This job is too hard
A history lesson

Development in the 1940s and 1950s
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

Container Group (aka Pod)

Container #1

Container #2

Container #3
Single Node Patterns: Sidecar

Sidecars augment and extend

Application Container

Sidecar Container

Container Group (aka "Pod")
Single Node Patterns: Sidecar

Sidecars augment and extend

- Web Server
- Sidecar Container
- Container Group (aka "Pod")
- Cloud Storage
Single Node Patterns: Sidecar

Sidecars augment and extend

- Web Server
- Sidecar Container
- Container Group (aka "Pod")
- Cloud Storage
Single Node Patterns: Ambassador

Ambassadors translate and represent

Container Group (aka "Pod")
Single Node Patterns: Ambassador

Ambassadors translate and represent

NodeJS Web App → Twem Proxy

Container Group (aka "Pod")

Memcache Shard #1
Memcache Shard #2
Memcache Shard #3
Single Node Patterns: Adapter

Adapters Standardize and Normalize

Application Container
Adapter
External Interface

Container Group (aka "Pod")
Single Node Patterns: Adapter

Adapters Standardize and Normalize

Java Application ➔ JMX ➔ Prometheus Java Exporter ➔ Prometheus Scraper

Container Group (aka "Pod")
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

Service Load Balancer

Replica 1  Replica 2  Replica 3  Replica 4
Multi-Node Patterns: Replicated

Reliable, redundant serving
Safe, zero-downtime rollouts

Service -> Load Balancer
Deployment -> Replica 1 Replica 2 Replica 3 Replica 4
Multi-Node Patterns: Replicated

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

https://metaparticle.io/tutorials
Multi-Node Patterns: Sharded

Sharding increases cache hit rates
Enables larger data-stores
Changes failure patterns
Multi-Node Patterns: Sharded

- Sharding increases cache hit rates
- Enables larger data-stores
- Changes failure patterns

Diagram:
- Shard Load Balancer
- Shard Router
- Shard #1
- Shard #2
- Shard #3
Multi-Node Patterns: Sharded

Service -> Shard Load Balancer

Deployment -> Shard Router Shard Router Shard Router Shard Router

StatefulSet -> Shard #1 Shard #2 Shard #3
Multi-Node Patterns: Sharded

Service -> Shard Load Balancer
Replicated -> Shard Router
Deployment -> Shard Router
StatefulSet -> Shard Router
Shard #1
Shard #2
Shard #3
Multi-Node Patterns: Sharded

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

[https://metaparticle.io/tutorials](https://metaparticle.io/tutorials)
Multi-Node: Replicated & Sharded

Shard Load Balancer

Shard Router  Shard Router  Shard Router  Shard Router

Nodes: Black  Orange  Brown
class ShardedReplicatedService : ShardedService {
    ReplicatedService[] replicatedShards;

    public ShardedReplicatedService() {
        ...
    }
    ...
}
Agenda

- Patterns
  - Single Node Patterns
  - Multi-Node patterns
- Tools
  - Cluster Daemons
  - Cluster Agents
  - Intent-based APIs
Tools: The cluster as the new node

Cluster Daemons

Cluster Services

Intent-based APIs

Container API: Unified Compute Substrate
Cluster Daemons

Container API: Unified Compute Substrate

Distributed systems are become microservices
Cluster Daemons

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

Distributed systems are becoming microservices

Cert Manager Daemon

HTTP Service (Ingress)
Cluster Daemons

Distributed systems are becoming microservices

Cert Manager Daemon → HTTP Service (Ingress)

`kubernetes.io/tls-acme: "true"`
Cluster Daemons

Distributed systems are becoming microservices

Cert Manager Daemon ➔ HTTP Service (Ingress) ➔ Certificate

`kubernetes.io/tls-acme: "true"`
Intent-based APIs

State your intent, not how to build it.
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
Intent-based APIs

I SHOULD RUN A LOAD TEST
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
Intent-based APIs

Load Test Extension API -> Create 3rd party LoadTest API -> Kubernetes API Server

/apis/loadtests/...
Intent-based APIs

Load Test Extension API

Watch /apis/loadtests/...

Kubernetes API Server

/apis/loadtests/...
Intent-based APIs

`Load Test Extension API` → `Kubernetes API Server` → `Create /apis/loadtests/my-cool-loadtest`

Watch `/apis/loadtests/...`

`/apis/loadtests/...`

Microsoft
Intent-based APIs

- Load Test Extension API
- Create LoadTest Pods
- Kubernetes API Server
Intent-based APIs

Load Test Extension API → Create LoadTest Pods → Kubernetes API Server → my-cool-service
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
Cluster Services

I think I’ll use statsd

I think I’ll use prometheus

Monitoring?
Cluster Services

I think I’ll quit.
Cluster Services

Create Pods

Kubernetes API Server

Cluster Wide Monitoring

Microsoft
Cluster Services

Kubernetes API Server

Cluster Wide Monitoring

Get Pods
Cluster Services

- Kubernetes API Server
- Cluster Wide Monitoring
- Monitor Pods
Questions?