IOS-XR7 Innovations
SZTP, App-Hosting, Programmability, Security

Akshat Sharma, Technical Leader, TME


BRKSPG-2024
Agenda

- The Network OS Overton Window
- Ever-Changing Web and SP Deployment Landscape
- Security + Automation = Hitting the sweet spot!
- Ownership Establishment Basics (RFC 8366)
- Secure ZTP (SZTP) based on RFC 8572
- Application Hosting: Making life easy on Fixed and Modular platforms
- Programmability: APIs at every layer of the Network Stack!
- Security/Trust: Trust tied to HW → Secure Boot + Runtime Security!
“The Overton Window for the Networking Industry is expanding”
The Expanding Overton Window for Network OS Features

- Near-Zero Automation
- No box failure acceptable
- No redundancy in network design
- Single "neck to choke" for support
- Stringent security (vendor enabled)
- SMEs and manpower for deployment

- CLI (expect style) automation
- Secure Boot with BIOS protection
- Chip/HW protection
- Secure Asset Transfer
- Trust tied to HW

- Secure TLS based APIs (netconf, gRPC etc.)
- Redundancy protocols, Backup paths
- MPLS
- SNMP

- ZTP is a must

- ISSU is a must
- Complex one-off features
- Third-party operators for network installations

- APIs for CLI automation
- UEFI secure boot
- Vendor Features to solve critical problems
- RSVP

- Software-Only Security Approach
- Redundancy protocols, Backup paths
- MPLS
- SNMP

- Secure TLS based APIs (netconf, gRPC etc.)
- Removable/ modular Features
- ZTP is a must

- Disaggregation of HW & SW
- Custom Routing Protocols
- Controllers (only) for Traffic-Engineering

- True “SDN”
- Manage on your own
- >3000:1 device/admin ratio
- Minimal Security (secure network boundaries)

- Less automation, High Security
- More automation, Less Stringent Security

- 2019+
- <2014

- 2014-2019
- Less automation, High Security
- More automation, Less Stringent Security

- Completely Automated Deployment (Day0 – DayN)
- APIs at every layer of Stack
- On-box Apps
- SRv6
- ISSU not required

- 2019+
- <2014

- 2014-2019
- Less automation, High Security
- More automation, Less Stringent Security

- Cisco Live! BRKSPG-2024
Ever-Changing Web and SP Deployment Landscape
Access +5G Deployments:
Large number of XR7 devices with NCS540L and NCS55xx
Facebook’s Express Back Bone (EBB) Network

• Centralized BGP Route Injectors
• sFlow collectors to feed active demands into the Controller
• Traffic engineering controller, to compute and programs optimum routes
• Open/R agents running on-box to provide IGP and messaging functionality.
• LSP agents, also running on-box to interface with the device forwarding tables on behalf of the central controller.

Google’s Espresso Metro

IOS-XR’s lock-step Journey
IOS-XR’s answer to the expanding Overton Window

2014

Present

Data Stores
- redis
- mongoDB
- cassandra
- MySQL
- Amazon Web Services

App Servers
- Heroku
- NGINX
- Amazon Web Services

Services
- Twitter
- PayPal
- apigee

Front End
- Sencha
- HTML
- iOS

Service Orchestration, Automation & Analytics
- Docker
- Mesosphere
- Splunk
- Netflix

Infra Software
- Tailor
- Saltstack
- Kibana

Silicon Abstraction
- DPDK
- P4
- Open NSL
- Open Data Plane

Network Protocols
- BGP
- ISIS
- OSPF

Silicon Abstraction, Infra, Software, Service Orchestration, Automation & Analytics

2014

Present
The IOS XR Evolution Journey

**IOS XR**
- 32-bit QNX-based
- SMU based patches
- Highly reliable, large scale routing
- Core and edge use cases

**IOS XR 6+**
- 64-Bit Linux-based
- Merchant and Cisco silicon
- Cloud-Scale Ready!
  - Model-driven management + Telemetry
  - Automated device onboarding – ZTP, iPXE
  - Hosted third-party software

**IOS XR 7+**
- Advanced flexibility for custom use cases
  - Model-driven APIs at all layers
- Security enhancements – Establish trust in the HW, SW & Network
- Simplification & Flexible Consumption
  - Disaggregated SW Offer
  - Optional SW packages

**OS Evolution**
Security + Automation = Hitting the sweet spot!
Security + Automation: Marriage NOT made in heaven!

- Secure TLS based APIs (netconf, gRPC etc.)
- Removable/ modular Features
- ZTP is a must

- CLI (expect style) automation
- Secure Boot with BIOS protection
- Chip/HW protection
- Secure Asset Transfer
  - Trust tied to HW

- Secure ZTP

- Completely Automated Deployment (Day0 – DayN)
- APIs at every layer of Stack
  - On-box Apps
  - SRv6
  - ISSU not required

Unthinkable
- Acceptable
- Sensible
- Popular

Sensible
- Acceptable

Radical
- Acceptable
- Sensible
- Popular

Secure/Trusted Application-Hosting

2019+
Security and Automation: Finding the sweet spot

- It’s pretty well known that Security and Convenience are usually at loggerheads.
- Precisely why marrying concepts from opposite sides of the Overton window timeline is difficult.
- Secure ZTP (RFC 8572) is a big step forward in the industry for large Datacenter and 5G deployments.
- So is the ability to run trusted third-party apps and binaries.
- RFC 8366 details some of Ownership establishment methodologies that make these capabilities possible.
Ownership Establishment Basics (RFC 8366)
Cisco TAm – Hardware-based Trust Anchor
Available on all shipping XR7 platforms (ACT2/Aikido chips)

- Anti-Theft and Anti-Tamper Chip Design
- Built-In Crypto Functions
- Hardware Entropy for RNG*
- Secure Storage

- Hardware designed to provide both End-user and supply chain protections
  - End-user protections include highly secure storage of user credentials, passwords, settings.
  - Supply chain protections -- Cisco SUDI (secure unique device identifier) inserted during manufacturing
- Secured at Manufacturing. No user intervention required
- Ideal for embedded computing like routers and Wi-Fi access points
Unique hardware Identity (SUDI)

“How do I know this is really my router?”

- Unique cryptographic key embedded in hardware trust anchor module within every IOS XR Router
  - Secure Unique Device Identifier (SUDI)
  - Provides 802.1AR Secure Device Identity
  - Immutable key imbedded in Trust Anchor Module at time of manufacture
  - Signed by Cisco for proof of authenticity
  - Includes PID and Serial number of device
- Cryptographically strong identification of remote hardware
- Establishes unique, immutable hardware identity
What’s in the TAm Chip?

TAm’s core functionality

• Microloader
• UEFI DB for Cisco’s keys to validate the boot artifacts and OS
• Imprint DB for Chipguard to store the hashes of ECIDs of CPU & ASIC
• Encryption key for hybrid TAm storage (on disk)
• SUDI certificate and Attestation Key
• PCRs for extending hashes (boot and run time)
• Persistent across reloads and Disk Wipeouts

Additional Functionality (Uses on-chip Secure Storage)

• Owner Certificate (OC)
• Sensitive Feature Control Flags
  • Enable/disable Secure ZTP
  • Enable/disable anti-theft protection
Establishing Ownership on device: Owner/Customer Certificate.
Establishing Ownership on a new device: Owner Certificates

• By default, Cisco hardware trusts only Cisco as a root CA through certificates burnt into TAm by Cisco Manufacturing

• To extend trust on a new network device the network operator needs to burn their own certificate into the hardware TAm

• To do this, the device must accept the chain of trust associated with the owner cert – cue RFC 8366

How does a router trust an owner/customer certificate

Using Ownership Vouchers (RFC 8366)
Ownership Voucher (O.V.) (RFC 8366)

Yang model for O.V.

```
module: ietf-voucher

yang-data voucher-artifact:
  +---- voucher
    +---- created-on : yang:date-and-time
    +---- expires-on?: yang:date-and-time
    +---- assertion : enumeration
    +---- serial-number : string
    +---- idevid-issuer?: binary
    +---- pinned-domain-cert : binary
    +---- domain-cert-revocation-checks?: boolean
    +---- nonce?: binary
    +---- last-renewal-date?: yang:date-and-time
```

General purpose voucher used to establish ownership in SZTP (RFC 8572) and non-ZTP scenarios (running/provisioned systems)

- CMS artifact signed by the Manufacturer (Cisco) for each HW-TAm enabled node.

- Two Node (read Route-Processor/RP) identifiers:
  - Serial Number: Serial number of the router whose ownership must be established.
  - Pinned-domain-cert (PDC): A Customer root or intermediate certificate that acts as the chain of trust for other intermediate certs used by the customer.

How do you get the Owner cert into the TAm?

(XR Release 7.5.1)

Based on RFC 8366, Ownership vouchers (O.V.) are used to establish a trust chain for Owner Certificates.

Using preferred onboarding technique:
1) CLI/API
2) SZTP

1) Using CLI/API, accept an owner certificate along with an ownership voucher (OV signed by Cisco)

2) Using SZTP, a ZTP server offers bootstrapping data that contains an OV and an owner certificate. SZTP automatically burns owner certificate into the TAm.
ok, I’ll bite.

How do I get an O.V. for my router?
O.V. Generation: MASA Server for SZTP
(Available Mid 2021)

- Automated O.V. generation per device Serial Number (i.e., up to 2 O.V.s per device, one for each RP) in Real time
- **MASA (Manufacturer Authorized Signing Authority)** is a cloud Service that is operated by the Manufacturer (Cisco) to help ratify that Serial Numbers actually belong (i.e., were sold) to the requesting customer
- **Once ratified, the OVs are generated and downloaded by the Secure ZTP Server (e.g., Cisco Crosswork SZTP server) via the cloud API**
- This is a one-time activity for each device or a group of devices
O.V. Generation: MASA Server for non-ZTP Scenarios

(Available Mid 2021)

- Automated O.V. generation per device Serial Number (i.e., up to 2 O.V.s per device, one for each RP) in Real time
- **MASA (Manufacturer Authorized Signing Authority)** is a cloud Service that is operated by the Manufacturer (Cisco) to help ratify that Serial Numbers actually belong (i.e., were sold) to the requesting customer
- Once ratified, the OVs are generated and downloaded by the customer via the cloud API
- This is a one-time activity for each device or a group of devices
Secure ZTP (SZTP) based on RFC 8572
Automated Provisioning using Classic ZTP

ZTP Artifacts
- Images to Download
- CLI Configuration
- ZTP script:
  - Native Python scripts
  - Native bash scripts

Automatically initiated over Management port and all production/data ports

Initial DHCPv4/v6 and/or SLAAC (IPv6) + DHCPv6 messages

Artifact Downloaded
Access Networks and 5G Deployment Considerations

• **Tree based Build-out is the ideal strategy:**
  - No out-of-Band Management network to work with
  - Already provisioned device acts as a DHCP relay for the next device in the tree

• **Security is critical:**
  - Access devices are typically in insecure locations
  - Would greatly benefit from secure device onboarding techniques.

• **Vlan discovery:**
  - Data (Production) ports would be utilized for ZTP
  - These data ports might need to communicate with upstream device over a VLAN
Security Considerations for ZTP

**Router/Client Validation**
Server must validate router/client cert (SUDI cert) before offering artifacts/secrets/configs

**Network/Server Validation**
Router/client must validate the server offering artifacts

**Artifact Validation**
The artifact downloaded from the ZTP/Web server must be validated before being loaded/executed
Secure ZTP (SZTP) workflow (based on RFC8572)

ZTP Artifacts
- Image
- CLI Configuration
- ZTP script:
  - Native Python scripts
  - Native bash scripts,

IOS-XR

DHCP Server

ZTP Server (Bootstrap Server 1)

Restconf Server

Web Server (Artifacts)

Initial DHCPv4/v6 Messages - New option
143/136

SZTP YANG model interaction

Artifact downloaded

From ZTP” to “Secure ZTP” with RFC8572

Router/Client Validation
Server must validate router/client cert (SUDI cert) before offering artifacts/secrets/configs

Network/Domain Validation
Router/client must validate the server offering artifacts

Artifact Validation
The artifact downloaded from the ZTP server must be validated before being loaded/executed

Artifact Validation
The artifact downloaded from the ZTP server must be validated before being loaded/executed

ZTP Server
Router/client

ZTP Server (Bootstrap Server)
Restconf Server
Web Server (Artifacts)

Router/Device validation:
SUDI cert validation + challenge response

ZTP/Web Server
Router/client

Bootstrapping Data
Signed Scripts/Configs
Ownership Voucher (O.V.)

Cisco Live!
#CiscoLive
© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public
SZTP Artifacts (RFC 8572): ZTP Network/Server + Artifact Validation

- Device needs Bootstrapping Data to validate Server and Artifacts
- Order of validation: CIA signature → owner cert → O.V.
- O.V. is signed by Cisco, so ultimate trust established by manufacturer (Cisco)

CIA contains scripts/configs/redirect-URLs

Conveyed Information artifact (CIA)

CIA signature based on owner cert

Owner cert validated using O.V.

Ownership Voucher (RFC 8366, signed by Cisco)

What about the SZTP server?
Introducing Cisco Crosswork SZTP server (Release 4.0)

DHCP Server

Initial DHCPv4/v6 Messages – SZTP options 143/136

RestConf Server

Config/Script/Image Server

Crosswork SZTP APIs

SZTP

Crosswork Zero-Touch Provisioning (Classic)

ZTP Job scheduler
(home-grown, cluster schedulers like Kubernetes, marathon etc.)

RestConf Server

Config/Script/Image Server

Initial DHCPv4/v6 Messages – SZTP options 143/136

ZTP YANG model interaction

#CiscoLive BRKSPG-2024 © 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public 35
Application Hosting: Making life easy (and Secure) on Fixed and Modular platforms
What is a “non-XR” application?

Linux Applications that serve network and operational roles on IOS-XR platforms and are **NOT** part of the IOS-XR codebase

These applications largely come from the following sources:

**Custom Applications**  
(Developed and supported by Network Operators)  
E.g. SWAN, Custom DDOS apps, Customized Open/R, Custom automation scripts (python/bash/binaries)

**Open-Source Applications**  
(Developed and supported by the OSS community)  
E.g. xr-auditor, iperf, hping, netnorad, Open/R (open-source version), ISC-DHCP client/server/relay

**Cisco and Partner Applications**  
(Developed and supported by official Cisco Partners)  
E.g. Netrounds, Radware, thousandeyes
Using the owner cert to onboard Application RPM keys

- Starting with XR7 release 7.5.1, only signed applications can be onboarded on to XR platforms.

- The basic workflow is shown alongside:
  - **Owner(Customer) certificate** is onboarded into the hardware TAm of the RP (or both RPs for an HA platform).
  - A Key Package signed using the owner certificate is ratified by XR against the owner cert in TAm and is used to validate application signatures.
  - **The Application Package** to be onboarded is signed using the key in the Key package and is installed using IOS-XR install CLI/APIs on a running system, or SZTP (ztp script calling install) or at boot in a GISO (Install invoked during boot).
  - The Applications inside the application package can be scripts/compiled-binaries or Docker (container-based) applications.
Non-XR on-box Application Onboarding Scenarios

Running-System

XR CLI/APIs over SSH, netconf, gRPC can be used to onboard all artifacts:
1) Ownership Voucher (OV)
2) Owner Cert (burn into TAm)
3) RPM GPG key package
4) Application Package RPM

Secure ZTP (RFC 8572)

a) Using SZTP server

- SZTP onboards the owner cert (with OV)
- SZTP script onboards the Application and Key Package

b) Using USB with SZTP (RFC 8572) compliant file format

Golden ISO (GISO)

- Owner-Cert+ OV must be onboarded outside the GISO flow (using SZTP or CLI/API)
- GISO only packages the Application Package RPM and Key Package

GISO tool

Golden ISO

Base ISO

XR install

ZTP

USB

iPXE

TAm
Introducing XR AppMgr (Release 7.5.1): Consistent Application Management on Fixed and Modular Platforms

- Manages Application packages automatically across Dual-RP systems
- Enables Activation of App in XR configuration
- Support Automatic respawning of an activated App across RP failovers for dual-RP systems and across reloads/power-cycle etc.
- Support Monitoring capabilities for the application (Docker container, systemd Service), the apphosting Infra and the XR AppMgr itself.
- Support adjustment of Apphosting Infrastructure Constraints (Docker Daemon Settings, cgroups settings etc.)
- Support individual Application actions (Start/Stop/Kill/Install/Remove/Update)
- Provide Appropriate CLI and YANG APIs for each capability.
Application Lifecycle (7.5.1+)

Build
- Generate Owner Cert
- Build the application package RPM
- Sign using GPG key
- Create Key Package signed using Owner cert

Extend Trust in hardware (Establish ownership)
- Install Owner Cert into TAm
- Possible ways:
  - SZTP
  - XR API/CLI

Onboard the Application package and keys
- Ratify and Accept Key Package using XR7 Security API
- Ratify and Accept Application package XR7 Install CLI/API

Register the application and activate (Config)
- Register App
- Activate App
- XR AppMgr starts managing application lifecycle

Monitor, debug remediate
Through XR AppMgr:
- Monitor the Status of the Application
- Perform Application operations (Start, Stop, kill, Remove, Install/Reinstall)
Programmability: APIs at every layer of the Network Stack!
API-Driven, Layered SW Architecture

Management/Presentation Layer
- Provides access to configure and manage the stack through Network config/oper DB: Yang Models, CLI.

Application/Protocol Layer
- Provides APIs into the Routing Protocols (BGP, IGP, SR, etc.)

Network Infrastructure Layer / Service Adaptation Layer
- Acts as the bridge between the Application Layer and the HW
- Presents abstractions to the Application/Protocol Layer

System OS
- Linux Kernel
- BSP (Board Support Package) - Boot Loader, Device Drivers, etc.

ASIC SDK and drivers for the SDK

Hardware
- Consists of ASIC/Chipset from HW vendors + CPU, Fans, Sensors

3rd Party Agent + Telemetry

OSS

Management
- CLI, Netconf, SNMP, Syslog, SSH

Applications / Protocol Stack
- BGP, ISIS, OSPF, LDP, SR, L2 Protocols

Network Infrastructure / Service Adaptation
- RIB, Label Manager, BFD, Interface and more

System OS + BSP

ASIC SDK

CPU

NPU ASIC

Fans, Sensors, Optics, etc.
Model-Driven Yang-Based Manageability APIs
# Model-Driven Manageability

<table>
<thead>
<tr>
<th>Controller Orchestrator</th>
<th>Apps</th>
<th>Model-Driven SDKs YANG Development Kit (YDK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed-loop automation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model-Driven Telemetry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model-Driven Configuration</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protocol</th>
<th>NETCONF</th>
<th>gRPC</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Encoding</th>
<th>XML</th>
<th>JSON</th>
<th>GPB</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Transport</th>
<th>SSH</th>
<th>TCP</th>
<th>HTTP</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Models</th>
<th>YANG Models (native, open)</th>
</tr>
</thead>
</table>

- Apps
- SDK
- Model-Driven Manageability

- Protocol: NETCONF, gRPC
- Encoding: XML, JSON, GPB
- Transport: SSH, TCP, HTTP
- Models: YANG Models (native, open)
OpenConfig Model Support

SYSDM

Data

Config / oper models

YANG

Model-Driven Manageability

Management Protocol

gRPC Network Management Interface (gNMI)

protobuf

Operational Commands

gRPC Network Operations Interface (gNOI)

protobuf

Route (RIB) and Label (LSD) Manipulation

gRPC Routing Information Base Interface (gRIBI)

protobuf

Route (RIB) and Label (LSD) Manipulation

"IOS-XR support slated for release 7.5.1"
Yang-Based Streaming Telemetry
How Do You See Telemetry?

Find tons of Content on Streaming Telemetry with IOS-XR on:
https://xrdocs.io/telemetry/
“Pushing” More Data Really Does Work Better

- More counter data
- Reduction in CPU load
- Faster collection

Time to collect all data (chassis, 576x100GE)

- Interface counters
- Memory

Graphs showing:
- Counters over time (5s, 10s, 15s, 20s)
- CPU load with SNMP and Telemetry destinations
- Time to collect data with comparison between Telemetry and SNMP
Model-Driven Control-Plane APIs based on gRPC Service-Layer (SL) APIs
Service Layer API Architecture

Service Layer APIs bypass SysDB giving higher performance

CLI, Yang Models, Streaming Telemetry

Network Infrastructure Layer (Service Layer)

Manageability Layer

- gRPC  Server (Router)
  Protobuf Schema/Model
  - Initialization RPCs
  - RIB RPCs
  - MPLS RPCs
  - Interface RPCs
  - BFD RPCs

- SL-API Functionality Verticals

- On-Box Client
  - gRPC stub
  - Python, C++, go

- Off-Box Client
  - gRPC stub
  - Python, C++, go

Request

Response
Service Layer API Example Use Cases

Traffic Engineering and Path Selection

- Engineering paths for applications through Route/label manipulation, all based on user specific logic

Programmable Route Downloads

- Programmable route downloads to CDN PoP routers to optimize TCAM space

Bring your own Protocol/Agent

- On-box agents and custom protocols that co-exist with standard protocols to influence routing

More info on SL API: [https://xrdocs.io/cisco-service-layer/](https://xrdocs.io/cisco-service-layer/)
Security/Trust:
Trust tied to HW → Secure Boot + Runtime Security!
## Trusted Network – Strategic Roadmap

| Establish Trust in Hardware | • Enhanced Hardware Integrity Verification  |
|                            | • Hardware Crypto and Identity (SUDI)      |
|                            | Protects against:                         |
|                            | • Counterfeit Hardware                    |
|                            | • Hardware Tampering                      |
| Verify Trust in OS         | • Process Level Signature Verification    |
|                            | • Secure Storage for Secrets / Keys       |
|                            | Protects Against:                         |
|                            | • “Boot-kit” Attacks                      |
|                            | • Malware injection                       |
| Maintain Trust at Runtime  | • Runtime Protections: ASLR / W^X          |
|                            | • Control Plane Protection                |
|                            | Protects against:                         |
|                            | • Remote Exploits                         |
|                            | • Denial of Service                       |
| Visualize Trust            | • Boot Integrity Verification             |
|                            | • Process Integrity Measurement           |
|                            | Enables:                                 |
|                            | Detection of compromise and Trust Posture Report |
Establishing Trust with Secure Boot

Server OS Starts at UEFI BIOS

- Power On
- Bootloader
- OS Kernel

Cisco Root-of-Trust begins in hardware

- Power On
- Hardware Anchor
- CPU Micro loader
- Bootloader
- OS Kernel
Secure boot process: Diving Deeper

**BIOS launch and verification**

1. Cisco public keys in TAm (PK, KEK, DB) are used to verify signatures during the initial boot process. At powerup, a microloader in the TAm first verifies the digital signature of the BIOS using the LDWM key in TAm.

2. BIOS then executes verification of the hardware against Known Good Values (KGVs) of the hardware inside the database in TAm. These known good values are programmed by manufacturing. In the case there is a failure, then the failure is logged.

**Bootloader launch and verification**

Next, the BIOS verifies the digital signature of bootloader using the <platform-family> key in TAm DB.

**Kernel, initrd, grub-config verification**

1. Bootloader is launched by BIOS. Bootloader then takes help of BIOS to verify kernel, initrd, and grub-config.

2. Each verification operation is logged. Initrd is then expanded to create the root file system.

**Kernel modules verification**

1. Kernel is launched and the required keys (PK, KEK, IMA, RPM) are loaded into the kernel keyrings.

2. Kernel then verifies the kernel modules, and the results are logged.

**XR process launch**

Finally, XR processes are launched and each process is subject to the IMA policy checks to verify signatures on their hashes before launch.

**XR RPM installation**

1. IOS XR install process installs IOS XR RPMs that are part of the image.

2. The IOS XR install process uses the RPM key loaded from TAm to verify the signatures on all RPMs before installing them.

**OS process boot**

IMA, which is used to validate signatures at runtime, is launched with appropriate IMA policy to validate the init process.
Pushing the envelope with Network CI/CD workflows
Enabling Network CI/CD with IOS-XR7

Base Artifact Sources
- CCO
- XR RPMs
- Images (Golden ISOs, Base ISOs)

DevHub

Continuous Integration Infrastructure
- Jenkins
- KitchenCI

Applications
- Source Code
- Containers
- gitlab
- github
- LXC

Test Automation
- CQE

Test Configurations
- gitlab
- github

Virtual Hardware
- KVM
- docker
- Vagrant
- VIRL

Canary Hardware Lab

Lab Orchestration
- Topology
- Applications

Production Repository
- gitlab
- github

Production Orchestration
- Topology
- Applications

Configuration And Report Backup
- Rancid
- Oxidize

Secure ZTP
- DHCP Server
- ZTP Server

TEST

PRODUCTION

Canary Hardware Lab

TEST PRODUCTION

Hardware Lab

Canary

#CiscoLive

BRKSPG-2024
© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

58
IOSXR7: Cloud-Ready, by design.
Thank you
Continue your education

- Demos in the Cisco campus
- Meet the engineer 1:1 meetings
- Walk-in labs
- Related sessions