ons
NORTH AMERICA
OPEN NETWORKING //
Enabling Collaborative
Development & Innovation

Next-Generation NFV Orchestration

Tal Liron
Principal Software Engineer, Red Hat

Hosted By
THE LINUX FOUNDATION | LF NETWORKING

Don’t Tear Down the House

“Shall I rewrite or revise
My October symphony?
Or as an indication
Change the dedication
From revolution to revelation?”
-- Pet Shop Boys

Seachange in Orchestration

1. Lifecycle Management ⇒ Scheduling
2. Orchestra Conductor ⇒ Orchestra Coordinator
3. Workflows ⇒ Policies
4. Standard Models ⇒ Standard Grammars

Three Lifecycle Kingdoms

1. Network Services
   a. Abstractions of *extremely* heterogeneous topologies
   b. SDN
2. Network Functions
   a. Raw virtual resources
   b. Configuring the hardware
3. Infrastructure
   a. Edge *requires* automation

Lifecycle Management: The Unwinnable War

- Complex workflow (otherwise we wouldn't have to "manage" it)
- Must keep track of state of many and diverse resources
  (A lot of state: logs, timings, databases, queues)
- When things go wrong the system is left in indeterminate state
  (Things go wrong a lot; clouds are expected to be unreliable)
- Complexity of automation reflects the complexity of the lifecycle
- So, now we have even more things that can go wrong

Surrender to Inversion of Control (IoC)

- Don’t even try to manage lifecycle (because you’ll always do a bad job)
- Express what you need (“intent-based”, “scheduling”) and let the infrastructure do its thing (it knows best)
- The Hollywood Principle: “Don’t Call Us, We’ll Call You”
- “Coordinator” instead of “conductor”
- This is what “cloud-native” specifically means for orchestration

What Happened to LCM in Kubernetes?

1. It's hardcoded:
   a. container or VM image is loaded
   b. networks are assigned
   c. configs are mapped
   d. entry point is called
2. Welcome to cloud native! Application will now manage itself
3. ...With the help of operators (next slide)

From Conductor to Coordinator

The Evolution of Hopes and Dreams

Prehistory

chroot/cages

Monit

supervisord

OSGi

Lightweight VMs

Docker

Mesos

LXD

Microservices

Kubernetes

OpenShift

Marathon

Service Mesh

Istio

Linkerd

Conduit

(Hystrix, Finagle)

portability

isolation

composition

coordination

Kubernetes + Operators

- Kubernetes is an *extensible* orchestration platform
- Containers *or* virtual machines
- Add your own domain-specific resource types with their own embedded micro-manager (S-VNFM)
- We can build a **Kubernetes Network Function Extension** (KNFE), essentially a cloud-native G-VNFM

Policy-Driven Orchestration

- Policies are restrictions or suggestions for orchestration
- A “resource” in Kubernetes is really a kind of policy
- An operator attempts to enforce that policy
- Higher-level policies: rules and regulations, checks and balances, quotas and exceptions
- Machine learning

Incessant Remodeling

- Pardon our dust!
- Standards are always behind the technology
- The “lowest common denominator” model purposely disables platform and vendor advantages
- One size *never* fits all
- Workarounds, hacks, square pegs in round holes

Standard Grammars

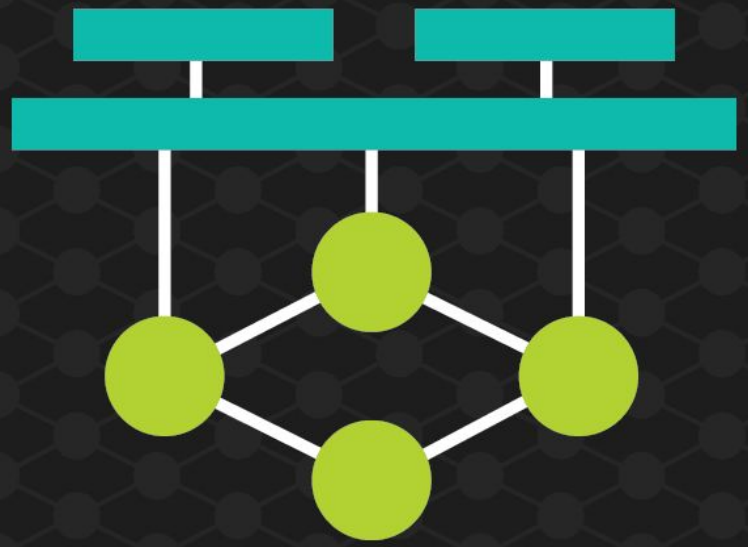
- Vendors will tell you what models work for them (they know best)
- Batteries included: they also bring the software to manage and configure their quirky models (Kubernetes operators)
- We just need to make sure our system can consume and operate these models

Grammar != Language

- TOSCA is OK, whatever
- Its “Simple Profile” and “Simple Profile for NFV” are awful (lowest-common denominator models)
- Need to support *arbitrary* node and interface types and extensibility
- TOSCA doesn’t matter, its (currently unused) features do

A Standard Grammar for Policies

- Make sense in the terms of the models (extension of TOSCA?)
- But also in terms of external systems (OSS/BSS)
- Policies come from various internal and external departments
- Can be machine-generated
- Publishable



ons

NORTH AMERICA

OPEN NETWORKING //
Enabling Collaborative
Development & Innovation

Hosted By

THE LINUX FOUNDATION | LF NETWORKING