Setting up Docker on macOS

Yesterday (February 26, 2017) I spent some time trying to learn how to use Docker and getting it setup on my computer, this post documents what I did to get it working as there were no straight-forward guides on this process.

Step 0: Install Homebrew

I try to use Homebrew to install everything I can as it gets managed for me. This was why I choose to install via Homebrew rather than attempting to install the Docker Mac App.

> Note: This is entirely for personal usage, I would recommend you use their official Mac app (which does provide the same support through Hypervisor.framework) for anything you plan to put into production. This guide provides an alternative way to get the same support without their application.

Step 1: Install docker

This step is a given, `brew install docker` to get Docker installed locally.

Step 2: Install docker-machine

This tool is used to create and setup virtual machines and is not bundled with the install of Docker, so run `brew install docker-machine`.

Step 3: Using Hypervisor.framework

To create a virtual machine, you need a driver for it. The driver does the work of interfacing with the software that runs the virtual machine. Often you will use VirtualBox/VMWare/Parallels/etc to provide the base for the virtual machine to run on. However, macOS comes with a special framework (Hypervisor.framework) that allows you to create and run virtualized systems from the userland. This is something that I wanted to take advantage of, thus avoiding the need to install additional kernel extensions to run the virtual machines.

Luckily I was able to find an existing project that allows for this framework to be used as the virtual machine driver: docker-machine-driver-xhyve. To install, run `brew install docker-machine-driver-xhyve` and follow the additional

install instructions that are presented at the end of the install process (also find them here).

Step 4: Creating a new Docker Host

With all the components install, you can go ahead and run the command to setup a new host: `docker-machine create --driver xhyve default`. This is going to use the driver that was installed in the previous step to create a new Docker host.

Step 5: Using the Host with Docker

Once the host is up and running, run `docker-machine env default` and add the environment variables it lists to your current working environment (there may be instructions here on how to get this stuff to get automatically loaded into your shell on start-up). Example:

```
$ docker-machine env default
set -gx DOCKER_TLS_VERIFY "1";
set -gx DOCKER_HOST "tcp://192.168.64.2:2376";
set -gx DOCKER_CERT_PATH "/Users/Samantha/.docker/machine/machines/defa
set -gx DOCKER_MACHINE_NAME "default";
# Run this command to configure your shell:
# eval (docker-machine env default)
```

Step 6: Ready to Go

At this point you should be ready to start using Docker to pull down images or create your own. I'm still learning how to do the rest of this stuff with Docker, but before I got any farther I wanted to document how I got started so other people don't have to spend the same number of hours I did searching around and piecing information together.

Step 7: Stopping

So the point of Docker is to run small self-contained instances of specific applications. These are still virtual machines running on your computer regardless if they are in use or not. To save resources, you can stop these virtual machines from running through the `docker-machine stop ...` interface.