Microsoft

Using Azure Service Fabric to Manage Containers

Ryan McIntyre

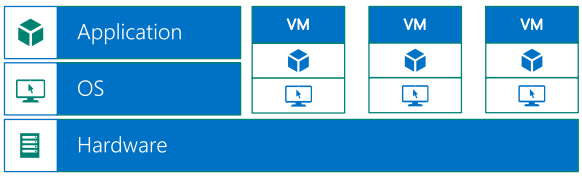
Agenda

Intro to containers
Intro to Service Fabric
Service Fabric container orchestration capabilities
Demos!
Conclusion

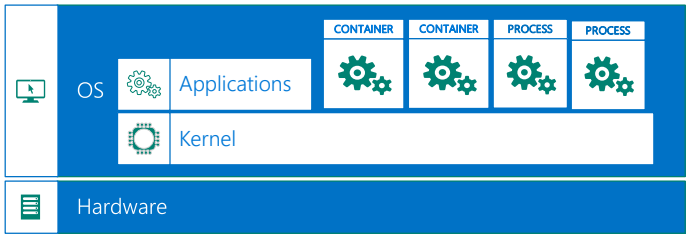
Containers: What, Why, and How?

What is a container?

Traditional virtual machines = hardware virtualization

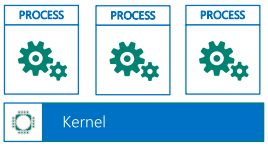


Containers = Operating system virtualization



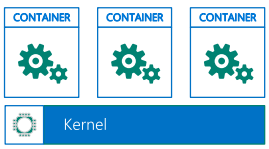
Processes

Maximum speed and density



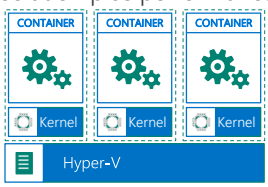
Docker Containers

Speed and density

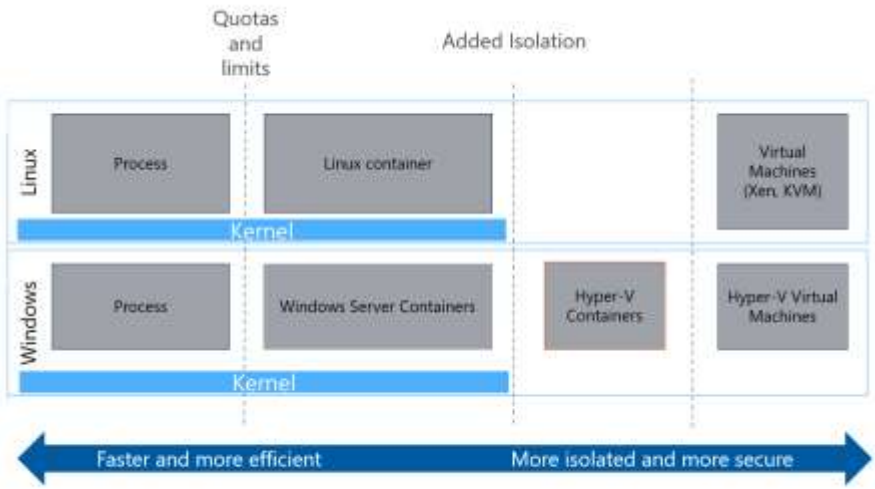


Hyper-V Containers

Isolation plus performance



Virtualization and isolation levels



Why containers?

Mitigate “noisy neighbors”

Reduce “works on my machine” vs. in production

Lift and shift existing applications (ex: IIS)

Lift and shift of existing container images

Portability, agility, avoidance of lock-in

How to orchestrate containers?

Containers need to discover and talk to each other

Containers need to be resilient to hardware and software faults

Containers need to be "placed" appropriately

Containers need to be scaled in/out on demand

Containers need to have rolling upgrades with zero downtime

How to orchestrate containers?

Containers need to discover and talk to each other

Containers need to be resilient to hardware and software faults

Containers need to be "placed" appropriately

Containers need to be scaled in/out on demand

Containers need to have rolling upgrades with zero downtime

Service Fabric provides all this out-of-the-box. It is highly optimized and is data-aware….

Containers are NOT microservices

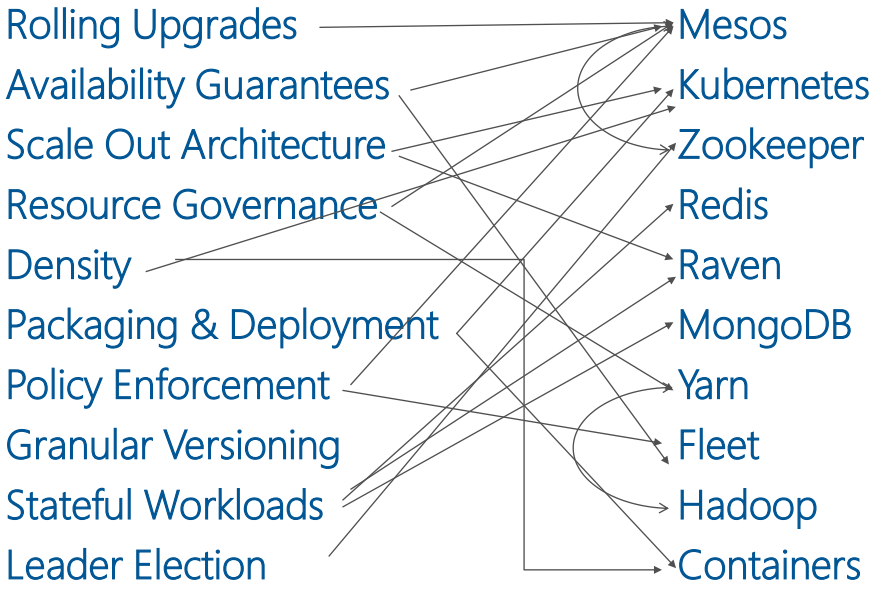
Well, you can still put a large monolithic application inside a container….

Microservices are an application design pattern:

- Small units of responsibility
- Structured interfaces and communication
- Potentially different technology choices
- Generally horizontally scalable

Service Fabric

In the wild… go create a PaaS yourself!



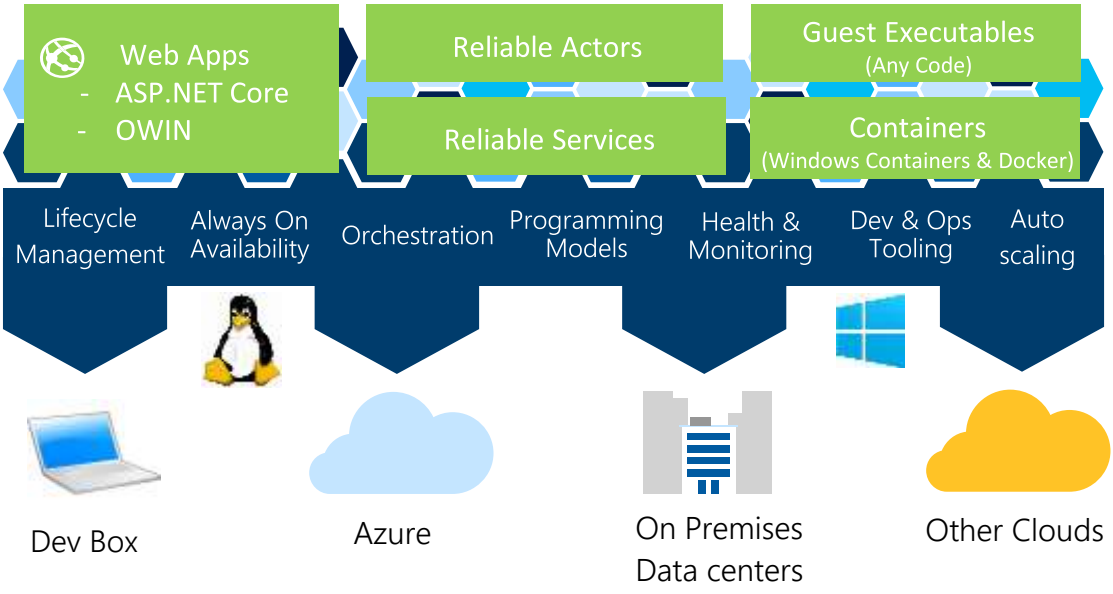
?

An easier, battle-tested approach

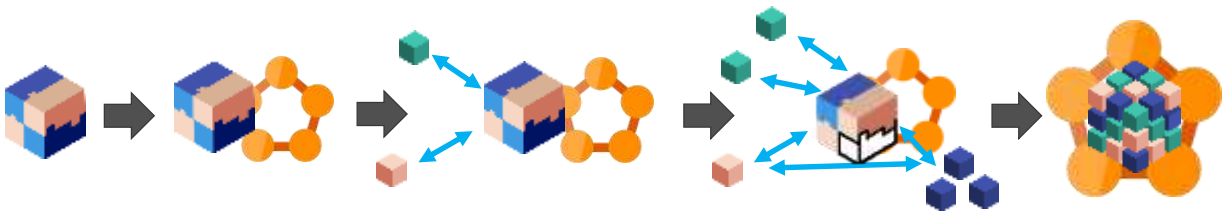
Rolling Upgrades
Availability Guarantees
Scale Out Architecture
Resource Governance
Density
Packaging & Deployment
Policy Enforcement
Granular Versioning
Stateful Workloads
Leader Election

Service Fabric

Azure Service Fabric



Migrating a traditional application to microservices



1) Traditional app

...You can stop at any stage

Service Fabric – Some Numbers

- Scale: Thousands of Nodes
- Density: Tens of Thousands of Workloads per Node
- Reliability: Detect and Respond to Failure in Seconds
- Manageability: Deploy and Upgrade in Minutes, automatically roll back on errors

Orchestration capabilities become critical when you want scale and reliability: There is simply too much moving for people to address every contingency.

Powered by Service Fabric…

SQL Database
2.0 million DBs

Document DB
Billions transactions/day

IoT Hub
10 of Ks devices & millions of messages

Event Hubs
640bn events/day

Skype

Cortana

Intune

Dynamics

Power BI

Service Fabric Orchestration

Why we need orchestration:

Rules

- Place workloads based on specific rules
- Update service requirements
- Place workloads based on resource consumption and node capacities

Optimizations

- Dynamically adjust resource consumption
- Balance and rebalance on the fly
  - Add/Remove workloads
  - Add/Remove nodes
  - Go over capacity

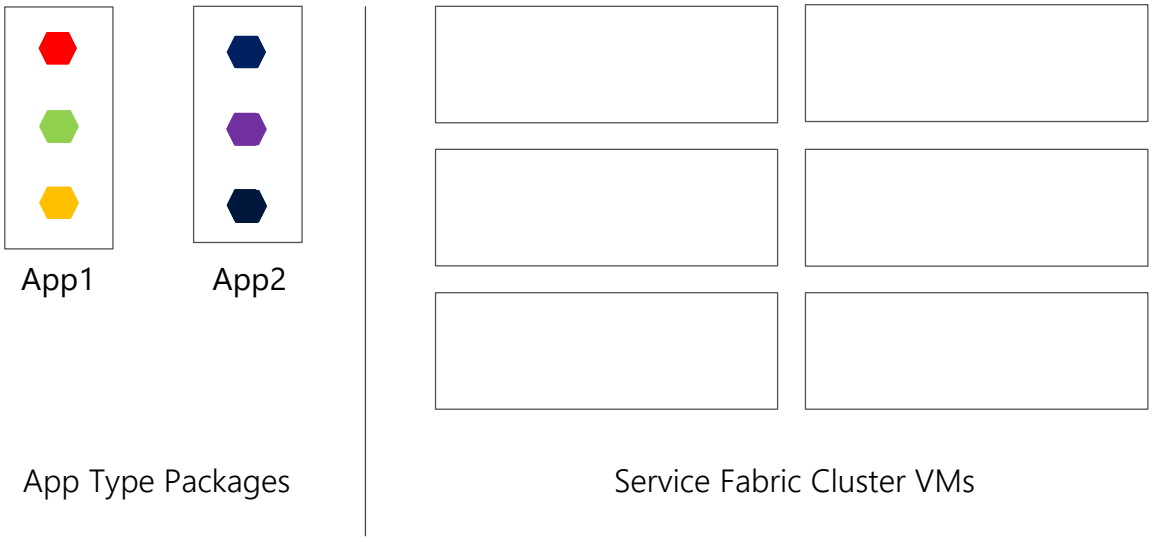
Processes

- Automated Monitored Rolling Upgrades (w/ Rollback)
  - While respecting rules & optimizations

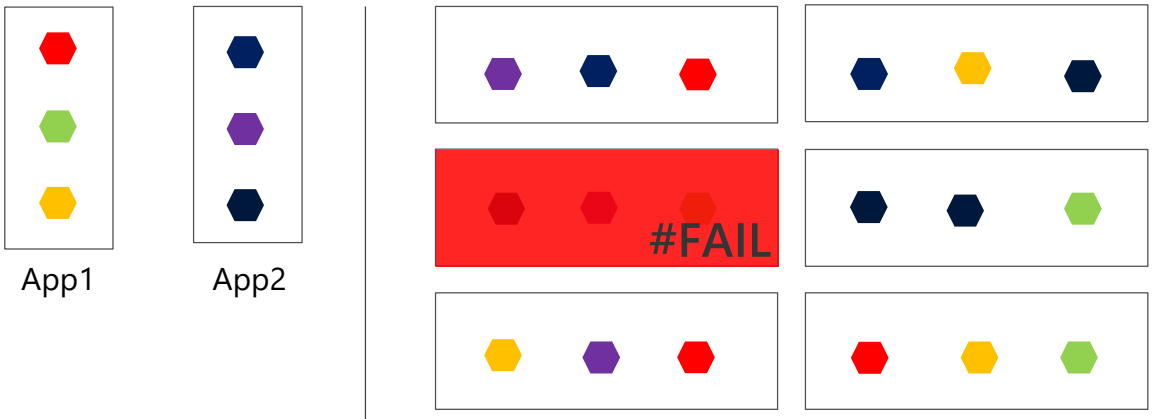
Service Fabric

- Service Fabric was there before orchestration was cool
- Platform agnostic – Windows/Linux, On-Prem/Cloud
- Offers much more…
  - Rich data-aware orchestration capabilities
  - Integrated operational health and safety checks
  - Rolling upgrades and health monitoring
  - A PaaS platform with APIs to build natively distributed apps
  - Service Fabric microservices can run inside/outside/SxS with containers
  - Integrated IDEs
  - Devbox debugging (on Windows/Linux/Mac)
  - About 1/3rd ***of all cores in Azure*** run on Service Fabric

Service Fabric - Deployment



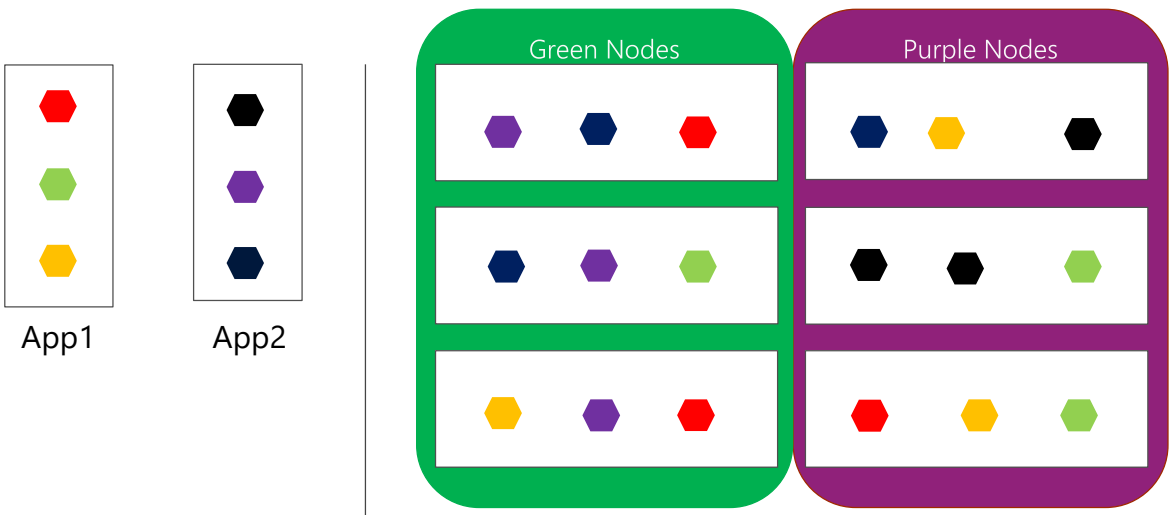
Handling machine failures



App Type Packages

Service Fabric Cluster VMs

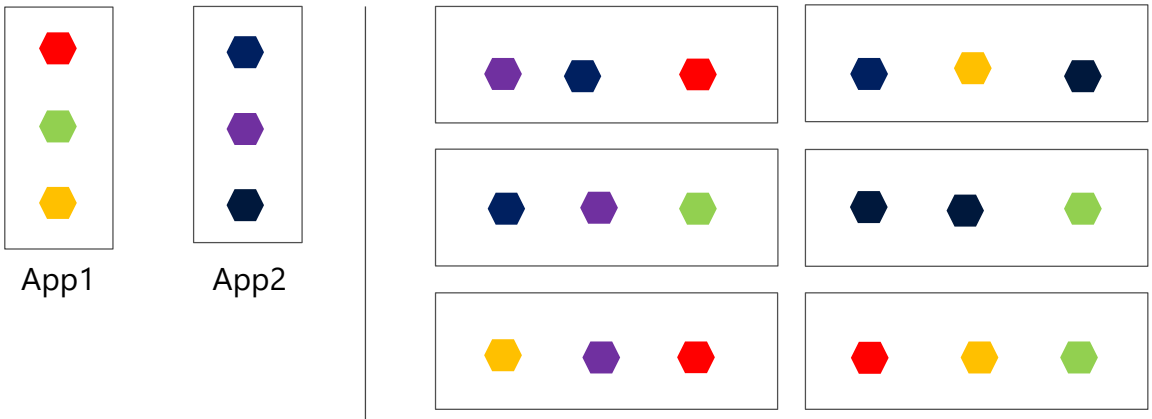
Orchestration basics - Constraints



App Type Packages

Service Fabric Cluster VMs

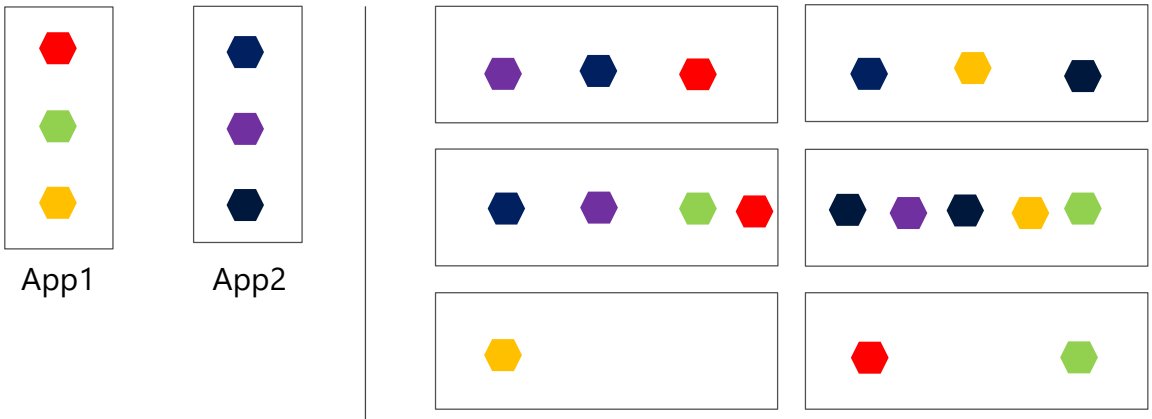
Orchestration basics - Capacity



App Type Packages

Service Fabric Cluster VMs

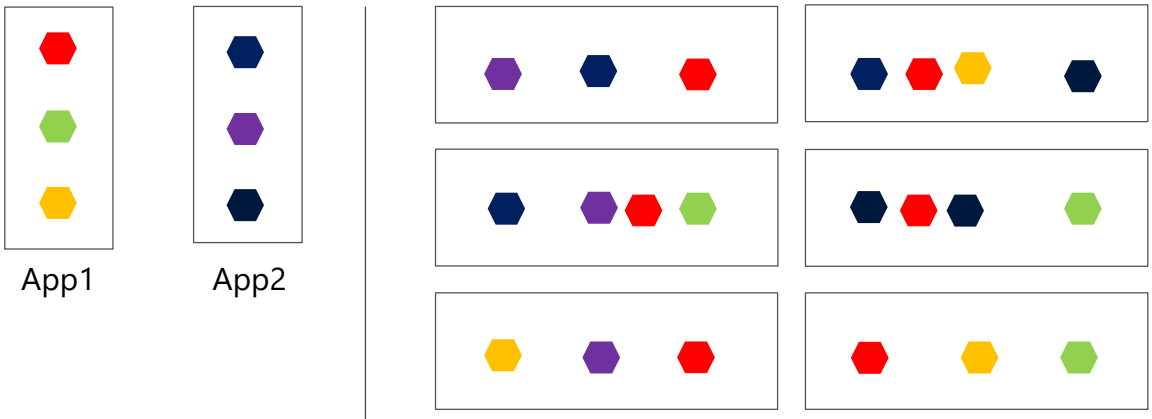
Orchestration basics - Balancing



App Type Packages

Service Fabric Cluster VMs

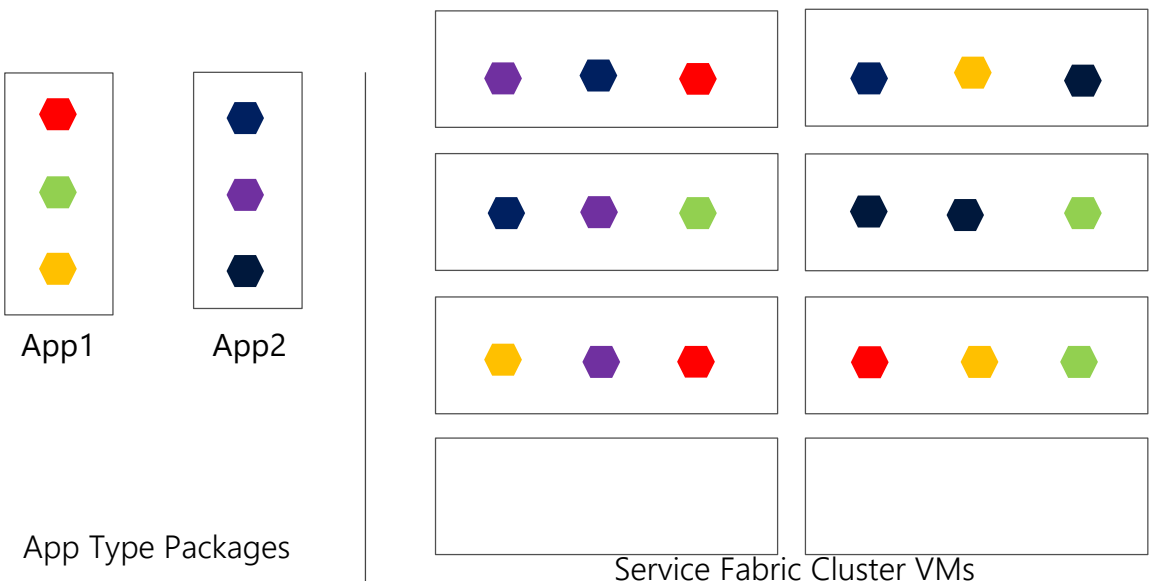
Orchestration basics – Scaleout services



App Type Packages

Service Fabric Cluster VMs

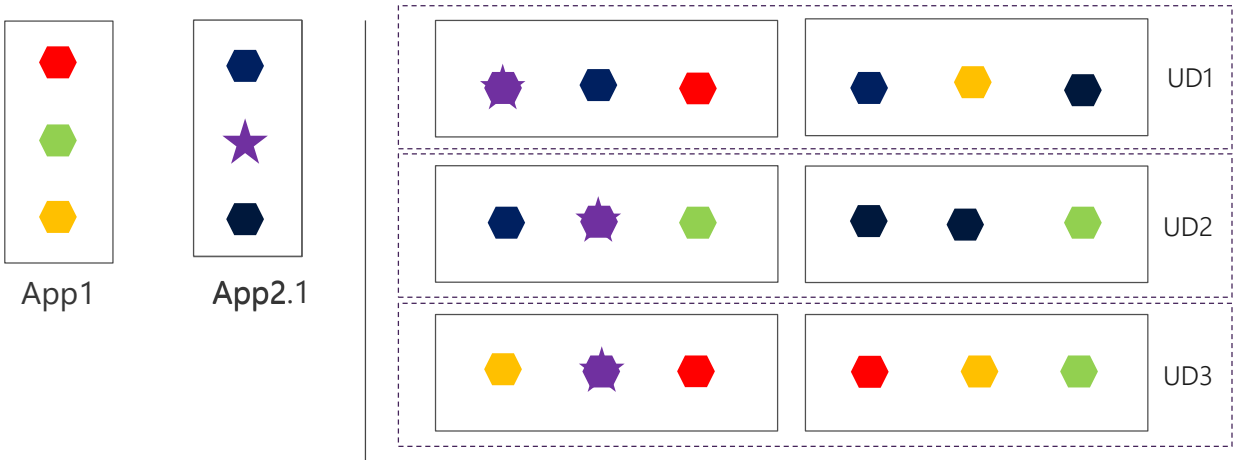
Orchestration basics – Scaleout cluster



App Type Packages

Service Fabric Cluster VMs

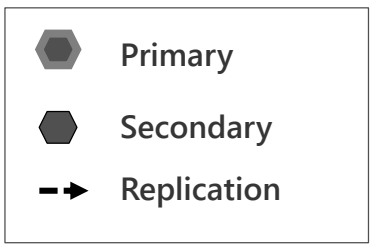
Orchestration basics - Upgrade

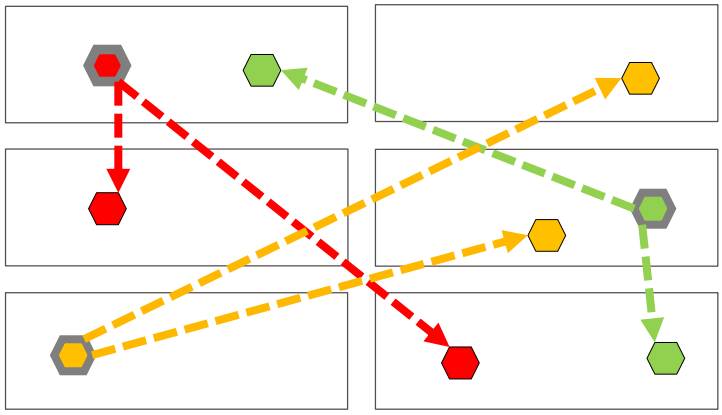


App Type Packages

Service Fabric Cluster VMs

Data-aware: Stateful microservices





Service Fabric Cluster VMs

Demos, Demos.....

Deployment
Failover
Capacity
Constraints
Scaleout
Zero downtime upgrade

Conclusion

Service Fabric as an orchestrator: Roadmap

- Support for Windows Containers
  - In preview today
  - GA ~May 17
- Support for Linux Containers
  - In preview today
  - GA ~Sep 17
- Features coming
  - Better networking - DNS support, public IPs, vNETs amongst containers…
  - Volume drivers, diagnostics, much more…

Session objectives and takeaways

Learn about containers & orchestrating them with Service Fabric on Windows and Linux

Service Fabric is Microsoft's best-in-class container orchestrator

Service Fabric is a battle-tested hyper-scale data-aware orchestration platform