Designing Distributed Systems

BRENDAN BURNS

DISTINGUISHED ENGINEER – MICROSOFT AZURE

CO-FOUNDER – KUBERNETES PROJECT

Microsoft

This job is too hard

A history lesson

Development in the 1940s and 1950s

Microsoft

A history lesson

Development in the 1940s and 1950s

Problem: Assembly Language

Solution: Fortran (1954)

Patterns: Knuth – Art of Computer Programming (1962-1968)

A history lesson

Development in the 1970s and 1980s

A history lesson

Development in the 1970s and 1980s

Problem: Large codebases, interchangeable teams.

Solution: Object Oriented Programming (c. 1985 [really much earlier])

Patterns: Gang of Four – Design Patterns: Elements of Reusable Object-Oriented Software (1994)

A (modern) history lesson

Development in the 2000s and 2010s

A (modern) history lesson

Development in the 2000s and 2010s

Problem: Distributed applications, scale and reliability

Solution: Containers and Orchestration (2013-2014 [and earlier])

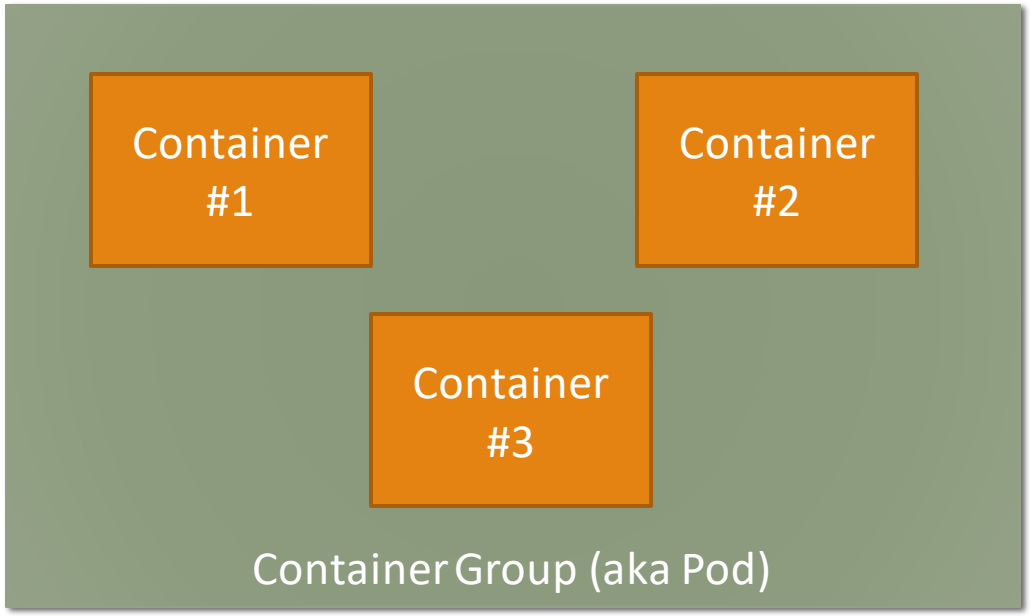
Patterns: Designing Distributed Systems - OSCON 2018?

Agenda

- Patterns
  - Single Node Patterns
  - Multi-Node patterns
- Tools
  - Cluster Daemons
  - Cluster Agents
  - Intent-based APIs

Single Node Patterns

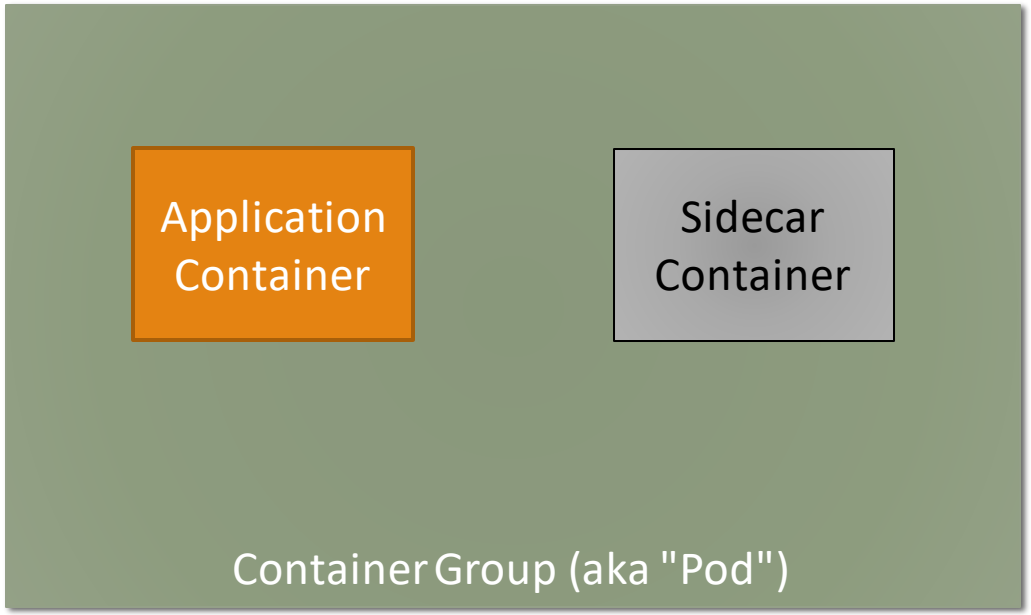
Focus on component re-use and organization



Microsoft

Single Node Patterns: Sidecar

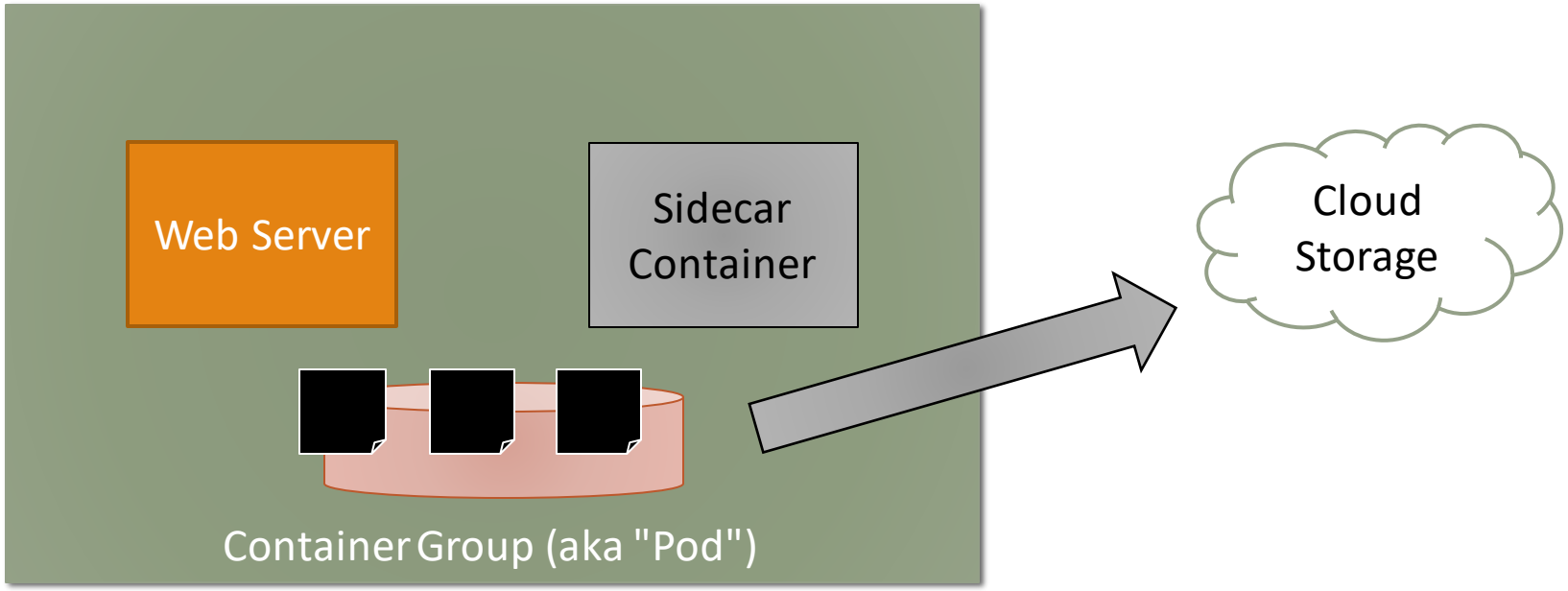
Sidecars augment and extend



Microsoft

Single Node Patterns: Sidecar

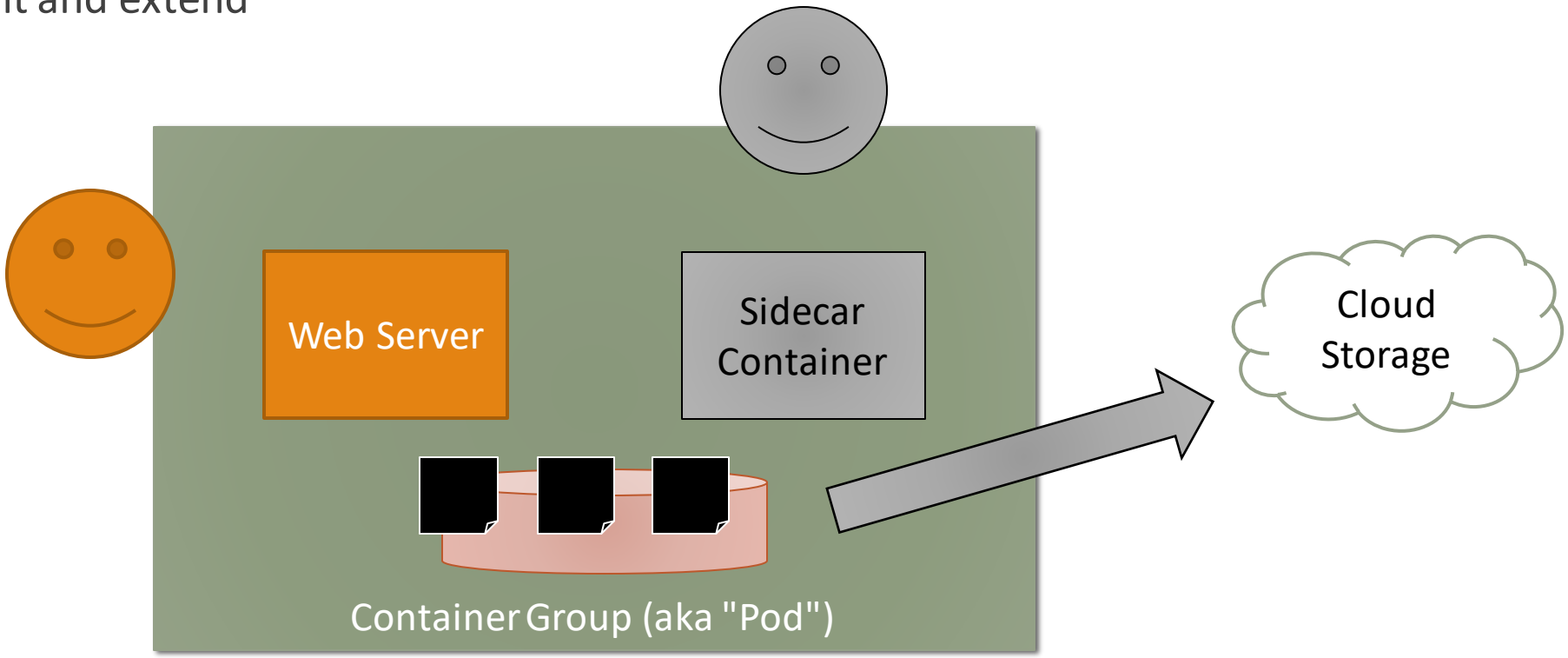
Sidecars augment and extend



Microsoft

Single Node Patterns: Sidecar

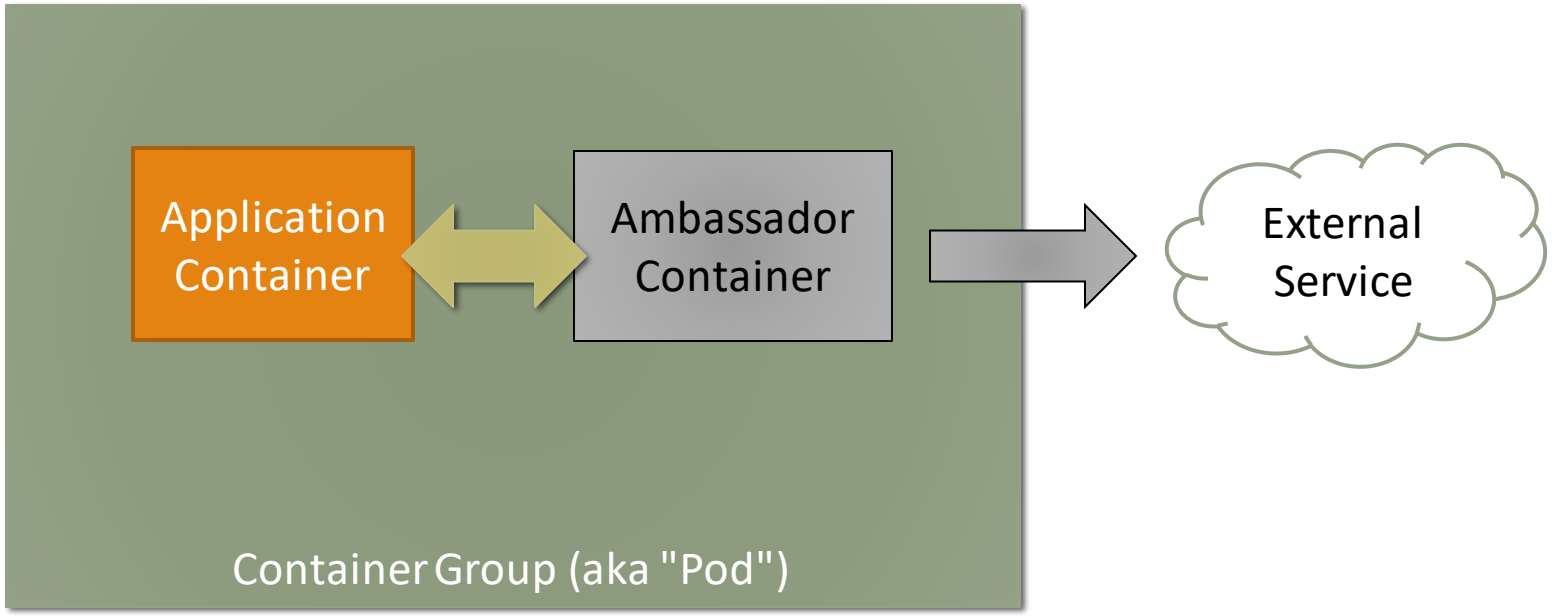
Sidecars augment and extend



Microsoft

Single Node Patterns: Ambassador

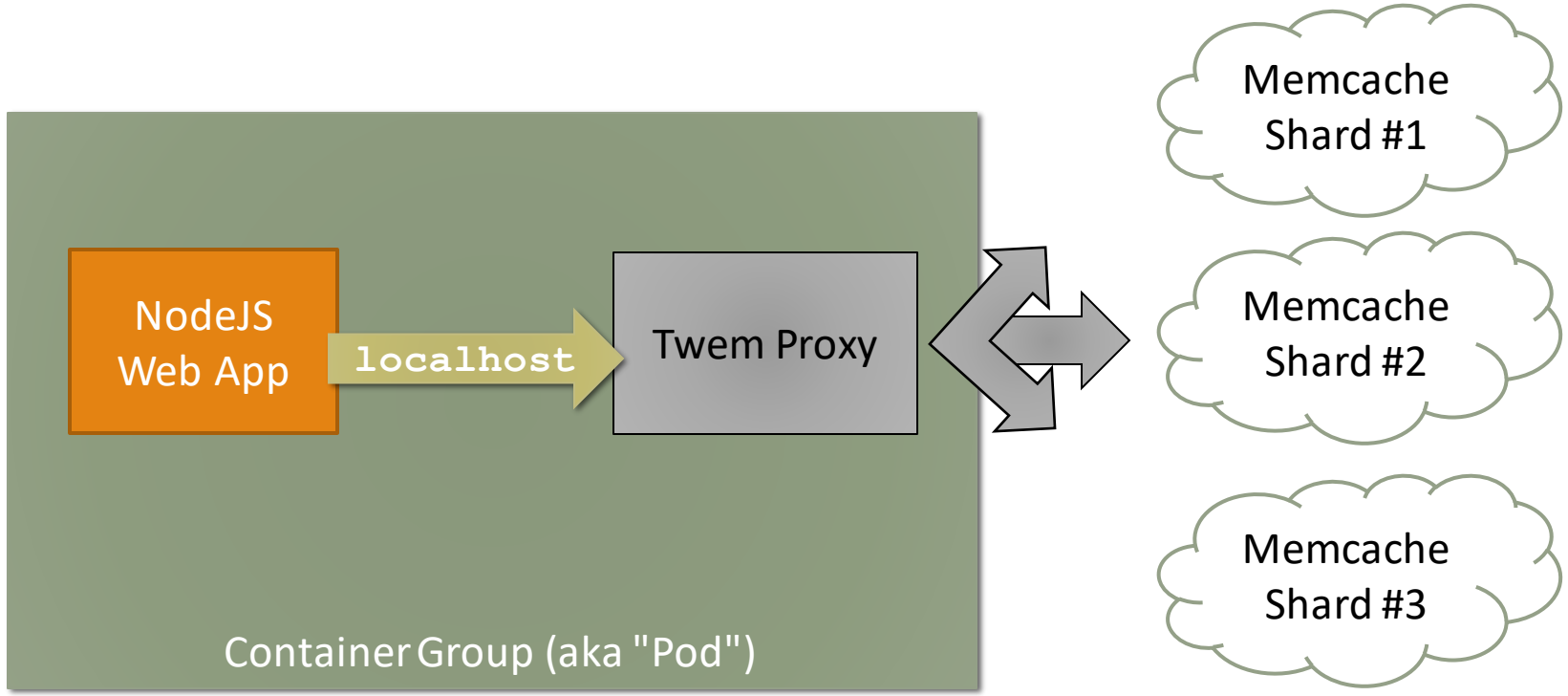
Ambassadors translate and represent



Microsoft

Single Node Patterns: Ambassador

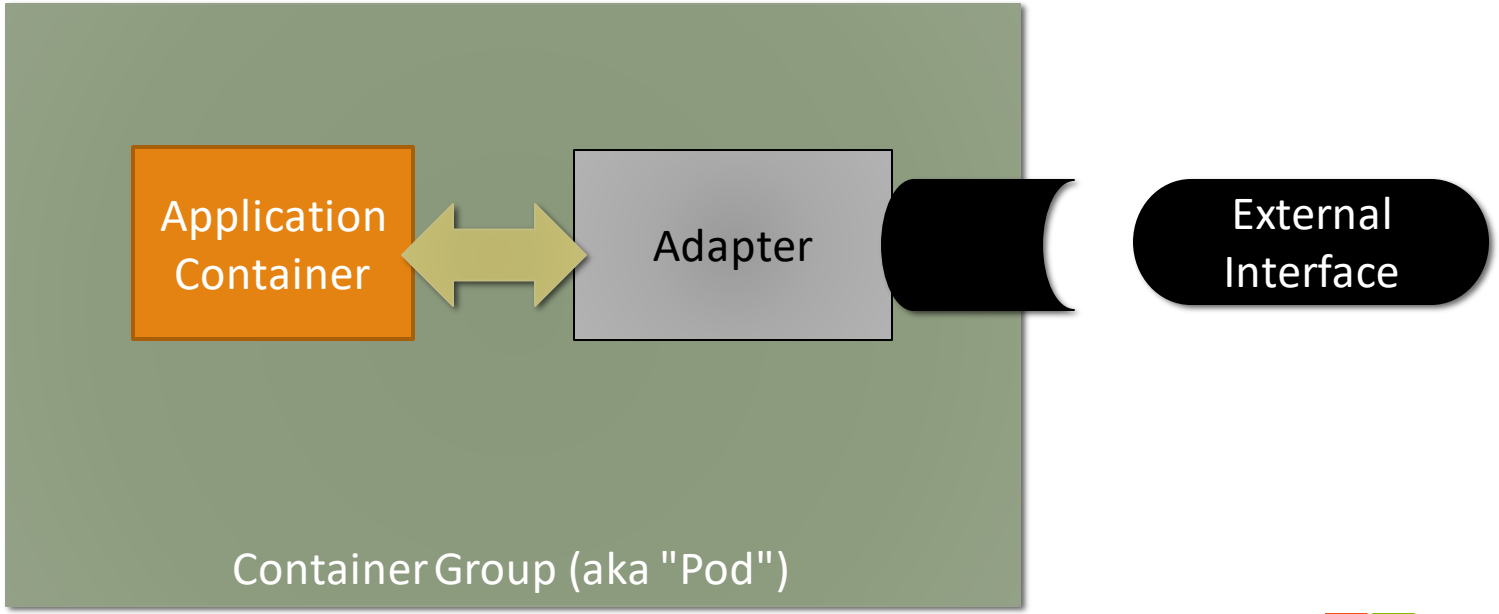
Ambassadors translate and represent



Microsoft

Single Node Patterns: Adapter

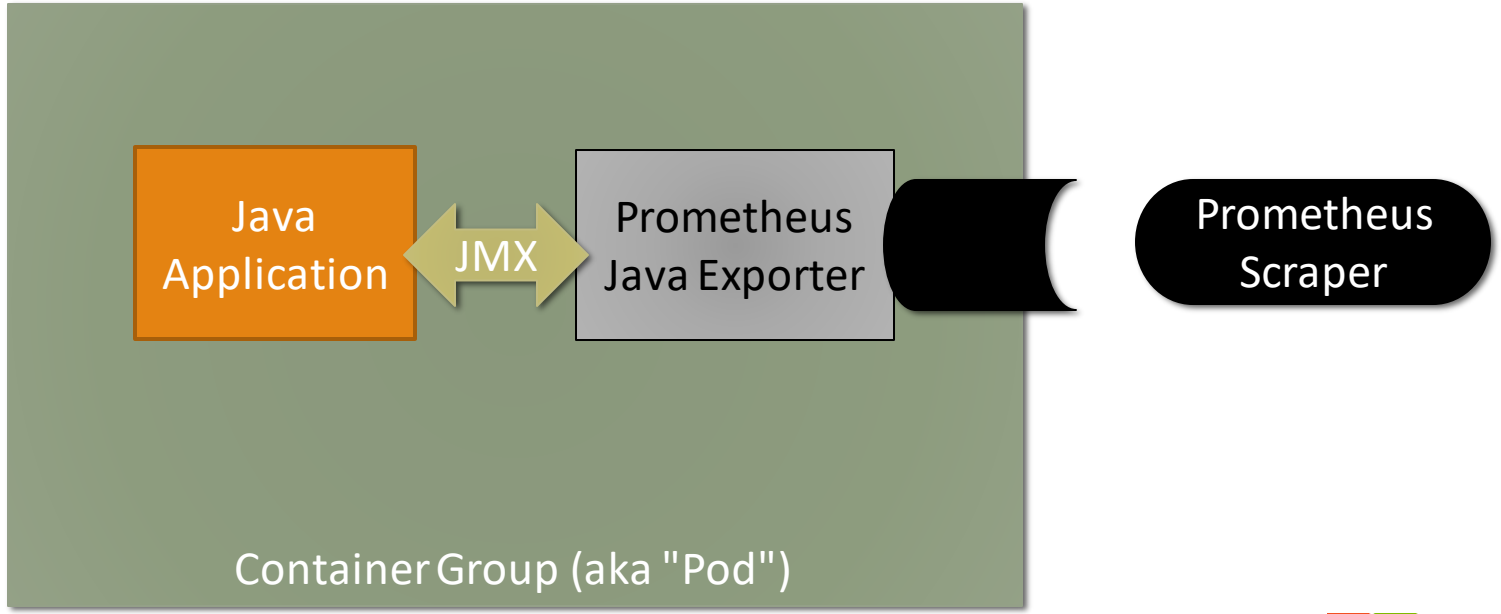
Adapters Standardize and Normalize



Microsoft

Single Node Patterns: Adapter

Adapters Standardize and Normalize



Microsoft

Multi-Node Patterns

Share and learn from existing "best practices"

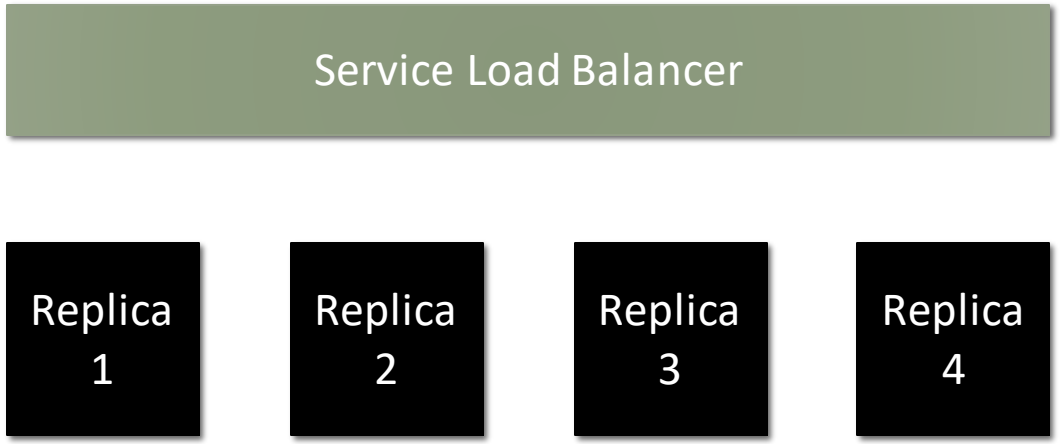
Give us a common vocabulary for discussing our systems

Enable the development of shared implementations

Multi-Node Patterns: Replicated

Reliable, redundant serving

Safe, zero-downtime rollouts

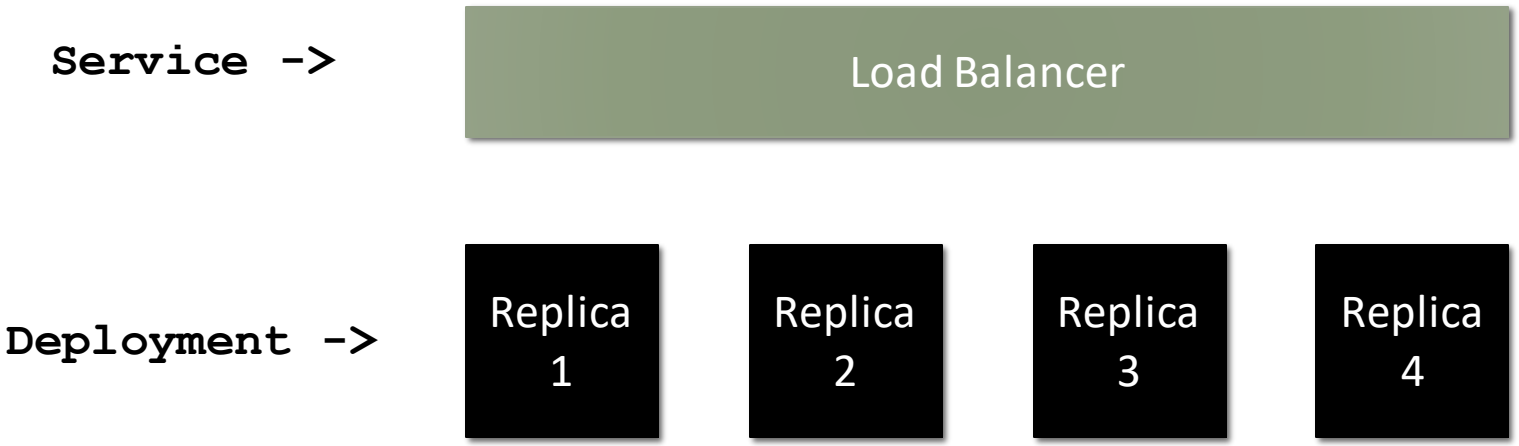


Microsoft

Multi-Node Patterns: Replicated

Reliable, redundant serving

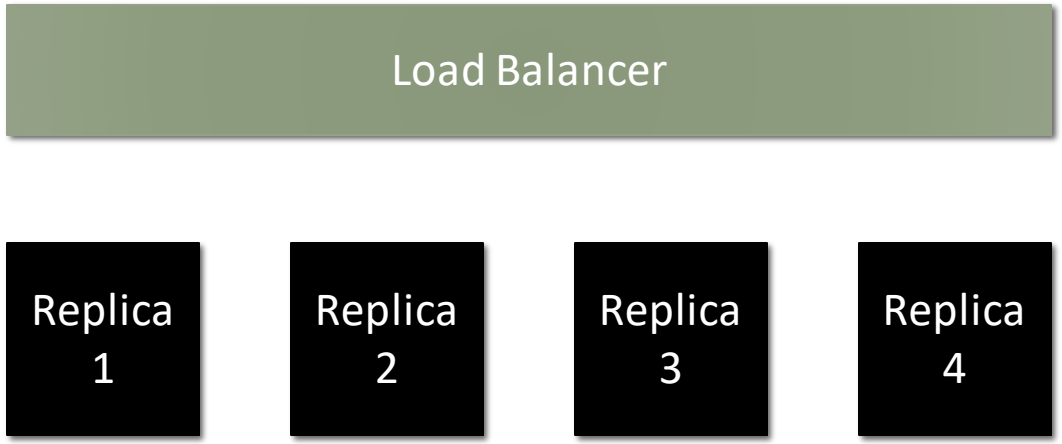
Safe, zero-downtime rollouts



Microsoft

Multi-Node Patterns: Replicated

```
const myService =
  new ReplicatedService(
    image: "acr.io/brendan/my-server:v1",
    ports: [ 8080 ],
    replicas: 4);
```



https://metaparticle.io/tutorials

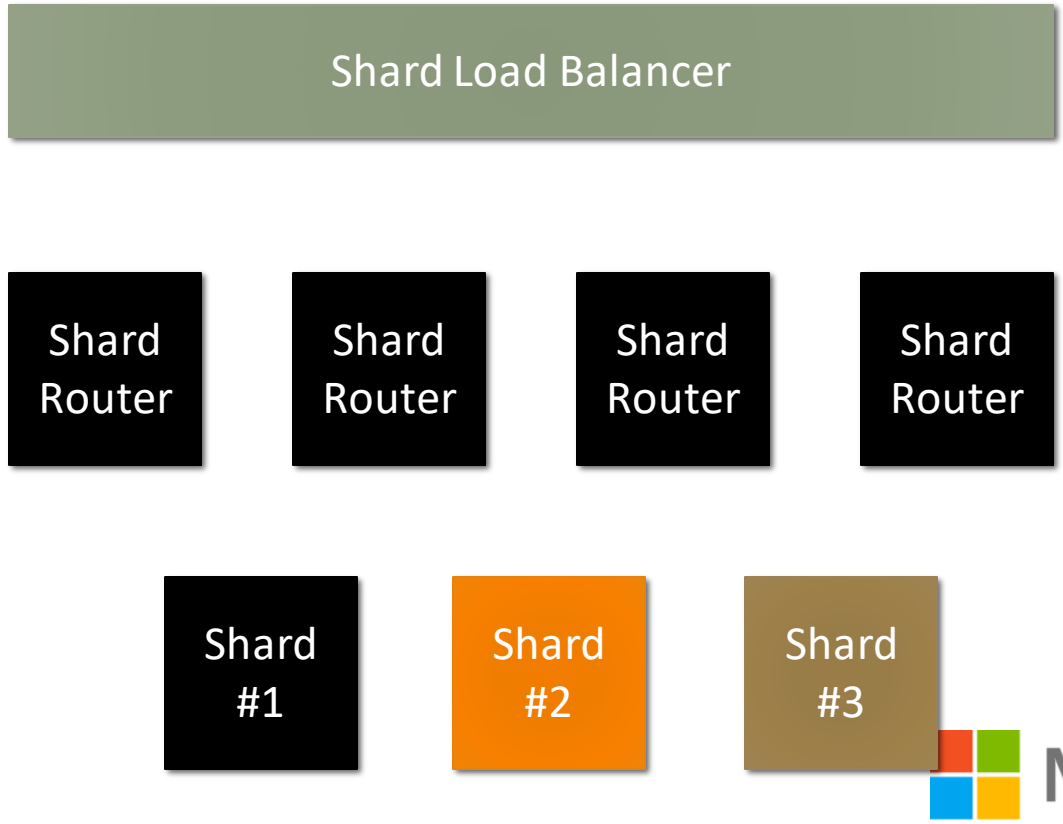
Microsoft

Multi-Node Patterns: Sharded

Sharding increases cache hit rates

Enables larger data-stores

Changes failure patterns



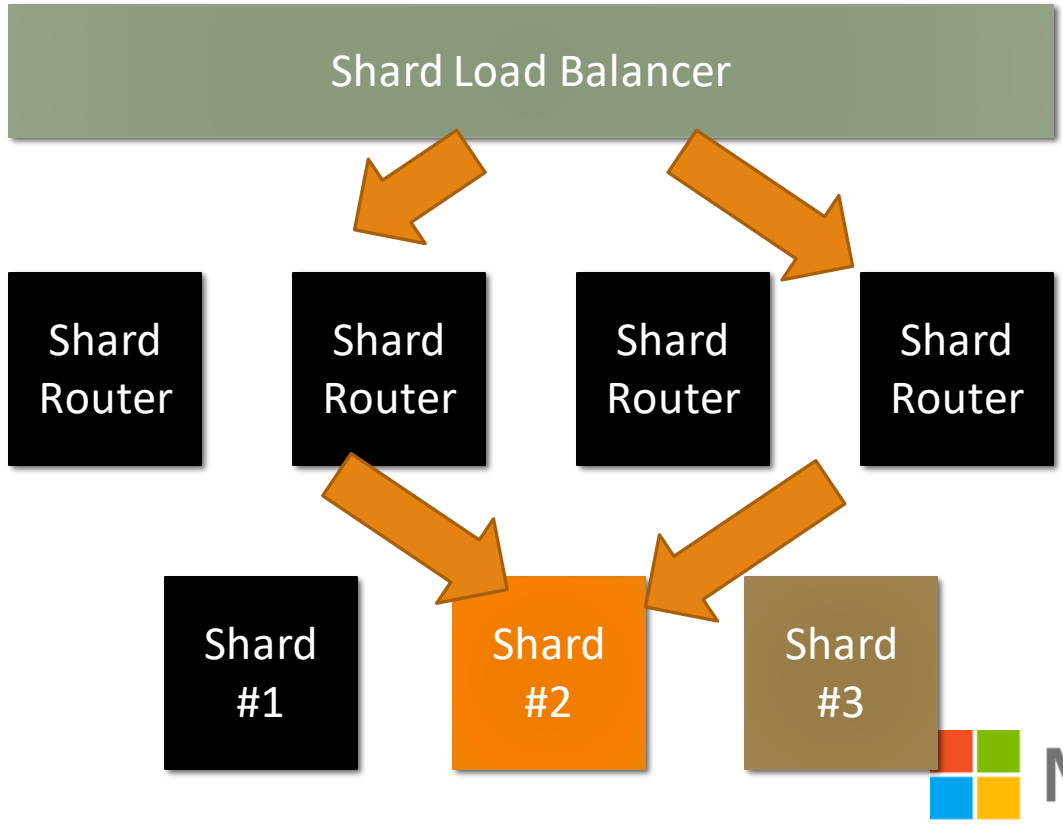
Microsoft

Multi-Node Patterns: Sharded

Sharding increases cache hit rates

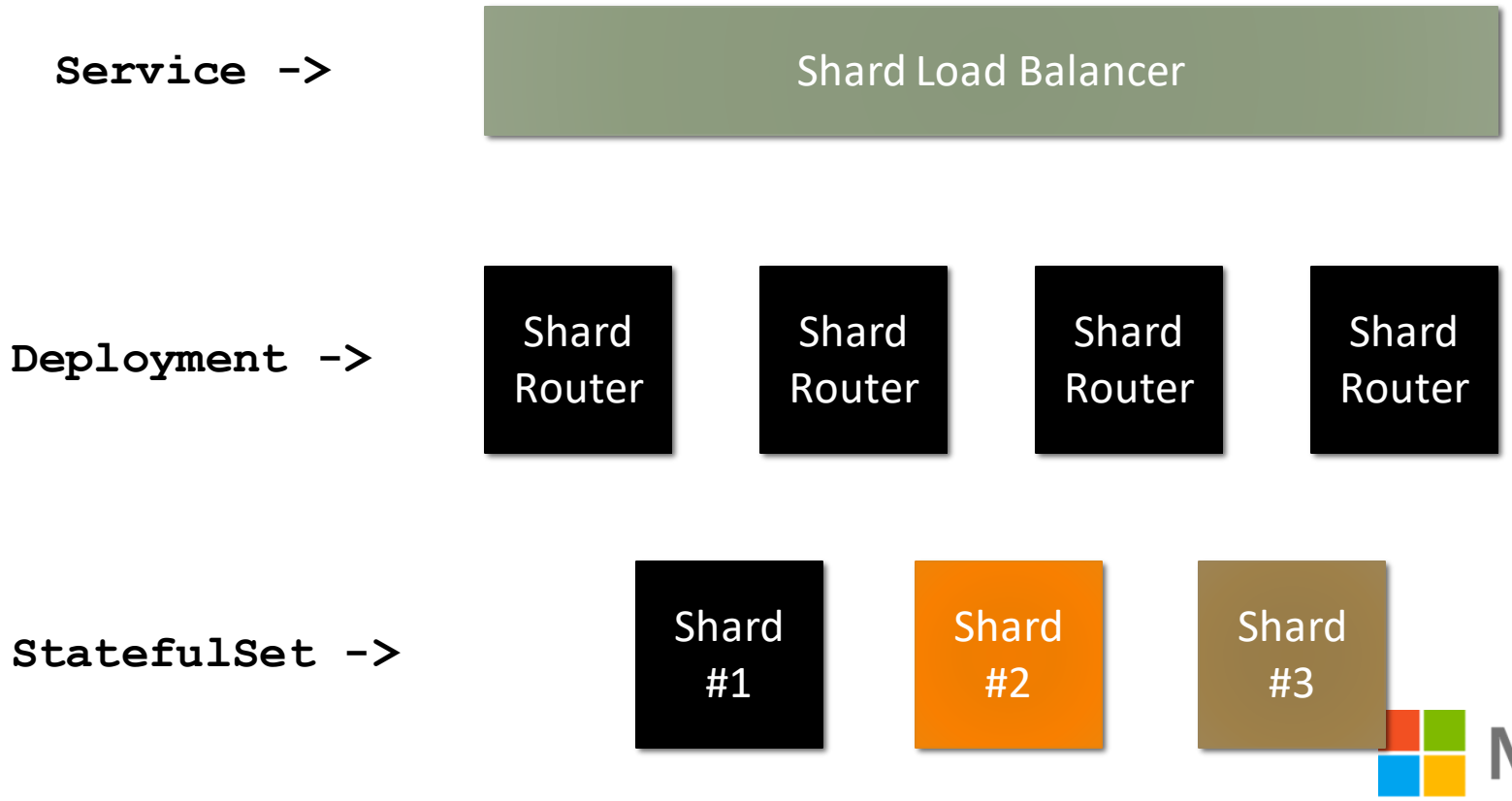
Enables larger data-stores

Changes failure patterns



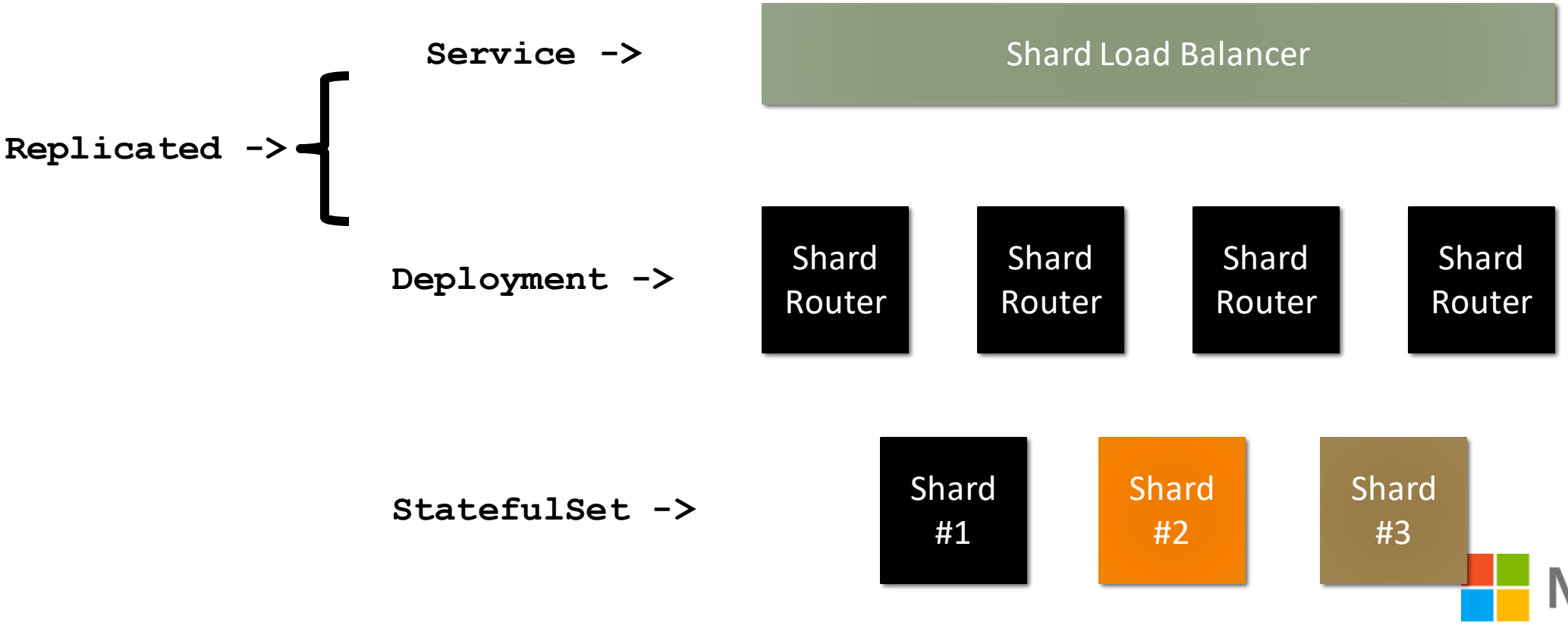
Microsoft

Multi-Node Patterns: Sharded



Microsoft

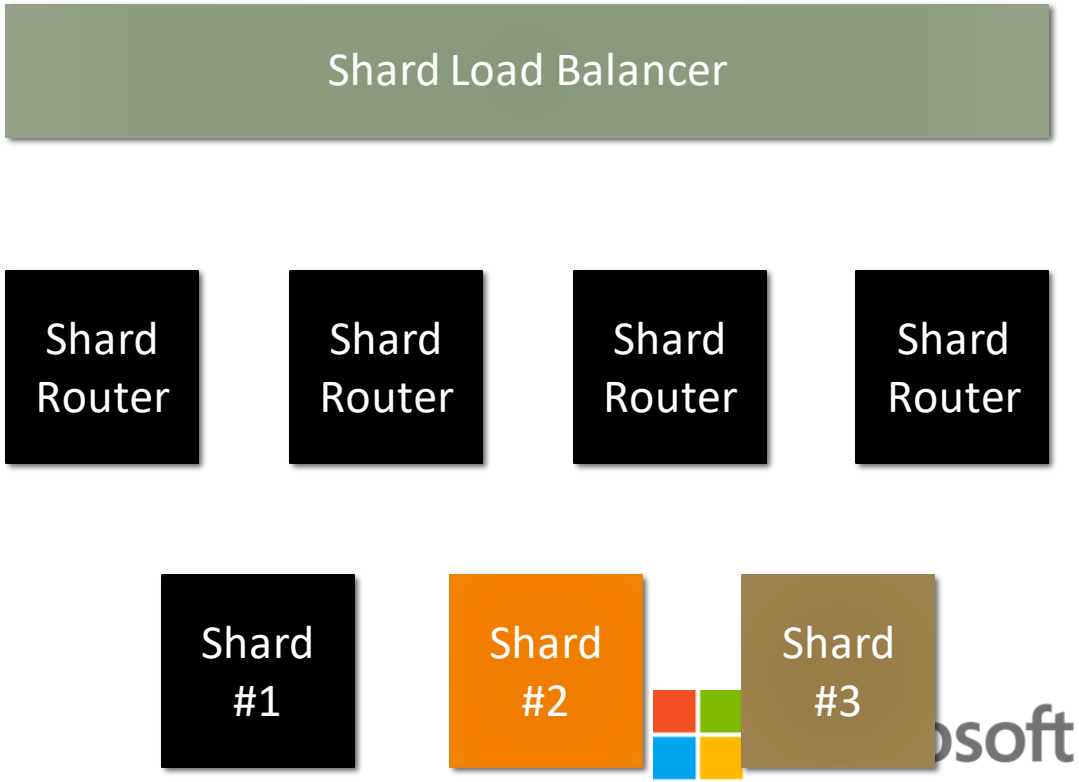
Multi-Node Patterns: Sharded



Microsoft

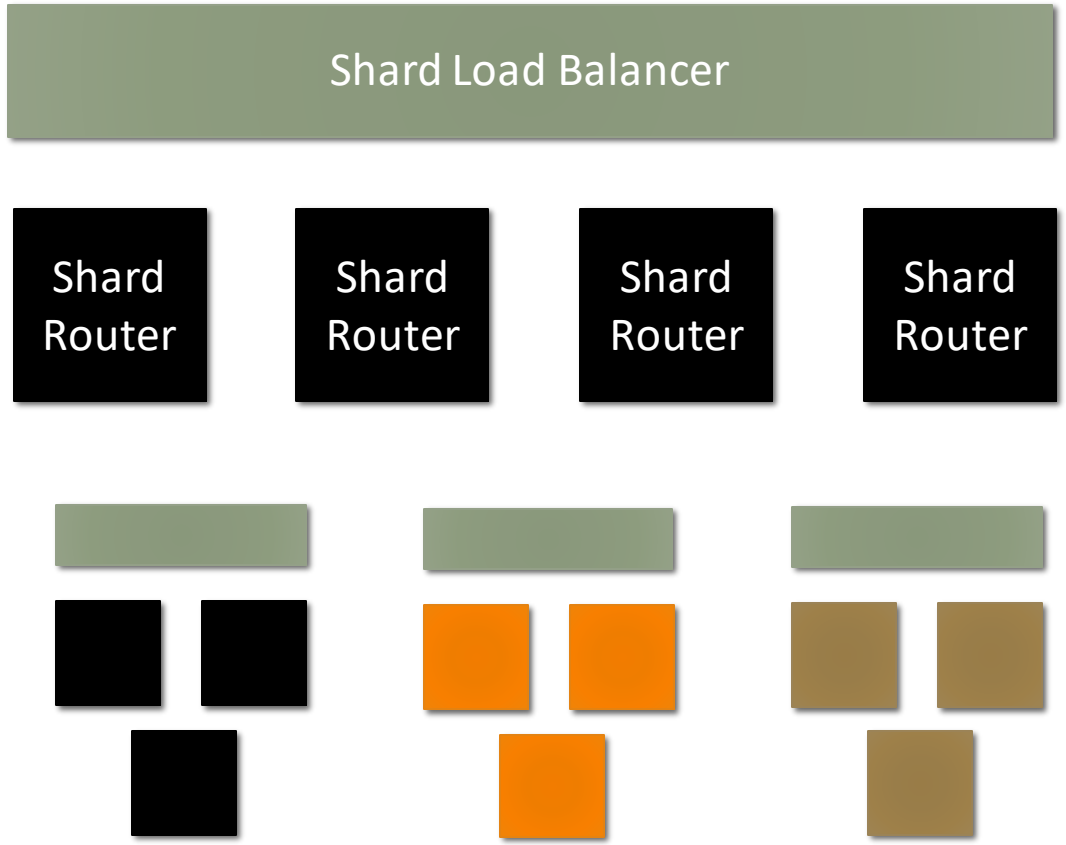
Multi-Node Patterns: Sharded

```
const myService =
  new ShardedService(
    image: "acr.io/brendan/my-server:v1",
    shardRegexp: "/users/(.*)/.*",
    ports: [ 8080 ],
    shards: 4);
```



https://metaparticle.io/tutorials

Multi-Node: Replicated & Sharded

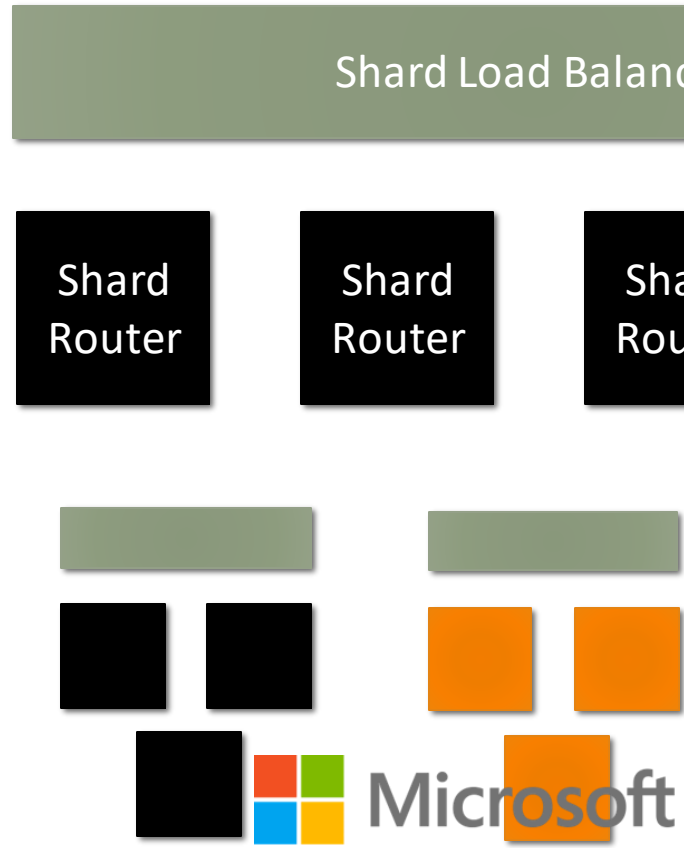


Microsoft

Multi-Node: Replicated & Sharded

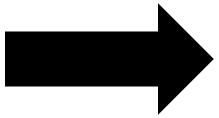
```
class ShardedReplicatedService : ShardedService {
    ReplicatedService[] replicatedShards;

    public ShardedReplicatedService() {
        …
    }
    …
}
```



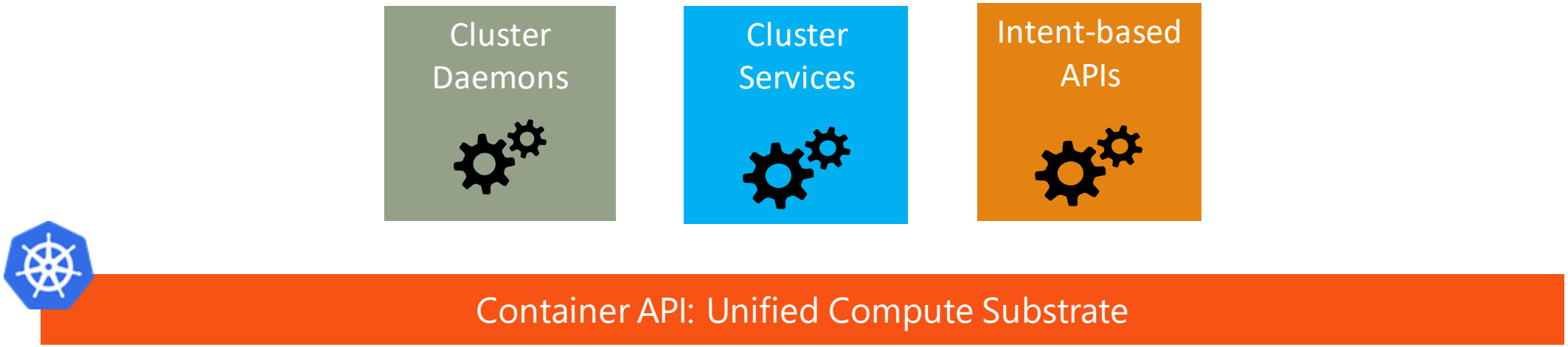
Agenda

- Patterns
  - Single Node Patterns
  - Multi-Node patterns
- Tools
  - Cluster Daemons
  - Cluster Agents
  - Intent-based APIs



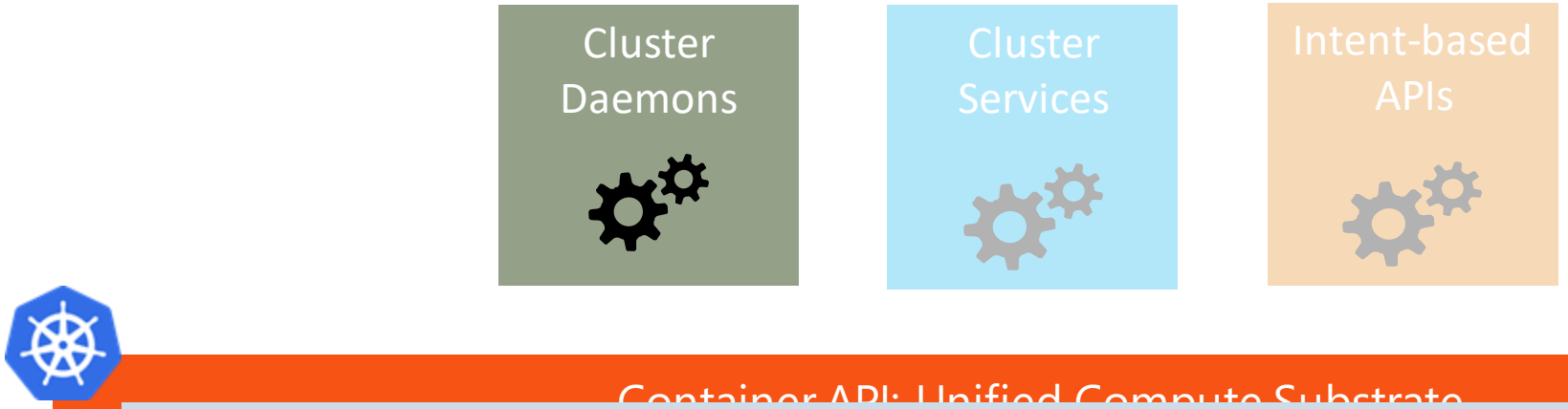
Microsoft

Tools: The cluster as the new node



Microsoft

Cluster Daemons



Distributed systems are become microservices

Microsoft

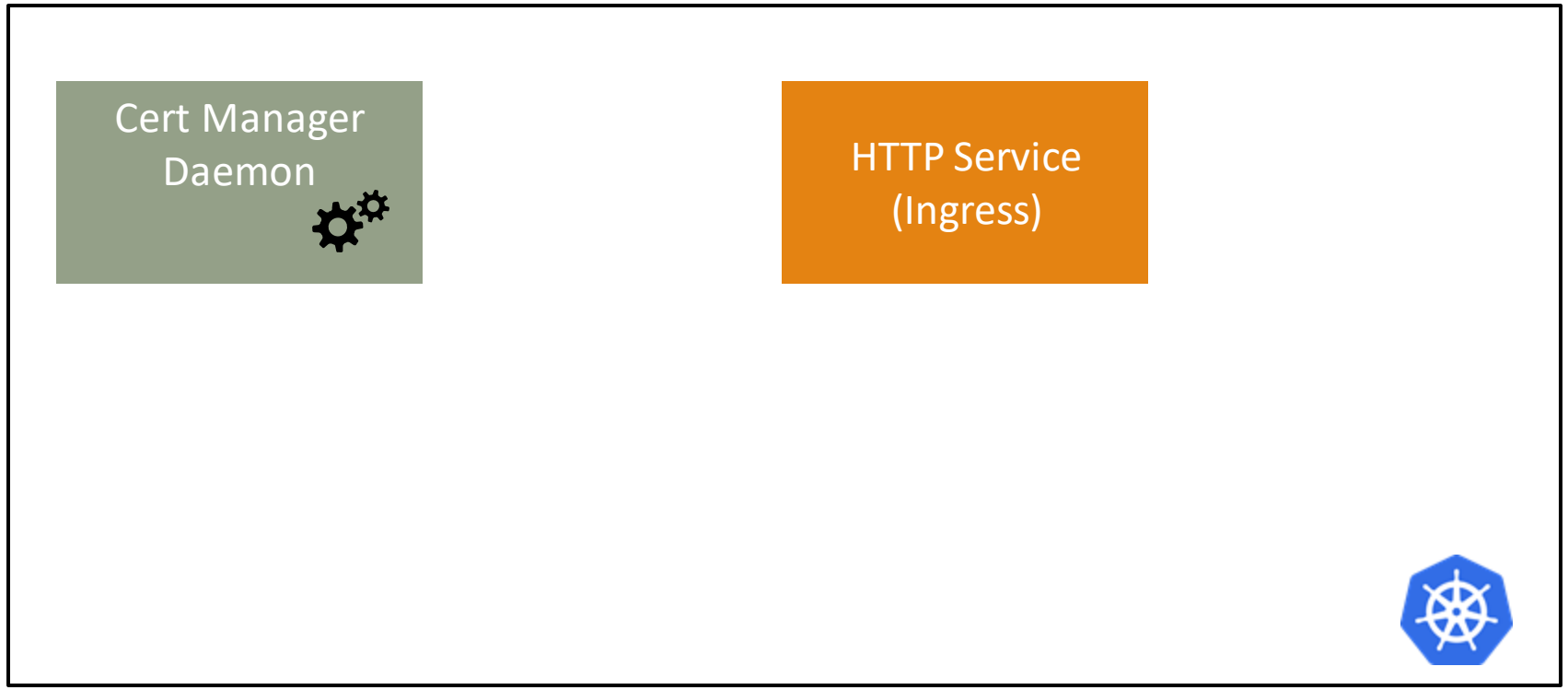
Cluster Daemons

- Atomic, value-added behaviors based on API object state
- Deployed ***onto the cluster itself***
- Enable “auto-magic” experiences

Microsoft

Cluster Daemons

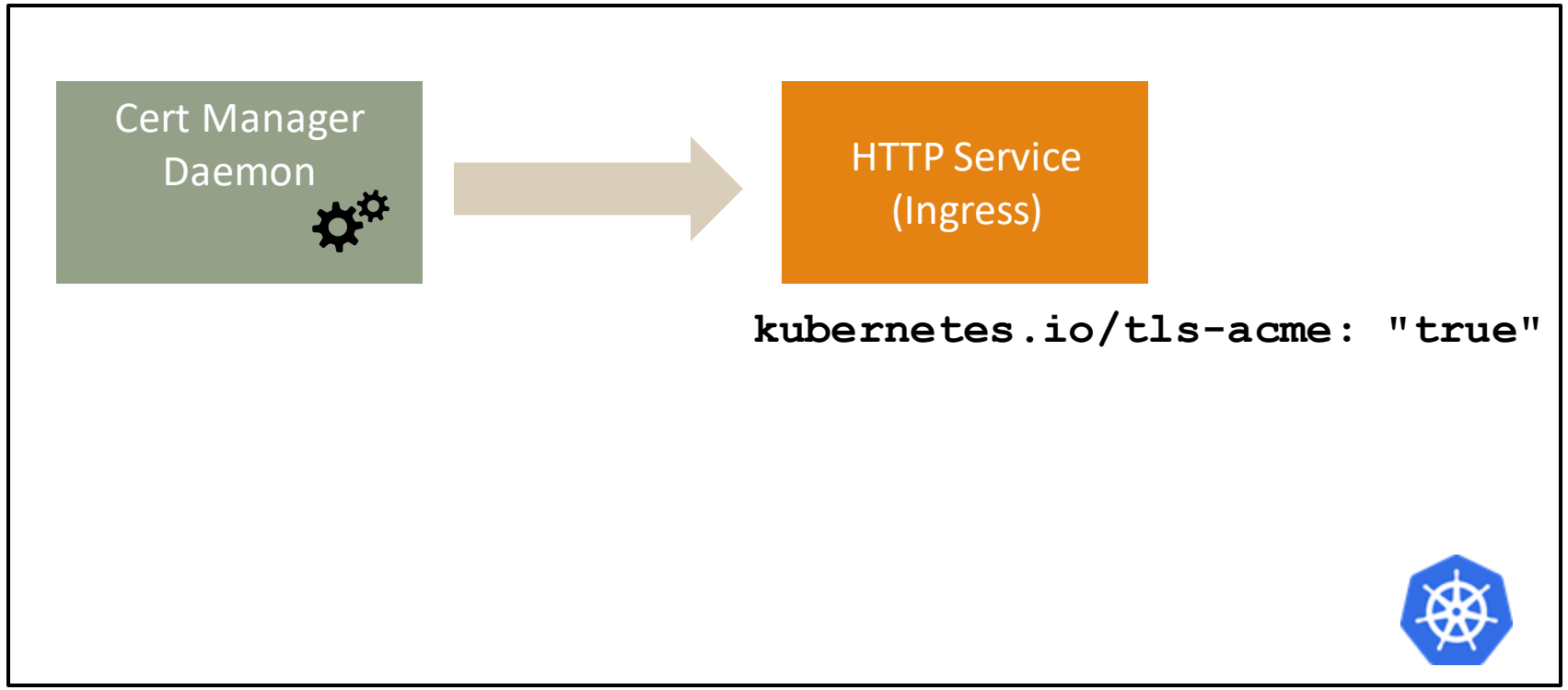
Distributed systems are becoming microservices



Microsoft

Cluster Daemons

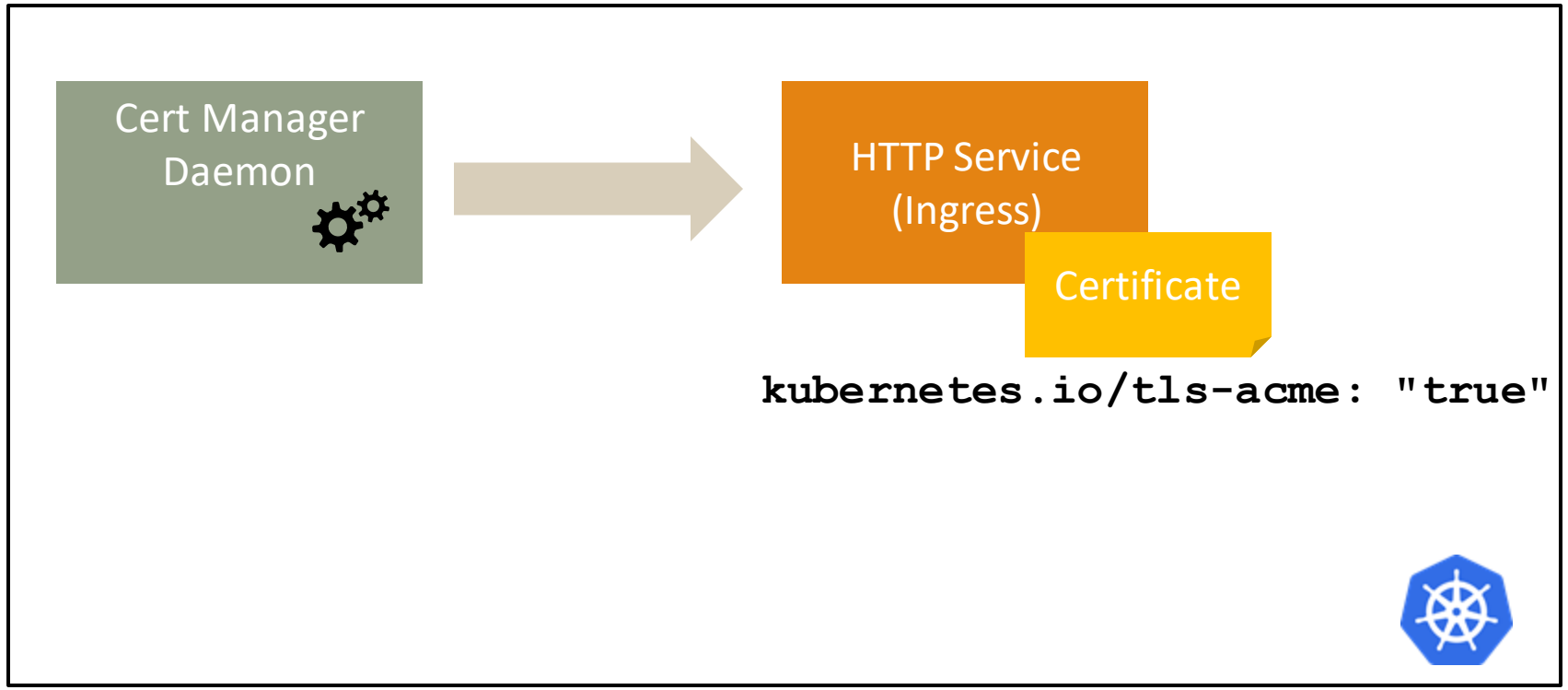
Distributed systems are becoming microservices



Microsoft

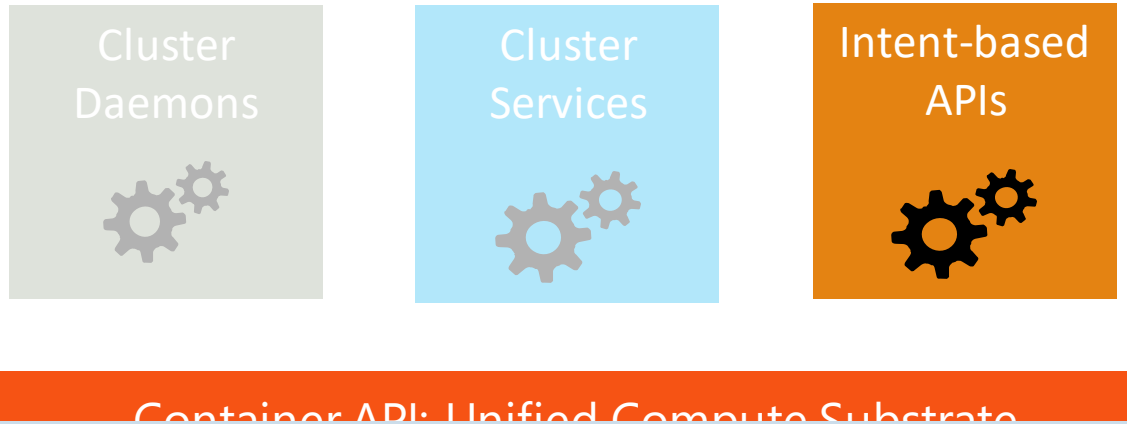
Cluster Daemons

Distributed systems are becoming microservices



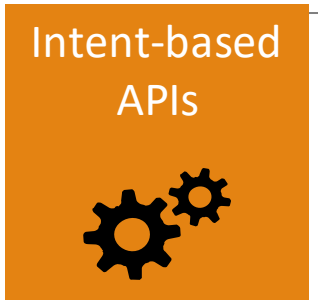
Microsoft

Intent-based APIs



Microsoft

Intent-based APIs



Too much time is spent building things we should just state
- Or, too little time is spent building such things

Wasted, duplicate effort

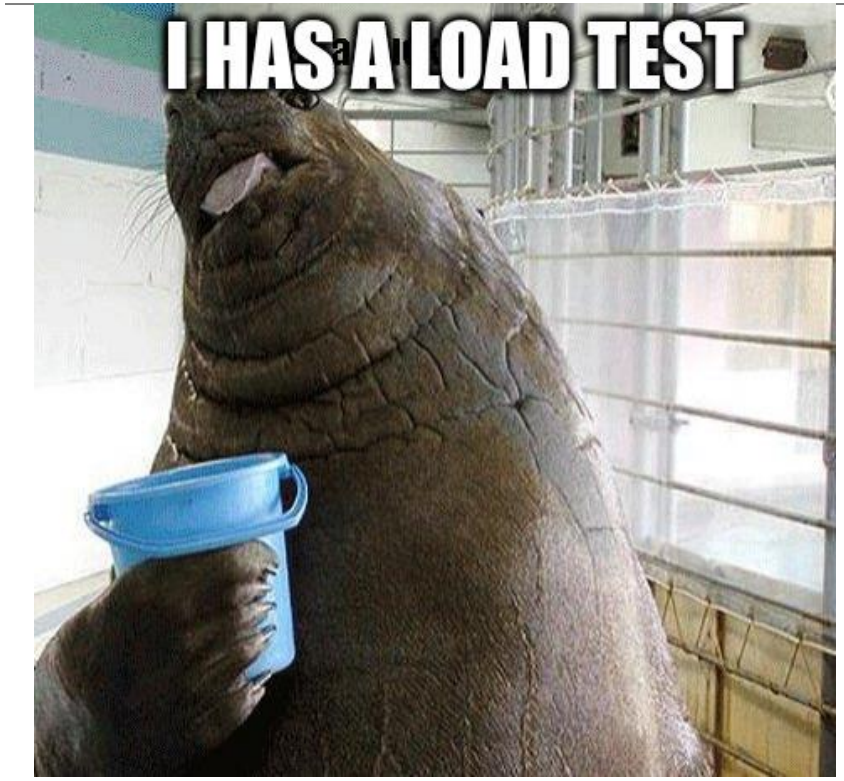
Barriers to entry for many people

Microsoft

Intent-based APIs

Microsoft

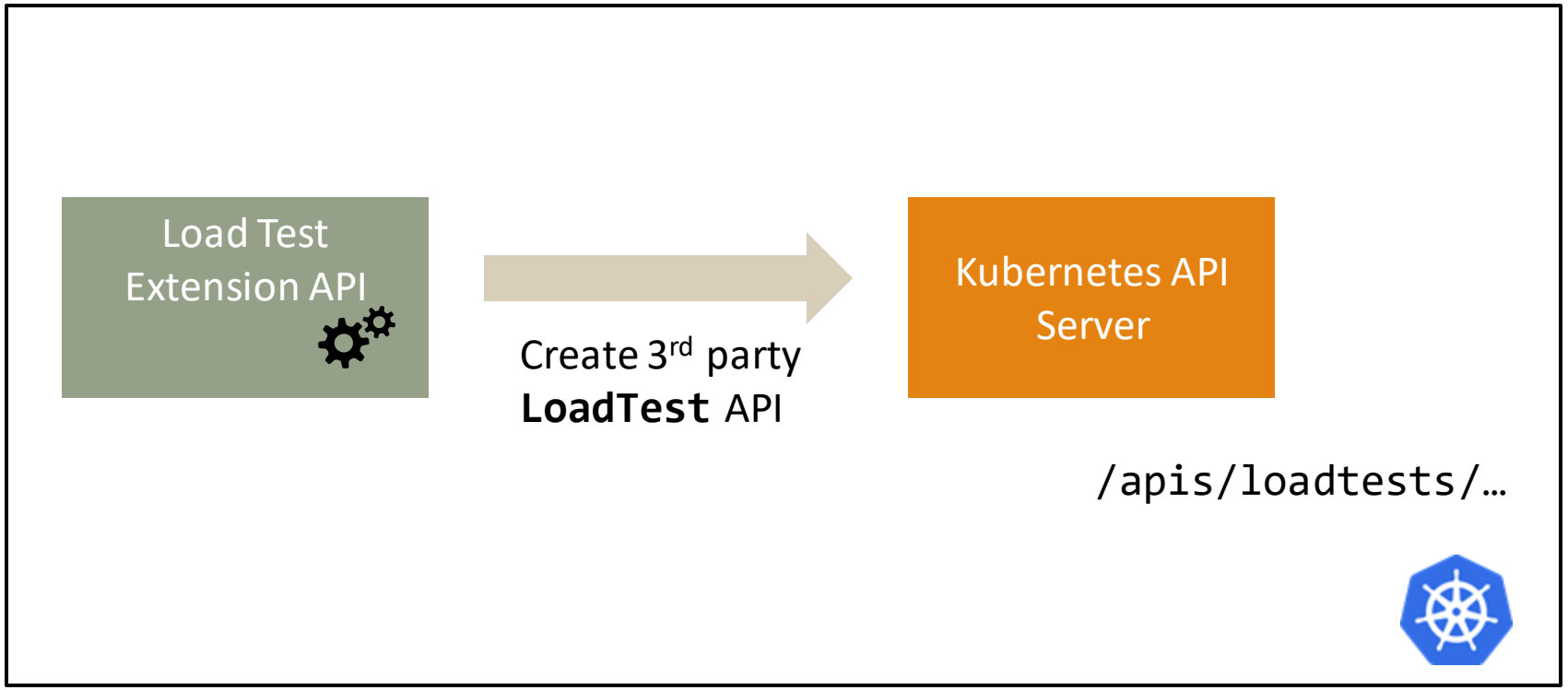
Intent-based APIs




```
apiVersion: loadtest/v1
kind: LoadTest
metadata:
  name: my-cool-loadtest
  labels:
     …
spec:
  service: my-cool-service
  requestPerSecond: 10k
  tests:
    - test:
      - /some/path
      - /some/other/path
```

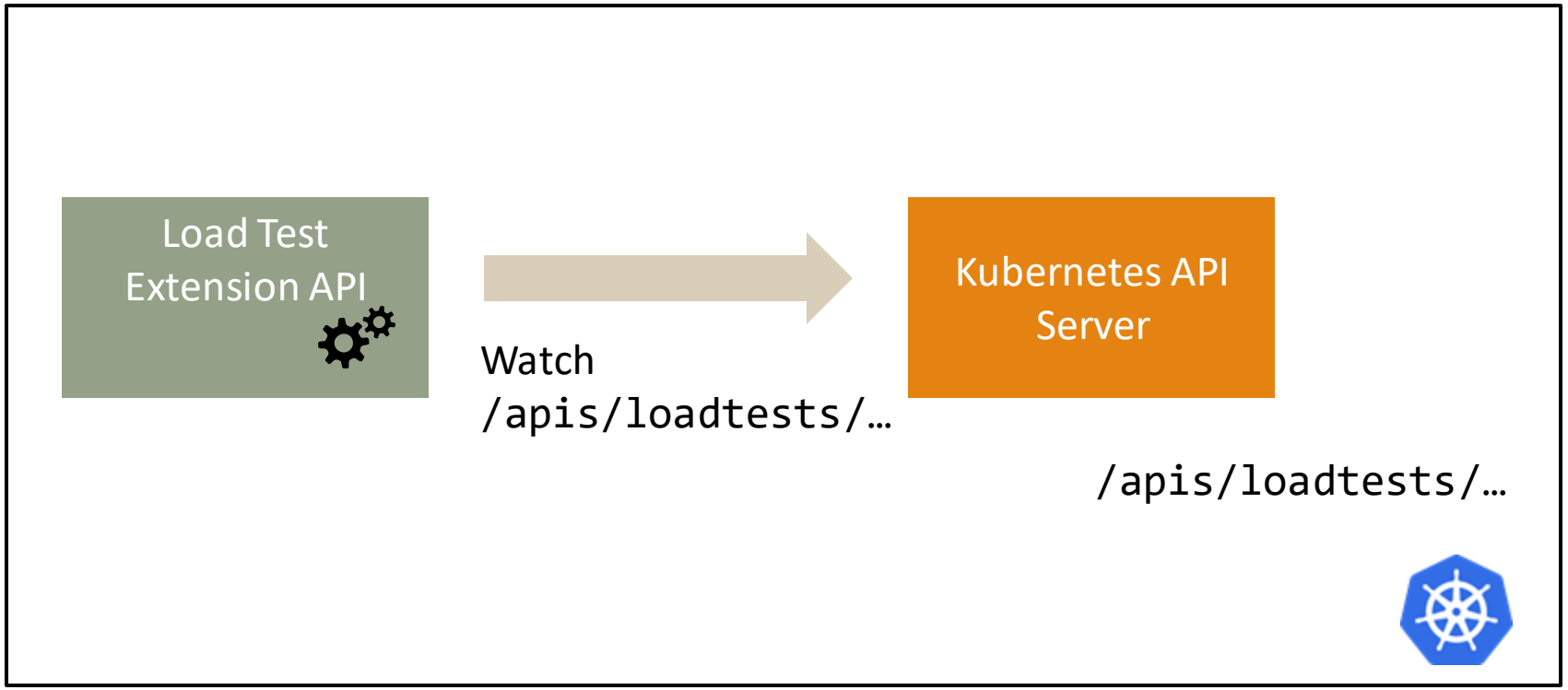
Microsoft

Intent-based APIs



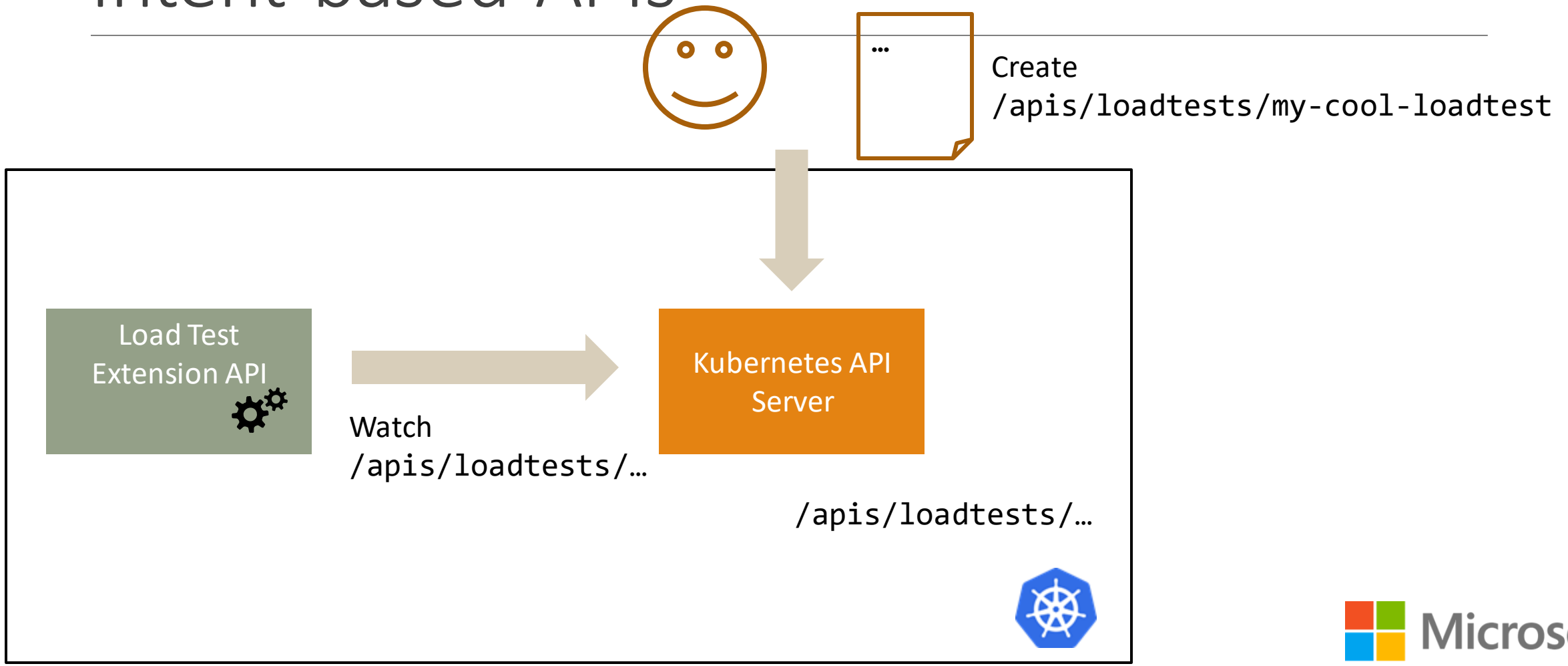
Microsoft

Intent-based APIs



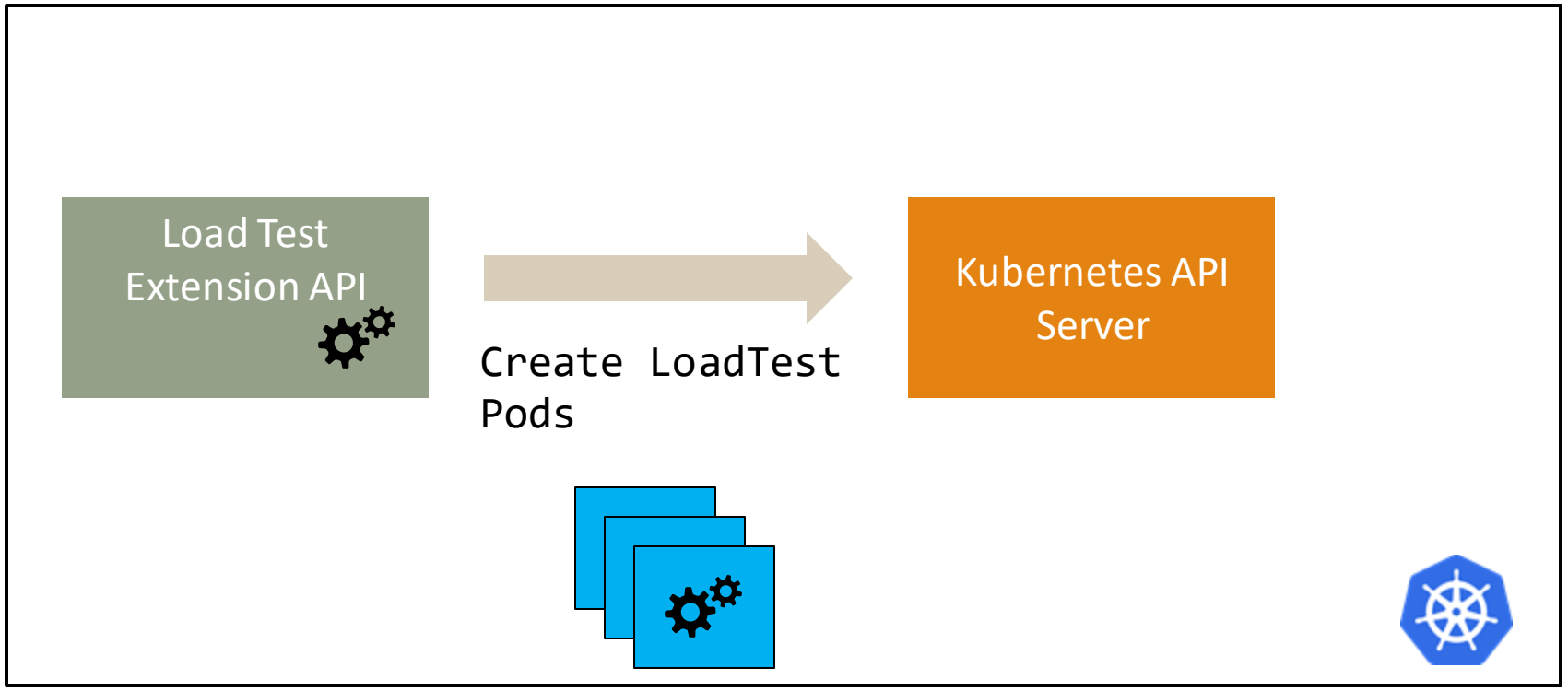
Microsoft

Intent-based APIs



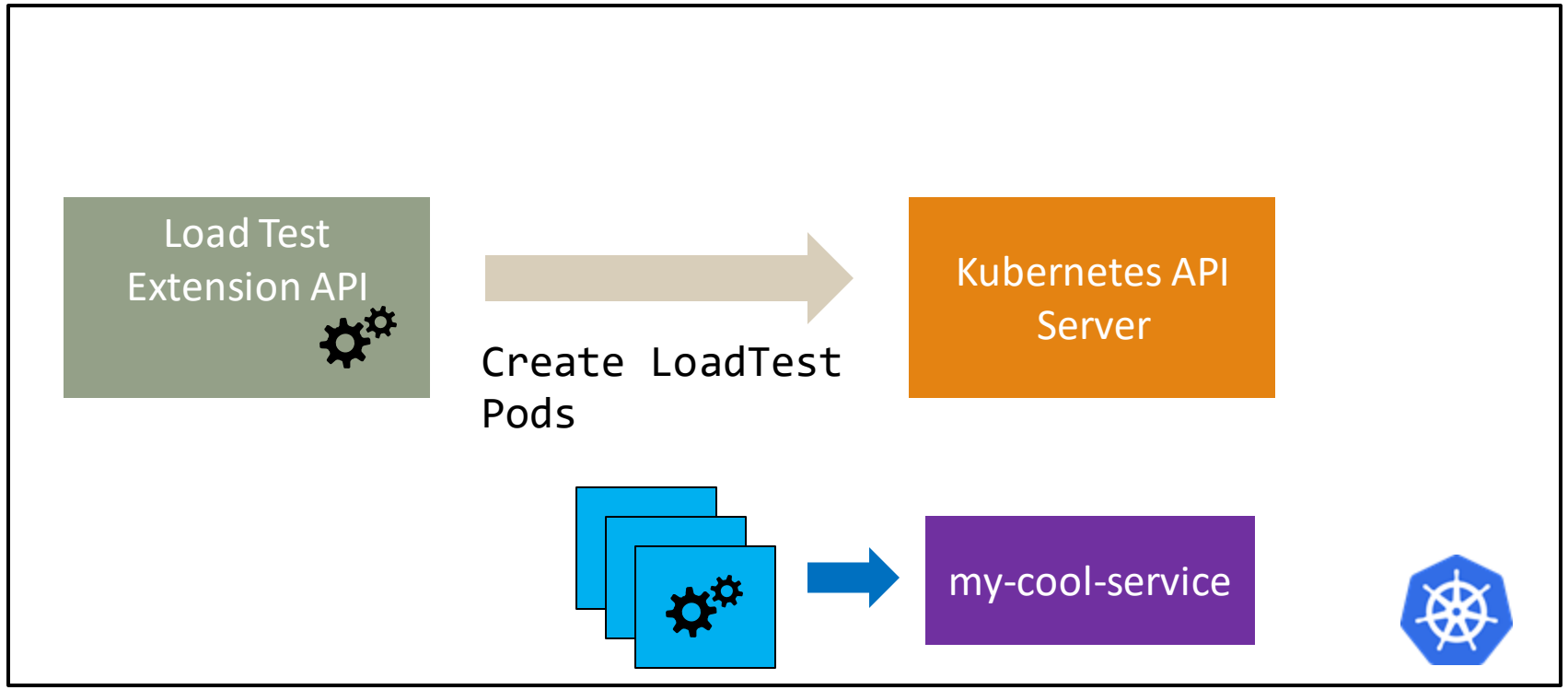
Microsoft

Intent-based APIs



Microsoft

Intent-based APIs



Microsoft

Cluster Services

Cluster wide standardization of common services

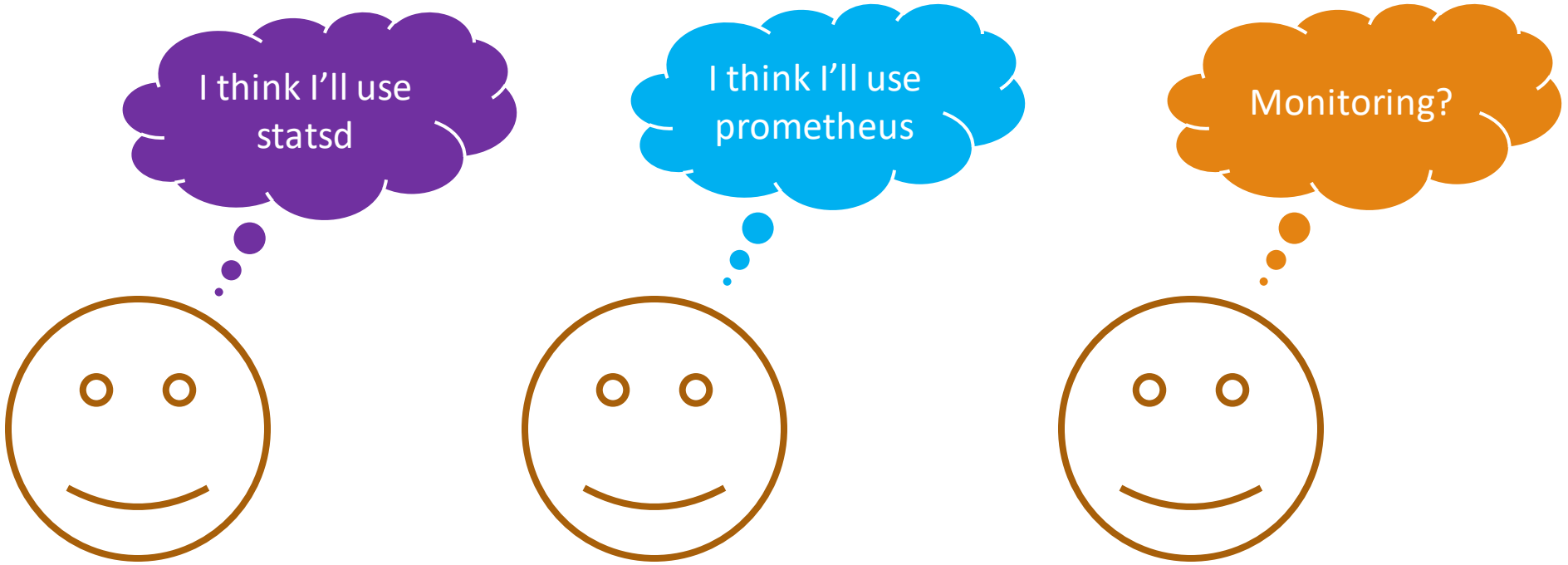
- e.g Logging, Monitoring, Security

Automatically enabled by ***deploying into the cluster***

Ensure consistency, concentration of skills & best practices

Microsoft

Cluster Services

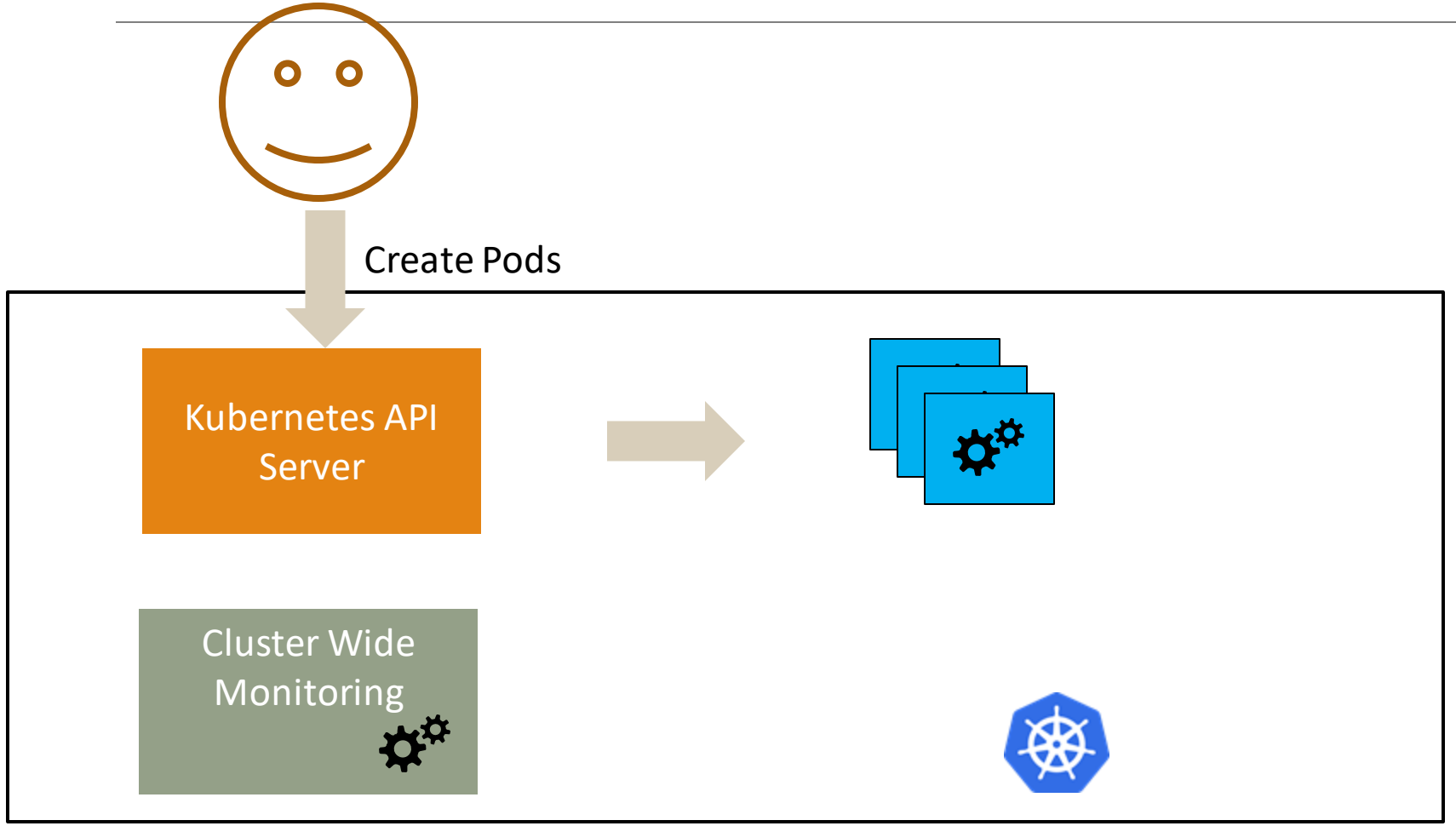


Microsoft

Cluster Services

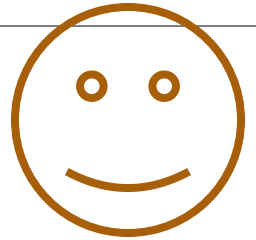
Microsoft

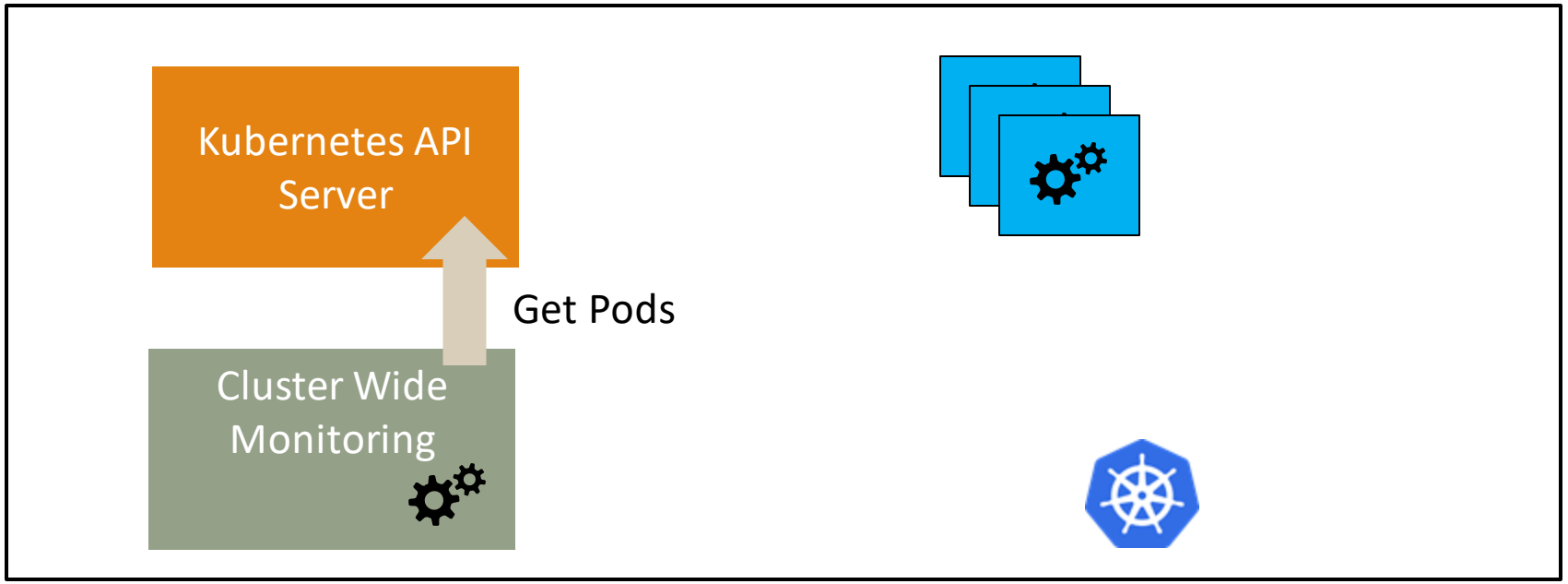
Cluster Services



Microsoft

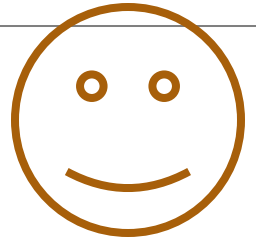
Cluster Services

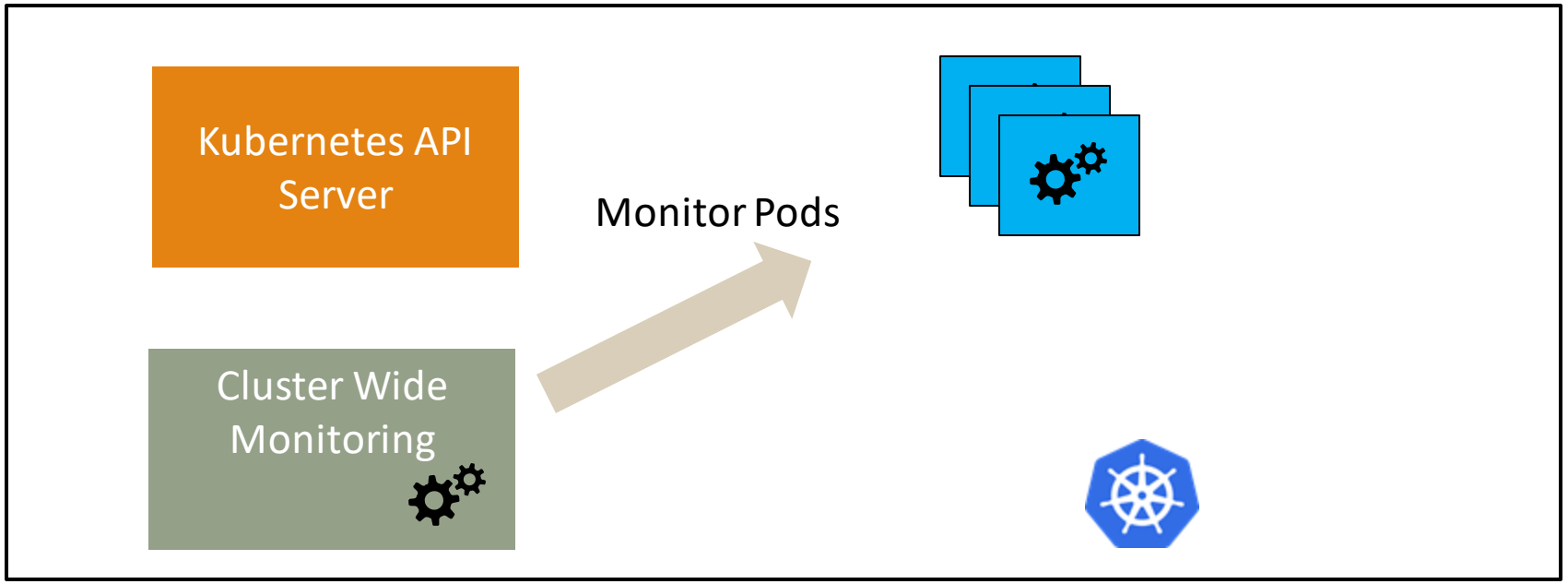




Microsoft

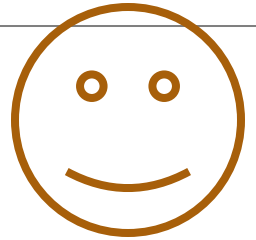
Cluster Services

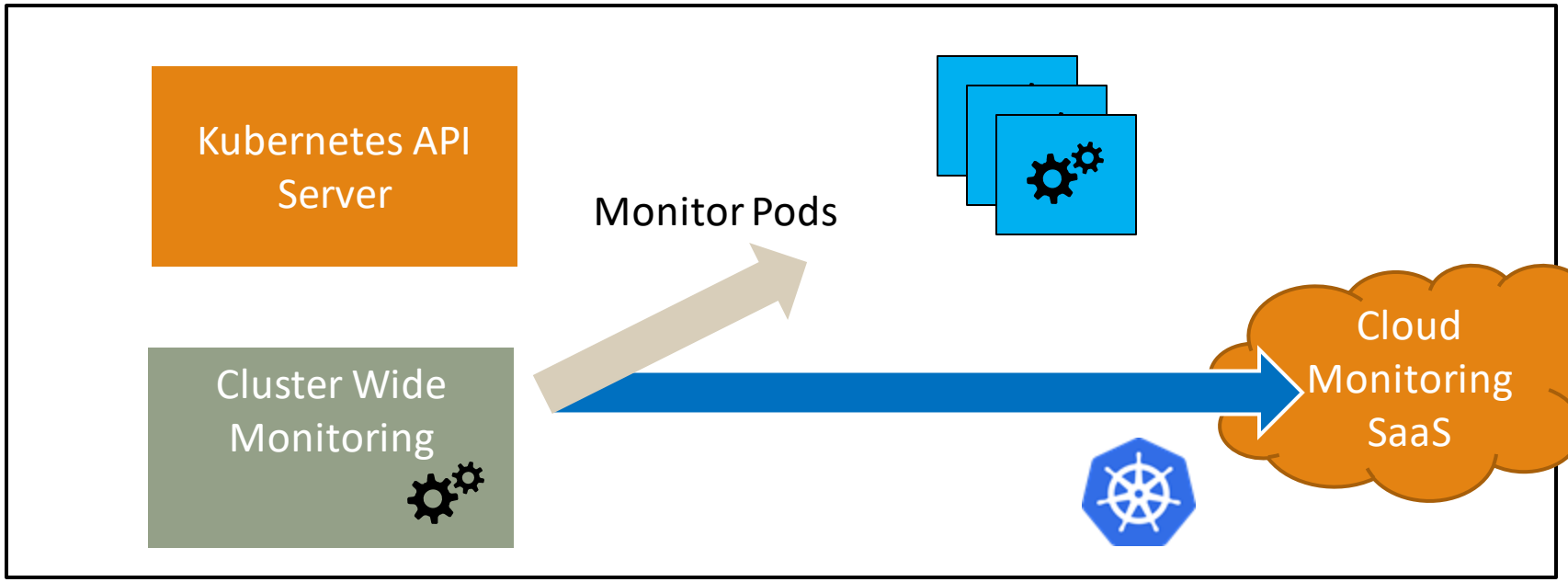




Microsoft

Cluster Services





Microsoft

Questions?