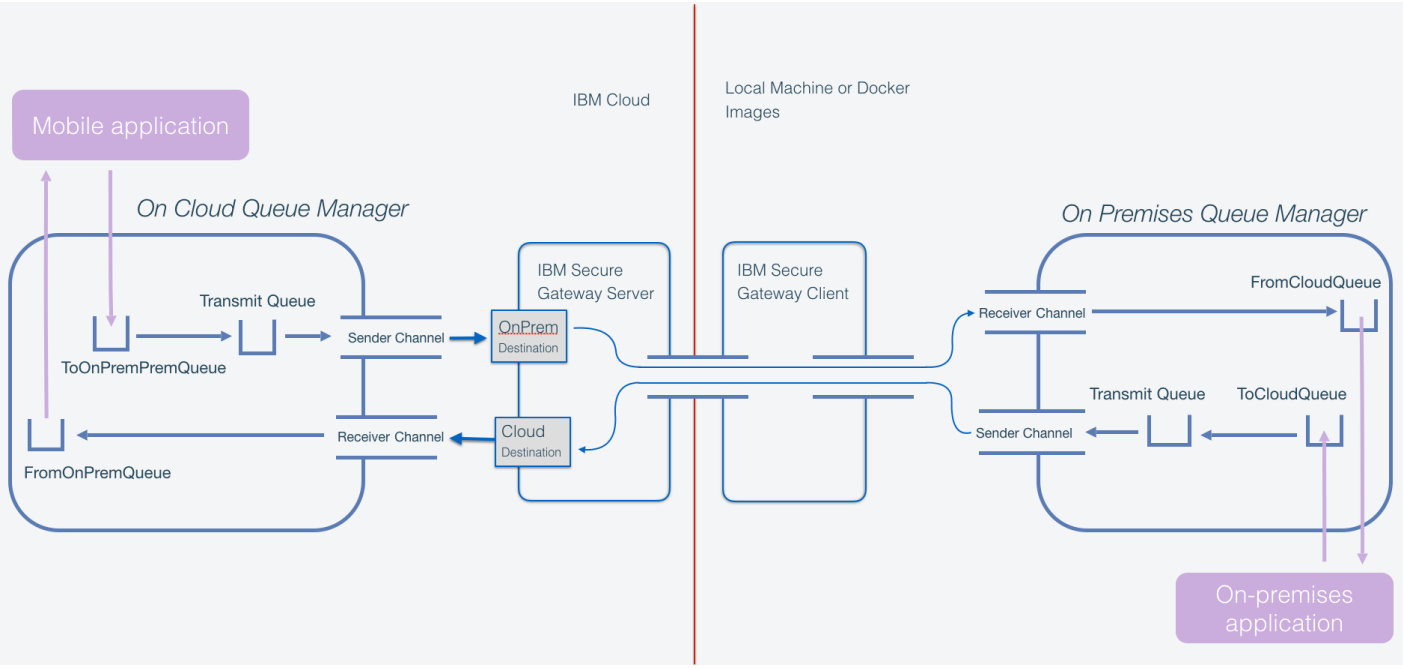
Connecting an on-premise queue manager to an IBM MQ On Cloud queue manager via the IBM Secure Gateway.



1. TABLE OF CONTENTS

2. CONNECTING TO AN ON-PREMISES QUEUE MANAGER

2.1. OVERVIEW

This document describes how to connect a cloud-based IBM MQ Queue Manager to an 'on premise' Queue Manager via the IBM Secure Gateway. The content is divided into sections describing how to establish a connection via the secure gateway using MQSC commands or by using the MQ Console user interface. The final section describes how to add Mutual TLS authentication.

If you copy and paste commands from this document, they may not copy in the correct format e.g. incorrect single quotes may need to be re-typed

1. INITIAL SETUP

The system is composed of three components:

1. A cloud-hosted queue manager deployed using the MQ service in IBM Cloud. In the following pages this is referred to as “oncloudQM”.
2. An “on-premises” queue manager that will be connected to the cloud queue manager. The queue manager in this document is called “onprem”
3. An IBM Secure Gateway for tunnelling the sender/receiver channels between the two queue managers.

3. THE STEPS TO FOLLOW

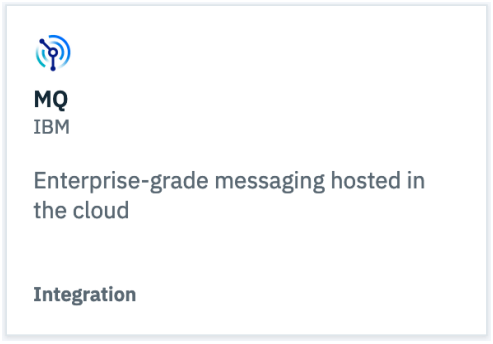
1. Create a cloud based MQOnCloud Queue Manager.
2. Create a local MQ Queue Manager. We tested this with a docker image running RedHat 8.1 Linux - but any IBM MQ Queue Manager should work.
3. Create a cloud based Secure Gateway, install its client near to your on-premises queue manager. We installed the supplied Docker Image, but you could also follow the instructions to download and install the client directly on your hardware.
4. Create channels to pass messages between the two Queue Managers.
5. Install TLS security.

4. CREATE AN MQ ON CLOUD QUEUE MANAGER

If you already have a queue manager, you can skip this section.

Log into your IBM Cloud dashboard.

In the IBM Cloud dashboard, select 'Catalog', then 'Integrate'. You will find the tile you need in the 'integrate' category.



Click on the MQ tile - note the service name you are creating and click 'Create'.

You now have a service running, but no queue manager, so click 'Create' in the queue managers list window. Give your queue manager a name, and a display name.

Select the size of queue manager which is appropriate for your requirements and click 'Create'.

The creation of a queue manager sets up all the cloud infrastructure for you - this can take a few minutes. When this is finished, your queue manager will show in a list, and will have a status of 'Running'. You must wait until that state shows before proceeding.

In order to be able to connect to the queue manager later, you will need to gather the connection information. You can do this as soon as the queue manager shows "running".

Click on your queue manager in the list and select "Connection Information". Download the Plain Text version and save the details in a file for later use.

Your text should look similar to this:

```
Platform: IBM MQ on Cloud

Queue manager name: MyCloudQM
Hostname: mycloudqm-c699.qm.us-preprod.mqcloud.ibm.com
Listener port: 31835

Application channel name: CLOUD.APP.SVRCONN
Administration channel name: CLOUD.ADMIN.SVRCONN

Deployment location: bmx-us-south

MQ Web Console login: https://web-mycloudqm-c699.qm.us-preprod.mqcloud.ibm.com/
ibmmq/console
```

If you do not already have one, you will also need a username and password to administer via the web console. You can create a user as follows.

Above the list of queue managers there are two tabs "user credentials" and "application credentials". Click the "user credentials" tab and create a new user. In the email address field give your own email address and click the button to “Generate MQ username”.

The name which is generated is the username which will be required to access the administration console later. The password will be the API key associated with your account - which you can reset when you first log into the console if you have forgotten it.

Follow the same procedure in the application credentials tab to create an application. Note that this tab asks for an API key name and will create an API key for you. This is not an account-wide API key - it applies only to this application, so every application has a different password. Download this, you can use it to post messages directly to the MQ On IBM Cloud queue manager.

You will use the hostname and the port to connect the secure gateway later. You will use the administration channel to help configure TLS later. Note that you are also given the address of the web console. You can use this in a browser, or alternatively you can click on the ‘Administer’ button next to the Connection Information button you used earlier.

Open the details panel of your queue manager and select “Administration” from the menu. Your login details will be displayed, and you shall have the opportunity to reset your API key if you do not know it - by selecting the “runmqsc” button and resetting the API key.

Finally select the “Launch MQ Console” button - enter the credentials and log in - if you are using the most recent versions of IBM MQ On Cloud, single sign-on is enabled and you will be redirected to the console without the need to enter your credentials.

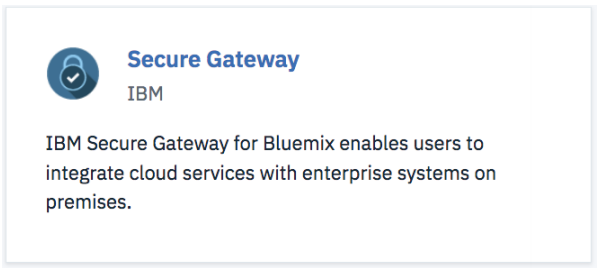
This document assumes that you already have a local queue manager, and can access it with administrative tools such as runmqsc and the MQ Console.

If you do not have a local queue manager, you can get a docker image with a test queue manager by following the instructions here:

https://github.com/ibm-messaging/mq-container/blob/master/docs/usage.md

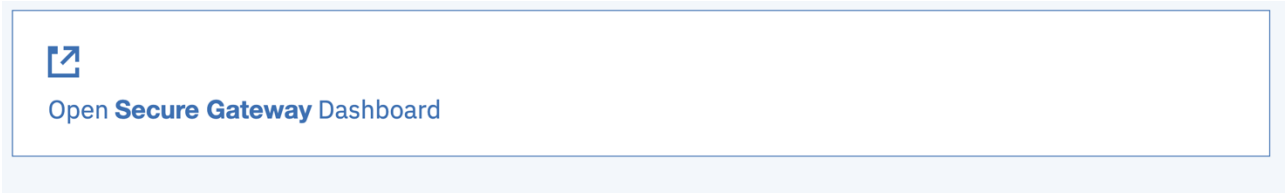
5. CREATE THE CLOUD-BASED SECURE GATEWAY

In a browser, go back to your IBM cloud catalog, and from the ‘integrate’ group select the ‘IBM Secure gateway’. Click ‘Create’.



You will be creating a two-way traffic of messages which may be initiated by either end of the flow, so in the IBM Secure Gateway you will need two destinations configured, and one client downloaded to the local environment.

First the gateway needs to be created. Once your service instance has been created, click the following button which will take you to the secure gateway dashboard:



On the dashboard, there is a large button that has a + symbol and says “Add Gateway”. Click this and name the gateway something memorable. Leave the other fields set to their defaults and click “add gateway”.

Click on “destinations” then select the big “+” under destinations. You will be asked to choose whether the destination is in the cloud or on-premise. The first destination will be “on premise”. This is for delivering messages from MQ on Cloud to the on-premise queue manager. Select the on-premise radio-button and click ‘Next’.

The host and port of your destination are those of your local queue manager. (In my case a docker image IP address, and port 1414).

Note: If you are using a docker image, the following command can be used to find the local IP address of your image:

```
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}
{{end}}' container_name_or_id
```

Click ‘Next’.

You will not be using TLS for now, so select “TCP” for the protocol, and do not upload certificates.

Click ‘Next’.

Select “none” when asked what authentication your destination enforces.

Click ‘Next’.

If you want to add IP table rules to permit traffic only from your MQ on Cloud, you can do that here.

Finally give your destination a name and create it.

Follow the same procedure for a second destination, but this time make it a Cloud destination. The resource hostname and port come from the hostname and listener port fields in the connection information file you downloaded earlier.

The client port is the port on which the client will be listening when it is started. It will probably be 1415 because the client - on startup - looks for available ports to use starting at 1414, and will find the queue manager already using that port. Enter 1415 for now, but when you start the client (next section) look for the message "`Creating TCP server for reverse connections. Listening on port 1415`". If the port is not 1415, return and update it by editing the destination configuration.

The created destinations appear as grey panels, showing their names. There are icons in those panels, and the 'wheel' icon will show properties. The properties of the Cloud To on-prem destination will show a host and port which the Secure Gateway has generated to allow your MQ On Cloud queue manager to connect.

If a connection cannot be established, there will be a red hand icon in the top right - do not worry about that for now, as we have not configured a client yet.

Now click the title 'Clients(0), and the big "+" to connect a client.

Here you have the choice of downloading a native client, or a docker image. If your system supports the native client, that is the easier option. Follow either the instructions for downloading and running the docker image, or download the client and the IBM MQ documentation will tell you how to install and start it - depending in your operating system.

When your client starts, you should see the client connect back to the server:

```
[2018-01-23 09:04:13.830] [INFO] (Client ID 1) No password provided. The UI will
not require a password for access
[2018-01-23 09:04:13.840] [WARN] (Client ID 1) UI Server started. The UI is not
currently password protected
[2018-01-23 09:04:13.841] [INFO] (Client ID 1) Visit localhost:9003/dashboard to
view the UI.
cli> [2018-01-23 09:04:14.087] [INFO] (Client ID 12) Setting log level to INFO
[2018-01-23 09:04:16.753] [INFO] (Client ID 12) The Secure Gateway tunnel is
connected
[2018-01-23 09:04:16.931] [INFO] (Client ID MyVZM7A0Ph6_rWZ) Your Client ID is
MyVZM7A0Ph6_rWZ
MyVZM7A0Ph6_rWZ> [2018-01-23 09:04:16.943] [INFO] (Client ID MyVZM7A0Ph6_rWZ)
Creating TCP server for reverse connections. Listening on port 1415
```

Note that your command prompt is now inside a running client instance, and you have the client commands available to you. Try 'help'.

Next the on-premise gateway needs to be given permissions. In the gateway client terminal enter the following command:

```
acl allow <IP/Hostname of Local queue manager>:<Port of local queue manager>
```

The above values are the same ones that were specified within the on-premise destination. Once you have entered this command, if you had a red hand showing on your on-premise destination it should now have disappeared.

A useful command to type in the client is start with is 'l DEBUG' - which switches on debug trace.

Now you have a client running, and that command prompt is fully occupied until you use the command "quit", so leave that command prompt now, and open a new one.

You will need the localhost address of the client to configure the local outbound channel in the queue manager. For the native client this will be localhost:<port> where port is the listener port shown in the client startup trace. If you are using a docker image for the client, then use the docker inspect command again to find its IP address.

```
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}
{{end}}' Gateway_container_name_or_id
```

6. CONFIGURING QUEUE MANAGERS USING RUNMQSC COMMANDS

Section 2.8 covers configuring the queue managers using the web console. The web console is an easier way to configure the queues. This section covers using runmqsc commands which will be more familiar to MQ administrators. These commands can be copied and pasted with occasional values substituted in where shown. Any places where substitution is required is shown within < > parentheses. Ensure you remove these parentheses when you substitute in the correct variable.

The runmqsc terminal needs to be run either in the docker image for the local queue manager, or just on the local machine command prompt if you do not have a docker image.

Start by running the command tool:

```
runmqsc
```

This will enable us to administer the queue manager via this bash terminal

Create a sender channel. This will allow us to send messages to the gateway. Run the following command:

```
define channel(ON_PREM.ON_CLOUD) chltype(sdr)
conname('<IP_ADDRESS_OF_GATEWAY_CLIENT >(1414)') xmitq(LOCALXMITQ) trptype(tcp)
```

Create a receiver channel, this channel will receive messages, run:

```
define channel(ON_CLOUD.ON_PREM) chltype(RCVR) trptype(TCP)
```

This will create a new receiver channel for you.

We now have a sender channel and receiver channel, however they are unable to communicate until we specify their channel authentication records. Run the following two commands:

```
SET CHLAUTH(ON_PREM.ON_CLOUD) TYPE (ADDRESSMAP) ADDRESS('*') USERSRC(CHANNEL)
SET CHLAUTH(ON_CLOUD.ON_PREM) TYPE (ADDRESSMAP) ADDRESS('*') USERSRC(CHANNEL)
```

Next we need to create the transmission queue. We specified “LOCALXMITQ” as the transmission queue when we created our sender channel. Run this command to create it:

```
define QLOCAL (LOCALXMITQ) Usage(XMITQ)
```

Next we need to create a remote queue, set the remote queue manager name to be the same as the name as your IBM Cloud Queue Manager .

```
define QRemote(TOCLOUDQ) XMITQ(LOCALXMITQ) RNAME(FROMPREMQ) RQMNAME(<YOUR CLOUD
QM NAME>)
```

Create a local queue to receive messages by running:

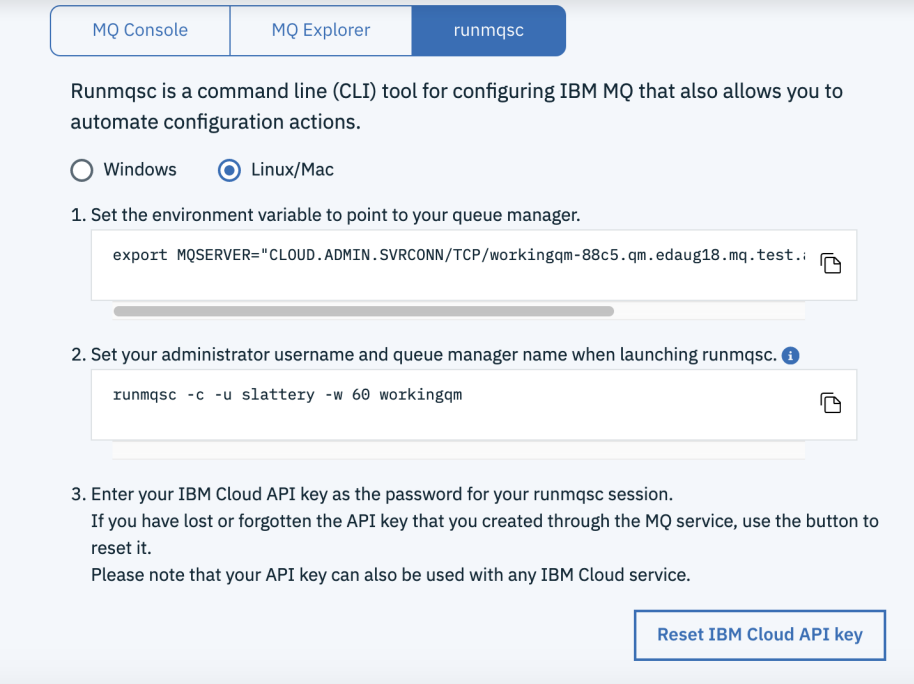
```
define QLOCAL(FROMCLOUDQ)
```

You have now configured the local queue manager, and can exit from runmqsc, or keep the session open for later.

```
exit
```

Next configure the remote queue manager

In the following steps you will require a platform API key file. You may already have one of these , but you can reset the key if you have lost it. This file can be downloaded from the administration panel of your queue manager on the IBM cloud console. Select “runmqsc” then “Reset IBM Cloud API key” as shown below.



Note: You will be configuring the remote queue manager from a local runmqsc for which you will need the runmqsc client. If you do not have it, it is available for download from the following link:

https://developer.ibm.com/messaging/mq-downloads/

First, export the server name for the cloud queue manager:

```
export MQSERVER="CLOUD.ADMIN.SVRCONN/TCP/<CLOUD_HOSTNAME>(<CLOUD_PORT>)"
```

This will target the console at the on-cloud queue manager
The hostname and port are specified in the connection info file downloaded earlier.

```
runmqsc -c -u <MQ USERNAME> -w60 <QUEUE_MANAGER_NAME>
```

This will start a window to administer your MQ on Cloud queue manager, the MQ Username is your cloud username.

The queue manager name is the name of your queue manager which can be found in the connection_info file.

After entering this command you will be prompted for a password. The password is the Platform API key which you downloaded earlier.

Create a receiver channel, this channel will receive messages, run:

```
define channel(ON_PREM.ON_CLOUD) chltype(RCVR) trptype(TCP)
```

This will create a new receiver channel for you. Notice that the name is the same as the sender channel on the local queue manager.

Create a sender channel. This will allow you to send messages to the gateway. Run the following command:

```
define channel(ON_CLOUD.ON_PREM) chltype(sdr)
conname('<GATEWAY_DEST_HOSTNAME>(<GATEWAY_DEST PORT>)') xmitq(CLOUDXMITQ)
trptype(tcp)
```

The connection details are those published by the OnPremise destination in the secure gateway. Go back to the secure gateway panel and click the 'wheel' properties icon in the bottom right of the grey panel representing the destination, and copy the "Cloud host:port". Note that IBM MQ requires the hostname and port to be in the form host.name(port) - with braces instead of the usual colon.
We now have a sender channel and receiver channel, however they are unable to communicate until we specify their channel authentication records. Run the following two commands:

```
SET CHLAUTH(ON_PREM.ON_CLOUD) TYPE (ADDRESSMAP) ADDRESS('*') USERSRC(CHANNEL)
SET CHLAUTH(ON_CLOUD.ON_PREM) TYPE (ADDRESSMAP) ADDRESS('*') USERSRC(CHANNEL)
```

Next we need to create the transmission queue. We specified "CLOUDXMITQ" as the transmission channel when we created our sender channel. Run this command to create it:

```
define QLOCAL (CLOUDXMITQ) Usage(XMITQ)
```

Create a local queue to receive messages by running:

```
define qlocal(FROMPREMQ)
```

Next we need to create a remote queue, set the remote queue manager name to be the same as the name as your local queue Manager .

```
define QREMOTE(TOPREMQ) XMITQ(CLOUDXMITQ) RNAME(FROMCLOUDQ) RQMNAME(<YOUR LOCAL QM NAME>)
```

Your On Cloud queue manager is fully configured.

Keep the runmqsc session open, as it is needed in the next step

7. TESTING YOUR CONNECTION

In the runmqsc window for your on cloud queue manager, enter the following command:

```
START CHANNEL(ON_CLOUD.ON_PREM)
```

This starts the sender channel on the on cloud queue manager.

Now go to a runmqsc session on your local queue manager, and start the sender on the local end:

```
START CHANNEL(ON_PREM.ON_CLOUD)
```

Now refresh any and all MQ on Cloud browser windows you currently have open so that they pick up the changes made. Then enter the MQ on Cloud web console.

You can use the standard amqsput tool to send messages from the local queue manager to the remote one, and you can use the web console to send messages back from the remote queue manager to the local one. Check that the messages arrive in the expected queues.

```
./opt/mqm/samp/bin/amqsput TOCLOUDQ <YOUR LOCAL QUEUE MANAGER NAME>
my new message <enter><enter>
```

8. USING THE CONSOLE TO CONFIGURE QUEUE MANAGERS

This section is an alternative to the above sections 2.6 and 2.7

In order to complete this section, you must have MQ Consoles open on both the local and cloud based queue managers.

In the MQ Console window belonging to the local queue manager, click “+” in the “channels” panel.

Enter a memorable channel name (I called mine “ON_PREM.ON_CLOUD’).

The channel type is Sender - the local queue manager is going to initiate the send.

The ‘conn name’ is the IP address of the gateway client. This will be localhost or the IP address of the docker image, depending on how you installed the client. The port will be the “listening on” port shown on startup of the client - probably 1415 Enter the value in the following format:

```
172.17.0.3(1415)
```

The transmission queue is the queue which the channel will use to send messages - you have not created that yet, so just give it a memorable name (mine was ‘LOCALXMITQ’).

Now create another channel, called “ON_CLOUD.ON_PREM, of type “Receiver”. The other parameters are all the same as above - there is no IP address or port - this is for incoming messages.

These two channels must be authorised to communicate.

If you do not already have a panel headed “Channel Authentication Records” then click on the ‘Add Widget’ icon at the top of the page, and choose the “Channel Authentication Records” panel.

Add a new authentication record for each channel, by clicking the plus icon (+) in the new panel.

Select ‘Allow’ and from the identity list choose “Address”, and click “Next”.

The channel profile is the name of the channel (ON_PREM.ON_CLOUD), and the address is “*” (without the inverted commas). Select 'Channel' for user source and leave the MCA User ID field blank. Click Next. It is not necessary to add anything in either the Description or the Custom fields. Now click "Create".

Do the same for the cloud inbound channel (ON_CLOUD.ON_PREM)

Next you will create the queues:

Add a new local queue called “LOCALXMITQ”. When it is created, click on the ‘properties’ icon , and set the usage to “Transmission”. This will be the transmission queue for outbound messages to the cloud.

Add a new remote queue called “TOCLOUDQ”.

In the properties panel - the remote queue name is “FROMCLOUDQ”. The remote queue manager is the name of your IBM MQ On Cloud queue manager. The transmission queue is “LOCALXMITQ”

Create a new local queue called “FROMCLOUDQ”.

These are all the artefacts you need locally, next you must add the corresponding queues and channels in the MQ On Cloud queue manager.

Go back to the web console belonging to the MQ On Cloud queue manager.

Create a channel called “ON_PREM.ON_CLOUD” of type ‘receiver’ - to match the ‘sender’ on the local side.

Create a channel called “ON_CLOUD.ON_PREM” of type ‘sender’ - to match the receiver at the On-Prem side.

The transmission queue is “CLOUDXMITQ”. The connection details are those published by the OnPremise destination in the secure gateway. Go back to the secure gateway panel and click the ‘wheel’ properties icon in the bottom right of the grey panel representing the destination, and copy the “Cloud host:port”. Note that IBM MQ requires the hostname and port to be in the form host.name(port) - with braces instead of the usual colon.

Add the same two authentication records as you did for your local queue manager channels.

Create your CLOUDXMITQ transmission queue.

Add a new local queue to your on-cloud queue manager called “FROMPREMQ” to receive messages from the local queue manager.

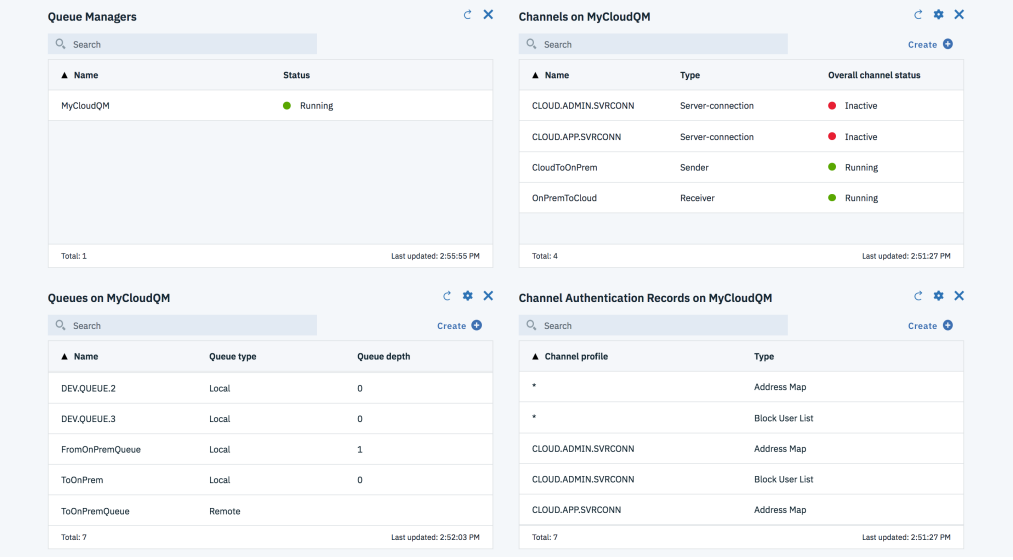
Add a new remote queue to your on-cloud queue manager called “TOPREMQ” to send messages.Edit its properties, and set the Remote Queue to “FROMCLOUDQ”. Set the remote queue manager to the name of your on-premises queue manager. The transmission queue should be CLOUDXMITQ.

You should now have all the pieces in place to run the system without TLS security.

In each console, select the sender channel, and click the ‘start’. You should see them bind, then start.

With these two channels started, you should be able to put a message on the TOCLOUDQ in the local queue manager, and see it appear in the cloud queue manager FROMPREMQ. You can put a message on the queue with using the web console (using the ‘down-arrow’ icon in the queue panel)
(Note you have to refresh the panel in the remote console to see the change).

You should also be able to put a message on the TOPREMQ in the cloud queue manager, and see it appear in the FROMCLOUDQ on the local server.



9. INSTALLING MUTUAL (TWO WAY) TLS SECURITY

The two queue managers can be configured to use TLS security end-to-end through the secure gateway without terminating the TLS at the gateway.

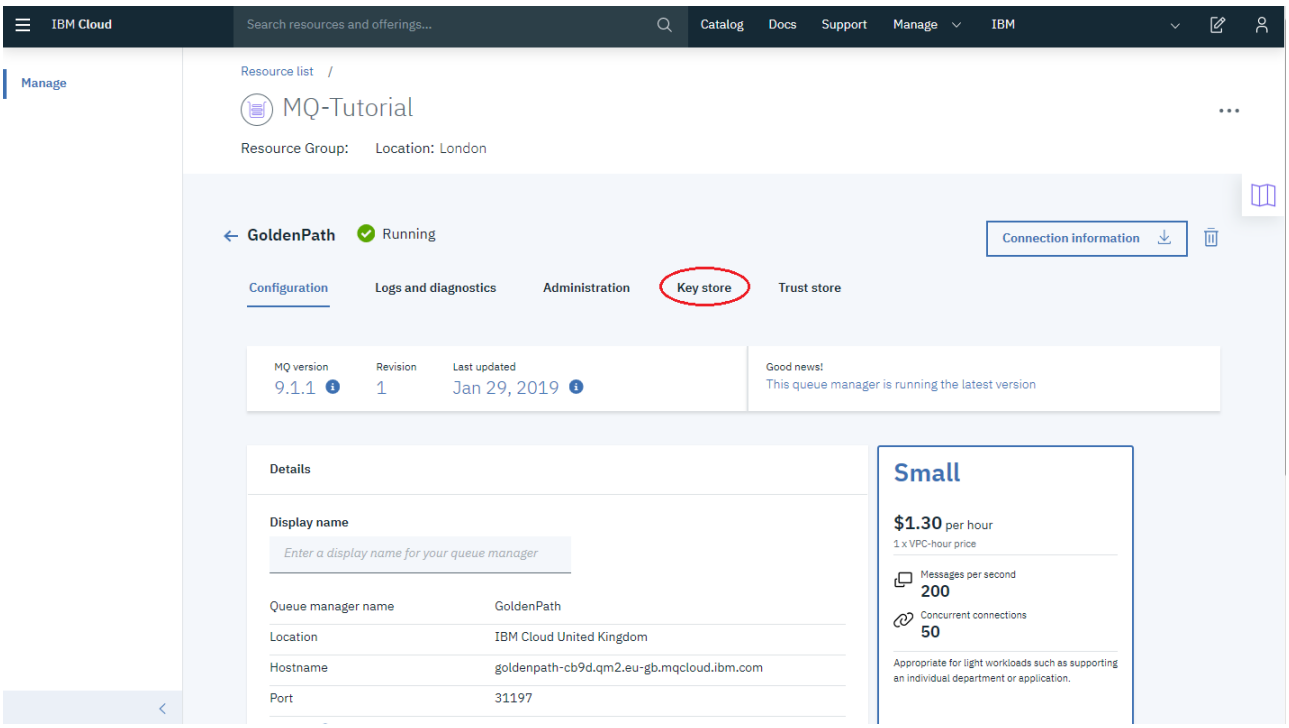
You will need a command prompt to access the MQ command line interface for the local queue manager as you will be importing certificates into the key.kdb associated with your queue manager.

The message conversations are initiated by both ends of the flow, so you will need to put the certificate from the local queue manager into the trust store of the cloud queue manager, and also insert the certificate from the cloud queue manager into the key.kdb for the local queue manager. You also need to set a cipher spec on both ends of both the channels.

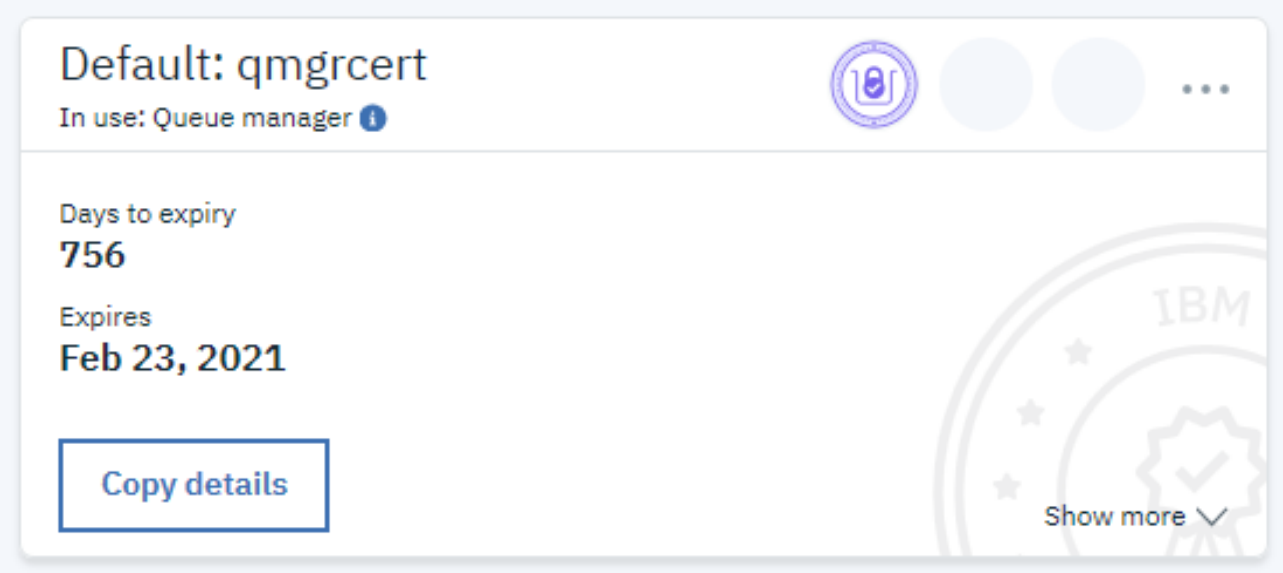
The local queue manager will require the certificate from the IBM Cloud queue manager, but it will also require the whole chain of certificates up to the root. You probably have the Digicert Root certificate anyway, but it can be found in the MQ on Cloud queue manager trust store.

In the MQ On Cloud console Queue Manager details page, there is a tab labelled “Key Store”. Selecting this tab shows the certificates currently in use by your queue manager. Clicking on the three dots menu in the titlebar of a certificate allows you to download the public part to a file. This action will also check whether the certificate is self-signed, and if not, it will download the corresponding key chain up to the issuer of the certificate into the same file.

Alternatively, you could open the IBM MQ Console in a browser (for example FireFox) which supports downloading certificates.



A certificate labeled qmgrcert can be seen, click the three dots in the corner of the certificate, then on the download public certificate button. This can be seen below



A public certificate called “qmgrcert.pem” will be downloaded.

Next a trust store for the local queue manager needs to be created and the public certificate added so that it knows to trust the on-cloud queue manager.

Navigate to the directory “ssl” with the following command

```
cd var/mqm/qmgrs/<YOUR LOCAL QUEUE MANAGER NAME>/ssl
```

Note: the ssl directory is where the key.kdb file will be by default, but your queue manager may have a different path. You can check where the queue manager expects to find the kdb file using the “DISPLAY QMGR ALL” command in runmqsc and checking the SSLKEYR value.

Run the following command to create a key database on the local queue manager - if you do not already have one.

```
runmqakm -keydb -create -db key.kdb -pw <ANY PASSWORD> -stash
```

Add the on cloud certificate to the database by running the following command

```
runmqakm -cert -add -db key.kdb -stashed -label ibmwebspheremq<your cloud
qmname>     -file  <your downloadedfile>
```

Your certificate is now added to the trust store of the local queue manager. Now it is time to pass a local certificate into the on cloud queue manager.

Create a self-signed certificate and add it to the key database using the following command:

```
runmqakm -cert -create -db key.kdb -stashed -dn "CN=onprem,O=IBM,C=GB" -label
ibmwebspheremqonprem
```

Extract the public part of the new certificate by using the following command:

```
runmqakm -cert -extract -db key.kdb -stashed -label ibmwebspheremqonprem -file
cert.pem
```

Navigate in your browser to “trust store” on your MQ on cloud page. It is located next to the keystore tab we accessed earlier. Click the “import certificate” button, browse to the certificate you just exported into cert.pem and upload it to the queue manager.

Now we have all the certificates exchanged, so we need to switch on TLS and tell the queue managers which certificate to use.

Firstly in the local queue manager , runmqsc and DISPLAY QMGR ALL. Check if the value of CERTLABL is the correct value (mine is ibmwebspheremqonprem) . If not, then set it with
ALTER QMGR CERTLABL(‘ibmwebspheremqonprem’).

The rest of the process can be done in the two web consoles.

Go to the local web console, and select the ON_PREM.ON_CLOUD sender channel. In the SSL section, set the cipher spec to ANY_TLS12 (or any other supported spec you require).
Set the certificate to be ibmwebsperemqonprem. This is the certificate served up by the sender channel for the receiver to trust.

For the ON_CLOUD.ON_PREM channel, set the cipher spec to ANY_TLS12, set the ssl authentication to be “Required”, and set the cert label also to be ibmwebspheremqonprem.

Now select the queue manager in the queue managers widget, and from the “three dots” menu choose “Refresh Security” and click on SSL to complete the refresh.

Now go to the web console of the cloud queue manager, and repeat the same procedure for the two channels. Remember that the ON_PREM.ON_CLOUD channel at this end is the receiver, and the ON_CLOUD_ON_PREM channel is the sender. The certificate name on this end of the channels will be “qmgrcert” - That is the default certificate supplied with MQ on Cloud queue managers - if you have swapped in a different certificate for your queue manager - use the name of that one instead.

Now refresh the security for the cloud queue manager, and wait a short while while the security is refreshed. You should now be able to start the sender channels using the web consoles, and send TLS secured messages between the two queue managers.

10. APPENDIX:IF THE TLS ENABLED CHANNELS DO NOT START

If the channels do not start, then the certificate exchange has probably failed. There are some good tools to help you locate the issues which might have caused this.

First, examine your certificate pem files using openssl:

openssl x509 -in <pemfile> -text -noout

Check that the certificates are all readable, and that you have a full chain of certificates available. If you have a root certificate, make sure that it has a “Basic Constraints: CA:TRUE”.

Check the certificate chain in the key.kdb. Is there a complete chain from the certificates up to a self-signed ca certificate? Check that for each certificate which has an Issuer, the issuing certificate exists in the kdb. Check that the final certificate in the chain has an Issuer which is the same as its Subject, that that it has Basic Constraints: CA:TRUE.

runmqakm -cert -list -db key.kdb -stashed

There is a tool to validate certificates called mqcertck - the documentation is here:

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.1.0/com.ibm.mq.ref.adm.doc/q120895_.htm

There is also improved diagnostics for certificate management documented here:

https://www.ibm.com/support/pages/node/1081227?lang=en

There may also be information in the error logs for your queue manager. You can download the logs by selecting the “Logs and Diagnostics” tab in the queue manager details view.

The logs for your local qm will be located in /var/mqm/qmgrs/<qm>/errors/AMQ___.LOG - look for errors such as 575010 from above documentation.

Examine the channels using DISPLAY CHANNEL (name) and check the CERTLABL field is the name of your certificate, and that the SSLCIPH fiels is the right cipher spec.

Examine the queue manager using DISPLAY QMGR ALL and check that the SSLKEYR field has the correct location of the key.kdb file.