Lab 5:

Virtualization with   
Kernel-based Virtual Machines

Datacenter Virtualization

2024-2025

© Daan Pareit

CONTENT

[1 Introduction 3](#_Toc182994171)

[1.1 Learning goals 4](#_Toc182994172)

[1.1.1 Knowledge 4](#_Toc182994173)

[1.1.2 Skills 4](#_Toc182994174)

[1.2 Prerequisites 4](#_Toc182994175)

[2 Installing Red Hat Enterprise Linux (RHEL) 6](#_Toc182994176)

[2.1 Preface 6](#_Toc182994177)

[2.2 Preparing your OS for nested virtualization 6](#_Toc182994178)

[2.3 Creating the VM to emulate the hypervisor 6](#_Toc182994179)

[2.4 Installation 8](#_Toc182994180)

[3 Using KVM 12](#_Toc182994181)

[3.1 Configuration 12](#_Toc182994182)

[3.2 Creating and exploring VMs via CLI 14](#_Toc182994183)

[3.2.1 As a regular user 15](#_Toc182994184)

[3.2.2 As root/sudo 18](#_Toc182994185)

[3.3 Managing VMs via software: virt-manager 24](#_Toc182994186)

[3.4 Managing VMs Via a web interface: cockpit 28](#_Toc182994187)

# 

# Introduction

***Kernel-based Virtual Machine (KVM)*** has existed for a pretty long time, as it has been part of the Linux Kernel since version 2.6.20, released in February 2007. Ubuntu supports it, including management tools, since version 12.04.



Architecture wise, KVM works again differently than Hyper-V/XEN (which are similar) and ESXi (with its monolithic design).

KVM (as opposed to Hyper-V) is a virtualization layer that is built into the Linux Kernel. Through time, the line between ESXi (also a virtualization enabled kernel) and KVM has grown thinner. But, of course, the main advantage for KVM over ESXi is its open-source nature. This usually means that end users can adjust, add new features more quickly, have more supported hardware and fix bugs themselves.

## Learning goals

### Knowledge

* Know how Red Hat management works
* Know what KVM is and how it works

### Skills

* Be able to install & manage Red Hat Enterprise Linux (RHEL)
* Be able to install & manage KVM VMs

## Prerequisites

This lab will be performed on VMware Workstation Pro or VMware Fusion Pro. See earlier lab on how to install this software.

You’ll need the following (all/most of this you should already have from previous labs or other course):

* A Red Hat developer account
  + Red Hat Enterprise Linux is not free, but it is open source. This means it is legal to download and develop the operating system, but not legal to use it in production without a (paid) subscription.
  + However, developer licenses are free. You’ll also have to obtain such a free developer license:
    - Open <https://developers.redhat.com> and **Create a Red Hat account**, remember your username and password, as it is needed during the installation.
    - Verify your account via the confirmation e-mail. Note: when using your @student.howest.be e-mail address; you’ll probably find the confirmation mail in your spam folder 😊.
    - Finally, surf to <https://www.redhat.com/wapps/sso/login.html> to complete your registration: choose a subscription that matches ‘personal’ or “Red Hat Developer” and complete the required information.
    - It could take a while (> 15 min, less than 12 hours) before your developer subscription is linked to your account. Verify you have an active subscription at <https://console.redhat.com/subscriptions/inventory>
* Download the ‘Boot ISO’-file of Red Hat Enterprise Linux to a folder on your laptop.   
  We need the **rhel-9.<x>-x86\_64-boot.iso** file which is available here:   
  <https://developers.redhat.com/products/rhel/download/>
* To test a low-resource-requiring VM running with KVM, we will boot into a simple Operating System. We’ll use a PuppyLinux distro “VoidPup64”, for which the ISO file can be found on SourceForge: <https://sourceforge.net/projects/pb-gh-releases/files/latest/download>
* We’ll use a tool like MobaXterm (shareware) or SmarTTY (free) to have an X server running.

# Installing Red Hat Enterprise Linux (RHEL)

## Preface

We will now install a VM with “Red Hat Enterprise Server” to serve as the hypervisor for running VMs with KVM. This should only take a few minutes.

## Preparing your OS for nested virtualization

A hypervisor’s purpose is to run virtual machines. But if you install the hypervisor itself within a virtual machine in your VMware Workstation/Fusion, the VMs running on the hypervisor will be VMs running inside your virtual machine. This is what we call **nested virtualization** and the VMs running inside the hypervisor in your VM, are sometimes called **vVMs** (virtual VMs).

If you already have an active virtualization layer on your host OS, running vVMs on your RHEL VM would add two more layers of virtualization, thus three in total. That would be too complex to handle and your VM will fail to run.

This is most notably the case if you are running Windows 11 with Hyper-V enabled on your host (required for e.g. WSL2). In that case:

* You need to (temporarily) disable Hyper-V on your Windows 11: see ‘hyperv-vs-vmware.txt’ on Leho
* If you get the warning that Intel VT-x/EPT is not supported on your platform, it might be due to Credential Guard. It can be disabled using this Microsoft PowerShell script: <https://www.microsoft.com/en-us/download/details.aspx?id=53337>

## Creating the VM to emulate the hypervisor

If you are enrolled in the “Linux Server Security” course, you already have a RHEL VM. In that case you have the choice between:

* Or use a (linked) clone of that VM for this lab and:
  + Make sure the network settings are OK, i.e. set the IP address and routing table correct to match the correct vmnet or have your pfSense up and running for Internet connectivity.
  + Enable the “Intel VT-x / AMD-V” option for the processor.
  + Proceed to section 3 “Using KVM”.
* Or, of course, you are free to create another one from scratch as well as described below.

If you’re not enrolled in the aforementioned course, then create a new VM for your RHEL:

1. Create a new VM within VMware Workstation, make sure to select “Red Hat Enterprise Linux 9 64-bit”.
2. Choose ‘I will install the OS later’ to avoid VMware’s ‘Easy Install’ to kick in.
3. Choose a name and location for the VM files. Preferably this location should not be synchronized with the cloud (onedrive) to avoid instability.
4. Give a bit more than 20GB as its hard drive. Store the virtual disk as a single file.
5. Customize hardware to **enable the “Intel VT-x / AMD-V” option**, just as you did for VMware ESXi or Microsoft Hyper-V.

## Installation

Now, let’s install RHEL:

1. Attach the ISO and start the installation: select the first option (**Install Red Hat Enterprise Linux 9.x**).
2. Use these settings as installation options:
   1. Use English (US) as its language
   2. Select your *Keyboard* (for Azerty use “French (Belgian)”)
   3. Select the time zone in *Time & Date*
   4. Click on *Network & Host Name*
      1. Switch the Ethernet button to **“on”** (should be ok by default)
      2. Change hostname to **rhel-<firstname>-<lastname>** and click ***Apply*** and then ***Done.*** This is important to verify your individual assignments later! Labs will be zero graded without personalized prompts.
   5. Click on *Connect to Red Hat* and enter the credentials from your free Red Hat Developers account. Disable ‘Red Hat Insights’ and click on ‘Register’.

If successful, you should read ‘The system is registered’ on top and ‘1 subscription attached to the system’ or ‘subscribed in Simple Content Access mode’ at the bottom.

In case of authentication failure:

* + 1. Verify your password by typing it in the username field, to detect a possible wrong keyboard layout.
    2. Verify you have a developer subscription linked to your account at <https://console.redhat.com/subscriptions/inventory>
    3. Still no luck? You still need a valid developer subscription as described above, but try this alternative method instead:
       1. Download the ‘DVD iso’ (10 GB) instead of the ‘Boot iso’ (1 GB) from <https://developers.redhat.com/products/rhel/download> . Note: you’ll have to scroll way down to the ‘All Downloads’ section. Make sure you download the ‘rhel-9.4-x86\_64-dvd.iso’ file.
       2. In the CD/DVD settings menu of your VM in VMware Workstation, change to this iso file and reboot the VM. Follow the previous steps again but skip the ‘Connect to Red Hat’ section for now.
       3. Continue with the next steps after installation to activate:
       4. Go to <https://console.redhat.com/insights/connector/activation-keys>
       5. Below the ‘Activation Keys’ title, you read an ‘Organization ID’. Take note of this number, you’ll need it in a minute.
       6. Now, create an activation key
          1. Choose a name and remember that one, you’ll need it as well.
          2. Select ‘Latest release’ as workload
          3. As System Purpose, select ‘RHEL Server’ role, ‘Self Support’ Service Level and ‘Development/Test’ usage
       7. Now, in your RHEL, as root user, execute following commands:
          1. subscription-manager register --activationkey=<your key name> --org=<your organization number>
          2. subscription-manager attach --auto
          3. subscription-manager status
       8. This suffices for activation. However, if you also want your RHEL to be connected to ‘Red Hat Insights’ for remote monitoring through their web interface, than you can also optionally run the following command:
          1. insight-client --register
  1. Click *Installation Destination* and accept the proposed automatic partitioning by clicking ‘Done’ at the top
  2. Select *Software Selection* and select “**Minimal Install**”
  3. Configure the *Root Password*
  4. To work in a proper way, also create a regular user account (and enable it to be an administrator)
  5. Lastly, *Begin Installation*

RHEL will start installing, a minimal install only downloads about 450MB of extra packages

# Using KVM

## Configuration

KVM itself is part of the Kernel, so it does not have to be installed. However, the tools required to manage KVM must be installed:

1. Installing KVM management
   1. Use this command to install all required management libraries and tools  
      *sudo dnf install qemu-kvm libvirt virt-install virt-viewer*
   2. Multiple systemd units have been created. To start these virtualization services, use:

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

*A computer screen with white text

Description automatically generated*

1. With *lsmod* you’ll see the different modules which are loaded into the kernel. Notice that “kvm” and “kvm\_intel” or “kvm\_amd” are indeed loaded to make use of the hardware supported Intel VT-x/AMD-V for binary CPU translation which our future VMs will be able to use.

A screen shot of a computer code

Description automatically generated

1. Execute the *virt-host-validate* command. Most importantly the “Checking for hardware virtualization” should PASS. You might get warning about other features, but these are less important here.

A screenshot of a computer program

Description automatically generated

1. We want to graphically connect to the screens of our future VMs -which we’ll create in a minute with KVM. However, we don’t have a desktop installed on RHEL. That’s why we’ll use SSH X-forwarding:
   1. Install X11 related packages and a graphical terminal ‘xterm’ as small graphical program to verify our setup:

*sudo dnf install xorg-x11-xauth xterm*

* 1. In your /etc/ssh/sshd\_config file, make sure X11 Forwarding is enabled and restart the sshd systemd service.

A computer screen shot of text

Description automatically generated

* 1. Use tools like MobaXterm (shareware) or SmarTTY (free) to have an X server running when logging in over SSH to RHEL.  
     From Linux guests the “-X” parameter is required with the *ssh* command.
  2. Verify X forwarding works by connecting with the host and running the command:  
     *xterm*🡪 This command should result in a graphical terminal being shown

A screenshot of a computer

Description automatically generated

* 1. Running ‘*sudo xterm*’ however fails to X forward out of the box. To make sudo commands work with X forwarding, use the following:

A screenshot of a computer screen

Description automatically generated

It worked hahahaha

*sudo xauth add $(xauth -f .Xauthority list)*

Note: you’ll get confusingly a warning ‘xauth: file /root/.Xauthority does not exist’, but that means that it is created afterwards.

Note: when starting a new SSH session, you’ll have to execute this again to update the authentication for sudo.

A black screen with white text

Description automatically generated

Now, running ‘*sudo xterm*’ should work, as any other sudo command with X forwarding.

1. We’ll need an ISO file to install a new VM, therefore upload the ISO file of the VoidPup distro to the /var folder in RHEL, e.g via ‘scp’.

A black screen with blue and white text

Description automatically generated

## Creating and exploring VMs via CLI

### As a regular user

We’ll create a VM via CLI:

1. With ***virt-install***, you can create a VM. We’ll first create a Puppy Linux to run from the live ISO “CD”. No disk space is assigned to this VM, so changes made during the session will not be preserved. Run this as regular user without sudo in a terminal with X forwarding:

*virt-install --name puppy --memory 512 --vcpus 1 --disk none --livecd --osinfo voidlinux --cdrom /var/VoidPup64-22.02-241009.iso*

A screenshot of a computer

Description automatically generated

1. You should get the Puppy Linux graphical console now via ***virt-viewer*** in a separate window via X forwarding. If you would close it, you can always regain access by executing:

*virt-viewer puppy*

Remember from earlier courses: if you want your prompt back and still have your virt-viewer active, you can suspend it (ctrl+z) and make it run in the background (bg). If you want it back in the foreground again, use ‘jobs’ or ‘fg’

**That’s nice to know, love how I can have virtual machines with virtual machines.**

Let’s explore the command line KVM manager ***virsh*** :

1. Type “*virsh*” to enter the “Virtual Shell” and manage VM’s.  
    Type in the commands below and take note if their purpose
   1. virsh # *help*
   2. virsh # *list*

A close-up of a list

Description automatically generated

* 1. virsh # *dominfo 1*

A screenshot of a computer program

Description automatically generated

**Q: What is the value for the “security model”? (this should ring a bell if you have/had the course ‘Linux Server Security’)**

**Selinux.**

* 1. virsh # *suspend 1*
  2. virsh # *list*

A black and white text

Description automatically generated

* 1. virsh # *resume 1*
  2. virsh # *reboot 1*

A computer screen shot of a computer code

Description automatically generated

* 1. virsh # *reset 1***Q: What is the difference between ‘reboot’ and ‘reset’ if you try these commands several times?**

Reset – power off

Reboot – just restart of the machine

* 1. virsh # *destroy 1***Q: execute a “list” command again but now with ‘--all', is it really gone?**

A computer screen shot of a computer program

Description automatically generated

This thing is just shutting it down (at least it looks like it).

* 1. To restart the VM:
     1. virsh # *start puppy*
     2. exit virsh and have access to the VM again with: *virt-viewer puppy*

A screenshot of a computer

Description automatically generated

* 1. If you need to completely remove a VM, you’ll need one more command:
     1. virsh # *destroy puppy*
     2. virsh # *undefine puppy*

A screenshot of a computer program

Description automatically generated

Let’s explore now the network the VM has:

1. If you still have your VM, start it with ***virsh*** or recreate it with ***virt-install*** (still without sudo)
2. In Puppy Linux, open a terminal and find the IP address of this virtual machine.

**Q: What is its IP address? Does the VM have Internet connectivity (e.g. ping 8.8.8.8)?**

**A screenshot of a computer screen

Description automatically generated**

**Yes, it has an IP address and it can ping.**

1. Check the IP address of your RHEL host.

**Q: Does it have an IP address in the same range as your Puppy Linux?**

**A screenshot of a computer program

Description automatically generated**

**Does not seem like it.**

1. Now try pinging 10.0.2.2 on your Puppy Linux, that should succeed.

A screenshot of a computer

Description automatically generated

It works, but no clue way.

1. We’ll prove that 10.0.2.2 is the address of your RHEL even though not listed in your IP settings:
   1. Start ‘*ncat -l 12345*’ on your RHEL to listen on port 12345
   2. On Puppy, start a telnet to this port for IP address 10.0.2.2: ‘*telnet 10.0.2.2 12345*’
   3. Everything you type in Puppy is now shown on RHEL, thus 10.0.2.2 must be used by RHEL!

A screenshot of a computer

Description automatically generated



The reason why this work the way it does while we don’t see any related network settings on RHEL, is called “User Networking (SLIRP)”. It is the default networking backend and generally is the easiest to use, providing a virtual NAT network. It does not require root privileges. However, it has poor performance, and the guest is not directly accessible from the host or the external network. (See https://wiki.qemu.org/Documentation/Networking )

A diagram of a network

Description automatically generated

### As root/sudo

Now let’s create the same VM with sudo and focus on how we have more elaborate networking:

1. With ***virsh*** destroy and undefine any existing VM for/as your regular user.

A screenshot of a computer

Description automatically generated

1. Become root or -even better- always use sudo but make sure your X forwarding works for sudo (cfr xauth command earlier)
2. No, create the same Puppy Linux but now as sudo or root with ***virt-install***

*sudo virt-install --name puppy --memory 512 --vcpus 1 --disk none --livecd --osinfo voidlinux --cdrom /var/VoidPup64-22.02-241009.iso*

1. In Puppy Linux, open a terminal and find the IP address of this virtual machine.

**Q: What is its IP address? Does the VM have Internet connectivity (e.g. ping 8.8.8.8)?**

**It has an IP address.**

**A screen shot of a computer

Description automatically generated**

**A computer screen with white text

Description automatically generated**

**It can also ping.**

1. We can also obtain this network information of the VM via virsh (as root/sudo) on RHEL:
   1. virsh # *domiflist puppy*

A close-up of a diagram

Description automatically generated

* 1. virsh # *domifaddr puppy*

A close-up of a line

Description automatically generated

1. Now, check the IP address of your RHEL host.

**Q: Does it have an IP address in the same range as your Puppy Linux? Which one? Which network interface(s) are used to accomplish this?**

**A screenshot of a computer

Description automatically generated**

**Yes, now it has one. I thought I did something wrong in the first case. The interface virbr0 seems to be the one used.**

1. Info about the virtual switch/network can be obtained and managed with:
   1. virsh # *net-list –all*

A black text on a white background

Description automatically generated

* 1. virsh # *net-edit default*

A screenshot of a computer code

Description automatically generated

1. Note that on RHEL you’ll also see how this virtual networking is dealt with:
   1. There is a systemd service for the libvirt network, check this:

*sudo systemctl status virtnetworkd*

*A screenshot of a computer program

Description automatically generated*

**Q: What binaries are started/running to hand out the IP addresses for the virtual network?**

**Dnsmaq, and virnetworkd**

**Q: What config file is this binary using? Verify it matches the settings of “virsh net-edit default”**

**A screenshot of a computer

Description automatically generated**

**It does look like the configuration we had**

* 1. There are firewall tables/rules created in nftables (again familiar if you have/had the course Linux Server Security 😉) to implement the NAT:

*nft list tables*

*iptables -L -t nat*

Note: these tables are only present when a VM is running

A screenshot of a computer

Description automatically generated

**Q: what is the target/keyword in these rules for NATing?**

**MASQUERADE**

## Managing VMs via software: virt-manager

There is a more elaborate KVM management software called ***virt-manager***.

1. Install the graphical “virt-manager” tool via dnf.
2. After the installation, from an X forwarding enabled terminal, run virt-manager (with sudo or as root)

A screenshot of a computer

Description automatically generated

1. You can consider the virt-manager running on KVM/QEMU as an open-source alternative to your VMware Workstation/Fusion. Explore the Virtual Machine Manager environment:
   1. Click on Edit > Preferences   
      **Q: Can you find the key combo to release the mouse cursor?**

A screenshot of a computer

Description automatically generated

Ctrl+Alt

* 1. Select a VM in the list. Double click will show the VM’s screen.

A screenshot of a computer

Description automatically generated

1. Let’s create a new VM with virt-manager. And to do things a little bit different than the ordinary way with a local ISO file you first have to download (as you’ve done zillions of time already), we’ll use an online image to install from this time.
   1. Click the  icon to open the wizard to create a new VM, we will create one for Debian
   2. Select “Network Install (HTTP)” as the source since we do not want to use a local ISO file.
   3. As an example for Debian (but we can actually enter any other URL that’s usable):  
      <https://deb.debian.org/debian/dists/stable/main/installer-amd64/> 🡪Auto detection of the Guest OS might fail, so manually look for “Debian 12”

*A screenshot of a computer

Description automatically generated*

* 1. Just click forward. Maybe change the memory, depending on your host, but keep most defaults, give it the name “debian-on-KVM-<firstname>-<lastname>”
  2. Continue with the installation, perform a standard Debian installation, but disable the Desktop Environment packages. Also choose the hostname to be “debian-on-KVM-<firstname>-<lastname>”

A screenshot of a computer

Description automatically generated

Don’t forget to select the /dev/vda drive to install the GRUB boot loader on!

* 1. You now have a fully operational Debian VM within your RHEL!

A screenshot of a computer

Description automatically generated

* 1. In the window of your Debian vVM, change the View to ‘Details’ and inspect the storage.   
     **Q: What file extension does your Virtual Disk have?**

Qcow2, situated in /var/lib/libvirt/images **Q: What does it mean?**

Q – QEMU, cow – copy-on-write, so similar to just Hyper-V storage VHD and etc.

* 1. On your shell, verify with ‘sudo virsh list’ you now indeed have another VM running.

A screen shot of a computer code

Description automatically generated

* 1. Check the IP settings of your RHEL.

**Q: What do you notice concerning network interfaces now that you have a second VM running?**

**A screenshot of a computer program

Description automatically generated**

**Another interface has been added – vnet2**

1. Now, power off your RHEL and disable Intel VT-x / AMD-V for your RHEL VM. Power on RHEL, start virt-manager again and start the installation of another debian (no need to fully complete it).

**Q: Is this possible without VT-x? If not, why? If yes, do you notice a difference?**

**A screenshot of a computer

Description automatically generated**

**Of course it can not work, because we are doing nested virtualization.**

After this test, re-enable VT-x again.

## Managing VMs Via a web interface: cockpit

In the linux world, ‘cockpit’ is favored by Red Hat to become the web interface for system management.

1. First install and enable cockpit :  
   *dnf install cockpit  
   systemctl enable --now cockpit.socket*

*A screenshot of a computer program

Description automatically generated*

1. Now surf in your web browser to your Red Hat’s IP address on port 9090, the default cockpit port. Login with your regular user account and find indeed system information. Click and confirm the ‘turn on administrative access’ top message.

A screenshot of a computer

Description automatically generated

1. Virtual machines are not shown yet, therefore install the plugin:  
   *dnf install cockpit-machines*
2. Refresh the cockpit web page and you’ll find a new section ‘Virtual Machines’.

A screenshot of a computer

Description automatically generated

1. Verify that your debian VM is now also accessible within the web interface and you can login to your debian from within the web page console.

A screenshot of a computer

Description automatically generated

**Q: Take a screenshot, cleary showing your browser, your personal VM in cockpit and a login session to it within the web console.**Note: Close any open graphical console window of your vVM which might still be open (virt-viewer or virt-manager), else the Console connection from within the web page will fail (only one graphical connection can be open at a time).