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

V1.0 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

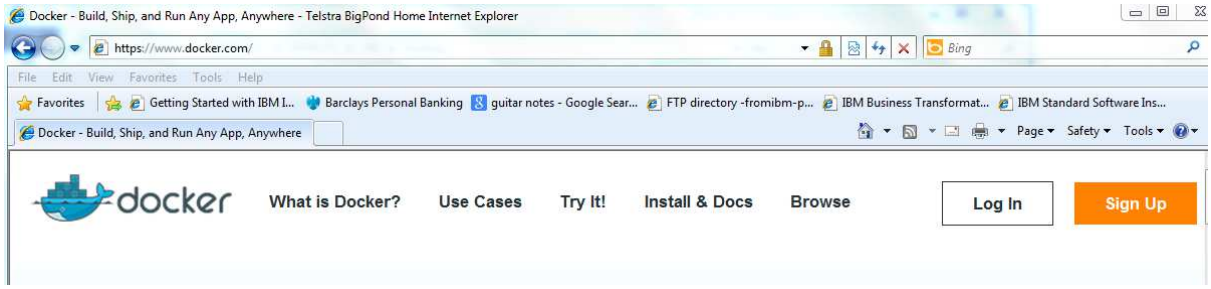
Pre-requisites

None

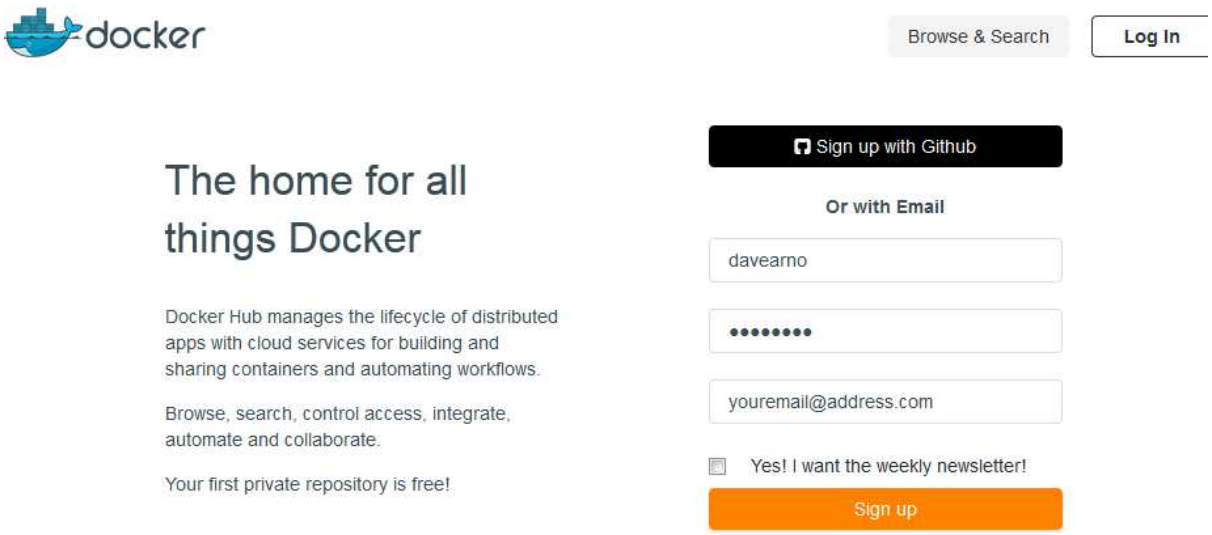
Part A – Get and Install

Get a Docker ID

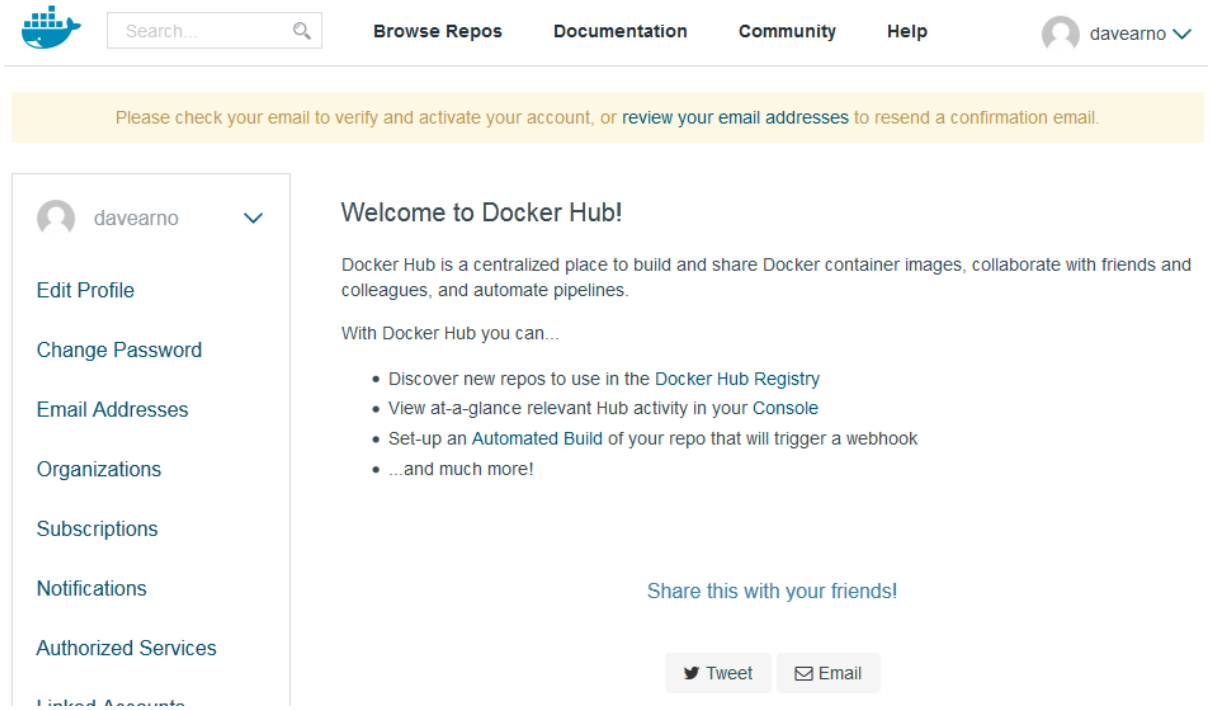
Go to the Docker website https://www.docker.com/



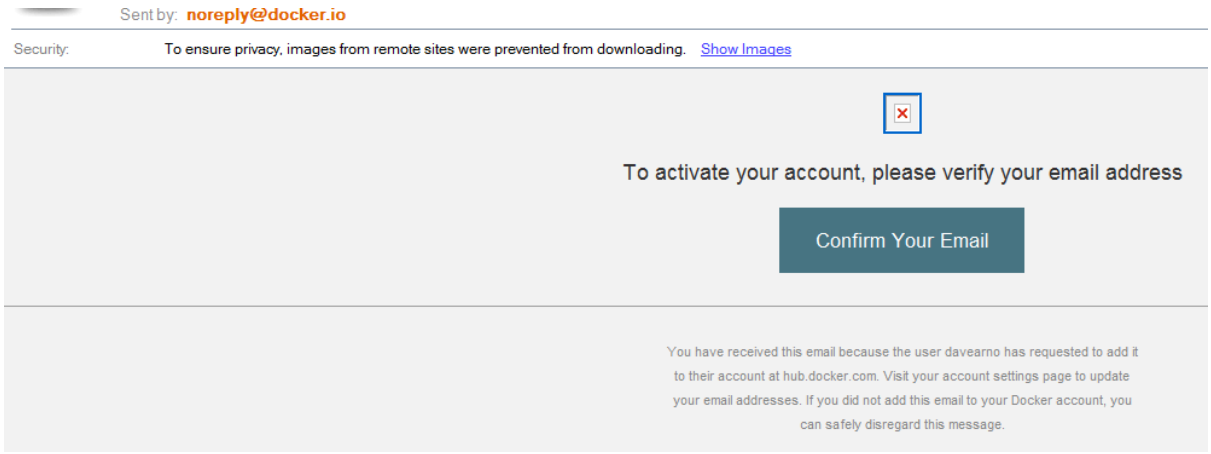
Select Sign Up



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



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

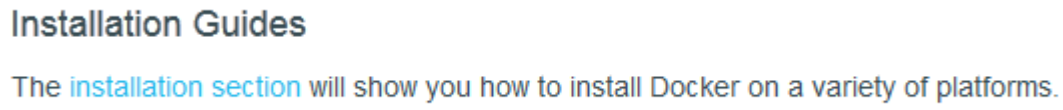


Hit the Confirm Your Email button

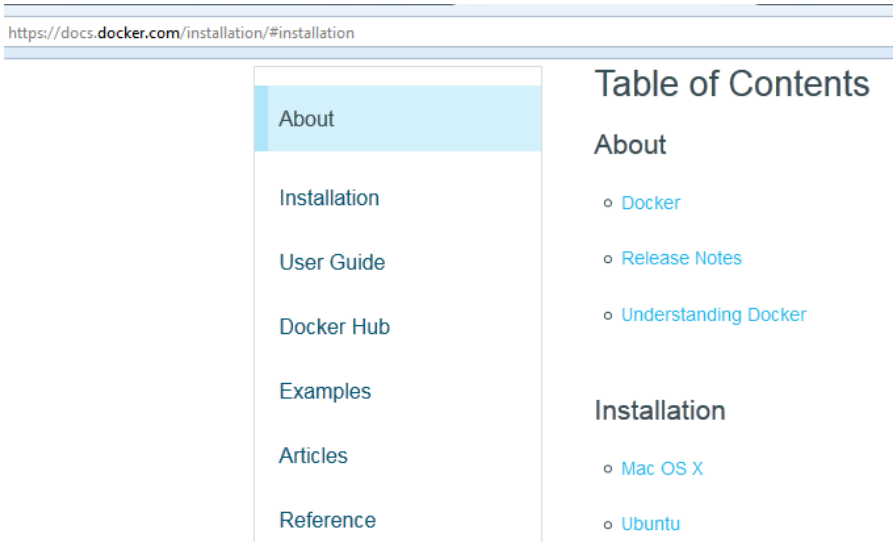
Get and install Docker on the operating system of your choice

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

Select Install & Docs



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



Select your target platform

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

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

Verify the Docker install and set up

Check your Docker version and explore the docker commands

(Note on Mac OS X, the sudo prefix is not usually required)

```
> boot2docker status
```

```
/Users/mmalc> boot2docker status
running
```

```
> docker version
```

```
/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
```

Some useful commands for getting help

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

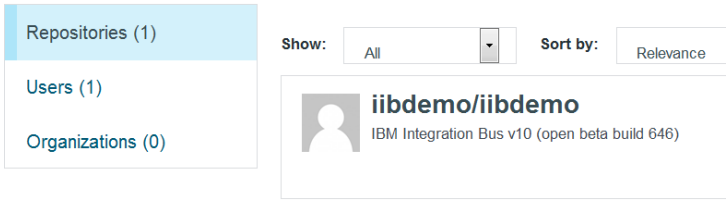
Part B – Instantiate and verify the Docker container

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



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
```

start as a Docker container.

```
> 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
```

LList 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

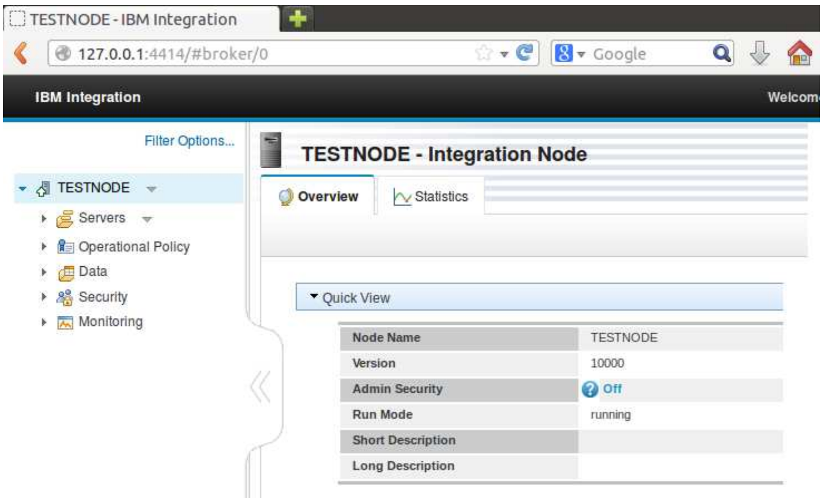
First, on Mac OS X there is a small in-memory boot2docker virtual machine (vm) that is used to host the docker daemon etc. We need this ip address before we can send messages to the docker container.

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



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'
```