New way of Migrating/Setting up Docker on Windows 10 Operating System Instead of Linux System

White Paper

SIBSANKAR BERA
sibsankarb4@gmail.com
sibsankar166@gmail.com

January 2019 / Sibsankar Bera

Section 1

Executive Summary

Welcome to my executive summary page, I am Sibsankar Bera, I have more than eight years of industry experience across Asia and Europe in CMMI level 5 organizations. Following studies showing the advantages and new way of migrating docker from linux to windows 10 system. Docker is natively built on and for linux environment, but it does support windows and mac os as well in a certain way. Let's find out the same in this study. Most of the people face lots of challenges to migrate/setup docker in windows platform. In this study step by step process of the same has been shown. Before begin let me introduce what is docker, why we need it, what are advantages/benefits of it. In todays market there are lots more containerization solution available apart from docker, still docker is the market leader. Let us find the reason behind it and details discussion of docker deployment on windows platform.

Section 2

Introduction

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code. In this article we will focus on Windows 10 machine.

In a way, Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application. Docker Container is a standardized unit which can be created on the fly to deploy a application or environment. It could be an Ubuntu container, CentOS container, etc. to full-fill the requirement from an operating system point of view. Also, it could be an application-oriented container like CakePHP container or a Tomcat-Ubuntu container etc.

Section 3

Understand with an Example

A company needs to develop a Java Application. In order to do so the developer will setup an environment with tomcat server installed in it. Once the application is developed, it needs to be tested by the tester. Now the tester will again set up tomcat environment from the scratch to test the application. Once the application testing is done, it will be deployed on the production server. Again the production needs an environment with tomcat installed on it, so that it can host the Java application. If you see the same tomcat environment setup is done thrice. There are some issues that I have listed below with this approach:
1) There is a loss of time and effort.
2) There could be a version mismatch in different setups i.e. the developer & tester may have installed tomcat 7, however the system admin installed tomcat 9 on the production server.
Now, I will show you how Docker container can be used to prevent this loss.
In this case, the developer will create a tomcat docker image (A Docker Image is nothing but a blueprint to deploy multiple containers of the same configurations) using a base image like Ubuntu, which is already existing in Docker Hub (Docker Hub has some base docker images available for free) . Now this image can be used by the developer, the tester and the system admin to deploy the tomcat environment. This is how docker container solves the problem.
However, now you would think that this can be done using Virtual Machines as well. However, there is catch if you choose to use virtual machine. Let's see a comparison between a Virtual machine and Docker Container to understand this better.
Consider a situation depicted in the above image. I have a host system with 16 Gigabytes of RAM and I have to run 3 Virtual Machines on it. To run the Virtual Machines in parallel, I need to divide my RAM among the Virtual Machines. Suppose I allocate it in the following way:

6 GB of RAM to my first VM,
4 GB of RAM to my second VM, and
6 GB to my third VM.
In this case, I will not be left with anymore RAM even though the usage is:

My first VM uses only 4 GB of RAM – Allotted 6 GB – 2 GB Unused & Blocked
My second VM uses only 3 GB of RAM – Allotted 4 GB – 1 GB Unused & Blocked
My third VM uses only 2 GB of RAM – Allotted 6 GB – 4 GB Unused & Blocked
This is because once a chunk of memory is allocated to a Virtual Machine, then that memory is blocked and cannot be re-allocated. I will be wasting 7 GB (2 GB +

1 GB + 4 GB) of RAM in total and thus cannot setup a new Virtual Machine. This is a major issue because RAM is a costly hardware.
So, how can I avoid this problem?
If I use Docker, my CPU will allocates exactly the amount of memory that is required by the Docker Container.
My first container will use only 4 GB of RAM – Allotted 4 GB – 0 GB Unused & Blocked
My second container will use only 3 GB of of RAM – Allotted 3 GB – 0 GB Unused & Blocked
My third container will use only 2 GB of RAM – Allotted 2 GB – 0 GB Unused & Blocked
Since there is no allocated memory (RAM) which is unused, I save 7 GB (16 – 4 – 3 – 2) of RAM by using Docker Container. I can even create additional containers from the leftover RAM and increase my productivity.
So here Docker Container clearly wins over Virtual machine as I can efficiently use my resources as per my need.

Section 4

The Prerequisites

The Hardware and Software Requirement

- CPU dual core 2 GHz or better
- RAM 2 GB or more
- HDD at least 2 GB of free space
- Additional Requirements at least 2 GB of swap
- A static IP address
- OS amd64 Linux distribution with kernel version 3.10 or later
- Windows 7 or later platform having kernel version 3.10 or later
- Additional Requirements:
  Docker: version 1.10 or later

The Port Used

When installing UCP on a host, make sure the following ports are open:

Hosts	Direction	Port	Purpose
controllers, nodes	in	TCP 443 (configurable)	Web app and CLI client access to UCP.
controllers, nodes	in	TCP 2375	Heartbeat for nodes, to ensure they are running.
controllers	in	TCP 2376 (configurable)	Swarm manager accepts requests from UCP controller.
controllers, nodes	in, out	UDP 4789	Overlay networking.
controllers, nodes	in, out	TCP + UDP 7946	Overlay networking.
controllers, nodes	in	TCP 12376	Proxy for TLS, provides access to UCP, Swarm, and Engine.
controller	in	TCP 12379	Internal node configuration, cluster configuration, and HA.
controller	in	TCP 12380	Internal node configuration, cluster configuration, and HA.
controller	in	TCP 12381	Proxy for TLS, provides access to UCP.
controller	in	TCP 12382	Manages TLS and requests from swarm manager.
controller	in	TCP 12383	Used by the authentication storage backend.
controller	in	TCP 12384	Used by authentication storage backend for replication across controllers.
controller	in	TCP 12385	The port where the authentication API is exposed.
controller	in	TCP 12386	Used by the authentication worker.

Section 5

Installation Process

README FIRST for Docker Toolbox and Docker Machine users: Docker Desktop for Windows requires Microsoft Hyper-V to run. The Docker Desktop for Windows installer enables Hyper-V for you, if needed, and restarts your machine. After Hyper-V is enabled, VirtualBox no longer works, but any VirtualBox VM images remain. VirtualBox VMs created with docker-machine (including the default one typically created during Toolbox install) no longer start. These VMs cannot be used side-by-side with Docker Desktop for Windows. However, you can still use docker-machine to manage remote VMs.

System Requirements:

- Windows 10 64bit: Pro, Enterprise or Education (1607 Anniversary Update, Build 14393 or later).
- Virtualization is enabled in BIOS. Typically, virtualization is enabled by default. This is different from having Hyper-V enabled. For more detail see Virtualization must be enabled in Troubleshooting.
- CPU SLAT-capable feature.
- At least 4GB of RAM.

- **What the Docker Desktop for Windows install includes:** The installation provides Docker Engine, Docker CLI client, Docker Compose, Docker Machine, and Kitematic.
- Containers and images created with Docker Desktop for Windows are shared between all user accounts on machines where it is installed. This is because all Windows accounts use the same VM to build and run containers.
- Nested virtualization scenarios, such as running Docker Desktop for Windows on a VMWare or Parallels instance might work, but there are no guarantees. For more information, see Running Docker Desktop for Windows in nested virtualization scenarios.

Install Docker Desktop for Windows desktop app

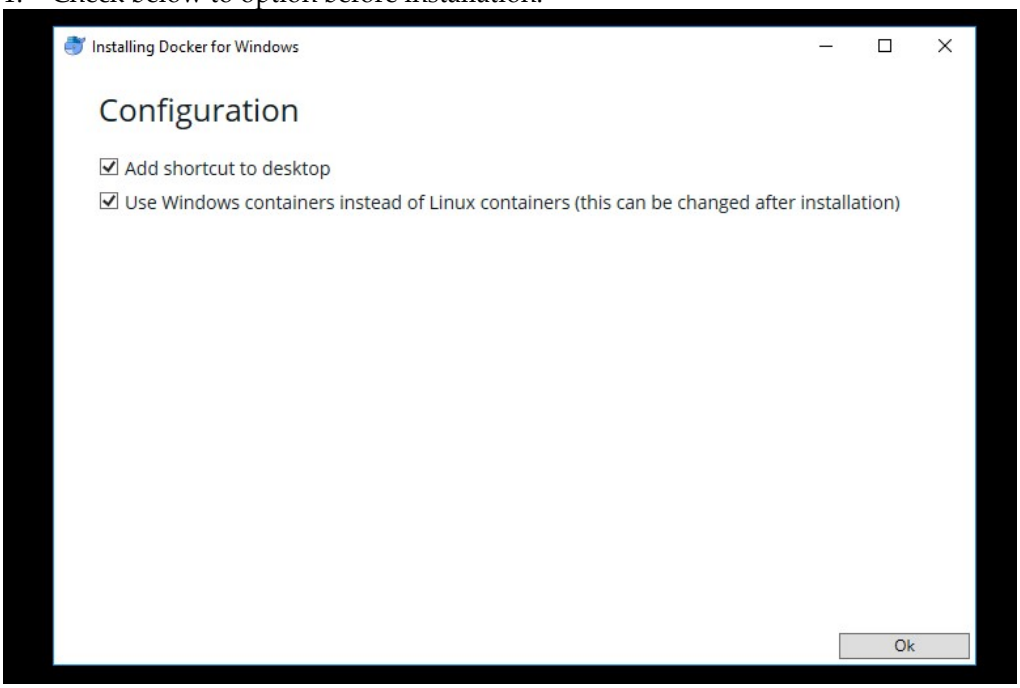
Double-click Docker Desktop for Windows Installer.exe to run the installer.If you haven't already downloaded the installer (Docker Desktop Installer.exe), you can get it from *download.docker.com*. It typically downloads to your Downloads folder, or you can run it from the recent downloads bar at the bottom of your web browser.

Follow the install wizard to accept the license, authorize the installer, and proceed with the install.

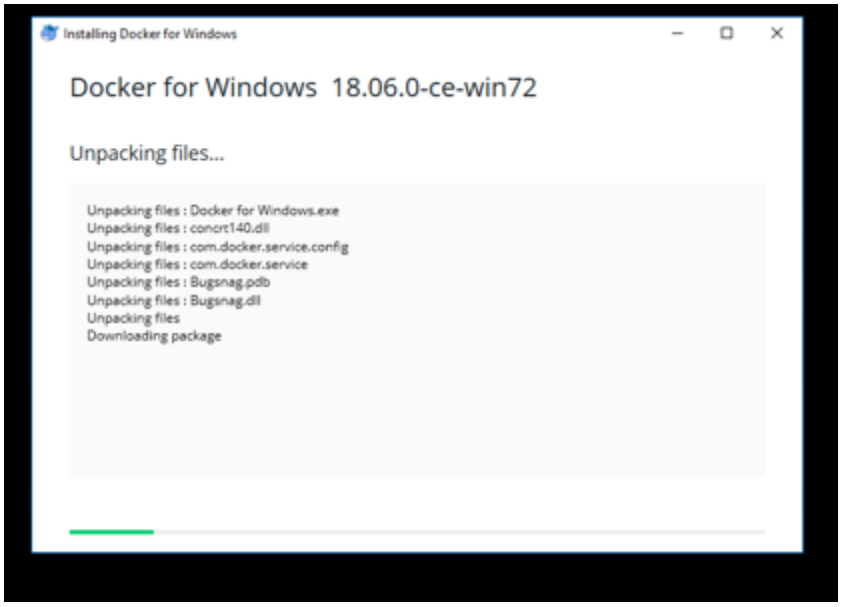
You are asked to authorize Docker.app with your system password during the install process. Privileged access is needed to install networking components, links to the Docker apps, and manage the Hyper-V VMs.

Click Finish on the setup complete dialog to launch Docker.

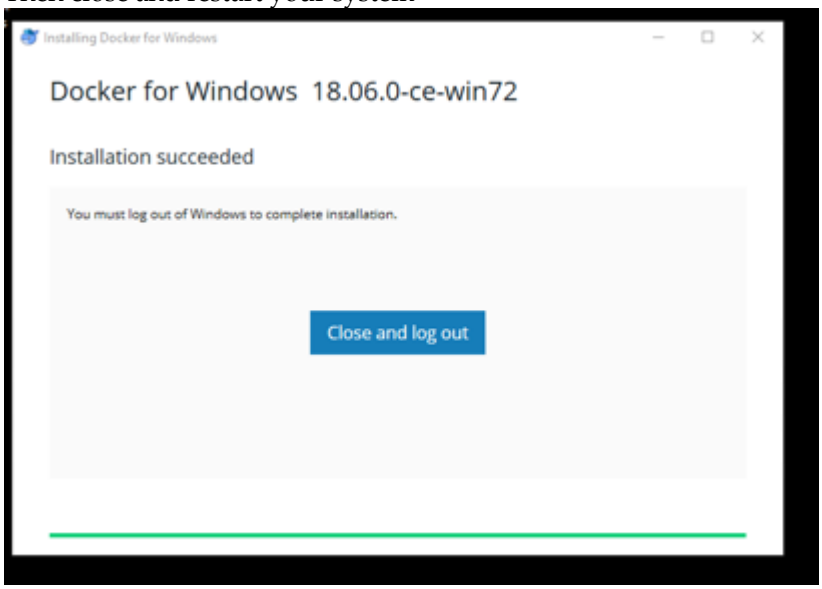
1. Check below to option before installation.



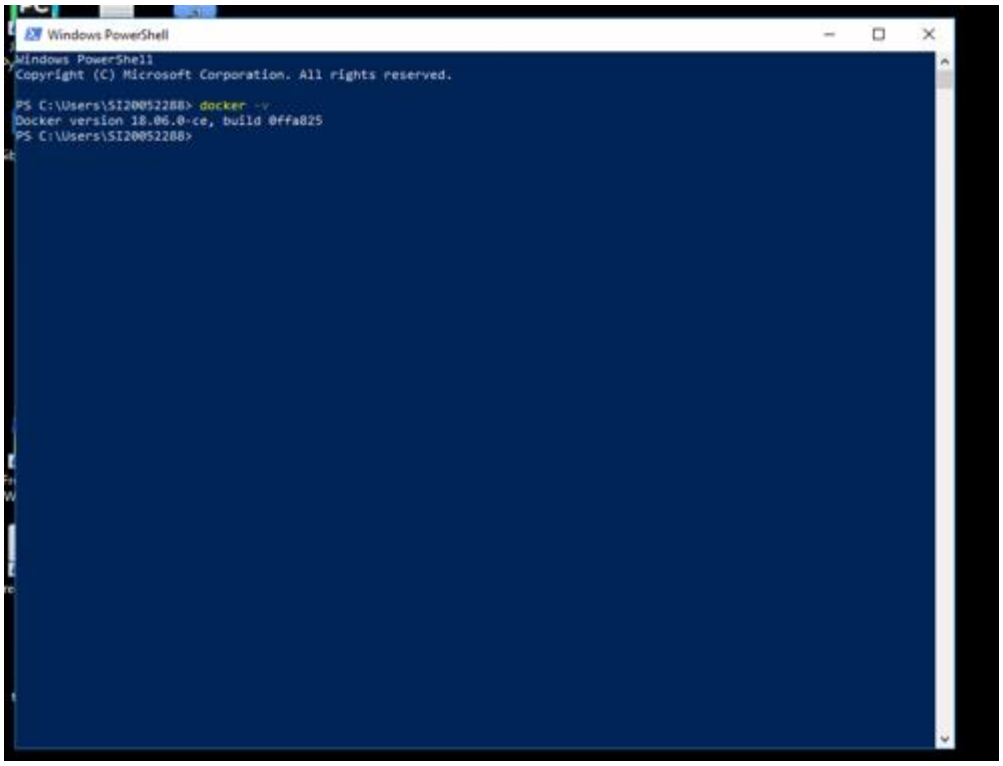
2. Then it will start installing



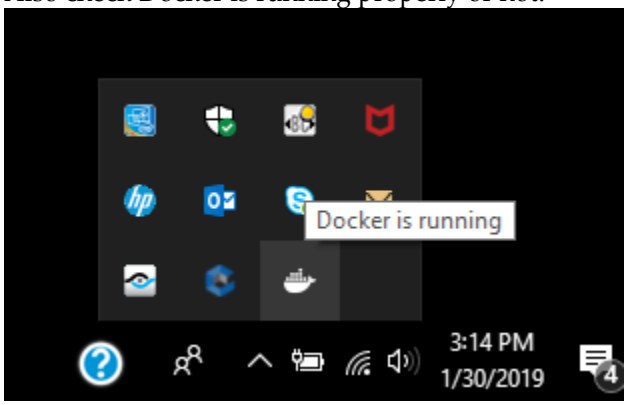
3. Then close and restart your system



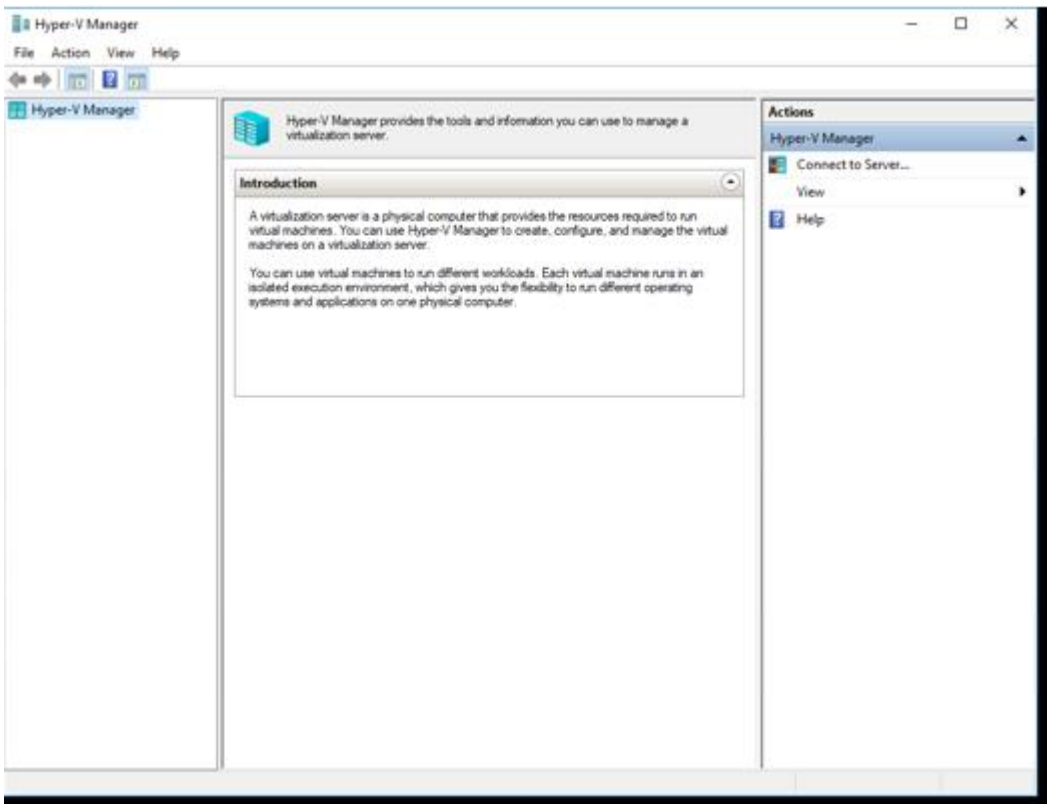
4. Then open windows powershell and check docker version



Also check Docker is running properly or not.



5. Make sure Hyper-V is enabled for organizing vm



Hyper-V is automatically enabled on a Docker Desktop for Windows installation. To enable it manually, see instructions on how to manually enable Hyper-V on the Microsoft developer network.

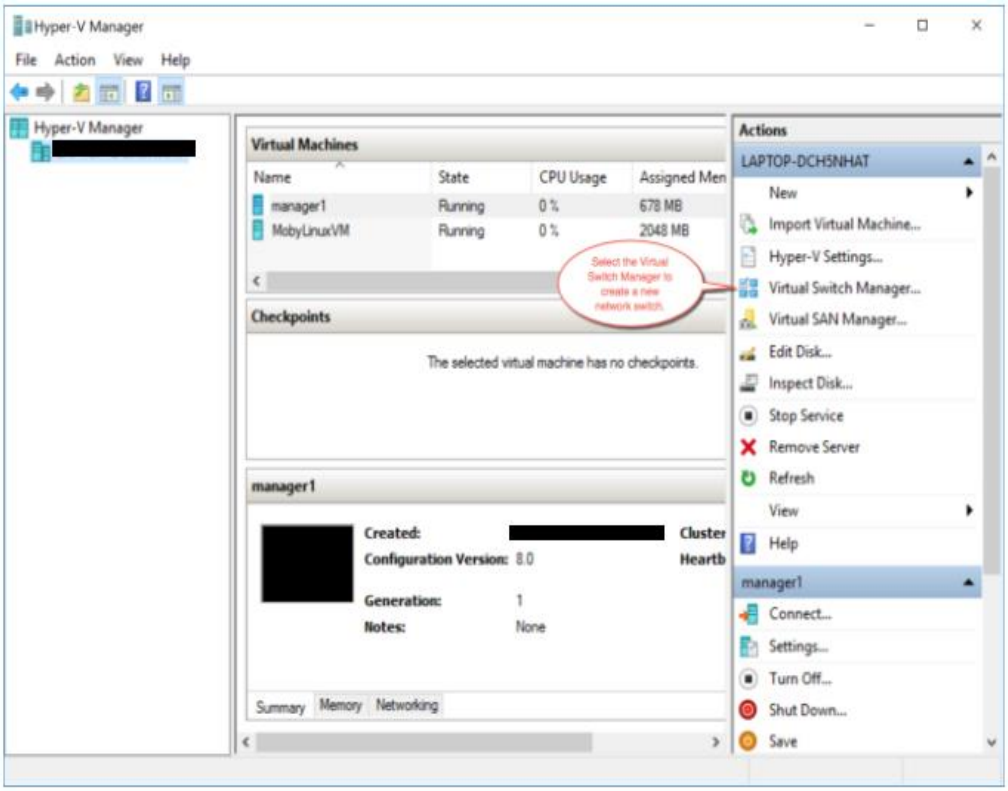
2. SET UP A NEW EXTERNAL NETWORK SWITCH (OPTIONAL)

Note: If you already have an external network switch, skip this setup and use that one instead.

Make sure you have Ethernet connectivity while you are doing this.

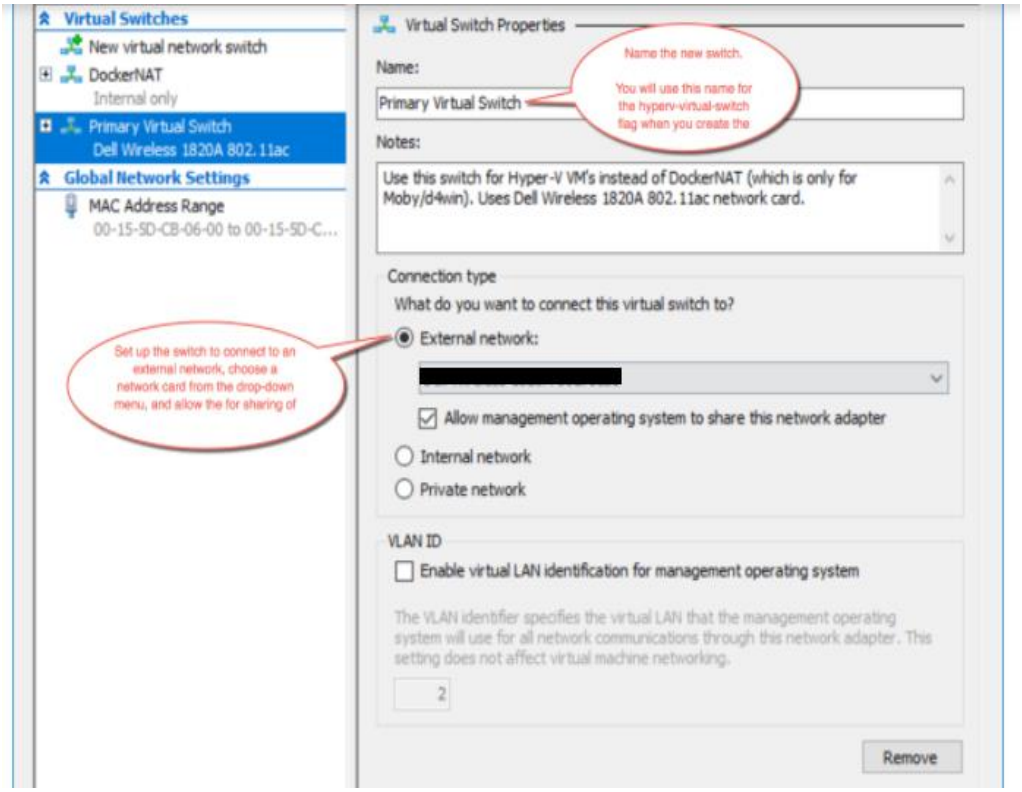
Open the Hyper-V Manager. (On Windows 10, search for the Hyper-V Manager in the lower left search field.)

Select the Virtual Switch Manager on the right-hand Actions panel.



Set up a new external network switch to use instead of DockerNAT network switch (for Moby), which is set up by default when you install Docker Desktop for Windows. If you already have another network switch set up, use that one instead but make sure it is an external switch.)

For this example, we created a virtual switch called Primary Virtual Switch.


Virtual Switches
New virtual network switch
DockerNAT
Internal only
Primary Virtual Switch
Dell Wireless 1820A 802.11ac
Global Network Settings
MAC Address Range
00-15-5D-CB-06-00 to 00-15-5D-C...
Virtual Switch Properties
Name:
Primary Virtual Switch
Name the new switch.
You will use this name for the hyperv-virtual-switch flag when you create the
Notes:
Use this switch for Hyper-V VM's instead of DockerNAT (which is only for Moby/d4win). Uses Dell Wireless 1820A 802.11ac network card.
Connection type
What do you want to connect this virtual switch to?
External network:
Set up the switch to connect to an external network, choose a network card from the drop-down menu, and allow the for sharing of
Allow management operating system to share this network adapter
Internal network
Private network
VLAN ID
Enable virtual LAN identification for management operating system
The VLAN identifier specifies the virtual LAN that the management operating system will use for all network communications through this network adapter. This setting does not affect virtual machine networking.
2
Remove

Section 6

Creating Docker Nodes

Reboot your desktop system to clear out any routing table problems. Without a reboot first, docker-machine create ... might get hung up on Waiting for host to start.... If you are still hung on "Waiting for host to start..." after you reboot, make sure you selected the correct network in the Virtual Switch Manager.

- **CREATE THE NODES WITH DOCKER MACHINE AND THE MICROSOFT HYPER-V DRIVER**
- Start an "elevated" PowerShell with administrator privileges. To do this, search for PowerShell, then right click and choose Run as administrator.
- Run the docker-machine create commands to create machines.
- For example, if you follow along with the Swarm mode tutorial which asks you to create three networked host machines, you can create these swarm nodes: manager1, worker1, worker2.
- Use the Microsoft Hyper-V driver and reference the new virtual switch you created.

docker-machine create -d hyperv --hyperv-virtual-switch <NameOfVirtualSwitch> <nameOfNode>

Here is an example of creating a `manager1` node:

PS C:\Users\SI20052288> docker-machine create -d hyperv --hyperv-virtual-switch "Primary Virtual Switch" manager1

Running pre-create checks...

Creating machine...

(manager1) Copying C:\Users\<your_username>\.docker\machine\cache\boot2docker.iso to C:\Users\<your_username>\.docker\machine\machines\manag

er1\boot2docker.iso...

(manager1) Creating SSH key...

(manager1) Creating VM...

(manager1) Using switch "Primary Virtual Switch"

(manager1) Creating VHD

(manager1) Starting VM...

(manager1) Waiting for host to start...

Waiting for machine to be running, this may take a few minutes...

Detecting operating system of created instance...

Waiting for SSH to be available...

Detecting the provisioner...

Provisioning with boot2docker...

Copying certs to the local machine directory...

Copying certs to the remote machine...

Setting Docker configuration on the remote daemon...

Checking connection to Docker...

Docker is up and running!

To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: C:\Program Files\Docker\Docker\Resources\bin\docker-machine.exe env manager1

PS C:\Users\SI20052288>

Use the same process, driver, and network switch to create the other nodes.
For our example, the commands are:

docker-machine create -d hyperv --hyperv-virtual-switch "Primary Virtual Switch" worker1
docker-machine create -d hyperv --hyperv-virtual-switch "Primary Virtual Switch" worker2

The virtual machines you create with docker-machine create show up in the Hyper-V Manager under "Virtual Machines", but you need to manage them with docker-machine commands and not through the Hyper-V Manager.
Now you are done with migrating/set up docker in windows 10 system. You can now run any command in docker vm like create/pull/push image, creating/deleting/start/stop services. Some basic commands are given below,

Save and restore data/image
The following procedure can be used to save/restore images and container data, for example, if you want to switch between Edge and Stable, or reset your VM disk:

- Use *docker save -o images.tar myImage1 [myImage2 ...]* to save any images you want to keep
- Use *docker export -o myContainner1.tar new_container1* to export containers you want to keep.
- Uninstall the current app & Install a different version of the app (Stable or Edge), or reset your VM disk.

- Use *docker load -i images.tar* to reload previously saved images.

- Use *docker import -i myContainer1.tar* to create a filesystem image corresponding to previously exported containers.

Following are the commands which can be ran now:

- docker –version
- docker pull
- docker run
- docker ps
- docker ps -a
- docker exec
- docker stop
- docker kill
- docker commit
- docker login
- docker push
- docker images
- docker rm
- docker rmi
- docker build

Section 7

Conclusion

We've had a pretty good tour through what Docker is and how to setup in windows platform and make it runnable poperly, and how it can benefit you and your organization. We also mapped some of the common pitfalls. We have tried to impart to you many of the small pieces of wisdom that we picked up from running Docker in production. Our personal experience has shown that the promise of Docker is realistically achievable, and we've seen significant benefits in our organization as a result. Like other powerful technologies, Docker is not without its downsides, but the net result has been a big positive for us, our teams, and our organization. If you implement the Docker workflow and integrate it into the processes you already have in your organization, there is every reason to believe that you can benefit from it as well. So let's quickly review the problems that Docker is designed to help you solve and some of the power it brings to the table. Please find my below profile for more result/details/discussion

https://github.com/sibsankarb4

https://www.linkedin.com/in/sibsankar-bera-351a5199