FortiGate-VM on OpenStack - Installation Guide

Version 6.0.1

FORTINET DOCUMENT LIBRARY
https://docs.fortinet.com

FORTINET VIDEO GUIDE
https://video.fortinet.com

FORTINET BLOG
https://blog.fortinet.com

CUSTOMER SERVICE & SUPPORT
https://support.fortinet.com

FORTINET COOKBOOK
https://cookbook.fortinet.com

FORTINET TRAINING & CERTIFICATION PROGRAM
https://www.fortinet.com/support-and-training/training.html

NSE INSTITUTE
https://training.fortinet.com

FORTIGUARD CENTER
https://fortiguard.com/

END USER LICENSE AGREEMENT
https://www.fortinet.com/doc/legal/EULA.pdf

FEEDBACK
Email: techdocs@fortinet.com

February 25, 2019
FortiGate-VM on OpenStack 6.0.1 Installation Guide
01-601-498214-20190225

TABLE OF CONTENTS

About FortiGate-VM on OpenStack

FortiGate virtual appliances allow you to mitigate blind spots by implementing critical security controls within your virtual infrastructure. They also allow you to rapidly provision security infrastructure whenever and wherever it is needed. FortiGate virtual appliances feature all the security and networking services common to hardware-based FortiGate appliances. You can deploy a mix of FortiGate hardware and virtual appliances, operating together and managed from a common centralized management platform.

OpenStack-based clouds provide the environment needed for elastic, on-demand multitenant applications. Networks are transitioning to new models more suited to the cloud with SDN, NFV, and Virtual Network Infrastructure, and their relationships between networking, security orchestration, and policy enforcement.

Our OpenStack Neutron solution embraces the software-defined security framework providing out-of-the-box integration so that advanced network security can be seamlessly applied in logical and dynamic environments.

This document describes how to deploy a FortiGate virtual appliance in an OpenStack environment. You can install FGT-VM64-KVM and FOS-VM64-KVM firmware into an OpenStack environment.

Preparing for deployment

This document assumes that before deploying the FortiGate-VM virtual appliance on the OpenStack virtual platform, you have addressed the following requirements:

Virtual Environment

The OpenStack software is installed on a physical server with sufficient resources to support the FortiGate-VM and all other virtual machines that will be deployed on the platform.

If the FortiGate-VM virtual machine will be configured to operate in transparent mode, ensure that the OpenStack environment includes virtual switches configured to support the operation of the FortiGate-VM before you create the FortiGate-VM virtual machine.

If you will be setting up multiple FortiGate-VMs in a FortiGate Clustering Protocol (FGCP) High Availability (HA) cluster:

- Make sure to purchase identical FGT-VM64-KVM or FOS-VM64-KVM licenses
- Be prepared to set up a dedicated network in the OpenStack environment for the HA heartbeat, see Verifying HA cluster status on page 23

Management software

The VMware management software, OpenStack Horizon, is installed on a computer with network access to the OpenStack server.

Connectivity

An Internet connection is required for the FortiGate-VM to contact FortiGuard to validate its license. If the FortiGate-VM is in a closed environment, it must be able to connect to a FortiManager to validate the FortiGate-VM license. See "Validating the FortiGate-VM license with FortiManager".

Configuring resources

Before you start the FortiGate-VM for the first time, ensure that the following resources are configured as specified by the FortiGate-VM virtual appliance license:

- Disk sizes
- CPUs
- RAM
- Network settings

To configure the resources for a FortiGate-VM deployed on OpenStack, use the OpenStack Horizon client.

Registering the FortiGate-VM virtual appliance

Registering the FortiGate-VM virtual appliance with Customer Service & Support allows you to obtain the FortiGate-VM virtual appliance license file.

To register the FortiGate-VM virtual appliance:

1. Log in to the Customer Service & Support site using a support account, or select **Sign Up** to create an account.
2. In the main page, under **Asset**, select **Register/Renew**.
3. In the **Registration** page, enter the registration code that was emailed to you, and select **Register** to access the registration form.
4. Complete and submit the registration form.
5. In the registration acknowledgment page, click the **License File Download** link.
6. Save the license file (`.lic`) to your local computer. See "Uploading the FortiGate-VM virtual appliance license" or "Validating the FortiGate-VM license using FortiManager" for information about uploading the license file to your FortiGate-VM via the web-based manager.

Downloading the FortiGate-VM virtual appliance deployment package

FortiGate-VM deployment packages are found on the Customer Service & Support site. In the Download drop-down menu, select VM Images to access the available VM deployment packages.

In the Select Product drop-down menu, select FortiGate.

In the Select Platform drop-down menu, select KVM.

Select the FortiOS version you want to download.

There are two files available for download: the file required to upgrade from an earlier version and the file required for a new deployment.

Click the Download button and save the file.

For more information see the FortiGate product datasheet available on the Fortinet web site, https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/FortiGate_VM.pdf.

You can also download the following resources for the firmware version:

- FortiOS Release Notes
- FORTINET-FORTIGATE MIB file
- FSSO images
- SSL VPN client

Deployment package contents for OpenStack

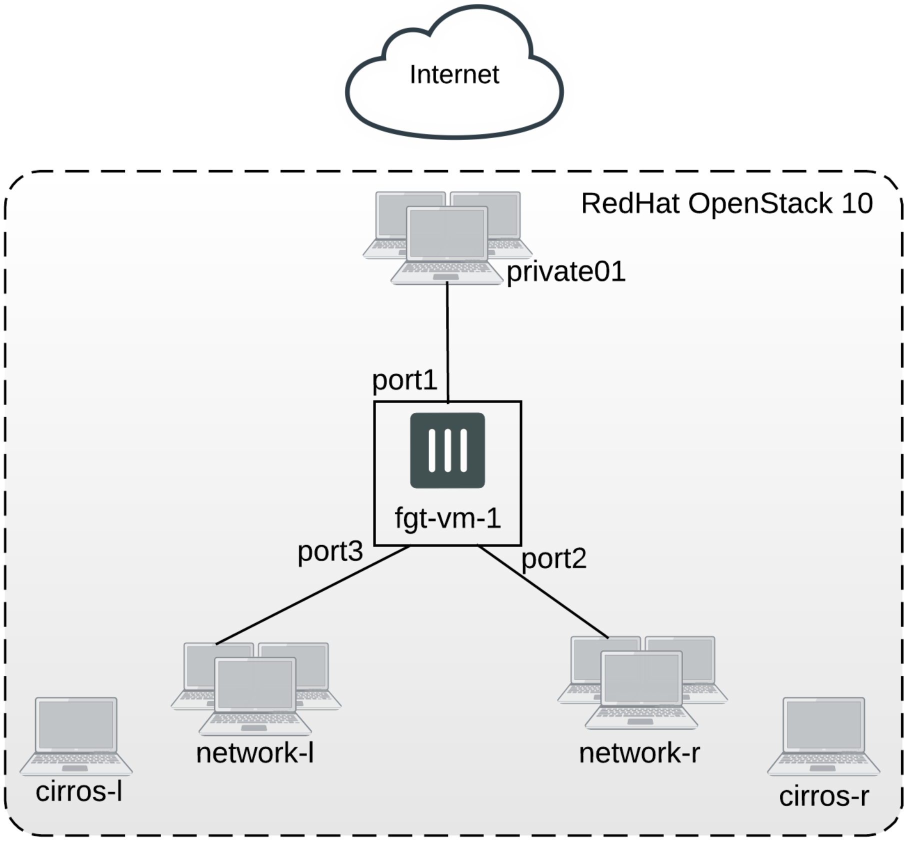
The FORTINET.out.kvm.zip contains only `fortios.qcow2`, the FortiGate VM system hard disk in qcow2 format. You will need to manually:

- create a 32GB log disk
- specify the virtual hardware settings

Deploying a FortiGate-VM instance in an OpenStack environment

This example shows how to set up FortiGate-VM instance in an OpenStack 10 environment. The FortiGate-VM instance is connected to a private network (private01) and protects two networks (network-l and network-r). Each network includes a CirrOS instance (cirros-l and cirros-r) for testing.

Example FortiGate-VM configuration in an OpenStack environment



Setting up networks in OpenStack

From the OpenStack environment command line, enter the following commands to create network-r and network-l.

```
$ source overcloudrc_tenant01
```

```
$ openstack network create network-r
$ openstack subnet create subnet-r --network network-r --subnet-range 172.32.0.0/24 --dns-
nameserver 208.91.112.53

$ openstack network create network-l

$ openstack subnet create subnet-l --network network-l --subnet-range 172.33.0.0/24 --dns-
nameserver 208.91.112.53
```

Add the CirrOS instances to network-r and network-l:

```
$ openstack server create --flavor m1.tiny --image cirros035 --security-group web --nic net-
id=network-r  cirros-r

$ openstack server create --flavor m1.tiny --image cirros035 --security-group web --nic net-
id=network-l  cirros-l
```

Deploying a FortiGate-VM instance into the configured networks

From the OpenStack command line, enter the following commands to deploy the fgt-vm-1 FortiGate-VM instance. These commands use the standard license file you receive when you register your FortiGate-VM instance (in this example, FGVM080000103268.lic).

```
$ openstack server create --flavor m1.fortigate --image fgtb1486 --user-data /home/stack-
/openstack/cloud-init/userdata.txt --config-drive=true --file license=/home/stack/FG-
licenses/FGVM080000103268.lic --security-group web --nic net-id=private01 --nic net-id=ne-
etwork-r --nic net-id=network-l --nic net-id=ha-sync fgt-vm-1
```

Creating a userdata.txt file to pre-configure a FortiGate-VM instance

The following example `userdata.txt` file sets up a FortiGate-VM instance (fgt-vm-1) with a basic default configuration customized for your environment and requirements. This example configures interfaces, adds a DNS server, and configures two firewall policies that allow devices in network-l and network-r to access the private01 network and the Internet through the private01 network.

Example userdata.txt file for fgt-vm-1

The following example `userdata.txt` file could be used for fgt-vm-1.

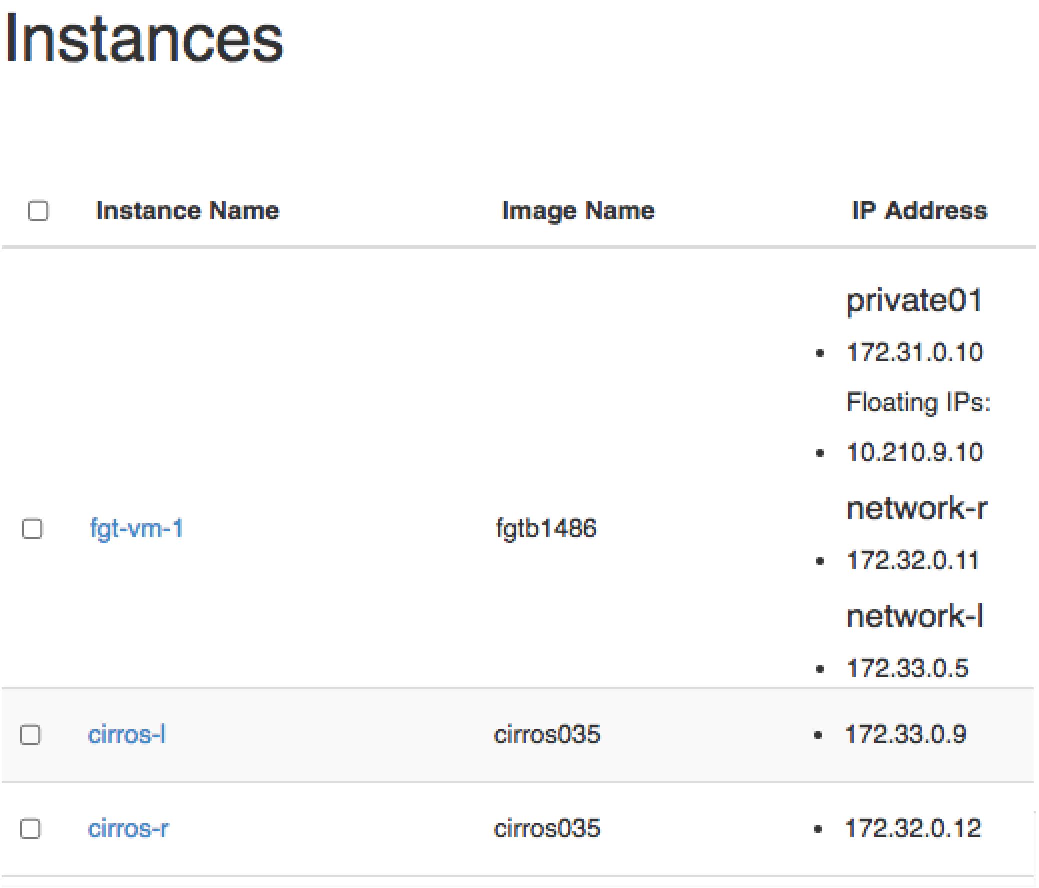
```
#FGT VM Config File

config sys global
set hostname fgt-vm-1
end
config system interface
edit port1
set mode dhcp
set allowaccess http https ssh ping
```

```
next
edit port2
set mode dhcp
set defaultgw disable
set allowaccess http https ssh ping
next
edit port3
set mode dhcp
set defaultgw disable
set allowaccess http https ssh ping
next
end
config system dns
set primary 208.91.112.53
end
config firewall policy
edit 1
set name "network-l internet access"
set dstintf "port3"
set srcintf "port1"
set srcaddr "all"
set dstaddr "all"
set action accept
set schedule "always"
set service "ALL"
set nat enable
next
edit 2
set name "network-r internet access"
set dstintf "port2"
set srcintf "port1"
set srcaddr "all"
set dstaddr "all"
set action accept
set schedule "always"
set service "ALL"
set nat enable
end
config system central-management
set include-default-servers disable
set type fortimanager
set fmg 10.210.8.25
config server-list
edit 1
set server-type update rating
set server-address 10.210.8.25
end
end
```

Disabling port security for the FortiGate-VM and CirrOS instances

In OpenStack, the networking component (called Neutron) only allows traffic with known IP/MAC address combinations. This makes the network very secure; however, normal firewall traffic would contain very many IP/MAC address combinations, and it's not practical to add them all to the configuration. Instead, to allow normal firewall traffic, you need to disable port security for your FortiGate-VM instance. For more information, see Managing port level security in OpenStack.

Use the Horizon **Instances** view to verify the IP addresses of the FortiGate-VM instance, the CirrOS instances, and the networks that the interfaces are connected to. For example:



Instances

☐	Instance Name	Image Name	IP Address
☐	fgt-vm-1	fgtb1486	private01 • 172.31.0.10 Floating IPs: • 10.210.9.10 network-r • 172.32.0.11 network-l • 172.33.0.5
☐	cirros-l	cirros035	• 172.33.0.9
☐	cirros-r	cirros035	• 172.32.0.12

From the OpenStack command line, run the following bash script to disable port security on the FortiGate-VM interfaces.

```
#!/bin/bash
echo
echo 'Disable port_security on fgt-vm-1'
echo
echo
`source /home/stack/overcloudrc_tenant01`
FGT='fgt-vm-1'
IPADDR=`openstack server show $FGT | grep addresses | awk -F "|" '{print $3}' | awk -F "=" '
{print $2}' | awk -F ";" '{print $1}'`
PORTID=`openstack port list | grep $IPADDR | awk -F "|" '{print $2}'`
neutron port-update $PORTID --no-security-groups --port_security_enabled=False
echo
echo $IPADDR
echo `openstack port show $PORTID`
echo
IPADDR=`openstack server show $FGT | grep addresses | awk -F "|" '{print $3}' | awk -F "=" '
{print $3}' | awk -F ";" '{print $1}'`
```

```
PORTID=`openstack port list | grep $IPADDR | awk -F "|" '{print $2}'`
`neutron port-update $PORTID --no-security-groups --port_security_enabled=False`
echo
echo $IPADDR
echo `openstack port show $PORTID`
echo
IPADDR=`openstack server show $FGT | grep addresses | awk -F "|" '{print $3}' | awk -F "=" '
{print $4}' | awk -F ";" '{print $1}'`
PORTID=`openstack port list | grep $IPADDR | awk -F "|" '{print $2}'`
`neutron port-update $PORTID --no-security-groups --port_security_enabled=False`
echo
echo $IPADDR
echo `openstack port show $PORTID`
echo
IPADDR=`openstack server show $FGT | grep addresses | awk -F "|" '{print $3}' | awk -F "=" '
{print $5}' | awk -F ";" '{print $1}'`
PORTID=`openstack port list | grep $IPADDR | awk -F "|" '{print $2}'`
`neutron port-update $PORTID --no-security-groups --port_security_enabled=False`
echo
echo $IPADDR
echo `openstack port show $PORTID`
echo
```

From the OpenStack command line, associate floating IPs to the FortiGate-VM by entering the following command:

```
openstack server add floating ip fgt-vm-1 10.210.9.10
```

Setting up the FortiGate-VM network configuration

From the CLI of the FortiGate-VM instance, enter the following commands to change the FortiGate-VM interfaces from DHCP to static and add IP addresses. The IP addresses assigned to the interfaces must be on the subnets of the networks that the interfaces are connected to.

```
config system interface
   edit "port1"
      set mode static
      set ip 172.31.0.3 255.255.255.0
      set allowaccess ping https ssh http
   next
   edit "port2"
      set mode static
      set ip 172.32.0.9 255.255.255.0
      set allowaccess ping https ssh http
   next
   edit "port3"
      set mode static
      set ip 172.33.0.4 255.255.255.0
      set allowaccess ping https ssh http
end
```

Enter the following command to add a static route.

```
config router static
   edit 1
   set gateway 172.31.0.1
```

```
    set device "port1"
end
```

Verifying Internet access

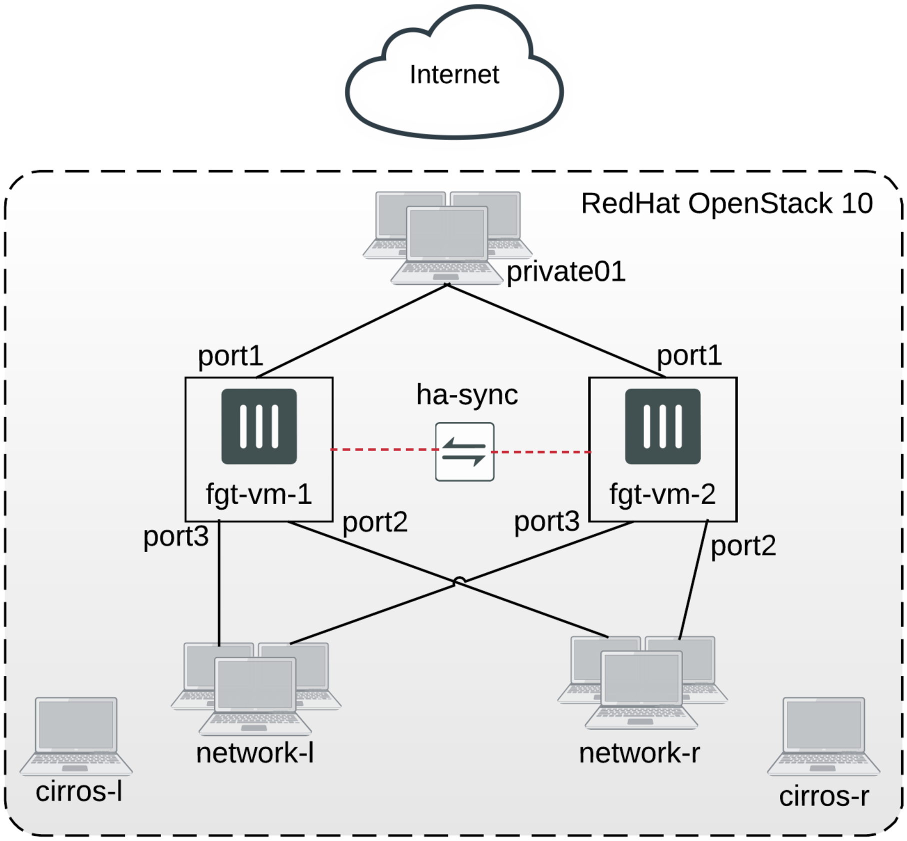
Log in to the cirros instances on each network, and attempt to access an address on the Internet or on the private01 network if internet access is not possible from the private01 network. One way to do this is to ping an IP address on the private01 network or on the Internet.

Deploying two FortiGate-VM instances in an HA configuration in an OpenStack environment

This example shows how to set up FortiGate Clustering Protocol (FGCP) HA with two FortiGate-VM instances in an OpenStack 10 environment. The FortiGate-VMs are connected to a private network (private01) and protect two networks (network-l and network-r). Each network includes a CirrOS instance (cirros-l and cirros-r) for testing.

To support HA heartbeat communication, the OpenStack environment also includes a network named ha-sync configured with the subnet used by the HA heartbeat interfaces (169.254.0.0/24).

Example FortiGate-VM HA configuration in an OpenStack environment



Setting up networks in OpenStack

From the OpenStack environment command line, enter the following commands to create network-r and network-l.

```
$ source overcloudrc_tenant01

$ openstack network create network-r

$ openstack subnet create subnet-r --network network-r --subnet-range 172.32.0.0/24 --dns-
nameserver 208.91.112.53

$ openstack network create network-l

$ openstack subnet create subnet-l --network network-l --subnet-range 172.33.0.0/24 --dns-
nameserver 208.91.112.53
```

Add the CirrOS instances to network-r and network-l:

```
$ openstack server create --flavor m1.tiny --image cirros035 --security-group web --nic net-
id=network-r  cirros-r

$ openstack server create --flavor m1.tiny --image cirros035 --security-group web --nic net-
id=network-l  cirros-l
```

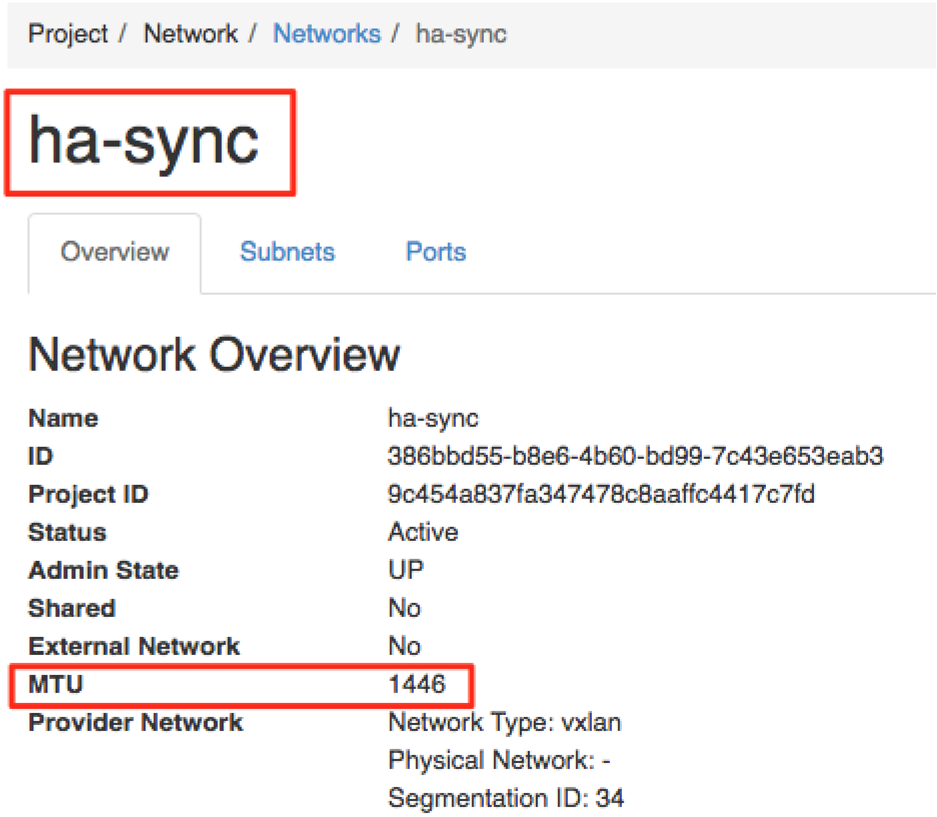
Setting up the ha-sync network for the HA heartbeat

From the OpenStack environment command line, enter the following commands to create the HA-sync network to be used for HA heartbeat communication.

```
$ openstack network create ha-sync

$ openstack subnet create subnet-ha --network ha-sync --subnet-range 169.254.0.0/24 --dns-
nameserver 208.91.112.53
```

Verifying the MTU assigned to the ha-sync network

You can use the OpenStack Horizon **Networks** view to verify the MTU assigned to the ha-sync network.



Network Overview

Name	ha-sync
ID	386bbd55-b8e6-4b60-bd99-7c43e653eab3
Project ID	9c454a837fa347478c8aaffc4417c7fd
Status	Active
Admin State	UP
Shared	No
External Network	No
MTU	1446
Provider Network	Network Type: vxlan Physical Network: - Segmentation ID: 34

Deploying two FortiGate-VMs into the configured networks

From the OpenStack command line, enter the following commands to deploy two FortiGate-VM instances (fgt-vm-1 and fgt-vm-2). These commands use the standard license files you receive when you register your FortiGate-VMs (in this example, FGVM080000103268.lic and FGVM080000109643.lic).

```
$ openstack server create --flavor m1.fortigate --image fgtb1486 --user-data /home/stack-
/openstack/cloud-init/userdata.txt --config-drive=true --file license=/home/stack/FG-
licenses/FGVM080000103268.lic --security-group web --nic net-id=private01 --nic net-id=ne-
etwork-r --nic net-id=network-l --nic net-id=ha-sync fgt-vm-1

$ openstack server create --flavor m1.fortigate --image fgtb1486 --user-data /home/stack-
/openstack/cloud-init/userdata.txt --config-drive=true --file license=/home/stack/FG-
licenses/FGVM080000109643.lic --security-group web --nic net-id=private01 --nic net-id=ne-
etwork-r --nic net-id=network-l --nic net-id=ha-sync fgt-vm-2
```

Creating userdata.txt files to pre-configure FortiGate-VM instances

The following example `userdata.txt` file sets up a FortiGate-VM instance with a basic default configuration customized for your environment and requirements. This example configures interfaces, and adds a DNS server and two firewall policies that allow any traffic to pass between the port2 and port3 interfaces. These policies make it easier to test HA failover.

In addition, the MTU of the port4 interface is set to be compatible with the OpenStack 10 environment, which by default, has an MTU of 1446. (In the `userdata.txt` file, the MTU of port4 is set to 1400.) Using the same MTU setting as the OpenStack 10 environment enables the HA heartbeat interfaces to communicate effectively over the ha-sync network.

See these pages for more information on RedHat OpenStack networks and MTU values:

- https://access.redhat.com/solutions/2980001
- https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html/networking_guide/sec-mtu

Example userdata.txt file for fgt-vm-1

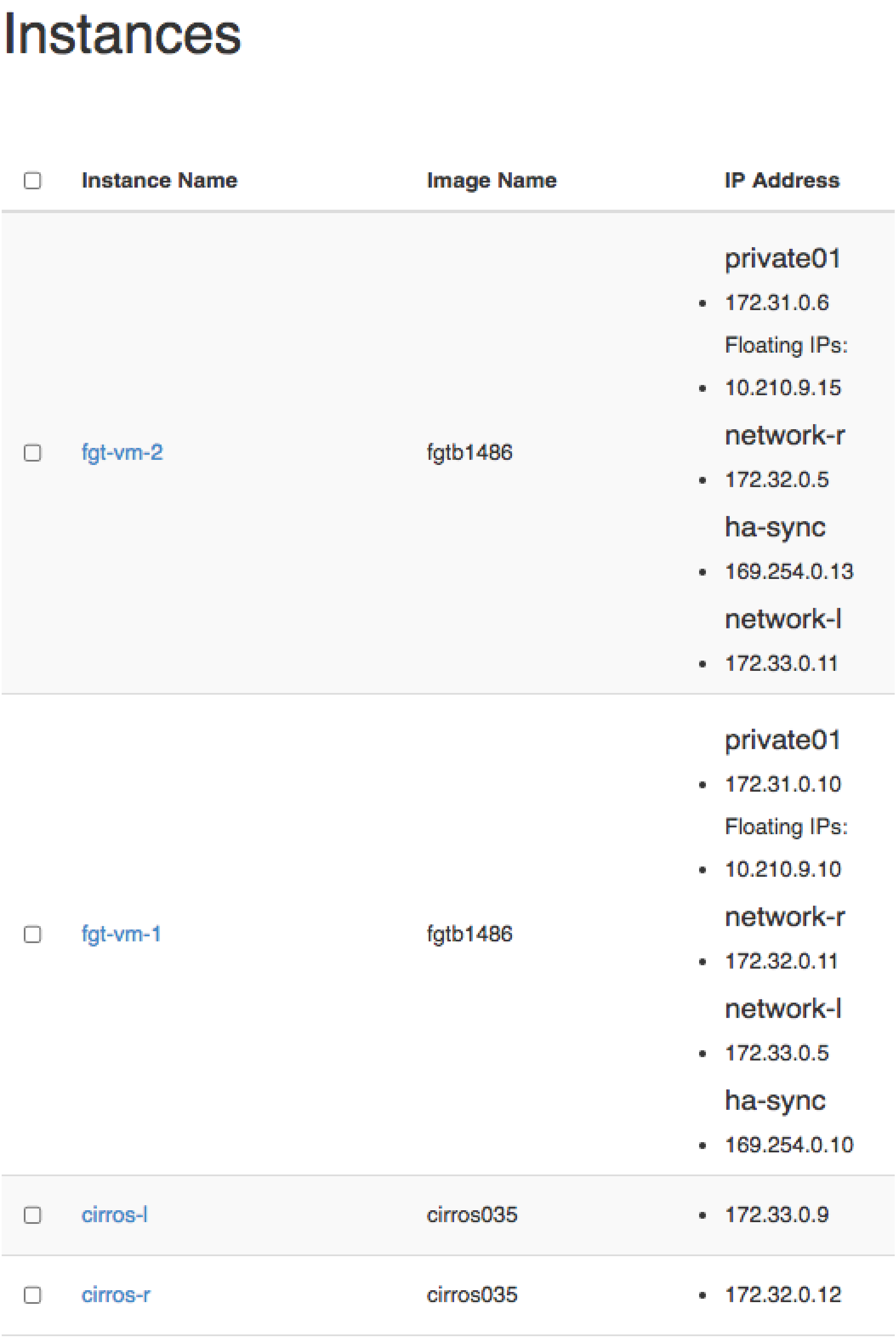
The following example `userdata.txt` file could be used for fgt-vm-1. The `userdata.txt` file for fgt-vm-2 would be the same except for the hostname.

```
#FGT VM Config File

config sys global
set hostname fgt-vm-1
end
config system interface
edit port1
set mode dhcp
set allowaccess http https ssh ping
next
edit port2
set mode dhcp
set defaultgw disable
set allowaccess http https ssh ping
next
edit port3
set mode dhcp
set defaultgw disable
set allowaccess http https ssh ping
next
edit port4
set mtu-override enable
set mtu 1400
next
end
config system dns
set primary 208.91.112.53
end
config firewall policy
edit 1
set name "Allow port2 to port3"
set dstintf "port2"
set srcintf "port3"
set srcaddr "all"
set dstaddr "all"
set action accept
set schedule "always"
set service "ALL"
set nat enable
next
edit 2
set name "Allow port3 to port2"
set dstintf "port3"
set srcintf "port2"
set srcaddr "all"
set dstaddr "all"
set action accept
set schedule "always"
set service "ALL"
set nat enable
end
config system central-management
set include-default-servers disable
set type fortimanager
set fmg 10.210.8.25
config server-list
edit 1
set server-type update rating
set server-address 10.210.8.25
end
end
```

Disabling port security for the FortiGate-VM and CirrOS instances

In OpenStack, the networking component (called Neutron) only allows traffic with known IP/MAC address combinations. This makes the network very secure; however, normal firewall traffic would contain very many IP/MAC address combinations and it's not practical to add them all to the configuration. Instead, to allow normal firewall traffic, you need to disable port security for the FortiGate-VMs. For more information, see Managing port level security in OpenStack.

Use the RedHat OpenStack Horizon **Instances** view to verify the IP addresses of the FortiGate-VM, the CirrOS instances, and the networks that the interfaces are connected to. For example:



Instances

□	Instance Name	Image Name	IP Address
□	fgt-vm-2	fgtb1486	private01 • 172.31.0.6 Floating IPs: • 10.210.9.15 network-r • 172.32.0.5 ha-sync • 169.254.0.13 network-l • 172.33.0.11
□	fgt-vm-1	fgtb1486	private01 • 172.31.0.10 Floating IPs: • 10.210.9.10 network-r • 172.32.0.11 network-l • 172.33.0.5 ha-sync • 169.254.0.10
□	cirros-l	cirros035	• 172.33.0.9
□	cirros-r	cirros035	• 172.32.0.12

From the OpenStack command line, run the following bash script to disable port security on the FortiGate-VM interfaces.

```
#!/bin/bash
echo
echo 'Disable port_security on fgt-vm-1'
echo
echo
`source /home/stack/overcloudrc_tenant01`
FGT='fgt-vm-1'
IPADDR=`openstack server show $FGT | grep addresses | awk -F "|" '{print $3}' | awk -F "=" '
{print $2}' | awk -F ";" '{print $1}'`
PORTID=`openstack port list | grep $IPADDR | awk -F "|" '{print $2}'`
neutron port-update $PORTID --no-security-groups --port_security_enabled=False
echo
echo $IPADDR
echo `openstack port show $PORTID`
echo
IPADDR=`openstack server show $FGT | grep addresses | awk -F "|" '{print $3}' | awk -F "=" '
{print $3}' | awk -F ";" '{print $1}'`
PORTID=`openstack port list | grep $IPADDR | awk -F "|" '{print $2}'`
`neutron port-update $PORTID --no-security-groups --port_security_enabled=False`
echo
echo $IPADDR
echo `openstack port show $PORTID`
echo
IPADDR=`openstack server show $FGT | grep addresses | awk -F "|" '{print $3}' | awk -F "=" '
{print $4}' | awk -F ";" '{print $1}'`
PORTID=`openstack port list | grep $IPADDR | awk -F "|" '{print $2}'`
`neutron port-update $PORTID --no-security-groups --port_security_enabled=False`
echo
echo $IPADDR
echo `openstack port show $PORTID`
echo
IPADDR=`openstack server show $FGT | grep addresses | awk -F "|" '{print $3}' | awk -F "=" '
{print $5}' | awk -F ";" '{print $1}'`
PORTID=`openstack port list | grep $IPADDR | awk -F "|" '{print $2}'`
`neutron port-update $PORTID --no-security-groups --port_security_enabled=False`
echo
echo $IPADDR
echo `openstack port show $PORTID`
echo
echo 'Disable port-security on fgt-vm-2'
echo
FGT='fgt-vm-2'
IPADDR=`openstack server show $FGT | grep addresses | awk -F "|" '{print $3}' | awk -F "=" '
{print $2}' | awk -F ";" '{print $1}'`
PORTID=`openstack port list | grep $IPADDR | awk -F "|" '{print $2}'`
neutron port-update $PORTID --no-security-groups --port_security_enabled=False
echo
echo $IPADDR
echo `openstack port show $PORTID`
echo
IPADDR=`openstack server show $FGT | grep addresses | awk -F "|" '{print $3}' | awk -F "=" '
{print $3}' | awk -F ";" '{print $1}'`
PORTID=`openstack port list | grep $IPADDR | awk -F "|" '{print $2}'`
`neutron port-update $PORTID --no-security-groups --port_security_enabled=False`
echo
echo $IPADDR
echo `openstack port show $PORTID`
echo

IPADDR=`openstack server show $FGT | grep addresses | awk -F "|" '{print $3}' | awk -F "=" '
{print $4}' | awk -F ";" '{print $1}'`
PORTID=`openstack port list | grep $IPADDR | awk -F "|" '{print $2}'`
`neutron port-update $PORTID --no-security-groups --port_security_enabled=False`
echo
echo $IPADDR
echo `openstack port show $PORTID`
echo
IPADDR=`openstack server show $FGT | grep addresses | awk -F "|" '{print $3}' | awk -F "=" '
{print $5}' | awk -F ";" '{print $1}'`
PORTID=`openstack port list | grep $IPADDR | awk -F "|" '{print $2}'`
`neutron port-update $PORTID --no-security-groups --port_security_enabled=False`
echo
echo $IPADDR
echo `openstack port show $PORTID`
echo
```

From the OpenStack command line, associate floating IPs to the two FortiGate-VMs, by entering the following commands:

```
openstack server add floating ip fgt-vm-1 10.210.9.10

openstack server add floating ip fgt-vm-2 10.210.9.14
```

Setting up the FortiGate-VM HA configuration

From the CLI of each FortiGate-VM instance, configure both FortiGate-VMs for HA. Both FortiGate-VM instances must have the same HA configuration; for example:

```
config system ha
   set group-name "group-01"
   set mode a-p
   set password <password>
   set hbdev "port4" 50
   set override disable
   set monitor "port2"
end
```

Completing the FortiGate-VM network configuration

From each FortiGate-VM instance CLI, enter the following commands to change the FortiGate-VM interfaces from DHCP to static, add IP addresses, and add a static route. The IP addresses assigned to the interfaces must be on the subnets of the networks that the interfaces are connected to.

The example shows the fgt-vm-1 configuration. The fgt-vm-2 configuration would be the same except for the interface IP addresses.

```
config system interface
   edit "port1"
```

```
      set mode static
      set ip 172.31.0.3 255.255.255.0
      set allowaccess ping https ssh http
   next
   edit "port2"
      set mode static
      set ip 172.32.0.9 255.255.255.0
      set allowaccess ping https ssh http
   next
   edit "port3"
      set mode static
      set ip 172.33.0.4 255.255.255.0
      set allowaccess ping https ssh http
end

config router static
   edit 1
   set gateway 172.31.0.1
   set device "port1"
end
```

Testing HA operation and failover

This section describes how to verify that a FortiGate-VM HA cluster in an OpenStack environment is operating normally and will failover successfully.

On the cirros-l instance console (see the diagram Deploying two FortiGate-VMs into the configured networks on page 16, start a continuous ping to the IP address of cirros-r. On the cirros-r instance console, start a continuous ping to the IP address of cirros-l:

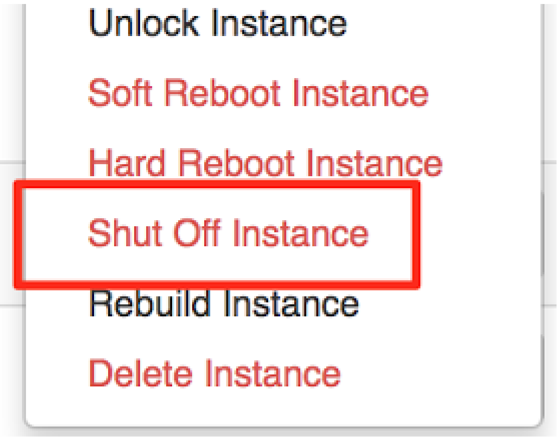
```
$ ping 172.32.0.11
PING 172.32.0.11 (172.32.0.11): 56 data bytes
64 bytes from 172.32.0.11: seq=0 ttl=63 time=0.402 ms
64 bytes from 172.32.0.11: seq=0 ttl=63 time=0.433 ms
64 bytes from 172.32.0.11: seq=0 ttl=63 time=0.502 ms
64 bytes from 172.32.0.11: seq=0 ttl=63 time=0.408 ms
64 bytes from 172.32.0.11: seq=0 ttl=63 time=0.362 ms
```

On both FortiGate-VMs, use the following diagnose command to sniff ICMP packets. You should only see packets going through the primary unit.

```
fgt-vm-1 # diagnose sniffer packet any 'icmp' 4
interfaces =[any]
filters= [icmp]
109.413710 port_ha in 169.251.0.1 - > 169.251.0.2: icmp: 169.251.0.1 udp port 53
unreachable
111.797651 port2 in 172.32.0.11 - > 172.33.0.12: icmp: echo request
111.797676 port3 out 172.33.0.1 - > 172.33.0.12: icmp: echo request
111.797932 port3 in 172.33.0.12 - > 172.33.0.1: icmp: echo reply
111.797910 port2 out 172.33.0.12 - > 172.32.0.11: icmp: echo reply
112.372066 port3 in 172.33.0.12 - > 172.32.0.11: icmp: echo request
112.372081 port2 out 172.32.0.9 - > 172.32.0.11: icmp: echo request
112.372225 port2 in 172.32.0.11 - > 172.32.0.9: icmp: echo reply
112.372232 port3 out 172.32.0.11 - > 172.33.0.12: icmp: echo reply
```

```
112.797831 port2 in 172.32.0.11 - > 172.33.0.12: icmp: echo request
112.797839 port3 out 172.33.0.1 - > 172.33.0.12: icmp: echo request
112.798019 port3 in 172.33.0.12 - > 172.33.0.1: icmp: echo reply
112.798021 port2 out 172.33.0.12 - > 172.32.0.11: icmp: echo reply
```

Shut down the primary unit. You can do this from the OpenStack Horizon **Instances** list:



After failover, enter the following diagnose command from the new primary unit to verify that the pings are now going through that unit.

```
fgt-vm-2 # diagnose sniffer packet any' icmp' 4
interfaces= [any]
filter s= [icmp]
0.360973 port3 in 172.33.0.12 - > 172.32.0.11: icmp: echo request
0.360983 port2 out 172.32.0.9 - > 172.32.0.11: icmp: echo request
0.361220 port2 in 172.32.0.11 - > 172.32.0.9: icmp: echo reply
0.361222 port3 out 172.32.0.11 - > 172.33.0.12: icmp: echo reply
0.785522 port2 in 172.32.0.11 - > 172.33.0.12: icmp: echo request
0.785527 port3 out 172.33.0.4 - > 172.33.0.12: icmp: echo request
0.785688 port3 in 172.33.0.12 - > 172.33.0.4: icmp: echo reply
0.785690 port2 out 172.33.0.12 - > 172.32.0.11: icmp: echo reply
1.360860 port3 in 172.33.0.12 - > 172.32.0.11: icmp: echo request
1.360864 port2 out 172.32.0.9 - > 172.32.0.11: icmp: echo request
1.361025 port2 in 172.32.0.11 - > 172.32.0.9: icmp: echo reply
1.361027 port3 out 172.32.0.11 - > 172.33.0.12: icmp: echo reply
```

Restart the FortiGate-VM instance that you shut down. After a short while it should re-join the cluster.

Verifying HA cluster status

On a FortiGate-VM in an HA cluster, you can use the following command to verify the status of the cluster:

```
fgt-vm # diagnose sys ha status
HA information
Statistics
    traffic.local = s:0 p:42311 b:9008646
    traffic.total = s:0 p:42316 b:9009528
    activity.fdb  = c:0 q:0
```

```
Model=80008, Mode=2 Group=0 Debug=0
nvcluster=1, ses_pickup=0, delay=0

[Debug_Zone HA information]
HA group member information: is_manage_master=1.
FGVM080000109643: Master, serialno_prio=0, usr_priority=128, hostname=fgt-vm
FGVM080000103268:  Slave, serialno_prio=1, usr_priority=128, hostname=fgt-vm

[Kernel HA information]
vcluster 1, state=work, master_ip=169.254.0.1, master_id=0:
FGVM080000109643: Master, ha_prio/o_ha_prio=0/0
FGVM080000103268:  Slave, ha_prio/o_ha_prio=1/1
```

The following command shows similar information:

```
fgt-vm # get system ha status
HA Health Status: OK
Model: FortiGate-VM64-KVM
Mode: HA A-P
Group: 0
Debug: 0
Cluster Uptime: 0 days 02:04:26
Cluster state change time: 2017-09-01 03:08:19
Master selected using:
   <2017/09/01 03:08:19> FGVM080000109643 is selected as the master because it has the largest
value of serialno.
ses_pickup: disable
override: disable
Configuration Status:
   FGVM080000109643(updated 2 seconds ago): in-sync
   FGVM080000103268(updated 0 seconds ago): out-of-sync
System Usage stats:
   FGVM080000109643(updated 2 seconds ago):
        sessions=4, average-cpu-user/nice/system/idle=0%/0%/0%/100%, memory=55%
   FGVM080000103268(updated 0 seconds ago):
        sessions=0, average-cpu-user/nice/system/idle=0%/0%/0%/100%, memory=54%
HBDEV stats:
   FGVM080000109643(updated 2 seconds ago):
        port4: physical/10000full, up, rx-bytes/packets/dropped/errors=15043566/61878/0/0,
tx=158364378/146977/0/0
   FGVM080000103268(updated 0 seconds ago):
        port4: physical/10000full, up, rx-bytes/packets/dropped/errors=29442835/61625/49/0,
tx=25246662/68626/0/0
MONDEV stats:
   FGVM080000109643(updated 2 seconds ago):
        port2: physical/10000full, up, rx-bytes/packets/dropped/errors=1892/8/0/0,
tx=173710/307/0/0
   FGVM080000103268(updated 0 seconds ago):
        port2: physical/10000full, up, rx-bytes/packets/dropped/errors=174390/306/0/0,
tx=2352/13/0/0
Master: fgt-vm          , FGVM080000109643
Slave : fgt-vm          , FGVM080000103268
number of vcluster: 1
vcluster 1: work 169.254.0.1
Master:0 FGVM080000109643
Slave :1 FGVM080000103268
```

The command `diagnose system ha checksum show` shows whether the configurations of the FortiGate-VMs in the cluster are synchronized. If the configurations are synchronized, both sets of checksums should match.

```
fgt-vm # diagnose sys ha checksum show
is_manage_master()=1, is_root_master()=1
debugzone
global: 33 6f ee 5b 78 a5 22 84 39 ec 36 d3 1c 54 7c 78
root: 40 0d fb 04 12 41 df ad f1 64 14 03 ff ec f5 01
all: d3 2f 6f bb a6 e7 77 db 27 75 81 b2 94 f3 fd 68

checksum
global: 33 6f ee 5b 78 a5 22 84 39 ec 36 d3 1c 54 7c 78
root: 40 0d fb 04 12 41 df ad f1 64 14 03 ff ec f5 01
all: d3 2f 6f bb a6 e7 77 db 27 75 81 b2 94 f3 fd 68
```

If the checksums do not match, you can use the `diagnose sys ha checksum show` and `diagnose sys ha checksum show global` commands to show more detailed checksum results. The following example shows the first few lines of output of the `diagnose sys ha checksum show global` command:

```
diagnose sys ha checksum show global
system.global: 2c79958c132639dfe61ab782a2f213ec
system.accprofile: 7d79452c78377be2616149264a18fd5c
system.vdom-link: 00000000000000000000000000000000
wireless-controller.inter-controller: 00000000000000000000000000000000
wireless-controller.global: 00000000000000000000000000000000
wireless-controller.vap: 00000000000000000000000000000000
system.switch-interface: 00000000000000000000000000000000
system.interface: 8690699bc33c7c15b20e017876cf1e37
...
```

If the configurations are synchronized, all the checksums displayed using these commands from both FortiGate-VMs should match. If they do not, you can use the output to see what parts of the configuration are not synchronized.

Optimizing FortiGate-VM performance

The FortiGate VM and OpenStack performance optimization techniques described in this section can improve the performance of your FortiGate VM by optimizing the hardware and the OpenStack host environment for network- and CPU-intensive performance requirements of FortiGate-VMs.

SR-IOV

FortiGate VMs installed on OpenStack platforms support Single Root I/O virtualization (SR-IOV) to provide FortiGate VMs with direct access to hardware devices. Enabling SR-IOV means that one PCIe device (CPU or network card) can function for a FortiGate-VM as multiple separate physical devices (CPUs or network devices). SR-IOV reduces latency and improves CPU efficiency by allowing network traffic to pass directly between a FortiGate VM and a network card without passing through the OpenStack kernel and without using virtual switching.

FortiGate VMs benefit from SR-IOV because SR-IOV optimizes network performance and reduces latency. FortiGate VMs do not use OpenStack features that are incompatible with SR-IOV so you can enable SR-IOV without negatively affecting your FortiGate-VM.

SR-IOV hardware compatibility

SR-IOV requires that the hardware on which your OpenStack host is running has BIOS, physical NIC, and network driver support for SR-IOV.

To enable SR-IOV, your OpenStack platform must be running on hardware that is compatible with SR-IOV and with FortiGate-VMs. FortiGate-VMs require network cards that are compatible with ixgbevf or i40evf drivers.

For optimal SR-IOV support, install the most up to date ixgbevf or i40evf network drivers.

Create SR-IOV VFs

This section describes how to create Virtual Functions (VFs) for SR-IOV-compatible Intel network interfaces. An SR-IOV VF is a virtual PCIe device that you must add to OpenStack to allow your FortiGate-VM to use SR-IOV to communicate with a physical ethernet interface or Physical Function (PF).

Enable SR-IOV in the host system's BIOS by enabling VT-d.

Enable IOMMU for Linux by adding `intel_iommu=on` to kernel parameters. Do this by adding the following line to the `/etc/default/grub` file:

```
GRUB_CMDLINE_LINUX_DEFAULT="nomdmonddf nomdmonisw intel_iommu=on
```

Save your changes and from the Linux command line enter the following commands to update `grub` and reboot the host device.

```
# udate-grub
# reboot
```

On each compute node, create VFs using the PCI SYS interface:

```
# echo '7' > /sys/class/net/eth3/device/sriov/numvfs
```

If the previous command produces a `Device or resource busy` error message, you need to set `sriov_numvfs` to 0, before setting it to the new value.

Optionally determine the maximum number of VFs a PF can support:

```
# cat /sys/class/net/eth3/device/sriov_totalvfs
```

Enter the following command to make sure an SR-IOV interface is up and verify its status.

```
# ip link set eth3 up
# ip link show eth3
```

Enter the following command to verify that the VFs have been created:

```
# lspci | grep Ethernet
```

Enter the following command to make sure the VFs are re-created when the system reboots:

```
# echo "echo '7' > /sys/class/net/eth3/device/sriov_numvfs" >> /etc/rc.local
```

White listing PCI devices

You must white list SR-IOV devices so their traffic can pass through OpenStack to the FortiGate VM. The following example shows how to white list SR-IOV devices by modifying the `nova-compute` service. (You can also edit the `pci_passthrough_whitelist` parameter to add whitelisting.)

To modify the `nova-compute` service, open the `nova.comp` file and add the following line:

```
pci_passthrough_whitelist = { "devname": "eth3", "physical_network": "physnet2"}
```

This setting adds traffic from eth3 to the `physnet2` physical network and allows physnet2 traffic to pass through OpenStack to your FortiGate VM.

After entering this command, restart the `nova-compute` service.

Configuring neutron-server

Use the following steps to configure OpenStack `neutron-server` to support SR-IOV:

Add the `sriovnicswitch` as mechanism driver, edit the `ml2_conf.ini` file and add the following line:

```
mechanism_drivers = openvswitch,sriovnicswitch
```

Find the `vendor_id` and `product_id` of the VFs that you created. For example,

```
# lspci -nn | grep -i ethernet
87:00.0 Ethernet controller [0200]: Intel Corporation 82599 10 Gigabit Dual Port Backplane
      Connection [8086:10f8] (rev 01)
87:10.1 Ethernet controller [0200]: Intel Corporation 82599 Ethernet Controller Virtual
      Function [8086:10ed] (rev 01)
87:10.3 Ethernet controller [0200]: Intel Corporation 82599 Ethernet Controller Virtual
      Function [8086:10ed] (rev 01)
```

Add the following line to the `ml2_conf_sriov.ini` on each controller:

```
supported_pci_vendor_devs = 8086:10ed
```

In this example the `vendor_id` is `8086` and the `product_id` is `10ed`.

Add ml2_conf_sriov.ini to the neutron-server daemon. Edit the initialization script to configure the neutron-server service to load the SR-IOV configuration file. Include the following lines:

```
--config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugin.ini
--config-file /etc/neutron/plugins/ml2/ml2_conf_sriov.ini
```

Restart the neutron-server service.

Configure the nova-scheduler controller

To complete this step, on controllers running the nova-scheduler service, add PciPassthroughFilter to the scheduler_default_filters parameter and add the following new line under the [DEFAULT] section in nova.conf:

```
[DEFAULT]
scheduler_default_filters = RetryFilter, AvailabilityZoneFilter, RamFilter, ComputeFilter,
     ComputeCapabilitiesFilter, ImagePropertiesFilter, ServerGroupAntiAffinityFilter,
     ServerGroupAffinityFilter, PciPassthroughFilter
scheduler_available_filters = nova.scheduler.filters.all_filters
scheduler_available_filters = nova.scheduler.filters.pci_passthrough_
     filter.PciPassthroughFilter
```

Restart the nova-scheduler service.

Enable the neutron sriov-agent process

To enable the sriov-agent process, on each compute node, edit the sriov_agent.ini file and add the following:

Under [securitygroup] add:

```
firewall_driver = neutron.agent.firewall.NoopFirewallDriver
```

Under [sriov_nic] add:

```
physical_device_mappings = physnet2:eth3
exclude_devices =
```

The example physical_device_mappings setting includes one mapping between the physical network (physnet2) and one VF called eth3. If you have multiple VFs connected to the same physical network, you can add them all using the following syntax that shows how to add two VFs to physnet2.

```
physical_device_mappings = physnet2:eth3,physnet2:eth4
```

Also in the example, exclude_devices is empty and all VFs associatd with eth3 may be configured by the agent. You can also use exclude_devices to exclude specific VFs, for example to exclude eth1 and eth2:

```
exclude_devices = eth1:0000:07:00.2; 0000:07:00.3, eth2:0000:05:00.1; 0000:05:00.2
```

Enter the following command to verify that the neutron sriov_agent runs successfully:

```
# neutron-sriov-nic-agent --config-file /etc/neutron/neutron.conf --config-file
     /etc/neutron/plugins/ml2/sriov_agent.ini
```

Finally, you should enable the neutron sriov_agent service.

Assign SR-IOV virtual interfaces to a FortiGate VM

After SR-IOV has been added to your OpenStack host, you can now launch FortiGate-VM instances with neutron SR-IOV ports. Use the following steps:

Use the following command to display, the ID of the neutron network where you want the SR-IOV port to be created.

```
$ net_id=`neutron net-show net04 | grep "\ id\ " | awk '{ print $4 }'`
```

Use the following command to create the SR-IOV port. This command sets `vnic_type=direct`. Other options include `normal`, `direct-physical`, and `macvtap`:

```
$ port_id=`neutron port-create $net_id --name sriov_port --binding:vnic_type direct | grep "\
     id\ " | awk '{ print $4 }'`
```

Create the VM.This example includes the SR-IOV port created in the previous step:

```
$ nova boot --flavor m1.large --image ubuntu_14.04 --nic port-id=$port_id test-sriov
```

FortiGate-VM interrupt affinity

In addition to enabling SR-IOV in the VM host, to fully take advantage of SR-IOV performance improvements you need to configure interrupt affinity for your FortiGate-VM. Interrupt affinity (also called CPU affinity) maps FortiGate-VM interrupts to the CPUs that are assigned to your FortiGate-VM. You use a CPU affinity mask to define the CPUs that the interrupts are assigned to.

A common use of this feature would be to improve your FortiGate-VM's networking performance by:

- On the VM host, add multiple host CPUs to your FortiGate-VM.
- On the VM host, configure CPU affinity to specify the CPUs that the FortiGate-VM can use.
- On the VM host, configure other VM clients on the VM host to use other CPUs.
- On the FortiGate-VM, assign network interface interrupts to a CPU affinity mask that includes the CPUs that the FortiGate-VM can use.

In this way, all of the available CPU interrupts for the configured host CPUs are used to process traffic on your FortiGate interfaces. This configuration could lead to improve FortiGate-VM network performance because you have dedicated VM host CPU cycles to processing your FortiGate-VM's network traffic.

You can use the following CLI command to configure interrupt affinity for your FortiGate-VM:

```
config system affinity-interrupt
   edit <index>
      set interrupt <interrupt-name>
      set affinity-cpumask <cpu-affinity-mask>
   next
```

Where:

`<interrupt-name>` the name of the interrupt to associate with a CPU affinity mask. You can view your FortiGate-VM interrupts using the `diagnose hardware sysinfo interrupts` command. Usually you would associate all of the interrupts for a given interface with the same CPU affinity mask.

`<cpu-affinity-mask>` the CPU affinity mask for the CPUs that will process the associated interrupt.

For example, consider the following configuration:

- The port2 and port3 interfaces of a FortiGate-VM send and receive most of the traffic.
- On the VM host you have set up CPU affinity between your FortiGate-VM and four CPUs (CPU 0, 1 , 2, and 3)
- SR-IOV is enabled and SR-IOV interfaces use the i40evf interface driver.

The output from the `diagnose hardware sysinfo interrupts` command shows that port2 has the following transmit and receive interrupts:

```
i40evf-port2-TxRx-0
i40evf-port2-TxRx-1
i40evf-port2-TxRx-2
i40evf-port2-TxRx-3
```

The output from the `diagnose hardware sysinfo interrupts` command shows that port3 has the following transmit and receive interrupts:

```
i40evf-port3-TxRx-0
i40evf-port3-TxRx-1
i40evf-port3-TxRx-2
i40evf-port3-TxRx-3
```

Use the following command to associate the port2 and port3 interrupts with CPU 0, 1 , 2, and 3.

```
config system affinity-interrupt
   edit 1
      set interrupt "i40evf-port2-TxRx-0"
      set affinity-cpumask "0x0000000000000001"
   next
   edit 2
      set interrupt "i40evf-port2-TxRx-1"
      set affinity-cpumask "0x0000000000000002"
   next
   edit 3
      set interrupt "i40evf-port2-TxRx-2"
      set affinity-cpumask "0x0000000000000004"
   next
   edit 4
      set interrupt "i40evf-port2-TxRx-3"
      set affinity-cpumask "0x0000000000000008"
   next
   edit 1
      set interrupt "i40evf-port3-TxRx-0"
      set affinity-cpumask "0x0000000000000001"
   next
   edit 2
      set interrupt "i40evf-port3-TxRx-1"
      set affinity-cpumask "0x0000000000000002"
   next
   edit 3
      set interrupt "i40evf-port3-TxRx-2"
      set affinity-cpumask "0x0000000000000004"
   next
   edit 4
      set interrupt "i40evf-port3-TxRx-3"
      set affinity-cpumask "0x0000000000000008"
   next
end
```

FortiGate-VM affinity packet re-distribution

With SR-IOV enabled on the VM host and interrupt affinity configured on your FortiGate-VM there is one additional configuration you can add that may improve performance. Most common network interface hardware has restrictions on the number of RX/TX queues that it can process. This can result in some CPUs being much busier than others and the busy CPUs may develop extensive queues.

You can get around this potential bottleneck by configuring affinity packet re-distribution to allow overloaded CPUs to redistribute packets they receive to other less busy CPUs. The may result in a more even distribution of packet processing to all of the available CPUs.

You configure packet redistribution for interfaces by associating an interface with an affinity CPU mask. This configuration distributes packets set and received by that interface to the CPUs defined by the CPU affinity mask associated with the interface.

You can use the following CLI command to configure affinity packet redistribution for your FortiGate-VM:

```
config system affinity-packet-redistribution
   edit <index>
      set interface <interface-name>
      set affinity-cpumask <cpu-affinity-mask>
   next
```

Where:

`<interface-name>` the name of the interface to associate with a CPU affinity mast.

`<cpu-affinity-mask>` the CPU affinity mask for the CPUs that will process packets to and from the associated interface.

For example, you can improve the performance of the interrupt affinity example shown in the following command to allow packets sent and received by the port3 interface to be re-distributed to CPUs according to the 0xE CPU affinity mask.

```
config system affinity-packet-redistribution
   edit 1
      set interface port3
      set affinity-cpumask "0xE"
   next
```

FortiGate-VM MTU setting

For optimal performance you can set the MTU of your FortiGate VM interfaces to be compatible with the OpenStack 10 environment, which by default, has an MTU of 1446.

To set the MTU of a FortiGate interface to 1400:

```
config system interface
   edit port1
      set mtu-override enable
      set mtu 1400
   end
```

See these pages for more information on RedHat OpenStack networks and MTU values:

- https://access.redhat.com/solutions/2980001
- https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html/networking_guide/sec-mtu