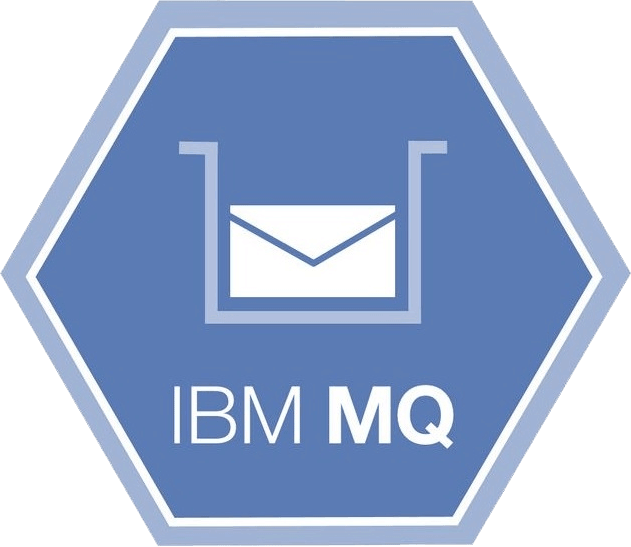


Manual de Implementación IBM MQ on Containers



Índice

[Introducción 3](#_Toc890090770)

[Implementación 3](#_Toc1897533292)

[Openshift Conectado - Cargue Catalogo IBM 3](#_Toc253280265)

[Openshift Desconectado - Catalogo IBM Bastion Host 4](#_Toc1839734369)

[Prerrequisitos instalación 4](#_Toc1500900686)

[Download CASE files 5](#_Toc1288978788)

[Mirror de imagenes 5](#_Toc28478405)

[Creacion catalog Source 8](#_Toc1388517270)

[Cargue catalogo Redhat 8](#_Toc5824769)

[Despliegue Operador IBM MQ 9](#_Toc232640130)

[Creacion de secret para pull image MQ 11](#_Toc1706263226)

[Prueba MQ 16](#_Toc989327094)

[Configureacion certificados TLS para Queue Manager deployment 19](#_Toc706225613)

[Creacion config map con comandos MQSC (ver comandos) 19](#_Toc875274786)

[Creacion route Queue Manager 19](#_Toc1612897036)

[Creacion Queue Manager desde yaml 20](#_Toc2074417514)

[Adicion de colas o modificacion en configuraciones existentes (Manual) 24](#_Toc160693193)

[Despliegue Continuo – Tekton/Openshift pipelines 27](#_Toc310087609)

[Cluster Privado Openshift 27](#_Toc735821068)

[Tekton / Openshift Pipelines 27](#_Toc1114695898)

[Construccion de Imagen con cliente OC 27](#_Toc292962251)

[Definicion task 29](#_Toc857500606)

[Definición Pipeline 29](#_Toc946539584)

[Errores 30](#_Toc635334472)

[ Despliegue de Operador en namespace diferente a Openshift-operators 30](#_Toc769946580)

[ Ratificar el tipo de storage dependiendo del availability que se elija. 31](#_Toc2131833456)

[ Ejecución de pipeline con usuario default 31](#_Toc112171600)

[Adicionales 32](#_Toc1432026300)

[Expand disk vm azure 32](#_Toc1347844764)

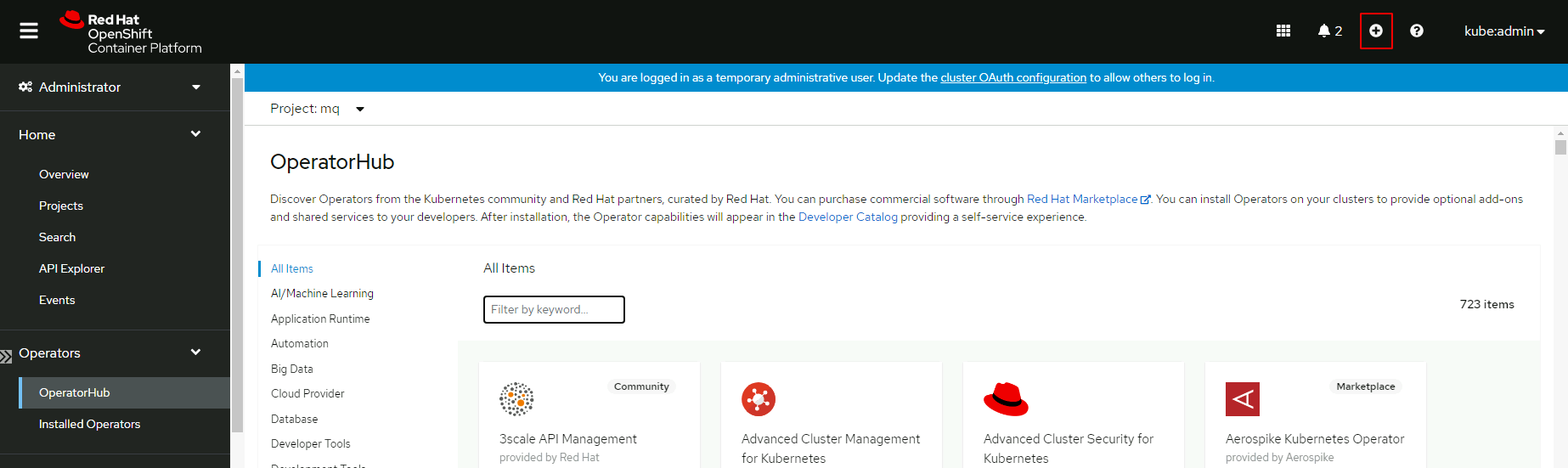
# Introducción

IBM MQ puede transportar cualquier tipo de datos como mensajes, lo que permite a las empresas crear arquitecturas flexibles y reutilizables, como entornos de arquitectura orientada a servicios (SOA) o en nuestro caso arquitecturas orientadas a microservicios. Funciona con una amplia gama de plataformas informáticas, aplicaciones, servicios web y protocolos de comunicación para la entrega de mensajes de alta seguridad. IBM MQ proporciona una capa de comunicaciones para la visibilidad y el control del flujo de mensajes y datos dentro y fuera de su organización.

# Implementación

## Openshift Conectado - Cargue Catalogo IBM ￼

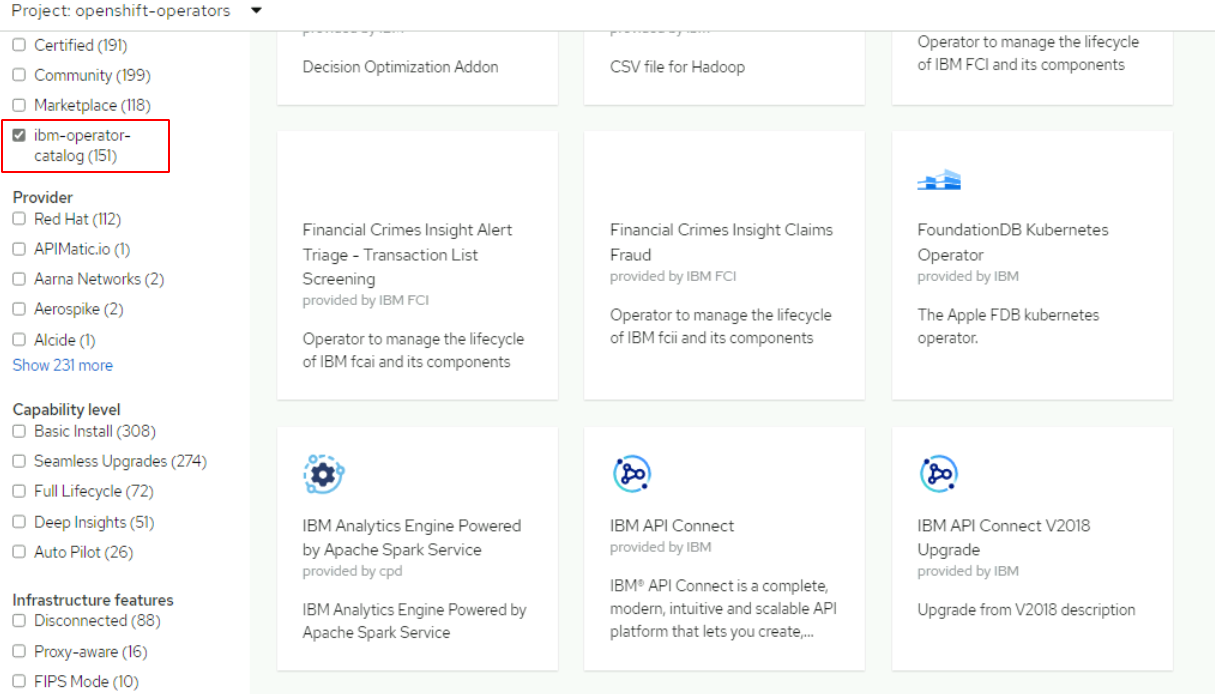
1. Nos dirigimos al boton +



1. Adicionamos catalogo IBM con el siguiente yaml

|  |
| --- |
| apiVersion: operators.coreos.com/v1alpha1 kind: CatalogSource metadata:  name: ibm-operator-catalog  namespace: openshift-marketplace spec:  displayName: ibm-operator-catalog   publisher: IBM Content  sourceType: grpc  image: docker.io/ibmcom/ibm-operator-catalog  updateStrategy:  registryPoll:  interval: 45m |

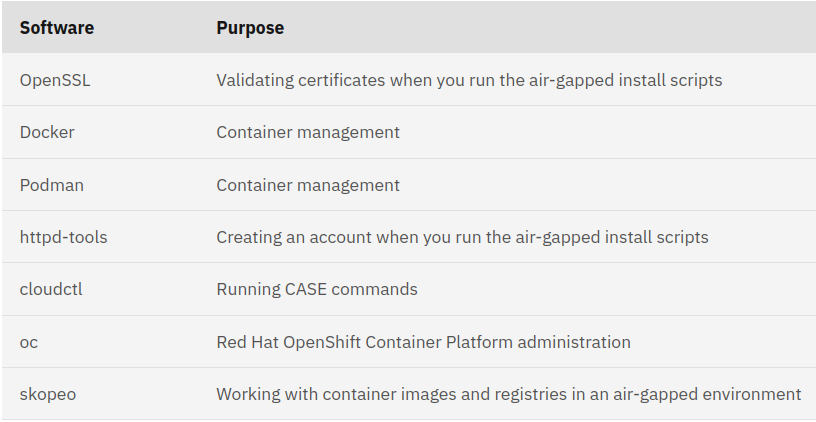
1. Click en el boton Create
2. una vez desplegado dirigirse a Operators > Operator hub



## Openshift Desconectado - Catalogo IBM Bastion Host ￼

<https://www.ibm.com/docs/en/cloud-paks/cp-integration/2021.4?topic=cluster-adding-catalog-sources-bastion-host>

Prerrequisitos instalación￼



### Download CASE files￼

Para nuestro ejemplo vamos a utilizar el directorio $HOME=/data

|  |
| --- |
| export CASE\_NAME=ibm-integration-platform-navigator export CASE\_VERSION=1.6.1 export CASE\_ARCHIVE=${CASE\_NAME}-${CASE\_VERSION}.tgz export CASE\_INVENTORY\_SETUP=platformNavigatorOperator export OFFLINEDIR=$HOME/offline export CASE\_REPO\_PATH=https://github.com/IBM/cloud-pak/raw/master/repo/case |

|  |
| --- |
| cloudctl case save \  --repo $CASE\_REPO\_PATH \  --case $CASE\_NAME \  --version $CASE\_VERSION \  --outputdir $OFFLINEDIR |

### Mirror de imagenes￼

La imagen necesaria para la instalacion es la de IBM MQ por lo cual vamos a definir las siguientes variables de entorno basados en eso.

|  |
| --- |
| export CASE\_NAME=ibm-mq export CASE\_VERSION=1.8.1 export CASE\_ARCHIVE=${CASE\_NAME}-${CASE\_VERSION}.tgz export CASE\_INVENTORY\_SETUP=ibmMQOperator |

|  |
| --- |
| cloudctl case launch \  --case SHOME/offline/$CASE\_ARCHIVE \  --inventory $CASE\_INVENTORY\_SETUP \  --action configure-creds-airgap \  --args "--registry $LOCAL\_DOCKER\_REGISTRY --user admin --pass passw0rd" |

|  |
| --- |
| cloudctl case launch \  --case $HOME/offline/$CASE\_ARCHIVE \  --inventory $CASE\_INVENTORY\_SETUP \  --action configure-creds-airgap \  --args "--registry cp.icr.io --user cp --pass eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJJQk0gTWFya2V0cGxhY2UiLCJpYXQiOjE2NDQ3NzQ3MTEsImp0aSI6IjZhNzBiMTRjNTRjMTQ2ZTVhMTRkZGY5M2U0MmMwODRmIn0.1PPrf5FWBT9c7hBITLl506Z70RVB-o0L4m3zL8Q1QbI" ￼ |

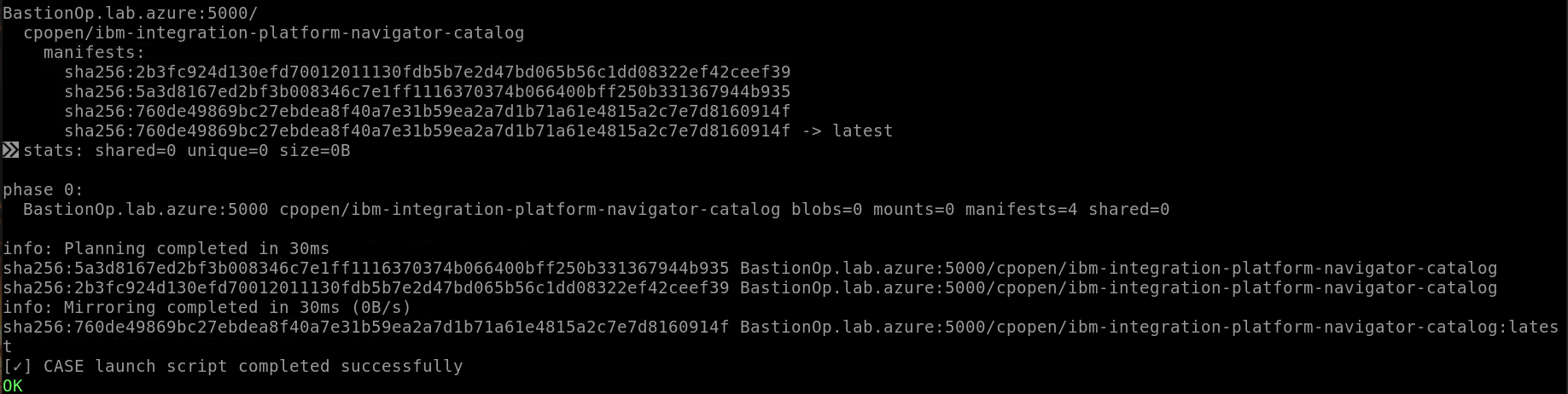
Configuracion final de Mirror y configuracion de cluster

|  |
| --- |
| export NAMESPACE=cp4i  oc create namespace $NAMESPACE  export CASE\_NAME=ibm-cp-integration  export CASE\_VERSION=2.4.0  export CASE\_ARCHIVE=${CASE\_NAME}-${CASE\_VERSION}.tgz  export CASE\_INVENTORY\_SETUP=operator  export OFFLINEDIR=$HOME/offline  export OFFLINEDIR\_ARCHIVE=offline.tgz  export CASE\_REPO\_PATH=https://github.com/IBM/cloud-pak/raw/master/repo/case  export CASE\_LOCAL\_PATH=$OFFLINEDIR/$CASE\_ARCHIVE  export LOCAL\_DOCKER\_REGISTRY\_HOST=BastionOp.lab.azure  export LOCAL\_DOCKER\_REGISTRY\_PORT=5000  export LOCAL\_DOCKER\_REGISTRY=$LOCAL\_DOCKER\_REGISTRY\_HOST:$LOCAL\_DOCKER\_REGISTRY\_PORT  export LOCAL\_DOCKER\_USER=admin  export LOCAL\_DOCKER\_PASSWORD=passw0rd |

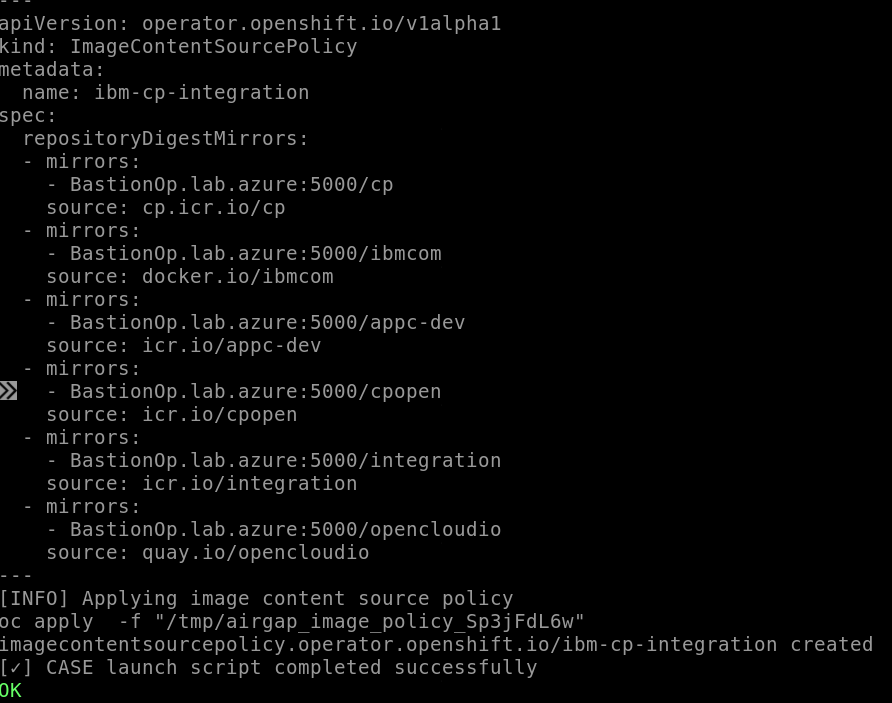
|  |
| --- |
| cloudctl case launch \  --case $HOME/offline/$CASE\_ARCHIVE \  --inventory $CASE\_INVENTORY\_SETUP \  --action configure-creds-airgap \  --args "--registry $LOCAL\_DOCKER\_REGISTRY --user $LOCAL\_DOCKER\_USER --pass $LOCAL\_DOCKER\_PASSWORD" |

Este proceso puede tomar Varias Horas

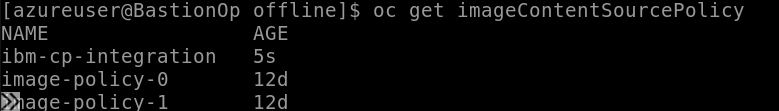
|  |
| --- |
| cloudctl case launch \  --case $CASE\_LOCAL\_PATH \  --inventory $CASE\_INVENTORY\_SETUP \  --action mirror-images \  --args "--registry $LOCAL\_DOCKER\_REGISTRY --inputDir $OFFLINEDIR --resource-group ibmcp4iDefault" |



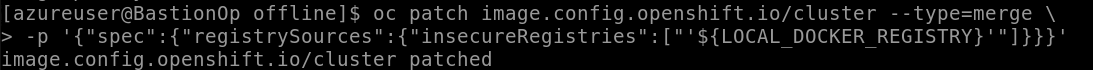
|  |
| --- |
| cloudctl case launch \  --case $HOME/offline/$CASE\_ARCHIVE \  --inventory $CASE\_INVENTORY\_SETUP \  --action configure-cluster-airgap \  --namespace $NAMESPACE \  --args "--registry $LOCAL\_DOCKER\_REGISTRY --user $LOCAL\_DOCKER\_USER --pass $LOCAL\_DOCKER\_PASSWORD --inputDir $OFFLINEDIR" |



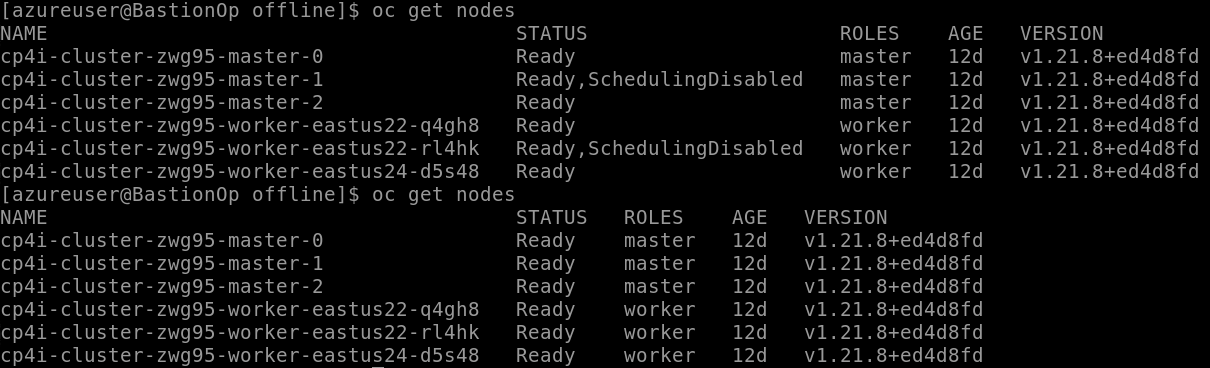
|  |
| --- |
| oc get imageContentSourcePolicy |



|  |
| --- |
| oc patch image.config.openshift.io/cluster --type=merge \ -p '{"spec":{"registrySources":{"insecureRegistries":["'${LOCAL\_DOCKER\_REGISTRY}'"]}}}' |



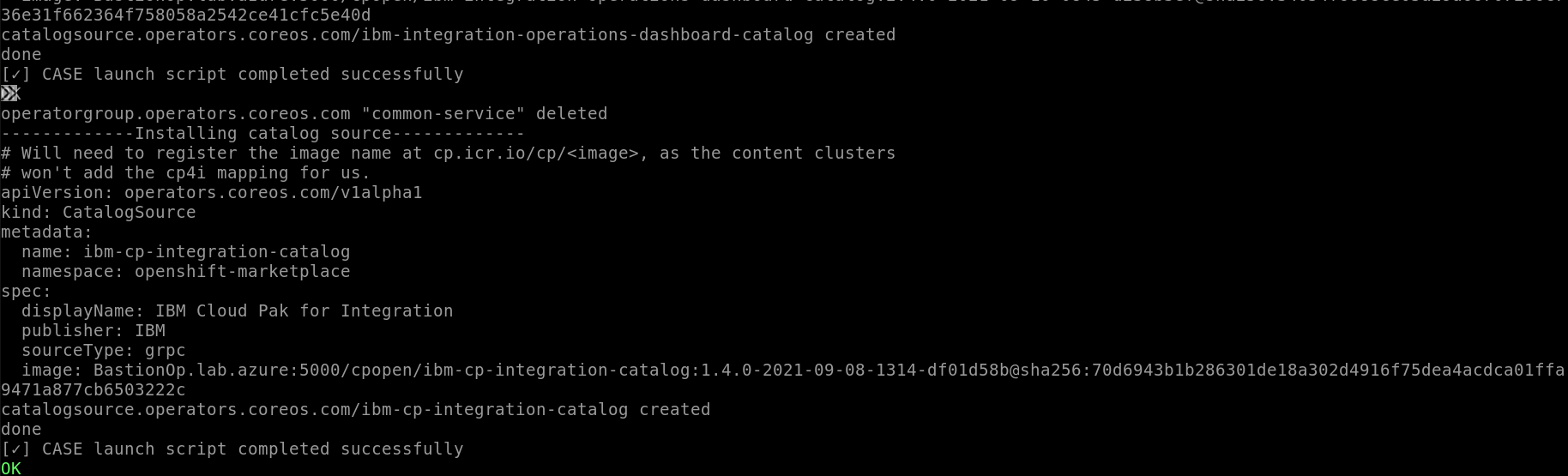
|  |
| --- |
| Oc get nodes |



### Creacion catalog Source

|  |
| --- |
| export NAMESPACE=cp4i |

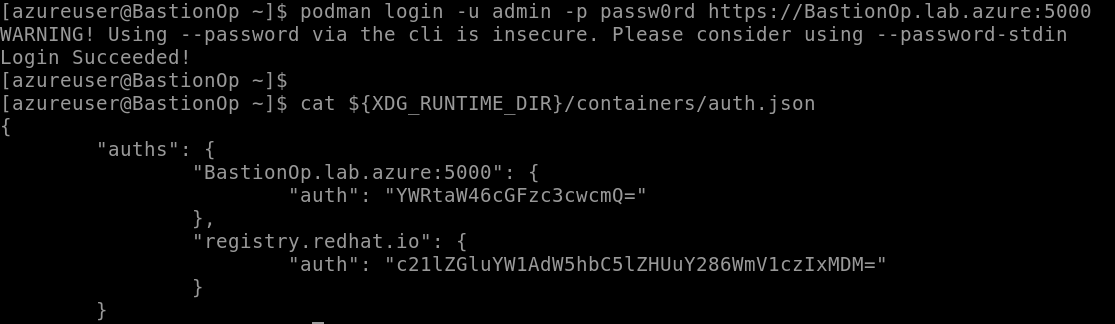
|  |
| --- |
| cloudctl case launch \  --case $HOME/offline/${CASE\_ARCHIVE} \  --inventory ${CASE\_INVENTORY\_SETUP} \  --action install-catalog \  --namespace ${NAMESPACE} \  --args "--registry ${LOCAL\_DOCKER\_REGISTRY} --inputDir $HOME/offline --recursive" |



|  |
| --- |
| oc get pods -n openshift-marketplace oc get catalogsource -n openshift-marketplace |

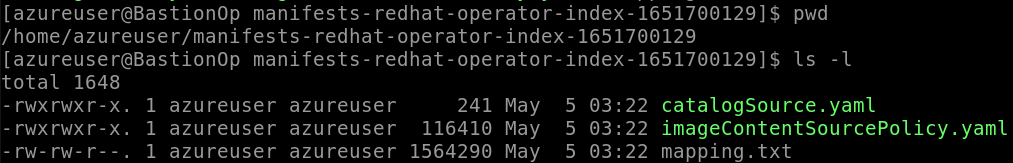
### Cargue catalogo Redhat￼

|  |
| --- |
| podman login registry.redhat.io  podman login -u admin -p passw0rd https://BastionOp.lab.azure:5000 |



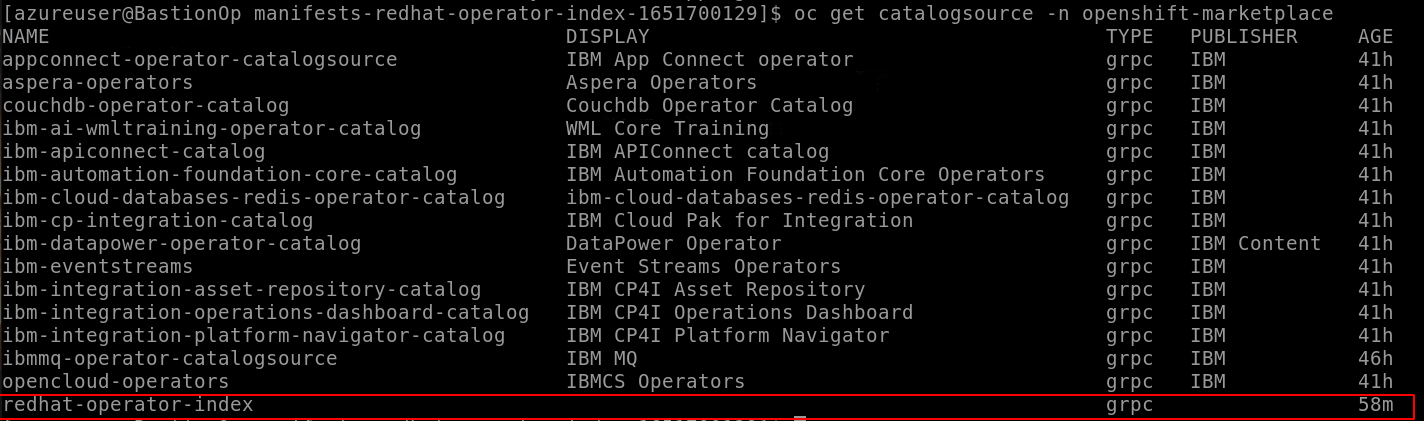
|  |
| --- |
| REG\_CREDS=${XDG\_RUNTIME\_DIR}/containers/auth.json  oc adm catalog mirror registry.redhat.io/redhat/redhat-operator-index:v4.6 BastionOp.lab.azure:5000/olm-mirror -a ${REG\_CREDS} --insecure |

Una vez finalizado encontraremos una carpeta con la definicion del catalogo



|  |
| --- |
| oc create -f <path/to/manifests/dir>/imageContentSourcePolicy.yaml  oc apply -f catalogSource.yaml |

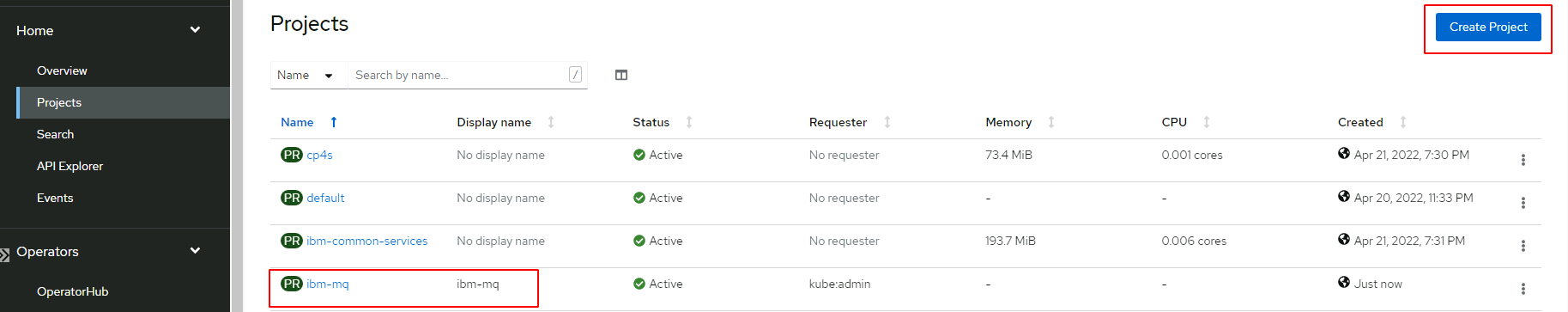
|  |
| --- |
| oc get catalogsource –n openshift-marketplace |



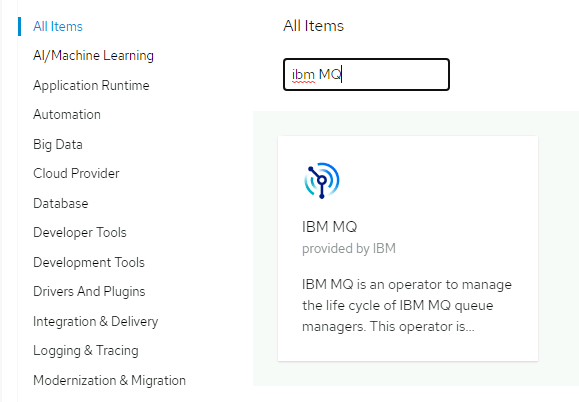
## Despliegue Operador IBM MQ￼

<https://www.ibm.com/docs/en/ibm-mq/9.1?topic=openshift-installing-mq-operator-using-web-console>

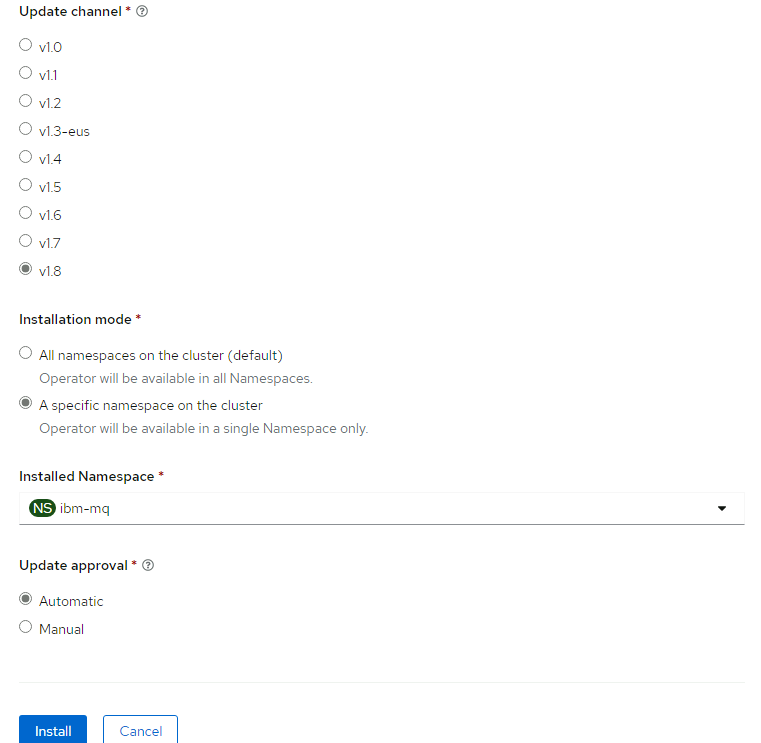
1. En el panel de Navegacion dirigimos a home > projects
2. Creamos un nuevo proyecto con el boton Create Project y le damos un nombre y descripcion por ejemplo nombre: ibm-mq descripcion: ibm-mq



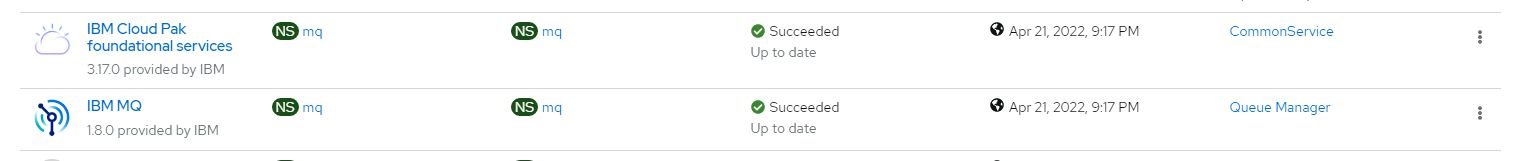
1. En el panel de navegacion nos dirigimos a Operators > OperatorHub
2. En el cuadro de filtro buscamos por IBM MQ y damos click en Install



1. Seleccionamos la ultima version disponible y elegimos el namespace previamente creado



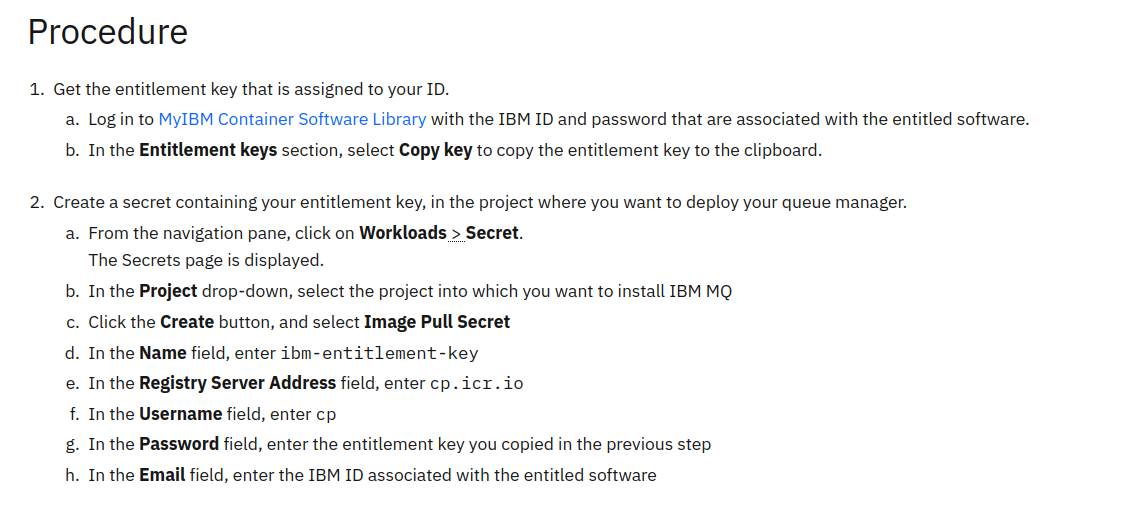
1. Una vez finalizada la instalacion en el menu Operators > Installed Operators nos debera aparecer el Operador de IBM MQ



Nota: el despliegue del Operador unicamente se pudo hacer en el namespace Openshift Operators esto debido a que por defecto al momento de la creacion hace un llamado a un servicio ***ibm-mq-webhook.openshift-operators.svc*** [***(ver error)***](#_Despliegue_de_Operador)***.*** Este endpoint no es modificable al momento de realizar la instalacion por lo cual se deja como una limitante del producto.

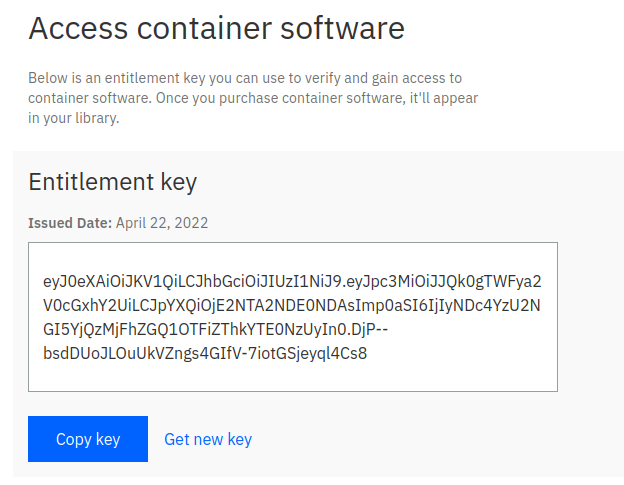
### Creacion de secret para pull image MQ￼

<https://www.ibm.com/docs/en/ibm-mq/9.2?topic=pyrhopm-preparing-your-red-hat-openshift-project-mq-using-red-hat-openshift-web-console>

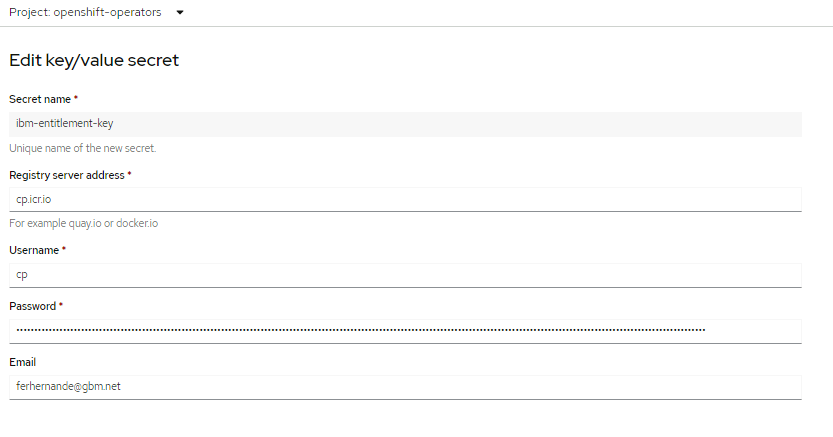


Para obtener el password ingresar a la siguiente ruta y copiar el key

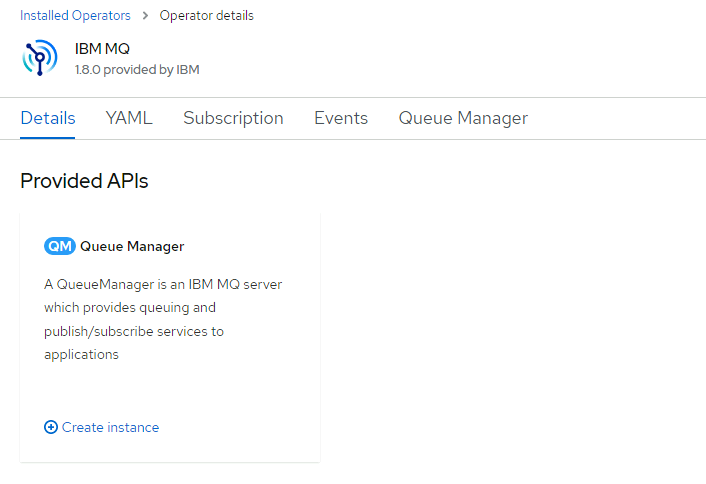
<https://myibm.ibm.com/products-services/containerlibrary>



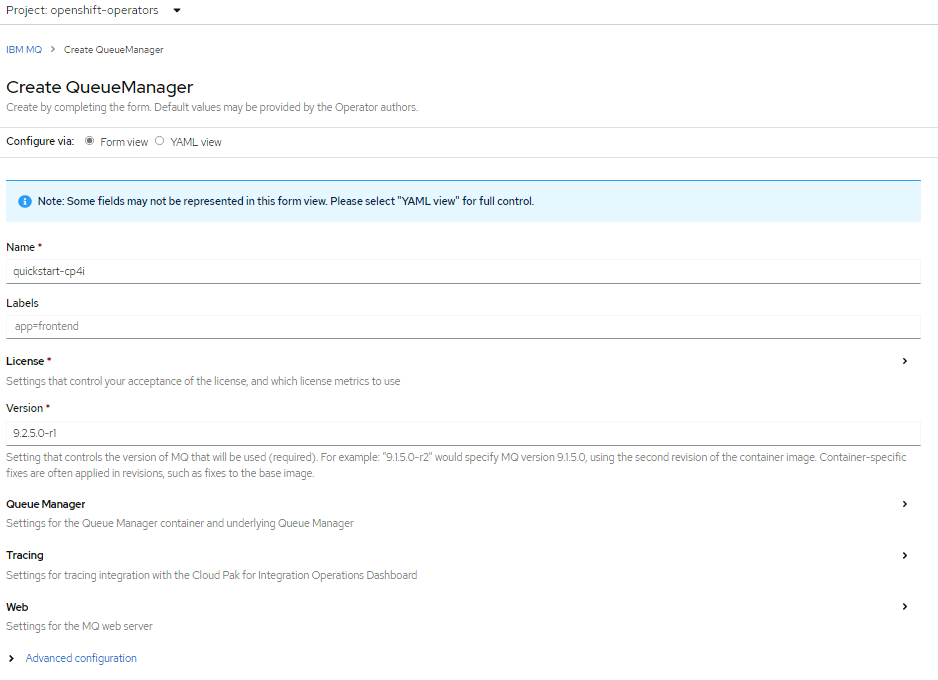
De esta forma deberia quedar creado nuestro secret en el namespace Openshift-operators



1. Ingresamos en el IBM MQ y procedemos con la creacion del Queue Manager



1. En este apartado vamos a diligenciar los campos nombre, etiquetas, version

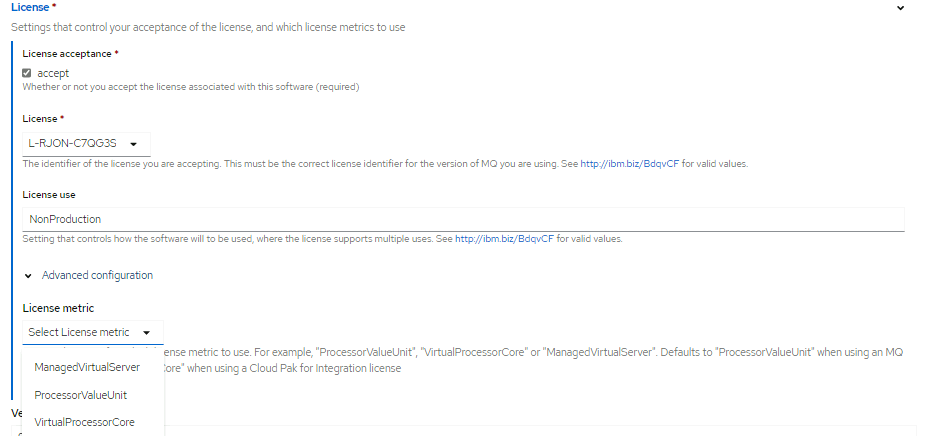


1. Licencia

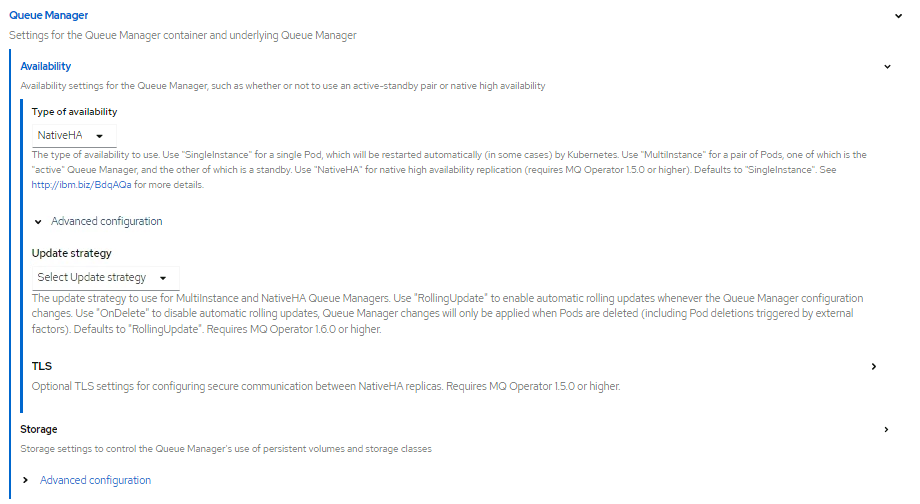
Dependera de aquella que el cliente alla adquirido para mayor detalle en el siguiente link encontrara el listado de las licencias disponibles

<https://www.ibm.com/docs/en/ibm-mq/9.2?topic=mqibmcomv1beta1-licensing-reference>

Habilitamos la opcion Accept, seleccionamos la licencia y dependiendo de la misma en el apartado License use podremos seleccionar entre (Production,NonProduction, development)



1. Queue Manager



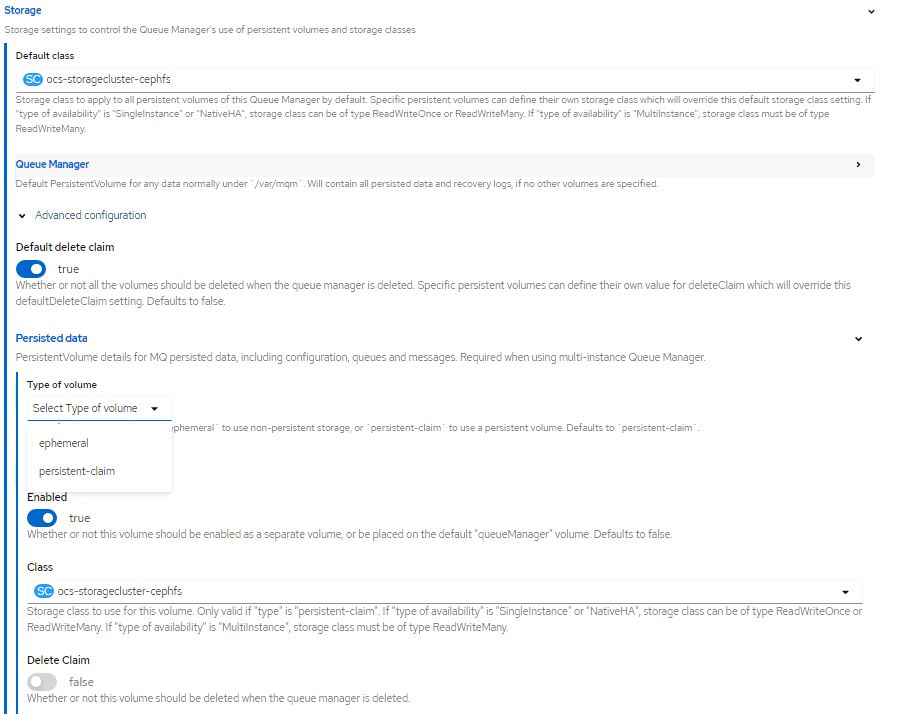
El tipo de disponibilidad a utilizar. Utilice "Instancia única" para un único pod, que Kubernetes reiniciará automáticamente (en algunos casos). Utilice "MultiInstance" para un par de Pods, uno de los cuales es el Administrador de colas "activo" y el otro es un modo de espera. Utilice "NativeHA" para la replicación nativa de alta disponibilidad (requiere MQ Operator 1.5.0 o superior). El valor predeterminado es "Instancia única”

Update strategy: se recomienda usar RollingUpdate con el fin de actualizar poco a poco los pods desplegados sin generar fuera de servicio y garantizando la estabilidad de la nueva version.

Storage: dependiendo del tipo de disponibilidad debemos elegir un storageclass con caracteristica RWX (read Write Many) o RWO (Read Write Once) si es single instance o NativeHA. Si es MultiInstance unicamente se adminte RWX. Para este ejemplo se utilizo el storage class de OCS (ocs-storagecluster-cephfs)

Habilitar la opcion Default delete Claim con el fin de garantizar la depuracion el cluster

Persistent data: si se requiere que las colas y mensajes sean persistentes seleccionar un persistem claim (necesario para configuración multiInstance)



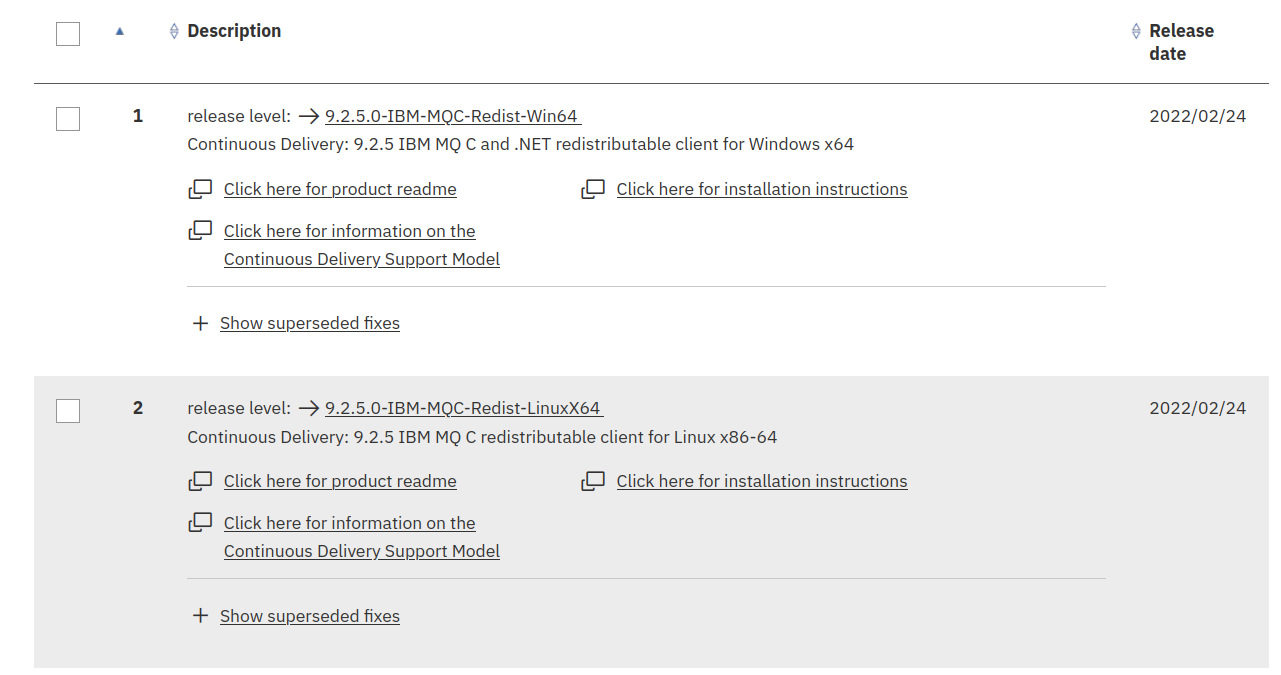
1. Tracing
2. Web

## Prueba MQ ￼

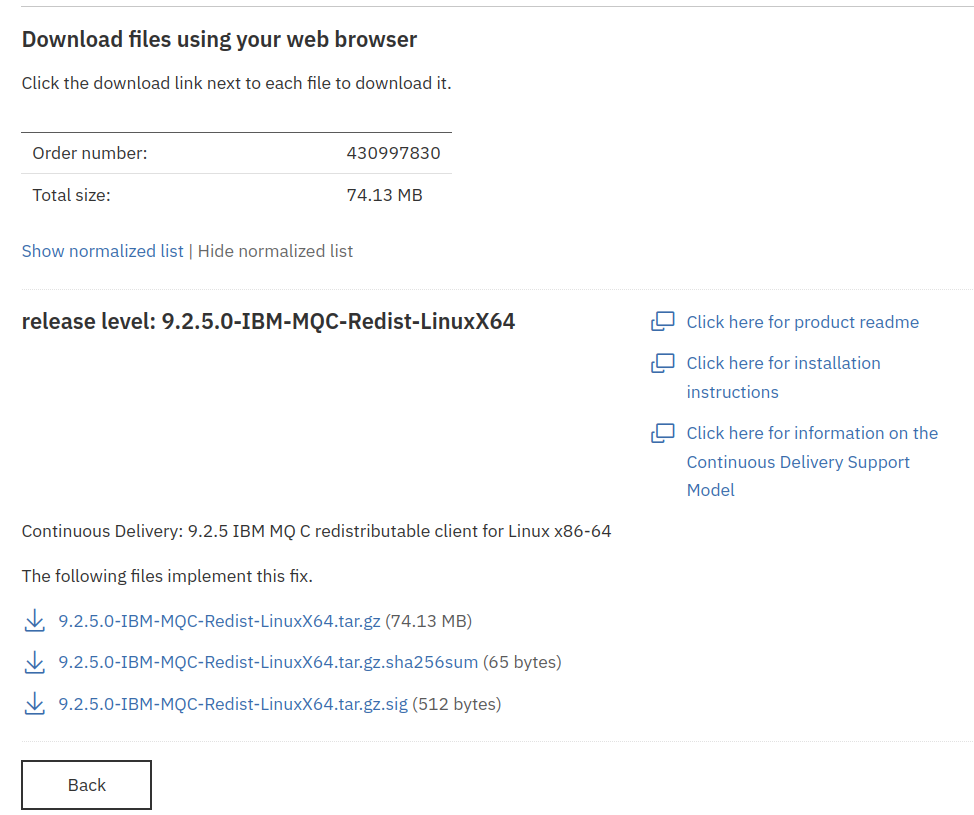
<https://www.ibm.com/docs/en/ibm-mq/9.2?topic=manager-example-configuring-tls>

1. Ingresar a fix central para descargar el cliente MQ

[https://www.ibm.com/support/fixcentral/swg/selectFixes?parent=ibm~WebSphere&product=ibm/WebSphere/WebSphere+MQ&release=9.2.0.0&platform=All&function=fixid&fixids=\*IBM-MQC-Redist-](https://www.ibm.com/support/fixcentral/swg/selectFixes?parent=ibm~WebSphere&product=ibm/WebSphere/WebSphere+MQ&release=9.2.0.0&platform=All&function=fixid&fixids=*IBM-MQC-Redist-)\*



1. Para nuestro caso puntual utilizaremos el cliente para Linux

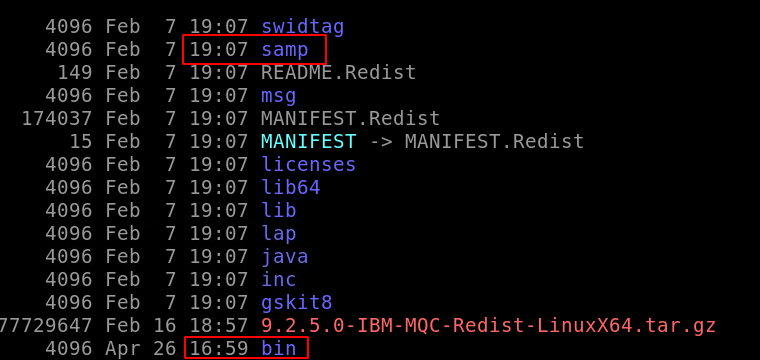


1. Tambien podemos descargar directamente desde la linea de comandos con wget

|  |
| --- |
| wget https://ak-delivery04-mul.dhe.ibm.com/sdfdl/v2/sar/CM/WS/0abaq/0/Xa.2/Xb.jusyLTSp44S0eQvgboPXUhLGv6TVACNTiPaH0JTA8GZyYmqLx-ad1AV62FI/Xc.CM/W  S/0abaq/0/9.2.5.0-IBM-MQC-Redist-LinuxX64.tar.gz/Xd./Xf.LPR.D1VK/Xg.11742706/Xi.habanero/XY.habanero/XZ.wqLco-Fk88XwMu0u0kXoqem52nhH6oiL/9.2.5.0-IBM-MQC  -Redist-LinuxX64.tar.gz |

1. Descomprimir en ruta deseada que para nuestro ejemplo llamaremos /data

|  |
| --- |
| tar xvf 9.2.5.0-IBM-MQC-Redist-LinuxX64.tar.gz |

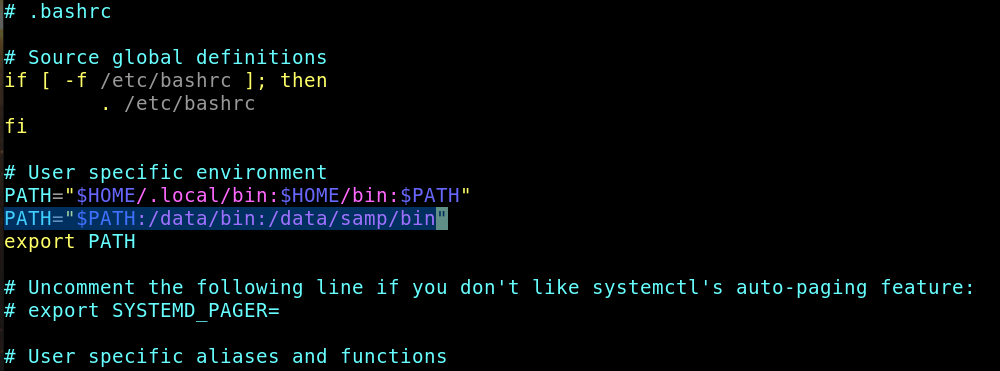


1. Para poder ejecutar los comandos de los pasos siguientes es recomendable cargar en nuestro PATH los siguientes binarios /data/bin y /data/samp/bin. En el home del usuario encontraras el archivo .bashrc
2. Editar el archivo

|  |
| --- |
| vi ~/.bashrc |

1. Adicionar la siguiente linea antes del export de la variable path

|  |
| --- |
| PATH="$PATH:/data/bin:/data/samp/bin"  source ~/.bashrc |



1. Generar una llave privada autofirmada con su respectiva llave publica (se requiere tener instalado openssl o cualquier otro software que nos permita generar certificados)

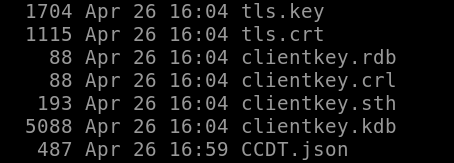
|  |
| --- |
| openssl req -newkey rsa:2048 -nodes -keyout tls.key -subj "/CN=localhost" -x509 -days 3650 -out tls.crt |

1. La Key de la base de datos es usada como trustore para el cliente de aplicacion. Crear el client key database

|  |
| --- |
| runmqakm -keydb -create -db clientkey.kdb -pw password -type cms -stash |

1. Adicionar la llave publica previamente creada a la llave de cliente de base de datos.

|  |
| --- |
| runmqakm -cert -add -db clientkey.kdb -label mqservercert -file tls.crt -format ascii -stashed |



### Configureacion certificados TLS para Queue Manager deployment￼

1. Crear TLS secret con la key previamente generada

|  |
| --- |
| oc create secret tls example-tls-secret --key="tls.key" --cert="tls.crt" -n openshift-operators |

### Creacion config map con comandos MQSC

([ver comandos](https://www.ibm.com/docs/en/ibm-mq/9.0?topic=queues-define-qlocal))￼

1. Crearemos el configmap con el detalle de la nueva cola, canal

|  |
| --- |
| apiVersion: v1 kind: ConfigMap metadata:  name: example-tls-configmap data:  tls.mqsc: |  DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE   DEFINE CHANNEL(SECUREQMCHL) CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCAUTH(OPTIONAL) SSLCIPH('ANY\_TLS12\_OR\_HIGHER')  SET CHLAUTH(SECUREQMCHL) TYPE(BLOCKUSER) USERLIST('nobody') ACTION(ADD) |

### Creacion route Queue Manager￼

1. Creamos el route correspondiente al service de nuestro queue Manager

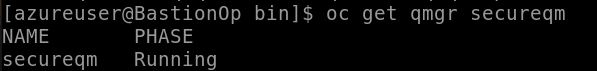
|  |
| --- |
| apiVersion: route.openshift.io/v1 kind: Route metadata:  name: example-tls-route  spec:  host: secureqmchl.chl.mq.ibm.com  to:  kind: Service  name: secureqm-ibm-mq  port:  targetPort: 1414  tls:  termination: passthrough |

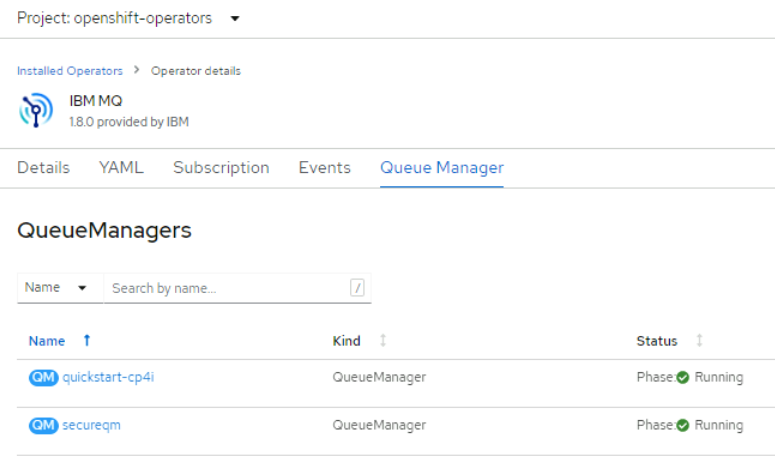
### Creacion Queue Manager desde yaml￼

|  |
| --- |
| apiVersion: mq.ibm.com/v1beta1 kind: QueueManager metadata:  name: secureqm spec:  license:  accept: true  license: L-RJON-C7QG3S  use: Production  queueManager:  name: SECUREQM  mqsc:  - configMap:  name: example-tls-configmap  items:  - tls.mqsc  availability:  type: NativeHA  updateStrategy: RollingUpdate  storage:  defaultClass: ocs-storagecluster-cephfs  persistedData:  enabled: false  queueManager:  class: ocs-storagecluster-cephfs  deleteClaim: true  size: 2Gi  type: persistent-claim  recoveryLogs:  enabled: false  template:  pod:  containers:  - env:  - name: MQSNOAUT  value: 'yes'  name: qmgr  version: 9.2.5.0-r1  web:  enabled: true  pki:  keys:  - name: example  secret:  secretName: example-tls-secret  items:   - tls.key  - tls.crt |

1. Para verificar que el queue manager fue desplegado verifiquelo con el siguiente comando

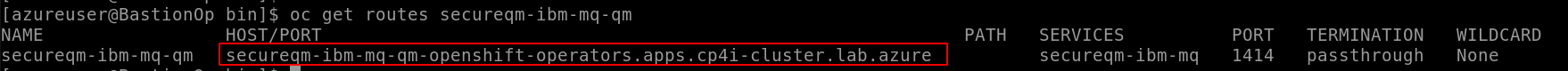
|  |
| --- |
| oc get qmgr secureqm |





1. Validar url del route creado

|  |
| --- |
| oc get routes secureqm-ibm-mq-qm |



Creacion archivo CCDT.json

1. Este archivo contiene todos los datos del queue manager, la estructura es la siguiente

|  |
| --- |
| {  "channel":  [  {  "name": "SECUREQMCHL",  "clientConnection":  {  "connection":  [  {  "host": " secureqm-ibm-mq-qm-openshift-operators.apps.cp4i-cluster.lab.azure ",  "port": 443  }  ],  "queueManager": "SECUREQM"  },  "transmissionSecurity":  {  "cipherSpecification": "ECDHE\_RSA\_AES\_128\_CBC\_SHA256"  },  "type": "clientConnection"  }  ] } |

Cargue variables de entorno

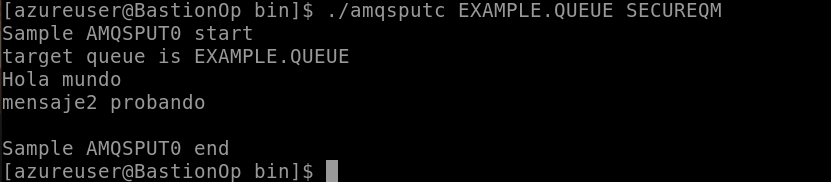
1. Estas vaiables seran leidas por los binarios amqsputc(poner mensaje en la cola) y amqsgetc(obtener mensaje)

|  |
| --- |
| export MQCCDTURL='<full path to file>/CCDT.JSON' export MQSSLKEYR='<full path to file>/clientkey'  export MQCCDTURL='/data/mq/bin/CCDT.json' export MQSSLKEYR='/data/mq/bin/clientkey' |

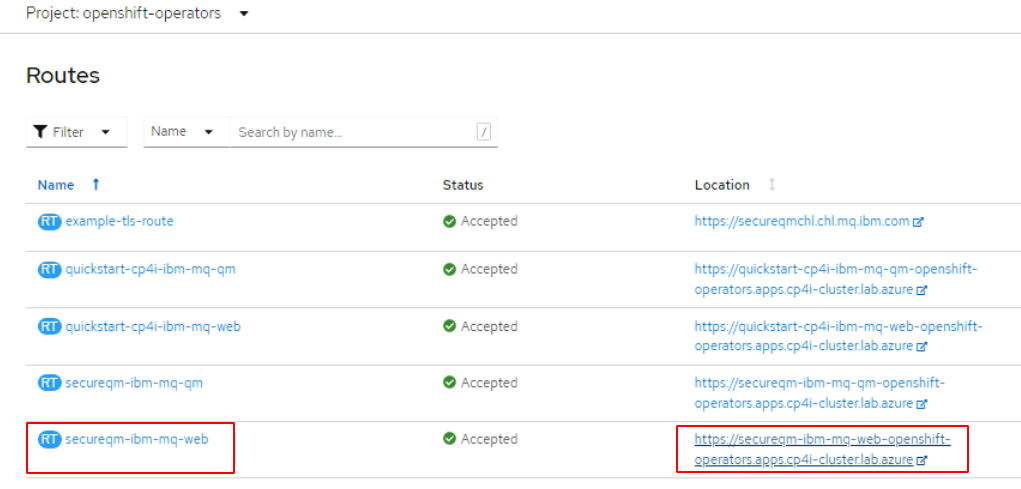
1. Para poner mensaje haremos uso del comando amqsputc

|  |
| --- |
| amqsputc EXAMPLE.QUEUE SECUREQM |

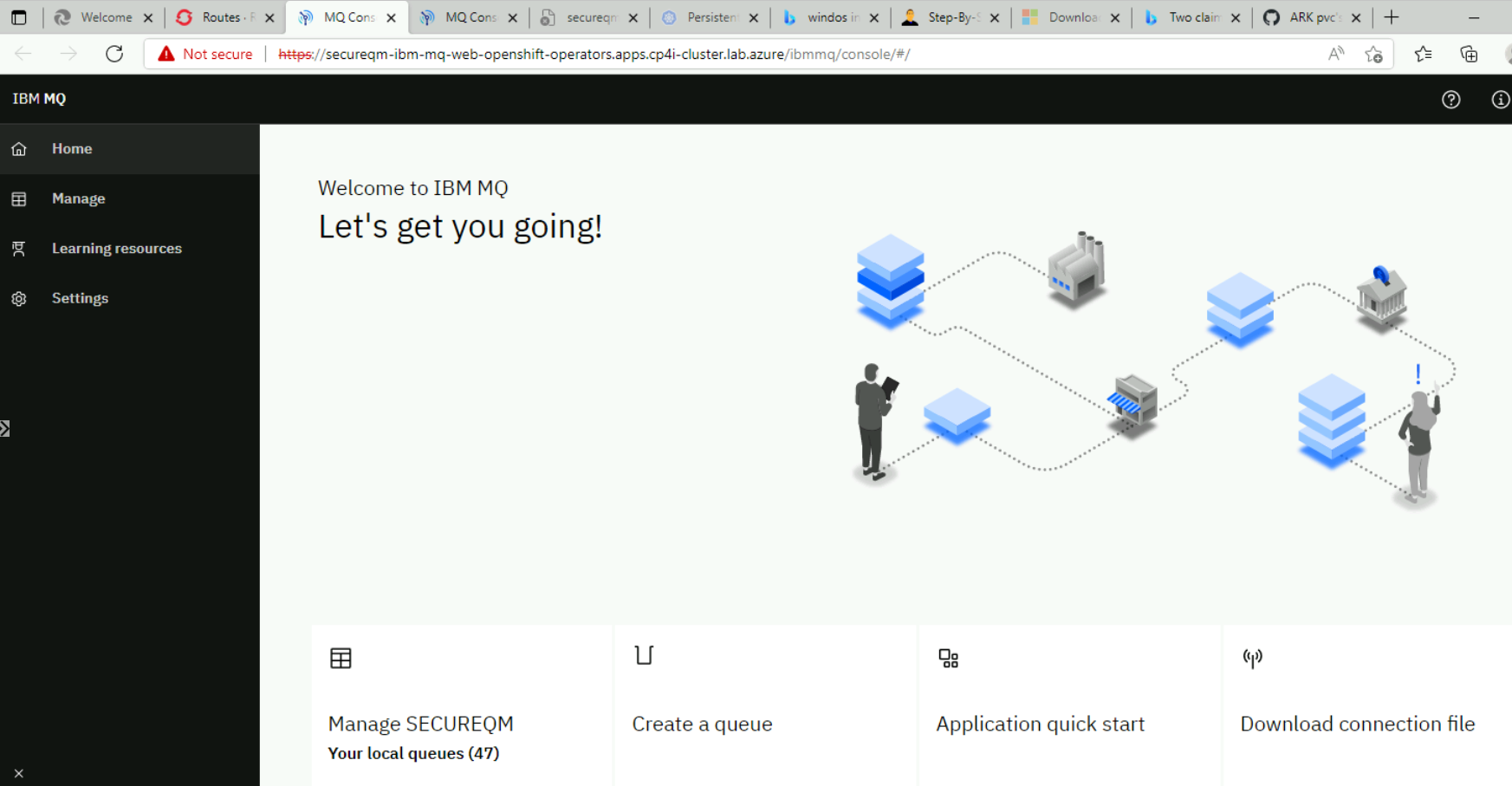
Para la prueba dejaremos dos mensajes “Hola mundo” y “mensaje2 probando”

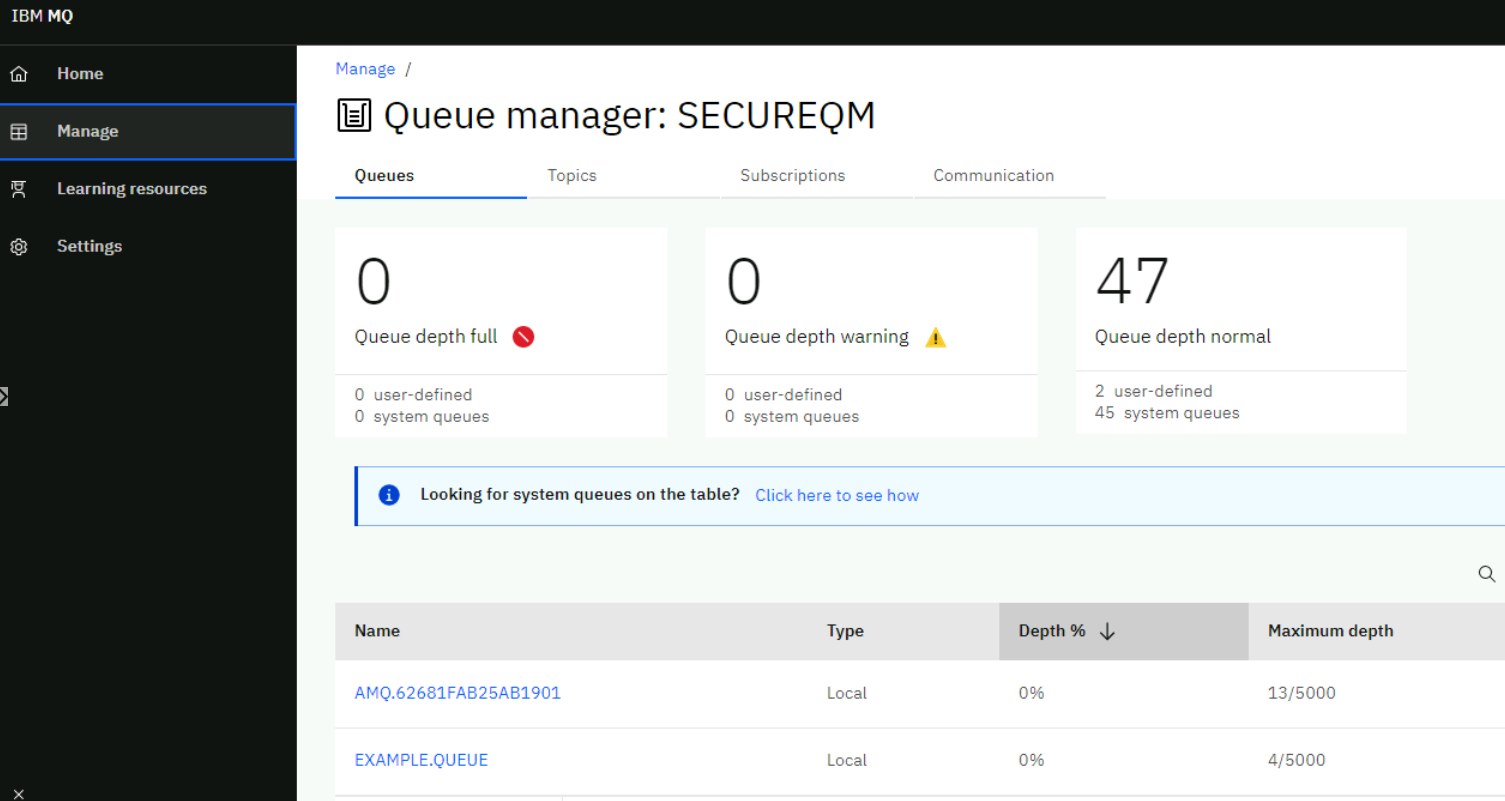


1. Para ingresar al Modulo web nos dirigimos a la consola de Openshift > networking > Routes y hacemos click en el location del queue manager previamente creado

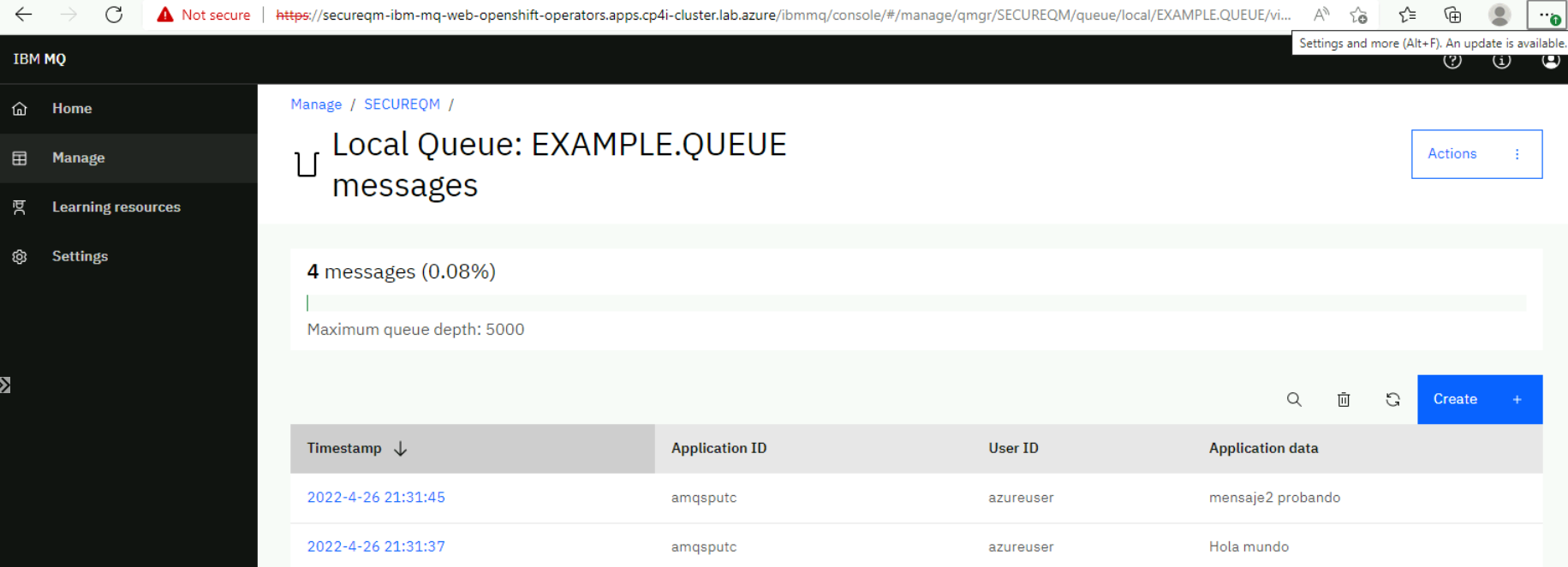


1. Modulo web



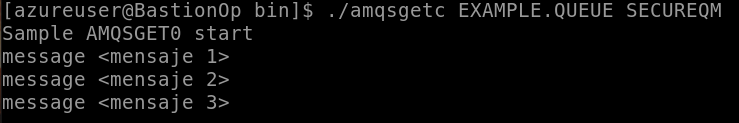


1. En manage > Nombre de cola creada



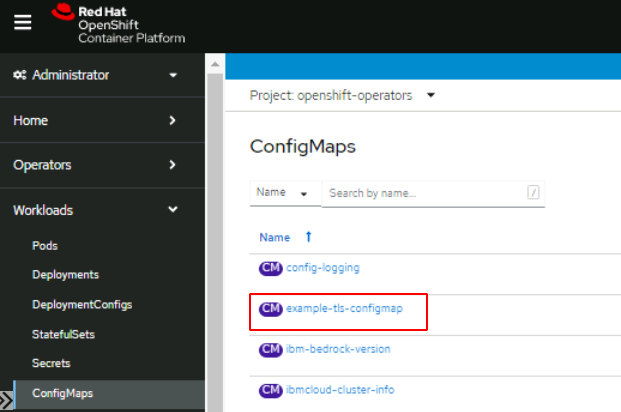
1. Tambien podemos verificar los mensajes en la cola con el comando

|  |
| --- |
| amqsgetc EXAMPLE.QUEUE SECUREQM |



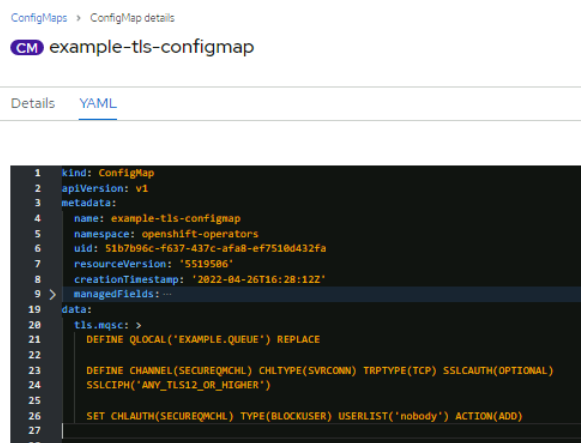
### Adicion de colas o modificacion en configuraciones existentes (Manual)￼

1. Para este ejemplo crearemos una cola Remota y modificaremos el atributo de profundidad de la cola previamente creada. Vamos a la consola de Openshift > Workloads > configmap
2. Seleccionamos el configmap previamente creado ([ver configmap](#_Creacion _config_map))

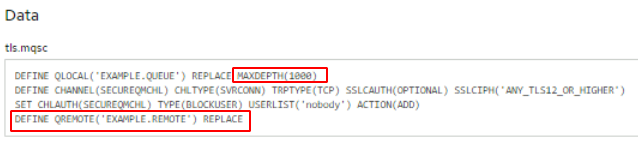


1. Editamos el Yaml y damos en guardar

previo

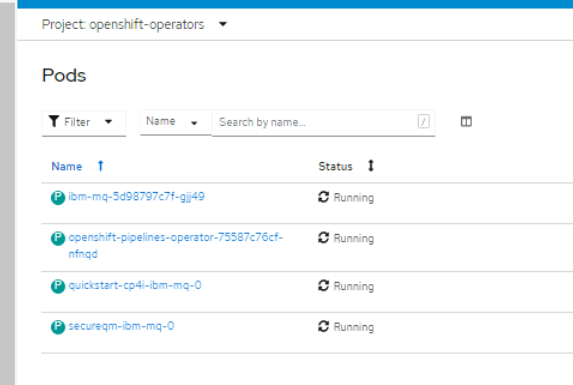


posterior

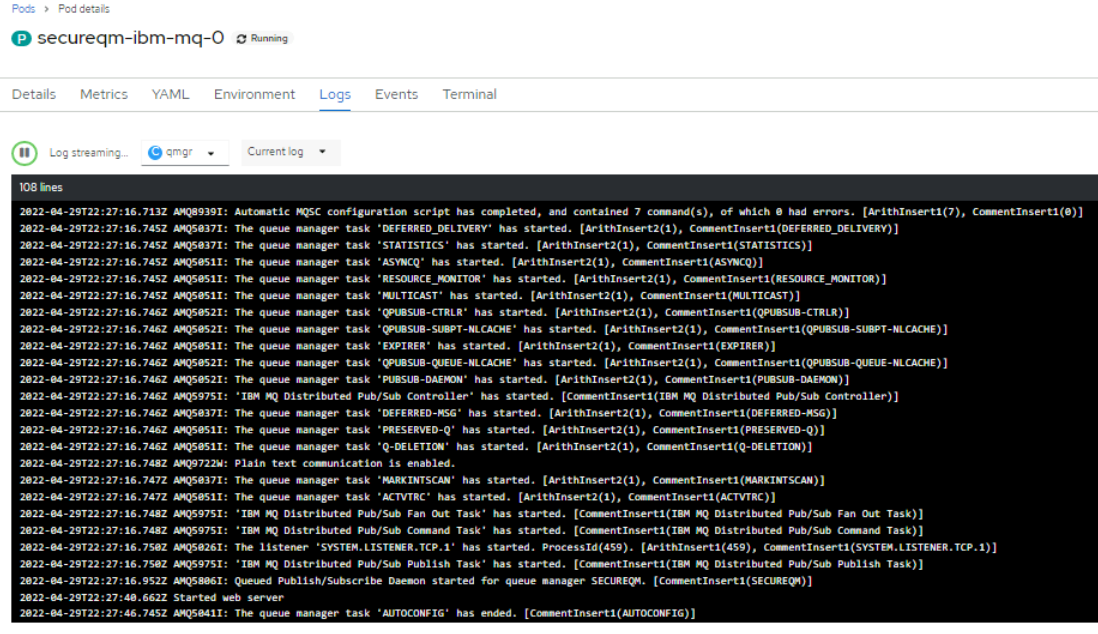


1. Una vez realizados los cambios nos dirigimos a workloads > pods y eliminamos el que corresponda a nuestro queue manager

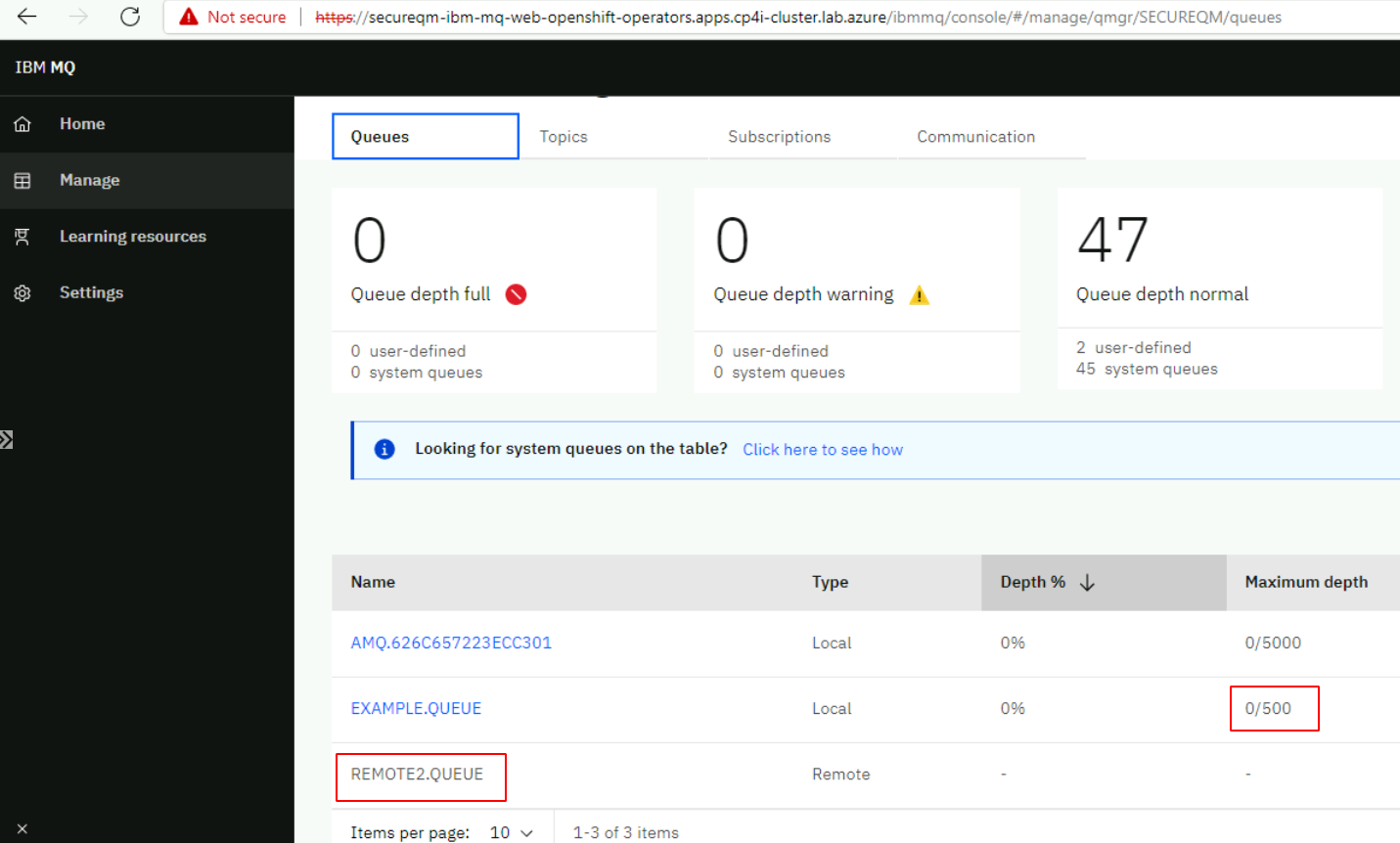
Nota: Toda creación o modificacion realizada en el portal sera borrada, si desea hacer de la misma permanente debe incluirlo en el configmap



Verificar que la subida no presente errores. De ser asi verificar el yaml del configmap donde detalla los comandos MQSC



1. Verificamos en la consola de Queue Manager



## Despliegue Continuo – Tekton/Openshift pipelines￼

Vamos a desplegar la ultima version hasta la fecha de tekton para ello ejecutamos el siguiente comando

|  |
| --- |
| kubectl apply -f https://storage.googleapis.com/tekton-releases/pipeline/previous/v0.35.0/release.yaml |

### Cluster Privado Openshift￼

|  |
| --- |
| oc image mirror gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/webhook:v0.35.0=BastionOp.lab.azure:5000/gcr.io/tekton-releases/github.co  m/tektoncd/pipeline/cmd/webhook:v0.35.0 -a ${REG\_CREDS} --insecure |

|  |
| --- |
| oc image mirror gcr.io/tekton-releases/github.com/tektoncd/pipeline/cmd/controller:v0.35.0=BastionOp.lab.azure:5000/gcr.io/tekton-releases/github  .com/tektoncd/pipeline/cmd/controller:v0.35.0 -a ${REG\_CREDS} --insecure |

## Tekton / Openshift Pipelines

### Construccion de Imagen con cliente OC

Para la ejecucion de las task requerimos una imagen que contenga el cliente oc para poder ejecutar los (oc apply –f [route, configmap, queuemanager].yaml)

Haremos uso de nuestra maquina Bastion

Creamos un directorio llamado docker en nuestro home, seguido creamos un archivo Dockerfile

|  |
| --- |
| mkdir docker  cd docker  vi Dockerfile |

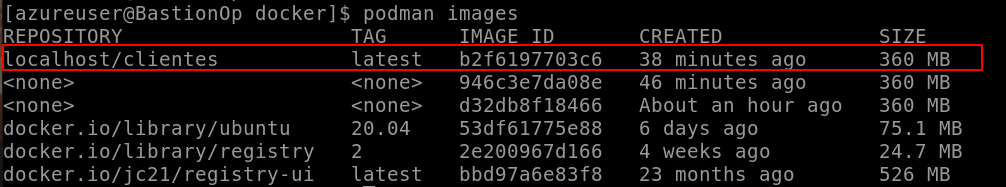


|  |
| --- |
| ##Descargamos una versión concreta de UBUNTU, a través del tag  FROM ubuntu:20.04  MAINTAINER gbm  ##Actualizamos el sistema  RUN apt-get update  ##Instalamos nginx  RUN apt-get install -y nginx  ##Arrancamos NGINX a través de ENTRYPOINT para que no pueda ser modificado en la creación del contenedor  ENTRYPOINT ["/usr/sbin/nginx", "-g", "daemon off;"]  ##Exponemos el Puerto 80  EXPOSE 80  ##inicializacion archivo kubeconfig  RUN mkdir /.kube  RUN > /.kube/config  RUN chmod -R 777 /.kube/config  ##instalacion Curl, telnet, wget, oc  RUN apt-get install -y curl  RUN apt-get install -y telnet  RUN apt-get install -y wget  RUN wget https://mirror.openshift.com/pub/openshift-v4/x86\_64/clients/ocp/stable-4.8/openshift-client-linux.tar.gz  RUN tar -xvf openshift-client-linux.tar.gz -C /bin oc kubectl |

Construimos nuestra imagen basandonos en este Dockerfile

|  |
| --- |
| podman build . -t clientes  podman images |

Nota: asegurese que se encuentra dentro de la carpeta docker



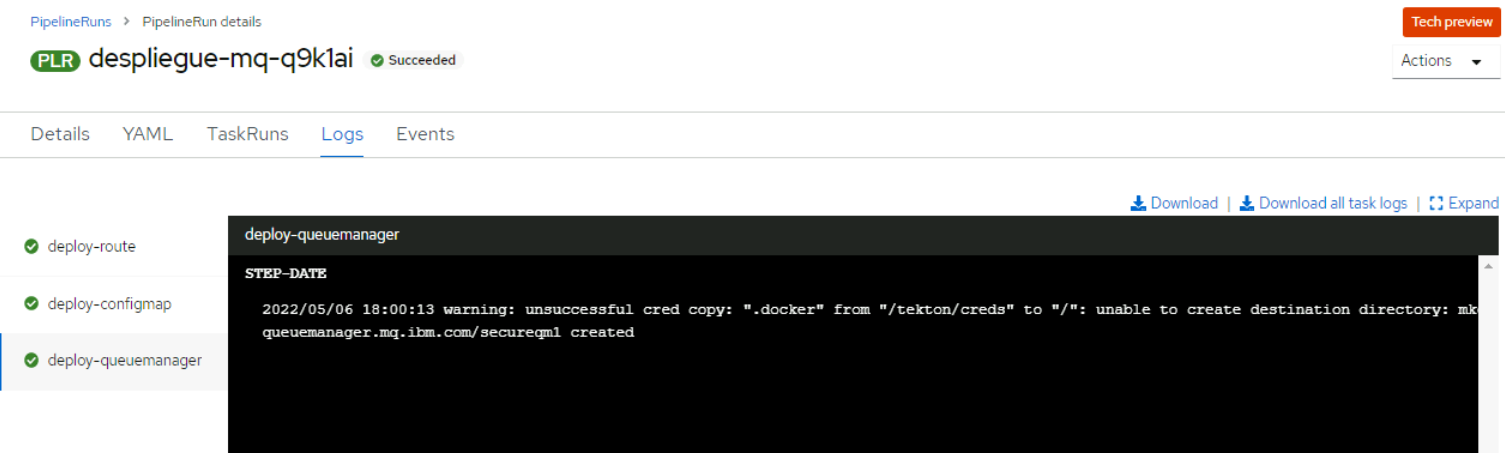
|  |
| --- |
| podman push localhost/clientes BastionOp.lab.azure:5000/clientes/clientes -a ${REG\_CREDS} --insecure |

## Definicion task

|  |
| --- |
| apiVersion: tekton.dev/v1beta1  kind: Task  metadata:  name: deploy-yaml  namespace: tekton-pipelines  spec:  params:  - default: ''  description: yamlfile alocated in the bastion host  name: yamlfile  type: string  steps:  - args:  - '-c'  - oc apply -f $(params.yamlfile)  command:  - /bin/bash  image: 'BastionOp.lab.azure:5000/clientes/clientes'  name: date |

### Definición Pipeline

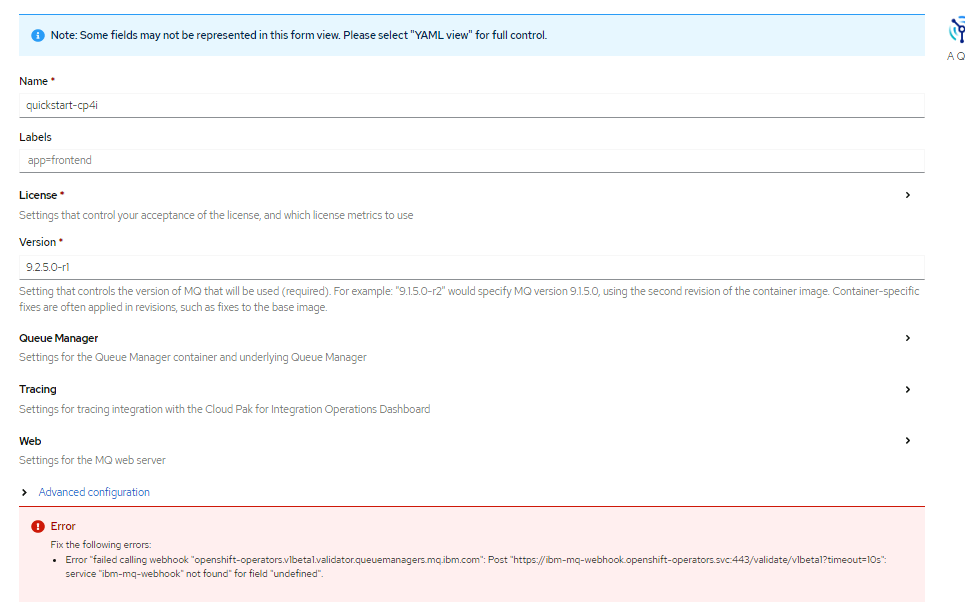
|  |
| --- |
| apiVersion: tekton.dev/v1beta1  kind: Pipeline  metadata:  name: despliegue-mq  namespace: tekton-pipelines  spec:  tasks:  - name: deploy-route  params:  - name: yamlfile  value: 'http://BastionOp.lab.azure:80/IBMMQ/route.yaml'  taskRef:  kind: Task  name: deploy-yaml  - name: deploy-configmap  params:  - name: yamlfile  value: 'http://BastionOp.lab.azure:80/IBMMQ/configmap.yaml'  taskRef:  kind: Task  name: deploy-yaml  - name: deploy-queuemanager  params:  - name: yamlfile  value: 'http://BastionOp.lab.azure:80/IBMMQ/queuemanager.yaml'  runAfter:  - deploy-configmap  - deploy-route  taskRef:  kind: Task  name: deploy-yaml |



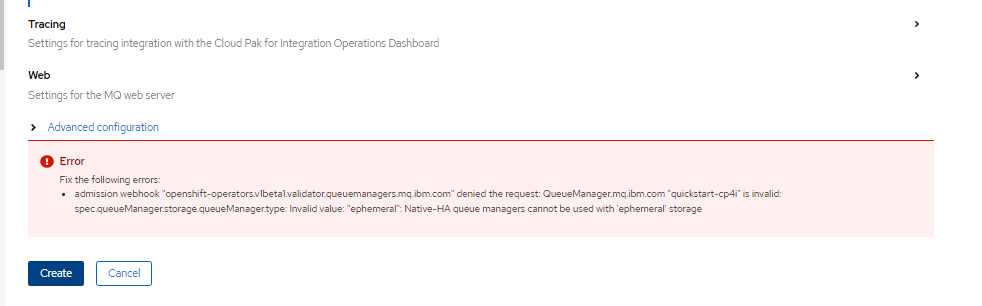
Si llega a presentar algun tipo de errores en el namespace donde se realizo la creacion del Queue Manager dirijase a la seccion de errores (ver aqui)

## Errores￼

### Despliegue de Operador en namespace diferente a Openshift-operators

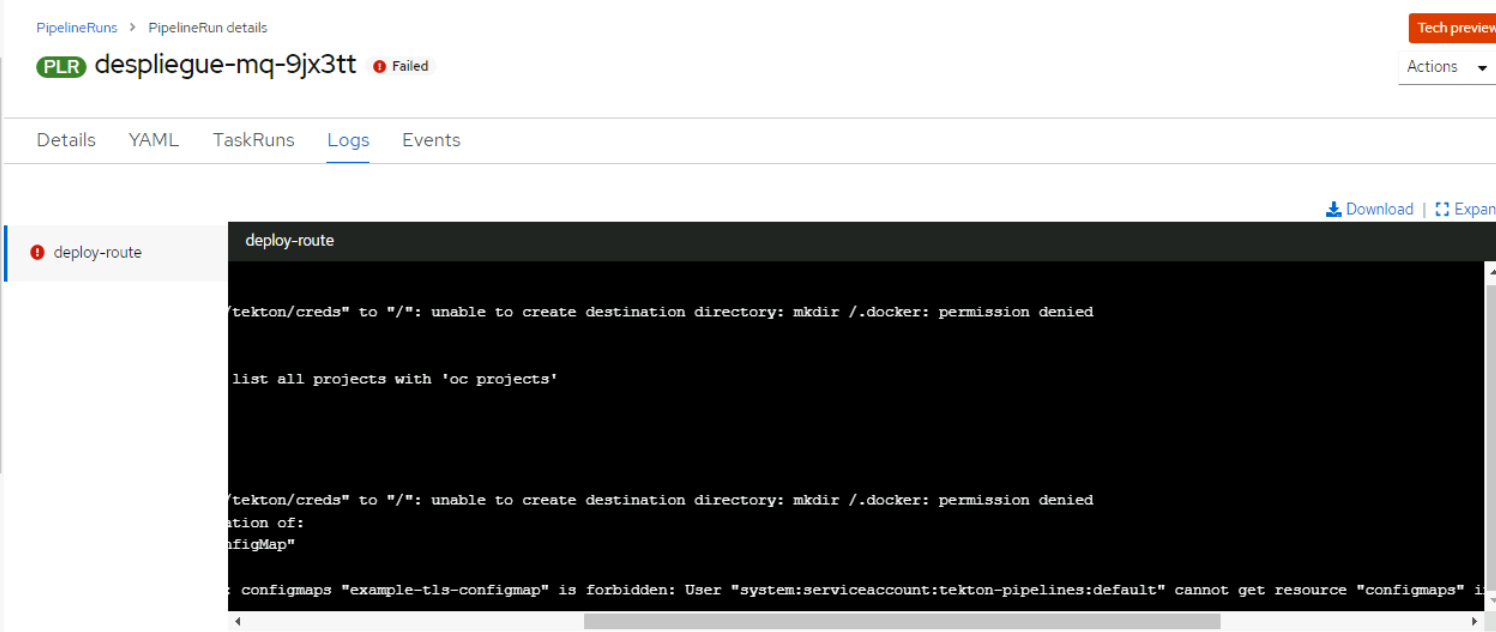


### Ratificar el tipo de storage dependiendo del availability que se elija.



### Ejecución de pipeline con usuario default

|  |
| --- |
| from server for: "http://BastionOp.lab.azure:80/IBMMQ/configmap.yaml": configmaps "example-tls-configmap" is forbidden: User "system:serviceaccount:tekton-pipelines:default" cannot get resource "configmaps" in API group "" in the namespace "openshift-operators" |



Solución:

Dependiendo de donde fueron creados los pipelines debemos dar el rol a la service account definida

|  |
| --- |
| oc adm policy add-role-to-user edit system:serviceaccount:tekton-pipelines:default --rolebinding-name=role-tekton-op-operators -n tekton-pipelines |

## Adicionales

### Expand disk vm azure

https://docs.microsoft.com/en-us/azure/virtual-machines/linux/expand-disks

