21-MQ-MFT

A close up of a logo

Description automatically generated

**IBM MQ Advanced**

LAB: MQMFT Agent in Container

(using MQ 9.2.2)

[1. Introduction 3](#_Toc74654827)

[2. Entering commands for this lab 4](#_Toc74654828)

[3. Setup configuration for the lab 5](#_Toc74654829)

[3.1 Run queue manager in a container 6](#_Toc74654830)

[3.2 Run agents in container 8](#_Toc74654831)

[4. Automate transfer with Resource monitor 11](#_Toc74654832)

[End of Lab Guide 13](#_Toc74654833)

# Introduction

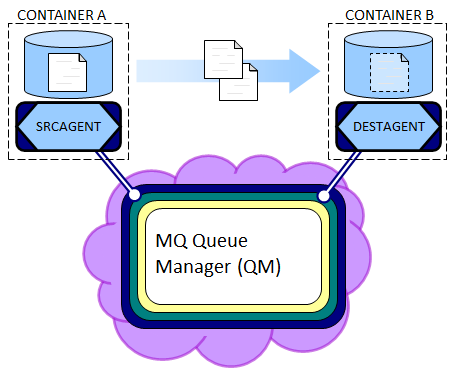
This lab introduces MQ Managed File Transfer Agent in a Docker image. The Docker imager image is available in the DockerHub (<https://hub.docker.com/r/ibmcom/mqmft>). The currently available docker image is developed using MQ Managed File Transfer Redistributable Package version 9.2.2 CD release.

IBM MQ Managed File Transfer transfers files between systems in a managed and auditable way, regardless of file size or the operating systems used. You can use Managed File Transfer to build a customized, scalable, and automated solution that enables you to manage, trust, and secure file transfers. Managed File Transfer eliminates costly redundancies, lowers maintenance costs, and maximizes your existing IT investments.

In this lab, you will:

* Create a queue manager in a Docker container using the Docker image from DockerHub.(<https://hub.docker.com/r/ibmcom/mq>)~~.~~ This queue manager will be used as Coordination, Command and Agent queue manager.
* Create two MQMFT agents, SRCAGENT and DESTAGENT running in separate containers.
* Create a resource monitor to automatically transfer files from source file system to destination file system.

Diagram 1: Topology used for this lab



**Basic Requirements**

* Linux Operation System. Preferably RHEL 8.1 or Ubuntu 18.x. Instructions in this document were developed using *podman* on RedHat Enterprise Linux 8.1.
* As this lab involves usage of containers, this lab requires Docker Container Runtime.to be installed on your machine.
* If you are using RedHat Enterprise Linux v8 or higher, then *podman* is already pre-installed and installation of Docker runtime is not required. All *podman* commands of this lab can be run using *docker* as well.
* If you are using other version of Linux, for example Ubntu then download and install Docker Runtime from <https://docs.docker.com/engine/install/ubuntu/>. Docker Runtime for other Linux distributions can also be downloaded from Docker website.

**Key Note:**

The instructions for this lab have been developed using Podman Container Runtime on RedHat Enterprise Linux 8.1.

**Further information**

* IBM MQ Knowledge Centre (<https://www.ibm.com/docs/en/ibm-mq/9.2?topic=overview-managed-file-transfer>)
* IBM MQ Container (<https://hub.docker.com/r/ibmcom/mq>)
* IBM MQ Managed File Transfer Container - (<https://hub.docker.com/r/ibmcom/mqmft>)

**Acknowledgements**

This lab was written by Shashikanth R Thambrahalli (shashikanth@in.ibm.com), MQ Software Developer,

# Entering commands for this lab

In this lab, you will come across some quite long commands to enter. You will be using Linux Shel like ***bash***. To avoid a lot of typing, you may copy the commands from this document and execute on the Shell.

**Some points to consider**

* There may be hyphens (-) introduced in this document where text is split between lines; you should remove these,
* Sometimes a command may be similar to a previous one that you have entered, and it may be quicker to use the up arrow in the command prompt to retrieve the earlier command and then editing it there on the command line.

# Setup configuration for the lab

* In this lab you will create a single queue manager running in a container. This queue manager will be used as Managed File Transfer Coordination, Command and Agent queue manager.
* The MQSC scripts to create required Managed File Transfer objects have also been provided with the lab.
* The queue manager running in the container will use host file system for its logs. Hence you will create a persistent volume as well.
* You will be creating two Managed File Transfer Agents running in containers.
* The host file system will be used as source and destination file system for agent. For this purpose, you will be creating tow directories on the host file system. The directory will be mounted into a container as a file system.
* This document accompanies mftlab.tar file containing scripts, config JSON file, samples etc. Unpack the tar file into your home directory using the following command:

tar xvf mftlab.tar

The directory will contain the following:

./mftlab/qm

* coordsetup.mqsc
* destagent.mqsc
* srcagent.mqsc
* setqmaut.sh

./mftlab/agent

* agentconfig.json - A JSON file containing the configuration information required for creating agent containers. [Here](https://github.com/ibm-messaging/mft-cloud/blob/mftubi/docs/agentconfig.md) are details of each attribute in the JSON file. **You will need to replace the value of host name attribute with the host name or IP address of the machine where you are running this lab. Run ifconfig command to get this information. Use the IP address/host name from eth0.**

./mftlab/srcdir

* \*.csv sample files used for testing resource monitor

Initial steps for setting up lab

|  |  |
| --- | --- |
|  | Upload the mftlab.tar file using scp or WinScp to home directory. |
|  | Open a Linux command prompt session. You will be in your home directory /home/student. If not, change to this directory.  Unpack the mftlab.tar in the current directory using  tar xvf mftlab.tar |

## Run queue manager in a container

We shall now create the queue manager and required queue manager object for Managed File Transfer.

|  |  |
| --- | --- |
|  | Now create a docker volume, **mqmftdata** to be as persistent volumen for the queue manager  podman volume create mqmftdata    Name of the volume created will be displayed after successful completion of the command. Verify by running the following command  podman volume ls |
|  | Now create the queue manager, MQMFT. Docker image for creating the queue manager will be downloaded from DockerHub.  Note:   1. Name of the queue manager, MQMFT is passed via environment variable MQ\_QMGR\_NAME 2. License is acceptanced via environment variable LICENSE. 3. Queue manager listener port will be 1414 4. Volume mqmftdata will be mounted as /mnt/mqm 5. As -d option is used, the container will running in the background. 6. Container will be named mqmftqm   podman run \  --env LICENSE=accept \  --env MQ\_QMGR\_NAME=MQMFT \  --publish 1414:1414 \  --publish 9443:9443 \  --detach \  --volume mqmftdata:/mnt/mqm \  -d \  --name=mqmftqm \  docker.io/ibmcom/mq |
|  | The command will download MQ container image from DockerHub, if it’s not already available on the local registry and runs the container.  Verify queue manager is running with the following command    Important Note:  Run the following command if you want to stop the container.  podman stop mqmftqm  Run the following command to remove the container name.  podman rm mqmftqm |
|  | Next step is to configure the queue manager for using with Managed File Transfer. As the same queue manager is being used as Coordination, Command and Agent, all objects will be created in the same qeue manager.  You will be logging into the queue manager for creating the queue manager. You can manually create the objects required for Managed File Transfer. However MQSC scripts have been provided with lab. You can just copy the script files to queue manager container and simply pipe them to *runmqsc* command.   1. Copy the MQSC scripts to queue manager container.   podman cp mftlab/qm/coordsetup.mqsc mqmftqm:/coordsetup.mqsc  podman cp mftlab/qm/destagent.mqsc mqmftqm:/destagent.mqsc  podman cp mftlab/qm/srcagent.mqsc mqmftqm:/srcagent.mqsc  podman cp mftlab/qm/setqmaut.sh mqmftqm:/setqmaut.sh   1. Run the following command to login into the queue manager container   podman exec -it mqmftqm /bin/bash   1. Run dspmq comand and verify queue manager is running   dspmq   1. Create coordination queue manager objects. Run the following command   runmqsc MQMFT < coordsetup.mqsc     1. We will have two agents in this lab. So create the required queue manager objects for the two agents. SRCAGNT and DESTAGNT will be the name of agents.   Run the following to create objects for source agent SRCAGENT  runmqsc MQMFT < srcagent.mqsc     1. Run the following to create objects for source agent DESTAGENT   runmqsc MQMFT < destagent.mqsc     1. As the agents and queue manager runs in different containers, you will need to setup authorities on the objects created above so that agents can connect. Run the following Shell script to setup the required authorities.   ./setqmaut.sh   1. Run the following command to exit out of queue manager container.   exit  This completes the queue manager configuration. |

## Run agents in container

To create a Managed File Transfer Agent, name of the agent, agent queue manager name, queue manager host name, port, channel, coordination queue manager details, command queue manager details, and a bunch of other information is required. You may want to further customize the agent by specifying additional properties in agent’s properties file. All the configuration information must be available for agent container to start. This information is provided through a JSON file located on the host file system. This file gets loaded into container and agent is then configured.

**You will need to replace the value of host name attribute with the host name or IP address of the machine where you are running this lab. Run ifconfig command to get this information. Use the IP address/host name from eth0.**

|  |  |
| --- | --- |
|  | A sample agent configuration JSON file is made available with this lab. The contents of the JSON file are self-explanatory. The file is in the $HOME*/mftlab/agent* directory |
|  | Create the following two directories in current directory on the host file system.  mkdir srcdir  mkdir destdir  Provide permissions so that agent containers can read/write from/to the mounted directory.  chmod 777 srcdir  chmod 777 destdir  Copy a sample file to srcdir for running tests later in the lab  cp ./mftlab/samplecsv/airtravel.csv ./srcdir |
|  | Run source agent, SRC AGENT in container in the background. Remove the -d parameter if you would like to run in the foreground in which case you will need to start on the command shell. When running in the foreground, logs will be displayed on the console and this helps to debug any issue with running of the container.  Description of the parameters:  **-v ./mftlab/agent** – Path on the source file system mounted into container as /mftagentcfg. This is directory where configuration information JSON file required for setting up an agent is located.  **-v ./srcdir:/mountpath** – Path on the host file system mounted into container as /mountpath. This will be the directory where the agent will pick files to transfer  **MFT\_AGENT\_NAME=SRCAGENT** – Name of the agent  **MFT\_LOG\_LEVEL="info"** – Level of logging. *info* displays high level log informationof container creation on the console. *verbose –* displays low level logs including contents of agent’s output0.log file.  **LICENSE=accept** – Accept product license  **MFT\_AGENT\_CONFIG\_FILE=/mftagentcfg/agentconfig.json** – Name of JSON file containing agent configuration information.  Execute the following command to run SRCAGENT in a container.  podman run \  -v /home/student/mftlab/agent:/mftagentcfg \  -v /home/student/srcdir:/mountpath \  --env MFT\_AGENT\_NAME=SRCAGENT \  --env MFT\_LOG\_LEVEL="verbose" \  --env LICENSE=accept \  --env MFT\_AGENT\_CONFIG\_FILE=/mftagentcfg/agentconfig.json \  --name srcagent \  -d \  docker.io/ibmcom/mqmft |
|  | Once the command completes, the output would be something like below    Once the command completes, run the following command verify if the container is running  podman ps  The output would be similar as below |
|  | Similarly run destination agent container now.  podman run \  -v /home/student/mftlab/agent:/mftagentcfg \  -v /home/student/destdir:/mountpath \  --env MFT\_AGENT\_NAME=DESTAGENT \  --env MFT\_LOG\_LEVEL="verbose" \  --env LICENSE=accept \  --env MFT\_AGENT\_CONFIG\_FILE=/mftagentcfg/agentconfig.json \  -d \  --name destagent \  docker.io/ibmcom/mqmft  After the command is complete, verify the container status by running  podman ps    This completes running of agents in a container phase. |

# Automate transfer with Resource monitor

Now that you have configured queue manager and agents, it’s time to automate transfers using a resource monitor. Resource monitor is major feature of MQ Managed File Transfer Agent. It helps to automatically trigger transfers at the occurrence of an event, for example arrival of a file with certain pattern name in a directory or MQ Queue. You can read more about [Resource Monitors](https://www.ibm.com/docs/en/ibm-mq/9.2?topic=resources-mft-resource-monitoring-concepts) in Knowledge Center.

Remember you mounted /srcdir of the host file system into srcagent container as /mountpath directory. Similarly, /destdir was mounted as /mountpath of destagent container.

|  |  |
| --- | --- |
|  | Run the following command login to source agent container  podman exec -it srcagent /bin/bash  Run the following command to status of available agents  fteListAgents  The output would be something like below |
|  | Before setting up resource monitor, let’s verify transfer works. Submit a transfer request using fteCreateTransfer command.  fteCreateTransfer -rt -1 -sa SRCAGENT -sm MQMFT -da DESTAGENT -de overwrite -dm MQMFT -df "/mountpath/airtravel.csv" "/mountpath/airtravel.csv"    View the status of transfer by running the *mqfts* utility. This utility displays transfer status by parsing *capture0.log* file located in source agent’s log directory. |
|  | To view more details of the transfer, run *mqfts –id=<transfer id>*. For example:  mqfts --id=414d51204d514d46542020202020202044bfbd60019b0040 |
|  | Now it’s time to automate transfers using a resource monitor. You will create a Directory type resource monitor that monitors a directory for certain pattern of files. It will transfer file from that directory when files of matching pattern are placed in the directory.  In the below example you will create a resource monitor that monitors “/mountpath/srcdir/input” directory every 5 seconds for “.csv” files and transfers them to “/mountpath/destdir/output” folder on the destination agent.  The fteCreateTransfer -gt option creates a file in the current directory. You may not have access to current directory. Hence task.xml file will be created in /mountpath directory.  Now run the following commands to create transfer definition for the monitor FILEMON.  **Important note: The ‘$’ must be prefixed with escape character ‘\’ on bash shell, otherwise it will be ignored when the command is run.**  fteCreateTransfer -gt /mountpath/task.xml -sa SRCAGENT -sm MQMFT -da DESTAGENT -dm MQMFT -sd delete -de overwrite -dd "/mountpath/output" "**\$**{FilePath}"    Then run the following command to create resource monitor  fteCreateMonitor -ma SRCAGENT -mn FILEMON -md "/mountpath/input" -pi 5 -pu SECONDS -c -tr "match,\*.csv" -f -mt /mountpath/task.xml    Verify the resource monitor creation by running the following command  fteListMonitors -v -mn FILEMON -ma SRCAGENT |
|  | Now that resource monitor has been created and started, exit the shell of srcagent container to come back to host systems shell.   1. For your convenience the /mountpath/samplecsv directory already has some .csv files. So copy the csv files to srcdir directory.   mkdir -p /home/student/srcdir/input  cp /home/student/mftlab/samplecsv/input/\*.\* /home/student/srcdir/input   1. After few seconds, verify that transfer has completed, and files are indeed available in /home/student/destdir/output directory 2. You can also verify the transfer status by logging into srcagent container and running mqfts command   podman exec -it srcagent /bin/bash    This completes the setting up of automated transfers using resource monitors.  Logout of srcagent container shell, if you had logged in. |
|  | Resource monitor triggers a transfer only if any new files arrive in the monitored directory or any existing files are modified. To verify this, create a new .csv file by running the following command:  touch ./srcdir/input/newsalary.csv  A transfer will be trigged when resource monitor starts the next poll. The polling interval of resource monitor is set to 5 seconds, a transfer will be triggered within 5 seconds. Verify the contents of /destdir/output after 5 seconds.  You can also verify the resource monitor transfer triggers transfers only when files of a matching pattern arrive in the monitored folder. Create a file by running the following command  touch ./srcdir/input/oldperks.xls  Verify the contents of /destdir/output after 5 seconds, oldperks.xls file should not be present. |
|  | It’s now time to explore other commands of Managed File Transfer   1. Stop resource monitor usingfteStopMonitor -ma SRCAGENT -mn FILEMON command. Place new files with .csv extension in ./srcdir on the host file system and see if the new files are transferred. 2. Start monitor using fteStartMonitor -ma SRCAGENT -mn FILEMON command and verify new csv files you placed are transferred. 3. Stop agent using fteStopAgent SRCAGENT command and verify the container is still running. 4. Restart the agent using fteStartAgent SRCAGENT command. |
|  | Once you have explored other commands of Manged File Transfer, stop all containers with the commands below  podman stop mqmftqm  podman stop srcagent  podman stop destagent  Verify that containers have stopped by running command  podman ps  The command output should not list any containers. |

# End of Lab Guide