The Container Story: Run your apps and tools natively on Cisco boxes

Per Jensen, per@cisco.com
June, 2015
“My Stuff”

- **Fast feature delivery** – independent from network OS release cycle
- **Custom feature delivery** – “for me only”
- **Integration** into existing/custom tool chains
- **Security domains** – isolate “my stuff” from network OS
# Network Element Customization: Examples

## DataCenter - DevOps
- DevOps Configuration Automation Toolchain Integration (Puppet, Chef, Ansible, CFEngine, ..)
- Linux tools (e.g. Ganglia)
- Custom scripts in Python

## Service Provider
- Custom analytics/stats aggregation
- Custom VPN
- Custom routing
- Service chaining
- Custom OS (combining open-source and vendor-source)

## IoT
- Single device hosting for networking and non-networking functions – incl. DNS-, DHCP-, SIP intercept, load balancing, AP-light, OOB management proxy, ...
- Protocol gateway (for protocols used in different embedded deployments)
- Custom routing (e.g. active-active)
- Linux-tools

## Enterprise
- Custom analytics (e.g. brownout analysis)
- Integration with Enterprise apps (Unified Comms)
- Custom routing (e.g. military)
- Custom traffic manipulation
Towards An Eco-System For Customization

- Empower Cisco-Services, ISV/Partners, Customers to innovate and customize
- Custom feature delivery independent from Network OS software release cycle

Examples:
- Automated management (tools/scripts)
- Automated system visibility
- Automated fault detections
- Proprietary business and networking functions
- …
Linux Containers

• A Linux container lets you run a Linux system within another Linux system.

• A container is a group of processes on a Linux machine.

• Those processes form an isolated environment.

• Inside the container, it looks like a VM.

• Outside the container, it looks like normal processes running on the machine.

• It looks like a VM, but it is more efficient: Containers = Lightweight Virtualization
Containers are almost like Virtual Machines

• Containers have their own network interface (and IP address)
  • Can be bridged, routed... just like with Xen, KVM etc.

• Containers have their own filesystem
  • For example a Debian host can run Fedora container (and vice-versa)

• Security: Containers are isolated from each other
  • Two containers can't harm (or even see) each other

• Resource Control: Containers are isolated and can have dedicated resources
  • Soft & hard quotas for RAM, CPU, I/O...
Containers and Virtual Machines

Containers are isolated but share OS and where appropriate bins/libraries.
Containers: Classes of Use-Cases

- Extend Network Elements’ Functionality
- Web-Scale Micro-Services
- Network Function Virtualization
Motivation: Containers on Network Elements

• Network OS Independence
  • System Modularity
  • Limit kernel dependencies

• Enable multiple constituents to provide for device enhancements
  • Cisco Development, Cisco Services, Partners, Customers, …

• Leverage existing tools/tool-chains and eco-systems
  • Integration of network device with server-centric tool chains

• Isolation / Security
  • Application-focused deployment, as opposed to base-OS focused deployment
    (“escape dependency hell”)
Containers for network elements: Enabling Plugins

- Virtualized environment to host “applications” on a Cisco device.
  - Wide range of “applications” – shell, virtual services, plugins…
  - Applications can be developed and release independent from Network OS release cycles

- Application Examples:
  - Cisco Virtual Services:
    - Integrated Appliance: ISR4451X-WAAS (KVM)
    - Linux shell: Guest shell (LXC)
  - Cisco Plugins:
    - Features with decoupled release cycles: Puppet Plugin, Chef Plugin (LXC), Splunk universal forwarder plugin
  - Third Party Services
Container Network Models

Shared
Applications inside the container appear as applications running natively on the host
Examples: Nexus 3k, 9k, Cat 3k, 4k, Titanium

- Container 1
  Network namespace: Host
  Container interfaces: eth0, eth1, eth2
- Container 2
  Network namespace: Host
  Container interfaces: veth0, veth1, veth2

- Host platform
  Network namespace: Host
  Interfaces are directly mapped to container
  eth0, eth1, eth2

Dedicated
Applications inside the container appear as appliances on a subnet reachable from the host
Examples: ASR 1k, CSR 1kv, ISR4k, ISR 819

- Container 1
  Network namespace: N1
  Container interfaces:
  - veth0
  - veth1
  - veth2
- Container 2
  Network namespace: N2
  Container interfaces:
  - veth0
  - veth1
  - veth2

- Host platform
  Network namespace: Host
  Multiple bridges and virtual topologies possible
  Physical interfaces: eth0, eth1, eth2

- Linux Bridge
- Forwarding Plane
- VPG
Containers On Network Elements: High-Level Architecture

- **Host Kernel**
  - IOSd
  - NXOS
  - vman
  - libvirt
  - libcontainer

- **Host User Space**
  - System Processes
  - API/onePK

- **Virtual Service (container)**
  - Packet Connectivity
  - Storage
  - Console/Logging
  - Resource Control
  - Separate Namespace

- **On Linux-based systems**
- Application packaged as .ova

Manages virtualization infrastructure and virtual service lifecycle (install, activate, ...)

Presentation ID © 2015 Cisco and/or its affiliates. All rights reserved. Cisco Public
OVA - Open Virtual Appliance

OVA is a tarball of these files:

- Configuration for the virtual environment
  - Amount of host resources for the guest
  - Guest storage
  - Network interfaces

- Apps (E.g., Puppet, Chef, …)

- Package manifest
- Authentication Certificate
- Disk image (.iso)
- File systems

Apps
(E.g., Puppet, Chef, …)
Virtual Service Deployment Workflow: Hosted Service Deployment Model

router#virtual-service install name <app_name> package <file_uri>
router#virtual-service uninstall name <app_name>
router#virtual-service upgrade name <app_name> package <file_uri>
router#show virtual-service connect
router#show log
router#interface VirtualPortGroup1 ip address 3.3.3.1 255.255.255.0
router#virtual-service <app-name> interface virtualPortGroup1 ip address 3.3.3.2 profile app-model-1
router#virtual-service <app-name> activate
router#virtual-service global
router#virtual-service list
router#virtual-service detail name <app-name>
router#virtual-service utilization name <app-name>

Guest Shell as a virtual service
EEM and Guest Shell:
EEM triggers Guest Shell commands/scripts as actions

```
dt-n9k3# conf t
Enter configuration commands, one per line. End with CNTL/Z.
dt-n9k3(config)# feature evmed
```

```
dt-n9k3(config)#
```

Enable enhanced EEM event detectors

```
event manager applet test
    event cli match "show version"
        action 1 cli guestshell python eem.py
        Run Guest Shell command when applet triggers
```

```
dt-n9k3# show version
%Command blocked by event manager policy
```

```
dt-n9k3% 2014 Aug 4 12:18:48 dt-n9k3 %$ VDC-1 %$ %USER-0-SYSTEM_MSG: %ONEP-0-TESTAPP: Syslog Message from EEM python guestshell run - onep
```

Towards An Open Ecosystem For Network Components

“Cisco”
Cisco focused apps

Integration Plugins
(Puppet, Chef, OpenFlow,…)

“Open Packages”
Cisco + 3rd-party packages

Guest Shell

“Reference Ecosystem”
Cisco + 3rd-party packages + reference environment

Orchestration & Lifecycle Management

Portable Network Components
In Open Reference Environment

Presentation ID  © 2015 Cisco and/or its affiliates. All rights reserved. Cisco Public
Open Application Container: Developer Perspective

- Common Environment for Network Functions
  - Physical Router/Switch - Virtual Router/Switch
  - Server/Application (incl. “Controller”)

- Common Development Experience
  - Linux-style programming experience
  - Typical networking functions available to developer as integrated libraries
  - Distributed/Centralized/Combined NFs

- Common Runtime Experience
  - Run network function at the most optimal place in the network (device, controller, application server), depending on NF’s needs (performance, scale, ease of use)

- Open Ecosystem / Open-Source
Summary

- **Containers as a tool to complement the OS of the network element**
  - Fast feature delivery – independent from network OS release cycle
  - Enable custom feature delivery – “for me only”
  - Tool to provide for easy integration into existing/custom tool chains
  - Security domains – isolate “my stuff” from network OS

- **Deployment examples**
  - Integration tools (Puppet, Chef, .. plugins)
  - Guest-Shell
  - Network Applications (Distributed Network Analytics, Self Learning Networks, …)
ISR 4000 Architecture and Virtualization
Cisco ISR 4000 Family I/O Design

Management Interface
out-of-band control plane connection directly to a management network

Front-Panel GE
- RJ45/SFP GE Interfaces
- PoE+ available on some models

Network Interface Modules (NIMs)
- Larger and more powerful than EHWICs
- Up to 8 ports per module
- DSPs directly on modules

Optional Drive NIM for Embedded Applications
- RAID 1 for data protection
- Single HD (future) and dual SSD options

USB Connections
- 2 times type A for file storage
- USB type B console in addition to RJ45 console and aux ports

Enhanced Service Modules
- Compatible with Cisco® ISR G2
- Up to 10-Gbps connection to system
- Faster and more powerful than SMs
Cisco ISR 4400 Series Architecture

Service containers live here

Control Plane (1 core) and Services Plane (3 cores)

Data Plane (6 or 10 cores)

Multigigabit Fabric

IOSd

FPGE

ISC

SM-X

NIM
ISR 4k – Scaling Application features and performance
Cisco onePK - Write Once, Run Anywhere

Process Hosting
- Cisco® Network Operating System
- Container
  - Embedded Network Services

End-Point Hosting
- Cisco® Network Operating System
- External Server
  - Feature or Application

Blade
- Network Services and Applications
  - Container
  - Feature
Service Virtualization for Networking

Service Containers

- Dedicated virtualized compute resources
- CPU, disk, memory for each service
- Easily repurpose resources
- Industry-standard hypervisor

Benefits

- Better performing network services
- Ease of deployment with zero footprint; no truck roll
- Greater security through fault isolation
- High reliability
- Flexibility to upgrade network services independent of router IOS® Software
## ISR-WAAS Scalability

<table>
<thead>
<tr>
<th>Platform</th>
<th>WAAS Option</th>
<th>Hardware</th>
<th>TCP Connections</th>
<th>Target Optimized WAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>4451</td>
<td>ISR-WAAS</td>
<td>16GB DRAM, 32GB Flash, 2x200GB SSD</td>
<td>2500</td>
<td>150 Mbps</td>
</tr>
<tr>
<td></td>
<td>vWAAS</td>
<td>UCS-E single or doublewide</td>
<td>2500-8500</td>
<td>200 Mbps+</td>
</tr>
<tr>
<td>4431</td>
<td>ISR-WAAS</td>
<td>16GB DRAM, 16GB Flash, 1x200GB SSD</td>
<td>1300</td>
<td>100 Mbps</td>
</tr>
<tr>
<td></td>
<td>vWAAS</td>
<td>UCS-E single or doublewide</td>
<td>1300-7300</td>
<td>80 Mbps+</td>
</tr>
<tr>
<td>4351</td>
<td>ISR-WAAS</td>
<td>16GB DRAM, 16GB Flash, 1x200GB SSD</td>
<td>1300</td>
<td>100 Mbps</td>
</tr>
<tr>
<td></td>
<td>vWAAS</td>
<td>UCS-E single or doublewide</td>
<td>1300-7300</td>
<td>80 Mbps+</td>
</tr>
<tr>
<td>4331</td>
<td>ISR-WAAS</td>
<td>16GB DRAM, 16GB Flash, 1x200GB SSD</td>
<td>750</td>
<td>75 Mbps</td>
</tr>
<tr>
<td></td>
<td>vWAAS</td>
<td>UCS-E singlewide</td>
<td>3900</td>
<td>80 Mbps+</td>
</tr>
<tr>
<td>4321</td>
<td>ISR-WAAS</td>
<td>8GB DRAM, 8GB Flash, 1x200GB SSD</td>
<td>200</td>
<td>50 Mbps</td>
</tr>
</tbody>
</table>
WAAS Profiles and Requirements
4451 Example

- Minimum requirements for WAAS on the Cisco® 4451 ISR are 8 GB CP DRAM, 16 GB Flash, and one 200 GB SSD.

<table>
<thead>
<tr>
<th>Profile Specifications</th>
<th>Router Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Name</td>
<td>Connections</td>
</tr>
<tr>
<td>ISR-WAAS-750</td>
<td>750</td>
</tr>
<tr>
<td>ISR-WAAS-1300</td>
<td>1300</td>
</tr>
<tr>
<td>ISR-WAAS-2500</td>
<td>2500</td>
</tr>
</tbody>
</table>

- EZConfig Script can have WAAS plus AppNav up and running in less than 10 minutes

```
router# service waas enable
**************************************************************************
**** Entering WAAS service interactive mode. ****
**** You will be asked a series of questions, and your answers ****
**** will be used to modify this device's configuration to ****
**** enable a WAAS Service on this router. ****
**************************************************************************
Continue? [y]:
At any time: ? for help, CTRL-C to exit.
```
Cisco Catalyst Switches – from Access to Backbone
Virtualization on several levels

- **Catalyst 3650**
  - Lower end fixed switching with Unified Access

- **Catalyst 3850**
  - Advanced fixed switching with Unified Access

- **Catalyst 4500E with SUP8-E**
  - Flexible, scalable, feature-rich modular access

**TCO**

**End-to-end Security**

**Application Visibility**

**Investment Protection**

**Performance & Scale**
IOS XE Evolution

- Modern IOS to enable multi-core CPU
- Easy customer migration
- While maintaining IOS functionality and look and feel
- Allow hosted applications like Wireshark
Catalyst 3850 Switch

- Freeware
- Bundled with Operating System
- Software Process
- Quick & Easy Remote Analysis
- Does NOT replace SPAN
POD1#sh monitor capture MY_CAP

Status Information for Capture MY_CAP
Target Type: Interface: Vlan, Ingress: 11
Status : Inactive

Filter Details:
  IPv4
  Source IP: any
  Destination IP: any
  Protocol: any

Buffer Details:
  Buffer Type: LINEAR (default)

File Details:
  Associated file name: flash:test.pcap

Limit Details:
  Number of Packets to capture: 0 (no limit)
  Packet Capture duration: 10
  Packet Size to capture: 0 (no limit)
  Packets per second: 0 (no limit)
  Packet sampling rate: 0 (no sampling)

POD1#sh monitor capture file flash:test.pcap

1 0.000000 00:00:00:00:00:00 -> 54:78:1a:be:c1:10 IEEE
  802.11 Probe Request, SN=0, FN=0, Flags=........

2 3.000000 00:00:00:00:00:00 -> 54:78:1a:be:c1:10 IEEE
  802.11 Probe Request, SN=0, FN=0, Flags=........

3 6.000000 00:00:00:00:00:00 -> 54:78:1a:be:c1:10 IEEE
  802.11 Probe Request, SN=0, FN=0, Flags=........

4 6.495961 11.1.1.101 -> 11.1.1.1  DTLSv1.0
  Application Data

5 6.496968 11.1.1.101 -> 11.1.1.1  CAPWAP CAPWAP-
  Control - WTP Event Request

6 6.499974 00:00:00:00:00:00 -> 54:78:1a:be:c1:10 IEEE
  802.11 Probe Request, SN=0, FN=0, Flags=........

7 6.502964 11.1.1.101 -> 11.1.1.1  DTLSv1.0
  Application Data

8 6.502964 11.1.1.101 -> 11.1.1.1  CAPWAP CAPWAP-
  Control - WTP Event Request

POD1#
Controller based on Container → LXC

• **LXC (LinuX Containers)** is an *operating-system-level virtualization* environment for running multiple isolated *Linux* systems (containers) on a single Linux control host.

• LXC provides a virtual environment that has its own process and network facility (CPU, memory, block I/O, network, etc. space..)

Source: [http://wiki.archlinux.org/index.php/Linux_Containers](http://wiki.archlinux.org/index.php/Linux_Containers)
Some background on FOG and the origins of IOx
The Internet of Things is Already Here

Rapid Adoption rate of digital infrastructure: 5X faster than electricity and telephony

50 Billion “Smart Objects”

Source: Cisco IBSG, 2011
IoT Requires Distributed Computing

Traditional Computing Model
(Terminal-mainframe, Client-server, Web)
IoT Requires Distributed Computing

IoT Computing Model
(Data Volume, Security, Resiliency, Latency)

DATACENTER/CLOUD

FOG

DEVICE
Paradigm Shift with Fog

Unified Platform

Network

Compute

Storage

STORE

ANALYZE

ACT

NOTIFY
Making Fog a Reality

IoT Computing Model
(Data Volume, Security, Resiliency, Latency)

DATA CENTER/CLOUD

IOx enables FOG

DEVICE
Cisco IOx Enables Applications At The Network Edge

Platforms at the Network Edge
- Application Management
- IOS
- IOx
- Application Store

Operating Systems
- Distributed Applications
- IOx SDK and Middleware Services

Embedded Compute
Embedded Storage
Accessible Interfaces

Sensors and Endpoints

© 2014 Cisco and/or its affiliates. All rights reserved.
IOx resource constraints, does it matter

- The HW platforms that we discuss with IOx today are resource constrained, but it’s irrelevant

- Focus on the use cases, rather than the HW capabilities
  - Getting more data into the network for analysis (basic sensor interactions)
  - Analyzing data at the edge to create the “Data-to-Information” conversation
  - Simple control systems at the edge for resiliency
  - HW consolidation for unnecessary servers at the edge

- Platforms with more power will become supported over time, here is a sampling of things to come:

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Virtual CPU</th>
<th>Compute (MHz units)</th>
<th>Memory (MB)</th>
<th>Persistent Disk</th>
<th>Access to Interfaces (including Types)</th>
<th>Resource allocation as documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>cgr1120</td>
<td>sc1</td>
<td>500</td>
<td>200</td>
<td>1G Flash</td>
<td>seu</td>
<td>Guest</td>
</tr>
<tr>
<td>cgr1240</td>
<td>sc1</td>
<td>500</td>
<td>200</td>
<td>1G Flash</td>
<td>seu</td>
<td>Guest</td>
</tr>
<tr>
<td>819</td>
<td>dc1</td>
<td>400</td>
<td>256</td>
<td>1G Flash</td>
<td>seg</td>
<td>Guest</td>
</tr>
<tr>
<td>809</td>
<td>sc2</td>
<td>1200</td>
<td>1GB</td>
<td>4GB Flash</td>
<td>seuga</td>
<td>Split TBD</td>
</tr>
<tr>
<td>829</td>
<td>sc2</td>
<td>1200</td>
<td>1GB</td>
<td>4GB Flash</td>
<td>seuga</td>
<td>Split TBD</td>
</tr>
<tr>
<td>UCS-E Single-Wide</td>
<td>dc2</td>
<td>1000</td>
<td>4 to 16GB</td>
<td>500GB/1TB spinning</td>
<td>eu</td>
<td>Guest</td>
</tr>
<tr>
<td>Additional Module</td>
<td>dc4</td>
<td>1700</td>
<td>2/8GB</td>
<td>32 to 256GB SSD</td>
<td>eu</td>
<td>Guest</td>
</tr>
</tbody>
</table>
Infinite possibilities in IOx, but where are we today

AKA: What can I do tonight, not next week or next month
Available Now: CGR 1120 and 1240 platforms with IOx

- CGR 1120 and 1240 support IOx with a Guest Operating Systems
  - 400Mhz CPU available for Guest OS + Application
  - 200MB of Memory for Guest + Application
  - 1GB SD Flash Card for Storage

- IOS CLI for Start/Stop/Upload of Guest Operating System

- Available in Solution Partner pages:
  - Sample CGR BOM for IOx Prototyping
  - Emulation VM with resource constraints configured
  - Application Development Guide
  - Performance Documentation
  - Yocto Linux image for use as Guest OS
Available Now: CGR1120 and CGR1240 with IOx
What is coming next and where are we taking IOx

Note: From here on in this prezo the capabilities and roadmap are all forward looking and subject to change
Cisco IOx: Types of Applications in the future

VM Packaged Applications
- Apps packaged as complete virtual machines, not tied to Host OS.
- Ideal for BYOI
- Linux, VxWorks, Windows etc
- Limited App management

Native/Container Applications
- Self-contained Native apps but tied to Host OS environment
- Low-level access, Custom RPMs
- Optionally, can be packaged as LXC containers

PaaS Style Applications
- Self-contained apps, portable
- Write-once, deploy across platforms
- Python, Java, Node.js etc
Cisco Iox: Application Management

Application Management

• Middleware between sensors and applications Logical entity for centralized app management
• Provides an “app” centric view
• Interfaces with App Store to browse available apps, app functionality etc
• Resource/Capability matching
• Northbound APIs for complete centralized life cycle management
• Standalone or Integrated into existing device NMS, SDN controller
Capability phasing: *work in progress*

Leverage what exists, test the market

- Release of CGR1K with Partner Provided VM and embedded CAF-lite
- BYOI on CGR1K

Expand GW Services

- Release 819 image for VM type app in IOx
- Data-In-Motion (DMO) integrated as a GW Middleware Service
- GW Middleware Service for IOS controlled interfaces from IOx (3G/4G/Wifi/etc.)
- *POC* for 3850
- CG-NMS with IOx Application Management
- Application Manager v1 (Standalone and service)
  - Scalability up to 500 IOx nodes initially

Expand Platform Support, Add Paas Type

- Release Container/Native app type support for CGR1K
- 809/829 support for IOx capabilities
- Release Container/Native app type support for 8xx
- Compute Module with CGR 1120 and 1240 (BYOI local and platform)
- Protocol Gateway Middleware Service Phase 1
- IOx IP Camera App Integration
- Paas Type with Java/Python for CGRs, 8xx
- App Manager V2
  - Scale to 5000 nodes

Switching and RTOS

- Switching platform releases (specific platforms TBD based on 3850 POCs)
- RTOS (real time operating system) with TSN (time sensitive networking) SDK capabilities
- General IOS-XE Platforms
- App Store v1
Thank you
TOMORROW starts here.