JUNIPER NETWORKS | Engineering Simplicity

vJunos-switch Deployment Guide for KVM

Published
2023-07-21

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

vJunos-switch Deployment Guide for KVM
Copyright © 2023 Juniper Networks, Inc. All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide | v

1 Understand vJunos-switch

vJunos-switch Overview | 2

Overview | 2

Key Features Supported | 3

Benefits and Uses | 3

Limitations | 4

vJunos-switch Architecture | 4

2 Hardware and Software Requirements vJunos-switch on KVM

Minimum Hardware and Software Requirements | 8

3 Install and Deploy vJunos-switch on KVM

Install vJunos-switch on KVM | 11

Prepare the Linux Host Servers to Install vJunos-switch | 11

Deploy and Manage vJunos-switch on KVM | 11

Set Up the vJunos-switch Deployment on the Host Server | 12

Verify the vJunos-switch VM | 17

Configure vJunos-switch on KVM | 19

Connect to vJunos-switch | 19

Configure Active Ports | 20

Interface Naming | 20

Configure the Media MTU | 21

4 Troubleshoot

Troubleshoot vJunos-switch | 23

Verify That the VM is Running | 23

Verify CPU Information | **24**

View Log Files | **25**

Collect Core Dumps | **25**

About This Guide

Use this guide to install the virtual Junos-switch (vJunos-switch).

The vJunos-switch is a virtual version of the Junos-based EX switching platform. It represents a Juniper switch running Junos® operating system (Junos OS) in the kernel-based virtual machine (KVM) environment. The vJunos-switch is based on Juniper Networks® vMX Virtual Router (vMX) nested architecture.

This guide also includes basic vJunos-switch configuration and management procedures.

After installing and configuring the vJunos-switch as covered in this guide, refer to Junos OS documentation for information about additional software configuration.

RELATED DOCUMENTATION

- Junos OS for EX Series Documentation

1

CHAPTER

Understand vJunos-switch

vJunos-switch Overview | 2

vJunos-switch Architecture | 4

vJunos-switch Overview

SUMMARY

This topic provides an overivew, key features supported, benefits, and limitations of the vJunos-switch.

IN THIS SECTION

- Overview | 2
- Key Features Supported | 3
- Benefits and Uses | 3
- Limitations | 4

Overview

IN THIS SECTION

- vJunos-switch Installation Overview | 3

Read this topic for an overview of the vJunos-switch.

The vJunos-switch is a virtual version of a Juniper switch that runs the Junos OS. You can install a vJunos-switch as a virtual machine (VM) on an x86 server.

You can configure and manage the vJunos-switch in the same way as you manage a physical switch.

The vJunos-switch is a single virtual machine (VM) that you can use only in labs and not in the production environment. The vJunos-switch is built using EX9214 as a reference Juniper switch and supports a single Routing Engine and single Flexible PIC Concentrator (FPC).

The vJunos-switch supports a bandwidth of up to 100 Mbps aggregated over all the interfaces. You don't need to purchase a bandwidth license for using the vJunos-switch.

Instead of using hardware switches, you can use the vJunos-switch to start the Junos software for testing the network configurations and protocols.

vJunos-switch Installation Overview

You can install the software components of the vJunos-switch on an industry-standard x86 server running a Linux KVM hypervisor (Ubuntu 18.04, 20.04, 22.04 or Debian 11 Bullseye).

On servers running the KVM hypervisor, you can also run applicable third-party software. You can install multiple vJunos-switch instances on a single server.

Key Features Supported

This topic provides you the list and details of the key features that are supported and validated on vJunos-switch. For details on configuration of these features see the feature guides at: User Guides.

The vJunos-switch supports the following key features:

- Supports up to 96 switch interfaces
- Can simulate data center IP underlay and overlay topologies.
- Supports EVPN-VXLAN leaf functionality
- Supports Edge-Routed Bridging (ERB)
- Supports EVPN LAG multihoming in EVPN-VXLAN (ESI-LAG)

Benefits and Uses

The benefits and use cases of the vJunos-switch on standard x86 servers are as follows:

- **Reduced capital expenditure (CapEx) on lab—**The vJunos-switch is available for free to build test labs reducing costs associated with physical switches.
- **Reduced deployment time—**You can use the vJunos-switch to build and to test topologies virtually without building expensive physical labs. Virtual labs can be built instantly. As a result, you can reduce costs and delays associated with deployments on the physical hardware.
- **Eliminate need and time for lab hardware—**The vJunos-switch helps you eliminate waiting time for lab hardware to arrive after procurement. vJunos-switch is available for free and can be downloaded instantly.
- **Education and training—**Allows you to build labs for learning and education services for your employees.

- **Proof of concept and validation testing**—You can validate various data center switching topologies, pre-build configurations examples, and get automation ready.

Limitations

The vJunos-switch has the following limitations:

- Has a single Routing Engine and single FPC architecture.
- Does not support in-service software upgrade (ISSU).
- Does not support attachment or detachment of interfaces when it is running.
- SR-IOV for the vJunos-switch use cases and throughput is not supported.
- Due to its nested architecture, the vJunos-switch cannot be used in any deployments that launch the instances from within a VM.
- Supports a maximum bandwidth of 100 Mbps over all the interfaces.

  NOTE: Bandwidth licenses are not provided as there is no need for a bandwidth license. License check message might come up. Ignore the license check messages.

- You cannot upgrade the Junos OS on a running system. Instead, you must deploy a new instance with the new software.
- Multicast is not supported.

RELATED DOCUMENTATION

Minimum Hardware and Software Requirements | 8

vJunos-switch Architecture

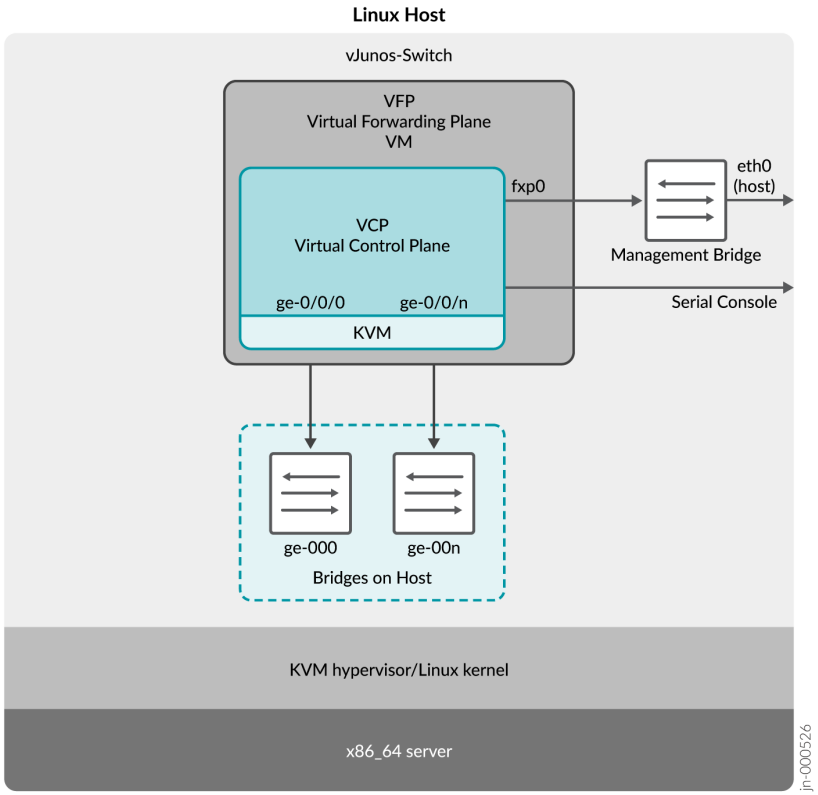
The vJunos-switch is a single, nested VM solution in which the virtual forwarding plane (VFP) and the Packet Forwarding Engine (PFE) reside in the outer VM. When you start the vJunos-switch, the VFP

starts a nested VM that runs the Junos Virtual Control Plane (VCP) image. KVM hypervisor is used to deploy VCP.

The term “nested” refers to the VCP VM being nested within the VFP VM, as shown in Figure 1 on page 5.

The vJunos-switch can support up to 100 Mbps of throughput using 4 cores and 5GB of memory. Any additional cores and memory configured gets allocated to the VCP. VFP does not need additional memory apart from the minimum footprint supported. The 4 cores and 5GB memory is sufficient for lab use cases.

Figure 1: vJunos-switch Architecture



The vJunos-switch architecture is organized in layers:

- The vJunos-switch is at the top layer.
- The KVM hypervisor and the related system software described in the software requirements section are in the middle layer.
- The x86 server is in the physical layer at the bottom.

Understanding this architecture can help you plan your vJunos-switch configuration.

After you create the vJunos-Switch instance, you can use the Junos OS CLI to configure the vJunos-switch interfaces in the VCP. The vJunos-switch supports Gigabit Ethernet interfaces only.

2

CHAPTER

Hardware and Software Requirements vJunos-switch on KVM

Minimum Hardware and Software Requirements | 8

Minimum Hardware and Software Requirements

This topic provides you the list of hardware and software requirements to start a vJunos-switch instance.

Table 1 on page 8 lists the hardware requirements for vJunos-switch.

Table 1: Minimum Hardware Requirements for vJunos-switch

Description	Value
Sample system configuration	For lab simulation and low performance (less than 100 Mbps) use cases, any Intel x86 processor with VT-x capability. Intel Ivy Bridge processors or later. Example of Ivy Bridge processor: Intel Xeon E5-2667 v2 @ 3.30 GHz 25 MB cache
Number of cores	A minimum of four cores are required. The software allocates three cores to the VFP and one core to the VCP, which is sufficient for most use cases. Any additional cores will be provided to VCP as three cores is sufficient to support the data plane needs of VFP.
Memory	A minimum of 5GB memory is required. Approximately 3GB memory will be allocated to VFP and 2 GB to the VCP. If more than 6 GB of total memory is provided, then VFP memory is capped at 4GB, and the additional memory is allocated to VCP.
Other requirements	• Intel VT-x capability. • Hyperthreading (recommended) • AES-NI

Table 2 on page 9 lists the software requirements for vJunos-switch.

Table 2: Software Requirements for Ubuntu

Description	Value
Operating system **NOTE**: Only English localization is supported.	• Ubuntu 22.04 LTS • Ubuntu 20.04 LTS • Ubuntu 18.04 LTS • Debian 11 Bullseye
Virtualization	• QEMU-KVM The default version for each Ubuntu or Debian version is sufficient. The apt-get install qemu-kvm installs this default version.
Required packages **NOTE**: Use the `apt-get install` *`pkg name`* or `sudo apt-get install <pkg-name>` commands to install a package.	• qemu-kvm virt-manager • libvirt-daemon-system • virtinst libvirt-clients bridge-utils
Supported Deployment Environments	QEMU-KVM using libvirt Also, the EVE-NG bare metal deployment is supported. Note: vJunos-switch is not supported on EVE-NG or any other deployments that launch vJunos from within a VM due to the constraints of deeply nested virtualization.
vJunos-switch Images	The images can be accessed from the lab download area of juniper.net at: Test Drive Juniper

CHAPTER 3

Install and Deploy vJunos-switch on KVM

Install vJunos-switch on KVM | 11

Deploy and Manage vJunos-switch on KVM | 11

Configure vJunos-switch on KVM | 19

Install vJunos-switch on KVM

SUMMARY

Read this topic to understand how to install the vJunos-switch in the KVM environment.

IN THIS SECTION

- Prepare the Linux Host Servers to Install vJunos-switch | 11

Prepare the Linux Host Servers to Install vJunos-switch

This section applies to both Ubuntu and Debian host servers.

1. Install the standard package versions for your Ubuntu or Debian host server to ensure that the servers meet the minimum hardware and software requirements.
2. Verify that Intel VT-x technology is enabled. Run the `lscpu` command on your host server.

   The **Virtualization** field in the output of the `lscpu` command will display **VT-x**, if VT-x is enabled. If VT-x is not enabled, then see your server documentation to learn how to enable it in BIOS.

Deploy and Manage vJunos-switch on KVM

SUMMARY

Read this topic to understand how to deploy and manage the vJunos-switch instance after you install it.

IN THIS SECTION

- Set Up the vJunos-switch Deployment on the Host Server | 12
- Verify the vJunos-switch VM | 17

This topic describes:

- How to bring up the vJunos-switch on the KVM servers using libvirt.
- How to choose the amount of CPU and memory, set up the required bridges for connectivity, and configure the serial port.

- How to use relevant XML file sections for the configurations and selections listed earlier.

> **NOTE**: Download the sample XML file and the vJunos-switch image from the Juniper website.

Set Up the vJunos-switch Deployment on the Host Server

This topic describes how to set up the vJunos-switch deployment on the host server.

> **NOTE**: This topic highlights only a few sections of the XML file that are used to deploy vJunos-switch through libvirt.
>
> The entire XML file **vjunos.xml** is available for download along with the VM image and associated documentation on the vJunos Lab Software Downloads page.

Install the packages mentioned in the Minimum Software requirements section, if the packages are not already installed. See "Minimum Hardware and Software Requirements" on page 8

1. Create a Linux bridge for each Gigabit Ethernet interface of the vJunos-switch that you plan to use.

   ```
   # ip link add ge-000 type bridge

   # ip link add ge-001 type bridge
   ```

   In this case, the instance will have ge-0/0/0 and ge-0/0/1 configured.

2. Bring up each Linux Bridge.

   ```
   ip link set ge-000 up

   ip link set ge-001 up
   ```

3. Make a live disk copy of the provided QCOW2 vJunos image.

   ```
   # cd /root

   # cp vjunos-switch-23.1R1.8.qcow2 vjunos-sw1-live.qcow2
   ```

   Make a distinct copy for each vJunos that you plan to deploy. This ensures that you do not make any permanent changes on the original image. The live image must also be writable by the userid deploying vJunos-switch—typically the root user.

4. Specify the number of cores provided to vJunos by modifying the following stanza.

The following stanza specifies the number of cores provided to vJunos. The minimum needed cores are 4 and are sufficient for lab use cases.

```
<cpu>
    <arch>x86_64</arch>
    <model>IvyBridge</model>
    <topology cores="4" sockets="1" threads="1"/>
    <model fallback="allow">qemu64</model>
    <feature name="vmx" policy="require"/>
</cpu>
```

The default number of cores needed is 4 and is sufficient for most applications. This is the minimum CPU supported for vJunos-switch. You can leave the CPU model as IvyBridge. Later generation Intel CPUs will also work with this setting.

5. Increase the memory if needed by modifying the following stanza.

```
<?xml version="1.0"?>
<domain xmlns:ns0="http://libvirt.org/schemas/domain/qemu/1.0" type="kvm">
    <name>vjunos-sw1</name>
    <memory unit="KiB">5242880</memory>
    <currentMemory unit="KiB">5242880</currentMemory>
    <vcpu placement="static">4</vcpu>
```

The following example shows the default memory required by the vJunos-switch. The default memory is sufficient for most applications. You can increase the value if needed. It also shows the name of the specific vJunos-switch being spawned, which is **vjunos-sw1** in this case.

6. Specify the name and location of your vJunos-switch image by modifying the XML file as shown in the following example.

```
<disk device="disk" type="file">
    <driver cache="writeback" name="qemu" type="qcow2"/>
    <source file="/root/vjunos-sw1-live.qcow2"/>
    <target dev="vda" bus="virtio"/>
</disk>
```

You must provide each vJunos VM on the host with its own uniquely named QCOW2 image. This is required for libvirt and QEMU-KVM.

7. Create the disk image.

```
# ./make-config.sh <juniper.conf> <config.qcow2>
```

The vJunos-switch accepts an initial configuration by connecting a second disk to the VM instance that contains the configuration. Use the provided script *make-config.sh* to create the disk image.

The XML file references this configuration drive as shown below:

```
<disk device="disk" type="file">
    <driver cache="writeback" name="qemu" type="qcow2"/>
    <source file="/root/config.qcow2"/>
    <target dev="vdb" bus=”virtio”/>
</disk>
```

NOTE: If you do not prefer initial configuration, then remove the above stanza from the XML file.

8. Set up the management Ethernet port.

```
<interface type='direct'>
    <source dev='eth0' mode='bridge'/>
    <model type='virtio'/>
    <alias name='net0'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
```

This example allows you to connect to the VCP “fxp0” that is the management port from outside the host server on which vJunos-switch resides.

You need to have a routable IP address configured for fxp0, either through a DHCP server or using standard CLI configuration.

The “eth0” in the stanza below refers to the host server interface which provides connectivity to the external world and should match the name of this interface on your host server.

If you are not using Dynamic Host Configuration Protocol (DHCP), then, after the vJunos-switch is up and running, telnet to its console and configure the IP address for “fxp0” using CLI configuration as shown below:

NOTE: The configurations below are just examples or sample configuration snippets. You might also have to set up a static route configuration.

```
# set interfaces fxp0 unit 0 family inet address 10.92.249.111/23

# set routing-options static route 0.0.0.0/0 next-hop 10.92.249.254
```

9. Enable SSH to the VCP management port.

   `# set system services ssh root-login allow` command.

10. Create a Linux bridge for each port that you specify in the XML file.

```
<interface type="bridge">
    <source bridge="ge-000"/>
    <model type="virtio"/>
    <mtu size='9600'/>
    <alias name="net1"/>
    <address bus="0x00" domain="0x0000" function="0x0" slot="0x08" type="pci"/>
</interface>
<interface type="bridge">
    <source bridge="ge-001"/>
    <model type="virtio"/>
    <mtu size='9600'/>
    <alias name="net2"/>
    <address bus="0x00" domain="0x0000" function="0x0" slot="0x09" type="pci"/>
</interface>
```

The port names are specified in the following stanza. The convention for the vJunos-switch is to use ge-0xy where "xy" specifies the actual port number. In the following example, ge-000 and ge-001 are the port numbers. These port numbers will map to the Junos ge-0/0/0 and ge-0/0/1 interfaces respectively. As mentioned earlier, you need to create a Linux bridge for each port that you specify in the XML file.

11. Provide a unique serial console port number for each vJunos-switch on your host server.

    In the following example, the unique serial console port number is "8610".

```
<serial type="tcp">
    <source host="127.0.0.1" mode="bind" service="8610"/>
    <protocol type="telnet"/>
    <target port="0"/>
```

```
    <alias name="serial0"/>
</serial>
```

Do not modify the following smbios stanza. It tells vJunos that it is a vJunos-switch.

```
<ns0:commandline>
    <ns0:arg value="-smbios"/>
    <ns0:arg value="type=1,product=VM-VEX"/>
</ns0:commandline>
```

12. Create vJunos-sw1 VM using the **vJunos-sw1.xml** file.

    ```
    # virsh create vjunos-sw1.xml
    ```

    The term "sw1" is used to indicate that this is the first vJunos-switch VM that is being installed. Subsequent VMs can be named vjunos-sw2, and vjunos-sw3 and so on.

    As result, the VM is created and the following message is displayed:

    Domain vjunos-sw1 created from vjunos-sw1.xml

13. Check */etc/libvirt/qemu.conf* and uncomment the following XML lines if these lines were commented out.

    Some examples of valid values are given below. Uncomment the specified lines.

    ```
    #       user = "qemu"   # A user named "qemu"
    #       user = "+0"     # Super user (uid=0)
    #       user = "100"    # A user named "100" or a user with uid=100#user = "root"              <<<
    uncomment  this line
    #
    #group = "root" <<< uncomment  this line
    ```

14. Restart libvirtd and create the vJunos-switch VM again.

    ```
    # systemctl restart libvirtd
    ```

15. Shut down the vJunos-switch deployed on the Host Server safely (if needed). Use the `# virsh shutdown vjunos-sw1` command to shutdown vJunos-switch.

    When you execute this step, a shutdown signal sent to the vJunos-switch instance allows it to shutdown gracefully.

    The following message is displayed.

    Domain 'vjunos-sw1' is being shutdown

NOTE: Do not use the “virsh destroy” command as this command can corrupt the vJunos-switch VM disk.

If your VM stops booting after using the “virsh destroy” command, then, create a live QCOW2 disk copy of the provided original QCOW2 image.

Verify the vJunos-switch VM

This topic describes how to verify whether the vJunos-switch is up and running.

1. Verify if the vJunos-switch is up and running.

```
# virsh list
```

```
# virsh list
 Id   Name         State
----------------------------
 74   vjunos-sw1   running
```

2. Connect to the serial console of the VCP.

   You can find the port to connect to the serial console of the VCP from the XML file. Also, you can login to the serial console of the VCP through the “telnet localhost *<portnum>*” where portnum is specified in the XML configuration file:

   NOTE: The telnet port number needs to be unique for each vJunos-switch VM residing on the host server.

```
# telnet localhost 8610
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
root@:~ #
```

3. Disable auto image upgrade.

If you have not supplied any initial Junos configuration in the steps above, then the vJunos-switch will, by default, attempt to DHCP for the initial network setup.

If you don't have a DHCP server that can supply the Junos configuration, you can get repeated messages as shown below:

"Auto Image Upgrade"

You can disable these messages as follows:

```
[edit]]
user@host# set system  root-authentication plain-text-password
New password: <type password here>
Retype new password: <retype password here>
root# delete chassis auto-image-upgrade
[edit]
root# commit
commit complete
```

4. Verify if the ge interfaces specified in your vJunos-switch xml file are up and available. Use the `show interfaces terse` command.
   For example, if the vJunos-switch XML definition file specifies two virtual NICs connected to "ge-000" and "ge-001", then ge-0/0/0 and ge-0/0/1 interfaces should be in the link "up" state when you verify using the `show interface output` command as shown below.

```
root> show interfaces terse
Interface               Admin Link Proto    Local                 Remote
ge-0/0/0                up    up
ge-0/0/0.16386          up    up
lc-0/0/0                up    up
lc-0/0/0.32769          up    up   vpls
pfe-0/0/0               up    up
pfe-0/0/0.16383         up    up   inet
                                   inet6
pfh-0/0/0               up    up
pfh-0/0/0.16383         up    up   inet
pfh-0/0/0.16384         up    up   inet
ge-0/0/1                up    up
ge-0/0/1.16386          up    up
ge-0/0/2                up    down
ge-0/0/2.16386          up    down
```

```
ge-0/0/3                up     down
ge-0/0/3.16386          up     down
[snip]
```

5. Verify that a vnet inetrface under each corresponding "ge" bridge is configured. Use the `brctl` command on the host server, after you start the vJunos-switch as shown below:

```
# ip link add ge-000 type bridge
# ip link show ge-000
bridge name    bridge id        STP enabled    interfaces
ge-000         8000.fe54009a419a    no         vnet1
# ip link show ge-001
bridge name    bridge id        STP enabled    interfaces
ge-001         8000.fe5400e9f94f    no         vnet2
```

Configure vJunos-switch on KVM

SUMMARY

Read this topic to understand how to configure the vJunos-switch in the KVM environment.

IN THIS SECTION

- Connect to vJunos-switch | **19**
- Configure Active Ports | **20**
- Interface Naming | **20**
- Configure the Media MTU | **21**

Connect to vJunos-switch

Telnet to the serial console number specified in the XML file to connect to vJunos-switch. See details provided in "Deploy and Manage vJunos-switch on KVM" on page 11.

For example:

```
# telnet localhost 8610
```

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
root@:~ # cli
root>
```

You can also SSH to the vJunos-switch VCP.

Configure Active Ports

This section describes how to configure the number of active ports.

You can specify the number of active ports for the vJunos-switch to match the number of NICs added to the VFP VM. The default number of ports is 10, but you can specify any value in the range of 1 through 96.

Run the `user@host# set chassis fpc 0 pic 0 number-of-ports 96` command to specify the number of active ports. Configure the number of ports at the [edit chassis fpc 0 pic 0] hierarchy level.

Interface Naming

The vJunos-switch supports only Gigabit Ethernet (ge) interfaces.

You cannot change the interface names to 10-Gigabit Ethernet (xe) or 100-Gigabit Ethernet (et). If you attempt to change the interface names, then these interfaces will still show as “ge” when you run the `show configuration` or `show interfaces terse` commands.

Here is an example output of the “show configuration” CLI command when users attempt to change the interface name to “et”:

```
chassis {
    fpc 0 {
        pic 0 {
            ##
            ## Warning: statement ignored: unsupported platform (ex9214)
            ##
            interface-type et;
        }
```

```
    }
}
```

Configure the Media MTU

You can configure the media maximum transmission unit (MTU) in the range 256 through 9192. MTU values outside the above mentioned range are rejected.

You must configure the MTU by including the MTU statement at the [edit interface interface-name] hierarchy level.

Configure the MTU.

```
[edit]
user@host# set interface ge-0/0/0 mtu <mtu>
```

NOTE: The maximum supported MTU value is 9192 bytes.

For example:

```
[edit]
user@host# set interface ge-0/0/0 mtu 9192
```

4

CHAPTER

Troubleshoot

Troubleshoot vJunos-switch | 23

Troubleshoot vJunos-switch

SUMMARY

Use this topic to verify your vJunos-switch configuration and for any troubleshooting information.

IN THIS SECTION

- Verify That the VM is Running | 23
- Verify CPU Information | 24
- View Log Files | 25
- Collect Core Dumps | 25

Verify That the VM is Running

- Verify whether the vJunos-switch is running after you install it.

```
virsh list
```

The `virsh list` command displays the name and state of the virtual machine (VM). The state can be: running, idle, paused, shutdown, crashed, or dying.

```
# virsh list
 Id   Name        State
--------------------------
 72   vjunos-switch   running
```

- You can stop and start the VMs with the following `virsh` commands:
  - `virsh shutdown`—Shutdown the vJunos-switch.
  - `virsh start`—Start an inactive VM that you defined previously.

NOTE: Do not use the “virsh destroy” command as that can corrupt the vJunos-switch VM disk.

If your VM stops and does not boot after using the `virsh destroy` command, then create a live QCOW2 disk copy of the original QCOW2 image provided.

Verify CPU Information

Use the `lscpu` command on the host server to display CPU information.

The output displays information such as the total number of CPUs, the number of cores per socket, and the number of CPU sockets.

For example, the following codeblock shows the information for an Ubuntu 20.04 LTS host server supporting a total of 32 CPUs.

```
root@vjunos-host:~# lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   46 bits physical, 48 bits virtual
CPU(s):                          32
On-line CPU(s) list:             0-31
Thread(s) per core:              2
Core(s) per socket:              8
Socket(s):                       2
NUMA node(s):                    2
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           62
Model name:                      Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
Stepping:                        4
CPU MHz:                         2593.884
CPU max MHz:                     3400.0000
CPU min MHz:                     1200.0000
BogoMIPS:                        5187.52
Virtualization:                  VT-x
L1d cache:                       512 KiB
L1i cache:                       512 KiB
L2 cache:                        4 MiB
L3 cache:                        40 MiB
NUMA node0 CPU(s):               0-7,16-23
```

```
NUMA node1 CPU(s):               8-15,24-31
[snip]
```

View Log Files

View the system logs using the `show log` command on the vJunos-switch instance.

```
root > show log ?
```

The `root > show log ?` command displays the list of log files available for viewing.

For example, to view the chassis daemon (chassisd) logs run the `root > show log chassisd` command.

Collect Core Dumps

Use the `show system core-dumps` command to view the collected core file. You can transfer these core dumps to an external server for analysis through the fxp0 management interface on the vJunos-switch.