How to setup TLS security with .NET Core Windows client.

Specifications

- IBM MQ Service on IBM Cloud V 9.3.3.2
- IBM MQ dotNet Client V 9.3.3.1
- Windows Client Machine (I used Windows 2022 Server for my testing)

Prerequisites

- Deploy IBM MQ Service in IBM Cloud. (The lite service will work)
- Deploy a Queue Manager within your Cloud MQ service
- Windows Client Machine
  - Install openssl
  - Install Visual Studio Developer Edition
  - Install IBM MQ Client V9.3.3.1

Enable SSL on your Windows Client.

You MUST have SSL policy enabled on your client machine. To ensure that you have this setup please run gpedit.exe and verify.
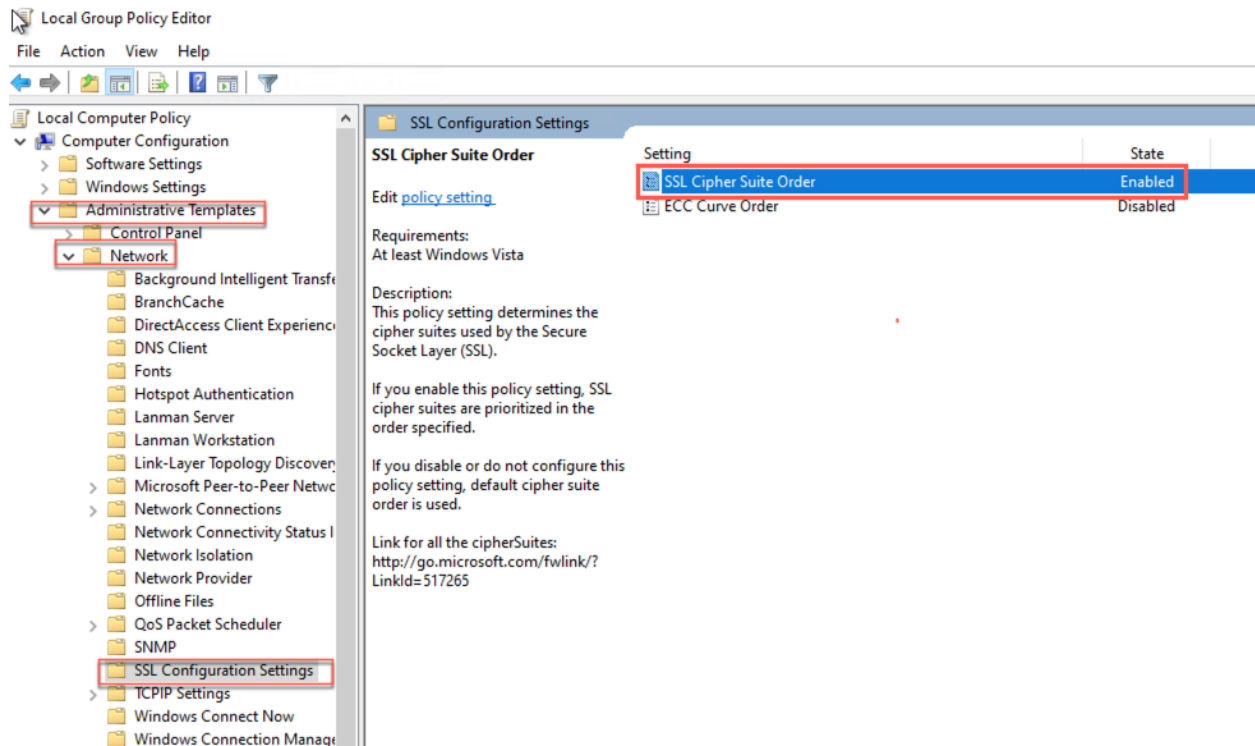
If the above setting is not enabled, you will need to enable this.
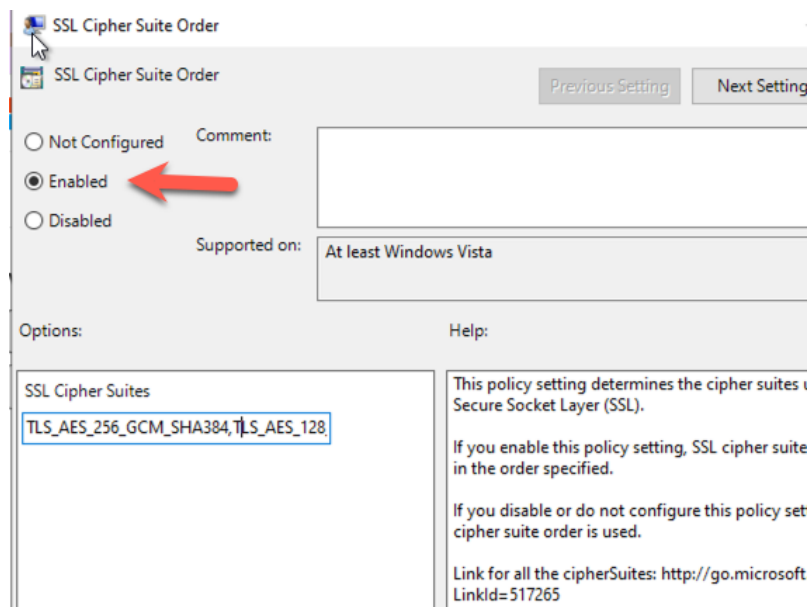
NOTE: *You can leave the default cipher Suites order field to the default setting. We just need to make sure that we are using one of the values in this filed when we make a connection with the client application.*

Create SSL Cert / Key

You will need to create a ssl cert and key using the below command. To do this you need to have openssl installed in your machine.

*** Example Command ***
openssl req -newkey rsa:2048 -nodes -keyout krierclientKey.pem -x509 -days 365 -out krierclientCert.pem

Beow is an example of the settings I used. The only thing that is important to note on this is the FQDN name. You need to set this to the fully qualified domain name (host name) of your MQ Server. You can find this information in the connection information for your queue manager in the cloud.

```
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:IA
Locality Name (eg, city) []:Granger
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IBM
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:myqmgr-4e4a.qm.us-south.mq.appdomain.cloud
Email Address []:dakrier@us.ibm.com
```

Combine the Cert / Key into a single PEM file

\# Combine the private key and public certificate into a single file
cat krierclientKey.pem > Krier.pem
cat krierclientCert.pem >> Krier.pem

Note: *if you are doing this on a windows machine, you can use a text editor to create a new file and paste the contents of the two PEM files into a NEW PEM file.*

## Create the pfx (pk12) file for the Windows Cert Store

openssl pkcs12 -export -out krier.pfx -in krier.pem

Note: *We will use this new singe PEM file with both the key and the cert on the IBM Public Cloud Queue Manager.*
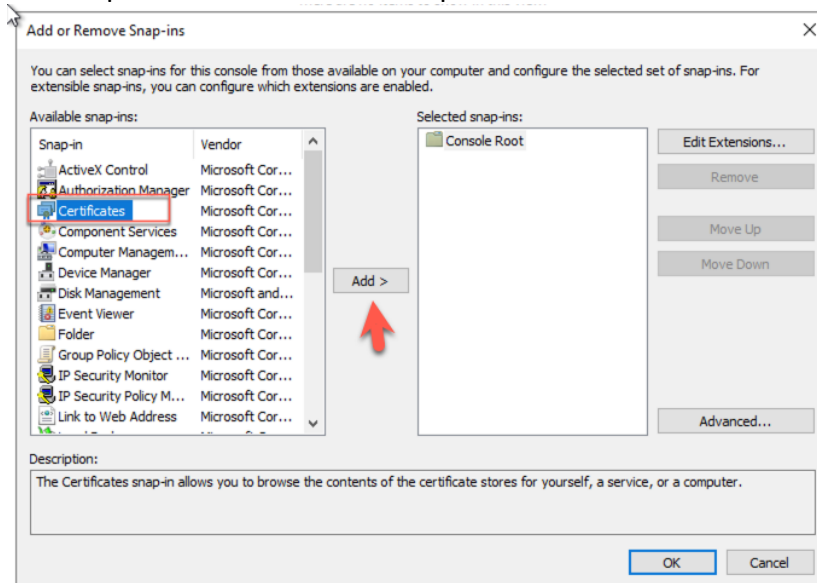
Import the pfx file into BOTH personal key store and Windows Trust Store

On your Windows client, you will need to run the MMC to modify the certificates for windows.

From the run menu, type mmc and hit enter. Open this app below.
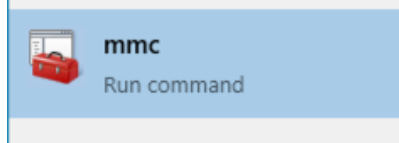
From the file menu, select Add Remove Snap In and select the Certificates from the left hand menu option. Add that to the snap in and select OK.

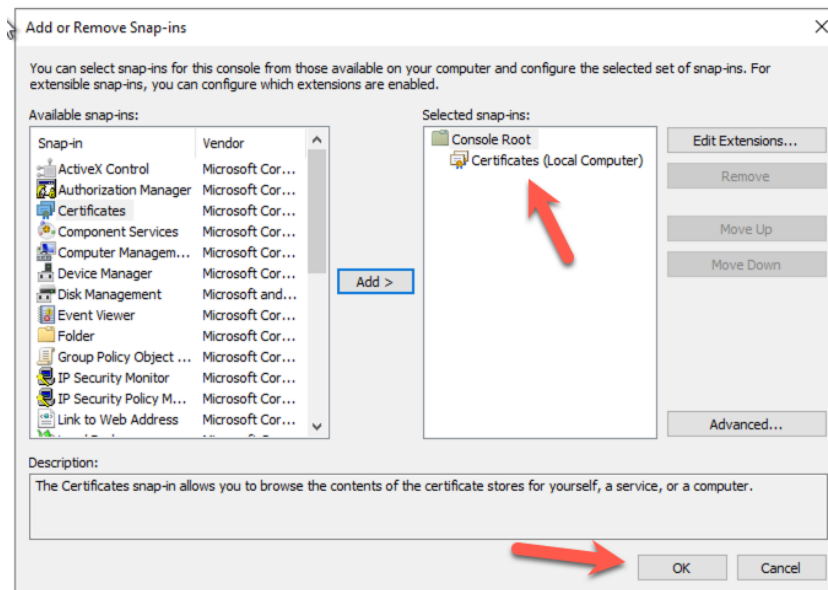In our example, we will be using the Local Computer (*SYSTEM) keystore repository.
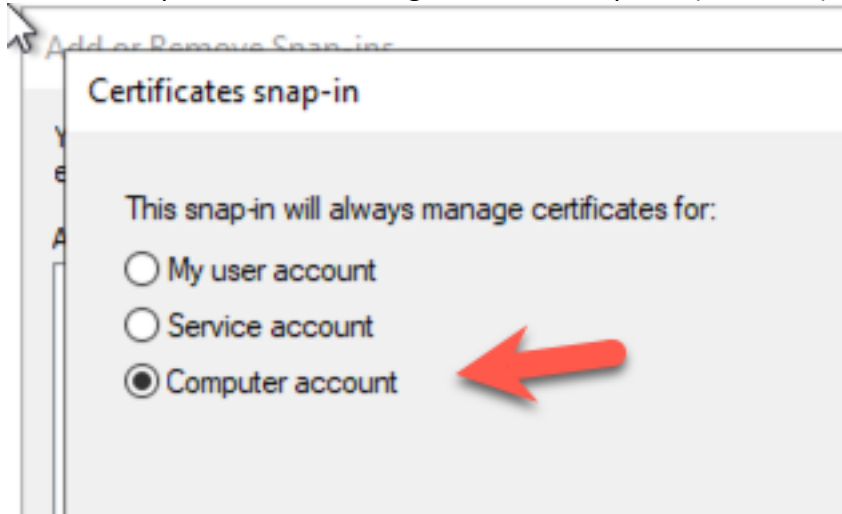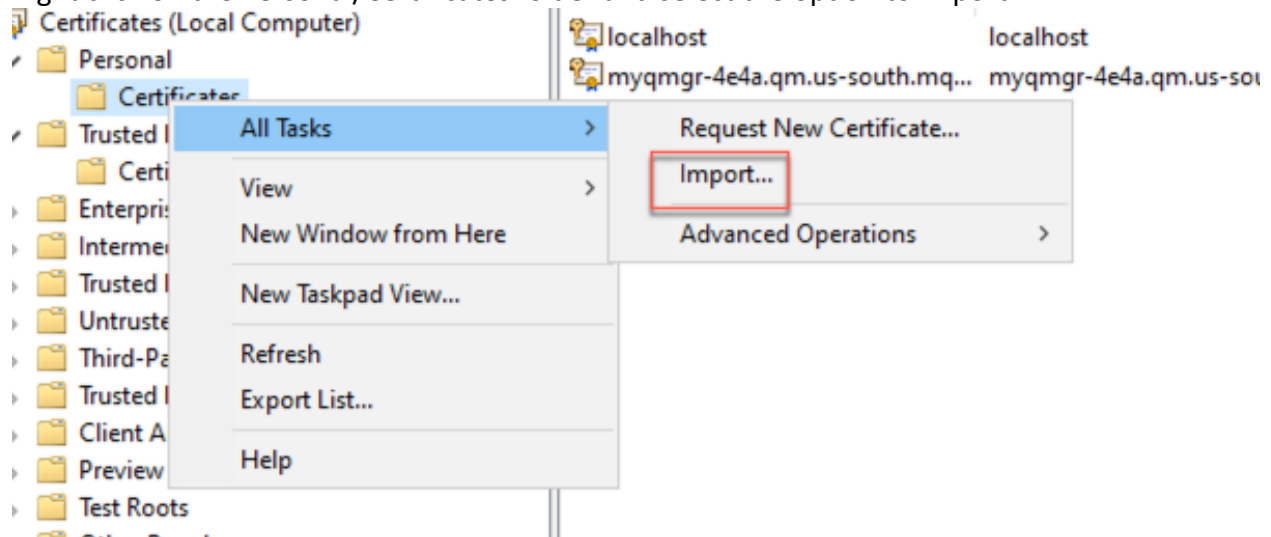
We need to add our pfx file to BOTH the personal repository and the trusted repository.
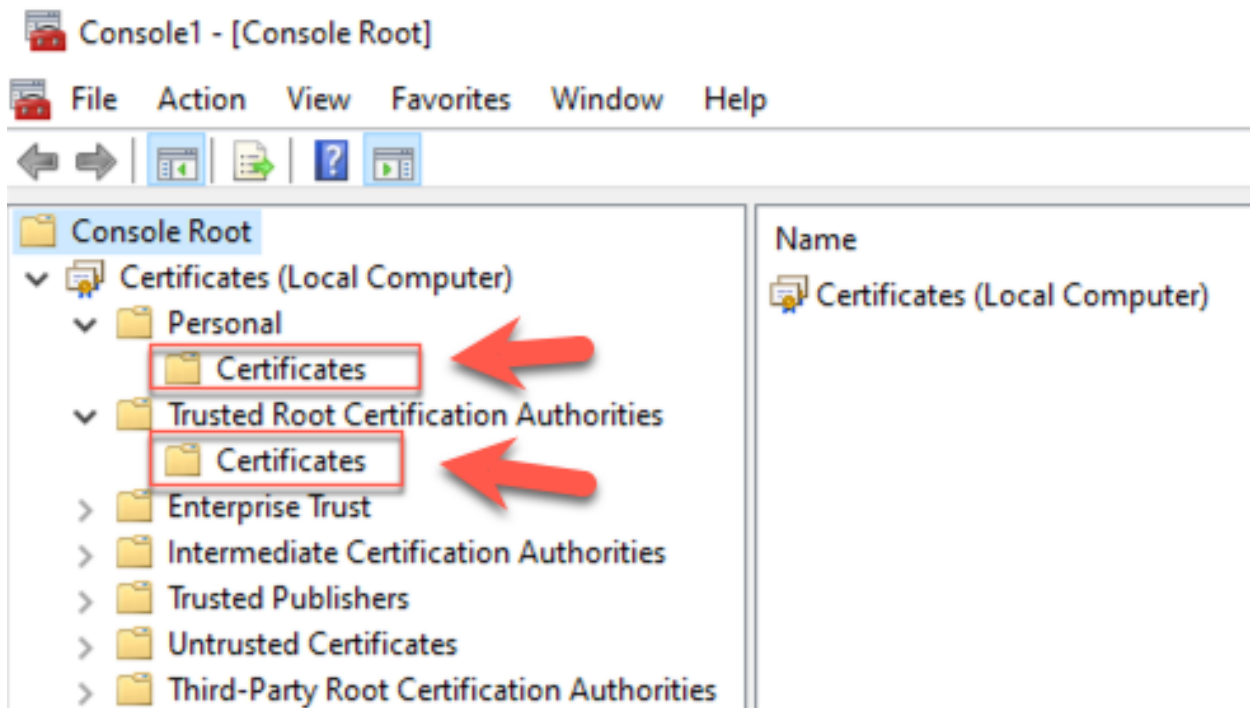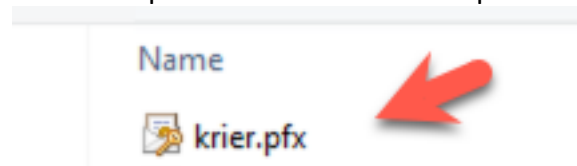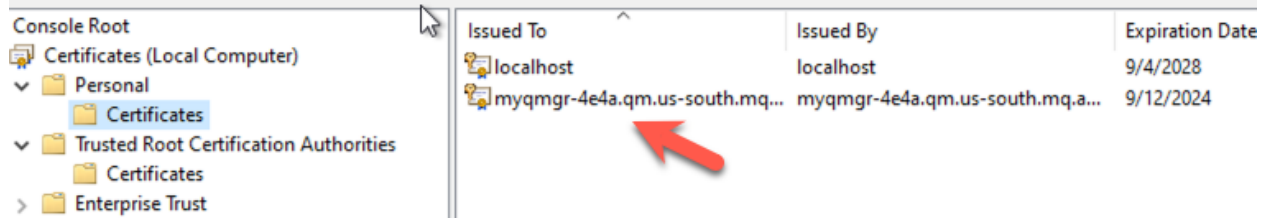
Right click on the Personal/Certificates folder and select the option to import
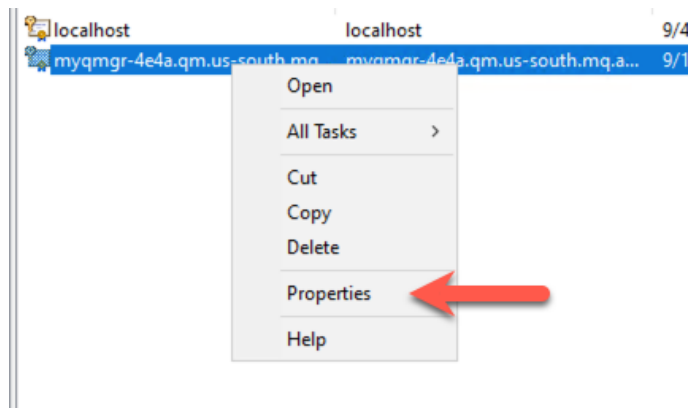

Select the pfx file we created in the previous step..

You should now see that in your personal key store.

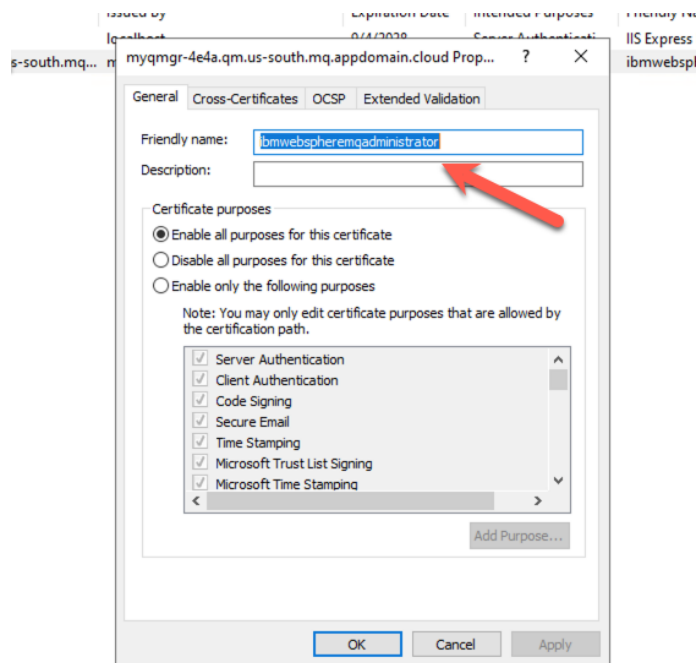Modify the label (friendly name of the cert you just imported)

Right click on the cert in your keystore. Select properties.

This is VERY IMPORTANT.. Give your cert a friendly name in this format.

ibmwebspheremq + <The USERID of your Windows User who will be running the Application>

Note: I am logged in with the administrator userid, so that's what I used in my example below.

IMPORTANT.. You need to perform the same steps to import the pfx file into the Trusted Root Certification Authorities keystore
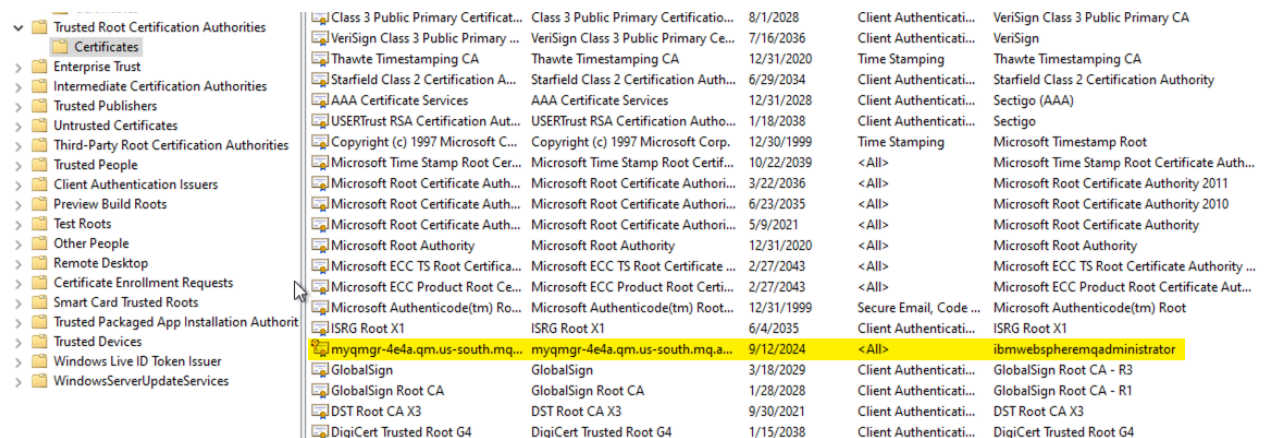
Import the PEM into the keystroke for the QM on the cloud.

Log into your queue manager on the IBM Cloud.

Select the Key Store tab from the management console

Select IMPORT
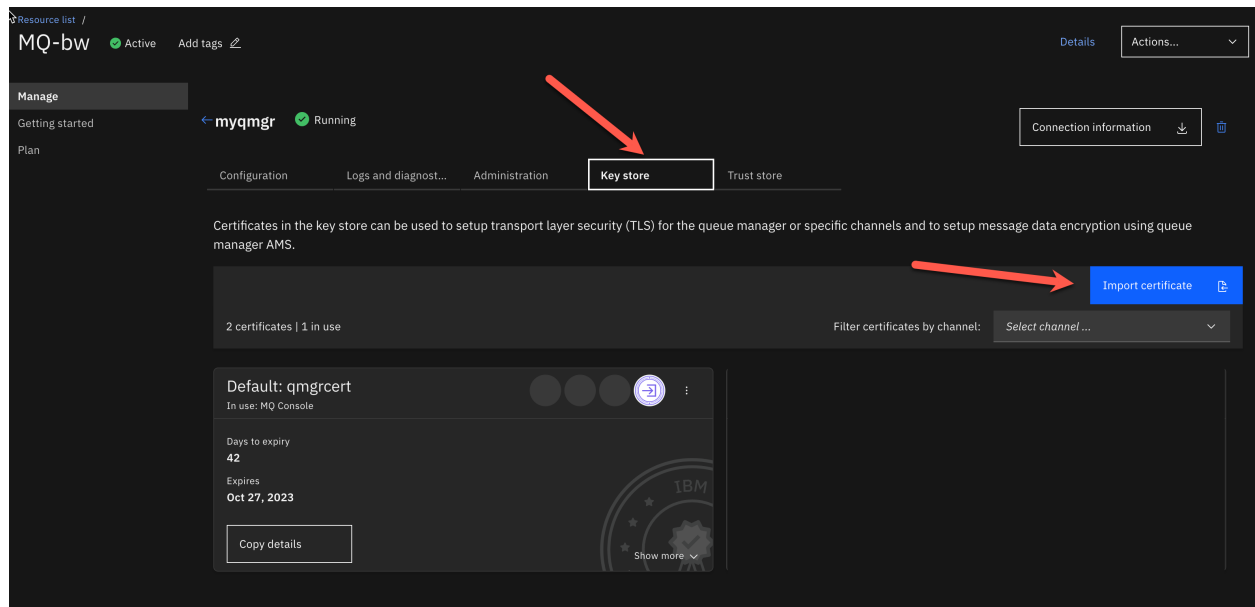
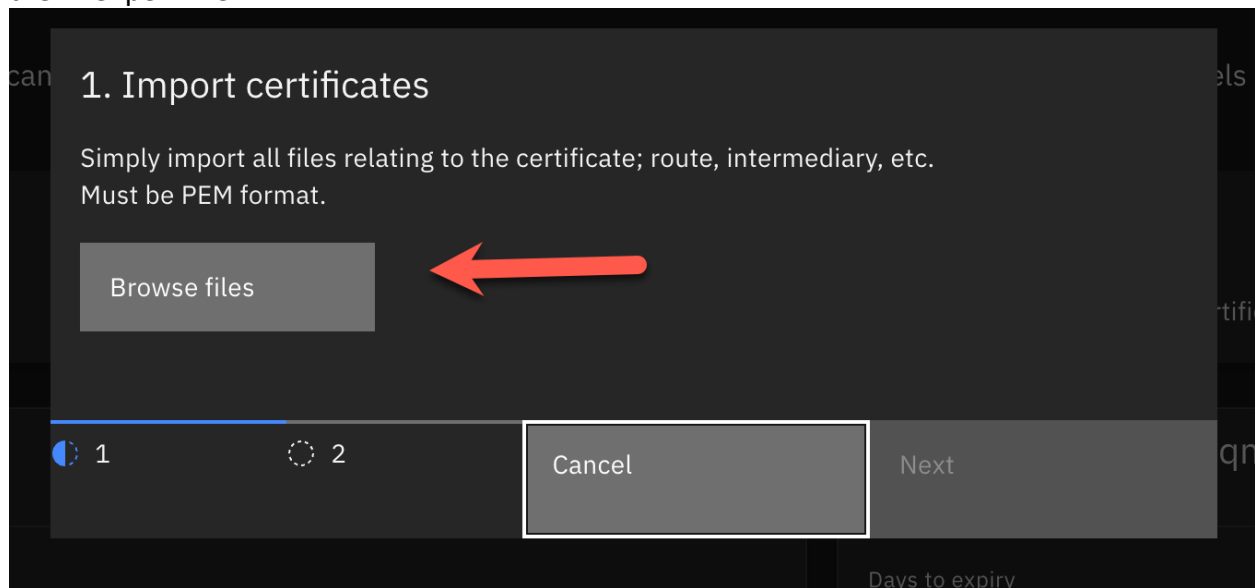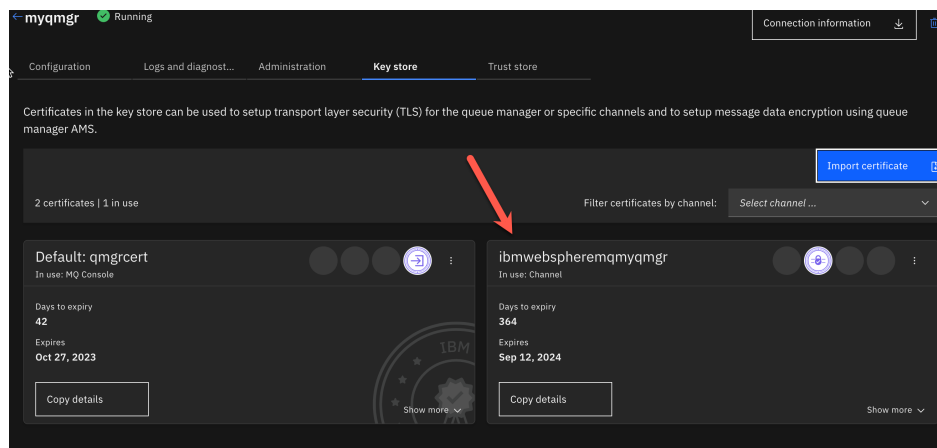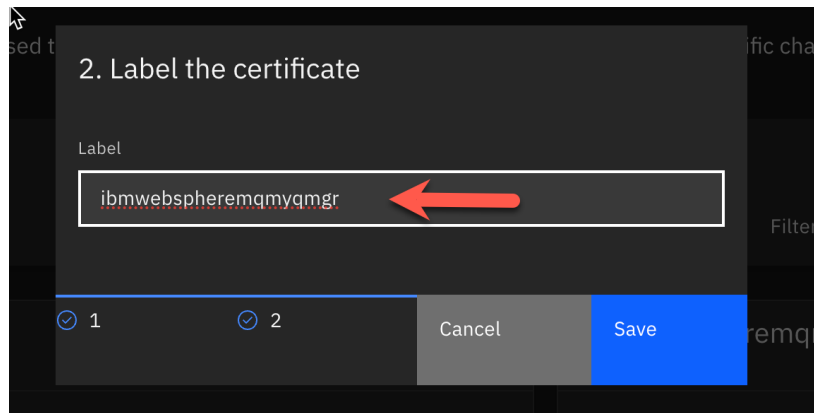Browse to locate your PEM file. This will be the PEM file with your cert & key.. In my example it's the krier.pem file.

Give your cert a name (Lable). This is important to label it correctly. .

You need to label it as follows.

ibmwebspheremq   + <Your queue manager name>

2. Label the certificate

Label

ibmwebspheremqmyqmgr

| ⊘ 1 | ⊘ 2 | Cancel | Save |

← **myqmgr** ✓ Running

Connection information ⤓ 🗑

| Configuration | Logs and diagnost... | Administration | **Key store** | Trust store |

Certificates in the key store can be used to setup transport layer security (TLS) for the queue manager or specific channels and to setup message data encryption using queue manager AMS.

Import certificate ⎘

2 certificates | 1 in use   Filter certificates by channel: *Select channel ...* ⌄

Default: qmgrcert
In use: MQ Console

Days to expiry
42

Expires
Oct 27, 2023

Copy details   Show more ⌄

ibmwebspheremqmyqmgr
In use: Channel

Days to expiry
364

Expires
Sep 12, 2024

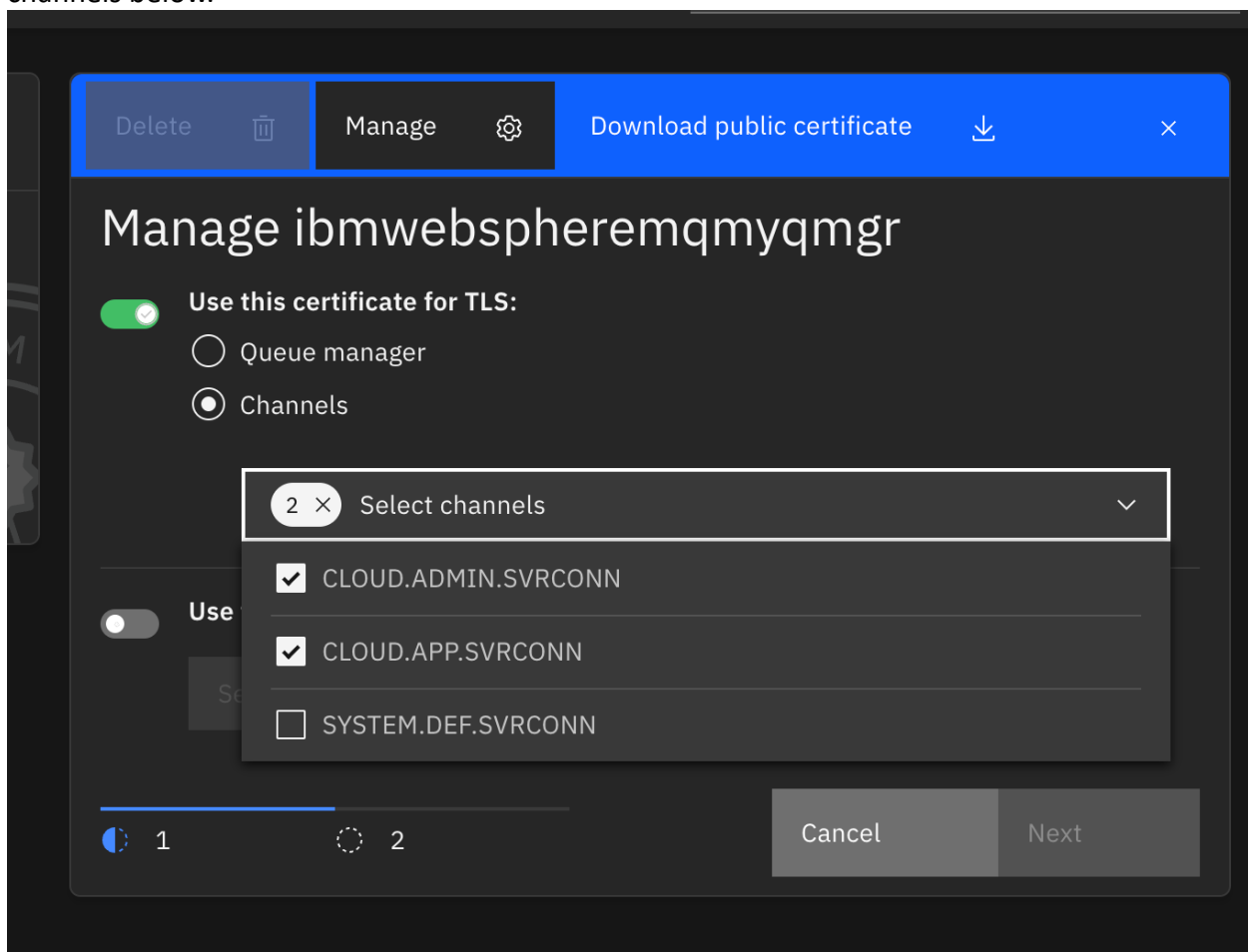Copy details   Show more ⌄

Now we need to ensure that the cert is used for our channels.

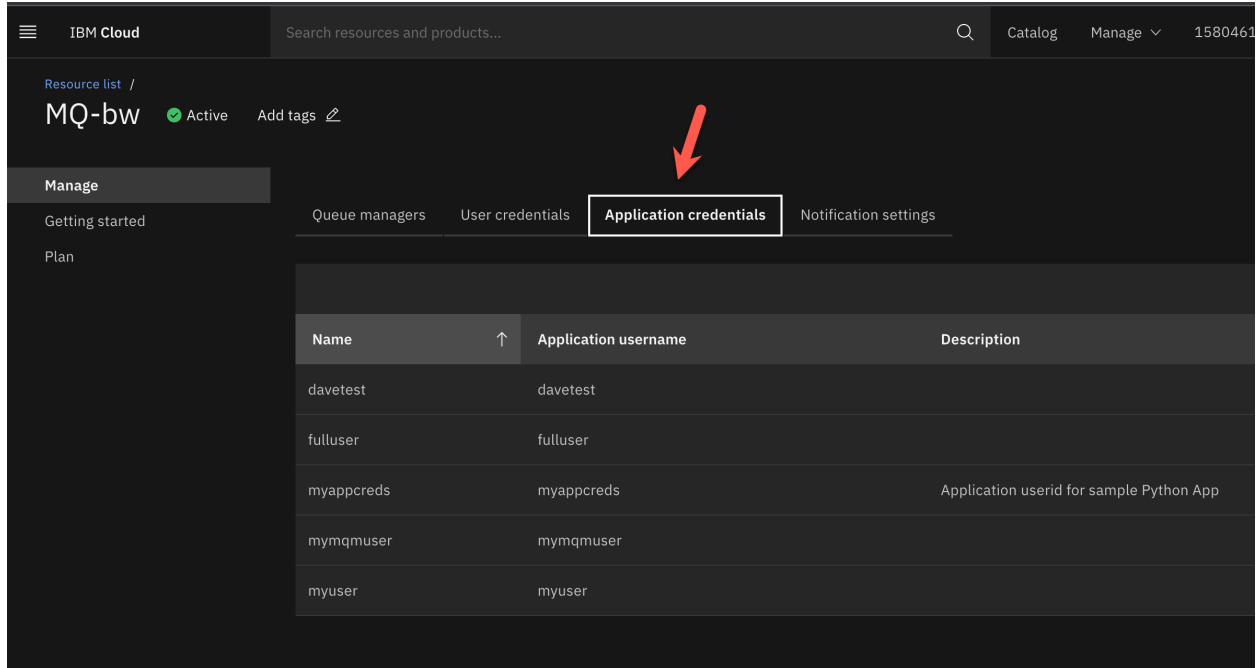Select the 3 dots for the qert to see the management button.
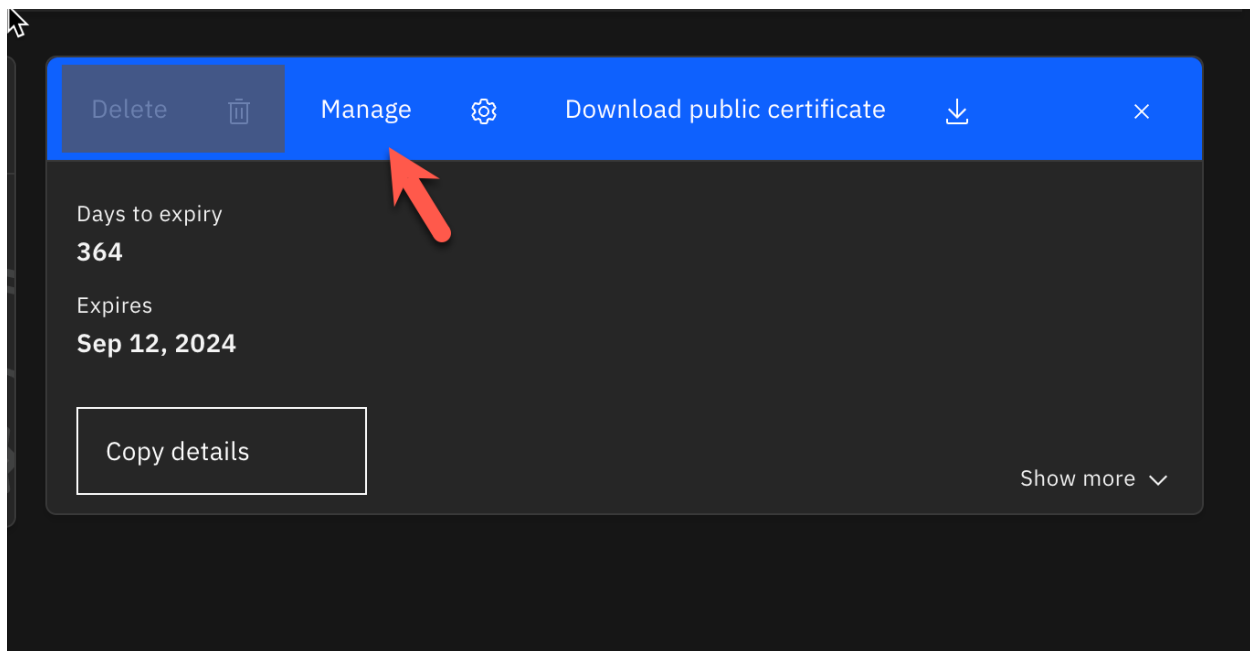
Select to use this cert for the channels you are connecting to. In our case it's both of these channels below.

Create an USERID that will be used for channel authentication.

On the initial page for your IBM Cloud MQ Service, you will have the option to create a new application credential.
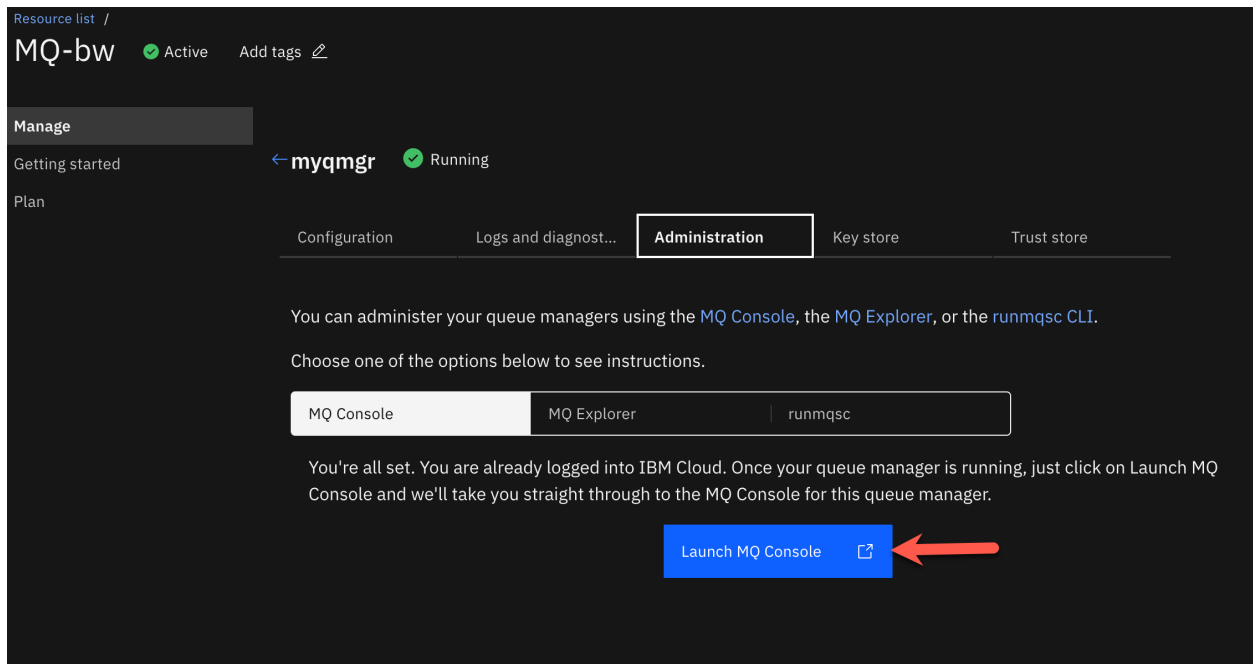

Create an application credential.

Take note of your APIKEY, that will be your password when connecting.
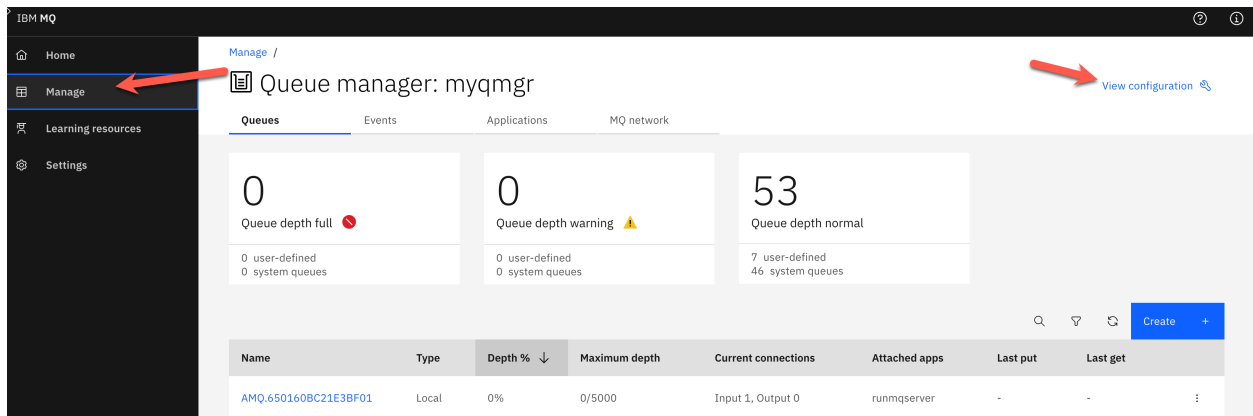
Modify the SSL of the QM qm.ini Queue Manager configuration.

Next we need to make a few updates to our queue manager settings. Log into the MQ Console

Select manage on the left hand side to bring up the management console.

Select the "view configuration" link in the upper right hand corner.

Edit the SSL settings.

Remove the cert label value and save your settings. It should be blank.

General		
Extended	SSL	
Cluster		
Repository	SSL Key repository	/var/mqm/qmgrs/myqmgr/ssl/key
Communication		
Events	Cryptographic hardware	
SSL	SSL Key repository password	
Statistics		
Online monitoring	Revocation namelist	
Statistics monitoring	Certificate validation policy	Any
Account monitoring		
Log	Cert label	
Publish/Subscribe	SSL reset count	0
	SSL FIPS required	No
	Suite B strength	None

Refresh TLS Security

At the top of the configuration page you will see a ACTION button.

Select the button and refresh the TLS settings

Get your connection information for your queue manager
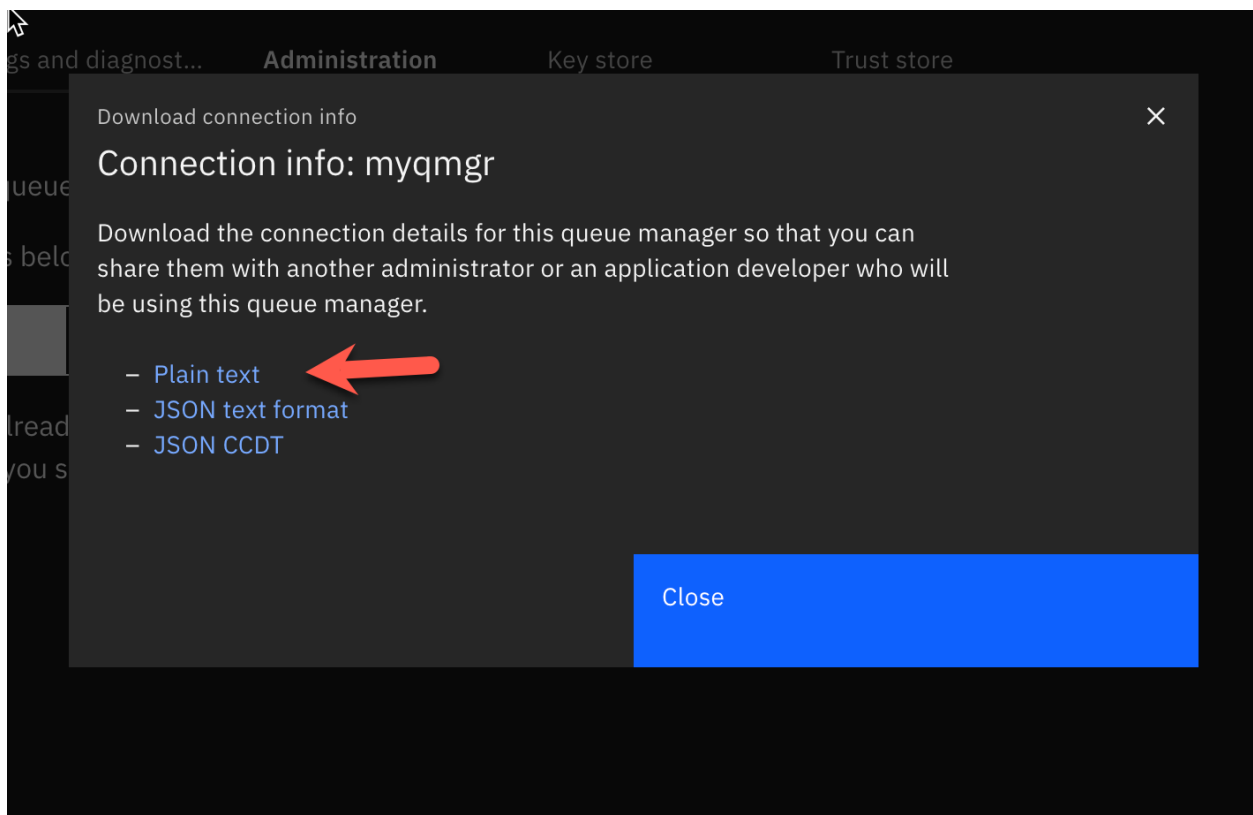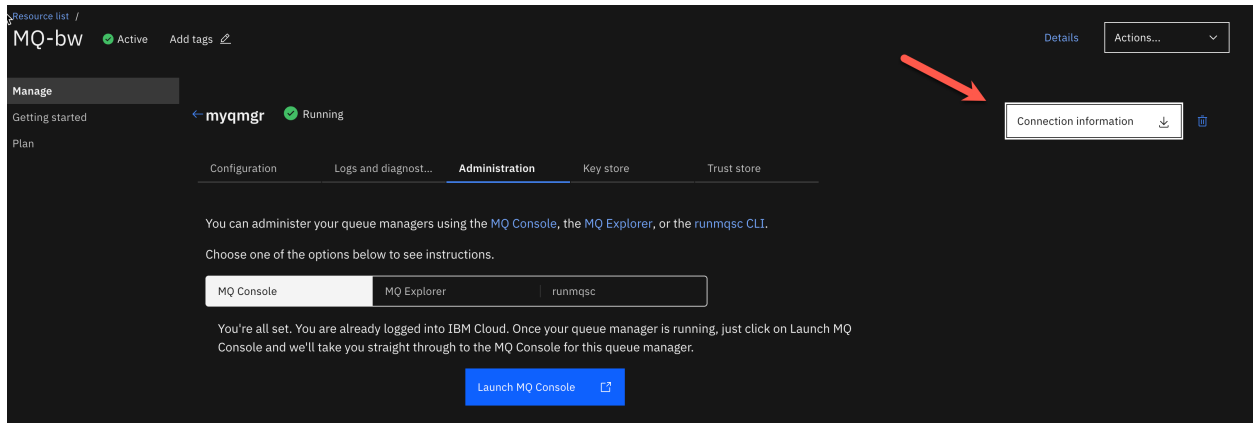
We will need this information for our application.

Running a Sample Client .NET App

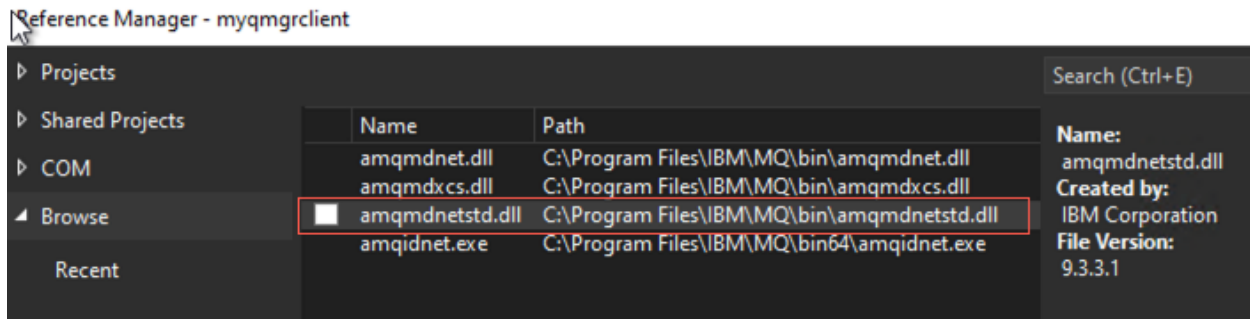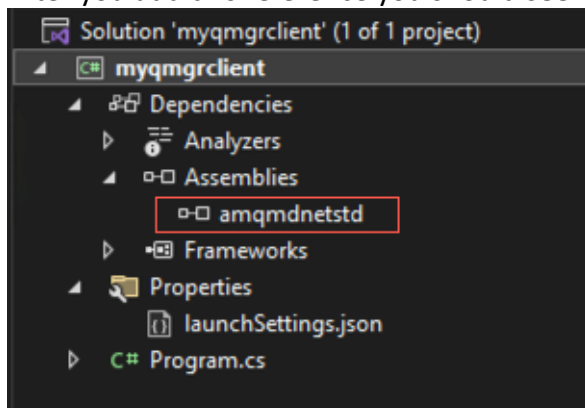I am using Visual Studio code to test my applications. You need to use the IBM MQ .Net client library to connect. You have 2 options to setup your project to use the MQ Client libraries.

Option 1.) Add a reference to the amdmdnetstd.dll from the installed windows mq client.
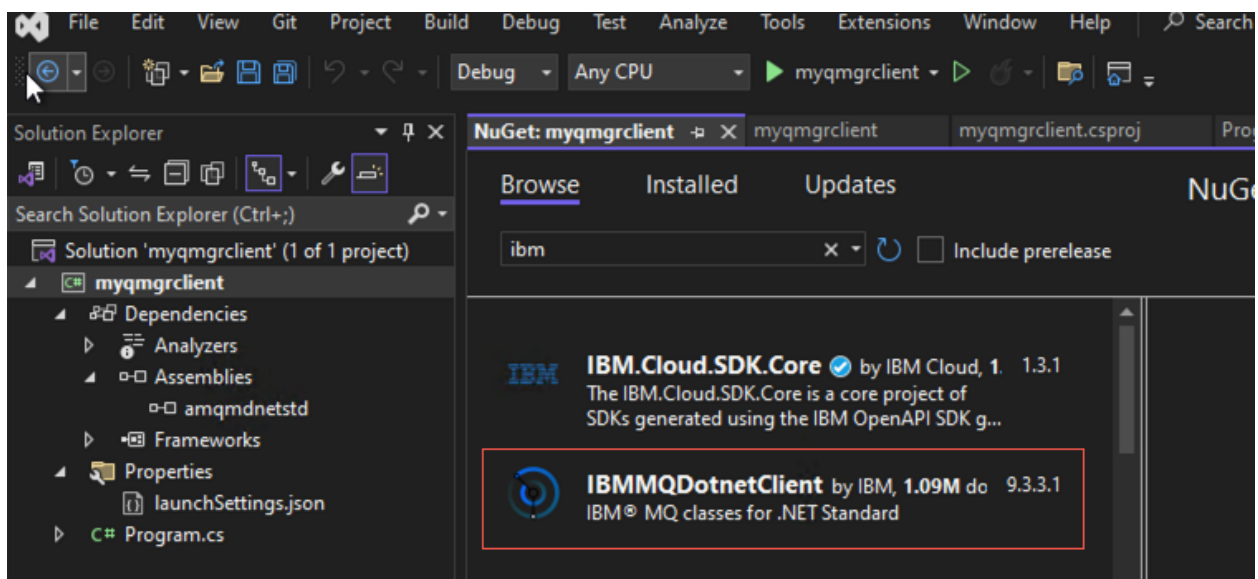
After you add this reference you should see it in your project as displayed below.

The advantage of this option is that you can run MQ Client Tracing, to help you debug.

Option 2.) You can use nuGet from Visual Studio IDE to add the IBM MQ Client to your project.

Search for packages from IBM and you will find the IBMMQDotnetClient package.

*** Below is the sample code ***

```csharp
using IBM.WMQ;
using System;
using System.Collections;
using System.Text;

namespace myqmgrclient
{
    internal class Program
    {
        static void Main(string[] args)
        {
            string strQueueManagerName = "myqmgr";
            string strChannelName = "CLOUD.APP.SVRCONN";
            string strQueueName = "DEV.QUEUE.1";
            string strServerName = "myqmgr-4e4a.qm.us-south.mq.appdomain.cloud";
            int intPort = 30762;
            string strMsg = "Hello IBM, this is a message";


            Hashtable queueProperties = new Hashtable
            {
                { MQC.HOST_NAME_PROPERTY, strServerName },
                { MQC.CHANNEL_PROPERTY, strChannelName },
                { MQC.PORT_PROPERTY, intPort },
                { MQC.CONNECT_OPTIONS_PROPERTY, MQC.MQCNO_RECONNECT },
                { MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_MANAGED },
                { MQC.SSL_CERT_STORE_PROPERTY, "*SYSTEM" },
                { MQC.SSL_CIPHER_SPEC_PROPERTY, "TLS_RSA_WITH_AES_128_CBC_SHA256" }
            };

            //Set Username
            MQEnvironment.UserId = "davetest";

            //Set Passowrd
            MQEnvironment.Password = "F-5QDP8lO_cI0j7521wEchXxzd2Yv7DeA_gPtttqa28ASBV";

            //Define a Queue Manager
            try
            {
                Console.WriteLine(Environment.UserName.ToLower());
                Console.WriteLine(MQEnvironment.CertificateLabel);
                MQQueueManager myQM =
                    new MQQueueManager(strQueueManagerName, queueProperties);

                // creating a message object
                MQMessage queueMessage = new MQMessage();
                queueMessage.WriteString("test message");

                //Define a Queue
```

```
        var queue = myQM.AccessQueue
        (strQueueName, MQC.MQOO_OUTPUT + MQC.MQOO_FAIL_IF_QUIESCING);
        MQPutMessageOptions queuePutMessageOptions = new MQPutMessageOptions();
        queue.Put(queueMessage, queuePutMessageOptions);
        queue.Close();
        Console.WriteLine("Success");
      }
      catch (Exception ex)
      {
        Console.WriteLine(ex);
        Console.WriteLine(ex.Message);
      }
      Console.ReadLine();
    }
  }
}
```

Note: There are more samples provided with the MQ Client.

Location: C:\Program Files\IBM\MQ\tools\dotnet\samples\

Debugging – Connectivity Issues..

On the Client Application

I would highly recommend you turn on MQ Tracing for your windows client.

IMPORTANT: This requires you to reference the IBM MQ Client DLL from the installed MQ Client on your local windows machine.

You will need to set 3 environment variables for your application.

Select the debug menu and then select debug properties to show this window below. Set the environment variables.

NOTE: *You need to close down Visual Studio and reopen it to get the variables to be picked up by the editor.*
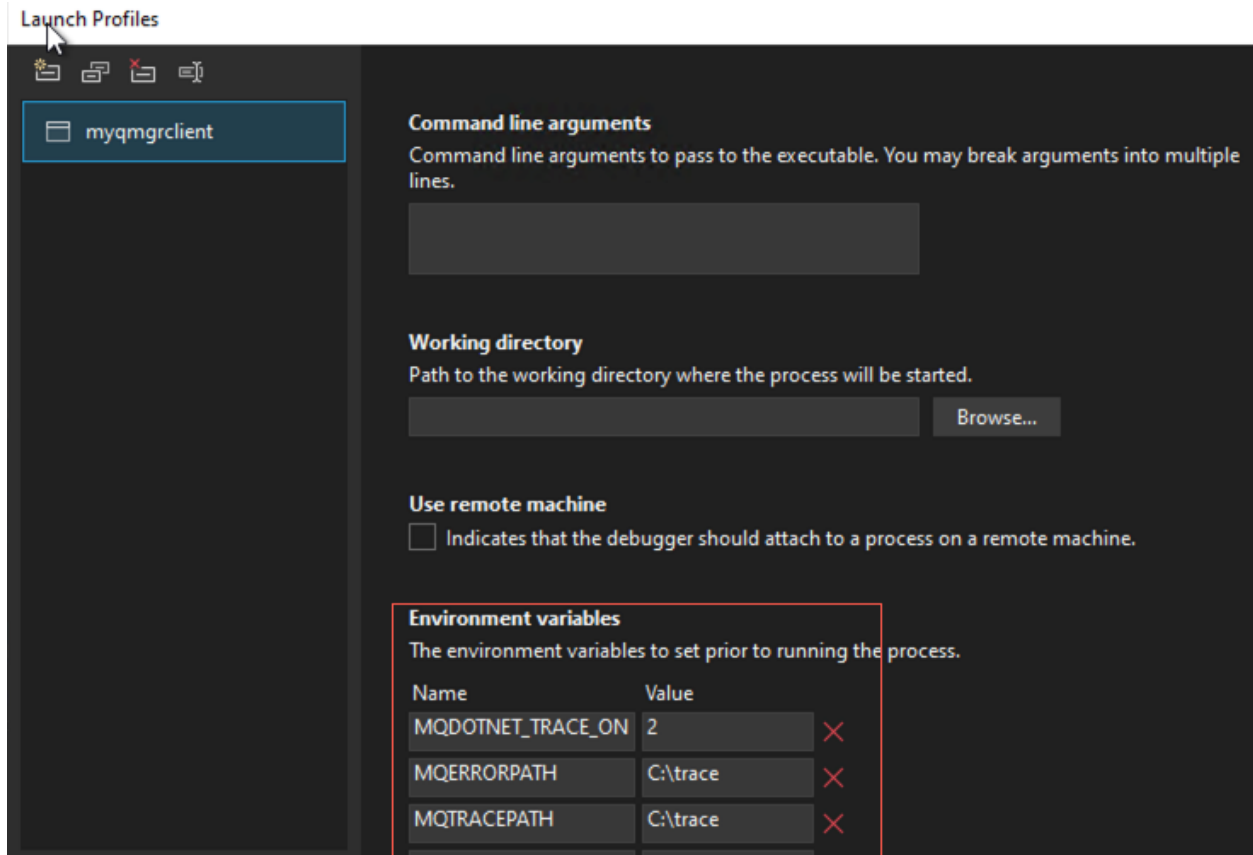
On the MQ Server.

You can see any connection errors in the logs on the server. Below is a screen shot of the logs.

Links of interest

- [Connecting to IBM MQ with SSL Article](#)
- [How to import intermediate and root certificates via MMC](#)
- [Troubleshooting MQ SSL](#)
- [Sample code repository](#)
- [(Download) – Sample instructions for SSL](#)
- [Python SSL Unmanaged .Net Framework Sample How To](#)
- [MQ Documentation – Configure TLS for Managed .NET App](#)
- [Cyper Spec for MQ Information](#)
- [Key Repository's for Managed .Net Applications](#)
- [Running .Net Core application on Linux](#)
- [How to perform common IBM Management Cert Tasks](#)
- [IBM MQ in Kubernetes – Enabling TLS with signed CA](#)
- [Generating PK12 Certs](#)
- [Creating self signed Certs for use with IBM MQ on Cloud](#)
- [Stack Overflow – 2393 Error - Resolution](#)
- [Stack Overflow – MQ TLS Error Resolution with Tracing](#)

- [How to create a PEM file](#)
- [Documentation – Understanding the Certificate Label Requirements](#)
- [How to create a pfx file from cert and private key](#)
-