vMX Openstack Quick Start Guide

`

Rev #	Date	Revised by	Comments
1.0	12/20/16	Anjali Kulkarni & Pratik Maru	Initial Draft for vMX support on Openstack with Redhat and Ubuntu

Table of Contents

Introduction

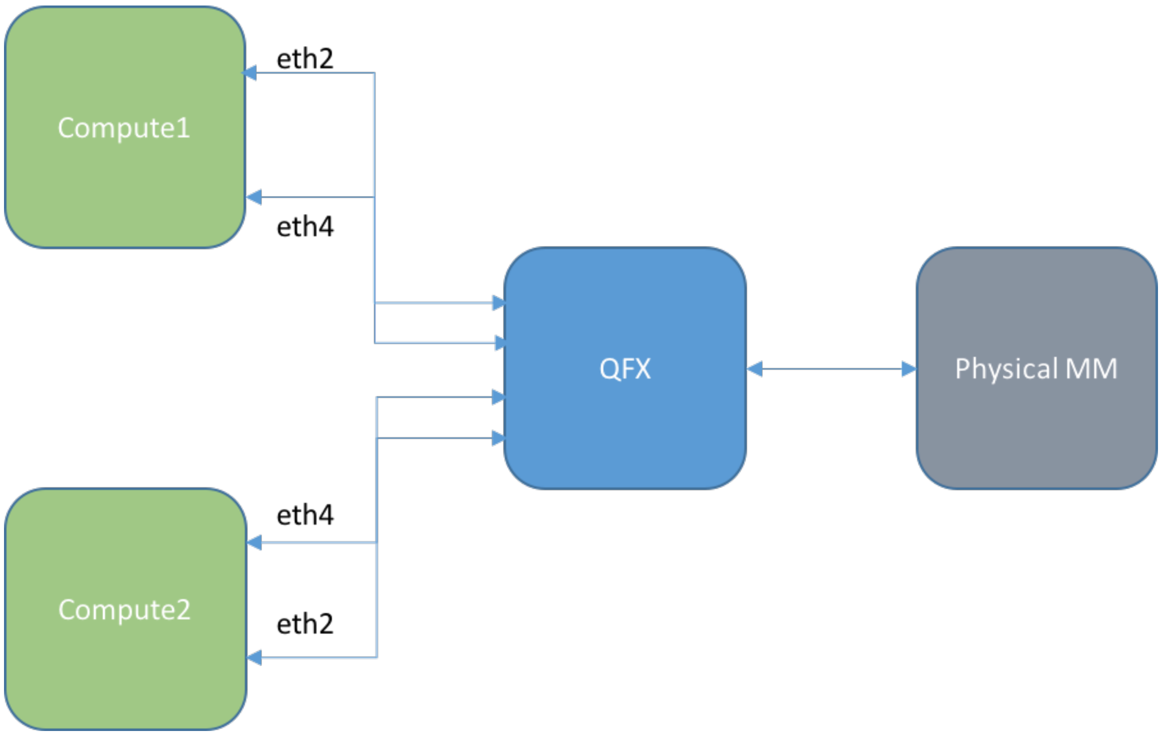
The purpose of this document is help users in launching vMX on RHOSP and Ubuntu Openstack. This guide is applicable to Redhat and Ubuntu, and the differences are noted as appropriately in the steps listed in the guide.

It provides details on openstack configurations required to launch vMX with Virtio and SR-IOV interfaces. It is assumed that basic openstack installation is done on controller and compute nodes.

This document is applicable to vlan provider network configurations. The networks created for traffic flow in virtio and SRIOV modes are vlan provider networks. See Ref. [1] for more details on how to configure provider networks. In provider networks, forwarding is done by a combination of software and hardware (which in our case is a QFX switch).

Topology Diagram

Below is the topology diagram and sample configuration for our testing. Each compute node has 2 10G NIC cards for traffic (The management traffic will be on a separate network not shown bellow).
Out of the two NIC cards, one of them will be used for VirtIO networks and another one will be used for SRIOV NIC. If we need more than one SRIOV nic card, then number of physical Ethernet cards required for VM traffic = **1 + No. of SRIOV nics required**.



In above diagram, eth2 is used for all the VirtIO networks and eth4 is used for SRIOV networks. RE-PFE connection is on the VirtIO network. Also, if 2 vmx's are connected to each other via a Virtio port, it will use the eth2 interface.
For our setup, thus, we will have 2 physnet networks defined in openstack - physnet1 is for Virtio connectivity, whereas physnet2 is for SRIOV connections.
There is also a Controller node, not shown in the diagram, which is connected via eth2 to the QFX switch.

Pre-Installation Steps and Configurations

Configurations on Controller Node

Configure below on Controller Node of Openstack:-

1. Follow Steps below 1-4 for vMX operation.
2. Follow step 5 if using vlan provider OVS networks for virtio networks. More information in Step 5.
3. Follow steps 6-7 if you need to configure SRIOV. Steps 6-7 can be skipped, if SRIOV functionality is not needed.

1. Nova Configuration for Pinning/Hugepages/Affinity

To enable pinning and hugepages we need following configs on controller node (the ones in Red are added). We recommend this config be added irrespective of whether pinning is on or off (Hugepages is always on)

In /etc/nova/nova.conf, append the lines in red to your entry of scheduler_default_filters:

```
scheduler_default_filters=RetryFilter,AvailabilityZoneFilte
r,RamFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePr
opertiesFilter,CoreFilter,NUMATopologyFilter,AggregateInsta
nceExtraSpecsFilter, PciPassthroughFilter,
ServerGroupAffinityFilter, ServerGroupAntiAffinityFilter
```

Then do:

Redhat:

```
     systemctl restart openstack-nova-scheduler.service
```

Ubuntu:

```
     service nova-scheduler restart
```

2. Update Openstack defaults:

Need to update the defaults as:

```
nova quota-class-update --cores 100 default
nova quota-class-update --ram 102400 default
nova quota-class-update --instances 100 default
```

Check that the defaults are updated, using “`nova quota-defaults`”

3. Heat version to resolve Liberty bug

To avoid hitting the bug shown below during stack-creation for nested heat templates:
https://bugzilla.redhat.com/show_bug.cgi?id=1327080
https://bugzilla.redhat.com/show_bug.cgi?id=1330384
Update the heat package as shown below, if necessary:

Redhat:

Before updating, the older heat version is:

```
[root@controller vmx_templates]# rpm -qa | grep heat
openstack-heat-engine-5.0.1-5.el7ost.noarch
openstack-heat-api-5.0.1-5.el7ost.noarch
python-heatclient-1.0.0-1.el7ost.noarch
openstack-heat-common-5.0.1-5.el7ost.noarch
```

Then to upgrade, do:

```
yum update openstack-heat-engine
```

After upgrading, you will see (-5 changed to –6):

```
[root@controller vmx_templates]# rpm -qa | grep heat
openstack-heat-common-5.0.1-6.el7ost.noarch
openstack-heat-engine-5.0.1-6.el7ost.noarch
python-heatclient-1.0.0-1.el7ost.noarch
openstack-heat-api-5.0.1-6.el7ost.noarch
```

This new package is part of rhel-7-server-openstack-8-rpms

Ubuntu:

4. Other Packages

Make sure the following packages are installed:

Redhat:

```
yum install redhat-lsb-core yum install numactl
```

Ubuntu:

```
apt-get install lsb-release
apt-get install numactl
```

Configuration steps to use Vlan Provider OVS Networks for Virtio network on Controller Node

Note, this section is applicable only if you are using vlan provider network for Virtio networks (for example if the communication between RE and PFE which done via virtio network is done on a vlan-provider OVS network). This section lists only the relevant configurations. For detailed configuration, look at:

Below steps assume that each compute has minimum two NIC cards dedicated for VM traffic. Out of the two nic cards, one of them will be used for VirtIO networks and another one will be used for SRIOV nic.

5. Virtio configuration for vlan provider networks on 10G NIC interface eth2.

On the Controller node:

- /etc/neutron/plugins/ml2/ml2_conf.ini, enable the vlan mechanism driver by adding vlan to the existing list of values:
  ```
  [ml2]
  type_drivers = vxlan,flat,vlan
  ```

- Add bridge mappings in/etc/neutron/plugins/ml2/ml2_conf.ini by adding the following line:
  ```
  bridge_mappings =physnet1:br-vlan
  ```

- Configure the network_vlan_ranges setting to reflect the physical network and VLAN ranges in use for physnet1, which is our virtio network:
  ```
  [ml2_type_vlan]
  network_vlan_ranges =physnet1:2100:2198
  ```

- After doing the above changes, restart neutron server:
  Redhat:
  ```
      systemctl restart neutron-server
  ```
  Ubuntu:
  ```
      service neutron-server restart
  ```

- An OVS bridge, say “br-vlan” is added. (This is the same “br-vlan” which was added in bridge_mappings in ml2_conf.ini above on controller). To this bridge, the eth2 interface, which will be used for Virtio communication between VMs, is added. This is shown below:
  ```
   ovs-vsctl add-br br-vlan
    ovs-vsctl add-port br-vlan eth2
  ```

Additional configuration steps to enable SRIOV on Controller Node

Follow steps 6-7 only if SRIOV is required. Below steps assume that each compute has minimum two NIC cards dedicated for VM traffic.

Out of the two nic cards, one of them will be used for VirtIO networks and another one will be used for SRIOV nic. If we need more than one SRIOV nic card, then number of physical Ethernet cards required for VM traffic = **1 + No. of SRIOV nics required**.

6. Neutron configuration for SRIOV

- Edit /etc/neutron/plugins/ml2/ml2_conf.ini to add "sriovnicswitch" as mechanism drivers and "physnet2" as another network_vlan_ranges.

```
# mechanism_drivers =
mechanism_drivers =openvswitch,sriovnicswitch

[ml2_type_vlan]
network_vlan_ranges =physnet1:2100:2198,physnet2:1500:1505
```

- Edit /etc/neutron/plugins/ml2/ml2_conf_sriov.ini and details about PCI devices.

```
[ml2_sriov]
supported_pci_vendor_devs = 8086:10ed
```

Redhat:

- Edit /usr/lib/systemd/system/neutron-server.service, add below highlighted line.

```
ExecStart=/usr/bin/neutron-server --config-file
/usr/share/neutron/neutron-dist.conf --config-dir
/usr/share/neutron/server --config-file
/etc/neutron/neutron.conf --config-file
/etc/neutron/plugin.ini --config-dir
/etc/neutron/conf.d/common --config-dir
/etc/neutron/conf.d/neutron-server --config-file
/etc/neutron/plugins/ml2/ml2_conf_sriov.ini --log-file
/var/log/neutron/server.log
```

Restart the service:

```
systemctl restart neutron-server
```

Ubuntu:

- Edit /etc/init/neutron-server.conf, add below highlighted line.

```
script
  [ -x "/usr/bin/neutron-server" ] || exit 0
  [ -r /etc/default/openstack ] && .
/etc/default/openstack
  [ -r /etc/default/neutron-server ] && .
/etc/default/neutron-server
  [ -r "$NEUTRON_PLUGIN_CONFIG" ] &&
DAEMON_ARGS="$DAEMON_ARGS --config-
file=$NEUTRON_PLUGIN_CONFIG  --config-
file=/etc/neutron/plugins/ml2/ml2_conf_sriov.ini"
```

Restart the service:

```
service neutron-server restart
```

7. Scheduler configuration for SRIOV

- To allow proper scheduling of SR-IOV devices, the Compute scheduler needs to use FilterScheduler with the PciPassthroughFilter filter. Apply this configuration in the nova.conf file on the Controller node, if is not already present.

- Restart the scheduler service

  Redhat:

  ```
  systemctl restart openstack-nova-scheduler
  ```

  Ubuntu:

  ```
  service nova-scheduler restart
  ```

Configurations on Compute Node

Follow steps below 1-3, on all Compute Nodes of Openstack. Follow step 4 if using vlan provider OVS networks for virtio networks. If SRIOV functionality is desired, follow steps 5-8.

1. Nova configuration to support metadata for vMX

Note, this is not required from release <> onwards.

For vMX to get metadata correctly, we need:

On compute nodes, in /etc/nova/nova.conf:

```
config_drive_format=vfat
```

Then do:

Redhat:

```
systemctl restart openstack-nova-compute
```

Ubuntu:

```
service nova-compute restart
```

2. Boot configuration for Hugepages

We configure hugepages at boot time, on each compute node as follows (you need to reboot after this step):

Redhat:

```
grubby --update-kernel=ALL --args="hugepagesz=2M
hugepages=24576”
grub2-install /dev/sda
reboot
```

Ubuntu:

Add “hugepagesz=2M hugepages=24576“ configuration into /etc/default/grub, under

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash
intel_iommu=on hugepagesz=2M hugepages=24576"
update-grub
reboot
```

After reboot, check Hugepages allocated via:
cat /proc/meminfo | grep Huge

3. Enable IOMMU

- Add "intel_iommu=on " configuration into /etc/default/grub, under

  ```
  GRUB_CMDLINE_LINUX="crashkernel=auto
  rd.lvm.lv=rhel_compute01/root rd.lvm.lv=rhel_compute01/swap
  biosdevname=0 net.ifnames=0 rhgb quiet intel_iommu=on"
  ```

- Regenerate the grub file

  Redhat:
  ```
  grub2-mkconfig -o /boot/grub2/grub.cfg
  ```
  Ubuntu:
  ```
  update-grub
  ```

- Reboot the compute node.

Configuration steps to use Vlan Provider OVS Networks for Virtio network on Compute Node

Note, this section is applicable only if you are using vlan provider network for Virtio networks (for example if the communication between RE and PFE which done via virtio network is done on a vlan-provider OVS network). This section lists only the relevant configurations on Compute. For detailed configuration, look at:

Below steps assume that each compute has minimum two NIC cards dedicated for VM traffic.

Out of the two nic cards, one of them will be used for VirtIO networks and another one will be used for SRIOV nic. If we need more than one SRIOV nic card, then number of physical Ethernet cards required for VM traffic = **1 + No. of SRIOV nics required**.

4. Add bridge for Virtio network, and configure physnet1:

- An OVS bridge, say "br-vlan" is added. (This is the same "br-vlan" which was added in bridge_mappings in ml2_conf.ini above on controller). To this bridge, the eth2 interface, which will be used for Virtio communication between VMs, is added. This is shown below:
  ```
   ovs-vsctl add-br br-vlan
    ovs-vsctl add-port br-vlan eth2
  ```

- In /etc/neutron/plugins/ml2/openvswitch_agent.ini, add yellow highlighted part:
  ```
  bridge_mappings =public_physnet:br-public,physnet1:br-vlan
  ```

- Restart neutron service

- Redhat:
```
     systemctl restart neutron-server.service
     systemctl restart neutron-openvswitch-agent.service
     systemctl restart nova-compute
```

  Ubuntu:
```
   service nova-compute restart
   service neutron-plugin-openvswitch-agent restart
```

Additional configuration steps to enable SRIOV on Compute Node

Follow steps 5-8 on Compute node, only if SRIOV is being used.

5. Load Customized IXGBE driver

Before compiling the driver, make sure gcc and make are installed:

Redhat:
```
     sudo yum install make gcc
     sudo yum install kernel kernel-devel
```

Ubuntu:
```
     sudo apt-get update
     sudo apt-get install make gcc
```

Unload default ixgbe driver, compile custom Juniper driver, and load it:

- `Untar the vmx-bundle.`
- `cd <vmx_bundle_file_system>/drivers/ixgbe-3.19.1/src`
- `make`
- `rmmod ixgbe`
- `insmod  ./ixgbe.ko`
- `make install`

Driver Verification

```
[root@compute01 src]# ethtool -i eth2
driver: ixgbe
version: 3.19.1
firmware-version: 0x800002d8
…

[root@compute01 src]# ethtool -i eth4
driver: ixgbe
version: 3.19.1
firmware-version: 0x800002d8
…
```

6. Create VFs on physical device
   - `echo '1' > /sys/class/net/eth4/device/sriov_numvfs`

In above setup, we need one VF for eth4 (since it is used for SRIOV traffic), which is the value put in sriov_numvfs. Note, currently vMX supports only 1 VF per NIC.

Verification for VF creation (see the text in bold)

```
root@compute01 src]# ip link show eth2
18: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop master ovs-
system state DOWN mode DEFAULT qlen 1000
    link/ether 08:9e:01:82:ab:38 brd ff:ff:ff:ff:ff:ff

[root@compute01 src]# ip link show eth4
19: eth4: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
mode DEFAULT qlen 1000
    link/ether 08:9e:01:82:ab:39 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, spoof checking on, link-state
auto
[root@compute01 src]#
```

Post VF creation commands

Execute following commands to make the interfaces up and ensure that SRIOV traffic passes through it
Please note the all the below commands on “eth4” are important for SRIOV to work. Skipping any of these may result in some or the other issues.

```
[root@compute01 src]# ifconfig eth2 up

[root@compute01 src]# ifconfig eth4 up
[root@compute01 src]# ifconfig eth4 promisc
[root@compute01 src]# ifconfig eth4 allmulti
[root@compute01 src]# ifconfig eth4 mtu 9192
[root@compute01 src]# ip link set eth4 vf 0 spoofchk off

[root@compute01 src]# ip link show eth4
19: eth4: <BROADCAST,MULTICAST,ALLMULTI,PROMISC,UP,LOWER_UP> mtu
9192 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether 08:9e:01:82:ab:39 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, spoof checking off, link-state
auto
```

7. SR-IOV Agent

- Install SR-IOV agent
  Redhat:
  ```
  yum install openstack-neutron-sriov-nic-agent
  ```
  Ubuntu:
  ```
  apt-get install neutron-plugin-sriov-agent
  ```

- Edit /etc/neutron/plugins/ml2/ sriov_agent.ini, and add entry for physical_device_mappings

  ```
  [sriov_nic]
  physical_device_mappings = physnet2:eth4
  ```

- Edit SRIOV agent service file
  Redhat:
  - Edit /usr/lib/systemd/system/neutron-sriov-nic-agent.service, and add highlighted text in yellow below:

    ```
    ExecStart=/usr/bin/neutron-sriov-nic-agent --config-
    file /usr/share/neutron/neutron-dist.conf --config-
    file /etc/neutron/neutron.conf --config-file
    /etc/neutron/plugins/ml2/sriov_agent.ini --config-dir
    /etc/neutron/conf.d/common
    ```

  - Enable and start sriov-agent
    ```
    # systemctl enable neutron-sriov-nic-agent.service
    # systemctl start neutron-sriov-nic-agent.service
    ```

  - Verify if the agent has successfully started
    ```
    # systemctl status neutron-sriov-nic-agent.service
    ```

  Ubuntu:
  - Edit /etc/init/neutron-plugin-sriov-agent.conf, make sure the text highlighted in yellow is present (if not already present, add it)

    ```
    script
      [ -x "/usr/bin/neutron-sriov-nic-agent" ] || exit 0
      DAEMON_ARGS="--config-
    file=/etc/neutron/plugins/ml2/ml2_conf_sriov.ini --
    config-file=/etc/neutron/plugins/ml2/sriov_agent.ini"
      [ -r /etc/default/openstack ] && .
    /etc/default/openstack
    ```

```
[ -r /etc/default/$UPSTART_JOB ] && .
```

- Ensure that '/etc/neutron/plugins/ml2/sriov_agent.ini' has correct permissions and group of the file is 'neutron'.

  ```
  -rw-r--r-- 1 root neutron 1107 Dec 15 16:01
  sriov_agent.ini
  ```

- Start sriov-agent
  ```
  # service neutron-plugin-sriov-agent start
  ```

- Verify if the agent has successfully started
  ```
  # service neutron-plugin-sriov-agent status
  ```

8. Nova Changes

- Edit /etc/nova/nova.conf, add pci_passthrough_whitelist entry for the SRIOV device:

  ```
  #pci_passthrough_whitelist =
  pci_passthrough_whitelist = {"devname":"eth4",
  "physical_network": "physnet2"}
  ```

  Here physnet2 is the physical network we are using, and eth4 is physical NIC card, which we are planning to use for SRIOV. Please change according to your setup.

- Restart nova-compute service

  Redhat:
  ```
      systemctl restart openstack-nova-compute
  ```
  Ubuntu:
  ```
      service nova-compute restart
  ```

vMX installation Steps

After following openstack configuration steps above, we now need to create nova flavors and glance images for vRE and vPFE components of vMX. Steps to do the same are listed below. For simplicity, scripts are provided to create the flavors and images, with options to use high performance features such as pinning and hugepages. The names, configuration and other details are taken as input in a configuration file vmx.conf, which is then provided as input to the scripts, to create flavors/images as shown below.

To download the scripts and heat templates, do:

```
git clone git@git.juniper.net:anjali/openstack-heat.git
cd openstack-heat/openstack
```

Flavors

All files referred to below are present under `openstack-heat/openstack`
To create flavors, vMX package provides a configuration file "`openstack-heat/openstack /scripts/vmx.conf`", which has to be correctly setup as per the requirement.

This configuration file will be provided as input to another script "`openstack-heat/openstack/scripts/vmx_osp_create_flavor.py`" , which upon execution will generate a file, which will consists of complete set of commands to create flavors.

Execution of this script, will create flavors.

Example vMX.conf

```
# cat vmx.conf
# vmx.conf
# Configuration file for Flavors

HOST:
    virtualization-type      : openstack
    cpu-pinning              : on
    #compute                 : <compute1>, <compute2>

---
#vRE VM parameters
CONTROL_PLANE:
    re-flavor-name           : re-test
    vcpus                    : 2
    memory-mb                : 4096
```

```
---
#vPFE VM parameters
FORWARDING_PLANE:
    pfe-flavor-name           : pfe-test
    memory-mb                 : 12288
    vcpus                     : 7
```

Explanation for vMX.conf file

cpu-pinning

This option is provided to enable or disable cpu-pinning feature for flavors. By default, vMX will use cpu-pinning, and it is recommended to always keep this knob on.

compute (optional)

This field specifies the hostnames of the compute servers, on which to run the vMX instances. vMX launched with flavors, created with specific set of compute nodes, will only be spawned on the above specified compute hosts. Thus, it can be used to run the vMX instances on specified subset of compute hosts, rather than on any compute host (as it would be if this field is omitted).
You can give a comma-separated list of compute server names (not IP), to indicate the compute field.

If this knob is not provided, flavor will use the output of “nova hypervisor-list” to get the list of compute nodes.

Minimum requirements for vRE and vPFE in vmx.conf:

vRE
2 vCPU, 4G memory given. Note, for Ubuntu, we need to give 1vCPU, greater than 1vCPU may run into some issues.

vPFE:
For Performance mode:
Min 7 vCPU
Min 12 G memory
If less than 7 vCPU are given to vPFE, then it will automatically switch to lite mode.

How to use the script

Execute vmx_osp_create_flavor.py with vmx.conf

```
# ./vmx_osp_create_flavor.py vmx.conf

Openstack Flavor Creation

Generating Flavor for vRE
Generating Flavor for vPFE
```

Upon completion it will generate a file "vmx_osp_flavors.sh" with complete set of commands required for flavor creation.

Execute this generated file and flavors will be created:

```
# sh vmx_osp_flavors.sh
```

Images

To install the vMX Images, you can use the script "`openstack-heat/openstack/scripts/vmx_osp_images.sh`" provided under vMX package.

This script will add vRE image in qcow2 format with virtio vif_mode, whereas for vPFE image it will use vmdk format with virtio vif_mode.

How to use the script

To use the script you have to provide four options in below listed order

- RE Image Name
- RE Image Location
- PFE Image Name
- PFE Image Location

# sh vmx_osp_images.sh <re_image_name> <re_image_location> <pfe_image_name> <pfe_image_location>

```
# sh vmx_osp_images.sh re-test /var/tmp/junos-vmx-x86-64-15.1F-
20160817.0.qcow2 fpc-test /var/tmp/vFPC-20160810.img
```

Junos Configuration File

This is the junos configuration file loaded on boot. It is referenced in : `openstack-heat/openstack/vmx-components/vms/re.yaml`
If you need to add any configuration on junos, for example, the address on ge-0/0/0, then that address modification can be done in this file, before starting the vMX.

Either use vmx_baseline.conf provided in sources (kept in `openstack-heat/openstack/vmx-components/vms/ vmx_baseline.conf`), OR create the file vmx_baseline.conf with the text shown below. Make sure `openstack-heat/openstack/vmx-components/vms/re.yaml` references the path to your vmx_baseline.conf

```
groups {
    re0 {
        system {
            host-name %hostname%;
            backup-router %gateway%;
        }
        interfaces {
            fxp0 {  # Management/telnet Interface
                unit 0 {
                    family inet {
                        address %re0_ip%/%netmask%; #
Management/telnet address
                    }
                }
            }
        }
    }
    re1 {
        system {
            host-name %hostname%1;
            backup-router %gateway%;
        }
        interfaces {
            fxp0 {  # Management/telnet Interface
                unit 0 {
                    family inet {
                        address %re1_ip%/%netmask%; #
Management/telnet address
                    }
                }
            }
        }
    }
    global {
        system {
            debugger-on-panic;
            debugger-on-break;
            dump-on-panic;
            services {
                finger;
                ftp;
                rlogin;
                rsh;
                ssh;
                telnet;
                xnm-clear-text;
            }
            syslog {
                host log {
                    kernel info;
                    any notice;
                    pfe info;
                    interactive-commands any;
                }
                file messages {
                    kernel info;
                    any notice;
                    authorization info;
                    pfe info;
                    archive world-readable;
                }
                file security {
                    interactive-commands any;
                    archive world-readable;
                }
            }
            processes {
                routing enable;
                ntp enable;
                management enable;
                watchdog enable;
                snmp enable;
                inet-process enable;
                mib-process enable;
            }
        }
        chassis {
            dump-on-panic;
        }
        interfaces {
            lo0 {     # Local Loopback interface.
                unit 0 {
                    family inet {
                        address %lo0-ip%/32 {
                            primary;
                        }
                    }
                    family iso {
```

```
                        address %lo0-iso%;
                    }
                    family inet6 {
                        address %lo0-inet6%/128 {
                            primary;
                        }
                    }
                }
            }
        }
        snmp {
            interface fxp0.0;
            community public {
                authorization read-only;
            }
            community private {
                authorization read-write;
            }
        }
        routing-options {
            router-id %router-ip%;
      }
    }
}
apply-groups [ global re0 re1];
system {
    ports {
        console log-out-on-disconnect;
    }
}
```