Microsoft

Introduction to Service Fabric

Microsoft Services

Conditions and Terms of Use

Microsoft Confidential

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet website references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Copyright and Trademarks

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see **Use of Microsoft Copyrighted Content** at
https://www.microsoft.com/en-us/legal/intellectualproperty/permissions/default.aspx

Microsoft®, Internet Explorer®, Outlook®, SkyDrive®, Windows Vista®, Zune®, Xbox 360®, DirectX®, Windows Server® and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Agenda

- Key concepts of Azure Service Fabric
- Understanding the architecture of Azure Service Fabric
- Reliable Services
- Reliable Actors
- Service Upgrades
- Capacity Planning

Microsoft

Introduction to Service Fabric

Service Fabric Overview

Microsoft Services

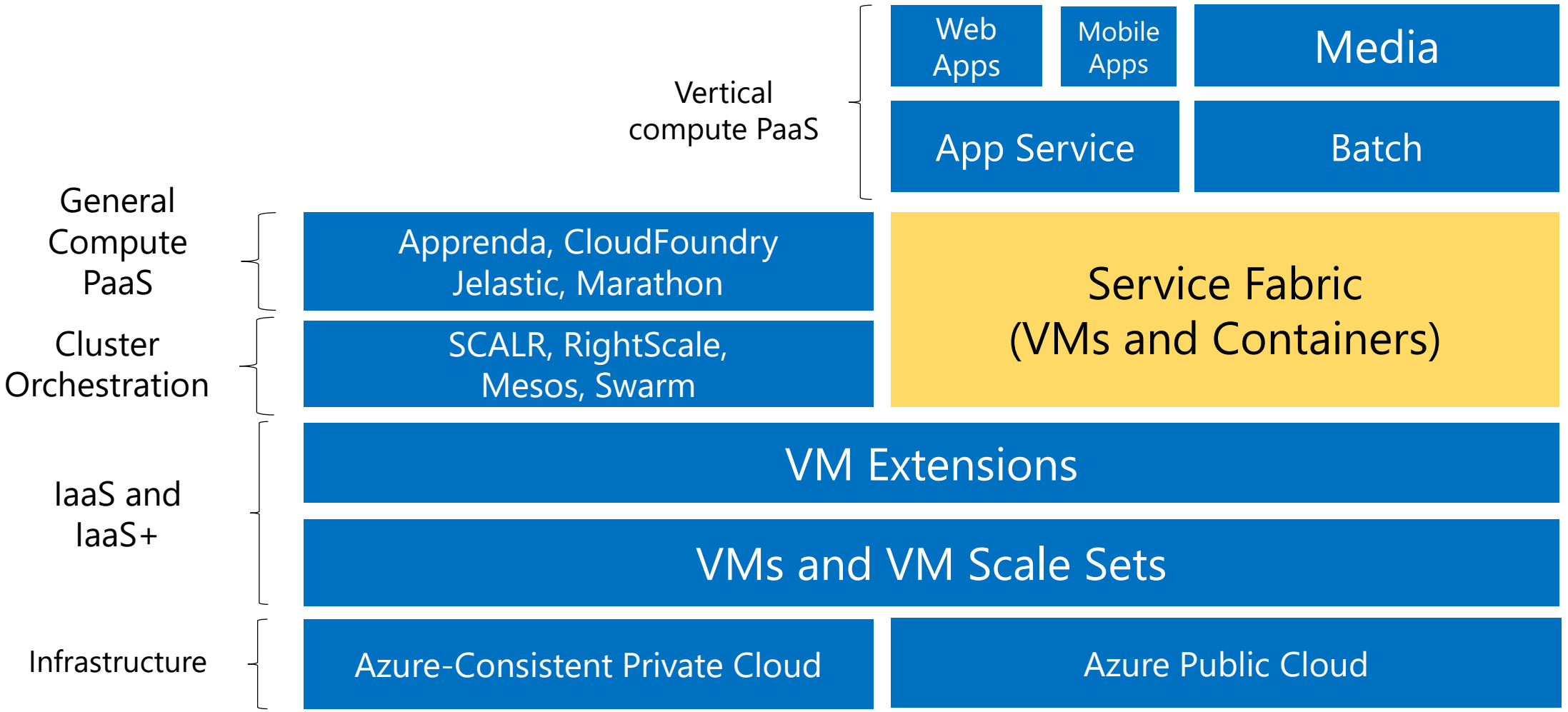
Current distributed computing challenges

- Developing massively scalable applications that are self-healing
- Dynamically scaling applications
- Handling infrastructure failures
- Managing the lifecycle of the application without downtime, ie, upgrades

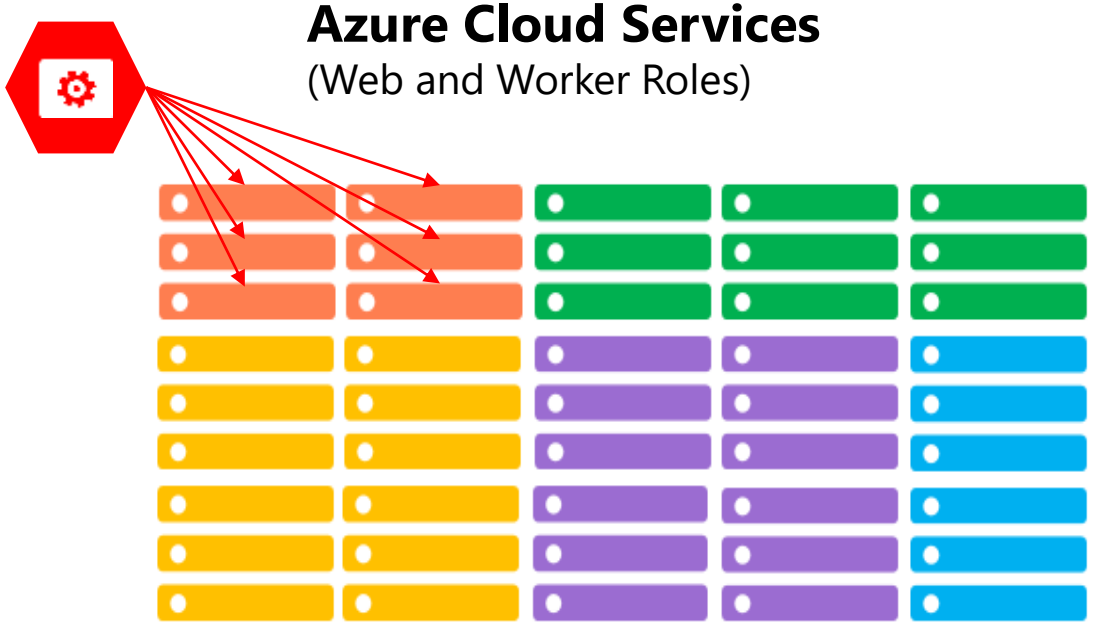
What is Azure Service Fabric?

"A distributed systems platform that makes it possible to build scalable, reliable, low-latency managed services and applications for the cloud."

Open Choices at Every Layer

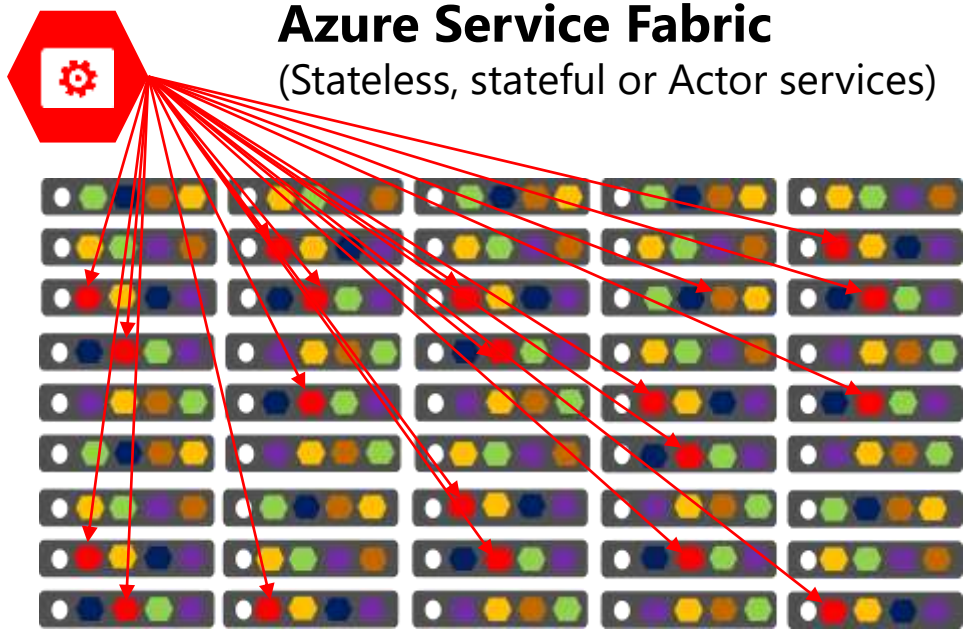


Azure Cloud Services vs. Azure Service Fabric



Azure Cloud Services
(Web and Worker Roles)

- 1 service instance per VM with uneven workloads
- Lower compute density
- Slow in deployment and upgrades
- Slower in scaling and disaster recovery

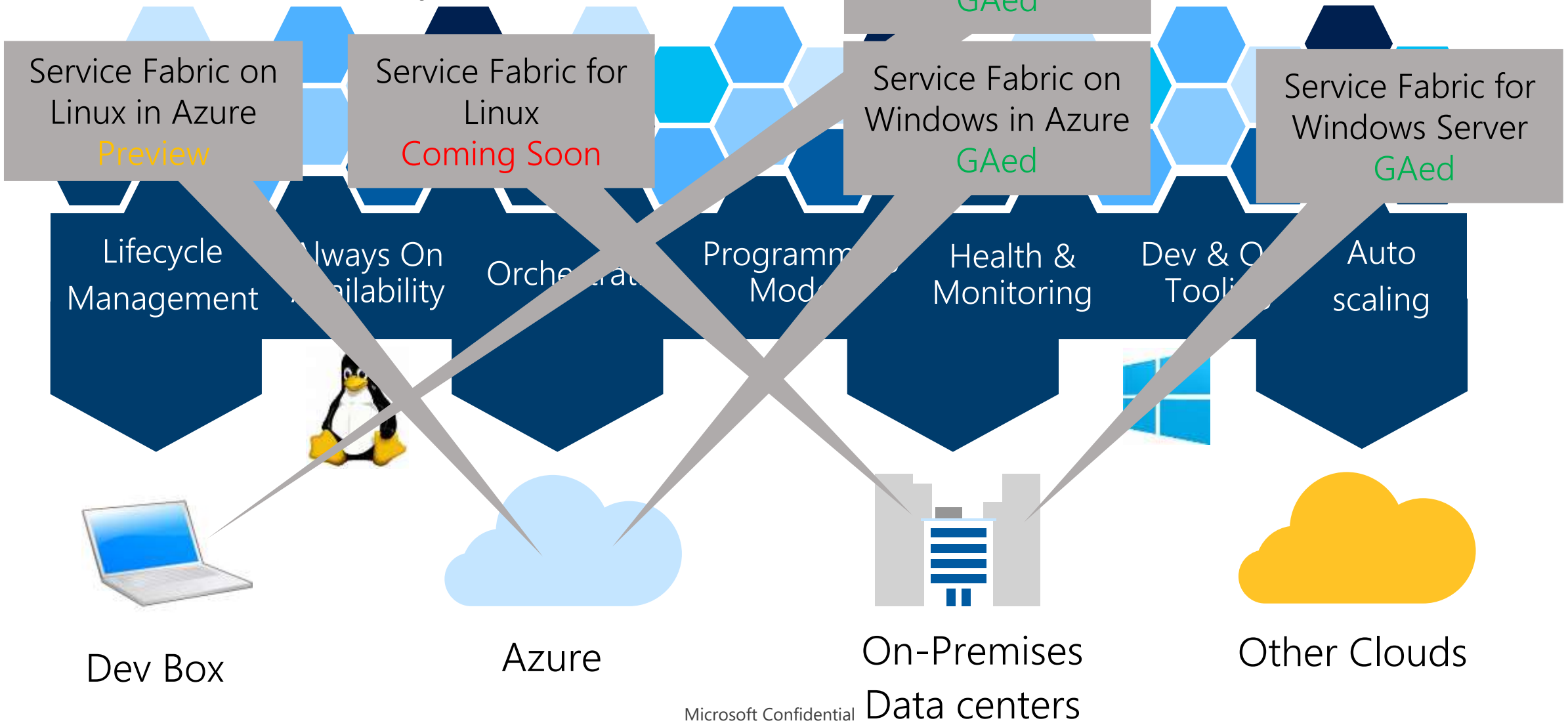


Azure Service Fabric
(Stateless, stateful or Actor services)

- Many microservices per VM
- High microservices density
- Fast deployment and upgrades
- Fast scaling microservices

Azure Service Fabric

Windows/Linux OS, Any Cloud



Service Fabric Capabilities

- Application deployment services:
  - Rolling update with rollback - Upgrade and patch microservices within applications independently
  - Strong versioning
  - Side-by-side support
- Leadership election
- Naming service for discovery of applications
- Partitioning support
- Azure Load balancing and placement constraints
- Consistent state replication framework
- Ability to scale-up or scale-down your Service Fabric cluster

Example PaaS Services Built On Service Fabric

Service	Description
Azure SQL Database	Scale-out relational database
Azure DocumentDB	NoSQL store for JSON documents Integrated with O365
Azure Power BI	BI Pro Data Analysis Services
Azure Networking	Regional Network Manager for cross cluster/DC VNET
Azure Compute and Networking	Resource Providers for Compute (CRP), Networking (NRP), Storage (SRP)
Event Hubs	Streaming messaging and event processing
Service Bus	Service Bus Resource Provider (SBRP)
Intune	Unified management of PCs and devices on the cloud.
Bing Cortana	Personal assistant

In production for six years
We're releasing the same bits we run

Service Fabric and microservices

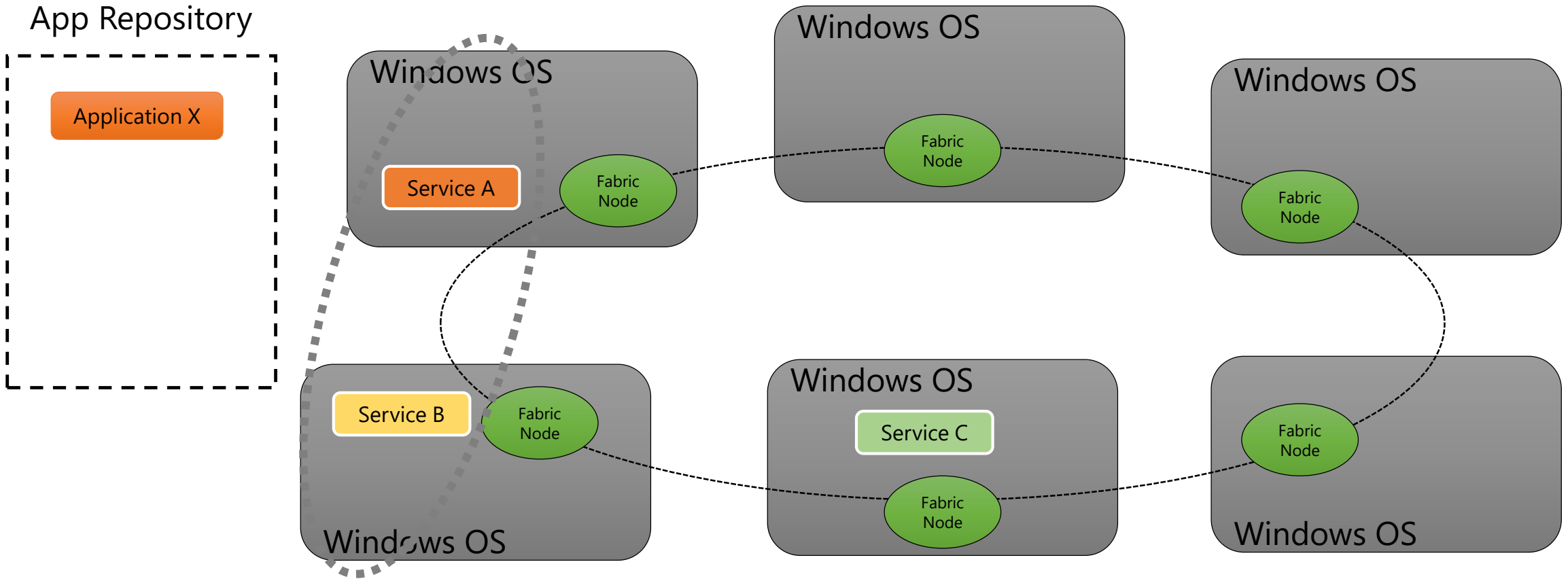
- A microservice is whatever you want it to be:
  - ASP.NET application *(no support for applications that require IIS)*
  - node.js, Java VMs
  - Arbitrary .exe
- Stateless microservices
  - Stateless microservices (e.g. protocol gateways, web proxies, etc.) do not maintain any mutable state outside of any request and its response from the service.
  - Azure Cloud Services worker roles are an example of stateless service
- Stateful microservices
  - Reliability of state through replication and local persistence
  - Reduces the complexity and number of components in traditional three-tier architecture

Service Fabric and microservices (con't)

- A microservice is (*logic* + *state*) that is independently versioned, deployed, and scaled
- Have a unique name that can be resolved
  e.g. fabric:/myapplication/myservice
- Interacts with other microservices over well defined interfaces and protocols like REST
- Remains always logically consistent in the presence of failures
- Are hosted inside a process (code + config)
- Are typically developed by a small engineering team

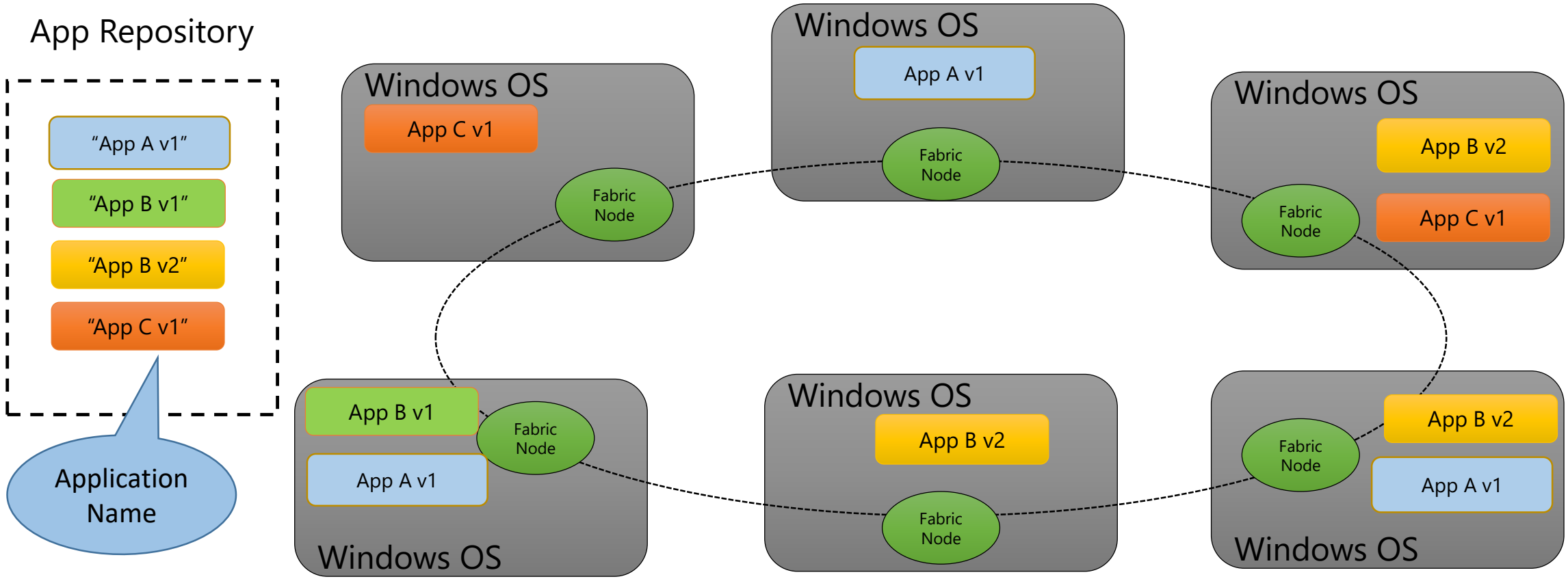
A Service Fabric application is…

- A set of services that make up the application
- Used for packaging, deployment and versioning
- Staged in an app repository; copied and launched on demand



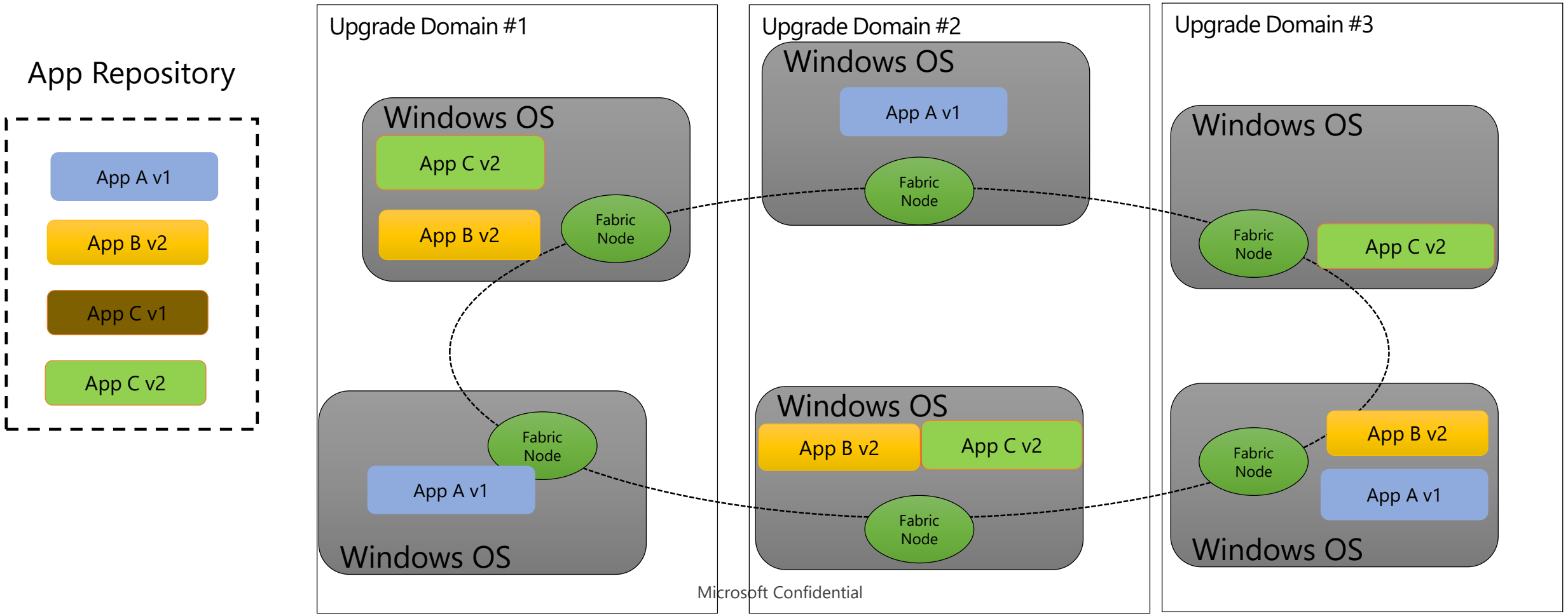
Service Fabric Application "Instances"

- Multiple copies of the same app type and version per cluster – different app names
- Each application instance is managed and versioned independently

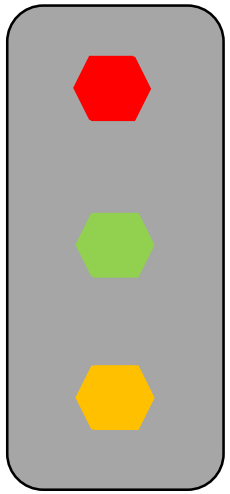


Application Upgrade

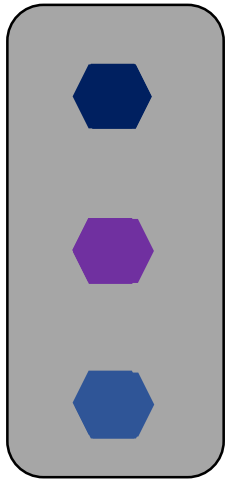
- "Rolling Upgrade" to ensure service availability during upgrade
- Upgrades performed per upgrade domain



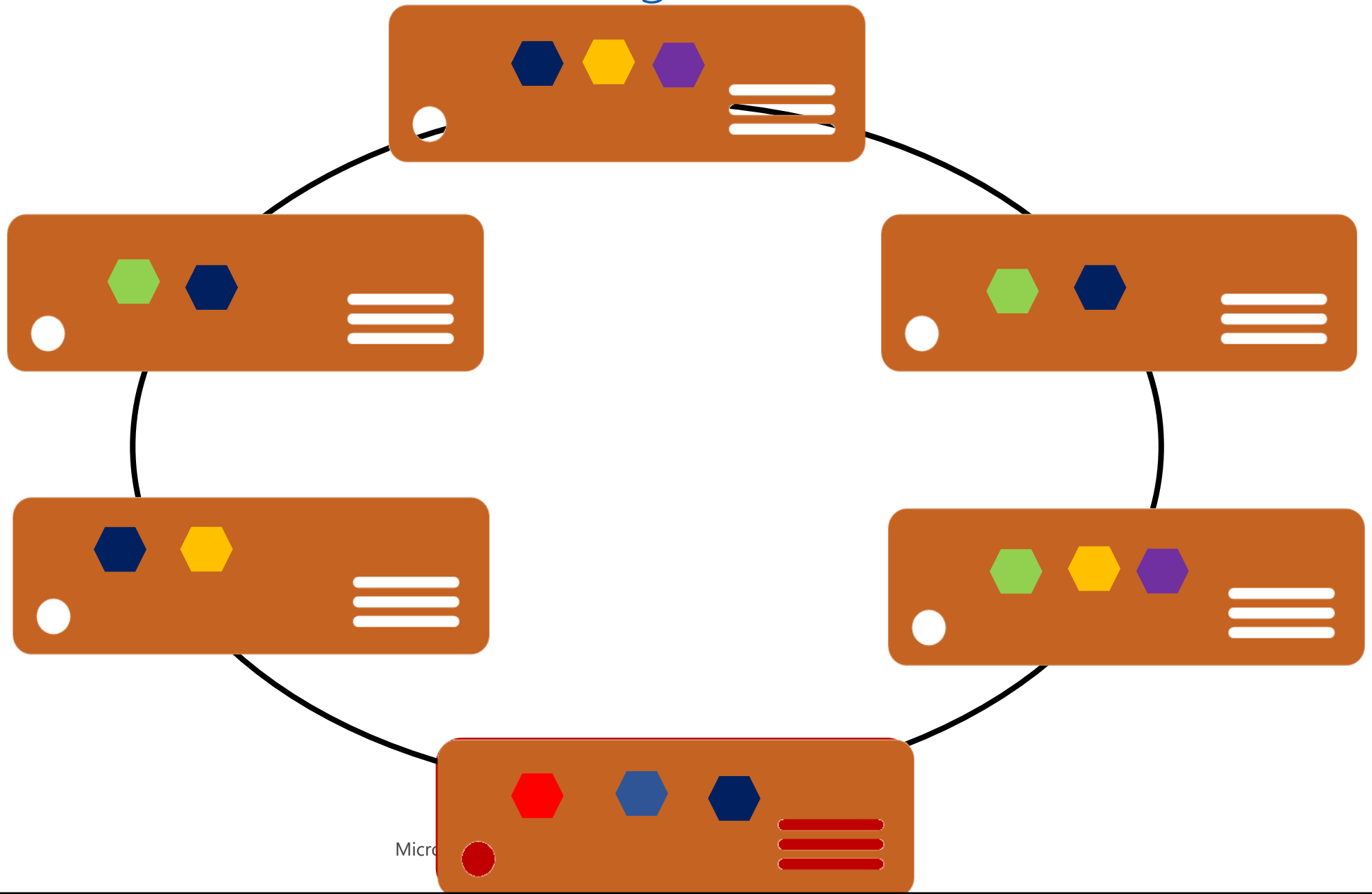
Service Fabric cluster microservice management



App1



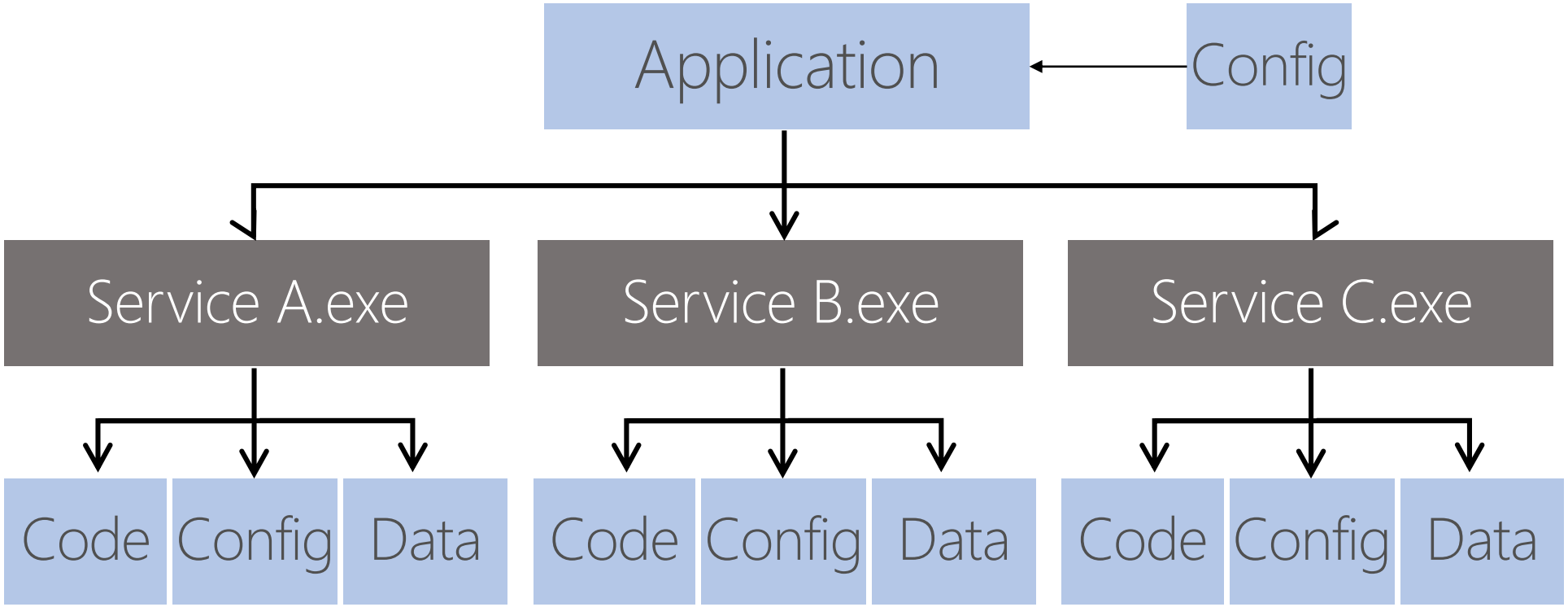
App2



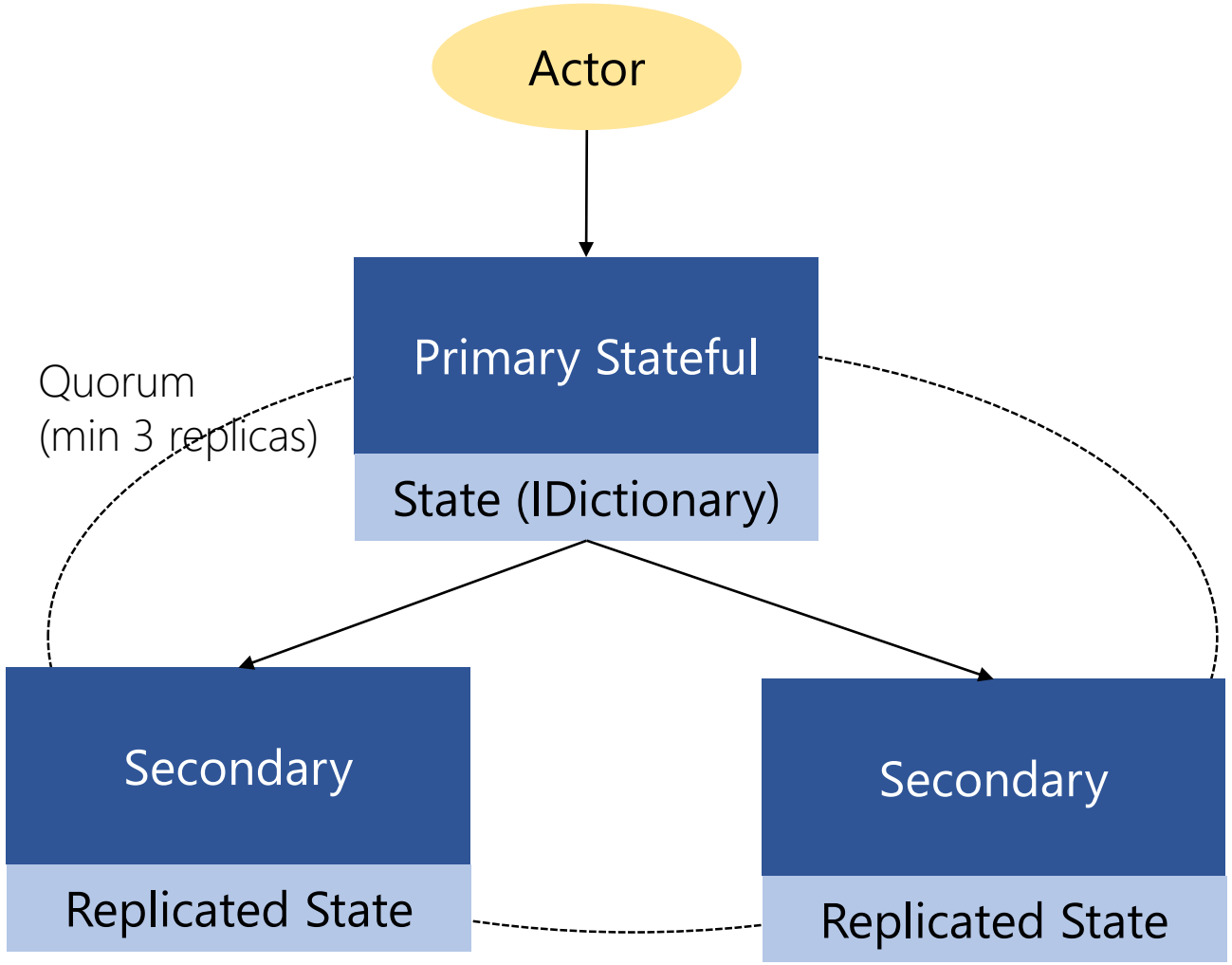
Service Fabric Key Concepts

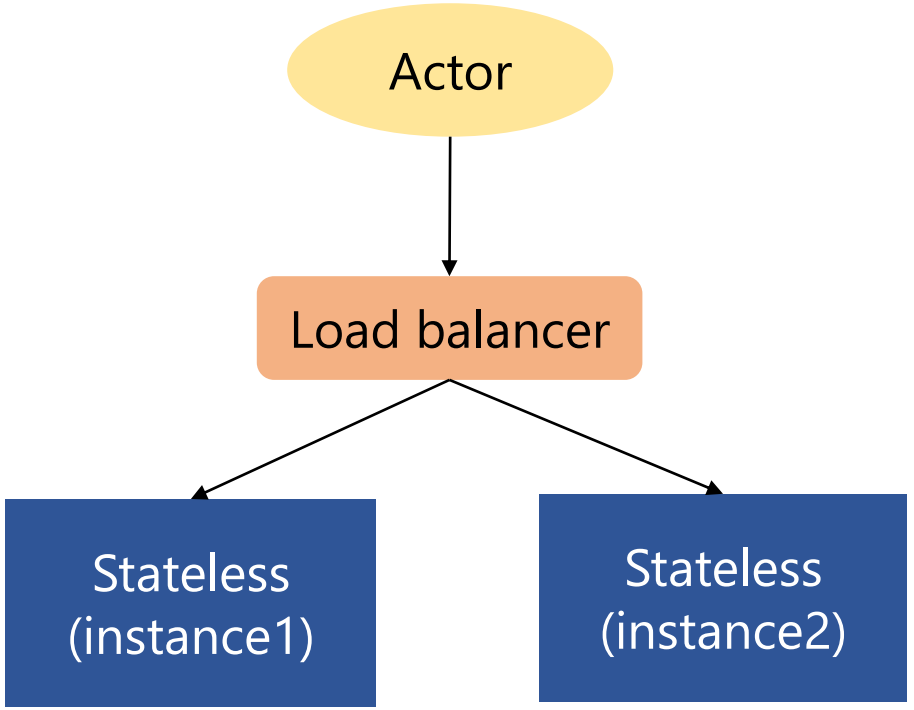
- Cluster
- Node
- Application/Application Type
- Service/Service Type
- Application Package

Service Fabric Application Model



Stateful and Stateless Services





Demonstration

Installing the Service Fabric components on your machine

Demonstration

Creating a Service Fabric Cluster in the Portal

Microsoft

Introduction to Service Fabric

What questions should I ask?

Microsoft Services

Questions for You…

- What are you trying you do independently – without the use of Service Fabric?
- Do you scale all or part of the app?
- How do you version or upgrade the app?
- Do you use different technologies in the app?
- Do you have different teams building the app?
- Have you thought about how you scale?
- Does your app need to scale or is there a fixed scale limit?
- Do you know how the different parts of the app are required to scale?
- How do you partition your data?

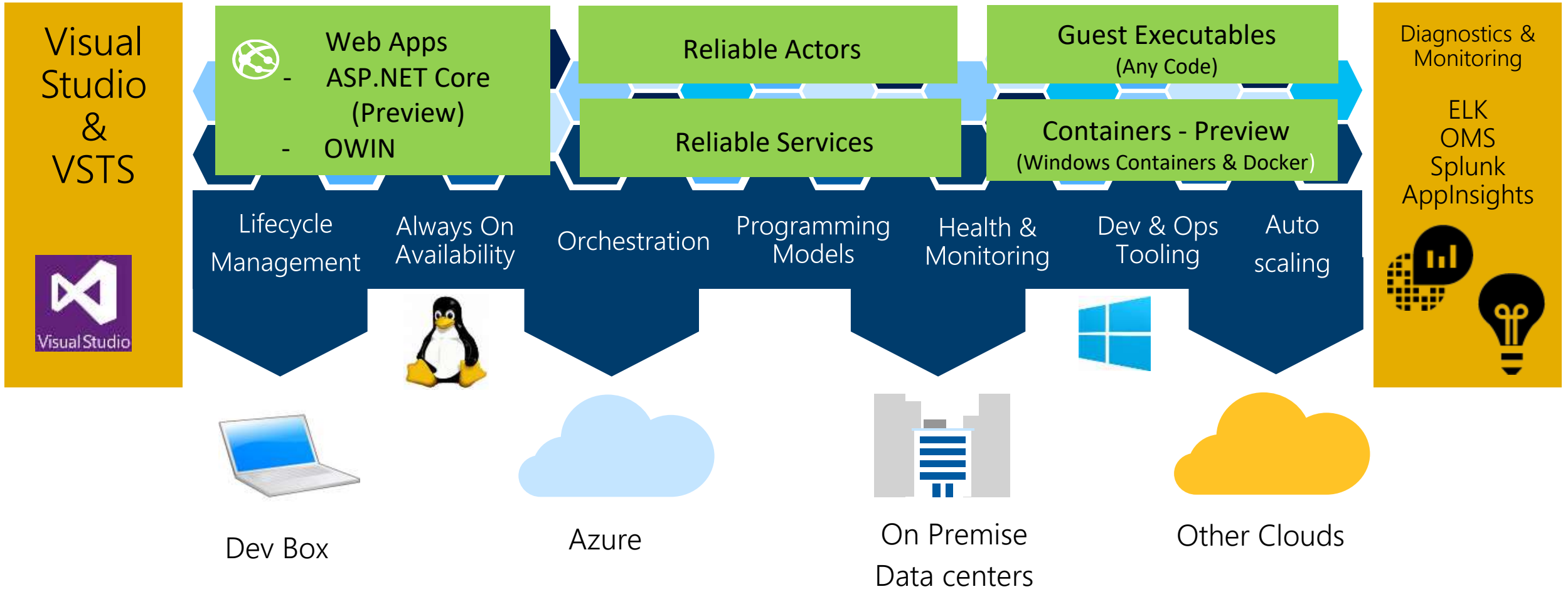
Microsoft

Introduction to Service Fabric

Programming Models

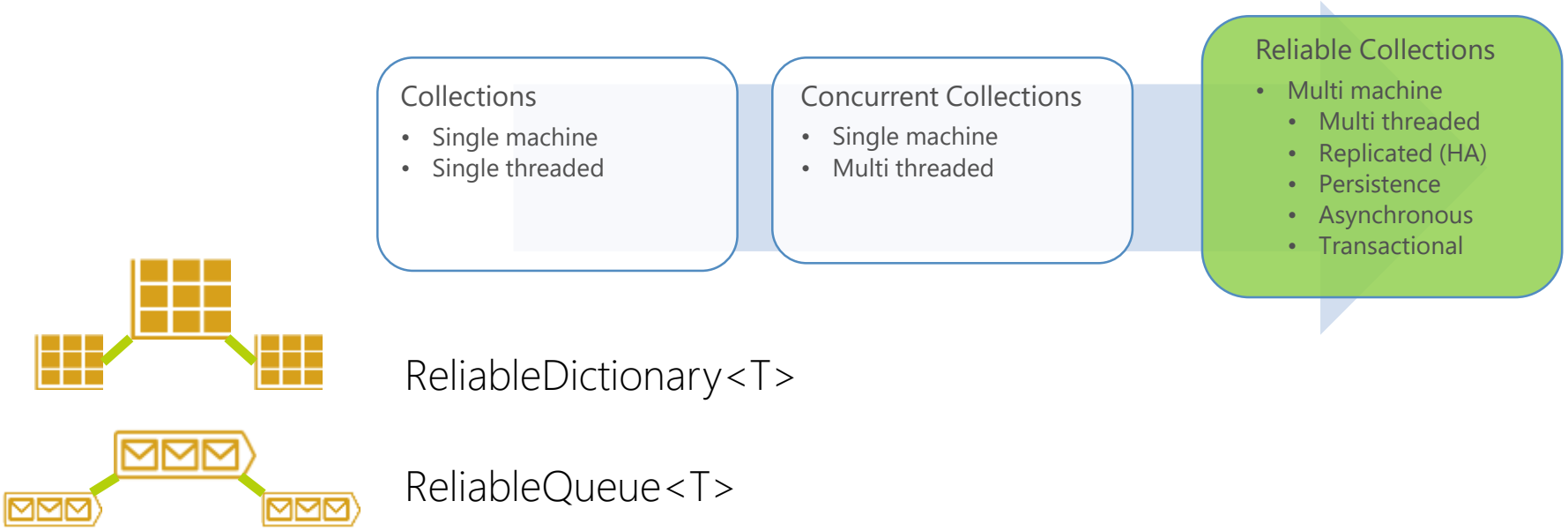
Microsoft Services

Service Fabric Programming Models & CI/CD



Reliable Services API

- Build stateless services using existing technologies such as ASP.NET
- Build stateful services using reliable collections
- Manage the concurrency and granularity of state changes using transactions
- Communicate with services using the technology of your choice
  - e.g. WebAPI and WCF



Reliable Actor API

- Build reliable stateless and stateful objects with a virtual Actor programming model
- Suitable for applications with multiple independent units of state and compute
- Automatic state management and turn based concurrency (single threaded execution)

Microsoft

Introduction to Service Fabric

Reliable Services

Microsoft Services

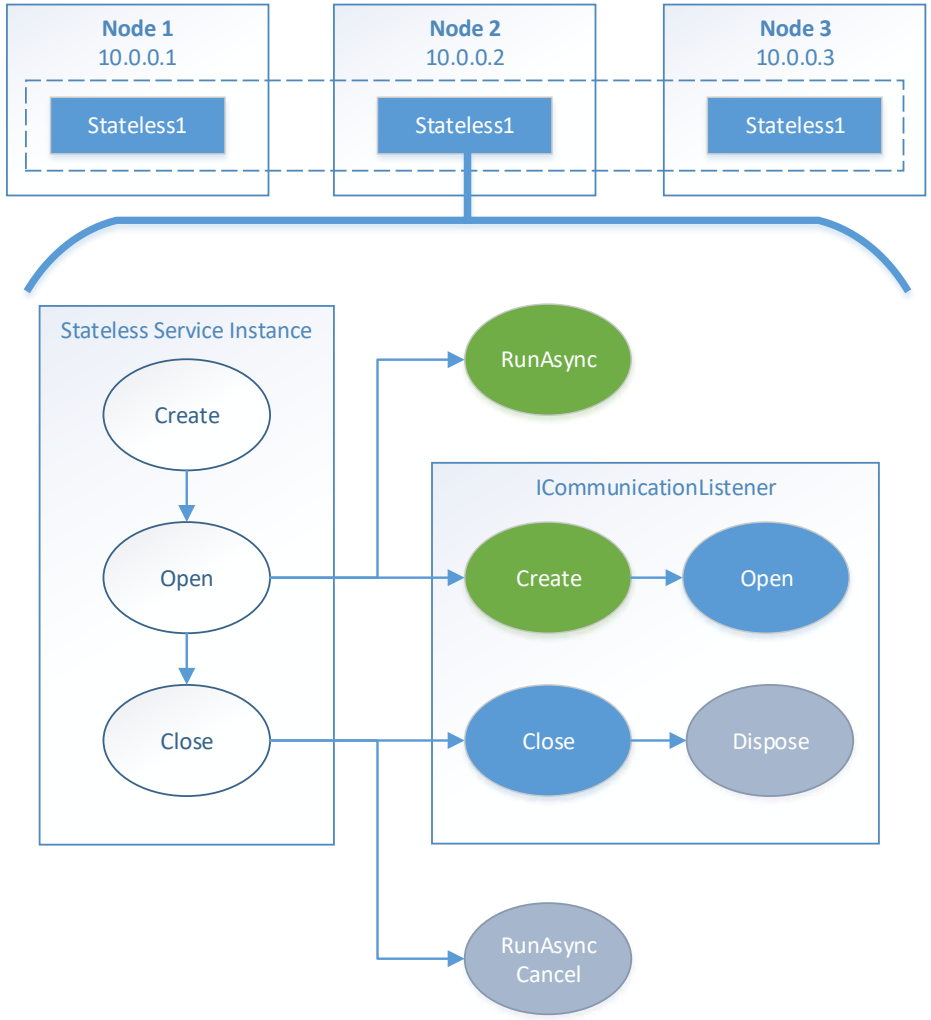
Reliable Services (Stateless & Stateful)

- Programming model benefits
  - Networking naming service support
  - Self-reporting of instance's load metrics
  - Integrate with code, config, & data upgrades
  - Efficiency (higher density): Multiple services (types & instances) in a single process
  - Your class object is notified of service start/stop - similar to Azure Cloud Services' RoleEntryPoint

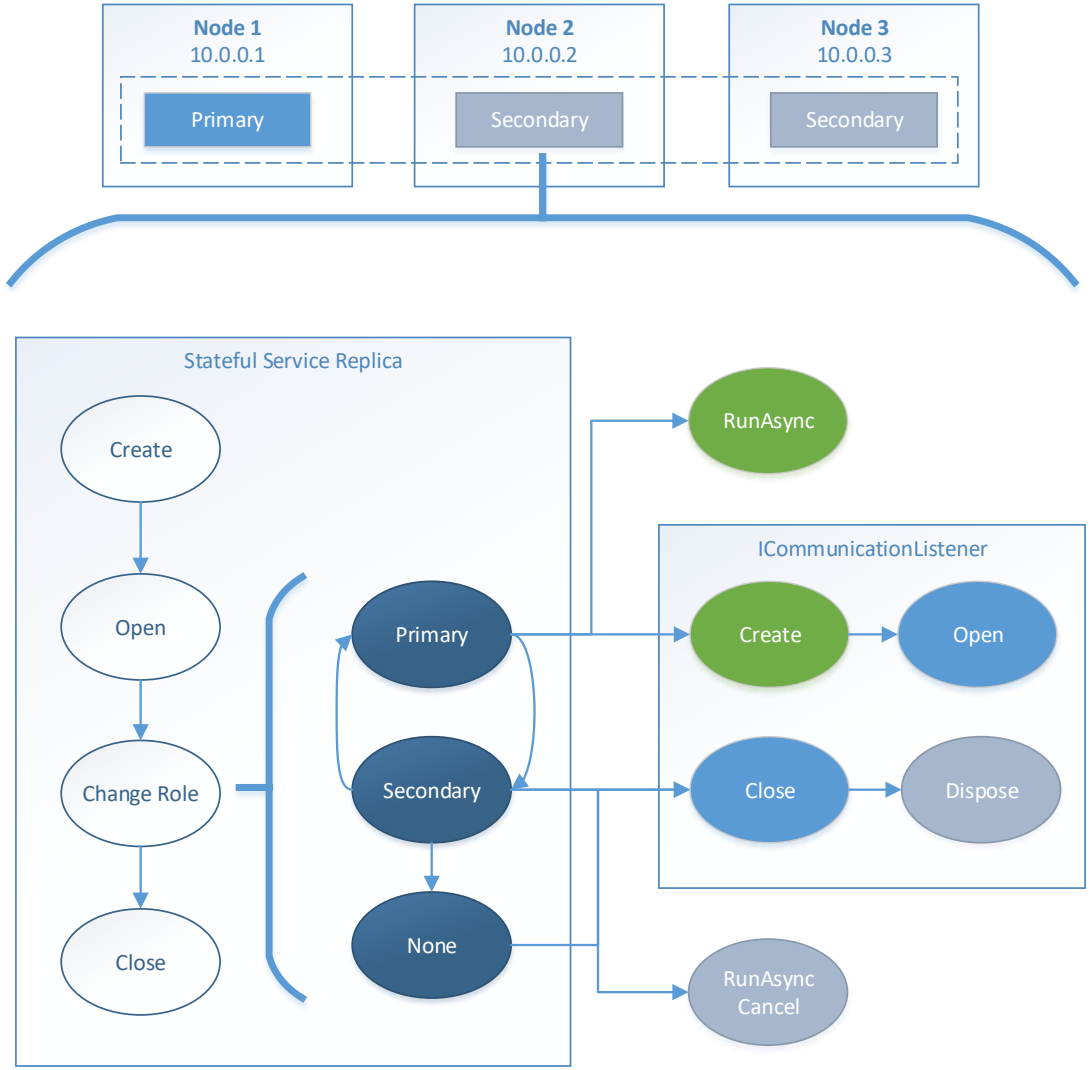
Types of Reliable Services

- Stateless Reliable Services
- Stateful Reliable Services

Stateless service lifecycle



Stateful service lifecycle



Reliable Services Service Lifecycle Operations

Reliable services provides a simple life cycle to plug in the code

- CreateReplicaListeners/CreateServiceInstanceListeners – determines the protocol the service will use to listen for traffic
- RunAsync - Business logic to runs when service starts ~ generally not for a long running process
- OnCloseAsync – graceful close of service
- OnAbort – abrupt halt to a service

Visual Studio's Service Template

- App project has
  - "Services" node referencing 1+ service projects
  - Application Parameters: Local.1Node.xml, Local.5Node.xml & Cloud.xml
  - Publish Profiles: Local.1Node.xml, Local.5Node.xml & Cloud.xml
  - Scripts: Deploy-FabricApplication.ps1
  - ApplicationManifest.xml
  - Right-click to Build, Edit manifest versions, Package, Publish

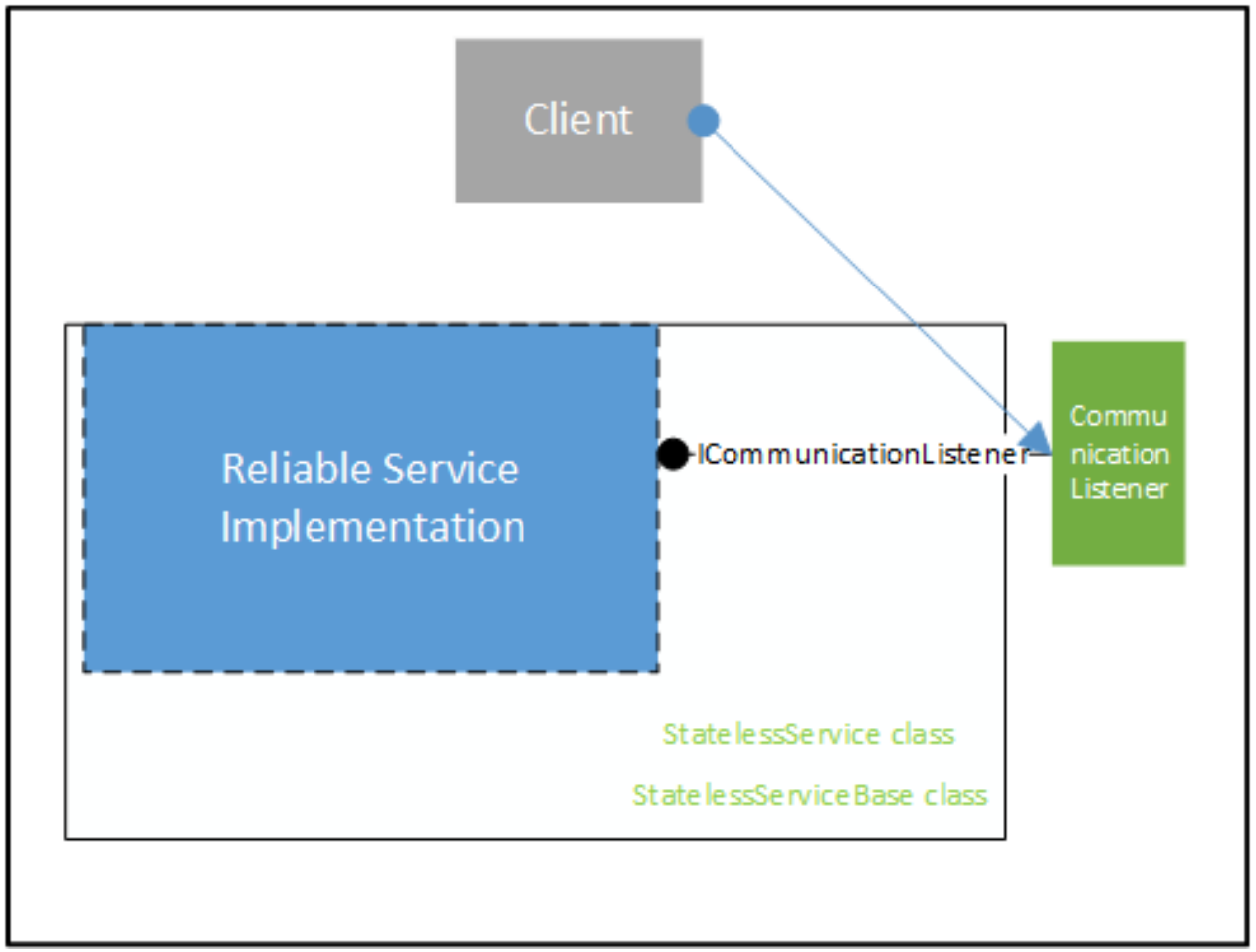
Microsoft

Introduction to Service Fabric

Stateless Reliable Services

Microsoft Services

Architecture - Stateless



- No state is maintained
- Replication of service deploys service in different nodes

Stateless Service Base Class

```
namespace Microsoft.ServiceFabric.Services.Runtime {
   public abstract class StatelessService {

      // Override this for a polling service
      // IMPORTANT: SF signals cancellationToken when moving/upgrading instance; honor this!
      protected virtual Task RunAsync(CancellationToken cancellationToken);

      // Override this for a listening service
      protected virtual IEnumerable<ServiceInstanceListener> CreateServiceInstanceListeners();

      protected virtual Task OnCloseAsync(CancellationToken cancellationToken);  // Graceful close (Dispose)
      protected virtual void OnAbort();                                          // Non-graceful close
   }
}
```

What are Stateless Services used for?

- Web API interfaces ~ no UI
- Services where data is going to be stored externally but not with the service
- Authentication layer for OWIN

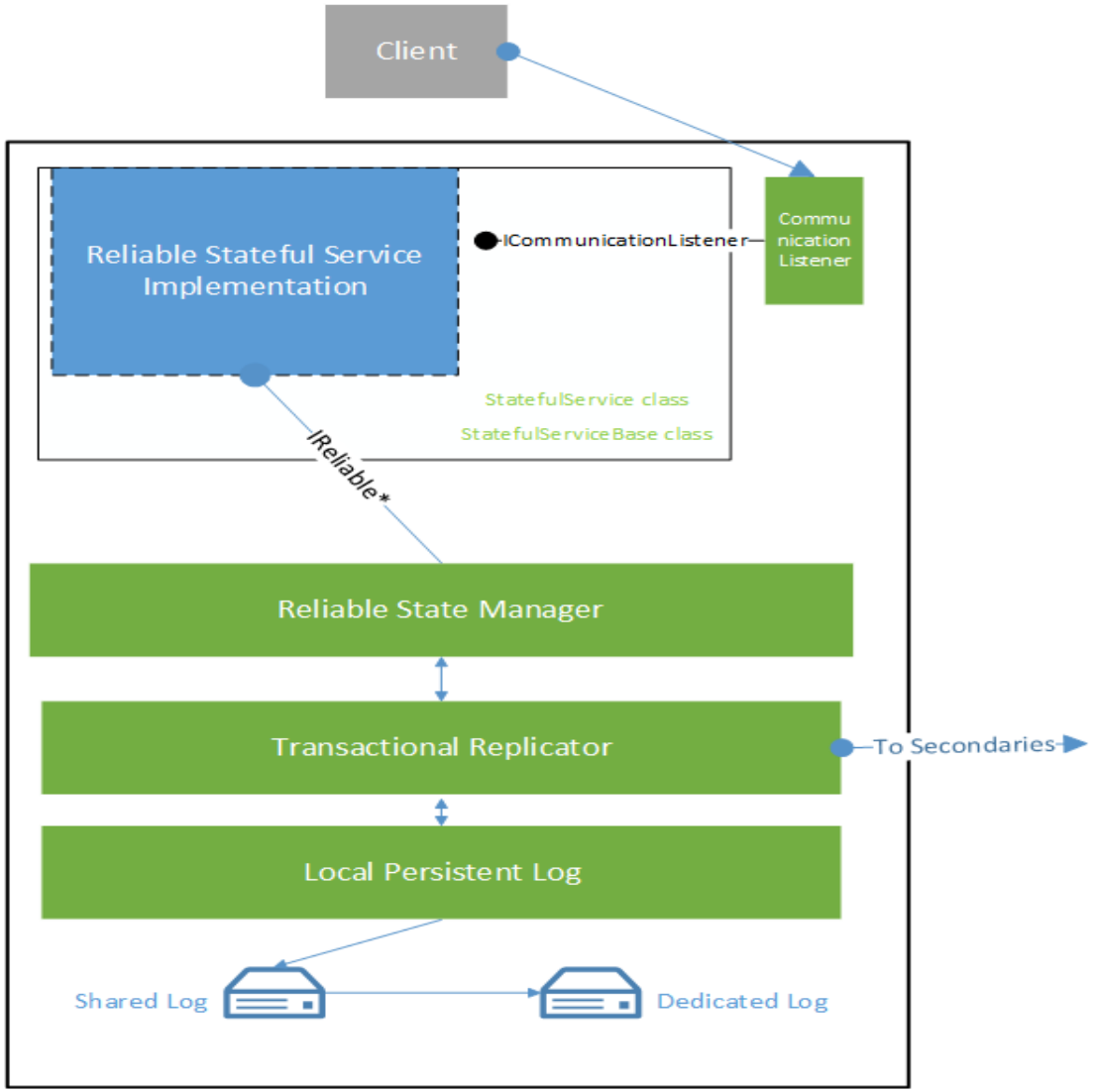
Microsoft

Introduction to Service Fabric

Stateful Reliable Services

Microsoft Services

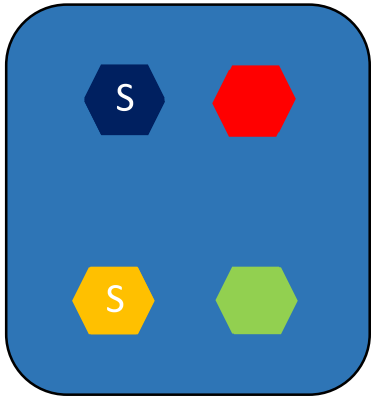
Architecture



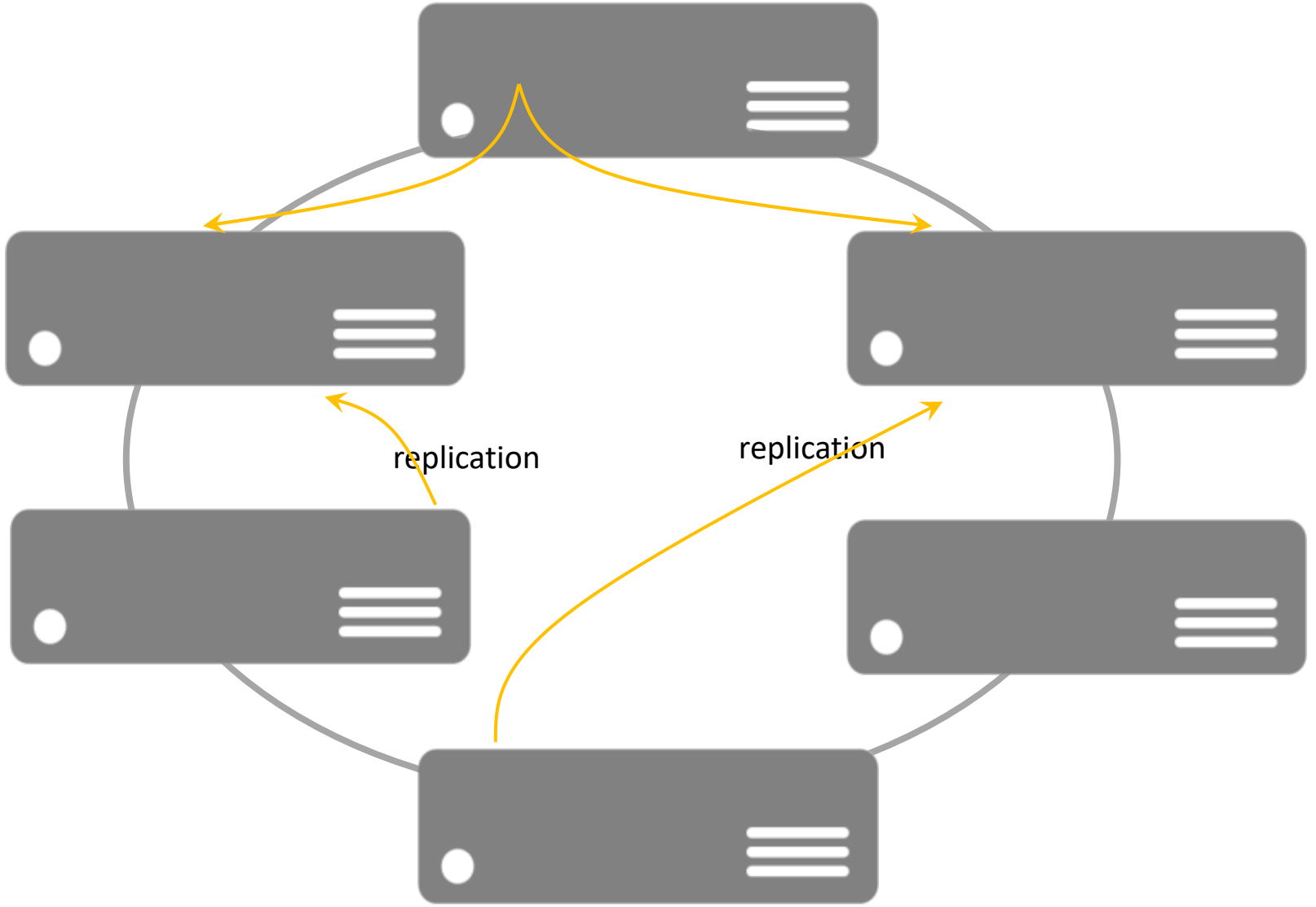
- Persists the state across multiple replicas
- Uses Reliable Collections
- It has one primary and multiple secondary replicas
- Provides partitioning scheme

Stateful Services

Application with four services



Replica placement respects FDs/UDs assigned.



Stateful Reliable Services

- Provides Reliable Dictionary/Queue collection classes
  - Enables low-latency access to HOT data
  - Reduces dependency on external storage services
  - Partitioned for data scalability
  - Replicated for availability
  - Transacted (within a partition) for ACID semantics
  - Persisted for quick node failure recovery

Microsoft

Introduction to Service Fabric

Reliable Actors

Microsoft Services

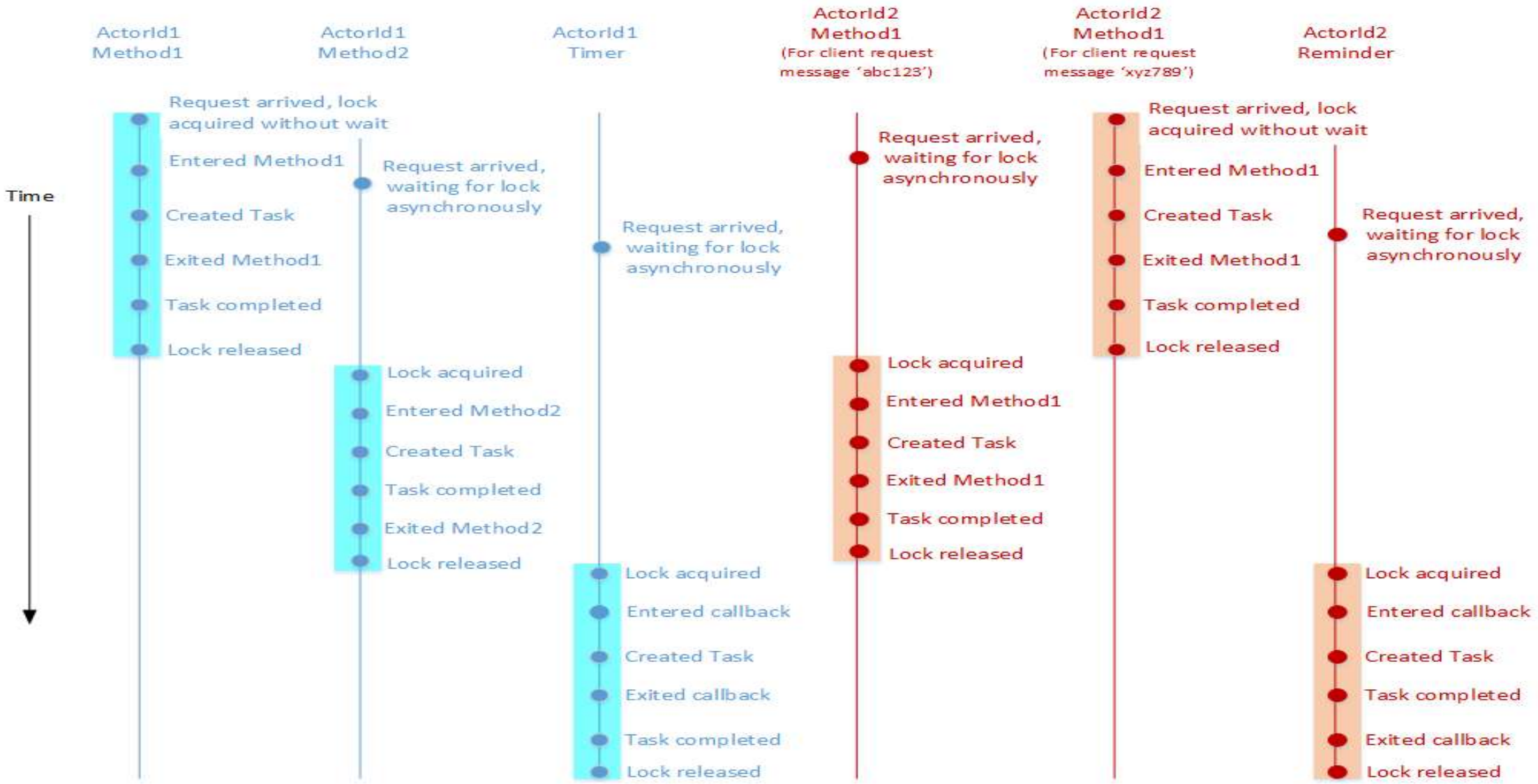
Many Actor Models Provide 2 Main Features

- Turn-based threading
  - Pro: No concurrency issues in a single Actor
  - Con: Hurts scalability as no operations execute simultaneously (including reads)
    - New requests are blocked while current request runs
    - You must ensure that all operations are short; beware I/O requests to other Actor/server
    - Can't query an Actor's progress while it is in progress
- Automatic partition placement
  - Pro: Actors are distributed across partitions giving location transparency
  - Con: All Actor access requires network communication
    - Client must be .NET since protocol is proprietary
    - Causes performance issues when doing query/batch operations
    - Cross-actor transactions are not supported
    - Note: You must still decide # partitions based on Actor resource requirements

Service Fabric Reliable Actor model

- Based on the Actor Pattern
- Always stateful – instance ID guarantees that calls to same instance ID gets same actor
- Asynchronous single threaded model
- API to build stateless and stateful objects through the virtual actor programming mode
- Provides actor proxy for actor-to-actor and client-to-actor communication
- Runtime automatically maintains lifecycle of actors
- https://azure.microsoft.com/en-us/documentation/articles/service-fabric-reliable-actors-introduction/

Actor turn-based concurrency



Reliable Actor timers

- Simple wrapper around the .Net timer type
- Respects turn-based concurrency of actor model
- Must be registered and unregistered by code in the actor
- When a timer is due the Actor runtime will call your callback method
- No other Actor methods or timer/reminder callbacks will be in progress until this callback completes
- Actor runtime saves actor objects state when callback is complete
- All timers are stopped (unregistered) whenever garbage collection takes place

Reliable Actor Reminders

- Similar to Timer, respects turned-based concurrency
- Reminders are triggered under all circumstances – until you unregister them (timers will stop triggering after garbage collection)
- The Actor runtime persists information about the actor's reminders
- Must be registered and unregistered by code in the actor
- When a reminder is due the Actor runtime will call your callback method
- Actor must implement the IRemindable interface ~ you will need to implement the ReceiveReminderAsync method
- Actor runtime saves actor objects state when ReceiveReminderAsync is complete

Reliable Actor reentrancy

- Reliable actors support logical call context-based re-entrancy (default)
  - ActorA -> ActorB->ActorC…ActorC can call back to ActorA
- Any other message that is a part of a different context will be blocked
- Two options for Actor reentrancy
  - LogicalCallContext (default)
  - Disallowed – disables reentrancy
- Reentrancy is usually configured when the service is registered

Actors - Keep the following guidance in mind

- Actors are good for small individual pieces of code
- Actors are good for smaller discrete things where you do not build large graphs of things
- Actors are particularly poor for looking across a wide range of devices etc
- Best practice is to have a reliable dictionary store all your messages that come in and then you spin actors from all those
- Start by first using the Service model and understand the service model well, then consider using the actor model
- If you have an actor that stores state and you are finished with the actor, make sure you delete the actor to delete the state

Demonstration

Stateless and Stateful Microservices

Microsoft

Introduction to Service Fabric

Service and Application Upgrades

Microsoft Services

Rolling Upgrades

- Services are taken down during code package updates, but not during config or data packages upgrades
- Upgrades are performed on cluster Update Domains
- Upgrades Types
    - Monitored Auto: By Upgrade Domain (UD) one by one , health checks and failure action: (can do auto roll back)
    - Unmonitored Auto: Upgrade Domain (UD) one by one, no health checks
    - Unmonitored Manual: You manually upgrade each Upgrade Domain (UD)

Microsoft

Introduction to Service Fabric

Capacity Planning

Microsoft Services

Service Fabric Clusters

- Clusters in Azure
  - A set of VM scale set machines
  - Deployed through the Azure Portal, ARM templates or PowerShell
- Standalone clusters on Windows Server
  - You can deploy and run Service Fabric in ANY environment that allows interconnected Windows Server computers
  - A setup package is provided by Microsoft with sample scripts/templates https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-cluster-creation-for-windows-server

Capacity Planning: How Many Nodes?

- It's all about getting your work done on PC/VM with limited hardware resources
- Think about near & long term resource requirements
- Since each PC/VM has limited resources, you must spread your workload across many nodes to
  - Get data to fit (RAM/Disk); most important for stateful apps (depending on amount of data)
  - Get speed (CPU/network); most important for stateless apps

Capacity Planning: How Many Nodes? (con't)

- Judging CPU/network for apps is hard: depends greatly on the computation complexity & specific I/O patterns
  - However, for stateless apps, you can monitor performance and simply increase/decrease nodes to accommodate the load
- Judging RAM/disk is easier; estimate data sizes, costs, and expected trends with a spreadsheet
  - GB per PC/VM can be disk or RAM (whichever you desire)
- Consider app's future & management issues (failures & upgrades impacting scalability, etc.)
- Service Fabric VM Cost Calculator

Resources

- Download the Service Fabric developer SDK

  http://aka.ms/ServiceFabricSDK

  http://aka.ms/ServiceFabric
- Download the samples for GitHub

  http://github.com/Azure/ServiceFabric-Samples
- Learn from the tutorials and videos

  http://aka.ms/ServiceFabricdocs