Design Patterns for Container-Based Distributed Systems

Brendan Burns

David Oppenheimer

Google Cloud Platform

Usenix HotCloud 2016

A Brief History

1985 - “The C++ Programming Language”

1994 - “Design Patterns: Elements of Reusable Object-Oriented Software”

1998 - J2EE

2004 - Map/Reduce

2006 - Borg

2013 - Docker

2014 - Kubernetes

Why did people develop patterns?

“Good fences do good neighbors make”

- Robert Frost

Object

Interface

Patterns provide:

A common language… improves collaboration

A common workspace… enables best-of-breed solutions

A common implementation.. because we develop libraries and frameworks

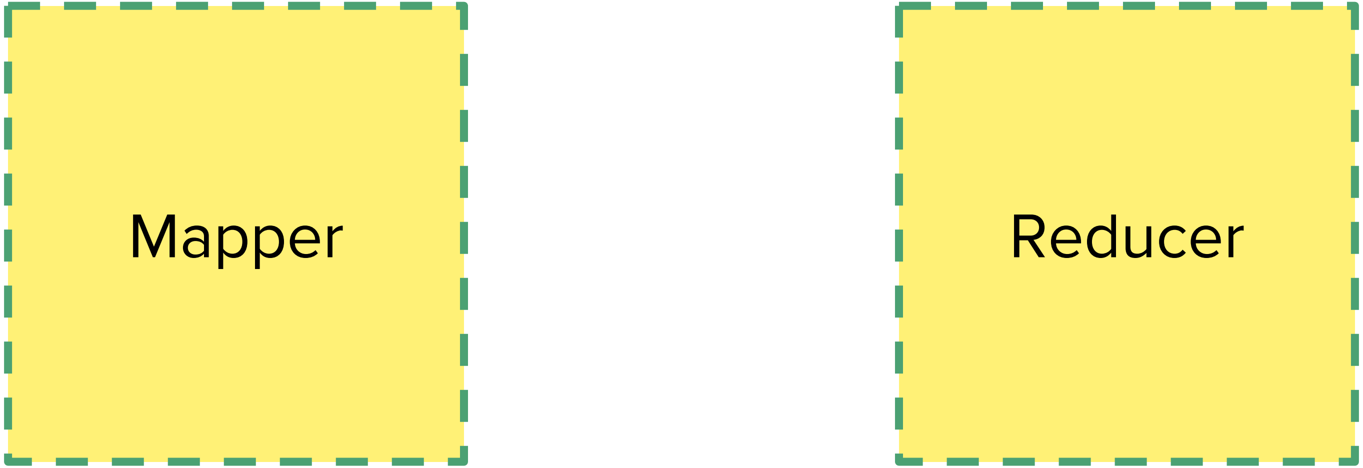
A common foundation… on which to build higher level systems

A simpler experience… makes good practices easier to adopt

A concrete example...

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean, Sanjay Ghemawat

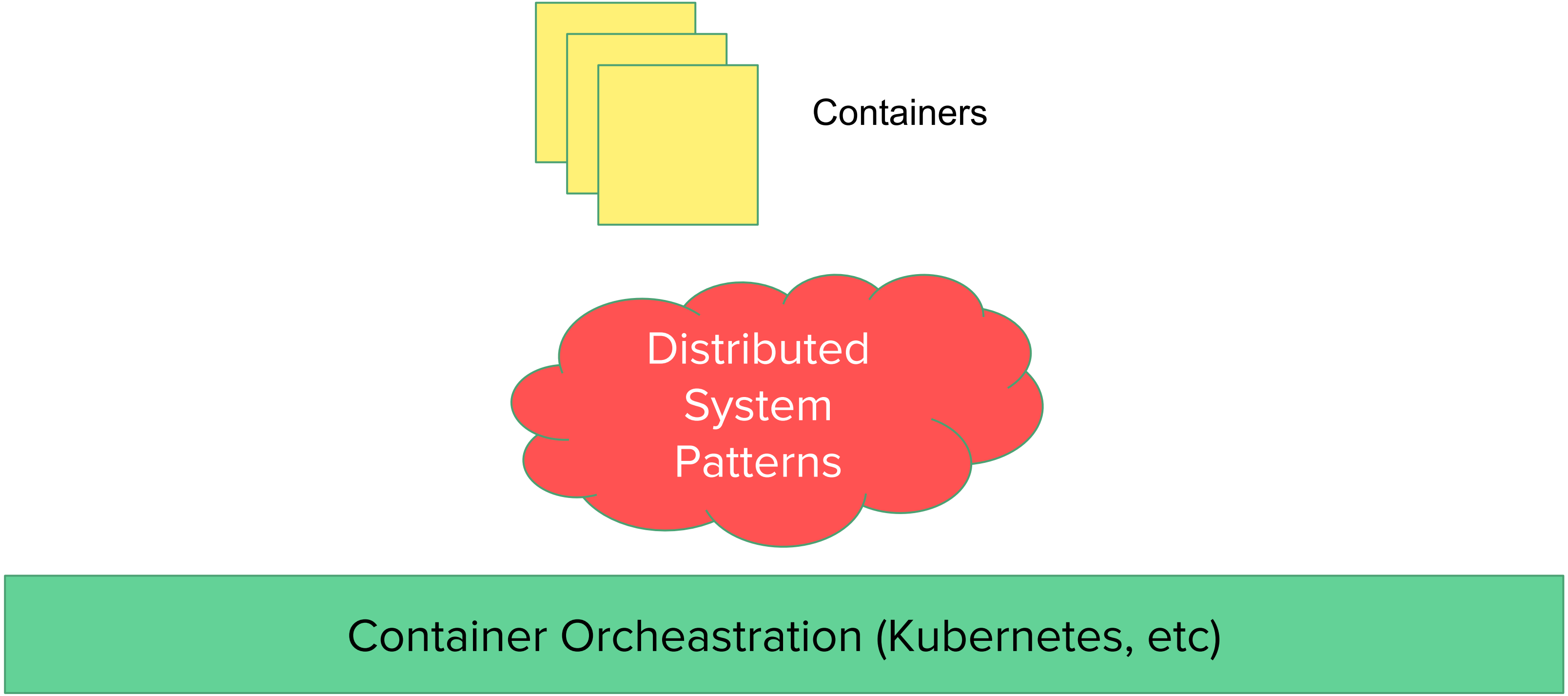


Mapper

Reducer

But it’s still only one pattern, and one(ish) language

But it’s still only one pattern, and one(ish) language


Containers
Distributed System Patterns
Container Orcheastration (Kubernetes, etc)

Container Patterns for Distributed Systems

Single-node multi-container patterns

- Sidecar
- Ambasador
- Adapter

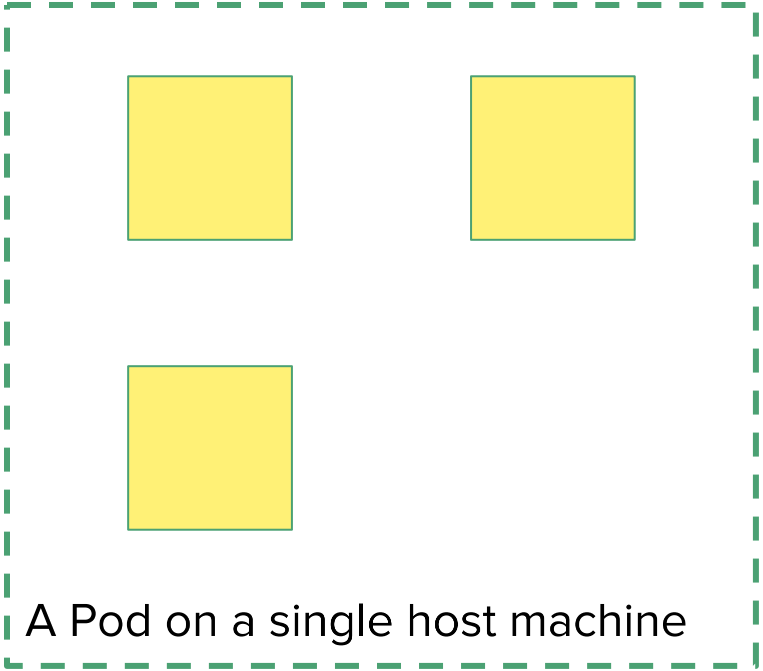
Distributed container patterns

- Leader election
- Work queue
- Scatter/gather
- ...

Single-node multi-container patterns

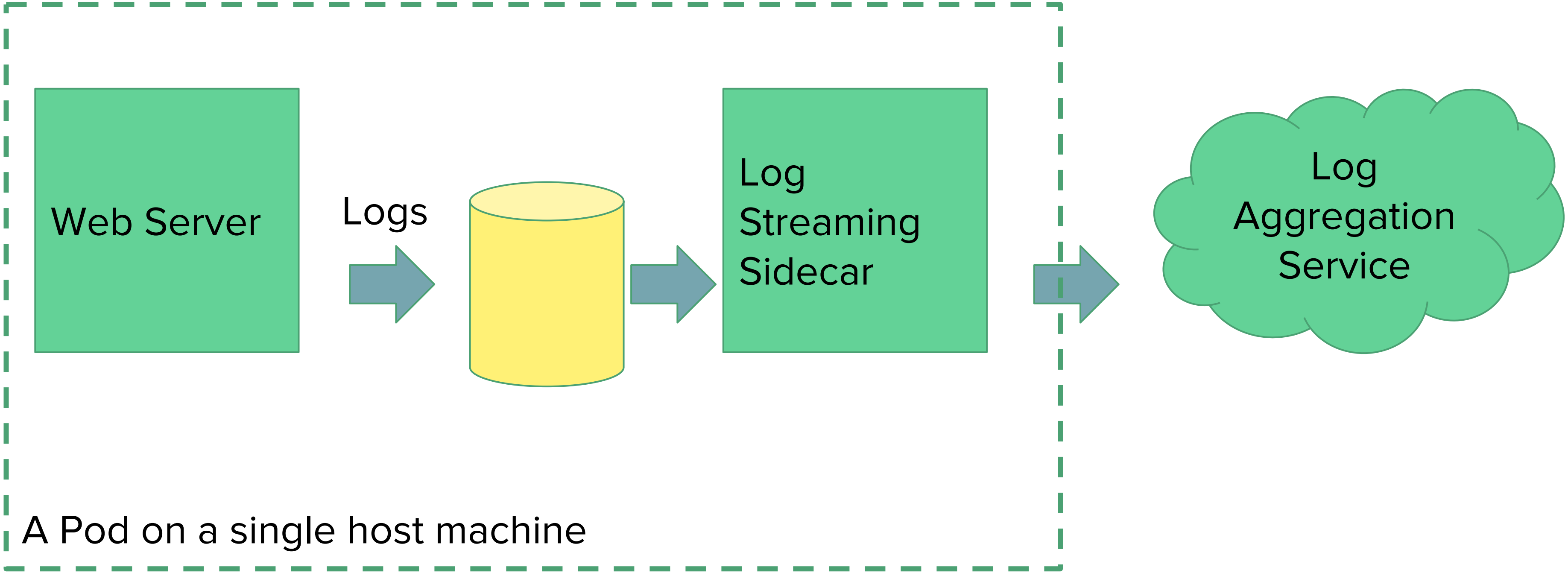
Pod

- Multiple containers
- Shared namespaces
- Shared IP address
- Atomic unit of scheduling



Sidecar

Sidecars extend and enhance



Web Server

Logs

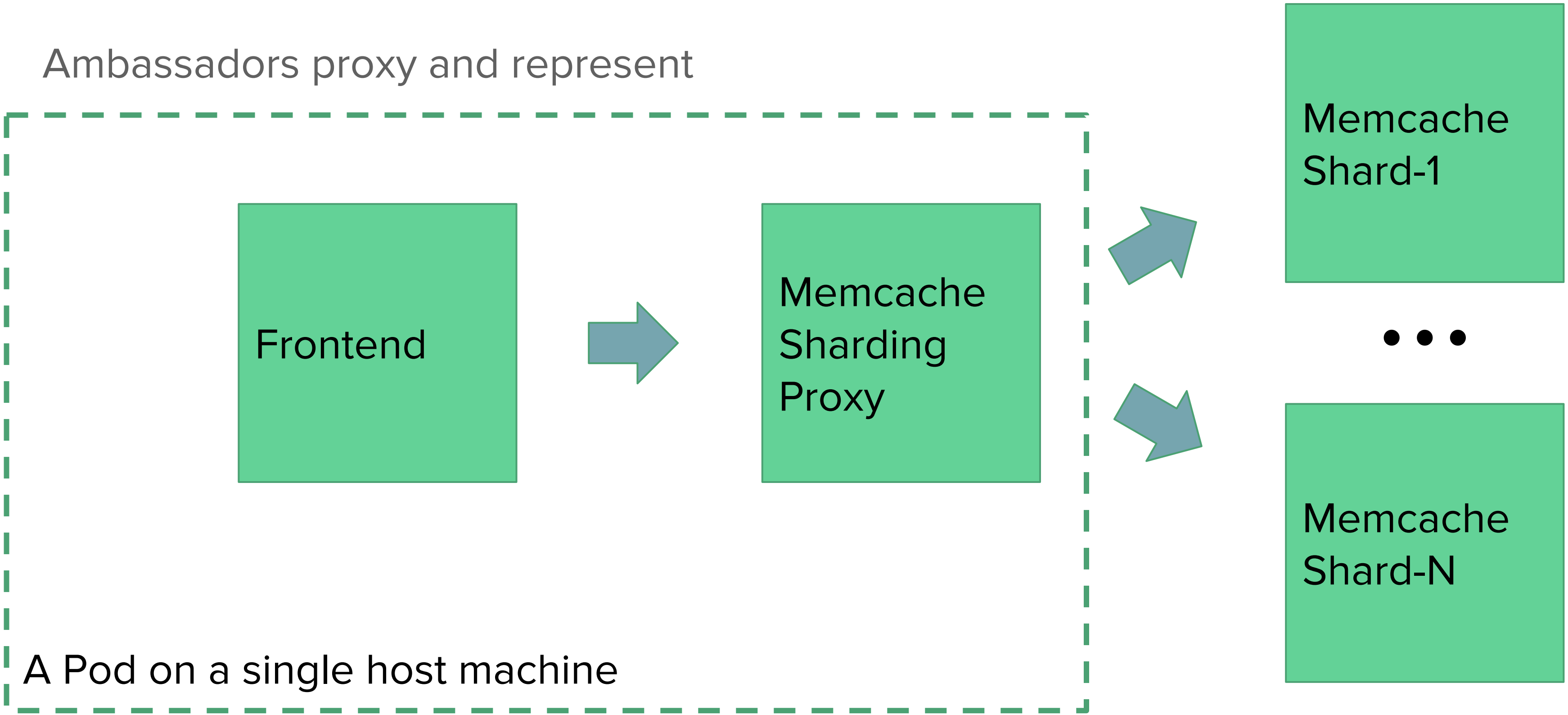
Log Streaming Sidecar

Log Aggregation Service

A Pod on a single host machine

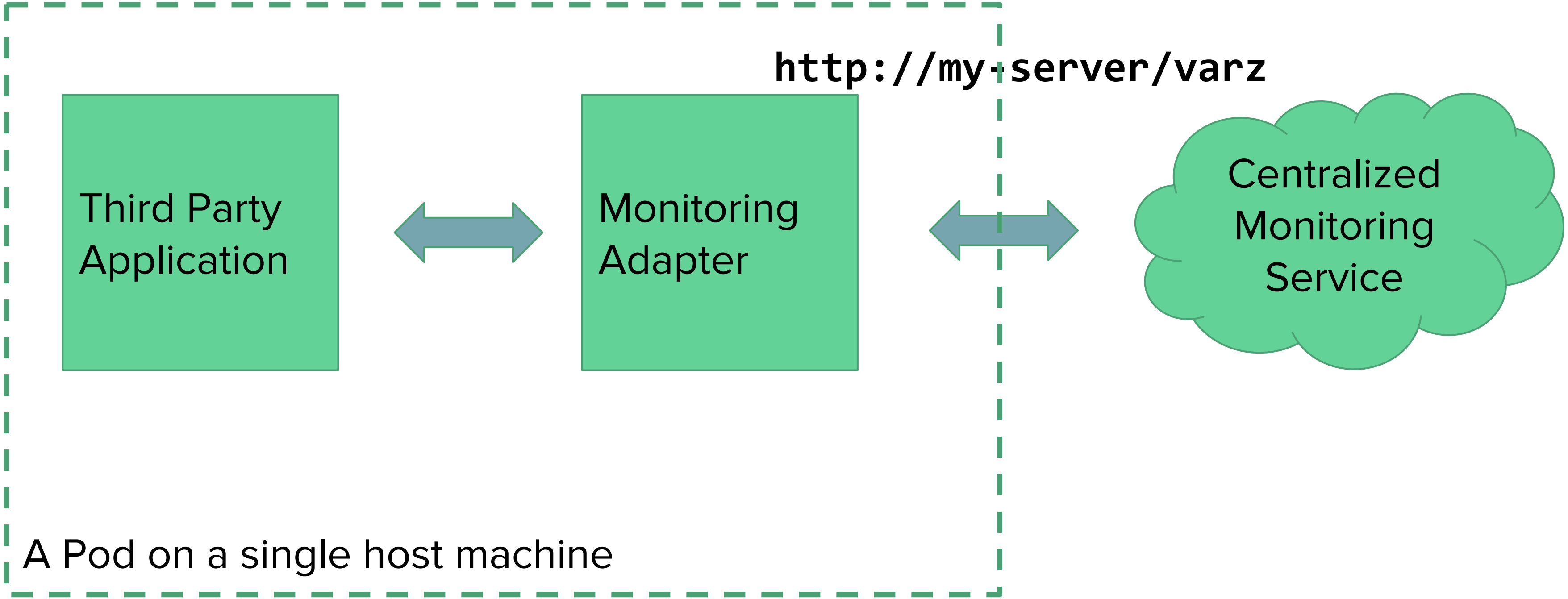
Ambassador

Ambassadors proxy and represent



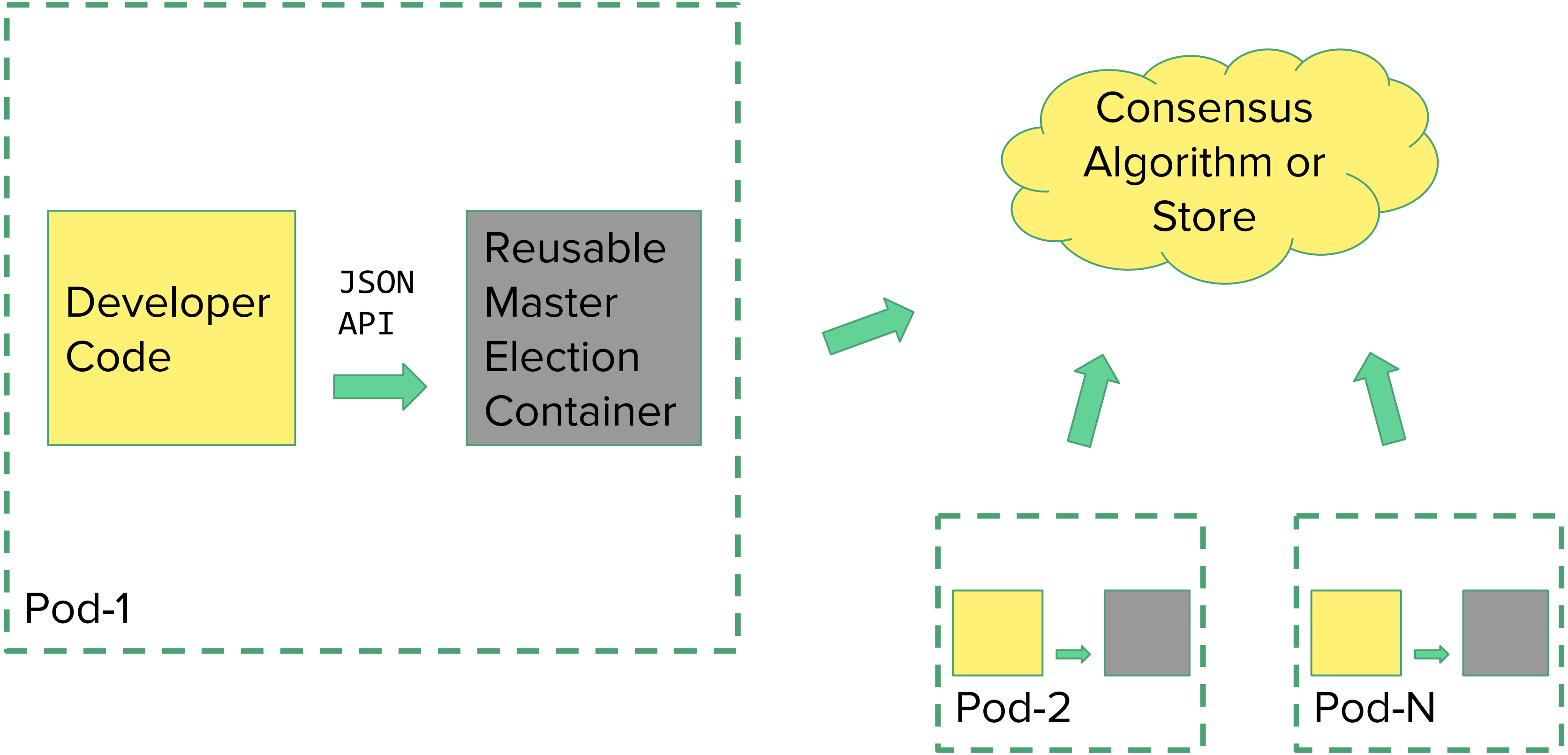
Adapters

Adapters normalize and present

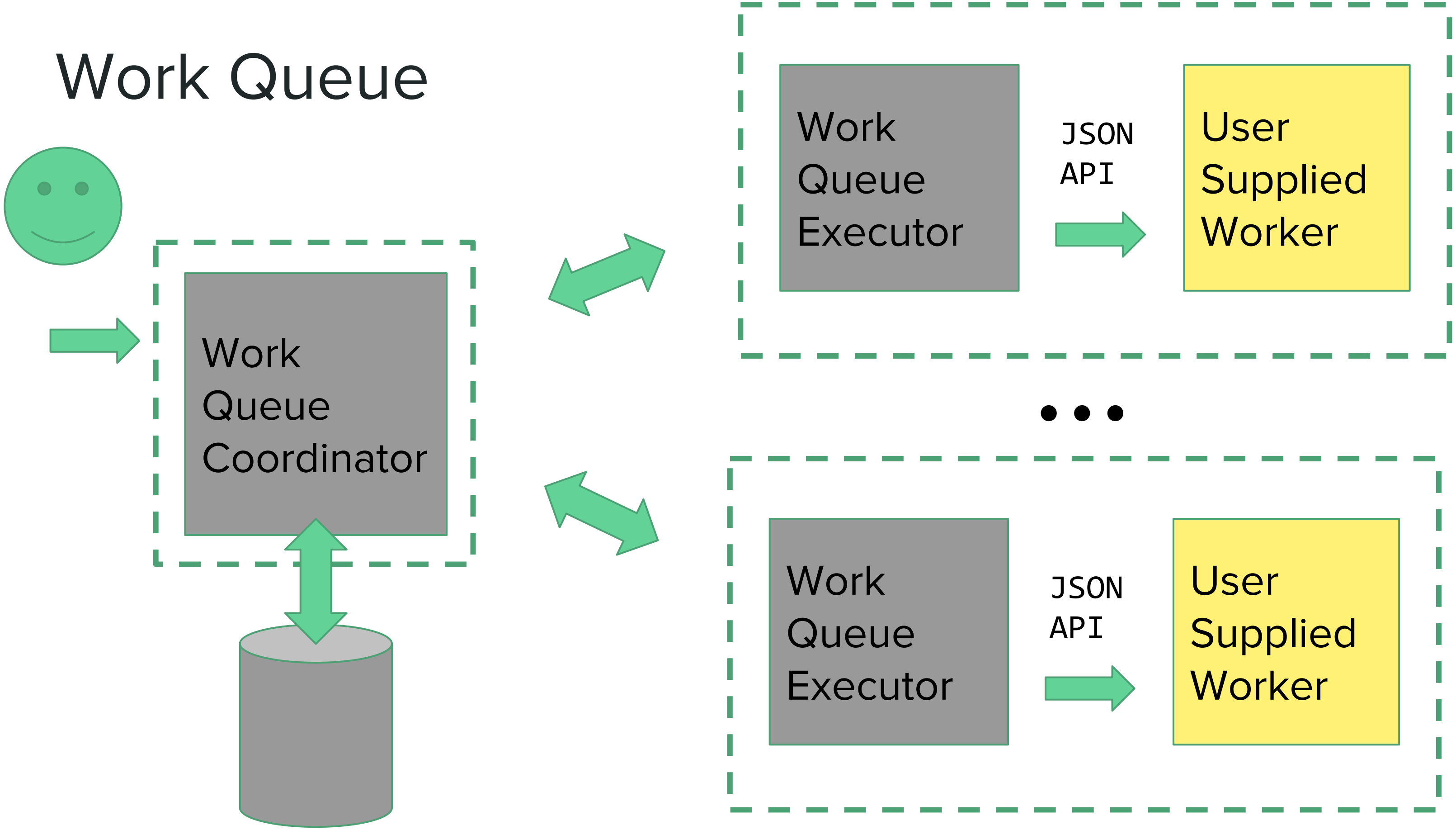


Distributed System Patterns

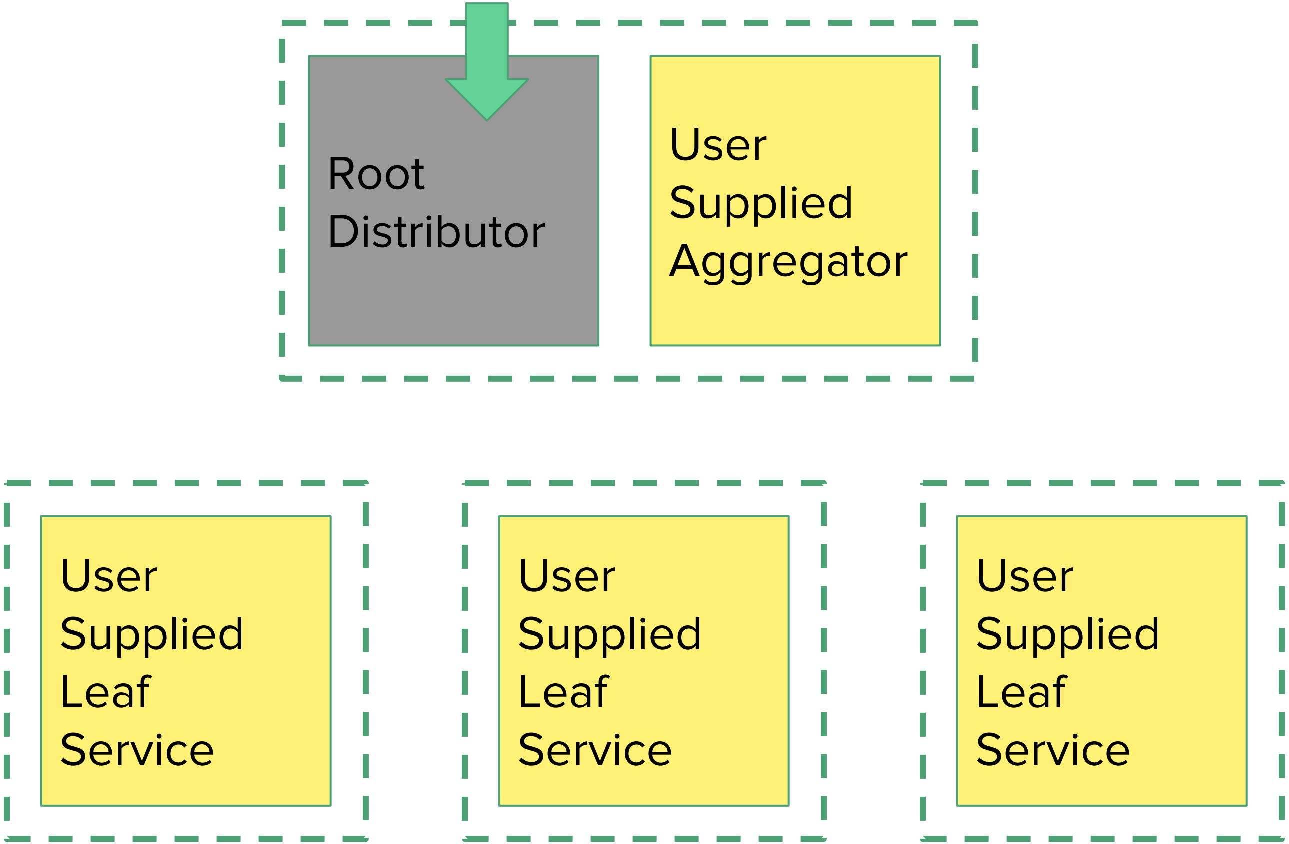
Leader Election


Developer Code
JSON API
Reusable Master Election Container
Pod-1
Consensus Algorithm or Store
Pod-2
Pod-N

Work Queue


Work Queue Coordinator
Work Queue Executor
JSON API
User Supplied Worker
Work Queue Executor
JSON API
User Supplied Worker
• • •

Scatter/Gather (aka Fan out/Fan in)



Root Distributor

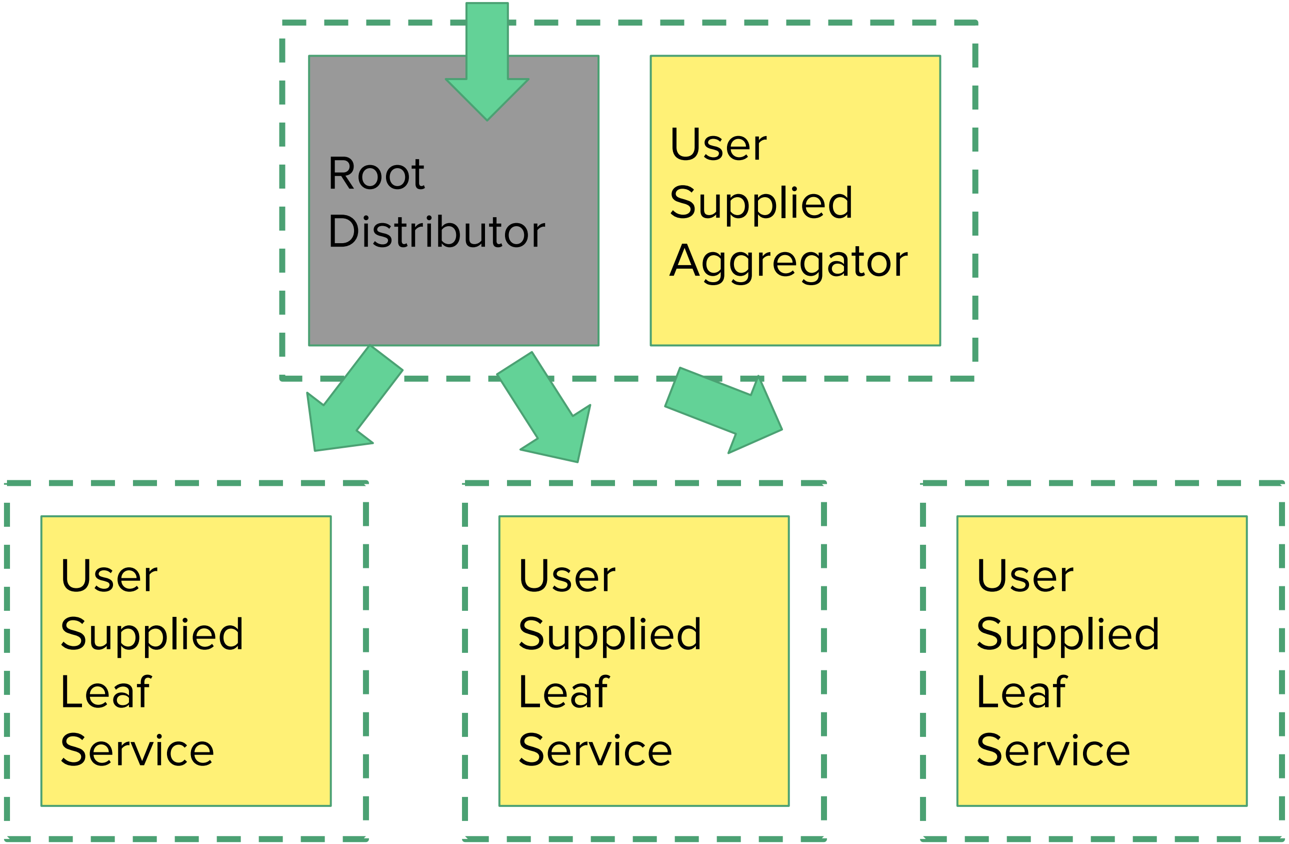
User Supplied Aggregator

User Supplied Leaf Service

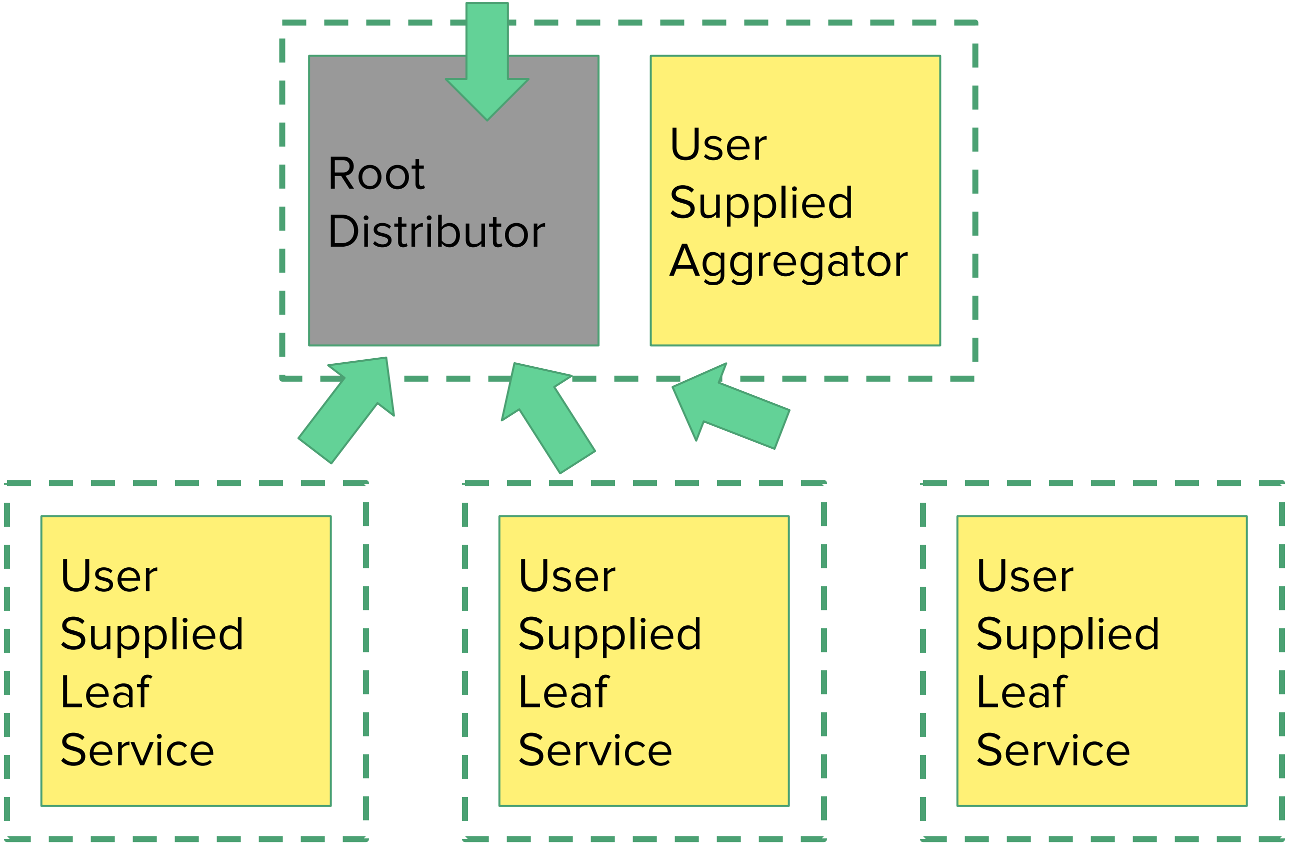
User Supplied Leaf Service

User Supplied Leaf Service

Scatter/Gather (aka Fan out/Fan in)


Root Distributor
User Supplied Aggregator
User Supplied Leaf Service
User Supplied Leaf Service
User Supplied Leaf Service

Scatter/Gather (aka Fan out/Fan in)



Root Distributor

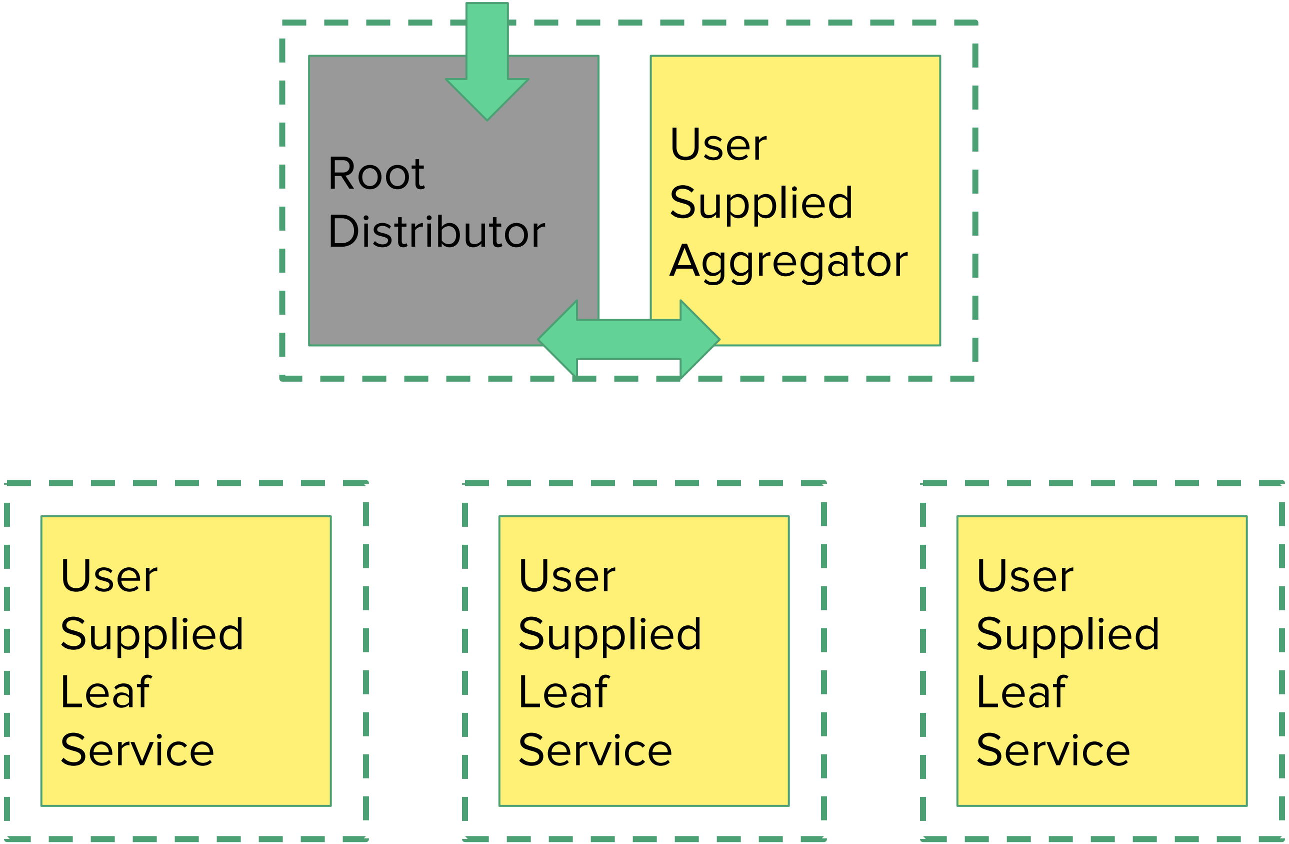
User Supplied Aggregator

User Supplied Leaf Service

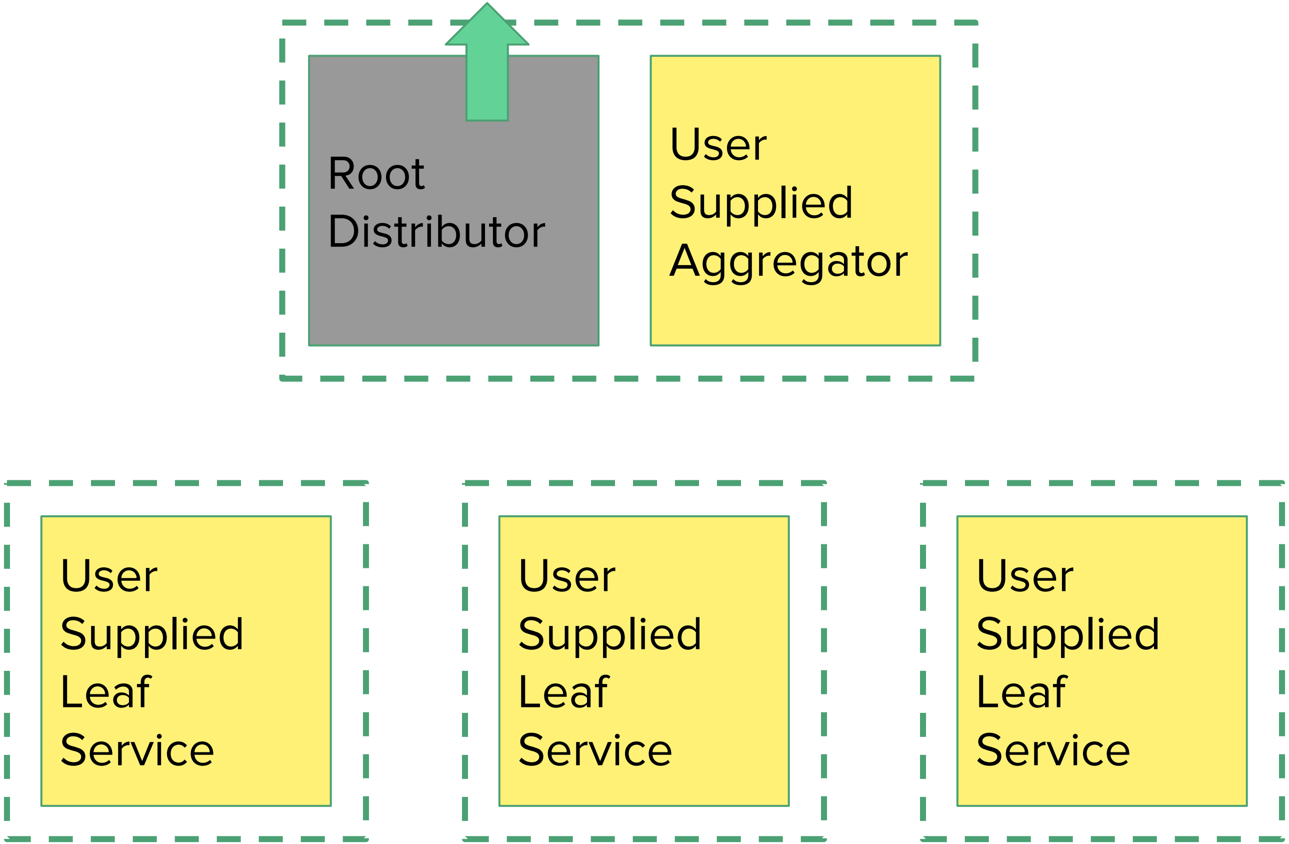
User Supplied Leaf Service

User Supplied Leaf Service

Scatter/Gather (aka Fan out/Fan in)


Root Distributor
User Supplied Aggregator
User Supplied Leaf Service
User Supplied Leaf Service
User Supplied Leaf Service

Scatter/Gather (aka Fan out/Fan in)



Root Distributor

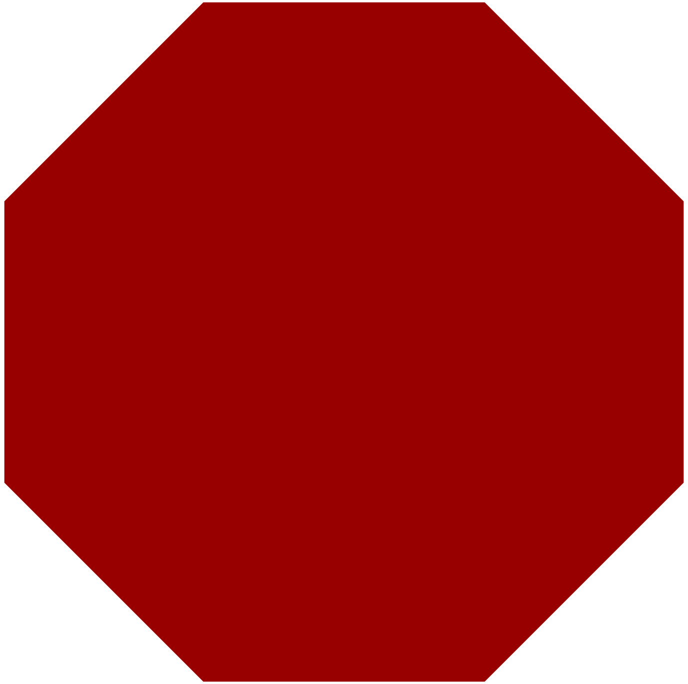
User Supplied Aggregator

User Supplied Leaf Service

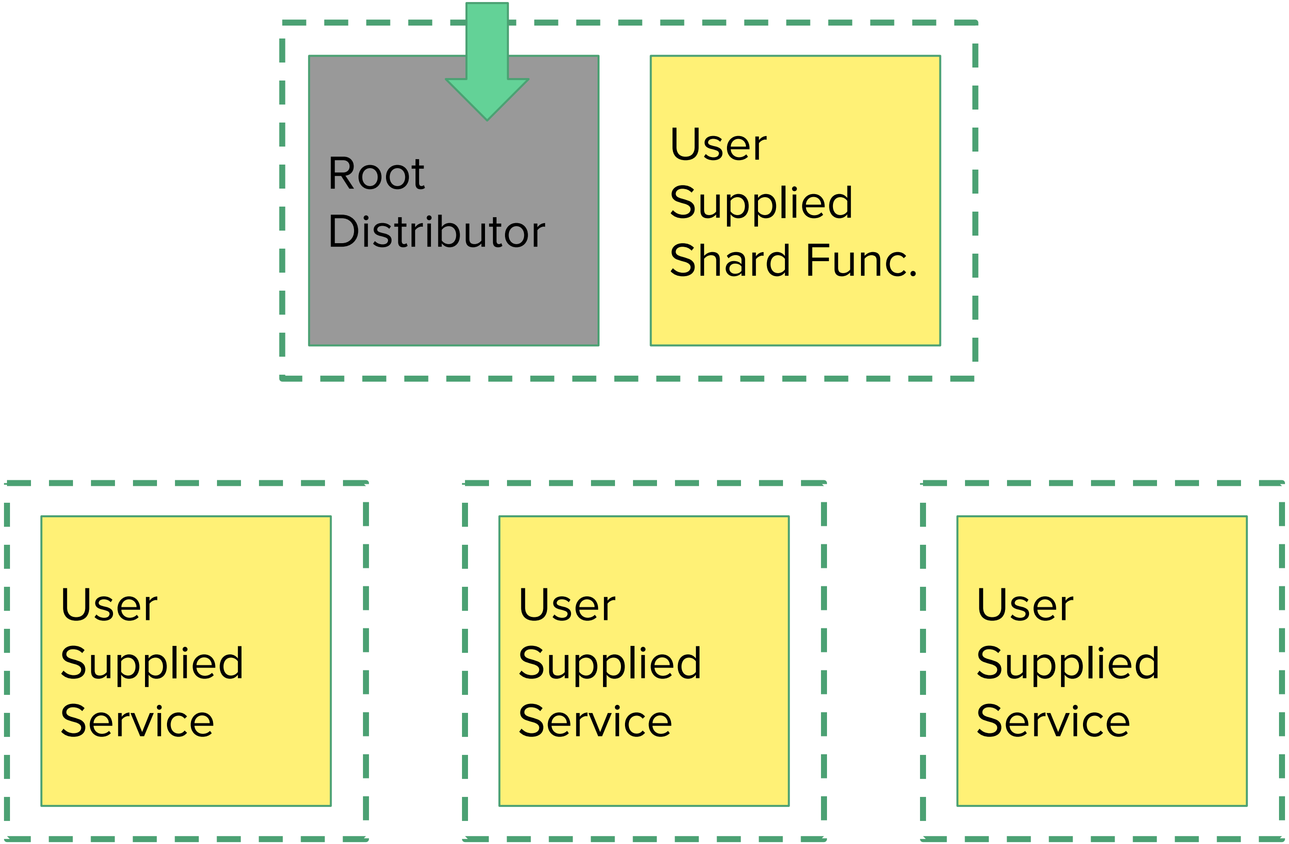
User Supplied Leaf Service

User Supplied Leaf Service

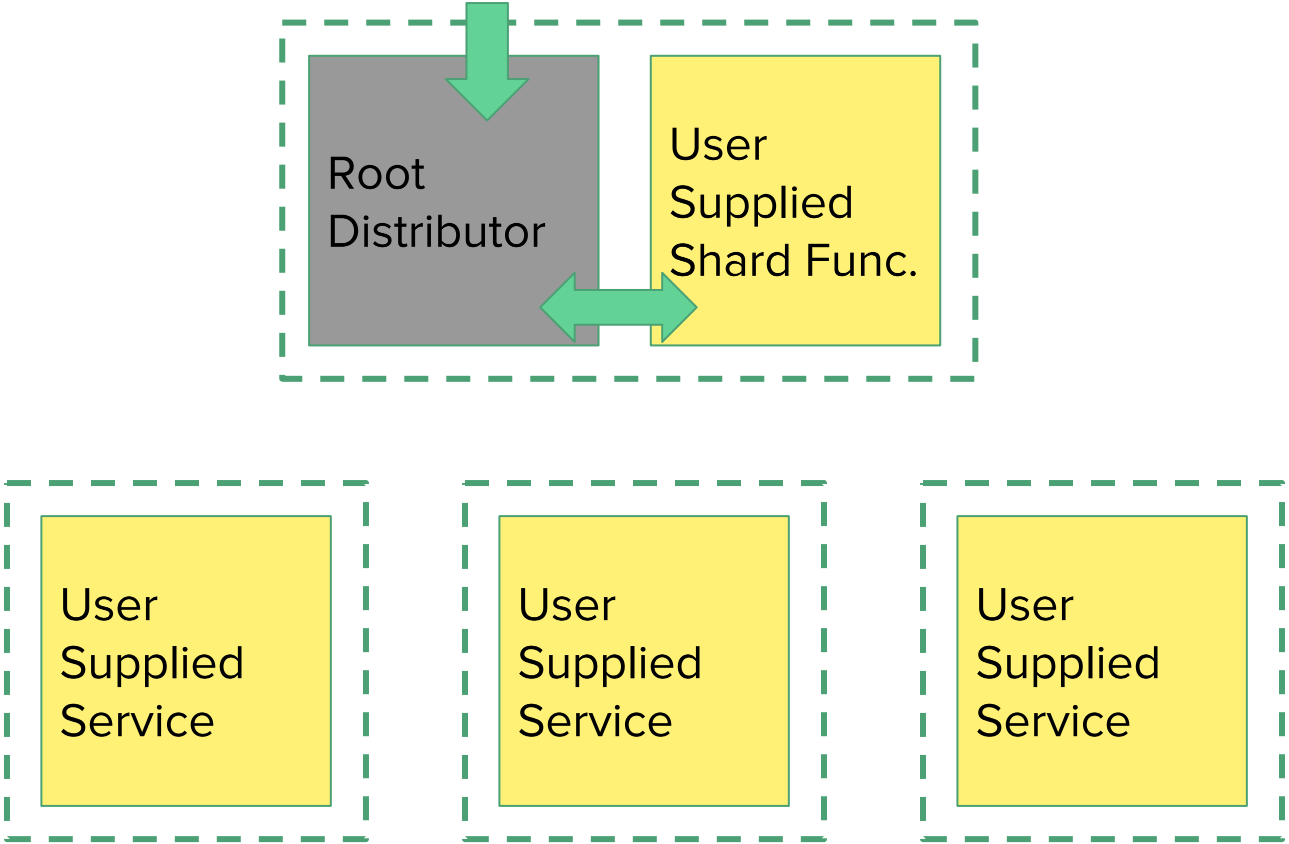
Thanks!



Sharded Service



Sharded Service



Root Distributor

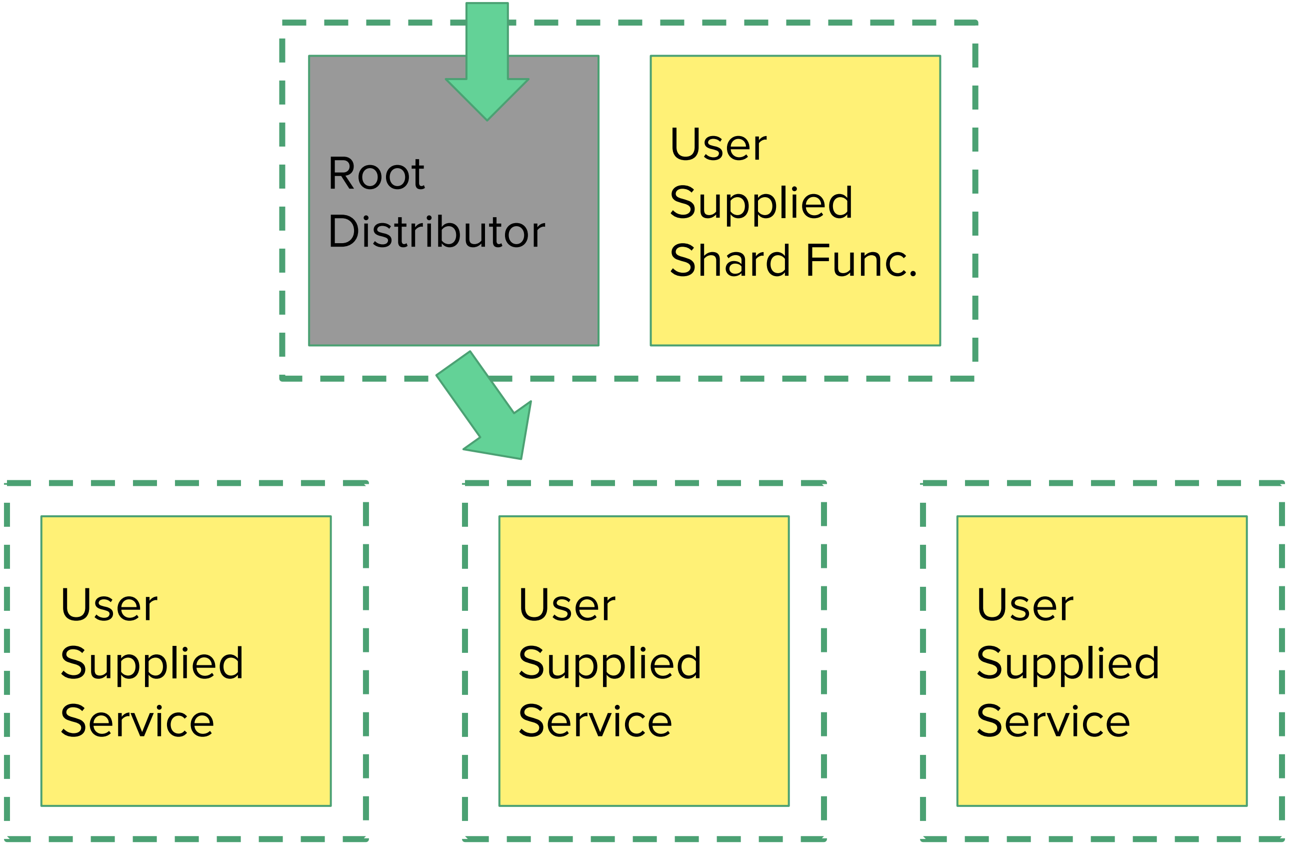
User Supplied Shard Func.

User Supplied Service

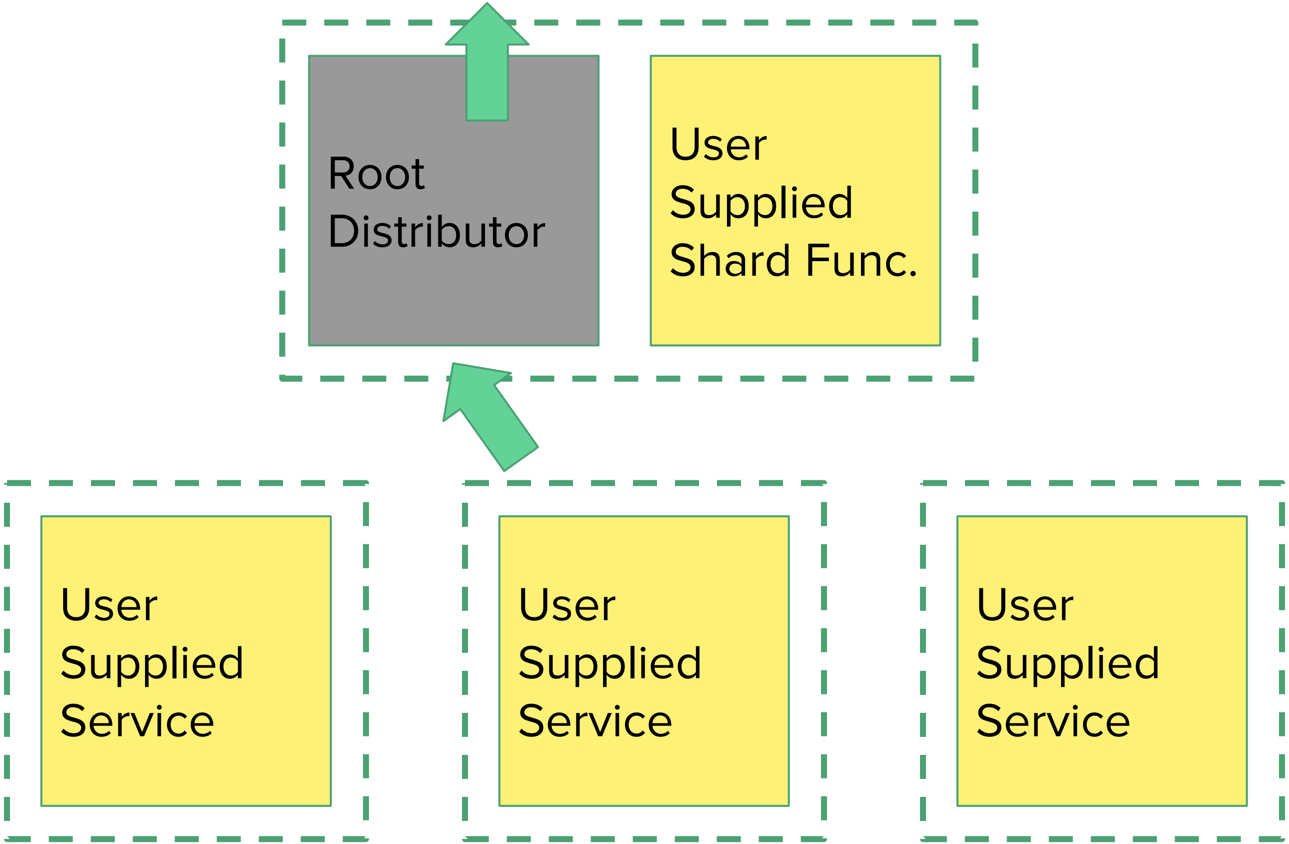
User Supplied Service

User Supplied Service

Sharded Service



Sharded Service



Root Distributor

User Supplied Shard Func.

User Supplied Service

User Supplied Service

User Supplied Service