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

V1.0

Contents

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

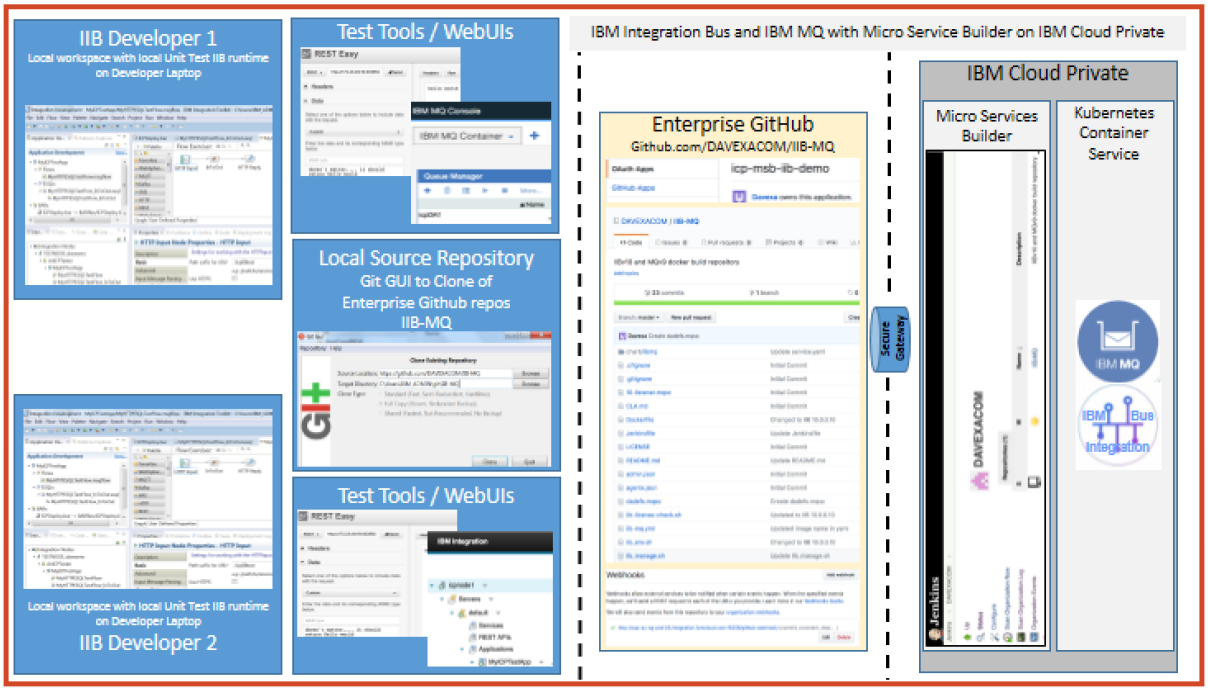
During the construction of this document the ICP instance in the Sydney IIC Pure Application system was upgraded and rebuilt. Therefore, IP address and port numbers in screen shots will have changed. For example

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:*portnumber*	172.23.50.111:*portnumber*

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.

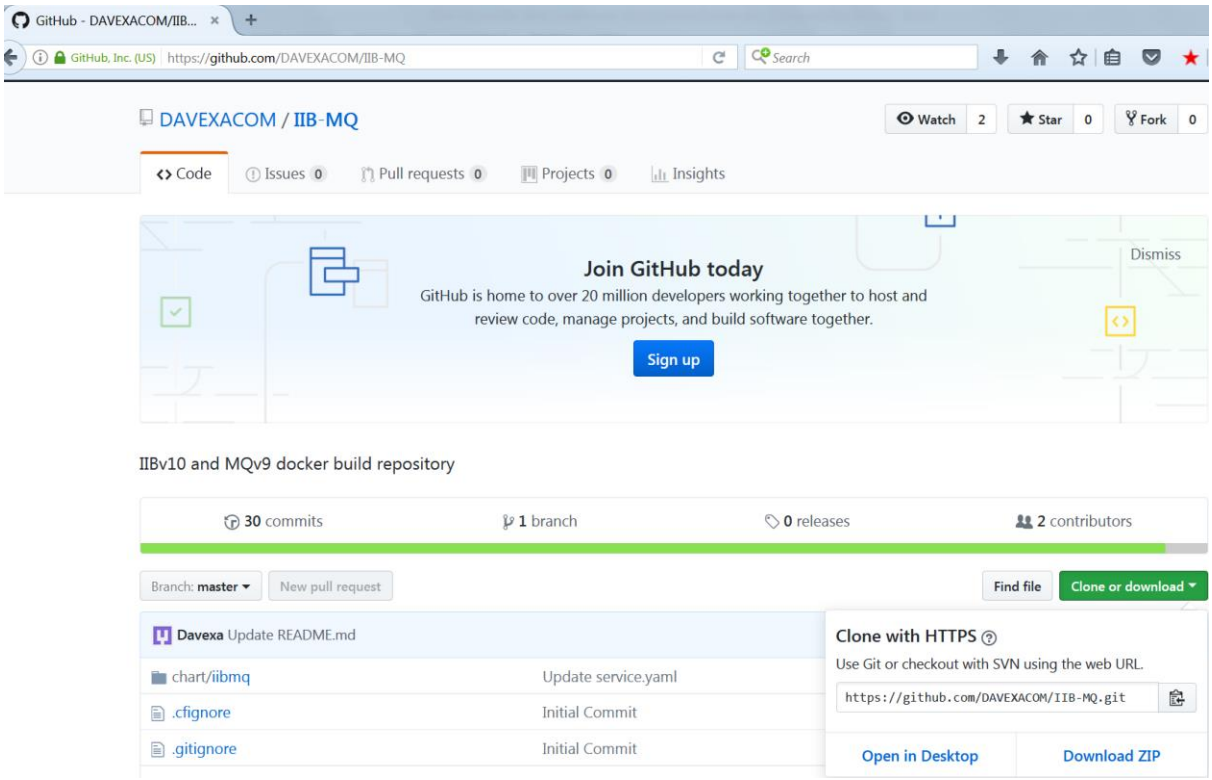
IIB & MQ Container on ICP from Github repository via MSB pipeline – Important Parameters

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	23a8537c82a19a358fb957bbfa7fba3d369eeb6b (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

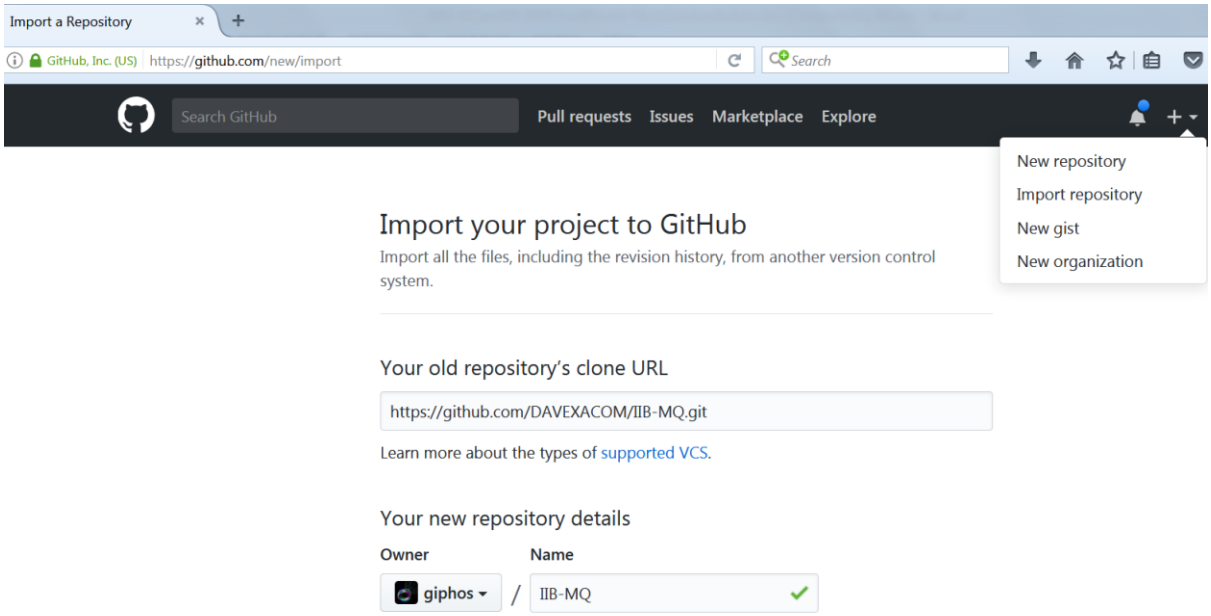
Clone the DAVEXACOM\IIB-MQ GitHub Repository

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



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



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

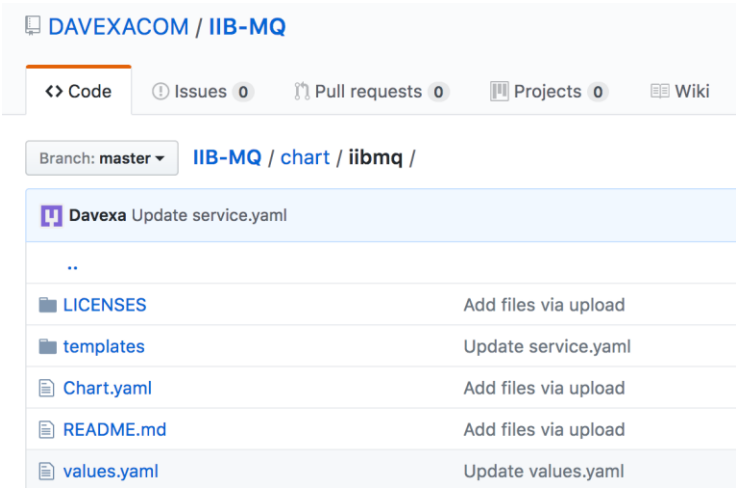
Exploring the Github repository IIB-MQ

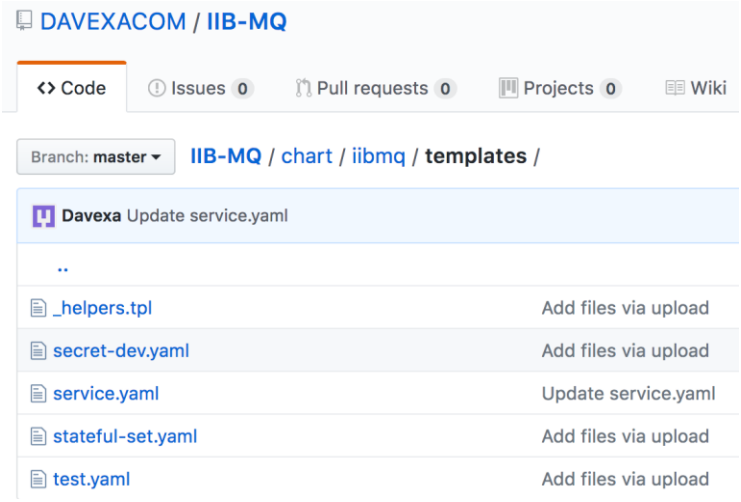
The Jenkins File

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.

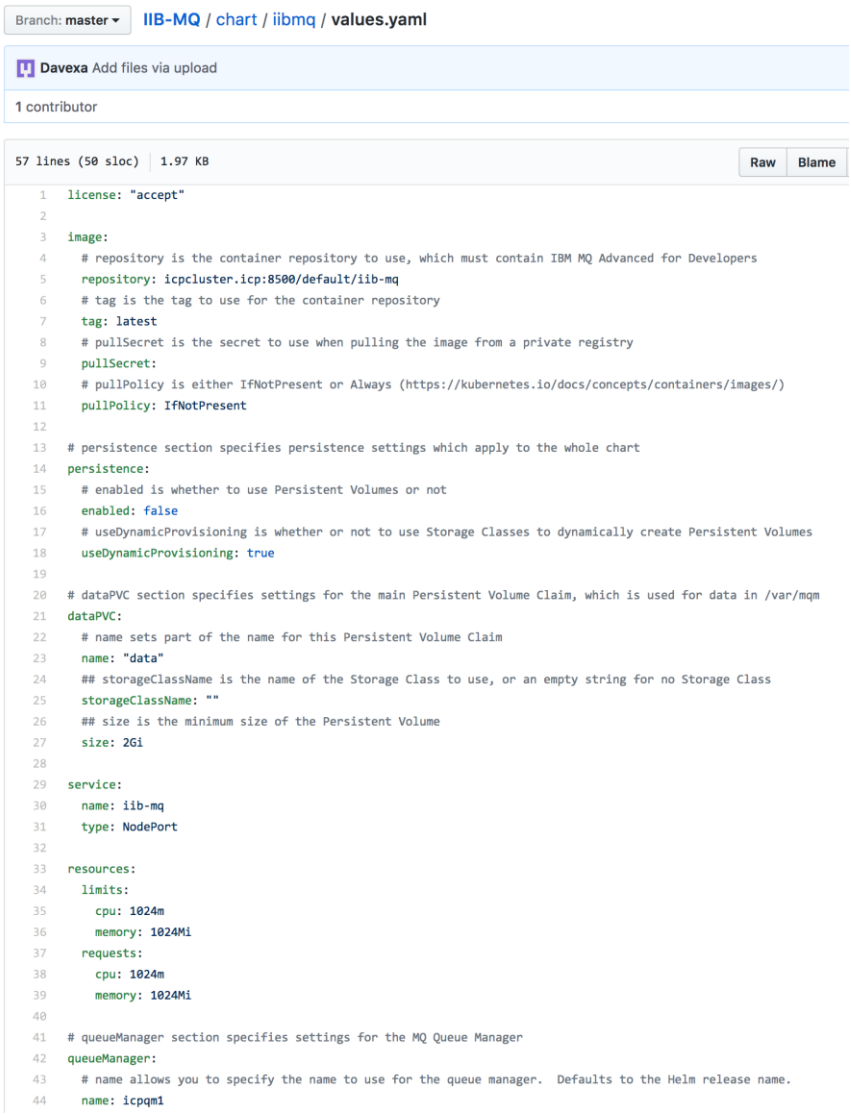
The Helm Chart files





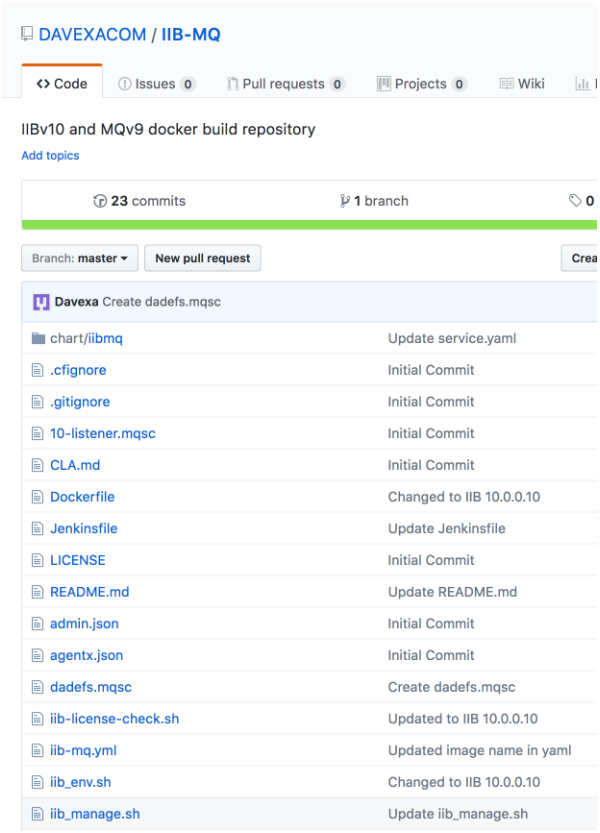
Values.yaml

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.




```
license: "accept"

image:
  # repository is the container repository to use, which must contain IBM MQ Advanced for Developers
  repository: icpcluster.icp:8500/default/iib-mq
  # tag is the tag to use for the container repository
  tag: latest
  # pullSecret is the secret to use when pulling the image from a private registry
  pullSecret:
  # pullPolicy is either IfNotPresent or Always (https://kubernetes.io/docs/concepts/containers/images/)
  pullPolicy: IfNotPresent

# persistence section specifies persistence settings which apply to the whole chart
persistence:
  # enabled is whether to use Persistent Volumes or not
  enabled: false
  # useDynamicProvisioning is whether or not to use Storage Classes to dynamically create Persistent Volumes
  useDynamicProvisioning: true

# dataPVC section specifies settings for the main Persistent Volume Claim, which is used for data in /var/mqm
dataPVC:
  # name sets part of the name for this Persistent Volume Claim
  name: "data"
  ## storageClassName is the name of the Storage Class to use, or an empty string for no Storage Class
  storageClassName: ""
  ## size is the minimum size of the Persistent Volume
  size: 2Gi

service:
  name: iib-mq
  type: NodePort

resources:
  limits:
    cpu: 1024m
    memory: 1024Mi
  requests:
    cpu: 1024m
    memory: 1024Mi

# queueManager section specifies settings for the MQ Queue Manager
queueManager:
  # name allows you to specify the name to use for the queue manager.  Defaults to the Helm release name.
  name: icpqm1
```

IBM MQ how the repository files relate to each other



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  N 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**

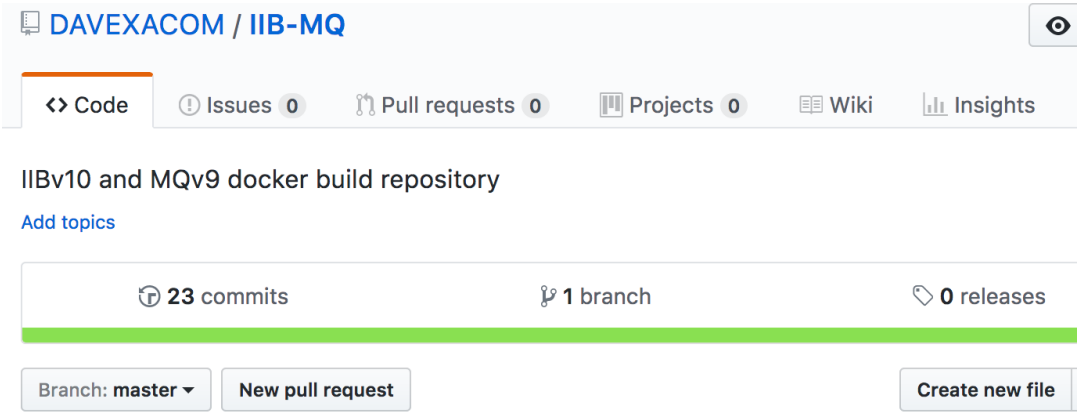
```
127   # Set entrypoint to run management script
128   ENTRYPOINT ["iib_manage.sh"]
```

Iib_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:

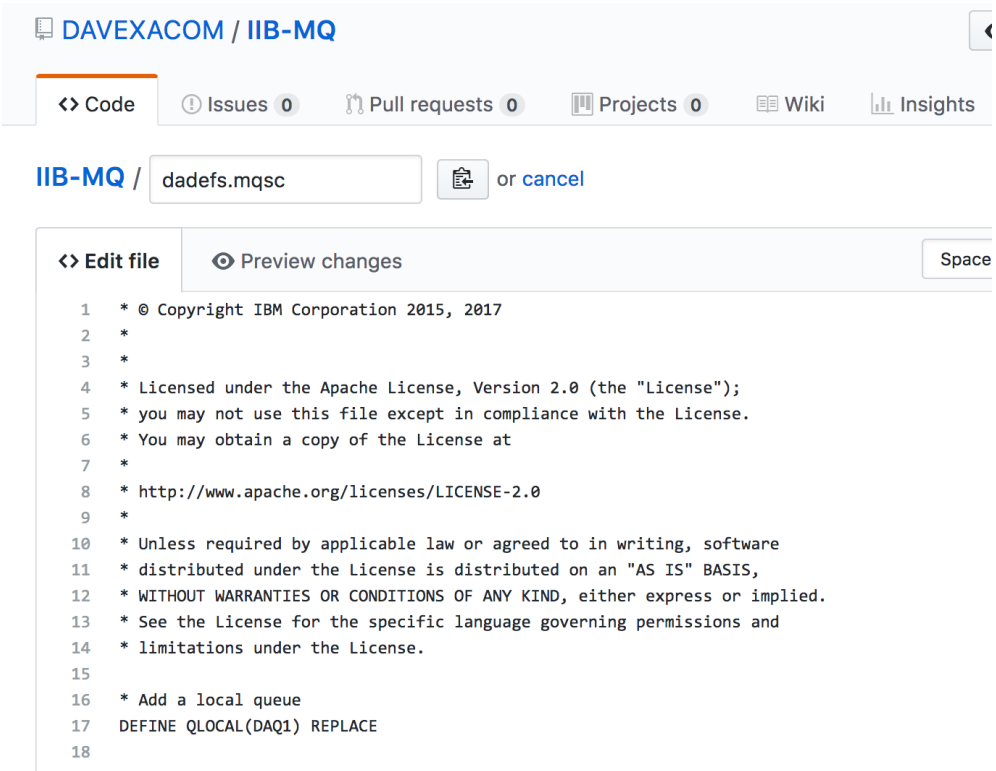
```
for MQSC_FILE in $(ls -v /etc/mqm/*.mqsc); do
   runmqsc ${MQ_QMGR_NAME} < ${MQSC_FILE}
done
```

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

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



Add your own MQ definitions and commit the changes.



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

Untested example below of the loop to deploy all BARs

for BAR_FILE in $(ls -v /etc/mqm/*.bar); do

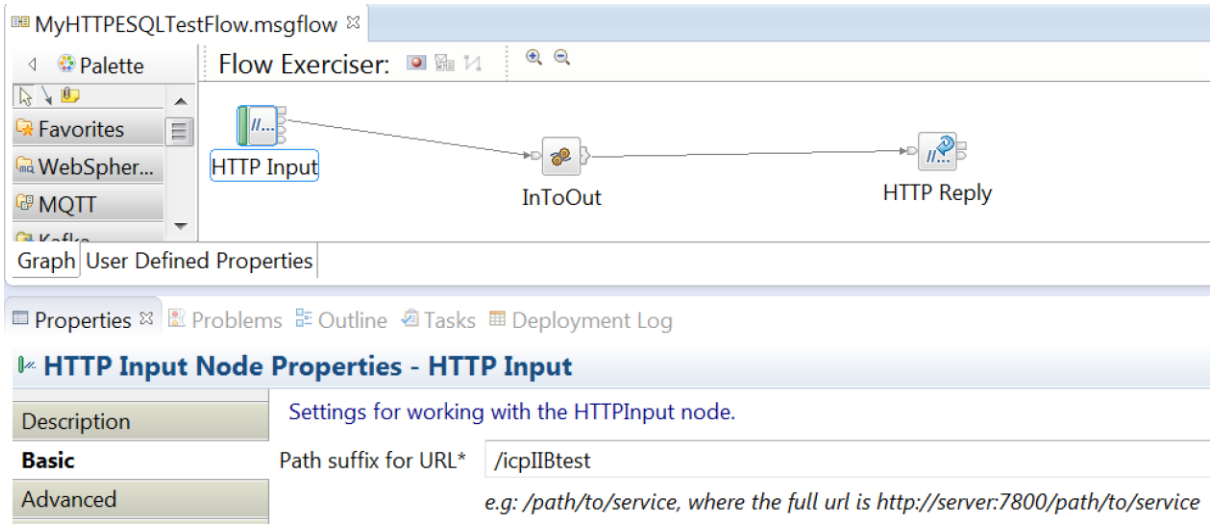
mqsideploy ${NODE_NAME} -e ${EXEC_NAME} -a ${BAR_FILE} –m

done

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

Optional Section – Create a BAR file in an IIB Toolkit

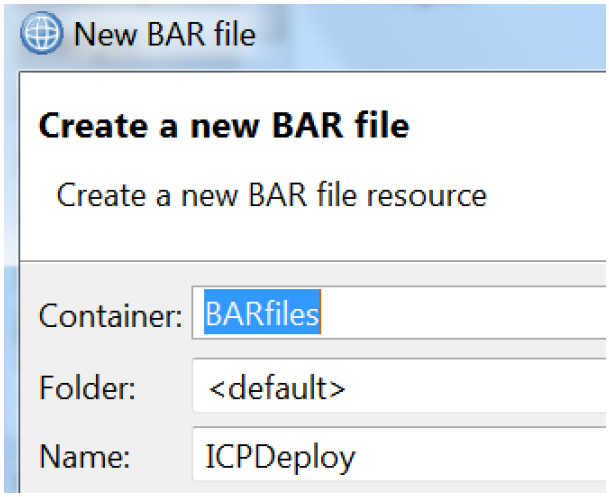
In this example Message flow MyHTTPESQLTestFlow is in MyICPTestApp IIB application



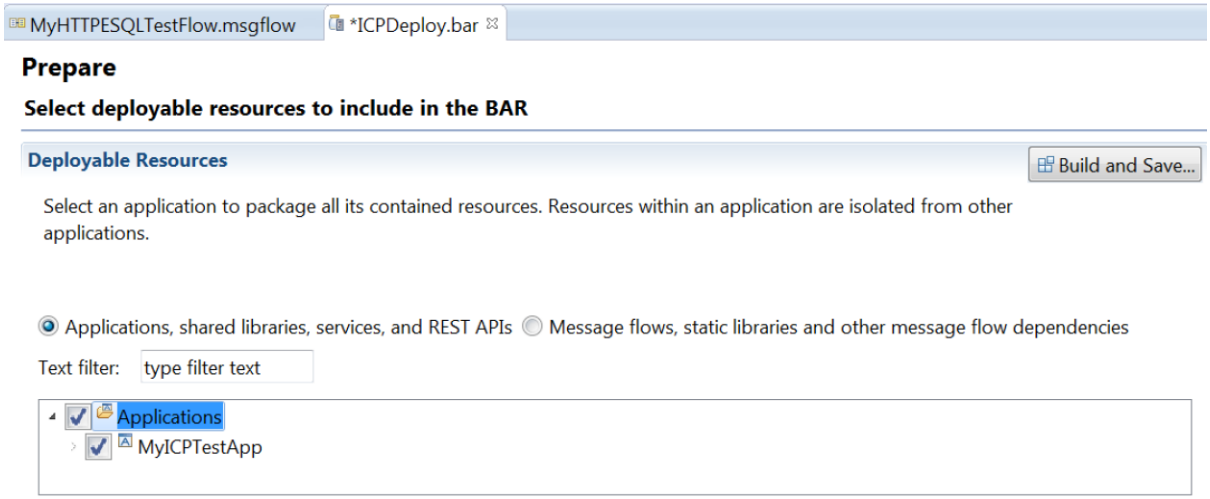
Compute Node InToOut references the following ESQL

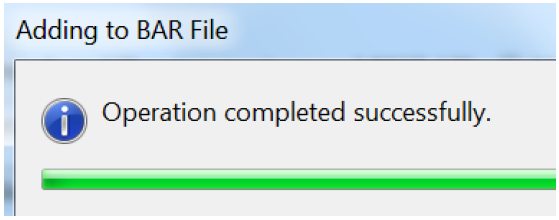
```
CREATE COMPUTE MODULE MyHTTPESQLTestFlow_InToOut
	CREATE FUNCTION Main() RETURNS BOOLEAN
	BEGIN
		CALL CopyMessageHeaders();

		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

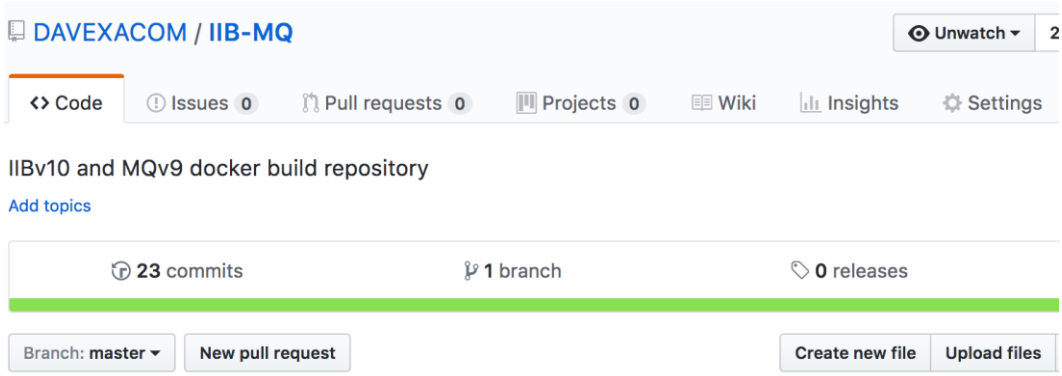


Select MyICPTestApp and hit Build and Save

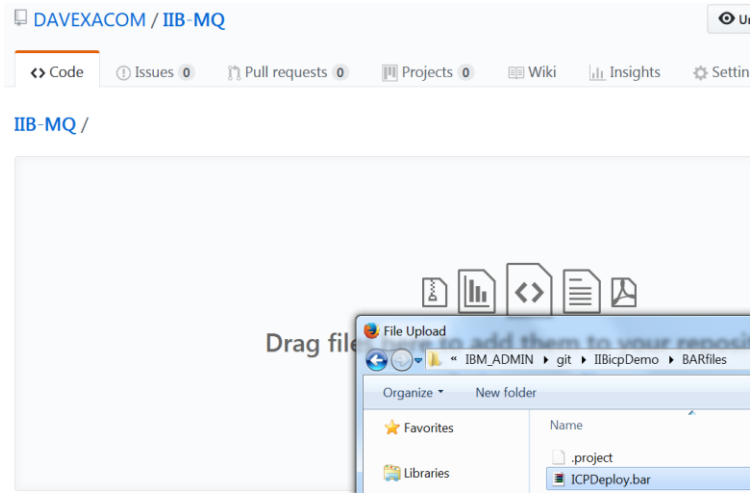




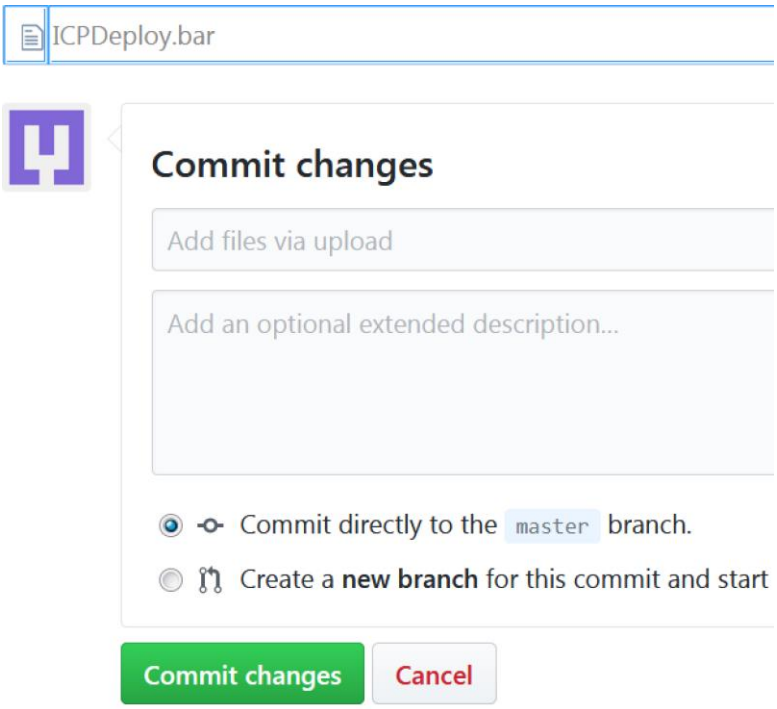
Optional Section – Upload the BAR file to the IIB-MQ repo on GitHub



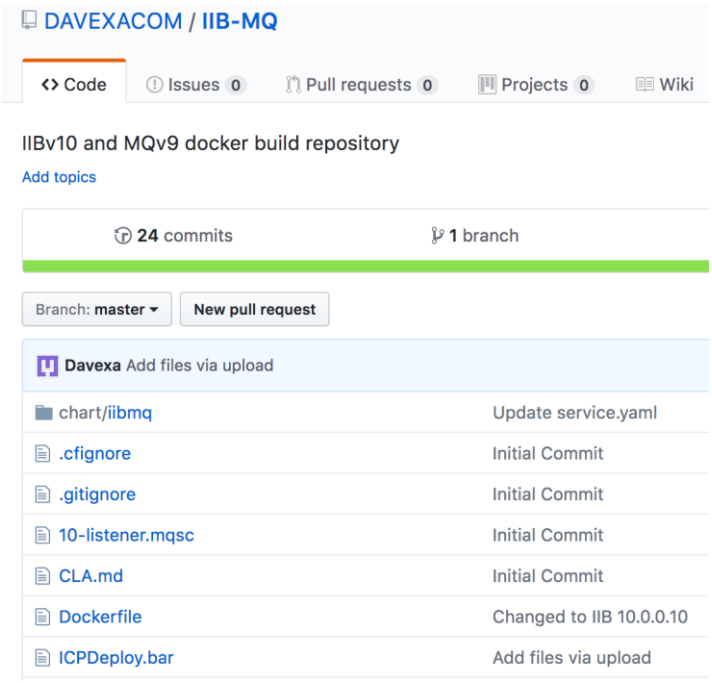
Navigate to your BAR file location in the IIB workspace on the local file system



Upload and commit changes

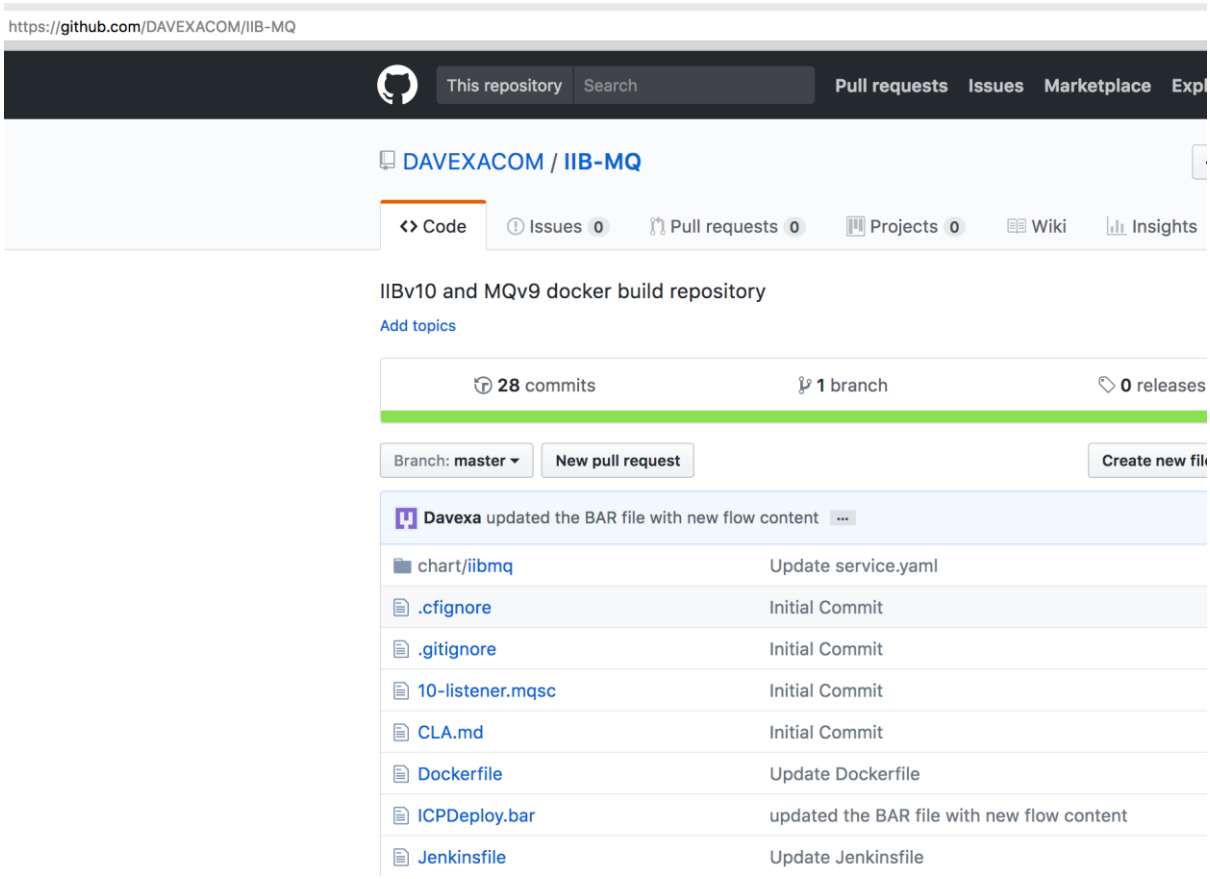


Check the respository for your BAR file.



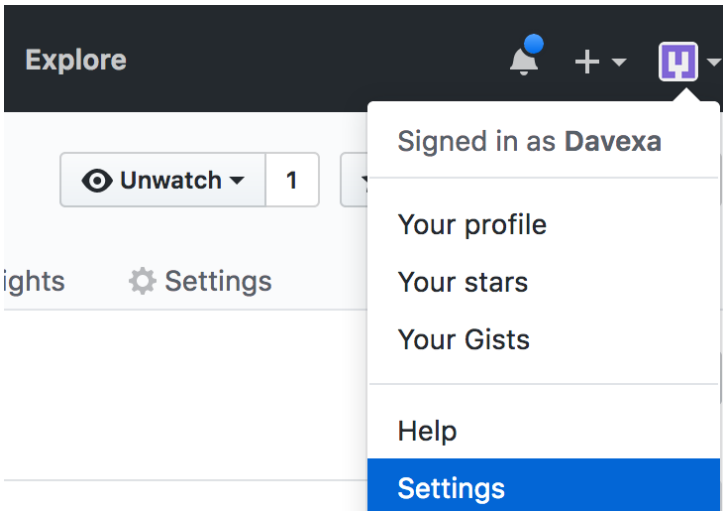
Configuring Github repository IIB-MQ

OAuth Access for Github repository IIB-MQ in organization DAVEXACOM

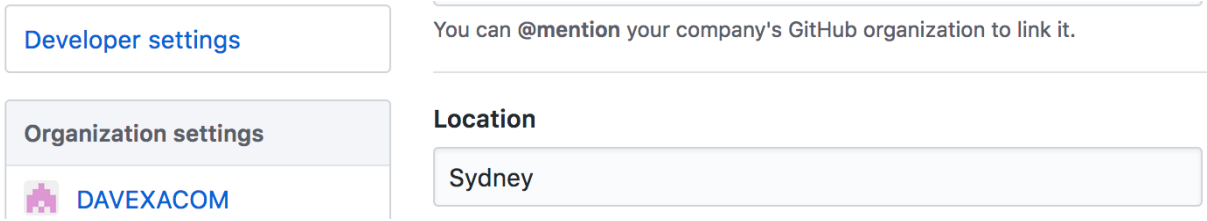


Github Personal Access Token – Create a token

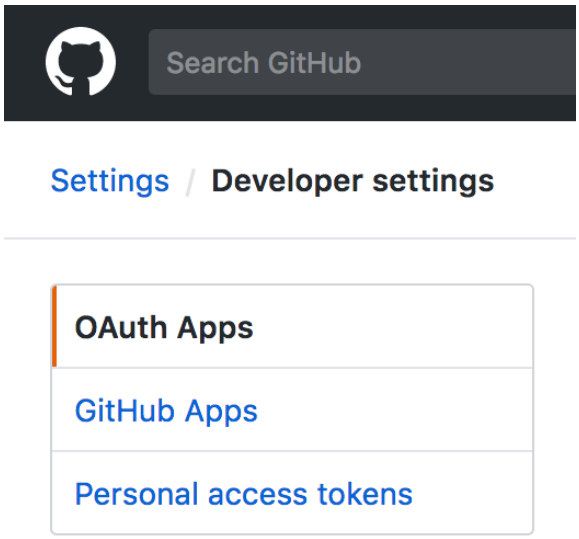
Developer Settings – Personal Access tokens



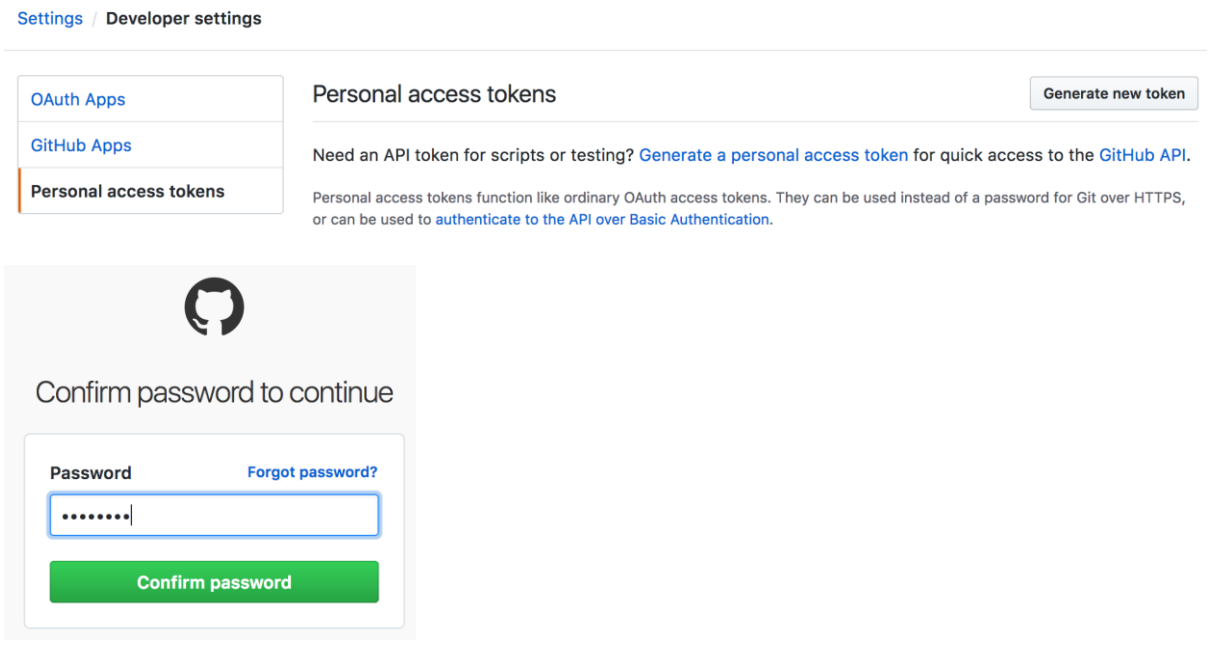
Select Developer settings



Select personal access tokens



Generate new token



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?

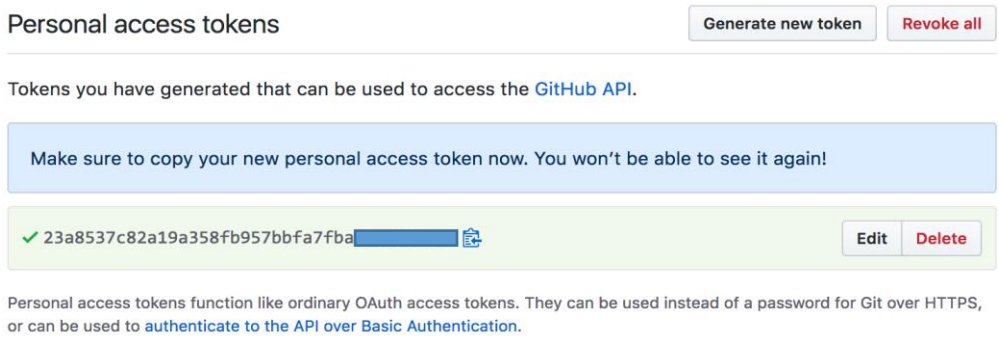
Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

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

Generate token Cancel

Hit Generate token

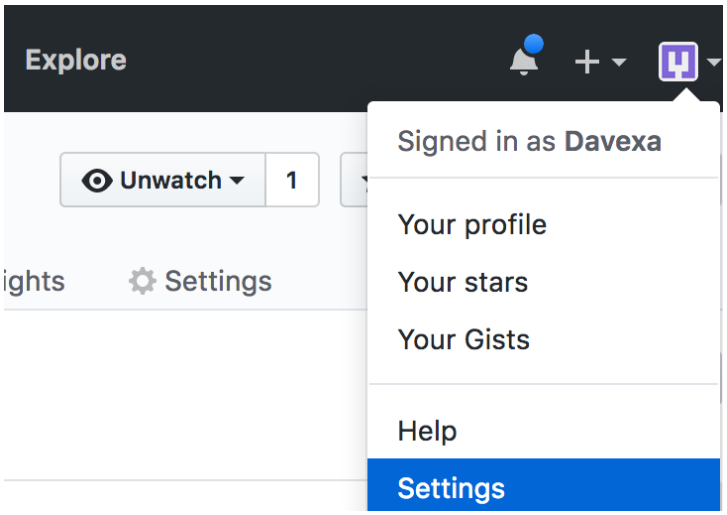


Copy and paste the token for later use.

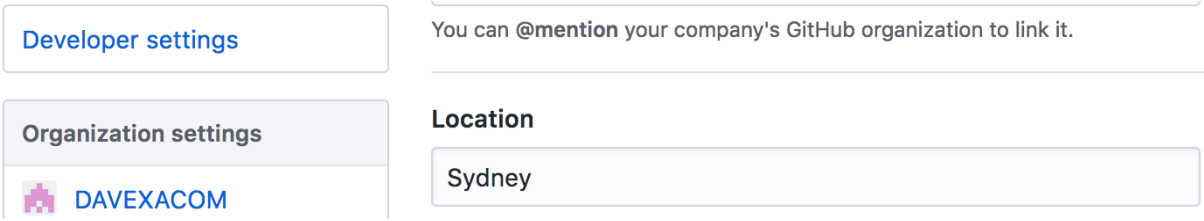
OAuth Apps – Register a new application

Register a new OAuth application

Developer Settings – Set up OAuth

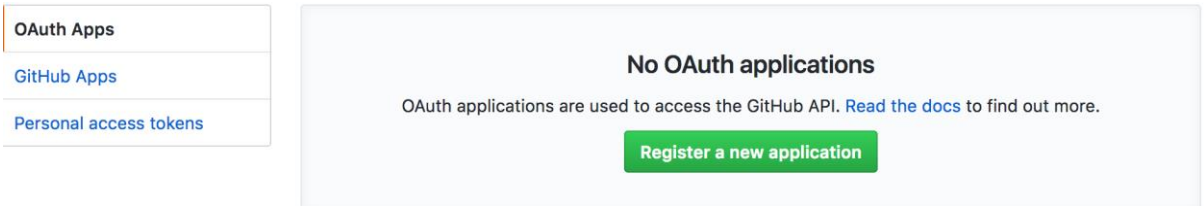


Select Developer settings



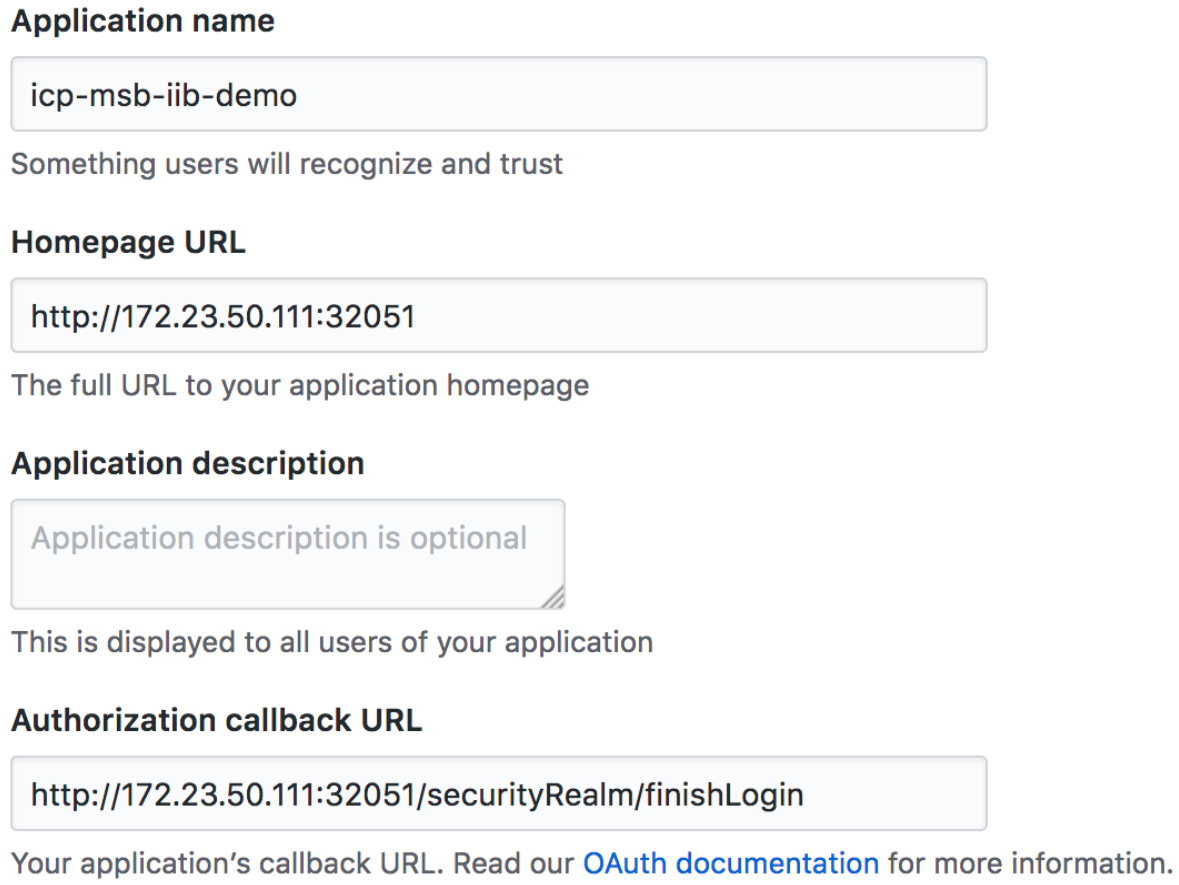
OAuth Apps – Register a new application

Register a new OAuth application

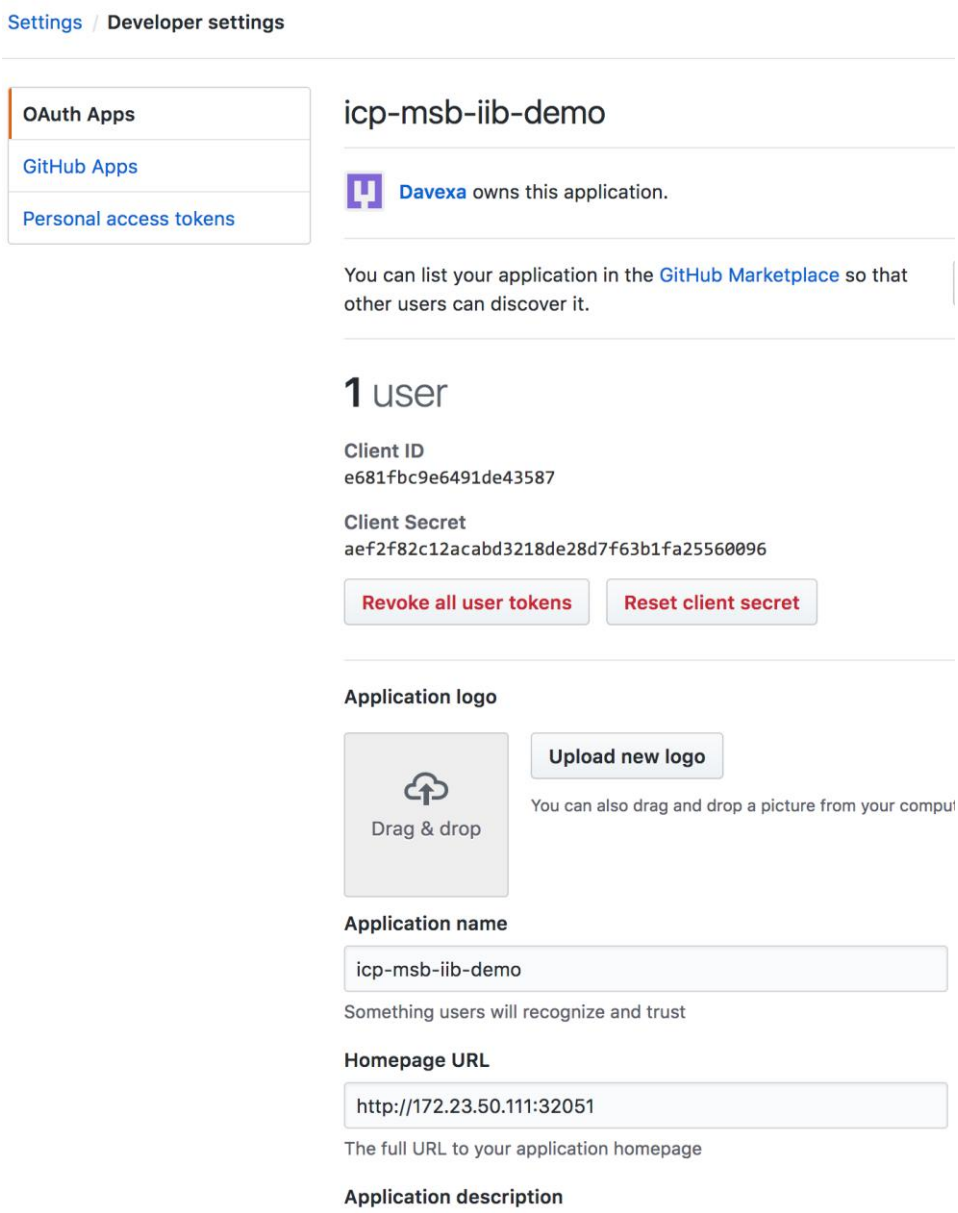


Important! Pick a port number for the pipeline. You will use it when you create the pipeline. (you may not get it right first time as the port might be in use when you create the pipeline. If you have to use a different port at pipeline creation time. You will need to come back here and change this port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

Lets assume 32051 for now!



Click the green register application buttin

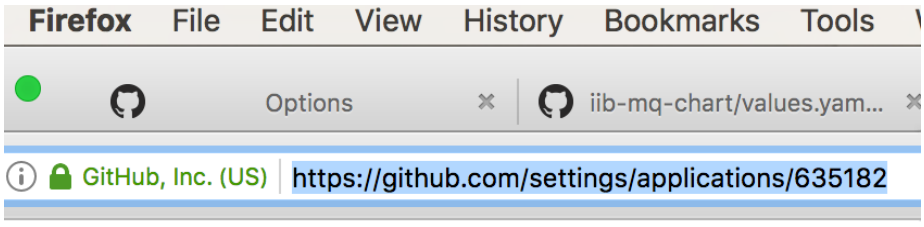


Capture OAuth Client ID and Secret

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.



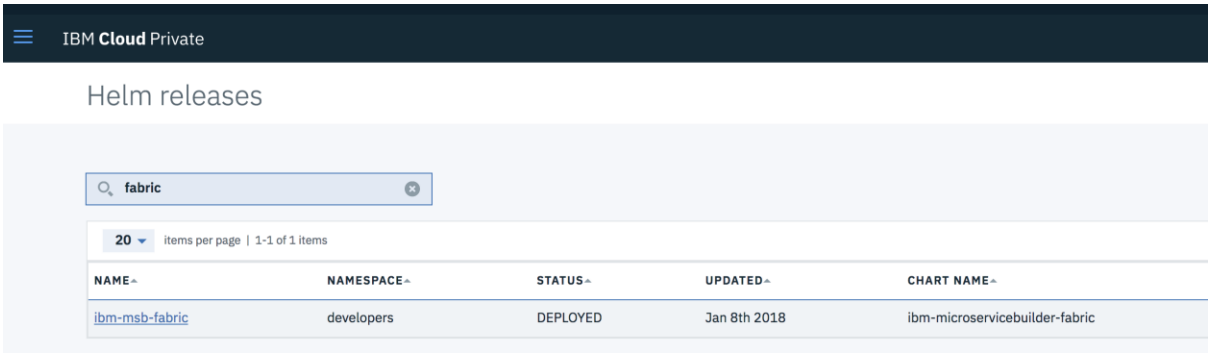
IBM Cloud Private – Configure and Deploy Micro Service Builder

Jenkins is the key component of the Micro Service Builder Pipelie

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”



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

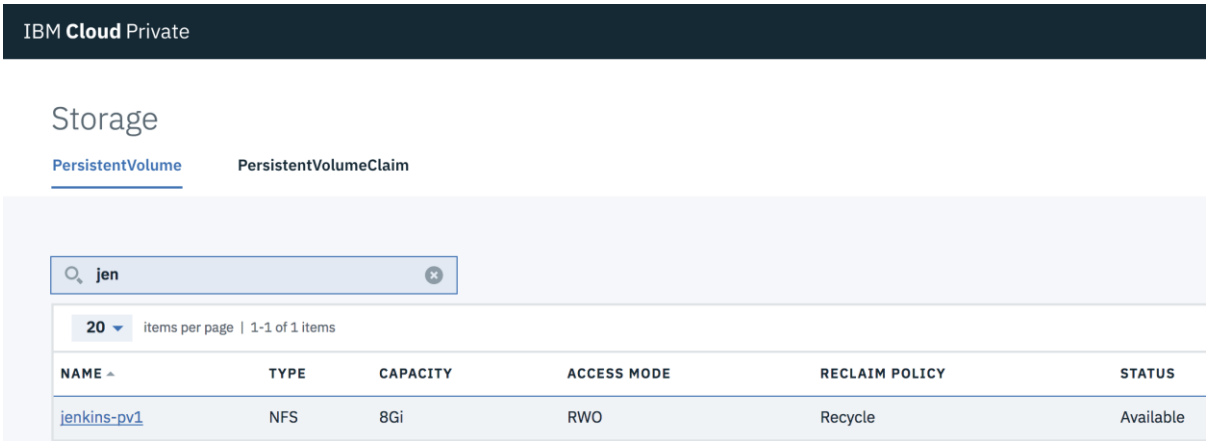
Check Jenkins Persistent Volume

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



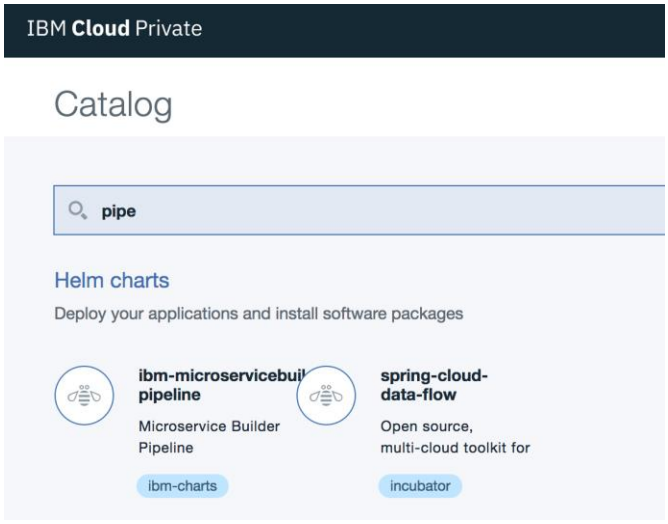
If there is no persistent volume that can be identified for use by jenkins then either create one or ask the ICP administrator.

Configure and Deploy Micro Service Builder Pipeline

Log into the console

Select Catalog

Filter on pipe



Select ibm-microservicebuilder pipeline

Configure the ibm-microservicebuilder pipeline helm chart

Hit the configure button

Set the Github settings

Repos Pattern is .*

OAuth.User and Admins are your Github user name

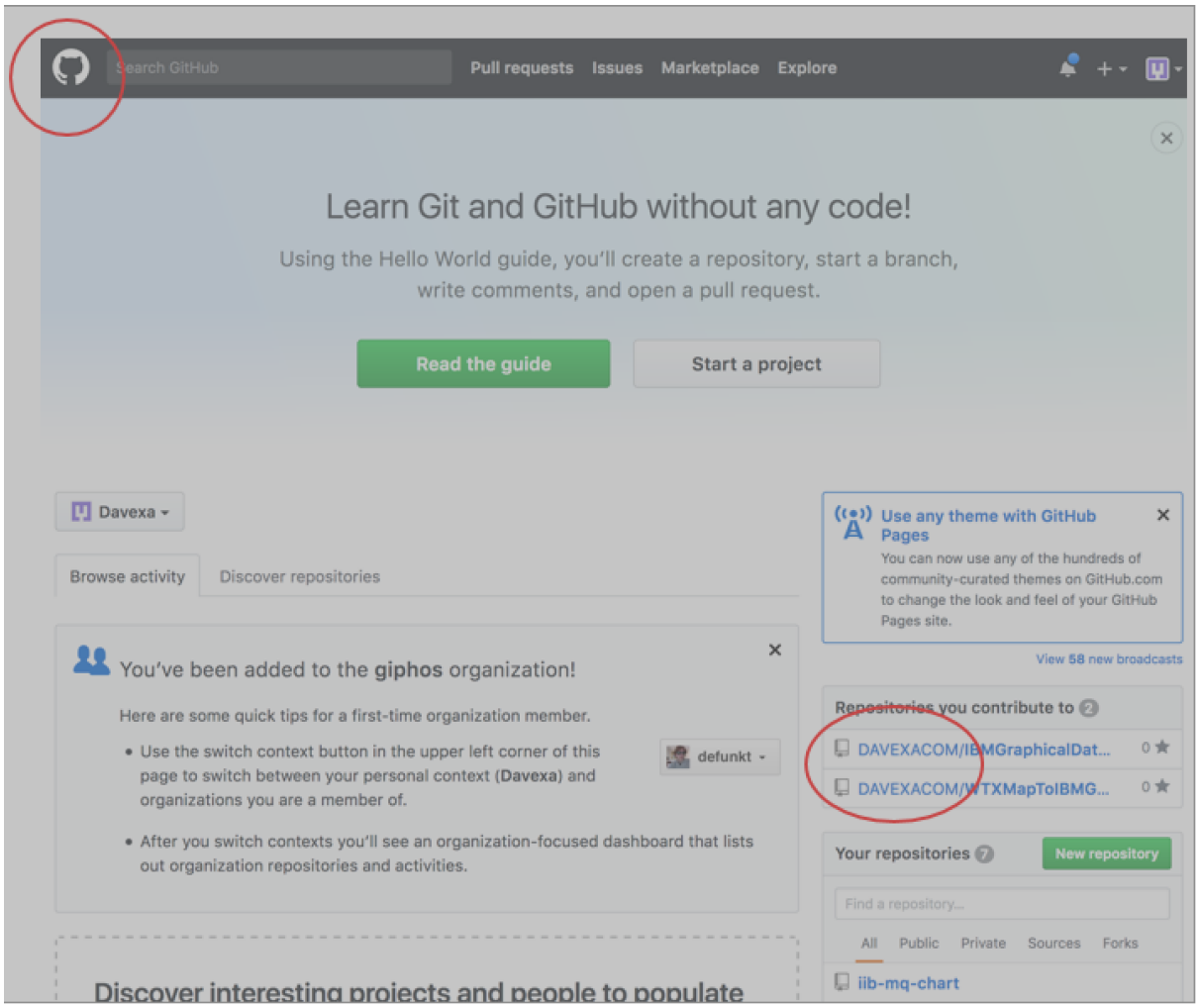
OAuth.Token is the personal access token

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



OAuth Apps

GitHub Apps

Personal access tokens

icp-msb-iib-demo

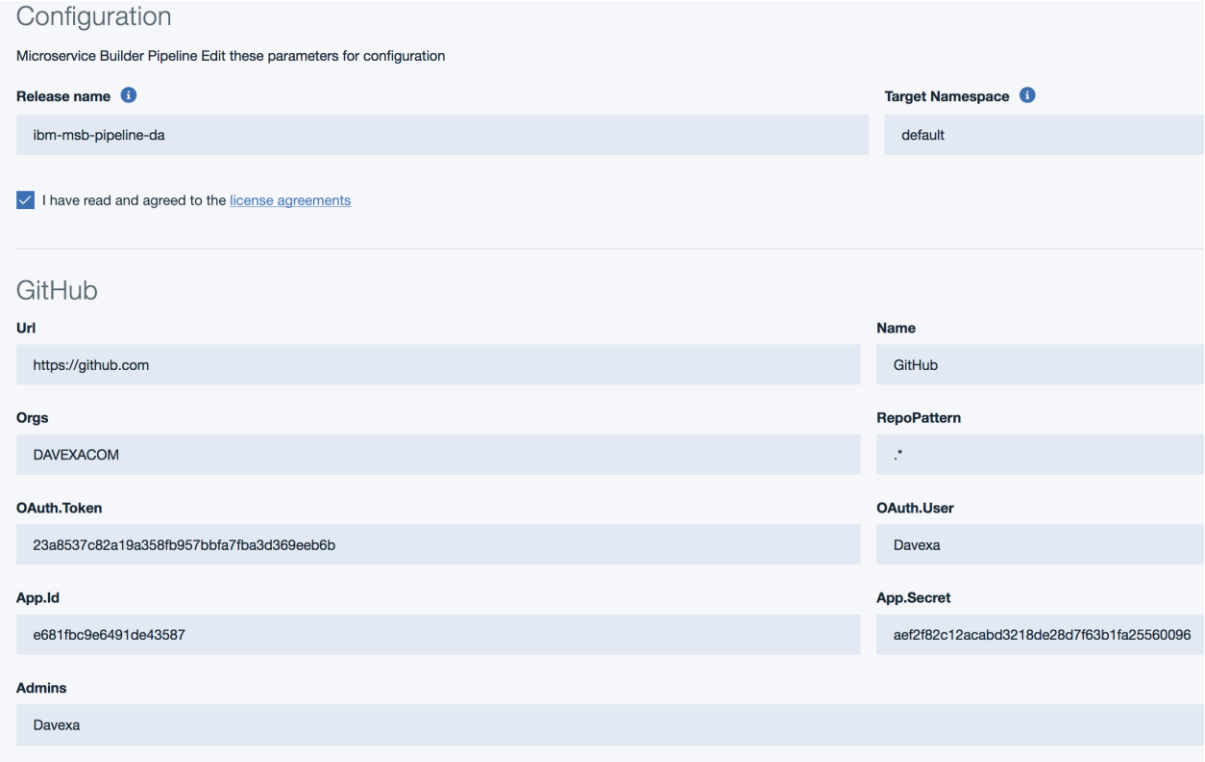
Davexa owns this application.

You can list your application in the GitHub Marketplace so that other users can discover it.

1 user

Client ID
`e681fbc9e6491de43587`

Client Secret
`aef2f82c12acabd3218de28d7f63b1fa25560096`



Pipeline Settings

Registry.URL is icpcluster.icp:8500/default

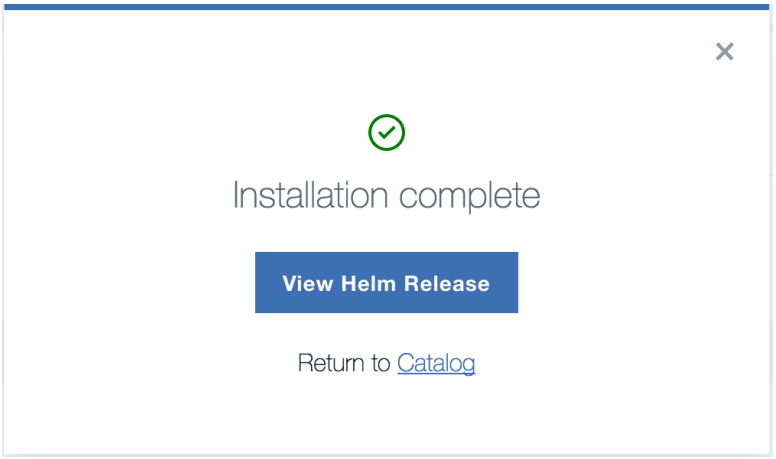
Master Settings

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.

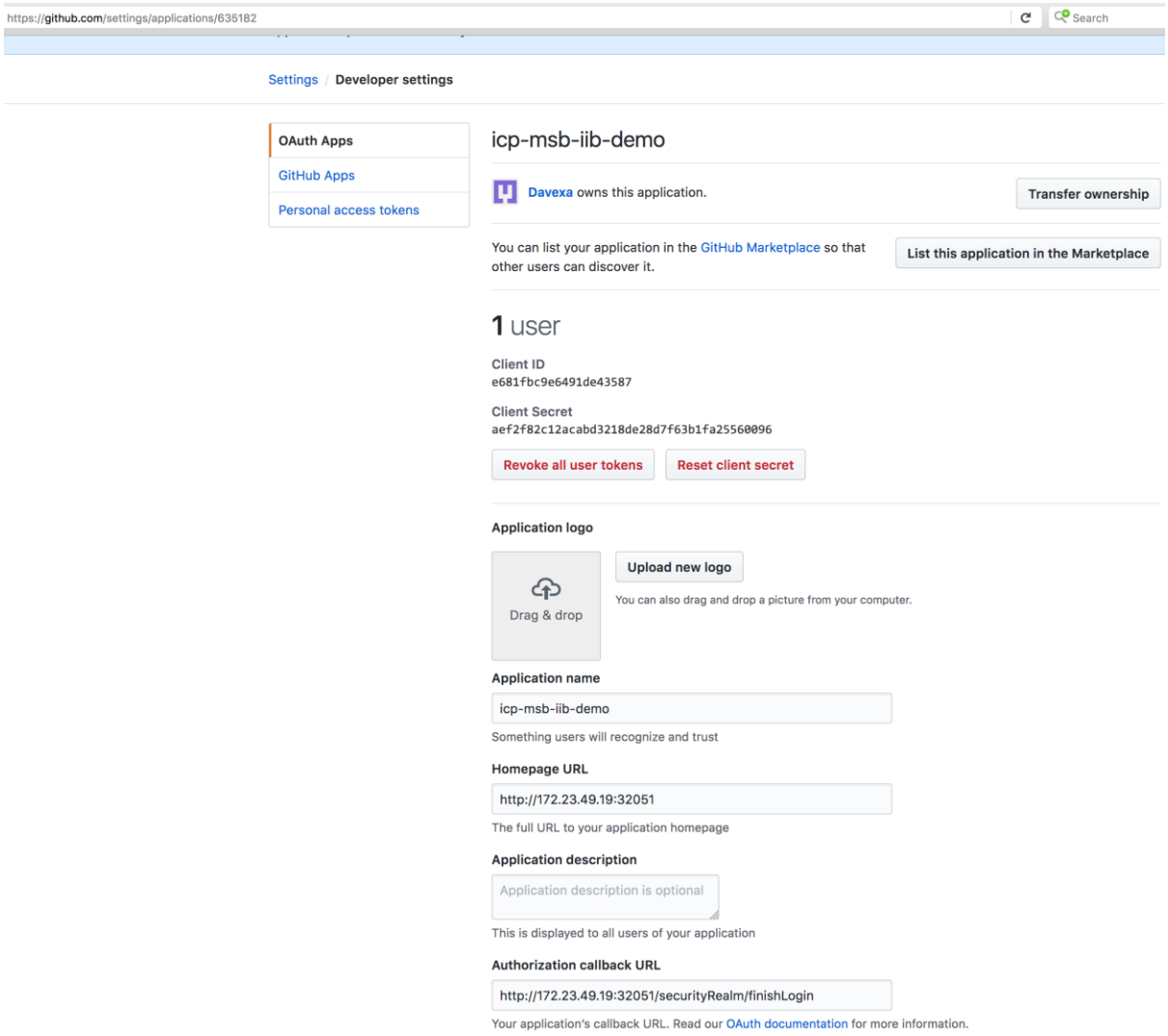
Although we check on a persistent volume for Jenkins we don't have to use it. I haven't used one in this example.

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.



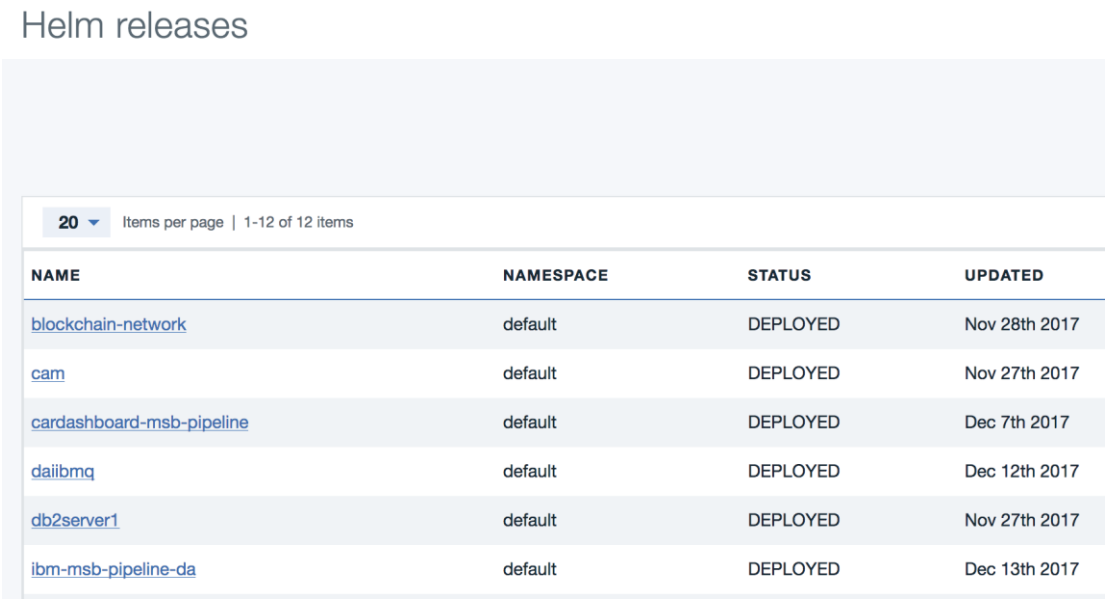
Note: Port number appears in 2 places. Homepage URL and Authorization call back

Then hit the update button.

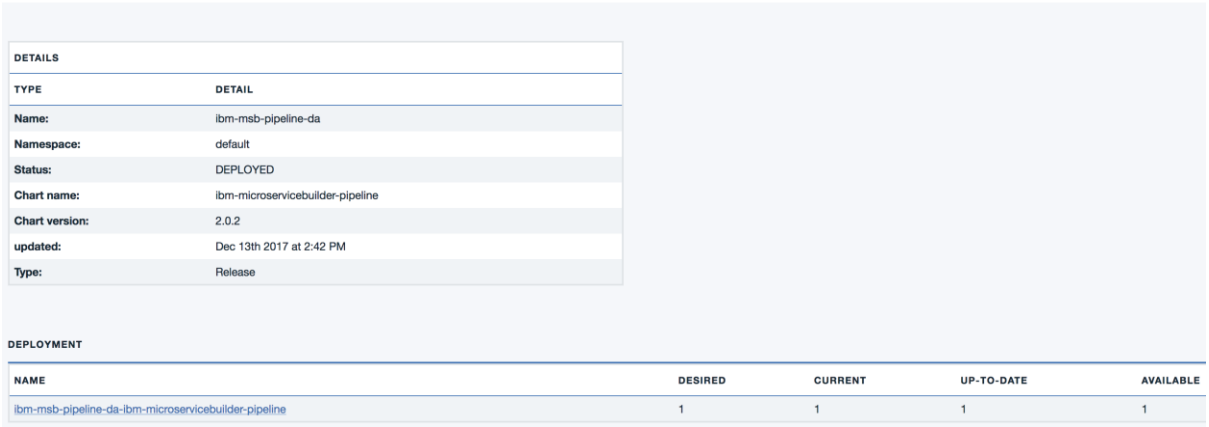
Your Github port will now match the icp msb pipeline parameters.

Click on View Helm Release

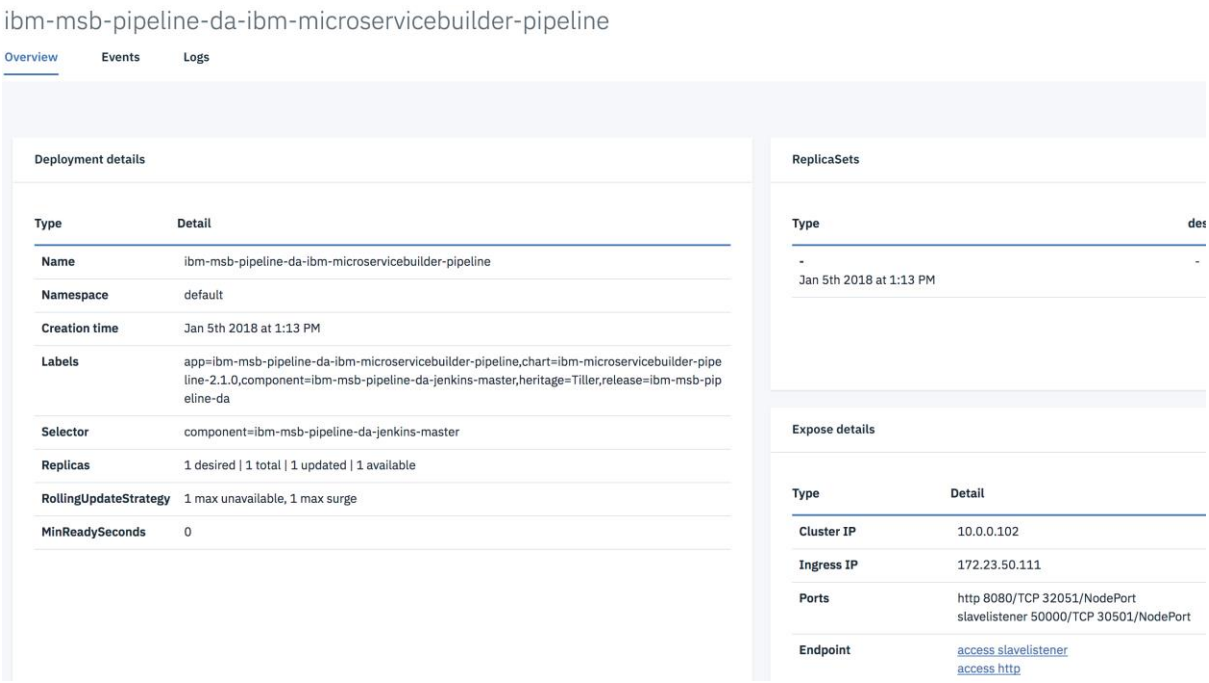
Review the Micro Service Builder pipeline helm release



Select ibm-msb-pipeline-da



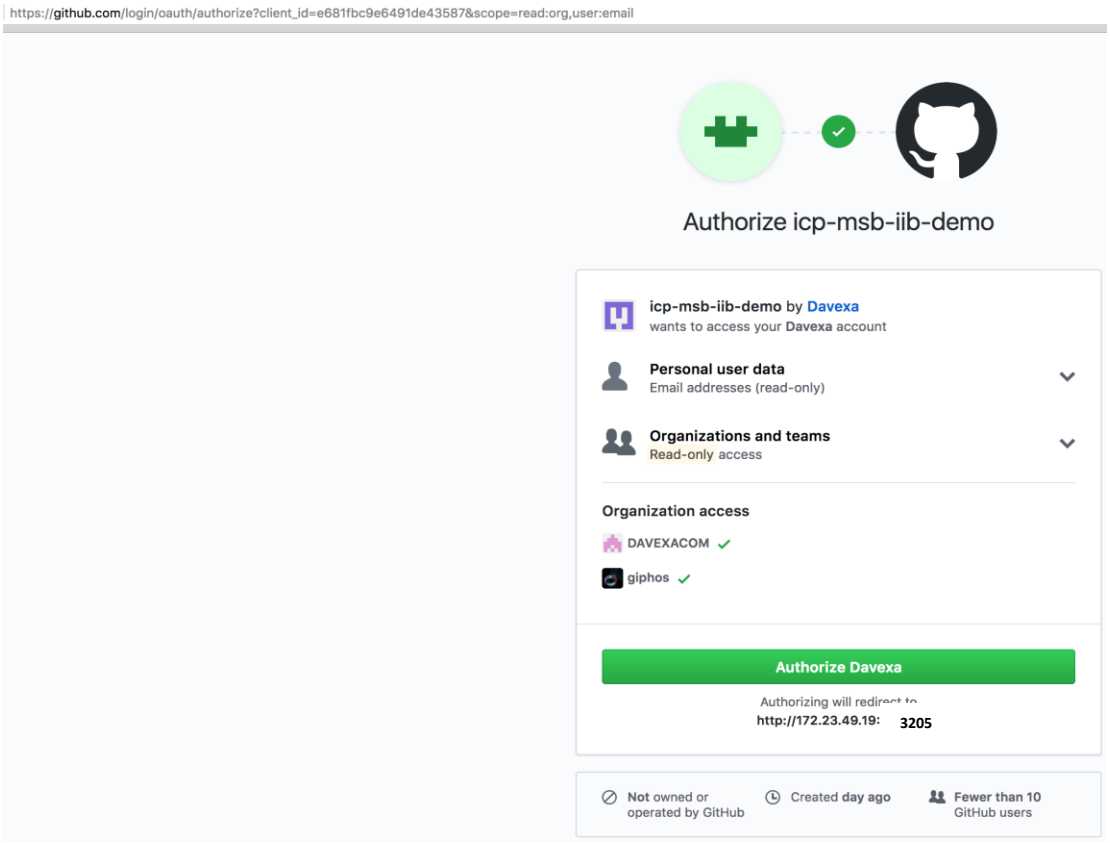
Select the Deployment link



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 Authoried Davexa (in this case) to access the organization repository for OAuth

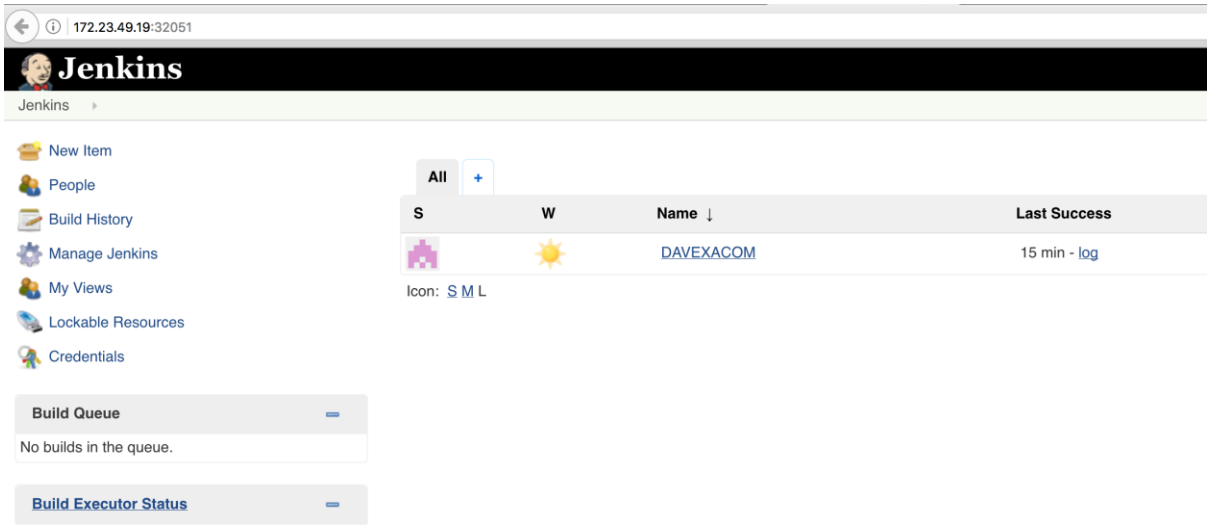


If when you hit the agree Authorize Davexa you get the following

Error

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

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

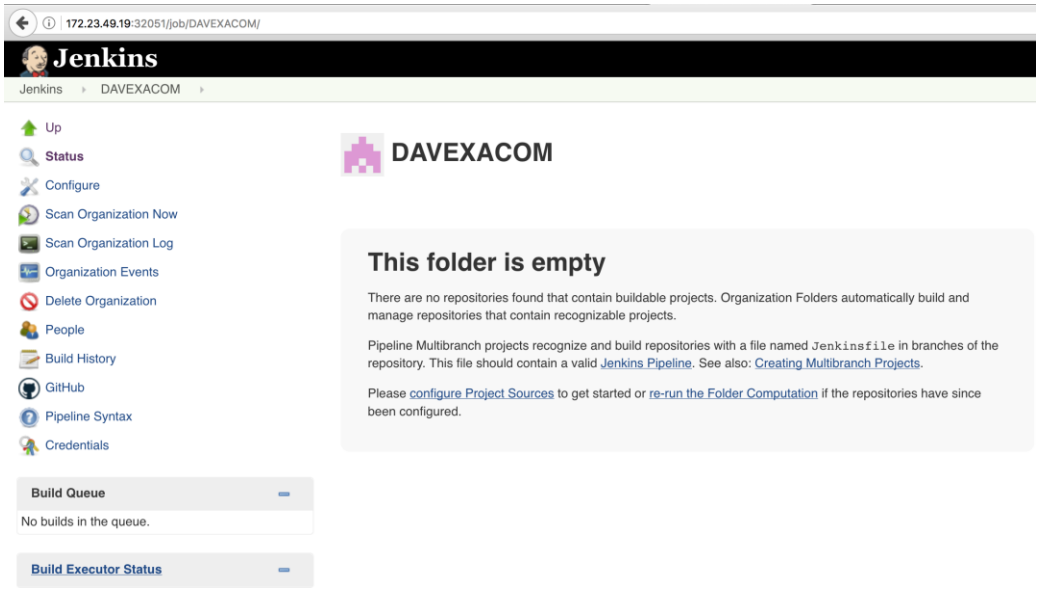


Micro Services Builder - Jenkins

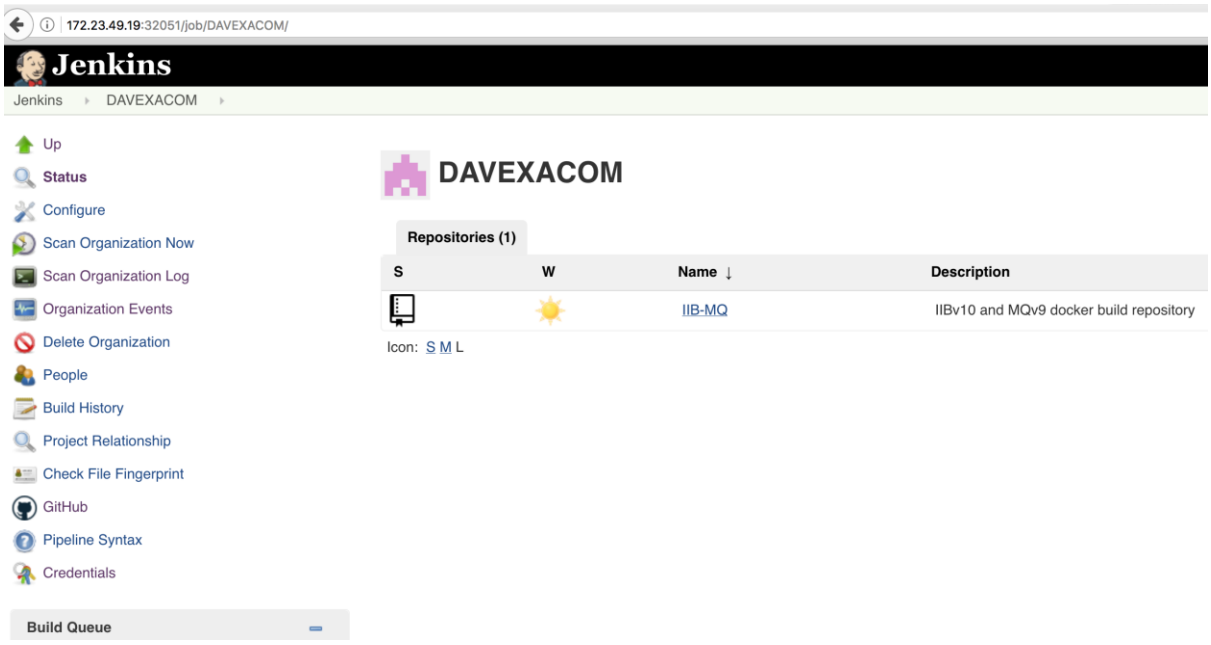
Checking Jenkins connection to Github

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.



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

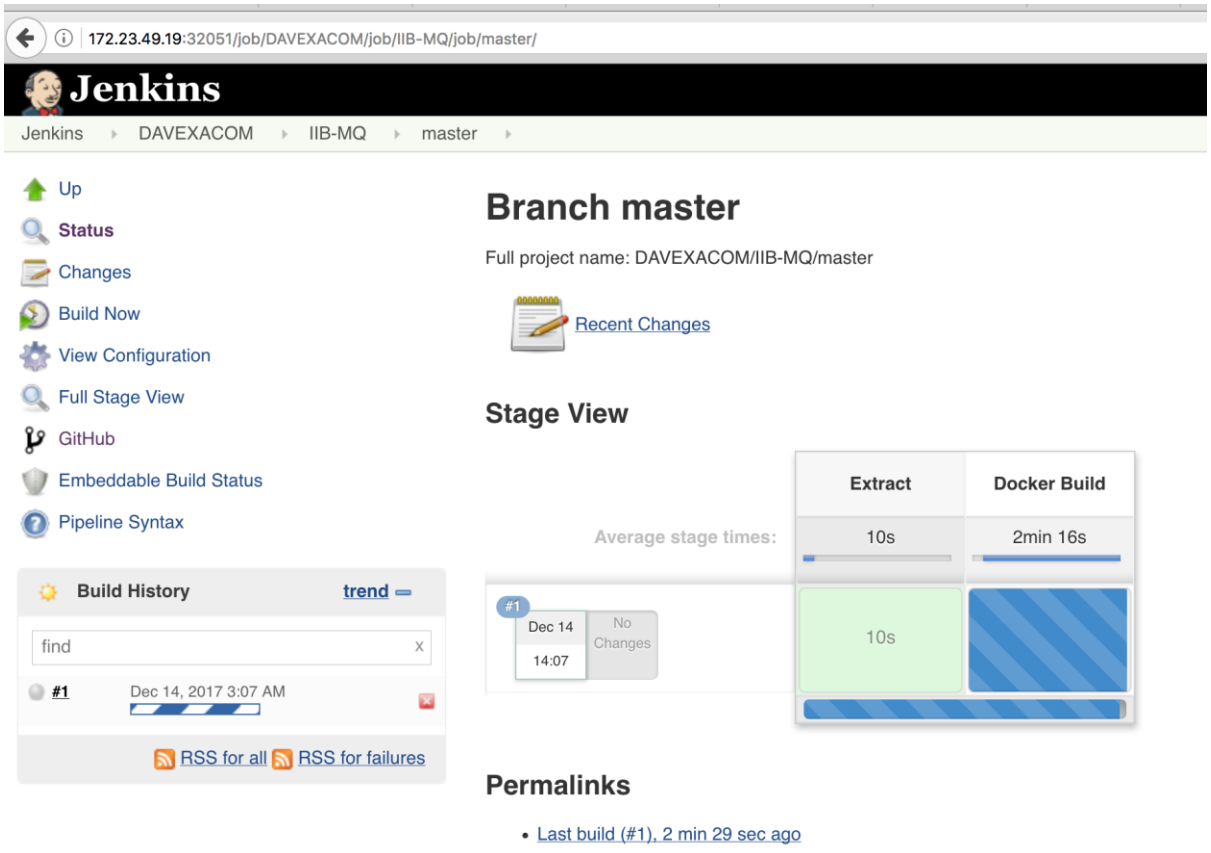
57 lines (50 sloc) | 1.97 KB | Raw | Blame

```yaml
license: "accept"

image:
  # repository is the container repository to use, which must contain IBM MQ Advanced for Developers
  repository: icpcluster.icp:8500/default/iib-mq
  # tag is the tag to use for the container repository
  tag: latest
  # pullSecret is the secret to use when pulling the image from a private registry
  pullSecret:
  # pullPolicy is either IfNotPresent or Always (https://kubernetes.io/docs/concepts/containers/images/)
  pullPolicy: IfNotPresent

# persistence section specifies persistence settings which apply to the whole chart
persistence:
  # enabled is whether to use Persistent Volumes or not
  enabled: false
  # useDynamicProvisioning is whether or not to use Storage Classes to dynamically create Persistent Volumes
  useDynamicProvisioning: true

# dataPVC section specifies settings for the main Persistent Volume Claim, which is used for data in /var/mqm
dataPVC:
  # name sets part of the name for this Persistent Volume Claim
  name: "data"
  ## storageClassName is the name of the Storage Class to use, or an empty string for no Storage Class
  storageClassName: ""
  ## size is the minimum size of the Persistent Volume
  size: 2Gi

service:
  name: iib-mq
  type: NodePort

resources:
  limits:
    cpu: 1024m
    memory: 1024Mi
  requests:
    cpu: 1024m
    memory: 1024Mi

# queueManager section specifies settings for the MQ Queue Manager
queueManager:
  # name allows you to specify the name to use for the queue manager.  Defaults to the Helm release name.
  name: icpqm1
```

I changed the queueManager->name to icpDA1 for example

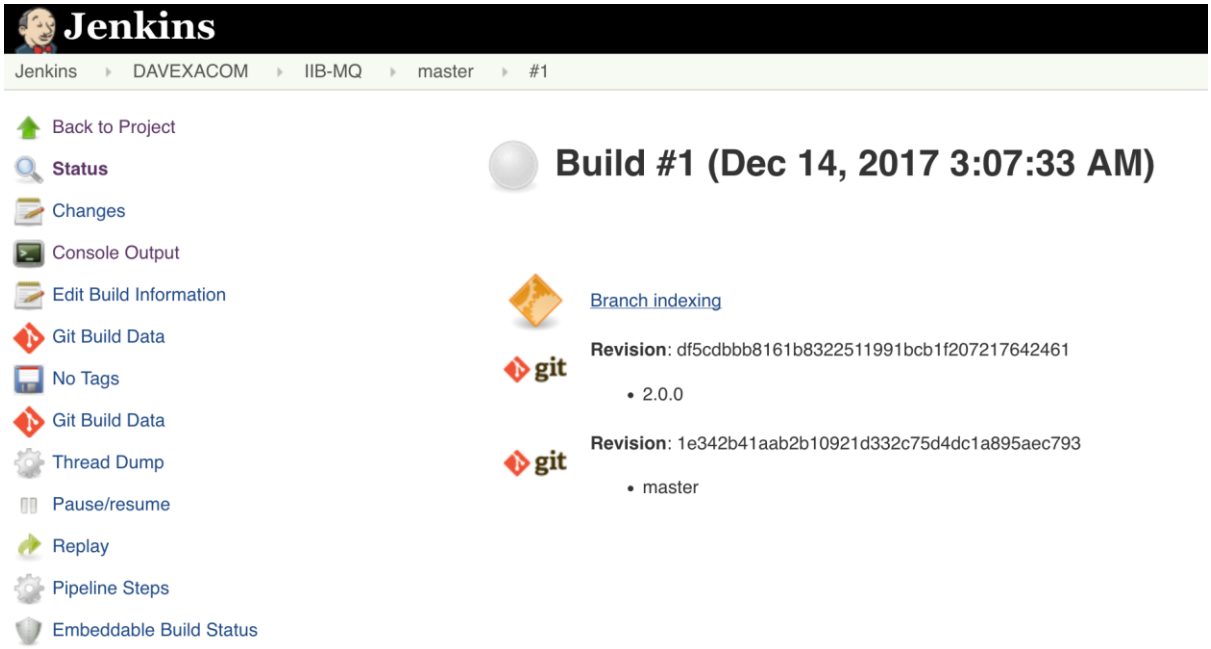
Starting the Jenkins build process manually

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.

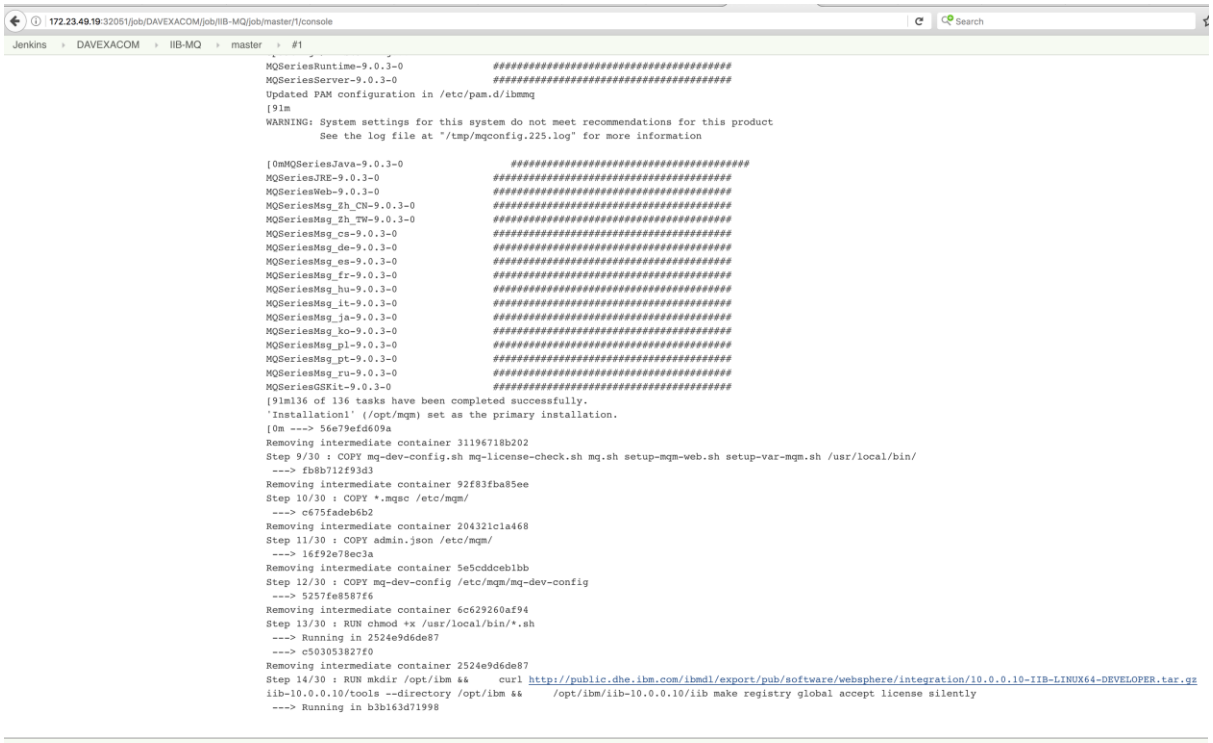


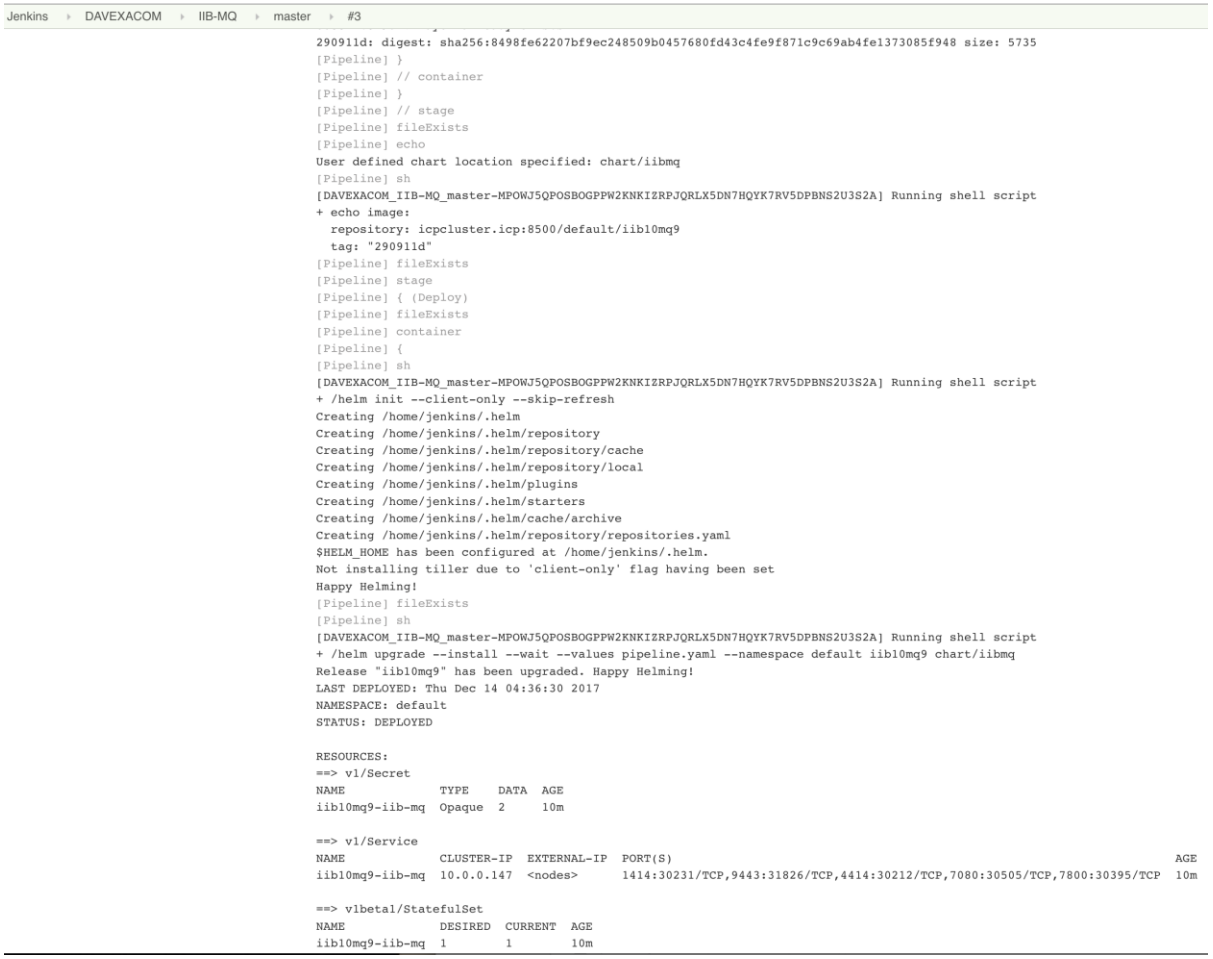
Following the Jenkins build logs

Click on the #1 in Build history



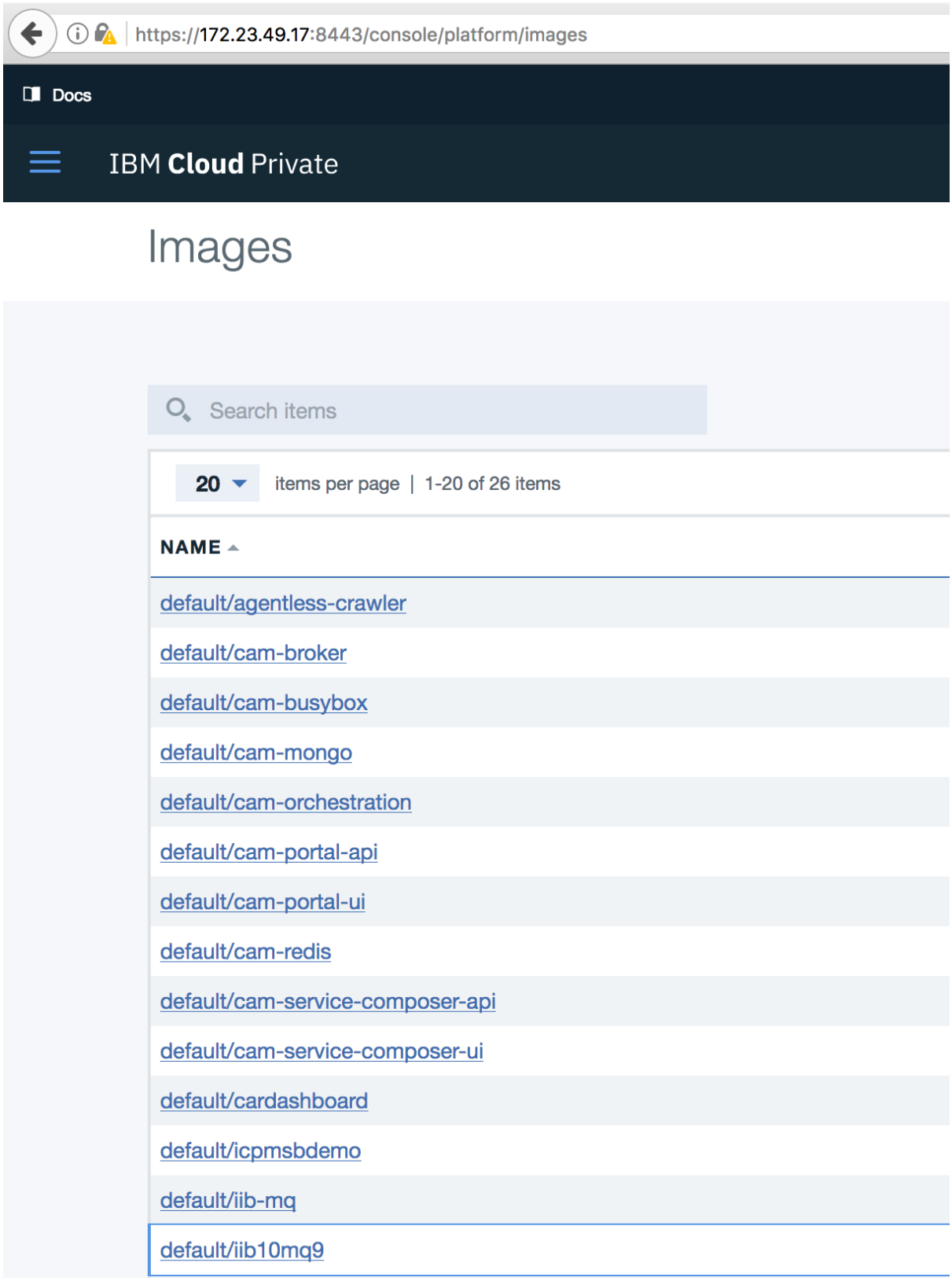
Click on Console Output


172.23.49.19:32051/job/DAVEXACOM/job/IIB-MQ/job/master/1/console
Search
Jenkins ▸ DAVEXACOM ▸ IIB-MQ ▸ master ▸ #1
MQSeriesRuntime-9.0.3-0 ##########
MQSeriesServer-9.0.3-0 ##########
Updated PAM configuration in /etc/pam.d/ibmmq
[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 ##########
MQSeriesMsg_Zh_CN-9.0.3-0 ##########
MQSeriesMsg_Zh_TW-9.0.3-0 ##########
MQSeriesMsg_cs-9.0.3-0 ##########
MQSeriesMsg_de-9.0.3-0 ##########
MQSeriesMsg_es-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 ---> 56e79efd609a
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/
---> fb8b712f93d3
Removing intermediate container 92f83fba85ee
Step 10/30 : COPY *.mqsc /etc/mqm/
---> c675fadeb6b2
Removing intermediate container 20432lc1a468
Step 11/30 : COPY admin.json /etc/mqm/
---> 16f92e78ec3a
Removing intermediate container 5e5cddceb1bb
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/*.sh
---> Running in 2524e9d6de87
---> c503053827f0
Removing intermediate container 2524e9d6de87
Step 14/30 : RUN mkdir /opt/ibm && curl http://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-MPOWJ5QPOSBOGPPW2KNKIZRPJQRLX5DN7HQYK7RV5DPBNS2U3S2A] 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-MPOWJ5QPOSBOGPPW2KNKIZRPJQRLX5DN7HQYK7RV5DPBNS2U3S2A] 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-MPOWJ5QPOSBOGPPW2KNKIZRPJQRLX5DN7HQYK7RV5DPBNS2U3S2A] 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-MPOWJ5QPOSBOGPPW2KNKIZRPJQRLX5DN7HQYK7RV5DPBNS2U3S2A] 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


[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



IBM **Cloud** Private

Images / default/iib10mq9 /

default/iib10mq9

Overview

Image details

TYPE	DETAIL
Name	default/iib10mq9
Owner	default
Scope	global
Tags	

Checking ICP Helm Release

IBM **Cloud** Private

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
icpmsbdemo	default	DEPLOYED	Dec 6th 2017	icpmsbdemo
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-iib-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 iibswitch, please wait...
iibswitch started.
BIP8071I: Successful command completion.
BIP8071I: Successful command completion.
BIP8096I: Successful command initiation, check the system log to ensure that the component 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

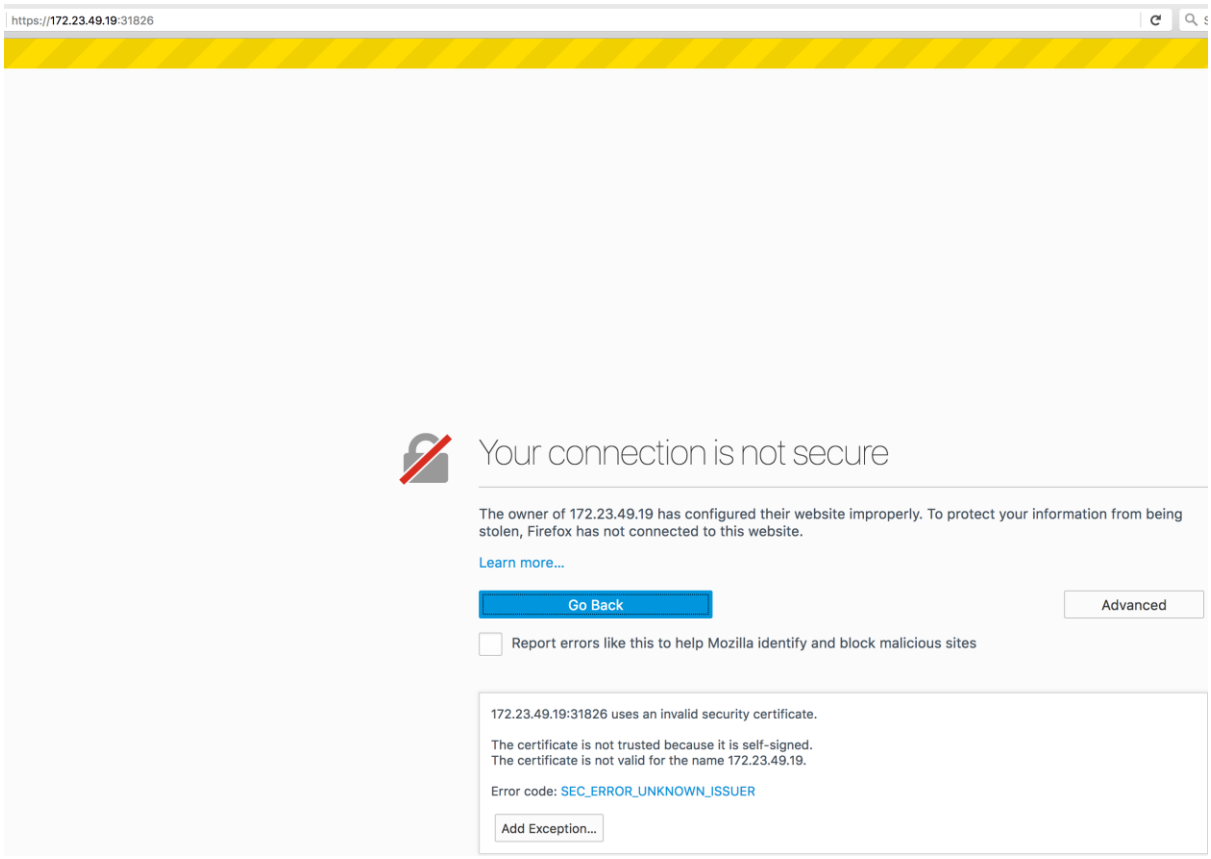
Service details

TYPE	DETAIL
Name	iib10mq9-iib-mq
Namespace	default
Creation time	Dec 14th 2017 at 3:26 PM
Type	NodePort
Labels	app=iib10mq9-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
Session affinity	None

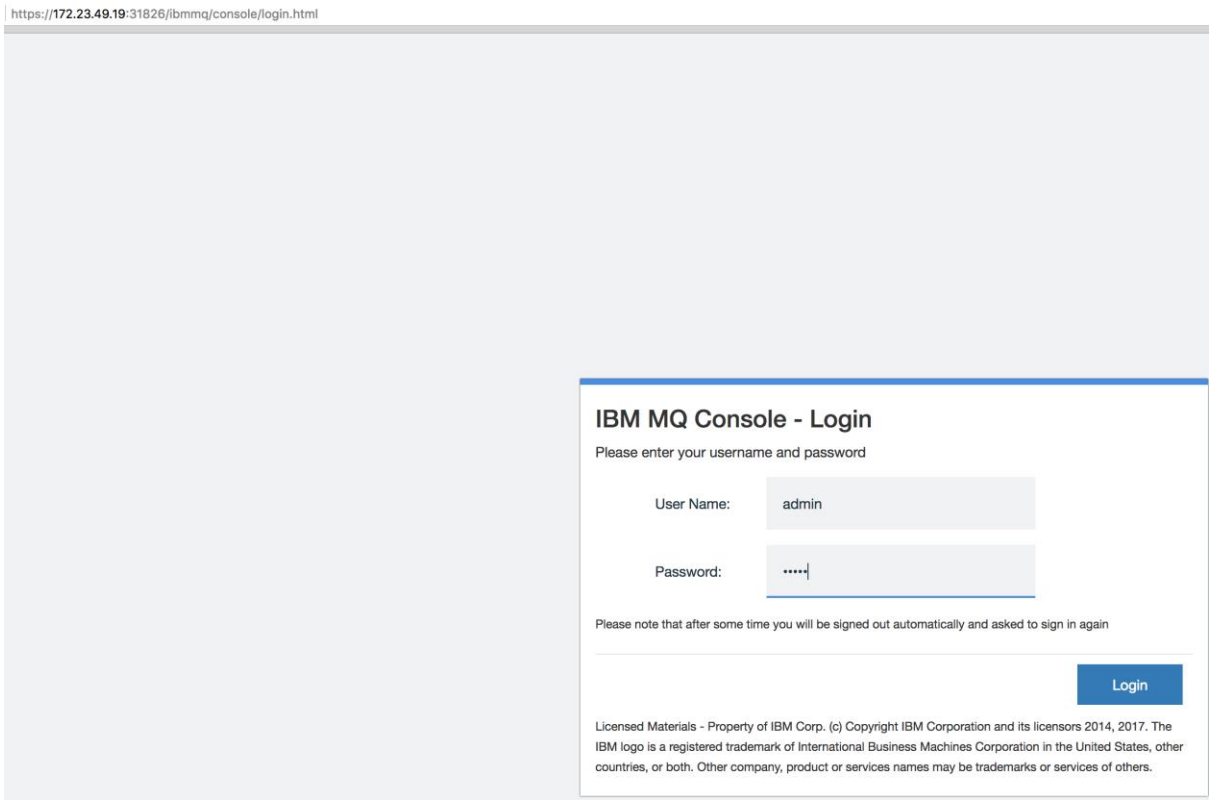
Select the iib-mq-web 31826/TCP link.

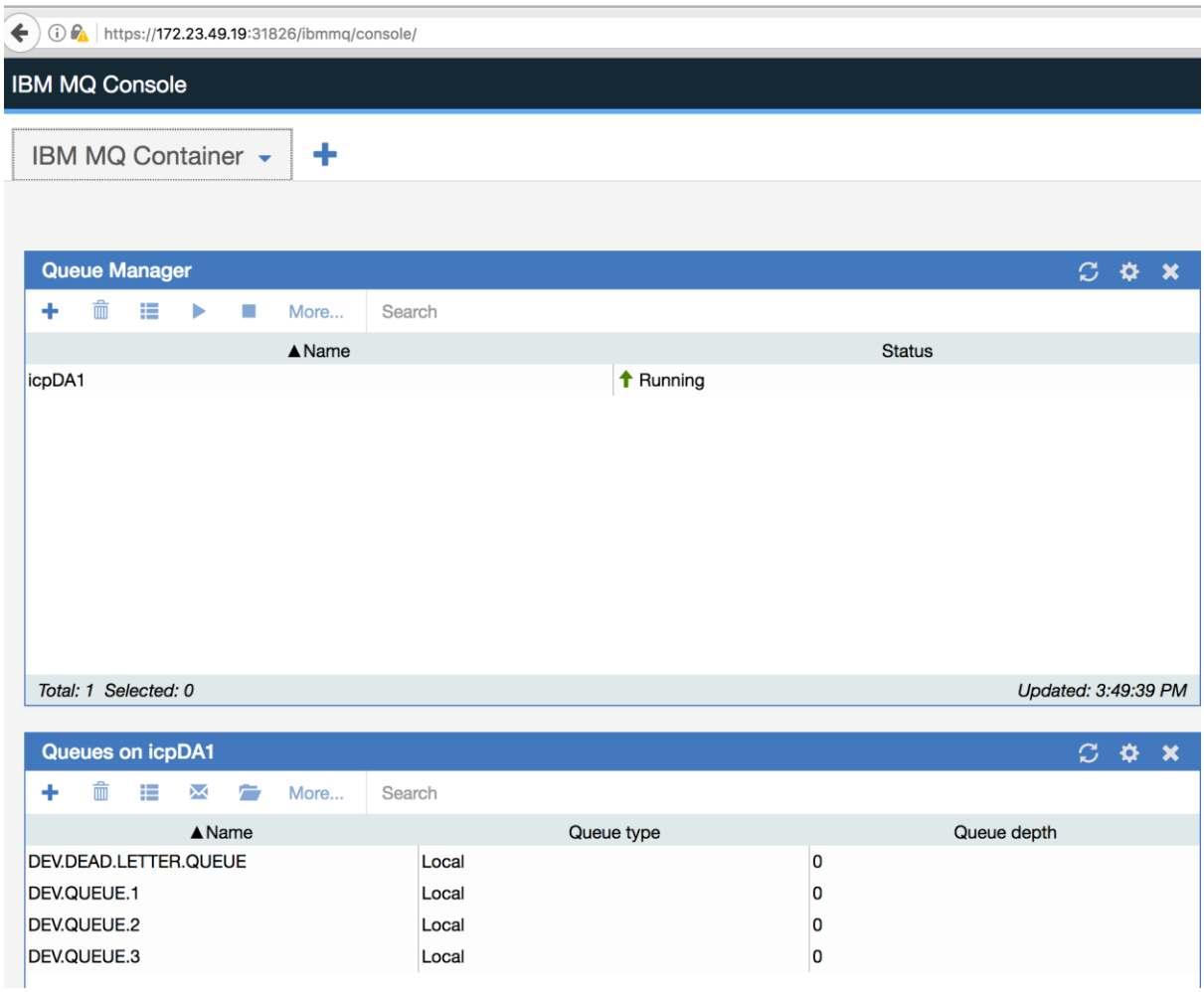
In the browser add the https:// up front

Add certificate

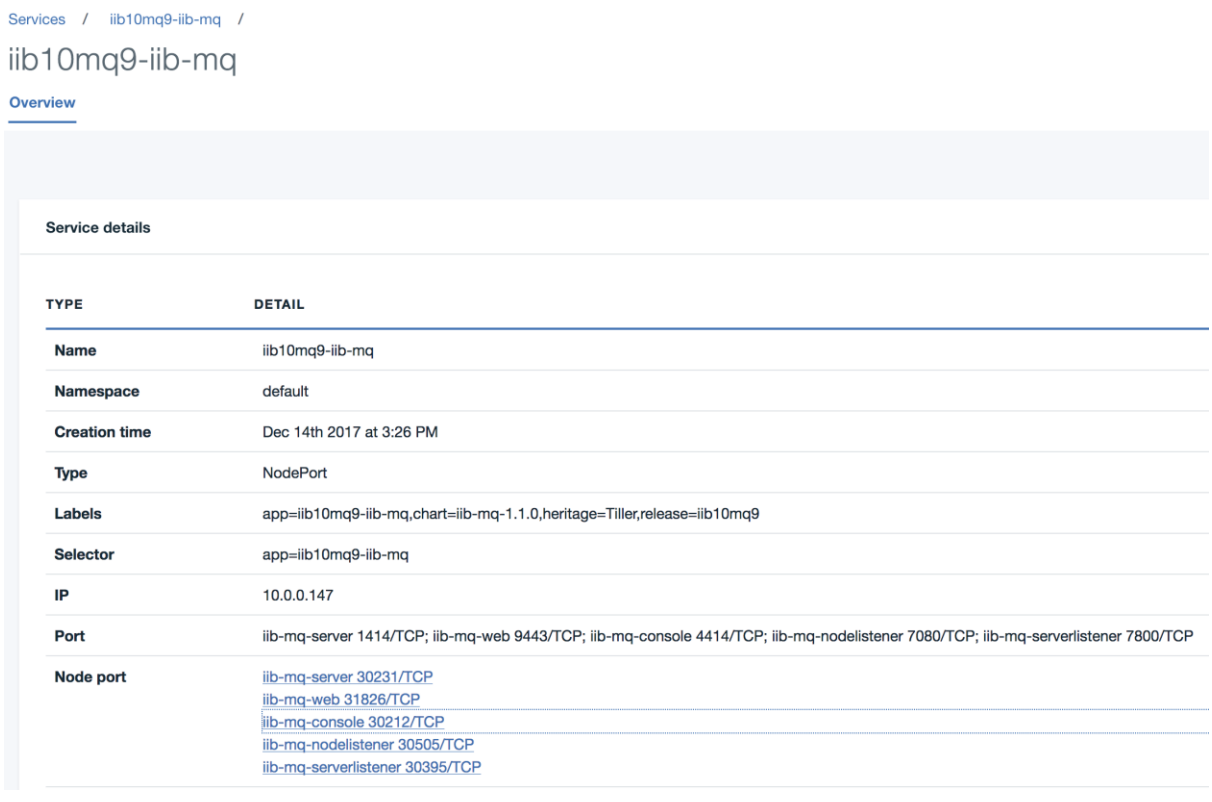


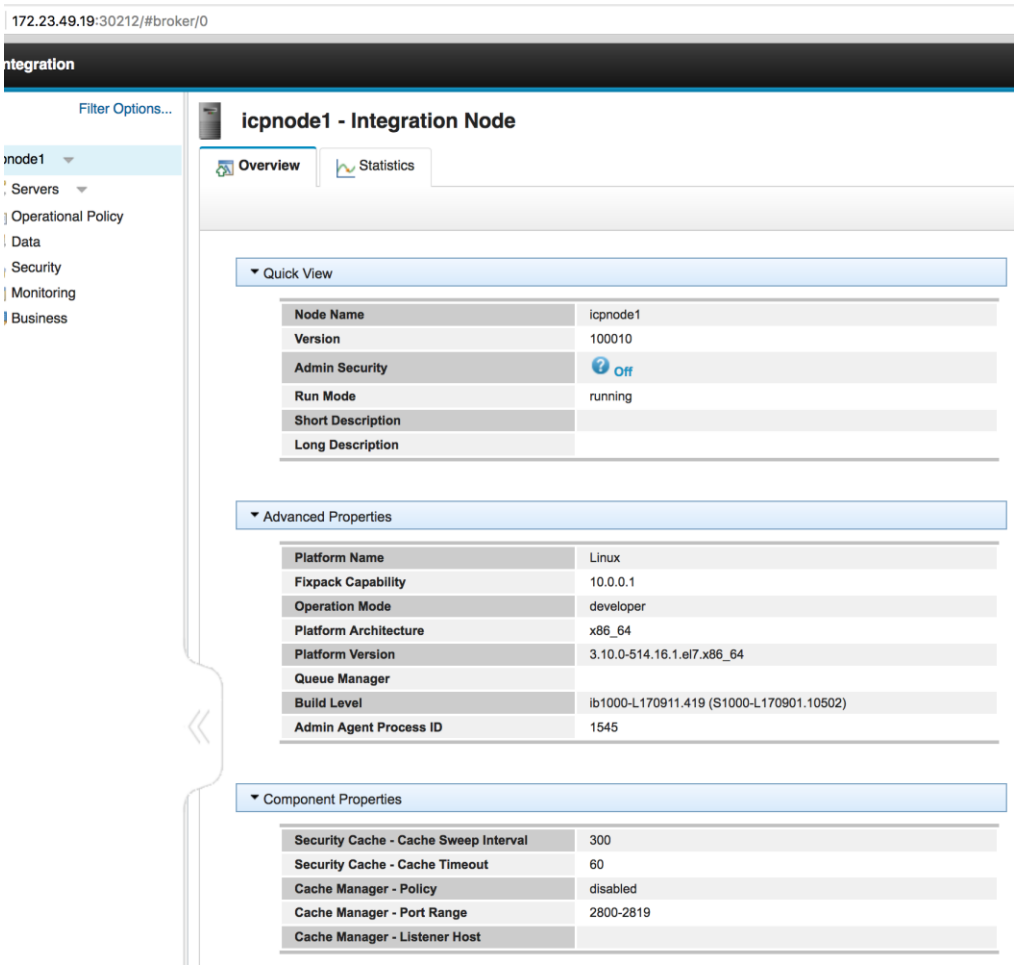
Log into MQ console with admin/passw0rd





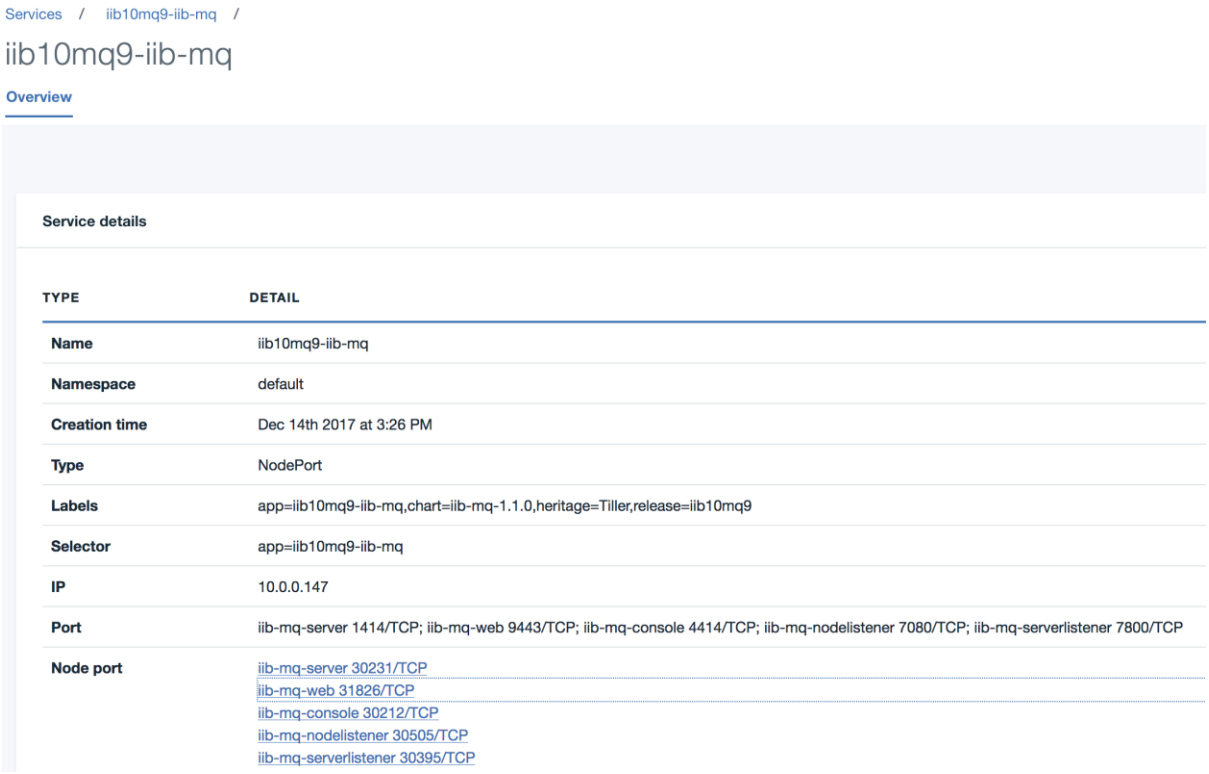
Connect to IIB Web UI





Test the deployed message flow

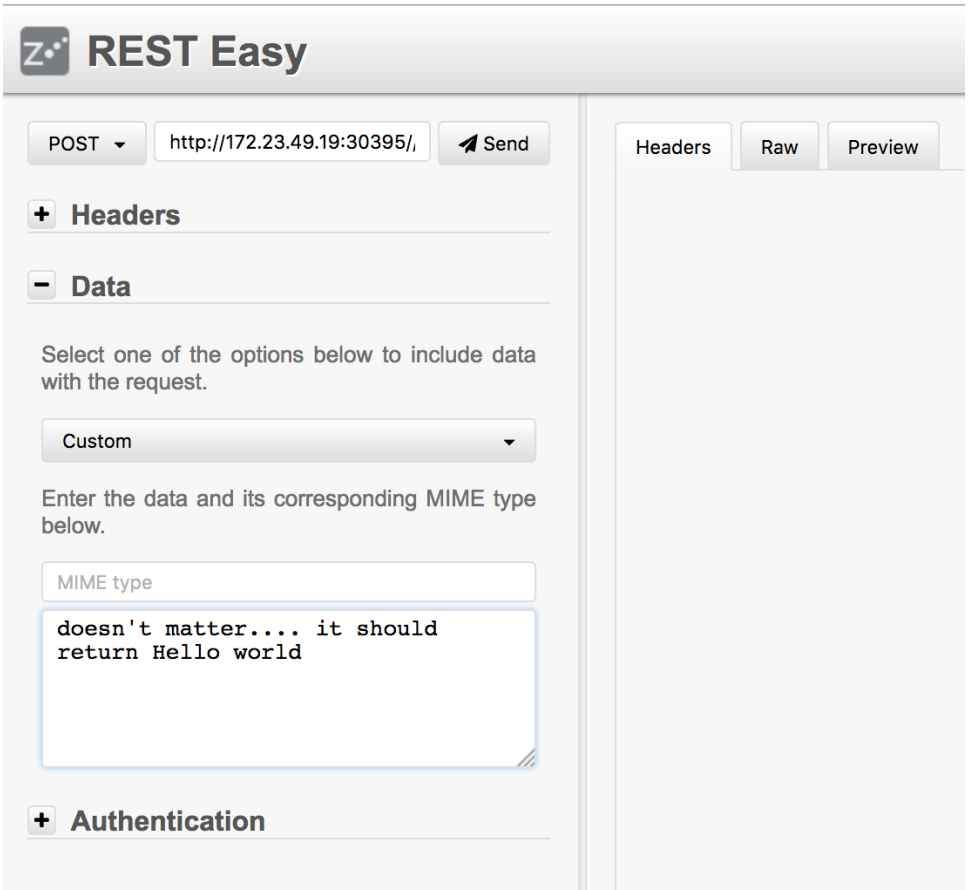
From the ICP Console identify the IIB server listener port for HTTP connections



In this example 7800 (the IIB HTTP listener port) is represented by 30395

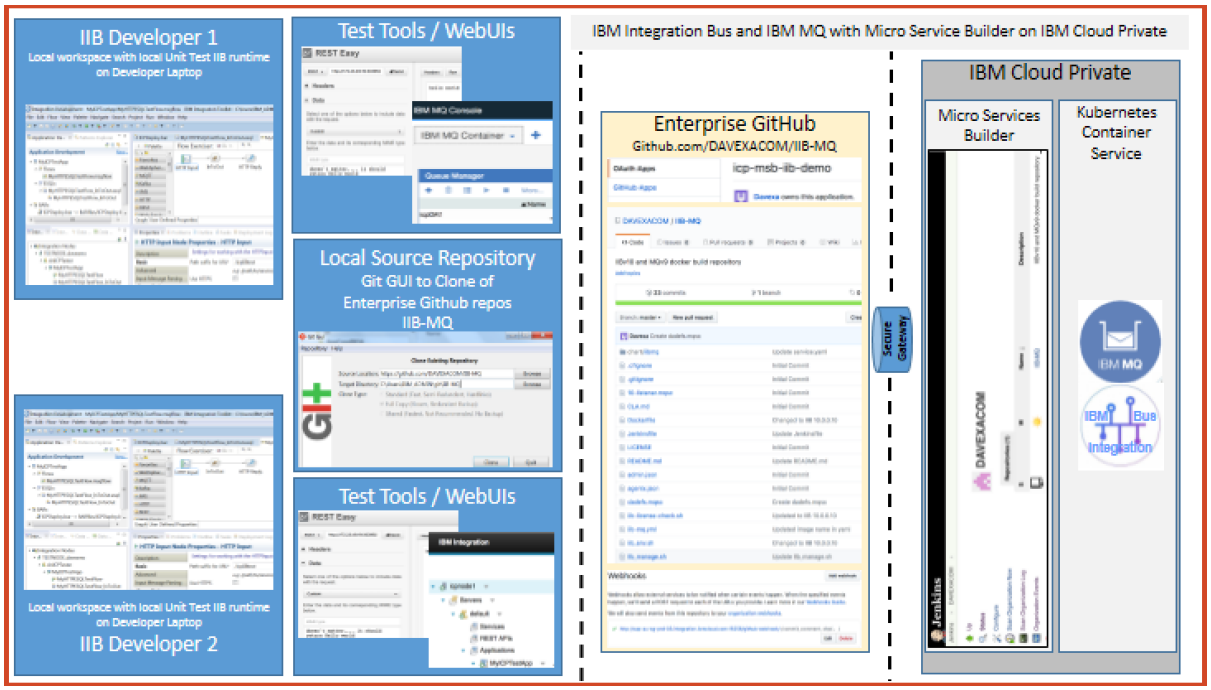
Using an HTTP/REST test client like RestEasy for example

URL: http://172.23.50.111:30395//icpIIBtest



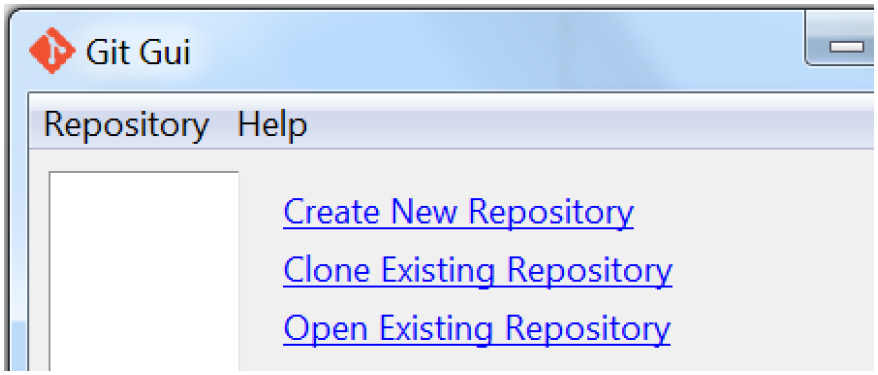
Automating the Build end to end

Solution Overview Diagram

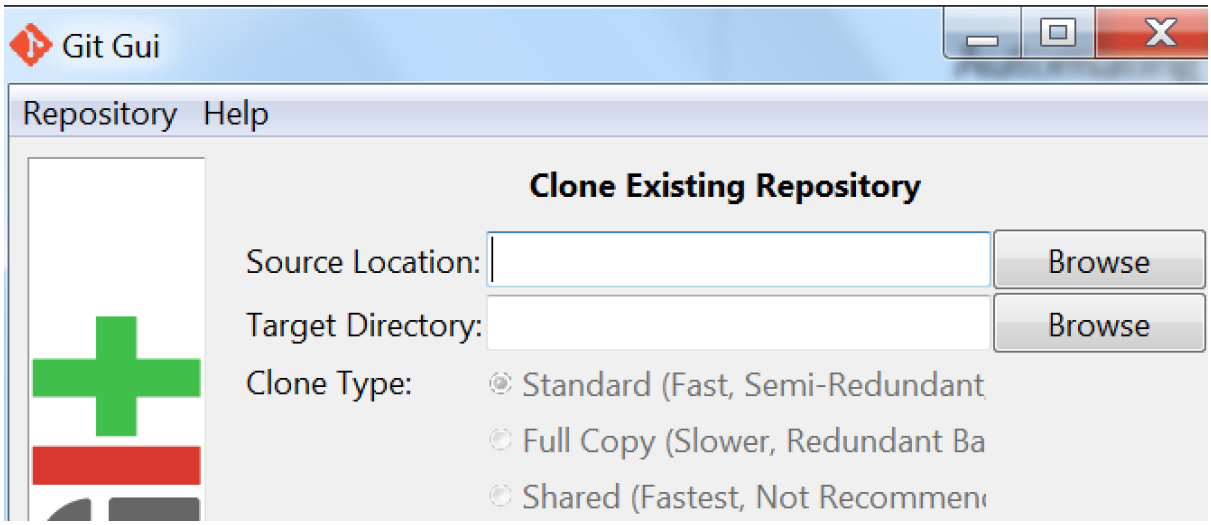


Creating a local clone of the GitHub Repos IIB-MQ

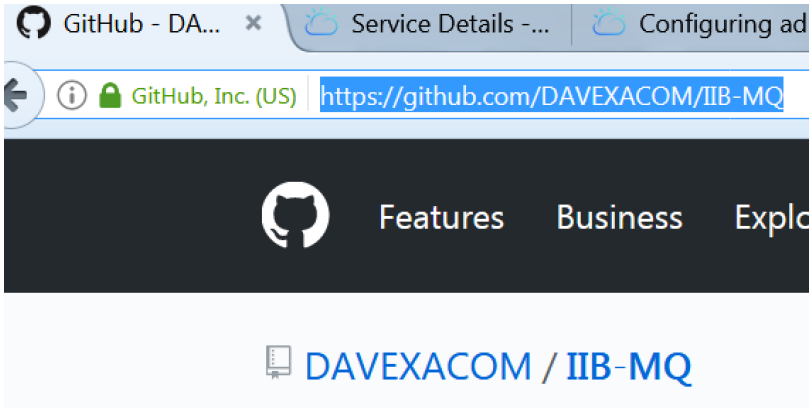
Example using GitGUI



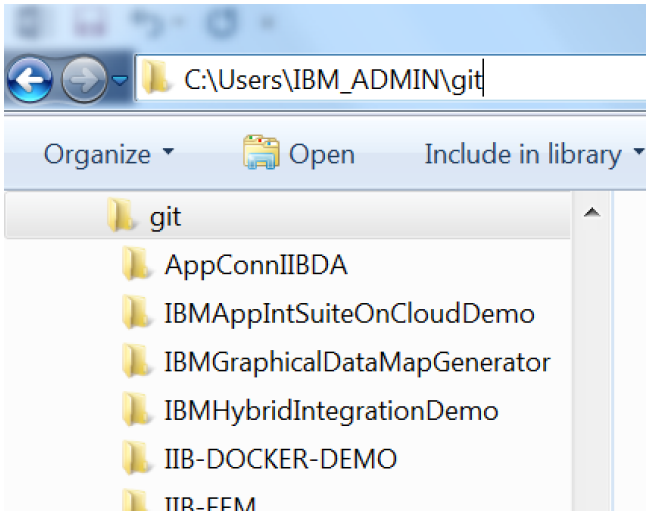
Select Clone Existing Repository



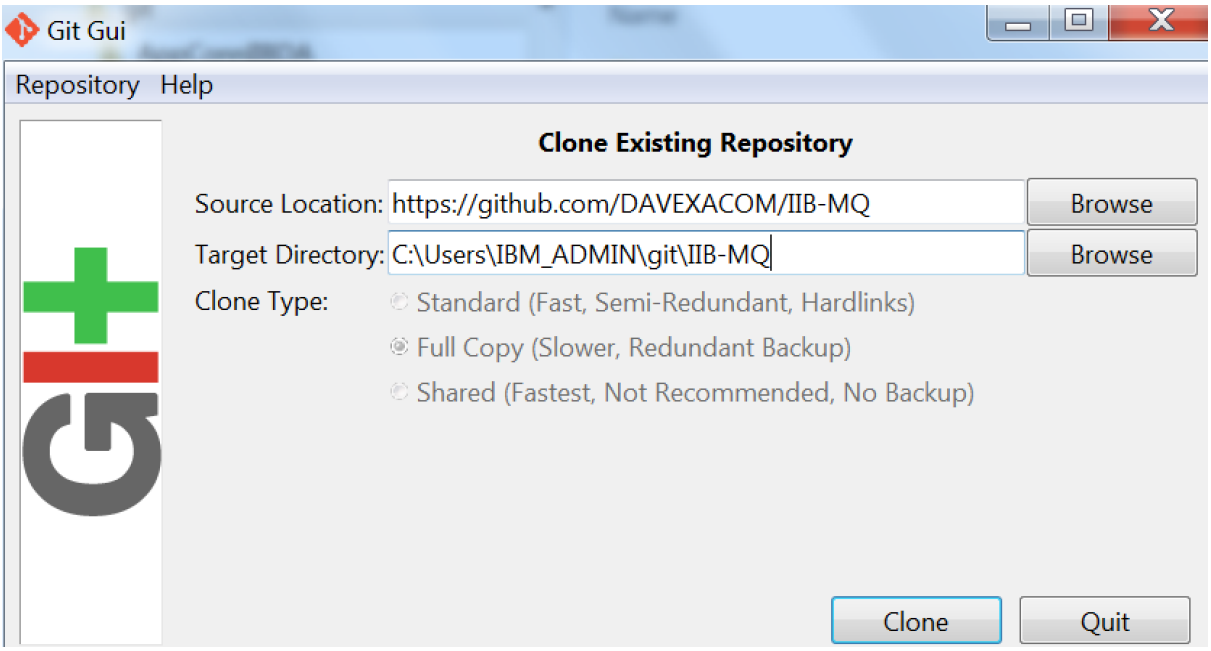
Cut and paste the URL from the GitHub repository to the Git Gui Source Location



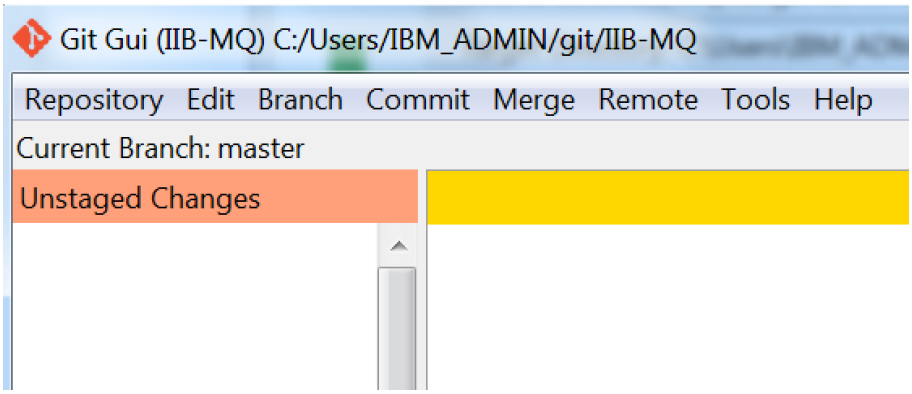
Ut and paste the directory path to your local Git Client to the Target Directory… for example



Then add the “IIB-MQ” and hit clone



You now have a local clone.



Creating the GitHub WebHook to drive Jenkins

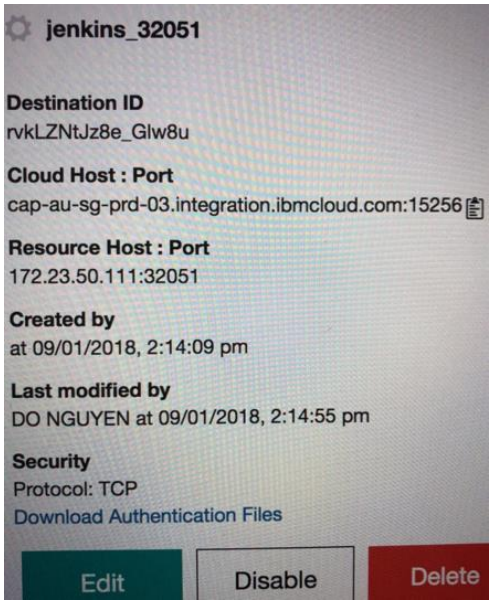
Optional – Use IBM Secure Gateway to connect Github to ICP on a Private Network

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).

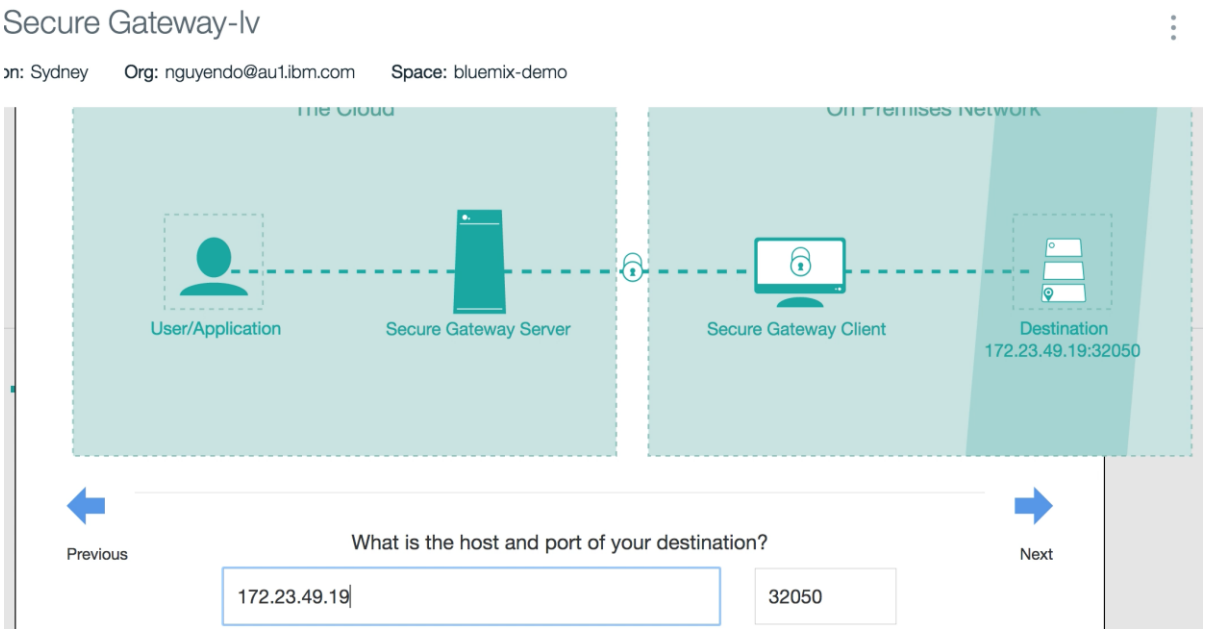
Here is the secure gateway I am using



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

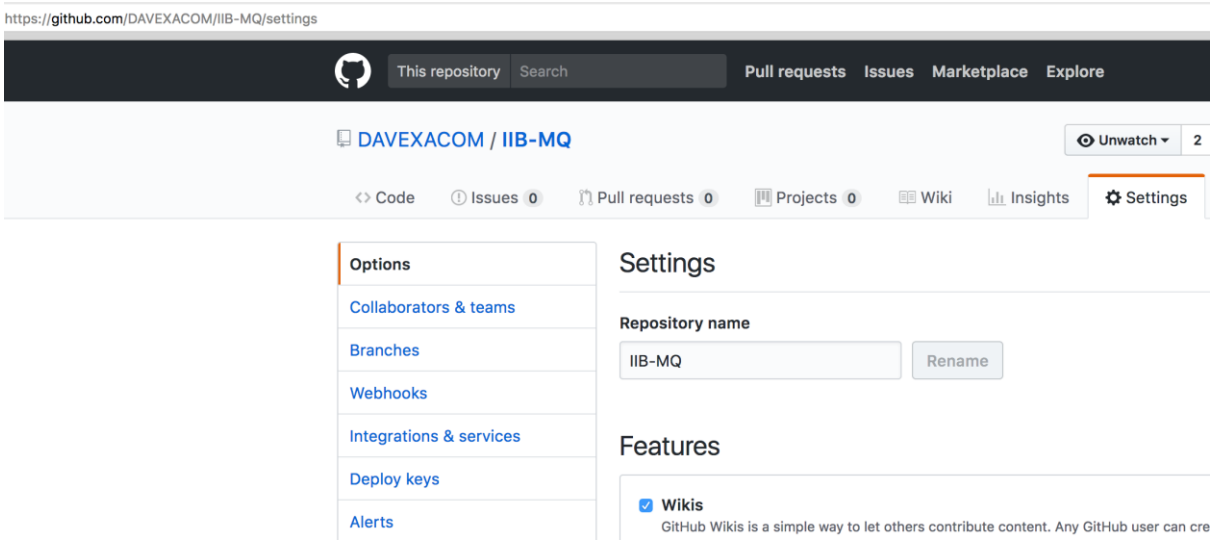
You can use the kubeclt command to pull the IBM Secure Gateway image and push it into ICP. You will then need to ssh into the container and set up the ACLs. In a similar way to that shown below (IP/Ports are not a match for this lab doc)

```
-bash-4.2# docker run -it ibmcom/secure-gateway-client r32C030W8kd_prod_au-syd -t eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
eWQiLCJpYXQiOjE1MTE4NDMyOTksImV4cCI6MTUxOTYxOTI5OX0.Iy0YWunAg-20ajiVDeeYK1lEBzoAXlNpTULonsP5Ja8
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
--------------------------------------------------------------------------------------------
```

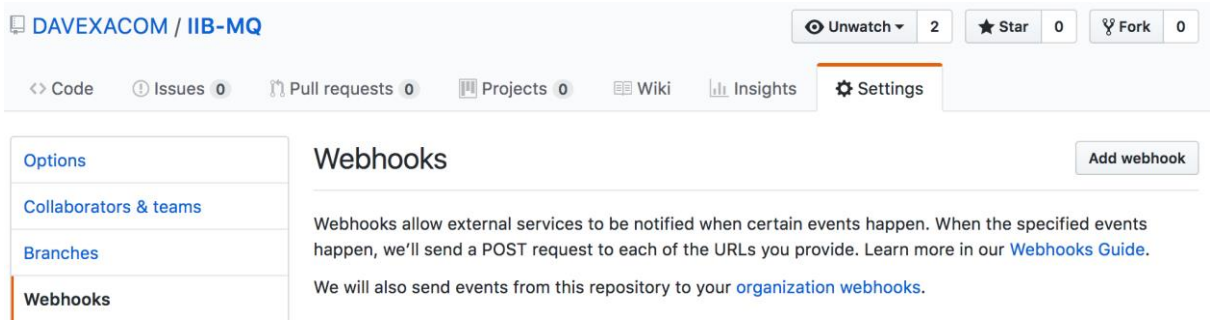


Setting up the Webhook in Github

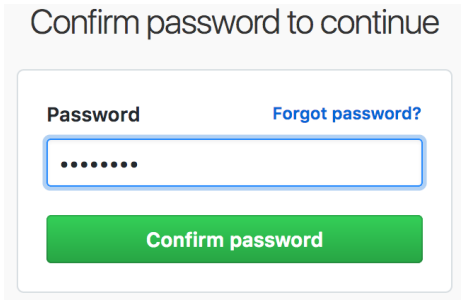
Go to your Github browser, and your organization (DAVEXACOM in this example) and select the IIB-MQ repos. Click on settings.

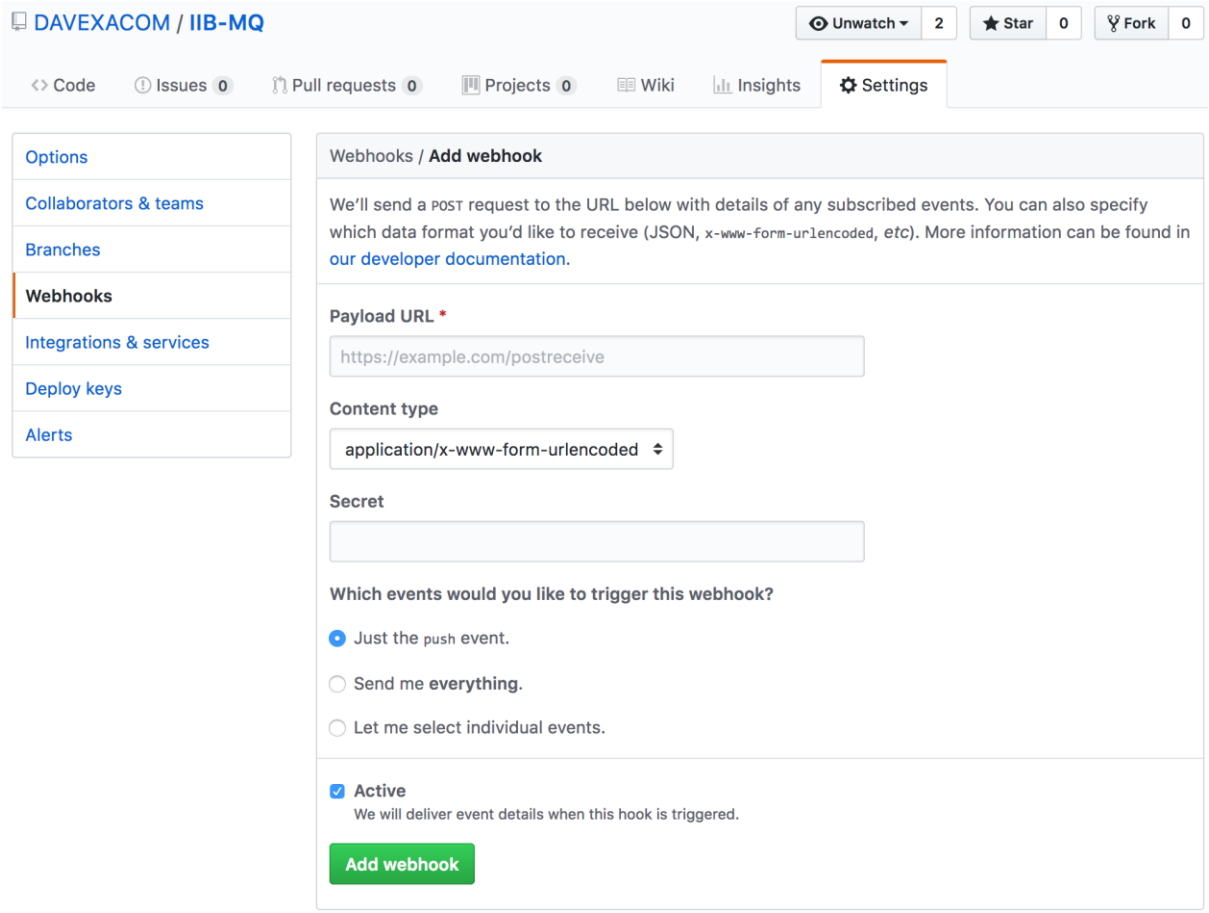


Select Webhooks->Add Webhooks



Confirm password if prompted



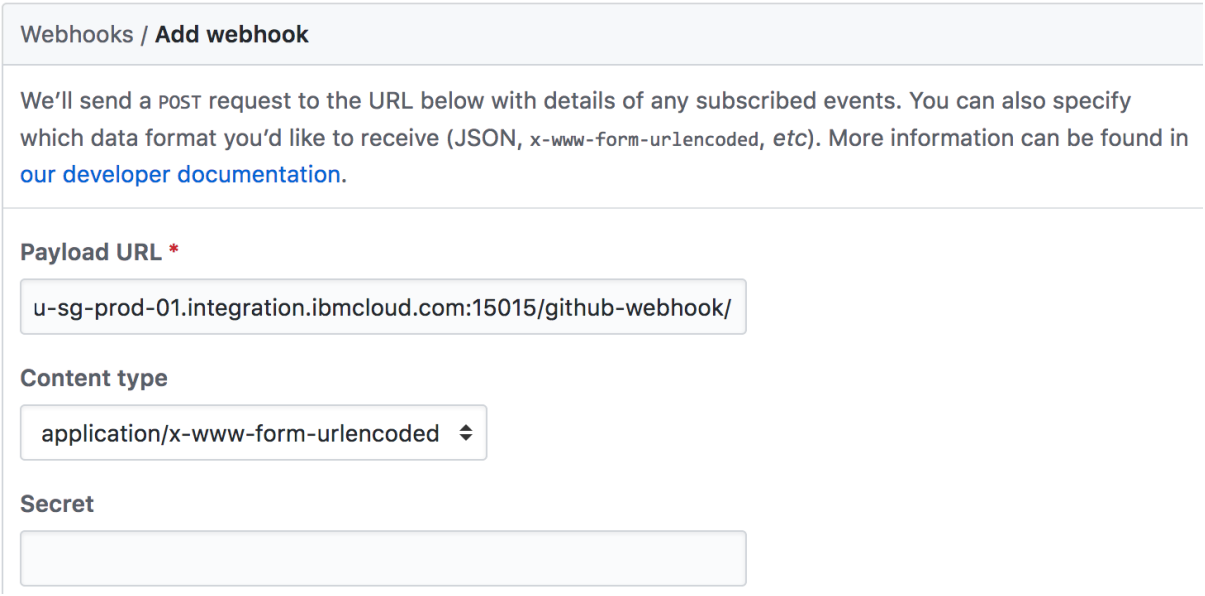


The Payload URL is the address and port number of the Jenkins instance on ICP. However, because in this case ICP sits on a private network we will use the public URL offered by the IBM Cloud (Bluemix) secure gateway service to resolves to the ICP jenkins endpoint.

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



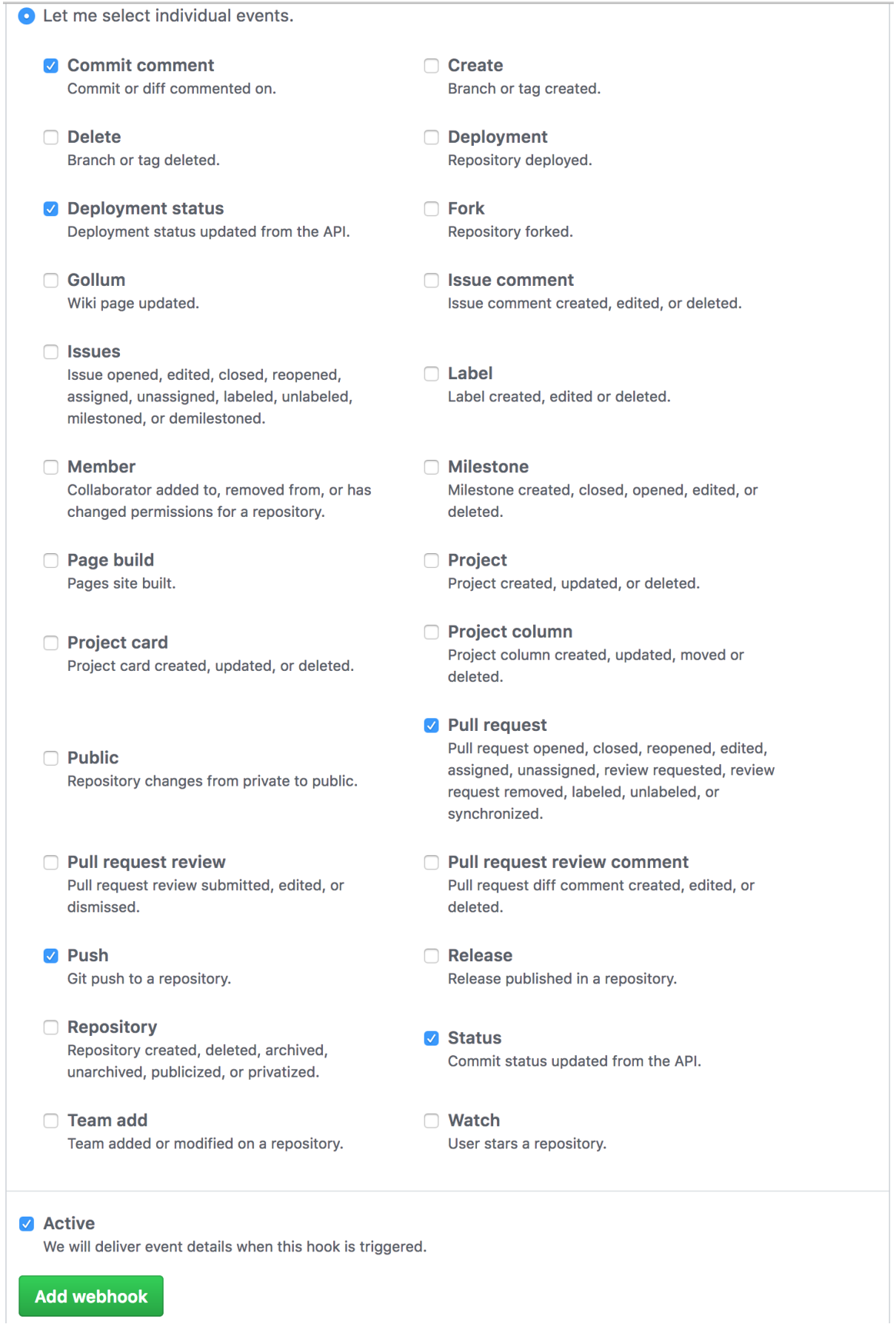
Next select which events to cause the webhook to trigger.

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



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
```

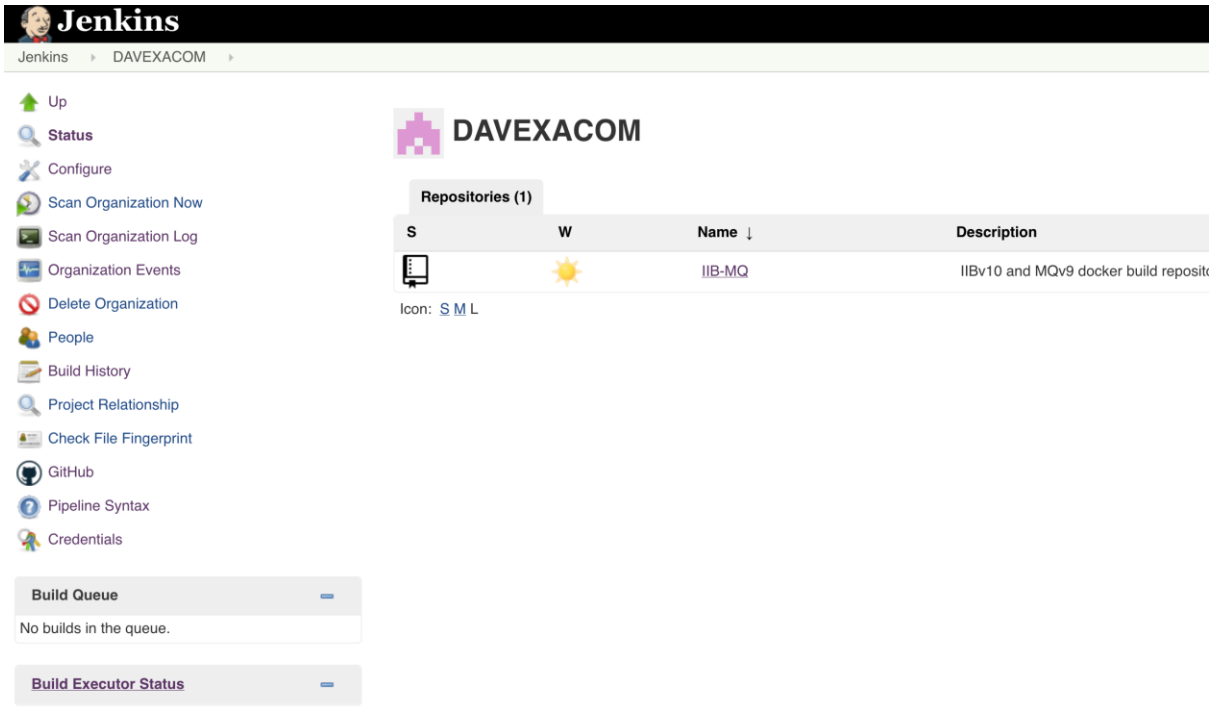
Payload

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

Testing the automated build process

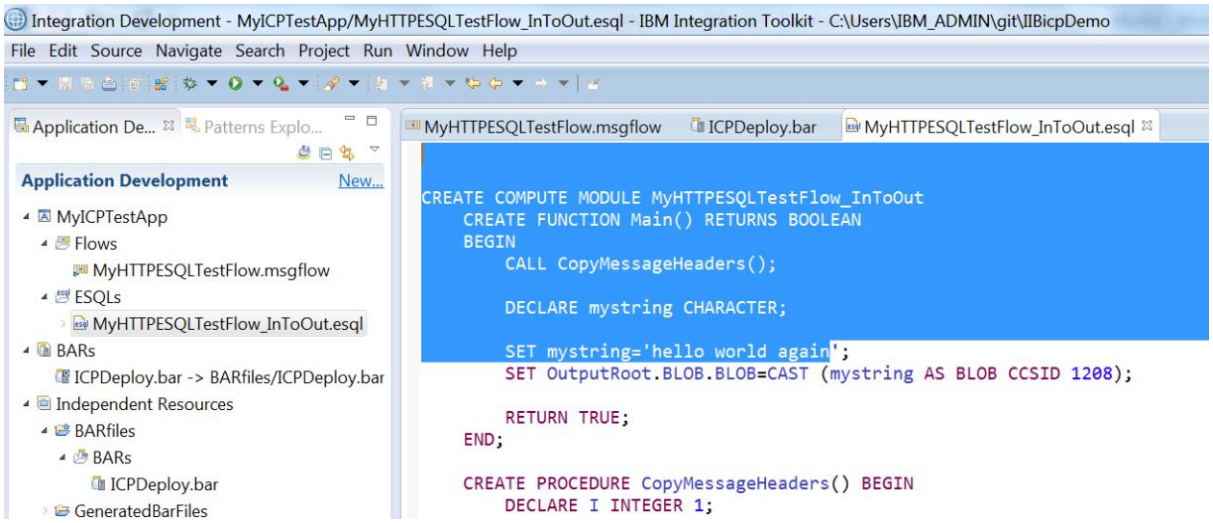
Review Jenkins Status

There should be no builds in the queue at this point

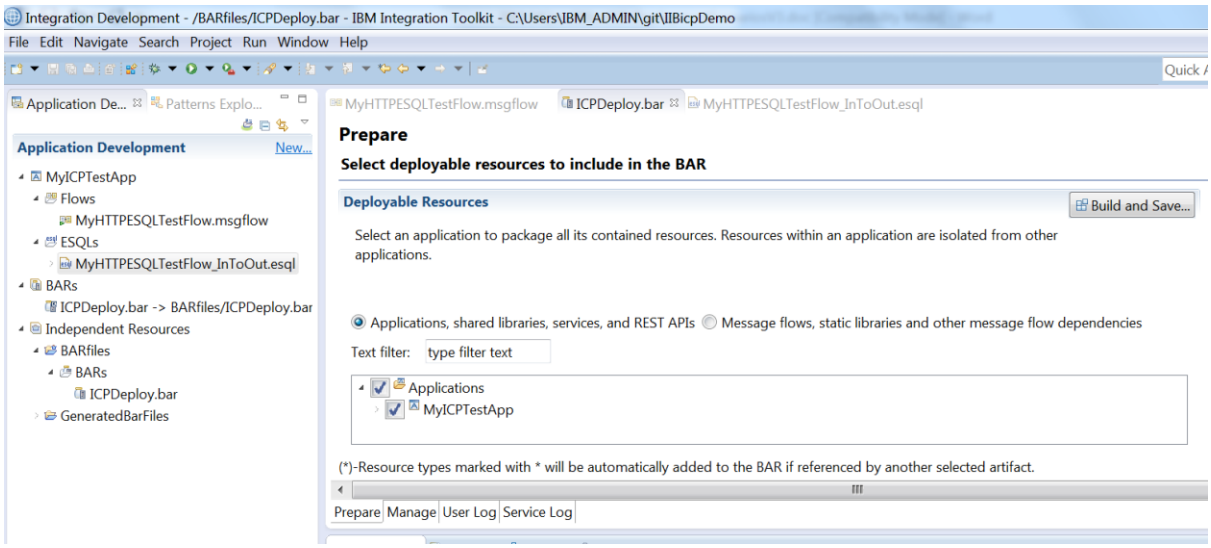


Update the IIB message flow

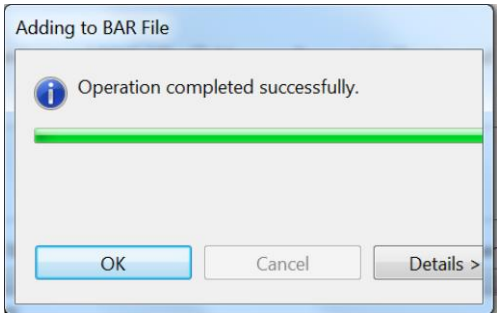
Make a change to the IIB message flow that will be deployed into the ICP IIB runtime as part of the build. Change “Hello world” to “Hello world again” for example.



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



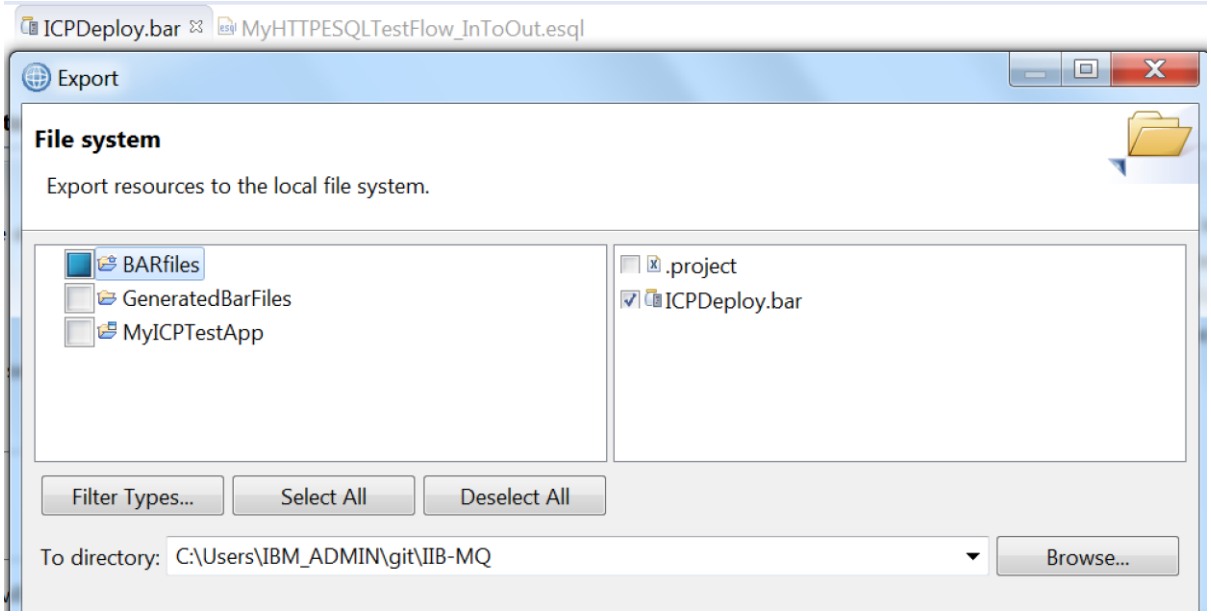
Build and save the BAR file.



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

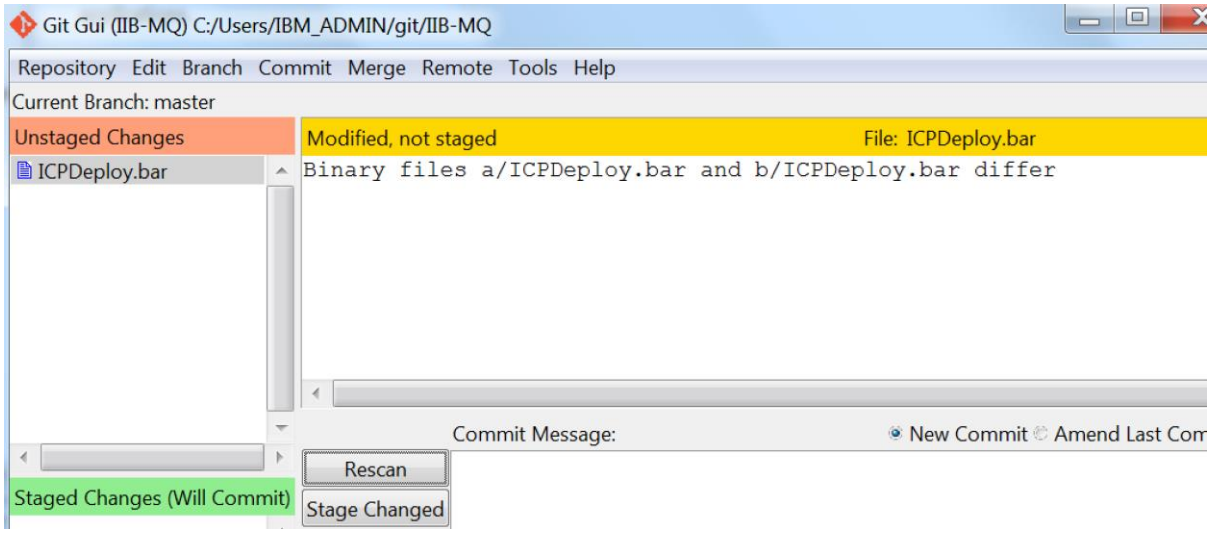
Export the updated BAR file to the File System. File->Export->File System



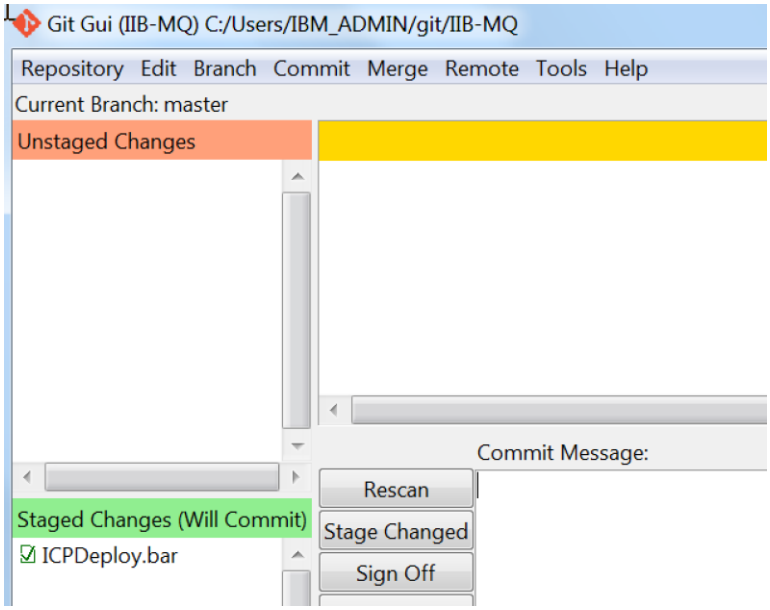
Set the target directory to your local clone of GitHub IIB-MQ repository

Push to GitHub IIB-MQ repository

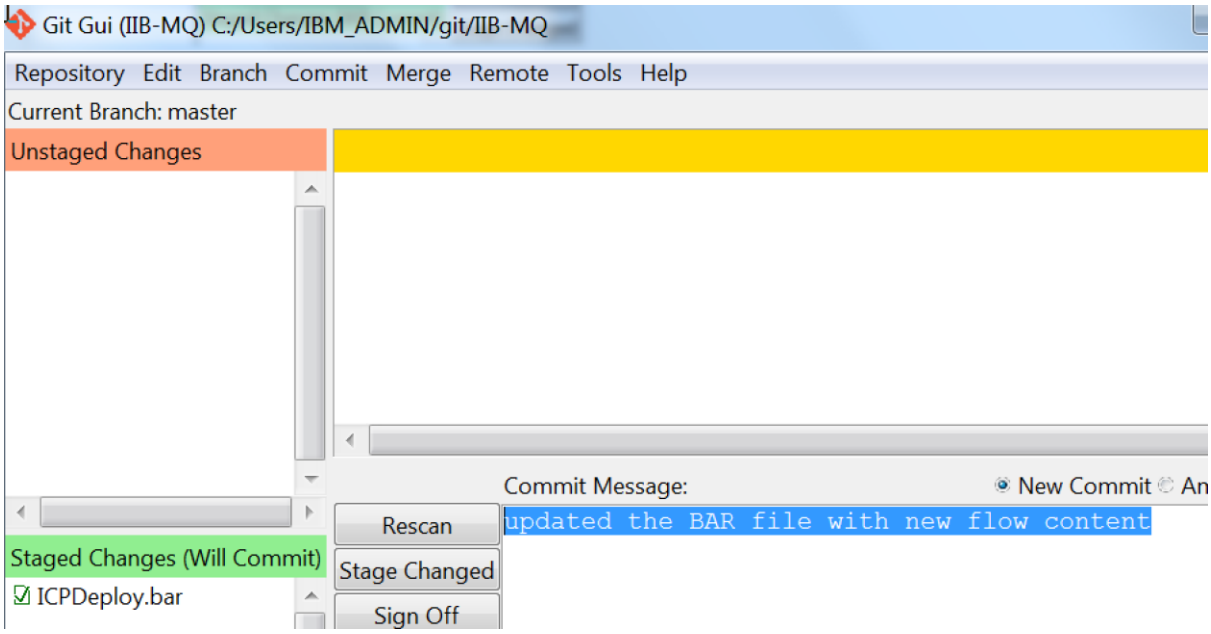
In Git GUI or equivalent hit rescan



Stage changed

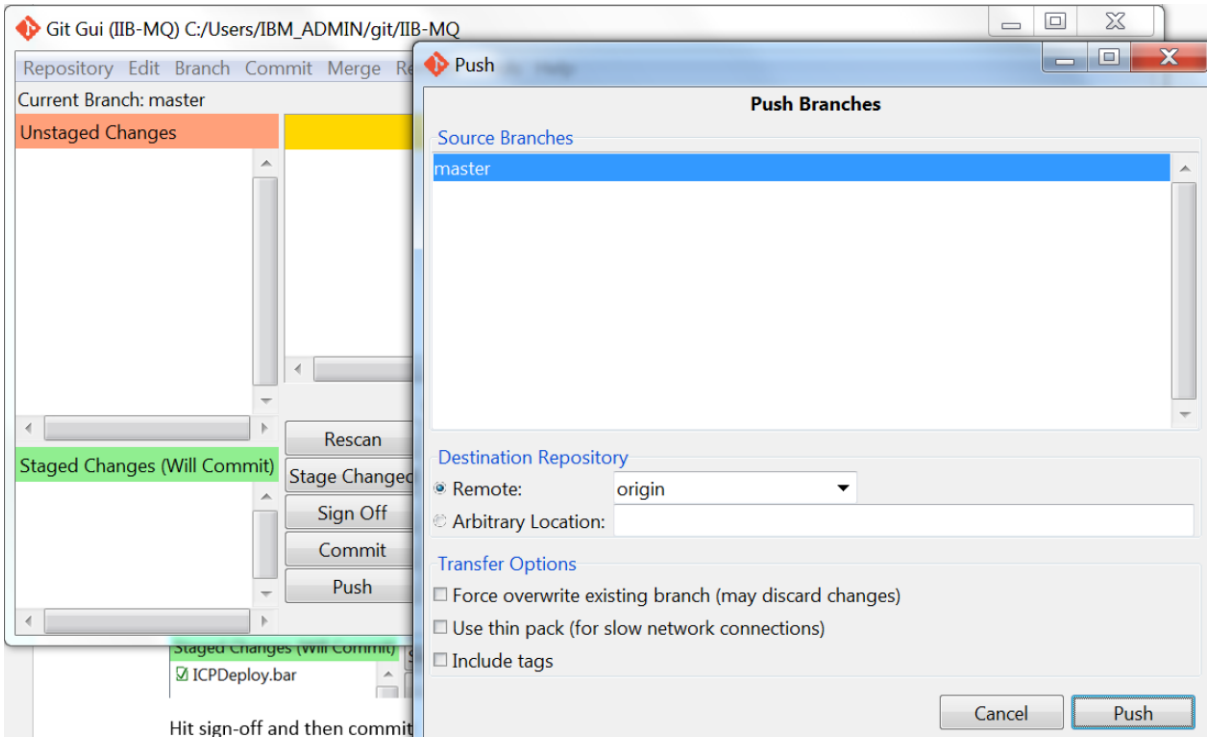


Add a commit message

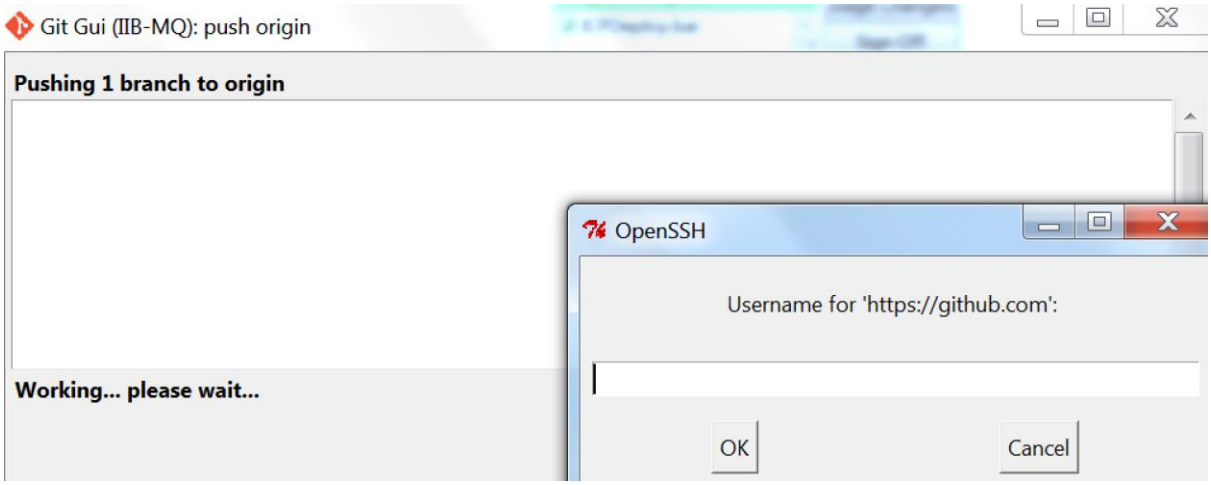


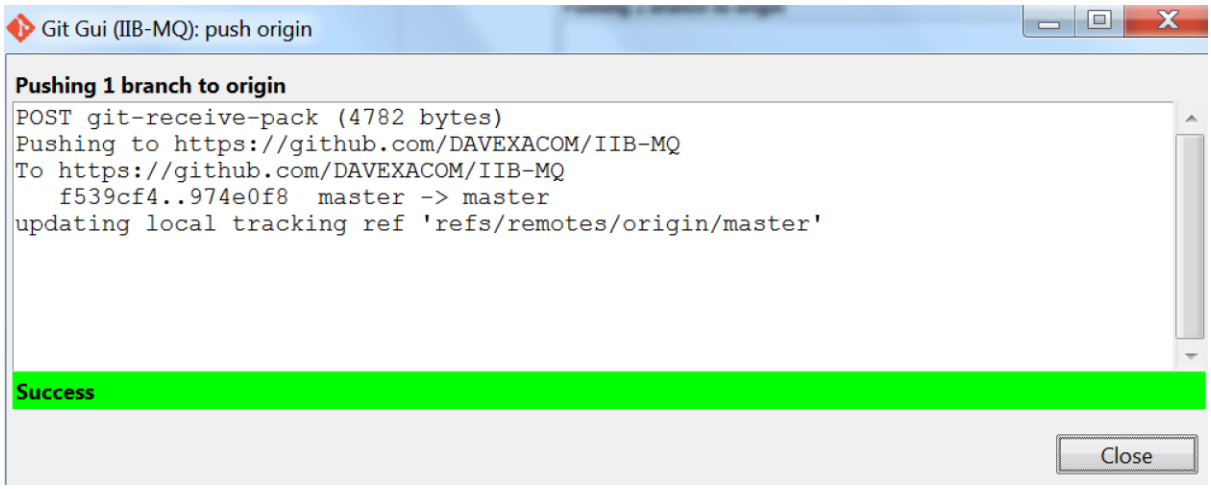
Hit sign-off and then commit

Then start the Push



Sign in with the credentials for your github organization.





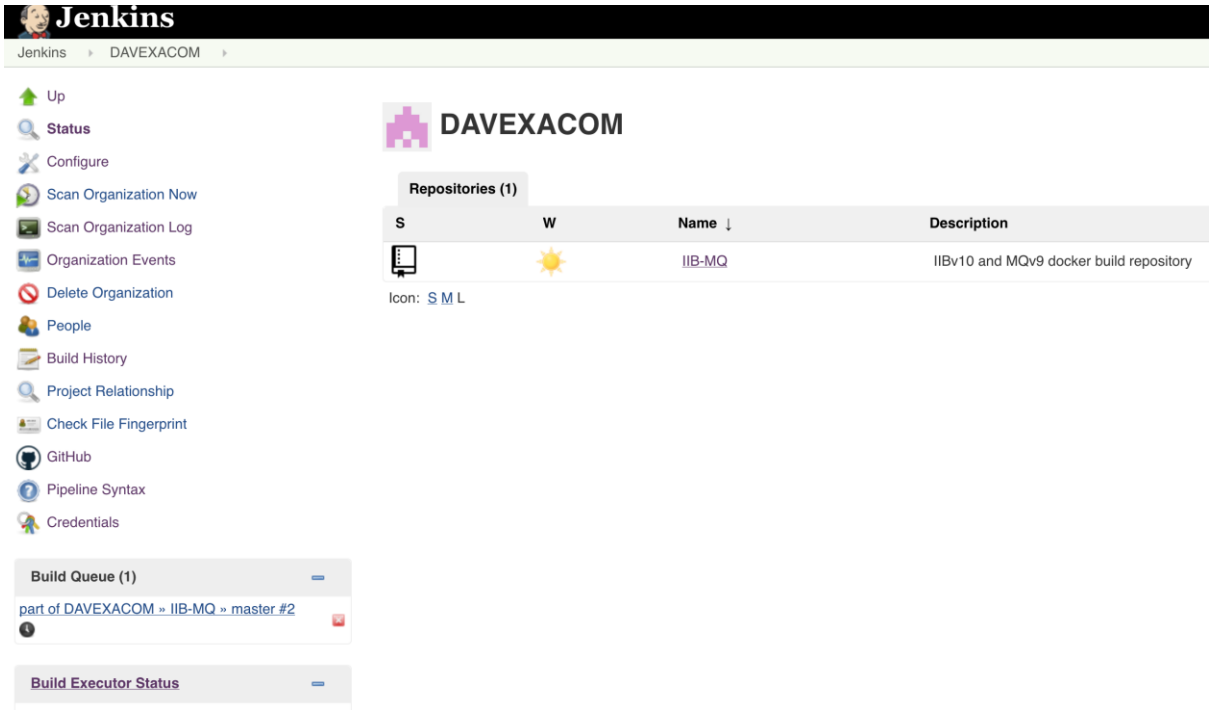
Check the GitHub repository Status

You should see ICPDeploy.bat has been updated

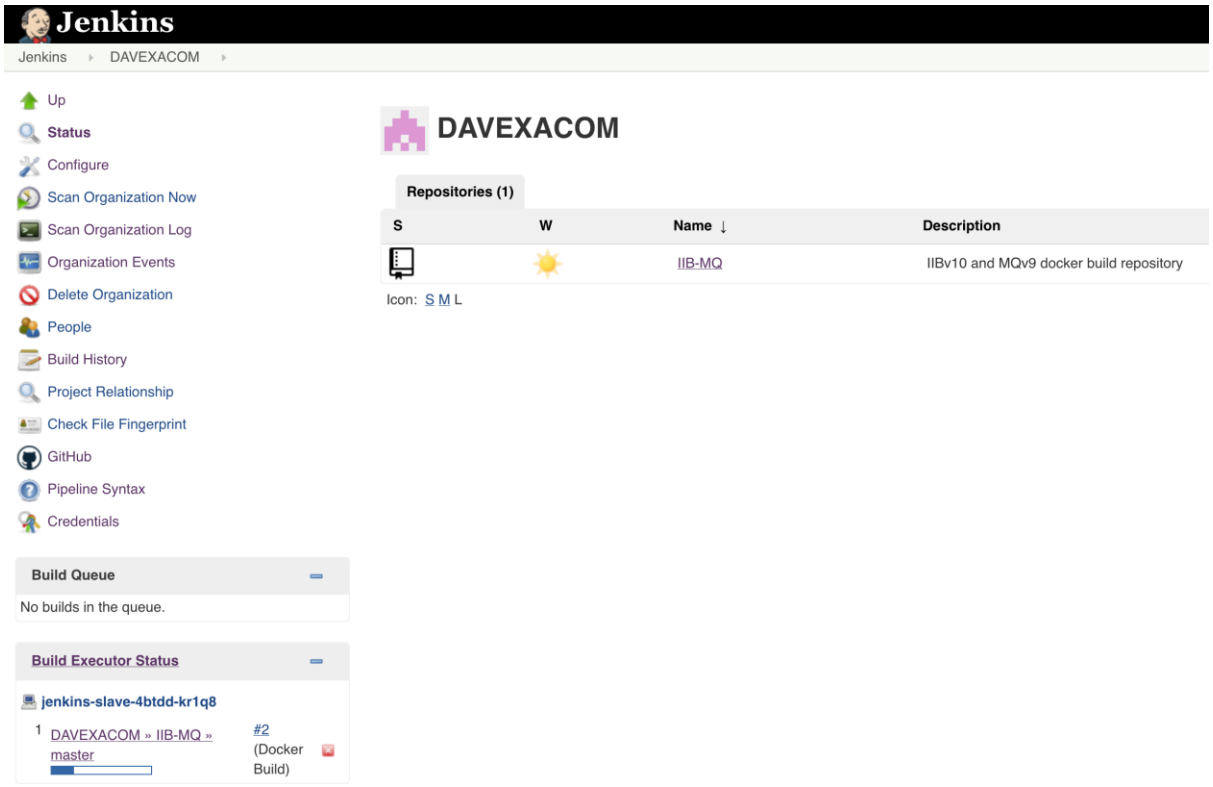
chart/iibmq	Update service.yaml	20 days ago
.cfignore	Initial Commit	9 months ago
.gitignore	Initial Commit	9 months ago
10-listener.mqsc	Initial Commit	9 months ago
CLA.md	Initial Commit	9 months ago
Dockerfile	Update Dockerfile	19 days ago
ICPDeploy.bar	updated the BAR file with new flow content	4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

Review Jenkins Status Again

Things may take a moment to kick off

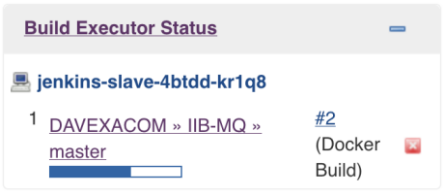


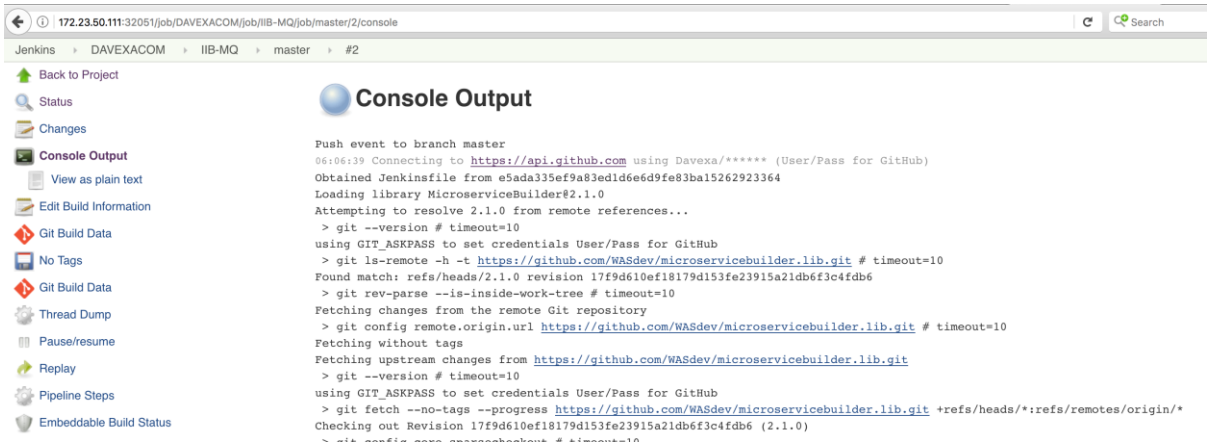
And once an Executor becomes available you should see the following.



Check the Jenkins build logs

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



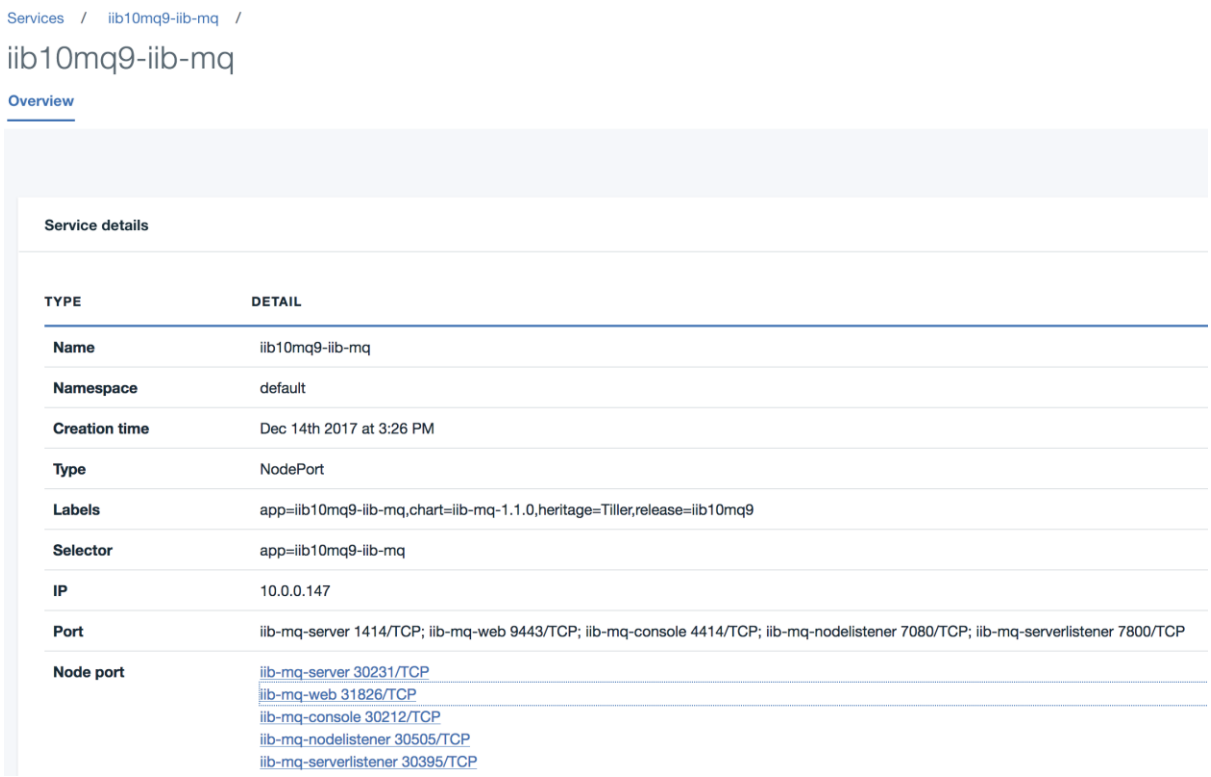


Once complete the tail of the output will look similar to below.

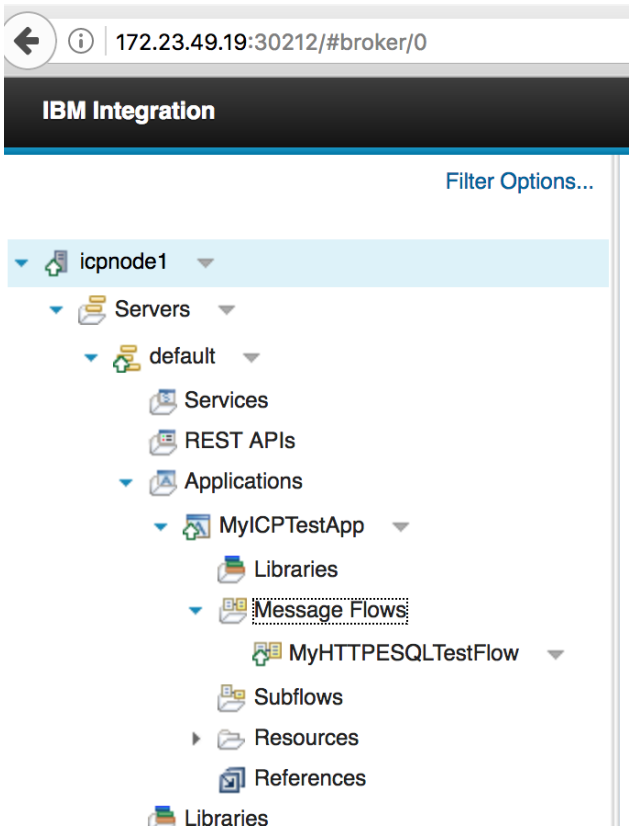
```
[DAVEXACOM_IIB-MQ_master-MPOWJ5QPOSBOGPPW2KNKIZRPJQRLX5DN7HQYK7RV5DPBNS2U3S2A] Running shell script
+ /helm init --client-only --skip-refresh
Creating /home/jenkins/.helm
Creating /home/jenkins/.helm/repository
Creating /home/jenkins/.helm/repository/cache
Creating /home/jenkins/.helm/repository/local
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-MPOWJ5QPOSBOGPPW2KNKIZRPJQRLX5DN7HQYK7RV5DPBNS2U3S2A] 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)

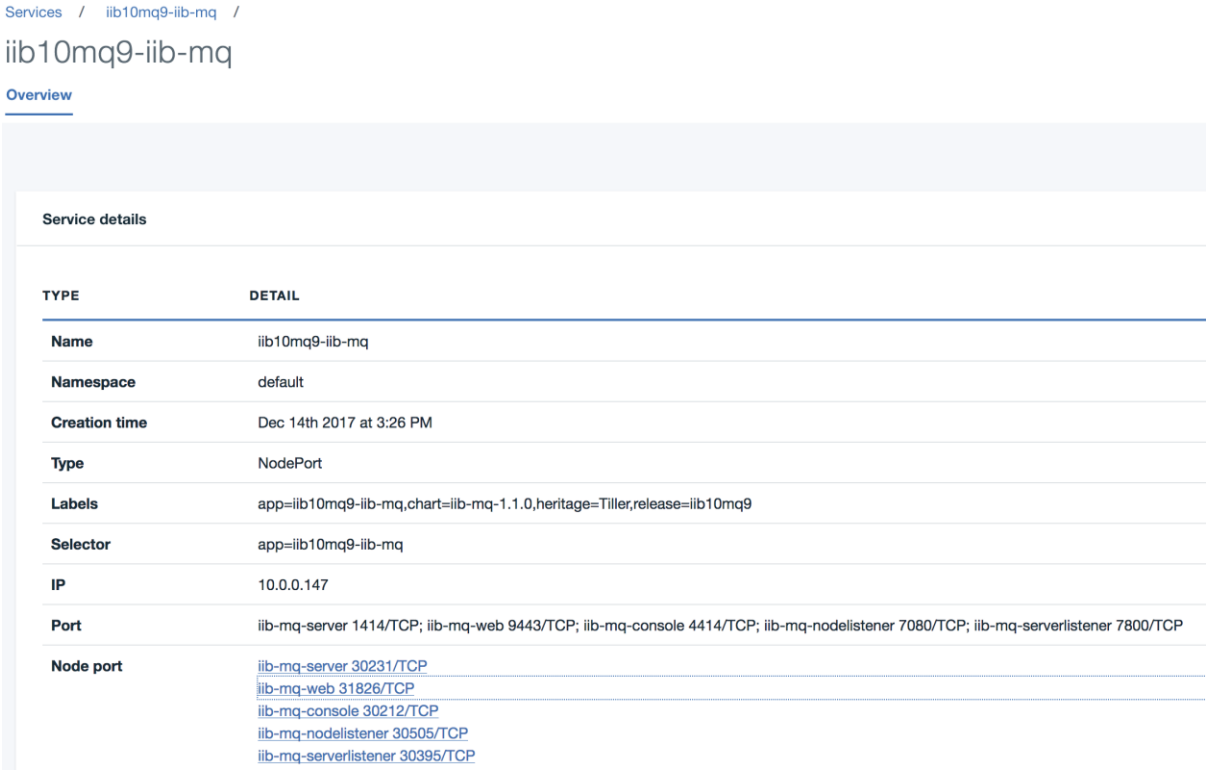


Expand the icpnode1->servers->default->Applications->MyICPTestApp->Message Flows



Test the deployed message flow

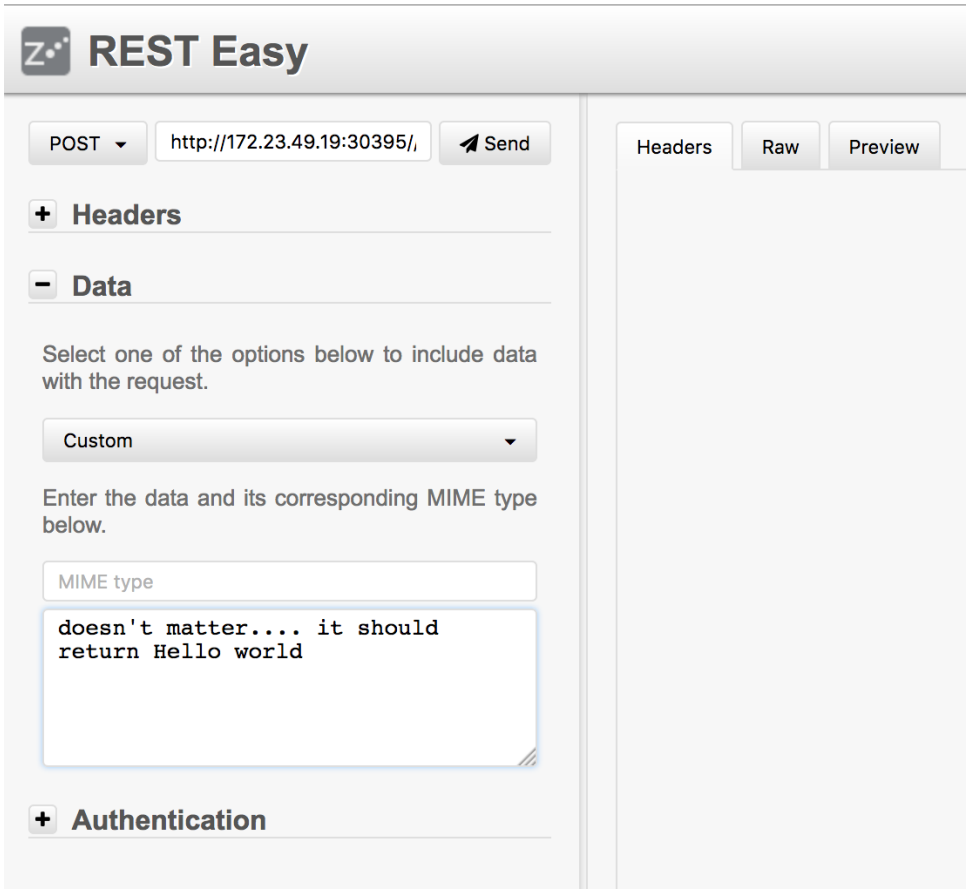
From the ICP Console identify the IIB server listener port for HTTP connections



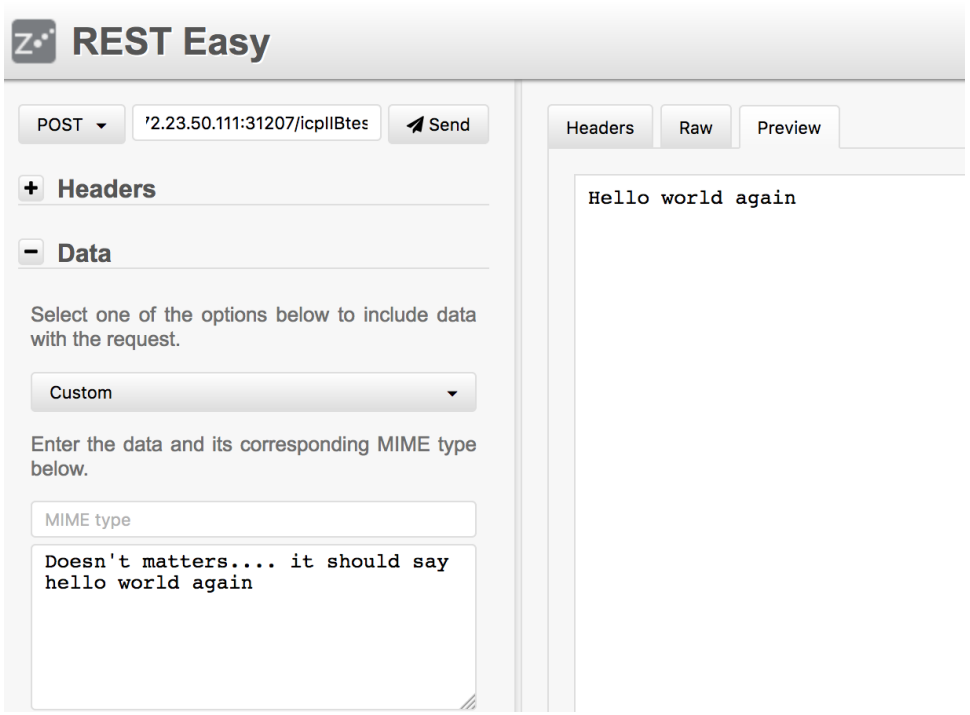
In this example 7800 (the IIB HTTP listener port) is represented by 30395

Using an HTTP/REST test client like RestEasy for example

URL: http://172.23.50.111:30395//icpIIBtest



Hit Send and check the result in preview

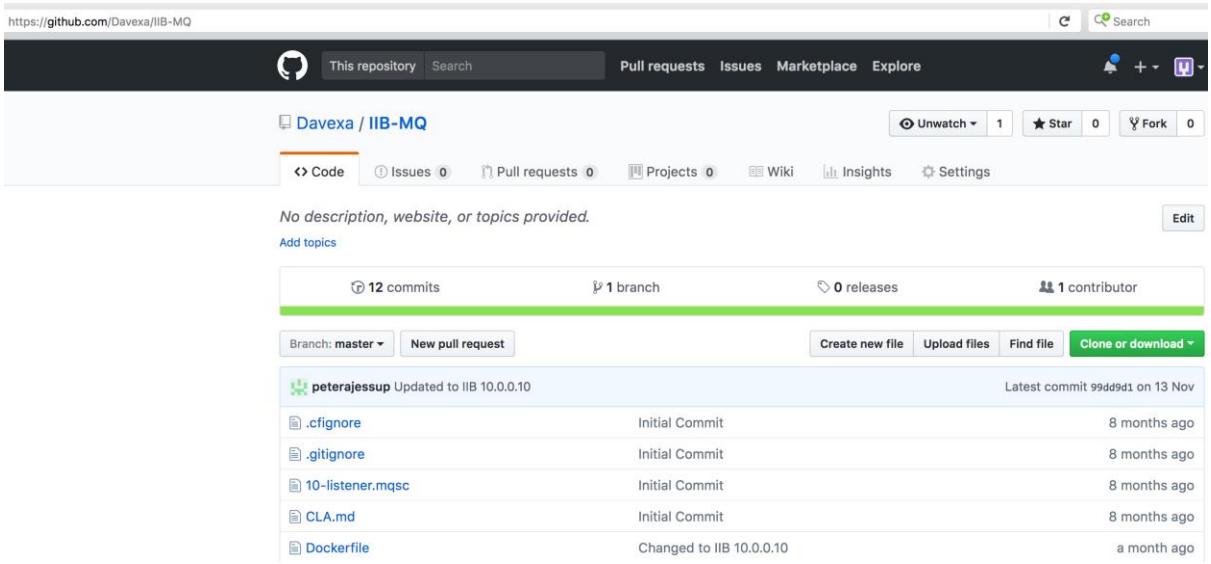


Reference Section

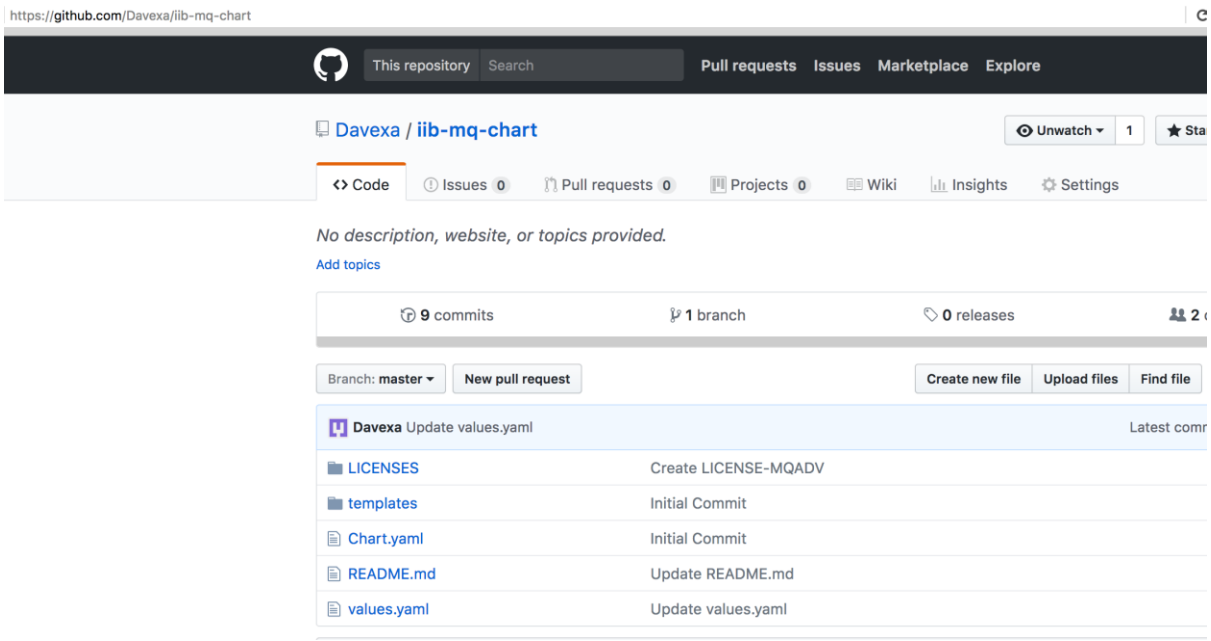
How the DAVEXACOM/IIB-MQ Repository in GitHub was created

Source GitHub Repositories to leverage

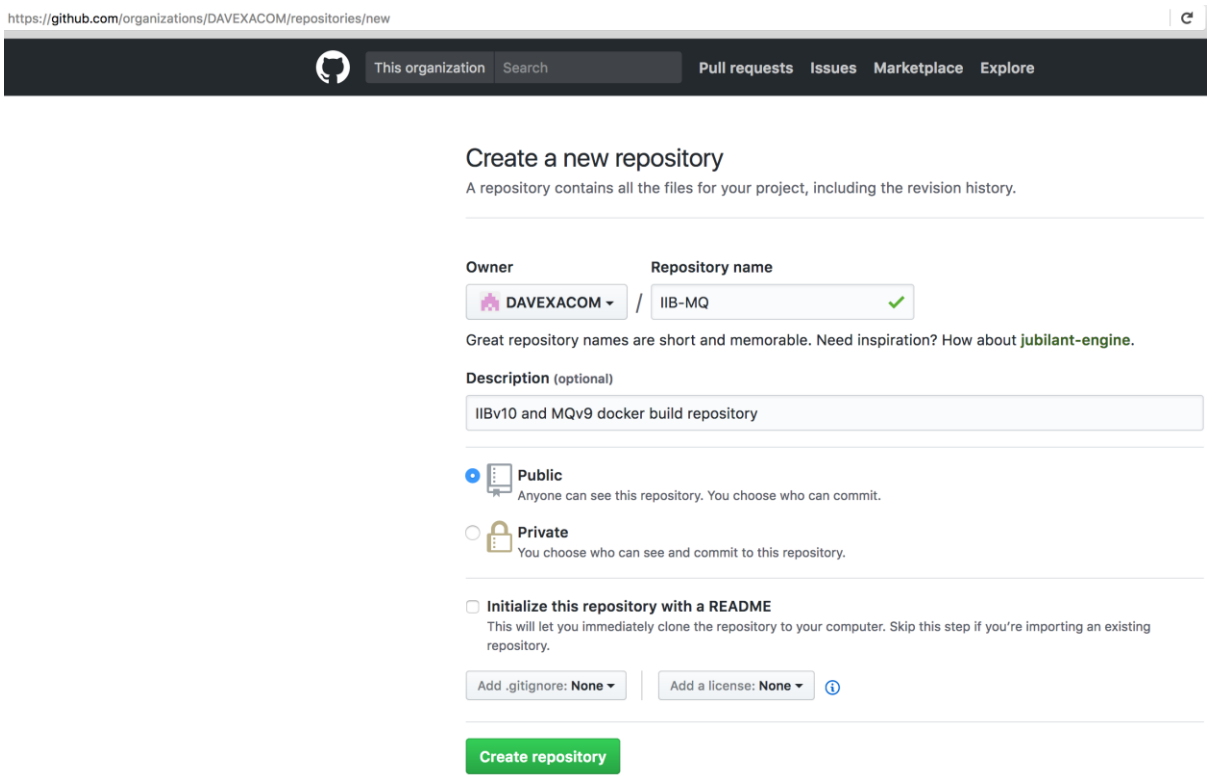
1. IIB an MQ Docker build repository



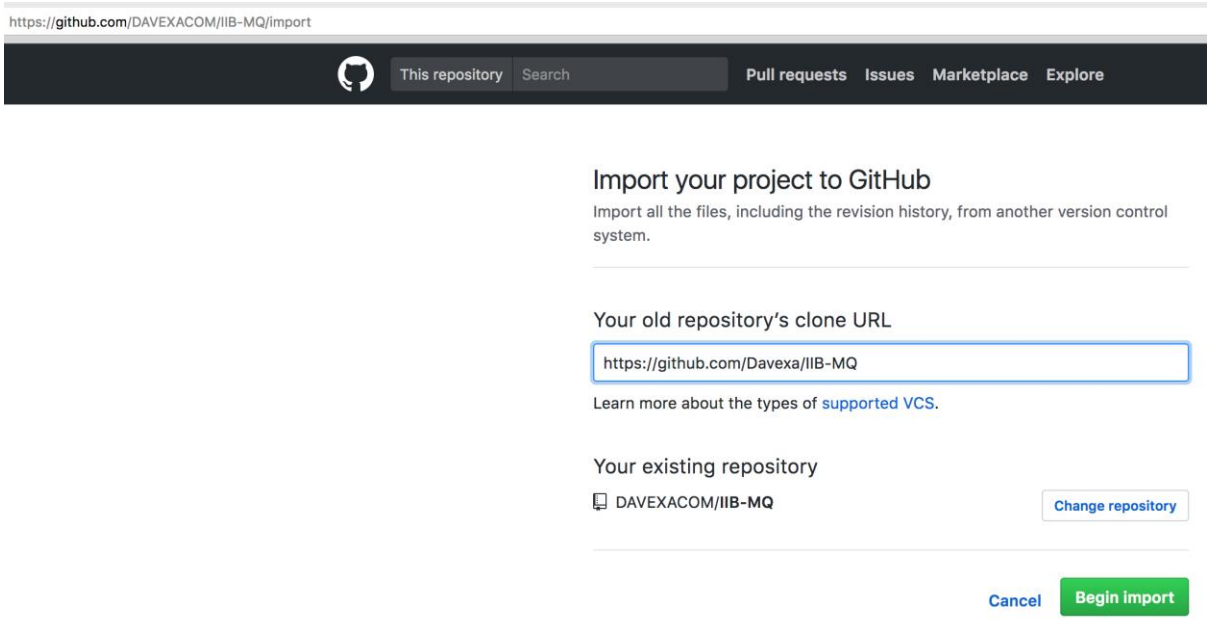
2. IIB and MQ Helm Chart repository

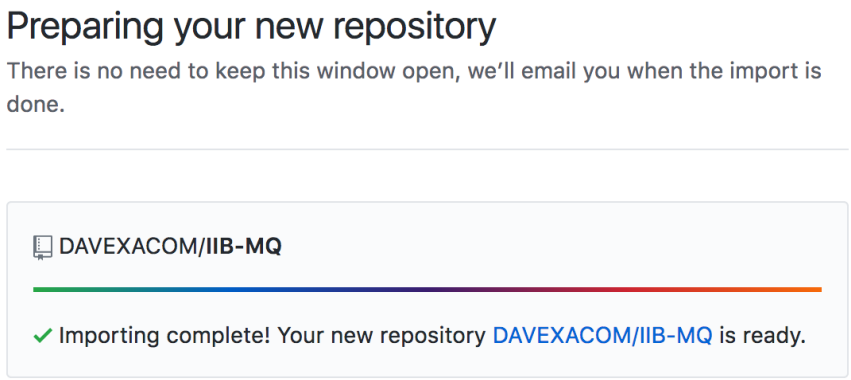


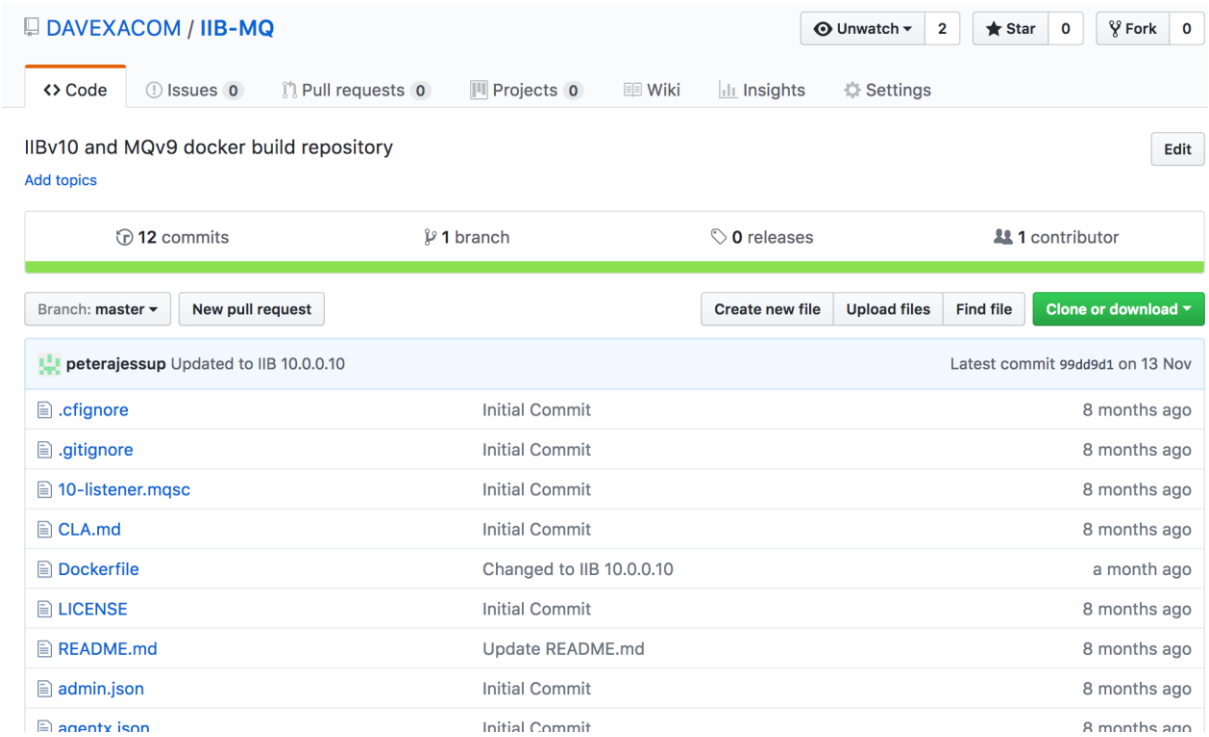
Target GitHub Repository in Github Organization



Import from existing repos

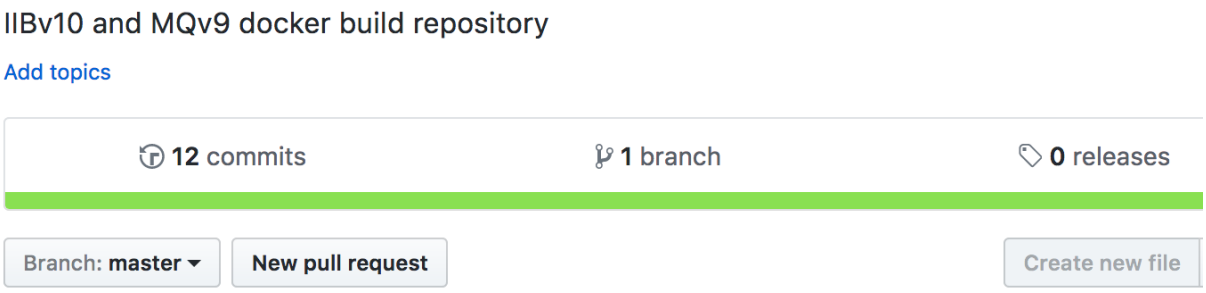






Add a Jenkins file to the repository

Use Create new file to create the Jenkinsfile

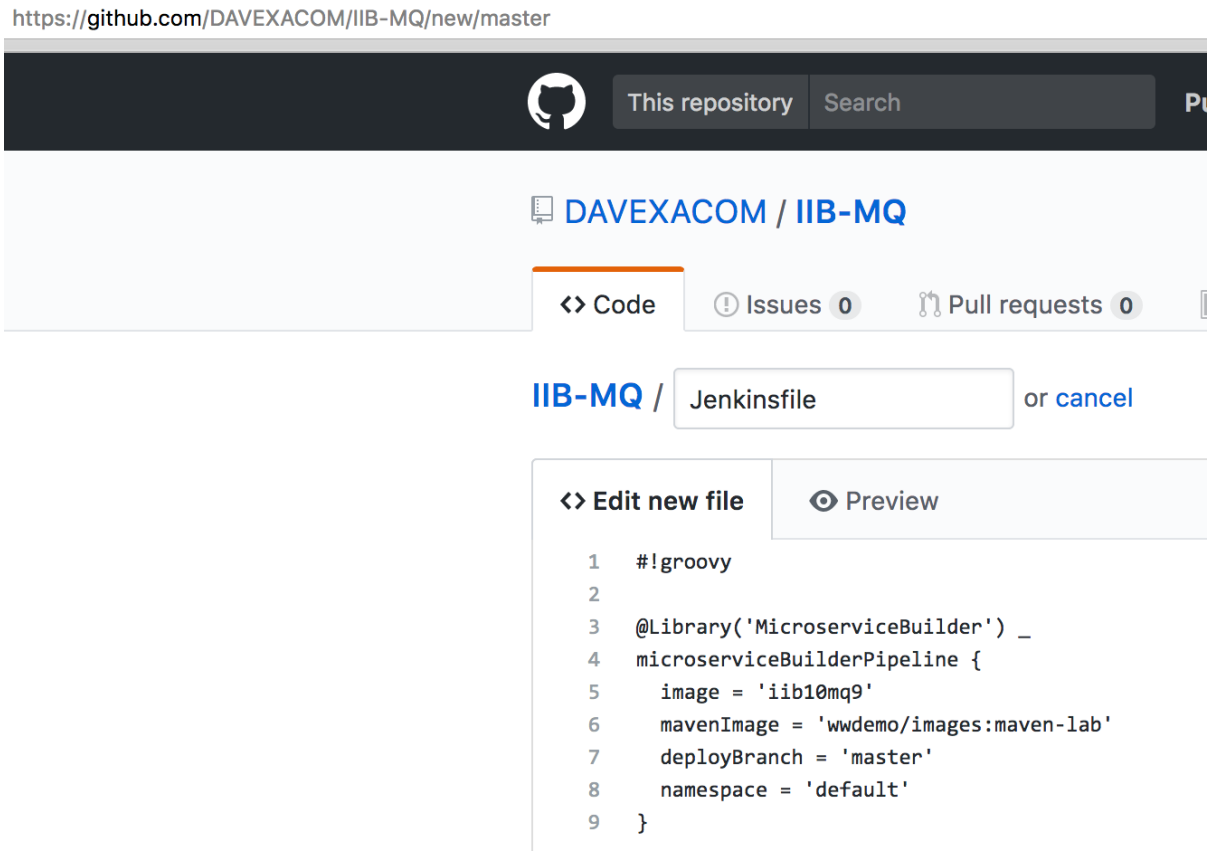


Name the file Jenkinsfile

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

#!groovy

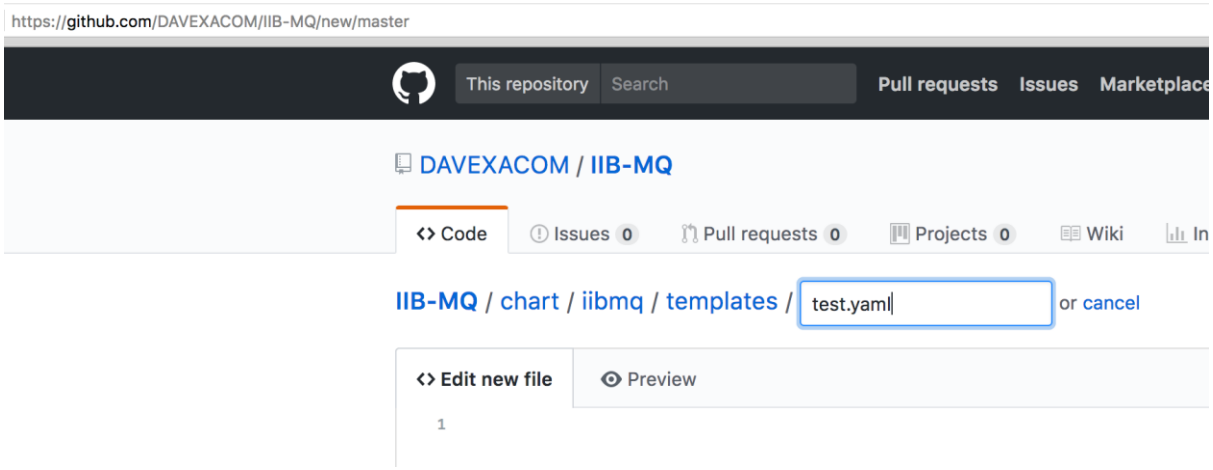
```
@Library('MicroserviceBuilder') _
microserviceBuilderPipeline {
  image = 'iib10mq9'
  mavenImage = 'wwdemo/images:maven-lab'
  deployBranch = 'master'
  namespace = 'default'
}
```

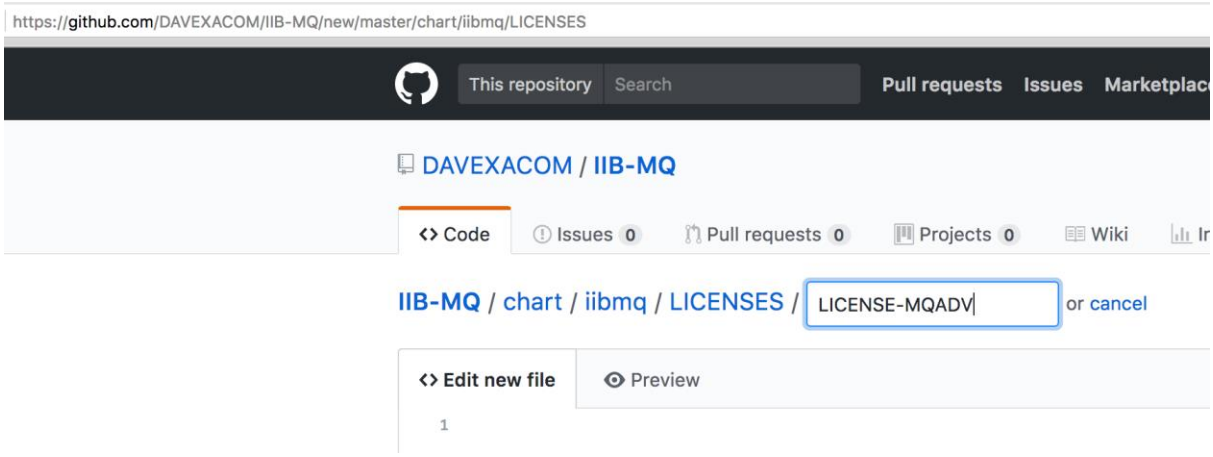


Page down and hit commit

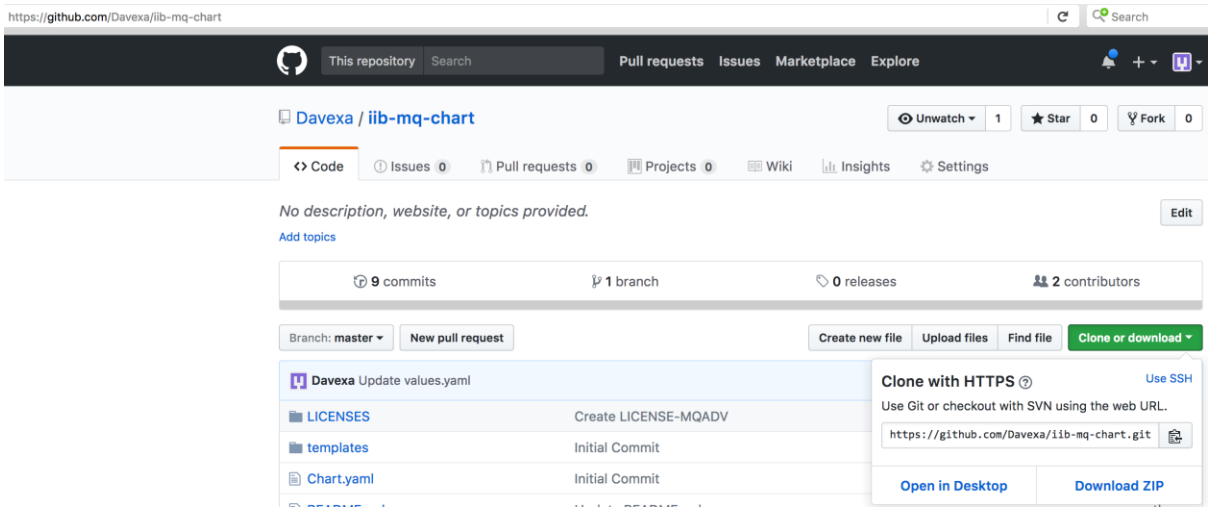
Add the IIB and MQ Helm chart files to the repository

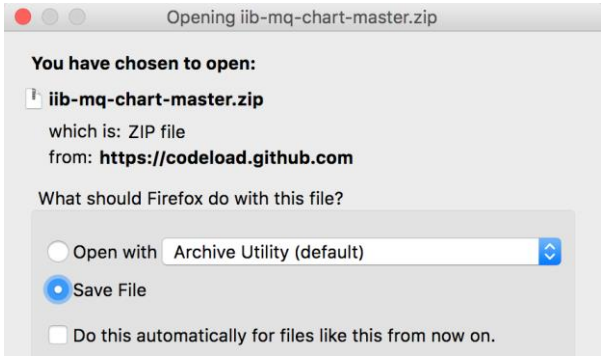
Create the directory structure chart/iibmq/templates and chart/iibmq/LICENSE

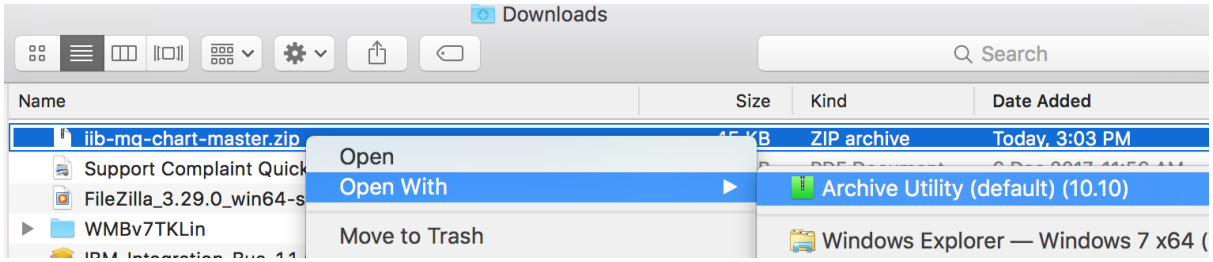


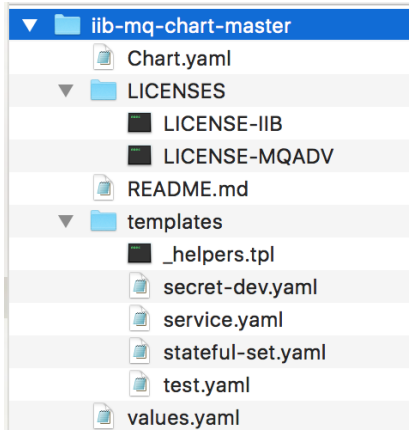


Download from the iib-mq-chart repository to the local file system

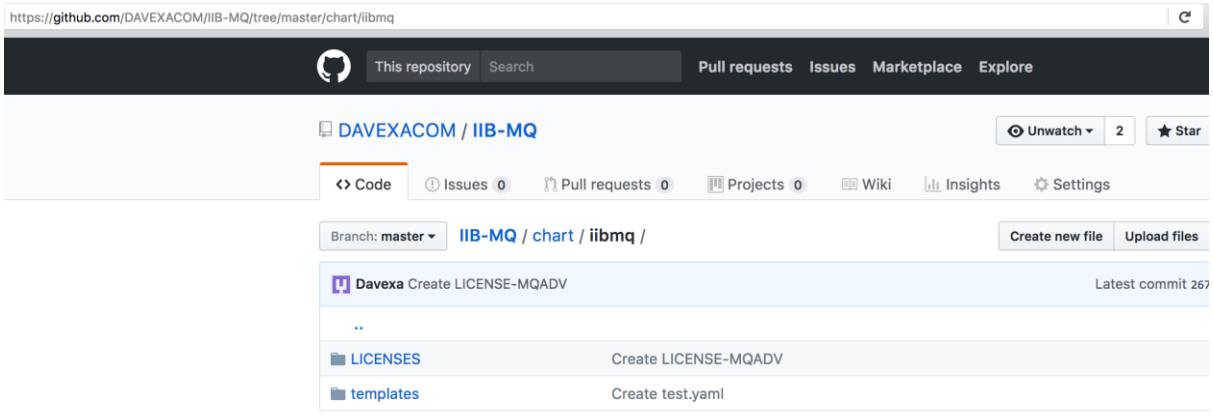


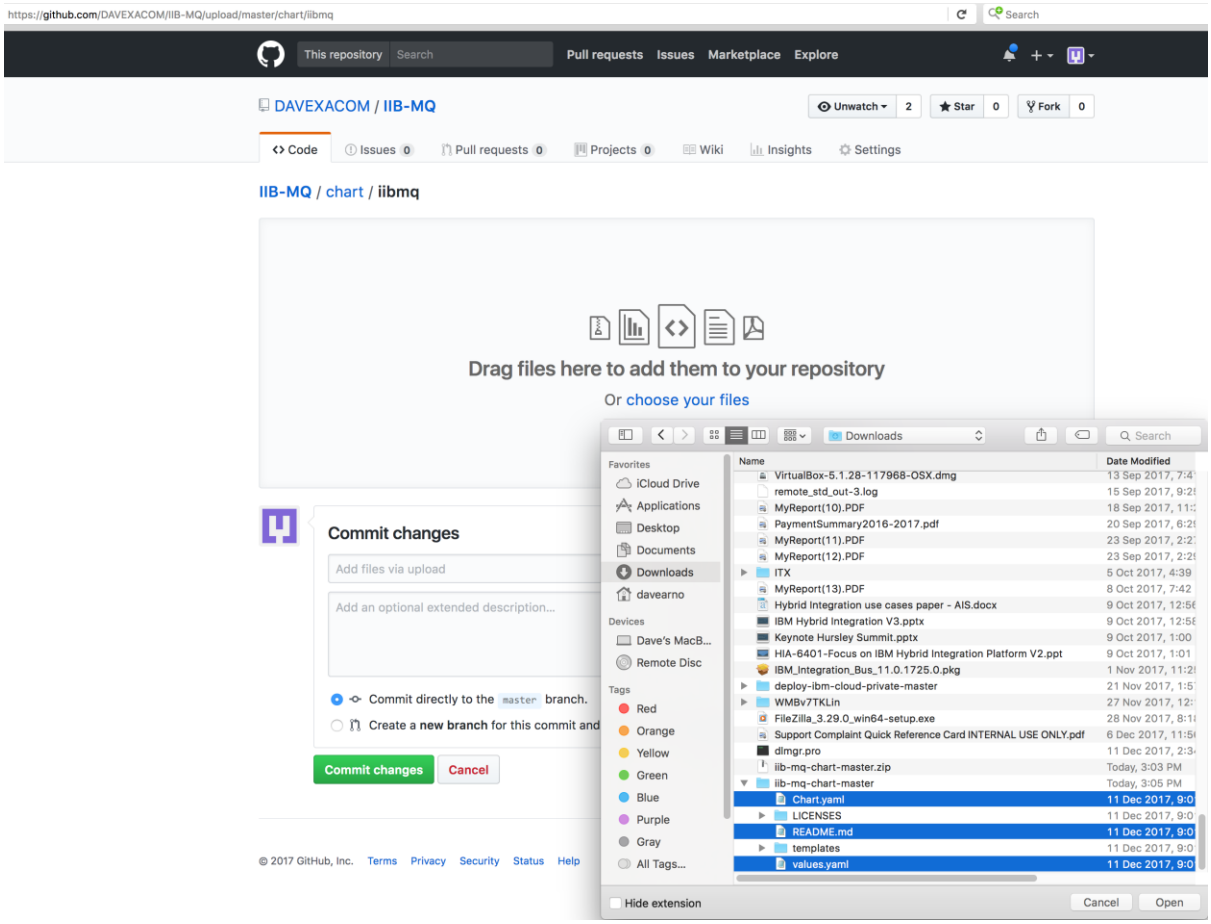




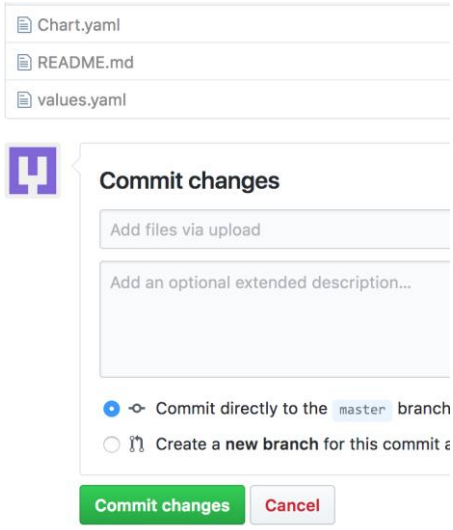


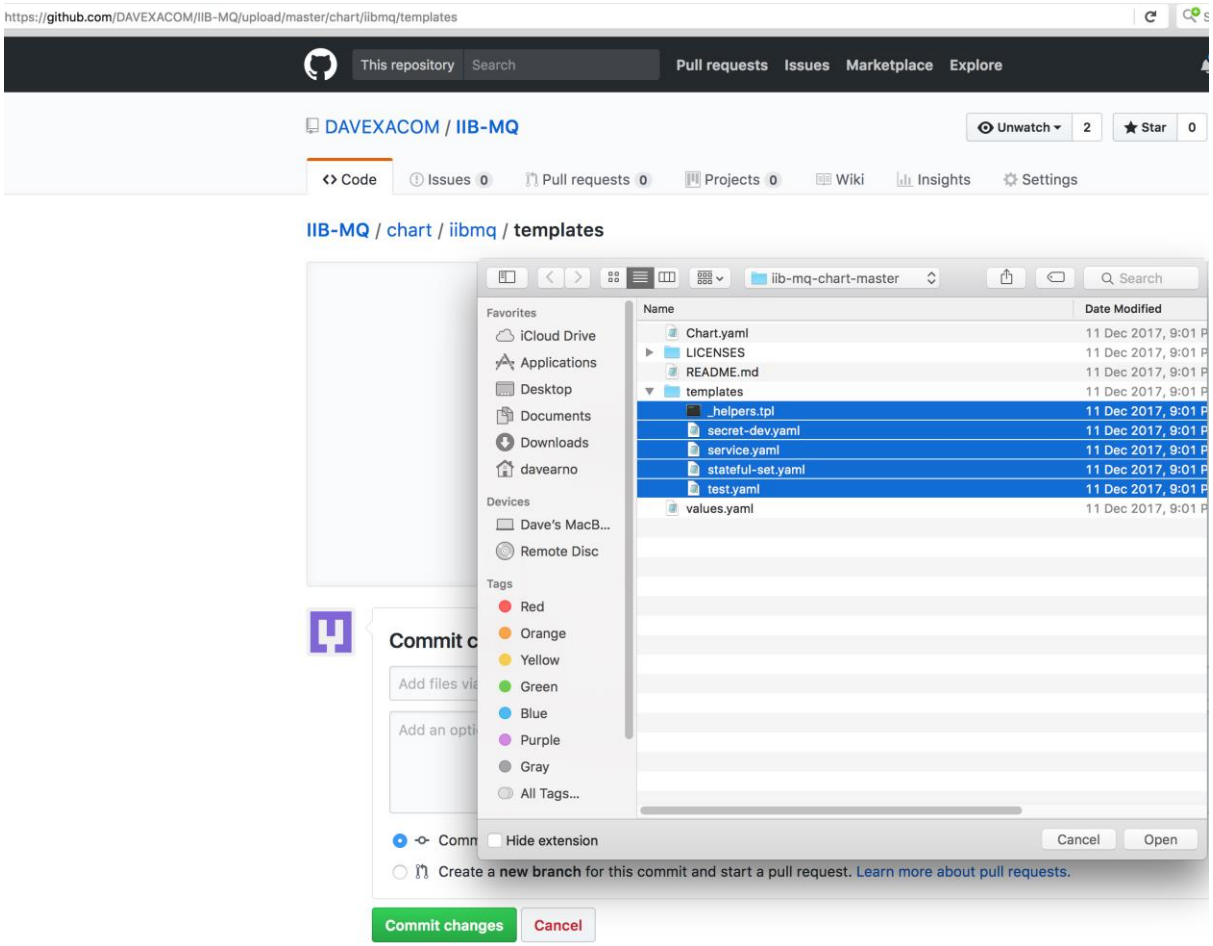
Import to from the iib-mq-chart repository to the local file system

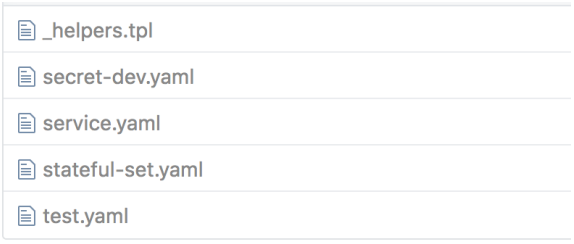


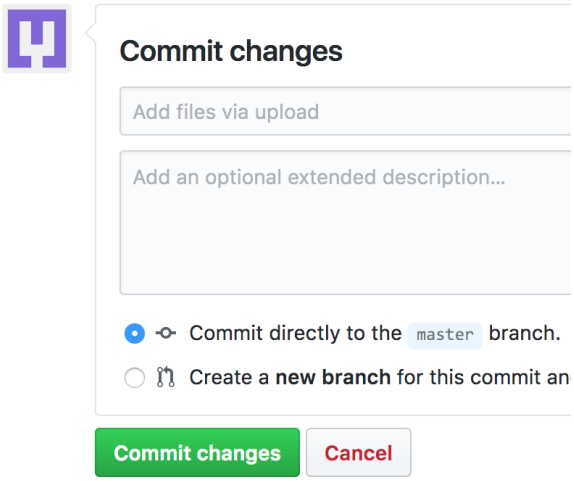


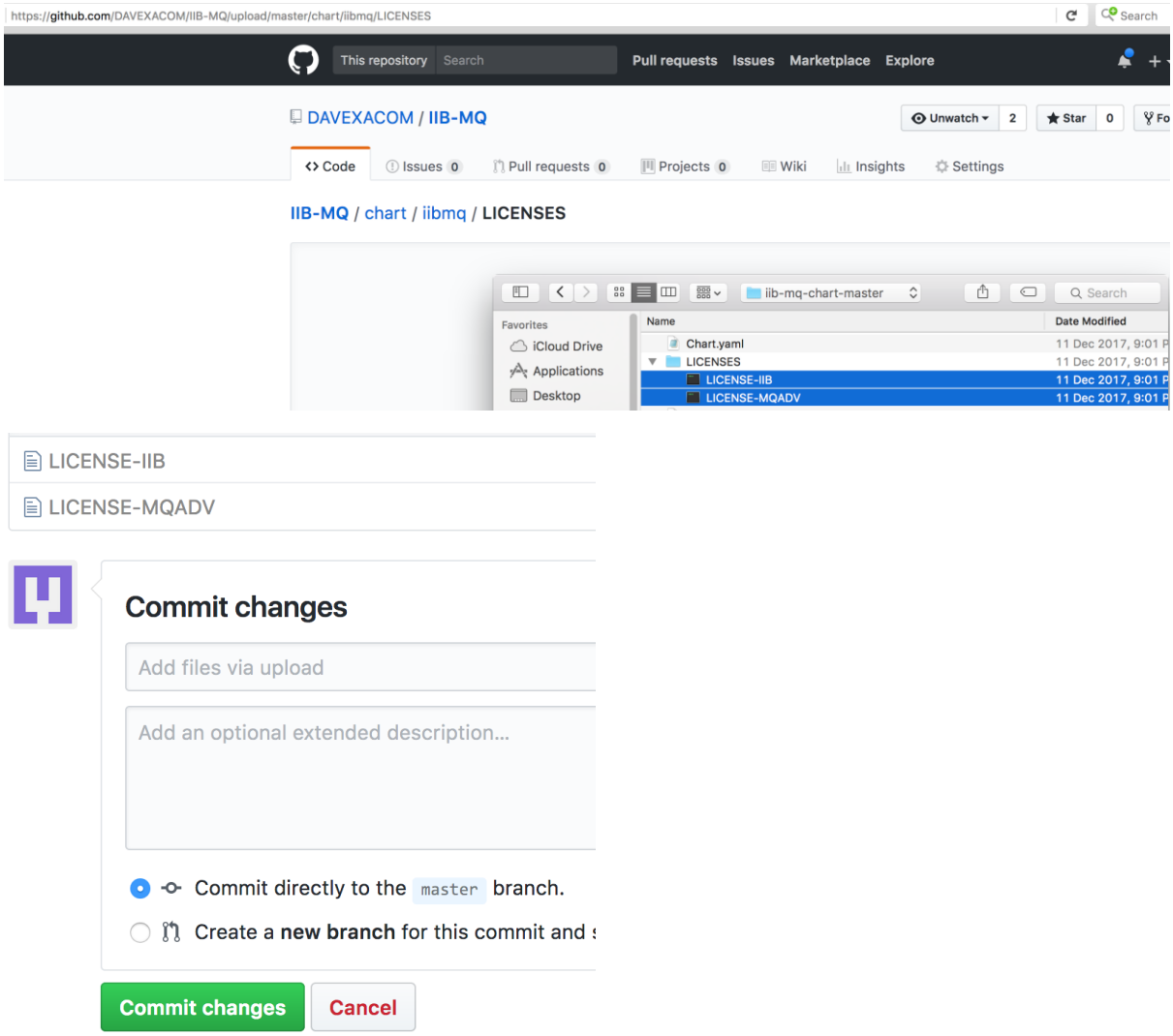
Commit changes




https://github.com/DAVEXACOM/IIB-MQ/upload/master/chart/iibmq/templates
This repository
Search
Pull requests
Issues
Marketplace
Explore
DAVEXACOM / IIB-MQ
Unwatch
2
Star
0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
IIB-MQ / chart / iibmq / templates
iib-mq-chart-master
Search
Favorites
iCloud Drive
Applications
Desktop
Documents
Downloads
davearno
Devices
Dave's MacB...
Remote Disc
Tags
Red
Orange
Yellow
Green
Blue
Purple
Gray
All Tags...
Name
Date Modified
Chart.yaml
LICENSES
README.md
templates
_helpers.tpl
secret-dev.yaml
service.yaml
stateful-set.yaml
test.yaml
values.yaml
11 Dec 2017, 9:01 P
Hide extension
Cancel
Open
Create a new branch for this commit and start a pull request. Learn more about pull requests.
Commit changes
Cancel


_helpers.tpl
secret-dev.yaml
service.yaml
stateful-set.yaml
test.yaml


Commit changes
Add files via upload
Add an optional extended description...
Commit directly to the master branch.
Create a new branch for this commit an
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

This repository Search Pull requests

DAVEXACOM / IIB-MQ

Code Issues 0 Pull requests 0 Projects

Branch: master IIB-MQ / Jenkinsfile

Davexa Create Jenkinsfile

1 contributor

10 lines (8 sloc) 188 Bytes

```groovy
#!groovy

@Library('MicroserviceBuilder') _
microserviceBuilderPipeline {
  image = 'iib10mq9'
  mavenImage = 'wwdemo/images:maven-lab'
  deployBranch = 'master'
  namespace = 'default'
}
```

After :

DAVEXACOM / IIB-MQ

Code Issues 0 Pull requests 0

Branch: master IIB-MQ / Jenkinsfile

Davexa Update Jenkinsfile

1 contributor

11 lines (9 sloc) 218 Bytes

```groovy
#!groovy

@Library('MicroserviceBuilder') _
microserviceBuilderPipeline {
  image = 'iib10mq9'
  mavenImage = 'wwdemo/images:maven-lab'
  chartFolder = 'chart/iibmq'
  deployBranch = 'master'
  namespace = 'default'
}
```