**Converting VM images to Docker containers**

[January 26, 2022](https://andygreenphd.com/2022/01/26/converting-vm-images-to-docker-containers/)[Andy Green, Ph.D.](https://andygreenphd.com/author/andygreenphd/)

Rate This

I’m starting to experiment with Docker containers. From a pedagogical perspective, I see lots of opportunities to quickly and easily script environments to support hands-on lab objectives for my students. Being able to create a series of images, push them to a public Docker repo, and then store a Docker compose script in a Github repo for students to access and run as needed just seems like the way to go. Students can choose to run the lab environment locally, or they can stand up a cloud-based VM in AWS, Azure, Digital Ocean, Linode… you get the idea.

I have a small library of existing VMs that may be useful, and I’m lazy enough that I don’t want to rebuild them from scratch as containers. I figured there has to be a way to convert VMs on my Win10 system to Docker containers, and there is. However, it is not a simple process and is time-consuming. I am documenting the steps I followed below, in the hopes that it can help someone else down the road, and for me to refer to in the future. A word of caution – I tested this by converting a Linux VM. I have not tested with a Windows VM, but my expectation is that the conversion process will still work.

These instructions were pieced together from the following sources:

* <https://www.vinnie.work/blog/2021-03-19-virtualmachine-to-docker/>
* <https://dannyda.com/2020/06/25/how-to-use-qemu-img-command-to-convert-between-vmdk-raw-qcow2-vdi-vhd-vhdx-formats-disk-images-qemu-img-create-snapshot-resize-etc/>
* <https://www.unixmen.com/run-kali-linux-2-0-in-docker-container/>

My current environment is a Win10 Enterprise System with WSL and Ubuntu app installed.

1. Open your Ubuntu app and jump into root by typing su root
2. Fully update Ubuntu by typing apt-get update && apt-get upgrade
3. We will use qemu-img to convert the existing VM VDMK file to a RAW file for Docker use. Install qemu-img by typing apt-get install qemu-utils and accept all prompts
4. Ubuntu automatically maps the Windows C partition for access. My VMs are in separate folders in the Windows partition located in c:\agreen\VMs To access that folder, I typed cd /mnt/c/agreen\VMs (NOTE: you will need to use the directory name on your system where you have your VMs stored)
5. Next, you need to convert your existing VMDK to a raw file for further use. Prior to beginning the conversion process, I created a directory named container under the specific VM folder of the image I’m converting. I’ll use this folder to hold my conversion work product.
6. Now you’re ready to convert the VMDK to a raw file. The syntax is qemu-img convert -O raw <source VMDK file> <destination>. In my case, I typed qemu-img convert-O raw image.vdmk container/image.raw (WARNING – This conversion process can take a while, depending on VMDK size)
7. Once the conversion process is complete, you need to look at the partition table on the new RAW file in order to get details necessary to mount the file for further use. I typed parted -s container/image.raw unit b print to get the data I needed in order to mount the partition. Below is my output – yours may vary. The important thing to pick up is the value in the “Start” column for the boot sector. In my case, it was **1045876**

Text

Description automatically generated*Sample output from the parted command*

1. Next, I had to mount the partition for use. I created a mount point by typing mkdir /mnt/container
2. Next, I mounted the RAW file by typing mount -o loop,ro,offset=1045876 container/image.raw /mnt/container
3. Next, I verified a successful mount by typing ls /mnt/container

*Sample output from the ls command*

1. Now that we have access to the file system of the VM, we need to put the entire partition in a tarball. I did this by typing tar -C /mnt/container -czf image.tar.gz container/. (WARNING – this may take a while, depending on partition size)
2. Now that we have the tarball, it’s time to import it into Docker. Ensure Docker is running on your system, then open a PowerShell terminal in Administrator mode and navigate to the container folder that has the tarball. The syntax to import the tarball into Docker is docker import <filename> <repository>:<tag>. In my case, I typed docker import image.tar.gz demotest:1.0 (WARNING – this may take a while, depending on tarball size)
3. Once the import process completes, type docker images to get details on your new Docker image that you’ll need to launch a new container.

*Sample output from the docker images command*

1. Now you can start a new container from your docker image by typing docker run -i -t <image id> <commands>. Since this is a Linux-based image, I needed to launch the bash shell on startup. In my case, I typed docker run -i -t 891dcfcad752 /bin/bash
2. Success! My container is now up and running, and I can move around as needed within the environment.

Graphical user interface, website

Description automatically generated*Directory listing from inside my container*

You can also look at the state of the image and container by using Docker Desktop:

Graphical user interface, application

Description automatically generated*Output from the “Images” tab inside Docker Desktop*Graphical user interface, application, Teams

Description automatically generated*Output from the “Containers/Apps” tab inside Docker Desktop* showing a running container

You can stop the container inside Docker Desktop by hovering over the container name and clicking the “stop” button, or you can use the docker stop <container\_name> command. In my case, I typed docker stop xenodochial\_darwin

*Output from docker stop command*

The output of the container name after running the command is confirmation that the container has stopped running. You can also verify the container’s state in Docker Desktop as well.

Graphical user interface, application

Description automatically generatedOutput from the “Containers/Apps” tab inside Docker Desktop showing a stopped container

That’s it! We now have a VM image converted into a fully functional Docker container.