

VSI OpenVMS on KVM

Configuration Guide

Table of contents

1. Introduction	4
2. Before you begin	5
2.1 OpenVMS on KVM Prerequisites	5
2.1.1 VSI OpenVMS Version Requirements	5
2.1.2 Hardware Requirements	5
2.1.3 Recommended KVM Virtual Machine Settings	5
2.1.4 Installation Requirements	7
2.2 KVM Configuration Methods	9
2.3 Interacting With a KVM VM as an Unprivileged User	11
3. Creating a KVM Virtual Machine	12
4. Booting VSI OpenVMS on KVM	15
5. Connecting to the OpenVMS Console	16
6. Administering a KVM Virtual Machine	19
6.1 Cloning a KVM Virtual Machine	19
6.2 Deleting a KVM Virtual Machine	20
6.3 Managing Disks on KVM	21
6.3.1 Viewing Disk Properties	21
6.3.2 Creating Disks	22
6.3.3 Increasing Disk Size	22
6.3.4 Attaching Disks to an Existing Virtual Machine	23
6.3.5 Converting Images	24
6.4 Managing Network on KVM	25
6.4.1 Adding a Network Interface to an Existing Virtual Machine	25
6.4.2 Network Bridges	26
6.4.2.1 Adding a Bridge Interface to an Existing KVM Virtual Machine	26
6.4.2.2 Next Steps	27
6.5 Advanced Virtio-net Devices Configuration	28
6.6 Disabling SELinux File Labeling	29
7. Customizing a KVM Virtual Machine	30
8. Advanced Configuration	31
8.1 Secure Boot	31
8.1.1 Secure Boot Overview	31
8.1.2 Secure Boot in KVM Configuration	32
8.1.2.1 Secure Boot in UEFI	33
8.1.3 Adding OpenVMS Certificate to the Machine Owner Key Database	35

8.2	Configuring Fibre Channel PCI Passthrough on KVM Virtual Machine	37
8.2.1	Fibre Channel PCI Passthrough on KVM Virtual Machine Overview	37
8.2.1.1	Supported Platforms and Hardware	37
8.2.1.2	Prerequisites	38
8.2.2	IOMMU Hardware Extensions in UEFI/BIOS	39
8.2.2.1	Enabling IOMMU Hardware Extensions in UEFI/BIOS	39
8.2.3	IOMMU Support in Linux Kernel	41
8.2.3.1	Enabling IOMMU in Linux Kernel	41
8.2.4	Assigning Fibre Channel HBA to Virtual Machine	43
8.2.4.1	Verification	46
8.3	Booting OpenVMS from a Fibre Channel Disk	48
8.4	Mapping Storage Controllers to Disk Devices	54
8.4.1	Preparing to Map Storage Controllers to Disk Devices	54
8.4.2	SATA Controllers	55
8.4.3	Virtio-SCSI Controllers	57
8.4.4	Verification on OpenVMS	59
9.	Troubleshooting	60
9.1	KVM Installation and Configuration Issues	60
9.1.1	Error Messages	60
9.1.1.1	<code>command not found</code>	60
9.1.1.2	<code>No ACPI MCFG table was found. This processor may not have a PCIe configuration space.</code>	60
9.1.1.3	<code>(virt-manager:54736): Gtk-WARNING **: 08:42:13.294: cannot open display:</code>	61
9.1.2	Rainbow Screen	61
9.1.3	Non-Persistent Device Boot Order	62
9.1.4	System Time Synchronization	62
9.2	IOMMU Configuration Issues	63
9.2.1	System Hang	64
9.2.2	Error Messages	65
10.	Reference	66
10.1	virt-install Command Reference	66

1. Introduction

Configure a KVM virtual machine on Oracle Linux or Red Hat Enterprise Linux for use with VSI OpenVMS.

While other distributions of Linux are not officially supported by VSI, the information presented in this guide should still be applicable.

2. Before you begin

2.1 OpenVMS on KVM Prerequisites

Review the requirements and prerequisites below to prepare to configure your KVM virtual machine (VM).

2.1.1 VSI OpenVMS Version Requirements

VSI OpenVMS [Version 9.2-2](#) or higher

2.1.2 Hardware Requirements

See the [Supported Platforms](#) page for the currently supported hardware.

2.1.3 Recommended KVM Virtual Machine Settings

VSI recommends that your OpenVMS x86-64 virtual machine be configured with the following minimum settings depending on your version of VSI OpenVMS:

V9.2-2

Settings	Requirements
Minimum number of CPUs	2
Chipset	Q35
Memory	6 GB. If the installation fails due to lack of memory, you can temporarily increase it for the installation and reduce it afterwards if your host system has enough RAM to allocate.
Operating system	Other 64-bit
Virtual disk	At least 30 GB for the system files. Optionally, if disk space is limited, use a dump off system disk (DOSD); the recommended minimum DOSD size is 8 GB. For more information, refer to the VSI OpenVMS x86-64 V9.2-2 Release Notes .
Boot option	• Disable Secure Boot (if present) • Enable UEFI or EFI (BIOS is not supported)
Console communication	Performed via <code>virsh</code> . Note that only COM1 serial port is supported.
Serial line	An additional serial line for non-console login.

Settings	Requirements
Network	An E1000 adapter. VSI strongly advises against manual configuration of a DECnet MAC address on any interface.
CD/DVD device	Physical or virtual, with the OpenVMS installation kit (.ISO file).

V9.2-3

Settings	Requirements
Minimum number of CPUs	2
Chipset	Q35
Memory	6 GB. If the installation fails due to lack of memory, you can temporarily increase it for the installation and reduce it afterwards if your host system has enough RAM to allocate.
Operating system	Other 64-bit
Virtual disk	At least 30 GB for the system files. Optionally, if disk space is limited, use a dump off system disk (DOSD); the recommended minimum DOSD size is 8 GB. For more information, refer to the VSI OpenVMS x86-64 V9.2-3 Release Notes .
Boot option	• Disable Secure Boot (if present) • Enable UEFI or EFI (BIOS is not supported)
Console communication	Performed via <code>virsh</code> . Note that only COM1 serial port is currently supported. Guest Console can be used as an alternative to serial ports.
Network	An E1000 or an E1000e adapter. Also, two types of para-virtualized virtio Network Interface Cards (NICs). VSI strongly advises against manual configuration of a DECnet MAC address on any interface.
CD/DVD device	Physical or virtual, with the OpenVMS installation kit (.ISO file) assigned to it.

E9.2-4

Settings	Requirements
Minimum number of CPUs	2

Settings	Requirements
Chipset	Q35
Memory	6 GB. If the installation fails due to lack of memory, you can temporarily increase it for the installation and reduce it afterwards if your host system has enough RAM to allocate.
Operating system	Other 64-bit
Virtual disk	At least 30 GB for the system files. Optionally, if disk space is limited, use a dump off system disk (DOSD); the recommended minimum DOSD size is 8 GB. For more information, refer to the VSI OpenVMS x86-64 E9.2-4 Release Notes .
Boot option	Enable UEFI or EFI (BIOS is not supported)
Console communication	Performed via <code>virsh</code> . Note that only COM1 serial port is currently supported. Guest Console can be used as an alternative to serial ports.
Network	An E1000 or an E1000e adapter. Also, two types of para-virtualized virtio Network Interface Cards (NICs). VSI strongly advises against manual configuration of a DECnet MAC address on any interface.
CD/DVD device	Physical or virtual, with the OpenVMS installation kit (.ISO file) assigned to it.

2.1.4 Installation Requirements

Before you proceed with installing your KVM virtual machine (VM), you must perform the following steps from the CLI to configure your environment.

1. Open your terminal and run the following commands to install and enable the packages required to perform operations with your virtual machine:

```
sudo dnf install qemu-kvm libvirt virt-install virt-manager
sudo systemctl enable --now libvirtd
```

2. Edit the `/etc/libvirt/qemu.conf` file in the following way:

- Uncomment the line `#user = "qemu"` by removing the leading hash sign (#) and replace `qemu` with the name of your Linux user or the name of a service account.
- Uncomment the line `#group = "qemu"` by removing the leading hash sign (#) and replace `qemu` with `libvirt`.

Make sure that both `user` and `group` are set as described above. Otherwise, you will not be able to use your virtual machine.

The example below shows what the two lines must look like:

```
# Some examples of valid values are:
#
#     user = "qemu"    # A user named "qemu"
#     user = "+0"      # Super user (uid=0)
#     user = "100"     # A user named "100" or a user with uid=100
#
user = "username"
# The group for QEMU processes run by the system instance. It can be
# specified in a similar way to user.
group = "libvirt"
```

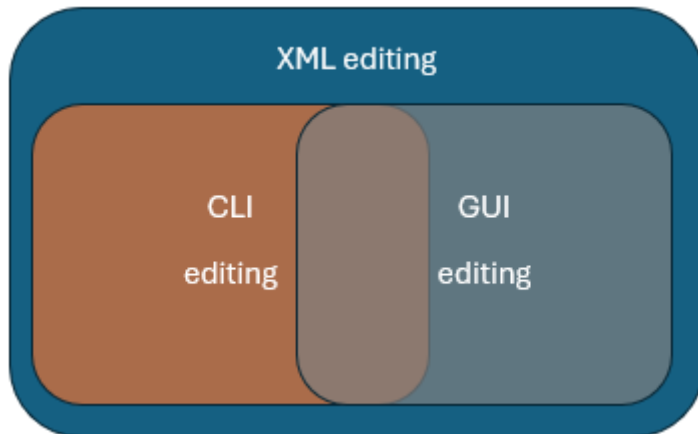
3. Now, for the changes to take effect, execute the following command:

```
sudo systemctl restart libvirtd
```

2.2 KVM Configuration Methods

There are several methods to configure a KVM virtual machine, and each method is suitable for different purposes.

The diagram below shows the scope of customization possibilities depending on the method that you are using – CLI, GUI, or editing the XML file directly.



There are pros and cons to each configuration method. Review the table below to decide which method is appropriate for you.

	Pros	Cons
CLI (virsh)	• Allows for a quick start with predefined templates or scripts. • Best for default deployments.	• Special configurations require crafting long commands. • Command syntax can change with updates.
Virtual Machine Manager GUI (virt-manager)	• Best for testing new features, prototyping, and learning. • Quite powerful, especially when coupled with the internal XML editor.	Some settings are unavailable.
Web-based GUI (Cockpit)	• Friendly UI. • Provides easy management of configured virtual machines (can be done remotely via a web interface).	Limited ability to create virtual machines and change essential settings.

	Pros	Cons
XML editing	• Best for advanced deployments. • Provides the most customizability.	Higher risk of configuration errors.

Regardless of the method you use to configure your KVM virtual machine, all settings will be stored in an XML configuration file.

This guide discusses how to perform tasks with your KVM virtual machine using CLI, Virtual Machine Manager GUI, or by editing the XML configuration directly.

 Note

A window manager is required on Linux to use the Virtual Machine Manager GUI.

The Cockpit GUI is also a good tool for managing a KVM virtual machine without the need for a terminal, as you can access your virtual machine's serial console via a browser. Further documentation on Cockpit can be found in this [Red Hat Enterprise Linux article](#).

2.3 Interacting With a KVM VM as an Unprivileged User

For a single unprivileged user to be able to execute KVM-related commands without `sudo`, the following steps must be performed:

1. Add the unprivileged user to the `libvirt` group by executing the following command:

```
sudo usermod -aG libvirt username
```

2. When logged in as the unprivileged user, run the following:

```
export LIBVIRT_DEFAULT_URI="qemu:///system"
```

Tip

This change is not persistent when logging out or restarting the system. To make it persistent, add the `export LIBVIRT_DEFAULT_URI="qemu:///system"` command to the `~/.bashrc` file. This way, the command will be executed each time the user logs in to the system.

3. Creating a KVM Virtual Machine

If this is your first time dealing with KVM, VSI recommends that you start by copying and pasting the commands from the CLI examples in the sections below.

Before you begin setting up your KVM virtual machine:

1. Review the requirements and recommended settings for your KVM virtual machine discussed in [OpenVMS on KVM Prerequisites](#).
2. Download the VSI OpenVMS OE ISO file from the [VSI Service Portal](#).
3. Configure a network bridge. For instructions on how to do so, refer to the [Configuring and managing networking guide](#).

The command below will create a working virtual machine configuration depending on your version of VSI OpenVMS for x86-64.

V9.2-3 and earlier

Copy the command and use your values for the parts in bold.

```
bridgemac="bridge-interface-mac"
vmname="virtual-machine-name"
vmdesc="virtual-machine-description"
isopath="/path/to/vms/install.iso"
diskpath="/path/to/vms/disk.qcow2"
bridgename="network-bridge-name"
sudo virt-install \
--name $vmname \
--description $vmdesc \
--memory=8192 \
--vcpus=4 \
--cpu host-model \
--controller type=scsi,model=virtio-scsi \
--disk device=cdrom,path=$isopath,bus=sata \
--disk path=$diskpath,\
size=30,bus=scsi,cache=writethrough,sparse=no \
--network=bridge:$bridgename,mac=$bridgemac,model='virtio-net' \
--machine q35 \
--os-variant none \
--boot fd \
--boot loader=/usr/share/OVMF/OVMF_CODE.secboot.fd,loader.readonly=yes,\
loader.type=pflash,loader_secure=yes \
--features vmport.state=off,smm.state=on \
--watchdog model=itco,action=none \
--serial pty \
--noreboot \
--autostart
```

E9.2-4

Copy the command and use your values for the parts in bold.

```

bridgemac="bridge-interface-mac"
vmname="virtual-machine-name"
vmdesc="virtual-machine-description"
isopath="/path/to/vms/install.iso"
diskpath="/path/to/vms/disk.qcow2"
bridgename="network-bridge-name"
sudo virt-install \
--name $vmname \
--description $vmdesc \
--memory=8192 \
--vcpus=4 \
--cpu host-model \
--controller type=scsi,model=virtio-scsi \
--disk device=cdrom,path=$isopath,bus=sata \
--disk path=$diskpath,\
size=30,bus=scsi,cache=writethrough,sparse=no \
--network=bridge:$bridgename,mac=$bridgemac,model='virtio-net' \
--machine q35 \
--os-variant none \
--boot fd \
--boot loader=/usr/share/edk2/ovmf/OVMF_CODE.secboot.fd,\
loader.readonly=yes,loader.secure=yes,loader.type=pflash,\
nvram.template=/usr/share/edk2/ovmf/OVMF_VARS.secboot.fd \
--features vmport.state=off,smm.state=on \
--watchdog model=itco,action=none \
--serial pty \
--noreboot \
--autostart

```

For a detailed description of the `virt-install` command used in the examples above, along with the recommended settings and alternatives that may depend on your system requirements, refer to [virt-install Command Reference](#).

For the full documentation on the `virt-install` command, run the following:

```
man virt-install
```

After the `virt-install` is executed, your virtual machine will be installed and configured, but will not be running.

In order to start the virtual machine instance, run:

```
sudo virsh start vm-name
```

Attention

If you are an unprivileged user, perform the steps described in [Interacting with a VM as an Unprivileged User](#) before you start using your KVM virtual machine.

 Caution

VSI does not recommend creating new virtual machines using the Cockpit GUI, as necessary customization options are not available there. However, this can be an easy way to manage your virtual machines once they are created. See [KVM Configuration Methods](#) for details.

4. Booting VSI OpenVMS on KVM

The booting procedure may differ depending on your version of VSI OpenVMS.

V9.2-3 and earlier

At the `BOOTMGR>` prompt, execute **BOOT *device-name***, where *device-name* is the name of the virtual CD/DVD device to which your VSI OpenVMS installation ISO is assigned.

For more information, see the [VSI OpenVMS x86-64 Boot Manager User Guide](#).

E9.2-4

1. At the `BOOTMGR>` prompt, execute **BOOT *device-name***, where *device-name* is the name of the virtual CD/DVD device to which your VSI OpenVMS installation ISO is assigned.

Alternatively, you can use a Fibre Channel disk as a bootable device. See [Booting OpenVMS from a Fibre Channel Disk](#) on how to configure the UEFI accordingly.

For more information, see the [VSI OpenVMS x86-64 Boot Manager User Guide](#).

2. It is possible to boot VSI OpenVMS with Secure Boot enabled. To do so, you are required to register an OpenVMS-specific certificate first.

- If you have created a virtual machine with Secure Boot enabled by default (for example, using the script in the [Creating a KVM Virtual Machine](#) section), follow the procedure for manual certificate registration in [Adding OpenVMS Certificate to the Machine Owner Key Database](#).
- If the Secure Boot functionality is disabled by default in your VM configuration, refer to the [Secure Boot](#) section for more details on how to enable it.

5. Connecting to the OpenVMS Console

You can connect to your OpenVMS KVM virtual machine using the methods discussed in this section.

via Guest Console

Info

You need the Virtual Machine Manager GUI installed on your system and graphics enabled in your VM to be able to use the Guest Console.

1. Open **Virtual Machine Manager** either from the **Application Launcher** or by running the `virt-manager` command in your CLI. A new window will open.
2. Double-click the VM where you wish to start the Guest Console. The Guest Console will open in a new window.
3. At the `BOOTMGR>` prompt, enter **OPAO** to connect to the operator terminal.

For more information about the Guest Console, refer to the [VSI OpenVMS x86-64 V9.2-3 Installation Guide](#) and the [VSI OpenVMS x86-64 Boot Manager User Guide](#).

via virsh

Use the following command to connect to the OpenVMS operator terminal:

```
sudo virsh console vm-name
```

via Terminal Emulator

Follow the steps below to connect to your KVM virtual machine via Telnet using a terminal emulator.

1. Open your terminal emulator (e.g. PuTTY) and specify the IP address and port of your virtual machine.

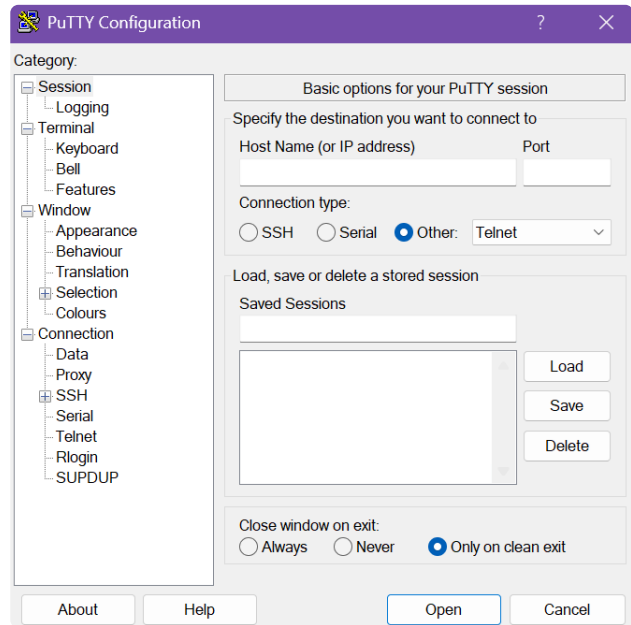
Note

The following steps contain examples of how to configure the PuTTY terminal emulator.

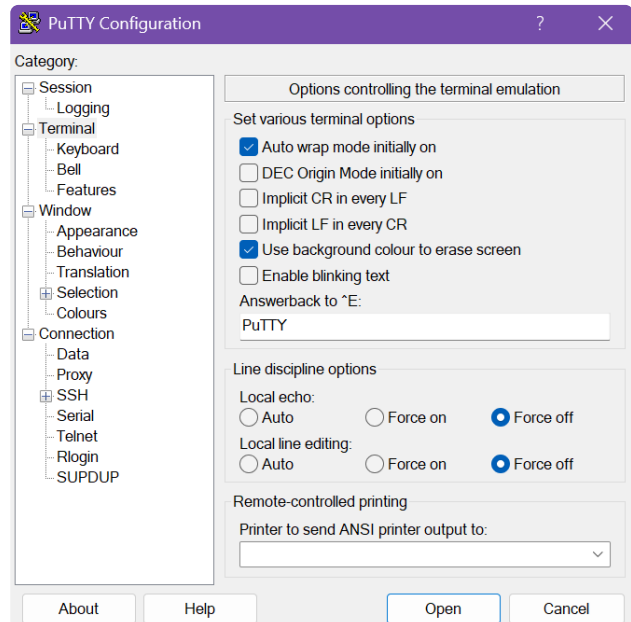
While VSI tests with PuTTY, other terminal utilities can also be used, but may have different restrictions and incompatibilities.

2. Edit your terminal emulator settings as specified below:

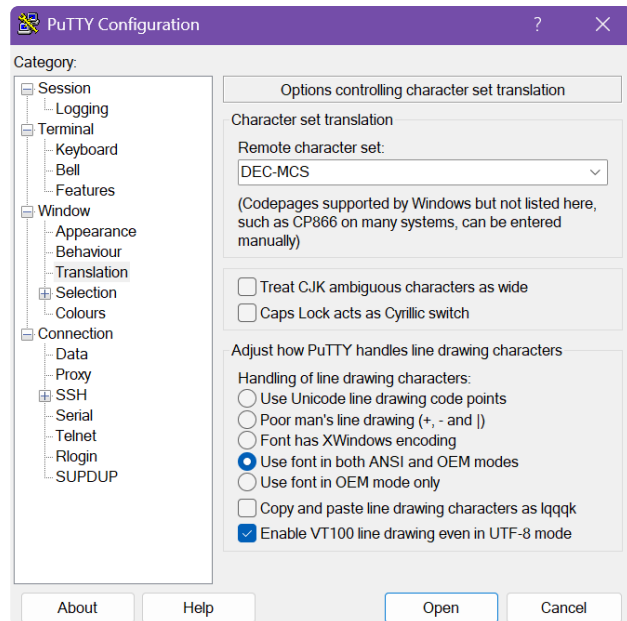
- **Connection type:** Telnet



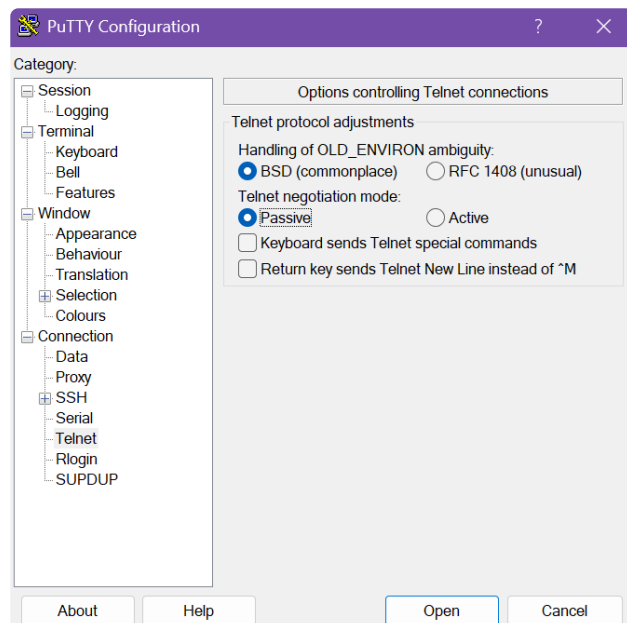
- **Implicit CR in every LF:** Unchecked
- **Implicit LF in every CR:** Unchecked
- **Local echo:** Force off
- **Local line editing:** Force off



- **Remote character set:** DEC-MCS
- **Enable VT100 line drawing even in UTF-8 mode:** Checked



- **Telnet negotiation mode:** Passive
- **Return Key Sends Telnet New Line instead of ^M:** Unchecked



3. Click **Open**.

6. Administering a KVM Virtual Machine

6.1 Cloning a KVM Virtual Machine

VSI recommends cloning your KVM virtual machine using CLI to avoid data loss and ensure proper cloning.

1. Turn off your virtual machine using the `virsh destroy vm-name` command.
2. Run the following:

```
virt-clone -o original-vm-name -n new-vm-name --auto-clone --nonsparse
```

This command will clone the VM along with the associated disks and ensure that they are thick-provisioned.

Note

Thick provisioning of OpenVMS disks corresponds to full preallocation of disk space in KVM virtual machines configuration.

6.2 Deleting a KVM Virtual Machine

Depending on the preferred method, follow the procedures below to delete your KVM virtual machine.

CLI

Run the following command:

```
virsh undefine vm-name --nvram
```

This command will delete the VM but keep the associated disks. If you want to remove the disks as well, add `--remove-all-storage` to the command.

Caution

If your installation ISO is attached at the time this command is run, it will also be deleted.

GUI

1. Open **Virtual Machine Manager** either from the **Application Launcher** or by running the `virt-manager` command in your CLI. A new window will open.
2. Right-click on the VM you wish to delete and click **Delete** in the context menu.
3. You will be prompted to delete the associated disk files. Select the ones that you want to delete and click **Delete**. Confirm the delete action if necessary.

6.3 Managing Disks on KVM

Use the `qemu-img` command to manage your disks.

For the full documentation on `qemu-img`, refer to <https://www.qemu.org/docs/master/tools/qemu-img.html>.

6.3.1 Viewing Disk Properties

To view the properties of a disk file, use the following command, specifying the name of your disk file:

```
qemu-img info disk-file-name.qcow2
```

Note

If the disk file is in use by the virtual machine, you must turn it off first or add the `-U` argument on the `qemu-img` command. However, using this argument is not considered safe, because it forces the image to be shared between processes.

The output will look similar to the following:

```
image: disk.qcow2
file format: qcow2
virtual size: 1 GiB (1073741824 bytes)
disk size: 1 GiB
cluster_size: 65536
Format specific information:
  compat: 1.1
  compression type: zlib
  lazy refcounts: false
  refcount bits: 16
  corrupt: false
  extended l2: false
Child node '/file':
  filename: disk.qcow2
  protocol type: file
  file length: 1 GiB (1074135040 bytes)
  disk size: 1 GiB
  Format specific information:
    extent size hint: 1048576
```

Note the difference between `virtual size` and `disk size`. The `virtual size` is the size that the file is allowed to expand to, whereas `disk size` specifies the actual space it takes up on the host machine.

If your disk is not fully allocated/thick-provisioned, the `virtual size` value is likely higher than the `disk size` value.

✖ Restriction

VSI OpenVMS does *not* support thin-provisioned disks, so you must always check that your disk is thick-provisioned.

6.3.2 Creating Disks

Before you create a disk, check the [Recommended KVM Virtual Machine Settings](#) for your version of VSI OpenVMS.

✖ Restriction

VSI OpenVMS does *not* support disks larger than 1.98 TB.

Below is an example of a command that creates a thick-provisioned 20 GB disk.

```
qemu-img create -f qcow2 -o preallocation=full disk-name.qcow2 20G
```

The arguments used in this command, along with the recommended options, are explained in the table below.

Arguments	Argument options	Description	Recommended options
-f		The file format for the disk.	It is recommended that you use <code>qcow2</code> or <code>raw</code> . <code>qcow2</code> supports encryption and compression, while <code>raw</code> provides marginally better performance.
-o		Specifies the path to the target file.	You <i>must</i> specify <code>preallocation=full</code> when creating a new disk for OpenVMS.

6.3.3 Increasing Disk Size

To resize a hard drive, use the `qemu-img resize` command. In the following example, the disk size is expanded by 10 GB.

```
qemu-img resize --preallocation=full -f qcow2 disk-name.qcow2 +10G
```

Notice the difference in syntax compared to creating a disk. Instead of using the `-o` switch, `--preallocation=full` is specified. This is necessary to keep the disk thick-provisioned, as by default, the `resize` command only extends the virtual size of the disk, it does not pre-allocate the space for it, effectively converting the disk to thin-provisioned, which is not supported on OpenVMS.

Warning

VSI does not recommend using the `resize` utility to shrink the disk as this can lead to data loss.

6.3.4 Attaching Disks to an Existing Virtual Machine

To attach a disk to an existing VM, you must first identify what device is next available for mounting.

1. Execute the following command on your host system to list the devices used by your virtual machine:

```
sudo virsh domblklist vm-name
```

The output will be similar to the following:

Target	Source
sda	/home/user/X86E9240E.ISO
sdb	/home/user/disk.qcow2

The Target column shows the virtual device names that are already in use. In this case, `sdc` can be the next available virtual device name.

Tip

The letters a, b, c, etc. in the virtual device (target) name correspond to the `unit` value in the disk device configuration and the `index` value in the assigned SCSI controller configuration. a corresponds to the value 0, b corresponds to 1, etc.

2. Run the following command, replacing `vm-name`, `/path/to/disk/disk-name.qcow2`, and `target-device-name` with appropriate values:

```
sudo virsh attach-disk vm-name /path/to/disk/disk-name.qcow2 target-device-name --driver qemu --subdriver qcow2 --cache writethrough --persistent
```

By default, the disk is attached to the VM as a SCSI device. In order to attach the disk as a SATA device, add `--targetbus sata` to the command line above.

The arguments used in this command, along with the recommended options, are explained in the table below.

Arguments	Argument options	Description	Recommended options
<code>--driver</code>	<code>qemu</code>		
<code>--subdriver</code>			This should be the same format as the disk, i.e. if your disk is <code>.qcow2</code> , specify <code>qcow2</code> .
<code>--cache</code>	<code>writethrough</code>	The cache mode to be used by the disk.	This is required to be <code>writethrough</code> .
<code>--persistent</code>		This makes the disk a permanent attachment.	

For the full documentation on the `virsh attach-disk` command, refer to the [attach-disk usage description](#).

Important

In certain scenarios, adding more than five Virtio-SCSI disks may result in KVM generating an additional controller using the lsilogic model name. This must be fixed by remapping the additional devices to a Virtio-SCSI controller as described in [Virtio-SCSI Controllers](#).

6.3.5 Converting Images

It is possible to convert images from one format to another using the `qemu-img convert` command.

For the full documentation on the `qemu-img convert` command, refer to the [Converting between image formats](#) article.

For information on how to convert `raw/qcow2` disks to `vmdk` disks supported on VMware ESXi, refer to the [Converting Hypervisor Images](#) section.

6.4 Managing Network on KVM

6.4.1 Adding a Network Interface to an Existing Virtual Machine

You can add a network interface to an existing virtual machine via CLI, GUI, or by editing the XML configuration directly.

CLI

Run the following command, replacing *vm-name* with the name of your VM, *interface-name* with the name of the interface, and *address* with the MAC address of the interface you want to add:

```
sudo virsh attach-interface vm-name --model virtio \
--target macvTapDirect --type direct --source interface-name \
--config --live --managed --mac address
```

GUI

1. Open **Virtual Machine Manager** either from the **Application Launcher** or by running the `virt-manager` command in your CLI. A new window will open.
2. Select the VM that you want to edit and click the **Open** button. A new window will open.
3. Click the light bulb icon (**Show virtual hardware details**). Then click the **Add Hardware** button in the bottom-left corner. A new window will open.
4. Click **Network** in the menu on the left.
5. In the **Network source** drop-down menu, select **Macvtap device**. The **Device name** field will appear.
6. Type your host interface name into the **Device name** field.
7. Type your host interface MAC address into the **MAC address** field.
8. In the **Device model** drop-down menu, select **virtio**.
9. Click **Finish**.

Click on the newly created device in the left menu of the **Virtual Hardware Details** window and make sure that the **Link state** field shows a selected checkbox next to "active".

XML

Add the following lines to your XML configuration file:

```
<interface type="direct">
.
.
.
<source dev="target-interface" mode="passthrough"/>
<model type="virtio"/>
.
.
```

```
.
</interface>
```

6.4.2 Network Bridges

Network bridges allow multiple virtual machines to communicate using the same physical network card.

A virtual bridge configuration will be dependent on the environment that it needs to work in. VSI recommends following the steps described in the [Configuring and managing networking guide](#) by Red Hat Enterprise Linux on how to configure a network bridge on your Linux machine.

To use a network bridge interface when creating a VM, you can specify the following `--network` option on the `virt-install` command:

```
--network=bridge:bridge-name,mac=bridge-interface-mac,model='virtio-net'
```

To add a network bridge to an existing virtual machine, perform the steps listed in [Adding a Bridge Interface to an Existing KVM Virtual Machine](#).

6.4.2.1 Adding a Bridge Interface to an Existing KVM Virtual Machine

Attention

Some network configuration changes may require TCP/IP, LAN Failover, or VLAN definitions reconfiguration. It is recommended to record the VM NIC configuration before adding, removing, or changing NICs. See [Next Steps](#) for more information.

You can add a bridge interface to an existing virtual machine via CLI, GUI, or by editing the XML configuration.

CLI

Run the following command, replacing `vm-name` with the name of your VM and `bridge-name` with the name of the bridge interface:

```
sudo virsh attach-interface --type bridge --source bridge-name --model e1000e
vm-name
```

GUI

1. Open **Virtual Machine Manager** either from the **Application Launcher** or by running the `virt-manager` command in your CLI. A new window will open.
2. Select the VM that you want to edit and click the **Open** button. A new window will open.
3. Click the light bulb icon (**Show virtual hardware details**). Then, click the **Add Hardware** button in the bottom-left corner. A new window will open.
4. Click **Network** in the menu on the left.
5. In the **Network source** drop-down menu, select **Bridge device**.

6. Type your bridge interface name into the **Device name** field.
7. Type your bridge interface MAC address into the **MAC address** field.
8. In the **Device model** drop-down menu, select the model type that suits your system needs.
9. Click **Finish**.

Click on the newly created device in the left menu of the **Virtual Hardware Details** window and make sure that the **Link state** field shows a selected checkbox next to "active".

XML

Add the following lines to your XML configuration file, specifying the name of the configured bridge interface and your value for the MAC address:

```
<interface type='bridge'>
  <mac address='52:54:00:46:eb:6f' />
  <source network='bridge-name' />
  <model type='e1000e' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0' />
</interface>
```

Note

Make sure that the `bus` and `mac address` values are unique within a single VM configuration.

6.4.2.2 Next Steps

After the network configuration change, you may need to reconfigure TCP/IP, LAN Failover, or VLAN definitions.

1. Boot OpenVMS and issue **MCR LANCP SHOW CONFIGURATION** (or **SHOW CONFIGURATION/USERS**), comparing the adapter types and MAC addresses against the VM NIC configuration.
2. Adjust your applications (TCP/IP, LAN Failover, VLAN definitions) to use the correct network adapters.

For information on how to do so, refer to the [System Manager's Manual Volume 2: Tuning, Monitoring, and Complex Systems](#).

6.5 Advanced Virtio-net Devices Configuration

If you are using virtio-net, it is important that you understand the different types of virtio-net devices.

When you create a virtio-net device, the model type "virtio" will be used by default:

```
<interface type="network">
  <mac address="52:54:00:00:00:00"/>
  <source network="default"/>
  <model type="virtio"/>
  <address type="pci" domain="0x0000" bus="0x06" slot="0x00" function="0x0"/>
</interface>
```

However, the real model is either virtio-transitional or virtio-non-transitional. OpenVMS supports both models, but the default will be chosen by the hypervisor.

- A virtio-transitional device can work with virtio 0.9 or virtio 1.0 guest drivers and only works with PCI controllers. Note that only up to **9** devices of this type can be attached to a virtual machine at any given time.
- A virtio-non-transitional device can only work with virtio 1.0 guest drivers. The libvirt tool will connect the device to PCIe or PCI controller. It allows for more throughput and is therefore the recommended version.

In certain cases, it may be beneficial to explicitly set the model type to "virtio-non-transitional". You can do so by specifying the model type on the `virt-install` command or by editing the XML file.

CLI

```
--network type=direct,source=$networkdev,source_mode=passthrough,
mac=$mac,model=virtio-non-transitional \
```

XML

```
<interface type="network">
  <mac address="52:54:00:00:00:00"/>
  <source network="default"/>
  <model type="virtio-non-transitional"/>
  <address type="pci" domain="0x0000" bus="0x06" slot="0x00" function="0x0"/>
</interface>
```

For more information on the virtio device models, refer to the [Virtio device models](#) article.

6.6 Disabling SELinux File Labeling

You might need to disable the SELinux file labeling in order to use existing virtual disks previously assigned to a different virtual machine.

1. Open your CLI and execute `sudo nano /etc/libvirt/qemu.conf`.
2. Uncomment the line `#remember_owner = 1` by removing the leading hash sign (#) and set the `remember_owner` parameter value to 0.

Your configuration should look like this:

```
# Whether libvirt should remember and restore the original
# ownership over files it is relabeling. Defaults to 1, set
# to 0 to disable the feature.
remember_owner = 0
```

3. For the changes to take effect, execute the following command:

```
sudo systemctl restart libvirtd
```

7. Customizing a KVM Virtual Machine

If you have found a situation where a given setting needs to be edited, but it is not possible to do so from the CLI or Cockpit GUI, you can edit the XML configuration file in one of the two ways described below.

Editing XML via Virtual Machine Manager GUI

Perform the following steps to edit your XML KVM virtual machine configuration using the **Virtual Machine Manager** GUI.

1. Open the **Virtual Machine Manager** either from the **Application Launcher** or by running the `virt-manager` command in your CLI. A new window will open.
2. In the top menu of the **Virtual Machine Manager** window, click **Edit > Preferences**, then select the checkbox **Enable XML editing**.
3. Click **Close**.
4. Open the virtual machine that you wish to edit by selecting the VM and clicking the **Open** button. A new window will open.
5. Click the light bulb icon (**Show virtual hardware details**). Then, in the **Overview** section, click the **XML** tab. You can now edit the XML file.
6. After you have made the necessary changes, click **Apply**.
7. Once the XML file is saved, the changed settings will be shown in the **Virtual Machine Manager** GUI. However, you must reboot the virtual machine for the changes to take effect.

Editing XML via a Linux Text Editor

Warning

Do not attempt to edit the XML configuration file of your virtual machine directly, run the commands specified below instead.

By default, the Vim editor will be used if you run the `virsh edit` command to edit the XML file.

To use a different editor, you must define it as `EDITOR`. For example, to use the nano editor, run:

```
sudo EDITOR=/usr/bin/nano virsh edit vm-name
```

After you have edited the XML file, you must reboot the virtual machine for the changes to take effect.

8. Advanced Configuration

8.1 Secure Boot

8.1.1 Secure Boot Overview

Starting with VSI OpenVMS E9.2-4, the Secure Boot feature is supported via the First Stage Bootloader.

The OpenVMS Boot Manager is signed with an OpenVMS-specific certificate and is validated by the First Stage Bootloader before it is loaded. To establish trust between the First Stage Bootloader and OpenVMS, the OpenVMS signing certificate must be added to the First Stage Bootloader's Machine Owner Key (MOK) database as described in [Adding OpenVMS Certificate to the Machine Owner Key Database](#).

Once the certificate is registered and the Secure Boot option is enabled, OpenVMS can boot successfully while maintaining the Secure Boot chain of trust.

Use the following command to see whether Secure Boot is enabled on your system:

```
$ WRITE SYS$OUTPUT F$GETSYI("SECURE_BOOT")
```

Output	Description
0	Disabled
1	Enabled

For more information, see [VSI OpenVMS Guide to System Security](#).

8.1.2 Secure Boot in KVM Configuration

You can enable or disable the Secure Boot option when running the command to create a new virtual machine or after the virtual machine has been created.

When Creating a VM

Enable Secure Boot

In order to create a KVM virtual machine with Secure Boot **enabled** by default, use the following `--boot` parameters on the `virt-install` command when setting up your VM:

```
--boot loader=/usr/share/OVMF/OVMF_CODE.secboot.fd, \
loader.readonly=yes,loader.secure=yes,loader.type=pflash, \
nvram.template=/usr/share/OVMF/OVMF_VARS.secboot.fd \
```

Disable Secure Boot

In order to create a KVM virtual machine with Secure Boot **disabled** by default, use the following `--boot` parameters on the `virt-install` command when setting up your VM:

```
--boot loader=/usr/share/OVMF/OVMF_CODE.secboot.fd, \
loader.readonly=yes,loader.secure=no,loader.type=pflash, \
nvram.template=/usr/share/OVMF/OVMF_VARS.fd \
```

Existing VM

You can enable Secure Boot on an existing virtual machine by editing its XML configuration. To disable Secure Boot, follow the instructions in the [Secure Boot in UEFI](#) section.

1. Run the following command to use the nano editor:

```
sudo EDITOR=/usr/bin/nano virsh edit vm-name
```

2. Edit the `os` element as follows, specifying the name of your VM:

```
<os>
  <type arch="x86_64" machine="q35">hvm</type>
  <firmware>
    <feature enabled='yes' name='enrolled-keys'/>
    <feature enabled='yes' name='secure-boot'/>
  </firmware>
  <loader readonly='yes' secure='yes' type='pflash' format='raw'>
    /usr/share/OVMF/OVMF_CODE.secboot.fd</loader>
  <nvram template='/usr/share/OVMF/OVMF_VARS.secboot.fd'
    templateFormat='raw' format='raw'>
    /var/lib/libvirt/qemu/nvram/vm-name_VARS.fd</nvram>
```

```

.
.
.
</os>

```

Make sure that the `enrolled-keys` and `secure-boot` features are enabled as shown above.

3. Reset the NVRAM configuration by running the command below.

Caution

Resetting NVRAM will remove all VM-specific UEFI settings, including boot options and any certificates or keys enrolled after NVRAM was created.

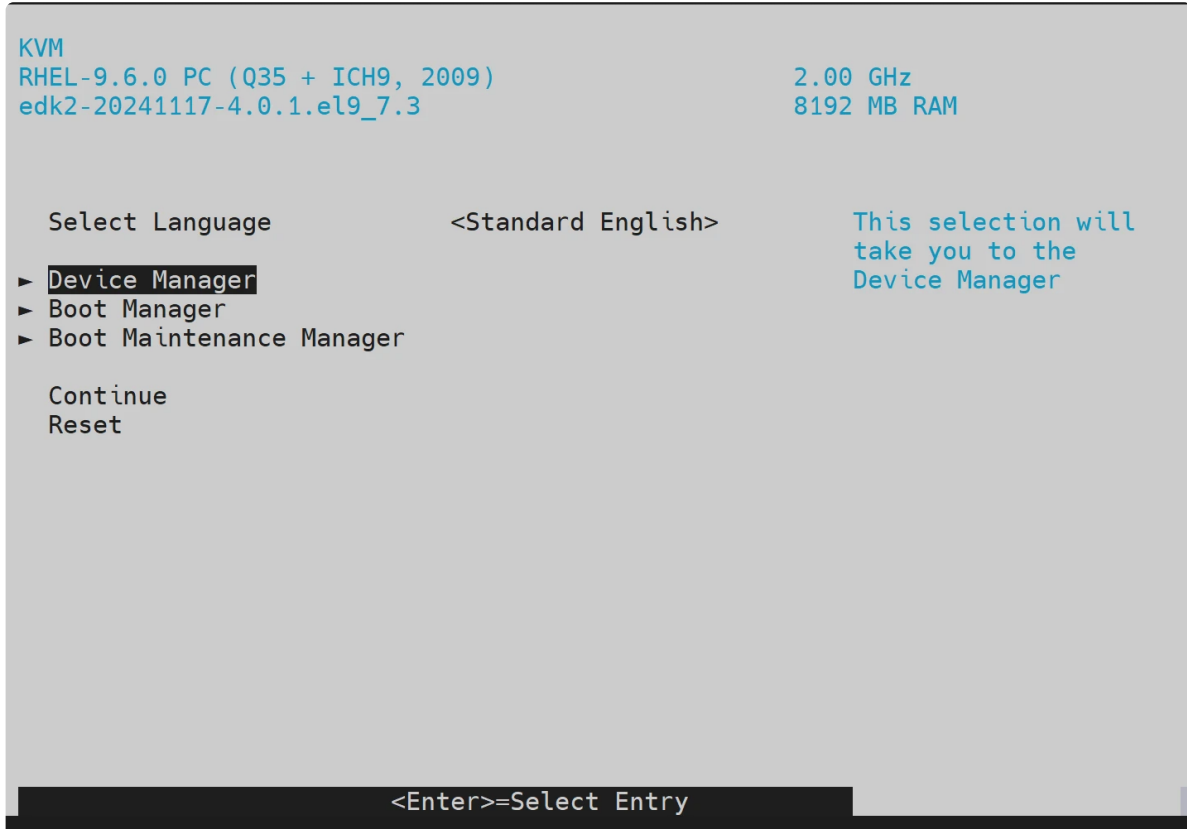
```
sudo virsh start vm-name --reset-nvram
```

After you have configured the Secure Boot option, you can enable/disable it in the UEFI menu.

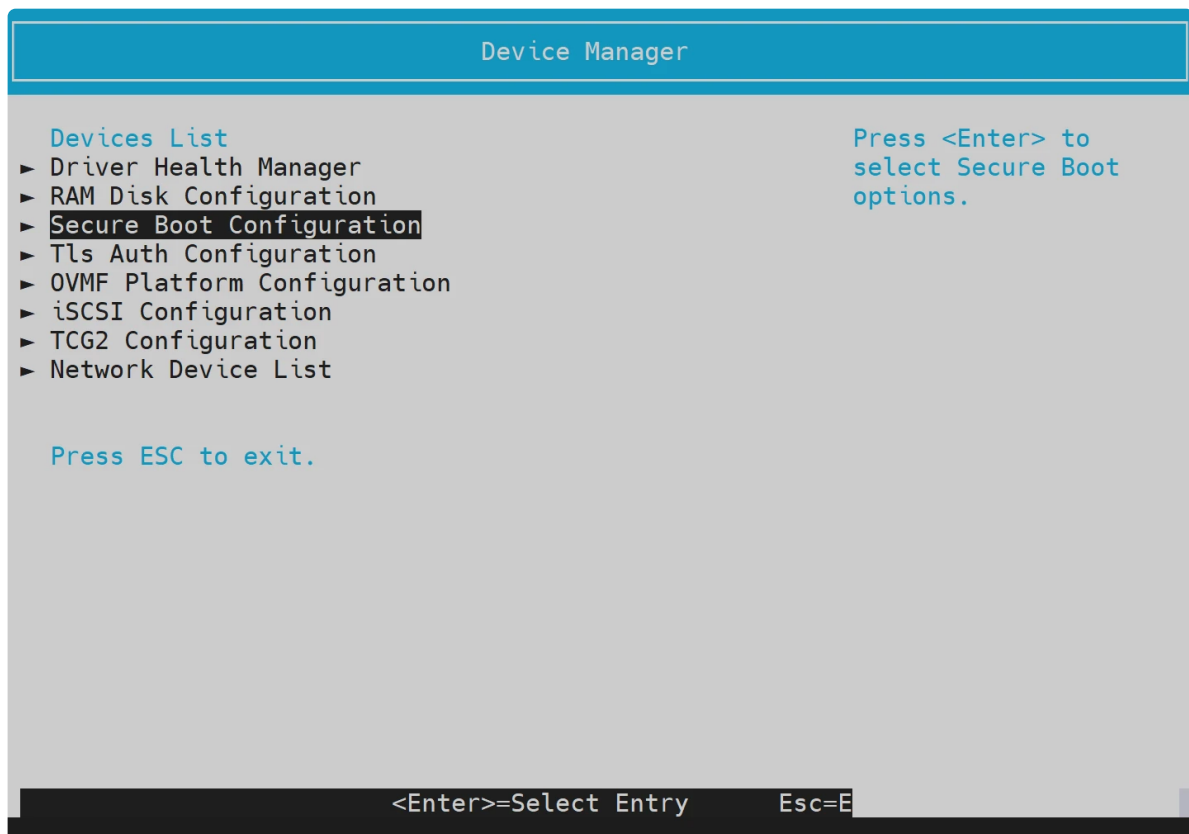
8.1.2.1 Secure Boot in UEFI

Follow the steps below to enable or disable Secure Boot in UEFI.

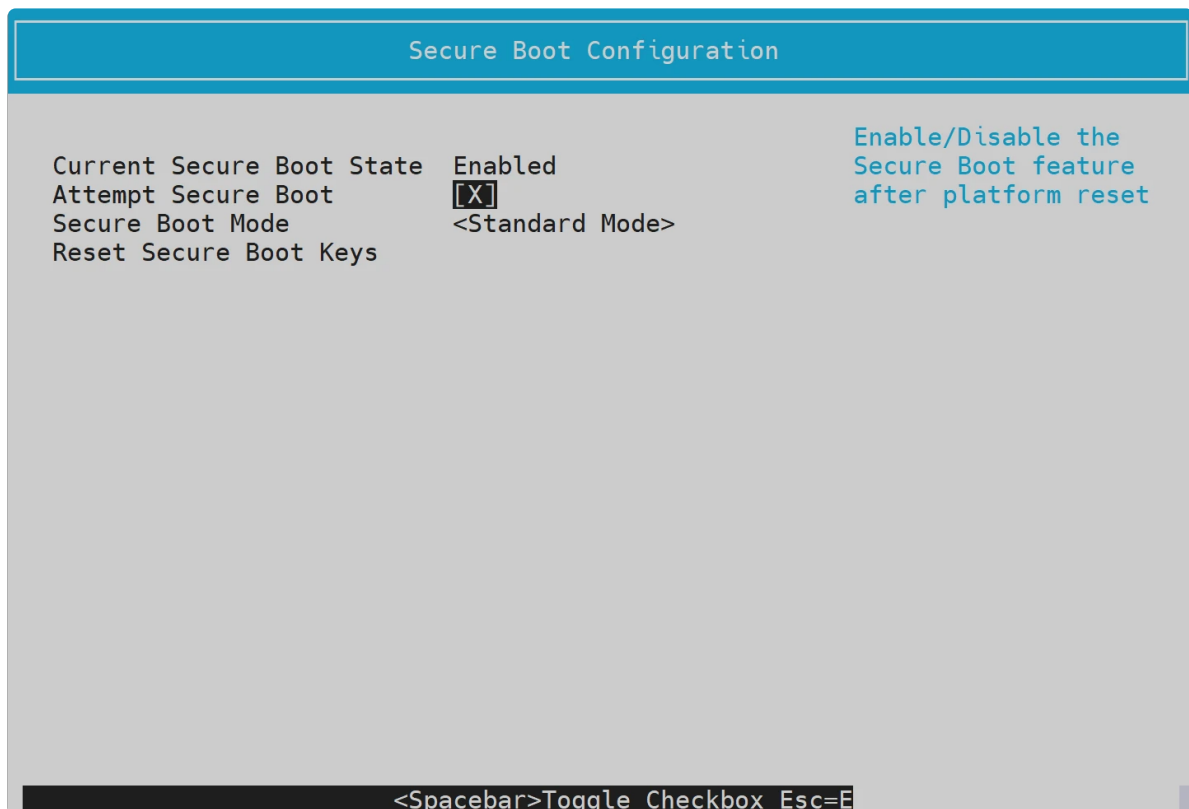
1. Start your KVM virtual machine and enter **EXIT** at the `BOOTMGR>` prompt.
2. Press `[Esc]`. The UEFI screen will appear.



3. Select **Device Manager** and then **Secure Boot Configuration**.



4. Select the **Attempt Secure Boot** option and press to toggle the enabled/disabled state. The "X" symbol enables the Secure Boot option; empty square brackets ("[]") disable the Secure Boot option.



5. Press to exit the UEFI menu.

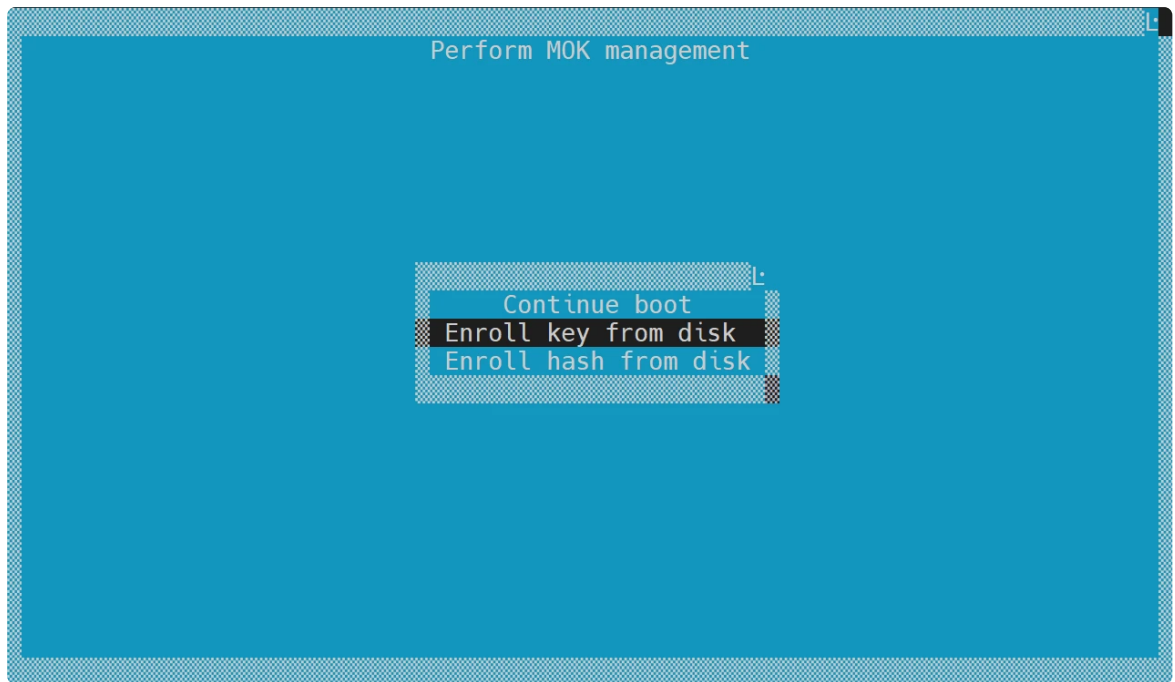
8.1.3 Adding OpenVMS Certificate to the Machine Owner Key Database

In order to boot VSI OpenVMS with Secure Boot enabled, you must register the OpenVMS key in the Machine Owner Key (MOK) database.

There are two methods to do so: manual and automatic. For automatic registration, ensure that OpenVMS has been booted at least once before enabling Secure Boot.

Manual registration

1. Enable Secure Boot as described in the [Secure Boot in KVM Configuration](#) section and start your virtual machine.
2. The UEFI key management utility will load. You will see a countdown at the bottom of the screen. Press before it runs out and the system starts booting.
3. Use the arrow keys to select **Enroll key from disk** and press .



4. Locate the **VMS_CERTIF.DER** file at the following path: *your-disk*/EFI/VMS/VMS_CERTIF.DER.



5. Select **Continue**.
6. At the **Enroll the key(s)?** prompt, select **Yes**.
7. Select **Reboot**.
8. Now you will be able to boot to the OpenVMS Boot Manager with Secure Boot.

Automatic registration

The OpenVMS Boot Manager automatically registers the certificate when the First Stage Bootloader is present.

1. Boot OpenVMS with Secure Boot disabled so that the certificate is registered automatically.
2. Shut down OpenVMS.
3. Enable Secure Boot in your VM configuration and UEFI as described in the [Secure Boot in KVM Configuration](#) section.
4. You will now be able to boot to the VSI Boot Manager with Secure Boot enabled.

8.2 Configuring Fibre Channel PCI Passthrough on KVM Virtual Machine

8.2.1 Fibre Channel PCI Passthrough on KVM Virtual Machine Overview

This section describes the configuration steps necessary for setting up Fibre Channel PCI passthrough for an OpenVMS instance running on a KVM virtual machine. This allows OpenVMS to directly access and utilize a physical Fibre Channel Host Bus Adapter (HBA) connected to the host.

Passthrough technology uses an input-output memory management unit (IOMMU) which can be referred to as VT-d on Intel-based hosts and AMD-Vi, AMD-V, or SVM on AMD-based hosts. Throughout this guide, the generic term IOMMU is used, but be aware that the terminology on your system may differ depending on your CPU.

Important

- PCI passthrough is based on PCI slot granularity. That is, all functions of a multi-function PCI device must be assigned to the same virtual machine. For example, a dual-port Fibre Channel HBA must have both ports (i.e. functions) assigned to the same virtual machine. Individual ports on the same HBA cannot be assigned to more than one virtual machine.
- If more than one virtual machine needs to have control over a Fibre Channel HBA, there must be more than one Fibre Channel HBA installed on the host system.
- N_Port ID Virtualization (NPIV) is not supported on VSI OpenVMS.

8.2.1.1 Supported Platforms and Hardware

Operating systems

- Red Hat Enterprise Linux (RHEL) 9.x
- Oracle Linux (OL) 9.x

Host Bus Adapters

Refer to the [Supported Platforms](#) page for the full list of supported HBA devices.

VSI OpenVMS Version

VSI OpenVMS V9.2-3 + [Update V3](#) with the [PERF_UPD V1.0 ECO kit](#) installed, or any newer version of VSI OpenVMS x86-64.

KVM Version

PCI passthrough has been tested on systems running KVM version 9.1.0 and 10.1.0.

Linux Kernel Version

Linux kernel version 5.2 or higher

8.2.1.2 Prerequisites

Before you start setting up PCI passthrough, you must make sure that the IOMMU support is enabled on your host system both on the UEFI/BIOS side and in the Linux kernel. In order to do so, follow the steps described in the next sections.

Info

Note that the commands from the examples in the next sections may require root privileges to run. Execute `sudo -s` to be able to run all commands without specifying `sudo`.

8.2.2 IOMMU Hardware Extensions in UEFI/BIOS

First, you must check whether the IOMMU extensions are enabled in the UEFI/BIOS.

1. Run the `lscpu | grep -i virtualization` command to determine whether virtualization is enabled on your system.

- On an Intel-based host system, the output will look similar to this:

```
Virtualization:          VT-x
```

- On an AMD-based host system, the output will look similar to this:

```
Virtualization:          AMD-V
```

If the `lscpu | grep -i virtualization` command does not produce any output, the virtualization is disabled on your system. Proceed to [Enabling IOMMU Hardware Extensions in UEFI/BIOS](#) for next steps.

2. Run the `dmesg | grep -i IOMMU` command to check whether IOMMU is set to the correct operation mode.

- If the command returns `Default domain type: Passthrough`, proceed to the [IOMMU Support in Linux Kernel](#) section to determine whether the IOMMU support is enabled in the Linux kernel.
- If the command returns `Default domain type: Translated`, then you need to change the operation mode to `Passthrough` as described in the [Enabling IOMMU in Linux Kernel](#) section.

8.2.2.1 Enabling IOMMU Hardware Extensions in UEFI/BIOS

In order to enable IOMMU support in the UEFI/BIOS, you will need either iLO/iDRAC or physical access to the server.

1. Access your system UEFI/BIOS settings.
2. Refer to the documentation for your system to navigate to the virtualization menu.
3. In the virtualization menu, find the IOMMU setting specific to your system¹ and select **Enabled**.
4. Save your settings.

The IOMMU support is now enabled in the UEFI/BIOS.

¹ Depending on your system, the name of the IOMMU setting may be one of the following: Intel (R) VT-d, AMD (R) IOMMU, AMD I/O Virtualization Technology, etc. If you have trouble finding the appropriate setting, refer to the documentation specific to your system.

⚠ Important

On all AMD-based systems and Intel-based systems running OL, enabling the IOMMU support in the UEFI/BIOS will automatically enable it in the Linux kernel, and you can proceed with configuring a Fibre Channel PCI passthrough, as described in the section called [Assigning Fibre Channel HBA to Virtual Machine](#).

On Intel-based systems running RHEL, you must manually enable the IOMMU support in the Linux kernel. Refer to the section called [IOMMU Support in Linux Kernel](#) for next steps.

8.2.3 IOMMU Support in Linux Kernel

In addition to enabling the IOMMU hardware extensions in the UEFI/BIOS, the IOMMU support must be enabled in the host kernel.

Check whether the IOMMU support is enabled in the Linux kernel by running the `virt-host-validate` command:

If enabled

If the IOMMU support is enabled, the command output will look similar to this:

```
.
.
.
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
.
.
.
```

If disabled

If the IOMMU support is disabled, the command output will look similar to this:

```
.
.
.
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.
.
.
.
```

8.2.3.1 Enabling IOMMU in Linux Kernel

Follow the steps below to enable the IOMMU support in the Linux kernel.

1. Regenerate the GRUB configuration with the `intel_iommu=on` and `iommu=pt` parameters using one of the following commands:

- On Intel-based hosts, run:

```
sudo grubby --args="intel_iommu=on iommu=pt" --update-kernel DEFAULT
```

- On AMD-based hosts, run:

```
sudo grubby --args="iommu=pt" --update-kernel DEFAULT
```

2. Reboot the system for the changes to take effect.

8.2.4 Assigning Fibre Channel HBA to Virtual Machine

First, identify the appropriate Fibre Channel HBA to use for PCI passthrough using CLI. Then, you can use the Virtual Machine Manager GUI or Linux Cockpit to assign the device to your virtual machine.

For the full list of supported Fibre Channel HBA devices, refer to the [Supported Platforms](#) page.

1. Create a text file. For example:

```
nano iommu_groups.sh
```

2. Copy and paste the following script¹ to list all PCI/PCIe cards that can be used for passthrough into your newly created file:

```
#!/bin/bash
shopt -s nullglob
for g in $(find /sys/kernel/iommu_groups/* -maxdepth 0 -type d | sort -V); do
    echo "IOMMU Group ${g##*/}:"
    for d in $g/devices/*; do
        echo -e "\t$(lspci -nns ${d##*/})"
    done;
done;
```

3. Save the changes and exit the editor.
4. Add execute permissions to the file containing the script by running the following command:

```
chmod +x iommu_groups.sh
```

5. Run the script:

```
./iommu_groups.sh
```

Your output will look similar to this:

```
.
.
.
IOMMU Group 8:
    11:00.0 Fibre Channel [0c04]: QLogic Corp. ISP2722-based 16/32Gb Fibre
Channel to PCIe Adapter [1077:2261] (rev 01)
    11:00.1 Fibre Channel [0c04]: QLogic Corp. ISP2722-based 16/32Gb Fibre
Channel to PCIe Adapter [1077:2261] (rev 01)
IOMMU Group 9:
    12:00.0 Fibre Channel [0c04]: QLogic Corp. ISP2722-based 16/32Gb Fibre
Channel to PCIe Adapter [1077:2261] (rev 01)
    12:00.1 Fibre Channel [0c04]: QLogic Corp. ISP2722-based 16/32Gb Fibre
```

1. This script has been sourced from the following webpage: https://wiki.archlinux.org/title/PCI_passthrough_via_OVMF#Ensuring_that_the_groups_are_valid

```
Channel to PCIe Adapter [1077:2261] (rev 01)
```

```
.
```

```
.
```

```
.
```

6. In your output, find a group that contains a Fibre Channel HBA that you want to pass through to the virtual machine. The two entries in a group represent PCI addresses and functions that you will need when assigning the Fibre Channel HBA to your VM.
7. Use one of the methods below to assign a Fibre Channel HBA to your virtual machine.

Warning

Make sure to shut down your virtual machine before you assign an HBA device to it.

Virtual Machine Manager

1. Open **Virtual Machine Manager** either from the **Application Launcher** or by running the `virt-manager` command in your CLI. A new window will open.
2. Select the VM that you want to edit and click the **Open** button. A new window will open.
3. Click the light bulb icon (**Show virtual hardware details**). Then, click the **Add Hardware** button in the bottom-left corner. A new window will open.
4. In the menu on the left, select **PCI Host Device**.
5. From the list on the right, you must select and add both entries from the group that you found in [this step](#). Select the first entry and click **Finish**. You will be redirected to the **Virtual Hardware Details** window.
6. Click the **Add Hardware** button in the bottom-left corner once again. A new window will open.
7. From the list on the right, find the second entry for the device you are assigning to your VM. Select the entry and click **Finish**.

You should now be able to see both entries listed on the right pane of the **Virtual Hardware Details** window.

Cockpit GUI

Using Linux Cockpit to assign PCI devices to a virtual machine is the easiest and quickest method. Linux Cockpit performs all of the host operating system interactions required for PCI passthrough.

1. If Linux Cockpit is not installed on your system by default, refer to the steps described in the [Introduction to Cockpit](#) article on how to install and enable Linux Cockpit (for both RHEL and OL systems).
2. Log in to Linux Cockpit by going to `your-linux-system-ip-address:9090` from your browser.
3. Enter your credentials and click **Log in**.
4. In the menu on the left, select **Virtual Machines**.
5. Click on the name of the VM to which you want to assign a Fibre Channel HBA. You will see your virtual machine configuration details.

6. Scroll down to the Host devices section and click **Add host device**. A pop-up window will open.
7. Select **PCI** as the device type to list all the PCI devices available for PCI passthrough.
8. Select both PCI device entries that you found using [this script](#). Click **Add**.

⚡ Caution

If you do not assign both functions of a multi-function PCI device to the VM, the host system may crash.

9. Click **Add** to add both functions of the Fibre Channel HBA device.
10. You should now see both functions of the device assigned to your virtual machine on the VM configuration page.

XML

1. Create two empty XML files on the host. For example:

```
nano pt-dev-conf-1.xml
nano pt-dev-conf-2.xml
```

2. Copy and paste the following XML configuration into each of the newly created files:

```
<hostdev mode='subsystem' type='pci' managed='yes'>
<driver name='vfio' />
<source>
  <address domain='0x0000' bus='0x00' slot='0x00' function='0x0' />
</source>
</hostdev>
```

3. Edit the files so that the values in the line
`<address domain='0x0000' bus='0x00' slot='0x00' function='0x0' />` correspond to each of the entries in the group containing the HBA device you are adding.

For example, if you are adding the following device:

```
11:00.0 Fibre Channel [0c04]: QLogic Corp. ISP2722-based 16/32Gb Fibre
Channel to PCIe Adapter [1077:2261] (rev 01)
11:00.1 Fibre Channel [0c04]: QLogic Corp. ISP2722-based 16/32Gb Fibre
Channel to PCIe Adapter [1077:2261] (rev 01)
```

For the first entry, the XML configuration will look like this:

```
<hostdev mode='subsystem' type='pci' managed='yes'>
<driver name='vfio' />
<source>
  <address domain='0x0000' bus='0x11' slot='0x00' function='0x0' />
</source>
</hostdev>
```

For the second entry, it will look like this:

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <driver name='vfi0' />
  <source>
    <address domain='0x0000' bus='0x11' slot='0x00' function='0x1' />
  </source>
</hostdev>
```

Important

The two configurations of the Fibre Channel HBA device must be saved in separate files.

4. Save the changes to the XML configuration files.
5. Run the following command to add the first Fibre Channel HBA configuration to your virtual machine configuration file:

```
virsh attach-device --file --persistent pt-dev-conf-1.xml
```

6. Run the following command to add the second Fibre Channel HBA configuration to your virtual machine configuration file:

```
virsh attach-device --file --persistent pt-dev-conf-2.xml
```

Note

After you have added both Fibre Channel HBA configurations, you may safely delete the XML files that you created.

Tip

You can assign a Fibre Channel HBA device to your virtual machine during installation if you use the `--host-device` argument on the `virt-install` command. Specify the PCI addresses of the appropriate devices that can be found using this script. For example:

```
--host-device 11:00.0 --host-device 11:00.1
```

The number of the arguments must be equal to the number of entries in the corresponding group.

8.2.4.1 Verification

You can verify that the Fibre Channel HBA device has been assigned to the virtual machine by opening the System Dump Analyzer (SDA) and executing **CLUE CONFIG/ADAPTER** in OpenVMS running on your VM:

```
$ ANALYZE/SYSTEM
```

```
OpenVMS system analyzer

SDA> CLUE CONFIG/ADAPTER
TR Adapter    ...  Hose Bus      ...  Node G SIN   iVec    SCB  Port Slot Device Name / HW-Id
-----
.
.
.
10 PCIE      ...    0 07 PCIE  ...    700 0017    00D0    1500 FGA:    0 QLogic ISP2722 PCIe Fibre
11 PCIE      ...    0 08 PCIE  ...    800 0016    00CF    14F0 FGB:    0 QLogic ISP2722 PCIe Fibre
```

Alternatively, it is possible to determine whether the Fibre Channel HBA is assigned to the virtual machine by the presence of the Fibre Channel FG devices as follows:

```
$ SHOW DEVICES FG
Device          Device          Error
Name            Status          Count
FGA0:           Online          0
FGB0:           Online          0
```

If not yet configured, you can now create storage devices on the storage controller and map them to the Fibre Channel HBA assigned to the virtual machine.

For more information on mapping tape devices, refer to [Guidelines for OpenVMS Cluster Configurations](#).

For instructions on how to map Fibre Channel volumes to OpenVMS guests, refer to your storage array documentation and support.

8.3 Booting OpenVMS from a Fibre Channel Disk

Perform the procedures described in this section and configure UEFI to allow booting from a Fibre Channel disk.

1. At the BOOTMGR> prompt, execute the **EXIT** command.

This will cause the UEFI firmware to reset. Interrupt the normal firmware startup by hitting Esc many times until the UEFI firmware menu appears. If the Boot Manager appears, repeat the procedure.

```
VSI OpenVMS Boot Manager: V9.2-* Build 160

ENABLED:  Symmetric Multi-Processing
ENABLED:  Crash Dump Processing
ENABLED:  Legacy Serial Port Console for remote COM 1
ENABLED:  Using Environment ROM

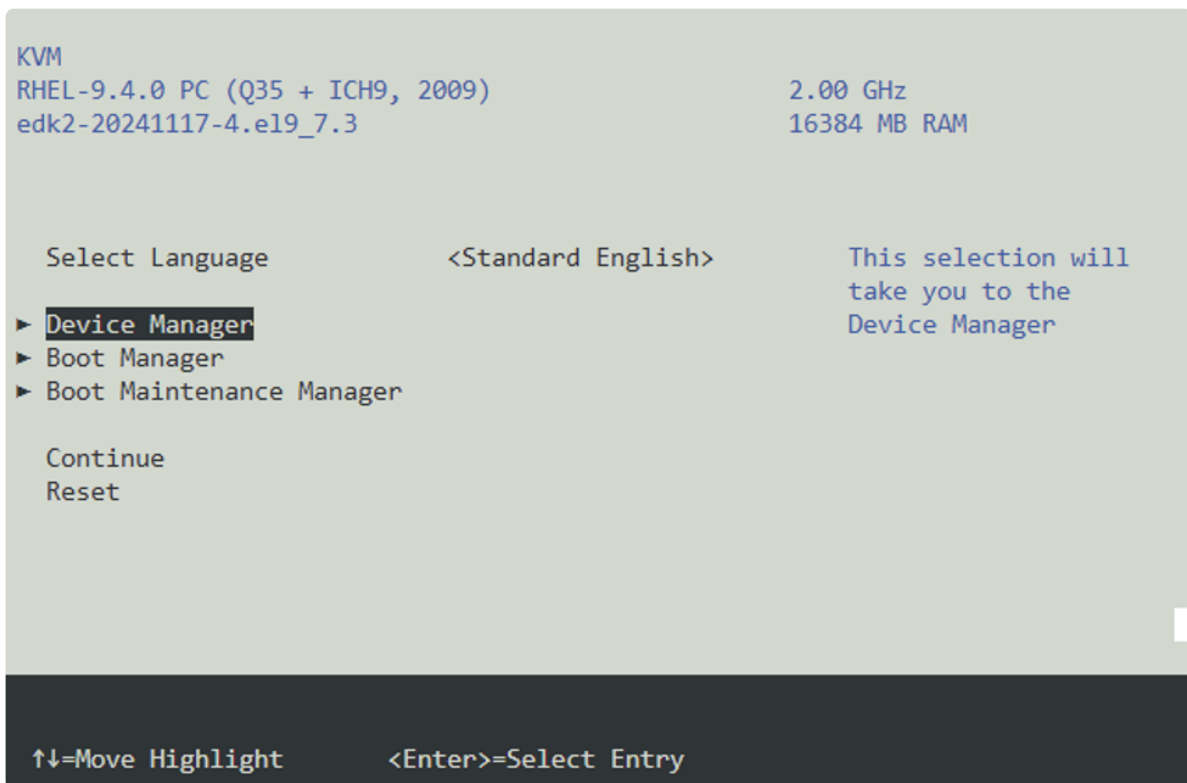
Checking Required Processor Features:  PASSED
Identifying devices. This may take a while.  COMPLETED

VIRTUAL MACHINE GUEST:
KVM (tm) No Mouse support; Use Command or Arrow Keys

BOOT MANAGER DEVICE:  DKA0
DEFAULT BOOT COMMAND: BOOT $1$DGA210 0x00 0x00000001

BOOTMGR> EXIT
```

2. The UEFI menu will appear. Use the arrow keys to highlight the **Device Manager** and press Enter.



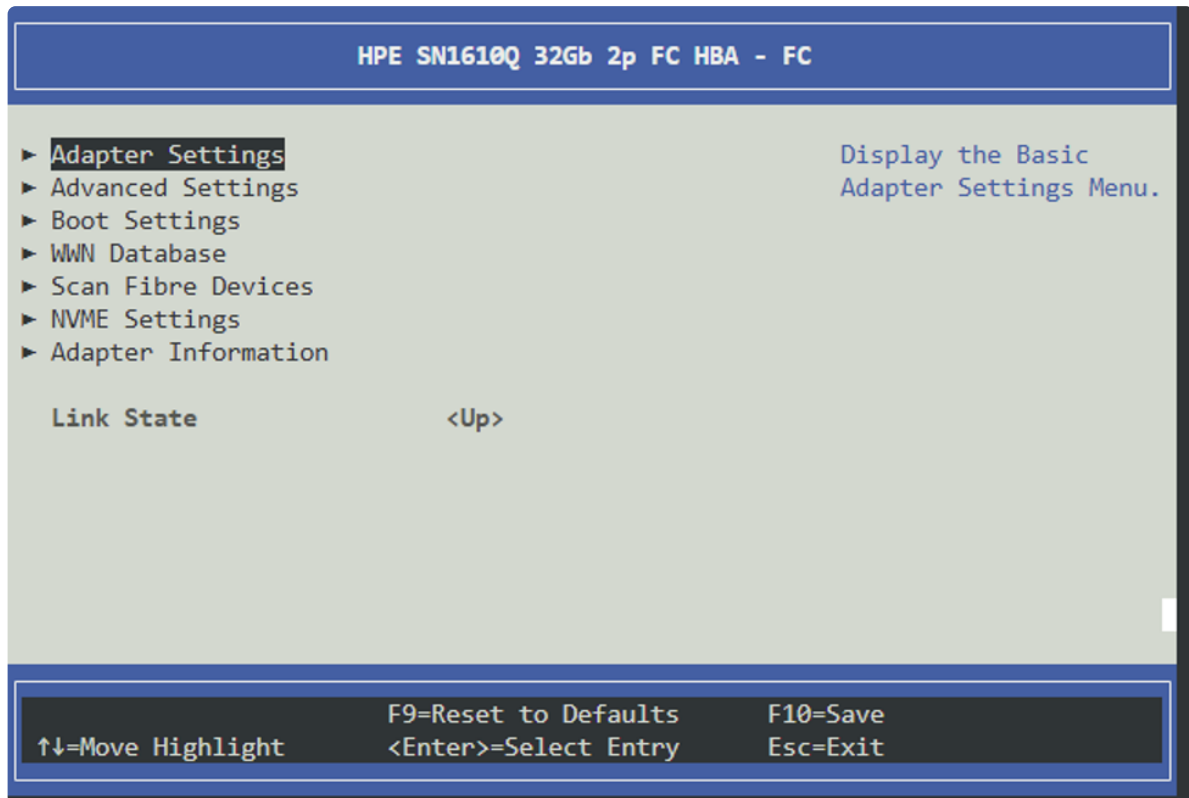
3. You will see the UEFI Device Manager screen. Use the arrow keys to highlight the first QLogic adapter in the menu and press . Note that depending on your environment, the adapter model may be different.



4. The UEFI QLogic Setup Utilities home screen will appear.

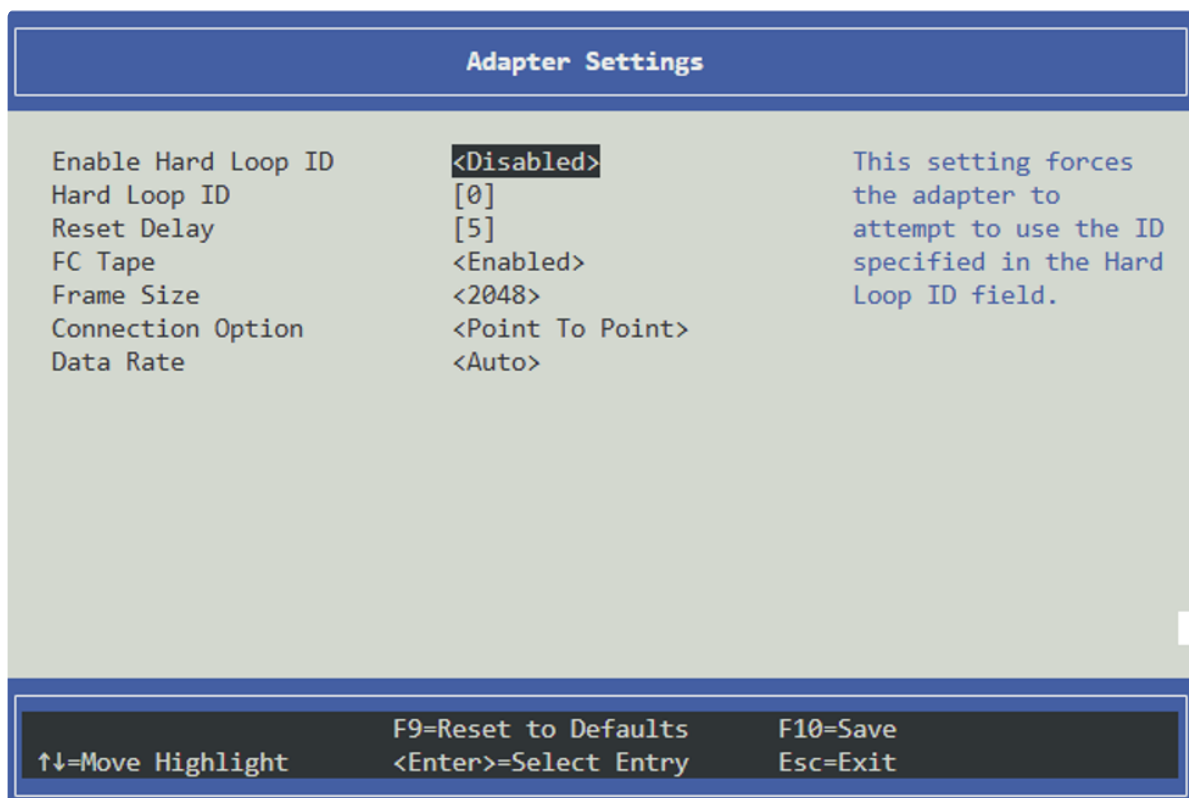
Note the **Link State** status. If the status is not <Up>, there are problems with cabling and possibly the Fibre Channel switch.

Select **Adapter Settings**.



5. Use the arrow keys to navigate and change the adapter settings as indicated below.

Since OpenVMS does not support the **Arbitrated Loop** mode, it is critical that the **Connection Option** be set to **<Point To Point>** mode.



6. Press **Esc** to return to the QLogic Setup Utilities home screen. Select **Boot Settings** and press **Enter**.
7. Set the **Boot Settings** as indicated below. These settings are critical to enable boot support. Make sure to set all fields exactly as shown below.

Boot Settings		
Selective Login	<Disabled>	Specifies that the driver is to use the WWN Database as a list of devices that the adapter is permitted to login. Enable this option to limit the adapter device discovery to devices matching those in the WWN Database.
Selective Lun Login	<Disabled>	
FCScanLevel Variable	<1>	
Fabric Assigned Boot LUN	<Disabled>	
World Login	<Enabled>	
Adapter Driver	<Enabled>	
↑↓=Move Highlight F9=Reset to Defaults F10=Save <Enter>=Select Entry Esc=Exit		

- Press **Esc** to return to the QLogic Setup Utilities home screen.
8. Optional: Open the **WWN Database** setup menu to configure up to four specific Logical Unit Numbers (LUNs). It is used only when **FCScanLevel** is set to **0**.

WWN Database		
Drive 0 WWPN	<0000000000000000:~>	Selective Login WWPN. Suffix '+' implies target is online, '-' implies target is offline.
Drive 0 LUN	<0000:~>	
Drive 1 WWPN	<0000000000000000:~>	
Drive 1 LUN	<0000:~>	
Drive 2 WWPN	[0]	
Drive 2 LUN	[0]	
Drive 3 WWPN	[0]	
Drive 3 LUN	[0]	

↑↓=Move Highlight	F9=Reset to Defaults <Enter>=Select Entry	F10=Save Esc=Exit
-------------------	--	----------------------

Press **Esc** to return to the QLogic Setup Utilities home screen.

9. Select **Scan Fibre Devices** to scan for new Fibre Devices on the Storage Area Network (SAN). Once the devices have been scanned, they can be added to the WWN Database.

Scan Fibre Devices	
Fibre Channel device scan is complete.	
Press ESC to continue.	

↑↓=Move Highlight	Esc=Exit
-------------------	----------

Press **Esc** to return to the QLogic Setup Utilities home screen.

10. Open the **NVME Settings** menu and make sure the **FC NVME** parameter is set to **<Disabled>**. VSI OpenVMS does not support Non-Volatile Memory Express (NVMe).

NVME Settings		
FC NVME	<Disabled>	Enable FC NVME storage support.
Host NQN	nqn.2014-08.org.nvmexpress:uuid:44310fb8-2d0c-4c62-b686-a2e6c9f2ee2f	
Host ID	B80F31440C2D624CB686A2E6C9F2EE2F	
Storage 0	<Disabled>	
Storage 0 WWPN	[0]	
Storage 0 WWNN	[0]	
Storage 0 NQN	[0]	
Storage 0 Controller ID	[0]	
Storage 0 Namespace ID	[0]	
Storage 1	<Disabled>	

↑↓=Move Highlight F9=Reset to Defaults F10=Save
 <Enter>=Select Entry Esc=Exit

Press **Esc** to return to the QLogic Setup Utilities home screen.

11. Press **F10** to save your changes.

Each Fibre Channel port is configured separately. On a multiport adapter, return to the **Device Manager** menu by pressing **Esc** and repeat the setup process for the next port in the list. Make sure all settings are saved back to NVRAM.

12. After you have configured all the ports, press **Esc** until the UEFI firmware is restarted and you see the BOOTMGR> prompt again.
13. At the BOOTMGR> prompt, execute **DEVICE** and find the name of the Fibre Channel disk you wish to boot OpenVMS from. Then issue **BOOT device-name**.

8.4 Mapping Storage Controllers to Disk Devices

8.4.1 Preparing to Map Storage Controllers to Disk Devices

This section provides steps on mapping your virtual disk devices to SATA and Virtio-SCSI storage controllers for use with VSI OpenVMS on a KVM virtual machine.

Important

- On KVM, OpenVMS supports only SATA and Virtio-SCSI controllers.
- Starting with E9.2-4, VSI OpenVMS supports both SATA and Virtio-SCSI virtual CD/DVD-ROM devices.
- KVM supports a maximum of six disk devices per one SATA controller and a maximum of 255 devices per one Virtio-SCSI controller. VSI does not recommend mapping more than six disk devices to either a SATA or a Virtio-SCSI storage controller.
- SCSI devices with a LUN number higher than 7 (or more than 8 devices) are not supported by the OVMF Virtio-SCSI driver and will not appear as bootable devices in the Boot Manager.

Make sure that your VM is shut down before you assign storage controllers to your virtual disk devices.

To map SATA devices, refer to the [SATA Controllers](#) section.

To map SCSI devices, refer to [Virtio-SCSI Controllers](#) section.

8.4.2 SATA Controllers

Perform the following steps to map your SATA disk device to a SATA controller.

1. Open **Virtual Machine Manager** either from the **Application Launcher** or by running the `virt-manager` command in your CLI. A new window will open.
2. In the top menu of the **Virtual Machine Manager** window, click **Edit > Preferences**, then select the checkbox **Enable XML editing**.
Click **Close**.
3. Open your virtual machine configuration window by selecting the VM and clicking the **Open** button. A new window will open.
4. Click the light bulb icon (**Show virtual hardware details**). From the list on the left, select the SATA disk device to which you want to map a SATA controller.
5. In the right pane, open the XML tab and find the following line:

```
<address type="drive" controller="0" bus="0" target="0" unit="0"/>
```

Edit the line so that the value of the `controller` attribute is equal to the amount of SATA controllers that already exist in your VM configuration. If there are no SATA controllers in your configuration, specify 0 as the controller value.

Click **Apply**. A new SATA controller will appear in the left pane.

Attention

If the new controller did not appear, choose a higher number for the controller attribute.

6. Click on the newly created SATA controller, then open the **XML** tab in the right pane. Your configuration should look similar to this:

```
<controller type="sata" index="0">
  <address type="pci" domain="0x0000" bus="0x10" slot="0x03" function="0x0"/>
</controller>
```

The index attribute value is the controller number. In order for the controller to be mapped to the disk device, the values in both the controller and the disk device configuration must be the same. For example:

- Controller XML configuration:

```
<controller type="sata" index="1">
  <address type="pci" domain="0x0000" bus="0x10" slot="0x03" function="0x0"/>
</controller>
```

- Disk device XML configuration:

```
<disk type="file" device="disk">
  <driver name="qemu" type="qcow2" cache="writethrough"/>
  <source file="sata_disk.qcow2"/>
  <target dev="sdd" bus="sata"/>
  <address type="drive" controller="1" bus="0" target="0" unit="3"/>
</disk>
```

7. You can check whether the disk device has been properly mapped to the controller by selecting the SATA controller in the left pane of the **Virtual Hardware Details** window and opening the **Details** tab.

The **Devices** field will list the devices that are mapped to the selected controller.

8.4.3 Virtio-SCSI Controllers

Perform the following steps to map your virtual disk device to a Virtio-SCSI controller.

1. Open **Virtual Machine Manager** either from the **Application Launcher** or by running the `virt-manager` command in your CLI. A new window will open.
2. In the top menu of the **Virtual Machine Manager** window, click **Edit > Preferences**, then select the checkbox **Enable XML editing**.
Click **Close**.
3. Open your virtual machine configuration window by selecting the VM and clicking the **Open** button. A new window will open.
4. Click the light bulb icon (**Show virtual hardware details**).
 - If you already have a Virtio-SCSI controller that you want to map to your virtual disk device, proceed to [Step 5](#).
 - If you want to create a new Virtio-SCSI controller, perform the following substeps:
 - a. Click the **Add Hardware** button at the bottom of the left pane. A new window will open.
 - b. In the left pane, select **Controller**. In the right pane, make sure that the **Type** is specified as **SCSI** and **Model** is **VirtIO SCSI**. If necessary, open the drop-down menus and select the specified type and model. Click **Finish**.
 - c. A new Virtio-SCSI controller will appear in the pane on the left.
5. In the left pane, click on the Virtio-SCSI controller that you want to map to your SCSI disk device. Open the **XML** tab in the right pane.

Your configuration will look similar to this:

```
<controller type="scsi" index="4" model="virtio-scsi">
  <address type="pci" domain="0x0000" bus="0x06" slot="0x00" function="0x0"/>
</controller>
```

Notice the value of the `index` attribute. You will need to specify the same number as the value of the `controller` attribute in your disk configuration as described in the next step.

6. In the left pane, click on the SCSI disk device that you want to map to your Virtio-SCSI controller, then open the **XML** tab in the right pane.

Your configuration will look similar to this:

```
<disk type="file" device="disk">
  <driver name="qemu" type="qcow2" cache="writethrough"/>
  <source file="scsi_disk.qcow2"/>
  <target dev="sdc" bus="scsi"/>
  <address type="drive" controller="4" bus="0" target="0" unit="2"/>
</disk>
```

Make sure that the `controller` value in the disk configuration matches the `index` value in the controller configuration as shown above.

7. You can check whether the disk device has been properly mapped to the controller by selecting the SCSI controller in the left pane of the **Virtual Hardware Details** window and opening the **Details** tab.

The **Devices** field will list the devices that are mapped to the selected controller.

8.4.4 Verification on OpenVMS

Boot OpenVMS and execute the appropriate command to see the devices used by the system.

- To list the disk devices, issue **SHOW DEVICES DK**.
- To list the controller devices, issue **SHOW DEVICES PK**.

For example:

```
$ SHOW DEVICES DK
```

Device Name	Device Status	Error Count	Volume Label	Free Blocks	Trans Count	Mnt Cnt
DKC0:	Online	0				
DKD500:	Online	0				
DKE3:	Online	0				
DKA0:	Online wrtlck	0				
DKA200:	Online	0				

```
$ SHOW DEVICES PK
```

Device Name	Device Status	Error Count
PKB0:	Online	0
PKC0:	Online	0
PKD0:	Online	0
PKE0:	Online	0
PKA0:	Online	0

Note the unit number in the device names. It corresponds to the `unit` value in the XML configuration of your disk device. For the SATA disk devices, the unit number is converted to the `nnn` format, whereas for the Virtio-SCSI devices, it strictly corresponds to the unit number.

For example, 500 in `DKD500:` is the unit number in the `nnn` format specified in the XML configuration as follows:

```
<address type="drive" controller="3" bus="0" target="0" unit="5"/>
```

Execute the **SHOW DEVICES *device-name* /FULL** command to view the details on a particular device.

For information on OpenVMS device naming conventions, refer to the [Device Name article](#).

9. Troubleshooting

9.1 KVM Installation and Configuration Issues

This section lists possible errors and their resolution.

9.1.1 Error Messages

9.1.1.1 command not found

When running the `virt-install` command, one of the parameters returns something like this:

```
--memory=8192: command not found
```

This is most commonly caused by an erroneous space character after a backslash. This has been replicated below with hidden characters enabled for better visualization (see line 8):

```
1 myVMName="myMachineName" \
2 myVMDescription="My description" \
3 sudo virt-install \
4 --name="${myVMName}" \
5 --description="${myVMDescription}" \
6 --ram=6000 \
7 --vcpus=3 \
8 --cpu host \
9 --disk device=cdrom,path=${installISOLocationFullPath},bus=sata,boot_order=1,shareable=on \
10 --disk path=${rootedBaseFullPath},size=32,bus=sata,cache=writethrough,sparse=no,discard=unmap \
11 --disk path=${ldBaseFullPath},size=24,bus=sata,cache=writethrough,sparse=no,discard=unmap \
12 --disk path=${sysDiskFullPath},size=8,bus=sata,boot_order=2,cache=writethrough,sparse=no,discard=unmap \
13 --disk device=cdrom,path=${updateKitsISOLocationFullPath},bus=sata,cache=none,shareable=on \
14 --disk device=cdrom,path=${scratchPadISOLocationFullPath},bus=sata,cache=none,shareable=on \
15 --disk device=cdrom,path=${updateKitsISOLocationFullPath},bus=sata,cache=none,shareable=on \
16 --network=bridge:virbr0,mac=52:54:00:BE:EF:01,model='e1000' \
17 --network=bridge:virbr0,mac=52:54:00:BE:EF:02,model='e1000' \
18 --network=bridge:virbr0,mac=52:54:00:BE:EF:03,model='virtio-net' \
19 --graphics none \
20 --os-type none \
21 --os-variant none \
22 --machine q35 \
23 --boot loader=/usr/share/OVMF/OVMF_CODE.secboot.fd,loader.readonly=yes,loader.type=pflash,loader_secure=yes \
24 --features vmport.state=off,smm.state=on \
25 --serial pty \
26 --noreboot \
27 --autostart \
28 #--network="default",model='e1000' \
29 #--host-device pci_0000_00_05_0
```

9.1.1.2 No ACPI MCFG table was found.

This processor may not have a PCIe configuration space.

This error message usually indicates that the wrong chipset was selected:

```
%%%%%%%%%% VSI OpenVMS (tm) x86-64 Console %%%%%%%%%%
```

```
THE GUEST CONSOLE HAS BEEN SUSPENDED
USE A TERMINAL UTILITY FOR OPA0 ACCESS
```

```
No ACPI MCFG table was found. This processor may not have a PCIe configuration
space.
```

```
VSI Primary Kernel SYSBOOT
```

```
%SYSBOOT-F-NOPXML, The PXML database could not be initialized
```

```
%SYSBOOT-F-NOINITMMMG, Unable to initialize SYSBOOT memory management, status =
00000124
```

This can be resolved by editing the virtual machine XML as described below in the section called [Rainbow Screen](#).

9.1.1.3

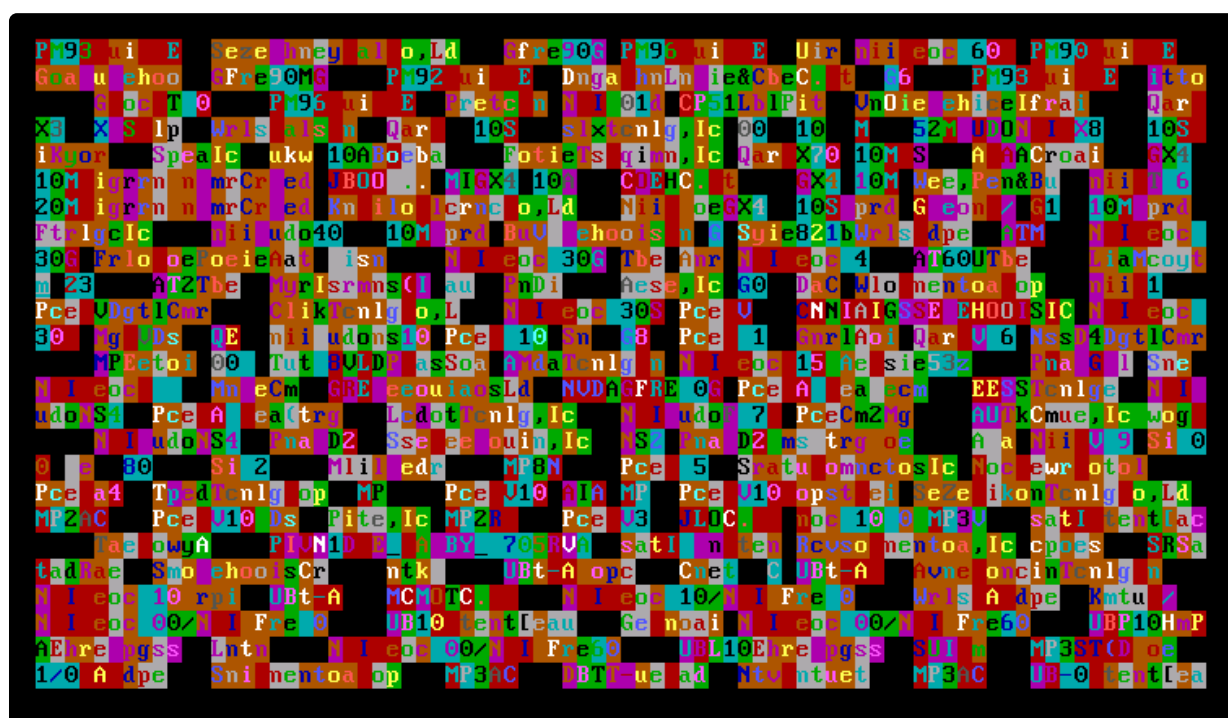
```
(virt-manager:54736): Gtk-WARNING **: 08:42:13.294: cannot open display:
```

When connecting to a virtual machine via SSH and using the `virt-manager` command, the **Virtual Machine Manager** GUI does not open.

This can be resolved by setting up X11 forwarding on your system or in a terminal emulator.

9.1.2 Rainbow Screen

Your screen looks like this when your virtual machine launches:



This happens because of an incorrectly selected chipset option when configuring the virtual machine.

This can be resolved by running the following command:

```
virt-xml vm-name --edit --convert-to-q35
```

Alternatively, you can manually edit your VM XML configuration file. Make sure that the `<os>` element is specified in the following way:

```
<os firmware="efi">
  <type arch="x86_64" machine="pc-q35-8.2">hvm</type>
  .
  .
```

```
.
</os>
```

9.1.3 Non-Persistent Device Boot Order

Due to a bug in KVM, as of libvirt version 9.7.0, changes made to the device boot order in UEFI are not persistent after the VM is restarted. Detailed description of the issue can be found at the following link: [libvirt issue](#).

The workaround suggested by the KVM community to avoid the non-persistent boot order issue involves the utilization of a non-present floppy disk device (`fd`) as the default boot option. The procedure depends on whether or not you have already created a virtual machine.

Existing VM

1. Edit the XML configuration of your KVM virtual machine and add `<boot dev='fd' />` inside the `<os>` element. Make sure to remove other boot device specifications, for example:
`<boot dev='cdrom' />`, `<boot dev='hd' />`, etc.

For information on how you can edit your VM XML configuration, refer to [Customizing Your KVM Virtual Machine](#).

2. Reboot your VM for the changes to take effect.

When creating a VM

When running `virt-install` to set up a new virtual machine, specify the `--boot fd` option and make sure to remove any `boot_order=n` specifications.

See example in [Creating a KVM Virtual Machine](#).

9.1.4 System Time Synchronization

When a KVM VM is powered on, the system time is set based on the value of the `CLOCK OFFSET` parameter in the VM configuration file. The default value is `utc`.

Depending on the physical location of the host system, this might lead to differences between system times of the VM and the host.

To resolve this problem, use the `virsh edit` command to edit the XML configuration file of your virtual machine and change the value of the `CLOCK OFFSET` parameter to `localtime`, like so:

```
<clock offset='localtime'>
```

For more information, see the official documentation for your Linux distribution.

Starting with the E9.2-4 release, VSI OpenVMS provides tools to synchronize time between a VM and a host system. Refer to the [Synchronizing Time Between an OpenVMS Virtual Machine and a Host System](#) section for details.

9.2 IOMMU Configuration Issues

This section describes possible reasons why mapping an HBA device to the host system may fail and ways to resolve the problem.

- Verify that the version of your Linux kernel and KVM support IOMMU functionality.
 - To check the version of the Linux kernel, run `uname -r`.
 - To check the version of KVM, run `/usr/libexec/qemu-kvm --version`.

Refer to the [Supported Platforms and Hardware](#) section for the list of supported versions.

- Make sure that the IOMMU functionality is part of your host boot process by running the following command:

```
dmesg | grep -i "iommu"
```

Your output should look similar to the following:

```
[ 0.000000] Command line: BOOT_IMAGE=(hd1,gpt2)/
vmlinuz-5.14.0-611.16.1.el9_7.x86_64 root=/dev/mapper/rhel-root ro resume=
dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet
crashkernel=1G-2G:192M,2G-64G:256M,64G-:512M intel_iommu=on iommu_pt
[ 0.043575] Kernel command line: BOOT_IMAGE=(hd1,gpt2)/
vmlinuz-5.14.0-611.16.1.el9_7.x86_64 root=/dev/mapper/rhel-root ro resume=
dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet
crashkernel=1G-2G:192M,2G-64G:256M,64G-:512M intel_iommu=on iommu_pt
[ 0.043714] DMAR: IOMMU enabled
[ 0.163101] DMAR-IR: IOAPIC id 12 under DRHD base 0xfbffc000 IOMMU 2
[ 0.163103] DMAR-IR: IOAPIC id 11 under DRHD base 0xf37fc000 IOMMU 1
[ 0.163104] DMAR-IR: IOAPIC id 10 under DRHD base 0xeaffc000 IOMMU 0
[ 0.163105] DMAR-IR: IOAPIC id 8 under DRHD base 0xe27fc000 IOMMU 3
[ 0.163106] DMAR-IR: IOAPIC id 9 under DRHD base 0xe27fc000 IOMMU 3
[ 0.441716] iommu: Default domain type: Translated
[ 0.441716] iommu: DMA domain TLB invalidation policy: lazy mode
.
.
.
```

If the `dmesg | grep -i "iommu"` command does not produce any output, you can run the `find /sys/kernel/iommu_groups/ -type d -maxdepth 1` command which should produce the output similar to the following:

```
/sys/kernel/iommu_groups/
/sys/kernel/iommu_groups/55
/sys/kernel/iommu_groups/83
.
.
.
```

If you do not get any output from either `dmesg | grep -i "iommu"` or `find /sys/kernel/iommu_groups/ -maxdepth 1 -type d`, go back and make sure you have performed the steps described in the [Enabling IOMMU Support in BIOS/UEFI](#) and [Enabling IOMMU Support in Linux Kernel](#) sections.

- Make sure that the necessary drivers are installed and can be loaded by running the `lsmod | grep "kvm"` command.
- On AMD-based hosts, your output will look similar to the following if the drivers are installed:

```
kvm_amd          245760  4
kvm              1404928  3  kvm_amd
ccp              163840  1  kvm_amd
```

- On Intel-based hosts, your output will look similar to the following if the drivers are installed:

```
kvm_intel        356352  0
kvm              970752  1  kvm_intel
irqbypass        12288  1  kvm
```

9.2.1 System Hang

After configuring PCI passthrough, it may be impossible to start OpenVMS on a KVM virtual machine because the system hangs on boot.

Open your CLI and perform the following steps to resolve this issue:

1. Reboot your OpenVMS system by sending the reset signal to your VM by executing `virsh reset vm-name`, and then, at the `BOOTMGR>` prompt, try booting OpenVMS.

If the system still hangs on boot, do the following:

- a. Execute `virsh destroy vm-name` to shut down your VM.
- b. Restart your VM by executing `virsh start vm-name`.
- c. Boot OpenVMS.

If the system still hangs on boot, proceed to the next step.

2. Deassign both ports of the Fibre Channel HBA device by running the `virsh nodedev-detach pci_XXXX_XX_XX_X` command, replacing the `XXXX_XX_XX_X` with the PCI addresses for both ports of the device that you found in the [Assigning Fibre Channel HBA to Virtual Machine](#) section.
3. Boot OpenVMS and check the system version by executing the **PRODUCT SHOW PRODUCT VMS/FULL** command:

```
$ PRODUCT SHOW PRODUCT VMS/FULL
-----
PRODUCT          KIT TYPE    STATE    MAINTENANCE
-----
```

```
VSI X86VMS VMS V9.2-3 Oper System Installed VSI X86VMS VMS923X_DCL V4.0
VSI X86VMS VMS923X_PCSI V1.0
VSI X86VMS VMS923X_PERF_UPD V1.0
VSI X86VMS VMS923X_UPDATE V3.0
```

Make sure that you are running [VSI OpenVMS for x86-64 Version 9.2-3 + Update V3](#) or higher and have the [PERF_UPD V1.0 ECO kit](#) installed.

4. Try re-assigning the Fibre Channel HBA device to your VM. If the problem persists, contact VSI Support via support@vmssoftware.com.

9.2.2 Error Messages

When executing the `sudo virsh start vm-name` command, the system produces the errors specified below:

```
error: Failed to start domain 'vm-name'
error: Requested operation is not valid: PCI device xxxx:xx:xx.x is in use by
driver QEMU, domain vm-name
```

These errors signify that the Fibre Channel HBA device is in use by another VM.

To resolve this, shut down the virtual machine that uses the Fibre Channel HBA and run `sudo virsh start vm-name` again.

Alternatively, assign each VM their designated Fibre Channel HBA device.

10. Reference

10.1 virt-install Command Reference

Here you will find description of the arguments used on the `virt-install` command in the [Creating a KVM Virtual Machine](#) section.

Arguments	Argument options	Description	Recommendation
<code>--name=</code>		The name of your virtual machine. Later on, this name will also be used as a virtual machine identifier.	Use a simple name, without spaces or special characters, as this name will be needed in scripts and other settings/selections.
<code>--description=</code>		A human-readable textual description of your VM.	Useful when having more than one guest and/or multiple system administrators.
<code>--memory=</code>		RAM in MB.	Check the Recommended KVM Virtual Machine Settings for your version of OpenVMS.
<code>--vcpus=</code>		Number of virtual cores.	Check the Recommended KVM Virtual Machine Settings for your version of OpenVMS. If more granularity is needed, it is also possible to specify the number of sockets and cores per socket. For example: <code>--vcpus=4,sockets=4,cores=1</code>.
<code>--cpu</code>	<code>host-model</code>	The CPU model to be used by the guest.	

Arguments	Argument options	Description	Recommendation
<code>--controller</code>	<code>type=</code>	Adds a virtual controller to the virtual machine configuration.	
	<code>model=</code>		
<code>--disk</code>	<code>device=cdrom</code>		Used for loading ISO media (i.e. the operating environment (OE) image).
	<code>path=</code>		Absolute path for disk images.
	<code>bus=sata</code> <code>bus=scsi</code>		OpenVMS on KVM supports only Virtio-SCSI and SATA controllers. Virtio-SCSI controllers provide better I/O performance and lesser CPU load. The lsilogic controller and various types of lsisas are not supported (see VSI OpenVMS x86-64 E9.2-4 Release Notes).
	<code>boot_order=</code>		The order in which the device will boot.
	<code>size=</code>		The disk size in GB. If you are creating a new disk, this parameter is required. If you are going to use an existing disk, this parameter is optional. Refer to the Recommended Virtual Machine Settings for minimum disk size for your VSI OpenVMS version.
	<code>cache=writethrough</code>		This is the one mode supported by OpenVMS, do not change it. If you do not specify a cache mode, <code>writeback</code> will be used by default, <i>which is not supported and may lead to data loss.</i>
	<code>sparse=no</code>		This option will create thick-provisioned disks, which is a requirement. Thin-provisioned disks are <i>not</i> supported on OpenVMS.

Arguments	Argument options	Description	Recommendation
<code>--network=</code>	<code>bridge=</code>		The name of the bridge will depend on your host settings: <code>virbr0</code> , <code>br0</code> , <code>bridge0</code> , etc.
	<code>model=e1000</code> <code>model=e1000e</code> <code>model=virtio-net</code>		The following network devices are supported: • <code>virtio-net</code> (does <i>not</i> support Virtual Local Area Network (VLAN) interfaces, refer to Advanced Virtio-net Devices Configuration for more details.) • <code>e1000e</code> • <code>e1000</code> Refer to the Recommended Virtual Machine Settings section for more details.
	<code>mac=your-mac-address</code>		Replace with the MAC address of your Ethernet device that you wish to pass through to your VM.
	<code>type=direct</code> <code>source=iface</code> <code>source_mode=mode</code>		This specifies the network pass-through device and the macvtap mode. More details about different modes can be found in the Setting the NIC model article. If you require multiple virtual machines to communicate using a single network card, this method is not applicable.
<code>--graphics</code>	<code>none</code>	Graphical console mode.	Use <code>none</code> if you are not using the Guest Console.
	<code>vnc</code>		
	<code>sdl</code>		
	<code>spice</code>		If you are going to use the Guest Console or other graphical software, such as DECwindows, select the appropriate option or omit the argument and let <code>virsh</code> set the default.

Arguments	Argument options	Description	Recommendation
<code>--os-variant</code>	<code>none</code>	Disables OS auto-detection.	
<code>--machine</code>	<code>q35</code>	Chipset selection.	Essential for UEFI mode. If you have configured this incorrectly the first time, the only way to correct it is via editing the XML file.
<code>--boot</code>	<code>loader=/path/loader.fd</code> <code>loader.readonly=yes</code> <code>loader.type=pflash</code> <code>loader.secure=yes</code> <code>nvram.template=template.fd</code>		This is specific to the Linux distribution. Not only the path, but also the file name can be different. For Red Hat Enterprise Linux and Oracle Linux, see examples in Creating a KVM VM.
	<code>fd</code>	Floppy disk boot device.	Specifying a floppy disk as the default boot device ensures that the user-defined boot order persists across VM reboots. See Non-Persistent Device Boot Order for more information.
<code>--features</code>	<code>vmport.state=off</code> <code>smm.state=on</code>		This is necessary for UEFI.
<code>--serial</code>	<code>pty</code>		This option creates a local serial connection to your virtual machine.
	<code>tcp</code> <code>host=0.0.0.0:port</code> <code>mode=bind</code> <code>protocol=telnet</code>		This option creates a Telnet TCP/IP-accessible port into your OPA0: terminal. Note that this traffic is not password-protected or encrypted and requires opening the corresponding port in the Linux firewall settings.

Arguments	Argument options	Description	Recommendation
<code>--watchdog</code>	<code>model=itco</code> <code>action=none</code>	Watchdog is an agent that monitors the host for a hang or crash and takes an action specified.	This is automatically applied to every virtual machine created in KVM, and while it has no effect on OpenVMS, the <code>none</code> action is specified to ensure it will not take any action.
<code>--noreboot</code>			This stops the VM from automatically being rebooted on updates to the configuration from either the GUI, CLI, or XML.
<code>--autostart</code>		Automatically boots the guest VM once the host has booted and KVM core has started.	This option is <i>not</i> the same as the AUTOACTION BOOT command within the OpenVMS Boot Manager. VSI recommends that you leave this on, unless you want your VM to be enabled but switched off after initialization.