Agenda

• Microservices introduction
• Containers
• Container Management
• Kubernetes architecture
• Adoption challenges for microservices
• Container Networking
• Service Mesh
• Key Takeaways
• Experiences in integrating ODL with microservices
• Experiences with ODL CNI plugins and COE
• Q&A
A monolithic application puts all its functionality into a single process... 

... and scales by replicating the monolith on multiple servers

A microservices architecture puts each element of functionality into a separate service... 

... and scales by distributing these services across servers, replicating as needed.

Reference / Image credit : https://martinfowler.com/articles/microservices.html
Containers

• Microservices is an architectural guidance for building apps
• Apps can be built as
  – Services on a single OS on a bare-metal [Issues: Services can have conflicting library versions. Dependency management is an issue]
  – Each service in a VM [Issues: Compute utilization unoptimized]
  – Each service in a container
    • Lightweight and isolated execution environment
    • Consistent environment across development, test, staging and production
    • Granular control on workload placement
    • Better options for horizontal scaling
    • Improved resource utilization
• Microservices does not dictate use of containers (Eg. Netflix)
  – But containers are a great way to decompose large applications
The need for container management

• Services will always have failures. Create a resilient system to deal with issues, rather than targeting to develop perfect microservice components

• “Pet” vs “Cattle” approach

• Typical management functions
  - Configure / Deploy
  - Upgrade
  - Scale
  - Discover
  - Load Balance
  - Network
  - Decide Placement
  - Federate
  - Authenticate
  - Predict resource needs
  - Manage life-cycle
  - Manage quota
  - Monitor
  - Query
  - Health-check
Challenges with microservices adoption

• Existing applications and VNFs almost need a rewrite/reorganize to migrate to the microservices architecture model. Needs huge investments

• Increased East-West network traffic between components because of the distributed model

• Difficulty in enforcing security/policy, because of the large attack surface
Container Networking - Introduction

- **Single Host**
  - Docker models (Bridge, Host, Container)
  - Linux MACVLAN / IPVLAN
  - Direct attachment to SRIOV
- **Multi Host**
  - L2 - Flannel
  - L3 - Calico
- **External world interaction**
- **IP address management**
- **Port allocation**

Image credit: https://thenewstack.io/hackers-guide-kubernetes-networking/
Application Networking requirements

• Application networking needs (L7)
  – Discover services
  – Handle timeouts / retries
  – Load balance / rate-limit
  – Implement circuit-breakers
  – Distributed tracing

• Service Mesh
  – Separate network functions from business logic
  – Push network-functions into infra
  – Facilitates fault & latency injection

Image credit: Oreily/Nginx
Istio Architecture

Image credit: https://istio.io/docs/concepts/what-is-istio/arch.svg
Key Takeaways

• Containers are a great way to decompose large applications
• Container orchestration/management needed to operate container based applications at scale
• Service Mesh is an essential component of microservices development
  – Policy/Security
  – Observability
  – Uniformity
ODL integration with microservices bus

Developer Interfaces
- RESTCONF
- JAVA
- ZeroMQ

Value Added Applications
- Flow Manager
- Topology Manager
- Path Computation

Controller Core
- OPEN DAYLIGHT

Southbound Plugins
- OpenFlow
- NETCONF
- OVSDB
- PCEP
- PCMM
- BGP

3rd Party containerized Applications
- Analytics
- Monitoring & Telemetry
- Custom

ZMQ Messaging Bus
- Containerized South Bound Interfaces
- CLI
- TL1
- REST
- Custom

Legacy Devices
- EMS

NFV
Virtual

Physical

3rd-Party
Container Networking Challenges

• Currently built for homogenous, high throughput, enterprise-centric application clusters

• Needs more tweaking for L2/L3 use-cases of Telcos

• Enabling container orchestration frameworks to access and leverage the advanced networking capabilities of commercial switch vendors is desirable

• Operators don’t want to give up key capabilities in one area of the system (networking) for gains in another (compute)

Reference / Image credit: https://github.com/ligato/networkservicemesh
Extensions to Opendaylight COE for physical underlay

Kubernetes

Grpc

COE watcher

Grpc

Kubelet

Grpc

Kubeproxy

Grpc

COE CNI

IPTables

OVS

VPP/Switch/TOR

JSON RPC

Fabric MNG

ODL

Ansible

SALT

SRIOV

HW offload

Netconf

Netvirt

Kubeproxy

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR

Kubelet

COE CNI

IPTables

OVS

VPP/Switch/TOR
OPEN SOURCE NETWORKING DAYS