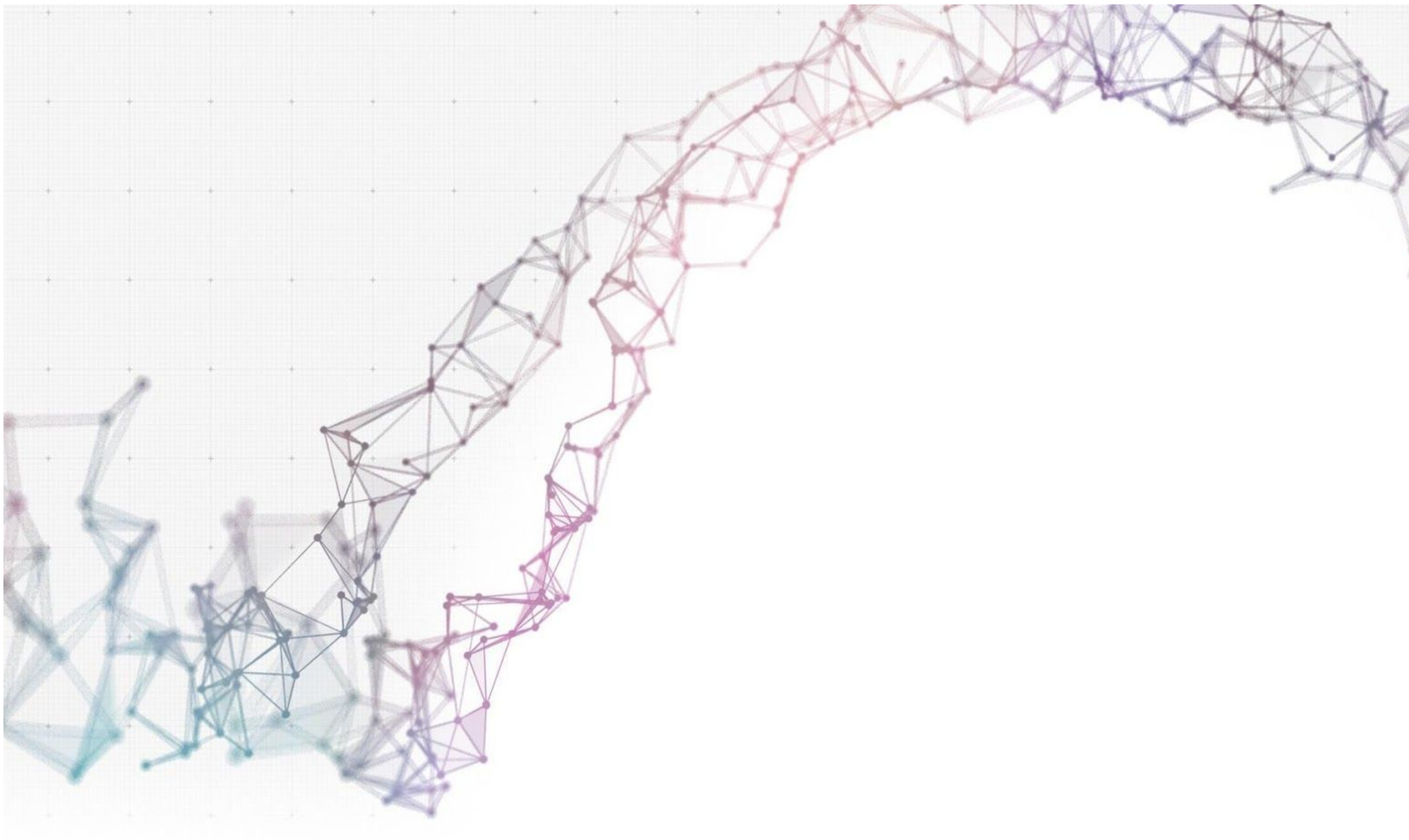


INSTALLATION RUNBOOK FOR Juniper vMX

Application Type:	**12VNF virtual router**
vMX Version:	**15.1F5-S1.5**
MOS Version:	**8.0**
OpenStack version:	**Liberty**

1 Introduction

This document is to serve as a detailed Deployment Guide for Juniper vMX (MOS) deployed with Mirantis OpenStack (MOS). This document describes the reference architecture, installation steps for validated vMX with Mirantis OpenStack, limitations and testing procedures.

1.1 Target Audience

This guide is designed for OpenStack Administrators who are deploying Juniper vMX with Mirantis OpenStack.

2 Application Overview

2.1 Juniper vMX Overview

Juniper vMX router is a virtual version of the MX Series 3D Universal Edge Router. Like the MX Series router, the vMX router runs Junos Operating System (Junos OS) and supports Junos OS packet handling and forwarding modeled after the Trio chipset. Configuration and management of vMX routers are the same as for physical MX Series routers, allowing you to add the vMX router to existing network without having to update your Operations Support Systems (OSS).

vMX runs on an industry-standard x86 server with Linux operating system, applicable third-party software, and the Kernel-based Virtual Machine (KVM) hypervisor. vMX software components come in one software package. As part of the package, an orchestration script is included, which encompasses a configuration file that you may customize for your vMX deployment. You may also install multiple vMX instances on one server.

vMX consists of two virtual machine (VM) components:

- **Virtual Control Plane (VCP)—**Also known as vRE, a virtual machine that runs Junos OS Routing Engine and FPC microkernel software.

- **Virtual Forwarding Plane (VFP)—**Also known as vPFE, a virtual machine that runs the Packet Forwarding Engine, referred to as RIOT, which is modeled after the Trio chipset. vMX traffic comes in through the physical NICs of the host and is sent into the VFP VM through virtual NICs using SR-IOV (device pass-through) or virtio (paravirtualized device drivers). The vPFE image contained in the ISO is only a virtio version.

The connectivity between the two VMs is managed through the following networks:

- Internal network (br-int) for communication between the VCP and the VFP
- External network (br-ext) for management access to the VMs

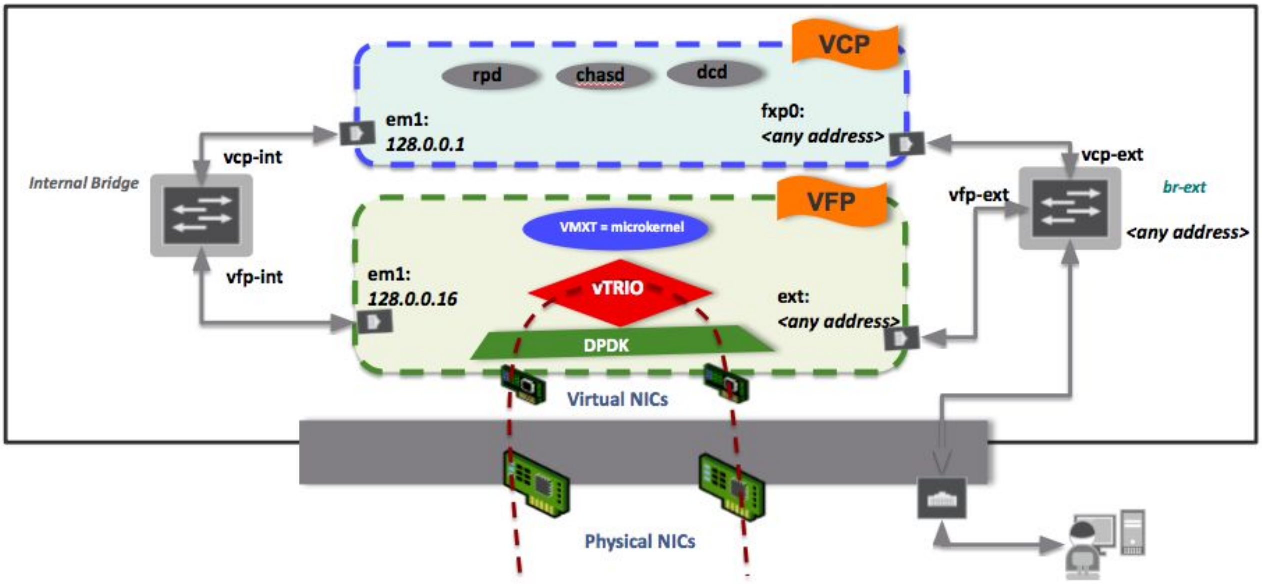


Figure 1 vMX Architecture

3 Joint Reference Architecture

The reference architecture used for this run book is as follow:

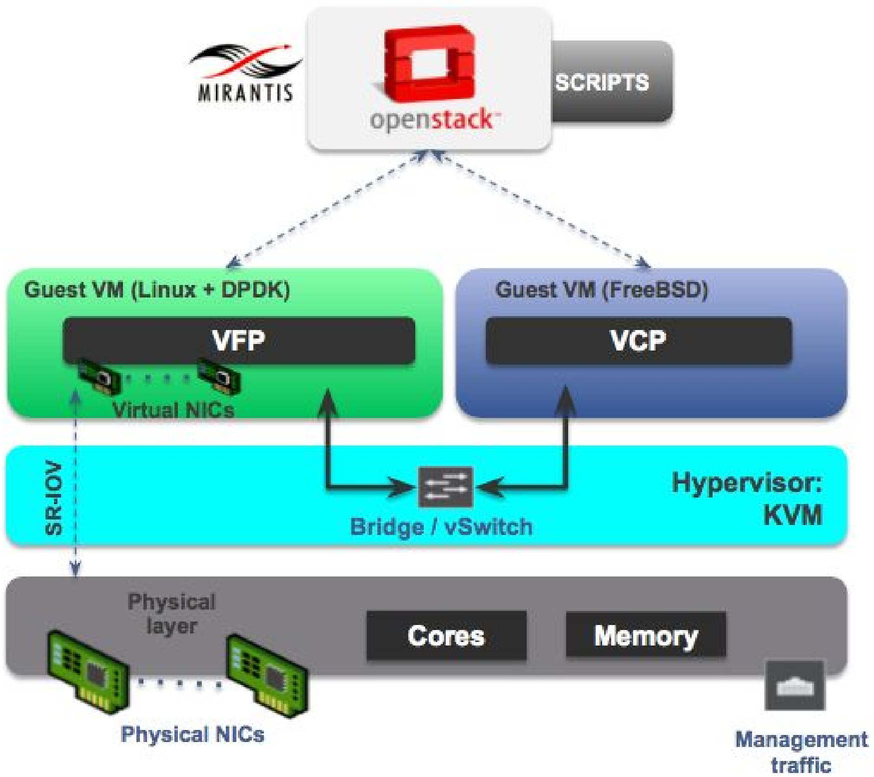


Figure 2 vMX joint architecture with Mirantis OpenStack

Below are the high level steps to bring up vMX on Mirantis OpenStack:

1. **Creation of OpenStack environment**
   Download and install MOS 8.0
2. **Preinstall checks**
   Check versions of OS, grub, kernel, KVM, libvirt, virsh connectivity test to QEMU, IXGBE
3. **Setup internals**
   Create and Deploy Mirantis OpenStack environment
4. **Bring up vMX**
   Bring up VCP and VFP
   Affinities CPU Cores for best performance

OpenStack Heat template is also developed for vMX deployment, and the workflow is illustrated as follow:

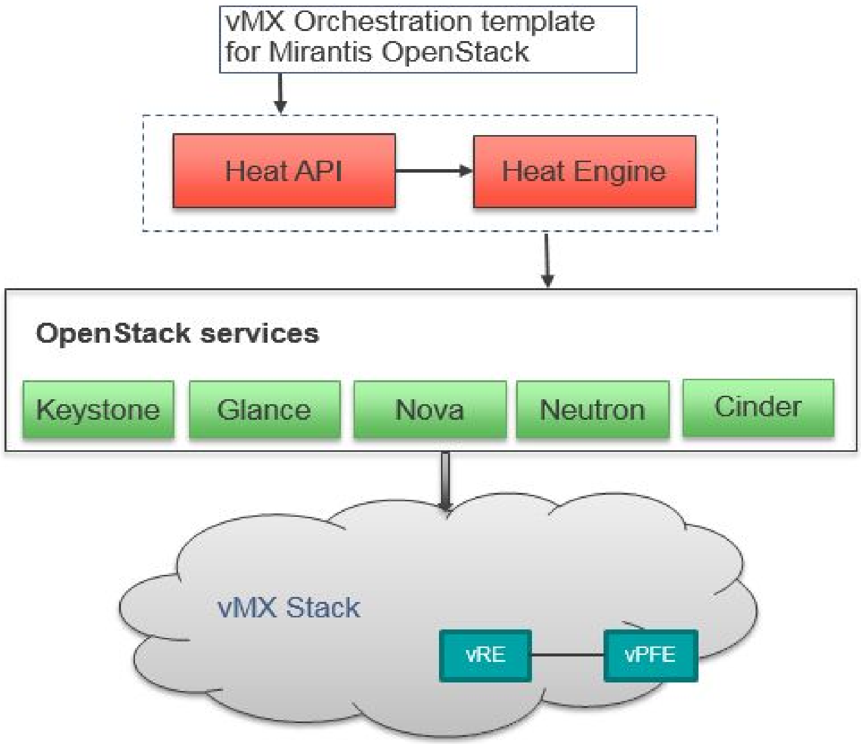


Figure 3 Heat template basic workflow on MOS 8.0

4 Physical & Logical Network Topology

The setup and testing in this runbook will be based on the logical and physical topology outlined in *Figure 4*.

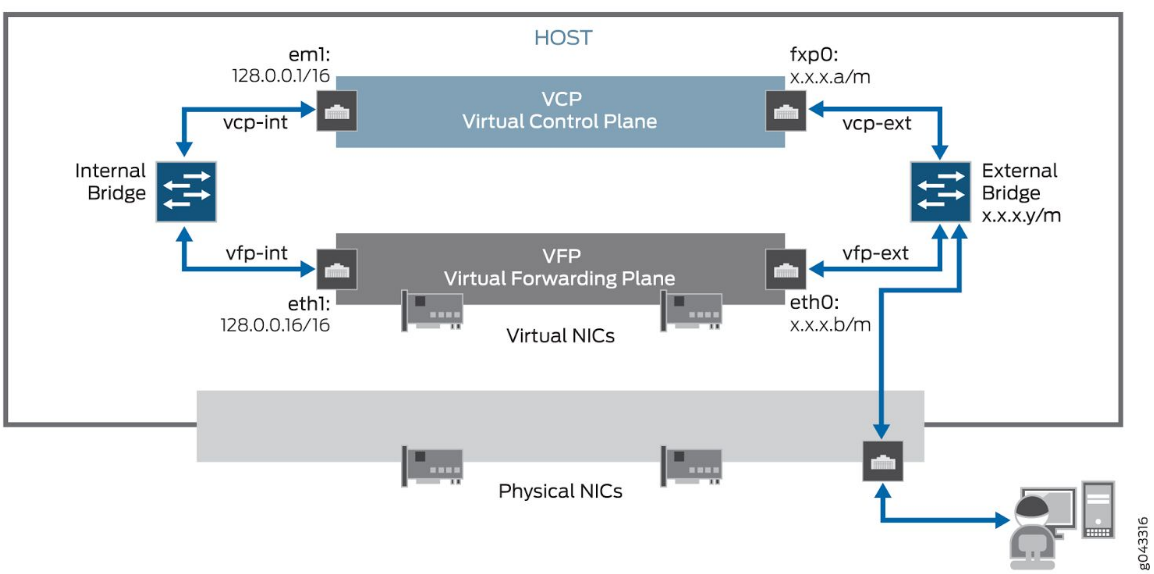


Figure 4 vMX topology on Mirantis OpenStack

5 Installation & Configuration

Hardware Requirements
Below are the hardware requirements to run the vMX image on Mirantis OpenStack.

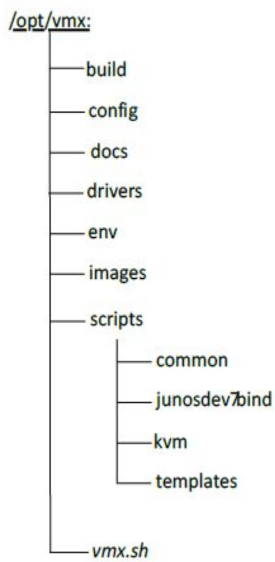
Table 1 - Hardware Configuration

Component	Values
CPU	For lab simulation and low performance (less than 100 Mbps) use cases, any x86 processor (Intel or AMD) with VT-d capability. Be sure to specify the vPFE_lite image in the vmx.conf file. For all other use cases, Intel Ivy Bridge processors or later are required. Example of Ivy Bridge processor: Intel Xeon E5-2667 v2 @ 3.30 GHz 25 MB Cache For Single Root I/O Virtualization (SR-IOV) NIC type, use Intel 82599-based PCI-Express cards (10 Gbps) and Ivy Bridge processors.
Memory	Minimum: 10 GB (2 GB for VCP, 6 GB for VFP, 2 GB for host OS)
Storage	Local or NAS Each vMX instance will require ~1G of disk storage
Other requirements	Hyperthreading: Not recommended NIC: Intel 82599 for 10GBE

Software Requirements
Table 2 - vMX Packaging

Directory / Filename	Description
build/	Location where the different vmx instances are installed by the orchestration scripts
config/	Where the example scripts are located: vmx.conf – Configuration file for defining vMX parameters. vmx-junosdev.conf – vMX interface binding file. All configuration files are in the YAML Format.
docs/	Location of this document
env/	OS environment settings
images/	The VM images are located here: jinstall64-vmx-*.img – Software image file for VCP. vmxhdd.img – Software image file for VCP file storage. vPFE_lite-*.img – Software image file for VFP (lite version). Use this image for lab simulation and low performance (less than 100 Mbps) applications. vPFE_*.img – Software image file for VFP (performance version).
scripts/	Juniper Networks orchestration scripts.
vmx.sh	Main orchestration script.

The directory structure is as follows:



5.1 MOS environment preparation

Compute node must be a bare metal node, as Juniper vMX does not support nested virtualization.

5.1.1 MOS installation

5.1.2 Creation of OpenStack environment

Using Fuel Web UI the following cluster was created:

- OS – Ubuntu 14.04
- Mode - HA
- Hypervisor – KVM
- Networking – Neutron + VLAN/VXLAN
- Storage – any
- Additional services – any

Steps:

1. Download 8.0 MOS ISO from Mirantis website.
2. Boot the Fuel Master node.
3. Create a new OpenStack environment in the following configuration:
    a. KVM as hypervisor
    b. Neutron+VLAN or VXLAN as networking option

5.1.3 Health Check Results

OpenStack Testing Framework (OSTF also known as Health Checks) provides ability to verify the cluster operability in the post-deployment stage. You can find more information about Health Checks in the official Mirantis OpenStack documentation.

5.2 vMX installation steps

1. Please contact with Juniper to obtain vMX images and vMX Heat template.

2. Create Glance images:
```
# glance image-create --min-disk 20 --min-ram 2048 --property hw_cdrom_bus=ide
--property hw_disk_bus=ide --property hw_vif_model=e1000 --file
jinstall64-vmx-15.1F5-S1.5-domestic.img --name VCP
# glance image-create --min-disk 4 --min-ram 8192 --property hw_cdrom_bus=ide
--property hw_disk_bus=ide --property hw_vif_model=virtio --file vFPC-20160503.img
--name VFP
```

3. Create flavors for vMX according to its requirements:
```
# nova flavor-create VCP auto 2048 20 1
# nova flavor-create VFP auto 8192 4 3
```

4. Create required networks and subnets with necessary properties (in our case we've used 4 networks: - vMX_GE000_net, vMX_GE001_net, internal and management):
```
# neutron net-create vMX_public_net
# neutron subnet-create vMX_public_net 10.20.25.0/24 --name vMX_public_net_subnet
--enable-dhcp
# neutron net-create vMX_GE000_net
# neutron subnet-create vMX_GE000_net 10.20.20.0/24 --name vMX_GE000_net_subnet
--enable-dhcp
# neutron net-create vMX_GE001_net
# neutron subnet-create vMX_GE001_net 10.20.21.0/24 --name vMX_GE001_net_subnet
--enable-dhcp
```

   Internal network will be created automatically via Heat.
   Only if you plan to access vMX instances via external network, you must connect vMX public (management) network with OpenStack external network via a router:
```
# neutron router-interface-add router04 vMX_public_net_subnet
```

5. Prepare Heat files:
   a. Create an env file:
```
# cat vmx_heat.env
parameters:
  vmx_ident: vmx1
  hostname_re: vmx_re0
  vmx_vre_name: vmx1_re0
  vmx_vpfe_name: vmx1_fpc
  vmx_oam_network: c98e1253-0891-4308-b7eb-adf7132e16c8
  vmx_vre_img: VCP
  vmx_vpfe_img: VFP
  vmx_vre_flavor: VCP
  vmx_vpfe_flavor: VFP
  ge000_network: 060808d4-050f-487f-a648-c69e8d0955ad
```

ge001_network: 5b95b900-9614-43ae-a3f4-6bb827a691ee

b. Add a management network gateway in a template file, only if you plan to access vMX instances via external network:
In the “instance_vre” section in the “metadata” subsection specify an ip address of vMX_public_net port which is in the router04 (in our case it is 10.20.25.1).

```
 instance_vre:
    type: OS::Nova::Server
...
<cut here>
...
      metadata:
        gateway: 10.20.25.1
```

6. Launch vMX instance:
`# heat stack-create -f ./vmx-heat-mono.yaml -e ./vmx_heat.env  vMX`
In Horizon, open the console and wait until vMX instances are installed (login prompt should appear):
vCP:
```
Wind River Linux 6.0.0.13 host-10-20-25-14 tty0

host-10-20-25-14 login:
```

vFP:
```
vmx_re0 (ttyv0)     BTX version is 1.02

login:
```

Now you can connect to vMX instances by ssh via management network.
If you plan to access vMX instances via external network, you must also associate floating IPs.

6 Basic verification on vMX

Verify MIC and PIC are shown

```
root@vmx_re0> show chassis hardware
Hardware inventory:
Item             Version  Part number  Serial number     Description
Chassis                                VM5758320B7F      VMX
Midplane
Routing Engine 0                                         RE-VMX
CB 0                                                     VMX SCB
CB 1                                                     VMX SCB
FPC 0                                                    Virtual FPC
  CPU            Rev. 1.0 RIOT         123XYZ987
```

```
MIC 0                                              Virtual
  PIC 0                 BUILTIN      BUILTIN          Virtual
```

1. Verify all interfaces show up correctly

```
root@vmx_re0> show interfaces terse
Interface               Admin Link Proto    Local                 Remote
ge-0/0/0                up    up
ge-0/0/0.0              up    up   inet     10.20.20.2/24
                                   multiservice
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-0/0/1.0              up    up   inet     10.20.21.2/24
                                   multiservice
ge-0/0/2                up    down
ge-0/0/3                up    down
ge-0/0/4                up    down
ge-0/0/5                up    down
ge-0/0/6                up    down
ge-0/0/7                up    down
ge-0/0/8                up    down
ge-0/0/9                up    down
cbp0                    up    up
demux0                  up    up
dsc                     up    up
em1                     up    up
em1.0                   up    up   inet     10.0.0.4/8
                                            128.0.0.1/2
                                            128.0.0.4/2
                                   inet6    fe80::fa16:3eff:fec3:a691/64
                                            fec0::a:0:0:4/64
                                   tnp      0x4
esi                     up    up
fxp0                    up    up
fxp0.0                  up    up   inet     10.20.25.15/24
gre                     up    up
ipip                    up    up
irb                     up    up
jsrv                    up    up
jsrv.1                  up    up   inet     128.0.0.127/2
lo0                     up    up
```

```
lo0.16384               up    up   inet     127.0.0.1           --> 0/0
lo0.16385               up    up   inet
lsi                     up    up
mtun                    up    up
pimd                    up    up
pime                    up    up
pip0                    up    up
pp0                     up    up
tap                     up    up
vtep                    up    up
```

2. Verify connectivity between VFP and VCP
   When the VCP and VFP connection is established, *show interfaces terse* command in the VCP CLI displays the ge-0/0/x interfaces and the following messages appear in the VFP syslog file (/var/log/messages):
   RPIO: Accepted connection from 172.16.0.1:50896 <-> vPFE:3000
   RPIO: Accepted connection from 172.16.0.1:56098 <-> vPFE:3000
   HOSTIF: Accepted connection

   The two management interfaces (VCP VM and VFP VM) should be able to reach each other. For example:
   ```
   root@vmx_re0> ping 10.20.25.14
   PING 10.20.25.14 (10.20.25.14): 56 data bytes
   64 bytes from 10.20.25.14: icmp_seq=0 ttl=64 time=2.135 ms
   ```

3. Verify VMs are running
   Verify that the VMs are running after vMX is installed, by using *nova list* command.