Building container-based NFV solutions with OPNFV, ONAP and VPP on Arm platform

Tina Tsou <tina.tsou@arm.com>
Trevor Tao <trevor.tao@arm.com>
Agenda

• Background
• Project Auto
• Compass4nfv on Arm
• Container4nfv on Arm
• OPNFV CI/CD for Arm
• VPP and Auto on Arm
Background
Projects with Arm

Armband
• The purpose of this project is simply to integrate and test all aspects of OPNFV releases on ARM-based servers.

Yardstick
• A test framework with test cases and test stimuli to enable NFV-I performance verification

Auto
• This project focuses on ONAP component integration and verification with OPNFV reference platforms/scenarios

Compass4nfv
• An installer project based on open source project Compass, which provides automated deployment and management of OpenStack and other distributed systems
• Ansible is used by default.
• Our main installer for OPNFV Container4NFV project

Container4NFV
• Provide a container full-stack environment where VNF can run, including data plane VNF and control plane VNF. Let the platform support container and virtualization technology. Collect requirement for containerized NFVs.
• Previously named as OpenRetriever
• What are we focusing on for building Arm’s containerized NFV infrastructure now
Project Auto
This project focuses on **ONAP component integration and verification with OPNFV reference platforms/scenarios**, through primarily a post-install process in order to avoid impact to OPNFV installer projects.

Related Project Opera: developing OPNFV-installer supported scenarios that can deploy and verify ONAP as a whole.
Auto (ONAP-Automated OPNFV)

Validate ONAP (Open Network Automation Platform) as NFV Orchestrator and VNF Manager in OPNFV ecosystem; Auto project [home page](#).

Show added value of:

- Automation using closed loops (defined in CLAMP), policies (defined in Policy Framework), and DCAE (real-time monitoring, execution of closed loops and policies; also alarm correlation)

- Design-time portal-based (as well as API-based control) streamlined VNF lifecycle management: Onboarding (with SDC, to define VSPs with VLMs, and end-to-end Services), Deployment (with VID and MSO), and Operations (with persistent inventory data in AAI)
Auto (ONAP-Automated OPNFV) Use Cases

Three specific use cases for Auto:

1. **Edge Cloud** - autonomy of Edge Cloud management
   - Autonomy enabled by *systematic* catalog-based VNF deployment through SDC/VID/MSO, *automated* monitoring and management through MSO, DCAE, CLAMP, Policies, and an array of controllers

2. **Resilience** - improvements through ONAP
   - Failure recovery time reduction with ONAP, thanks to automated monitoring and management

3. **Enterprise vCPE** - ensure high performance, enterprise-grade vCPEs in Edge Cloud)
   - Rely on *enterprise-grade vCPE VNFs*, properly onboarded (tested, certified, approved: multiple Roles in onboarding process), and properly monitored and managed for *performance assurance* (SLAs and High Availability: redundancy, recovery)
Auto ONAP on Kubernetes Architecture

Kubernetes Client
Could be a k8s node or collocated server

OpenStack
Rackspace or vanilla

DCAE VM
vFirewall VM

Kubernetes Host(s)

Rancher Container Management System (optional)

createAll.bash
Onap-parameters.yaml

kubectl client
config container (stopped)
kubectl server
rancher agent container
rancher server container

DCAE docker container (1)
docker containers (70+16 filebeat)
/dockerdata-nfs
Auto (ONAP-Automated OPNFV) Status

Test cases:

- Top-down definition (methodology like FuncTest)
- For initial Auto project purposes, test environment may not need to be as complete and systematic as YardStick, FuncTest, or Robot; later, alignment with FuncTest and YardStick will be sought
- Script development environment target: Python3, maximize similarity between 3 use cases

Dependency on DCAE, deployment on K8S (hopefully from OOM, for ONAP Beijing release, end March 2018; external dependency: containerized Cloudify)

Infrastructure deployment: two CPU architectures (x86 and Arm)

- 2 pods (6 physical servers each, one x86, one Arm) at UNH IOL (University New Hampshire, Interoperability Lab)
- x86 pod: ONAP on Kubernetes on Bare Metal; no DCAE yet;
- Arm pod: target is ONAP on Kubernetes on OpenStack VMs; also no DCAE yet;
- installation tools:
  - x86: (TBC)
  - Arm: MCP for OpenStack, new tests for K8S (existing methods don’t work)
- Note: goal is to have ONAP on Kubernetes in both pods
UC2 Resilience Improvements: Module Design

Main
Interactive Menu:
View Test Definitions
Edit Test Definitions
Run A Test
View Test Results

Manage Test Definitions
View/Edit/Delete:
physical resources
cloud resources
VNFs
recipients (OS, cloud/VNF managers)
challenge definitions
optional metrics
test definitions
Initialize Data (initial storage population)

Persistent Test Data
(File, CSV, binary, DB)

Run Test
Receive/retrieve chosen test def info
pre-test (pings)
launch test:
create execution instances of Test and Challenge
simulate challenge
get time T1
loop:
wait for VNF recovery
optional other metrics
store data and logs
get time T2
stop challenge
reset (with ONAP MSO)
store data and logs
post-tests
logs

Interface With Servers/OS
CLI send/receive
API send/receive

Interface With Cloud Managers
CLI send/receive
API send/receive

Interface With VNF Managers and NFV Orchestrators
CLI send/receive
API send/receive

Persistent Test Data
Auto (ONAP-Automated OPNFV): VNFs in scope

<table>
<thead>
<tr>
<th>VNF</th>
<th>x86 pod</th>
<th>Arm pod</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>onboarded</td>
<td>deployed</td>
</tr>
<tr>
<td>vFW</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>vLB</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>vCPE</td>
<td>n</td>
<td>n</td>
</tr>
</tbody>
</table>

\[\textbf{vFW} = \text{virtual Firewall}\]
\[\textbf{vLB} = \text{virtual Load Balancer}\]
\[\textbf{vCPE} = \text{virtual CPE (Customer-Premise Equipment)}\]
\[\textbf{Future} = \text{focus on more Edge Cloud VNFs; explore vlMS (ClearWater)}\]
ONAP Installation Target on Arm Pod: ONAP/K8S/VM

- Test Case
- VM-based (nova) VNF
- OpenStack Controller Node (mgt API)
  - Nova hypervisor, ...
  - OpenStack Compute Node
- OpenStack Compute Node
- OpenStack Network Node
  - neutron controllers, ...
- OpenStack Storage Node
  - cinder, swift controllers, ...
- Containerized Test Case
- Kubernetes Container Orchestration Engine (COE)
- K8S Cluster
  - K8S Pod
    - Container (app + libs)
    - Container (app + libs)
  - K8S Pod
    - Container (app + libs)
    - Container (app + libs)
- Container Runtime (Docker)
  - Container (app + libs)
  - Container (app + libs)
- VM (nova instance)
  - Guest OS
- VM (nova instance)
  - Guest OS
- VM (nova instance)
  - Guest OS
- App
  - VM (nova instance)
    - Guest OS
  - OpenStack Controller Node (mgt API)
    - Nova hypervisor, ...
    - OpenStack Compute Node
- OpenStack Compute Node
- OpenStack Network Node
  - neutron controllers, ...
- OpenStack Storage Node
  - cinder, swift controllers, ...
- Containerized Test Case

KVM (Kernel Virtual Machine)
Linux OS (Ubuntu 16.04 Xenial)

(2x) aarch64 Cavium ThunderX
  96 cores, 128G RAM, 447G SSD

(4x) aarch64 Cavium ThunderX
  48 cores, 64G RAM, 447G SSD
Compass4NFV on Arm
**Compass4nfv for Kubernetes – Now Arm Installer Support**

**Compass4NFV on Arm** (Yibo Cai, Di Xu):

**What We Have Done:**

Ported Compass4NFV docker images for AArch64 and uploaded to dockerhub Linaro repo.

- Compass-tasks-k8s
- Compass-deck
- Compass-mq
- Compass-cobbler
- Compass-db

Supported AArch64 bare metal deployment (CentOS7)

Supported deploying Kubernetes cluster on AArch64 virtual and bare-metal nodes.

Our ‘F’ release scenarios for Container4NFV would be based on the work in Compass4NFV

**Compass4NFV repo**
Container4NFV on Arm
Container-based NFV Ecosystem on Arm

ONAP supports multiple VNF environments by integrating with multiple VIMs, VNFMs, SDN Controllers, and even legacy equipment.

Kubernetes as COE

Multus plugin for Kubernetes as CNI

Flannel/DPDK/Vhost user CNI plugins integrated

Ref: https://wiki.opnfv.org/pages/viewpage.action?spaceKey=OpenRetriever&title=Container%27s+Architecture+for+Cloud+Native+NFV
Container Networking Acceleration with DPDK

K8s MASTER
- API Server
- Scheduler
- Controller Manager

etcd

K8s Node
- KUBELET
- KUBE-PROXY
- Flannel CNI

Containers
- Docker
- VNF

Kernel
- docker0 bridge
- Flannel0 bridge

K8s Node
- KUBELET
- KUBE-PROXY
- Multus CNI

Containers
- Docker
- VNF

VFIO/UIO
- SR-IOV CNI

Kernel
- PF
-VF
- VF

Docker0 bridge
Flannel0 bridge
VPP and Auto on Arm
Linking VPP and Auto for Arm architecture

VPP data-plane optimized VNFs (FD.io, DPDK), validated for Arm architecture

The VPP-optimized VNFs are containerized (Kubernetes/Docker)

The VPP-optimized VNFs can be managed by ONAP (onboarded and deployed as end-to-end Services)

ONAP itself (each component: SDC, CLAMP, DCAE, etc.) is also containerized and validated for Arm architecture

ONAP installer (OOM) and ONAP components leverage OPNFV installers such as Compass4nfv and Container4nfv

Auto use cases (current three, and future) will be tested on these VPP-optimized VNFs

Auto use cases will eventually align with OPNFV test frameworks (FuncTest, YardStick)

overall governance of Test-Frameworks/Auto-tests/ONAP/VPP-optimized-VNFs on Arm architecture: Armband
Use Case: (TBD)

- Container
  - VPP/DPDK
  - Flannel/Calico
  - 10GB E

- Container
  - VPP/DPDK
  - OFP
  - FlowCache
  - Nginx
  - Data Flow Server
  - Data Flow In

- Container
  - VPP/DPDK
  - Video Client

- VXLAN Overlay
Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!
감사합니다
धन्यवाद