



Virtual Routing in the Cloud

ciscopress.com

**Arvind Durai
Stephen Lynn
Amit Srivastava**

FREE SAMPLE CHAPTER

SHARE WITH OTHERS



Virtual Routing in the Cloud

Arvind Durai, CCIE No. 7016

Stephen Lynn, CCIE No. 5507 & CCDE No. 20130056

Amit Srivastava

Cisco Press

800 East 96th Street

Indianapolis, IN 46240 USA

Virtual Routing in the Cloud

Arvind Durai, CCIE No. 7016
Stephen Lynn, CCIE No. 5507 & CCDE No. 20130056
Amit Srivastava

Copyright© 2016 Cisco Systems, Inc.

Published by:
Cisco Press
800 East 96th Street
Indianapolis, IN 46240 USA

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the publisher, except for the inclusion of brief quotations in a review.

Printed in the United States of America

First Printing April 2016

Library of Congress Control Number: 2016934921

ISBN-13: 978-1-58714-494-3

ISBN-10: 1-58714-494-8

Warning and Disclaimer

This book is designed to provide information about CSR 1000V router and adoption of NFV technology in the cloud environment. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an “as is” basis. The authors, Cisco Press, and Cisco Systems, Inc. shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the discs or programs that may accompany it.

The opinions expressed in this book belong to the author and are not necessarily those of Cisco Systems, Inc.

Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Cisco Press or Cisco Systems, Inc., cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Readers' feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through email at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Publisher	Paul Boger
Associate Publisher	Dave Dusthimer
Business Operation Manager, Cisco Press	Jan Cornelssen
Executive Editor	Brett Bartow
Managing Editor	Sandra Schroeder
Development Editor	Ellie Bru
Senior Project Editor	Tonya Simpson
Copy Editor	Kitty Wilson
Technical Editor(s)	Matt Bollick, Ray Wong
Editorial Assistant	Vanessa Evans
Cover Designer	Mark Shirar
Composition	Mary Sudul
Indexer	Brad Herriman
Proofreader	The Wordsmithery LLC



Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV
Amsterdam, The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.



CCDE, CCENT, Cisco Eos, Cisco HealthPresence, the Cisco logo, Cisco Lumin, Cisco Nexus, Cisco StadiumVision, Cisco TelePresence, Cisco WebEx, DCE, and Welcome to the Human Network are trademarks. Changing the Way We Work, Live, Play, and Learn and Cisco Store are service marks; and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCRP, CCNA, CCNP, CCSP, CQVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, iQuick Study, IronPort, the IronPort logo, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0812R)

About the Authors

Arvind Durai, CCIE No. 7016, is an advanced services principal architect for Cisco Systems. His primary responsibility in the past 17 years has been in supporting major Cisco customers in the enterprise sector, including financial, retail, manufacturing, e-commerce, state government, utility (smart grid networks), and health-care sectors. Some of his focuses have been on security, multicast, network virtualization, and data center, and he has authored several white papers and design guides on various technologies. He has also been involved in data center design for more than 10 years and has designed many enterprise private cloud data center environments.

Arvind maintains two CCIE certifications: Routing and Switching, and Security. He holds a Bachelor of Science degree in Electronics and Communication, a Master's degree in Electrical Engineering (MS), and a Master's degree in Business Administration (MBA). He is a coauthor of two Cisco press books, *Cisco Secure Firewall Services Module* and *TcL Scripting for Cisco IOS*.

He has coauthored IEEE WAN smart grid architecture and has been a panel member for IEEE publications. Arvind also has presented in many industry forums, such as IEEE and Cisco LIVE.

Stephen Lynn, CCIE No. 5507 (Routing & Switching/WAN/Security) and CCDE No. 20130056, is an architect at Cisco Systems in the U.S. federal area. He has been with Cisco for more than 16 years and is a subject matter expert on enterprise network architecture. His focus is on large-scale network designs, including campus, WAN, and data center. As a recognized expert within Cisco and in the industry, Stephen has been working on large-scale, complex wide-area network designs in an enterprise environment. Stephen's focus has been on architectural designs involving 1,000 nodes to more than 10,000 nodes, leveraging technologies such as DMVPN, GET VPN, and FlexVPN to provide transport encryption and network segmentation over IP transport such as MPLS/Ethernet. Other areas of focus include high availability and convergence, QoS, Performance Routing (PfR), and network virtualization.

Stephen is a well-known speaker who has presented at several conferences and seminars worldwide. He holds a Bachelor of Science in Electrical Engineering from the University of Virginia. Stephen is based out of the Cisco office in Washington, DC.

Amit Srivastava is a senior manager with Equinix, Inc. At Equinix his team is responsible for global network and product fulfillment for Equinix's Cloud Exchange platform. Amit formerly worked as a technical leader with Cisco Systems, Inc. He has developed, tested, and enhanced network software for nearly 14 years. Before joining Cisco, he held positions in software application development, management, and testing.

Amit was involved in developing embedded applications for mobile devices in his engagement with Hughes Networks prior to joining Cisco.

Amit has been involved in the development cycles of new operating systems such as IOS XR and IOS XE and delivering features such as MPLS-based Layer 2 and 3 VPNs and traffic engineering. With IOS XE, Amit has worked with platforms such as ASR 1000 and CSR 1000V right from their inception, delivering enterprise-level features like IPsec, NAT, firewalls, NetFlow, AVC, and QoS. Amit holds a Bachelor of Science degree in Electrical Engineering.

About the Technical Reviewers

Ray Wong is a technical marketing engineer (TME) for Cisco Systems. In his more than eight years with Cisco, he has worked in multiple roles, from system testing, to solution design and validation, to technical marketing. He was a major contributor in the Cisco Virtual Office (CVO) solution. Together with his TME role for Cisco Cloud Services Router (CSR 1000V), he is also a subject matter expert for IOS VPN, including DMVPN, GET VPN, and FlexVPN.

Ray holds a Bachelor of Science degree in Computer Science and Mathematics from the University of Wisconsin–Madison. He is also a frequent speaker at Cisco Live events.

Matt Bollick has worked in technical marketing at Cisco for the past 19 years, running an obstacle course of technologies, including SNA, ATM and Ethernet switching, service provider aggregation, metro Ethernet, network management, and enterprise branch architectures. He has also worked on a variety of products, including the Cisco 7500, 7200, LS1010, 8540, 7300, and Cisco 10K before finally settling down for the past several years as the platform architect for the ISR series of branch routers. In his spare time, Matt is an avid SCUBA diver in North Carolina.

Dedications

From Arvind:

I am thankful to God for everything. I would like to dedicate this book to my wife, Monica, and my son, Akhhill, who have been extremely patient and supportive during my long working hours. I am grateful to my parents for their blessings and for providing me with strong values.

I would also like to thank my parents-in-law, brother and family, and brother-in-law and family for all their support and wishes.

From Stephen:

I would like to dedicate this book to my wonderful and beautiful wife, Angela, and to my two incredible children, Christina and Ethan. Without your love, sacrifice, and support, this book would not have been possible. Thanks for putting up with the late nights and weekends I had to spend behind the computer and on conference calls instead of playing games, building Legos, and doing other fun family activities.

From Amit:

I would like to dedicate this book to my wife, Reshma, my daughter, Aarushi, and my parents. Without their love and support, I would never have been able to work on this. I would also like to thank my parents-in-law and my entire extended family. Their love and support have always been unconditional.

Acknowledgments

Arvind Durai:

Thanks to my wife, Monica, for encouraging me to write my third book. She inspired me and helped keep my spirits up all the time and provided her thoughts in multiple sections of this book. Thank you!!!

It was great working with Amit and Stephen. Their excellent technical knowledge and passion for writing made this writing experience a pleasure. I am looking forward to more years of working together as colleagues and friends.

Stephen Lynn:

A debt of gratitude goes to my coauthors, Arvind and Amit. Your knowledge and dedication to this project are appreciated more than you will ever know.

Acknowledgements for this book wouldn't be complete without mentioning my wife, Angela, who has endured and supported me through all my endeavors.

Amit Srivastava:

Special thanks to Arvind and Stephen, from whom I learned a lot while writing this book. I look forward to their continued support.

Our Acknowledgement

Many people within Cisco have provided feedback and suggestions to make this a great book. Thanks to all who have helped in the process, especially Ray Blair and Matt Falkner, for providing insightful input during the proposal process. A special thank you goes to our technical editors, Ray Wong and Matt Bollick, for your technical accuracy and insight into the technologies. Special thanks to Dimitris Vlassopoulos for providing his NSO lab setup and sharing his insights!

A big thank you goes out to the production team for this book, Brett Bartow, Ellie Bru, and Tonya Simpson, who have been incredibly professional and a pleasure to work with, and for making this book possible.

Contents at a Glance

	Introduction	xv
Chapter 1	Introduction to Cloud	1
Chapter 2	Software Evolution of the CSR 1000	37
Chapter 3	Hypervisor Considerations for the CSR	59
Chapter 4	CSR 1000V Software Architecture	95
Chapter 5	CSR 1000V Deployment Scenarios	141
Chapter 6	CSR Cloud Deployment Scenarios	185
Chapter 7	CSR in the SDN Framework	223
Chapter 8	CSR 1000V Automation, Orchestration, and Troubleshooting	247
Appendix A	Sample Answer File for Packstack	293
	Index	319

Reader Services

Register your copy at www.ciscopress.com/title/9781587144943 for convenient access to downloads, updates, and corrections as they become available. To start the registration process, go to www.ciscopress.com/register and log in or create an account*. Enter the product ISBN 9781587144943 and click Submit. Once the process is complete, you will find any available bonus content under Registered Products.

*Be sure to check the box that you would like to hear from us to receive exclusive discounts on future editions of this product.

Contents

	Introduction	xv
Chapter 1	Introduction to Cloud	1
	Evolution of the Data Center	1
	Data Center Architecture Building Blocks	2
	Introduction to Virtualization in the Data Center	4
	Evolution of Virtualization	5
	Conceptual Architecture of Virtualization	5
	Types of Virtualization Technologies	6
	<i>Server Virtualization</i>	6
	<i>Types of Server Virtualization</i>	8
	<i>Storage Virtualization</i>	9
	<i>Types of Storage Virtualization</i>	11
	<i>Network Virtualization</i>	12
	<i>Network Virtualization Evolution</i>	13
	<i>Types of Network Virtualization</i>	14
	<i>Service Virtualization</i>	15
	Introduction to the Multitenant Data Center	16
	Introduction to Cloud Services	18
	Infrastructure as a Service (IaaS)	18
	Platform as a Service (PaaS)	19
	Software as a Service (SaaS)	20
	Cloud Deployment Models	20
	Cloud Design Considerations	21
	<i>Domain 1: Infrastructure and Environmental</i>	22
	<i>Domain 2: Abstraction and Virtualization</i>	23
	<i>Domain 3: Automation and Orchestration</i>	23
	<i>Domain 4: Customer Interface</i>	24
	<i>Domains 5 and 6: Service Catalog and Financials</i>	24
	<i>Domains 7 and 8: Platform and Application</i>	24
	<i>Domain 9: Security and Compliance</i>	24
	<i>Domain 10: Organization, Governance, and Process</i>	25
	Enterprise Connectivity to the Cloud	26
	<i>Internet for Transport</i>	26
	<i>Direct Connectivity to a Cloud Provider</i>	28
	Enterprise Cloud Adoption Challenges	29

- Software-Defined Networking 30
 - Open Networking Foundation 31
 - OpenDaylight Project 32
 - Network Function Virtualization 33
 - OpenStack 34
- Summary 35

Chapter 2 Software Evolution of the CSR 1000 37

- IOS Software Architecture 37
- IOS XE Architecture 39
 - The IOS XE Kernel 40
 - The IOS Daemon 40
 - The Forwarding Manager 41
 - The Interface Manager 41
 - The Platform Manager 41
- Cisco ASR 1000 System Architecture Overview 41
 - Route Processor 42
 - Embedded Service Processor 42
 - SPA Interface Processor 43
- Cloud Service Router 1000V Overview 44
- Deployment Requirements 45
- Elastic Performance and Scaling 47
- Rapid Deployment and Routing Flexibility in the Cloud 49
- CSR 1000V Deployment Examples 50
 - Secure Cloud VPN Gateway 50
 - Network Extension from Premises to Cloud 51
 - Segmentation Within a Cloud 52
- CSR 1000V Key Features 52
- Summary 57

Chapter 3 Hypervisor Considerations for the CSR 59

- Understanding Operating Systems 59
 - Operating System Design 60
 - Physical Resource Management* 60
 - Software Access to Physical Resources* 62
 - Kernels 63
 - Microkernels* 63

	<i>Hybrid Kernels</i>	64
	<i>The Cisco IOS Kernel</i>	64
	The Boot Process	66
	Linux Memory Management	69
	<i>Linux Swap Space and Memory Overcommit</i>	69
	<i>Linux Caching</i>	71
Understanding Hypervisors		71
	How Does a Hypervisor Compare to an Operating System?	72
	Type 1 Hypervisor Design	74
	<i>Monolithic Architecture</i>	74
	<i>Microkernel Architecture</i>	74
	<i>Core Partitioning</i>	75
ESXi Hypervisor		75
	Architectural Components of ESXi	75
	The VMkernel	75
	<i>Components of the VMkernel</i>	76
	Processes Running on the VMkernel	77
	Device Drivers	78
	File Systems	79
	Management	80
KVM		82
	Architectural Components of KVM/QEMU	84
	Guest Emulator (QEMU)	85
	Management Daemon (Libvirt)	88
	<i>User Tools (virsh, virt-manager)</i>	89
Hyper-V		91
Xen		92
Summary		94
Chapter 4	CSR 1000V Software Architecture	95
	System Design	95
	Virtualizing the ASR 1001 into the CSR 1000V	98
	CSR 1000V Initialization Process	99
	CSR 1000V Data Plane Architecture	100
	CSR 1000V Software Crypto Engine	103
	Life of a Packet on a CSR 1000V: The Data Plane	103
	Netmap I/O	104

- Packet Flow 106
- Device Initialization Flow* 106
- TX Flow* 107
- RX Flow* 108
- Unicast Traffic Packet Flow* 109

- Installing the CSR 1000V on a VMware Hypervisor 110
 - Bringing Up the VM with the CSR 1000V on ESXi 110
- Installing the CSR 1000V on a KVM Hypervisor 126
 - Bring Up the CSR 1000V as a Guest 126
- Performance Tuning of the CSR 1000V 137
- Summary 139

Chapter 5 CSR 1000V Deployment Scenarios 141

- VPN Services 141
 - Layer 2 VPNs 141
 - Layer 3 VPNs 142
 - Site-to-Site VPNs* 143
 - Remote Access VPNs* 147
- Use Cases for the CSR 1000V as a VPN Service Gateway 148
 - Enterprise Data Center Network Extension 148
 - The CSR 1000V as a VPN Gateway* 148
 - CSR for Secure Inter-Cloud Connectivity 152
 - Remote VPN Access into the Cloud 153
- BGP Route Reflector Use Case for the CSR 155
 - The CSR 1000V in a Hierarchical Route Reflector Use Case 157
- Planning for Future Branch Design with the CSR 1000V 162
 - Evolution of Branch Virtualization 164
- LISP and CSR 168
 - LISP Terminology 169
 - The LISP Data Plane 171
 - The LISP Control Plane 171
 - Typical LISP Use Cases 175
 - IP Mobility* 175
 - IPv6 Migration* 175
 - Network-to-Network Connectivity* 175
 - Network-to-Network Interconnection Topology and Configuration* 176
- Summary 183

Chapter 6	CSR Cloud Deployment Scenarios	185
	CSR in a Multitenant Data Center	185
	Cloudburst	190
	Direct Access Model	191
	Redirection Access Model	192
	The Cisco Inter-Cloud Fabric	194
	Private Cloud Deployment with CSR in OpenStack	195
	Introduction to OpenStack	196
	<i>Primary Use Case for OpenStack</i>	196
	<i>OpenStack Components</i>	197
	CSR Within OpenStack	206
	<i>CSR 1000V as a Neutron Router</i>	206
	<i>CSR 1000V as a Tenant Router</i>	209
	CSR 1000V in a Public Cloud	211
	Amazon Web Services Deployment for the CSR	211
	<i>Amazon Web Service Solutions</i>	211
	<i>Routing in AWS Clouds</i>	212
	CSR 1000V Deployment in AWS	216
	<i>Instantiate a CSR in AWS</i>	217
	Summary	222
Chapter 7	CSR in the SDN Framework	223
	Deploying OpenStack	225
	CSR as an OpenStack Tenant Deployment	235
	Instantiate CSR Plugin to OpenStack	242
	Summary	245
Chapter 8	CSR 1000V Automation, Orchestration, and Troubleshooting	247
	Automation	248
	BDEO	248
	NSO (Tail-f)	249
	<i>NSO Example for NFV Orchestration with OpenStack (Service Chain)</i>	252
	Orchestration	267
	Virtual Managed Services (VMS)	267
	Cisco Prime Network Services Controller (PNSC)	269
	CSR 1000V Troubleshooting	271
	Architecture Overview	271

I/O Configuration	272
<i>vSwitch</i>	272
<i>PCI Passthrough</i>	274
<i>SR-IOV (Single Root I/O Virtualization)</i>	274
<i>Host Configurations</i>	275
Debugging Packet Loss	276
<i>High-Level Packet Flow</i>	276
<i>ESXi Packet Debugging</i>	289
Summary	292

Appendix A Sample Answer File for Packstack 293

Index 319

Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in the IOS Command Reference. The Command Reference describes these conventions as follows:

- **Boldface** indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that are manually input by the user (such as a **show** command).
- *Italic* indicates arguments for which you supply actual values.
- Vertical bars (|) separate alternative, mutually exclusive elements.
- Square brackets ([]) indicate an optional element.
- Braces ({ }) indicate a required choice.
- Braces within brackets ({ [] }) indicate a required choice within an optional element.

Introduction

In today's business environment, enterprise customers are under more pressure than ever to innovate and adapt to new challenges and market conditions. Enterprises want to focus their investments on their core business while reducing IT spending.

The cloud offers enterprise customers many benefits, such as lower costs and flexibility. The cloud's elastic model enables a company to increase and decrease infrastructure capacity on demand. The usage-based model offered by the cloud helps governments and enterprises reduce costs while increasing business agility by moving applications to the cloud and consuming infrastructure resources from the cloud. This leads to enterprises looking at consuming network and IT services from the cloud rather than investing in in-house operations.

The enabling technology in unlocking the cloud is virtualization. Virtualization abstracts and isolates the computing hardware and underlying infrastructure into a logical resource pool, allowing key capabilities such as resource sharing, virtual machine (VM) isolation, and load balancing. These capabilities provide the fundamental building blocks for an agile and scalable cloud environment with rapid provisioning, workload sharing, and increased availability.

The surge in applications and IT service consumption moving to the cloud highlights the need for evolved technologies and network elements in the cloud that offer security and visibility to help businesses with performance and compliance verification. Evolved networks and network services enable the provider to offer cloud services with security, performance, and availability. The Cisco Cloud Services Router 1000V (CSR 1000V) is a fully virtualized software router that offers a platform for enterprises to extend the data center to the cloud and to enforce their policies in the cloud.

The Cisco CSR 1000V provides a transparent solution for extending IT services into provider-hosted clouds. The solution offers a rich set of features, including VPN, firewall, Network Address Translation (NAT), application visibility, and WAN optimization. These functions allow enterprise and cloud providers to build highly secure, scalable, and extensible cloud networks. In addition, the Cisco CSR 1000V supports a rich set of application programming interfaces (API), providing robust integration into software-defined networking (SDN) for automated provisioning of these networks and network services and allowing simplified management and orchestration, which help in driving down costs further.

Networks inherently carry vast amounts of information, including user locations, device capabilities, topologies, and end-to-end performance characteristics. When exposed appropriately through well-defined APIs, such information can be consumed by cloud applications to fine-tune and customize their efficient delivery. The future holds the promise of increasingly rich application–network interactions.

The primary objective of this book is to simplify design aspects and architectural details in a unified resource, augmenting Cisco's existing collection of installation and configuration guides for various cloud-related products and solutions. This book covers the key

virtualization technologies used in the cloud; it provides a concise, accessible presentation of cloud network services and the different types of operational environments in the cloud. Cloud networking service and delivery concepts are reinforced with illustrative examples; architecture of SDN orchestration and its connection to Cisco CSR 1000V network services are introduced and elaborated upon. In addition, the book reviews the building blocks of the CSR 1000V, covering its architecture and software design.

This book also explains network design and deployment scenarios for the Cisco CSR 1000V, which influence its pivotal role in the cloud environment. Furthermore, the book distills how intelligent networks help providers simplify cloud service management and reduce costs through efficient scaling and optimized capacity utilization. This book provides architectural knowledge that contextualizes the roles and capabilities of these advanced networks and network services, along with discussions of design factors essential for their insertion into cloud services:

- The book introduces the readers to the cloud and provides an overview to different types of cloud operational environments, including a prelude to the evolution of virtual routers.
- Virtualization is introduced as a pivotal technology in cloud adoption.
- The book covers the details of the operating systems and hypervisors on which virtual routers run. It provides details pertaining to the operational aspects of virtual routing.
- The reader is introduced to the architecture and software design of the Cisco CSR 1000V virtual router. The reader is subsequently introduced to a comprehensive set of APIs that can be leveraged by SDN.
- The book focuses on different designs and use cases and configuration examples for routing, secure extension of enterprises to the cloud, and VM mobility. It illustrates how the CSR 1000V addresses the challenges that an architect faces in migrating toward the cloud.
- This book covers the different management techniques available to simplify operational and monitoring aspects of cloud services.

Who Should Read This Book?

This book is targeted for a technical audience responsible for architecture, design, and deployment of data center and enterprise cloud services.

This book also caters to the next generation of cloud network operators to implement enterprise features in the cloud, leveraging the CSR 1000V.

After reading this book, you will have a better understanding of the following:

- Key virtualization concepts and cloud models
- CSR 1000V software architecture and design

- SDN and the CSR 1000V platform and API
- Simplification of data center multitenant design with the CSR 1000V
- Use cases for the CSR 1000V to simplify enterprise routing in the cloud
- Operational visibility, management, and control of an enterprise network in the cloud

How This Book Is Organized

This book is organized into the following chapters.

Chapter 1: Introduction to Cloud

This chapter introduces the concept of cloud computing. It describes the various cloud models available and how virtualization enables the present-day transition to the cloud. Multitenant data center designs are illustrated, and the concept of SDN is introduced here.

Chapter 2: Software Evolution of the CSR 1000

This chapter introduces the software evolution of the Cisco Cloud Services Router (CSR 1000V). It covers the infrastructure requirements and design considerations of a CSR 1000V, and it discusses the features that a CSR 1000V brings to the virtual routing realm.

Chapter 3: Hypervisor Considerations for the CSR

This chapter describes the different hypervisor technologies available on servers to manage the hardware resources for virtual machines. Hypervisor technology selection is an important consideration when deploying the CSR 1000V.

Chapter 4: CSR 1000V Software Architecture

This chapter describes the software design of the CSR 1000V. It details the control-plane and data-plane design of the CSR 1000V. It also describes licensing requirements, software implementation, and packet flow related to the CSR 1000V.

Chapter 5: CSR 1000V Deployment Scenarios

This chapter describes the common deployment scenarios for the CSR 1000V. It depicts these scenarios using configuration examples.

Chapter 6: CSR Cloud Deployment Scenarios

This chapter describes CSR 1000V deployments in the cloud and data center environments.

Chapter 7: CSR in the SDN Framework

This chapter describes SDN components. It also provides an overview of the CSR 1000V in the OpenStack framework. Case studies in this chapter aim to educate the reader on using the APIs for user-defined outcomes.

Chapter 8: CSR 1000V Automation, Orchestration, and Troubleshooting

This chapter provides an overview of CSR 1000V management tools for orchestration, monitoring, and troubleshooting. It also illustrates the operation workflow for deploying a CSR 1000V.

CSR 1000V Software Architecture

This chapter describes the software design of the CSR 1000V and details the data plane design. It also illustrates the software implementation and packet flow within the CSR 1000V, as well as how to bring up the CSR 1000V.

System Design

CSR 1000V is a virtualized software router that runs the IOS XE operating system. IOS XE uses Linux as the kernel, whereas the IOS daemon (IOSd) runs as a Linux process providing the core set of IOS features and functionality. IOS XE provides a native Linux infrastructure for distributing the control plane forwarding state into an accelerated data path. The control and data planes in IOS XE are separated into different processes, and the infrastructure to communicate between these processes supports distribution and concurrent processing. In addition, IOS XE offers inherent multicore capabilities, allowing you to increase performance by scaling the number of processors. It also provides infrastructure services for hosting applications outside IOSd.

Originally, IOS XE was designed to run on a system with redundant hardware, which supports physical separation of the control and data plane units. This design is implemented in the ASR 1006 and ASR 1004 series routers. The original ASR 1000 family hardware architecture consisted of the following main elements:

- Chassis
- Route processor (RP)
- Embedded service processor (ESP)
- SPA interface processors (SIP)

The RP is the control plane, whereas the ESP is the data plane. In an ASR 1006 and ASR 1004, the RP and ESP processes have separate kernels and run on different sets of hardware. ASR 1000 was designed for high availability (HA). The ASR 1006 is a fully

hardware redundant version of the ASR, and its RP and ESP are physically backed up by a standby unit. IOSd runs on the RP (as do the majority of the XE processes), and the RP is backed up by another physical card with its own IOSd process. The ASR 1004 and fixed ASR 1000s (ASR 1001-X and ASR 1002-X) do not have physical redundancy of the RP and ESP.

In the hardware-based routing platform for IOS XE, the data plane processing runs outside the IOSd process in a separate data plane engine via custom ASIC: QuantumFlow Processor (QFP). This architecture creates an important framework for the software design. Because these cards each have independent processors, the system disperses many elements of software and runs them independently on the different processors.

Tip The ASR 1000 platform first introduced IOS XE. Multiple products run IOS XE, including the following:

ASR 1000 family:

- ASR 1001-X
- ASR 1002-X
- ASR 1004
- ASR 1006
- ASR 1006-X
- ASR 1009-X
- ASR 1013

ASR 900 family:

- ASR 903

ISR family:

- ISR 4321
- ISR 4331
- ISR 4351
- ISR 4431
- ISR 4451-X

IOS XE retains the look and feel of IOS. However, because IOS runs as a Linux process, it enables the platform-independent code to reside inside the IOSd process running on the Linux kernel. By moving the platform-dependent code (drivers) outside the IOSd process, it makes IOS XE a very efficient software delivery model. Different platforms write their drivers and leverage the existing feature-rich control plane code from IOSd.

Multiple platforms run IOS XE. However, when understanding CSR 1000V architecture in this chapter, ASR 1000 is used as a hardware example because it was the first platform to run IOS XE.

As the need for smaller form factor ASRs arose, a one rack unit (RU) ASR 1000 was conceptualized and developed: ASR 1001. The ASR 1001 is a 64-bit architecture in which all processes (RP, SIP, and ESP) are controlled by a single CPU. The SPA interface complex, forwarding engine complex, and IOS XE middleware all access the same Linux kernel. This is achieved by mapping the RP, ESP, and SIP domains into logical process groups. The RP's process domain includes IOSd, a chassis manager process and forwarding manager. The ESP process domain includes the chassis manager process, QFP client/driver process, and forwarding manager.

The architecture diagram in Figure 4-1 provides a high-level overview of the major components.

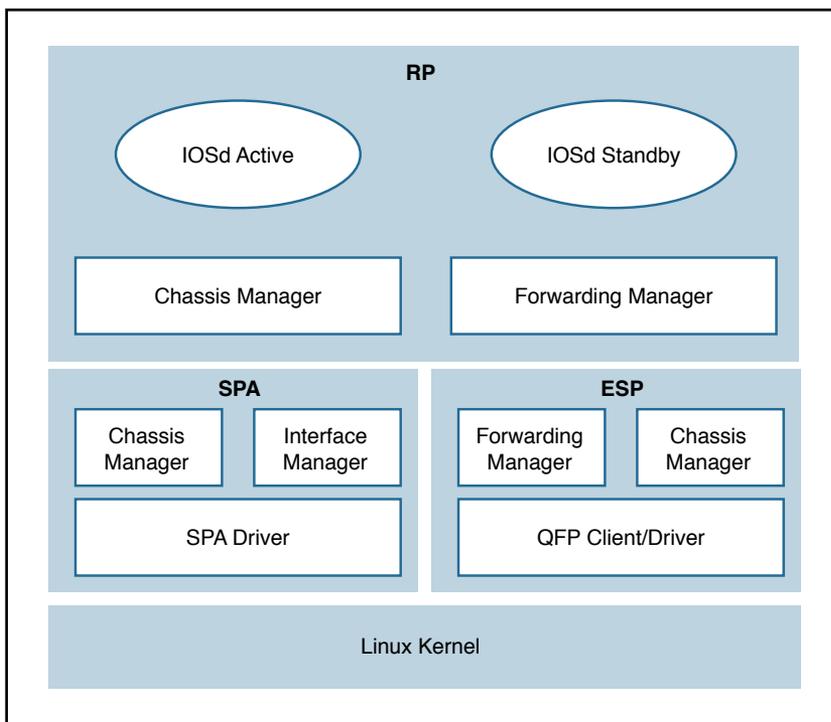


Figure 4-1 ASR 1001 Platform Logical Architecture

The details on grouping of the components are as follows:

- **RP**—RP mainly contains the IOS daemon (IOSd), the forwarding manager for RP (FMAN-RP), the chassis manager for RP (CMAN-RP), the kernel, and bootstrap utilities.

- **ESP (forwarding plane)**—ESP contains FMAN-FP and CMAN-FP, as well as QFP microcode and data plane drivers and crypto offload ASIC for handling hardware assist encryption.
- **SIP/SPA**—SIP/SPA houses the I/O interface for the chassis. It has its own CMAN and kernel process to handle the discovery, bootstrapping, and initialization of the physical interfaces.

Virtualizing the ASR 1001 into the CSR 1000V

There are a lot of commonalities between the system architectures of the CSR 1000V and the ASR 1001, and there are some differences as well. The CSR 1000V is essentially an ASR 1001 without the hardware. The following measures brought the ASR 1001 into the software-based design of the CSR 1000V:

- All the inter-unit communication with the SIP/CC was removed.
- The entire SIP/SPA interface complex was eliminated.
- The kernel utilities have been shared across the RP and ESP software complexes.
- The kernel utilities use the virtualized resources presented to it by the hypervisor.

The CSR is basically the ASR 1000 design stripped of its hardware components. When you compare the two designs, you find that the data path implementation is very different. This is because the ASR 1001 has a physical processor (the QFP) for running data path forwarding. In a CSR, the IOS XE data path is implemented as a Linux process.

The CSR 1000V is meant to leverage as much of the ASR 1001 architecture as possible. There are places in the CSR 1000V system where software emulation for hardware-specific requirements is needed. In general, the software architecture is kept the same, using the same grouping approach as for the hardware components. One of the major engineering efforts has been focused on migrating the QFP custom ASIC network processor capabilities onto general-purpose x86 CPU architectures and providing the distributed data path implementation for IOS XE. This effort creates a unique opportunity for Cisco to package this high-performance and feature-rich technology into the CSR 1000V. Figure 4-2 shows the high-level architecture of the CSR 1000V.

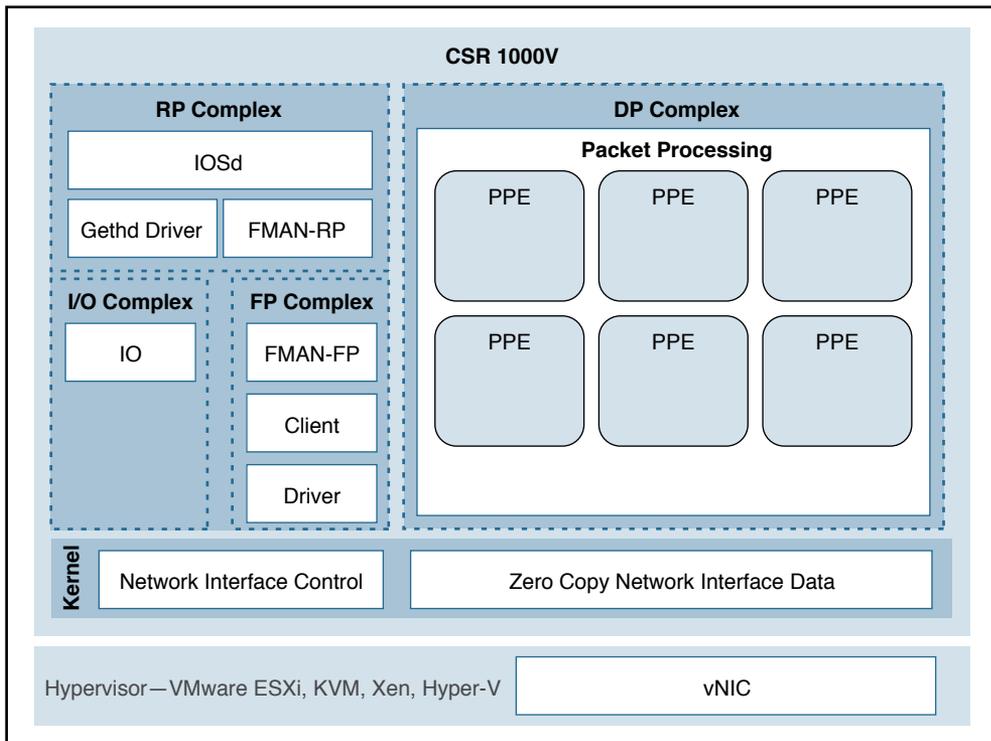


Figure 4-2 CSR 1000V High-Level Architecture

CSR 1000V Initialization Process

This section examines the initialization of a CSR 1000V running on a type 1 hypervisor. Refer to Chapter 2, “Software Evolution of the CSR 1000,” for details on the `IOSd` process running on the control plane.

When a CSR boots up as a virtual machine, interfaces are discovered by parsing the contents of `/proc/net/dev` on the Linux kernel. The `gethd` (Guest Ethernet Management Daemon) process performs the port enumeration at startup and then passes the interface inventory to the guest Ethernet driver within the `IOS` complex. The `IOSd` `gethd` driver then instantiates the Ethernet interfaces. This is how the I/O interfaces provided by the virtual NIC are managed by `IOS`.

The `gethd` process manages the interfaces on the CSR VM. It takes care of addition, removal, configuration, states, and statistics of the Ethernet interfaces on the CSR VM.

Figure 4-3 illustrates the CSR 1000V initialization sequence.

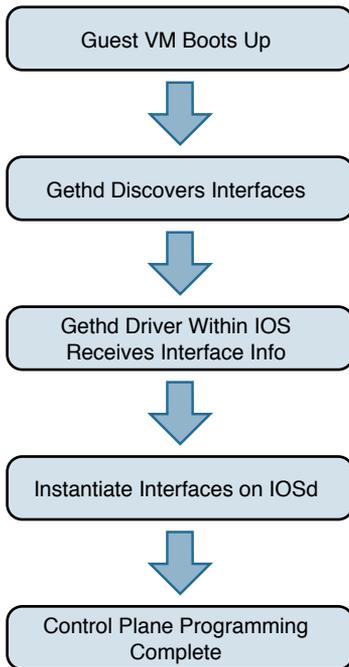


Figure 4-3 *CSR 1000V Initialization Sequence*

`gethd` is an important process that handles a variety of interface management functions, including interface removal/addition. It is an important part of the virtualized I/O used in CSR.

CSR 1000V Data Plane Architecture

Originally, IOS XE QFP data plane design consisted of four components: client, driver, QFP microcode (uCode), and crypto assist ASIC. Different ASR 1000 platforms package these components differently, but in general the four components are the same across platforms. CSR 1000V leverages the same client, driver, and uCode to support a multithread-capable packet processing data plane, with the exception of the crypto assist ASIC.

Figure 4-4 illustrates the CSR 1000V data plane architecture. The HW threads mentioned in the figure are packet processing engine (PPE) threads. The terms *HW* and *PPE* can be used interchangeably.

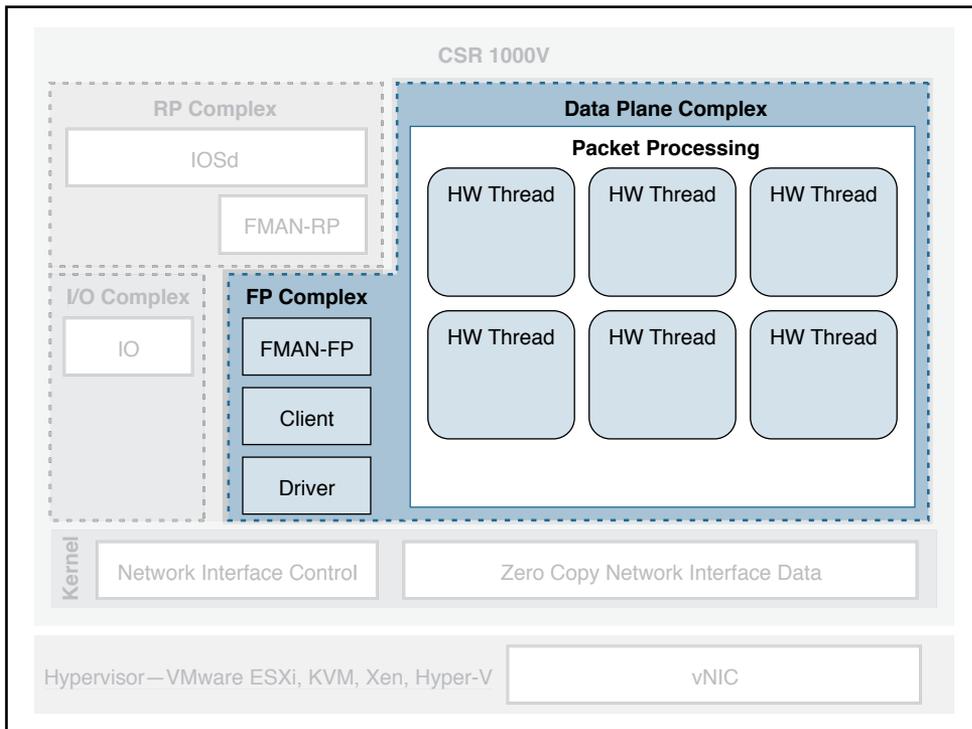


Figure 4-4 CSR 1000V Data Plane Architecture

The following is an overview of the three main components that make up the packet-processing data plane for CSR:

- Client**—The Client is software that ties together the control plane and the data plane. It is a collection of software modules that transform control plane information into various data plane forwarding databases and data structure updates. It is also responsible for updating the control plane with statistics from the data plane. It allocates and manages the resources of the uCode, including data structures in resource memory. The QFP Client is also responsible for restarting the QFP process in the event of failure. The Client provides a platform API layer that logically sits between IOSd and the uCode implementing the corresponding features. The Client API is called from FMAN-FP and then communicates with the uCode via both Interprocess Communicator (IPC) and shared memory interfaces provided by the Driver. Within the Client, feature processing support can be broken down into functional blocks known as Execution Agents (EA) and Resource Managers (RM). RMs are responsible for managing physical and logical objects, which are shared resources. An example of a physical object manager is the TCAM-RM, which manages allocation of TCAM resources, and an example of a logical object manager is the UIDB-RM, which manages the micro Interface Descriptor Block (uIDB) objects

used to represent various forms of interfaces. The data plane (uCode) uses uIDB objects to see the logical interfaces.

- **Driver**—The Driver is a software layer that enables software components to communicate with the hardware. It glues the software components to the hardware. The Driver is made up of libraries, processes, and infrastructure that are responsible for initialization, access, error detection, and error recovery. The Driver has hardware abstraction layering known as the Device Object (`devobj`) Model that allows it to support different QFP ASICs. Below the `devobj` API are implementations of various emulation and adaptation layers. In addition to the emulation and adaptation layers required to support the RMs listed in the Client section, the Driver is also responsible for coordinating memory access and IPC messaging between various QFP control plane software components and the QFP data plane packet processing uCode. The driver is completely segregated from the IOS code in an XE architecture, and this makes XE a very robust and flexible software architecture that offers complete separation of the control and data planes.
- **QFP uCode (packet processing)** —The uCode is where all the feature packet processing occurs. The uCode runs as a single process in the same VM/container as the Client and the Driver processes. IOSd initiates a packet process request through FMAN-FP. This request is then driven by the Client and the Driver interacting with the uCode to control the PPE behavior. The QFP uCode is broken up into four main components: Feature Code, Infrastructure, Platform Abstraction Layer (PAL), and Hardware Abstraction Layer (HAL). The PAL and HAL are essentially glue for the portability of software features to different hardware platforms. Originally, the PAL and HAL were designed for Cisco forwarding ASICs, such as QFP. In order for uCode software to run on top of x86 in a Linux environment, a new PAL layer is needed to support the specifics of the CSR 1000V platform. In addition, a new HAL is introduced for running QFP software on top of x86 in a Linux environment.

The intention is for the CSR 1000V data plane to leverage as much of the existing QFP code base as possible to produce a full-featured software data plane capable of leveraging the processing capacity and virtualization capabilities of modern multicore CPUs. One way to minimize changes to the existing QFP software code base is to emulate QFP hardware ASIC in such a way that the existing Client, Driver, and QFP uCode are not aware that they are running on a non-QFP platform. However, due to the complexity of QFP hardware and the differences in platform requirements, a pure emulation is impractical. There are some cases where we choose to emulate hardware because doing so is the straightforward approach for code leverage. In other cases, it is best to replace the corresponding functionality with an implementation that is compatible at an API level but may be a completely different algorithmic implementation.

CSR 1000V Software Crypto Engine

Cisco router platforms are designed to run IOS with hardware acceleration for crypto operations. Like other ASR 1000 platforms, the ASR 1001 includes a crypto acceleration engine on board to deliver crypto offload and to increase encryption performance. In this environment, the main processor performing the data path processing is offloaded from the computing-intensive crypto operations. Once the crypto offload engine completes the encrypt/decrypt operation, it generates an interrupt to indicate that the packet should be reinserted back into the forwarding path.

The CSR 1000V runs completely on general-purpose CPUs without an offload engine; therefore, the software implementation of the IPsec/crypto feature path is needed to support the encryption function. To that end, the CSR 1000V includes a software crypto engine that uses low-level cryptographic operations for encrypting and decrypting traffic. The software crypto engine is presented to the IOS as a slower crypto engine. One thing to note is the software crypto engine runs as an independent process within the CSR 1000V, and it therefore may run as a parallel process in a multicore environment. To improve the crypto performance of the CSR 1000V software router, the crypto data path is implemented to take advantage of the latest Advanced Encryption Standard (AES) crypto instruction set from Intel (AES-NI) for encryption/decryption operations.

The newer Intel processors, such as the Xeon Westmere-EP family and mobile Sandy Bridge family, provide instruction sets for enhancing Advanced Encryption Standard (AES-NI) cryptographic operations performance. These instructions are included in the CSR 1000V crypto library, along with other cryptographic and hash algorithms for low-level crypto operations. The crypto library is used by the software crypto engine as well as by other subsystems within IOS that require cryptographic operations. The inclusion of Intel's crypto instruction set allows the CSR 1000V to take advantage of the latest Intel CPUs for encryption and decryption operations in the data path.

Life of a Packet on a CSR 1000V: The Data Plane

Before we get into the details of packet flow for the CSR 1000V, it is important to understand the drivers that make it possible for the CSR VM to talk to physical devices and other software modules. These drivers act as software glue, relaying a packet to and from the physical wire. We have touched on the different hypervisors that enable the CSR VM to work on various x86 architectures. Here we discuss packet flow to and from a CSR VM.

Figure 4-5 shows the virtualization layers of a CSR 1000V VM.

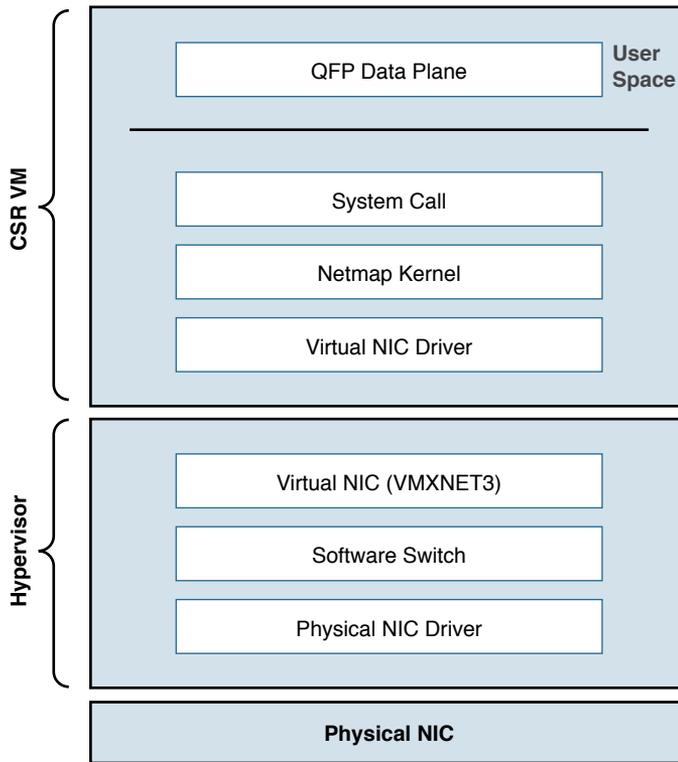


Figure 4-5 CSR VM Layers

From Figure 4-5, you can see that the hypervisor presents a virtual NIC to its guest VM by using a driver. This driver can either be a para-virtualized driver (for example, VMXNET3) or a real/emulated driver (for example, e1000). Para-virtualized drivers are native to hypervisors and perform much better than emulated drivers such as the e1000. Hypervisors support emulated drivers because they are required for full virtualization. Recall from Chapter 1, “Introduction to Cloud,” that in full virtualization, guest operating systems do not require any support from the hypervisor kernel and run as though on real hardware. Therefore, support for emulated drivers is required. However, the performance of emulated drivers is much lower than that of para-virtualized drivers. The CSR VM supports para-virtualized drivers only.

Netmap I/O

Netmap is an open-source I/O infrastructure package that enables the CSR VM to get rid of the multiple software layers in the traditional Linux networking stack I/O model. This results in faster I/O. Understanding the Netmap I/O model will help you better understand packet flow to and from a CSR VM. This section provides an overview of the Netmap I/O model and compares it with a Linux I/O model. It is important to understand the I/O model before drilling down to packet flow.

Netmap is designed to strip down software layers and get the frame from the wire to the data plane process in user space as quickly as possible. Netmap achieves this through the four building blocks of its I/O architecture:

- **Thin I/O stack**—Netmap bypasses the Linux networking stack to reduce overhead. Since the CSR data plane runs in the user space, when it wants an I/O architecture to deliver receive (Rx) frames from the NIC to the user space (data plane) and transmit (Tx) frames from the data plane to the NIC, it leverages Netmap’s thin I/O stack.
- **Zero copy**—Netmap maps all memory from rings (pool of memory buffers) in a way that makes them directly accessible in the data plane (user space). Hence there is no copy involved in getting the information to the user space. Preventing a copy operation saves a lot of time in an I/O model, and Netmap’s zero-copy model is very effective at increasing performance compared to a traditional Linux I/O model.
- **Simple synchronization**—The synchronization mechanism in Netmap is extremely simple. When you have the Rx packets on the ring, Netmap updates the count of new frames on the ring and wakes up threads that are sleeping to process the frames. On the Tx side, the write cursor is updated as a signal to announce the arrival of new frames on the Tx ring. Netmap then flushes the Tx ring.
- **Minimal ring manipulation**—In the Netmap I/O architecture, the ring is sized such that the producer accesses the ring from the head end, while the consumer accesses it from the tail. (*Producer* and *consumer* are terms associated with the process that tries to initiate the I/O process [producer] and a process that gets affected in trying to serve the producer [consumer].) The access to the ring is allowed simultaneously for the producer and the consumer. In a regular Linux I/O scenario, you would have to wait for the host to fill up the ring with pointers to buffers. When the ring is being serviced, Linux detaches the buffers from the ring and then replenishes the ring with new pointers.

An overview of the layers of software involved in building a CSR 1000V VM is illustrated previously in Figure 4-5. Figure 4-6 compares the Linux I/O model with the Netmap I/O model.

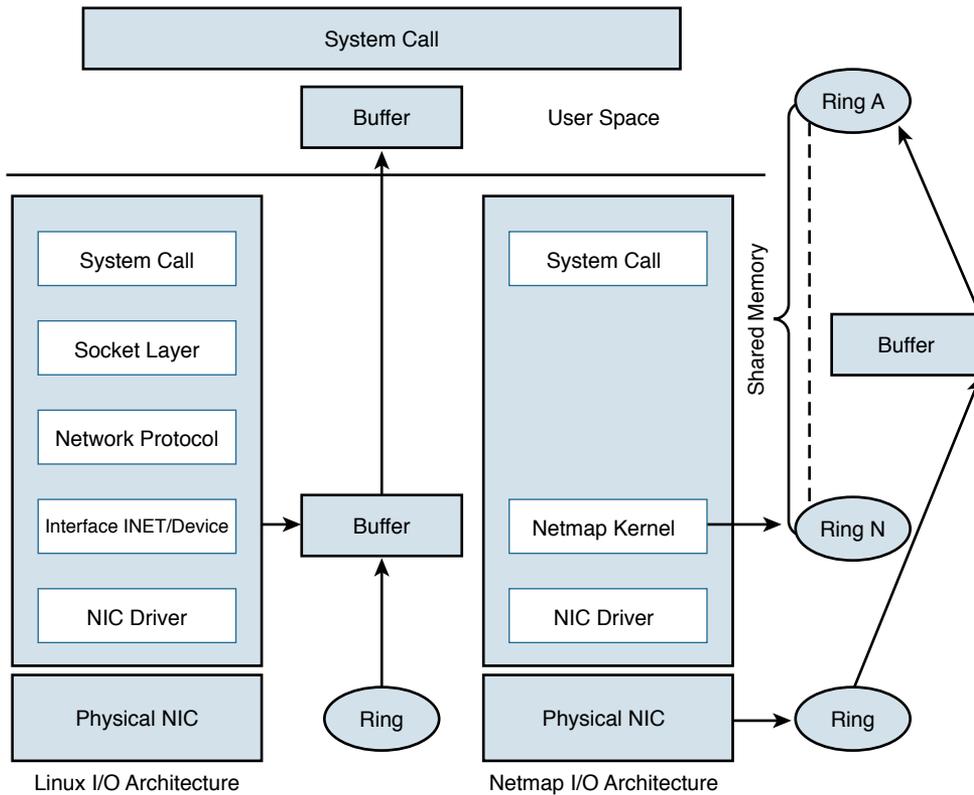


Figure 4-6 *Linux Versus Netmap I/O Comparison*

Packet Flow

There are three major data plane components:

- Rx thread
- Tx thread
- HQF (Hierarchical Queuing Framework) thread

All these components run on a single process within the QFP process umbrella. Multiple PPE threads serve requests within this QFP process. The following sections discuss the flow.

Device Initialization Flow

The following events take place to get the NIC (or vNIC, in a para-virtualized environment) ready for operation:

1. During boot-up, the platform code within IOSd discovers all Linux network interfaces. The platform code then maps these Linux interfaces—`eth0`, `eth1`, and so on—to `Gig0`, `Gig1`, and so on. After talking to the kernel, platform code sets up the interface state (up or down), sets the MTU, sets the ring size, and sets the MAC address.
2. The FMAN process creates the FMAN interfaces and then reaches out to the QFP client process to initialize the data-plane interface.
3. After the QFP process receives the initialization message from the Client process to create an interface, the QFP process then initializes an interface called micro-interface descriptor block (uIDB) in the data plane.
4. After the uIDB is created in the QFP process, the FMAN process binds this uIDB to the network interface name.
5. The component of the data-plane process responsible for interacting with the kernel now has to make sure that the interface created with the QFP process is registered and enabled within the Netmap component of the kernel.
6. With the new interfaces registered, the Netmap component communicates with the virtual NIC driver to initialize the physical NIC.
7. The vNIC driver opens the NIC, initiates the rings, and makes the NIC ready for operation.

TX Flow

The following events take place when there is a packet to be transmitted (Tx) by the CSR onto the wire:

1. The HQF thread detects that there are packets to be sent.
2. The HQF thread checks congestion on the transmit interface and checks the interface states.
3. If the transmit interface is not congested, HQF sends the frame. HQF can also wait to accumulate more frames, batch them, and then send them out.
4. The platform code locates the next available slot in the Tx ring and copies the frame from the source buffer into the Netmap buffer for transmission.
5. The platform code flushes the Tx ring.
6. Netmap forwards the flushed frames to the vNIC driver.
7. The vNIC driver initializes the NIC Tx slots.
8. The vNIC driver writes onto the Tx registers.
9. The vNIC driver cleans up the Tx ring of done slots.
10. The vNIC sends the frame on the wire and generates a notification on completion.

RX Flow

The following events occur whenever a CSR receives a packet to be processed:

1. The Rx thread (the thread that receives frames from the QFP process) issues a poll system call to wait for the new Rx frames.
2. When a new frame arrives, the NIC (or vNIC, in this case) accesses the vNIC Rx ring to get a pointer to the next Netmap buffers.
3. The vNIC puts the frame onto the next Netmap buffers.
4. The vNIC generates an Rx interrupt.
5. The Netmap Rx interrupt service routine runs the Rx threads.
6. The vNIC driver finds the new frame and creates memory buffers for it.
7. The Rx thread pushes the frame to the PPE thread for processing.

Figure 4-7 illustrates packet flow between different XE processes.

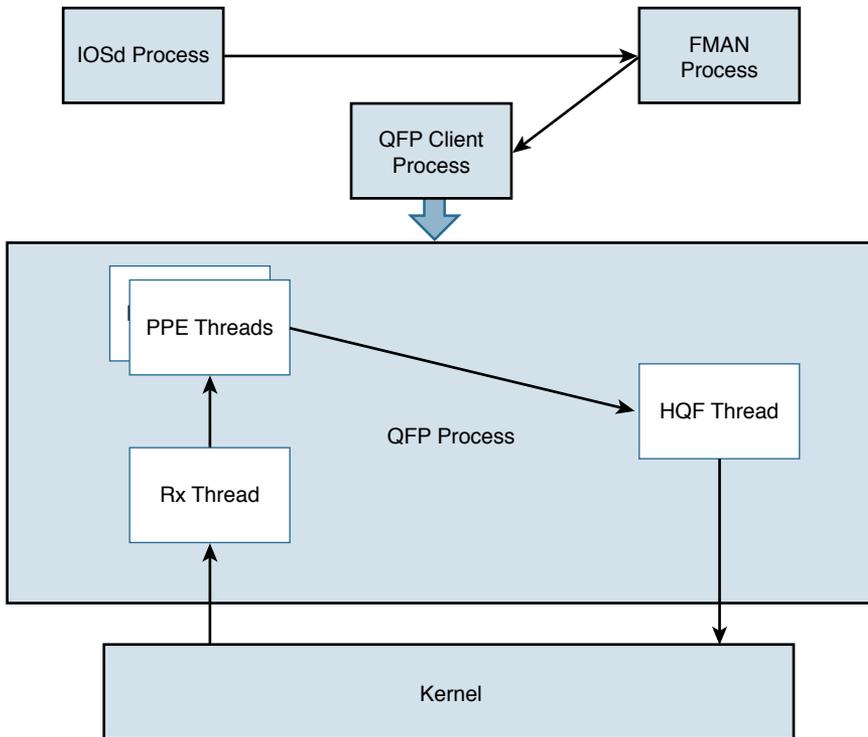


Figure 4-7 Flowchart for Packet Flow

Unicast Traffic Packet Flow

The Tx and Rx flows in Figure 4-7 detail how a packet is transmitted from the NIC (or vNIC, in a para-virtualized driver) to the QFP process. Now we can look at how the QFP process handles the packet after it gets it. The following steps examine a unicast IPv4 packet flow:

1. The QFP process receives the frame from the Netmap Rx and stores it in Global Packet Memory (GPM).
2. The Dispatcher copies the packet header from the GPM and looks for free PPE to assign. The packet remains in the GPM while it is being processed by the PPEs.
3. The Dispatcher assigns a free PPE thread to process the feature on the packet.
4. PPE threads process the packet and gather the packets. The *gather* process copies the packets into B4Q memory and sends the HQF thread a notification that there is a new packet in the B4Q memory.
5. HQF sends the packet by copying it from B4Q into the Netmap Tx ring, and then releases the B4Q buffer.
6. The Ethernet driver sends the frame and frees the Tx ring once the packet has been sent out.
7. Multicast IPsec packets are recycled from the HQF thread back to the in/out processing of the PPE threads.

Figure 4-8 illustrates the packet flow in the QFP process.

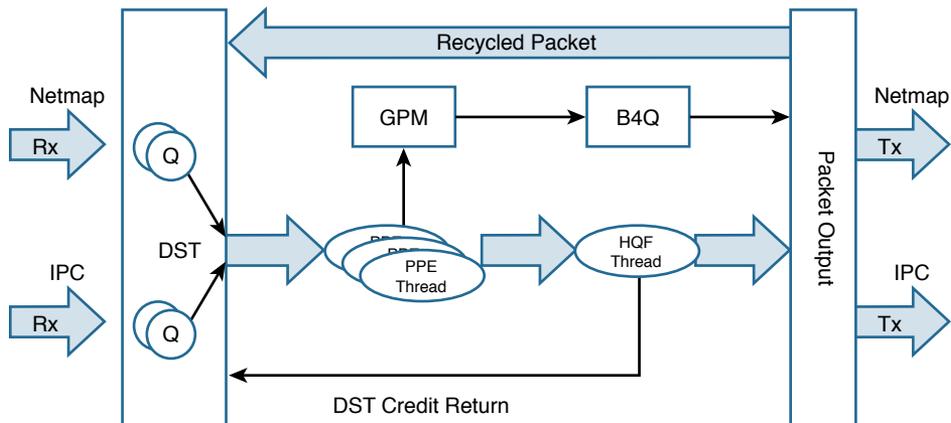


Figure 4-8 CSR 1000V Packet Flow in the QFP Process

Installing the CSR 1000V on a VMware Hypervisor

The process for installing the CSR 1000V on a VMware hypervisor has two phases:

1. Bring up the VM with the CSR 1000V on ESXi.
2. Connect the VNIC with the CSR 1000V.

These phases can be subdivided into the step-by-step procedures described in the following sections. To learn about automated provisioning using the BDEO (build, deploy, execute OVF), see Chapter 7, “CSR in the SDN Framework.”

The following steps assume ESXi is already installed. Please refer to the VMware ESXi installation guide for setting up the ESXi if it is not already installed.

Bringing Up the VM with the CSR 1000V on ESXi

Assuming ESXi is already installed, you can now follow these steps in the first phase of installing the CSR 1000V:

Step 1. Deploy the OVF template:

1. Download the OVF template from software.cisco.com and select CSR 1000V software.
2. Log on to the vSphere client, as shown in Figure 4-9.

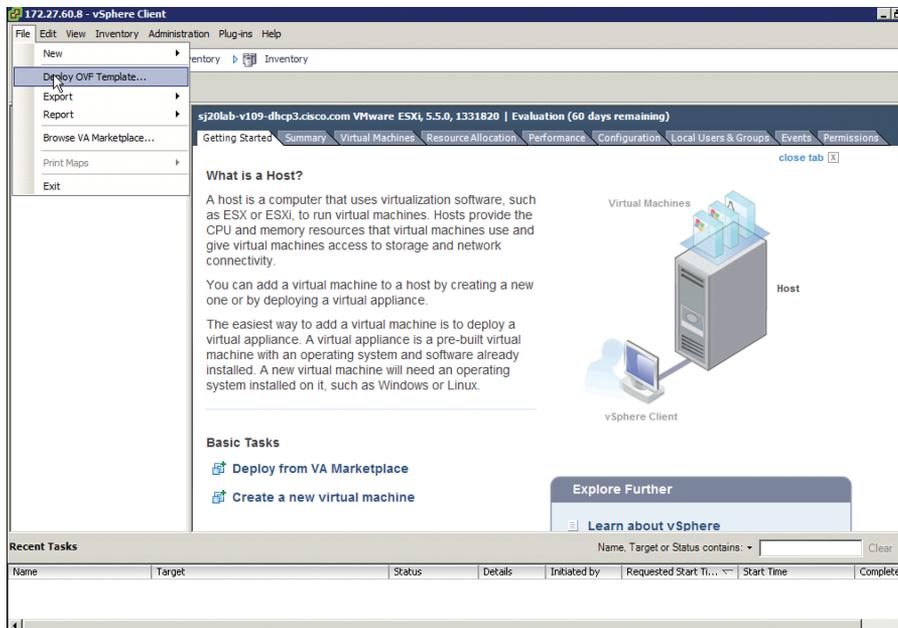


Figure 4-9 Installing the OVF Template for the CSR 1000V

3. Upload the CSR OVF file you downloaded from cisco.com as shown in Figure 4-9.

4. Select File, Deploy OVF Template, as shown in Figure 4-9.

Step 2. Upload the CSR OVF file as shown in Figure 4-10.

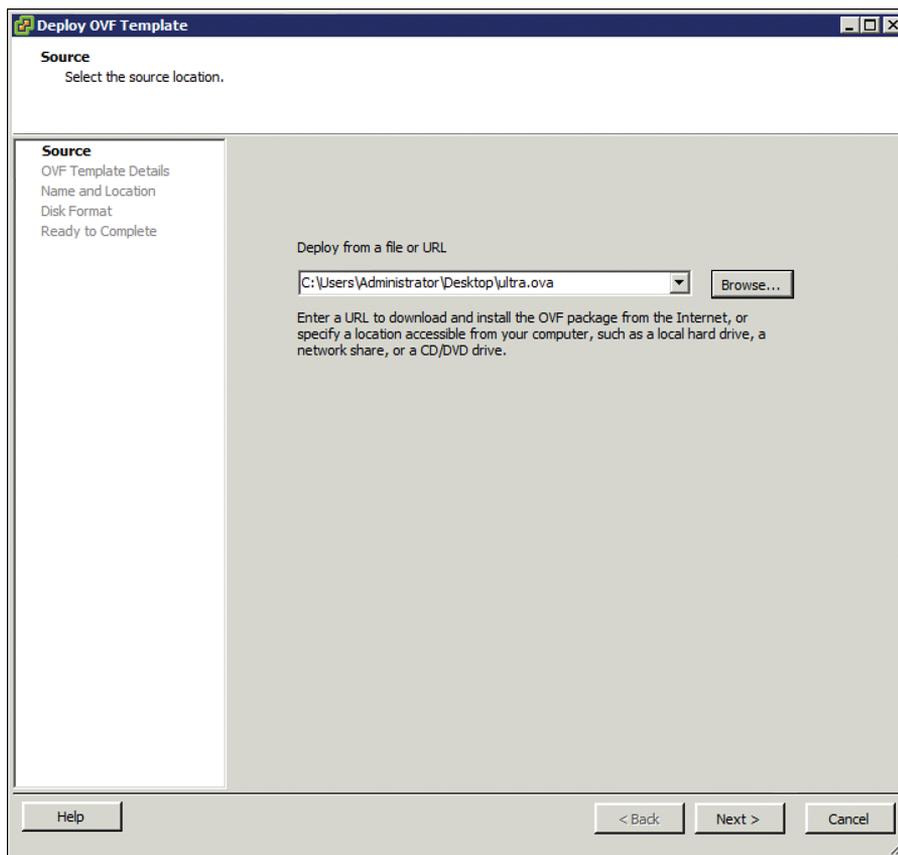


Figure 4-10 *Deploying the OVF Template: Selecting the Source*

Step 3. When the OVA upload is done, verify the OVF template details on the screen shown in Figure 4-11.

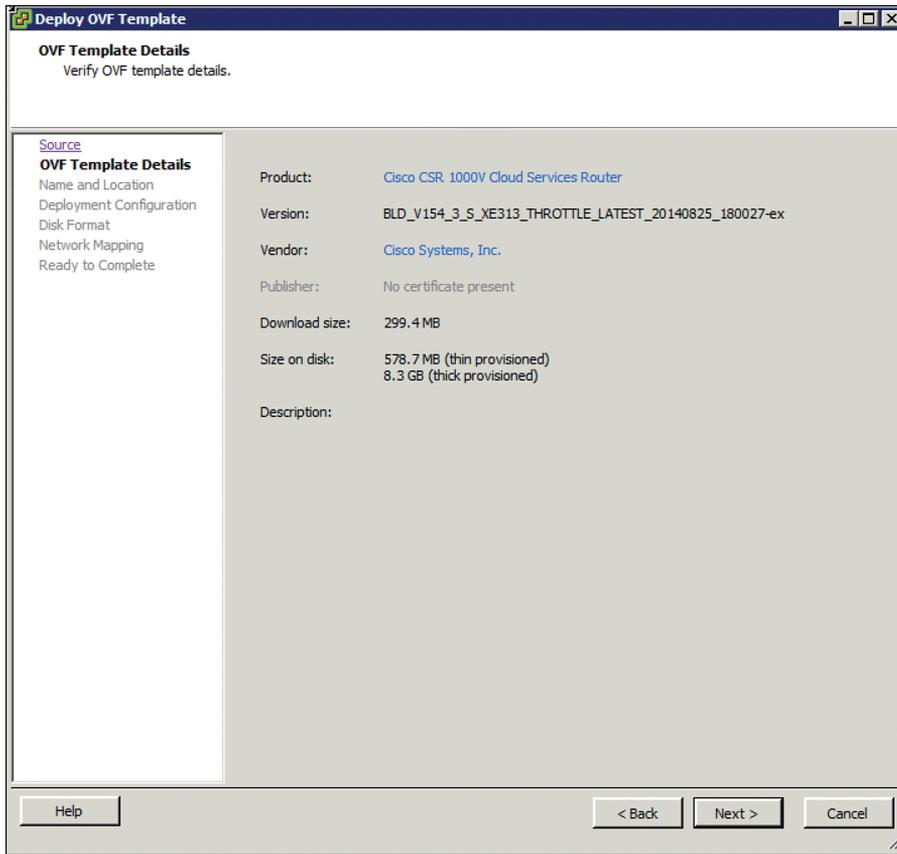


Figure 4-11 *Deploying the OVF Template: Verifying the Template Details*

The release information, product, size, and so on are received from the metadata. Follow the directions for creating the VM.

Complete the following deployment configuration, disk formatting, and network mapping screens, as shown in Figures 4-12 through 4-16:

1. As shown in Figure 4-12, select the hardware profile: Small, Medium, or Large vCPU and RAM, based on the deployment considerations. Refer to the hypervisor documentation for the exact small, medium, and large VM configurations. (You can change this configuration for memory even after the CSR 1000V is brought up.)

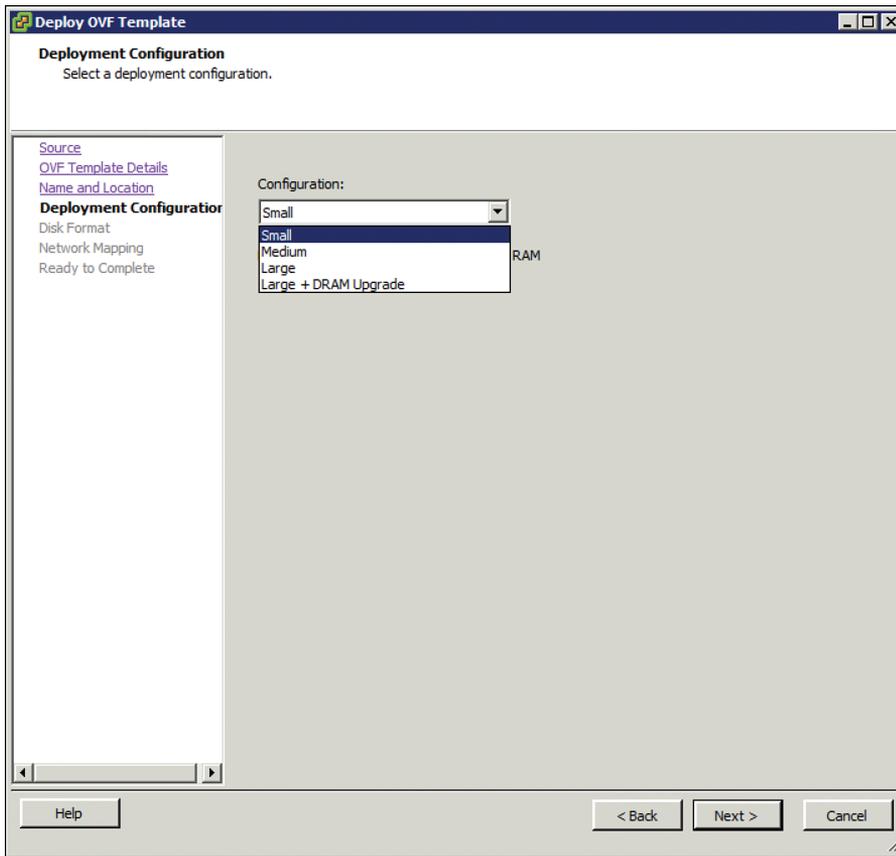


Figure 4-12 *Deploying the OVF Template: Selecting the System Memory Profile for CSR 1000V*

2. Select the appropriate type of disk formatting (see Figure 4-13), and then click Next:
 - **Thick Provision Lazy Zeroed**—With this option, a virtual disk is created with the amount of disk space it has asked for. However, the disk is not cleaned during virtual disk creation. It is cleaned only when you create the first VM on it.
 - **Thick Provision Eager Zeroed**—With this option, a virtual disk is created with the amount of disk space it has asked for. However, the disk is cleaned during virtual disk creation.
 - **Thin Provision**—Choose this option to save space. Initially, the space allocated to a thin disk is less. However, the virtual disk keeps growing as memory requirements grow.

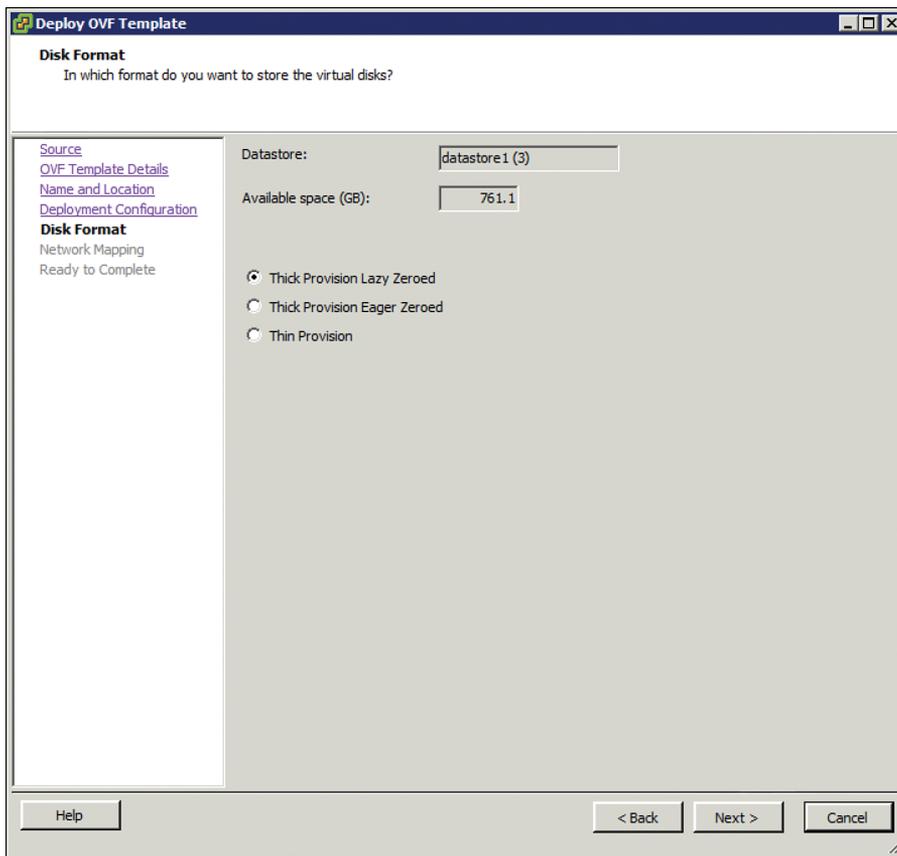


Figure 4-13 *Deploying the OVF Template: Choosing the Disk Provisioning Format*

Note The OVF used here is for version 3.13. You might see variations in the default settings with later versions. Please refer to Cisco release documentation.

3. On the screen shown in Figure 4-14, specify network mapping of the source networks (GigabitEthernet) to the destination networks (VM Network by default) mapping allocation.

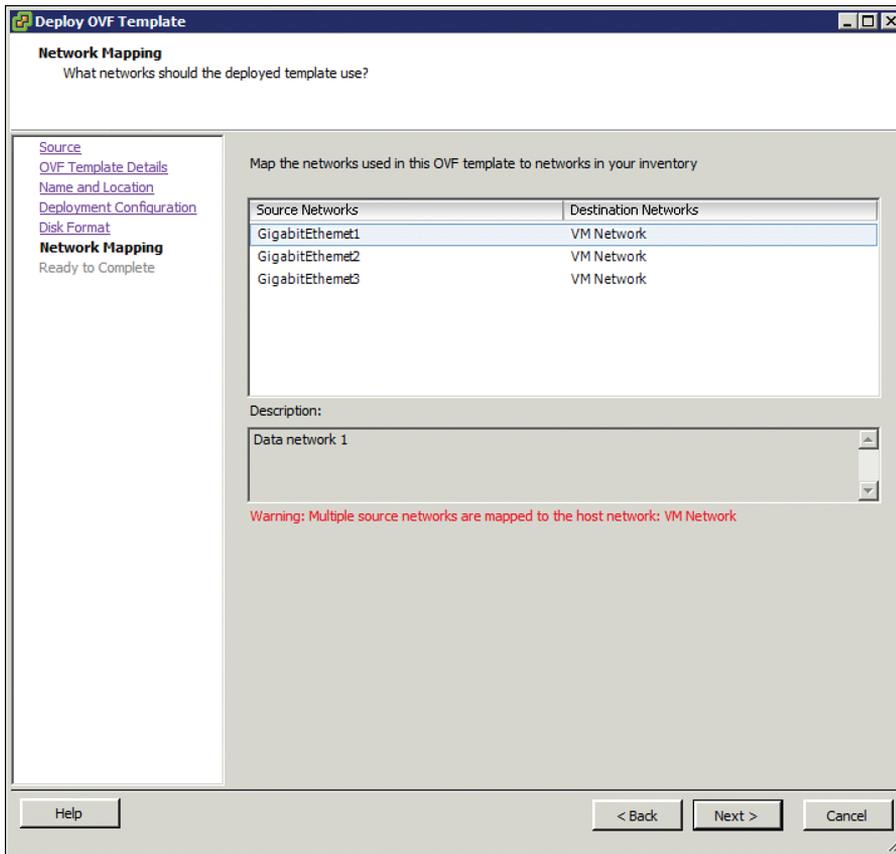


Figure 4-14 Deploying the OVF Template: Network Mapping

4. Look over the summary of the deployed CSR 1000V configuration, as shown in Figure 4-15, and click Finish.

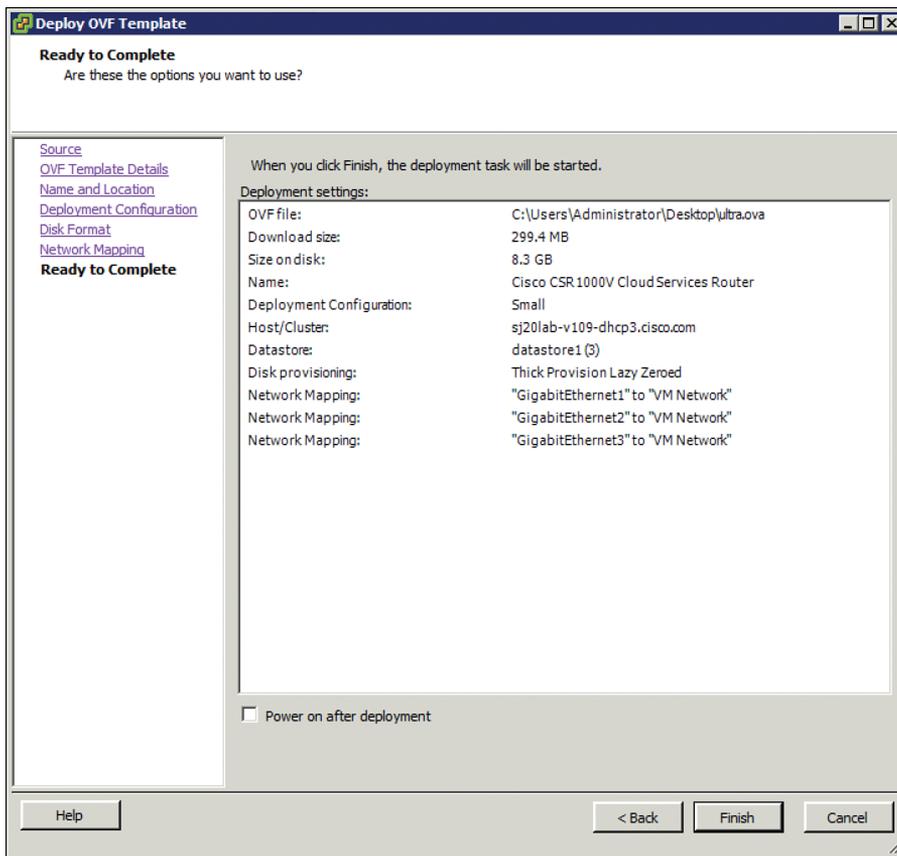


Figure 4-15 *Deploying the OVF Template: Checking the Settings*

- Step 4.** When the deployment of the CSR 1000V is complete, boot the router by selecting the VGA console from the GRUB menu on the Console tab shown in Figure 4-16.

Step 6. To add the serial port for console access, access the vCenter web client and select Virtual Hardware, Network Adaptor, Serial Port, as shown in Figure 4-18.

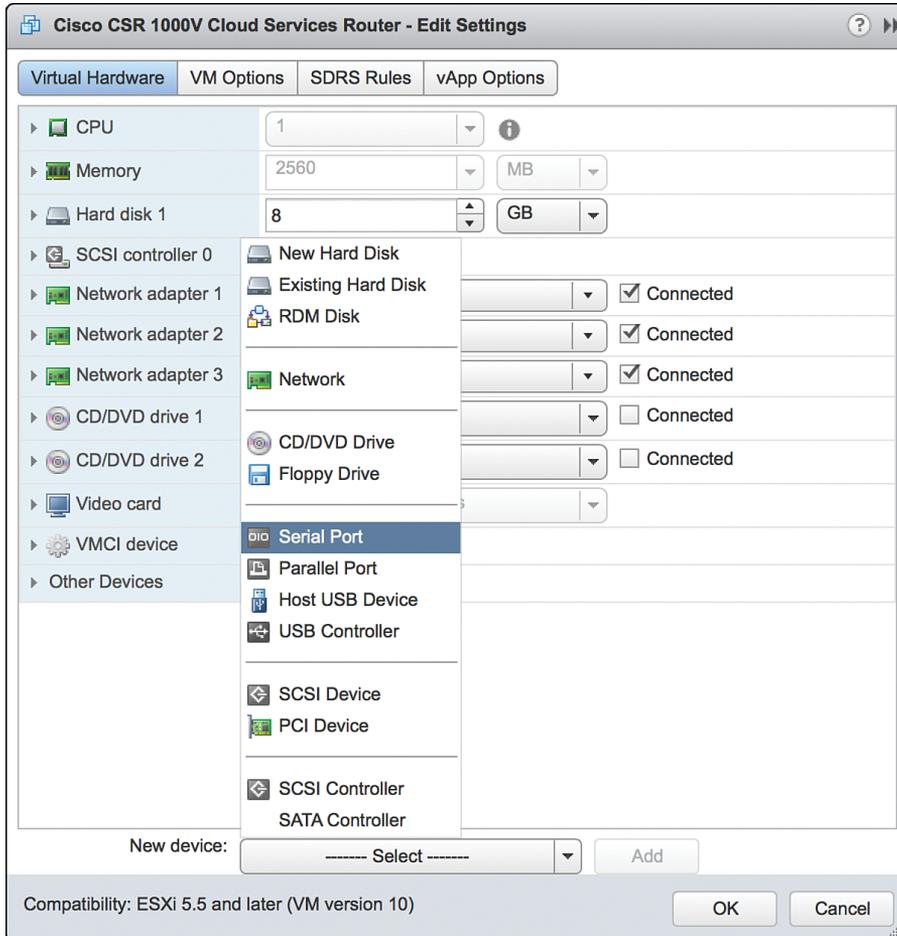


Figure 4-18 VM Access from the vCenter Web Client

Step 7. Shut down the guest OS as shown in Figure 4-19. (Note that this serial port will be used for terminal access to the CSR.)

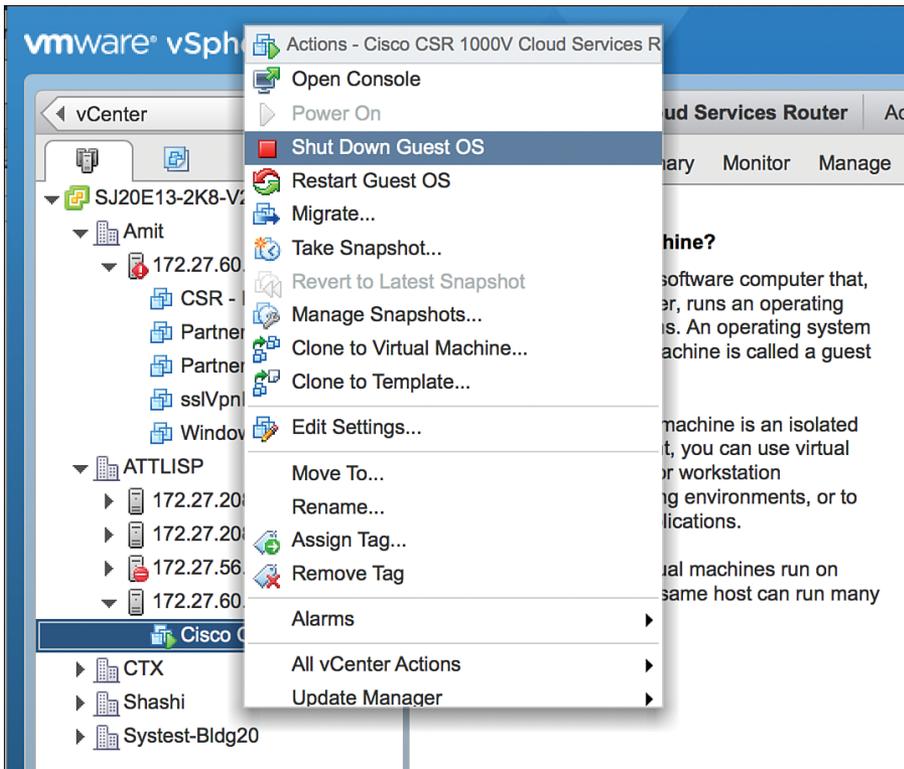


Figure 4-19 *Configuring the Serial Interface: Shutting Down the Router*

- Step 8.** Select Add New Device, New Serial Port and provide the IP address and terminal port details to access the CSR, as shown in Figure 4-20.

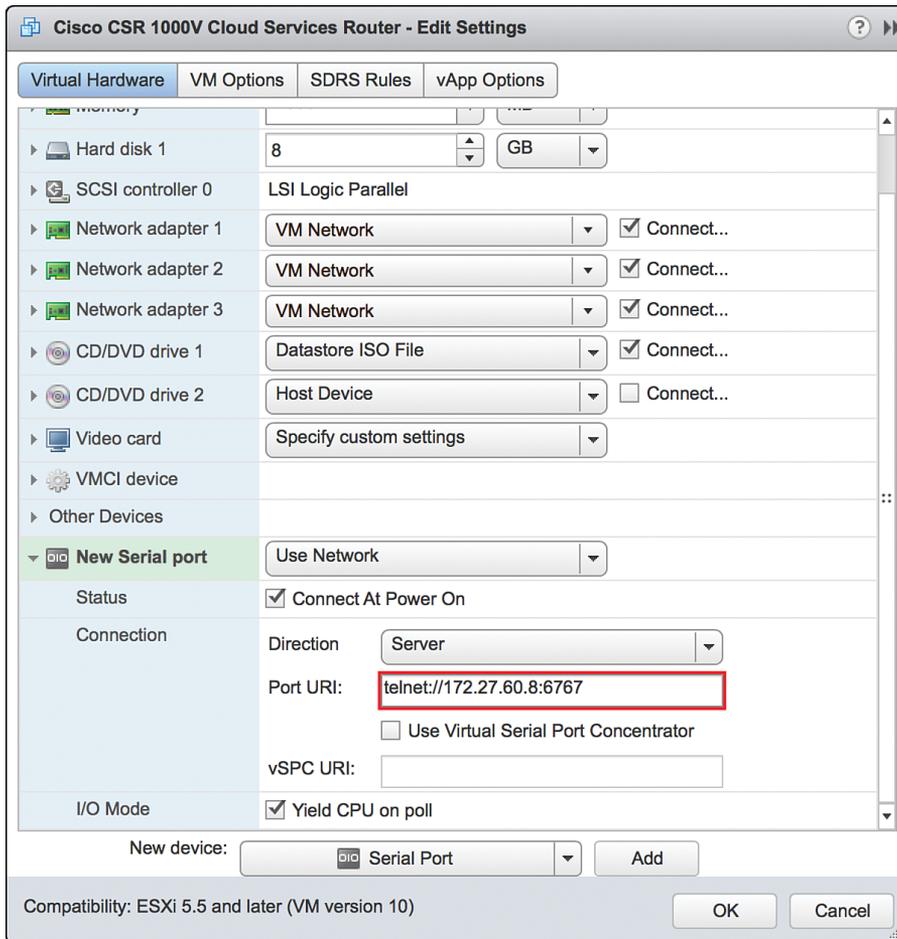


Figure 4-20 *Configuring the Serial Interface: Setting the Telnet Address*

- Step 9.** Go to vCenter and select Setting, Security Profile. Edit security configuration ports 23 and 1024 as shown in Figure 4-21. This is needed because by default ESXi blocks console access.

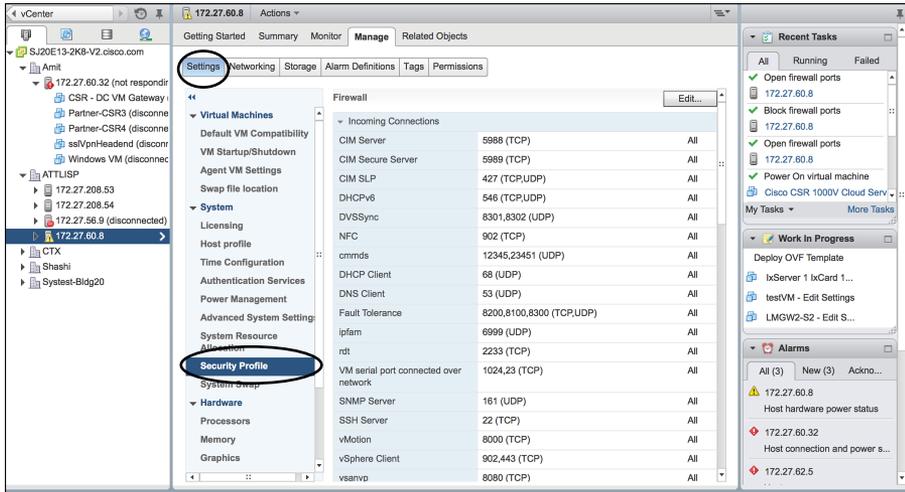


Figure 4-21 *Configuring the Serial Interface: Firewall Settings*

Step 10. Enable ports 23 and 1024 as shown in Figure 4-22.

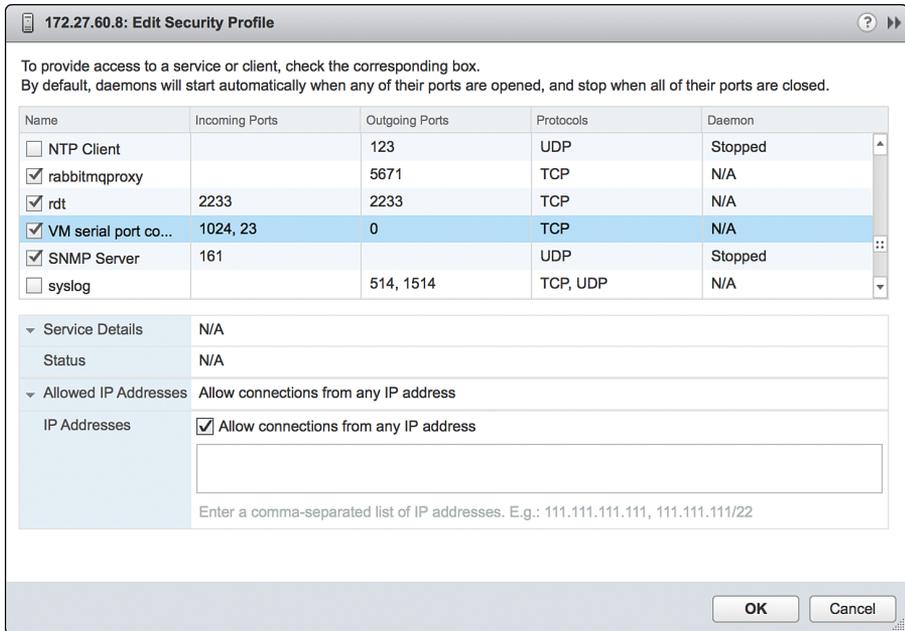


Figure 4-22 *Configuring the Serial Interface: Security Profile Detail*

Step 11. Use Telnet to verify the access from the PC. (It's a good practice to use SSH for accessing the CSR VM; however, for the sake of simplicity, this example shows Telnet access setup.) The EXSi hypervisor defaults the network connections to the VM Network virtual switch connection. The network

adapters are mapped to CSR interfaces. For example, GigabitEthernet1 is mapped to Network adapter 1, and so on. You can verify this by comparing the MAC address as illustrated in Figure 4-23.

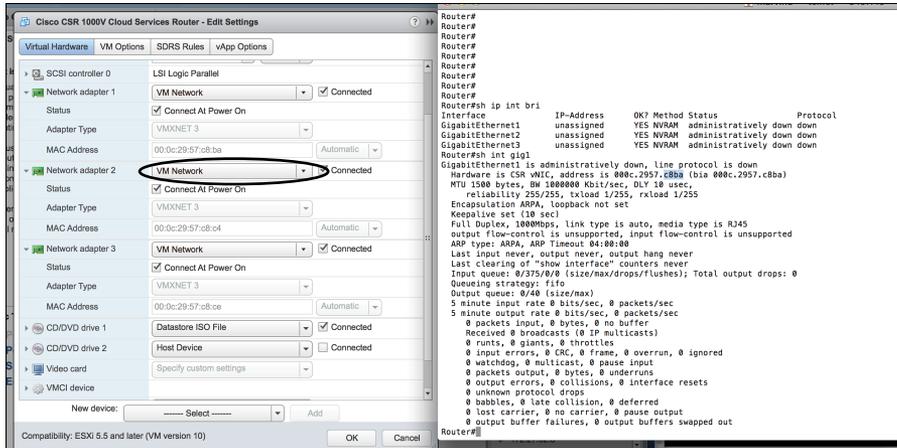


Figure 4-23 CSR 1000V Telnet Access Screen

Step 12. To remap the network adapters to corresponding vNICs, you should perform the following steps. From the vSphere client in the Edit Settings window, select New Device Add, Networking and add vNICs to the CSR as assigned interfaces (from the vCenter web client), as shown in Figures 4-24 through 4-27. (Allow all VLANs and create a bridgeForVNIC1 label for this connection.)

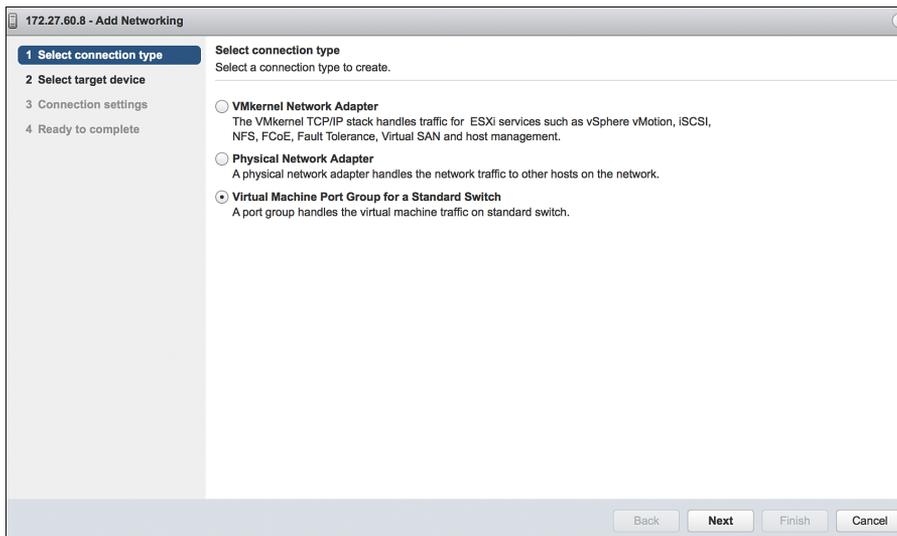


Figure 4-24 vNICs and the CSR 1000V: Selecting the Connection Type

1. Select the new vNIC, as shown in Figure 4-25, to create a new standard switch name.

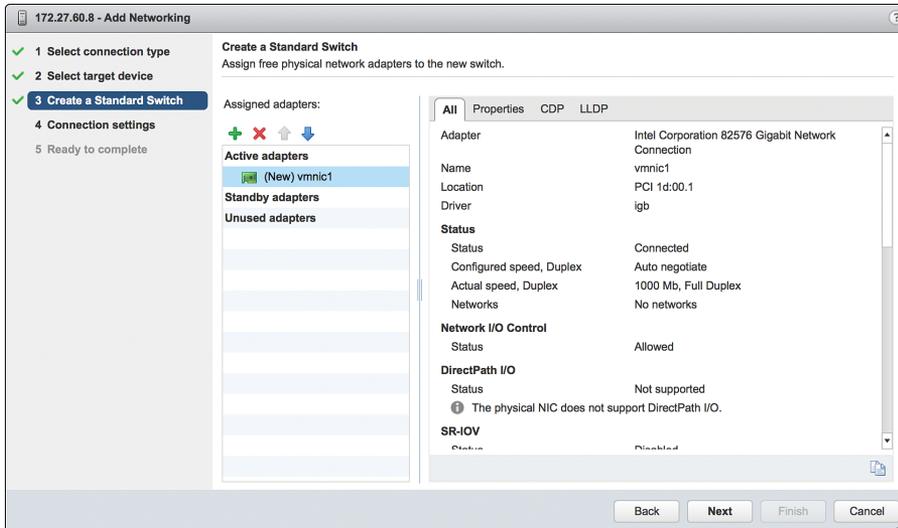


Figure 4-25 *vNICs and the CSR 1000V: Creating a Standard Switch*

2. Add VLANs and the network label assigned for the vNIC, as shown in Figure 4-26.

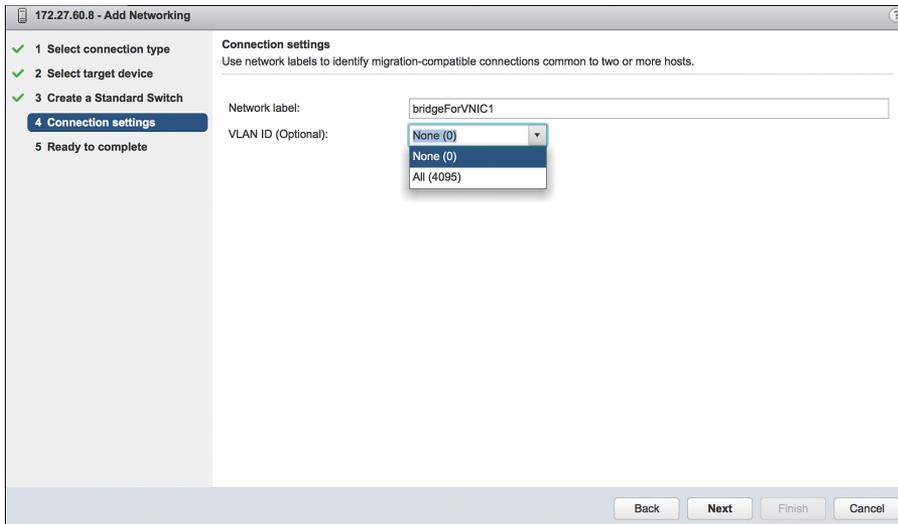


Figure 4-26 *vNICs and the CSR 1000V: Setting the Connection Settings*

3. Complete the configuration of the vNIC with a VLAN and label attachment that can be referenced in a vSwitch. Click Finish to complete this step, as shown in Figure 4-27.

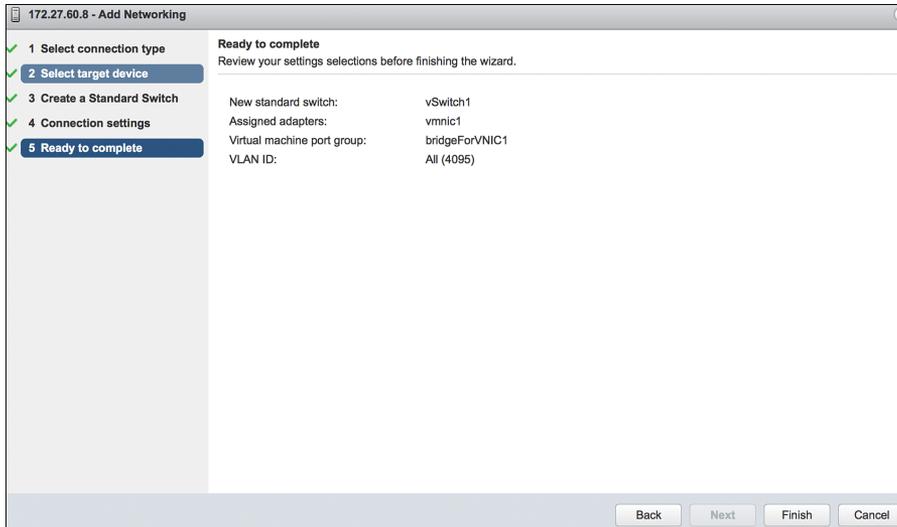


Figure 4-27 *vNICs and the CSR 1000V: Completing the Configuration*

- Step 13.** Go to the vSphere web client and select Virtual Machine, Network Adapter. In the Networking tab, look for the new bridgeForVNIC1 label you created earlier, as shown in Figure 4-28. You should note that this label acts as mapping between the CSR interface and the vNIC.

Repeat Steps 12 and 13 to remap additional network adapters to vNICs available to the CSR.

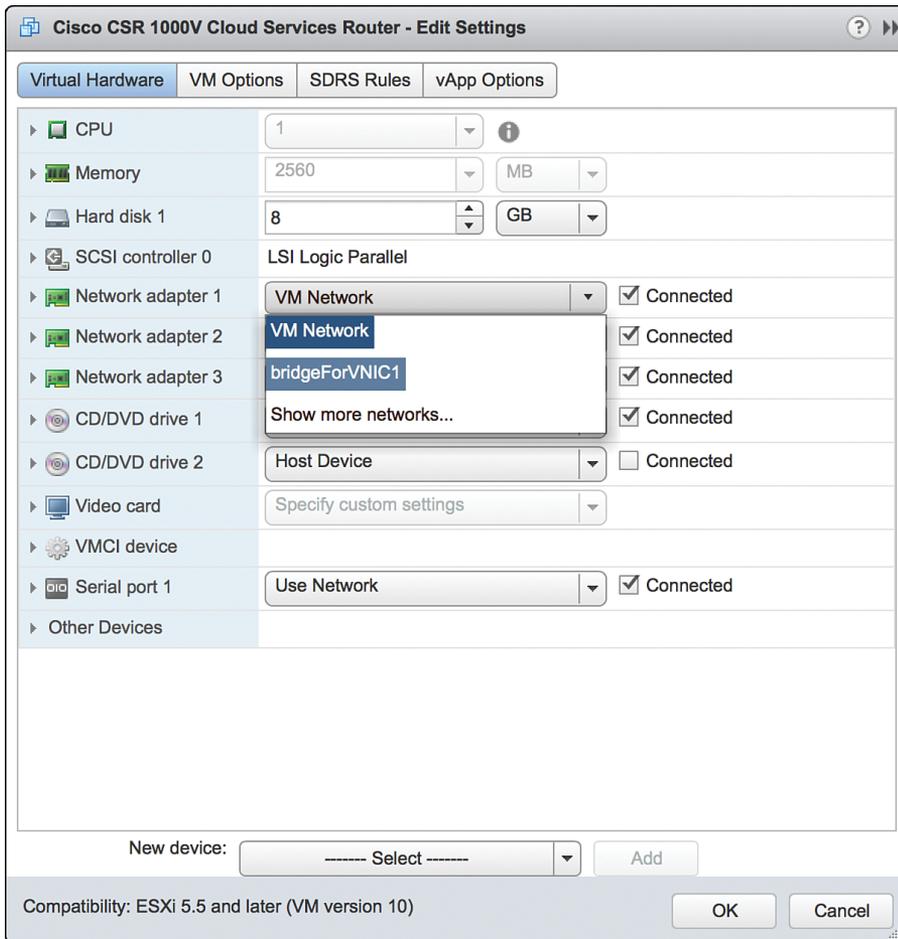


Figure 4-28 *vNICs and the CSR 1000V: Editing the Settings*

To map the network adapter to the vNIC created, select the vNIC label created in the previous step. The CSR 1000V is now configured and connected to the physical NIC, as shown in Figure 4-29.

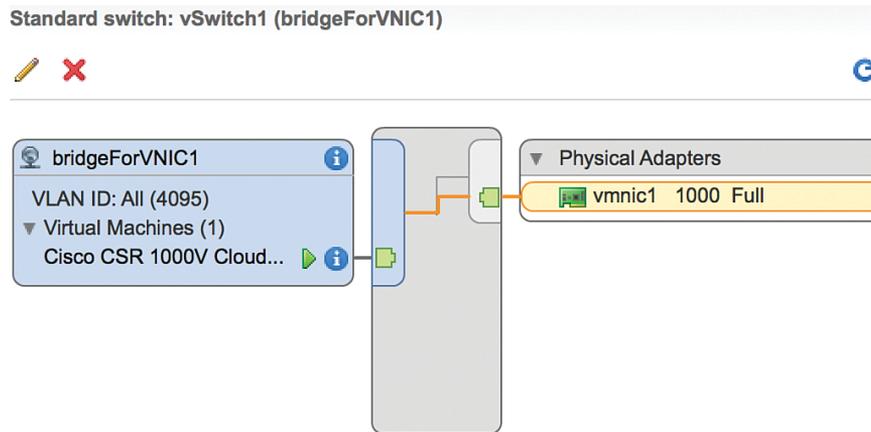


Figure 4-29 vNICs and the CSR 1000V: Interface Summary Screen

Installing the CSR 1000V on a KVM Hypervisor

The process for installing the CSR 1000V on a KVM hypervisor has two phases:

1. Bring up the VM with the CSR 1000V on ESXi.
2. Connect the vNIC with the CSR 1000V.

Bring Up the CSR 1000V as a Guest

Follow these steps to update essential packages on a Linux managed server so it can work as a type 1 hypervisor and run a CSR 1000V VM:

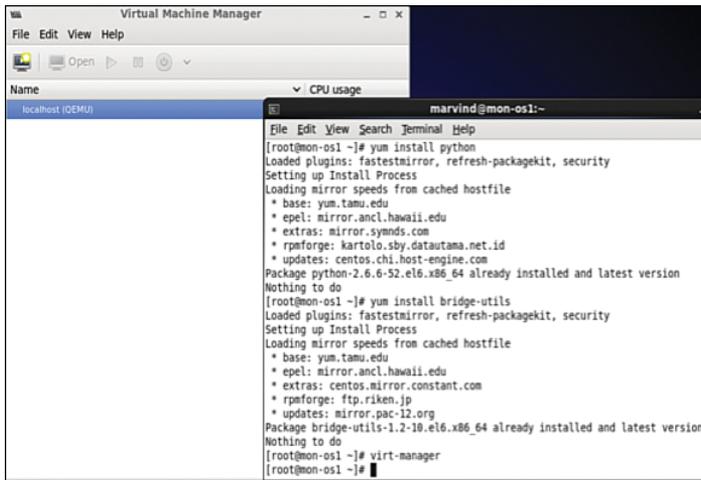
- Step 1.** Install the VM packages `virt-manager`, `qemu-kvm`, and `bridge-utils` like this:

```
apt-get install virt-manager
apt-get install qemu-kvm
apt-get install bridge-utils
```

or like this:

```
yum install virt-manager
yum install qemu-kvm
yum install bridge-utils
```

Figure 4-30 shows the installation of packages required for CSR creation.



```

Virtual Machine Manager
File Edit View Help
Open
Name CPU usage
localhost (QEMU) marvind@mon-os1:~
File Edit View Search Terminal Help
[root@mon-os1 ~]# yum install python
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Loading mirror speeds from cached hostfile
 * base: yum.tamu.edu
 * epel: mirror.ancl.hawaii.edu
 * extras: mirror.symnds.com
 * rpmforge: kartolo.sby.datautama.net.id
 * updates: centos.chi.host-engine.com
Package python-2.6.6-52.el6.x86_64 already installed and latest version
Nothing to do
[root@mon-os1 ~]# yum install bridge-utils
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Loading mirror speeds from cached hostfile
 * base: yum.tamu.edu
 * epel: mirror.ancl.hawaii.edu
 * extras: centos.mirror.constant.com
 * rpmforge: ftp.riken.jp
 * updates: mirror.pac-12.org
Package bridge-utils-1.2-10.el6.x86_64 already installed and latest version
Nothing to do
[root@mon-os1 ~]# virt-manager
[root@mon-os1 ~]#

```

Figure 4-30 Package Installation on a KVM Hypervisor

- Step 2.** Launch Virtual Machine Manager, which is the front end to KVM/QEMU that allows installation and management of CSR VMs, by selecting Application, System, Virtual Machine Manager.

Note Virtual Machine Manager could also be on a different path for your Linux server. Figure 4-31 shows the launch of the virtual machine from QEMU. Make sure you have XDesktop installed. Also note that VMM is not a mandatory requirement for using KVM/QEMU, especially when a graphical user interface is not present on a desktop.

Click the Create a New Virtual Machine icon, and the dialog shown in Figure 4-31 appears. Click the Forward button.

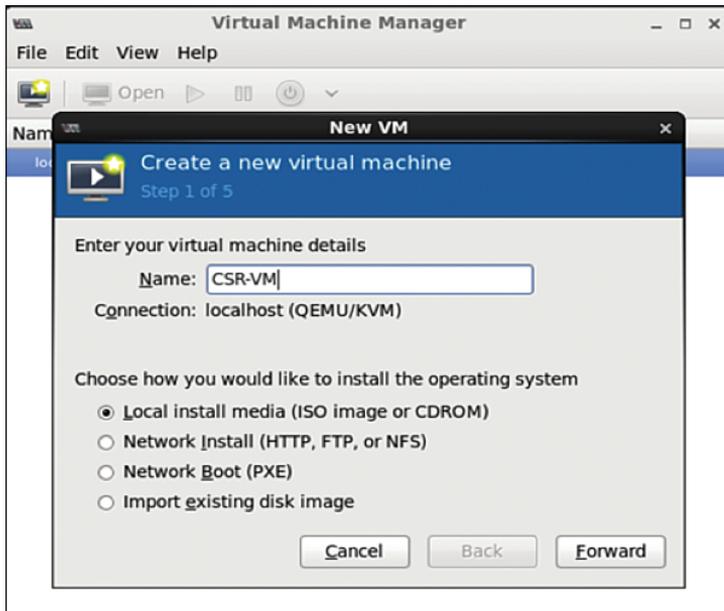


Figure 4-31 *Creating a Guest VM*

- Step 3.** Load the ISO image (which you download from software.cisco.com) for the CSR 1000V, as shown in Figure 4-32. Click the Forward button.

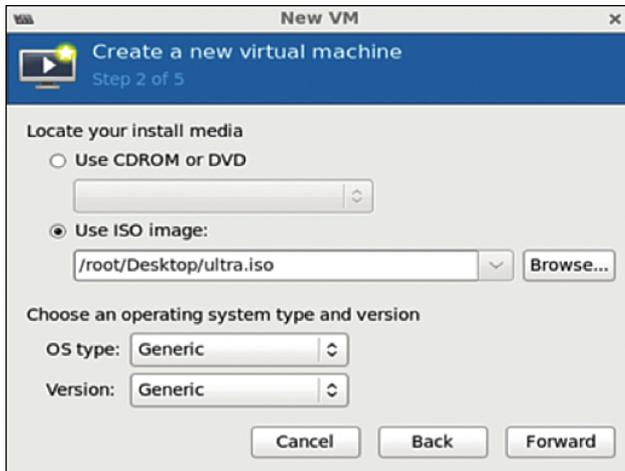


Figure 4-32 *ISO Image Bootup for the CSR 1000V*

Note Download the ISO CSR 1000V image to your local hard disk. When you download it, it is named `csr1000v-universalk9.<version>.std.iso`, but the file is renamed `ultra.iso` in the example shown.

- Step 4.** Allocate hardware resources for the guest VM as shown in Figure 4-33. (Refer to Table 2-2 in Chapter 2 for further allocation information.) Click Forward.

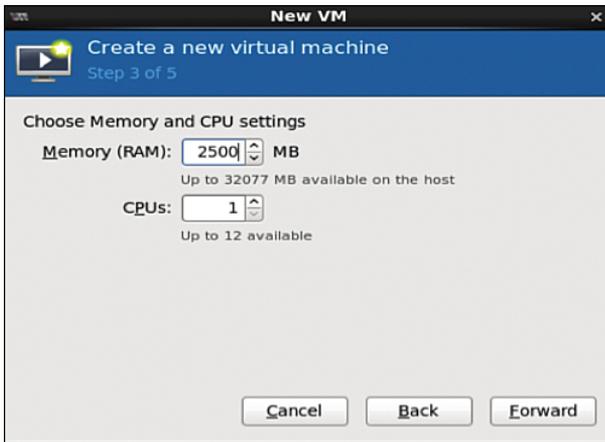


Figure 4-33 *Choosing Memory and CPU Settings*

- Step 5.** Select hardware resources, as shown in Figure 4-34, and click Forward.



Figure 4-34 *Selecting Hardware Resources*

Note If you do not check Allocate Entire Disk Now, only a small portion of memory asked for will be allocated. It will keep growing as memory requirements increase. Checking Allocate Entire Disk Now guarantees that much storage.

Step 6. Look over the hardware resources summary (see Figure 4-35) and make any changes needed. Click Finish.

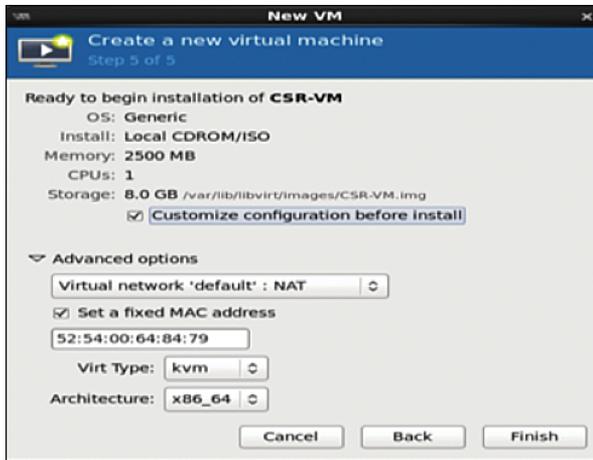


Figure 4-35 Resources Summary Snapshot

Step 7. To apply changes for the guest VM, select Application, System, Virtual Machine Manager and highlight the CSR installed in the VMM. Then click the Show Virtual Hardware Details tab and click the Add Hardware button, as shown in Figure 4-36.

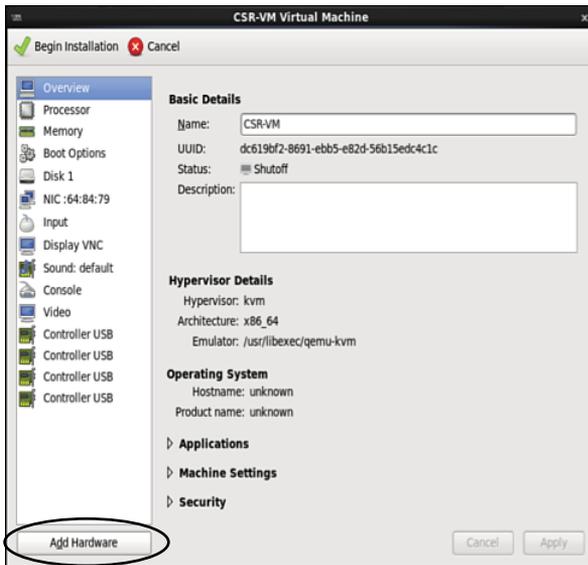


Figure 4-36 *Applying Hardware VM Changes*

- Step 8.** To create serial connection access for console access, select Serial, and then select TCP for Device Type and provide the telnet information, as shown in Figure 4-37.



Figure 4-37 *Creating the Serial Interface*

- Step 9.** In the Virtual Machine Manager, highlight the guest VM and shut it down (if it is not down already). (See Figure 4-38.)

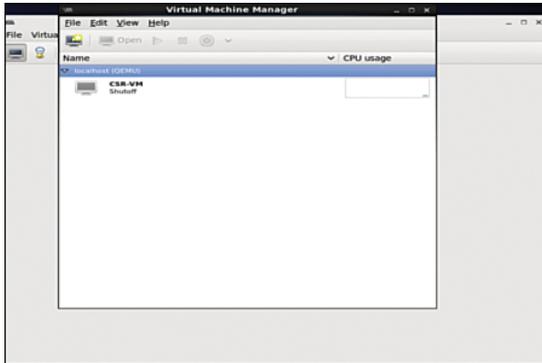


Figure 4-38 *Shutting Down the Guest VM*

The guest VM goes down, as shown in Figure 4-39.

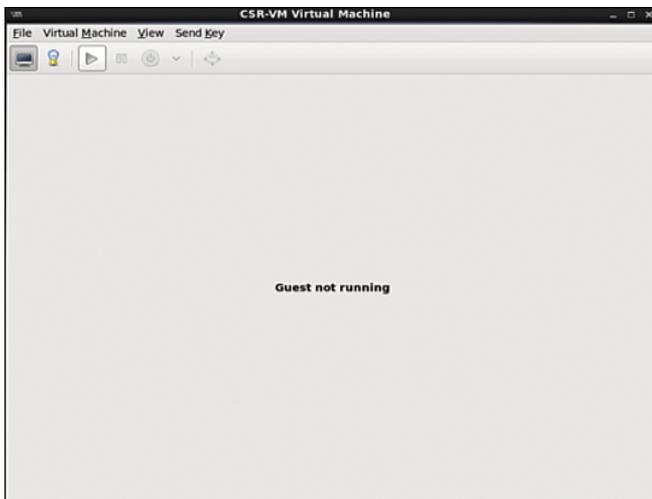
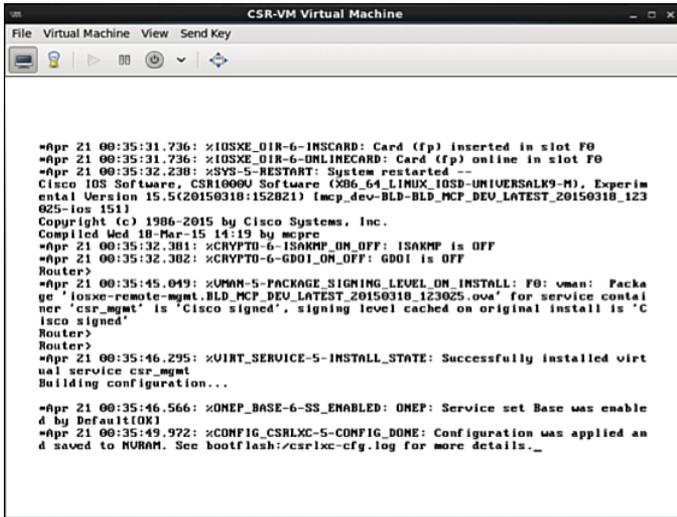


Figure 4-39 *Shutdown of the Guest VM*

- Step 10.** Access the router from the console, as shown in Figure 4-40. Make sure the VM is powered up before you try to access it.



```

CSR-VM Virtual Machine
File Virtual Machine View Send Key

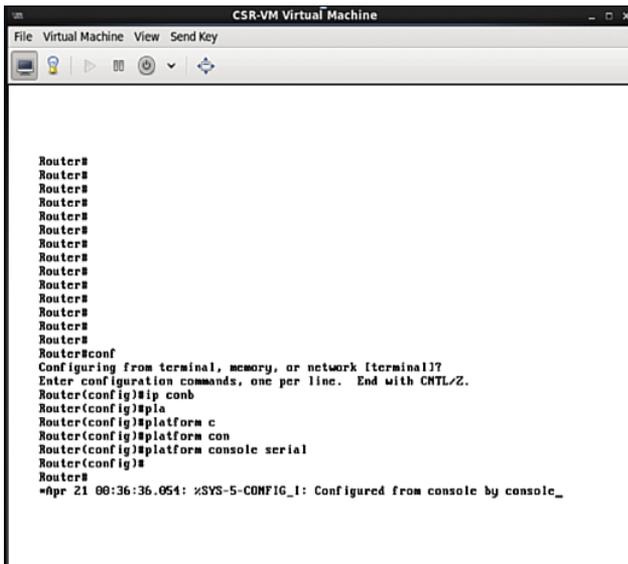
=Apr 21 00:35:31.736: %IOSXE_OIR-6-INSCARD: Card (fp) inserted in slot F0
=Apr 21 00:35:31.736: %IOSXE_OIR-6-ONLINECARD: Card (fp) online in slot F0
=Apr 21 00:35:32.238: %SYS-5-RESTART: System restarted --
Cisco IOS Software, CSR1000U Software (X86_64_LINUX_IOSD-UNIVERSALK9-M), Experim
ental Version 15.5(20150318:152021) [mcp_dev-BLD-BLD_MCP_DEV_LATEST_20150318_123
025-ios 1511]
Copyright (c) 1986-2015 by Cisco Systems, Inc.
Compiled Wed 18-Mar-15 14:19 by mcpre
=Apr 21 00:35:32.381: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is OFF
=Apr 21 00:35:32.382: %CRYPTO-6-GDOI_ON_OFF: GDOI is OFF
Router>
=Apr 21 00:35:45.049: %UPAN-5-PACKAGE_SIGNING_LEVEL_ON_INSTALL: F0: vman: Packa
ge 'iosxe-remote-mgmt.BLD_MCP_DEV_LATEST_20150318_123025.ova' for service contai
ner 'csr_mgmt' is 'Cisco signed', signing level cached on original install is 'C
isco signed'
Router>
Router>
=Apr 21 00:35:46.295: %VIRT_SERVICE-5-INSTALL_STATE: Successfully installed virt
ual service csr_mgmt
Building configuration...

=Apr 21 00:35:46.566: %ONEP_BASE-6-SS_ENABLED: ONEP: Service set Base was enable
d by Default[OK]
=Apr 21 00:35:49.972: %CONFIG_CSRLXC-5-CONFIG_DONE: Configuration was applied an
d saved to NVRAM. See bootflash:/csrlxc-cfg.log for more details._

```

Figure 4-40 Console Access to the KVM

Step 11. Use the serial interface command for telnet access: `platform console serial` and write `mem`, as shown in Figure 4-41.



```

CSR-VM Virtual Machine
File Virtual Machine View Send Key

Router#
Router#conf
Configuring from terminal, memory, or network [terminal]?
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip conb
Router(config)#ip
Router(config)#platform c
Router(config)#platform con
Router(config)#platform console serial
Router(config)#
Router#
=Apr 21 00:36:36.054: %SYS-5-CONFIG_I: Configured from console by console_

```

Figure 4-41 Router Console for Telnet Access

Step 12. Access the CSR 1000V via the telnet, as shown in Figure 4-42.

```

Terminal
File Edit View Search Terminal Help
to administratively d
Building configuration...
own
*Apr 21 00:43:30.239: %SYS-6-BOOTTIME: Time taken to reboot after reload = 385
seconds
*Apr 21 00:43:33.174: %VMAN-5-PACKAGE_SIGNING_LEVEL_ON_INSTALL: F0: vman: Packa
ge 'iosxe-remote-mgmt.BLD_MCP_DEV_LATEST_20150318_123025.ova' for service contai
ner 'csr_mgmt' is 'Cisco signed', matches signing level cached on original insta
ll, signing level allowed is 'Cisco signed'
*Apr 21 00:43:33.313: %VIRT_SERVICE-5-INSTALL_STATE: Successfully installed virt
ual service csr_mgmt
*Apr 21 00:43:35.059: %ONEP_BASE-6-SS_ENABLED: ONEP: Service set Base was enable
d by Default[OK]
*Apr 21 00:43:40.671: %CONFIG_CSRLXC-5-CONFIG_DONE: Configuration was applied an
d saved to NVRAM. See bootflash:/csrlxc-cfg.log for more details.%IOSXEBOOT-4-NE
T-SMP-AFFINITY: (local/local): reached 2 minutes limit for broken_parity_status
IOSXEBOOT-4-NET-SMP-AFFINITY: (rp/0): reached 2 minutes limit for broken_parity_
status
Router>
Router>
Router>
Router>

```

Figure 4-42 *Telnet Connection to the CSR 1000V*

- Step 13.** Ensure that your virtual machine is shut down, and then start vNIC provisioning by selecting Show Virtual Hardware Details, NIC, as shown in Figure 4-43.

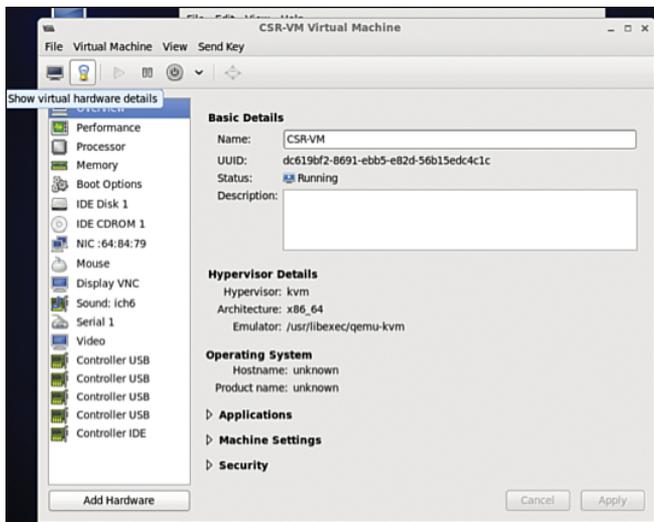


Figure 4-43 *Accessing CSR 1000V Network Settings*

- Step 14.** In the Virtual Machine Manager, select virtio as the device model (see Figure 4-44) because it is the para-virtualized driver in Linux. Using virtio is the best way to exploit the underlying kernel for I/O virtualization. It provides an efficient abstraction for hypervisors and a common set of I/O drivers.



Figure 4-44 *Selecting CSR 1000V Network Settings*

Select the virtual network with NAT to tie all VMs in the same bridge domain and NAT it to the outgoing physical interface (see Figure 4-45). Attach the other NIC to the bridge tap.

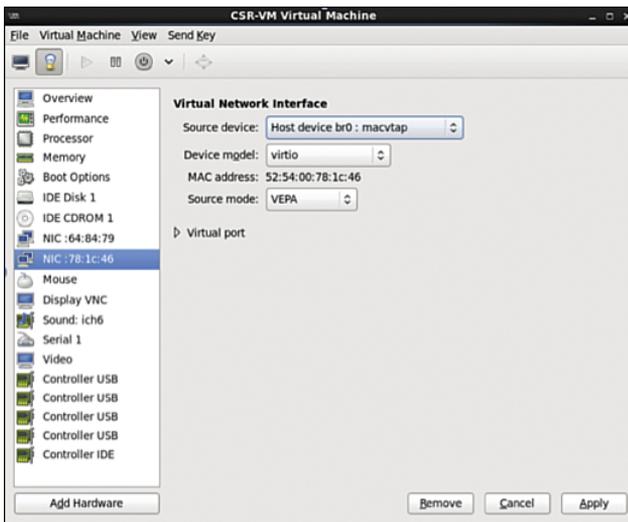


Figure 4-45 *CSR 1000V NIC Settings*

In KVM, `macvtap` is a combination of the `macvlan` driver and a Tap device. Here the function of the `macvlan` driver is to create virtual interfaces and map virtual interfaces to physical network interfaces. A unique MAC address identifies each virtual interface to the physical interface. A TAP interface is a software only interface that exists only in the kernel. You use Tap interfaces to

enable user-space networking and allow passing of datagrams directly between VMs instead of sending datagrams to and from a physical interface. The `macvtap` interface combines these two functions together (see Figure 4-46).

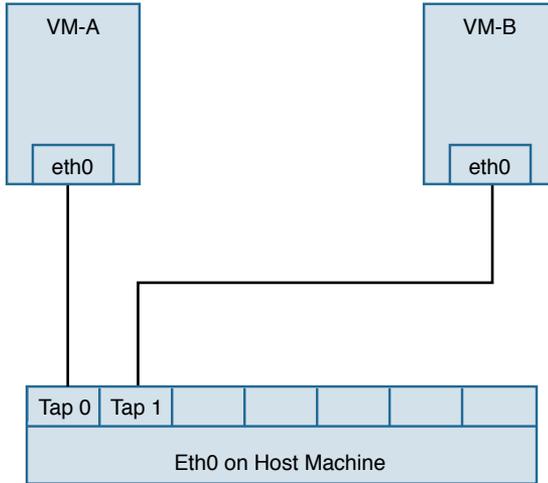


Figure 4-46 `macvtap` Diagram

Step 15. Configure the mapping of the vNIC to the physical interface:

1. Access the directory `/etc/network/interfaces/ifcfg-br0` on the Ubuntu host and view the bridge type (see Figure 4-47).

```

Terminal
File Edit View Search Terminal Help
DEVICE=br0
TYPE=Bridge
BOOTPROTO=static
ONBOOT=yes
TYPE=bridge
STP=on
*ifcfg-br0* 6L, 70C

```

Figure 4-47 Bridge Configuration File Output

- Hypervisor scheduling
- CPU pinning

This section reviews the common tuning practices for an ESXi host. The scheduler for ESXi is responsible for vCPU, IRQ (interrupt requests), and I/O threads. To provide equal treatment to all guest VMs, the scheduler provides allocation of equal resources of vCPU threads for scheduling. Note that you can relax coscheduling of threads to avoid synchronization latency.

To tweak the scheduling and resource allocation details, you must access the VM setting using vSphere client and follow these steps:

1. In the vSphere client inventory, right-click the virtual machine and select Edit Settings.
2. Click the Resources tab and select CPU.
3. Allocate the CPU capacity for this virtual machine.

The Processor Affinity setting (CPU pinning) restricts VMs to a particular set of cores by defining the affinity set. The scheduling algorithm aligns with process affinity for assigning the resources used for the tasks. Figure 4-49 assumes two tasks: Task 1 and Task 2. Task 1 has affinity to processor 1 and is using it. When Task 2 needs a resource, the scheduler uses a second processor. Task 2 then acquires affinity with the second processor.

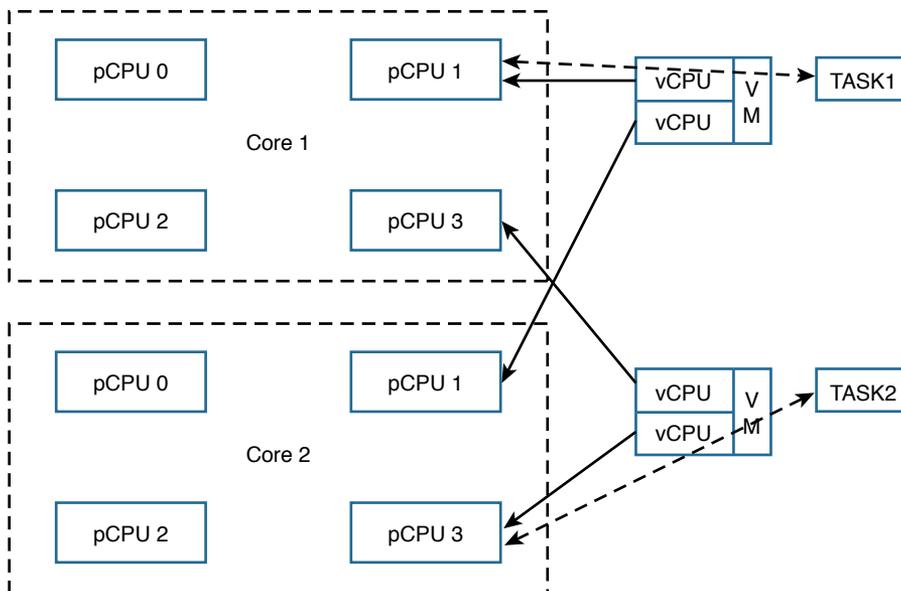


Figure 4-49 CPU Pining

To tweak these settings, access the vSphere client and follow these steps:

1. In the vSphere client inventory panel, select a virtual machine and select Edit Settings.
2. Select the Resources tab and select Advanced CPU.
3. Click the Run on Processor(s) button.

You achieve CPU pinning in KVM by issuing the following command:

```
sudo virsh vcpupin test 0 6
```

Hyperthreading by definition allows a single physical core to have two logical cores; that is, a single core can execute two threads at a given time. Each process from the guest VM can be split into multiple threads to a logical CPU, and the CPU can handle multiple threads of independent tasks. The main function of hyperthreading is to increase the number of tasks in the pipeline by creating parallel pipelines. By tweaking the process affinity option, you can restrict VMs to a particular set of cores and unhook the VM from processor scheduling. Most of the hypervisors use BIOS settings to modify the hyperthreading feature.

For predictable performance, the following best practices are recommended:

- Ensure that hyperthreading is turned off.
- Use CPU pinning to allow the guest VMs to dedicate one or more physical hardware CPUs for processing.
- For CSR 1000V performance optimization, it is important to understand the concept of DirectPath I/O and SR-IOV (single root I/O virtualization). These are driver virtualization and are beneficial for achieving very high packet rates with low latency. In DirectPath I/O, you can map only one physical function to one virtual machine. SR-IOV allows an admin to share a single physical device, so that multiple virtual machines can connect directly to the physical function.

These features are supported in all hypervisors, and it is important to understand the settings on the hypervisor deployed in order to optimize guest VM performance with features used on the hypervisor.

Summary

Now that you've read this chapter, you should have an understanding of the CSR 1000V data plane architecture, as well as packet flow. You should also have an understanding of the steps for bringing up a CSR 1000V on ESXi and KVM hypervisors.

This page intentionally left blank

Index

A

A9 processor, 1

active/active flow distribution to
 cloudburst, redirection access
 model, 193

adoption, Enterprise cloud, 29-30

algorithms, scheduling, 60-62

Amazon web services deployment, CSR,
 211-215

Amazon web services deployment, CSR
 1000V, 216-222

application-centric infrastructure
 (SDN), 224

Application Virtual Switch (AVS), 270

architecture, CSR 1000V, troubleshooting,
 271-272

ASR (Aggregation Service Router) 1000
 router, 41, 96-97

 architectural elements, 95

 ESPs (embedded service processors),
 42-43

 RP (route processor), 42

 SIP (SPA interface processor), 43

ASR (Aggregation Service Router)
 1001, 97

 logical architecture, 97

 virtualizing into CSR 1000V, 98

ATM (Asynchronous Transfer Mode), 13

attach-device command (virsh), 90

attach-disk command (virsh), 90

attach-interface command (virsh), 90

automation, 247-248

 BDEO tool, 248-249

 management, 247

 NSO tool, 249-251

NFV orchestration with OpenStack,
 252-253, 260-261, 264-266

 versus orchestration, 247

 provisioning, 247

availability zones, VPC, 214

AVC (Application Visibility and Control),
 CSR 1000V, 52-53

AVS (Application Virtual Switch), 270

AWS (Amazon web services) deployment
 CSR, 211-215

 CSR 1000V, 216-222

B

BadUidbSubIdx drop type (IOS), 285

Basic Input/Output System (BIOS), 67

BDEO tool, 248-249

BIOS (Basic Input/Output System), 67

BIOS settings, hosts, 276

block-based storage, 3

boot process (IOS), 66-67

BPG route reflector, CSR, 155-157

 hierarchical use, 157-162

BqsOor drop type (IOS), 285

branch design, CSR 1000V, planning,
 162-168

- show interface controller, 283
- show interfaces, 282
- speed, 278
- statistics drop, 285
- computation, data centers, 3**
- conceptual architecture, data center virtualization, 5-6**
- configuration**
 - CPU usage, 281-282
 - hardware and software speed, 278-279
 - hosts, 275-276
 - interface-to-NIC mapping, 281
 - memory usage, 282-288
- connectivity, Enterprise cloud, 26-28**
- containers, 8**
- control planes, LISP, 171-175**
- controlled resources, 83**
- core partitioning, hypervisors, 75**
- CPUs**
 - pining, 138
 - settings, hosts, 275
 - scheduling algorithms, 60-62
 - usage configuration, 281-282
- CPU scheduler, VMkernel, 76**
- create command (virsh), 89**
- Create Router dialog (OpenStack), 238**
- critical priority, ready queue, 65**
- Critical queue (IOS scheduler), 38**
- crypto engine, CSR 1000V, 103**
- crypto maps, IPsec VPNs, 143-144**
- CSR (cloud service router)**
 - BPG route reflector, 155-157
 - hierarchical use, 157-162*
 - host configuration, 275-276
 - hypervisors, 59
 - LISP (Locator/ID Separation Protocol), 168-169, 175
 - control plane, 171-175*
 - data plane, 171*
 - ETR (egress tunnel router), 169-170*
 - IP mobility, 175*
 - IPv6 migration, 175*
 - ITR (ingress tunnel router), 169-170*
 - MR (MAP Resolver), 170*
 - network-to-network connectivity, 175-176*
 - network-to-network interconnection configuration, 176-182*
 - PETR (proxy egress tunnel router), 170-171*
 - PITR (proxy ingress tunnel router), 170*
- OpenStack
 - instantiating Neutron plugin, 242-245*
 - tenant deployment, 235-242*
- public cloud deployment, Amazon web services, 211-215
- as Remote Access VPN server, 153-155
- remote VPN access into Cloud, 153-155
- secure inter-cloud connectivity, 152
- CSR 1000V, 37, 95**
 - AVC (Application Visibility and Control), 52-53
 - branch design planning, 162-164
 - virtualization, 164-168*
 - cloudburst, 190
 - Cisco inter-cloud fabric, 194-195*
 - data synchronization, 191*
 - direct access model, 191-192*
 - network connectivity, 190*
 - redirection access model, 192-193*
 - workload migration, 191*
 - data plane, 103-104
 - architecture, 100-102*
 - Netmap I/O, 104-105*
 - packet flow, 106-109*
 - DMVPN (Dynamic Multipoint VPN), 53
 - EEM (Embedded Event Manager), 54
 - initiation process, 99-100
 - installing
 - KVM hypervisor, 126-137*
 - VMware hypervisor, 110-125*
 - IP SLA (IP Service Level Agreement), 54
 - LISP (Location/ID Separation Protocol), 54
 - MPLS (Multiprotocol Label Switching) VPN, 54-55
 - multitenant data center, 185-190
 - zone connectivity, 188*
 - as Neutron router, 206-209
 - OTV (Overlay Transport Virtualization), 55
 - performance tuning, 137-139
 - PfR (Performance Routing), 55
 - private cloud deployment in OpenStack, 195-211

- public cloud deployment, 211
 - Amazon web services, 216-222*
- Radio Aware Routing, 56
- Redundancy Group Infrastructure, 56
- software crypto engine, 103
- system design, 95-98
- as tenant router, 209-211
- troubleshooting, 271
 - architecture overview, 271-272*
 - debugging packet loss, 276-292*
 - I/O configuration, 272-276*
- virtualizing ASR 1001 into, 98
- VM (virtual machine), 271
 - layers, 103*
- VPLS (Virtual Private LAN Services), 55
- VPN service gateway, 148-153
- VPN services, 141
 - L2VPNs, 141*
 - L3VPNs, 142-148*
- VXLAN (Virtual Extensible LAN), 56
- ZBFW (Zone Based Firewall), 56-57
- CSR 1000V routers, 44-45**
 - deployment requirements, 45-47
 - elastic performance and scaling, 47-48
 - network extension from premises to cloud, 51
 - rapid deployment, 49
 - routing flexibility, 49
 - secure cloud VPN gateway, 50-51
 - segmentation within cloud, 52
- CSRVR, bringing up as guest, 126-137**

D

- daemon (IOSd), 40-41**
- DAL (Database Abstraction Layer), 202**
- data, unstructured, 4**
- data centers, 1-2**
 - computation, 3
 - distributed servers, 2
 - evolution, 2
 - facilities, 3
 - multitenant, 16-18
 - CSR 1000V, 185-190*
 - logical diagram, 185*
 - virtual service block design, 186*
 - network fabric, 3
 - physical hardware, 1
 - SDN, 224-225
- service blocks
 - deployment, 188*
 - placement, 186*
- services, 3
- storage, 3
- utilization, 2
- virtualization, 2-5
 - conceptual architecture, 5-6*
 - evolution, 5*
 - network, 12-14*
 - server, 6-8*
 - service, 15-16*
 - storage, 9-12*
- data plane (CSR 1000V), 103-104**
 - architecture, 100-102
 - LISP, 171
 - Netmap I/O, 104-105
 - packet flow, 106
 - device initialization, 106-107*
 - Rx, 108*
 - Tx, 107*
 - unicast traffic, 109*
- data synchronization, CSR 1000V**
 - cloudburst, 191**
- Database Abstraction Layer (DAL), 202**
- DCUI (Direct Console User Interface)**
 - processes, VMkernel, 77
- dead queue (IOS scheduler), 65**
- debugging packet loss, 276**
 - CPU usage, 281-282
 - ESXi, 289-292
 - hardware and software configurations, 278-279
 - high-level packet flow, 276-277
 - interface-to-NIC mapping, 281
 - L2 MTU, 280
 - memory usage, 282-288
 - throughput license, 277-278
 - vSwitch packet drops, 289
- define command (virsh), 89**
- deploying OpenStack, 225-233**
 - CSR tenant, 235-242
- deployment models, cloud, 20-21**
- deployment requirements, CSR 1000V**
 - routers, 45-47
- descriptors, VNF, definition, 261-264**
- design**
 - CSR 1000V, 95-98
 - branch design, 162-168*

- hypervisors
 - core partitioning*, 75
 - microkernel architecture*, 74
 - monolithic architecture*, 74
 - operating systems, 60
 - physical resource management*, 60-62
 - software access to resources*, 62
 - design, cloud, 21-23
 - on-demand service, 21
 - destroy command (virsh), 89
 - detach-device command (virsh), 90
 - detach-disk command (virsh), 90
 - detach-interface command (virsh), 90
 - device-based network virtualization, 14-15
 - device-based storage virtualization, 11
 - device drivers
 - ESXi hypervisor, 78-79
 - legacy versus native, 79
 - device initialization packet flow, CSR
 - 1000V data plane, 106-107
 - direct access model, CSR 1000V
 - cloudburst, 191-192
 - Direct Connect, VPC, 214
 - Disabled drop type (IOS), 285
 - disaster recovery using cloudburst,
 - redirection access model, 193
 - DMVPNs (Dynamic Multipoint VPNs),
 - 144-145
 - CSR 1000V, 53
 - overlays, 190
 - DNS (domain name service), diagram, 168
 - Domain 0 (Xen), 93
 - Domain 10 framework
 - abstraction and virtualization, 23
 - automation and orchestration, 23
 - customer interface, 24
 - infrastructure and environment, 22-23
 - organization, governance, and process, 25-26
 - platform and application, 24
 - security and compliance, 24-25
 - service catalog and financials, 24
 - domain controllers, Glance, 202
 - Domain U (XEN), 93
 - Domain U PV (XEN), 93
 - domains, 201
 - dombkstat command (virsh), 90
 - domid command (virsh), 89
 - domifstat command (virsh), 90
 - dominfo command (virsh), 90
 - domname command (virsh), 90
 - domstate command (virsh), 90
 - domuuid command (virsh), 89
 - DR (disaster recovery) using cloudburst,
 - redirection access model, 193
 - Driver, CSR data plane, 102
 - dumpxml command (virsh), 89
 - Dynamic Multipoint VPN (DMVPN), 53, 144-145
- ## E
-
- EC2 (Elastic Compute Cloud),
 - Amazon, 211
 - EEM (Embedded Event Manager), CSR
 - 1000V, 54
 - Elastic Compute Cloud (EC2),
 - Amazon, 211
 - elastic IP, VPC, 213
 - elastic performance, CSR 1000V routers, 47-48
 - elasticity, cloud, 21
 - embedded service processors (ESPs), ASR (Aggregation Service Router), 42-43
 - encapsulation, GRE (Generic Routing Encapsulation), 142
 - enlightened guests, 8
 - Enterprise cloud
 - adoption challenges, 29-30
 - connectivity, 26-28
 - ESPs (embedded service processors)
 - ASR (Aggregation Service Router), 42-43
 - processes, 98
 - ESXi
 - bringing up VM with CSR 1000V, 110-125
 - mapping, 281
 - NIC stats, 290
 - opening screen, 80
 - packet debugging, 289-292
 - port stats, 292
 - VM list, 289

ESXi hypervisor, 75

- device drivers, 78-79
- file systems, 79-80
- management, 80-81
- VMkernel, 75-76

- CIM processes, 77-78*

- CPU scheduler, 76*

- DCUI processes, 77*

- memory management, 76*

- VMM processes, 77*

- VMX processes, 77*

ETR (egress tunnel router), 169-170**examples**

- Changing the Speed of the Interface, 279

- Configuration Script Sample, 240

- Control-Processor CLI Output, 282

- CSR 1000V as a DMVPN Hub, 149-150

- CSR as a Remote Access VPN Server with an AnyConnect Client, 153-155

- ESXi NIC Stats, 290

- ESXi Port Stats, 292

- Example of Using the glance CLI to Add a CSR 1000V Image, 236

- How to Check Throughput Levels and License Details, 277

- Interface Controller Output, 283-285

- Interface show Command, 280

- ISR as a DMVPN Spoke, 150-152

- Kernel Images Available to GRUB Are Listed in menu.lst, 67-68

- LMGW Configuration, 177-179

- MS/MR Configuration, 179-181

- Programs That Are Executed on the Full-Multiuser Run Level, 69

- QFP Feature Debugging Options, 286-288

- R1 Configuration, 159

- R2 Configuration, 159

- R3 Configuration, 161

- R4 Configuration, 161

- R5-1 Configuration, 160

- R5-2 Configuration, 160

- RR1 Configuration, 158

- RR2 Configuration, 158

- RR-3 Configuration, 160

- RR-4 Configuration, 161

- Sample Answer File for Packstack, 231-233

- Sample Definition of VNF Descriptors, 261-264

- Sample of Installed NED Verification, 260

- Sample NSO Initiation to Understand Input to Be Used in Service and Device Model Framework, 253-260

- Sample VNF Instantiation, 264

- show Commands, 182

- show interfaces CLI Output, 283

- Snapshot of BGP Update at R4, 162

- sw-nic CLI Output, 288

- VM List on an ESXi Host, 289

Extensible Messaging and Presence Protocol (XMPP), 32**F**

- facilities, data centers, 3

- FIFO (first in, first out) scheduling algorithm, 60

- file systems, VMFS, 79-80

- firewalls, ZBFW (Zone Based Firewall), 56

- first in, first out (FIFO) scheduling algorithm, 60

- Forwarding Manager (IOSd), 41

- full server virtualization, 8

- functions

- malloc(), 69

- vPATH, 270

G

- Generic Routing Encapsulation (GRE) tunnels, 142

- GET VPNs (Group Encrypted Transport VPN), 145-147

- Glance component (OpenStack), 34

- Glance project, OpenStack, 201-202

- GM (Group Member) device, 146

- GNU Grand Unified Boot Loader (GRUB), 66-68

- GRE (Generic Routing Encapsulation) tunnels, 142

- Group Encrypted Transport VPNs (GET VPNs), 145-147

- Group Member (GM) devices, 146

- groups, 201

- GRUB (GNU Grand Unified Boot Loader), 66-68

guest emulator (QEMU), 85-87
 guest mode, Linux kernel, 83

H

hair pinning of traffic, 27
 hardware, speed configuration, 278-279
 hardware hub VPN access, 27
 hardware VPN access, 26
 Heat project, OpenStack, 205
 help command (virsh), 89
 high-level packet flow, debugging, 276-277
 high priority, ready queue, 65
 High queue (IOS scheduler), 38
 Horizon component (OpenStack), 34
 Horizon project, OpenStack, 205
 host-based storage virtualization, 11
 host machines, 272
 hosts, configurations, 275-276
 hybrid cloud, 20
 hybrid kernels, 64
 Hyper-V, 91-92
 hypercalls, 72-73
 hyperthreading setting, BIOS, 276
 hypervisors, 59, 71-72, 94
 design, 74
 core partitioning, 75
 microkernel architecture, 74
 monolithic architecture, 74
 ESXi, 75
 device drivers, 78-79
 file systems, 79-80
 management, 80-81
 VMkernel, 75-78
 Hyper-V, 91-92
 KVM, 82-83
 architectural components, 84-85
 guest emulator, 85-87
 installing CSR 1000V on, 126-137
 Libvirt, 88-91
 PCI passthrough mode, 274
 software, 272
 versus operating systems, 72-73
 VMware, installing CSR 1000V on, 110-125
 Xen, 92-93

IaaS (Infrastructure as a Service), 2, 18-19
 idle queue (IOS scheduler), 65
 Infrastructure as a Service (IaaS). *See* IaaS (Infrastructure as a Service)
 initiation process, CSR 1000V, 99-100
 installation
 CSR 1000V on KVM hypervisor, 126-137
 CSR 1000V on VMware hypervisor, 110-125
 instantiation (VNF), 264-265
 inter-cloud fabric, CSR 1000V cloudburst, 194-195
 interface, changing speed, 278-279
 Interface Manager (IOSd), 41
 interface-to-NIC mapping, configuration, 281
 Internet for transport, Enterprise cloud connectivity, 26-28
 Internet gateway (VPC), 213
 Internetworking Operating System (IOS). *See* IOS (Internetworking Operating System)
 I/O configuration, CSR 1000V, troubleshooting, 272-276
 I/O stack, Netmap, 105
 IOS (Internetworking Operating System), 37-39
 boot process, 66-67
 IOSd (IOS daemon), 40-41
 kernel, 64
 scheduler, 65-66
 scheduler, 37
 XE architecture, 39
 kernel, 40
 XE drop types, 285
 IOSd (IOS daemon), 95
 IOS XE, 96
 versus IOS, 96-98
 IOS XE operating system, 95
 IP mobility, LISP, 175
 IP SLA (IP Service Level Agreement), CSR 1000V, 54
 IPsec VPNs, 142
 with crypto maps, 143-144
 offloading, 3

Ipsilon Networks, 13
 Ipv4NoAdj drop type (IOS), 286
 Ipv4NoRoute drop type (IOS), 286
 IPv6 migration, LISP, 175
 ITR (ingress tunnel router), 169-170

K

Kernel-based Virtual Machine (KVM). *See*
 KVM (Kernel-based Virtual Machine)
 kernels, 63
 ESXi hypervisor, VMkernel, 75-78
 hybrid, 64
 IOS, 64
 memory manager, 65-66
 scheduler, 65
 IOS XE, 40
 KVM (Kernel-based Virtual Machine),
 82-83
 architectural components, 84-85
 guest emulator, 85-87
 Linux, memory management, 69-71
 microkernels, 63
 hypervisor architecture, 74

Key Server (KS), 146
 Keystone component (OpenStack), 34
 Keystone project, OpenStack, 199-201
 KS (Key Server), 146
 KVM (Kernel-based Virtual Machine),
 82-83
 architectural components, 84-85
 guest emulator, 85-87
 hypervisor
 installing CSR 1000V on, 126-137
 Libvirt, 88-91

L

L2 MTU, configuration, 280
 L2VPNs (Layer 2 VPNs), 141
 L3VPNs (Layer 3 VPNs), 141-143
 DMVPNs (Dynamic Multipoint VPNs),
 144-145
 GET VPNs (Group Encrypted Transport
 VPN), 145-147
 GRE tunnels, 142
 IPsec VPNs, 142
 IPsec VPNs with crypto maps, 143-144

MPLS VPNs, 142
 site-to-site VPNs, 143
 SSL VPNs, 147-148
 legacy drivers versus native drivers, 79

Libvirt

 management daemon, 88
 user tools, 89-91
 virsh, 89-91

Linux

 memory management, 69
 caching, 71
 overcommitment, 69-70
 swap space, 69-71
 versus Netmap I/O, 105

LISP (Location/ID Separation Protocol),

 168-169, 175, 190
 control plane, 171-175
 CSR 1000V, 54
 data plane, 171
 ETR (egress tunnel router), 169-170
 IP mobility, 175
 IPv6 migration, 175
 ITR (ingress tunnel router), 169-170
 MR (MAP Resolver), 170
 network-to-network connectivity,
 175-176
 network-to-network interconnection
 configuration, 176, 179-182
 packet header, 172
 PETR (proxy egress tunnel router),
 170-171
 PITR (proxy ingress tunnel router), 170

LISP-to-MPLS Gateway (LMGW),

 176, 179
 list command (virsh), 89
 LMGW (LISP-to-MPLS Gateway),
 176, 179

Locator/ID Separation Protocol (LISP).

See LISP (Locator/ID Separation
 Protocol)

low priority, ready queue, 65
 Low queue (IOS scheduler), 38

M

mainframes, virtualization, 17
 malloc() function, 69
 management, automation, 247

management daemon (Libvirt), 88
 user tools, 89-91

map resolver (MR), 176

map server (MS), 176

MBR (Master Boot Record), 67

measured services, cloud, 21

medium priority, ready queue, 65

Medium queue (IOS scheduler), 38

memory, physical, 76

memory management

- Linux
 - caching*, 71
 - over commitment*, 69-70
 - swap space*, 69-71
- VMkernel, 76

memory manager (IOS), 65-66

memory usage, configuration, 282-288

microkernel architecture, hypervisors, 74

microkernels, 63

migrate command (virsh), 90

modes of operation (KVM), 83

monolithic architecture, hypervisors, 74

MPLS (Multiprotocol Label Switching), 14

- VPNs (virtual private networks), 54-55, 142

MPLS over GRE, 190

MR (MAP Resolver), 170

MTUs (maximum transmission units)

- L2, configuration, 280

multipoint L2VPNs, 141

Multiprotocol Label Switching (MPLS). *See* MPLS (Multiprotocol Label Switching)

multitenancy, 21, 185

multitenant data center, 16-18

- CSR, 185-190
 - zone connectivity*, 188
- logical diagram, 185
- virtual service block design, 186

N

NAS (network-attached storage), 3

native drivers versus legacy drivers, 79

NBAR2 (Network Based Application Recognition), 52

NEDs, installation verification, 260-261

NETCONF, 166-168

Netmap I/O

- CSR 1000V data plane, 104-105
- versus Linux, 105

network-attached storage (NAS), 3

Network Based Application Recognition (NBAR2), 52

network-based storage virtualization, 11

Network Configuration Protocol (NetConf), 32

network connectivity, CSR 1000V

- cloudburst, 190

network data centers, virtualization, 15

network extension from premises

- deployment, CSR 1000V routers, 51

network fabric, data centers, 3

Network Functions Virtualization (NFV), 15, 33, 156, 223

network-level hypervisor, virtualization, 15

network-to-network connectivity, LISP, 175-176

network-to-network interconnection

- configuration, LISP, 176-182

network virtualization, 12-13

- device-based, 14-15
- evolution, 13-14
- protocol-based, 14

networking

- NFV (network function virtualization), 33
- OpenStack, 34
- SDN (software-defined networking), 30-31
 - ONF (Open Networking Foundation)*, 31-32
 - OpenDaylight*, 32-33

networks

- orchestrating solutions, 247
- VPNs (virtual private networks), 141
 - DMVPNs (Dynamic Multipoint VPNs)*, 144-145
 - GET VPNs (Group Encrypted Transport VPN)*, 145-147
 - IPsec*, 142
 - IPsec with crypto maps*, 143-144
 - L2VPNs*, 141
 - L3VPNs*, 142-148
 - remote access into cloud*, 153-155
 - service gateway*, 148-153
 - site-to-site*, 143
 - SSL VPNs (Secure Sockets Layer VPN)*, 147-148

Neutron component (OpenStack), 34
Neutron plugin, instantiating CSR to OpenStack, 242-245
Neutron project, OpenStack, 202-203
Neutron routers, CSR 1000V, 206-209
NFV (Network Functions Virtualization), 15, 156, 223
 NSO orchestration with OpenStack, 252-253, 260-266
NFV MANO, 268
NFV Orchestrator, 268
Nova component (OpenStack), 34
Nova project, OpenStack, 198-199
NSO tool, 249-251
 initiation in service and device model framework, 253, 260
 installed NED verification, 260-261
 NFV orchestration with OpenStack, 252-253, 260-261, 264-266
 VNF (virtual network function)
 descriptor definition, 261, 264
 instantiation, 264-265
NVF (network function virtualization), 33

O

on-demand service, cloud, 21
ONF (Open Networking Foundation), 31-32
OpenDaylight, 32-33
OpenFlow protocol, 31-32
OpenStack, 34, 196, 225
 Create Network submenu, 237-238
 Create Router dialog, 238
 CSR, instantiating Neutron plugin to, 242-245
 dashboard login, 235
 deploying, 225-233
 CSR tenant, 235-242
 network subnet menu, 238
 private cloud deployment, CSR 1000V, 195-211
 projects, 197
 Ceilometer, 205
 Cinder, 204
 Glance, 201-202
 Heat, 205
 Horizon, 205

Keystone, 199-201
 Neutron, 202-203
 Nova, 198-199
 Sabara, 205
 Swift, 205
 Trove, 205
 tenants, 265-266
 VM image creation, 235
operating systems, 59
 boot process, 66-67
 design, 60
 physical resource management, 60-62
 software access to resources, 62
IOS XE, 95-96
 versus IOS, 96-98
 kernels, 63
 hybrid, 64
 IOS, 64-66
 microkernels, 63
 Linux, memory management, 69-71
 shared resource access, 72
 versus hypervisors, 72-73
 virtualization, 8
orchestration, 267
 network solutions, 247
 NFV MANO, 268
 PNSC (Prime Network Services Controller), 269-270
 versus automation, 247
 VMS (Virtual Managed Services), 267-268
OTV (Overlay Transport Virtualization), 190
 CSR 1000V, 55
overcommitment, memory, Linux, 69-70
Overlay Transport Virtualization (OTV). See OTV (Overlay Transport Virtualization)
overlays, SDN, 224

P

PaaS (Platform as a Service), 2, 18-19
packet flow, CSR 1000V data plane, 103
 device initialization, 106-107
 Netmap I/O, 104-105
 Rx, 108
 Tx, 107
 unicast traffic, 109

- packet loss, debugging, 276
 - CPU usage, 281-282
 - ESXi, 289-292
 - hardware and software configurations, 278-279
 - high-level packet flow, 276-277
 - interface-to-NIC mapping, 281
 - L2 MTU, 280
 - memory usage, 282-288
 - throughput license, 277-278
 - vSwitch packet drops, 289
- packets, LISP, 172
- Packstack, deploying OpenStack, 225-233
- paged memory, 61
- para-virtualization, server, 8
- partitioning, core, hypervisors, 75
- pay-as-you-use service, 21
- PCI passthrough mode, hypervisor software, 274
- Performance Routing (PfR), 55
- performance tuning, CSR 1000V, 137-139
- PETR (proxy egress tunnel router), 170-171
- PfR (Performance Routing), 55
- physical infrastructure, network virtualization, 15
- physical memory, 76
- physical resource management, operating systems, 60-62
- pinning, CPUs, 138
- PITR (proxy ingress tunnel router), 170
- Platform as a Service (PaaS). *See* PaaS (Platform as a Service)
- Platform Manager (IOSd), 41
- PNSC (Prime Network Services Controller), 248
 - orchestration, 269-270
- point-to-point L2VPNs, 141
- policy IPsec VPNs, 143
- pool manager (memory manager), 65
- Power setting, BIOS, 276
- preemption scheduling algorithm, 61
- Prime Network Services Controller (PNSC). *See* PNSC (Prime Network Services Controller)
- priority scheduling algorithm, 61
- private cloud, 20
- private cloud deployment in OpenStack, CSR 1000V, 195-211
- processes
 - ESP, 98
 - RP, 97
 - SIP/SPA, 98
 - VMkernel, 77-78
- programmable fabric, SDN, 224
- projects, OpenStack, 197
 - Ceilometer, 205
 - Cinder, 204
 - Glance, 201-202
 - Heat, 205
 - Horizon, 205
 - Keystone, 199-201
 - Neutron, 202-203
 - Nova, 198-199
 - Sahara, 205
 - Swift, 205
 - Trove, 205
- protocol-based network protocol, 14
- protocols
 - LISP (Location/ID Separation Protocol), 54, 168-169, 175
 - control plane*, 171-175
 - data plane*, 171
 - ETR (egress tunnel router)*, 169-170
 - IP mobility*, 175
 - IPv6 migration*, 175
 - ITR (ingress tunnel router)*, 169-170
 - MR (MAP Resolver)*, 170
 - network-to-network connectivity*, 175-176
 - network-to-network interconnection configuration*, 176, 179-182
 - PETR (proxy egress tunnel router)*, 170-171
 - PITR (proxy ingress tunnel router)*, 170
 - OpenFlow, 31-32
- provisioning, automation, 247
- public cloud, 20
- public cloud deployment
 - CSR, Amazon web services, 211-215
 - CSR 1000V, 211
 - Amazon web services*, 216-222

Q

QEMU (Quick Emulator)

- architectural components, 84-85
- guest emulator, 85-87
- KVM architecture, 87

QFP (QuantumFlow Processor), 43

- debugging options, 286-288

QFP uCode (packet processing), CSR data plane, 102

quit command (virsh), 90

R

Radio Aware Routing, CSR 1000V, 56

rapid deployment, CSR 1000V routers, 49

ready queue (IOS scheduler), 65

reboot command (virsh), 90

redirection access model, CSR 1000V
cloudburst, 192-193

Redundancy Group Infrastructure, CSR
1000V, 56

region manager (memory manager), 65

regions, VPC, 214

resource pooling, 21

REST API, 202

restore command (virsh), 90

resume command (virsh), 90

ring manipulation, Netmap I/O, 105

RLOC (Routing Locator), 170

round-robin scheduling algorithm, 60

route reflector (RR), 176

routers. *See also* CSR (cloud service router)

ASR 1000, 41

*ESPs (embedded service processors),
42-43*

RP (route processor), 42

SIP (SPA interface processor), 43

CSR (cloud service router)

BPG route reflector, 155-162

host configuration, 275-276

hypervisors, 59

*LISP (Locator/ID Separation
Protocol), 168-182*

OpenStack, 235-242

*public cloud deployment, Amazon
web services, 211-215*

*as Remote Access VPN server,
153-155*

*remote VPN access into Cloud,
153-155*

secure inter-cloud connectivity, 152

CSR 1000V, 44-45

deployment requirements, 45-47

*elastic performance and scaling,
47-48*

*network extension from premises to
cloud, 51*

rapid deployment, 49

routing flexibility, 49

secure cloud VPN gateway, 50-51

segmentation within cloud, 52

ETR (Egress Tunnel Router), 169-170

ITR (Ingress Tunnel Router), 169-170

Neutron, CSR 1000V, 206-209

PETR, 170-171

PITR, 170

software, 16

tenant, CSR 1000V, 209-211

routing

flexibility, 49

PfR (Performance Routing), 55

Radio Aware Routing, 56

tables, 213

RP (route processor)

ASR (Aggregation Service Router), 42

processes, 97

Rx packet flow, CSR 1000V data
plane, 108

S

S3 (Simple Storage Service), Amazon, 212

SaaS (Software as a Service), 2, 18-20

Sahara project, OpenStack, 205

SAN (storage-area network), 3

save command (virsh), 90

scaling CSR 1000V routers, 47-48

scheduler (IOS), 37, 65

scheduling algorithms, 60-62

SDN (software-defined networking), 1,
30-31, 223

abstract layer creation, 223

application-centric infrastructure, 224

data center, 224-225

- framework, 223
- ONF (Open Networking Foundation), 31-32
- OpenDaylight, 32-33
- overlays, 224
- programmable fabric, 224
- secure cloud VPN gateway deployment, CSR 1000V routers, 50-51
- secure inter-cloud connectivity, CSR, 152
- Secure Sockets Layer (SSL), 3
 - VPNs (SSL VPNs), 147-148
- security groups, VPC, 213
- segmentation within cloud, CSR 1000V routers, 52
- segmented memory allocation, 61
- server virtualization, 6-7
 - full, 8
 - OS (operating system), 8
 - para-, 8
 - storage, 10
- service blocks, data centers
 - deployment, 188
 - placement, 186
- service virtualization, 15-16
- services
 - data centers, 3
 - IaaS (Infrastructure as a Service), 18-19
 - PaaS (Platform as a Service), 19
 - SaaS (Software as a Service), 20
- Setmaxmem command (virsh), 90
- Setmem command (virsh), 90
- Setvcpus command (virsh), 90
- shared resources, access, operating systems, 72
- Shortest Process Next (SPN) scheduling algorithm, 60
- Shortest Remaining Time (SRT) algorithm, 61
- show commands, 181-182
- show interface command, 278-279
- show interface controller command, 283
- show interfaces command, 282
- shutdown command (virsh), 90
- Simple Storage Service (S3), Amazon, 212
- Single Root I/O Virtualization (SR-IOV), 274-275
- SIP (SPA interface processor), ASR (Aggregation Service Router), 43
- SIP/SPA processes, 98
- site-to-site VPNs, 143
- SNIA (Storage Networking Industry Association), 10
- sockets settings, hosts, 275
- software
 - access to physical resources, operating systems, 62
 - speed configuration, 278-279
- Software as a Service (SaaS). *See* SaaS (Software as a Service)
- software crypto engine, CSR 1000V, 103
- software-defined networking (SDN). *See* SDN (software-defined networking)
- software router, 16
- software VPN access, 27
- SPA interface processor (SIP), 43
- speed command, 278
- SpeedStep setting, BIOS, 276
- SPN (Shortest Process Next) scheduling algorithm, 60
- SR-IOV (Single Root I/O Virtualization), 274-275
- SRT (Shortest Remaining Time) algorithm, 61
- SSL (Secure Sockets Layer), 3
 - VPNs (Secure Sockets Layer VPNs), 147-148
- start command (virsh), 89
- static routing VPNs, 143
- statistics drop command, 285
- storage, 3
- storage-area network (SAN), 3
- Storage Networking Industry Association (SNIA), 10
- storage virtualization, 9-11
 - device-based, 11
 - hidden complexity, 12
 - host-based, 11
 - network-based, 11
 - performance, 12
 - thin provisioning, 12
- subnets, VPC, 212
- supervisors, 72
- suspend command (virsh), 90
- swap caches, 71
- swap files, 71

swapping memory, Linux, 69-71
 Swift component (OpenStack), 34
 Swift project, OpenStack, 205
 switches, vSwitch, 272-273
 packet drops, 289
 synchronization, Netmap, 105
 system design, CSR 1000V, 95-98

T

Tail-f framework, 166
 Tail-f tool, 249, 251
 NFV orchestration with OpenStack,
 252-253, 260-261, 264-266
 TailDrop drop type (IOS), 286
 tenant deployment, OpenStack, CSR,
 235-240, 242
 tenant routers
 CSR 1000V, 209-211
 tenants, 201
 OpenStack, 265-266
 thin client mode (SSL VPNs), 147
 thrashing, 71
 throughput, appropriate license, 277-278
 Tiny Code Generator (TCG), 87
 traffic, hair pinning, 27
 troubleshooting, CSR 1000V
 architecture overview, 271-272
 I/O configuration, 272-276
 Trove project, OpenStack, 205
 tunnel mode (SSL VPNs), 147
 Tx packet flow, CSR 1000V data
 plane, 107

U

uCode, CSR data plane, 102
 UnconfiguredIpv4Fia drop type (IOS), 286
 UnconfiguredIpv6Fia drop type (IOS), 286
 undefine command (virsh), 90
 unicast traffic packet flow, CSR 1000V
 data plane, 109
 UNIVAC-I mainframe computer, 1
 unstructured data, 4
 user tools, Libvit, 89-91
 utilization, data centers, 2

V

vCenter, 81
 Vcpuinfo command (virsh), 90
 Vcpupin command (virsh), 90
 VIM (Virtualized Infrastructure
 Manager), 268
 virsh command-line tool, 89-91
 Virtual Extensible LAN (VxLAN), 56, 190
 Virtual Machine File System (VMFS),
 79-80
 Virtual Machine Manager, 71, 127,
 130-134. *See also* hypervisors
 virtual machines, KVM (Kernel-based
 Virtual Machine), 82-83
 architectural components, 84-85
 guest emulator, 85-87
 Virtual Managed Services (VMS). *See* VMS
 (Virtual Managed Services)
 Virtual Private Cloud (VPC), Amazon, 211
 Virtual Private LAN Services (VPLS), 55
 virtual private networks (VPNs). *See* VPNs
 (virtual private networks)
 virtual routing and forwarding (VRF), 14
 virtual service block design, multitenant
 data centers, 186
 virtual switches, vSwitch, 272-273
 virtualization, 71
 ASR 1001 into CSR 1000V, 98
 branch, 164-168
 data centers, 2-4
 conceptual architecture, 5-6
 evolution, 5
 mainframes, 17
 network, 12-13
 device-based, 14
 evolution, 13-14
 protocol-based, 14
 NFV (Network Functions Virtualization),
 33, 156, 223
 OTV (Overlay Transport Virtualization),
 CSR 1000V, 55
 server, 6-8
 storage, 11
 service, 15-16
 SR-IOV (Single Root I/O Virtualization),
 274-275

- storage, 9-10
 - device-based*, 11
 - hidden complexity*, 12
 - host-based*, 11
 - network-based*, 11
 - thin provisioning*, 12
- Virtualized Infrastructure Manager (VIM)**, 268
- VMs (virtual machines)**
 - bringing up on ESXi with CSR 1000V, 110-125
 - layers, CSR, 103
- VMFS (virtual machine file system)**, 79-80
- VMkernel, ESXi hypervisor**, 75-76
 - CIM processes, 77-78
 - CPU scheduler, 76
 - DCUI processes, 77
 - memory management, 76
 - VMM processes, 77
 - VMX processes, 77
- VMM (virtual machine manager)**. *See* hypervisors
- VMS (Virtual Managed Services)**, 248
 - orchestration, 267-268
- VMware hypervisor, installing CSR 1000V on**, 110-125
- VMX processes, VMkernel**, 77
- VNF**
 - descriptor definition, 261, 264
 - instantiation, 264-265
 - Manager, 268
- vPATH function**, 270
- VPC (Virtual Private Cloud)**, 212
 - Amazon, 211
 - availability zones, 214
 - Direct Connect, 214
 - elastic IP, 213
 - Internet gateway, 213
 - regions, 214
 - routing tables, 213
 - security groups, 213
 - subnets, 212
- VPLS (Virtual Private LAN Services)**, CSR 1000V, 55
- VPNs (virtual private machines)**, 141
 - DMVPNs (Dynamic Multipoint VPNs), 53, 144-145
 - GET VPNs (Group Encrypted Transport VPN), 145-147
 - IPsec, 142
 - IPsec with crypto maps, 143-144
 - L2VPNs, 141
 - L3VPNs, 142-143
 - DMVPNs, 144-145
 - GET VPNs, 145-147
 - GRE tunnels, 142
 - IPsec VPNs, 142-144
 - MPLS VPNs, 142
 - site-to-site VPNs, 143
 - SSL VPNs, 147-148
 - MPLS (Multiprotocol Label Switching), 54
 - remote access into cloud, 153-155
 - service gateway, 148-153
 - site-to-site, 143
 - SSL VPNs (Secure Sockets Layer VPN), 147-148
- VRF (virtual routing and forwarding)**, 14
- vSwitch**, 272-273
 - packet drops, troubleshooting, 289
- VxLAN (Virtual Extensible LAN)**, 190
 - CSR 1000V, 56

W-X

Windows Server Virtualization, 91
 workload migration, CSR 1000V
 cloudburst, 191

XE architecture, IOS, 39
 kernel, 40

Xen hypervisor, 92-93
 XMPP (Extensible Messaging and Presence Protocol), 32

Y-Z

YANG, 166-168

ZBFW (Zone Based Firewall), CSR 1000V,
 56-57

zero-copy model, Netmap, 105
 Zone Based Firewall (ZBFW), 56
 zones, connectivity, multitenant data
 center with CSR 1000V, 188