Building resilient microservices with Azure Service Fabric

Ronald Harmsen

How big is a microservice?

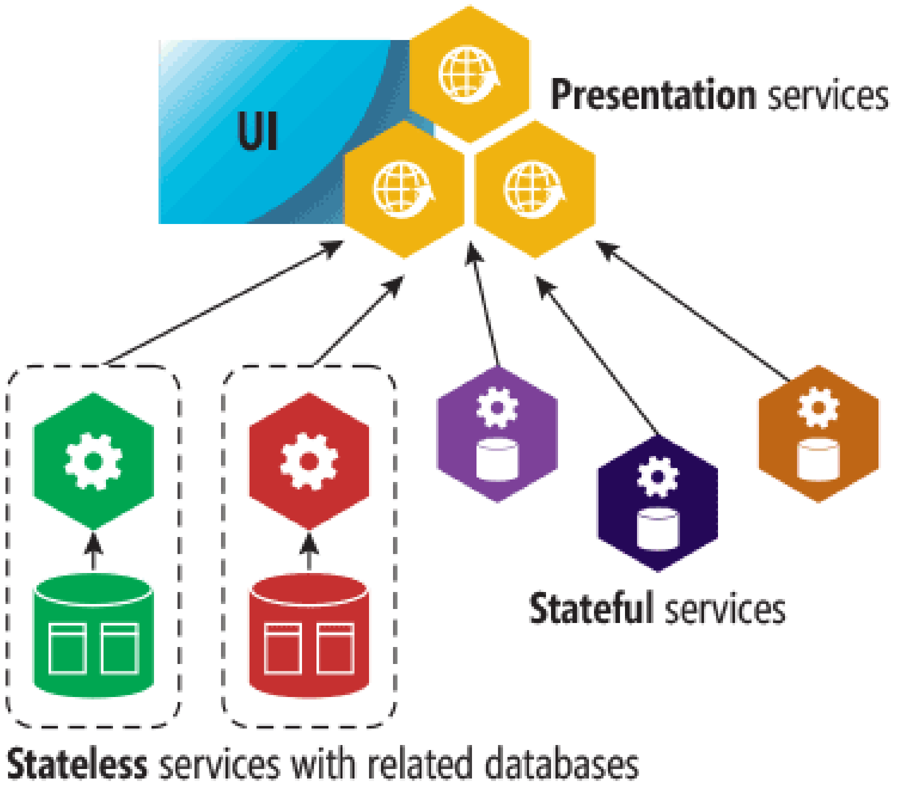
Microservices: a definition

"The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of ***independently deployable*** *services. While there is* <u>*no precise definition*</u> *of this architectural style, there are certain common characteristics around organization around* ***business capability****,* ***automated deployment****,* ***intelligence in the endpoints****, and* ***decentralized control of languages and data****. "*

Martin Fowler

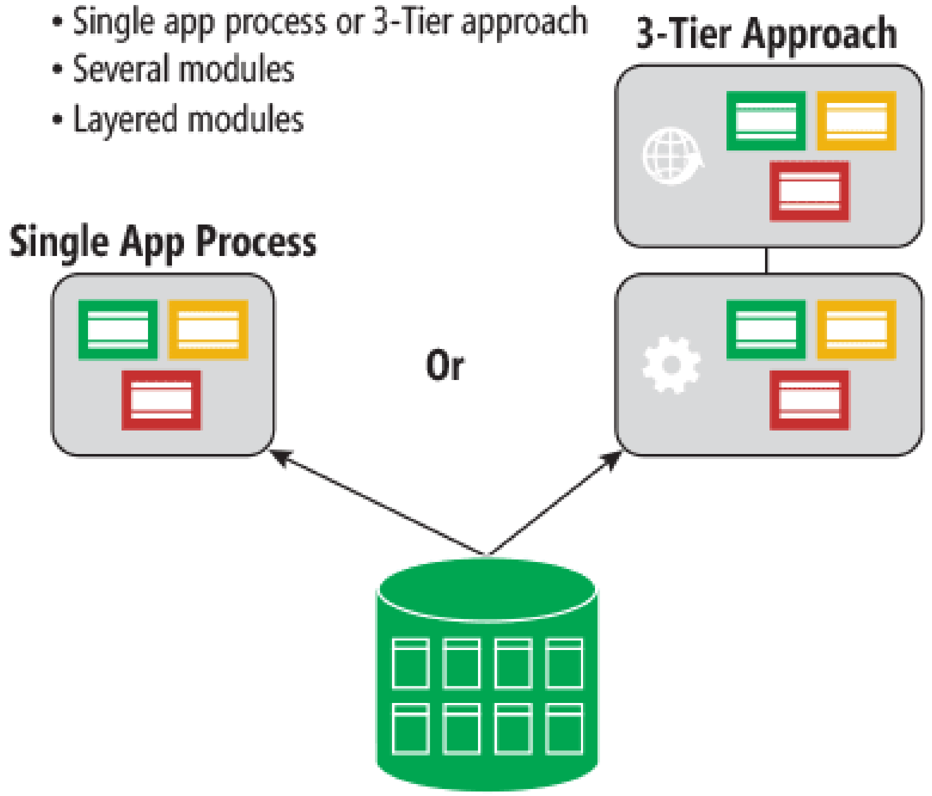
DevSum18

Microservices Approach



Model/Database per Microservice

Traditional Application



Single Monolithic Database

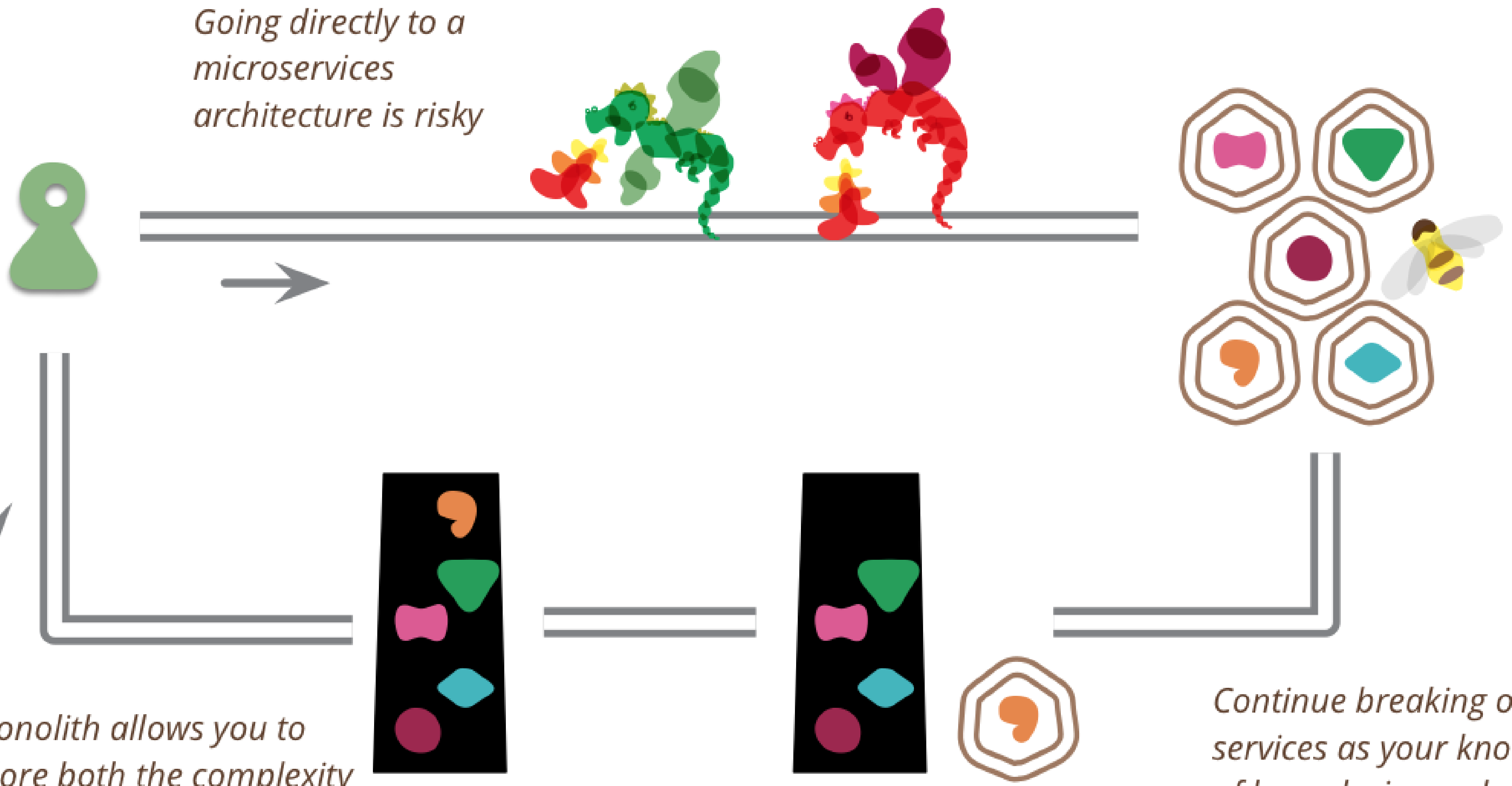
Do you need microservices?

- Need to deploy application parts independently?
- Different scaling or technology needs
- Continuous deployment
- Prepared for the increased complexity?

Microservices Envy

Starting a project without cause- and affliction

Monolith first development

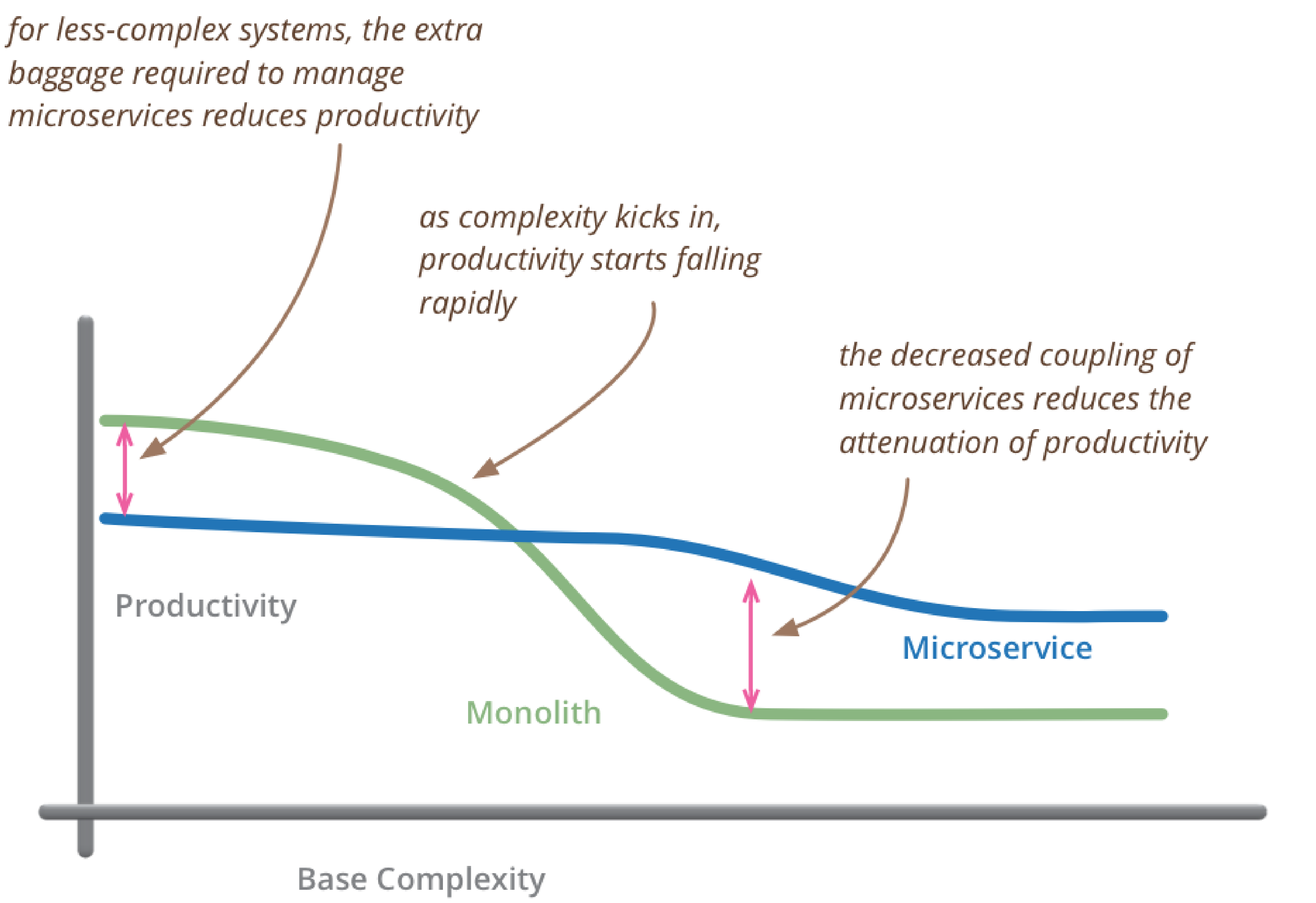


Going directly to a microservices architecture is risky

A monolith allows you to explore both the complexity of a system and its component boundaries

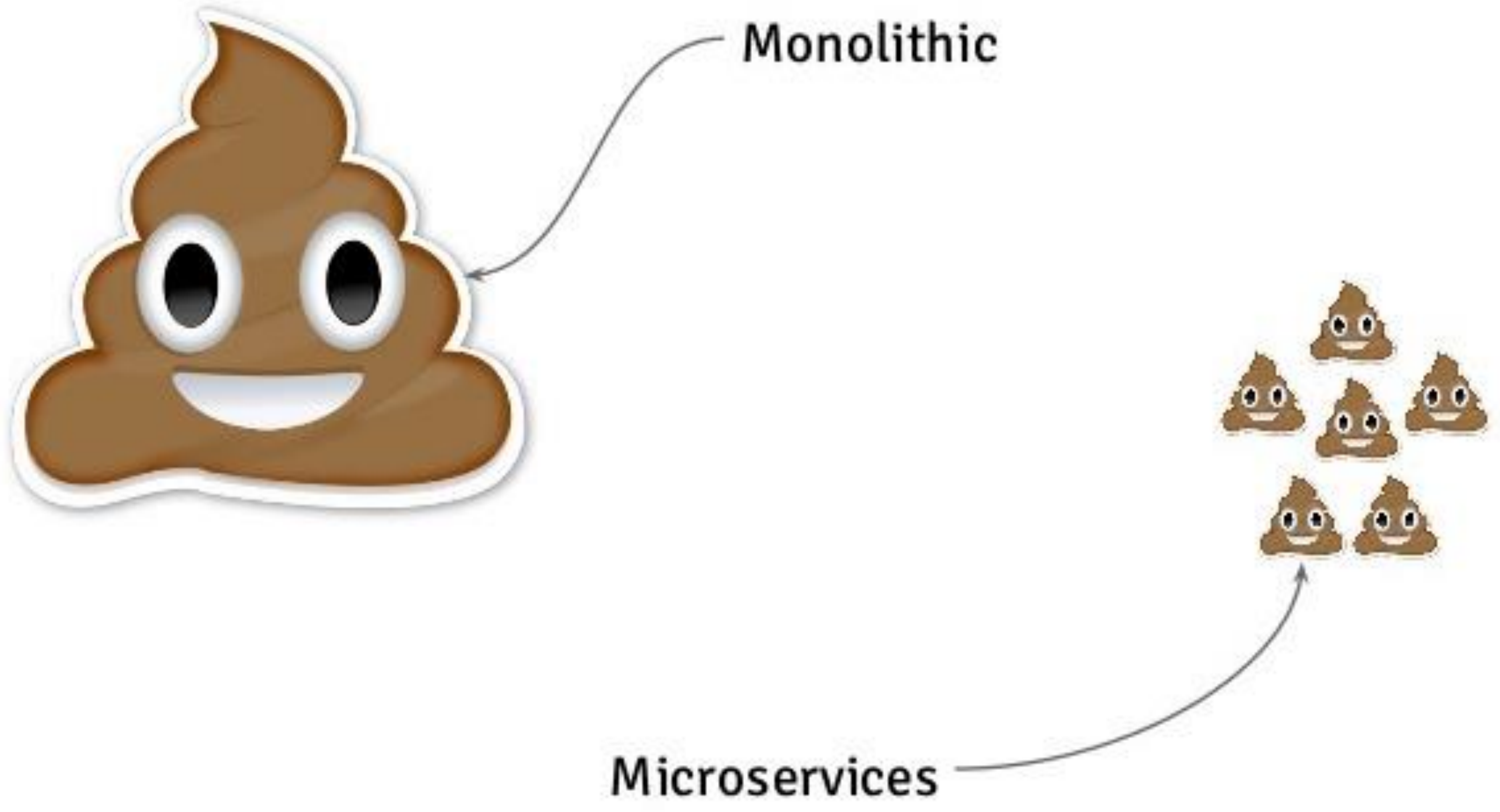
As complexity rises start breaking out some microservices

Continue breaking out services as your knowledge of boundaries and service management increases



but remember the skill of the team will outweigh any monolith/microservice choice

Monolithic vs Microservices



Microservice Drawbacks

- Tracking down failures is a nightmare
- Hard to debug
- Logging
- Services call each other, tracing messages?
- Managing security

Hosting microservices

HEROKU

amazon web services

DigitalOcean

docker

DevSum18

Microsoft Azure Service Fabric

DevSum18

Azure Service Fabric

- Deployment management system
  (package, deploy, manage)

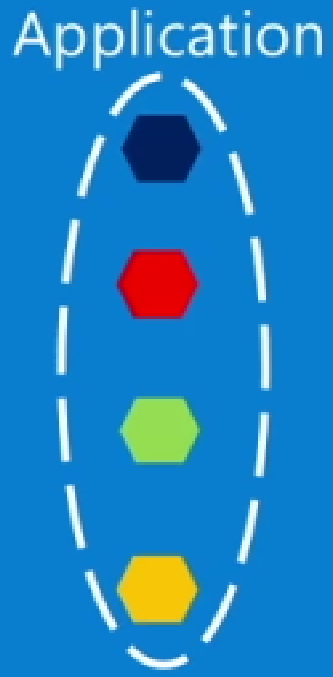
- Abstraction layer over infrastructure

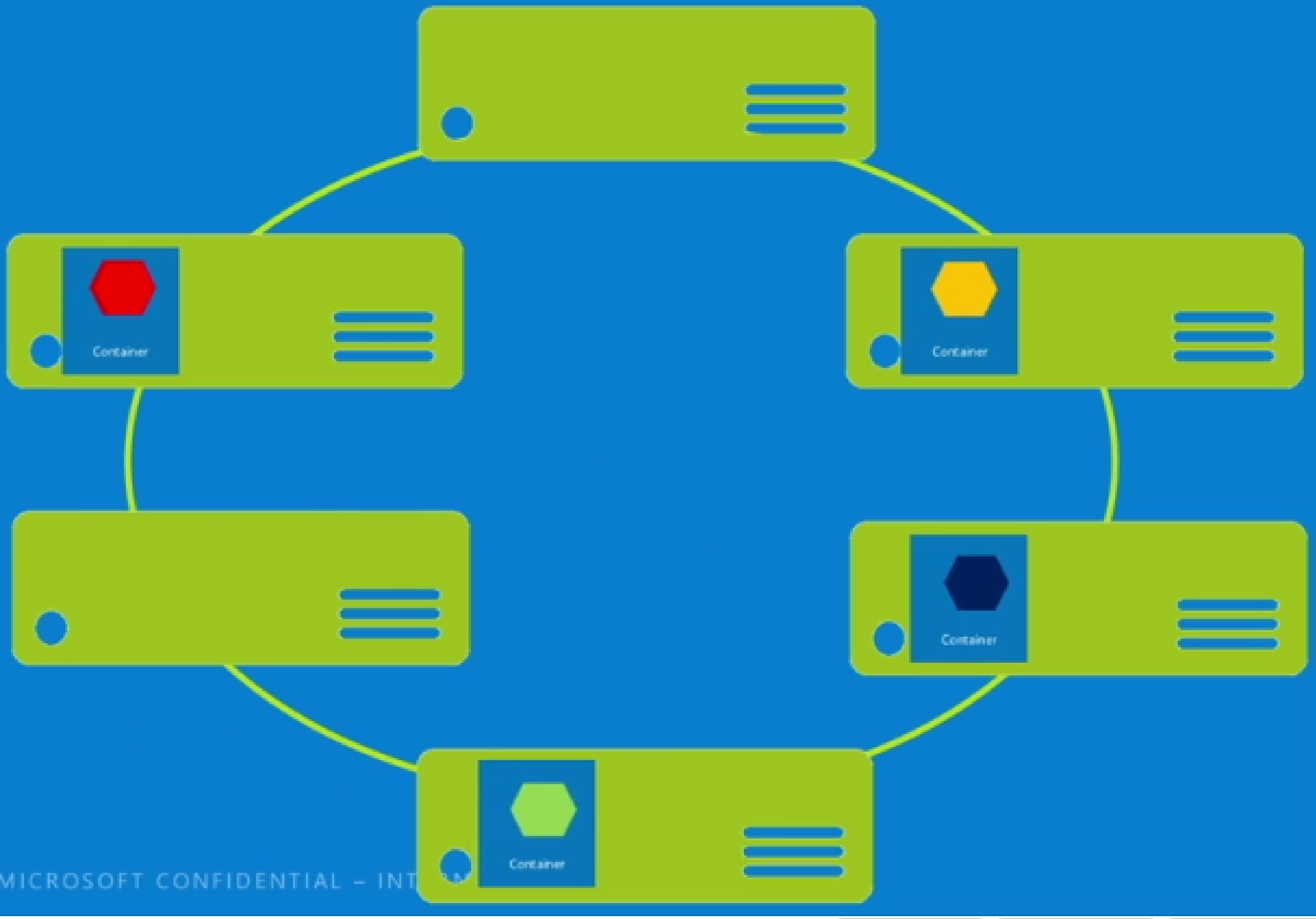
- Build to support highly scalable, distributed and robust micro-services architectures

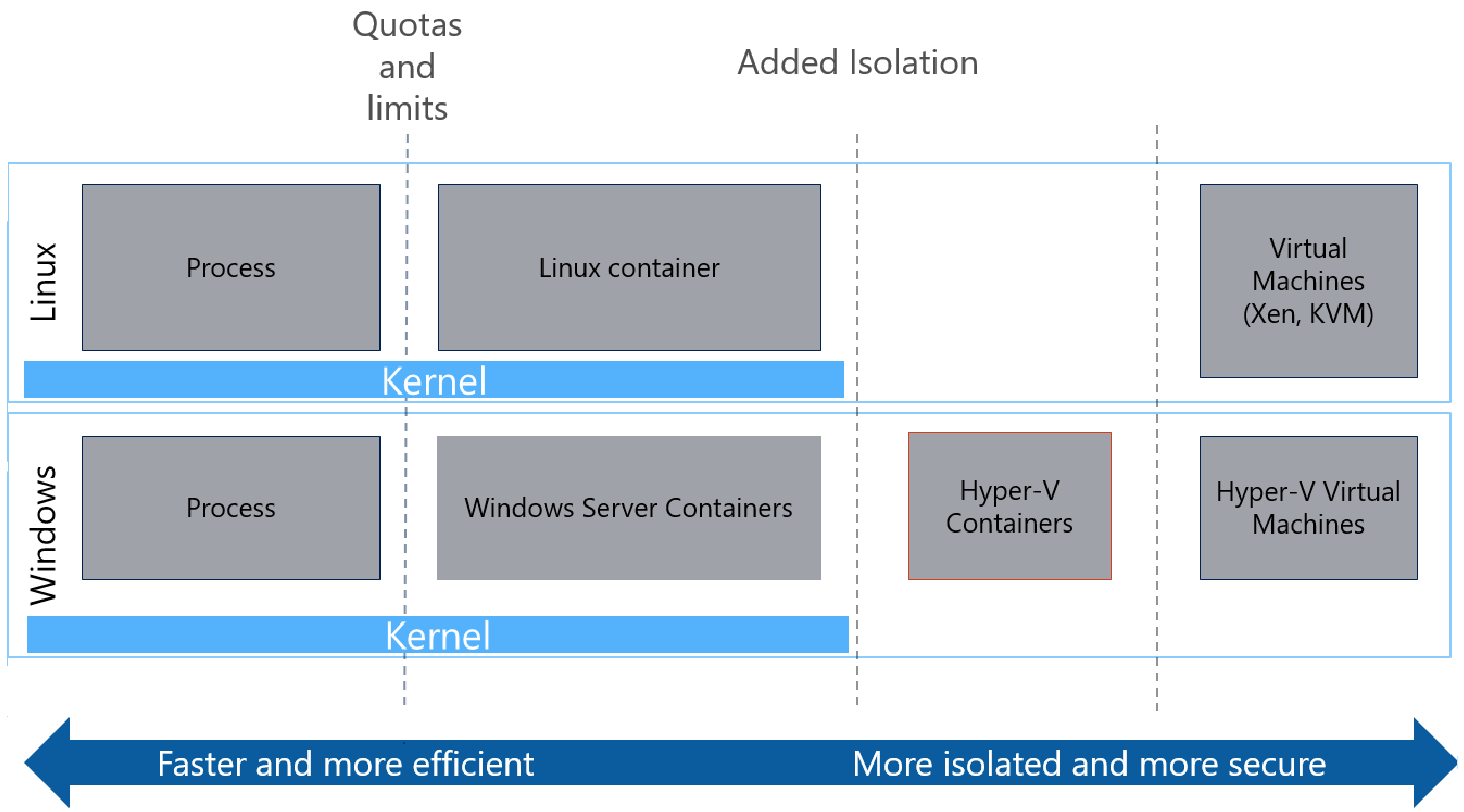
Azure Service Fabric

- Underlies a lot of Microsoft's systems:
  - Skype
  - Cortana
  - SQL Azure
  - Cosmos DB
  - Power BI
  - Event & IoT Hubs
  - Dynamics 365

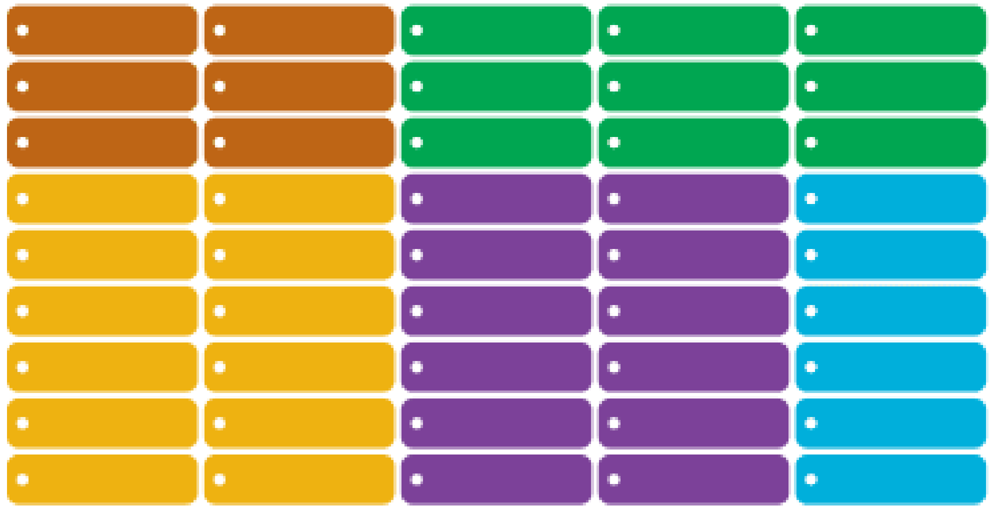
Application: A group of microservices





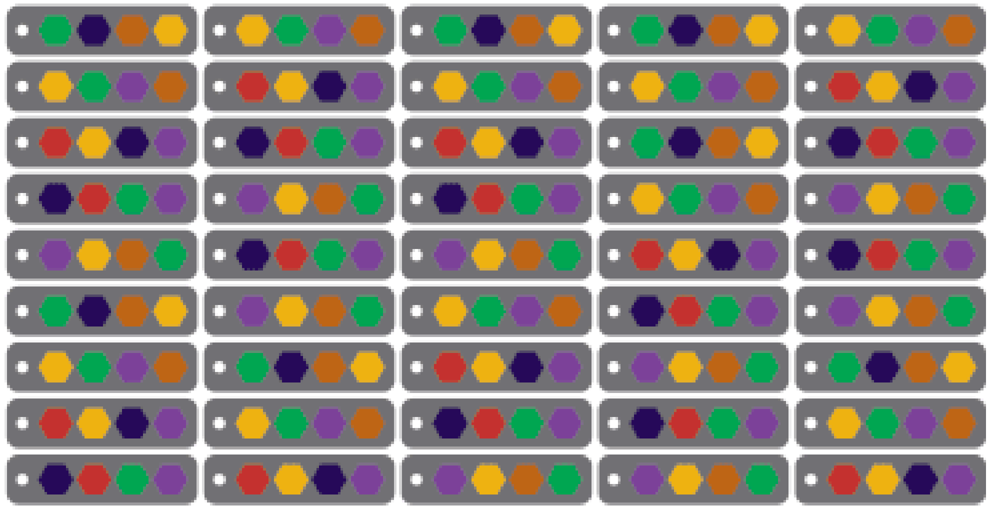

Quotas and limits
Added Isolation
Linux
Process
Linux container
Virtual Machines (Xen, KVM)
Kernel
Windows
Process
Windows Server Containers
Hyper-V Containers
Hyper-V Virtual Machines
Kernel
Faster and more efficient
More isolated and more secure

Azure Cloud Services
(Web and Worker Roles)



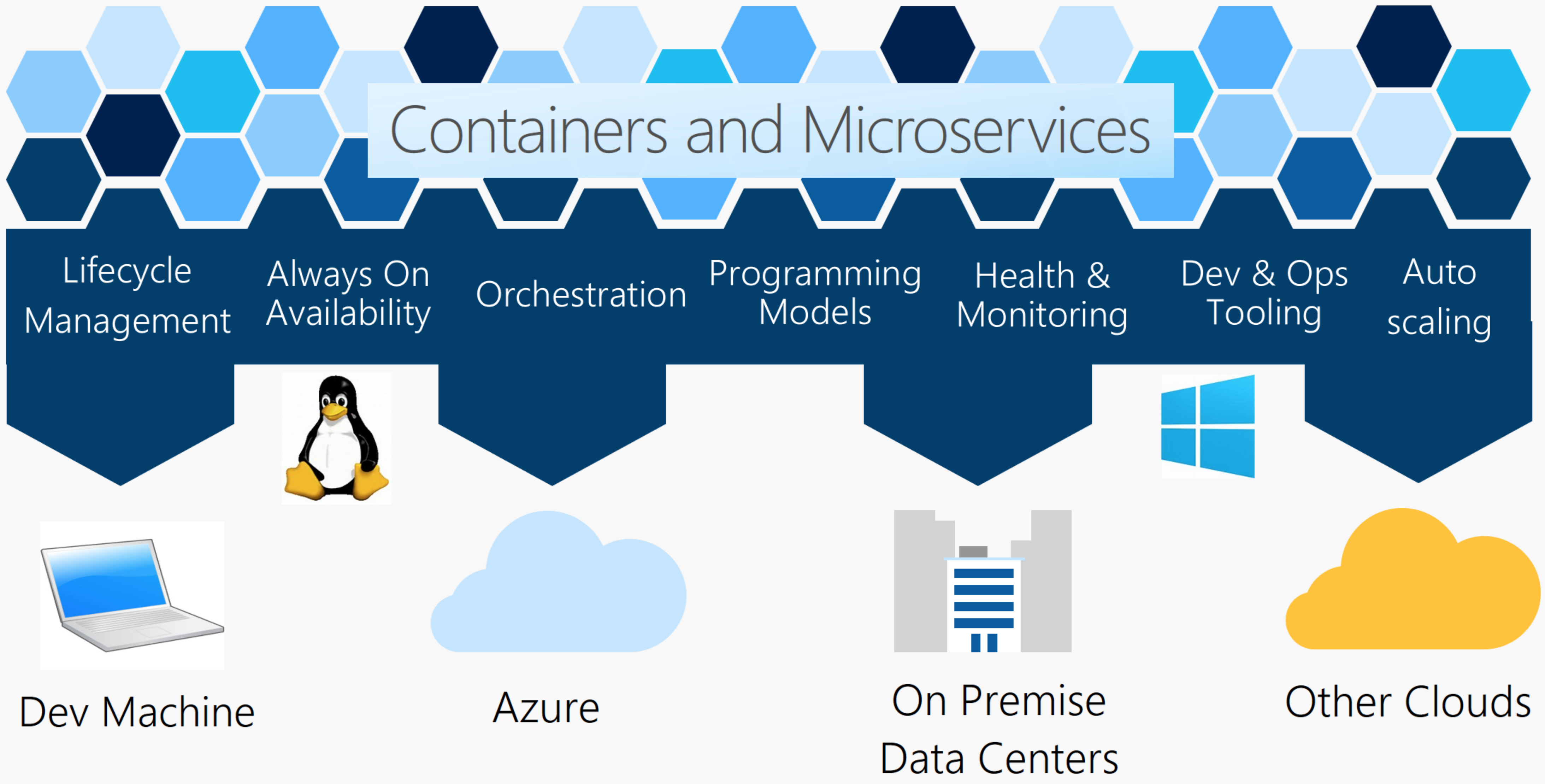
- 1 service per VM with uneven workloads
- Lower compute density
- Slow in deployment and upgrades
- Slower in scaling and disaster recovery

Azure Service Fabric
(Stateless, Stateful or Actor Services)



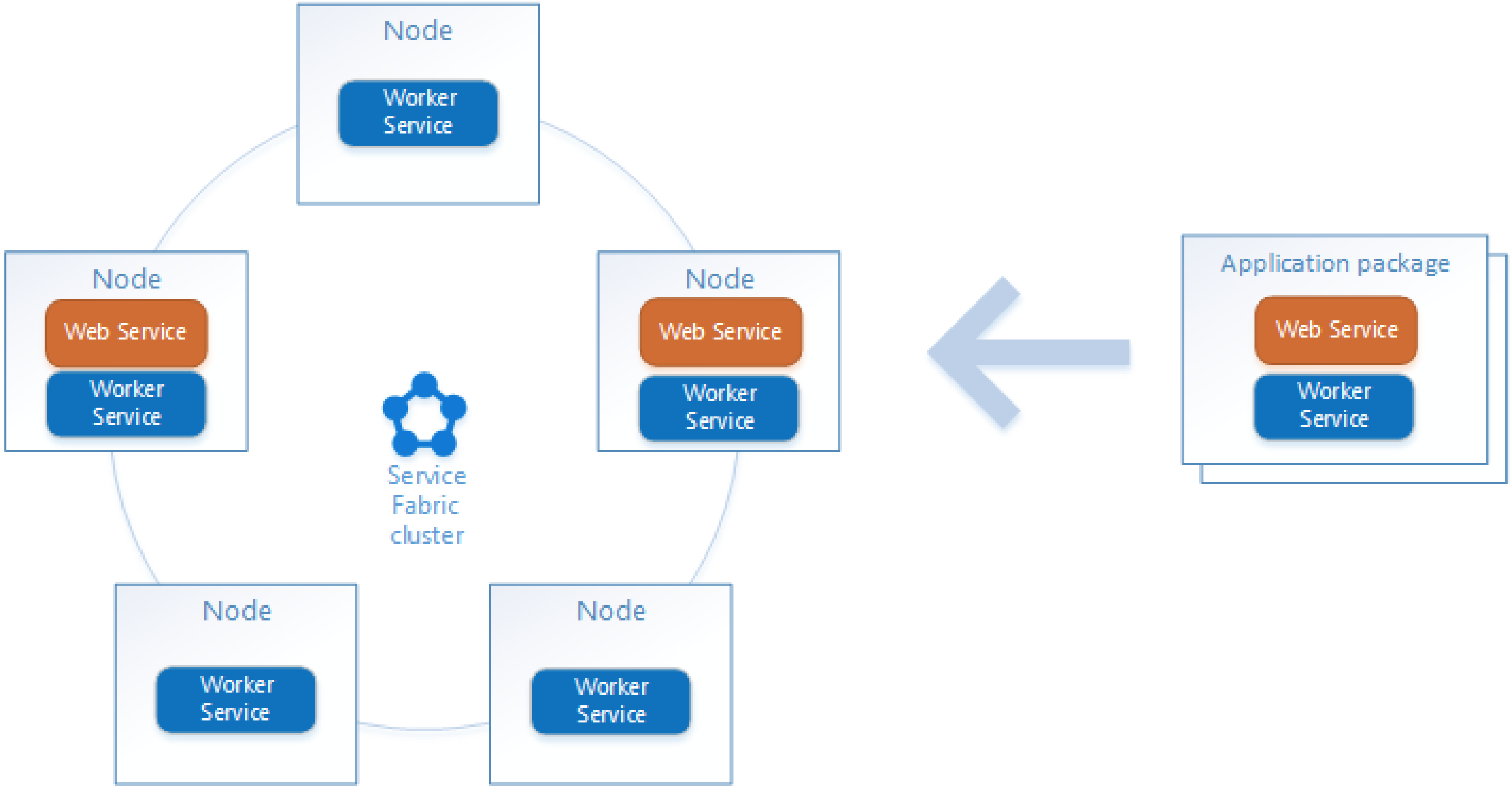
- Many microservices per VM
- High microservices density
- Fast deployment and upgrades
- Fast scaling microservices across the cluster

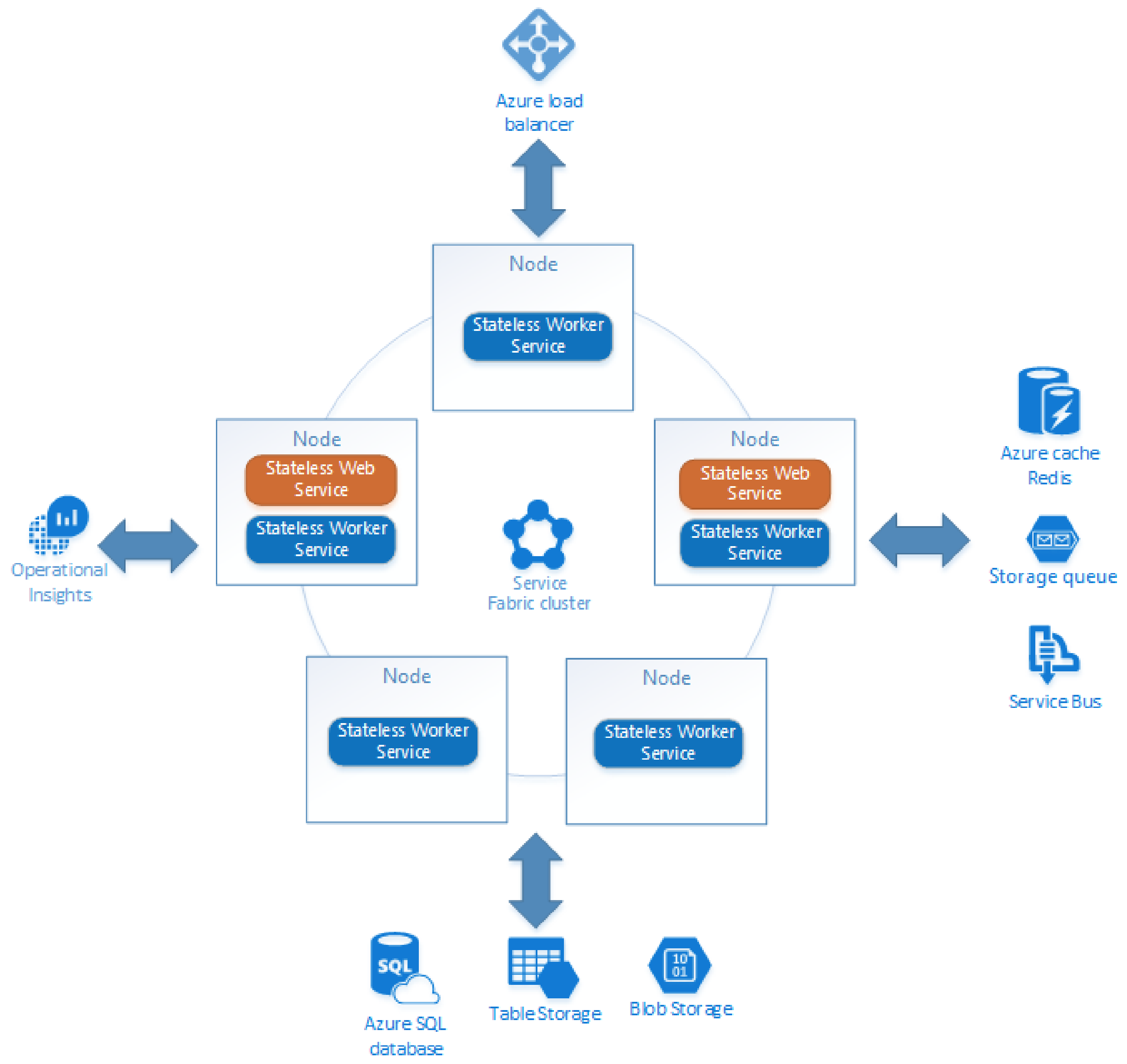
Azure Service Fabric



Built in features

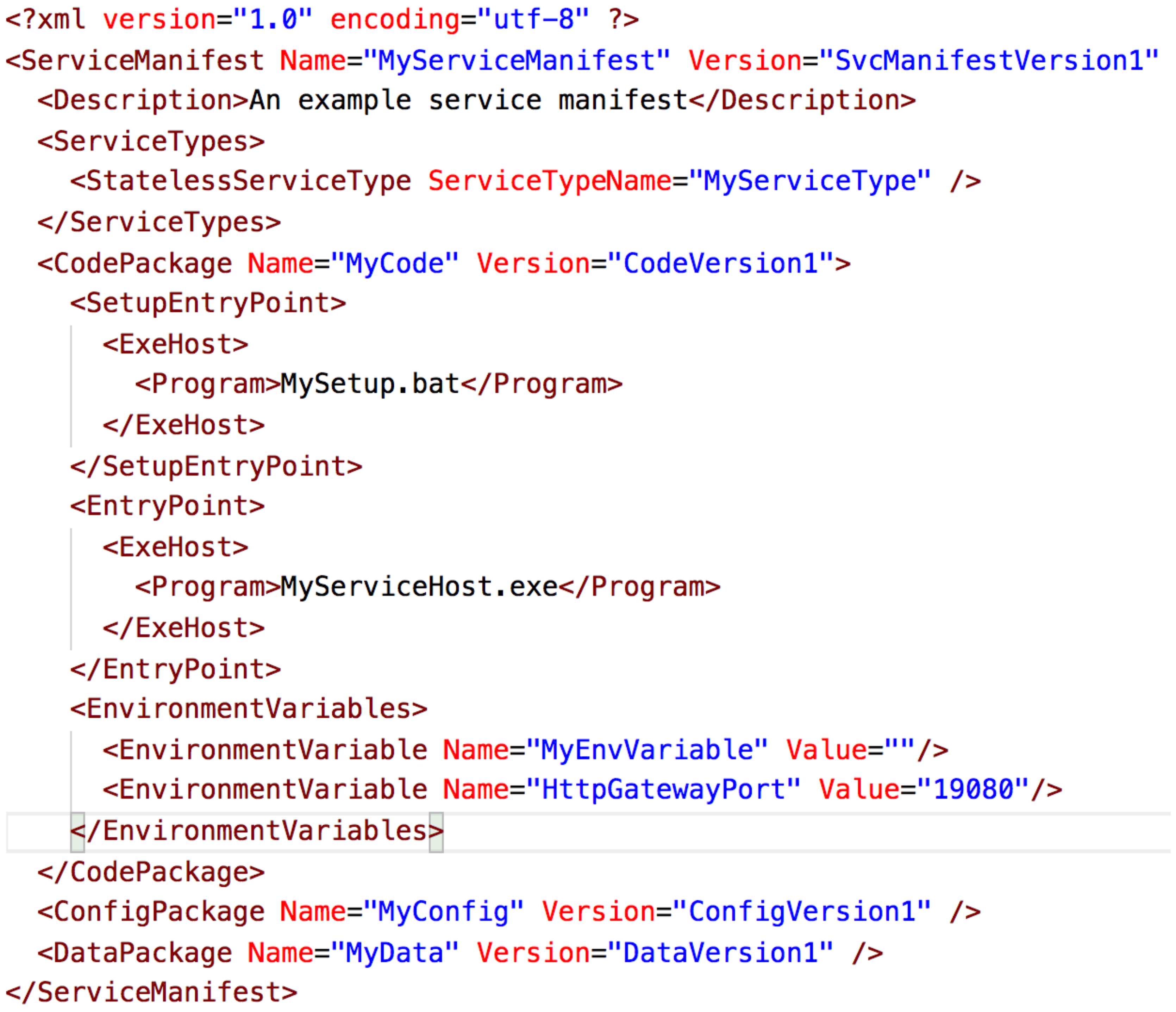
- Automatic Resource Balancing
- Built-in Failover and Replication
- Placement constraints
- Health monitoring
- Storage for stateful services


Node
Worker Service
Node
Web Service
Worker Service
Node
Web Service
Worker Service
Service Fabric cluster
Node
Worker Service
Node
Worker Service
Application package
Web Service
Worker Service


Azure load balancer
Node
Stateless Worker Service
Node
Stateless Web Service
Stateless Worker Service
Node
Stateless Web Service
Stateless Worker Service
Operational Insights
Service Fabric cluster
Azure cache Redis
Storage queue
Service Bus
Node
Stateless Worker Service
Node
Stateless Worker Service
Azure SQL database
Table Storage
Blob Storage

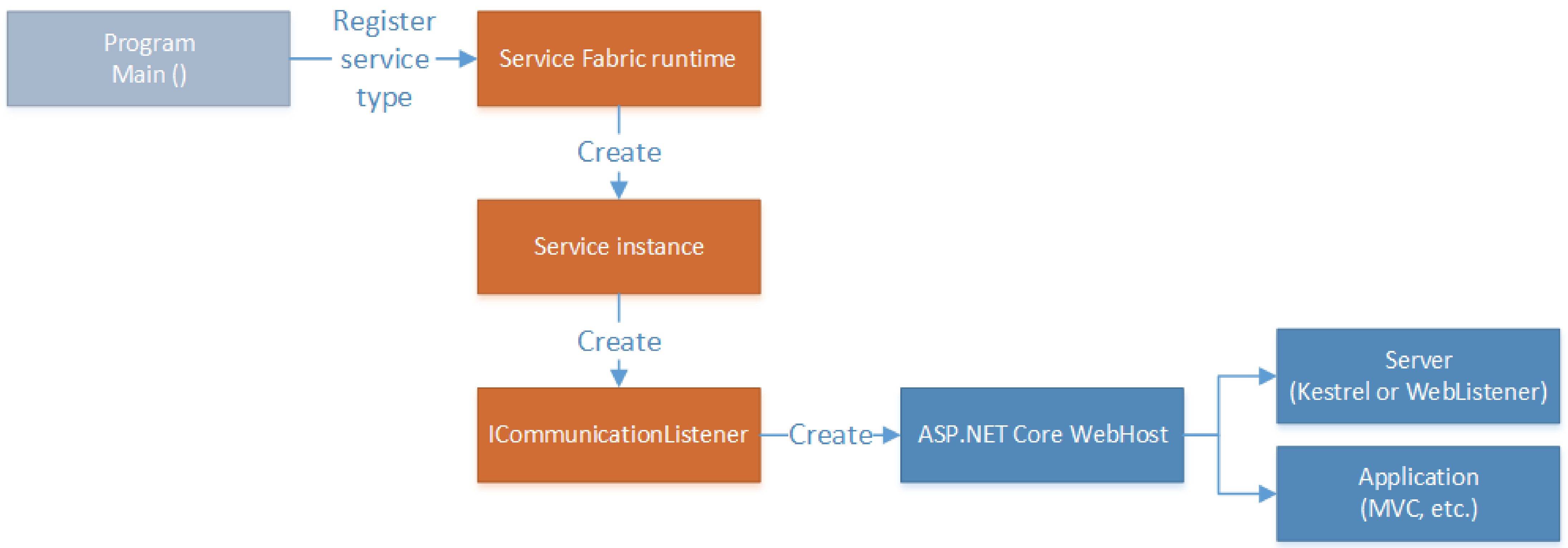
Deployment models

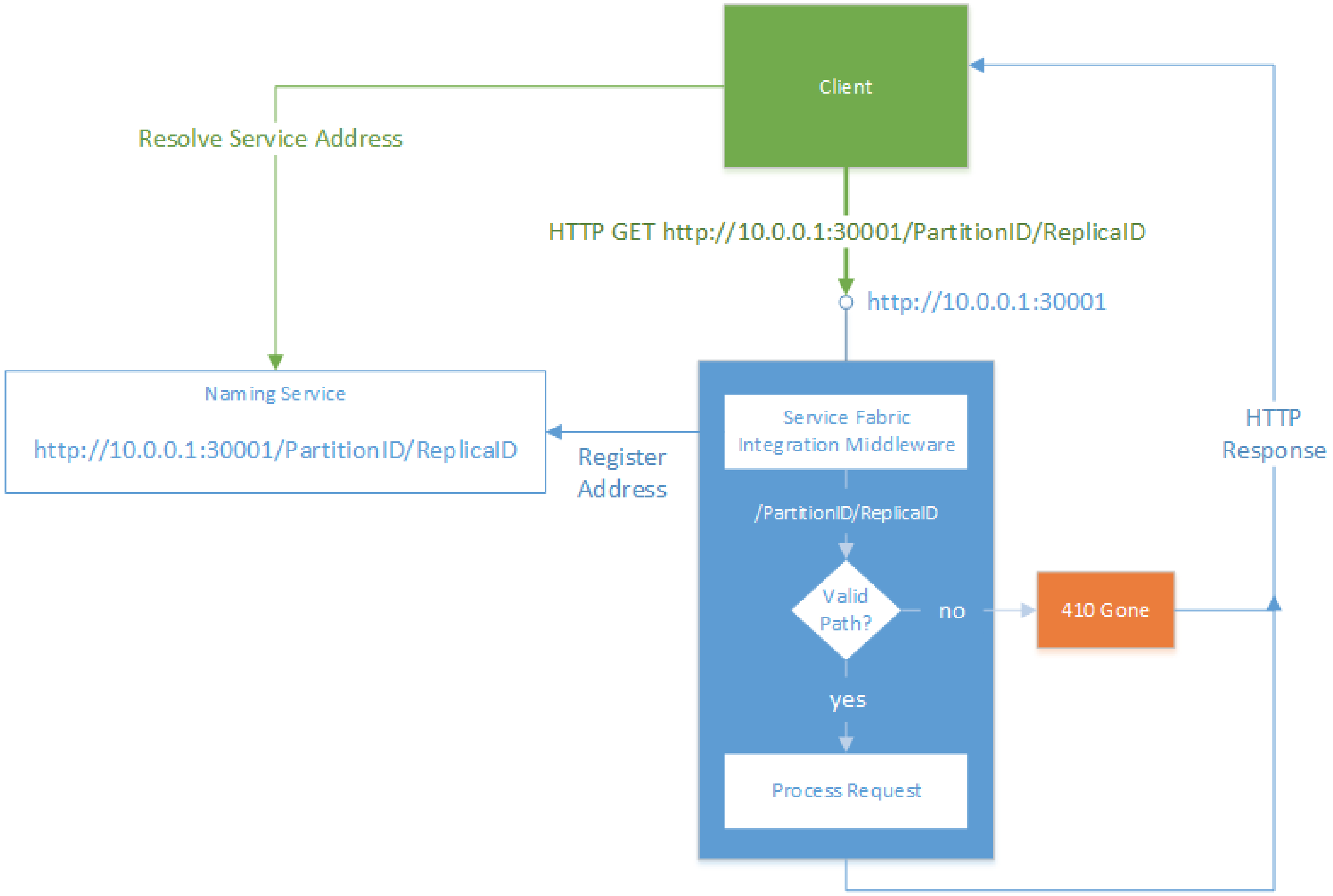
- Hosted executable
- Container
- Reliable service (stateful vs stateless)
- Reliable actor



```xml
<?xml version="1.0" encoding="utf-8" ?>
<ServiceManifest Name="MyServiceManifest" Version="SvcManifestVersion1"
  <Description>An example service manifest</Description>
  <ServiceTypes>
    <StatelessServiceType ServiceTypeName="MyServiceType" />
  </ServiceTypes>
  <CodePackage Name="MyCode" Version="CodeVersion1">
    <SetupEntryPoint>
      <ExeHost>
        <Program>MySetup.bat</Program>
      </ExeHost>
    </SetupEntryPoint>
    <EntryPoint>
      <ExeHost>
        <Program>MyServiceHost.exe</Program>
      </ExeHost>
    </EntryPoint>
    <EnvironmentVariables>
      <EnvironmentVariable Name="MyEnvVariable" Value=""/>
      <EnvironmentVariable Name="HttpGatewayPort" Value="19080"/>
    </EnvironmentVariables>
  </CodePackage>
  <ConfigPackage Name="MyConfig" Version="ConfigVersion1" />
  <DataPackage Name="MyData" Version="DataVersion1" />
</ServiceManifest>
```

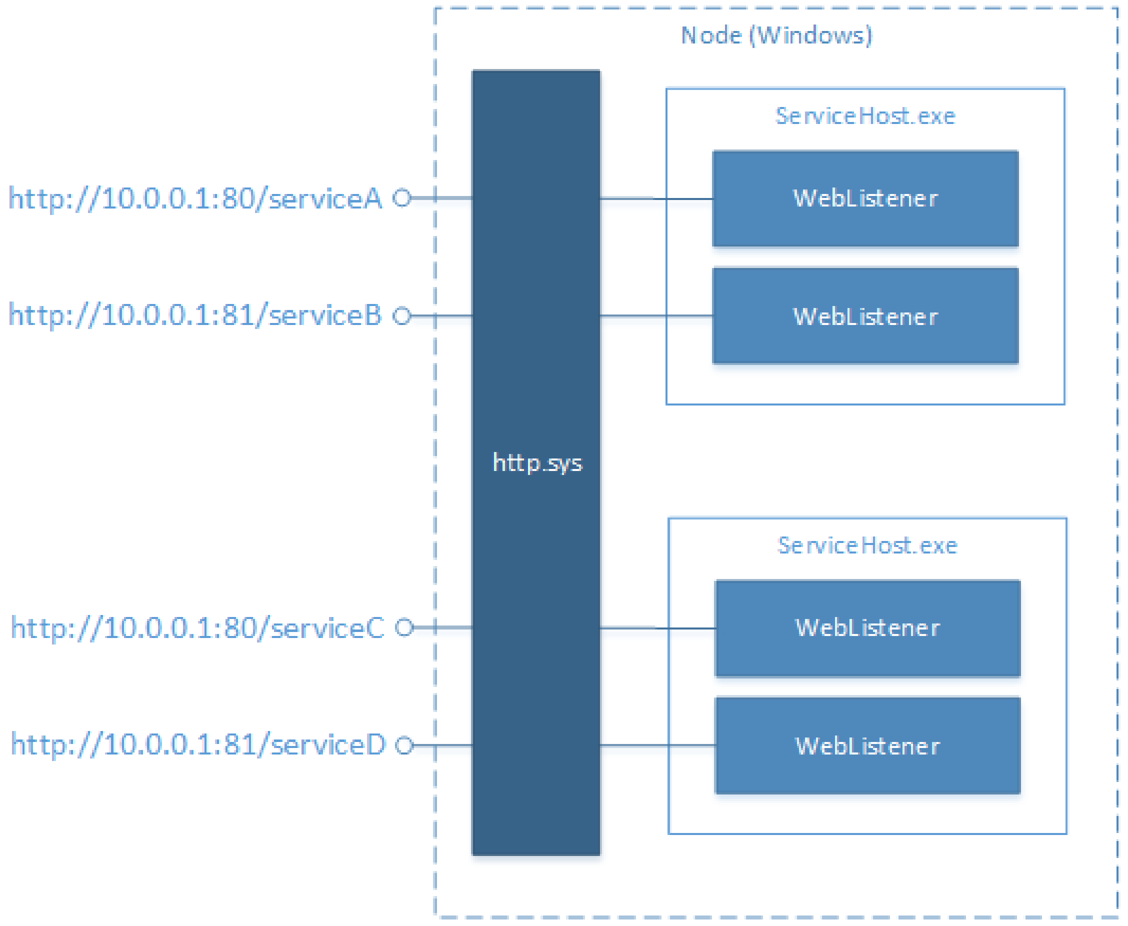
Reliable services



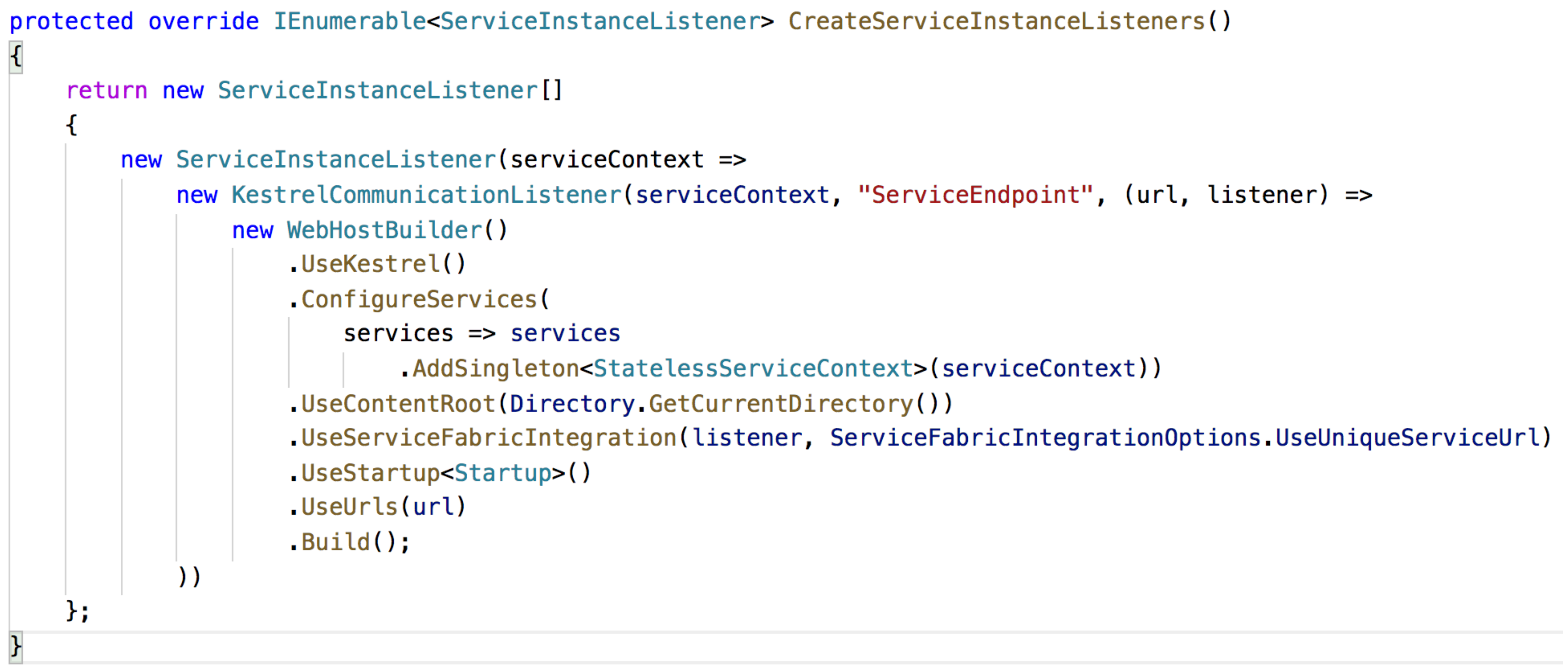


Client
Resolve Service Address
HTTP GET http://10.0.0.1:30001/PartitionID/ReplicaID
http://10.0.0.1:30001
Naming Service
http://10.0.0.1:30001/PartitionID/ReplicaID
Register Address
Service Fabric Integration Middleware
/PartitionID/ReplicaID
Valid Path?
no
410 Gone
yes
Process Request
HTTP Response

WebListener


Node (Windows)
http://10.0.0.1:80/serviceA
http://10.0.0.1:81/serviceB
http://10.0.0.1:80/serviceC
http://10.0.0.1:81/serviceD
http.sys
ServiceHost.exe
WebListener
WebListener
ServiceHost.exe
WebListener
WebListener

Kestrel



```
protected override IEnumerable<ServiceInstanceListener> CreateServiceInstanceListeners()
{
    return new ServiceInstanceListener[]
    {
        new ServiceInstanceListener(serviceContext =>
            new KestrelCommunicationListener(serviceContext, "ServiceEndpoint", (url, listener) =>
                new WebHostBuilder()
                    .UseKestrel()
                    .ConfigureServices(
                        services => services
                            .AddSingleton<StatelessServiceContext>(serviceContext))
                    .UseContentRoot(Directory.GetCurrentDirectory())
                    .UseServiceFabricIntegration(listener, ServiceFabricIntegrationOptions.UseUniqueServiceUrl)
                    .UseStartup<Startup>()
                    .UseUrls(url)
                    .Build();
            ))
    };
}
```

Stateful services & storing state

Storage

Collections

- Single Threaded

Concurrent Collections

- Multi-Threaded

Reliable Collections

- Multi-Node
  - Replicated (HA)
  - Persistence Option
  - Asynchronous
  - Transactional

Reliable Dictionary

```
using (ITransaction tx = StateManager.CreateTransaction()) {
        user.LastLogin = DateTime.UtcNow;
        await m_dic.AddAsync(tx, name, user);
        await tx.CommitAsync();
}
```

Reliable Queue

```
using (var txn = this.StateManager.CreateTransaction())
{
        await this.Queue.EnqueueAsync(txn, 10, cancellationToken);
        await this.Queue.EnqueueAsync(txn, 20, cancellationToken);

        await txn.CommitAsync();
}
```

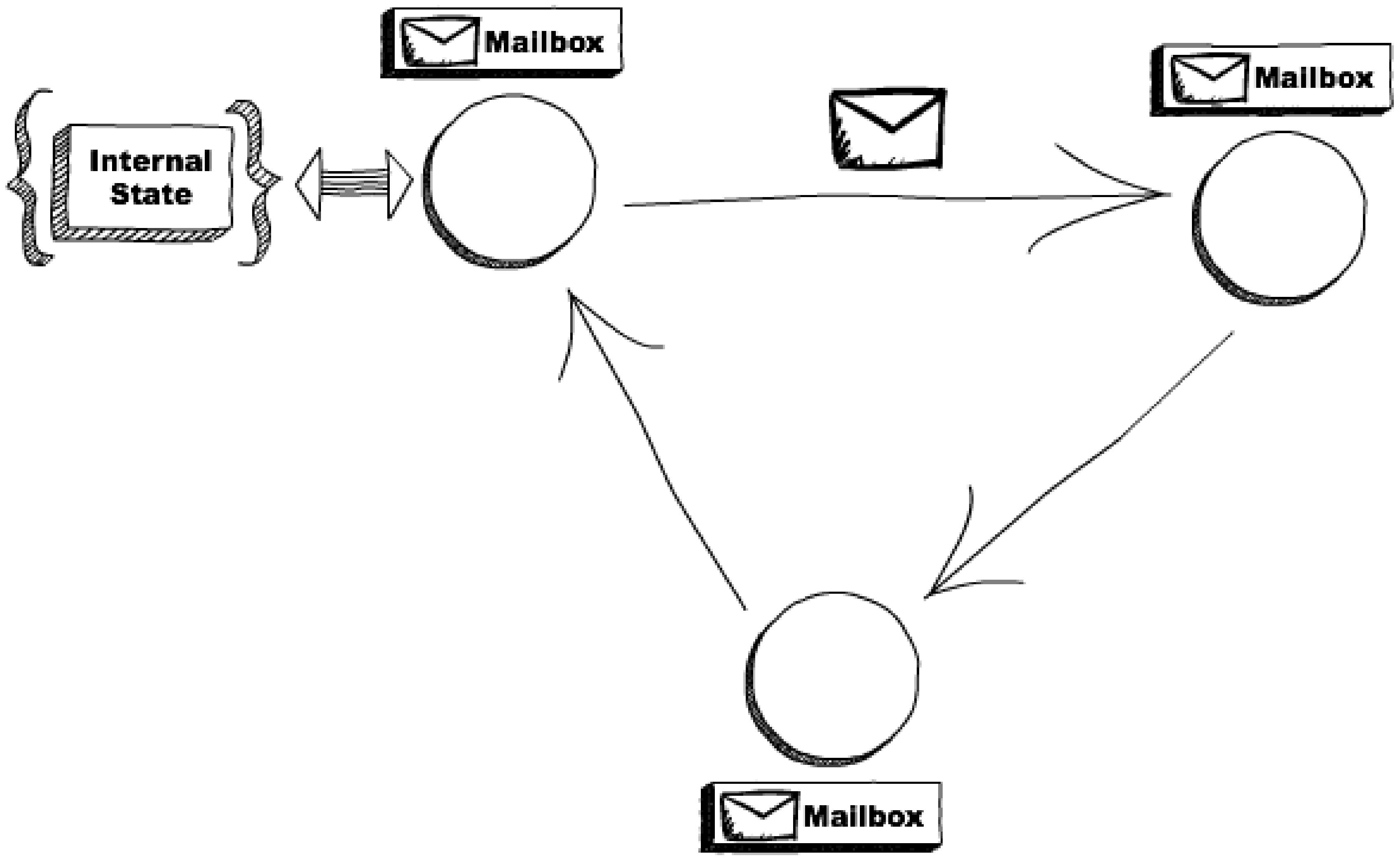
The Actor Model

Azure Service Fabric (Reliable) Actors

Actor Model

- First defined in 1973 by Hewitt, Bishop & Steiger

- Conceptual model for concurrent computation
  - Actors are computational entities with private state
  - No need for locks
  - No shared callstack due to message passing

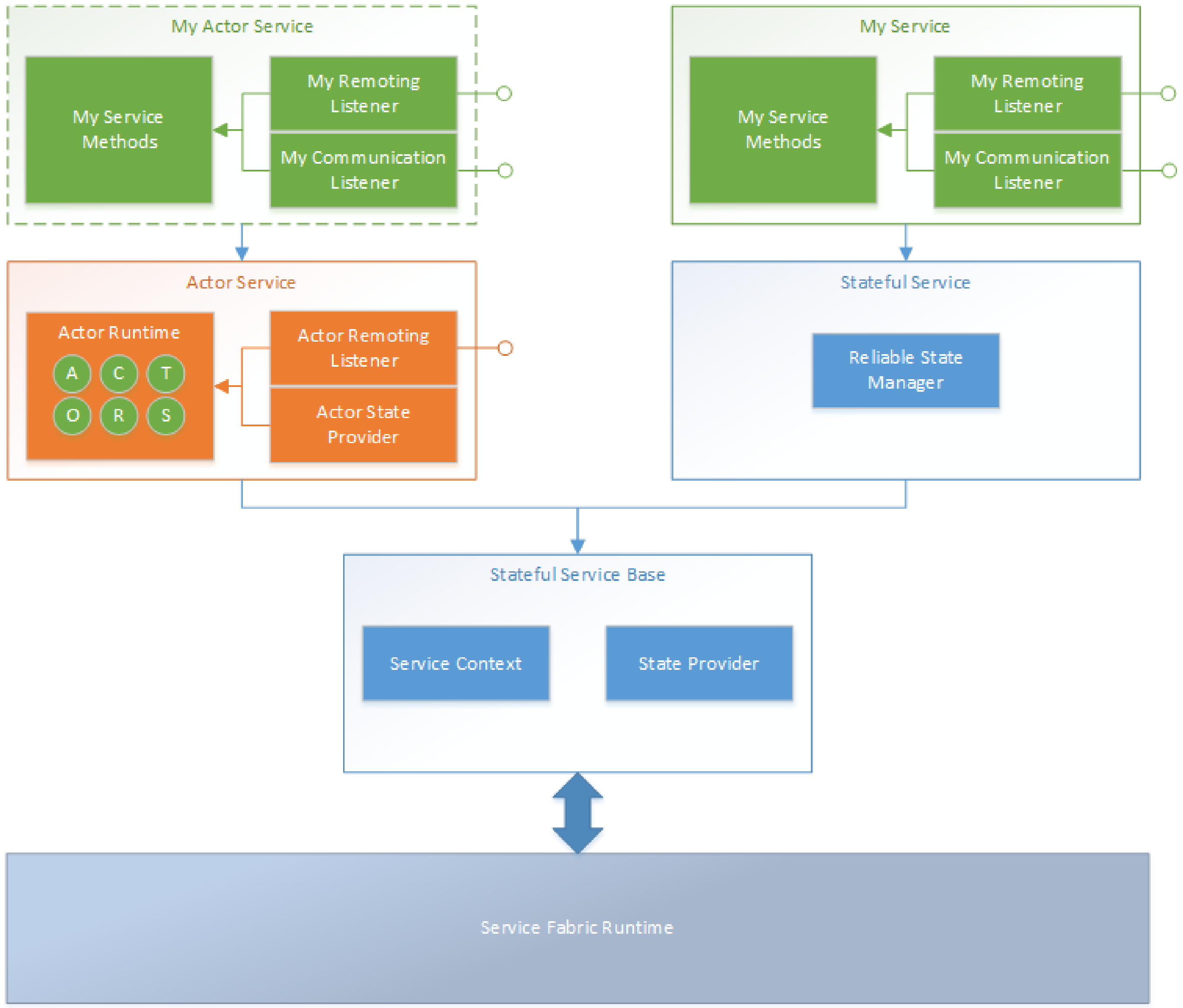

Mailbox
Internal State
Mailbox
Mailbox

What actors do

- Create more actors;
- Send messages to other actors;
- Designates what to do with the next message.

Actors in Azure Service Fabric

- Hosted inside an ActorService
- The ActorService is a stateful service which hosts:
  - Actor Runtime
  - Actor StateProvider


My Actor Service
My Service Methods
My Remoting Listener
My Communication Listener
My Service
My Service Methods
My Remoting Listener
My Communication Listener
Actor Service
Actor Runtime
A
C
T
O
R
S
Actor Remoting Listener
Actor State Provider
Stateful Service
Reliable State Manager
Stateful Service Base
Service Context
State Provider
Service Fabric Runtime

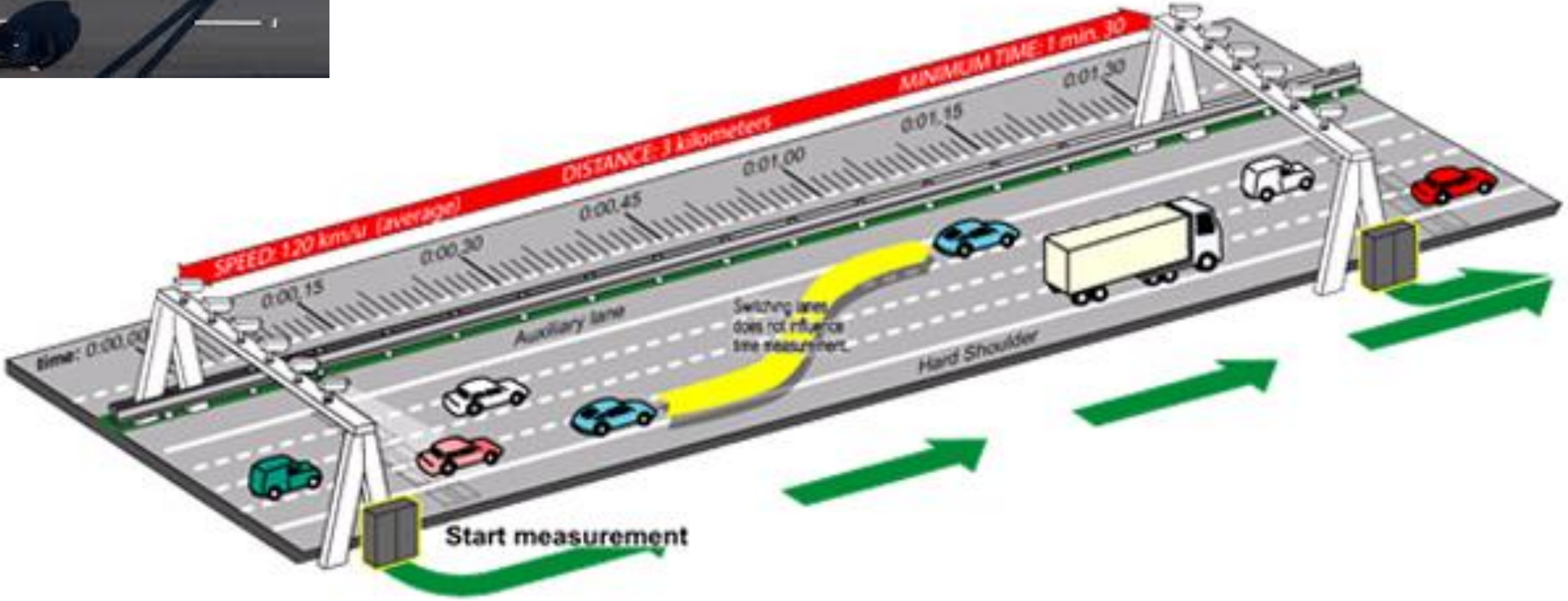
Reliable Actors

```
// Actor definition (State+Behaviour) to run in the back end.
// Object instances of this Actor class will be running transparently
// in the service back end.
public class VehicleActor : Actor<Vehicle>, IVehicleActor
{
  public void UpdateGpsPosition(GpsCoordinates coord)
  {
    // Update coordinates and trigger any data processing
    // through the State property on the base class Actor<TState>.
    this.State.Position= coord;
  }
}
```

Reliable Actors

```
// Client .NET code
ActorId actorId = ActorId.NewId();
string applicationName = "fabric:/IoTVehiclesActorApp";
IVehicleActor vehicleActorProxy =
  ActorProxy.Create<IVehicleActor>(actorId, applicationName);
vehicleActorProxy.UpdateGpsPosition(new GpsCoordinates(40.748440, -73.984559));
```

DevSum18


SPEED: 120 km/u (average)
DISTANCE: 3 kilometers
MINIMUM TIME: 1 min. 30
time: 0:00.00
0:00.15
0:00.30
0:00.45
0:01.00
0:01.15
0:01.30
Auxiliary lane
Switching lanes
does not influence
time measurement.
Hard Shoulder
Start measurement

The road ahead

- Azure Service Fabric Mesh
- Different configuration: yaml / json
- Better Docker Compose & Kubernetes support
- Less invasive
- Reliable storage can also run outside of ASF

And....
Last but not least
– don't forget to evaluate this session in the DevSum app!

GET IT ON Google Play

Download on the App Store

Download from Windows Phone Store