SwarmKit: Docker’s Simple Model for Complex Orchestration

Stephen Day
Docker, Inc.
ContainerCon+LinuxCon EU
October 2016
Stephen Day

Docker, Inc.
github.com/stevvooe
@stevvooe
SwarmKit

A new framework by Docker for building orchestration systems.
What is orchestration?
Orchestration

A Docker-oriented History

- Orchestration Systems
  - Mostly use a service based model
  - Mostly wraps docker
  - Challenging to use

- Standalone Swarm, July 2015
  - Scale containers
  - Docker API “native”
  - No higher-level abstraction
Why orchestration?
Example

$ docker network create -d overlay backend
31ue4lvbj4m301i7ef3x8022t

$ docker service create -p 6379:6379 --network backend redis
bhk0gw6f0bgrbhmedwt51fu16

$ docker service scale serene_euler=3
serene_euler scaled to 3

$ docker service ls
ID             NAME         REPLICAS IMAGE  COMMAND
dj0jh3bnojtm   serene_euler 3/3       redis
Why services?
What is wrong with containers?
Nodes

- Arbitrary cluster resources
- Connected across a common network
- Topology control
- Cryptographic identity
Services

- Express *desired state* of the cluster
- Abstraction to control a set of containers
- Enumerates resources, network availability, placement
- Leave the details of runtime to container process
- Implement these services by distributing processes across a cluster
Networks

- Defines broadcast domains
- Services can *attach* to networks
- Routing mesh will route connections to active service process
Simple systems *can* exhibit complex behavior
Orchestration

A control system for your cluster

D = Desired State
O = Orchestrator
C = Cluster
S_t = State at time t
Δ = Operations to converge S to D

https://en.wikipedia.org/wiki/Control_theory
Convergence

A functional view

\[ f(D, S_{n-1}, C) \rightarrow S_n \mid \text{min}(S-D) \]

D = Desired State
O = Orchestrator
C = Cluster
\( S_t \) = State at time \( t \)
Observability and Controllability

The Problem

Low Observability  High Observability

Failure  Process State  User Input
Data Model Requirements

- Represent difference in cluster state
- Maximize Observability
- Support Convergence
- Do this while being Extensible and Reliable
Show me your data structures and I’ll show you your orchestration system
Declarative
Declarative

$ docker network create -d overlay backend
31ue4lvbj4m301i7ef3x8022t

$ docker service create --network backend redis
bhk0gw6f0bgrbhmedwt51fu16
Reconciliation
Spec → Object
Task Model
Atomic Scheduling Unit of SwarmKit

Orchestrator

Object
Current State

Spec
Desired State

Task_0  Task_1  Task_n

Scheduler
Consistency
Versioned Updates
Consistency

service := getCurrentService()

spec := service.Spec

spec.Image = "my.serv/myimage:mytag"

update(spec, service.Version)
Only one component of the system can write to a field
Task State

Manager

New → Allocated → Assigned

Pre-Run

Preparing → Ready → Starting

Worker

Running

Terminal States

Rejected → Complete → Shutdown → Failed
Extensible
Task Model

Runtime

Prepare: setup resources
Start: start the task
Wait: wait until task exits
Shutdown: stop task, cleanly
Terminate: kill the task, forcefully
Update: update task metadata, without interruption
Remove: remove resources used by task
Reliable
SwarmKit doesn’t Quit
Architecture

Data Structures
Service Spec

Protobuf Example

```protobuf
message ServiceSpec {
  // Task defines the task template this service will spawn.
  TaskSpec task = 2 [(gogoproto.nullable) = false];

  // UpdateConfig controls the rate and policy of updates.
  UpdateConfig update = 6;

  // Service endpoint specifies the user provided configuration
  // to properly discover and load balance a service.
  EndpointSpec endpoint = 8;
}
```
Service Object

Protobuf Example

```protobuf
message Service {
    ServiceSpec spec = 3;

    // Runtime state of service endpoint. This may be different
    // from the spec version because the user may not have entered
    // the optional fields like node_port or virtual_ip and it
    // could be auto allocated by the system.
    Endpoint endpoint = 4;

    // UpdateStatus contains the status of an update, if one is in
    // progress.
    UpdateStatus update_status = 5;
}
```
Task

Protobuf Example

// Task specifies the parameters for implementing a Spec. A task is effectively
// immutable and idempotent. Once it is dispatched to a node, it will not be
// dispatched to another node.

message Task {
    TaskSpec spec = 3;
    string service_id = 4;
    uint64 slot = 5;
    string node_id = 6;
    TaskStatus status = 9;
    TaskState desired_state = 10;
    repeated NetworkAttachment networks = 11;
    Endpoint endpoint = 12;
    Driver log_driver = 13;
}
Blue Green Deployments

Applied

**Sillyproxy**
- Uses rolling updates to set proxy backends
- Desired state is encoded in environment variables
- Rolling updates can control traffic between backends
Distributed Period Scheduler

Applied

From a GitHub comment
- Scheduling criteria set via environment variables
  - Can leverage something like redis to do this, as well
- Leverage restarts to dispatch to available nodes
Future
Links

Documentation
- Docker Swarm Mode

Source Code
- SwarmKit
- SwarmKit Protobuf/GRPC

Interesting Topics
- Borg Paper
- Raft Consensus Algorithm
- Control Theory
Booth D38 @ LinuxCon + ContainerCon

Tues Oct 4th
- Build Distributed Systems without Docker, using Docker Plumbing Projects - Patrick Chanezon, David Chung and Captain Phil Estes
- Getting Started with Docker Services - Mike Goelzer
- Swarmkit: Docker’s Simplified Model for Complex Orchestration - Stephen Day
- User Namespace and Seccomp Support in Docker Engine - Paul Novarese
- Build Efficient Parallel Testing Systems with Docker - Docker Captain Laura Frank

Wed Oct 5th
- How Secure is your Container? A Docker Engine Security Update - Phil Estes
- Docker Orchestration: Beyond the Basics - Aaron Lehmann
- When the Going gets Tough, get TUF Going - Riyaz Faizullahbhoi and Lily Guo

Thurs Oct 6th
- Orchestrating Linux Containers while Tolerating Failures - Drew Erny
- Unikernels: When you Should and When you Shouldn’t - Amir Chaudhry
- Berlin Docker Meetup

Friday Oct 7th
- Tutorial: Comparing Container Orchestration Tools - Neependra Khare
- Tutorial: Orchestrate Containers in Production at Scale with Docker Swarm - Jerome Petazzoni
THANK YOU