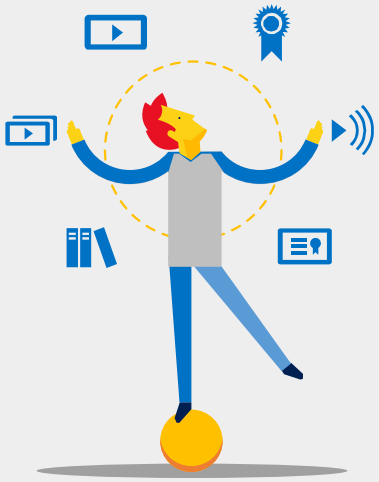
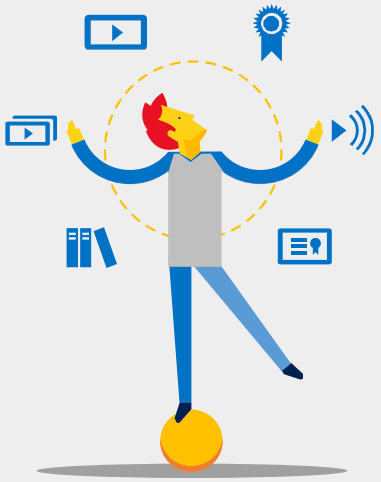
Service Fabric Patterns & Best Practices



Microsoft

Service Fabric Jumpstart

Mark Fussell
Haishi Bai

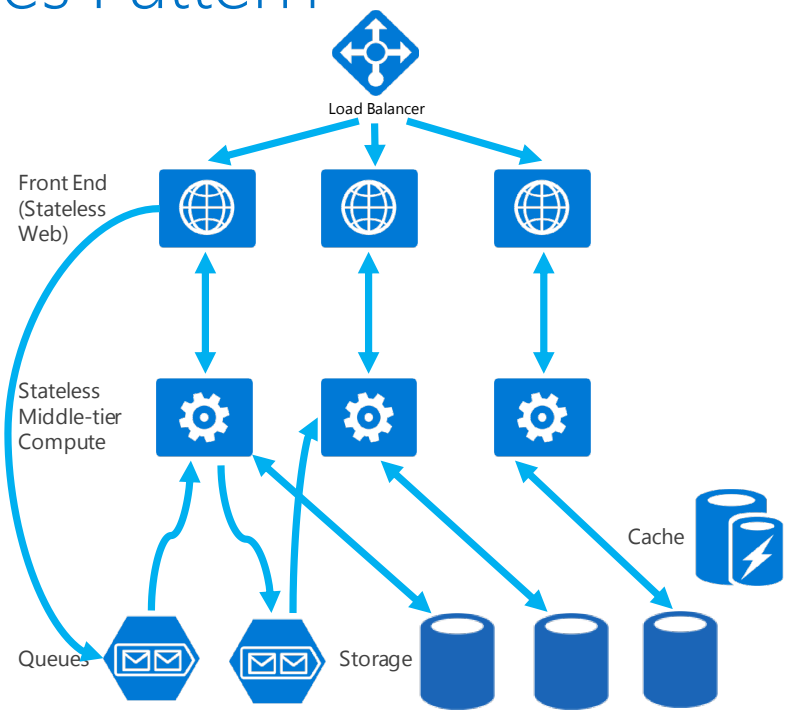


Microsoft

A lap around Service Fabric

Stateless Services Pattern

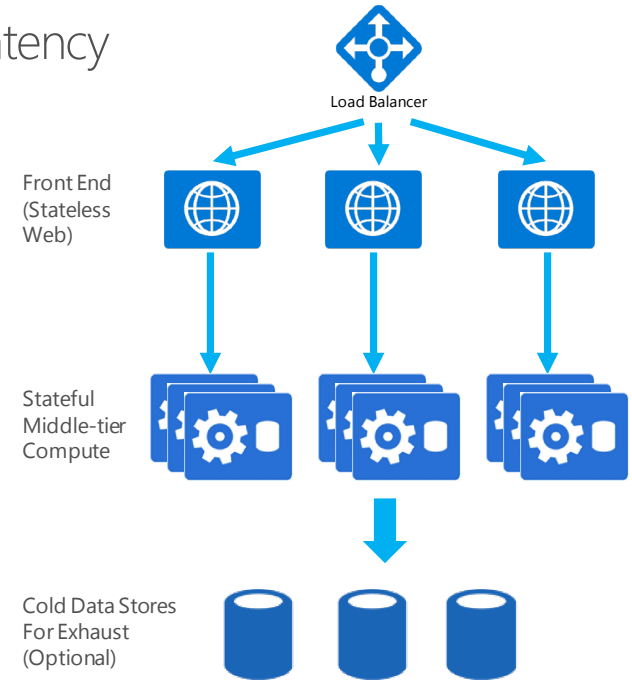
- Scale stateless services backed by partitioned storage
- Increase reliability and ordering with queues
- Reduce read latency with caches
- Manage your own transactions for state consistency
- More moving parts each managed differently



Stateful Services Pattern

Simplify design, reduce latency

- Application state resides in the compute tier
- Low latency reads and writes
- Partitions are first class at the service layer for scale-out
- Built in transactions
- Fewer moving parts
- External stores for exhaust and offline analytics



Problem

How do I design/dev/host an application on cloud?

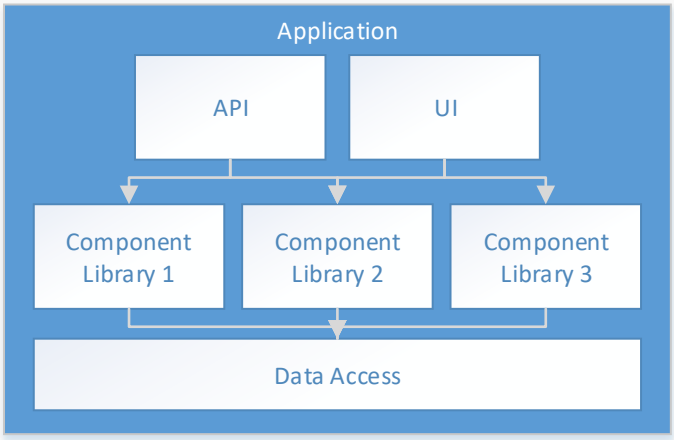
- You don't own the hardware
- Failure is norm
- Dynamic scale
- Security
- Managing cost
- Service operation
- Productivity

Q&A: Microservices and Service Fabric

- What is driving the change in application development? Why are we starting to talk about microservices?
- Why choose microservices and how would you define them?
- Can you talk about moving from a monolith to a microservice design? What have you seen and does this mean a rewrite of the application?
- We hear about contracts between microservices. What does this mean and how should I design for this?
- With these distributed microservices, what are other problems and challenges that developers have to think about up front?

Application design

Traditional application

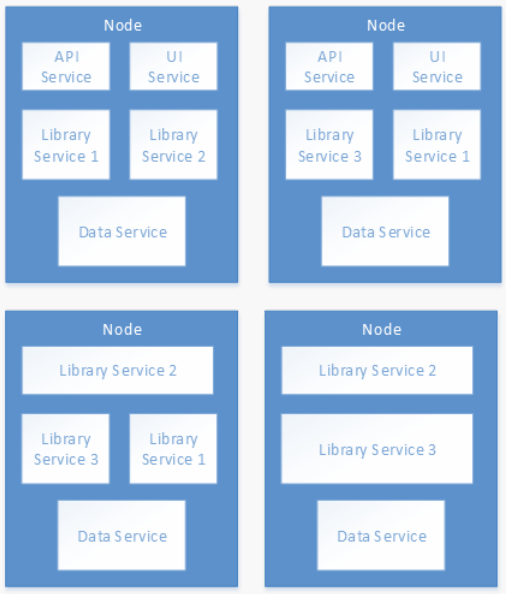


- Compile-time contract validation
- Local operations
- Easier to reason about

- Expensive to scale application
- Hard to scale data access
- Upgrades are long and costly

Application design

Application composed of microservices



- Runtime contract validation
- Network operations
- Harder to reason about

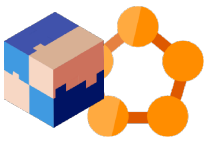
- Cheaper to scale application
- Easier to scale data access
- Upgrade continuously

Migrating a traditional application

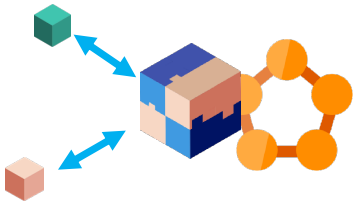
- Decide on the problems you are solving
  - Scale, agility, resilience
- Decided on a well define area to re-architect
- You can have mixture of traditional and microservice designs

Stages of migrating a traditional application

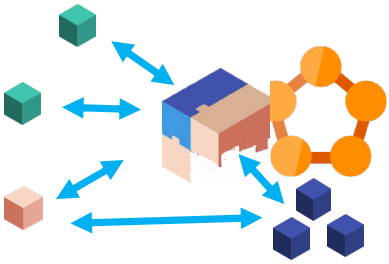
2) Traditional app hosted as guest executable or container in Service Fabric



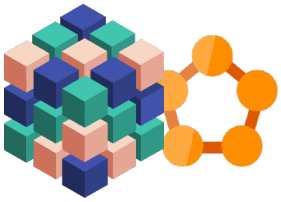
3) Traditional app with new microservices



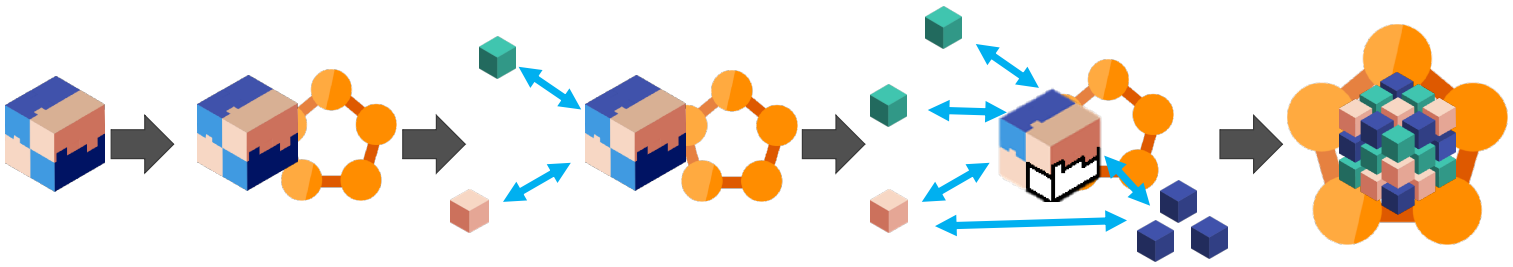
4) Breaking the Traditional app into microservice



5) Transformed into microservices



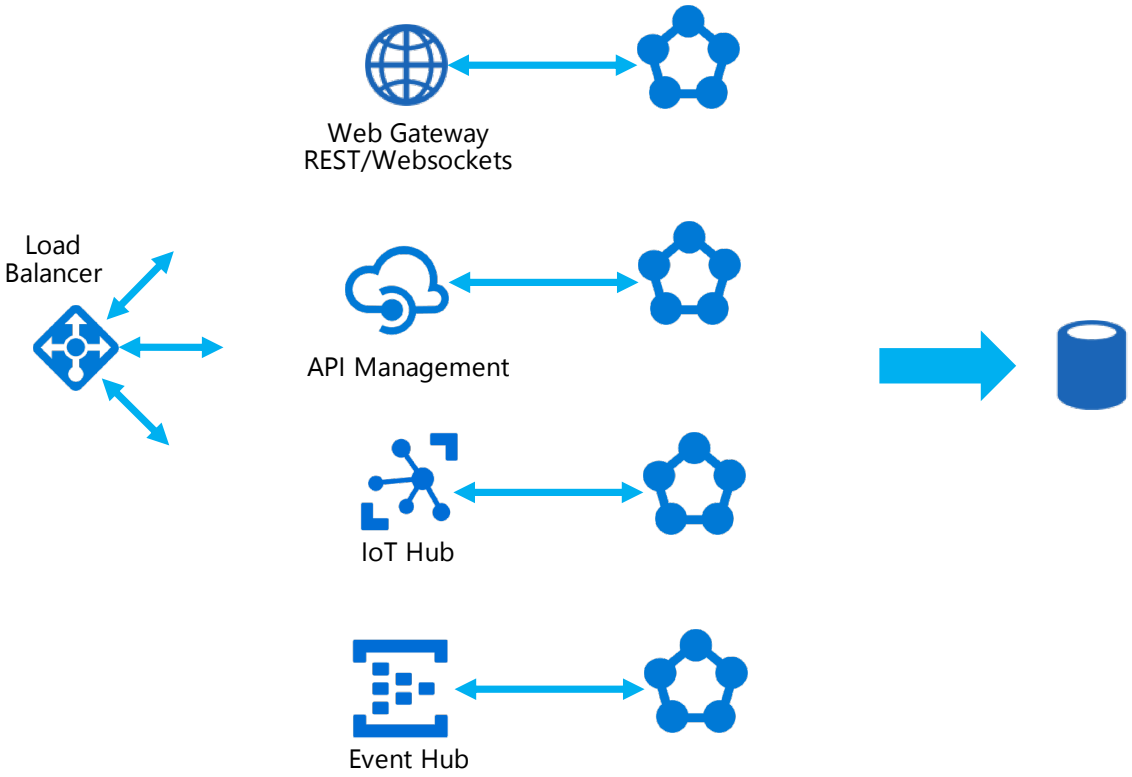
Migrating a traditional application



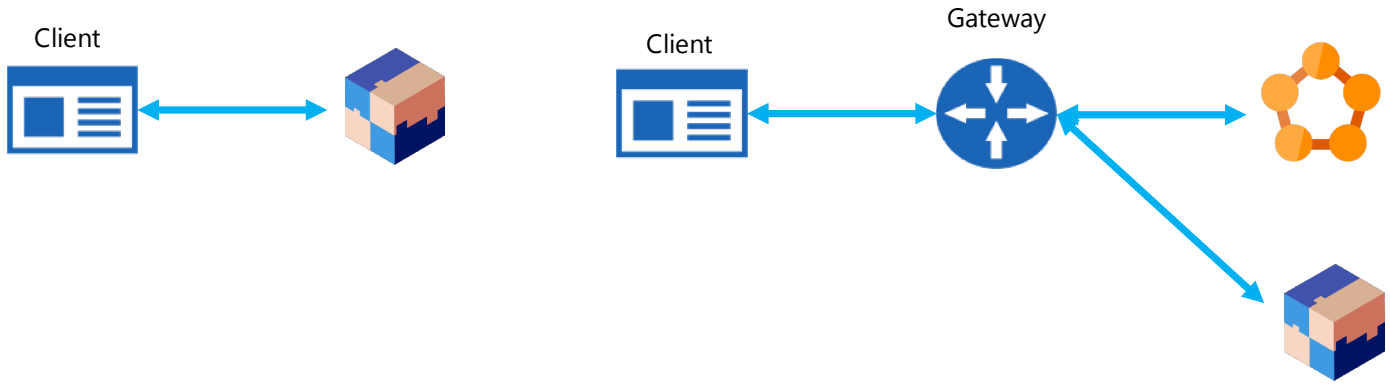
1)Traditional app

...You can stop at any stage

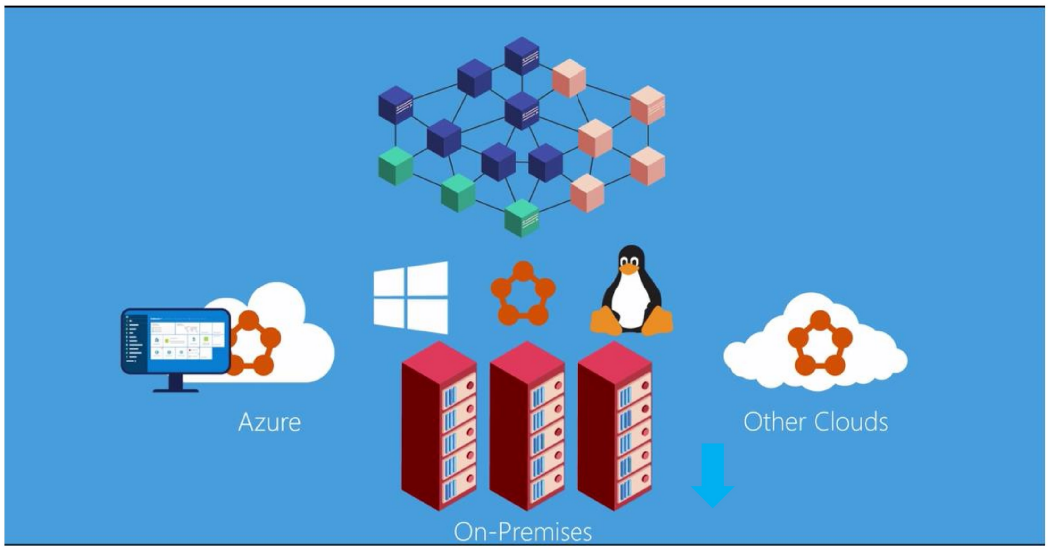
Common design pattern using gateways



Using a gateway to integrate a traditional app with Service Fabric



How do I explain Service Fabric ?



http://aka.ms/servicefabricvideo