Engineering
Simplicity

vJunosEvolved Deployment Guide

Published
2025-11-11

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

YEAR 2000 NOTICE

END USER LICENSE AGREEMENT

Table of Contents

About This Guide

Use this guide to install the virtual JunosEvolved (vJunosEvolved) on KVM. This guide includes basic vJunosEvolved configuration and management procedures.

vJunosEvolved is a virtual version of the Junos OS Evolved-based PTX switching platform that represents a Juniper Networks® switch running Junos OS Evolved in kernel-based virtual machine (KVM) environment.

After installing and configuring the vJunosEvolved platform as covered in this guide, refer to Junos® operating system Evolved (Junos OS Evolved) documentation for information about additional software configurations.

RELATED DOCUMENTATION

Junos OS Evolved Documentation

1

PART

vJunosEvolved Deployment for KVM

CHAPTER 1

Overview

IN THIS CHAPTER

What is vJunosEvolved

SUMMARY

This topic provides an overivew, key features, benefits, and limitations of vJunosEvolved.

IN THIS SECTION

Overview

IN THIS SECTION

vJunosEvolved is a virtual version of a Juniper router that runs the Junos OS Evolved. You can install a vJunosEvolved as a virtual machine (VM) on an x86 server.

You can configure and manage the vJunosEvolved in the same way as you manage a physical router.

vJunosEvolved is a single virtual machine (VM) that you can use only in labs and not in the production environment. The vJunosEvolved is built using PTX10001-36MR as a reference Juniper router, which is a fixed-configuration packet transport router on the Junos® OS Evolved platform.

The vJunosEvolved Routing Engine and the vBT-COSIM (a virtual BT chip) that performs the packet processing run on the same VM.

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

The vJunosEvolved virtual platform primarily acts as a test platform for lab simulations for the customers.

NOTE: Bandwidth licenses are not required and are not provided. You can ignore any license check messages if you get an alert.

vJunosEvolved Installation Overview

You can install the software components of vJunosEvolved on an industry-standard x86 server running a Linux KVM hypervisor (Ubuntu 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 vJunosEvolved instances on a single server.

Key Features

This topic provides you a list of key features that the vJunosEvolved platform supports.

The vJunosEvolved platform supports the following key features:

- Supports up to 12 switch interfaces and 72 channelized interfaces.
- Can simulate data center IP underlay and overlay topologies.
- Supports EVPN-VXLAN leaf and spine functionality
- Support EVPN-VXLAN border functionality (Type 5 to Type 5 stitching currently)
- Supports EVPN LAG multihoming in EVPN-VXLAN (ESI-LAG)
- Supports software upgrade and downgrade through

```
request system software
       add/rollback
```

For details on configuration of these features, refer to Software Documentation.

Benefits and Uses

The benefits and use cases of vJunosEvolved on standard x86 servers are as follows:

- **Reduce capital expenditure on physical lab**—You can use the vJunosEvolved platform for free to build test labs. This reduces the costs associated with physical switches and routers.
- **Reduce deployment time**—You can use the vJunosEvolved platform to build and test topologies virtually without building expensive physical labs. You can build virtual labs instantly, which helps in reducing costs and delays related to physical hardware deployment.
- **Eliminate lab hardware**—You can eliminate using lab hardware by downloading the vJunosEvolved platform instantly and for free
- **Educate and train**—You can use the vJunosEvolved platform to build labs for learning and education services for your employees.
- **Automate, build, and validate**—You can use the vJunosEvolved platform to validate various data center switching and routing topologies, pre-build configurations examples, and get automation ready.

The vJunosEvolved is intended only for lab use and not for commercial deployment.

Limitations

The vJunosEvolved has the following limitations:

- The vJunosEvolved has a fixed-form Junos OS Evolved architecture.
- Supports a maximum bandwidth of 2 Kpps or 3-5 Mbps over all the interfaces.
- To channelize WAN interfaces, run a boot argument with **channelized=yes** that is specified in the VM config and the interfaces speed config at vJunosEvolved CLI.

  Example: `set interfaces et-0/0/0 speed 25g number-of-sub-ports 8`
- For non-channelization, **channelized=no** is the default config.
- vJunosEvolved does not support single-root I/O virtualization (SR-IOV)
- COSIM reliably works up to 2000 pps across 128B-1500B packet length. You do not need a bandwidth license.
- For vJunosEvolved to function correctly as a Virtual Extensible LAN (VXLAN) Tunnel End Point (VTEP), you must configure tunnel termination using the `set forwarding-options tunnel-termination` command. Otherwise the traffic in the tunnel drops on the egress VTEP.

vJunosEvolved does not support the following features:

- Port mirroring
- Storm control
- Multichassis link aggregation (MC-LAG)
- VXLAN seamless stitching for Data Center Interconnect (DCI)
- Virtual Router Redundancy Protocol (VRRP)
- Multicast
- Q-in-Q tunneling through an Ethernet VPN–Virtual Extensible LAN (EVPN-VXLAN) fabric
- Layer-2 egress filtering on EVPN-VXLAN-enabled integrated routing and bridging (IRB) interfaces
- IPv6 underlay and overlay
- MAC filtering in EVPN-VXLAN fabric using a MAC list
- EVPN-VXLAN fabric enhanced loop detection using connectivity fault management (CFM)

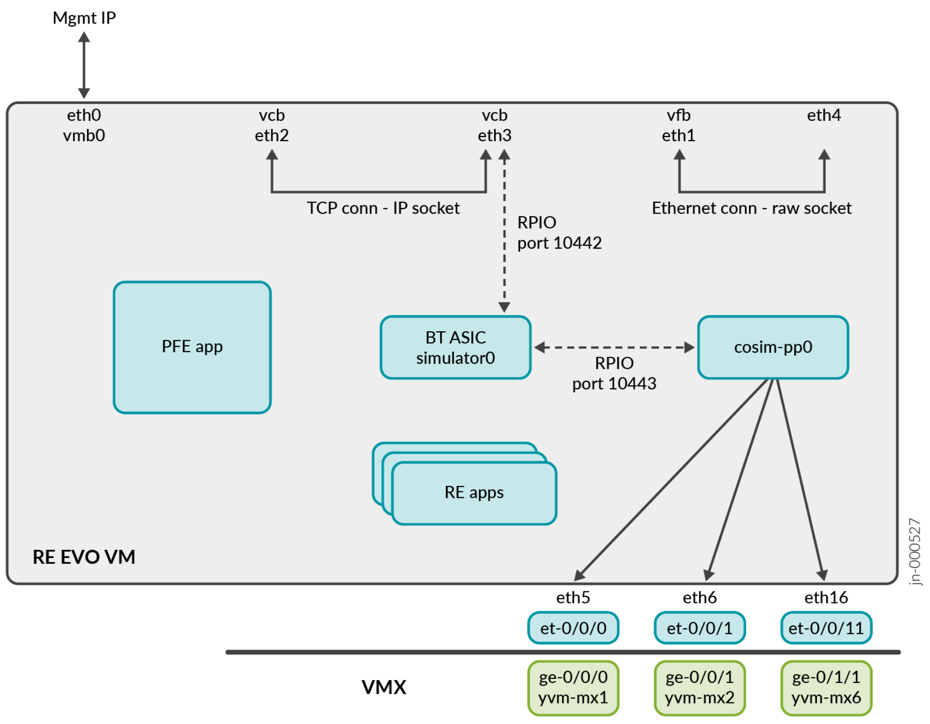
vJunosEvolved Architecture

The vJunosEvolved is a single VM solution where the Junos OS Evolved Routing Engine, Packet Forwarding Engine and vBT-COSIM run on the same VM. KVM is the hypervisor used to deploy Junos OS Evolved Linux VM.

When vJunosEvolved instance boots up, the vBT-COSIM simulation is also initialized along with Routing Engine and Packet Forwarding Engine.

vJunosEvolved can support up to 2000 pps of throughput (as per vBT-COSIM's specifications) using 4 cores and 8 GB memory. Any additional cores and memory configured are allocated to the Junos OS Evolved Routing Engine. For the anticipated lab use cases, 4 cores and 8 GB memory is sufficient.

Figure 1: vJunosEvolved Architecture



CHAPTER 2

vJunosEvolved Hardware and Software Requirements on KVM

IN THIS CHAPTER

Minimum Hardware and Software Requirements

This topic provides you a list of hardware and software requirements to start a vJunosEvolved instance.

Minimum Hardware Requirements for vJunosEvolved lists the hardware requirements for vJunosEvolved.

Table 1: Minimum Hardware Requirements for vJunosEvolved

Description	Value
Sample system configuration	For lab simulation and low performance (less than 2Kpps) use cases, any x86 processor (Intel or AMD) 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 4 cores are required. This is the standard support with vJunosEvolved platform and can be adjusted in the configuration based on requirement.
Memory	A minimum of 8GB is required.

Table 1: Minimum Hardware Requirements for vJunosEvolved *(Continued)*

Description	Value
Other requirements	• Intel VT-x capability. • Hyperthreading (recommended) • AES-NI

Table 2: Software Requirements for Ubuntu

Description	Value
Operating system **NOTE**: Only English localization is supported.	• Ubuntu 22.04 LTS • Ubuntu 20.04 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 • OVMF-Install the OVMF firmware on the host as the guest boots with UEFI BIOS.
Supported Deployment Environments	QEMU-KVM using libvirt You can run vJunosEvolved with EVE-NG in a VM or on bare metal.
	QEMU-KVM using libvirt EVE-NG supported for both bare metal and in a VM.

Table 2: Software Requirements for Ubuntu ***(Continued)***

Description	Value
vJunosEvolved Images	You can access the images from the vJunos Labs download area of juniper.net at: Free Virtual Junos OS Download for Labs

CHAPTER 3

Install and Deploy vJunosEvolved on KVM

IN THIS CHAPTER

Install vJunosEvolved on KVM

SUMMARY

Read this topic to understand how to install the vJunosEvolved in the KVM environment.

IN THIS SECTION

Prepare the Linux Host Servers to Install vJunosEvolved

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 vJunosEvolved on KVM

SUMMARY

Read this topic to understand how to deploy and manage the vJunosEvolved instance after you install it on KVM.

IN THIS SECTION

This topic describes:

- How to bring up vJunosEvolved 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 for deployment.

NOTE: Download the sample XML file and the vJunosEvolved image from the Juniper website.

Set Up the vJunosEvolved Deployment on the Host Server

This topic describes how to set up the vJunosEvolved deployment on the host server.

NOTE: This topic highlights only a few sections of the XML file that is used to deploy vJunosEvolved through libvirt.

The entire XML file **vJunosEvolved.xml** is available for download along with the VM image and associated documentation on the lab download page.

Install the packages mentioned in the minimum software requirements section, if they are not already installed. See Minimum Hardware Requirements for vJunosEvolved.

1. Create a Linux bridges for each WAN interface and configure them as up.

   For example, et-0/0/0, et-0/0/1, and so on, of vJunosEvolved that you plan to use.

   ```
   # ip link add et000 type bridge
   ```

   ```
   # ip link set et000 up
   ```

```
# ip link add et001 type bridge
```

```
# ip link set et001 up
```

2. Make a live disk copy of the provided QCOW2 vJunosEvolved image.

   ```
   # cd /root
   ```

   ```
   # cp vJunosEvolved-<release>.qcow2 vJunosEvolved-<release>-live.qcow2
   ```

   ```
   # chmod u+w vjunosEvolved-<release>-live.qcow2
   ```

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

3. Specify the number of cores provided to vJunosEvolved by modifying the following stanza. For the default memory of 8GB required by vJunosEvolved, use the following code snippet:

   ```
   <cpu>
     match="exact" mode="host-model">
     <topology cores="4" sockets="1" threads="1"/><model fallback="allow">qemu64</model>
     <feature name="svm" policy="disable"/>
   </cpu>
   ```

   NOTE: A sample **vJunosEvolved.xml** file is also available with the posted vJunosEvolved image. This document refers to key snippets from that sample file to illustrate the stanzas the you need to edit in the XML file.

   Use the sample **vJunosEvolved.xml** snippet files that are available with the posted vJunosEvolved image to prevent errors.

   The following codeblock provides an example CPU XML snippet, where the default number of cores required is 4, which is sufficient for most applications. You can increase the number of cores added, by modifying the below stanza.

   ```
   <domain xmlns:ns0="http://libvirt.org/schemas/domain/qemu/1.0" type="kvm">
   <name>vJunosEvo</name>
   <memory unit="KiB">8388608</memory>
   <currentMemory unit="KiB">8388608</currentMemory>
   <vcpu placement="static">4</vcpu>
   ```

You can increase the memory if needed. It also shows the name of the specific vJunosEvolved being spawned, which is vJunosEvo in this case.

4. Modify the name and location of your vJunosEvolved image.

   NOTE: For libvirt and QEMU-KVM, each vJunosEvolved VM on the host needs to be provided with its own uniquely named QCOW2 image.

   Use the following XML snippet to specify the name and location for your vJunosEvolved image:

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

5. Create the configuration disk image.

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

   The vJunosEvolved 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="raw"/>
    <source file="/root/config.img"/>
    <target dev="sda" bus=”usb”/>
</disk>
```

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

6. Set up the management Ethernet port.

   This step allows you to connect to the VM’s “re0:mgmt-0” that is the management port from outside the host server on which vJunosEvolved resides.

You need to have a routable IP address configured for re0:mgmt-0, either through a DHCP server or using standard CLI configuration.

```
<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>
```

The “eth0” in the stanza above 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 vJunosEvolved is up and running, telnet to it's console and configure the IP address for “re0:mgmt-0” using CLI configuration as shown below:

NOTE: The configurations in this step are examples or sample configuration snippet. You might have to set up a static route configuration as well.

```
# set interfaces re0:mgmt-0 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
```

Enable SSH to the RE management port.

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

7. Create a Linux bridge for each of the ports you specify in the XML file.

   The port names are specified in the following codeblock.

   The convention for vJunosEvolved is to use et00x. In the following example, et000 and et001 map to the Junos Evolved et-0/0/0 and et-0/0/1 interfaces respectively.

```

<interface>
  <interface type="bridge">
  <source bridge="et000"/>
  <model type="virtio"/>
  <mtu size='16000'/>
  <alias name="net1"/>
  <address bus="0x00" domain="0x0000" function="0x0" slot="0x08" type="pci"/>
</interface>
```

```
<interface type="bridge">
  <source bridge="et001"/>
  <model type="virtio"/>
  <mtu size='16000'/>
  <alias name="net2"/>
  <address bus="0x00" domain="0x0000" function="0x0" slot="0x09" type="pci"/>
</interface>
```

8. Provide a unique serial console port number for each vJunosEvolved on your host server.

   In this sample snippet "8610" is chosen.

```


</interface>
  <serial type="tcp">
  <source host="127.0.0.1" mode="bind" service="8610"/>
  <protocol type="telnet"/>
  <target port="0"/>
  <alias name="serial0"/>
</serial>
```

9. Create channelized or non-channelized interfaces.

   The “channelized=yes” in the command line arg provides an option to create channelized WAN interfaces. If nothing mentioned or value of “no” mentioned, then non-channelized interfaces are initialized at COSIM.

```


<ns0:commandline>
    <ns0:arg value="-smbios"/>
    <ns0:arg value="type=0,vendor=Bochs,version=Bochs"/>
    <ns0:arg value="-smbios"/>
    <ns0:arg value="type=3,manufacturer=Bochs"/>
    <ns0:arg value="-smbios"/>
    <ns0:arg
value="type=1,manufacturer=Bochs,product=Bochs,serial=chassis_no=0:slot=0:type=1:assembly_id
=0x0D20:platform=251:master=0: channelized=yes "/>
</ns0:commandline>
```

10. Boot vJunosEvolved using QEMU’s OVMF as mentioned below.

NOTE: From vJunos OS Release 24.1 release, you must boot vJunosEvolved only with UEFI BIOS. Otherwise, the BIOS hangs when you boot.

```
<ns0:arg value="-bios"/>
    <ns0:arg value="/usr/share/qemu/OVMF.fd"/>
```

11. Create vJunosEvolved VM using the vJunosEvolved.xml file.

```
# virsh create vJunosEvolved.xml
```

This creates the first vJunosEvolved VM. The subsequent VMs can be vJunosEvolved2,vJunosEvolved3 and so on.

Domain vJunosEvolved created from **vJunosEvolved.xml**

Verify the vJunosEvolved VM

This topic describes how to verify whether vJunosEvolved is up and running.

1. Verify if the vJunosEvolved is up and running.

NOTE: The XML file for download is "vJunosEvolved.xml". If you are creating more than one instance, then the domain and XML and live disk files names must be unique.
But for a single instance it looks like this:

```
# virsh list
 Id   Name          State
----------------------------
 74   vJunosEvolved running
```

2. Connect to the serial console of the Routing Engine VM.

   You can find the port to connect to from the XML file.

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

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

3. Verify whether the ET interfaces that you had specified in your XML file are up.

   `show interfaces terse`

   For example if "et000" and "et001" were specified in your XML file, then the et-0/0/0 and et-0/0/1 interfaces should be in "up" state. Other interfaces also shows up, but those interfaces can't pass traffic.

```
root> show interfaces terse
Interface               Admin Link Proto    Local                 Remote
et-0/0/0                up    up
et-0/0/0.16386          up    up   multiservice
pfh-0/0/0               up    up
pfh-0/0/0.16383         up    up   inet
et-0/0/1                up    up
et-0/0/1.16386          up    up   multiservice
[snip]
```

4. Verify whether a VNET interface under each corresponding "et" bridge is configured.

   Use the `brctl` command on the host server once vJunosEvolved is started. This command shows a vnet interface under each corresponding "et" bridge:

```
# brctl show et000
bridge name   bridge id         STP enabled     interfaces
et000         8000.fe54001a0d69 no              vnet13
# brctl show et001
bridge name    bridge id        STP enabled    interfaces
et001         8000.fe540077af98 no              vnet14
```

Configure vJunosEvolved on KVM

SUMMARY

Read this topic to understand the connections and configurations of the vJunosEvolved instances deployment on KVM

IN THIS SECTION

Connect to vJunosEvolved

Telnet to the serial console number specified in the XML file to connect to vJunosEvolved. See details provided in the example below:

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

Configure vJunosEvolved Channelization Interfaces

The vJunosEvolved configurations are the same as the hardware configurations in Junos Evolved CLI.

If channelization of WAN interfaces is required, then boot argument “channelized=yes” must be specified in the XML configuration and the interfaces speed should be configured at Junos Evolved CLI. Sample configuration shown below:

Run the **set interfaces et-0/0/0 speed 25g number-of-sub-ports 8** command.

```
[edit]
# set interfaces et-0/0/0 speed 25g number-of-sub-ports 8
> show interfaces terse
Interface               Admin Link Proto    Local                 Remote
et-0/0/0:0              up    up
et-0/0/0:0. 16386    up    up    inet  multiservice
et-0/0/0:1              up    up
```

```
et-0/0/0:1.16386        up    up   multiservice
et-0/0/0:2              up    up
et-0/0/0:2.16386        up    up   multiservice
et-0/0/0:3              up    up
et-0/0/0:3.16386        up    up   multiservice
et-0/0/0:4              up    up
et-0/0/0:4.16386        up    up   multiservice
et-0/0/0:5              up    up
et-0/0/0:5.16386        up    up   multiservice
et-0/0/0:6              up    up
et-0/0/0:6.16386        up    up   multiservice
et-0/0/0:7              up    up
et-0/0/0:7.16386        up    up   multiservice
```

You can specify the number of sub ports under a WAN interface to channelize. The default number of non-channelized ports is 12 (et-0/0/0 to et-0/0/11), but you can channelize each interface with 8 sub-ports at 25G speed. You must have this Junos Evolved CLI configuration when channelized option is set to "yes" in the XML. Otherwise, traffic won't pass through.

Table 3: Supported Ports and Ports Speed on vJunosEvolved

Physical Port	Ports	Speed
`et-0/0/0:0 to et-0/0/0:7`	8	8x25G
`et-0/0/1:0 to et-0/0/1:7`	8	8x25G
`et-0/0/2:0 to et-0/0/2:7`	8	8x25G
`et-0/0/3:0 to et-0/0/3:7`	8	8x25G
`et-0/0/4:0 to et-0/0/4:3`	4	4x25G
`et-0/0/5`	0	shutdown
`et-0/0/6:0 to et-0/0/6:3`	4	4x25G
`et-0/0/7`	0	shutdown
`et-0/0/8:0 to et-0/0/8:7`	8	8x25G

Table 3: Supported Ports and Ports Speed on vJunosEvolved *(Continued)*

Physical Port	Ports	Speed
`et-0/0/9:0 to et-0/0/9:7`	8	8x25G
`et-0/0/10:0 to et-0/0/10:7`	8	8x25G
`et-0/0/11:0 to et-0/0/11:7`	8	8x25G

Configure the Media MTU

You can configure the media maximum transmission unit (MTU) in the range 288 through 16000. 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 et-0/0/0 mtu <mtu>
```

NOTE: The maximum supported MTU value is 16000 bytes.

For example:

```
[edit]
user@host# set interface et-0/0/0 mtu 16000
```

CHAPTER 4

Verify and Troubleshoot vJunosEvolved on KVM

IN THIS CHAPTER

Verify vJunosEvolved on KVM

SUMMARY

Use this topic to verify your vJunosEvolved configurations and for any troubleshooting information.

IN THIS SECTION

Verify That the VM is Running

- Verify whether vJunosEvolved is running after you install it.

```
virsh list
```

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

```
# virsh list
 Id   Name          State
---------------------------
 72   vJunosEvo-RE1   running
```

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

NOTE: Do not use the `virsh destroy` command because this commmand can corrupt the vJunosEvolved 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

On the host server, use the `lscpu` command 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 information is 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 vJunosEvolved instance.

```
root > show log ?
```

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

For example, to view the EVO init logs, run the `root> show log evoinit.log` command.

The log files can be viewed from the **/var/log** directory of the vJunosEvolved RE. These logs are the standard vJunosEvolved log files that are also found on other Juniper Networks® products. The serial console can be used to log in to the Routing Engine VM. Alternatively, you can SSH to the Routing Engine VM and view the same information.

Some of the key log files collected are the following:

- Use the `request system debug-info` command to transfer all the system traces in a file named */var/tmp/debug_collector_<date_time>.tar.gz*
- */var/log/cosim.log* and */var/log/cosim_ppd.log*: COSIM traces during initialization.

Collect Core Files

Use the `show system core-dumps` command to view the collected core files. You can transfer these core files to an external server for analysis through the management interface on the Routing Engine.

The */var/crash* of the directory of the JunosEvolved Routing Engine stores all the core files. You can follow the standard procdures of the Junos OS to transfer the core files vJunosEvolved Routing Engine to an external host.

2

PART

vJunosEvolved-BX Deployment for KVM

CHAPTER 5

Overview

IN THIS CHAPTER

What is vJunosEvolved-BX

SUMMARY

This topic provides an overivew, key features, benefits, and limitations of vJunosEvolved-BX.

IN THIS SECTION

Overview

IN THIS SECTION

vJunosEvolved-BX is a virtual version of a Juniper platform PTX10002-36QDD that runs the Junos OS Evolved. You can install a vJunosEvolved-BX as a virtual machine (VM) on an x86 server.

vJunosEvolved-BX is a single virtual machine (VM) that you can use only in labs and not in the production environment. The vJunosEvolved-BX is built using PTX10002-36QDD as a reference Juniper router, which is a fixed-configuration packet transport router on the Junos® OS Evolved platform.

The vJunosEvolved-BX Routing Engine and the vBX-COSIM (a virtual BX chip) that performs the packet processing run on the same VM.

The same vJunosEvolved QCOW2 image is used with vJunosEvolved-BX specific QEMU arguments.

Instead of using hardware router, you can use the vJunosEvolved-BX to start the Junos software for testing the network configurations and protocols.

The vJunosEvolved-BX virtual platform primarily acts as a test platform for lab simulations for the customers.

NOTE: Bandwidth licenses are not required and are not provided. You can ignore any license check messages if you get an alert.

vJunosEvolved-BX Installation Overview

You can install the software components of vJunosEvolved-BX on an industry-standard x86 server running a Linux KVM hypervisor (Ubuntu 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 vJunosEvolved-BX instances on a single server.

Key Features

This topic provides you a list of key features that the vJunosEvolved-BX platform supports.

The vJunosEvolved-BX platform supports the following key features:

- Supports both vBX chips (BX0 and BX1) 36 non-channelized interfaces.
- Supports only 4x100G channelization with 144 channelized interfaces.
- Supports software upgrade and downgrade through

  ```
  request system software
          add/rollback
  ```

- By default both vBX chips are powered on. CLI configuration is available to power on or off each chip.

For details on configuration of these features, refer to Software Documentation.

Benefits and Uses

The benefits and use cases of vJunosEvolved on standard x86 servers are as follows:

- **Reduce capital expenditure on physical lab**—You can use the vJunosEvolved-BX platform for free to build test labs. This reduces the costs associated with PTX10002-36QDD physical routers.
- **Reduce deployment time**—You can use the vJunosEvolved-BX platform to build and test topologies virtually without building expensive physical labs. You can build virtual labs instantly, which helps in reducing costs and delays related to physical hardware deployment.
- **Eliminate lab hardware**—You can eliminate using lab hardware by downloading the vJunosEvolved-BX platform instantly and for free.
- **Educate and train**—You can use the vJunosEvolved-BX platform to build labs for learning and education services for your employees.
- **Automate, build, and validate**—You can use the vJunosEvolved-BX platform to validate various data center switching and routing topologies, pre-build configurations examples, and get automation ready.

The vJunosEvolved-BX is intended only for lab use and not for commercial deployment.

Limitations

The vJunosEvolved has the following limitations:

- The vJunosEvolved-BX has a fixed-form Junos OS Evolved architecture.
- It does not support in-service software upgrade (ISSU).
- You cannot attach or detach interfaces while VM is running.
- Supports a maximum bandwidth of 2 Kpps or 3-5 Mbps over all the interfaces.
- Only 4x100G channelization is supported. To channelize WAN interfaces, run a boot argument with channelized=yes that is specified in the VM configuration and the interfaces speed configuration at CLI.

  Example: `set interfaces et-0/0/0 speed 100g number-of-sub-ports 4`
- For non-channelization, channelized=no is the default configuration.
- vJunosEvolved-BX does not support single-root I/O virtualization (SR-IOV).
- COSIM reliably works up to 2000 pps across 128B-1500B packet length. You do not need a bandwidth license.
- vJunosEvolved-BX Routing Engine boots only with Normal Power mode. As this is virtual platform, Optimized Power mode can’t be applied here.

vJunosEvolved-BX does not support the following features:

- Port mirroring with policers
- Routing Engine and PFE Resiliency support
- Timing and Synchronization
- Firewall policer and Firewall filter flex offset match
- MACsec IFD support
- Inline CFM
- PFE and ASIC restart support
- Optics and EM policy

vJunosEvolved-BX Architecture

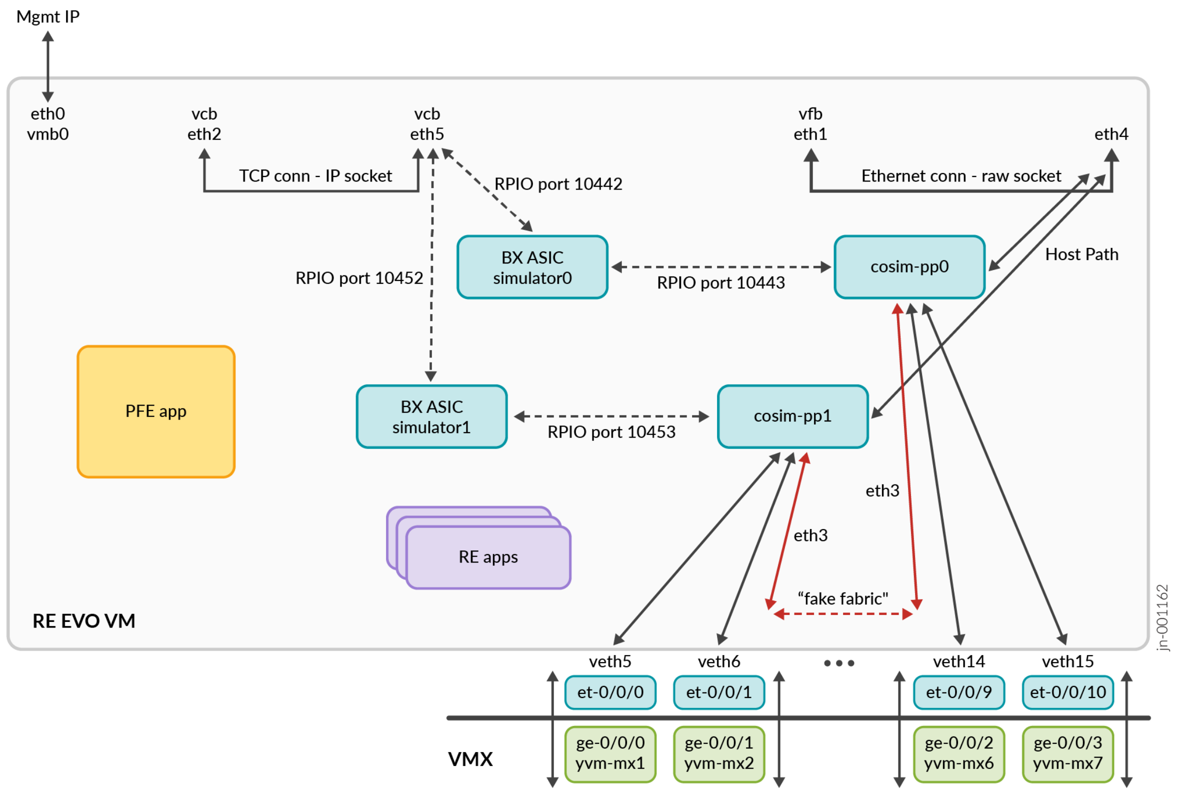
The vJunosEvolved-BX is a single VM solution where the Junos OS Evolved Routing Engine (RE), Packet Forwarding Engine (PFE) and vBX-COSIM run on the same VM. KVM is the hypervisor used to deploy Junos OS Evolved Linux VM.

When vJunosEvolved-BX instance boots up, the vBX-COSIM simulation is also initialized along with Routing Engine and Packet Forwarding Engine (PFE).

vJunosEvolved-BX can support up to 2000 pps of throughput (as per vBX-COSIM's specifications) using 4 cores and 8GB memory. Any additional cores and memory configured are allocated to the Junos OS Evolved Routing Engine. For the anticipated lab use cases, 4 cores and 8GB memory is sufficient.

The goal is here to use same vJunosEvolved QCOW2 image with vJunosEvolved-BX personality specifying at QEMU arguments.

Figure 2: vJunosEvolved-BX Architecture



CHAPTER 6

vJunosEvolved-BX Hardware and Software Requirements on KVM

IN THIS CHAPTER

Minimum Hardware and Software Requirements

This topic provides you a list of hardware and software requirements to start a vJunosEvolved-BX instance.

Minimum Hardware Requirements for vJunosEvolved lists the hardware requirements for vJunosEvolved-BX.

Table 4: Minimum Hardware Requirements for vJunosEvolved-BX

Description	Value
Sample system configuration	For lab simulation and low performance (less than 100 Mbps) use cases, any x86 processor (Intel or AMD) 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 4 cores is required. This is the standard support with vJunosEvolved platform and can be adjusted in the configuration that is based on requirement.
Memory	A minimum of 8 GB is required.

Table 4: Minimum Hardware Requirements for vJunosEvolved-BX *(Continued)*

Description	Value
Other requirements	• Intel VT-x capability • Hyperthreading (recommended) • AES-NI

Table 5: Software Requirements for Ubuntu

Description	Value
Operating system **NOTE**: Only English localization is supported.	• Ubuntu 22.04 LTS • Ubuntu 20.04 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 • OVMF-Install the OVMF firmware on the host as the guest boots with UEFI BIOS.
Supported Deployment Environments	QEMU-KVM using libvirt You can run vJunosEvolved with EVE-NG in a VM or on bare metal.
	QEMU-KVM using libvirt EVE-NG supported for both bare metal and in a VM.

Table 5: Software Requirements for Ubuntu ***(Continued)***

Description	Value
vJunosEvolved Images	You can access the images from the vJunos Labs download area of juniper.net at: Free Virtual Junos OS Download for Labs

CHAPTER 7

Install and Deploy vJunosEvolved-BX on KVM

IN THIS CHAPTER

Install vJunosEvolved-BX on KVM

SUMMARY

Read this topic to understand how to install the vJunosEvolved in the KVM environment.

IN THIS SECTION

Prepare the Linux Host Servers to Install vJunosEvolved-BX

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 vJunosEvolved-BX on KVM

SUMMARY

Read this topic to understand how to deploy and manage the vJunosEvolved-BX instance after you install it on KVM.

IN THIS SECTION

This topic describes:

- How to bring up vJunosEvolved-BX 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 for deployment.

NOTE: Download the sample XML file and the vJunosEvolved-BX image from the Juniper website.

Set Up the vJunosEvolved-BX Deployment on the Host Server

This topic describes how to set up the vJunosEvolved-BX deployment on the host server.

NOTE: This topic highlights only a few sections of the XML file that is used to deploy vJunosEvolved-BX through libvirt.
The entire XML file **vJunosEvolved-BX.xml** is available for download along with the VM image and associated documentation on the lab download page.

Install the packages mentioned in the minimum software requirements section, if they are not already installed. See Minimum Hardware Requirements for vJunosEvolved-BX.

1. Create a Linux bridges for each WAN interface and configure them as up.

   For example, et-0/0/0, et-0/0/1, and so on, of vJunosEvolved-BX that you plan to use.

   ```
   # ip link add et000 type bridge
   ```

   ```
   # ip link set et000 up
   ```

```
# ip link add et001 type bridge
```

```
# ip link set et001 up
```

2. Make a live disk copy of the provided QCOW2 vJunosEvolved-BX image.

   ```
   # cd /root
   ```

   ```
   # cp vJunosEvolved-<release>.qcow2 vJunosEvolved-BX-<release>-live.qcow2
   ```

   ```
   # chmod u+w vjunosEvolved-BX-<release>-live.qcow2
   ```

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

3. Specify the number of cores provided to vJunosEvolved-BX by modifying the following stanza. For the default memory of 8GB required by vJunosEvolved-BX, use the following code snippet:

   ```
   <cpu>
     match="exact" mode="host-model">
     <topology cores="4" sockets="1" threads="1"/><model fallback="allow">qemu64</model>
     <feature name="svm" policy="disable"/>
   </cpu>
   ```

   NOTE: A sample **vJunosEvolved-BX.xml** file is also available with the posted vJunosEvolved-BX image. This document refers to key snippets from that sample file to illustrate the stanzas the you need to edit in the XML file.

   Use the sample **vJunosEvolved-BX.xml** snippet files that are available with the posted vJunosEvolved-BX image to prevent errors.

   The following codeblock provides an example CPU XML snippet, where the default number of cores required is 4, which is sufficient for most applications. You can increase the number of cores added, by modifying the below stanza.

   ```
   <domain xmlns:ns0="http://libvirt.org/schemas/domain/qemu/1.0" type="kvm">
   <name>vJunosEvolved-BX</name>
   <memory unit="KiB">8388608</memory>
   <currentMemory unit="KiB">8388608</currentMemory>
   <vcpu placement="static">4</vcpu>
   ```

You can increase the memory if needed. It also shows the name of the specific vJunosEvolved-BX being spawned, which is vJunosEvo in this case.

4. Modify the name and location of your vJunosEvolved-BX image.

   NOTE: For libvirt and QEMU-KVM, each vJunosEvolved-BX VM on the host needs to be provided with its own uniquely named QCOW2 image.

   Use the following XML snippet to specify the name and location for your vJunosEvolved-BX image:

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

5. Create the configuration disk image.

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

   The vJunosEvolved 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="raw"/>
    <source file="/root/config.img"/>
    <target dev="sda" bus="usb"/>
</disk>
```

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

6. Set up the management Ethernet port.

   This step allows you to connect to the VM's “re0:mgmt-0” that is the management port from outside the host server on which vJunosEvolved-BX resides.

You need to have a routable IP address configured for re0:mgmt-0, either through a DHCP server or using standard CLI configuration.

```
<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>
```

The "eth0" in the stanza above 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 vJunosEvolved-BX is up and running, telnet to it's console and configure the IP address for "re0:mgmt-0" using CLI configuration as shown below:

NOTE: The configurations in this step are examples or sample configuration snippets. You might have to set up a static route configuration as well.

```
# set interfaces re0:mgmt-0 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
```

Enable SSH to the RE management port.

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

7. Create a Linux bridge for each of the ports you specify in the XML file.

   The vJunosEvolved-BX's 36 ports and it's vBX chip mapping are given in the table below.

Table 6: vJunosEvolved-BX Ports and vBX Chip Mapping

WAN port	BX0 or BX1	Default Speed
et-0/0/0 – et-0/0/8	BX1	1x400G
et-0/0/9 – et-0/0/26	BX0	1x400G
et-0/0/27 – et-0/0/35	BX1	1x400G

The port names are specified in the following codeblock.

The convention for vJunosEvolved-BX is to use et00x. In the following example, et000 and et001 map to the Junos Evolved et-0/0/0 and et-0/0/1 interfaces respectively.

```
<interface>
  <interface type="bridge">
  <source bridge="et000"/>
  <model type="virtio"/>
  <mtu size='16000'/>
  <alias name="net1"/>
  <address bus="0x00" domain="0x0000" function="0x0" slot="0x08" type="pci"/>
</interface>
<interface type="bridge">
  <source bridge="et001"/>
  <model type="virtio"/>
  <mtu size='16000'/>
  <alias name="net2"/>
  <address bus="0x00" domain="0x0000" function="0x0" slot="0x09" type="pci"/>
</interface>
```

8. Provide a unique serial console port number for each vJunosEvolved-BX on your host server.

   In this sample sinppet "8610" is chosen.

```
</interface>
  <serial type="tcp">
  <source host="127.0.0.1" mode="bind" service="8610"/>
  <protocol type="telnet"/>
  <target port="0"/>
  <alias name="serial0"/>
</serial>
```

9. Create channelized or non-channelized interfaces.

   The “channelized=yes” in the command line arg provides an option to create channelized WAN interfaces. If nothing mentioned or value of “no” mentioned, then non-channelized interfaces are initialized at COSIM.

```
<ns0:commandline>
```

```
    <ns0:arg value="-smbios"/>
    <ns0:arg value="type=0,vendor=Bochs,version=Bochs"/>
    <ns0:arg value="-smbios"/>
    <ns0:arg value="type=3,manufacturer=Bochs"/>
    <ns0:arg value="-smbios"/>     <ns0:arg
value="type=1,manufacturer=Bochs,product=Bochs,serial=chassis_no=0:slot=0:type=1:assembly_id
 =0x0DA9:platform=272:master=0: channelized=yes "/>
</ns0:commandline>
```

10. Boot vJunosEvolved-BX using QEMU's OVMF as mentioned below.

    NOTE: From vJunos OS Release 24.2R1 release, you must boot vJunosEvolved-BX only with UEFI BIOS. Otherwise, the BIOS hangs when you boot.

    ```
    <ns0:arg value="-bios"/>
        <ns0:arg value="/usr/share/qemu/OVMF.fd"/>
    ```

11. Create vJunosEvolved-BX VM using the vJunosEvolved-BX.xml file.

    ```
    # virsh create vJunosEvolved-BX.xml
    ```

    This creates the first vJunosEvolved VM. The subsequent VMs can be vJunosEvolved-BX2, vJunosEvolved-BX3 and so on.

    Domain vJunosEvolved-BX created from **vJunosEvolved-BX.xml**

Verify the vJunosEvolved-BX VM

This topic describes how to verify whether vJunosEvolved-BX is up and running.

1. Verify if the vJunosEvolved-BX is up and running.

   NOTE: The XML file for download is "vJunosEvolved-BX.xml". If you are creating more than one instance, then the domain and XML and live disk files names must be unique. But for a single instance it looks like this:

   ```
   # virsh list
    Id    Name          State
   ```

```
----------------------------
 74   vJunosEvolved-BX running
```

2. Connect to the serial console of the Routing Engine VM.

   You can find the port to connect to from the XML file.

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

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

3. Verify whether the ET interfaces that you had specified in your XML file are up.

   `show interfaces terse`

   For example if "et000" and "et001" were specified in your XML file, then the et-0/0/0 and et-0/0/1 interfaces should be in "up" state. Other interfaces also shows up, but those interfaces can't pass traffic.

```
root> show interfaces terse
Interface               Admin Link Proto    Local                 Remote
et-0/0/0                up    up
et-0/0/0.0              up    up   inet     1.1.1.1/24          multiservice
pfh-0/0/0               up    up
pfh-0/0/0.16383         up    up   inet
et-0/0/1                up    up
et-0/0/1.16386          up    up   multiservice
```

4. Verify whether a VNET interface under each corresponding "et" bridge is configured.

   Use the `brctl` command on the host server once vJunosEvolved-BX is started. This command shows a vnet interface under each corresponding "et" bridge:

```
# brctl show et000
bridge name   bridge id         STP enabled     interfaces
et000         8000.fe54001a0d69 no              vnet13
# brctl show et001
```

```
bridge name     bridge id         STP enabled     interfaces
et001          8000.fe540077af98 no                vnet14
```

Configure vJunosEvolved-BX on KVM

SUMMARY

Read this topic to understand the connections and configurations of the vJunosEvolved-BX instances deployment on KVM

IN THIS SECTION

Connect to vJunosEvolved-BX

Telnet to the serial console number specified in the XML file to connect to vJunosEvolved-BX. See details provided in the example below:

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

Configure vJunosEvolved-BX Channelization Interfaces

The vJunosEvolved-BX configurations are the same as the hardware configurations in Junos Evolved-BX CLI.

If channelization of WAN interfaces is required, then boot argument “channelized=yes” must be specified in the XML configuration and the interfaces speed should be configured at Junos Evolved-BX CLI. Only 4x100G channelization is supported with total of 144 channelized interfaces.

Sample configuration shown below:

Run the **set interfaces et-0/0/0 speed 100g number-of-sub-ports 4** command.

```
[edit]
# set interfaces et-0/0/0 speed 100g number-of-sub-ports 4
> show interfaces terse
Interface               Admin Link Proto  Local    Remote
et-0/0/0:0              up    up
et-0/0/0:0. 16386       up    up   inet   multiservice
et-0/0/0:1              up    up
et-0/0/0:1.16386        up    up   multiservice
et-0/0/0:2              up    up
et-0/0/0:2.16386        up    up   multiservice
et-0/0/0:3              up    up
et-0/0/0:3.16386        up    up   multiservice
```

You can specify the number of sub ports under a WAN interface to channelize. The default number of non-channelized ports is 36 (et-0/0/0 to et-0/0/35), but you can channelize each interface with 4 subports at 100G speed. You must have this Junos Evolved-BX CLI configuration when channelized option is set to "yes" in the XML. Otherwise, traffic won't pass through. Table to show vBX's mapping with channelized WAN ports.

Table 7: Supported Ports and Ports Speed on vJunosEvolved-BX

WAN Port	BX0 or BX1	Supported Port Speed
`et-0/0/0:0-3 to et-0/0/8:0-3`	`BX1`	`4x100G`
`et-0/0/9:0-3 to et-0/0/26:0-3`	`BX0`	`4x100G`
`et-0/0/27:0-3 to et-0/0/35:0-3`	`BX1`	`4x100G`

Configure the vBX Chips

By default, both BX is running, and all the 36 interfaces are visible at show interfaces terse.

If there is a need to power off BX's then following command is useful.

Chassis Configuration to power off vBX0

```
[edit]
```

```
user@host# set chassis fpc 0 pfe 0 power off
```

```
user@host#set chassis fpc 0 pfe 1 power off
```

```
user@host# commit
```

Chassis Configuration to power off vBX1

```
[edit]
```

```
user@host# set chassis fpc 0 pfe 2 power off
```

```
user@host##set chassis fpc 0 pfe 3 power off
```

```
user@host# commit
```

Configure the Media MTU

You can configure the media maximum transmission unit (MTU) in the range 288 through 16000. 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 et-0/0/0 mtu <mtu>
```

NOTE: The maximum supported MTU value is 16000 bytes.

For example:

```
[edit]
user@host# set interface et-0/0/0 mtu 16000
```

CHAPTER 8

Verify and Troubleshoot vJunosEvolved-BX on KVM

IN THIS CHAPTER

Verify vJunosEvolved-BX on KVM

SUMMARY

Use this topic to verify your vJunosEvolved-BX configurations and for any troubleshooting information.

IN THIS SECTION

Verify That the VM is Running

- Verify whether vJunosEvolved is running after you install it.

```
virsh list
```

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

```
# virsh list
 Id   Name         State
---------------------------
 72   vJunosEvolved-BX running
```

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

NOTE: Do not use the `virsh destroy` command because this commmand can corrupt the vJunosEvolved 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

On the host server, use the `lscpu` command 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 information is 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 vJunosEvolved-BX instance.

```
root > show log ?
```

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

For example, to view the EVO init logs, run the `root> show log evoinit.log` command.

The log files can be viewed from the **/var/log** directory of the vJunosEvolved-BX. These logs are the standard vJunosEvolved-BX log files that are also found on other Juniper Networks® products. The serial console can be used to log in to the Routing Engine VM. Alternatively, you can SSH to the Routing Engine VM and view the same information.

Some of the key log files collected are the following:

- Use the `request system debug-info` command to transfer all the system traces in a file named */var/tmp/debug_collector_<date_time>.tar.gz*
- Each vBX chips logs are available at: */var/log/cosim.log.0* and */var/log/cosim.log.1*.

  Also, cosim-pp traces during initialization are at: */var/log/cosim_ppd.log.0 and /var/log/cosim_ppd.log.1*

Collect Core Files

Use the `show system core-dumps` command to view the collected core files. You can transfer these core files to an external server for analysis through the management interface on the Routing Engine.

The */var/crash* of the directory of the JunosEvolved-BX stores all the core files. You can follow the standard procdures of the Junos OS to transfer the core files vJunosEvolved-BX to an external host.