Microsoft

Service Fabric Roadshow

CodeSlinging Cloud Native Microservices

Kevin Sauer and Lynn Orrell
Cloud App Dev & AI Architects
Intelligent Cloud Global Black Belt Team

Agenda

SSID: Microsoft
password:
SpacesOutlook

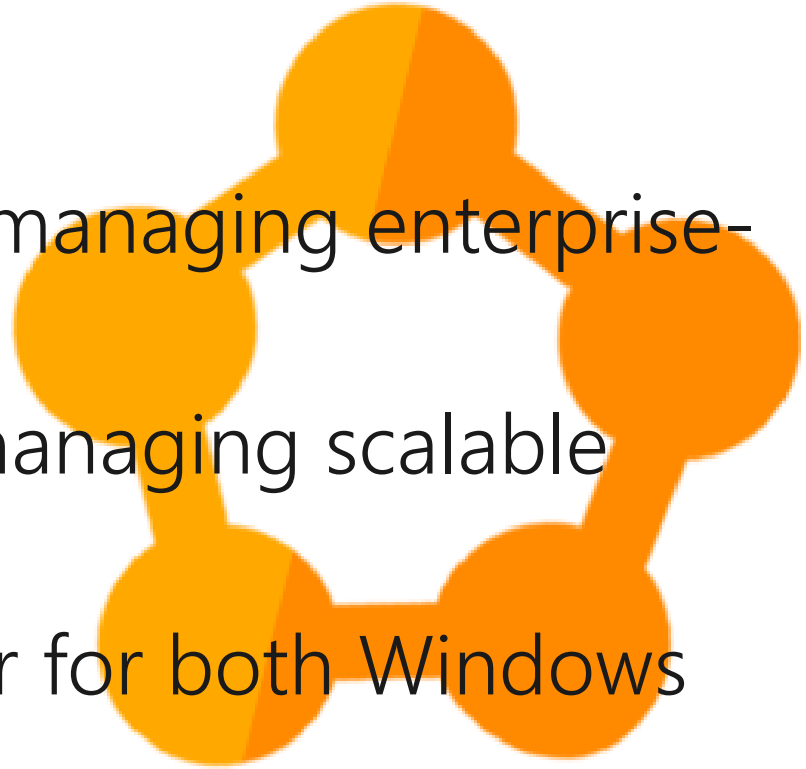
Morning

- Brief Introduction to Service Fabric
- Why and When should I use Service Fabric
- .Net Microservices in the Cloud
- Service Fabric Roadmap and Service Fabric Mesh

Afternoon

- Hackathon

What is Service Fabric?



- Middleware platform for building and managing enterprise-class, Tier-1 cloud-scale applications
- Simplifies packaging, deploying, and managing scalable microservices
- Enterprise grade container orchestrator for both Windows and Linux containers
- Designed to be scalable, reliable, and manageable
- Designed to allow developers and administrators to avoid solving complex infrastructure problems and focus on implementing mission-critical, demanding workloads

Azure Service Fabric: Microservices platform

Build, deploy, and operate applications as microservices, using any OS, at any scale, on any cloud

Build

Build new or modernize existing applications

Deploy

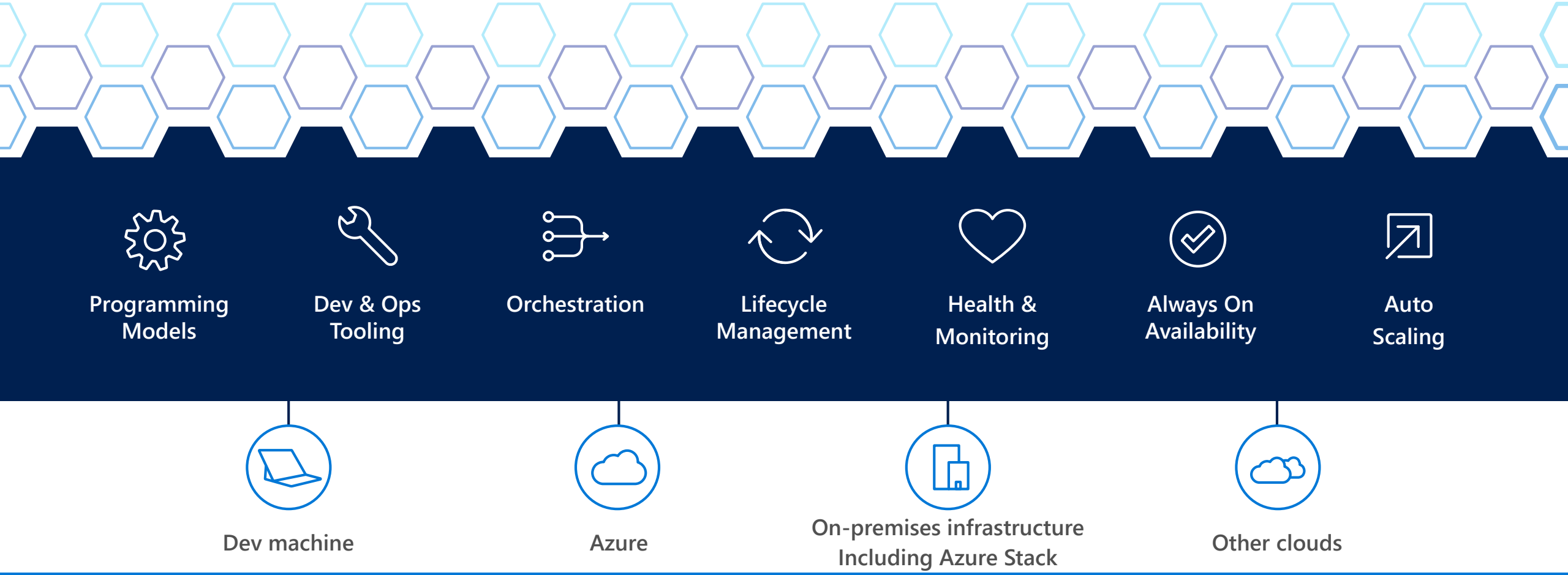
Deploy, run and orchestrate any code including containers

Operate

Manage and secure applications reliably at any scale

To build, deploy, and operate...

...containers or microservices on any OS on any cloud



Build: data-aware microservices

Programming Models

Dev & Ops Tooling

Orchestration

Lifecycle Management

Health & Monitoring

Always On Availability

Auto Scaling

Reliable Actors

Maximize uptime and scalability with stateful and stateless Actors

Reliable Services

Manage state reliability without a database, lowering latency. Use stateless services to write data to eternal data stores

Guest Executables

Run any existing code

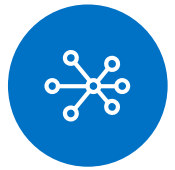
Containers

Orchestrate your Windows Server or Linux containers reliably at scale

.NET, or Java

Best platform for .NET and ASP.NET core with Visual Studio 2017 and Java using Eclipse and Jenkins

Deploy: any code on any OS



Programming Models

Dev & Ops Tooling

Orchestration

Lifecycle Management

Health & Monitoring

Always On Availability

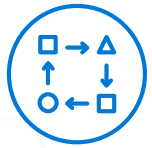
Auto Scaling

CI/CD

Use familiar tools: Visual Studio + Team Services for .NET or Jenkins + Yeomen for Java

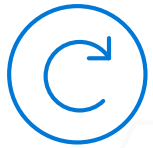
Docker Compose and Azure Container Instance

Orchestrate existing container applications or deploy single containers



Automate

Deploy or remove applications using PowerShell, CLI, Visual Studio, and other APIs



Rolling upgrades

Upgrade non-disruptively and roll-back in case of failures, automate with PowerShell

Monitor and diagnose

Generate, aggregate, and analyze events with built-in tooling and integration with Azure services such as AppInsights and OMS

Operate: on any cloud at any scale

Programming Models

Dev & Ops Tooling

Orchestration

Lifecycle Management

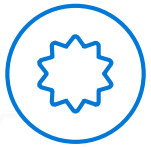
Health & Monitoring

Always On Availability

Auto Scaling

Use familiar tools

Such as Splunk, OMS, ELK, or AppInsights to gain deep insights or monitor application health



Use controlled chaos

Test graceful and ungraceful failure scenarios

Recover gracefully

Recover from machine or service failure gracefully; replicate data automatically

Secure at scale

Secure node-to-node communication and management access

Automated Platform Upgrades

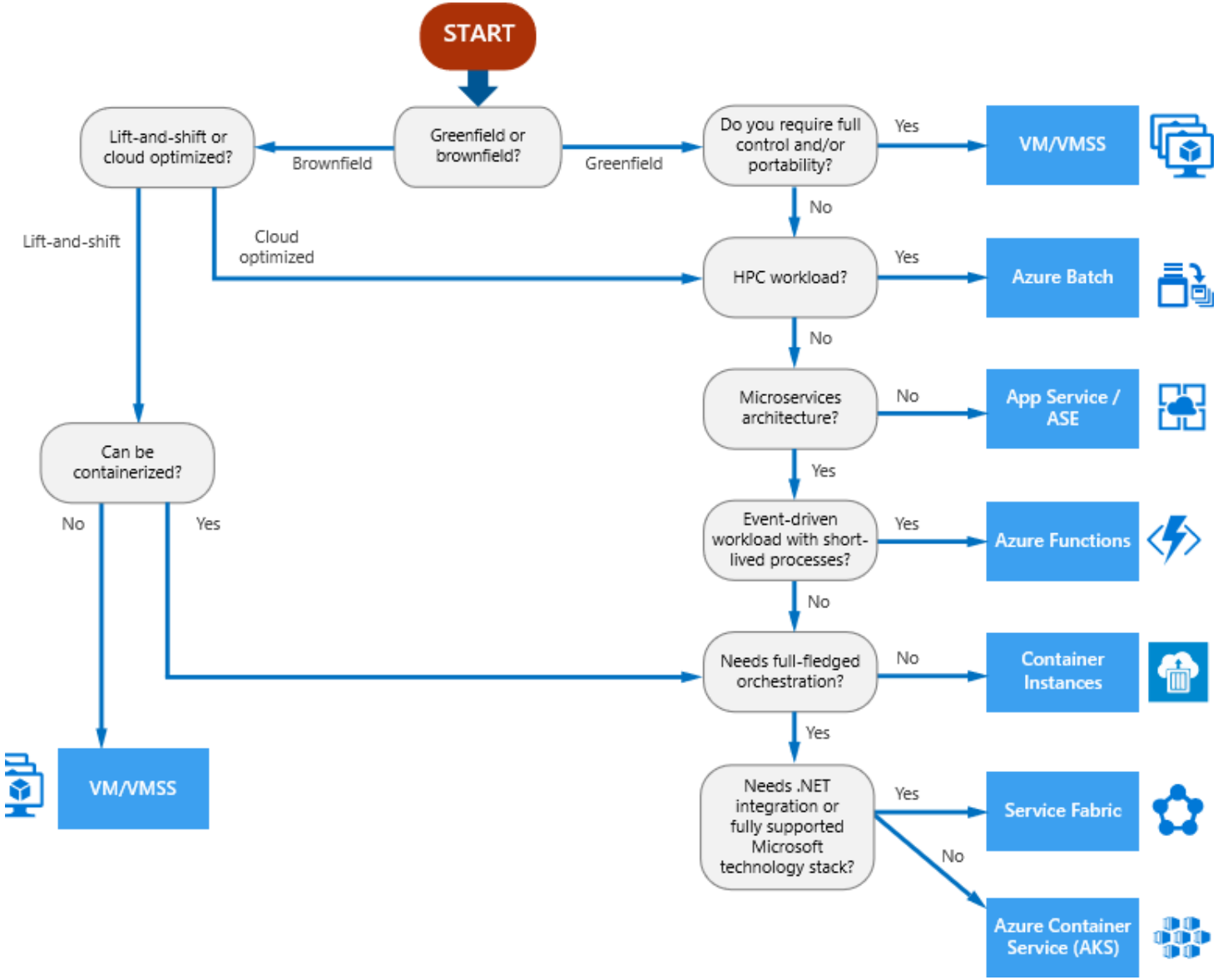
Service Fabric runtime and OS upgrades are automated for you or can choose to schedule them

When and Why would you use Azure Service Fabric?

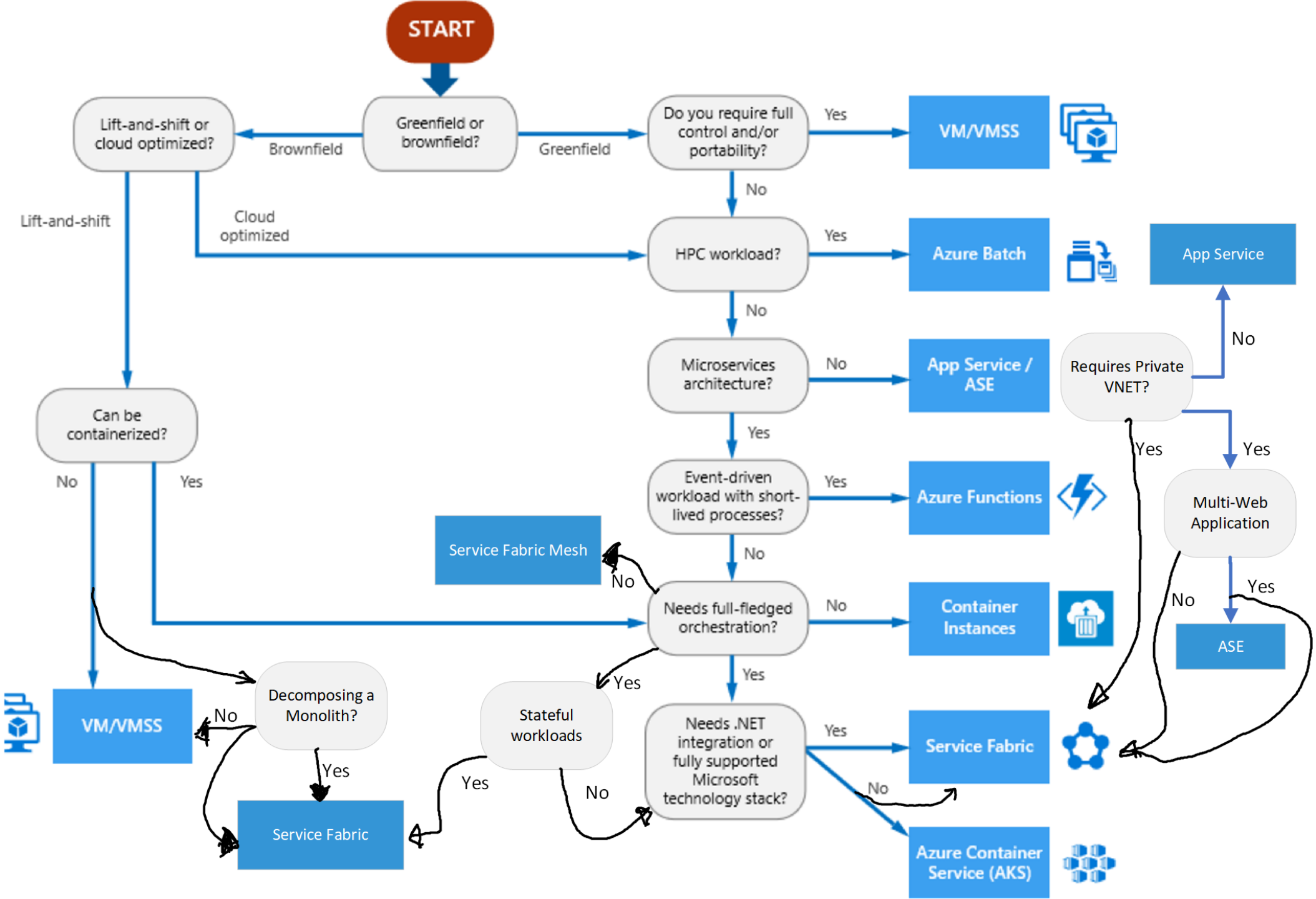
Compute Decision Tree

Compute Services covered:

- VM
- VMSS
- Azure Batch
- App Service
- App Service Environment
- Azure Functions
- Container Instances
- Service Fabric
- Azure Kubernetes Service



Compute Decision Tree v2



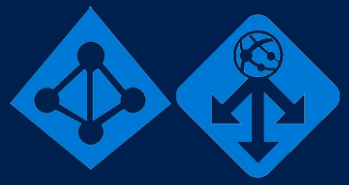
Scenarios powered by Service Fabric

Empowering customers of all sizes to achieve more

Lift & shift including containers

Artificial Intelligence Applications

Mission-critical business SaaS



Multi-tenet Applications

IoT data processing

Enterprise Microservice / API Catalog

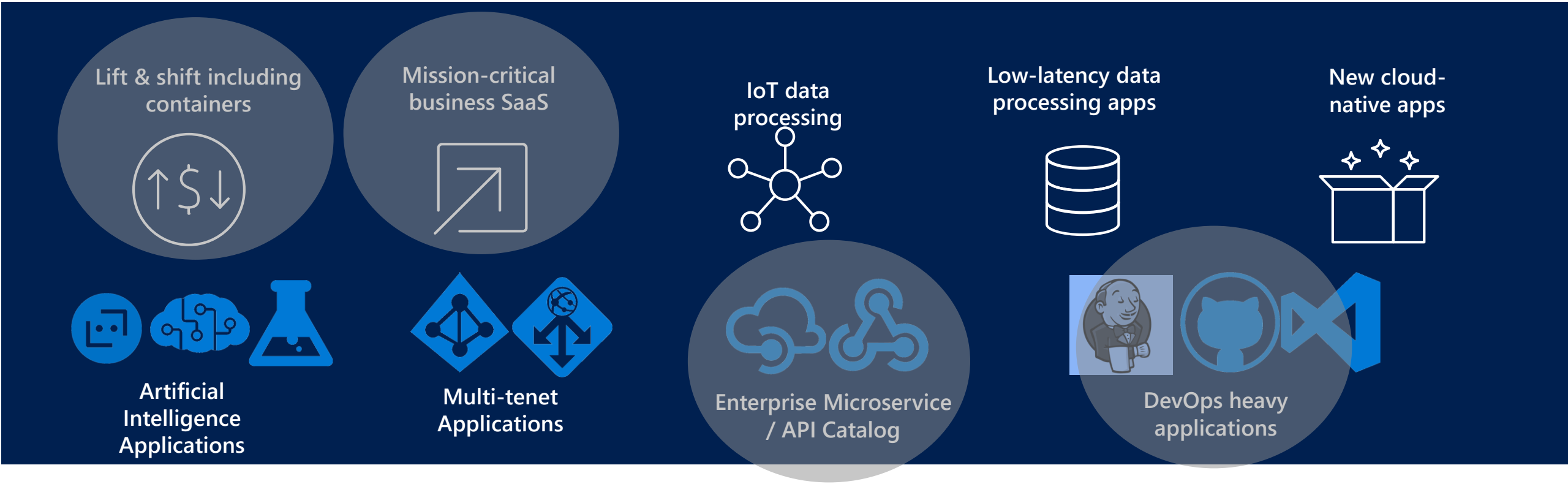
Low-latency data processing apps

DevOps heavy applications

New cloud-native apps

Scenarios powered by Service Fabric

Empowering customers of all sizes to achieve more



Lift and Shift (with and without Containers)

5 stages in a continuum…

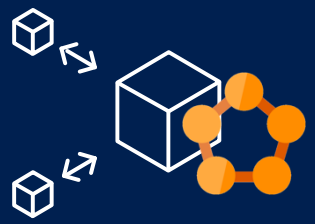
1

Traditional application

2

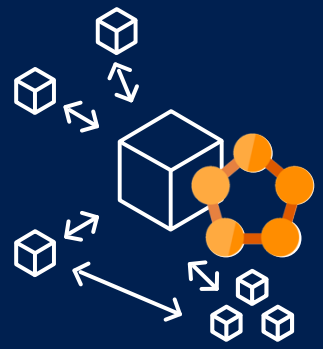
Traditional application hosted as guest executables or containers

3



Traditional application + new microservices

4



Parts of existing traditional application extracted

5

New or transformed microservices application

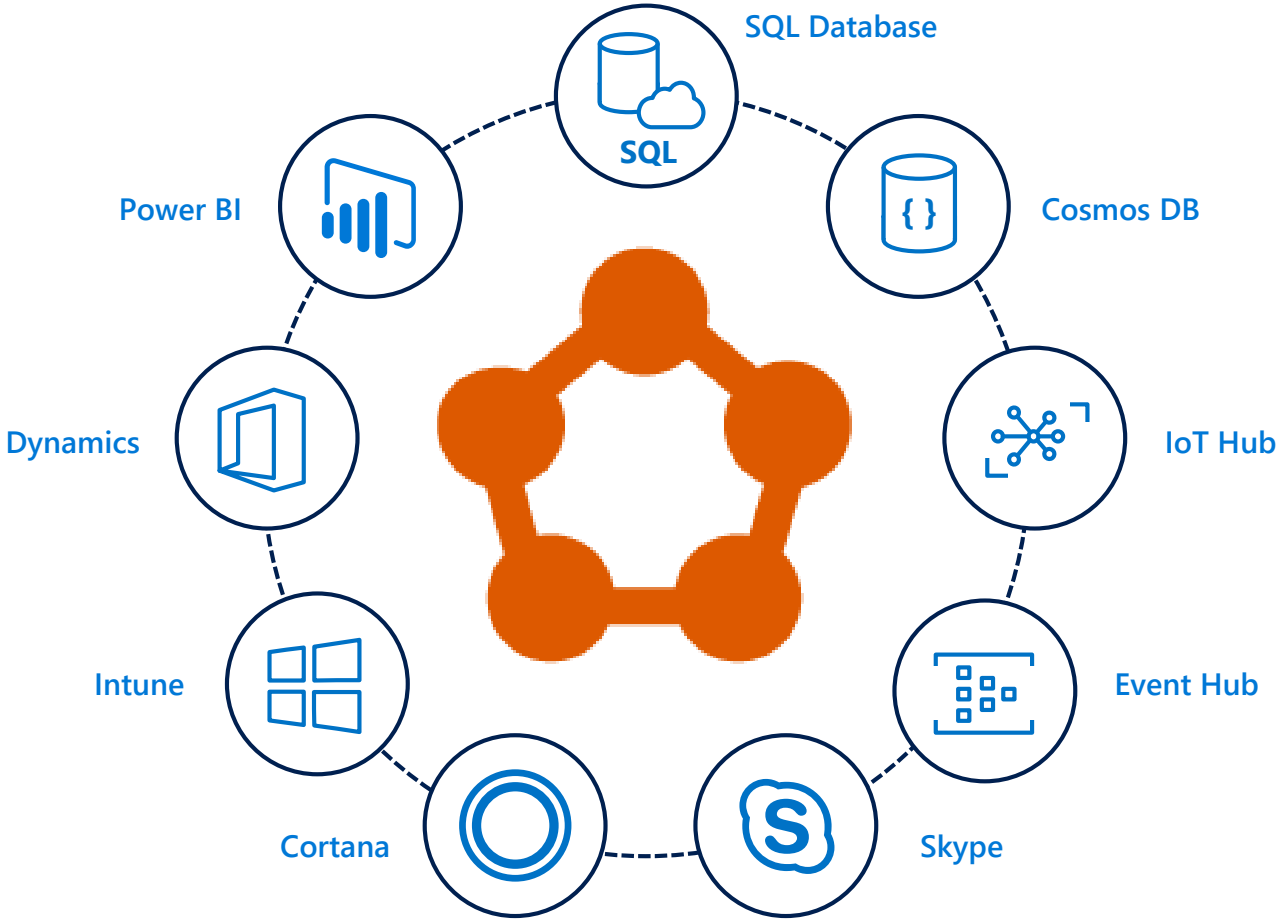
… Service Fabric supports any stage you choose

Business Critical Applications

Customer Profile:

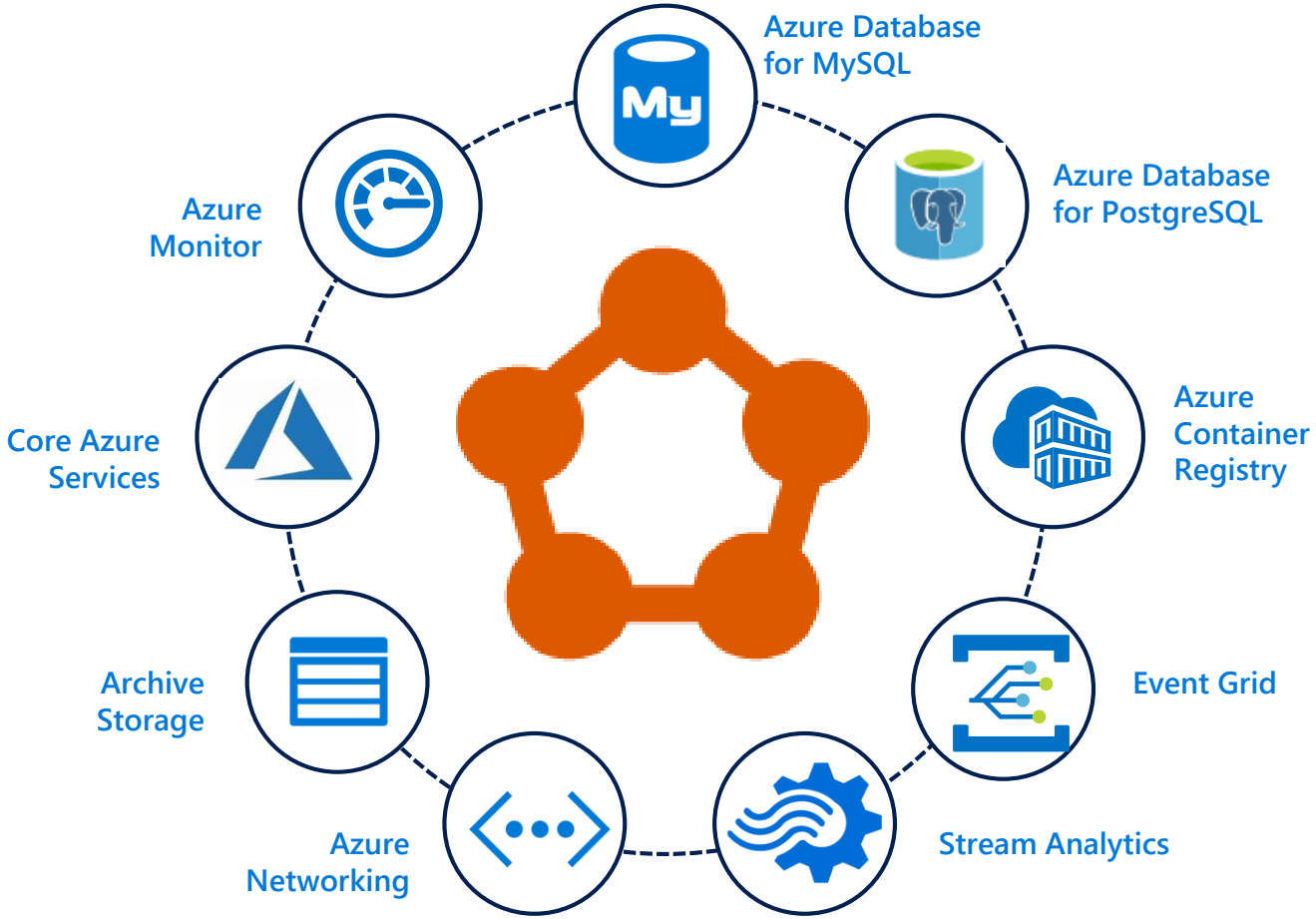
Microsoft

- Microsoft has **deep expertise** in running global services such as Cortana, Skype & Cosmos DB
- Service Fabric is the **foundational technology** powering these services & core Azure infra
- Sample scale of one of these services**: 60 billion events per day** with millions of databases
- About **30%** of Azure's cores run Service Fabric directly.
- Service Fabric Manages **Millions of Service upgrades** weekly within Microsoft alone
- May 23, 2018 – 2.0066 Trillion requests to Azure Event Hubs (from Dan Rosanova via Twitter)



Designed for mission critical tier 1 workloads

Microsoft



Business critical application customer use cases

MAERSK

Honeywell
THE POWER OF CONNECTED

accenture

THOMSON REUTERS

Wolters Kluwer

asos
discover fashion online

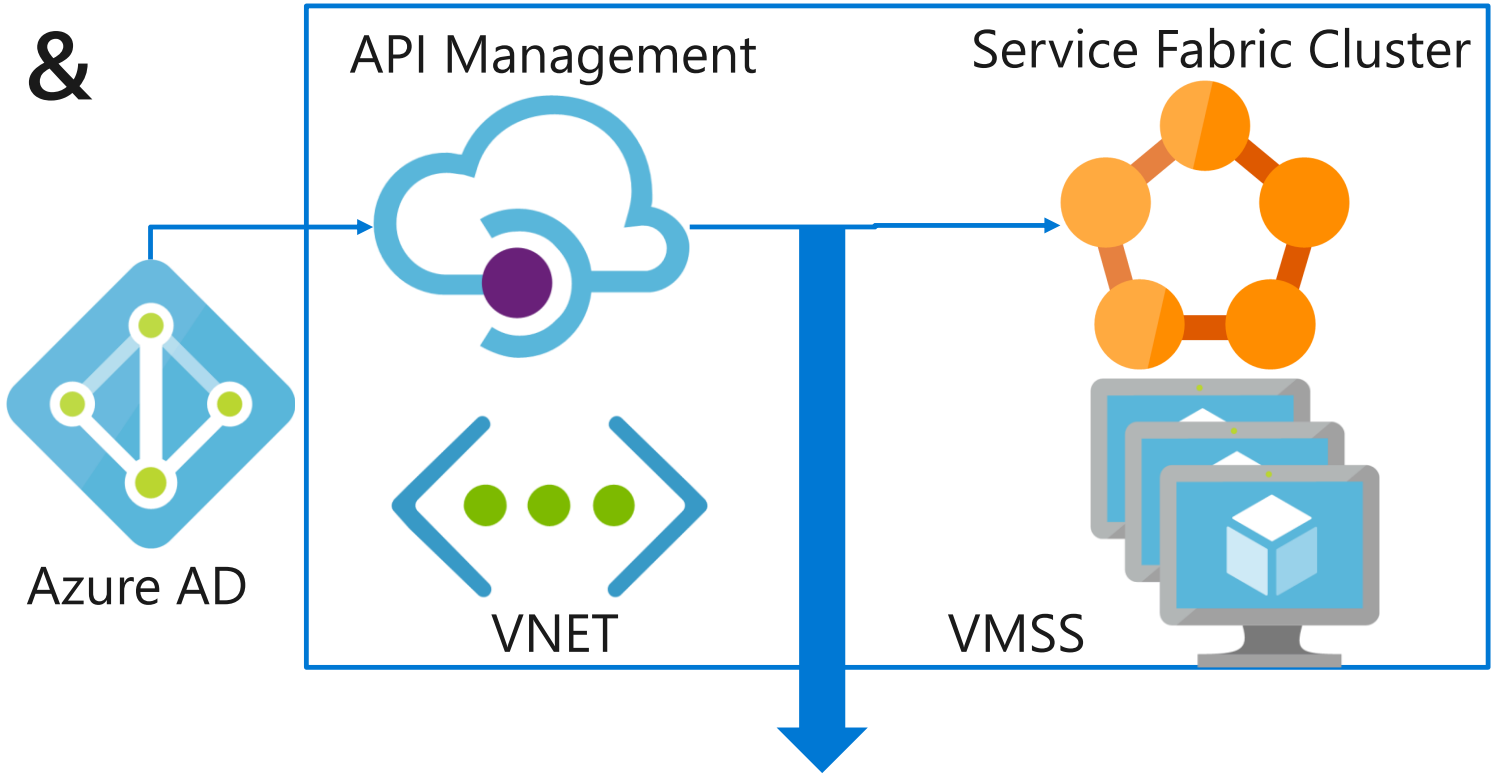
The Ultimate Driving Machine

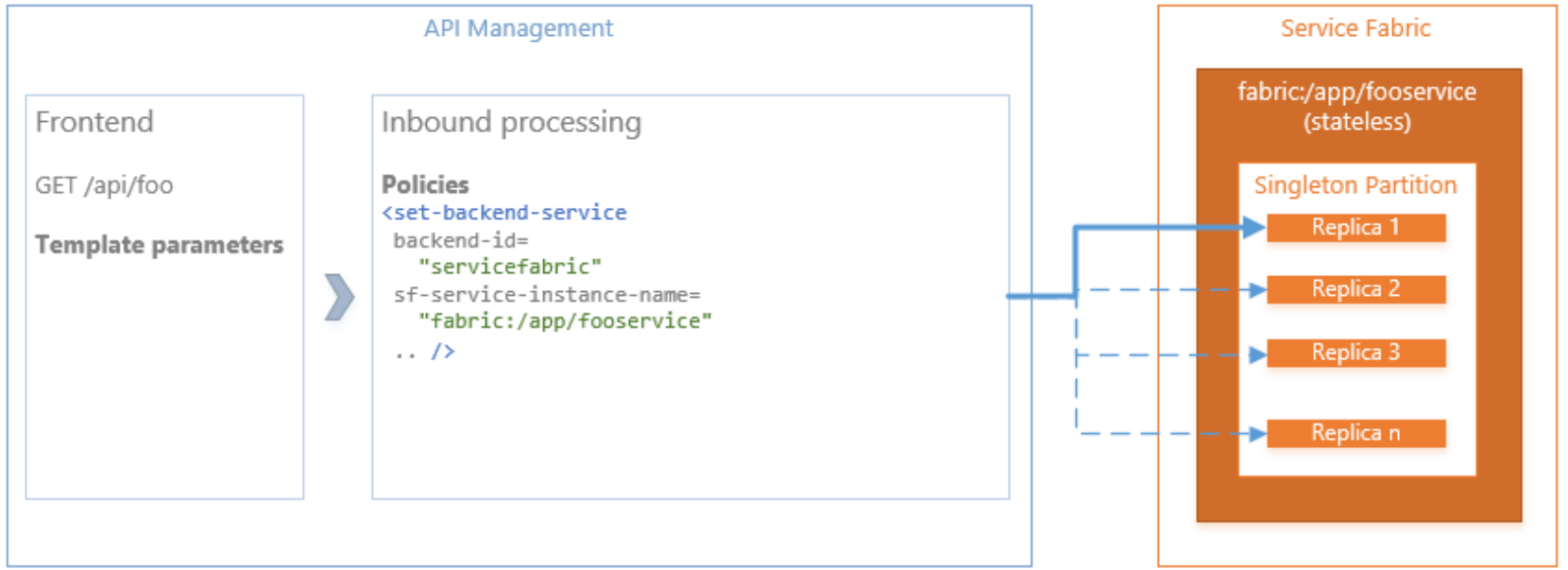
Alaska AIRLINES

OSIsoft

Enterprise Microservice & API Catalog

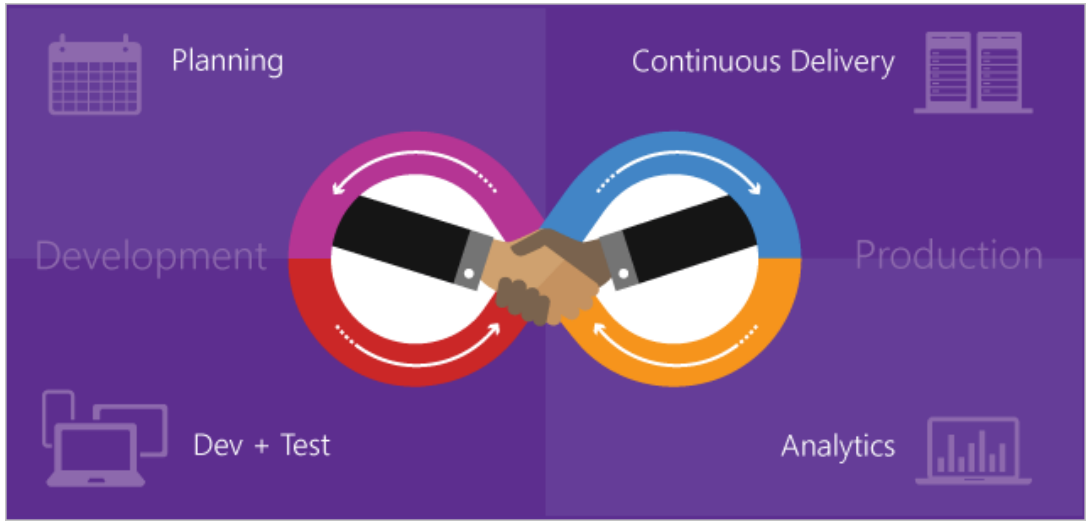
- API Management & SF on the same VNET
- Internal & Externally based API users (devs)
- APIM native integration with SF (no LB)
- Density on the VMSS
- Stateful, Stateless, Container based APIs on SF
- Extremely rich ASP.Net Core Stateful services





DevOps Heavy Applications

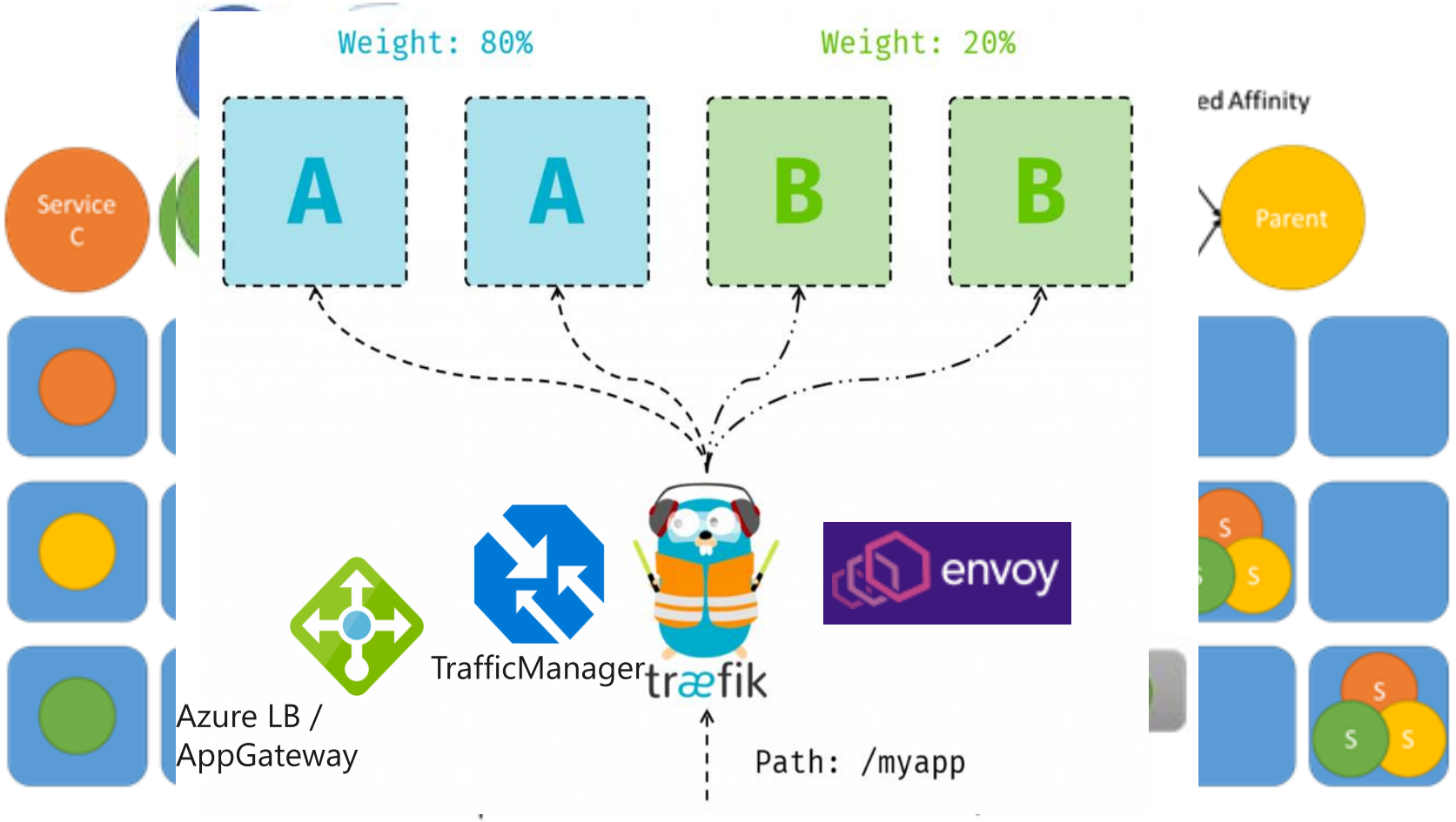
- Cross cutting concern, can be across all of the previously mentioned workloads
- SF is agnostic to the CICD technology
- Several release options (Rolling upgrade, VIP Swap, Controlled migration, etc)
- Service Fabric handles the runtime and underlying resource updates



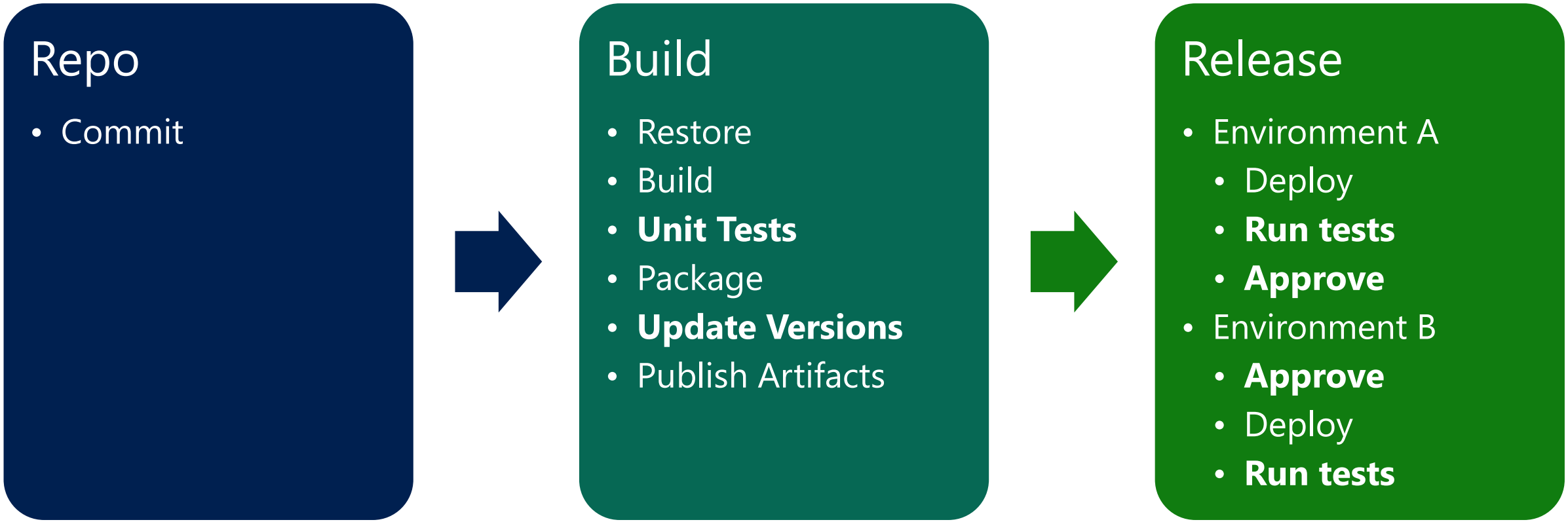
Mr. Donovan Brown: #RubDevOpsOnIt

Key Service Fabric DevOps Features

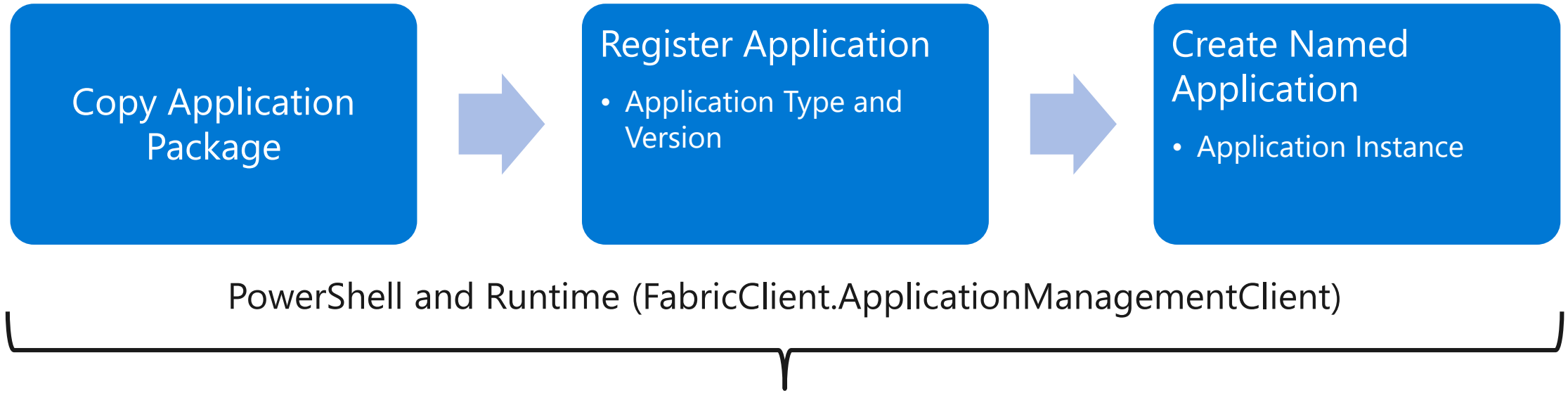
- Service Affinity
- Node Placement
- Multi-version, Blue-Green
- Chaos testing
- Open REST API, PowerShell, CLI



Service Fabric Build and Release Pipeline Example



Publish a Service Fabric Application



SDK → Publish-NewServiceFabricApplication.ps1

Visual Studio Project → Deploy-FabricApplication.ps1

Visual Studio Team Services → Deploy Service Fabric Application

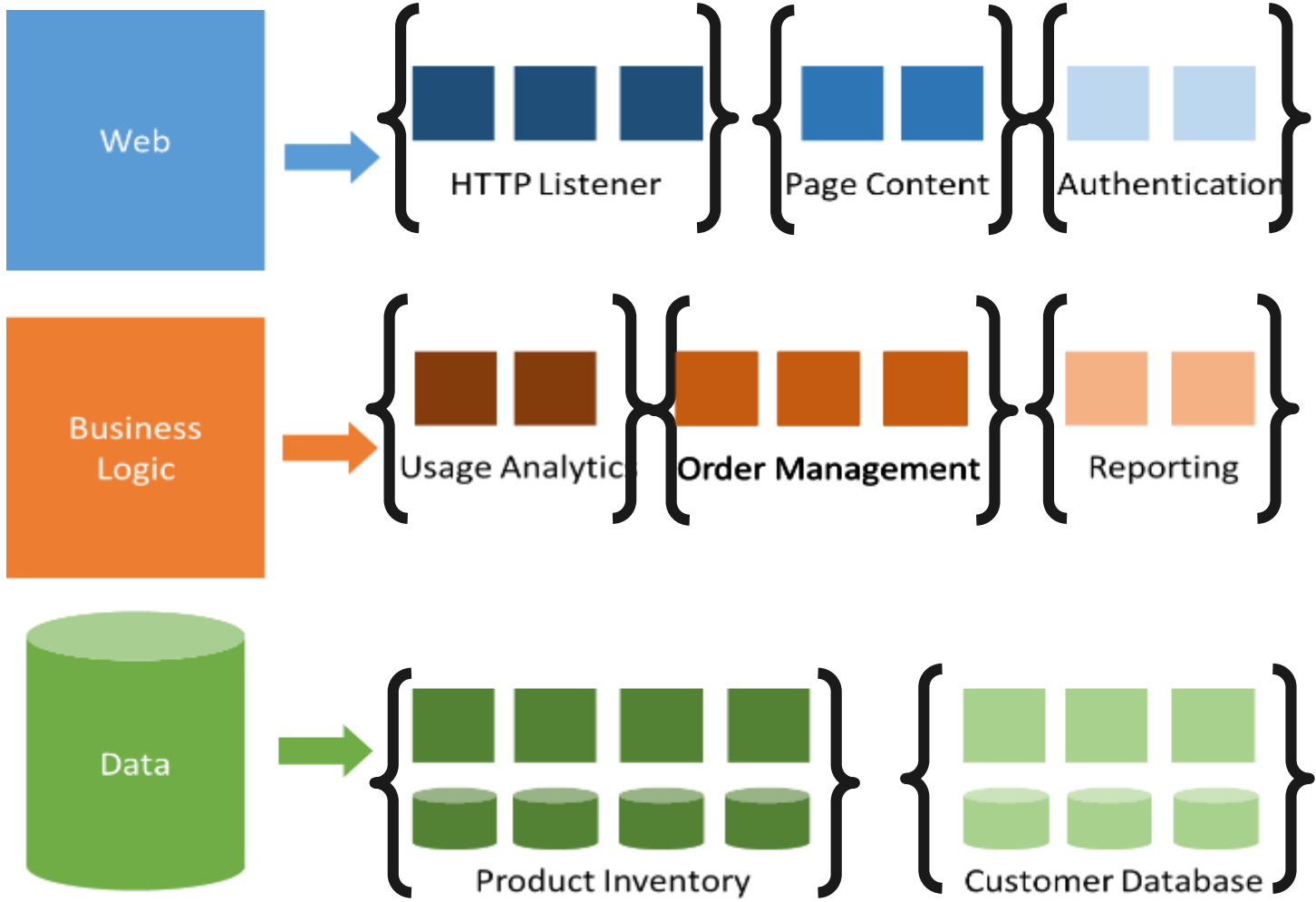
Other services that compliment SF

- **Gateways**: IOT Hub, Event Hub, API Management, Azure Load Balancer, AppGateway
- **Messaging**: Service Bus, Event Hub, IOT Hub, Stream Analytics
- **Persistent Storage**: Redis Service(cache), CosmosDB, SQL Azure DB, Azure Database for MySQL, Azure Database for PostGRES, Azure Data Lake Storage
- **Web front end**: App Services, App Service Environment
- **Bursting workloads**: Azure Functions
- **Integration**: Azure Logic Apps
- **Reverse Proxy**: Traefic, Envoy
- **Monitoring**: OMS, Application Insights
- **Networking / Traffic management**: Traffic Manager, Load Balancer, AppGateway

.Net Microservices

Microservices Lightning Fast Refresher

- Small, autonomous services
- Self Contained – single work unit
- Loosely coupled
- Independent deployments
- Persist their own data
- Communicate via API
- Can be independent tech stacks
- Containers != Microservices
- SOA != Microservices
- Distributed Object Oriented Programming



Distributed System Architecture Best Practices

- Design for self healing
- Make all things redundant - avoid having single points of failure
- Minimize coordination between application services to achieve scalability
- Design to scale out - scale horizontally, adding or removing as demand requires
- Partition around limits - to work around database, network, and compute limits.
- Design for operations – use the right tools for the operations team
- Use managed services - use PaaS rather than IaaS
- Use the best data store for the job
- Design for evolution - change is key for continuous innovation
- Build for the needs of business

An easier, battle-tested approach

Rolling Upgrades

Availability Guarantees

Scale Out Architecture

Resource Governance

Density

Packaging & Deployment

Policy Enforcement

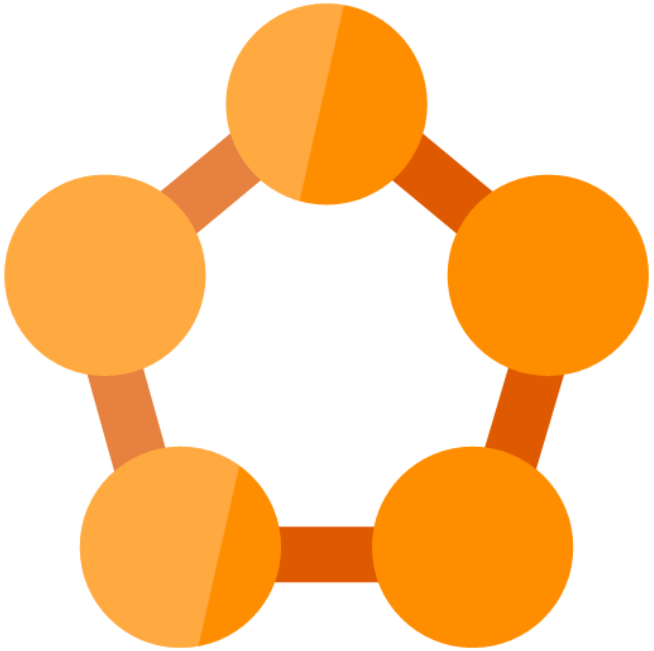
Granular Versioning

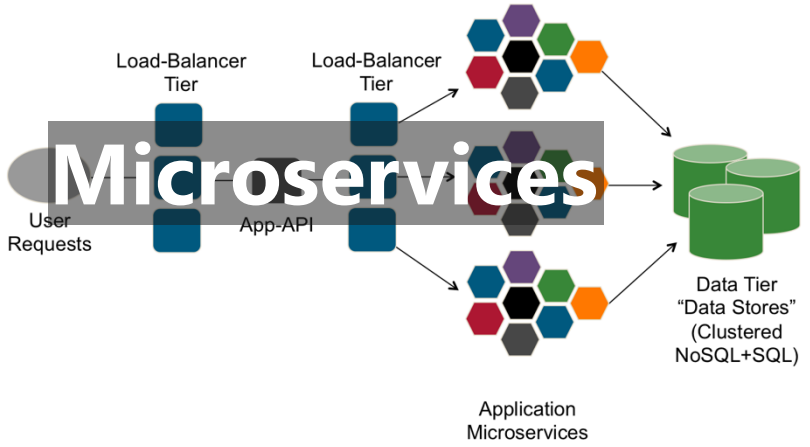
Stateful Workloads

Leader Election

Service Fabric

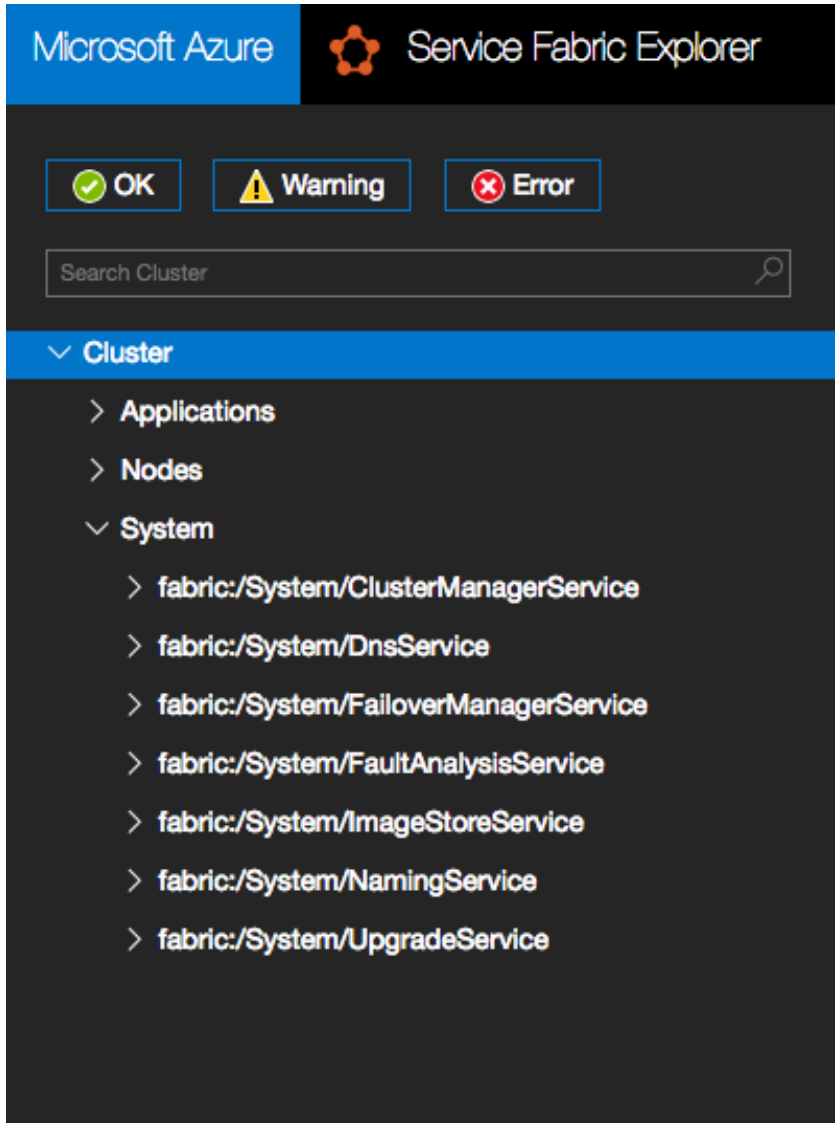
Service Fabric is the best choice for .Net microservices and Windows Container orchestration





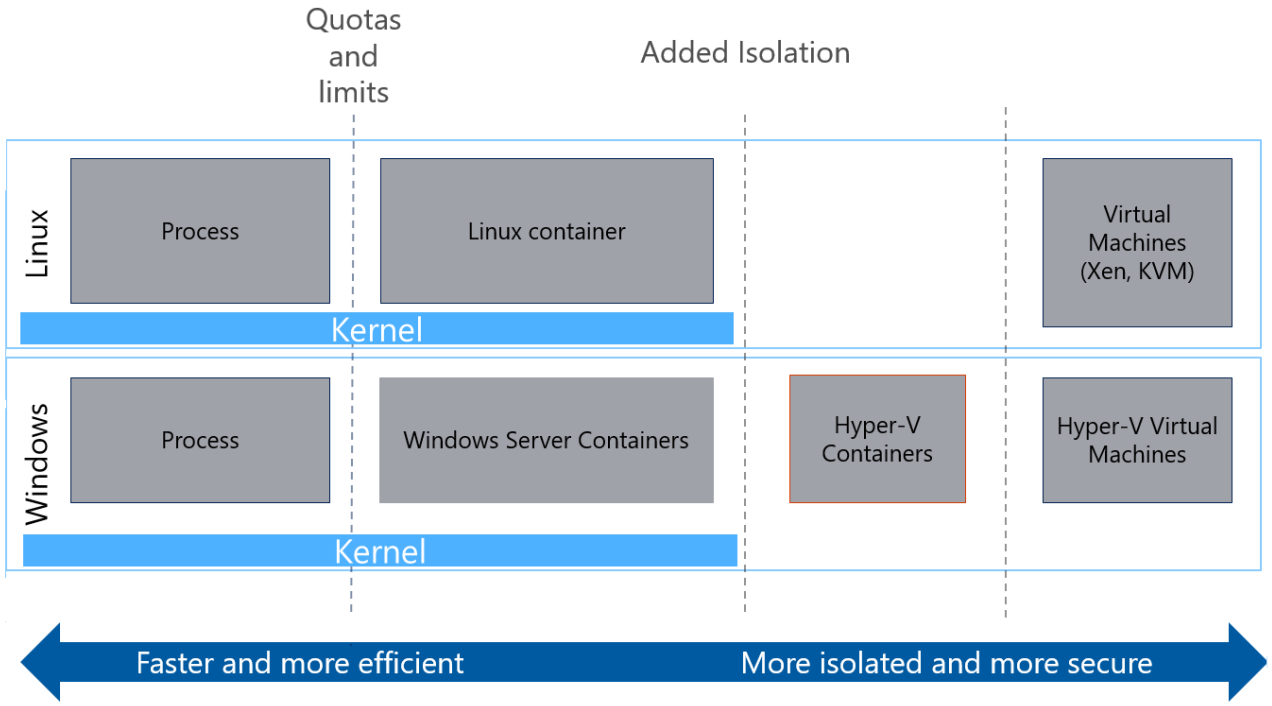
Service Fabric Microservice Architecture

- Service Fabric is a microservices based platform
- Developer support - deep Visual Studio integration
  - Local IDE Debugging
  - Local Cluster support
  - Log Streaming via Cloud Explorer
  - Project Templates
  - Deployment from Visual Studio
- VSTS pipeline integration
- Container, Guest Exe, Microservices model mixing
- Additional programming models like Reliable Actors

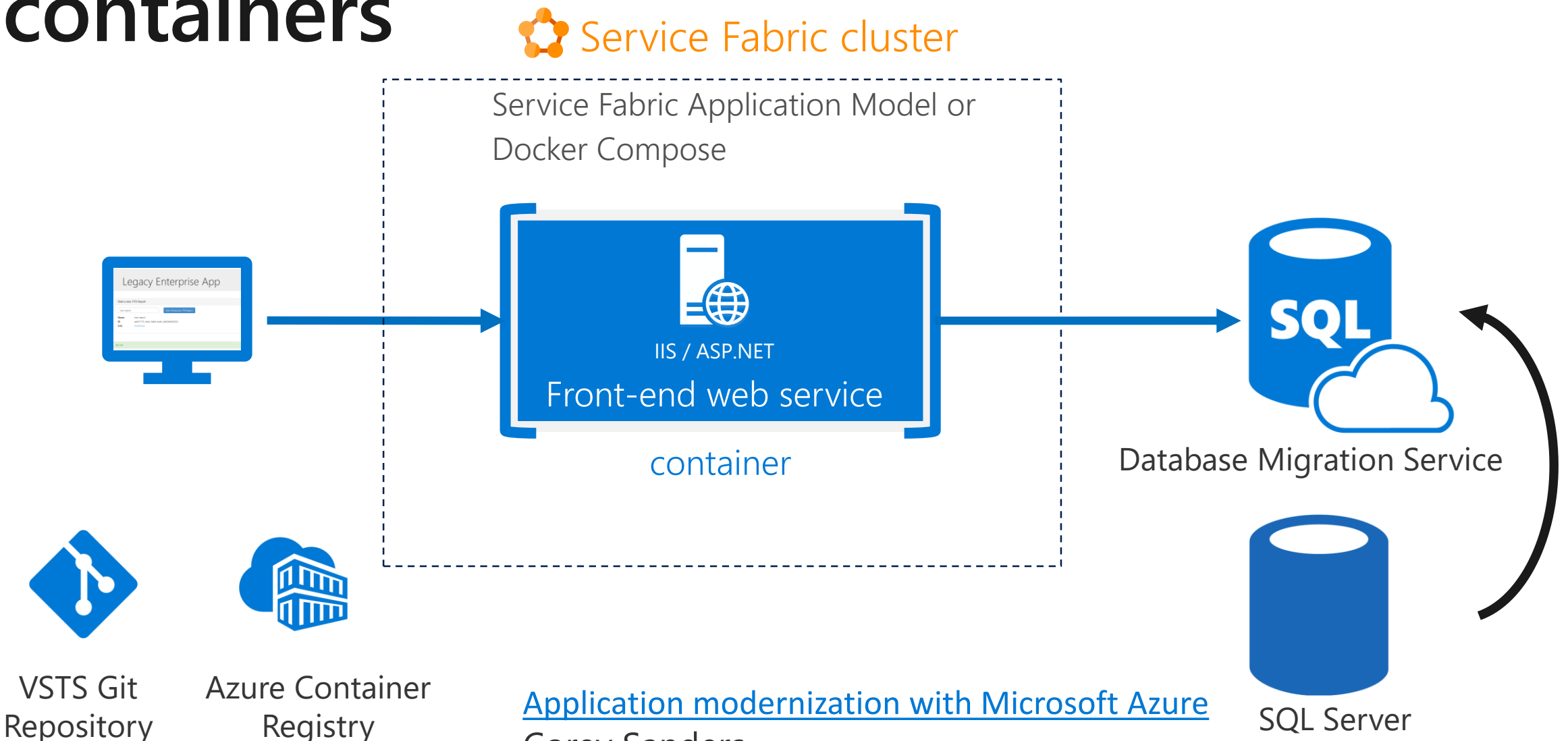


Windows Containers

- Only orchestrator with Native Support for Windows Containers
- Docker Compose, JSON (ARM)
- Visual Studio tooling, templates, debugging
- Resource Governance
- Windows Server Containers and Hyper-V Containers
- IP per Container, Container Groups, VPN Per App
- Stateful containers
  - Service Fabric local storage volume driver
- Dynamic Placement, Load Balancing, DNS



“Lift and Shift” existing applications to containers

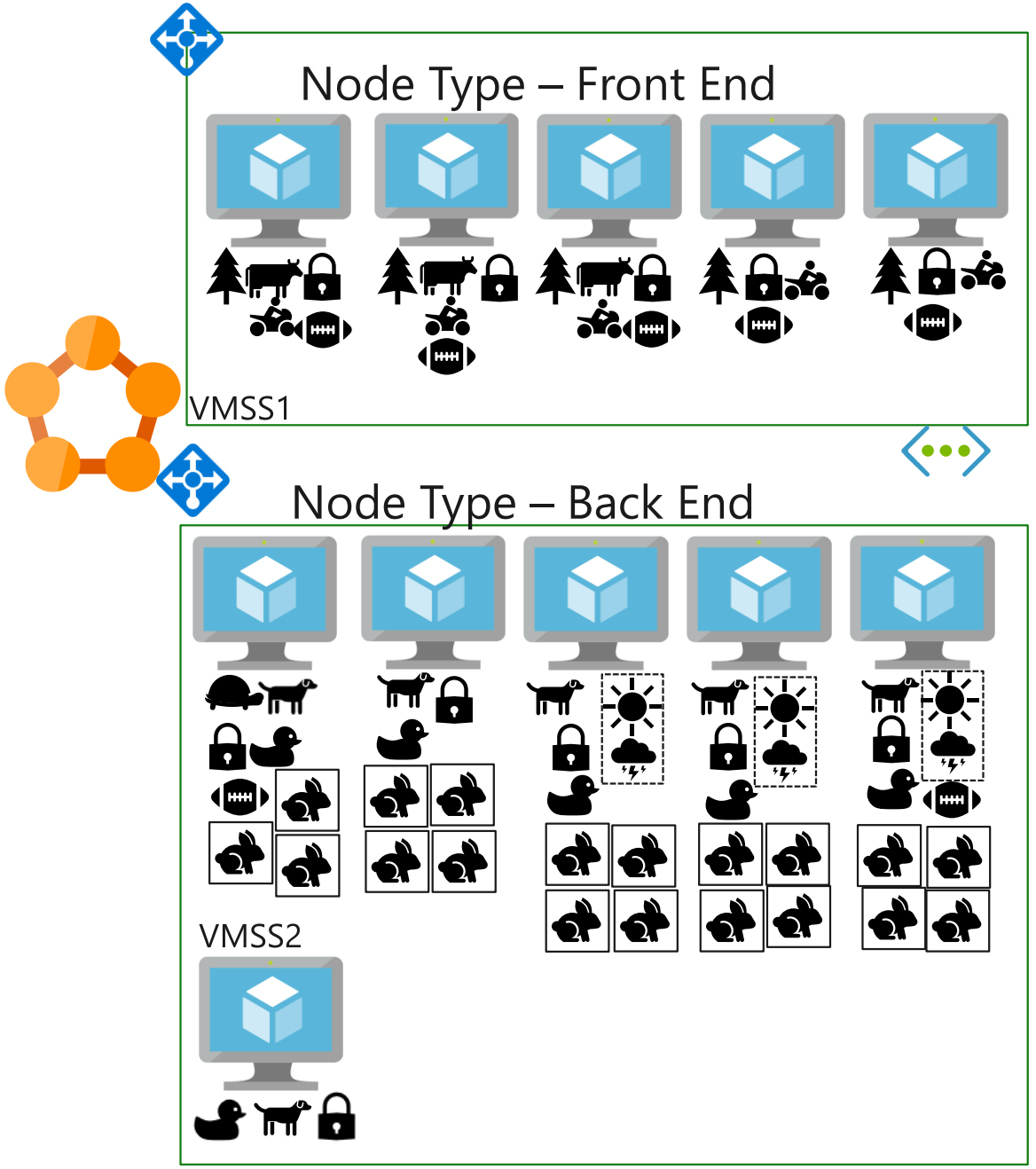


[Application modernization with Microsoft Azure]

Corey Sanders

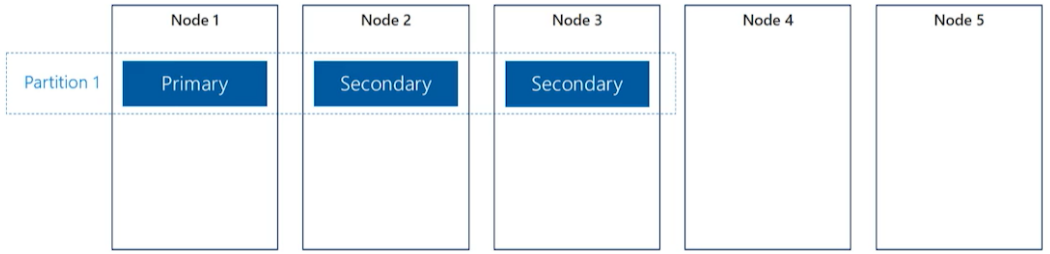
Stateless Services

- Native .Net Microservices (with and without containers)
- Web Services
- API Services
- Worker Processes that are listening to a messaging bus
- Transaction processing services
- Independent scale
- Independent deployments
- High density

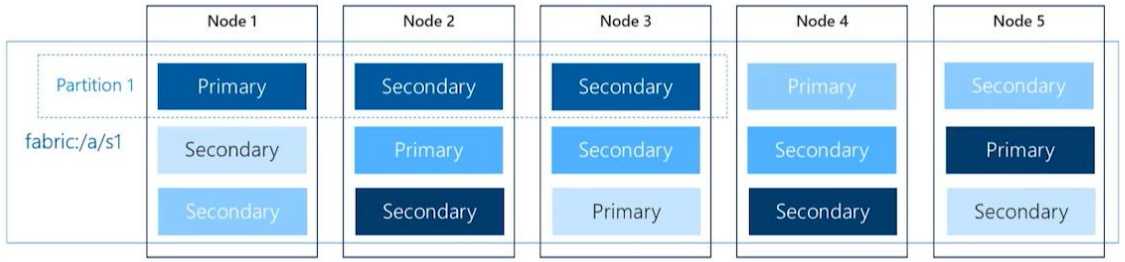


Stateful Services

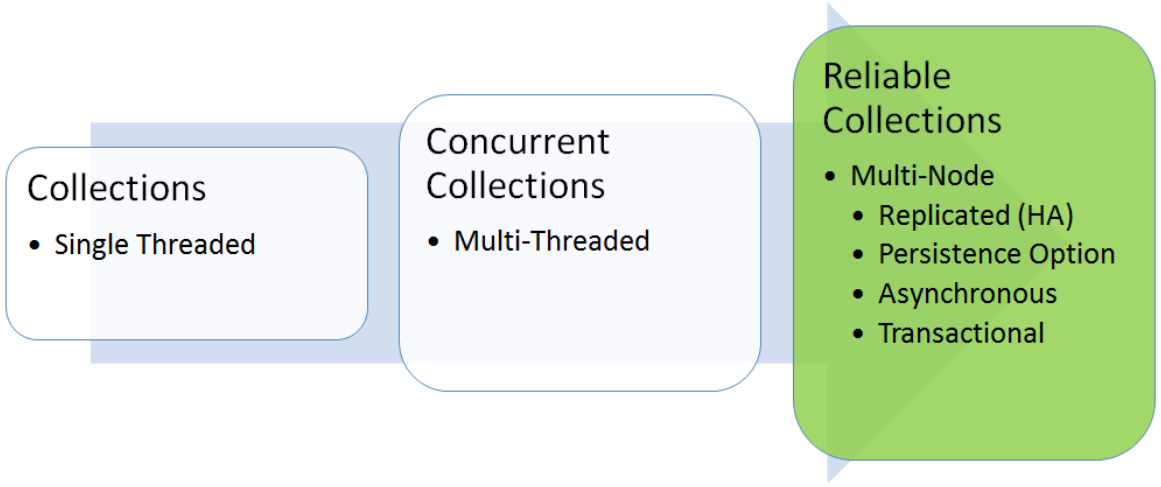
- Low latency reads and writes in native language
- Built in partitioning and data replication
- ASP.Net Core Stateful services
- Caching layer without volatility
- Pro tip: Data Contract Serialization
- Architecture tip: Partition with backing data store
- Moving to Nuget instead of SDK
- Lots of options for programming (queues, dictionaries)




```
sfctl service create --name fabric:/a/s1 --stateful --int-scheme-count 1 --target-replica-set-size 3 …
```




```
sfctl service create --name fabric:/a/s1 --stateful --int-scheme-count 5 --target-replica-set-size 3 …
```



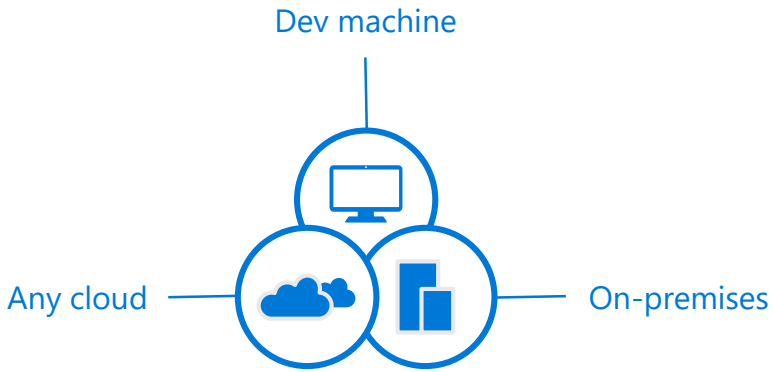
Service Fabric Roadmap + Service Fabric Mesh

"The most effective way to do it, is to do it."
Amelia Earhart

Service Fabric is powerful

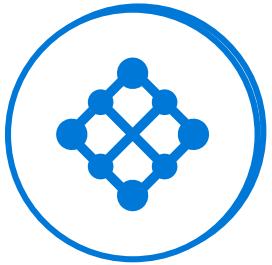
…. But it can be complex and the learning curve can be steep

Azure Service Fabric Mesh



Service Fabric Standalone

Bring your own infrastructure



Azure Service Fabric Mesh

Fully managed Azure hosted microservices platform

Service Fabric

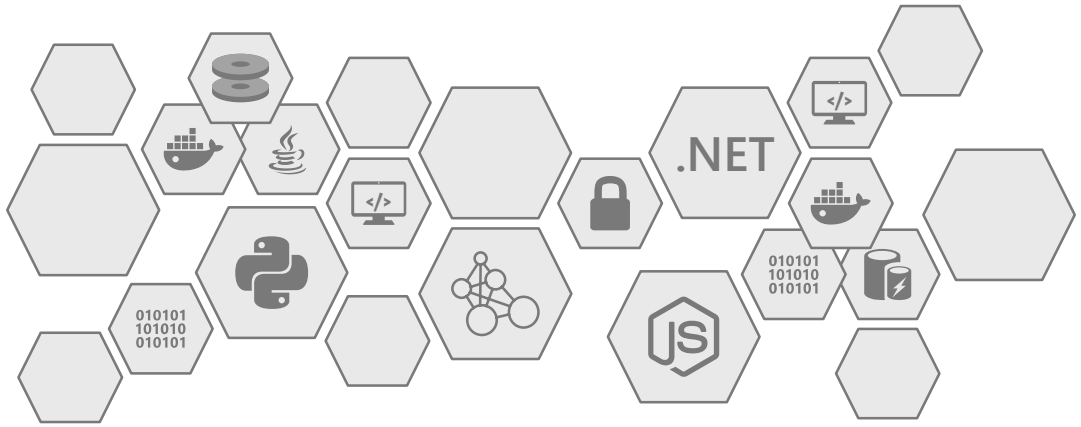
Azure Service Fabric Mesh

A fully-managed microservices platform for business critical applications

Microservices development

Serverless infrastructure

Scale on demand



Lifecycle management

Pay only for what you use

Intelligent traffic routing

Microservice and container orchestration

Security & compliance

Health & monitoring

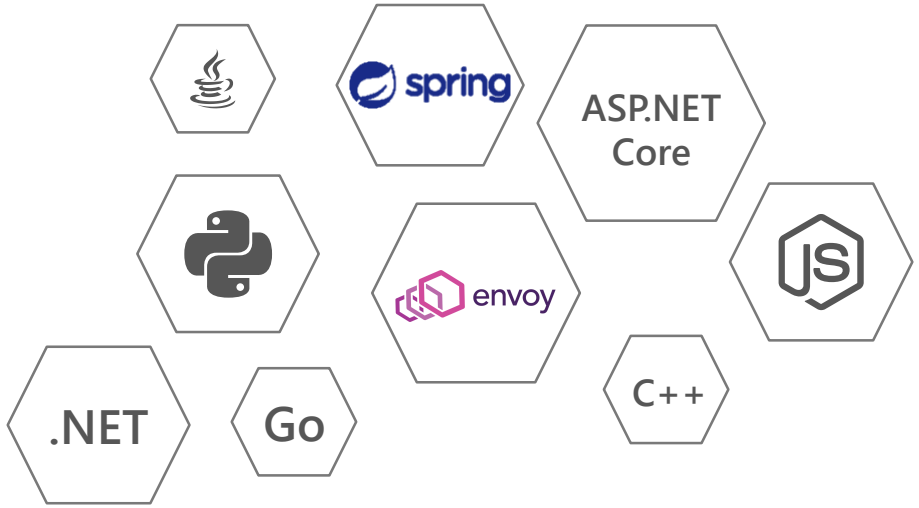
Service Fabric Mesh - Modernize

- Deploy anything and everything in a container
- Bring your own network to connect to other systems
- No code changes required
- No servers or VMs to manage with Service Fabric Mesh

Service Fabric

Service Fabric Mesh - Cloud-native

- Any language, any framework
- Service Fabric Libraries for multiple languages
- Easy H/A state storage with Reliable Collections
- Intelligent traffic routing and connectivity



Service Fabric

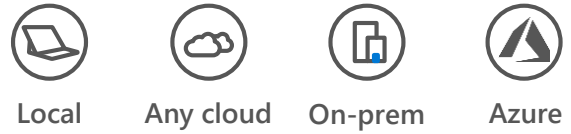
How you write your applications

Simplicity
Portability

Control
Integration

Docker Compose

- Run Docker Compose workloads on Service Fabric
- Limited Service Fabric integration
- Portability for existing applications described by Docker Compose
- Everything runs in containers



Service Fabric Resources

- Loosely coupled individually deployable resources
- Decoupled from runtime lifecycle
- Universal model for any language, framework, or arbitrary application
- Everything runs in containers

Application and Service Manifest

- Low-level control of Service Fabric platform primitives
- .NET and Java frameworks tightly integrated with runtime lifecycle
- Several different programming models
- Allows processes or containers

Service Fabric Mesh - Resources

A deployment model composed of individual resources.

Anything that can be deployed to Service Fabric is a resource.

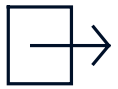

Applications and Services


Networks


Secrets


Volumes


Routing rules

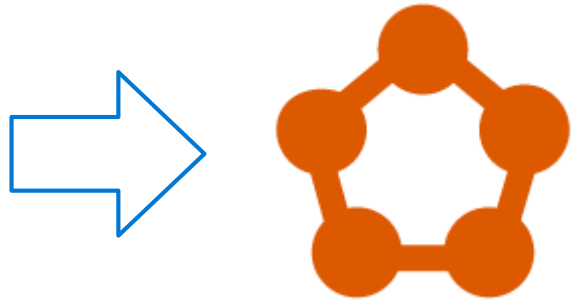

Auto-scale rules

Service Fabric Mesh Resources

Resources are described declaratively in YAML or JSON files.

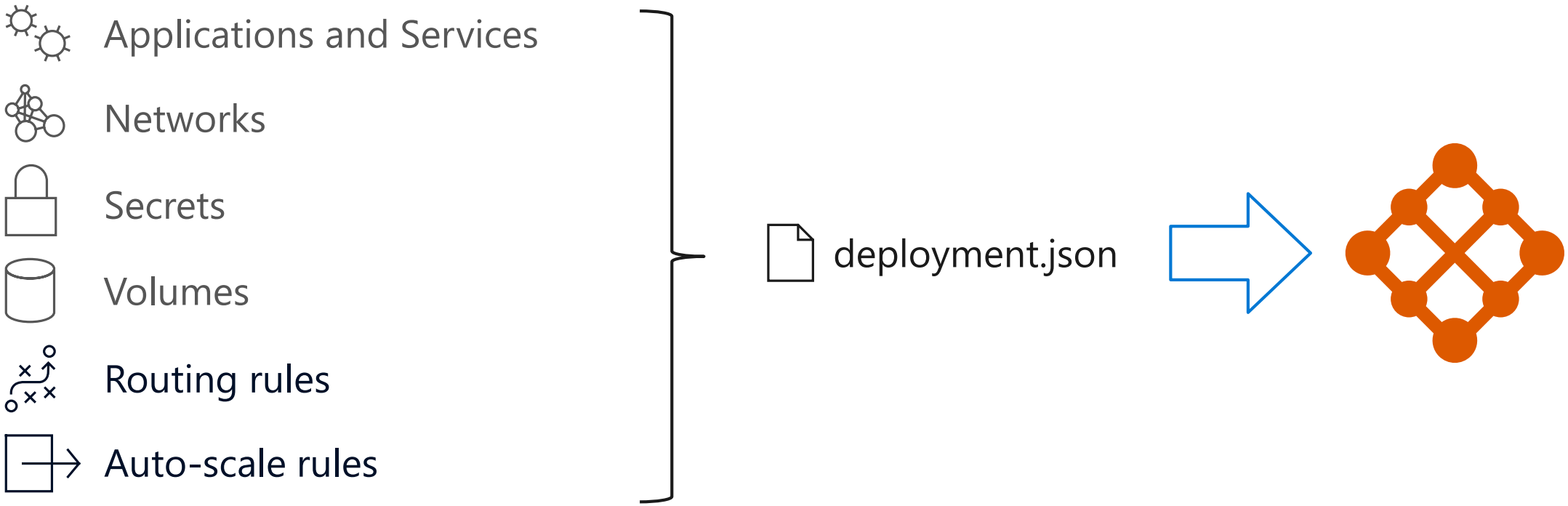
Resources can be deployed anywhere Service Fabric runs.

Applications and Services	my-app.yaml
Networks	my-app-network.yaml
Secrets	my-secrets.yaml
Volumes	my-volume.yaml
Routing rules	my-routes.yaml
Auto-scale rules	my-autoscale-rules.yaml



Service Fabric Mesh Resources

Resources can be grouped together in an Azure Resource Manager template when deployed to Azure Service Fabric Mesh.



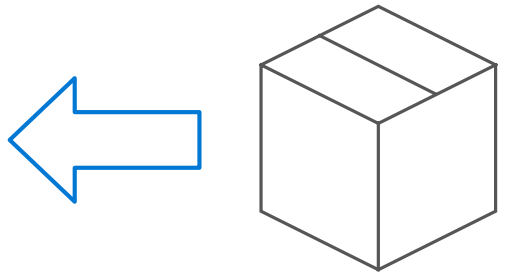
Service Fabric Mesh Resources

Any language, any framework

- No Service Fabric frameworks or base classes. Just containers.
- Service Fabric libraries provide integration with the Service Fabric environment.

Container

- .NET Core
- Java
- Node.js
- Python
- Ruby
- Go
- C++
- Tomcat/IIS/NGINX
- PHP



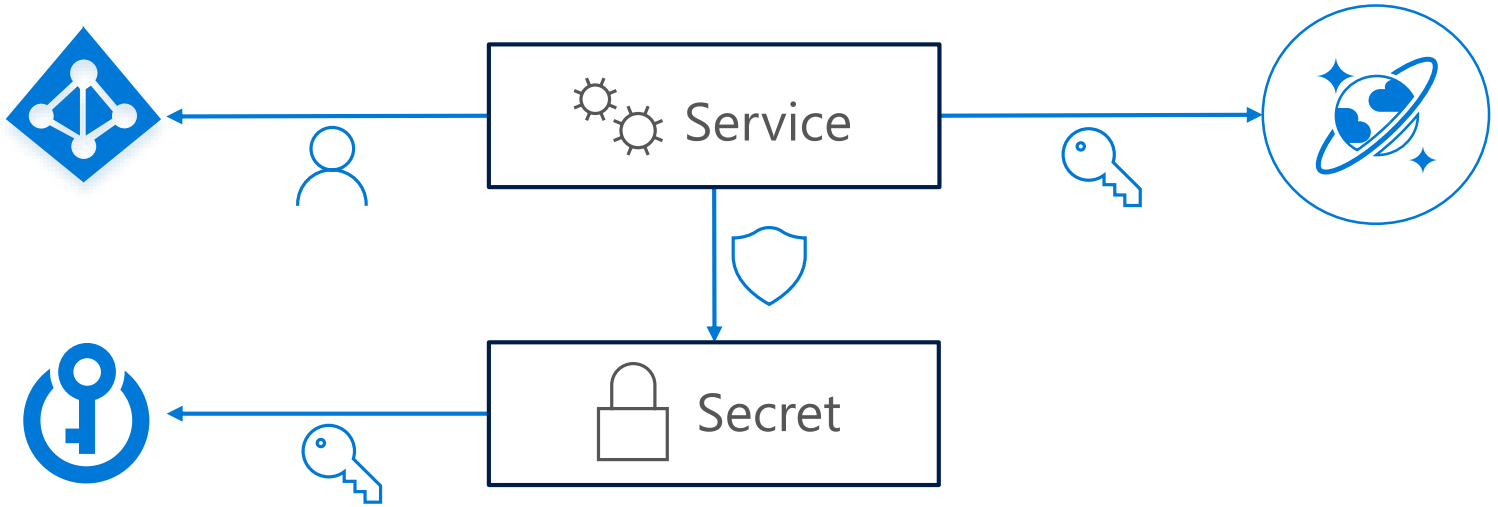
Service Fabric Libraries

- Reliable Collections
- Local H/A storage for ASP.NET Core
- SF API Client

Service Fabric Mesh - Secrets Resource

Inline or stored in Azure Key Vault

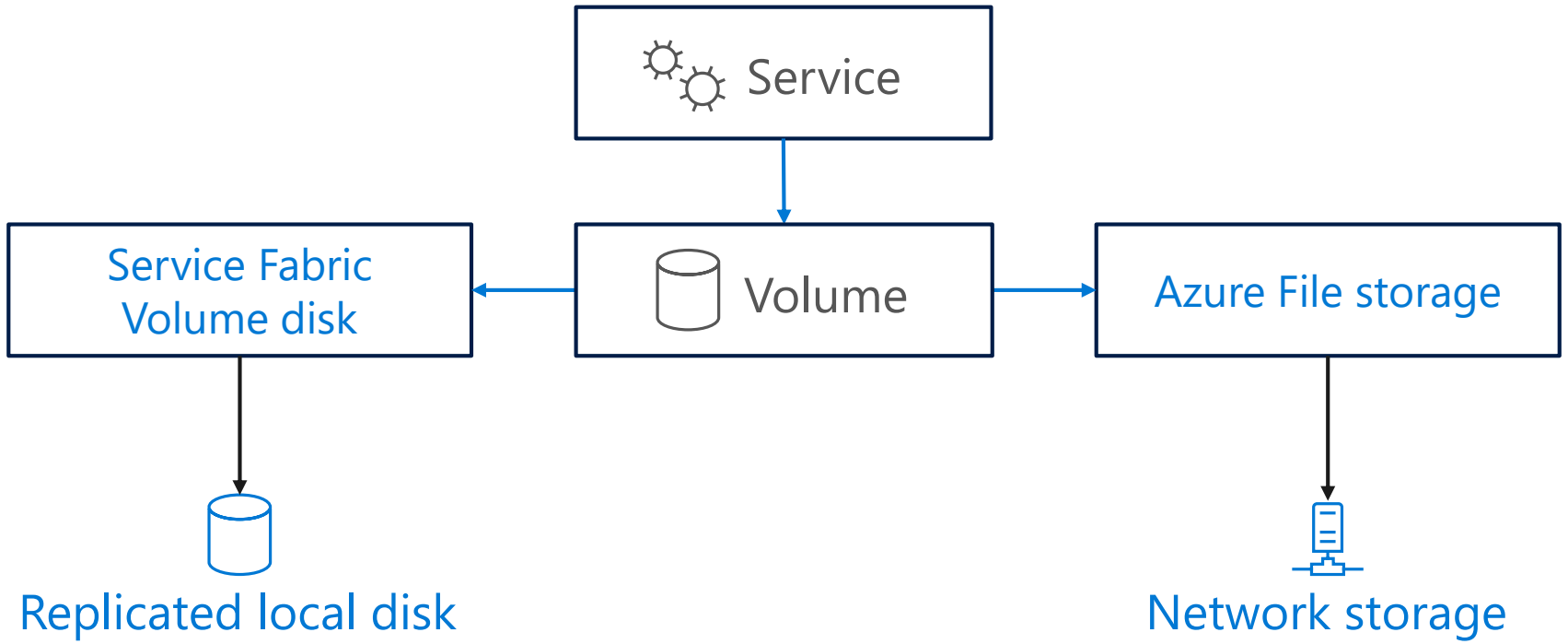
- Applications and service resources have Managed Service Identity (MSI) with AAD to be able to access secrets in Azure Key Vault
- Secrets and certificates can be auto-rolled over with policies



Service Fabric Mesh - Volume Resource

General-purpose file storage

- Read and write files using normal disk I/O file APIs.
- Backed by Azure File storage or Service Fabric Volume disk

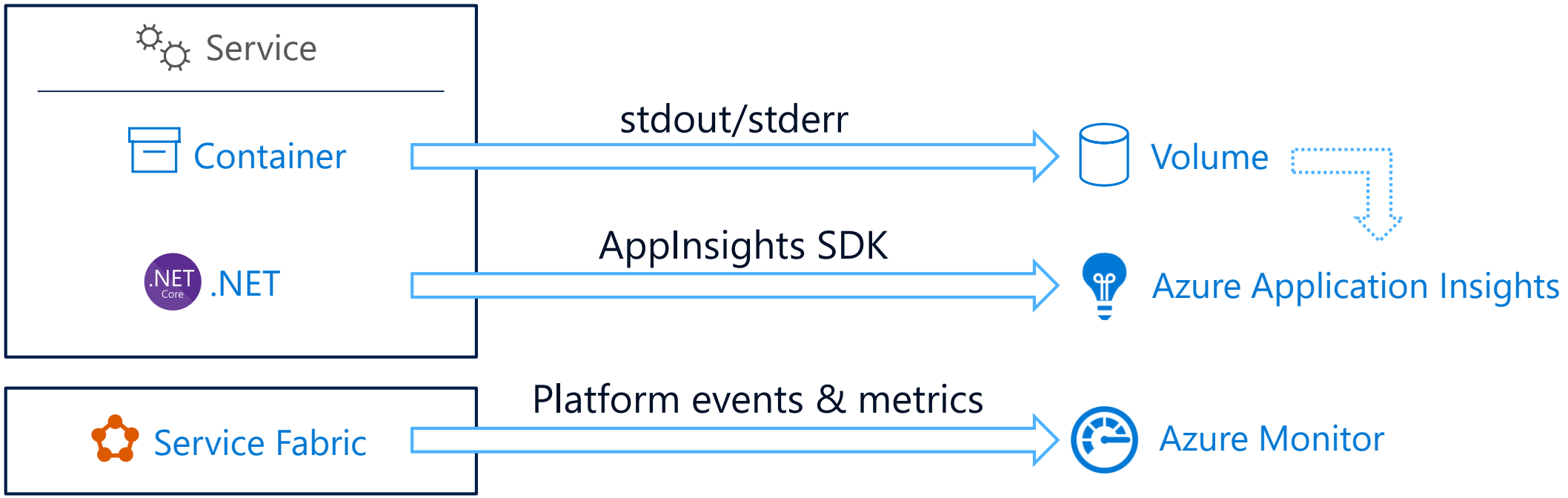


Service Fabric - Diagnostics and Monitoring

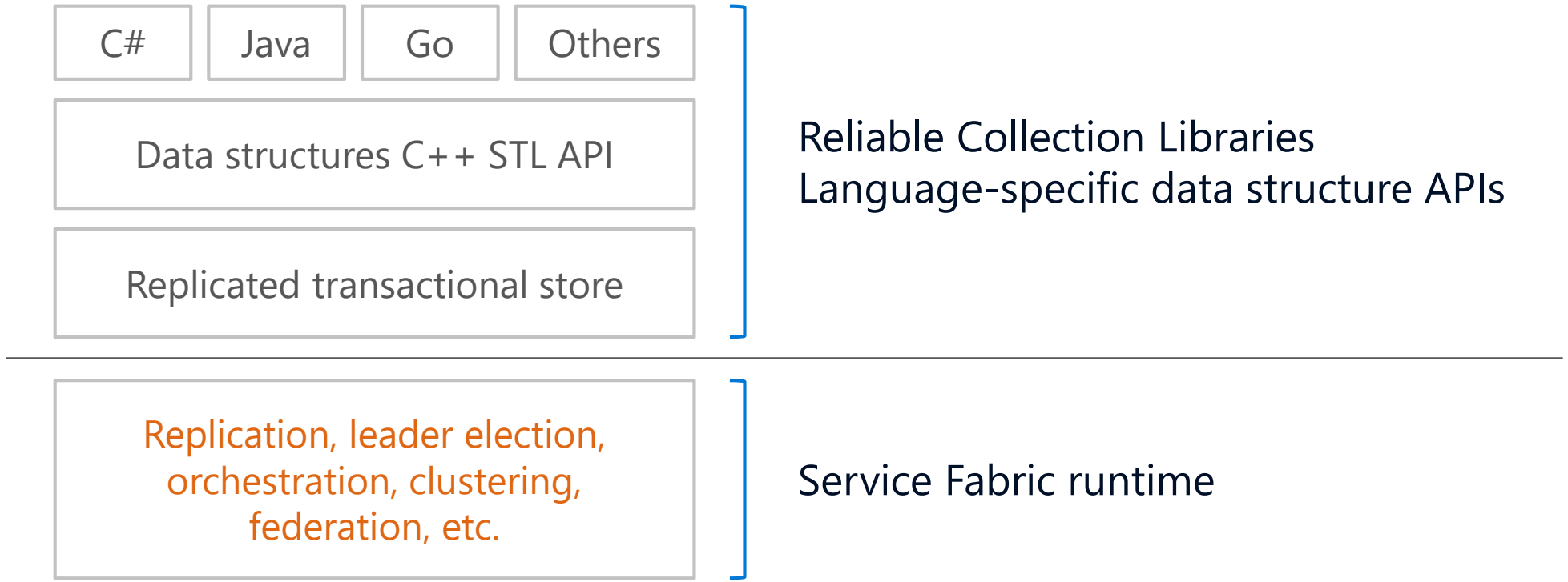
Containers write stdout/stderr logs to volumes

Azure Application Insights integration for .NET

Azure Monitor for platform events and container metrics



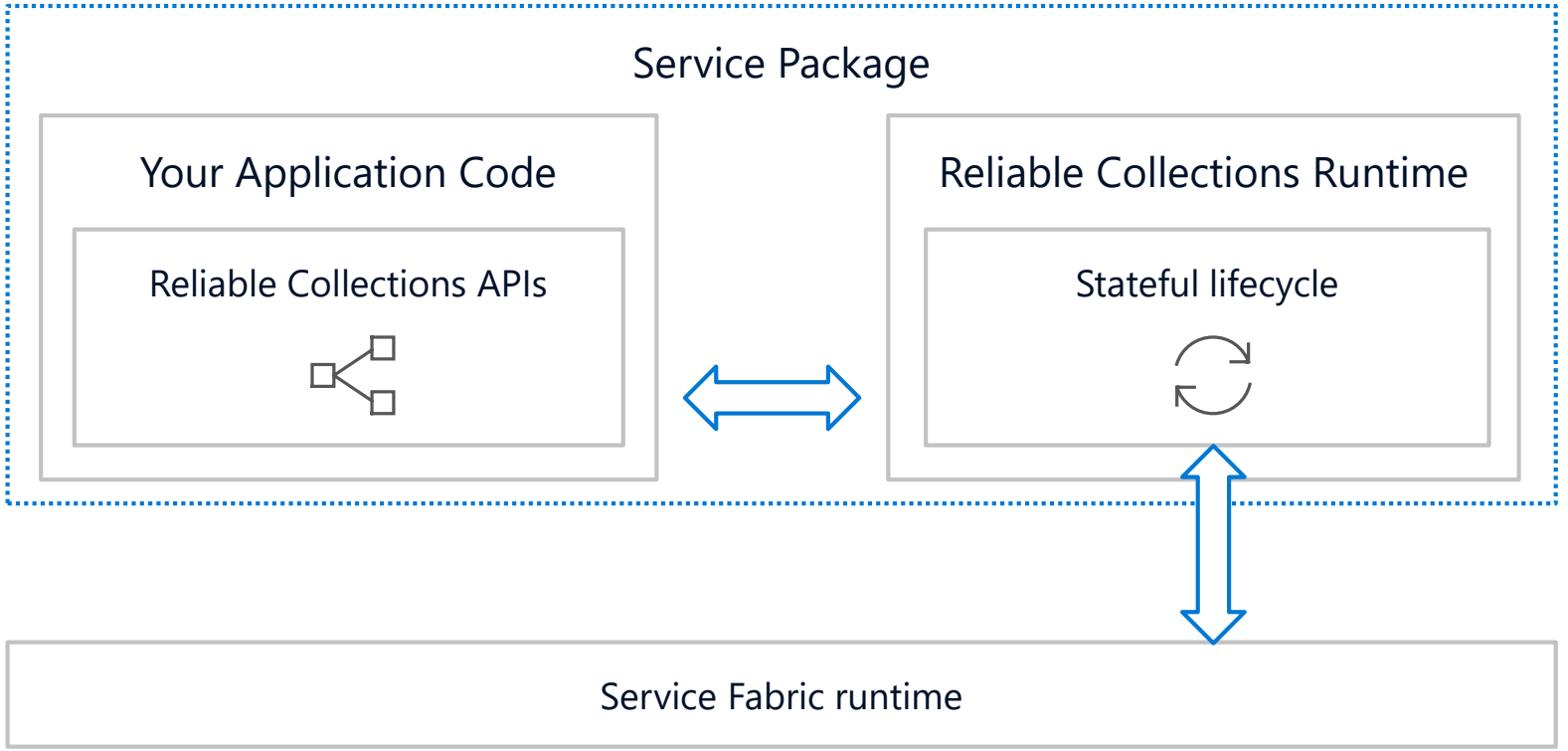
Service Fabric - Reliable Collections



Service Fabric Reliable Collections

Separate lifecycle

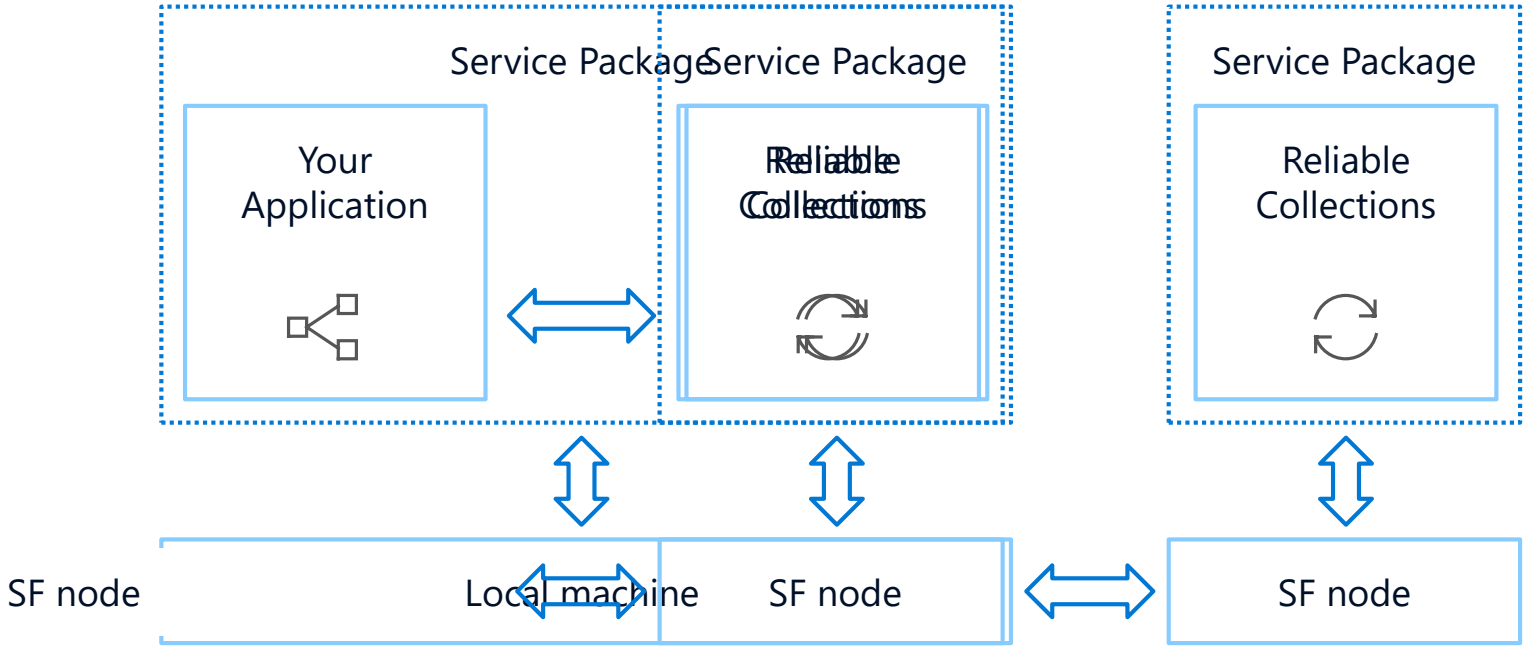
- Reliable Collections handle Service Fabric state machine events
- Your service is stateful, but your code looks stateless



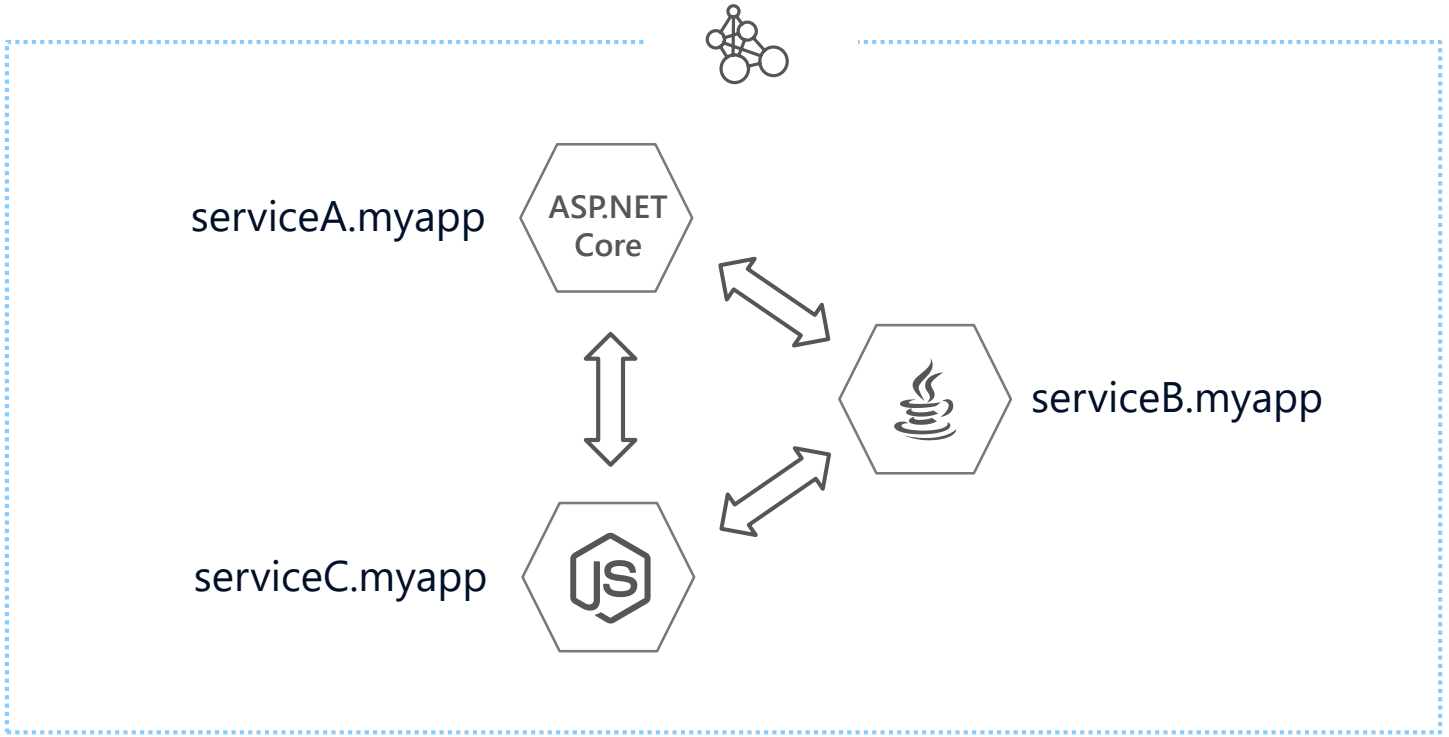
Service Fabric Reliable Collections

Transactional storage anywhere

- Local disk storage when running outside of Service Fabric
- Replicated and partitioned when running on Service Fabric



Service Fabric Intelligent traffic routing

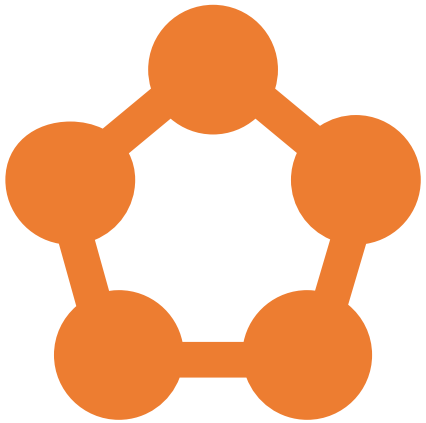



```
await httpClient.GetAsync(
    "http://serviceB.myapp/users/12345");
```

```
http.get('http://serviceB.myapp/users/12345',
    (response) => { … });
```

Service Fabric Intelligent traffic routing

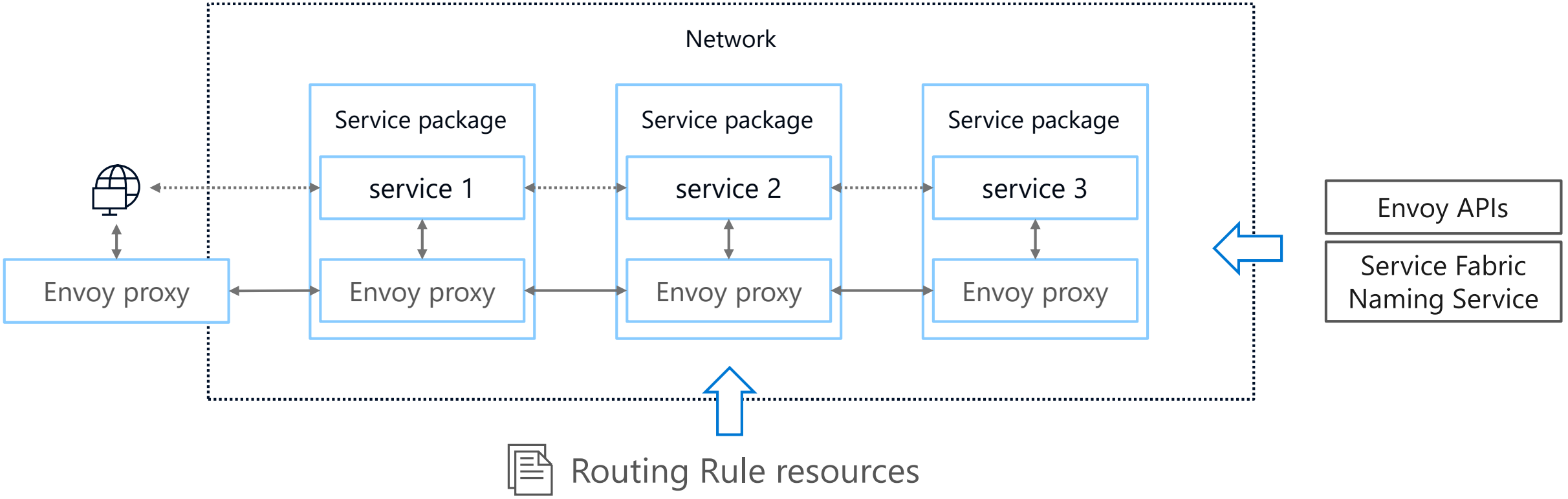
- Services address each other by hostnames
- Services do not implement platform-specific discovery APIs
- Services do not deal with network-level details
  - Retries, circuit breakers, throttling
- Services are unaware of the implementation details of other services
  - Stateless, stateful, partitioning, versions



Service Fabric Intelligent traffic routing

Advanced HTTP/HTTPS traffic routing with load balancing

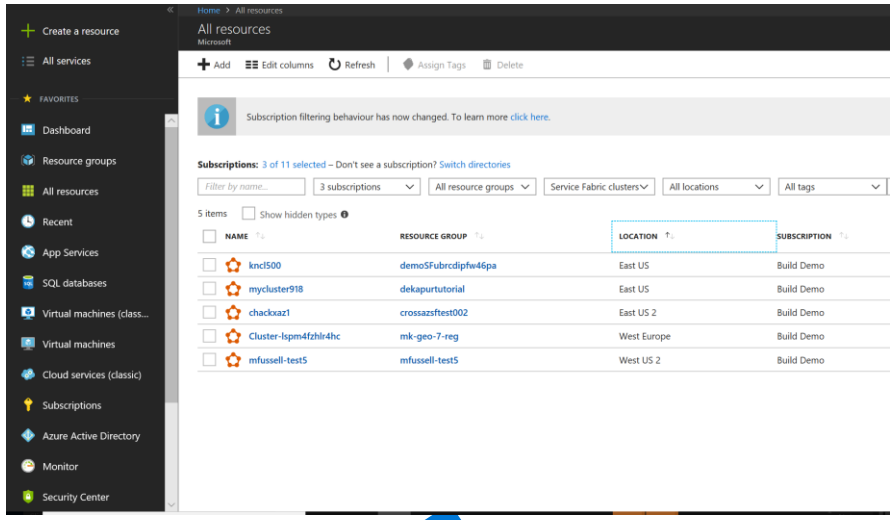
Proxy handles partition resolution and key hashing



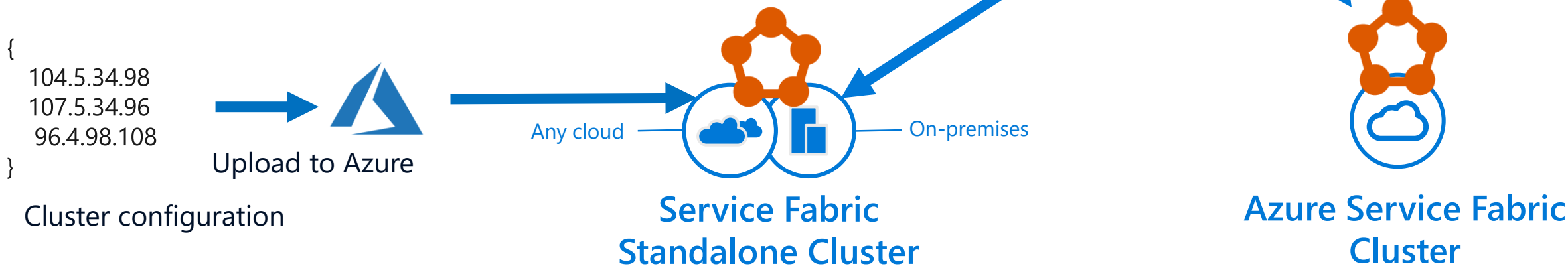
Additional Roadmap Items

- Managing both stand alone and Azure SF clusters in the portal
- Service Fabric Secret Store Service – allows agnostic secret storage with rich API support. Has strong integration with Azure Key Vault

Manage both on-premises and Azure clusters in the portal

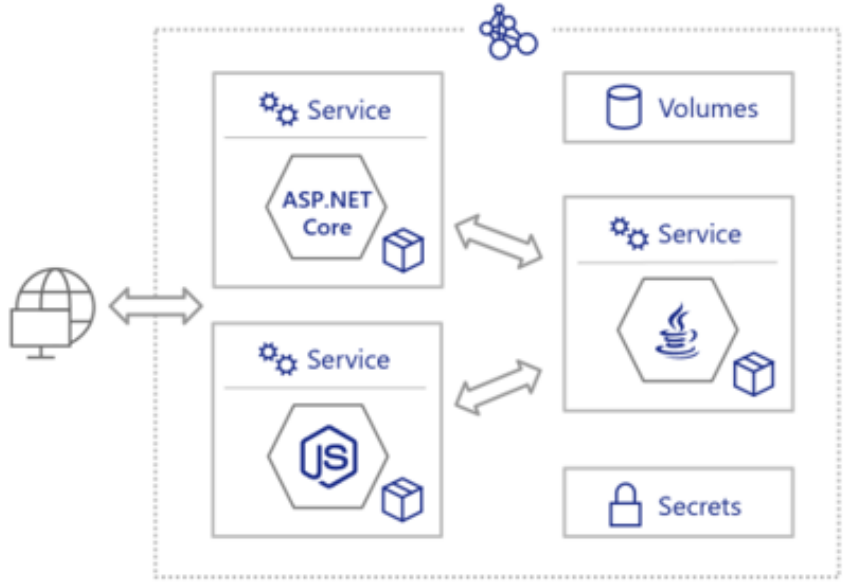


Standalone cluster package generated for Windows or Linux and deployed on-premises



The Future of Service Fabric

- Polyglot Services
- Described by Service Fabric Resources
- Enhanced through Reliable Collections
- Connected Through intelligent routing



Service Fabric

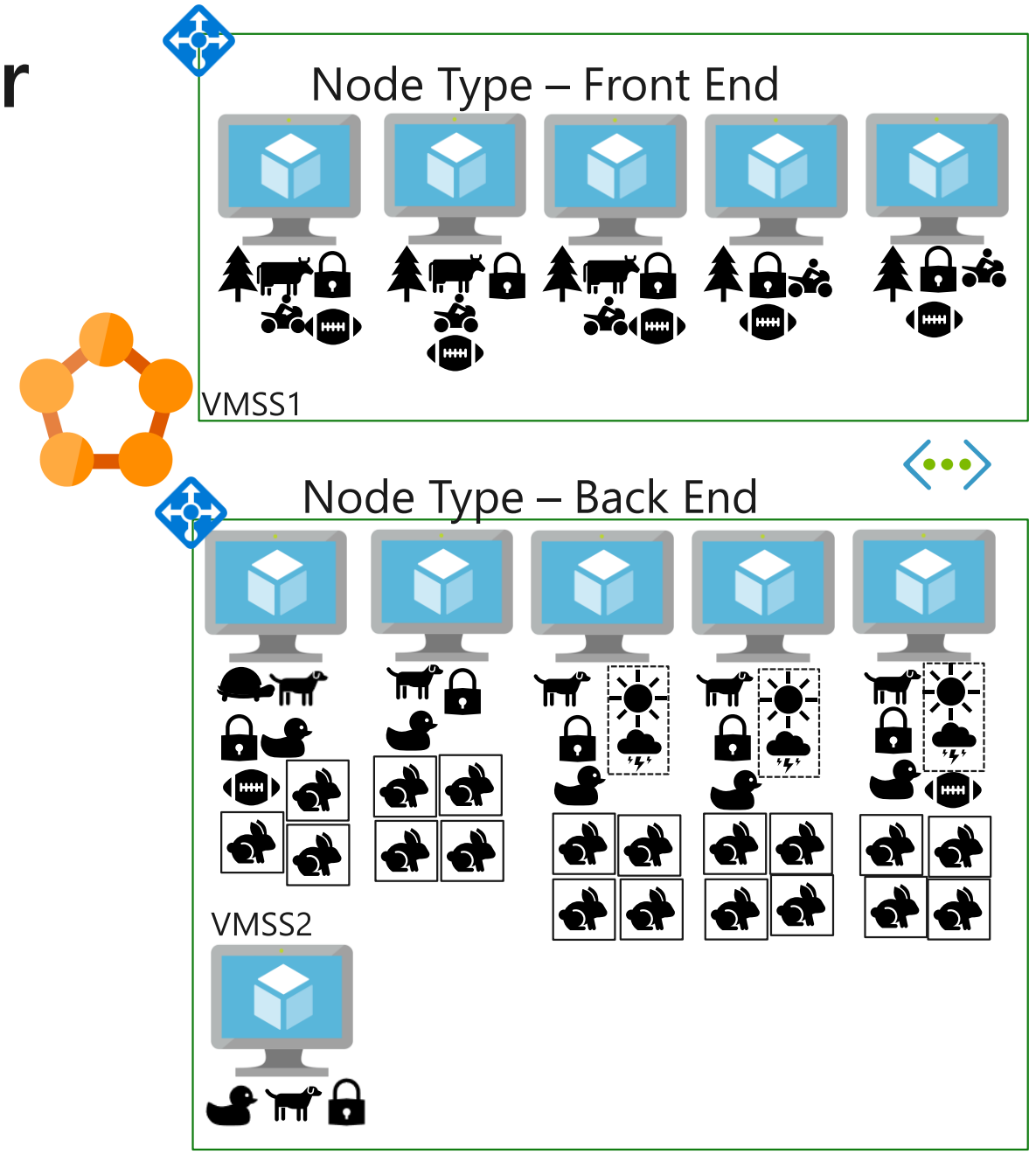
Dev Machine | Any Cloud | On-Prem | Azure Clusters | Mesh

Questions?

Service Fabric Hackathon

Microservices and Container Scenario

- Build the scenario out on the right
- Challenge:
  - Using Visual Studio Construct an application that defines all the stateless and container services in the illustration
- Challenge:
  - Set up the Cluster to resemble what is in the picture
  - Deploy the application into the cluster
- Core concepts:
  - Node Types
  - Placement constraints
  - Autoscaling
  - Service Affinity



DevOps in the Cloud

Prerequisites

- Azure CLI
- To create your service fabric cluster in Azure

VSTS Release and Build

- Create a new VSTS project to manage build and release pipelines

Challenge - Deploy to Azure

Description

- Provision a Service Fabric cluster in Azure. Pick an operating system that matches your development environment
- Deploy the application running 3 instances of the xtoph/nginx:v2 container using the sftctl CLI

Success Criteria

- Service Fabric explorer in the browser from an Azure URL

Challenge - Deploy using a VSTS Release

- Production applications rarely deployed from the CLI. Production uses CI/CD pipelines. Let's build a VSTS build/release pipeline to deploy the application into the cluster. In a real production environment, we'd build the container images, but for this challenge, we'll keep things simple and only deploy existing images.

Success Criteria

- Service Fabric Explorer shows 2 deployed applications, with 3 replicas each
- CI/CD Application Name includes the VSTS build number
- Application is accessible via Azure URL

Setting up Local Development

One of the benefits is developing locally on your machine. On Windows, that allows for a great debugging experience allowing F5 debugging wiht Visual Studio and Eclipse for native applications.

In these challenges you will:

- set up a development environment on your local machine
- deploy a containerized application into your cluster
- experience the resilience of service fabric applications

Challenge 1

- Setup your local dev box, either on Windows, Mac or Linux.

Success Criteria

- sfctl is connected to the clsuter
- Service Fabric explorer displays in the browser from http://localhost:19080/Explorer.

Challenge 2 - Deploying an Application

- Windows Deploy a containerized application running a single instance of the xtoph/webapp:1.0.0 container into your local environment. The webapp listens on port 5000. Make sure you configure Service Fabric to map the port.
- Linux Deploy a containerized application running a single instance of the xtoph/nginx:v1 container into your local environment. Nginx listens on port 80. Make sure you configure Service Fabric to map the port.

Success Criteria

- The application shows deployed in Service Fabric explorer
- Windows The MVC home page shows in your browser when accessed via the cluster DNS name
- Linux The NGINX index page shows in your browser when accessed
- Verify that it survives and "application crash" and update the application to the new version.

Challenge 3 - No Downtime Changes

- Using the sfctl CLI increase the number of instances of the xtoph/nginx:v1 container from 1 to 3. Then update the version from windows xtoph/webapp:1.0.0 to xtoph/webapps:2.0.0 Linux xtoph/nginx:v1 to xtoph/nginx:v2.
- Watch the Service Fabric explorer during update and scale operations

Success criteria

- 3 running instances of the Service displayed in Service Fabric explorer
- Browser shows windows MVC Version 2 page or Linux xtoph/nginx:v2 welcome page