



VMS Software

VSI OpenVMS on KVM

Configuration Guide

Publication Date: January 2026

Operating System: VSI OpenVMS x86-64 Version 9.2-2 or higher

Table of Contents

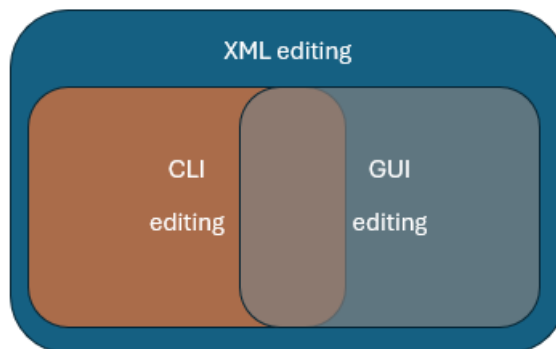
1. Before You Begin	3
2. Prerequisites	4
2.1. Installation Requirements	4
2.2. VSI OpenVMS Version Requirements	4
3. Creating a KVM Virtual Machine	5
4. Managing Your KVM Virtual Machine	6
4.1. Cloning a Virtual Machine	6
4.2. Deleting a Virtual Machine	6
4.3. Managing Disks	7
4.3.1. Viewing Disk Properties	7
4.3.2. Creating Disks	8
4.3.3. Increasing Disk Size	8
4.3.4. Attaching Disk to an Existing Virtual Machine	8
4.3.5. Converting Images	9
4.4. Managing Network	10
4.4.1. Adding a Network Interface to an Existing Virtual Machine	10
4.4.2. Creating Network Bridges	10
4.4.3. Advanced Virtio-net Devices Configuration	11
5. Customizing Your KVM Virtual Machine	11
5.1. Editing XML via Virtual Machine Manager GUI	12
5.2. Editing XML via a Linux Text Editor	12
6. Interacting With a VM as an Unprivileged User	12
7. Troubleshooting	14
7.1. Error Messages	14
7.2. Rainbow Screen	15
8. virt-install Command Reference	15

1. Before You Begin

This guide explains how to 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 herein should still be applicable.

There are three 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)	<ul style="list-style-type: none"> Allows for a quick start with predefined templates or scripts. Best for default deployments. 	<ul style="list-style-type: none"> Special configurations require crafting long commands. Command syntax can change with updates.
Virtual Machine Manager GUI (virt-manager)	<ul style="list-style-type: none"> 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)	<ul style="list-style-type: none"> 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.
XML editing	<ul style="list-style-type: none"> Best for advanced deployments. Provides the most customizability. 	Higher risk of configuration errors.

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

2. Prerequisites

2.1. Installation Requirements

Before you proceed with installing your KVM virtual machine, 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
```

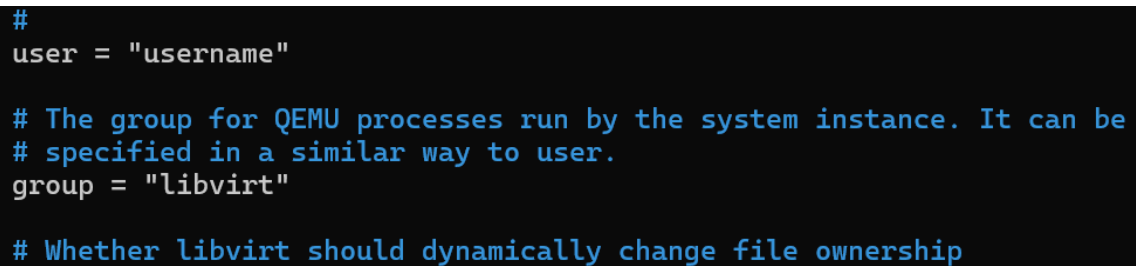
2. Edit the `/etc/libvirt/qemu.conf` file in the following way:

- 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`.

Important

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

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



```
#
user = "username"

# The group for QEMU processes run by the system instance. It can be
# specified in a similar way to user.
group = "libvirt"

# Whether libvirt should dynamically change file ownership
```

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

```
sudo systemctl restart libvirtd
```

2.2. VSI OpenVMS Version Requirements

Review the requirements and recommended virtual machine settings for your VSI OpenVMS version:

- [VSI OpenVMS x86-64 V9.2-2](#)
- [VSI OpenVMS x86-64 V9.2-3](#)

3. Creating a KVM Virtual Machine

Tip

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.

The command below will create a virtual machine configuration that works for any supported version of VSI OpenVMS for x86-64. Copy the command and replace the parts in bold with actual values.

```
networkdev="network_device_name"
mac=$(cat /sys/class/net/$networkdev/address)
sudo virt-install \
--name "vm_name" \
--description "vm_description" \
--memory=8192 \
--vcpus=4 \
--cpu host-model \
--controller type=scsi,model=virtio-scsi \
--disk device=cdrom,path=/path/to/vms/install.iso,bus=sata,boot_order=1 \
--disk path=/path/to/vms/disk.qcow2, \
size=30,bus=scsi,cache=writethrough,sparse=no,boot_order=2 \
--network type=direct,source=$networkdev,source_mode=passthrough, \
mac=$mac,model=virtio-net \
--graphics none \
--machine q35 \
--os-variant none \
--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
```

After the command above 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
```

Then, use the following command to connect to the OpenVMS Boot Manager:

```
sudo virsh console vm_name
```

For a detailed description of the `virt-install` command used in the [example above](#), along with the recommended settings and alternatives that may depend on your system requirements, refer to *Section 8, "virt-install Command Reference"*.

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

```
man virt-install
```

Attention

If you are an unprivileged user, perform the steps described in *Section 6, "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 *Section 4, "Managing Your KVM Virtual Machine"* for more details.

4. Managing Your KVM Virtual Machine

Sections below discuss how to perform common tasks with your KVM virtual machine using CLI, **Virtual Machine Manager** GUI, or by editing the XML configuration directly.

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 through the web. Further documentation on Cockpit can be found on the following webpage: <https://cockpit-project.org/running.html#rhel>.

4.1. Cloning a 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.

4.2. Deleting a Virtual Machine

Deleting a VM via 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.

Deleting a VM via Virtual Machine Manager 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.

4.3. Managing Disks

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>.

4.3.1. 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.

Important

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

4.3.2. Creating Disks

Before you create a disk, check the installation guide for your version of OpenVMS for the recommended disk size:

- [VSI OpenVMS x86-64 V9.2-2](#)
- [VSI OpenVMS x86-64 V9.2-3](#)

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, compression, and snapshots, while <code>raw</code> provides marginally better performance.
-o	<code>preallocation=full</code>	Specifies how the disk is provisioned.	You <i>must</i> specify <code>preallocation=full</code> when creating a new disk for OpenVMS.

For the full documentation on the `qemu-img` command, refer to <https://linux.die.net/man/1/qemu-img>.

4.3.3. 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.

Caution

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

4.3.4. Attaching Disk 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/X8609220E.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.

2. Run the following command, replacing `vm_name`, `/path/to/disk/disk_name.qcow2`, and `new_virtual_device_name` with appropriate values:

```

sudo virsh attach-disk vm_name /path/to/disk/disk_name.qcow2 new_virtual_device_name \
--driver qemu --subdriver qcow2 --cache writethrough --persistent

```

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>		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 <https://www.libvirt.org/manpages/virsh.html#attach-disk>.

4.3.5. Converting Images

To convert a disk from one format to another, use the following command:

```
qemu-img convert -f raw -O qcow2 source.img destination.qcow2
```

Note that this command does not delete the original disk.

The table below explains the arguments used in the `qemu-img convert` command.

Arguments	Description	Recommended options
<code>-f</code>	The input file format.	
<code>-O</code>	The output file format.	It is recommended that you use <code>qcow2</code> or <code>raw</code> . <code>qcow2</code> supports encryption, compression, and snapshots, while <code>raw</code> provides marginally better performance.

For the full documentation on the `qemu-img convert` command, refer to <https://docs.openstack.org/image-guide/convert-images.html>.

4.4. Managing Network

This section describes multiple ways to manage network and network interfaces for KVM virtual machines.

4.4.1. Adding a Network Interface to an Existing Virtual Machine

You can add an additional network interface to an existing virtual machine via CLI, GUI, or by editing XML 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 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.
 - a. In the **Network source** drop-down menu, choose **Macvtap device**. The **Device name** field will appear.
 - b. Type your host interface name into the **Device name** field.
 - c. Type your host interface MAC address into the **MAC address** field.
 - d. In the **Device model** drop-down menu, choose **virtio**.
 - e. Click **Finish**.

When you click on the newly created device in the left menu in the [Virtual Hardware Details](#) window, make sure that the **Link state** field shows a selected checkbox next to "active".

XML

Add the following line to your XML configuration file:

```
<source dev="target_interface" mode='direct' />
```

4.4.2. Creating Network Bridges

Some installations will require network bridges to be configured. For example, if you have multiple virtual machines that need 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.

Once the bridge has been created, you can replace the `--network` line in the `virt-install` command with the bridge interface, specifying the MAC of your bridge interface:

```
--network=bridge:bridge0,mac=bridge_interface_mac,model='virtio-net '
```

4.4.3. 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 and will choose one for you by default.

- 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 <https://libvirt.org/formatdomain.html#virtio-device-models>.

5. Customizing Your 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.

5.1. 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**.

Click **Close**.

3. Open the virtual machine you that wish to edit by selecting the VM and clicking the **Open** button. A new window will open.
4. 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.
5. After you have made the necessary changes, click **Apply**.
6. 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.

5.2. 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:

```
export EDITOR=/usr/bin/nano
sudo virsh edit vm_name
```

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

6. Interacting With a VM as an Unprivileged User

For an 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 into the system.

7. Troubleshooting

This section lists possible errors and their resolution.

7.1. Error Messages

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" \CRIF
2 myVMDescription="My description." \CRIF
3 sudo virt-install \CRIF
4 --name="{myVMName}" \CRIF
5 --description="{myVMDescription}" \CRIF
6 --ram=6000 \CRIF
7 --vcpus=3 \CRIF
8 --cpu host \CRIF
9 --disk device=cdrom,path={installISOLocationFullPath},bus=sata,boot_order=1,shareable=on \CRIF
10 --disk path={rootedBaseFullPath},size=32,bus=sata,cache=writethrough,sparse=no,discard=unmap \CRIF
11 --disk path={ldBaseFullPath},size=24,bus=sata,cache=writethrough,sparse=no,discard=unmap \CRIF
12 --disk device=cdrom,path={sysDiskFullPath},size=8,bus=sata,boot_order=2,cache=writethrough,sparse=no,discard=unmap \CRIF
13 --disk device=cdrom,path={updateKitsISOLocationFullPath},bus=sata,cache=none,shareable=on \CRIF
14 --disk device=cdrom,path={scratchPadISOLocationFullPath},bus=sata,cache=none,shareable=on \CRIF
15 --disk device=cdrom,path={updateKitsISOLocationFullPath},bus=sata,cache=none,shareable=on \CRIF
16 --network=bridge:virbr0,mac=52:54:00:BE:EF:01,model='e1000' \CRIF
17 --network=bridge:virbr0,mac=52:54:00:BE:EF:02,model='e1000' \CRIF
18 --network=bridge:virbr0,mac=52:54:00:BE:EF:03,model='virtio-net' \CRIF
19 --graphics none \CRIF
20 --os-type none \CRIF
21 --os-variant none \CRIF
22 --machine q35 \CRIF
23 --boot loader=/usr/share/OVMF/OVMF_CODE.secboot.fd,loader.readonly=yes,loader.type=pflash,loader_secure=yes \CRIF
24 --features vmport.state=off,smm.state=on \CRIF
25 --serial pty \CRIF
26 --noreboot \CRIF
27 --autostart \CRIF
28 #--network="default",model='e1000e' \CRIF
29 #--host-device pci_0000_00_05_0

```

%SYSBOOT-F-NOPXML

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

```

THE GUEST CONSOLE HAS BEEN SUSPENDED
USE A TERMINAL UTILITY FOR OPAO 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 managment, status = 00000124

```

This can be resolved by editing the virtual machine XML as described below in *Section 7.2, "Rainbow Screen"*.

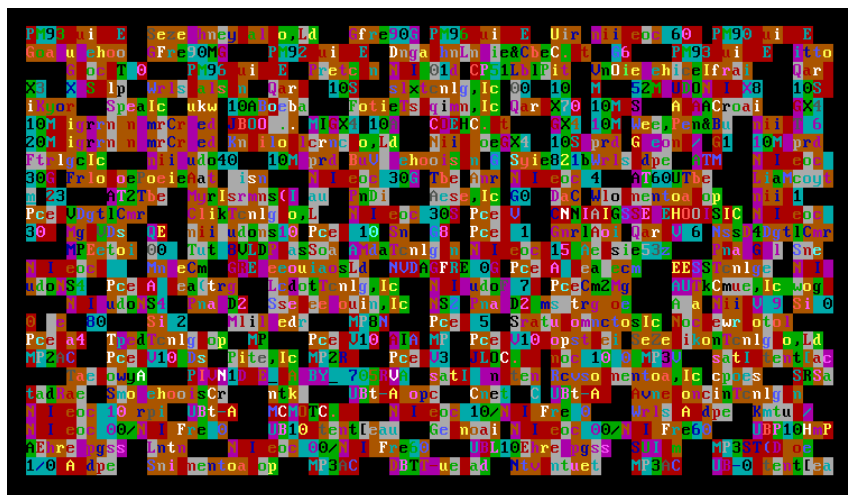
```
(virt-manager:54736): Gtk-WARNING **: 08:42:13.294: cannot open display:
```

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.

7.2. Rainbow Screen

Your screen renders 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>
  <firmware>
    <feature enabled="no" name="enrolled-keys"/>
    <feature enabled="no" name="secure-boot"/>
  </firmware>
  <loader readonly="yes" type="pflash">/usr/share/OVMF/OVMF_CODE.secboot.fd</loader>
  <bootmenu enable="yes"/>
</os>
```

8. virt-install Command Reference

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.

Arguments	Argument options	Description	Recommendation
<code>--description=</code>		A human-readable textual description of your VM.	Essential when having more than one guest and/or multiple system admins.
<code>--memory=</code>		RAM in MB.	Check the installation guide for your version of OpenVMS for the recommended values.
<code>--vcpus=</code>		Number of virtual cores.	Check the installation guide for your version of OpenVMS for the recommended values. 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.	
<code>--controller</code>	<code>type=</code> <code>model=</code>	Adds a virtual controller to the virtual machine configuration.	
<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 Supported Disk Types).
	<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 <i>Section</i>

Arguments	Argument options	Description	Recommendation
			2.2, "VSI OpenVMS Version Requirements" for minimum disk size for your VSI OpenVMS version.
	cache= writethrough		This is the one mode supported by OpenVMS, do not change it. If you do not specify a cache mode, writeback will be used by default, <i>which is not supported and may lead to data loss.</i>
	sparse=no		This option specifies that the disks must be thick-provisioned, which is a requirement.
--network=	bridge=		The name of the bridge will depend on your host settings: virbr0, br0, bridge0, etc.
	model=e1000 model=e1000e model=virtio-net		As of OpenVMS V9.2-3 , the following network devices are supported: <ul style="list-style-type: none"> • virtio-net (does <i>not</i> support Virtual Local Area Network (VLAN) interfaces, refer to <i>Section 4.4.3, "Advanced Virtio-net Devices Configuration"</i> for more details.) • e1000e • e1000 (use this if installing a previous version of OpenVMS)
	mac= <i>your_mac_address</i>		Replace with the MAC address of your Ethernet device that you wish to pass through to your VM.
	type=direct source= <i>iface</i> source_mode= <i>mode</i>		This specifies the network pass-through device and the macvtap mode. More details can be found about the different modes at https://libvirt.org/

Arguments	Argument options	Description	Recommendation
			formatdomain.html#setting-the-nic-model . If you require multiple virtual machines to communicate using a single network card, this method is not applicable. See <i>Section 4.4.2, "Creating Network Bridges"</i> for creating a network bridge to use instead.
--graphics	none vnc sdl spice	Graphical console mode.	Use none if you are not using the Guest Console. If you are going to use the Guest Console or other graphical utilities, such as DECwindows, select the appropriate option.
--os-variant	none	Disables OS auto-detection.	
--machine	q35	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.
--boot	loader=/path/ loader.fd loader.readonly=yes loader.type=pflash loader_secure=yes		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, the path is /usr/share/OVMF/ OVMF_CODE.secboot.fd
--features	vmport.state=off smm.state=on		This is necessary for UEFI.
--serial	pty		This option creates a local serial connection to your virtual machine.
	tcp host=0.0.0.0:port mode=bind		This option creates a Telnet TCP/IP-accessible port into your OPA0: terminal.

Arguments	Argument options	Description	Recommendation
	<code>protocol=telnet</code>		Note that this traffic is not password-protected or encrypted and requires opening the corresponding port in the Linux firewall settings.
<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 AUTO 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.