Setting up a Docker dev environment on Mac OS X

There are countless articles and posts on the web about settting up a docker dev environment on OS X, and all of them work more or less. Here's another one :-)

This solution uses the following..

- Virtualbox
- Vagrant
- Docker's native client for Mac OS X.

Virtualbox will be used to run an Ubuntu linux virtual machine. Vagrant will be used to start, stop and initially build the virtual machine with docker installed.

The native docker client for Mac OS X will be used to do all docker related activities.

Step 0 - Prepare your system for a docker dev environment

Create a directory for your docker development. I use `~/docker_dev`

```
[~]$ mkdir ~/docker_dev
```

Create a directory for local box files

```
[~]$ mkdir ~/docker_dev/boxes
```

Step 1 - Install Vagrant

Install Vagrant (remove old versions first).

Download the latest version of Vagrant from the Vagrant web site

Installation instructions are on the Vagrant site here

Step 2 - Install the docker client

Download and Install Mac OS X docker client from the Docker web site

```
[~]$ cd ~/docker_dev
[docker_dev]$ curl -o docker
http://get.docker.io/builds/Darwin/x86_64/docker-latest
```

Mark it executable.

```
[docker_dev]$ chmod +x docker
```

Move the executable file to somewhere on your path.

```
[docker_dev]$ sudo mv docker /usr/local/bin/
```

Check that it works.

```
[docker_dev]$ docker version
Client version: 0.8.0
Go version (client): go1.2
Git commit (client): cc3a8c8
2014/02/08 00:08:25 dial unix /avr/run/docker.sock: no such file or
directory
```

The client works, but cannot connect a docker daemon. That last line is ok for now, since we don't yet have our docker instance running.

Step 3 - Download a prepared vm box file

The good folks at Phusion have a number of pre-made vagrant boxes. For my dev environment I prefer to download and save a box file locally, for quicker build times.

Details on these boxes is available here

I used the ubuntu-12.04.3-amd64 box file for my environment.

Download the file and save it to the `~/docker_dev/boxes` directory.

All the Phusion virtual machines have a user named vagrant, and the password is vagrant.

The root password is also vagrant.

Step 4 - Get a local web service up and running

To quickly access box files locally I find it handy to use a small web server. However, you can also simply use local file access, check out the documentation on the vagrant web site

While strictly not necessary, I usually have one of these setup and running.
The one that I use is called Mongoose. However there are many others out there.

- Mongoose - available here

You could also try Pow from 37Signals.

- POW - get it from here

Install Mongoose or Pow or your favourite and point it/set the default document directory to ~/docker_dev/boxes

Step 5 - Create your Vagrantfile

Next, we will create a configuration file that starts up a virtual machine, and installs docker.

```
[~]$ cd ~/docker_dev
[docker_dev]$ touch Vagrantfile
```

Copy and paste the following into the file, and save it to your docker directory, The file name ***must*** be ***Vagrantfile***

```
#Vagrantfile for Docker install
#
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
    config.vm.box = "phusion-open-ubuntu-12.04-amd64"
    config.vm.box_url = "http://127.0.0.1:8080/ubuntu-12.04.3-amd64-
vbox.box"

    config.vm.network "private_network", ip: "10.2.0.10", netmask:
"255.255.0.0"
    config.vm.provider :virtualbox do |vb|
        vb.customize ["modifyvm", :id, "--nicpromisc2", "allow-all"]
    end

    config.vm.network "forwarded_port", guest: 4243, host: 4243

    if Dir.glob("#
{File.dirname(__FILE__)}/.vagrant/machines/default/*/id").empty?
    # Install Docker
        pkg_cmd = "wget -q -O - https://get.docker.io/gpg | apt-key add -;"
\
        "echo deb http://get.docker.io/ubuntu docker main >
/etc/apt/sources.list.d/docker.list;" \
        "apt-get update -qq; apt-get install -q -y --force-yes lxc-docker; "
        # Add vagrant user to the docker group
        pkg_cmd << "usermod -a -G docker vagrant; "
        config.vm.provision :shell, :inline => pkg_cmd
    end
end
## End of Vagrantfile
```

Note that on line 7, `config.vm.box_url` points to your localhost and the port is defaulted to 8080. If you have configured your local web server to listen on a different port you should

change the setting in the Vagrantfile to use whatever you have setup.

Step 6 - Launch your virtual machine

To launch the virtual machine simply run the command

```
[docker_dev]$ vagrant up
```

The very first time this command is executed it will take a few minutes, as the virtual machine is launched and docker is installed. Subsequent runs will be quicker.

Once the virtual machine is running, we need to chcek the ip address that the docker daemon is using. Run the following command

```
[docker_dev]$ vagrant ssh
```

This will connect you to the virtual machine. Next execute the following command

```
vagrant@ubuntu-12:~$ ifconfig
```

You should see output like the following:

```
vagrant@ubuntu-12:~$ ifconfig
docker0   Link encap:Ethernet  HWaddr 6a:13:c0:d1:db:83
          inet addr:172.17.42.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::414:96ff:fed4:c6ab/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:104562 errors:0 dropped:0 overruns:0 frame:0
          TX packets:105277 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:39536298 (39.5 MB)  TX bytes:7467093 (7.4 MB)

eth0      Link encap:Ethernet  HWaddr 08:00:27:84:1f:23
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe84:1f23/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5370 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4708 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1009251 (1.0 MB)  TX bytes:492507 (492.5 KB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:77:a9:5f
          inet addr:10.2.0.10  Bcast:10.2.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:104883 errors:0 dropped:0 overruns:0 frame:0
          TX packets:103803 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:1000
          RX bytes:6958172 (6.9 MB)  TX bytes:40939790 (40.9 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:96 errors:0 dropped:0 overruns:0 frame:0
          TX packets:96 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:7664 (7.6 KB)  TX bytes:7664 (7.6 KB)

lxcbr0    Link encap:Ethernet  HWaddr ca:97:37:a2:16:f2
          inet addr:10.0.3.1  Bcast:10.0.3.255  Mask:255.255.255.0
          inet6 addr: fe80::c897:37ff:fea2:16f2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:468 (468.0 B)

vethb2u3MC Link encap:Ethernet  HWaddr 6a:13:c0:d1:db:83
          inet6 addr: fe80::6813:c0ff:fed1:db83/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:104371 errors:0 dropped:0 overruns:0 frame:0
          TX packets:104722 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:40980002 (40.9 MB)  TX bytes:7430711 (7.4 MB)

vagrant@ubuntu-12:~$
```

The information we are looking for is the IP address for inet in the docker0 section at the start, and in the example above the IP address is 172.17.42.1

Next log out of the VM using the command `exit`

```
vagrant@ubuntu-12:~$ exit
logout
Connection to 127.0.0.1 closed.
[docker_dev]$
```

We next need create a route on our computer to the docker vm. Basically, any traffic trying to reach the docker subnet (172.17.0.0) should be routed to our new interface inside the vm (10.2.0.10)

```
[docker_dev]$ sudo route -n add -net 172.17.0.0 10.2.0.10
```

Step 7 Set environment variable for docker client

```
#Set the docker host env parameter
[docker_dev]$ export DOCKER_HOST=172.17.42.1

[docker_dev]$ echo $DOCKER_HOST
172.17.42.1
[docker_dev]$
```

I'd advise that you add `DOCKER_HOST=172.17.42.1` to your bash or zsh profile so it's always set when you open a terminal session.

Step 8 Configure docker to listen on TCP and local socket

In this step we will change the docker daemon configuration so that it listens on a TCP port, so that we can access it from our terminal directly.

We need to ssh into the VM and edit `/etc/init/docker.conf`

```
[docker_dev]$ vagrant ssh
vagrant@ubuntu-12:~$ sudo vi /etc/init/docker.conf
```

Edit the file by adding `-H 0.0.0.0:4243 -H unix:///var/run/docker.sock` to the daemon start-up command

The changed file should look like the following:

```
description "Docker daemon"

start on filesystem and started lxc-net
stop on runlevel [!2345]

respawn

script
  DOCKER=/usr/bin/$UPSTART_JOB
  DOCKER_OPTS=
  if [ -f /etc/default/$UPSTART_JOB ]; then
      . /etc/default/$UPSTART_JOB
  fi
  "$DOCKER" -H 0.0.0.0:4243 -H unix:///var/run/docker.sock -d $DOCKER_OPTS
end script
```

Please note: This is inherentaly insecure for a production system, but for local development I don't have an issue with it. However, please read the notes on the docker site about this.

The relevant section is here

Once you have saved the changes, close the file and exit the VM.

```
vagrant@ubuntu-12:~$ exit
logout
Connection to 127.0.0.1 closed.
[docker_dev]$
```

Step 9 Restart the VM

Restart the VM, this will also restart the docker daemon and set it to listen on the network interfaces for connections.

```
[docker_dev]$ vagrant reload
```

Now lets check the docker version again

```
[docker_dev]$ docker version
Client version: 0.8.0
Go version (client): go1.2
Git commit (client): cc3a8c8
Server version: 0.8.0
Git commit (server): cc3a8c8
Go version (server): go1.2
[docker_dev]$
```

Step 10 Run a docker container

Ok, lets fire up a container

The following command starts a continer and leaves you at the command prompt.

```
 [docker_dev]$ docker run -i -t ubuntu /bin/bash
```

Again, the first time the run it will take a minute or three as it will need to download the base images for the containter, but once done all futher commands will execute within a second or less.

Simply type `exit` to quit the contaier.

There is one issue that I have yet to resolve. After executing the exit command to quit the container, I need to press the enter key twice. However, its trivial and not something thats blocking anything I need to do.

Shared folders on the Virtual Machine

The virtual machine also mounts the directory where the Vagrantfile within the VM as /vagrant So all files located in `[docke_dev]$` are available within the VM. Note: This is the ubuntu virtual machine, NOT the docker containers.

Appendix I

Vagrantfile:

```
#Vagrantfile for Docker install
#
#Vagrantfile API/syntax version. Don't touch unless you know what you're
doing!
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
    config.vm.box = "phusion-open-ubuntu-12.04-amd64"
    config.vm.box_url = "http://127.0.0.1:8080/ubuntu-12.04.3-amd64-
vbox.box"

    config.vm.network "private_network", ip: "10.2.0.10", netmask:
"255.255.0.0"
    config.vm.provider :virtualbox do |vb|
        vb.customize ["modifyvm", :id, "--nicpromisc2", "allow-all"]
    end

    config.vm.network "forwarded_port", guest: 4243, host: 4243

    if Dir.glob("#
{File.dirname(__FILE__)}/.vagrant/machines/default/*/id").empty?
    # Install Docker
        pkg_cmd = "wget -q -O - https://get.docker.io/gpg | apt-key add -;"
\
        "echo deb http://get.docker.io/ubuntu docker main >
/etc/apt/sources.list.d/docker.list;" \
        "apt-get update -qq; apt-get install -q -y --force-yes lxc-docker; "
        # Add vagrant user to the docker group
        pkg_cmd << "usermod -a -G docker vagrant; "
        config.vm.provision :shell, :inline => pkg_cmd
    end
end
## Final
```