Docker Toolbox Setup - Windows

With the release of Docker 1.8, Windows users who have been learning Docker using Boot2Docker VM Tool, need to learn a few new concepts to get their environment in place for their Docker learning journey.

The guide below presents how to get the Docker Toolbox Setup on Windows and the general concepts behind Docker Machine, so that if you have been familiar with Boot2Docker before, your journey is a smooth. If this is what your first time installing the Docker toolchain on your Windows box, then you have come to the right place too!

Download the Installer

You need to have Windows 7.1,8, 8.1 to run the Docker Toolbox. The installation page is here and it makes sense to go through it to validate certain things about your Operating System.

The installer is available here and you should click on the Download (Windows) button to download the latest version of the installer.

Run the Installer

Once you download the Installer binary on your machine, please launch it and pay special attention to the following:

1. Please install in the default **C:** drive and not some other drive. This will avoid some issues with starting up the Docker Machine later.
2. In the Select Components, deselect Kitematic for now. It is better to install the command line tools while learning, so that you are aware what happens under the hood. I like Kitematic and suggest that you look at it, but for now, let us stick with the command line tools.

Go through the installation, selecting any defaults for the other steps.

What happened?

On successful installation, you should have the following:

A folder named C:\Program Files\Docker Toolbox is created and in that are several key files but we will mention a few of them here:

- **docker-machine.exe** : This is the utility to use to create/manage docker VMs on your computer. We will get to them in a while but thing of this as a commander shell that can create/manage one or more Docker VMs on your machine and set one of the Docker VMs i.e. machines as a target, which you can then use to execute standard commands with the docker.exe utility that is next.
- **docker.exe** : This is the standard docker client (which if you have used before -- should be familiar). This is now used directly from your Windows machine and it is able to communicate to the Docker Machine that you have set as a target. It will execute docker commands in that machine and show back the results.

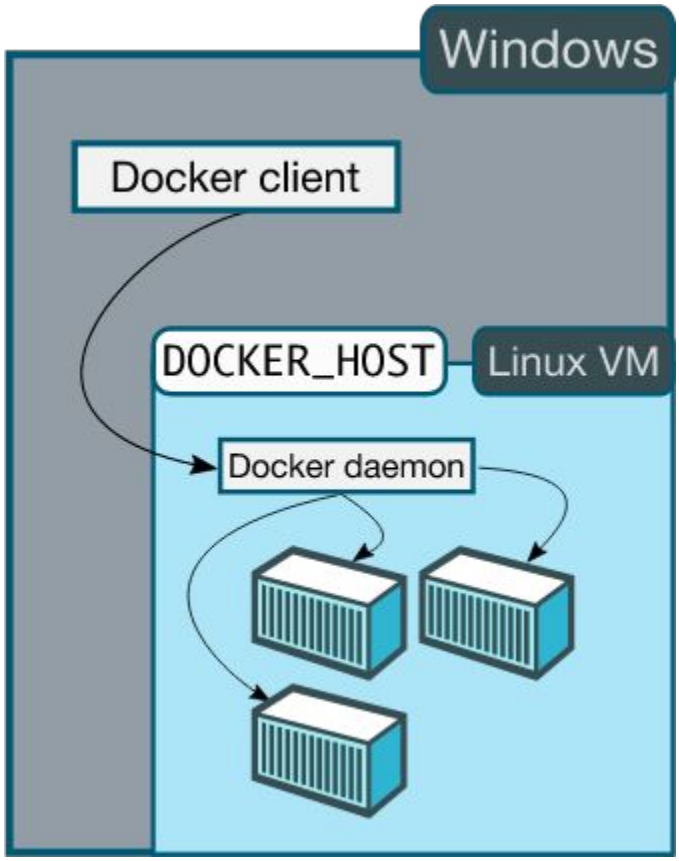
In addition to the folder above, you have 2 more things to keep in mind:

- The folder **C:\Program Files\Docker Toolbox** has been added to your Environment **PATH** variable. This means that you can run the **docker-machine.exe** and **docker.exe** client programs from anywhere. This is handy.
- There is a shortcut created on your Desktop named **“Docker Quickstart Terminal”**. This will actually launch the **./start.sh** file that is present in the **C:\Program Files\Docker Toolbox** folder.

The **Docker Quickstart Terminal** is a good thing to launch next. Double-click and launch it. It will create a default Docker Machine for you (named **default**). This means that if all goes well, you already have a Docker Machine, which is nothing but a Docker VM that is running on your machine with the Docker Daemon running inside it, ready to accept commands. Think of this as nothing but your Boot2Docker VM that is running (if you are familiar with that).

Docker Machine

For those of you, who are familiar with Boot2Docker and how it runs on Windows, this diagram should be familiar from the official Docker documentation:

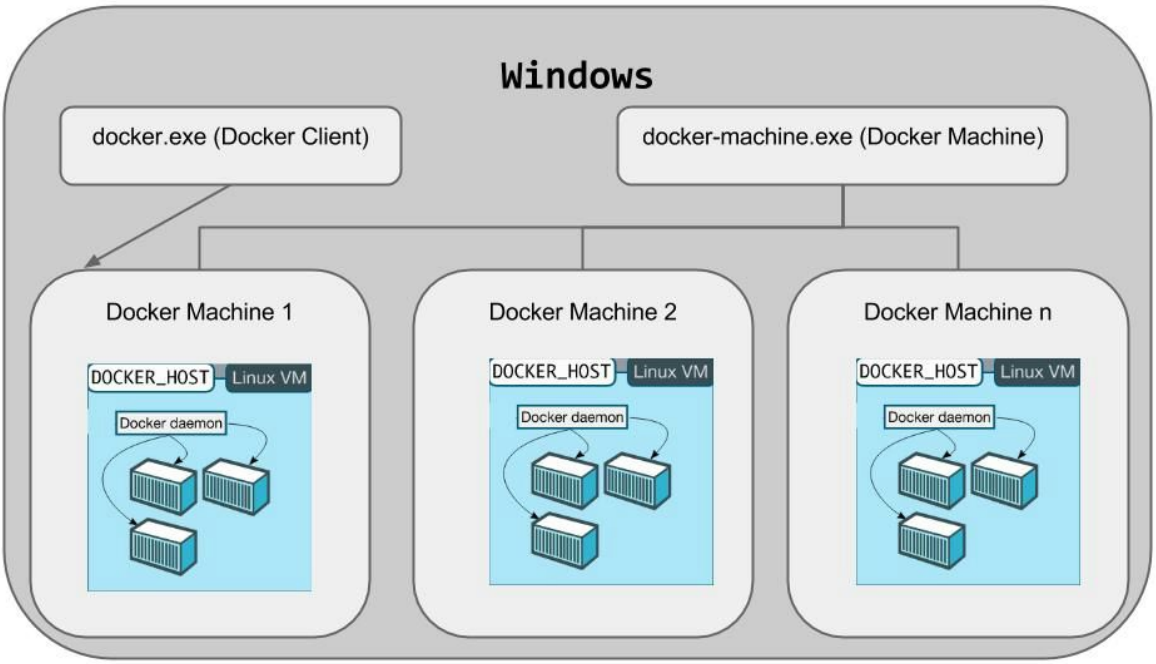


The DOCKER_HOST that you see in the above diagram is nothing but the boot2docker VM that was running inside Virtual Box.

With Docker Machine, you can now conceptually think of it as one instance of the DOCKER_HOST. What this means is that you can have multiple Docker Machines running on your computer. It's not just that but it is also a generic way to create Docker Hosts on not just your computer but also cloud providers, your own data center and more.

On Windows Machine, the driver is the Oracle VirtualBox system, which will use the Boot2Docker VM behind the scenes. It's starting to make sense, isn't it?

To better understand that, see the diagram below (it is simplified version of everything and may not be very accurate, but it has helped me to understand and put the tools in context vis-a-vis their roles):



There are a few key things here to help you understand the role of the Docker Client (docker.exe) and why there is a single arrow that is connecting it to one of the Docker Machines (Docker Machine 1).

But first up, let us be clear with the Docker Machine vis-a-vis the diagram that is shown above. You can have one or more Docker Machines running on your Computer as shown. You create and manage the Docker Machines (create, stop , start, etc) via the docker-machine.exe utility. It has its own list of commands, some of which we will see in the next section. Think of these as your Boot2Docker commands (if you are familiar with that).

Now, each Docker Machine is its own boss. Each Docker Machine or Host will have its own IP Address i.e. the IP Address of the VM.

So how do you run docker commands against each one. That is where the docker.exe client comes in. We shall see teh commands in a while, but what we have to do is that we need to use docker-machine.exe utility to set the environment for the Docker client (docker.exe). The Environment is nothing but setting certain Environment Variables that will set who the Docker Host (Docker Machine) is, the port, IP address and so on. Then when you run “docker <command>”, it will contact the Docker daemon running on that Docker Host (Docker Machine). This way, you can switch the environment to point to another Docker Machine and run docker commands against that one. That is one neat thing.

Note: When you run the Docker Quickstart Terminal shortcut on your machine, it creates one Docker Machine for you that is named **default**. I strongly recommend going through the Docker Machine concepts, it is a well written guide.

Docker Machine commands

As mentioned, the Docker Machine is a utility named docker-machine.exe and it is available in the PATH variable, so you can invoke it from anywhere.

It has several commands that are fairly intuitive and what you would expect. For e.g. creating new Docker Machines, Listing the current ones, finding status of the Docker Machines (running/stopped), Deleting them, upgrading them and more. You can find a full list of the Docker Machine commands over here.

Let us try some commands now. Open up a Command Prompt Window (cmd.exe).

Assuming that you have run the **Docker Quickstart Terminal** and that the **'default'** Docker machine was created, give the following command:

```
docker-machine ls
```

This should list out the Docker machines that you have on your computer.

```
NAME        ACTIVE   DRIVER       STATE     URL                         SWARM
default     virtualbox            Running   tcp://192.168.99.100:2376
dev         virtualbox            Stopped
```

You can see from my output that I have 2 Docker Machines (default and dev). I have created the dev machine and hence you see an extra machine in here.

Notice that the driver is mentioned as VirtualBox and on Windows, that is what the Driver is i.e. VirtualBox, which behind the scenes will use the Boot2Docker.iso image to launch the Linux VM container. For other drivers, check out the official drivers supported at the time of writing this.

Notice the IP Address that the default Docker Machine has, along with the standard port.

How do we create another Docker Machine. Well, we use the create command as given below:

```
docker-machine create --driver virtualbox machine007
```

This will take a while to create a machine and you should see the following output:

```
Creating VirtualBox VM...
Creating SSH key...
Starting VirtualBox VM...
Starting VM...
```

```
To see how to connect Docker to this machine, run: docker-machine env
machine007
```

Here we gave the name of the Docker Machine as **machine007**. We have to provide the driver, which for us on Windows is VirtualBox.

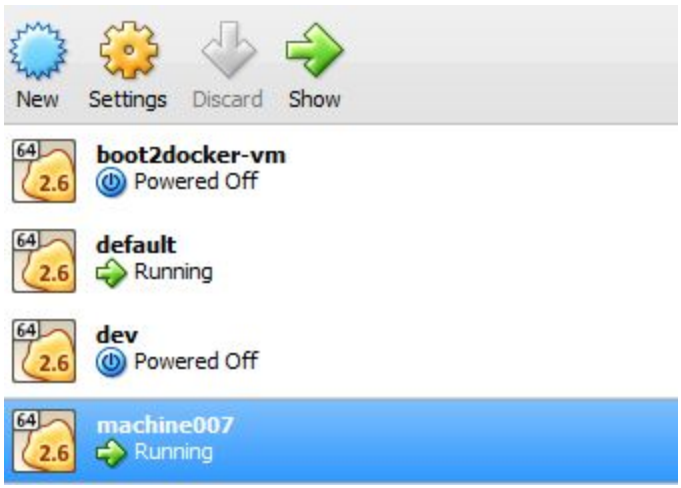
Now, if you do a ls command, you will see the 3 Docker Machines listed as given below:

```
NAME          ACTIVE   DRIVER       STATE     URL
SWARM
default                virtualbox   Running   tcp://192.168.99.100:2376
dev                    virtualbox   Stopped
machine007             virtualbox   Running   tcp://192.168.99.102:2376
```

Notice that machine007 has another IP address. You can try other commands like stopping a Docker Machine or starting a Docker Machine and so on. Some examples are given below:

```
docker-machine start dev
docker-machine stop default
docker-machine status dev
```

If you look at the Oracle Virtualbox GUI, you will see the VMs listed as shown below:



Running the Docker client

The final part of this puzzle but which is very important is how do we run docker.exe client to target a specific Docker Machine.

If you were paying close attention when you created a Docker Machine via the `docker-machine create` command, you would have seen that the last line of output was the following:
`To see how to connect Docker to this machine, run: docker-machine env machine007`

In short, we need to use docker-machine env command to set some environment variables that are used by the docker.exe client to connect to the specific Docker Host. What this means is that logically we need to tell the client where the Docker Host is, port, etc.

So, let us assume that we want to run the docker commands against the Docker machine named **dev**. Make sure that the Docker machine is running before you do this command.

On my machine, I first do a :

```
Prompt> docker-machine status dev
Stopped
```

This means that it is stopped. So I start it with the following command:

```
Prompt> docker-machine start dev
Starting VM...
Started machines may have new IP addresses. You may need to re-run the
`docker-machine env` command.
```

Now, let me do a `ls` command as shown below:

```
docker-machine ls
NAME         ACTIVE   DRIVER       STATE     URL
SWARM
default               virtualbox   Running   tcp://192.168.99.100:2376
dev                   virtualbox   Running   tcp://192.168.99.101:2376
machine007            virtualbox   Running   tcp://192.168.99.102:2376
```

Great, so I want to connect now to **dev** machine, so I give the following command:

```
docker-machine env --shell cmd dev
```

This gives us the following output:

```
set DOCKER_TLS_VERIFY=1
set DOCKER_HOST=tcp://192.168.99.101:2376
set DOCKER_CERT_PATH=C:\Users\irani_r\.docker\machine\machines\dev
set DOCKER_MACHINE_NAME=dev
# Run this command to configure your shell: copy and paste the above
values into your command prompt
```

What this means is that you need to manually run the above set commands. Copy and Paste each of the above **set** commands in your Command Prompt. Once you are done with that, what you have specified are Docker Machine details that the docker client can now use.

Now, go ahead and use your docker.exe client as before. Run commands like docker images, docker ps, docker info and so on. The commands will be executed only against the Docker machine **dev**.

If you want to run the docker commands against some other Docker Machine, use the docker-machine env command again to get the specific environment variables to set, set them and you are good to go.

Hope this sets you up now for your Docker journey if you have been familiar with Boot2Docker and are trying to wrap your head around Docker Machine and the new Toolbox.

Keep in mind that the Docker Toolbox is in Beta. So do expect a few things to not work at times and be patient.