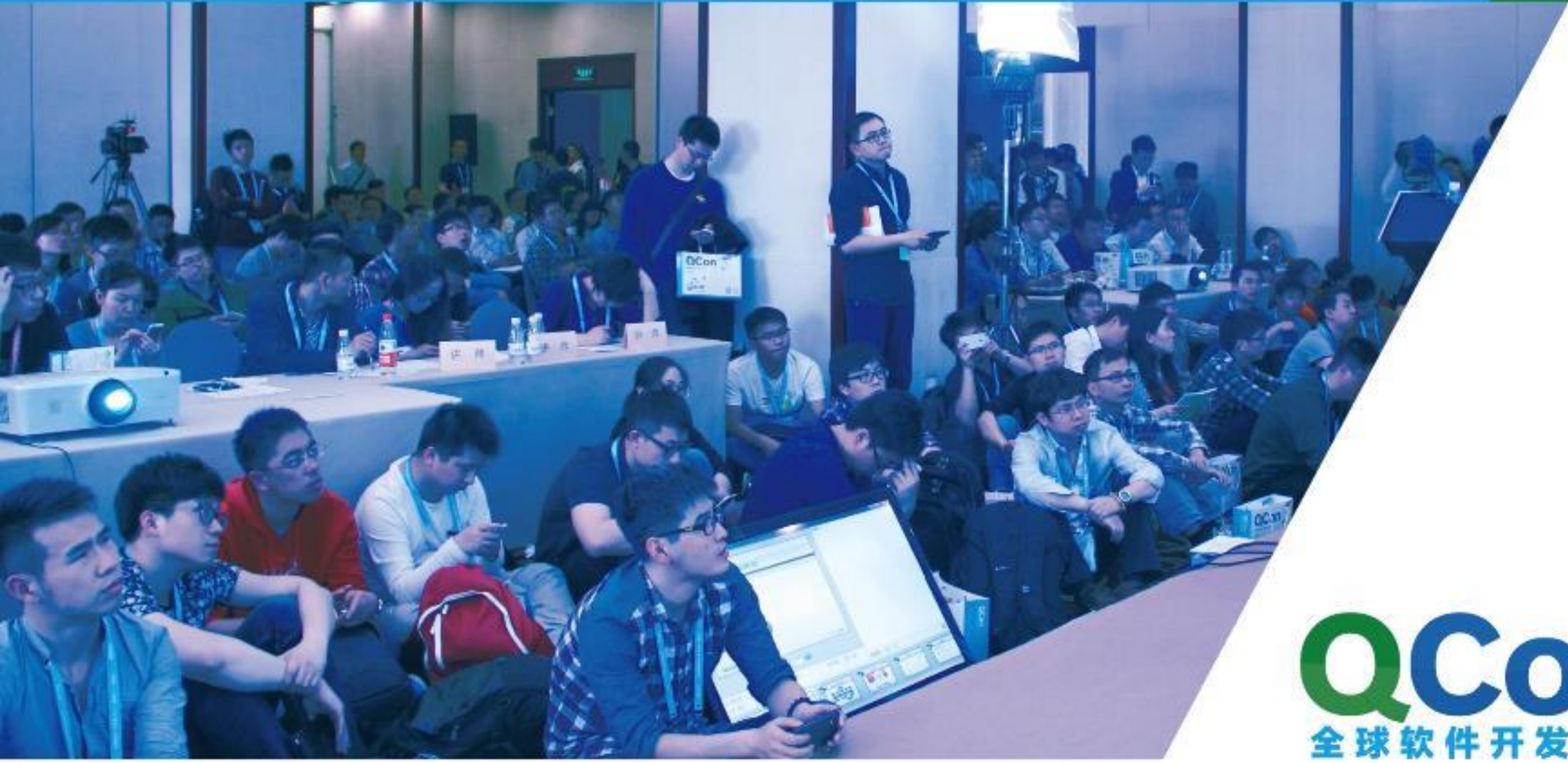
QCon全球软件开发大会

International Software Development Conference



QCon
全球软件开发大会

Azure PaaS v2 与微服务架构应用开发

王启霄
微软中国 开发体验与平台合作事业部

Objective

- Microservice introduction and key principles
- Service Fabric introduction and positioning within Microsoft development platform
- Learn how to Build Service Fabric services (stateless, stateful, actor-services)
- Learn on deployment of Service Fabric services locally

Takeaway:

- Microservices is key for high-scalable and complex/large applications
- Service Fabric is especially made for microservices approaches

What is the “Microservices” approach?

“Is this just a new hype or is it the next big thing in distributed computing?”

“Isn’t it just another SOA?”

“How small is a microservice?”

“I see **Domain-Driven Design principles** here, right?”

“Are microservices the right approach for any application?”

“Should I use the same technology, language and approach for every microservice?”

Traditional architecture approach

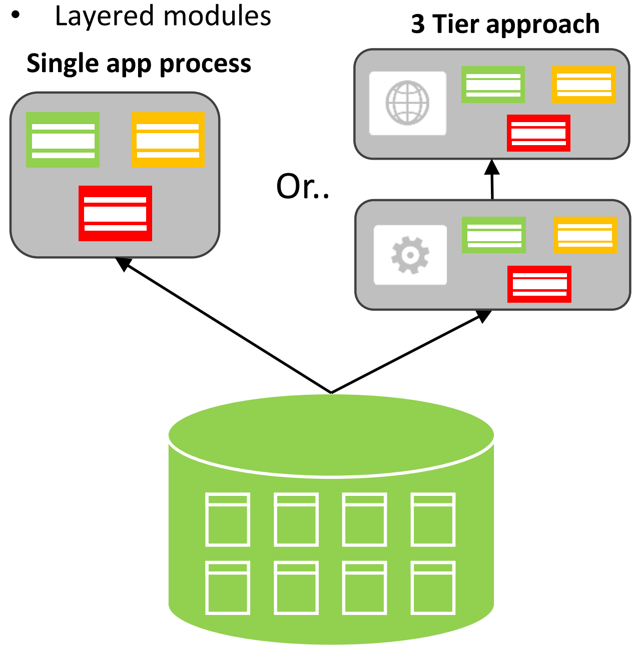
Microservices architecture approach

Microsoft

Data/State in Applications

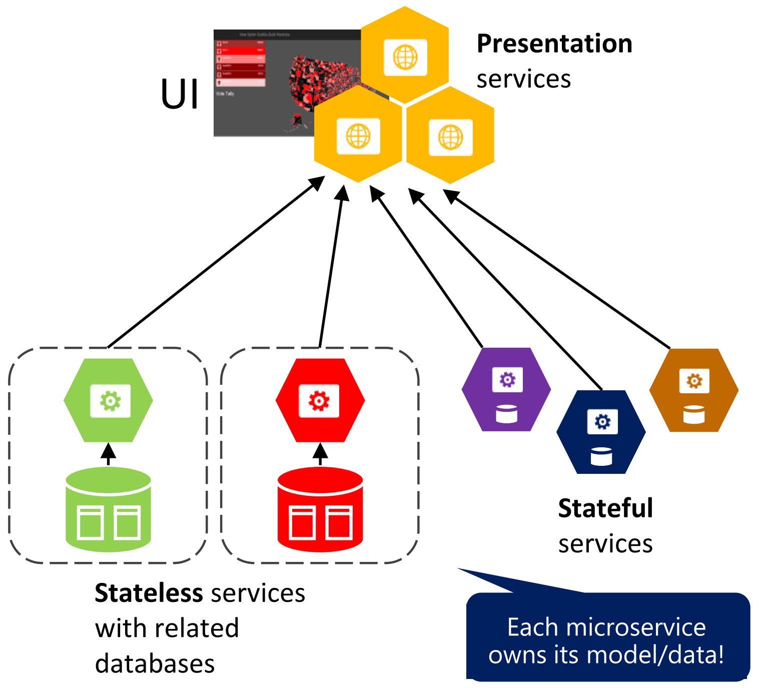
Traditional Application

- Single app process or 3 Tier approach
- Several modules
- Layered modules



Single monolithic database

Microservices approach



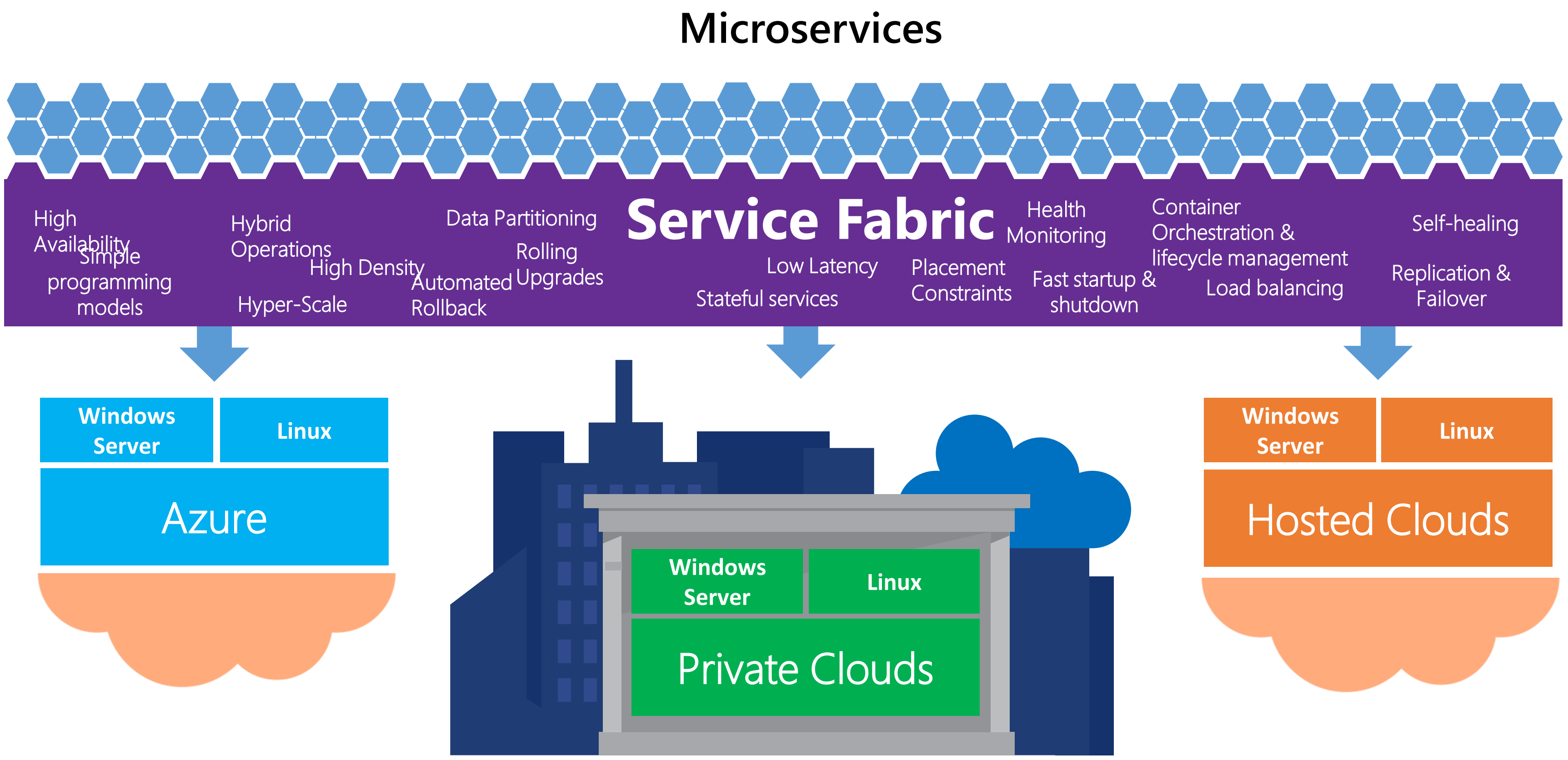
Stateless services with related databases

Microservices and databases

Azure Service Fabric

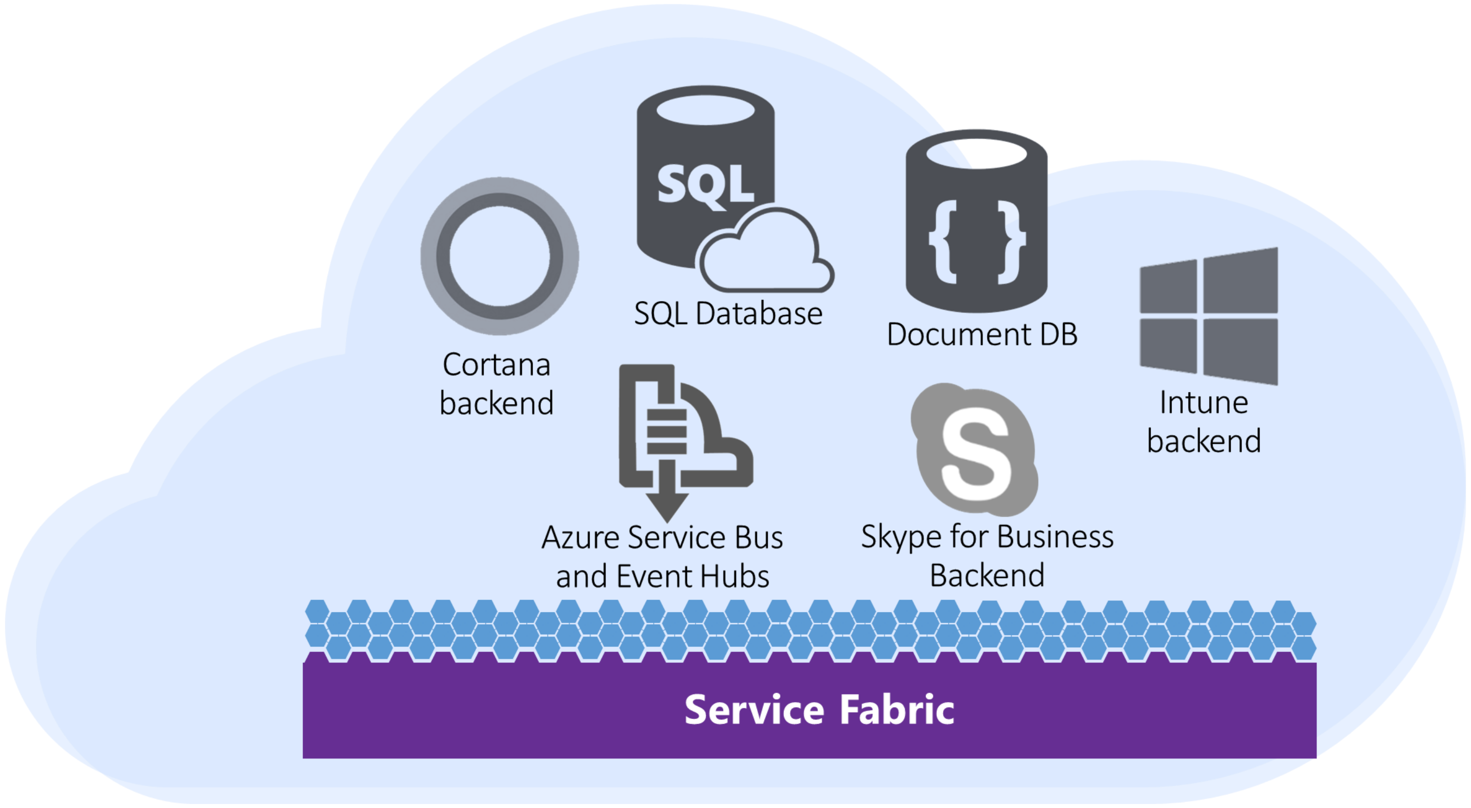
Microsoft

Microsoft **Azure Service Fabric**

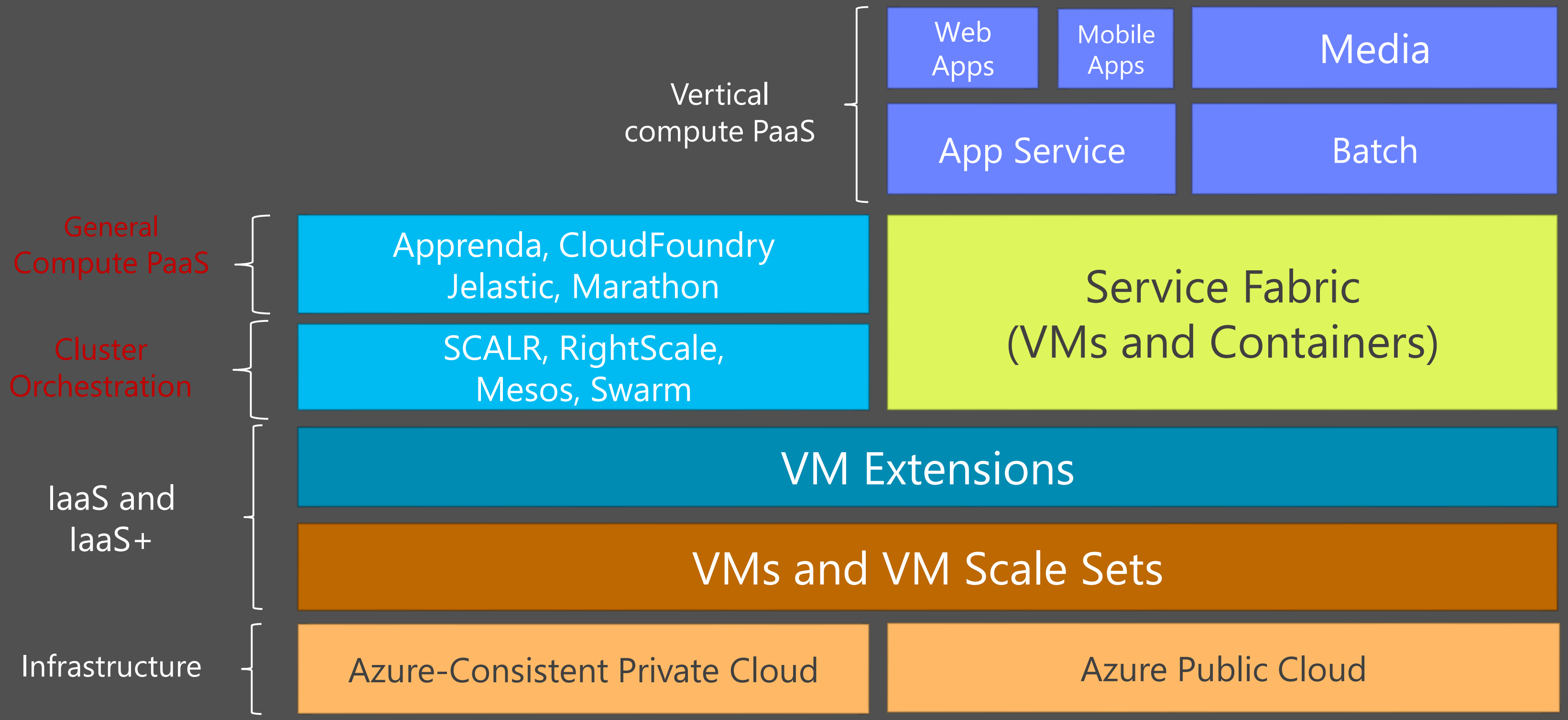


Proven platform for hyperscale

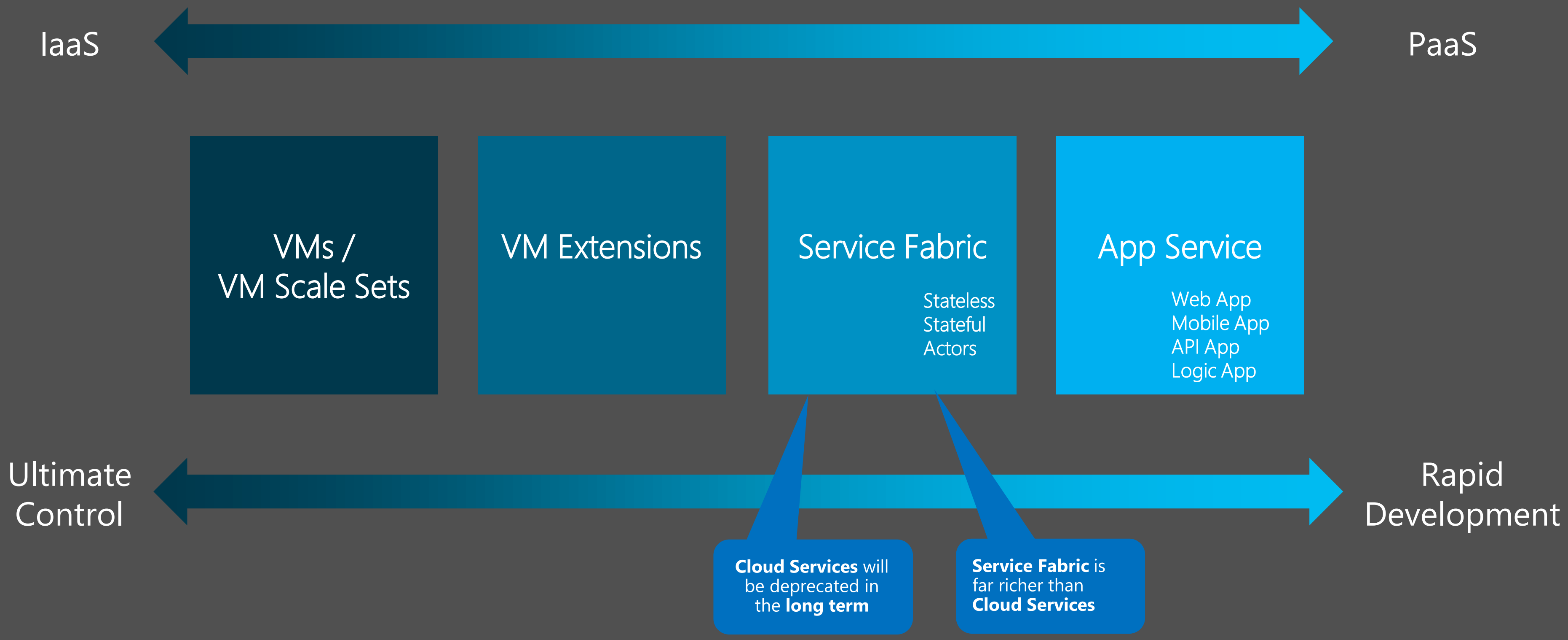
- Tested and internally used by Microsoft for quite a few years (aka. internally “Windows Fabric”)
- Service Fabric is the foundational platform for many high scalable services at Microsoft



Service Fabric position in Azure



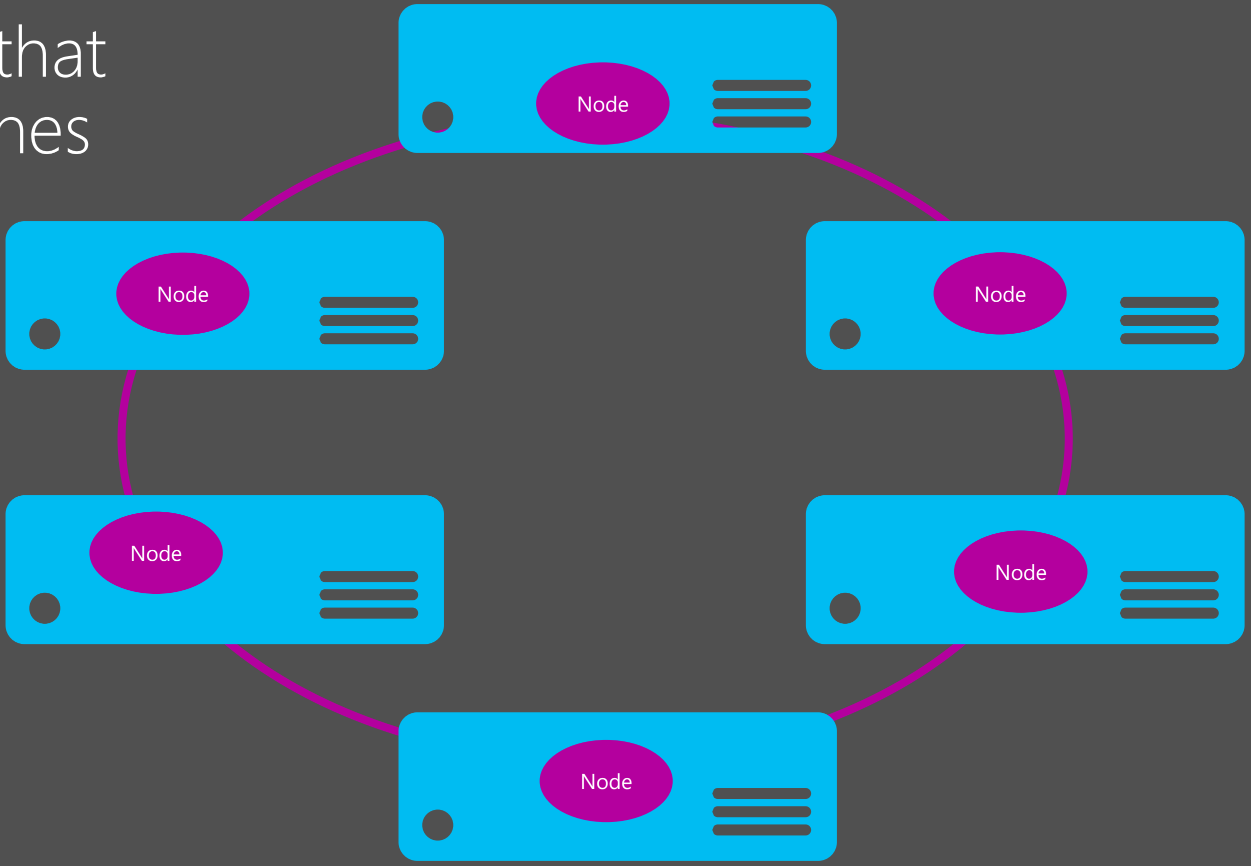
Azure Compute Continuum



Service Fabric Cluster: A federation of machines

A set of machines that Service Fabric stitches together to form a cluster

Clusters can scale to 1000s of machines



Cluster: System view

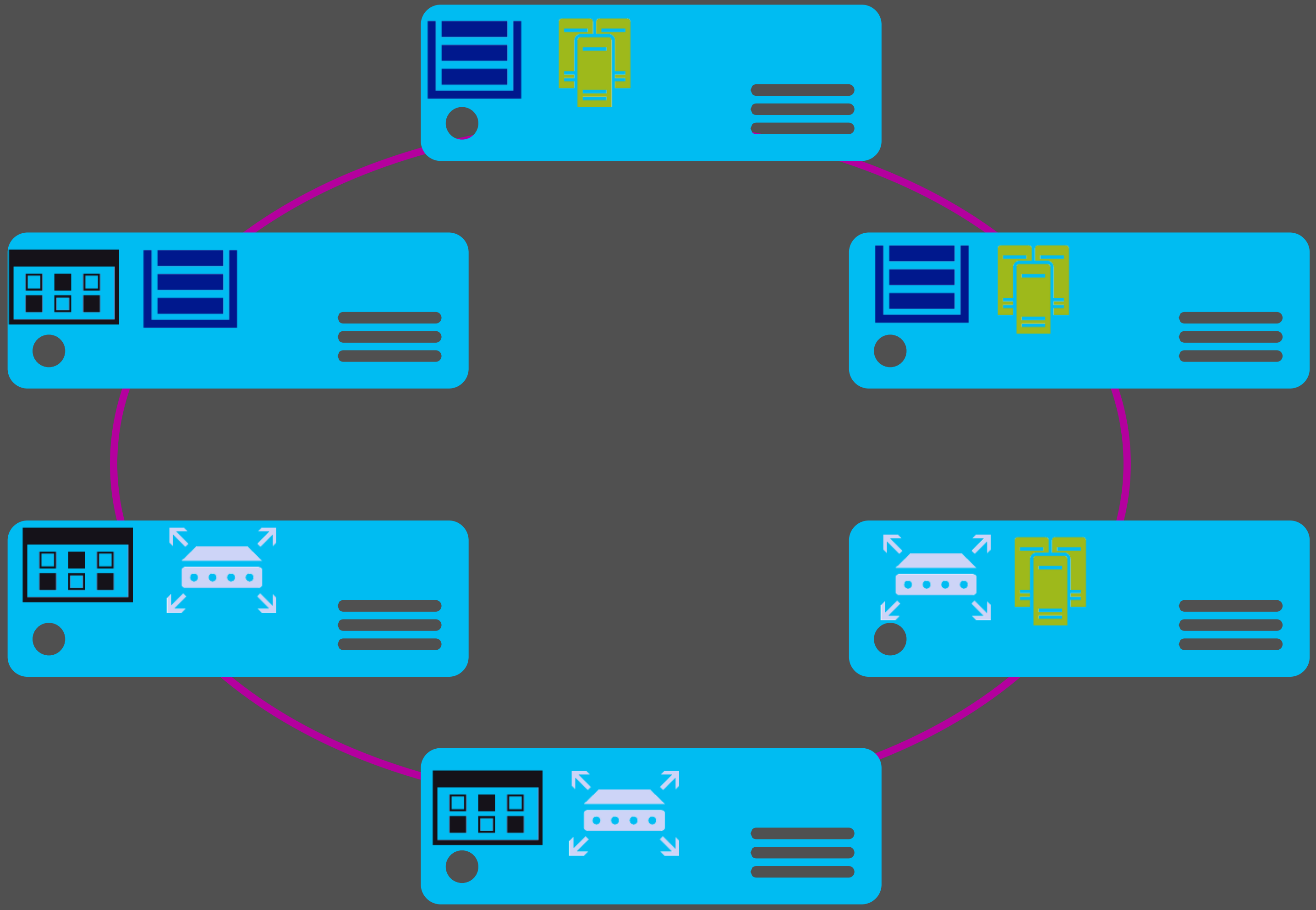
System Services

Failover manager

Cluster manager

Naming

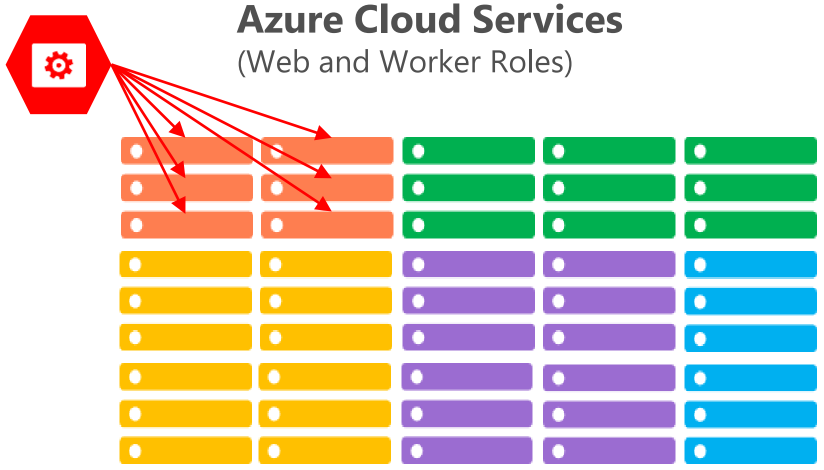
Image store



Microsoft

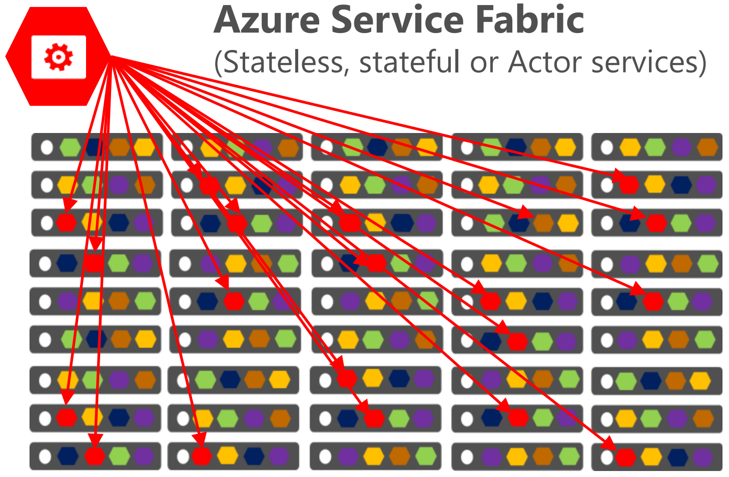
Comparing Azure Cloud Services vs. Azure Service Fabric

Azure Cloud Services
(Web and Worker Roles)



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

Azure Service Fabric
(Stateless, stateful or Actor services)

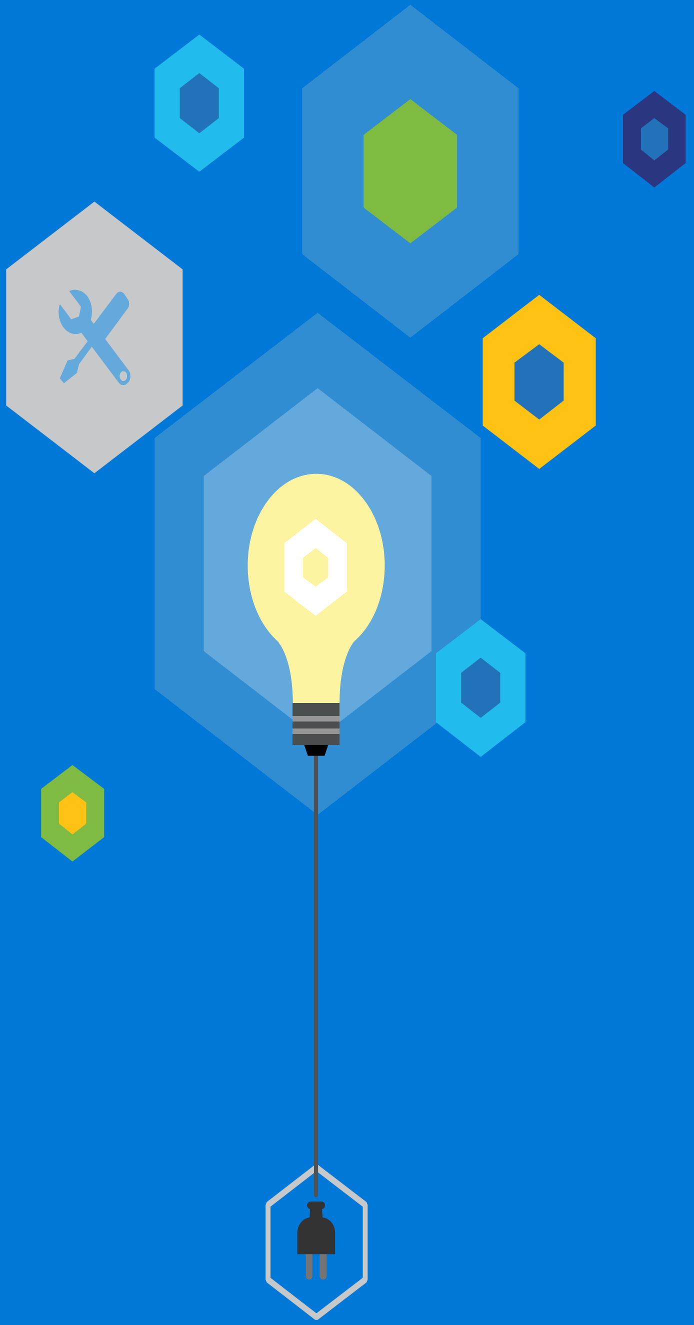


- Many microservices per VM
- High microservices density
- Fast deployment & upgrades
- Fast scaling microservices across the cluster

Demo

Service Fabric cluster
and microservices density

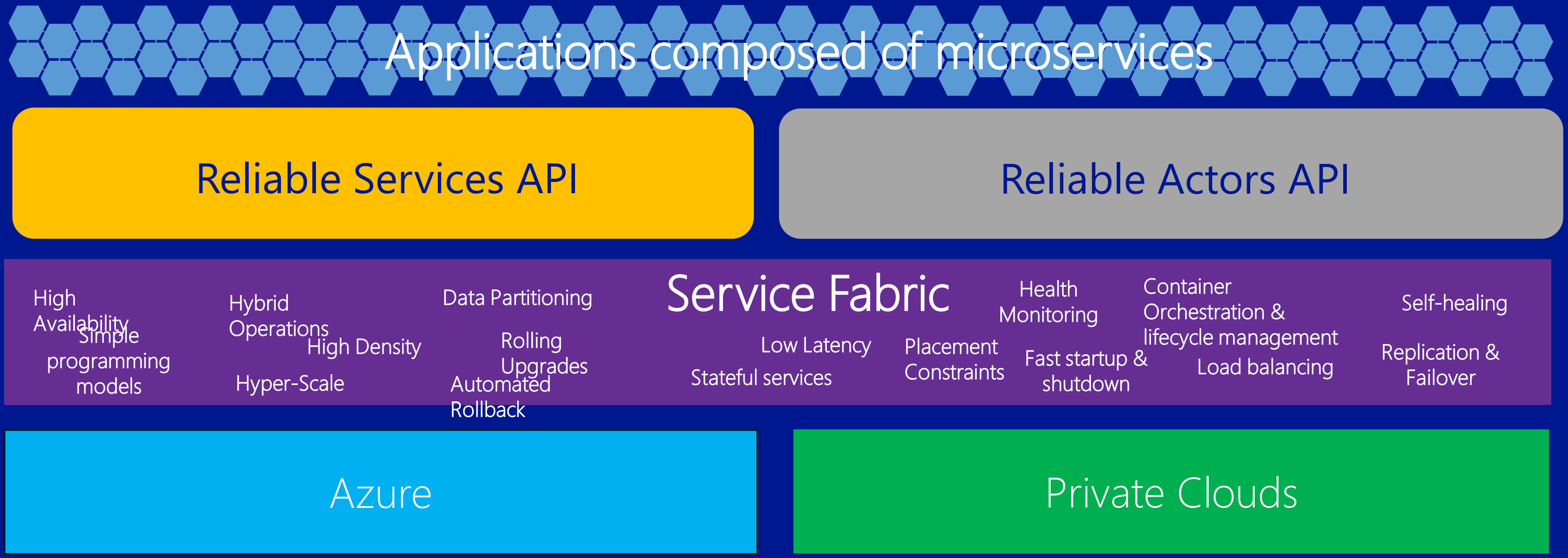
- Service Fabric Explorer
- Cluster viewer



Service Fabric Microservices

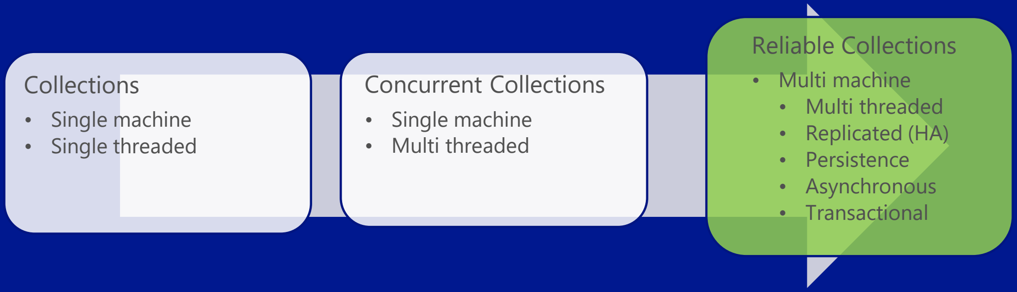
- A microservice is whatever you want it to be:
  - ASP.NET
  - node.js, Java VMs
  - Arbitrary .exe
- Stateless microservices
  - A microservice that has state where the state is persisted to external storage, such as Azure databases or Azure storage
    - e.g. Existing web (ASP.NET) and worker role applications
- Stateful microservices
  - Reliability of state through replication and local persistence
  - Reduces the complexity and number of components in traditional three-tier architecture

Service Fabric Programming Models



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



Collections
- Single machine
- Single threaded

Concurrent Collections
- Single machine
- Multi threaded

Reliable Collections
- Multi machine
  - Multi threaded
  - Replicated (HA)
  - Persistence
  - Asynchronous
  - Transactional

ReliableDictionary<T>

ReliableQueue<T>

DEMO

Reliable Service API

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)

DEMO - Reliable Actor API

Orchestration in Service Fabric:

- Rules
  - Place workloads based on specific rules
  - Update service requirements
  - Place workloads with static consumption and 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

THANKS