IBM Integration Bus V10.0

IIB Docker Container Deployment

Part A – Get and Install Docker on Mac OS X

Part B – Download, Start and test the IIB Docker Container

V1.4 February 2015

Table of Contents

Overview

Description

The steps in this lab will take you through set and getting started with IIB Docker containers

You will complete the following steps. If you are an existing Docker user you will be able to jump ahead of the initial instructions.

1. Part 1 – Get and install Docker
    a. Get a Docker ID
    b. Get and install Docker on the operating system of your choice
    c. Verify the Docker install and set up
2. Part 2 – Instantiate and verify the IIB runtime Docker container
    a. Obtain and instantiate the IIB runtime Docker container
    b. Verify the running container
    c. Verify the running IIB node in the container

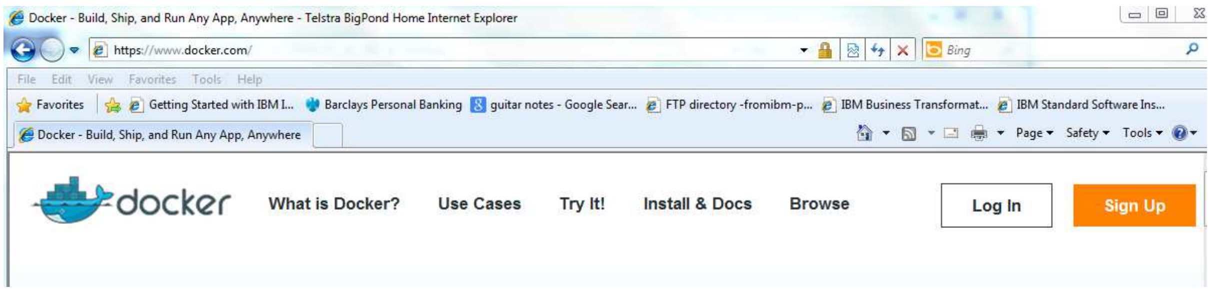
Pre-requisites

None

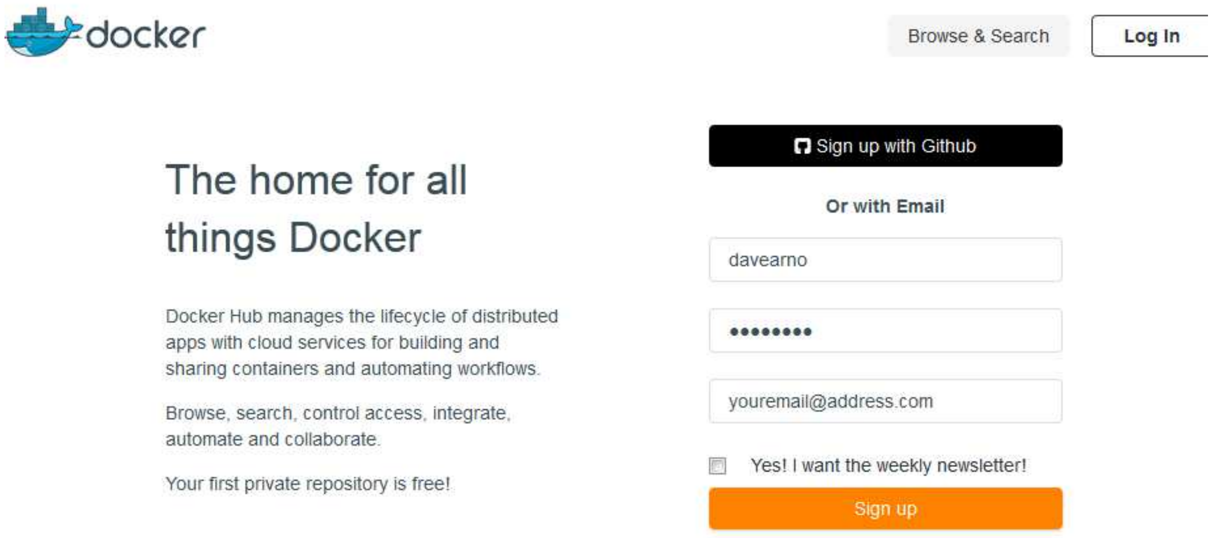
Part A – Get and Install

Get a Docker ID

Go to the Docker website https://www.docker.com/


Docker - Build, Ship, and Run Any App, Anywhere - Telstra BigPond Home Internet Explorer
https://www.docker.com/
File Edit View Favorites Tools Help
Favorites
Getting Started with IBM I...
Barclays Personal Banking
guitar notes - Google Sear...
FTP directory -fromibm-p...
IBM Business Transformat...
IBM Standard Software Ins...
Docker - Build, Ship, and Run Any App, Anywhere
Page
Safety
Tools
docker
What is Docker?
Use Cases
Try It!
Install & Docs
Browse
Log In
Sign Up

Select Sign Up


Browse & Search

Log In

The home for all things Docker

Docker Hub manages the lifecycle of distributed apps with cloud services for building and sharing containers and automating workflows.

Browse, search, control access, integrate, automate and collaborate.

Your first private repository is free!

Sign up with Github

Or with Email

davearno

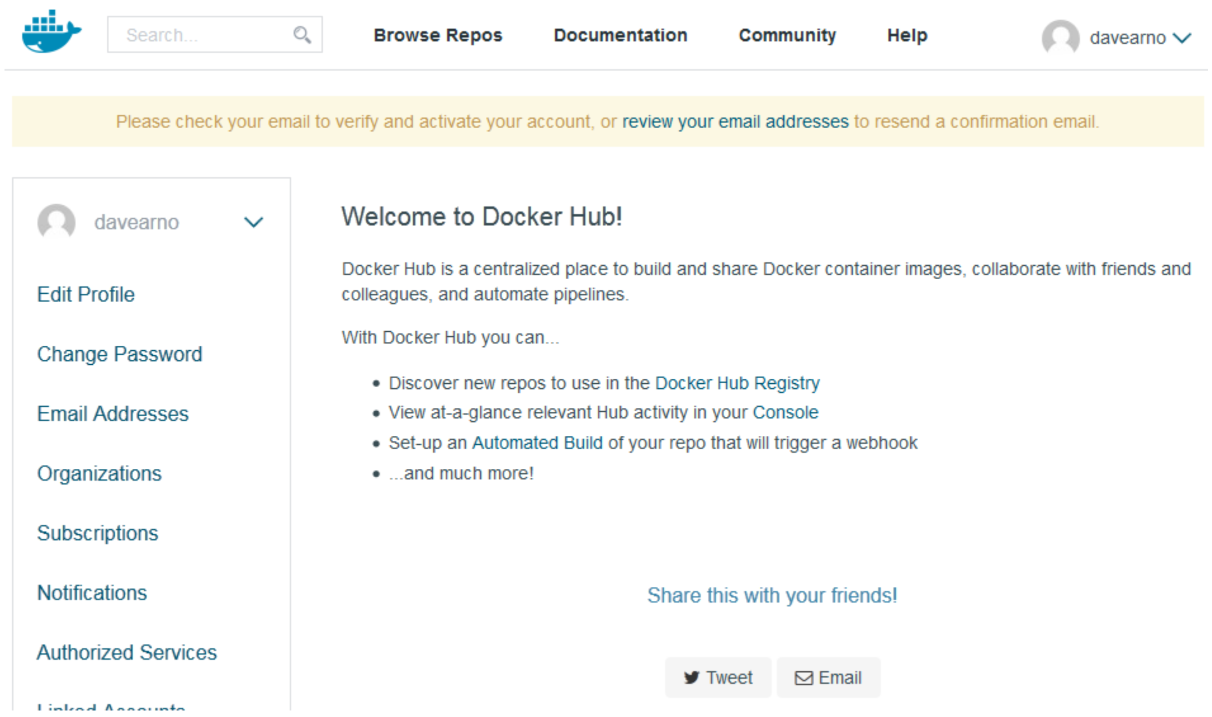
••••••••

youremail@address.com

Yes! I want the weekly newsletter!

Sign up

Choose a Docker ID name and password and supply and email address


Search...

Browse Repos

Documentation

Community

Help

davearno

Please check your email to verify and activate your account, or review your email addresses to resend a confirmation email.

davearno

Edit Profile

Change Password

Email Addresses

Organizations

Subscriptions

Notifications

Authorized Services

Welcome to Docker Hub!

Docker Hub is a centralized place to build and share Docker container images, collaborate with friends and colleagues, and automate pipelines.

With Docker Hub you can...

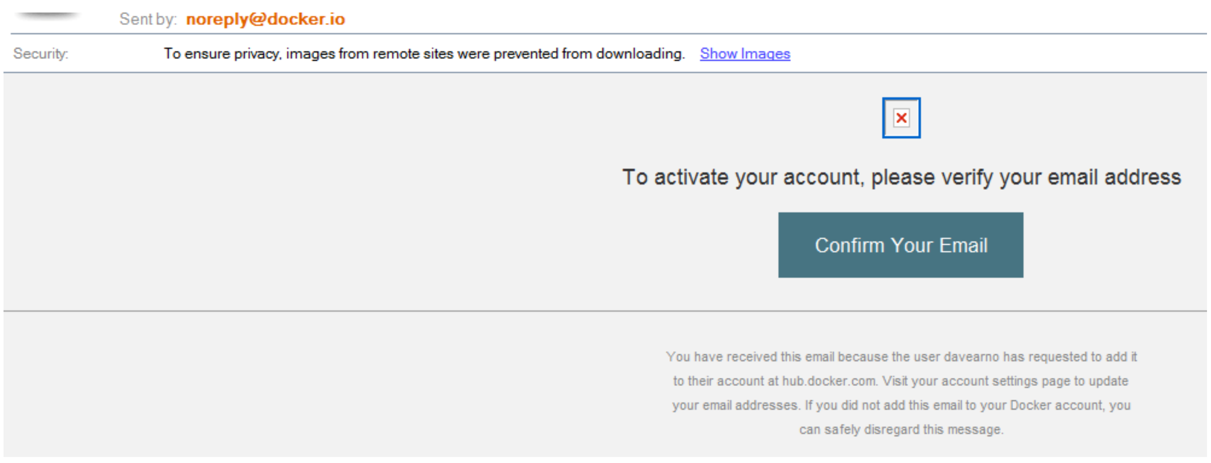
- Discover new repos to use in the Docker Hub Registry
- View at-a-glance relevant Hub activity in your Console
- Set-up an Automated Build of your repo that will trigger a webhook
- ...and much more!

Share this with your friends!

Tweet

Email

You will need to activate the Docker account via the confirmation email.


Sent by: noreply@docker.io

Security: To ensure privacy, images from remote sites were prevented from downloading. Show Images

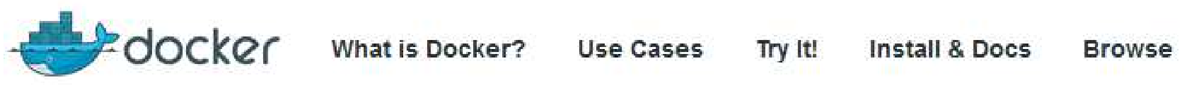
To activate your account, please verify your email address

You have received this email because the user davearno has requested to add it to their account at hub.docker.com. Visit your account settings page to update your email addresses. If you did not add this email to your Docker account, you can safely disregard this message.

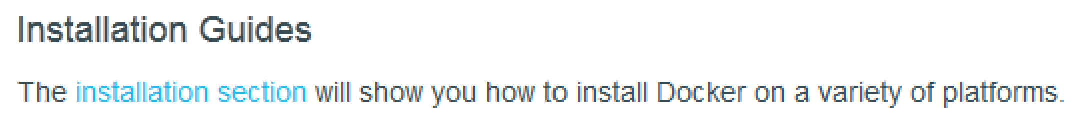
Hit the Confirm Your Email button

Get and install Docker on the operating system of your choice

Return to Docker website https://www.docker.com/



Select Install & Docs


Installation Guides

The installation section will show you how to install Docker on a variety of platforms.

Page down to the Installation Guides and click on the installation section link

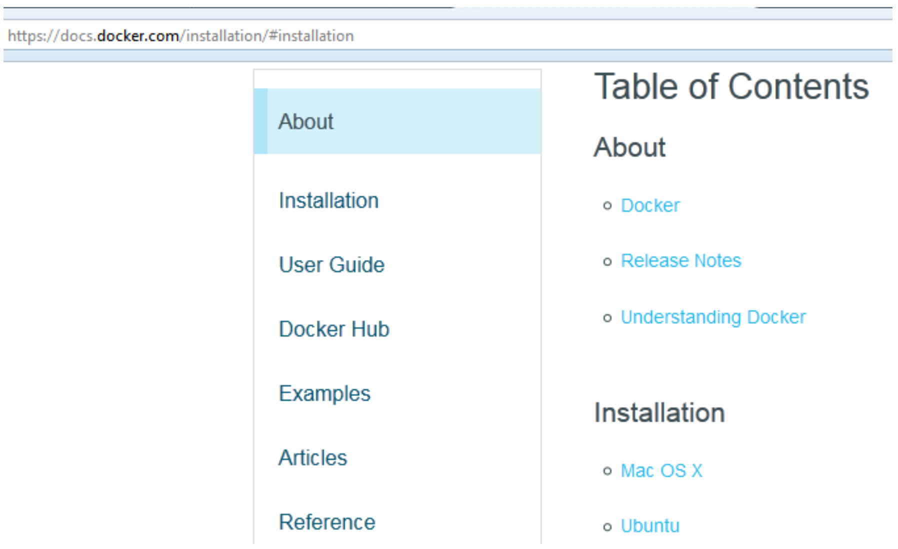

https://docs.docker.com/installation/#installation

Table of Contents

About

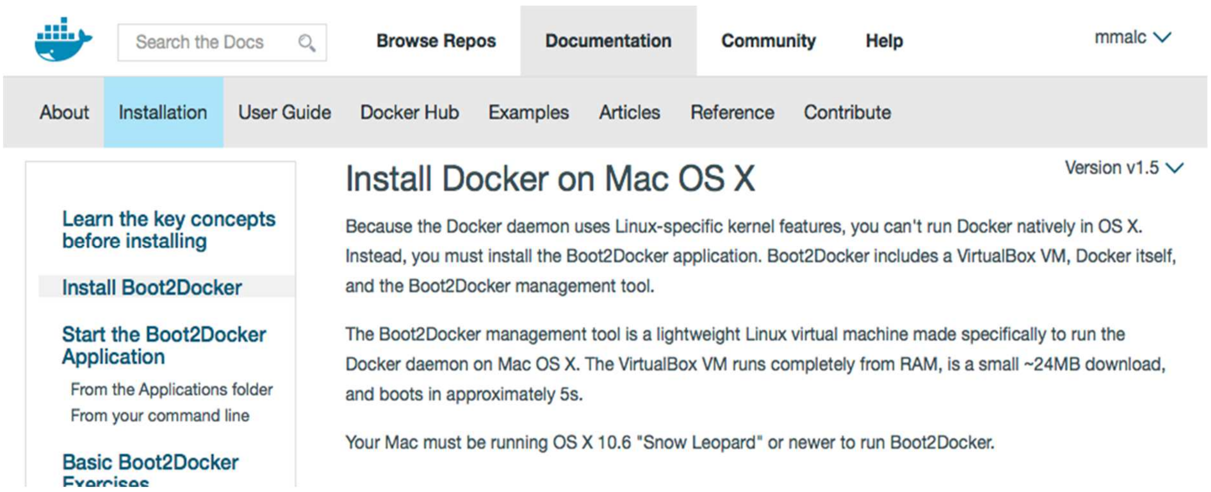
- Docker
- Release Notes
- Understanding Docker

Installation

- Mac OS X
- Ubuntu

Select your target platform

We will use Mac OS X for example purposes.
Click on Mac OS X



Install Docker on Mac OS X

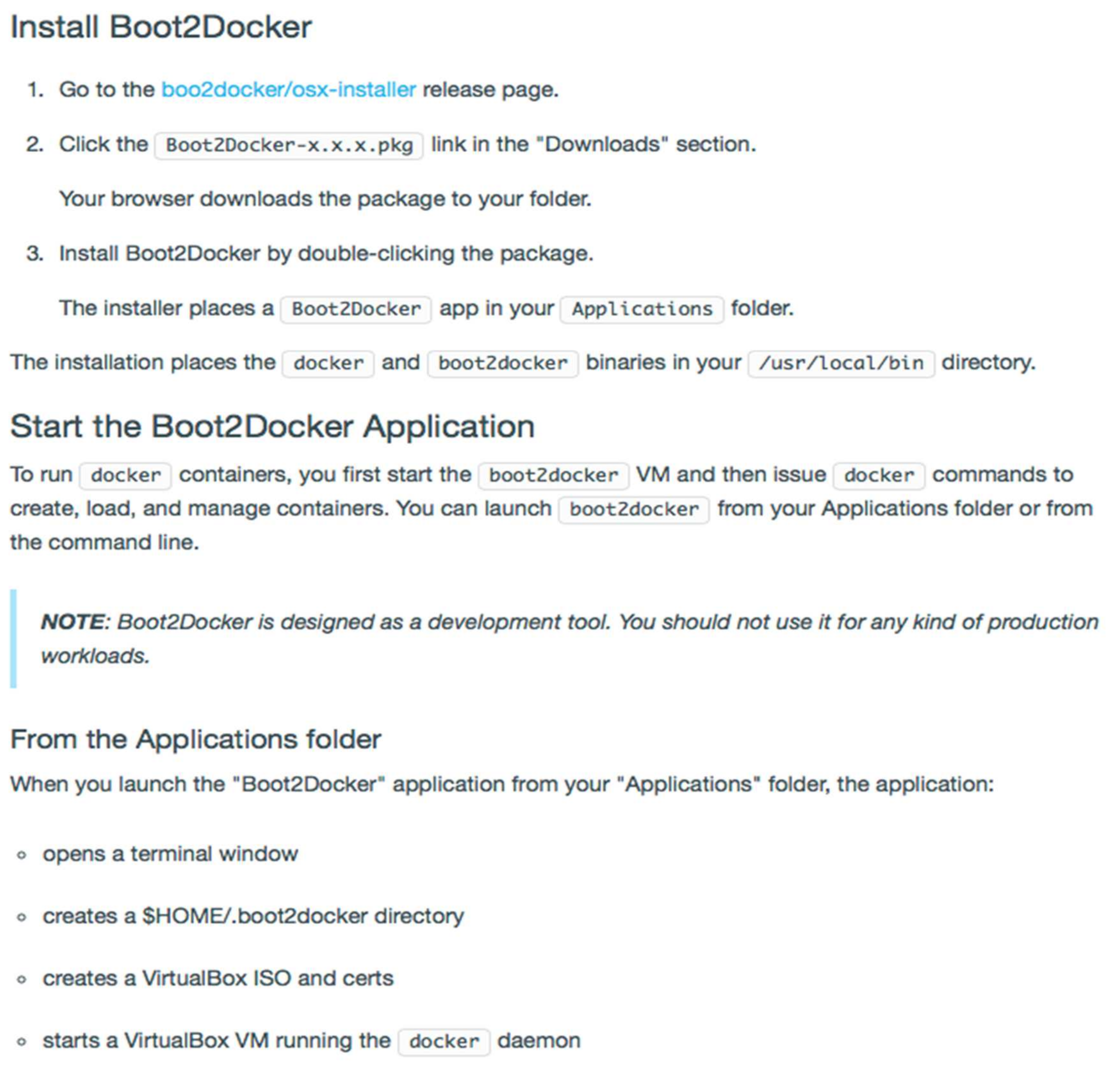
Because the Docker daemon uses Linux-specific kernel features, you can't run Docker natively in OS X. Instead, you must install the Boot2Docker application. Boot2Docker includes a VirtualBox VM, Docker itself, and the Boot2Docker management tool.

The Boot2Docker management tool is a lightweight Linux virtual machine made specifically to run the Docker daemon on Mac OS X. The VirtualBox VM runs completely from RAM, is a small ~24MB download, and boots in approximately 5s.

Your Mac must be running OS X 10.6 "Snow Leopard" or newer to run Boot2Docker.

Page down to the Install Boot2Docker and follow the instructions.

Ignoring the command that says docker hello-world.



Install Boot2Docker

1. Go to the boo2docker/osx-installer release page.
2. Click the `Boot2Docker-x.x.x.pkg` link in the "Downloads" section.

   Your browser downloads the package to your folder.
3. Install Boot2Docker by double-clicking the package.

   The installer places a `Boot2Docker` app in your `Applications` folder.

The installation places the `docker` and `boot2docker` binaries in your `/usr/local/bin` directory.

Start the Boot2Docker Application

To run `docker` containers, you first start the `boot2docker` VM and then issue `docker` commands to create, load, and manage containers. You can launch `boot2docker` from your Applications folder or from the command line.

> ***NOTE**: Boot2Docker is designed as a development tool. You should not use it for any kind of production workloads.*

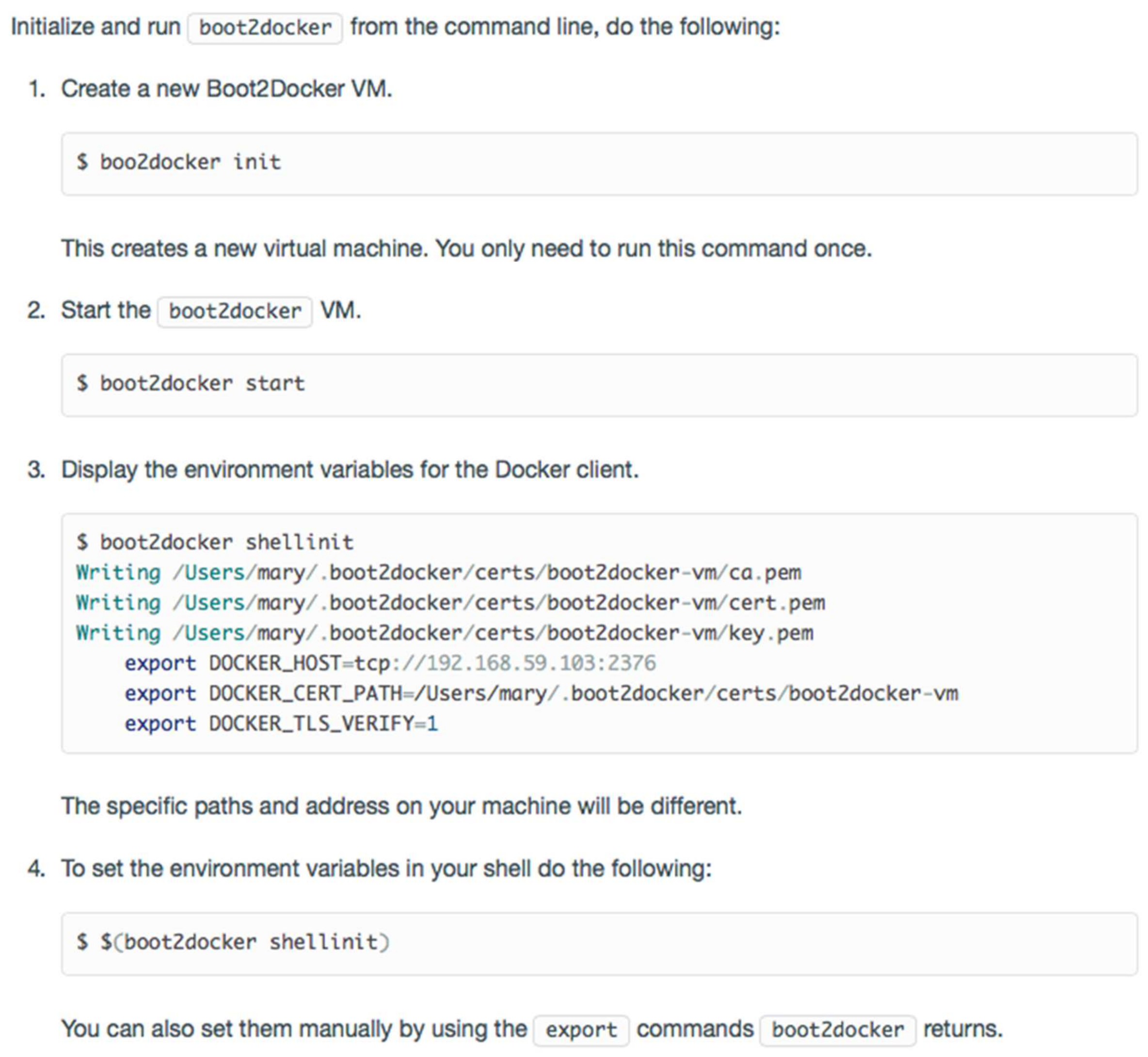
From the Applications folder

When you launch the "Boot2Docker" application from your "Applications" folder, the application:

- opens a terminal window
- creates a $HOME/.boot2docker directory
- creates a VirtualBox ISO and certs
- starts a VirtualBox VM running the `docker` daemon

Starting up the boot2docker environment

Typically this is done from the command line



Initialize and run `boot2docker` from the command line, do the following:

1. Create a new Boot2Docker VM.

   ```
   $ boo2docker init
   ```

   This creates a new virtual machine. You only need to run this command once.

2. Start the `boot2docker` VM.

   ```
   $ boot2docker start
   ```

3. Display the environment variables for the Docker client.

   ```
   $ boot2docker shellinit
   Writing /Users/mary/.boot2docker/certs/boot2docker-vm/ca.pem
   Writing /Users/mary/.boot2docker/certs/boot2docker-vm/cert.pem
   Writing /Users/mary/.boot2docker/certs/boot2docker-vm/key.pem
       export DOCKER_HOST=tcp://192.168.59.103:2376
       export DOCKER_CERT_PATH=/Users/mary/.boot2docker/certs/boot2docker-vm
       export DOCKER_TLS_VERIFY=1
   ```

   The specific paths and address on your machine will be different.

4. To set the environment variables in your shell do the following:

   ```
   $ $(boot2docker shellinit)
   ```

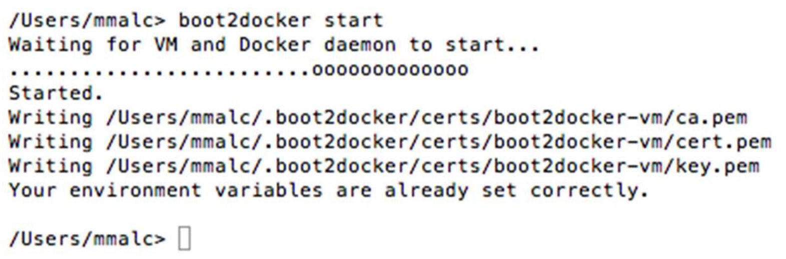
   You can also set them manually by using the `export` commands `boot2docker` returns.

It's worth adding the shell initialisation to your start up profile for future sessions.

e.g. I added the following to my .bashrc script

```
$(boot2docker shellinit)
```

Now when starting the boot2docker I see the following



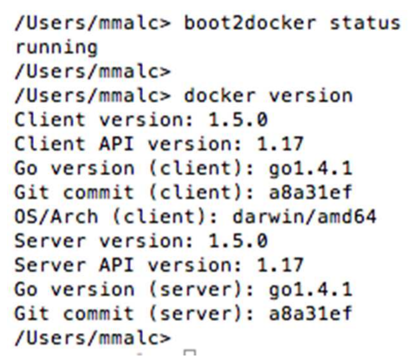
```
/Users/mmalc> boot2docker start
Waiting for VM and Docker daemon to start...
.........................ooooooooooooo
Started.
Writing /Users/mmalc/.boot2docker/certs/boot2docker-vm/ca.pem
Writing /Users/mmalc/.boot2docker/certs/boot2docker-vm/cert.pem
Writing /Users/mmalc/.boot2docker/certs/boot2docker-vm/key.pem
Your environment variables are already set correctly.

/Users/mmalc>
```

Verify the Docker install and set up

Check your Docker version and explore the docker commands

```
> boot2docker status
> docker version
```



```
/Users/mmalc> boot2docker status
running
/Users/mmalc>
/Users/mmalc> docker version
Client version: 1.5.0
Client API version: 1.17
Go version (client): go1.4.1
Git commit (client): a8a31ef
OS/Arch (client): darwin/amd64
Server version: 1.5.0
Server API version: 1.17
Go version (server): go1.4.1
Git commit (server): a8a31ef
/Users/mmalc>
```

Some useful commands for getting help

```
> docker --help
> docker command --help
```

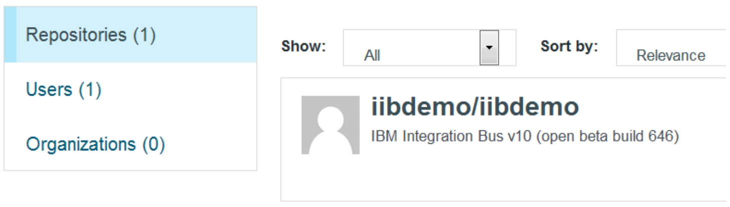
Part B – Instantiate and verify the Docker container

Obtain and instantiate the IIB runtime Docker container

Login to the Docker website https://hub.docker.com/ and search for the IIBDEMO image.

Type iibdemo and hit enter.

You should get the following result



iibdemo/iibdemo
IBM Integration Bus v10 (open beta build 646)

Click on iibdemo/iibdemo to review information about the IIB Demo Docker image.

Pull down the IIBDEMO image from the Docker registry server

The iibdemo image was built with both IIB v10 Beta and IBM MQ for Developers built in. Although no MQ queue managers have been created in the image at this time. Therefore, the image is currently 1.9GB in size.

```
> docker pull iibdemo/iibdemo
```

```
/Users/mmalc> docker pull iibdemo/iibdemo
Pulling repository iibdemo/iibdemo
03949285bd78: Download complete
511136ea3c5a: Download complete
5b12ef8fd570: Download complete
dade6cb4530a: Download complete
cb1b6d0cd2ed: Download complete
Status: Downloaded newer image for iibdemo/iibdemo:latest
```

List the docker images

```
> docker images
```

```
/Users/mmalc> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
iibdemo/iibdemo     latest              03949285bd78        7 days ago          3.087 GB
centos              latest              dade6cb4530a        2 weeks ago         210.1 MB
```

Creating and starting the Docker image as a container for the first time. On creation we are mapping the ports that the IIB runtime will use.

```
> docker run -di --user=iibadm --name=dev_esb01 -p 7080:7080 -p 1414:1414 -
p 4414:4414 -p 6666:6666 -p 7800:7800 --hostname=dev_esb01 --
workdir=/home/iibadm iibdemo/iibdemo:latest /bin/bash -l
```

```
/Users/mmalc> docker run -di --user=iibadm --name=dev_esb01 -p 7080:7080 -p 1414:1414 -p 4414:4414 -p 6666:6666 -p
7800:7800 --hostname=dev_esb01 --workdir=/home/iibadm iibdemo/iibdemo:latest /bin/bash -l
39628cafb5a00f7b6c55a4b9e5600bae3218b40aaa4226eca692173095c72828
```

Start the IIB node in the container and list running containers

```
> docker exec -id dev_esb01 /bin/bash -lc 'iib start TESTNODE'
> docker ps
```

```
/Users/mmalc> docker exec -id dev_esb01 /bin/bash -lc 'iib start TESTNODE'
/Users/mmalc> docker ps
CONTAINER ID        IMAGE                    COMMAND            CREATED             STATUS              PORTS
                                                                                    NAMES
39628cafb5a0        iibdemo/iibdemo:latest   "/bin/bash -l"     50 seconds ago      Up 48 seconds       0.0.0.0:1414->1414/tcp, 0.0.0.0:4414
->4414/tcp, 0.0.0.0:6666->6666/tcp, 0.0.0.0:7080->7080/tcp, 0.0.0.0:7800->7800/tcp   dev_esb01
```

Verify the running container and it's processes

```
> docker top container_name
```

```
/Users/mmalc> docker top dev_esb01
PID                 USER                COMMAND
2334                tc                  /bin/bash -l
2597                tc                  bipservice TESTNODE
2602                tc                  bipbroker TESTNODE
2658                tc                  bipMQTT -c /var/mqsi/components/TESTNODE/config/TESTNODE -p 11883
2672                tc                  DataFlowEngine TESTNODE 6f7e050e-d070-434d-949b-e07237c71e13 service
```

List the mapped ports for the container.

```
> docker port container_name
```

```
/Users/mmalc> docker port dev_esb01
7080/tcp -> 0.0.0.0:7080
7800/tcp -> 0.0.0.0:7800
1414/tcp -> 0.0.0.0:1414
4414/tcp -> 0.0.0.0:4414
6666/tcp -> 0.0.0.0:6666
```

Connect the IIB Web GUI to the running IIB node in the container

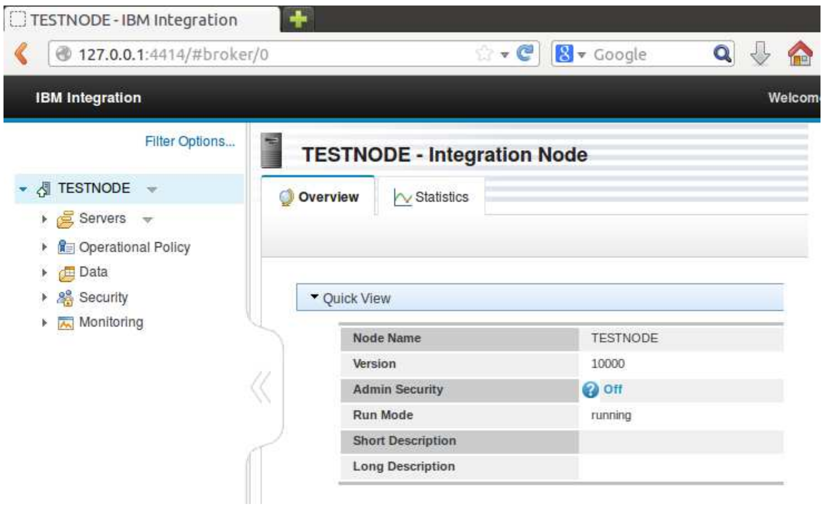
First, on Mac OS X there is a small in-memory boot2docker virtual machine (vm) that is used to host the docker daemon etc. We need this ip address before we can send messages to the docker container.

User the boot2docker ip command to find this ip address

```
> boot2docker ip
```

```
/Users/mmalc> boot2docker ip
192.168.59.103
```

Start your browser and enter the url http://192.168.59.103:4414



TESTNODE - Integration Node

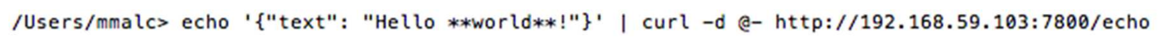
Node Name	TESTNODE
Version	10000
Admin Security	Off
Run Mode	running
Short Description	
Long Description	

Verify the running IIB node in the container

There is a message flow already deployed and running on the IIB Test Node in the IIBDEMO container. The message flow is a simple HTTP echo flow.

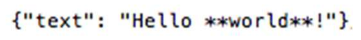
Use curl or your favourite tool to send an HTTP post to the message flow.

```
> echo '{"text": "Hello **world**!"}' | curl -d @-
http://192.168.59.103:7800/echo
```



```
/Users/mmalc> echo '{"text": "Hello **world**!"}' | curl -d @- http://192.168.59.103:7800/echo
```

It should respond with



```
{"text": "Hello **world**!"}
```

Stopping the running IIB node and the container

For reference purposes here are the commands for starting and stopping the container.

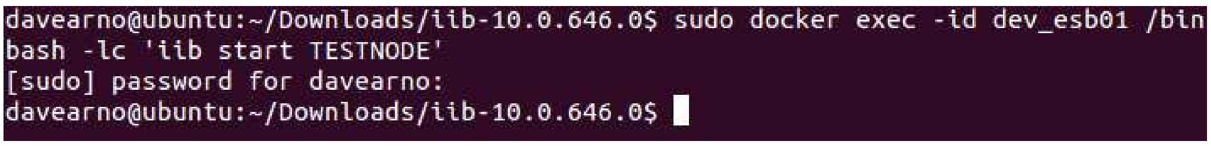
```
> docker exec -id  dev_esb01  "/bin/bash -lc 'iib stop TESTNODE' "

> docker stop dev_esb01
```

Re-starting IIB container and the IIB Node

```
> docker start dev_esb01

> docker exec -id dev_esb01 /bin/bash -lc 'iib start TESTNODE'
```



```
davearno@ubuntu:~/Downloads/iib-10.0.646.0$ sudo docker exec -id dev_esb01 /bin
bash -lc 'iib start TESTNODE'
[sudo] password for davearno:
davearno@ubuntu:~/Downloads/iib-10.0.646.0$
```