SMC lab platform

Spring 2022
David Morgan, instructor

A Virtual Machine for You

I have a VirtualBox virtual machine (VM). I wish to give you a copy. VirtualBox has a method for doing so. I can "export" my VM. That produces a file. I did it, and named the resultant file "fedora31-spring21.ova". We will use it in spring22. I wish to give you a copy of this file. You can then use the "import" feature of your copy of VirtualBox. Then, your VirtualBox will contain a copy of my VM for your use. fedora31-spring21.ova is big (around 4G).

GETTING THE FILE

I put the fedora31-spring21.ova file on Google Drive. You can reach it via this link:

https://drive.google.com/drive/folders/19SM407yvRbzWKiLy2RK-yuBjwjrwPp2N?usp=sharing

Please copy this link into a browser.

VERIFYING THE FILE

It is important that the file you receive be a correct, perfect copy of the one sent. Ascertain that, by using a command or program that produces the SHA1 hash of the file. For example try these command-line approaches:

in Windows' command box:	certutil -hashfile <name of file>
in linux:	sha1sum <name of file>
other linux-like platforms:	sha1 <name of file> -or- shasum -a 1 <name of file>

Or do an internet search for other tools where you'll find suggestions like:
https://www.raymond.cc/blog/7-tools-verify-file-integrity-using-md5-sha1-hashes/
(Many people have 7-zip installed; it contains the capability.)

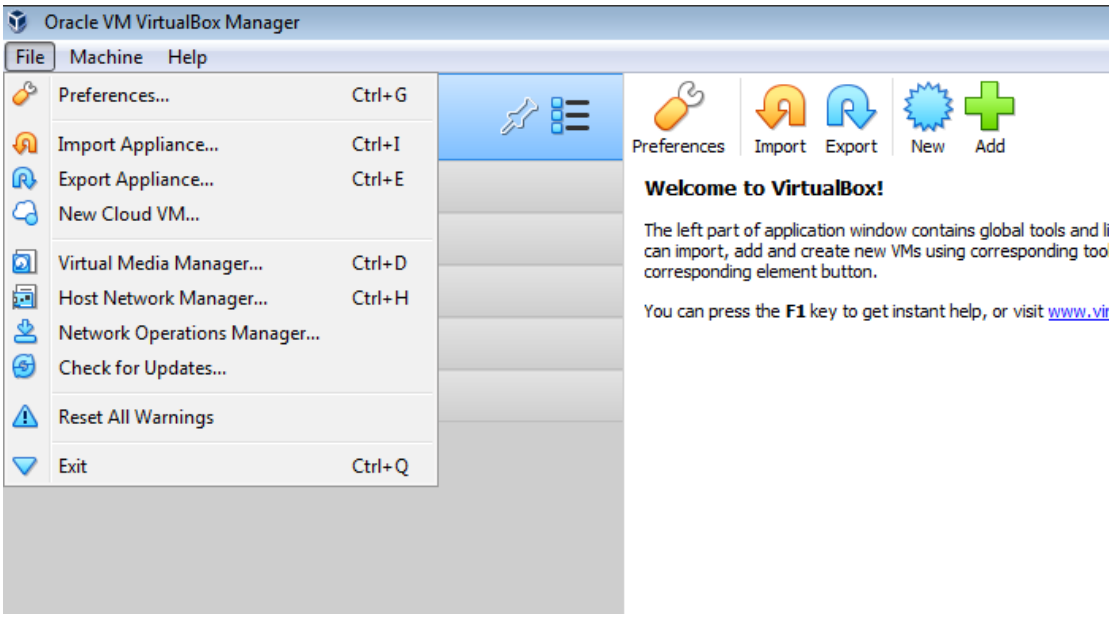
The sha1 hash value you get for the file must be:

47db276002e14d64540c62e17d0a9a9bc1c5eae4 fedora31-spring21.ova

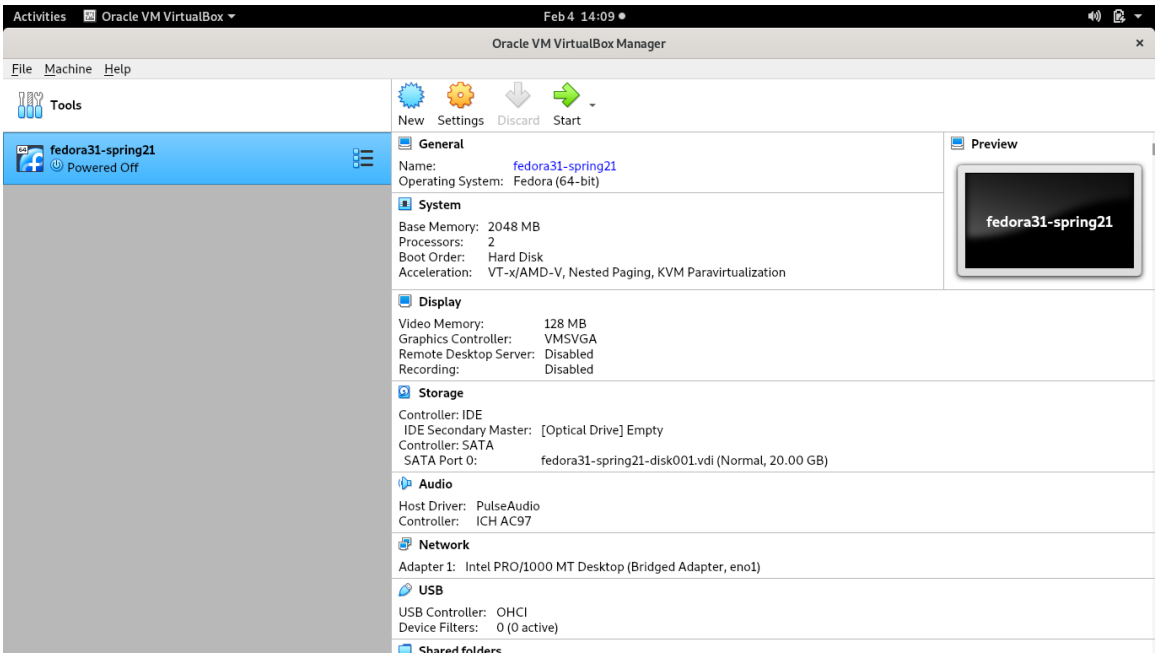
If you don't get the above value for your file (it's wrong) repeat the download attempt until you do. Don't proceed without having successfully verified the file.

IMPORT THE VM INTO VirtualBox

VirtualBox has an "Import Appliance" feature on its File menu:



Use it to specify and import fedora31-spring21.ova. As a result, a VM named fedora31-spring21 will appear.



Important preliminary before using this machine. There needs to be a snapshot of this machine named "base," reflecting its current, virgin state. (I may give you scripts that create VMs for lab exercises by cloning this one, and that depend on the snapshot.) Create and name "base" either via the graphical menus, or by running the command:

vboxmanage snapshot fedora31-spring21 take base

Virtual Machine Usage

fedora31-spring21 will be used as the basis for cloning multiple copies of itself. Some exercises need only one VM, others as many as five. The ones with multiple VMs will have them networked differently, in terms of topologies, interfaces, and addresses

PROVIDED SCRIPTS

For each exercise I have provided 4-6 scripts that set the stage automatically, cloning and connecting and addressing. (They're on Google Drive.) The purpose of the scripts is to 1) spare the student the setup steps, so the machine(s) and network come to you ready to use, and 2) facilitate tearing down and starting over, if desired, at little cost.

Execute the scripts as the non-root user. Every lab exercise has its own set of scripts. The sets are similar. They typically consist of these:

to start:
vmconfigure-populate.bat (or .sh for bash, on linux or Apple)
vmconfigure-construct-network.bat (if present)
vmconfigure-guestOS-internal-settings.bat (if present) OR vmconfigure-poweron.bat

to end:
vmconfigure-poweroff.bat
vmconfigure-destroy.bat

Execution order is important. It should be done in the order shown.

vmconfigure-populate clones the needed number of VMs from the base "fedora31-spring21" springboard machine. It is the physical equivalent of bringing some machines and putting them on a table.

vmconfigure-construct-network does the equivalent of physically provisioning computers with interfaces (NICs), bring some cables to the table, and plugging cables from certain interfaces on some machine to certain ones on other machines, depending on how you want things to be hooked up.

vmconfigure-guestOS-internal-settings does the equivalent of logging in to the machines then running some commands on them. (The chosen commands are hostname, ifconfig, and route.) Note these are commands belonging to the guest operating machine. The previous scripts ran commands that belong to VirtualBox. This one runs commands, in bash, that belong to linux or Apple (bash is the default shell, in both). Commands in the guest can only be executed in the guest, requiring that it be booted up first. That is why this script is an alternative to vmconfigure-poweron. The latter boots a machine and does nothing further with it. This former boots it, and having done so operates it a bit to set its hostname and/or configure its network. A ramification is that after VirtualBox powers on a machine, the script inserts a delay before trying to execute guest commands. That is because, exactly as with physical machines, though powering on is instantaneous booting is not. It takes half a minute or a minute every time you turn on your laptop or phone. You can't execute commands during that time, till the machine is booted. It's not ready yet. In the case of VirtualBox, if you jump the gun and try to execute guest commands before the guest has fully booted (impossible with a physical box), you will create error messages for yourself. Therefore, when vmconfigure-guestOS-internal-settings reaches its delay loop, let it delay. Be patient.

vmconfigure-poweroff turns the machines off. You can turn them on again, as you could physically. If you want to do that, again run vmconfigure-guestOS-internal-settings (if present) as some of the internal settings are transient and need to be set/reset each time you turn a machine on.

vmconfigure-destroy gets rid of the machines. Thereafter if you want to start over you can. Run vmconfigure-populate afresh and go from there.

Summary: using provided scripts for setting up experiments

Scripts are used. A set of them is provided for each experiment. A summary of the scripts' use and intended execution order follows.

"vmconfigure-populate" is first, creates machines
"vmconfigure-construct-network" is next, if it exists
"vmconfigure-guestOS-internal-settings" is next, it makes the internal settings after powering the machines on;
so if this one exists "vmconfigure-poweron" is not needed
"vmconfigure-poweron" is next, if there is no "vmconfigure-guestOS-internal-settings"

Now you can use your VM(s). When you are finished:

"vmconfigure-poweroff"
"vmconfigure-destroy" if you want to delete the machines, but not if you plan to come back to them later

If you do come back to them later,

"vmconfigure-guestOS-internal-settings" should be repeated, if present
"vmconfigure-poweron" if there is not "vmconfigure-guestOS-internal-settings" present

OBTAINING AND INSTALLING THE SCRIPTS

Scripts will be found on Google Drive (as above) in files:

vmconfigure-batch-scripts-windows.zip

vmconfigure-bash-scripts-linux-apple.zip

Download one of these according to your platform. Unzip. There will be a subdirectory/folder for each exercise you will be assigned, containing the scripts for that exercise.

SETTING THE PATH VARIABLE

The workhorse of the scripts I may give you is VirtualBox's "vboxmanage" command. It offers a command line equivalent for accomplishing almost anything that can be done, alternatively, through the Oracle VM VirtualBox Manager's graphical menus. It's for scripting. Therefore, when you run my scripts, it is necessary that their calls to vboxmanage find it. That is, it is necessary that vboxmanage be findable through the PATH variable. Where is vboxmanage?

Windows:	C:\Program Files\Oracle\VirtualBox\VBoxManage.exe
fedora linux:	/usr/bin/vboxmanage/vboxmanage
Mac:	/Applications/VirtualBox.app/Contents/MacOS/VBoxManage

Put their containing directories into your PATH variable before running scripts, namely:

Windows:	C:\Program Files\Oracle\VirtualBox
fedora linux:	/usr/bin/vboxmanage
Mac:	/Applications/VirtualBox.app/Contents/MacOS