Let’s Move Everything to Kubernetes!

ONAP “Dublin” Developer Forum
Jan 9, 2019

Tal Liron
Principal Software Engineer
NFV Partner Engineering
Poll

Is it a good idea to deploy all our network functions on Kubernetes?

1. No
2. It’s appropriate for a subset of use cases (edge?)
3. ?
4. Maybe later; for now let’s focus on OpenStack
5. YES!
Part 1
Why Kubernetes?
We’ve been doing it wrong

Lifecycle management is the enemy:

- 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
Paradigm shift (or: back to basics)

- “Scheduling” is intent-based
- “Life” has no “cycle”—it’s just a binary, either scheduled or not
- And so LCM is an implementation detail (wait for next slide)
- Also: containers are an implementation detail (we deal with “pods”)

“Legacy” orchestration has the wrong metaphor: it’s actually more like puppeteering than conducting an orchestra (tangle of strings)

In Kubernetes we are finally truly orchestrating (back to basics) where every part of the orchestra knows its music
What happened to LCM?

Two things:

1. It’s hardcoded:
   a. “container” image is loaded
   b. network is assigned
   c. configs are mapped
   d. entry point is called

2. Welcome to cloud native! Application will initialize itself
   We used to call this Inversion of Control (IoC)

Bottom line: LCM is not the orchestrator’s problem anymore
The Evolution of “Containers”

Prehistory
- chroot/cages
- Monit
- supervisors
- OSGi

Lightweight VMs
- Docker
- Mesos
- LXD

Microservices
- Kubernetes
- OpenShift
- Marathon

Service Mesh
- Istio
- Linkerd
- Conduit
  (Hystrix, Finagle)

portability
isolation

orchestration
composition
Part 2
Instead of OpenStack?!
Have your cake and eat it

I know what you’re going to say:
Kubernetes seems great, but CNFs don’t really exist yet

But …

We can fully support “legacy” VNFs on Kubernetes
And we should because reducing LCM is a big win

(We’re talking network functions; rest of data center can be whatever)
Kubernetes Network Function Extension

KNFE (pronounced “knife”)

Off-the-shelf:

- Kubernetes (production-ready distributions, like OpenShift)
- KubeVirt (the “V” in “VNF”)
- Multus (and maybe Network Service Mesh)
- Cluster API

Missing:

- LCM controller for VNFs
- Network-as-resource operator
- Network service operator

(All the above should be cloud-native, almost stateless, tiny)
KNFE Runtime Architecture

Heat + Mistral Adapter

- LCM Controller
- Network-as-resource Operator
- Network Service Operator

Connection Points

Pod
- Virtual Machine

Ansible, Chef, NETCONF, etc.

(P.S. All of this can be modeled in TOSCA)
Part 3
What About ONAP?
KNFE and ONAP

Manager (“NFV-O”):
SO, SDC, A&AI, SDN-C, DCAE, some S-VNFM

Embedded in infrastructure (“NFV-I”):
- Kubernetes = VIM + Multi-Cloud
- KNFE = G-VNFM + VF-C + App-C
- Custom operators = some S-VNFM
THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHat