
TOMORROW
starts here.
CISCO
Cisco live!

# Data Centre Fabric Design: Leveraging Network Programmability and Orchestration

BRKDCT-3641

Brenden Buresh

DC Technical Solutions Architect

#clmel

Cisco live!

# Agenda

- Introduction – Data Centre Trends
- 2-Tier Leaf Spine Architecture
- VXLAN Overview & Primer
- DevOps & Network Programmability
- Controllers & Orchestration Tools
- Conclusion – Data Centre Design Transition

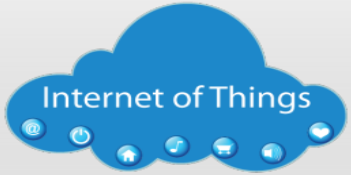
# Introduction – Data Centre Trends

Cisco live!

# New Data Centre Trends Cause Disruptions

Application Trends

## Big Data


25% CAGR—Big Data[1]
10G LoM[3]
75% Bare-Metal[2]

## Web 2.0 / DevOps





45% Multi-Hypervisor[4]
Linux Containers

## Public / Private Cloud


2/3rd of all Workloads in Cloud by 2017

Impact on IT Infrastructure

**A New Application Centric Infrastructure is Required**





Cisco live!

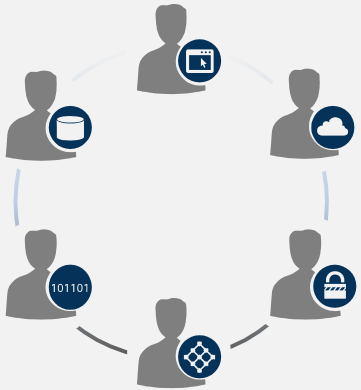
# Benefits of the Cisco Approach – Application Agility

## Delivery Agility Across the Application Lifecycle – Day Zero and Beyond

| Application Deployment Time | minutes | days | weeks |
|---|---|---|---|

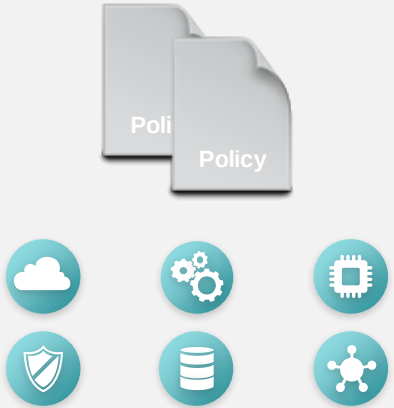
### Common Policy Framework

#### Faster Process Handling




### Policy-based Automation

#### Faster Rollout at Scale




### App + Infrastructure: Visibility

#### Faster Troubleshooting



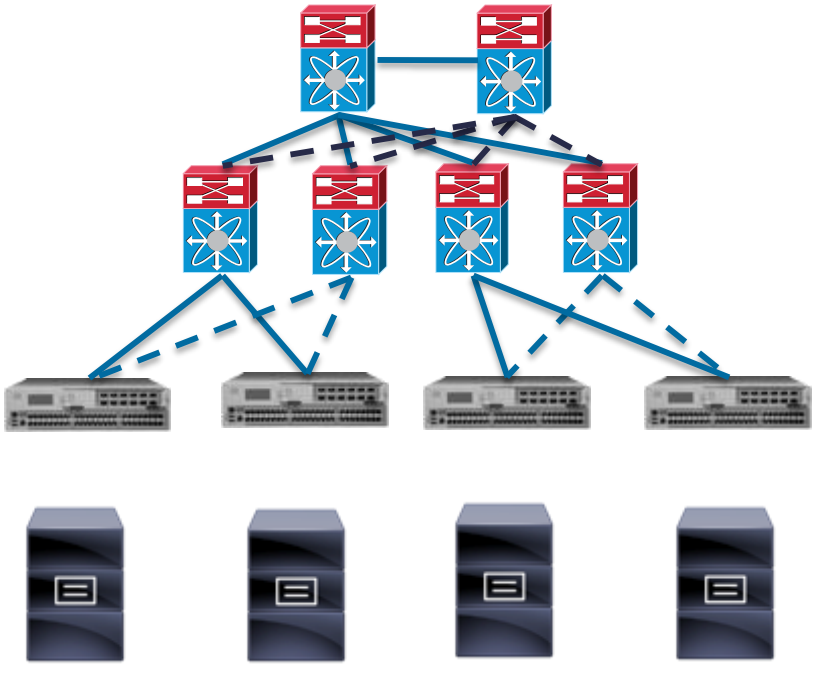


Cisco live!

# 2-Tier Leaf Spine Architecture

Cisco live!

# Networking Today



Designed for North / South campus traffic

East / West traffic is inefficient & can be unpredictable

Bandwidth & latency is not deterministic





Cisco live!

# Networking Challenges

## Poor Scalability

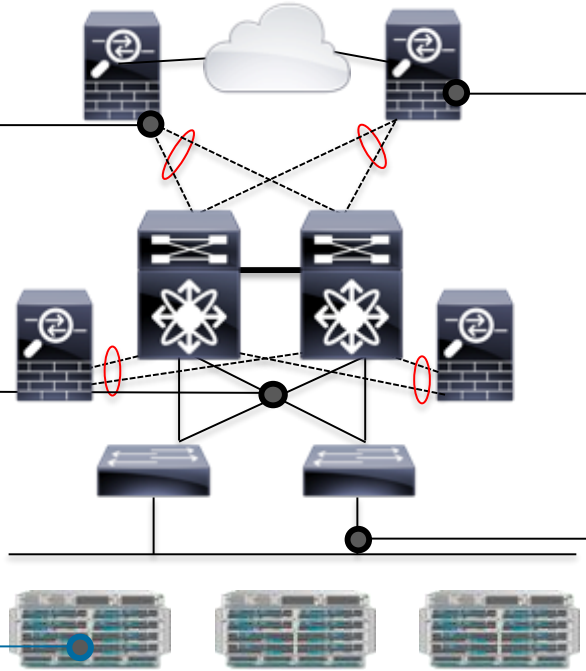
Hard to insert resources, power/port constraints, “fat” flows, expansion

## Low Versatility

Complex traffic engineering, VLAN/DRP, suboptimal paths

## Physical Network Limits Applications

Need intelligence to abstract application flows



## Policy Set Complication

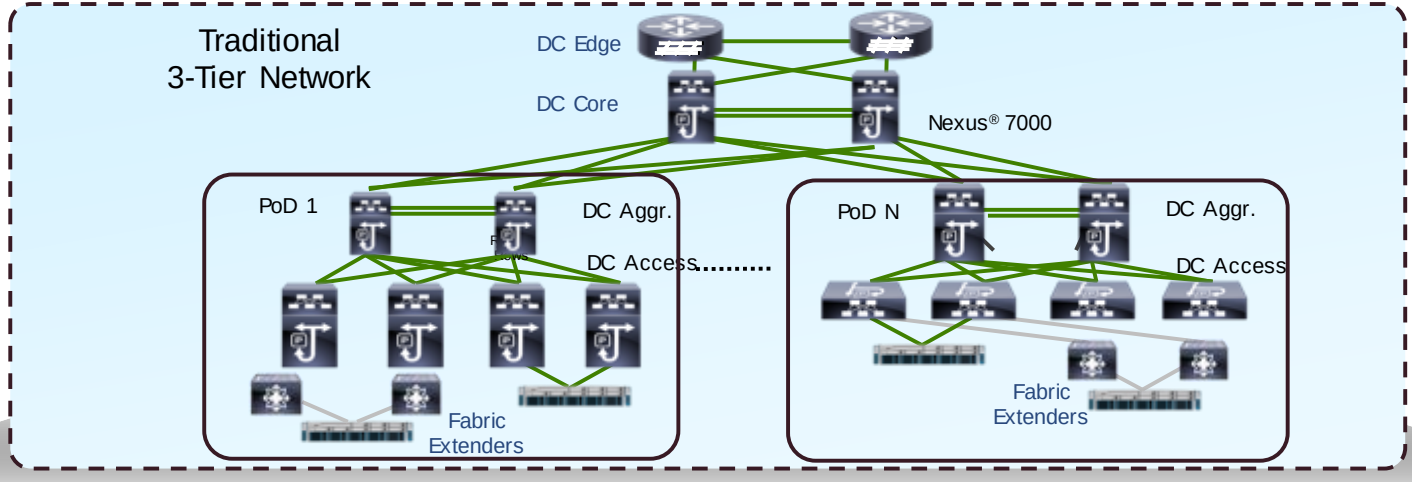
Overlapping rule sets, complex inheritance, oversubscription

## Cost of East-West Services

Multi-pass inspections, “slow” network traversal, “hairpinning”, waste of compute cycles

# Today's DC Challenges:

## Why Network Provisioning is Slow: Traditional 3-Tier Network Design



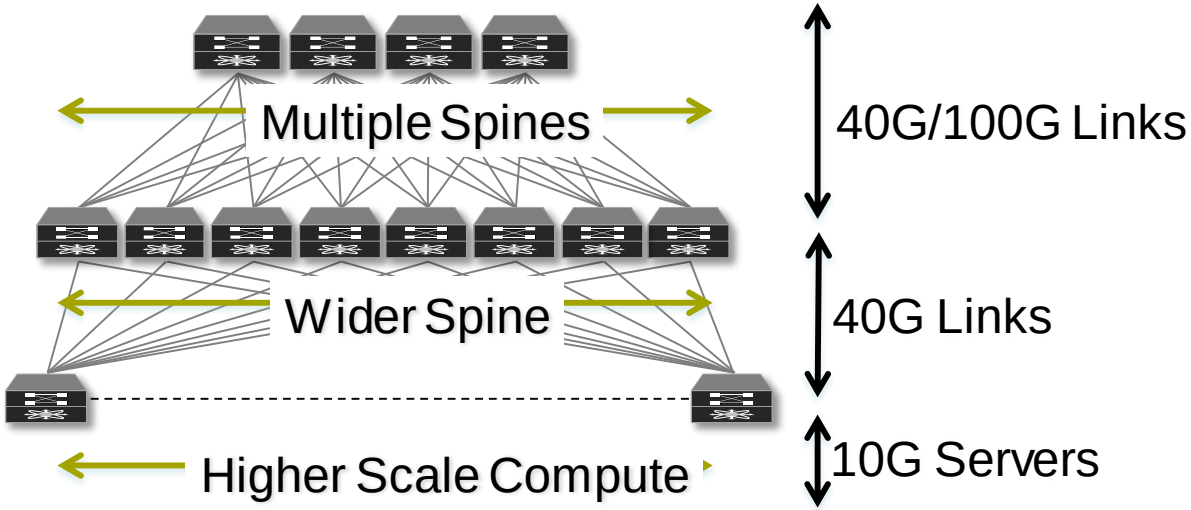

3-Tier DC Network Design





Cisco live!

# DC Fabric Trends

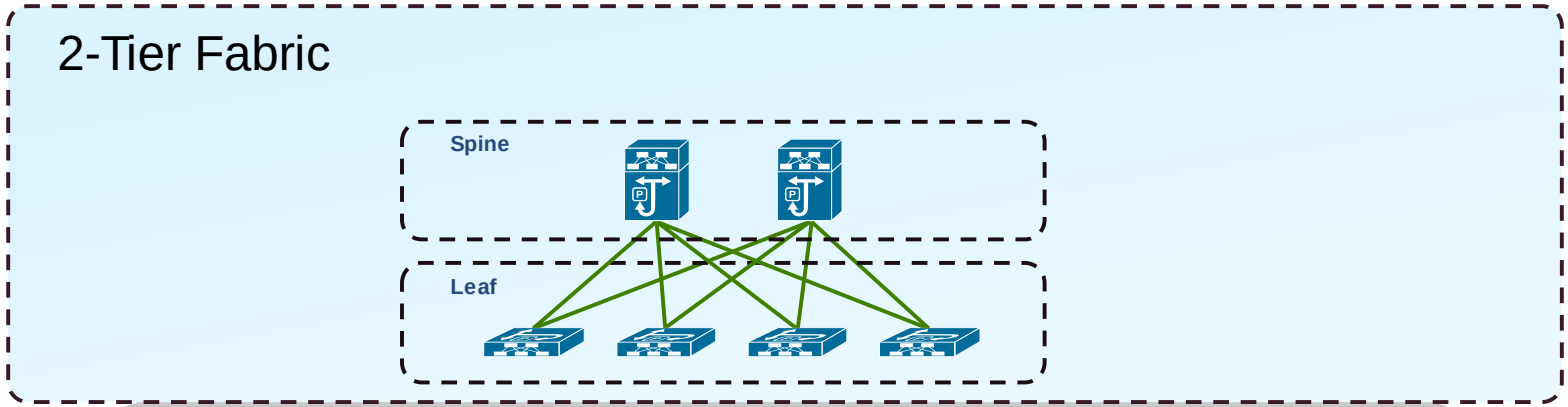



- Overall bandwidth capacity increases
- Routing
- Width of spines
- Redundancy Model
- Physical Infrastructure

Early Integration of 100G north-south focused – DC edge

# Solving Today's DC Challenges:

## Architecture: Spine-Leaf Fabric Design



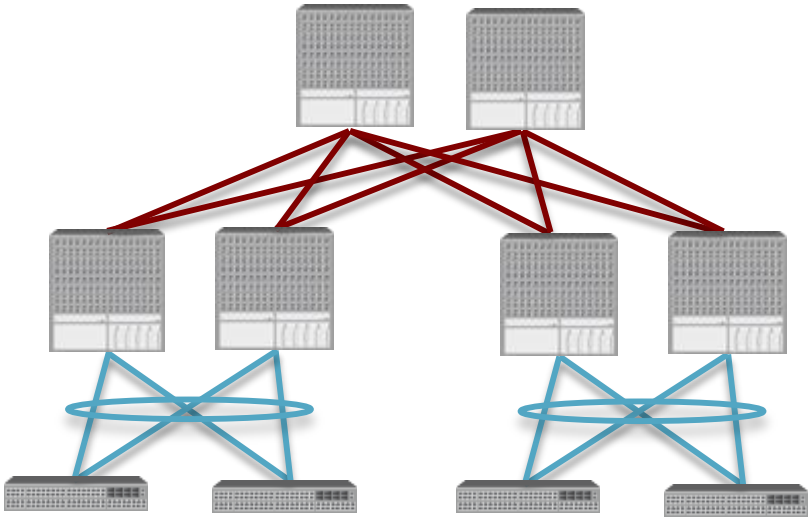

Spine-Leaf Fabric DC Network Design (Clos Fabric)



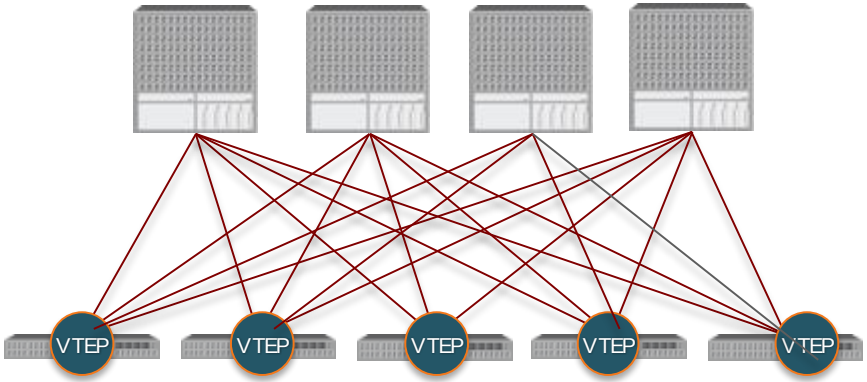


Cisco live!

# Three-Tier vs. Leaf Spine



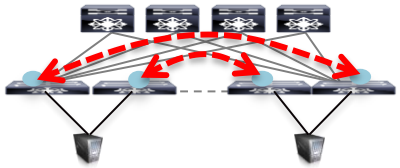
**Access and Aggregation**
**vPC**




**Routed Access (Leaf Spine) –**
**VxLAN**

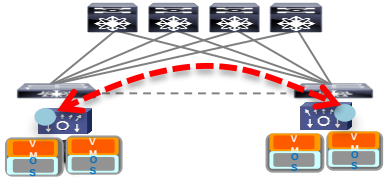
# Types of Overlay Edge Devices

## Network Overlays




- Router/switch end-points
- Protocols for resiliency/loops
- Traditional VPNs
- OTV, VXLAN, VPLS, LISP

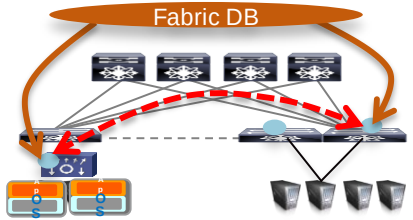
## Host Overlays




- Virtual end-points only
- Single admin domain
- VXLAN, NVGRE, STT

Tunnel End-points

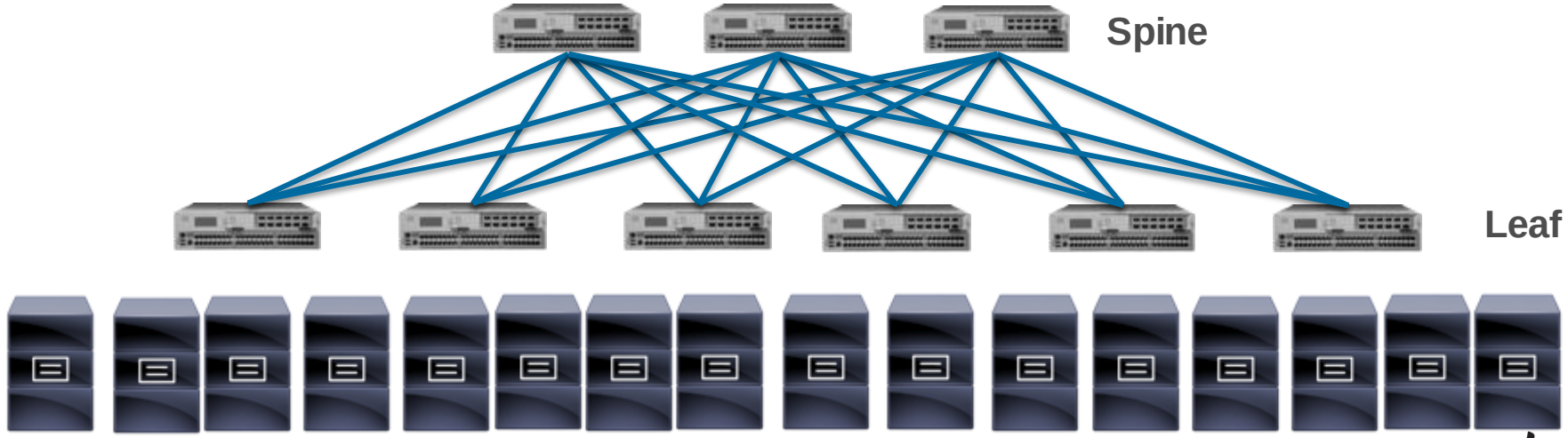
## Integrated Overlays




- Physical and Virtual
- Resiliency + Scale
- X-organisations/federation
- Open Standards

# Application Networking Infrastructure

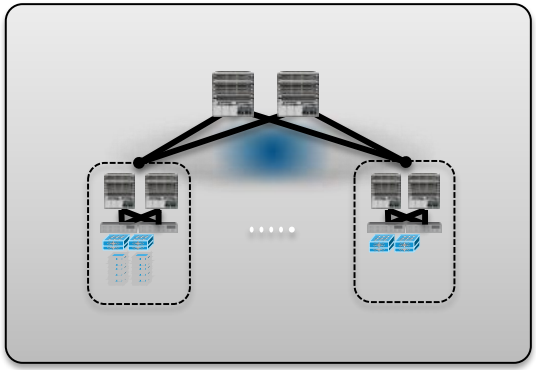
| | | | | |
|---|---|---|---|---|
| Automated | Deterministic Bandwidth | Deterministic Latency | Open & Secure | Highly Scalable |
| Policy Based | Physical & Virtual | Services Insertion | 40GbE Infrastructure | On Premise & Cloud |



# Modernising the Data Centre – P+V+C

## High Performance Secure Foundation – Open Programmable Interfaces

### EXISTING 2/3-TIER DESIGNS



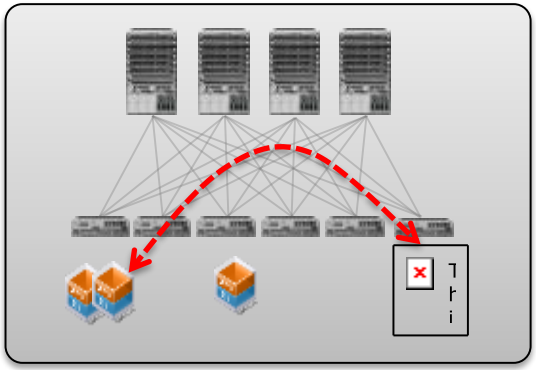
Modernised Operating System

Programmable Open APIs

Any 3rd Party Controller

Linux Containers

### PROGRAMMABLE SDN OVERLAY MODEL



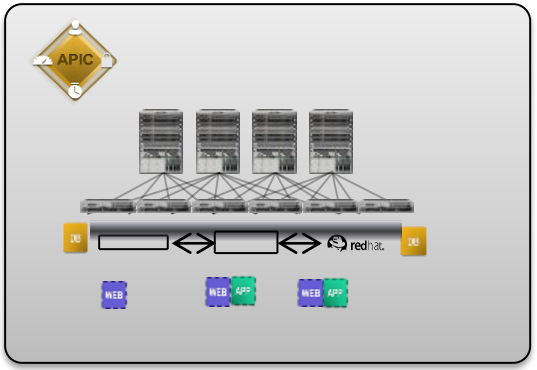
Integrated Network Virtualisation (no Gateways)

VXLAN Bridging, Routing, BGP

Linux Based DevOps

OPEN DAYLIGHT

### APPLICATION CENTRIC INFRASTRUCTURE




Any Hypervisor – No VM Tax

Physical & Virtual

Open API's & Controller

Group Based Policy Model





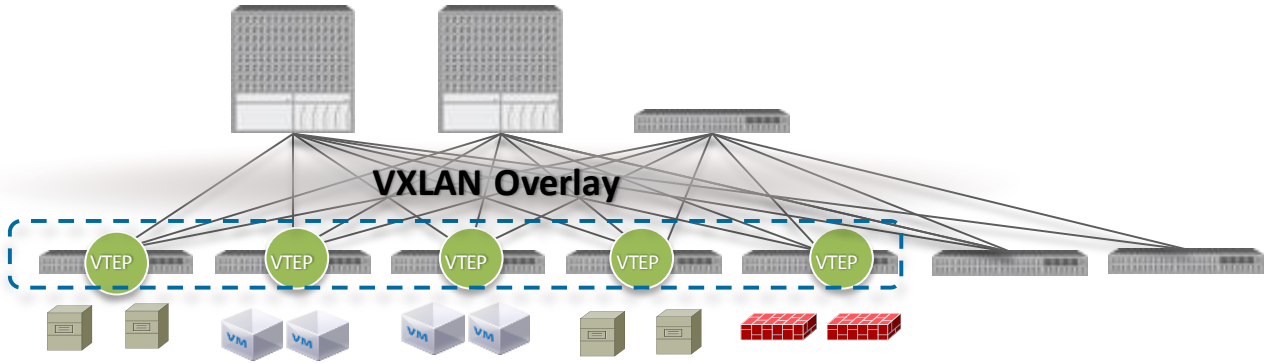
Cisco live!

# VXLAN Overview and Primer

Cisco live!

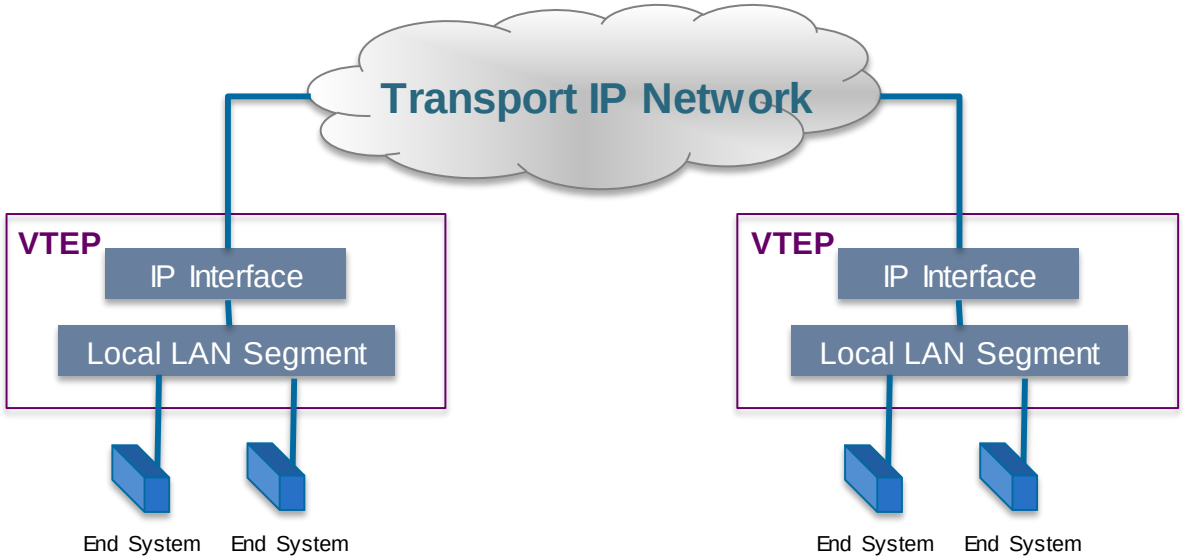
# Why VXLAN Overlay Was Created?

| Customer Needs | VXLAN Delivered |
|---|---|
| Any workload anywhere – VLANs limited by L3 boundaries | Any Workload anywhere- across Layer 3 boundaries |
| VM Mobility | Seamless VM Mobility |
| Scale above 4k Segments (VLAN limitation) | Scale up to 16M segments |
| Secure Multi-tenancy | Traffic & Address Isolation |



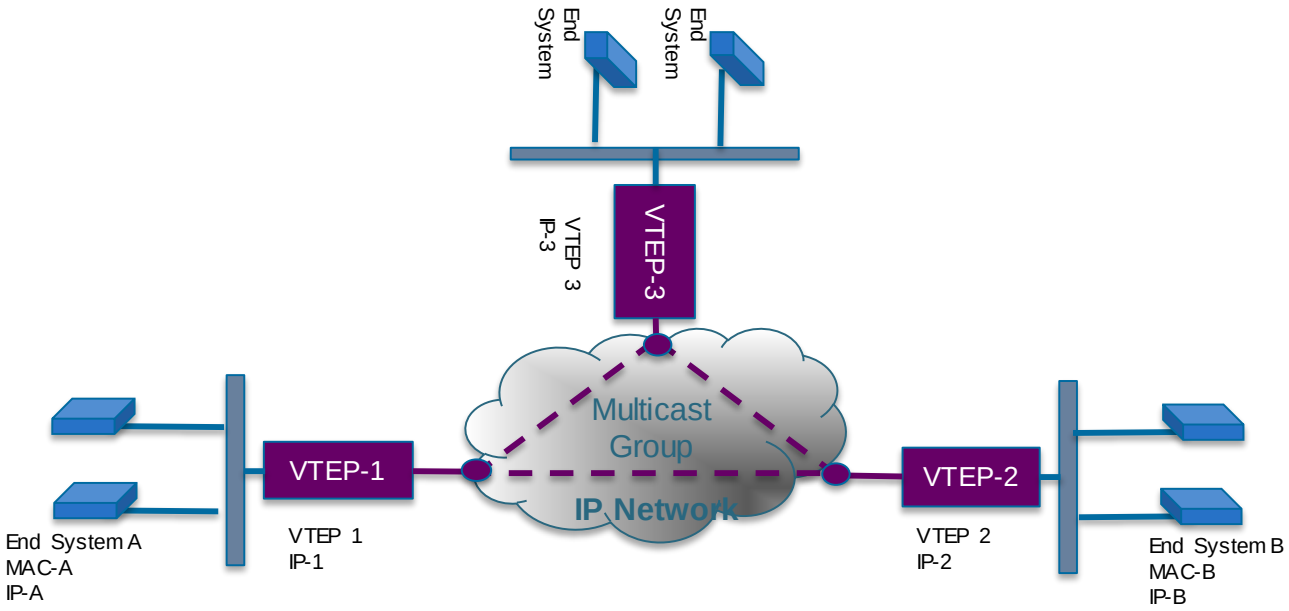
# VXLAN VTEP

VXLAN terminates its tunnels on VTEPs (Virtual Tunnel End Point). Each VTEP has two interfaces, one is to provide bridging function for local hosts, the other has an IP identification in the core network for VXLAN encapsulation/decapsulation.

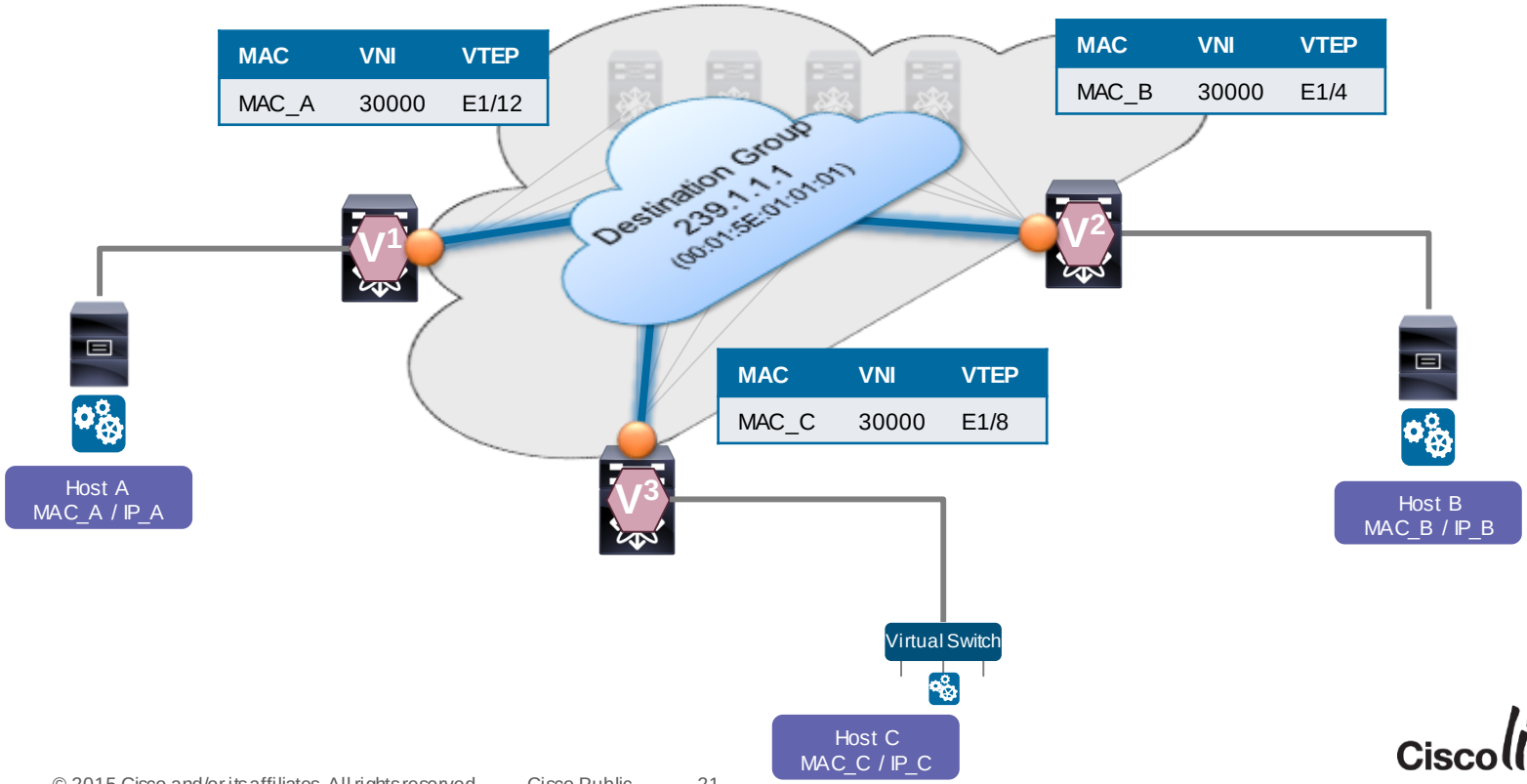


# VXLAN BUM Traffic over Transport Multicast

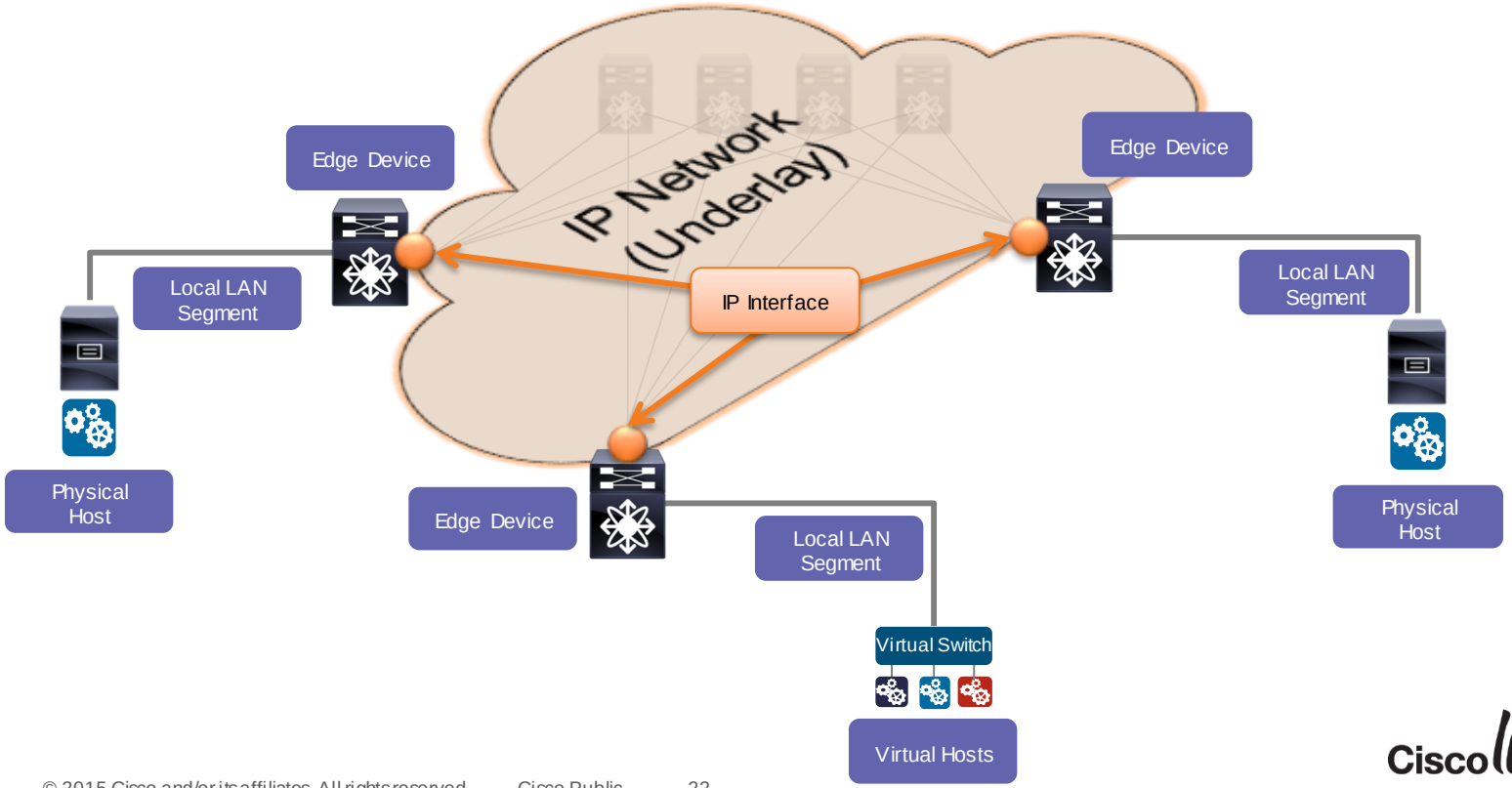
VXLAN BUM (Broadcast, Unknown Unicast and Multicast) traffic is transported over the VXLAN segment control multicast group.



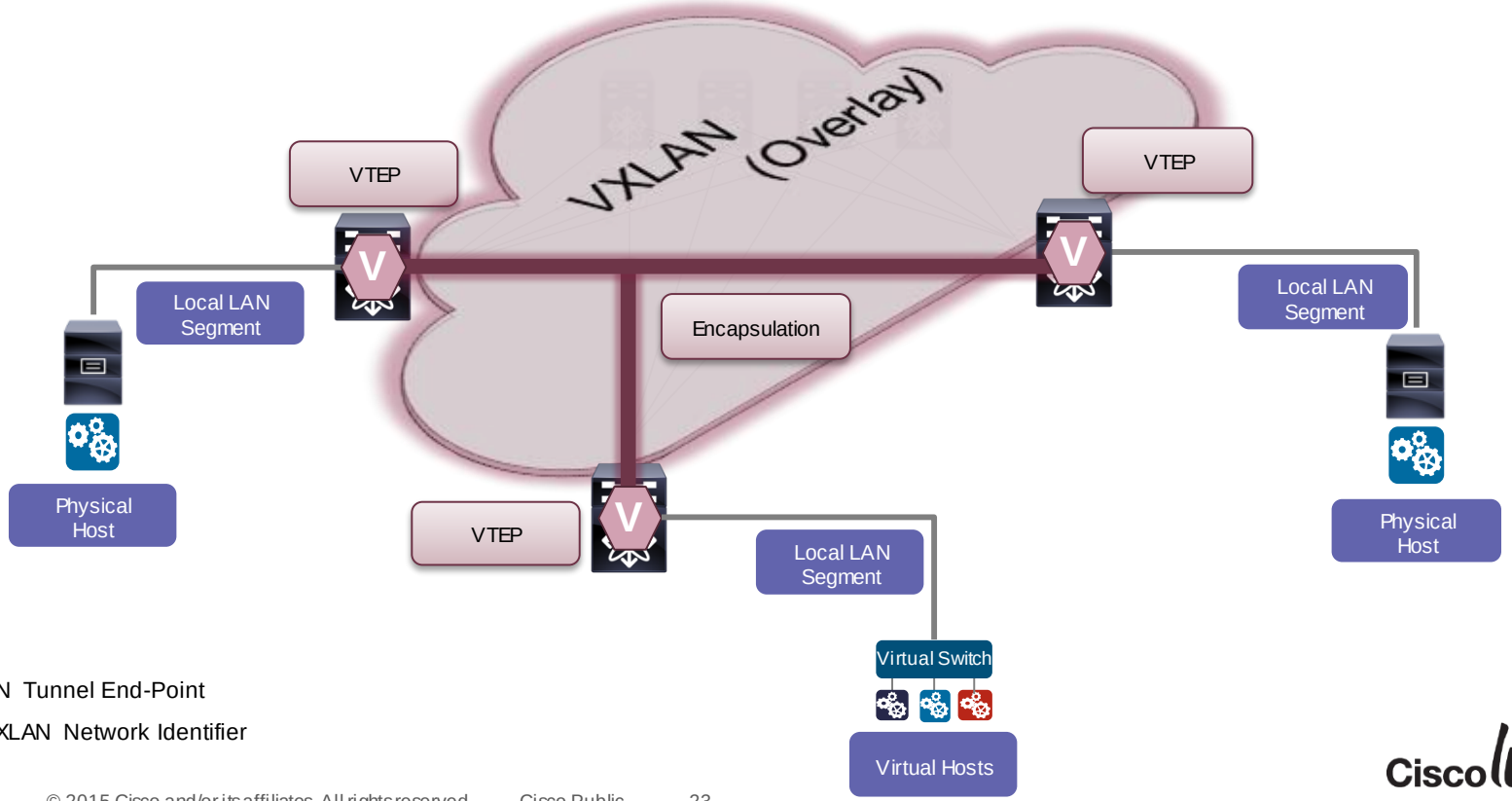
# VXLAN Flood and Learn



# VXLAN Taxonomy (1)



# VXLAN Taxonomy (2)




VTEP – VXLAN Tunnel End-Point

VNI/VNID – VXLAN Network Identifier



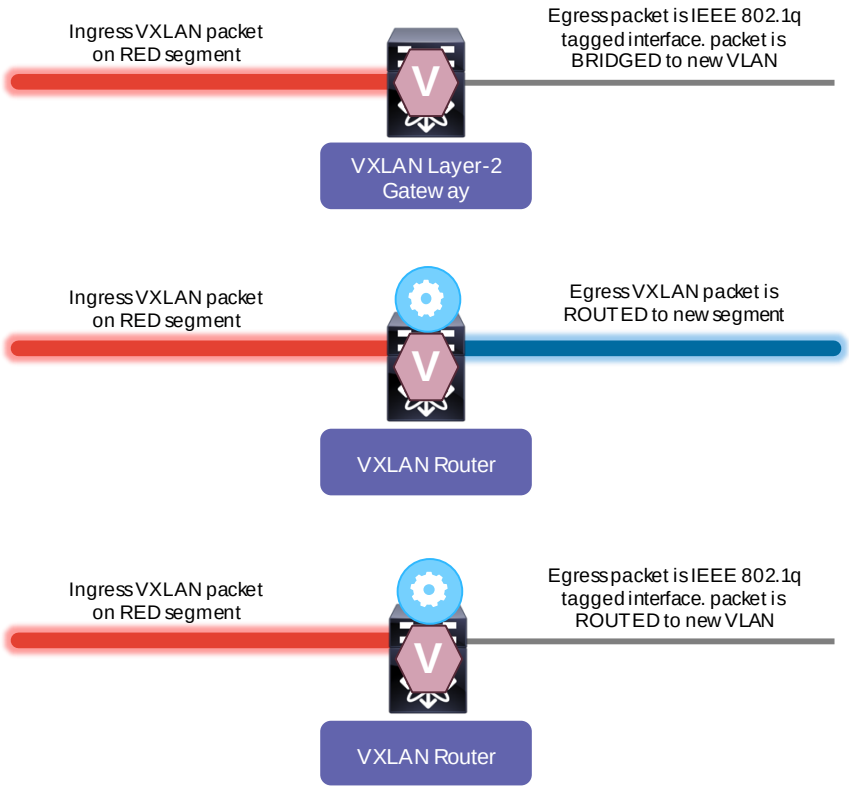


Cisco live!

# VXLAN Gateway Types

## VXLAN Taxonomy

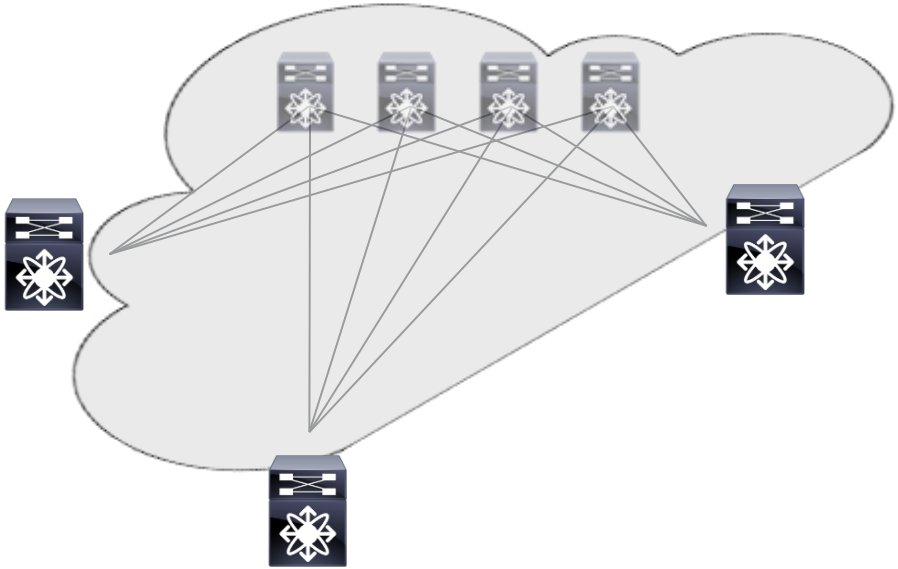
- VXLAN to VLAN Bridging
  - (Layer-2 Gateway)
- VXLAN-to-VXLAN Routing
  - (Layer-3 Gateway)
- VXLAN-to-VLAN Routing
  - (Layer-3 Gateway)



# VXLAN Deployment Considerations

## Underlay

- MTU and Overlays
- Unicast Routing Protocol and IP Addressing
- Multicast for BUM* Traffic Replication



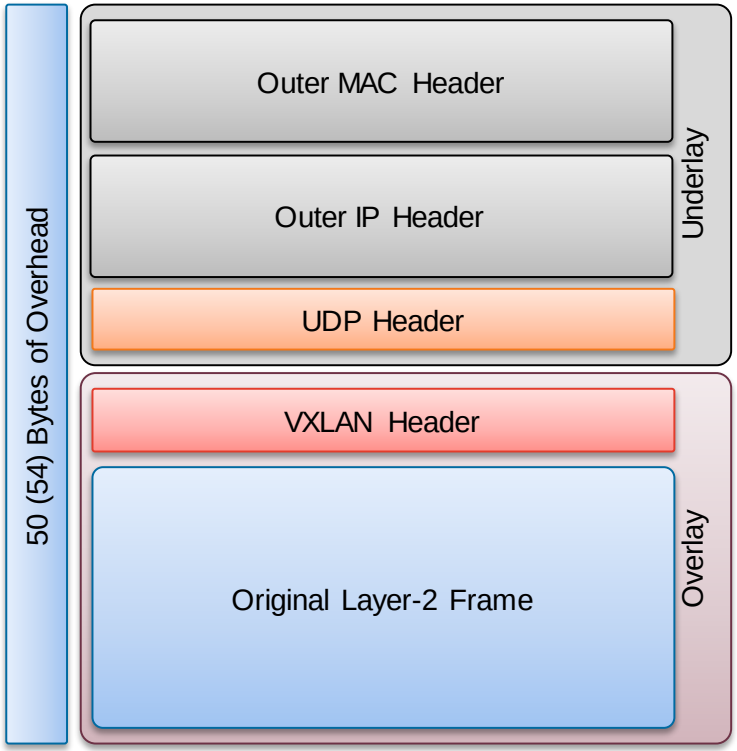




Cisco live!

# MTU and VXLAN

## Underlay



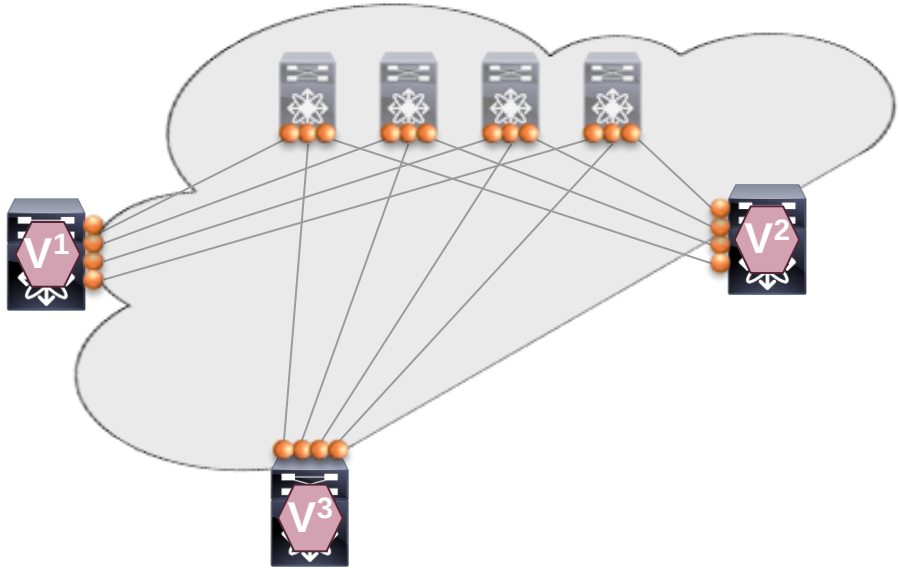

- VXLAN adds 50 Bytes to the Original Ethernet Frame
- Avoid Fragmentation by adjusting the IP Networks MTU
- Data Centres often require Jumbo MTU; most Server NIC do support up to 9000 Bytes
- Using a MTU of 9216* Bytes accommodates VXLAN Overhead plus Server max. MTU
- No Fragmentation Needed

*Cisco Nexus 5600/6000 switches only support 9192 Byte for Layer-3 Traffic

# Building Your IP Network – Interface Principles

## Underlay

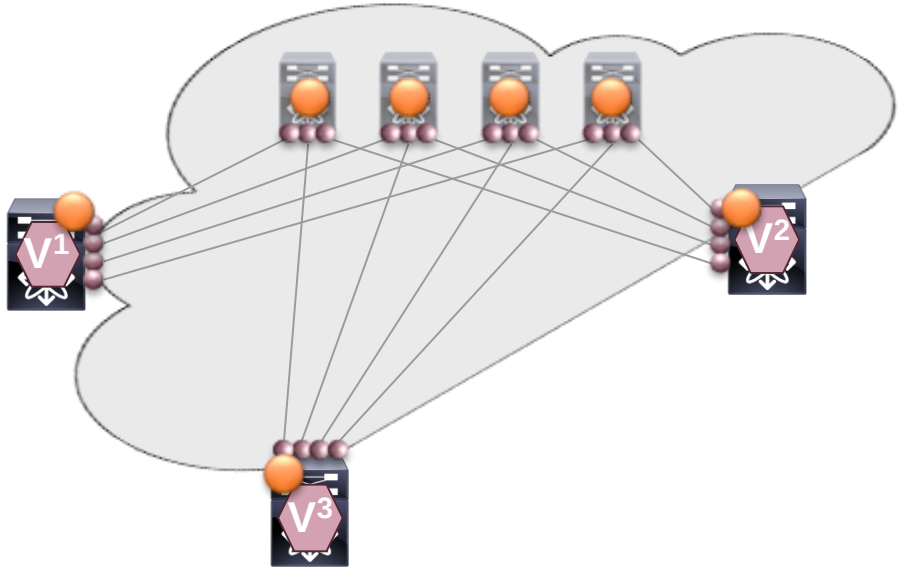
- Know your IP addressing and IP scale requirements
  - Best to use single Aggregate for all Underlay Links and Loopbacks
  - IPv4 only
  - For each Point-2-Point (P2P) connection, minimum /31 required
  - Loopback requires /32
- Routed Ports/Interfaces
  - Layer-3 Interfaces between Spine and Leaf (no switchport)
- VTEP uses Loopback as Source-Interface



# IP Unnumbered – Simplifying the Principles

## Underlay

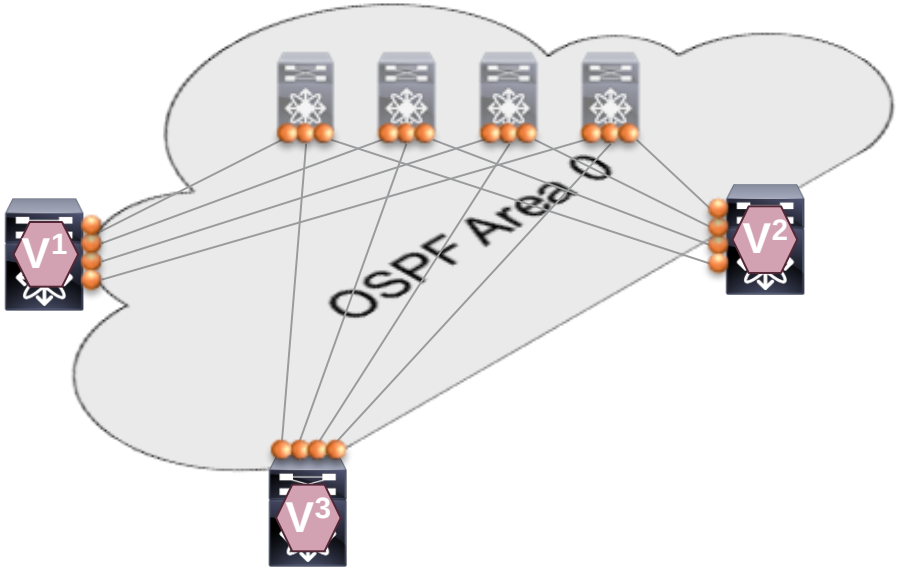
- **IP Unnumbered** – Single IP Address for multiple Interfaces
- Well-Known from Serial Interfaces (back in time)
- Used for Layer-3 Interfaces between Spine and Leaf (no switchport)
- For each Switch in the fabric, **single IP address** is sufficient
  - Loopback for VTEP
  - IP Unnumbered from Loopback for routed Interfaces




Check Platform & Release Support for Ethernet IP Unnumbered

# Building IP Network – Routing Protocols: OSPF
## Underlay

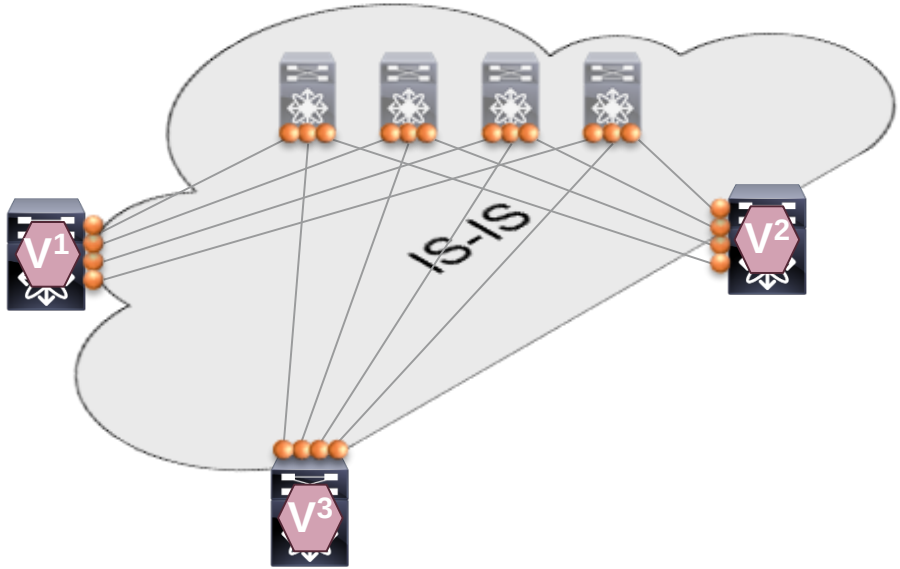
- OSPF – watch your Network type
  - Network Type Point-2-Point (P2P)
    - Preferred (only LSA type-1)
    - No DR/BDR election
    - Suits well for routed interfaces/ports (optimal from a LSA Database perspective)
    - Full SPF calculation on Link Change
  - Network Type Broadcast
    - Suboptimal from a LSA Database perspective (LSA type-1 & 2)
    - DR/BDR election
    - Additional election and Database Overhead



# Building IP Network – Routing Protocols: IS-IS

## Underlay

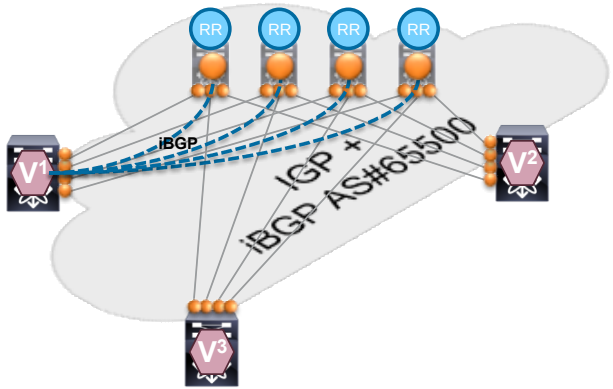
- IS-IS – what was this CLNS?
  - Independent of IP (CLNS)
  - Well suited for routed interfaces/ports
  - No SPF calculation on Link change; only if Topology changes
  - Fast Re-convergence
  - Not everyone is familiar with it



# Building IP Network – Routing Protocols: iBGP

## Underlay

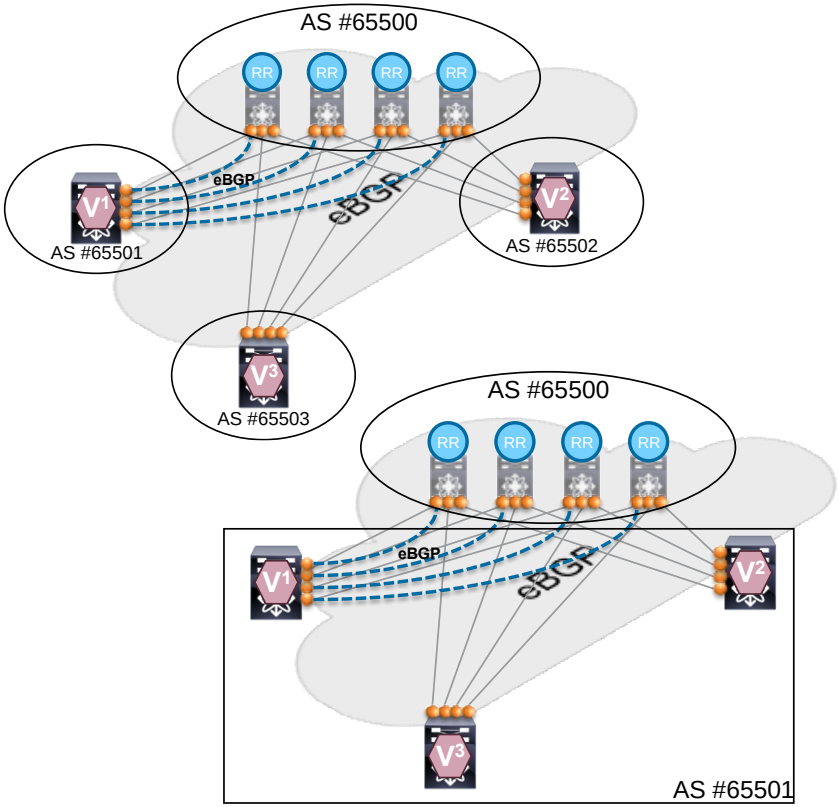
- iBGP + IGP = The Routing Protocol Combo
  - IGP for underlay topology & reachability (e.g. IS-IS, OSPF)
  - iBGP for VTEP (loopback) reachability
  - iBGP route-reflector for simplification and scale
  - Requires two routing protocols



# Building IP Network – Routing Protocols: eBGP

## Underlay

- eBGP
  - eBGP Peer is IP interface
    - Loopback would require additional IGP and eBGP multi-hop
  - Multiple Autonomous-Systems (AS)
    - Minimum amount of AS is two
  - Many BGP Neighbours
    - For each neighbouring p2p interface
  - AS Path
    - Src and Dst AS might be same
  - No Route-Reflector
    - But next-hop needs to be unchanged



# Multicast Routing
## Underlay

May use PIM-ASM or PIM-BiDir (Different hardware has different capabilities)

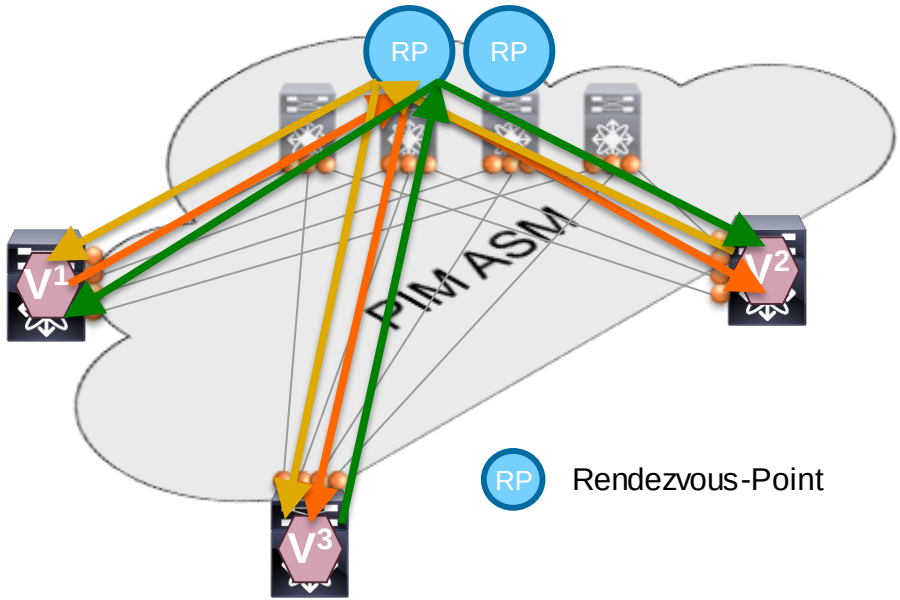
| | Nexus 1000v | Nexus 3000 | Nexus 5600 | Nexus 7000/F3 | Nexus 9000 | ASR 1000<br>CSR 1000 | ASR 9000 |
|---|---|---|---|---|---|---|---|
| **Multicast Mode** | IGMP v2/v3 | PIM ASM | PIM BiDir | PIM ASM / PIM BiDir | PIM ASM | PIM BiDir | PIM ASM / PIM BiDir |

- Spine and Aggregation Switches make good Rendezvous-Point (RP) Locations in Topologies
- Reserve a range of Multicast Groups (Destination Groups/DGroups) to service the Overlay and optimise for diverse VNIs
- In Spine/Leaf topologies with lean Spine
  - Use multiple Rendezvous-Point across the multiple Spines
  - Map different VNIs to different Rendezvous-Point for simple load balancing measure
  - Use Redundant Rendezvous-Pint
- Design a Multicast Underlay for a Network Overlay, Host VTEPs will leverage this Network

# Multicast Enabled Underlay: PIM ASM

## Underlay

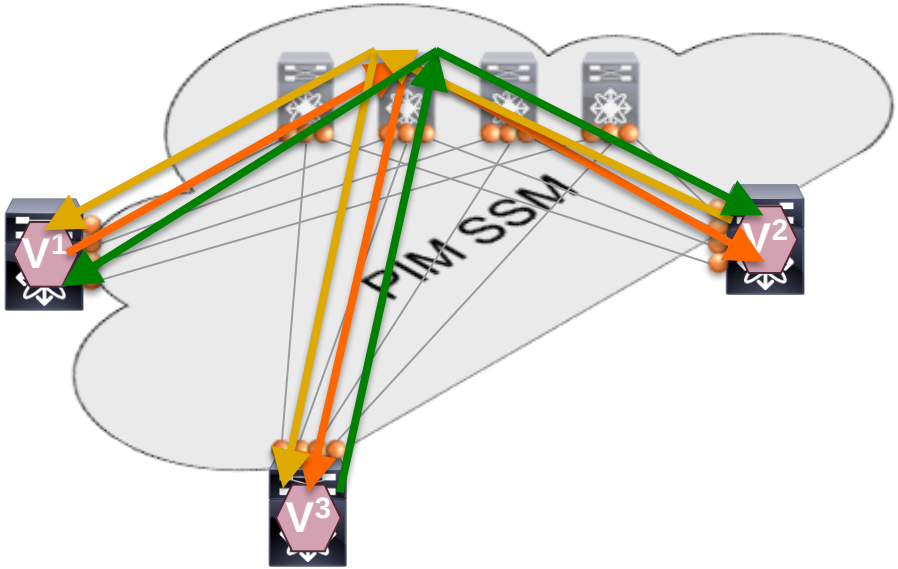
- PIM Sparse-Mode (ASM)
- Redundant Rendezvous-Point using PIM Anycast-RP or MSDP
- Source-Tree or Unidirectional Shared-Tree (Source-Tree shown)
  - Shared-Tree will always use RP for forwarding
- 1 Source-Tree per Multicast-Group per VTEP (each VTEP is Source & Receiver)
- Example from depicted topology
  - 3 VTEPs sharing same VNI and Multicast-Group mapping (single Multicast-Group)
  - 3x Source-Tree (1 per VTEP per Multicast-Group)



# Multicast Enabled Underlay: PIM SSM

## Underlay

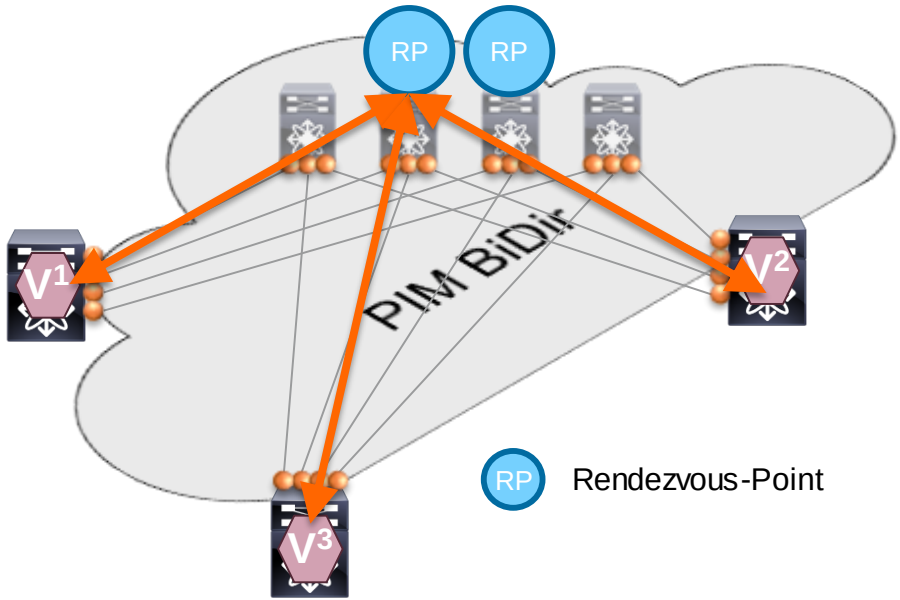
- PIM Source Specific Multicast (SSM)
- No Rendezvous-Point required
- Source-Tree or Unidirectional Shared-Tree
- 1 Source-Tree per Multicast-Group per VTEP (each VTEP is Source & Receiver)
- Example from showed Topology
  - 3 VTEPs sharing same VNI and Multicast-Group mapping (single Multicast-Group)
  - 3x Source-Tree (1 per VTEP per Multicast-Group)



# Multicast Enabled Underlay: PIM BiDir

## Underlay

- Bidirectional PIM (BiDir)
- Redundant Rendezvous-Point using Phantom-RP
- Building Bi-Directional Shared-Tree
  - Uses shortest path between Source and Receiver with RP as routing-vector
- 1 Shared-Tree per Multicast-Group
- Example from depicted topology
  - 3 VTEPs sharing same VNI and Multicast-Group mapping (single Multicast-Group)
  - 1x Shared-Tree (1 per Multicast-Group)



# VXLAN Design Considerations to Remember

## Multicast Enabled Underlay

- Multi-Destination Traffic (Broadcast, Unknown Unicast, etc.) needs to be replicated to ALL VTEPs serving a given VNI
  - Each VTEP is Multicast Source & Receiver
- For a given VNI, all VTEPs act as a Sender and a Receiver
- Head-End Replication will depend on hardware scale/capability
- Resilient, efficient, and scalable Multicast Forwarding is highly desirable
  - Choose the right Multicast Routing Protocol for your need (type/mode)
  - Use redundant Multicast Rendezvous Points (Spine/Aggregation generally preferred)
  - 99% percent of Overlay problems are in the Underlay (OTV experience)

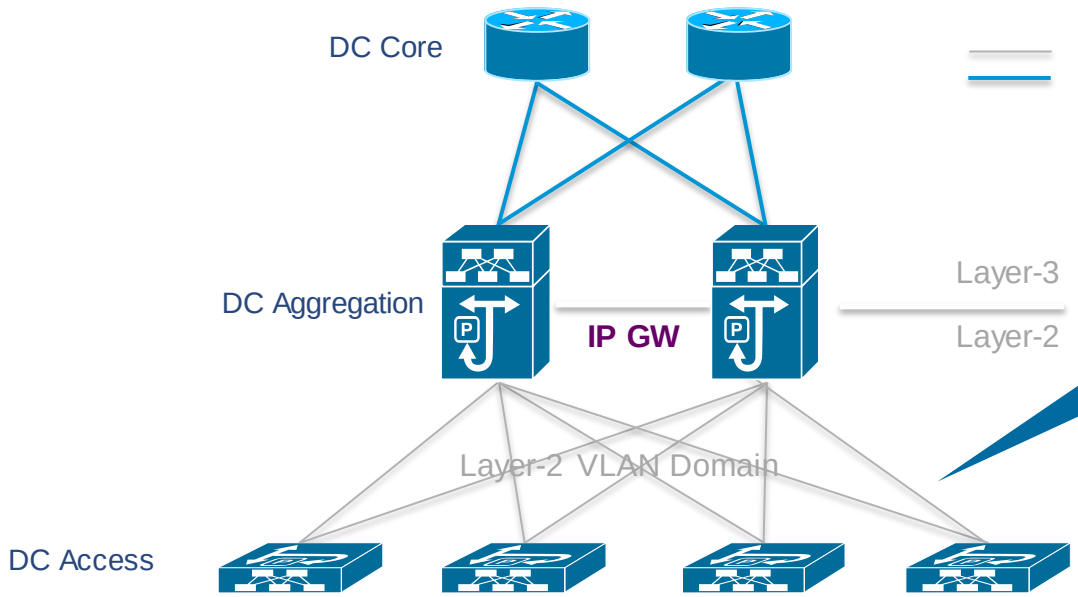
**Keep in Mind**

**Overlay Convergence = Underlay Convergence!**

# VXLAN Use Cases

## L2 STP/VPC Replacement – Routing Off-Box

### Traditional Layer-2 POD Design

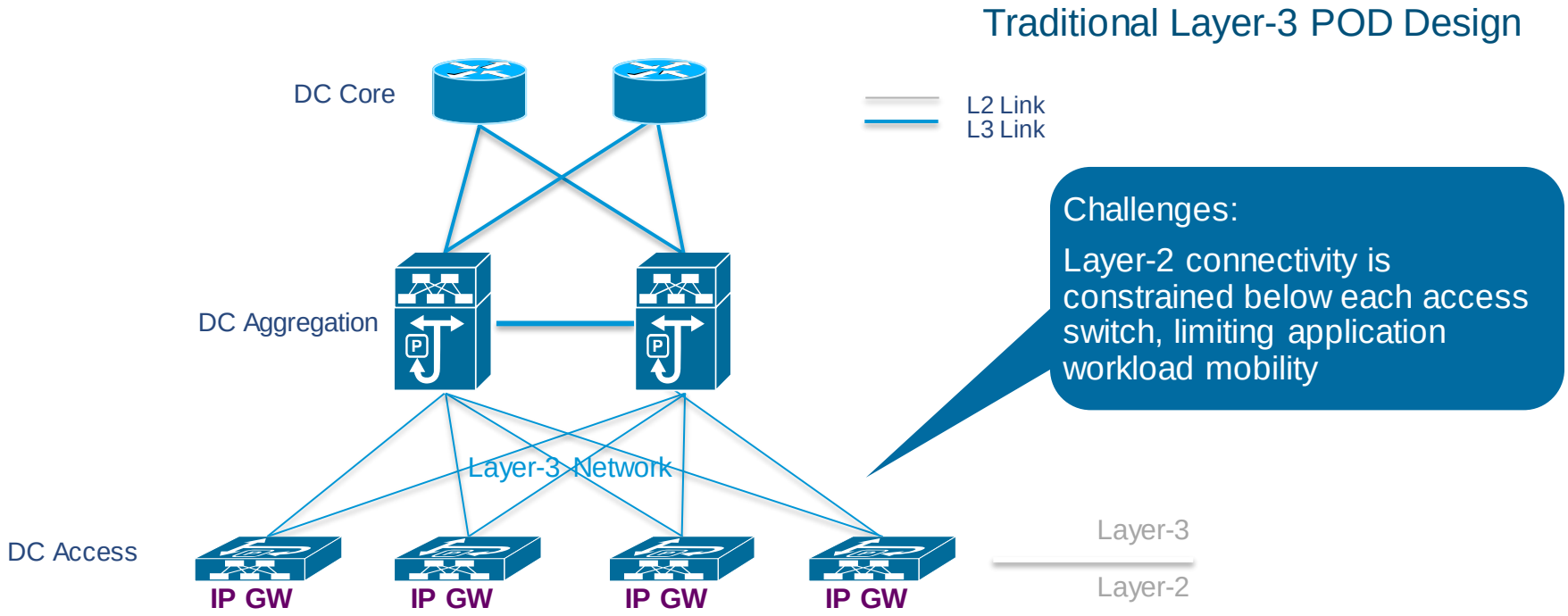



Challenges:

Hard to Scale --- Layer-2 fault domain size grows as the POD grows

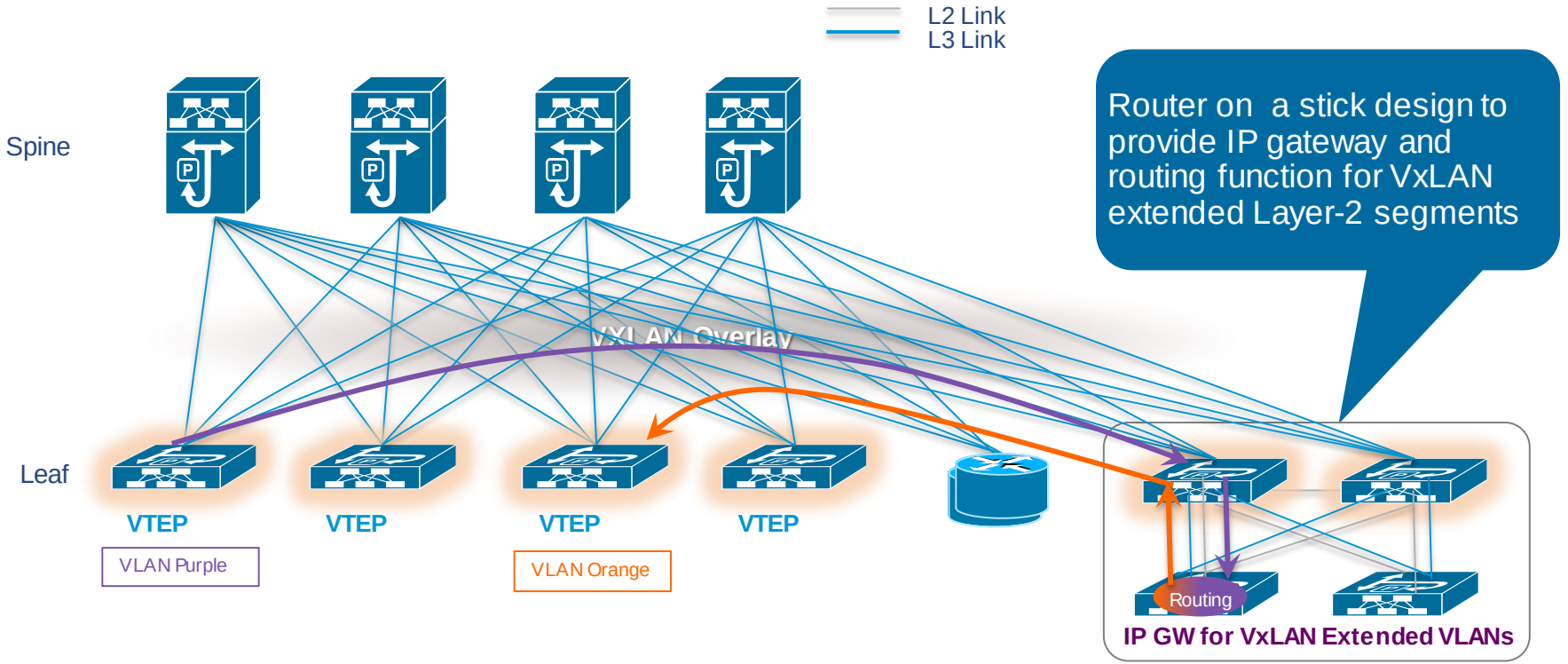
# VXLAN Use Cases

## L3 Routed Access Replacement – Routing Off-Box



# VXLAN Design - VXLAN Bridging

## Spine-Leaf Deployment: Router on a Stick Design with Routing Block



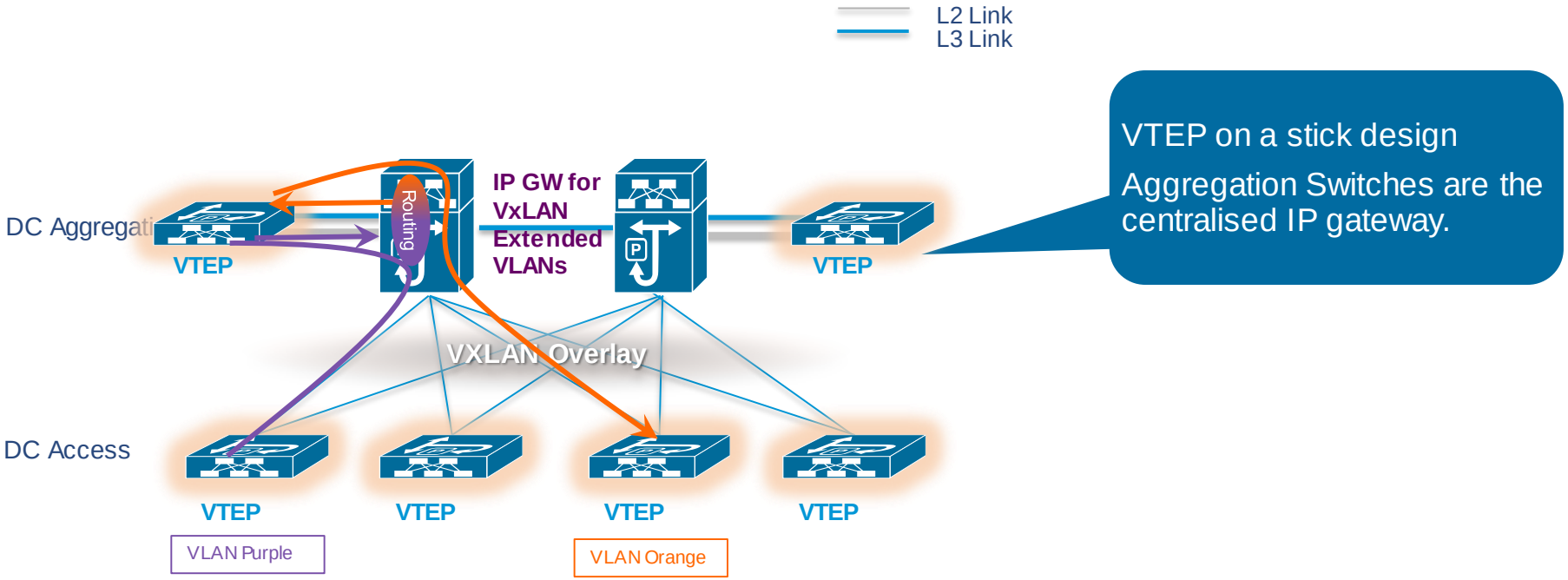





Cisco live!

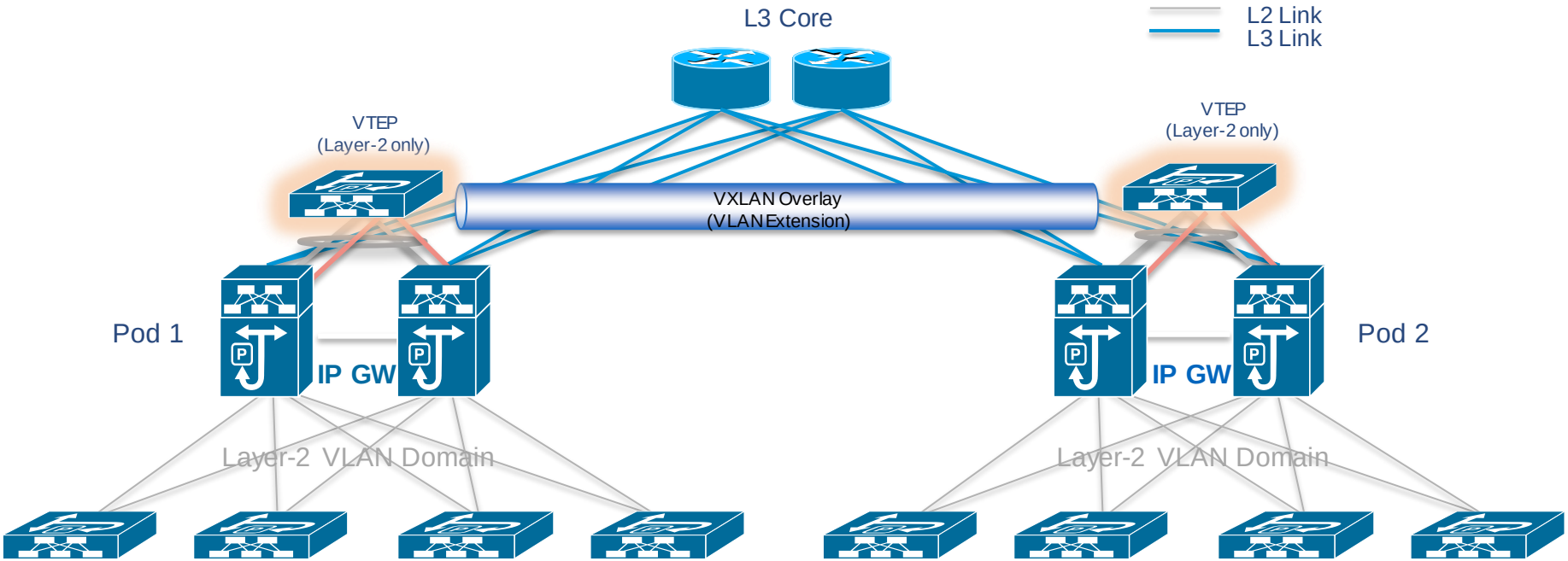
# VXLAN Design - VXLAN Bridging

## Spine Leaf Deployment: VTEP on a Stick Design with IP GW on Aggregation



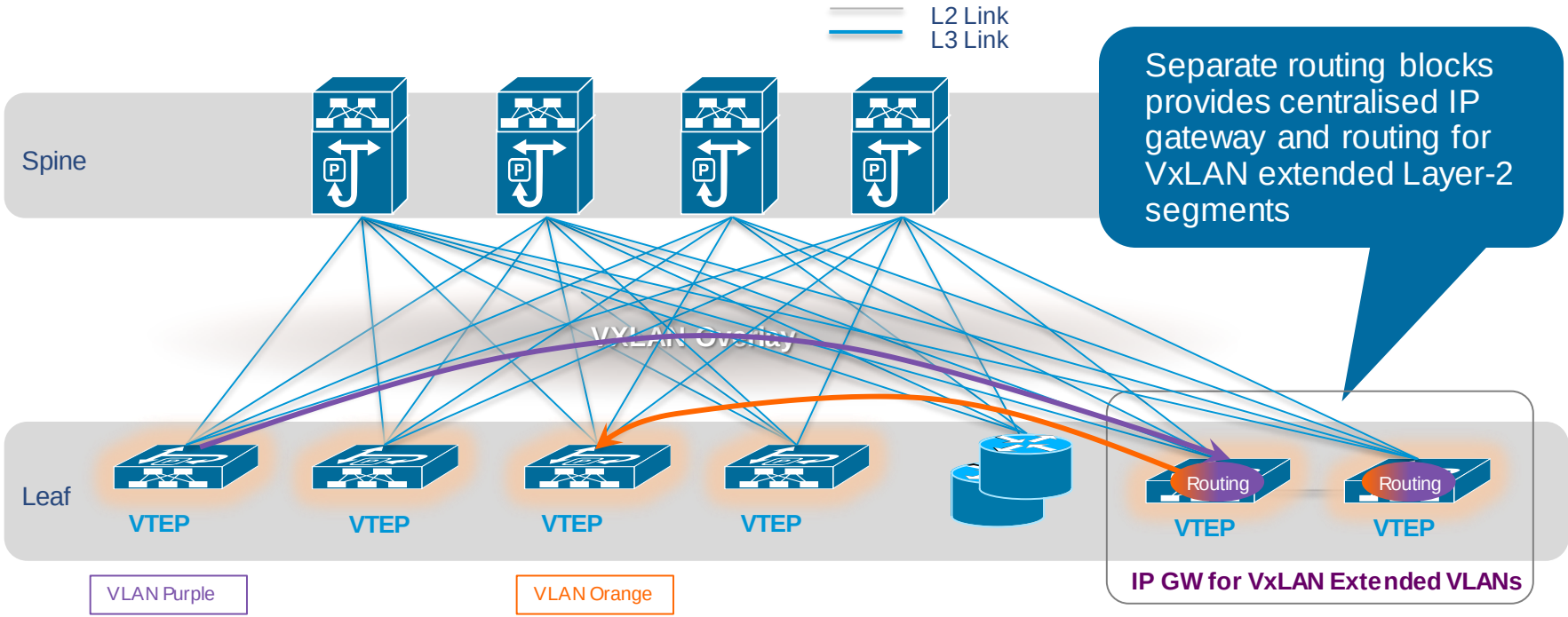
# VXLAN Design VXLAN Bridging

## Data Centre Interconnect (DCI): L2 Extension Across Pods



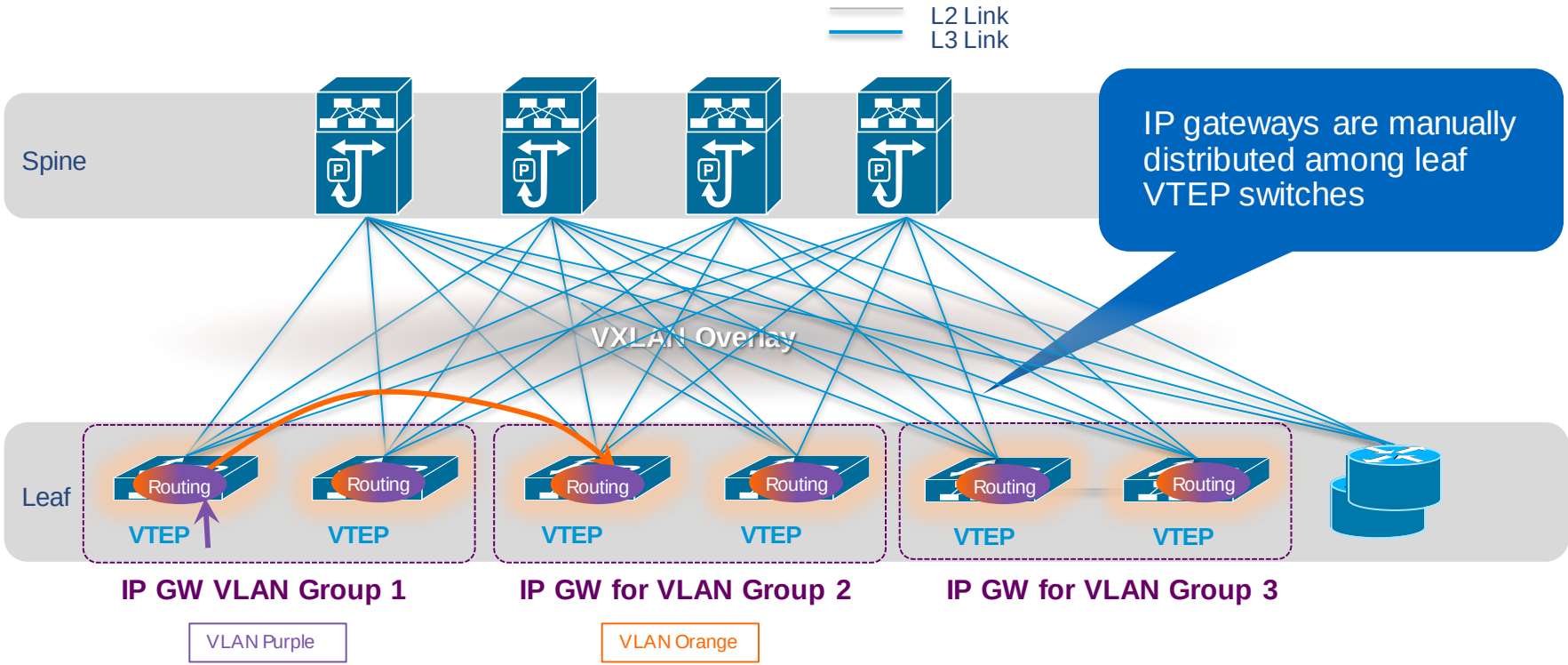
# VXLAN Design - VXLAN Bridging+Routing

## Spine-Leaf Deployment: Centralised IP Gateway



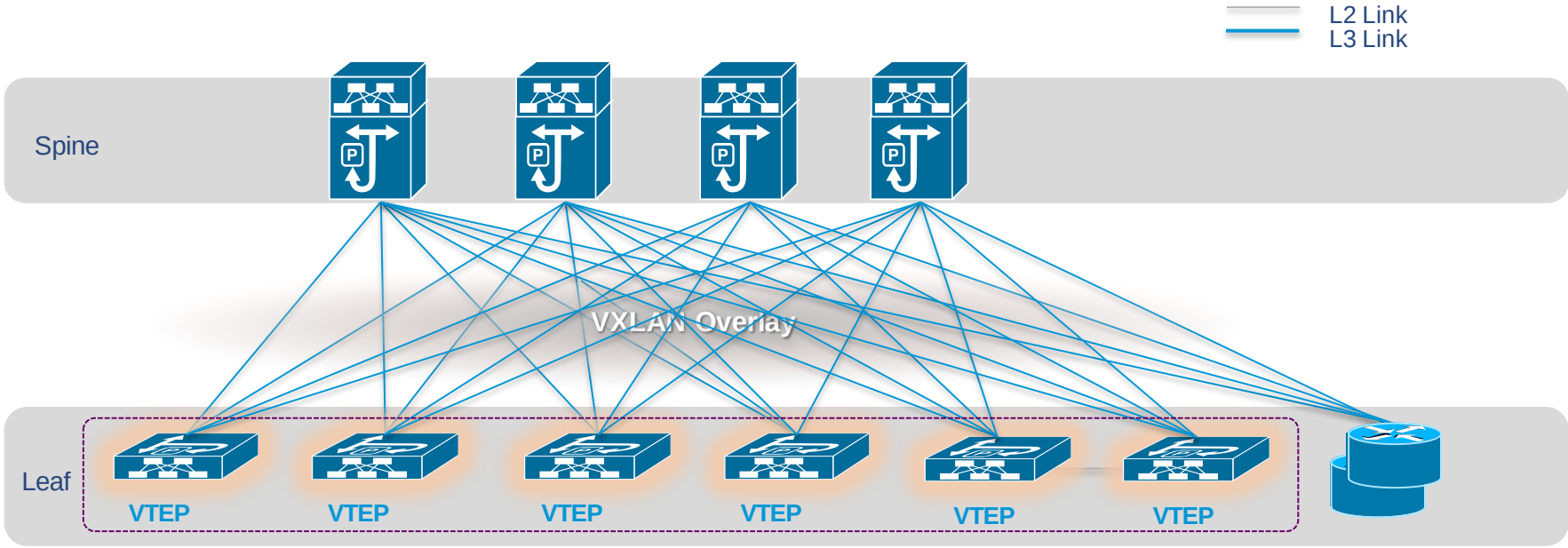
# VXLAN Design - VXLAN Bridging+Routing

## Spine-Leaf Deployment: Distributed IP Gateway



# VXLAN Design - VXLAN Bridging+Routing+Anycast GW

## Spine-Leaf Deployment: Distributed IP Gateway




**IP GW for ALL VxLAN Extended VLANs**

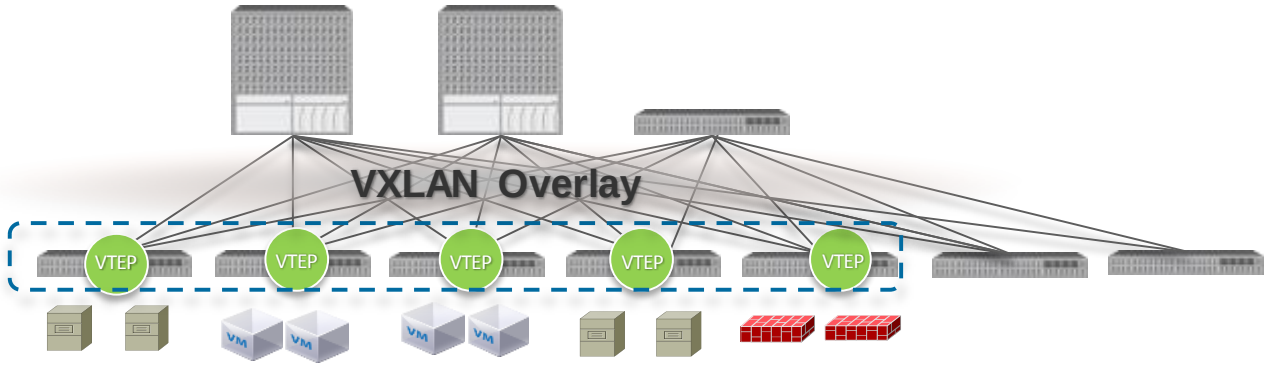




Cisco live!

# Challenges with Traditional VXLAN Deployments

## Scale and Mobility Limitations




**LIMITED SCALE**

Flood and learn (BUM)- Inefficient Bandwidth Utilisation

Resource Intensive – Large MAC Tables

**LIMITED WORKLOAD MOBILITY**

Centralised Gateways – Traffic Hair-pining

Sub-Optimal Traffic Flow

Barrier for Scaling out Large Data Centres and Cloud Deployments

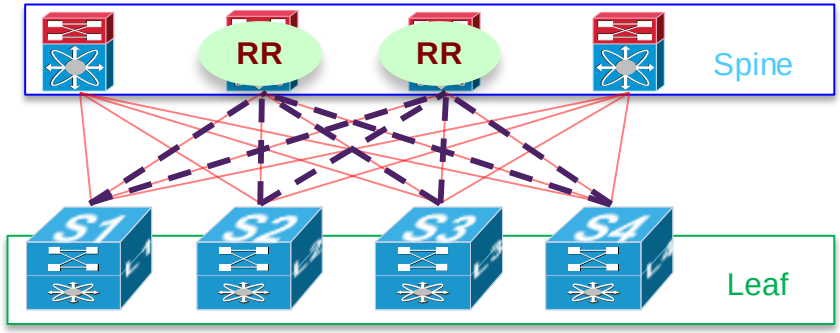




Cisco live!

# VXLAN BGP Control Plane

## Host and Subnet Route Distribution




iBGP Adjacencies

RR

Route-Reflectors deployed for scaling purposes

- Use MP-BGP with EVPN Address Family on the leaf nodes to distribute internal host/subnet routes and external reachability information
- MP-BGP also used to distribute IP multicast groups information
- MP-BGP enhancements to carry up to 100s of thousands of routes and reduce convergence time

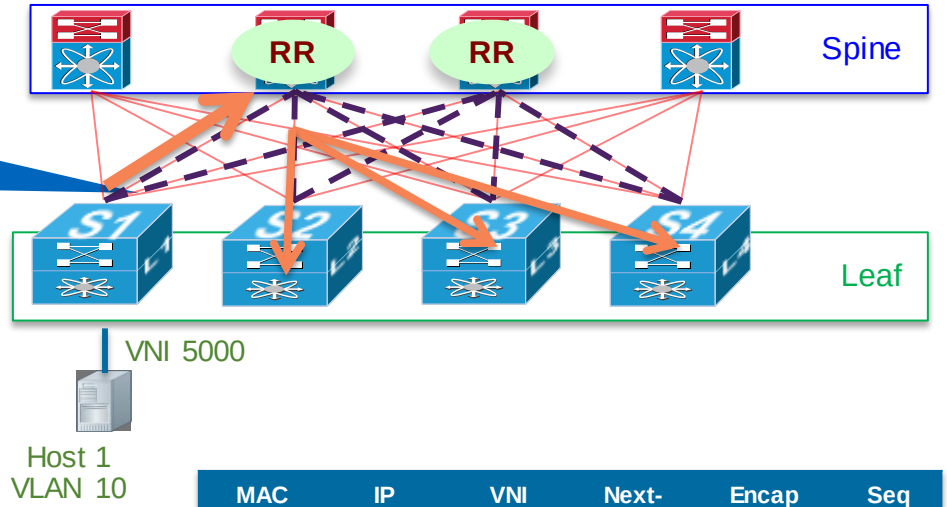
References: A Network Virtualisation Overlay Solution using EVPN (draft-sajassi-nvo3-evpn-overlay-01)

# VXLAN BGP Control Plane

## Host Advertisement

NLRI:
- Host MAC1, IP1
- NVE IP 1
- VNI 5000

Ext.Community:
- Encapsulation: VXLAN, NVGRE
- Cost/Sequence




| MAC | IP | VNI | Next-Hop | Encap | Seq |
|---|---|---|---|---|---|
| 1 | 1 | 5000 | IP1 | VXLAN | 0 |

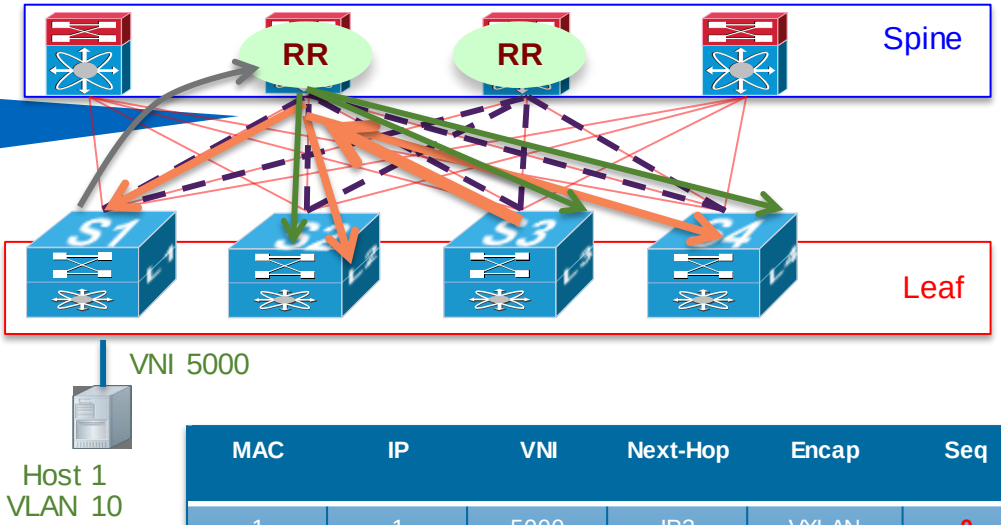
1. Host Attaches
2. Attachment VTEP advertises host's MAC (+IP) through BGP RR

# VXLAN BGP Control Plane

## Host Moves

NLRI:
- Host MAC1, IP3
- NVE IP 1
- VNI 5000

Ext.Community:
- Encapsulation: VXLAN, NVGRE
- Cost/Sequence 1

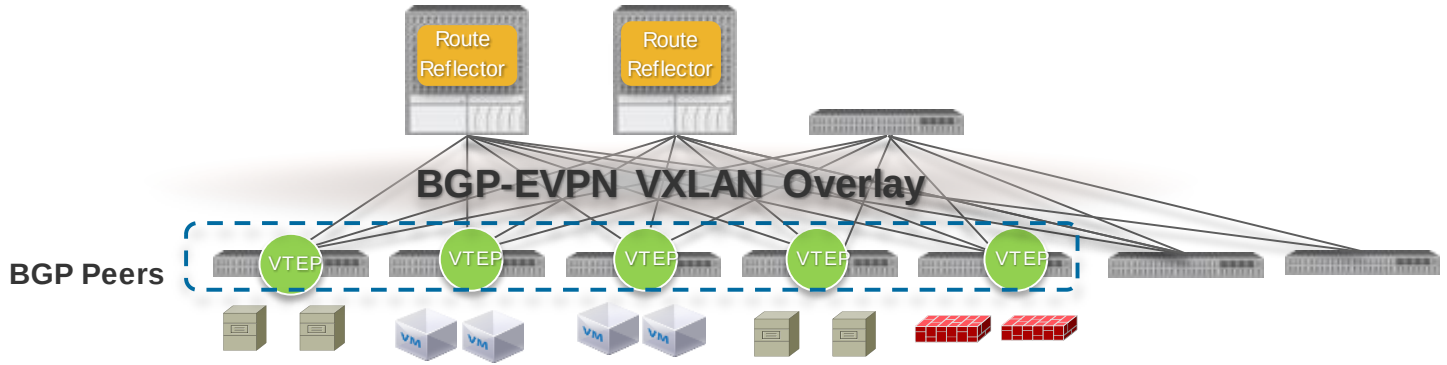



| MAC | IP | VNI | Next-Hop | Encap | Seq |
|---|---|---|---|---|---|
| 1 | 1 | 5000 | IP3 | VXLAN | 0 |

1. Host Moves behind switch S3
2. VTEP-3 (S3) detects Host1 and advertises H1 with seq #1
3. VTEP-1 (S1) sees more recent route and withdraws its advertisement

# Next Gen VXLAN Fabric w/BGP E-VPN Control Plane

## Delivering Multi-Tenancy and Seamless Host Mobility at Cloud Scale




**INTEROPERABLE**
Standards Based
BGP-EVPN
VXLAN

**INCREASED SCALE**
Eliminates Flooding
Conversational Learning
Policy-Based Updates

**OPTIMISED MOBILITY**
Distributed Anycast Gwy
Integrated Routing /Bridging
vPC & ECMP

**OPERATIONAL FLEXIBILITY**
Layer 2 or Layer 3
Controller Choice

**Breaking the Traditional VXLAN Scale Barriers**

# DevOps & Network Programmability

Cisco live!

# Network Automation Today

In a majority of environments:

- Stage configuration in Notepad, copy/paste
- Automation according to definition of fixed third party tools
- Conversational configuration via expect scripts

Challenges:

- Manual, repetitive, error-prone tasks
- Waste time & talent
- Network lags behind industry automation capabilities

**Pasting large configuration:**

```
% Invalid command at '^' marker.
RTPDC1N3K1(config-cmap-qos)#   class copp-s-igmp
RTPDC1N3K1(config-cmap-qos)#     police pps 400
                               ^
% Invalid command at '^' marker.
RTPDC1N3K1(config-cmap-qos)#   class copp-s-routingProto2
RTPDC1N3K1(config-cmap-qos)#     police pps 1300
                               ^
% Invalid command at '^' marker.
RTPDC1N3K1(config-cmap-qos)#   class copp-s-v6routingProto2
RTPDC1N3K1(config-cmap-qos)#     police pps 1300
                               ^
% Invalid command at '^' marker.
RTPDC1N3K1(config-cmap-qos)#
```

**Typo? Start from scratch**





Cisco live!

# The Business Need for Network Programmability

- Networks grow to accommodate Apps
- More devices to manage
- Save time with repetitive tasks

- Business relies on infrastructure
- Lost uptime is lost revenue
- Minimise human error


- *High value engineers* being asked to focus on strategic tasks and shift away from maintenance

# What's Driving DevOps Adoption?

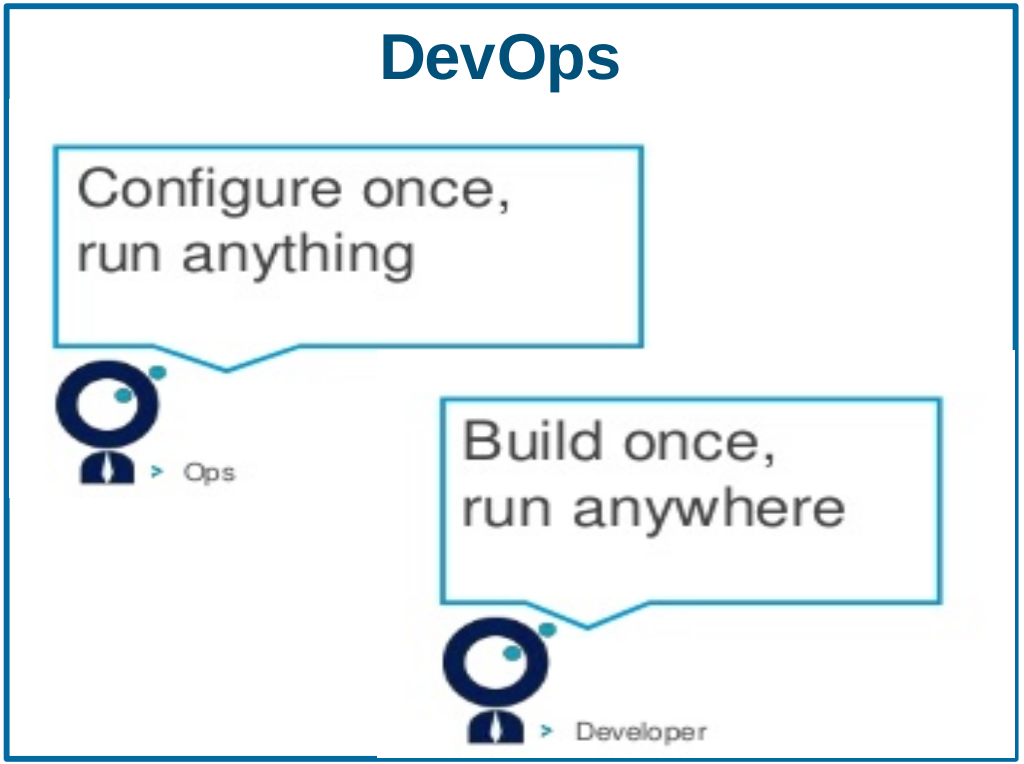
**"Organisations with High Performing DevOps organisations were 2.5x more likely to exceed profitability, market share and productivity goals…**

**…and had 50% higher market capitalisation growth over 3 years…"**

- **They're more agile**
  - 30x more frequent deployments
  - 8,000x faster lead time than their peers
- **They're more reliable**
  - 2x the change success rate
  - 12x faster MTTR

Source: Puppet Labs 2014 State Of DevOps - http://puppetlabs.com/2013-state-of-devops-infographic

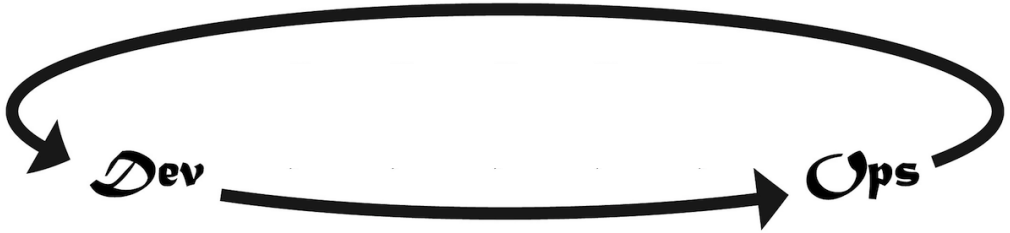
# App Development via DevOps is Changing Behaviour



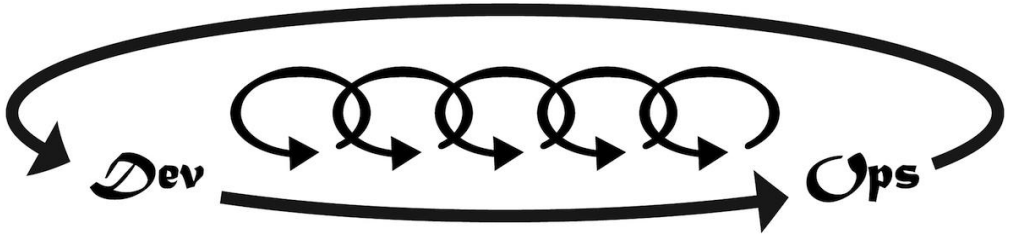
# DevOps – High Level Premises

1. Flow

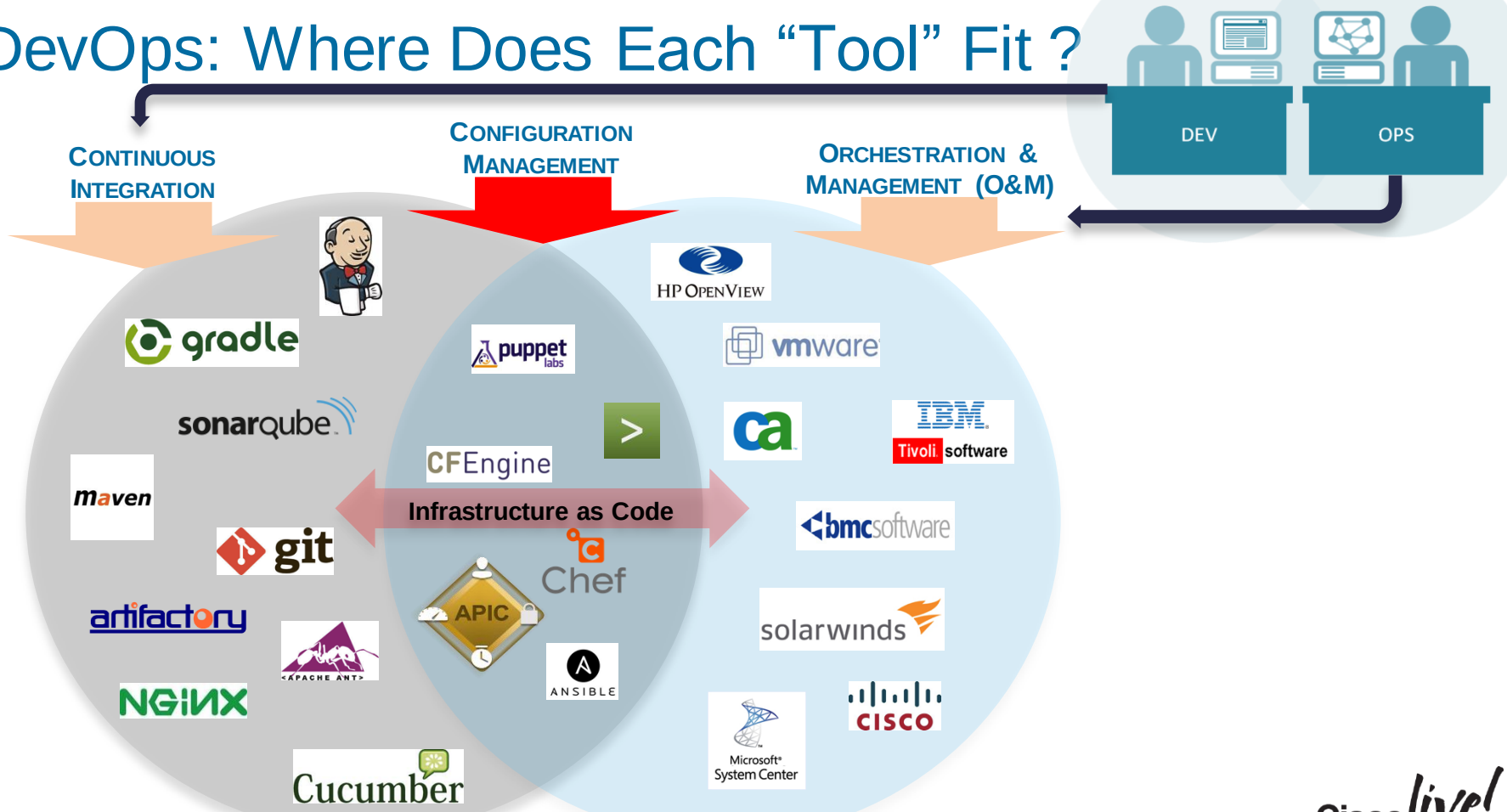

2. Feedback




3. Continuous Experimentation and Learning



# DevOps: Where Does Each "Tool" Fit ?



# What About the “Loss in Translation”?

## APPLICATION LANGUAGE

Exchange hadoop SAP SharePoint

- Application Tier Policy and Dependencies
- Security Requirements
- Service Level Agreement
- Application Performance
- Compliance
- Geo Dependencies
- Etc.

?

## NETWORK LANGUAGE

- VLAN
- IP Address
- Subnets
- Firewalls
- Quality of Service
- Load Balancer
- Access Lists

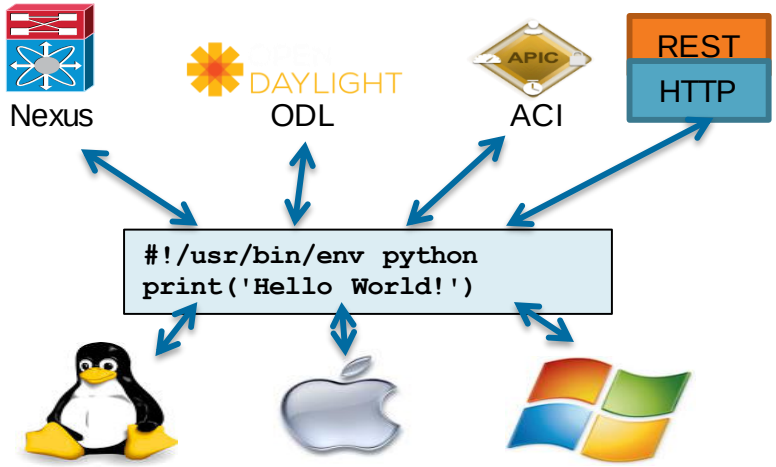




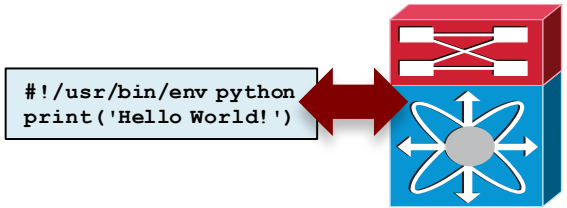
Cisco live!

# Programming: One Skill Applies to Many Tasks

- Nexus Portfolio
  - Python, Bash, NX-API
- ACI / APIC
  - REST, Python, etc.
- And outside…
  - All major OS'




## Programming Classifications

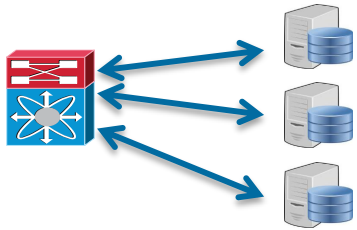



Off Box
Ex: REST, NX-API


On Box
Ex: Bash, Python

# Programmability Sample Use Cases

## Application Monitoring

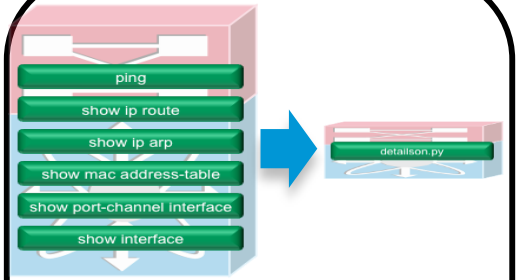


Proactive:

Get ahead of application issues and gather network information in real-time

Off-Box

## Super Commands

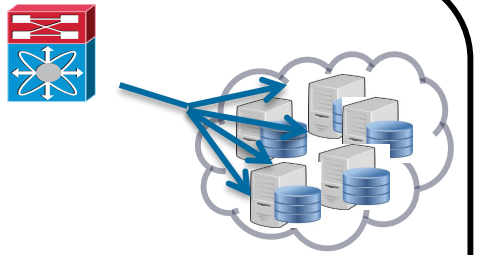



Efficient:

Create super-commands to encompass multiple troubleshooting steps

On-Box

## Topology Mapper



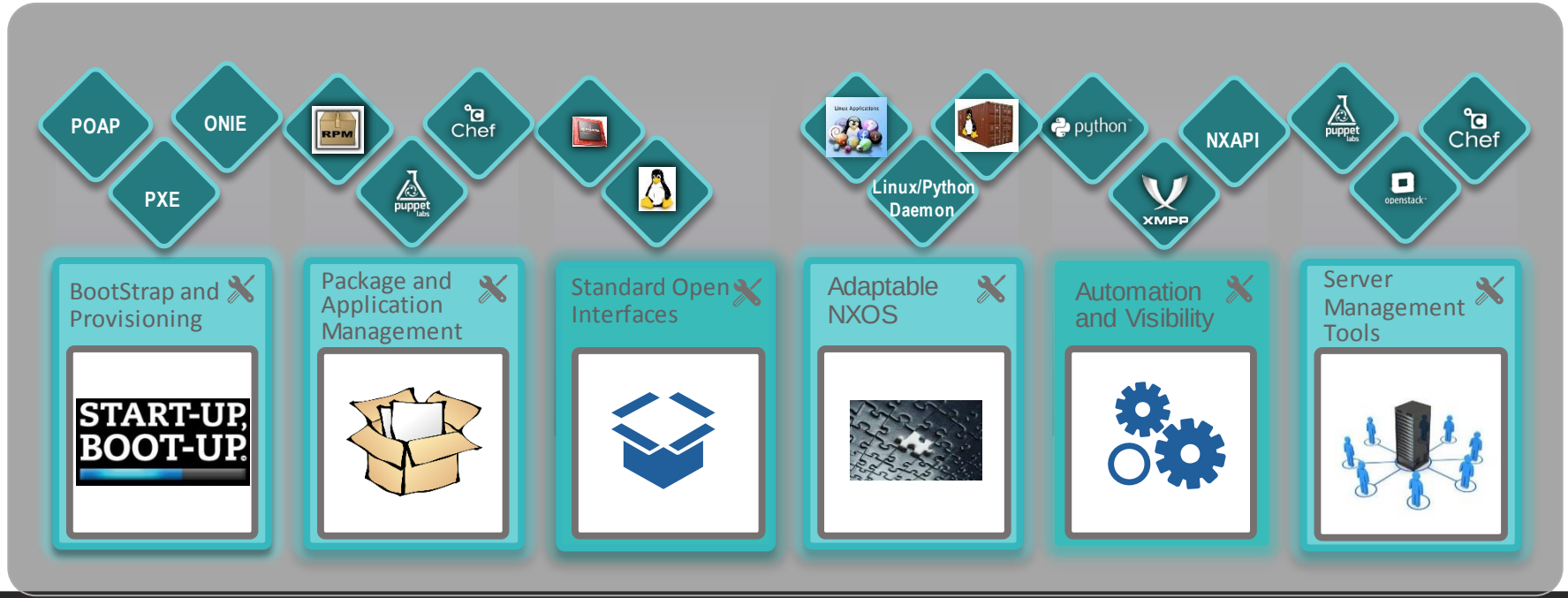
Scale:

Execute a push model for commands in multiple places

Off-Box

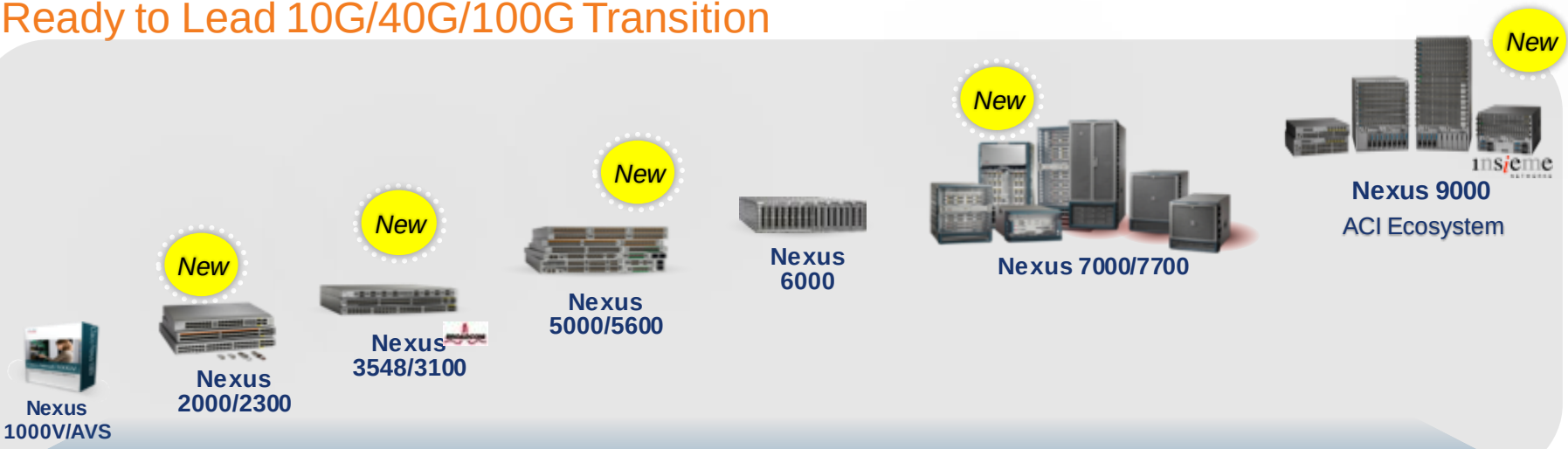
# NX-OS Evolution

## Enabling DevOps



# DC and Cloud Networking Portfolio – Nexus Family

## Ready to Lead 10G/40G/100G Transition




**OPEN**
APIs/ Open Source/ Application Policy Model

**HIGH PERFORMANCE FABRIC**
1/10/40/100 GE

**SCALABLE SECURE SEGMENTATION**
VXLAN, BGP-EVPN

| Resilient, Scalable Fabric | Workload Mobility Within/ Across DCs | LAN/SAN Convergence | Operational Efficiency—P-V-C | Architectural Flexibility |
|---|---|---|---|---|

**DELIVERING TO YOUR DATA CENTRE NEEDS**

# Nexus 9000 – Two Modes of Operation

**Standalone**

**ACI**

NXOS w/ Enhancements

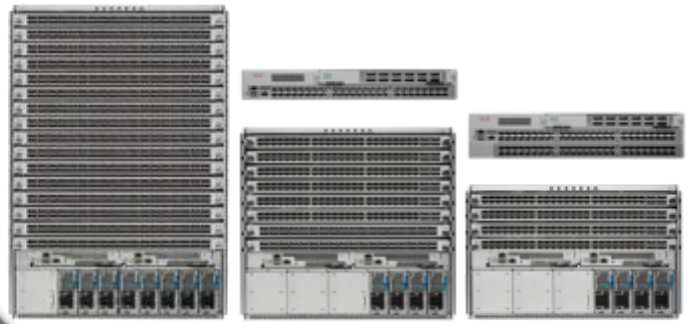
onePK

Bash Shell

python™

**Per-Box Programmability**

**Open, Flexible & Choice of Programmability Modes**



**1/10/40/100GE Common Platform**

Policy Controller iNXOS

python™

IFC

Bash Shell

**Centralised Fabric Programmability**

**Network Ops Driven, Switch Automation**

**Policy Based Fabric Automation**





Cisco live!

# Cisco NX-OS

## Programmability, Visibility, and Automation Suite

**Open Access & Programmable**

- BASH access
- Broadcom shell access
- Linux containers/Guest Shell
- NX-API
- XML/JSON RPC, NetConf
- Python scripting/Python Daemon
- OpenFlow support
- Cisco onePK™
- Customisable CLIs

**Automation and Orchestration**

- Puppet/Chef integration
- OpenStack network plugin
- OpenDaylight integration
- XMPP support
- Email support

**Visibility**

- vTracker
- Dynamic buffer monitoring
- Flow monitoring
- Enhanced Wireshark
- SMTP email “pipe” output
- Embedded Event Manager (EEM)

SNMP (v1, v2, v3), Syslog, NETCONF, RMON, CLI

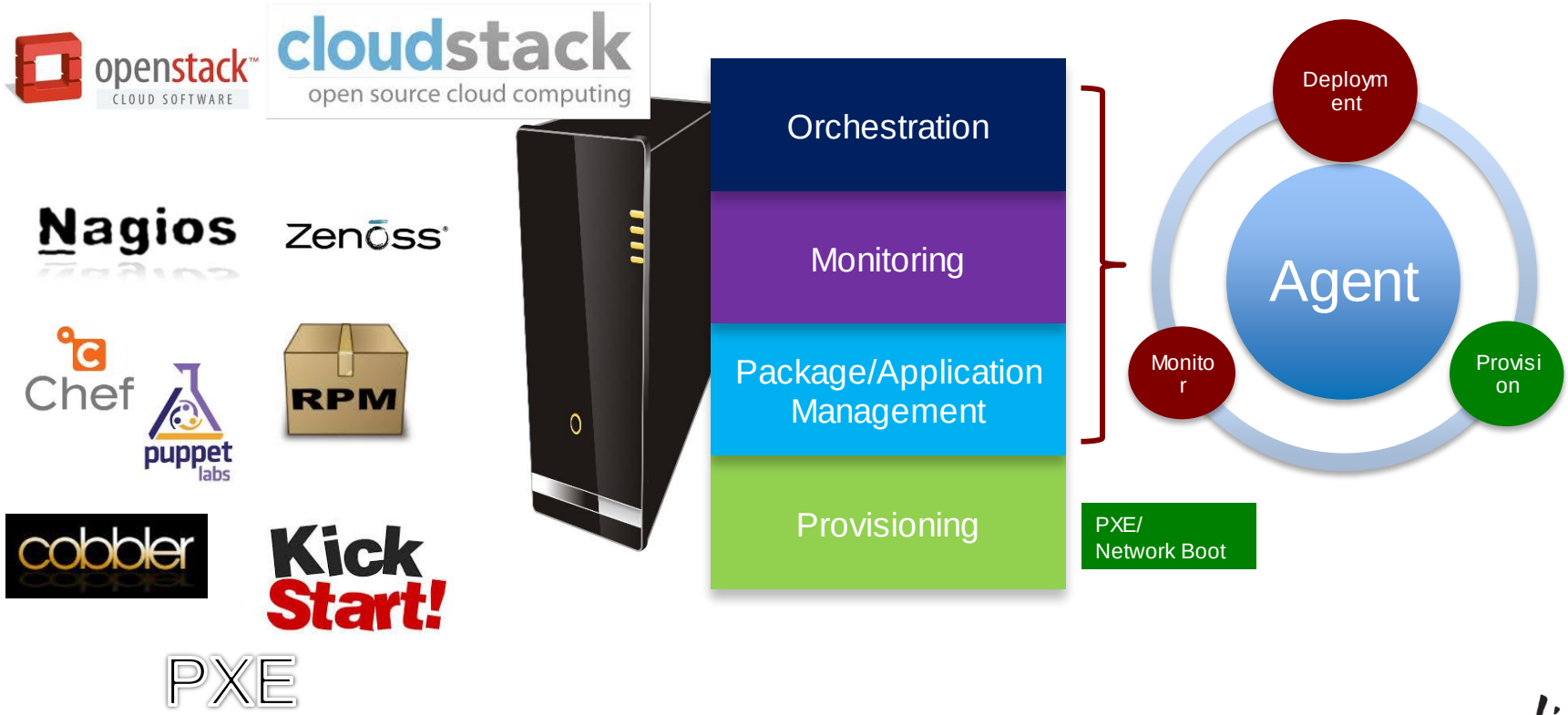




Cisco live!

# LINUX Server Management

## Dev-Ops








Cisco live!

# Consistent Dev-Ops Toolset and Operational Model

## Extended to the Network Devices

3rd Party DaeMONS

Python

Adaptable NXOS

Std Open API

Orchestration

Monitoring

Package/Application Management

Provisioning

BROADCOM

openstack™ CLOUD SOFTWARE

Linux netdevice Layer2

#/ lldpd
#/ tcpdump

NX-API

Chef

RPM

puppet labs

POAP

Object DRIVEN NX-API

Linux netdevice Layer3/4

#/ifconfig
#/ip route
#/ bgpd

CFEngine

PXE

onie

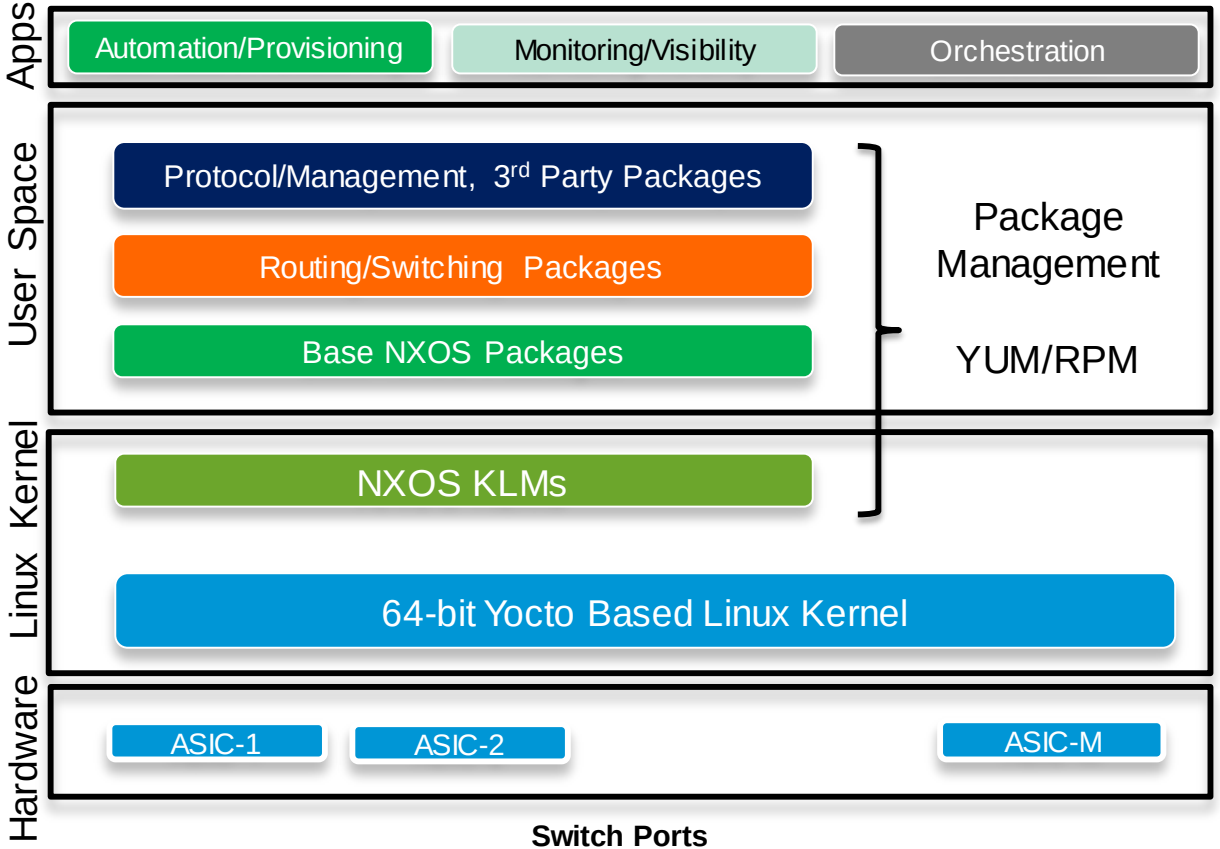




Cisco live!

# Open NX-OS Package Management

## Target Architecture

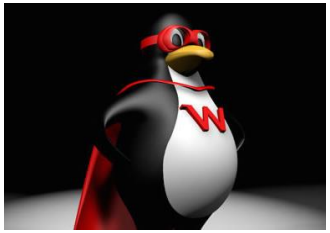



- Linux Kernel
- Yocto Toolchain
- RPM based packages for infrastructure services
- 3rd party daemon and packages (e.g. 3rd party routing

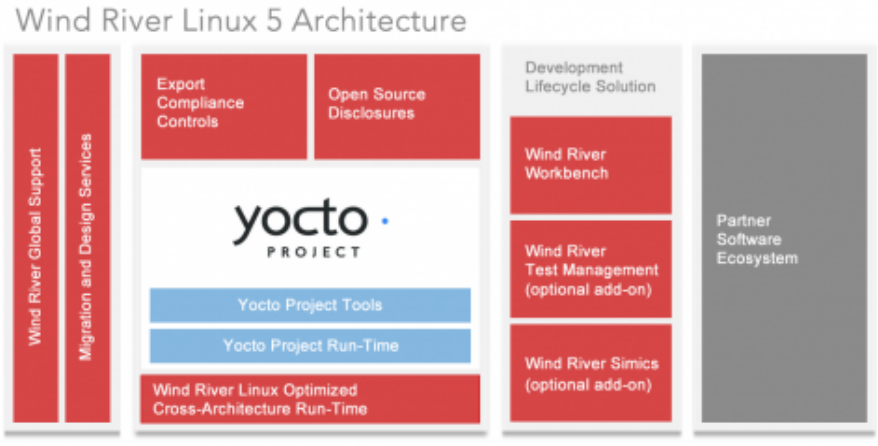
# Yocto Kernel

## Open Embedded Tool Chain

yocto PROJECT



- Open NX-OS leverages a 64-bit Wind River Linux Kernel with Yocto 1.2
- Provides open systems development environment
- Yocto Project uses a build system based on the OpenEmbedded (OE) project
- https://www.yoctoproject.org/downloads/core/denzil121




http://www.windriver.com/announces/yocto-project/



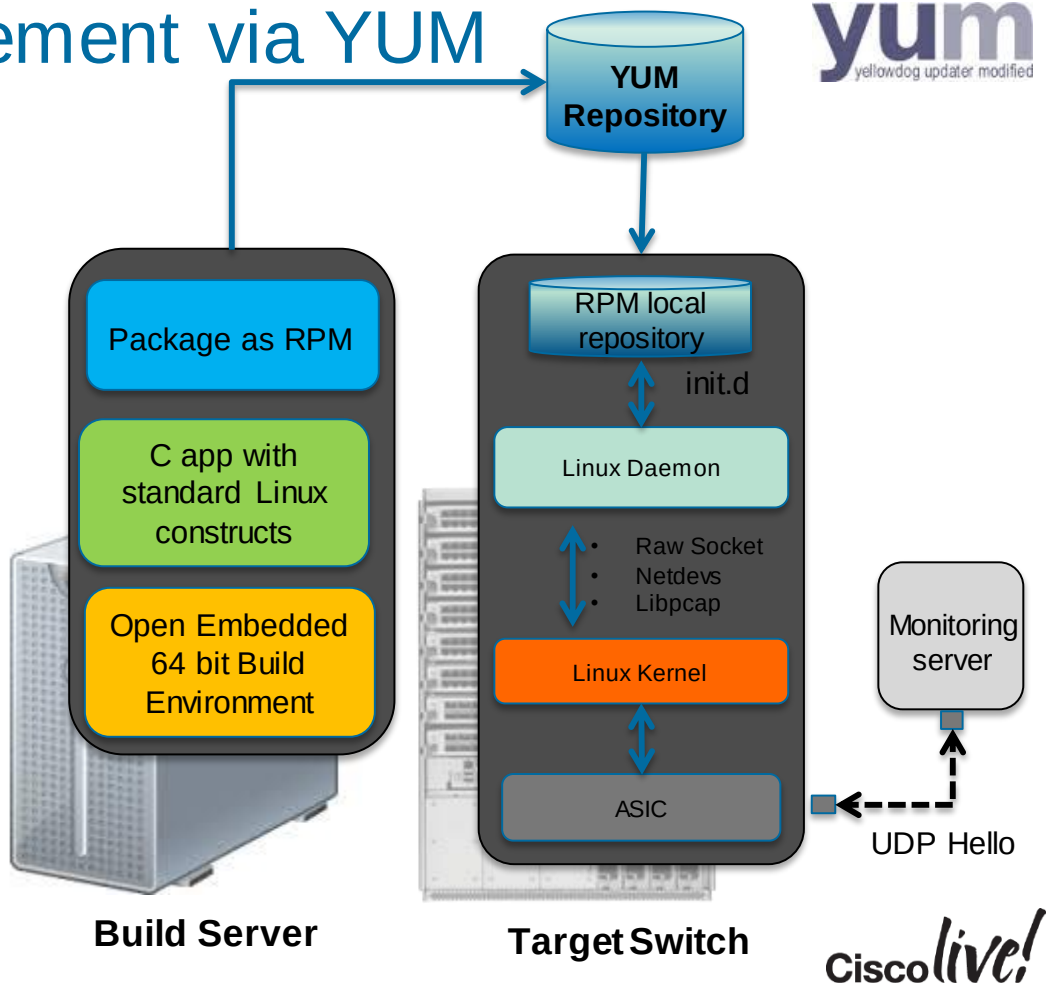


Cisco live!

# RPM Package Management via YUM

## LXC and Native Daemons

- Ability to install Linux Daemon in an LXC or in the NX-OS kernel
  - Install 3rd party apps like tcpdump, tcollector, iperf etc.
  - Install standard config management systems like Puppet/Chef
- Daemon managed via standard Linux interfaces
- Built-in support for YUM package manager
- Patching and upgrade using standard rpm/yum workflows
  - BGP can be upgraded via “yum update”



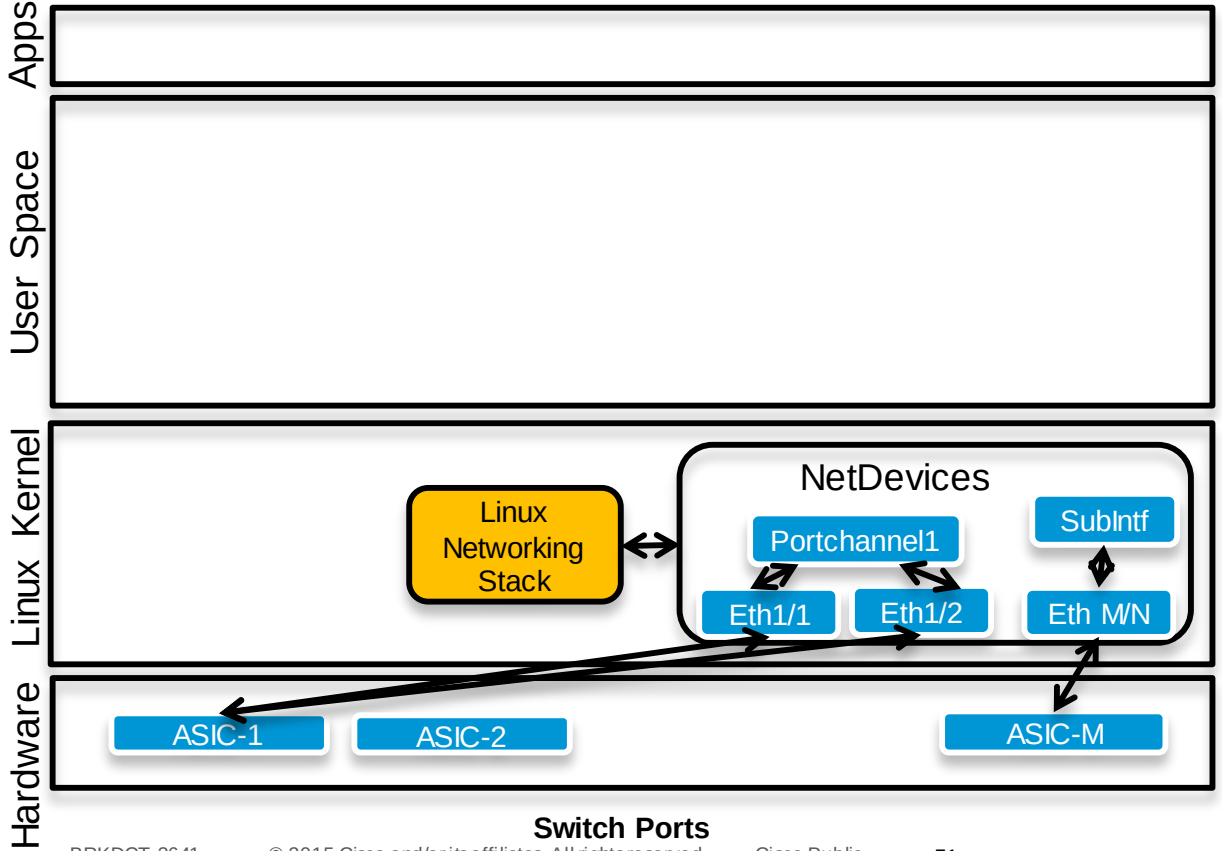
# NX-OS Linux Interfaces

## Bash Access

- Issue a CLI to gain access to Linux Bash Shell
- Leverage favorite Linux commands like ps, grep etc. available and could be used for further monitoring and scripting
- Bash shell has non-root privileges to protect against unintended operator errors
- Role-based access to Bash

# NX-OS Interfaces as Linux Netdevs

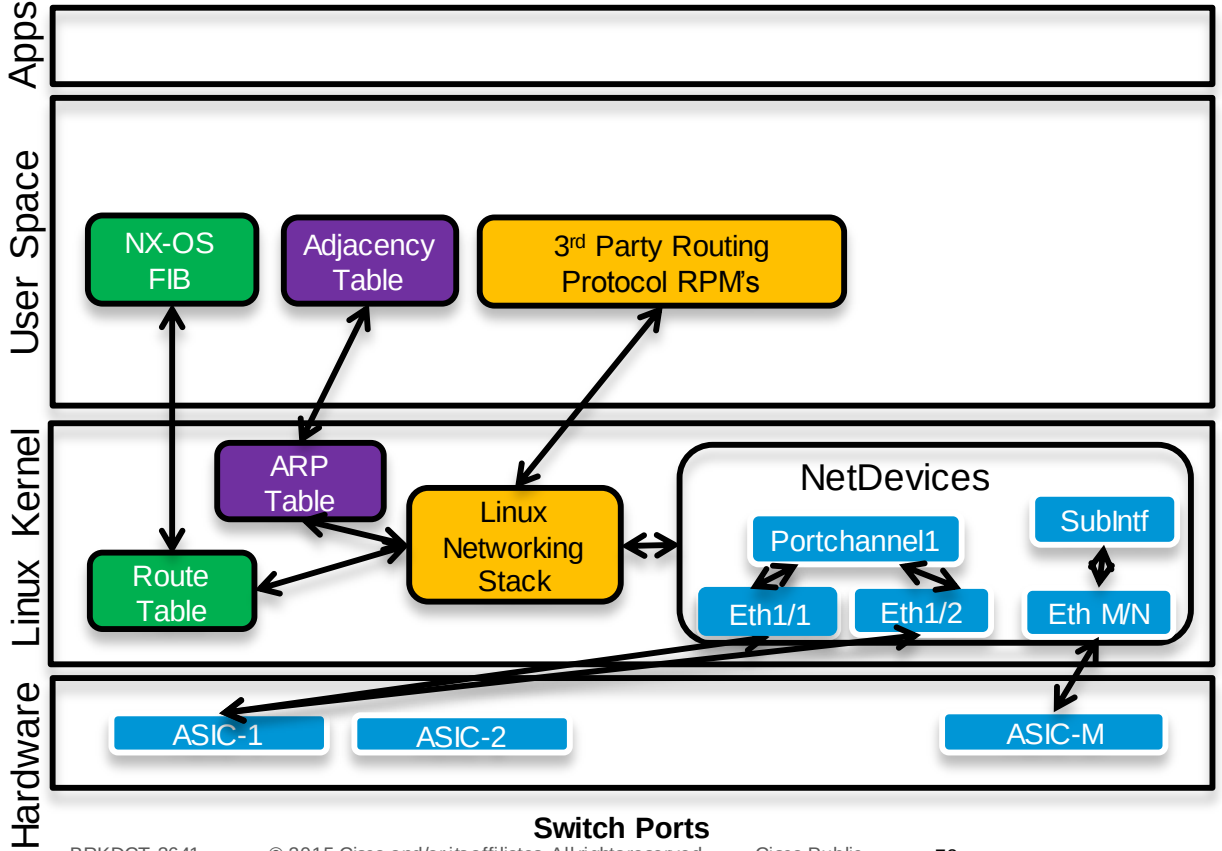
## Stage 1 – Access to Layer 1 and 2 NX-OS Constructs




- Access to switch infrastructure via Linux Network Stack
- Ifconfig, tcpdump, openLLDP etc.
- socket/libcap applications to send/snoop pkts to/from ASIC to CPU

# NX-OS Kernel Stack Interfaces

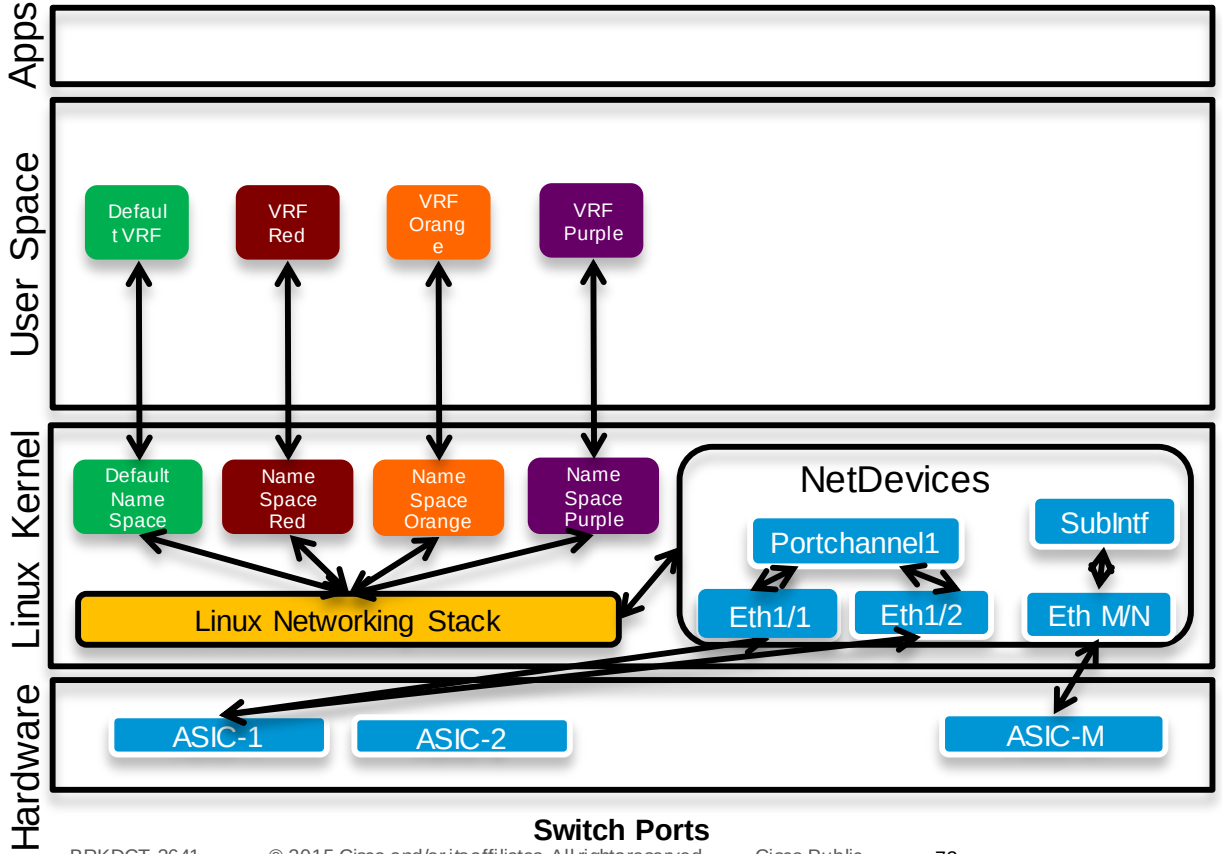
## Stage 2 – Access to Layer 3 NX-OS Constructs




- Allows 3rd party apps to inject routes to hardware using Linux interfaces
- Install 3rd party routing protocols built on Linux interfaces
- Access to ioctl kernel infrastructure

# NX-OS Kernel Stack Interfaces

## Stage 3 – Representing VRF Context via Linux Name




- Forwarding information within the 'VRF' context can be accessed via a corresponding Linux Name Space
- setns, ip-netns to change VRFs



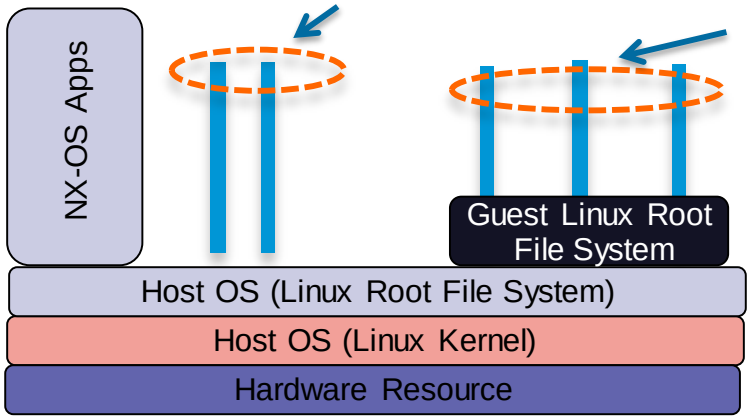


Cisco live!

# Linux Containers (LXC)

- Provides a secure and segregated operating environment for applications
- Can run either Cisco or Open Source applications
- Can use standard Linux distros
- OS Level Virtualisation
- Shared Kernel
- Shared physical resources
- Isolation through name spaces

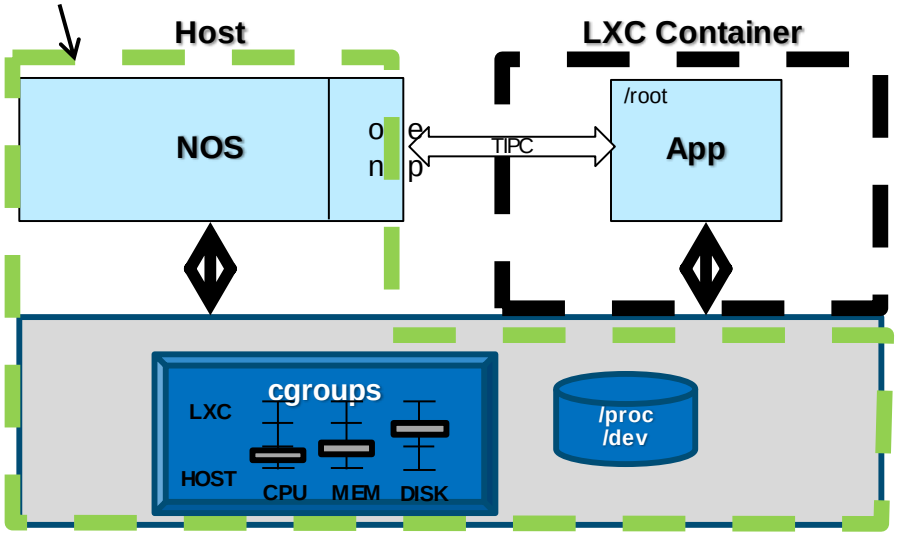
LXC is an operating system virtualisation technology that shares the host kernel with the guest but provides isolation through namespace extensions to the Linux kernel. http://linuxcontainers.org/



# Secure LXC Containers

- Enable Nexus 9000 switches to host customer applications using LXC virtualisation technologies
- Customers can compile & package their applications into OVAs for deployment on Nexus 9K

Trust Boundary for
Secure LXC Containers




**Base LXC to host trusted Cisco applications**

- Namespace separation with LXC
- Cgroups to limit resource usage

**Secure LXC to host customer applications and protect the integrity of the host system**

- Drop capabilities to limit a privileged user
- Use of Secure Linux technology, like SMACK, to address risks to host 3rd party applications running at root privilege
- Restrict TIPC

**Support for both 32-bit and 64-bit containers**

# LXC – Linux Containers (Cont'd)

LXC Benefits

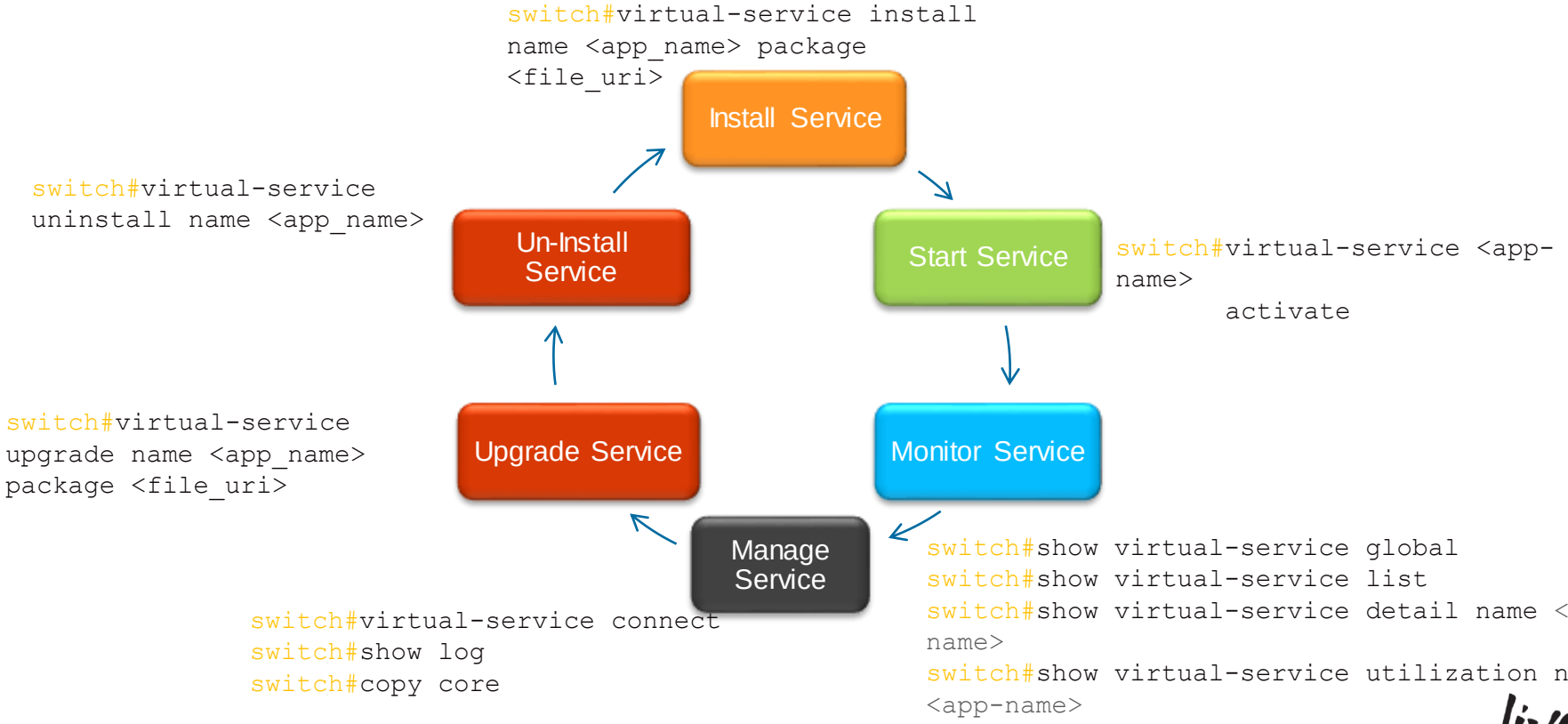
- Isolates Applications and Operating Systems
- Provides nearly native performance as LXC manages resource allocation in real-time
- More elastic than a full hypervisor
  - Less time to start
  - No need for a separate kernel boot
- Lightweight

LXC Limitations

- Shares kernel with underlying OS
- Only allows for Linux guests
- Not a full virtualisation stack
- Security depends on the host system

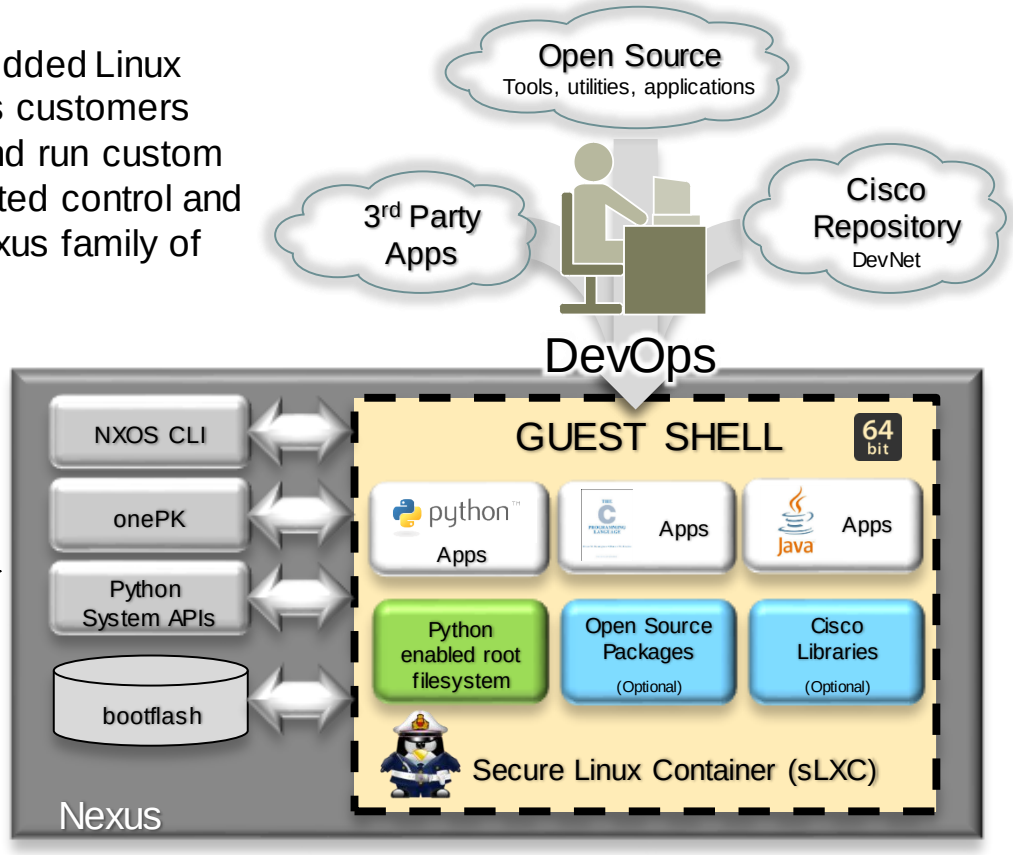
# LXC – Linux Containers (Cont'd)

## OVA Deployment Workflow



# Guest Shell

Guest Shell is an embedded Linux environment that allows customers (DevOps) to develop and run custom applications for automated control and management of the Nexus family of data centre switches.




### NXOS CLI interface
- Access the Guest Shell from NXOS CLI
- Access NXOS CLI from within the Guest Shell

### onePK APIs
- Access to a rich set of NXOS APIs for interface to management and datapath functions.

### Python System APIs
- BCM shell ?
- What else?

### bootflash
- Read/write access to the NXOS bootflash.

Guest Shell is automatically enabled. *Zero-touch*.

*64bit* application environment

Guest Shell ships with python support enabled.

C and Java support can be added through *YUM installs*.

Upgradeable rootfs packages

CISCO SECURITY CERTIFIED

Built on *Secure LXC*.





Cisco live!

# Guest Shell (Cont'd)

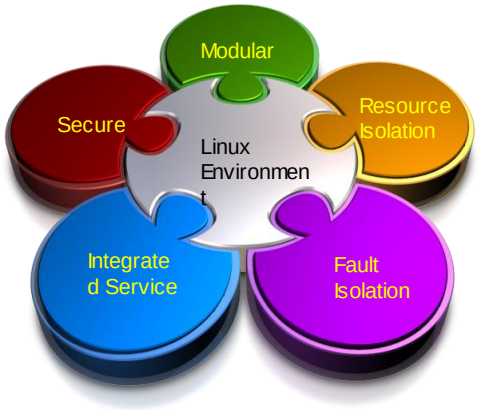
## Guest Shell "What"

- ✓ Linux Container Environment
  - ✓ Symbiotic relationship with Network OS.
  - ✓ Activated at boot time.
  - ✓ Application and programmatic interface habitat.
  - ✓ Can be resized as needed by user (via CLI).

**Allows users access to embedded Linux system**

## Guest Shell Innards

- ✓ RPM package manager (yum)
- ✓ Python interpreter (pip support)
- ✓ onePK libraries
- ✓ bootflash: access








Cisco live!

# Guest Shell (Cont'd)

- Guest Shell – LXC container installed and activated by default

```
n9396-1# show virtual-service list

Virtual Service List:

Name                     Status              Package Name
------------------------------------------------------------------------
guestshell+              Activated           guestshell.ova
```

- Access Guest Shell

```
n9396-1# guestshell
guestshell:~$
```

- Run Host Commands from Guest Shell:

```
guestshell:~$ dohost "sh ver | in 'System version' "
{0}{System version: 6.1(2)I2(3)}
guestshell:~$
```

- Run Guest Shell Commands from Host

```
n9396-1# guestshell run pwd
/home/guestshell
n9396-1#
```

- Change vrf for Guest Shell (by default it's in the default vrf)

```
guestshell:~$ chvrf management ping 171.70.42.150
PING 171.70.42.150 (171.70.42.150): 56 data bytes
64 bytes from 171.70.42.150: icmp_seq=0 ttl=54 time=1.589 ms
64 bytes from 171.70.42.150: icmp_seq=1 ttl=54 time=2.614 ms
```

- RPM Yum Install in Guest Shell

```
guestshell:~$ sudo chvrf management yum install bison
poky_1_5_1_x86_64                                        |  951 B     00:00
poky_1_5_1_x86_64/primary                                | 1.4 MB     00:00
poky_1_5_1_x86_64                                                   6582/6582
repository1                                              |  951 B     00:00
repository1/primary                                      | 1.3 MB     00:00
repository1                                                         6368/6368o
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package bison.i586 0:2.7.1-r0 set to be updated
--> Processing Dependency: libc.so.6 for package: bison-2.7.1-r0.i586
```

- Running Python in Guest Shell

```
guestshell:~$ which python
/usr/bin/python
guestshell:~$ python -V
Python 2.7.3
```

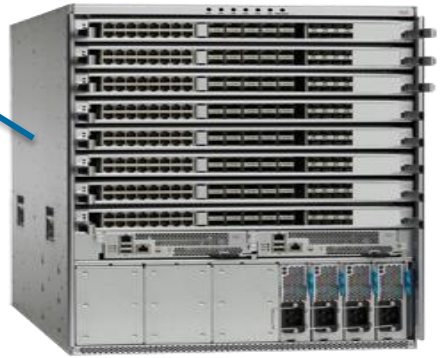
- Python PIP Install in Guest Shell
- (PIP is pre-installed in Guest Shell)

```
guestshell:~$ pip

Usage:
  pip <command> [options]

Commands:
  install                     Install packages.
  uninstall                   Uninstall packages.
  freeze                      Output installed packages in requirements format.
  list                        List installed packages.
  show                        Show information about installed packages.
  search                      Search PyPI for packages.
  zip                         Zip individual packages.
  unzip                       Unzip individual packages.
  bundle                      Create pybundles.
  help                        Show help for commands.
```

# Bash Access and Linux Containers


```
TME-1-9508-1# run bash
bash-4.2$
bash-4.2$ ifconfig -a
dummy0    Link encap:Ethernet  HWaddr a6:9f:04:2b:d3:ef
          BROADCAST NOARP  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:00:00:00:1b:01
          inet6 addr: fe80::200:ff:fe00:1b01/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:477374330 errors:0 dropped:0 overruns:0 frame:0
          TX packets:272305025 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:79582379696 (74.1 GiB)  TX bytes:58519512337 (54.5 GiB)

eth1      Link encap:Ethernet  HWaddr c0:67:af:a0:de:2e
          inet6 addr: fe80::c267:afff:fea0:de2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4813640 errors:0 dropped:0 overruns:0 frame:0
          TX packets:182072 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:706614536 (673.8 MiB)  TX bytes:91737078 (87.4 MiB)

eth2      Link encap:Ethernet  HWaddr 00:00:00:01:1b:01
          inet6 addr: fe80::200:ff:fe01:1b01/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9400  Metric:1
```

- Issue a CLI to gain access to Linux Bash Shell
- Leverage favorite Linux commands like ps, grep etc. available and could be used for further monitoring and scripting
- Role-based access to Bash





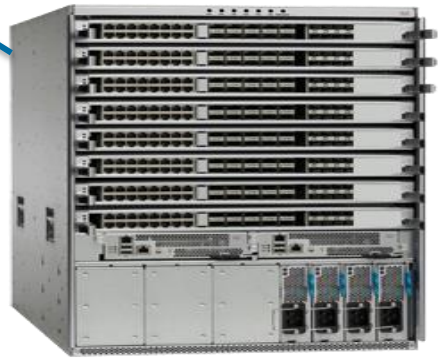


Cisco live!

# Bash Shell Access

- Monitor Memory Utilisation and Processes through Bash:
- Leverage the standard Linux command to monitor network processes

```
bash-4.2$ top
Shift + F
Select "N" for Memory

top - 15:00:48 up 1 day, 12:41,  4 users,  load average: 0.22, 0.28, 0.33
Tasks: 219 total,   2 running, 215 sleeping,   0 stopped,   2 zombie
Cpu(s):  9.7%us,  3.3%sy,  0.0%ni, 86.4%id,  0.1%wa,  0.2%hi,  0.3%si,  0.0%st
Mem:  16402508k total,  3452904k used, 12949604k free,   258260k buffers
Swap:        0k total,        0k used,        0k free,  1477268k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5343 svc-isan  20   0  345m  54m 7936 S    0  0.3   0:01.13 vpx1
 5361 root      20   0  371m  49m  32m S    0  0.3   3:10.56 clis
 5470 root      20   0  770m  48m  11m S    1  0.3  10:12.65 netstack
 5468 root      20   0  605m  44m 6952 S    0  0.3   0:11.99 arp
 5344 svc-isan  20   0  335m  43m 7768 S    0  0.3   0:01.06 pixm
 5683 root      20   0  324m  32m 8612 S    0  0.2   0:00.72 l2fm
 5675 root      20   0  320m  31m  14m S    2  0.2   9:29.22 ipqosmgr
 5508 root      20   0  597m  31m  10m S    0  0.2   0:41.30 snmpd
 5682 root      20   0  334m  30m  10m S    0  0.2   0:01.94 ethpm
 5681 root      20   0  321m  30m 7636 S    0  0.2   4:51.05 diag_port_lb
 5706 root      20   0  936m  26m  11m S    0  0.2   0:14.00 pim
 5664 root      20   0  314m  25m 8248 S    0  0.2   0:00.78 eltm
 5703 root      20   0  604m  23m 8676 S    0  0.1   4:03.24 ospf
```



# BCM Shell Access

- Issue a CLI to get shell access to underlying BCM chips
- Direct read/write access to hardware tables
- Can Peek/Poke underlying registers
- Python wrapper to get BCM Shell output


```
TME-1-9508-1# bcm-shell module 1
Warning: BCM shell access should be used with caution
Entering bcm shell on module 1
Available Unit Numbers: 0 1 2
bcm-shell.0> ^[[A^[[A^[[B

bcm-shell.0> l3 l3table show
l3 l3table show
Unit 0, free L3 table entries: 212960
Entry VRF IP address       Mac Address           INTF MOD PORT    CLASS HIT
147488 1    192.168.1.2       00:00:00:00:00:00  100006    0    0        0 y
149300 1    30.1.1.255        00:00:00:00:00:00  149150    0    0        0 n
150696 1    30.1.1.1          00:00:00:00:00:00  100012    0    0        0 n
152696 1    10.1.1.3          00:00:00:00:00:00  100007    0    0        0 y
154860 1    192.168.1.15      00:00:00:00:00:00  149150    0    0        0 n
156336 1    192.168.1.0       00:00:00:00:00:00  100000    0    0        0 n      (LOCAL ROUTE)
163452 1    192.168.1.13      00:00:00:00:00:00  149151    0    0        0 y      (LOCAL ROUTE)
165120 1    192.168.1.3       00:00:00:00:00:00  149150    0    0        0 n
166280 1    30.1.1.0          00:00:00:00:00:00  100000    0    0        0 n      (LOCAL ROUTE)
168280 1    10.1.1.2          00:00:00:00:00:00  100006    0    0        0 y
170444 1    192.168.1.14      00:00:00:00:00:00  100010    0    0        0 y
173968 1    192.168.1.1       00:00:00:00:00:00  149151    0    0        0 y      (LOCAL ROUTE)
174872 1    30.1.1.2          00:00:00:00:00:00  149151    0    0        0 n      (LOCAL ROUTE)
179036 1    192.168.1.12      00:00:00:00:00:00  100000    0    0        0 n      (LOCAL ROUTE)
183716 1    192.168.1.11      00:00:00:00:00:00  149150    0    0        0 n
184680 1    192.168.1.6       00:00:00:00:00:00  100007    0    0        0 y
186876 1    10.1.1.10         00:00:00:00:00:00  149151    0    0        0 n      (LOCAL ROUTE)
192308 1    192.168.1.9       00:00:00:00:00:00  149151    0    0        0 n      (LOCAL ROUTE)
193012 1    10.10.10.10       00:00:00:00:00:00  149151    0    0        0 y      (LOCAL ROUTE)
193528 1    192.168.1.4       00:00:00:00:00:00  100000    0    0        0 n      (LOCAL ROUTE)
201800 1    192.168.1.7       00:00:00:00:00:00  149150    0    0        0 n
```

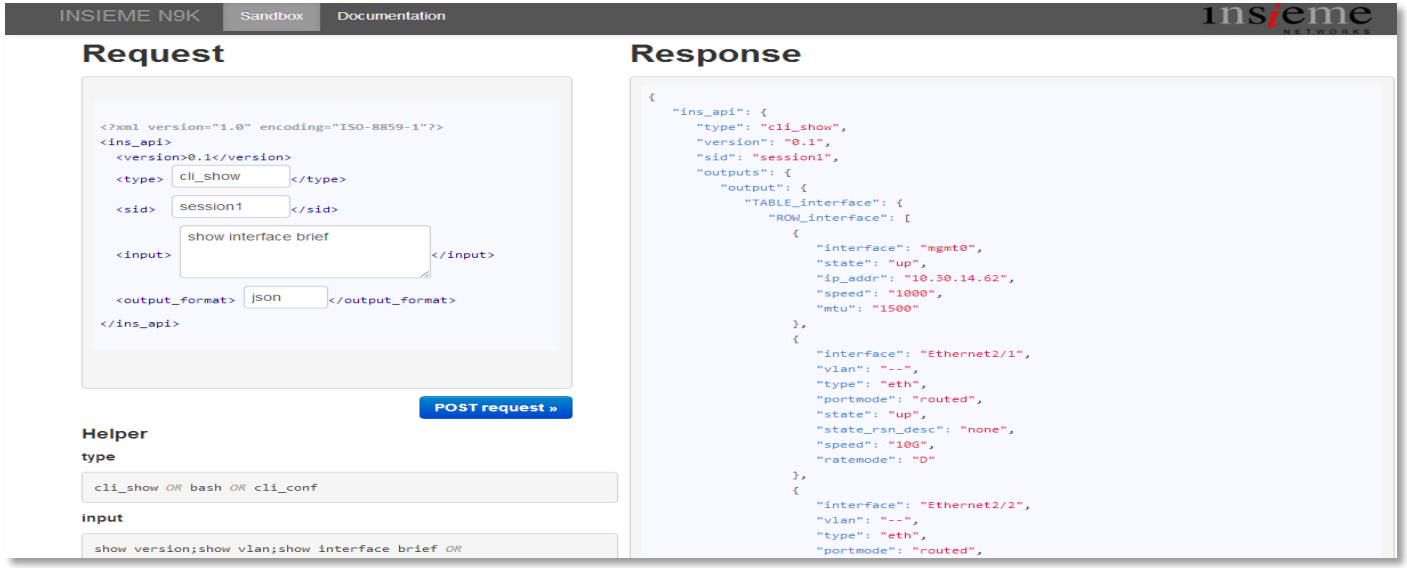




Cisco live!

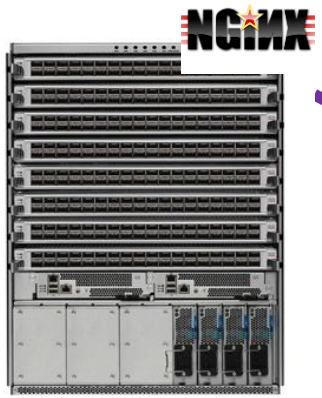
# Nexus 9000 - NX-API

- Open RPC API – Extensible to support REST
- Universal Access: http or https based
- Programmability Oriented
- Ready for Integration: CLI based input and structured output (JSON/XML)



# Nexus 9000 NX-API

Open RPC API – Extensible to support REST



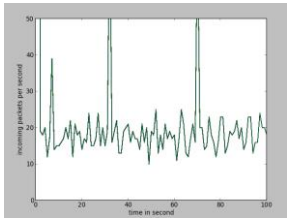

N9K

CLI Input

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ins_api>
  <type>cli_conf</type>
  <version>0.1</version>
  <sid>1</sid>
  <input>show interface brief </input>
  <output_format>xml</output_format>
</ins_api>
```

HTTP

```
{
  "ins_api": {
    "type": "cli_show",
    "version": "0.1",
    "sid": "session1",
    "outputs": {
      "output": {
        "TABLE_interface": {
          "ROW_interface": [
            {
              "interface": "mgmt0",
              "state": "up",
              "ip_addr": "172.21.128.227",
              "speed": "1000",
              "mtu": "1500"
            },
            {
              "interface": "loopback0",
              "state": "up"
            }
          ]
        }
      }
    }
  }
}
```



Programmability

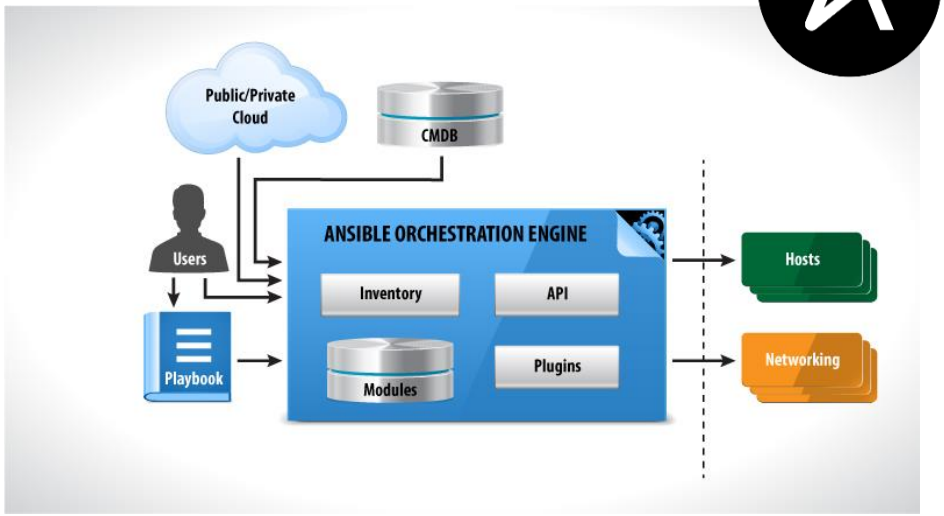




Cisco live!

# Ansible and NX-API

- Ansible
  - Agentless
  - SSH transport
  - Push Model
  - Python based
- Leverages NX-API




```
---
- hosts: switches
  remote_user: ansible
  tasks:
  - name : create vlan 100 with name ansible_vlan using nxapi
    cisco_vlan_nxapi: state=present name=ansible_vlan vlan_id=100
```

https://github.com/Mierdin/ansible-nxapi/tree/master

/www.jedelman.com/home/leveraging-cisco-nx-api-with-ansible-to-make-your-life-easier

# Python Scripting


- Built in Python Shell
- Can be used to execute CLI commands and reference Objects through Python interpreter
- Most commands can be executed to return the command output as a Python Dictionary
- Pass arguments to python scripts from CLI
- Integration with Embedded Event Manager (EEM)

# Community Code Development

- Visit us on GitHub: https://github.com/datacenter/nexus9000
- ACI and NX-OS code examples and libraries
- Open source and community developed tools by partners and 3rd party developers







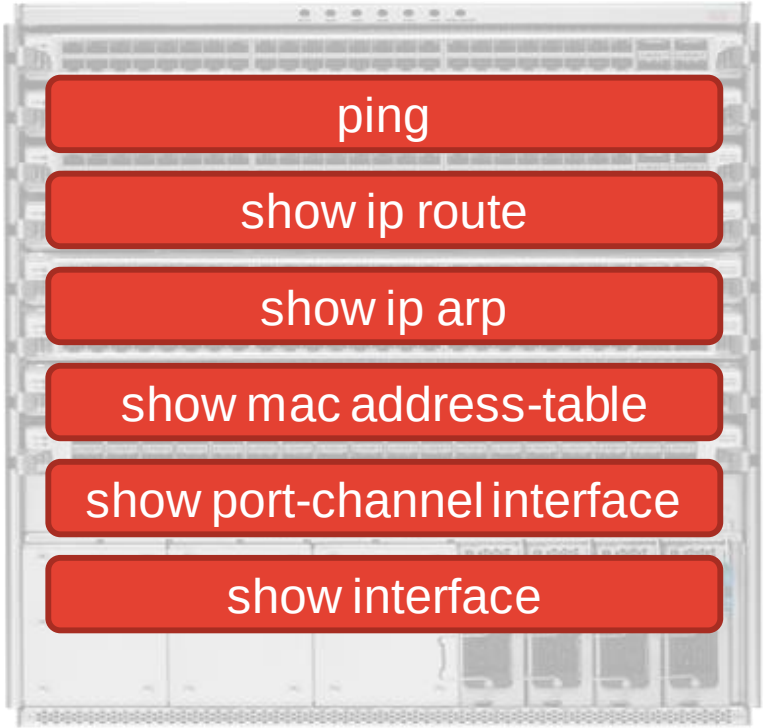


# Python Scripting Example

## Serviceability – Reduce Time-to-Resolution

Customer


TAC Engineer

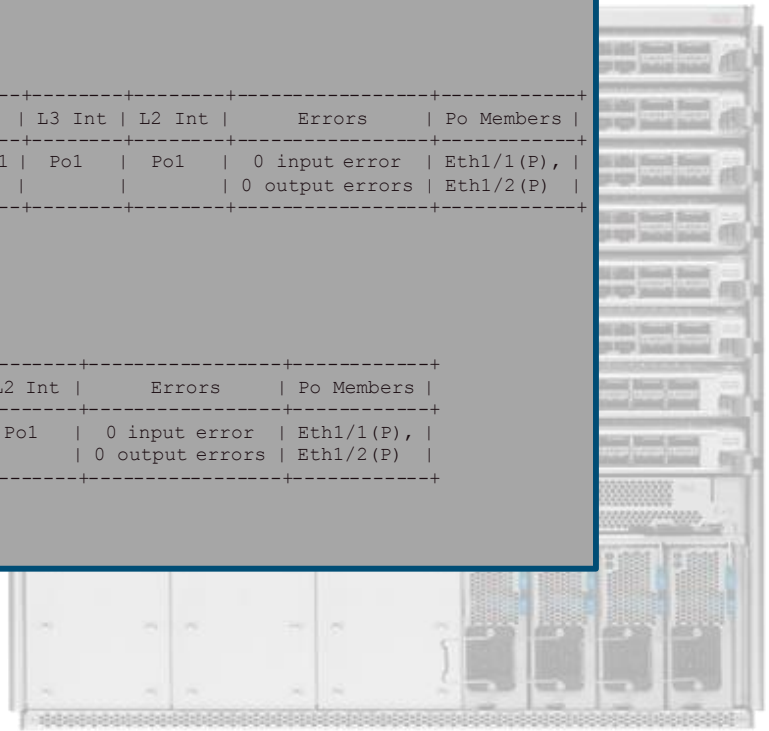


# Python Scripting Example

## Serviceability – Reduce Time-to-Resolution

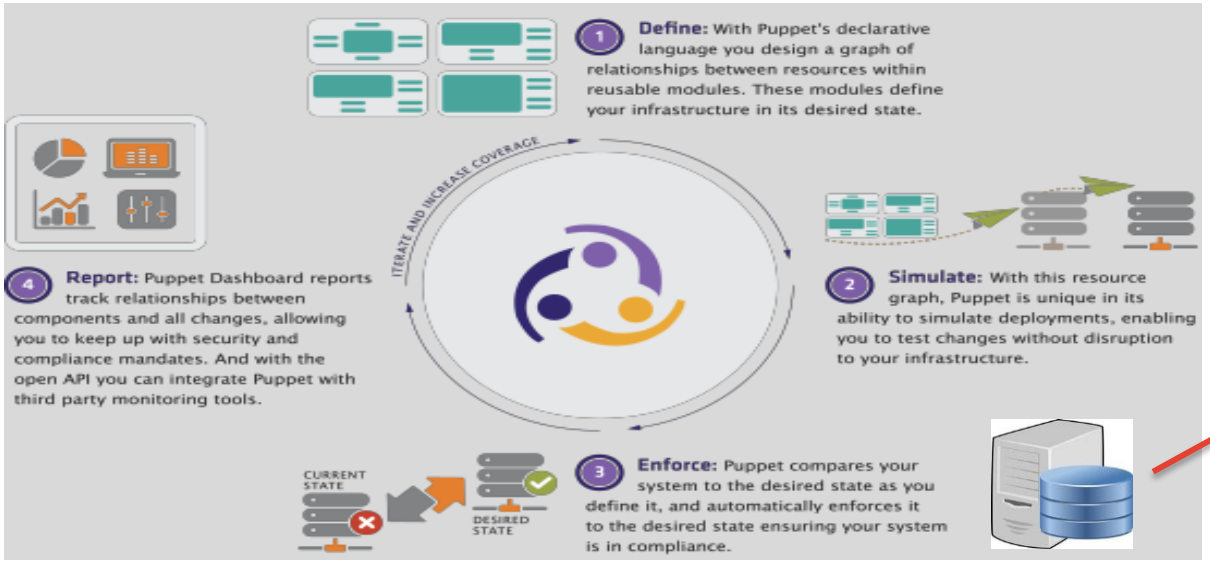
```
INSIEME# detailson 192.168.208.2

     Details for IP Address:  192.168.208.2

+---------------+----------------------+-----------------+---------------+--------+--------+-----------------+------------+
|   IP Address  |      Ping Result     |     Next Hop    |      MAC      | L3 Int | L2 Int |      Errors     | Po Members |
+---------------+----------------------+-----------------+---------------+--------+--------+-----------------+------------+
| 192.168.208.2 |   0.00% packet loss  | 10.1.1.1, ospf-1 | 30f7.0d9f.8801 |  Po1   |  Po1   |  0 input error  | Eth1/1(P), |
|               | 0.494/3.455/15.219 ms |                 |               |        |        | 0 output errors | Eth1/2(P)  |
+---------------+----------------------+-----------------+---------------+--------+--------+-----------------+------------+


     Enter Next IP to get details on (Press 0 to exit): 10.1.1.1

     Details for IP Address:  10.1.1.1

+------------+--------------------+----------+---------------+--------+--------+-----------------+------------+
| IP Address |     Ping Result    | Next Hop |      MAC      | L3 Int | L2 Int |      Errors     | Po Members |
+------------+--------------------+----------+---------------+--------+--------+-----------------+------------+
|  10.1.1.1  |  0.00% packet loss  | attached | 30f7.0d9f.8801 |  Po1   |  Po1   |  0 input error  | Eth1/1(P), |
|            | 0.578/0.67/0.945 ms |          |               |        |        | 0 output errors | Eth1/2(P)  |
+------------+--------------------+----------+---------------+--------+--------+-----------------+------------+


     Enter Next IP to get details on (Press 0 to exit):
```

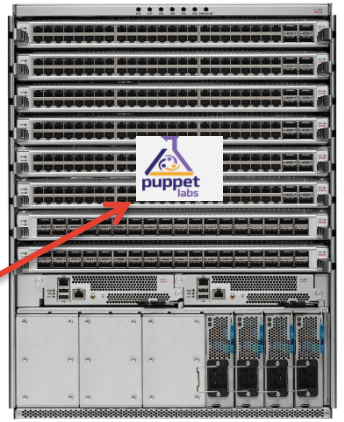
TAC Engineer



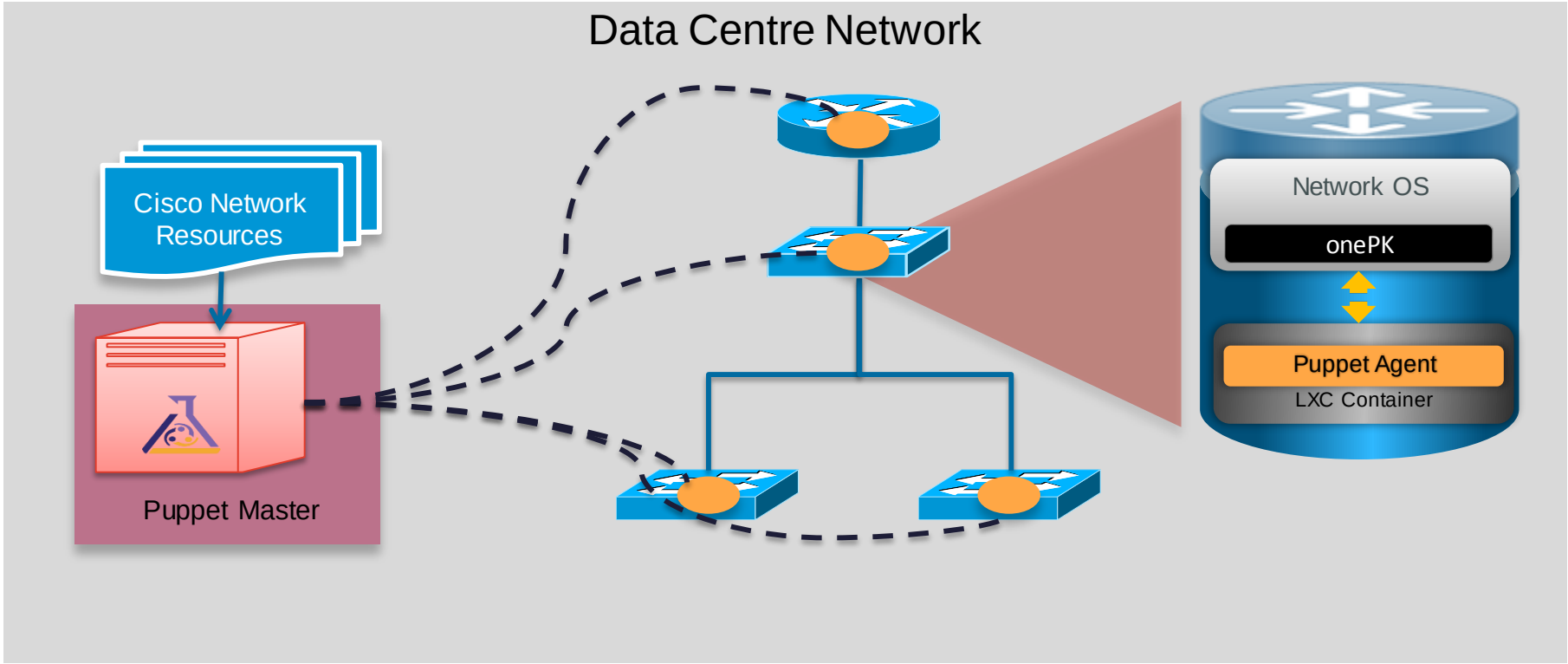
# Puppet/Chef Agents

- Initially will run in an LXC
- Will run as native daemons and installed via RPM – Q1CY15



# Cisco Puppet Plug-In: Architecture



# Cisco Puppet Resource Type Coverage: 1

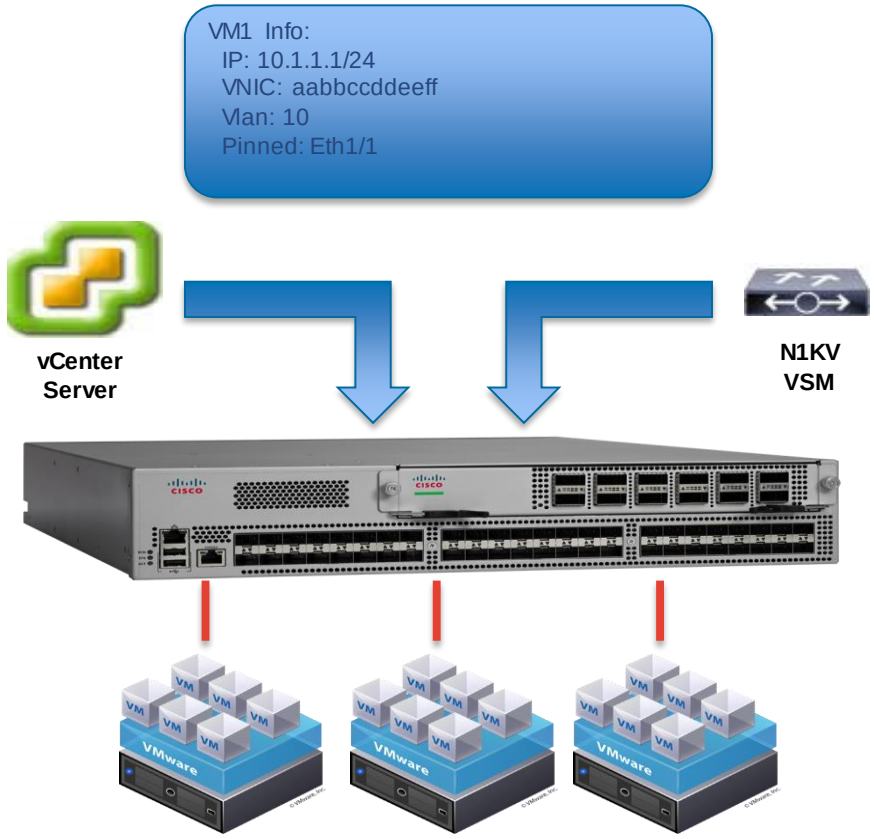
| Feature | Resource Name | Description |
|---|---|---|
| Cisco Device Access | cisco_device | Allows credentials for user access control & accounting |
| Base L2/L3 interface | cisco_interface | General interface & L2/L3 base settings |
| VLAN | cisco_vlan | Create/destroy of VLANs and general settings |
| Interface-vlan (SVI) | cisco_interface_vlan | Create/destroy of SVIs and SVI specific interface settings |
| VLAN Trunking Proto (VTP) | cisco_vtp | VTP global settings |
| SNMP | cisco_snmp_server<br>cisco_snmp_community<br>cisco_snmp_group<br>cisco_snmp_user | SNMP monitoring settings. Notification receiver settings not covered as of now. |
| OSPF | cisco_ospf<br>cisco_ospf_vrf<br>cisco_interface_ospf | OSPF instance create/destroy, per-VRF settings, and interface settings (area, cost, msg digest, etc) |

# Cisco Puppet Resource Type Coverage: 2

| Feature | Resource | Description |
|---|---|---|
| TACACS/AAA***<br><br>***full set not available at EFT target date | cisco_tacacs_server<br>cisco_tacacs_server_host<br>cisco_aaa_tacacs_group<br>cisco_aaa_authentication<br>cisco_aaa_authorization<br>cisco_aaa_accounting | • TACACS global settings<br>• TACACS per-host settings<br>• group association and settings<br>• mapping of groups to AAA features (authentication, authorization, accounting). |
| Raw Config CLI commands | cisco_command_config | Resource to directly apply blocks of configuration CLI commands. |

# vTracker – VM Visibility

- Ability to track VM information per port
  - List of VMs attached
  - VM's IP Addresses, VLAN, Port Group, vNIC, MAC address
- Integration with N1KV to
  - Provide Upstream/Downstream Views
  - Provision VLANs on trunks to ESX Hosts
- Trace VM Movement history in network (SPLUNK integration)
- Dynamic network policies



# VM Tracker Functionality

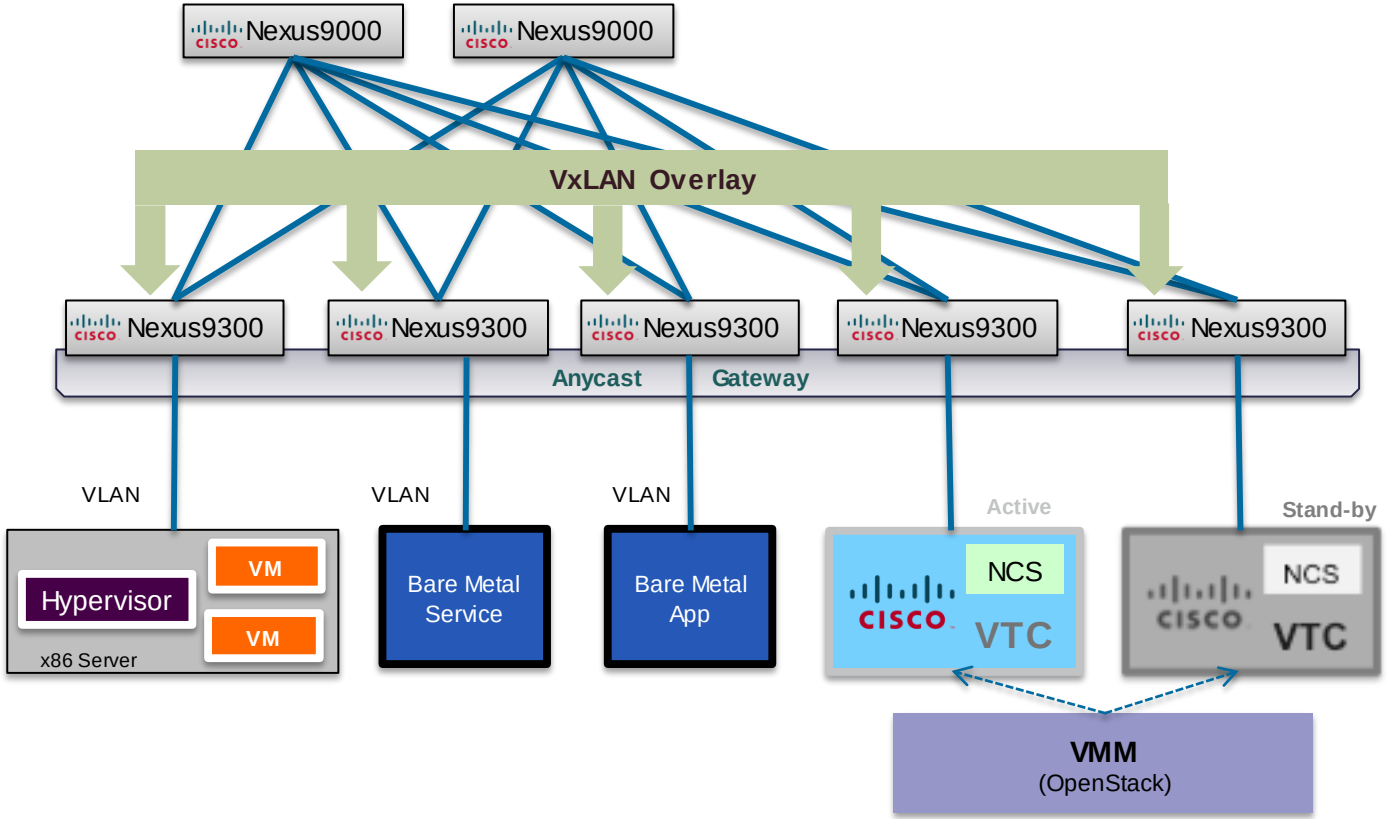
- Add/remove vlans on interface based on VLAN requirements of VMs in the host connected to that interface.
- Automatic creation and deletion of VLANs globally. This functionality can be enabled/disabled per connection.
- Instantaneously update VLAN configuration based on changes in vCenter.
- Support for Port-channel towards the host.
- Ability to configure VLAN range per connection to limit the scope of dynamic VLAN configuration.
- Ability to enable stickiness of user configured native vlan.
- Ability to enable/disable VM Tracker per interface.
- Connection to vCenter based on username/password or extension-key based certificate.
- Supports HA and Fault Tolerance features of vCenter.

# Controllers and Orchestration Tools

# Different Operations Modes with Nexus 9000

| Nexus 9000 Standalone (with VTS) | Application Centric Infrastructure (with APIC) |
|---|---|
|  |  |
| NX-OS Working w/ multiple SDN controllers (inclusive for NfV) | APIC data object / policy model integrated natively with NX-OS running on Nexus 9000 switches (spines and leaves) |
| Loosely coupled integration (custom integration and open programmability) | Tightly coupled integration – Out of the box ready system |
| Deploy for multiple topologies Leaf/Spine, 2-Tier Aggregation, Full Mesh | Deployed as a well-known CLOS topology. It's a system approach. |
| Interoperable w/ 3rd Party ToR Switches and WAN gear | Must be Nexus 9000 hardware for leaves and spines as well as ACI Software (switch code and APIC controller) |

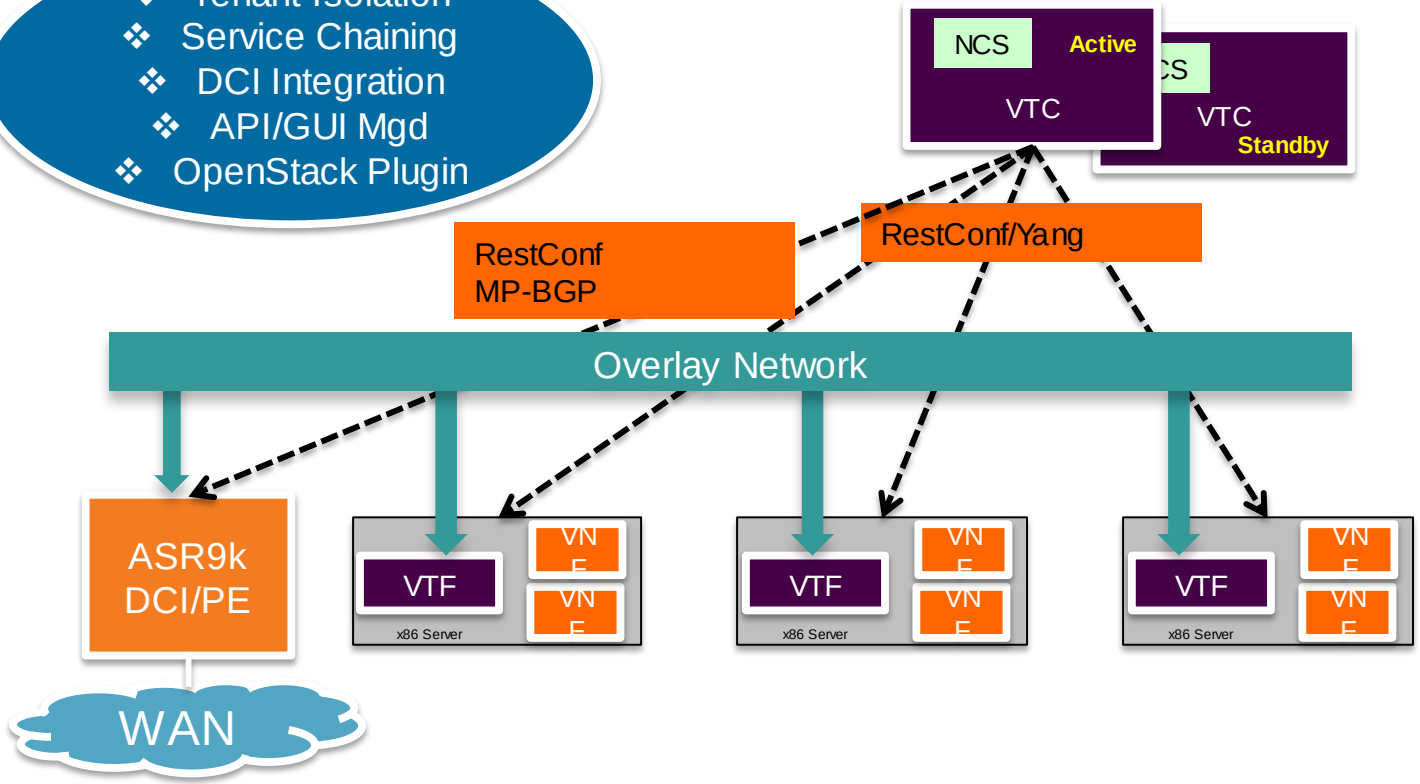
# Virtual Topology System (VTS) Phase 1.5




1. Network based L2 and L3 VxLAN for non-EVPN (multicast-based) and MP-BGP EVPN control plane
2. VTC provides orchestration/provisioning of Nexus 9300 VTEP services
3. OpenStack integration

# Virtual Topology System (VTS)

- Tenant Isolation
- Service Chaining
- DCI Integration
- API/GUI Mgd
- OpenStack Plugin




Virtual Topology Controller

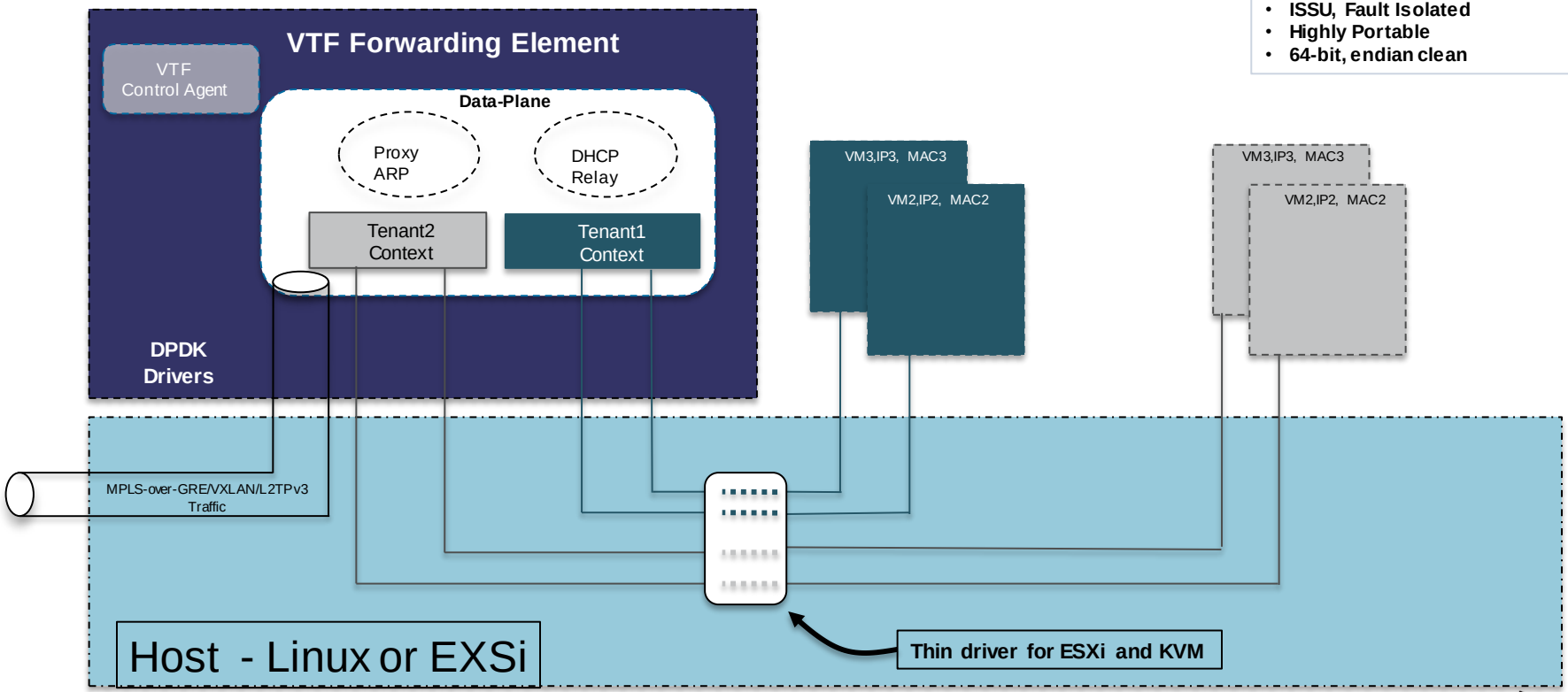
- Provisioning
- Overlay Control Plane

Virtual Topology Forwarder

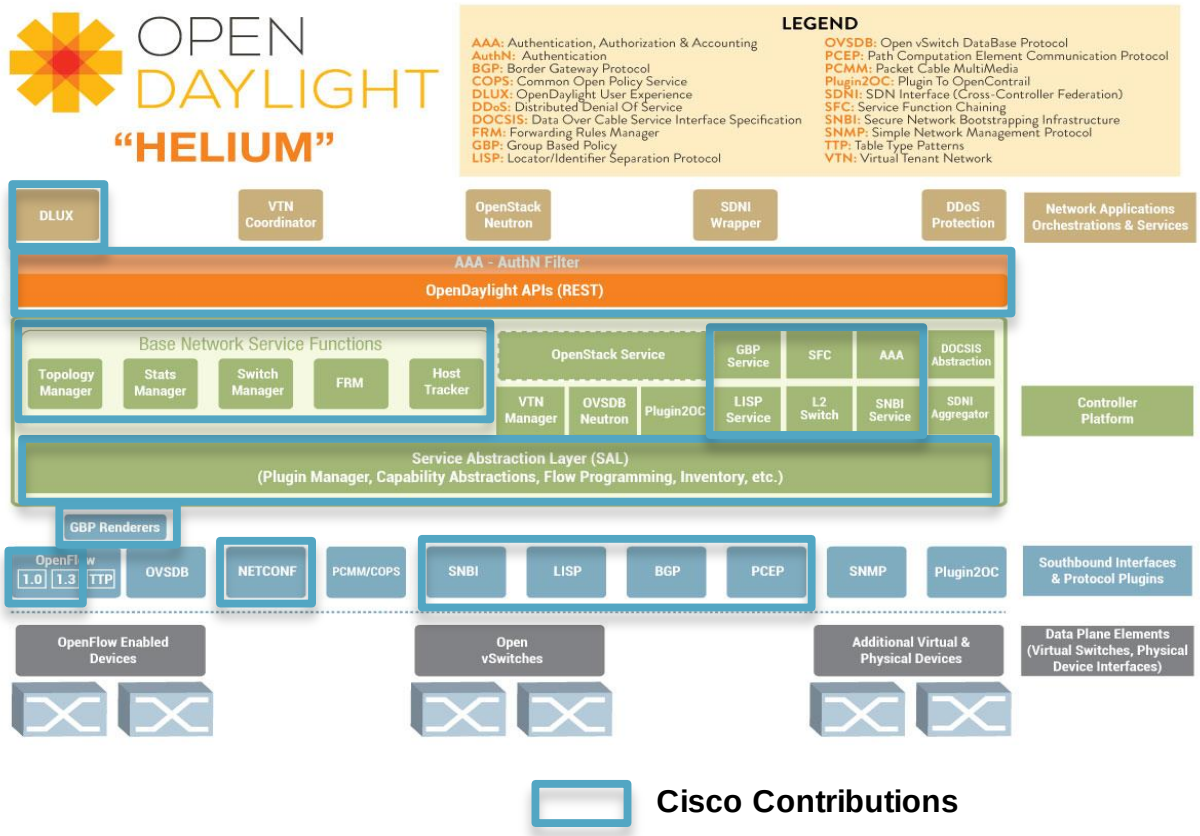
- Packet Processing Agent
- Operates in user-space

# Virtual Topology Forwarder (VTF)

- Industry's first user space forwarder
- Multi-threaded, 10G in single core
- IPv6 Support
- ISSU, Fault Isolated
- Highly Portable
- 64-bit, endian clean

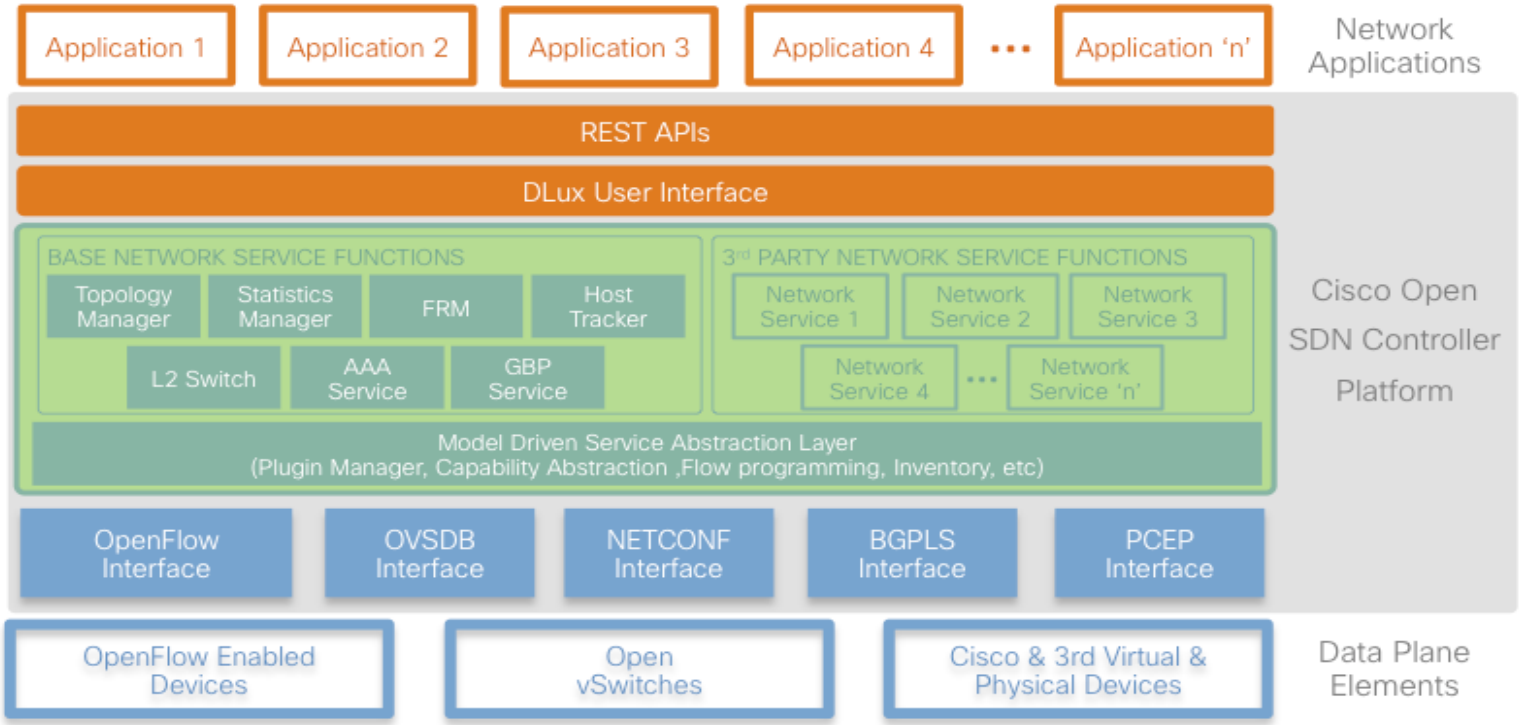


# OpenDaylight Controller



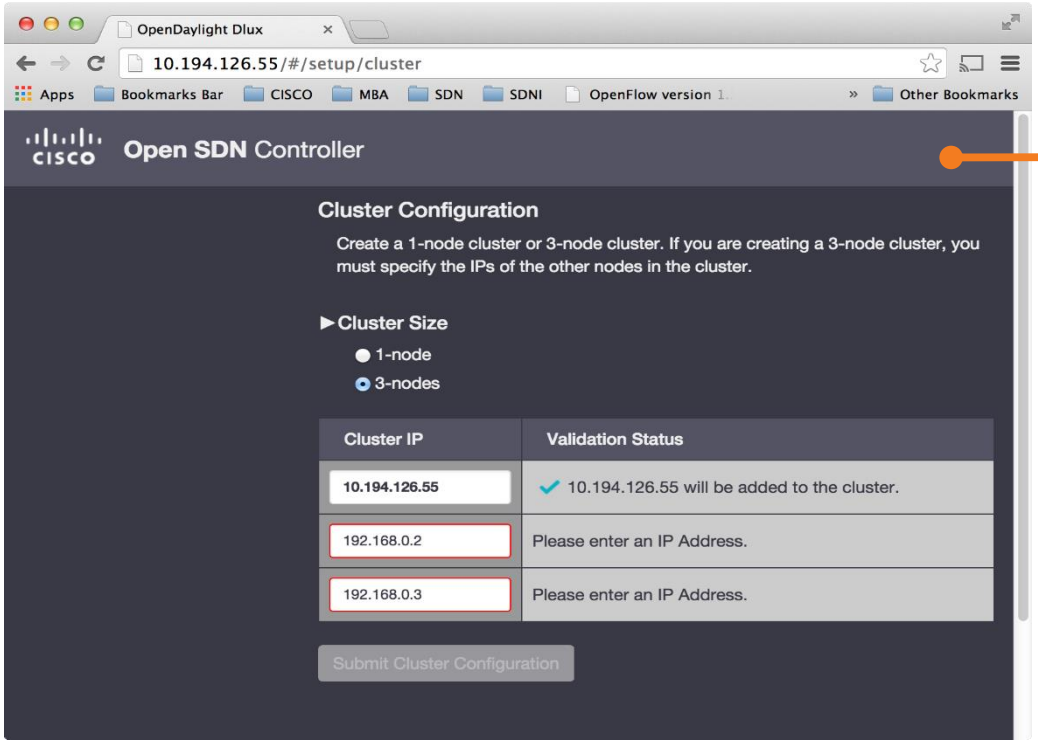

- Open platform for network programmability
- Enables SDN for networks at any size and scale
- New “Helium” release delivers new user interface and a much simpler and customisable installation process
- Customer can add value at any layer (Apps, Network Services, SB Plugins)

# Cisco Commercial Distribution of OpenDaylight




Cisco Open SDN Controller

# Cisco Open Controller Deployment Experience




## One Click Installation

Open Virtualisation (OVA) Format

VMware ESXi and Oracle Virtual Box support

Single “click” to select standalone vs clustered installation

Seamless software upgrades

# Application Centric Infrastructure (ACI)

## Open + Secure

## Apps + Infrastructure

SharePoint
Exchange
SAP
ORACLE


## Physical + Virtual + Containers

## On-Premises + Cloud

**Application Oriented Policy =** Operational Simplicity

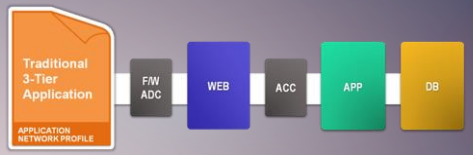
# Application Centric Infrastructure (ACI)

## Rapid Deployment of Applications onto Networks with Scale, Security, Full Visibility

Embedded Stateless L4 Firewall (zero trust)
Tenant Isolation
Group-based Security Policy* (3rd party included)
Whitelist Policy Enforcement
Fabric High-Availability

**NEXUS 9500, 9300 and AVS**

Declarative Policy Model
Fully Object-oriented and Open
Application Centric Desired State
Packaged deployment
Use, re-use and decommission with audit trails




**APPLICATION CENTRIC POLICY**

Centralised Management
Role-Based Access
Audit Logs
Health Monitoring
Open REST APIs


**CONTROLLER**

**ACI**

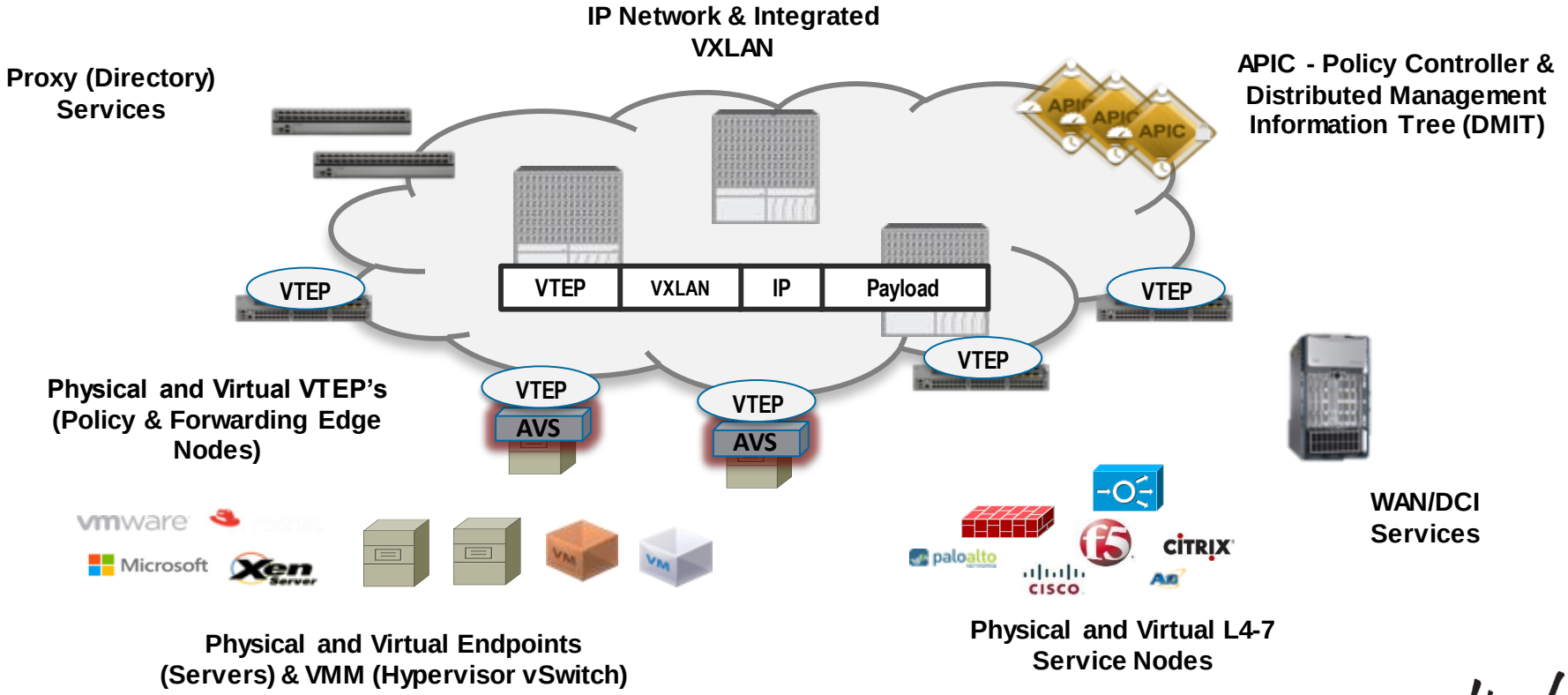




Cisco live!

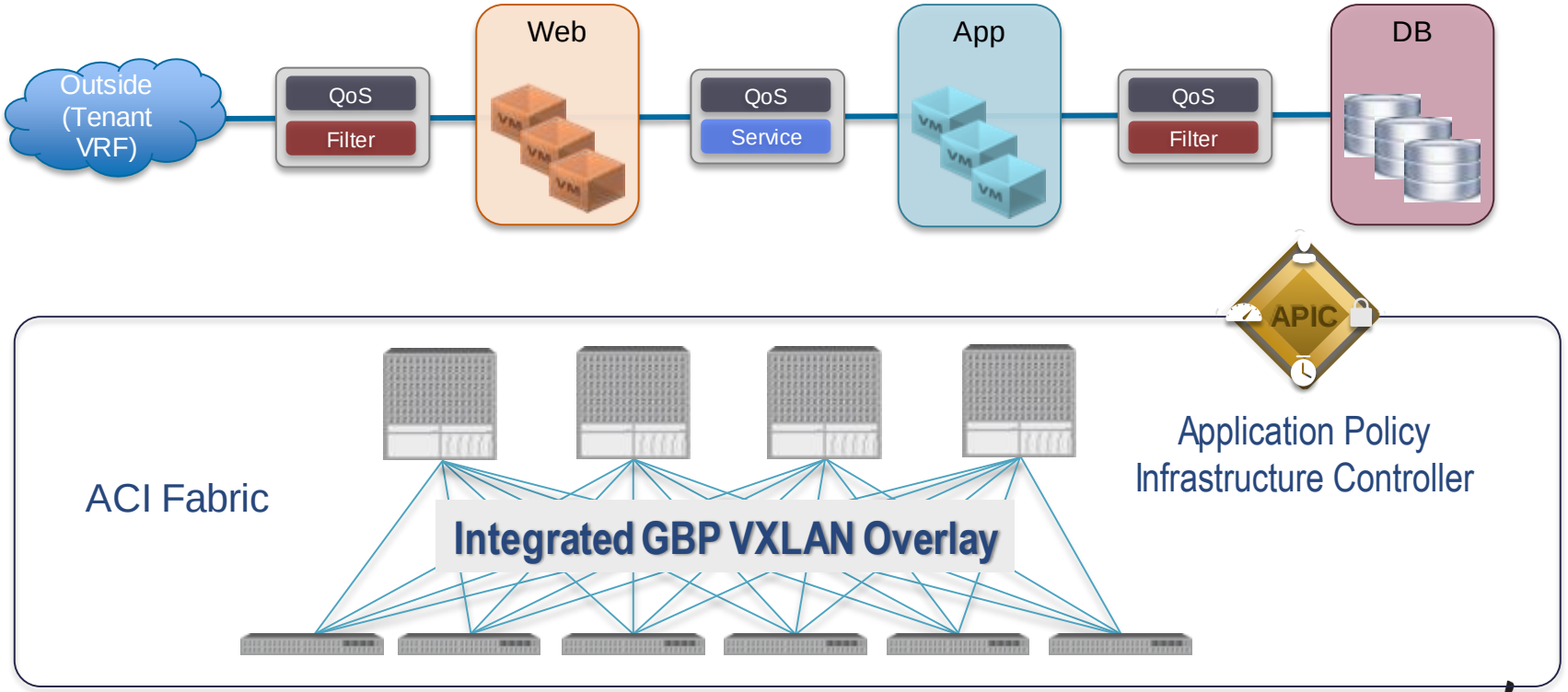
# ACI – Network Components

## A Policy Based IP Network

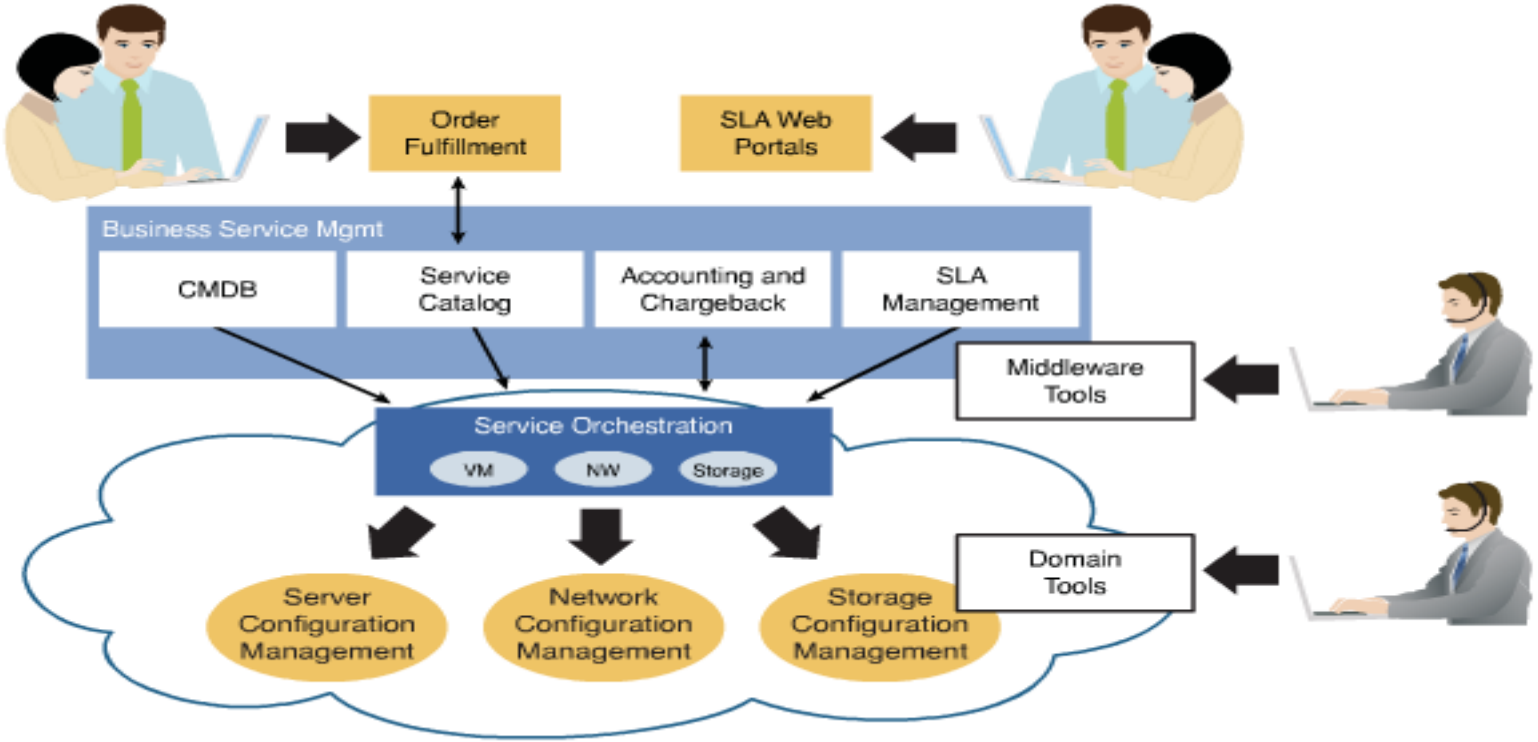


# ACI – Policy Components

## Logical Network Provisioning of Stateless Hardware

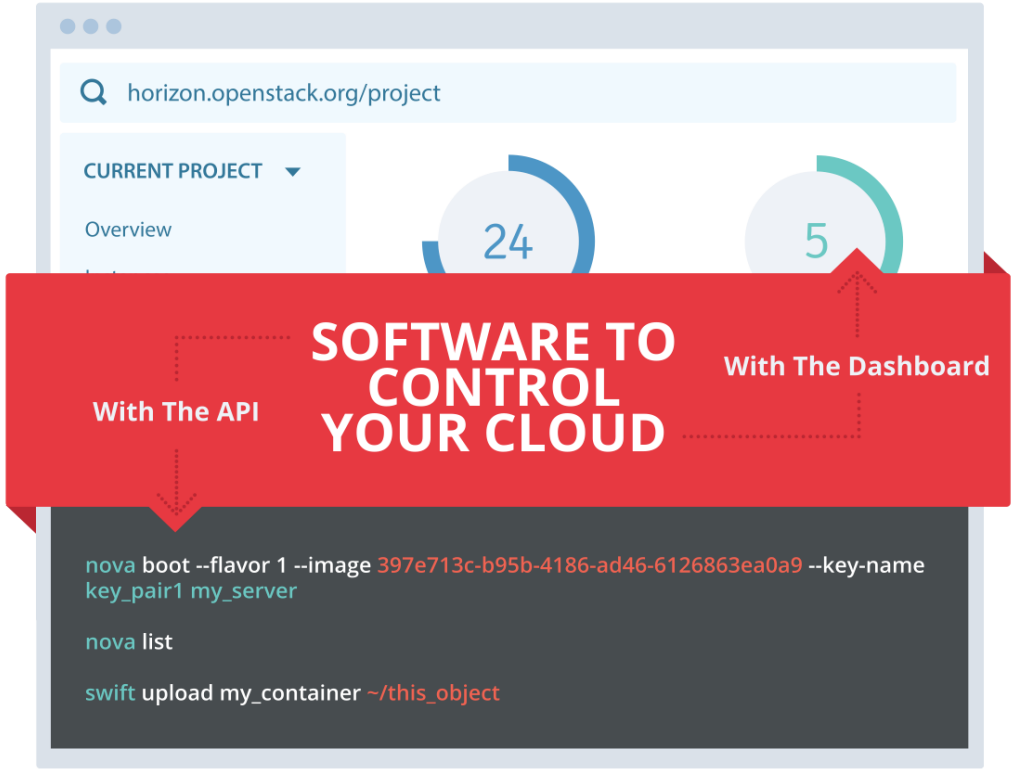


# Orchestration in the IT World



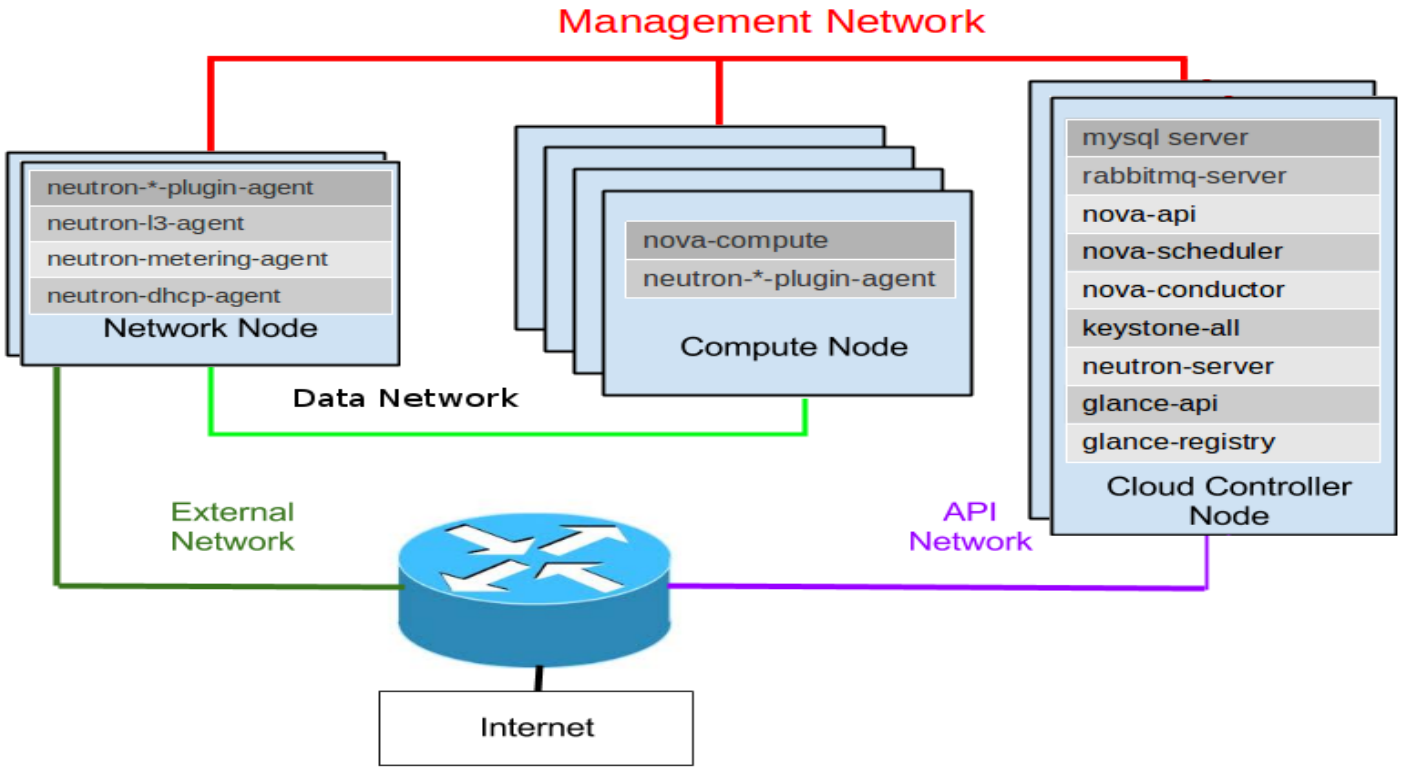
# OpenStack Cloud Computing Software

- Freely available, open source software allowing anyone to build their own private or public clouds.
- Open source and open APIs allows the customer to avoid being locked in to a single vendor
- Built by a growing community of contributors
- Opportunities for vendors to develop their own solutions and services

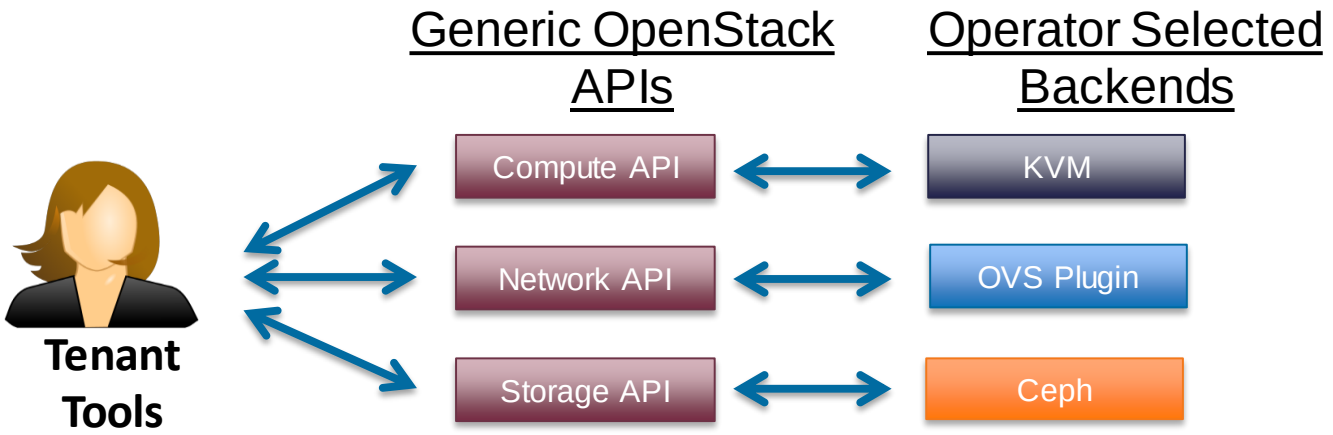



http://www.openstack.org/assets/welcome-guide/OpenStackWelcomeGuide.pdf

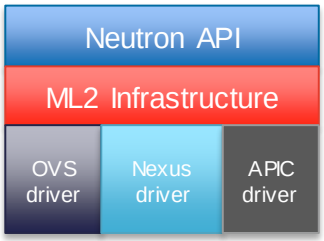
# OpenStack Architecture



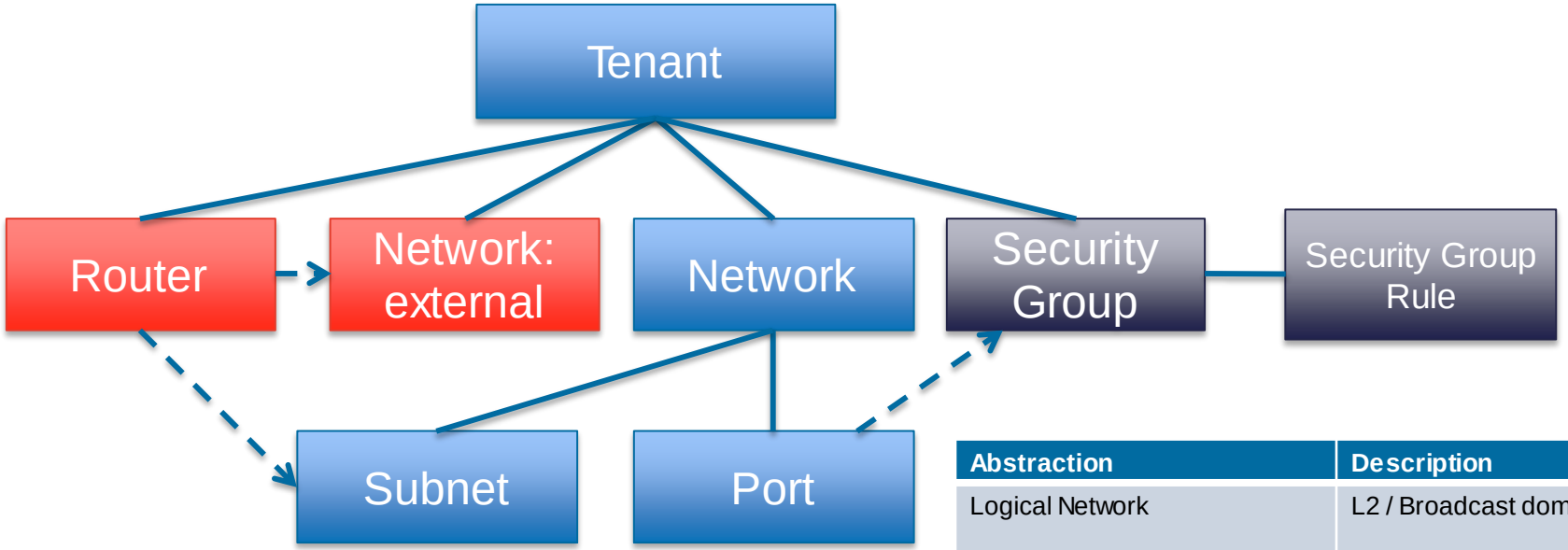
# OpenStack Plugin Model




- Cisco plugin supports multiple sub-plugins
- Modular L2 (ML2) evolution of Neutron
- Allow multiple plug-ins to exist as sub-plugin drivers



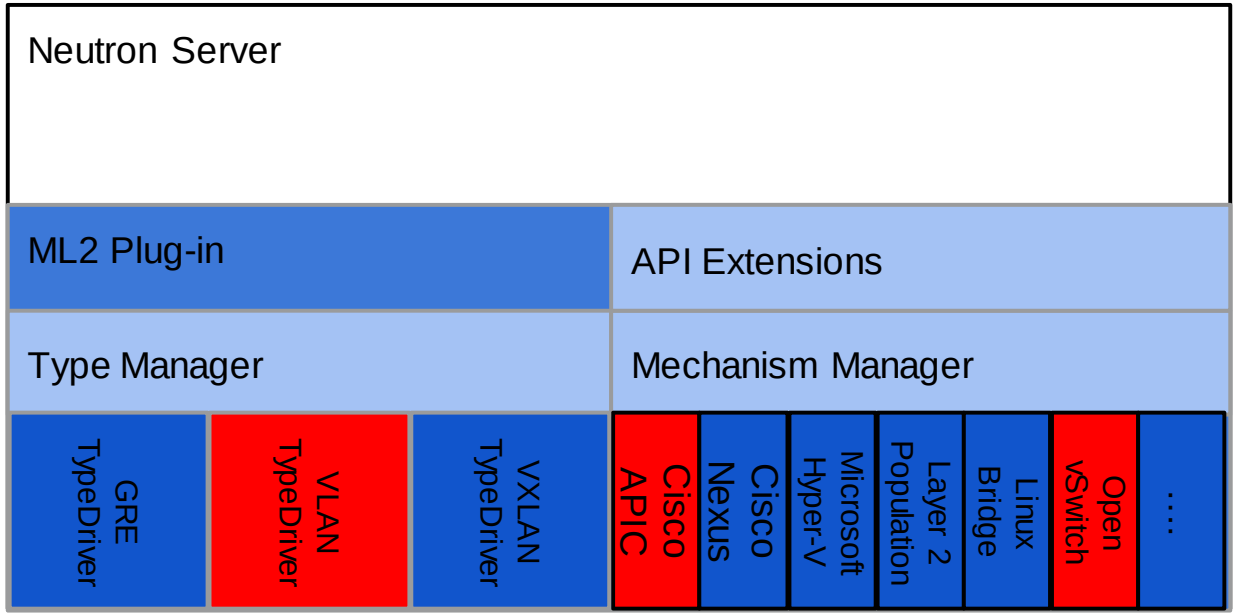
# OpenStack Neutron Model




L3 + External Net Extension | Core API | Sec Grp Extension

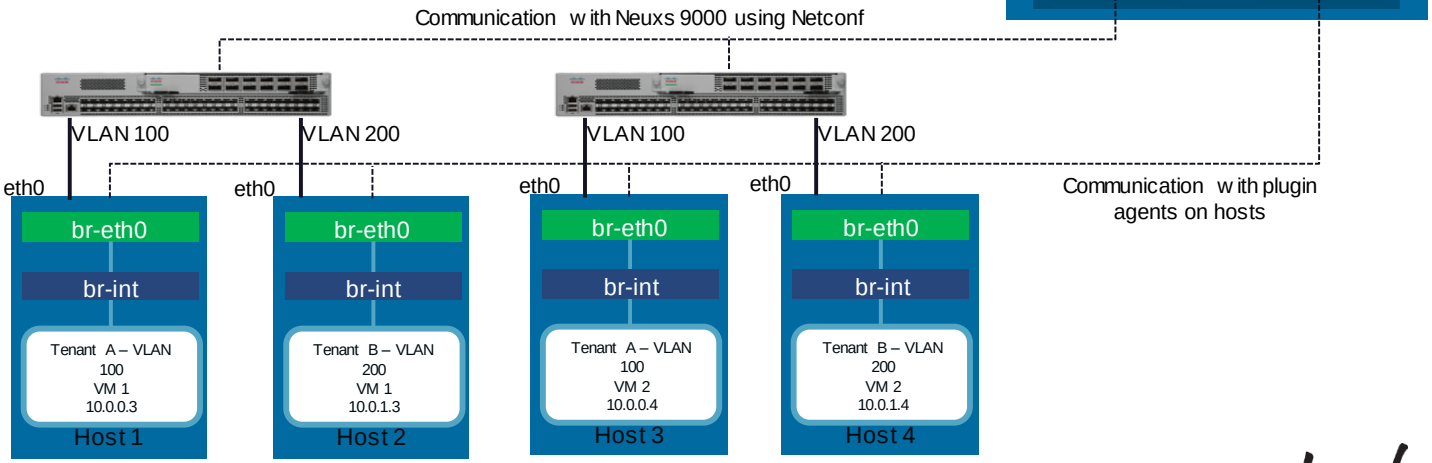
| Abstraction | Description |
|---|---|
| Logical Network | L2 / Broadcast domain |
| Logical Router | L3 domain |
| Subnet | Subnet (OpenStack IPAM / DHCP) |
| Security Group | Group-based ACL |

# OpenStack ML2 Architecture Diagram

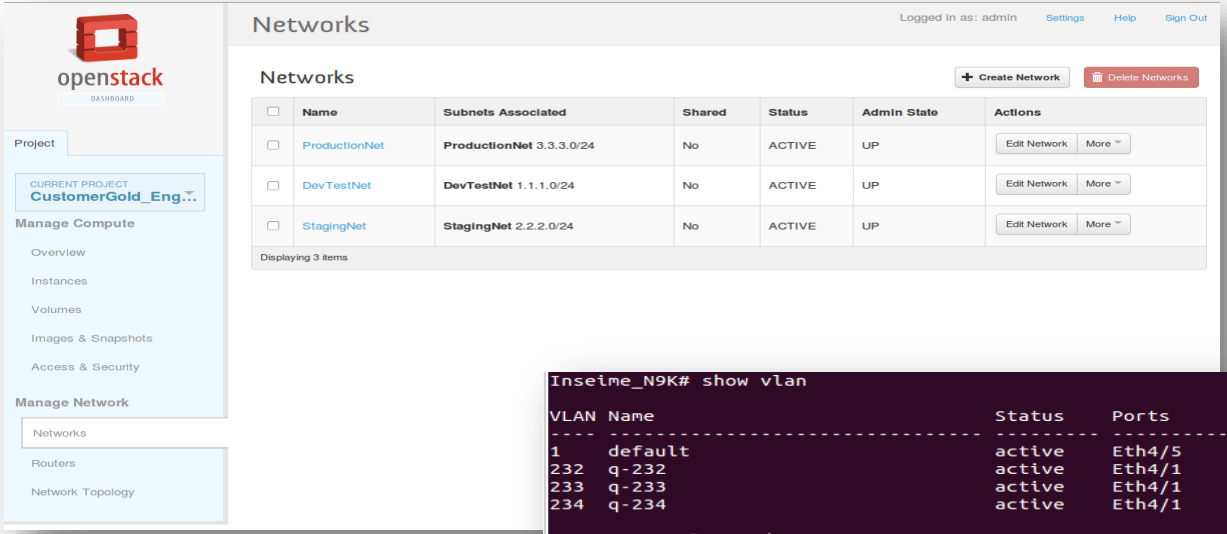


# OpenStack Network (Neutron) Plugin

- Enables fully automated compute, storage and network resource orchestration
- Support for Havana, Icehouse & Juno OpenStack releases
- Enable VLAN-based tenant separation
- Enhance efficient resource usage
- Leverages NX-OS NetConf-XML programmatic interface



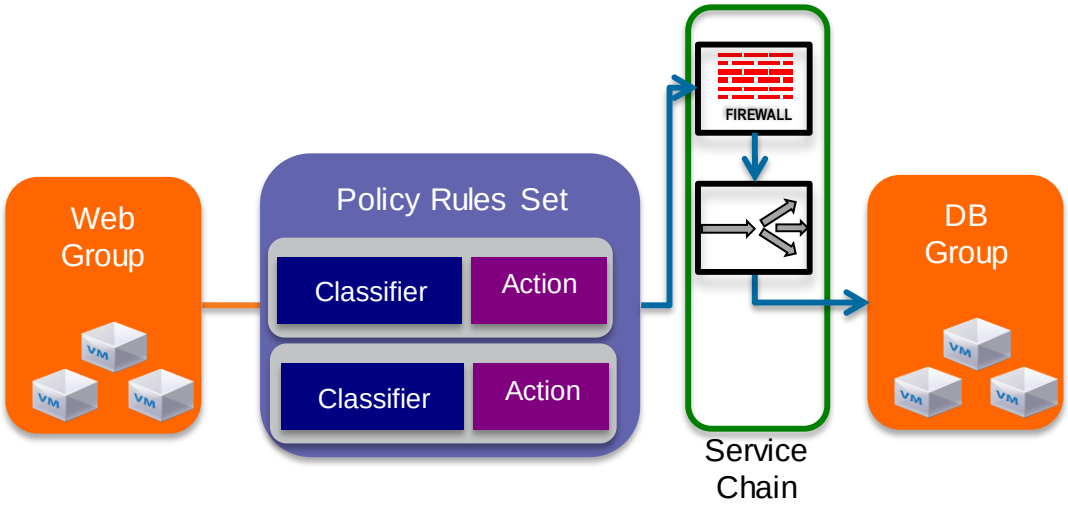
# OpenStack Network (Neutron) Plugin



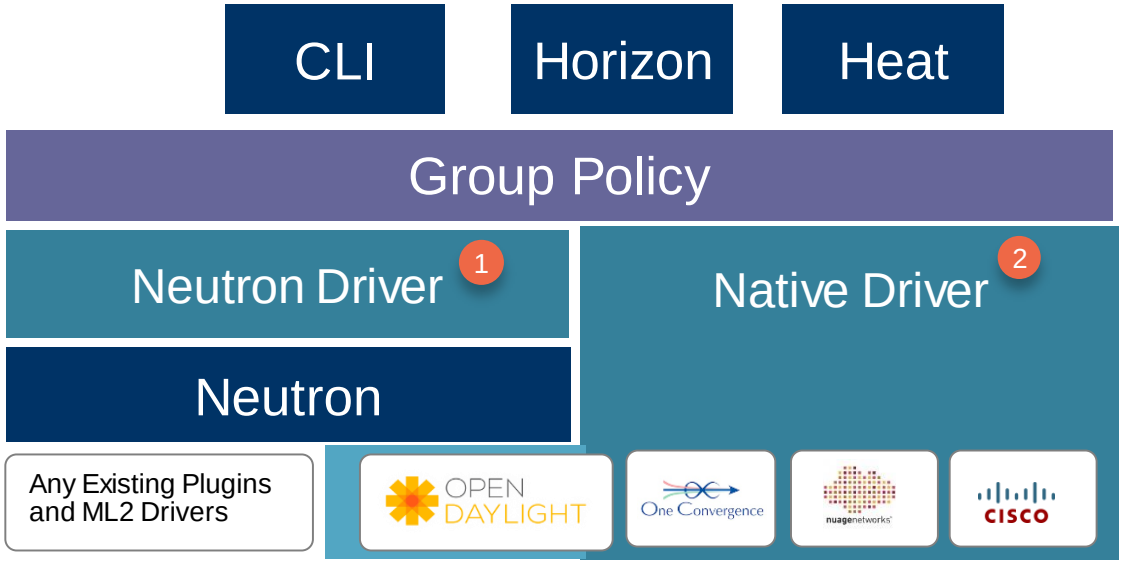

```
Inseime_N9K# show vlan

VLAN Name                                Status    Ports
---- ----------------------------------- --------- -------------------------------
1    default                             active    Eth4/5
232  q-232                               active    Eth4/1
233  q-233                               active    Eth4/1
234  q-234                               active    Eth4/1

VLAN Type  Vlan-mode
---- ----- ----------
1    enet  CE
232  enet  CE
233  enet  CE
234  enet  CE

Remote SPAN VLANs
------------------------------------------------------------------------------

Primary  Secondary  Type              Ports
-------  ---------  ----------------  ------------------------------------------
```

# Introducing Group-Based Policy

- Intent-based API for describing application requirements
- Separates concerns of tenants and operators
- Captures dependencies between tiers of an application
- Plugin model
  - Supports mapping to Neutron APIs
  - Supports “native” SDN drivers



# OpenStack GBP Overview

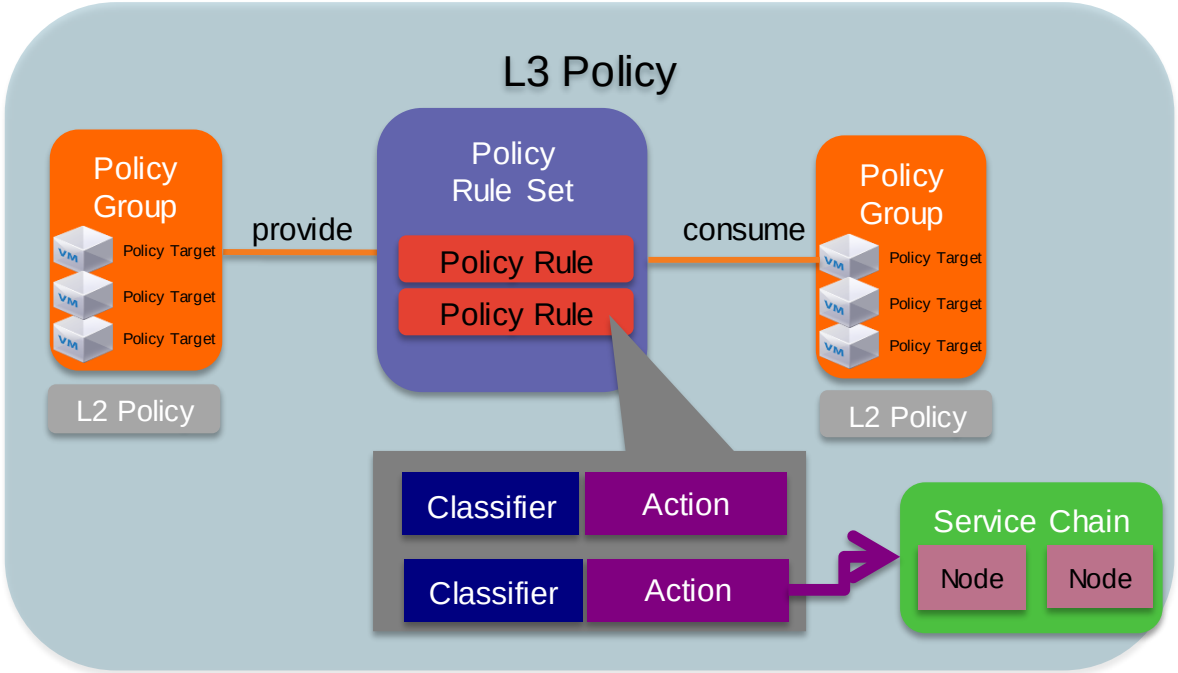



**Open model that is compatible with ANY physical or virtual networking backends**

1. Neutron Driver maps GBP to existing Neutron API and offers compatibility with any existing Neutron Plugin

2. Native Drivers exist for OpenDaylight as well as multiple vendors (Cisco, Nuage Networks, and One Convergence)

# Group-Based Policy Model




**Policy Group**: Set of endpoints with the same properties. Often a tier of an application.

**Policy RuleSet**: Set of Classifier / Actions describing how Policy Groups communicate.

**Policy Classifier**: Traffic filter including protocol, port and direction.

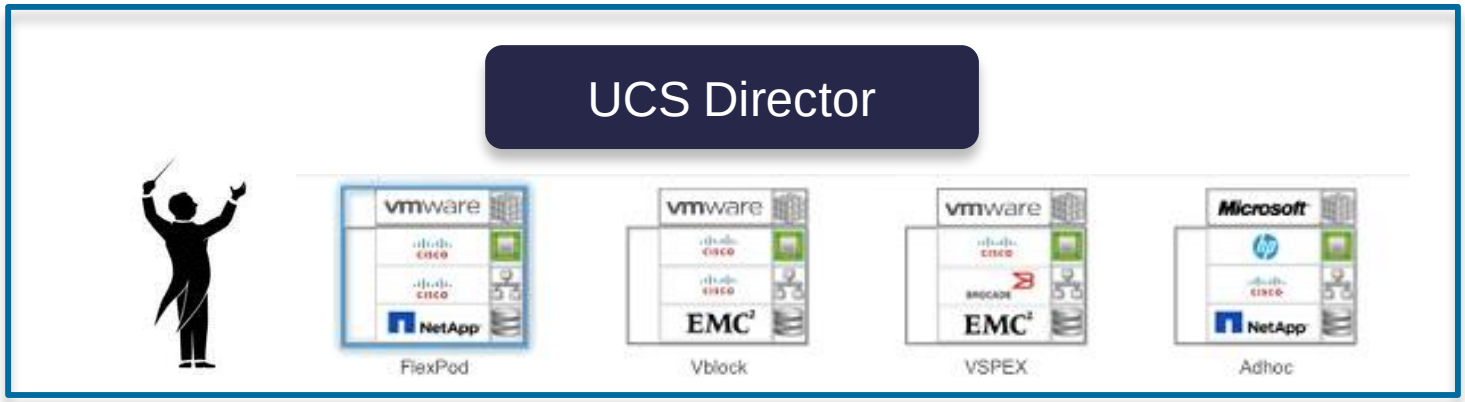
**Policy Action**: Behaviour to take as a result of a match. Supported actions include “allow” and “redirect”

**Service Chains**: Set of ordered network services between Groups.

**L2 Policy**: Specifies the boundaries of a switching domain. Broadcast is an optional parameter

**L3 Policy**: An isolated address space containing L2 Policies / Subnets

# Cisco UCS Director Value




- Acts as orchestra conductor across: compute, network, storage & virtualisation
- Replaces manual management of each layer with automated workflows
  Helps to remove silos from IT teams and IT resources
- IT manages data centre resources as single “team” with unified management
  Across physical and virtual resources
- Unfettered IT admins can now focus on new services for business

# Seamless Infrastructure Management - UCS Director



Manage across facilities

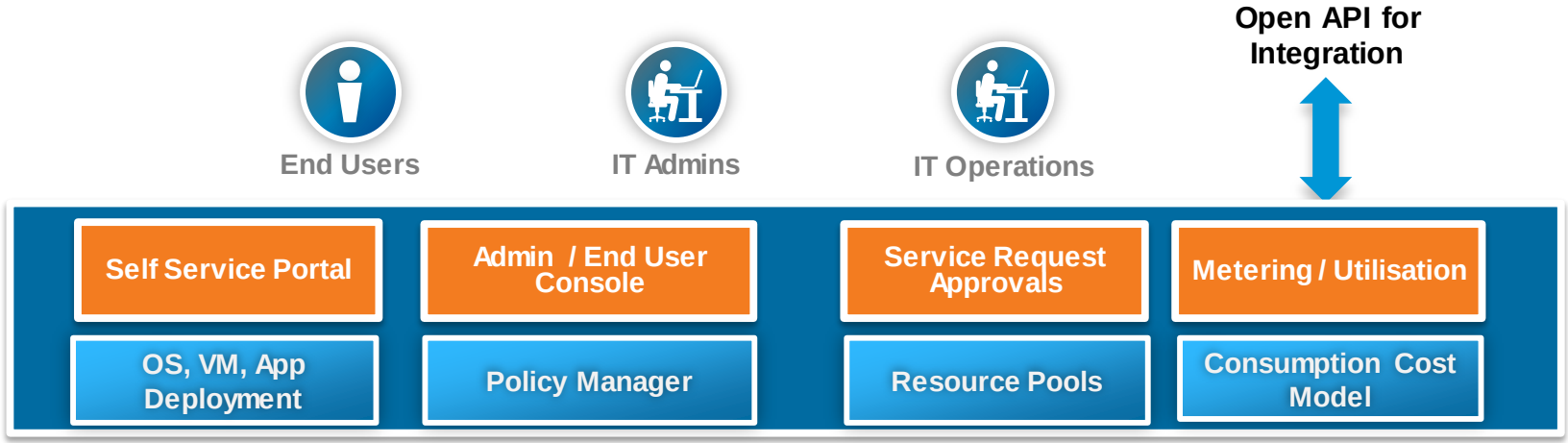
- Unify compute, network, storage – physical and virtualised
- Rapid, low-cost infrastructure deployment based on application requirements
- Consistent, robust deployment every time

Application optimised infrastructure

- New! Hyper-V support for Citrix networking
- New! VACS Integrated application containers
- New! UCS Mini and E-Series support

# UCS Director: Multi-Vendor Support

## Agility and Simplicity for Virtualised and Bare-Metal IT Services




**Centralised Lifecycle Management of Physical and Virtualisation Infrastructure**

# UCS Director - How Does it Work?

- Abstraction of applications, hardware and software into programmable tasks
- Tasks used to create automated workflows
  - API attached to task eliminating scripting
  - Pre-validated, run immediately after creation
- Workflows published into service catalog
- Dynamic orchestration that keeps business moving
- No other vendor that offers this capability


Workflow to provision VMWare Virtual Machine using ISO Image

# UCS Director

## Centralised Data Centre Automation

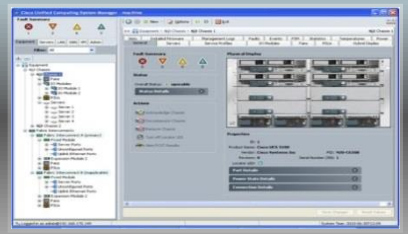
FlexPod

Vblock

VSPEX

Adhoc

**UCS MANAGER/ CENTRAL / IMC Supervisor**



- Manage single, multiple UCS domains
- Branch/remote offices

Network Service Containers

**APIC**


ACI & Nexus Fabric

**VACS**


Pre-defined virtual network services in 6 steps

Heterogeneous & UCS-based Integrated Infrastructures






Cisco live!

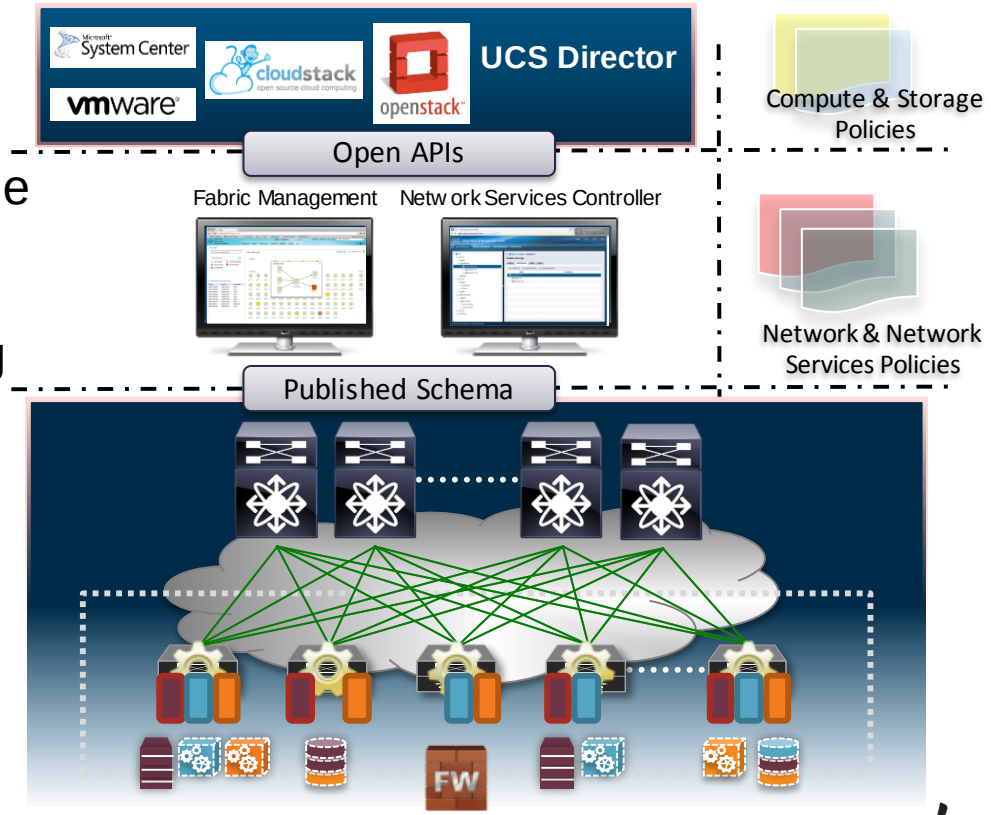
# Conclusion – Data Centre Design Transition

Cisco live!

# Workload Automation and Open Environment

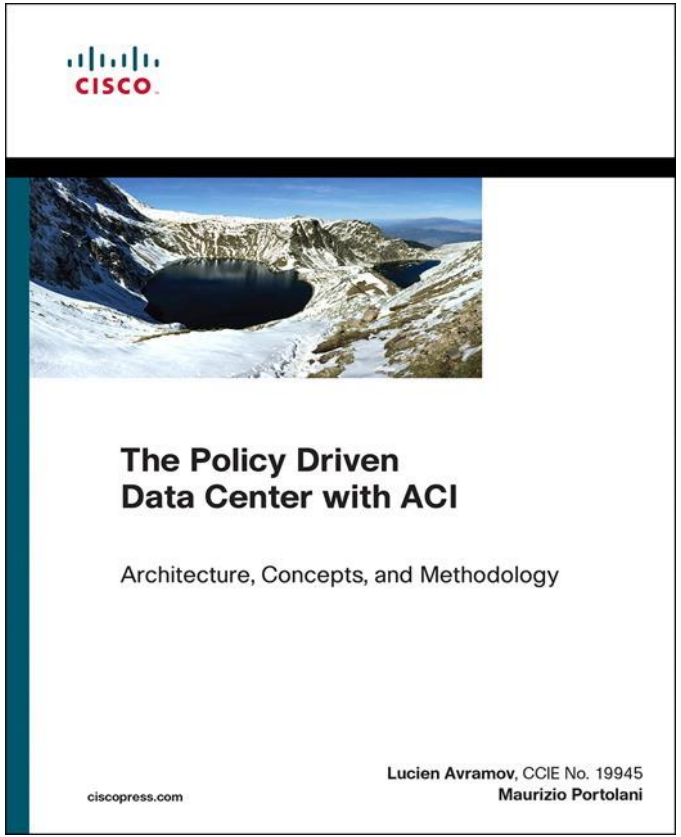
## Advantages

- Any workload, Anywhere, Anytime
- Open Integration: Orchestration
- Automated Scalable Provisioning
- Workload aware fabric



# Call to Action

- Visit the World of Solutions for
  - Cisco Data Centre
  - Walk in Labs
  - Technical Solution Clinics
- Meet the Engineer
- Lunch time Table Topics
- DevNet zone related labs and sessions
- Recommended Reading: for reading material and further resources for this session, please visit www.pearson-books.com




Q & A

# Complete Your Online Session Evaluation

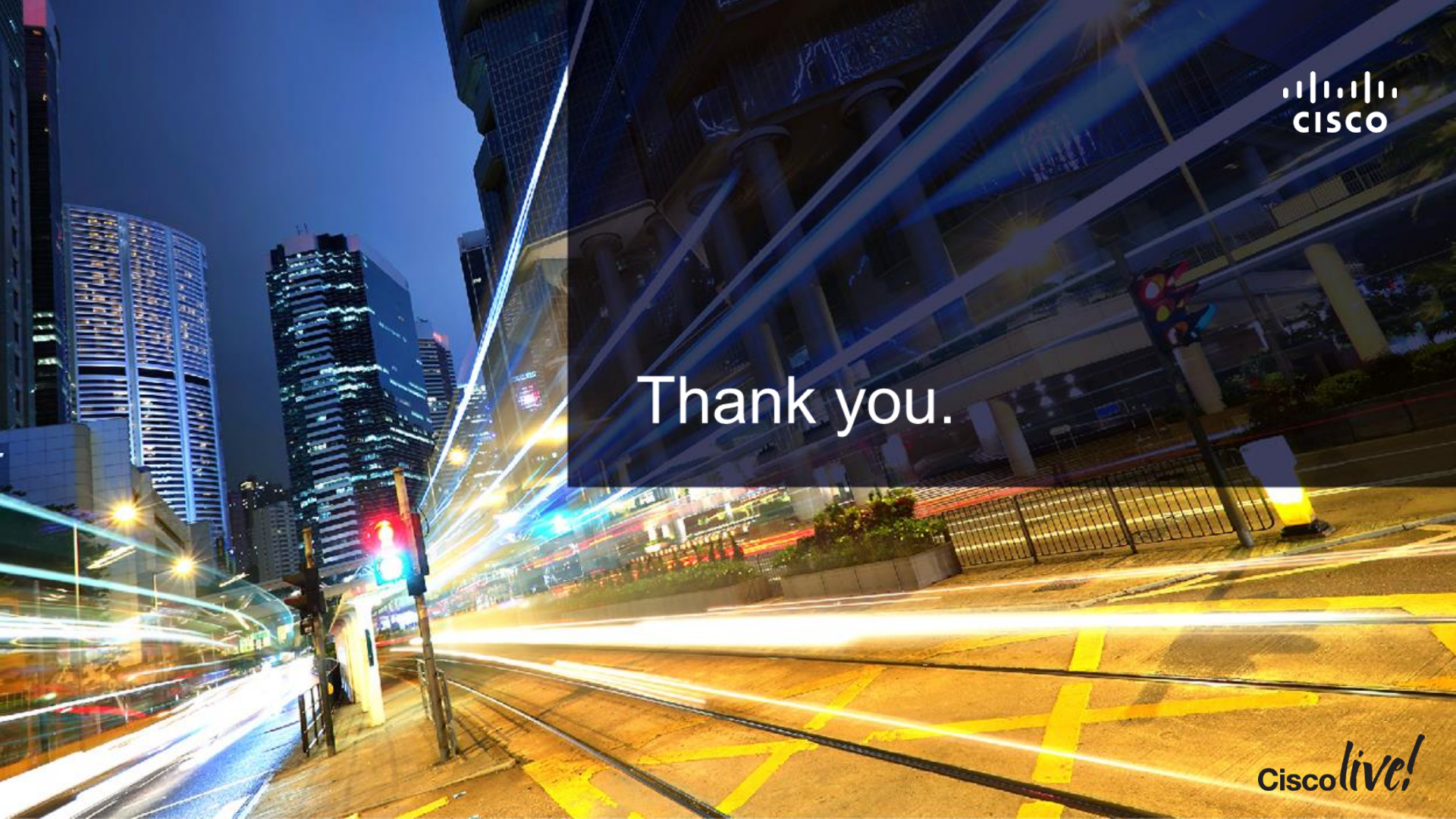
**Give us your feedback and receive a Cisco Live 2015 T-Shirt!**

Complete your Overall Event Survey and 5 Session Evaluations.

- Directly from your mobile device on the Cisco Live Mobile App
- By visiting the Cisco Live Mobile Site http://showcase.genie-connect.com/clmelbourne2015
- Visit any Cisco Live Internet Station located throughout the venue

T-Shirts can be collected in the World of Solutions on Friday 20 March 12:00pm - 2:00pm

**Learn online with Cisco Live!**
Visit us online after the conference for full access to session videos and presentations. www.CiscoLiveAPAC.com


Thank you.


CISCO