

IBM MQ
And
IBM Integration Bus
on
IBM Cloud Private
With
Micro Services Builder Pipeline

v1.0

Contents

Introduction	5
The ICP Instance used in this document and its location	5
Screenshots and instructions in this document.....	5
Scenario Overview	6
Scenario Diagram	6
Scenario Description	6
IIB & MQ Container on ICP from Github repository via MSB pipeline – Important Parameters.....	8
Clone the DAVEXACOM\IIB-MQ GitHub Repository.....	8
Exploring the Github repository IIB-MQ.....	10
The Jenkins File	10
The Helm Chart files.....	10
IBM MQ how the repository files relate to each other	12
Customizing the IIB-MQ GitHub Repo to reflect your MQ definitions	13
IBM Integration Bus how the repository files relate to each other.....	14
Customizing the IIB-MQ GitHub Repo to reflect your IIB BAR files	14
Optional Section – Upload existing or create a BAR file in an IIB Toolkit	15
Optional Section – Create a BAR file in an IIB Toolkit.....	15
Optional Section – Upload the BAR file to the IIB-MQ repo on GitHub	16
Configuring Github repository IIB-MQ.....	18
OAuth Access for Github repository IIB-MQ in organization DAVEXACOM	18
Github Personal Access Token – Create a token	18
Developer Settings – Personal Access tokens.....	18
OAuth Apps – Register a new application	21
Developer Settings – Set up OAuth.....	21
OAuth Apps – Register a new application.....	21
Capture OAuth Client ID and Secret	23
IBM Cloud Private – Configure and Deploy Micro Service Builder	24
Check Micro Service Builder Fabric Service	24
Check Jenkins Persistent Volume.....	24
Configure and Deploy Micro Service Builder Pipeline	25

Configure the ibm-microservicebuilder pipeline helm chart.....	25
Review the Micro Service Builder pipeline helm release.....	30
Authorize the ibm-msb-pipeline-da service to access github.....	31
Micro Services Builder - Jenkins.....	32
Checking Jenkins connection to Github	32
Before Starting the Jenkins build process	33
Starting the Jenkins build process manually.....	34
Following the Jenkins build logs.....	35
Checking ICP Image repository	38
Checking ICP Helm Release	39
Connect MQ Console	41
Connect to IIB Web UI.....	43
Test the deployed message flow	44
Automating the Build end to end	46
Solution Overview Diagram	46
Creating a local clone of the GitHub Repos IIB-MQ.....	46
Example using GitGUI.....	46
Creating the GitHub WebHook to drive Jenkins	49
Optional – Use IBM Secure Gateway to connect Github to ICP on a Private Network	49
Setting up the Webhook in Github	51
Testing the automated build process	56
Review Jenkins Status	56
Update the IIB message flow	56
Export the BAR file to the local clone of IIB-MQ.....	57
Push to GitHub IIB-MQ repository	58
Check the GitHub repository Status.....	60
Review Jenkins Status Again	60
Check the Jenkins build logs	61
Check the results in the IIB Web UI	63
Test the deployed message flow	64
Reference Section	66

How the DAVEXACOM/IIB-MQ Repository in GitHub was created	66
Source GitHub Repositories to leverage.....	66
1. IIB an MQ Docker build repository	66
2. IIB and MQ Helm Chart repository.....	66
Target GitHub Repository in Github Organization.....	66
Add a Jenkins file to the repository	68
Add the IIB and MQ Helm chart files to the repository	69
Download from the iib-mq-chart repository to the local file system	70
Import to from the iib-mq-chart repository to the local file system	71

Introduction

This document will explore the deploying of IBM MQ and IBM Integration Bus on IBM Cloud Private (ICP) using the Micro Services Builder pipeline to automate the delivery of integration.

The scenario described below simulates multiple IIB developers working independently and unit testing on their local machines.

Once they are happy with their work they “check-in” (export) their BAR files (and MQSC scripts) into a source repository.

These updates are pushed to public a Github IIB-MQ repository set up to deliver an IIB and MQ System Integration Test (SIT) environment on IBM Cloud Private (Kubernetes container Service).

A GitHub webhook initiates a Micro Services Builder Pipeline (Jenkins) process on ICP that has been registered as an OAuth application to GitHub.

The pipeline builds and deploys the IIB and MQ SIT environment with all the latest IIB and MQ artefacts deploy and ready for socialization testing.

The ICP Instance used in this document and its location

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

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.

Screenshots and instructions in this document

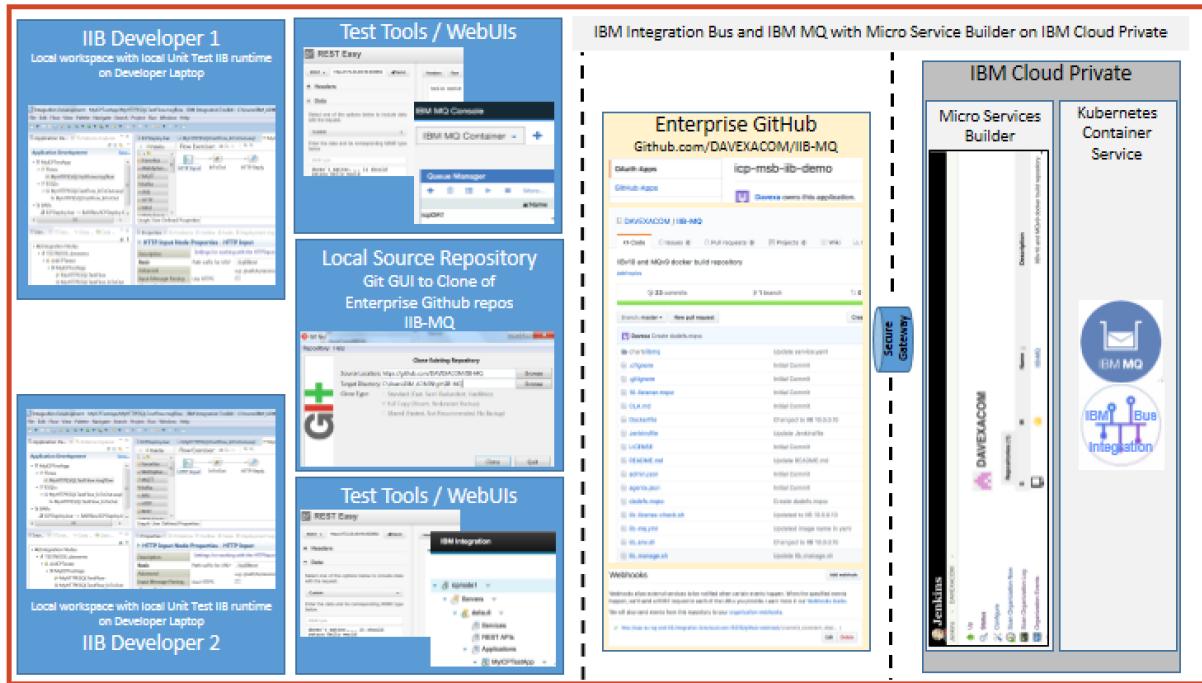
During the construction of this document the ICP instance in the Sydney IIC Pure Application system was upgraded and rebuilt. Therefore, IP address and port numbers in screen shots will have changed. For example

Details that changed	Before Upgrade/Rebuild	After Upgrade/Rebuild
ICP Console IP	https://172.23.49.17:8443/console/	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.

Scenario Overview

Scenario Diagram



Scenario Description

The basic premise of this scenario is configuring IBM Cloud Private Micro Services Builder Pipeline (Jenkins) and GitHub (Public) to deliver streamlined DevOps for IBM Integration Bus and IBM MQ.

Different organizations will “do integration” differently. As an IIB Developer I am set up in the following way.

1. I work in the IIB Toolkit on my laptop and unit test my message flows on my local IIB node
2. When I am happy with my work I export either BAR files or complete projects to Git on my local system. Git is a fast, scalable, distributed revision control system. <http://git.htmldocs.googlecode.com/git/git.html>.
3. I then use Git GUI to push changes to my public GitHub organization DAVEXACOM for safe keeping and sharing with other developers.

Other IIB Developers install the Eclipse Git perspective into their IIB Toolkit and push to GitHub directly. (Which in reality is a cleaner approach). Basically, it does not matter the key here is getting new or updated BAR files and MQSC files into the IIB-MQ repository in our GitHub organization.

This target IIB-MQ Github repository needs to be configured in three ways to interact with Micro Services Builder (MSB) Pipeline (Jenkins) on ICP.

1. Add MSB (Jenkins) OAuth application access to the repo
2. Create a WebHook to “call” MSB (Jenkins) pipeline URL when updates happen in the repo
3. Include a jenkinsfile to provide information about the repo file structure to Jenkins.

The DAVEXACOM\IIB-MQ repository is “ready to go” in terms of it’s file content and your first step will be to clone the repository into your organization. However, you still have to create the OAuth app and Webhook for your copy and optionally you may want to customize certain aspects/files in the repo.

This document will walk you through the following steps.

1. Git Clone the DAVEXACOM\IIB-MQ repository to your GitHub organization
2. Explore the IIB-MQ file structure and the relationships such that you understand how an IIB/MQ Helm release repo for ICP hangs together, specifically:
 - a. The Docker file
 - b. The Jenkins file
 - c. The helm chart YAMLS
 - d. The .SH that executes when the container starts
3. Optionally change or add IBM MQ MQSC files or IIB BAR files in your copy of the IIB-MQ repo
4. Configure your IIB-MQ repo for OAuth application access from IBM MSB (Jenkins)
5. Configure and Deploy IBM MSB Pipeline (Jenkins) on IBM Cloud Private.
6. Kick off your first IIB/MQ deployment manually using Jenkins “Scan Repository Now”
7. Connect IBM MQ Web Console/IIB WebUI
8. Test the MyHTTPESQLTestFlow (hello world message flow) deployed on IIB
9. Optionally create a local git clone on your IIB Toolkit workstation of your IIB-MQ repo so you can save you IIB Development work and push to the GitHub master repository (you don’t have to do this you can use another mechanism or simply upload changed or new bar files).
10. Create and configure the WebHook on your IIB-MQ Github repo to “call” the Jenkins URL on the ICP MSB pipeline to perform a new build when changes are made to the IIB-MQ repo.
11. Optionally. As Webhooks are inbound to Jenkins if the ICP instance is on a private network and you are using public Github you’ll need something like the IBM Cloud (Bluemix) Secure Gateway configured to offer a public IP address and port number to the Jenkins Webhook URL.
12. Update the MyHTTPESQLTestFlow (Project Interchange supplied) to change the “Hello World” message and Build and Save the BAR file.
13. Upload or Git Push the BAR file to your IIB-MQ repo
14. Follow the IBM MSB pipeline (Jenkins) console logs
15. Reconnect the IIB Web UI and IBM MQ Console
16. Retest the MyHTTPESQLTestFlow and observe the new “Hello World” message

Now everytime you make a change or add MQSC files or BAR files in your IIB-MQ repo Jenkins will rebuild your environment with the latest changes.

IIB & MQ Container on ICP from Github repository via MSB pipeline – Important Parameters

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 values	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	http://172.23.50.111:32050 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	https://github.com/settings/applications/635182	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

Clone the DAVEXACOM\IIB-MQ GitHub Repository

Click on this link or copy into your browser <https://github.com/DAVEXACOM/IIB-MQ>

IIBv10 and MQv9 docker build repository

30 commits 1 branch 0 releases 2 contributors

Branch: master New pull request

Davexa Update README.md
 chart/iibmq Update service.yaml
 .cignore Initial Commit
 .gitignore Initial Commit

Clone with HTTPS
<https://github.com/DAVEXACOM/IIB-MQ.git>

[Open in Desktop](#) [Download ZIP](#)

At this point you can either:

1. download the repository as a ZIP to your local system then login to GitHub, goto your organization and create an new IIB-MQ repository and upload all the files OR
2. Copy the URL from the box. Log into GitHub, goto your organization and import an New repository

Your old repository's clone URL
<https://github.com/DAVEXACOM/IIB-MQ.git>

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

Your new repository details

Owner	Name
giphos	IIB-MQ

From here on you will be working against you copy of the Repository in your organization

Exploring the Github repository IIB-MQ

The Jenkins File

The Jenkinsfile must be present in the repository in order for the Jenkins process in Micro Service Builder to scan the repository and recognise it as a repository it should perform a build for. Jenkins will skip over repositories in organizations it scans if there is no Jenkinsfile.

See below the Jenkins file names the image, tells Jenkins where to find the helm chart files specifies a Git deploy branch to work against and the target namespace on ICP to target.

Branch: master ▾ [IIB-MQ / Jenkinsfile](#)

 Davexa Update Jenkinsfile

1 contributor

11 lines (9 sloc) | 218 Bytes

```
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 }
```

The Helm Chart files

 DAVEXACOM / IIB-MQ

Code Issues 0 Pull requests 0 Projects 0 Wiki

Branch: master ▾ [IIB-MQ / chart / iibmq /](#)

 Davexa Update service.yaml

..

 LICENSES	Add files via upload
 templates	Update service.yaml
 Chart.yaml	Add files via upload
 README.md	Add files via upload
 values.yaml	Update values.yaml

 DAVEXACOM / IIB-MQ

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#)

Branch: master ▾ [IIB-MQ / chart / iibmq / templates /](#)

 Davexa Update service.yaml

..	
_helpers.tpl	Add files via upload
secret-dev.yaml	Add files via upload
service.yaml	Update service.yaml
stateful-set.yaml	Add files via upload
test.yaml	Add files via upload

Values.yaml

The values.yaml file in the GitHub repository is where you can 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
```

IBM MQ how the repository files relate to each other

The screenshot shows a GitHub repository page for 'DAVEXACOM / IIB-MQ'. The repository is named 'IIBv10 and MQv9 docker build repository'. It has 23 commits, 1 branch, and 0 issues. The commit list includes files like chart/iibmq, .cignore, .gitignore, 10-listener.mqsc, CLA.md, Dockerfile, Jenkinsfile, LICENSE, README.md, admin.json, agentx.json, dadefs.mqsc, iib-license-check.sh, iib-mq.yml, iib_env.sh, and iib_manage.sh. Each file has a brief description of its purpose.

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

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
--
```

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**

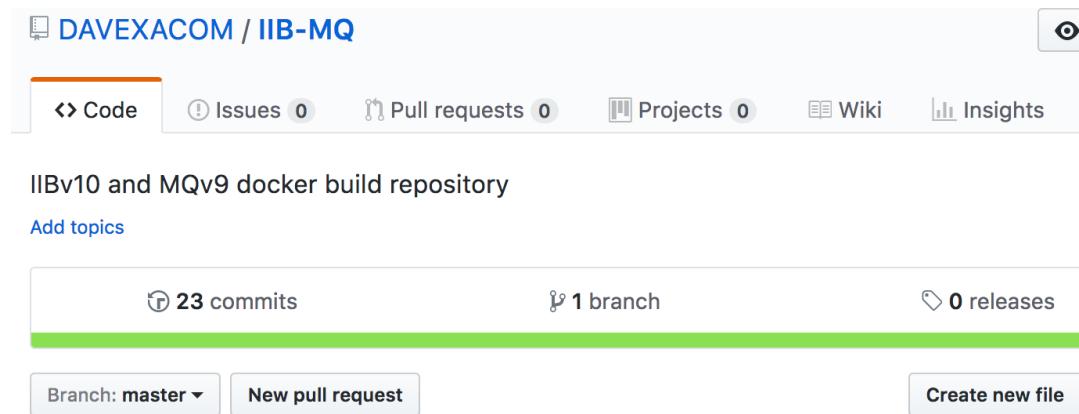
```
127 # Set entrypoint to run management script
128 ENTRYPOINT ["iib_manage.sh"]
```

`lib_manage.sh` is executed when the container starts up, creating QMgrs and starting them. Amongst other things it executes `runmqsc` for all MQSC files in the repository as follows:

```
for MQSC_FILE in $(ls -v /etc/mqm/*.mqsc); do
    runmqsc ${MQ_QMGR_NAME} < ${MQSC_FILE}
done
```

Customizing the IIB-MQ GitHub Repo to reflect your MQ definitions

Create a new .MQSC file or edit `dadefs.mqsc` (to make changes)



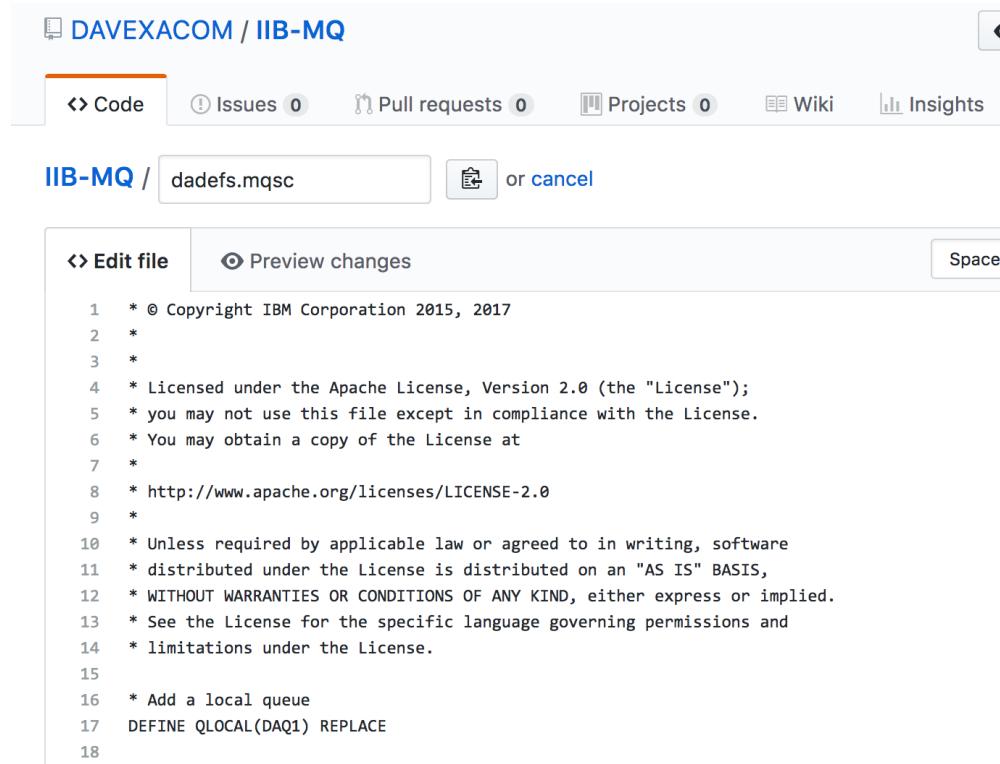
IIBv10 and MQv9 docker build repository

Add topics

23 commits 1 branch 0 releases

Branch: master ▾ New pull request Create new file

Add your own MQ definitions and commit the changes.



DAVEVACOM / IIB-MQ

IIB-MQ / dadefs.mqsc or cancel

Edit file Preview changes Space

```
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
```

IBM Integration Bus how the repository files relate to each other

Customizing the IIB-MQ GitHub Repo to reflect your IIB BAR files

Basically, we use the same approach as for MQ above to customize for IIB but Instead of creating a file we'll import a BAR file to the repository from a local IIB toolkit workspace or push from a local clone of the IIB-MQ repository.

The dockerfile to copies all ***.bar** file to mqm/etc along with the mqsc files

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
--
```

The iib_manage.sh file runs at docker container start time and creates and starts IIB nodes etc, and amongst other things executes the mqsideploy command. You can add extra lines to deploy additional BARs or code a for loop around the mqsideploy command for example:

The iib_manage.sh file on GitHub

```
140
141      fi
142      mqsidechangeproperties $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()
```

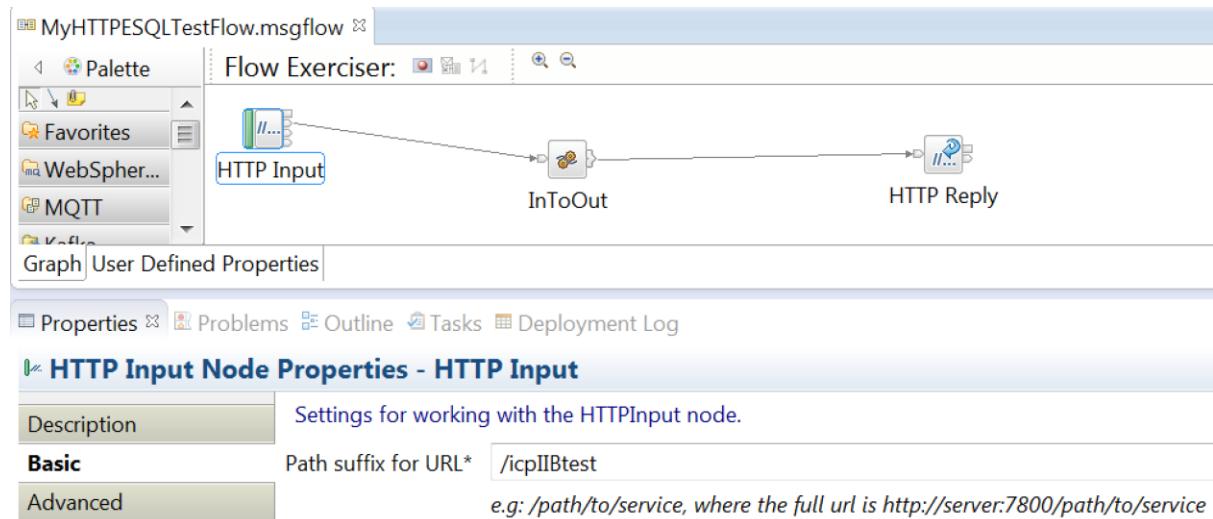
Untested example below of the loop to deploy all BARs

```
for BAR_FILE in $(ls -v /etc/mqm/*.bar); do
    mqsideploy ${NODE_NAME} -e ${EXEC_NAME} -a ${BAR_FILE} -m
done
```

Optional Section – Upload existing or create a BAR file in an IIB Toolkit

Optional Section – 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 MyHTTPESQLTestFlow_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

The screenshot shows the 'Create a new BAR file' dialog box. It has fields for 'Container' (set to 'BARfiles'), 'Folder' (set to '<default>'), and 'Name' (set to 'ICPDeploy').

Select MyICPTestApp and hit Build and Save

MyHTTPESQLTestFlow.msgflow *ICPDeploy.bar

Prepare

Select deployable resources to include in the BAR

Deployable Resources

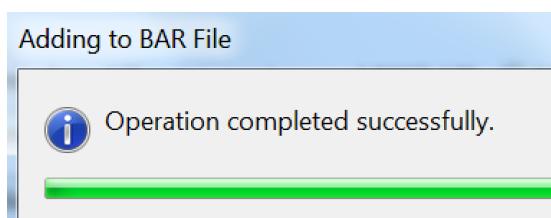
Select an application to package all its contained resources. Resources within an application are isolated from other applications.

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

Type filter text:

Applications
MyICPTestApp

Build and Save...



Optional Section – 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

DAVEXACOM / IIB-MQ

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

IIB-MQ /

Drag file

File Upload

Organize New folder

Favorites Libraries Name

.project ICPDeploy.bar

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**

Check the repository for your BAR file.

DAVEXACOM / IIB-MQ

Code Issues 0 Pull requests 0 Projects 0 Wiki

IIBv10 and MQv9 docker build repository

Add topics

24 commits 1 branch

Branch: master ▾ New pull request

 Davexa Add files via upload

 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

Configuring Github repository IIB-MQ

OAuth Access for Github repository IIB-MQ in organization DAVEXACOM

The screenshot shows the GitHub repository page for 'DAVEXACOM / IIB-MQ'. The URL is https://github.com/DAVEXACOM/IIB-MQ. The repository name is 'IIBv10 and MQv9 docker build repository'. It has 28 commits, 1 branch, and 0 releases. The commit list includes:

- chart/iibmq: Update service.yaml
- .cignore: Initial Commit
- .gitignore: Initial Commit
- 10-listener.mqsc: Initial Commit
- CLA.md: Initial Commit
- 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 is highlighted in blue.

Select Developer settings

Developer settings

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

Organization settings



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

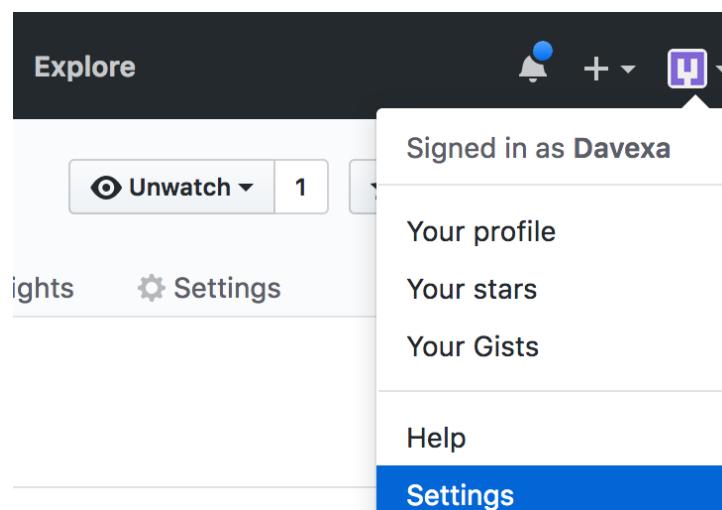
Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

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](#)

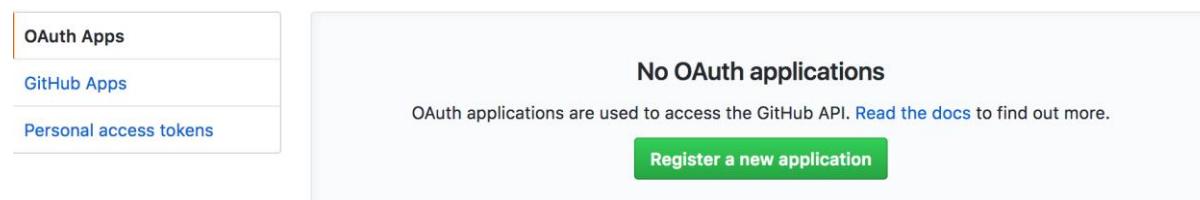


[Location](#)

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.

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

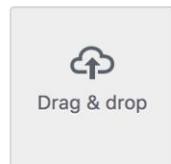
Client ID
e681fbc9e6491de43587

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

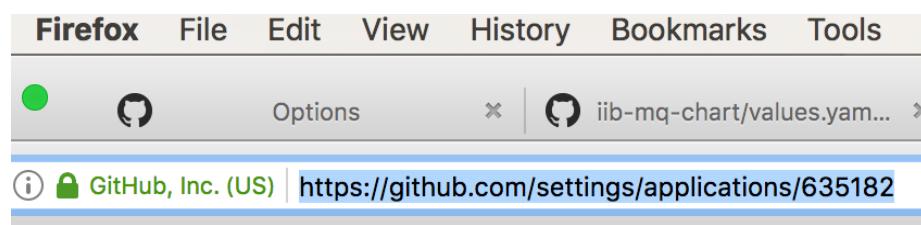
Application description

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 – Configure and Deploy Micro Service Builder

Jenkins is the key component of the Micro Service Builder Pipeline

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 ICP console. A search bar at the top contains the text 'fabric'. Below it, a table lists one item: 'ibm-msb-fabric' in the NAME column, 'developers' in the NAMESPACE column, 'DEPLOYED' in the STATUS column, and 'Jan 8th 2018' in 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 of 1.

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.

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' > 'PersistentVolume' section of the ICP console. A search bar at the top contains the text 'jen'. Below it, a table lists one item: 'jenkins-pv1' in the NAME column, 'NFS' in the TYPE column, '8Gi' in the CAPACITY column, 'RWO' in the ACCESS MODE column, 'Recycle' in the RECLAIM POLICY column, and 'Available' in the STATUS column. There are dropdown menus for items per page (set to 20) and a link to item 1 of 1.

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.

Configure and Deploy Micro Service Builder 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



spring-cloud-
data-flow

Microservice Builder
Pipeline

Open source,
multi-cloud toolkit for

ibm-charts

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 text and a bulleted list of tips. To the right of the modal, a notification bubble says "Use any theme with GitHub Pages". 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 "OAuth Apps" (highlighted with an orange border), "GitHub Apps", and "Personal access tokens". The main content area shows a single application entry for "icp-msb-iib-demo". It includes the application name, a "Davexa owns this application." badge, and a note about listing it in the GitHub Marketplace. Below this, there is a section for users, showing "1 user" with "Client ID" and "Client Secret" details.

Client ID
e681fb9e6491de43587

Client Secret
aef2f82c12acabd3218de28d7f63b1fa25560096

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

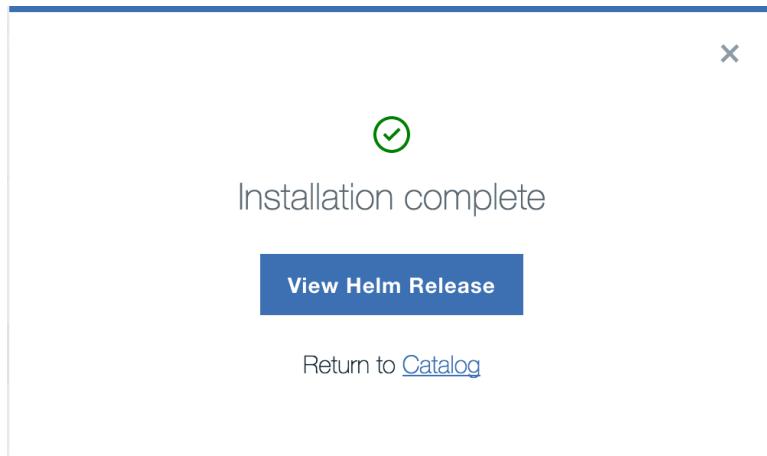
nodePort must be a port that is available, try 32050, 32051 etc

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.

Master	
Name	Image
jenkins-master	ibmcom/mb-jenkins
ImageTag	ImagePullPolicy
2.0.0	Always
ImagePullSecret	Component
Enter value	jenkins-master
Cpu	Memory
200m	256Mi
JavaOpts	ServicePort
-Xmx512m -Dfile.encoding=UTF-8 -Dhudson.security.ArtifactsPermission=true	8080
ServiceType	NodePort
NodePort	32051
HostName	ContainerPort
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 Micro Service 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		DESIRED	CURRENT	UP-TO-DATE	AVAILABLE
NAME		1	1	1	1
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
Name	ibm-msb-pipeline-da-ibm-microservicebuilder-pipeline	-	-
Namespace	default	-	-
Creation time	Jan 5th 2018 at 1:13 PM	-	-
Labels	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	-	-
Selector	component=ibm-msb-pipeline-da-jenkins-master	-	-
Replicas	1 desired 1 total 1 updated 1 available	-	-
RollingUpdateStrategy	1 max unavailable, 1 max surge	-	-
MinReadySeconds	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=e681fb9e6491de43587&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

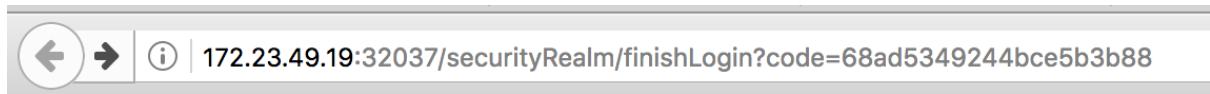
- DAVEVACOM ✓
- giphos ✓

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

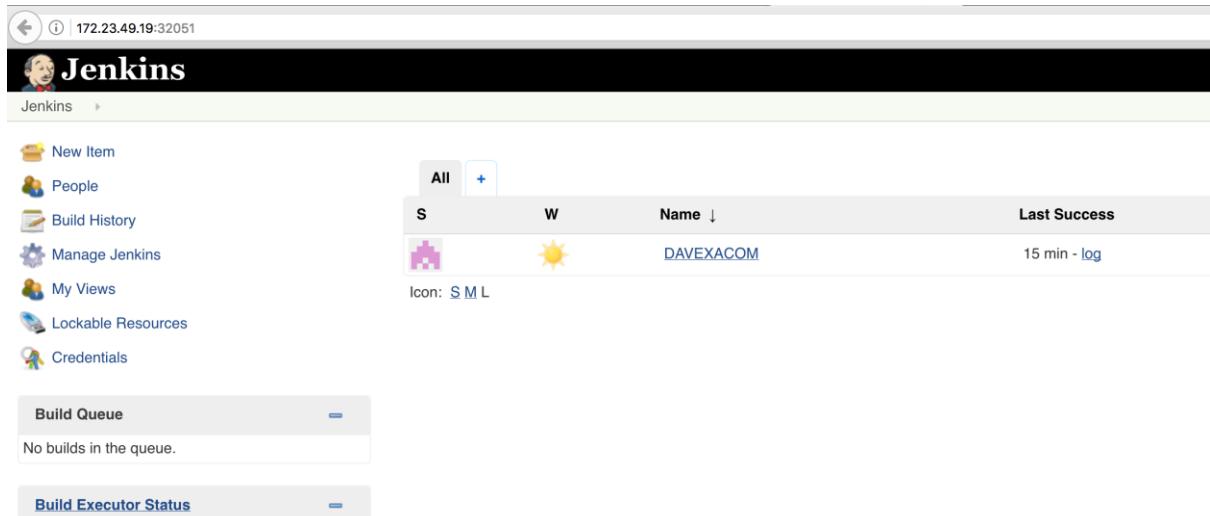
If when you hit the agree Authorize Davexa you get the following



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.

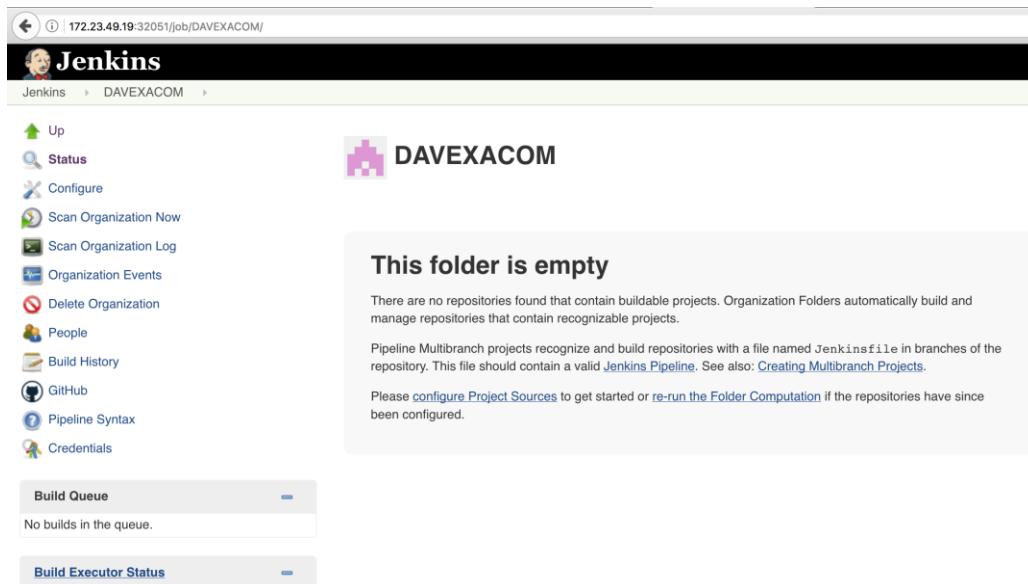


Micro Services Builder - Jenkins

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.



If all is correct it will look something like this.

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 this is a 'Build Queue' section. The main content area is titled 'DAVEXACOM' and shows a 'Repositories (1)' table. The table has columns for S (Status), W (Workstation), Name (sorted by Name), and Description. It lists one repository named 'IIB-MQ' with a status of 'S' (green) and 'W' (yellow). The description is 'IIBv10 and MQv9 docker build repository'. Below the table, it says 'Icon: S M L'.

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 manually

With no Github webhook in place at this point, 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
blockchain-network	default	DEPLOYED	Nov 28th 2017	ibm-blockchain-network
cam	default	DEPLOYED	Nov 27th 2017	ibm-cam-prod
cardashboard-msb-pipeline	default	DEPLOYED	Dec 7th 2017	cardashboard
daiibmq	default	DEPLOYED	Dec 12th 2017	iib-mq
db2server1	default	DEPLOYED	Nov 27th 2017	ibm-db2oltp-dev
ibm-msb-pipeline-da	default	DEPLOYED	Dec 13th 2017	ibm-microservicebuilder-pipeline
ibm-msb-pipeline	default	DEPLOYED	Dec 6th 2017	ibm-microservicebuilder-pipeline
ibm-msbf	default	DEPLOYED	Nov 28th 2017	ibm-microservicebuilder-fabric
icp-dp7	default	DEPLOYED	Dec 4th 2017	ibm-datapower-dev
icpmbsbdemo	default	DEPLOYED	Dec 6th 2017	icpmbsbdemo
iib-icp-100010	default	DEPLOYED	Dec 4th 2017	ibm-integration-bus-dev
iib10mq9	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

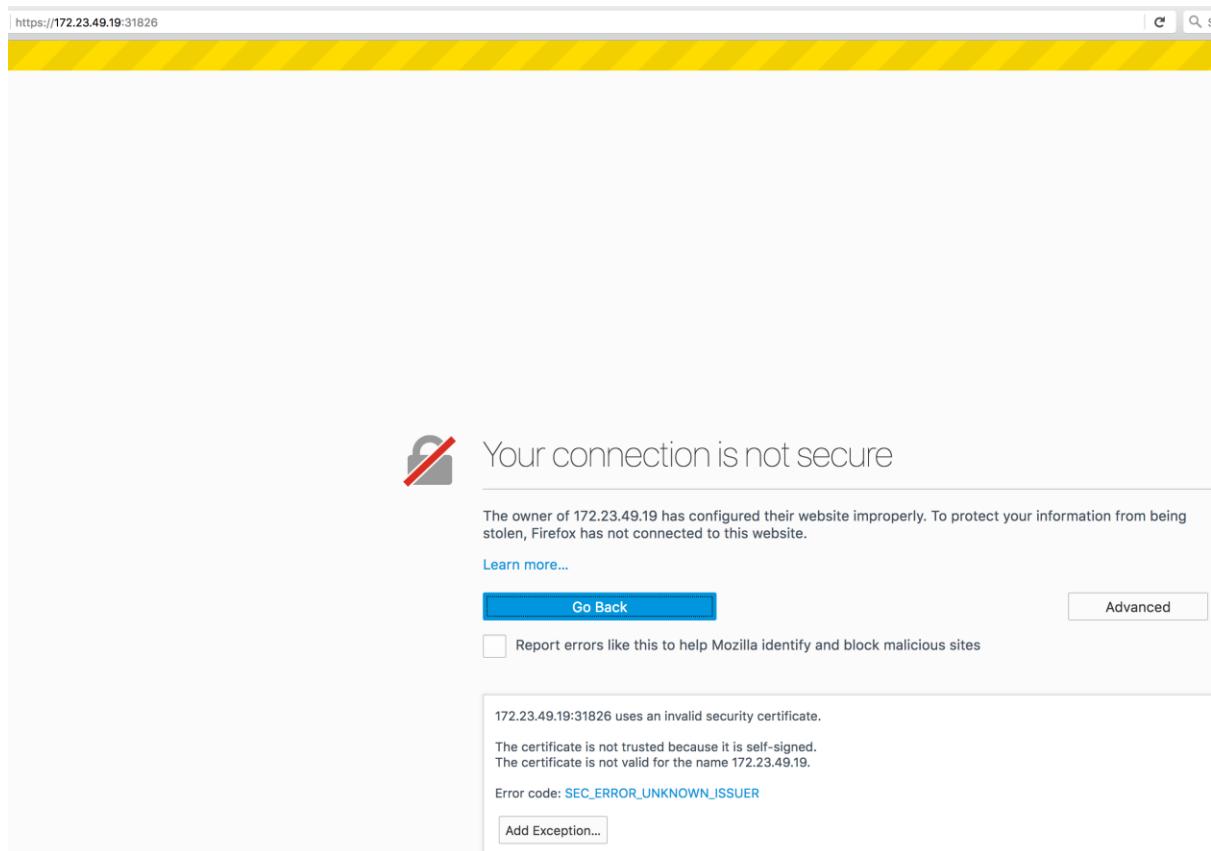
Overview

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

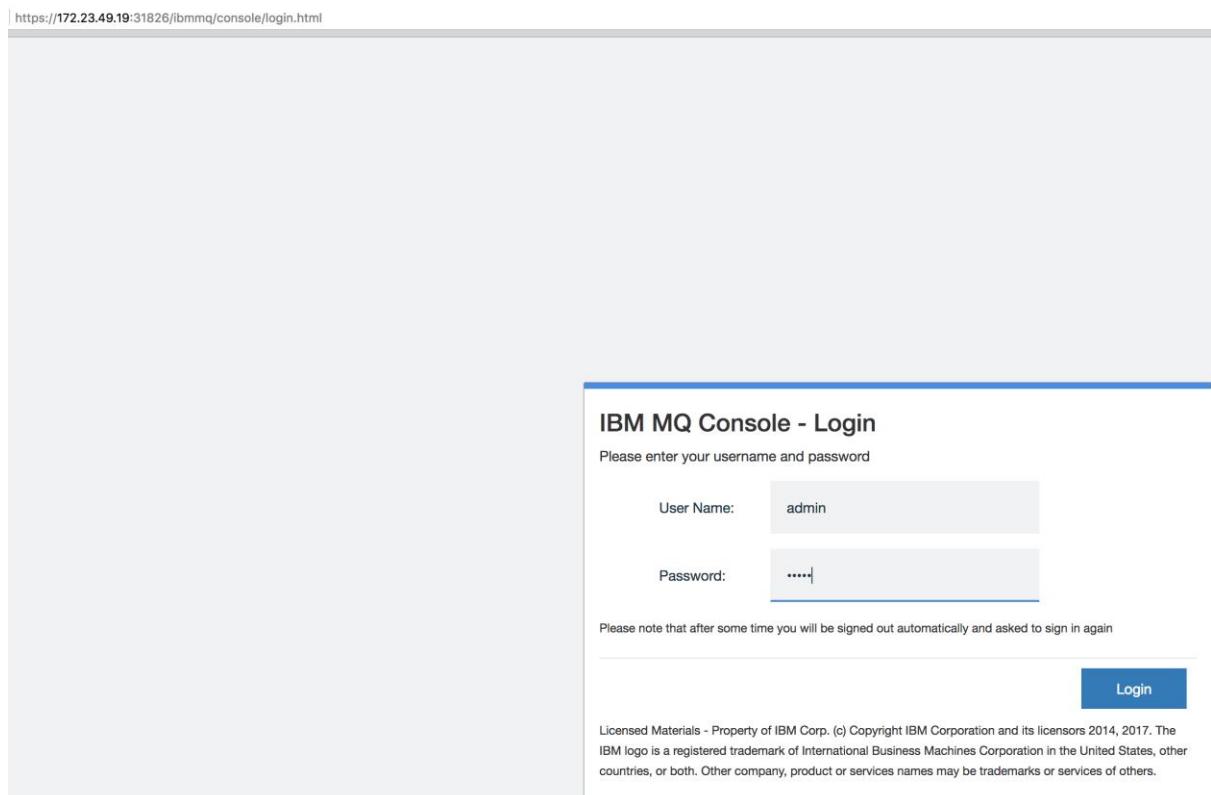
Select the iib-mq-web 31826/TCP link.

In the browser add the https:// up front

Add certificate



Log into MQ console with admin/passw0rd



IBM MQ Console

IBM MQ Container +

Queue Manager

Name	Status
icpDA1	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 iib-mq-console 30212/TCP iib-mq-nodelistener 30505/TCP iib-mq-serverlistener 30395/TCP

172.23.49.19:30212/#broker/0

Integration

Filter Options...

icnode1 - Integration Node

Overview Statistics

Quick View

Node Name	icnode1
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	

The screenshot shows the ICP (IBM Content Platform) console interface. The left sidebar has a tree view with 'icnode1' expanded, showing 'Servers', 'Operational Policy', 'Data', 'Security', 'Monitoring', and 'Business'. The main content area is titled 'icnode1 - Integration Node' and shows tabs for 'Overview' and 'Statistics'. Under 'Overview', there are three sections: 'Quick View' (with fields like Node Name, Version, Admin Security, Run Mode), 'Advanced Properties' (with fields like Platform Name, Fixpack Capability, Operation Mode, etc.), and 'Component Properties' (with fields like Security Cache settings and Cache Manager policy). The URL in the address bar is 172.23.49.19:30212/#broker/0.

Test the deployed message flow

From the ICP Console identify the IIB server listener port for HTTP connections

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 iib-mq-console 30212/TCP iib-mq-nodelistener 30505/TCP iib-mq-serverlistener 30395/TCP

In this example 7800 (the IIB HTTP listener port) 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

Headers

Data

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

Custom

Enter the data and its corresponding MIME type below.

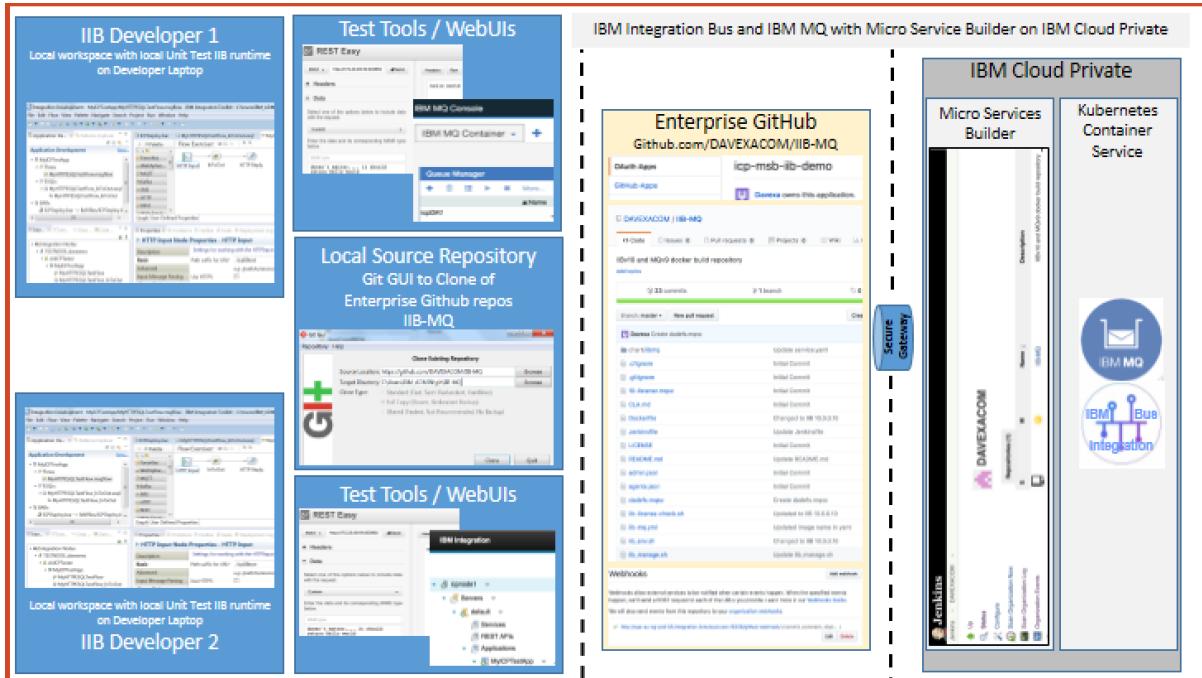
MIME type

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

Authentication

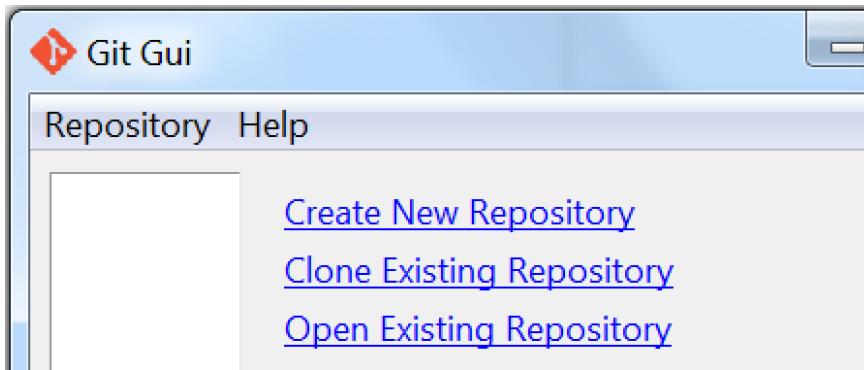
Automating the Build end to end

Solution Overview Diagram

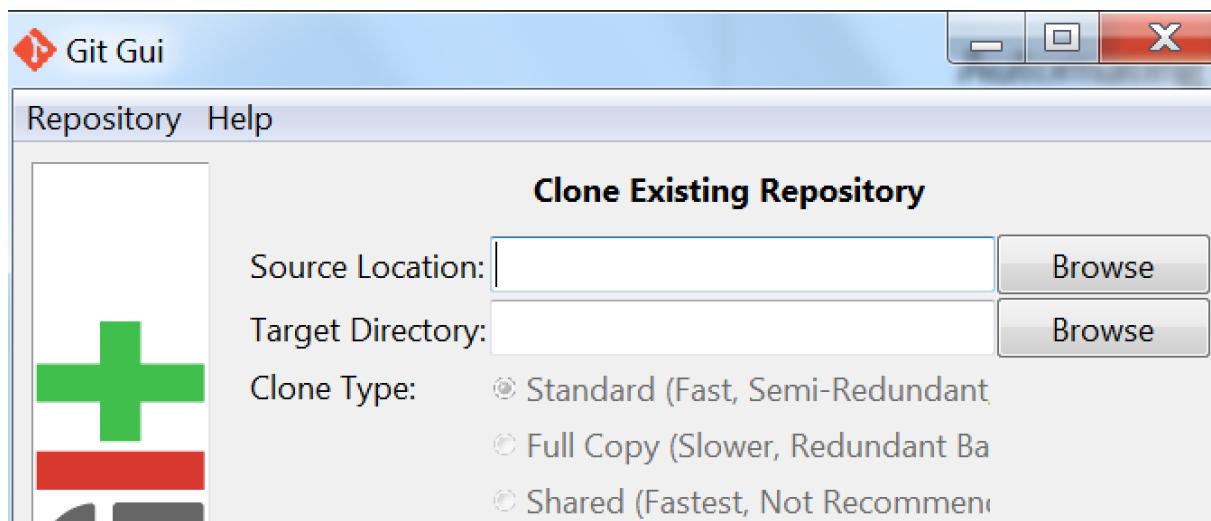


Creating a local clone of the GitHub Repos IIB-MQ

Example using GitGUI

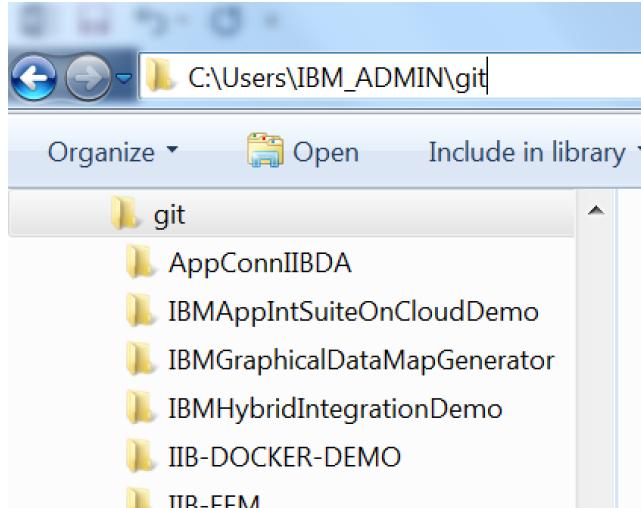


Select Clone Existing Repository

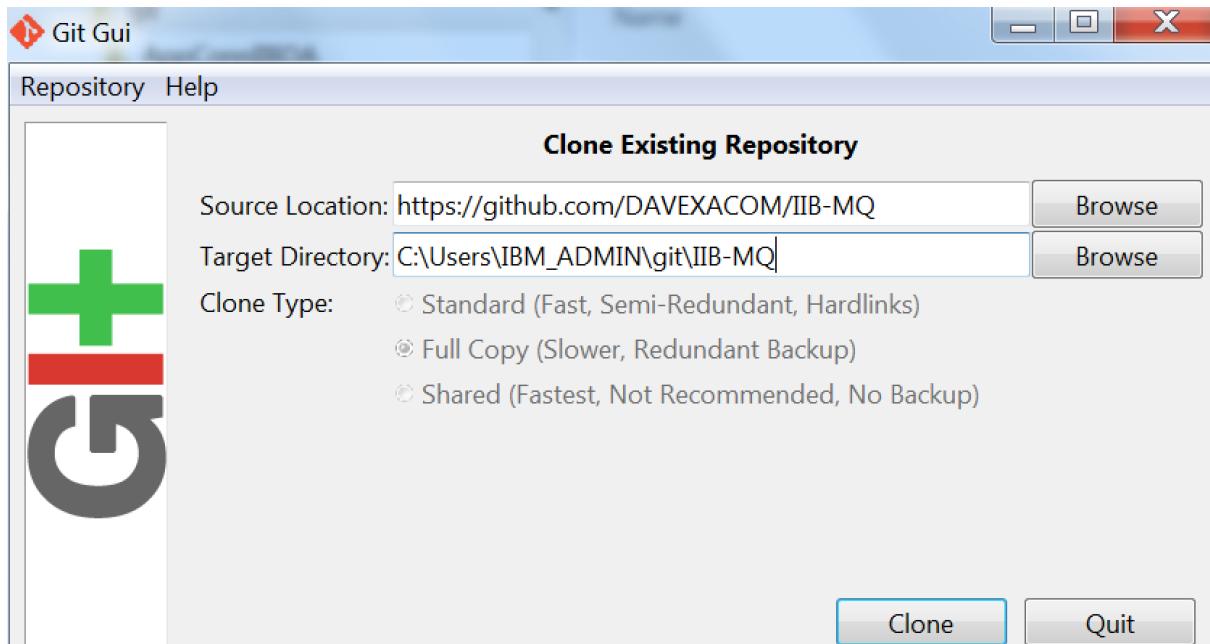


Cut and paste the URL from the GitHub repository to the Git Gui Source Location

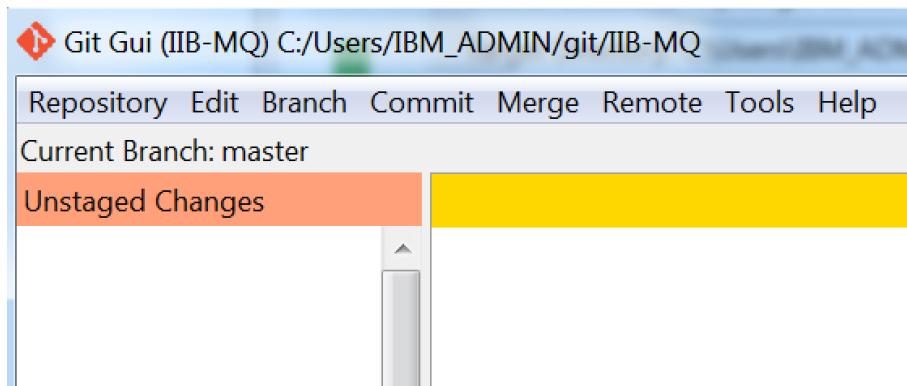
Put and paste the directory path to your local Git Client to the Target Directory... for example



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 a Private Network

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 the configuration details for a secure gateway named "jenkins_32051".
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: DO NGUYEN at 09/01/2018, 2:14:09 pm
Last modified by: DO NGUYEN at 09/01/2018, 2:14:55 pm
Security: Protocol: TCP
Buttons at the bottom: Edit (green), Disable (grey), 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>

The screenshot shows the Jenkins home page with the URL 172.23.50.111:32051. It features the Jenkins logo and navigation links like Jenkins, New Item, and Help.

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
eWQlCJpYXQiOjE1MTE4NDMyOTksImV4cCI6MTUxOTIxOTI5OX0.Iy0YWhnAg-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.

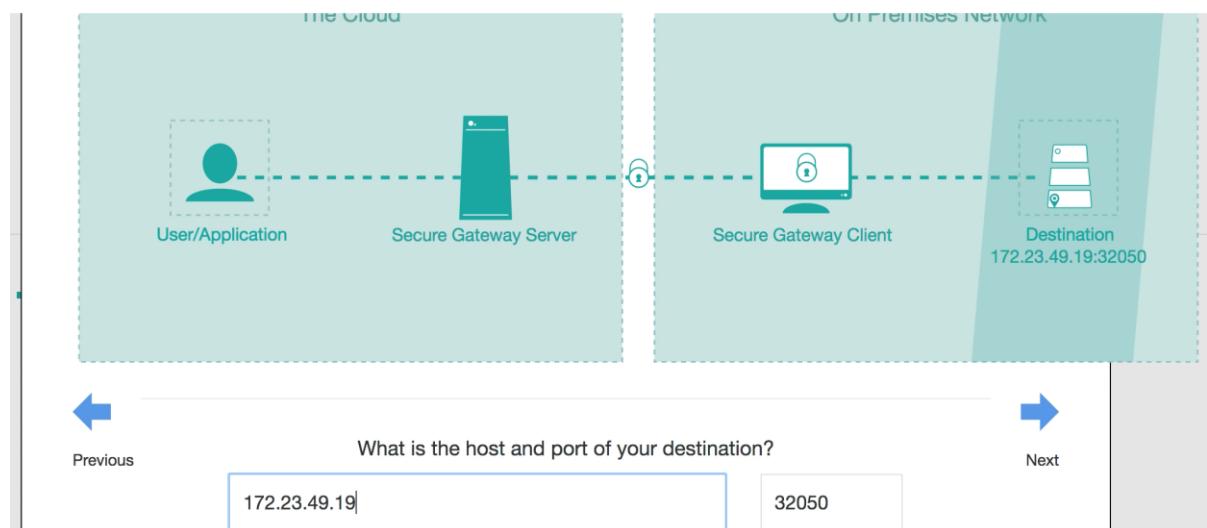
For client documentation, please view the ReadMe located at:
.rpm and .deb installers: /opt/ibm/securegateway/docs/
.dmg installer: <installation location>/ibm/securegateway/docs/
.exe installer: <installation location>\Secure Gateway Client\ibm\securegateway\docs\
*****
<press enter for the command 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

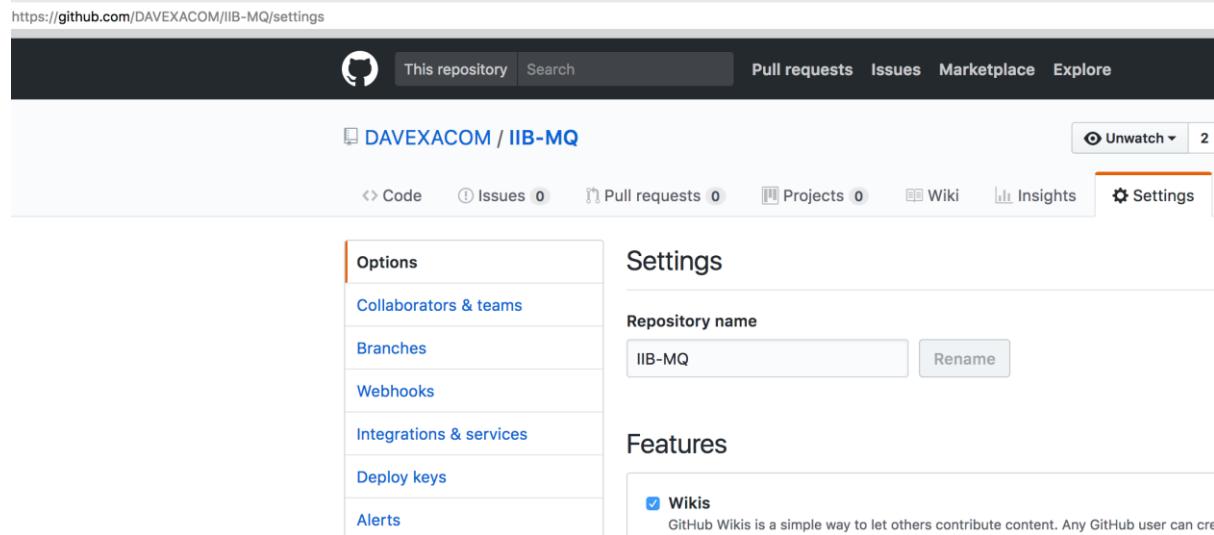
Secure Gateway-lv

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



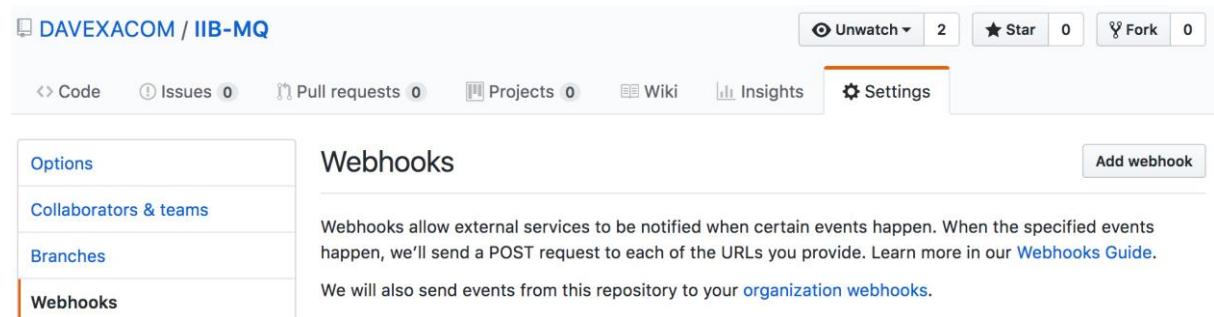
Setting up the Webhook in Github

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



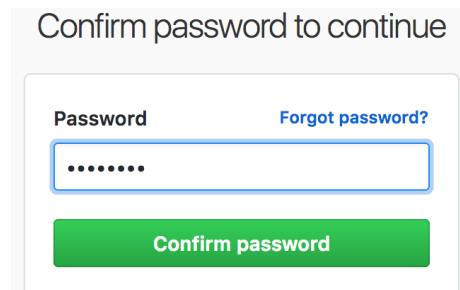
The screenshot shows the GitHub repository settings for 'DAVEXACOM / IIB-MQ'. The left sidebar has 'Webhooks' selected under 'Options'. The main area is titled 'Settings' with a 'Repository name' field containing 'IIB-MQ' and a 'Rename' button. Below it is a 'Features' section with a checked 'Wikis' checkbox and a descriptive note. A 'Wikis' icon is also present.

Select Webhooks->Add Webhooks



The screenshot shows the 'Webhooks' section of the GitHub repository settings. The left sidebar has 'Webhooks' selected under 'Options'. The main area is titled 'Webhooks' with an 'Add webhook' button. It contains a general description of what webhooks are and how they work, along with a note about organization webhooks.

Confirm password if prompted



A modal dialog box titled 'Confirm password to continue'. It has a 'Password' input field containing '*****' and a 'Forgot password?' link. Below the input field is a green 'Confirm password' button.

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 table for 'Repositories (1)'. There is one entry: 'IIB-MQ' with status 'S' (green), 'W' (yellow), and a sun icon. The 'Description' column indicates it's a 'IIBv10 and MQv9 docker build repository'. Below the table, it says 'Icon: S M L'. At the bottom, there are sections for 'Build Queue' (empty) and 'Build Executor Status'.

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 left sidebar shows 'Application Development' with projects like 'MyICPTestApp', 'Flows', 'ESQLs', 'BARs', and 'Independent Resources'. The main editor window is titled 'MyHTTPESQLTestFlow.msgflow' and contains ESQL code:

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

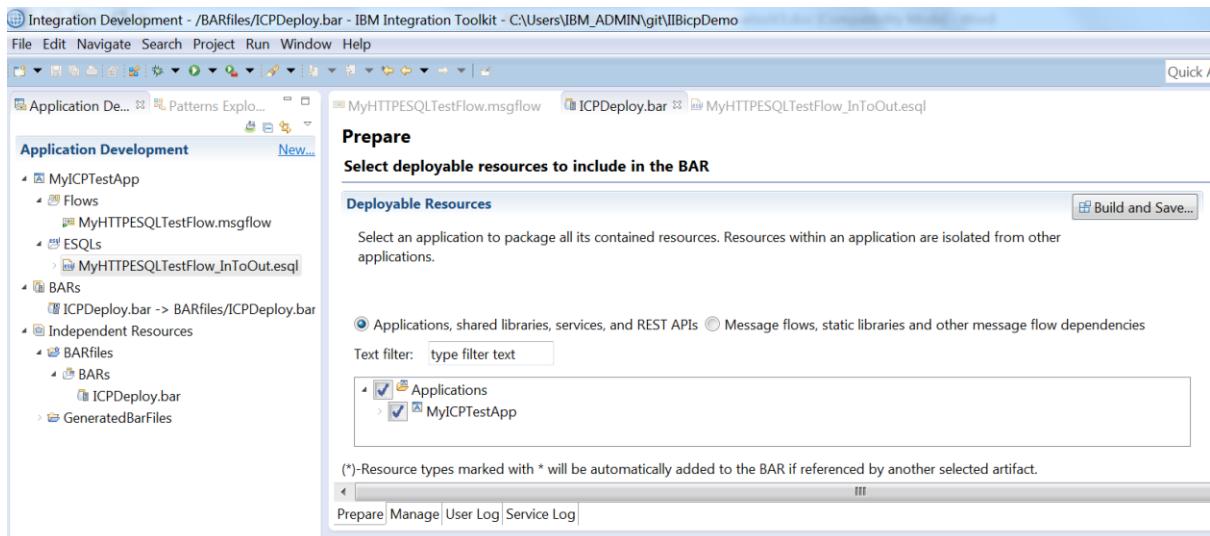
      DECLARE mystring CHARACTER;

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

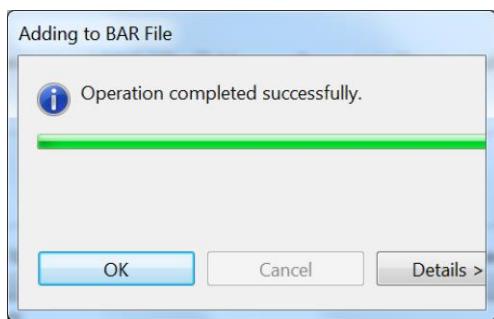
      RETURN TRUE;
    END;

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

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



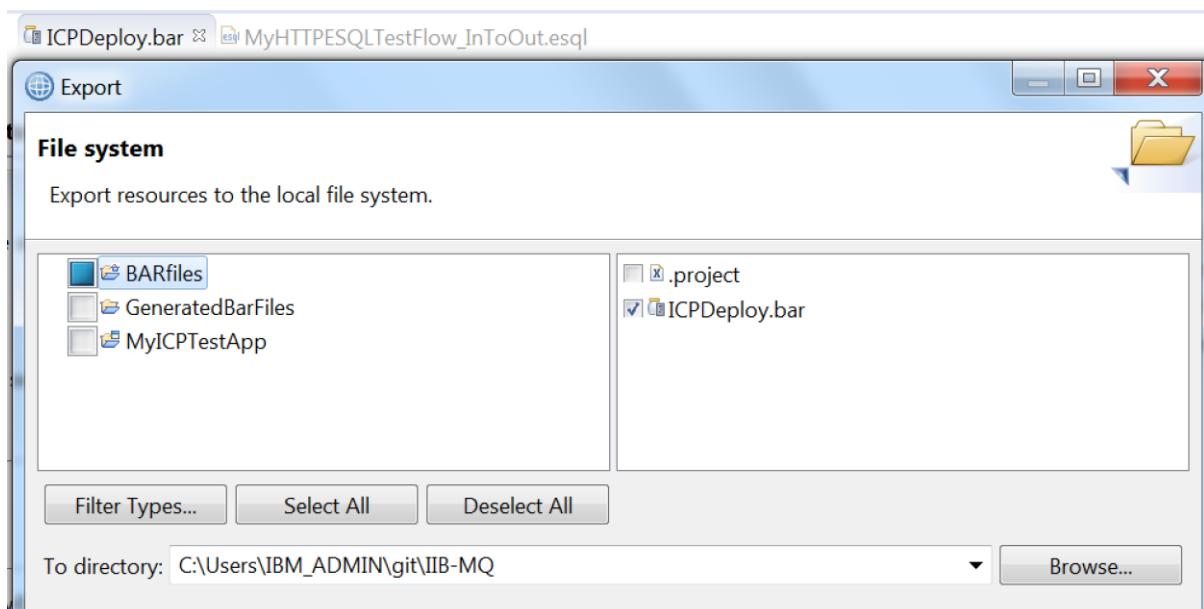
Build and save the BAR file.



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

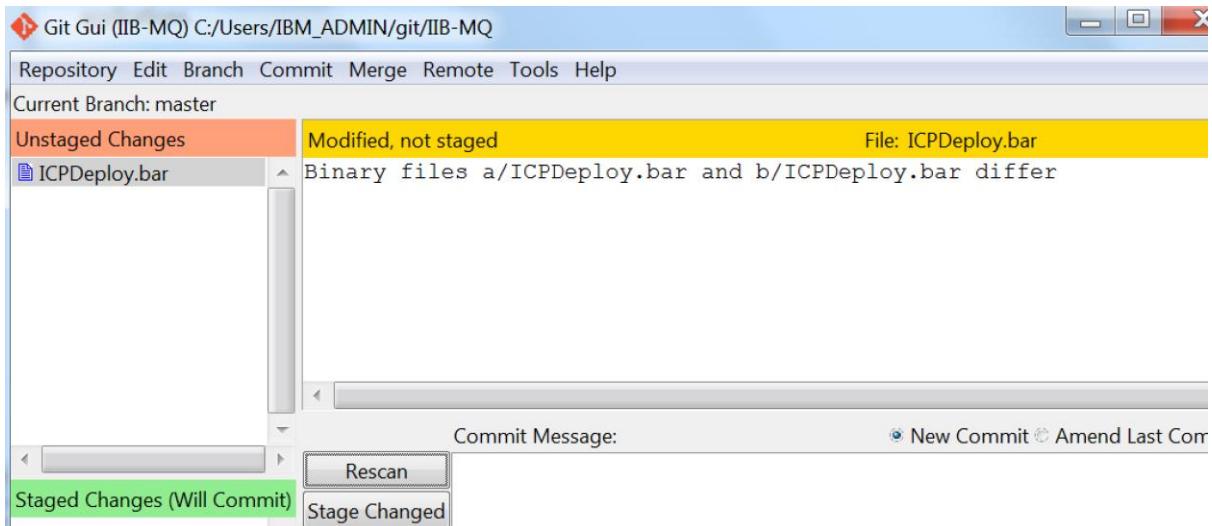
Export the updated BAR file to the File System. File->Export->File System



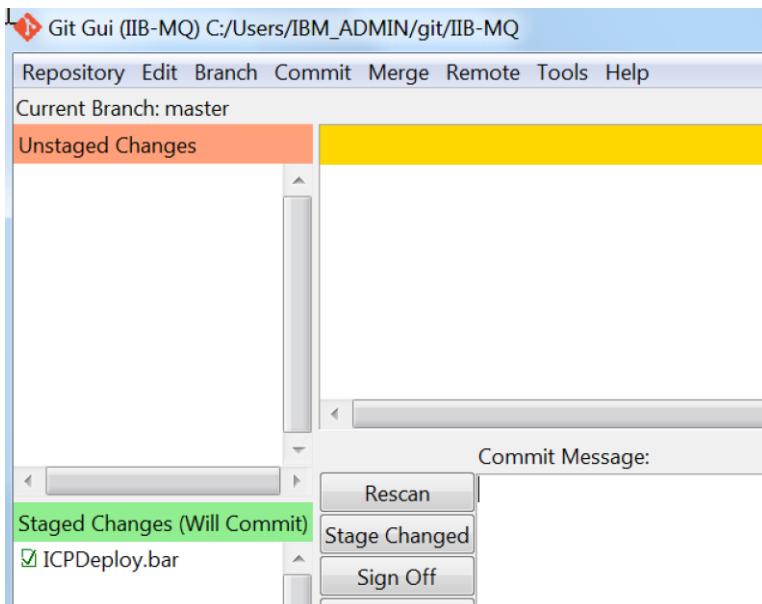
Set the target directory to your local clone of GitHub IIB-MQ repository

Push to GitHub IIB-MQ repository

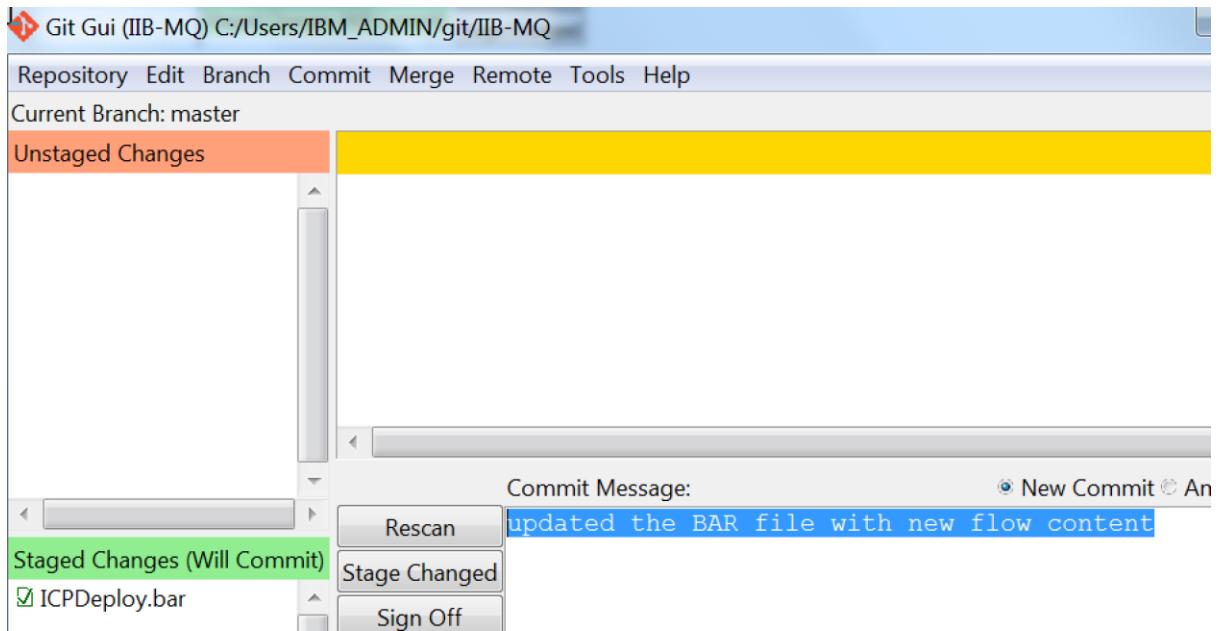
In Git GUI or equivalent hit rescan



Stage changed

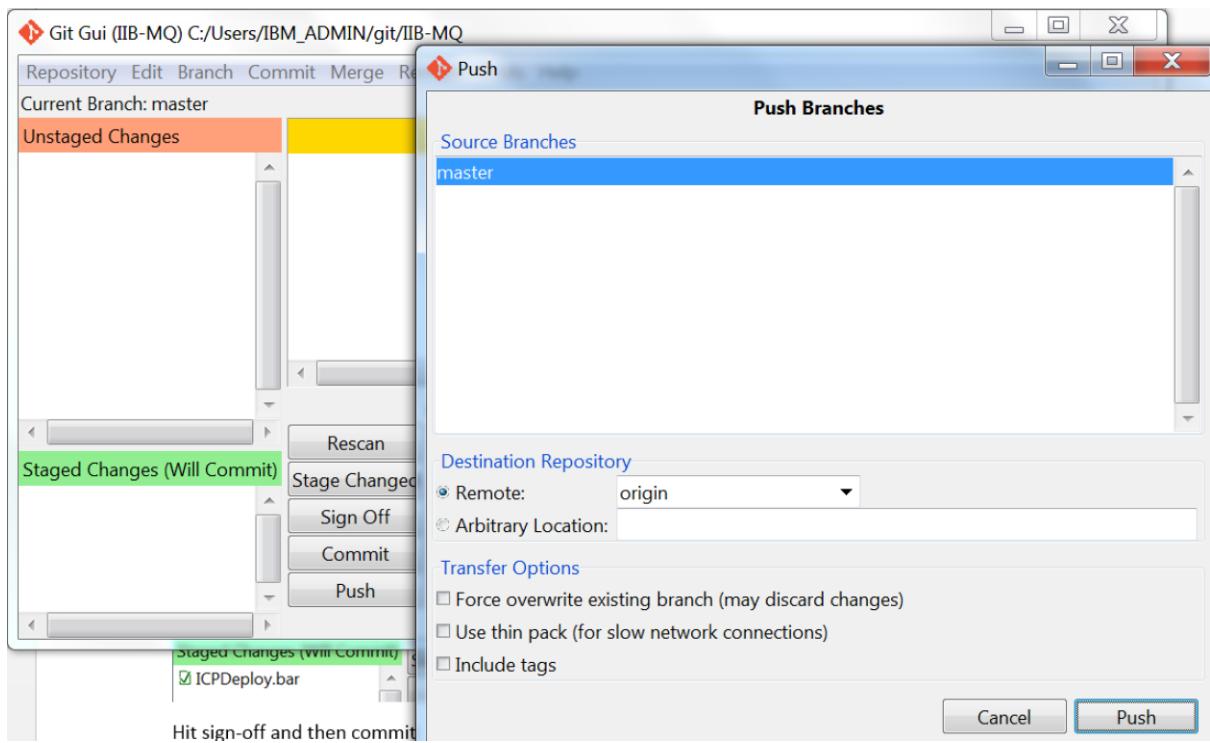


Add a commit message

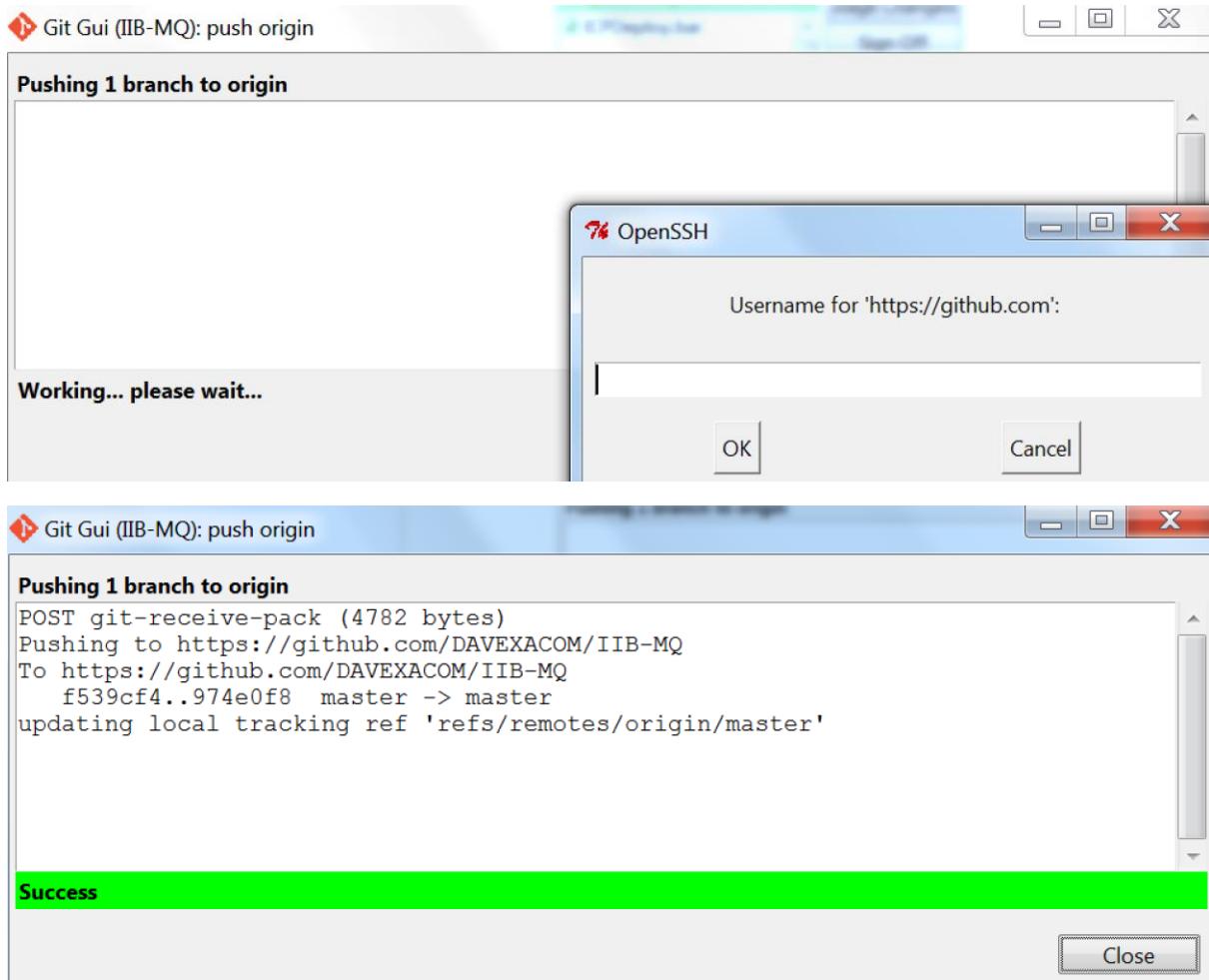


Hit sign-off and then commit

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. The table has columns for S (Status), W (Workload), Name (IIB-MQ), and Description (IIBv10 and MQv9 docker build repository). An icon for the repository is shown, along with links for S, M, and L.

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

This screenshot shows the same Jenkins interface as the previous one, but with a different state. The 'Build Executor Status' section now lists 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.'

Check the Jenkins build logs

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



172.23.50.11:32051/job/DAVEXACOM/job/IIB-MQ/job/master/2/console

Jenkins > DAVEXACOM > IIB-MQ > master > #2

[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	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

Expand the icpnode1->servers->default->Applications->MyICPTestApp->Message Flows

The screenshot shows the IBM Integration Bus (IIB) interface. At the top, there is a header bar with a back arrow, a help icon, and the IP address 172.23.49.19:30212/#broker/0. Below the header is a dark navigation bar with the text "IBM Integration". On the left side, there is a navigation tree:

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

Test the deployed message flow

From the ICP Console identify the IIB server listener port for HTTP connections

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 iib-mq-console 30212/TCP iib-mq-nodelistener 30505/TCP iib-mq-serverlistener 30395/TCP

In this example 7800 (the IIB HTTP listener port) 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

- 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

Headers Raw Preview

Hit Send and check the result in preview

REST Easy

POST ▾ 172.23.50.111:31207/icplIBtes ⚡ 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 matters.... it should say hello world again
```

Headers Raw Preview

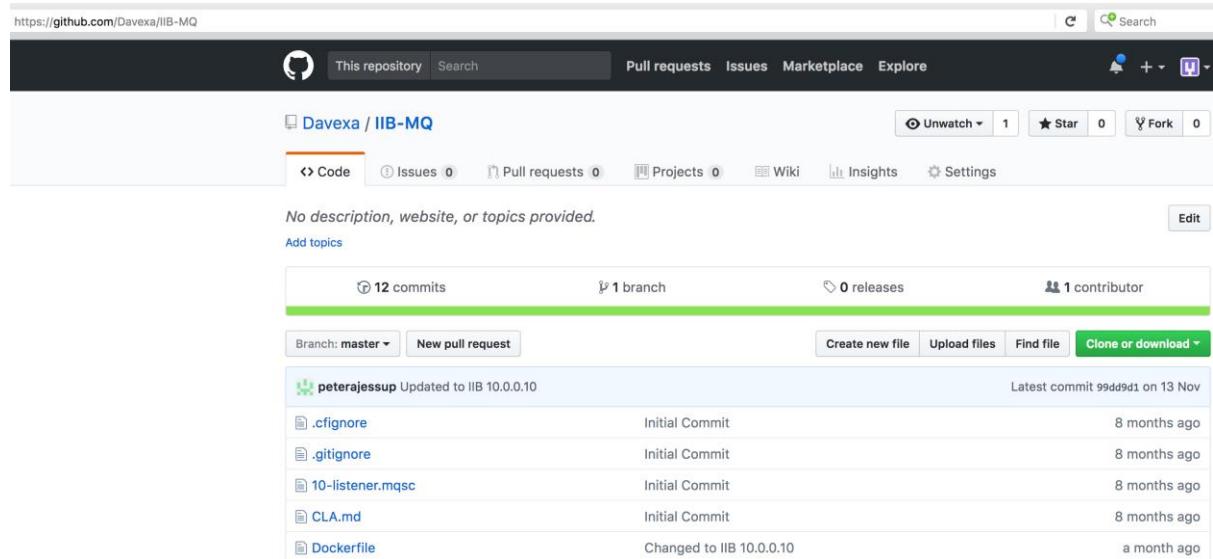
Hello world again

Reference Section

How the DAVEXACOM/IIB-MQ Repository in GitHub was created

Source GitHub Repositories to leverage

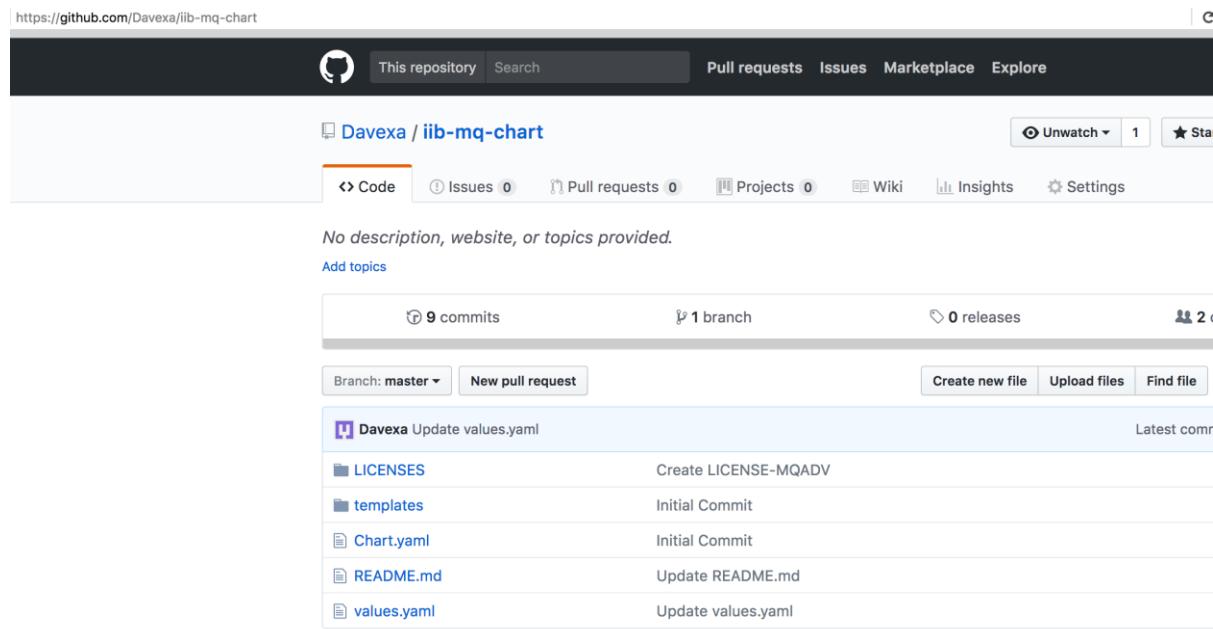
1. IIB an MQ Docker build repository



The screenshot shows the GitHub repository page for 'Davexa / IIB-MQ'. The repository has 12 commits, 1 branch, and 0 releases. The latest commit is from Nov 13, 2018. The commits are listed as follows:

File	Commit 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

2. IIB and MQ Helm Chart repository



The screenshot shows the GitHub repository page for 'Davexa / iib-mq-chart'. The repository has 9 commits, 1 branch, and 0 releases. The latest commit is from Nov 13, 2018. The commits are listed as follows:

File	Commit Message	Date
LICENSES	Create LICENSE-MQADV	Latest commit
templates	Initial Commit	
Chart.yaml	Initial Commit	
README.md	Update README.md	
values.yaml	Update values.yaml	

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

<https://github.com/Davexa/IIB-MQ>

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

Your existing repository

DAVEXACOM/IIB-MQ

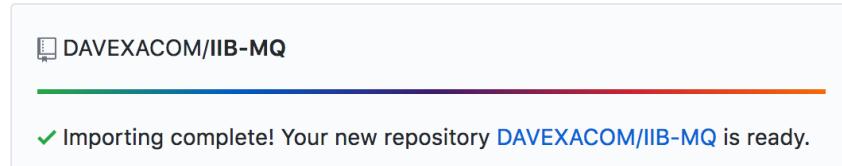
[Change repository](#)

[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.



The GitHub repository page for DAVEXACOM / IIB-MQ. The repository name is at the top. Below it, there's a header with "Code", "Issues 0", "Pull requests 0", "Projects 0", "Wiki", "Insights", and "Settings". A green progress bar spans across the top of the main content area. The main content shows "12 commits", "1 branch", "0 releases", and "1 contributor". A commit list follows, starting with a commit from peterajessup updating the repository to IIB 10.0.0.10. Other commits listed include .cignore, .gitignore, 10-listener.mqsc, CLA.md, Dockerfile, LICENSE, README.md, admin.json, and aempty.json.

File	Type	Message	Time
.cignore	Initial Commit	Updated to IIB 10.0.0.10	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
aempty.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

The GitHub repository page for DAVEXACOM / IIB-MQ. The main content area shows "12 commits", "1 branch", and "0 releases". Below this, there are buttons for "Branch: master", "New pull request", and "Create new file". A "Create new file" dialog box is open, prompting the user to "Name the file Jenkinsfile".

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 buttons 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'.

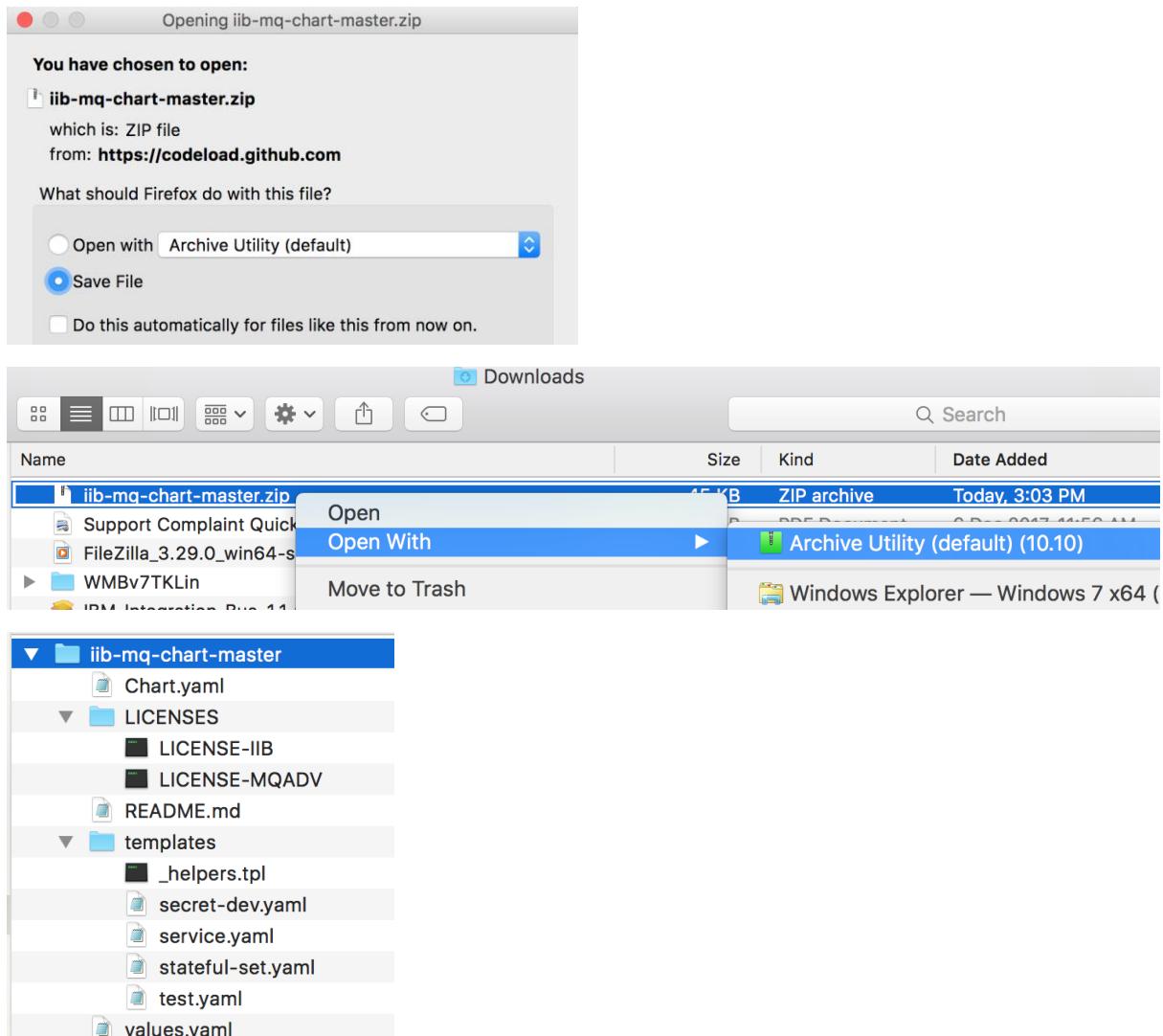
<https://github.com/DAVEXACOM/IIB-MQ/new/master/chart/iibmq/LICENSES>

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

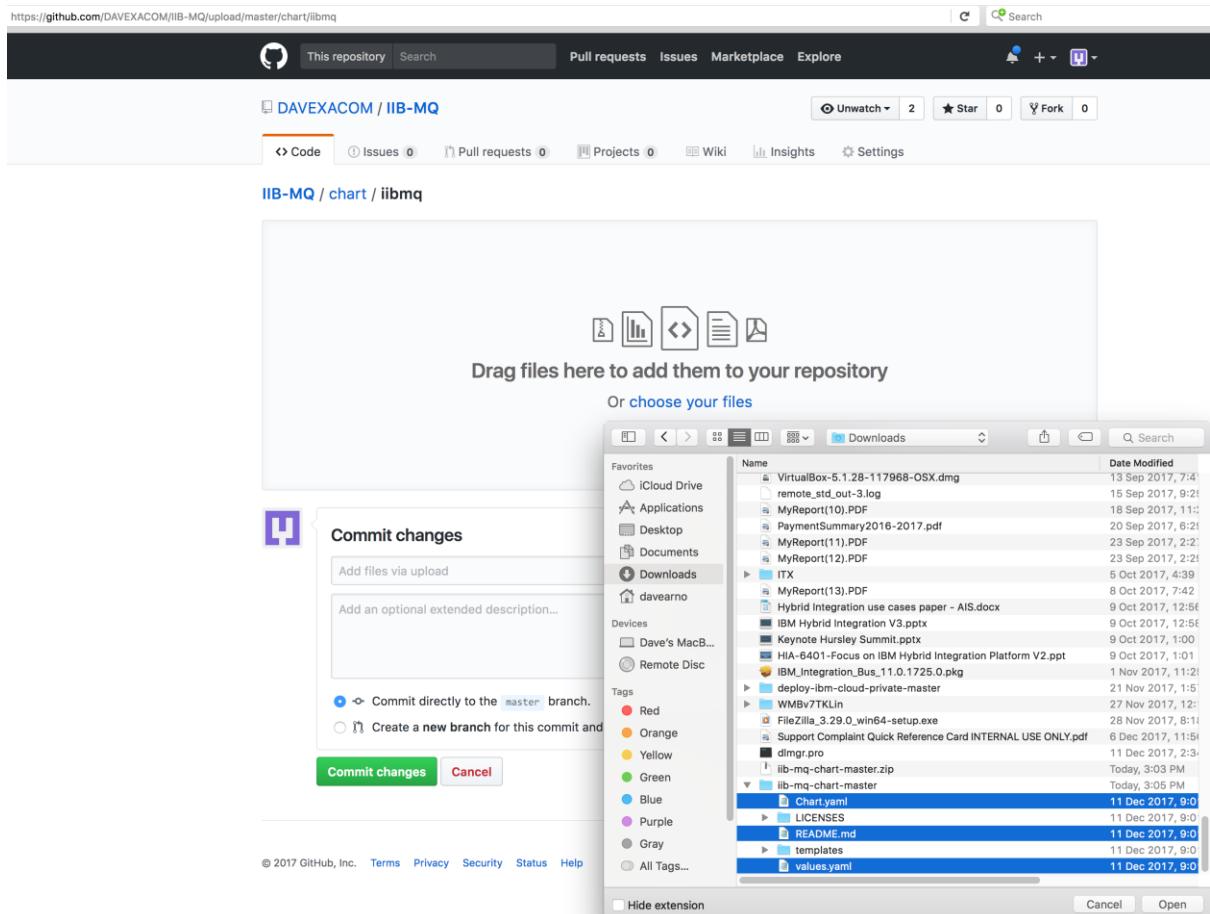
<https://github.com/Davexa/iib-mq-chart>

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'.

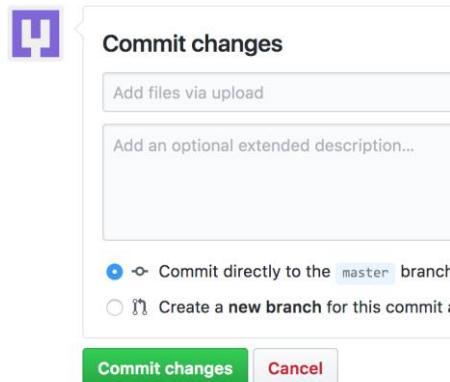


Import to from the iib-mq-chart repository to the local file system

The screenshot shows a GitHub repository page for 'DAVEXACOM / IIB-MQ / tree / master / chart / iibmq'. The repository has 2 stars and 0 pull requests. The code tab is selected. The repository contains a single commit by 'Davexa' titled 'Create LICENSE-MQADV' and a file named 'test.yaml'.



Commit changes



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 GitHub repository page. The repository is 'DAVEXACOM / IIB-MQ' and the path is 'IIB-MQ / chart / iibmq / LICENSES'. The commit dialog has a file browser window showing 'iib-mq-chart-master' folder contents: 'Chart.yaml', 'LICENSES' (selected), 'LICENSE-IIB', and 'LICENSE-MQADV'. Below the file browser are two files: 'LICENSE-IIB' and 'LICENSE-MQADV'. The commit changes form has a purple 'Commit changes' icon. It includes fields for 'Add files via upload' and 'Add an optional extended description...', and two radio button options: 'Commit directly to the master branch.' (selected) and 'Create a new branch for this commit and ...'. At the bottom are 'Commit changes' and 'Cancel' buttons.

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 button at the top says 'Branch: master ▾'. Below it, a card for 'Davexa Create Jenkinsfile' is shown, along with '1 contributor'. The Jenkinsfile content is displayed in a code block:

```
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 button at the top says 'Branch: master ▾'. Below it, a card for 'Davexa Update Jenkinsfile' is shown, along with '1 contributor'. The Jenkinsfile content is displayed in a code block:

```
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 }
```

