

Quick Start guide to IBM Cloud Pak for Integration on RHOS 3.11

Featuring

ACE

ACE+MQ

MQ

APIC

Event Streams

Contents

IBM Cloud Pak for Integration – Target Environment	6
RHOS (OKD) Access point and navigation.....	6
Getting key information from RHOS(OKD)	7
Get the onboard docker image registry.....	7
Get the pull secret.....	9
Get the Network file system	9
ICP4i via ICP Access point and navigation.....	10
Key Information from ICP	11
ICP login information for kubectl and oc commands	11
ICP Catalog	11
ICP4i proxy address.....	12
ICP4i console	13
ACE on ICP4i on RHOS via the ICP4i Navigator	13
Add an ace server and associate a BAR	13
Information you'll need to configure the release.....	16
Configuring the ACE release.....	16
Warnings	16
Configuration parameters.....	18
Observing the ACE deployment via ICP4i	20
Observing the ACE deployment via RHOS	22
ACE Integration Liveliness Probe	25
Github Source Repos for ACE Liveliness Probe (The SoE ACE project)	25
Description	25
Testing Liveliness Probe	26
ACE and ACE/MQ Instances with additional config via secrets	27
Overview	27
List of configurable items.....	27
Running the script to generate the secret	28
ACE on ICP4i on RHOS via Command Line Helm Release	34
Download and install command line tools.....	34
Login, setup and Helm install.....	35
Configuring an MQ Queue Manager deployed with ACE	38
MQ on ICP4i on RHOS via the ICP4i Navigator.....	41
MQ using the ICP4i console	41
Information you'll need to configure the release	41

Configuring the MQ (helm) release	42
MQ on ICP4i via Command line Helm Release	46
Download command line tools	46
Log into the cluster	47
Command line Helm deployment	48
Configuring your Queue Manager for client connection.....	50
Connect via the MQ Web Console	50
Alternative Configuration procedure for a Queue manager	57
Bash into the container and runmqsc.....	57
Testing the Queue Manager	59
Using the RoundTrip GUI	60
ACE+MQ via the ICP4i GUI	62
Modifying the queue manager connection security.....	69
Try connecting MQ Explorer	71
Connecting RFHUTILC to test	73
Using the RoundTrip GUI	75
Using RoundTrip for One-way testing.....	76
Using RoundTrip for RoundTrip testing – no correlID matching	77
Using RoundTrip for RoundTrip testing – correlID matching.....	78
Routing Messages through IDC to EAI	79
Deploying API Connect for IBM Cloud Pak for Integration	80
Prepare Endpoints	81
Management – all endpoints	81
Platform API Endpoint.....	81
Consumer API Endpoint	81
Cloud Admin UI	81
API Manager UI	81
Portal endpoints.....	81
Portal – director	81
Portal – web	81
Analytics endpoints.....	81
Analytics – ingestion	81
Analytics - client	81
Gateway endpoints	81
API Gateway.....	81
Gateway Service.....	81

Obtain the pull secret	81
Create the TLS secret	81
Increase vm.max_map_count.....	82
Storage Class	83
Create an Instance	83
SMTP Server	87
Configuring API Connect	87
Creating an Event Streams Environment.....	89
Perform the End to End IVT test	94
Appendix A – setting up the PSP	104
Appendix B – RoundTrip Batch file	104
Appendix C – MQ and ACE DNS naming	104
Appendix D – ACE GUI – avoid login	105
Appendix E – Useful references	106
Appendix F – connecting in an “Off RHOS” Queue manager.....	106
IDCQM1 – Definitions	106
Sending messages from Off RHOS Qmgr to EAIQM1 via IDCQM1(Gateway).....	107
Receiving messages to Off RHOS Qmgr from EAIQM1 via IDCQM1(Gateway)	111
Listener on off RHOS system.....	111
IPCONFIG on target off box system	111
On premise Firewall rules	111
Secure Gateway on IBM cloud	112
Secure client on premises.....	113
MQ configuration.....	115
Sending a message from RHOS EAIQM1 to “Off RHOS” Queue Manager.....	119
Deploy and Echo flow on EAIQM1 for TEST.IIB.Q1 and TEST.IIB.Q1.REP	121
Appendix G – Building custom MQ image for client access.....	122
Load ICP4i MQAdvanced Server image to local docker repos.....	122
Docker file	123
IVT.mqsc.....	123
Docker build	124
Local test	124
Access via RUNMQSC client	127
Push the image to Dockerhub.....	129
Add new ICP4i instance of MQ using custom image	132
Values for the charts	133

MQ Web Console	140
MQ Explorer	141
Using Runmqsc in client mode.....	143
Appendix H – MQ client Security for non ICP4i containers	145
MQ containers with admin/passw0rd and app/passw0rd as default users.....	145
MQSC that sets up access for client connections.	145
Access via MQ console.....	146
Access via MQ Explorer.....	146
Access via RUNMQSC client.....	147
Runmqsc manually.....	147
Runmqsc piping in a file	147

IBM Cloud Pak for Integration – Target Environment

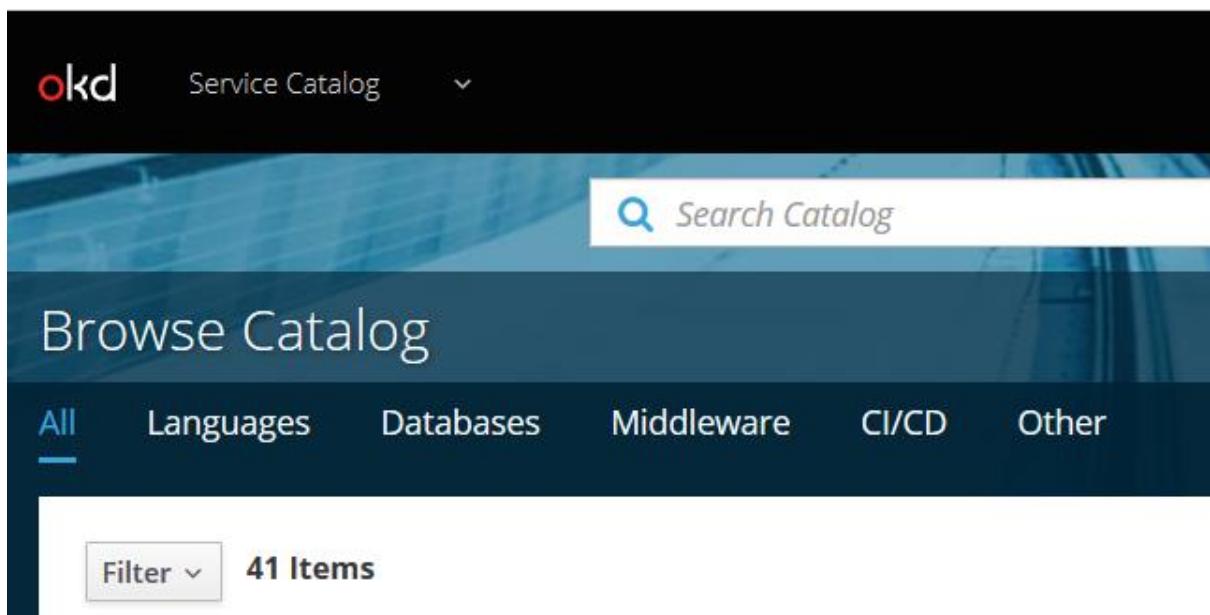
IBM Cloud – IBM ANZ Tech Sales instance – Public IPs

RHOS (OKD) Access point and navigation

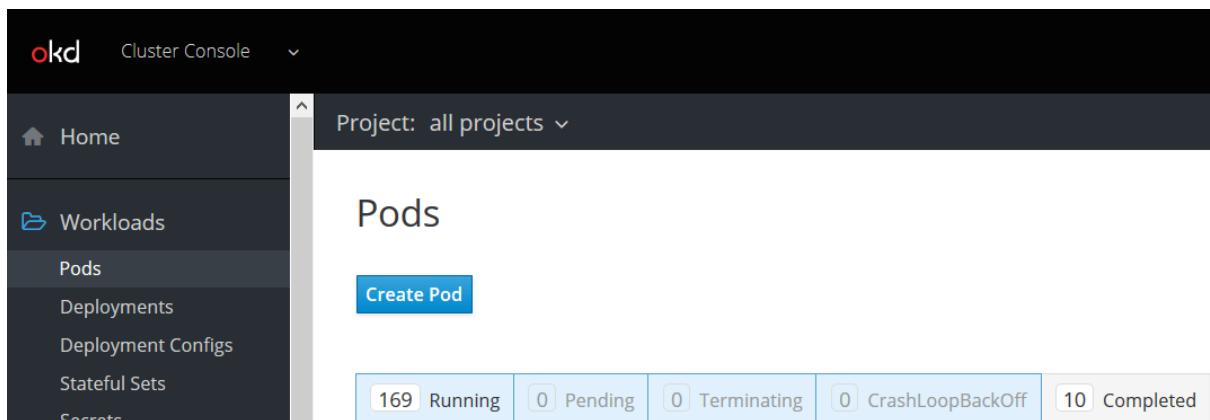
Openshift: <https://master.ocp.cloudnativekube.com:8443/> (admin/#####)



The screenshot shows the OKD login screen. It has a dark background with white text. On the left, there are fields for 'Username' (containing 'admin') and 'Password' (containing several dots). On the right, the text 'Welcome to OKD.' is displayed above a blue 'Log In' button.



The screenshot shows the OKD Service Catalog. At the top, there's a header with the 'okd' logo and a 'Service Catalog' dropdown. Below the header is a search bar with the placeholder 'Search Catalog'. The main area is titled 'Browse Catalog' and contains a horizontal navigation bar with tabs: 'All' (which is selected), 'Languages', 'Databases', 'Middleware', 'CI/CD', and 'Other'. Below the navigation bar, there's a 'Filter' button and a count of '41 Items'.



The screenshot shows the OKD Cluster Console for the 'Pods' section. The left sidebar has a 'Workloads' menu with options: 'Pods' (selected), 'Deployments', 'Deployment Configs', 'Stateful Sets', and 'Secrets'. The main area is titled 'Pods' and features a 'Create Pod' button. Below the button, there are four status boxes: '169 Running', '0 Pending', '0 Terminating', and '0 CrashLoopBackOff'. To the right of these boxes is another status box for 'Completed' with a value of '10'.

Project: ace ▾

Pods

[Create Pod](#)

2 Running	0 Pending	0 Terminating	0 CrashLoopBackOff	0 Completed	0 Failed	0 Unknown
NAME ↑	NAMESPACE	POD LABELS	NODE	STATUS	READINESS	
ace-1-ibm-ace-dashboard-icp4i-prod-6c4cff67d9-8px2p	NS ace	api=ibm-ace-dashboard-i... c... =ibm-ace-dashboard... heritage=Tiller	icp4inode2.ocp.cloudnativekube.com	Running	Ready	

Project: ace ▾

[ace-1-ibm-ace-dashboard-icp4i-prod-6c4cff67d9](#) > Pod Details

P ace-1-ibm-ace-dashboard-icp4i-prod-6c4cff67d9-8px2p

[Overview](#) [YAML](#) [Environment](#) [Logs](#) [Events](#) [Terminal](#)

```

1 kind: Pod
2 apiVersion: v1
3 metadata:
4   generateName: ace-1-ibm-ace-dashboard-icp4i-prod-6c4cff67d9-
5   annotations:
6     openshift.io/scc: ibm-anyuid-scc
7     productID: IBMAppConnectEnterprise_5737_I89_ICP4I_nonChargeable
8     productName: IBM Cloud Pak for Integration - IBM App Connect Enterprise
9     productVersion: 11.0.0.5
10  selfLink: >-
11    /api/v1/namespaces/ace/pods/ace-1-ibm-ace-dashboard-icp4i-prod-6c4cff67d9-8px2p
12

```

Getting key information from RHOS(OKD)

```
C:\openshift>oc login -u system:admin -n default
Authentication required for https://master.ocp.cloudnativekube.com:8443 (openshift)
Username: system:admin
Password:
```

Get the onboard docker image registry

```
C:\openshift>oc get route -n default
NAME          HOST/PORT                               PATH      SERVICES
docker-registry  docker-registry-default.apps.ocp.cloudnativekube.com
registry-console  registry-console-default.apps.ocp.cloudnativekube.com
```

Goto the registry console URL

<https://registry-console-default.apps.ocp.cloudnativekube.com/registry>

ATOMIC REGISTRY

The screenshot shows the main interface of the Atomic Registry. On the left, there's a sidebar with three items: 'Overview' (selected), 'Images', and 'Projects'. The main area has two sections: 'Images by project' and 'Images pushed recently'. In 'Images by project', there's a list of projects: ace, apic, aspera, cert-manager, cp4i, datapower, default, eventstreams, ibmcom, and integration. Each project name is followed by a lock icon. In 'Images pushed recently', there's a list of images with their tags: cp4i/icip-services:2.1.1, cp4i/icip-navigator:2.1.1, apic/k8s-datapower-monitor:2018.4.1-16-37918a2, apic/busybox:1.29-glibc, and apic/datapower-api-gateway:2018.4.1.7-312001-release-prod.

Click on ace

The screenshot shows the details for the 'ace' project. The sidebar shows 'Project: ace' selected. The main area has a table titled 'Images' with columns for 'Name', 'Tags', and 'Repository'. The table lists several images under the 'ace' project:

Name	Tags	Repository
ace/ibm-ace-content-server-prod	11.0.0.5.1-amd64	docker-registry.default.svc:5000/ace/ibm-ace-content-server-prod
ace/ibm-ace-dashboard-prod	11.0.0.5.1-amd64	docker-registry.default.svc:5000/ace/ibm-ace-dashboard-prod
ace/ibm-ace-designer-flows-prod	11.0.0.5.1-amd64	docker-registry.default.svc:5000/ace/ibm-ace-designer-flows-prod
ace/ibm-ace-icp-configuration-prod	11.0.0.5.1-amd64	docker-registry.default.svc:5000/ace/ibm-ace-icp-configuration-prod
ace/ibm-ace-mq-server-prod	11.0.0.5.1-amd64	docker-registry.default.svc:5000/ace/ibm-ace-mq-server-prod
ace/ibm-ace-mqclient-server-prod	11.0.0.5.1-amd64	docker-registry.default.svc:5000/ace/ibm-ace-mqclient-server-prod
ace/ibm-ace-server-prod	11.0.0.5.1-amd64	docker-registry.default.svc:5000/ace/ibm-ace-server-prod

For example this is the image in the onboard registry for an ace server (no MQ client no MQ server)

`docker-registry.default.svc:5000/ace/ibm-ace-server-prod`

Get the pull secret

NAME	TYPE	DATA	AGE
abace-ibm-ace-server-icp4i-prod-handle-384cc241-dockercfg-qxhzn	kubernetes.io/dockercfg	1	11m
abace-ibm-ace-server-icp4i-prod-handle-sec-384cc241-token-hrkbl	kubernetes.io/service-account-token	4	11m
abace-ibm-ace-server-icp4i-prod-handle-secret-serviceaccou4bx4f	kubernetes.io/service-account-token	4	11m
ace-1-ibm-ace-dashboard-icp4i-prod	Opaque	10	1d
ace-1-ibm-ace-dashboard-icp4i-prod-oidc	Opaque	2	1d
ace-1-ibm-ace-dashboard-icp4i-prod-ser-b89c56e3-dockercfg-5fpdj	kubernetes.io/dockercfg	1	1d
ace-1-ibm-ace-dashboard-icp4i-prod-serviceaccount-token-m6k9k	kubernetes.io/service-account-token	4	1d
ace-1-ibm-ace-dashboard-icp4i-prod-serviceaccount-token-sblgc	kubernetes.io/service-account-token	4	1d
builder-dockercfg-fz6xk	kubernetes.io/dockercfg	1	2d
builder-token-md651	kubernetes.io/service-account-token	4	2d
builder-token-s6ld9	kubernetes.io/service-account-token	4	2d
da-ace-echo-ibm-ace--623f-handle--6389-dockercfg-xbk6c	kubernetes.io/dockercfg	1	1h
da-ace-echo-ibm-ace--623f-handle--6389-token-4b9t6	kubernetes.io/service-account-token	4	1h
da-ace-echo-ibm-ace--623f-handle--6389-token-d9b6q	kubernetes.io/dockercfg	1	56m
daace-ibm-ace-server-icp4i-prod-handle-f5f55a05-dockercfg-9bfb4	kubernetes.io/service-account-token	4	56m
daace-ibm-ace-server-icp4i-prod-handle-sec-f5f55a05-token-jh7rn	kubernetes.io/service-account-token	4	56m
daace-ibm-ace-server-icp4i-prod-handle-secret-serviceaccouzb764	kubernetes.io/dockercfg	1	2d
default-dockercfg-8lm7v	kubernetes.io/service-account-token	4	2d
default-token-79zd2	kubernetes.io/service-account-token	4	2d
default-token-z1xdk	kubernetes.io/dockercfg	1	2d
deployer-dockercfg-k8tdp	kubernetes.io/service-account-token	4	2d
deployer-token-gcwbp	kubernetes.io/dockercfg	1	2d
deployer-token-zh6kj	kubernetes.io/service-account-token	4	2d

Get the Network file system

C:\SHARE\ProductsTechnology\IBM Cloud Pak for Integration\SetUp\yamls>\openshift\oc get sc	NAME	PROVISIONER	AGE
managed-nfs-storage (default)	nfs-storage	2d	

ICP4i via ICP Access point and navigation

ICP on Openshift: <https://icp-console.apps.ocp.cloudnativekube.com> (admin/#####)

Fast. Flexible.
Intelligent. Open.
Enterprise-grade.

Log in to your account

Username

admin

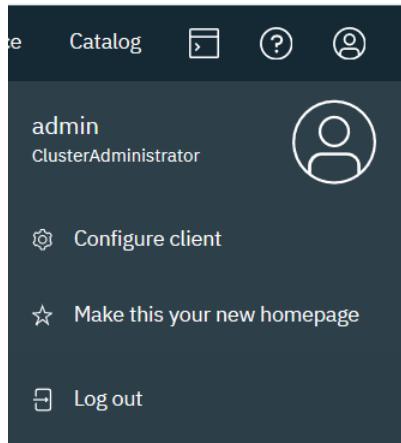
Password

*****|

Log in

Key Information from ICP

ICP login information for kubectl and oc commands



Configure client

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

Prerequisites:

[Install CLI tools](#)

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

```
set-cluster mycluster --server=https://console.apps.ocp.cloudnativekube.com:443
set-context mycluster-context --cluster=mycluster
set-credentials admin --token=cXIEjHAW15TlTbOdAlAomj8pGSEMLp0guM4FqQR4chUc
set-context mycluster-context --user=admin --namespace=default
use-context mycluster-context
```

The token can be used on the oc command line interface

```
C:\openshift>oc login https://master.ocp.cloudnativekube.com:8443 --token=cXIEjHAW15TlTbOdAlAomj8pGSEMLp0guM4FqQR4chUc
Logged into "https://master.ocp.cloudnativekube.com:8443" as "admin" using the token provided.

You have access to the following projects and can switch between them with 'oc project <projectname>':

* ace
  apic
  aspera
  cert-manager
  cp4i
  datapower
  default
  eventstreams
  ibmcom
  integration
```

ICP Catalog

<https://icp-console.apps.ocp.cloudnativekube.com/catalog/>

IBM Cloud Private CLUSTER mycluster

Catalog

Search the catalog...

All Categories >

Classification ▾ Cloud Platform ▾ Architecture ▾

AI & Watson

Blockchain

Business Automation

Data

Data Science & Analytics

DevOps

Integration

IoT

Network

Operations

Runtimes & Frameworks

Cloud Paks

ibm-icp4i-prod
A Helm chart for the IBM Cloud Pak for Integration Navigator

Helm Charts

3scale-gateway
service
3scale API Gateway

amp-apicast-
service
No description provided.

ICP4i proxy address

IBM Cloud Private CLUSTER mycluster

ConfigMaps / ibmcloud-cluster-info

ibmcloud-cluster-info

Overview

Type	Detail
Name	ibmcloud-cluster-info
Namespace	kube-public
Created	2 days ago
Data	cluster_address: icp-console.apps.ocp.cloudnativekube.com cluster_ca_domain: icp-console.apps.ocp.cloudnativekube.com cluster_kube_apiserver_host: console.apps.ocp.cloudnativekube.com cluster_kube_apiserver_port: 443 cluster_name: mycluster cluster_router_http_port: 8080 cluster_router_https_port: 443 edition: Enterprise Edition proxy_address: icp-proxy.apps.ocp.cloudnativekube.com proxy_ingress_http_port: 3080 proxy_ingress_https_port: 3443 version: 3.2.0.1907

ICP4i console

<https://icp-proxy.apps.ocp.cloudnativekube.com:3443/integration>

The screenshot shows the IBM Cloud Pak for Integration console interface. At the top, there's a navigation bar with the title 'IBM Cloud Pak for Integration' and a search bar labeled 'Find'. Below the navigation bar, there are several service tiles:

- API Connect**: Shows a status message 'Unlock business data and assets as APIs' and a note 'You haven't created any instances yet'. Includes a 'Add new instance' button.
- App Connect**: Shows an instance named 'ace' and another named 'ace-1'. Includes a 'Add new instance' button.
- MQ**: Shows an instance named 'mq' and another named 'mqdemo1'. Includes a 'Add new instance' button.
- Event Streams**: Shows a status message 'Deliver messages reliably with enterprise-grade messaging' and a note 'You haven't created any instances yet'. Includes a 'Add new instance' button.
- Aspera**: Shows a status message 'Transfer, exchange and deliver big data at maximum speed'.
- DataPower**: Shows a status message 'Control access to vital resources wherever they are'.

Click on ace-1 – for the ICP4i ACE dashboard

ACE on ICP4i on RHOS via the ICP4i Navigator

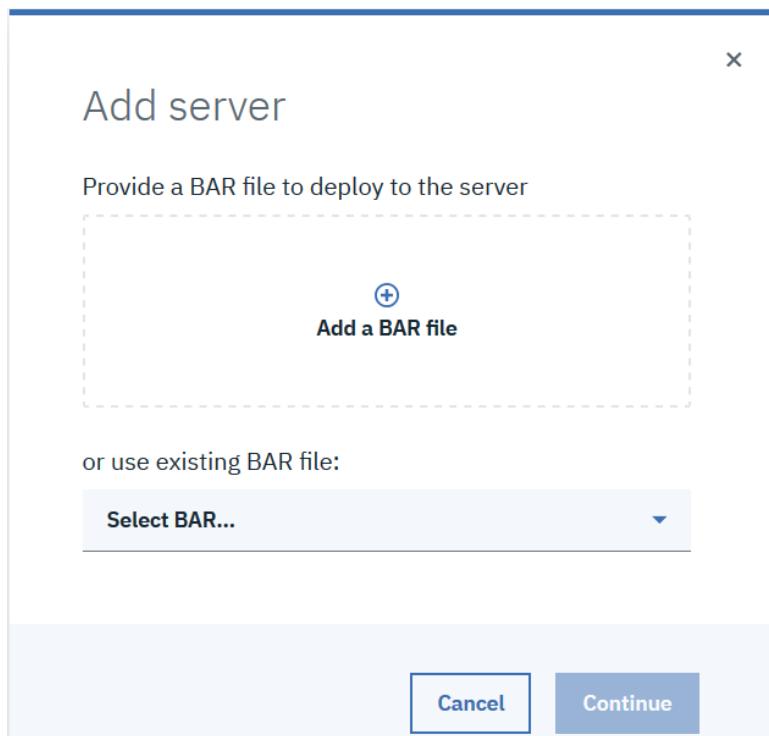
Add an ace server and associate a BAR

The screenshot shows the ICP4i Navigator interface with the 'Servers' tab selected. At the top, there's a header with a search bar labeled 'Search' and a 'Add server +' button. Below the header, there's a toolbar with various icons. The main area displays a table of servers:

Name	Type	Status
ace-1	ACE	Active

At the bottom of the page, there are two tabs: 'Integrations' and 'Servers', with 'Servers' being the active tab. A status message at the bottom right indicates 'Last refreshed: 10/20/2019, 7:32:23 PM Refresh'.

Add a server



Navigate to a bar

<https://github.com/DAVEXACOM/ACEonICPIntSupportingMaterial/tree/master/ace-livelinessProbe>

ACEonICPIntMicSoE > ACEonICPIntMicSoE-master > BARfiles		
<input type="checkbox"/> Name	Date modified	
.project	20/10/2019 5:39 PM	
SoE.bar	20/10/2019 5:39 PM	

Add server

Provide a BAR file to deploy to the server



or use existing BAR file:

Select BAR...

Cancel

Continue

Add server

You will now configure and install a Helm release to deploy to the server. It is important to copy the Content URL and select the current Namespace.

Content URL:

<https://ace-1-ibm-ace-dashboard-icp4i-prod:3443/v1/directories/SoE?23e24c6c-c709-4b5e-a7ec-0dd05b96394f> 

Namespace:

ace

If the integration server requires any configuration to be applied then you will need to use the following download to provide the configuration prior to install. Refer to the README.md inside the download on how to create the required secrets:

Cancel

Configure release 

Take a copy of the url to the BAR (content) file

<https://ace-1-ibm-ace-dashboard-icp4i-prod:3443/v1/directories/SoE?23e24c6c-c709-4b5e-a7ec-0dd05b96394f>

The screenshot shows the IBM Cloud Private interface for managing a Kubernetes cluster named 'mycluster'. The main navigation bar includes 'Create resource', 'Catalog', and a help icon. On the left, there's a sidebar with 'View All' and a link to the 'ibm-ace-server-icp4i-prod' release. The main content area is titled 'ibm-ace-server-icp4i-prod V 2.1.0' and contains tabs for 'Overview' (selected) and 'Configuration'. The 'Overview' tab displays information about the IBM Certified Container, including its helm chart usage by the IBM App Connect Dashboard, and links to 'ibm-entitled-charts', 'Licenses', 'Release Notes', and 'Qualification'. The 'Configuration' tab shows details like Type (IBM Certified Container), Version (2.1.0), and Published date (September 26, 2019). A large central section is titled 'IBM APP CONNECT ENTERPRISE' with a yellow logo, followed by an 'Introduction' section describing the product's purpose and benefits. A blue 'Configure' button is located at the bottom right of this section.

Information you'll need to configure the release

Collected information you need to configure the release.

ICP4i Proxy address (from ICP configMaps): icp-proxy.apps.ocp.cloudnativekube.com

Content URL (from ICP4i): <https://ace-1-ibm-ace-dashboard-icp4i-prod:3443/v1/directories/SoE?23e24c6c-c709-4b5e-a7ec-0dd05b96394f>

onboard docker image registry (from oc command):

example for ace with MQ server

`docker-registry.default.svc:5000/ace/ibm-ace-mq-server-prod`

`docker-registry.default.svc:5000/ace/ibm-ace-mq-server-prod:11.0.0.5.1`

example for ace with no MQ

`docker-registry.default.svc:5000/ace/ibm-ace-server-prod`

`docker-registry.default.svc:5000/ace/ibm-ace-server-prod:11.0.0.5.1`

replace `cp.icr.io` with `docker-registry.default.svc:5000/ace` in the configuration

pull secret(from oc command):

`deployer-dockercfg-k8tdp`

Managed NFS name (from oc command):

`managed-nfs-storage`

Configuring the ACE release

Warnings

These are not a problem – see appendix on PSP



Pod Security Conflict This chart requires a namespace with a ibm-anyuid-psp pod security policy.



Pod Security Warning Your ICP cluster is running all namespaces Unrestricted (ibm-anyuid-hostpath-psp) by default. This could pose a security risk.

Configuration parameters

Configuration

This helm chart is used by the IBM App Connect Dashboard to deploy servers. Edit these parameters for configuration.

Helm release name *

my-ace-rel

Target namespace *

ace

Target Cluster *

local-cluster

License *i

I have read and agreed to the [License agreement](#)

Parameters

To install this chart, additional configuration is needed in Quick start. To customize installation, view and edit All parameters.

Quick start

Required and recommended parameters to view and edit.

Content Server URL *

https://ace-1-ibm-ace-dashboard-icp4i-prod:3443/v1/directories/SoE?23e24c6c-c709-4b5e-a7ec-0dd05b96394f

Service

Service settings

Proxy Node IP or FQDN *i

icp-proxy.apps.ocp.cloudnativekube.com

Production usage

Which type of image to run

App Connect Enterprise only

Architecture scheduling preference *

amd64

Enable IBM App Connect Designer flows

Images

Define images to be used

Docker image for App Connect Enterprise *i

docker-registry.default.svc:5000/ace/ibm-ace-server-prod:11.0.0.5.1

Docker image for App Connect Enterprise with MQ client *

cp.icr.io/ibm-ace-mqclient-server-prod:11.0.0.5.1

Docker image for App Connect Enterprise with MQ server *

cp.icr.io/ibm-ace-mq-server-prod:11.0.0.5.1

Configurator Docker image *

cp.icr.io/ibm-ace-icp-configurator-prod:11.0.0.5.1

Image pull secret

deployer-dockercfg-k8tdp|

Service
Service settings

Service type *	Web UI Port *
NodePort	7600
HTTP port *	HTTPS port *
7800	7843
Switch AgentC Port	Switch AgentP Port
Enter value	Enter value
Switch Admin Port	Proxy Node IP or FQDN *
Enter value	icp-proxy.apps.ocp.cloudnativekube.com

Integration Server

Define configuration for the Integration Server

Integration server name

myintsrv|

Data persistent volume claims (PVCs)
Settings for the PVCs (applicable only when running with a queue manager)

Name *	Storage class name
data	managed-nfs-storage

Readiness probe

Settings for the readiness probe that checks if the integration server is ready

Initial delay (seconds) i

30

Install

Might have to wait a little while – then goto helm releases

Observing the ACE deployment via ICP4i

The screenshot shows the IBM Cloud Private interface. At the top, there is a blue header bar with a white 'X' icon on the left and the text 'IBM Cloud Private' in white on the right. Below this, the main content area has a light gray background. It features a navigation menu on the left with the following items: 'Overview' (bolded), 'Workloads' (with a dropdown arrow), 'Brokered Services', 'DaemonSets', 'Deployments', and 'Helm Releases'. The 'Workloads' item is currently expanded, showing these four sub-options.

Helm Releases

! You are currently viewing only the helm releases of this cluster.

Name	Namespace	Status	Chart name	Current version	Available version	Updated	Actions
my-ace-rel	ace	● Deployed	ibm-ace-server-icp4i-prod	2.1.0	Up To Date	October 20, 2019 08:06pm	Launch ▾

06pm

[Launch](#) ▾

webui

IBM App Connect

Server: myintsrv



Contents

! Search

⋮

LivelinessProbe

API

● Started

Server: myintsrv / API: LivelinessProbe

LivelinessProbe

[Documentation](#)

REST API Base URL

<http://icp-proxy.apps.ocp.cloudnativekube.com:30792/livelinessProbe/v1>

Observing the ACE deployment via RHOS

The screenshot shows the Red Hat OpenShift Cluster Console interface. The top navigation bar includes the 'okd' logo, 'Cluster Console' dropdown, and 'Project: ace' dropdown. The left sidebar has 'Home', 'Workloads', 'Pods', and 'Deployments' options, with 'Deployments' currently selected. A prominent blue button labeled 'Create Deployment' is visible. The main content area is titled 'Deployments' and displays a single deployment entry for 'my-ace-rel'. The deployment details include:

- Deployment name: my-ace-rel
- Namespace: ace
- Labels:
 - app=ibm-ace-server-icp4i-prod
 - chart=ibm-ace-server-icp4i-prod
 - heritage=Tiller
 - release=my-ace-rel
- Status: 3 of 3 pods
- Annotations:
 - app=ibm-ace-server-icp4i-prod, chart=ibm-ace-server-icp4i-prod, heritage=Tiller, release=my-ace-rel

Project: ace ▾

D my-ace-rel-ibm-ace-server-icp4i-prod

Overview YAML Pods Environment Events

Deployment Overview

DESIRED COUNT 3 pods >	UP-TO-DATE COUNT 3 pods	MATCHING PODS 3 pods	{ 3 available 0 unavailable
---------------------------	----------------------------	-------------------------	--------------------------------

NAME: my-ace-rel-ibm-ace-server-icp4i-prod UPDATE STRATEGY: RollingUpdate

NAMESPACE: NS ace MAX UNAVAILABLE: 1 of 3 pods

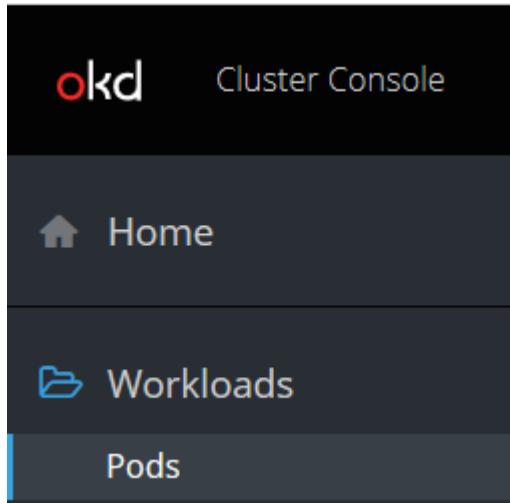
LABELS: app=ibm-ace-server-icp4i-prod, chart=ibm-ace-server-icp4i-prod, heritage=Tiller, release=my-ace-rel MAX SURGE: 1 greater than 3 pods

POD SELECTOR: Q app=ibm-ace-server-icp4i-prod, chart=ibm-ace-server-icp4i-prod, heritage=Tiller, release=my-ace-rel PROGRESS DEADLINE: 10m 0s

MIN READY SECONDS: Not Configured

Containers

NAME	IMAGE	RESOURCE LIMITS	PORTS
my-ace-rel-ibm-ace-server-icp4i-prod	docker-registry.default.svc:5000/ace/ibm-ace-	cpu: 1, memory: 1Gi	7600/TCP, 7800/TCP, 7843/TCP



Project: ace ▾

[my-ace-rel-ibm-ace-server-icp4i-prod-66cf9c496c](#) > Pod Details

P my-ace-rel-ibm-ace-server-icp4i-prod-66cf9c496c-gzwll

Overview	YAML	Environment	Logs	Events	Terminal
142 ▾	<pre> 143 ports: 144 - name: webui 145 containerPort: 7600 146 protocol: TCP 147 - name: ace-http 148 containerPort: 7800 149 protocol: TCP 150 - name: ace-https 151 containerPort: 7843 152 protocol: TCP 153 imagePullPolicy: IfNotPresent 154 volumeMounts: 155 - name: webusers 156 mountPath: /home/aceuser/initial-config/webusers 157 - name: odbcini 158 mountPath: /home/aceuser/initial-config/odbcini 159 - name: policy 160 mountPath: /home/aceuser/initial-config/policy 161 - name: serverconf 162 mountPath: /home/aceuser/initial-config/serverconf 163 - name: setdbparms 164 mountPath: /home/aceuser/initial-config/setdbparms </pre>				

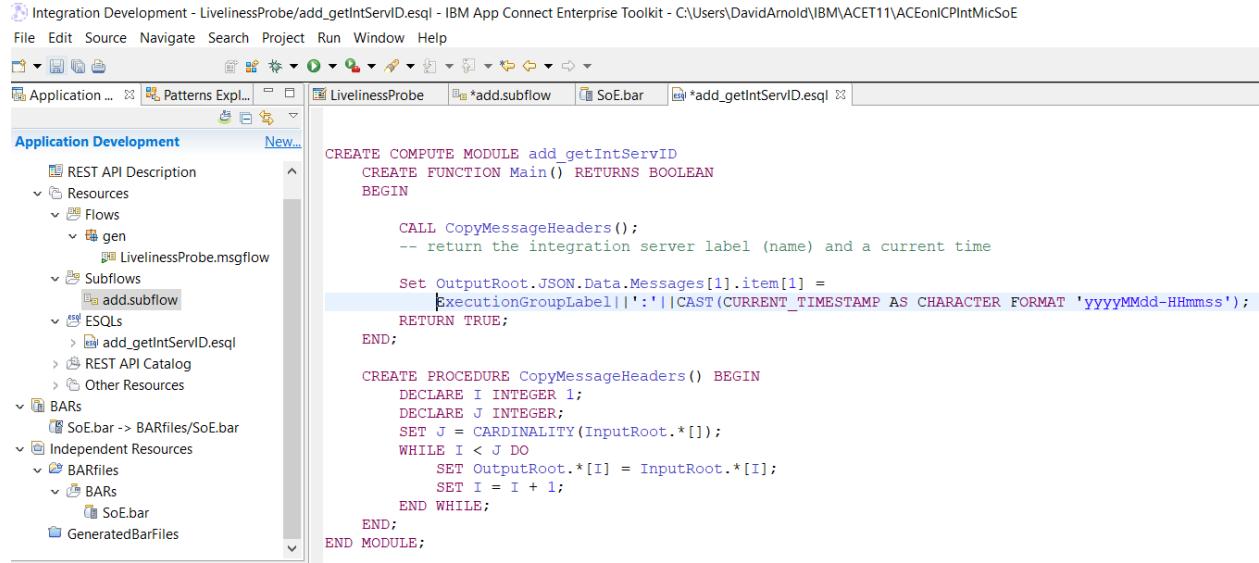
ACE Integration Liveliness Probe

Github Source Repos for ACE Liveliness Probe (The SoE ACE project)

<https://github.com/DAVEXACOM/ACEonICPIntMicSoE>

Description

The Liveliness Probe Service is an other restful service (this is not the service baked into the cloud pak images). Its just a user echo test service



```
CREATE COMPUTE MODULE add_getIntServID
  CREATE FUNCTION Main() RETURNS BOOLEAN
  BEGIN

    CALL CopyMessageHeaders();
    -- return the integration server label (name) and a current time

    Set OutputRoot.JSON.Data.Messages[1] =
      ExecutionGroupLabel||':'||CAST(CURRENT_TIMESTAMP AS CHARACTER FORMAT 'yyyyMMdd-HHmmss');
    RETURN TRUE;
  END;

  CREATE PROCEDURE CopyMessageHeaders() BEGIN
    DECLARE I INTEGER 1;
    DECLARE J INTEGER;
    SET J = CARDINALITY(InputRoot.*[]);
    WHILE I < J DO
      SET OutputRoot.*[I] = InputRoot.*[I];
      SET I = I + 1;
    END WHILE;
  END;
END MODULE;
```

Testing Liveliness Probe

Using RESTED rest client for firefox I this example

Base URI

<http://icp-proxy.apps.ocp.cloudnativekube.com:30792/livelinessProbe/v1>

input data:

{"Messages": ["test"]}

Return data

returns the integration server name plus a current timestamp for

Post

<http://icp-proxy.apps.ocp.cloudnativekube.com:30792/livelinessProbe/v1/message>

The screenshot shows a REST client interface with the following details:

- Request:** POST <http://icp-proxy.apps.ocp.cloudnativekube.com:30792/livelinessProbe/v1/message>
- Headers:** (dropdown menu)
- Basic auth:** (dropdown menu)
- Request body:** Type: Custom
Content:

```
{"Messages": ["test"]}
```
- Send request** button

Response (0.741s) - http://icp-proxy.apps.ocp.cloudnativekube.com:30792/livelinessProbe/v1/message

200 OK

Headers: (dropdown menu)

Content:

```
{"Messages": {"item": "myintsrv:20191020-094445"}}
```

ACE and ACE/MQ Instances with additional config via secrets

Overview

It is possible to create additional configuration items and push these into a secret which the ACE deployment will introspect and use for configuration when the container is created. In this way MQSC scripts and many ACE configuration tasks can be executed automatically when the helm release is deployed.

List of configurable items

The items listed below are configurable via a secret:

File	Description
adminPassword.txt	MQ Developer defaults - administrator password
appPassword.txt	MQ Developer defaults - app password
mqsc.txt	An mqsc file to run against the Queue Manager
keystorePassword	A password to set for the Integration Server's keystore
keystore-{keyname}.pass	The passphrase for the private key being imported, if there is one
keystore-{keyname}.key	The private key in PEM format
keystore-{keyname}.crt	The certificate in PEM format
truststorePassword.txt	A password to set for the Integration Server's truststore
truststore-{certname}.crt	The trust certificate in PEM format
odbc.ini	An odbc.ini file for the Integration Server to define any ODBC data connections
policy.xml	Policies to apply
policyDescriptor.xml	The policy descriptor file
serverconf.yaml	The server.conf.yaml
setdbparms.txt	Multi-line file containing the {ResourceName} {UserId} {Password} to pass to mqsisetdbparms command
adminusers.txt	Multi-line value containing the {UserName} {Password} to pass to mqsiwebuseradmin command to create viewer users with READ, WRITE, EXECUTE access to the server

File	Description
viewerusers.txt	Multi-line value containing the {UserName} {Password} to pass to mqsiwebuseradmin command to create admin users with READ access to the server

Running the script to generate the secret

To run the script:

- The kubectl command must be installed and available
- The kubectl environment must be configured to point to the intended target cluster
- The name of the configuration secret to be created must be supplied as an argument on the command line
- The name of RELEASE may be provided as a second argument on the command line if admin or viewer users are required

The script will look for any files detailed in the Configuration section in the same directory and if found will add them into keys in the generated secret. The script will then use the kubectl command to generate a secret containing the specified configuration for the integration server and a second secret containing the user credentials if the second argument of RELEASE NAME is specified.

The script is shown below:

```
-----begin snippet-----
#!/bin/bash

#
# Licensed Materials - Property of IBM
#
# 5737-H33
#
# (C) Copyright IBM Corp. 2018 All Rights Reserved.
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

TARGET_SECRET_NAME=$1
#####
# Build up the arguments for the kubectl command
#####
```

```
SECRET_ARGS=

if [ -s ./mqsc.txt ]; then

    SECRET_ARGS="${SECRET_ARGS} --from-file= mqsc=./mqsc.txt"

else

    SECRET_ARGS="${SECRET_ARGS} --from-literal= mqsc="

fi

if [ -s ./adminPassword.txt ]; then

    SECRET_ARGS="${SECRET_ARGS} --from-file= adminPassword=./adminPassword.txt"

else

    SECRET_ARGS="${SECRET_ARGS} --from-literal= adminPassword="

fi

if [ -s ./appPassword.txt ]; then

    SECRET_ARGS="${SECRET_ARGS} --from-file= appPassword=./appPassword.txt"

else

    SECRET_ARGS="${SECRET_ARGS} --from-literal= appPassword="

fi

if [ -s ./keystorePassword.txt ]; then

    SECRET_ARGS="${SECRET_ARGS} --from-file= keystorePassword=./keystorePassword.txt"

else

    SECRET_ARGS="${SECRET_ARGS} --from-literal= keystorePassword="

fi

if [ -s ./keystore-mykey.key ]; then

    SECRET_ARGS="${SECRET_ARGS} --from-file= keystoreKey-mykey=./keystore-mykey.key"

else

    SECRET_ARGS="${SECRET_ARGS} --from-literal= keystoreKey-mykey="

fi

if [ -s ./keystore-mykey.crt ]; then

    SECRET_ARGS="${SECRET_ARGS} --from-file= keystoreCert-mykey=./keystore-mykey.crt"

else

    SECRET_ARGS="${SECRET_ARGS} --from-literal= keystoreCert-mykey="

fi
```

```
if [ -s ./keystore-mykey.pass ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=keystorePass-mykey=./keystore-mykey.pass"
else
    SECRET_ARGS="${SECRET_ARGS} --from-literal=keystorePass-mykey="
fi

if [ -s ./truststorePassword.txt ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=truststorePassword=./truststorePassword.txt"
else
    SECRET_ARGS="${SECRET_ARGS} --from-literal=truststorePassword="
fi

if [ -s ./truststoreCert-mykey.crt ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=truststoreCert-mykey=./truststoreCert-mykey.crt"
else
    SECRET_ARGS="${SECRET_ARGS} --from-literal=truststoreCert-mykey="
fi

if [ -s ./odbc.ini ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=odbcini=./odbc.ini"
else
    SECRET_ARGS="${SECRET_ARGS} --from-literal=odbcini="
fi

if [ -s ./policy.xml ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=policy=./policy.xml "
else
    SECRET_ARGS="${SECRET_ARGS} --from-literal=policy="
fi

if [ -s ./policyDescriptor.xml ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=policyDescriptor=./policyDescriptor.xml "
else
    SECRET_ARGS="${SECRET_ARGS} --from-literal=policyDescriptor="
fi
```

```

if [ -s ./serverconf.yaml ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=serverconf=./serverconf.yaml "
else
    SECRET_ARGS="${SECRET_ARGS} --from-literal=serverconf="
fi

if [ -s ./setdbparms.txt ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=setdbparms=./setdbparms.txt "
else
    SECRET_ARGS="${SECRET_ARGS} --from-literal=setdbparms="
fi

if [ -s ./extensions.zip ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=extensions=./extensions.zip "
fi

if [ -s ./switch.json ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=switch=./switch.json "
fi

if [ -s ./agentx.json ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=agentx=./agentx.json "
fi

if [ -s ./agentp.json ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=agentp=./agentp.json "
fi

if [ -s ./agentc.json ]; then
    SECRET_ARGS="${SECRET_ARGS} --from-file=agentc=./agentc.json "
fi

#####
# Create the Kubernetes secret resource
#####
echo "Creating secret"
echo "kubectl create secret generic ${TARGET_SECRET_NAME}${SECRET_ARGS}"

```

```
oc create secret generic ${TARGET_SECRET_NAME}${SECRET_ARGS} -n ace  
-----end snippet-----
```

To run the script:

1. Connect to the RHOCOP via oc login.

Select Command line tools and follow the instruction here:

With the OpenShift command line interface (CLI), you can create applications and manage OpenShift projects from a terminal. You can download the `oc` client tool using the links below. For more information about downloading and installing it, please refer to the [Get Started with the CLI](#) documentation.

Download `oc` :

[Latest Release](#) ↗

After downloading and installing it, you can start by logging in. You are currently logged into this console as **admin**. If you want to log into the CLI using the same session token:
`oc login https://master.ocp.cloudnativekube.com:8443 --token=<hidden>`

2. Create an empty directory and populate it with the generateSecrets.sh script
3. Create the files described above required for your configuration in the same directory.

As an example, to set dbparms for external resource access to kafka for an integration server, a setdbparms file will contain something similar to:

```
kafka::KAFKA token KsTEutxASFTZtvqqlqQZKRrkFqeU_QqhWPkN8GdJrz95
```

!! NB: Take care not to include additional new lines at the end of the files (such as `keystorePassword.txt`) - some editors add these automatically on save and can result in the containers not starting due to password mismatch.

4. Make sure you're in the ACE namespace using the command : `oc project ace`
5. Run the command as shown, alternatively giving the second argument as the release name=, specifically for setting adminusers and viewusers for the webui (linux/mac syntax shown) :

```
./generateSecrets.sh my-secret [release1]
```

You should see something similar to

```
./generateSecrets.sh ace-secret
```

Creating secret

```
kubectl create secret generic my-secret --from-literal=mqsc= --from-literal=adminPassword= --from-literal=appPassword= --from-literal=keystorePassword= --from-literal=keystoreKey-mykey= --from-literal=keystoreCert-mykey= --from-literal=keystorePass-mykey= --from-file=truststorePassword=./truststorePassword.txt --from-file=truststoreCert-mykey=./truststoreCert-mykey.crt --from-literal=odbcini= --from-literal=policy= --from-literal=policyDescriptor= --from-file=serverconf=./serverconf.yaml --from-file=setdbparms=./setdbparms.txt  
secret/my-secret created
```

6. When deploying the ACE integration server, populate the following highlighted field with the secret name

Integration server name 	List of key aliases for the keystore
IS1	Enter value
List of certificate aliases for the truststore	Name of the default application
Enter value	Enter value
The name of the secret to create or to use that contains the server configuration	
my-secret	File system group ID
	Enter value

Example:

Add an MQSC file to an ACE/MQ Deployment to pre-configure access to the ACE Queue Manager

- Do steps 1-4 above.
- Create an MQSC file called **mqsc.txt** in its own directory

```
DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCONN)

SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)

ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE)
ADOPTCTX(NO)

SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')

define qlocal (RTREP.Q)
define qlocal(RTREQ.Q)

REFRESH SECURITY TYPE(CONNAUTH)
```

- Run the 'generateSecrets.sh ace-secret'

Do step 6 above.

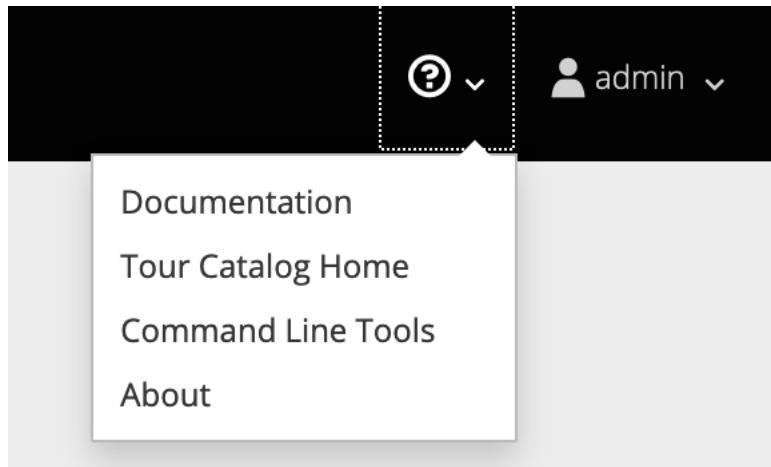
Test access to the queue manager as in the section [Testing the Queue manager](#)

ACE on ICP4i on RHOS via Command Line Helm Release

Download and install command line tools

Go to the registry console URL

<https://registry-console-default.apps.ocp.cloudnativekube.com/registry>



Select Command line tools and follow the instructions here:

Command Line Tools

With the OpenShift command line interface (CLI), you can create applications and manage OpenShift projects from a terminal. You can download the `oc` client tool using the links below. For more information about downloading and installing it, please refer to the [Get Started with the CLI](#) documentation.

[Download `oc`](#) : [Latest Release](#)

After downloading and installing it, you can start by logging in. You are currently logged into this console as **admin**. If you want to log into the CLI using the same session token:
`oc login https://master.ocp.cloudnativekube.com:8443 --token=<hidden>`

A screenshot of a web page section titled "Command Line Tools". It contains instructions for downloading the OpenShift CLI ("oc") tool. It also shows the current user is "admin" and provides a command-line example for logging in using the same session token.

Once logged in:

Switch to the ACE Project:

`oc project ace`

Get the cloudctl version from ICP:

Goto <https://icp-console.apps.ocp.cloudnativekube.com/console/tools/cli>

Install CLI tools

Install IBM Cloud Private CLI v3.2.0-634

You can download the IBM Cloud [cloudctl](#) for macOS, Windows, and Linux. Download the installer with the following curl command, then see [Installing the IBM® Cloud Private CLI](#) to complete your installation:

Select OS

Select

Download with curl

Command will appear in this field after you make a selection above.



Select your OS and download the file/tar or zip file.

Once the executable id identified copy it and rename it to cloudctl/cloudctl.exe

Follow the same process for the helm executable (the versions of the helm client and tiller on the server must match):

Install Helm CLI v2.12.3

Helm CLI (helm) is a tool for managing Kubernetes packages, or Helm charts. Deploy and manage applications, open source tools, and IBM software on your IBM Cloud Private environment. Download the installer with the following curl command, then see [Installing Helm](#) to complete your installation:

Select OS

Select

Download with curl

Command will appear in this field after you make a selection above.



Login, setup and Helm install

Login to the cluster as follows:

```
sudo cloudctl login -a https://icp-console.apps.ocp.cloudnativekube.com -u admin -p ##### -n default
```

Run the following helm command:

```
sudo helm init --client-only --skip-refresh
```

You will see the following in the output of this command:

```
$HELM_HOME has been configured at <your_helm_home>
```

Run the flowing commands:

```
export HELM_HOME=<your_helm_home>
export ICP_URL=https://icp-console.apps.ocp.cloudnativekube.com
```

The expression below points to the location of the repository used for icp4i

```
sudo helm repo add local-charts $ICP_URL/helm-repo/charts --ca-file $HELM_HOME/ca.pem --cert-file $HELM_HOME/cert.pem --key-file $HELM_HOME/key.pem
```

This has prepared you for running helm install against a chart, image(s) and configuration.

The following script installs an ACE image with MQ, asking for command line parameters for helm release name, integration server name and an optional secret. It installs a pre-loaded bar file called SoE.bar provided by the context URL.

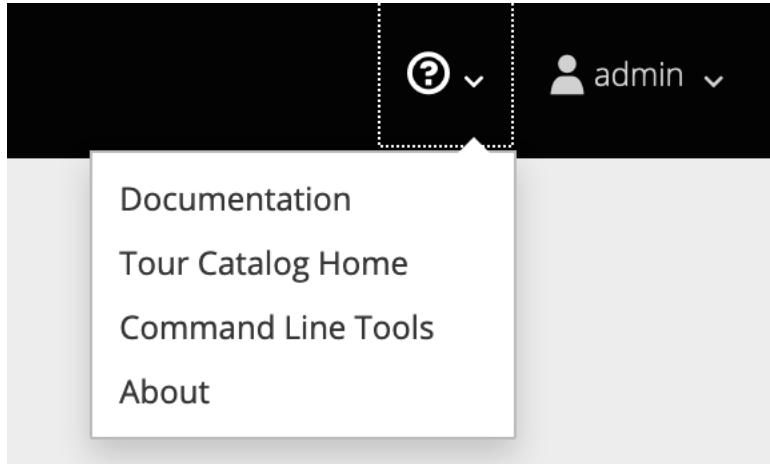
```
if [ "$2" == "" ]
then
echo "Please provide a integration server name as the second argument"
echo "Usage: installACE.sh <release_name> <Integration_server_name> <queue_manager_name>
[<secret_name>]"
exit 1
fi
if [ "$3" == "" ]
then
echo "Please provide a queue manager name as the third argument"
echo "Usage: installACE.sh <release_name> <Integration_server_name> <queue_manager_name>
[<secret_name>]"
exit 1
fi
if [ "$4" == "" ]
then
SECRET_ARG="nil"
else
SECRET_ARG=$4
fi
sudo helm install local-charts/ibm-ace-server-icp4i-prod
--name=$1
--set license=accept
--tls
--set integrationServer.configurationSecret=$SECRET_ARG
--set contentServerURL=https://ace-1-ibm-ace-dashboard-icp4i-
prod:3443/v1/directories/SoE?23e24c6c-c709-4b5e-a7ec-0dd05b96394f
--set imageType=acemqserver
--set image.acemq=docker-registry.default.svc:5000/ace/ibm-ace-mq-server-prod:11.0.0.5.1
--set image.pullPolicy=IfNotPresent
--set image.configurator=docker-registry.default.svc:5000/ace/ibm-ace-icp-configurator-
prod:11.0.0.5.1
```

```
--set acemq.qmname=$3
--set integrationServer.name=$2
--set service.iP=icp-proxy.apps.ocp.cloudnativekube.com
--set persistence.enabled=true
--set persistence.useDynamicProvisioning=true
--set dataPVC.name=data --set dataPVC.storageClassName=managed-nfs-storage
--set productionDeployment=false
--set metrics.enabled=false
```

Configuring an MQ Queue Manager deployed with ACE

Go to the registry console URL

<https://registry-console-default.apps.ocp.cloudnativekube.com/registry>



Select Command line tools and follow the instruction here:

A screenshot of the "Command Line Tools" section of the OpenShift Registry Console. It includes instructions for downloading the "oc" client tool, links to "Latest Release" and "Get Started with the CLI", and a terminal window showing the command "oc login https://master.ocp.cloudnativekube.com:8443 --token=<hidden>".

Once logged in:

Switch to the MQ Project

```
peters-mbp-4:ICP4i peterajessup$ oc project ace
```

Now using project "ace" on server "https://master.ocp.cloudnativekube.com:8443".

Get the ace running pods

```
peters-mbp-4:ICP4i peterajessup$ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
ace-1-ibm-ace-dashboard-icp4i-prod-6c4cff67d9-8px2p	2/2	Running	0	6d
acemq-ibm-ace-server-icp4i-prod-0	1/1	Running	0	22h
daace1-ibm-ace-server-icp4i-prod-0	1/1	Running	0	5d
my-ace-rel-ibm-ace-server-icp4i-prod-66cf9c496c-gzwll	1/1	Running	0	2d
my-ace-rel-ibm-ace-server-icp4i-prod-66cf9c496c-j4n4f	1/1	Running	0	2d

```
my-ace-rel-ibm-ace-server-icp4i-prod-66cf9c496c-znzkb 1/1     Running  0      2d
```

Identify your pod (assuming it's *acemq-ibm-ace-server-icp4i-prod-0* for the sake of this example)

If you're not sure of the ACE pod name navigate the Cloud Pak Foundation I/F under Helm Releases:

The screenshot shows the 'Workloads' section of the Helm Releases interface. The 'Overview' tab is selected at the top. Below it, the 'Workloads' tab is expanded, showing a list of resource types: Brokered Services, DaemonSets, Deployments, Helm Releases, Jobs, and StatefulSets.

Find the release name you specified in the Chart before deploying and click on this to show the pods:

Pod				
Name	READY	Status	RESTARTS	Age
acemq-ibm-ace-server-icp4i-prod-0	1/1	Running	0	22h

```
kubectl exec -it acemq-ibm-ace-server-icp4i-prod-0 -- /bin/bash
```

```
bash-4.2$ runmqsc
```

```
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
```

```
Starting MQSC for queue manager QM2.
```

Copy and paste the following into the mqsc tool:

```
DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCONN)
```

```
AMQ8014I: IBM MQ channel created.
```

```
SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)
```

AMQ8877I: IBM MQ channel authentication record set.

```
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE)
ADOPTCTX(YES)
```

AMQ8567I: IBM MQ authentication information changed.

```
SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')
```

AMQ8877I: IBM MQ channel authentication record set.

```
REFRESH SECURITY TYPE(CONNAUTH)
```

```
define qlocal (RTREP.Q)
```

AMQ8006I: IBM MQ queue created.

```
define qlocal(RTREQ.Q)
```

AMQ8006I: IBM MQ queue created.

Hi CTRL-C

5 MQSC commands read.

```
bash-4.2$ exit
```

```
exit
```

You're Done.

See the section on [Testing the Queue Manager](#) for how to test.

MQ on ICP4i on RHOS via the ICP4i Navigator

MQ using the ICP4i console

<https://icp-proxy.apps.ocp.cloudnativekube.com:3443/integration>

The screenshot shows the ICP4i Navigator interface. At the top, there's a category icon for 'MQ' and the word 'MQ'. Below this, a list of instances is shown: 'mq', 'mqdemo1', and 'mqivt1'. Each instance has a three-dot menu icon to its right. At the bottom of the list is a blue 'Add new instance' button with a plus sign icon.

Click on add a new instance.

Information you'll need to configure the release

Collected information you need to configure the release.

ICP4i Proxy address (from ICP configMaps): icp-proxy.apps.ocp.cloudnativekube.com

onboard docker image registry (from oc command)

<https://registry-console-default.apps.ocp.cloudnativekube.com/registry>

The screenshot shows the Atomic Registry interface. At the top, it says 'ATOMIC REGISTRY' twice. On the left is a sidebar with 'Overview', 'Images' (which is selected), and 'Projects'. In the main area, there's a dropdown for 'Project: mq'. Below that is a table titled 'Images' with columns 'Name' and 'Tags'. Two entries are listed:

Name	Tags
mq/ibm-mq-oidc-registration	2.1.0-amd64
mq/ibm-mqadvanced-server-integration	9.1.3.0-r1-amd64

MQ server image:

<docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-integration>

MQ Server image Tag:

[9.1.3.0-r1](#)

MQ Server OIDC registration image

`docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration`

MQ Server OIDC registration image tag:

`2.1.0`

Managed NFS name (from oc command):

`managed-nfs-storage`

Cluster Hostname (for deployment)

`icp-proxy.apps.ocp.cloudnativekube.com`

TLS Secret

`Ibm-mq-tls-secret`

Configuring the MQ (helm) release

Give it a unique name and add it to the mq namespace

The screenshot shows a Helm release configuration form. It includes fields for 'Helm release name *' (set to 'mqivt2'), 'Target namespace *' (set to 'mq'), 'Target Cluster *' (set to 'local-cluster'), and a 'License *' checkbox which is checked and linked to a 'License agreement' page.

Cluster host name - `icp-proxy.apps.ocp.cloudnativekube.com`

Parameters

To install this chart, additional configuration is needed in Quick start. To customize installation, view and edit All parameters.

[Quick start](#)
Required and recommended parameters to view and edit.

[TLS](#)

Configuration settings for TLS

[Cluster hostname *](#)

`icp-proxy.apps.ocp.cloudnativekube.com`

Enter the mq image container details

Image repo - `docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-integration`

Image tag – 9.1.3.0-r1

Results in:

docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-integration:9.1.3.0-r1-
amd64

Image
Configuration settings for the container image

Image repository *	Image tag * <small>i</small>
docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-integration	9.1.3.0-r1
Image pull secret	Image pull policy (for all images) *
Enter value	IfNotPresent

IBM Cloud Pak for Integration
Configuration settings for IBM Cloud Pak for Integration

Namespace where the platform navigator is installed *

integration

Enter the mq oidc rego container details

docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration

2.1.0

Single sign-on
Configuration settings for single sign-on

Registration image repository *	Registration image tag * <small>i</small>
docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration	2.1.0

Enter the TLS secret and check the box.

ibm-mq-tls-secret

TLS
Configuration settings for TLS

Generate Certificate

Cluster hostname *	Secret name <small>i</small>
icp-proxy.apps.ocp.cloudnativekube.com	ibm-mq-tls-secret

Configure persistence

managed-nfs-storage

Persistence
Configuration settings for Persistent Volumes

Enable persistence *

Use dynamic provisioning *

Data PVC
Configuration settings for the main Persistent Volume Claim

Name *	Storage Class name <small>i</small>
data	managed-nfs-storage
Size *	
2Gi	

Give your queue manager a name else it defaults

Queue manager name i

QM3

Uncheck 'Enable Metrics'

Metrics
Configuration settings for Prometheus metrics

Enable metrics *

Install

Leave EVERYTHING else as default and click Install.

Might have to wait a little while – then goto Stateful Sets

Overview

▼ Workloads

Brokered Services

Brokered Services

DaemonSets

Deployments

Helm Releases

Jobs

StatefulSets

mqivt2-ibm-mq	mq	1	1	2 minutes ago	...
---------------	----	---	---	---------------	-----

Wait until you see the Pod ready 1/1

Pods

Search Pods

Name	Namespace	Status	Host IP	Pod IP	Ready	Start Time
mqivt2-ibm-mq-0	mq	Running	135.90.89.42	10.131.3.22	1/1	3 minutes ago

items per page **20** ▾ | 1-1 of 1 items

1 of 1 pages

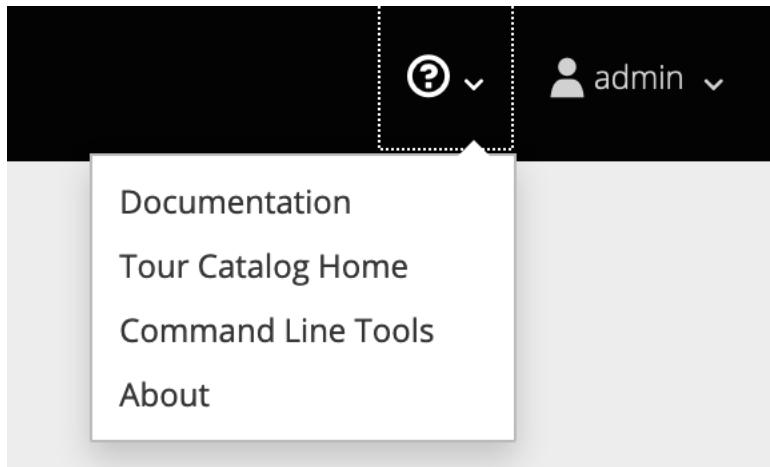
< **1** ▾

MQ on ICP4i via Command line Helm Release

Download command line tools

Go to the registry console URL

<https://registry-console-default.apps.ocp.cloudnativekube.com/registry>



Select Command line tools and follow the instructions here:

Command Line Tools

With the OpenShift command line interface (CLI), you can create applications and manage OpenShift projects from a terminal. You can download the `oc` client tool using the links below. For more information about downloading and installing it, please refer to the [Get Started with the CLI](#) documentation.

[Download `oc`](#) :

[Latest Release](#) ↗

After downloading and installing it, you can start by logging in. You are currently logged into this console as **admin**. If you want to log into the CLI using the same session token:

```
oc login https://master.ocp.cloudnativekube.com:8443 --token=<hidden>
```

Once logged in:

Switch to the MQ Project:

```
oc project mq
```

Get the cloudctl version from ICP:

Goto <https://icp-console.apps.ocp.cloudnativekube.com/console/tools/cli>

Install CLI tools

▼ Install IBM Cloud Private CLI v3.2.0-634

You can download the IBM Cloud [cloudctl](#) for macOS, Windows, and Linux. Download the installer with the following curl command, then see [Installing the IBM® Cloud Private CLI](#) to complete your installation:

Select OS

Select

Download with curl

Command will appear in this field after you make a selection above.

File icon

Select your OS and download the file/tar or zip file.

Once the executable id identified copy it and rename it to cloudctl/cloudctl.exe

Follow the same process for the helm executable (the versions of the helm client and tiller on the server must match):

▼ Install Helm CLI v2.12.3

Helm CLI (helm) is a tool for managing Kubernetes packages, or Helm charts. Deploy and manage applications, open source tools, and IBM software on your IBM Cloud Private environment. Download the installer with the following curl command, then see [Installing Helm](#) to complete your installation:

Select OS

Select

Download with curl

Command will appear in this field after you make a selection above.

File icon

Log into the cluster

Login to the cluster as follows:

```
sudo cloudctl login -a https://icp-console.apps.ocp.cloudnativekube.com -u admin -p ##### -n default
```

Run the following helm command:

```
sudo helm init --client-only --skip-refresh
```

You will see the following in the output of this command:

```
$HELM_HOME has been configured at <your_helm_home>
```

Run the flowing commands:

```
export HELM_HOME=<your_helm_home>  
export ICP_URL=https://icp-console.apps.ocp.cloudnativekube.com
```

The expression below points to the location of the repository used for icp4i

```
sudo helm repo add local-charts $ICP_URL/helm-repo/charts --ca-file $HELM_HOME/ca.pem --cert-file $HELM_HOME/cert.pem --key-file $HELM_HOME/key.pem
```

This has prepared you for running helm install against a chart, image(s) and configuration.

Command line Helm deployment

The MQ chart provides a sample install command and a list of parameters to pass to the installation.

The following script installs an MQ queue manager called ‘qmivt’ with a release name passed in as the first parameter and a queue manager name as the second parameter. The paramters are spaced out per line for clarity but the command should be executed as a single command with no new line characters.

```
sudo helm install ivt1 qmivt  
--name $1 local-charts/ibm-mqadvanced-server-integration-prod  
--set license=accept  
--tls  
--set productionDeployment=false  
--set image.repository=docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-integration  
--set image.tag=9.1.3.0-r1  
--set image.pullPolicy=IfNotPresent  
--set image.pullSecret=nil  
--set icp4i.namespace=integration  
--set sso.registrationImage.repository=docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration  
--set sso.registrationImage.tag=2.1.0  
--set sso.webAdminUsers="admin"  
--set sso.uniqueUserIdentifier="sub"  
--set tls.generate=true  
--set tls.hostname=icp-proxy.apps.ocp.cloudnativekube.com  
--set tls.secret=ibm-mq-tls-secret  
--set persistence.enabled=true  
--set persistence.useDynamicProvisioning=true
```

```
--set dataPVC.name="data"
--set dataPVC.storageClassName="managed-nfs-storage"
--set metrics.enabled=false
--set queueManager.name=$2
```

Configuring your Queue Manager for client connection

Go to Integration Navigator and click on your new MQ Service.

MQ

mq

mqdemo1

⋮

mqivt1

⋮

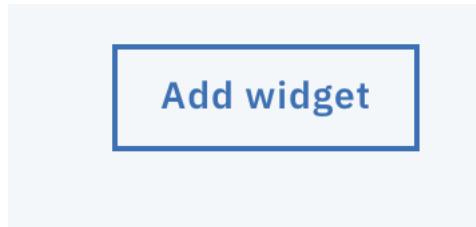
[mqivt2](#)

⋮

Connect via the MQ Web Console

Click through the security warnings to the MQ Console

Click Add Widget:



Select Queues:

Add a new widget

[Local Queue Managers](#) Manage local queue managers

[Chart](#) Monitor your MQ platform

Add a widget to display MQ object information for the specified queue manager
Queue manager:

QM3 ▾

[Queues](#) Configure destinations for messages

[Topics](#) Administrative objects for assigning attributes to topics

[Listeners](#) Configure processes to accept network requests

[Channels](#) Queue manager communication paths

[Client-connection Channels](#) Client connectivity details

[Authentication Information](#) Configure authentication mechanisms

[Subscriptions](#) Configure how subscriptions to topics are handled

[Channel Authentication Records](#) Control access to channels

[Close](#)

Repeat this for Channels, Channel Authentication records and Authentication Information.

Click on Create Channel:

Channels on QM3

⟳ ⚙ ✎

🔍 Search

Create +

▲ Name

Type

Overall channel status

Create a Channel

Channel name: * 

IVT.SVRCONN

Channel type: *

Server-connection

[Cancel](#)

[Create](#)

Click 'Create'

On Channel Authentication Records, click the Cog Icon

Channel Authentication Records on QM3   

▲ Channel profile	Type
*	Block User List

[Create](#) 

Select 'System Objects' -> Show, then click Save.

Repeat this step for Authentication Information as well.

Select the 'Block User List on Channel Authentication Records and hit Delete then confirm delete.

Channel Authentication Records on QM3   

[Delete](#)  [Properties](#)  1 item selected [Deselect](#)

▲ Channel profile	Type
*	Block User List

Click on 'Create'

Select 'Address' from the drop down list, then click Next.

Create a Channel Authentication Record

Rule Type:

Allow Block Warn

This rule will be used to allow access to inbound connections.

Identity:

Address

Matches the IP address or hostname of the remote machine.

Enter the channel name created earlier, enter an Asterix in the Address box and click Next.

Create a Channel Authentication Record

Channel profile: *

IVT.SVRCONN

Address: *

*|

Back

Next

Click on 'Map' and enter 'mqm' as the MCA user ID*

Create a Channel Authentication Record

User source:

Channel Map

MCA user ID: *

mqm

Back

Next

Click 'Next' then 'Create'.

Select the 'ID PWOS' line on the 'Authentication Information' widget and click on 'Properties'

Authentication Information on QM3		⋮	✖		
Name	Type	⋮	Delete	Properties	Deselect
SYSTEM.DEFAULT.AUTHINFO.CRLLDAP	CRL LDAP				
SYSTEM.DEFAULT.AUTHINFO.IDPWLDAP	IDPW LDAP				
SYSTEM.DEFAULT.AUTHINFO.IDPWOS	IDPW OS				
SYSTEM.DEFAULT.AUTHINFO.OCSP	OCSP				

Change 'Client Connections' to 'None' in the dropdown list

Properties for 'SYSTEM.DEFAULT.AUTHINFO.IDPWOS'

General	Local connections:	Client connections:	Adoption context:
User ID + password	Optional	None	Yes
Statistics	Failure delay:	Auth method:	
	1	Operating system	

Click 'Save', then 'Close'

On the Queue Manager Widget select the QM3 queue manager, click the '...' menu and select 'Refresh Security', then click on the 'Connection authentication' link.

The screenshot shows the 'Local Queue Managers' properties dialog. At the top, there are tabs for 'Properties' (which is selected and highlighted with a red circle), '...', '1 item selected', and 'Deselect'. Below the tabs, there is a table with two columns: 'Name' and 'Status'. A single row is visible for 'QM3', which is marked as 'Running' with a green dot. The entire dialog has a light gray background and a white header bar.

Refresh security

Refresh queue manager security on 'QM3'

The list of authorizations held internally by the [Authorization service](#) authorization services component is refreshed. This is the default value.

[Connection authentication](#)

Refreshes the cached view of the configuration for connection authentication.

Refreshes the cached view of the Secure Sockets Layer, or Transport Layer Security, key repository.

Also refreshed are the locations of the LDAP servers to be used for Certified Revocation Lists and the key repository, as well as any cryptographic hardware parameters specified through IBM MQ.



On the Queues widget click 'Create'.

Create a Queue

Queue name: * 

RTREP.Q

Queue type:

Local Remote Alias Model

Cancel

Create

Enter as shown and click 'Create'.

Repeat this for a queue called 'RTREQ.Q'

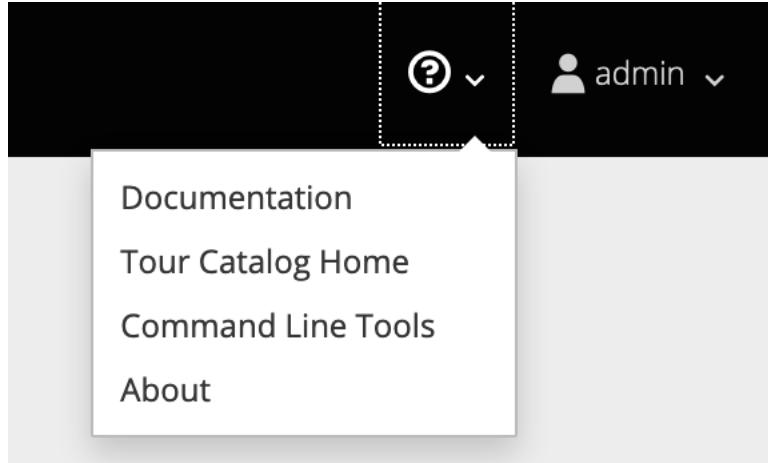
Alternative Configuration procedure for a Queue manager

Bash into the container and runmqsc

This procedure uses the remote login facility of the container.

Go to the registry console URL

<https://registry-console-default.apps.ocp.cloudnativekube.com/registry>



Select Command line tools and follow the instruction here:

A screenshot showing the "Command Line Tools" section of the OpenShift Registry Console. It includes instructions for downloading the "oc" client tool, links to "Latest Release" and "Get Started with the CLI" documentation, and a terminal-like interface showing the command "oc login https://master.ocp.cloudnativekube.com:8443 --token=<hidden>".

Once logged in:

Switch to the MQ Project

Oc project mq

```
peters-mbp-4:ICP4i peterajessup$ oc project mq
```

Now using project "mq" on server "https://master.ocp.cloudnativekube.com:8443".

Get the mq running pods

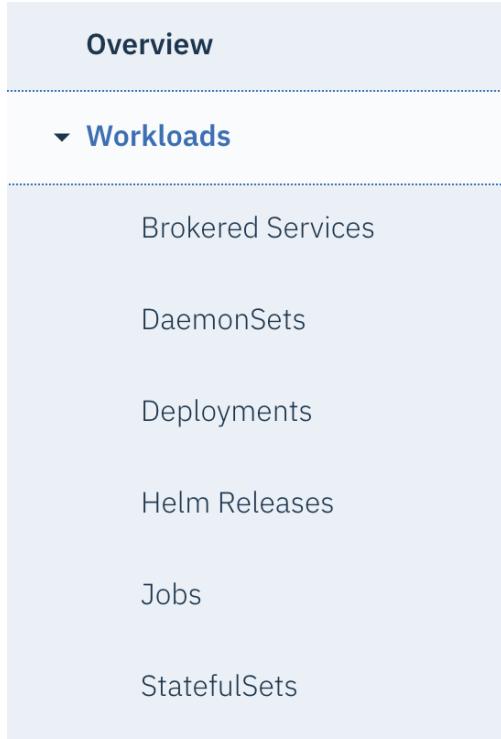
```
peters-mbp-4:ICP4i peterajessup$ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
mqdemo1-ibm-mq-0	1/1	Running	0	6d
mqdemo1-ibm-mq-registration-6d64j	0/1	Completed	0	6d
mqivt1-ibm-mq-0	1/1	Running	0	1d

```
mqivt1-ibm-mq-registration-jc5p9  0/1    Completed  0      1d
```

Identify your pod (assuming it's mqivt-ibm-mq-0 for the sake of this example)

If you're not sure of the MQ pod name navigate the Cloud Pak Foundation I/F under Helm Releases:



Find the release name you specified in the Chart before deploying and click on this to show the pods:

Pod					
Name	READY	Status	RESTARTS	Age	
mqivt1-ibm-mq-0	1/1	Running	0	47h	
mqivt1-ibm-mq-registration-jc5p9	0/1	Completed	0	47h	

Hint – the 'running pod' is the one you want.....

```
kubectl exec -it mqivt1-ibm-mq-0 -- /bin/bash
```

```
bash-4.2$ runmqsc
```

```
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
```

```
Starting MQSC for queue manager QM2.
```

Copy and paste the following into the mqsc tool:

```
DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCNN)
```

```
AMQ8014I: IBM MQ channel created.
```

```
SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)
```

AMQ8877I: IBM MQ channel authentication record set.

```
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE)
ADOPTCTX(YES)
```

AMQ8567I: IBM MQ authentication information changed.

```
SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')
```

AMQ8877I: IBM MQ channel authentication record set.

```
REFRESH SECURITY TYPE(CONNAUTH)
```

```
define qlocal (RTREP.Q)
```

AMQ8006I: IBM MQ queue created.

```
define qlocal(RTREQ.Q)
```

AMQ8006I: IBM MQ queue created.

Hi CTRL-C

5 MQSC commands read.

bash-4.2\$ exit

exit

Testing the Queue Manager.

Go to the Helm Releases view in the Cloud Pak Foundation interface

The screenshot shows the Helm Releases page of the IBM Cloud Private interface. At the top, there's a dark header with the 'IBM Cloud Private' logo and the 'mycluster' name. Below the header, the title 'Helm Releases' is centered. Underneath the title, a light blue message box contains the text: 'You are currently viewing only the helm releases of this cluster.' with an info icon.

Locate the release name you chose and click on the list item.

Scroll down to the 'Service' and note the port mapping 1414:xxxxx/TCP

Service					
Name	Type	Cluster IP	External IP	Port(s)	Age
mqivt2-ibm-mq	NodePort	172.30.1.110	<none>	9443:30526/TCP,1414:31028/TCP	37m

You will connect to your queue manager via the following settings:

Hostname: **icp-proxy.apps.ocp.cloudnativekube.com**

Port: **xxxxx – in this case it's 31028**

Channel Name: **IVT.SVRCONN**

Username – **not required**

Password – **not required**

Queue Manager - **<the_helm_release_name> or whatever you chose for the queue manager name**

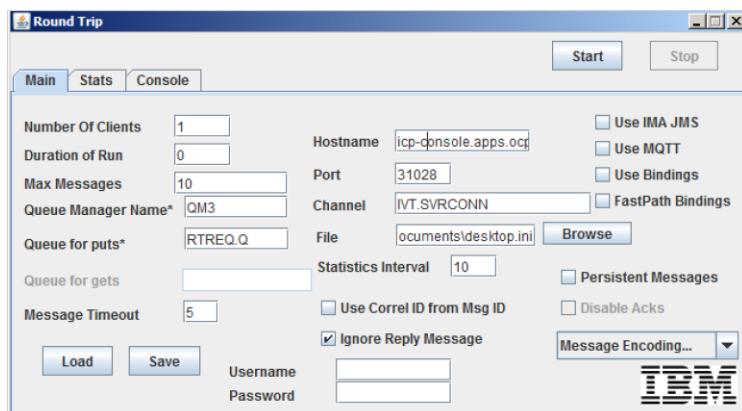
Queue Name - **RTREQ.Q**

Using the RoundTrip GUI

Unzip the RoundTrip.zip into its own directory.

Run the rt.sh script. You may need to modify the script to suit your own path to the java executable. See the appendix for the equivalent rt.bat file for windows users .

The UI should appear :



Fill in the hostname, port, channel and Queue Manager Names outlined above with your values.

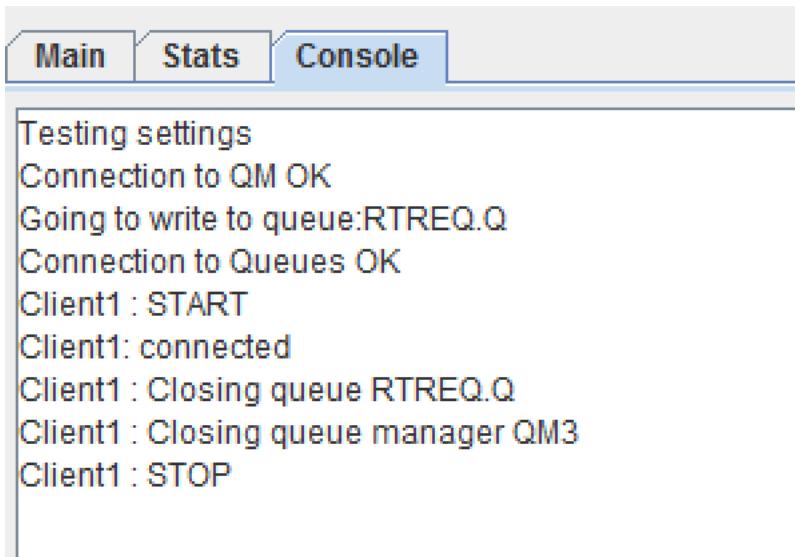
For a simple test, check 'Ignore Reply Message' and set the 'Queue for Puts' to 'RTREQ.Q'

Set Max Messages to 10.

Click on 'Browse' and select a message file to load for the payload data (small file is best)

Check the console tab and click 'Start'.

You should see something similar to:



Or you may see an error stack trace if something was not configured correctly.

You can click 'Save' to save the configuration to file for quick loading (use 'Load') when re-testing.

ACE+MQ via the ICP4i GUI

The screenshot shows the IBM Cloud Pak for Integration Platform home page. At the top, there is a navigation bar with a menu icon, the text "IBM Cloud Pak for Integration", and "Platform home". Below the navigation bar is a search bar with a magnifying glass icon and the placeholder text "Find". The main content area is divided into two sections: "API Connect" and "App Connect".

API Connect
Unlock business data and assets as APIs

App Connect
ace
ace-1

Select ace-1

Add server

The screenshot shows the "Servers" tab in the IBM Cloud Pak for Integration interface. At the top, there is a navigation bar with tabs for "Integrations" (which is highlighted) and "Servers". Below the navigation bar is a search bar with a magnifying glass icon and the placeholder text "Search". There is also a "Add server" button with a plus sign.

Server	Type	Release name	Status
IS1	Server (with MQ)	acefin-es	Started
IS2	Server	aceutil	Started

Select bar

Add server

Provide a BAR file to deploy to the server



Add a BAR file

or use existing BAR file:

RoundTripEcho



Cancel

Continue

Copy (and paste) the content URL

Add server

You will now configure and install a Helm release to deploy to the server. It is important to copy the Content URL and select the current Namespace.

Content URL:

<https://ace-1-ibm-ace-dashboard-icp4i-prod:3443/v1/directories/RoundTripEcho?0486b916-ddd4-42a3-94e3-c60d8d3f2145> 

Namespace:

ace

If the integration server requires any configuration to be applied then you will need to use the following download to provide the configuration prior to install. Refer to the README.md inside the download on how to create the required secrets:

[Cancel](#)

[Configure release !\[\]\(2a6a2a8c52fe433b8a86f86a09f1f793_img.jpg\)](#)

<https://ace-1-ibm-ace-dashboard-icp4i-prod:3443/v1/directories/RoundTripEcho?0486b916-ddd4-42a3-94e3-c60d8d3f2145>

optional step:

If the integration server requires any configuration to be applied then you will need to use the following download to provide the configuration prior to install. Refer to the README.md inside the download on how to create the required secrets:

[Download configuration package](#)

Hit Configure release



IBM APP CONNECT ENTERPRISE



Introduction

IBM® App Connect Enterprise is a market-leading lightweight enterprise integration engine that offers a fast, simple way for systems and applications to communicate with each other. As a result, it can help you achieve business value, reduce IT complexity and save money. IBM App Connect Enterprise supports a range of integration choices, skills and interfaces to optimize the value of existing technology investments.

Configure

Helm release name *

eaiqm1p

Target namespace *

ace

Target Cluster *

local-cluster

License * i

I have read and agreed to the [License agreement](#)

Paste in the content server URL

Proxy Node IP or FQDN

Get this from from the ICP main console -> configuration ->config-maps and filter on info. Find ibmcloud-cluster-info and it's the proxy address. i.e.=

icp-proxy.apps.ocp.cloudnativekube.com

Content Server URL *

`https://ace-1-ibm-ace-dashboard-icp4i-prod:3443/v1/directories/RoundTripEcho?0486b916-ddd4-42a3-94e3-c60d8d3f2145`

Service

Service settings

Proxy Node IP or FQDN * i

`icp-proxy.apps.ocp.cloudnativekube.com`

Production usage

Which type of image to run

Architecture scheduling preference *

App Connect Enterprise with MQ server

amd64

Configure the Image(s) to use the correct registry

Before:

Docker image for App Connect Enterprise with MQ server *

cp.icr.io/ibm-ace-mq-server-prod:11.0.0.5.1

After

Docker image for App Connect Enterprise with MQ server *

docker-registry.default.svc:5000/ace/ibm-ace-mq-server-prod:11.0.0.5.1

You can find all the images and their details in the onboard registry via the OKD console

<https://registry-console-default.apps.ocp.cloudnativekube.com/registry>

<https://registry-console-default.apps.ocp.cloudnativekube.com/registry#/images/ace/ibm-ace-mq-server-prod>

on board registry for ace / mq images (rather than the cloud managed service)

replace cp.icr.io with docker-registry.default.svc:5000/ace

Before

cp.icr.io/ibm-ace-mq-server-prod:11.0.0.5.1

After

docker-registry.default.svc:5000/ace/ibm-ace-mq-server-prod:11.0.0.5.1

Here's an example of all the possible images changed

Docker image for App Connect Enterprise * docker-registry.default.svc:5000/ace/ibm-ace-server-prod:11.0.0.5.1	Docker image for App Connect Enterprise with MQ client * docker-registry.default.svc:5000/ace/ibm-ace-mqclient-server-prod:11.0.0.5.1
Docker image for App Connect Enterprise with MQ server * docker-registry.default.svc:5000/ace/ibm-ace-mq-server-prod:11.0.0.5.1	Configurator Docker image * docker-registry.default.svc:5000/ace/ibm-ace-icp-configurator-prod:11.0.0.5.1

For the image pull secret

You can use the oc get secrets command

use `deployer-dockercfg-k8tdp` as image pull secret

Image pull secret i

`deployer-dockercfg-k8tdp`

Integration server name i

`ACEINTSRV1`

Replica count

`1`

Queue manager name i

`EAIQM1`

Use `oc get storage` command to find the storage classes available

Storage class name = `managed-nfs-storage`

Data persistent volume claims (PVCs)

Settings for the PVCs (applicable only when running with a queue manager)

Name *

`data`

Storage class name i

`managed-nfs-storage`

Size *

`2Gi`

readiness probe. Change `from 10 seconds` to 30 seconds as things might take a bit longer to start with a queue manager in there too

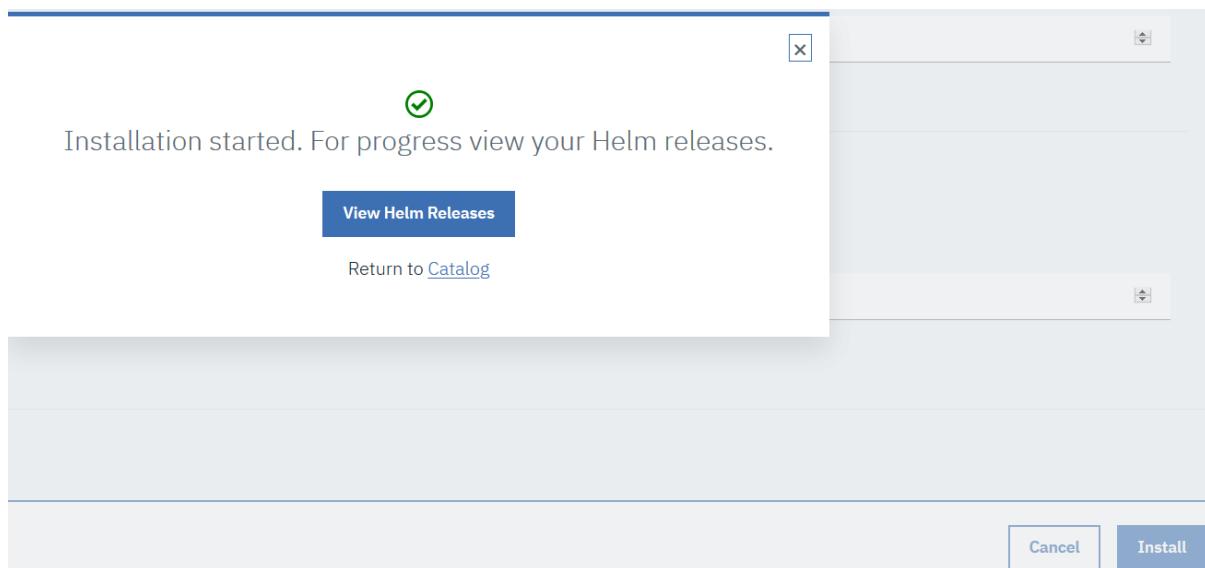
Readiness probe

Settings for the readiness probe that checks if the integration server is ready

Initial delay (seconds)

30

Click install



Wait a while and click view helm releases – filter via eai

The screenshot shows the "Helm Releases" page in the IBM Cloud Private interface. The top navigation bar includes the IBM Cloud Private logo and the word "mycluster". The main title is "Helm Releases". A message box at the top states: "You are currently viewing only the helm releases of this cluster." Below this, a search bar contains the text "eai". The table below lists the following data:

Name	Namespace	Status	Chart name
eaiqm1p	ace	● Deployed	ibm-ace-server-icp4i-prod

Modifying the queue manager connection security

You can BASH into the pod in order to execute a runmqsc command to enable remote admin because we did not use the configuration package to set a specific mqsc file.

Get the pod name from the helm release

Name
eaiqm1p-ib-1352-0

You can also use oc get pods (or kubectl)

Configure client

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

Prerequisites:

[Install CLI tools](#)

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

```
kubectl config set-cluster mycluster --server=https://console.apps.ocp.cloudnati^
kubectl config set-context mycluster-context --cluster=mycluster
kubectl config set-credentials admin --token=C99UTMlgDHdxoWS_cm0xGmFsundefined7I
kubectl config set-context mycluster-context --user=admin --namespace=default
kubectl config use-context mycluster-context
```

```

kubectl config set-cluster mycluster --
server=https://console.apps.ocp.cloudnativekube.com:443 --insecure-skip-tls-verify=true

kubectl config set-context mycluster-context --cluster=mycluster
kubectl config set-credentials admin --
token=C99UTMlgDHdxoWS_cm0xGmFsundefined7D_mxYfHYHOa758aPAO
kubectl config set-context mycluster-context --user=admin --namespace=default
kubectl config use-context mycluster-context

```

You can use the token in the oc login on the command line

```

oc login https://master.ocp.cloudnativekube.com:8443 --
token=C99UTMlgDHdxoWS_cm0xGmFsundefined7D_mxYfHYHOa758aPAO

```

switch projects

```
oc project ace
```

get the pods

```
oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
ace-1-ibm-ace-dashboard-icp4i-prod-6c4cff67d9-8px2p	2/2	Running	0	20d
acefin-es-ib-1352-0	1/1	Running	0	1d
aceutil-ibm-ace-server-icp4i-prod-6c96d85577-8mt6w	1/1	Running	0	4d
eaiqm1p-ib-1352-0	1/1	Running	0	33m

```
kubectl exec -it eaiqm1p-ib-1352-0 -- /bin/bash
```

runmqsc

```

TO MQ REPOS M NODE\ATO-MQREPOSM-NODE2-CHLMQSC>kubectl exec -it eaiqm1p-ib-1352-0 -- /bin/bash
bash-4.2$ runmqsc
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager EAIQM1.

```

Cut and paste in the following five command one at a time

```
DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCNN)
```

```
SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)
```

```
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE)
ADOPTCTX(YES)
```

```
SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')
```

```
REFRESH SECURITY TYPE(CONNAUTH)
```

What out for dodgy ‘‘ when cutting and pasting see below – you need to type over the “ in the bash

Note might be required **DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCNN) MCAUSER('mqm')**

```

Starting MQSC for queue manager EAIQM1.

DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCONN)
 1 : DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCONN)
AMQ8014I: IBM MQ channel created.
SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)
 2 : SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)
AMQ8877I: IBM MQ channel authentication record set.
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE) ADOPTCTX(YES)
 3 : ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE) ADOPTCTX(YES)
AMQ8567I: IBM MQ authentication information changed.
SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')
 4 : SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')
AMQ8405I: Syntax error detected at or near end of command segment below:-
SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER(@

AMQ8427I: Valid syntax for the MQSC command:

SET CHLAUTH( channel_profile )
  TYPE( ADDRESSMAP )
  ADDRESS( string )
  [ DESCRIPTOR( string ) ]
  [ CHCKCLNT( ASQMGR | REQDADM | REQUIRED ) ]
  [ CUSTOM( string ) ]
  [ USERSRC( CHANNEL | MAP | NOACCESS ) ]
  [ MCAUSER( string ) ]          [ WARN( NO | YES ) ]
  [ ACTION( ADD | REPLACE | REMOVE | REMOVEALL ) ]
SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')
 5 : SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')
AMQ8877I: IBM MQ channel authentication record set.
REFRESH SECURITY TYPE(CONNAUTH)
 6 : REFRESH SECURITY TYPE(CONNAUTH)
AMQ8560I: IBM MQ security cache refreshed.

```

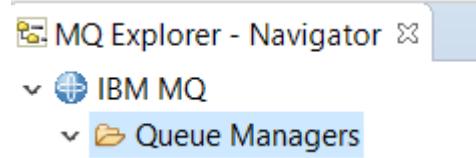
Type end to end runmqsc

```

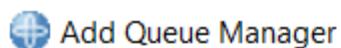
end
 7 : end
6 MQSC commands read.
One command has a syntax error.
All valid MQSC commands were processed.
bash-4.2$
```

Exit to exit the bash

Try connecting MQ Explorer



Click on Queue Managers and select add remote queue manager



Select the queue manager and connection method

Identify the queue manager to add and choose the connection method to use

Queue manager name:

How do you want to connect to this queue manager?

Connect directly

This option creates a new connection to the queue manager (recommended)

Queue manager name:

Connection details

Host name or IP address:

Port number:

Server-connection channel:

Queue manager name:

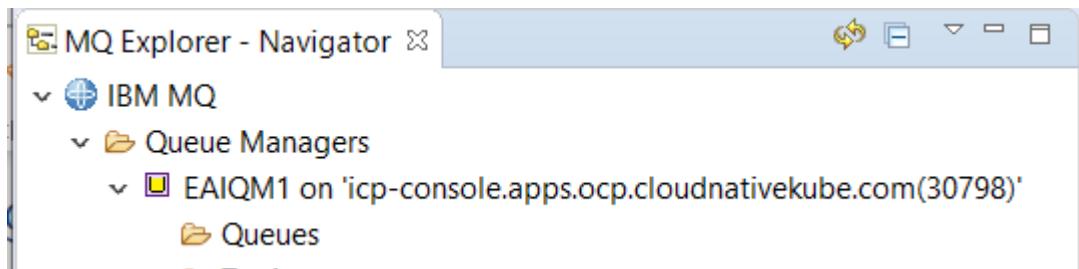
Enable user identification

User identification compatibility mode

Userid:

Password

No password



Point of note

I could not get into the queue manager with

SET MQSERVER=IVT.SVRCONN/TCP/icp-console.apps.ocp.cloudnativekube.com(30798)

and

runmqsc -c EAIQM1

until I put mqm in the MCA user ID on the IVT.SVRCONN

(I also deleted the * and another channel auth record – first but that didn't help)

The screenshot shows the MQ Explorer interface with the title bar 'MQ Explorer - Navigator'. The left pane displays the tree structure under EAIQM1:

- Queues
- Topics
- Subscriptions
- Channels
 - Client Connections
 - Chans
 - Listeners
 - Services
 - Process
 - Namelis
 - Authenti
 - Communi

The 'Chans' item is selected, and its properties are shown in the right pane. The 'Properties' tab is selected. A table titled 'Filter: Standard for Channels' lists the following channels:

Channel name	Channel type	Overall chann
IVT.SVRCONN	Server-connection	Running
SYSTEM.AUTO.RECEIVER	Receiver	Inactive
SYSTEM.AUTO.SVRCONN	Server-connection	Inactive

Below the table, the 'MCA' tab is selected in the properties panel. The 'MCA user ID:' field contains 'mqm'.

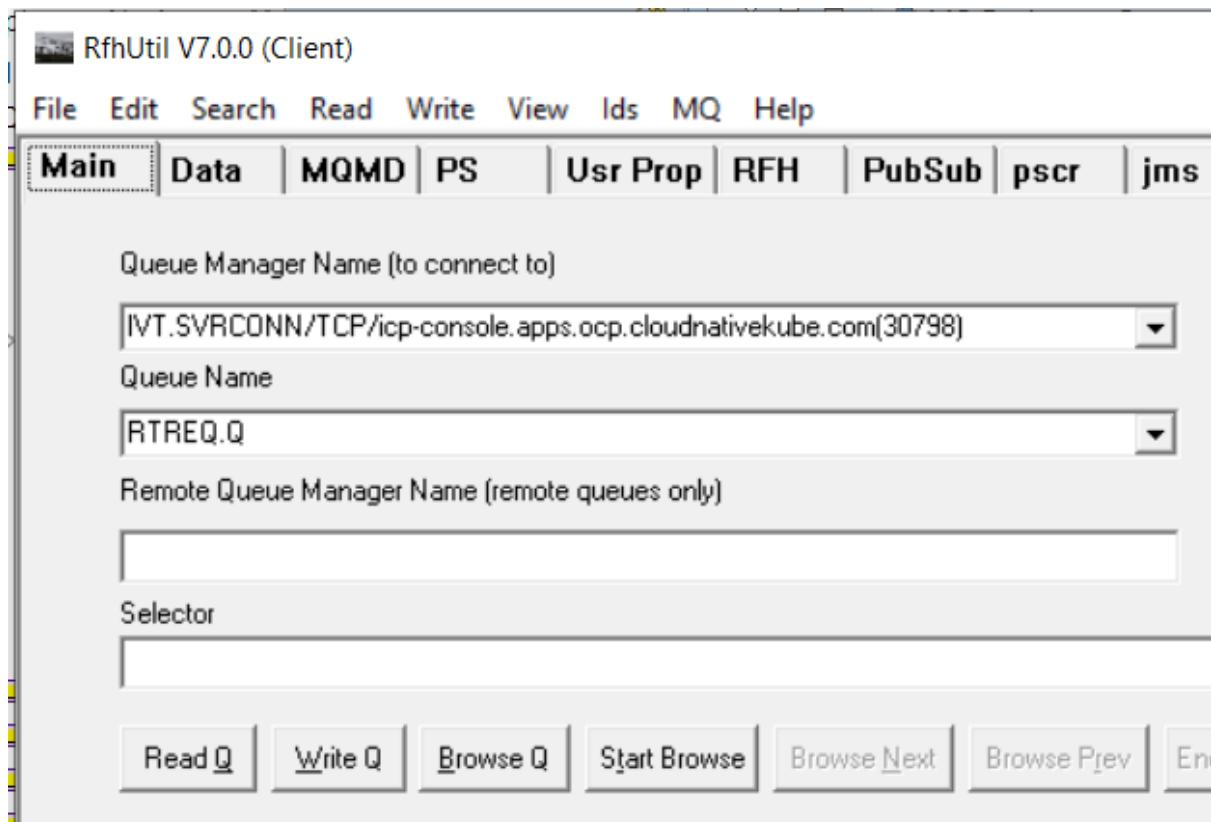
In a second attempt in the bash runmqsc I added the MCAUSER and that allows runmqsc -c to connect

Note might be required DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCONN) MCAUSER('mqm')

Connecting RFHUTILC to test

To connect to EAIQM1

IVT.SVRCONN/TCP/icp-console.apps.ocp.cloudnativekube.com(30798)



MQ Explorer - Navigator

IBM MQ

Queue Managers

EAIQM1 on icp-console.apps.ocp.cloudnativekube.com(30798)

- Queues
- Topics
- Subscriptions
- Channels
- Listeners

MQ Explorer - Content

Queues

Filter: Standard for Queues

Queue name	Queue type	Open input count	Open output count	Current queue depth
RTREP.Q	Local	0	0	0
RTREQ.Q	Local	1	0	0
TEST.IIB.Q1	Local	0	0	0

Open a file to load some data

File Name

C:\SHARE\SERVICES\DATools\RFHUTILv7\parmlst1.txt

Open File Save File Clear Data Clear All



11.42.15 Message sent to RTREQ.Q length=3455

Queues				
Filter: Standard for Queues				
Queue name	Queue type	Open input count	Open output count	Current queue depth
RTREP.Q	Local	0	1	1
RTREQ.Q	Local	1	1	0

Using the RoundTrip GUI

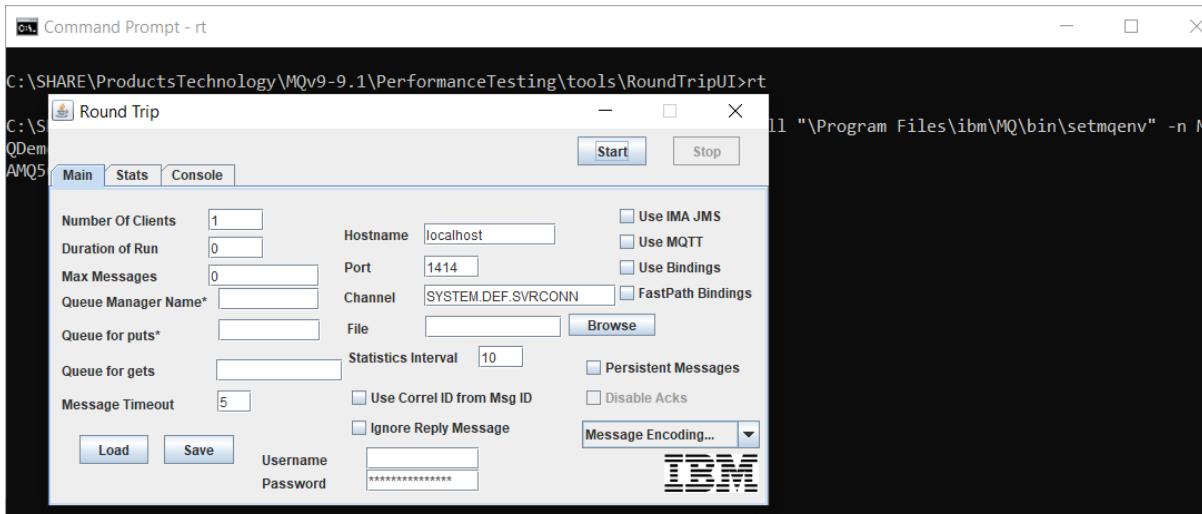
Unzip the RoundTrip.zip into it's own directory.

Run the rt.sh script. You may need to modify the script to suit your own path to the java executable. See the appendix for the equivalent rt.bat file for windows users .

RT.BAT for windows

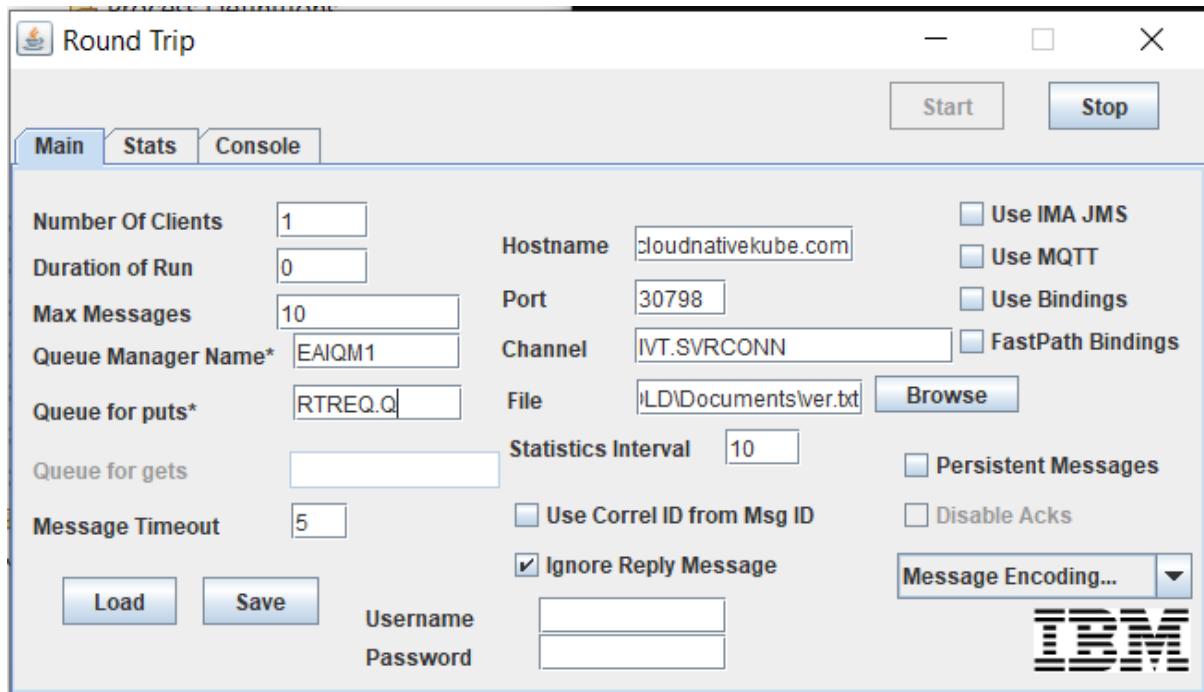
```
call "\Program Files\ibm\MQ\bin\setmqenv" -n MQDemo1 -x 64
```

```
rem "c:\Program Files\ibm\MQ\java\jre\bin\java" -classpath  
.connector.jar;.\fscontext.jar;.\com.ibm.mq.jar;.\com.ibm.mq.jmqi.jar;.\connector.jar;.\com.ibm.mq.headers.jar;.\com.ibm.mq.commonservices.jar;.\jms.jar;.\jndi.jar;.\imaclientjms.jar;.\jms.jar;.\GUIRoundTrip.jar;org.dyno.visual.swing.jar;org.dyno.visual.swing.layouts.jar  
com.ibm.mq.testclient.RoundTripF  
  
"c:\Program Files\ibm\MQ\java\jre\bin\java" -classpath  
.com.ibm.mq.allclient.jar;.\jms.jar;.\jndi.jar;.\imaclientjms.jar;.\jms.jar;.\GUIRoundTrip.jar;org.dyno.visual.swing.jar;org.dyno.visual.swing.layouts.jar com.ibm.mq.testclient.RoundTripF
```



The UI should appear :

Using RoundTrip for One-way testing



To connect to EAIQM1

Channel: IVT.SVRCONN

Hostname: icp-console.apps.ocp.cloudnativekube.com

Port: 30798

Fill in the hostname, port, channel and Queue Manager Names outlined above with your values.

For a simple test, check 'Ignore Reply Message' and set the 'Queue for Puts' to 'RTREQ.Q'

Set Max Messages to 10.

Click on 'Browse' and select a message file to load for the payload data (small file is best)

Check the console tab and click 'Start'.

The interface shows two main sections: 'Queue Managers' and 'Queues'.

Queue Managers:

- Tree view: Queue Managers > EAIQM1 on 'icp-console.apps.ocp.cloudnativekube.com'(30798) > Queues, Topics, Subscriptions, Channels, Listeners.

Queues:

Queue name	Queue type	Open input count	Open output count	Current queue depth
RTREP.Q	Local	0	1	11
RTREQ.Q	Local	1	1	0
TEST.IIB.Q1	Local	0	0	0

Result should be (1 message already on queue from RFHUTIL) 11 messages on the RTREP.Q as the ACEINTSRV1 RoundTrip echo flow is moving messages from RTREQ.Q to RTREP.Q

Clear the messages

Queues

Filter: Standard for Queues

Queue name	Queue type	Open input count	Open output count	Current queue depth	Put messages
RTREP.Q	Local	0	0	11	Allowed
RTREQ.Q	Local				
TEST.IIB.Q1	Local				

Clear queue

Queue manager name: EAIQM1
Queue name: RTREP.Q

Select clear type

Queue will be cleared using CLEAR command
 Queue will be cleared using MQGET API calls

Using RoundTrip for RoundTrip testing – no correlID matching

Round Trip

Main Stats Console Start Stop

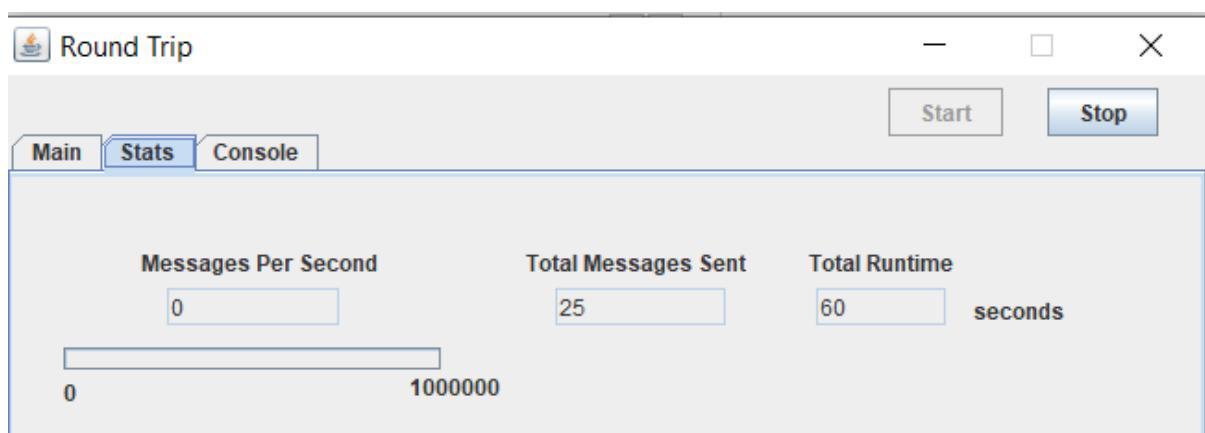
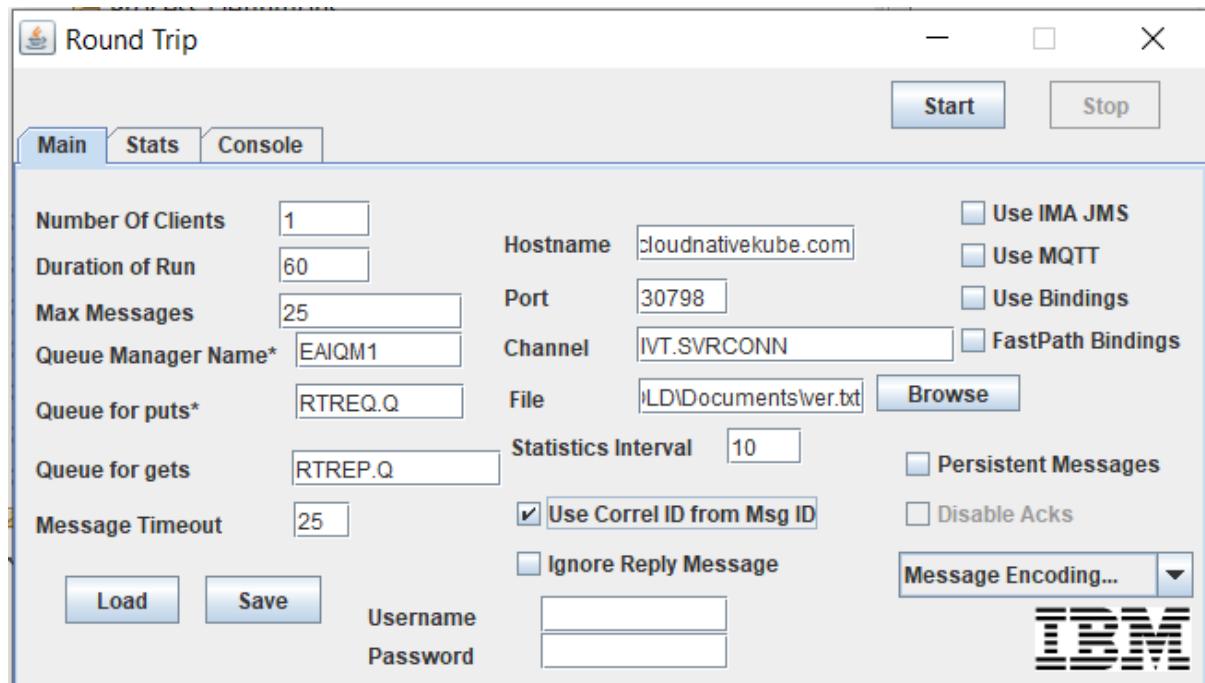
Number Of Clients: 1 Hostname: cloudnativekube.com Use IMA JMS
Duration of Run: 60 Use MQTT
Max Messages: 25 Use Bindings
Queue Manager Name*: EAIQM1 Channel: IVT.SVRCONN FastPath Bindings
Queue for puts*: RTREQ.Q File: C:\LDI\Documents\ver.txt Browse
Queue for gets: RTREP.Q Statistics Interval: 10 Persistent Messages
Message Timeout: 25 Use Correl ID from Msg ID Disable Acks
Load Save Username
Password Message Encoding... IBM

Round Trip

Main Stats Console Start Stop

Messages Per Second: 0 Total Messages Sent: 25 Total Runtime: 60 seconds
0 1000000

Using RoundTrip for RoundTrip testing – correlID matching



Routing Messages through IDC to EAI

The screenshot shows two windows of the IBM MQ Explorer tool. The left window, titled 'MQ Explorer - Navigator', displays a tree view of queue managers and their resources. The right window, titled 'TEST.IIB.Q1.A', is a detailed configuration and monitoring interface for a specific queue.

MQ Explorer - Navigator:

- IBM MQ
 - Queue Managers
 - EAIQM1 on 'icp-console.apps.ocp.cloudnativekube.com(30798)
 - EAIQM2 on 'icp-console.apps.ocp.cloudnativekube.com(32588)
 - IDCQM1 on 'icp-console.apps.ocp.cloudnativekube.com(30419)
 - IDCQM2 on 'icp-proxy.apps.ocp.cloudnativekube.com(31971)'

TEST.IIB.Q1.A:

File Edit Search Read Write View Ids MQ Help

Main Data MQMD PS Usr Prop RFH PubSub pscr jms

Queue Manager Name (to connect to): JV.T.SVRCONN/TCP/icp-proxy.apps.ocp.cloudnativekube.com(30419)

Queue Name: TEST.IIB.Q1.A

Remote Queue Manager Name (remote queues only): (empty)

Selector: (empty)

Buttons: Read Q, Write Q, Browse Q, Start Browse, Browse Next, Browse Prev, End Bro

File Name: C:\SHARE\SERVICES\DATools\RFHUTILv7\samples\bmsale1.txt **Data Size:** 55

Buttons: Open File, Save File, Clear Data, Clear All, Load Names, Set Conn Id

COBOL Copy Book File Name: (empty)

Checkboxes: Put/Get C, New, Get t, Get b

Message Log: 13:15:49 Message sent to TEST.IIB.Q1.A length=55

The screenshot shows the 'Queues' table from the 'MQ Explorer - Navigator' window. It lists three local queues: RTREQ.Q, RTREP.Q, and TEST.IIB.Q1, all of which have an open input count of 0 and an open output count of 1, resulting in a current queue depth of 1.

Queue name	Queue type	Open input count	Open output count	Current queue depth
RTREQ.Q	Local	0	1	0
RTREP.Q	Local	1	0	0
TEST.IIB.Q1	Local	0	1	1

Deploying API Connect for IBM Cloud Pak for Integration

The deployment information in this document is closely aligned with the Residency Playbook for CP4I which can be found here:

<https://pages.github.ibm.com/CASE/cloudpak-onboard-residency/integration/deploy-api-mgmt/>

Alternatively, there is a slightly older Recipe on DeveloperWorks on Deploying and Configuring IBM API Connect 2018 on ICP4I 2019.3.2.1 (on OCP 3.11)

<https://developer.ibm.com/recipes/tutorials/deploying-and-configuring-ibm-api-connect-2018-on-icp4i-2019-3-2-1-on-ocp-3-11>

Prepare Endpoints

We have to define the endpoint for each of the APIC subsystems. We can “construct” the endpoints by adding descriptive “prefixes” to the proxy URL

Management – all endpoints

The Management endpoints can all be the same like mgmt.apps.ocp.cloudnativekube.com, but we chose individual names for the endpoints.

Platform API Endpoint

platformapi.apps.ocp.cloudnativekube.com

Consumer API Endpoint

capi.apps.ocp.cloudnativekube.com

Cloud Admin UI

adminui.apps.ocp.cloudnativekube.com

API Manager UI

mgrui.apps.ocp.cloudnativekube.com

Portal endpoints

Portal – director

portaldirector.apps.ocp.cloudnativekube.com

Portal – web

portalweb.apps.ocp.cloudnativekube.com

Analytics endpoints

Analytics – ingestion

analingest.apps.ocp.cloudnativekube.com

Analytics - client

analclient.apps.ocp.cloudnativekube.com

Gateway endpoints

API Gateway

apigw.apps.ocp.cloudnativekube.com

Gateway Service

gwservice.apps.ocp.cloudnativekube.com

Obtain the pull secret

```
oc get secrets -n apic
```

The pull secret starts with deployer-dockercfg, in our case it was:

```
deployer-dockercfg-6mjp9
```

Create the TLS secret.

Setup the CLI environment, and make sure that helm command works correctly, for example run:

```
helm version --tls
```

and make sure that it has the connectivity to the server:

```
Client: &version.Version{SemVer:"v2.12.3",  
GitCommit:"eecf22f77df5f65c823aacd2dbd30ae6c65f186e", GitTreeState:"clean"}  
  
Server: &version.Version{SemVer:"v2.12.3+icp",  
GitCommit:"34e12adfe271fd157db8f9745affe84c0f603809", GitTreeState:"clean"}
```

Run the following command to create the secret:

```
kubectl create secret generic apic-ent-helm-tls --from-file=cert.pem=$HOME/.helm/cert.pem --from-file=ca.pem=$HOME/.helm/ca.pem --from-file=key.pem=$HOME/.helm/key.pem -n apic
```

where apic-ent-helm-tls is the name of the secret.

[Increase vm.max_map_count](#)

To check and increase vm.max_map_count we would need an *ssh* access to each of the cluster nodes.

The alternative is to create a DaemonSet which will do that for us. Prepare the yaml file with the following content:

```
apiVersion: extensions/v1beta1  
  
kind: DaemonSet  
  
metadata:  
  
labels:  
  
    k8s-app: sysctl-conf  
  
name: sysctl-conf  
  
namespace: kube-system  
  
spec:  
  
template:  
  
    metadata:  
  
        labels:  
  
            k8s-app: sysctl-conf  
  
spec:  
  
    containers:  
  
        - command:  
  
            - sh  
  
            - -c  
  
            - sysctl -w vm.max_map_count=262144 && while true; do sleep 86400;  
done  
  
        image: busybox:1.26.2  
  
        name: sysctl-conf
```

```
resources:

  limits:
    cpu: 10m
    memory: 50Mi

  requests:
    cpu: 10m
    memory: 50Mi

  securityContext:
    privileged: true

  terminationGracePeriodSeconds: 1
```

and run apply it with:

```
oc apply -f sysctl-conf.yaml
```

Storage Class

The block storage class is needed for APIC. You can obtain the class names with

```
oc get storageclass
```

The following class was available on ROKS:

```
managed-nfs-storage (default)    nfs-storage    1d
```

Create an Instance

- Open platform navigator and select API Connect / Add new instance
- Click *Continue*
- Define the helm release name, select apic namespace and the target cluster
 - a. apicdemo
 - b. apic
 - c. local-cluster
- Reduce deployment footprint for test/dev install
 - a. Uncheck Production Deployment
 - b. Select Mode = dev
 - c. Reduce Gateway Cluster Count to 1
 - d. Optionally: Reduce the Cassandra Cluster Size to 1
- Uncheck V5 Compatibility Mode

- Enter the helm TLS secret name and select storage class:

Parameters

To install this chart, additional configuration is needed in Quick start. To customize installation, view and edit All parameters.

Quick start

Required and recommended parameters to view and edit.

Global

Parameters common to operator and all subsystems

Storage Class *

managed-nfs-storage

Operator

Parameters for API Connect operator

Helm TLS Secret *

apic-ent-helm-tls

- Enter the endpoints defined above:

Management Parameters for management subsystem	
Platform API Endpoint * platformapi.apps.ocp.cloudnativekube.com	Consumer API Endpoint * capi.apps.ocp.cloudnativekube.com
Cloud Admin UI Endpoint * adminui.apps.ocp.cloudnativekube.com	API Manager UI Endpoint * mgrui.apps.ocp.cloudnativekube.com
Portal Parameters for portal subsystem	
Portal Director Endpoint * portaldirector.apps.ocp.cloudnativekube.com	Portal Web Endpoint * portalweb.apps.ocp.cloudnativekube.com
Analytics Parameters for analytics subsystem	
Analytics Ingestion Endpoint * analingest.apps.ocp.cloudnativekube.com	Analytics Client Endpoint * analclient.apps.ocp.cloudnativekube.com
Gateway Parameters for gateway subsystem	
API Gateway Endpoint * apigw.apps.ocp.cloudnativekube.com	Gateway Service Endpoint * gwservice.apps.ocp.cloudnativekube.com

- Turn High Performance Peering off

High Performance Peering *
Off

- Lookup the Image name and Docker Image Tags according to the information provided in the Atomic Registry
 - <https://registry-console-default.apps.ocp.cloudnativekube.com/registry>
 - Select Project: apic
 - Click on following images to get the details
 - apic/analytics-operator
 - apic/apiconnect-operator
 - apic/cassandra-operator
 - Note the Pull Repository Server and Tag
- Enter the information in the subsequent form sections

Storage Class *	managed-nfs-storage	Mode	dev
Certs Secret Enter value			
Operator Parameters for API Connect operator			
Architecture	amd64	Image *	apiconnect-operator
Docker Image Tag *	2019-09-17-15-59-ea366f01215e13fcfd1acd1b81e5006cb32483c3d	Docker Image Pull Policy *	IfNotPresent
Helm TLS Secret * apic-ent-helm-tls			
<input checked="" type="checkbox"/> Create Service Account			
Service Account Name apicsvcac			
Storage Class *	managed-nfs-storage	Mode	dev
Certs Secret Enter value			
Operator Parameters for API Connect operator			
Architecture	amd64	Image *	apiconnect-operator
Docker Image Tag *	2019-09-17-15-59-ea366f01215e13fcfd1acd1b81e5006cb32483c3d	Docker Image Pull Policy *	IfNotPresent
Helm TLS Secret * apic-ent-helm-tls			
<input checked="" type="checkbox"/> Create Service Account			
Service Account Name apicsvcac			
Max Memory (GB) 6			
<input type="checkbox"/> v5 Compatibility Mode			
<input type="checkbox"/> Enable TMS			
TMS Peering Storage Size (GB)	10	High Performance Peering *	Off
Image datapower-api-gateway	Image Tag 2018.4.1.7-312001-release-prod		
Monitoring Image k8s-datapower-monitor	Monitoring Image Tag 2018.4.1-16-37918a2		

- Click on **Install**, the confirmation message will appear

- You can check the status of the pods also with the command:

```
oc get pods -n apic
```

- When deployment is completed, all pods must be in Running or Completed state.

SMTP Server

In order to configure the API Connect, we need a SMTP server. If we don't have one, we can run the Mailhog, a fake SMTP server ready for any Kubernetes environment.

- Make sure you are properly connected/logged into the server with oc, cloudctl and helm
- Install Mailhog with:

```
helm install --name mailhog stable/mailhog --tls
```

- The service is of ClusterIP type, so in order to read mails, we must change it to the NodePort, or create Route, or simply port-forward the pod. For example, if the pod name (obtained with `oc get pods -n apic`) is `mailhog-55c8c548dc-dphpj` we can run:

```
kubectl port-forward mailhog-55c8c548dc-dphpj 8025:8025 -n apic
```

and then access mails from the local browser on `http://127.0.0.1:8025`

Configuring API Connect

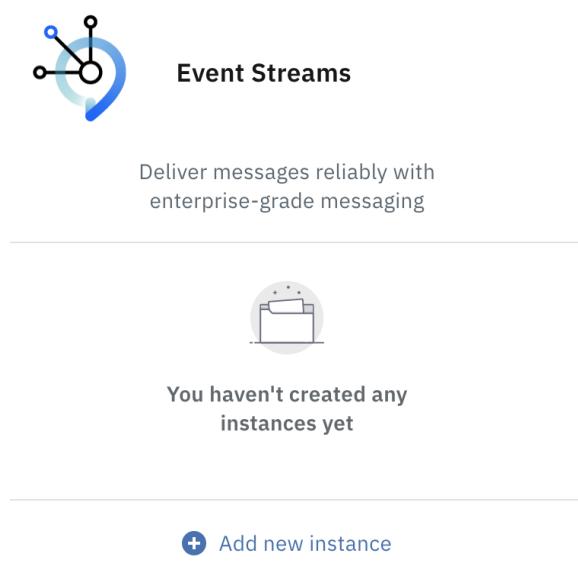
- Open the Cloud Management Console using the previously defined endpoint, in our case it was: <https://adminui.apps.ocp.cloudnativekube.com/admin/>
- Select IBM Common Services user registry, default username and password in this case are admin/ #####
- Under **Resources/Notifications** define the SMTP server
- For our Mailhog server enter ClusterIP address and port according to the installation response
 - IP: 172.30.233.85
 - Port: 1025
- And select the SMPT server defined under resources:
- Start with the Topology configuration
- Register service:
- Start with the Gateway, select the version that you defined under the Helm release properties when you started creating the instance. In our case it was DataPower API Gateway
- Give some name to the service (e.g. **gateway1**) enter the **endpoints** and click on **Save**
- The confirmation message should appear on the top right of the screen
- Click on *Register service* again and select Analytics
- Give some name to the service, enter Management endpoint (the one that you defined for **analytics client**) and click **Save**
- The confirmation message should appear on the top right of the screen
- Repeat the same with portal:
 - Name: portal1
 - Endpoint: portaldirector.apps.ocp.cloudnativekube.com
 - Website URL: portalweb.apps.ocp.cloudnativekube.com
- The confirmation appears again

- Click on Associate Analytics Service to associate analytics with the gateway
- Select the analytics service
- Click on Provider organizations and add new organization
- Give some name to the organization
 - Title: DemoOrg
- Define the owner
 - Select IBM Common Services user registry
 - Username: admin
- After you submit the organization will appear on the list
- Navigate to the API Manager, in our case the endpoint was:
<https://mgrui.apps.ocp.cloudnativekube.com/manager/demoorg/>
- Login as the owner (defined in the previous step), the API Manager page should open
- Select Gateway Services and assign the gateway to the service
- Navigate to the catalog and then settings
 - Click Portal and click Create

With that, your API Connect instance is ready for usage!

Creating an Event Streams Environment

Open the Integration Navigator and select Create a New Instance from the Event Streams tile



Give the new instance a name, select the EventStreams Namespace and select 'Local Cluster'

Configuration
IBM Event Streams based on Apache Kafka. Edit these parameters for configuration.

Helm release name <small>ⓘ</small>	ev1
Target namespace *	eventstreams
Target Cluster *	local-cluster
License *	
<input checked="" type="checkbox"/> I have read and agreed to the License agreement	

Create a new Cert secret name (Can be anything) and use the command 'OC get secret' from the EventStreams namespace or project tp determine the image pull secret.

Choose an external hostname for the instance (we're using the default proxy entry point here)

Configuration
IBM Event Streams based on Apache Kafka. Edit these parameters for configuration.

Helm release name <small>ⓘ</small>	ev1
Target namespace *	eventstreams
Target Cluster *	local-cluster
License *	
<input checked="" type="checkbox"/> I have read and agreed to the License agreement	

Enter the Docker image registry and leave all other fields as default.

Generate Certificate for Security *

Certificate Secret Name *	Docker image registry *
evsecret	docker-registry.default.svc:5000/eventstreams
Image pull secret *	Image pull policy *
default-dockercfg-7p95g	IfNotPresent
File system group ID	Architecture *
Enter value	amd64 platforms
Pod to pod encryption *	Kubernetes internal DNS domain name *
Disabled	cluster.local

Leave all other fields at their default values and Click install.

Wait for all the pods to come up. Hint use 'oc get pods' in the Eventstreams namespace.

```
Peters-MacBook-Pro-4:ICP4i peterajessup$ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
ev1-es-access-controller-deploy-5d5fd45477-6gmr5	2/2	Running	0	4m
ev1-es-access-controller-deploy-5d5fd45477-6nk4l	2/2	Running	0	4m
ev1-es-collector-deploy-8676b89c84-db4ww	2/2	Running	0	4m
ev1-es-elastic-sts-0	2/2	Running	0	4m
ev1-es-elastic-sts-1	2/2	Running	0	4m
ev1-es-indexmgr-deploy-8694764d4d-h8mj5	2/2	Running	0	4m
ev1-es-kafka-sts-0	5/5	Running	2	4m
ev1-es-kafka-sts-1	5/5	Running	1	4m
ev1-es-kafka-sts-2	5/5	Running	0	4m
ev1-es-proxy-deploy-79db8f6bf5-77bgs	1/1	Running	0	3m
ev1-es-proxy-deploy-79db8f6bf5-sfk87	1/1	Running	0	3m
ev1-es-proxy-deploy-79db8f6bf5-w8d6q	1/1	Running	0	4m
ev1-es-rest-deploy-7569668f8b-xjx4v	3/3	Running	0	4m
ev1-es-rest-producer-deploy-7dbd55b4f5-75n54	1/1	Running	0	4m
ev1-es-rest-proxy-deploy-54f9cc7f89-n8xv2	1/1	Running	0	4m
ev1-es-schemaregistry-sts-0	2/2	Running	0	2m
ev1-es-ui-deploy-c4b89946d-wrbt4	2/2	Running	0	4m
ev1-es-ui-oauth2-client-reg-52wpk	0/1	Completed	0	4m

ev1-ibm-es-zookeeper-sts-0	2/2	Running	0	4m
ev1-ibm-es-zookeeper-sts-1	2/2	Running	0	4m
ev1-ibm-es-zookeeper-sts-2	2/2	Running	0	4m

Open the Event Streams Instance from the instance name link in the Event Streams Tile on the Integration Home page:

eventstreams

[ev1](#)

⋮

Configure a new topic called ivt using the ‘New Topic Button ;follow the wizard on the interface using all defaults.

ev1

Name	Replicas	Partitions
ivt	3	1

Click on ‘Connect to this cluster’ from the EventStreams UI Home Page and click on the Pem Certificate to download to your local machine. Save this away.

Cluster connection

[Connect a client](#)

[Sample code](#)

[Geo-replication](#)

To connect an application or tool to this cluster, you will need the address of a bootstrap server, a certificate and an API key.

Bootstrap server

Your application or tool will make its initial connection to the cluster using the bootstrap server.

icp-proxy.apps.ocp.cloudnativekube.com:30817



Certificates

A certificate is required by your Kafka clients to connect securely to this cluster.

Java truststore
Use this for a Java client

 **Truststore password:** password

PEM certificate
Use this for anything else



API key

To connect securely to Event Streams, your application or tool needs an API key with permission to access the cluster and resources such as topics.



Name your application

Provide a name for your application

Connecting your tool or application will generate a service ID using your application name, a service policy and an API key in IBM Cloud Private. Additional API keys can be generated in the IBM Cloud Private Cluster Management Console or CLI.

[IBM Cloud Private Cluster Management Console](#)

Take note of the bootstrap server host and port and use the API Key wizard to create a new application name and api key. Give the application a name and select ‘Produce and Consume’. Provide a topic name and leave the ‘Consumer Groups’ button set to ‘All’. Click ‘Generate’ and copy the API Key and save away.

API key

To connect securely to Event Streams, your application or tool needs an API key with permission to access the cluster and resources such as topics.



Name your application

What do you want it to do?

-
-
-
-

API key

To connect securely to Event Streams, your application or tool needs an API key with permission to access the cluster and resources such as topics.



Start again 

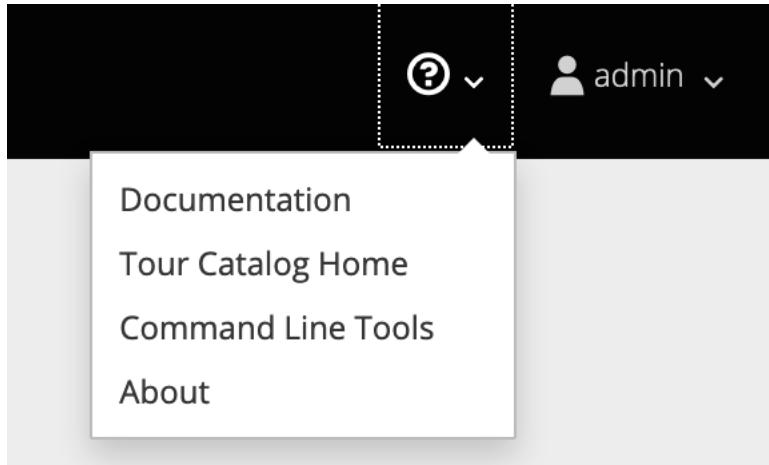
Which consumer group? All

Perform the End to End IVT test

Log into openshift via the command line.

Go to the registry console URL

<https://registry-console-default.apps.ocp.cloudnativekube.com/registry>



Select Command line tools and follow the instruction here:

Command Line Tools

With the OpenShift command line interface (CLI), you can create applications and manage OpenShift projects from a terminal. You can download the `oc` client tool using the links below. For more information about downloading and installing it, please refer to the [Get Started with the CLI](#) documentation.

[Download `oc`](#):

[Latest Release](#) ↗

After downloading and installing it, you can start by logging in. You are currently logged into this console as **admin**. If you want to log into the CLI using the same session token:

`oc login https://master.ocp.cloudnativekube.com:8443 --token=<hidden>`



Login via cloudctl:

```
sudo cloudctl login -a https://icp-console.apps.ocp.cloudnativekube.com -u admin -p ##### -n default
```

Once logged in, copy the pem certificate saved from the Event Streams configuration into an empty directory. Stay in this directory for all further commands.

Copy the pem file to a file named “truststoreCert-mykey.crt”

Create a new file called serverconf.yaml and place the following snippet into it and save it.

-----Begin snippet-----

ResourceManagers:

JVM:

```
truststoreType: 'JKS'
```

```
truststoreFile: '/home/aceuser/ace-server/truststore.jks'
```

```
truststorePass: 'setdbparms::truststore'
```

Defaults:

```
policyProject: 'DefaultPolicies'
```

Policies:

HTTPSCConnector: 'HTTPS'

-----End Snippet-----

Create a new file called setdbparms.txt and add the following snippet, replacing <api_key> with the api key you saved from the Event Streams instance.

-----Begin snippet-----

```
kafka::KAFKA token <api_key>
setdbparms::truststore dummy password
```

-----End Snippet-----

Create a new file called truststorePassword.txt and place the text ‘password’ on the first line of this file.

-----Begin snippet-----

password

-----End Snippet-----

Run the generateSecrets.sh defined earlier in this document by creating this file in the same directory as the files created earlier in this section:

Chmod a+x generateSecrets.sh

./generateSecrets.sh ivt-secret

This creates a secret called ivt-secret for use in the HELM chart.

Locate the ivt5.bar and ivt.zip (Project Interchange file) located in the IVT Artifacts directory under here: <https://ibm.ent.box.com/folder/90729333577>

We will use this bar file in the ACE deployment.

queue manager called ‘QM1’, HELM release name ‘mqtest’, with the following objects (as defined earlier in this document)

Channel: IVT.SVRCONN

Security: Disabled

Queues: REREQ.Q, RT.REP.Q

Queue Manager Name : QM1

Internal to Cluster Host Name: mqtest-ibm-mq.mq.svc

Port : 1414

This ACE deployment relies on an Event Streams deployment as defined in an earlier section, with a topic called IVT.

You can change these parameters by modifying the ivt.zip in the ACE toolkit and re-deploying the bar file after re-building.

To deploy the ace server we will use the HEL command line and the following script called installACE-MQC.sh

-----Begin snippet-----

```
if [ "$2" == "" ]
then
echo "Please provide a integration server name as the second argument"
echo "Usage: installACE-MQC.sh <release_name> <Integration_server_name> <content Server URL>
[<secret_name>]"
exit 1
fi
if [ "$4" == "" ]
then
SECRET_ARG="nil"
else
SECRET_ARG=$4
fi
if [ "$3" == "" ]
then
echo "Please provide the content server URL as the 3rd argument"
echo "Usage: installACE-MQC.sh <release_name> <Integration_server_name> <content Server URL>
[<secret_name>]"
exit 1
fi
echo Secret=$SECRET_ARG
sudo helm install local-charts/ibm-ace-server-icp4i-prod --name=$1 --set license=accept --tls --set
aceonly.replicaCount=1 --set integrationServer.configurationSecret=$SECRET_ARG --set
integrationServer.truststoreCertNames=mykey --set contentServerURL=$3 --set
imageType=acemqclient --set image.acemq=docker-registry.default.svc:5000/ace/ibm-ace-mqclient-
server-prod:11.0.0.5.1 --set image.pullPolicy=IfNotPresent --set image.configurator=docker-
registry.default.svc:5000/ace/ibm-ace-icp-configurator-prod:11.0.0.5.1 --set
integrationServer.name=$2 --set integrationServer.configurationSecret=$SECRET_ARG --set
service.iP=icp-proxy.apps.ocp.cloudnativekube.com --set persistence.enabled=true --set
persistence.useDynamicProvisioning=true --set dataPVC.name=data --set
dataPVC.storageClassName=managed-nfs-storage --set productionDeployment=false --set
metrics.enabled=false
```

-----End Snippet-----

Note the arguments to this script, and ensure you have the correct content server URL as defined when deploying the bar in the GUI for a new ace instance – here's an example:

Add server

You will now configure and install a Helm release to deploy to the server. It is important to copy the Content URL and select the current Namespace.

Content URL:

<https://acedashboard-ibm-ace-dashboard-icp4i-prod:3443/v1/directories/ivt5?0ba4bd6a-8d44-4411-98bd-c5aaeb72e8ef>



Namespace:

ace

If the integration server requires any configuration to be applied then you will need to use the following download to provide the configuration prior to install. Refer to the README.md inside the download on how to create the required secrets:

[Download configuration package](#)

Create the script file on your local machine, and execute the script as follows:

-----Begin Snippet-----

Chmod a+x installACE-MQC.sh

./installACE-MQC.sh aceivt IS1 https://acedashboard-ibm-ace-dashboard-icp4i-prod:3443/v1/directories/ivt5?0ba4bd6a-8d44-4411-98bd-c5aaeb72e8ef ivt-secret

-----End Snippet-----

This will deploy an MQ Client only ACE instance with the supplied bar file and secret.

After deployment check the pods:

-----Begin Snippet-----

oc get pods

NAME	READY	STATUS	RESTARTS	AGE
acedashboard-ibm-ace-dashboard-icp4i-prod-546b78c6c6-xp5qr	2/2	Running	0	4d
aceivt-ibm-ace-server-icp4i-prod-5f87ccfbf7-q29l9	1/1	Running	0	9m

-----End Snippet-----

Check the Event Streams instance for a connected consumer group called 'ivtConsumer'.

ev1

The screenshot shows the 'Consumer groups' tab in the Confluent Cloud interface. A search bar at the top contains the placeholder 'Type to search consumer groups'. Below it is a table with two sections. The first section, 'Consumer group ID', lists 'ivtConsumer' with 1 active member, 0 unconsumed partitions, and a green 'STABLE' status. The second section, 'Topic', shows details for the 'ivt' topic with partition 0 having consumer ID 'consumer-1-4a5d629e-2b43-4dd6-ae67-5207595e2e5a', current offset 6, and log end offset 6.

Test the ACE instance by clicking on the instance tile on the ACE dashboard and drilling down into the REST API which is deployed:

The screenshot shows the ACE dashboard with an instance tile for 'IVTRest API'. The tile has a green 'Started' status bar at the bottom. Below the tile, the text 'IVTRest' and 'API' is displayed. The main area shows the REST API documentation with the base URL 'http://icp-proxy.apps.ocp.cloudnativekube.com:30327/ivtrest/v1' and the OpenAPI document URL 'http://icp-proxy.apps.ocp.cloudnativekube.com:30327/ivtrest/v1/swagger.json'. A POST request is shown in the API browser with the endpoint '/doiIvt'.

The REST API base URL *port number* will be different for each deployment.

Open up a REST testing tool and enter the following URL (from the example URL above) into the tool's URL:

<http://icp-proxy.apps.ocp.cloudnativekube.com:30327/ivtrest/v1/dolvt>

Use the following as the body for the POST operation:

```
{  
  "name": "anyname",
```

```

"phone":"123545454",
"email": "anyname@host.com"
}

```

Test the POST.

A json payload should be returned in ALL UpperCase characters.

```
{"name":"ANYNAME","phone":"123545454","email":"ANYNAME@HOST.COM"}
```

Check the console of the queue manager (or other tool) for the response message in the RTREQ.Q and the RTREP.Q

Check that the Consumer Group Offset in the Event Streams UI for the group ivtConsumer has increased by one.

Now we'll create an API in API Connect to proxy this API.

Log into the designer UI of the API Connect Instance created by following the instructions in the document 'Deploying API Connect on ICP4i' located here:

<https://ibm.ent.box.com/folder/90729333577>

Click on Develop AIS and Products and click 'Add' and select 'API'

Develop

APIs and Products Add

Click on 'From Target Service' and click 'Next'

Fill in the table as shown (note the target service URL port number for you will be different)

Create API from Target Service

Info	
Enter details of this API	
Title	IVT
Name	ivt
Version	1.0.0
Base path (optional)	/ivt
Description (optional)	
Target Service URL	
Enter the URL for the target service you would like to proxy	
Target service URL	<input type="text" value="http://icp-proxy.apps.ocp.cloudnativekube.com:30327/vtrest/v1/doit"/>
Cancel Next	

Click Next and accept the defaults, click 'Next' and then 'Edit API'

Click on ‘Definitions’ and click ‘Add’

The screenshot shows the 'Definitions' section of a web application. On the left, there's a sidebar with links: API Setup, Security Definitions, Security, Paths, Definitions (which is highlighted in blue), Properties, and Target Services. The main area has a title 'Definitions' with a subtitle 'API request and response payload structures are composed using OpenAPI schema definitions.' Below this is a table with columns 'NAME', 'TYPE', and 'DESCRIPTION'. A single row is present, showing a small icon of a person with a gear, followed by the text 'No items found'.

In the name field type ‘ivtPayload’ then click ‘Add’ adjacent to the properties section.

This screenshot shows the 'ivtPayload' definition creation form. It has sections for 'Name' (containing 'ivtPayload'), 'Type' (set to 'object'), and 'Description (optional)'. Below these is a 'Properties' section with a table. The table has columns: PROPERTY NAME, PROPERTY TYPE, PROPERTY EXAMPLE, PROPERTY DESCRIPTION, and DELETE. One row is shown with 'name' as the property name, 'string' as the type, and a dropdown menu currently set to 'string'. There are checkboxes for 'Additional properties' and 'Edit schema' at the bottom of the table.

Enter the field name and select type of ‘String’.

Add two more properties called ‘phone’ and ‘email’ of type ‘String’.

Click ‘Save’ at the top of the form.

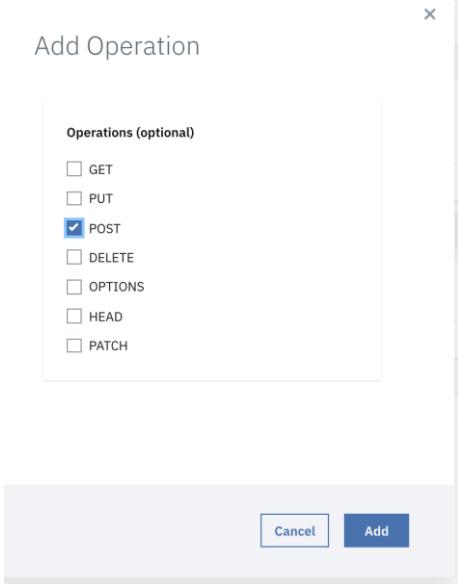
This screenshot shows the 'ivtPayload' definition creation form after adding three properties. The 'Properties' table now contains three rows: 'name' (type string), 'phone' (type string), and 'email' (type string). The 'Edit schema' button is visible at the bottom of the table. At the top of the page, there are tabs for 'Design' (which is selected), 'Source', and 'Assembly'. To the right, there are status indicators for 'Stopped', 'No Errors', and a 'Save' button.

Select ‘Paths’ from the top level menu and delete the path labelled ‘/’ using the dot menu on the right hand side.

Click ‘Add’

In the Path Name field type ‘/dolv’. The forward slash is required.

Adjacent to the operations section click ‘Add’



Click 'Save'

Click on the newly Created path, then on the Operation labelled 'POST'

Path											
Paths identify the REST resources exposed by the API. An operation combines a path with an HTTP verb, parameters, and definitions for requests and responses. Learn more											
Path name <input type="text" value="/doIvt"/>											
Path Parameters <table border="1"> <thead> <tr> <th>REQUIRED</th> <th>NAME</th> <th>LOCATED_IN</th> <th>TYPE</th> <th>DESCRIPTION</th> <th>DELETE</th> </tr> </thead> </table>						REQUIRED	NAME	LOCATED_IN	TYPE	DESCRIPTION	DELETE
REQUIRED	NAME	LOCATED_IN	TYPE	DESCRIPTION	DELETE						
Operations <table border="1"> <thead> <tr> <th>NAME</th> <th>... POST</th> </tr> </thead> </table>						NAME	... POST				
NAME	... POST										

Click 'Add' Adjacent to the Parameters Section.

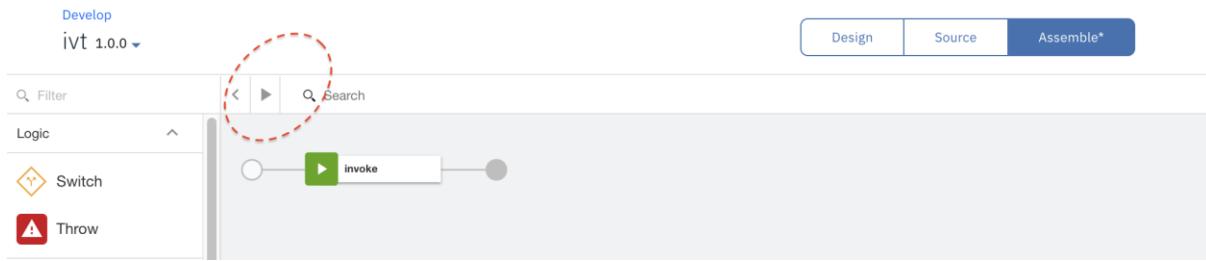
Parameters													
REQUIRED	NAME	LOCATED IN	TYPE	DESCRIPTION	DELETE								
Response <table border="1"> <thead> <tr> <th>STATUS CODE</th> <th>SCHEMA</th> <th>DESCRIPTION</th> <th>DELETE</th> </tr> </thead> <tbody> <tr> <td>200</td> <td><input type="text" value="string"/></td> <td><input type="text" value="SUCCESS"/></td> <td></td> </tr> </tbody> </table>						STATUS CODE	SCHEMA	DESCRIPTION	DELETE	200	<input type="text" value="string"/>	<input type="text" value="SUCCESS"/>	
STATUS CODE	SCHEMA	DESCRIPTION	DELETE										
200	<input type="text" value="string"/>	<input type="text" value="SUCCESS"/>											

Fill out the form as follows:

Parameters					
REQUIRED	NAME	LOCATED IN	TYPE		
<input type="checkbox"/>	ivtObj	<input type="text" value="body"/>	<input type="text" value="ivtPayload"/>		

Click 'Save' at the top of the page.

Click on 'Assemble' at the page top center to bring up the Assembly.



Click on the grey arrow highlighted above.

Click on ‘Activate API’ and wait for it to be published.

You’ll see a result similar to:

Catalog: sandbox

Product: ivt-auto-product

Plan: Default Plan

Application: sandbox-test-app

Click on the drop down showing ‘product_api_operation’. Select ‘post/dolv’t’.

Operation
Choose an operation to invoke:
<code>product_api_operation</code>
<code>post /dolv't</code>
Identification
clientId
<code>3de03c7e7d73dcb2a03949f37ea5f2d9</code>
parameters
ivtObj
Show schema Generate

Click on ‘Generate’ to generate a test message.

Test ×

Setup

Catalog: sandbox

Product: none selected

Plan: none selected

Application: sandbox-test-app

Activate API

Click on ‘Invoke’ at the base of the form.

You should see something similar to:

Response

Status code:

200 OK

Response time:

3835ms

Headers:

content-type: application/json;charset=UTF-8

Body:

```
{  
  "name": "WILLIAM KENNEDY",  
  "phone": "(957) 847-4665",  
  "email": "IVIVI@AWA.AL"  
}
```

As before check the MQ queues and consumer group offset in Event Streams.

This concludes the end-to-end IVT test.

Appendix A – setting up the PSP

```
C:\SHARE\ProductsTechnology\IBM Cloud Pak for Integration\SetUp\yamls>dir  
Volume in drive C is Windows  
Volume Serial Number is 3CBA-3B12  
  
Directory of C:\SHARE\ProductsTechnology\IBM Cloud Pak for Integration\SetUp\yamls  
  
18/10/2019 12:44 PM <DIR> .  
18/10/2019 12:44 PM <DIR> ..  
18/10/2019 12:44 PM 836 ibm-anyuid-psp.yaml
```

```
C:\SHARE\ProductsTechnology\IBM Cloud Pak for Integration\SetUp\yamls>\openshift\oc config current-context  
ace/master-ocp-cloudnativekube-com:8443/admin
```

```
C:\SHARE\ProductsTechnology\IBM Cloud Pak for Integration\SetUp\yamls>\openshift\oc create -f ibm-anyuid-psp.yaml  
podsecuritypolicy.policy/ibm-anyuid-psp created
```

```
C:\SHARE\ProductsTechnology\IBM Cloud Pak for Integration\SetUp\yamls>\openshift\oc get psp  
NAME      PRIV   CAPS  
SGROUP    SUPGROUP  READONLYROOTFS  VOLUMES  
ibm-anyuid-psp  false    SETPCAP,AUDIT_WRITE,CHOWN,NET_RAW,DAC_OVERRIDE,FOWNER,FSETID,KILL,SETUID,SETGID,NET_BIND_SERVICE,SYS_CHROOT,SETFCAP  
unAsAny   RunAsAny  false        configMap,emptyDir,projected,secret,downwardAPI,persistentVolumeClaim  
tekton-pipelines  false  
ustRunAs  MustRunAs false        emptyDir,configMap,secret
```

Appendix B – RoundTrip Batch file

You may need to modify the path to your own java installation.

```
"c:\Program Files\ibm\MQ\java\jre\bin\java" -classpath  
.\\com.ibm.mq.allclient.jar;\\.jms.jar;\\.jndi.jar;\\.imaclientjms.jar;\\.jms.jar;\\.GUIRoundTrip.jar;org.dyno  
.visual.swing.jar;org.dyno.visual.swing.layouts.jar com.ibm.mq.testclient.RoundTripF
```

Appendix C – MQ and ACE DNS naming

For the ace/mq services it follows the pattern described here:

<https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/>

The dns name is <service_name>.<namespace>."svc"

So for the ace release "acefin-es" we lookup the service name as

"acefin-es-ibm-ace-server-icp4i-prod"

Using the same convention the dns name is

"acefin-es-ibm-ace-server-icp4i-prod.ace.svc"

For An MQ of Helm release name mqvt3 - has a QMGR=QM3

The MQ connection information for clients inside the cluster is as follows:

mqivt3-ibm-mq.mq.svc:1414

In order to connect to that queue manager. Created a SENDER channel on another queue manager in the cluster. The connname will be

CONNNAME = mqivt3-ibm-mq.mq.svc(1414)

Therefore the naming convention is **helmRelease-ibm-mq.mq.svc**

DNS name for MQ only in mq namespace Example :Helm Release = repqm1p

DNS = repqm1p-ibm-mq.mq.svc

DNS name for ACE/MQ in ace namespace Example: Helm Release = eaiqm1

DNS = eaiqm1p-ibm-ace-server-icp4i-prod.ace.svc

Appendix D – ACE GUI – avoid login

Clicking on the webUI from Helm release

<http://icp-console.apps.ocp.cloudnativekube.com:31867/login?redirectedFrom=%2F>

results in a

IBM App Connect

Login to your account

Username

Password

Log in

To avoid this change the URL to use the **proxy** entry point not the console

<http://icp-proxy.apps.ocp.cloudnativekube.com:31867/>

Appendix E – Useful references

<https://developer.ibm.com/messaging/2019/08/05/configure-and-install-ibm-mq-queue-managers-on-ibm-cloud-pak-for-integration/>

Appendix F – connecting in an “Off RHOS” Queue manager

IDCQM1 – Definitions

*** Place holder definitions for an OFF RHOS cluster queue manager connecting in via DQM
*** A queue manager called OFFRHOSQM with the opposite definitions will be required on-premises

* Receiver channel for OFF RHOS cluster queue manager connecting in via DQM

```
DEFINE CHANNEL('TO.QMGR_NAME.OFFRHOSQM') +
  CHLTYPE(RECEIVER) +
  TRPTYPE(TCP) +
  REPLACE
```

* Local transmission queue to OFFRHOSQM

```
DEFINE QLOCAL('OFFRHOSQM') +
  DEFPSIST(YES) +
  DEFBIND(NOTFIXED) +
  DISTL(NO) +
  MAXDEPTH(5000) +
  USAGE(XMITQ)+
  REPLACE
```

* Sender channel to OFF RHOS cluster queue manager connecting via DQM

```
DEFINE CHANNEL('TO.OFFRHOSQM.QMGR_NAME') +
  CHLTYPE(SENDER) +
  TRPTYPE(TCP) +
  CONNAME('ToBeDefined(9999)')+
  XMITQ('OFFRHOSQM')+
  REPLACE
```

```
DEFINE QREMOTE('TEST.IIB.Q1.REP') +
  RQMNAME('OFFRHOSQM') +
  RNAME('TEST.IIB.Q1.REP') +
  CLUSTER('EAICLUSTER.ENV_ID') +
  REPLACE
```

Note the following screen captures need to be updated to reflect the naming of objects above that are configured into the IDCQM1 queue manager on RHOS

Sending messages from Off RHOS Qmgr to EAIQM1 via IDCQM1(Gateway)

Channel receiver on IDCQM1

The screenshot shows the IBM MQ Explorer interface. On the left, there is a tree view of queue managers and their components. A context menu is open over a channel named 'TO.IDCQM1.FROM.MyAPP'. The menu path 'Edit -> Properties' is highlighted. The properties dialog for this channel is displayed on the right, showing the 'General' tab with the following settings:

Channel name:	TO.IDCQM1.FROM.MyAPP
Type:	Receiver
Description:	[empty]
Transmission protocol:	TCP
Overall channel status:	Running

Create a local windows off rhos queue manager called MyAPPLQMGR with default settings

The screenshot shows the IBM MQ Explorer interface. On the left, under 'IBM MQ' > 'Queue Managers', a new queue manager 'MyAPPLQMGR' is being created. On the right, the 'Queues' table shows the newly created queue 'TEST.IIB.Q1' and the existing local queue 'IDCQM1'.

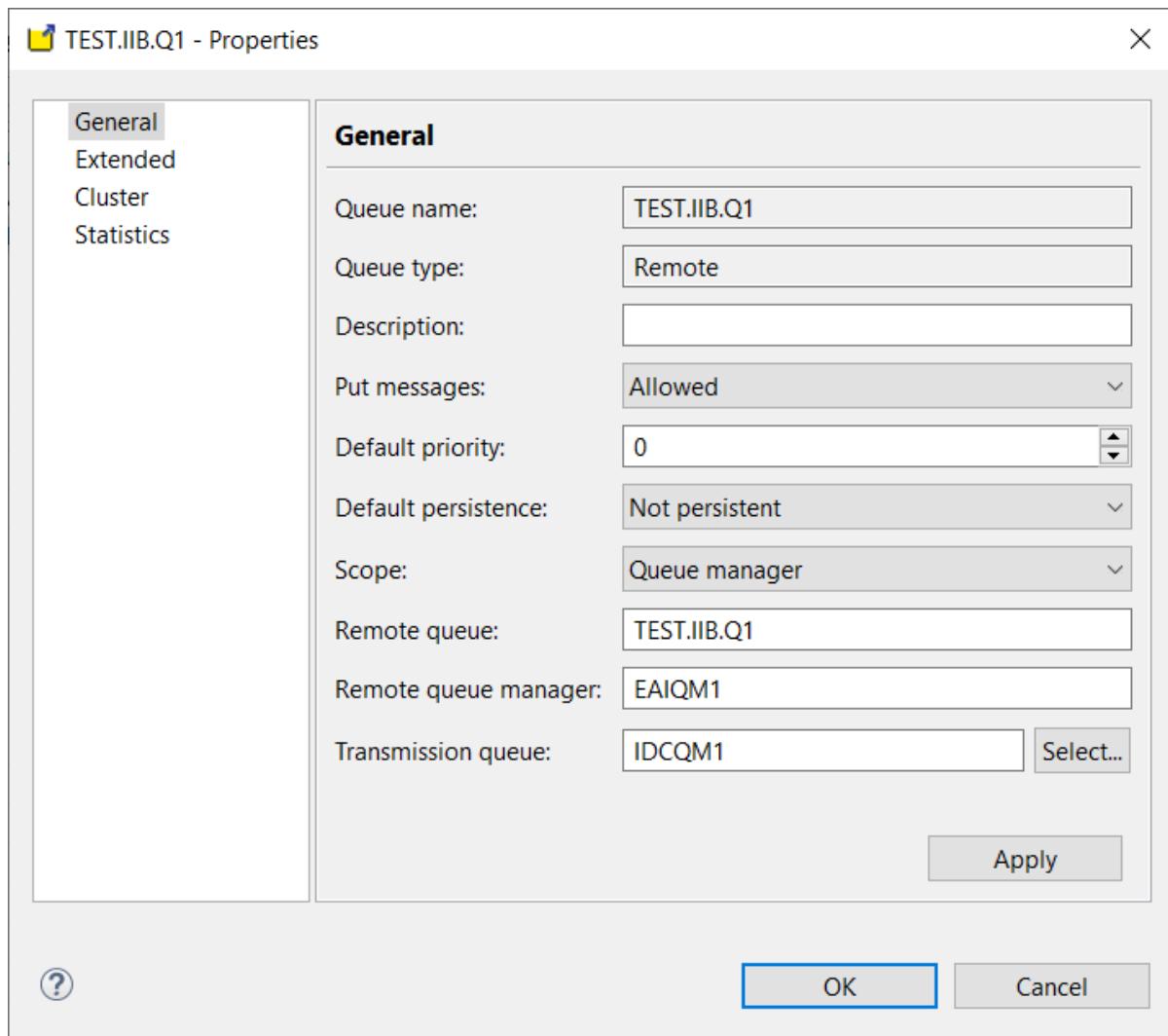
Queue name	Queue type	Open input count	Open output count	Current queue depth	Put messages	Get messages	Remote queue	Remote queue manager
IDCQM1	Local	1	1	0	Allowed	Allowed		
TEST.IIB.Q1	Remote						TEST.IIB.Q1	EAIQM1

Create a transmission queue for IDCQM1 on OFFRHOS queue manager

The screenshot shows the 'IDCQM1 - Properties' dialog. The left sidebar has tabs for General, Extended, Cluster, Triggering, Events, Storage, and Statistics. The 'General' tab is selected. The configuration fields are as follows:

Queue name:	IDCQM1
Queue type:	Local
Description:	[empty]
Put messages:	Allowed
Get messages:	Allowed
Default priority:	0
Default persistence:	Not persistent
Scope:	Queue manager
Usage:	Transmission

Create a remote queue definition on MyAPPLQMGR to send messages to EAIQM1 via IDCQM1



On off RHOS QM create a Sender channel to IDCQM1(matches the name of the receiver on IDCQM1)

TO.IDCQM1.FROM.MyAPP Sender Running icp-proxy.apps.ocp.cloudnativekube.com(30419) IDCQM1

TO.IDCQM1.FROM.MyAPP - Properties

General

Channel name: TO.IDCQM1.FROM.MyAPP

Type: Sender

Description:

Transmission protocol: TCP

Connection name: *icp-proxy.apps.ocp.cloudnativekube.com(30419)

Transmission queue: *IDCQM1

Local communication address:

Overall channel status: Running

Put a test message from MyAPPLQMGR

Queue Managers

- EAIQM1 on 'icp-console.apps.ocp.cloudnativekube.com(30798)'
 - EAIQM2 on 'icp-console.apps.ocp.cloudnativekube.com(32588)'
- IDCQM1 on 'icp-console.apps.ocp.cloudnativekube.com(30419)'
 - IDCQM2 on 'icp-proxy.apps.ocp.cloudnativekube.com(31971)'
- MyAPPQMGR
 - Queues
 - Topics

Queues

Filter: Standard for Queues	
Queue name	Queue type
IDCQM1	Local
TEST.IIB.Q1	Remote

TEST.IIB.Q1	Remote	Allowed	TEST
<p>Put test message</p> <p>Put message to:</p> <p>Queue manager: MyAPPQMGR</p> <p>Queue: TEST.IIB.Q1</p> <p>Message data: Hello EAIQM1 on RHOS the other side of IDCQM1 from MyAPPLQMGR on windows</p> <p> ⓘ The queue which will receive the test message is on this computer. The message will be put directly on the queue.</p> <p>Put message Close</p>			

Check TEST.IIB.Q1 on EAIQM1

Queue Managers

- EAIQM1 on 'icp-console.apps.ocp.cloudnativekube.com(30798)'
 - Queues
 - Topics
 - Subscriptions
 - Channels
 - Listeners

Queues

Filter: Standard for Queues				
Queue name	Queue type	Open input count	Open output count	Current queue depth
RTREP.Q	Local	0	0	0
RTREQ.Q	Local	1	0	0
TEST.IIB.Q1	Local	0	1	1

TEST.IIB.Q1 Local 0 1 Allowed Allowed

Message browser

Queue Manager Name: EAIQM1
Queue Name: TEST.IIB.Q1

Position	Put date/time	User identifier	Put application name	Format	Total length
1	06/11/2019 4:36:29 PM	davidarnold	MQ Explorer 9.1.3	MQSTR	72

Message 1 - Properties

General Report Context Identifiers Segmentation Data

Scheme: Standard for M
Last updated: 16:37:22 ()
All available messages

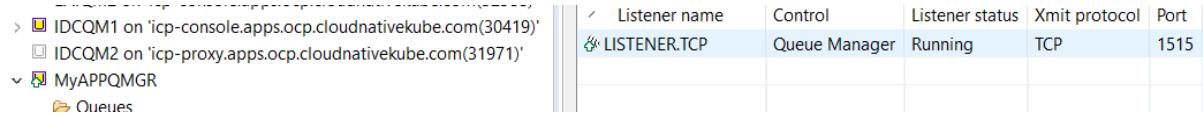
Data

Total length: 72
Data length: 72
Format: MQSTR
Coded character set identifier: 1208
Encoding: 546
Message data: Hello EAIQM1 on RHOS the other side of ID
Message data bytes:

00000	48	65	6C	6C	6F	20	45	4	^
00010	52	48	4F	53	20	74	68	€	
00020	69	64	65	20	6F	66	20	4	
00030	6F	6D	20	4D	79	41	50	£	
00040	20	77	69	6E	64	6F	77	7	

Receiving messages to Off RHOS Qmgr from EAIQM1 via IDCQM1(Gateway)

Listener on off RHOS system



	Listener name	Control	Listener status	Xmit protocol	Port
✓ LISTENER.TCP	Queue Manager	Running	TCP	1515	

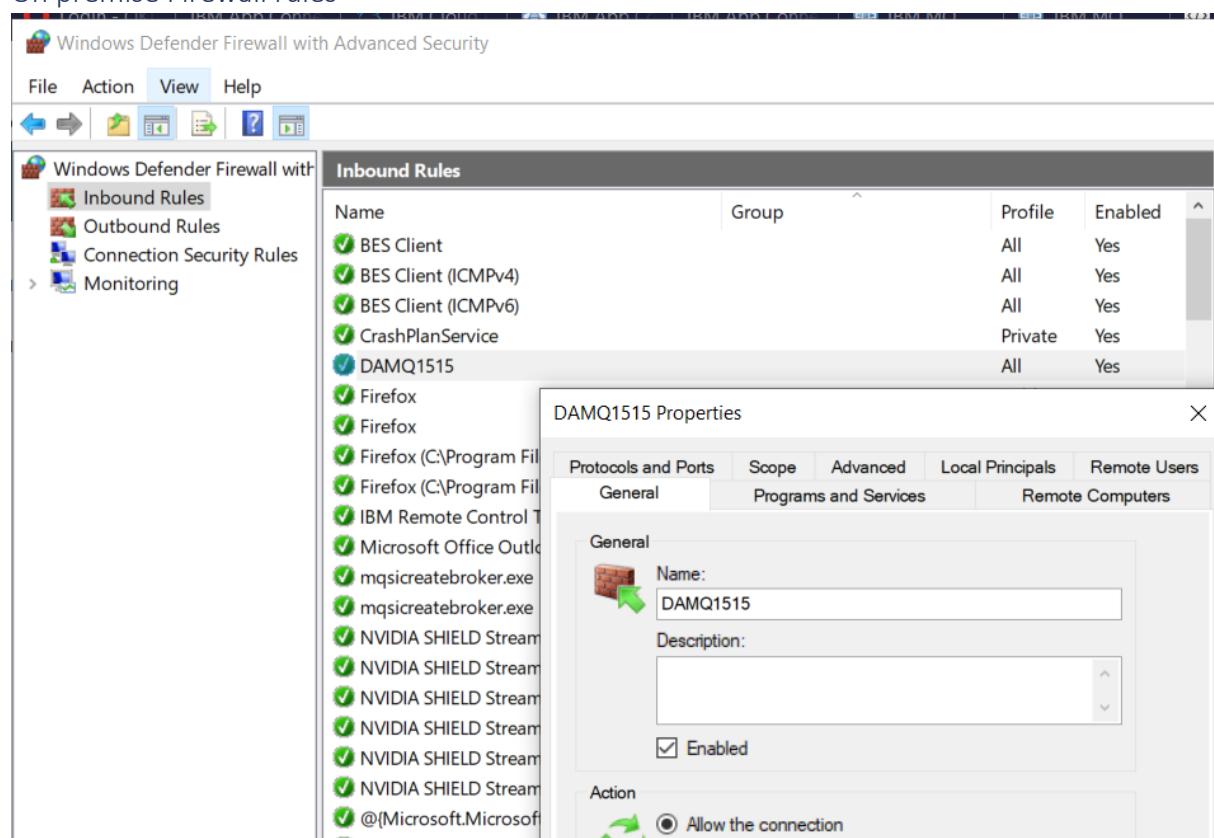
Queues

IPCONFIG on target off box system

```
Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . : au.ibm.com
IPv6 Address . . . . . : 2620:1f7:381f:844::31:367
Link-local IPv6 Address . . . . . : fe80::541e:ece4:e55:42a%9
IPv4 Address . . . . . : 9.198.78.76
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
Link-local IPv6 Address . . . . . : fe80::541e:ece4:e55:42a%9
IPv4 Address . . . . . : 169.254.1.1
Subnet Mask . . . . . : 255.255.0.0
Default Gateway . . . . . :
```

On premise Firewall rules



Windows Defender Firewall with Advanced Security

Inbound Rules

Name	Group	Profile	Enabled
BES Client		All	Yes
BES Client (ICMPv4)		All	Yes
BES Client (ICMPv6)		All	Yes
CrashPlanService		Private	Yes
DAMQ1515		All	Yes
Firefox			
Firefox			
Firefox (C:\Program File			
Firefox (C:\Program Fil			
IBM Remote Control T			
Microsoft Office Outlo			
mqsicreatebroker.exe			
mqsicreatebroker.exe			
NVIDIA SHIELD Stream			
@(Microsoft.Microso			

DAMQ1515 Properties

Protocols and Ports

General	Scope	Advanced	Local Principals	Remote Users
Programs and Services				
Remote Computers				

General

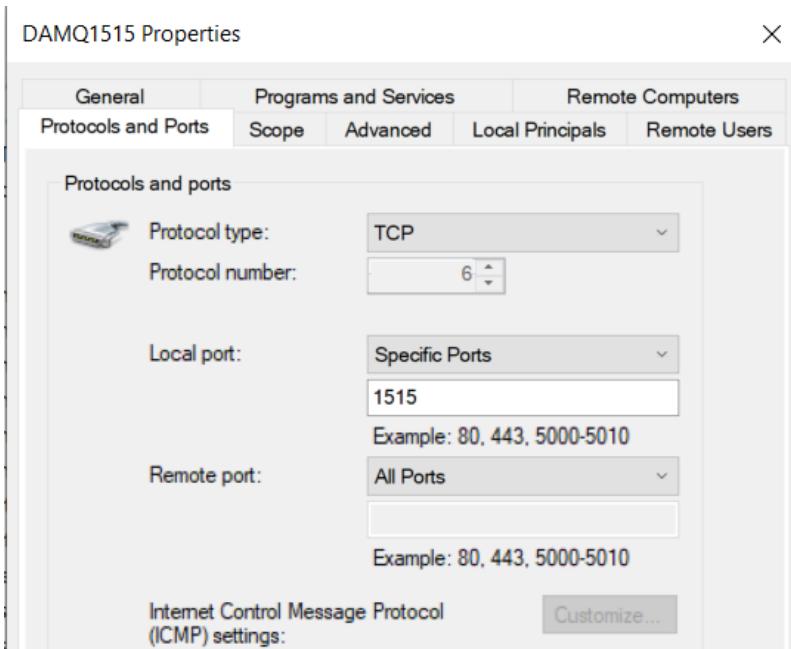
Name: DAMQ1515

Description:

Enabled

Action

Allow the connection



Secure Gateway on IBM cloud

You will need to use a service like the IBM Secure Gateway on IBM Public Cloud to provide a public IP address for your queue manager on premises.

Resource list /

Secure Gateway-wp

Location: Sydney Org: davearno@au1.ibm.com Space: IBMHybridIntDemo

myGW

0.00 MB Total Inbound

1.0
0.5
0

1:04 1:44 2:24 3:04 3:44

Destinations

DAT480MQ1515

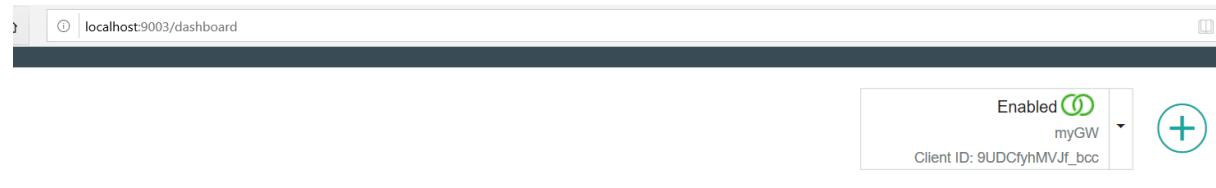
Destination ID: 9UDClyhMVJf_Ypn0Y
Cloud Host: Port: cap-au-sg-prd-01.securegateway.appdomain.cloud:15092
Resource Host: Port: 9.198.78.76:1515
Created at: 11/7/2019, 12:17:06 PM
Last modified at: 11/7/2019, 12:17:25 PM
Security: Protocol: TCP
[Download Authentication Files](#)

Edit Disable Delete

4 9:44 10:24 11:04 11:44 12:24

Clients (1)

Secure client on premises



Secure Gateway Client

⊕ Notifications (0)

The dashboard features three main sections:

- Access Control List**: Represented by a lock icon.
- View Logs**: Represented by a document icon.
- Connection Info**: Represented by a cloud icon.



Access Control List Manager

Access Control List entries determine what the client is allowed to do.

Use the tables below to manually input the individual host:port entries. For bulk entries, use the "Import" page. For more information on the Access Control List, view the [readme file](#) in the [GitHub repository](#).

Allow access [\(i\)](#)

Resource Hostname	:	Port	+
<input type="checkbox"/> cap-au-sg-prd-01.securegateway.appdomain.cloud:15092			

localhost:9003/connection-info

Back

Enabled myGW Client ID: 9UDCfyhMVJf_bcc

Connection Information

0.0 MB Total Inbound	0.0 MB Total Outbound	1 Active Connections	14 Total Connections
----------------------	-----------------------	----------------------	----------------------

myGW Details		Index	Host:Port	Inbound	Outbound
Client ID	9UDCfyhMVJf_bcc	14	9.198.78.76:1515	0.0 MB	0.0 MB
Gateway ID	myGW				

MQ configuration

Receiver channel on MyAPPQMGR (the “off RHOS” queue manager)

Queue Managers

- EAIQM1 on 'tcp-console.apps.ocp.cloudnativekube.com(30798)'
- EAIQM2 on 'tcp-console.apps.ocp.cloudnativekube.com(32588)'
- IDCQM1 on 'tcp-console.apps.ocp.cloudnativekube.com(30419)'
- IDCQM2 on 'tcp-proxy.apps.ocp.cloudnativekube.com(31971)'
- MyAPPQMGR
 - Queues
 - Topics
 - Subscriptions
 - Channels
 - Listeners
 - Services
 - Process Definitions
 - Namelists
 - Authentication Information
 - Communication Information
- QM1 on 'tcp-console.apps.ocp.cloudnativekube.com(32581)'
- REPMQM1 on 'tcp-proxy.apps.ocp.cloudnativekube.com(31198)'
- REPMQM2 on 'tcp-proxy.apps.ocp.cloudnativekube.com(30051)'

Queue Manager IDCQM1

Connection QuickView:

Connection status	Disconnected
Connection type	Local
Connection name	

TO.MYAPPQMGR - Properties

General	Channel name: TO.MYAPPQMGR
Extended	Type: Receiver
MCA	Description:
Exits	Transmission protocol: TCP
Message retry	Overall channel status: Running
SSL	
Statistics	

Transmission queue and sender channel on IDCQM1

EAIQM1 on 'tcp-console.apps.ocp.cloudnativekube.com(32300)

- ✓ IDCQM1 on 'tcp-console.apps.ocp.cloudnativekube.com(30419)'
 - Queues
 - Topics
 - Subscriptions
 - Channels
 - Listeners
 - Services
 - Process Definition
 - Namelists
 - Authentication Info
 - Communication Info
 - Security Policies

MyAPPQMGR

- REPMQM1 on 'tcp-pr...
- REPMQM2 on 'tcp-pr...
- REPQM1 on 'tcp-pr...
- REPQM2 on 'tcp-pr...

Queue Manager Cluster:

- JMS Administered Obj...
- Managed File Transfer
- Service Definition Repo...

New Local Queue

Change properties

Change the properties of the new Local Queue

General	General
Extended	Queue name: MyAPPQMGR
Cluster	Queue type: Local
Triggering	Description:
Events	Put messages: Allowed
Storage	Get messages: Allowed
Statistics	Default priority: 0
	Default persistence: Not persistent
	Scope: Queue manager
	Usage: Transmission

Sender channel

IDCQM1 on 'tcp-console.apps.ocp.cloudnativekube.com(30419)'

- Queues
- Topics
- Subscriptions
- Channels
- Listeners
- Services
- Process Definitions
- Namelists
- Authentication Info
- Communication Info
- Security Policies

MyAPPQMGR

- Queues
- Topics
- Subscriptions
- Channels
- Listeners
- Services
- Process Definitions
- Namelists
- Authentication Info
- Communication Info

QM1 on 'tcp-console.ap...

Channel name	Channel type	Overall channel status	Conn name
IDCQM1	Server-connection	Inactive	
IVT.SVRCONN	Server-connection	Running	
TO.IDCQM1	Cluster-receiver	Inactive	idcqm1p-ibm-mq...
TO.IDCQM1.FROM.MyAPP	Receiver	Running	
TO.M.IDCQM1	Cluster-receiver	Inactive	idcqm1p-ibm-mq...
TO.MYAPPQMGR	Sender	Running	cap-au-sg-prd-01

TO.MYAPPQMGR - Properties

General

General	General
Extended	Channel name: TO.MYAPPQMGR
MCA	Type: Sender
Exits	Description:
LU6.2	Transmission protocol: TCP
Retry	Connection name: * cap-au-sg-prd-01.securegateway.appdomain.cloud(15092)
SSL	Transmission queue: * MyAPPQMGR
Statistics	Local communication address:
	Overall channel status: Running

Start the channel

IDCQM1 on 'tcp-console.apps.ocp.cloudnativekube.com(30419)'

- Queues
- Topics
- Subscriptions
- Channels
- Client Connections
- Channel Authentication Records

Channel name	Channel type
IDCQM1	Server-connection
IVT.SVRCONN	Server-connection
TO.IDCQM1	Cluster-receiver
TO.IDCQM1.FROM.MyAPP	Receiver
TO.M.IDCQM1	Cluster-receiver
TO.MyAPP.FROM.IDCQM1	Sender

Create a local queue on “Off RHOS” Queue Manager

The screenshot shows the IBM MQ Explorer interface. On the left, there is a tree view of objects under 'MyAPPQMGR'. Some objects are expanded to show their sub-components. A 'Queues' node is selected. On the right, there is a table showing queue statistics:

Queue name	Queue type	Open input count	Open output count	Curr
IDCQM1	Local	1	1	0
TEST.IIB.Q1	Remote			

A 'New Local Queue' dialog box is open in the foreground. It contains the following fields:

- Create a Local Queue**: The title of the dialog.
- Name:** A text input field containing **TEST.IIB.Q1.REP**.
- Select an existing object from which to copy the attributes for the new object.**: A dropdown menu currently showing **SYSTEM.DEFAULT.LOCAL.QUEUE**.
- Select...**: A button next to the dropdown menu.
- When this wizard completes, another wizard can be started automatically to create a matching object.**: A descriptive text.
- Start wizard to create a matching JMS Queue**: A checkbox.

Create a Remote Queue Definition on IDCQM1 RHOS routing Queue Manager and share in the EAICLUSTER



New Remote Queue Definition

Change properties

Change the properties of the new Remote Queue Definition

General
Extended
Cluster

General

Queue name:	TEST.IIB.Q1.REP
Queue type:	Remote
Description:	Reply to Off RHOS Queue Manager
Put messages:	Allowed
Default priority:	0
Default persistence:	Not persistent
Scope:	Queue manager
Remote queue:	TEST.IIB.Q1.REP
Remote queue manager:	MyAPPQMGR



< Back

Next >

Finish

Cancel



New Remote Queue Definition



Change properties

Change the properties of the new Remote Queue Definition

General
Extended
Cluster

Cluster

Sharing in Clusters

- Not shared in a cluster
- Shared in cluster
- Shared in a list of clusters

EAICLUSTER.000

Default bind type:

CLWL queue rank:

CLWL queue priority:



< Back

Next >

Finish

Cancel

Sending a message from RHOS EAIQM1 to “Off RHOS” Queue Manager

Use RFHUTILC to connect to EAIQM1

Put a message to TEST.IIB.Q1.REP

Queue Managers

- EAIQM1 on 'tcp-console.apps.ocp.cloudnativekube.com(30798)'
 - Queues
 - Topics
 - Subscriptions
 - Channels
 - Listeners
 - Services
 - Process Definitions
 - Namelists
 - Authentication Information
 - Communication Information
 - Security Policies
- EAIQM2 on 'tcp-console.apps.ocp.cloudnativekube.com(32588)'
- IDCQM1 on 'tcp-console.apps.ocp.cloudnativekube.com(30419)'
- IDCQM2 on 'tcp-proxy.apps.ocp.cloudnativekube.com(31971)'

MyAPPQMR

- Queues
- Topics
- Subscriptions
- Channels
- Listeners
- Services
- Process Definitions
- Namelists
- Authentication Information
- Communication Information

QM1 on 'tcp-console.apps.ocp.cloudnativekube.com(32581)'

Main Data MQMD PS Usr Prop RFH PubSub pscr jms

Queue Manager Name (to connect to)
IVT.SVRCONN/TCP/tcp-console.apps.ocp.cloudnativekube.com(30798)

Queue Name
TEST.IIB.Q1.REP

Remote Queue Manager Name (remote queues only)

Selector

Read Q Write Q Browse Q Start Browse Browse Next Browse Prev End Br

File Name
C:\SHARE\SERVICES\DATools\RFHUTILv7\samples\ibmsale1.txt Data Size 55

Open File Save File Clear Data Clear All Load Names Set Conn Id

COBOL Copy Book File Name

Put/Get
 New
 Get I
 Get L
 Get I
 Get L

13.49.52 Message sent to TEST.IIB.Q1.REP length=55
13.49.24 Connected to IVT.SVRCONN/TCP/tcp-console.apps.ocp.cloudnative

Main Data MQMD PS Usr Prop RFH PubSub

Message Data (55) from C:\SHARE\SERVICES\DATools\RFHUTILv7\sample

```
00000000 <IBM><Price>00010000</Price><Shares>
00000032 res>1000</Shares></IBM>
```

Browse the TEST.IIB.Q1.REP queue

QM1 on 'tcp-console.apps.ocp.cloudnativekube.com(32581)'
 REPQM1 on 'tcp-proxy.apps.ocp.cloudnativekube.com(31198)'
 REPQM2 on 'tcp-console.apps.ocp.cloudnativekube.com(30854)'
 REPQM1 on 'tcp-proxy.apps.ocp.cloudnativekube.com(30153)'
 REPQM2 on 'tcp-proxy.apps.ocp.cloudnativekube.com(30718)'

Queue Manager Clusters

JMS Administered Objects

Managed File Transfer

Service Definition Repositories

Queue name	Queue type	Open input count	Open output count	Current queue depth	Put messages
IDCQM1	Local	1	1	0	Allowed
TEST.IIB.Q1	Remote				Allowed
TEST.IIB.Q1.REP	Local	0	0	1	Allowed

Message browser

Queue Manager Name: MyAPPQMR
Queue Name: TEST.IIB.Q1.REP

Position	Put date/time	User identifier	Put application name	Format	Total length
1	07/11/2019 1:49:52 PM		Tools\RFHUTILv7\rfhutilc.exe		55

Message 1 - Properties

General	Data
Report	Total length: 55
Context	Data length: 55
Identifiers	Format:
Segmentation	Coded character set identifier: 437
Data	Encoding: 546
	Message data: <IBM><Price>00010000</Price><Shares>

Deploy and Echo flow on EAIQM1 for TEST.IIB.Q1 and TEST.IIB.Q1.REP



```
DECLARE Delay EXTERNAL int 0;
DECLARE CopyMsgIDtoCorrID EXTERNAL boolean true;
DECLARE returnValue BOOLEAN;

CREATE COMPUTE MODULE RoundTripEcho_MsgIDCorIDandDelay
    CREATE FUNCTION Main() RETURNS BOOLEAN
    BEGIN
        SET returnValue = SLEEP(Delay);
        CALL CopyMessageHeaders();
        Set OutputRoot.BLOB.BLOB = x'4441313233' || InputRoot.BLOB.BLOB;

        If(CopyMsgIDtoCorrID)
        THEN Set OutputRoot.MQMD.CorrelId = InputRoot.MQMD.MsgId;
        END IF;

        RETURN TRUE;
    END;

CREATE PROCEDURE CopyMessageHeaders() BEGIN
    DECLARE I INTEGER 1;
    DECLARE J INTEGER;
    SET J = CARDINALITY(InputRoot.*[]);
    WHILE I < J DO
        SET OutputRoot.*[I] = InputRoot.*[I];
        SET I = I + 1;
    END WHILE;
END;
END MODULE;
```

Appendix G – Building custom MQ image for client access

Load ICP4i MQAdvanced Server image to local docker repos

Windows (C:) > SHARE > ProductsTechnology > MQv9-9.1 > ICP4iMQAdv-image > images

<input type="checkbox"/> Name	Date modified	Type
ibm-mqadvanced-server-integration_9.1.3.0-r4-amd64.tar	9/11/2019 1:14 AM	GZ File
ibm-mq-oidc-registration_2.2.0-amd64.tar	9/11/2019 1:14 AM	GZ File
icp4i-od-agent_1.0.0-amd64.tar	9/11/2019 1:14 AM	GZ File
icp4i-od-collector_1.0.0-amd64.tar	9/11/2019 1:14 AM	GZ File

docker load < ibm-mqadvanced-server-integration_9.1.3.0-r4-amd64.tar.gz

```
Directory of C:\SHARE\ProductsTechnology\MQv9-9.1\ICP4iMQAdv-image\images

13/12/2019 07:02 PM <DIR> .
13/12/2019 07:02 PM <DIR> ..
09/11/2019 01:14 AM 161,053,184 ibm-mq-oidc-registration_2.2.0-amd64.tar.gz
09/11/2019 01:14 AM 942,485,504 ibm-mqadvanced-server-integration_9.1.3.0-r4-amd64.tar.gz
09/11/2019 01:14 AM 310,161,408 icp4i-od-agent_1.0.0-amd64.tar.gz
09/11/2019 01:14 AM 321,895,424 icp4i-od-collector_1.0.0-amd64.tar.gz
        4 File(s) 1,735,595,520 bytes
        2 Dir(s) 217,878,286,336 bytes free

C:\SHARE\ProductsTechnology\MQv9-9.1\ICP4iMQAdv-image\images>docker load < ibm-mqadvanced-server-integration_9.1.3.0-r4-amd64.tar.gz
cd28e8b3d42a: Loading layer [=====>] 7.68kB/7.68kB
84eaf14cef7b: Loading layer [=====>] 6.144kB/6.144kB
dc025946c48b: Loading layer [=====>] 806MB/806MB
084a5de0f8d3: Loading layer [=====>] 2.56kB/2.56kB
a0cf9052d6e2: Loading layer [=====>] 12.39MB/12.39MB
8c20517487b4: Loading layer [=====>] 5.876MB/5.876MB
0199fa08053a: Loading layer [=====>] 10.24kB/10.24kB
7f6264a97cf1: Loading layer [=====>] 11.78kB/11.78kB
1355ce09db14: Loading layer [=====>] 3.584kB/3.584kB
f9d44c67deb4: Loading layer [=====>] 18.28MB/18.28MB
c4b465bf4f6a: Loading layer [=====>] 7.68kB/7.68kB
79163b2a33c5: Loading layer [=====>] 2.146MB/2.146MB
2c903f0b7155: Loading layer [=====>] 3.072kB/3.072kB
87ba76dd2431: Loading layer [=====>] 3.584kB/3.584kB
1c321fdcb962: Loading layer [=====>] 7.168kB/7.168kB
f561b484f28b: Loading layer [=====>] 2.152MB/2.152MB
2b68e45b087e: Loading layer [=====>] 7.168kB/7.168kB
a10f6d83153f: Loading layer [=====>] 21.5kB/21.5kB
1cfdf819e9f0e: Loading layer [=====>] 3.237MB/3.237MB
364112d1df0e: Loading layer [=====>] 261.6kB/261.6kB
9f4789d1e0ed: Loading layer [=====>] 3.072kB/3.072kB
07cdf8094239: Loading layer [=====>] 3.497MB/3.497MB
6eb6a773b24: Loading layer [=====>] 3.738MB/3.738MB
ac7598f58291: Loading layer [=====>] 3.738MB/3.738MB
Loaded image: ibm-mqadvanced-server-integration:9.1.3.0-r4-amd64

C:\SHARE\ProductsTechnology\MQv9-9.1\ICP4iMQAdv-image\images>
```

(C:) > Users > DAVIDARNOLD > myDocker > ibm-mqadvanced-server-ivt

<input type="checkbox"/> Name	Date modified	Type
dockerfile	12/12/2019 3:10 PM	File
ivt.mqsc	12/12/2019 4:55 PM	MQSC File

Docker file

```
FROM ibm-mqadvanced-server-integration:9.1.3.0-r4-amd64
```

```
USER mqm
```

```
COPY ivt.mqsc /etc/mqm/
```

```
IVT.mqsc
```

```
DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCONN)  
SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)  
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE)  
ADOPTCTX(YES)  
SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')  
REFRESH SECURITY TYPE(CONNAUTH)
```

OR – if this is based off a non ICP4i mq image

* Connection authentication

```
DEFINE AUTHINFO('ORG.AUTHINFO') AUTHTYPE(IDPWOS) CHCKCLNT(REQDADM)  
CHCKLOCL(OPTIONAL) ADOPTCTX(YES) REPLACE
```

```
ALTER QMGR CONNAUTH('ORG.AUTHINFO')
```

```
REFRESH SECURITY(*) TYPE(CONNAUTH)
```

* Channels (Application + Admin)

```
DEFINE CHANNEL('IVT.SVRCONN') CHLTYPE(SVRCONN) REPLACE
```

```
DEFINE CHANNEL('ORG.APP.SVRCONN') CHLTYPE(SVRCONN) MCAUSER('app') REPLACE
```

* Channel authentication rules

```
SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS) DESCRIPTOR('Back-stop rule -  
Blocks everyone') ACTION(REPLACE)
```

```
SET CHLAUTH('ORG.APP.SVRCONN') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(CHANNEL)  
CHCKCLNT(REQUIRED) DESCRIPTOR('Allows connection via APP channel') ACTION(REPLACE)
```

```
SET CHLAUTH('IVT.SVRCONN') TYPE(BLOCKUSER) USERLIST('nobody') DESCRIPTOR('Allows admins on  
ADMIN channel') ACTION(REPLACE)
```

```
SET CHLAUTH('IVT.SVRCONN') TYPE(USERMAP) CLNTUSER('admin') USERSRC(CHANNEL)  
DESCRIPTOR('Allows admin user to connect via ADMIN channel') ACTION(REPLACE)
```

* Authority records

```
SET AUTHREC GROUP('mqclient') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
```

```

SET AUTHREC PROFILE('ORG.**') GROUP('mqclient') OBJTYPE(QUEUE)
AUTHADD(BROWSE,GET,INQ,PUT)

SET AUTHREC PROFILE('ORG.**') GROUP('mqclient') OBJTYPE(TOPIC) AUTHADD(PUB,SUB)

```

Docker build

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ace11002mqc91intms1	1.0	ec39bb5df84f	2 days ago	1.59GB
ibmcom/mq	latest	268baf40303f	7 days ago	927MB
ibmcom/ace-mq	latest	0f5a883d8a95	4 weeks ago	2.51GB
ibmcom/ace-mqclient	latest	d71cee6dfd5f	4 weeks ago	1.94GB
ibmcom/ace	latest	d33c5a966d7b	4 weeks ago	1.63GB
ibm-mqadvanced-server-integration	9.1.3.0-r4-amd64	ee41039b9552	5 weeks ago	932MB

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt> docker build -t davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64 .

```

C:\Users\DAVIDARNOLD\myDocker\mqsrvmqsc>docker build -t davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64 .
Sending build context to Docker daemon 4.096kB
Step 1/3 : FROM ibmcom/mq
--> 268baf40303f
Step 2/3 : USER mqm
--> Using cache
--> f976d4712a79
Step 3/3 : COPY ivt.mqsc /etc/mqm/
--> Using cache
--> 0c686c880b92
Successfully built 0c686c880b92
Successfully tagged davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64

```

Local test

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
davexacom/ibm-mqadvanced-server-ivt	9.1.3.0-r4-amd64	0c686c880b92	3 days ago	927MB

docker run -e LICENSE=accept -e MQ_QMGR_NAME=QM1 -P davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker run -e LICENSE=accept -e MQ_QMGR_NAME=QM1 -P davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64

2019-12-13T08:29:20.120Z CPU architecture: amd64

2019-12-13T08:29:20.121Z Linux kernel version: 4.9.184-linuxkit

2019-12-13T08:29:20.121Z Container runtime: docker

2019-12-13T08:29:20.121Z Base image: Red Hat Enterprise Linux Server 7.7 (Maipo)

Break

Starting MQSC for queue manager QM1.

1 : DEFINE CHANNEL(IVT.SVRCCONN) CHLTYPE(SVRCCONN)

AMQ8014I: IBM MQ channel created.

2 : SET CHLAUTH(IVT.SVRCCONN) TYPE(BLOCKUSER) USERLIST(nobody)

AMQ8877I: IBM MQ channel authentication record set.

3 : ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS)
CHKCLNT(NONE) ADOPTCTX(YES)

AMQ8567I: IBM MQ authentication information changed.

4 : SET CHLAUTH(IVT.SVRCCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')

AMQ8877I: IBM MQ channel authentication record set.

5 : REFRESH SECURITY TYPE(CONNAUTH)

AMQ8560I: IBM MQ security cache refreshed.

5 MQSC commands read.

No commands have a syntax error.

All valid MQSC commands were processed.2019-12-13T08:29:22.125Z

Metrics are disabled

2019-12-13T09:09:17.684Z AMQ8024I: IBM MQ channel initiator started.

2019-12-13T09:09:17.706Z AMQ5806I: Queued Publish/Subscribe Daemon started for queue manager QM1.

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS		NAMES		

```
b5e2c8034d8c    davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64  
"runmqinteg...r..." 26 seconds ago   Up 25 seconds   0.0.0.0:32770->1414/tcp,  
0.0.0.0:32769->9157/tcp, 0.0.0.0:32768->9443/tcp  relaxed_lamport
```

Connect MQ Explorer

The screenshot shows the 'Queue Managers' configuration screen in IBM MQ Explorer. The top navigation bar includes 'IBM MQ' and 'Queue Managers'. A prominent 'Add Queue Manager' button is at the top left. Below it, a section titled 'Select the queue manager and connection method' instructs the user to identify the queue manager to add and choose the connection method. The 'Queue manager name:' field contains 'QM1'. The 'Connection details' section includes fields for 'Host name or IP address:' (localhost), 'Port number:' (32770), and 'Server-connection channel:' (IVT.SVRCONN). In the 'User identification' section, 'Enable user identification' is checked, while 'User identification compatibility mode' is unchecked. The 'Userid:' field contains 'mqm', and the 'Password' section shows 'No password' selected.

IBM MQ

Queue Managers

Add Queue Manager

Select the queue manager and connection method

Identify the queue manager to add and choose the connection method to use

Queue manager name: QM1

Connection details

Host name or IP address: localhost

Port number: 32770

Server-connection channel: IVT.SVRCONN

Queue manager name: QM1

Enable user identification

User identification compatibility mode

Userid: mqm

Password

No password

MQ Explorer - Navigator

- IBM MQ
 - Queue Managers
 - EAIQM1 on 'tcp-console.apps.ocp.cloudnativekube.com(30798)'
 - EAIQM2 on 'tcp-console.apps.ocp.cloudnativekube.com(32588)'
 - IDCQM1 on 'tcp-console.apps.ocp.cloudnativekube.com(30419)'
 - IDCQM2 on 'tcp-proxy.apps.ocp.cloudnativekube.com(31971)'
 - QM1 on 'localhost(32805)'
 - Queues
 - Topics
 - Subscriptions
 - Channels
 - Client Connections
 - Channel Authentication Records

MQ Explorer - Content

Channels

Channel name	Channel type	Overall channel status
IVT.SVRCONN	Server-connection	Running

Access via RUNMQSC client

```
SET MQSERVER=IVT.SVRCONN/TCP/localhost(32770)
```

manual

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>runmqsc -c QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager QM1.

def ql(DA.Q)
  1 : def ql(DA.Q)
AMQ8006I: IBM MQ queue created.
end
  2 : end
2 command responses received.
```

scripted

1q-npwd.mqsc - Notepad

```
File Edit Format View Help
define ql(DA) replace
```

```
C:\temp>SET MQSERVER=IVT.SVRCONN/TCP/localhost(32805)

C:\temp>runmqsc -c QM1 < 1q-npwd.mqsc
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager QM1.

      1 : define ql(DA) replace
AMQ8006I: IBM MQ queue created.
2 command responses received.
```

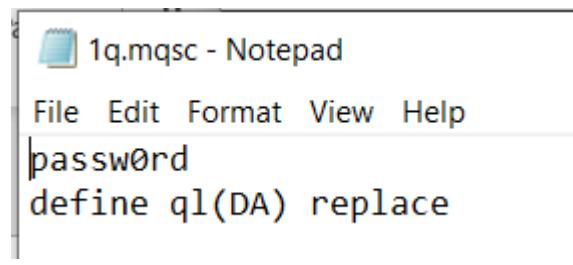
Note if you are using a queue manager where the username is password protected

Manual running of mqsc

```
C:\temp>runmqsc -c -u admin QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Enter password:
*****
Starting MQSC for queue manager QM1.

end
      1 : end
0 command responses received.
```

If -u is on the command and we are piping in, runmqsc assumes the first line is the password.



```
1q.mqsc - Notepad
File Edit Format View Help
passw0rd
define ql(DA) replace
```

```
C:\temp>runmqsc -c -u admin QM1 < 1q.mqsc
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Enter password:
Starting MQSC for queue manager QM1.

      1 : define ql(DA) replace
AMQ8006I: IBM MQ queue created.
2 command responses received.
```

Push the image to Dockerhub

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
davexacom/ibm-mqadvanced-server-ivt	9.1.3.0-r4-amd64	2237fea38967	17 minutes ago	932MB

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
davexacom/ibm-mqadvanced-server-ivt	9.1.3.0-r4-amd64	e269ad126613	5 minutes ago	932MB

<https://hub.docker.com/repositories>

The screenshot shows the Docker Hub interface. At the top, there's a search bar and navigation links for Explore, Repositories, Organizations, Get Help, and a user account dropdown. Below that, a dropdown menu shows 'davexacom' and a search bar for repository names. A 'Create Repository +' button is visible. The main area displays three repository cards for 'davexacom':

- ace11002mqc91intms1**: Updated a year ago, 0 stars, 13 downloads, PUBLIC.
- ace11002mqc91soe**: Updated a year ago, 0 stars, 43 downloads, PUBLIC.
- ace11002mqc91intms2**: Updated a year ago, 0 stars, 12 downloads, PUBLIC.

docker push davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker push davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64
```

The push refers to repository [docker.io/davexacom/ibm-mqadvanced-server-ivt]

```
0d8459f27ecc: Pushing [=====] 3.072kB
ac7598f58291: Pushing [=====] 1.378MB/3.735MB
6eb6a773bf24: Pushing [=====] 1.378MB/3.735MB
07cdf8094239: Pushing [=====] 561.2kB/3.49MB
9f4789d1e0ed: Pushing [=====] 3.072kB
```

364112d1df0e: Waiting

cd28e8b3d42a: Pushed

2705f79af6db: Layer already exists

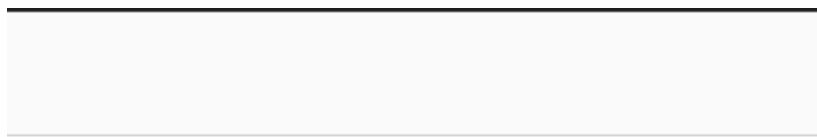
```
ce459cc53899: Layer already exists 9.1.3.0-r4-
amd64: digest: sha256:040e8e0aca3307369b10cce8411f97e147a9a92bb7514619a16c2a78093c085f
size: 5963
```

<https://hub.docker.com/repository/docker/davexacom/ibm-mqadvanced-server-ivt>

The screenshot shows the Docker Hub interface for the repository `davexacom / ibm-mqadvanced-server-ivt`. At the top, there's a header bar with links for ICP4i on CPSystem, Login - RHOS-OKD-IB..., ICOnRHOS-KD-IBM C..., and ICP4ionRHOS-OKD-IB... A banner at the top right encourages users to try the two-factor authentication beta. Below the header is a search bar and navigation tabs for General, Tags, Builds, Timeline, Collaborators, Webhooks, and Settings. The General tab is selected. The main content area displays the repository details: a globe icon, the repository name `davexacom / ibm-mqadvanced-server-ivt`, a note that it has no description, and a timestamp indicating it was last pushed "a few seconds ago".

<https://registry-console-default.apps.ocp.cloudnativekube.com/registry#/images/mq>

Images			New image stream
Name	Tags	Repository	
<code>mq/ibm-mq-oidc-registration</code>	<code>2.1.0-amd64</code>	<code>docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration</code>	
<code>mq/ibm-mqadvanced-server-integration</code>	<code>9.1.3.0-r1-amd64</code>	<code>docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-integration</code>	



⊕ New image stream

Repository

Docker.io/davexacom/ibm-mqadvanced-server-ivt

Create new image stream / or change image stream

Change image stream

Name ibm-mqadvanced-server-ivt

Project mq

Populate

Pull from docker.io/davexacom/ibm-mqadvanced-server-ivt

Tags 9.1.3.0-r4-amd64

Remote registry is insecure

mq/ibm-mqadvanced-server-ivt 1.0-amd64 9.1.3.0-r4-amd64 docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt

Image stream Tags

Access Policy **Images may only be pulled by specific users or groups**

Pulling repository docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt

Image count 2

To push an image to this image stream:

```
$ sudo docker tag myimage docker-registry-default.apps.ocp.cloudnativekube.com/mq/ibm-mqadvanced-server-ivt:tag
$ sudo docker push docker-registry-default.apps.ocp.cloudnativekube.com/mq/ibm-mqadvanced-server-ivt:tag
```

Pulling repository to use in helm charts

docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt

Add new ICP4i instance of MQ using custom image

<https://icp-proxy.apps.ocp.cloudnativekube.com/integration>

The screenshot shows a list of MQ instances under the 'mq' namespace. The instances listed are 'idcab1' and 'mqtest'. There are three dots indicating more instances. At the bottom left is a blue button labeled '+ Add new instance'.

Add a queue manager based messaging capability

You'll be deploying [IBM MQ](#) to provide this capability. Please work with your IBM Cloud Private cluster administrator to ensure the following dependencies are satisfied:

- a unique [namespace](#) must exist for the sole use of IBM MQ. Multiple instances can be deployed in the same namespace. This namespace must allow deployments that require a [security context constraint](#) of type : **ibm-anyuid-scc**.
- a [storage class](#) or [persistent volume](#) needs to be provided for persistent storage

Close

Continue

ibm-mqadvanced-server-integration-prod V 4.1.0

[Overview](#) [Configuration](#)

 IBM Certified Container

IBM MQ queue manager for Cloud Pak for Integration

[ibm-entitled-charts](#)

[Licenses](#) | [Release Notes](#) | [Qualification](#)

IBM Certified Container

4.1.0 ▾

 IBM MQ Advanced

Introduction

This chart deploys a single IBM® MQ version 9.1.3 middleware that simplifies and accelerates the integration of multiple platforms. It uses message queues to facilitate a messaging solution for cloud, mobile, Internet of Things, and enterprise environments.

Hit configure

Configure

Values for the charts

Cluster Hostname (for deployment)

icp-proxy.apps.ocp.cloudnativekube.com

MQ server image:

docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt

MQ Server image Tag:

9.1.3.0-r4

MQ Server OIDC registration image

docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration

MQ Server OIDC registration image tag:

2.1.0

Managed NFS name (from oc command):

managed-nfs-storage

TLS Secret

ibm-mq-tls-secret

IBM MQ queue manager for Cloud Pak for Integration. Edit these parameters for configuration.

Helm release name *

idcqm1p

Target namespace *

mq

Target Cluster *

local-cluster

License * 

I have read and agreed to the [License agreement](#)

icp-proxy.apps.ocp.cloudnativekube.com

Quick start

Required and recommended parameters to view and edit.

TLS

Configuration settings for TLS

Cluster hostname *

icp-proxy.apps.ocp.cloudnativekube.com

MQ server image:

[docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt](#)

MQ Server image Tag:

[9.1.3.0-r4](#)

Results in the following image and tag being used

[docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ict:9.1.3.0-r4-amd64](#)

Image
Configuration settings for the container image

Image repository *	Image tag *
docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt	9.1.3.0-r4
Image pull secret	Image pull policy (for all images) *
Enter value	IfNotPresent

MQ Server OIDC registration image

docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration

MQ Server OIDC registration image tag:

2.1.0

Single sign-on
Configuration settings for single sign-on

Registration image repository *	Registration image tag *
docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration	2.1.0

TLS Secret

Ibm-mq-tls-secret

TLS
Configuration settings for TLS

Generate Certificate

Cluster hostname *	Secret name
icp-proxy.apps.ocp.cloudnativekube.com	Ibm-mq-tls-secret

Managed NFS name (from oc command):

managed-nfs-storage

Persistence

Configuration settings for Persistent Volumes

Enable persistence *

Use dynamic provisioning *

Data PVC

Configuration settings for the main Persistent Volume Claim

Name *

data

Storage Class name 

managed-nfs-storage

Size *

2Gi

Queue manager

Configuration settings for the Queue Manager

Queue manager name 

IDCQM1

Metrics

Configuration settings for Prometheus metrics

Enable metrics * 

Readiness probe

Configuration settings for the MQ readiness probe, which checks when the MQ listener is running

Initial delay (seconds)

30



Install

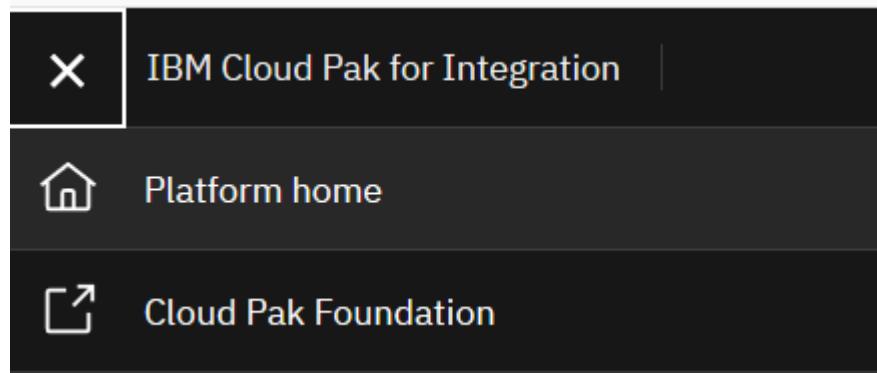
Use the little X to retain your populated chart for re-use if things go wrong.



Installation started. For progress view your Helm releases.

[View Helm Releases](#)

Return to [Catalog](#)



X IBM Cloud Private CL my

Overview

▼ Workloads

- Brokered Services
- DaemonSets
- Deployments
- Helm Releases

Helm Releases

i You are currently viewing only the helm releases of this cluster.

Name	Namespace	Status	Chart name
idcqmq1p	mq	● Deployed	ibm-mqadvanced-server-integration-prod

Job

Name	COMPLETIONS	DURATION	Age
idcqm1p-ibm-mq-registration	1/1	7s	119s

Pod

Name	READY	Status	RESTARTS	Age
idcqm1p-ibm-mq-0	1/1	Running	0	118s
idcqm1p-ibm-mq-registration-fknmq	0/1	Completed	0	119s

Role

Name	Age
idcqm1p-ibm-mq	2m

Service

Name	TYPE	Cluster IP	External IP	Port(s)
idcqm1p-ibm-mq	NodePort	172.30.223.140	<none>	9443:31495/TCP,1414:32182/TCP

ServiceAccount

Name	SECRETS	Age
idcqm1p-ibm-mq	2	2m

StatefulSet

Name	Desired	Current	Age
idcqm1p-ibm-mq	1	1	119s

Events

Search Events

Type	Source	Count	Reason	Message
Normal	statefulset-controller	1	SuccessfulCreate	create Pod idcqm1p-ibm-mq-0 in StatefulSet idcqm1p-ibm-mq successful

items per page **20** | 1-1 of 1 items

Pods					
<input type="text"/> Search Pods					
Name	Namespace	Status	Host IP	Pod IP	Ready
idcqm1p-ibm-mq-0	mq	Running	135.90.82.92	10.129.1.228	1/1
items per page <select>20</select> 1-1 of 1 items					

StatefulSets / idcqm1p-ibm-mq / idcqm1p-ibm-mq-0

idcqm1p-ibm-mq-0

Overview Containers [Events](#)

Search Events

Type	Source	Count	Reason	Message
Normal	kubelet icp4inodeme3.ocp.cloudnativekube.com	1	Started	Started container
Normal	kubelet icp4inodeme3.ocp.cloudnativekube.com	1	Pulled	Container image "docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64" already present on machine
Normal	kubelet icp4inodeme3.ocp.cloudnativekube.com	1	Created	Created container

MQ Web Console

IBM Cloud Private CLUSTER mycluster Create resource Catalog 

Helm Releases

 You are currently viewing only the helm releases of this cluster.

idc 

Name	Namespace	Status	Chart name	Current version	Available version	Updated	Actions
idcab1	mq	● Deployed	ibm-mqadvanced-server-integration-prod	4.0.0	5.0.0 	December 13, 2019 11:17am	Launch ▾
idcqm1p	mq	● Deployed	ibm-mqadvanced-server-integration-prod	4.1.0	5.0.0 	December 16, 2019 02:51pm	Launch ▲
oidcclient-watcher	kube-system	● Deployed	oidcclient-watcher	3.2.1906-rhel	Up To Date	December 4, 2019 09:00pm	console-https



Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to `icp-proxy.apps.ocp.cloudnativekube.com`. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

[Learn more...](#)

[Go Back \(Recommended\)](#)

[Advanced...](#)

`icp-proxy.apps.ocp.cloudnativekube.com:31495` uses an invalid security certificate.

The certificate is not trusted because it is self-signed.

Error code: `MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT`

[View Certificate](#)

[Go Back \(Recommended\)](#)

[Accept the Risk and Continue](#)

Add Widget - Channels

The screenshot shows the IBM MQ Explorer interface. At the top, there's a navigation bar with icons for home, back, forward, and search, followed by the URL `https://icp-proxy.apps.ocp.cloudnativekube.com:31495/ibmmq/console/`. Below the URL, there are tabs for 'Started' and 'Running' queue managers, and links for 'Login - RHOS-OKD-IB...', 'ICPonRHOS-KD-IBM C...', and 'ICP4ionRHOS-OKD-IB...'. A 'Tab 1' tab is selected. On the left, under 'Local Queue Managers', there's a table with one entry: 'IDCQM1' in the Name column and 'Running' in the Status column. On the right, under 'Channels on IDCQM1', there's a table with one entry: 'IVT.SVRCONN' in the Name column and 'Server-connection' in the Type column.

Name	Status
IDCQM1	Running

Name	Type
IVT.SVRCONN	Server-connection

MQ Explorer

Add remote queue manager

IBM MQ Explorer (Installation1)

File Edit Window Help

MQ Explorer - Navigator 

IBM MQ

Queue Managers

Add Queue Manager

Select the queue manager and connection method

Identify the queue manager to add and choose the connection method to use

Queue manager name:

IDCQM1

How do you want to connect to this queue manager?

Connect directly

This option creates a new connection to the queue manager (recommended)

Add Queue Manager

Specify new connection details

Provide details of the connection you want to set up

Queue manager name:

IDCQM1

Connection details

Host name or IP address: icp-proxy.apps.ocp.cloudnativekube.com

Port number: 32182

Server-connection channel: IVT.SVRCONN

Add Queue Manager

Specify user identification details

Provide a userid name and password

Queue manager name: IDCQM1

Enable user identification

User identification compatibility mode

Userid: mqm

Password

No password

The screenshot shows the MQ Explorer interface. The Navigator pane on the left lists 'IBM MQ' with 'Queue Managers' expanded, showing entries for EAIQM1, EAIQM2, IDCQM1, and IDCQM1 (selected). Below these are 'Queues', 'Topics', and 'Subscriptions'. The 'Channels' node is also visible. The Content pane on the right is titled 'Channels' and displays a table with one row:

Channel name	Channel type	Overall channel status
IVT.SVRCONN	Server-connection	Running

Using Runmqsc in client mode

```
SET MQSERVER=IVT.SVRCONN/TCP/icp-proxy.apps.ocp.cloudnativekube.com(32182)
```

```
Runmqsc -c IDCQM1
```

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>SET MQSERVER=IVT.SVRCONN/TCP/icp-proxy.apps.ocp.cloudnativekube.com(32182)
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>runmqsc -c IDCQM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager IDCQM1.

def ql(manual.q)
  1 : def ql(manual.q)
AMQ8006I: IBM MQ queue created.
end
  2 : end
2 command responses received.
```

```
C:\temp>type 1q-npwd.mqsc
define ql(DA) replace

C:\temp>runmqsc -c IDCQM1 < 1q-npwd.mqsc
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager IDCQM1.

      1 : define ql(DA) replace
AMQ8006I: IBM MQ queue created.
2 command responses received.
```

Appendix H – MQ client Security for non ICP4i containers

MQ containers with admin/passw0rd and app/passw0rd as default users

IBM MQ for developer container images have admin/passw0rd and app/passw0rd as default users

MQSC that sets up access for client connections.

* Connection authentication

```
DEFINE AUTHINFO('ORG.AUTHINFO') AUTHTYPE(IDPWOS) CHCKCLNT(REQDADM)  
CHCKLOCL(OPTIONAL) ADOPTCTX(YES) REPLACE
```

```
ALTER QMGR CONNAUTH('ORG.AUTHINFO')
```

```
REFRESH SECURITY(*) TYPE(CONNAUTH)
```

* Channels (Application + Admin)

```
DEFINE CHANNEL('IVT.SVRCONN') CHLTYPE(SVRCONN) REPLACE
```

```
DEFINE CHANNEL('ORG.APP.SVRCONN') CHLTYPE(SVRCONN) MCAUSER('app') REPLACE
```

* Channel authentication rules

```
SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS) DESCRIPTOR('Back-stop rule -  
Blocks everyone') ACTION(REPLACE)
```

```
SET CHLAUTH('ORG.APP.SVRCONN') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(CHANNEL)  
CHCKCLNT(REQUIRED) DESCRIPTOR('Allows connection via APP channel') ACTION(REPLACE)
```

```
SET CHLAUTH('IVT.SVRCONN') TYPE(BLOCKUSER) USERLIST('nobody') DESCRIPTOR('Allows admins on  
ADMIN channel') ACTION(REPLACE)
```

```
SET CHLAUTH('IVT.SVRCONN') TYPE(USERMAP) CLNTUSER('admin') USERSRC(CHANNEL)  
DESCRIPTOR('Allows admin user to connect via ADMIN channel') ACTION(REPLACE)
```

* Authority records

```
SET AUTHREC GROUP('mqclient') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
```

```
SET AUTHREC PROFILE('ORG.**') GROUP('mqclient') OBJTYPE(QUEUE)  
AUTHADD(BROWSE,GET,INQ,PUT)
```

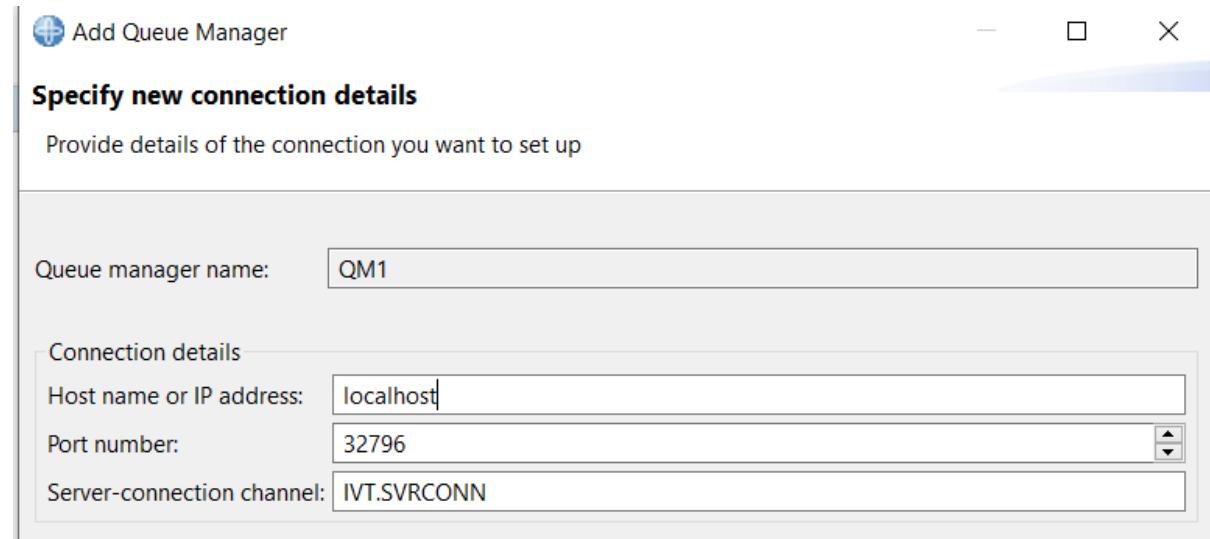
```
SET AUTHREC PROFILE('ORG.**') GROUP('mqclient') OBJTYPE(TOPIC) AUTHADD(PUB,SUB)
```

Access via MQ console

<https://localhost:32795>

Login in with admin/passw0rd

Access via MQ Explorer



Specify user identification details

Provide a userid name and password

Queue manager name: QM1

Enable user identification

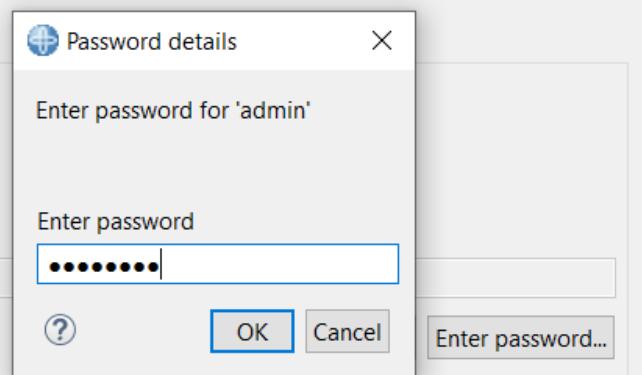
User identification compatibility mode

Userid: admin

Password

- No password
- Prompt for password
- Use saved password

Saved password:



Access via RUNMQSC client

```
SET MQSERVER=IVT.SVRCONN/TCP/localhost(32796)
```

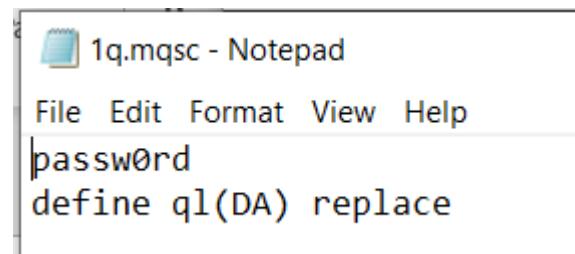
Runmqsc manually

```
C:\temp>runmqsc -c -u admin QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Enter password:
*****
Starting MQSC for queue manager QM1.

end
      1 : end
0 command responses received.
```

Runmqsc piping in a file

If -u is on the command and we are piping in, runmqsc assumes the first line is the password.



```
1q.mqsc - Notepad
File Edit Format View Help
passw0rd
define ql(DA) replace
```



```
C:\temp>runmqsc -c -u admin QM1 < 1q.mqsc
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Enter password:
Starting MQSC for queue manager QM1.

      1 : define ql(DA) replace
AMQ8006I: IBM MQ queue created.
2 command responses received.
```