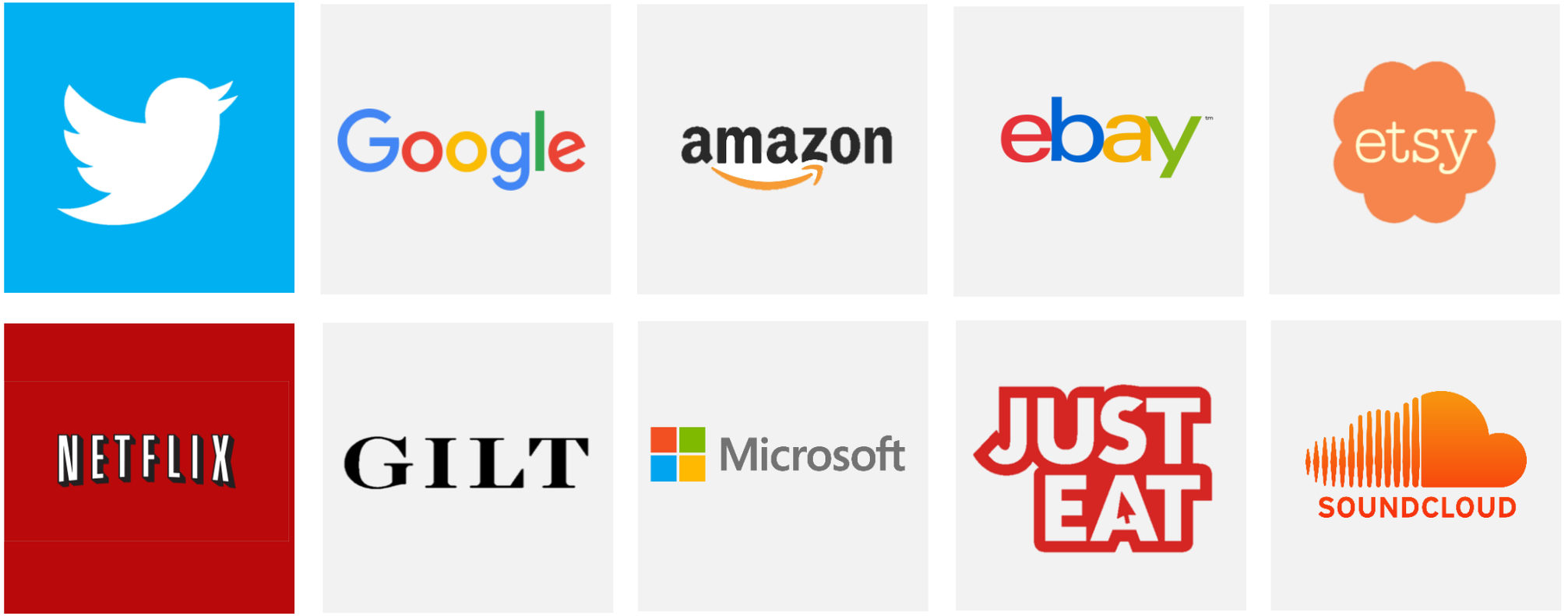
Microservices

Just another buzzword?

Joni Collinge
Technical Evangelist – Microsoft
@dotjson



"Microservices are not a silver bullet… but, benefits > costs."

– Matt Heath, Technical Lead @ Hailo

Why are they using them?

What does “microservice” mean?

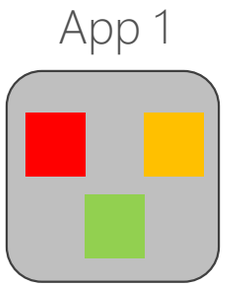
- Distributed architectural design style
- A microservice is an isolated component of a microservice system
- Encapsulation and modularization at the service level
- How small actually is micro?
- It is a refinement of SOA, DDD, and Component-based SE
- Complimented by cloud computing, DevOps and new workloads

Monolithic application approach

: Presentation

: Business Logic

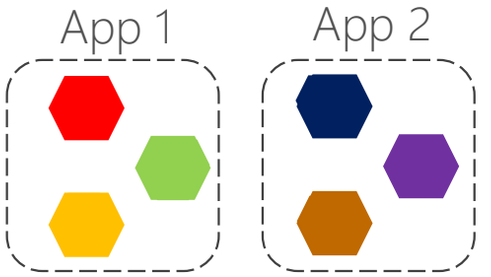
: Data Access



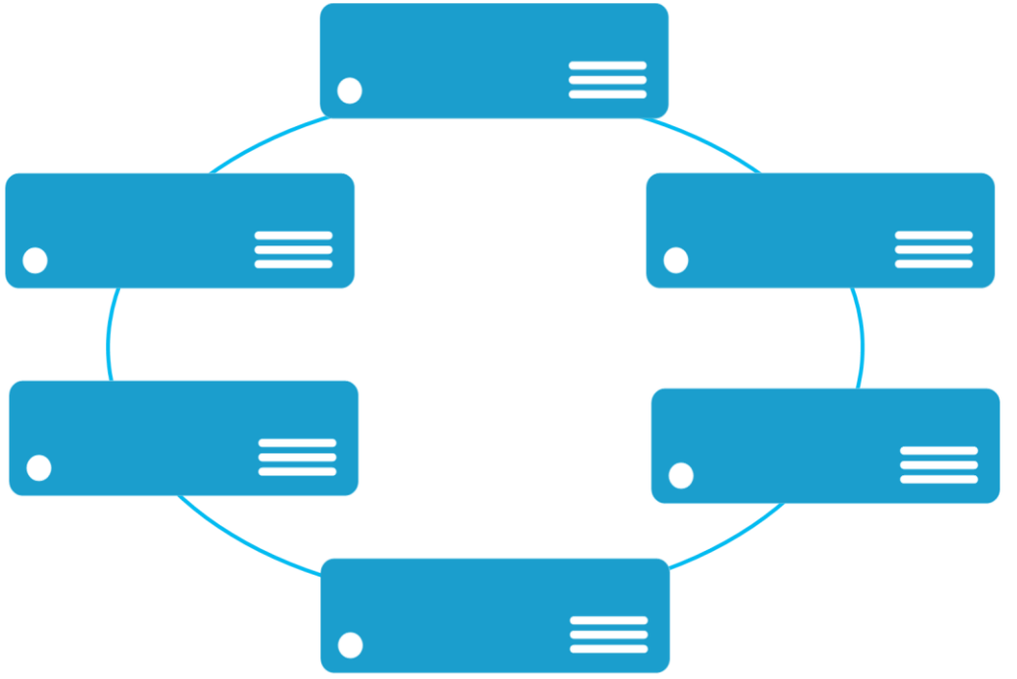
- Scales by cloning the app on multiple servers/VMs/containers

Microservices application approach

- A microservice application separates functionality into separate smaller services.

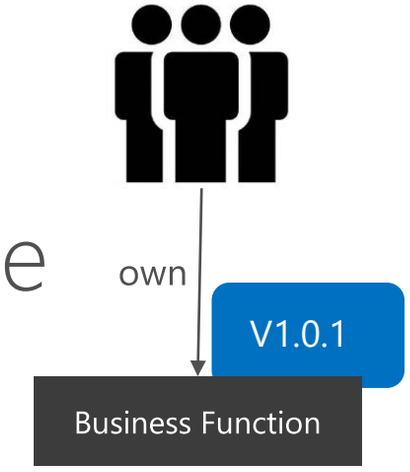


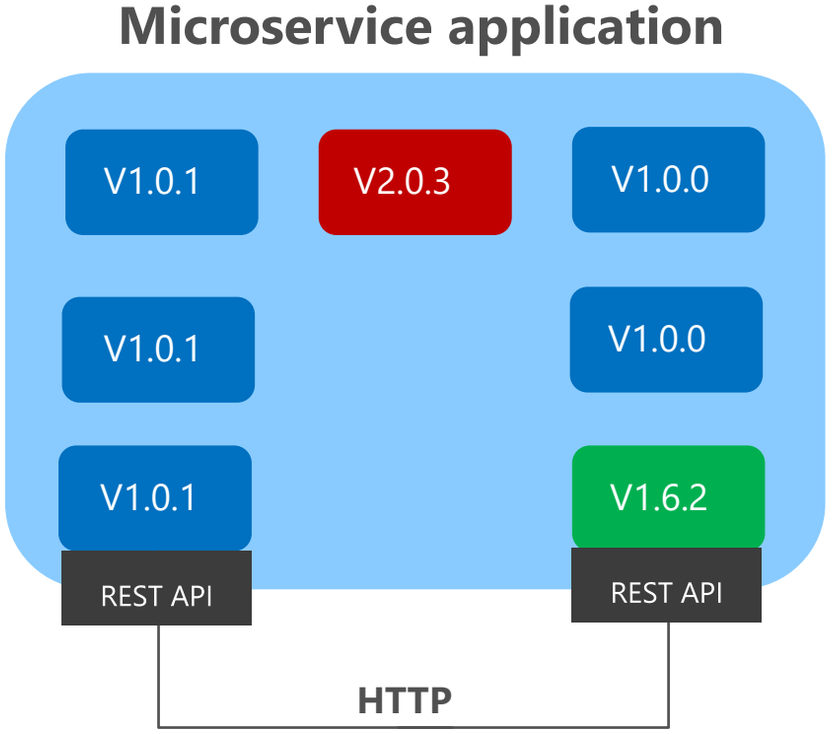
- Scales out by deploying each service independently creating instances of these services across servers/VMs/containers



Microservice Principles

A r
- i ed
- i able
- independently scalable





A microservice will:

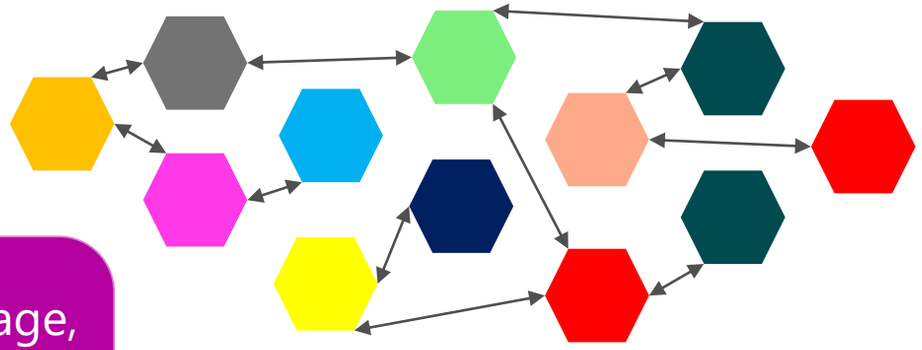
A microservice can:
- be developed by a small x-discipline engineering team
- talk to other microservices via standard protocols
- contain both code and dedicated state
- use other microservices via standard interfaces
- encapsulate a business function
- update, fail and restart whilst the system is running
- support mixed languages, tools and platforms*

What's the catch?

Circuit breakers, bulkheads?

How do I perform complex queries?

What about network latency?



CAP Theory?!

Who authorises who?

How do the microservices know about each other?

What language, Protocol, Interface, Should we use?

How do we monitor our system?

What happens if a service fail?

"Turns out that selecting the runtime and language is just one step in a long journey to a microservices architecture."

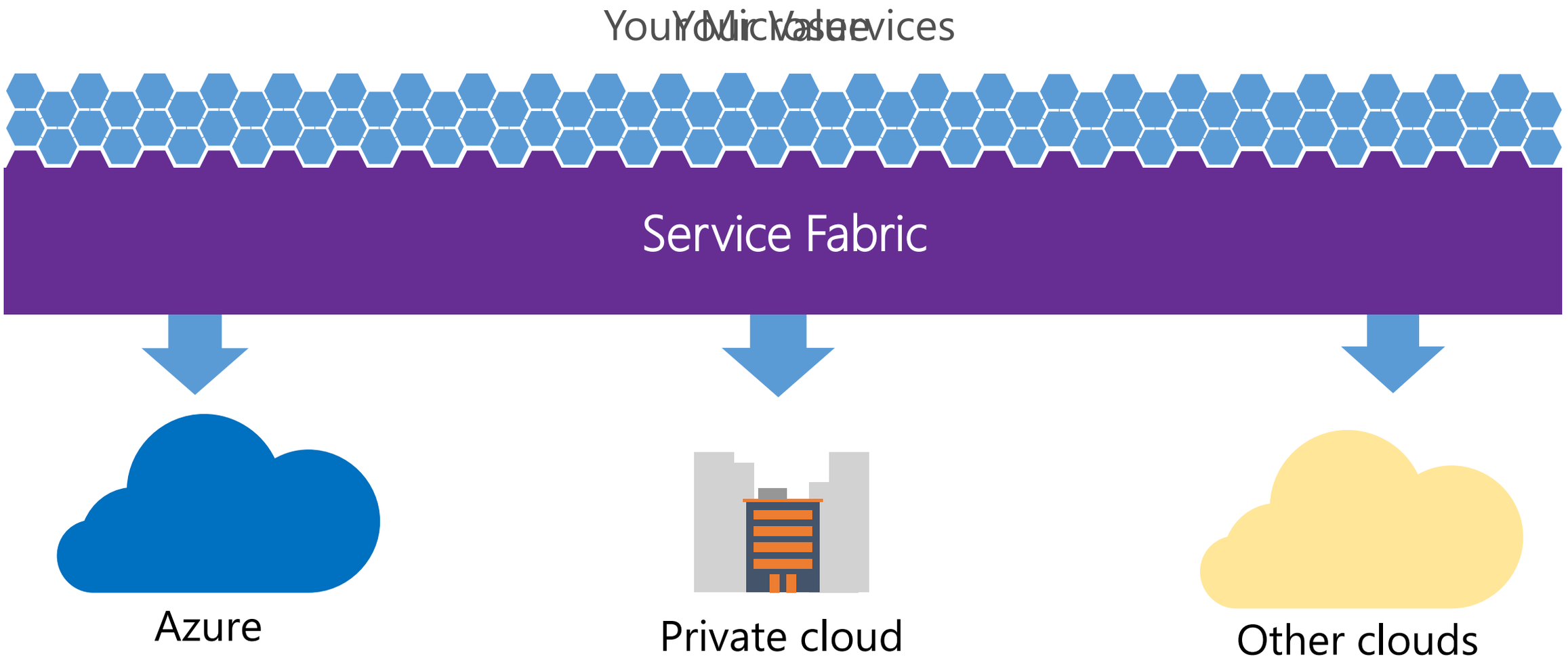
- Phil Calçado, Director @ DigitalOcean (formerly SoundCloud)

Service Fabric PUBLIC PREVIEW

Managing the complexity

Microsoft Service Fabric

A platform for reliable, hyperscale, microservice-based applications



Programming Models

Reliable Actors

STATELESS STATEFUL

- Many small independent units of state (i.e. sensor, smart car, etc.)
- Single threaded objects
- Service Fabric manages concurrency, granularity of state and communication

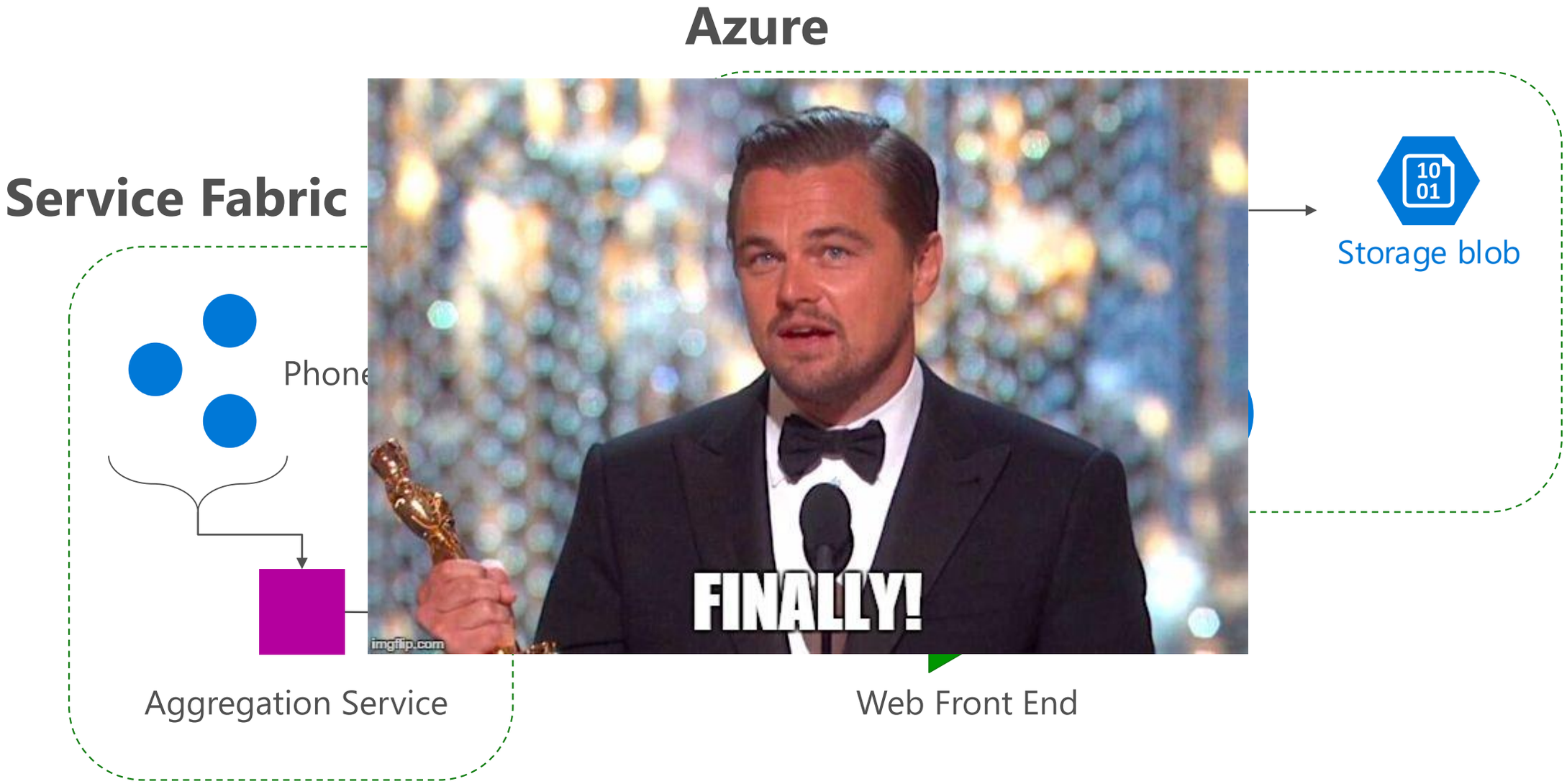
Reliable Services

STATELESS STATEFUL

- You can use reliable collections (Dictionary and Queue) to store and manage state.
- You manage concurrency, granularity of state and communication

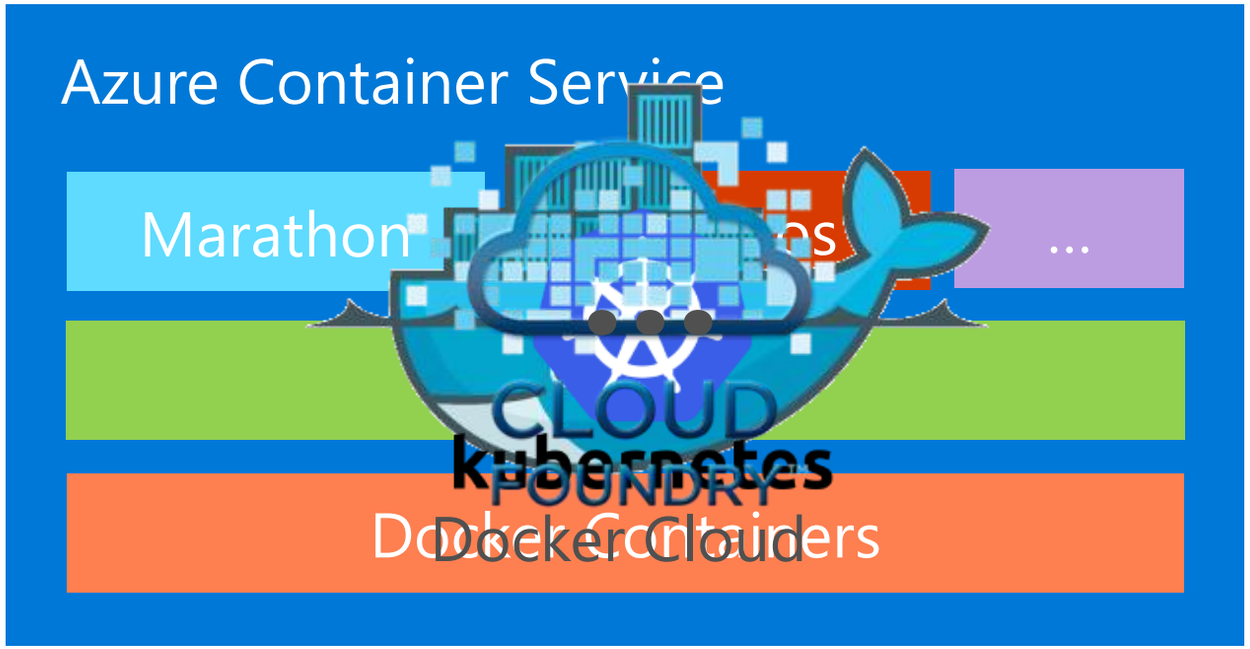
*Stateful state is stored on the nodes not in a database

Demo


Azure
Service Fabric
Phone
Aggregation Service
Web Front End
Storage blob
10
01
FINALLY!
imgflip.com

Antifragility

Microservices in Azure



Summary

Scale

Scaling microservices in isolation provides higher resource utilisation and reduces costs.

Agility

Small codebases, teams and deployment cycles enables rapid development and shortens recovery

Any Questions?

Availability

Replication of services and state enables redundancy

Flexibility

Ability to use mixed languages, tools and platforms within each microservice

Adaptability

Isolated service errors allows for service degradation.
Live updates supports an evolving app