Kubernetes in the Datacenter

Squarespace’s Journey Towards Self-Service Infrastructure

Kevin Lynch
klynch@squarespace.com
2013: <50 engineers

- “Whatever works”
- Build product
- Grow fast
2014: ~75 engineers

- “Whatever works”
- Too much firefighting
- Not enough new features
- Microservices FTW!
2016: 100+ engineers

- Scalable + Reliable
- Developers can move faster
- Squarespace can move faster
Traditional Provisioning Process

**Manual**
- Find Resources (CPU, RAM, Disk)
- Find IP
- Configure VLAN
- Configure Firewall
- Update Ansible Inventory

**DNS Updates**

**PXE Boot**

**Install OS**

**Configure OS**

**Install App!**

**Install App Dependencies**

**Configure Monitoring**
Containerization & Kubernetes Orchestration

Static infrastructure and microservices do not mix!

- Difficult to find resources
- Slow to provision and scale
- Shoehorning “Cattle” into “Pets” mentality
- System was too complex for new engineers
2017: 200+ engineers

- Self-Service Infrastructure
- Operations can move faster
- Squarespace can move faster
kubectl apply -f app.yaml
Self-Service Compute

- Java Spring Boot
- FluentD
  - Logging
- Consul
  - Service Discovery
  - K/V

Pod

Java Service

fluentd

consul

resources:
  requests:
    cpu: 2
    memory: 4Gi
  limits:
    cpu: 2
    memory: 4Gi
Self-Service Compute

- CGroup assigned to each pod
- Completely Fair Scheduler (CFS)
  - Schedules a task based on CPU Shares
  - Throttles a task once it hits CPU Quota
- OOM Killed when memory limit exceeded

```
resources:
  requests:
    cpu: 2
    memory: 4Gi
  limits:
    cpu: 2
    memory: 4Gi
```
● Shares = CPU Request * 1024
● Total Kubernetes Shares = # Cores * 1024
● Quota = CPU Limit * 100ms
● Period = 100ms

resources:
  requests:
    cpu: 2
    memory: 4Gi
  limits:
    cpu: 2
    memory: 4Gi
- Shares = 2048
- Total Kubernetes Shares = 65536
- Quota = 200ms
- Period = 100ms

```yaml
resources:
  requests:
    cpu: 2
    memory: 4Gi
  limits:
    cpu: 2
    memory: 4Gi
```
GC Threads were using up most of the CPU Quota
- 64 GC Threads
- 128 Jetty Threads
- 64 ForkJoin Threads
Libraries call `Runtime.getRuntime().availableProcessors()`
- Jetty
- ForkJoinPool
- GC Threads
- ???

JVM detects cores via `sysconf(_SC_NPROCESSORS_ONLN)`

CGroups does not limit `_SC_NPROCESSORS_ONLN`
**Self-Service Compute**

- Provide a base Java container to calculate resources
- Detect maximum # of “cores” assigned
  - `/sys/fs/cgroup/cpu/cpu.cfs_quota_us` divided by `/sys/fs/cgroup/cpu/cpu.cfs_period_us`
- Automatically tune the JVM
  - `-XX:ParallelGCThreads=${core_limit}`
  - `-XX:ConcGCThreads=${core_limit}`
  - `-Djava.util.concurrent.ForkJoinPool.common.parallelism=${core_limit}`
● Use Linux **LD_PRELOAD** to override **availableProcessors()**

```c
#include <stdlib.h>
#include <unistd.h>

int JVM_ActiveProcessorCount(void) {
    char* val = getenv("CONTAINER_CORE_LIMIT");
    return val != NULL ? atoi(val) : sysconf(_SC_NPROCESSORS_ONLN);
}
```

- Java Spring Boot
- Netflix Ribbon
  - Automatic Retries
  - Client Side Load Balancing
- Netflix Hystrix
  - Circuit Breaking
- Consul
  - Service Discovery
Self-Service Networking

Pod
Service
fluentd
consul

VM
Service
fluentd
consul
● Kubernetes CNI (Container Network Interface) is pluggable
● Different plugins for different network topologies
  ○ Flannel
  ○ Calico
  ○ Weave
  ○ Kubenet
  ○ VXLAN
- Project Calico
- No network overlay required!
  - No MTU issues
  - No performance impact
  - No ingress/egress issues
- Communicates directly with existing Layer 3 network
- BGP Peering with Top of Rack switch
Spine and Leaf Layer 3 Clos Topology

- Simple to understand
- Easy to scale
- Predictable and consistent latency (hops = 2)
- Anycast support
- All work is performed at the leaf/ToR switch
- Each leaf switch is separate Layer 3 domain
- Each leaf is a separate BGP domain (ASN)
- No Spanning Tree Protocol issues seen in L2 networks (convergence time, loops)
Spine and Leaf Layer 3 Clos Topology

- **Spine**
- **Leaf**

Network:
- 10.0.1/24
- 10.0.2/24
- 10.0.3/24
- 10.0.4/24
Spine and Leaf Layer 3 Clos Topology

Diagram showing a network topology with Spine and Leaf layers connected by Anycast points.
Spine and Leaf Layer 3 Clos Topology
Kubernetes Networking

- Kubernetes Masters
  - Calico-Node Agent
- Kubernetes Nodes
  - Pods
  - Calico-Node Agent
Kubernetes Networking

Kubernetes Masters
- Pod IPs
- Service IP Range
- API Server IP
- API Server
- Calico-Node Agent

Kubernetes Nodes
- Pod IPs
- Service IP Range
- Pods
- Calico-Node Agent
- Inefficient aggregations
- Loss of precision
- Ephemeral instances are expensive
- How much is too much?
  - Combinatoric Explosion
Self-Service Metrics

- Host based alerting
  - App and system tightly coupled
- Difficult to route alerts
  - Application?
  - System?
  - Hypervisor?
- Difficult to create alerts on SLAs
  - Confusing to create
  - Expensive queries
- Automatic discovery
- No loss of precision
- Arbitrary time intervals
- Stores tagged data
  - Service
  - Pod
  - Endpoint
- Efficient for ephemeral instances
Prometheus Operator

Prometheus Operator

Team A
Prometheus

Team B
Prometheus
Prometheus Operator

Team A
Prometheus

A1
A2

Team B
Prometheus

B1
B2
ALERT A1ErrorRate
  IF rate(responseCodes{service="A1", code="500"}[5m]) > 0
  FOR 1m
  LABELS {severity="critical", team="A"}
Self Service Storage

- Multiple Access Patterns
  - Block
  - Shared
  - Object
- Simple to scale
- Commodity hardware
- Automatic replication
- Independent of Kubernetes
Self Service Storage

- Default StorageClass
- RBD Provisioner Pod
- Block RBD
- Shared CephFS
- Object RADOS
- Storage
Self Service Storage

- PVC
- StatefulSet Pod
- Default StorageClass
- RBD Provisioner Pod
- ReadWriteOnce PV
- Block RBD
- Shared CephFS
- Object RADOS
- Storage
Self Service Storage

Default StorageClass

Block RBD

Shared CephFS

Object RADOS

Storage

PVC

RBD Provisioner Pod

ReadWriteOnce PV
Self Service Storage

- Shared StorageClass
  - Block RBD
  - Shared CephFS
  - Object RADOS

- PVC
- Service Pod
- Shared PV
- CephFS Provisioner Pod
- Storage
Self Service Storage

Service Pod

Block RBD

Shared CephFS

Object RADOS

Storage
- 20+ new services planned for Q1
- True “micro” services
  - Small
  - Experimental
- VM services migrated quickly
QUESTIONS?

Thank you!

Kevin Lynch
klynch@squarespace.com

squarespace.com/careers