**Running MQ in a container.**

**Version control**

|  |  |
| --- | --- |
| Created by | Marcos Lara |
| Version | 001 – draft - version |
| When | 21-Mar-2025. |
| Objective | Provide a detailed guide on configuring MQ to run in a container on VirtualBox VM |

**Notes:**

* This document is part of the MTS installation in the VirtualBox VM. If you have already created your Red Hat 8 VM, we can skip some parts of this document.
* This intention is intended for use as a test on your development machine, simplifying your integration tests with MTS.
* This is being created from memory; I will need to run some tests to ensure everything is working as it should.

Install a VirtualBox virtual machine with Linux Red Hat 8.

Create a user: mtsadmin

Create a group: mts

Add the user mtsadmin to the group mts

Disable the password for the “Wheel” group

Login in your VM as mtsadmin

Create a directory that will be used as ‘share’ between your redhat8 and the container:

$ mkdir ~/shared

$ cd ~/shared

Create the following scripts in the ‘shared’ directory. These will be used to initialise the container. (Note that the same approach will be applied to the RGW container, and the steps will be included in a separate document.)

**[mtsadmin@localhost shared]** $ ls -ltr

total 16

startup.mqsc

Create the following scripts at the HOME of the mtsadmin user:

**[mtsadmin@localhost scripts]** $ cat mq\_clean\_restart.sh

#!/bin/bash

#

# This script is designed to clean the MQ container by deleting

# the existing one and creating a new container. When the new

# container starts, a script runs to configure the users

# "marcoslaranz" and "mtsadmin" for access to the QManager and

# to create the necessary queues.

#

### Clean all ##

getContainerId() {

echo $(podman ps | grep icr | cut -c 1-3)

}

getContainerIdNotRunning() {

echo $(podman ps -a | grep icr | cut -c 1-3)

}

contid=$(getContainerId)

if [ "$contid" != "" ]; then

echo $(date): The container is running with Id = $contid .Removing old MQ installation ...

podman stop $contid

podman rm $contid

podman volume rm mqtest

podman volume rm sharemq

else

#Check if the container is not running

contid=$(getContainerIdNotRunning)

if [ "$contid" != "" ]; then

echo $(date): The container is NOT running with Id = $contid .Removing old MQ installation ...

podman rm $contid

podman volume rm mqtest

podman volume rm sharemq

fi

fi

echo Creating the volumes:

podman volume create mqtest

podman volume create sharemq

## Start again ##

#podman run -d --name mq -p 1414:1414 -p 9443:9443 -v mqtest:/mnt/mqm -v /home/mtsadmin/sharemq:/opt/sharemq:z -e LICENSE=accept -e MQ\_QMGR\_NAME=QM1 -e MQ\_DEV=false -e MQ\_ADMIN\_PASSWORD=passw0rd -e MQ\_AP\_PASSWORD=passw0rd icr.io/ibm-messaging/mq:latest

podman run -d --name mq -p 1414:1414 -p 9443:9443 -v mqtest:/mnt/mqm -v /home/mtsadmin/sharemq:/opt/sharemq:z -e LICENSE=accept -e MQ\_QMGR\_NAME=SERVQM -e MQ\_DEV=false -e MQ\_ADMIN\_PASSWORD=passw0rd -e MQ\_AP\_PASSWORD=passw0rd icr.io/ibm-messaging/mq:latest

#podman run -d --name mq -p 1414:1414 -p 9443:9443 -v mqtest:/mnt/mqm -v /home/mtsadmin/sharemq/startup.mqsc:/mnt/mqm/startup.mqsc -e LICENSE=accept -e MQ\_QMGR\_NAME=QM1 -e MQ\_DEV=false -e MQ\_ADMIN\_PASSWORD=passw0rd -e MQ\_AP\_PASSWORD=passw0rd icr.io/ibm-messaging/mq:latest

# the script startup.mqsc is not running automatically as it should

#podman run -d --name mq -p 1414:1414 -p 9443:9443 -v mqtest:/etc/mqm -v /home/mtsadmin/sharemq/startup.mqsc:/etc/mqm/startup.mqsc:z -e LICENSE=accept -e MQ\_QMGR\_NAME=QM1 -e MQ\_DEV=false -e MQ\_ADMIN\_PASSWORD=passw0rd -e MQ\_AP\_PASSWORD=passw0rd icr.io/ibm-messaging/mq:latest

contid=$(getContainerId)

echo The new container was created with the following ID = $contid

#podman exec -it $contid bash

#podman exec -it $contid runmqsc QM1 -f /opt/sharemq/initqm1.txt

#podman exec $contid runmqsc QM1 -f /opt/sharemq/testqm.txt

## podman exec -it 14e tail -f /mnt/mqm/data/qmgrs/QM1/errors/AMQERR01.LOG

#/etc/mqm//startup.mqsc

**[mtsadmin@localhost shared]**$ cat startup.mqsc

DEFINE LISTENER(LISTENER) TRPTYPE(TCP) CONTROL(QMGR) PORT(1414) REPLACE

START LISTENER(LISTENER)

DEF CHL(SWIFT.CH) CHLTYPE(SVRCONN) REPLACE

SET CHLAUTH(SWIFT.CH) TYPE(USERMAP) CLNTUSER('mtsadmin') USERSRC(CHANNEL) DESCR('ALLOWS mtsadmin USER TO CONNECT VIA CHANNEL') ACTION(REPLACE)

SET CHLAUTH(SWIFT.CH) TYPE(USERMAP) CLNTUSER('mtsadmin') USERSRC(MAP) MCAUSER(1001) DESCR('ALLOW mtsadmin AS 1001') ACTION(REPLACE)

SET AUTHREC OBJTYPE(QMGR) PRINCIPAL('mtsadmin') AUTHADD(ALL)

SET AUTHREC PROFILE(\*) OBJTYPE(QUEUE) PRINCIPAL('mtsadmin') AUTHADD(ALL)

REFRESH SECURITY(\*) TYPE(CONNAUTH)

define qlocal(ABC\_SWF1\_STG\_RCV)

define qlocal(ABC\_SWF1\_STG\_SND)

define qlocal(BNK.5SQ.SOA\_SSO\_1.IN)

define qlocal(BNK.5SQ.SOA\_SSO\_1.OUT)

define qlocal(BNK.5SQ.SOA\_SUI\_1.IN)

define qlocal(BNK.5SQ.SOA\_SUI\_1.OUT)

define qlocal(BNK.5SQ.V2B\_SOA\_1.IN)

define qlocal(BNK.5SQ.V2B\_SOA\_1.OUT)

define qlocal(BNK.6SQ.TSI\_OUT\_MB)

define qlocal(BNK.6SQ.TSI\_REPLY\_MB)

define qlocal(BNK.6SQ.TSI\_STAGE\_MB)

define qlocal(BNK\_AMH.BNK\_MTS.SWIFTOTHER)

define qlocal(BNK\_AMH.BNK\_MTS.SWIFTOTHER\_ML5)

define qlocal(BNK\_AMH.BNK\_MTS.SWIFTOTHER.STAGE)

define qlocal(BNK\_MTS.BNK\_AMH.SEND1)

define qlocal(BNK\_MTS.BNK\_AMH.SEND1.STAGE)

define qlocal(BNK\_MTS.BNK\_AMH.SEND2\_ML5)

define qlocal(BNK\_MTS.BNK\_AMH.SEND2.STAGE\_ML5)

define qlocal(BNK\_MTS.CBA\_HU.SEND1.STAGE)

define qlocal(BNK\_MTS.GPI.DLV)

define qlocal(BNK\_MTS.GPI.SND)

define qlocal(BNK\_P2A.CPH.API.RCV)

define qlocal(BNK\_P2A.CPH.API.RESP)

define qlocal(BNK.PRO.ISOCLP10\_TRANSMIT)

define qlocal(BNK.PRO.ISOCLP50\_TRANSMIT)

define qlocal(BNK.PRO.ISOCLP51\_TRANSMIT)

define qlocal(BNK.PRO.ISOFCA40\_TRANSMIT)

define qlocal(BNK.PRO.ISOFCA81\_TRANSMIT)

define qlocal(BNK.PRO.ISOODB60\_TRANSMIT)

define qlocal(BNK.PRO.ISOODB61\_IN)

define qlocal(BNK.PRO.ISOODB61\_TRANSMIT)

define qlocal(BNK.PRO.ISOTRM70\_TRANSMIT)

define qlocal(BNK.PRO.ISOTRM71\_IN)

define qlocal(BNK.PRO.ISOTRM71\_TRANSMIT)

define qlocal(BNK.SWF1.RCVABC\_SWF1\_STG\_RCV)

define qlocal(BNK.SWF1.RCV.STG)

define qlocal(BNK.V60.SVC.QA.RTI1.RCV)

define qlocal(BNK.V60.SVC.QA.RTI1.RCVSTG)

define qlocal(BNK.V60.SVC.QA.RTI1.REPLY2)

define qlocal(BNK.V60.SVC.QA.RTI1.STG)

define qlocal(BNK.V60.SVC.QA.RTI1.XMT)

define qlocal(BNK.V60.SVC.QA.RTI2.RCV)

define qlocal(BNK.V60.SVC.QA.RTI2.RCVSTG)

define qlocal(BNK.V60.SVC.QA.RTI2.STG)

define qlocal(BNK.V60.SVC.QA.RTI2.XMT)

define qlocal(CAO\_ACKPNDQ)

define qlocal(CBC\_ACKPNDQ)

define qlocal(CCO\_ACKPNDQ)

define qlocal(FNI\_ACKPNDQ)

define qlocal(GAO\_ACKPNDQ)

define qlocal(GCO\_ACKPNDQ)

define qlocal(GPI\_ACKPNDQ)

define qlocal(INP\_ACKPNDQ)

define qlocal(MAS\_ACKPNDQ)

define qlocal(OAD\_ACKPNDQ)

define qlocal(QA1.CBT1.STAGE)

define qlocal(QA1.CBT1.TARGET)

define qlocal(QA1.CBT2.TARGET)

define qlocal(SRR\_ACKPNDQ)

define qlocal(SSR\_ACKPNDQ)

end

**Next Steps**

* Accessing MQ using the command line
* Accessing MQ using MQExplorer
  + There are two options: you can use an MQExplorer if you have already installed it, or you can run it as a container and then access it using a browser.
* Exporting data from MTS database
* Loading files into the RGW database