

IBM Integration Bus

V10.0 (710)

IIB Docker Container Deployment

Part A – Get and Install Docker

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

V1.5 March 2015

Table of Contents

| | |
|--|----|
| Overview | 3 |
| Description | 3 |
| Pre-requisites | 3 |
| Part A – Get and Install | 3 |
| Get a Docker ID | 3 |
| Get and install Docker – Option 1 – for Ubuntu via Curl | 4 |
| Get and install Docker – Option 2 via the Docker online instructions for all platforms | 5 |
| Verify the Docker install and set up | 7 |
| Part B – Instantiate and verify the Docker container | 9 |
| Obtain and instantiate the IIB runtime Docker container | 9 |
| Pull down the IIBDEMO image from the Docker registry server | 9 |
| Check the status of the IIB container | 10 |
| List running containers | 10 |
| Verify the running container and it's processes | 10 |
| Connect the IIB Web GUI to the running IIB node in the container | 11 |
| Verify the running IIB node in the container | 11 |
| Stopping the running IIB node and the container | 12 |
| Re-starting IIB container and the IIB Node | 12 |

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

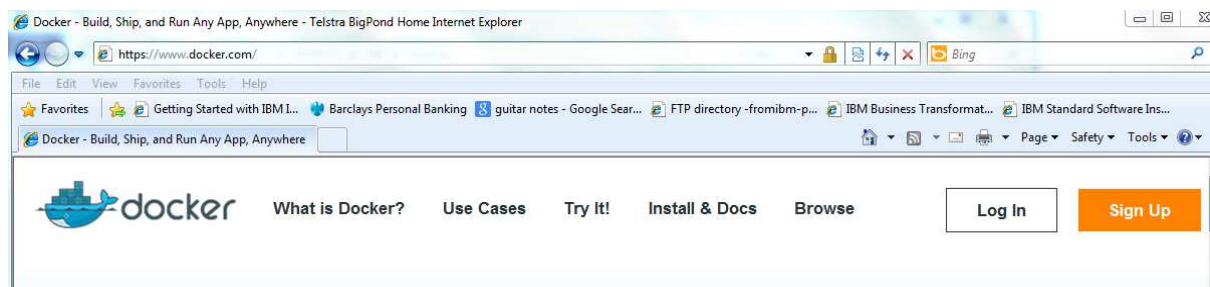
Pre-requisites

None

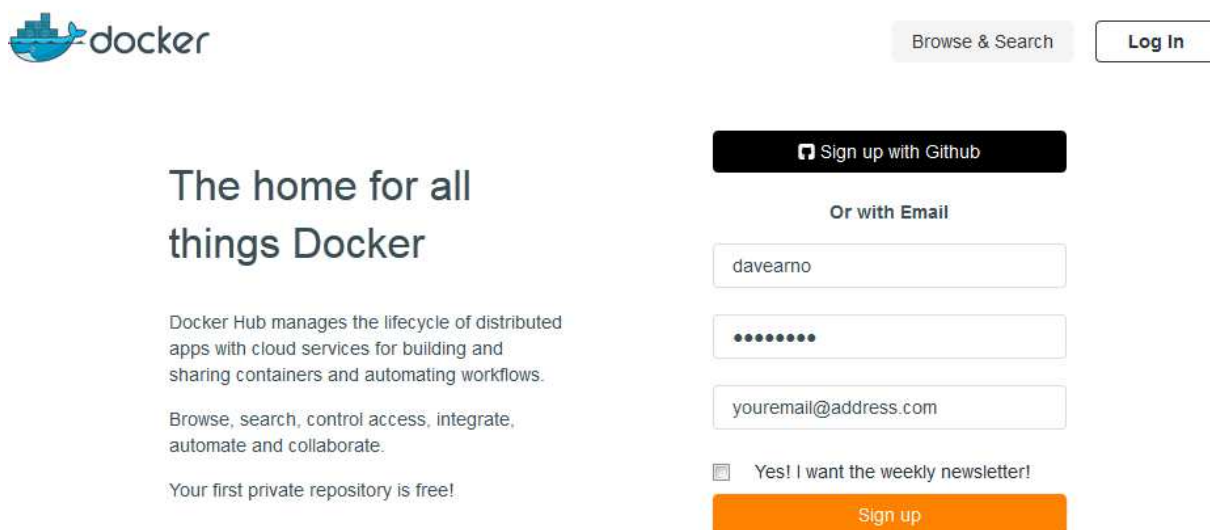
Part A – Get and Install

Get a Docker ID

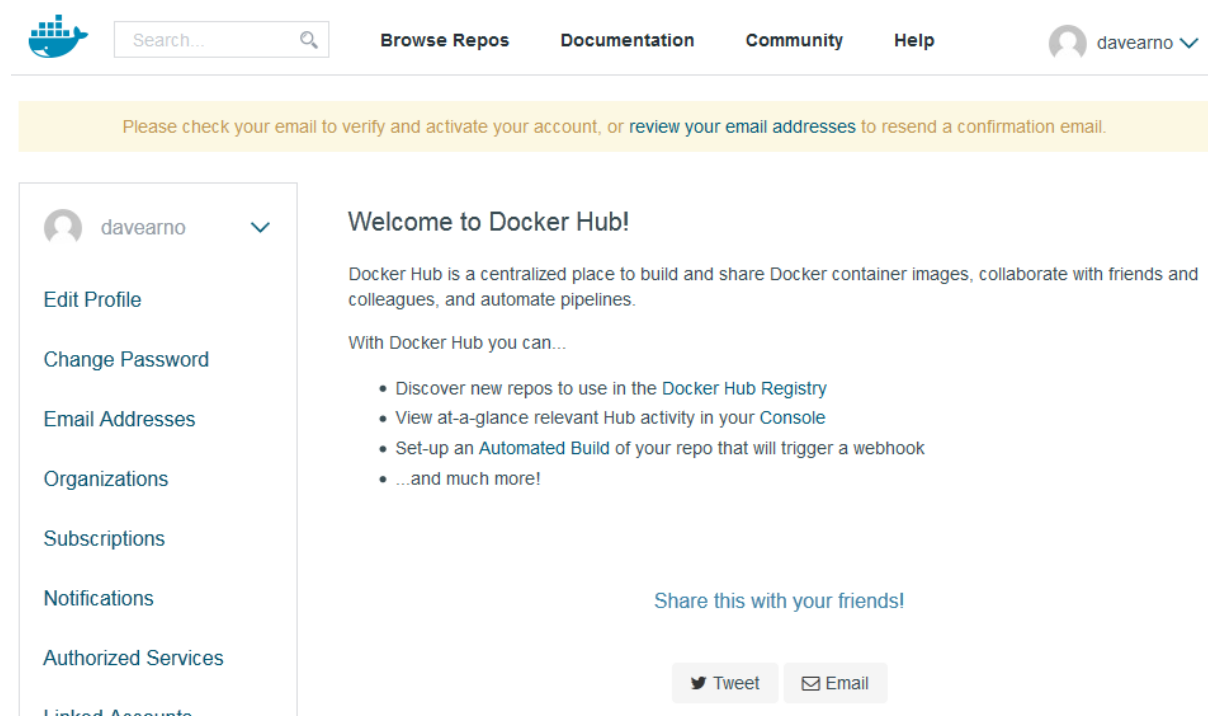
Go to the Docker website <https://www.docker.com/>



Select Sign Up

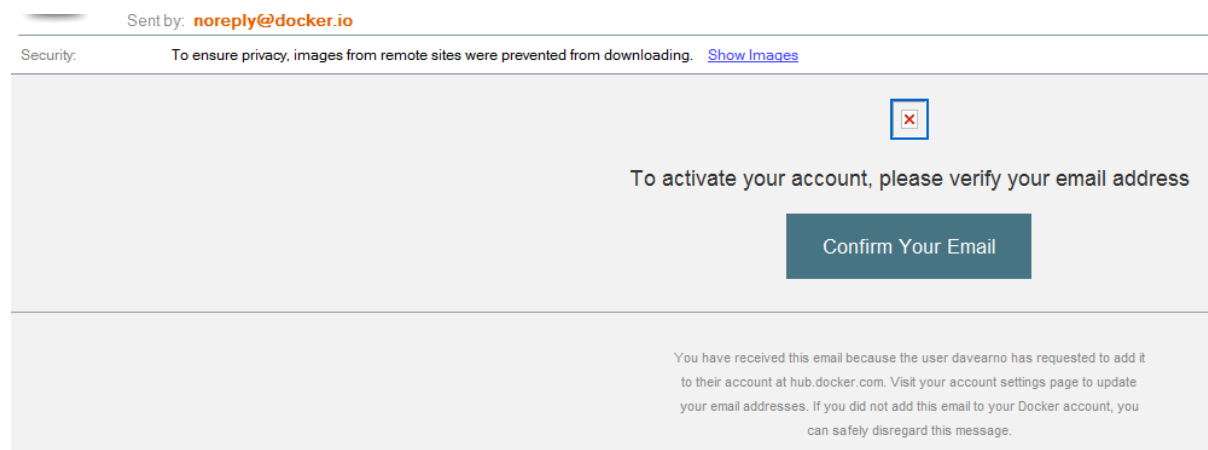


Choose a Docker ID name and password and supply and email address



The screenshot shows the Docker Hub account setup page for a user named 'davearno'. At the top, there's a navigation bar with links for 'Browse Repos', 'Documentation', 'Community', and 'Help'. A search bar is also present. Below the navigation bar, a yellow banner states: 'Please check your email to verify and activate your account, or review your email addresses to resend a confirmation email.' The main content area is titled 'Welcome to Docker Hub!' and includes a brief description of Docker Hub as a centralized place to build and share Docker container images. It also lists features like discovering new repos, viewing Hub activity, and setting up automated builds. A sidebar on the left contains links for 'Edit Profile', 'Change Password', 'Email Addresses', 'Organizations', 'Subscriptions', 'Notifications', and 'Authorized Services'. At the bottom right, there are buttons to 'Share this with your friends!' via 'Tweet' or 'Email'.

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



The screenshot shows an email from 'noreply@docker.io' with the subject 'Security: To ensure privacy, images from remote sites were prevented from downloading. Show Images'. The main body of the email contains a red 'X' icon and the text 'To activate your account, please verify your email address'. Below this is a large blue button labeled 'Confirm Your Email'. At the bottom, there is a disclaimer: '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 – Option 1 – for Ubuntu via Curl

```
$ sudo apt-get install curl
$ curl -sSL https://get.docker.com/ubuntu/ | sudo sh
```

```
davearno@ubuntu:~$ curl -sSL https://get.docker.com/ubuntu/ | sudo sh
Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --homedir /tmp/tmp.7w1pqf4rkm --no-auto-check-trustdb --trust-model always --keyring /etc/apt/trusted.gpg --primary-keyring /etc/apt/trusted.gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 36A1D7869245C8950F966E9208576A8BA88D21E9
gpg: requesting key A88D21E9 from hkp server keyserver.ubuntu.com
gpg: key A88D21E9: "Docker Release Tool (releasedocker) <docker@dotcloud.com>" not changed
gpg: Total number processed: 1
gpg: unchanged: 1
Ign http://us.archive.ubuntu.com trusty InRelease
```

If you installed via option 1 skip forward to the “[Verifying the Docker Install and Set Up](#)” on [page 7](#).

Get and install Docker – Option 2 via the Docker online instructions for all platforms
Return to Docker website <https://www.docker.com/>

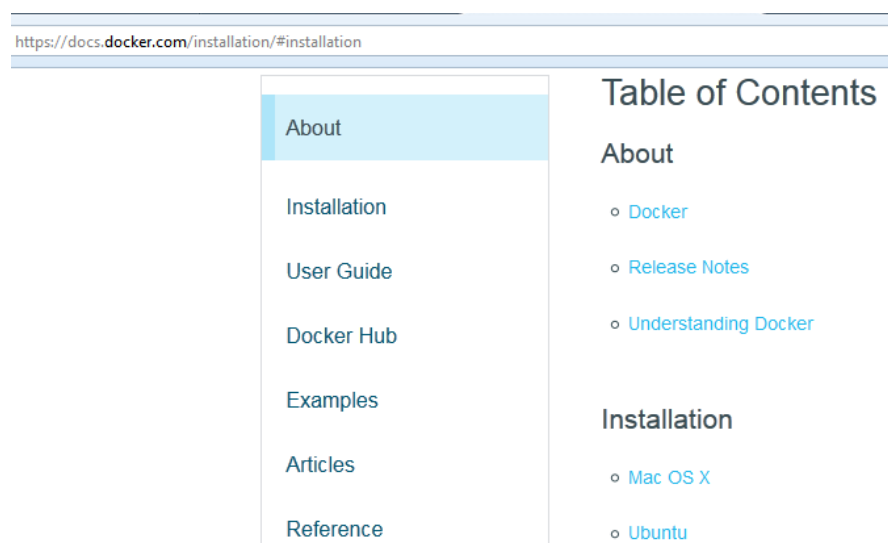


Select Install & Docs

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



Select your target platform

We will use Ubuntu for example purposes.
Click on Ubuntu



Page down to the Docker maintained package installation and follow the instructions.

Docker-maintained Package Installation

If you'd like to try the latest version of Docker:

First, check that your APT system can deal with `https` URLs: the file `/usr/lib/apt/methods/https` should exist. If it doesn't, you need to install the package `apt-transport-https`.

```
[ -e /usr/lib/apt/methods/https ] || {  
  apt-get update  
  apt-get install apt-transport-https  
}
```

Then, add the Docker repository key to your local keychain.

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 36A1D7869245C8950F966E92D8576A8BA88D21E9
```

You can copy and paste the commands from below:

Check that the APT can deal with HTTPS urls

```
$ ls -lt /usr/lib/apt/methods/https  
davearno@ubuntu:~$ ls -lt /usr/lib/apt/methods/https  
-rwxr-xr-x 1 root root 76928 Apr 10 2014 /usr/lib/apt/methods/https
```

Add the Docker repository key to your local keychain.

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys
```

```
36A1D7869245C8950F966E92D8576A8BA88D21E9  
davearno@ubuntu:~$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --  
recv-keys 36A1D7869245C8950F966E92D8576A8BA88D21E9  
[sudo] password for davearno:  
Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --homedir  
/tmp/tmp.jGSfo0LTta --no-auto-check-trustdb --trust-model always --keyring /et  
c/apt/trusted.gpg --primary-keyring /etc/apt/trusted.gpg --keyserver hkp://keyse  
rver.ubuntu.com:80 --recv-keys 36A1D7869245C8950F966E92D8576A8BA88D21E9  
gpg: requesting key A88D21E9 from hkp server keyserver.ubuntu.com  
gpg: key A88D21E9: public key "Docker Release Tool (releasedocker) <docker@dotcl  
oud.com>" imported  
gpg: Total number processed: 1  
gpg: imported: 1 (RSA: 1)
```

Add the Docker repository to your apt sources list, update and install the `lxcd-docker` package. You may receive a warning that the package isn't trusted. Answer yes to continue installation.

```
$ sudo sh -c "echo deb https://get.docker.com/ubuntu docker main\  
> /etc/apt/sources.list.d/docker.list"
```

```
$ sudo apt-get update
```

```

Hit http://security.ubuntu.com trusty-security/universe Sources
Hit http://us.archive.ubuntu.com trusty-updates Release
Hit http://security.ubuntu.com trusty-security/multiverse Sources
Hit http://us.archive.ubuntu.com trusty-backports Release
Hit http://security.ubuntu.com trusty-security/main amd64 Packages
Hit http://us.archive.ubuntu.com trusty/main Sources
Hit http://security.ubuntu.com trusty-security/restricted amd64 Packages
Hit http://us.archive.ubuntu.com trusty/restricted Sources
Hit http://us.archive.ubuntu.com trusty/universe Sources
Hit http://security.ubuntu.com trusty-security/universe amd64 Packages
Hit http://us.archive.ubuntu.com trusty/multiverse Sources
Hit http://security.ubuntu.com trusty-security/multiverse amd64 Packages
Hit http://us.archive.ubuntu.com trusty/main amd64 Packages
Hit http://security.ubuntu.com trusty-security/main i386 Packages
Hit http://us.archive.ubuntu.com trusty/restricted amd64 Packages
Ign https://get.docker.com docker/main Translation-en_US
Ign https://get.docker.com docker/main Translation-en
Fetched 7,610 B in 2min 41s (47 B/s)
Reading package lists... Done
davearno@ubuntu:~$

```

Install Docker on your system:

```
$ sudo apt-get install lxc-docker
```

```

davearno@ubuntu:~$ sudo apt-get install lxc-docker
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  lxc-docker-1.5.0
The following packages will be REMOVED:
  docker.io
The following NEW packages will be installed:
  lxc-docker lxc-docker-1.5.0
0 upgraded, 2 newly installed, 1 to remove and 466 not upgraded.
Need to get 4,635 kB of archives.
After this operation, 9,555 kB disk space will be freed.
Do you want to continue? [Y/n] Y
Get:1 https://get.docker.com/ubuntu/ docker/main lxc-docker-1.5.0
,632 kB]
0% [1 lxc-docker-1.5.0 7,702 B]

```

Verify the Docker install and set up

To verify that everything has worked as expected. This step will pulldown and start a minimal ubuntu docker container. You can cancel it if the download looks time consuming.

```
$ sudo docker run --rm -i -t ubuntu /bin/bash
```

Enable tab-completion on your system:

```
$ source /etc/bash_completion.d/docker.io
```

Finally, add your user to the Docker group

```
$ sudo adduser username docker
```

```

davearno@ubuntu:~$ sudo adduser davearno docker
Adding user 'davearno' to group 'docker' ...
Adding user davearno to group docker
Done.

```

Check your Docker version and explore the docker commands

```
$ sudo docker -version
```

```
davearno@ubuntu:~$ docker --version
Docker version 1.5.0, build a8a31ef
```

Some useful commands for getting help

```
$ sudo docker --help
$ sudo docker command --help
```


Part B – Instantiate and verify the Docker container

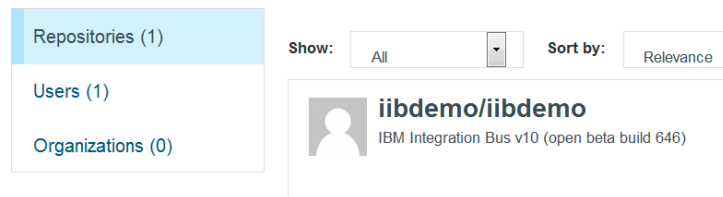
Obtain and instantiate the IIB runtime Docker container

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.

```
$sudo docker pull iibdemo/iibdemo
```

```
davearno@ubuntu:~$ sudo docker pull iibdemo/iibdemo
[sudo] password for davearno:
Pulling repository iibdemo/iibdemo
03949285bd78: Pulling image (latest) from iibdemo/iibdemo, endpoint: https://reg
03949285bd78: Downloading 4.319 MB/1.972 GB 2h22m54s
511136ea3c5a: Download complete
5b12ef8fd570: Download complete
dade6cb4530a: Download complete
cb1b6d0cd2ed: Download complete
```

```
davearno@ubuntu:~$ sudo docker pull iibdemo/iibdemo
[sudo] password for davearno:
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
davearno@ubuntu:~$
```

List the docker images

```
$ sudo docker images
```

```
davearno@ubuntu:~$ sudo docker images
[sudo] password for davearno:
REPOSITORY          TAG             IMAGE ID        CREATED
VIRTUAL SIZE
iibdemo/iibdemo     latest         4fbc74d64d7d   25 hours ago
1.044 GB
<none>              <none>        03949285bd78   4 weeks ago
```

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.

```
$sudo docker run -d -e IIBNODE=NSW -v /etc/localtime:/etc/localtime:ro -v /dev/log:/dev/log -p 4414:4414 -p 11883:11883 -p 7800:7800 -p 7080:7080 -p 49001:49001 --name=grumpy_koala iibdemo/iibdemo:latest
```

```
davearno@ubuntu:~$ sudo docker run -d -e IIBNODE=NSW -v /etc/localtime:/etc/localtime:ro -v /dev/log:/dev/log -p 4414:4414 -p 11883:11883 -p 7800:7800 -p 7080:7080 -p 49001:49001 --name=grumpy_koala iibdemo/iibdemo:latest
08bbb7720613481ce53380f866f9cf50ac1e5992ea22248764fa14ce2bb95e05
```

Check the status of the IIB container

```
$ sudo docker logs grumpy_koala
```

```
davearno@ubuntu:~$ sudo docker logs grumpy_koala
cinitd[1]: cinitd Copyright 2015 Martin Essam - all rights reserved
cinitd[1]: env: PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
cinitd[1]: env: HOSTNAME=08bbb7720613
cinitd[1]: env: IIBNODE=NSW
cinitd[1]: env: IIBADMIN=iibadm
cinitd[1]: env: IIBVERSION=10.0.710.0
cinitd[1]: env: IIBUILDDIR=/tmp/build
cinitd[1]: env: IIBUTILITYSOFTWARE=curl tar rsyslog
cinitd[1]: env: IIBIMAGE_NAME=iib-10.0.710.0.tar.gz
cinitd[1]: env: IIBCONFIGURELOC=/home/iibadm
```

```
Tue Mar 17 01:28:24 PDT 2015 I Creating Integration servers deploying the relevant bar files
Tue Mar 17 01:28:24 PDT 2015 I Recieved the following configuration,
Tue Mar 17 01:28:24 PDT 2015 I eCmd run echo [ OATLEY:echoService.bar PANANIA: NARRABEAN: ]
[ OATLEY:echoService.bar PANANIA: NARRABEAN: ]
Tue Mar 17 01:28:24 PDT 2015 I eCmd run su - iibadm -c "mqsicreateexecutiongroup NSW -e OATLEY"

MQSI 10.0.0.0 BETA
/opt/ibm/iib-10.0.710.0/server

BIP1124I: Creating integration server 'OATLEY' on integration node 'NSW'...
BIP1117I: The integration server was created successfully.

The integration node has initialized the integration server.
BIP8071I: Successful command completion.

Tue Mar 17 01:28:59 PDT 2015 I eCmd run su - iibadm -c "mqsideploy NSW -e OATLEY -a /home/iibadm/configuration/NSW/barfiles/echoService.bar"
```

List running containers

```
$ sudo docker ps
```

```
davearno@ubuntu:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS            PORTS              NAMES
08bbb7720613       iibdemo/iibdemo:latest  "/sbin/cinitd --trac 6 minutes ago
Up 6 minutes       0.0.0.0:4414->4414/tcp, 0.0.0.0:7080->7080/tcp, 0.0.0.0:7800->7800/tcp, 0.0.0.0:11883->11883/tcp, 0.0.0.0:49001->49001/tcp  grumpy_koala
```

Verify the running container and it's processes

```
$sudo docker top container_name
```

```
davearno@ubuntu:~$ sudo docker top grumpy_koala
UID                PID                PPID                C
STIME             TTY                TIME               CMD
root              6185              1557               0
01:28             ?                  00:00:00           /sbin/cinitd --trace
davearno          6255              6185               0
01:28             ?                  00:00:00           bipservice NSW
davearno          6260              6255               3
01:28             ?                  00:00:24           bipbroker NSW
davearno          6302              6260               0
01:28             ?                  00:00:00           bipMQTT -c /var/mqsi
/components/NSW/config/NSW -p 11883
davearno          6371              6260               1
01:28             ?                  00:00:11           DataFlowEngine NSW 3
b43f6ae-0173-4c82-a855-ecc389ce8ed5 OATLEY
davearno          6505              6260               1
01:29             ?                  00:00:08           DataFlowEngine NSW c
d0ff0b3-e74e-4e24-97bf-98a75dbefc66 PANANIA
davearno          6598              6260               1
01:29             ?                  00:00:07           DataFlowEngine NSW 1
05a9568-f207-4040-a8cb-87483d37dd6e NARRABEAN
```

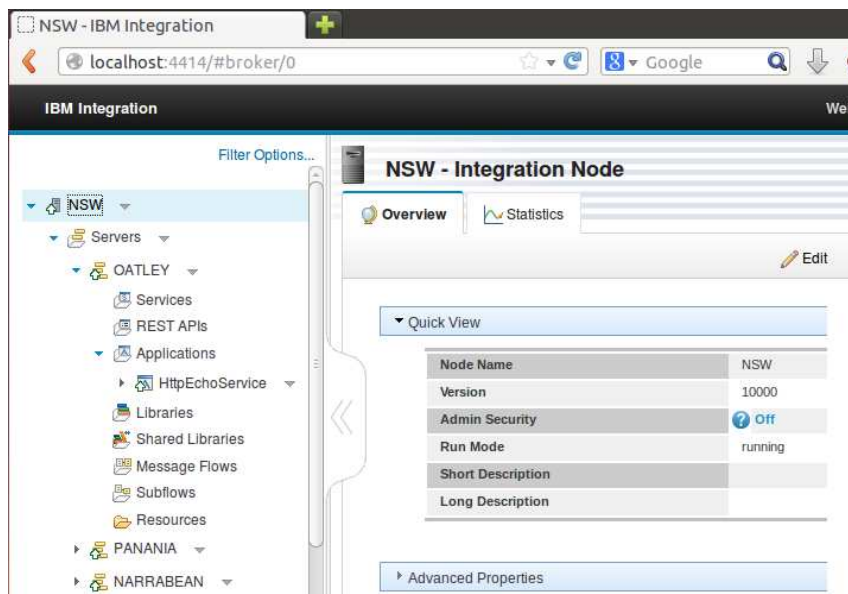
List the mapped ports for the container.

```
$sudo docker port container_name
```

```
davearno@ubuntu:~$ sudo docker port grumpy_koala
11883/tcp -> 0.0.0.0:11883
4414/tcp -> 0.0.0.0:4414
49001/tcp -> 0.0.0.0:49001
7080/tcp -> 0.0.0.0:7080
7800/tcp -> 0.0.0.0:7800
```

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

Start your browser and enter the url <http://127.0.0.1:4414>



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://127.0.0.1:7800/echo
```

```
davearno@ubuntu:~$ echo '{"text": "Hello **world**!"}' | curl -d @- http://127.0.0.1:7800/echo
{"text": "Hello **world**!"}davearno@ubuntu:~$
```

Stopping the running IIB node and the container

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

```
$docker exec -id grumpy_koala "/bin/bash -lc 'iib stop NSW' "
```

```
$docker stop grumpy_koala
```

```
davearno@ubuntu:~$ docker exec -id grumpy_koala "/bin/bash -lc 'iib stop NSW' "
davearno@ubuntu:~$ docker stop grumpy_koala
grumpy_koala
```

Re-starting IIB container and the IIB Node

```
$docker start grumpy_koala
```

```
davearno@ubuntu:~$ docker start grumpy_koala
grumpy_koala
```

```
$docker exec -id grumpy_koala /bin/bash -lc 'iib start NSW'
```

```
davearno@ubuntu:~$ docker exec -id grumpy_koala /bin/bash -lc 'iib start NSW'
```