Traditional MQ/IIB

Based ESB

On

IBM Cloud Private

Contents

[Introduction 4](#_Toc704364)

[Overview diagram - A Traditional ESB pattern on IBM Cloud Private 4](#_Toc704365)

[Scenario Description 4](#_Toc704366)

[Supporting Collateral on Github 6](#_Toc704367)

[Helm Release Repository for 4-Way MQ Cluster with Local IIB 6](#_Toc704368)

[Container Build, Load (optional Helm release) IIB and MQ container 6](#_Toc704369)

[Pre-Req work – Customization of Microclimate/Jenkins build scripts on ICP 7](#_Toc704370)

[Documentation 7](#_Toc704371)

[Customization Instruction guide 7](#_Toc704372)

[Create a project in Microclimate and Deploy 9](#_Toc704373)

[Troubleshooting 10](#_Toc704374)

[Setting up the ESB on ICP Demo – Part 1 the MQ and IIB Image 11](#_Toc704375)

[Create Microclimate project to build the MQ with IIB image 11](#_Toc704376)

[Microclimate project/Jenkins pipeline builds the MQ with IIB image 17](#_Toc704377)

[Following the Jenkins build process as it happens 17](#_Toc704378)

[Checking the results of IIB-MQ-SPLIT-HELM build on IBM Cloud Private 23](#_Toc704379)

[Validating our IIB/MQ build and helm release 27](#_Toc704380)

[Deleting and cleaning up our IIB-MQ-SPLIT-HELM release 32](#_Toc704381)

[Setting up the ESB on ICP Demo – Part 2 the MQ Image and ESB Helm Release 36](#_Toc704382)

[Review the MQ4WAYCLUS-LCIIB values.yaml for APPQM1 and APPQM2 in Github 36](#_Toc704383)

[Create Microclimate project for the MQ image build and MQ-4Way Cluster Helm Release 39](#_Toc704384)

[Checking the results of MQ4WAYCLUS-IIBLC build on ICP 47](#_Toc704385)

[Validating the MQ4WAYCLUS-LCIIB release 53](#_Toc704386)

[Testing GWQM1 MQ 54](#_Toc704387)

[Testing GWQM2 MQ 57](#_Toc704388)

[Testing APPQM1 MQ 58](#_Toc704389)

[Testing APPQM2 MQ 59](#_Toc704390)

[Testing APPNODE1 IIB 60](#_Toc704391)

[Testing APPNODE2 IIB 62](#_Toc704392)

[Removing the Helm Release and cleaning up 64](#_Toc704393)

[Changing the version of MQ from 905 to 904 or vice versa for the GWQM1 and GWQM2 containers. 68](#_Toc704394)

[Updating the Github repository – MQ4WAYCLUS-LCIIB 68](#_Toc704395)

[Edit the docker file to change the MQ installation version 68](#_Toc704396)

[Perform a new build and Helm Release 69](#_Toc704397)

[Rerun the Jenkins pipeline 69](#_Toc704398)

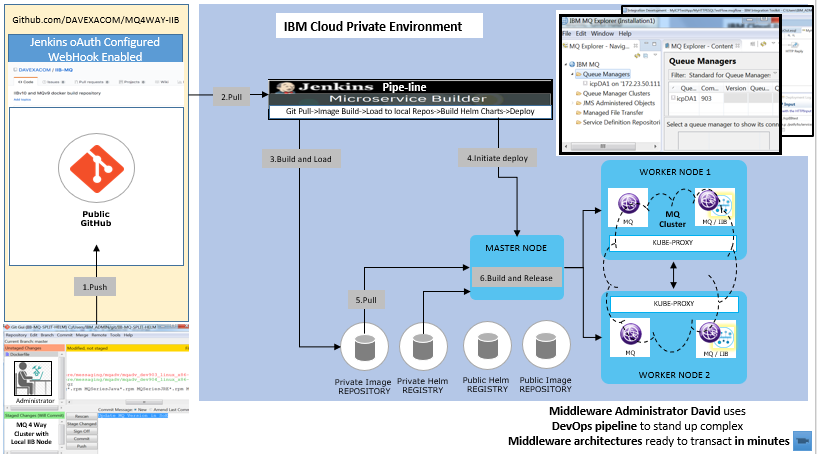
[Check the results of the Helm Release on ICP 70](#_Toc704399)

[Validate the version of MQ on GWQM1 72](#_Toc704400)

# Introduction

This setup and demo guide explores delivering a complex integration and messaging pattern on IBM Cloud Private using Helm, Github and Microclimate/Jenkins DevOps pipelines.

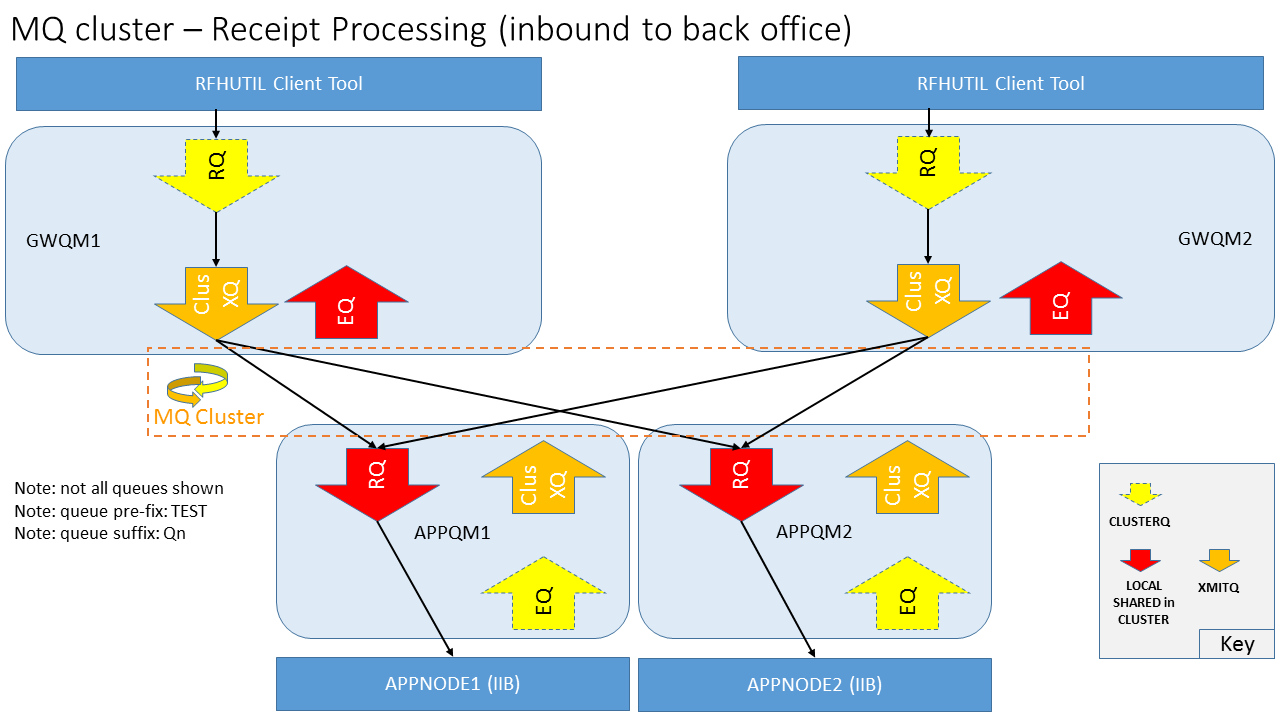
## Overview diagram - A Traditional ESB pattern on IBM Cloud Private

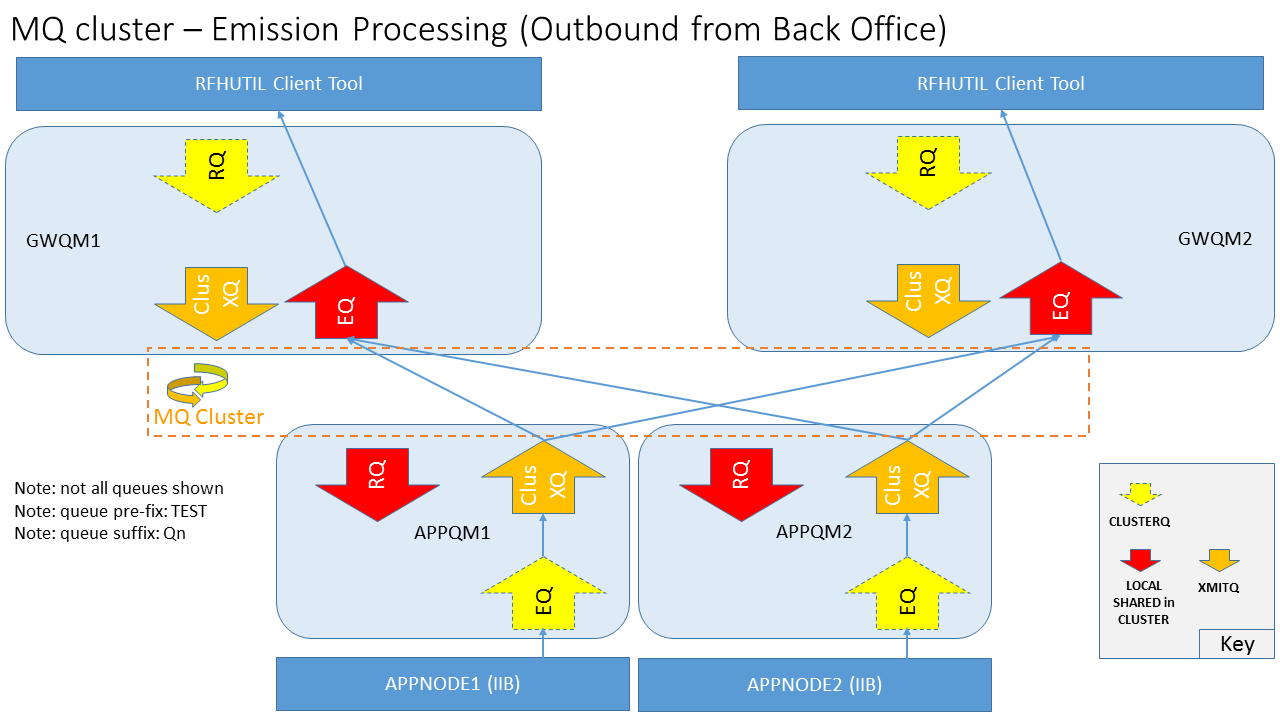


## Scenario Description

The scenario shown above explores using a single Helm Release on IBM Cloud Private to deploy a 4 Way MQ cluster with 2 active/active Gateway queue managers and active/active back office application queue managers each with a locally connected IBM Integration Bus node. The docker images and helm charts are stored on Public Github and the ICP Devops tooling, Micro Climate (Jenkins) is used to build the images and deploy the Helm Release. This 4 container deployment initializes on ICP with all IBM MQ channels resolved and started plus IBM Integration Bus message flows deployed, started and ready to service messages for back office.

The following two diagrams show how the MQ Cluster and IIB nodes process inbound (receipt) and outbound (emission) messages.





# Supporting Collateral on Github

## Helm Release Repository for 4-Way MQ Cluster with Local IIB

<https://github.com/DAVEXACOM/MQ4WAYCLUS-LCIIB>

The MQ4WAYCLUS-LCIIB repository delivers:

* The MQ Docker Image build used by GWQM1 and GWQM2 containers
* Helm charts for the full GWQM1 (MQ only), GWQM2 (MQ Only), APPQM1(MQ with IIB) and APPQM2(MQ with IIB) helm release.

The Helm release relies on two Github repositories because we have two different images to deploy as part of the release. Two containers running the MQ Only Image and two containers running the MQ with IIB image. Therefore both images will need to be build and pushed to the ICP image repository

## Container Build, Load (optional Helm release) IIB and MQ container

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

The IIB-MQ-SPLIT-HELM repository delivers the MQ+IIB Docker Image build, which is used by APPQM1 and APPQM2 in the helm charts in MQ4WAYCLUS-LCIIB.

IIB-MQ-SPLIT-HELM does container helm charts such that this repository can also be used standalone to deliver a single MQ+IIB container build that will be leveraged by the Helm charts in MQ4WAYCLUS-LCIIB. Therefore, we must ensure it is built and available in the ICP image repository before initiating the Helm release MQ4WAYCLUS-LCIIB.

# Pre-Req work – Customization of Microclimate/Jenkins build scripts on ICP

## Documentation

The following documentation for this pre-req work is kept up to date here:

<https://github.com/cloudnativedemo/icp-notes/blob/master/microclimate_notes.md>

## Customization Instruction guide

#### **1. Create pipeline deployment namespace**

kubectl create namespace microclimate-pipeline-deployments

#### **2. Edit ClusterImagePolicy**

kubectl edit clusterimagepolicies ibmcloud-default-cluster-image-policy

To add the following:

- name: mycluster.icp:8500:\*

- name: docker.io/maven:\*

- name: docker.io/lachlanevenson/k8s-helm:\*

- name: docker.io/jenkins/\*

#### **3. Create Docker Registry secret to microclimate namespace**

kubectl create secret docker-registry microclimate-registry-secret \

--docker-server=mycluster.icp:8500 \

--docker-username=admin \

--docker-password=admin \

--docker-email=null

#### **4. Initialise Helm and login**

helm init --client-only --skip-refresh

cloudctl login -a https://mycluster.icp:8443 -u admin -p admin -c id-mycluster-account -n default --skip-ssl-validation

#### **5. Create Helm secret**

export HELM\_HOME=$HOME/.helm

kubectl create secret generic microclimate-helm-secret --from-file=cert.pem=$HELM\_HOME/cert.pem --from-file=ca.pem=$HELM\_HOME/ca.pem --from-file=key.pem=$HELM\_HOME/key.pem

#### **6. Create Docker Registry secret for microclimate-pipeline-deployments namespace**

kubectl create secret docker-registry microclimate-pipeline-secret \

--docker-server=mycluster.icp:8500 \

--docker-username=admin \

--docker-password=admin \

--docker-email=null \

--namespace=microclimate-pipeline-deployments

#### **7. Update ImagePullSecret for microclimate-pipeline-deployments namespace**

kubectl patch serviceaccount default --namespace microclimate-pipeline-deployments -p '{"imagePullSecrets": [{"name": "microclimate-pipeline-secret"}]}'

#### **8. Customise Jenkins library**

By default, the Jenkins library parameter is pointing to <https://github.com/microclimate-dev2ops/jenkins-library> This Jenkins library was a part of the Microclimate DevOps process. When a pipeline is created within a project in Microclimate, microclimate will create a Jenkins pipeline. The pipeline uses this library to .. 1. Pull the code from github repo . .. 2. Build a Docker image based on a Dockerfile found in the repo . .. 3. Authenticate and push the image into ICP's private registry . .. 4. Notify Microclimate to move to the next stage (e.g. deploy) . .. 5. Microclimate 'helm deploy' the helm chart found in the repo (by default it's under the /chart directory) .

Unfortunately, Microclimate only deploy it's supported project types e.g. Swift, NodeJS, Java/Liberty or Springboot. The easiest way to address this limitation is to fork and update the Jenkins library and inject the 'helm deploy' scriptlet onto step 4 (line 400 of microserviceBuilderPipeline.groovy)

container ('helm') {

echo "Attempting to deploy the test release"

def deployCommand = "helm install ${realChartFolder} --values pipeline.yaml --namespace ${namespace} --name ${helmRelease}"

if (fileExists("chart/overrides.yaml")) {

deployCommand += " --values chart/overrides.yaml"

}

if (helmSecret) {

echo "Adding --tls to your deploy command"

deployCommand += helmTlsOptions

}

testDeployAttempt = sh(script: "${deployCommand} > deploy\_attempt.txt", returnStatus: true)

if (testDeployAttempt != 0) {

echo "Warning, did not deploy the test release into the test namespace successfully, error code is: ${testDeployAttempt}"

echo "This build will be marked as a failure: halting after the deletion of the test namespace."

}

printFromFile("deploy\_attempt.txt")

}

* ***Note:*** in my deployCommand, I've created one new variable ${helmRelease}. The variable is defined on the top of the script (line 56 of the microserviceBuilderPipeline.groovy). Alternatively, you can just reuse ${image} as your helm release name .

def helmRelease = (config.releaseName ?: config.image ?: "").trim()

* My forked updated Jenkins library repo can be found [here](https://github.com/cloudnativedemo/jenkins-library) .

#### **9. Deploy Microclimate helm chart**

**Via Helm command line**

* **Add ibm-charts Helm repo**

helm repo add ibm-charts https://raw.githubusercontent.com/IBM/charts/master/repo/stable/

* **Deploy microclimate Helm chart**

helm install --name microclimate --namespace <target namespace> --set global.rbac.serviceAccountName=micro-sa,jenkins.rbac.serviceAccountName=pipeline-sa,global.ingressDomain=172.23.52.247.nip.io,jenkins.Pipeline.Template.RepositoryUrl=https://github.com/cloudnativedemo/jenkins-library.git,jenkins.Pipeline.Template.Version=master ibm-charts/ibm-microclimate --tls

***Note:*** Replace <172.23.52.247> with your <PROXY\_IP>

#### **Via ICP catalog**

* Select ibm-microclimate from ICP catalog > click Configure
* Provide values for the following parameters:
  + Helm release name: your-microclimate-release-name
  + Namespace: default (or your preferred namespace)
  + Microclimate hostname: microclimate.172.23.52.247.nip.io (replace with your <microclimate.PROXY\_IP.nip.io> or your own hostname)
  + Ensure that you've already created Persistent Volumes for Microclimate and Jenkins
  + Service account name for Portal: micro-sa
  + Jenkins library repository: <https://github.com/cloudnativedemo/jenkins-library.git>
  + Jenkins hostname: jenkins.172.23.52.247.nip.io (replace with your <jenkins.PROXY\_IP.nip.io> or your own hostname)
  + Service account name: pipeline-sa
  + Click deploy

### Create a project in Microclimate and Deploy

Once the Microclimate helm deployment completed, you can start to deploy your custom project

1. Make sure that your project contains a Dockerfile, Jenkinsfile and a chart directory (for helm chart)
2. Launch Microclimate (<https://microclimate.172.23.52.247.nip.io> - replace with your own microclimate hostname) and accept licensing agreement (for first launch only)
3. Select Projects > Click New Project
4. Select Java project type and provide a project name > click Next
5. Select Microprofile/J2EE and keep default value for Context root > click Create
6. Once the project is created, select Pipeline on the left menu
7. Click Create pipeline, and provide name and github repo of the pipeline > click Create pipeline to create a Jenkins pipeline
8. Switch to Jenkins (<https://jenkins.172.23.52.247.nip.io> - replace with your Jenkins hostname) to see if the pipeline has been created and built (refer to the troubleshooting section below if you have to wait for too long)

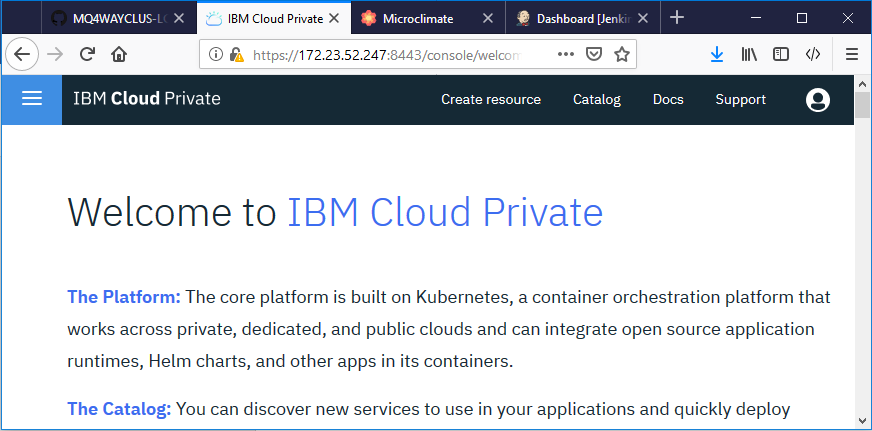
### Troubleshooting

* When your Jenkins pipeline keeps looking for an executor for too long, there's probably an error occurred within your Jenkins containers. To identify the issue:
  + Identify the Jenkins pod name: kubectl get pods -n <NAME\_SPACE\_WHERE\_JENKINS\_INSTALLED> | grep jenkins
  + View the log: kubectl log -n <NAME\_SPACE\_WHERE\_JENKINS\_INSTALLED> <JENKINS\_POD\_NAME> -f
  + Most of the case I found caused by cluster image policy is not defined, you might need to update the default clusterimagepolicy
    - kubectl edit clusterimagepolicies ibmcloud-default-cluster-image-policy

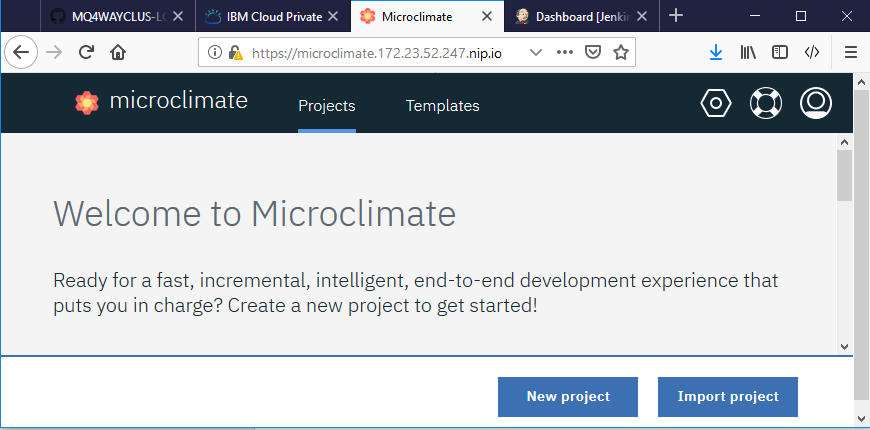
# Setting up the ESB on ICP Demo – Part 1 the MQ and IIB Image

## Create Microclimate project to build the MQ with IIB image

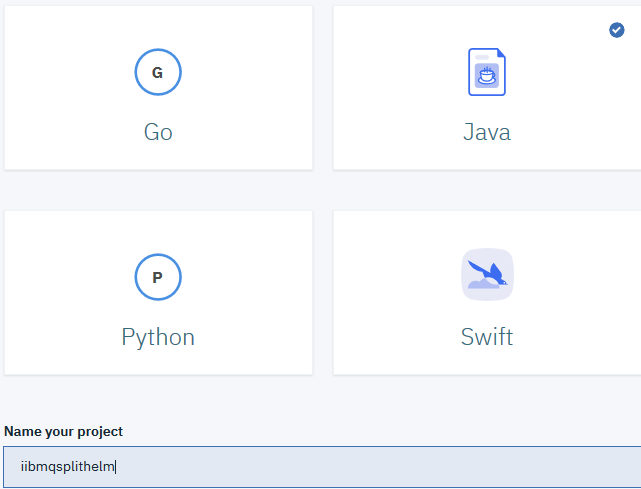
Log into IBM Cloud Private



Connect to the Microclimate service https://microclimate.ICPipAddress.nip.io



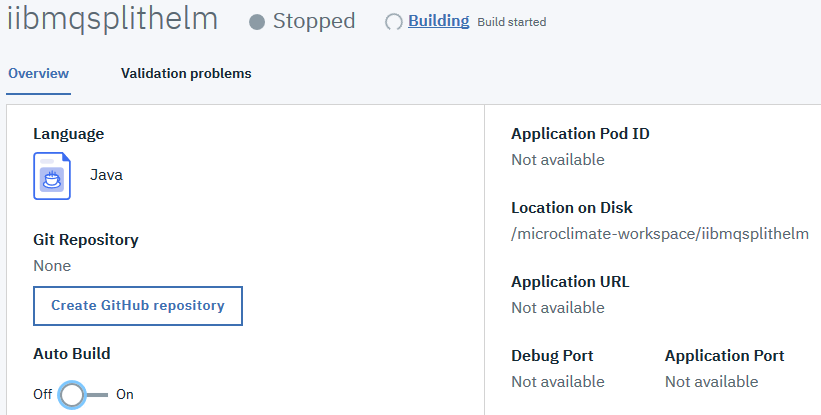
Create a new Java project and name it to represent the github repository you wish to build from



Use the MicroProfile / Java EE (IIB/MQ will work fine with these settings) and hit create.

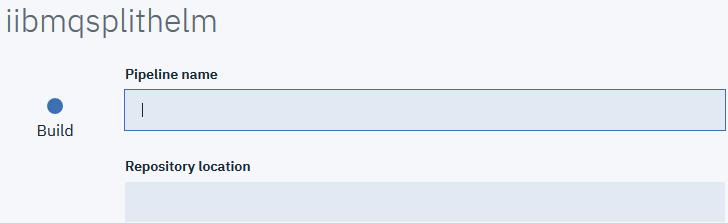
Turn Auto Build to OFF. We will primarily interact with the Jenkins pipeline directly.



Select the Pipeline icon and then select Create pipeline

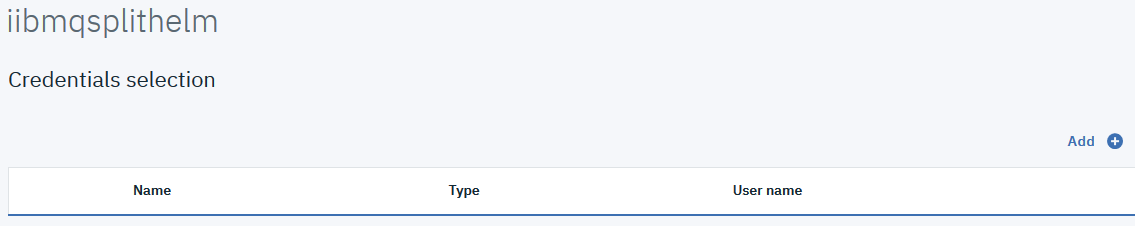
 

Leave the pipeline name and repos location blank for now and click on Select credentials



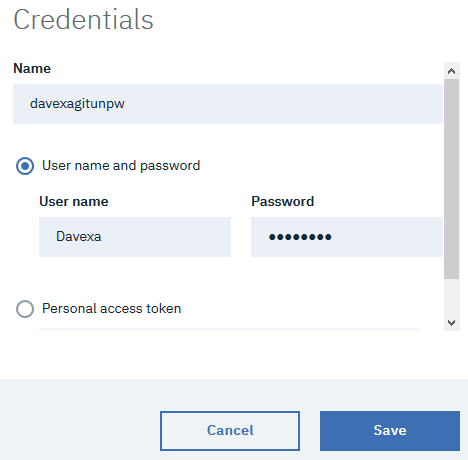


If this is your first project you will need to hit Add and add credentials

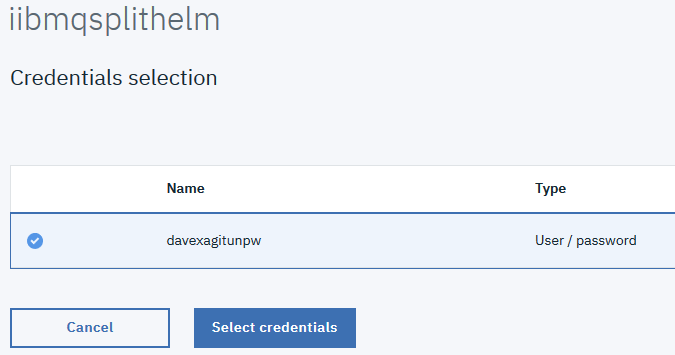




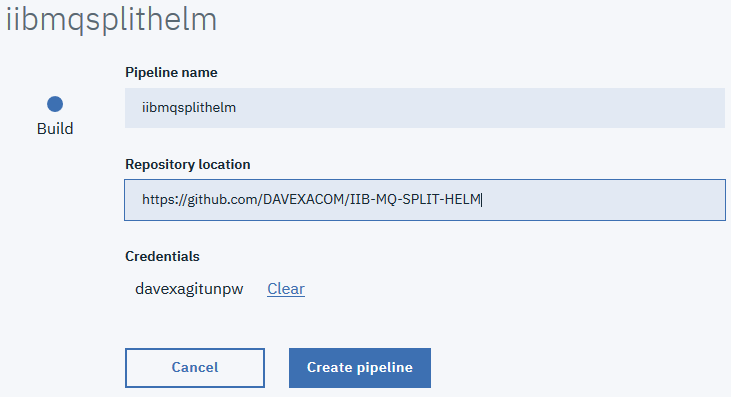
Enter your Github User name and password or leverage a Personal Access Token from github



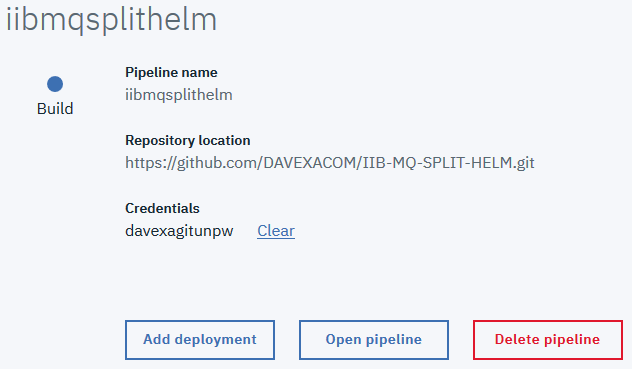
Save the credentials. You can then click on them and hit select.



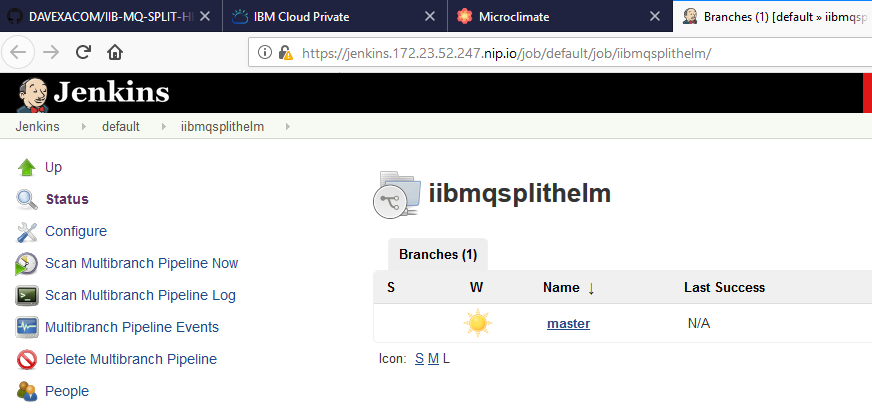
Next add a pipeline name and repos location



Hit Create pipeline



Hit open pipeline and Jenkins will be launched and opened in a new browser window

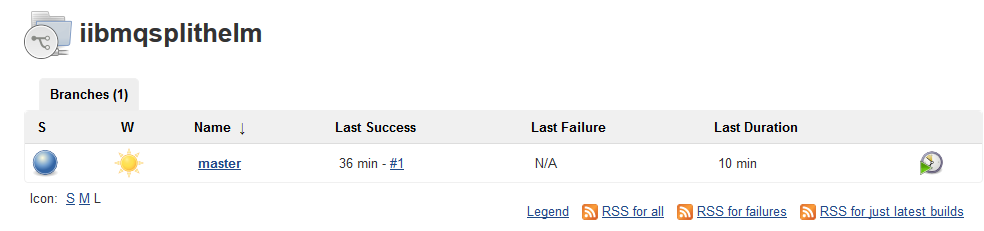


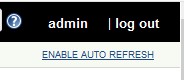
The Jenkins pipeline is created and the first build is initiated

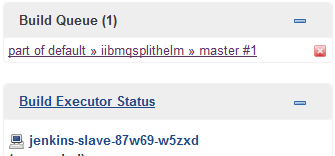
## Microclimate project/Jenkins pipeline builds the MQ with IIB image

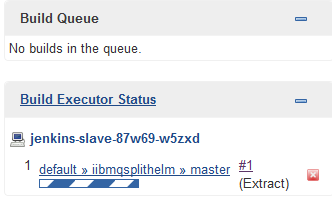
### Following the Jenkins build process as it happens

The following screen shot is a capture of the Jenkins pipeline after the first initiated build is finished. This screen shot is for your reference only. If you want to start subsequent builds after the initial build is finished. For example you make changes to files in the github repository you can start a new build by hitting the circle with the green triangle on it. **You do not need to hit this button at this point**

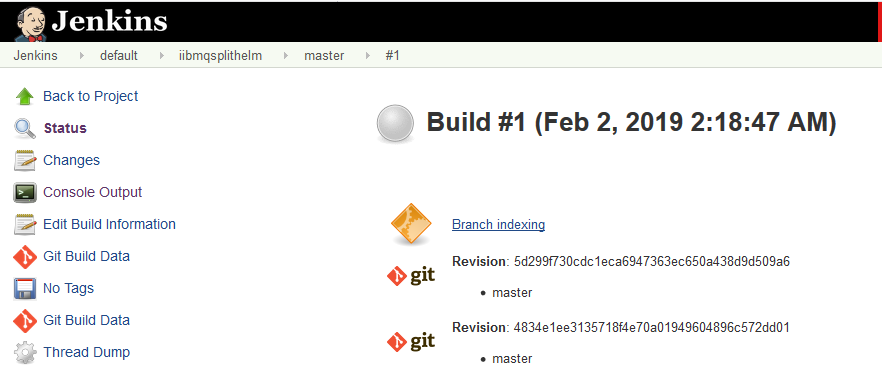




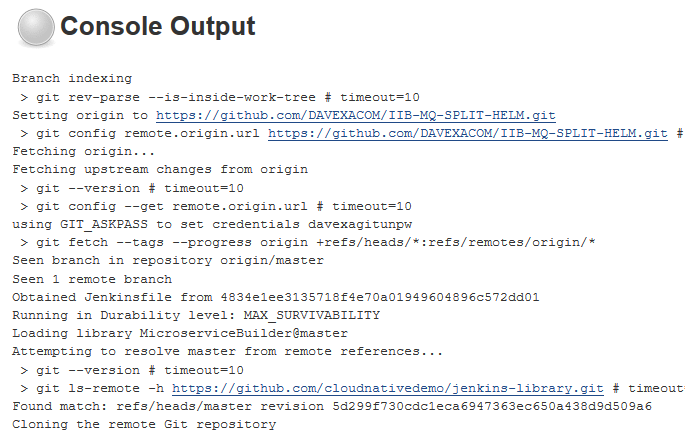


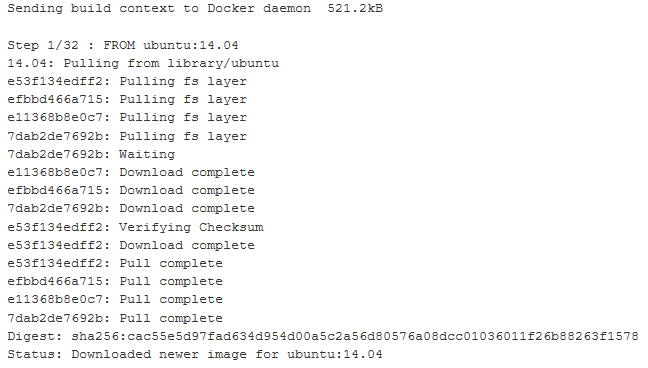


Click on the master #1 link

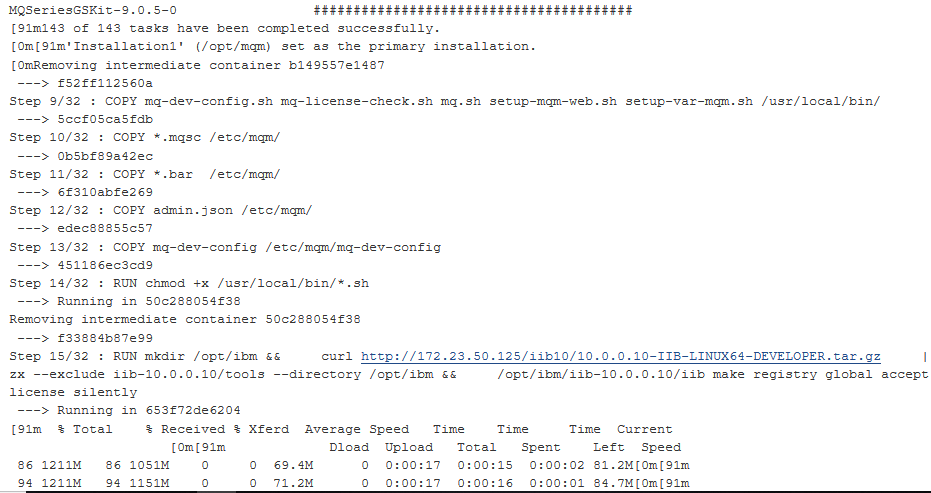


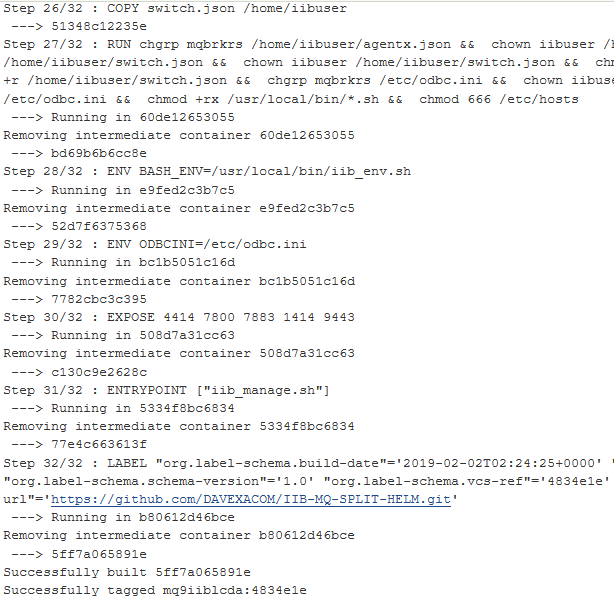
Click on Console Output

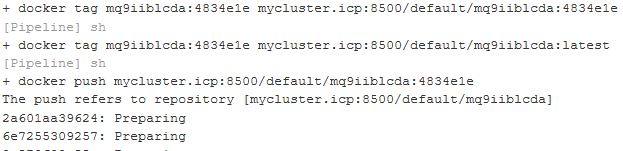


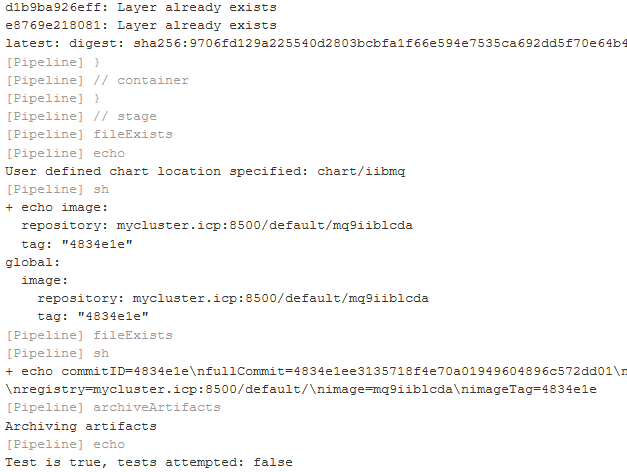




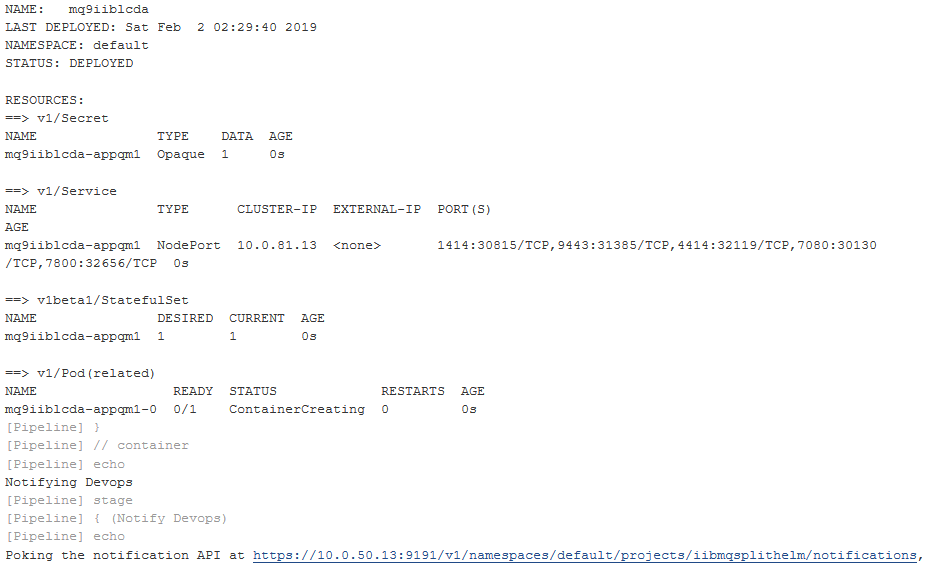


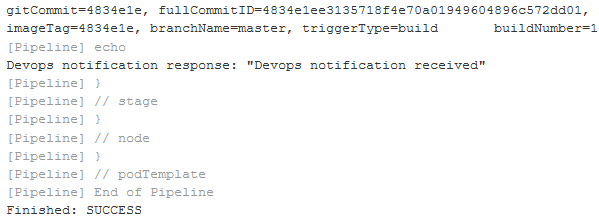






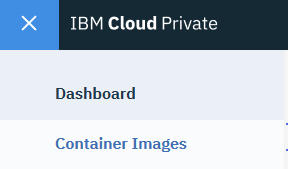




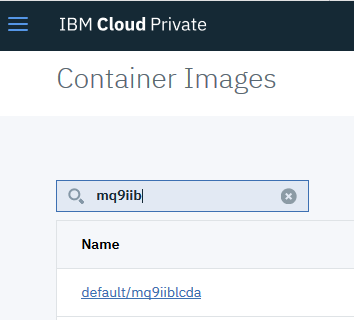


### Checking the results of IIB-MQ-SPLIT-HELM build on IBM Cloud Private

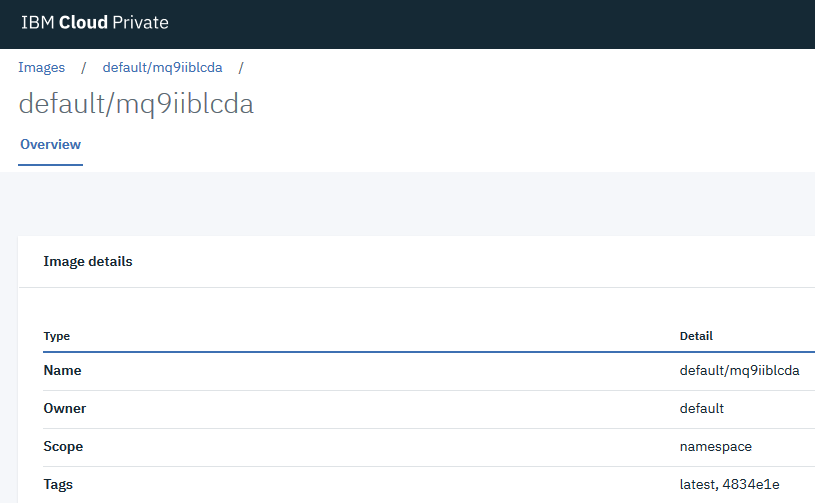
From the IBM Cloud Private console select the container images



Filter on mq9iib



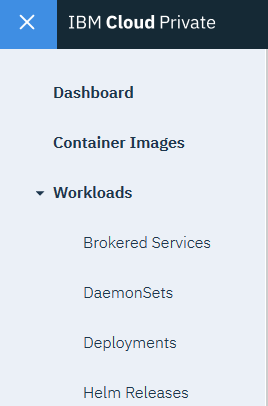
Click on the link to check the image details



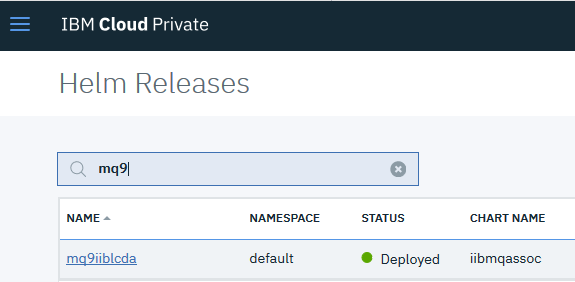
Make of note of the tag 4834e1e (in this example). We might need this in the helm charts for other repository build as “latest” does not always seem to work but we’ll try it 😊

Next let’s review the Helm releases.

Note: In the scope of this full demo we don’t really need a release for the IIB/MQ Image, we just need the image on the ICP. That said, there is value in checking that the Image delivers a runtime container that operates as expected.



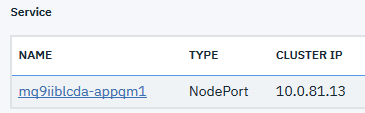
Select Helm Releases and use the Search filter to find mq9iiblcda



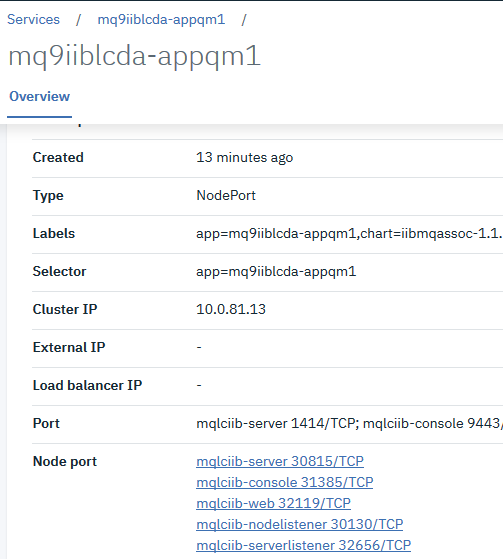
Click on mp9iiblcda link and explore the release.

Note this is a single container running a combined IIBv10 runtime with a local MQ Queue Manager. Exploring and testing this release allows to prove we have a working container build to be leveraged by MQ4WAYCLUSIIBLC (the MQ 4-Way cluster with IIB local binding) Helm release in the next part of the demo set up.

Navigate to the Service and click on the link



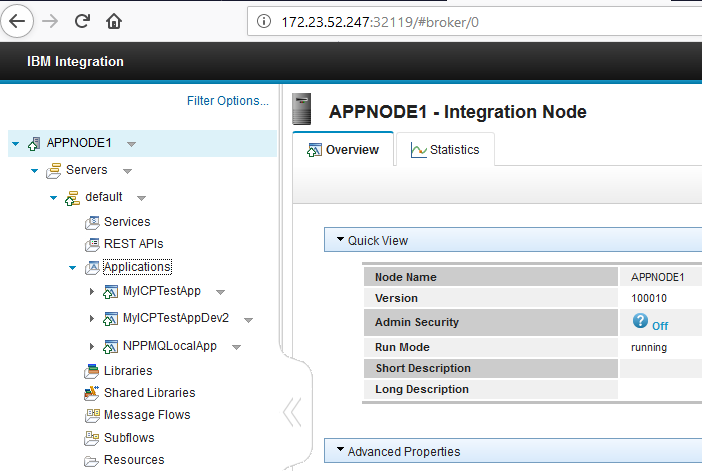
Here you’ll find the links to for the port nodes (connection) links for the MQ and IIB services



### Validating our IIB/MQ build and helm release

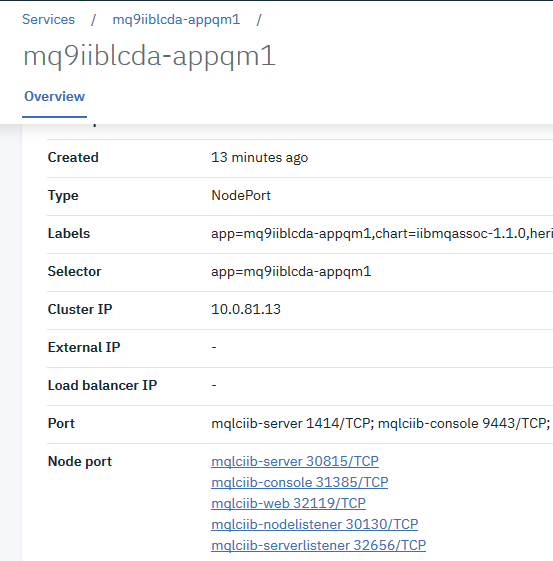
Let’s check the IIB node first

Click on the mqlciib-web nnnnn/TCP link



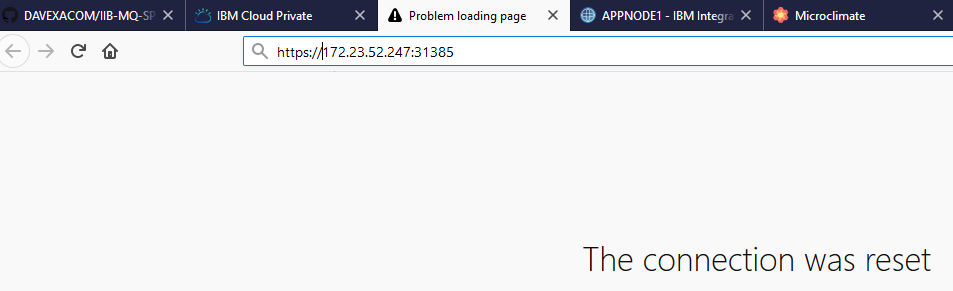
Expand the APPNODE1 and integration server and you see a set of deployed applications.

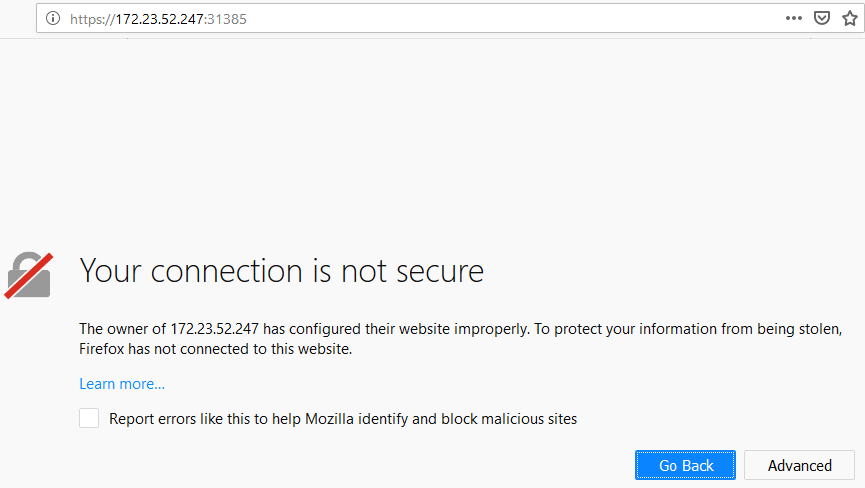
Next Let’s validate MQ. Back on the Helm Release services ICP Console window



Click on the mqlciib-console nnnnn/TCP link

This will fail. You need to add https:// to the URL and hit enter again

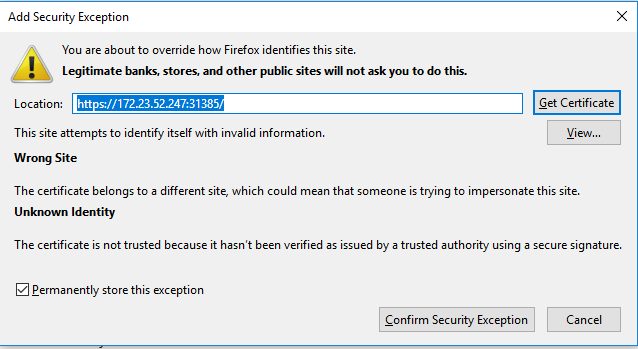




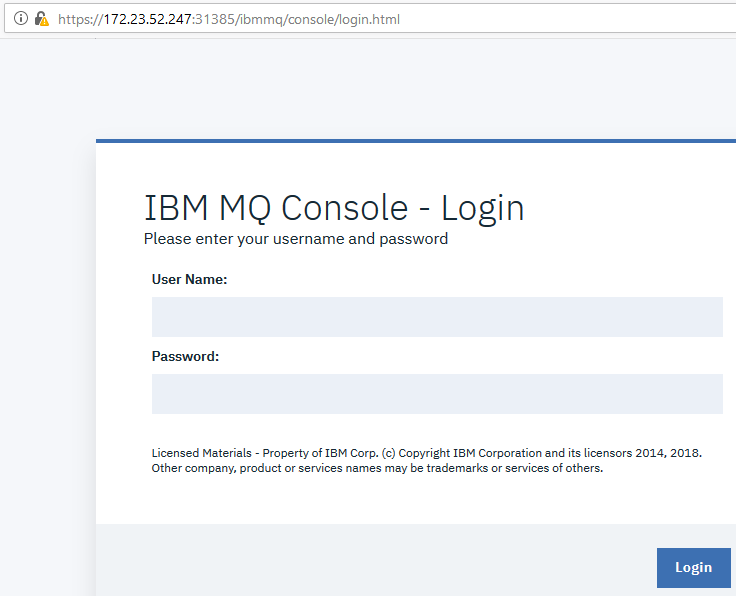
Then hit Advanced button and then Add Exception button



At the next dialog hit Confirm Security Exception



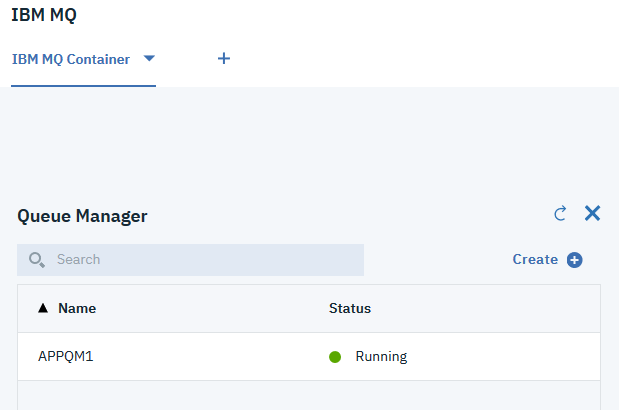
Log into the MQ Console



User Name: admin

Password: passw0rd (note that’s passw0rd with a zero)

The MQ Queue Manager APPQM1 is displayed

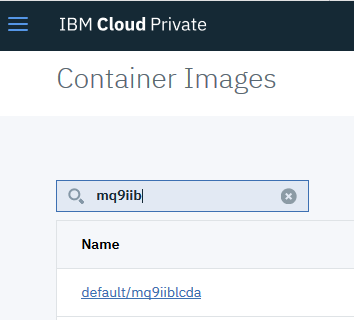


Looks like our IIB node and our MQ Queue manager are in good shape.

I’d say that proves the IIB-MQ-SPLIT-HELM image build, load to ICP repository and resulting runtime container with IIB and MQ works fine.

Next we’ll remove the Helm Release and clean up. Remember we only need the image

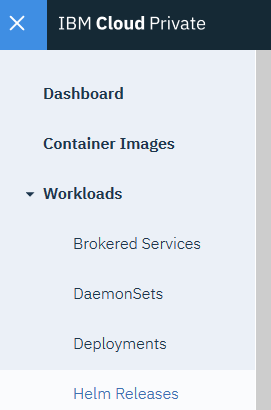
## Deleting and cleaning up our IIB-MQ-SPLIT-HELM release



We don’t need this release as part of our ESB. The MQ4WAYCLUS-LCIIB Helm Release will deliver a full 4 container release leveraging this image for the two “back office nodes” in the ESB pattern.

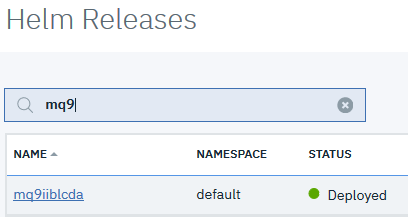
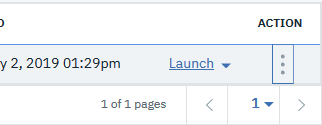
First close the browser tabs for the IIB WebUI and the MQ Console.

Then select Workloads->Helm Releases



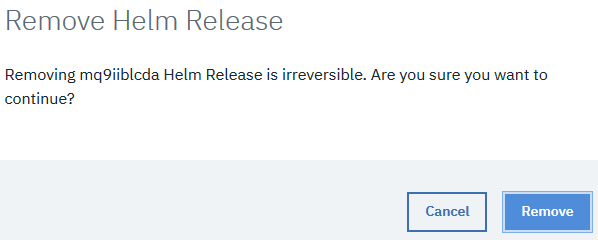
In the Helm releases filter for mq9

For the mq9iiblcda, on the right hand side click the action button (3 vertical dots)

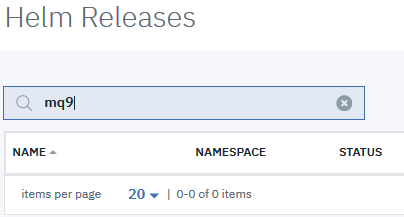
 

And select DELETE

Then at the pop up window select REMOVE

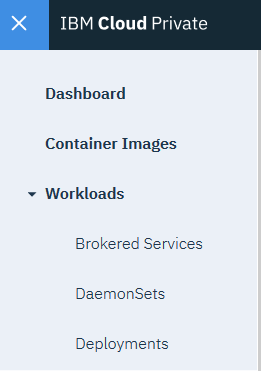


If you get an error, try again and refresh the screen. Sometimes there is a timeout waiting for the response for the remove. A refresh should show it has actually worked

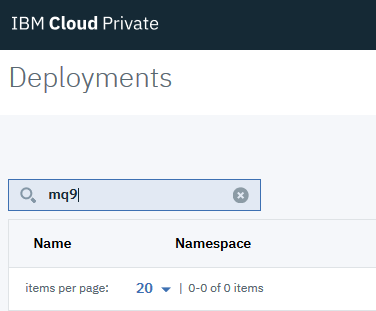


Double check the removal was successful by checking in the Deployments.

Select Workloads->Deployments



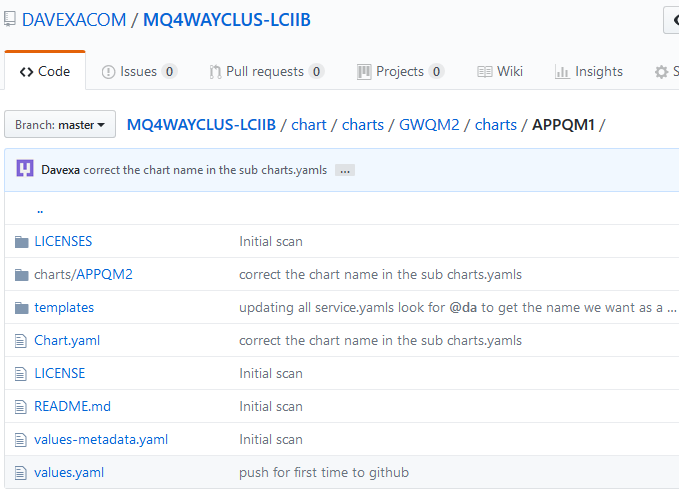
Filter on mq9



Refresh the browser window until the no entries are shown.

# Setting up the ESB on ICP Demo – Part 2 the MQ Image and ESB Helm Release

## Review the MQ4WAYCLUS-LCIIB values.yaml for APPQM1 and APPQM2 in Github



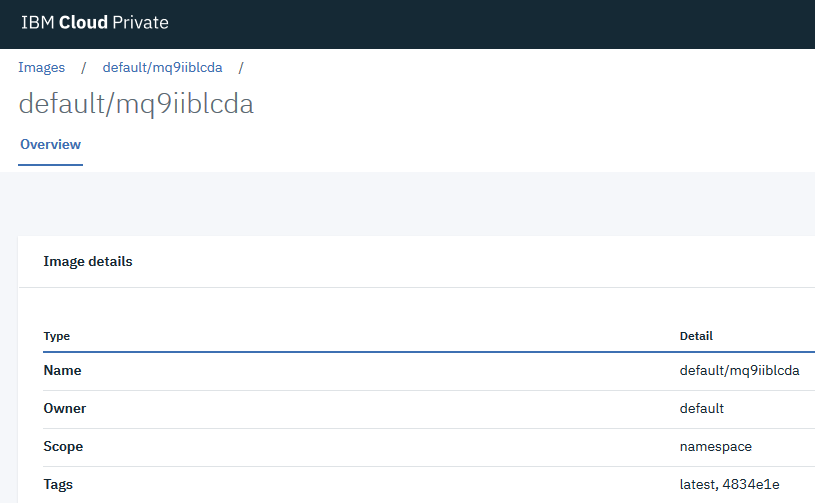
Select values.yaml.

See below the **TAG name** for the image **mqiiblcda** in the previous part.

In part 1 we built a new image and loaded it onto ICP therefore we have new tags.



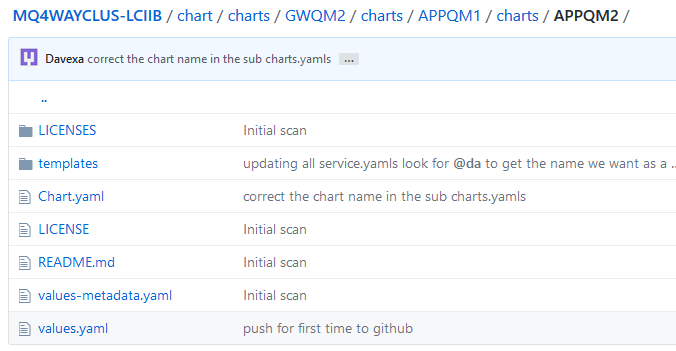
As you can see we have a mis-match compared to the tags for our image on ICP.



Lets edit the values.yaml and correct the tag name. Use the Edit button on githib

Make the change to the tag name and hit the commit button 

Now you must repeat the process for the values.yaml for APPQM2

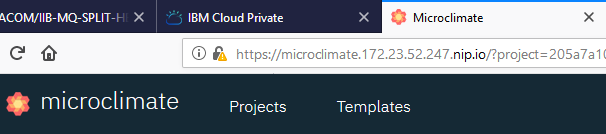


The github repository DAVEXACOM/MQ4WAYCLUS-LCIIC is now set up to

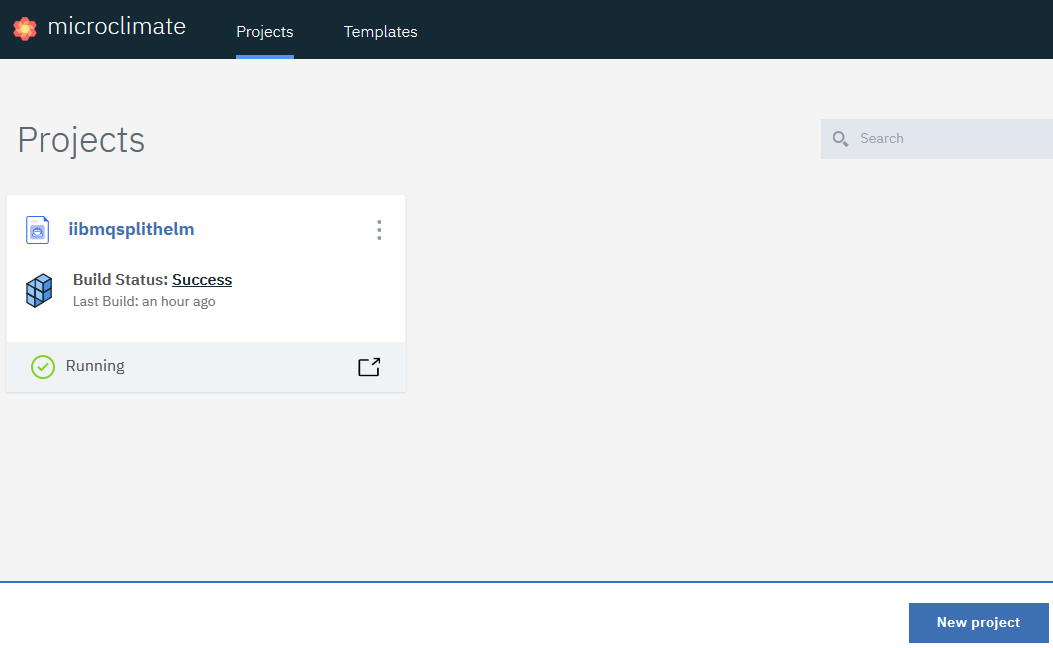
* Build and MQ Only image and load it onto the ICP image repository
* Perform a helm release delivering 4 containers
  + Two containers using the MQ Only image (built from this repos)
  + Two containers using the MQ with IIB image on ICP that we built in part 1

## Create Microclimate project for the MQ image build and MQ-4Way Cluster Helm Release

Got to the Microclimate browser tab

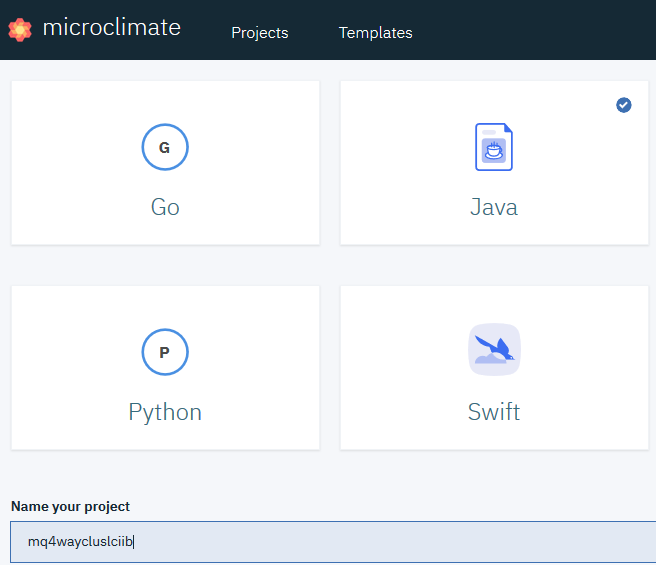


Click on Projects

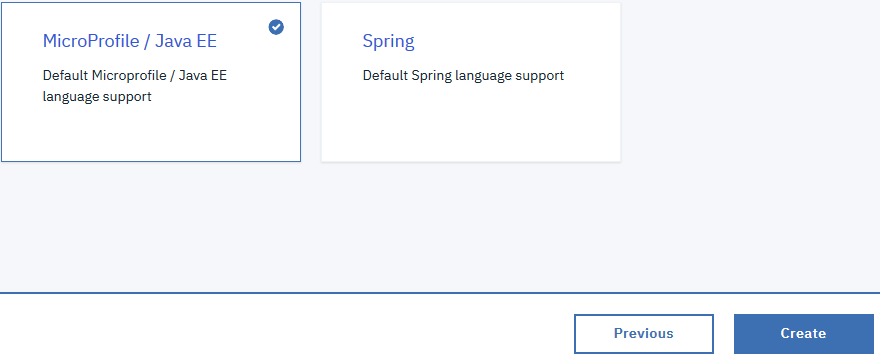


Select New Project

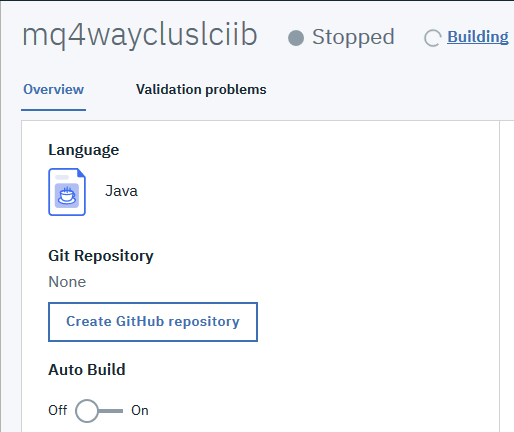
Select Java and enter the name for the project using a name that identifies the mq4wayclus-lciib repository



Click next and select MicroProfile / Java EE

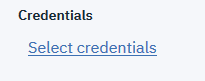


Click Create

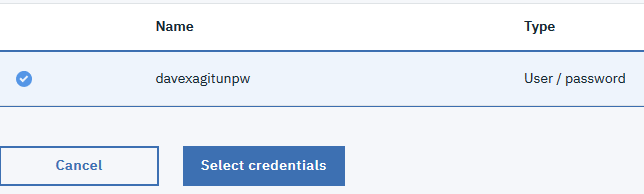


click on the pipeline icon on the left hand side 

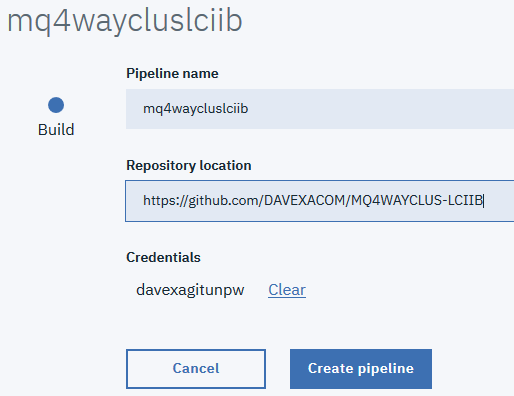
Click on create pipeline button 

click on select credentials 

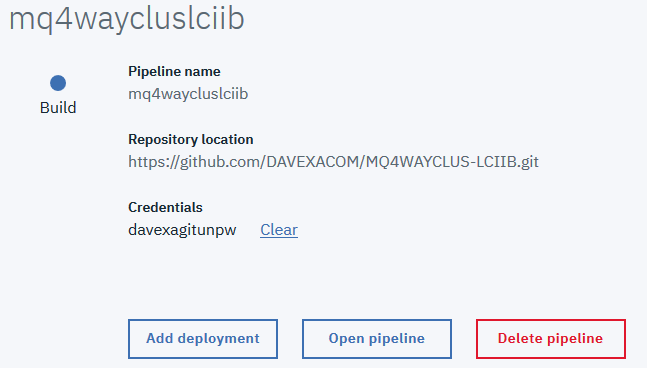
select the credentials you set up in Part 1. In this exampe davexagitunpw user/password

Hit Select credentials

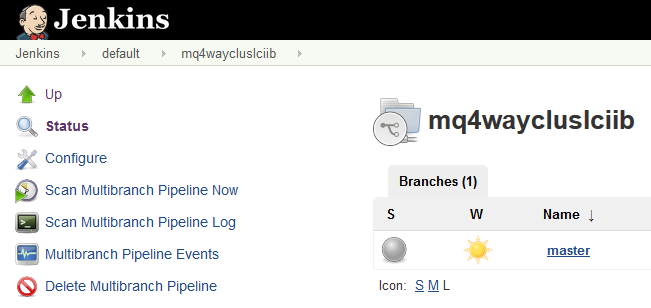
Fill in the pipeline name and repos location and select create the pipeline



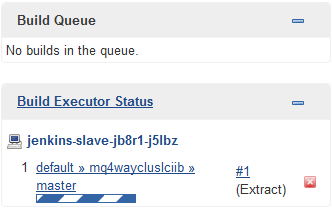
Open the pipeline by clicking on open pipeline



This will launch a new Jenkins client browser window



Page down and you will see a build is initiated (occasionally you need to wait for an available slave)

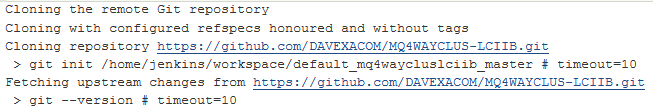


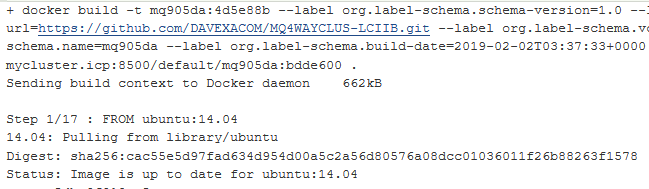
Click on the #1 to follow the initial build and deploy

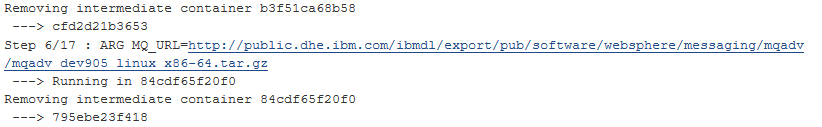


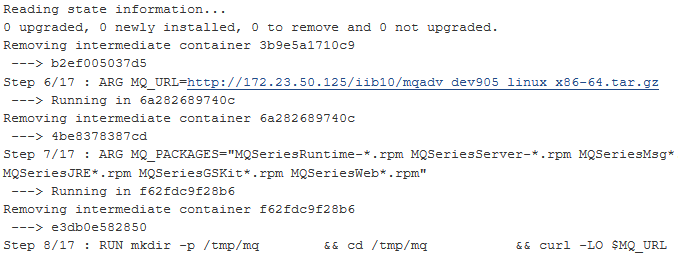
Click on console Output and review the build. I have captured portions of a successful build and release log for your reference.

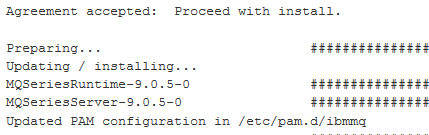


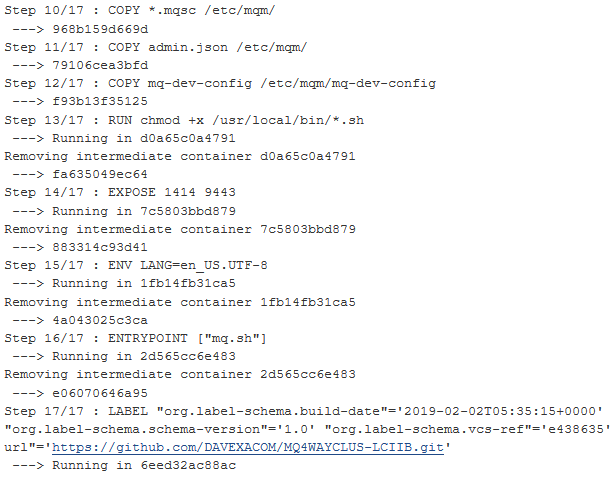


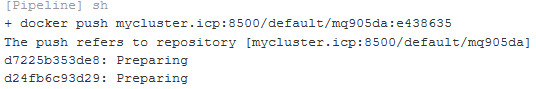


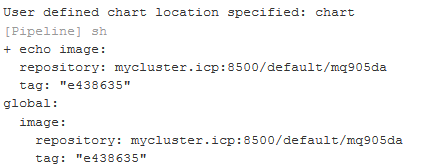


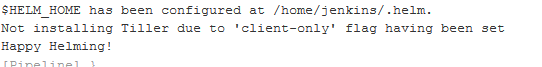




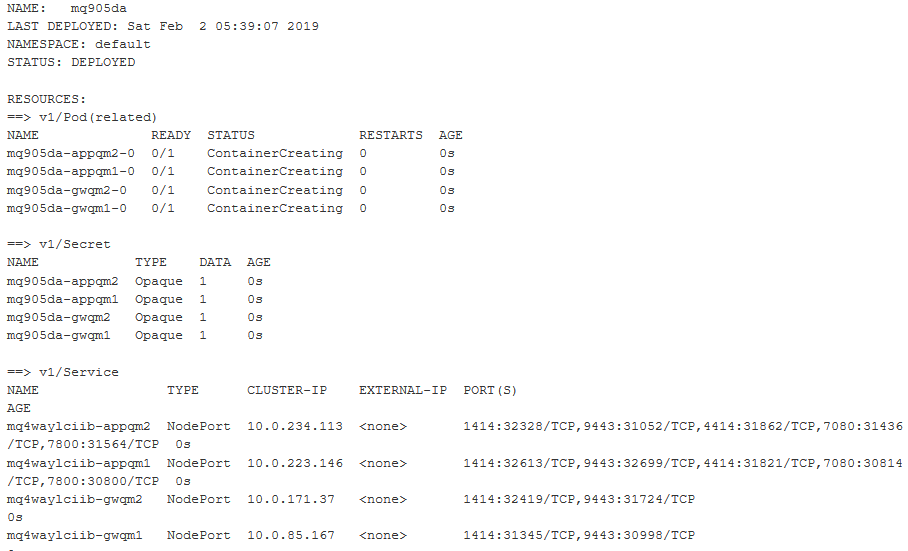








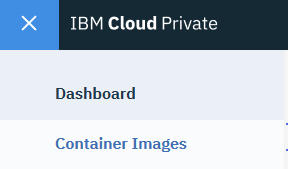




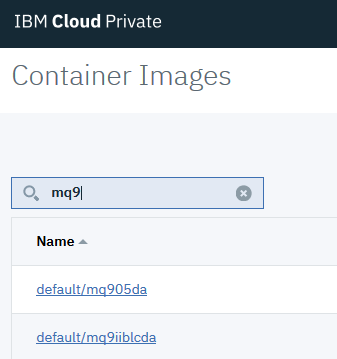


## Checking the results of MQ4WAYCLUS-IIBLC build on ICP

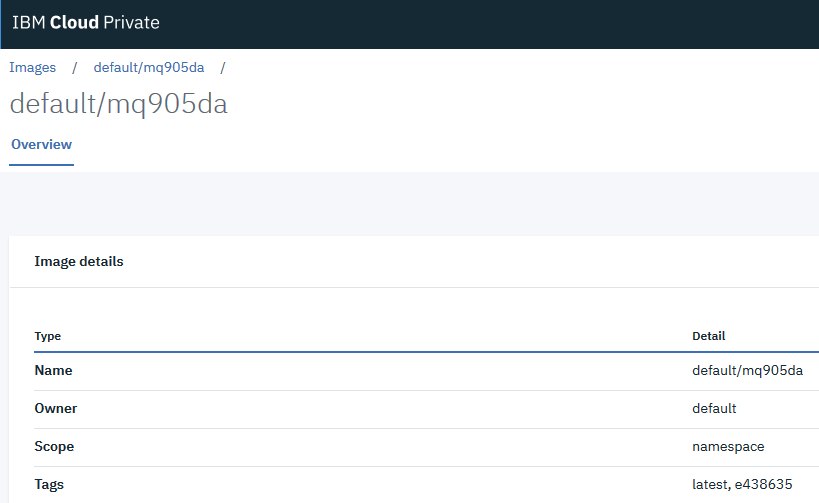
From the ICP Console select Container Images



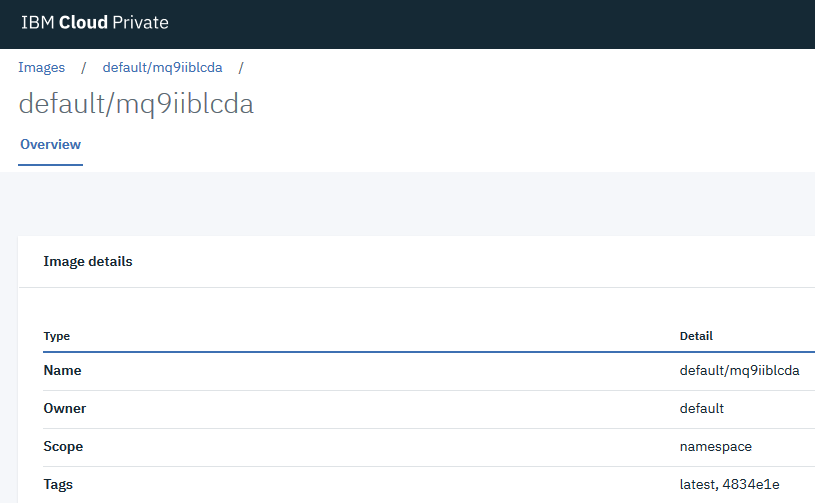
Filter on mq9



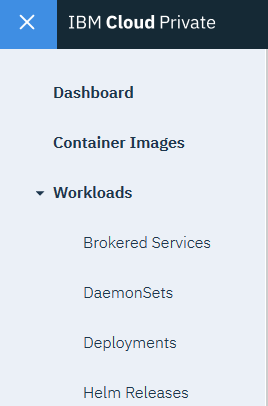
Click on the link for default/mq905da and check out the image details



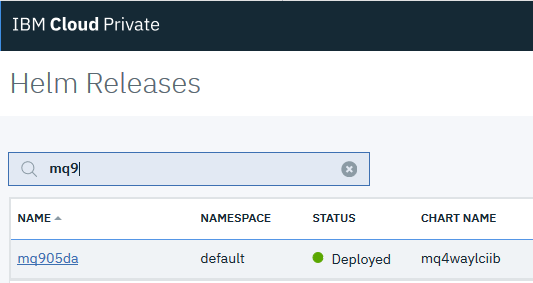
Click on the link for mq9iiblcda (image build and loaded in the first part of the document)



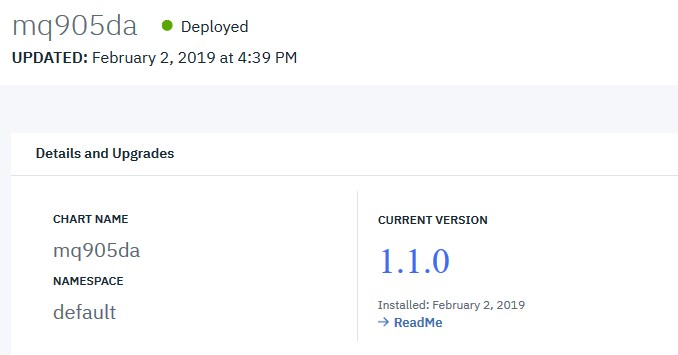
Now lets take a look at the Helm Release. From IBM Cloud Private select Workloads-> Helm releases



Select Helm Releases and use the Search filter with mq9 to find mq905da



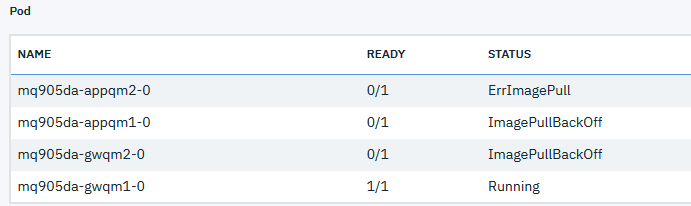
Click on mp905da link and explore the release.



Scroll down and you’ll see the 4 pods each housing a container running GWQM1,GWQM2, APPQM1 and APPQM2

Now for this first build and deploy of the Helm release we may see all but the GWQM1 pod fail. The reason for this is the tagging of the images.

You may see the following



This could be for a number of reasons. Things to check:

The high level chart files in the repository are for GWQM1 this generally works because it has the container build associated with it and Jenkins gets involved and overrides the repository and the tag values rather than relying on those in the charts.

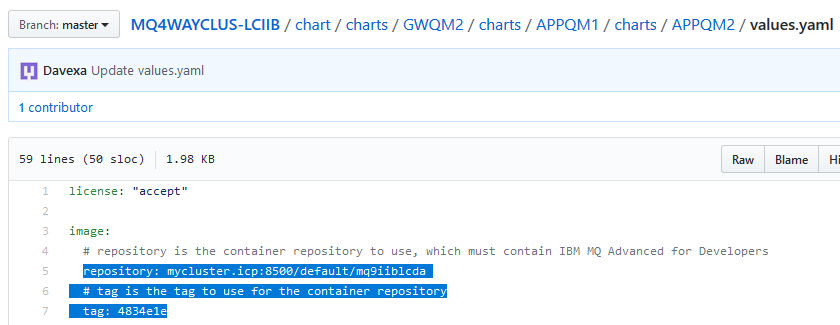
The others, GWQM2, APPQM1 and APPQM2 don’t have this luxury so we need to ensure that the values.yaml for each really have the tag names and repository URLs that are correct.

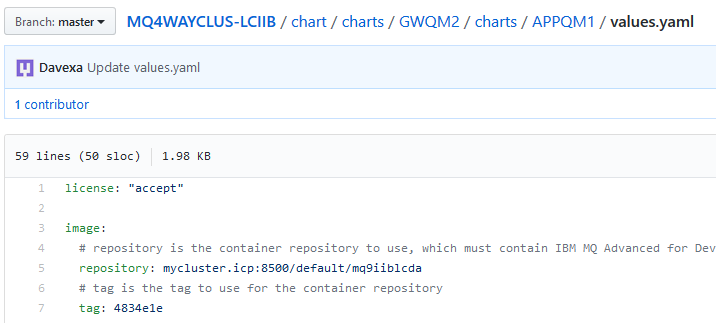
If using the tag latest has not worked for APPQM1 and APPQM2 go back and put an actual tag name in their values.yaml file but most importantly check the repository URLs. (look out for icpcluster / mycluster). You can check your target cluster name on your instance of ICP using the client connection information available via the icon on the top right corner of the ICP Console.

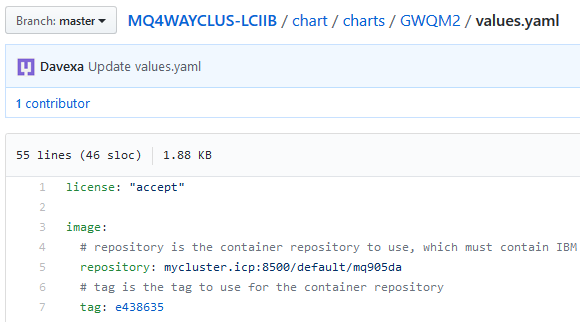
For the GWQM2 MQ only image we did not know the tag name ahead of time as it was only build and tagged on the initial build, therefore this will need to be fixed up in it’s values.yaml chart file.

If your Jenkins pipeline script customization is correct in theory using latest in your values.yamls files should ensure these mismatches don’t happen.

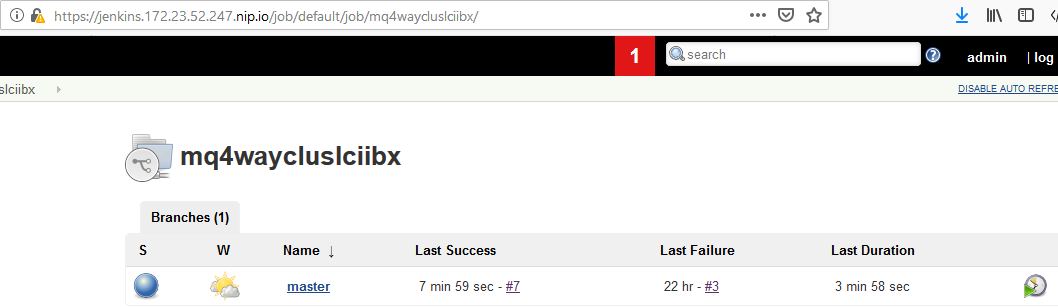
Go back and correct these and then run a new build from Jenkins if necessary





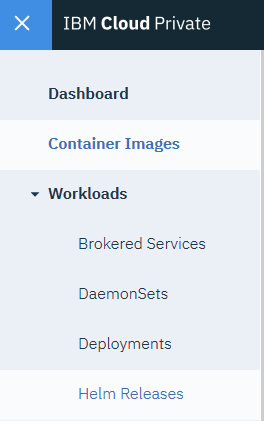


Re-run the Jenkins pipeline by clicking on the Circle with the green triangle.

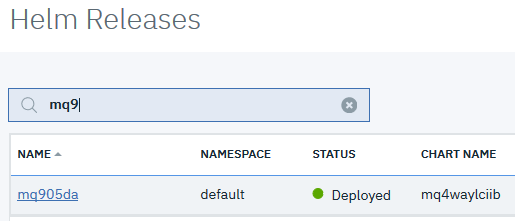


## Validating the MQ4WAYCLUS-LCIIB release

On successful completion of the build and release. Check the Helm Release

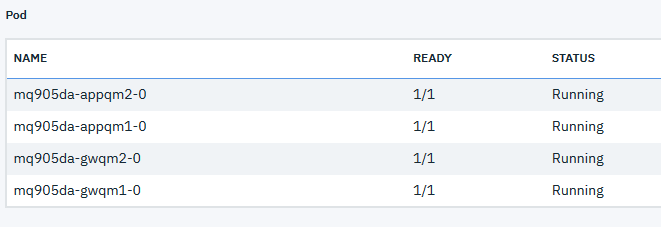


Filter on mq9

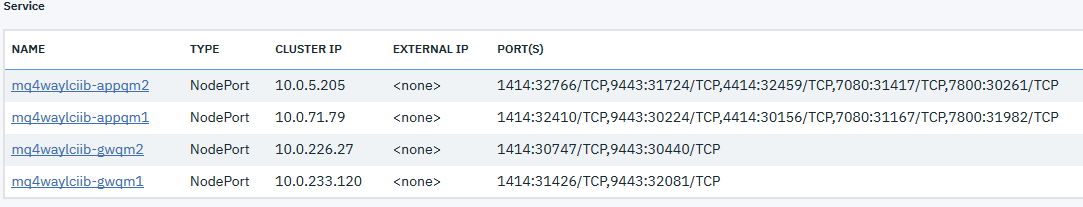


Click on mq905da

And if all is correct you should see all pods running this time.

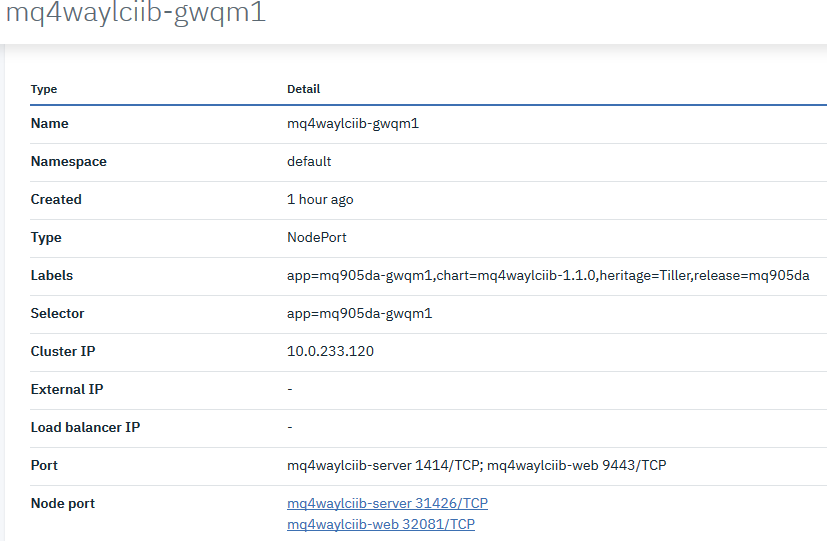


Scroll down to services



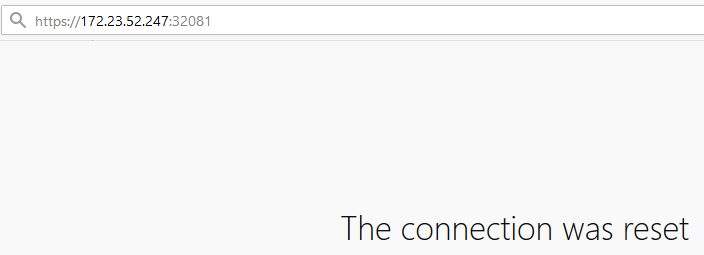
### Testing GWQM1 MQ

Select GWQM1

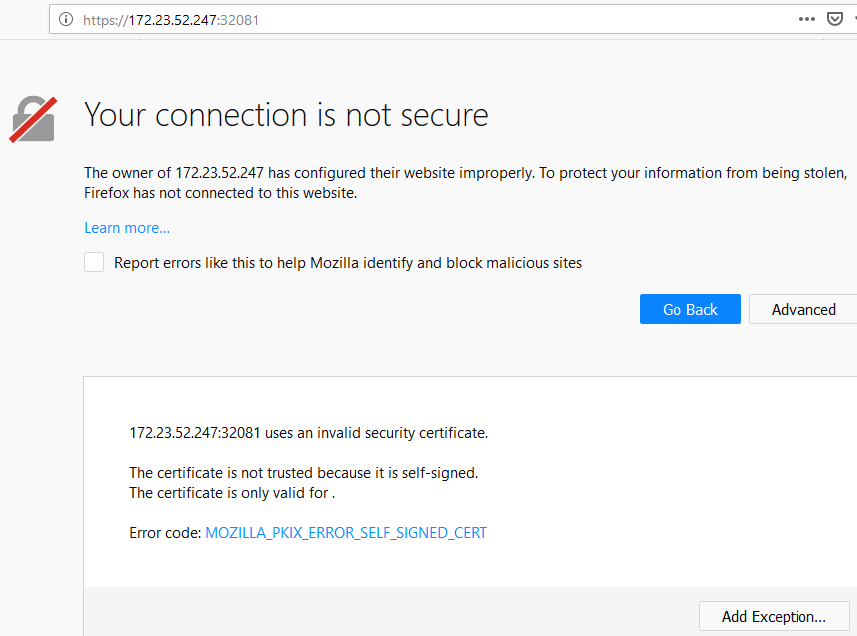


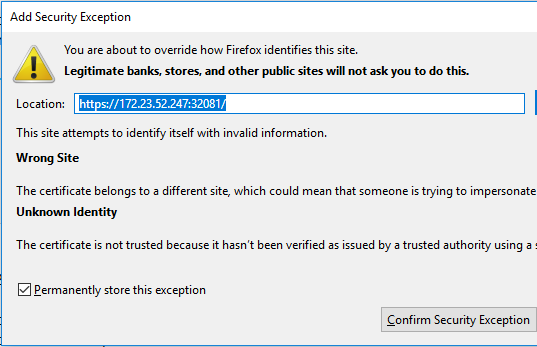
Click on the web console link [mq4waylciib-web nnnn/TCP](http://172.23.52.247:32081)

Add the HTTPS:// to the URL



Add the exception and confirm security

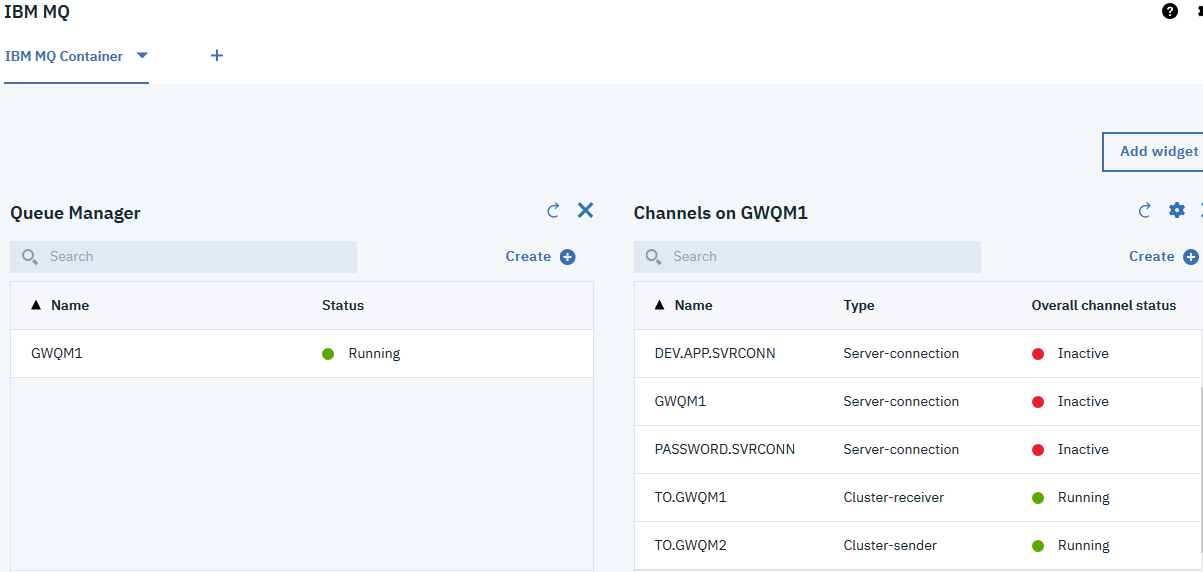




Login into the MQ console with admin/passw0rd



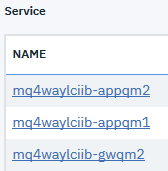
You’ll see GWQM1 is running and there are connected Cluster Receiver and Sender channels



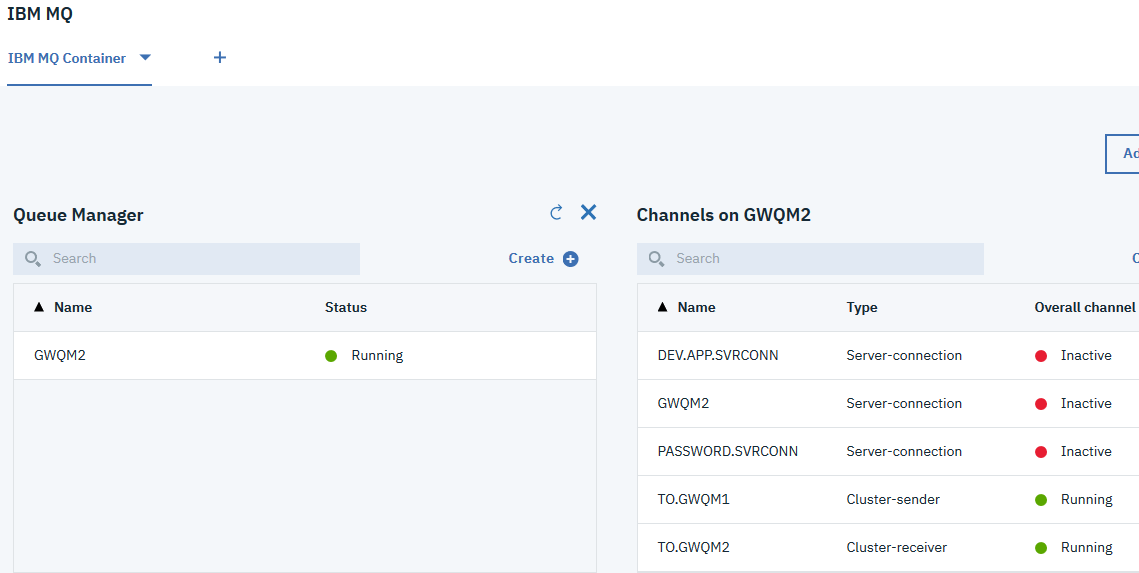
Close the MQ console browser and tab and Repeat the process for GWQM2.

### Testing GWQM2 MQ

Select the GWQM2 service



And repeat the process to bring up the MQ console

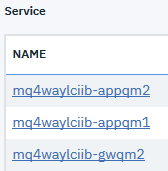


Now lets check APPQM1 queue manager

Close the MQ console browser and tab and Repeat the process for APPQM1.

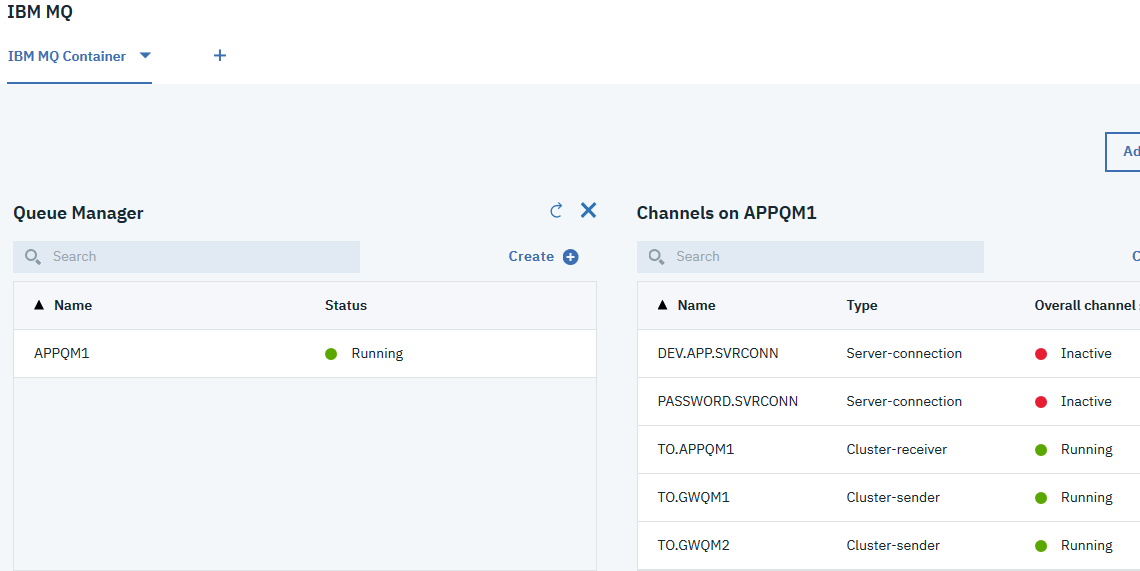
### Testing APPQM1 MQ

Select the APPQM1 service



And repeat the process to bring up the MQ console using the following link after selecting the mq4waylciib-appqm1 service.

[mq4waylciib-console 30224/TCP](http://172.23.52.247:30224)

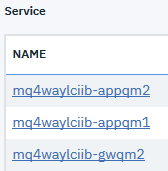


Now lets check APPQM2 queue manager

Close the MQ console browser and tab and Repeat the process for APPQM2.

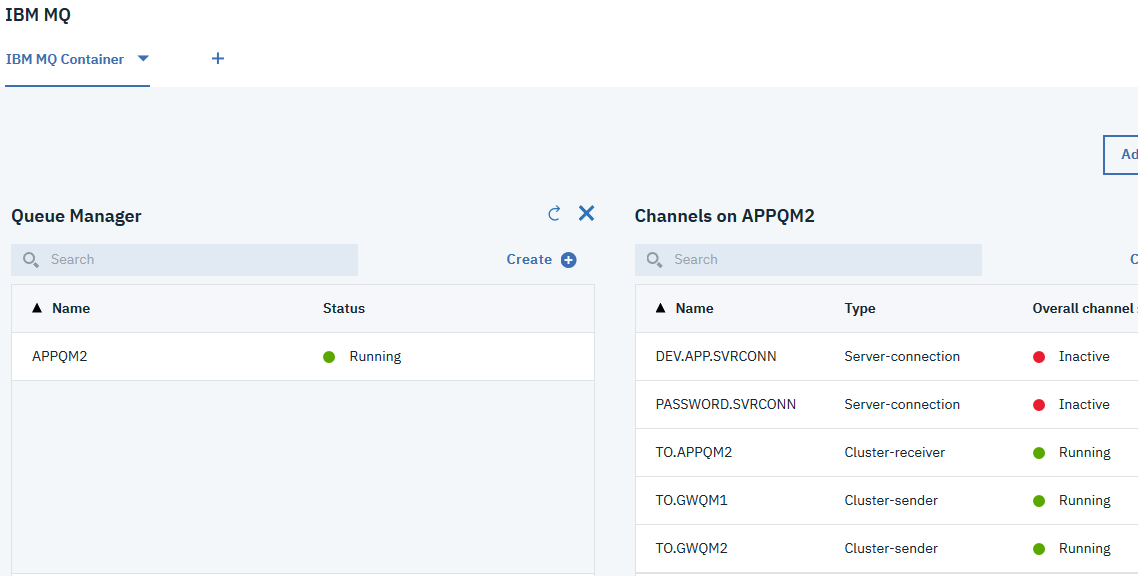
### Testing APPQM2 MQ

Select the APPQM2 service



And repeat the process to bring up the MQ console using the following link after selecting the mq4waylciib-appqm2 service.

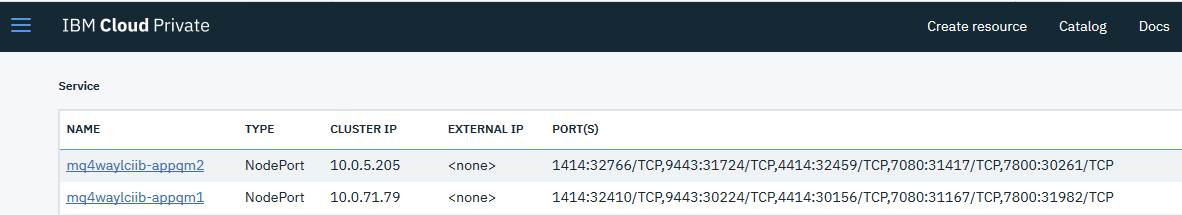
[mq4waylciib-console 31724/TCP](http://172.23.52.247:30224)



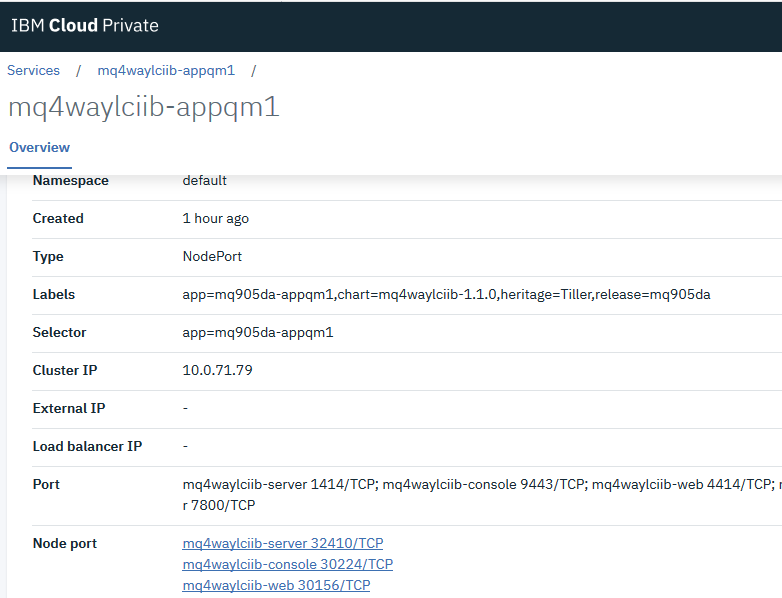
Now lets check the IIB nodes servicing APPQM1 and APPQM2

### Testing APPNODE1 IIB

From the services in the MQ905da Helm Release

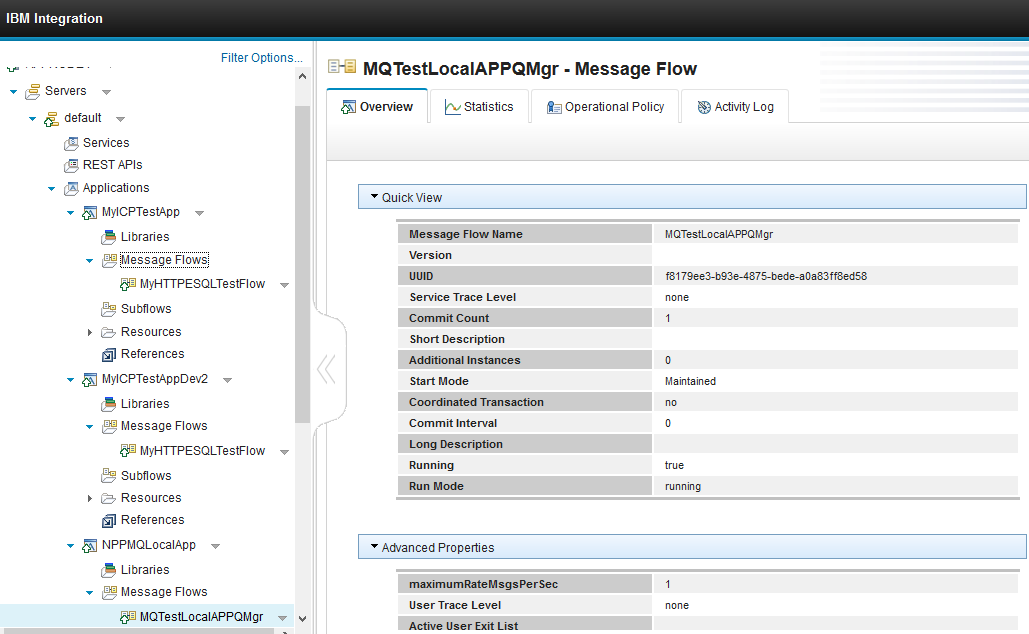


Select APPQM1 service



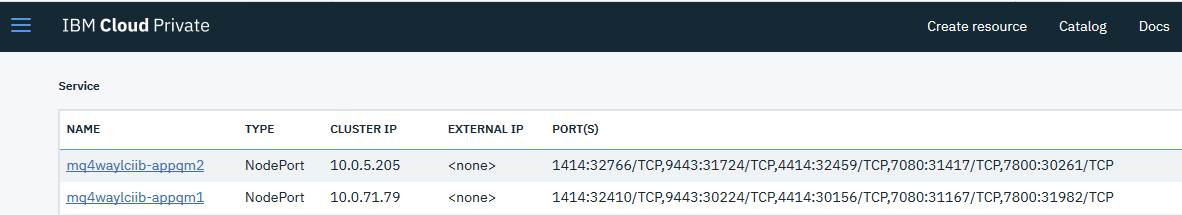
[mq4waylciib-web nnnnn/TCP](http://172.23.52.247:30156)

observe the IIB Node with applications and message flows deployed

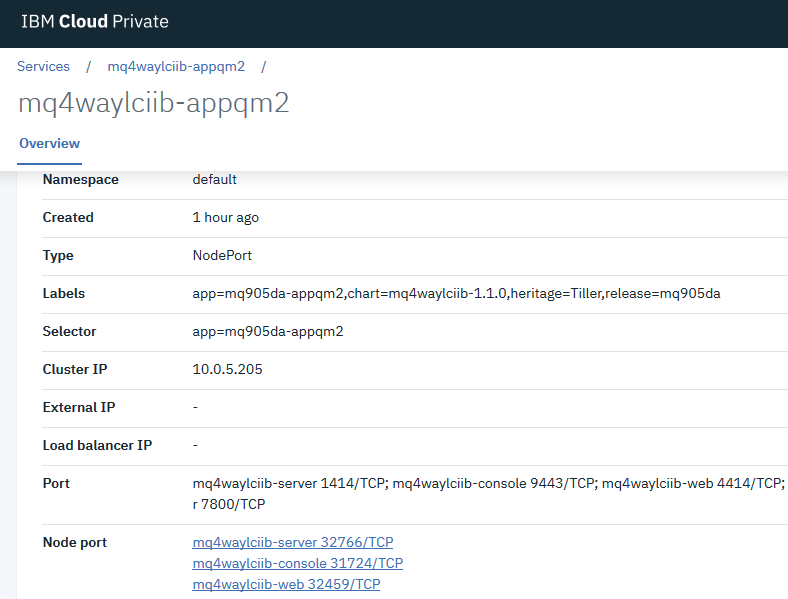


### Testing APPNODE2 IIB

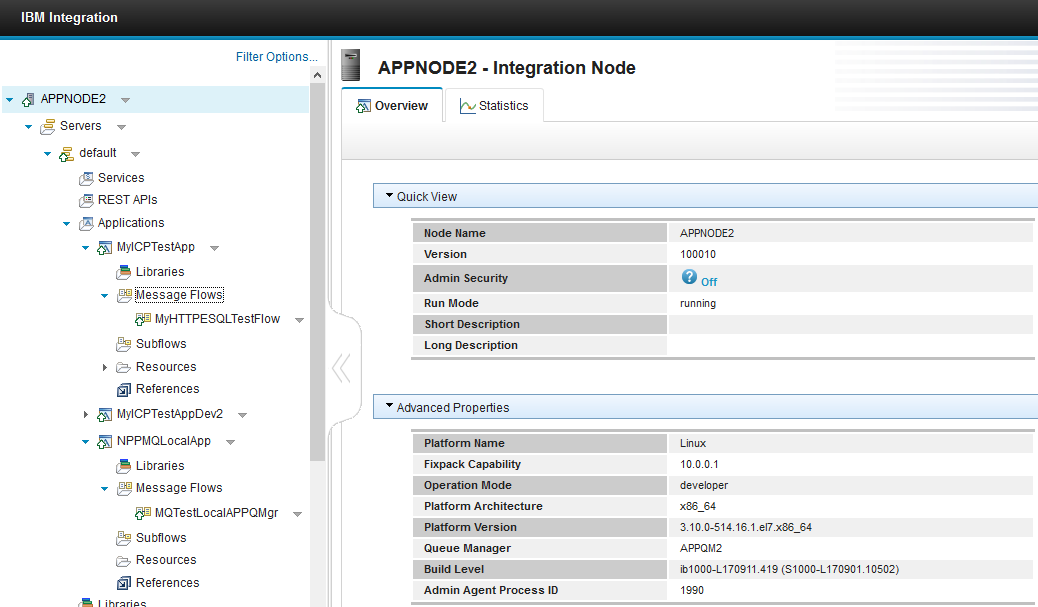
From the services in the MQ905da Helm Release



Select APPQM2 service

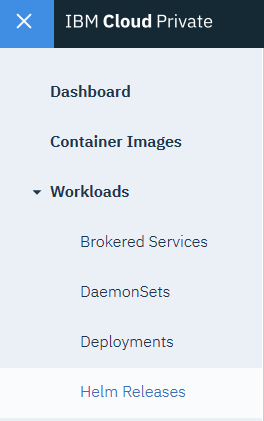


[mq4waylciib-web nnnnn/TCP](http://172.23.52.247:30156)

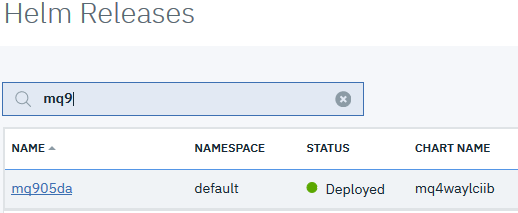


## Removing the Helm Release and cleaning up

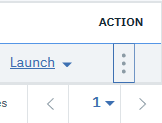
Select Workloads->Helm Release



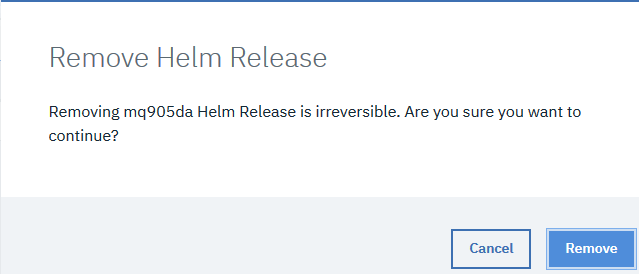
Filter on mq9



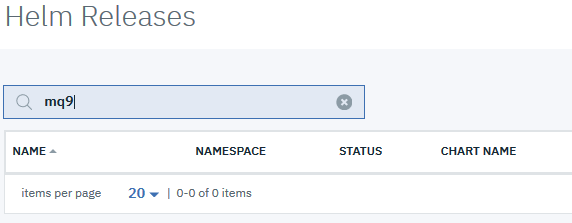
On the mq905da release

 Use the ACTION button (three vertical dots) to select DELETE

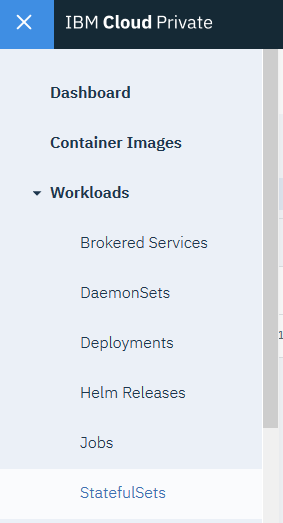
Select Remove at the dialog



Refresh the browser and check the release has been removed

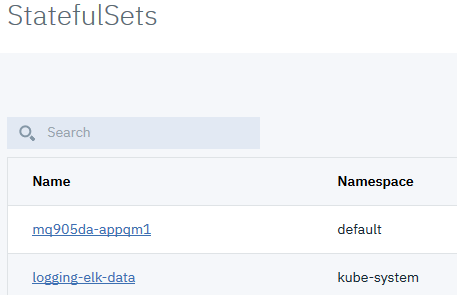


Check the stateful sets. Workloads->Stateful Sets

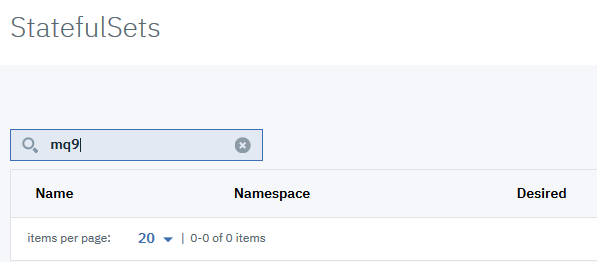


Some times it can take a couple of minutes for the stateful sets to be removed.

Filter on MQ9 and hit refresh a few times they should disappear.



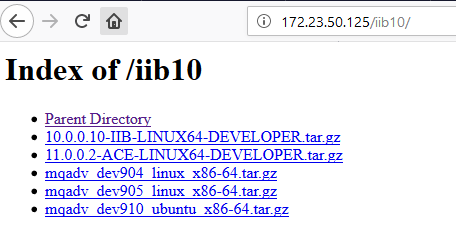
You should see all the mq9 stateful sets disappear



# Changing the version of MQ from 905 to 904 or vice versa for the GWQM1 and GWQM2 containers.

## Updating the Github repository – MQ4WAYCLUS-LCIIB

I have copies of the MQ installation media on a localized drive rather than the public download URL (the public download site works fine but the builds are slower as a result).



### Edit the docker file to change the MQ installation version



Hit the Pencil Icon on the right



Scroll down and amend the ARG MQ\_URL= line to point to different version of the MQ install media



Change the target MQ Installation media file,

ARG MQ\_URL=http://172.23.50.125/iib10/mqadv\_dev905\_linux\_x86-64.tar.gz

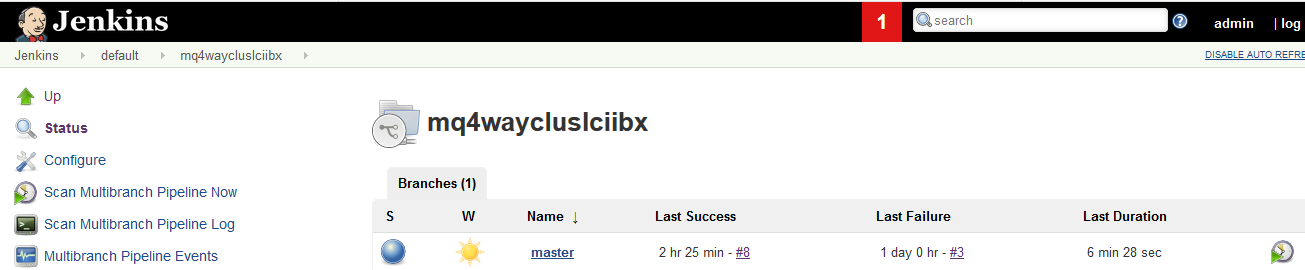
ARG MQ\_URL=http://172.23.50.125/iib10/mqadv\_dev904\_linux\_x86-64.tar.gz

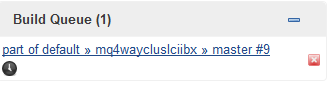
And commit the change 

## Perform a new build and Helm Release

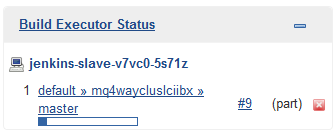
### Rerun the Jenkins pipeline

Goto you Jenkins console window and click the circle with the green triangle on the right





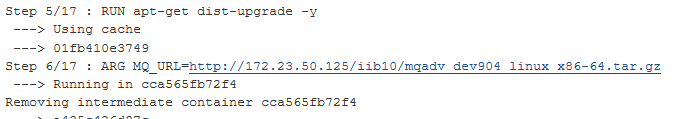
Wait for the slave to initiate the build



Click on the #9 (in this example) and navigate to the console log.

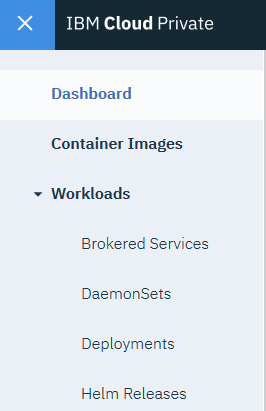
select Console Output to review the build/deploy/release

Note we are now building an image with MQ v904 in this example.

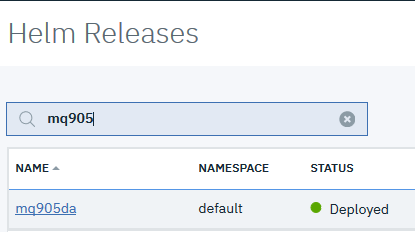


### Check the results of the Helm Release on ICP

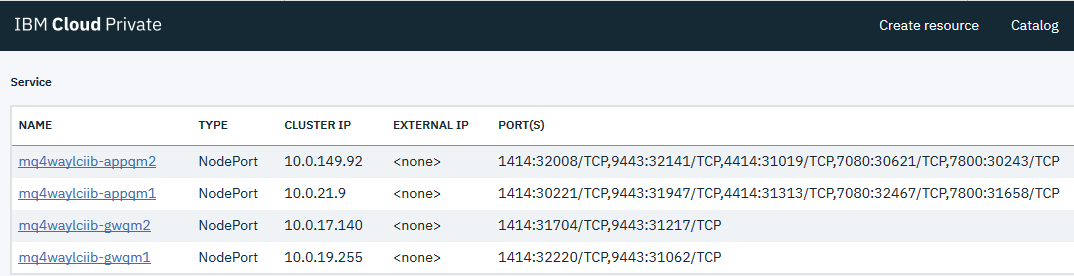
Workloads->Helm Releases



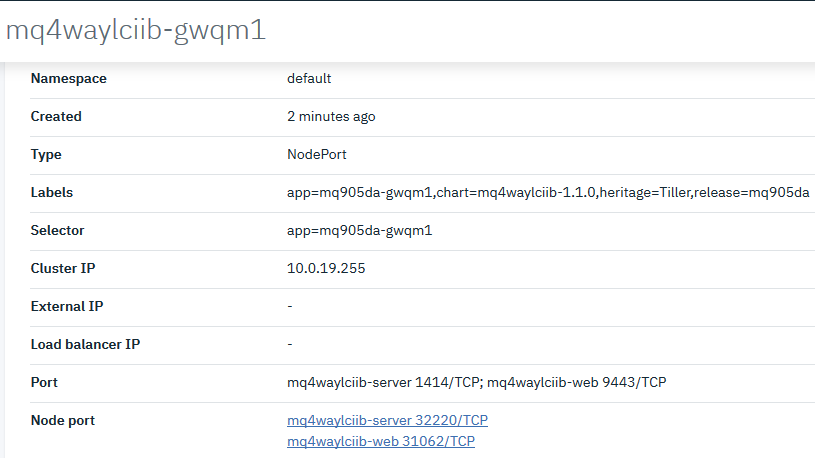
Filter on mq905



Click on mq905da and scroll down to the services



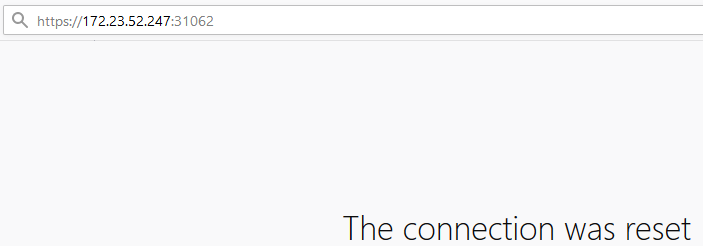
Click on mq4waylciib-gwqm1 link for GWQM1



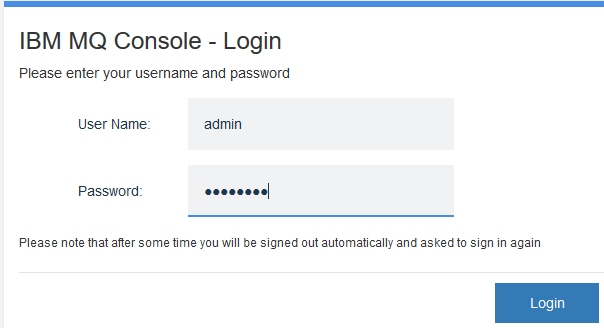
### Validate the version of MQ on GWQM1

Select the mq4waylciib-web nnnnn/TCP link for the MQ console

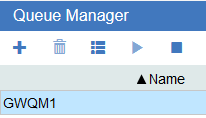
Add the HTTPS:// to the URL



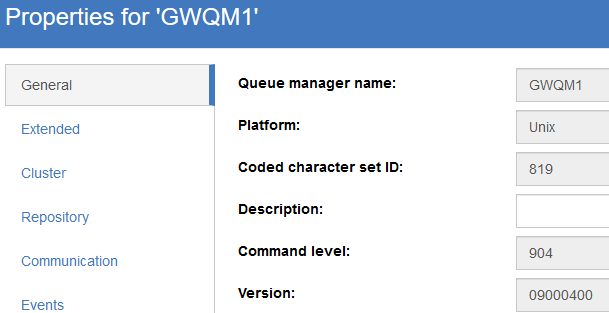
Add and confirm the security exception in the browser



Login with admin/passw0rd



Select GWQM1 and Click on the properties button 



Our Gateway Queue Managers has had its version changed.

The same will be true for GWQM2.