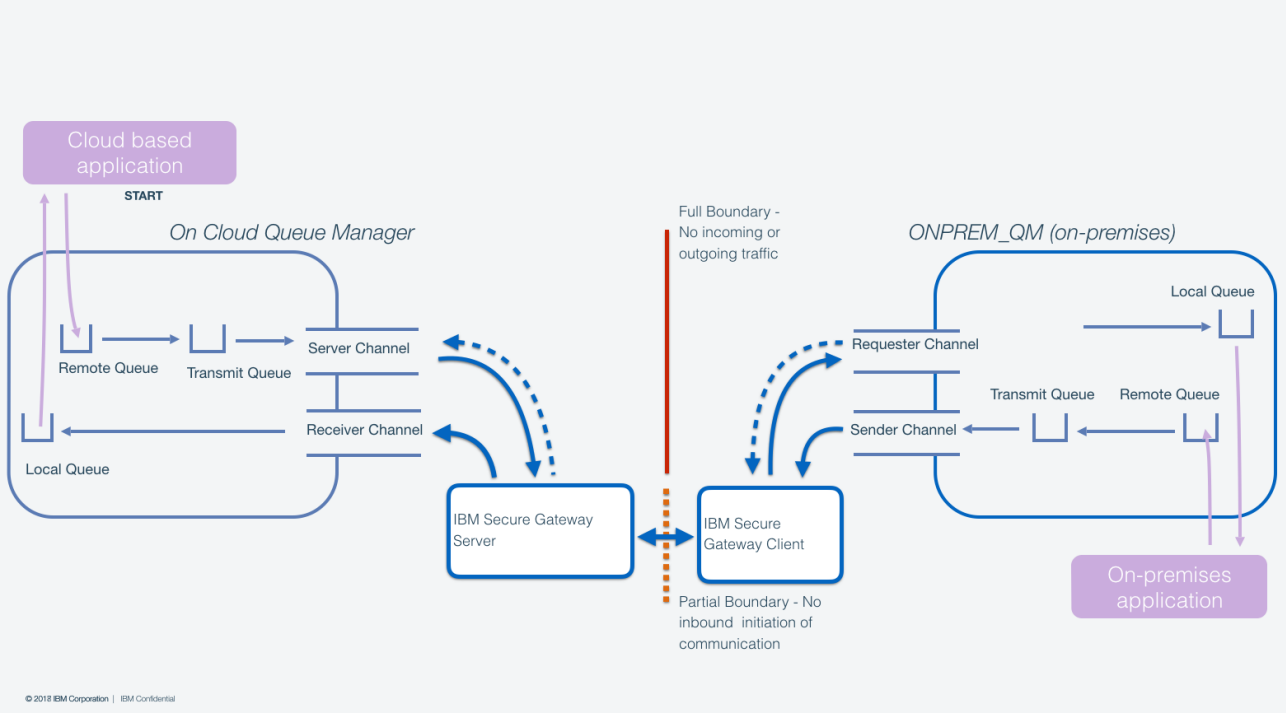
Connecting an on-premise queue manager to an IBM MQ On Cloud queue manager via the IBM Secure Gateway.



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 two sections. In the first section you will learn how to set up the two queue managers and the gateway, then configure channels to allow two-way message passing without encryption.

In the second section you will add end-to-end TLS security, using the features of the IBM MQ channels.

In the following description, commands you should type are shown in `monospace`

Any places where you should replace the text with a value from your configuration are in curly brackets {}

For example:

```
docker exec  -it {your_imageid} /bin/bash
```

2.2. INITIAL SETUP

There are two components of this system, which you will need to have installed :

1. A cloud-hosted queue manager deployed using the MQ service in IBM Cloud. In the following pages this is referred to as “MyCloudQM”.
2. An “on-premises” queue manager that will be connected to the cloud queue manager

2.3. THE STEPS TO FOLLOW

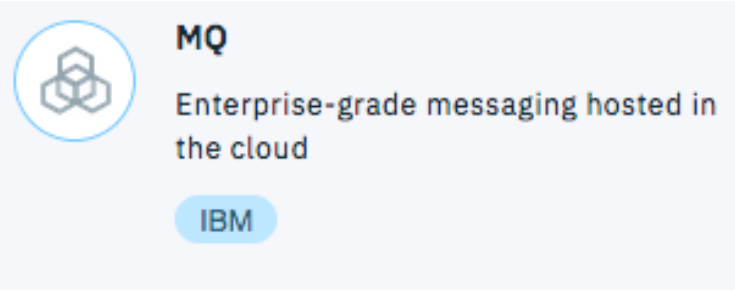
1. Create a cloud based MQOnCloud Queue Manager.
2. Create a local Docker image containing a MQ Queue Manager. (optional)
3. Create a cloud based Secure Gateway, install its client in a second Docker Image and configure the Secure Gateway to link the two Queue Managers.
4. Create channels to pass messages between the two Queue Managers.
5. Install TLS security.

2.4. 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 should see 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 permissions" and "application permissions". Click the "user permissions" 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 permissions 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 “Adminstration” from the menu. Your login details will be displayed, and you have to opportunity to reset your API key if you do not know it.

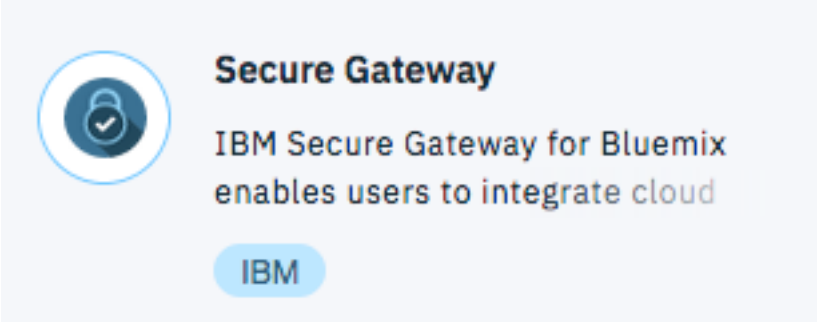
Finally select the “Launch MQ Console” button - enter the credentials and log in.

2.5. CREATE A LOCAL QUEUE MANAGER

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

2.6. CREATE THE CLOUD-BASED SECURE GATEWAY

In a browser, go back you your IBM cloud catalog, and from the ‘integrate’ group select the ‘IBM Secure gateway’. Click ‘Create’. The number of destinations and clients is zero, as you have not configured any yet.



Select the big “+” under destinations. You will be asked to choose whether the destination is in the cloud or on-premise. You are building a system in which all the transactions are initiated by the on-premise queue manager, so the destination is “Cloud”. Select that radio-button, and see that the schema changes . Click ‘Next’

Fill in the resource host name, port and client port. The first two come from the server connection information you downloaded earlier. The client port is 1414 as configured in your local queue manager. Click ‘Next’

Select TCP as the protocol. Click ‘Next’
Select ‘None’ for authentication. Click ‘Next’
Give your destination a name, and click ‘Add Destination’.

Now click the title ‘Clients(0), and the big “+” to connect a client.

When asked how you would like to connect this gateway, click on the Docker image.

Open a new command prompt on your local machine, and paste the first command from the secure gateway window into the command prompt and press enter.

You should see the client start up and 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 1414
```

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

The most 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.

In the new window type:

```
docker ps
```

You should see the process corresponding to the secure gateway client.

You will need the docker internal IP address of the client to configure the queue manager. As you have just typed ‘docker ps’, you have the ID of the gateway image on the screen.

Type:

```
docker inspect {gateway_container_id}
```

Somewhere near the end of the inspected information, you will see the IP address:

```
….
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "IPAddress": "172.17.0.3",
        "IPPrefixLen": 16,
….
```

2.7. CREATE CHANNELS TO PASS MESSAGES BETWEEN THE TWO QUEUE MANAGERS.

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 “CloudOutboundChannel’).

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 - with port 1414 - in the following format:

```
172.17.0.2(1414)
```

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 ‘ToCloud’).

Now create another channel, called “CloudInboundChannel, of type “Requester”. The other parameters are all the same as above.

These two channels must be authorised to communicate. - 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 (CloudOutboundChannel or InboundChannel), and the address is “*” (without the inverted commas). Now click “Create”.

Do the same for the cloud inbound channel.

Next you will create the queues:

Add a new local queue called “ToCloud”.

When it is created, click on the ‘properties’ icon , and set the usage to “Transmission”.

Add a new remote queue called “MyCloudQueue”.

In the properties panel, the remote queue name is also “MyCloudQueue”.

The remote queue manager is the name of your IBM MQ On Cloud queue manager.

The transmission queue is “ToCloud”

Create a new local queue called “MyLocalQueue”.

Create a new local queue called “ToClient”.

Next you must add the corresponding queues 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 “CloudOutboundChannel of type ‘receiver’ - to match the ‘sender’ on the local side.

Create a channel called “CloudInboundChannel’ of type ‘server’ - to serve messages on request from the local ‘requester’ channel. The transmission queue is “ToClient”.

Add the same two authentication records you added to the local queue manager.

Add a new local queue called “ToClient”. Edit its properties and set the usage to ‘transmission’.

Add a new local queue called “MyCloudQueue” to pick up messages sent from the local queue manager.

Add a new remote queue called “MyLocalQueue” to hold messages to be accepted by the local requester. Edit its properties, and set the Remote Queue to “MyLocalQueue”.

Set the remote queue manager to the name of your on-premises queue manager and set the transmission queue to “ToClient” and save.

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

Go back to the web console for the local queue manager - highlight the CloudOutboundChannel channel, and click the ‘start’. You should see it bind, then start.

Do the same with the requester channel (It should also start).

With these two channels started, you should be able to put a message on the MyCloudQueue in the local queue manager, and see it appear in the cloud queue manager. You can put a message on the queue with using the web console (using the 'envelope' 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 MyLocalQueue in the cloud queue manager, and see it appear in the MyLocalQueue on the local server.

2.8. INSTALLING END TO END 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 as the console does not yet support one of the features you need. All this configuration can be done from the local side using mqsc and

the command prompt, as the cloud queue manager already has a running admin channel which the local queue manager can talk to.

The message conversations are initiated by the on-premise queue manager, so it needs the MQ On Cloud certificate. You also need to set a cipher spec on both ends of both the channels.

To get the cloud side root certificate, go back to the cloud MQ console in your browser. Click the green padlock in the web address field, and view the certificate. The browser shows a chain of certificates up to the root - and it is that one you need.

Select the root certificate and export it to your directory which you can access from the local queue manager.

Next either create a key store , or add the certificate to the key store.

```
cd /var/mqm/qmgrs/QM1/ssl

ls
```

If you have a `key.kbd` already, skip the next step.

```
runmqakm -keydb -create -db key.kdb -pw {choose a password} -stash
runmqakm -cert -add -db key.kdb -stashed -label DigiCertRootCA -file /tmp/
DigiCertGlobalRootCA.crt
```

Verify that your key is present:

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

Change the files to be owned by mqm:

```
ls -l
chown mqm:mqm *
```

Now you have the certificate filed away, you need to give each end of the channels a cipher spec.

The cipher specs are described here:

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.sec.doc/q014260_.htm

As an example, I have chosen to use:

ECDHE_RSA_AES_128_CBC_SHA256

At the command prompt in the local queue manager docker image:

```
runmqsc {queue manager name}
alter chl('CloudOutboundChannel') chltype(sdr)
sslciph(ECDHE_RSA_AES_128_CBC_SHA256)
alter chl('CloudInboundChannel') chltype(rqstr)
sslciph(ECDHE_RSA_AES_128_CBC_SHA256)
```

Next, you will to use runmqsc on the MQ On Cloud queue manager, so quit the local one with ‘exit’.

You need to set an environment variable to say which server you are managing:

```
export MQSERVER='CLOUD.ADMIN.SVRCONN/TCP/{address of your cloud manager}
({port})'
```

Run the runmqsc command again, against the cloud queue manager:

```
runmqsc -c -u {user} {cloud_qm_name}
```

The user is the username you created for yourself earlier.

The -u will prompt for and password - this is the API key you use to access the cloud resource.

Add the same cipher specs...

```
alter chl('CloudOutboundChannel') chltype(rcvr)
sslciph(ECDHE_RSA_AES_128_CBC_SHA256)
alter chl('CloudInboundChannel') chltype(sdr)
sslciph(ECDHE_RSA_AES_128_CBC_SHA256)
exit
```

Finally, go back to each of the channels in each of the web consoles - click the 'properties' icon, and select "SSL". Check that the cipher spec is in place.

In the cloud queue manager, mark the SSL authentication as 'Optional'. This is because the local queue manager is not providing a certificate to start the process, only the cloud QM provides a certificate to the local one.

You should now be able to restart the two channels from the local side, and see TLS protected message flows.