USC CS530 Security Systems, lab platform

Fall 2020

David Morgan, lab 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 "fedora30-fall20.ova". I wish to give you a copy of it. 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.

fedora30-fall20.ova is big (nearly 5G). So distributing it to you faces two challenges: it is time-consuming and error prone. To facilitate transmitting it to you, I have split fedora30-fall20.ova into 10 component fragments named fedora30-fall20-frag00 through fedora30-fall20-frag09. Except the last one, fedora30-fall20-frag09, all are 500 million bytes in size.

GETTING THE FILES

I put the files into a shared folder in USC's Google Drive. You will download from there. The files are in two forms. First, the entire, large file is there:

fedora30-fall20.ova

Second, so are its 10 fragment files. They are in the form of 3 zip files containing them. The zip files' names are:

drive-download-20200901T182853Z-001.zip
drive-download-20200901T182853Z-002.zip
drive-download-20200901T182853Z-003.zip

Either download the large file, or the 3 zips which you then unzip to yield 10 fragments, as follows:

- go to the course home page for our CS530 course in D2L
- follow the "Software Download" link to a folder named "CS530 Lab Software"
- you will see the 4 files referenced above
- download the one, or the three others

VERIFYING THE FILES

After downloading, use a command or program that produces the SHA1 hash of each file. For example try these approaches:

in Windows' command box:	certutil -hashfile fedora30-fall20-frag00
in linux:	sha1sum fedora30-fall20-frag00
other linux-like platforms:	sha1 fedora30-fall20-frag00, or shasum -a 1 fedora30-fall20-frag00

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 7zip installed; it contains the capability.)

The sha1 hash values you get for the files must be:

f1a92dcaa048bceab0ec28cb99c973ec443662e0 fedora30-fall20-frag00
8a5da4d40e6539877c84a4cccb3f76546dc542ef fedora30-fall20-frag01
8d1f6921087665cbbce3293d1cebdf1d5d62f4d5 fedora30-fall20-frag02
56535f0c6c9820735f7864b303354691a2ed30c7 fedora30-fall20-frag03
1be9539207c0d5376da63e7a19176fc0dfa60e82 fedora30-fall20-frag04
9e2cb3d9e94090ed8d168601ed5a0fae9efb9cf6 fedora30-fall20-frag05
e3c91704c3b77c0bd982f48c6dbc55f413e9a573 fedora30-fall20-frag06
5cc589add897dc3ad6f4ea6a326fdc2adfd43434 fedora30-fall20-frag07
4e139cfc18540b30e7e35133754b763221d4b4d7 fedora30-fall20-frag08
cdae5bca5188f427d6477f03932675d25cd4d26b fedora30-fall20-frag09

If you get different values for any file, re-download it. Don't proceed without having successfully verified all the files.

RECOMBINE THE FILES

If you are working with the fragments you need to recombine them.

in Windows' command box:

```
copy /b fedora30-fall20-frag00+fedora30-fall20-frag01+fedora30-fall20-frag02+fedora30-fall20-
frag03+fedora30-fall20-frag04+fedora30-fall20-frag05+fedora30-fall20-frag06+fedora30-fall20-
frag07+fedora30-fall20-frag08+fedora30-fall20-frag09 fedora30-fall20.ova
```

or in linux:

```
cat fedora30-fall20-frag0? > fedora30-fall20.ova
```

(the above Windows command is single and continuous; it must be typed manually; sorting in Windows does not produce the needed file order.)

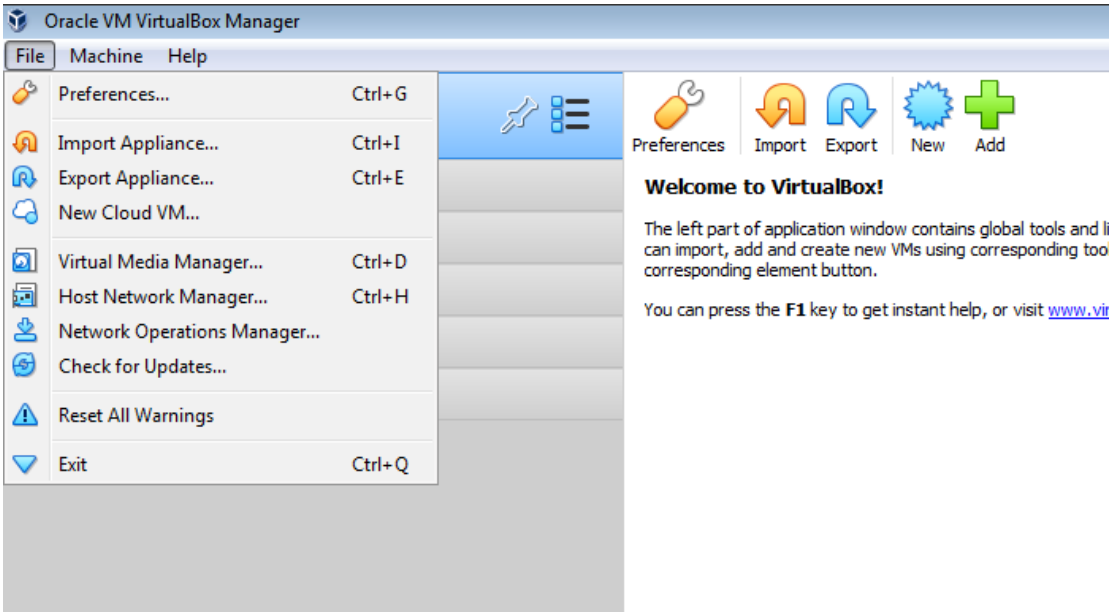
Once you have fedora30-fall20.ova, verify its correctness by running your sha1 utility on it. The resultant sha1 hash value must be:

37012ebd90e970d81b059c1e12a6ffeaae6093d5 fedora30-fall20.ova

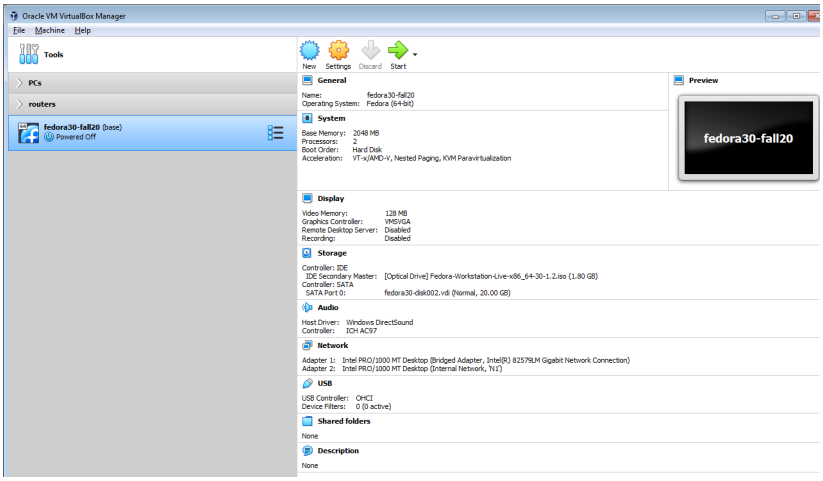
If you get a different value don't proceed; repeat above procedure(s) until you do.

IMPORT THE VM INTO VirtualBox

Run VirtualBox (i.e., the "Oracle VM VirtualBox Manager" program) as a non-root user. It has an "Import Appliance" feature on its File menu:



Use it to specify and import fedora30-fall20.ova. As a result, a VM named fedora30-fall20 will appear.



fedora30-fall20 will be used in a half-dozen or more exercises. For other exercises there will be a couple other machines.

Important prelimiary, there needs to be a snapshot of this machine named "base," reflecting its current, virgin state. (The scripts that create VMs for lab exercises depend on the snapshot.) Create and name it either via the graphical menus, or by running the command:

vboxmanage snapshot fedora30-fall20 take base

Virtual Machine Usage

fedora30-fall20 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. 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 "fedora30-fall20" 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.

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 of the 10 exercises, containing the scripts for that exercise.

SETTING THE PATH VARIABLE

The workhorse of these modest scripts is VirtualBox's "vboxmanage" command. It offers a command line equivalent for accomplishing almost anything that can be done, alternatively, through the 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?

C:\Program Files\Oracle\VirtualBox	on my Windows machine
/usr/bin/vboxmanage	on my fedora linux machine

Put those directories into your PATH variable before running scripts.