DevOps Hands-On

Lab 01 – Docker Machine

- From https://github.com/docker/machine

  Machine lets you create Docker hosts on your computer, on cloud providers, and inside your own data center. It creates servers, installs Docker on them, then configures the Docker client to talk to them.

Lab 01 – Docker Machine

From
https://docs.docker.com/machine/concepts/

To create a virtual machine, you supply Docker Machine with the name of the driver you want to use.

The driver determines where the virtual machine is created.

For example, on a local Mac or Windows system, the driver is typically Oracle VirtualBox.

For provisioning physical machines, a generic driver is provided. For cloud providers, Docker Machine supports drivers such as AWS, Microsoft Azure, Digital Ocean, and many more.

Lab 01– Docker Machine

- docker-machine may already be on your system
- open a terminal and type docker-machine
- if it is not there get it from:

  https://github.com/docker/machine/releases/

Lab 01– Docker Machine

To create a VM in Virtual box that hosts a docker run:

```
docker-machine create -d virtualbox DevOpsLab01
```

When finished run:

```
docker-machine ls
```

Lab 01– Docker Machine

Docker is installed in the new VM we just created.

Next, we can connect Docker running on our machine to the VM we just created running Docker

Lab 01– Docker Machine

Run **docker images** on your machine, you will see the Docker images on your machine

Now, lets tell Docker on our machine to use Docker in the new VM we just created and ran

Lab 01– Docker Machine

To see how to connect our local Docker to the VM created by Docker Machine run command:

docker-machine env DevOpsLab01

this command tells you what to run to connect,

eval $(docker-machine env DevOpsLab01)

Lab 01− Docker Machine

Now, lets run **docker images** again, if everything is setup properly − you will not see any images.

This is because − we have not created any images in the new VM running Docker.

Since our local Docker is now accessing Docker running VM, we can create and run Docker images in the VM from Docker running on our laptop

Lab 01− Docker Machine

Let's create and run hello, world in the VM:

docker run hello-world

Now run **docker images** again − you should see hello,world in Docker running in the VM

Lab 01– Docker Machine

To reset our local Docker back to work with Docker on our machine (and not Docker in the VM) run:

docker-machine env --unset
eval $(docker-machine env --unset)

As you can see:
eval $(docker-machine env DevOpsLab01) **and**
eval $(docker-machine env --unset)

Simply set and unset environment variables

Lab 01– Docker Machine

To stop the VM running Docker:

```
docker-machine kill DevOpsLab01
```

To remove the VM running docker:

```
docker-machine rm DevOpsLab01
```

Lab 02

- Clone the Docker Labs repo from GitHub into a folder on your machine:

  ```
  git clone https://github.com/docker/labs.git
  ```

Lab 02

- cd into folder labs/beginner/static-site

Lab 02

Dockerfile:

```
FROM nginx
ENV AUTHOR=Docker

WORKDIR /usr/share/nginx/html
COPY Hello_docker.html /usr/share/nginx/html

CMD cd /usr/share/nginx/html && sed -e s/Docker/"$AUTHOR"/ Hello_docker.html >
index.html ; nginx -g 'daemon off;'
```

Lab 02

Build the Docker images:

docker build .

Let's tag the new image:

docker images #to get image-id, should be at the top of the list

tag b27c46d4bed4 staticsite:1.0

#NOTE: your image id will be different

Lab 02

Now let's run the image in detached mode:

```
docker run -d -P b27c46d4bed4
#again your image-id will be different
```

Lab 02

Now let's run the image in detached mode:

docker run -d -P b27c46d4bed4
#again your image-id will be different

-d is run detached
-P is publish all exposed network ports to rand ports in the host

Lab 02

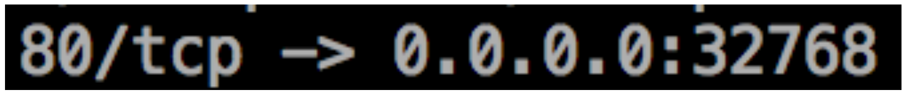
Now let's get the ports that were published using −P:

```
docker ps #get the container id
```

Lab 02

docker port faefa023be00

#your container id will be different

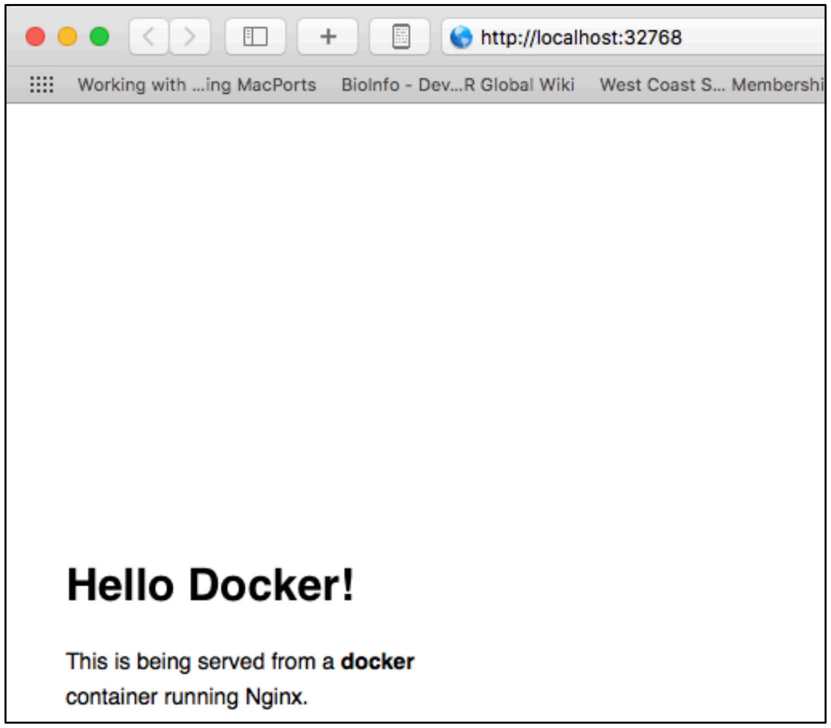


```
80/tcp -> 0.0.0.0:32768
```

Lab 02

Using a web browser, access the static web site:
http://localhost:32786

NOTE: your port may be different


http://localhost:32768
Working with ...ing MacPorts
BioInfo - Dev...R Global Wiki
West Coast S... Membershi

Hello Docker!

This is being served from a **docker** container running Nginx.

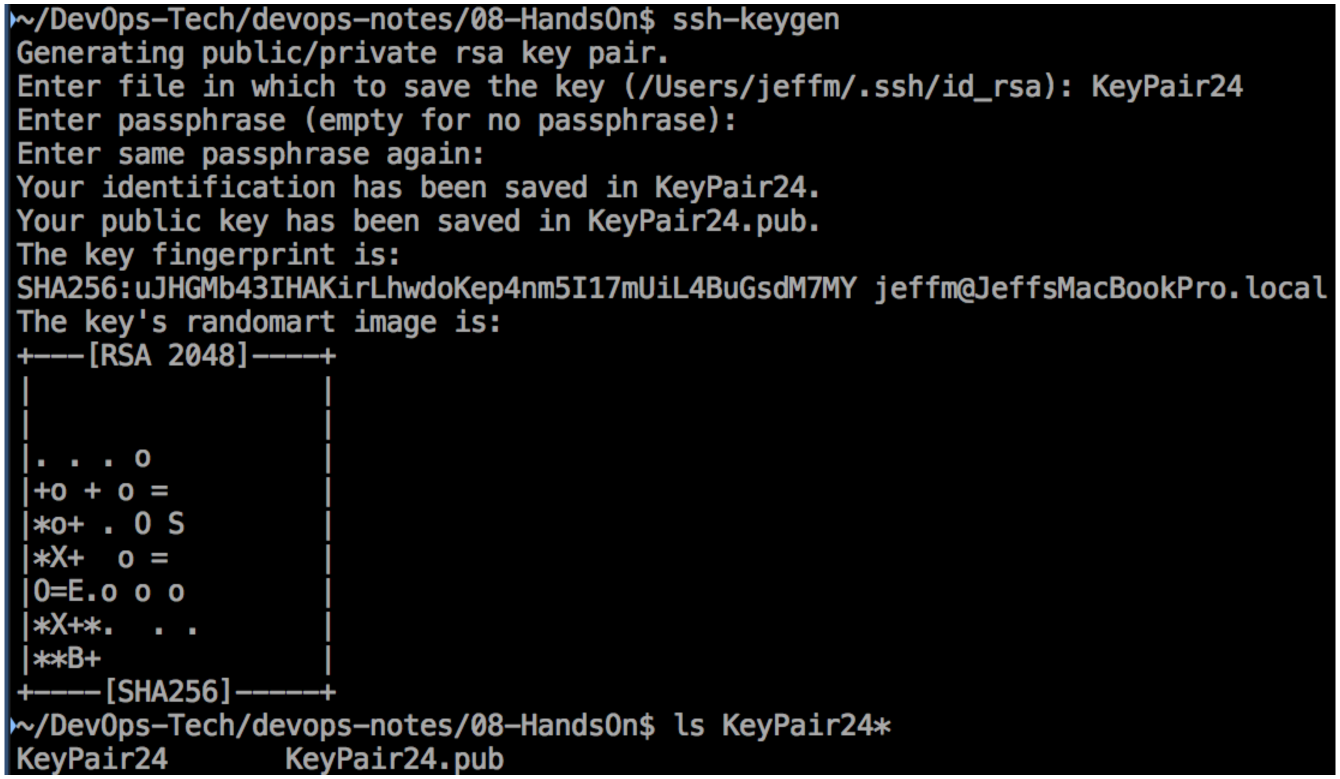
Lab 03 – Docker in AWS

NOTE: you will not be able to run the code in this lab until you:

- signup for an AWS account
- download and install the aws cli
- create and download AWS access keys (not the same as ssh keys)
- configure the AWS CLI to use the downloaded AWS access keys

Lab 03 – Docker in AWS

Create both public an private ssh keys using ssh-keygen



```
~/DevOps-Tech/devops-notes/08-HandsOn$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/jeffm/.ssh/id_rsa): KeyPair24
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in KeyPair24.
Your public key has been saved in KeyPair24.pub.
The key fingerprint is:
SHA256:uJHGMb43IHAKirLhwdoKep4nm5I17mUiL4BuGsdM7MY jeffm@JeffsMacBookPro.local
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|                 |
|. . . o          |
|+o + o =         |
|*o+ . O S        |
|*X+  o =         |
|O=E.o o o        |
|*X+*.  . .       |
|**B+             |
+----[SHA256]-----+
~/DevOps-Tech/devops-notes/08-HandsOn$ ls KeyPair24*
KeyPair24       KeyPair24.pub
```

just hit return
for passpharse

KeyPair24 is the
private key

KeyPair24.pub is
the public key

Lab 03 – Docker in AWS

create vm running docker in aws – the items in bold will have to be changed to match your values in ec2:

#NOTE: the command must be all on 1 line when you run it.

```
docker-machine -D create
--driver amazonec2
--amazonec2-vpc-id vpc-eb0eb48c
--amazonec2-subnet-id subnet-d8044cbf
--amazonec2-security-group awsclass01
--amazonec2-region us-west-1
--amazonec2-instance-type t2.micro
--amazonec2-zone "a"
--amazonec2-ssh-keypath ./KeyPair24
ucsc.devops.docker
```

Lab 03 – Docker in AWS

```
#see the docker running in aws in our local docker machine's images
docker-machine ls

#have our local Docker console point to the docker running in AWS
eval $(docker-machine env ucsc.devops.docker)

#run hello-world in the AWS docker
docker run hello-world

#redirect our local Docker to access the Docker service locally
eval $(docker-machine env -unset)
```

Lab 03 – Docker in AWS

#get the ip address of the VM running Docker in AWS
docker-machine ip ucsc.devops.docker

#ssh into the VM running Docker in AWS using docker-machine
docker-machine ssh ucsc.devops.docker

#look at the Docker images in our AWS based Docker in the ssh terminal
sudo docker images

#logout of ssh
logout

Lab 03 – Docker in AWS

```
#Stop the VM running Docker in AWS using docker-machine
docker-machine stop ucsc.devops.docker

#terminate the VM using docker-machine
docker-machine rm -y ucsc.devops.docker
```