Unleash Opportunities With Compute options on Azure

Sajeetharan S

Cloud Solution Architect (APAC,OCP)

"Azure be the cloud you love"

www.Sajeetharan.com @kokkisajee sasinnat@Microsoft.com

Welcome!

HOW DO I ASK A QUESTION?

- If you have a technical or content-related question, please use the Q&A window
- We will address the questions as they come in

CAN I VIEW THIS PRESENTATION AFTER THE WEBINAR?

- There will not be a recording of the session, slides will be shared in the GitHub repository
- Due to the PII we are not keeping the recording of the session

Agenda!

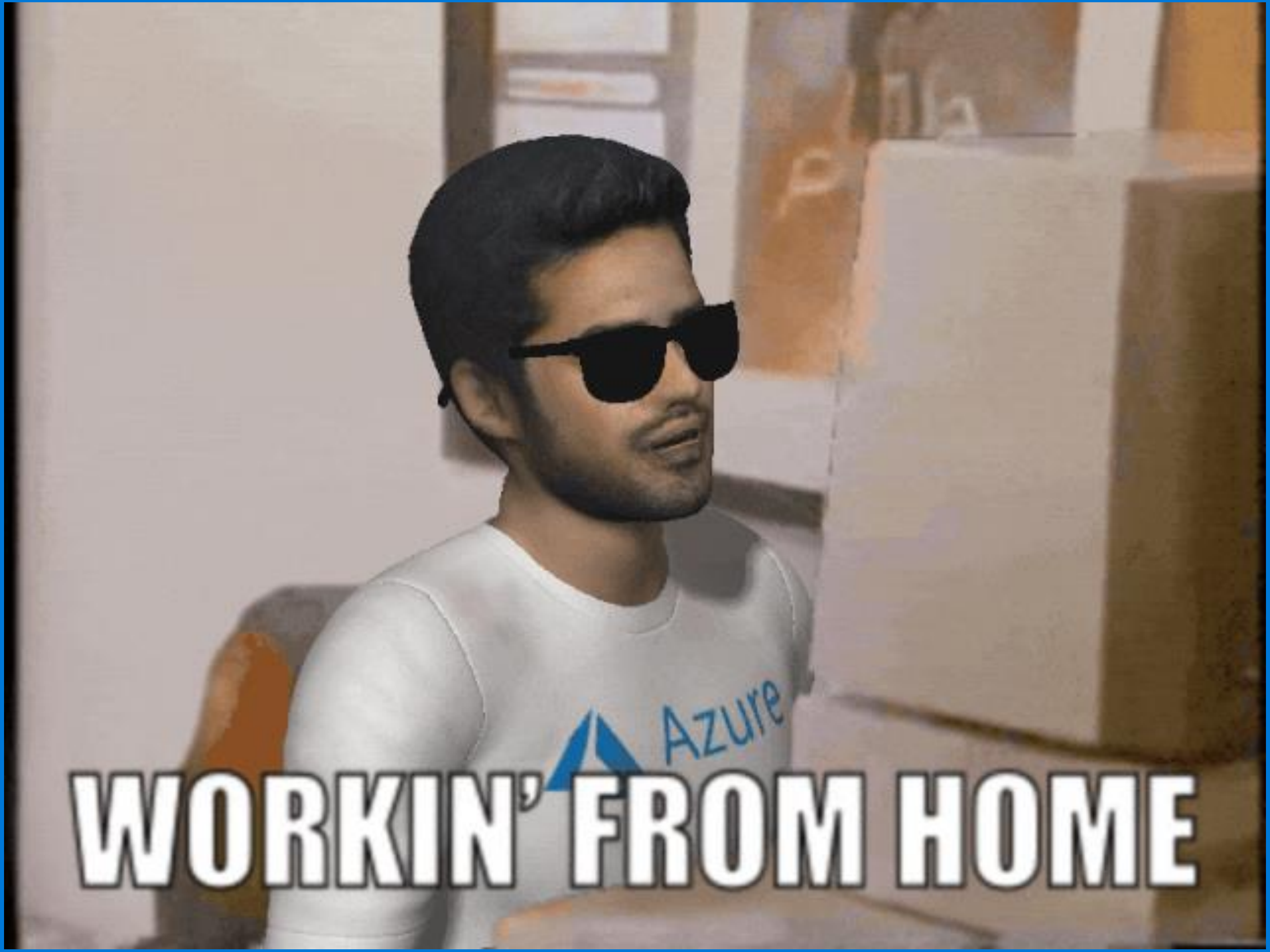
- Why Application Innovation is important?
- Decision Matrix for the compute options on Azure
- Overview of each compute services and recent updates
- Why you need to design differently on the cloud
- Reference Architecture
- Demo on Event Driven Scaling (AKS + Azure functions)
- Kahoot Quiz

POLL:

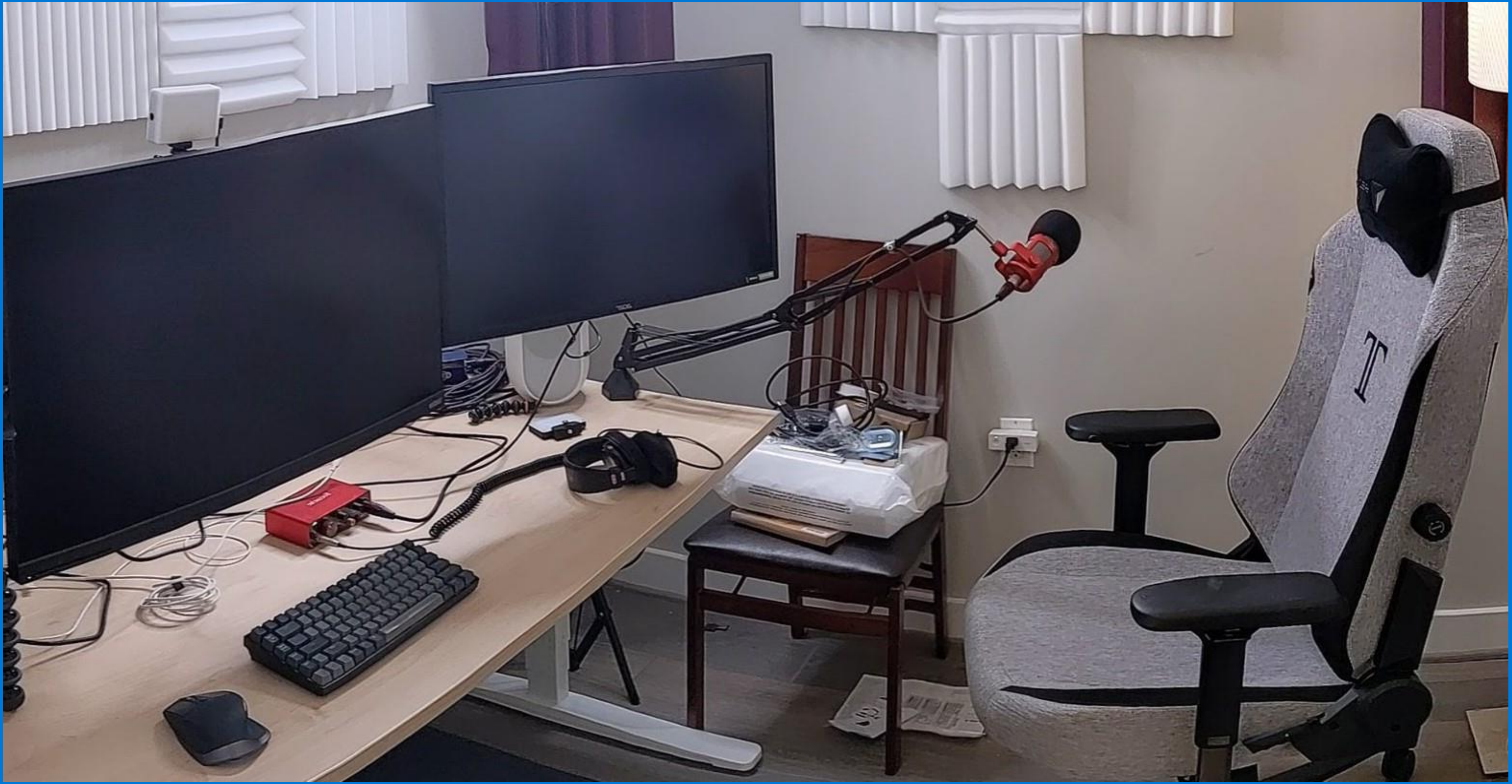
What languages do you use?

What are the compute services that you are familiar with on Azure ?

Let me share a story



Sajee's work from home setup



Traditional application has a set of challenges

Aging infrastructure

- Aging hardware, operating systems, and business applications in the datacenter can impact:
- Operational costs, efficiency, and reliability
- Capital expenditure requirements
- Security, audit, and regulatory compliance

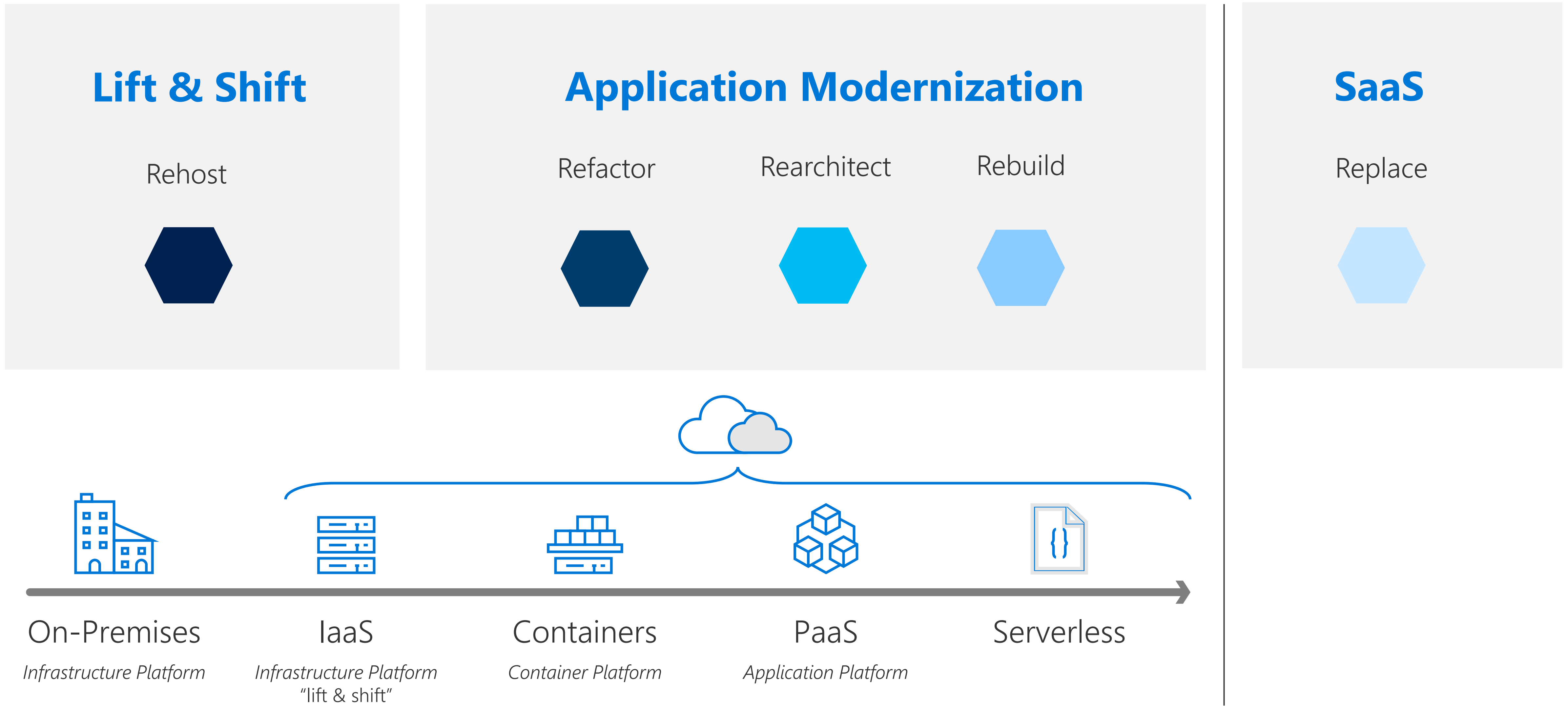
Lack of agility

- Deployment time of new services
- Operation is time (and budget) consuming
- Innovation is happening outside IT inside business areas

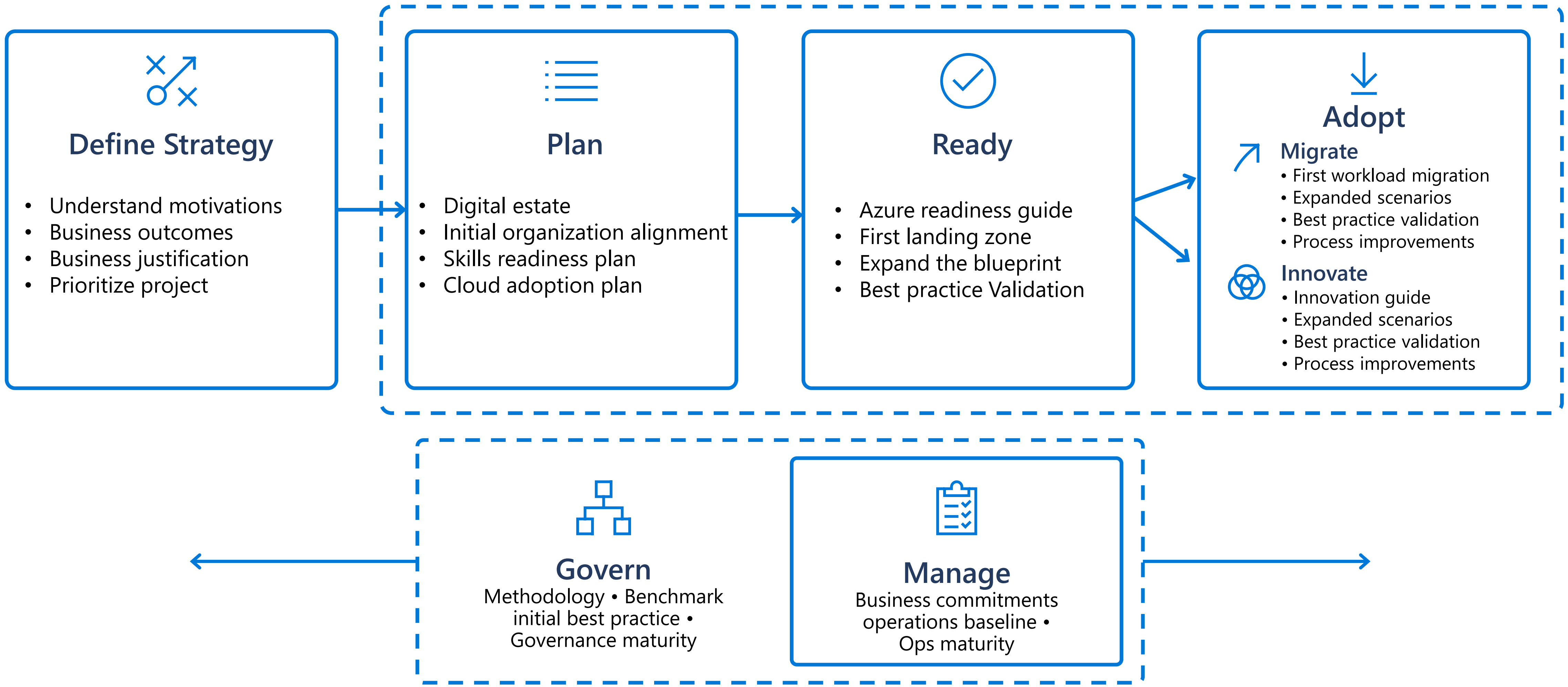
Legacy applications

- Longer release cycles, monolithic and highly coupled architecture
- Highly IT dependent
- Low application performance and time-to-market compromise business agility

The (application) Journey to the Cloud



Azure Cloud Adoption Framework



Define Strategy

- Understand motivations
- Business outcomes
- Business justification
- Prioritize project

Plan

- Digital estate
- Initial organization alignment
- Skills readiness plan
- Cloud adoption plan

Ready

- Azure readiness guide
- First landing zone
- Expand the blueprint
- Best practice Validation

Adopt

Migrate

- First workload migration
- Expanded scenarios
- Best practice validation
- Process improvements

Innovate

- Innovation guide
- Expanded scenarios
- Best practice validation
- Process improvements

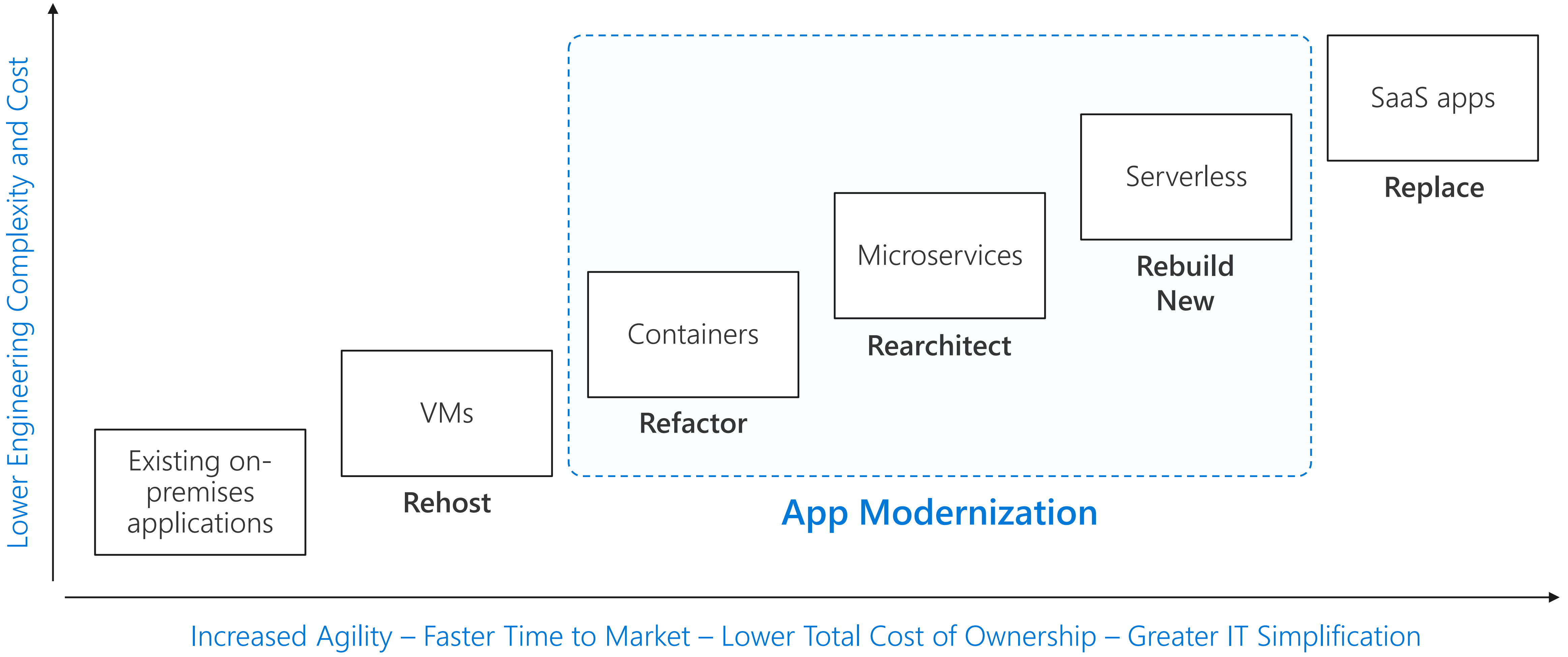
Govern

Methodology • Benchmark
initial best practice •
Governance maturity

Manage

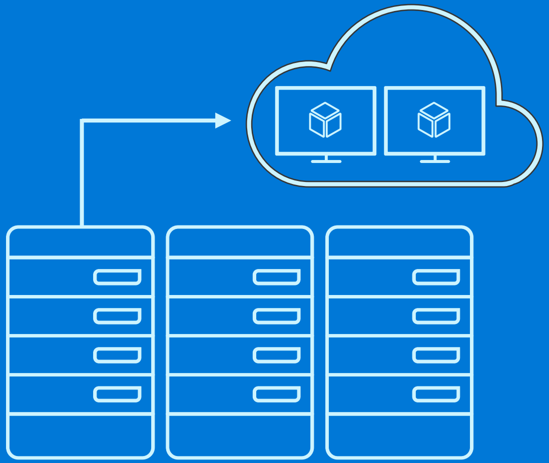
Business commitments
operations baseline •
Ops maturity

Cloud app continuum


Lower Engineering Complexity and Cost
Existing on-premises applications
VMs
Rehost
Containers
Refactor
Microservices
Rearchitect
Serverless
Rebuild New
SaaS apps
Replace
App Modernization
Increased Agility – Faster Time to Market – Lower Total Cost of Ownership – Greater IT Simplification

Disclaimer : Not be the case on every scenario!

Lift and Shift(Rehost)



Definition:
Redeploy the application to a different hardware environment or change the application's infrastructure configuration

When to consider

- Ideal when your goal is to improve operational efficiencies, and free up data center space
- Maintenance apps for which the hardware is not worth additional investment
- Compute-intensive applications that are built for parallelism but don't require high-performance interprocess communications (IPC) and have independent datasets, and applications for which load balancing already increases scalability and availability.

Benefits

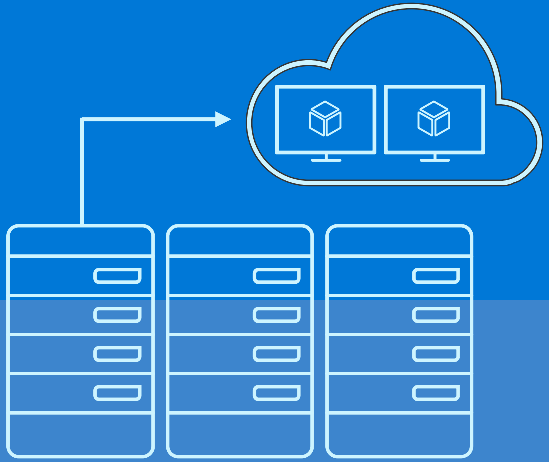
- Drives instant reduction in TCO - 30% on average
- No need to manage data centers
- Enjoy flexible and scalable infrastructure

Core technologies

- VM, VM Scale Set

Source: Decision Point for Choosing a Cloud Application Migration Strategy, Gartner. Published: 29 March 2016

Refactor



Definition

Modify your application so that it can begin to take advantage of cloud capabilities for agility, elasticity and minimized resource use

When to consider

- You want to leverage existing development skills and codebase is paramount
- When code portability is a concern.
- You prefer a quick way to modernize your apps

Benefits

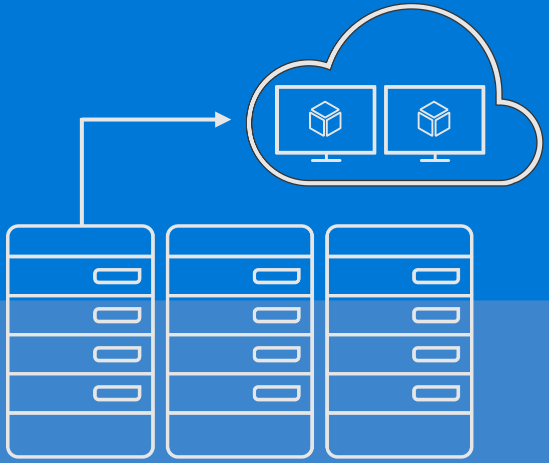
- **Drive continuous innovation by leveraging built-in DevOps for PaaS or using Containers.**
- Existing programming models, languages and frameworks that can be easily used and extended.
- Easily scale up or down to meet the changing needs of the business

Core technologies

- Containers, container ochestration
- DevOps tools

Source: Decision Point for Choosing a Cloud Application Migration Strategy, Gartner. Published: 29 March 2016

Rebuild



Definition:
Build new application using cloud native environment. Wherever possible, prioritize high-productivity PaaS - model driven or rapid application development

When to consider

- You want to build for cloud-native PaaS environments from ground up.
- Leverage previous investment in a cloud platform, e.g. when customer data has already moved to the Cloud.
- Rapid prototyping is crucial or the scope of a current application is too limited in terms of functionality and lifespan.

Benefits

- Reduce TCO
- Fully leverage the cloud native capabilities and build applications faster
- Expedite your business innovation

Core technologies

- Serverless, PaaS

Source: Decision Point for Choosing a Cloud Application Migration Strategy, Gartner. Published: 29 March 2016

Choosing migration strategy and technology

		Objectives	Cloud strategy					Options to consider
			Rehost	Refactor	Rearchitect	Re-build	Replace	
Innovation	1	Deliver new capabilities faster				✓		PaaS, Serverless
	2	Provide multichannel access, including mobile				✓	✓	PaaS, Serverless
	3	Enable business agility with continuous innovation		✓	✓			PaaS, Containers
Differentiation	1	More easily integrate with other web and cloud apps			✓	✓		PaaS, Serverless
	2	Infuse intelligence into processes leveraging existing investments		✓	✓			PaaS, Serverless
	3	Increase agility & support scalability requirements of existing applications more cost effectively		✓	✓			PaaS, Containers
Record	1	Free up data center space quickly	✓				✓	VMs, SaaS
	2	Reduce capital expenditure of existing applications	✓				✓	VMs, SaaS
	3	Achieve rapid time to cloud	✓					VMs

Note: Some of the objective might apply to more than one category of applications

What is Cloud Native App Development?

Package app code & dependencies in **Kubernetes** containers

Deploy as **microservices**

Manage app with **DevOps** processes & tools

By 2020, 35% of production apps will be cloud native

Source: IDC FutureScape: Worldwide Developer and DevOps 2019 Predictions, Oct 2018

Common cloud native scenarios

Modernize business critical applications

SaaS delivery

Real-time telemetry

Geo-distributed applications

Key Components of Cloud Native

Containers

Tool to package your app, run it portably on different hosts in a consistent way

Serverless

Platform for running and scaling apps where almost all of the operations tasks are managed by the cloud provider. Optimized to let developers focus on code and business value.

Kubernetes

Platform to manage and scale your app reliably (made up of containers) that may span many physical and virtual machines.

A tool for operations, not development

Azure: The Power Of Choice

Compute

Virtual Machines | Container Services | App Service | Functions

More Control ⟷ Focus on the App

Customer-managed
(IaaS)

Platform-managed
(PaaS)

Code-only
(serverless)

Azure: The Power Of Choice

Application Hosting

Virtual Machines

Customer-managed
(IaaS)

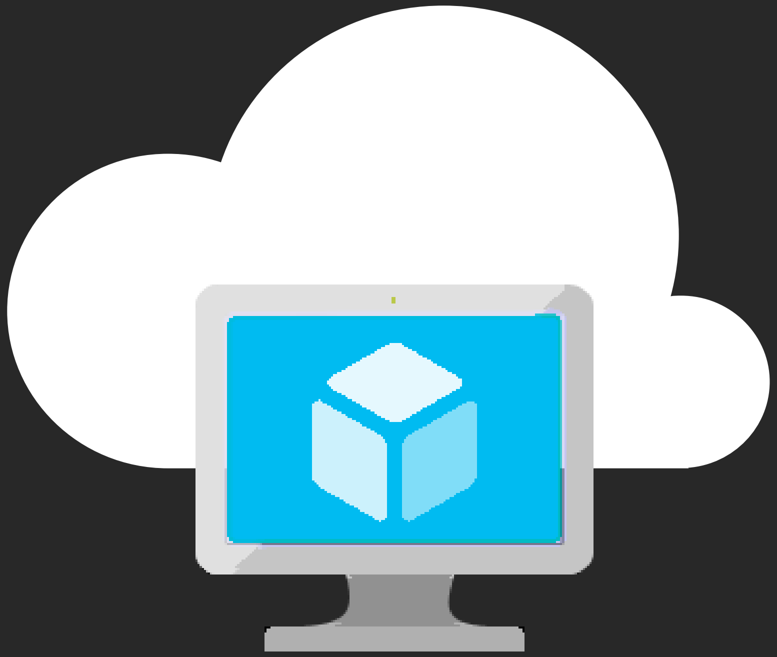
Virtual Machines

Ubuntu, Red Hat, Windows, SUSE, CoreOS

DevOps Extensions with Chef and Puppet

Multiple sizes

Hundreds of items in marketplace



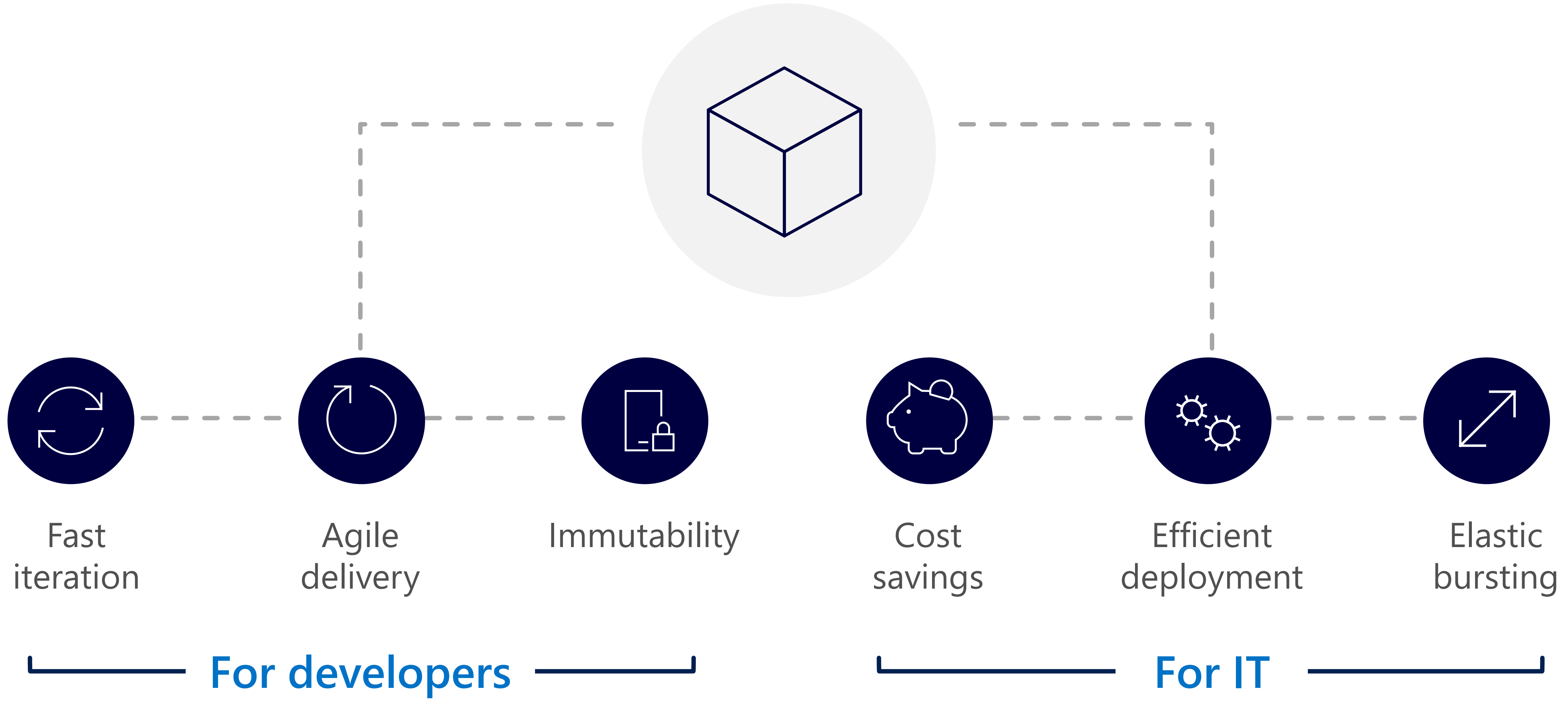
Azure: The Power Of Choice

Virtual Machines

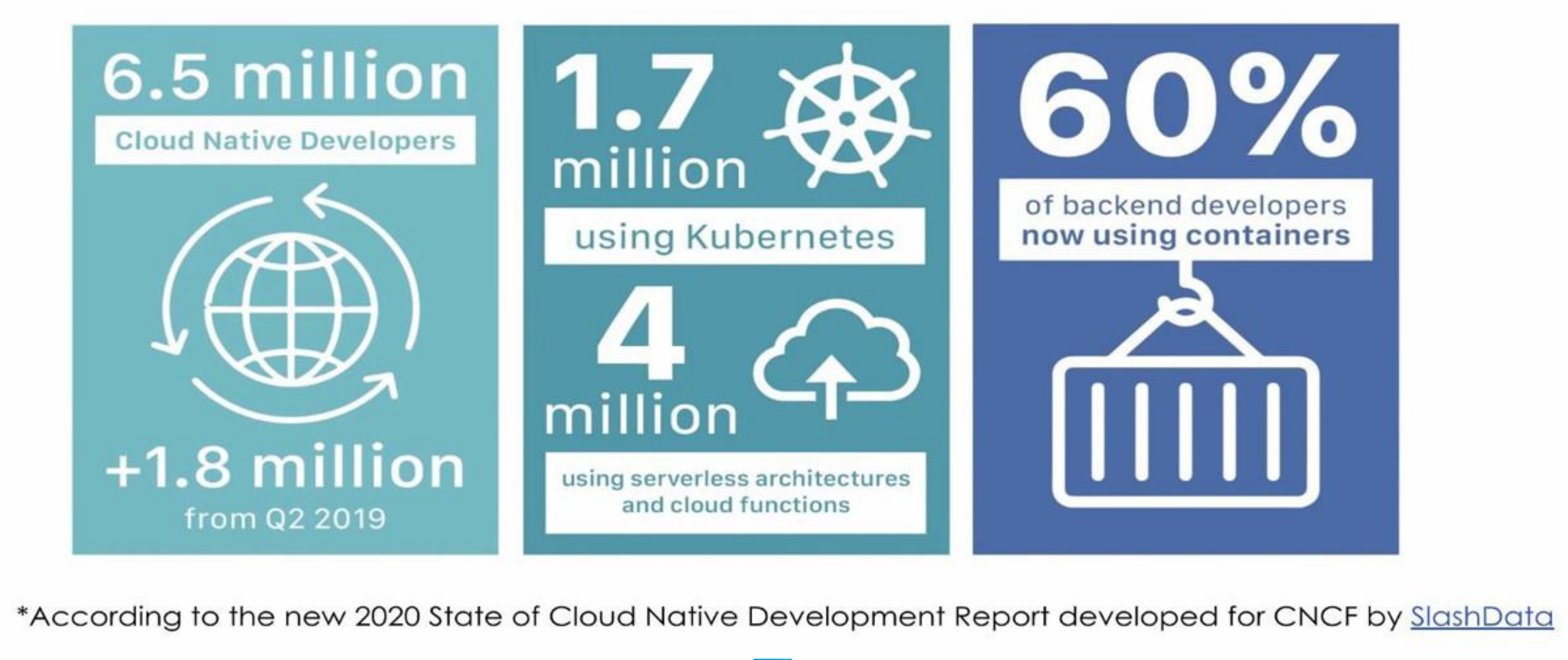
Containers

Customer-managed
(IaaS)

The container advantage



Developers love Kubernetes



6.5 million

Cloud Native Developers

+1.8 million

from Q2 2019

1.7 million

using Kubernetes

4 million

using serverless architectures and cloud functions

60%

of backend developers **now using containers**

*According to the new 2020 State of Cloud Native Development Report developed for CNCF by SlashData

MICROSOFT CONFIDENTIAL

AKS: Simplify the deployment, management, and operations of Kubernetes

Deploy and manage Kubernetes with ease

Accelerate containerized application development

Set up CI/CD in a few clicks

Secure your Kubernetes environment

Scale and run applications with confidence

Work how you want with open-source tools & APIs

Scenarios for AKS

Lift and shift to containers

Cost saving

without refactoring your app

Microservices

Agility

Faster application development

Machine learning

Performance

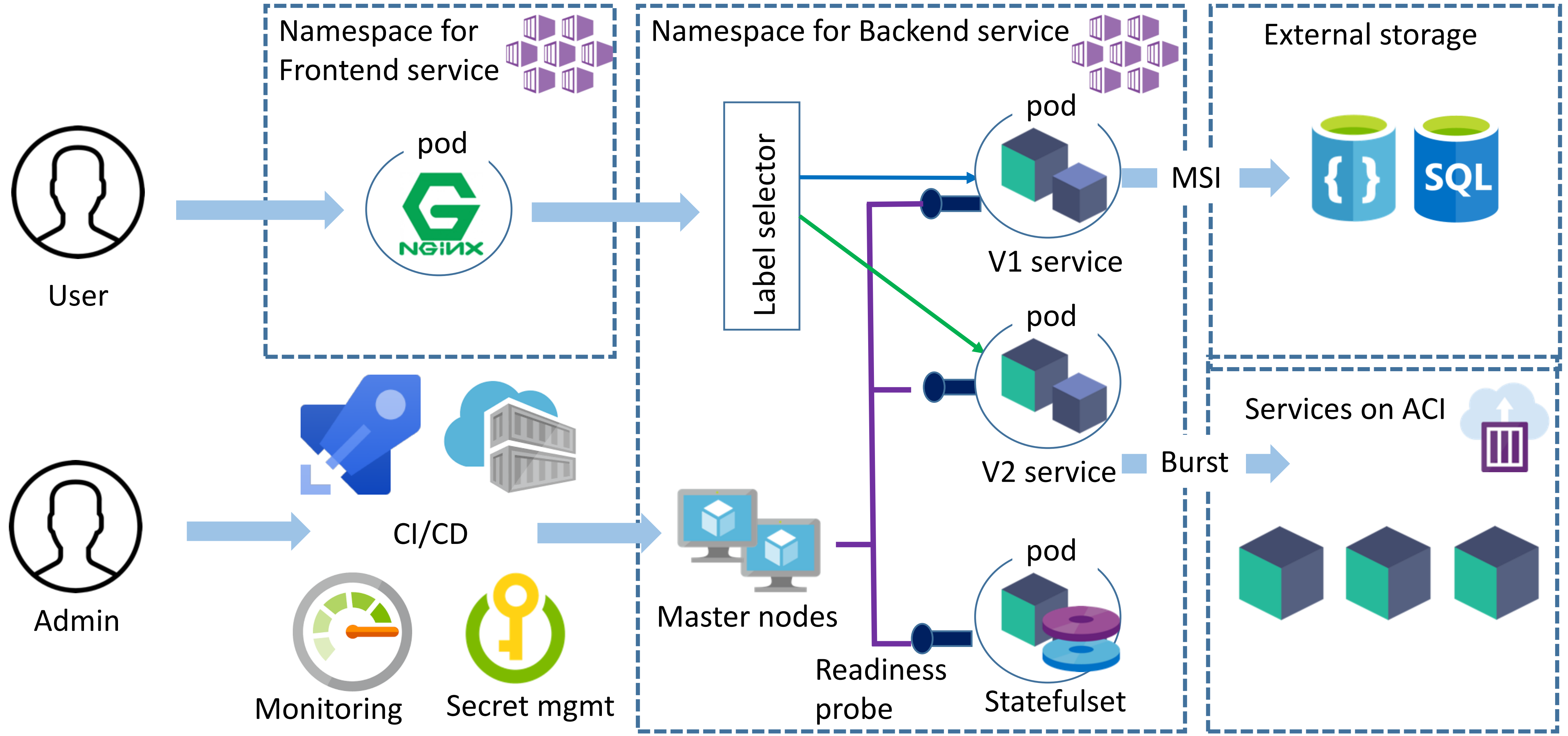
Low latency processing

IoT

Portability

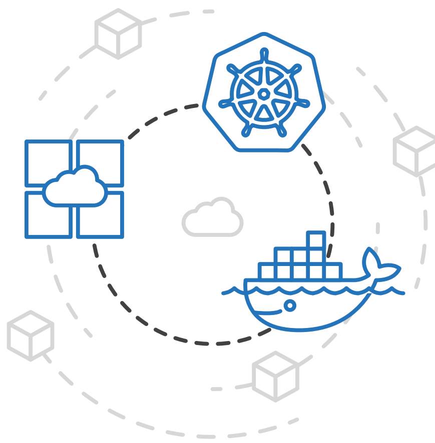
Build once, run anywhere

Microservices with AKS

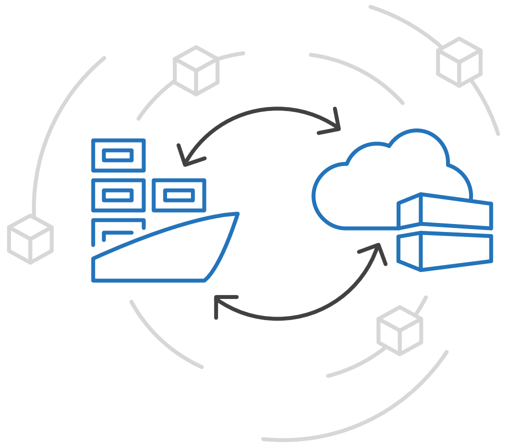


Azure Container Registry

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

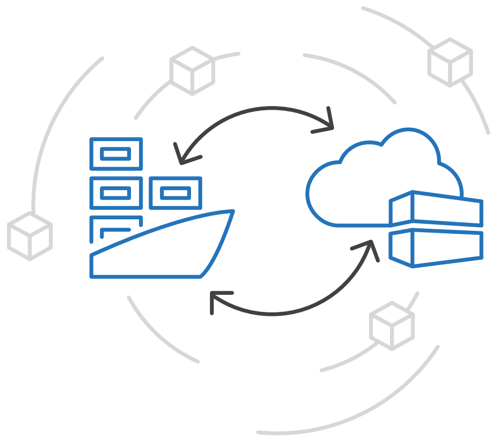
Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

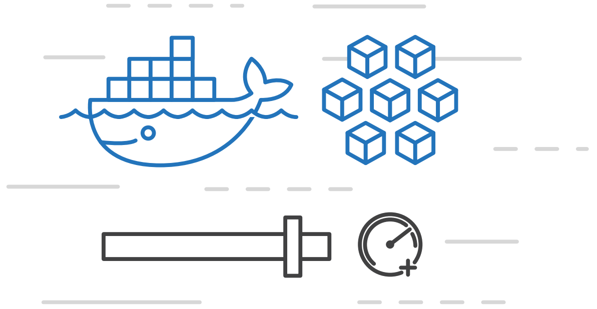
Azure Container Registry

Azure Container Instances (ACI)

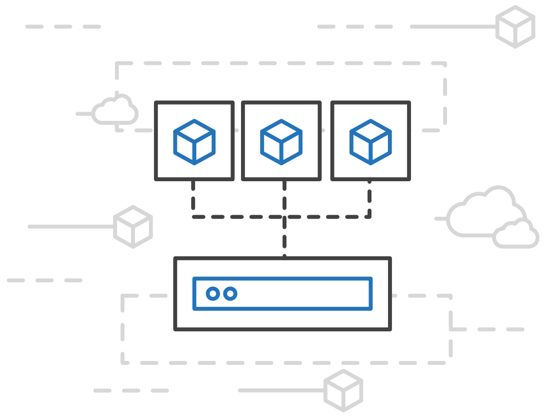
Easily run containers on Azure with a single command



Start using containers right away



Cloud-scale container capacity



Hyper-visor isolation

Azure: The Power Of Choice

Compute

Virtual Machines

Container Service

App Service

More Control

Focus on the App

Customer-managed
(IaaS)

Platform-managed
(PaaS)

Azure App Service

Quickly build, deploy and scale powerful cloud applications without worrying about infrastructure

High productivity	.NET, Node, Java, Docker, PHP, Ruby, Python	Staging & deployment	Testing in production	App gallery marketplace
Fully managed	Auto scale & load balancing	High availability w/ auto patching	Reduced operations costs	Backup & recovery
Enterprise grade	Global data center footprint	Hybrid support	AAD integration	Secure & compliance

Bring what you have

Use the framework, container, or OS of your choice on a fully managed platform.

Code...

.NET, Node, Java, PHP, Ruby, Python

→

Container...

→

OS...

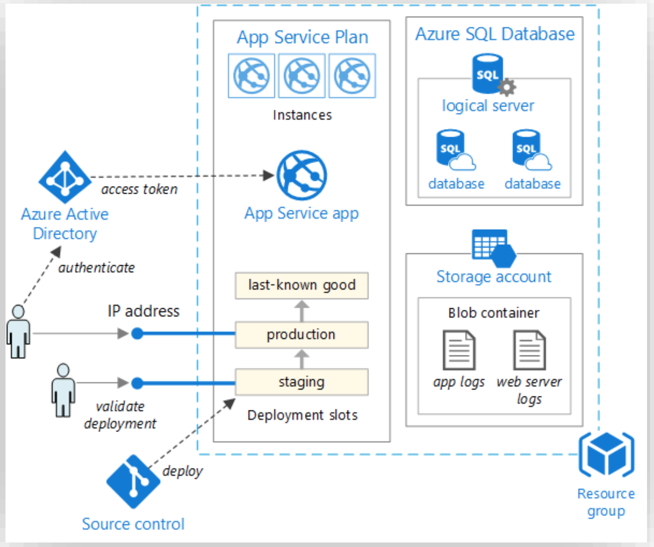
→

Azure App Service

- Staging & deployment
- Testing in production
- App Monitoring & Diagnostics
- Auto scale & load balancing
- High availability w/ auto patching
- Reduced operations costs
- Backup & recovery
- Security & compliance
- Global data center footprint
- AAD integration

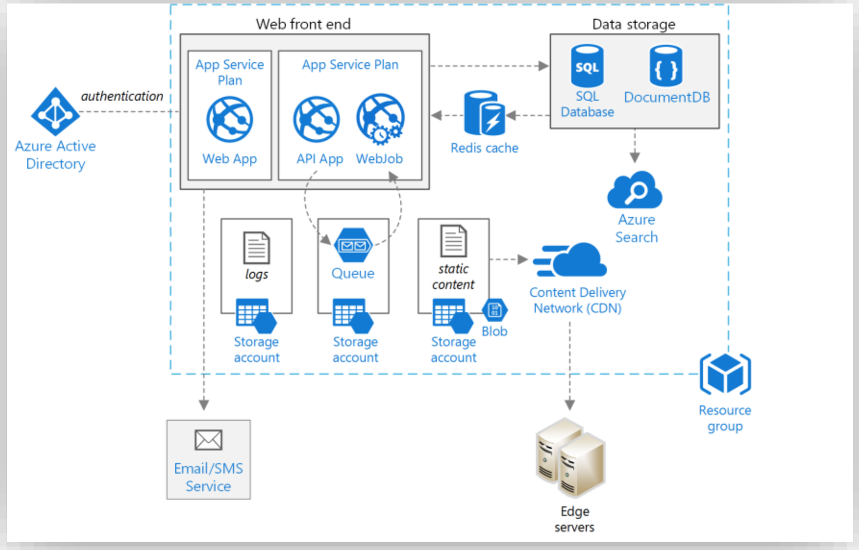
Reference Architecture for Managed Web

Basic Web



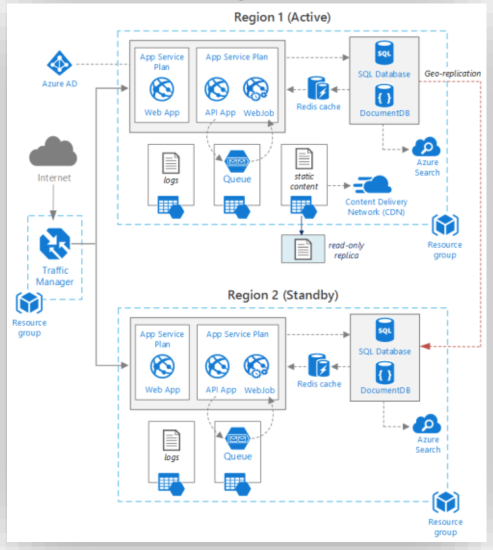
✓Service plan
✓Deployment
✓Authentication
✓SQL DB
✓Diagnostics

Improving Scalability



✓WebJobs
✓Cache
✓CDN
✓Other storages
✓ API App

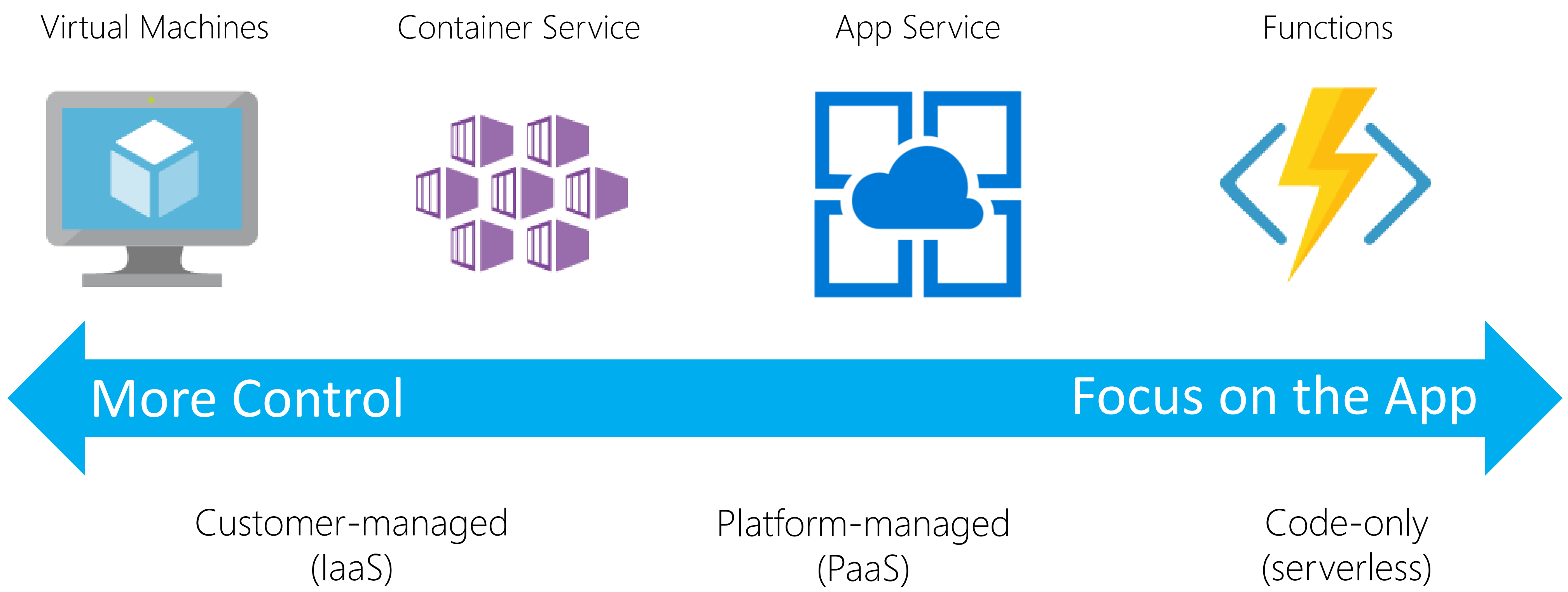
Improving Availability



✓Hosting in paired region
✓Traffic manager
✓Geo-replication

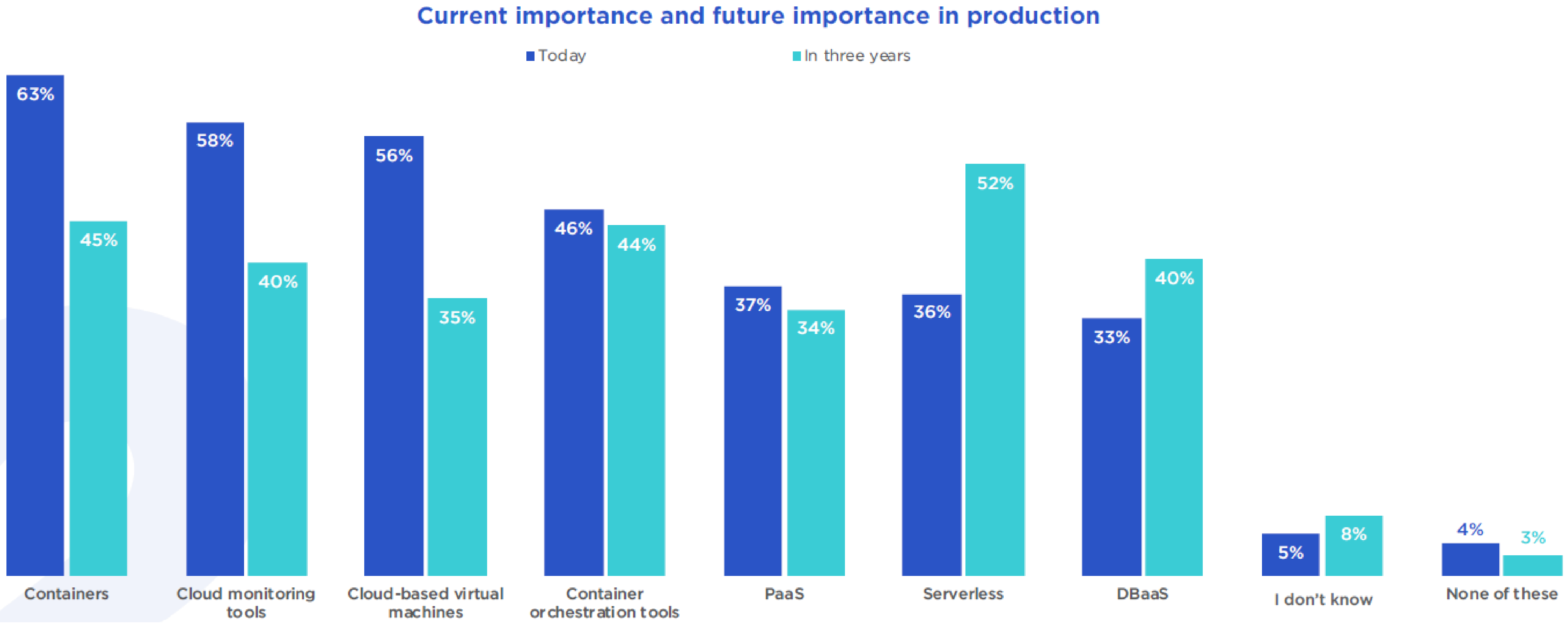
Azure: The Power Of Choice

Compute



Serverless stands out as the technology

% of professional backend developers by technology importance for deploying applications in the cloud (Q4 2019 n= 3,430)



Source: SlashData Developer Economics Q4 2019, professional back-end services / cloud developers worldwide

What is serverless?

Full abstraction of servers
Developers can just focus on their code—there are no distractions around server management, capacity planning, or availability.

Instant, event-driven scalability
Application components react to events and triggers in near real-time with virtually unlimited scalability; compute resources are used as needed.

Pay-per-use
Only pay for what you use: billing is typically calculated on the number of function calls, code execution time, and memory used.*

*Supporting services, like storage and networking, may be charged separately.

Azure serverless ecosystem

IoT Hubs

Storage

Cognitive Services

App Insights

Cosmos DB

Front Door

Event Hubs

Media Services

Container instances

Active Directory

Stream Analytics

Monitor

Event-driven serverless offerings

Functions

Logic Apps

Event Grid

API Management

Managed Identity

Key Vault

Service Bus

DevOps

SQL

CDN

- IDE integration
- Local development
- Flexible deployment options
- Built-in security
- Rich monitoring
- Compliance and management

FaaS is at the center of serverless

Single responsibility

Functions are single-purposed, reusable pieces of code that process an input and return a result

Short lived

Functions don't stick around when finished executing, freeing up resources for further executions

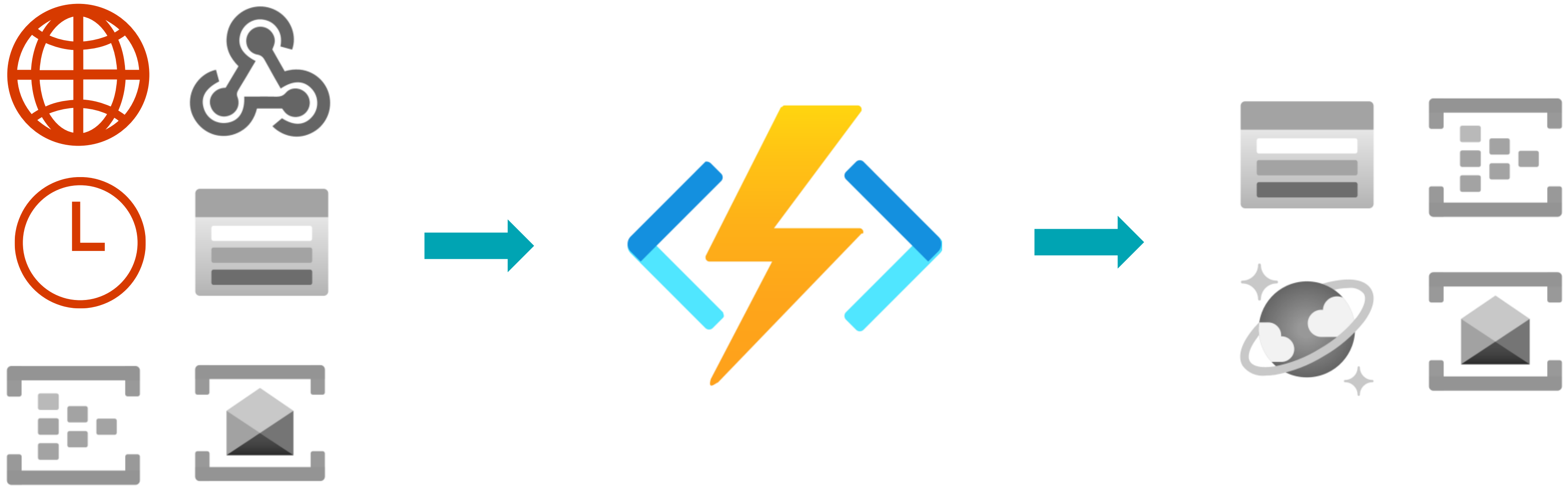
Stateless

Functions don't hold any persistent state and don't rely on the state of any other processes

Event driven & scalable

Functions respond to predefined events, and are instantly replicated as many times as needed

Azure Functions Programming Model



React to timers, HTTP, or events from your favorite Azure services, with more on the way

Events

Author functions in C#, F#, Node.JS, Java, Python, PowerShell, and more

Stateful orchestrations w/ Durable Functions

Code

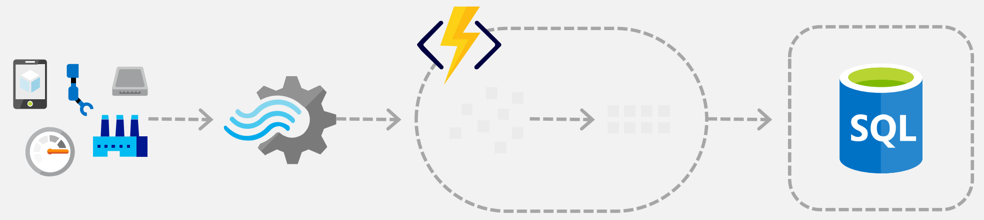
Send results to an ever-growing collection of services

Outputs

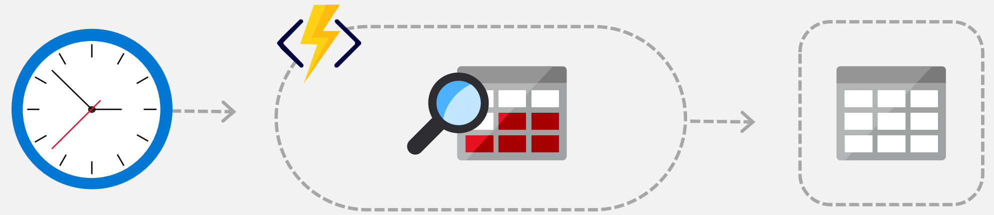
Scenarios for Serverless

Anything that needs to respond to events

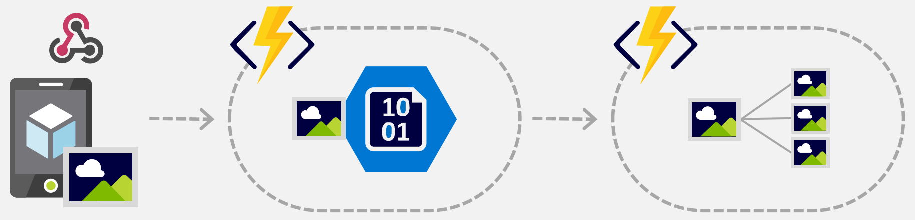
Real-time stream processing


SQL
Millions of devices feed into Stream Analytics
Transform to structured data
Store data in SQL DB

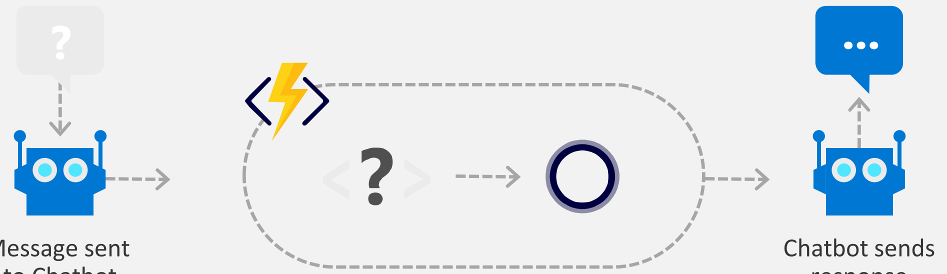
Timer-based processing



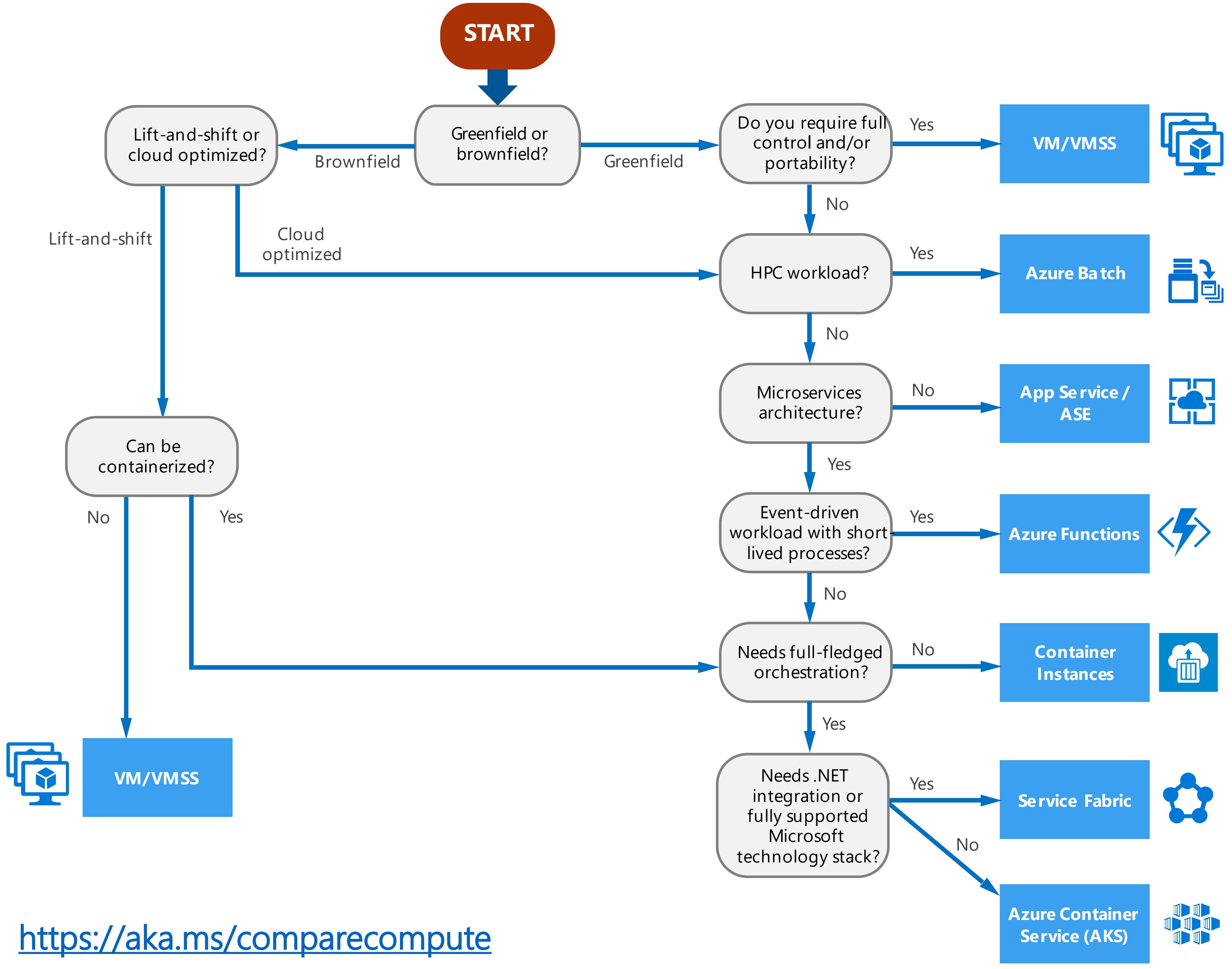
Backends (Mobile/IoT/Web)



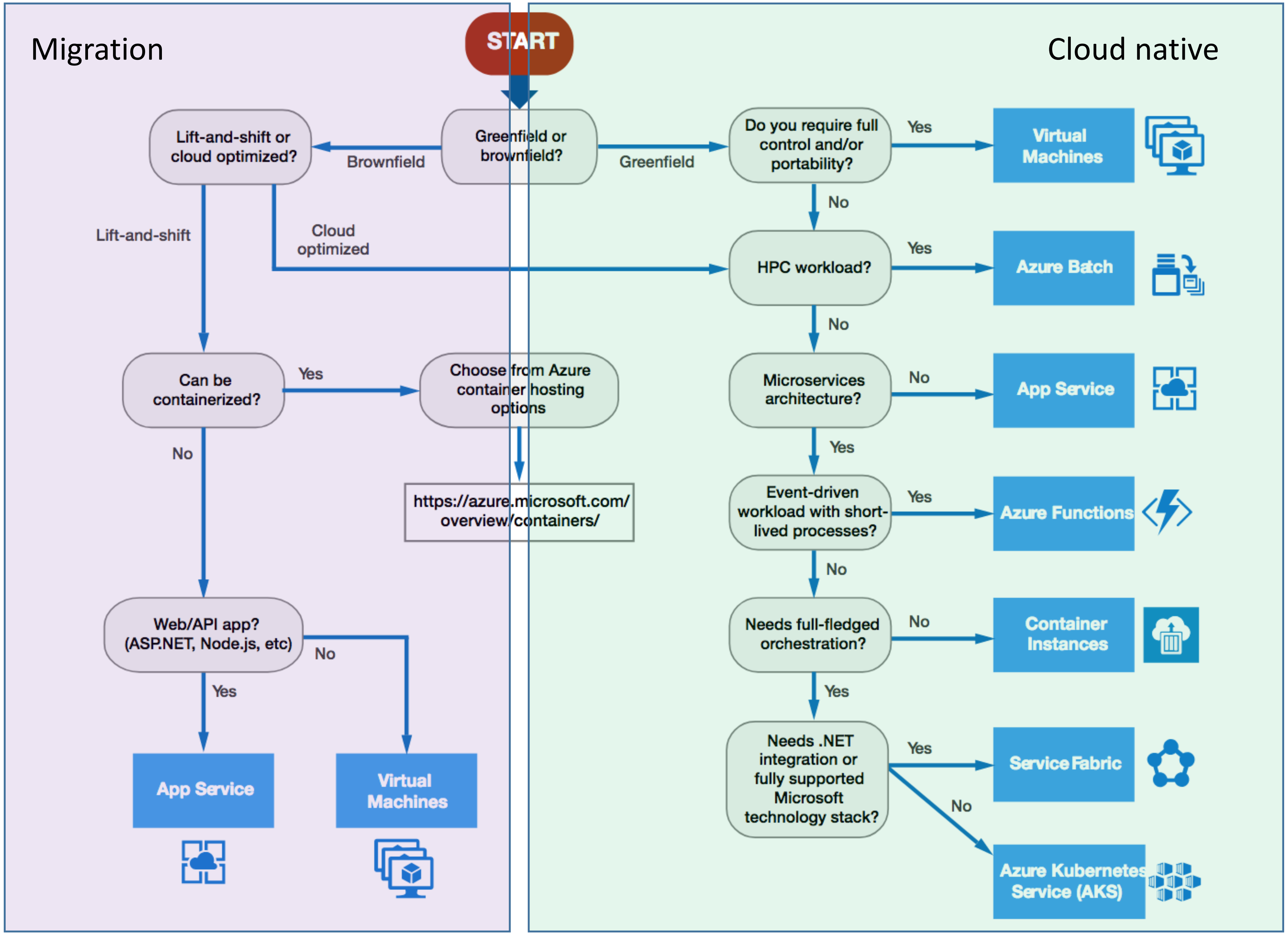
Real-time bot messaging



Azure Doc: Decision Tree


START
Greenfield or brownfield?
Brownfield
Greenfield
Lift-and-shift or cloud optimized?
Lift-and-shift
Cloud optimized
Do you require full control and/or portability?
Yes
VM/VMSS
No
HPC workload?
Yes
Azure Batch
No
Microservices architecture?
No
App Service / ASE
Yes
Event-driven workload with short lived processes?
Yes
Azure Functions
No
Needs full-fledged orchestration?
No
Container Instances
Yes
Needs .NET integration or fully supported Microsoft technology stack?
Yes
Service Fabric
No
Azure Container Service (AKS)
Can be containerized?
No
Yes
VM/VMSS

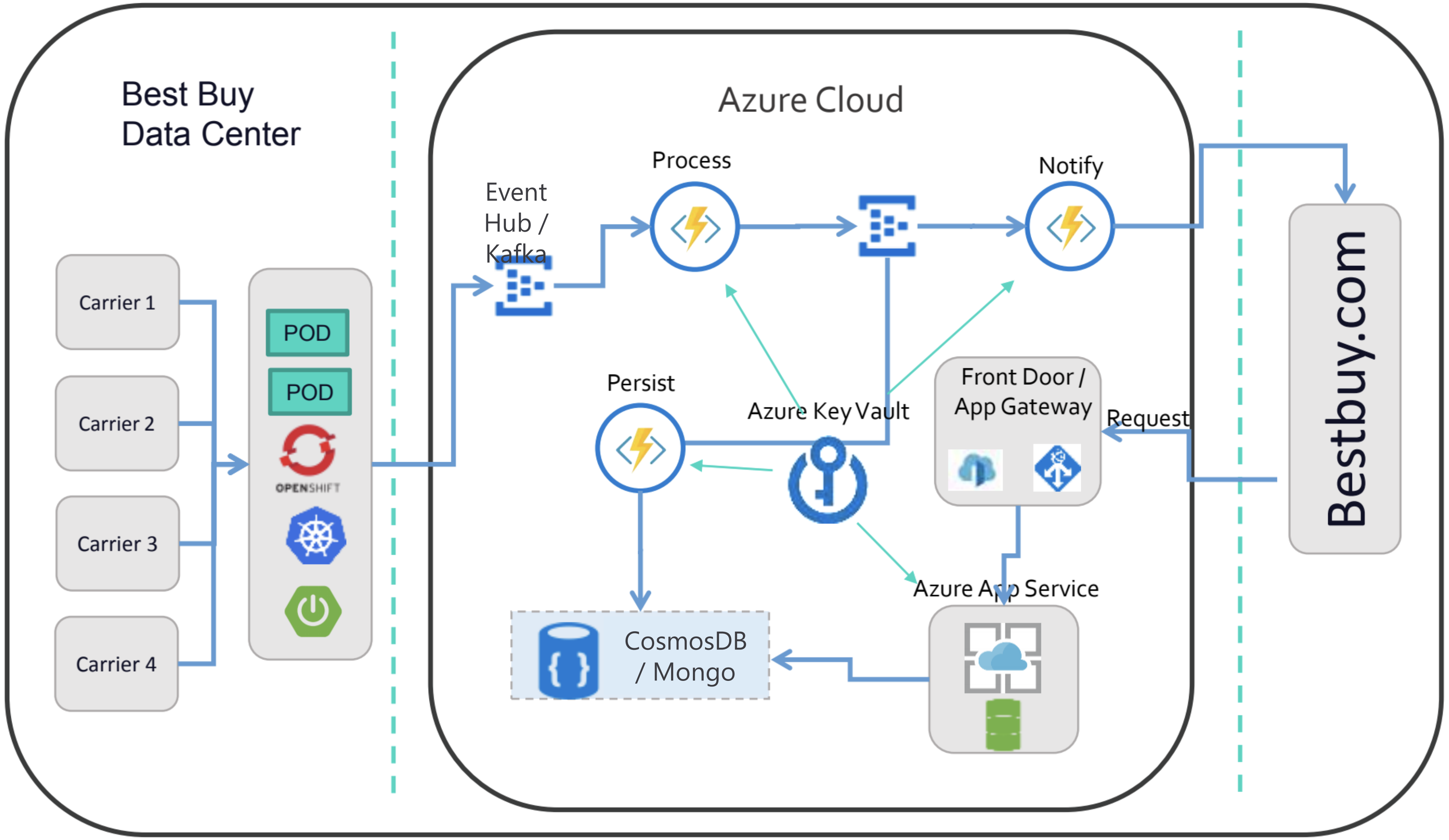
https://aka.ms/comparecompute


Migration
Cloud native
START
Greenfield or brownfield?
Brownfield
Greenfield
Lift-and-shift or cloud optimized?
Lift-and-shift
Cloud optimized
Can be containerized?
Yes
Choose from Azure container hosting options
https://azure.microsoft.com/overview/containers/
No
Web/API app? (ASP.NET, Node.js, etc)
Yes
No
App Service
Virtual Machines
Do you require full control and/or portability?
Yes
Virtual Machines
No
HPC workload?
Yes
Azure Batch
No
Microservices architecture?
No
App Service
Yes
Event-driven workload with short-lived processes?
Yes
Azure Functions
No
Needs full-fledged orchestration?
No
Container Instances
Yes
Needs .NET integration or fully supported Microsoft technology stack?
Yes
Service Fabric
No
Azure Kubernetes Service (AKS)

Note: Analyse the scenarios holistically

https://docs.microsoft.com/en-us/azure/architecture/guide/technology-choices/compute-decision-tree

- Must be able to scale to billions of events
- If an error occurs, Kafka should retry / not checkpoint
- **Ordering must be preserved**



Default Kubernetes Scaling is not well suited for Event Driven Applications

It can only react to the symptom, not the cause

KEDA

Kubernetes-based event driven autoscaling

Open source component to provide function-like scale in Kubernetes for any container

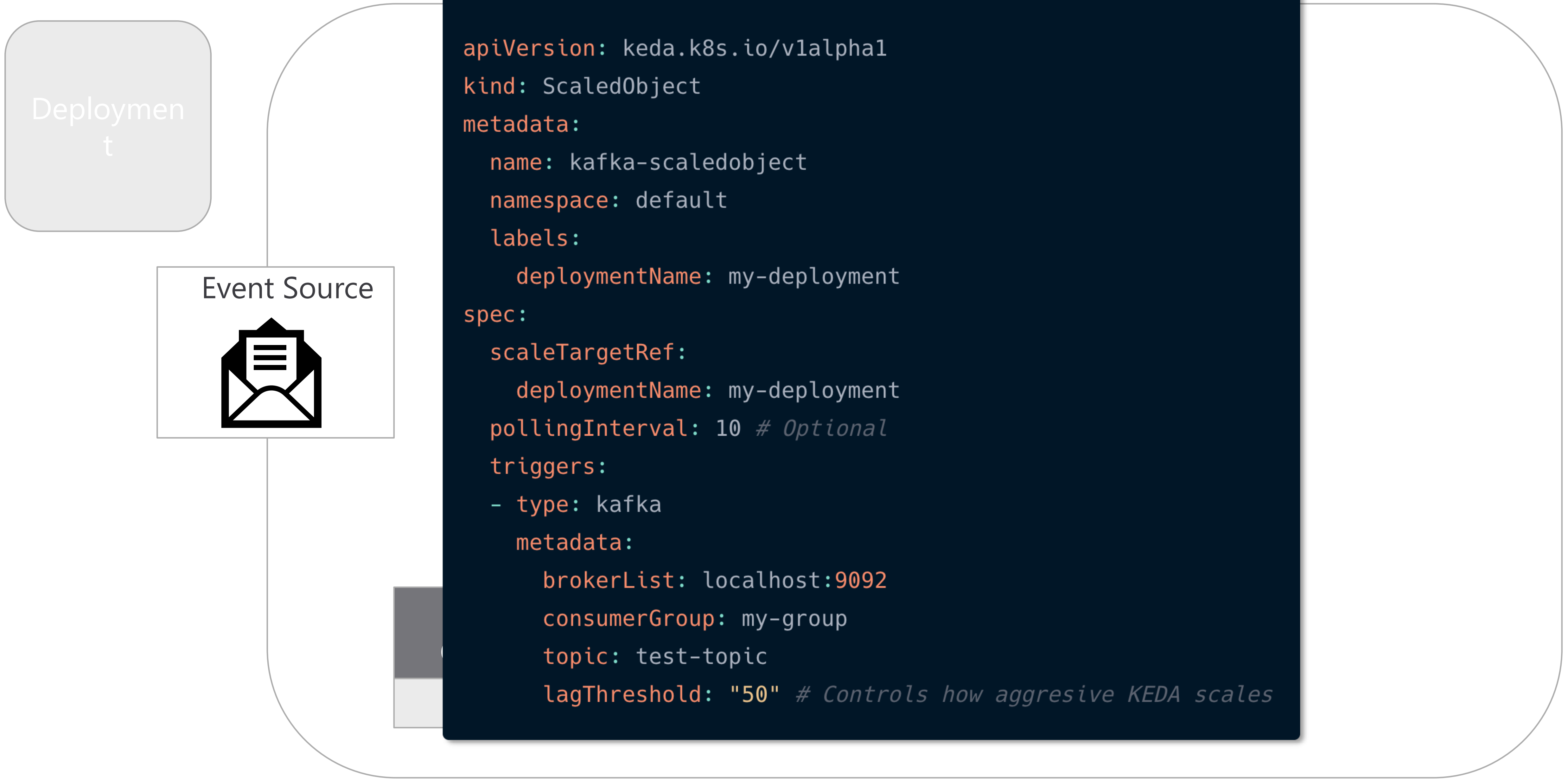
Azure Functions native tooling and trigger support

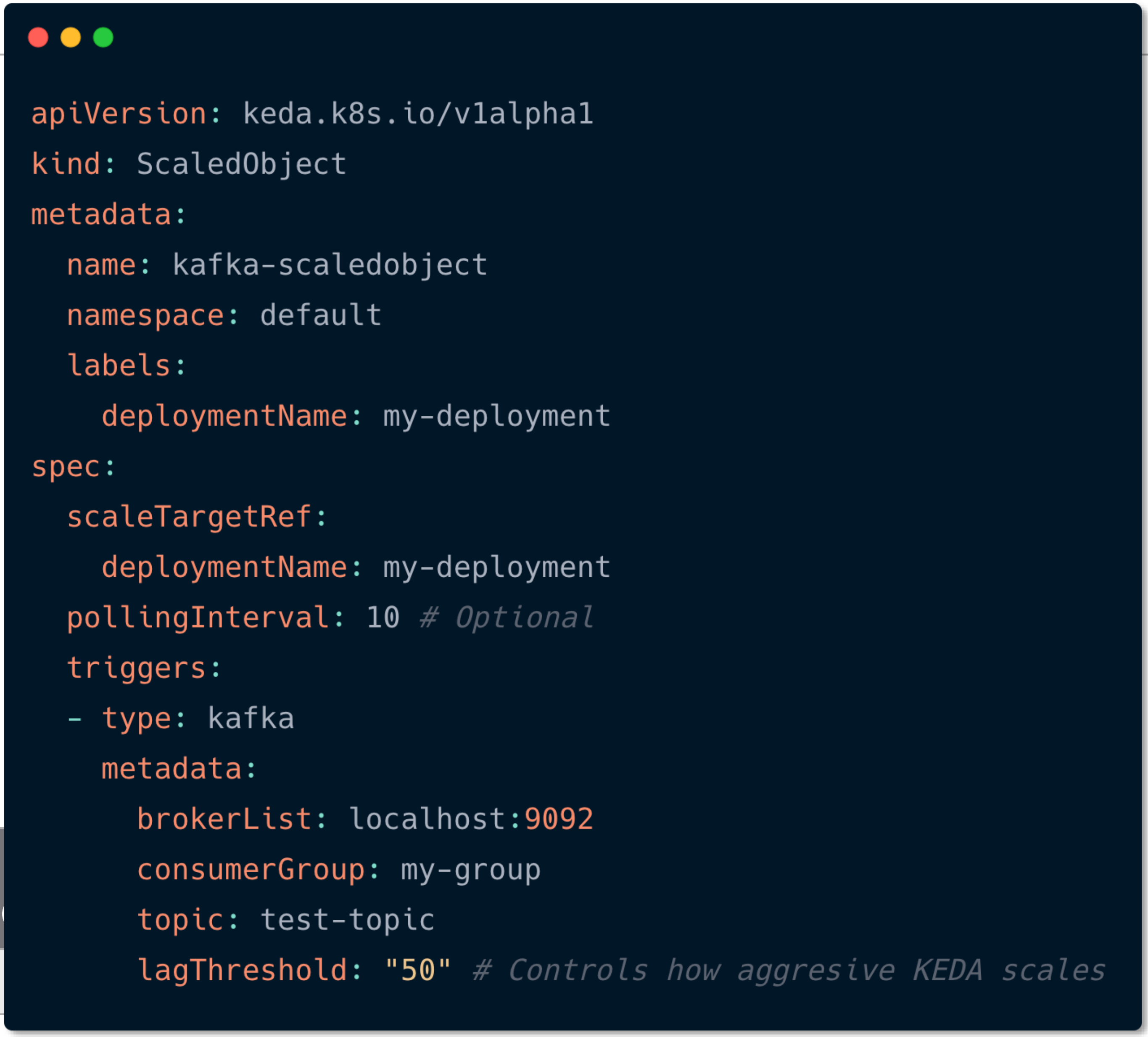
Scale to zero or scale to thousands

Ported into any cluster – new or existing

https://github.com/kedacore/keda

ScaledObject.yaml





```yaml
apiVersion: keda.k8s.io/v1alpha1
kind: ScaledObject
metadata:
  name: kafka-scaledobject
  namespace: default
  labels:
    deploymentName: my-deployment
spec:
  scaleTargetRef:
    deploymentName: my-deployment
  pollingInterval: 10 # Optional
  triggers:
  - type: kafka
    metadata:
      brokerList: localhost:9092
      consumerGroup: my-group
      topic: test-topic
      lagThreshold: "50" # Controls how aggresive KEDA scales
```

Demo

Pre-Requisites :

- Vscode
- Function core tools
- Docker

Services used :

- Azure Functions
- AKS
- Storage Account
- Azure Container Registry

https://github.com/sajeetharan/azFnKedaProcessOrder

10 rules for building on the cloud

https://aka.ms/azure-design-principles

1. Everything fails all the time - **Design for self healing**
2. Avoid single points of failure - **Make all things redundant**
3. Don't rely on other components - **Minimize coordination**
4. You're not going to right size the first time - **Design to scale out**
5. The cloud is powered by real servers - **Partition around limits**
6. It's hard to troubleshoot at 3am - **Design for operations**
7. Let us make your job easier - **Use managed services**
8. Different data has different needs - **Choose the right data store for the job**
9. The only constant is change - **Design for evolution**
10. We're not doing this just for fun, right? - **Build for the needs of business**

Microsoft

Let’s have some fun! Join Kahoot here

© Copyright Microsoft Corporation. All rights reserved.