KVM Administration Guide

FortiOS 8.0

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

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

FORTINET BLOG
https://blog.fortinet.com

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

FORTINET TRAINING & CERTIFICATION PROGRAM
https://www.fortinet.com/training-certification

FORTINET TRAINING INSTITUTE
https://training.fortinet.com

FORTIGUARD LABS
https://www.fortiguard.com

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

FEEDBACK
Email: techdoc@fortinet.com

April 21, 2026
FortiOS 8.0 KVM Administration Guide
01-80-1054288-20260421

TABLE OF CONTENTS

About FortiGate-VM on KVM

FortiGate-VMs 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-VMs feature all the security and networking services common to hardware-based FortiGate appliances. You can deploy a mix of FortiGate hardware and VMs, operating together and managed from a common centralized management platform.

This document describes how to deploy a FortiGate-VM in a KVM environment.

FortiGate-VM models and licensing

FortiGate-VM offers perpetual licensing (normal series and v-series) and annual subscription licensing. See VM license for details.

After you submit an order for a FortiGate-VM, Fortinet sends a license registration code to the email address that you entered on the order form. Use this code to register the FortiGate-VM with Customer Service & Support, then download the license file. After you upload the license to the FortiGate-VM and validate it, your FortiGate-VM is fully functional.

FortiGate-VM evaluation license

The FortiOS permanent trial license requires a FortiCare account. This trial license has limited features and capacity. See Permanent trial mode for FortiGate-VM for details.

FortiGate-VM virtual licenses and resources

The primary requirement for provisioning a FortiGate-VM may be the number of interfaces it can accommodate rather than its processing capabilities. In some cloud environments, options with a high number of interfaces tend to have high numbers of vCPUs.

FortiGate-VM licensing does not restrict whether the FortiGate can work on a VM instance in a public cloud that uses more vCPUs than the license allows. The number of vCPUs that the license indicates does not restrict the FortiGate from working, regardless of how many vCPUs the virtual instance includes. However, only the licensed number of vCPUs process traffic and management tasks. The FortiGate-VM does not use the rest of the vCPUs.

License	1 vCPU	2 vCPU	4 vCPU	8 vCPU	16 vCPU	32 vCPU
FGT-VM08	OK	OK	OK	OK	The FortiGate-VM uses 8 vCPUs for traffic and management and does not use the rest.	

You can provision a VM instance based on the number of interfaces you need and license the FortiGate-VM for only the processors you need.

Public compared to private clouds

The behavior differs between private and public clouds:

- Private clouds (VMware ESXi/KVM/Xen/Microsoft Hyper-V): both licensed vCPUs and RAM are affected. FortiOS does not have licensed RAM size restrictions. Having at least 4 GB of RAM for proper FortiGate-VM operation is recommended, especially if unified threat management, zero trust network access, or proxy is enabled.
- Public clouds (AWS/Azure/GCP/OCI/AliCloud): only licensed vCPU is affected.

For example, you can activate FG-VM02 on a FGT-VM with 4 vCPUs and there is no limit on the RAM size when running on a private VM platform.

Likewise, you can activate FG-VM02 on a FGT-VM c5.2xlarge EC2 instance with 8 vCPUs running on AWS. Only 2 vCPU is consumable, and there is no limit on the RAM size. You can refer to licenses for public clouds as bring your own license.

Preparing for deployment

This documentation assumes that before deploying the FortiGate-VM on the KVM virtual platform, you have addressed the following requirements:

Virtual environment

You have installed the KVM software on a physical server with sufficient resources to support the FortiGate-VM and all other VMs deployed on the platform.

If you configure the FortiGate-VM to operate in transparent mode, or include it in a FortiGate clustering protocol high availability cluster, configure any virtual switches to support the FortiGate-VM's operation before you create the FortiGate-VM.

Connectivity

The FortiGate-VM requires an internet connection to contact FortiGuard to validate its license. A FortiGate-VM in a closed environment must be able to connect to a FortiManager to validate the FortiGate-VM license. See Validating the FortiGate-VM license with FortiManager on an air-gapped environment on page 21.

Configuring resources

Before you start the FortiGate-VM for the first time, ensure that you have configured the following resources as the FortiGate-VM license specifies:

- Disk sizes
- CPUs
- RAM
- Network settings

To configure settings for FortiGate-VM on the server:

1. In the Virtual Machine Manager, locate the VM name, then select *Open* from the toolbar.
2. Select *Add Hardware*.
3. In the *Add Hardware* window select *Storage*.
4. Select *Create a disk image on the computer's harddrive* and set the size to 30 GB.

If you know your environment will expand in the future, increasing the hard disk size beyond 30 GB is recommended. The VM license limit is 2 TB.

5. Enter the following information:

Device type	Virtio disk
Cache mode	Default
Storage format	raw

Even though raw is the storage format listed, the qcow2 format is also supported.

6. Select *Network* to configure or add more network interfaces. The *Device type* must be *Virtio*. A new VM includes one network adapter by default. You can add more through the *Add Hardware* window. FortiGate-VM requires four network adapters. You can configure network adapters to connect to a virtual switch or to network adapters on the host computer.
7. Select *Finish*.

Registering the FortiGate-VM

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

To register the FortiGate-VM:

1. Log in to the Customer Service & Support site using a support account, or create an account.
2. In the main page, under *Asset*, select *Register Now*.
3. In the *Registration* page, enter the registration code that you received via email, and select *Register* to access the registration form.
4. If you register the s-series subscription model, the site prompts you to select one of the following:
   a. Click *Register* to newly register the code to acquire a new serial number with a new license file.
   b. Click *Renew* to renew and extend the licensed period on top of the existing serial number, so that all features on the VM node continue working uninterrupted upon license renewal.
5. Complete and submit the registration form.
6. In the registration acknowledgment page, click the *License File Download* link.
7. Save the license file (`.lic`) to your local computer. See Uploading the FortiGate-VM license on page 20 or Validating the FortiGate-VM license with FortiManager on an air-gapped environment on page 21 for information about uploading the license file to your FortiGate-VM via the GUI.

Downloading the FortiGate-VM 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.

To download the FortiGate-VM deployment package:

1. In the *Select Product* drop-down menu, select *FortiGate*.
2. In the Select Platform drop-down menu, select KVM.
3. 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.
4. Click the *Download* button and save the file.

For more information, see the FortiGate datasheet.

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

The FORTINET.out.kvm.zip contains only fortios.qcow2, the FortiGate-VM system hard disk in qcow2 format. You must manually:

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

Cloud-init

You can use the `cloud-init` service for customizing a prepared image of a virtual installation. The `cloud-init` service is built into the virtual instances of FortiGate-VM that are found on the support site so that you can use them on a VM platform that supports the use of the service. To customize the installation of a new FortiGate-VM instance, you must combine the seed image from the support site with user data information customized for each new installation.

Hypervisor platforms such as QEMU/KVM, BSD, and Hyper-V support the use of this service on most major Linux distributions. A number of cloud-based environments such as VMware and AWS also support it.

You can use the `cloud-init` service to help install different instances based on a common seed image by assigning hostnames, adding SSH keys, and settings particular to the specific installation. You can add other more general customizations such as the running of post install scripts.

While `cloud-init` is the service used to accomplish the customized installations of VMs, various other programs, depending on the platform, are used to create the customized ISOs used to create the images that will build the FortiGate-VM.

For more information, see the cloud-init documentation.

The basic steps of the process are:

1. Ensure that the needed software is on the system.
2. Prepare the files to customize the seed image.
3. Collect the customizing files into a single folder.
4. Convert the folder to an ISO image.
5. Install the image on the VM platform.

Installing the software

For instructions on installing software, you should refer to the installation instructions for your desired Linux distribution. For example, if using Red Hat Enterprise Linux, see Product Documentation for Red Hat Enterprise Linux.

You must install an environment that can run `cloud-init` and `mkisofs`.

Preparing the files for the customized image

Preparing the user_data file

The `cloud-init` service passes a script to newly created VMs, in this case FortiGate-VM. The title of the file is `user_data`. All configuration on the FortiGate is done through the configuration file so components of the the scripts follow the syntax of the configuration file or commands being entered through the CLI.

The following example content is from a basic user_data file:

```
#this is for fgt init config file. Can only handle fgt config.
config sys interface
   edit port1
      set mode dhcp
      set allowaccess http https ssh ping telnet
   next
end
config sys dns
   set primary 8.8.8.8
   unset secondary
```

```
end
config sys global
   set hostname cloud-init-test
end
```

License file

The other file that is used to configure the customized install contains the license key. Take the license key that you receive from Fortinet and place it into a text file. This file is named `0000` without any extension.

Preparing the folder

There are no requirements for where to place the holding folder that will be used to create the new ISO image, but there are requirements as to the folder structure within the folder. The cloud-init must find specific content in specific folders to work correctly. The folder structure should be as follows:

```
<holding folder>
   /openstack
      /content
         0000
      /latest
         user_data
```

It may seem counterintuitive to use the folder name `openstack` in an instance where the target VM platform is not OpenStack, but a number of utilities are common to both OpenStack and KVM environments.

Converting the folder to an ISO image

Once you have your user_data file and the license key file, you can create an ISO image containing all of the files that are used to customize the seed image of the FortiGate-VM. This is done using the `mkisofs` utility.

The syntax of the command is:

```
mkisofs [options] [-o <filename of new ISO> pathspec [pathspec...]
```

Some of the options are:

Option	Description
-o <filename>	Sets the filename of the resulting ISO image file.
pathspec [pathspec...]	direction to the folder(s) that are to be included in the ISO image file. Separate the paths with a space.
-input-charset	Input charset that defines the characters used in local file names. To get a list of valid charset names, use the command `mkisofs -input-charset help`. To get a 1:1 mapping, you may use `default` as charset name.
- R	Generate SUSP and RR records using the Rock Ridge protocol to further describe the files on the iso9660 filesystem.

Option	Description
- r	This is like the -R option, but file ownership and modes are set to more useful values. The uid and gid are set to zero, because they are usually only useful on the author's system, and not useful to the client. All the file read bits are set true, so that files and directories are globally readable on the client.

Example:

The iso-folder holds the data structure for the new ISO image. The `/home/username/test` folder contains the `iso-folder`folder.The name for the new ISO image is fgt-bootstrap.iso.

```
cd /home/username/test
sudo mkisofs -R -r -o fgt-bootstrap.iso iso-folder
```

Installing the ISO in the VM platform

The following table contains some of the more common options used in setting up a FortiGate-VM image. Not all of them are required. To get a complete listing of the options, at the command prompt, type in the command `virt-install --help` or `virt-install -h`.

Option	Description
--connect <option>	This connects the VM image to a non-default VM platform. If one is not specified, libvirt will attempt to choose the most suitable default platform. Some valid options are: • qemu:///system Creates KEM and QEMU guests run by the system. This is the most common option. • qem:///session Creates KEM and QEMU guests run as a regular user. • xen:/// For connecting to Xen.
--name <name> -n <name>	New guest VM instance name. This must be unique amongst all guests known to the hypervisor on the connection, including those not currently active. To redefine an existing guest, use the `virsh` tool
--memory <option>	Memory to allocate for the guest, in MiB. (This deprecates the `-r`/`--ram` option.) Sub-options are available, like: • `maxmemory` • `hugepages` • `hotplugmemorymax` • `hotplugmemoryslots`
--vcpus <options>	Number of virtual cpus to configure for the guest.

Option	Description
	If 'maxvcpus' is specified, the guest will be able to hotplug up to MAX vcpus while the guest is running, but will start up with VCPUS. Use `--vcpus=?` to see a list of all available sub options.
--cdrom <options> -c <options>	File or device used as a virtual CD-ROM device. It can be path to an ISO image, or to a CDROM device. It can also be a URL from which to fetch/access a minimal boot ISO image. The URLs take the same format as described for the "--location" argument. If a cdrom has been specified via the "--disk" option, and neither "--cdrom" nor any other install option is specified, the "--disk" cdrom is used as the install media.
--location <options> -l <options>	Distribution tree installation source. virt-install can recognize certain distribution trees and fetches a bootable kernel/initrd pair to launch the install. With libvirt 0.9.4 or later, network URL installs work for remote connections. virt-install will download kernel/initrd to the local machine, then upload the media to the remote host. This option requires the URL to be accessible by both the local and remote host. `--location` allows things like --extra-args for kernel arguments, and using --initrd-inject. If you want to use those options with CDROM media, you have a few options: • Run virt-install as root and do --location ISO • Mount the ISO at a local directory, and do --location DIRECTORY • Mount the ISO at a local directory, export that directory over local http, and do --location http://localhost/DIRECTORY The "LOCATION" can take one of the following forms: • `http://host/path` An HTTP server location containing an installable distribution image. • `ftp://host/path` An FTP server location containing an installable distribution image. • nfs:host:/path or nfs://host/path An NFS server location containing an installable distribution image. This requires running virt-install as root. • DIRECTORY Path to a local directory containing an installable distribution image. Note that the directory will not be accessible by the guest after initial boot, so the OS installer will need another way to access the rest of the install media. • ISO Mount the ISO and probe the directory. This requires running virt-install as root, and has the same VM access caveat as DIRECTORY.
--import	Skip the OS installation process, and build a guest around an existing disk image. The device used for booting is the first device specified via "`--disk`" or "`--filesystem`".

Option	Description
--disk <options>	Specifies media to use as storage for the guest, with various options. The general format of a disk string is `--disk opt1=val1,opt2=val2,...` When using multiple options, separate each option with a comma (no spaces before or after the commas). Example options: • `size` size (in GiB) to use if creating new storage example: size=10 • `path` A path to some storage media to use, existing or not. Existing media can be a file or block device. Specifying a non-existent path implies attempting to create the new storage, and will require specifying a 'size' value. Even for remote hosts, virt-install will try to use libvirt storage APIs to automatically create the given path. If the hypervisor supports it, path can also be a network URL, like http://example.com/some-disk.img . For network paths, they hypervisor will directly access the storage, nothing is downloaded locally. • `format` Disk image format. For file volumes, this can be 'raw', 'qcow2', 'vmdk', etc. See format types in libvirt: Storage Management for possible values. This is often mapped to the driver_type value as well. If not specified when creating file images, this will default to `.qcow2`. If creating storage, this will be the format of the new image. If using an existing image, this overrides libvirt's format auto-detection. The disk option deprecates `-f/--file`, `-s/--file-size`, `--nonsparse`, and `--nodisks`. Use `--disk=?` to see a list of all available sub options.
--network <options> -w <options>	Connect the guest to the host network. The value for <options> can take one of 4 formats: • `bridge=BRIDGE` Connect to a bridge device in the host called "BRIDGE". Use this option if the host has static networking config & the guest requires full outbound and inbound connectivity to/from the LAN. Also use this if live migration will be used with this guest. • `network=NAME` Connect to a virtual network in the host called "NAME". Virtual networks can be listed, created, deleted using the "virsh" command line tool. In an unmodified install of "libvirt" there is usually a virtual network with a name of "default". Use a virtual network if the host has dynamic networking (eg NetworkManager), or using wireless. The guest will be NATed to the LAN by whichever connection is active.

Option	Description
	• `type=direct,source=IFACE[,source_mode=MODE]` Direct connect to host interface IFACE using macvtap. • `user` Connect to the LAN using SLIRP. Only use this if running a QEMU guest as an unprivileged user. This provides a very limited form of NAT. • `none` Tell `virt-install` not to add any default network interface. Use `--network=?` to see a list of all available sub options. See details at libvirt: Domain XML format - Network interfaces. This option deprecates `-m/--mac`, `-b/--bridge`, and `--nonetworks`
--noautoconsole	This stops the system from automatically trying to connect to the guest console. The default behavior is to launch `virt-viewer` to run a GUI console or run the `virsh console` command to display a text version of the console.

Example:

This will take the iso image made in the previous file and install it into the VM platform giving the name Example_VM to the FortiGate-VM instance.

```
virt-install --connect qemu:///system --noautoconsole --name Example_VM --memory 1024 --vcpus 1 -
      -import --disk fortios.qcow2,size=3 --disk fgt-logs.qcow2,size=3 --disk
      /home/username/test/fgt-bootstrap.iso,device=cdrom,bus=ide,format=raw,cache=none --network
      bridge=virbr0,model=virtio
```

What the options are doing in the example:

- `--connect qemu:///system` - connects the image to the QEMU platform.
- `--noautoconsole` - prevents a console from automatically starting up after the installation is completed.
- `--name Example_VM` - sets the name of the FortiGate-VM to Example_VM.
- `--memory 1024` - allocates 1024 MB (1 GB) of RAM to the VM.
- `--vcpus 1` - allocates 1 virtual cpu to the VM.
- `--import` - instead of running an installation process, builds a VM around an existing VM image based on the first instance of the --disk setting.
- `--disk fortios.qcow2` - uses the fortios.qcow2 file to build a disk with the included content, into the VM.
- `--disk fgt-logs.qcow2,size=3` - Because no file with the name fgt-logs.qcow2 is found, an empty disk is created in the VM with a size of 3 GB.
- `--disk /home/username/test/fgt-bootstrap.iso,device=cdrom,bus=ide,format=raw,cache=none` - sets up a virtual cdrom drive as if it was on an IDE bus holding a virtual CD in it with no cache and the data in RAW format. This virtual CD is based on the file fgt-bootstrap.iso. While is will work if the command is run from the folder that holds the file, you can also include the path to the file.
- `--network bridge=virbr0,model=virtio` - connects the VM to the virtual bridge virbr0 using a virtio model virtual network adapter.

Before running the command, ensure that QEMU/KVM is running properly.

You should be able to start the instance by running the command:

```
virsh --connect qemu:///system start Example_VM
```

Deployment

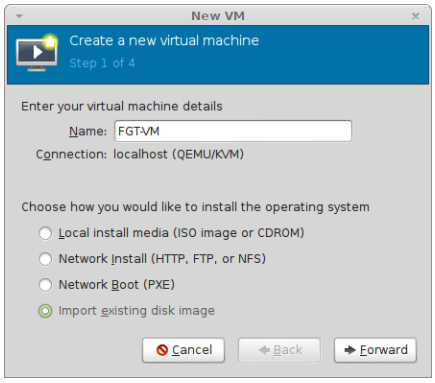
The deployment uses cases in this document describe the tasks required to deploy a FortiGate-VM virtual appliance on a KVM server. Before you deploy a FortiGate-VM, ensure that you have met the requirements that Preparing for deployment on page 7 describes and that you have extracted the correct deployment package to a folder on the local computer. See Downloading the FortiGate-VM deployment package on page 9.

After you deploy a FortiGate-VM and upload a full license to replace the default evaluation license, you can power on the FortiGate-VM and test connectivity.

Deploying the FortiGate-VM

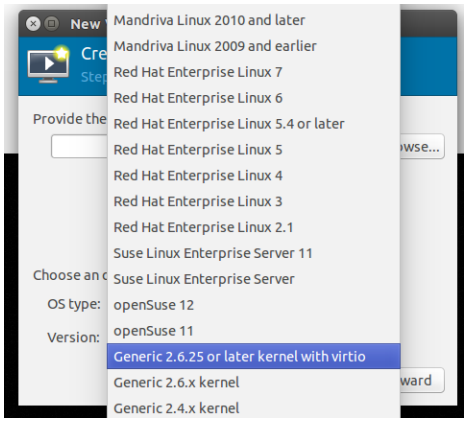
To create the FortiGate-VM virtual machine:

1. Launch Virtual Machine Manager (virt-manager) on your KVM host server. The *Virtual Machine Manager* homepage opens.
2. In the toolbar, select *Create a new virtual machine*.

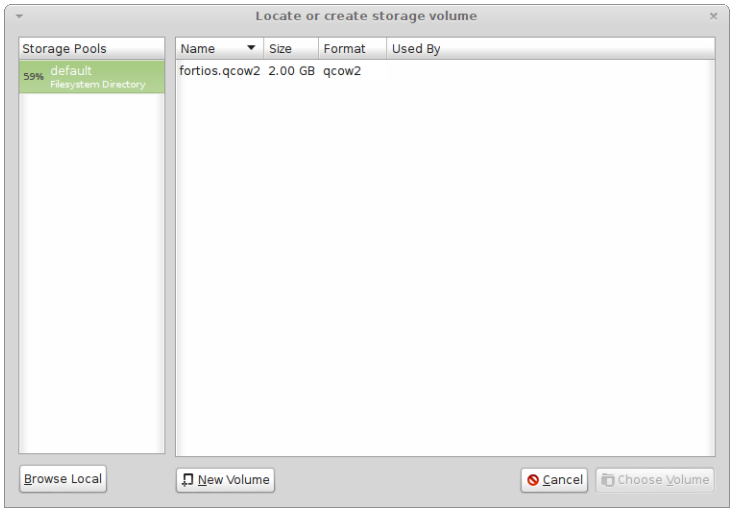


3. Enter a *Name* for the VM, FGT-VM for example.
4. Ensure that *Connection* is localhost (this is the default).
5. Select *Import existing disk image*.
6. Select *Forward*.
7. In *OS Type* select *Linux*.

8. In *Version*, select a Generic version with virtio.



9. Select *Browse*.



10. If you copied the fortios.qcow2 file to `/var/lib/libvirt/images`, it will be visible on the right. If you saved it somewhere else on your server, select Browse Local and find it.
11. Choose *Choose Volume*.
12. Select *Forward*.
13. Specify the amount of memory and number of CPUs to allocate to this VM. Whether or not the amounts can exceed the license limits will depend on the FortiOS version. See FortiGate-VM virtual licenses and resources on page 5
14. Select *Forward*.
15. Expand *Advanced options*. A new VM includes one network adapter by default.
16. Select a network adapter on the host computer. Optionally, set a specific MAC address for the virtual network interface.
17. Set *Virt Type* to *virtio* and *Architecture* to *qcow2*.
18. Select *Finish*.

Initial settings

After you deploy a FortiGate-VM on the KVM server, perform the following tasks:

- Connect the FortiGate-VM to the network so that it can process network traffic and maintain license validity.
- Connect to FortiGate-VM GUI via a web browser for easier administration.
- Ensure that the full license file is uploaded to the FortiGate-VM.
- If you are in a closed environment, enable validation of the FortiGate-VM license against a FortiManager on your network.

Network configuration

The first time you start the FortiGate-VM, you will have access only through the console window of your KVM server environment. After you configure one FortiGate network interface with an IP address and administrative access, you can access the FortiGate-VM GUI.

Configuring port 1

VM platform or hypervisor management environments include a guest console window. On the FortiGate-VM, this provides access to the FortiGate console, equivalent to the console port on a hardware FortiGate unit. Before you can access the GUI, you must configure FortiGate-VM port1 with an IP address and administrative access.

To configure the port1 IP address:

1. In your hypervisor manager, start the FortiGate-VM and access the console window. You may need to press *Enter* to see a login prompt.
2. At the FortiGate-VM login prompt enter the username `admin`. By default there is no password. Press Enter.
3. Using CLI commands, configure the port1 IP address and netmask:
```
config system interface
   edit port1
      set mode static
      set ip 192.168.0.100 255.255.255.0
   next
end
```
4. To configure the default gateway, enter the following CLI commands:
```
config router static
   edit 1
      set device port1
      set gateway <class_ip>
   next
end
```

You must configure the default gateway with an IPv4 address. FortiGate-VM must access the Internet to contact the FortiGuard Distribution Network to validate its license.

5. To configure your DNS servers, enter the following CLI commands:

```
config system dns
   set primary <Primary DNS server>
   set secondary <Secondary DNS server>
end
```

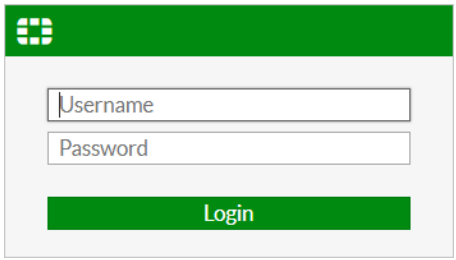
The default DNS servers are `208.91.112.53` and `208.91.112.52`.

Connecting to the FortiGate-VM GUI

You connect to the FortiGate-VM GUI via a web browser by entering the IP address assigned to the port 1 interface (see Configuring port 1 on page 19) in the browser location field. You must enable HTTP and/or HTTPS access and administrative access on the interface to ensure that you can connect to the GUI. If you only enabled HTTPS access, enter "https://" before the IP address.

When you use HTTP rather than HTTPS to access the GUI, certain web browsers may display a warning that the connection is not private.

On the FortiGate-VM GUI login screen, enter the default username "admin", then select *Login*. FortiOS does not assign a default password to the admin user.



Fortinet recommends that you configure a password for the admin user as soon as you log in to the FortiGate-VM GUI for the first time.

Uploading the FortiGate-VM license

Before using the FortiGate-VM, you must enter the license file that you downloaded from Customer Service & Support upon registration.

To upload the FortiGate-VM license file via the GUI:

1. Do one of the following to access the license upload window:
   - In *Dashboard* > *Status* window, in the *Virtual Machine* widget, click the *FGVMEV* (FortiGate-VM Evaluation) *License* icon. This reveals a menu of selections to take you directly to the *FortiGate VM*

License window or to the *FortiGuard Details* window.
- Go to *System* > *FortiGuard*. In the *License Information* section, go to the *Virtual Machine* row and click *FortiGate VM License*.

2. In the *Evaluation License* dialog, select *Enter License*. The license upload page opens.
3. Select *Upload* and locate the license file (`.lic`) on your computer.
4. Select *OK* to upload the license file.
5. Refresh the browser to log in.
6. Enter `admin` in the *Name* field and select *Login*. The VM registration status appears as valid in the *License Information* widget after the FortiGuard Distribution Network or FortiManager for closed networks validates the license.

> Modern browsers can have an issue with allowing connecting to a FortiGate if the encryption on the device is too low. If this happens, use an FTP/TFTP server to apply the license.

To upload the FortiGate-VM license file via the CLI:

You can also upload the license file using the following CLI command:

```
execute restore vmlicense {ftp | tftp} <filenmame string> <ftp server>[:ftp port]
```

Example:

The following is an example output when using a TFTP server to install a license:

```
execute restore vmlicense tftp license.lic 10.0.1.2
   This operation will overwrite the current VM license!Do you want to continue? (y/n)y
   Please wait...Connect to tftp server 10.0.1.2 ...
   Get VM license from tftp server OK.
   VM license install succeeded.
   Rebooting firewall.
```

> This command automatically reboots the firewall without giving you a chance to back out or delay the reboot.

Validating the FortiGate-VM license with FortiManager on an air-gapped environment

You can validate your FortiGate-VM license with some FortiManager models. To determine whether your FortiManager has the VM activation feature, see the FortiManager datasheet.

In an air-gapped environment, the FortiGate and FortiManager do not have internet connectivity.

To validate your FortiGate-VM with your FortiManager:

1. To configure your FortiManager as a closed network, enter the following CLI command on your FortiManager:
   ```
   config fmupdate publicnetwork
   ```

```
   set status disable
end
```

2. To configure FortiGate-VM to use FortiManager as its override server, enter the following CLI commands on your FortiGate-VM:

```
config system central-management
   set mode normal
   set type fortimanager
   set fmg <FortiManager IPv4 address>
   config server-list
      edit 1
         set server-type update
         set server-address <FortiManager IPv4 address>
      end
   end
   set fmg-source-ip <Source IPv4 address when connecting to the FortiManager>
   set include-default-servers disable
   set vdom <Enter the virtual domain name to use when communicating with the FortiManager>
end
```

3. Request the account entitlement file from Fortinet Customer Service & Support as Requesting account entitlement files describes.
4. Upload the received entitlement file to FortiManager as Uploading account entitlement files describes.
5. Load the FortiGate-VM license file in the FortiOS GUI:
   a. Go to *System* > *Dashboard* > *Status*.
   b. In the *License Information* widget, in the *Registration Status* field, select *Update*.
   c. Browse for the `.lic` license file and select *OK*.
6. To activate the FortiGate-VM license, enter the `execute update-now` command on your FortiGate-VM.
7. To check the FortiGate-VM license status, enter the following CLI commands on your FortiGate-VM:

```
get system status
   Version: Fortigate-VM v8.0,buildXXXX,120910
   Virus-DB: 15.00361(2024-08-24 17:17)
   Extended DB: 15.00000(2024-08-24 17:09)
   Extreme DB: 14.00000(2024-08-24 17:10)
   IPS-DB: 3.00224(2024-10-28 16:39)
   FortiClient application signature package: 1.456(2024-01-17 18:27)
   Serial-Number: FGVM02Q105060000
   License Status: Valid
   BIOS version: 04000002
   Log hard disk: Available
   Hostname: Fortigate-VM
   Operation Mode: NAT
   Current virtual domain: root
   Max number of virtual domains: 10
   Virtual domains status: 1 in NAT mode, 0 in TP mode
   Virtual domain configuration: disable
   FIPS-CC mode: disable
   Current HA mode: standalone
   Distribution: International
   Branch point: 511
   Release Version Information: MR3 Patch 4
   System time: Wed Jan 18 11:24:34 2024

diagnose hardware sysinfo vm full
   UUID: 564db33a29519f6b1025bf8539a41e92
   valid: 1
```

```
status: 1
code: 200 (If the license is a duplicate, code 401 displays)
warn: 0
copy: 0
received: 45438
warning: 0
recv: 201201201918
dup:
```

Licensing timeout

In closed environments without internet access, you must license the FortiGate-VM offline using a FortiManager as a license server. If the FortiGate-VM cannot validate its license within the 30-day license timeout period, the FortiGate discards all packets, effectively ceasing operation as a firewall.

The license status goes through some changes before it times out. See VM license.

There is only a single log entry after the FortiGate-VM cannot access the license server for the license expiration period. When you search the logs for the reason that the FortiGate is offline, there is no long error log list that draws attention to the issue. There is only one entry.

Testing connectivity

You can now power on your FortiGate-VM.

Select the FortiGate-VM in the VM list. In the toolbar, select *Console* then select *Start*.

The ping utility is the usual method to test connectivity to other devices. For this, you need the console on the FortiGate-VM.

In FortiOS, the command for the ping utility is `execute ping` followed by the IP address you want to connect to.

Before you configure the FortiGate-VM for use in production, ensure that connections between it and all required resources can be established.

- If the FortiGate-VM will provide firewall protection between your network and the internet, verify that it can connect to your internet access point and to resources on the internet.
- If the FortiGate-VM is part of a Fortinet Security Fabric, verify that it can connect to all devices in the Fabric.
- Verify that each node on your network can connect to the FortiGate-VM.

Configuring your FortiGate-VM

For information about configuring and operating the FortiGate-VM after successful deployment and startup on the hypervisor, see the FortiOS Administration Guide.

HA

FortiGate-VM high availability (HA) supports having two virtual machines (VM) in an HA cluster on the same physical server or on different physical servers. In both cases, the two VMs run on the same hypervisor, such as KVM. The primary consideration is that all interfaces involved can communicate efficiently over TCP/IP connection sessions.

Heartbeat

There are two options for setting up the HA heartbeat: unicast and broadcast. Broadcast is the default HA heartbeat configuration. However, the broadcast configuration may not be ideal for FortiGate-VM because it may require special settings on the host. In most cases, the unicast configuration is preferable.

Differences between the unicast and broadcast heartbeat setups are:

- The unicast method does not change the FortiGate-VM interface MAC addresses to virtual MAC addresses.
- Unicast HA only supports two FortiGate-VMs.
- Unicast HA heartbeat interfaces must be connected to the same network and you must add IP addresses to these interfaces.

Unicast

You can configure the unicast settings in the FortiOS CLI:

```
config system ha
   set unicast-hb {enable/disable}
   set unicast-hb-peerip {Peer heartbeat interface IP address}
end
```

Setting	Description
`unicast-hb`	Enable or disable default unicast HA heartbeat.
`unicast-hb-peerip`	IP address of the HA heartbeat interface of the other FortiGate-VM in the HA cluster.

Broadcast

Broadcast HA heartbeat packets are non-TCP packets that use Ethertype values 0x8892, 0x8891, and 0x8890. These packets use automatically assigned link-local IPv4 addresses in the 169.254.0.x range for HA heartbeat

interface IP addresses.

For FortiGate-VMs to support a broadcast HA heartbeat configuration, you must configure the virtual switches that connect heartbeat interfaces to operate in promiscuous mode and support MAC address spoofing.

In addition, you must configure the VM platform to allow MAC address spoofing for the FortiGate-VM data interfaces. This is required because in broadcast mode, the FGCP applies virtual MAC addresses to FortiGate data interfaces, and these virtual MAC addresses mean that matching interfaces of the FortiGate-VM instances in the cluster have the same virtual MAC addresses.

Promiscuous mode

KVM's Virtual Machine Manager does not have the ability to set a virtual network interface to promiscuous mode. This is done to the host's physical network interface. When KVM creates a VM, it also creates a tap interface as well as a new MAC address for it. Once the host's physical interface is set to promiscuous mode, it must be connected to a bridge device that the VM uses to connect to the network outside of the host.

Because this configuration is done on the host and not the VM, the methodology depends on the host's operating system distribution and version.

Setting up the network interfaces and bridge devices requires using an account with root privileges.

Optimizing FortiGate-VM performance

This section describes FortiGate-VM and KVM performance optimization techniques that can improve your FortiGate-VM performance by optimizing the hardware and the KVM host environment for FortiGate-VM's network- and CPU-intensive performance requirements.

SR-IOV

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

FortiGate-VMs benefit from SR-IOV because SR-IOV optimizes network performance and reduces latency and CPU usage. FortiGate-VMs do not use KVM features that are incompatible with SR-IOV, so you can enable SR-IOV without negatively affecting your FortiGate-VM. SR-IOV implements an I/O memory management unit (IOMMU) to differentiate between different traffic streams and apply memory and interrupt translations between the physical functions (PF) and virtual functions (VF).

Setting up SR-IOV on KVM involves creating a PF for each physical network card in the hardware platform. Then, you create VFs that allow FortiGate-VMs to communicate through the PF to the physical network card. VFs are actual PCIe hardware resources and only a limited number of VFs are available for each PF.

SR-IOV hardware compatibility

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

To enable SR-IOV, your KVM platform must run on hardware that is compatible with SR-IOV and with FortiGate-VMs. FortiGate-VMs require network cards that are compatible with the supported drivers. See PF and VF SR-IOV driver and virtual SPU support for supported driver versions. As well, the host hardware CPUs must support second level address translation (SLAT).

For optimal SR-IOV support, install the most up-to-date network drivers. Fortinet recommends i40e/Iavf drivers because they provide four TxRx queues for each VF and ixgbevf only provides two TxRx queues.

Enabling SR-IOV support for Intel systems

Use the following steps to enable SR-IOV support for KVM host systems that use Intel CPUs. These steps involve enabling and verifying Intel VT-d specifications in the BIOS and Linux kernel. You can skip these steps if you have already enabled VT-d.

On an Intel host PC, Intel VT-d BIOS settings provide hardware support for directly assigning a physical device to a VM.

To enable SR-IOV support for Intel systems:

1. View the BIOS settings of the host machine and enable VT-d settings if they are not already enabled. You may have to review the manufacturer's documentation for details.
2. Activate Intel VT-d in the Linsux kernel by adding the `intel_iommu=on` parameter to the kernel line in the `/boot/grub/grub.conf` file. For example:
```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-330.x86_645)
   root (hd0,0)
   kernel /vmlinuz-2.6.32-330.x86_64 ro root=/dev/VolGroup00/LogVol00 rhgb quiet intel_iommu=on
   initrd /initrd-2.6.32-330.x86_64.img
```
3. Restart the system.

Enabling SR-IOV support for AMD systems

Use the following steps to enable SR-IOV support for KVM host systems that use AMD CPUs. These steps involve enabling the AMD IOMMU specifications in the BIOS and Linux kernel. You can skip these steps if you have already enabled AMD IOMMU.

On an AMD host PC, IOMMU BIOS settings provide hardware support for directly assigning a physical device to a VM.

To enable SR-IOV support for AMD systems:

1. View the BIOS settings of the host machine and enable IOMMU settings if they are not already enabled. You may have to review the manufacturer's documentation for details.
2. Append `amd_iommu=on` to the kernel command line in `/boot/grub/grub.conf` so that AMD IOMMU specifications are enabled when the system starts up.
3. Restart the system.

Verifying that Linux and KVM can find SR-IOV-enabled PCI devices

You can use the `lspci` command to view the list of PCI devices and verify that your SR-IOV supporting network cards are on the list. The following output example shows some example entries for the Intel 82576 network card:

```
# lspci
03:00.0 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
03:00.1 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
```

Optionally modifying the SR-IOV kernel modules

If the device is supported, the kernel should automatically load the driver kernel module. You can enable optional parameters using the `modprobe` command. For example, the Intel 82576 network interface card uses the `igb` driver kernel module.

```
# modprobe igb [<option>=<VAL1>,<VAL2>,
   # lsmod |grep
   igb 87592
   dca 6708 1 igb
```

Attaching an SR-IOV network device to a FortiGate-VM

You can enable SR-IOV for a FortiGate-VM by creating a Virtual Function (VF) and attaching the VF to your FortiGate-VM.

Activating and verifying an SR-IOV VF

The `max_vfs` parameter of the `igb` module allocates the maximum number of VFs. The `max_vfs` parameter causes the driver to spawn multiple VFs.

Before activating the maximum number of VFs enter the following command to remove the `igb` module:

```
# modprobe -r igb
```

Restart the `igb` module with `max_vfs` set to the maximum supported by your device. For example, the valid range for the Intel 82576 network interface card is 0 to 7. To activate the maximum number of VFs supported by this device enter:

```
# modprobe igb max_vfs=7
```

Make the VFs persistent by adding `options igb max_vfs=7` to any file in `/etc/modprobe.d`. For example:

```
# echo "options igb max_vfs=7" >>/etc/modprobe.d/igb.conf
```

Verify the new VFs. For example, you could use the following `lspci` command to list the newly added VFs attached to the Intel 82576 network device. Alternatively, you can use `grep` to search for `Virtual Function`, to search for devices that support VFs.

```
# lspci | grep 82576 0b:00.0 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection
      (rev 01)
0b:00.1 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection(rev 01)
0b:10.0 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.1 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.2 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.3 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.4 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.5 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.6 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:10.7 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:11.0 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:11.1 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:11.2 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:11.3 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:11.4 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
0b:11.5 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
```

Use the `-n` parameter of the `lspci` command to find the identifier for the PCI device. The PFs correspond to `0b:00.0` and `0b:00.1`. All VFs have `Virtual Function` in the description.

Verify that the devices exist with virsh

The `libvirt` service must recognize a PCI device before you can add it to a VM. `libvirt` uses a similar notation to the `lspci` output.

Use the `virsh nodedev-list` command and the `grep` command to filter the Intel 82576 network device from the list of available host devices. In the example, `0b` is the filter for the Intel 82576 network devices. This may vary for your system and may result in additional devices.

```
# virsh nodedev-list | grep 0b
pci_0000_0b_00_0
pci_0000_0b_00_1
pci_0000_0b_10_0
pci_0000_0b_10_1
pci_0000_0b_10_2
pci_0000_0b_10_3
pci_0000_0b_10_4
pci_0000_0b_10_5
pci_0000_0b_10_6
pci_0000_0b_11_7
pci_0000_0b_11_1
pci_0000_0b_11_2
pci_0000_0b_11_3
pci_0000_0b_11_4
pci_0000_0b_11_5
```

The serial numbers for the VFs and PFs should be in the list.

Get device details with virsh

The `pci_0000_0b_00_0` is one of the PFs and `pci_0000_0b_10_0` is the first corresponding VF for that PF. Use `virsh nodedev-dumpxml` to get device details for both devices.

Example device details for the `pci_0000_0b_00_0` PF device:

```
 # virsh nodedev-dumpxml pci_0000_0b_00_0
<device>
   <name>pci_0000_0b_00_0</name>
   <parent>pci_0000_00_01_0</parent>
   <driver>
      <name>igb</name>
   </driver>
   <capability type='pci'>
      <domain>0</domain>
      <bus>11</bus>
      <slot>0</slot>
      <function>0</function>
      <product id='0x10c9'>82576 Gigabit Network Connection</product>
      <vendor id='0x8086'>Intel Corporation</vendor>
```

```
  </capability>
</device>
```

Example device details for the `pci_0000_0b_10_0` PF device:

```
# virsh nodedev-dumpxml pci_0000_0b_10_0
<device>
   <name>pci_0000_0b_10_0</name>
   <parent>pci_0000_00_01_0</parent>
   <driver>
      <name>igbvf</name>
   </driver>
   <capability type='pci'>
      <domain>0</domain>
      <bus>11</bus>
      <slot>16</slot>
      <function>0</function>
      <product id='0x10ca'>82576 Virtual Function</product>
      <vendor id='0x8086'>Intel Corporation</vendor>
   </capability>
</device>
```

You must use this information to specify the bus, slot, and function parameters when you add the VF to a FortiGate-VM. A convienent way to do this is to create a temporary xml file and copy the following text into that file.

```
<interface type='hostdev' managed='yes'>
     <source>
       <address type='pci' domain='0' bus='11' slot='16' function='0'/>
     </source>
   </interface>
```

You can also include additional information about the VF such as a MAC address, VLAN tag, and so on. If you specify a MAC address, the VF will always have this MAC address. If you do not specify a MAC address, the system generates a new one each time the FortiGate-VM restarts.

Add the VF to a FortiGate-VM

Enter the following command to add the VF to a FortiGate-VM. This configuration attaches the new VF device immediately and saves it for subsequent FortiGate-VM restarts.

```
virsh attach-device MyFGTVM <temp-xml-file> --config
```

Where *MyFGTVM* is the name of the FortiGate-VM for which to enable SR-IOV, and `<temp-xml-file>` is the temporary XML file containing the VF configuration.

After this configuration, when you start up the FortiGate-VM it detects the SR-IOV VF as a new network interface.

Interrupt affinity

In addition to enabling SR-IOV in the VM host, to fully take advantage of SR-IOV performance improvements you must 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 is 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 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
end
```

Where:

- `<interrupt-name>` is 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 associate all of the interrupts for a given interface with the same CPU affinity mask.
- `<cpu-affinity-mask>` is 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
```

Packet-distribution affinity

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 redistribution 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 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
end
```

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 redistributed to CPUs according to the 0xE CPU affinity mask.

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

Configuring QAT on a FortiGate-VM with SR-IOV

You can configure the FortiGate-VM to use a card called Intel QuickAssist (QAT) through IPsec VPN, which accelerates traffic-handling performance. Configuring QAT consists of the following steps:

Checking the host information

Check the host information by entering the following commands:

```
[root@localhost net]# cat /etc/os-release
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"
CENTOS_MANTISBT_PROJECT="CentOS-7"
CENTOS_MANTISBT_PROJECT_VERSION="7"
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="7"
[root@localhost net]# lscpu
Architecture: x86_64
```

```
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 96
On-line CPU(s) list: 0-95
Thread(s) per core: 2
Core(s) per socket: 24
Socket(s): 2
NUMA node(s): 2
Vendor ID: GenuineIntel
CPU family: 6
Model: 85
Model name: Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz
Stepping: 4
CPU MHz: 2700.000
BogoMIPS: 5400.00
Virtualization: VT-x
L1d cache: 32K
L1i cache: 32K
L2 cache: 1024K
L3 cache: 33792K
NUMA node0 CPU(s):
      0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,
      66,68,70,72,74,76,78,80,82,84,86,88,90,92,94
NUMA node1 CPU(s):
      1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,55,57,59,61,63,65,
      67,69,71,73,75,77,79,81,83,85,87,89,91,93,95
Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx
      fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts
      rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl
      vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_
      deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch epb cat_l3 cdp_l3 intel_
      ppin intel_pt ssbd mba ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_
      adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap
      clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc cqm_
      mbm_total cqm_mbm_local dtherm ida arat pln pts pku ospke spec_ctrl intel_stibp flush_l1d
```

Enabling SR-IOV and creating the VF

Setting up SR-IOV on KVM involves creating a PF for each physical network card in the hardware platform. VFs allow FortiGate-VMs to communicate through the physical network card. VFs are actual PCIe hadware resources and only a limited number of VFs are available for each PF. Each QAT addon card creates three PFs with a maximum capacity of 16 VFs each. Ensure that the VM and QAT card are on the same NUMA node.

To configure SR-IOV and create VFs, see SR-IOV support for virtual networking.

After enabling SR-IOV and setting the VF numbers, review the QAT and NIC VFs:

```
[root@localhost net]# lshw -c processor -businfo
Bus info Device Class Description
========================================================
cpu@0 processor Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz
cpu@1 processor Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz
pci@0000:b1:00.0 processor C62x Chipset QuickAssist Technology
pci@0000:b1:01.0 processor Intel Corporation
pci@0000:b1:01.1 processor Intel Corporation
pci@0000:b1:01.2 processor Intel Corporation
```

```
pci@0000:b1:01.3 processor Intel Corporation
pci@0000:b1:01.4 processor Intel Corporation
pci@0000:b1:01.5 processor Intel Corporation
pci@0000:b1:01.6 processor Intel Corporation
pci@0000:b1:01.7 processor Intel Corporation
pci@0000:b1:02.0 processor Intel Corporation
pci@0000:b1:02.1 processor Intel Corporation
pci@0000:b1:02.2 processor Intel Corporation
pci@0000:b1:02.3 processor Intel Corporation
pci@0000:b1:02.4 processor Intel Corporation
pci@0000:b1:02.5 processor Intel Corporation
pci@0000:b1:02.6 processor Intel Corporation
pci@0000:b1:02.7 processor Intel Corporation
pci@0000:b3:00.0 processor C62x Chipset QuickAssist Technology
pci@0000:b3:01.0 processor Intel Corporation
pci@0000:b3:01.1 processor Intel Corporation
pci@0000:b3:01.2 processor Intel Corporation
pci@0000:b3:01.3 processor Intel Corporation
pci@0000:b3:01.4 processor Intel Corporation
pci@0000:b3:01.5 processor Intel Corporation
pci@0000:b3:01.6 processor Intel Corporation
pci@0000:b3:01.7 processor Intel Corporation
pci@0000:b3:02.0 processor Intel Corporation
pci@0000:b3:02.1 processor Intel Corporation
pci@0000:b3:02.2 processor Intel Corporation
pci@0000:b3:02.3 processor Intel Corporation
pci@0000:b3:02.4 processor Intel Corporation
pci@0000:b3:02.5 processor Intel Corporation
pci@0000:b3:02.6 processor Intel Corporation
pci@0000:b3:02.7 processor Intel Corporation
pci@0000:b5:00.0 processor C62x Chipset QuickAssist Technology
pci@0000:b5:01.0 processor Intel Corporation
pci@0000:b5:01.1 processor Intel Corporation
pci@0000:b5:01.2 processor Intel Corporation
pci@0000:b5:01.3 processor Intel Corporation
pci@0000:b5:01.4 processor Intel Corporation
pci@0000:b5:01.5 processor Intel Corporation
pci@0000:b5:01.6 processor Intel Corporation
pci@0000:b5:01.7 processor Intel Corporation
pci@0000:b5:02.0 processor Intel Corporation
pci@0000:b5:02.1 processor Intel Corporation
pci@0000:b5:02.2 processor Intel Corporation
pci@0000:b5:02.3 processor Intel Corporation
pci@0000:b5:02.4 processor Intel Corporation
pci@0000:b5:02.5 processor Intel Corporation
pci@0000:b5:02.6 processor Intel Corporation
pci@0000:b5:02.7 processor Intel Corporation
```

The above example has one physical QAT card with three QAT accelerators:

```
[root@localhost ~]# lspci | grep Ethernet
86:00.0 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ (rev 02)
86:00.1 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ (rev 02)
86:02.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:02.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:02.2 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:02.3 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:02.4 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
```

```
86:02.5 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:02.6 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:02.7 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0a.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0a.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0a.2 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0a.3 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0a.4 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0a.5 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0a.6 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0a.7 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:00.0 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ (rev 02)
88:00.1 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ (rev 02)
88:02.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:02.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:02.2 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:02.3 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:02.4 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:02.5 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:02.6 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:02.7 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:0a.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:0a.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:0a.2 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:0a.3 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:0a.4 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:0a.5 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:0a.6 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:0a.7 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
[root@localhost ~]# lshw -c network -businfo | grep X710
pci@0000:86:00.0 p5p1 network Ethernet Controller X710 for 10GbE SFP+
pci@0000:86:00.1 p5p2 network Ethernet Controller X710 for 10GbE SFP+
pci@0000:88:00.0 p7p1 network Ethernet Controller X710 for 10GbE SFP+
pci@0000:88:00.1 p7p2 network Ethernet Controller X710 for 10GbE SFP+
```

Deploying the FortiGate-VM on KVM

Deploying a FortiGate-VM on KVM with QAT support consists of the following steps:

1. Create the FortiGate-VM on KVM.
2. Inject SR-IOV network VFs into the FortiGate-VM.
3. Configure interrupt affinities.

To create the FortiGate-VM on KVM:

To create a FortiGate-VM on KVM, refer to Deploying the FortiGate-VM on page 17.

To inject SR-IOV network VFs into the FortiGate-VM:

You can inject an SR-IOV network VF into a Linux KVM VM using one of the following ways:

- Connecting an SR-IOV VF to a KVM VM by directly importing the VF as a PCI device using the PCI bus information that the host OS assigned to it when it was created

- Using the Virtual Manager GUI
- Adding an SR-IOV network adapter to the KVM VM as a VF network adapter connected to a macvtap on the host
- Creating an SR-IOV VF network adapter using a KVM virtual network pool of adapters

See Configure SR-IOV Network Virtual Functions in Linux* KVM*.

In the following example, virtual network adapter pools were created for KVM04:

```
[root@localhost ~]# vnlist
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 default              active     yes           yes
 p5p1-pool            active     no            no
 p5p2-pool            active     no            no
 p7p1-pool            active     no            no
 p7p2-pool            active     no            no

[root@localhost ~]# vf2pf


Virtual Functions on Intel Corporation Ethernet Controller X710 for 10GbE SFP+. (p5p1):
PCI BDF         Interface
=======         =========
0000:86:02.0    p5p1_0
0000:86:02.1    p5p1_1
0000:86:02.2    p5p1_2
0000:86:02.3    p5p1_3
0000:86:02.4    p5p1_4
0000:86:02.5    p5p1_5
0000:86:02.6    p5p1_6
0000:86:02.7    p5p1_7

Virtual Functions on Intel Corporation Ethernet Controller X710 for 10GbE SFP+. (p5p2):
PCI BDF         Interface
=======         =========
0000:86:0a.0    p5p2_0
0000:86:0a.1    p5p2_1
0000:86:0a.2    p5p2_2
0000:86:0a.3    p5p2_3
0000:86:0a.4    p5p2_4
0000:86:0a.5    p5p2_5
0000:86:0a.6    p5p2_6
0000:86:0a.7    p5p2_7

Virtual Functions on Intel Corporation Ethernet Controller X710 for 10GbE SFP+. (p7p1):
PCI BDF         Interface
=======         =========
0000:88:02.0    p7p1_0
0000:88:02.1    p7p1_1
0000:88:02.2    p7p1_2
0000:88:02.3    p7p1_3
0000:88:02.4    p7p1_4
```

```
0000:88:02.5    p7p1_5
0000:88:02.6    p7p1_6
0000:88:02.7    p7p1_7

Virtual Functions on Intel Corporation Ethernet Controller X710 for 10GbE SFP+. (p7p2):
PCI BDF         Interface
=======         =========
0000:88:0a.0    p7p2_0
0000:88:0a.1    p7p2_1
0000:88:0a.2    p7p2_2
0000:88:0a.3    p7p2_3
0000:88:0a.4    p7p2_4
0000:88:0a.5    p7p2_5
0000:88:0a.6    p7p2_6
0000:88:0a.7    p7p2_7
```

The XML file is as follows. `<cputune>` locks the virtual CPUs to the same NUMA node, while `<hostdev mode='subsystem' type='pci' managed='yes'>` creates the QAT VFs:

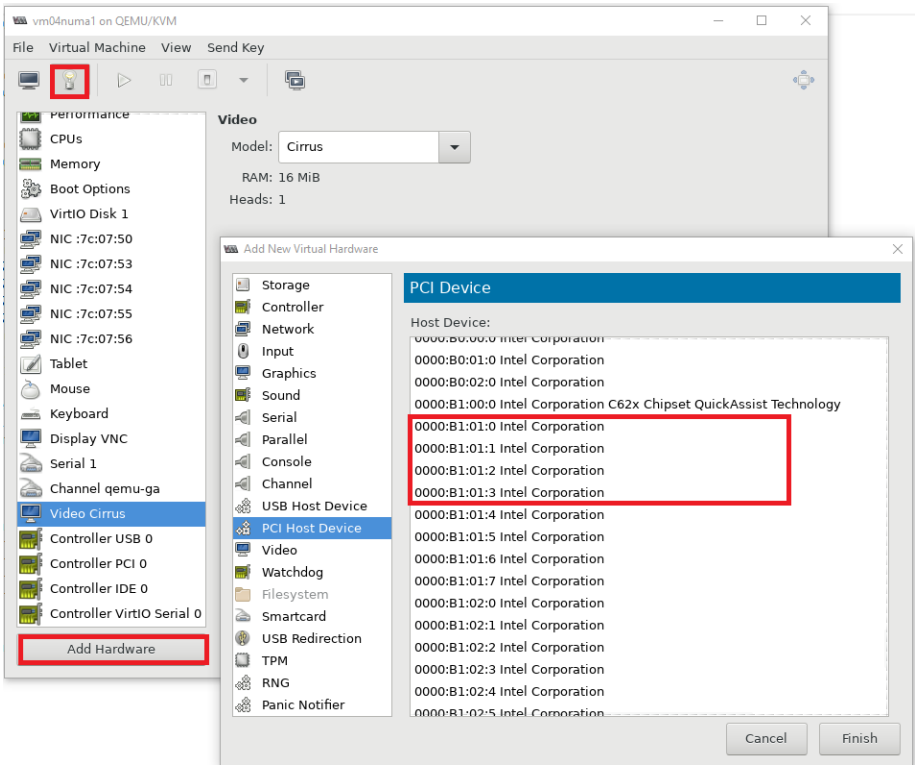
```
[root@localhost ~]# virsh dumpxml vm04numa1
<domain type='kvm'>
  <name>vm04numa1</name>
  <uuid>fc5e1cec-8b4e-4bb8-9f89-e86f1abfffeb</uuid>
  <memory unit='KiB'>6291456</memory>
  <currentMemory unit='KiB'>6291456</currentMemory>
  <memoryBacking>
    <hugepages>
      <page size='1048576' unit='KiB'/>
    </hugepages>
  </memoryBacking>
  <vcpu placement='static'>4</vcpu>
  <cputune>
    <vcpupin vcpu='0' cpuset='17'/>
    <vcpupin vcpu='1' cpuset='19'/>
    <vcpupin vcpu='2' cpuset='21'/>
    <vcpupin vcpu='3' cpuset='23'/>
    <emulatorpin cpuset='17,19,21,23'/>
  </cputune>
  <numatune>
    <memory mode='strict' nodeset='1'/>
  </numatune>
  <os>
    <type arch='x86_64' machine='pc-i440fx-rhel7.6.0'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
  </features>
  <cpu mode='custom' match='exact' check='partial'>
    <model fallback='allow'>Skylake-Server-IBRS</model>
  </cpu>
  <clock offset='utc'>
```

```
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='hpet' present='no'/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
    <suspend-to-mem enabled='no'/>
    <suspend-to-disk enabled='no'/>
  </pm>
  <devices>
    <emulator>/usr/libexec/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'/>
      <source file='/var/lib/libvirt/images/vm04numa1.0984'/>
      <target dev='vda' bus='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
    </disk>
    <controller type='usb' index='0' model='ich9-ehci1'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x7'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci1'>
      <master startport='0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'
multifunction='on'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci2'>
      <master startport='2'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x1'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci3'>
      <master startport='4'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x2'/>
    </controller>
    <controller type='pci' index='0' model='pci-root'/>
    <controller type='ide' index='0'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1'/>
    </controller>
    <controller type='virtio-serial' index='0'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
    </controller>
    <interface type='direct'>
      <mac address='52:54:00:7c:07:50'/>
      <source dev='em1' mode='bridge'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
    </interface>
    <interface type='network'>
      <mac address='52:54:00:7c:07:53'/>
      <source network='p5p1-pool'/>
      <model type='i40e'/>
```

```
      <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x0'/>
    </interface>
    <interface type='network'>
      <mac address='52:54:00:7c:07:54'/>
      <source network='p5p2-pool'/>
      <model type='i40e'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x0b' function='0x0'/>
    </interface>
    <interface type='network'>
      <mac address='52:54:00:7c:07:55'/>
      <source network='p7p1-pool'/>
      <model type='i40e'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x0c' function='0x0'/>
    </interface>
    <interface type='network'>
      <mac address='52:54:00:7c:07:56'/>
      <source network='p7p2-pool'/>
      <model type='i40e'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x0d' function='0x0'/>
    </interface>
    <serial type='tcp'>
      <source mode='bind' host='0.0.0.0' service='10004'/>
      <protocol type='telnet'/>
      <target type='isa-serial' port='0'>
        <model name='isa-serial'/>
      </target>
    </serial>
    <console type='tcp'>
      <source mode='bind' host='0.0.0.0' service='10004'/>
      <protocol type='telnet'/>
      <target type='serial' port='0'/>
    </console>
    <channel type='unix'>
      <target type='virtio' name='org.qemu.guest_agent.0'/>
      <address type='virtio-serial' controller='0' bus='0' port='1'/>
    </channel>
    <input type='tablet' bus='usb'>
      <address type='usb' bus='0' port='1'/>
    </input>
    <input type='mouse' bus='ps2'/>
    <input type='keyboard' bus='ps2'/>
    <graphics type='vnc' port='5904' autoport='no' listen='0.0.0.0' keymap='en-us'>
      <listen type='address' address='0.0.0.0'/>
    </graphics>
    <video>
      <model type='cirrus' vram='16384' heads='1' primary='yes'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
    </video>
    <hostdev mode='subsystem' type='pci' managed='yes'>
      <source>
        <address domain='0x0000' bus='0xb1' slot='0x01' function='0x0'/>
      </source>
```

```
      <address type='pci' domain='0x0000' bus='0x00' slot='0x0e' function='0x0'/>
    </hostdev>
    <hostdev mode='subsystem' type='pci' managed='yes'>
      <source>
        <address domain='0x0000' bus='0xb1' slot='0x01' function='0x1'/>
      </source>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x0f' function='0x0'/>
    </hostdev>
    <hostdev mode='subsystem' type='pci' managed='yes'>
      <source>
        <address domain='0x0000' bus='0xb1' slot='0x01' function='0x2'/>
      </source>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x10' function='0x0'/>
    </hostdev>
    <hostdev mode='subsystem' type='pci' managed='yes'>
      <source>
        <address domain='0x0000' bus='0xb1' slot='0x01' function='0x3'/>
      </source>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x11' function='0x0'/>
    </hostdev>
    <memballoon model='virtio'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
    </memballoon>
  </devices>
</domain>
```

The following shows the Virtual Manager GUI:



To configure interrupt affinities:

The example topology is as follows:

TestCenter----KVM port2----KVM port3===IPSEC tunnel===KVM port5-----KVM port4----TestCenter

After configuring the IPsec tunnel in the KVM, you must configure interrupt affinities and CPU masking to improve throughput for FortiGate-VM platforms. In this example, you can configure interrupt affinities as follows. You must manually set `cpu-affinity` for each QAT VF. Otherwise, they only go through the first CPU.

```
config system affinity-interrupt
    edit 1
        set interrupt "i40evf-port2-TxRx-0"
        set affinity-cpumask "0x01"
    next
    edit 2
        set interrupt "i40evf-port2-TxRx-1"
        set affinity-cpumask "0x01"
    next
    edit 3
        set interrupt "i40evf-port2-TxRx-2"
        set affinity-cpumask "0x01"
    next
    edit 4
        set interrupt "i40evf-port2-TxRx-3"
        set affinity-cpumask "0x01"
    next
    edit 5
        set interrupt "i40evf-port3-TxRx-0"
        set affinity-cpumask "0x02"
    next
    edit 6
        set interrupt "i40evf-port3-TxRx-1"
        set affinity-cpumask "0x02"
    next
    edit 7
        set interrupt "i40evf-port3-TxRx-2"
        set affinity-cpumask "0x02"
    next
    edit 8
        set interrupt "i40evf-port3-TxRx-3"
        set affinity-cpumask "0x02"
    next
    edit 9
        set interrupt "i40evf-port4-TxRx-0"
        set affinity-cpumask "0x04"
    next
    edit 10
        set interrupt "i40evf-port4-TxRx-1"
        set affinity-cpumask "0x04"
    next
    edit 11
        set interrupt "i40evf-port4-TxRx-2"
```

```
        set affinity-cpumask "0x04"
    next
    edit 12
        set interrupt "i40evf-port4-TxRx-3"
        set affinity-cpumask "0x04"
    next
    edit 13
        set interrupt "i40evf-port5-TxRx-0"
        set affinity-cpumask "0x08"
    next
    edit 14
        set interrupt "i40evf-port5-TxRx-1"
        set affinity-cpumask "0x08"
    next
    edit 15
        set interrupt "i40evf-port5-TxRx-2"
        set affinity-cpumask "0x08"
    next
    edit 16
        set interrupt "i40evf-port5-TxRx-3"
        set affinity-cpumask "0x08"
    next
    edit 17
        set interrupt "qat_00:14.00"
        set affinity-cpumask "0x01"
    next
    edit 18
        set interrupt "qat_00:15.00"
        set affinity-cpumask "0x02"
    next
    edit 19
        set interrupt "qat_00:16.00"
        set affinity-cpumask "0x04"
    next
    edit 20
        set interrupt "qat_00:17.00"
        set affinity-cpumask "0x08"
    next
end
```

This way, all four CPUs are balanced:

```
FGVM04TM19001384 (global) # get system performance status
CPU states: 0% user 2% system 0% nice 63% idle 0% iowait 0% irq 35% softirq
CPU0 states: 0% user 1% system 0% nice 69% idle 0% iowait 0% irq 30% softirq
CPU1 states: 0% user 3% system 0% nice 55% idle 0% iowait 0% irq 42% softirq
CPU2 states: 0% user 1% system 0% nice 72% idle 0% iowait 0% irq 27% softirq
CPU3 states: 0% user 2% system 0% nice 59% idle 0% iowait 0% irq 39% softirq
Memory: 6131096k total, 1092248k used (17.8%), 4908480k free (80.1%), 130368k freeable (2.1%)
Average network usage: 3825681 / 3813407 kbps in 1 minute, 1392107 / 1389299 kbps in 10 minutes,
      632093 / 631744 kbps in 30 minutes
Average sessions: 37 sessions in 1 minute, 31 sessions in 10 minutes, 24 sessions in 30 minutes
Average session setup rate: 0 sessions per second in last 1 minute, 0 sessions per second in last 10
      minutes, 0 sessions per second in last 30 minutes
Virus caught: 0 total in 1 minute
```

```
IPS attacks blocked: 0 total in 1 minute
Uptime: 0 days, 0 hours, 7 minutes
```

For how CPU interrupt affinity optimizes FortiGate-VM performance, see Technical Note: Optimize FortiGate-VM performance by configuring CPU interrupt affinity.

```
FGVM04TM19001384 (global) # diagnose hardware sysinfo interrupts
           CPU0       CPU1       CPU2       CPU3
  0:         26          0          0          0   IO-APIC-edge      timer
  1:          9          0          0          0   IO-APIC-edge      i8042
  4:         15          0          0          0   IO-APIC-edge      serial
  8:          0          0          0          0   IO-APIC-edge      rtc
  9:          0          0          0          0   IO-APIC-fasteoi   acpi
 10:          0          0          0          0   IO-APIC-fasteoi   uhci_hcd:usb3, uhci_hcd:usb4
 11:         16          0          0          0   IO-APIC-fasteoi   ehci_hcd:usb1, uhci_hcd:usb2
 12:          3          0          0          0   IO-APIC-edge      i8042
 14:          0          0          0          0   IO-APIC-edge      ata_piix
 15:          0          0          0          0   IO-APIC-edge      ata_piix
 40:          0          0          0          0   PCI-MSI-edge      virtio1-config
 41:        629          0          0          0   PCI-MSI-edge      virtio1-requests
 42:          0          0          0          0   PCI-MSI-edge      virtio3-config
 43:        978          0          0          0   PCI-MSI-edge      virtio3-input.0
 44:          1          0          0          0   PCI-MSI-edge      virtio3-output.0
 45:     255083          0          0          0   PCI-MSI-edge      qat_00:14.00
 46:         17     537891          0          0   PCI-MSI-edge      qat_00:15.00
 47:         17          0    1244511          0   PCI-MSI-edge      qat_00:16.00
 48:         17          0          0    1224563   PCI-MSI-edge      qat_00:17.00
 49:        173          0          0          0   PCI-MSI-edge      i40evf-0000:00:0a.0:mbx
 50:     119912          0          0          0   PCI-MSI-edge      i40evf-port2-TxRx-0
 51:          1     200309          0          0   PCI-MSI-edge      i40evf-port2-TxRx-1
 52:          1          0     538905          0   PCI-MSI-edge      i40evf-port2-TxRx-2
 53:          1          0          0     532128   PCI-MSI-edge      i40evf-port2-TxRx-3
 54:        172          0          0          0   PCI-MSI-edge      i40evf-0000:00:0b.0:mbx
 55:     254849          0          0          0   PCI-MSI-edge      i40evf-port3-TxRx-0
 56:          1     443186          0          0   PCI-MSI-edge      i40evf-port3-TxRx-1
 57:          1          0     600793          0   PCI-MSI-edge      i40evf-port3-TxRx-2
 58:          1          0          0     850484   PCI-MSI-edge      i40evf-port3-TxRx-3
 59:        172          0          0          0   PCI-MSI-edge      i40evf-0000:00:0c.0:mbx
 60:      72971          0          0          0   PCI-MSI-edge      i40evf-port4-TxRx-0
 61:          1     376044          0          0   PCI-MSI-edge      i40evf-port4-TxRx-1
 62:          1          0     531843          0   PCI-MSI-edge      i40evf-port4-TxRx-2
 63:          1          0          0     539088   PCI-MSI-edge      i40evf-port4-TxRx-3
 64:        172          0          0          0   PCI-MSI-edge      i40evf-0000:00:0d.0:mbx
 65:     197132          0          0          0   PCI-MSI-edge      i40evf-port5-TxRx-0
 66:          1     421851          0          0   PCI-MSI-edge      i40evf-port5-TxRx-1
 67:          1          0     850741          0   PCI-MSI-edge      i40evf-port5-TxRx-2
 68:          1          0          0     600896   PCI-MSI-edge      i40evf-port5-TxRx-3
NMI:          0          0          0          0   Non-maskable interrupts
LOC:      41936      46038      41842      46773   Local timer interrupts
SPU:          0          0          0          0   Spurious interrupts
PMI:          0          0          0          0   Performance monitoring interrupts
IWI:          0          0          0          0   IRQ work interrupts
RES:     282399       1450        787        751   Rescheduling interrupts
```

```
CAL:         46        106         62        107   Function call interrupts
TLB:         10         11          5          8   TLB shootdowns


FGVM04TM19001384 (vdom-1) # diagnose vpn ipsec status
All ipsec crypto devices in use:
QAT:
    Encryption (encrypted/decrypted)
        null             : 0                0
        des              : 0                0
        3des             : 0                0
        aes              : 48025403         29252461
        aes-gcm          : 0                0
        aria              : 0                0
        seed             : 0                0
        chacha20poly1305  : 0                0
    Integrity (generated/validated)
        null              : 0                0
        md5              : 0               0
        sha1             : 47967645        29250506
        sha256           : 0                0
        sha384           : 0                0
        sha512           : 0                0

SOFTWARE:
    Encryption (encrypted/decrypted)
        null             : 0                0
        des              : 0                0
        3des             : 0                0
        aes              : 0                0
        aes-gcm          : 0                0
        aria             : 0                0
        seed             : 0                0
        chacha20poly1305 : 0                0
    Integrity (generated/validated)
        null             : 0                0
        md5              : 0               0
        sha1             : 0                0
        sha256           : 0                0
        sha384           : 0                0
        sha512           : 0                0

Test Results:
1360 bytes IPSEC packet loss results with QAT in KVM04: 10,654m(v6.2 build 0984)
```

Enhancing FortiGate-VM performance with DPDK and vNP offloading

DPDK and vNP enhance FortiGate-VM performance by offloading part of packet processing to user space while using a kernel bypass solution within the operating system. You must enable and configure DPDK with FortiOS CLI commands.

FortiOS 8.0 supports DPDK for KVM environments.

FortiOS 8.0 supports IPv6.

The DPDK+vNP offloading-capable version of FortiOS only supports FortiGate instances with multiple vCPUs. Minimum required RAM sizes differ from those on regular FortiGate-VM models without offloading. Allocating as much RAM size as the licensed limit for maximum performance, as shown, is recommended. FortiOS 8.0 does not restrict RAM size by license. Therefore, you can allocate as much memory as desired on 8.0-based DPDK-enabled FortiGate-VMs:

Model name	RAM size (licensed limit)
FG-VM02(v)	No restriction
FG-VM04(v)	
FG-VM08(v)	
FG-VM16(v)	
FG-VM32(v)	

You can enable DPDK up to 64 vCPUs.

For 7.6.3 and later versions, a CPU:RAM ratio of 1:2 is recommended for all FortiGate models, regardless of the number of cores. For 7.6.2 and earlier versions, a CPU-to-RAM ratio of 1:3 is recommended for low-end models (FGT-VM instances with 8 cores or fewer). For instances with 16 cores or more, a 1:2 ratio is recommended.

FortiOS supports encrypted traffic for IPsec VPN and not for SSL VPN. Disabling the DPDK option using the CLI or adopting regular FortiGate-VM builds is recommended when using SSL VPN. For encrypted traffic support for IPsec VPN with DPDK, FortiOS also adds support for the following:

- CBC cipher suite
- Increasing maximum number of IPsec VPN tunnels
- Terminating tunnel on LAG

Enabling DPDK+vNP offloading may result in fewer concurrent sessions when under high load than when DPDK+vNP offloading is not enabled and the same FortiGate-VM license is used.

Enabling DPDK in polling mode results in high CPU usage.

Enabling DPDK+vNP offloading using the FortiOS CLI

Provided that you obtained a DPDK+vNP offloading-capable FortiOS build, the following provides the configuration to enable the capability:

- DPDK global settings on page 47
- DPDK CPU settings on page 50
- DPDK diagnostic commands on page 52

FortiOS supports SNMP to poll DPDK-related status. For details, see the corresponding MIB file that Fortinet provides.

DPDK global settings

To enable DPDK operations for the FortiGate-VM:

1. In the FortiOS CLI, enter the following commands to enable DPDK operation:
```
config dpdk global
   set status enable
   set interface port1
end
```
2. The CLI displays the following message:
```
Status and interface changes will trigger system reboot and take effect after the reboot.
Do you want to continue? (y/n)
```
Press y to reboot the device.

Before system reboot, you must check if other DPDK settings are configured properly. You must enable at least one network interface for DPDK. The example enables port1. You can enable other interfaces as desired. If you do not set an interface, a prompt displays and the change is discarded. See To enable a network interface to run DPDK operation: on page 48.

To enable DPDK multiqueue mode:

Enabling multiqueue at network RX/TX helps DPDK better balance the workload onto multiple engines.

1. In the FortiOS CLI, enter the following commands to enable DPDK operation:
```
config dpdk global
   set multiqueue enable
end
```

2. The CLI displays the following message:
   ```
   Multiqueue change will trigger IPS restart and will take effect after the restart. Traffic may
         be interrupted briefly.
   Do you want to continue? (y/n)
   ```
   Press y to reboot IPS engine.

To set the percentage of main memory allocated to DPDK huge pages and packet buffer pool:

You can configure the amount of main memory (as a percentage) allocated to huge pages, which are dedicated to DPDK use. You can also configure the amount of main memory (as a percentage) allocated to the DPDK packet buffer pool.

Enter the following commands to set these amounts:

```
config dpdk global
   set hugepage-percentage [X]
   set mbufpool-percentage [Y]
end
```

Changing `mbufpool-percentage` requires IPS engine to restart (no reboot).

`set mbufpool-percentage` is deprecated in FortiOS 7.6.3 and later versions.

Huge page memory is mounted at system startup and remains mounted as long as the FortiGate-VM is running. Packet buffer pool memory is drawn from huge pages. Therefore, the packet buffer pool amount (Y) must not exceed the huge pages amount (X).

In practice, it is mandated that Y is lesser than or equal to X - 5 to leave 5% memory overhead for other DPDK data structures. The range of X is between 10 and 50, and the range of Y is between 5 and 45.

Setting X too high may force FortiOS to enter conserve mode. Setting X too low may result in insufficient memory for DPDK operation and failure of initialization.

During FortiOS DPDK Helper environment initialization, RTE memory zones are drawn from huge memory pages. The system tries to reserve continuous memory chunks for these memory zones with best effort. Therefore, the amount of huge page memory is slightly larger than the amount of memory that RTE memory zones use. To gain insight into how RTE memory zones reserve memory spaces, run the `diagnose dpdk statistics show memory` command.

To enable a network interface to run DPDK operation:

You must enable at least one network interface to run DPDK operation.

```
config dpdk global
   set interface "portX" "portY"
end
```

You must enable at least one network interface for DPDK. Otherwise, DPDK early initialization during system startup fails and falls back to a disabled state. In this example, if there are two network interfaces that you intend to use, you can specify `set interface port1 port2`.

Enabling DPDK is only available for physical network interfaces.

To enable DPDK monitor engine:

Enabling DPDK monitor engine is optional.

1. In the FortiOS CLI, enter the following commands to enable DPDK monitor engine:
```
config dpdk global
   set sleep-on-idle enable
end
```
2. The CLI displays the following message:
```
sleep-on-idle change will trigger IPS restart and will take effect after the restart. Traffic
      may be interrupted briefly.
Do you want to continue? (y/n)
```
Press y to reboot IPS engine.

By default, DPDK monitor engine is disabled. When enabled, only one DPDK engine polls DPDK-enabled interfaces. When packets arrive, corresponding DPDK entries are activated. This helps when services other than firewall or IPS engine, such as antivirus, WAD, or web filter, are running and performance degradation is observed while DPDK performance statistics show that DPDK engines are not fully used. Latency may increase due to the time needed to activate the proper DPDK engines by the monitor engine.

To enable elastic buffer (temporary memory buffer):

Enabling elastic buffer is optional.

1. In the FortiOS CLI, enter the following commands to enable elastic memory buffer:
```
config dpdk global
   set elasticbuffer enable
end
```
2. The CLI displays the following message:
```
elasticbuffer change will trigger IPS restart and will take effect after the restart. Traffic
      may be interrupted briefly.
Do you want to continue? (y/n)
```
Press y to reboot IPS engine.

By default, elastic buffer is disabled. When enabled, an elastic buffer takes effect to store packets in case of traffic burst. The feature helps to reduce packet drops when received packets peak under system overload by storing packets in the buffer and processing them afterward. This feature is experimental.

To enable per-session accounting:

Enabling per-session accounting is optional.

1. In the FortiOS CLI, enter the following commands to enable per session accounting:
```
config dpdk global
   set per-session-accounting enable|disable|traffic-log-only
end
```

2. The CLI displays the following message:
   ```
   per-session-accounting change will trigger IPS restart and will take effect after the restart.
         Traffic may be interrupted briefly.
   Do you want to continue? (y/n)
   ```
   Press y to reboot IPS engine.

By default, per-session accounting is configured only for traffic logs, which results in per-session accounting being enabled when you enable traffic logging in a policy.

Per-session accounting is a logging feature that allows FortiOS to report the correct bytes per packet numbers per session for sessions offloaded to a vNP process. This information appears in traffic log messages, FortiView, and `diagnose` commands. Per-session accounting can affect vNP offloading performance. You should only enable per-session accounting if you need the accounting information. A similar feature is available for physical FortiGate NP6 processors.

DPDK CPU settings

On the FortiGate-VM, a DPDK engine is attached to an IPS engine, which shares the same process and is mapped to a CPU. A processing pipeline of four stages handles a packet from RX to TX:

1. DPDK RX
2. vNP
3. IPS
4. DPDK TX

You can freely determine the CPUs enabled for each pipeline stage by running the following commands:

```
config dpdk cpus
   set [X] [Y]
end
```

Here X is one of the pipeline stages: `rx-cpus`, `vnp-cpus`, `ips-cpus`, and `tx-cpus`.

Y is a string expression of CPU IDs, which contains comma-delimited individual CPU IDs or ranges of CPU IDs separated by a dash.

The example enables CPUs 0, 2, 4, 6, 7, 8, 9. 10, and 15 to run the vNP pipeline stage:

```
set vnp-cpus 0,2,4,6-10,15
```

In FortiOS 8.0, Y can also be a special token string `all`, which means to use all available CPUs to run that pipeline stage. The system automatically determines the number of available CPUs. `all` is the default value of each pipeline stage's CPU setting.

The example uses all available CPUs to run the IPS pipeline stage:

```
set ips-cpus all
```

You must enable at least one CPU for each pipeline stage. Otherwise, DPDK early initialization fails.

Isolating CPUs that DPDK engine uses

To improve DPDK performance, you can isolate the CPUs that the DPDK engine uses from other services, except for processes that have affinity explicitly set by a user configuration or their implementation.

```
config dpdk cpus
    set isolated-cpus <CPUs>
end
```

Input CPU IDs or ranges separated by commas, or `none` to not isolate CPUs for DPDK. For example, enter `1-3,5,6-9` to isolate CPUs 1,2,3,5,6,7,8, and 9.

Both the lower and upper bounds of a range must be explicitly specified. The range of isolated CPU IDs is [1-0], and CPU ID 0 is not allowed. The isolated CPU IDs must be DPDK enabled CPUs.

Reserving CPUs for DPDK may not always produce optimal performance. Users should experiment with a combination that works best for their deployment. For example, on a FortiGate VM with eight CPUs, the following configurations could be used to optimize different deployments:

To optimize CPS with logging to disk (session/sec):

```
config dpdk cpus
    set rx-cpus "1-1"
    set vnp-cpus "1-7"
    set ips-cpus "1-7"
    set tx-cpus "1-7"
    set isolated-cpus "1-7"
end
```

To optimize proxy antivirus performance:

```
config dpdk cpus
    set rx-cpus "1-5"
    set vnp-cpus "1-5"
    set ips-cpus "1-5"
    set tx-cpus "1-5"
    set isolated-cpus "1-5"
end
```

To optimize proxy DLP performance:

```
config dpdk cpus
    set rx-cpus "1-5"
    set vnp-cpus "1-5"
    set ips-cpus "1-5"
    set tx-cpus "1-5"
end
```

DPDK diagnostic commands

To view DPDK-related logs:

Enter the following command to view DPDK-related logs:

```
diagnose dpdk log show [log type]
```

Currently, FortiOS provides two DPDK-related logs:

Log	Records kept
`early-init`	DPDK's early initialization procedure during system startup
`fdh`	Warnings and errors met during the initialization of FortiOS DPDK helper (FDH), i.e. DPDK engines

Ensure that you double-check whether DPDK early initialization was successful. If successful, the end of the early-init log shows the following:

```
DPDK sanity test passed
```

If the DPDK early initialization was unsuccessful, refer to DPDK global settings on page 47 to see if the DPDK-related options were properly set.

The early init-log also keeps records of last-edited DPDK configuration, enabled CPUs/ports, binding/unbinding of drivers, device PCI info, and so on.

To view DPDK-related statistics:

Enter the following command to view DPDK-related statistics:

```
diagnose dpdk statistics show [stats type]
```

Currently, FortiOS provides four types of DPDK-related statistics:

- `engine`: provides per-DPDK engine statistics
- `port`: provides per-DPDK port statistics
- `vnp`: provides per-vNP engine statistics
- `memory`: provides a quick view of memory size reserved by each RTE memory zone

To reset statistics, enter the following command:

```
diagnose dpdk statistics clear all
```

This command resets engine and port statistics to zeroes, but does not affect vNP and memory statistics.

To check if traffic is properly forwarded, load-balanced, and offloaded to fast path:

A useful way to check whether traffic is properly forwarded is to check the port statistics. This shows the number of received/transmitted/dropped packets in each DPDK-enabled port.

```
--------------------------------------------------------------------------------
FortiOS DPDK Helper Port Stats
--------------------------------------------------------------------------------

                                      Total              port2

---------- DPDK RX Stage ---------------------------------------------------------
dpdkrx_rx_pkts:                           0                  0

---------- DPDK TX Stage ---------------------------------------------------------
dpdktx_tx_pkts:                           0                  0
dpdktx_drop_pkts:                         0                  0
dpdktx_drop_oversized_pkt:                0                  0
```

Checking engine statistics is helpful in understanding how traffic is load-balanced among DPDK engines at each pipeline stage.

```
--------------------------------------------------------------------------------
FortiOS DPDK Helper Engine Stats
--------------------------------------------------------------------------------

                                Total
CPU ID:

---------- DPDK RX Stage ---------------------------------------------------------
dpdkrx_rx_pkts:                     2
dpdkrx_tx_pkts:                     2
dpdkrx_drop_pkts:                   0

---------- VNP Stage -------------------------------------------------------------
vnp_rx_pkts:                        2
vnp_tx_pkts:                        1
vnp_tx_drop_pkts:                   0
vnp_to_ips_pkts:                    0
vnp_to_ips_drop_pkts:               0

---------- IPS Stage -------------------------------------------------------------
ips_rx_pkts:                        0
ips_tx_pkts:                        0
ips_drop_pkts:                      0
ips_rej_pkts:                       0

---------- DPDK TX Stage ---------------------------------------------------------
dpdktx_rx_pkts:                     1
dpdktx_tx_pkts:                     1
dpdktx_drop_pkts:                   0
dpdktx_drop_oversized_pkt:          0


                           Engine  0        Engine  1        Engine  2        Engine  3
CPU ID:                             0                1                2                3

---------- DPDK RX Stage ---------------------------------------------------------
dpdkrx_rx_pkts:                     2                0                0                0
dpdkrx_tx_pkts:                     2                0                0                0
dpdkrx_drop_pkts:                   0                0                0                0

---------- VNP Stage -------------------------------------------------------------
vnp_rx_pkts:                        0                0                0                0
vnp_tx_pkts:                        0                0                0                0
vnp_tx_drop_pkts:                   0                0                0                0
vnp_to_ips_tx_pkts:                 0                0                0                0
vnp_to_ips_tx_drop_pkts:            0                0                0                0

---------- IPS Stage -------------------------------------------------------------
ips_rx_pkts:                        0                0                0                0
ips_tx_pkts:                        0                0                0                0
ips_drop_pkts:                      0                0                0                0
ips_rej_pkts:                       0                0                0                0

---------- DPDK TX Stage ---------------------------------------------------------
dpdktx_rx_pkts:                     0                0                0                1
dpdktx_tx_pkts:                     0                0                0                1
dpdktx_drop_pkts:                   0                0                0                0
dpdktx_drop_oversized_pkt:          0                0                0                0
```

Checking vNP statistics provides insights to how traffic is offloaded from the slow path (traversing the kernel) to the fast path (firewall and IPS operations quickly processed by the vNP engine). In particular, observe the number of session search engine (SSE) entries pushed from kernel or IPS to vNP engine, shown bolded (`ctr_sse_entries`). The number of packets going through the SSE fast path is also important and is bolded (`ctr_fw_and_ips_fpath`).

```
--------------------------------------------------------------------------------
FortiOS DPDK Helper VNP Stats
--------------------------------------------------------------------------------

                                  Total
CPU ID:

---------- VNP Internal ----------------------------------------------------------
ctr_ctx_alloc:                    2
ctr_ctx_alloc_fail:               0
ctr_ctx_free:                     2
ctr_to_kernel:                    2
ctr_from_kernel:                  1
ctr_sse:                          0
ctr_sse_cmd:                      0
ctr_sse_delmiss:                  0
ctr_sse_msg:                      0
ctr_sse_pruned:                   0
ctr_fw_and_ips_fpath:             0

ctr_sse_entries:                  0

err_sse_batch_size:               0
err_sse_unknown_cmd:              0
err_sse_full:                     0
err_sse_tbl_alloc_fail:           0
err_sse_inv_oid:                  0
err_fp_no_act:                    0
drop_inv_port:                    0
drop_inv_ip_cksum:                0
drop_inv_tcp_cksum:               0
drop_inv_udp_cksum:               0
drop_oversized_pkt:               0
--------------------------------------------------------------------------------


                              Engine  0        Engine  1        Engine  2        Engine  3
CPU ID:                       0                1                2                3

---------- VNP Internal ----------------------------------------------------------
ctr_ctx_alloc:                    0                0                0                0
ctr_ctx_alloc_fail:               0                0                0                0
ctr_ctx_free:                     0                0                0                0
ctr_to_kernel:                    0                0                0                0
ctr_from_kernel:                  0                0                0                0
ctr_sse:                          0                0                0                0
ctr_sse_cmd:                      0                0                0                0
ctr_sse_delmiss:                  0                0                0                0
ctr_sse_msg:                      0                0                0                0
ctr_sse_pruned:                   0                0                0                0
ctr_fw_and_ips_fpath:             0                0                0                0

ctr_sse_entries:                  0                0                0                0

err_sse_batch_size:               0                0                0                0
err_sse_unknown_cmd:              0                0                0                0
err_sse_full:                     0                0                0                0
err_sse_tbl_alloc_fail:           0                0                0                0
err_sse_inv_oid:                  0                0                0                0
err_fp_no_act:                    0                0                0                0
drop_inv_port:                    0                0                0                0
drop_inv_ip_cksum:                0                0                0                0
drop_inv_tcp_cksum:               0                0                0                0
drop_inv_udp_cksum:               0                0                0                0
drop_oversized_pkt:               0                0                0                0
--------------------------------------------------------------------------------
```

To see DPDK CPU settings, run the following commands. In this case, N is the number of CPUs that the FortiGate-VM uses.

```
show dpdk cpus
   config dpdk cpus
      set rx-cpus "0-N"
      set vnp-cpus "0-N"
      set ips-cpus "0-N"
      set tx-cpus "0-N"
   end
```

To view DPDK performance:

The `diagnose dpdk performance show` command provides near real-time performance of each DPDK engine, in particular, the CPU usage. The system provides the following response:

```
---------------------------------------
CPU usages
---------------------------------------
                                        Engine  0        Engine  1        Engine  2        Engine  3
2018:12:10 15:17:52                rx:              0.0              0.0              0.0              0.0
2018:12:10 15:17:52                vnp:             0.0              0.0              0.0              0.0
2018:12:10 15:17:52                ips:             0.0              0.0              0.0              0.0
2018:12:10 15:17:52                tx:              0.0              0.0              0.0              0.0
2018:12:10 15:17:52                idle:          100.0            100.0            100.0            100.0
---------------------------------------


                                        Engine  4        Engine  5        Engine  6        Engine  7
2018:12:10 15:17:52                rx:              0.0              0.0              0.0              0.0
2018:12:10 15:17:52                vnp:             0.0              0.0              0.0              0.0
2018:12:10 15:17:52                ips:             0.0              0.0              0.0              0.0
2018:12:10 15:17:52                tx:              0.0              0.0              0.0              0.0
2018:12:10 15:17:52                idle:          100.0            100.0            100.0            100.0
---------------------------------------


                                        Engine  8        Engine  9        Engine 10        Engine 11
2018:12:10 15:17:52                rx:              0.0              0.0              0.0              0.0
2018:12:10 15:17:52                vnp:             0.0              0.0              0.0              0.0
2018:12:10 15:17:52                ips:             0.0              0.0              0.0              0.0
2018:12:10 15:17:52                tx:              0.0              0.0              0.0              0.0
2018:12:10 15:17:52                idle:          100.0            100.0            100.0            100.0
---------------------------------------


                                        Engine 12        Engine 13        Engine 14        Engine 15
2018:12:10 15:17:52                rx:              0.0              0.0              0.0              0.0
2018:12:10 15:17:52                vnp:             0.0              0.0              0.0              0.0
2018:12:10 15:17:52                ips:             0.0              0.0              0.0              0.0
2018:12:10 15:17:52                tx:              0.0              0.0              0.0              0.0
2018:12:10 15:17:52                idle:          100.0            100.0            100.0            100.0
---------------------------------------
```

This provides better insight into how many CPUs to allocate to each pipeline stage.

Best practices

This document serves as a best practices guide covering use cases where the Fortinet FortiGate virtual appliance is used in mobile networks. This document aims to provide technically focused details and guidance around building the FortiGate-VM with recommendations on tuning the deployment to maximize performance.

The document assumes prior knowledge of the following topics:

- Computer hardware
- Hypervisors and virtualization
- SR-IOV
- DPDK

FortiGate-VM

- Supported on private and public clouds across many vendors
- Delivered in hypervisor-specific formats
- Licensed based on number of vCPUs (no restriction on RAM)
- Compatible with different network virtualization technologies, including:
  - virtIO / VMXNET3
  - PCI passthrough
  - SR-IOV
  - OVS-DPDK
- Supports common NICs, including the following for SR-IOV:
  - Intel cards compatible with igb, ixgbe and i40e drivers (1/10/25/40 Gbps)
  - Intel cards compatible with ice driver (100 Gbps) (FortiOS 6.4.1 and later versions)
  - Mellanox 100G cards (mlx_4 and mlx_5)
  - Broadcom 100G cards (P2100G) (FortiOS 6.4.3 and later versions)
- Internal implementation of DPDK to deliver the vSPU, giving massive performance benefits with no dependency on hypervisor

The FortiGate-VM product evolves quickly to support new hardware from third parties. Fortinet is continuously enhancing the product. Therefore, installing the latest GA release from one of the two latest FortiOS is recommended.

FortiGate vSPU

Virtual security processing units (vSPU), introduced in FortiOS 6.2.3, refer to the combination of the FortiOS virtual Network Processor (vNP) and DPDK libraries operating within the FortiGate-VM. vNP is the software emulation of a subset of the Fortinet Network Processor.

DPDK provides data plane libraries and the polling-mode driver (PMD), which enables offload of packet processing from the system kernel to user space. This allows the creation of high-speed networking applications, such as the vNP.

vSPU is implemented within the FortiGate-VM, allowing the virtual appliance to be optimized:

- vNP runs in user space, and the kernel is bypassed when vNP is handling the traffic.
- PMD means that traffic is taken from the NIC card without relying on CPU interrupts.

That means that for certain FortiGate-VM use cases, you can employ vSPU to make more effective use of CPU resource and achieve higher throughput.

You can activate vSPU by configuration on a per-CPU basis. Each CPU activated for vNP function is presented as a processing engine.

The following summarizes how FortiOS handles traffic when multiple CPUs are enabled for vSPU. You cannot change this behavior through configuration:

FortiOS version	Description
7.0.1 and earlier versions	Traffic balancing is based on the L3 header information. For best performance, a significant variation in source and destination IP addresses are needed to load all vSPUs evenly.
7.0.2 and later versions	Traffic balancing is based on the L3 and L4 header information. The hash used to balance across the DPDK engines is based on L4 source and destination port numbers in addition to L3 addresses. Therefore, loading more vSPUs evenly should be easier.

The vSPU is analogous to the physical NP found in physical appliances. Session creation is performed in the kernel, then offloaded to the vSPU, as the hardware offloads traffic to the NP.

For more information, including a diagram of the fastpath architecture, see Performance as a Key Attribute of Fortinet.

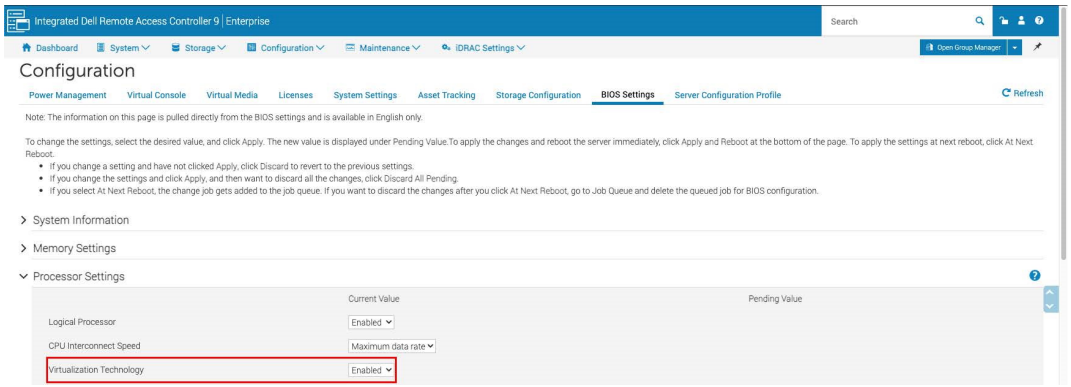
The vNP is beneficial if the IP payload is UDP or TCP. Other traffic traverses the device without benefiting from fastpath.

Server BIOS considerations

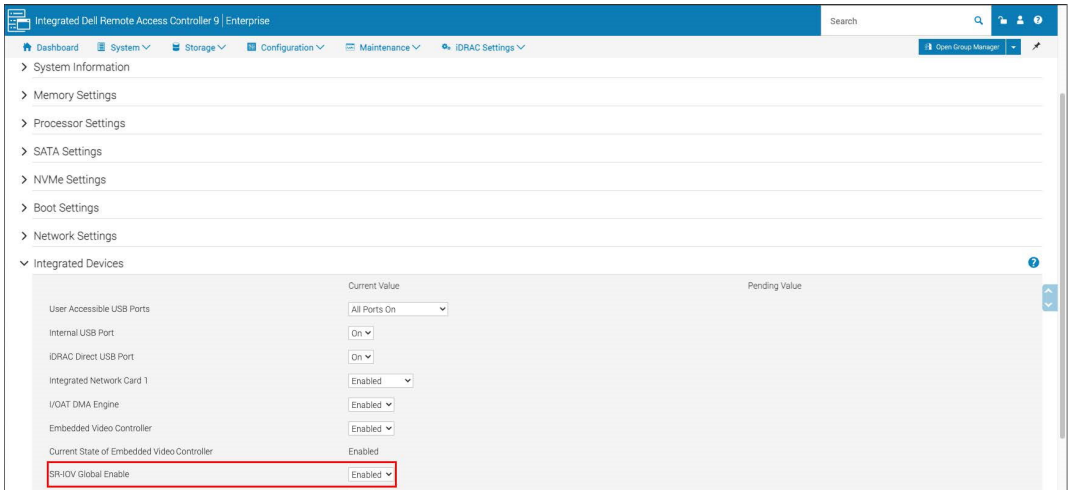
Typically, BIOS settings are necessary to enable SR-IOV and optimize resource usage.

As the exact configuration depends on the BIOS vendor and version used, researching these settings in the applicable vendor documentation is recommended. This document uses example settings based on a Dell PowerEdge R740.

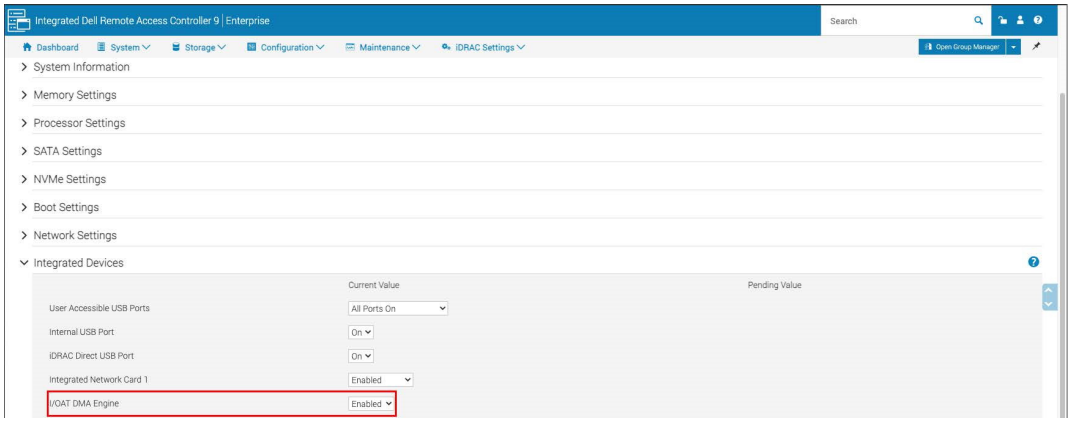
- Ensure that I/O memory management unit) is enabled. For the example hardware, the relevant setting is *Virtualization Technology*.



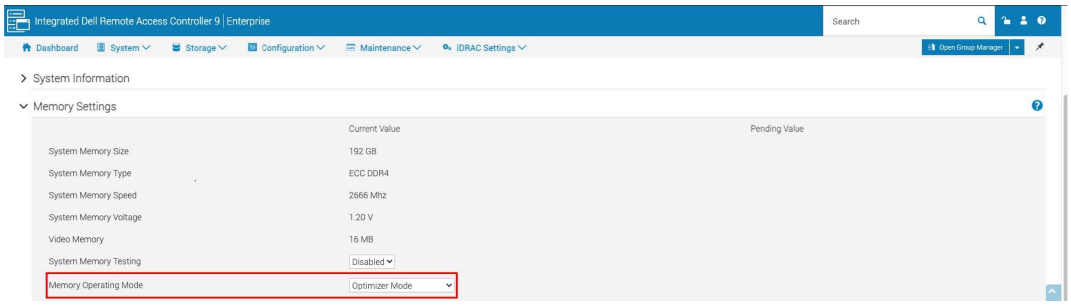
- Ensure that SR-IOV is enabled. For the example hardware, the relevant setting is *SR-IOV Global Enable*.



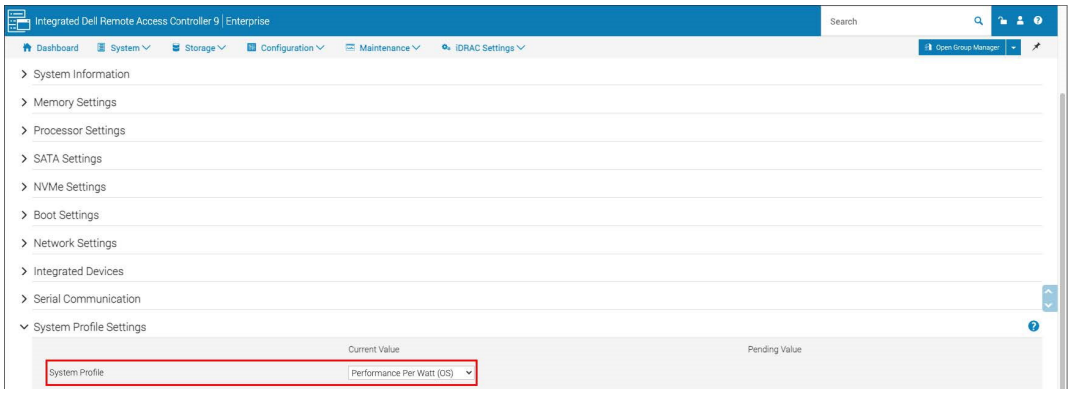
- Ensure that *I/OAT DMA Engine* is enabled. Intel and Mellanox hardware support this feature.



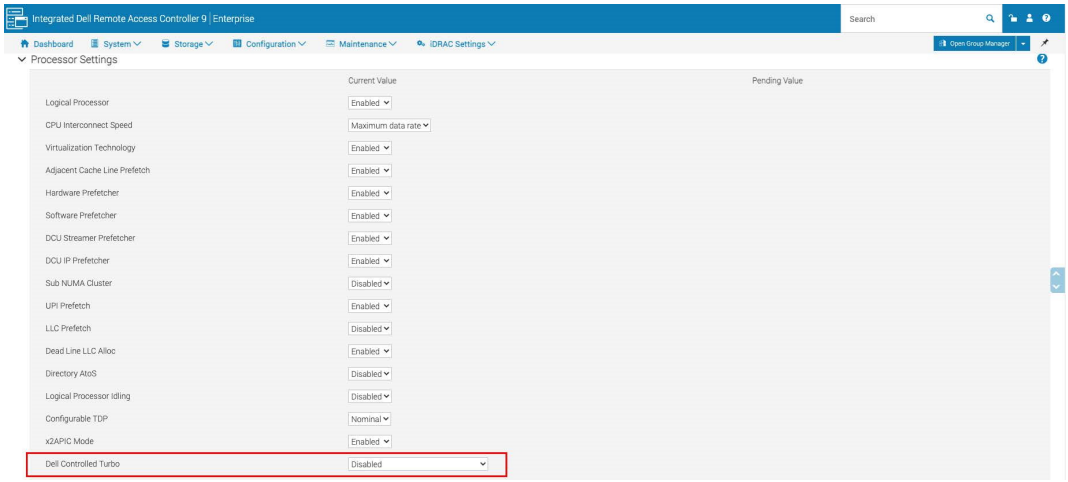
- High end servers have several memory operation modes . You must select the mode that gives the most memory to the operating system and maximum performance. For the example hardware, *Optimizer Mode* is selected from the *Memory Operating Mode* dropdown list.



- High end servers differ in BIOS recommendations about achieving the highest performance and lowest latency for options such as power saving and turbo boost. For the example hardware, *Performance Per Watt (OS)* is selected from the *System Profile* dropdown list. This means that these settings are managed within the host OS.



- You should disable the BIOS turbo setting if vSPU is in use. PMD takes the CPU load to 100%, which means that the processor would continually be overclocked, which is undesirable. If not using vSPU, leaving this option disabled is still recommended to avoid unpredictable CPU usage. For the example hardware, *Dell Controlled Turbo* is disabled.



The BIOS tasks ensure key features are enabled to ensure that the generic performance settings are set correctly to get the system to best complement the FortiGate-VM.

Hypervisor and OS tuning

The kernel component of KVM has been included in mainline Linux since version 2.6.20. The userspace component of KVM has been included mainline QEMU as of 1.3. The solution discussed uses both KVM components, and a tool called “virsh” to manage virtual machines. However, later versions are used.

You should consult the FortiOS Release Notes to determine the Fortinet recommendations on Linux versions. This document includes the steps taken on Red Hat Enterprise Linux 8 to create a performant FortiGate-VM deployment. This provides an outline of the steps needed on any KVM-based deployment, regardless of Linux choice.

As Fortinet does not provide support for the hypervisor nor the OS, a deviation from the release note recommendations should not cause concern. However, Fortinet expertise may not be as fluent in these circumstances.

The NIC is probably the most important consideration to achieve a performant firewall. Handling network I/O correctly and efficiently is of great importance. The main considerations, which will be covered in more detail later in the document, are:

- Traffic NICs should support SR-IOV. PCI-passthrough may be an alternative option but has little flexibility.
- Avoid OEM NICs. For example, a Dell-branded Intel XXV710 NIC may not have the required firmware version available to achieve a working solution.
- The number of NIC ports and thus the number of network queues/buffers used for traffic is important when considering a FortiGate-VM deployment without vSPU. Allow effective use of the CPUs.

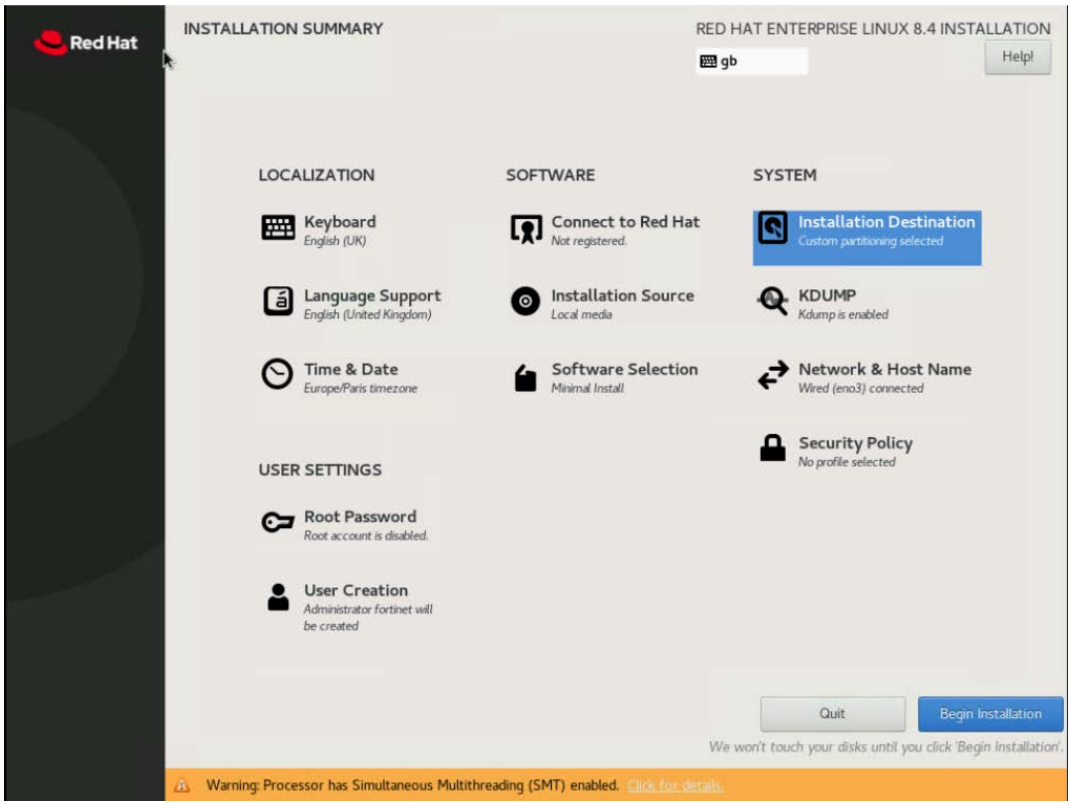
The kernel, QEMU, virsh, and NIC vendor and firmware/driver versions are typically outside of the deployment scope for Fortinet. However, they are important to achieve a stable and performant solution. Therefore, you should take due caution around the version choices to select these optimally. These items will be the first things to check if the performance is suboptimal or if, in fact, the deployment is unexpectedly not functioning as designed.

RHEL 8 versions

RHEL

```
[root@rhel-tiger-14-6 ~]# cat /etc/redhat-release
Red Hat Enterprise Linux release 8.4 (Ootpa)
[root@rhel-tiger-14-6 ~]# uname -r
4.18.0-305.25.1.el8_4.x86_64
```

- Consult the installation instructions that Red Hat provides.
- For a fresh installation, selecting *Minimal Install* for *Software Selection* is recommended, as you will add additional packages as needed.



- You should register RHEL and consider restricting the update process before performing an update:

```
[root@rhel-tiger-14-6 ~]# subscription-manager register
Registering to: subscription.rhsm.redhat.com:443/subscription
Username: xxxxxx
Password: xxxxxx
The system has been registered with ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
The registered system name is: rhel-tiger-14-6

[root@rhel-tiger-14-6 ~]# subscription-manager release --set=8.4
Release set to: 8.4

[root@rhel-tiger-14-6 ~]# yum -y update
<output omitted for brevity>

[root@rhel-tiger-14-6 ~]# reboot
```

This document continues to describe the installation of specific drivers for the NICs used in the system. This is effectively a kernel module, which must be compiled against the running kernel. While this is not a problem if you can compile the module every time that the kernel is updated, the Intel-provided driver did not compile against the newer kernel in RHEL 8.5.

This is likely due to RHEL 8.5 being new at time of writing. Intel may update their package accordingly. However, it illustrates the importance of exercising due care around updating the system, as changing the driver version could impact both functionality and performance.

Hypervisor and tools

```
[root@rhel-tiger-14-6 ~]# virsh version --daemon
Compiled against library: libvirt 6.0.0
Using library: libvirt 6.0.0
Using API: QEMU 6.0.0
Running hypervisor: QEMU 4.2.0
Running against daemon: 6.0.0
```

Check that the CPUs support virtualization:

```
[root@rhel-tiger-14-6 ~]# grep -io vmx /proc/cpuinfo | uniq
vmx
[root@rhel-tiger-14-6 ~]# lscpu | grep Virtual
Virtualization: VT-x
```

For Intel processors, `vmx` indicates that virtualisation is supported. For AMD processors, use `svm`.

Install and enable the virtualization modules and tools:

```
[root@rhel-tiger-14-6 ~]# yum -y module install virt
<output omitted for brevity>

[root@rhel-tiger-14-6 ~]# yum -y install virt-install virt-viewer cockpit cockpitmachines
<output omitted for brevity>

[root@rhel-tiger-14-6 ~]# systemctl enable --now cockpit.socket
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket → /usr/lib/
systemd/system/cockpit.socket.
```

`virt-manager` is considered deprecated in RHEL 8, so cockpit is used as a replacement. Cockpit does more than visualize VMs and is assessed by using a browser on TCP port 9090, e.g. https://10.210.14.6:9090/.

Install additional tools that can be useful when operating such a system:

```
[root@rhel-tiger-14-6 ~]# yum -y install sysstat tcpdump numactl hwloc
<output omitted for brevity>
```

NIC versions

NICs generally have three components to consider when considering SR-IOV:

- Firmware (or NVM)
- PF driver
- VF driver

> Care should be taken to ensure the drivers and firmware/non-volatile memory image are aligned as per manufacturer's recommendations. For example, the Intel X700 series recommendations are detailed in the Feature Support Matrix.

The VF driver is part of the FortiGate-VM instance. The driver versions are documented here. There is only one VF driver version per FortiOS version. This offers the least flexibility when aligning versions, making it the starting point.

As an example, the system used in this document has Intel XL710 NIC cards and is running FortiOS 7.0.2. Consulting the two resources above:

- VF driver: IAVF 4.1.1

Software Release Version	NVM Version	NVM Update Tool Version	i40e (Windows)	i40e (Linux)[1]	i40evf/ iavf[2] (Linux)[1,3]	i40en (ESX)	ixl (FreeBSD)	QSFP Config. Utility (QCU)	Ethernet Port Config. Tool (EPCT)
19.3	4.24 4.25	1.24.9.0	19.3	1.0.15	N/A	N/A	N/A	N/A	N/A
19.4	4.26	1.24.18.1	19.4	1.1.23	N/A	N/A	1.2.4	N/A	N/A
20.0	4.42	1.24.33.8	20.0	1.2.37 1.2.38	N/A	1.2.48	1.3.6	1.24.35.1	N/A
20.3	4.53	1.25.20.03	20.3	1.3.38 1.3.39.1	N/A	1.3.38	1.4.5	2.25.18.03	N/A
20.4.1	4.53	1.25.20.12	20.4.1	1.3.46 1.3.47	N/A	1.3.45	1.4.8	2.25.18.3	N/A
20.7	5.02	1.26.17.9	20.7	1.4.25 1.5.16	1.4.15 1.5.14	For ESX 6.0: 1.5.8 For ESX 6.5: 1.5.8 For ESX 6.7: 1.7.1	1.4.26 1.4.27	2.26.17.6 2.27.10.1	N/A
20.7.1	5.02 5.03[4]	1.26.17.11	20.7.1						
	5.04								
21.1 21.3 22.2	5.05	1.26.17.11 1.28.19.4	21.1 21.3 22.0 22.2	1.5.25[5] 1.6.42[5] 2.0.19 2.0.23	1.5.14 1.6.41 2.0.16 2.0.22	1.7.11	1.6.8 1.7.10 1.7.11	2.28.1.6 2.28.19.5 2.28.22.4	N/A
22.6 22.9 22.10 23.1 23.2	6.01	1.30.2.11 1.30.22.1 1.30.22.3	22.6 22.9 22.10 23.1 23.2	2.1.26 2.3.6 2.4.3 2.4.6 2.4.10	3.0.8 3.2.5 3.4.2 3.5.6 3.5.13	1.7.11	1.7.12 1.9.5 1.9.7 1.9.8	2.30.2.9 2.30.22.0 2.30.23.0 2.32.6.6	N/A
23.4	6.80	1.32.20.28	23.4	2.7.12	3.6.11	1.7.11	1.10.4	2.32.20.28	N/A
23.5.2	6.80	1.32.20.30	23.5.2	2.7.29	3.6.15	1.7.11	1.10.4	2.32.20.28	N/A
24.0	7.00	1.33.15.1	24.0	2.8.43	3.7.34	1.8.6	1.11.9	2.33.15.1	N/A
24.3	7.10	1.34.17.3	24.3	2.10.19.30	3.7.61.20	1.9.5	1.11.20	2.34.17.3	1.34.17.5
25.0	7.20	1.34.22.6	25.0	2.10.19.82	3.7.61.20	1.10.6	1.11.22	2.34.17.3	1.34.22.5
25.1	7.30	1.35.23.3	25.1	2.11.29	3.9.5	1.10.9.0	1.11.29	EOL	1.35.23.2
25.2	8.00	1.35.33.4	25.2	2.12.5	4.0.1	1.10.9.0	1.12.2	EOL	1.35.33.3
25.4	8.10	1.35.42.7	25.4	2.14.13	4.0.1	1.10.9.0	1.12.3	EOL	1.35.42.7
25.5	8.15	1.35.42.7	25.5	2.14.13	4.0.1	1.10.9.0	1.12.3	EOL	1.35.49.0
26.0	8.20	1.35.57.4	26.0	2.14.13	4.0.2	1.12.3.0	1.12.13	EOL	1.35.57.1
26.2	8.30	1.37.1.1	26.2	2.15.9	4.1.1	1.13.1.0	1.12.16	EOL	1.37.1.0
26.4	8.40	1.37.13.5	26.4	2.16.11	4.2.7	1.13.1.0	1.12.24	EOL	1.37.13.3
26.6	8.50	1.37.28.0	26.6	2.17.4	4.2.7	2.1.5.0	1.12.29	EOL	1.37.28.0

- Software Release Version: 26.2
- PF driver: i40e 2.15.9
- NVM version: 8.30

> Any deviation from this alignment would need to be diligently tested and may still cause later supportability issues.

Instructions for Mellanox public repositories can be found at Mellanox Technologies Ltd. Public Repository. You can find firmware update instructions at Firmware Update Instructions.

Having said that, the example system in this document breaks this rule. As it is not carrying production traffic, then it allows future versions to be considered without any negative impact. The example prints are based upon Intel's Software Release Version 26.6 which importantly contains some fixes for issues observed on other hypervisors.

```
[root@rhel-tiger-14-6 ~]# ethtool -i ens5f0
driver: i40e
version: 2.17.4
firmware-version: 8.50 0x8000b6c7 1.3082.0expansion-rom-version:
bus-info: 0000:86:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yes
```

> RHEL 8 packages the i40e kernel module and reports the kernel version as the driver version. This may not be helpful, as if the above command is run before installing the desired i40e driver, the version reported is the kernel version.

Search, download, and transfer to the versions desired to the server:

- Google: intel xl710 driver 2.17.4
- Google: intel xl710 nvm 8.50

Compile and install the driver, following the vendor's instructions:

```
[root@rhel-tiger-14-6 src]# yum -y install make kernel-devel gcc elfutils-libelf-devel
<output omitted for brevity>

[root@rhel-tiger-14-6 fortinet]# tar xzf i40e-2.17.4.tar.gz
<output omitted for brevity>

[root@rhel-tiger-14-6 fortinet]# cd i40e-2.17.4/src/

[root@rhel-tiger-14-6 src]# make install
<output omitted for brevity>

[root@rhel-tiger-14-6 src]# reboot
```

Upgrade the firmware, following the vendor's instructions:

```
[root@rhel-tiger-14-6 fortinet]# yum -y install unzip
<output omitted for brevity>

[root@rhel-tiger-14-6 fortinet]# unzip 700Series_NVMUpdatePackage_v8_50.zip
```

```
Archive:  700Series_NVMUpdatePackage_v8_50.zip
  inflating: 700Series_NVMUpdatePackage_v8_50_EFI.zip
  inflating: 700Series_NVMUpdatePackage_v8_50_ESX.tar.gz
  inflating: 700Series_NVMUpdatePackage_v8_50_FreeBSD.tar.gz
  inflating: 700Series_NVMUpdatePackage_v8_50_Linux.tar.gz
  inflating: 700Series_NVMUpdatePackage_v8_50_Windows.zip

[root@rhel-tiger-14-6 fortinet]# tar xzf 700Series_NVMUpdatePackage_v8_50_Linux.tar.gz
<output omitted for brevity>

[root@rhel-tiger-14-6 fortinet]# cd 700Series/Linux_x64/

[root@rhel-tiger-14-6 Linux_x64]# ./nvmupdate64e
<output omitted for brevity>

[root@rhel-tiger-14-6 src]# reboot
```

You can check the results via the previous `ethtool` command and/or `modinfo`:

```
[root@rhel-tiger-14-6 ~]# modinfo i40e
filename:       /lib/modules/4.18.0-305.25.1.el8_4.x86_
64/updates/drivers/net/ethernet/intel/i40e/i40e.ko
version:        2.17.4
license:        GPL
description:    Intel(R) 40-10 Gigabit Ethernet Connection Network Driver
author:         Intel Corporation, <e1000-devel@lists.sourceforge.net>
rhelversion:    8.4
srcversion:     5941E76EE679E4C3E0A7AEF
alias:          pci:v00008086d0000158Bsv sdbcsci*
alias:          pci:v00008086d0000158Asv sdbcsci*
alias:          pci:v00008086d000037D3sv sdbcsci*
alias:          pci:v00008086d000037D2sv sdbcsci*
alias:          pci:v00008086d000037D1sv sdbcsci*
alias:          pci:v00008086d000037D0sv sdbcsci*
alias:          pci:v00008086d000037CFsv sdbcsci*
alias:          pci:v00008086d000037CEsv sdbcsci*
alias:          pci:v00008086d00000D58sv sdbcsci*
alias:          pci:v00008086d00000CF8sv sdbcsci*
alias:          pci:v00008086d00001588sv sdbcsci*
alias:          pci:v00008086d00001587sv sdbcsci*
alias:          pci:v00008086d0000104Fsv sdbcsci*
alias:          pci:v00008086d0000104Esv sdbcsci*
alias:          pci:v00008086d000015FFsv sdbcsci*
alias:          pci:v00008086d00001589sv sdbcsci*
alias:          pci:v00008086d00001586sv sdbcsci*
alias:          pci:v00008086d0000101Fsv sdbcsci*
alias:          pci:v00008086d00001585sv sdbcsci*
alias:          pci:v00008086d00001584sv sdbcsci*
alias:          pci:v00008086d00001583sv sdbcsci*
alias:          pci:v00008086d00001581sv sdbcsci*
alias:          pci:v00008086d00001580sv sdbcsci*
alias:          pci:v00008086d00001574sv sdbcsci*
alias:          pci:v00008086d00001572sv sdbcsci*
```

```
depends:
name:           i40e
vermagic:       4.18.0-305.25.1.el8_4.x86_64 SMP mod_unload modversions
parm:           debug:Debug level (0=none,...,16=all) (int)
parm:           l4mode:L4 cloud filter mode: 0=UDP,1=TCP,2=Both,-1=Disabled(default) (int)
```

The kernel uses `vermagic` to ensure that the module is compiled against the running kernel. This reiterates the importance to fully control and test OS updates prior to releasing in production.

GRUB

`virt-host-validate` is a tool available to check if the hypervisor has been prepared:

```
[root@rhel-tiger-14-6 ~]# virt-host-validate
  QEMU: Checking for hardware virtualization                                 : PASS
  QEMU: Checking if device /dev/kvm exists                                   : PASS
  QEMU: Checking if device /dev/kvm is accessible                            : PASS
  QEMU: Checking if device /dev/vhost-net exists                             : PASS
  QEMU: Checking if device /dev/net/tun exists                               : PASS
  QEMU: Checking for cgroup 'cpu' controller support                         : PASS
  QEMU: Checking for cgroup 'cpuacct' controller support                     : PASS
  QEMU: Checking for cgroup 'cpuset' controller support                      : PASS
  QEMU: Checking for cgroup 'memory' controller support                      : PASS
  QEMU: Checking for cgroup 'devices' controller support                     : PASS
  QEMU: Checking for cgroup 'blkio' controller support                       : PASS
  QEMU: Checking for device assignment IOMMU support                         : PASS
  QEMU: Checking if IOMMU is enabled by kernel                               : WARN (IOMMU appears
to be disabled in kernel. Add intel_iommu=on to kernel cmdline arguments)
  QEMU: Checking for secure guest support                                    : WARN (Unknown if
this platform has Secure Guest support)
```

`QEMU: Checking for secure guest support` is relevant to AMD or IBM processors only and is of no concern in this case.

You need IOMMU to access physical devices directly, but it is missing from the kernel command line:

```
[root@rhel-tiger-14-6 ~]# cat /proc/cmdline BOOT_IMAGE=(hd1,gpt2)/vmlinuz-4.18.0-305.25.1.el8_
4.x86_64 root=/dev/mapper/vg1-root ro crashkernel=auto resume=/dev/mapper/vg1-swap
rd.lvm.lv=vg1/root rd.lvm.lv=vg1/swap rhgb quiet
```

For performant systems, you must consider CPU and memory configuration. You can use `tuned` to take care of CPU and handle memory directly.

Direct update

```
[root@rhel-tiger-14-6 ~]# grubby --update-kernel=ALL --args="intel-iommu=on iommu=pt hugepagesz=1G
default_hugepagesz=1G hugepages=160 transparent_hugepage=never selinux=0"
[root@rhel-tiger-14-6 ~]# reboot
```

Command	Description
`intel-iommu=on iommu=pt`	Enable Intel IOMMU driver and put into passthrough (adapter does not need DMA translation) mode. This is needed for SR-IOV.
`hugepagesz=1G default_ hugepagesz=1G hugepages=160 transparent_ hugepage=never`	Optimizes how memory is used, especially in systems with a lot of it. The number of hugepages is shared equally across the NUMA nodes, and the number depends on the amount of memory installed.

In this case, there are 160 hugepages of size 1 GiB each to be used with the FortiGate-VMs. The important point is to avoid starving the hypervisor/OS and leave enough non-hugepage memory per NUMA.

```
[root@rhel-tiger-14-6 ~]# virsh nodeinfo
CPU model:           x86_64
CPU(s):              72
CPU frequency:       3258 MHz
CPU socket(s):       1
Core(s) per socket:  18
Thread(s) per core:  2
NUMA cell(s):        2
Memory size:         196373424 KiB

[root@rhel-tiger-14-6 ~]# expr 196373424 / 1024 / 1024
187
```

187 GiB total leaves 27 GiB outside of hugepage definition for the hypervisor/OS. Leaving approximately 16 GiB per NUMA is recommended. However, it is not an exact science. In this case, 12 x 16 GiB DIMMs (192 GiB) are known to be physically installed and 160 GiB has been configured in hugepages.

CPU sockets in `virsh nodeinfo` are counted per NUMA cell as per Bugzilla.

Command	Description
`selinux=0`	Disable SELinux.

If you must have SELinux enabled as part of your build standard, you can enable it. However, this may impact overall performance.

On Ubuntu systems, SELinux may be replaced with AppArmor. You can disable it using `apparmor=0`.

```
[root@rhel-tiger-14-6 ~]# cat /proc/cmdline
BOOT_IMAGE=(hd1,gpt2)/vmlinuz-4.18.0-305.25.1.el8_4.x86_64 root=/dev/mapper/vg1-root ro
crashkernel=auto resume=/dev/mapper/vg1-swap rd.lvm.lv=vg1/root rd.lvm.lv=vg1/swap rhgb quiet
intel-iommu=on iommu=pt hugepagesz=1G default_hugepagesz=1G hugepages=160 transparent_
hugepage=never selinux=0
```

```
[root@rhel-tiger-14-6 ~]# virt-host-validate
  QEMU: Checking for hardware virtualization                                 : PASS
  QEMU: Checking if device /dev/kvm exists                                   : PASS
  QEMU: Checking if device /dev/kvm is accessible                            : PASS
  QEMU: Checking if device /dev/vhost-net exists                             : PASS
  QEMU: Checking if device /dev/net/tun exists                               : PASS
  QEMU: Checking for cgroup 'cpu' controller support                         : PASS
  QEMU: Checking for cgroup 'cpuacct' controller support                     : PASS
  QEMU: Checking for cgroup 'cpuset' controller support                      : PASS
  QEMU: Checking for cgroup 'memory' controller support                      : PASS
  QEMU: Checking for cgroup 'devices' controller support                     : PASS
  QEMU: Checking for cgroup 'blkio' controller support                       : PASS
  QEMU: Checking for device assignment IOMMU support                         : PASS
  QEMU: Checking if IOMMU is enabled by kernel                               : PASS
  QEMU: Checking for secure guest support                                    : WARN (Unknown if
this platform has Secure Guest support)
```

Tuned indirect update

You can use tuned for CPU partitioning: isolating CPUs from use by the Linux OS, except a CPU usage measurement made by the kernel every second. You can find more information at TUNED_PROFILES_CPU_PARTITIONING(7). Essentially, you must optimize this on the hardware, depending on number of CPUs and per NUMA node.

The following provides an example of the considerations, using the Dell PowerEdge R740 as an example:

```
[root@rhel-tiger-14-6 ~]# lscpu egrep "^(NUMA |Socket|Core|Thread)"
Thread(s) per core:  2
Core(s) per socket:  18
Socket(s):           2
NUMA node(s):        2
NUMA node0 CPU(s):
0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,6
8,70
NUMA node1 CPU(s):
1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,55,57,59,61,63,65,67,6
9,71

[root@rhel-tiger-14-6 ~]# tuned-adm active
Current active profile: throughput-performance

[root@rhel-tiger-14-6 ~]# cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor wc -l
72

[root@rhel-tiger-14-6 ~]# cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor sort -u
performance

[root@rhel-tiger-14-6 ~]# cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_max_freq sort -u
3700000

[root@rhel-tiger-14-6 ~]# cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_min_freq sort -u
```

```
1200000

[root@rhel-tiger-14-6 ~]# cat /sys/devices/system/cpu/cpu*/cpufreq/cpuinfo_cur_freq sort -u
cat: '/sys/devices/system/cpu/cpu*/cpufreq/cpuinfo_cur_freq': No such file or directory
```

In this case, there are two NUMA nodes with one socket each, equating to 36 vCPUs per NUMA. Considering the FortiGate-VM sizing options, using 32 vCPUs per NUMA for VMs makes sense. This leaves four vCPUs for the host machine to use for housekeeping.

The current tuned profile is for throughout performance, which has some relevance to the CPU scaling governor settings. You can reperform these commands once you have configured `tuned` as desired and the outputs here are for comparison. The current frequency is not set in this case:

```
[root@rhel-tiger-14-6 ~]# yum -y install tuned-profiles-cpu-partitioning
<output omitted for brevity>

[root@rhel-tiger-14-6 ~]# touch /etc/tuned/cpu-partitioning-variables.conf

[root@rhel-tiger-14-6 ~]# chown root:root /etc/tuned/cpu-partitioning-variables.conf

[root@rhel-tiger-14-6 ~]# chmod 644 /etc/tuned/cpu-partitioning-variables.conf

[root@rhel-tiger-14-6 ~]# cat /etc/tuned/cpu-partitioning-variables.conf
# Examples:
# isolated_cores=2,4-7
# isolated_cores=2-23
#
# To disable the kernel load balancing in certain isolated CPUs:
# no_balance_cores=5-10
isolated_cores=4-35,40-71
no_balance_cores=4-35,40-71

[root@rhel-tiger-14-6 ~]# tuned-adm profile cpu-partitioning

[root@rhel-tiger-14-6 ~]# tuned-adm active
Current active profile: cpu-partitioning

[root@rhel-tiger-14-6 ~]# reboot
```

In this example, vCPUs 0-3 and 36-39 are not declared and are the housekeeping resources, while the others are used in VMs.

When the `tuned` profile is activated, changes are embedded itself into the kernel command line via GRUB:

```
[root@rhel-tiger-14-6 ~]# cat /proc/cmdline
BOOT_IMAGE=(hd1,gpt2)/vmlinuz-4.18.0-305.25.1.el8_4.x86_64 root=/dev/mapper/vg1-root ro
crashkernel=auto resume=/dev/mapper/vg1-swap rd.lvm.lv=vg1/root rd.lvm.lv=vg1/swap rhgb quiet
intel-iommu=on iommu=pt hugepagesz=1G default_hugepagesz=1G hugepages=160 transparent_
hugepage=never selinux=0 skew_tick=1 nohz=on nohz_full=4-35,40-71 rcu_nocbs=4-35,40-71 tuned.non_
isolcpus=000000f0,0000000f intel_pstate=disable nosoftlockup
```

`tuned` has added the following parameters:

Parameter	Description
`skew_tick=1`	Ensures that the ticks per CPU do not occur simultaneously by skewing their start times. Skewing the start times of the per-CPU timer ticks decreases the potential for lock conflicts, reducing system jitter for interrupt response times.
`nohz=on`	Turns off the timer tick on an idle CPU.
`nohz_full=4-35,40-71`	Turns off the timer tick on a CPU when there is only one runnable task on that CPU. Needs `nohz` to be set to on.
`rcu_nocbs=4-35,40-71`	To allow the user to move all RCU offload threads to a housekeeping CPU.
`tuned.non_ isolcpus=000000f0,0000000f`	The CPU mask of the CPUs left for the host to use, in our example 000000f0,0000000f => 0x000000F00000000F => CPUs 0-3, 36-39 (c.f. CPU Affinity Calculator).
`intel_pstate=disable`	Prevents the Intel idle driver from managing power state and CPU frequency.
`nosoftlockup`	Prevents the kernel from detecting soft lockups in user threads.

The `isolcpus` parameter is considered deprecated. Instead, `tuned` is using CPUsets/affinity to partition CPUs. It does what it can to keep the kernel noise/housekeeping away from CPUs that are intended to use for VMs.

Using the lowest number physical cores for housekeeping is good practice. You can use `lstopo-no-graphics` to ensure that the appropriate ones are selected. The following shows a snippet of the `lstopo-no-graphics` output:

```
[root@rhel-tiger-14-6 ~]# lstopo-no-graphics
Machine (187GB total)
  Package L#0
    NUMANode L#0 (P#0 93GB)
    L3 L#0 (25MB)
      L2 L#0 (1024KB) + L1d L#0 (32KB) + L1i L#0 (32KB) + Core L#0
        PU L#0 (P#0)
        PU L#1 (P#36)
      L2 L#1 (1024KB) + L1d L#1 (32KB) + L1i L#1 (32KB) + Core L#1
        PU L#2 (P#2)
        PU L#3 (P#38)
      L2 L#2 (1024KB) + L1d L#2 (32KB) + L1i L#2 (32KB) + Core L#2
        PU L#4 (P#4)
        PU L#5 (P#40)
<output omitted for brevity>
  Package L#1
    NUMANode L#1 (P#1 94GB)
    L3 L#1 (25MB)
      L2 L#18 (1024KB) + L1d L#18 (32KB) + L1i L#18 (32KB) + Core L#18
        PU L#36 (P#1)
        PU L#37 (P#37)
      L2 L#19 (1024KB) + L1d L#19 (32KB) + L1i L#19 (32KB) + Core L#19
```

```
        PU L#38 (P#3)
        PU L#39 (P#39)
      L2 L#20 (1024KB) + L1d L#20 (32KB) + L1i L#20 (32KB) + Core L#20
        PU L#40 (P#5)
        PU L#41 (P#41)
<output omitted for brevity>
```

- *Core L#0* = Physical Core with hwloc index 0
- *PU L#0 (P#0)* = Processing Unit with hwloc index 0: processor 0
- *PU L#1 (P#36)* = Processing Unit with hwloc index 1: processor 36

You can see how the vCPU number relates to the physical core. In this case, physical core 0 has two threads identified as vCPU 0 and vCPU 36.

`tuned` has also taken care of the CPU scaling governor. Clock scaling allows changing the CPU clock speed on the fly between a minimum and maximum value. This is good for some compute hardware but for a performant system, the CPU must work at the maximum frequency it can. It must be in performance mode:

```
[root@rhel-tiger-14-6 ~]# cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor wc -l
72

[root@rhel-tiger-14-6 ~]# cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor sort -u
performance

[root@rhel-tiger-14-6 ~]# cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_max_freq sort -u
3001000

[root@rhel-tiger-14-6 ~]# cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_min_freq sort -u
1200000

[root@rhel-tiger-14-6 ~]# cat /sys/devices/system/cpu/cpu*/cpufreq/cpuinfo_cur_freq sort -u
3001000
```

Change is due to the addition of `intel_pstate=disable` to the kernel command line. You no longer overclock the CPU. Overclocking is not advised as it makes the performance less predictable. When using vSPU, overclocking leads to the CPUs being permanently overclocked, which could lead to problems such as overheating.

In the case of the aforementioned outputs, you can see that all 72 CPUs have been set to performance mode. All CPUs are operating at the maximum frequency that they can support without overclocking. You configure this setting per CPU and should configure all CPUs used for the FortiGate-VM this way as a minimum requirement.

sysctl

You can use `sysctl` to modify kernel parameters at runtime. For a performant system, the following tuning can be persisted across system restarts using a file in the appropriate directory. The following shows an example:

```
[root@rhel-tiger-14-6 ~]# touch /etc/sysctl.d/10-tiger.conf

[root@rhel-tiger-14-6 ~]# chown root:root /etc/sysctl.d/10-tiger.conf
```

```
[root@rhel-tiger-14-6 ~]# chmod 644 /etc/sysctl.d/10-tiger.conf

[root@rhel-tiger-14-6 ~]# cat /etc/sysctl.d/10-tiger.conf
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.core.netdev_budget=900
vm.swappiness = 0
net.ipv4.tcp_timestamps=0
net.core.netdev_max_backlog=250000
net.core.rmem_max=4194304
net.core.wmem_max=4194304
net.core.rmem_default=4194304
net.core.wmem_default=4194304
net.core.optmem_max=4194304
net.ipv4.tcp_rmem=4096 87380 4194304
net.ipv4.tcp_wmem=4096 65536 4194304
net.ipv4.tcp_low_latency=1

[root@rhel-tiger-14-6 ~]# reboot
```

- `net.core.netdev_budget=900` represents the maximum number of packets that a single CPU poll receives. The default is 300, but this is tuned up for high load systems.
- `vm.swappiness = 0` shows that swap (virtual memory) is disabled. This helps achieve low latency operations.

SELinux

SELinux is already disabled in the kernel command line, thanks to GRUB. Therefore, the `/etc/sysconfig/selinux` settings are ignored. The following changes reinforce this and are considered good practice.

If you must have SELinux enabled as part of your build standard, you can enable it. However, this may impact overall performance.

```
[root@rhel-tiger-14-6 ~]# sed -i s/^SELINUX=.*$/SELINUX=disabled/ /etc/selinux/config

[root@rhel-tiger-14-6 ~]# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

- `SELINUX=disabled`: disable SELinux.

You can check SELinux status using one of the following commands:

```
[root@rhel-tiger-14-6 ~]# sestatus
SELinux status:                 disabled
```

```
[root@rhel-tiger-14-6 ~]# getenforce
Disabled
```

Firewalld

You should disable the system firewall, as it is not required:

```
[root@rhel-tiger-14-6 ~]# systemctl is-enabled firewalld
enabled

[root@rhel-tiger-14-6 ~]# systemctl is-active firewalld
active

[root@rhel-tiger-14-6 ~]# systemctl disable --now firewalld
Removed /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.

[root@rhel-tiger-14-6 ~]# systemctl is-enabled firewalld
disabled

[root@rhel-tiger-14-6 ~]# systemctl is-active firewalld
inactive
```

NetworkManager

NetworkManager is the default tool in RHEL 8 for managing network service. For a best practice setup, you will configure NetworkManager to ignore the interfaces being configured for use with the FortiGate-VM:

```
[root@rhel-tiger-14-6 ~]# touch /etc/NetworkManager/conf.d/10-tiger.conf

[root@rhel-tiger-14-6 ~]# chown root:root /etc/NetworkManager/conf.d/10-tiger.conf

[root@rhel-tiger-14-6 ~]# chmod 644 /etc/NetworkManager/conf.d/10-tiger.conf

[root@rhel-tiger-14-6 ~]# cat /etc/NetworkManager/conf.d/10-tiger.conf
[keyfile]
unmanaged-devices=interface-name:ens4f0;interface-name:ens4f1;interface-name:ens4f2;interface-
name:ens4f3;interface-name:ens5f0;interface-name:ens5f1;interface-name:ens5f2;interface-
name:ens5f3

[root@rhel-tiger-14-6 ~]# systemctl reload NetworkManager

[root@rhel-tiger-14-6 ~]# nmcli device status
```

```
DEVICE      TYPE      STATE                   CONNECTION
eno3        ethernet  connected               eno3
virbr0      bridge    connected (externally)  virbr0
eno1        ethernet  unavailable             --
eno2        ethernet  unavailable             --
eno4        ethernet  unavailable             --
ens1f0      ethernet  unavailable             --
ens1f1      ethernet  unavailable             --
ens2f0      ethernet  unavailable             --
ens2f1      ethernet  unavailable             --
ens4f0      ethernet  unmanaged               --
ens4f1      ethernet  unmanaged               --
ens4f2      ethernet  unmanaged               --
ens4f3      ethernet  unmanaged               --
ens5f0      ethernet  unmanaged               --
ens5f1      ethernet  unmanaged               --
ens5f2      ethernet  unmanaged               --
ens5f3      ethernet  unmanaged               --
lo          loopback  unmanaged               --
virbr0-nic  tun       unmanaged               --
```

NIC queues (ring buffer size)

Maximize the receive queue/buffer on the NIC to optimize throughput. This is not expressly needed, but maximizing the transmit queue is also performed:

```
[root@rhel-tiger-14-6 ~]# ethtool -g ens4f0
Ring parameters for ens4f0:
Pre-set maximums:
RX:             4096
RX Mini:        n/a
RX Jumbo:       n/a
TX:             4096
Current hardware settings:
RX:             4096
RX Mini:        n/a
RX Jumbo:       n/a
TX:             4096
```

You must set this setting for each system restart, so persisting the changes by creating a system service is advised:

```
[root@rhel-tiger-14-6 ~]# touch /usr/local/bin/tiger-nic-rb-handler

[root@rhel-tiger-14-6 ~]# chown root:root /usr/local/bin/tiger-nic-rb-handler

[root@rhel-tiger-14-6 ~]# chmod 755 /usr/local/bin/tiger-nic-rb-handler

[root@rhel-tiger-14-6 ~]# cat /usr/local/bin/tiger-nic-rb-handler
#!/bin/bash
```

```
lookup() {
  if [[ -z $1 ]] ; then
    echo ""
  else
    awk -v "id=$1" 'BEGIN { FS = "=" } $1 == id { print $2 ; exit }' $2
  fi
}

start() {
  for _ethdev in ${ethlist//;/ } ; do
    ethdev=${_ethdev#*:}
    if [[ $(ethtool -g $ethdev 2>/dev/null wc -l) -gt 6 ]] ; then
      ethtool -G $ethdev $(ethtool -g $ethdev head -6 egrep [RT]X: xargs tr [:upper:] [:lower:]
sed 's/://g')
    fi
  done
}

ethlist=$(lookup unmanaged-devices /etc/NetworkManager/conf.d/10-tiger.conf)

case $1 in
  start) "$1" ;;
esac

[root@rhel-tiger-14-6 ~]# touch /etc/systemd/system/tiger-nic-rb-handler.service

[root@rhel-tiger-14-6 ~]# chown root:root /etc/systemd/system/tiger-nic-rb-handler.service

[root@rhel-tiger-14-6 ~]# chmod 644 /etc/systemd/system/tiger-nic-rb-handler.service

[root@rhel-tiger-14-6 ~]# cat /etc/systemd/system/tiger-nic-rb-handler.service
[Unit]
Description=Set NIC ring buffer

[Service]
Type=oneshot
ExecStart=/usr/local/bin/tiger-nic-rb-handler start
ExecStop=/usr/local/bin/tiger-nic-rb-handler stop
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target

[root@rhel-tiger-14-6 ~]# systemctl enable --now tiger-nic-rb-handler
Created symlink /etc/systemd/system/multi-user.target.wants/tiger-nic-rb-handler.service →
/etc/systemd/system/tiger-nic-rb-handler.service.
```

The script simply sources the list of unmanaged interfaces from the NetworkManager configuration created earlier and uses `ethtool` to find and set the ring buffers to their maximum values. The filename in the script must match the filename created earlier.

Network virtual functions

You must create network virtual functions (VFs) to use with the FortiGate-VM.

To determine which NICs are capable of running VFs:

```
[root@rhel-tiger-14-6 ~]# lspci -d ::0x200 -vv | egrep "Ethernet controller|SR-IOV"
01:00.0 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
01:00.1 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
19:00.0 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ (rev 01)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
19:00.1 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ (rev 01)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
3b:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 02)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
3b:00.1 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 02)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
5e:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 02)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
5e:00.1 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 02)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
86:00.0 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 02)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
86:00.1 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 02)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
86:00.2 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 02)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
86:00.3 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 02)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
af:00.0 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 02)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
af:00.1 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 02)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
af:00.2 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 02)
        Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
af:00.3 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 02)
 Capabilities: [160] Single Root I/O Virtualization (SR-IOV)
```

To determine the maximum amount of VFs that a PF can run:

```
[root@rhel-tiger-14-6 ~]# cat /sys/class/net/ens4f0/device/sriov_totalvfs
32
```

A useful way to see available NICs, the device name, and PCIe bus address is as follows:

```
[root@rhel-tiger-14-6 ~]# lshw -c network -businfo
Bus info          Device      Class          Description
========================================================
pci@0000:01:00.0  eno3        network        I350 Gigabit Network Connection
pci@0000:01:00.1  eno4        network        I350 Gigabit Network Connection
pci@0000:19:00.0  eno1        network        Ethernet Controller X710 for 10GbE SFP+
pci@0000:19:00.1  eno2        network        Ethernet Controller X710 for 10GbE SFP+
```

```
pci@0000:3b:00.0  ens1f0      network        Ethernet Controller E810-C for QSFP
pci@0000:3b:00.1  ens1f1      network        Ethernet Controller E810-C for QSFP
pci@0000:5e:00.0  ens2f0      network        Ethernet Controller E810-C for QSFP
pci@0000:5e:00.1  ens2f1      network        Ethernet Controller E810-C for QSFP
pci@0000:86:00.0  ens5f0      network        Ethernet Controller XL710 for 40GbE QSFP+
pci@0000:86:00.1  ens5f1      network        Ethernet Controller XL710 for 40GbE QSFP+
pci@0000:86:00.2  ens5f2      network        Ethernet Controller XL710 for 40GbE QSFP+
pci@0000:86:00.3  ens5f3      network        Ethernet Controller XL710 for 40GbE QSFP+
pci@0000:af:00.0  ens4f0      network        Ethernet Controller XL710 for 40GbE QSFP+
pci@0000:af:00.1  ens4f1      network        Ethernet Controller XL710 for 40GbE QSFP+
pci@0000:af:00.2  ens4f2      network        Ethernet Controller XL710 for 40GbE QSFP+
pci@0000:af:00.3  ens4f3      network        Ethernet Controller XL710 for 40GbE QSFP+
```

There are standard ways to persist this across system restart. The following is an example script that is installed as a system service:

```
[root@rhel-tiger-14-6 ~]# touch /usr/local/bin/tiger-nic-vf-handler

[root@rhel-tiger-14-6 ~]# chown root:root /usr/local/bin/tiger-nic-vf-handler

[root@rhel-tiger-14-6 ~]# chmod 755 /usr/local/bin/tiger-nic-vf-handler

[root@rhel-tiger-14-6 ~]# cat /usr/local/bin/tiger-nic-vf-handler
#!/bin/bash

numvfs=2

lookup() {
  if [[ -z $1 ]] ; then
    echo ""
  else
    awk -v "id=$1" 'BEGIN { FS = "=" } $1 == id { print $2 ; exit }' $2
  fi
}

start() {
  for _ethdev in ${ethlist//;/ } ; do
    ethdev=${_ethdev#*:}
    echo $numvfs > /sys/class/net/${ethdev}/device/sriov_numvfs
    sleep 2
    for ((vfnum=0;vfnum<${numvfs};vfnum++)); do
      ip link set $ethdev vf $vfnum spoofchk off trust on
      for iface in $(ls /sys/class/net) ; do
        if [[ $(readlink /sys/class/net/${iface}) =~ $(readlink
/sys/class/net/${ethdev}/device/virtfn${vfnum} xargs -n 1 basename) ]] ; then
          nmcli device set $iface managed no
        fi
      done
    done
    ip link set dev $ethdev up
  done
}
```

```
stop() {
  for _ethdev in ${ethlist//;/ } ; do
    ethdev=${_ethdev#*:}
    echo 0 > /sys/class/net/${ethdev}/device/sriov_numvfs
  done
}

ethlist=$(lookup unmanaged-devices /etc/NetworkManager/conf.d/10-tiger.conf

case $1 in
  startstop) "$1" ;;
esac

[root@rhel-tiger-14-6 ~]# touch /etc/systemd/system/tiger-nic-vf-handler.service

[root@rhel-tiger-14-6 ~]# chown root:root /etc/systemd/system/tiger-nic-vf-handler.service

[root@rhel-tiger-14-6 ~]# chmod 644 /etc/systemd/system/tiger-nic-vf-handler.service

[root@rhel-tiger-14-6 ~]# cat /etc/systemd/system/tiger-nic-vf-handler.service
[Unit]
Description=Create/Destroy unmanaged virtual functions and NetworkManager
Before=libvirtd.service

[Service]
Type=oneshot
ExecStart=/usr/local/bin/tiger-nic-vf-handler start
ExecStop=/usr/local/bin/tiger-nic-vf-handler stop
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target

[root@rhel-tiger-14-6 ~]# systemctl enable --now tiger-nic-vf-handler
Created symlink /etc/systemd/system/multi-user.target.wants/tiger-nic-vf-handler.service →
/etc/systemd/system/tiger-nic-vf-handler.service.
```

The script simply sources the list of unmanaged interfaces from the NetworkManager configuration and sets up the VFs accordingly. The filename in the script must match the filename created earlier. You may want to update the number of VFs created with this script by updating the script variable accordingly. The script is set to run before the libvirtd service is started to allow dependent configuration to be persisted.

The `spoofchk off` setting allows the VM to define the MAC addresses that it associates to interfaces rather than those that the host. This is important when considering the deployment of LAGs and for FortiGate Clustering Protocol vMAC operation.

The `trust on` setting is important to ensure that the VF tracks and follows the status of the PF, allowing the VM to detect interface down accordingly. This setting is also mandatory for LAG.

You may create the VFs as follows. Note the PCI addresses:

```
[root@rhel-tiger-14-6 ~]# lspci egrep "Eth.+Virt"
86:02.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
```

```
86:02.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:06.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:06.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0a.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0a.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0e.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0e.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
af:02.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
af:02.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
af:06.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
af:06.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
af:0a.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
af:0a.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
af:0e.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
af:0e.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02
```

Note that you can see the mapping of PF to VF as follows:

```
[root@rhel-tiger-14-6 ~]# ls -l /sys/class/net/ens4f0/device/virtfn*
lrwxrwxrwx 1 root root 0 Nov 17 22:44 /sys/class/net/ens4f0/device/virtfn0 -> ../0000:af:02.0
lrwxrwxrwx 1 root root 0 Nov 17 22:44 /sys/class/net/ens4f0/device/virtfn1 -> ../0000:af:02.1
```

virsh storage pool

A default storage pool is defined as a place for VM disks to be placed:

```
[root@rhel-tiger-14-6 ~]# mkdir -p /home/qemu

[root@rhel-tiger-14-6 ~]# chown qemu:qemu /home/qemu

[root@rhel-tiger-14-6 ~]# virsh pool-list
 Name   State   Autostart
-------------------------

[root@rhel-tiger-14-6 ~]# virsh pool-define-as --name default --type dir --target /home/qemu
Pool default defined

[root@rhel-tiger-14-6 ~]# virsh pool-autostart default
Pool default marked as autostarted

[root@rhel-tiger-14-6 ~]# virsh pool-start default
Pool default started

[root@rhel-tiger-14-6 libvirt]# virsh pool-list
 Name      State    Autostart
-------------------------------
 default   active   yes
```

virsh network

To avoid hardcoding the PCI address of a VF into a guest's configuration, a libvirt network with a device pool containing all of the VFs of an SR-IOV device is created. The guest then references this network, such that on guest startup, a VF is assigned from the pool:

```
[root@rhel-tiger-14-6 ~]# touch /var/lib/libvirt/network/ens4f0.xml

[root@rhel-tiger-14-6 ~]# chown root:root /var/lib/libvirt/network/ens4f0.xml

[root@rhel-tiger-14-6 ~]# chmod 644 /var/lib/libvirt/network/ens4f0.xml

[root@rhel-tiger-14-6 ~]# cat /var/lib/libvirt/network/ens4f0.xml
<network>
  <name>ens4f0-pool</name>
  <forward mode='hostdev' managed='yes'>
    <pf dev=ens4f0/>
  </forward>
</network>

[root@rhel-tiger-14-6 ~]# virsh net-define /var/lib/libvirt/network/ens4f0.xml
Network ens4f0-pool defined from /var/lib/libvirt/network/ens4f0.xml

[root@rhel-tiger-14-6 ~]# virsh net-autostart ens4f0-pool
Network ens4f0-pool marked as autostarted

[root@rhel-tiger-14-6 ~]# virsh net-start ens4f0-pool
Network ens4f0-pool started

[root@rhel-tiger-14-6 libvirt]# virsh net-list
 Name          State    Autostart   Persistent
----------------------------------------------
 default       active   yes         yes
 ens4f0-pool   active   yes         yes
 ens4f1-pool   active   yes         yes
 ens4f2-pool   active   yes         yes
 ens4f3-pool   active   yes         yes
 ens5f0-pool   active   yes         yes
 ens5f1-pool   active   yes         yes
 ens5f2-pool   active   yes         yes
 ens5f3-pool   active   yes         yes

[root@rhel-tiger-14-6 ~]# virsh net-dumpxml ens4f0-pool
<network>
  <name>ens4f0-pool</name>
  <uuid>bcb39207-503f-4af8-824c-45bf3756facf</uuid>
  <forward mode='hostdev' managed='yes'>
    <pf dev='ens4f0'/>
    <address type='pci' domain='0x0000' bus='0xaf' slot='0x02' function='0x0'/>
    <address type='pci' domain='0x0000' bus='0xaf' slot='0x02' function='0x1'/>
  </forward>
</network>
```

Because these networks are configured as autostart, you must ensure that the VFs have been created before the libvirtd service is started.

Virtual machine

The first thing that is required is the creation of a simple VM that you can edit to configure the exact configuration required:

```
 [root@rhel-tiger-14-6 ~]# mkdir /home/qemu/forticarrier-vm

 [root@rhel-tiger-14-6 ~]# cd /home/qemu/forticarrier-vm/

 [root@rhel-tiger-14-6 forticarrier-vm]# unzip /var/lib/libvirt/images/FGT_VM64_KVM-v7.0.2-
build0234-FORTINET.out.kvm.zip
 Archive:  /var/lib/libvirt/images/FGT_VM64_KVM-v7.0.2-build0234-FORTINET.out.kvm.zip
   inflating: fortios.qcow2

 [root@rhel-tiger-14-6 forticarrier-vm]# virt-install --noautoconsole --name FCR-VM-TIGER-14-48 --
memory 1024 --vcpus 1 --import --disk /home/qemu/forticarrier-vm/fortios.qcow2,size=2 --disk
/home/qemu/forticarrier-vm/fgt-logs.qcow2,size=30 --network type=direct,source=eno3,source_
mode=bridge,model=virtio --noreboot
 WARNING  No operating system detected, VM performance may suffer. Specify an OS with --os-variant
for optimal results.

 Starting install...
 Allocating 'fgt-logs.qcow2' | 30 GB  00:00:00
                                                                 Domain creation completed.
 You can restart your domain by running:
   virsh --connect qemu:///system start FCR-VM-TIGER-14-48

 [root@rhel-tiger-14-6 forticarrier-vm]# virsh list --all
  Id   Name                 State
 ------------------------------------
  -    FCR-VM-TIGER-14-48   shut off
```

You edit the VM definition using `virsh edit FCR-VM-TIGER-14-48`, which updates `/etc/libvirt/qemu/FCR-VM-TIGER-14-48.xml`. The following subsections discuss sections of this configuration to note. The libvirt.org website details the libvirt xml items.

Memory allocation

The `memory` is the maximum that the guest is allocated at boot time and the `currentMemory`, which defaults to the same value as `memory`, is the actual allocation of memory to the guest. In the example, 80 GB has been configured for both:

```
<memory unit='KiB'>83886080</memory>
<currentMemory unit='KiB'>83886080</currentMemory>
```

Hugepages

The hypervisor is configured to allocate the memory for the guest using hugepages instead of the native page size. The size matches the 1 GiB hugepages created earlier in Direct update on page 67.

```
  <memoryBacking>
    <hugepages>
      <page size='1048576' unit='KiB'/>
    </hugepages>
  </memoryBacking>
```

CPU pinning

`vcpu` represents the vCPU allocated and seen by the guest. `cpuset` represents the physical CPU thread. Comparing this to the earlier print in Tuned indirect update on page 68, only the CPUs from one NUMA node are selected, matching the memory, and this should correlate to the NIC.

`emulatorpin` specifies that the emulator is pinned to all vCPUs in the definition.

```
<vcpu placement='static'>32</vcpu>
  <cputune>
    <vcpupin vcpu='0' cpuset='5'/>
    <vcpupin vcpu='1' cpuset='41'/>
    <vcpupin vcpu='2' cpuset='7'/>
    <vcpupin vcpu='3' cpuset='43'/>
    <vcpupin vcpu='4' cpuset='9'/>
    <vcpupin vcpu='5' cpuset='45'/>
    <vcpupin vcpu='6' cpuset='11'/>
    <vcpupin vcpu='7' cpuset='47'/>
    <vcpupin vcpu='8' cpuset='13'/>
    <vcpupin vcpu='9' cpuset='49'/>
    <vcpupin vcpu='10' cpuset='15'/>
    <vcpupin vcpu='11' cpuset='51'/>
    <vcpupin vcpu='12' cpuset='17'/>
    <vcpupin vcpu='13' cpuset='53'/>
    <vcpupin vcpu='14' cpuset='19'/>
    <vcpupin vcpu='15' cpuset='55'/>
    <vcpupin vcpu='16' cpuset='21'/>
    <vcpupin vcpu='17' cpuset='57'/>
    <vcpupin vcpu='18' cpuset='23'/>
    <vcpupin vcpu='19' cpuset='59'/>
    <vcpupin vcpu='20' cpuset='25'/>
    <vcpupin vcpu='21' cpuset='61'/>
    <vcpupin vcpu='22' cpuset='27'/>
    <vcpupin vcpu='23' cpuset='63'/>
    <vcpupin vcpu='24' cpuset='29'/>
    <vcpupin vcpu='25' cpuset='65'/>
    <vcpupin vcpu='26' cpuset='31'/>
    <vcpupin vcpu='27' cpuset='67'/>
    <vcpupin vcpu='28' cpuset='33'/>
    <vcpupin vcpu='29' cpuset='69'/>
```

```
    <vcpupin vcpu='30' cpuset='35'/>
    <vcpupin vcpu='31' cpuset='71'/>
    <emulatorpin
cpuset='5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,41,43,45,47,49,51,53,55,57,59,61,63,65,67,69,
71'/>
  </cputune>
```

When considering CPU pinning for the best performance, hyperthreading may be a concern. Refer again to the `lstopo-no-graphics` output:

```
[root@rhel-tiger-14-6 ~]# lstopo-no-graphics
 Machine (187GB total)
 <output omitted for brevity>
   Package L#1
     NUMANode L#1 (P#1 94GB)
     L3 L#1 (25MB)
       L2 L#18 (1024KB) + L1d L#18 (32KB) + L1i L#18 (32KB) + Core L#18
         PU L#36 (P#1)
         PU L#37 (P#37)
       L2 L#19 (1024KB) + L1d L#19 (32KB) + L1i L#19 (32KB) + Core L#19
         PU L#38 (P#3)
         PU L#39 (P#39)
       L2 L#20 (1024KB) + L1d L#20 (32KB) + L1i L#20 (32KB) + Core L#20
         PU L#40 (P#5)
         PU L#41 (P#41)
       L2 L#21 (1024KB) + L1d L#21 (32KB) + L1i L#21 (32KB) + Core L#21
         PU L#42 (P#7)
         PU L#43 (P#43)
       L2 L#22 (1024KB) + L1d L#22 (32KB) + L1i L#22 (32KB) + Core L#22
         PU L#44 (P#9)
         PU L#45 (P#45)
       L2 L#23 (1024KB) + L1d L#23 (32KB) + L1i L#23 (32KB) + Core L#23
         PU L#46 (P#11)
         PU L#47 (P#47)
       L2 L#24 (1024KB) + L1d L#24 (32KB) + L1i L#24 (32KB) + Core L#24
         PU L#48 (P#13)
         PU L#49 (P#49)
       L2 L#25 (1024KB) + L1d L#25 (32KB) + L1i L#25 (32KB) + Core L#25
         PU L#50 (P#15)
         PU L#51 (P#51)
       L2 L#26 (1024KB) + L1d L#26 (32KB) + L1i L#26 (32KB) + Core L#26
         PU L#52 (P#17)
         PU L#53 (P#53)
       L2 L#27 (1024KB) + L1d L#27 (32KB) + L1i L#27 (32KB) + Core L#27
         PU L#54 (P#19)
         PU L#55 (P#55)
       L2 L#28 (1024KB) + L1d L#28 (32KB) + L1i L#28 (32KB) + Core L#28
         PU L#56 (P#21)
         PU L#57 (P#57)
       L2 L#29 (1024KB) + L1d L#29 (32KB) + L1i L#29 (32KB) + Core L#29
         PU L#58 (P#23)
         PU L#59 (P#59)
```

```
        L2 L#30 (1024KB) + L1d L#30 (32KB) + L1i L#30 (32KB) + Core L#30
          PU L#60 (P#25)
          PU L#61 (P#61)
        L2 L#31 (1024KB) + L1d L#31 (32KB) + L1i L#31 (32KB) + Core L#31
          PU L#62 (P#27)
          PU L#63 (P#63)
        L2 L#32 (1024KB) + L1d L#32 (32KB) + L1i L#32 (32KB) + Core L#32
          PU L#64 (P#29)
          PU L#65 (P#65)
        L2 L#33 (1024KB) + L1d L#33 (32KB) + L1i L#33 (32KB) + Core L#33
          PU L#66 (P#31)
          PU L#67 (P#67)
        L2 L#34 (1024KB) + L1d L#34 (32KB) + L1i L#34 (32KB) + Core L#34
          PU L#68 (P#33)
          PU L#69 (P#69)
        L2 L#35 (1024KB) + L1d L#35 (32KB) + L1i L#35 (32KB) + Core L#35
          PU L#70 (P#35)
          PU L#71 (P#71)
  <output omitted for brevity>
```

To get the best performance from a vCPU, it may be necessary to avoid using the second thread on the physical core. The `lstopo-no-graphics` output indicates that the hyperthreading causes two vCPUs to share the same L1 and L2 cache. For example, vCPUs 5 and 41 are sharing. This may lead to contention and performance degradation.

However, using the second core to increase the number of vCPUs is better than not providing it to the VM. The consideration is thus using the FortiGate-VM license effectively across the hardware.

The L3 cache is shared across the whole NUMA node in this case, so there is potential for contention there.

NUMA node tuning

The configuration ensures only local memory is allocated to the guest such that the remote NUMA memory is not used. The nodeset should therefore be the NUMA node associated to the NIC.

```
<numatune>
   <memory mode='strict' nodeset='1'/>
</numatune>
```

Management I/O

The management interface was defined during the creation of the VM:

```
<interface type='direct'>
   <mac address='52:54:00:fc:ea:99'/>
   <source dev='eno3' mode='bridge'/>
   <model type='virtio'/>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
```

The creation of this interface automatically adds a MAC address which may be less than random. That is, should another KVM host be in the same collision domain, without changing this parameter there could be a conflict, making the management access unreliable.

The OUI registered for QEMU is 52:54:00, and the following can be used to automatically generate and output a random MAC address to use instead and hopefully avoid this issue:

```
MACADDR="52:54:00:$(dd if=/dev/urandom bs=512 count=1 2>/dev/null | md5sum | sed 's/^\(..\)\(..\)\
      (..\).*$/\1:\2:\3/')"; echo $MACADDR
```

SR-IOV VF

The exact configuration depends on the environment and exactly what is needed from with it. Considerations include throughput requirements, functional separation, and resiliency model adopted.

```
<interface type='network'>
   <mac address='52:54:00:10:01:02'/>
   <source network='ens4f0-pool'/>
   <vlan>
      <tag id='1001'/>
   </vlan>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x08' function='0x0'/>
</interface>
```

The example here sets up an SR-IOV device from the previously defined pool. If considered useful, the MAC address can be set to help easily identify the interface in the VM (as long as there are no conflicts within the same collision domain). Importantly, a VLAN filter has been defined: This VLAN is transparent to the VM as the tag is added/removed without any knowledge at the VM. The external connectivity to the server (likely a switch) will need to support VLAN tagging appropriately. This VLAN is used to filter the traffic sent between the PF and VF, and as such is useful to prevent unwanted traffic being flooded to all VFs and to keep the traffic logically separated.

Starting the VM

The VM is configured fully, it can be started and the console can be used for the initial configuration of FortiOS. It is worth capturing a copy of the XML configuration.

```
 [root@rhel-tiger-14-6 ~]# virsh dumpxml FCR-VM-TIGER-14-48 > /home/qemu/forticarrier-vm/FCR-VM-
TIGER-14-48.xml

 [root@rhel-tiger-14-6 ~]# virsh start FCR-VM-TIGER-14-48
 Domain FCR-VM-TIGER-14-48 started

 [root@rhel-tiger-14-6 ~]# virsh console FCR-VM-TIGER-14-48
```

Once the VM is running, using the command `lstopo-no-graphics --ps` allows the verification of the CPU pinning, and so on.

FortiGate-VM

Now the hypervisor is configured and the VM is running. Consideration for the FGVM setup is needed. These considerations could impact the decisions made on the hypervisor, so some updates may be required.

Depending on the FortiGate-VM use case, you should consider one or more of the following:

- SR-IOV will typically be used for any performant deployment. This does have a drawback on the VM mobility.
- vSPU gives significant performance uplift in many scenarios and will continue to develop.
- Not using vSPU or not using all CPUs for vSPU may be more performant in some scenarios.
- Without vSPU, balancing interrupts across all CPUs by using affinity settings is key to getting maximum performance.

SR-IOV, LAGs, and affinity

For use cases that do not currently benefit from vSPU, the best that you can do is load balancing across all CPUs. You can best achieve this using SR-IOV, LAGs, and CPU affinity settings.

You likely need link aggregation, if not for throughput, for resiliency. Considerations for LAG differ when considering VFs, but the main concepts are the same. The following diagram represents an example LAG-based topology based on having two NIC cards, each with two ports in a single NUMA node.

This scenario tolerates the following:

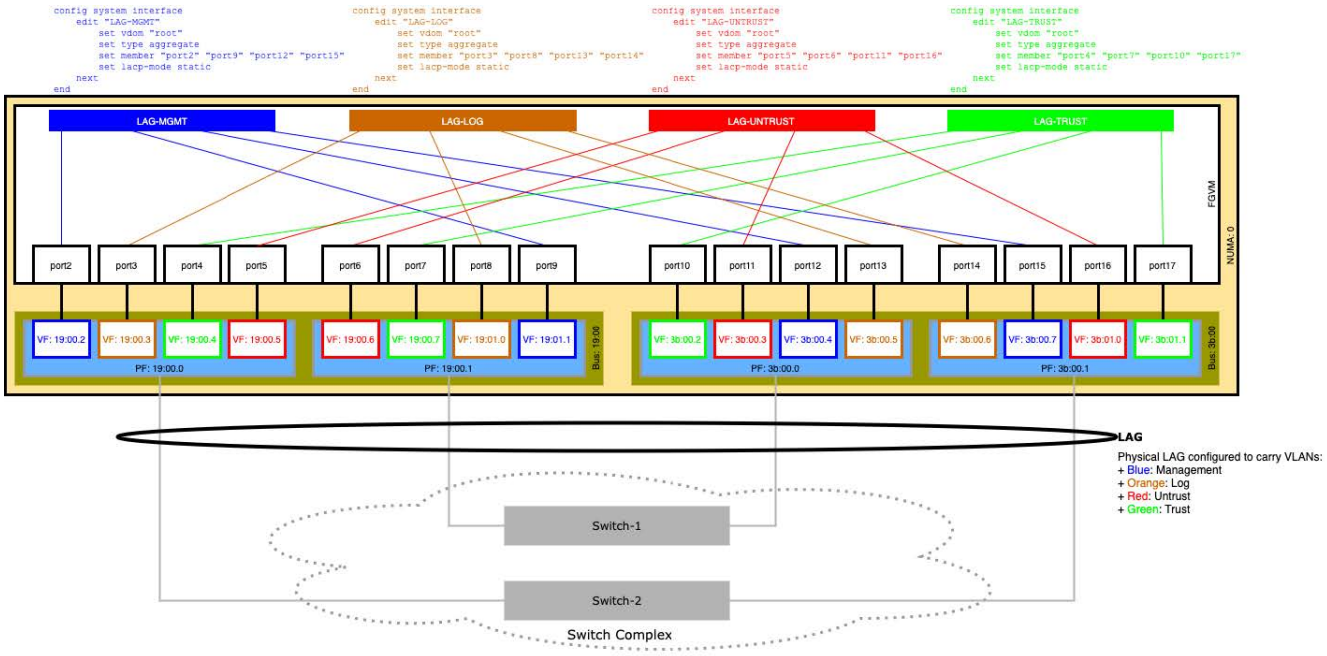
- NIC port/link failure
- NIC card failure
- Switch failure

The design also stresses the need for the `trust on` setting discussed earlier, as the VF must react upon the status of the PF, as LACP is not going to provide the functionality it would do in an

appliance-based deployment.

You must configure LACP mode as static in this deployment scenario.

In this diagram, the PF is using an external VLAN tag to separate traffic to the respective VFs and the VM is unaware of this external VLAN.



Without vSPU, there is no PMD, and the NIC uses the interrupts are used to signal that there is network traffic that the CPU must process. To get a performant system without using vSPU, you must take care to balance the amount of interrupts that each CPU receives.

Using the same layout as the diagram displays, find the relevant system interrupts/queues:

```
diagnose hardware sysinfo interrupts  grep "CPUport"
           CPU0       CPU1   <...>   CPU15
 47:      119912          0   <...>       0   PCI-MSI-edge      iavf-port2-TxRx-0
 48:           0     200309   <...>       0   PCI-MSI-edge      iavf-port2-TxRx-1
 49:           0          0   <...>       0   PCI-MSI-edge      iavf-port2-TxRx-2
 50:           0          0   <...>       0   PCI-MSI-edge      iavf-port2-TxRx-3
 <...>
 67:      254849          0   <...>       0   PCI-MSI-edge      iavf-port6-TxRx-0
 68:           0     443186   <...>       0   PCI-MSI-edge      iavf-port6-TxRx-1
 69:           0          0   <...>       0   PCI-MSI-edge      iavf-port6-TxRx-2
 70:           0          0   <...>       0   PCI-MSI-edge      iavf-port6-TxRx-3
 <...>
 87:       72971          0   <...>       0   PCI-MSI-edge      iavf-port10-TxRx-0
 88:           0     376044   <...>       0   PCI-MSI-edge      iavf-port10-TxRx-1
 89:           0          0   <...>       0   PCI-MSI-edge      iavf-port10-TxRx-2
 90:           0          0   <...>       0   PCI-MSI-edge      iavf-port10-TxRx-3
 <...>
 107:     197132          0   <...>       0   PCI-MSI-edge      iavf-port14-TxRx-0
 108:          0     421851   <...>       0   PCI-MSI-edge      iavf-port14-TxRx-1
 109:          0          0   <...>       0   PCI-MSI-edge      iavf-port14-TxRx-2
 110:          0          0   <...>       0   PCI-MSI-edge      iavf-port14-TxRx-3
 <...>
 122:          0          0   <...>       0   PCI-MSI-edge      iavf-port17-TxRx-0
 123:          0          0   <...>       0   PCI-MSI-edge      iavf-port17-TxRx-1
 124:          0          0   <...>       0   PCI-MSI-edge      iavf-port17-TxRx-2
```

```
125:         0          0   <...>  345768   PCI-MSI-edge      iavf-port17-TxRx-3
<...>
```

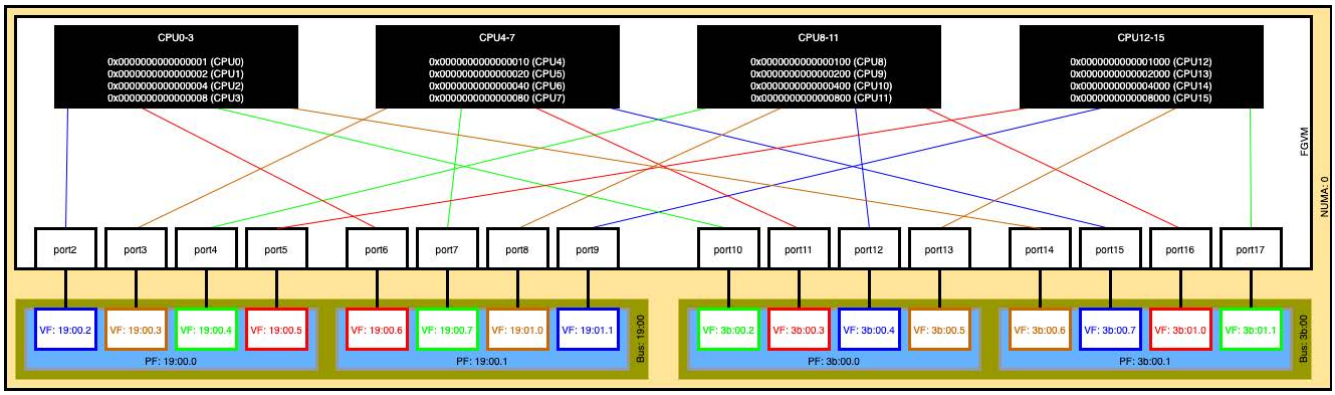
The interrupt names can differ. For example, the Mellanox ConnectX-5 NIC card has ten interrupts/queues per port named port2-0 through to port2-9.

The idea is to spread the interrupts across CPUs to balance the load across all system resources. Using the interrupt names as per the print, you can pin them to particular CPUs:

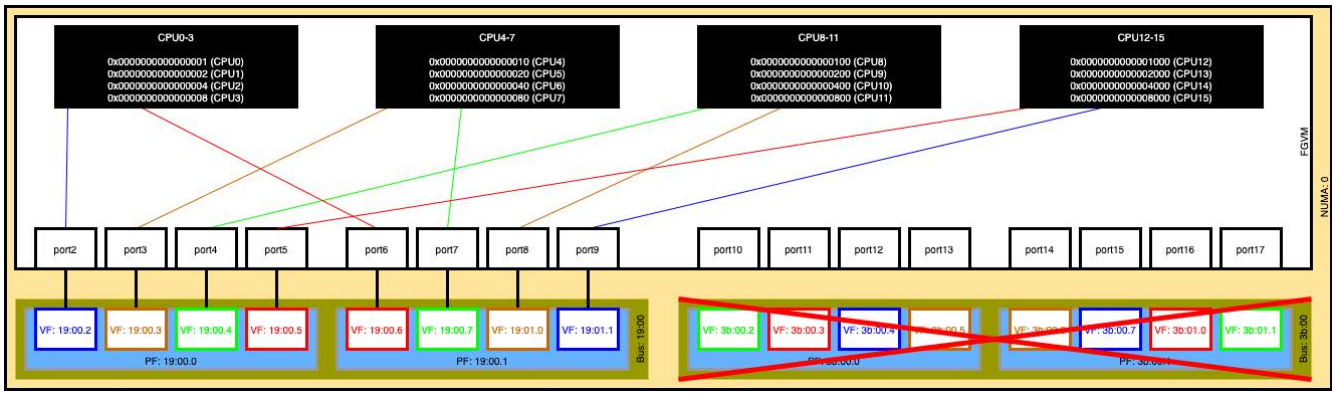
```
config system affinity-interrupt
    edit 20
        set interrupt "iavf-port2-TxRx-0"
        set affinity-cpumask "0x0000000000000001"
    next
    edit 21
        set interrupt "iavf-port2-TxRx-1"
        set affinity-cpumask "0x0000000000000002"
    next
    edit 22
        set interrupt "iavf-port2-TxRx-2"
        set affinity-cpumask "0x0000000000000004"
    next
    edit 23
        set interrupt "iavf-port2-TxRx-3"
        set affinity-cpumask "0x0000000000000008"
    next
<...>
    edit 60
        set interrupt "iavf-port6-TxRx-0"
        set affinity-cpumask "0x0000000000000001"
    next
    edit 61
        set interrupt "iavf-port6-TxRx-1"
        set affinity-cpumask "0x0000000000000002"
    next
    edit 62
        set interrupt "iavf-port6-TxRx-2"
        set affinity-cpumask "0x0000000000000004"
    next
    edit 63
        set interrupt "iavf-port6-TxRx-3"
        set affinity-cpumask "0x0000000000000008"
    next
<...>
    edit 170
        set interrupt "iavf-port17-TxRx-0"
        set affinity-cpumask "0x0000000000001000"
    next
    edit 171
        set interrupt "iavf-port17-TxRx-1"
        set affinity-cpumask "0x0000000000002000"
```

```
    next
    edit 172
        set interrupt "iavf-port17-TxRx-2"
        set affinity-cpumask "0x0000000000004000"
    next
    edit 173
        set interrupt "iavf-port17-TxRx-3"
        set affinity-cpumask "0x0000000000008000"
    next
end
```

This is a mapping of the four queues on an interface to one of four CPUs in a group, but also reusing the group of four CPUs across four interfaces as the following diagrams. This interleaving of the functions gets an even interrupt distribution, which gives the most performant deployment scenario.



In case of a failure, for example of the NIC card, this interleaving model ensures that the traffic interfaces where most traffic is expected are processed by different CPUs as the diagram shows, keeping the performance to a maximum.



Working out how best to balance the interrupts is the main thing to address in these circumstances. In the example case, each port has four queues/interrupts that you can map, making a VM16 effective with four PFs. The SR-IOV VLAN filtering and resultant LAG configuration provides interleaving, which helps balance the load across all CPUs.

Simularly, it may be that a VM32 is best serviced with eight PFs. It may be that the NIC card allows configuration of how many PFs are presented. For example, you may use an NIC presenting 4 x 10G more effectively across the CPUs than 1 x 40G.

Without much flexibility in using transparent VLANs or number of PFs, affining some services such as IPS, logging, or Web Filter to CPUs unused for traffic and providing effective CPU use may be the best option.

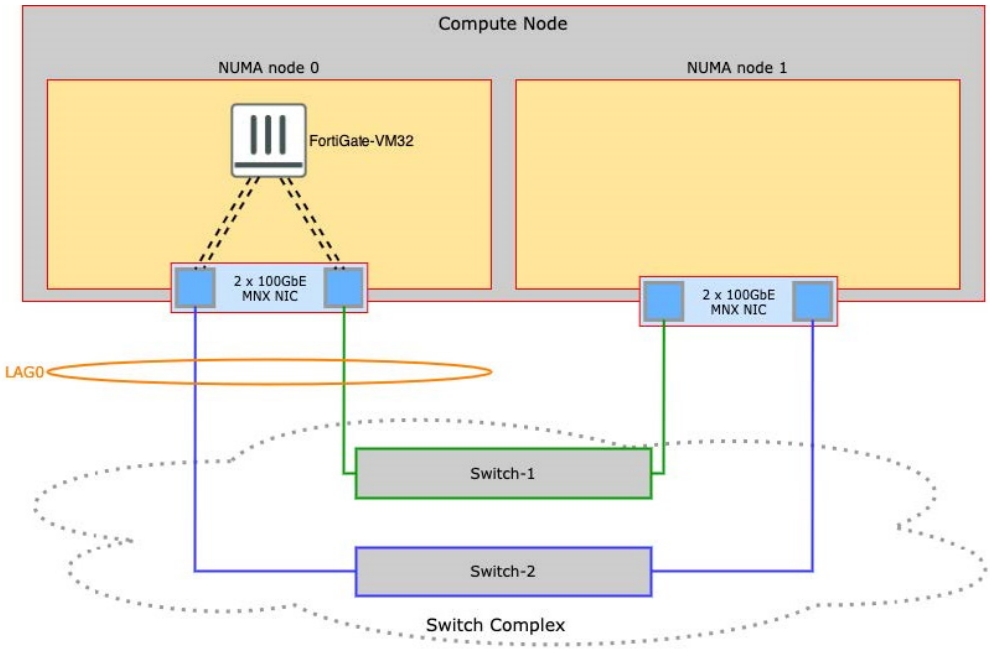
Effectively, there is significant flexibility, which should allow you to find a sweet spot of performance in most scenarios.

vSPU

Following is a diagram and the associated configuration. In the diagram, the blue square represents the PF. The dotted line represents the VF. Two VFs are defined for each PF. The configuration uses VLANs 1000 and 1001 to direct the traffic between VF and PF. The VM is unaware of the VLAN.

VLAN 1000 and VLAN 1001 are on opposites sides of the firewall. However, as this is presented to the VM as four devices, you can use link aggregation across the two PFs to cater for an element of resiliency.

The vSPU deployment negates the need for balancing interrupt requests and traffic is balanced across the vSPUs based upon IP headers.



FortiGate vSPU on page 57 describes balancing the traffic with vSPU. It does not require the configuration to use a balancing technique.

DPDK global settings

You must first enable DPDK and associate it to the interfaces which DPDK will polled for traffic. Enabling DPDK for the first time requires a system reboot.

```
config dpdk global
   set status enable
   set interface "port2" "port3" "port4" "port5"
   set multiqueue enable
```

```
   set sleep-on-idle disable
   set elasticbuffer disable
   set per-session-accounting traffic-log-only
   set hugepage-percentage 25
   set mbufpool-percentage 20
end
```

See DPDK global settings on page 47 for a detailed explanation of these configuration items.

You can then use these interfaces as normal in FortiOS. The following uses these interfaces to create LAGs to handle traffic:

```
config system interface
   edit "LAG-IN"
      set vdom "root"
      set ip 10.0.0.254 255.255.0.0
      set allowaccess ping
      set type aggregate
      set member "port2" "port3"
      set lldp-reception disable
      set lldp-transmission disable
      set snmp-index 7
      set lacp-mode static
   next
   edit "LAG-OUT"
      set vdom “root”
      set ip 10.1.0.254 255.255.0.0
      set allowaccess ping
      set type aggregate
      set member "port4" "port5"
      set lldp-reception disable
      set lldp-transmission disable
      set snmp-index 8
      set lacp-mode static
   next
end
```

DPDK CPU settings

The CPUs acting as DPDK engines are specified. They are four stages, a processing pipeline, for handling packets from Rx to vNP to IPS toTx. Generally, the simplest allocation model, enabling all CPUs to all stages, gives the best results.

```
config dpdk cpus
   set rx-cpus "0-31"
   set vnp-cpus "0-31"
   set ips-cpus "0-31"
   set tx-cpus "0-31"
end
```

See DPDK CPU settings on page 50 for a detailed explanation of these configuration items.

There are times when ringfencing CPUs to be used for purposes other than DPDK can provide a more performant system.

DPDK diagnostics

See DPDK diagnostic commands on page 52 for a detailed explanation of these configuration items.

Early initialization

You can use the DPDK early initialization log to check that you have configured DPDK correctly. For example, you can confirm that the CPUs and interfaces are bound to DPDK usage.

```
diagnose dpdk log show early-init
 ------------------------------------------------------------------
     DPDK early initialzation starts at 2020-11-26 09:53:14(UTC)
 ------------------------------------------------------------------
 Content of early configuration file:
     status=1
     multiqueue=1
     sleep-on-idle=0
     elasticbuffer=0
     per-session-accounting=1
     hugepage-percentage=25
     nr_hugepages=10090
     interfaces=port2 port3 port4 port5
     cpus=0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
     rxcpus=0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
     vnpcpus=0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
     ipscpus=0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
     txcpus=0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
 Parse config file success!

 Check CPU definitions 'cpus'
 Check CPU definitions 'rxcpus'
 Check CPU definitions 'ipscpus'
 Check CPU definitions 'vnpcpus'
 Check CPU definitions 'txcpus'
 Check CPUs success!

 Huge page allocation done

 Ports enabled for DPDK:
     port2
     port3
     port4
     port5
 Port name to device name mapping:
     port1: eth0
     port2: eth1
     port3: eth2
     port4: eth3
     port5: eth4
     port6: eth5
```

```
    port7: eth6
    port8: eth7
    port9: eth8
    port10: eth9
    port11: eth10
    port12: eth11
    port13: eth12
    port14: eth13
    port15: eth14
    port16: eth15
    port17: eth16
    port18: eth17
    port19: eth18
    port20: eth19
    port21: eth20
    port22: eth21
    port23: eth22
    port24: eth23

Start enabling DPDK kernel driver for port 'port2'...
Getting PCI device info for eth1...
reading pci dev /sys/class/net/eth1
link path: ../../devices/pci0000:00/0000:00:08.0/net/eth1
Device info of eth1:
    dev_name: eth1
    macaddr: 52:54:00:7c:08:31
    pci_vendor: 0x15b3
    pci_device: 0x1018
    pci_id: 0000:00:08.0
    pci_domain: 0
    pci_bus: 0
    pci_devid: 8
    pci_function: 0
    guid: n/a
Device eth1 is mlx5_core name changed to slv1
Creating DPDK kernel driver for device eth1...
Add VNP dev: eth1 PCI: 0000:00:08.0, Succeeded
DPDK kernel driver for eth1 successfully created
DPDK kernel driver enabled for port 'port2' (device name 'eth1')

Start enabling DPDK kernel driver for port 'port3'...
Getting PCI device info for eth2...
reading pci dev /sys/class/net/eth2
link path: ../../devices/pci0000:00/0000:00:09.0/net/eth2
Device info of eth2:
    dev_name: eth2
    macaddr: 52:54:00:7c:08:32
    pci_vendor: 0x15b3
    pci_device: 0x1018
    pci_id: 0000:00:09.0
    pci_domain: 0
    pci_bus: 0
```

```
    pci_devid: 9
    pci_function: 0
    guid: n/a
Device eth2 is mlx5_core name changed to slv2
Creating DPDK kernel driver for device eth2...
Add VNP dev: eth2 PCI: 0000:00:09.0, Succeeded
DPDK kernel driver for eth2 successfully created
DPDK kernel driver enabled for port 'port3' (device name 'eth2')

Start enabling DPDK kernel driver for port 'port4'...
Getting PCI device info for eth3...
reading pci dev /sys/class/net/eth3
link path: ../../devices/pci0000:00/0000:00:0a.0/net/eth3
Device info of eth3:
    dev_name: eth3
    macaddr: 52:54:00:7c:08:33
    pci_vendor: 0x15b3
    pci_device: 0x1018
    pci_id: 0000:00:0a.0
    pci_domain: 0
    pci_bus: 0
    pci_devid: 10
    pci_function: 0
    guid: n/a
Device eth3 is mlx5_core name changed to slv3
Creating DPDK kernel driver for device eth3...
Add VNP dev: eth3 PCI: 0000:00:0a.0, Succeeded
DPDK kernel driver for eth3 successfully created
DPDK kernel driver enabled for port 'port4' (device name 'eth3')

Start enabling DPDK kernel driver for port 'port5'...
Getting PCI device info for eth4...
reading pci dev /sys/class/net/eth4
link path: ../../devices/pci0000:00/0000:00:0b.0/net/eth4
Device info of eth4:
    dev_name: eth4
    macaddr: 52:54:00:7c:08:34
    pci_vendor: 0x15b3
    pci_device: 0x1018
    pci_id: 0000:00:0b.0
    pci_domain: 0
    pci_bus: 0
    pci_devid: 11
    pci_function: 0
    guid: n/a
Device eth4 is mlx5_core name changed to slv4
Creating DPDK kernel driver for device eth4...
Add VNP dev: eth4 PCI: 0000:00:0b.0, Succeeded
DPDK kernel driver for eth4 successfully created
DPDK kernel driver enabled for port 'port5' (device name 'eth4')
Bind ports success!
```

```
Make UIO nodes success!

DPDK sanity test passed
```

DPDK engine utilization

You can use `diagnose dpdk performance show` to see how the DPDK engines are loaded. This could be the information source for tuning the system or spotting irregular traffic load balancing.

> The CPU usage will be reported at 100% while the CPU is configured as a DPDK engine because of the PMD. This output is the source to look at for proper utilization reporting. These utilizations are available by SNMP.

MPStat

You can use the `mpstat` utility when trying to understand where a system is losing performance or to find an indication of an issue.

```
diagnose sys mpstat 2 3
Gathering data, wait 2 sec, press any key to quit.
..0..1
TIME         CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal   %idle
05:55:32 PM all   88.70    0.00   11.30    0.00    0.00    0.00    0.00    0.00
               0   90.00    0.00   10.00    0.00    0.00    0.00    0.00    0.00
               1   91.00    0.00    9.00    0.00    0.00    0.00    0.00    0.00
               2   89.00    0.00   11.00    0.00    0.00    0.00    0.00    0.00
               3   89.50    0.00   10.50    0.00    0.00    0.00    0.00    0.00
               4   86.50    0.00   13.50    0.00    0.00    0.00    0.00    0.00
               5   90.50    0.00    9.50    0.00    0.00    0.00    0.00    0.00
               6   89.00    0.00   11.00    0.00    0.00    0.00    0.00    0.00
               7   89.00    0.00   11.00    0.00    0.00    0.00    0.00    0.00
               8   88.50    0.00   11.50    0.00    0.00    0.00    0.00    0.00
               9   86.50    0.00   13.50    0.00    0.00    0.00    0.00    0.00
              10   88.50    0.00   11.50    0.00    0.00    0.00    0.00    0.00
              11   88.50    0.00   11.50    0.00    0.00    0.00    0.00    0.00
              12   88.50    0.00   11.50    0.00    0.00    0.00    0.00    0.00
              13   87.50    0.00   12.50    0.00    0.00    0.00    0.00    0.00
              14   89.00    0.00   11.00    0.00    0.00    0.00    0.00    0.00
              15   91.00    0.00    9.00    0.00    0.00    0.00    0.00    0.00
              16   90.50    0.00    9.50    0.00    0.00    0.00    0.00    0.00
              17   90.50    0.00    9.50    0.00    0.00    0.00    0.00    0.00
              18   91.50    0.00    8.50    0.00    0.00    0.00    0.00    0.00
              19   91.00    0.00    9.00    0.00    0.00    0.00    0.00    0.00
              20   85.00    0.00   15.00    0.00    0.00    0.00    0.00    0.00
              21   90.00    0.00   10.00    0.00    0.00    0.00    0.00    0.00
              22   87.50    0.00   12.50    0.00    0.00    0.00    0.00    0.00
              23   92.50    0.00    7.50    0.00    0.00    0.00    0.00    0.00
              24   89.50    0.00   10.50    0.00    0.00    0.00    0.00    0.00
```

```
              25   84.00    0.00   16.00    0.00    0.00    0.00    0.00    0.00
              26   85.00    0.00   15.00    0.00    0.00    0.00    0.00    0.00
              27   91.00    0.00    9.00    0.00    0.00    0.00    0.00    0.00
              28   87.50    0.00   12.50    0.00    0.00    0.00    0.00    0.00
              29   84.50    0.00   15.50    0.00    0.00    0.00    0.00    0.00
              30   87.50    0.00   12.50    0.00    0.00    0.00    0.00    0.00
              31   88.50    0.00   11.50    0.00    0.00    0.00    0.00    0.00
<output omitted for brevity>
```

As discussed, the idle time for this print reports as 0% because of PMD.

Of particular importance in this print is `%steal`. If this is not zero, something is not optimized, as the hypervisor is stealing CPU cycles from the VM.

Change log

Date	Change description
2026-04-21	Initial release.

www.fortinet.com