vMX Contrail Openstack Quick Start Guide

`

Rev #	Date	Revised by	Comments
1.0	02/14/17	Anjali Kulkarni & Pratik Maru	Initial Draft for vMX support on Contrail based Openstack on Ubuntu

Table of Contents

Introduction

The purpose of this document is help users in launching vMX on Ubuntu Contrail based Openstack distribution. This guide is applicable to Liberty release of Openstack for both DPDK and Non-DPDK enabled vRouter, 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 SR-IOV using VFD daemon (written by ATT). It is assumed that basic contrail based openstack installation is done on controller and compute nodes, and those installation steps are not covered in this guide.

Specific installation details for DPDK enabled vRouter setup

Please keep in mind following points while installing the setup with DPDK enabled vRouter, although various configuration requirements are also added at appropriate places.

- Additional configuration in testbed.py file to install DPDK based vRouter. Details can be found at this link [dpdk-with-vrouter]

```
env.dpdk = {
    host2: { 'huge_pages' : '50', 'coremask' : '0xf'},
    host3: { 'huge_pages' : '50', 'coremask' : '0xf'},
}
```

- Enabled iommu=pt in /etc/default/grub on compute node.

- If using contrail-3.0.3.0-69 build. On computes, libvirt is not upgrading to required version even though deb packages are in contrail local repo. We need libvirtd –version libvirtd (libvirt) 1.2.16 but compute will come up with version 1.2.12. The fix for this problem is upgrade these packages manually with below commands and restart nova-compute and libvirt-bin services:

```
dpkg -i /opt/contrail/contrail_install_repo/libvirt-bin_1.2.16-
2ubuntu11.15.10.4~cloud0_amd64.deb
dpkg -i /opt/contrail/contrail_install_repo/libvirt0_1.2.16-
2ubuntu11.15.10.4~cloud0_amd64.deb
```

- All the VM's should have huge pages enabled, otherwise VM won't be able to transmit traffic.
- Verify if DPDK is enabled on compute nodes

```
:~# contrail-status
== Contrail vRouter ==
supervisor-vrouter:           active
contrail-vrouter-agent        active
contrail-vrouter-dpdk         active
contrail-vrouter-nodemgr      active
```

- Ensure the NIC cards used for contrail-vhost network (network which connect various compute and controller nodes) and SRIOV network (one which will be provided input to vfd) are from different network cards.

```
For example, below vhost0 and eth3 are from same parent network
```

card (0000:06:00.*), whereas eth1 is from different card. So ideally interface from same parent card should not be used for vhost and sriov(vfd)

```
# ethtool -i vhost0
driver: ixgbe
version: 3.9.17-NAPI
firmware-version: 0x800002b3
bus-info: 0000:06:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: no

# ethtool -i eth3
driver: ixgbe
version: 3.15.1-k
firmware-version: 0x800002b3
bus-info: 0000:06:00.1
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: no

# ethtool -i eth1
driver: igb
version: 5.0.5-k
firmware-version: 1.63, 0x80000a05
bus-info: 0000:03:00.1
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: no
```

Pre-Installation Steps and Configurations

Configurations on Controller Node

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:

```
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`"

Additional configuration steps to enable SRIOV on Controller Node

3. Apply patch

Patch is available under latest vmx- bundle. Download the bundle and untar the patch. Patch will available under openstack/kilo/openstack_vfd_patches

Execute below commands on compute node to apply the patch.

```
cd /
patch -p1 < $path/openstack/kilo/openstack_vfd_patches/heat.patch
patch -p1 < $path/openstack/kilo/openstack_vfd_patches/neutron.patch
```

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:

```
service nova-compute restart
```

2. Boot configuration for Hugepages and IOMMU

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

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

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash intel_iommu=on
iommu=pt hugepagesz=2M hugepages=24576
default_hugepagesz=2M"
```

Please follow below steps to ensure that VMs with hugepages enabled gets successfully deployed, otherwise you may hit below mentioned error:

"unable to create backing store for hugepages: Permission denied"

Above error will be seen in /var/log/nova/nova-compute.log on compute nodes. To avoid this error, please follow below steps

- Create a new directory for libvirt to mount/use huge pages

  ```
  mkdir -p /run/hugepages/kvm/
  ```

- Mount huge table fs over this directory

```
mount -t hugetlbfs hugetlbfs-kvm /run/hugepages/kvm/
```

- Provide this directory input explicitly in etc/libvirt/qemu.conf as mount location for hugetlbfs

```
# cat /etc/libvirt/qemu.conf | grep hugetlbfs
# If provided by the host and a hugetlbfs mount point is
configured,
#     hugetlbfs_mount = ["/dev/hugepages2M", "/dev/hugepages1G"]
hugetlbfs_mount = "/run/hugepages/kvm"
```

- Add "KVM_HUGEPAGES=1" in /etc/default/qemu-kvm

```
# cat /etc/default/qemu-kvm
# Set to 1 to enable KSM, 0 to disable KSM, and AUTO to use
default settings.
# After changing this setting restart the qemu-kvm service.
KSM_ENABLED=AUTO
SLEEP_MILLISECS=200
# To load the vhost_net module, which in some cases can
speed up
# network performance, set VHOST_NET_ENABLED to 1.
VHOST_NET_ENABLED=0

# Set this to 1 if you want hugepages to be available to
kvm under
# /run/hugepages/kvm
KVM_HUGEPAGES=1
```

- Reboot the compute node.

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

```
# cat /proc/meminfo | grep Huge
AnonHugePages:     49152 kB
HugePages_Total:   24576
HugePages_Free:    24576
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

3. Enable IOMMU

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

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash intel_iommu=on
iommu=pt hugepagesz=2M hugepages=24576
default_hugepagesz=2M"
"
```

4. Enable IOMMU=pt

If you are running with DPDK enabled vRouter, add this one more variable in default command line.

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash intel_iommu=on
iommu=pt hugepagesz=2M hugepages=24576
default_hugepagesz=2M"
```

Please note, this is only needed, when running in DPDK enabled vRouter environment.

After editing the grub file, execute below commands for it to take effect.

```
update-grub
reboot
```

Additional configuration steps to enable SRIOV on Compute Node

5. Enabled VT-d in BIOS.

Make sure vt-d is enabled in BIOS

6. Enable ASPM in Bios

Make sure ASPM is enabled in BIOS

To verify if it is enabled

```
# lspci -vv | grep ASPM | grep Enabled
             LnkCtl:      ASPM L1 Enabled; RCB 64 bytes Disabled- CommClk+
             LnkCtl:      ASPM L1 Enabled; RCB 64 bytes Disabled- CommClk+
```

7. Apply patch

Patch is available under latest vmx- bundle. Download the bundle and untar the patch. Patch will available under openstack/kilo/openstack_vfd_patches

Execute below commands on compute node to apply the patch.

```
cd /
patch -p1 < $path/openstack/kilo/openstack_vfd_patches/nova.patch
patch -p1 < $path/openstack/kilo/openstack_vfd_patches/sriov.patch
```

8. Unload ixgbevf driver.

Execute below command to unload ixgbevf driver.

```
# modprobe -r ixgbevf
```

9. Load vfio-pci kernel module

Execute below command to load vfio-pci module

```
# modprobe vfio-pci
```

10. Create VFs on physical device

```
# echo '32' > /sys/class/net/eth4/device/sriov_numvfs
```

In above setup, we need only one VF for eth4 (since it is used for SRIOV traffic), but still we are creating 32 VF's, this is due to a known ATT-VFD issues. Please Note, though we have created 32 VF's but we will be using only one.

Verification for VF creation

```
:~# lspci -nn | grep Ether |grep Virtual | wc -l
32
:~#
```

*We have assumed that we have only SRIOV interface on which we have created the VFs.

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 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
```

*Only one VF, i.e. VF 0 (which is of interest) is shown here, but in original output 31 more VFs will be available.

11. Nova Changes

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

```
     #pci_passthrough_whitelist =
pci_passthrough_whitelist = { "address": "0000:04:10.0",
"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.
Note, we have created 32 VFs but we are only going to push the address of VF 0, which we intend to use in out setup.

- Add VF-agent entry also in nova.conf

```
[vf_agent]
use_vf_agent=True
```

- Restart nova-compute service

```
service nova-compute restart
```

12. VFD Agent

- Install ATT-VFD debian package

Execute below commands to install vfd debian package

```
apt-get update
apt-get install python-docopt
apt-get clean
apt-get update
dpkg -i attlrvfd_1.0j-1_amd64.deb
```

- Copy vfd.cfg from sample file and edit as per below insttructions.

```
# cp /etc/vfd/vfd.cfg.sample /etc/vfd/vfd.cfg
```

Sample vfd.cfg

```
~# cat /etc/vfd/vfd.cfg.sample
{
     "comments": [
```

```
            "sample conf file, fill the pciids",
            "cp vfd.cfg.sample vfd.cfg",
            "initctl start vfd",
            "default_mtu is used if mtu is omitted from a
pciid object."
         ],

         "fifo": "/var/lib/vfd/request",
         "log_dir": "/var/log/vfd",
         "log_keep": 60,
         "init_log_level": 2,
         "log_level": 2,
         "config_dir": "/var/lib/vfd/config",
         "cpu_mask": "0x01",
         "dpdk_log_level": 2,
         "dpdk_init_log_level": 8,
         "default_mtu": <value>,

         "pciids": [
             { "id": "<pciid1>" },
             { "id": "<pciid2>", "mtu": <value> },
             ....
         ]
}
```

- Edited vfd.cfd file

```
~# cat /etc/vfd/vfd.cfg
{
     "comments": [
            "sample conf file, fill the pciids",
            "cp vfd.cfg.sample vfd.cfg",
            "initctl start vfd",
            "default_mtu is used if mtu is omitted from a
pciid object."
         ],

         "fifo": "/var/lib/vfd/request",
         "log_dir": "/var/log/vfd",
         "log_keep": 60,
         "init_log_level": 2,
         "log_level": 2,
         "config_dir": "/var/lib/vfd/config",
         "cpu_mask": "0x01",
         "dpdk_log_level": 2,
         "dpdk_init_log_level": 8,
         "default_mtu": 9001,
```

```
            "pciids": [
                { "id": "0000:04:00.0" }
            ]
}
```

Please note the changes in default_mtu value and pci-id's i.e. ids of PF's.

- Start VFD daemon

```
# service vfd start
```

- Verify if the daemon has successfully started

```
~# service vfd status
vfd start/running, process 47339
```

- Verify if the PF's has been shown under iplex output and link for PF is up.

```
~# iplex show all
{ "state": "OK", "msg": "
PF/VF  ID    PCIID           Link      Speed     Duplex    RX pkts   RX
bytes  RX errors RX dropped    TX pkts   TX bytes  TX errors    Spoofed
pf      0    0000:04:00.0    UP        10000          1        127
41021          0          0          0          0          0          0
```

13. Libvirt Changes

Ensure correct permissions are provided for VFIO

- Edit /etc/libvirt/qemu.conf, under "cgroup_device_acl = [", add below entry

```
"/dev/vfio/vfio"
```

The cgroup device should look like below

```
cgroup_device_acl = [
    "/dev/null", "/dev/full", "/dev/zero",
    "/dev/random", "/dev/urandom",
    "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
    "/dev/rtc",
"/dev/hpet","/dev/net/tun","/dev/vfio/vfio",
]
```

- Restart libvirtd:

```
service libvirt-bin 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.

Flavors

To create flavors, vMX package provides a configuration file "orch/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 "orch/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                : 2048

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

Note: Occam (BSD10) based images will not come up with multicores on Ivybridge machines. It is known issue and more details are available under provided bug link: https://bugs.launchpad.net/qemu/+bug/1329956

Two possible work arounds are possible for this problem:

1. Use only single core for Occam based RE images.
2. Add line “options kvm-intel enable_apicv=N “ in file "/etc/modprobe.d/kvm-intel.conf" on compute node and reboot the node.

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

Always needs 2 vCPU, 4G memory

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
```

Note: If running under DPDK enabled vRouter environment, enabled huge pages for vRE flavor as well.

```
# nova flavor-key  re-test set hw:mem_page_size=2048
```

Images

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

This script will add vRE image in qcow2 format with e1000 vif_mode, whereas for vPFE image it wil 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 : scripts/openstack/vmx_templates/liberty/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 scripts/openstack/vmx_templates/liberty), OR create the file vmx_baseline.conf with the text shown below. Make sure scripts/openstack/vmx_templates/liberty/re.yaml references the path to your vmx_baseline.conf (in latest release, the vmx_baseline.conf in re.yaml references to the vmx_baseline.conf in the same directory which is scripts/openstack/vmx_templates/liberty, in older releases it is /var/tmp/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;
    }
}
```