

IBM MQ  
And  
IBM Integration Bus

Explored on

IBM Cloud Private

## Contents

Introduction .....	6
Connecting to the ICP Console.....	7
Connecting to ICP from the command line.....	8
ICP Knowledge Center link.....	8
Get the icp cluster details .....	8
Connect the command line environment to the ICP cluster .....	8
Get the Kubernetes Pod details – see what's running.....	9
IBM MQ on ICP via the ICP Console .....	10
Deploying MQ via Helm .....	10
Helm Chart Configuration .....	11
View the Helm Release .....	11
Review the MQ Helm Release details in the ICP console .....	12
Connect the IBM MQ Web Console .....	13
Running command line commands in the MQ Container .....	14
Using RUNMQSC to configure the queue manager .....	15
Connecting the MQ Explorer .....	15
Add a new user name in the mqm group .....	15
Use RUNMQSC to set up channel authentication.....	15
Connect from MQ Explorer.....	16
Some points of note.....	18
IBM Integration Bus on ICP via the ICP Console .....	19
Installing IIB by Helm Chart.....	19
Configure the Helm Chart template .....	20
Check the IIB Helm Release .....	21
Capture the IIB IP address and Ports .....	21
Connect the Web UI.....	23
Connect the IIB Toolkit to the runtime node .....	23
Deploying a BAR file .....	25
Testing an IIB message flow.....	26
Test using a tool like Postman or Resteasy.....	26

IIB & MQ Container on ICP from Github repository by hand .....	28
MQ and IIB Docker repository on github.....	28
Git Clone the IIB-MQ repository to a local repository .....	28
Build the IIB-MQ image.....	28
Copy the IIB-MQ image to the ICP repository.....	29
Ensure the terminal can ping the correct cluster via hostname.....	29
Docker login .....	30
Docker push the image to ICP.....	30
Check the IIB-MQ image via the ICP Console.....	31
Adding a new (custom) IIB and MQ Helm chart to ICP .....	32
MQ and IIB Helm Chart repository on github .....	32
Edit the values.yaml file .....	32
Git Clone the Helm Chart repository to local machine (not ICP in this example).....	33
Configure the Kubernetes client to connect to ICP .....	34
Install the Helm Chart from the local machine (not ICP UI in this example) .....	34
Check the deploy in the ICP console.....	35
IIB & MQ Container on ICP from Github repository via MSB pipeline.....	38
Creating the IIB-MQ Repository in GitHub Organization.....	38
Source GitHub Repositories to leverage .....	38
1. IIB an MQ Docker build repository .....	38
2. IIB and MQ Helm Chart repository.....	38
Target GitHub Repository in Github Organization.....	38
Add a Jenkins file to the repository .....	40
Add the IIB and MQ Helm chart files to the repository .....	41
Download from the iib-mq-chart repository to the local file system .....	42
Import to from the iib-mq-chart repository to the local file system .....	43
Important Parameters used in this example .....	48
Configuring Github repository .....	49
Github repository IIB-MQ in organization DAVEXACOM .....	49
Github Personal Access Token – Create a token .....	49
Developer Settings – Personal Access tokens.....	49

OAuth Apps – Register a new application .....	52
Developer Settings – Set up OAuth.....	52
OAuth Apps – Register a new application.....	52
Capture OAuth Client ID and Secret .....	54
IBM Cloud Private Console – Check Micro Service Builder Fabric Service .....	55
IBM Cloud Private Console – Check Jenkins Persistent Volume.....	55
IBM Cloud Private Console – Deploy MicroServiceBuild Pipeline .....	55
Configure the ibm-microservicebuilder pipeline helm chart.....	56
Review the microservice builder pipeline helm release .....	61
Authorize the ibm-msb-pipeline-da service to access github.....	62
IBM Cloud Private – Jenkins.....	63
Checking Jenkins connection to Github .....	63
Before Starting the Jenkins build process.....	65
Starting the Jenkins build process.....	66
Following the Jenkins build logs.....	67
Checking ICP Image repository .....	70
Checking ICP Helm Release .....	71
Connect MQ Console .....	73
Connect to IIB Web UI.....	75
Adding your own MQSC for MQ files and rebuilding with Jenkins .....	77
How it hangs together .....	77
Updating IIB-MQ GitHub Repo in DAVEXACOM Org.....	78
Re-run the ICP Microservices builder Jenkins pipeline.....	78
Check the results in the IBM MQ Console .....	79
Adding your own BAR files for IIB and rebuilding with Jenkins .....	80
How it hangs together .....	80
Create a BAR file in an IIB Toolkit.....	80
Upload the BAR file to the IIB-MQ repo on GitHub .....	82
Edit the dockerfile on GitHub .....	84
Edit the iib_manage.sh file on GitHub .....	84
Re-run the ICP Microservices builder Jenkins pipeline.....	85

Check the results in the IIB Web UI .....	86
Test the deployed message flow .....	87
Automating the Build end to end .....	91
Solution Overview Diagram .....	91
Creating a local clone of the GitHub Repos IIB-MQ.....	91
Example using GitGUI.....	91
Creating the GitHub WebHook to drive Jenkins .....	94
Optional – Use IBM Secure Gateway to connect Github to ICP on Pure .....	94
Setting up the Webhook in Github .....	95
Testing the automated build process .....	101
Review Jenkins Status .....	101
Update the IIB message flow .....	101
Export the BAR file to the local clone of IIB-MQ.....	102
Push to GitHub IIB-MQ repository .....	103
Check the GitHub repository Status.....	105
Review Jenkins Status Again .....	105
Check the Jenkins build logs .....	106
Check the results in the IIB Web UI .....	108
Test the deployed message flow .....	109

## Introduction

This document will explore the basics of deploying and connecting to IBM MQ and IBM Integration Bus on IBM Cloud Private (ICP).

The ICP environment I used is located on the IBM Sydney (IIC) Pure Application System (Bluemix Local) rack in St Leonards.

This instance of ICP on the PureApp is not public and VPN access to the IIC is required. However, the instructions (VPN connection aside) have been verified against ICP running on non-VPN accessible instances such as laptops.

During the construction of this document the ICP instance in the Sydney IIC Pure Application system was upgraded and rebuilt. Therefore, IP address and port numbers in screen shots will have changed. For example

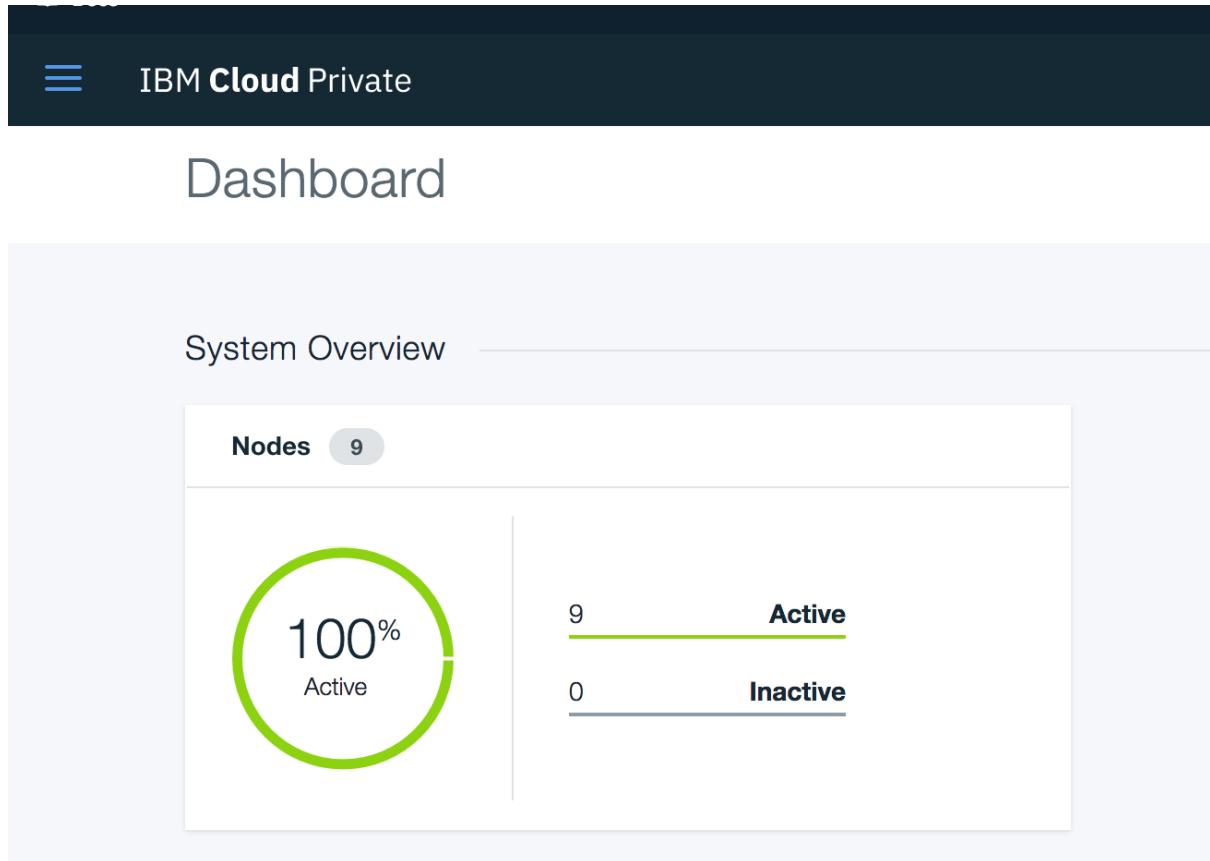
Details that changed	Before Upgrade/Rebuild	After Upgrade/Rebuild
ICP Console IP	<a href="https://172.23.49.17:8443/console/">https://172.23.49.17:8443/console/</a>	https://172.23.50.112:8443/console/
ICP deployment ingress IP	172.23.49.19: <i>portnumber</i>	172.23.50.111: <i>portnumber</i>

Where possible in the text I have made these IP address changes, however I have not recaptured all the screen shots.

## Connecting to the ICP Console

Point your favourite Browser at : <https://172.23.50.112:8443/console/>

At the login screen admin/admin to log into ICP console



# Connecting to ICP from the command line.

## *ICP Knowledge Center link*

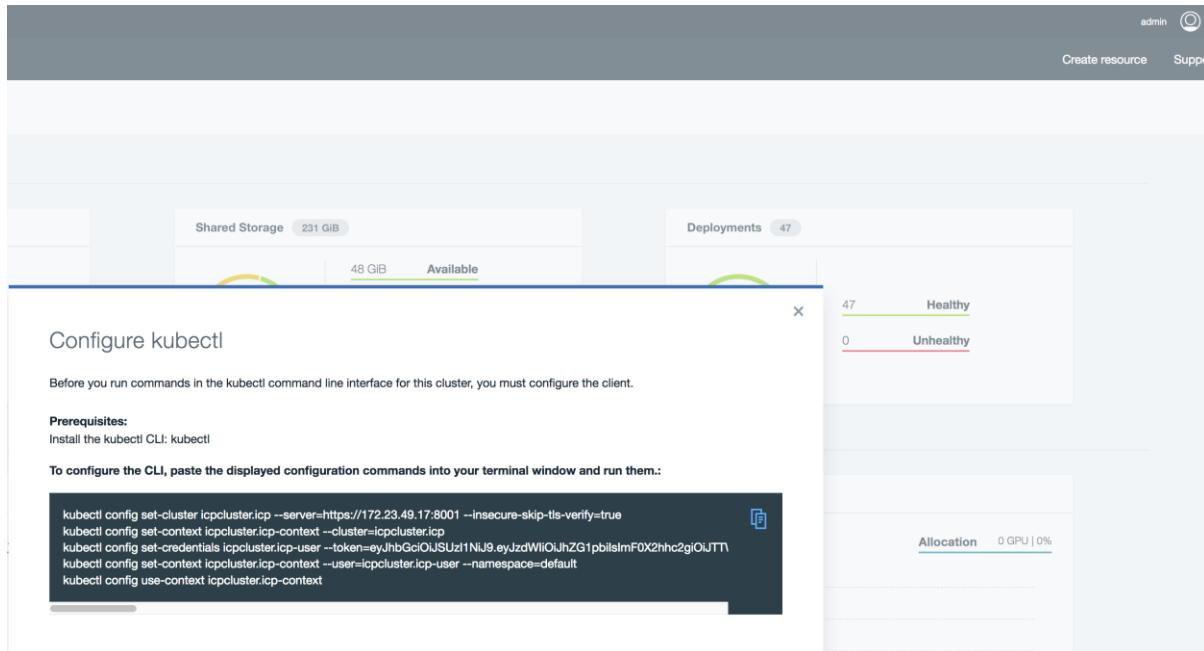
**Knowledge Center can be found here if you need it**

[https://www.ibm.com/support/knowledgecenter/SSBS6K\\_2.1.0/manage\\_cluster/cfc\\_cli.html](https://www.ibm.com/support/knowledgecenter/SSBS6K_2.1.0/manage_cluster/cfc_cli.html)

This link takes you to the kubernetes CLI install and set up

## *Get the icp cluster details*

Select User Name > Configure client, which is in the upper right of the window. The cluster configuration details display, resembles the following code:



Use the “Copied” icon and run in a terminal on your machine where kubernetes cli is installed and set up

## *Connect the command line environment to the ICP cluster*

```
kubectl config set-cluster icpcluster.icp --server=https://172.23.50.112:8001 --insecure-skip-tls-verify=true
```

```
kubectl config set-context icpcluster.icp-context --cluster=icpcluster.icp
```

```
kubectl config set-credentials icpcluster.icp-user --
token=eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJhZG1pbilsImF0X2hhc2giOiJTUU1raV80Yk9xTlRTZmVXWnBoN1F3liwiaXNzljoiaHR0cHM6Ly9pY3BjbHVzdGVyLmljcDo5NDQzMjL29pZGMvZW5kcG9pbnQvT1AiLCJhdWQiOjMzFkZDgxMjVmNjg3ZjlxMzE1Y2ZjYTEyYTY5ZjI4MiIsImV4cCI6MTUxMTk3MTI2OSwiaWF0IjoxNTEwOTI4MDY5fQ.XAOX5mFAeADXuvmENY01N0Rk46MStjBdwPkKfpFSEoPJRjpTCsFZ1uh6rjuNffZiZLMZO-
VZti3V4725_qyzkZGCNMIsqX9V6t3lJtkTDahKDQDWiisqQ2JvY1O07qJQXxgheGih2UnR3NLQj_A5H01BLUSgts5kN4Nh_0DuD6QojsVMxh-kOS9Y352peRcpUyg5IBJdPbLHicAeNodbvVglYw-
```

```
amqnqnmldy_v09lsTV_rl7vBfLr0RrikuzBGLOVcXXpDAGc7UIfnb2txcNco4T1qsfWbJzw2Rwr4Ta2PcdQ  
gFg3hckBx6-degxmeSMYj3cjsrw1UU39bhzmpHA
```

```
kubectl config set-context icpcluster.icp-context --user=icpcluster.icp-user --namespace=default  
kubectl config use-context icpcluster.icp-context
```

### *Get the Kubernetes Pod details – see what's running*

```
Daves-MacBook-Pro-2:deploy-ibm-cloud-private-master davearno$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
blockchain-ca-2437173632-sfhlw	1/1	Running	0	18h
blockchain-orderer-1019217726-jc9vs	1/1	Running	0	18h
blockchain-org1peer1-3689184489-9jcj8	1/1	Running	0	18h
blockchain-org2peer1-2132069490-bshv0	1/1	Running	0	18h
composer-playground-943199952-4tb02	1/1	Running	4	18h
composer-rest-server-1507354567-m993w	1/1	Running	0	18h
db2server1-ibm-db2oltp-dev-2982159975-hxq2s	1/1	Running	0	1d
ibm-msb-pipeline-ibm-microservicebuilder-pipeline-146566793qkl5	1/1	Running	0	15h
ibm-msbf-zipkin-3680589052-cgfjw	1/1	Running	0	16h
icpmsbdemo-deployment-3190653047-r9c8r	1/1	Running	0	14h
javacloudtrader4db2-3324903285-9v2lm	1/1	Running	1	1d

# IBM MQ on ICP via the ICP Console

*Deploying MQ via Helm*

Go to the Catalog (filter on MQ)

The screenshot shows the IBM Cloud Private Catalog interface. At the top, there is a dark header bar with the text "IBM Cloud Private" and a three-line menu icon. Below this is a light-colored search bar containing a magnifying glass icon and the text "MQ". Underneath the search bar, the word "Catalog" is displayed in a large, bold, black font. The main content area is titled "Helm charts" and contains the sub-instruction "Deploy your applications and install software packages". Two chart entries are visible: "ibm-mqadvanced-server-dev" and "rabbitmq". Each entry includes a small circular icon with a gear and a bar chart, the chart name, a brief description, and a blue "ibm-charts" button. The "ibm-mqadvanced-server-dev" entry also has a "Configure" button below it.

Select **ibm-mqadvanced-server-dev** and the following page details how to install the helm chart from the command line (listing all the parameters further down the page)

OR

Use the Configure button

ibm-mqadvanced-server-dev V 1.0.2

The screenshot shows the Helm chart details for 'IBM MQ queue manager' version 1.0.2. It includes the chart icon (IBM MQ logo), basic metadata (published Nov 23rd 2017, type Chart), and a brief description of IBM MQ as messaging middleware. Below this is the 'Introduction' section, which states that the chart deploys a single IBM MQ Advanced for Developers server (queue manager) into an IBM Cloud private or other Kubernetes environment. The 'Prerequisites' section lists 'Kubernetes 1.6 or greater, with beta APIs enabled'. The 'Installing the Chart' section contains a command-line snippet for Helm installation:

```
helm install --name foo stable/ibm-mqadvanced-server-dev --set license=accept
```

This command accepts the [IBM MQ Advanced for Developers license](#) and deploys an MQ Advanced for Developers server on the Kubernetes cluster. The [configuration](#) section lists the parameters that can be configured during installation. A tip at the bottom suggests using `kubectl get all -l release=foo` to see all resources deployed by the chart.

[Uninstalling the Chart](#)

[Configure](#)

## Helm Chart Configuration

### From the Configure button

Tick the accept license

Set the release name : mq-da-icpqm1 (don't use any capitals or a number for 1<sup>st</sup> character)

Set Persistence to false

Service ->Type: NodePort (not clusterIP)

Set a queue manager name : icpQM1

Admin.password : passw0rd (you must set a password)

Hit install

### View the Helm Release

The dialog box displays a green checkmark icon and the text 'Installation complete'. It features a blue 'View Helm Release' button and a link to 'Return to Catalog'.

## Click on View Helm Release

Helm releases

Helm releases					
NAME		NAMESPACE	STATUS	UPDATED	CHART NAME
blockchain-chaincode	default	DEPLOYED	Nov 28th 2017	ibm-blockchain-chaincode	1.0.0
blockchain-channel	default	DEPLOYED	Nov 28th 2017	ibm-blockchain-channel	1.0.0
blockchain-composer	default	DEPLOYED	Nov 28th 2017	ibm-blockchain-composer	1.0.0
blockchain-network	default	DEPLOYED	Nov 28th 2017	ibm-blockchain-network	1.0.1
cam	default	DEPLOYED	Nov 27th 2017	ibm-cam-prod	1.0.1
db2server1	default	DEPLOYED	Nov 27th 2017	ibm-db2oltp-dev	1.0.0
ibm-msb-pipeline	default	DEPLOYED	Nov 29th 2017	ibm-microservicebuilder-pipeline	2.0.2
ibm-msbf	default	DEPLOYED	Nov 28th 2017	ibm-microservicebuilder-fabric	2.0.1
icpmsbdemo	default	DEPLOYED	Nov 29th 2017	icpmsbdemo	1.0.0
mq-da-icpqm1	default	DEPLOYED	Nov 29th 2017	ibm-mqadvanced-server-dev	1.0.2

mq-da-icpqm1

DETAILS	
TYPE	DETAIL
Name:	mq-da-icpqm1
Namespace:	default
Status:	DEPLOYED
Chart name:	ibm-mqadvanced-server-dev
Chart version:	1.0.2
updated:	Nov 29th 2017 at 3:48 PM
Type:	Release

SECRET			
NAME	TYPE	DATA	AGE
mq-da-icpqm1-ibm-mqadvan	Opaque	1	2m

SERVICE				
NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
mq-da-icpqm1-ibm-mqadvan	10.0.0.169	<none>	1414/TCP9443/TCP	2m

Now wait for the service to spin up

*Review the MQ Helm Release details in the ICP console*

IBM Cloud Private

Services / mq-da-icpqm1-ibm-mqadvan / mq-da-icpqm1-ibm-mqadvan

Overview

Service details

Type	Detail
Name	mq-da-icpqm1-ibm-mqadvan
Namespace	default
Creation time	Nov 29th 2017 at 3:48 PM
Type	NodePort
Labels	app= mq-da-icpqm1-ibm-mqadvan, chart=ibm-mqadvanced-server-dev-1.0.2, heritage=Tiller, release= mq-da-icpqm1
Selector	app= mq-da-icpqm1-ibm-mqadvan
IP	10.0.0.169
Port	qmgr-server 1414/TCP; qmgr-web 9443/TCP
Node port	<a href="#">qmgr-server 30464/TCP</a> <a href="#">qmgr-web 32489/TCP</a>
Session affinity	None

## Connect the IBM MQ Web Console

Hover over the Node port links and the IP address will appear bottom left.

Click on qmgr-server 30464/TCP link and a new browser tab will open (you have to add https://)

Default username is **admin** and the password as the one set at configuration time (passw0rd)

IBM MQ Console - Login

Please enter your username and password

User Name:

Password:

Please note that after some time you will be signed out automatically and asked to sign in again

Licensed Materials - Property of IBM Corp. (c) Copyright IBM Corporation and its licensors 2014, 2017. The IBM logo is a registered trademark of International Business Machines Corporation in the United States, other countries, or both. Other company, product or services names may be trademarks or services of others.

The screenshot shows the IBM MQ Console interface with four main panes:

- Queue Manager**: Shows one queue manager named "icpQM1" in a "Running" status.
- Channels on icpQM1**: Shows two channels: "DEVADMIN.SVRCONN" and "DEVAPPSVRCONN", both of which are "Inactive".
- Queues on icpQM1**: Shows four queues: "DEV.DEAD.LETTER.QUEUE", "DEVQUEUE.1", "DEVQUEUE.2", and "DEVQUEUE.3", all of which are "Local" type with a depth of 0.
- Topics on icpQM1**: Shows one topic named "DEV.BASE.TOPIC" with a topic string of "dev/".

Add additional Qmgr information panes (add widget) – listeners is a good one

The dialog box is titled "Add a new widget" and contains the following information:

- Local Queue Managers**: Manage local queue managers. A "Chart" link is available to monitor the MQ platform.
- Add a widget to display MQ object information for the specified queue manager**: A dropdown menu labeled "Queue manager:" is set to "icpQM1".
- Object Types** (Listed below the dropdown):
  - Queues**: Configure destinations for messages.
  - Topics**: Administrative objects for assigning attributes to topics.
  - Listeners**: Configure processes to accept network requests.
  - Channels**: Queue manager communication paths.
  - Client-connection Channels**: Client connectivity details.
  - Authentication Information**: Configure authentication mechanisms.
  - Subscriptions**: Configure how subscriptions to topics are handled.
  - Channel Authentication Records**: Control access to channels.

A "Close" button is at the bottom right of the dialog.

## Running command line commands in the MQ Container

### Bash into a container

Issue a `kubectl get pods` command

```
Daves-MacBook-Pro-2:deploy-ibm-cloud-private-master davearno$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
blockchain-ca-2437173632-sfhlw     1/1    Running   0          19h
blockchain-orderer-1019217726-jc9vs 1/1    Running   0          19h
blockchain-org1peer1-3689184489-9cj8 1/1    Running   0          19h
blockchain-org2peer1-2132069490-bshv0 1/1    Running   0          19h
composer-playground-943199952-4tb02 1/1    Running   4          19h
composer-rest-server-1507354567-m993w 1/1    Running   0          19h
db2server1-ibm-db2oltp-dev-2982159975-hxq2s 1/1    Running   0          2d
```

```

ibm-msb-pipeline-ibm-microservicebuilder-pipeline-146566793qkl5 1/1     Running  0
16h
ibm-msbf-zipkin-3680589052-cgfjw                         1/1     Running  0      17h
icpmsbdemo-deployment-3190653047-r9c8r                   1/1     Running  0      15h
javacloudtrader4db2-3324903285-9v2lm                  1/1     Running  1      1d
mq-da-icpqm1-ibm-mqadvan-0                            1/1     Running  0      9m

```

*/Issue kubectl exec -it mq-da-icpqm1-ibm-mqadvan-0 bash*

### Using RUNMQSC to configure the queue manager

```

Daves-MacBook-Pro-2:deploy-ibm-cloud-private-master davearno$ kubectl exec -it mq-da-
icpqm1-ibm-mqadvan-0 bash
(mq:9.0.4.0)root@mq-da-icpqm1-ibm-mqadvan-0:/# runmqsc icpQM1
5724-H72 (C) Copyright IBM Corp. 1994, 2017.
Starting MQSC for queue manager icpQM1.

```

```

dis ql(*)
  1 : dis ql(*)
AMQ8409I: Display Queue details.
  QUEUE(DEV.DEAD.LETTER.QUEUE)          TYPE(QLOCAL)

```

### *Connecting the MQ Explorer*

Having bashed into your container - create a user with a password that is in the MQ group I used davearno

Got the instructions from here  
<https://github.com/ibm-messaging/mq-docker>

### Add a new user name in the mqm group

```

useradd davearno -G mqm && \
echo davearno:passw0rd | chpasswd

```

```
(mq:9.0.3.0)root@mq-rel-1-ibm-mqadvanced-0:/# useradd davearno -G mqm && \
```

### Use RUNMQSC to set up channel authentication

```

(mq:9.0.3.0)root@mq-rel-1-ibm-mqadvanced-0:/# runmqsc icpQM1
5724-H72 (C) Copyright IBM Corp. 1994, 2017.
Starting MQSC for queue manager icpQM1.

```

```

DEFINE CHANNEL(PASSWORD.SVRCONN) CHLTYPE(SVRCONN) REPLACE
  1 : DEFINE CHANNEL(PASSWORD.SVRCONN) CHLTYPE(SVRCONN) REPLACE
AMQ8014: IBM MQ channel created.
SET CHLAUTH(PASSWORD.SVRCONN) TYPE(BLOCKUSER) USERLIST('nobody')
DESCR('Allow privileged users on this channel')

```

2 : SET CHLAUTH(PASSWORD.SVRCONN) TYPE(BLOCKUSER) USERLIST('nobody')  
 DESCRIPTOR('Allow privileged users on this channel')  
 AMQ8877: IBM MQ channel authentication record set.  
**SET CHLAUTH(\*) TYPE(ADDRESSMAP) ADDRESS(\*) USERSRC(NOACCESS)  
 DESCRIPTOR('BackStop rule')**

3 : SET CHLAUTH(\*) TYPE(ADDRESSMAP) ADDRESS(\*) USERSRC(NOACCESS)  
 DESCRIPTOR('BackStop rule')  
 AMQ8883: Channel authentication record already exists.  
**SET CHLAUTH(PASSWORD.SVRCONN) TYPE(ADDRESSMAP) ADDRESS(\*)  
 USERSRC(CHANNEL) CHCKCLNT(REQUIRED)**

4 : SET CHLAUTH(PASSWORD.SVRCONN) TYPE(ADDRESSMAP) ADDRESS(\*)  
 USERSRC(CHANNEL) CHCKCLNT(REQUIRED)  
 AMQ8877: IBM MQ channel authentication record set.  
**ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS)  
 ADOPTCTX(YES)**

5 : ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)  
 AUTHTYPE(IDPWOS) ADOPTCTX(YES)  
 AMQ8567: IBM MQ authentication information changed.  
**REFRESH SECURITY TYPE(CONNAUTH)**

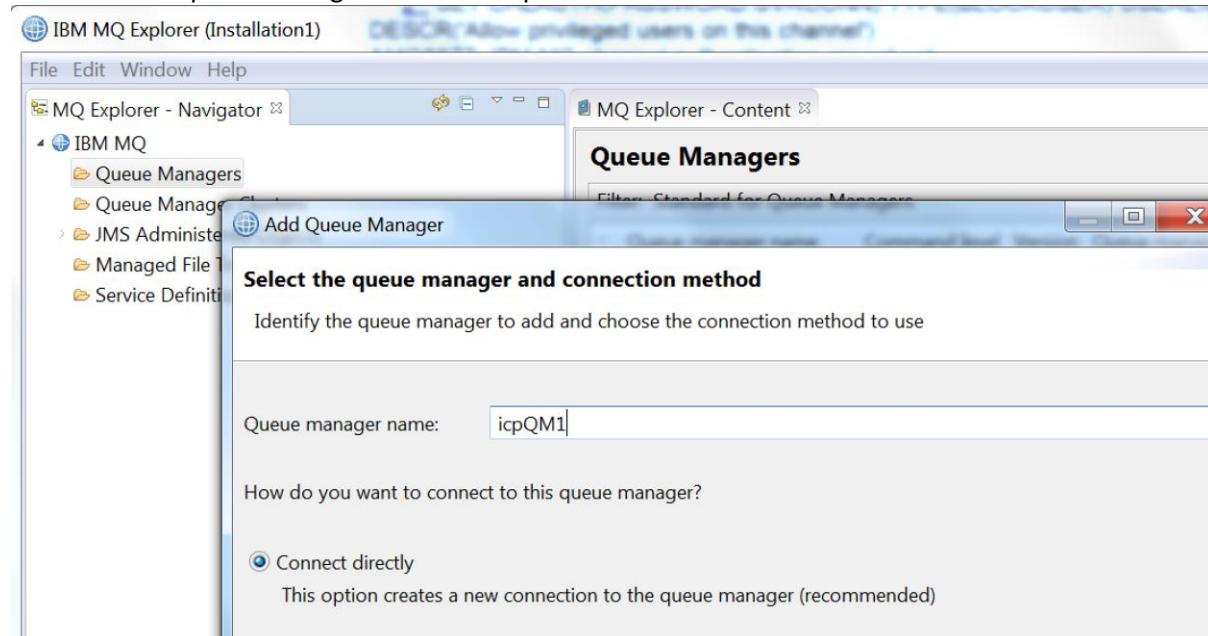
6 : REFRESH SECURITY TYPE(CONNAUTH)

end

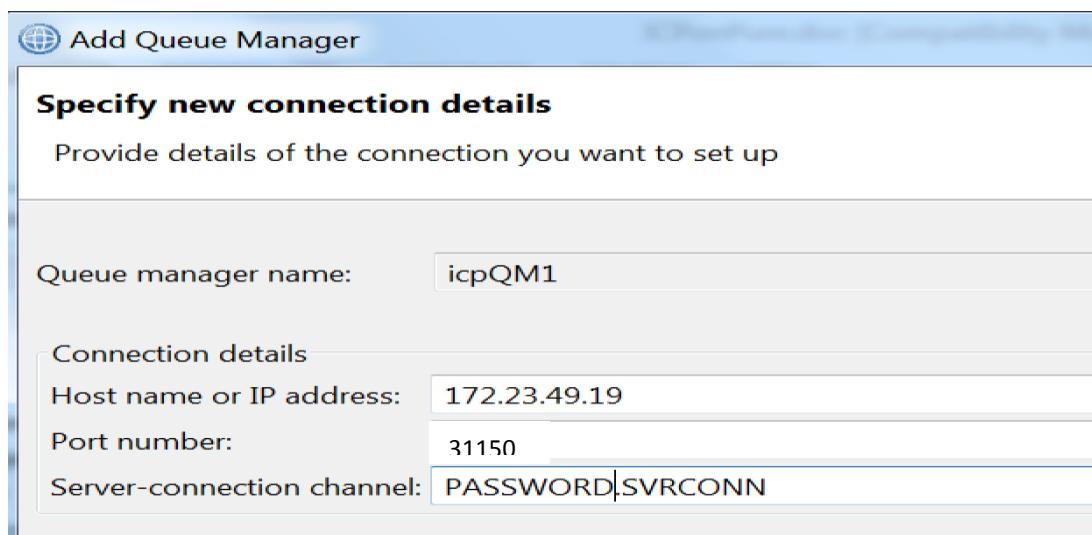
## Connect from MQ Explorer

From your MQ explorer - add remote queue manager connect using your  
 PASSWORD.SVRCONN and enable user identification with a password

### Add a remote queue manager in the MQExplorer

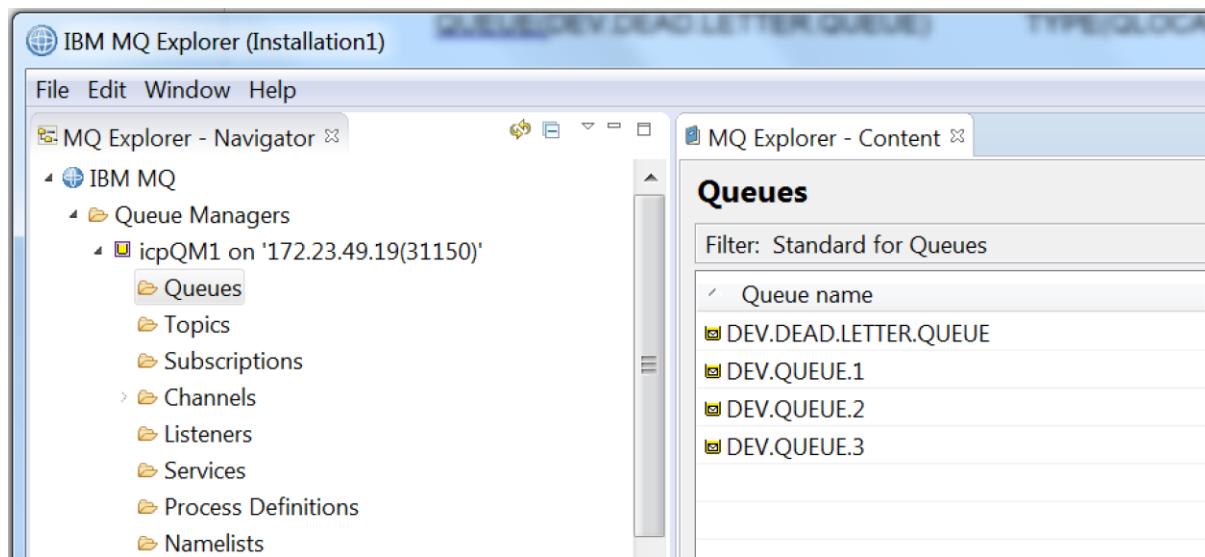
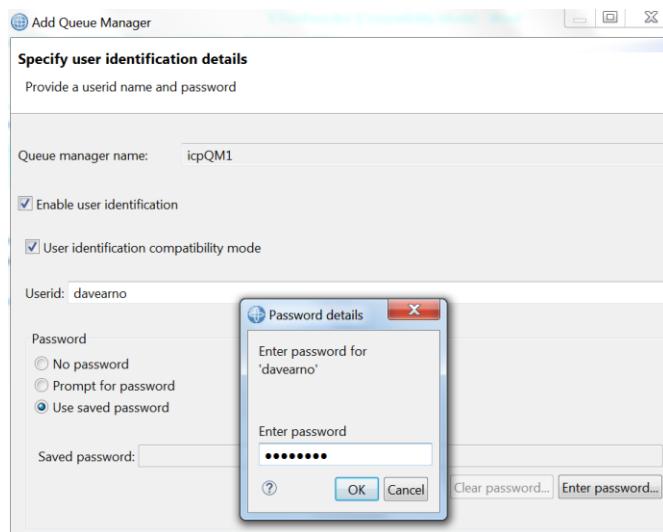


Next specify the host name (port is 31150 not 1414) and the PASSWORD.SVRCONN



Next leverage the username/password you set up using the following commands:

```
useradd davearno -G mqm && \
echo davearno:passw0rd | chpasswd
```



## Some points of note

The container is running in the following loop. If you endmqm your icpQM1 the container will stop and be restarted in it's initial state i.e. your userid to mqm group and icpQM1 new SVRCONN channel will be gone.

```
# Loop until "dspmq" says the queue manager is not running any more
until [ "state" != "RUNNING" ]; do
    sleep 5
done
```

You can however create a second queue manager once "bashed in" and start and stop that as much as you like.

# IBM Integration Bus on ICP via the ICP Console

## *Installing IIB by Helm Chart*

Go to the catalog and filter on integration

The screenshot shows the IBM Cloud Private Catalog interface. At the top, there is a dark header bar with the text "IBM Cloud Private". Below it, a light blue search bar contains the text "integration". The main content area is titled "Catalog" and displays a list of Helm charts under the heading "Helm charts". Deploy your applications and install software packages.

Chart Name	Description	Type
<b>ibm-integration-bus-dev</b>	IBM Integration Bus for application integration, routing and transformation	ibm-charts
<b>spring-cloud-data-flow</b>	Open source, multi-cloud toolkit for building data integration and real-time data processing pipelines.	incubator
<b>ibm-mongodb</b>	NoSQL document database for JSON-like documents	ibm-charts

Click on ibm-integration-bus-dev

## Configure the Helm Chart template

Fill in the template configuration values

Be careful to use lower case characters, avoid numerics up front and only dashes no dots.

IBM Cloud Private

View all

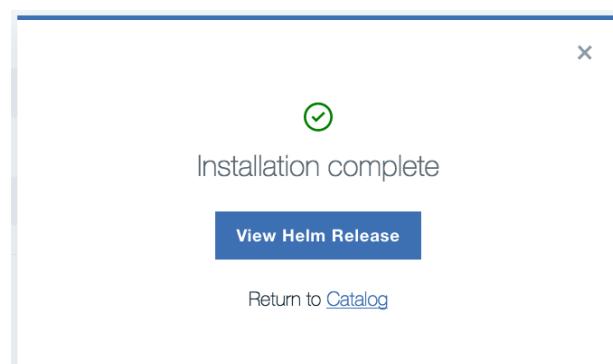
Deploy ibm-integration-bus-dev V 1.0.3

Configuration

IBM Integration Bus for application integration, routing and transformation Edit these parameters for configuration

Release name	ibl-icp-100010	Target Namespace	default
<input checked="" type="checkbox"/> I have read and agreed to the <a href="#">license agreements</a>			
license	accept		
image		tag	10.0.0.10
repository	ibmcom/lib	pullPolicy	IfNotPresent
pullSecret	Enter value		
resources			
limits.cpu	2	limits.memory	2048Mi
requests.cpu	1	requests.memory	512Mi
nodename	IIB_ICPNODE		
servername	IIB_DEFAULT		
service			
type	NodePort	webuiPort	4414
serverlistenerPort	7800	nodeListenerPort	7080

Click Install



## *Check the IIB Helm Release*

Check the Workloads->Helm releases

Helm releases

20 Items per page   1-12 of 12 items				
NAME	NAMESPACE	STATUS	UPDATED	CHART NAME
blockchain-chaincode	default	DEPLOYED	Nov 28th 2017	ibm-blockchain-chaincode
blockchain-channel	default	DEPLOYED	Nov 28th 2017	ibm-blockchain-channel
blockchain-composer	default	DEPLOYED	Nov 28th 2017	ibm-blockchain-composer
blockchain-network	default	DEPLOYED	Nov 28th 2017	ibm-blockchain-network
cam	default	DEPLOYED	Nov 27th 2017	ibm-cam-prod
db2server1	default	DEPLOYED	Nov 27th 2017	ibm-db2oltp-dev
ibm-msb-pipeline	default	DEPLOYED	Nov 29th 2017	ibm-microservicebuilder-pipeline
ibm-msbf	default	DEPLOYED	Nov 28th 2017	ibm-microservicebuilder-fabric
icp-dp7	default	DEPLOYED	Dec 4th 2017	ibm-datapower-dev
icpmbsdemo	default	DEPLOYED	Nov 29th 2017	icpmbsdemo
ib-icp-100010	default	DEPLOYED	Dec 4th 2017	ibm-integration-bus-dev

## *Capture the IIB IP address and Ports*

Select your iib release

iib-icp-100010

DETAILS	
TYPE	DETAIL
Name:	iib-icp-100010
Namespace:	default
Status:	DEPLOYED
Chart name:	ibm-integration-bus-dev
Chart version:	1.0.3
updated:	Dec 4th 2017 at 2:51 PM
Type:	Release

SERVICE			
NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)
ib-icp-100010-ibm-integ	10.0.0.245	<nodes>	4414:32037/TCP,7800:32358/TCP,7080:31554/TCP

DEPLOYMENT			
NAME	DESIRED	CURRENT	UP-TO-DATE
ib-icp-100010-ibm-integ	1	1	1

Click on service

Services / iib-icp-100010-ibm-integ /

## iib-icp-100010-ibm-integ

### Overview

#### Service details

TYPE	DETAIL
Name	iib-icp-100010-ibm-integ
Namespace	default
Creation time	Dec 4th 2017 at 2:51 PM
Type	NodePort
Labels	app=iib-icp-100010-ibm-integ,chart=ibm-integration-bus-dev-1.0.3,heritage=Tiller,release=iib-icp-100010
Selector	app=iib-icp-100010-ibm-integ
IP	10.0.0.245
Port	webui 4414/TCP; serverlistener 7800/TCP; nodelistener 7080/TCP
Node port	<a href="#">webui 32037/TCP</a> <a href="#">serverlistener 32358/TCP</a> <a href="#">nodelistener 31554/TCP</a>
Session affinity	None

### Observe the Node ports

webui 4414/TCP; serverlistener 7800/TCP; nodelistener 7080/TCP

[webui 32037/TCP](#)

[serverlistener 32358/TCP](#)

[nodelistener 31554/TCP](#)

## Connect the Web UI

Click on the webUI link or copy the IP address and port to a browser. It may take a couple of minutes for ICP to deploy and run up IIB.

The screenshot shows the IBM Integration Bus (IIB) Web UI interface. The title bar indicates the current page is 'IIB\_ICPNODE - IBM Integrat...'. The left sidebar navigation includes 'IBM Integration' (selected), 'Filter Options...', 'IIB\_ICPNODE' (selected), 'Servers', 'IIB\_DEFAULT' (selected), 'Services' (REST APIs, Applications, Libraries, Shared Libraries, Message Flows, Subflows, Resources), 'Operational Policy', 'Data', 'Security', 'Monitoring', and 'Business'. The main content area displays the 'IIB\_ICPNODE - Integration Node' details. It has tabs for 'Overview' (selected) and 'Statistics'. Under 'Quick View', the node name is IIB\_ICPNODE, version is 100010, Admin Security is Off, Run Mode is running, and there are fields for Short Description and Long Description. Under 'Advanced Properties', the platform name is Linux, fixpack capability is 10.0.0.1, operation mode is developer, architecture is x86\_64, platform version is 3.10.0-514.16.1.el7.x86\_64, queue manager is Queue Manager, build level is ib1000-L170911.419 (S1000-L170901.10502), and admin agent process ID is 121. Under 'Component Properties', security cache sweep interval is 300, security cache timeout is 60, cache manager policy is disabled, port range is 2800-2819, and cache manager listener host is listed.

## Connect the IIB Toolkit to the runtime node

webui 4414/TCP; serverlistener 7800/TCP; nodelistener 7080/TCP

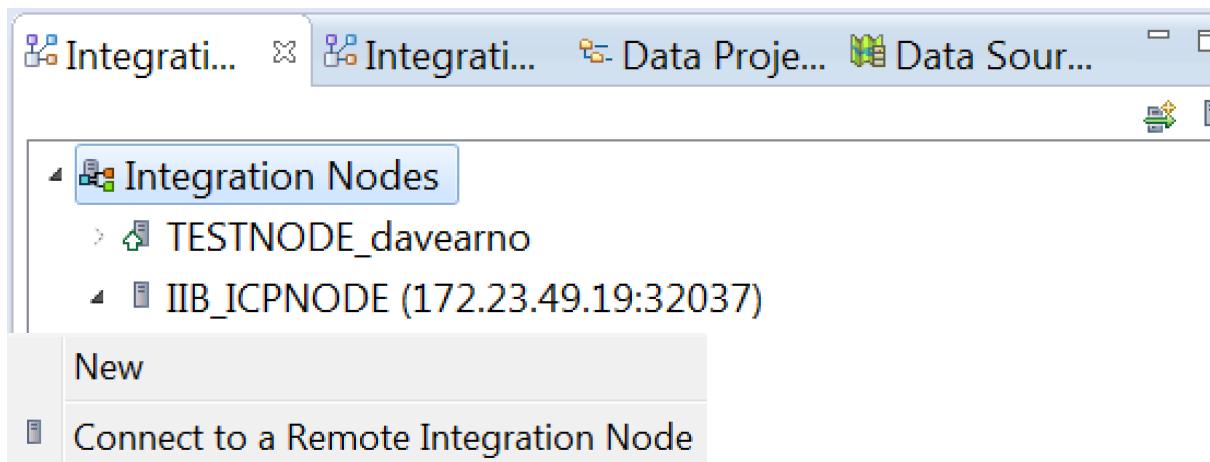
[webui 32037/TCP](#)

[serverlistener 32358/TCP](#)

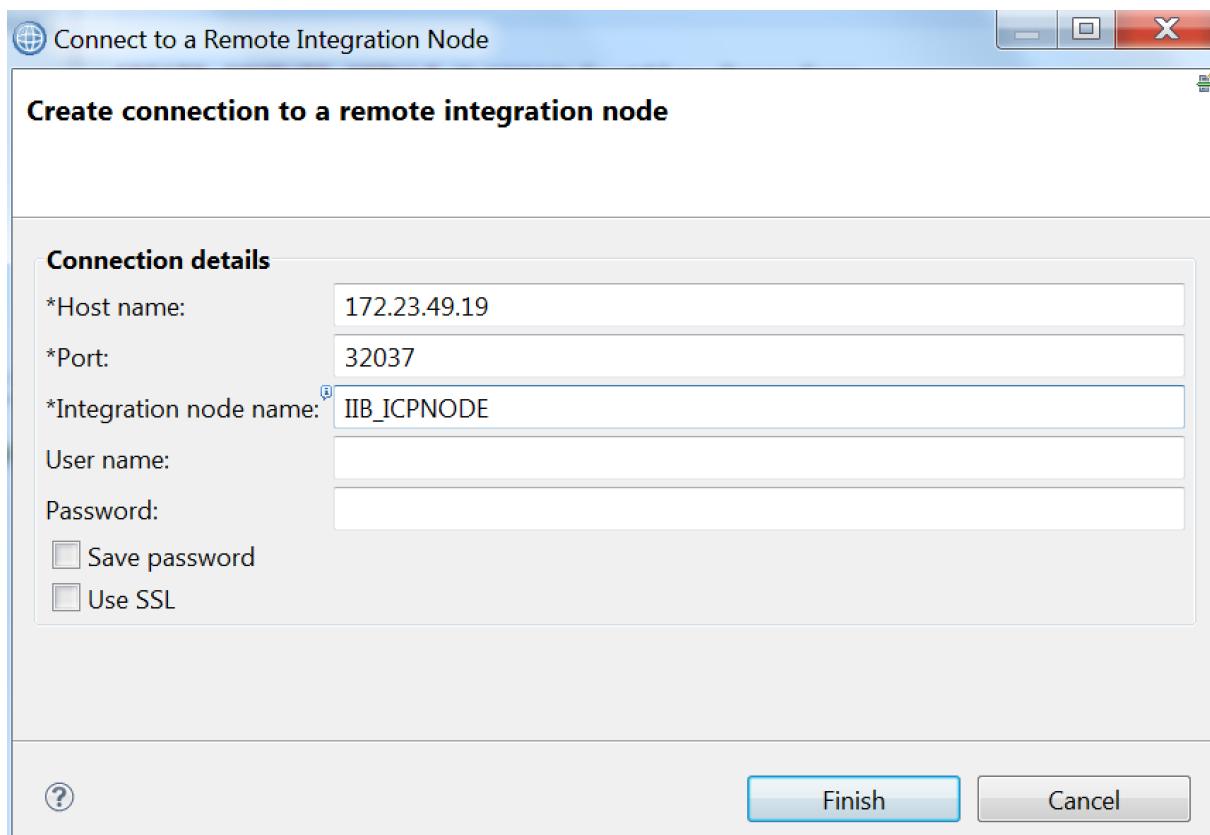
[nodelistener 31554/TCP](#)

The IIB Toolkit will use the webUI ip address and port

Ctrl-select Integration Nodes and select Connect to a Remote Integration Node

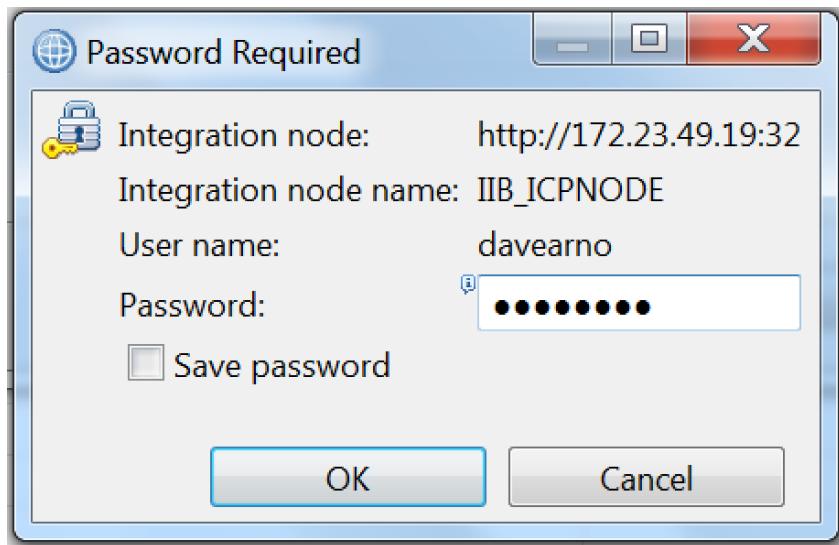


Fill in the details



A userid/password dialog may well be opened.

I used password: passw0rd. Not sure where davearno is being picked up from. It is a username I created in the MQ ICP container but that shouldn't be relevant! Davearno is the username of my Windows image that the toolkit it running in – but passw0rd is not the password.



### Deploying a BAR file

Assuming you have a BAR file ready go. Deploy a BAR file to IIB\_DEFAULT integration server

The screenshot shows the "Application Development" perspective in a software interface. The "MyICPTestApp" project is expanded, showing:

- Flows**:
  - MyHTTPESQLTestFlow.msgflow
- ESQLs**:
  - MyHTTPESQLTestFlow\_InToOut.esql
  - MyMQESQLTestFlow\_Transform.esql
- Independent Resources**

Below the project tree, there are tabs for "Integrati...", "Integrati...", and "Data Proj...".

The "Integration Nodes" view shows the deployment status of "MyICPTestApp" across different nodes:

- TESTNODE\_davearno (Green checkmark)
- IIB\_ICPNODE (172.23.49.19:32037) (Green checkmark)
- IIB\_DEFAULT (Green checkmark)
- MyICPTestApp (Green checkmark)

The screenshot shows the IBM Integration Bus (IIB) graphical interface. On the left, there is a palette with various integration standards: MQTT, Kafka, JMS, HTTP, REST, Web Services, SCA, and WebSphere. In the center, there is a diagram area with a blue rounded rectangle labeled "HTTP Input". Below the diagram is a toolbar with tabs: Graph (selected), User Defined Properties, Properties, Problems, Outline, Tasks, and Deployment Log. The main content area is titled "HTTP Input Node Properties - HTTP Input". It has a sidebar with tabs: Description (selected), Basic, Advanced, Input Message Parsing, and Parser Options. The "Basic" tab contains settings for working with the HTTPInput node. The "Path suffix for URL\*" field is set to "/httpinhere" with a note "e.g: /path/to/service, \". The "Use HTTPS" checkbox is unchecked.

*Testing an IIB message flow*

---

webui 4414/TCP; serverlistener 7800/TCP; nodelistener 7080/TCP

---

[webui 32037/TCP](#)

[serverlistener 32358/TCP](#)

[nodelistener 31554/TCP](#)

Test using a tool like Postman or Resteasy

<http://172.23.50.111:32358/httpinhere> do a POST and use the serverlistener port

( | chrome://resteasyc/content/resteasyc.html

# REST Easy

POST  /172.23.49.19:32358/

**+ Headers**

**- Data**

Select one of the options below to include data with the request.

Custom

Enter the data and its corresponding MIME type below.

MIME type
hello

# IIB & MQ Container on ICP from Github repository by hand

## *MQ and IIB Docker repository on github*

The screenshot shows the GitHub repository page for `Davexa/IIB-MQ`. Key statistics displayed are 12 commits, 1 branch, 0 releases, and 1 contributor. The latest commit, made by `peterajessup`, was updated to IIB 10.0.0.10 and occurred 29 days ago. The commit list includes files like `.gitignore`, `10-listener.mqsc`, `CLA.md`, `Dockerfile`, `LICENSE`, and `README.md`.

### *Git Clone the IIB-MQ repository to a local repository*

In this example I am ssh into a VM image on the Pure App that has all the icp/docker/helm clients installed. If you have all these tools you will not need to do this.

Git clone `http://github.com.Davexa/IIB-MQ.git`

```
...cloud-private-master—(mq:9.0.4.0)root@mq-da-icpqm1-ibm-mqadvan-0:/—bash ~/Downloads/deploy-ibm-cloud-private-master—ssh root@172.23.52.143
Last login: Wed Nov 29 14:49:00 on ttys001
[Daves-MacBook-Pro-2:deploy-ibm-cloud-private-master davearno$ ssh root@172.23.52.143
The authenticity of host '172.23.52.143' (172.23.52.143) can't be established.
ECDSA key fingerprint is SHA256:UclkefeyUjg+YwpVnGhh/N0FF3A1qDkZdk9lhyeBfgY4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.23.52.143' (ECDSA) to the list of known hosts.
root@172.23.52.143's password:
Last login: Mon Dec 11 13:35:56 2017 from 172.23.1.171
|-bash-4.2# ls
anaconda-ks.cfg dockerImages git ibm-cloud-auto-mgr-x86_64-2.1.0.tar.gz ldap push.sh
|-bash-4.2# cd git/
|-bash-4.2# ls
cfc-crawler container-service-getting-started-wt repos
|-bash-4.2# git clone https://github.com/Davexa/IIB-MQ.git
Cloning into 'IIB-MQ'...
remote: Counting objects: 60, done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 60 (delta 24), reused 60 (delta 24), pack-reused 0
Unpacking objects: 100% (60/60), done.
|-bash-4.2# ls
cfc-crawler container-service-getting-started-wt IIB-MQ repos
|-bash-4.2# cd IIB-MQ/
|-bash-4.2# ls
10-listener.mqsc CLA.md iib-license-check.sh kernel_settings.sh mq-dev-config mq.sh README.md switch.json
admin.json Dockerfile iib_manage.sh LICENSE mq-dev-config.sh odbc.ini setup-mqm-web.sh
agentx.json iib_env.sh iib-mq.yml login.defs mq-license-check.sh odbcinst.ini setup-var-mqm.sh
|-bash-4.2# vi Dockerfile
|-bash-4.2# docker build -t icpcluster.icp:8500/default/ii-mq:latest .
```

### *Build the IIB-MQ image*

Docker build –t icpcluster.icp:8500/default/ii-mq:latest .

```

|-bash-4.2# docker build -t icpcluster.icp:8500/default/iib-mq:latest .
Sending build context to Docker daemon 1.136 MB
Step 1/30 : FROM ubuntu:14.04
14.04: Pulling from library/ubuntu
01a4f8387457: Pull complete
c887940e680c: Pull complete
5432573ac160: Pull complete
027ee9a9665e: Pull complete
5611db80430d: Pull complete
Digest: sha256:7b3b72e6a388c24c0cc3e0d1aade3364c52f03e54bd5ab440f8fd93c69203733
Status: Downloaded newer image for ubuntu:14.04
--> d6ed29ffda6b
Step 2/30 : MAINTAINER Sam Rogers srogers@uk.ibm.com
--> Running in d914938c8e3d
--> 5719840be833
Removing intermediate container d914938c8e3d
Step 3/30 : RUN export DEBIAN_FRONTEND=noninteractive && apt-get update && apt-get install -y --no-install-recommends
coreutils curl debianutils findutils gawk grep libc-bin lsb-release mount passwd
d tar util-linux
--> Running in ea016ca1eb6a
Ign http://archive.ubuntu.com trusty InRelease
Get:1 http://security.ubuntu.com trusty-security InRelease [65.9 kB]
Get:2 http://archive.ubuntu.com trusty-updates InRelease [65.9 kB]
Get:3 http://archive.ubuntu.com trusty-backports InRelease [65.9 kB]
Get:4 http://security.ubuntu.com trusty-security/universe Sources [80.7 kB]

Step 6/30 : ARG MQ_URL=http://public.dhe.ibm.com/ibmdl/export/pub/software/websphere/messaging/mqadv/mqadv_dev903_linux_x86-64.tar.gz
--> Running in 7228d6aed3c6
--> 029ea64c09c2
Removing intermediate container 7228d6aed3c6
Step 7/30 : ARG MQ_PACKAGES="MQSeriesRuntime-*.rpm MQSeriesServer-*.rpm MQSeriesMsg*.rpm MQSeriesJava*.rpm MQSeriesJRE*.rpm MQSeriesGSKit*.rpm MQSeriesWeb*.rpm"
--> Running in ab85ccaa87a0
--> ca79cd6dd083
Removing intermediate container ab85ccaa87a0
Step 8/30 : RUN mkdir -p /tmp/mq && cd /tmp/mq && curl -LO $MQ_URL && tar -zxf ./*.tar.gz && groupadd --gid 1000 mqm && useradd -G mqm root && cd /tmp/mq/MQServer && ./mqlicense.sh -text_only -accept && rpm -ivh --force-debian $MQ_PACKAGES && /opt/mqm/bin/setmqinst -p /opt/mqm -i && rm -rf /tmp/mq && rm -rf /var/mqm && sed -i 's/PASS_MAX_DAYS/t0/' /etc/login.defs && sed -i 's/pam_unix.so/obscure sha512 minlen=8/' /etc/pam.d/common-password && sed -i 's/PASS_MIN_DAYS/t0/PASS_MIN_DAYS/t1/' /etc/login.defs && sed -i 's/pam_unix.so/obscure sha512 minlen=8/' /etc/pam.d/common-password
--> Running in fbed010d10b0
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
100  684M  100  684M    0      0  2592K  0:04:30  0:04:30  --::-- 2556K
MQServer/
MQServer/MQSeriesXRService-9.0.3-0.x86_64.rpm
MQServer/RFADMFS/

Step 14/30 : RUN mkdir /opt/ibm && curl http://public.dhe.ibm.com/ibmdl/export/pub/software/integration/10.0.0.10-IIIB-LINUX64-DEVELOPER.tar.gz | tar zx --exclude iib-10.0.0.10/tools --directory /opt/ibm && /opt/ibm/iib-10.0.0.10/iib make registry global accept license silently
--> Running in 3f88c94a015c
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
1 98 1211M  98 1193M    0      0  2620K  0:07:53  0:07:46  0:00:07 2632K^C

License accepted
Group 'mqbrkrs' will be created
--> 34240887929b
Removing intermediate container 3d648c420875
Step 15/30 : COPY kernel_settings.sh /tmp/
--> 06653ccb96e0
Removing intermediate container 48f4fa474a7d
Step 16/30 : RUN echo "IIB_10:" > /etc/debian_chroot && touch /var/log/syslog && chown syslog:adm /var/log/syslog && nsel_settings.sh;sync && /tmp/kernel_settings.sh
--> Running in 6023ablea9b3
--> c3b8c0841816
Removing intermediate container 6023ablea9b3
Step 17/30 : RUN useradd --create-home --home-dir /home/iibuser -G mqbrkrs,sudo,mqm iibuser && sed -e 's/^%sudo     .*/%sudo
D:ALL/g' -i /etc/sudoers
--> Running in 1dac954c24ab
--> 1dac954c24ab
Removing intermediate container f90d1b868247
Step 30/30 : ENTRYPOINT iib_manage.sh
--> Running in cb69dda0159d
--> f949560c55bc
Removing intermediate container cb69dda0159d
Successfully built f949560c55bc
-bash-4.2#

```

## *Copy the IIB-MQ image to the ICP repository*

Ensure the terminal can ping the correct cluster via hostname

Change the hosts file to ensure icpcluster.icp is pointing to the correct Pure App IP address in this example 172.23.50.112

```

[-bash-4.2# vi /etc/hosts
[-bash-4.2# ping icpcluster.icp
PING icpcluster.icp (172.23.49.17) 56(84) bytes of data.
64 bytes from icpcluster.icp (172.23.49.17): icmp_seq=1 ttl=63 time=0.181 ms
64 bytes from icpcluster.icp (172.23.49.17): icmp_seq=2 ttl=63 time=0.162 ms
^C
--- icpcluster.icp ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.162/0.171/0.181/0.016 ms
[-bash-4.2# docker login icpcluster.icp:8500
[Username (admin): admin
[Password:
Login Succeeded

```

## Docker login

Log in with admin/admin using command

Docker login icpcluster.icp:8500

Check the contents of the repository

```

[-bash-4.2# docker images
REPOSITORY                                TAG      IMAGE ID      CREATED        SIZE
icpcluster.icp:8500/default/iib-mq        latest   f949560c55bc  4 minutes ago  2.22 GB
icpcluster.icp:8500/default/watson-talk   latest   18d3b4a1f59f  5 days ago    746 MB
icpcluster.icp:8500/default/watson         latest   fcb436d4b877  5 days ago    746 MB
node                                         6        2fcc51d4e73d  6 days ago    662 MB
hitomitak/agentless-crawler                latest   96e469d3dfaf  6 days ago    712 MB
icpcluster.icp:8500/default/agentless-crawler  latest   96e469d3dfaf  6 days ago    712 MB

```

Docker push the image to ICP

```

-----
[-bash-4.2# docker push icpcluster.icp:8500/default/iib-mq:latest
The push refers to a repository [icpcluster.icp:8500/default/iib-mq]
fd298fd47be7: Pushed
00387f1a1382: Pushed
b5a36b864b9c: Pushed
580eba8bd60b: Pushed

c088f4b849d4: Pushed
c08b59ef4a3d: Pushed
latest: digest: sha256:55398b2aba08f71bc8dc499323fc3e700566b17f03b9bc14a2aa86e5a5f51b0d size: 5735

```

Check it was successful

```

[-bash-4.2# docker images
REPOSITORY                                TAG      IMAGE ID      CREATED        SIZE
icpcluster.icp:8500/default/iib-mq        latest   f949560c55bc  12 minutes ago  2.22 GB
icpcluster.icp:8500/default/watson-talk   latest   18d3b4a1f59f  5 days ago    746 MB

```

*Check the IIB-MQ image via the ICP Console*

The screenshot shows the IBM Cloud Private interface. On the left, there's a sidebar with a blue header 'IBM Cloud Private' and a white body containing 'Dashboard', a collapsed 'Platform' section (Nodes, Network, Storage), and a selected 'Images' section. The main area is titled 'Images' and shows a list of Docker images. A search bar at the top says 'Search items'. Below it, a dropdown shows '20 items per page | 1-20 of 25 items'. The list is sorted by 'NAME' (with an arrow indicating it's sorted ascending). The images listed are: default/agentless-crawler, default/cam-broker, default/cam-busybox, default/cam-mongo, default/cam-orchestration, default/cam-portal-api, default/cam-portal-ui, default/cam-redis, default/cam-service-composer-api, default/cam-service-composer-ui, default/cardashboard, default/icpmsbdemo, and default/iib-mq. The 'default/iib-mq' entry is highlighted with a light gray background.

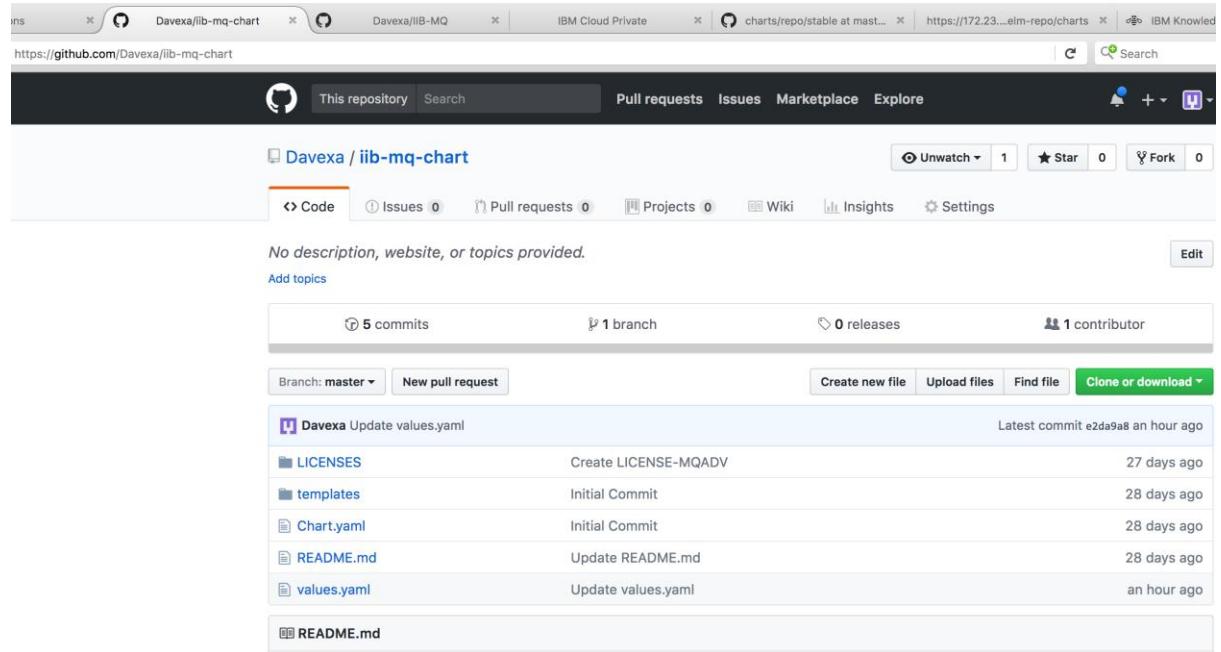
Click on default/iib-mq

The screenshot shows the 'default/iib-mq' image details page. The top navigation bar says 'IBM Cloud Private' and the URL is 'Images / default/iib-mq /'. The page has a 'Overview' tab selected. Below it, there's a 'Image details' section with a table. The table has two columns: 'TYPE' and 'DETAIL'. The rows are: Name (default/iib-mq), Owner (default), Scope (global), and Tags (empty).

TYPE	DETAIL
Name	default/iib-mq
Owner	default
Scope	global
Tags	

# Adding a new (custom) IIB and MQ Helm chart to ICP

## *MQ and IIB Helm Chart repository on github*



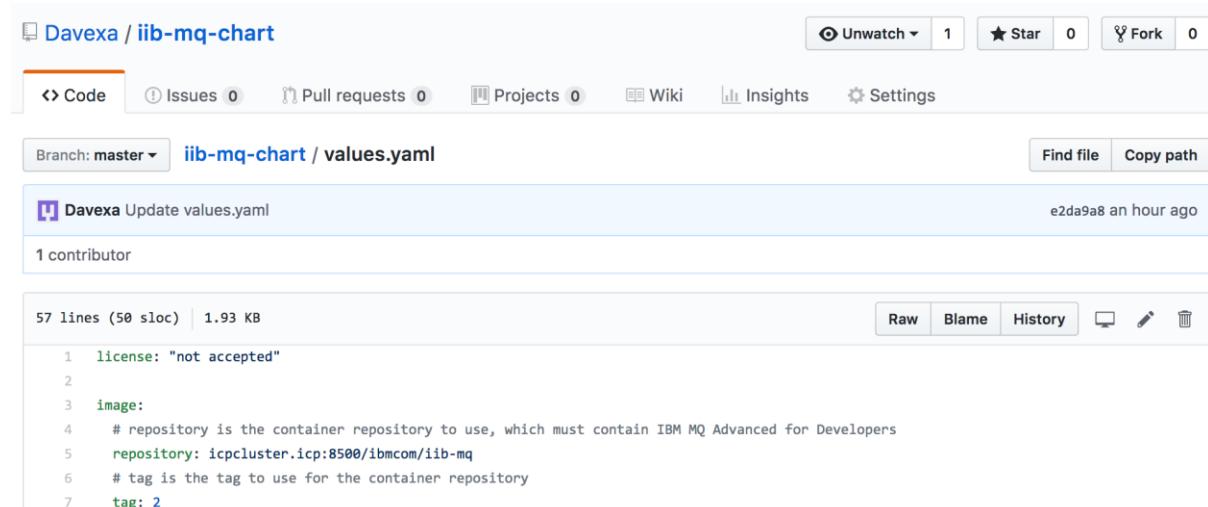
The screenshot shows the GitHub repository page for `Davexa/iib-mq-chart`. The repository has 5 commits, 1 branch, 0 releases, and 1 contributor. The latest commit was made an hour ago. The repository contains files such as `LICENSES`, `templates`, `Chart.yaml`, `README.md`, and `values.yaml`.

### Edit the values.yaml file

At this point you have a choice to make.

- 1) Edit and commit the values.yaml here on github
- 2) Edit the values.yaml after the git clone to local system leaving the github copy “vanilla”

In this example we'll perform the edit on Github and then commit before the clone.



The screenshot shows the GitHub file editor for `iib-mq-chart/values.yaml`. The file contains the following YAML code:

```
license: "not accepted"
image:
  # repository is the container repository to use, which must contain IBM MQ Advanced for Developers
  repository: icpcluster.icp:8500/ibmcom/iib-mq
  # tag is the tag to use for the container repository
  tag: 2
```

Edit the values.yaml file and ensure the repository and tag are correct.

The tag will need to be **latest**

The repository will need to resolve to the icp instance and check that the namespace is correct **default** in this case.

Set the license to “accept”



```
license: "accept"
image:
  # repository is the container repository to use, which must contain IBM MQ Advanced for Developers
  repository: icpcluster.icp:8500/default/iib-mq
  # tag is the tag to use for the container repository
  tag: latest
  # pullSecret is the secret to use when pulling the image from a private registry
```

Page down and fill in the IIB and MQ specific values. Remember the spaces after the colons.

```
queueManager:
  name: icpqm1
  dev:
    adminPassword: passw0rd
    appPassword: passw0rd
  iib:
    iibnodename: icpnodel
    iibservername: default
  nameOverride:
```

### *Git Clone the Helm Chart repository to local machine (not ICP in this example)*

In this case the local machine is the VM image on the Pure App where all the command line tools are installed. If you have this set up on your machine you will not need to be ssh into the Pure.

```
[~]# ls
10-listener.mqsc  CLA.md      iib-license-check.sh  kernel_settings.sh  mq-dev-config      mq.sh          README.md      switch.json
admin.json        Dockerfile   iib_manage.sh     LICENSE           mq-dev-config.sh  odbc.ini      setup-mqm-web.sh
agentx.json       iib_env.sh  iib-mq.yml       login.defs       mq-license-check.sh  odbcinst.ini  setup-var-mqm.sh
[~]# cd ..
[~]# ls
cfc-crawler  container-service-getting-started-wt  IIB-MQ  repos
[~]# git clone https://github.com/Davexa/iib-mq-chart.git
Cloning into 'iib-mq-chart'...
remote: Counting objects: 28, done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 28 (delta 10), reused 20 (delta 6), pack-reused 0
Unpacking objects: 100% (28/28), done.
[~]# ls
cfc-crawler  container-service-getting-started-wt  IIB-MQ  iib-mq-chart  repos
```

## Configure the Kubernetes client to connect to ICP

### Configure kubectl

Before you run commands in the kubectl command line interface for this cluster, you must configure the client.

#### Prerequisites:

Install the kubectl CLI: kubectl

To configure the CLI, paste the displayed configuration commands into your terminal window and run them.:

```
kubectl config set-cluster icpcluster.icp --server=https://172.23.49.17:8001 --insecure-skip-tls-verify=true
kubectl config set-context icpcluster.icp-context --cluster=icpcluster.icp
kubectl config set-credentials icpcluster.icp-user --token=eyJhbGciOiJSUzI1NiJ9.eyJzdWIi0iJhZG1pbilsmF0X2hhc2gi0iJke18yVU9od0xfZnBrNGkx
ZEJhR22ZniwiaKNzIjoiHR0chMM6Ly9pY3BjhVzdGVyLmjcd0SN0gzL29pZGMvZW5kcG9pbnQvT1AiLCJhdWQ0iJKMzFkZDgXVmNjg3ZjIxMzE1Y2ZjYTETy5ZjI4MiIsImV4CC16MTU
xHzaA40DE0NSwiaWF0IjoXNTExMDQ0OTQ1fQ.Nfd5indDszngvF8qMYRGUsrKawNuf_YD13Hl9nd5oPAwoijPnm9bxjfE6Y0LGdutEb0wlDnHx1z0U2j4eeoAM1D0x4Ba5A-YM5Mc6mkT
d8ng08Sj6p5D3yR_xm50D3gyUvNAWBMG6hb9sN0802rUlaCpj0wm9j7nEdelsdkt-YkicDMGmijqSxwV63p_08Z1hMznuk_Td51bkEFBNdzVfllrYdMDgTvfmss_zz7jR081j6ycghIxj2lHo
P-TL7N0Pd0ifkQK-b1aL8tXmdG20BIMkdCvDcztL3fx-7w78mAiG1mt444ka9NWxk6c5-SrcUjnqImnwA
User "icpcluster.icp-user" set.
-bash-4.2# kubectl config set-context icpcluster.icp-context --user=icpcluster.icp-user --namespace=default
Context "icpcluster.icp-context" modified.
-bash-4.2# kubectl config use-context icpcluster.icp-context
Switched to context "icpcluster.icp-context".
```

### Install the Helm Chart from the local machine (not ICP UI in this example)

```
[root@localhost ~]# bash-4.2# pwd
/root/git
[root@localhost ~]# bash-4.2# ls
cfc-crawler container-service-getting-started-wt IIB-MQ iib-mq-chart repos
[root@localhost ~]# bash-4.2# helm install --name daiibmq iib-mq-chart
```

```
NAME: daiibmq
LAST DEPLOYED: Tue Dec 12 15:39:37 2017
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1beta1/StatefulSet
NAME          DESIRED   CURRENT   AGE
daiibmq-iib-mq 1         1         1s

==> v1/Secret
NAME          TYPE      DATA   AGE
daiibmq-iib-mq  Opaque    1      1s

==> v1/Service
NAME          CLUSTER-IP  EXTERNAL-IP  PORT(S)           AGE
daiibmq-iib-mq  10.0.0.251 <nodes>     1414:30447/TCP,9443:30019/TCP,4414:30116/TCP,7080:31450/TCP,7800:31847/TCP  1s
```

*Check the deploy in the ICP console*

The screenshot shows the IBM Cloud Private dashboard. In the top navigation bar, there is a blue button with a white 'X' icon and the text "IBM Cloud Private". Below the navigation bar, the word "Dashboard" is displayed in bold. Under "Platform", there is a section titled "Workloads" with a downward arrow icon. Below "Workloads", the heading "Helm releases" is shown, followed by a table listing one deployment.

Name	Namespace	Status	Updated	Type
daiibmq	default	DEPLOYED	Dec 12th 2017	iib-mq

Below the table, a detailed view of the "daiibmq" deployment is shown. It includes sections for "DETAILS", "SECRET", and "SERVICE".

**DETAILS**

TYPE	DETAIL
Name:	daiibmq
Namespace:	default
Status:	DEPLOYED
Chart name:	iib-mq
Chart version:	1.1.0
updated:	Dec 12th 2017 at 3:39 PM
Type:	Release

**SECRET**

NAME	TYPE	DATA
daiibmq-iib-mq	Opaque	1

**SERVICE**

NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)
daiibmq-iib-mq	10.0.0.251	<nodes>	1414:30447/TCP,9443:30019/TCP,4414:30116/TCP,7080:31450/TCP,7800:31847/TCP

Select the Stateful set

StatefulSet / daiibmq-iib-mq / daiibmq-iib-mq-0 /

## daiibmq-iib-mq-0

Overview Containers Events Logs

### Pod details

TYPE	DETAIL
Name	daiibmq-iib-mq-0
Namespace	default
Start time	Dec 12th 2017 at 3:53 PM
Labels	QM_IDENTIFIER=daiibmq,app=daiibmq-iib-mq,chart=iib-mq-1.1.0,controller-revision-hash=daiibmq-iib-mq-4259841562,heritage=Tiller,release=daiibmq
PodSecurityPolicy applied	default
Node	172.23.49.9
IP	10.1.20.134
Status	Running

Overview Containers Events Logs

NAME	IMAGE	PORT	STATE
qmgr	icpcluster.icp:8500/default/iib-mq:latest	1414/TCP;9443/TCP;4414/TCP;7080/TCP;7800/TCP	Running

Overview Containers Events Logs

Search items

20 ▾ items per page | 1-5 of 5 items

TYPE	SOURCE	COUNT	REASON	MESSAGE
Normal	default-scheduler	1	Scheduled	Successfully assigned daiibmq-iib-mq-0 to 172.23.49.9
Normal	kubelet 172.23.49.9	1	SuccessfulMountVolume	MountVolume.SetUp succeeded for volume "default-token-qrz9z"
Normal	kubelet 172.23.49.9	1	Pulled	Container image "icpcluster.icp:8500/default/iib-mq:latest" already present on machine
Normal	kubelet 172.23.49.9	1	Created	Created container
Normal	kubelet 172.23.49.9	1	Started	Started container

## daiibmq-iib-mq-0

Overview    Containers    Events    Logs

```
BIP8071I: Successful command completion.
-----
Starting syslog
-----
Configuring db access
BIP8071I: Successful command completion.
Starting node icpnodel
BIP8096I: Successful command initiation, check the system log to ensure that the component has started.
Starting Switch Server
-----
S starting Switch
Starting iibswitch, please wait...
iibswitch started.
BIP8071I: Successful command completion.
BIP8071I: Successful command completion.
BIP8096I: Successful command initiation, check the system log to ensure that the component has started.
QMNAME(icpqml)                                     STATUS(Running)
IBM MQ Queue Manager icpqml is now fully running
```

# IIB & MQ Container on ICP from Github repository via MSB pipeline

*Creating the IIB-MQ Repository in GitHub Organization*

*Source GitHub Repositories to leverage*

## 1. IIB an MQ Docker build repository

No description, website, or topics provided.

Add topics

12 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

peterajessup Updated to IIB 10.0.0.10 Latest commit 99dd9d1 on 13 Nov

File	Commit Message	Time
.cignore	Initial Commit	8 months ago
.gitignore	Initial Commit	8 months ago
10-listener.mqsc	Initial Commit	8 months ago
CLA.md	Initial Commit	8 months ago
Dockerfile	Changed to IIB 10.0.0.10	a month ago

## 2. IIB and MQ Helm Chart repository

No description, website, or topics provided.

Add topics

9 commits 1 branch 0 releases 2 contributors

Branch: master New pull request Create new file Upload files Find file

Davexa Update values.yaml Latest commit 99dd9d1 on 13 Nov

File	Commit Message	Time
LICENSES	Create LICENSE-MQADV	Nov 13, 2023
templates	Initial Commit	Nov 13, 2023
Chart.yaml	Initial Commit	Nov 13, 2023
README.md	Update README.md	Nov 13, 2023
values.yaml	Update values.yaml	Nov 13, 2023

*Target GitHub Repository in Github Organization*

https://github.com/organizations/DAVEXACOM/repositories/new

This organization Search Pull requests Issues Marketplace Explore

### Create a new repository

A repository contains all the files for your project, including the revision history.

Owner Repository name

DAVEXACOM / IIB-MQ ✓

Great repository names are short and memorable. Need inspiration? How about [jubilant-engine](#).

Description (optional)

IIBv10 and MQv9 docker build repository

**Public**  
Anyone can see this repository. You choose who can commit.

**Private**  
You choose who can see and commit to this repository.

Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None Add a license: None ⓘ

**Create repository**

## Import from existing repos

https://github.com/DAVEXACOM/IIB-MQ/import

This repository Search Pull requests Issues Marketplace Explore

### Import your project to GitHub

Import all the files, including the revision history, from another version control system.

Your old repository's clone URL

Learn more about the types of [supported VCS](#).

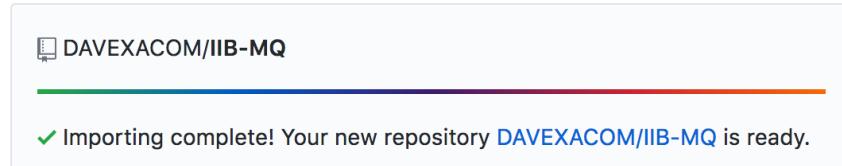
Your existing repository

DAVEXACOM/IIB-MQ [Change repository](#)

[Cancel](#) **Begin import**

## Preparing your new repository

There is no need to keep this window open, we'll email you when the import is done.



A screenshot of a GitHub repository page. The repository name is "DAVEXACOM / IIB-MQ". The page shows basic statistics: 12 commits, 1 branch, 0 releases, and 1 contributor. A commit list is shown, all made by user "peterajessup" on Nov 13. The commits include ".cignore", ".gitignore", "10-listener.mqsc", "CLA.md", "Dockerfile", "LICENSE", "README.md", "admin.json", and "anentx.json". The Dockerfile was updated to IIB 10.0.0.10. The README.md was updated to reflect this change. The commits are dated from 8 months ago to a month ago.

File	Type	Message	Date
.cignore	Initial Commit		8 months ago
.gitignore	Initial Commit		8 months ago
10-listener.mqsc	Initial Commit		8 months ago
CLA.md	Initial Commit		8 months ago
Dockerfile	Changed to IIB 10.0.0.10		a month ago
LICENSE	Initial Commit		8 months ago
README.md	Update README.md		8 months ago
admin.json	Initial Commit		8 months ago
anentx.json	Initial Commit		8 months ago

### Add a Jenkins file to the repository

Use Create new file to create the Jenkinsfile

IIBv10 and MQv9 docker build repository

Add topics



Name the file Jenkinsfile

Cut and paste the following or similar – you might want to change the image name if you prefer

```
#!groovy
```

```

@Library('MicroserviceBuilder') _

microserviceBuilderPipeline {

    image = 'iib10mq9'

    mavenImage = 'wwdemo/images:maven-lab'

    deployBranch = 'master'

    namespace = 'default'

}

```

<https://github.com/DAVEXACOM/IIB-MQ/new/master>

The screenshot shows a GitHub repository page for 'DAVEXACOM / IIB-MQ'. At the top, there's a navigation bar with a GitHub icon, 'This repository', and a search bar. Below the header, the repository name 'DAVEXACOM / IIB-MQ' is displayed with a 'Code' button. Underneath, there are tabs for 'Issues 0' and 'Pull requests 0'. A search bar at the bottom has 'Jenkinsfile' typed into it, with 'cancel' as an option. The main area contains a code editor with the following Groovy script:

```

1  #!groovy
2
3  @Library('MicroserviceBuilder') _
4  microserviceBuilderPipeline {
5      image = 'iib10mq9'
6      mavenImage = 'wwdemo/images:maven-lab'
7      deployBranch = 'master'
8      namespace = 'default'
9 }

```

Page down and hit commit

*Add the IIB and MQ Helm chart files to the repository*

Create the directory structure chart/iibmq/templates and chart/iibmq/LICENSE

<https://github.com/DAVEXACOM/IIB-MQ/new/master>

This screenshot shows a GitHub repository page for 'DAVEXACOM / IIB-MQ'. A modal window is open for creating a new file named 'test.yaml'. The modal has tabs for 'Edit new file' and 'Preview'. The preview area contains the number '1'. The URL in the address bar is 'https://github.com/DAVEXACOM/IIB-MQ/new/master'.

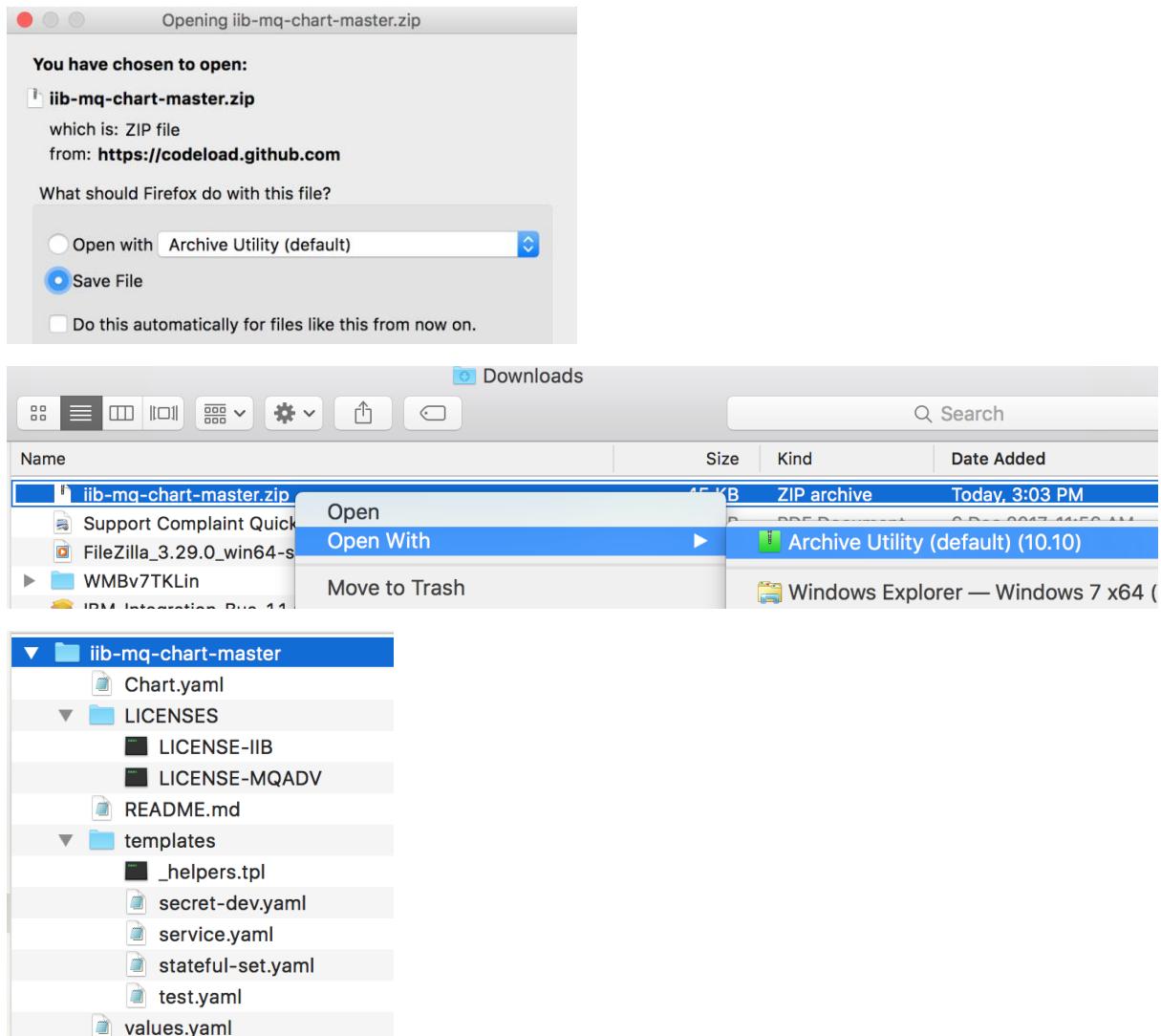
<https://github.com/DAVEXACOM/IIB-MQ/new/master/chart/iibmq/LICENSES>

This screenshot shows a GitHub repository page for 'DAVEXACOM / IIB-MQ'. A modal window is open for creating a new file named 'LICENSE-MQADV'. The modal has tabs for 'Edit new file' and 'Preview'. The preview area contains the number '1'. The URL in the address bar is 'https://github.com/DAVEXACOM/IIB-MQ/new/master/chart/iibmq/LICENSES'.

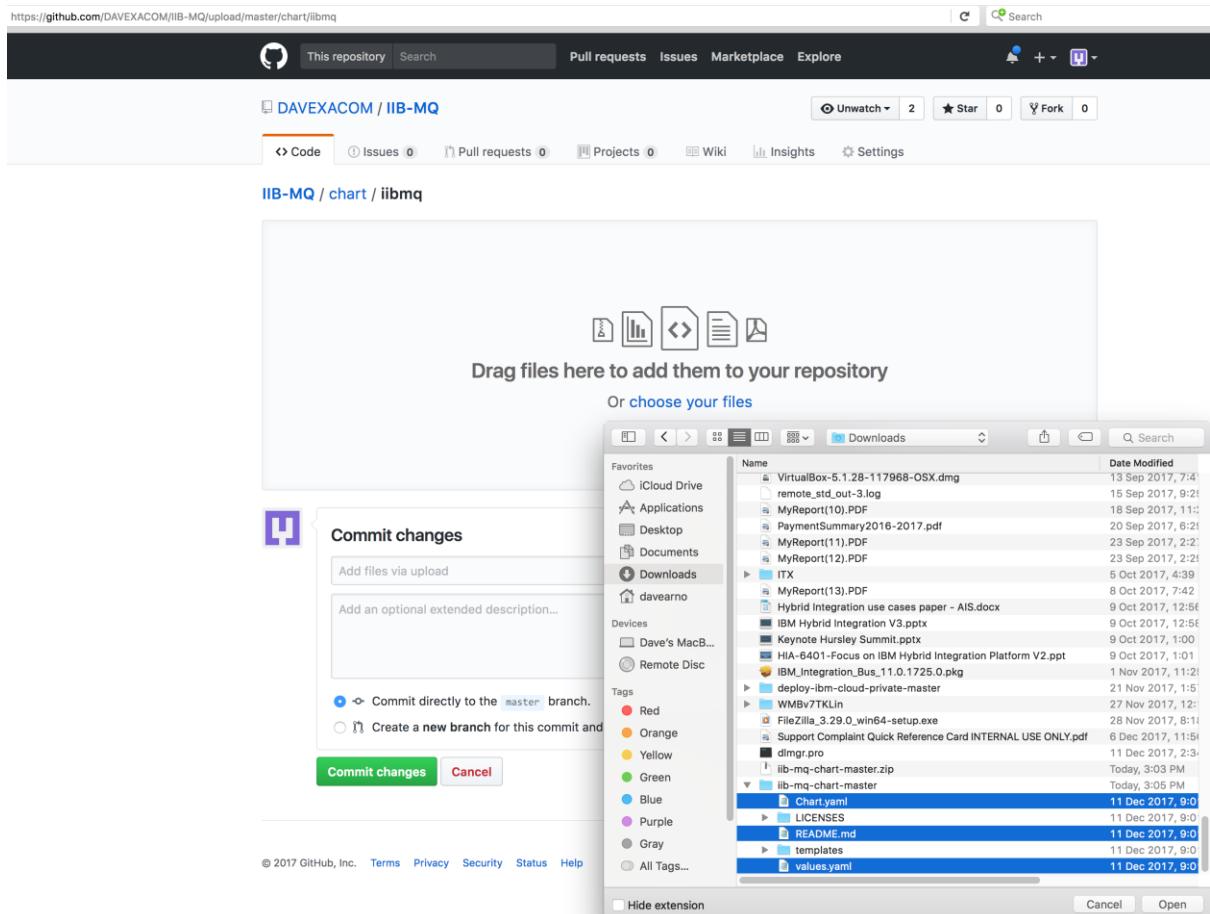
## Download from the iib-mq-chart repository to the local file system

<https://github.com/Davexa/iib-mq-chart>

This screenshot shows a GitHub repository page for 'Davexa / iib-mq-chart'. The repository has 9 commits, 1 branch, 0 releases, and 2 contributors. It includes links for 'Create new file', 'Upload files', 'Find file', and a prominent green 'Clone or download' button. A sidebar on the right provides cloning options via HTTPS or SSH, along with links to 'Open in Desktop' and 'Download ZIP'. The URL in the address bar is 'https://github.com/Davexa/iib-mq-chart'.



## Import to from the iib-mq-chart repository to the local file system



## Commit changes

The 'Commit changes' dialog is shown again, overlaid on the GitHub interface. The local file browser is still visible, showing the same set of files as in the previous screenshot. The GitHub interface elements like the header and other repository details are partially obscured by the dialog.

https://github.com/DAVEEXACOM/IIB-MQ/upload/master/chart/iibmq/templates

The screenshot shows a GitHub repository page for 'DAVEEXACOM / IIB-MQ'. A modal dialog is open for committing changes. The left side of the dialog shows a file tree with the path 'lib-mq-chart-master/templates'. The 'templates' folder contains several YAML files: '\_helpers.tpl', 'secret-dev.yaml', 'service.yaml', 'stateful-set.yaml', and 'test.yaml'. The right side of the dialog displays a table of files with their names, date modified, and file type icons. At the bottom of the dialog are 'Commit changes' and 'Cancel' buttons.

Name	Date Modified
Chart.yaml	11 Dec 2017, 9:01 PM
LICENSES	11 Dec 2017, 9:01 PM
README.md	11 Dec 2017, 9:01 PM
templates	11 Dec 2017, 9:01 PM
_helpers.tpl	11 Dec 2017, 9:01 PM
secret-dev.yaml	11 Dec 2017, 9:01 PM
service.yaml	11 Dec 2017, 9:01 PM
stateful-set.yaml	11 Dec 2017, 9:01 PM
test.yaml	11 Dec 2017, 9:01 PM
values.yaml	11 Dec 2017, 9:01 PM

**Commit changes**

Add files via upload

Add an optional extended description...

Commit directly to the `master` branch.

Create a new branch for this commit and...

**Commit changes** **Cancel**

https://github.com/DAVEXACOM/IIB-MQ/upload/master/chart/iibmq/LICENSES

The screenshot shows a GitHub commit dialog over a file browser window. The file browser displays a folder structure for 'iib-mq-chart-master' containing 'Chart.yaml', 'LICENSES', 'LICENSE-IIB', and 'LICENSE-MQADV'. The 'LICENSES' file is selected. The commit dialog has fields for 'Add files via upload' and 'Add an optional extended description...', with the 'Commit directly to the master branch.' option selected. Buttons for 'Commit changes' and 'Cancel' are at the bottom.

DAVEXACOM / IIB-MQ

IIB-MQ / chart / iibmq / LICENSES

LICENSE-IIB

LICENSE-MQADV

### Commit changes

Add files via upload

Add an optional extended description...

Commit directly to the `master` branch.

Create a new branch for this commit and :

**Commit changes** **Cancel**

Update the Jenkinsfile to reference the helm charts files in the chart folder

Before :

<https://github.com/DAVEXACOM/IIB-MQ/blob/master/Jenkinsfile>

The screenshot shows a GitHub repository page for 'DAVEXACOM / IIB-MQ'. The 'Code' tab is selected. A 'Branch: master' dropdown shows 'IIB-MQ / Jenkinsfile'. The code editor displays the following Jenkinsfile:

```
1  #!groovy
2
3  @Library('MicroserviceBuilder') _
4  microserviceBuilderPipeline {
5      image = 'iib10mq9'
6      mavenImage = 'wwdemo/images:maven-lab'
7      deployBranch = 'master'
8      namespace = 'default'
9 }
```

After :

The screenshot shows the same GitHub repository page for 'DAVEXACOM / IIB-MQ'. The 'Code' tab is selected. A 'Branch: master' dropdown shows 'IIB-MQ / Jenkinsfile'. The code editor displays the following Jenkinsfile, which includes additional configuration for a chart folder:

```
1  #!groovy
2
3  @Library('MicroserviceBuilder') _
4  microserviceBuilderPipeline {
5      image = 'iib10mq9'
6      mavenImage = 'wwdemo/images:maven-lab'
7      chartFolder = 'chart/iibmq'
8      deployBranch = 'master'
9      namespace = 'default'
10 }
```

### *Important Parameters used in this example*

As you go through the set up you need to capture these values as you go for use in later steps

Parameter Description	My example value	details
Github organization	DAVEXACOM	Github details
Github repository	IIB-MQ	Github details
Github username	Davexa	Github details
Github Personal access token	23a8537c82a19a358fb957bbfa7fba3d369 eeb6b (not the real token)	Github details
Application Name	Icp-msb-iib-demo	Github OAuth registration
Homepage URL	<a href="http://172.23.50.111:32050">http://172.23.50.111:32050</a> port dependent on Jenkins instance	Github OAuth registration needs the Jenkins URL
Authorization Call back URL	http://172.23.50.111:32050/securityRealm/finishLogin	Github OAuth registration needs the Jenkins URL
Client ID	e681fbc9e6491de43587	Github OAuth reg returns
mavenImage	'wwdemo/images:maven-lab'	Jenkins File
URL to OAuth Application details	<a href="https://github.com/settings/applications/635182">https://github.com/settings/applications/635182</a>	Github details
Github webhook payload URL	{Jenkins IpAddress}:[Jenkins port]/github-webhook/	URL github uses to publish to jenkins. Might be a Secure G/W endpoint rather than direct

## *Configuring Github repository*

### Github repository IIB-MQ in organization DAVEXACOM

https://github.com/DAVEXACOM/IIB-MQ

The screenshot shows the GitHub repository page for 'IIB-MQ' under the 'DAVEXACOM' organization. The repository description is 'IIBv10 and MQv9 docker build repository'. It has 28 commits, 1 branch, and 0 releases. A recent commit by 'Davexa' is shown: 'updated the BAR file with new flow content'. The repository contains files like 'chart/iibmq', '.cignore', '.gitignore', '10-listener.mqsc', 'CLA.md', 'Dockerfile', 'ICPDeploy.bar', and 'Jenkinsfile'. The Dockerfile and ICPDeploy.bar files have specific commit messages.

File	Commit Message
Dockerfile	Update Dockerfile
ICPDeploy.bar	updated the BAR file with new flow content
Jenkinsfile	Update Jenkinsfile

### *Github Personal Access Token – Create a token*

#### Developer Settings – Personal Access tokens

The screenshot shows the GitHub developer settings page. The user is signed in as 'Davexa'. The 'Settings' option in the sidebar is highlighted. The main area displays sections for 'Your profile', 'Your stars', and 'Your Gists'. A 'Help' link is also visible.

Select Developer settings

## Developer settings

You can @mention your company's GitHub organization to link it.

## Organization settings



DAVEXACOM

## Location

Sydney

Select personal access tokens



Search GitHub

[Settings](#) / [Developer settings](#)

### OAuth Apps

### GitHub Apps

### Personal access tokens

Generate new token

[Settings](#) / [Developer settings](#)

OAuth Apps

GitHub Apps

Personal access tokens

### Personal access tokens

[Generate new token](#)

Need an API token for scripts or testing? [Generate a personal access token](#) for quick access to the [GitHub API](#).

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).



Confirm password to continue

Password

[Forgot password?](#)

.....

[Confirm password](#)

## New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

### Token description

Token for remote DevOps tooling

What's this token for?

### Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories
<input type="checkbox"/> <b>repo:status</b>	Access commit status
<input type="checkbox"/> <b>repo_deployment</b>	Access deployment status
<input type="checkbox"/> <b>public_repo</b>	Access public repositories
<input type="checkbox"/> <b>repo:invite</b>	Access repository invitations
<input checked="" type="checkbox"/> <b>admin:org</b>	Full control of orgs and teams
<input type="checkbox"/> <b>write:org</b>	Read and write org and team membership
<input type="checkbox"/> <b>read:org</b>	Read org and team membership
<input checked="" type="checkbox"/> <b>admin:public_key</b>	Full control of user public keys
<input type="checkbox"/> <b>write:public_key</b>	Write user public keys
<input type="checkbox"/> <b>read:public_key</b>	Read user public keys
<input checked="" type="checkbox"/> <b>admin:repo_hook</b>	Full control of repository hooks
<input type="checkbox"/> <b>write:repo_hook</b>	Write repository hooks
<input type="checkbox"/> <b>read:repo_hook</b>	Read repository hooks
<input checked="" type="checkbox"/> <b>admin:org_hook</b>	Full control of organization hooks
<input checked="" type="checkbox"/> <b>gist</b>	Create gists
<input checked="" type="checkbox"/> <b>notifications</b>	Access notifications
<input checked="" type="checkbox"/> <b>user</b>	Update all user data
<input type="checkbox"/> <b>read:user</b>	Read all user profile data
<input type="checkbox"/> <b>user:email</b>	Access user email addresses (read-only)
<input type="checkbox"/> <b>user:follow</b>	Follow and unfollow users
<input type="checkbox"/> <b>delete_repo</b>	Delete repositories
<input checked="" type="checkbox"/> <b>admin:gpg_key</b>	Full control of user gpg keys <a href="#">(Developer Preview)</a>
<input type="checkbox"/> <b>write:gpg_key</b>	Write user gpg keys
<input type="checkbox"/> <b>read:gpg_key</b>	Read user gpg keys

**Generate token**

[Cancel](#)

Hit Generate token

## Personal access tokens

[Generate new token](#)

[Revoke all](#)

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your new personal access token now. You won't be able to see it again!

✓ 23a8537c82a19a358fb957bbfa7fba 

[Edit](#)

[Delete](#)

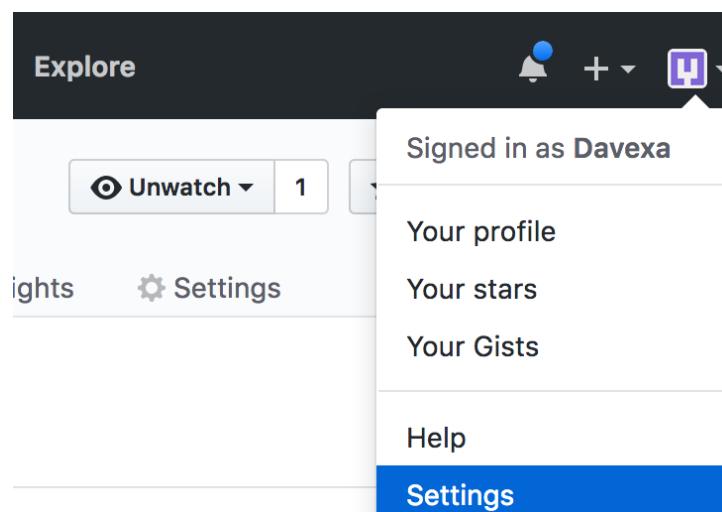
Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Copy and paste the token for later use.

## OAuth Apps – Register a new application

Register a new OAuth application

## Developer Settings – Set up OAuth



The screenshot shows the GitHub developer settings interface. At the top, there's a dark header bar with the word 'Explore' on the left, a bell icon, a plus sign, and a user profile icon. Below this is a navigation bar with 'Signed in as Davexa'. The 'Settings' option in this bar is highlighted with a blue background. To the left of the main content area, there are two buttons: 'Unwatch' with a count of 1 and a gear icon labeled 'Settings'. The main content area has sections for 'Your profile', 'Your stars', and 'Your Gists'. At the bottom of this section is a 'Help' link and a 'Settings' button, which is also highlighted with a blue background.

Select Developer settings

[Developer settings](#)

You can @mention your company's GitHub organization to link it.

[Organization settings](#)

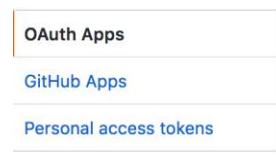


[Location](#)

Sydney

## OAuth Apps – Register a new application

Register a new OAuth application



The screenshot shows the GitHub OAuth apps registration page. On the left, there's a sidebar with three options: 'OAuth Apps' (which is selected and highlighted with a border), 'GitHub Apps', and 'Personal access tokens'. The main content area has a heading 'No OAuth applications' and a sub-heading 'OAuth applications are used to access the GitHub API. [Read the docs](#) to find out more.' At the bottom of this area is a green 'Register a new application' button.

No OAuth applications

OAuth applications are used to access the GitHub API. [Read the docs](#) to find out more.

[Register a new application](#)

Important! Pick a port number for the pipeline. You will use it when you create the pipeline. (you may not get it right first time as the port might be in use when you create the pipeline. If you have to use a different port at pipeline creation time. You will need to come back here and change this port in Github OAuth application. So do not close this window.

Lets assume 32051 for now!

#### Application name

icp-msb-iib-demo

Something users will recognize and trust

#### Homepage URL

http://172.23.50.111:32051

The full URL to your application homepage

#### Application description

Application description is optional

This is displayed to all users of your application

#### Authorization callback URL

http://172.23.50.111:32051/securityRealm/finishLogin

Your application's callback URL. Read our [OAuth documentation](#) for more information.

Click the green register application button

- [OAuth Apps](#)
- [GitHub Apps](#)
- [Personal access tokens](#)

## icp-msb-iib-demo

 Davexa owns this application.

You can list your application in the [GitHub Marketplace](#) so that other users can discover it.

### 1 user

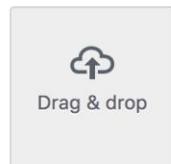
**Client ID**  
e681fbc9e6491de43587

**Client Secret**  
aef2f82c12acabd3218de28d7f63b1fa25560096

[Revoke all user tokens](#)

[Reset client secret](#)

### Application logo



[Upload new logo](#)

You can also drag and drop a picture from your computer.

### Application name

icp-msb-iib-demo

Something users will recognize and trust

### Homepage URL

http://172.23.50.111:32051

The full URL to your application homepage

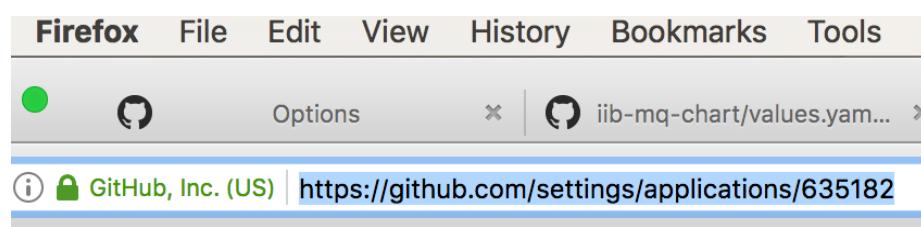
### Application description

## Capture OAuth Client ID and Secret

Copy and Paste the Client ID – you'll need it later.

Copy and Paste the Client Secret – you'll need it later.

Copy of the Oauth Application Github URL from the Browser URL bar as well.



## *IBM Cloud Private Console – Check Micro Service Builder Fabric Service*

The Micro Service Builder Pipeline pre-reqs that a Micro Service Builder Fabric service has been deployed into the ICP instance in the same namespace that your MSB pipeline (Jenkins) will be deployed.

From the ICP Console select Workloads->Helm Releases and filter on “fabric”

The screenshot shows the 'Helm releases' section of the IBM Cloud Private console. A search bar at the top contains the text 'fabric'. Below it, a table lists one item: 'ibm-msb-fabric' under the NAME column, 'developers' under the NAMESPACE column, 'DEPLOYED' under the STATUS column, and 'Jan 8th 2018' under the UPDATED column. The CHART NAME column shows 'ibm-microservicebuilder-fabric'. There are dropdown menus for items per page (set to 20) and a link to item 1-1 of 1 items.

NAME	NAMESPACE	STATUS	UPDATED	CHART NAME
ibm-msb-fabric	developers	DEPLOYED	Jan 8th 2018	ibm-microservicebuilder-fabric

If there is no helm release deploy for ibm-microservicebuilder-fabric showing as DEPLOYED in the namespace that you will deploy your pipeline talk to your ICP administrator about getting one set up.

## *IBM Cloud Private Console – Check Jenkins Persistent Volume*

Depending on the nature of your Jenkins set up you may want to use a persistent store. In these examples I don't use one. However, below shows you have to check if a persistent store for Jenkins has already been set up.

Log into the console

Select Platform->Storage

Filter on jenkins

The screenshot shows the 'Storage' section of the IBM Cloud Private console, specifically the 'PersistentVolume' tab. A search bar at the top contains the text 'jen'. Below it, a table lists one item: 'jenkins-pv1' under the NAME column, 'NFS' under the TYPE column, '8Gi' under the CAPACITY column, 'RWO' under the ACCESS MODE column, 'Recycle' under the RECLAIM POLICY column, and 'Available' under the STATUS column.

NAME	TYPE	CAPACITY	ACCESS MODE	RECLAIM POLICY	STATUS
jenkins-pv1	NFS	8Gi	RWO	Recycle	Available

If there is no persistent volume that can be identified for use by jenkins then either create one or ask the ICP administrator.

## *IBM Cloud Private Console – Deploy MicroServiceBuild Pipeline*

Log into the console

Select Catalog

Filter on pipe

IBM Cloud Private

## Catalog

pipe

### Helm charts

Deploy your applications and install software packages



**ibm-microservicebuilder pipeline**  
Microservice Builder Pipeline  
ibm-charts



**spring-cloud-data-flow**  
Open source, multi-cloud toolkit for  
incubator

Select ibm-microservicebuilder pipeline

Configure the ibm-microservicebuilder pipeline helm chart

Hit the configure button

### Installing the Chart

To install the chart with the release name `pipeline`:

```
helm install --name pipeline ibm-microservicebuilder-pipeline
```

This command deploys a Microservice Builder Jenkins pipeline on the Kubernetes cluster. The `configuration` section lists the parameters that can be configured during installation.

**Tip:** See all the resources deployed by the chart using `kubectl get all -l release=pipeline`

### Uninstalling the Chart

Configure

## Set the Github settings

Repos Pattern is `*`

OAuth.User and Admins are your Github user name

OAuth.Token is the personal access token

App.id is client ID from the OAuth registration in github

App.secret is client secret from the OAuth registration in github

Orgs is the Github organization

Note: To ensure you use the Github organization and not the GitHub user name. From a browser on which you are logged into github click on the silhouette of the cat icon. See below

The screenshot shows the GitHub homepage. At the top, there is a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. A red circle highlights the user icon in the top-left corner. Below the header, a banner reads "Learn Git and GitHub without any code!" with a "Read the guide" button in green and a "Start a project" button in white. In the main content area, a modal window appears with the title "You've been added to the giphos organization!". It contains tips for organization members, such as switching contexts between personal and organization accounts. To the right of the modal, a notification bubble says "Use any theme with GitHub Pages". It informs users that they can now use any of the hundreds of community-curated themes on GitHub.com to change the look and feel of their GitHub Pages site. Below these notifications, there is a section titled "Repositories you contribute to" with two entries: "DAVEXACOM/IBMGraphicalDat..." and "DAVEXACOM/WTXMapToIBMG...". A red circle highlights the second repository entry. At the bottom of the page, there is a section titled "Discover interesting projects and people to populate".

The screenshot shows the GitHub OAuth Apps page. On the left, there is a sidebar with three options: "OAuth Apps" (which is selected and highlighted with an orange border), "GitHub Apps", and "Personal access tokens". The main content area displays a single application card for "icp-msb-iib-demo". The card shows the application name, a "Client ID" (e681fb9c9e6491de43587), and a "Client Secret" (aef2f82c12acabd3218de28d7f63b1fa25560096). Below the card, it says "Davexa owns this application." and "You can list your application in the GitHub Marketplace so that other users can discover it." At the bottom, it shows "1 user".

## Configuration

Microservice Builder Pipeline Edit these parameters for configuration

### Release name ?

ibrn-msb-pipeline-da

### Target Namespace ?

default

I have read and agreed to the [license agreements](#)

## GitHub

### Url

https://github.com

### Name

GitHub

### Orgs

DAVEXACOM

### RepoPattern

.\*

### OAuth.Token

23a8537c82a19a358fb957bbfa7fba3d369eeb6b

### OAuth.User

Davexa

### App.Id

e681fbc9e6491de43587

### App.Secret

aef2f82c12acabd3218de28d7f63b1fa25560096

### Admins

Davexa

## Pipeline Settings

Registry.URL is icpcluster.icp:8500/default

### Pipeline

#### Build

true

#### Deploy

true

#### Test

true

#### Debug

false

#### DeployBranch

master

#### Registry.Url

icpcluster.icp:8500/default

#### Registry.Secret

admin.registrykey

#### TargetNamespace

Enter value

#### Template.RepositoryUrl

https://github.com/WASdev/microservicebuilder.lib.git

#### Template.Version

2.0.0

#### ChartFolder

chart

#### ManifestFolder

manifests

#### LibertyLicenseJar.BaseUrl

Enter value

#### LibertyLicenseJar.Name

wlp-core-license.jar

## Master Settings

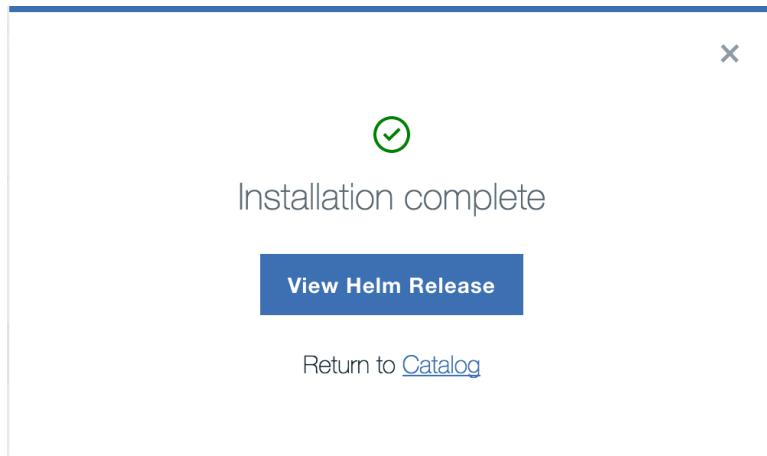
nodePort must be a port that is available, try 32050, 32051 etc

If you pick a port that is in use. You will be informed with a message top of screen when you attempt the install (see below). The Helm release will have failed but the release of the name you provided will have been created. You'll need to open a second browser tab go and delete it via ICP->Workloads->Helm releases before changing the nodePort number and trying the install again.

<b>Master</b>	
<b>Name</b>	<b>Image</b>
jenkins-master	ibmcom/mb-jenkins
<b>ImageTag</b>	<b>ImagePullPolicy</b>
2.0.0	Always
<b>ImagePullSecret</b>	<b>Component</b>
Enter value	jenkins-master
<b>Cpu</b>	<b>Memory</b>
200m	256Mi
<b>JavaOpts</b>	<b>ServicePort</b>
-Xmx512m -Dfile.encoding=UTF-8 -Dhudson.security.ArtifactsPermission=true	8080
<b>ServiceType</b>	<b>NodePort</b>
NodePort	32051
<b>HostName</b>	<b>ContainerPort</b>
Enter value	8080

Although we check on a persistent volume for Jenkins we don't have to use it. I haven't used one in this example.

Leave everything else to default and hit install.



Check the Helm release to ensure it worked. If you chose a port that was already in use remember the Helm release will have failed but the release of the name you provided will have been created. You'll need to open a second browser tab go and delete it via ICP->Workloads->Helm releases before changing the nodePort number and trying the install again.

Important! Now go back to GitHub OAuth and ensure you have the correct the port number 32051 (or whatever it turned out to be in this example) as we took a punt on the port being available – see below.

https://github.com/settings/applications/635182

Search

Settings / Developer settings

OAuth Apps    GitHub Apps    Personal access tokens

icp-msb-iib-demo

Davexa owns this application.

Transfer ownership

You can list your application in the [GitHub Marketplace](#) so that other users can discover it.

List this application in the Marketplace

1 user

Client ID  
e681fbc9e6491de43587

Client Secret  
aef2f82c12acabd3218de28d7f63b1fa25560096

Revoke all user tokens    Reset client secret

Application logo

Drag & drop    Upload new logo

You can also drag and drop a picture from your computer.

Application name  
icp-msb-lib-demo

Something users will recognize and trust

Homepage URL  
http://172.23.49.19:32051

The full URL to your application homepage

Application description

Application description is optional

This is displayed to all users of your application

Authorization callback URL  
http://172.23.49.19:32051/securityRealm/finishLogin

Your application's callback URL. Read our [OAuth documentation](#) for more information.

Note: Port number appears in 2 places. Homepage URL and Authorization call back

Then hit the update button.

Your Github port will now match the icp msb pipeline parameters.

Click on View Helm Release

## Review the microservice builder pipeline helm release

### Helm releases

NAME	NAMESPACE	STATUS	UPDATED
blockchain-network	default	DEPLOYED	Nov 28th 2017
cam	default	DEPLOYED	Nov 27th 2017
cardashboard-msb-pipeline	default	DEPLOYED	Dec 7th 2017
daiibmq	default	DEPLOYED	Dec 12th 2017
db2server1	default	DEPLOYED	Nov 27th 2017
ibm-msb-pipeline-da	default	DEPLOYED	Dec 13th 2017

Select ibm-msb-pipeline-da

ibm-msb-pipeline-da

DETAILS					
TYPE	DETAIL				
Name:	ibm-msb-pipeline-da				
Namespace:	default				
Status:	DEPLOYED				
Chart name:	ibm-microservicebuilder-pipeline				
Chart version:	2.0.2				
updated:	Dec 13th 2017 at 2:42 PM				
Type:	Release				

DEPLOYMENT					
NAME		DESIRED	CURRENT	UP-TO-DATE	AVAILABLE
ibm-msb-pipeline-da-ibm-microservicebuilder-pipeline		1	1	1	1

Select the Deployment link

## ibm-msb-pipeline-da-ibm-microservicebuilder-pipeline

Overview    Events    Logs

Deployment details		ReplicaSets	
Type	Detail	Type	Detail
<b>Name</b>	ibm-msb-pipeline-da-ibm-microservicebuilder-pipeline	-	-
<b>Namespace</b>	default	-	-
<b>Creation time</b>	Jan 5th 2018 at 1:13 PM	-	-
<b>Labels</b>	app=ibm-msb-pipeline-da-ibm-microservicebuilder-pipeline,chart=ibm-microservicebuilder-pipeline-2.1.0,component=ibm-msb-pipeline-da-jenkins-master,heritage=Tiller,release=ibm-msb-pipeline-da	-	-
<b>Selector</b>	component=ibm-msb-pipeline-da-jenkins-master	-	-
<b>Replicas</b>	1 desired   1 total   1 updated   1 available	-	-
<b>RollingUpdateStrategy</b>	1 max unavailable, 1 max surge	-	-
<b>MinReadySeconds</b>	0	-	-

## Authorize the ibm-msb-pipeline-da service to access github

Click on access http link

Note: you may not get this Authorize if you have previously Authorized Davexa (in this case) to access the organization repository for OAuth

[https://github.com/login/oauth/authorize?client\\_id=e681fbc9e6491de43587&scope=read:org,user:email](https://github.com/login/oauth/authorize?client_id=e681fbc9e6491de43587&scope=read:org,user:email)

Authorize icp-msb-iib-demo

icp-msb-iib-demo by **Davexa**  
wants to access your Davexa account

**Personal user data**  
Email addresses (read-only)

**Organizations and teams**  
Read-only access

**Organization access**

- DAVEXACOM ✓
- giphos ✓

**Authorize Davexa**

Authorizing will redirect to  
<http://172.23.49.19: 3205>

Not owned or operated by GitHub    Created day ago    Fewer than 10 GitHub users

If when you hit the agree Authorize Davexa you get the following

172.23.49.19:32037/securityRealm/finishLogin?code=68ad5349244bce5b3b88

# Error

Watch out for the port 32037 mismatch to the 32051! You'll need to go back and ensure the Git and ICP MSB pipeline have matched ports as described previously

If you have your ports correctly matched you should get the Jenkins console.

Jenkins

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

Credentials

Build Queue

No builds in the queue.

Build Executor Status

No builds in the queue.

S	W	Name ↓	Last Success
		DAVEXACOM	15 min - <a href="#">log</a>

Icon: S M L

## IBM Cloud Private – Jenkins

### Checking Jenkins connection to Github

Click on the organization name DAVEXACOM in this example

If you get a screen like below it means your repository is not structured as Jenkins expects. i.e. it does not have a Jenkinsfile perhaps. Or you have a credentials problem.

DAVEXACOM

This folder is empty

There are no repositories found that contain buildable projects. Organization Folders automatically build and manage repositories that contain recognizable projects.

Pipeline Multibranch projects recognize and build repositories with a file named Jenkinsfile in branches of the repository. This file should contain a valid Jenkins Pipeline. See also: [Creating Multibranch Projects](#).

Please [configure Project Sources](#) to get started or [re-run the Folder Computation](#) if the repositories have since been configured.

Up

Status

Configure

Scan Organization Now

Scan Organization Log

Organization Events

Delete Organization

People

Build History

GitHub

Pipeline Syntax

Credentials

Build Queue

No builds in the queue.

Build Executor Status

No builds in the queue.

If you get the above you should check the scan organization log. **If it's a credentials problem and you can't work out why... then you can go into Configure.**

The screenshot shows the Jenkins configuration interface for the 'DAVEXACOM' organization. On the left, there's a sidebar with various links like Up, Status, Configure, Scan Organization Now, etc. The main area has fields for Name (DAVEXACOM), Display Name, and Description. Under the 'Projects' section, it shows GitHub Organization with API endpoint set to GitHub and a dropdown for Credentials containing 'Davexa/\*\*\*\*\*\*'. A note explains that only 'username with password' credentials are supported. Below this, the Owner is listed as DAVEXACOM. The 'Build Queue' and 'Build Executor Status' sections are shown below.

hit Add next to Credentials and select the organization

This screenshot shows the 'Add Credentials' dialog for the 'DAVEXACOM' provider. It includes fields for Domain (Global credentials (unrestricted)), Kind (Username with password), Username (Davexa), Password (\*\*\*\*\*), and ID/Description. At the bottom are 'Add' and 'Cancel' buttons. To the right, a preview window shows the current credentials list with 'DAVEXACOM' and 'Jenkins' entries.

Add these credentials and make sure they are selected in the Configure screen

If all is correct it will look something like this.

The screenshot shows the Jenkins interface for the 'DAVEXACOM' organization. On the left is a sidebar with various navigation links. The main area displays a single repository named 'IIB-MQ' with a brief description: 'IIBv10 and MQv9 docker build repository'. The repository has a status icon (yellow sun) and a build history icon.

**Repositories (1)**

S	W	Name ↓	Description
		IIB-MQ	IIBv10 and MQv9 docker build repository

Icon: [S](#) [M](#) [L](#)

**Build Queue**

## Before Starting the Jenkins build process

You may want to change the values.yaml file in the GitHub repository in order to customize the Queue Manager Name, IIB Node name etc that will be created in the runtime container by the Jenkins build.

Branch: master ▾ IIB-MQ / chart / iibmq / values.yaml

 Davexa Add files via upload

1 contributor

57 lines (50 sloc) | 1.97 KB

Raw Blame

```
1 license: "accept"
2
3 image:
4   # repository is the container repository to use, which must contain IBM MQ Advanced for Developers
5   repository: icpcluster.icp:8500/default/iib-mq
6   # tag is the tag to use for the container repository
7   tag: latest
8   # pullSecret is the secret to use when pulling the image from a private registry
9   pullSecret:
10  # pullPolicy is either IfNotPresent or Always (https://kubernetes.io/docs/concepts/containers/images/)
11  pullPolicy: IfNotPresent
12
13 # persistence section specifies persistence settings which apply to the whole chart
14 persistence:
15   # enabled is whether to use Persistent Volumes or not
16   enabled: false
17   # useDynamicProvisioning is whether or not to use Storage Classes to dynamically create Persistent Volumes
18   useDynamicProvisioning: true
19
20 # dataPVC section specifies settings for the main Persistent Volume Claim, which is used for data in /var/mqm
21 dataPVC:
22   # name sets part of the name for this Persistent Volume Claim
23   name: "data"
24   ## storageClassName is the name of the Storage Class to use, or an empty string for no Storage Class
25   storageClassName: ""
26   ## size is the minimum size of the Persistent Volume
27   size: 2Gi
28
29 service:
30   name: iib-mq
31   type: NodePort
32
33 resources:
34   limits:
35     cpu: 1024m
36     memory: 1024Mi
37   requests:
38     cpu: 1024m
39     memory: 1024Mi
40
41 # queueManager section specifies settings for the MQ Queue Manager
42 queueManager:
43   # name allows you to specify the name to use for the queue manager. Defaults to the Helm release name.
44   name: icpqm1
```

I changed the queueManager->name to icpDA1 for example

## Starting the Jenkins build process

With no Github webhook in place to publish to Jenkins to tell it to start a build we will use the Scan Organization Now button.

172.23.49.19:32051/job/DAVEXACOM/job/IIB-MQ/job/master/ Jenkins

**Branch master**

Full project name: DAVEXACOM/IIB-MQ/master

**Stage View**

Extract	Docker Build
10s	2min 16s
10s	2min 16s

**Build History**

- #1 Dec 14, 2017 3:07 AM

**Permalinks**

- [Last build \(#1\), 2 min 29 sec ago](#)

## Following the Jenkins build logs

Click on the #1 in Build history

172.23.49.19:32051/job/DAVEXACOM/job/IIB-MQ/job/master/#1 Jenkins

**Build #1 (Dec 14, 2017 3:07:33 AM)**

**Branch indexing**

**Revision:** df5cdbbb8161b8322511991bcb1f207217642461  
• 2.0.0

**git** **Revision:** 1e342b41aab2b10921d332c75d4dc1a895aec793  
• master

**Console Output**

**Back to Project**

**Status**

**Changes**

**Console Output**

**Edit Build Information**

**Git Build Data**

**No Tags**

**Git Build Data**

**Thread Dump**

**Pause/resume**

**Replay**

**Pipeline Steps**

**Embeddable Build Status**

Click on Console Output

Jenkins > DAVEXACOM > IIB-MQ > master > #1

```

172.23.49.19:32051/job/DAVEXACOM/job/IIB-MQ/job/master/1/console

MQSeriesRuntime-9.0.3-0 ##### MQSeriesServer-9.0.3-0 #####
Updated PAM configuration in /etc/pam.d/lbmq
[91m
WARNING: System settings for this system do not meet recommendations for this product
See the log file at "/tmp/mqconfig.225.log" for more information

[0mMQSeriesJava-9.0.3-0 ##### MQSeriesJRE-9.0.3-0 #####
MQSeriesWeb-9.0.3-0 ##### MQSeriesWeb-9.0.3-0 #####
MQSeriesMsg_2b_CN-9.0.3-0 ##### MQSeriesMsg_2b_TW-9.0.3-0 #####
MQSeriesMsg_cS-9.0.3-0 ##### MQSeriesMsg_dG-9.0.3-0 #####
MQSeriesMsg_eG-9.0.3-0 ##### MQSeriesMsg_fR-9.0.3-0 #####
MQSeriesMsg_hU-9.0.3-0 ##### MQSeriesMsg_iT-9.0.3-0 #####
MQSeriesMsg_jA-9.0.3-0 ##### MQSeriesMsg_kO-9.0.3-0 #####
MQSeriesMsg_pL-9.0.3-0 ##### MQSeriesMsg_pT-9.0.3-0 #####
MQSeriesMsg_rU-9.0.3-0 ##### MQSeriesGSKit-9.0.3-0 #####
[91m136 of 136 tasks have been completed successfully.
'Installation1' [/opt/mqm] set as the primary installation.
[0m --> 56e79efdf609a
Removing intermediate container 31196718b202
Step 9/30 : COPY mq-dev-config.sh mq-license-check.sh mq.sh setup-mqm-web.sh setup-var-mqm.sh /usr/local/bin/
--> fbb8712f93d3
Removing intermediate container 9ff23fbab5ee
Step 10/30 : COPY *.mgsc /etc/mqm/
--> c675fadefb62
Removing intermediate container 204321c1a468
Step 11/30 : COPY admin.json /etc/mqm/
--> 16f92e78ec1a
Removing intermediate container 5e5cdceeb1bb
Step 12/30 : COPY mq-dev-config /etc/mqm/mq-dev-config
--> 5257fe8587f6
Removing intermediate container 6c629260af94
Step 13/30 : RUN chmod +x /usr/local/bin/*
--> Running in 2524e9d6de87
--> c503053827f0
Removing intermediate container 2524e9d6de87
Step 14/30 : RUN rm -rf /opt/ibm && curl https://public.dhe.ibm.com/ibmdl/export/pub/software/websphere/integration/10.0.0.10-IIB-LINUX64-DEVELOPER.tar.gz
iib-10.0.0.10/tools --directory /opt/ibm && /opt/ibm/iib-10.0.0.10/iib make registry global accept license silently
--> Running in b3b163d71998

```

---

Jenkins > DAVEXACOM > IIB-MQ > master > #3

```

290911d: digest: sha256:8498fe62207bf9ec248509b0457680fd43c4fe9f871c9c69ab4fe1373085f948 size: 5735
[Pipeline]
[Pipeline] // container
[Pipeline]
[Pipeline] // stage
[Pipeline] fileExists
[Pipeline] echo
User defined chart location specified: chart/iibmq
[Pipeline] sh
[DAVEXACOM_IIB-MQ_master-MPOWJ5QPOSBOGPPW2KNKIZRPJQRXLX5DN7HQYK7RV5DPBNS2U3S2A] Running shell script
+ echo image:
repository: icpcluster.icp:8500/default/iib10mq9
tag: "290911d"
[Pipeline] fileExists
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] fileExists
[Pipeline] container
[Pipeline] {
[Pipeline] sh
[DAVEXACOM_IIB-MQ_master-MPOWJ5QPOSBOGPPW2KNKIZRPJQRXLX5DN7HQYK7RV5DPBNS2U3S2A] Running shell script
+ /helm init --client-only --skip-refresh
Creating /home/jenkins/.helm
Creating /home/jenkins/.helm/repository
Creating /home/jenkins/.helm/repository/cache
Creating /home/jenkins/.helm/repository/local
Creating /home/jenkins/.helm/plugins
Creating /home/jenkins/.helm/starters
Creating /home/jenkins/.helm/cache/archive
Creating /home/jenkins/.helm/repository/repositories.yaml
$HELM_HOME has been configured at /home/jenkins/.helm.
Not installing tiller due to 'client-only' flag having been set
Happy Helming!
[Pipeline] sh
[DAVEXACOM_IIB-MQ_master-MPOWJ5QPOSBOGPPW2KNKIZRPJQRXLX5DN7HQYK7RV5DPBNS2U3S2A] Running shell script
+ /helm upgrade --install --wait --values pipeline.yaml --namespace default iib10mq9 chart/iibmq
Release "iib10mq9" has been upgraded. Happy Helming!
LAST DEPLOYED: Thu Dec 14 04:36:30 2017
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
=> v1/Secret
NAME          TYPE      DATA  AGE
iib10mq9-iib-mq Opaque  2     10m

=> v1/Service
NAME          CLUSTER-IP  EXTERNAL-IP  PORT(S)           AGE
iib10mq9-iib-mq  10.0.0.147  <nodes>   1414:30231/TCP,9443:31826/TCP,4414:30212/TCP,7080:30505/TCP,7800:30395/TCP  10m

=> v1beta1/StatefulSet
NAME          DESIRED  CURRENT  AGE
iib10mq9-iib-mq  1        1        10m

```

```
$HELM_HOME has been configured at /home/jenkins/.helm.
Not installing tiller due to 'client-only' flag having been set
Happy Helming!
[Pipeline] fileExists
[Pipeline] sh
[DAVEXACOM_IIB-MQ_master-MPOWJ5QPOSBOGPPW2KNKIZRPJQRXLX5DN7HQYK7RV5DPBNS2U3S2A] Running shell script
+ /helm upgrade --install --wait --values pipeline.yaml --namespace default iibl0mq9 chart/iibmq
Release "iibl0mq9" has been upgraded. Happy Helming!
LAST DEPLOYED: Thu Dec 14 04:36:30 2017
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/Secret
NAME          TYPE      DATA  AGE
iibl0mq9-iib-mq  Opaque   2    10m

==> v1/Service
NAME            CLUSTER-IP  EXTERNAL-IP  PORT(S)           AGE
iibl0mq9-iib-mq  10.0.0.147  <nodes>     1414:30231/TCP,9443:31826/TCP,4414:30212/TCP,7080:30505/TCP,7800:30395/TCP  10m

==> v1beta1/StatefulSet
NAME            DESIRED  CURRENT  AGE
iibl0mq9-iib-mq  1        1        10m

[Pipeline] }
[Pipeline] // container
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // podTemplate
[Pipeline] End of Pipeline

Could not update commit status. Message: Server returned HTTP response code: 201, message: 'Created' for URL: https://api.github.com/repos/DAVEXACOM/IIB-MQ/statuses/290911dc47be82503419de524a4caae5d903e037

Finished: SUCCESS
```

## Checking ICP Image repository

The screenshot shows the IBM Cloud Private console interface. At the top, there is a header bar with a back arrow, a warning icon, and the URL <https://172.23.49.17:8443/console/platform/images>. Below the header, a navigation bar includes a 'Docs' link and the 'IBM Cloud Private' logo. The main content area is titled 'Images'. It features a search bar labeled 'Search items' with a magnifying glass icon. Below the search bar, there is a pagination control showing '20 ▾ items per page | 1-20 of 26 items'. The main list is titled 'NAME ▾' and contains the following items:

- [default/agentless-crawler](#)
- [default/cam-broker](#)
- [default/cam-busybox](#)
- [default/cam-mongo](#)
- [default/cam-orchestration](#)
- [default/cam-portal-api](#)
- [default/cam-portal-ui](#)
- [default/cam-redis](#)
- [default/cam-service-composer-api](#)
- [default/cam-service-composer-ui](#)
- [default/cardashboard](#)
- [default/icpmsbdemo](#)
- [default/iib-mq](#)
- [default/iib10mq9](#)

Images / default/iib10mq9 /

# default/iib10mq9

## Overview

### Image details

TYPE	DETAIL
Name	default/iib10mq9
Owner	default
Scope	global
Tags	

## Checking ICP Helm Release

### Helm releases

20 ▾ Items per page   1-13 of 13 items				
NAME	NAMESPACE	STATUS	UPDATED	CHART NAME
<a href="#">blockchain-network</a>	default	DEPLOYED	Nov 28th 2017	ibm-blockchain-network
<a href="#">cam</a>	default	DEPLOYED	Nov 27th 2017	ibm-cam-prod
<a href="#">cardashboard-msb-pipeline</a>	default	DEPLOYED	Dec 7th 2017	cardashboard
<a href="#">daiibmq</a>	default	DEPLOYED	Dec 12th 2017	iib-mq
<a href="#">db2server1</a>	default	DEPLOYED	Nov 27th 2017	ibm-db2oltp-dev
<a href="#">ibm-msb-pipeline-da</a>	default	DEPLOYED	Dec 13th 2017	ibm-microservicebuilder-pipeline
<a href="#">ibm-msb-pipeline</a>	default	DEPLOYED	Dec 6th 2017	ibm-microservicebuilder-pipeline
<a href="#">ibm-msbf</a>	default	DEPLOYED	Nov 28th 2017	ibm-microservicebuilder-fabric
<a href="#">icp-dp7</a>	default	DEPLOYED	Dec 4th 2017	ibm-datapower-dev
<a href="#">icpmbsbdemo</a>	default	DEPLOYED	Dec 6th 2017	icpmbsbdemo
<a href="#">iib-icp-100010</a>	default	DEPLOYED	Dec 4th 2017	ibm-integration-bus-dev
<a href="#">iib10mq9</a>	default	DEPLOYED	Dec 14th 2017	iib-mq

## iib10mq9

DETAILS	
TYPE	DETAIL
Name:	iib10mq9
Namespace:	default
Status:	DEPLOYED
Chart name:	iib-mq
Chart version:	1.1.0
updated:	Dec 14th 2017 at 3:36 PM
Type:	Release

SERVICE			
NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)
iib10mq9-iib-mq	10.0.0.147	<nodes>	1414:30231/TCP,9443:31826/TCP,4414:30212/TCP,7080:30505/TCP,7800:30395/TCP

STATEFULSET		
NAME	DESIRED	CURRENT
iib10mq9-iib-mq	1	1

## iib10mq9

DETAILS	
TYPE	DETAIL
Name:	iib10mq9
Namespace:	default
Status:	DEPLOYED
Chart name:	iib-mq
Chart version:	1.1.0
updated:	Dec 14th 2017 at 3:36 PM
Type:	Release

SERVICE			
NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)
iib10mq9-iib-mq	10.0.0.147	<nodes>	1414:30231/TCP,9443:31826/TCP,4414:30212/TCP,7080:30505/TCP,7800:30395/TCP

STATEFULSET		
NAME	DESIRED	CURRENT
iib10mq9-iib-mq	1	1

StatefulSet / iib10mq9-iib-mq / iib10mq9-lib-mq-0 /

## iib10mq9-iib-mq-0

Overview Containers Events Logs

```
Starting syslog
-----
Configuring db access
BIP8071I: Successful command completion.
Starting node icpnode1
Starting Switch Server
BIP8096I: Successful command initiation, check the system log to ensure that the component started without
-----
Server mqweb started with process ID 459.
S starting Switch
Starting libswitch, please wait...
iibswitch started.
BIP8071I: Successful command completion.
BIP8071I: Successful command completion.
BIP8096I: Successful command initiation, check the system log to ensure that the component started without
QMNAME(icpDA1)                                     STATUS(Running)
IBM MQ Queue Manager icpDA1 is now fully running
```

## Connect MQ Console

ICP->Helm Releases->iib10mq9-iib-mq->services

SERVICE			
NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)
iib10mq9-iib-mq	10.0.0.147	<nodes>	1414:30231/TCP,9443:31826/TCP,4414:30212/TCP,7080:30505/TCP,7800:30395/TCP

Services / iib10mq9-iib-mq /

## iib10mq9-iib-mq

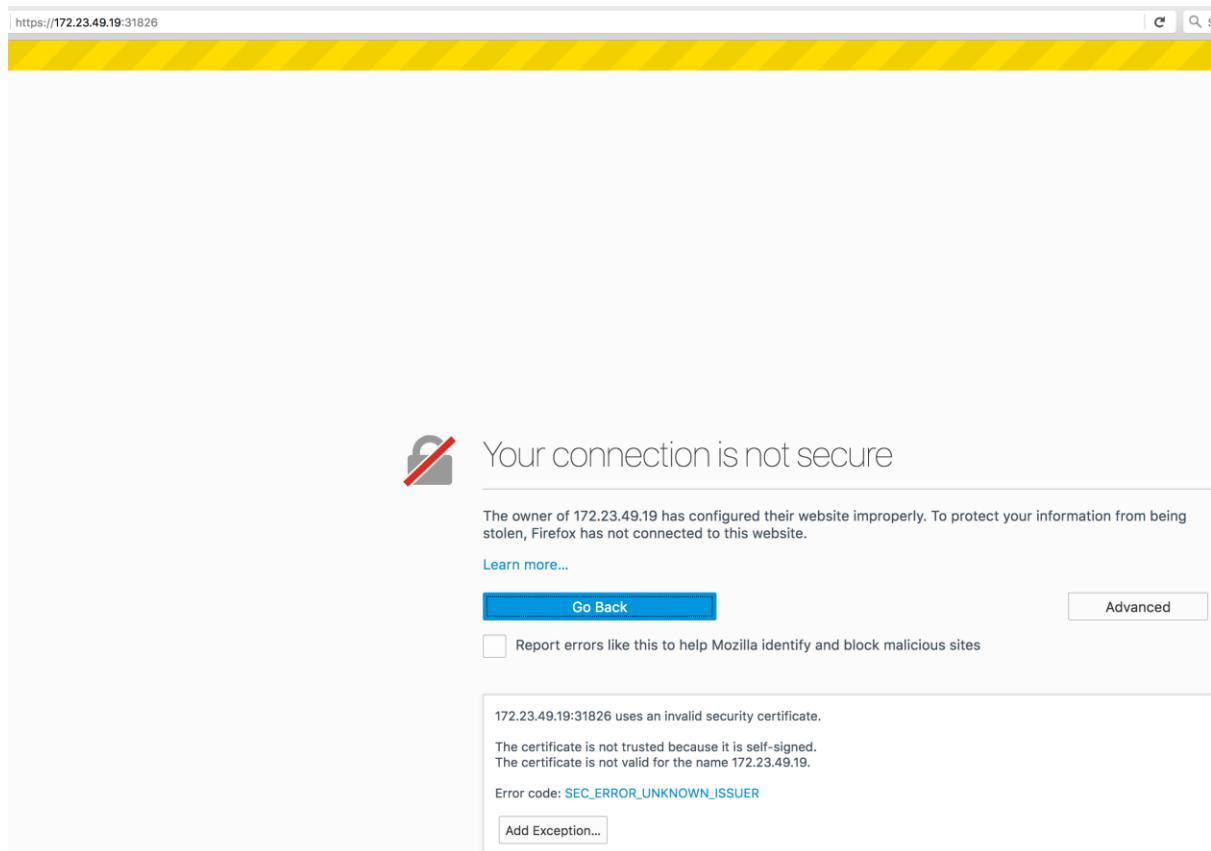
Overview

Service details	
TYPE	DETAIL
Name	iib10mq9-iib-mq
Namespace	default
Creation time	Dec 14th 2017 at 3:26 PM
Type	NodePort
Labels	app=iib10mq9-lib-mq,chart=iib-mq-1.1.0,heritage=Tiller,release=iib10mq9
Selector	app=iib10mq9-lib-mq
IP	10.0.0.147
Port	iib-mq-server 1414/TCP; iib-mq-web 9443/TCP; iib-mq-console 4414/TCP; iib-mq-nodelistener 7080/TCP; iib-mq-serverlistener 7800/TCP
Node port	iib-mq-server 30231/TCP iib-mq-web 31826/TCP iib-mq-console 30212/TCP iib-mq-nodelistener 30505/TCP iib-mq-serverlistener 30395/TCP
Session affinity	None

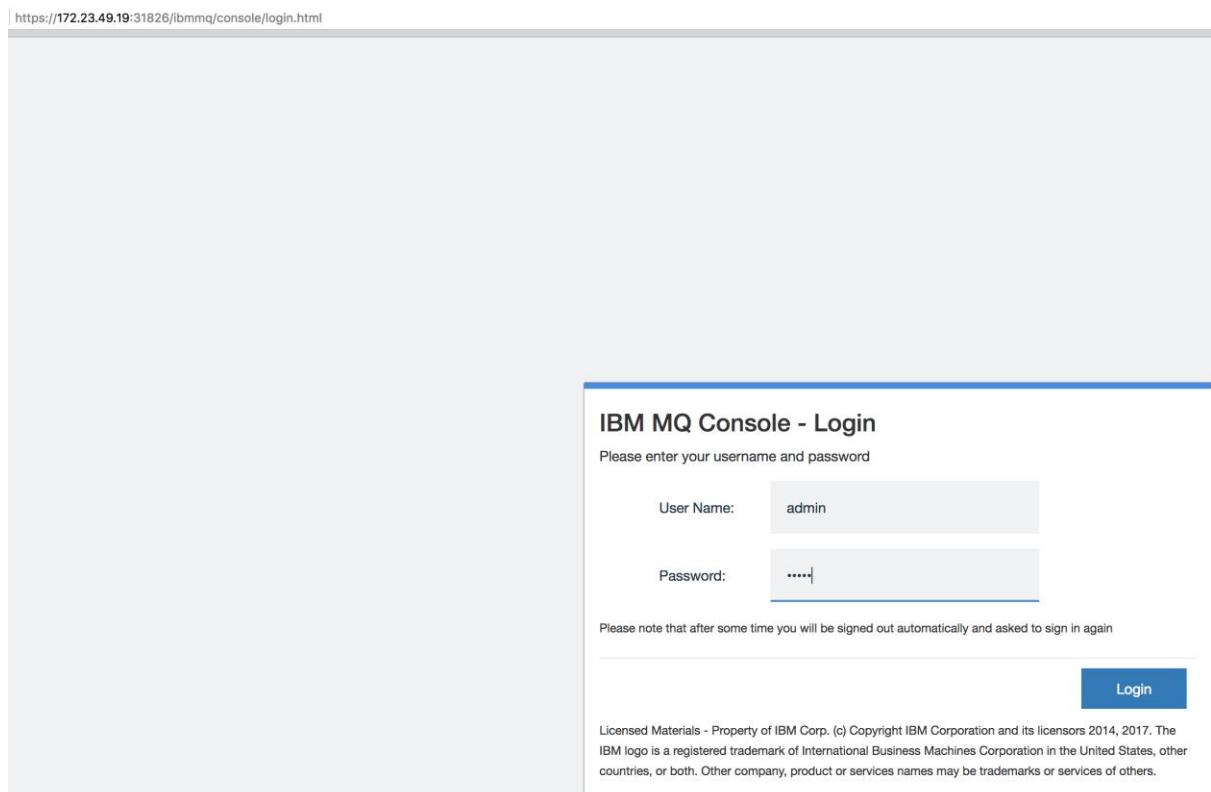
Select the iib-mq-web 31826/TCP link.

In the browser add the https:// up front

Add certificate



Log into MQ console with admin/passw0rd



https://172.23.49.19:31826/ibmmq/console/

## IBM MQ Console

IBM MQ Container +

### Queue Manager

Name	Status
icpDA1	<span style="color: green;">▲</span> Running

Total: 1 Selected: 0 Updated: 3:49:39 PM

### Queues on icpDA1

Name	Queue type	Queue depth
DEV.DEAD.LETTER.QUEUE	Local	0
DEV.QUEUE.1	Local	0
DEV.QUEUE.2	Local	0
DEV.QUEUE.3	Local	0

## Connect to IIB Web UI

Services / iib10mq9-iib-mq /

iib10mq9-iib-mq

[Overview](#)

### Service details

TYPE	DETAIL
Name	iib10mq9-iib-mq
Namespace	default
Creation time	Dec 14th 2017 at 3:26 PM
Type	NodePort
Labels	app=iib10mq9-iib-mq,chart=iib-mq-1.1.0,heritage=Tiller,release=iib10mq9
Selector	app=iib10mq9-iib-mq
IP	10.0.0.147
Port	iib-mq-server 1414/TCP; iib-mq-web 9443/TCP; iib-mq-console 4414/TCP; iib-mq-nodelistener 7080/TCP; iib-mq-serverlistener 7800/TCP
Node port	iib-mq-server 30231/TCP iib-mq-web 31826/TCP <b>iib-mq-console 30212/TCP</b> iib-mq-nodelistener 30505/TCP iib-mq-serverlistener 30395/TCP

**Integration**

Filter Options...

node1 ▾  
| Servers ▾  
| Operational Policy  
| Data  
| Security  
| Monitoring  
| Business

## icpnode1 - Integration Node

Overview Statistics

▼ Quick View

Node Name	icpnode1
Version	100010
Admin Security	Off
Run Mode	running
Short Description	
Long Description	

▼ Advanced Properties

Platform Name	Linux
Fixpack Capability	10.0.0.1
Operation Mode	developer
Platform Architecture	x86_64
Platform Version	3.10.0-514.16.1.el7.x86_64
Queue Manager	
Build Level	ib1000-L170911.419 (S1000-L170901.10502)
Admin Agent Process ID	1545

▼ Component Properties

Security Cache - Cache Sweep Interval	300
Security Cache - Cache Timeout	60
Cache Manager - Policy	disabled
Cache Manager - Port Range	2800-2819
Cache Manager - Listener Host	

## *Adding your own MQSC for MQ files and rebuilding with Jenkins*

### How it hangs together

The screenshot shows a GitHub repository page for 'DAVEXACOM / IIB-MQ'. The repository has 23 commits, 1 branch, and 0 issues. The description is 'IIBv10 and MQv9 docker build repository'. A specific commit by 'Davexa' titled 'Create dadefs.mqsc' is highlighted, showing changes to various files including configuration files and scripts related to the Docker build process.

The docker file copies all **.mqsc** in the github repository to a /mqm/etc in the runtime container. In our example that is 10-listener.mqsc and dadefs.mqsc

```
COPY mq-dev-config.sh mq-license-check.sh mq.sh setup-mqm-web.sh setup-var-mqm.sh /usr/local/bin/  
COPY *.mqsc /etc/mqm/  
COPY admin.json /etc/mqm/  
  
COPY mq-dev-config /etc/mqm/mq-dev-config
```

The docker file copies a bunch of **.sh** including **iib-manage.sh** to the /usr/local/bin

```
# Copy in script files  
COPY iib_manage.sh /usr/local/bin/  
COPY iib-license-check.sh /usr/local/bin/  
COPY iib_env.sh /usr/local/bin/
```

The docker file sets an **entry point** for the runtime container that is **iib-manage.sh**

```
127 # Set entrypoint to run management script  
128 ENTRYPOINT ["iib_manage.sh"]
```

**iib\_manage.sh** is executed when the container starts up

## Updating IIB-MQ GitHub Repo in DAVEXACOM Org

Create a new file called dadefs.mqsc or similar

The screenshot shows the GitHub repository page for 'DAVEXACOM / IIB-MQ'. At the top, there's a header with the repository name and a search bar. Below the header, a navigation bar has 'Code' selected. Underneath the navigation bar, it says 'IIBv10 and MQv9 docker build repository'. There's a 'Add topics' button. Below that, a summary bar shows '23 commits', '1 branch', and '0 releases'. At the bottom of the summary bar, there are buttons for 'Branch: master ▾', 'New pull request', and 'Create new file'.

Add your definitions and commit the changes.

The screenshot shows the 'Edit file' interface for the 'dadefs.mqsc' file. The file content is the Apache License 2.0 text, starting with a copyright notice and ending with a definition of the license terms. The code is as follows:

```
1 * © Copyright IBM Corporation 2015, 2017
2 *
3 *
4 * Licensed under the Apache License, Version 2.0 (the "License");
5 * you may not use this file except in compliance with the License.
6 * You may obtain a copy of the License at
7 *
8 * http://www.apache.org/licenses/LICENSE-2.0
9 *
10 * Unless required by applicable law or agreed to in writing, software
11 * distributed under the License is distributed on an "AS IS" BASIS,
12 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 * See the License for the specific language governing permissions and
14 * limitations under the License.
15 *
16 * Add a local queue
17 DEFINE QLOCAL(DAQ1) REPLACE
18
```

Re-run the ICP Microservices builder Jenkins pipeline

Select Scan Organization Now

The screenshot shows the Jenkins interface for the 'DAVEXACOM' organization. On the left, there's a sidebar with links: 'Up', 'Status', 'Configure', 'Scan Organization Now', 'Scan Organization Log', 'Organization Events', and 'Delete Organization'. The main area has a title 'DAVEXACOM' with a pink icon. Below it is a table titled 'Repositories (1)'. It contains one row with columns 'S', 'W', and 'Name ↓'. The 'Name' column shows 'IIB-MQ' with an orange sun icon. At the bottom, it says 'Icon: S M L'.

The build will start and you can select the #N build number and then you can follow the build on the console

The screenshot shows the Jenkins build console for the 'IIB-MQ' job under the 'master' branch, specifically build #4. The log output is as follows:

```
Step 26/30 : RUN chgrp mqbrkrs /home/iibuser/agentx.json && /switch.json && chmod +r /home/iibuser/agentx.json && chmod +rx /usr/local/bin/*.sh && chmod 666 /etc/hosts
--> Running in fb251bd670b7
--> daef3f905207
Removing intermediate container fb251bd670b7
Step 27/30 : ENV BASH_ENV /usr/local/bin/iib_env.sh
--> Running in ac0da88cc844
--> c86ac62eae9e
Removing intermediate container ac0da88cc844
```

### Check the results in the IBM MQ Console

You'll need to reconnect once the build has finished and the previous container replaced.

The screenshot shows the IBM MQ Console interface. At the top, there's a header bar with the title "IBM MQ Console". Below it, a navigation bar has "IBM MQ Container" selected. A blue plus icon is on the right of the navigation bar.

The main area contains two windows:

- Queue Manager**: This window lists a single queue manager named "icpDA1" which is "Running".
- Queues on icpDA1**: This window lists a single queue named "DAQ1" with a "Queue type" of "Local" and a "Queue depth" of "0".

At the bottom of the interface, there are status messages: "Total: 1 Selected: 0" and "Updated: 12:53:21 PM".

*Adding your own BAR files for IIB and rebuilding with Jenkins*

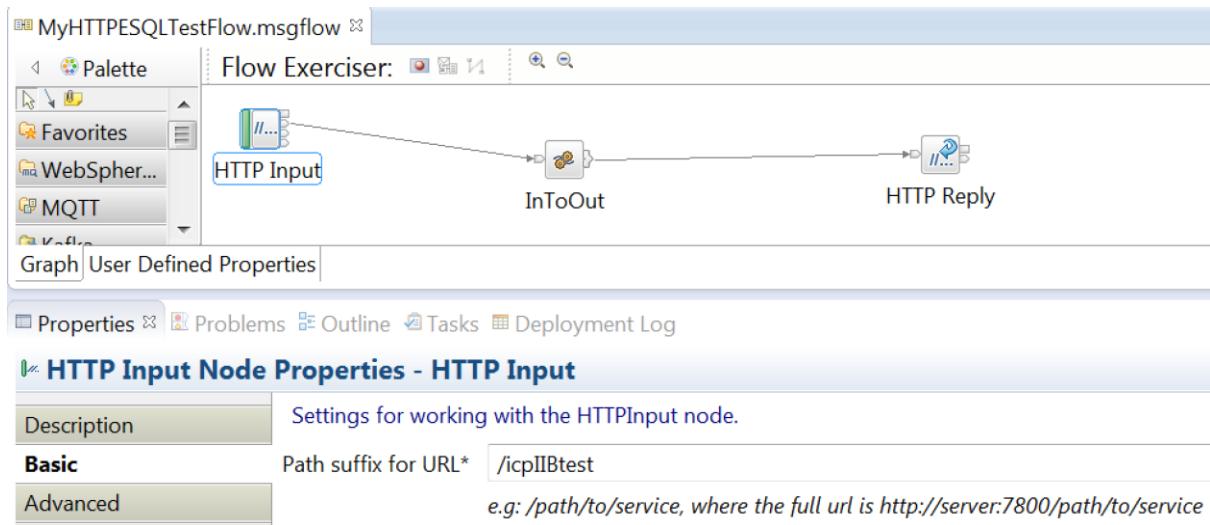
### How it hangs together

Basically, we'll use the same approach with the following changes

1. Instead of creating a file we'll import a BAR file to the repository from a local IIB toolkit workspace.
2. Change the dockerfile to copy all **\*.bar** file to mqm/etc along with the mqsc files
3. Change iib\_manage.sh to execute the mqsideploy command

### Create a BAR file in an IIB Toolkit

In this example Message flow MyHTTPESQLTestFlow is in MyICPTestApp IIB application



Compute Node InToOut references the following ESQL

```

CREATE COMPUTE MODULE MyHTTPPESQLTestFlow_InToOut
    CREATE FUNCTION Main() RETURNS BOOLEAN
    BEGIN
        CALL CopyMessageHeaders();

        DECLARE mystring CHARACTER;

        SET mystring='hello world';
        SET OutputRoot.BLOB.BLOB=CAST (mystring AS BLOB CCSID 1208);

        RETURN TRUE;
    END;

```

File->New-Broker Archive (BAR) file

<b>Create a new BAR file</b>	
Create a new BAR file resource	
Container:	<input type="text" value="BARfiles"/>
Folder:	<input type="text" value="&lt;default&gt;"/>
Name:	<input type="text" value="ICPDeploy"/>

Select MyICPTestApp and hit Build and Save

MyHTTPESQLTestFlow.msgflow \*ICPDeploy.bar

## Prepare

### Select deployable resources to include in the BAR

#### Deployable Resources

Select an application to package all its contained resources. Resources within an application are isolated from other applications.

Applications, shared libraries, services, and REST APIs  Message flows, static libraries and other message flow dependencies

Type filter text:

- Applications
- MyICPTestApp

**Build and Save...**

### Adding to BAR File

Operation completed successfully.

## Upload the BAR file to the IIB-MQ repo on GitHub

DAVEXACOM / IIB-MQ

Unwatch 2

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

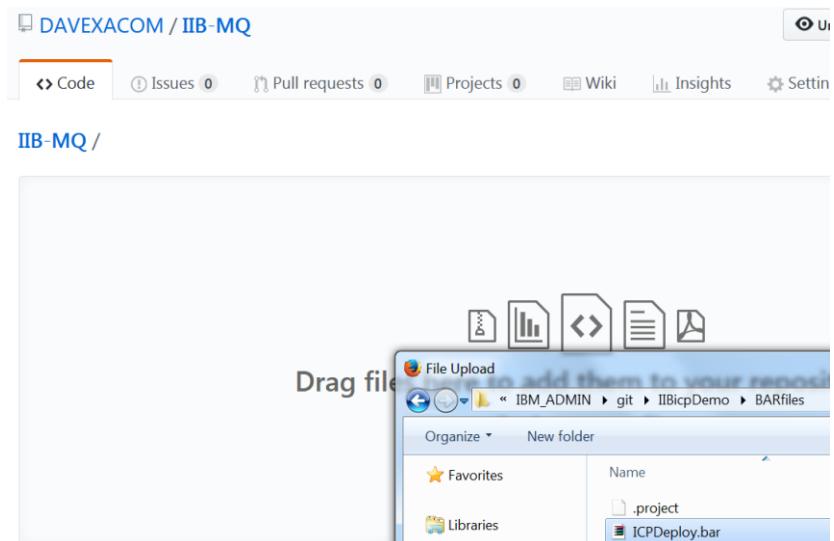
IIBv10 and MQv9 docker build repository

Add topics

23 commits 1 branch 0 releases

Branch: master New pull request Create new file Upload files

Navigate to your BAR file location in the IIB workspace on the local file system



Upload and commit changes

ICPDeploy.bar

### Commit changes

Add files via upload

Add an optional extended description...

Commit directly to the `master` branch.

Create a **new branch** for this commit and start

**Commit changes** **Cancel**

IIBv10 and MQv9 docker build repository

Add topics

24 commits 1 branch

Branch: master New pull request

Davexa Add files via upload

File	Description
chart/iibmq	Update service.yaml
.cignore	Initial Commit
.gitignore	Initial Commit
10-listener.mqsc	Initial Commit
CLA.md	Initial Commit
Dockerfile	Changed to IIB 10.0.0.10
ICPDeploy.bar	Add files via upload

## Edit the dockerfile on GitHub

```
67
68     COPY mq-dev-config.sh mq-license-check.sh mq.sh setup-mqm-web.sh setup-var-mqm.sh /usr/local/bin/
69     COPY *.mqsc /etc/mqm/
70     COPY *.bar /etc/mqm/
71     COPY admin.json /etc/mqm/
72
73     COPY mq-dev-config /etc/mqm/mq-dev-config
74
75     IN chmod +x /usr/local/bin/*.sh
--
```

### Select commit

**Commit changes**

## Edit the iib\_manage.sh file on GitHub

Add the following line.

```
mqsideploy $NODE_NAME -e $EXEC_NAME -a /etc/mqm/ICPDeploy.bar -m
```

```

140
141     fi
142     mqsichangeproperties $NODE_NAME -e $EXEC_NAME -o ComIbmIIBSwitchManager
143
144     mqsistop $NODE_NAME
145     mqsistart $NODE_NAME
146
147     mqsideploy $NODE_NAME -e $EXEC_NAME -a /etc/mqm/ICPDeploy.bar -m
148
149 }
150
151 monitor()

```

Select commit

**Commit changes**

Re-run the ICP Microservices builder Jenkins pipeline

Select Scan Organization Now

S	W	Name ↓
		IIB-MQ

Icon: [S](#) [M](#) [L](#)

The build will start and you can select the #N build number and then you can follow the build on the console

```

Step 26/30 : RUN chgrp mqbrkrs /home/iibuser/agentx.json && chmod +r /home/iibuser/agentx.json && chmod +rx /usr/local/bin/*.sh && chmod 666 /etc/hosts
---> Running in fb251bd670b7
---> daef3f905207
Removing intermediate container fb251bd670b7
Step 27/30 : ENV BASH_ENV /usr/local/bin/iib_env.sh
---> Running in ac0da88cc844
---> c86ac62eae9e
Removing intermediate container ac0da88cc844

```

## Check the results in the IIB Web UI

From ICP console -> Workloads -> Helm releases select iib10mq9 then select Services to get to the link to the Web UI (iib-mq-web 31826 in this example)

Services / [iib10mq9-iib-mq](#) /

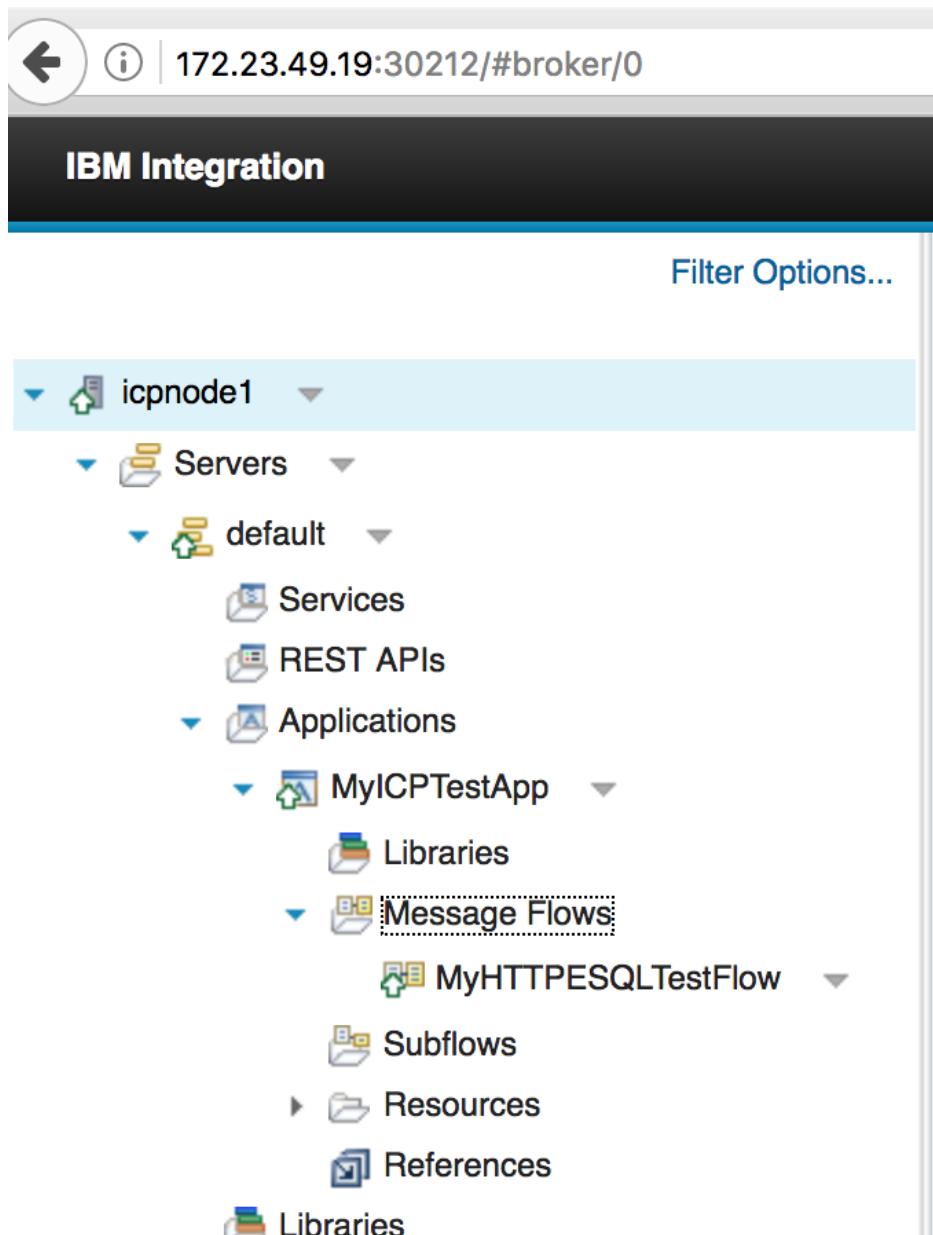
[iib10mq9-iib-mq](#)

[Overview](#)

---

**Service details**

TYPE	DETAIL
Name	iib10mq9-iib-mq
Namespace	default
Creation time	Dec 14th 2017 at 3:26 PM
Type	NodePort
Labels	app=iib10mq9-iib-mq,chart=iib-mq-1.1.0,heritage=Tiller,release=iib10mq9
Selector	app=iib10mq9-iib-mq
IP	10.0.0.147
Port	iib-mq-server 1414/TCP; iib-mq-web 9443/TCP; iib-mq-console 4414/TCP; iib-mq-nodelistener 7080/TCP; iib-mq-serverlistener 7800/TCP
Node port	<a href="#">iib-mq-server 30231/TCP</a> <a href="#">iib-mq-web 31826/TCP</a> <a href="#">iib-mq-console 30212/TCP</a> <a href="#">iib-mq-nodelistener 30505/TCP</a> <a href="#">iib-mq-serverlistener 30395/TCP</a>



Test the deployed message flow

From the ICP Console identify the IIB server listener port.

Services / iib10mq9-iib-mq /

## iib10mq9-iib-mq

### Overview

Service details	
TYPE	DETAIL
Name	iib10mq9-iib-mq
Namespace	default
Creation time	Dec 14th 2017 at 3:26 PM
Type	NodePort
Labels	app=iib10mq9-iib-mq,chart=iib-mq-1.1.0,heritage=Tiller,release=iib10mq9
Selector	app=iib10mq9-iib-mq
IP	10.0.0.147
Port	iib-mq-server 1414/TCP; iib-mq-web 9443/TCP; iib-mq-console 4414/TCP; iib-mq-nodelistener 7080/TCP; iib-mq-serverlistener 7800/TCP
Node port	<a href="#">iib-mq-server 30231/TCP</a> <a href="#">iib-mq-web 31826/TCP</a> <a href="#">iib-mq-console 30212/TCP</a> <a href="#">iib-mq-nodelistener 30505/TCP</a> <a href="#">iib-mq-serverlistener 30395/TCP</a>

In this example 7800 is represented by 30395

Using an HTTP/REST test client like RestEasy for example

URL: <http://172.23.50.111:30395//icpIIBtest>



# REST Easy

POST ▾

http://172.23.49.19:30395/

Send

Headers

Raw

Preview

## + Headers

## - Data

Select one of the options below to include data with the request.

Custom ▾

Enter the data and its corresponding MIME type below.

MIME type

```
doesn't matter.... it should  
return Hello world
```

## + Authentication



# REST Easy

POST ▾

http://172.23.49.19:30395/i

Send

Headers

Raw

Preview

## + Headers

## - Data

Select one of the options below to include data with the request.

Custom ▾

Enter the data and its corresponding MIME type below.

MIME type

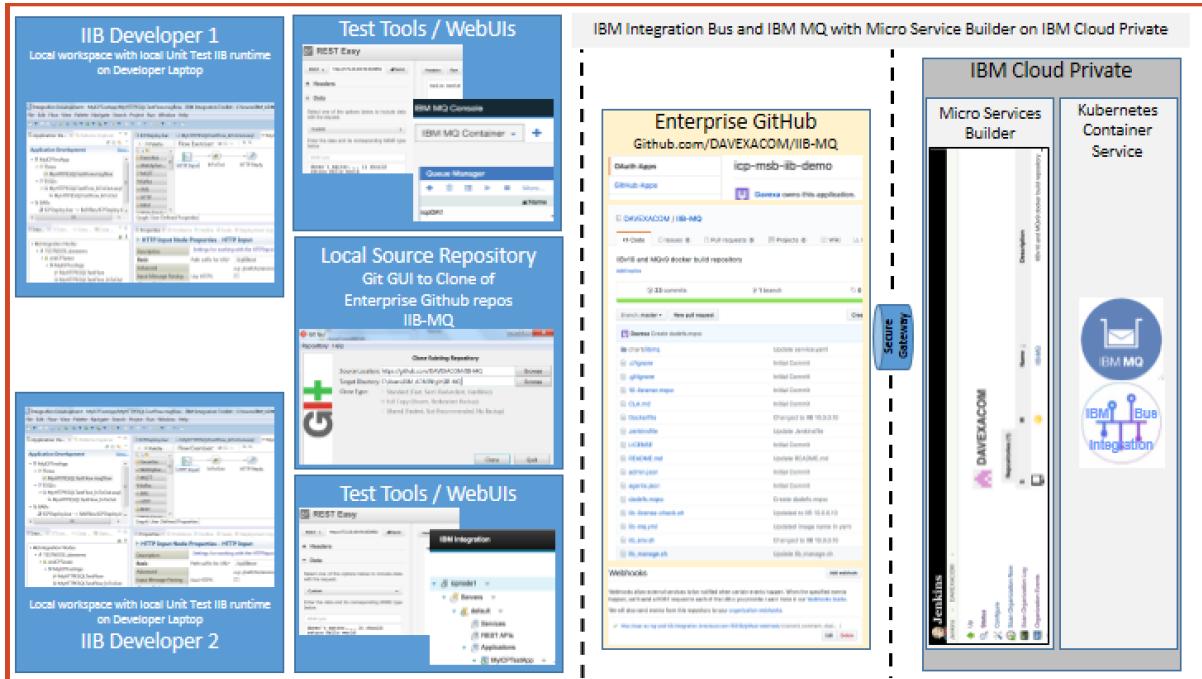
doesn't matter.... it should return Hello world

## + Authentication

hello world

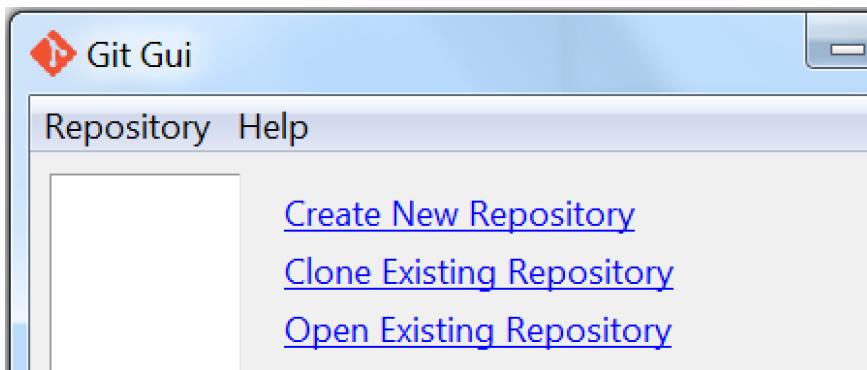
# Automating the Build end to end

## Solution Overview Diagram

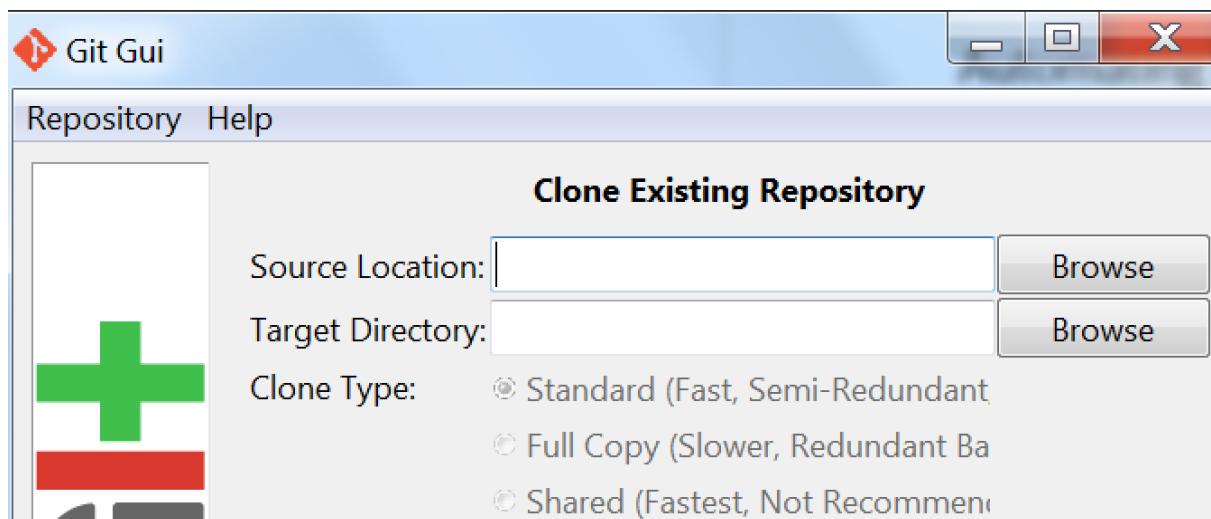


## Creating a local clone of the GitHub Repos IIB-MQ

Example using GitGUI



Select Clone Existing Repository



Cut and paste the URL from the GitHub repository to the Git Gui Source Location

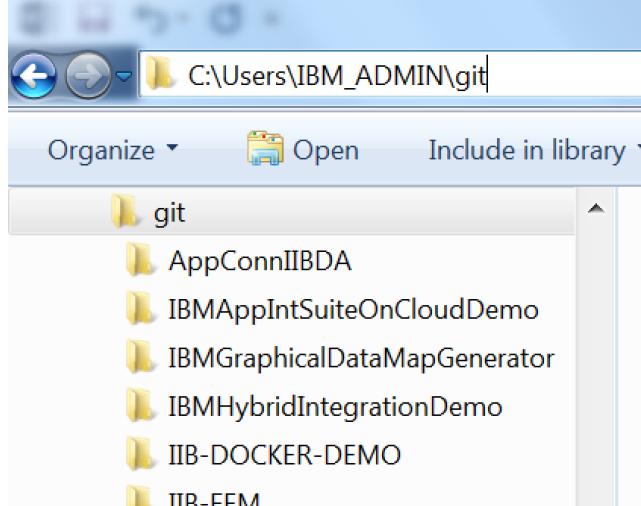
GitHub - DA... Service Details -... Configuring ad

GitHub, Inc. (US) https://github.com/DAVEEXACOM/IIB-MQ

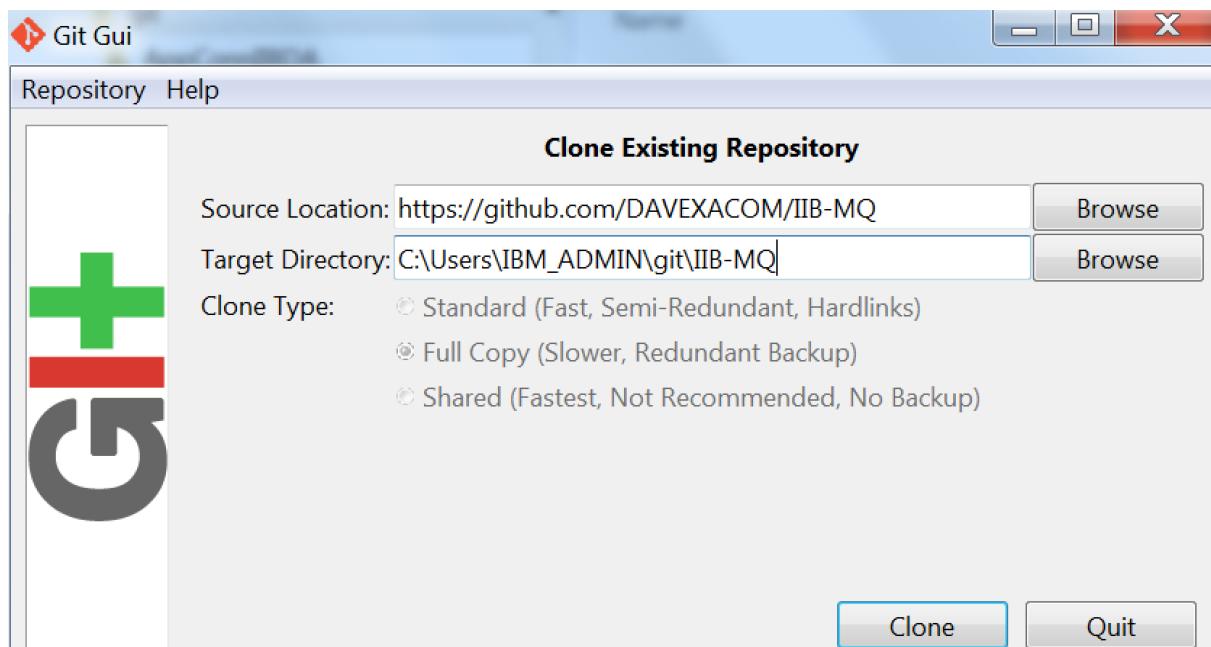
Features Business Explore

DAVEEXACOM / IIB-MQ

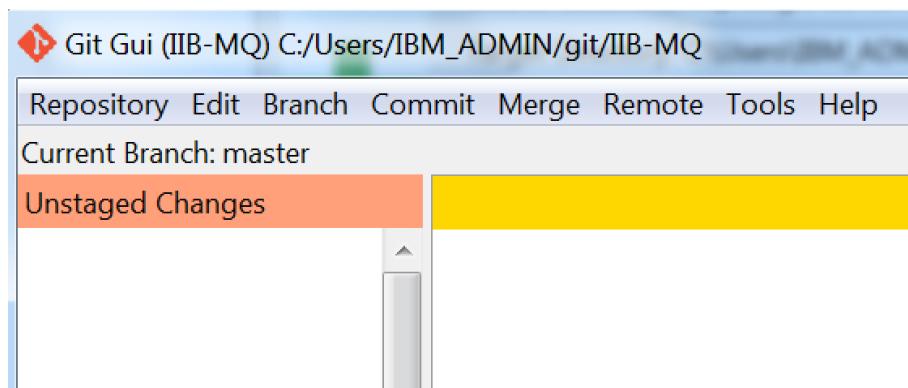
Put and paste the directory path to your local Git Client to the Target Directory... for example



Then add the "IIB-MQ" and hit clone



You now have a local clone.



## *Creating the GitHub WebHook to drive Jenkins*

### Optional – Use IBM Secure Gateway to connect Github to ICP on Pure

Because the ICP instance I am using in my example sits on a non public IBM network, for inbound traffic such as a web hook publication from GitHub I need a secure gateway. This section will show you how to configure this IBM Public Cloud (Bluemix) service.

The secure gateway will offer a public URL that Github can use to trigger the Jenkins build pipeline on the ICP instance on the private network.

So now you have a Public IP address and port for Github to securely connect to Jenkins on the ICP instance running in a private network (The IBM Innovation Center Sydney in this example).

Here is the secure gateway I am using

The screenshot shows a configuration page for an IBM Secure Gateway rule. The rule is titled "jenkins\_32051". The details are as follows:

- Destination ID:** rvKLZNtJz8e\_Glw8u
- Cloud Host : Port:** cap-au-sg-prd-03.integration.ibmcloud.com:15256
- Resource Host : Port:** 172.23.50.111:32051
- Created by:** at 09/01/2018, 2:14:09 pm
- Last modified by:** DO NGUYEN at 09/01/2018, 2:14:55 pm
- Security:** Protocol: TCP  
Download Authentication Files

At the bottom, there are three buttons: "Edit" (green), "Disable" (grey), and "Delete" (red).

The public IPAddr:port <http://cap-au-sg-prd-03.integration.ibmcloud.com:15256> is offered with secure connection to the actual Jenkins IPAddr:port <http://172.23.49.111:32051>



You can use the kubectl command to pull the IBM Secure Gateway image and push it into ICP. You will then need to ssh into the container and set up the ACLs. In a similar way to that shown below (IP/Ports are not a match for this lab doc)

```

-bash-4.2# docker run -it ibmcom/secure-gateway-client r32C030W8kd_prod_au-syd -t eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
eWQlCJpYXQiOjE1MTE4NDMyOTksImV4cCI6MTUxOTYxOTI5OX0.eyJ0YWhuAg-20ajivDDeeYK1lEBzoAXlNpTULonsP5Ja8
IBM Bluemix Secure Gateway Client Version 1.8.0fp3
*****
You are running the IBM Secure Gateway Client for Bluemix. When you enter the provided docker
command the IBM Secure Gateway Client for Bluemix automatically downloads as a Docker image and
is executed on your system/device. This is released under an IBM license. The license agreement
for IBM Secure Gateway Client for Bluemix is available at the following location:

http://www.ibm.com/software/sla/sladb.nsf/lilookup/986C7686F22D4D3585257E13004EA6CB?OpenDocument

Your use of the components of the package and dependencies constitutes your acceptance of this
license agreement. If you do not want to accept the license, immediately quit the container by
closing the terminal window or by entering 'quit' followed by the ENTER key. Then, delete any
pulled Docker image from your device.

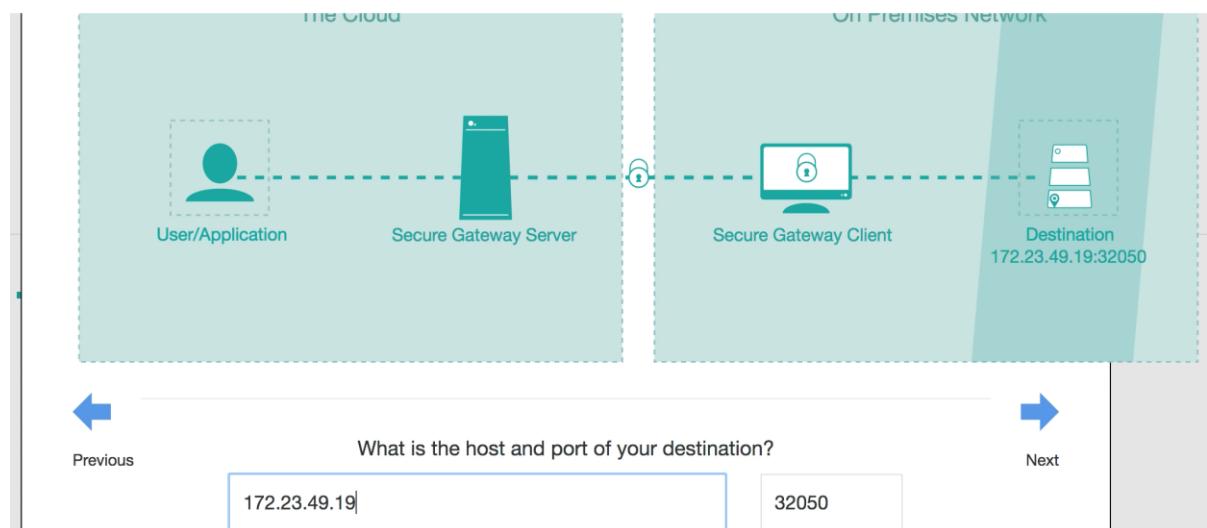
For client documentation, please view the ReadMe located at:
.rpm and .deb installers: /opt/ibm/securegateway/docs/
.dmg installer: <installation location>/ibm/securegateway/docs/
.exe installer: <installation location>\Secure Gateway Client\ibm\securegateway\docs\
*****
<press enter for the command line>
[2017-11-28 08:32:22.244] [INFO] (Client ID 1) No password provided. The UI will not require a password for access
[2017-11-28 08:32:22.259] [WARN] (Client ID 1) UI Server started. The UI is not currently password protected
[2017-11-28 08:32:22.259] [INFO] (Client ID 1) Visit localhost:9003/dashboard to view the UI.
cli> [2017-11-28 08:32:22.595] [INFO] (Client ID 10) Setting log level to INFO
[2017-11-28 08:32:23.032] [INFO] (Client ID 10) The Secure Gateway tunnel is connected
[2017-11-28 08:32:23.056] [INFO] (Client ID r32C030W8kd_dIp) Your Client ID is r32C030W8kd_dIp
[2017-11-28 08:32:23.059] [INFO] (Client ID r32C030W8kd_dIp) Synchronizing ACL rules
r32C030W8kd_dIp> acl allow 172.23.49.19:32050
r32C030W8kd_dIp> show acl
r32C030W8kd_dIp>
-----
-- Secure Gateway Client Access Control List --
Connections to unlisted rules are currently: denied

```

hostname:port	path	value
172.23.49.19:32050		Allow
172.23.49.19:30656		Allow

## Secure Gateway-lv

on: Sydney    Org: nguyendo@au1.ibm.com    Space: bluemix-demo



## Setting up the Webhook in Github

Go to your Github browser, and your organization (DAVEXACOM in this example) and select the IIB-MQ repos. Click on settings.

<https://github.com/DAVEXACOM/IIB-MQ/settings>

This repository Search Pull requests Issues Marketplace Explore

DAVEXACOM / IIB-MQ Unwatch 2

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

**Options**

- Collaborators & teams
- Branches
- Webhooks
- Integrations & services
- Deploy keys
- Alerts

**Settings**

Repository name IIB-MQ Rename

**Features**

Wikis GitHub Wikis is a simple way to let others contribute content. Any GitHub user can cre

Select Webhooks->Add Webhooks

DAVEXACOM / IIB-MQ Unwatch 2 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

**Options**

- Collaborators & teams
- Branches
- Webhooks

**Webhooks** Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

We will also send events from this repository to your [organization webhooks](#).

Confirm password if prompted

Confirm password to continue

Password [Forgot password?](#)

.....

Confirm password

DAVEXACOM / IIB-MQ

Unwatch 2 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

**Options**

**Collaborators & teams**

**Branches**

**Webhooks**

**Integrations & services**

**Deploy keys**

**Alerts**

**Webhooks / Add webhook**

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.

**Payload URL \***

**Content type**

**Secret**

Which events would you like to trigger this webhook?

Just the push event.

Send me everything.

Let me select individual events.

Active  
We will deliver event details when this hook is triggered.

**Add webhook**

The Payload URL is the address and port number of the Jenkins instance on ICP. However, because in this case ICP sits on a private network we will use the public URL offered by the IBM Cloud (Bluemix) secure gateway service to resolves to the ICP jenkins endpoint.

<http://cap-au-sg-prod-03.integration.ibmcloud.com:15256> (or similar)

And add the */github-webhook/* suffix

<http://cap-au-sg-prod-03.integration.ibmcloud.com:15256/github-webhook/> or similar

Webhooks / [Add webhook](#)

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

**Payload URL \***

u-sg-prod-01.integration.ibmcloud.com:15015/github-webhook/

**Content type**

application/x-www-form-urlencoded ▾

**Secret**

Next select which events to cause the webhook to trigger.

Select the radio button “Let me select individual events”

Then tick

- Push
- Pull Request
- Commit Comment
- Status
- Deployment status

Ensure “Active” is ticked and then hit the Add Webhook button

**Add webhook**

Let me select individual events.

**Commit comment**

Commit or diff commented on.

**Create**

Branch or tag created.

**Delete**

Branch or tag deleted.

**Deployment**

Repository deployed.

**Deployment status**

Deployment status updated from the API.

**Fork**

Repository forked.

**Gollum**

Wiki page updated.

**Issue comment**

Issue comment created, edited, or deleted.

**Issues**

Issue opened, edited, closed, reopened, assigned, unassigned, labeled, unlabeled, milestone, or demilestone.

**Label**

Label created, edited or deleted.

**Member**

Collaborator added to, removed from, or has changed permissions for a repository.

**Milestone**

Milestone created, closed, opened, edited, or deleted.

**Page build**

Pages site built.

**Project**

Project created, updated, or deleted.

**Project card**

Project card created, updated, or deleted.

**Project column**

Project column created, updated, moved or deleted.

**Pull request**

Pull request opened, closed, reopened, edited, assigned, unassigned, review requested, review request removed, labeled, unlabeled, or synchronized.

**Pull request review**

Pull request review submitted, edited, or dismissed.

**Pull request review comment**

Pull request diff comment created, edited, or deleted.

**Push**

Git push to a repository.

**Release**

Release published in a repository.

**Repository**

Repository created, deleted, archived, unarchived, publicized, or privatized.

**Status**

Commit status updated from the API.

**Team add**

Team added or modified on a repository.

**Watch**

User stars a repository.

**Active**

We will deliver event details when this hook is triggered.

**Add webhook**

You should see the WebHook added with a green tick next to it.

## Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

We will also send events from this repository to your [organization webhooks](#).

✓ <http://cap-au-sg-prd-03.integration.ibmcloud.com:15256/github-webhook/> (commit\_comment, depl... )

Edit Delete

If you get

## Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

We will also send events from this repository to your [organization webhooks](#).

⚠ <http://cap-au-sg-prod-01.integration.ibmcloud.com:15015/github-webhook/> (commit\_comment, depl... )

Edit Delete

Hit edit to check your settings and also check connectivity all the way to the ICP Jenkins instance.

Once corrected you'll need to hit the "redeliver" button before you get the green tick (or wait for a retry which could take some time)

### Recent Deliveries

✓  [53999a30-f500-11e7-921e-b599cf2b0580](#)

2018-01-09 16:45:40

...

Request

Response 200

⟳ Redeliver

⌚ Completed in 0 seconds.

### Headers

```
Request URL: http://cap-au-sg-prd-03.integration.ibmcloud.com:15256/github-webhook/
Request method: POST
content-type: application/x-www-form-urlencoded
Expect:
User-Agent: GitHub-Hookshot/8f9d70f
X-GitHub-Delivery: 53999a30-f500-11e7-921e-b599cf2b0580
X-GitHub-Event: ping
```

### Payload

```
{
  "zen": "Avoid administrative distraction.",
  "hook_id": 19803163,
```

## Testing the automated build process

### Review Jenkins Status

There should be no builds in the queue at this point

The screenshot shows the Jenkins interface for the 'DAVEXACOM' organization. On the left, there's a sidebar with various links like 'Up', 'Status', 'Configure', etc. The main area is titled 'DAVEXACOM' and shows a 'Repositories (1)' section. A table lists one repository: 'IIB-MQ'. The table has columns for 'S' (Status), 'W' (Last modified), 'Name', and 'Description'. The status is green ('S'), the last modified date is 'Icon: S M L', the name is 'IIB-MQ', and the description is 'IIBv10 and MQv9 docker build repository'. Below the table, it says 'Icon: S M L'.

### Update the IIB message flow

Make a change to the IIB message flow that will be deployed into the ICP IIB runtime as part of the build. Change “Hello world” to “Hello world again” for example.

The screenshot shows the IBM Integration Toolkit interface. The title bar says 'Integration Development - MyICPTestApp/MyHTTPESQLTestFlow\_InToOut.esql - IBM Integration Toolkit - C:\Users\IBM\_ADMIN\git\IIBcpDemo'. The menu bar includes File, Edit, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left pane shows a tree view of the application structure: 'Application Development' node expanded to show 'Flows' (containing 'MyHTTPESQLTestFlow.msgflow'), 'ESQLs' (containing 'MyHTTPESQLTestFlow\_InToOut.esql'), 'BARs' (containing 'ICPDeploy.bar'), and 'Independent Resources' (containing 'BARfiles' with 'ICPDeploy.bar'). The right pane displays the content of 'MyHTTPESQLTestFlow\_InToOut.esql'. The code is as follows:

```
CREATE COMPUTE MODULE MyHTTPESQLTestFlow_InToOut
  CREATE FUNCTION Main() RETURNS BOOLEAN
    BEGIN
      CALL CopyMessageHeaders();

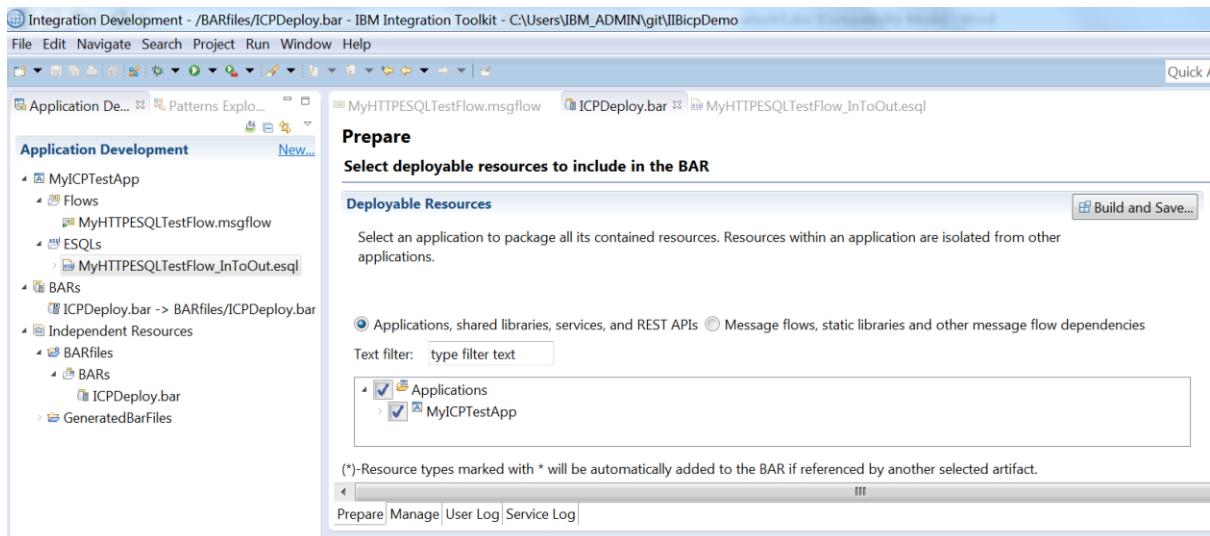
      DECLARE mystring CHARACTER;

      SET mystring='hello world again';
      SET OutputRoot.BLOB.BLOB=CAST (mystring AS BLOB CCSID 1208);

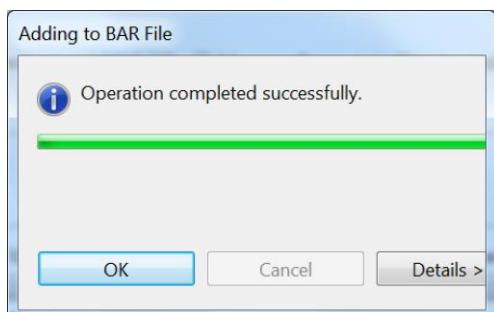
      RETURN TRUE;
    END;

  CREATE PROCEDURE CopyMessageHeaders() BEGIN
    DECLARE I INTEGER 1;
```

Change the mystring text to be returned when MyHttpESQLTestFlow is called.



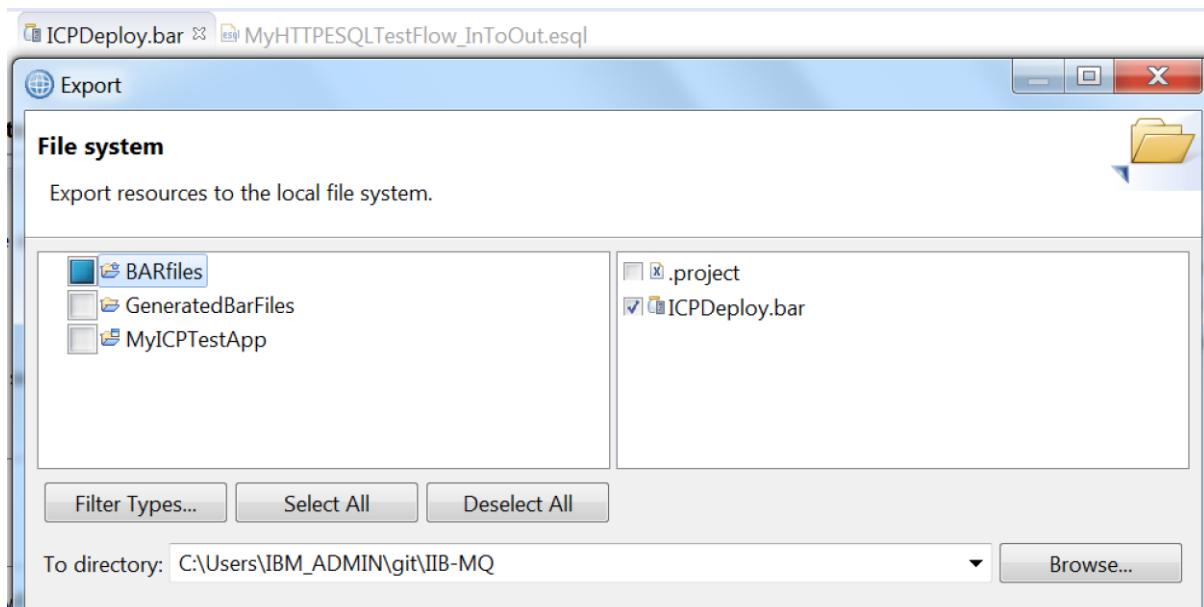
Build and save the BAR file.



You can perform a local deployment and test of the updated bar file if you like.

## Export the BAR file to the local clone of IIB-MQ

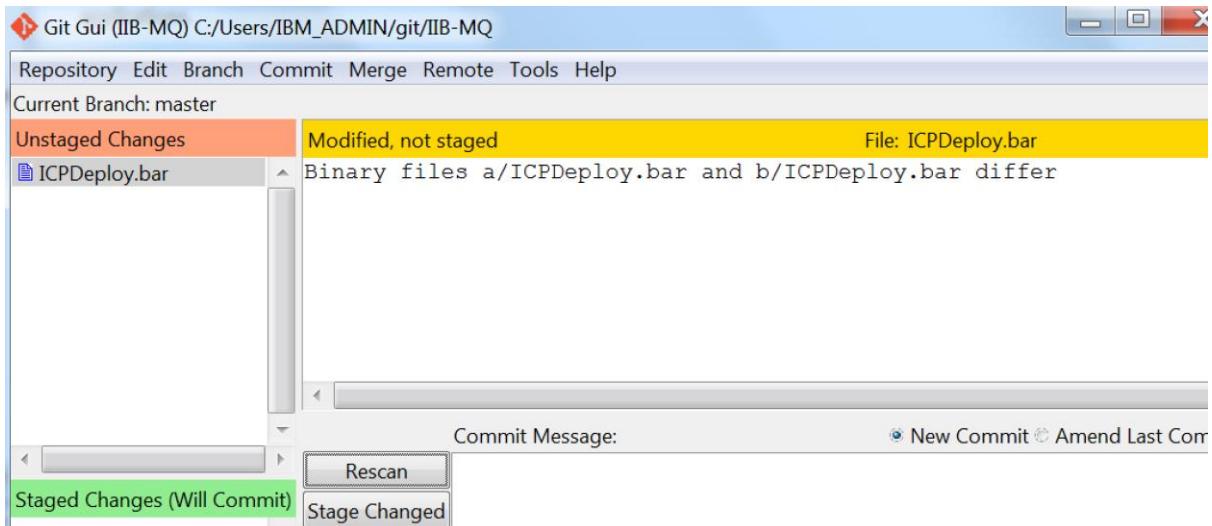
Export the updated BAR file to the File System. File->Export->File System



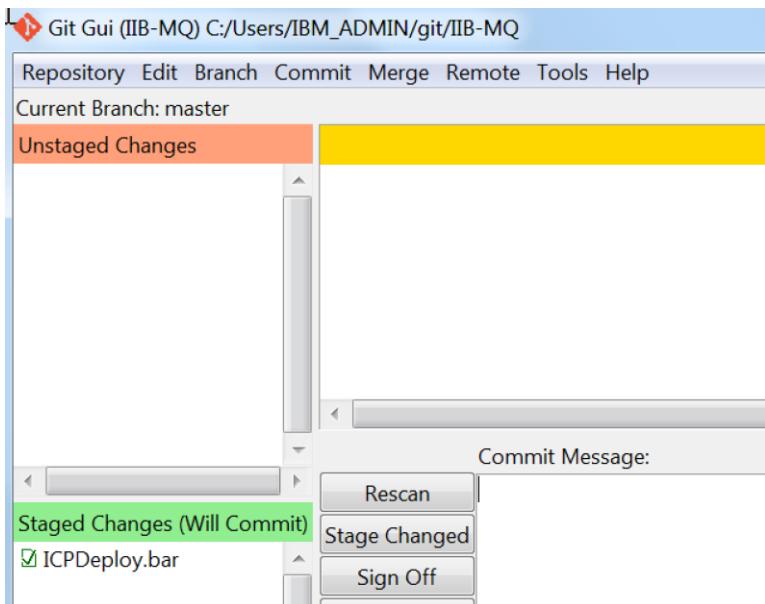
Set the target directory to your local clone of GitHub IIB-MQ repository

## Push to GitHub IIB-MQ repository

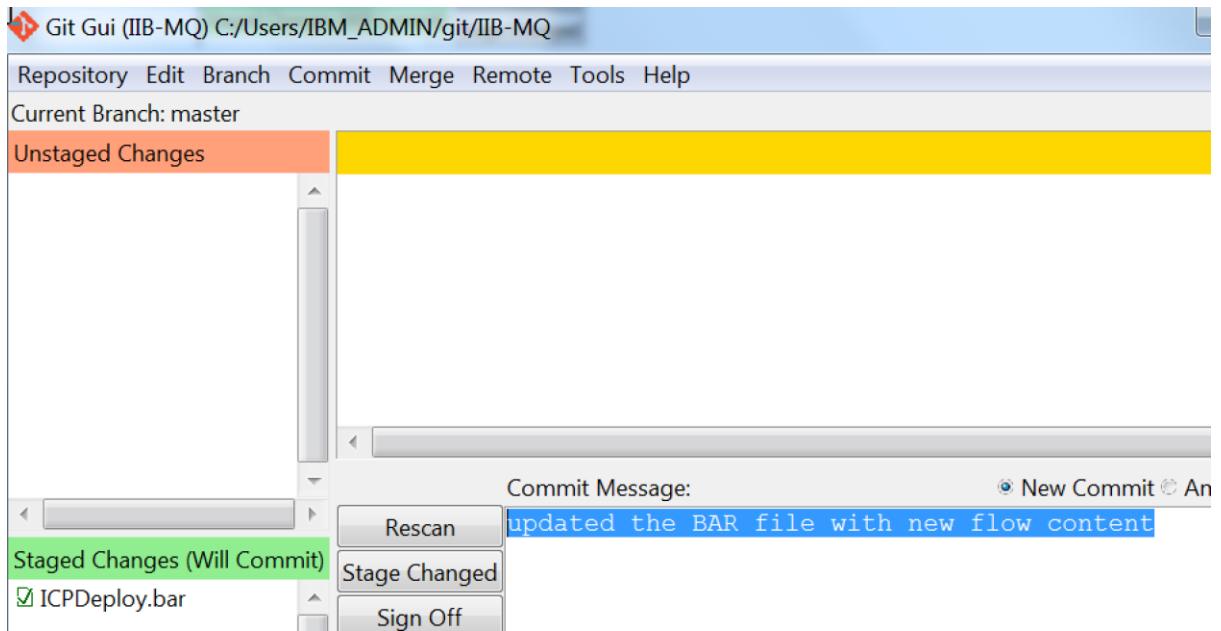
In Git GUI or equivalent hit rescan



Stage changed

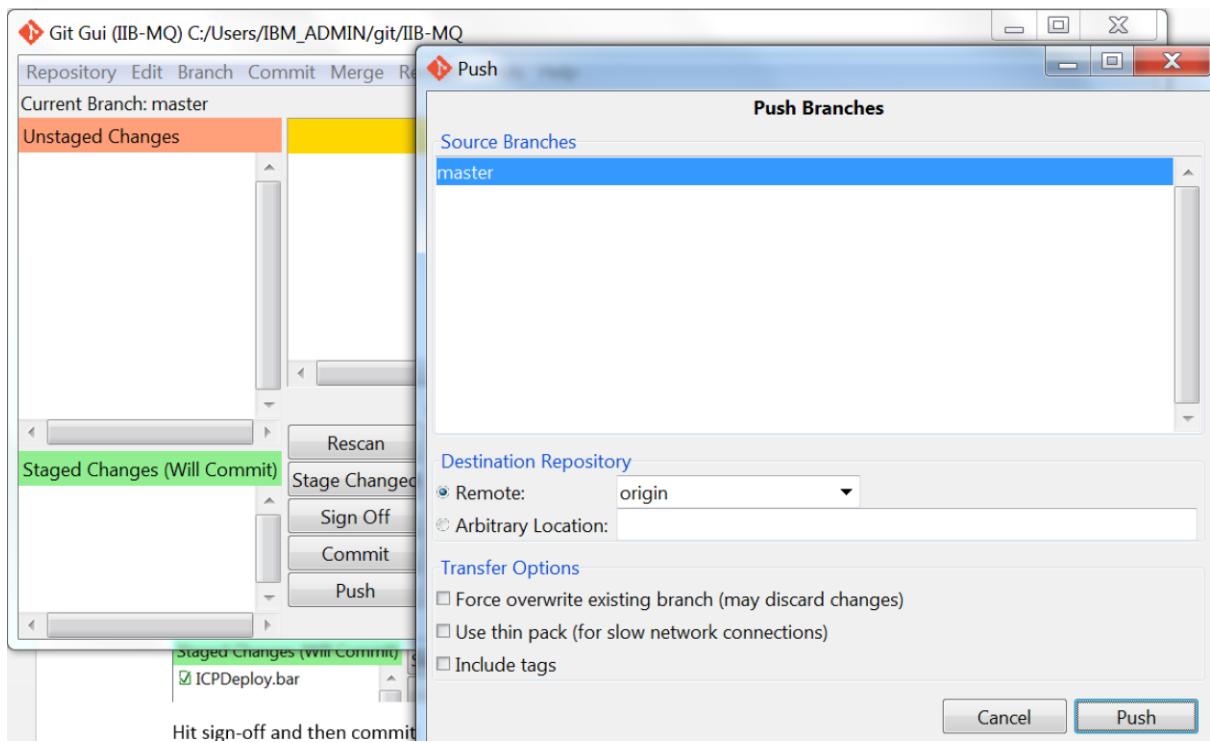


Add a commit message

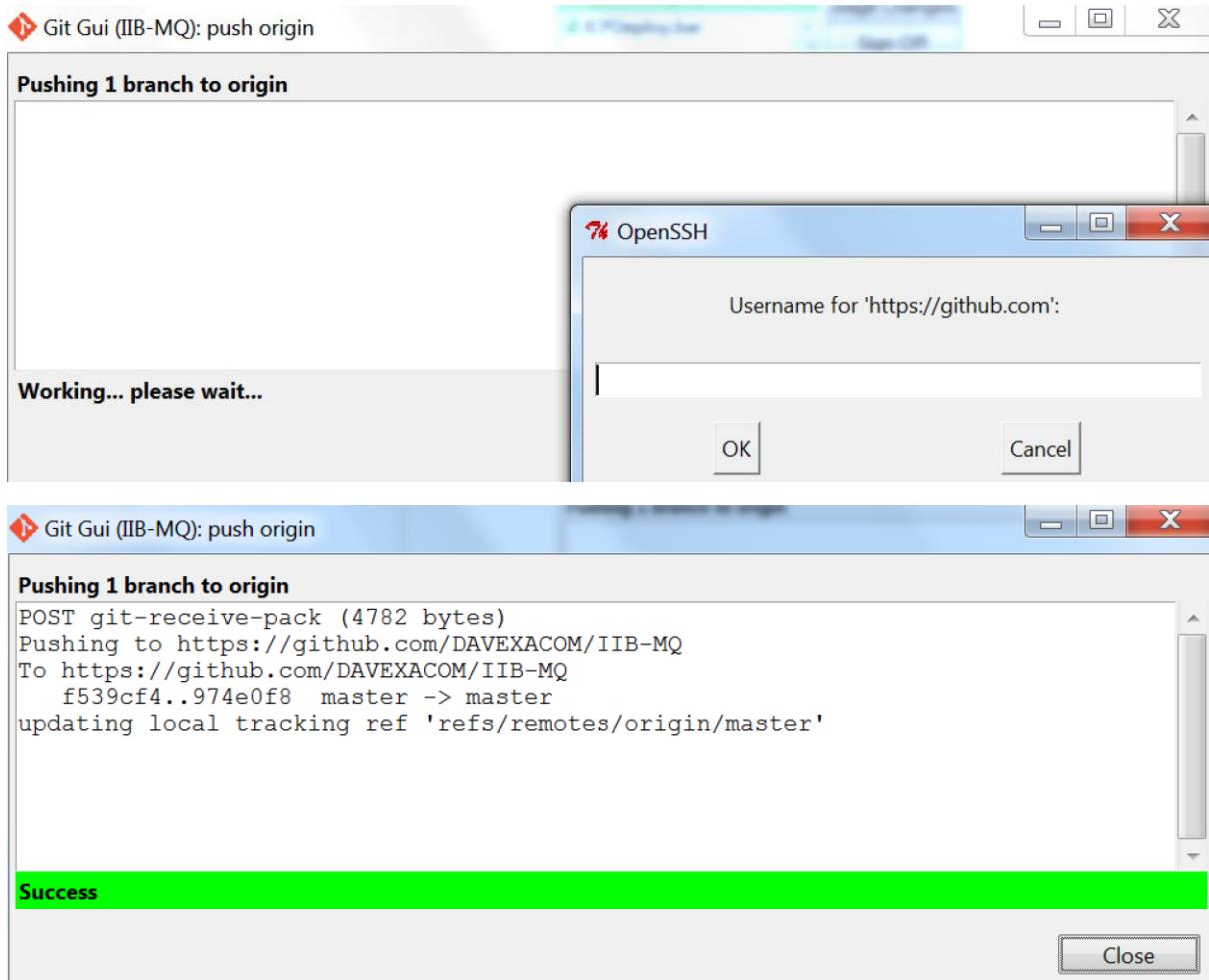


Hit sign-off and then commit

Then start the Push



Sign in with the credentials for your github organization.



## Check the GitHub repository Status

You should see ICPDeploy.bat has been updated

cnart/iibmq	update service.yaml	20 days ago
.cignore	Initial Commit	9 months ago
.gitignore	Initial Commit	9 months ago
10-listener.mqsc	Initial Commit	9 months ago
CLA.md	Initial Commit	9 months ago
Dockerfile	Update Dockerfile	19 days ago
ICPDeploy.bat	updated the BAR file with new flow content	4 minutes ago

## Review Jenkins Status Again

Things may take a moment to kick off

The screenshot shows the Jenkins interface for the 'DAVEXACOM' organization. On the left, there's a sidebar with various navigation links: Up, Status, Configure, Scan Organization Now, Scan Organization Log, Organization Events, Delete Organization, People, Build History, Project Relationship, Check File Fingerprint, GitHub, Pipeline Syntax, and Credentials. Below these are sections for 'Build Queue (1)' and 'Build Executor Status'. The 'Build Queue' section shows a single build entry: 'part of DAVEXACOM » IIB-MQ » master #2'. The 'Build Executor Status' section shows one available executor: 'jenkins-slave-4btdd-kr1q8'. The main content area is titled 'DAVEXACOM' and contains a 'Repositories (1)' table:

S	W	Name ↓	Description
		IIB-MQ	IIBv10 and MQv9 docker build repository

Icon: S M L

And once an Executor becomes available you should see the following.

This screenshot shows the same Jenkins interface for the 'DAVEXACOM' organization, but with a different state. The 'Build Executor Status' section now shows two executors: 'jenkins-slave-4btdd-kr1q8' and 'jenkins-slave-4btdd-kr1q8'. The 'Build Queue' section is empty, displaying the message 'No builds in the queue.' The 'Build Queue (1)' section from the previous screenshot is still present but inactive.

## Check the Jenkins build logs

Click on the **#n** and then select Console Output



1 DAVEXACOM » IIB-MQ » #2  
master (Docker Build)

[Back to Project](#)

[Status](#)

[Changes](#)

**Console Output**

[View as plain text](#)

[Edit Build Information](#)

[Git Build Data](#)

[No Tags](#)

[Git Build Data](#)

[Thread Dump](#)

[Pause/resume](#)

[Replay](#)

[Pipeline Steps](#)

[Embeddable Build Status](#)

**Console Output**

```

Push event to branch master
06:06:39 Connecting to https://api.github.com using Davexa/***** (User/Pass for GitHub)
Obtained Jenkinsfile from e5ada335ef9a83ed1d6e6d9fe83ba15262923364
Loading library MicroserviceBuilder@2.1.0
Attempting to resolve 2.1.0 from remote references...
> git --version # timeout=10
using GIT_ASKPASS to set credentials User/Pass for GitHub
> git ls-remote -h -t https://github.com/WASdev/microservicebuilder.lib.git # timeout=10
Found match: refs/heads/2.1.0 revision 17f9d610ef18179d153fe23915a21db6f3c4fdb6
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/WASdev/microservicebuilder.lib.git # timeout=10
Fetching without tags
Fetching upstream changes from https://github.com/WASdev/microservicebuilder.lib.git
> git --version # timeout=10
using GIT_ASKPASS to set credentials User/Pass for GitHub
> git fetch --no-tags --progress https://github.com/WASdev/microservicebuilder.lib.git +refs/heads/*:refs/remotes/origin/*
Checking out Revision 17f9d610ef18179d153fe23915a21db6f3c4fdb6 (2.1.0)
~ git config core.sparsecheckout # timeout=10

```

Once complete the tail of the output will look similar to below.

```

[DAVEXACOM_IIB-MQ_master-MPOWJ5QPOSBOGPPW2KNK1ZRPJQRLX5DN7HQYK7RV5DPBNS2U3S2A] Running shell script
+ /helm init --client-only --skip-refresh
Creating /home/jenkins/.helm
Creating /home/jenkins/.helm/repository
Creating /home/jenkins/.helm/repository/cache
Creating /home/jenkins/.helm/repository/local
Creating /home/jenkins/.helm/plugins
Creating /home/jenkins/.helm/starters
Creating /home/jenkins/.helm/cache/archive
Creating /home/jenkins/.helm/repository/repositories.yaml
$HELM_HOME has been configured at /home/jenkins/.helm.
Not installing Tiller due to 'client-only' flag having been set
Happy Helming!
[Pipeline] fileExists
[Pipeline] sh
[DAVEXACOM_IIB-MQ_master-MPOWJ5QPOSBOGPPW2KNK1ZRPJQRLX5DN7HQYK7RV5DPBNS2U3S2A] Running shell script
+ /helm upgrade --install --wait --values pipeline.yaml --namespace default iib10mq9 chart/iibmq
Release "iib10mq9" has been upgraded. Happy Helming!
LAST DEPLOYED: Tue Jan  9 06:17:48 2018
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1beta1/StatefulSet
NAME          DESIRED   CURRENT   AGE
iib10mq9-iib-mq  1         1        49m

==> v1/Secret
NAME          TYPE      DATA   AGE
iib10mq9-iib-mq Opaque    2      49m

==> v1/Service
NAME          CLUSTER-IP  EXTERNAL-IP  PORT(S)           AGE
iib10mq9-iib-mq  10.0.0.14  <nodes>     1414:31892/TCP,9443:31416/TCP,4414:32681/TCP,7080:30010/TCP,7800:31207/TCP  49m

[Pipeline] }
[Pipeline] // container
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // podTemplate
[Pipeline] End of Pipeline

```

## Check the results in the IIB Web UI

From ICP console -> Workloads -> Helm releases select iib10mq9 then select Services to get to the link to the Web UI (iib-mq-web 31826 in this example)

Services / [iib10mq9-iib-mq](#) /

[iib10mq9-iib-mq](#)

[Overview](#)

---

**Service details**

TYPE	DETAIL
Name	iib10mq9-iib-mq
Namespace	default
Creation time	Dec 14th 2017 at 3:26 PM
Type	NodePort
Labels	app=iib10mq9-iib-mq,chart=iib-mq-1.1.0,heritage=Tiller,release=iib10mq9
Selector	app=iib10mq9-iib-mq
IP	10.0.0.147
Port	iib-mq-server 1414/TCP; iib-mq-web 9443/TCP; iib-mq-console 4414/TCP; iib-mq-nodelistener 7080/TCP; iib-mq-serverlistener 7800/TCP
Node port	<a href="#">iib-mq-server 30231/TCP</a> <a href="#">iib-mq-web 31826/TCP</a> <a href="#">iib-mq-console 30212/TCP</a> <a href="#">iib-mq-nodelistener 30505/TCP</a> <a href="#">iib-mq-serverlistener 30395/TCP</a>

The screenshot shows the IBM Integration Console interface. At the top, there is a header bar with a back arrow, an information icon, and the URL `172.23.49.19:30212/#broker/0`. Below the header is a dark navigation bar with the text "IBM Integration". On the right side of the navigation bar is a "Filter Options..." link. The main content area displays a hierarchical deployment tree. The root node is "icpnode1", which has a "Servers" node. Under "Servers" is a "default" node, which contains "Services", "REST APIs", and an "Applications" node. The "Applications" node has a child node "MyICPTTestApp", which contains "Libraries", "Message Flows", "Subflows", "Resources", "References", and another "Libraries" node. The "Message Flows" node is currently selected, indicated by a dotted border around its icon.

- icpnode1
  - Servers
    - default
      - Services
      - REST APIs
      - Applications
        - MyICPTTestApp
          - Libraries
          - Message Flows
          - Subflows
          - Resources
          - References
          - Libraries

Test the deployed message flow

From the ICP Console identify the IIB server listener port.

Services / iib10mq9-iib-mq /

## iib10mq9-iib-mq

### Overview

Service details	
TYPE	DETAIL
Name	iib10mq9-iib-mq
Namespace	default
Creation time	Dec 14th 2017 at 3:26 PM
Type	NodePort
Labels	app=iib10mq9-iib-mq,chart=iib-mq-1.1.0,heritage=Tiller,release=iib10mq9
Selector	app=iib10mq9-iib-mq
IP	10.0.0.147
Port	iib-mq-server 1414/TCP; iib-mq-web 9443/TCP; iib-mq-console 4414/TCP; iib-mq-nodelistener 7080/TCP; iib-mq-serverlistener 7800/TCP
Node port	<a href="#">iib-mq-server 30231/TCP</a> <a href="#">iib-mq-web 31826/TCP</a> <a href="#">iib-mq-console 30212/TCP</a> <a href="#">iib-mq-nodelistener 30505/TCP</a> <a href="#">iib-mq-serverlistener 30395/TCP</a>

In this example 7800 is represented by 30395

Using an HTTP/REST test client like RestEasy for example

URL: <http://172.23.50.111:30395//icpIIBtest>



# REST Easy

POST ▾

http://172.23.49.19:30395/

Send

Headers

Raw

Preview

## + Headers

## - Data

Select one of the options below to include data with the request.

Custom ▾

Enter the data and its corresponding MIME type below.

MIME type

```
doesn't matter.... it should  
return Hello world
```

## + Authentication

Hit Send and check the result in preview



# REST Easy

POST ▾

'2.23.50.111:31207/icpIIBytes

Send

+ Headers

- Data

Select one of the options below to include data with the request.

Custom ▾

Enter the data and its corresponding MIME type below.

MIME type

Doesn't matter.... it should say  
hello world again

Headers

Raw

Preview

Hello world again

