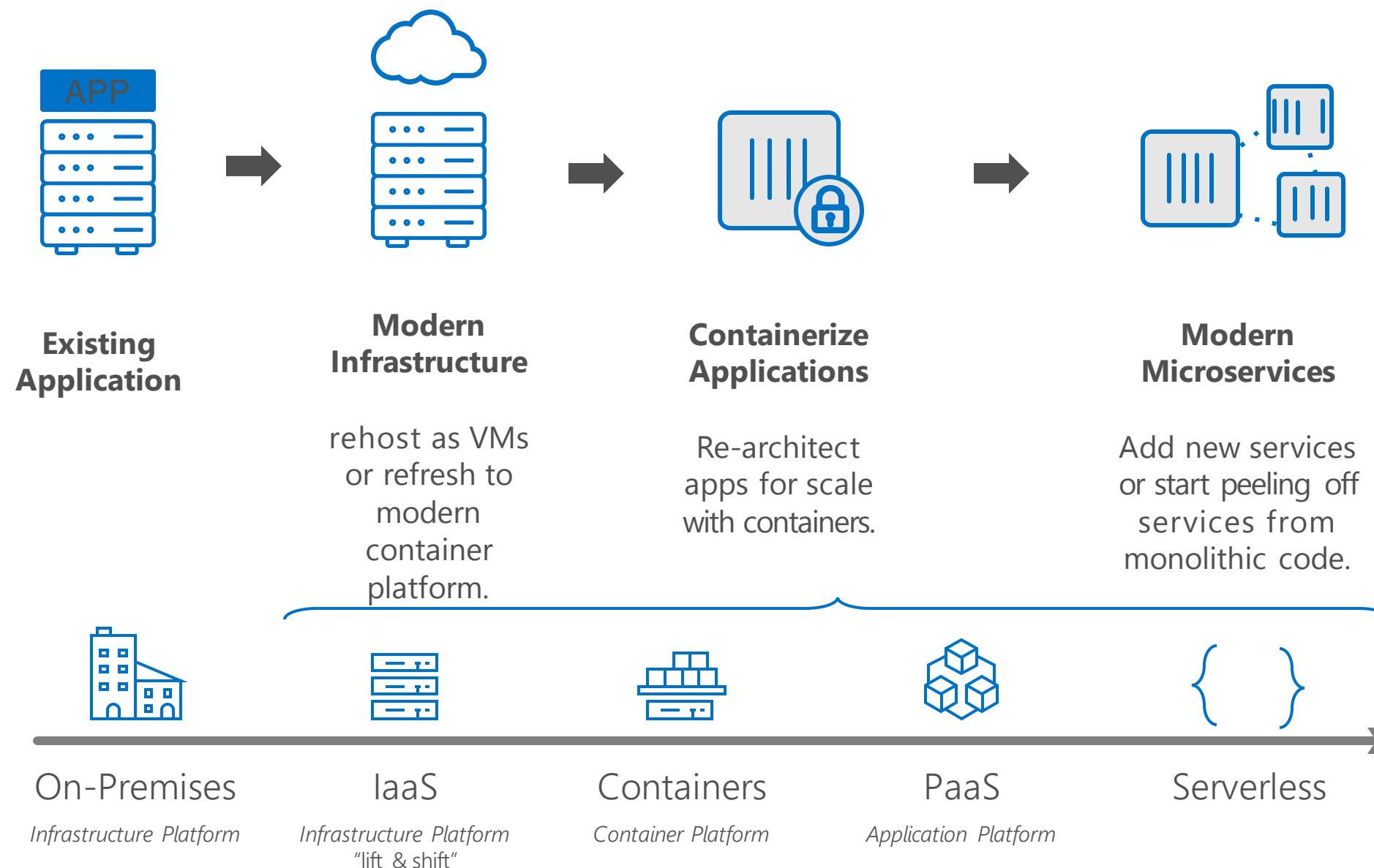


Global reach



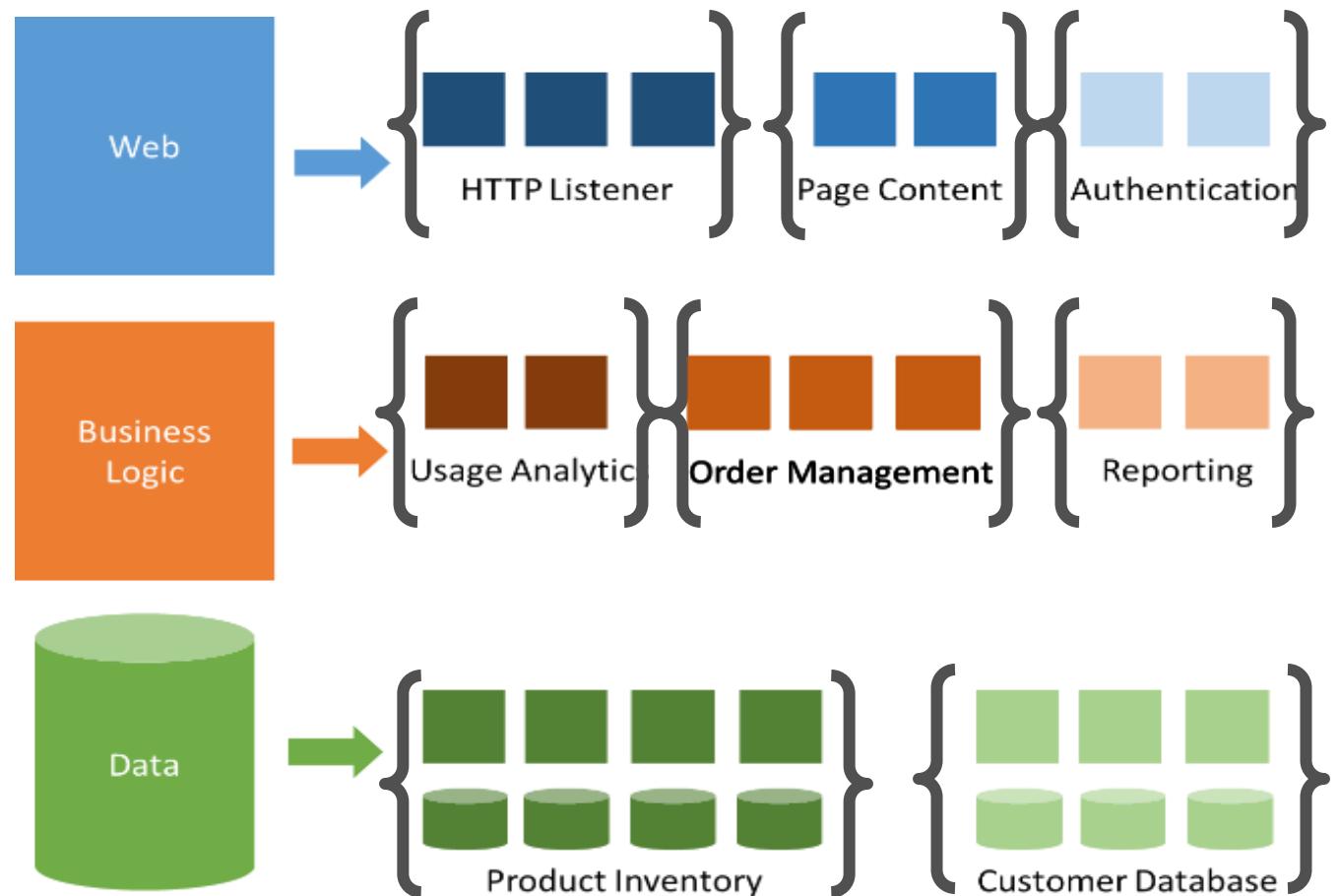
Rapid innovation  
-> time to market

# From traditional app to modern app



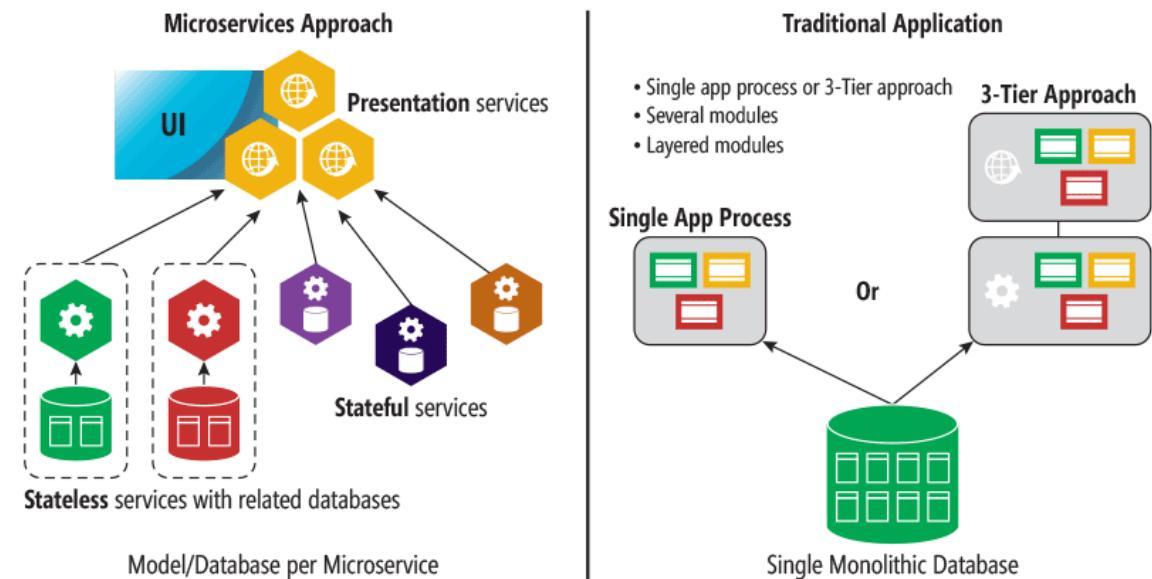
# Modernization with microservices

- Individually built and deployed
- Small, independent services
- Integrate using published API
- Fine-grained, loosely coupled



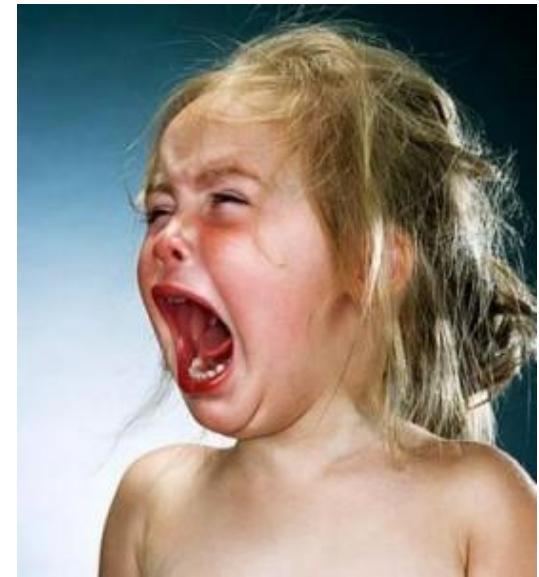
# Microservices Benefits

- ✓ Independent deployments
  - ✓ Enables continuous delivery
  - ✓ No downtime upgrades
  - ✓ Improved scale and resource utilization per service
  - ✓ Fault isolation
  - ✓ Security isolation
  - ✓ Services can be distributed across multiple servers or environments
- ✓ Multiple languages / diversity
  - ✓ Smaller, focused teams
  - ✓ Code can be organized around business capabilities
  - ✓ Autonomous developer teams



# Microservices – The Hard Part

- ✓ Deployment is complex
- ✓ Testing is difficult
- ✓ Debugging is difficult
- ✓ Monitoring/Logging is difficult
- ✓ New service versions must support old/new API contracts
- ✓ Distributed databases make transactions hard
- ✓ Cluster and orchestration tools overhead
- ✓ Distributed services adds more network communication
  - ✓ Increased network hops
  - ✓ Requires failure/recovery code
  - ✓ Need service discovery solution
- ✓ Advanced DevOps capability: short-term pain for long-term gain





## **THE TWELVE-FACTOR APP**

<http://12factor.net>

# 12-Factor Apps (1-5)

1. Single root repo; don't share code with another app
2. Deploy dependent libs with app
3. No config in code; read from environment vars
4. Handle unresponsive app dependencies robustly
5. Strictly separate build, release, & run steps
  - Build: Builds a version of the code repo & gathers dependencies
  - Release: Combines build with config ReleaseId (immutable)
  - Run: Runs app in execution environment

# 12-Factor Apps (6-12)

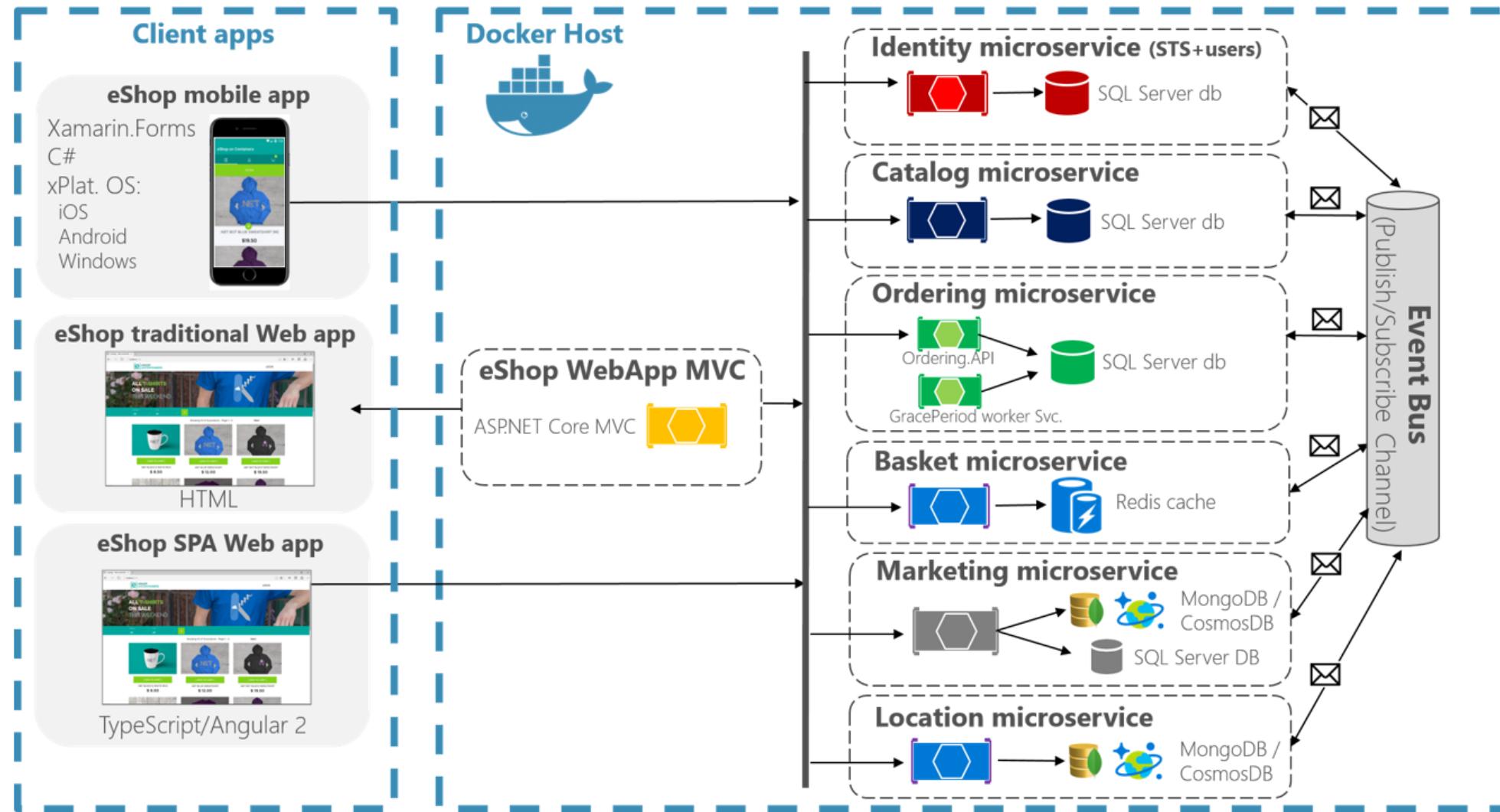
6. App executes as 1+ stateless process & shares nothing
7. App listens on ports; avoid using (web) host
8. Use processes for isolation; multiple for concurrency
9. Processes can crash/be killed quickly & start fast
10. Keep dev, staging, & prod environments similar
11. Log to stdout (dev=console; prod=file & archived)
12. Deploy & run admin tasks (scripts) as processes

# Microservices, Containers & Serverless

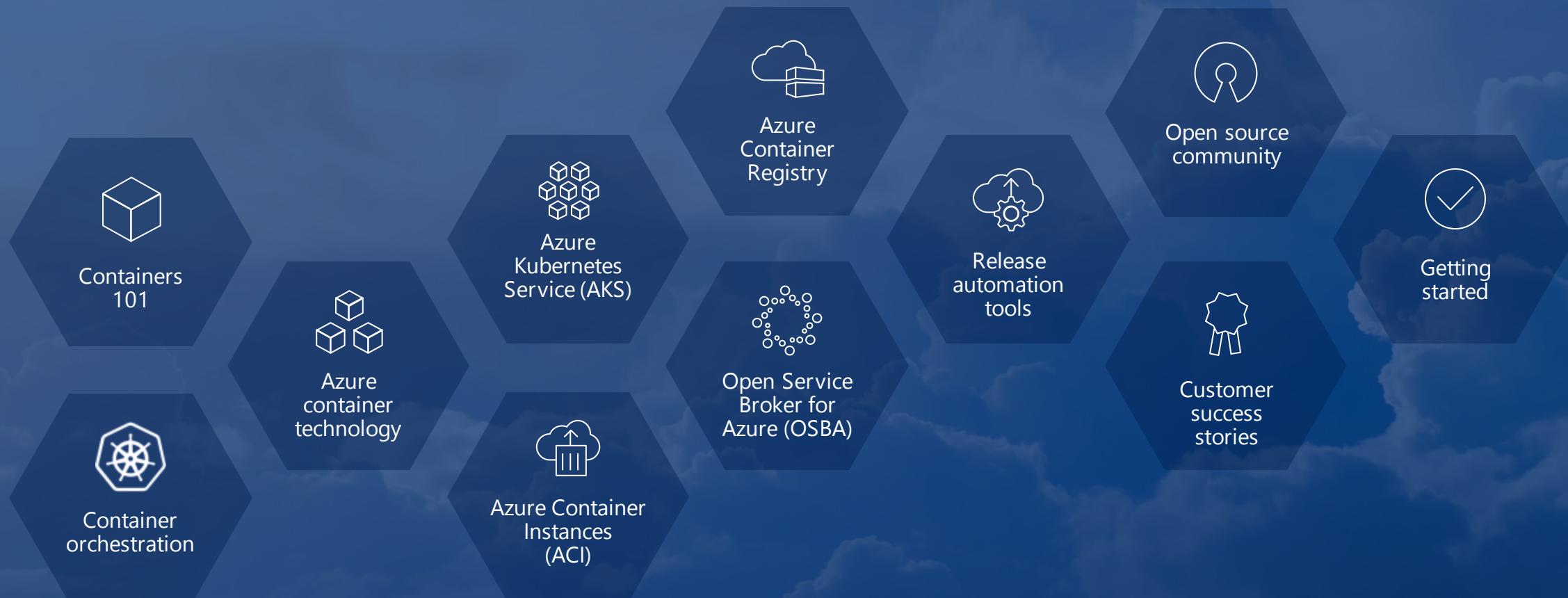
“Microservices is an architectural design point;  
containers & serverless are implementation details that  
often helps.”

# eShopOnContainers Reference App

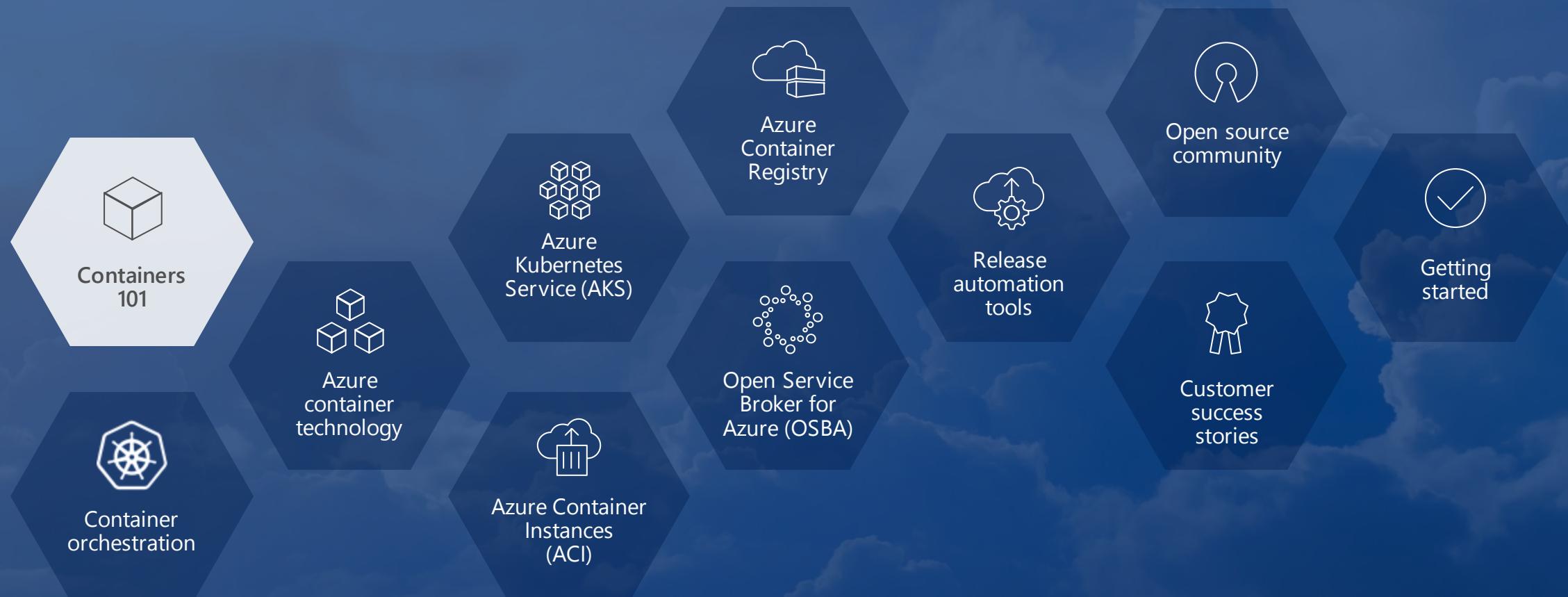
## eShopOnContainers Reference Application - Architecture



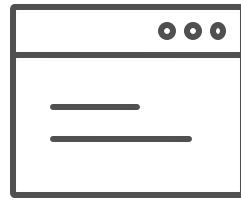
# Table of Contents



# Containers 101



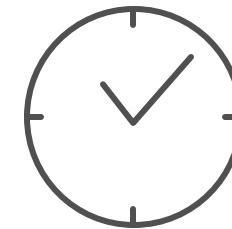
# What we hear from **developers**



I need to create applications  
at a competitive rate without  
worrying about IT



New applications run smoothly  
on my machine but malfunction  
on traditional IT servers



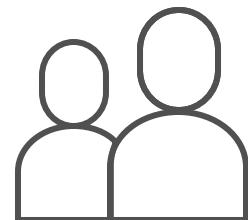
My productivity and application  
innovation become suspended  
when I have to wait on IT



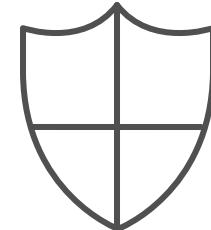
# What we hear from **IT**



I need to manage servers  
and maintain compliance  
with little disruption



I'm unsure of how to integrate  
unfamiliar applications, and I  
require help from developers

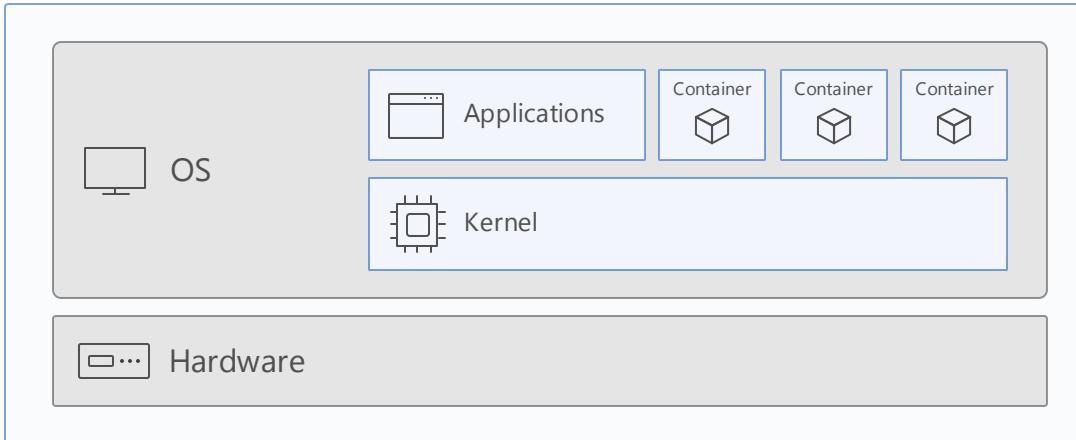


I'm unable to focus on both  
server protection and  
application compliance

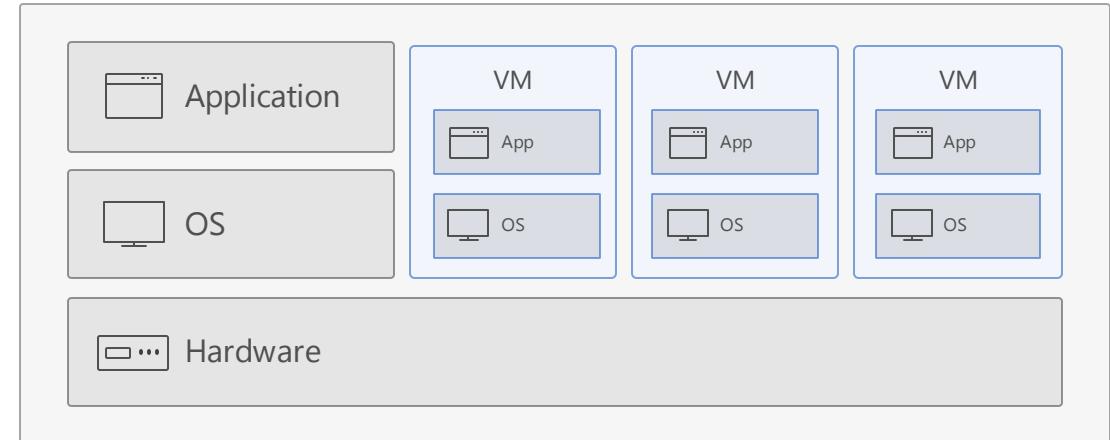


# What is a **container**?

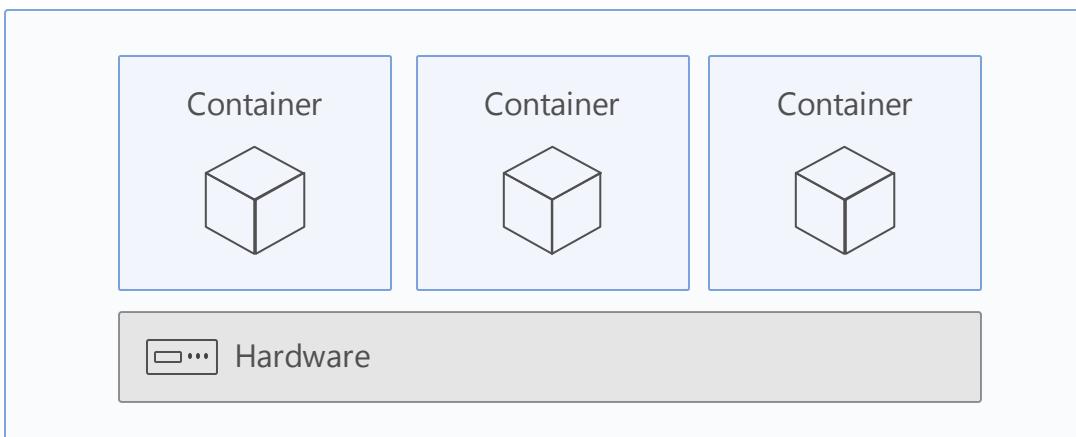
**Containers** = operating system virtualization



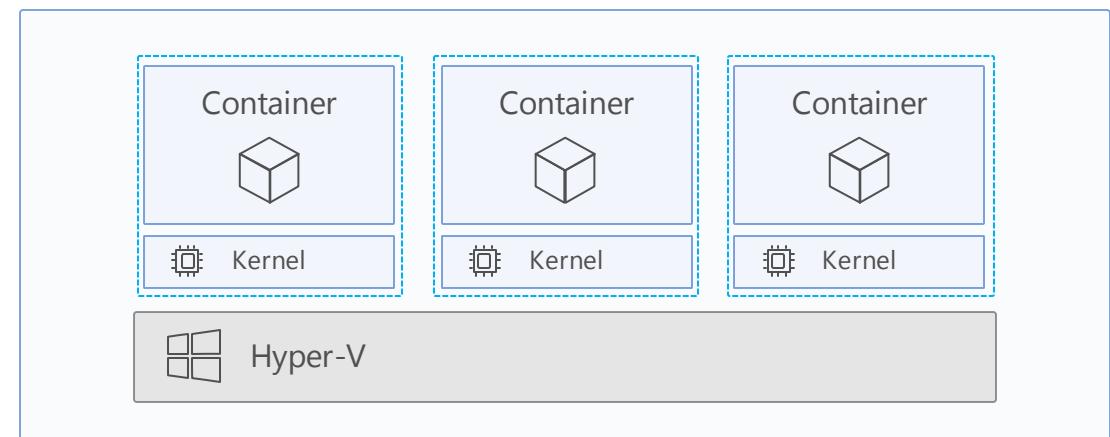
Traditional virtual machines = hardware virtualization



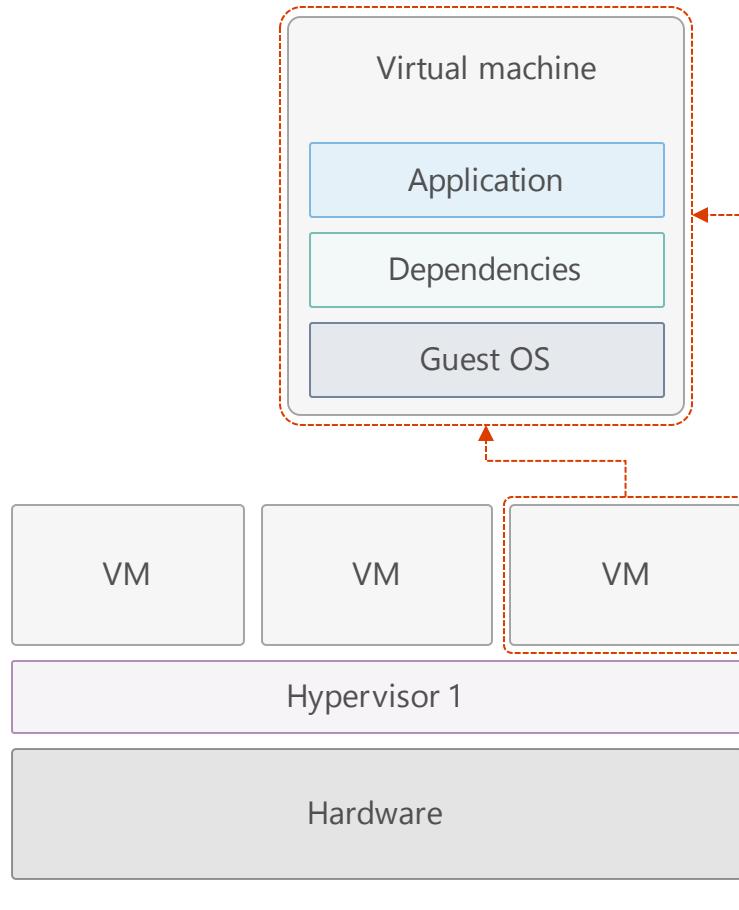
**Windows Server containers:** maximum speed and density



**Hyper-V containers:** isolation plus performance

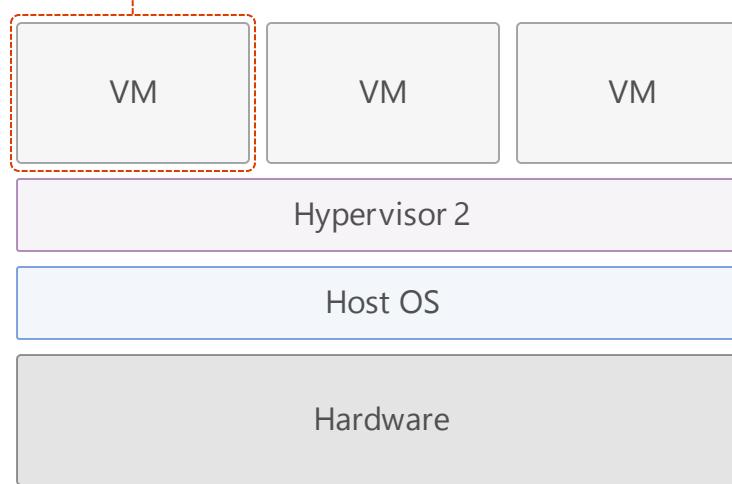


# Virtualization versus **containerization**

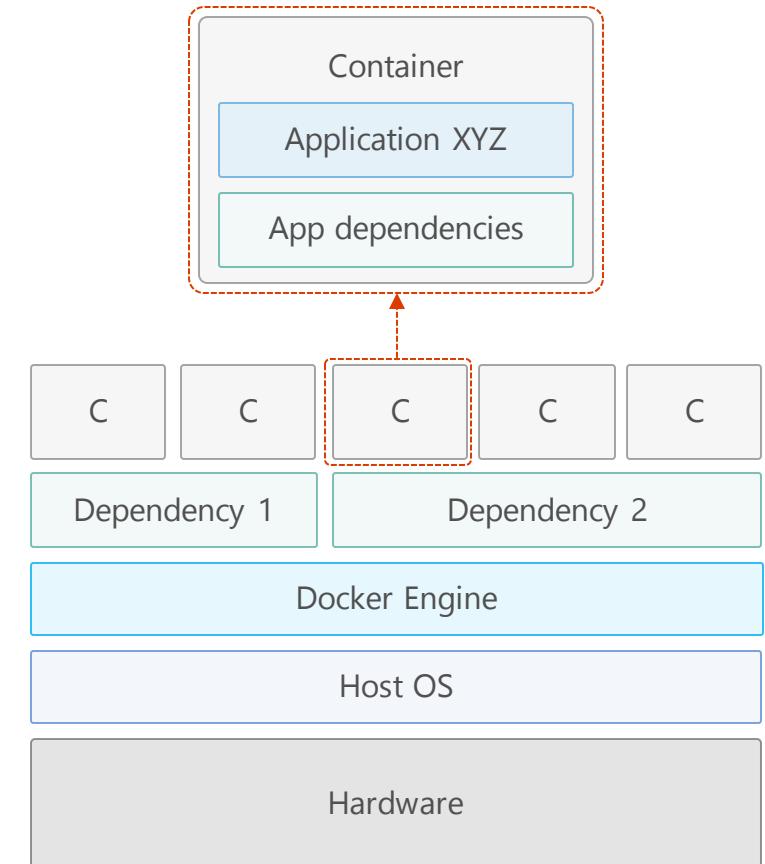


**Virtualization**

Type 1



Type 2



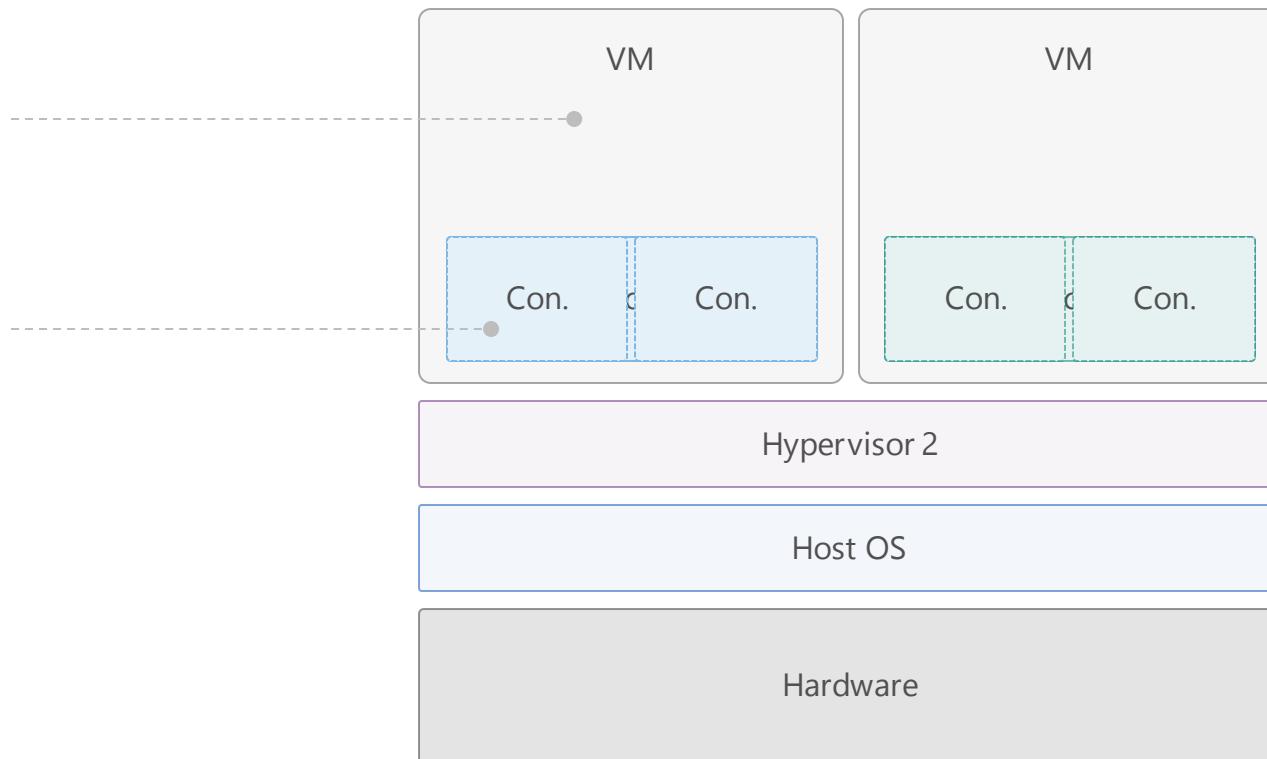
**Containerization**

# The container **advantage**

## Traditional virtualized environment

Low utilization of container resources

Containerization of applications and their dependencies

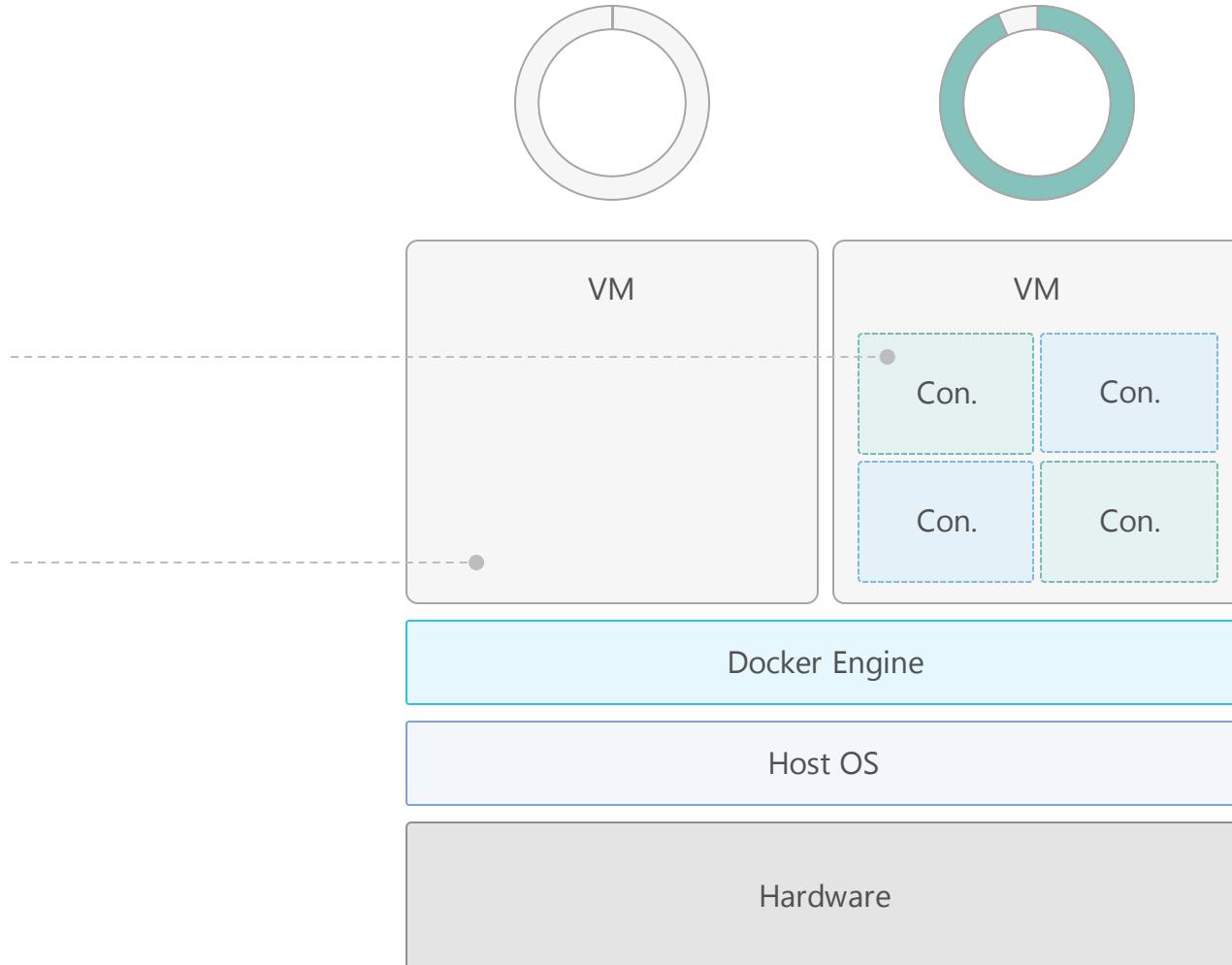


# The container **advantage**

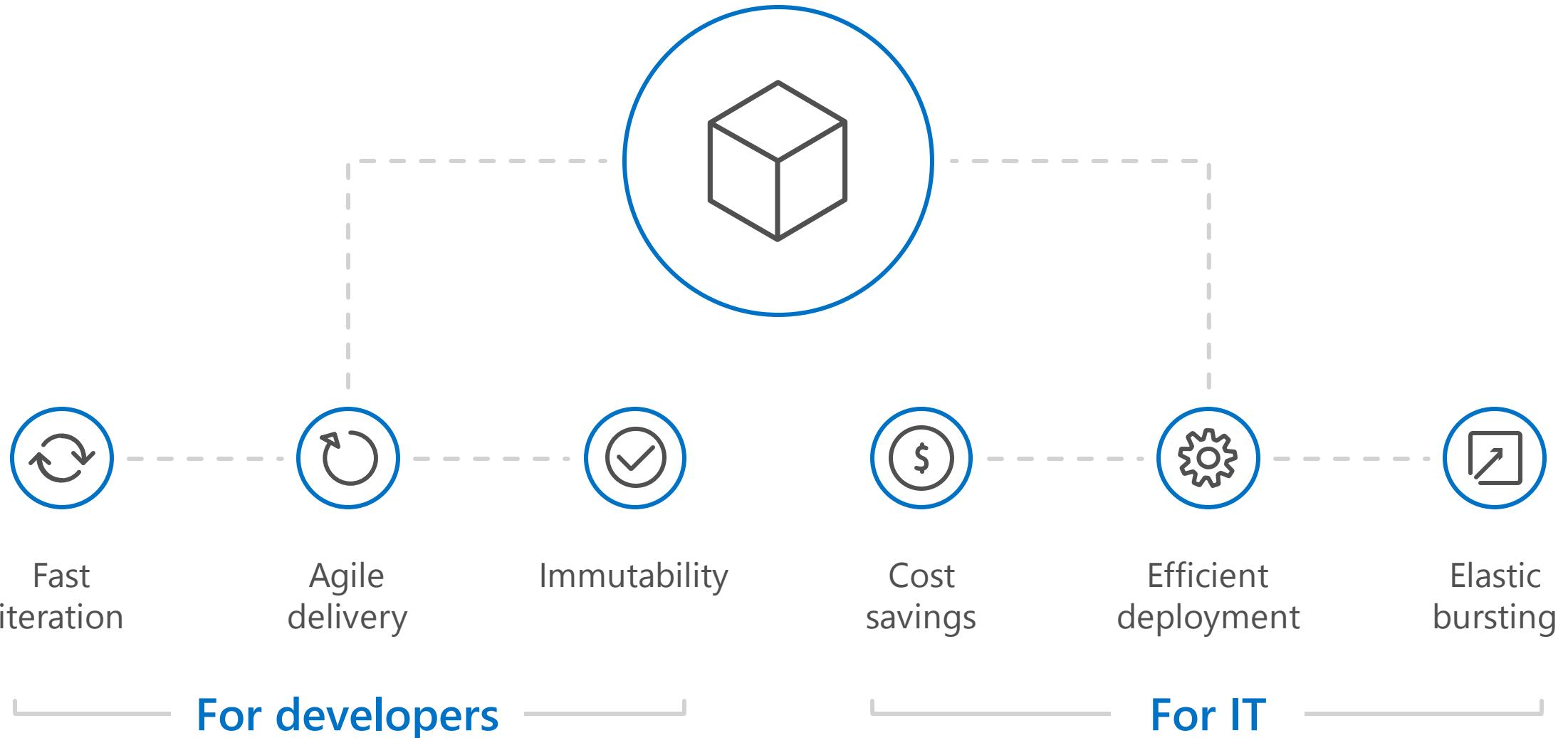
## Containerized environment

Migrate containers and their dependencies to underutilized VMs for improved density and isolation

Decommission unused resources for efficiency gains and cost savings



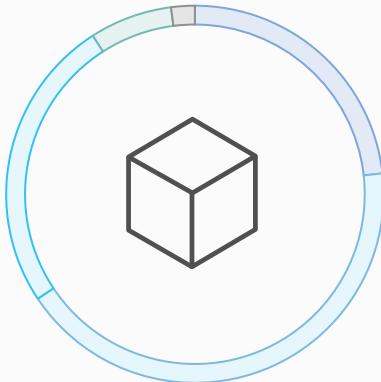
# The container **advantage**



# Containers are gaining **momentum**

Does your organization currently use container technologies?<sup>1</sup>

- 23% My org. is evaluating container technologies
- 42% Yes, my org. currently uses container technologies
- 25% No, my org. is not using container technologies
- 7% Not sure
- 2% Not applicable

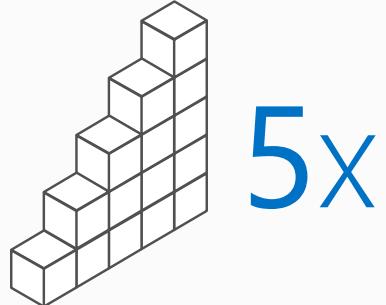


Larger companies are leading adoption.<sup>2</sup>

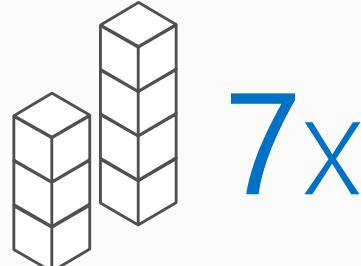
Nearly 60% of organizations running 500 or more hosts are classified as **container dabblers** or adopters.



The average company **QUINTUPLES** its container usage within 9 months.<sup>1</sup>



Container hosts often run **SEVEN** containers at a time.<sup>1</sup>



Containers churn 9 times **FASTER** than VMs.<sup>1</sup>



Source:

1: Datadog: 8 Surprising Facts About Real Docker Adoption; 2: DZone: The DZone Guide to Deploying and Orchestration Containers

# Docker, Docker, Docker

Containers have been around for many years

Linux kernel: cgroups, namespaces

Docker Inc. did not invent them

They created open source software to build and manage containers

Docker makes containers easy

Super easy. Fast learning curve

Docker is a container format and a set of tools

Docker CLI, Docker Engine, Docker Swarm, Docker Compose, Docker Machine



# Docker Container Anatomy

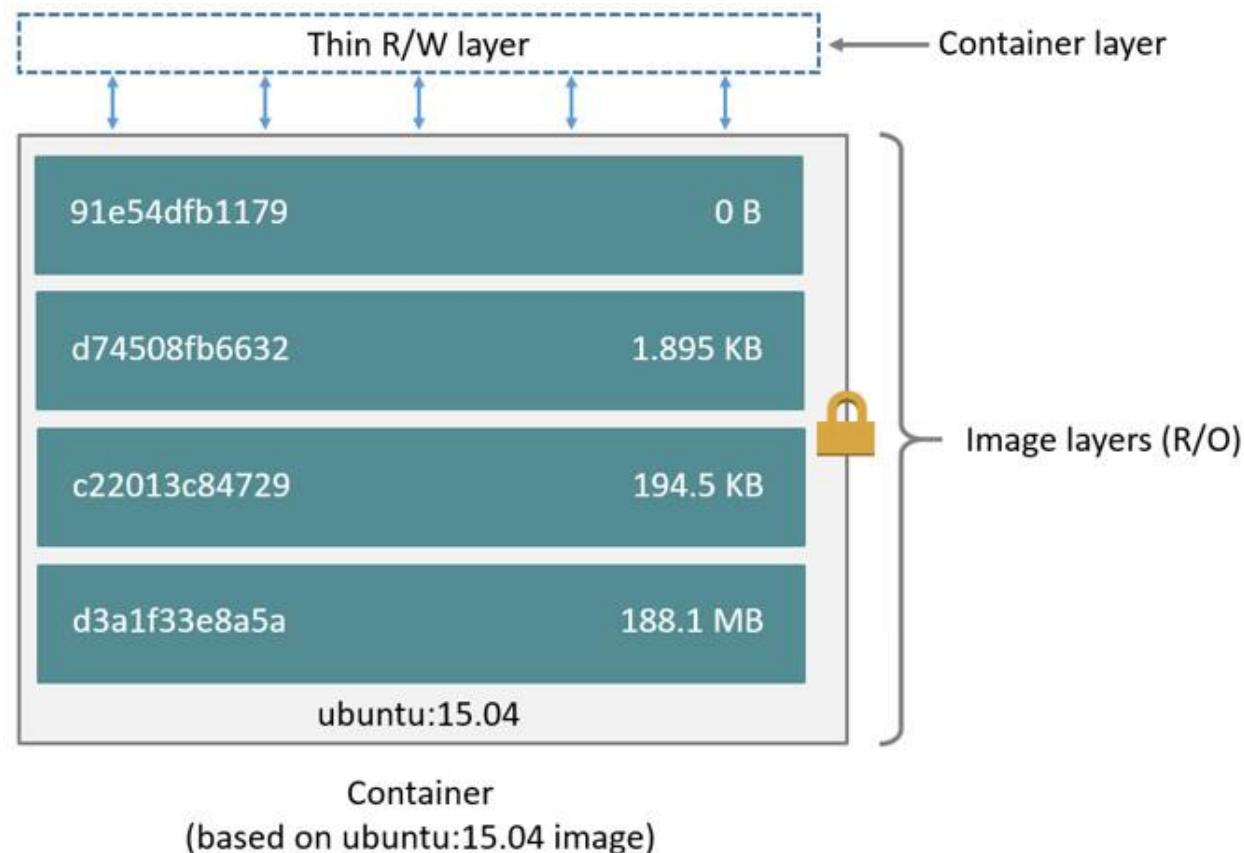
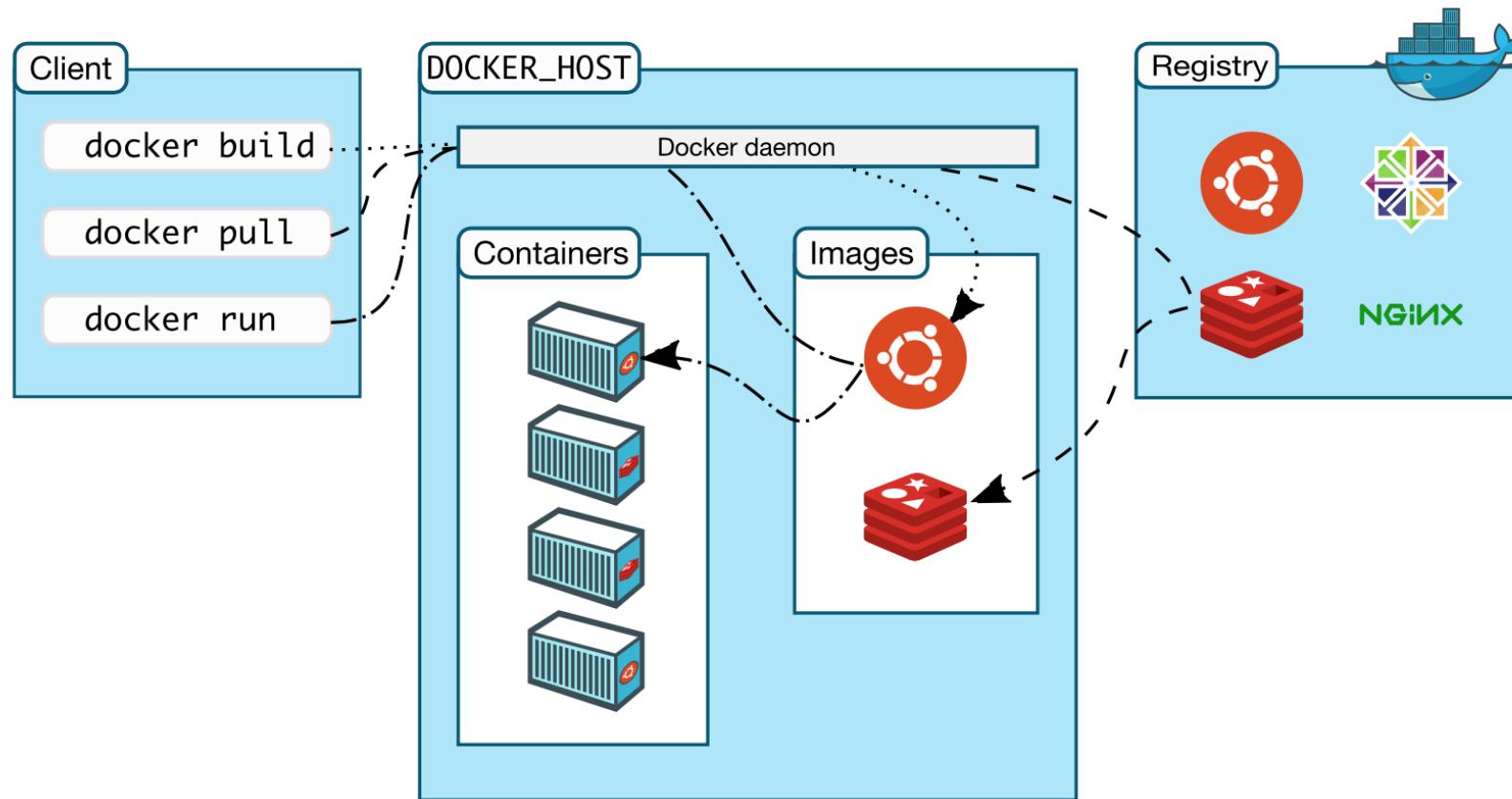


Image source: <https://docs.docker.com/engine/userguide/storagedriver/imagesandcontainers/#images-and-layers>

# Docker Architecture



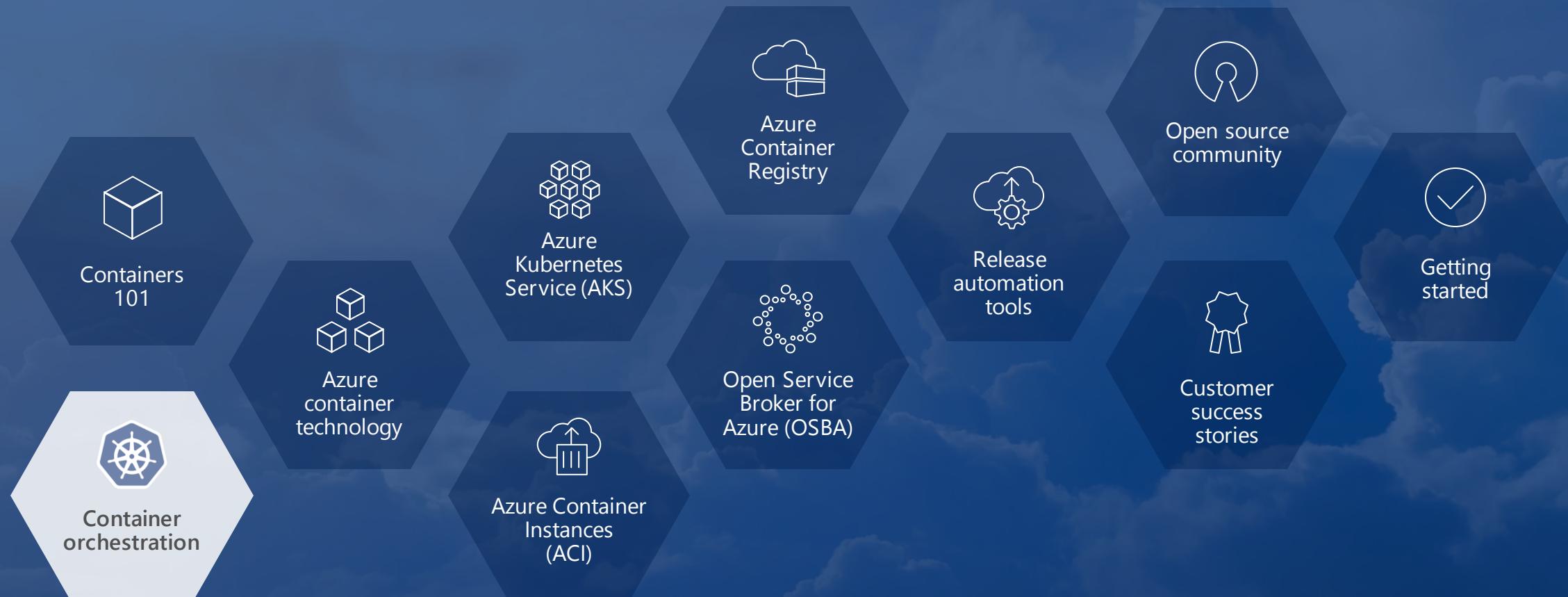
# Demo

- Dockerfile Anatomy
- Container Creation (docker build)
- Image Tagging (docker tag)
- Working with Images (docker push & docker pull)

Labs: 1, 2

[https://github.com/OSSCanada/aks\\_partnerworkshop](https://github.com/OSSCanada/aks_partnerworkshop)

# Container orchestration





*"Distributed apps are sufficiently complicated that they need to be flown by the instruments"*



# The elements of **orchestration**



Scheduling



Affinity/anti-affinity



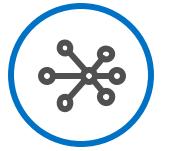
Health monitoring



Failover



Scaling



Networking



Service discovery



Coordinated app upgrades

# Kubernetes: the de-facto orchestrator



## Portable

Public, private, hybrid,  
multi-cloud

## Extensible

Modular, pluggable,  
hookable, composable

## Self-healing

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

# Kubernetes: empowering you to do more



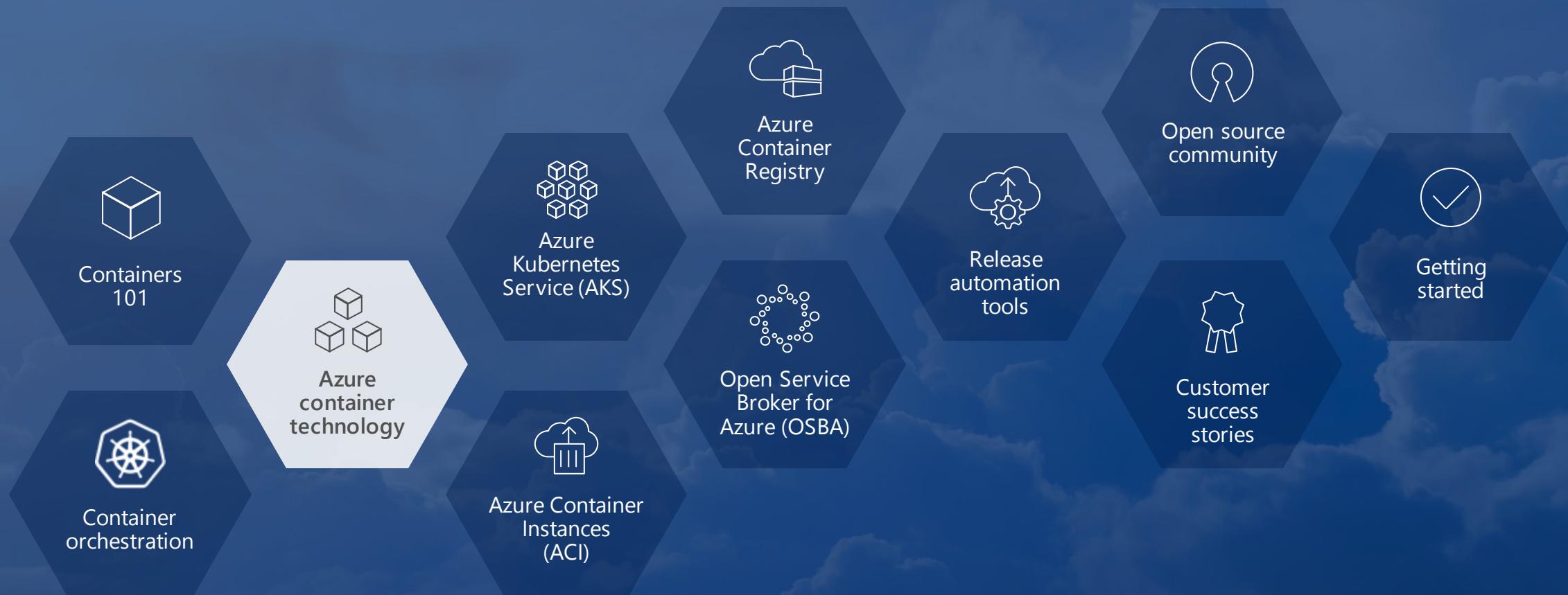
Deploy your  
applications quickly  
and predictably

Scale your  
applications on  
the fly

Roll out  
new features  
seamlessly

Limit hardware  
usage to required  
resources only

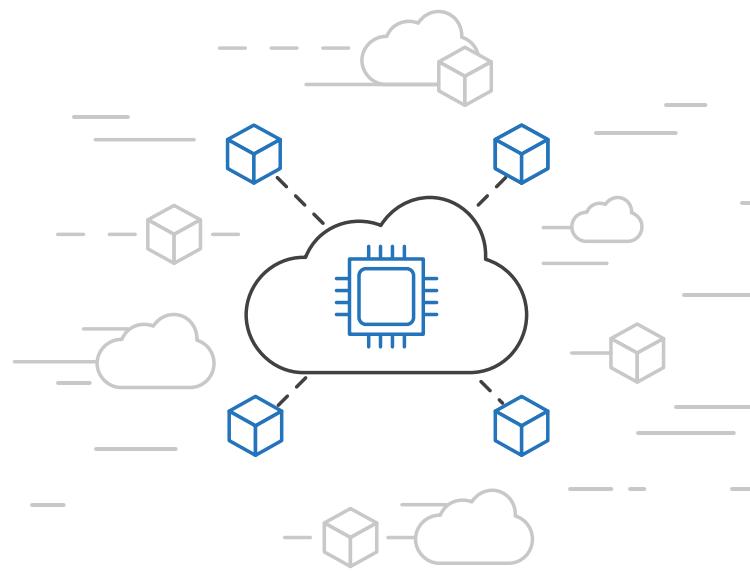
# Azure container technology



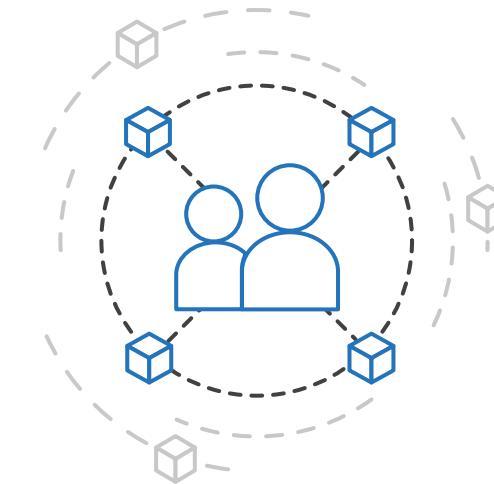
# Azure container **strategy**



Embrace containers  
as ubiquitous

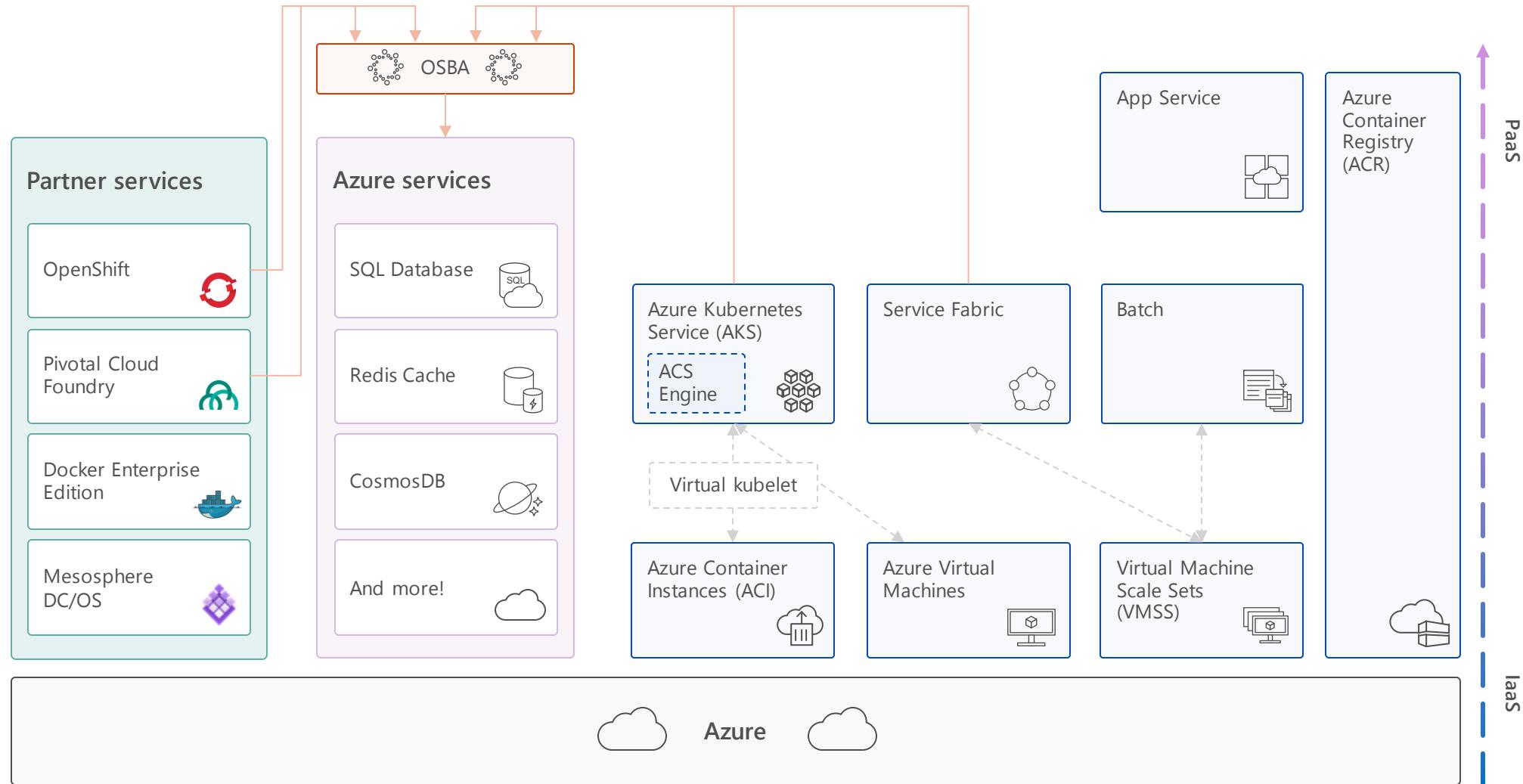


Support containers  
across the compute  
portfolio

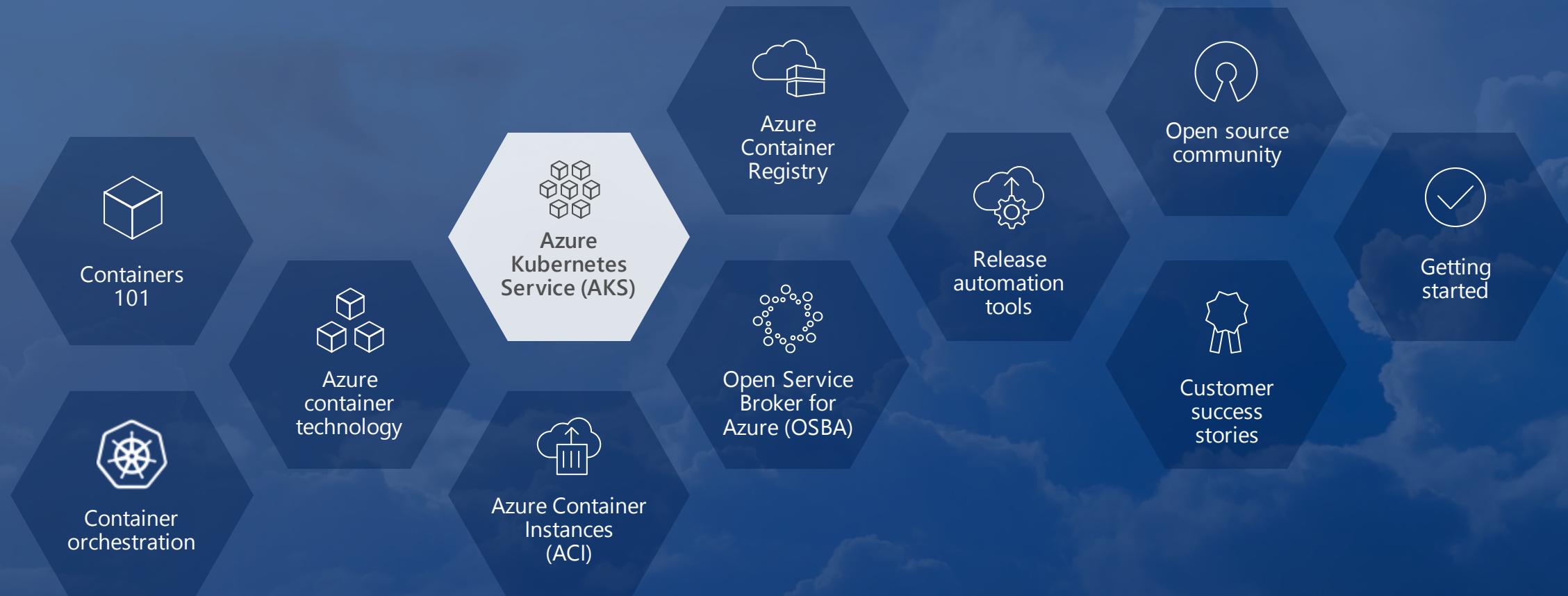


Democratize  
container technology

# Azure container ecosystem



# Azure Kubernetes Service (AKS)





Azure Kubernetes Service (AKS)



Azure Container Instances (ACI)



Azure Container Registry

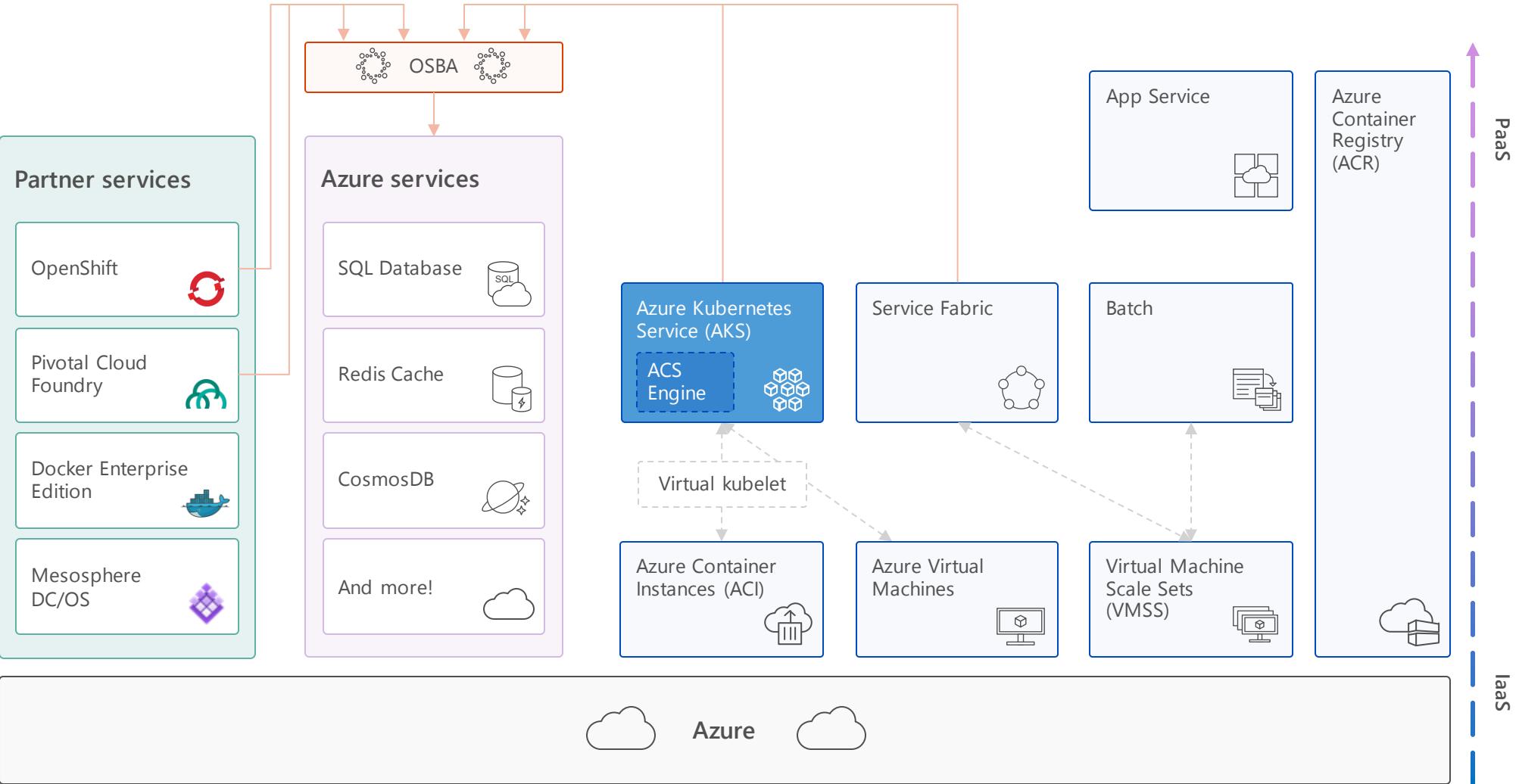


Open Service Broker API (OSBA)



Release Automation Tools

# Azure Kubernetes Service (AKS)





Azure Kubernetes  
Service (AKS)



Azure Container  
Instances (ACI)



Azure Container  
Registry



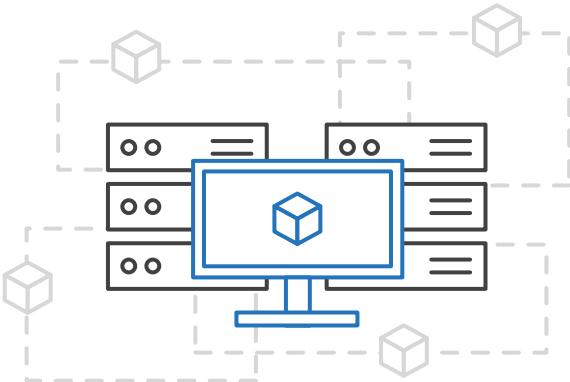
Open Service  
Broker API (OSBA)



Release  
Automation Tools

# Azure Kubernetes Service (AKS)

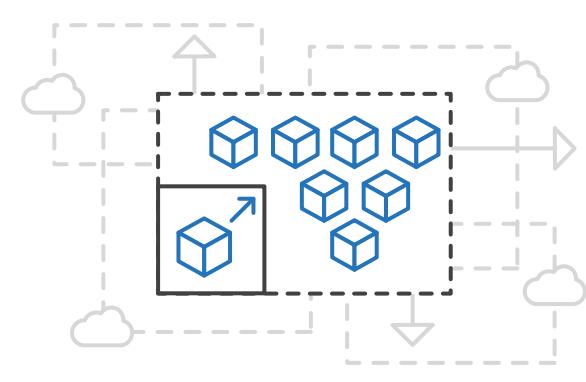
Simplify the deployment, management, and operations of Kubernetes



Focus on your  
containers not the  
infrastructure



Work how you  
want with open-  
source APIs



Scale and run  
applications with  
confidence



Azure Kubernetes Service (AKS)



Azure Container Instances (ACI)



Azure Container Registry



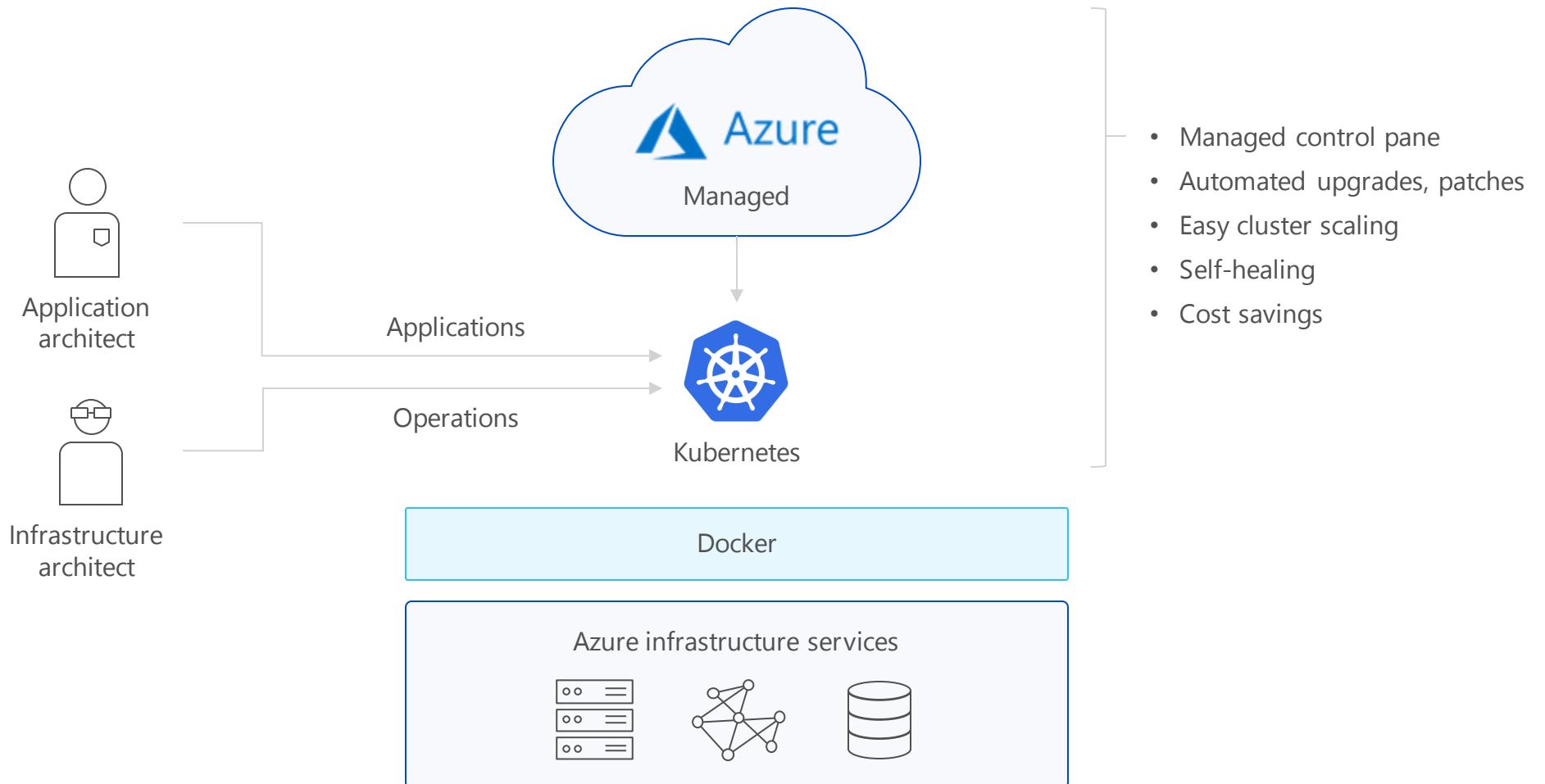
Open Service Broker API (OSBA)



Release Automation Tools

# Azure Kubernetes Service (AKS)

A fully managed Kubernetes cluster





Azure Kubernetes  
Service (AKS)



Azure Container  
Instances (ACI)



Azure Container  
Registry



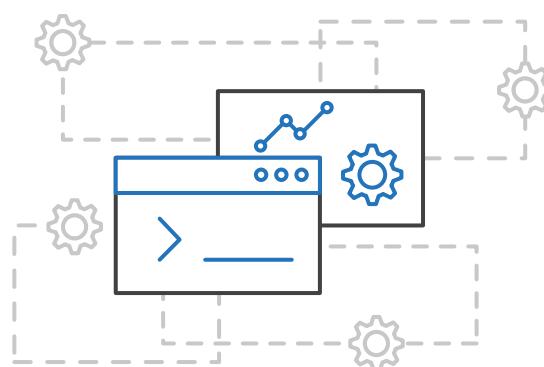
Open Service  
Broker API (OSBA)



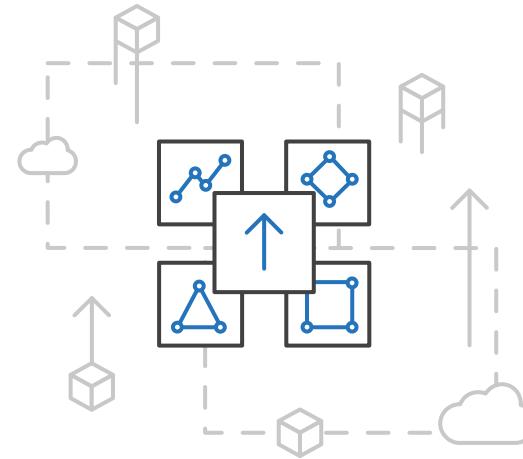
Release  
Automation Tools

# Azure Kubernetes Service (AKS)

## Azure Kubernetes Service Engine (ACS-Engine)



A proving ground  
for new features



Enables custom  
deployments



Available  
on GitHub

# Demo

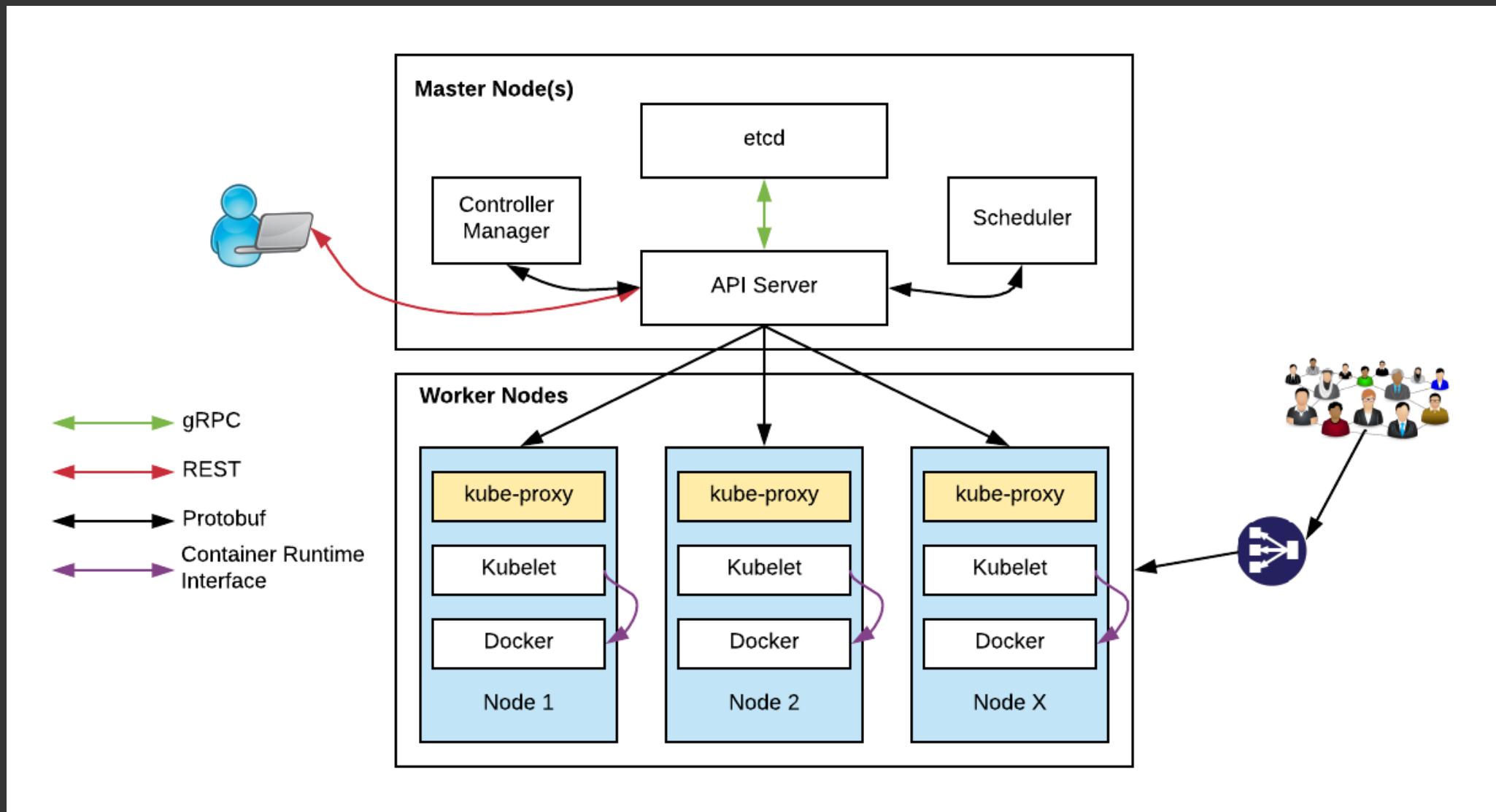
- Cluster Creation
- Get Nodes
- Get Namespaces
- Get Deployments
- Get Pods
- Get Services

Labs: 3, 4, 5

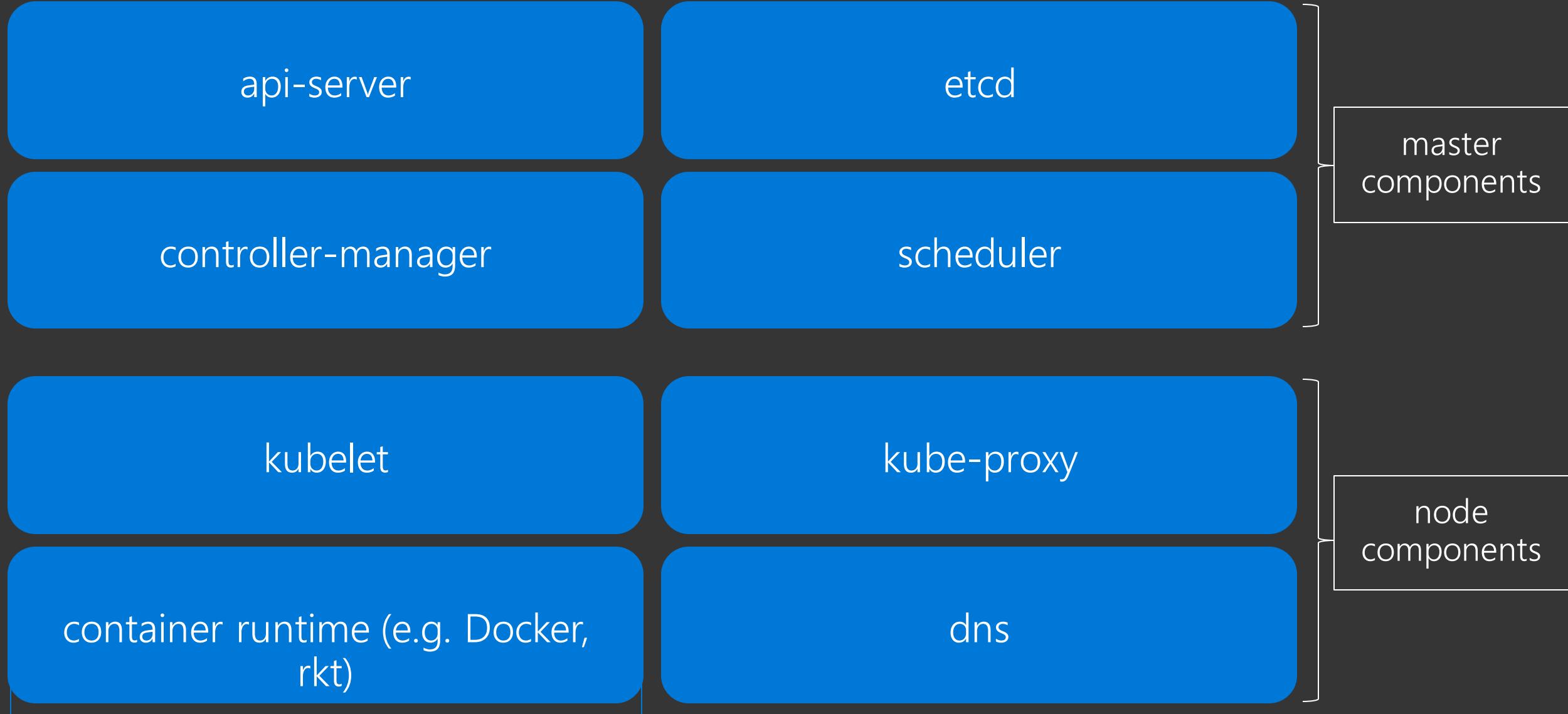
[https://github.com/OSSCanada/aks\\_partnerworkshop](https://github.com/OSSCanada/aks_partnerworkshop)

# Kubernetes Concepts

# Kubernetes Architecture



# Kubernetes Architecture Components



# kubectl: CLI to run commands against a Kubernetes cluster

- Swiss Army Knife: run deployments, exec into containers, view logs, etc.
- Syntax largely the same as docker
- Pronounced “koob sea tee el”...
  - Or “koob cuddle”

# Kubernetes Resources

pod

deployment

service

replica set

ingress

daemon set, job

namespace

secret, config-map

# Kubernetes Resources

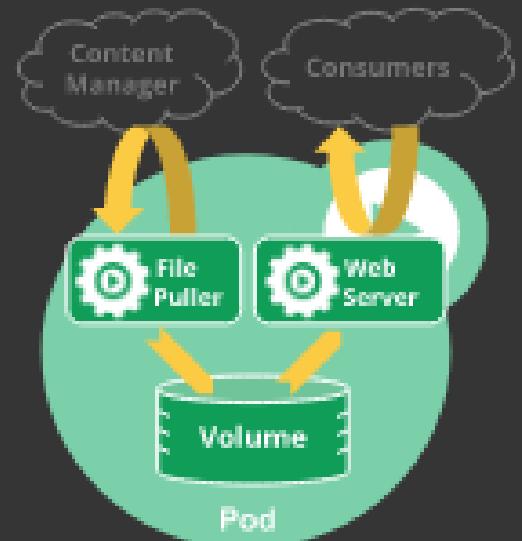
pod

deployment

service

# What is a Pod?

- Pod is the basic building block in kubernetes
- Pods are how containers are delivered
- Can be multiple containers (eg - side car)
- Encapsulates container(s), storage, network IP, and options on how to run
- Use Deployment resources to deploy
  - ReplicaSet
  - StatefulSet
  - DaemonSet
  - Job
  - InitContainer



# What is a Deployment?

- Provides declarative updates for Pods and Replica Sets
- Deployment describes "desired state"
- Can:
  - Create deployment to rollout ReplicaSet
  - Declare new state for pods (eg – new imageTag)
  - Rollback to earlier state
  - Scale up/down
  - Check rollout history
  - Clean-up

# What is a Service?

- Defines a logical set of pods (*your microservice*)
- Essentially a virtual load balancer in front of pods

# Kubernetes manifest: Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: redis-django
  labels:
    app: web
spec:
  containers:
    - name: key-value-store
      image: redis
      ports:
        - containerPort: 6379
    - name: frontend
      image: django
      ports:
        - containerPort: 8000
```

# Interact with pods

```
$ kubectl get po --all-namespaces
```

```
$ kubectl describe po/my-pod
```

```
$ kubectl logs my-pod
```

```
# Run bash in container
```

```
$ kubectl exec -it my-pod bash
```

# Kubernetes manifest: Deployment

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: smackweb-deploy
spec:
  replicas: 5
  template:
    metadata:
      labels:
        app: smackweb
    spec:
      containers:
        - name: smackweb
          image: chzbrgr71/smackweb
          ports:
            - containerPort: 8080
```

# Service Types

- ClusterIP:
  - Exposes the service on a cluster-internal IP. Choosing this value makes the service only reachable from within the cluster
- NodePort:
  - Exposes the service on each Node's IP at a static port (the NodePort)
  - Connect from outside the cluster by requesting <NodeIP>:<NodePort>
- LoadBalancer:
  - Exposes the service externally using a cloud provider's load balancer

# Kubernetes manifest: Service

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  type: ClusterIP
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

## Pod

**Name:** my-app-xaj712

**Labels:**

version=1

app=my-app

# ReplicaSet

**Replicas:** 2

**Label Selectors:**

version=1

app=my-app

## Pod

**Name:** my-app-xaj712

**Labels:**

version=1

app=my-app

## Pod

**Name:** my-app-lka2ja

**Labels:**

version=1

app=my-app

# Service: My App

```
graph LR; Service[Service: My App] --- RS[ReplicaSet]; RS --- Pod1[Pod]; RS --- Pod2[Pod]
```

**Name:** my-app

**Label Selectors:**

app=my-app

## ReplicaSet

**Replicas:** 2

**Label Selectors:**

version=1

app=my-app

### Pod

**Name:** my-app-xaj712

**Labels:**

version=1

app=my-app

### Pod

**Name:** my-app-lka2ja

**Labels:**

version=1

app=my-app

# Service: My App

**Name:** my-app

**Label Selectors:**

app=my-app

## ReplicaSet

**Replicas:** 2

**Label Selectors:**

version=1

app=my-app

### Pod

**Name:** my-app-xaj712

**Labels:**

version=1

app=my-app

### Pod

**Name:** my-app-lka2ja

**Labels:**

version=1

app=my-app

### Pod

**Name:** my-app-123hfa

**Labels:**

version=canary

app=my-app

# Service: My App

**Name:** my-app

**Label Selectors:**

app=my-app

## ReplicaSet

**Replicas:** 2

**Label Selectors:**

version=1

app=my-app

### Pod

**Name:** my-app-xaj712

**Labels:**

version=1

app=my-app

### Pod

**Name:** my-app-lka2ja

**Labels:**

version=1

app=my-app

## ReplicaSet

**Replicas:** 1

**Label Selectors:**

version=2

app=my-app

# Service: My App

**Name:** my-app

**Label Selectors:**

app=my-app

## ReplicaSet

**Replicas:** 2

**Label Selectors:**

version=1

app=my-app

### Pod

**Name:** my-app-xaj712

**Labels:**

version=1

app=my-app

### Pod

**Name:** my-app-lka2ja

**Labels:**

version=1

app=my-app

## ReplicaSet

**Replicas:** 1

**Label Selectors:**

version=2

app=my-app

### Pod

**Name:** my-app-19sfd

**Labels:**

version=2

app=my-app

# Service: My App

**Name:** my-app

**Label Selectors:**

app=my-app

## ReplicaSet

**Replicas:** 1

**Label Selectors:**

version=1

app=my-app

### Pod

**Name:** my-app-xaj712

**Labels:**

version=1

app=my-app

## ReplicaSet

**Replicas:** 2

**Label Selectors:**

version=2

app=my-app

### Pod

**Name:** my-app-19sdfd

**Labels:**

version=2

app=my-app

### Pod

**Name:** my-app-xaj712

**Labels:**

version=2

app=my-app

# Service: My App

**Name:** my-app

**Label Selectors:**

app=my-app

## ReplicaSet

**Replicas:** 0

**Label Selectors:**

version=1

app=my-app

## ReplicaSet

**Replicas:** 2

**Label Selectors:**

version=2

app=my-app

## Pod

**Name:** my-app-19sdfd

**Labels:**

version=2

app=my-app

## Pod

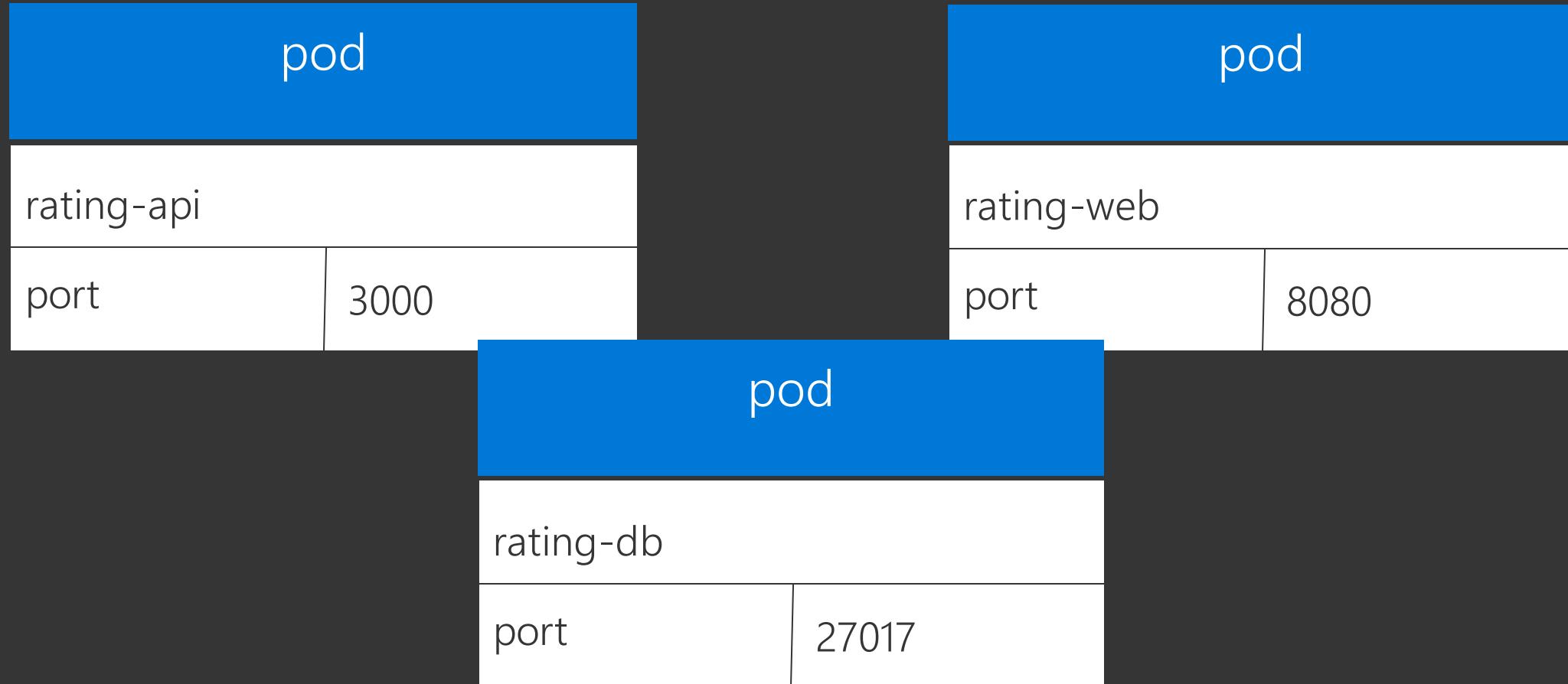
**Name:** my-

app-0q2a87

**Labels:**

version=2

app=my-app



# Deployment

label	app=heroes-web , tier=web
image	azurecr.io/rating-web:v1

pod

heroes-web-1

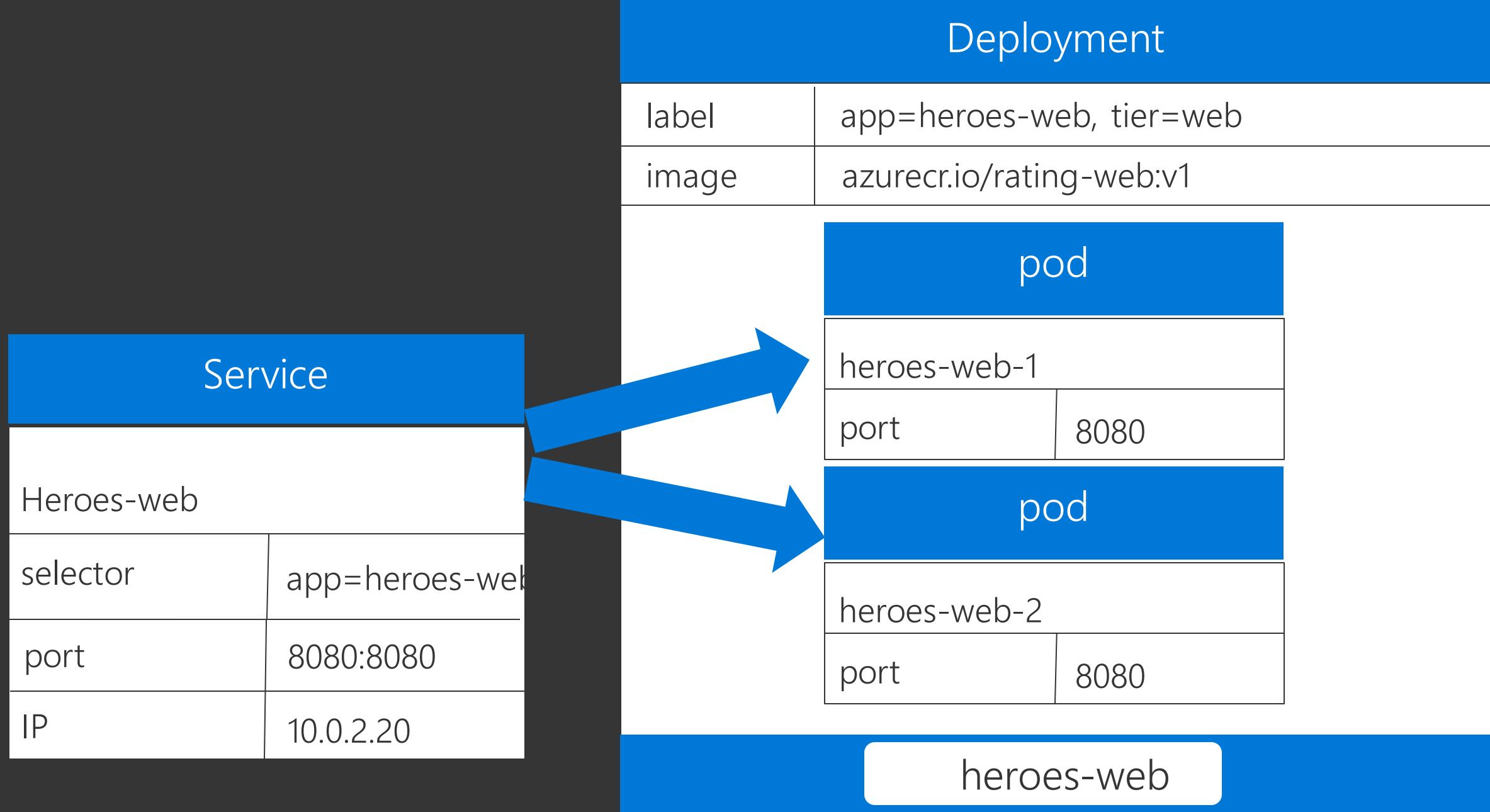
port 8080

pod

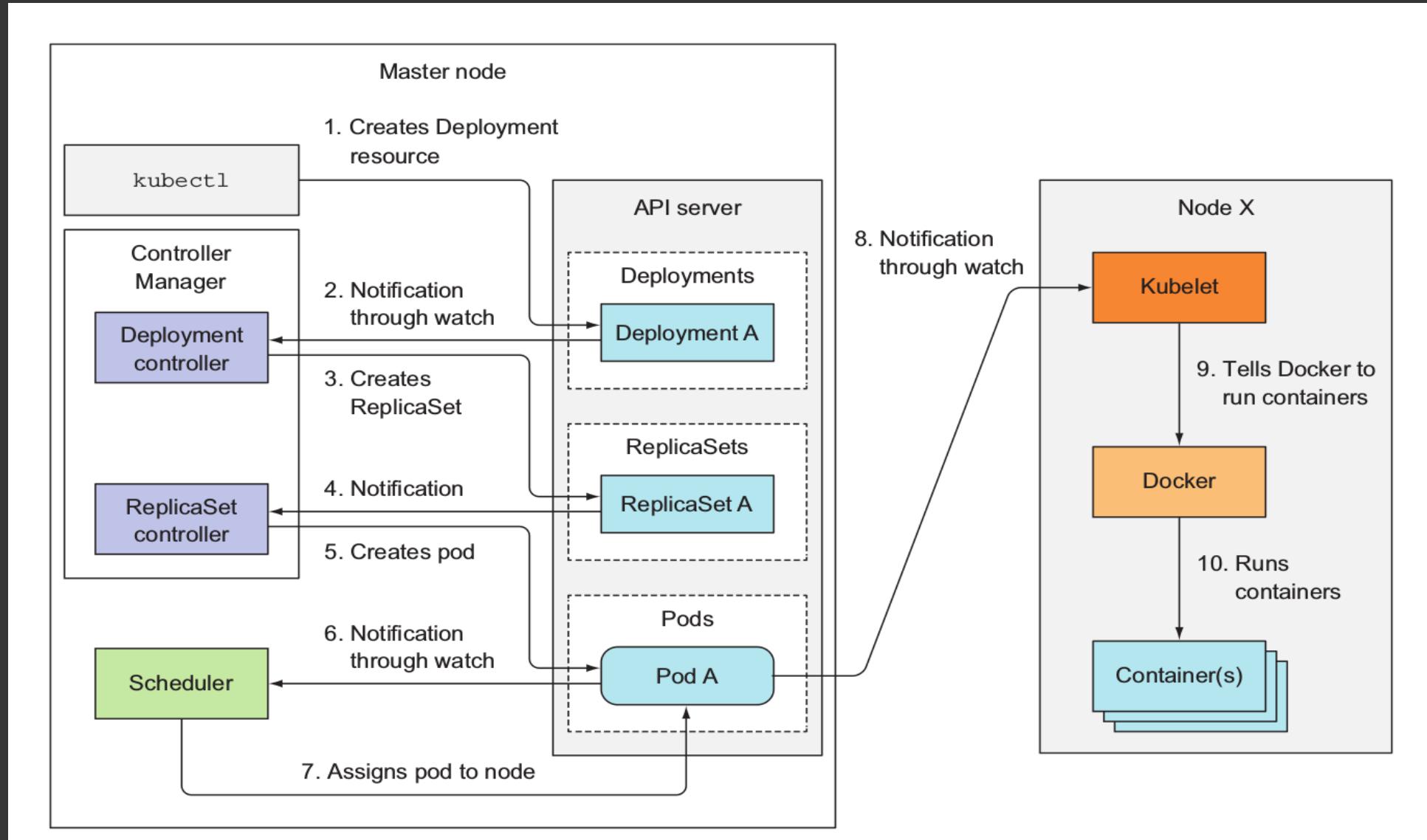
heroes-web-2

port 8080

heroes-web



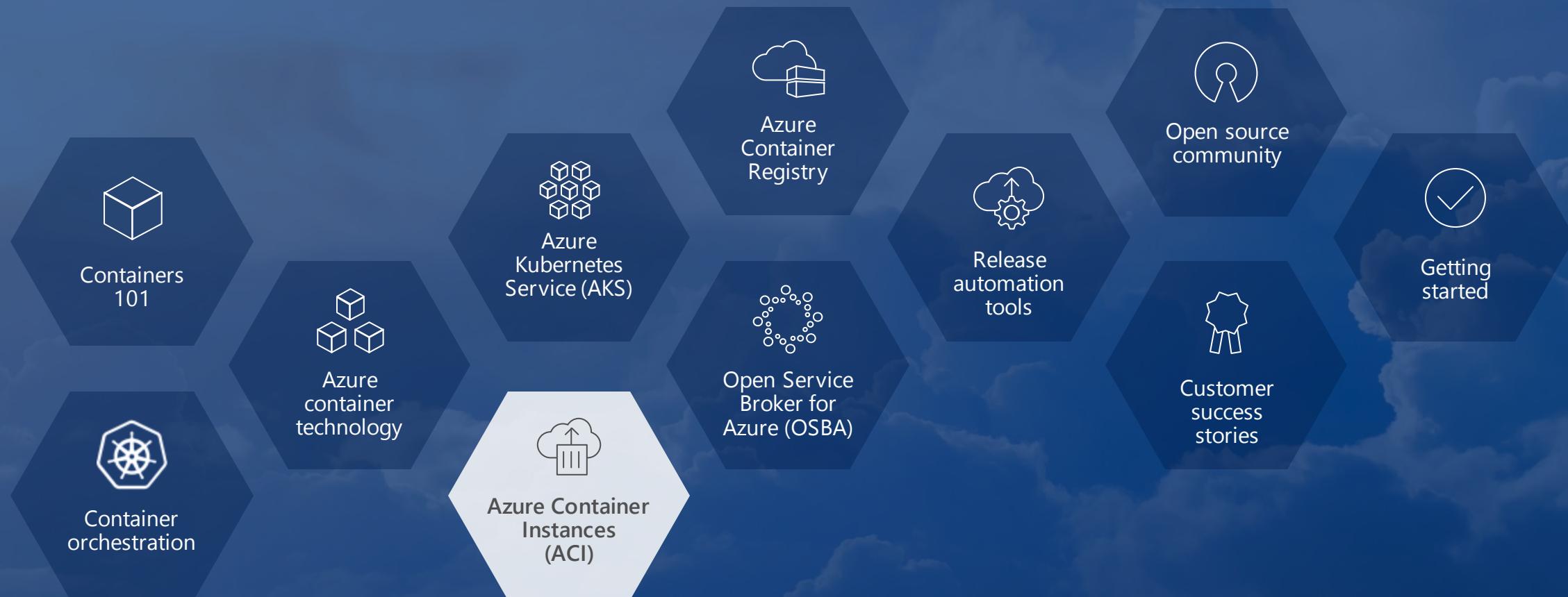
# Deployment Events



# Demo

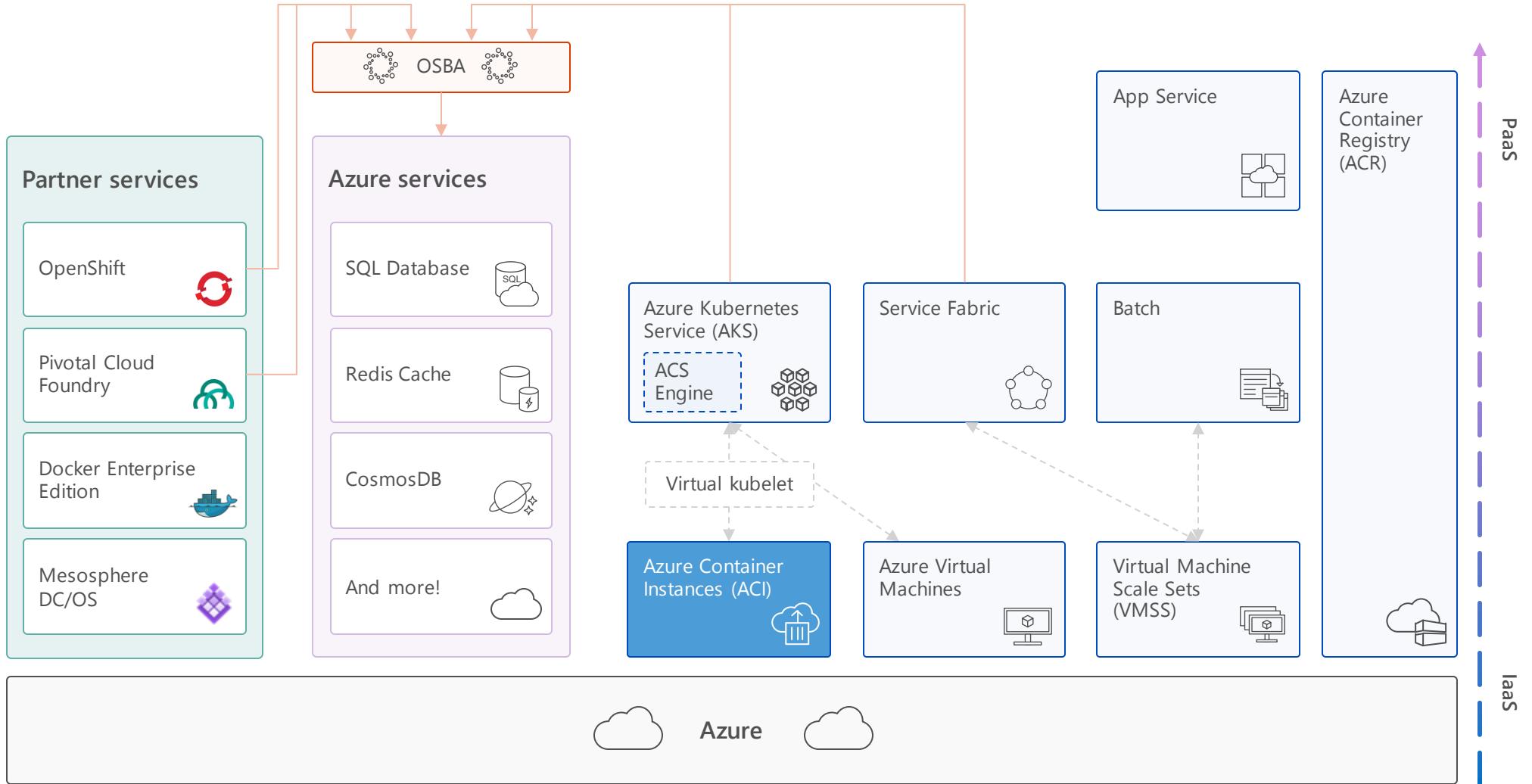
- Create Pod
- Create Deployment
- Add Service to Deployment
- Rolling Update of Deployment

# Azure Container Instances (ACI)





# Azure Container Instances (ACI)





Azure Kubernetes  
Service (AKS)



Azure Container  
Instances (ACI)



Azure Container  
Registry



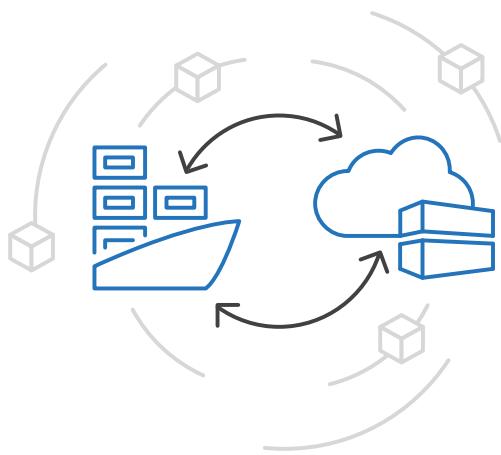
Open Service  
Broker API (OSBA)



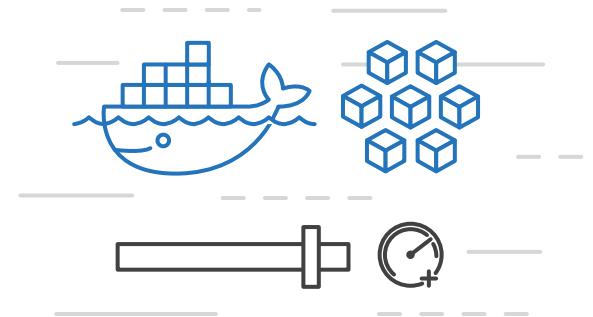
Release  
Automation Tools

# Azure Container Instances (ACI)

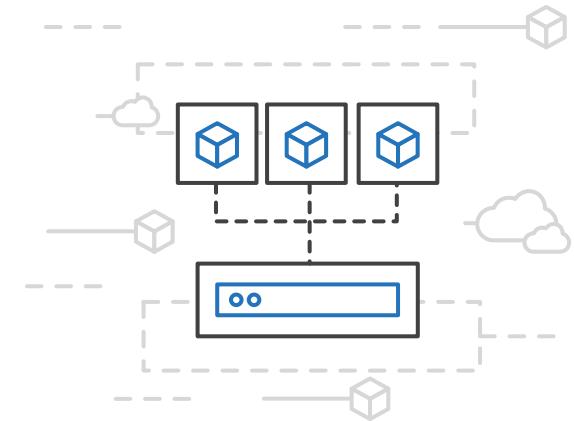
Easily run containers on Azure with a single command



Start using  
containers right away



Cloud-scale  
container capacity



Hyper-vision  
isolation



Azure Kubernetes  
Service (AKS)



Azure Container  
Instances (ACI)



Azure Container  
Registry



Open Service  
Broker API (OSBA)



Release  
Automation Tools

# Azure Container Instances (ACI)

Get started easily

```
$ az container create --name mycontainer --image microsoft/aci-helloworld --resource-group myResourceGroup --ip-address public
```

```
  "ipAddress": {  
    "ip": "52.168.86.133",  
    "ports": [...]  
  },  
  "location": "eastus",  
  "name": "mycontainer",  
  "osType": "Linux",  
  "provisioningState": "Succeeded",
```

```
$ curl 52.168.86.133
```

```
<html>  
<head>  
  <title>Welcome to Azure Container Instances!</title>  
</head>
```



Azure Kubernetes Service (AKS)



Azure Container Instances (ACI)



Azure Container Registry



Open Service Broker API (OSBA)



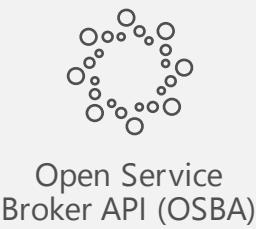
Release Automation Tools

The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with links for Chrome, File, Edit, View, History, Bookmarks, People, Window, Help, and a Microsoft Azure logo. Below the navigation bar is a search bar with the placeholder "Search resources, services, and docs". On the right side of the header, there are icons for notifications, help, and a user profile with the email gamonroy@microsoft.com.

The main content area is titled "New" and features a "Search the Marketplace" input field. To the left of the search field is a sidebar with a "Create a resource" button and a "All services" section containing various service icons and names. A "Containers" category is highlighted with a blue selection bar.

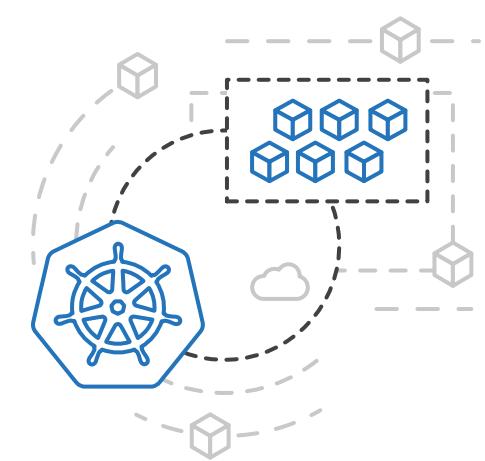
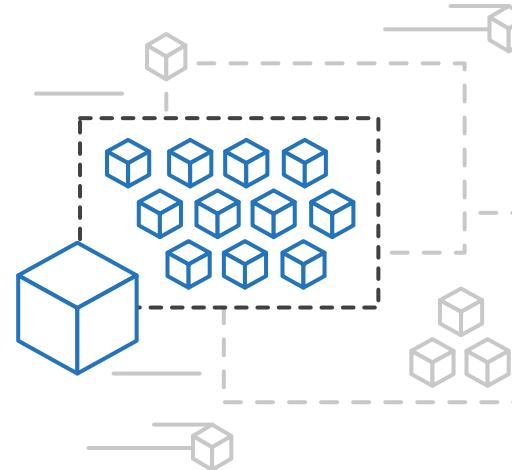
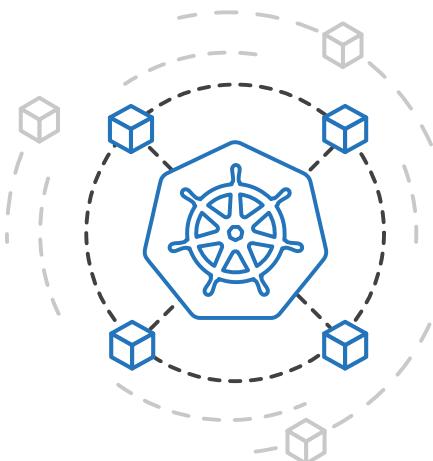
The main content area displays a grid of service offerings under the "Featured" heading. Each item includes a small icon, the service name, and a "Quickstart tutorial" link. The items listed are:

- Kubernetes Service
- Container Service
- Container Instances
- Container Registry
- Service Fabric Cluster
- Web App for Containers
- Batch Service
- DC/OS on Azure
- RancherOS
- Docker on Ubuntu Server



# Azure Container Instances (ACI)

## ACI Connector for Kubernetes





Azure Container Instances (ACI)



Azure Container Registry



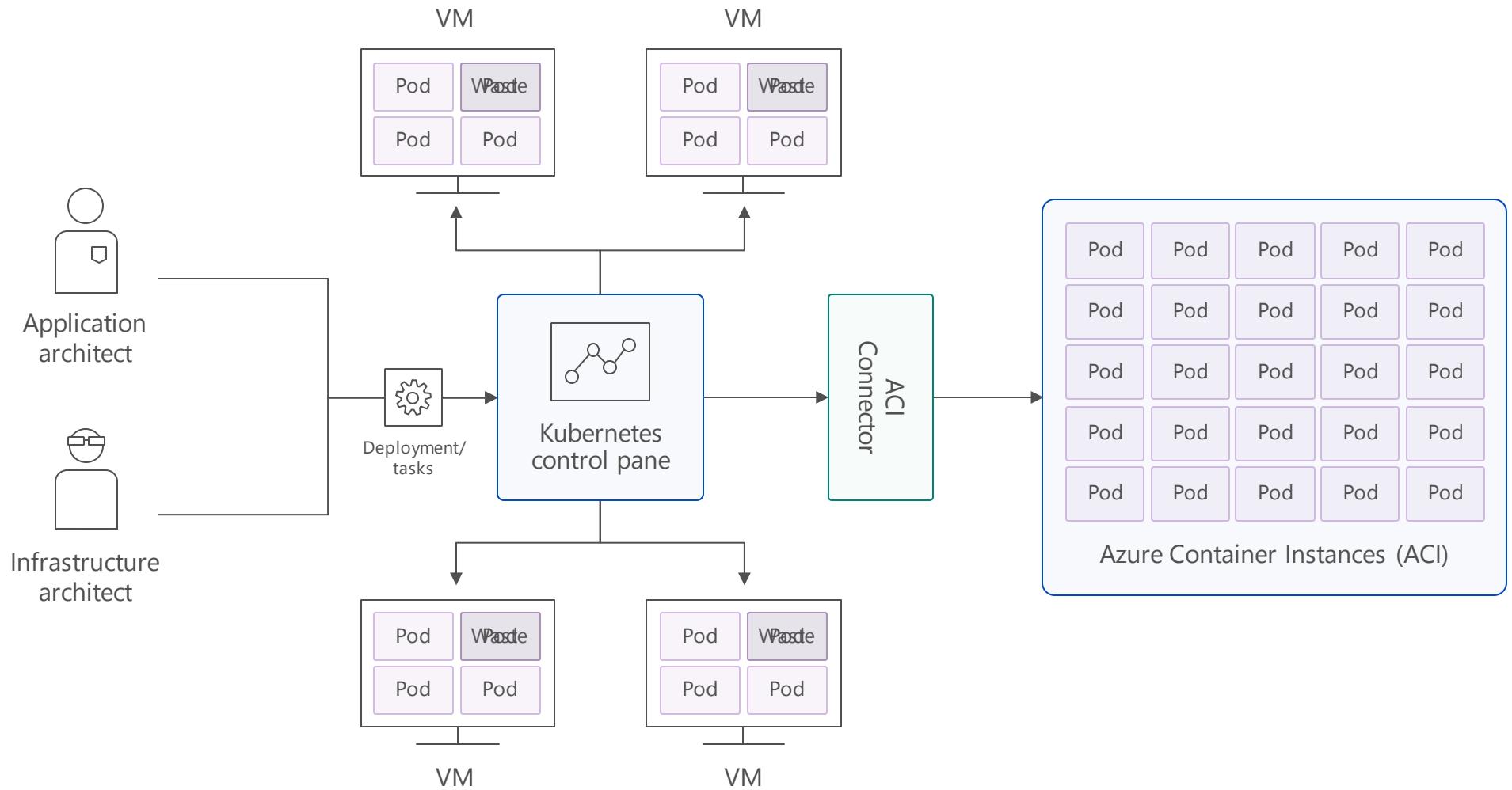
Open Service Broker API (OSBA)



Release Automation Tools

# Azure Container Instances (ACI)

## Bursting with the ACI Connector





Azure Kubernetes Service (AKS)



Azure Container Instances (ACI)



Azure Container Registry



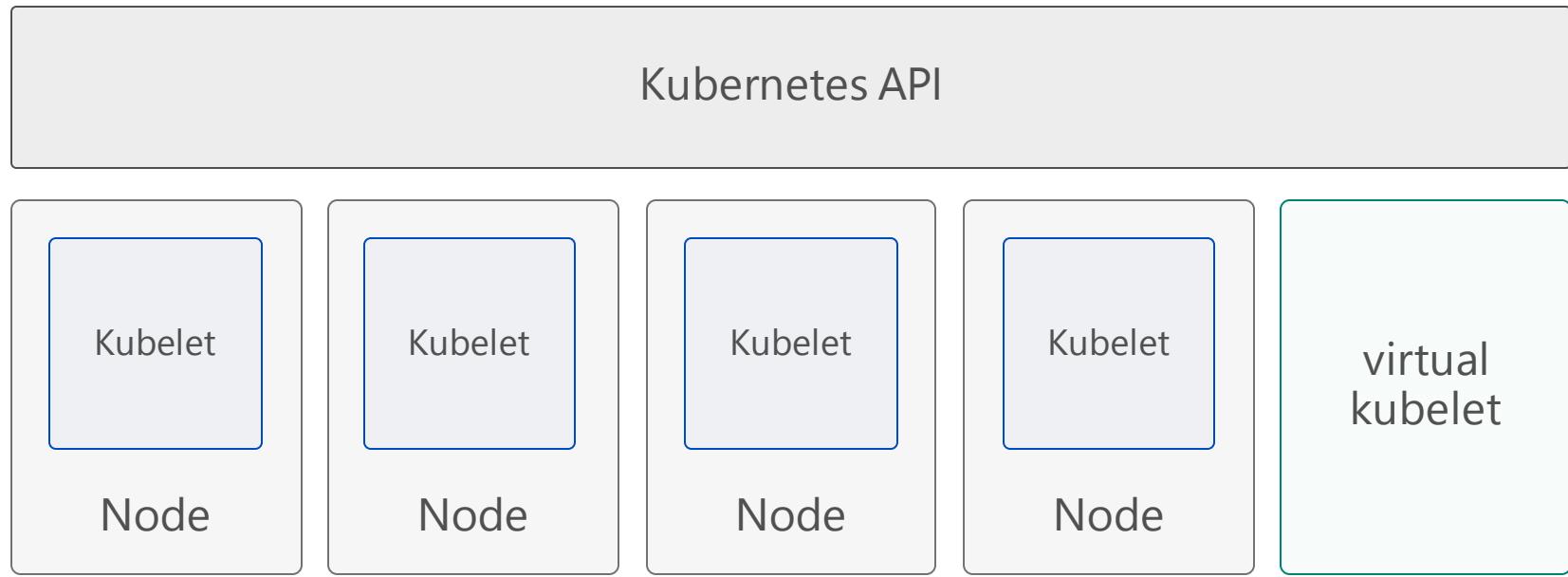
Open Service Broker API (OSBA)



Release Automation Tools

# Azure Container Instances (ACI)

## Virtual Kubelet



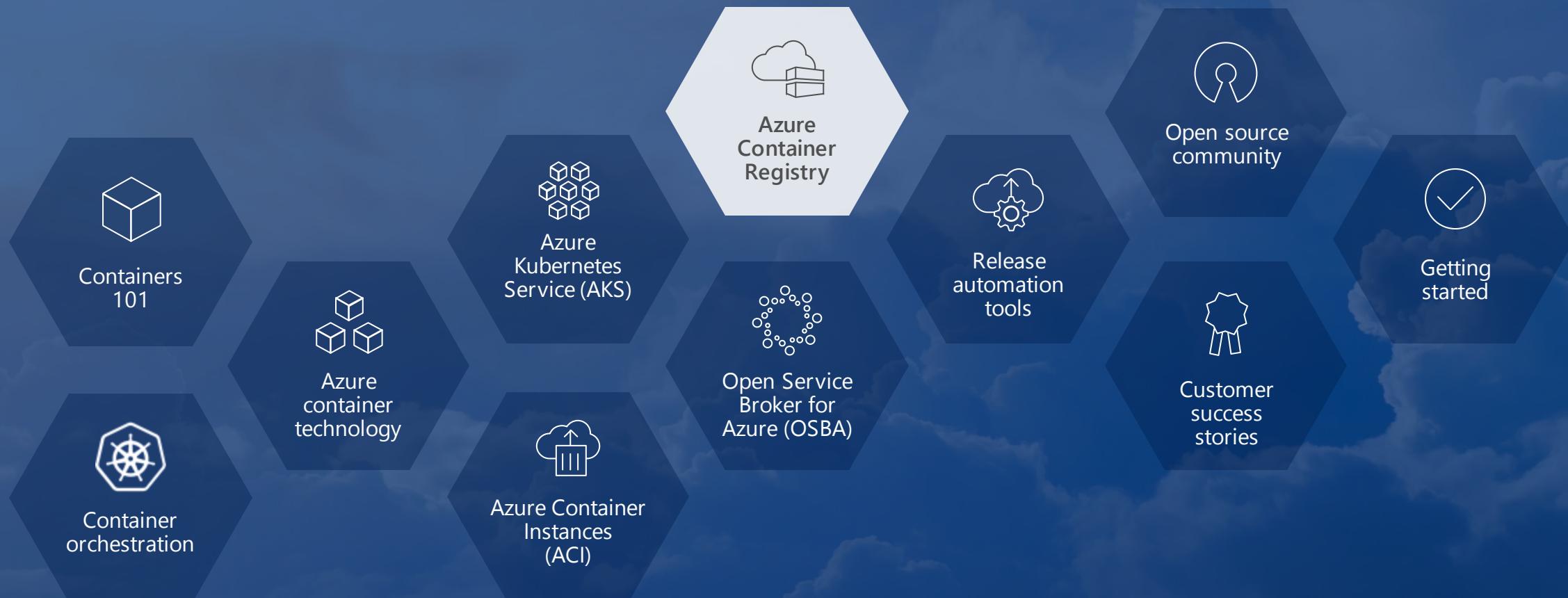
Typical kubelets implement the pod and container operations for each node as usual.

Virtual kubelet registers itself as a “node” and allows developers to program their own behaviors for operations on pods and containers.

# Demo

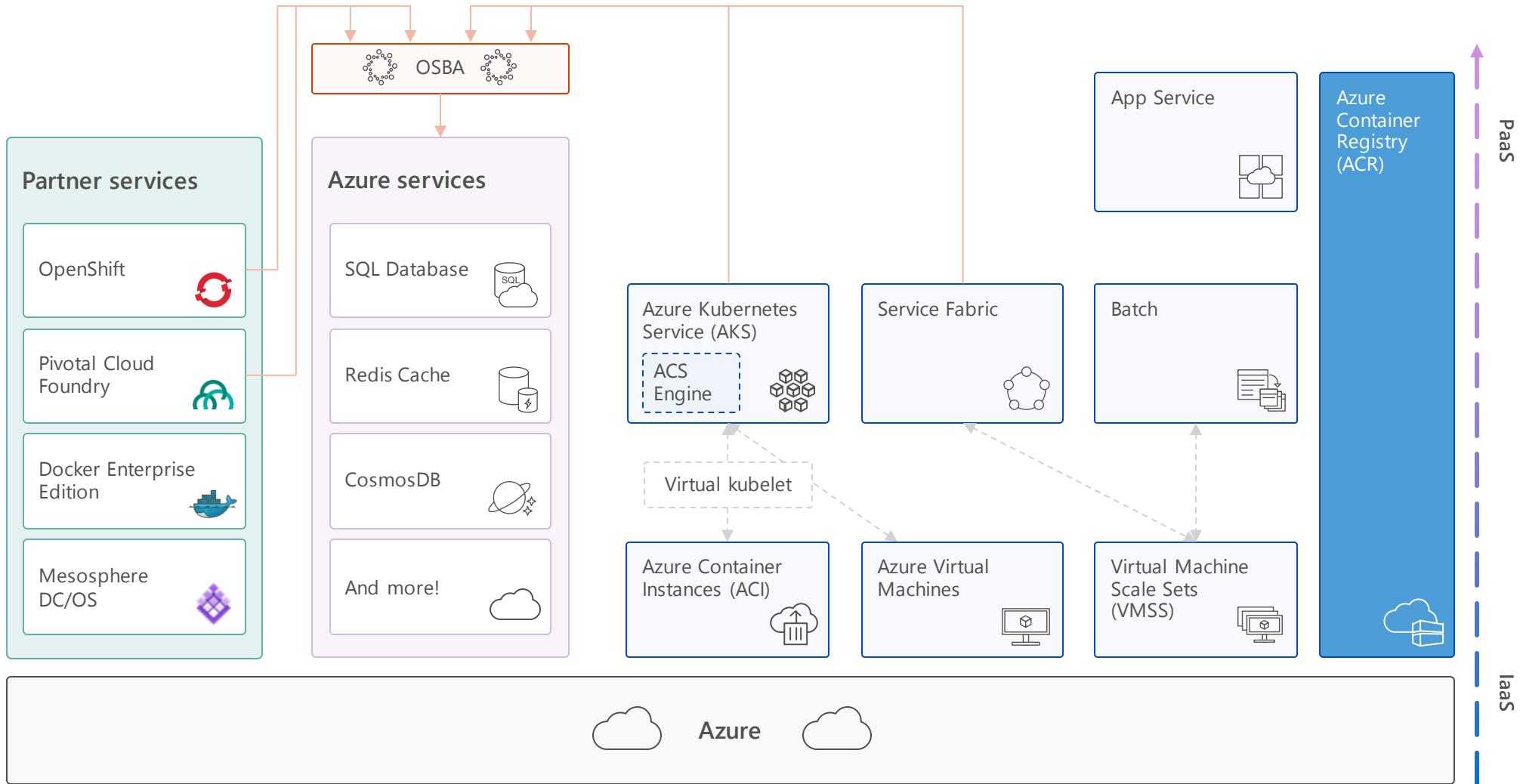
- ACI Deployment

# Azure Container Registry





# Azure Container Registry





Azure Kubernetes  
Service (AKS)



Azure Container  
Instances (ACI)



Azure Container  
Registry



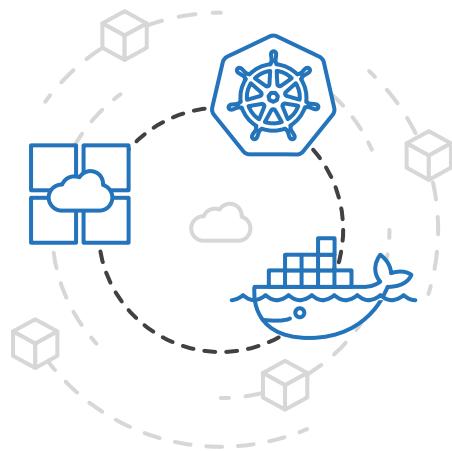
Open Service  
Broker API (OSBA)



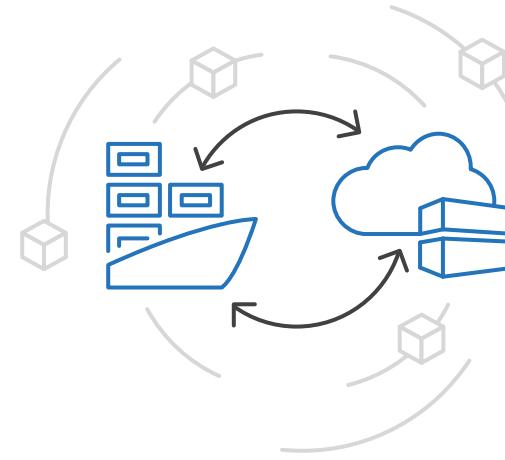
Release  
Automation Tools

# Azure Container Registry

Manage a Docker private registry as a first-class Azure resource



Manage images for all  
types of containers



Use familiar, open-  
source Docker CLI tools



Azure Container Registry  
geo-replication



Azure Kubernetes Service (AKS)



Azure Container Instances (ACI)



Azure Container Registry



Open Service Broker API (OSBA)



Release Automation Tools

Chrome File Edit View History Bookmarks People Window Help

Secure | https://ms.portal.azure.com/#@microsoft.onmicrosoft.com/resource/subscriptions/4f48eeae-9347-40c5-897b-46af1b8811ec/resourceGroups/acr/providers/Microsoft.ContainerRegistry/registries/gabrtv/...

Microsoft Azure

Report a bug Search resources, services, and docs

gamonroy@microsoft... MICROSOFT

Create a resource

All services

FAVORITES

Dashboard

All resources

Resource groups

App Services

SQL databases

SQL data warehouses

Azure Cosmos DB

Virtual machines

Load balancers

Storage accounts

Virtual networks

Azure Active Directory

Monitor

Advisor

Security Center

Cost Management + Billing

Help + support

Container services

Kubernetes services

Home > Resource groups > acr > gabrtv

gabrtv Container registry

Overview Activity log Access control (IAM) Tags Quick start

Move Delete Update

Essentials

Resource group acr

Location East US

Subscription name Gabe Internal

Subscription ID 4f48eeae-9347-40c5-897b-46af1b8811ec

Login server gabrtv.azurecr.io

Creation date 11/15/2017, 3:26 AM PST

SKU Premium

Provisioning state Succeeded

Registry metrics

Registry quota usages

USED 1.6 GiB

REMAINING 498.4 GiB

500GiB SIZE QUOTA

Container security integrations

Aqua Security

Aqua provides development-to-production lifecycle controls for securing containerized applications.

Azure Marketplace

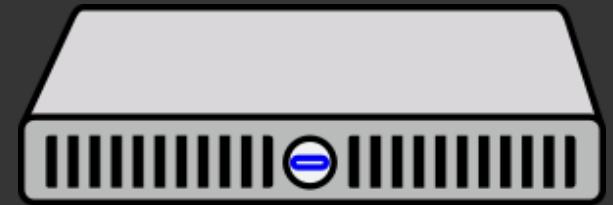
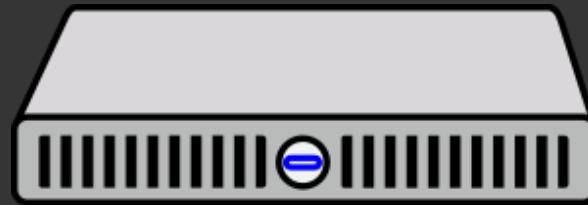
Twistlock

Providing vulnerability management and runtime protection across your environments.

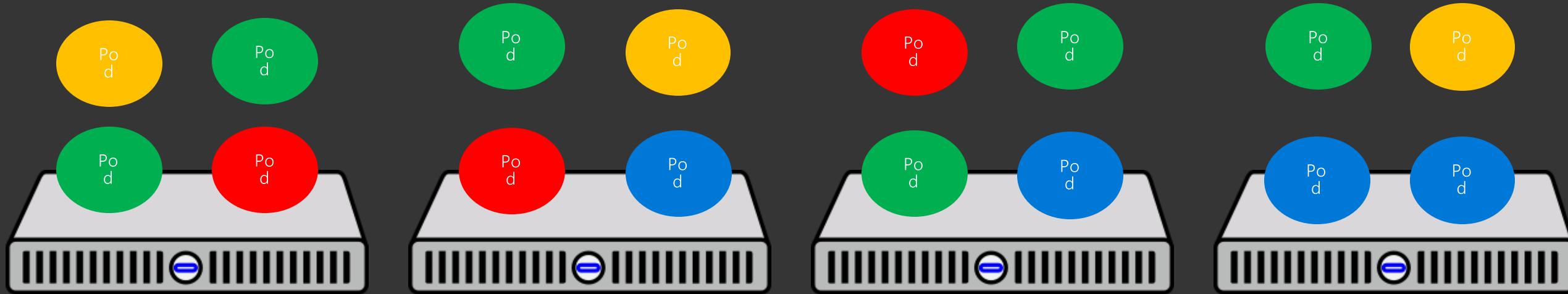
Azure Marketplace

# Monitoring

# Current monitoring is static in nature



# Need to monitor for dynamic environments



# Application Health and Readiness

- Kubelets on workers use liveness and readiness probes to know when to restart a container or start directing traffic
  - Liveness check: when to restart
  - Readiness check: when to forward Service traffic to a pod

# Monitoring: What indicators matter in Kubernetes

- Cluster health: total number of nodes, pods, etc
- Node health: available compute resources, health
- Application health
- Tooling Examples
  - Datadog
  - Prometheus
  - InfluxData
  - Cloud-specific: OMS (Azure), CloudWatch (AWS)

# Logging

- Pod logs: applications must log to standard out!
- Cluster-wide event logs
- Logging forwarder solutions
  - fluentd
  - logstash
- Log indexers
  - OMS Log Analytics
  - Splunk
  - ELK stack

# Scaling Applications

# Application Scaling Strategies

## Infrastructure Level

Increase/decrease the number of VM's running in the cluster

Azure infrastructure function

via AKS API/CLI (utilizes Azure Availability Sets)

Auto-scaling coming to AKS soon

## Application Level

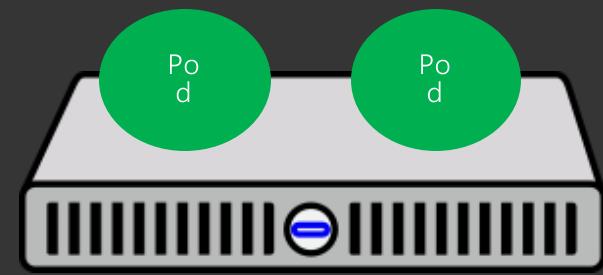
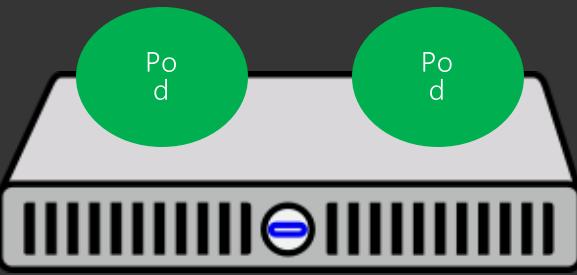
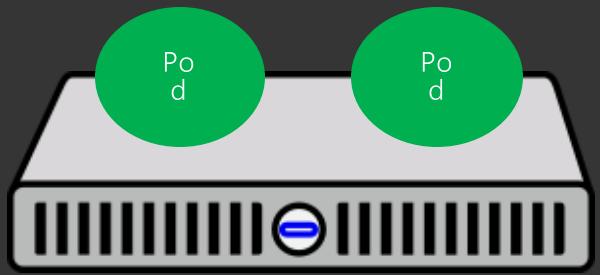
Increase/decrease the number of pods for a given service

Handled by the orchestrator or the application itself

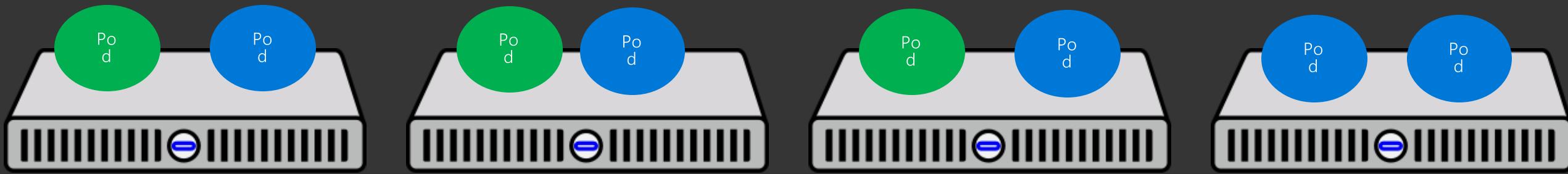
Kubernetes Horizontal Pod Autoscaling

These functions should be coordinated

# Scaling Pods



# Scaling Host



# Scaling the ACS Cluster

Azure CLI

 Copy

 Try It

```
az aks scale --name myK8sCluster --resource-group myResourceGroup --agent-count 1
```

# Demo

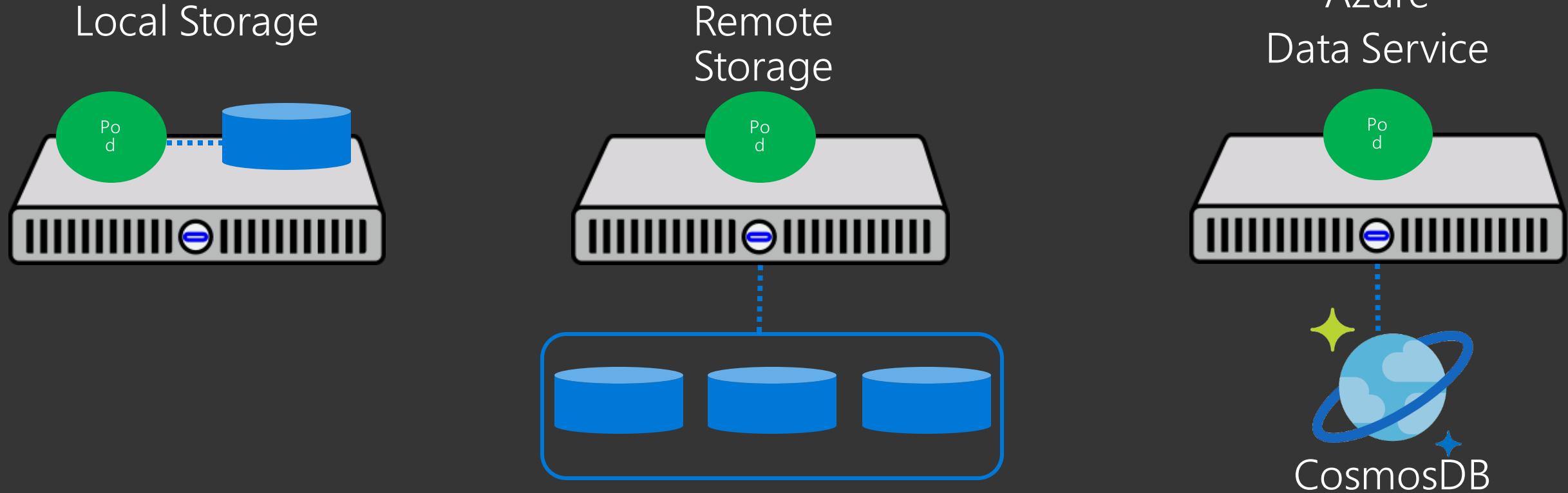
- Scale existing Deployment

Labs: 6, 7

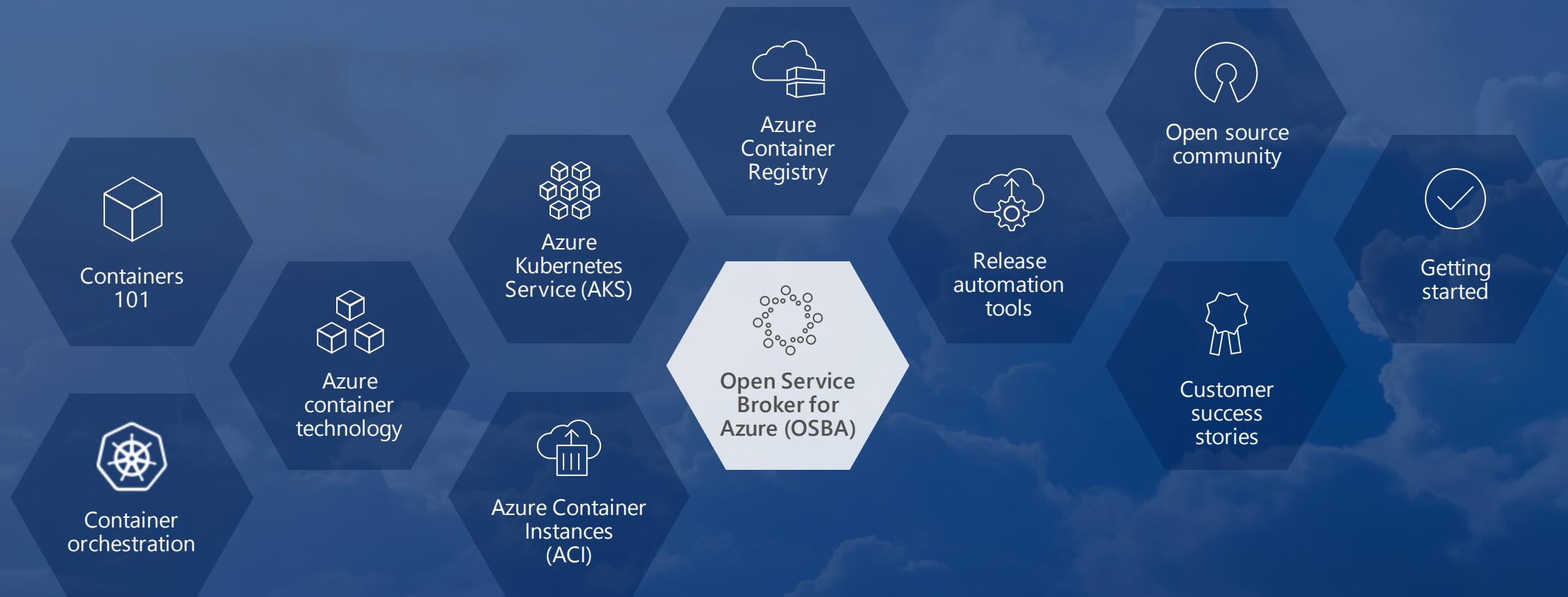
[https://github.com/OSSCanada/aks\\_partnerworkshop](https://github.com/OSSCanada/aks_partnerworkshop)

# Managing Storage and State

# Managing State For Containers

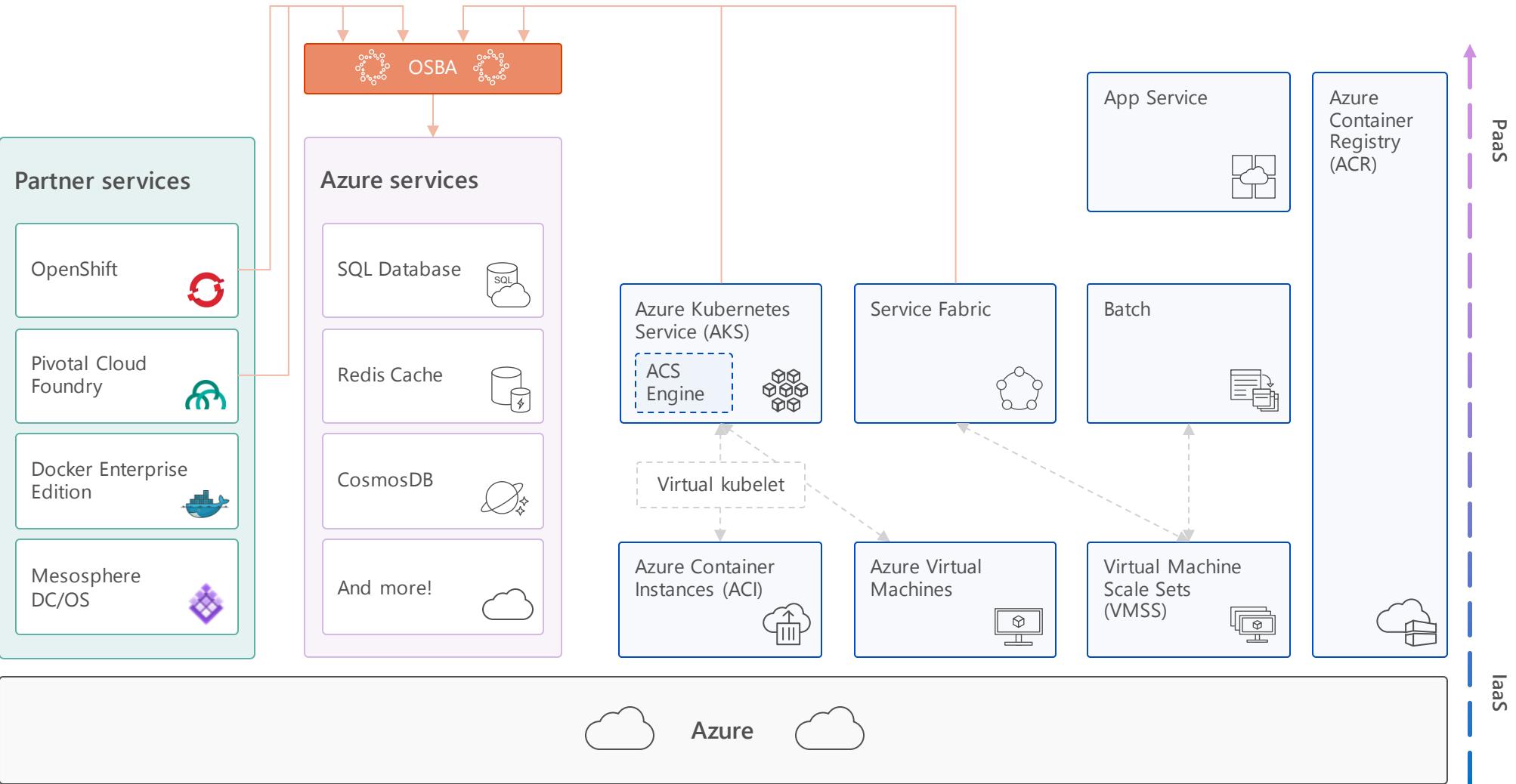


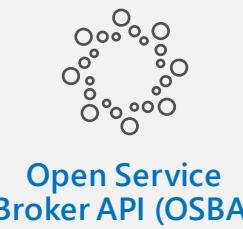
# Open Service Broker for Azure





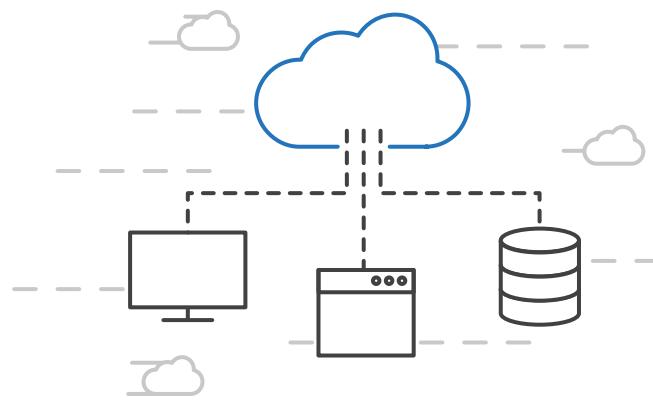
# Open Service Broker for Azure (OSBA)





# Open Service Broker for Azure (OSBA)

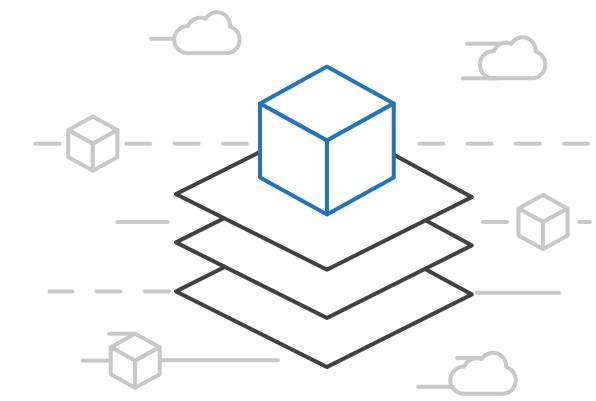
Connecting containers to Azure services and platforms



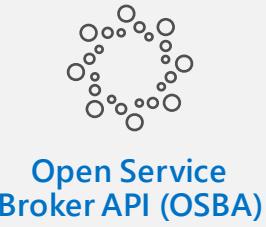
A standardized way to connect with Azure services



Simple and flexible service integration



Compatible across numerous platforms



# Open Service Broker for Azure (OSBA)

An implementation of the Open Service Broker API

Azure SQL Database



Redis Cache



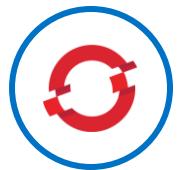
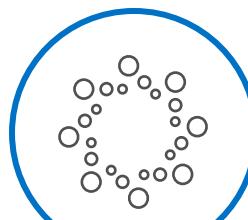
CosmosDB



And more!



Open Service Broker  
for Azure (OSBA)



OpenShift



Cloud Foundry



Service Fabric  
(Coming soon)



Kubernetes  
(AKS)



Azure Kubernetes  
Service (AKS)



Azure Container  
Instances (ACI)



Azure Container  
Registry



Open Service  
Broker API (OSBA)



Release  
Automation Tools

# Open Service Broker for Azure (OSBA)

Getting started with ease

```
$ helm repo add azure Azure/helm-charts
```

```
$ helm install azure/service-broker
```

```
$ helm install azure/wordpress
```

# Upgrading Kubernetes in AKS

Azure CLI

 Copy

 Try It

```
az aks get-versions --name myK8sCluster --resource-group myResourceGroup --output table
```

Azure CLI

 Copy

 Try It

```
az aks upgrade --name myK8sCluster --resource-group myResourceGroup --kubernetes-version 1.8.1
```

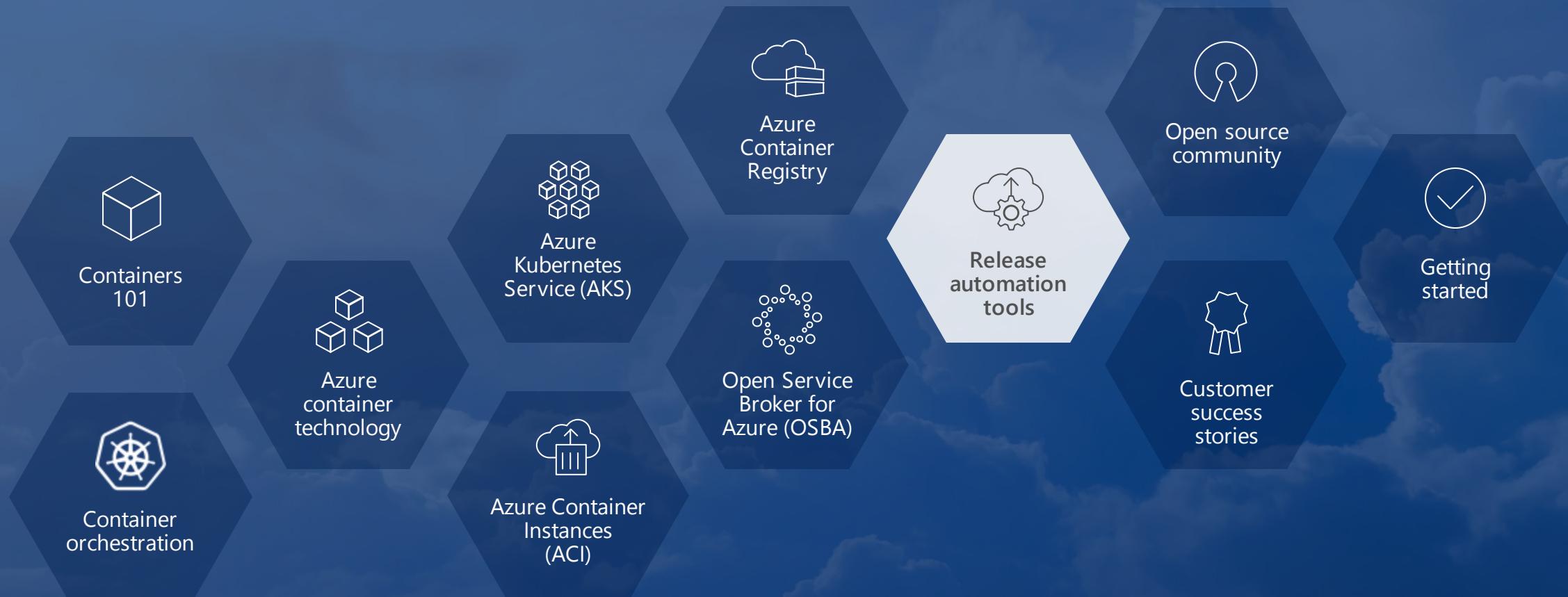
# Demo

- Container Persistence with Azure Disk
- Version Upgrade Options

Labs: 8, 9, 10

[https://github.com/OSSCanada/aks\\_partnerworkshop](https://github.com/OSSCanada/aks_partnerworkshop)

# Release automation tools





Azure Kubernetes  
Service (AKS)



Azure Container  
Instances (ACI)



Azure Container  
Registry



Open Service  
Broker API (OSBA)



Release  
Automation Tools

# Release automation tools

Simplifying the Kubernetes experience



Streamlined  
Kubernetes  
development



The package  
manager for  
Kubernetes



Event-driven  
scripting for  
Kubernetes



Visualization  
dashboard for  
Brigade





Azure Kubernetes  
Service (AKS)



Azure Container  
Instances (ACI)



Azure Container  
Registry



Open Service  
Broker API (OSBA)



Release  
Automation Tools

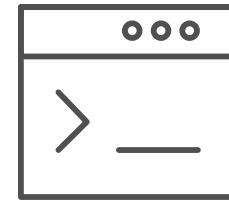
# Draft

Simple app development and deployment – into any Kubernetes cluster



## Simplified development

Using two simple commands, developers can now begin hacking on container-based applications without requiring Docker or even installing Kubernetes themselves



## Language support

Draft detects which language your app is written in, and then uses packs to generate a Dockerfile and Helm Chart with the best practices for that language



Azure Kubernetes  
Service (AKS)



Azure Container  
Instances (ACI)



Azure Container  
Registry



Open Service  
Broker API (OSBA)



Release  
Automation Tools

# Helm

The best way to find, share, and use software  
built for Kubernetes



## Manage complexity

Charts can describe complex apps; provide repeatable app installs, and serve as a single point of authority

## Easy updates

Take the pain out of updates with in-place upgrades and custom hooks

## Simple sharing

Charts are easy to version, share, and host on public or private servers

## Rollbacks

Use `helm rollout` to roll back to an older version of a release with ease



Azure Kubernetes  
Service (AKS)



Azure Container  
Instances (ACI)



Azure Container  
Registry



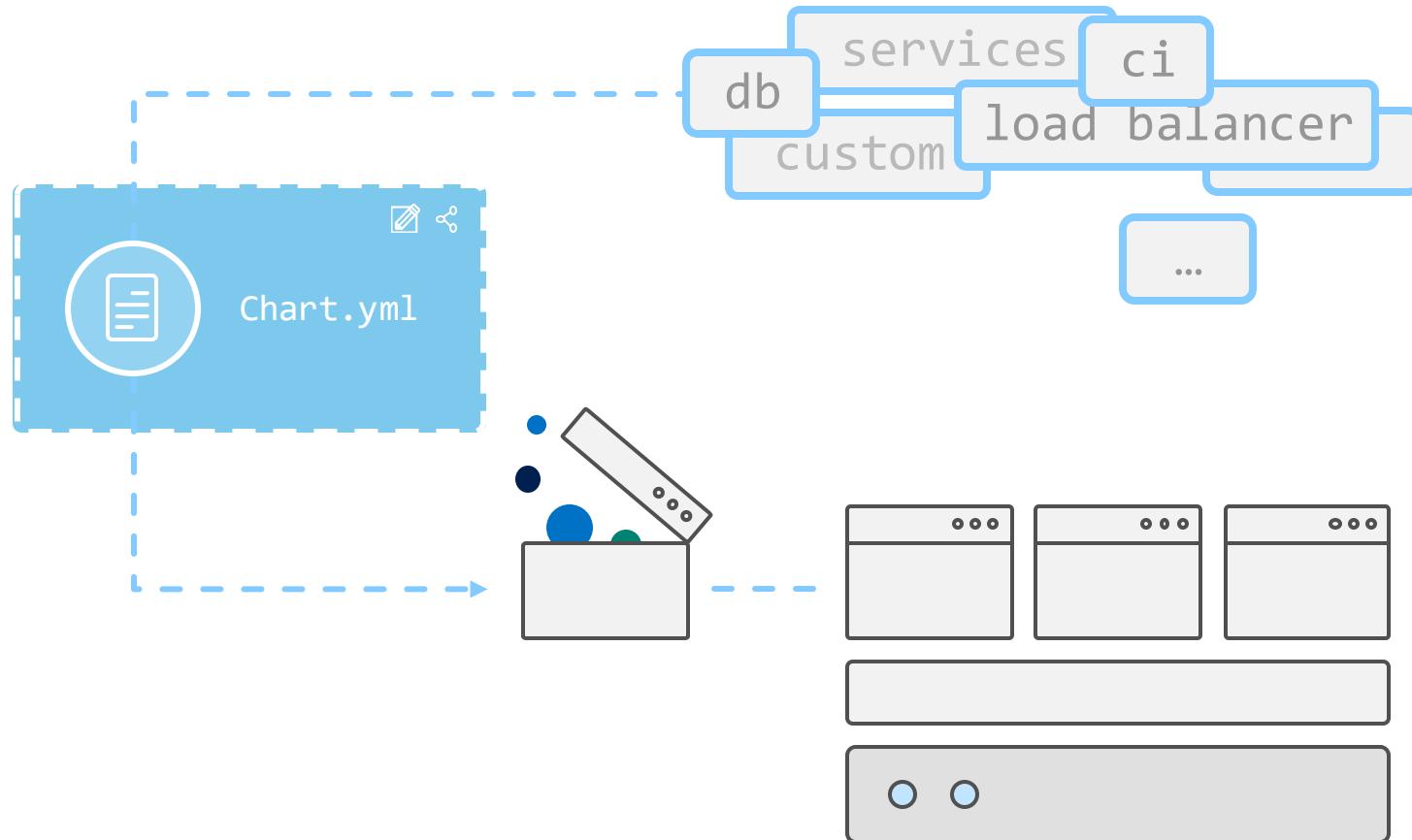
Open Service  
Broker API (OSBA)



Release  
Automation Tools

# Helm

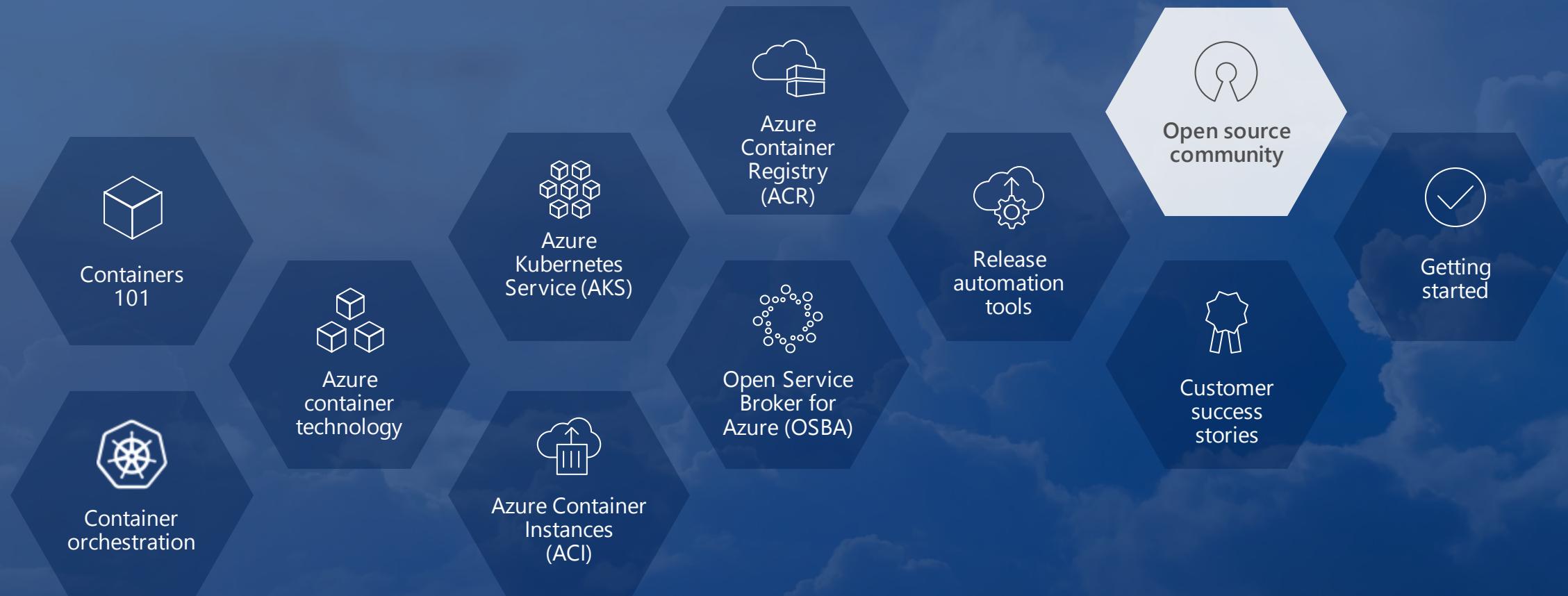
Helm Charts helps you define, install, and upgrade even the most complex Kubernetes application



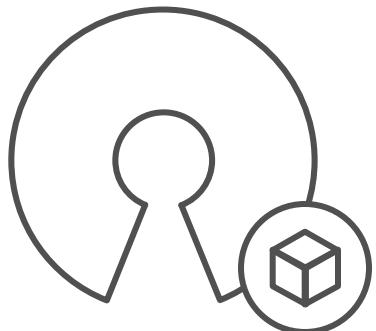
# Demo

- Install via Helm
- Development using Draft

# Open source community



# Community culture



Open source container  
code contributions



Numerous open source  
project builds



Open source community  
leadership



# Open Source @ Microsoft



## Ross Gardler

Apache Foundation  
Principal Program Manager



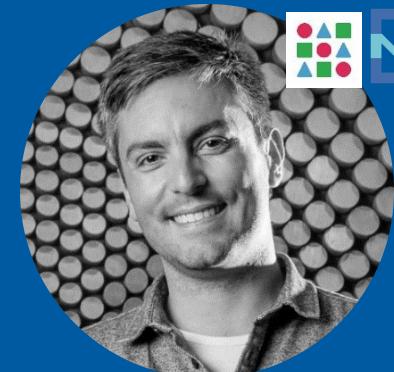
## Jessie Frazelle

Docker / Linux  
Principal Cloud Developer Advocate



## Brendan Burns

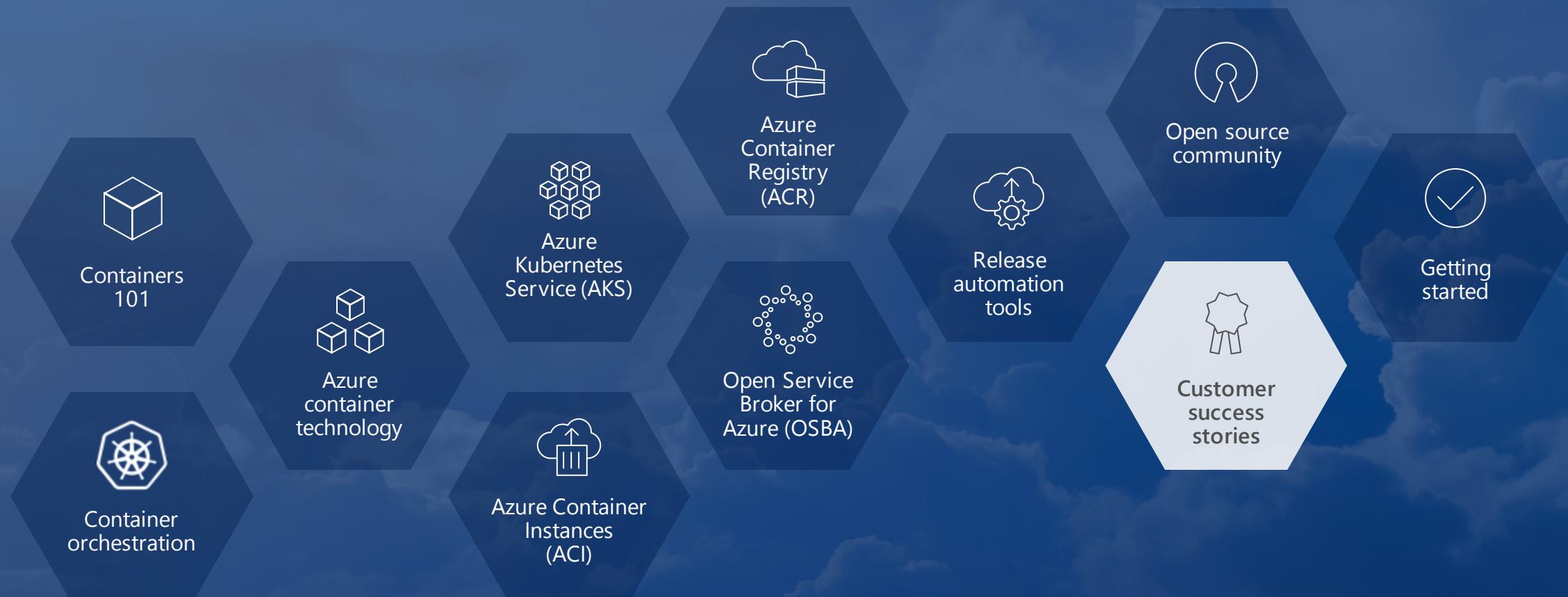
Kubernetes  
Distinguished Engineer



## Gabe Monroy

Deis / CNCF  
Principal Program Manager

# Customer success stories



# Rapidly growing software company attracts customers with seamless cloud demo experience

Headquartered in Freiburg, Germany, Jedox develops enterprise performance management software for businesses around the globe. Jedox chose Microsoft Azure to help it scale and grow. The company needed a more lightweight compute unit than virtual machines to power its website demos and provide a good customer experience. Jedox deployed Azure Container Instances to spin up customer demos on demand, improve provisioning speed, build confidence in the cloud, and lower IT costs.



## Products and Services

Microsoft Azure  
Microsoft Azure Container Instances

## Organization Size

170 employees

## Industry

Professional Services

## Country

Germany



# Altair Engineering democratizes HPC access using the cloud

Altair Engineering makes computer-aided engineering software used to create stronger, better, more innovative products, from bicycles to space shuttles. To make its HyperWorks and Inspire applications available to more engineers, Altair delivers them through its Unlimited platform running in Microsoft Azure. In Azure, these solutions have access to affordable high-performance computing (HPC) resources, which enables low-cost collaboration among large engineering teams and opens up new markets for Altair.



## Products and Services

Microsoft Azure  
Microsoft Azure Container Service  
Microsoft Azure Virtual Machines

## Organization Size

2,000 Employees

## Industry

Manufacturing

## Country

United States





# Energy company electrifies pace of innovation and expansion

An aerial photograph showing a large-scale solar farm. Numerous dark blue solar panels are arranged in long, parallel rows across a field. A narrow path or road cuts through the panels. To the left and right of the solar array, there are patches of dry, brown grass and some green vegetation.

Ambit Energy provides electricity and natural-gas services in deregulated markets around the world. It uses technology as a competitive differentiator, employing microservices, DevOps, and continuous deployment to speed software development. To stand up infrastructure just as quickly, Ambit uses Microsoft Azure services such as Azure Container Service, together with infrastructure as code and open source technologies, to completely automate infrastructure provisioning. By implementing Azure, Ambit can move dramatically faster to enhance its services and enter new markets. Infrastructure redundancy is flexible and worry-free. And costs are 22 percent lower, which helps Ambit compete in the crowded electricity market. Because Ambit's cloud journey is gradual, it appreciates the fact that Azure is a great hybrid-cloud enabler, connecting easily to Ambit datacenters.



## Products and services

Microsoft Azure  
Container Service

## Organization size

1,000 employees

## Industry

Power and utilities

## Country

United States

## Business need

Optimize operational efficiency





Azure



## Siemens Health leverages technology to connect medical devices to the cloud through AKS

Digitization and networking between healthcare providers and software development companies are essential to value-based care. Moving from the development of value-added services into becoming more of a platform provider, it became important for Siemens to adopt a microservices approach to application delivery. To that end, Siemens adopted Azure Container Service (AKS) to run their microservices-based apps. AKS puts Siemens in a position not only to deploy business logic in Docker containers—including the orchestration—but also enables them to use an applicant gateway and API management to manage exposure, control, and to meter the access continuously. With their cloud-based development approach, Siemens has driven newfound product development agility. This project is already having a positive impact within the healthcare industry.

**SIEMENS**

### Products and services

Microsoft Azure  
Container Service

### Organization size

100,000+ employees

### Industry

Healthcare

### Country

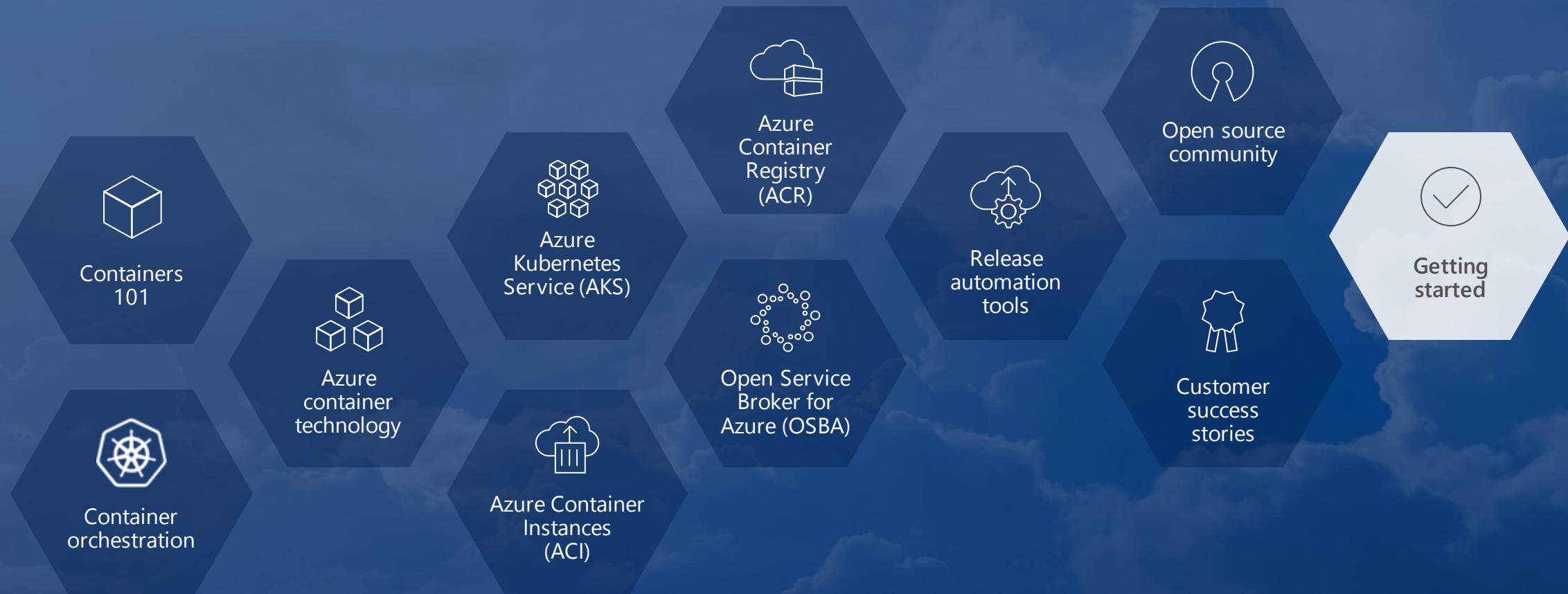
Germany

### Business need

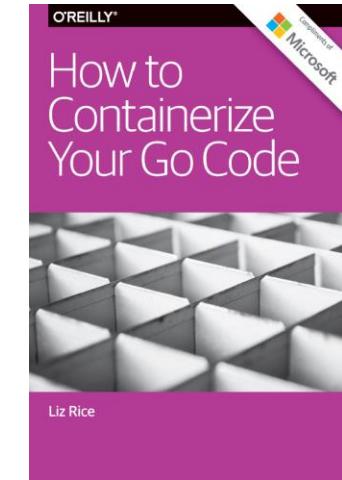
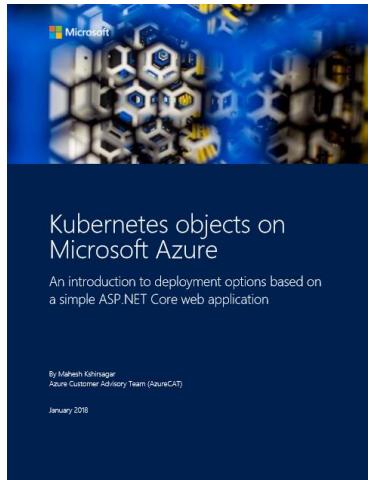
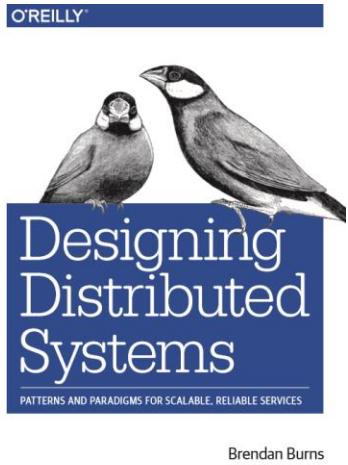
Faster application  
development



# Getting started



# Download **free ebooks**



# Reminder about Tomorrow

- Break into Partner Teams & Bring Your Own App
  - Web/API + Database
- Challenge-Based
  - Challenge-based with Proctor Sign-off
- No Step-by-Step, but Proctor Support
  - Here to help you with your App

# Questions?

