Red Hat Enterprise Linux 10-beta

Configuring and managing Linux virtual machines

Setting up your host, creating and administering virtual machines, and understanding virtualization features

Last Updated: 2025-03-04

Red Hat Enterprise Linux 10-beta Configuring and managing Linux virtual machines

Setting up your host, creating and administering virtual machines, and understanding virtualization features

Abstract

Virtualization features of Red Hat Enterprise Linux (RHEL) allow you to configure your host to run virtual machines (VMs). With virtual machines, you can use different operating systems and have separate computing environments on your local host.

Table of Contents

RHEL BETA RELEASE

Red Hat provides Red Hat Enterprise Linux Beta access to all subscribed Red Hat accounts. The purpose of Beta access is to:

- Provide an opportunity to customers to test major features and capabilities prior to the general availability release and provide feedback or report issues.
- Provide Beta product documentation as a preview. Beta product documentation is under development and is subject to substantial change.

Note that Red Hat does not support the usage of RHEL Beta releases in production use cases. For more information, see the Red Hat Knowledgebase solution What does Beta mean in Red Hat Enterprise Linux and can I upgrade a RHEL Beta installation to a General Availability (GA) release?.

CHAPTER 1. BASIC CONCEPTS OF VIRTUALIZATION IN RHEL

If you are unfamiliar with the concept of virtualization or its implementation in Linux, the following sections provide a general overview of virtualization in RHEL 10: its basics, advantages, components, and other possible virtualization solutions provided by Red Hat.

1.1. WHAT IS VIRTUALIZATION?

RHEL 10 provides the *virtualization* functionality, which enables a machine running RHEL 10 to *host* multiple virtual machines (VMs), also referred to as *guests*. VMs use the host's physical hardware and computing resources to run a separate, virtualized operating system (*guest OS*) as a user-space process on the host's operating system.

In other words, virtualization makes it possible to have operating systems within operating systems.

VMs enable you to safely test software configurations and features, run legacy software, or optimize the workload efficiency of your hardware. For more information about the benefits, see Advantages of virtualization.

For more information about what virtualization is, see the Virtualization topic page.

Next steps

- To start using virtualization in Red Hat Enterprise Linux 10, see Enabling virtualization in Red Hat Enterprise Linux {ProductNumber}.
- In addition to Red Hat Enterprise Linux 10 virtualization, Red Hat offers a number of specialized virtualization solutions, each with a different user focus and features. For more information, see Red Hat virtualization solutions .

1.2. ADVANTAGES OF VIRTUALIZATION

Using virtual machines (VMs) has the following benefits in comparison to using physical machines:

- **Flexible and fine-grained allocation of resources**
  A VM runs on a host machine, which is usually physical, and physical hardware can also be assigned for the guest OS to use. However, the allocation of physical resources to the VM is done on the software level, and is therefore very flexible. A VM uses a configurable fraction of the host memory, CPUs, or storage space, and that configuration can specify very fine-grained resource requests.

  For example, what the guest OS sees as its disk can be represented as a file on the host file system, and the size of that disk is less constrained than the available sizes for physical disks.
- **Software-controlled configurations**
  The entire configuration of a VM is saved as data on the host, and is under software control. Therefore, a VM can easily be created, removed, cloned, migrated, operated remotely, or connected to remote storage.
- **Separation from the host**
  A guest OS runs on a virtualized kernel, separate from the host OS. This means that any OS can be installed on a VM, and even if the guest OS becomes unstable or is compromised, the host is not affected in any way.
- **Space and cost efficiency**

A single physical machine can host a large number of VMs. Therefore, it avoids the need for multiple physical machines to do the same tasks, and thus lowers the space, power, and maintenance requirements associated with physical hardware.

- **Software compatibility**
  Because a VM can use a different OS than its host, virtualization makes it possible to run applications that were not originally released for your host OS. For example, using a RHEL 7 guest OS, you can run applications released for RHEL 7 on a RHEL 10 host system.

1.3. VIRTUAL MACHINE COMPONENTS AND THEIR INTERACTION

Virtualization in RHEL 10 consists of the following principal software components:

Hypervisor

The basis of creating virtual machines (VMs) in RHEL 10 is the *hypervisor*, a software layer that controls hardware and enables running multiple operating systems on a host machine.

The hypervisor includes the **Kernel-based Virtual Machine (KVM)**module and virtualization kernel drivers. These components ensure that the Linux kernel on the host machine provides resources for virtualization to user-space software.

At the user-space level, the **QEMU** emulator simulates a complete virtualized hardware platform that the guest operating system can run in, and manages how resources are allocated on the host and presented to the guest.

In addition, the **libvirt** software suite serves as a management and communication layer, making QEMU easier to interact with, enforcing security rules, and providing a number of additional tools for configuring and running VMs.

XML configuration

A host-based XML configuration file (also known as a *domain XML* file) determines all settings and devices in a specific VM. The configuration includes:

- Metadata such as the name of the VM, time zone, and other information about the VM.
- A description of the devices in the VM, including virtual CPUs (vCPUS), storage devices, input/output devices, network interface cards, and other hardware, real and virtual.
- VM settings such as the maximum amount of memory it can use, restart settings, and other settings about the behavior of the VM.

Component interaction

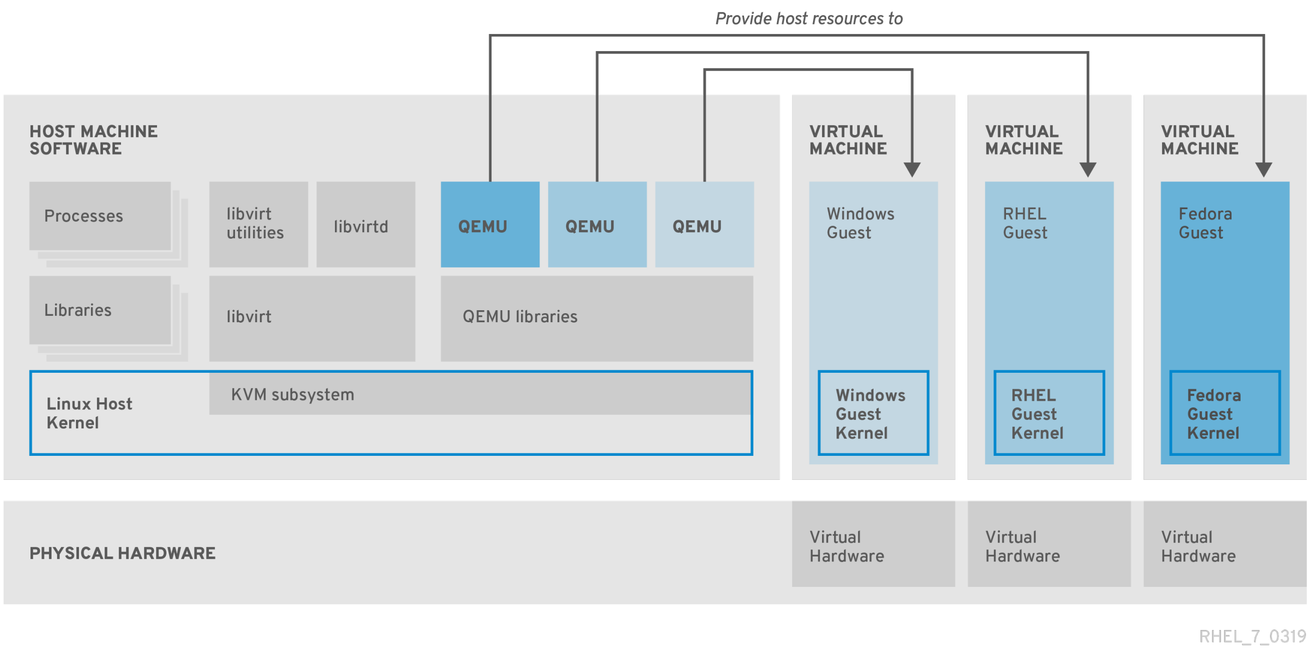
When a VM is started, the hypervisor uses the XML configuration to create an instance of the VM as a user-space process on the host. The hypervisor also makes the VM process accessible to the host-based interfaces, such as the **virsh**, **virt-install**, and **guestfish** utilities, or the web console GUI.

When these virtualization tools are used, libvirt translates their input into instructions for QEMU. QEMU communicates the instructions to KVM, which ensures that the kernel appropriately assigns the resources necessary to carry out the instructions. As a result, QEMU can execute the corresponding user-space changes, such as creating or modifying a VM, or performing an action in the VM’s guest operating system.

NOTE

While QEMU is an essential component of the architecture, it is not intended to be used directly on RHEL 10 systems, due to security concerns. Therefore, **qemu-*** commands are not supported by Red Hat, and it is highly recommended to interact with QEMU by using libvirt.

Figure 1.1. RHEL 10 virtualization architecture



1.4. TOOLS AND INTERFACES FOR VIRTUALIZATION MANAGEMENT

You can manage virtualization in RHEL 10 by using the command line (CLI) or several graphical user interfaces (GUIs).

Command-line interface

The CLI is the most powerful method of managing virtualization in RHEL 10. Prominent CLI commands for virtual machine (VM) management include:

- **virsh** - A versatile virtualization command-line utility and shell with a great variety of purposes, depending on the provided arguments. For example:
  - Starting and shutting down a VM - **virsh start** and **virsh shutdown**
  - Listing available VMs - **virsh list**
  - Creating a VM from a configuration file - **virsh create**
  - Entering a virtualization shell - **virsh**

  For more information, see the **virsh(1)** man page on your system.
- **virt-install** - A CLI utility for creating new VMs. For more information, see the **virt-install(1)** man page on your system.
- **virt-xml** - A utility for editing the configuration of a VM.

- **guestfish** - A utility for examining and modifying VM disk images. For more information, see the **guestfish(1)** man page on your system.

Graphical interfaces

You can use the following GUIs to manage virtualization in RHEL 10:

- The **RHEL 10 web console**, also known as *Cockpit*, provides a remotely accessible and easy to use graphical user interface for managing VMs and virtualization hosts.

1.5. RED HAT VIRTUALIZATION SOLUTIONS

The following Red Hat products are built on top of RHEL 10 virtualization features and expand the KVM virtualization capabilities available in RHEL 10.

OpenShift Virtualization

Based on the KubeVirt technology, OpenShift Virtualization is a part of the Red Hat OpenShift Container Platform, and makes it possible to run virtual machines in containers.
For more information about OpenShift Virtualization see the Red Hat Hybrid Cloud pages.

Red Hat OpenStack Platform (RHOSP)

Red Hat OpenStack Platform offers an integrated foundation to create, deploy, and scale a secure and reliable public or private OpenStack cloud.
For more information about Red Hat OpenStack Platform, see the Red Hat Customer Portal or the Red Hat OpenStack Platform documentation suite .

CHAPTER 2. PREPARING RHEL TO HOST VIRTUAL MACHINES

To use virtualization in RHEL 10, you must install virtualization packages and ensure your system is configured to host virtual machines (VMs). The specific steps to do this vary based on your CPU architecture.

2.1. PREPARING AN AMD64 OR INTEL 64 SYSTEM TO HOST VIRTUAL MACHINES

To set up a KVM hypervisor and create virtual machines (VMs) on an AMD64 or Intel 64 system running RHEL 10, follow the instructions below.

Prerequisites

- Red Hat Enterprise Linux 10 is installed and registered on your host machine.
- Your system meets the following hardware requirements to work as a virtualization host:
  - The following minimum system resources are available:
    - 6 GB free disk space for the host, plus another 6 GB for each intended VM.
    - 2 GB of RAM for the host, plus another 2 GB for each intended VM.

Procedure

1. Install the virtualization hypervisor packages.

```
# dnf install qemu-kvm libvirt virt-install virt-viewer
```

2. Start the virtualization services:

```
# for drv in qemu network nodedev nwfilter secret storage interface; do systemctl start
virt${drv}d{,-ro,-admin}.socket; done
```

Verification

1. Verify that your system is prepared to be a virtualization host:

```
# virt-host-validate
[...]
QEMU: Checking for device assignment IOMMU support         : PASS
QEMU: Checking if IOMMU is enabled by kernel               : WARN (IOMMU appears to be
disabled in kernel. Add intel_iommu=on to kernel cmdline arguments)
LXC: Checking for Linux >= 2.6.26                          : PASS
[...]
LXC: Checking for cgroup 'blkio' controller mount-point    : PASS
LXC: Checking if device /sys/fs/fuse/connections exists    : FAIL (Load the 'fuse' module to
enable /proc/ overrides)
```

2. If all **virt-host-validate** checks return a **PASS** value, your system is prepared for
   If any of the checks return a **FAIL** value, follow the displayed instructions to fix the problem.

If any of the checks return a **WARN** value, consider following the displayed instructions to improve virtualization capabilities.

Troubleshooting

- If KVM virtualization is not supported by your host CPU, **virt-host-validate** generates the following output:

  ```
  QEMU: Checking for hardware virtualization: FAIL (Only emulated CPUs are available, performance will be significantly limited)
  ```

  However, VMs on such a host system will fail to boot, rather than have performance problems.

  To work around this, you can change the **<domain type>** value in the XML configuration of the VM to **qemu**. Note, however, that Red Hat does not support VMs that use the **qemu** domain type, and setting this is highly discouraged in production environments.

2.2. PREPARING AN IBM Z SYSTEM TO HOST VIRTUAL MACHINES

To set up a KVM hypervisor and create virtual machines (VMs) on an IBM Z system running RHEL 10, follow the instructions below.

Prerequisites

- The following minimum system resources are available:
  - 6 GB free disk space for the host, plus another 6 GB for each intended VM.
  - 2 GB of RAM for the host, plus another 2 GB for each intended VM.
  - 4 CPUs on the host. VMs can generally run with a single assigned vCPU, but Red Hat recommends assigning 2 or more vCPUs per VM to avoid VMs becoming unresponsive during high load.
- Your IBM Z host system is using a z13 CPU or later.
- RHEL 10 is installed on a logical partition (LPAR). In addition, the LPAR supports the *start-interpretive execution* (SIE) virtualization functions.
  To verify this, search for **sie** in your **/proc/cpuinfo** file.

  ```
  # grep sie /proc/cpuinfo
  features        : esan3 zarch stfle msa ldisp eimm dfp edat etf3eh highgprs te sie
  ```

Procedure

1. Install the virtualization packages:

   ```
   # dnf install qemu-kvm libvirt virt-install
   ```

2. Start the virtualization services:

   ```
   # for drv in qemu network nodedev nwfilter secret storage interface; do systemctl start virt${drv}d{,-ro,-admin}.socket; done
   ```

Verification

1. Verify that your system is prepared to be a virtualization host.

```
# virt-host-validate
[...]
QEMU: Checking if device /dev/kvm is accessible            : PASS
QEMU: Checking if device /dev/vhost-net exists             : PASS
QEMU: Checking if device /dev/net/tun exists               : PASS
QEMU: Checking for cgroup 'memory' controller support      : PASS
QEMU: Checking for cgroup 'memory' controller mount-point  : PASS
[...]
```

2. If all **virt-host-validate** checks return a **PASS** value, your system is prepared for
   If any of the checks return a **FAIL** value, follow the displayed instructions to fix the problem.

   If any of the checks return a **WARN** value, consider following the displayed instructions to improve virtualization capabilities.

Troubleshooting

- If KVM virtualization is not supported by your host CPU, **virt-host-validate** generates the following output:

  ```
  QEMU: Checking for hardware virtualization: FAIL (Only emulated CPUs are available,
  performance will be significantly limited)
  ```

  However, VMs on such a host system will fail to boot, rather than have performance problems.

  To work around this, you can change the **<domain type>** value in the XML configuration of the VM to **qemu**. Note, however, that Red Hat does not support VMs that use the **qemu** domain type, and setting this is highly discouraged in production environments.

2.3. PREPARING AN ARM 64 SYSTEM TO HOST VIRTUAL MACHINES

To set up a KVM hypervisor for creating virtual machines (VMs) on an ARM 64 system (also known as **AArch64**) that runs RHEL 10, follow the instructions below.

Prerequisites

- The following minimum system resources are available:
  - 6 GB free disk space for the host, plus another 6 GB for each intended guest.
  - 4 GB of RAM for the host, plus another 4 GB for each intended guest.

Procedure

1. Install the virtualization packages:

   ```
   # dnf install qemu-kvm libvirt virt-install
   ```

2. Start the virtualization services:

```
# for drv in qemu network nodedev nwfilter secret storage interface; do systemctl start
virt${drv}d{,-ro,-admin}.socket; done
```

Verification

1. Verify that your system is prepared to be a virtualization host:

   ```
   # virt-host-validate
   [...]
   QEMU: Checking if device /dev/vhost-net exists              : PASS
   QEMU: Checking if device /dev/net/tun exists                : PASS
   QEMU: Checking for cgroup 'memory' controller support       : PASS
   QEMU: Checking for cgroup 'memory' controller mount-point   : PASS
   [...]
   QEMU: Checking for cgroup 'blkio' controller support        : PASS
   QEMU: Checking for cgroup 'blkio' controller mount-point    : PASS
   QEMU: Checking if IOMMU is enabled by kernel                : WARN (Unknown if this platform
   has IOMMU support)
   ```

2. If all **virt-host-validate** checks return a **PASS** value, your system is prepared for creating virtual machines.
   If any of the checks return a **FAIL** value, follow the displayed instructions to fix the problem.

   If any of the checks return a **WARN** value, consider following the displayed instructions to improve virtualization capabilities.

2.4. ENABLING QEMU GUEST AGENT FEATURES ON YOUR VIRTUAL MACHINES

To use certain features on a virtual machine (VM) hosted on your RHEL 10 system, you must first configure the VM to use the QEMU Guest Agent (GA).

For a complete list of these features, see Virtualization features that require QEMU Guest Agent .

The specific steps required to configure QEMU GA on a VM differ based on the guest operating system used by the VM:

- For Linux VMs, see Enabling QEMU Guest Agent on Linux guests .

2.4.1. Enabling QEMU Guest Agent on Linux guests

To allow a RHEL host to perform a certain subset of operations on a Linux virtual machine (VM), you must enable the QEMU Guest Agent (GA).

You can enable QEMU GA both on running and shut-down VMs.

Procedure

1. Create an XML configuration file for the QEMU GA, for example named **qemuga.xml**:

   ```
   # touch qemuga.xml
   ```

2. Add the following lines to the file:

```
<channel type='unix'>
  <source mode='bind' path='/var/lib/libvirt/qemu/f16x86_64.agent'/>
  <target type='virtio' name='org.qemu.guest_agent.0'/>
</channel>
```

3. Use the XML file to add QEMU GA to the configuration of the VM.

    - If the VM is running, use the following command:

    ```
    # virsh attach-device <vm-name> qemuga.xml --live --config
    ```

    - If the VM is shut-down, use the following command:

    ```
    # virsh attach-device <vm-name> qemuga.xml --config
    ```

4. In the Linux guest operating system, install the QEMU GA:

    ```
    # dnf install qemu-guest-agent
    ```

5. Start the QEMU GA service on the guest:

    ```
    # systemctl start qemu-guest-agent
    ```

Verification

To ensure that QEMU GA is enabled and running on the Linux VM, do any of the following:

- In the guest operating system, use the **systemctl status qemu-guest-agent | grep Loaded** command. If the output includes **enabled**, QEMU GA is active on the VM.

- Use the **virsh domfsinfo *<vm-name>*** command on the host. If it displays any output, QEMU GA is active on the specified VM.

Additional resources

- Virtualization features that require QEMU Guest Agent

2.4.2. Virtualization features that require QEMU Guest Agent

If you enable QEMU Guest Agent (GA) on a virtual machine (VM), you can use the following commands on your host to manage the VM:

virsh shutdown --mode=agent

This shutdown method is more reliable than **virsh shutdown --mode=acpi**, because **virsh shutdown** used with QEMU GA is guaranteed to shut down a cooperative guest in a clean state.

virsh domfsfreeze and **virsh domfsthaw**

Freezes the guest file system in isolation.

virsh domfstrim

Instructs the guest to trim its file system, which helps to reduce the data that needs to be transferred during migrations.

IMPORTANT

If you want to use this command to manage a Linux VM, you must also set the following SELinux boolean in the guest operating system:

```
# setsebool virt_qemu_ga_read_nonsecurity_files on
```

virsh domtime

Queries or sets the guest's clock.

virsh setvcpus --guest

Instructs the guest to take CPUs offline, which is useful when CPUs cannot be hot-unplugged.

virsh domifaddr --source agent

Queries the guest operating system's IP address by using QEMU GA. For example, this is useful when the guest interface is directly attached to a host interface.

virsh domfsinfo

Shows a list of mounted file systems in the running guest.

virsh set-user-password

Sets the password for a given user account in the guest.

virsh set-user-sshkeys

Edits the authorized SSH keys file for a given user in the guest.

IMPORTANT

If you want to use this command to manage a Linux VM, you must also set the following SELinux boolean in the guest operating system:

```
# setsebool virt_qemu_ga_manage_ssh on
```

Additional resources

- Enabling QEMU Guest Agent on Linux guests

2.5. SETTING UP THE WEB CONSOLE TO MANAGE VIRTUAL MACHINES

Before using the RHEL 10 web console to manage virtual machines (VMs), you must install the web console virtual machine plug-in on the host.

Prerequisites

- You have installed the RHEL 10 web console.
  For instructions, see Installing and enabling the web console .

Procedure

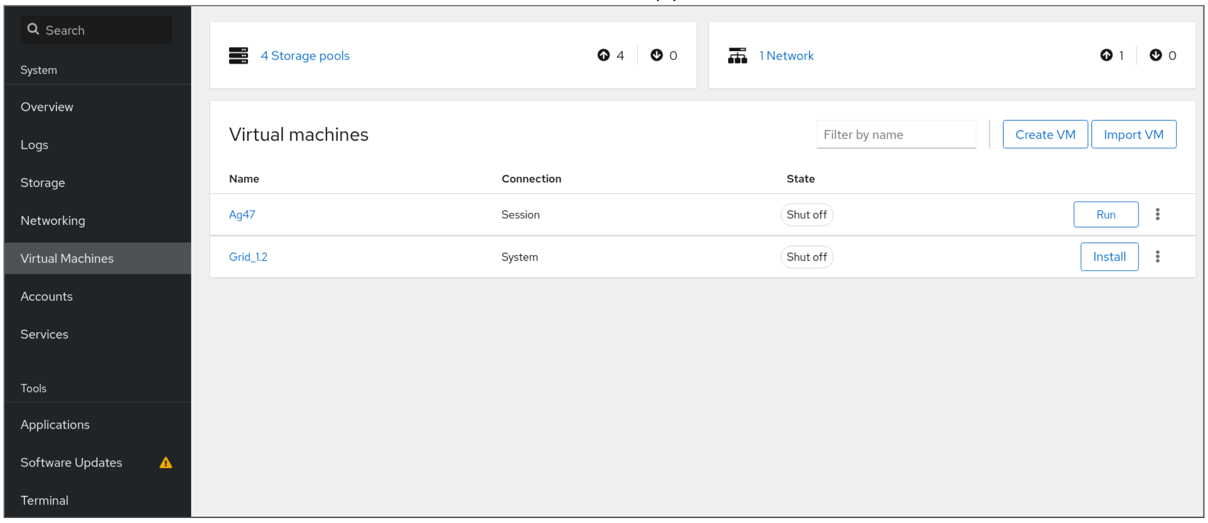
- Install the **cockpit-machines** plug-in.

```
# dnf install cockpit-machines
```

Verification

1. Log in to the RHEL 10 web console.
   For details, see Logging in to the web console .

2. If the installation was successful, **Virtual Machines** appears in the web console side menu.



Virtual machines

Name	Connection	State
Ag47	Session	Shut off
Grid_1.2	System	Shut off

Additional resources

- Managing systems by using the RHEL 10 web console

CHAPTER 3. CREATING VIRTUAL MACHINES

To create a virtual machine (VM) in RHEL 10, you can use the command line or the RHEL 10 web console.

3.1. CREATING VIRTUAL MACHINES BY USING THE COMMAND LINE

You can create a virtual machine (VM) on your RHEL 10 host by using the **virt-install** utility.

Prerequisites

- Virtualization is enabled on your host system.
- You have a sufficient amount of system resources to allocate to your VMs, such as disk space, RAM, or CPUs. The recommended values may vary significantly depending on the intended tasks and workload of the VMs.
- An operating system (OS) installation source is available locally or on a network. This can be one of the following:
    - An ISO image of an installation medium
    - A disk image of an existing VM installation

    WARNING

    Installing from a host CD-ROM or DVD-ROM device is not possible in RHEL 10. If you select a CD-ROM or DVD-ROM as the installation source when using any VM installation method available in RHEL 10, the installation will fail. For more information, see the Red Hat Knowledgebase.

- Optional: A Kickstart file can be provided for faster and easier configuration of the installation.

Procedure

To create a VM and start its OS installation, use the **virt-install** command, along with the following mandatory arguments:

- **--name**: the name of the new machine
- **--memory**: the amount of allocated memory
- **--vcpus**: the number of allocated virtual CPUs
- **--disk**: the type and size of the allocated storage
- **--cdrom** or **--location**: the type and location of the OS installation source

Based on the chosen installation method, the necessary options and values can vary. See the commands below for examples:

- The following command creates a VM named **demo-guest1** that installs the Windows 10 OS from an ISO image locally stored in the **/home/username/Downloads/Win10install.iso** file. This VM is also allocated with 2048 MiB of RAM and 2 vCPUs, and an 80 GiB qcow2 virtual disk is automatically configured for the VM.

```
# virt-install \
    --name demo-guest1 --memory 2048 \
    --vcpus 2 --disk size=80 --os-variant win10 \
    --cdrom /home/username/Downloads/Win10install.iso
```

- The following command creates a VM named **demo-guest2** that uses the **/home/username/Downloads/rhel10.iso** image to run a RHEL 10 OS from a live CD. No disk space is assigned to this VM, so changes made during the session will not be preserved. In addition, the VM is allocated with 4096 MiB of RAM and 4 vCPUs.

```
# virt-install \
    --name demo-guest2 --memory 4096 --vcpus 4 \
    --disk none --livecd --os-variant rhel10.0 \
    --cdrom /home/username/Downloads/rhel10.iso
```

- The following command creates a RHEL 10 VM named **demo-guest3** that connects to an existing disk image, **/home/username/backup/disk.qcow2**. This is similar to physically moving a hard drive between machines, so the OS and data available to demo-guest3 are determined by how the image was handled previously. In addition, this VM is allocated with 2048 MiB of RAM and 2 vCPUs.

```
# virt-install \
    --name demo-guest3 --memory 2048 --vcpus 2 \
    --os-variant rhel10.0 --import \
    --disk /home/username/backup/disk.qcow2
```

  Note that the **--os-variant** option is highly recommended when importing a disk image. If it is not provided, the performance of the created VM will be negatively affected.

- The following command creates a VM named **demo-guest4** that installs from the **http://example.com/OS-install** URL. For the installation to start successfully, the URL must contain a working OS installation tree. In addition, the OS is automatically configured by using the **/home/username/ks.cfg** kickstart file. This VM is also allocated with 2048 MiB of RAM, 2 vCPUs, and a 160 GiB qcow2 virtual disk.

```
# virt-install \
    --name demo-guest4 --memory 2048 --vcpus 2 --disk size=160 \
    --os-variant rhel10.0 --location http://example.com/OS-install \
    --initrd-inject /home/username/ks.cfg --extra-args="inst.ks=file:/ks.cfg console=tty0
console=ttyS0,115200n8"
```

  In addition, if you want to host demo-guest4 on an RHEL 10 on an ARM 64 host, include the following lines to ensure that the kickstart file installs the **kernel-64k** package:

```
%packages
-kernel
kernel-64k
%end
```

- The following command creates a VM named **demo-guest5** that installs from a **RHEL10.iso** image file in text-only mode, without graphics. It connects the guest console to the serial console. The VM has 16384 MiB of memory, 16 vCPUs, and 280 GiB disk. This kind of installation is useful when connecting to a host over a slow network link.

  ```
  # virt-install \
     --name demo-guest5 --memory 16384 --vcpus 16 --disk size=280 \
     --os-variant rhel10.0 --location RHEL10.iso \
     --graphics none --extra-args='console=ttyS0'
  ```

- The following command creates a VM named **demo-guest6**, which has the same configuration as demo-guest5, but resides on the 192.0.2.1 remote host.

  ```
  # virt-install \
     --connect qemu+ssh://root@192.0.2.1/system --name demo-guest6 --memory 16384 \
     --vcpus 16 --disk size=280 --os-variant rhel10.0 --location RHEL10.iso \
     --graphics none --extra-args='console=ttyS0'
  ```

- The following command creates a VM named **demo-guest-7**, which has the same configuration as demo-guest5, but for its storage, it uses a DASD mediated device **mdev_30820a6f_b1a5_4503_91ca_0c10ba12345a_0_0_29a8**, and assigns it device number **1111**.

  ```
  # virt-install \
     --name demo-guest7 --memory 16384 --vcpus 16 --disk size=280 \
     --os-variant rhel10.0 --location RHEL10.iso --graphics none \
     --disk none --hostdev
  mdev_30820a6f_b1a5_4503_91ca_0c10ba12345a_0_0_29a8,address.type=ccw,address.cssid
  =0xfe,address.ssid=0x0,address.devno=0x1111,boot-order=1 \
     --extra-args 'rd.dasd=0.0.1111'
  ```

  Note that the name of the mediated device available for installation can be retrieved by using the **virsh nodedev-list --cap mdev** command.

Troubleshooting

- If **virt-install** fails with a **cannot find default network** error:
  - Ensure that the **libvirt-daemon-config-network** package is installed:

    ```
    # dnf info libvirt-daemon-config-network
    Installed Packages
    Name         : libvirt-daemon-config-network
    [...]
    ```

  - Verify that the **libvirt** default network is active and configured to start automatically:

    ```
    # virsh net-list --all
     Name      State    Autostart   Persistent
    --------------------------------------------
     default   active   yes         yes
    ```

  - If it is not, activate the default network and set it to auto-start:

```
# virsh net-autostart default
Network default marked as autostarted

# virsh net-start default
Network default started
```

- If activating the default network fails with the following error, the **libvirt-daemon-config-network** package has not been installed correctly.

  ```
  error: failed to get network 'default'
  error: Network not found: no network with matching name 'default'
  ```

  To fix this, re-install **libvirt-daemon-config-network**:

  ```
  # dnf reinstall libvirt-daemon-config-network
  ```

- If activating the default network fails with an error similar to the following, a conflict has occurred between the default network's subnet and an existing interface on the host.

  ```
  error: Failed to start network default
  error: internal error: Network is already in use by interface ens2
  ```

  To fix this, use the **virsh net-edit default** command and change the **192.0.2.*** values in the configuration to a subnet not already in use on the host.

Additional resources

- **virt-install (1)** man page on your system
- Creating virtual machines by using the web console

3.2. CREATING VIRTUAL MACHINES BY USING THE WEB CONSOLE

You can create virtual machines (VMs) in a GUI on a RHEL 10 host by using the web console.

3.2.1. Creating new virtual machines by using the web console

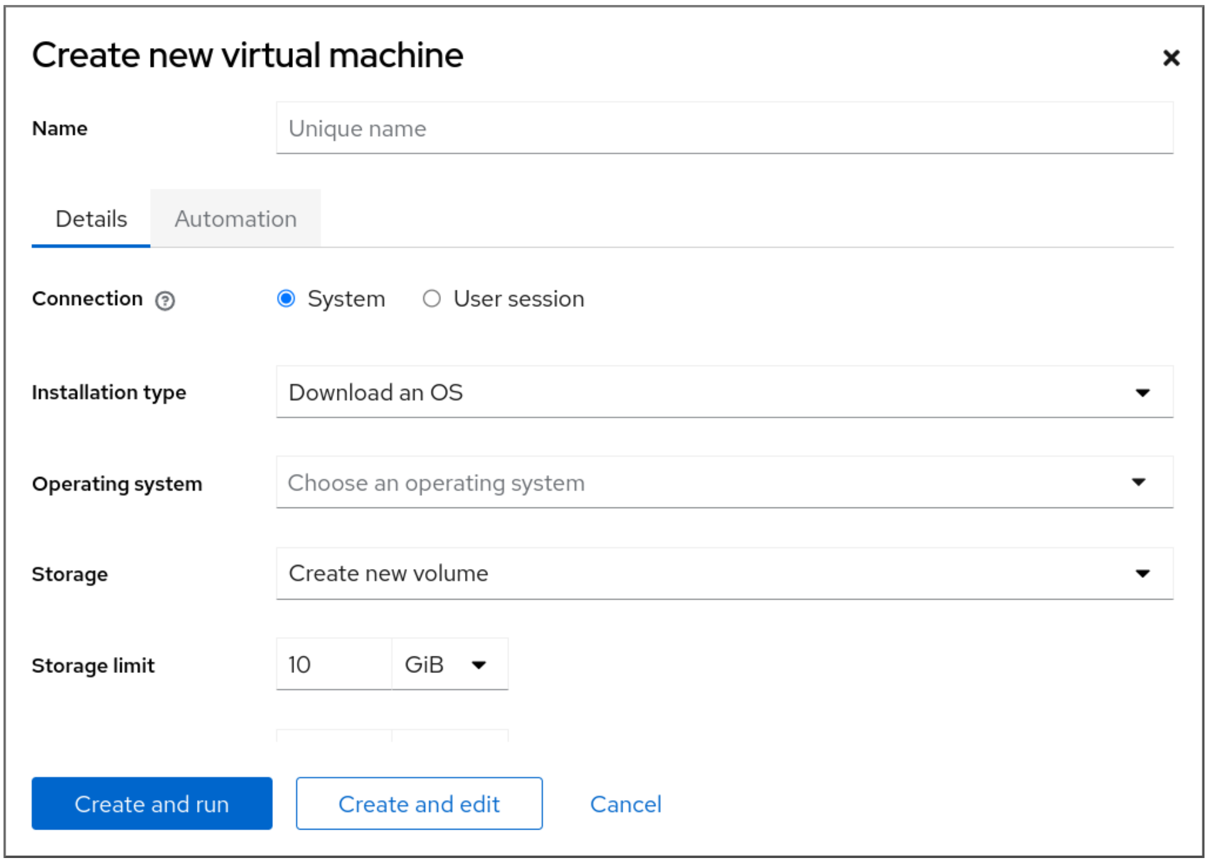
You can create a new virtual machine (VM) on a previously prepared host machine by using the RHEL 10 web console.

Prerequisites

- You have installed the RHEL 10 web console.
  For instructions, see Installing and enabling the web console .
- Virtualization is enabled on your host system .
- The web console VM plug-in is installed on your host system .
- You have a sufficient amount of system resources to allocate to your VMs, such as disk space, RAM, or CPUs. The recommended values might vary significantly depending on the intended tasks and workload of the VMs.

Procedure

1. In the **Virtual Machines** interface of the web console, click **Create VM**.
   The **Create new virtual machine** dialog appears.



Create new virtual machine

Name	Unique name
Connection	[✓ System] User session
Installation type	Download an OS
Operating system	Choose an operating system
Storage	Create new volume
Storage limit	10 GiB

2. Enter the basic configuration of the VM you want to create.
   - **Name** - The name of the VM.
   - **Connection** - The level of privileges granted to the session. For more details, expand the associated dialog box in the web console.
   - **Installation type** - The installation can use a local installation medium, a URL, a PXE network boot, a cloud base image, or download an operating system from a limited set of operating systems.
   - **Operating system** - The guest operating system running on the VM. Note that Red Hat provides support only for a limited set of guest operating systems.

     NOTE

     To download and install Red Hat Enterprise Linux directly from web console, you must add an offline token in the **Offline token** field.

- **Storage** - The type of storage.
- **Storage Limit** - The amount of storage space.
- **Memory** - The amount of memory.

  1. Create the VM:

- If you want the VM to automatically install the operating system, click **Create and run**.
- If you want to edit the VM before the operating system is installed, click **Create and edit**.

Next steps

- Installing guest operating systems by using the web console

Additional resources

- Creating virtual machines by using the command line

3.2.2. Creating virtual machines by importing disk images with the web console

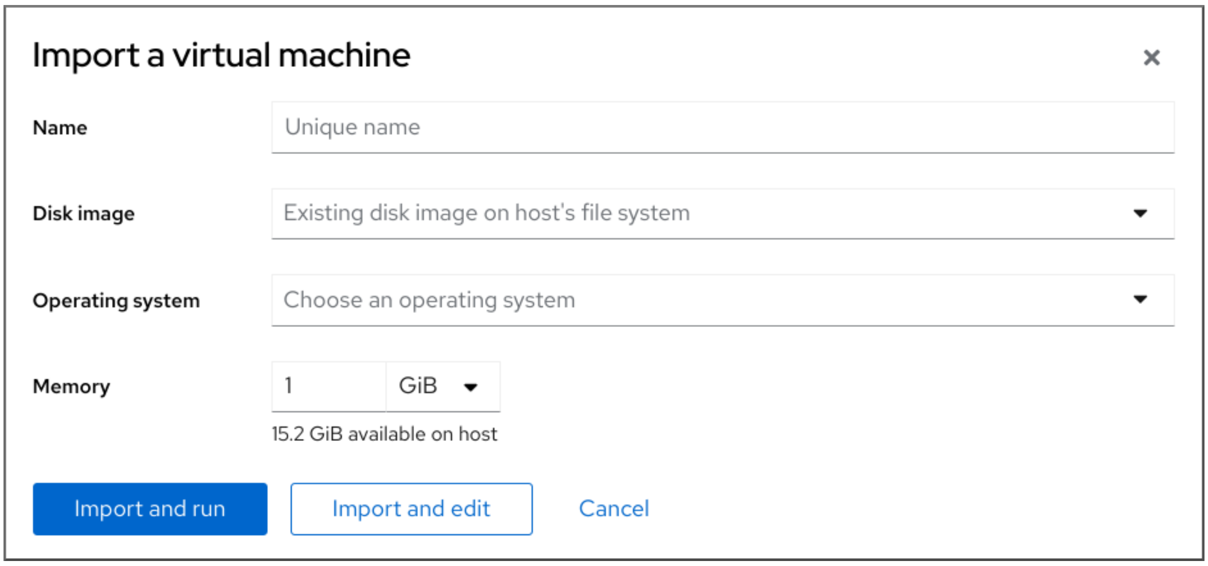
You can create a virtual machine (VM) by importing a disk image of an existing VM installation in the RHEL 10 web console.

Prerequisites

- You have installed the RHEL 10 web console.
  For instructions, see Installing and enabling the web console .
- The web console VM plug-in is installed on your system .
- You have a sufficient amount of system resources to allocate to your VMs, such as disk space, RAM, or CPUs. The recommended values can vary significantly depending on the intended tasks and workload of the VMs.
- You have downloaded a disk image of an existing VM installation.

Procedure

1. In the **Virtual Machines** interface of the web console, click **Import VM**.
   The **Import a virtual machine dialog** appears.



Import a virtual machine

Name	Unique name
Disk image	Existing disk image on host's file system
Operating system	Choose an operating system
Memory	1 GiB 15.2 GiB available on host

2. Enter the basic configuration of the VM you want to create:
   - **Name** - The name of the VM.

  - **Disk image** - The path to the existing disk image of a VM on the host system.
  - **Operating system** - The operating system running on a VM disk. Note that Red Hat provides support only for a limited set of guest operating systems.
- **Memory** - The amount of memory to allocate for use by the VM.

  1. Import the VM:
     - To install the operating system on the VM without additional edits to the VM settings, click **Import and run**.
     - To edit the VM settings before the installation of the operating system, click **Import and edit**.

3.2.3. Installing guest operating systems by using the web console

When a new virtual machine (VM) boots for the first time, you must install an operating system on the VM.

NOTE

If you click **Create and run** or **Import and run** while creating a new VM, the installation routine for the operating system starts automatically when the VM is created.

Prerequisites

- You have installed the RHEL 10 web console.
  For instructions, see Installing and enabling the web console .
- The web console VM plugin is installed on your host system .

Procedure

1. Log in to the RHEL 10 web console.
   For details, see Logging in to the web console .
2. In the **Virtual Machines** interface, click the VM on which you want to install a guest OS.
   A new page opens with basic information about the selected VM and controls for managing various aspects of the VM.
3. Optional: Change the firmware.

   NOTE

   You can change the firmware only if you selected **Create and edit** or **Import and edit** while creating a new VM and if the OS is not already installed on the VM.

   a. Click the firmware.
   b. In the **Change Firmware** window, select the required firmware.
   c. Click **Save**.
4. Click **Install**.

The installation routine of the operating system runs in the VM console.

Troubleshooting

- If the installation routine fails, delete and recreate the VM before starting the installation again.

3.2.4. Creating virtual machines with cloud image authentication by using the web console

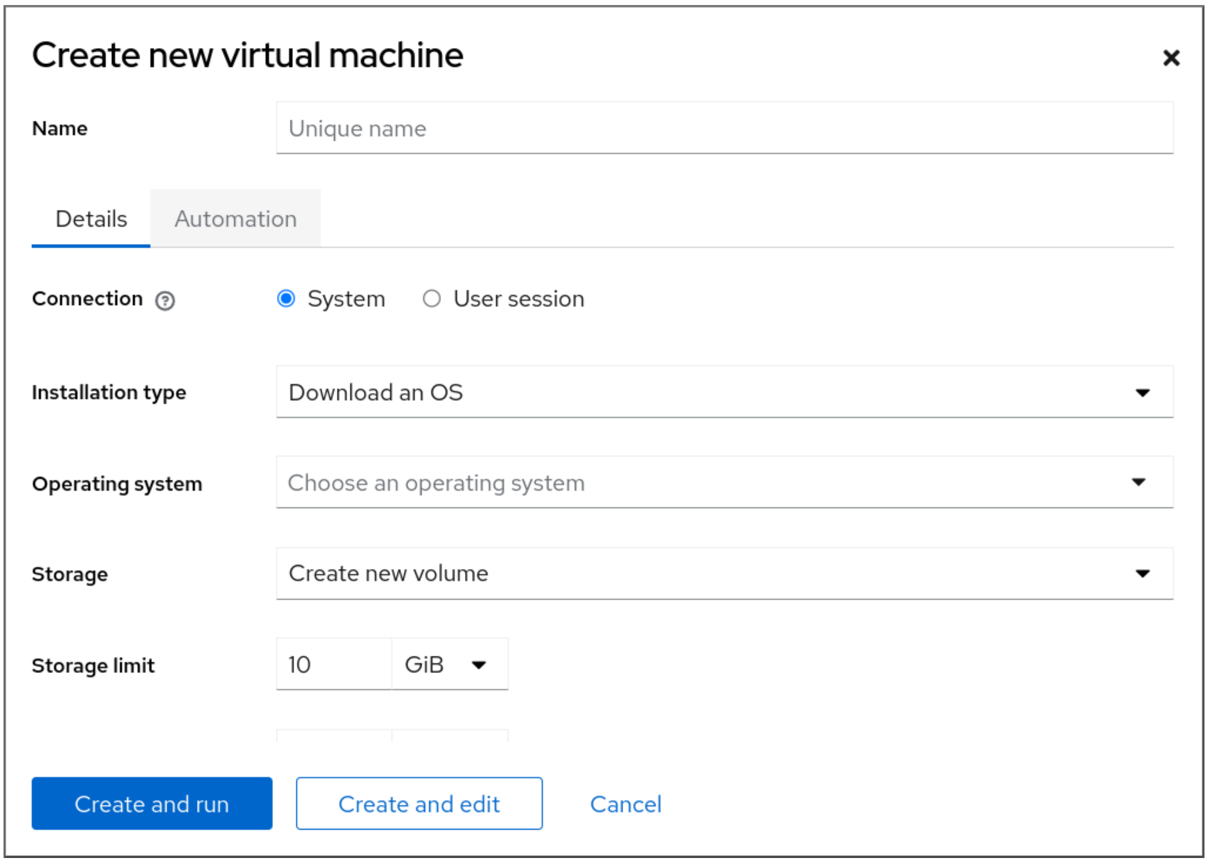
By default, distro cloud images have no login accounts. However, by using the RHEL web console, you can now create a virtual machine (VM) and specify the root and user account login credentials, which are then passed to cloud-init.

Prerequisites

- You have installed the RHEL 10 web console.
  For instructions, see Installing and enabling the web console .
- The web console VM plug-in is installed on your system .
- Virtualization is enabled on your host system.
- You have a sufficient amount of system resources to allocate to your VMs, such as disk space, RAM, or CPUs. The recommended values may vary significantly depending on the intended tasks and workload of the VMs.

Procedure

1. Log in to the RHEL 10 web console.
   For details, see Logging in to the web console .
2. In the **Virtual Machines** interface of the web console, click **Create VM**.
   The Create new virtual machine dialog appears.



Name: Unique name

Connection: [✓ System] User session

Installation type: Download an OS

Operating system: Choose an operating system

Storage: Create new volume

Storage limit: 10 GiB

3. In the **Name** field, enter a name for the VM.

4. On the **Details** tab, in the **Installation type** field, select **Cloud base image**

5. In the **Installation source** field, set the path to the image file on your host system.

6. Enter the configuration for the VM that you want to create.

    - **Operating system** - The VM's operating system. Note that Red Hat provides support only for a limited set of guest operating systems.

- **Storage** - The type of storage with which to configure the VM.

- **Storage Limit** - The amount of storage space with which to configure the VM.

- **Memory** - The amount of memory with which to configure the VM.

    1. Click on the **Automation** tab.
       Set your cloud authentication credentials.

- **Root password** - Enter a root password for your VM. Leave the field blank if you do not wish to set a root password.

- **User login** - Enter a cloud-init user login. Leave this field blank if you do not wish to create a user account.

- **User password** - Enter a password. Leave this field blank if you do not wish to create a user account.

    1. Click **Create and run**.
       The VM is created.

Additional resources

- Installing an operating system on a VM

CHAPTER 4. STARTING VIRTUAL MACHINES

To start a virtual machine (VM) in RHEL 10, you can use the command line interface or the web console GUI.

Prerequisites

- Before a VM can be started, it must be created and, ideally, also installed with an OS. See Creating virtual machines.

4.1. STARTING A VIRTUAL MACHINE BY USING THE COMMAND LINE

You can use the command line interface (CLI) to start a shut-down virtual machine (VM) or restore a saved VM. By using the CLI, you can start both local and remote VMs.

Prerequisites

- An inactive VM that is already defined.
- The name of the VM.
- For remote VMs:
  - The IP address of the host where the VM is located.
  - Root access privileges to the host.

Procedure

- For a local VM, use the **virsh start** utility.
  For example, the following command starts the *demo-guest1* VM.

  ```
  # virsh start demo-guest1
  Domain 'demo-guest1' started
  ```

- For a VM located on a remote host, use the **virsh start** utility along with the QEMU+SSH connection to the host.
  For example, the following command starts the *demo-guest1* VM on the 192.0.2.1 host.

  ```
  # virsh -c qemu+ssh://root@192.0.2.1/system start demo-guest1

  root@192.0.2.1's password:

  Domain 'demo-guest1' started
  ```

Additional resources

- The **virsh start --help** command

4.2. STARTING VIRTUAL MACHINES BY USING THE WEB CONSOLE

If a virtual machine (VM) is in the *shut off* state, you can start it by using the RHEL 10 web console. You can also configure the VM to be started automatically when the host starts.

Prerequisites

- You have installed the RHEL 10 web console.
  For instructions, see Installing and enabling the web console .
- The web console VM plug-in is installed on your system.
- An inactive VM that is already defined.
- The name of the VM.

Procedure

1. In the **Virtual Machines** interface, click the VM you want to start.
   A new page opens with detailed information about the selected VM and controls for shutting down and deleting the VM.
2. Click **Run**.
   The VM starts, and you can connect to its console or graphical output .
3. Optional: To configure the VM to start automatically when the host starts, toggle the **Autostart** checkbox in the **Overview** section.
   If you use network interfaces that are not managed by libvirt, you must also make additional changes to the systemd configuration. Otherwise, the affected VMs might fail to start, see starting virtual machines automatically when the host starts .

4.3. STARTING VIRTUAL MACHINES AUTOMATICALLY WHEN THE HOST STARTS

When a host with a running virtual machine (VM) restarts, the VM is shut down, and must be started again manually by default. To ensure a VM is active whenever its host is running, you can configure the VM to be started automatically.

Prerequisites

- A created virtual machine

Procedure

1. Use the **virsh autostart** utility to configure the VM to start automatically when the host starts.
   For example, the following command configures the *demo-guest1* VM to start automatically.

```
# virsh autostart demo-guest1
Domain 'demo-guest1' marked as autostarted
```

2. If you use network interfaces that are not managed by **libvirt**, you must also make additional changes to the systemd configuration. Otherwise, the affected VMs might fail to start.

NOTE

These interfaces include for example:

- Bridge devices created by **NetworkManager**
- Networks configured to use **<forward mode='bridge'/>**

a. In the systemd configuration directory tree, create a **virtqemud.service.d** directory if it does not exist yet.

```
# mkdir -p /etc/systemd/system/virtqemud.service.d/
```

b. Create a **10-network-online.conf** systemd unit override file in the previously created directory. The content of this file overrides the default systemd configuration for the **virtqemud** service.

```
# touch /etc/systemd/system/virtqemud.service.d/10-network-online.conf
```

c. Add the following lines to the **10-network-online.conf** file. This configuration change ensures systemd starts the **virtqemud** service only after the network on the host is ready.

```
[Unit]
After=network-online.target
```

Verification

1. View the VM configuration, and check that the *autostart* option is enabled.
   For example, the following command displays basic information about the *demo-guest1* VM, including the *autostart* option.

```
# virsh dominfo demo-guest1
Id:             2
Name:           demo-guest1
UUID:           e46bc81c-74e2-406e-bd7a-67042bae80d1
OS Type:        hvm
State:          running
CPU(s):         2
CPU time:       385.9s
Max memory:     4194304 KiB
Used memory:    4194304 KiB
Persistent:     yes
Autostart:      enable
Managed save:   no
Security model: selinux
Security DOI:   0
Security label: system_u:system_r:svirt_t:s0:c873,c919 (enforcing)
```

2. If you use network interfaces that are not managed by libvirt, check if the content of the **10-network-online.conf** file matches the following output.

```
$ cat /etc/systemd/system/virtqemud.service.d/10-network-online.conf
[Unit]
After=network-online.target
```

▪

Additional resources

- The **virsh autostart --help** command
- Starting virtual machines by using the web console .

CHAPTER 5. CONNECTING TO VIRTUAL MACHINES

To interact with a virtual machine (VM) in RHEL 10, you need to connect to it by doing one of the following:

- If you need to interact with a VM graphical display without using the web console, use the Virt Viewer application. For details, see Opening a virtual machine graphical console by using the command line.
- When a graphical display is not possible or not necessary, use an SSH terminal connection .
- When the virtual machine is not reachable from your system by using a network, use the virsh console.

Prerequisites

- The VMs you want to interact with are installed and started.

5.1. OPENING A VIRTUAL MACHINE GRAPHICAL CONSOLE BY USING THE COMMAND LINE

You can connect to a graphical console of a KVM virtual machine (VM) by opening it in the **Virt Viewer** utility.

Prerequisites

- Your system, as well as the VM you are connecting to, must support graphical displays.
- If the target VM is located on a remote host, connection and root access privileges to the host are needed.

Procedure

- To connect to a local VM, use the following command and replace *guest-name* with the name of the VM you want to connect to:

```
# virt-viewer guest-name
```

- To connect to a remote VM, use the **virt-viewer** command with the SSH protocol. For example, the following command connects as root to a VM called *guest-name*, located on remote system 192.0.2.1. The connection also requires root authentication for 192.0.2.1.

```
# virt-viewer --direct --connect qemu+ssh://root@192.0.2.1/system guest-name
root@192.0.2.1's password:
```

Verification

If the connection works correctly, the VM display is shown in the **Virt Viewer** window.

You can interact with the VM console by using the mouse and keyboard in the same manner you interact with a real machine. The display in the VM console reflects the activities being performed on the VM.

Troubleshooting

- If clicking in the graphical console does not have any effect, expand the console to full screen. This is a known issue with the mouse cursor offset.

Additional resources

- **virt-viewer** man page on your system

5.2. CONNECTING TO A VIRTUAL MACHINE BY USING SSH

You can interact with the terminal of a virtual machine (VM) by using the SSH connection protocol.

Prerequisites

- You have network connection and root access privileges to the target VM.
- If the target VM is located on a remote host, you also have connection and root access privileges to that host.
- Your VM network assigns IP addresses by **dnsmasq** generated by **libvirt**. This is the case for example in **libvirt** NAT networks.
  Notably, if your VM is using one of the following network configurations, you cannot connect to the VM by using SSH:
  - **hostdev** interfaces
  - Direct interfaces
  - Bridge interaces
- The **libvirt-nss** component is installed and enabled on the VM's host. If it is not, do the following:
  a. Install the **libvirt-nss** package:

  ```
  # dnf install libvirt-nss
  ```

  b. Edit the **/etc/nsswitch.conf** file and add **libvirt_guest** to the **hosts** line:

  ```
  ...
  passwd:     compat
  shadow:     compat
  group:      compat
  hosts:      files libvirt_guest dns
  ...
  ```

Procedure

1. When connecting to a remote VM, SSH into its physical host first. The following example demonstrates connecting to a host machine **192.0.2.1** by using its root credentials:

```
# ssh root@192.0.2.1
root@192.0.2.1's password:
Last login: Mon Sep 24 12:05:36 2021
root~#
```

2. Use the VM’s name and user access credentials to connect to it. For example, the following connects to to the **testguest1** VM by using its root credentials:

```
# ssh root@testguest1
root@testguest1's password:
Last login: Wed Sep 12 12:05:36 2018
root~]#
```

Troubleshooting

- If you do not know the VM’s name, you can list all VMs available on the host by using the **virsh list --all** command:

```
# virsh list --all
Id    Name                          State
----------------------------------------------------
2     testguest1                    running
-     testguest2                    shut off
```

Additional resources

- Upstream libvirt documentation

5.3. OPENING A VIRTUAL MACHINE SERIAL CONSOLE BY USING THE COMMAND LINE

By using the **virsh console** command, it is possible to connect to the serial console of a virtual machine (VM).

This is useful when the VM: * Does not provide VNC protocols, and thus does not offer video display for GUI tools.

- Does not have a network connection, and thus cannot be interacted with using SSH.

Prerequisites

- The GRUB boot loader on your host must be configured to use serial console. To verify, check that the **/etc/default/grub** file on your host contains the **GRUB_TERMINAL=serial** parameter.

```
$ sudo grep GRUB_TERMINAL /etc/default/grub
GRUB_TERMINAL=serial
```

- The VM must have a serial console device configured, such as **console type='pty'**. To verify, do the following:

```
# virsh dumpxml vm-name | grep console

<console type='pty' tty='/dev/pts/2'>
</console>
```

- The VM must have the serial console configured in its kernel command line. To verify this, the **cat /proc/cmdline** command output on the VM should include *console=<console-name>*, where *<console-name>* is architecture-specific:

- For AMD64 and Intel 64: **ttyS0**
- For ARM 64: **ttyAMA0**

  NOTE

  The following commands in this procedure use **ttyS0**.

  ```
  # cat /proc/cmdline
  BOOT_IMAGE=/vmlinuz-3.10.0-948.el7.x86_64 root=/dev/mapper/rhel-root ro
  console=tty0 console=ttyS0,9600n8 rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb
  ```

  If the serial console is not set up properly on a VM, using **virsh console** to connect to the VM connects you to an unresponsive guest console. However, you can still exit the unresponsive console by using the **Ctrl+]** shortcut.

- To set up serial console on the VM, do the following:

  i. On the VM, enable the **console=ttyS0** kernel option:

  ```
  # grubby --update-kernel=ALL --args="console=ttyS0"
  ```

  ii. Clear the kernel options that might prevent your changes from taking effect.

  ```
  # grub2-editenv - unset kernelopts
  ```

  iii. Reboot the VM.

- The **serial-getty@*<console-name>*** service must be enabled. For example, on AMD64 and Intel 64:

  ```
  # systemctl status serial-getty@ttyS0.service

  ○ serial-getty@ttyS0.service - Serial Getty on ttyS0
     Loaded: loaded (/usr/lib/systemd/system/serial-getty@.service; enabled; preset: enabled)
  ```

Procedure

1. On your host system, use the **virsh console** command. The following example connects to the *guest1* VM, if the libvirt driver supports safe console handling:

   ```
   # virsh console guest1 --safe
   Connected to domain 'guest1'
   Escape character is ^]

   Subscription-name
   Kernel 3.10.0-948.el7.x86_64 on an x86_64

   localhost login:
   ```

2. You can interact with the virsh console in the same way as with a standard command-line interface.

Additional resources

- **virsh** man page on your system
- Configuring Serial Console Logs on a VM (video)

5.4. CONFIGURING VNC PASSWORDS

To manage access to the graphical output of a virtual machine (VM), you can configure a password for the VNC console of the VM.

With a VNC password configured on a VM, users of the VMs must enter the password when attempting to view or interact with the VNC graphical console of the VMs, for example by using the **virt-viewer** utility.

IMPORTANT

VNC passwords are not a sufficient measure for ensuring the security of a VM environment. For details, see QEMU documentation on VNC security.

In addition, the VNC password is saved in plain text in the configuration of the VM, so for the password to be effective, the user must not be able to display the VM configuration.

Prerequisites

- The VM that you want to protect with a VNC password has VNC graphics configured.
  To ensure that this is the case, use the **virsh dumpxml** command as follows:

```
# virsh dumpxml <vm-name> | grep graphics

 <graphics type='vnc' ports='-1' autoport=yes listen=127.0.0.1>
 </graphics>
```

Procedure

1. Open the configuration of the VM that you want to assign a VNC password to.

```
# virsh edit <vm-name>
```

2. On the **<graphics>** line of the configuration, add the **passwd** attribute and the password string. The password must be 8 characters or fewer.

```
 <graphics type='vnc' ports='-1' autoport=yes listen=127.0.0.1 passwd='<password>'>
```

   - Optional: In addition, define a date and time when the password will expire.

```
 <graphics type='vnc' ports='-1' autoport=yes listen=127.0.0.1 passwd='<password>'
passwdValidTo='2025-02-01T15:30:00'>
```

   In this example, the password will expire on February 1st 2025, at 15:30 UTC.

3. Save the configuration.

Verification

1. Start the modified VM.

   ```
   # virsh start <vm-name>
   ```

2. Open a graphical console of the VM, for example by using the **virt-viewer** utility:

   ```
   # virt-viewer <vm-name>
   ```

   If the VNC password has been configured properly, a dialogue window appears that requests you to enter the password.

CHAPTER 6. DELETING VIRTUAL MACHINES

To delete virtual machines in RHEL 10, use the command line interface or the web console GUI.

6.1. DELETING VIRTUAL MACHINES BY USING THE COMMAND LINE

To delete a virtual machine (VM), you can remove its XML configuration and associated storage files from the host by using the command line.

Prerequisites

- Back up important data from the VM.
- Shut down the VM.
- Make sure no other VMs use the same associated storage.

Procedure

- Use the **virsh undefine** utility.
  For example, the following command removes the *guest1* VM, its associated storage volumes, and non-volatile RAM, if any.

```
# virsh undefine guest1 --remove-all-storage --nvram
Domain 'guest1' has been undefined
Volume 'vda'(/home/images/guest1.qcow2) removed.
```

Additional resources

- **virsh undefine --help** command
- **virsh** man page on your system

6.2. DELETING VIRTUAL MACHINES BY USING THE WEB CONSOLE

You can delete a virtual machine (VM) and its associated storage files from the host by using the RHEL 10 web console.

Prerequisites

- You have installed the RHEL 10 web console.
  For instructions, see Installing and enabling the web console .
- The web console VM plug-in is installed on your system .
- Back up important data from the VM.
- Make sure no other VM uses the same associated storage.
- Optional: Shut down the VM.

Procedure

1. Log in to the RHEL 10 web console.
   For details, see Logging in to the web console .
2. In the **Virtual Machines** interface, click the Menu button ⋮ of the VM that you want to delete.
   A drop down menu appears with controls for various VM operations.
3. Click **Delete**.
   A confirmation dialog appears.
4. Optional: To delete all or some of the storage files associated with the VM, select the checkboxes next to the storage files you want to delete.
5. Click **Delete**.
   The VM and any selected storage files are deleted.

CHAPTER 7. VIEWING INFORMATION ABOUT VIRTUAL MACHINES

When you need to adjust or troubleshoot any aspect of your virtualization deployment on RHEL 10, the first step you need to perform usually is to view information about the current state and configuration of your virtual machines (VMs). To do so, you can use the command line.

7.1. VIEWING VIRTUAL MACHINE INFORMATION BY USING THE COMMAND LINE

To retrieve information about virtual machines (VMs) on your host and their configurations, you can use the **virsh** command-line utility.

Procedure

- To obtain a list of VMs on your host:

```
# virsh list --all
Id   Name              State
----------------------------------
1    testguest1             running
-    testguest2            shut off
-    testguest3            shut off
-    testguest4            shut off
```

- To obtain basic information about a specific VM:

```
# virsh dominfo testguest1
Id:             1
Name:           testguest1
UUID:           a973666f-2f6e-415a-8949-75a7a98569e1
OS Type:        hvm
State:          running
CPU(s):         2
CPU time:       188.3s
Max memory:     4194304 KiB
Used memory:    4194304 KiB
Persistent:     yes
Autostart:      disable
Managed save:   no
Security model: selinux
Security DOI:   0
Security label: system_u:system_r:svirt_t:s0:c486,c538 (enforcing)
```

- To obtain the complete XML configuration of a specific VM:

```
# virsh dumpxml testguest2

<domain type='kvm' id='1'>
  <name>testguest2</name>
  <uuid>a973434f-2f6e-4ěša-8949-76a7a98569e1</uuid>
  <metadata>
[...]
```

- For information about a VM's disks and other block devices:

```
# virsh domblklist testguest3
 Target   Source
---------------------------------------------------------------
 vda      /var/lib/libvirt/images/testguest3.qcow2
 sda      -
 sdb      /home/username/Downloads/virt-p2v-1.36.10-1.el7.iso
```

- To obtain information about a VM's file systems and their mountpoints:

```
# virsh domfsinfo testguest3
Mountpoint   Name   Type   Target
------------------------------------
 /           dm-0   xfs
 /boot       vda1   xfs
```

- To obtain more details about the vCPUs of a specific VM:

```
# virsh vcpuinfo testguest4
VCPU:           0
CPU:            3
State:          running
CPU time:       103.1s
CPU Affinity:   yyyy

VCPU:           1
CPU:            0
State:          running
CPU time:       88.6s
CPU Affinity:   yyyy
```

- To list all virtual network interfaces on your host:

```
# virsh net-list --all
 Name       State    Autostart   Persistent
---------------------------------------------
 default    active   yes         yes
 labnet     active   yes         yes
```

For information about a specific interface:

```
# virsh net-info default
Name:           default
UUID:           c699f9f6-9202-4ca8-91d0-6b8cb9024116
Active:         yes
Persistent:     yes
Autostart:      yes
Bridge:         virbr0
```

CHAPTER 8. CLONING VIRTUAL MACHINES

To quickly create a new virtual machine (VM) with a specific set of properties, you can *clone* an existing VM.

Cloning creates a new VM that uses its own disk image for storage, but most of the clone's configuration and stored data is identical to the source VM. This makes it possible to prepare multiple VMs optimized for a certain task without the need to optimize each VM individually.

8.1. HOW CLONING VIRTUAL MACHINES WORKS

Cloning a virtual machine (VM) copies the XML configuration of the source VM and its disk images, and makes adjustments to the configurations to ensure the uniqueness of the new VM. This includes changing the name of the VM and ensuring it uses the disk image clones. Nevertheless, the data stored on the clone's virtual disks is identical to the source VM.

This process is faster than creating a new VM and installing it with a guest operating system, and can be used to rapidly generate VMs with a specific configuration and content.

If you are planning to create multiple clones of a VM, first create a VM *template* that does not contain:

- Unique settings, such as persistent network MAC configuration, which can prevent the clones from working correctly.
- Sensitive data, such as SSH keys and password files.

For instructions, see Creating virtual machines templates.

Additional resources

- Cloning a virtual machine by using the command line
- Cloning a virtual machine by using the web console

8.2. CREATING VIRTUAL MACHINE TEMPLATES

To create multiple virtual machine (VM) clones that work correctly, you can remove information and configurations that are unique to a source VM, such as SSH keys or persistent network MAC configuration. This creates a VM *template*, which you can use to easily and safely create VM clones.

You can create VM templates by using the **virt-sysprep** utility or you can create them manually based on your requirements.

8.2.1. Creating a virtual machine template by using virt-sysprep

To create a cloning template from an existing virtual machine (VM), you can use the **virt-sysprep** utility. This removes certain configurations that might cause the clone to work incorrectly, such as specific network settings or system registration metadata. As a result, **virt-sysprep** makes creating clones of the VM more efficient, and ensures that the clones work more reliably.

Prerequisites

- The **guestfs-tools** package, which contains the **virt-sysprep** utility, is installed on your host:

```
# dnf install guestfs-tools
```

- The source VM intended as a template is shut down.
- You know where the disk image for the source VM is located, and you are the owner of the VM's disk image file.
  Note that disk images for VMs created in the system connection of libvirt are located in the **/var/lib/libvirt/images** directory and owned by the root user by default:

```
# ls -la /var/lib/libvirt/images
-rw-------.  1 root root  9665380352 Jul 23 14:50 a-really-important-vm.qcow2
-rw-------.  1 root root  8591507456 Jul 26  2017 an-actual-vm-that-i-use.qcow2
-rw-------.  1 root root  8591507456 Jul 26  2017 totally-not-a-fake-vm.qcow2
-rw-------.  1 root root 10739318784 Sep 20 17:57 another-vm-example.qcow2
```

- Optional: Any important data on the source VM's disk has been backed up. If you want to preserve the source VM intact, clone it first and turn the clone into a template.

Procedure

1. Ensure you are logged in as the owner of the VM's disk image:

```
# whoami
root
```

2. Optional: Copy the disk image of the VM.

```
# cp /var/lib/libvirt/images/a-really-important-vm.qcow2 /var/lib/libvirt/images/a-really-important-vm-original.qcow2
```

This is used later to verify that the VM was successfully turned into a template.

3. Use the following command, and replace */var/lib/libvirt/images/a-really-important-vm.qcow2* with the path to the disk image of the source VM.

```
# virt-sysprep -a /var/lib/libvirt/images/a-really-important-vm.qcow2
[   0.0] Examining the guest ...
[   7.3] Performing "abrt-data" ...
[   7.3] Performing "backup-files" ...
[   9.6] Performing "bash-history" ...
[   9.6] Performing "blkid-tab" ...
[...]
```

Verification

- To confirm that the process was successful, compare the modified disk image to the original one. The following example shows a successful creation of a template:

```
# virt-diff -a /var/lib/libvirt/images/a-really-important-vm-orig.qcow2 -A /var/lib/libvirt/images/a-really-important-vm.qcow2
- - 0644       1001 /etc/group-
- - 0000        797 /etc/gshadow-
= - 0444         33 /etc/machine-id
```

```
[...]
- - 0600       409 /home/username/.bash_history
- d 0700         6 /home/username/.ssh
- - 0600       868 /root/.bash_history
[...]
```

Additional resources

- The **OPERATIONS** section in the **virt-sysprep** man page on your system
- Cloning a virtual machine by using the command line

8.2.2. Creating a virtual machine template manually

To create a template from an existing virtual machine (VM), you can manually reset or unconfigure a guest VM to prepare it for cloning.

Prerequisites

- Ensure that you know the location of the disk image for the source VM and are the owner of the VM's disk image file.
  Note that disk images for VMs created in the system connection of libvirt are by default located in the **/var/lib/libvirt/images** directory and owned by the root user:

  ```
  # ls -la /var/lib/libvirt/images
  -rw-------.  1 root root  9665380352 Jul 23 14:50 a-really-important-vm.qcow2
  -rw-------.  1 root root  8591507456 Jul 26  2017 an-actual-vm-that-i-use.qcow2
  -rw-------.  1 root root  8591507456 Jul 26  2017 totally-not-a-fake-vm.qcow2
  -rw-------.  1 root root 10739318784 Sep 20 17:57 another-vm-example.qcow2
  ```

- Ensure that the VM is shut down.
- Optional: Any important data on the VM's disk has been backed up. If you want to preserve the source VM intact, clone it first and edit the clone to create a template.

Procedure

1. Configure the VM for cloning:
   a. Install any software needed on the clone.
   b. Configure any non-unique settings for the operating system.
   c. Configure any non-unique application settings.
2. Remove the network configuration:
   a. Remove any persistent udev rules by using the following command:

      ```
      # rm -f /etc/udev/rules.d/70-persistent-net.rules
      ```

NOTE

If udev rules are not removed, the name of the first NIC might be **eth1** instead of **eth0**.

b. Remove unique information from the **NMConnection** files in the **/etc/NetworkManager/system-connections/** directory.

i. Remove MAC address, IP address, DNS, gateway, and any other **unique** information or non-desired settings.

```
*ID=ExampleNetwork
BOOTPROTO="dhcp"
HWADDR="AA:BB:CC:DD:EE:FF"                <- REMOVE
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
UUID="954bd22c-f96c-4b59-9445-b39dd86ac8ab" <- REMOVE
```

ii. Remove similar **unique** information and non-desired settings from the **/etc/hosts** and **/etc/resolv.conf** files.

3. Remove registration details:

- For VMs registered on the Red Hat Network (RHN):

```
# rm /etc/sysconfig/rhn/systemid
```

- For VMs registered with Red Hat Subscription Manager (RHSM):
  - If you do not plan to use the original VM:

```
# subscription-manager unsubscribe --all # subscription-manager unregister # subscription-manager clean
```

  - If you plan to use the original VM:

```
# subscription-manager clean
```

NOTE

The original RHSM profile remains in the Portal along with your ID code. Use the following command to reactivate your RHSM registration on the VM after it is cloned:

```
# subscription-manager register --consumerid=71rd64fx-6216-4409-bf3a-e4b7c7bd8ac9
```

4. Remove other unique details:

a. Remove SSH public and private key pairs:

```
# rm -rf /etc/ssh/ssh_host_example
```

b. Remove the configuration of LVM devices:

```
# rm /etc/lvm/devices/system.devices
```

c. Remove any other application-specific identifiers or configurations that might cause conflicts if running on multiple machines.

5. Remove the **gnome-initial-setup-done** file to configure the VM to run the configuration wizard on the next boot:

```
# rm ~/.config/gnome-initial-setup-done
```

NOTE

The wizard that runs on the next boot depends on the configurations that have been removed from the VM. In addition, on the first boot of the clone, it is recommended that you change the hostname.

8.3. CLONING A VIRTUAL MACHINE BY USING THE COMMAND LINE

For testing, to create a new virtual machine (VM) with a specific set of properties, you can clone an existing VM by using the command line.

Prerequisites

- The source VM is shut down.
- Ensure that there is sufficient disk space to store the cloned disk images.
- Optional: When creating multiple VM clones, remove unique data and settings from the source VM to ensure the cloned VMs work properly. For instructions, see Creating virtual machine templates.

Procedure

- Use the **virt-clone** utility with options that are appropriate for your environment and use case.
  Sample use cases
  - The following command clones a local VM named **example-VM-1** and creates the **example-VM-1-clone** VM. It also creates and allocates the **example-VM-1-clone.qcow2** disk image in the same location as the disk image of the original VM, and with the same data:

```
# virt-clone --original example-VM-1 --auto-clone
Allocating 'example-VM-1-clone.qcow2'                    | 50.0 GB  00:05:37

Clone 'example-VM-1-clone' created successfully.
```

  - The following command clones a VM named **example-VM-2**, and creates a local VM named **example-VM-3**, which uses only two out of multiple disks of **example-VM-2**:

```
# virt-clone --original example-VM-2 --name example-VM-3 --file
/var/lib/libvirt/images/disk-1-example-VM-2.qcow2 --file /var/lib/libvirt/images/disk-2-
example-VM-2.qcow2
```

```
Allocating 'disk-1-example-VM-2-clone.qcow2'                    | 78.0 GB  00:05:37
Allocating 'disk-2-example-VM-2-clone.qcow2'                    | 80.0 GB  00:05:37

Clone 'example-VM-3' created successfully.
```

- To clone your VM to a different host, migrate the VM without undefining it on the local host. For example, the following commands clone the previously created **example-VM-3** VM to the **192.0.2.1** remote system, including its local disks. Note that you require root privileges to run these commands for **192.0.2.1**:

```
# virsh migrate --offline --persistent example-VM-3 qemu+ssh://root@192.0.2.1/system
root@192.0.2.1's password:

# scp /var/lib/libvirt/images/<disk-1-example-VM-2-clone>.qcow2
root@192.0.2.1/<user@remote_host.com>://var/lib/libvirt/images/

# scp /var/lib/libvirt/images/<disk-2-example-VM-2-clone>.qcow2
root@192.0.2.1/<user@remote_host.com>://var/lib/libvirt/images/
```

Verification

1. To verify the VM has been successfully cloned and is working correctly:
   a. Confirm the clone has been added to the list of VMs on your host:

```
# virsh list --all
Id   Name                  State
---------------------------------------
-    example-VM-1          shut off
-    example-VM-1-clone    shut off
```

   b. Start the clone and observe if it boots up:

```
# virsh start example-VM-1-clone
Domain 'example-VM-1-clone' started
```

Additional resources

- **virt-clone (1)** man page on your system

8.4. CLONING A VIRTUAL MACHINE BY USING THE WEB CONSOLE

To create new virtual machines (VMs) with a specific set of properties, you can clone a VM that you had previously configured by using the web console.

NOTE

Cloning a VM also clones the disks associated with that VM.

Prerequisites

- You have installed the RHEL 10 web console.
  For instructions, see Installing and enabling the web console .

- The web console VM plug-in is installed on your system.
- Ensure that the VM you want to clone is shut down.

Procedure

1. Log in to the RHEL 10 web console.
   For details, see Logging in to the web console .
2. In the Virtual Machines interface of the web console, click the Menu button ⋮ of the VM that you want to clone.
   A drop down menu appears with controls for various VM operations.
3. Click **Clone**.
   The Create a clone VM dialog appears.
4. Optional: Enter a new name for the VM clone.
5. Click **Clone**.
   A new VM is created based on the source VM.

Verification

- Confirm whether the cloned VM appears in the list of VMs available on your host.