Microsoft

January 30, 2019 — 8:30 AM - 5:00 PM | Detroit, Michigan

Azure Dev Day

Learn, architect, and develop solutions on Azure

#AzureDevDays

for developers, by developers

Learn.

Connect.

Explore.

Microsoft

Containers and Microservices

Michael Meadows, Principal Software Engineer
Insight

Learn.

Connect.

Explore.

About

Michael Meadows

Principal Engineer

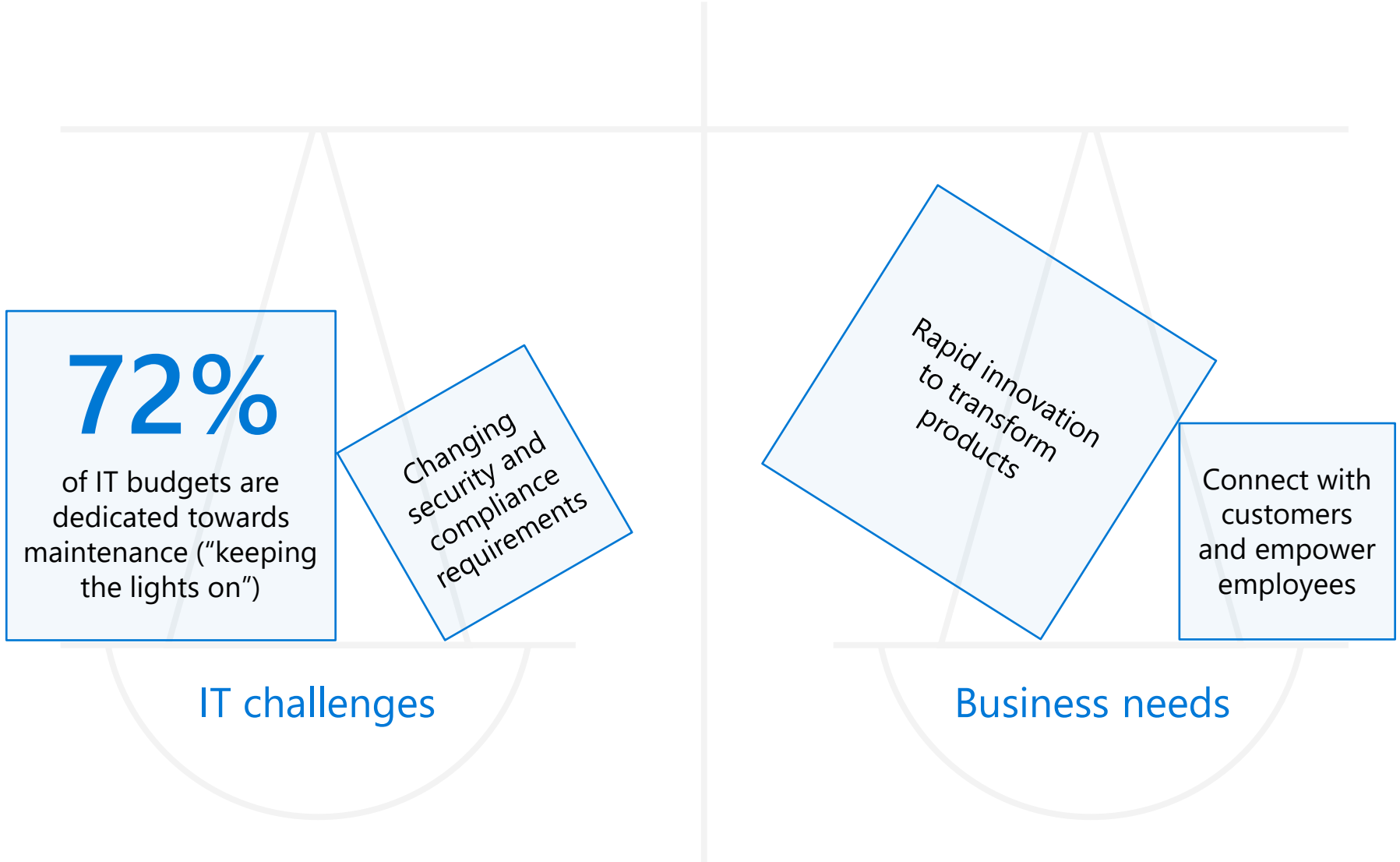
Application and Solution Architecture

Cloud and Dev Ops

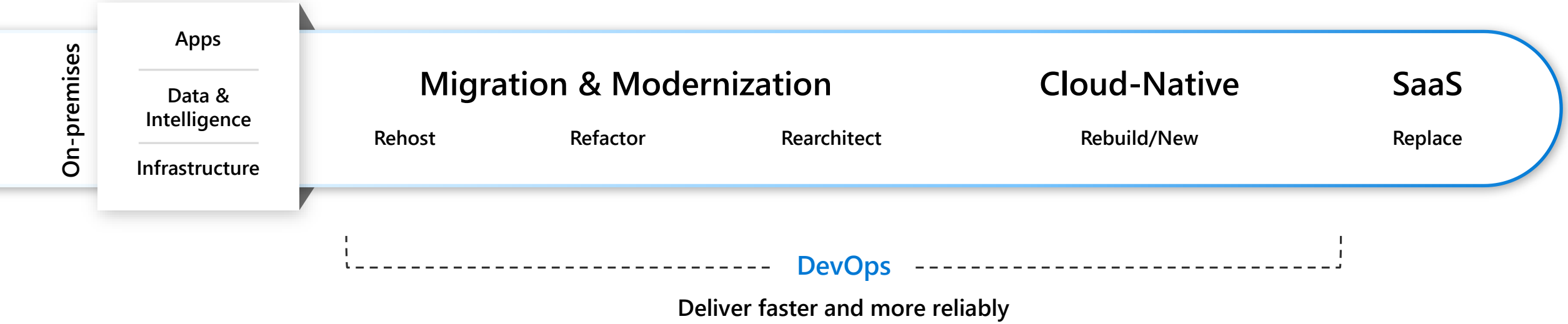
Insight | Digital Innovation

A global technology company committed to helping clients innovate smarter so they can create digital experiences that drive differentiation, competitive advantage, and loyalty.

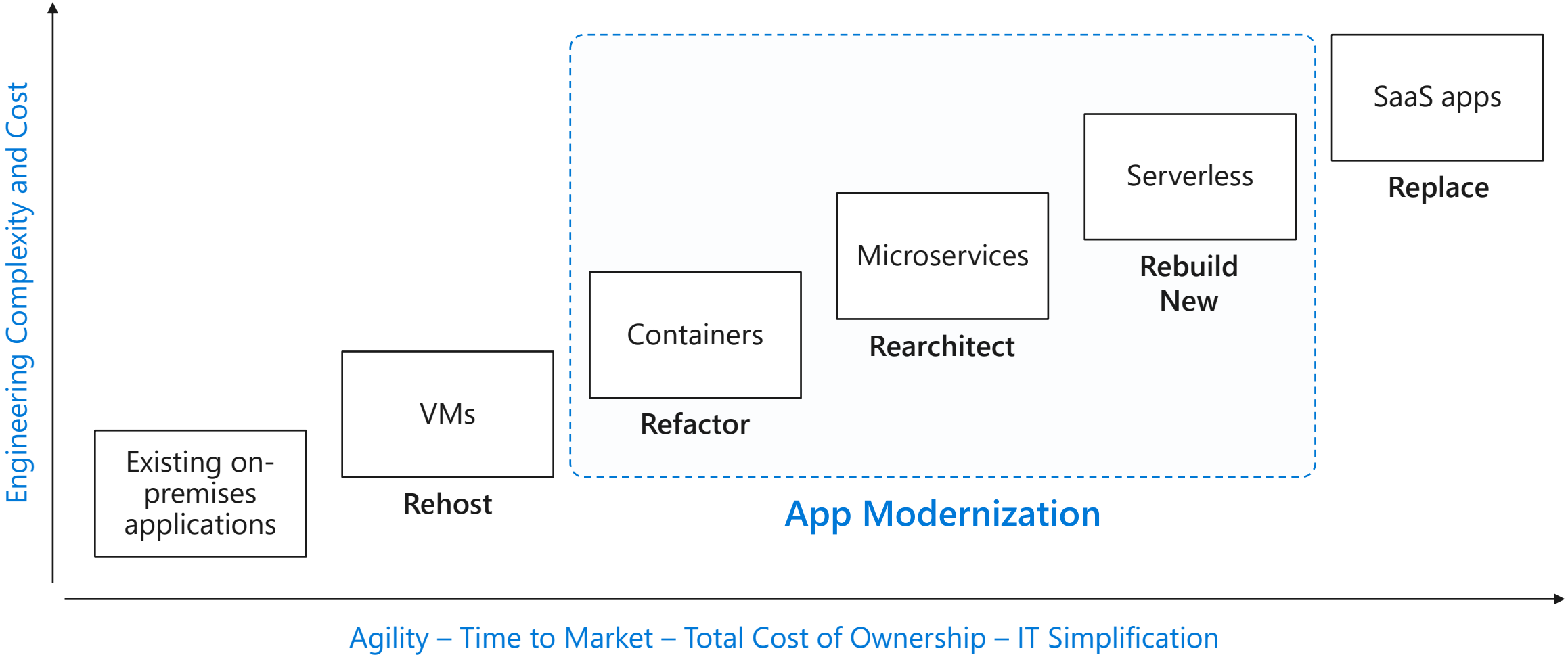
Balancing IT and business



The **journey** to the cloud



Cloud app **continuum**

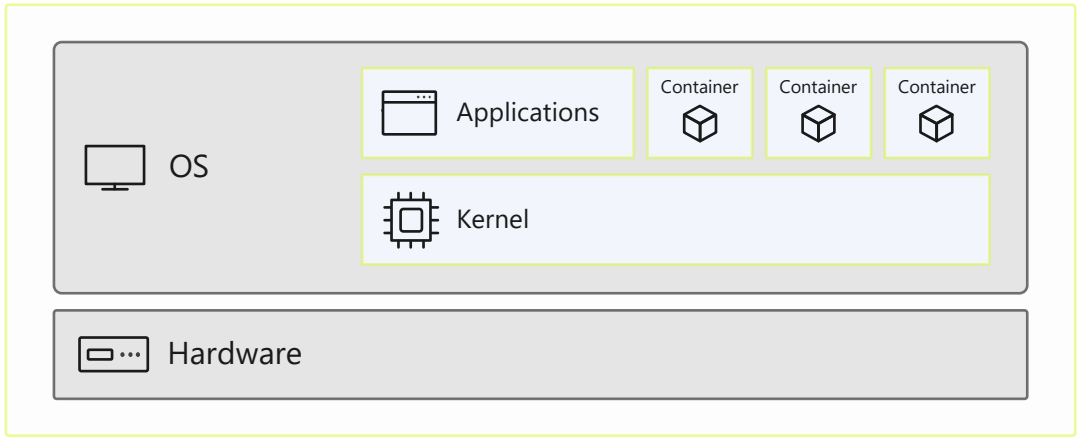


Why it **matters**

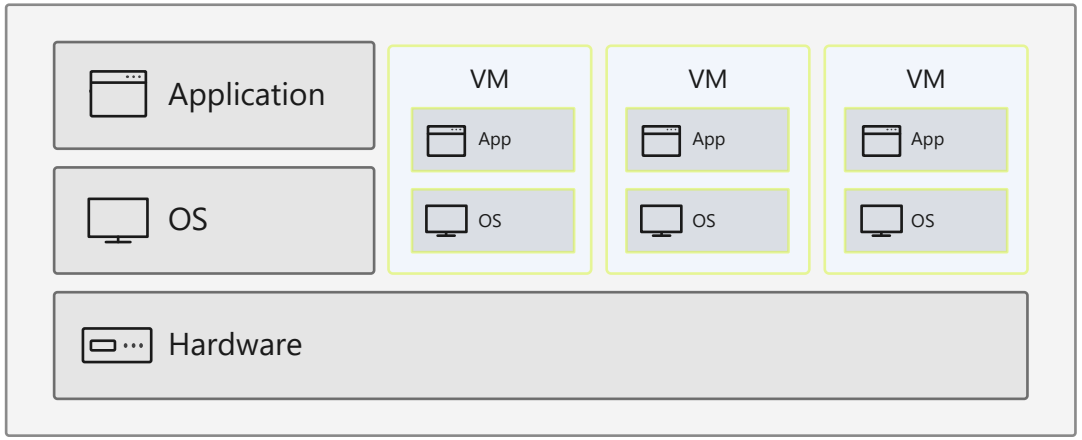
	Traditional	Containers	Microservices	Serverless
Time to Market	Months	Weeks	Days	Hours
Deployment	Big Releases	Frequent Releases	Continuous	Just-In-Time
Feature Development	Big Teams	Small Teams	Distributed Teams	Individual Contributors
Rollout of Features	All-or-Nothing	Variable	Domain Boundary	Feature Boundary
Architectural Complexity	Low	Medium	High	Depends (Probably High)
Resource Utilization Density	Low	Transitional	High	Doesn't Matter
Versioning	Simple	Transitional	Complex	Depends (Probably Complex)
Tooling	Minimal	High	Very High	Depends on Platform

What is a container?

Containers = operating system virtualization



Traditional virtual machines = hardware virtualization

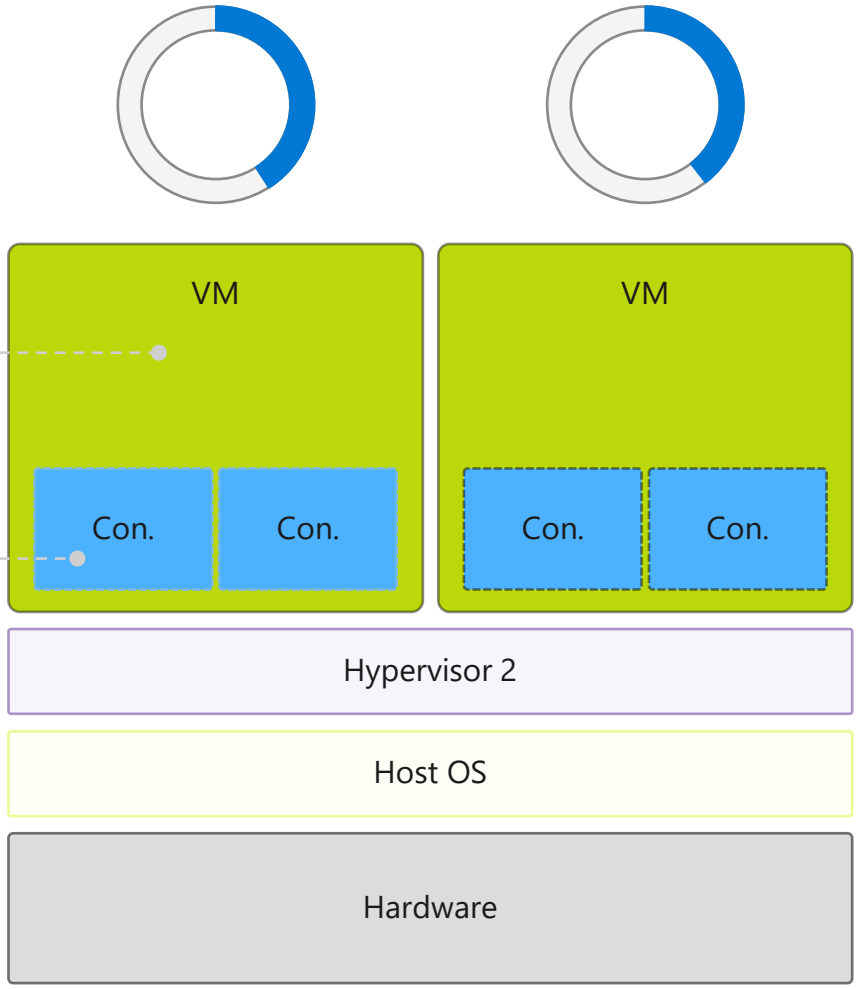


The container advantage

Traditional virtualized environment

Low utilization of container resources

Containerization of applications and their dependencies

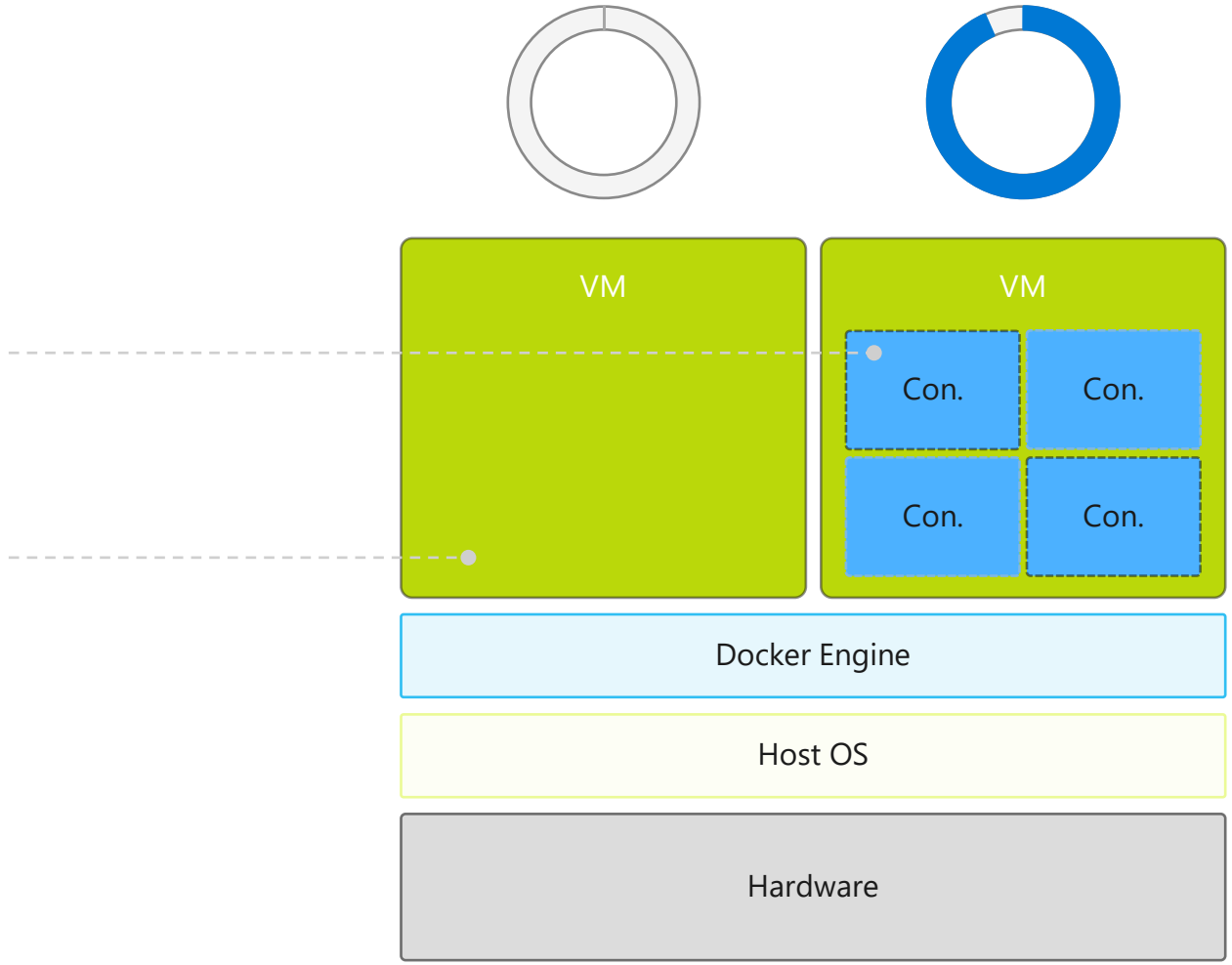


The container advantage

Containerized environment

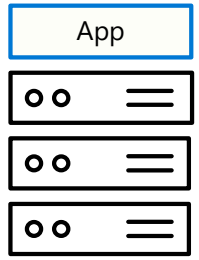
Migrate containers and their dependencies to underutilized VMs for improved density and isolation

Decommission unused resources for efficiency gains and cost savings

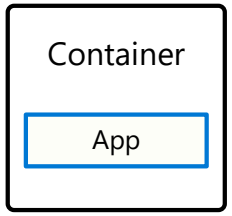


How can containers help your app modernization journey?

From **traditional systems** to a **portfolio of modern apps**



Existing/New Applications



Lift and shift to containers

Each host often runs **8** containers at a time[1]

Microservices

Seconds to deploy, churns **12x** faster[1]

Continuous innovation with DevOps

46x more frequent for code delivery[2]

1: Datadog Report: 8 Surprising Facts About Real Docker Adoption; 2: 2017 state of DevOps Report

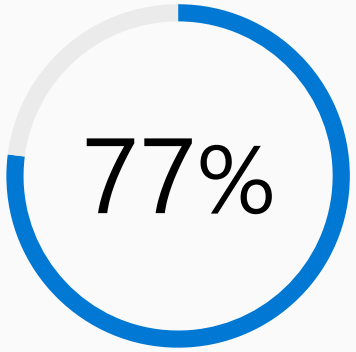
Containers **momentum**

"By 2020, more than **50%** of enterprises will run **mission-critical, containerized cloud-native applications** in production."

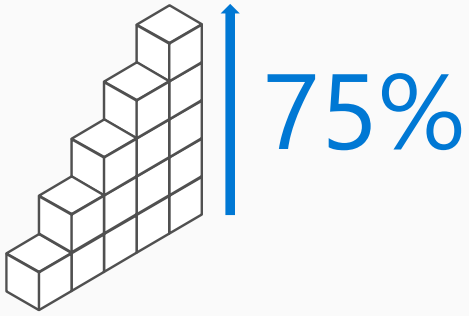
Gartner

Half of container environment is orchestrated.[1]

77% of companies[2] who use container orchestrators choose Kubernetes.

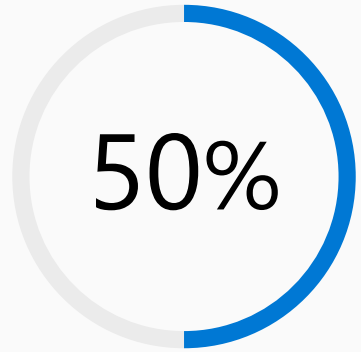


The average size of a container deployment has grown 75% in one year. [1]



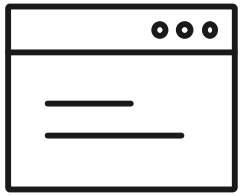
Larger companies are leading the adoption.[1]

Nearly **50%** of organizations[1] running 1000 or more hosts have adopted containers.



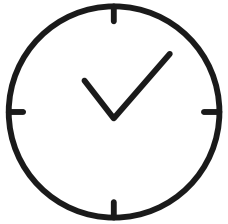
[1] Datadog report: 8 Surprising Facts About Real Docker Adoption
[2] CNCF survey: cloud-native-technologies-scaling-production-applications

What we hear from developers



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

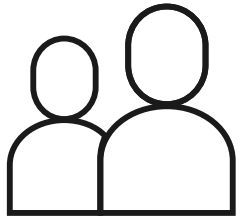
New applications run smoothly on my machine but malfunction on traditional IT servers



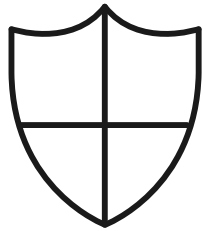
My productivity and application innovation become suspended when I have to wait on IT

What we hear from IT

I need to manage servers and maintain compliance with little disruption

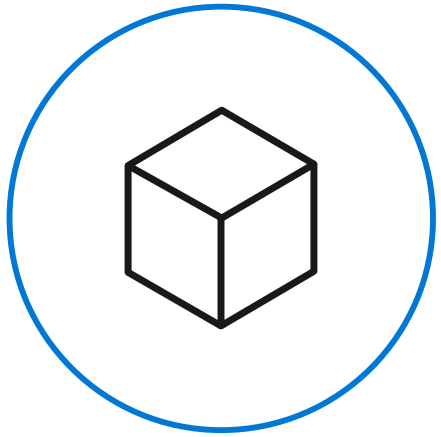


I'm unsure of how to integrate unfamiliar applications, and I require help from developers



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

The container win-win



Fast iteration

Agile delivery

Immutability

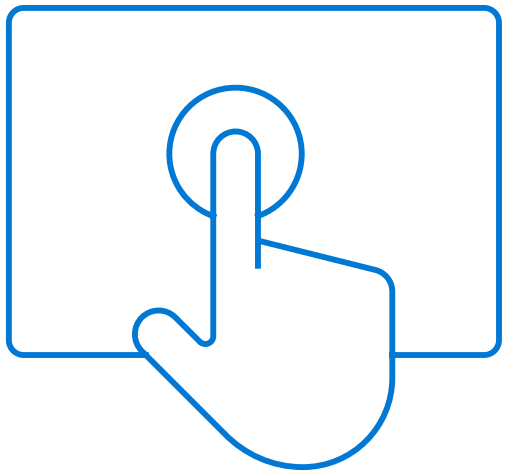
Cost savings

Efficient deployment

Elastic bursting

For developers

For IT



Demo

Getting started with containers

What are **microservices**?

A Software Architectural Style

Applications are composed of small, independent modules that communicate with each other using well-defined APIs. Not platform specific.

Decoupled

These service modules are highly decoupled building blocks that are small enough to implement a single functionality but together can form larger systems

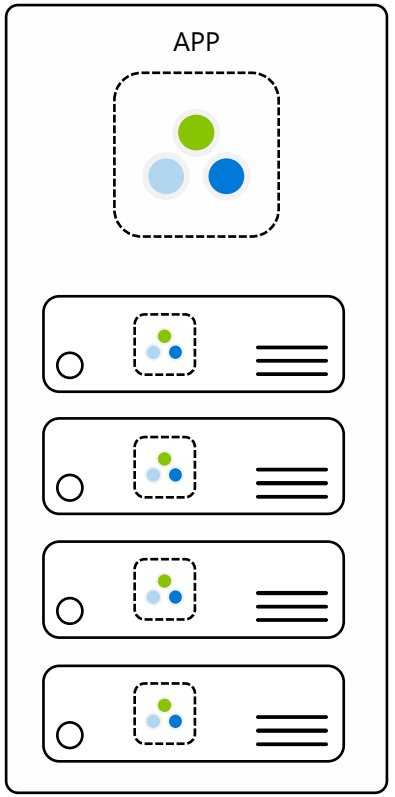
Independently versioned, deployed & scaled

With a microservices architecture, developers can create, manage and improve application services independently, even using different languages

Containers provide the consistent format and isolation desired by microservices.

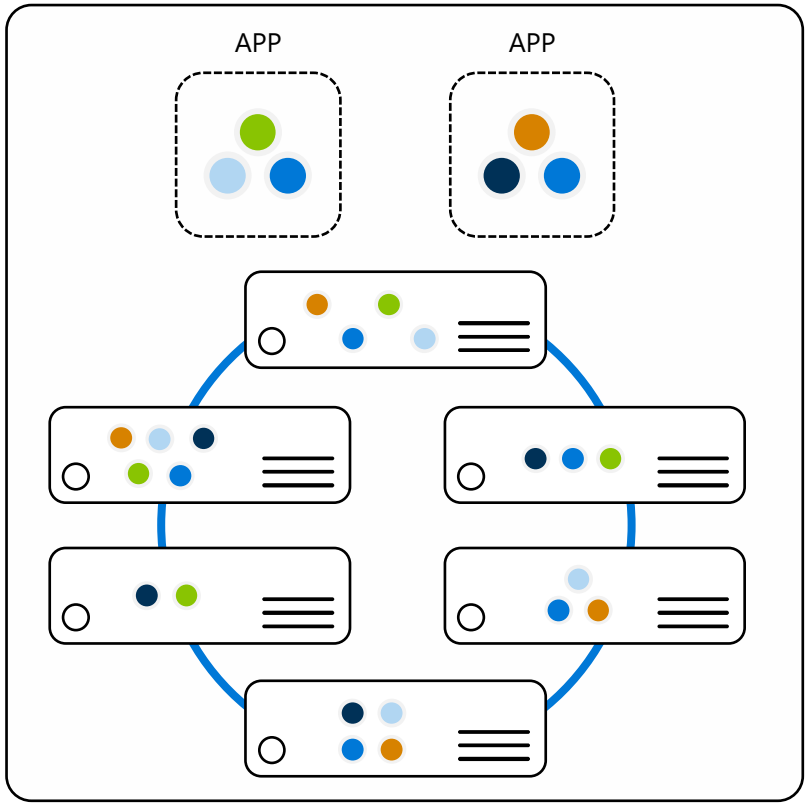
Monolithic

Large, all-inclusive app

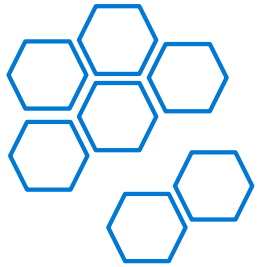


Microservices

Small, independent services



Microservices: Benefits



Encapsulates business functionality into small targeted services, organized around business capabilities

Services deploy frequently and evolve independently

Services scale Independently

Enables technology diversity

- Mix multiple programming platforms and data-storage technologies—best tool for the job
- Rewriting/modernizing a single service is feasible
- "Future-proofs" application investment against obsolete technology stacks

Failure in one service less likely to cause system-wide failure

Microservices: Challenges

No free lunch…

- Architectural and operational complexity increase
- Remote calls escalate network I/O, congestion and latency
- Distributed services expose points of failure, reducing reliability
- Decentralized data require eventual consistency delays
- Integration and versioning concerns become critical
- Service discovery and routing concerns must be handled
- Testing—stubs/mocks become key
- Orchestration, management, and monitoring are mandatory

Microservice candidates

Most solutions do not warrant a microservices architecture

Consider microservices, when…

- Large, strategic enterprise systems that need to align business capabilities/features
- Systems that require a high release velocity: Frequent feature releases with high confidence
- Applications developed by heterogenous teams with expertise in different technology stacks – Polyglot languages and tools
- Application with components that must scale independently

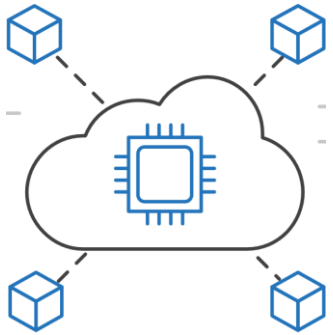
How **Azure** helps

Flexibility

Deploy containerized applications in your preferred environment

Productivity

Accelerate containerized application development



Trust

Manage, monitor, and secure your containers easily

Containers in Azure

App Service

Deploy web apps or APIs using containers in a PaaS environment

Service Fabric

Modernize .NET applications to microservices using Windows Server containers

Kubernetes Service

Scale and orchestrate Linux containers using Kubernetes

Container Instance

Elastically burst from your Azure Kubernetes Service (AKS) cluster

Ecosystem

Bring your Partner solutions that run great on Azure

Azure Container Registry

Docker Hub

Choice of developer tools and clients

Azure Container Instances (ACI)

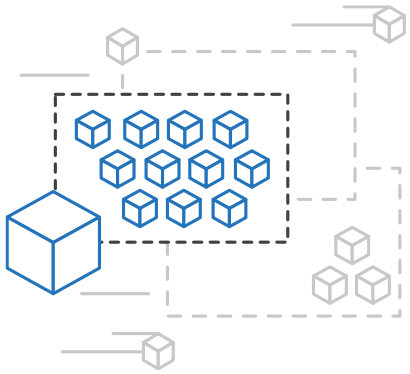
Easily deploy and run containers with a single command

Launch container instances in seconds

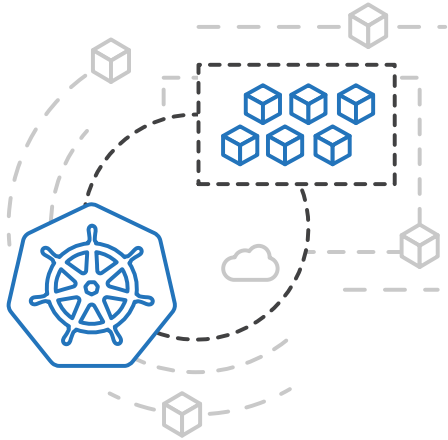
Cost effective per second billing

Azure Container Instances (ACI)

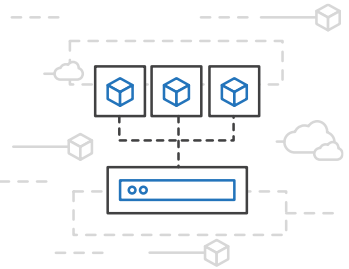
Easily run containers on demand without managing servers



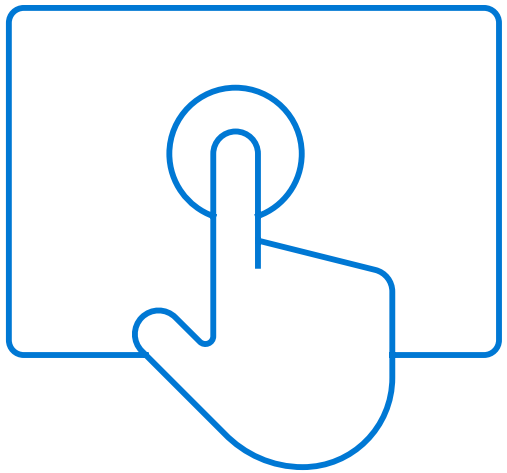
Run containers without managing servers



Increase infrastructure agility with containers on demand



Secure applications with hypervisor isolation

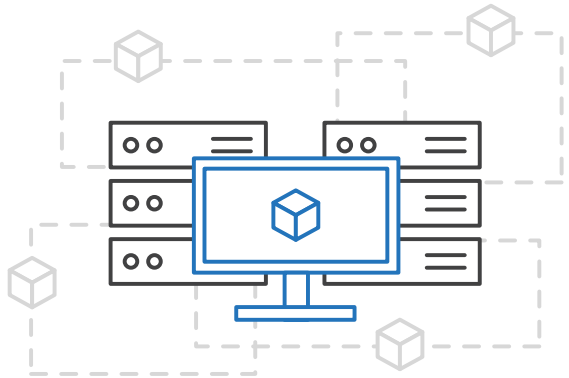


Demo

Spinning up a container using ACI

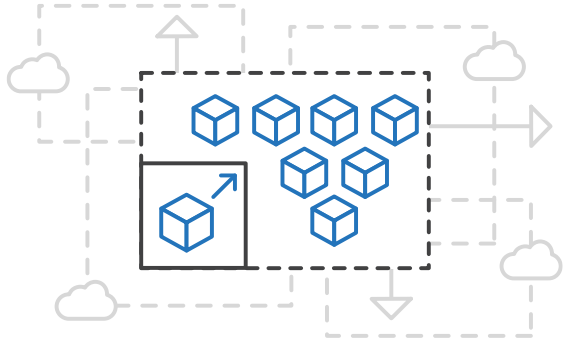
Azure Kubernetes Service (AKS)

Simplify the deployment, management, and operations of Kubernetes



Focus on your containers not the infrastructure

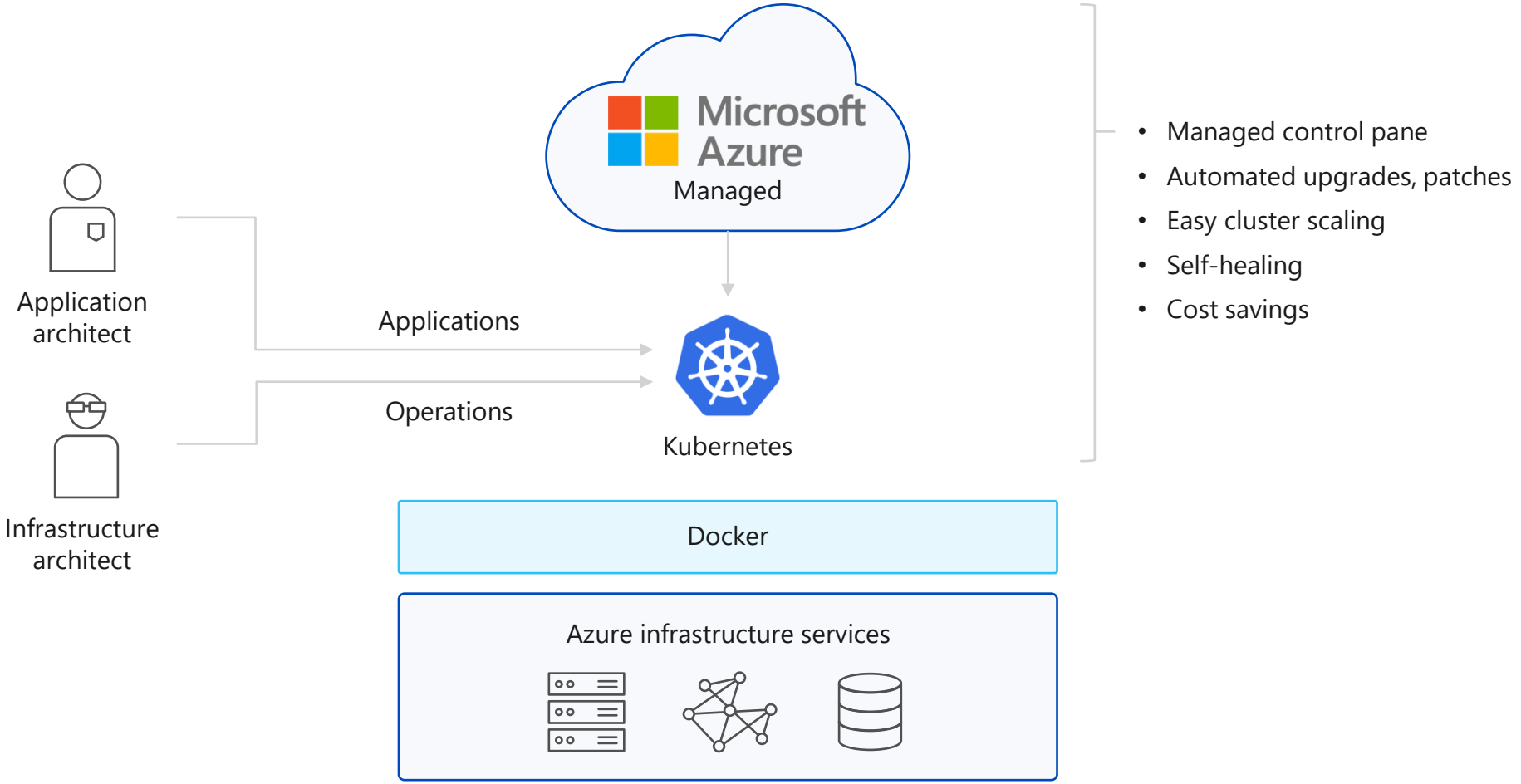
Work how you want with open-source APIs



Scale and run applications with confidence

Azure Kubernetes Service (AKS)

A fully managed Kubernetes cluster



- Managed control pane
- Automated upgrades, patches
- Easy cluster scaling
- Self-healing
- Cost savings

Azure Kubernetes Service (AKS)

Get started easily

```
$ az aks create -g myResourceGroup -n myCluster --generate-ssh-keys
\ Running ..

$ az aks install-cli
Downloading client to /usr/local/bin/kubectl ..

$ az aks get-credentials -g myResourceGroup -n myCluster
Merged "myCluster" as current context ..

$ kubectl get nodes
NAME                          STATUS    AGE       VERSION
aks-mycluster-36851231-0      Ready     4m        v1.8.1
aks-mycluster-36851231-1      Ready     4m        v1.8.1
aks-mycluster-36851231-2      Ready     4m        v1.8.1
```

Azure Kubernetes Service (AKS)

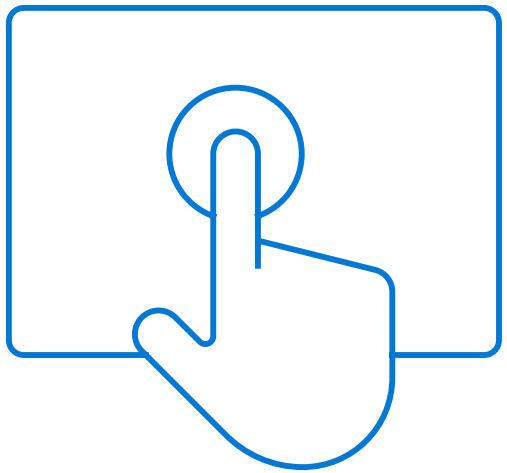
Manage an AKS cluster

```
$ az aks list –o table
Name                Location    ResourceGroup     KubernetesRelease  ProvisioningState
------------------  ----------  -------------  ------------------  ------------------
myCluster           westus2     myResourceGroup               1.7.7  Succeeded

$ az aks upgrade -g myResourceGroup -n myCluster –-kubernetes-version 1.8.1
\ Running ..

$ kubectl get nodes
NAME                          STATUS    AGE       VERSION
aks-mycluster-36851231-0      Ready     12m       v1.8.1
aks-mycluster-36851231-1      Ready     8m        v1.8.1
aks-mycluster-36851231-2      Ready     3m        v1.8.1

$ az aks scale -g myResourceGroup -n myCluster --agent-count 10
\ Running ..
```

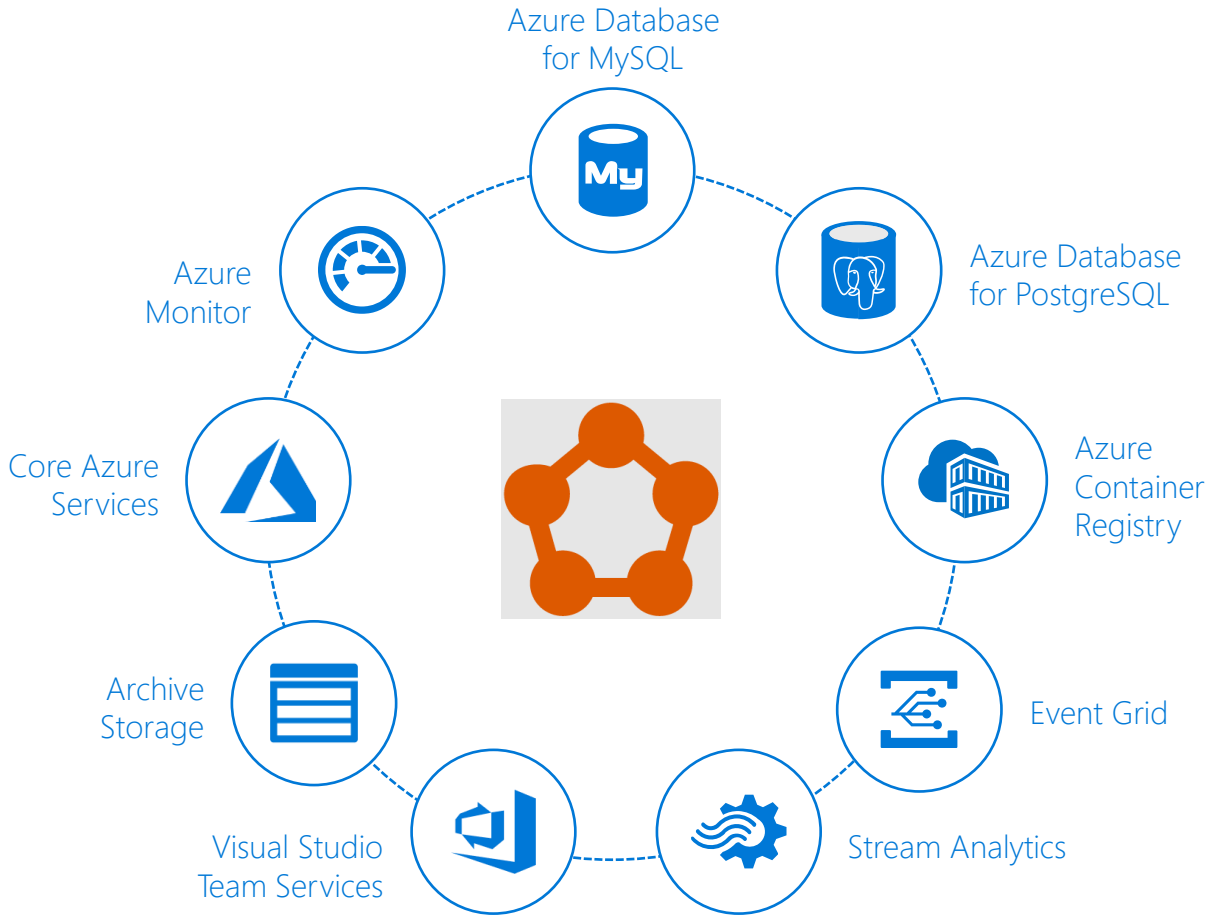


Demo

Deploy containers on AKS

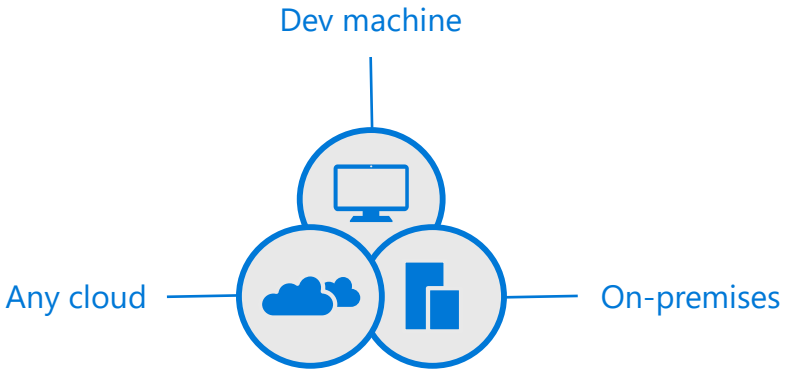
Powering Azure and Microsoft services

Service Fabric is designed for mission-critical services



Service Fabric

Azure Service Fabric Offerings



Service Fabric Standalone

Bring your own infrastructure

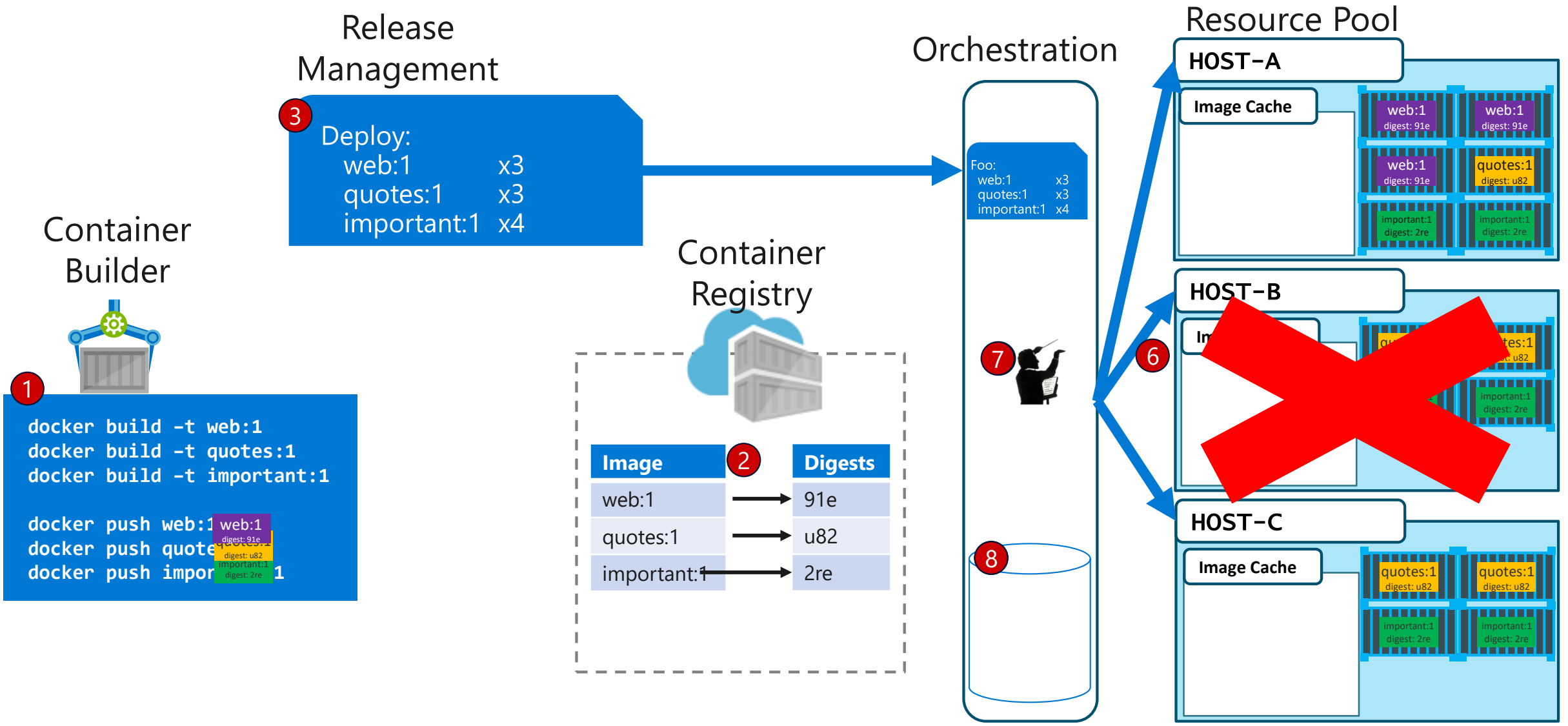
Azure Service Fabric

Dedicated Azure clusters

Azure Service Fabric Mesh

Serverless microservices

Full Control ⟷ *Fully managed*


Release Management
3
Deploy:
web:1 x3
quotes:1 x3
important:1 x4
Container Builder
1
docker build -t web:1
docker build -t quotes:1
docker build -t important:1
docker push web:1
docker push quote
docker push impor 1
web:1
digest: 91e
digest: u82
important:1
digest: 2re
Container Registry
Image
2
Digests
web:1
91e
quotes:1
u82
important:1
2re
Orchestration
Foo:
web:1 x3
quotes:1 x3
important:1 x4
7
8
6
Resource Pool
HOST-A
Image Cache
web:1
digest: 91e
quotes:1
digest: u82
important:1
digest: 2re
HOST-B
HOST-C

Thank You!