

Custom MQ Images on RHOS

First section covers RHOS 3.11 with no TLS connectivity
required

Second section covers RHOS 3.11 and RHOS 4.2
Where 4.2 requires TLS to connect clients from outside
of the RHOS Cluster

Contents

Purpose of this document	5
Part 1 - Custom MQ Image on RHOS 3.1.1 with ICP4i (no TLS)	6
Load Official ICP4i MQAdvanced Server image to local docker repos.....	7
Docker File and MQSC script for custom image	7
Docker file – run an mqsc script	8
IVT.mqsc – enable clients to connect	8
Docker build – Create image FROM ibm-mqadvanced-server	9
Perform Local testing on docker workstation.....	9
Start the container – docker run.....	9
Connect MQ Explorer.....	11
Access via RUNMQSC client	12
manual	12
scripted	12
Push the new image to Dockerhub.....	14
Create a new Image Stream on RHOS.....	15
Pulling repository to use in helm charts	16
ICP4i MQ new instance using custom image	17
Add new ICP4i instance of MQ	17
Configure Helm chart.....	18
Values for the charts	18
Verifying the Helm release via ICP foundation	22
Starting the MQ Web Console	25
Add Widget - Channels.....	26
Connecting MQ Explorer.....	27
Using Runmqsc in client mode.....	28
Part 2 - Custom MQ Image on RHOS 3.11 and 4.2 with ICP4i - TLS	30
Documentation from the Helm chart with annotations.....	30
Configuring MQ objects	30
JSON log output	32
Supplying certificates to be used for TLS	32
Supplying certificates which contain the public and private keys	32
Creating TLS key and certs set – In the local “Admin” environment.....	35
Points of note.....	35

Summary of steps	35
Local Environment	36
Runmqsc TLSPRQM1 to create SVRCONN to use	36
Step 1: Client: Create SSL client key database	37
Step 2: Client: Create certificate	38
Step 3: Client: Extract the public SSL client certificate and copy it to the SSL server side	39
Step 4: Server: Create SSL server key database Step 5: Server: Create certificate	40
Step 5: Server: Create certificate	41
Step 6: Server: Add the SSL client certificate to the Queue Manager's key database.	44
Step 7: Server: Extract the public TLS server certificate and copy it to the TLS client side	45
Step 8: Client: Add the SSL client certificate to the Client's key database.	46
Step 9: Server: Run MQSC commands for TLS server side queue manager	47
Step 10: Verifying the TLS set up with MQCERTCK.....	49
File read/write attributes on the .KDB.....	49
Using MQCERTCK	50
Step 11: Client: use runmqsc in client mode to connect to the TLSPRQM1 and administer it	52
Using MQSERVER for the client channel won't work	52
Using MQCCDT for the client channel	53
Set up the CCDT	54
SERVERside TLS files – end of exercise	57
CLIENTside TLS files – end of exercise.....	57
Step 12: Client: connect MQ Explore to queue manager via “Add remote Queue Manager”	58
Creating the custom MQ Image Layer for ICP4i on RHOS 4.2	60
TLS enabled custom image layer – Build Files	60
Docker file – run mqsc scripts and copy .KDB file to image.....	60
TLSPRQM1.mqsc – enable TLS clients to connect	61
NOTLS.mqsc – add channel for in cluster clients to connect.....	62
Key database files and password stash files	62
Docker build – Create image FROM ibm-mqadvanced-server	62
Perform Local testing on docker workstation.....	64
Start the container – docker run.....	64
Connect MQ Explorer.....	65
Connect RUNMQSC via Non TLS client	67
Connect RUNMQSC via TLS client	67
Set up the Client Channel Definition Table	67
Optionally set the CCDT environment variables.....	68

Clear the MQSERVER environment variable so the CCDT is used	69
Set the MQSSLKEYR environment variable for the client side.....	70
Testing TLS custom MQ Image Layer on RHOS 3.11	72
Pushing the custom image to docker hub	72
Create a new Image Stream on RHOS 3.11.....	73
Pulling repository to use in helm charts on 3.11	74
Add new ICP4i instance of MQ custom TLS layer on RHOS 3.11	74
Configure Helm chart.....	76
Values for the charts	76
Verifying the Helm release via ICP foundation	80
Configure client side CCDT for TLSPRQM1 on RHOS 3.11.....	83
Using pre-configured collateral to connect	85
Custom Layer MQ Image for TLS.....	85
Deploy an ICP4i MQ instance.....	85
CCDT	85
client side TLS files	85
https://github.com/DAVEXACOM/Exploring-ICP4i-RHOS/tree/master/TLS%20-%20custom%20layer%20MQ%20Image/clientside%20TLS%20files	85
Set/Clear Environment Variables.....	85
Connecting runmqsc in client mode	85
Testing TLS custom MQ Image Layer on RHOS 4.2	87
Pushing the custom image to docker hub	87
Appendix of useful-ish info and links	90
CCDT setting up.....	94
Debugging	95
Lacking a certificate – didn't add the certlab to CLNTCONN in CCDT.....	95
Not authorized – used mqm instead of MUSR_MQADMIN in the chl auth rec	97
No CipherSpec – used MQSERVER instead of CCDT.	98
Debugging in local container	99

Purpose of this document

ICP4i on RHOS 4.2 requires that “off cluster” clients connect via TLS to queue managers deployed in the cluster.

In scenarios where administrators wish to connect to MQ using existing tools or processes it is likely that they will do so against an MQ Image where a custom layer has been added over the top of the IBM MQ base image. Such a layer would simply enable client connectivity for “off cluster” tools.

When using RHOS 3.11 this is fairly straight forward. When using RHOS 4.2 we need TLS connection and therefore we need to consider that the administrators system (laptop etc) needs to be set up with keys and certs that will be interchanged with the queue managers on cluster.

This document assumes that “you” are that administrator and your system (laptop) has never used TLS for MQ connectivity before.

PART 1 of the document – this is simply showing how to create a custom layer on the MQ Image and make it available in an RHOS 3.11 cluster with ICP4i and connect administration tools over a specific SVRCONN channel





PART 2 of the document – this builds to connecting to queue managers on RHOS 4.2 over TLS from your laptop.

- a) Setting up TLS between a client connected tools on the lap top to a queue manager running natively on your laptop
- b) Reusing the collateral in a) above to build a custom docker image/container and have client connect tools on the lap top connect over TLS to a queue manager in the container running on the laptop.
- c) Push the custom image to dockerhub, use an image stream on RHOS 3.11 to pull the image onto RHOS 3.11 cluster. ICP4i (Helm) deploy of the custom image. Connect tools on the lap top to the queue manager on ICP4i on RHOS 3.11 via TLS
- d) **To be Done** Push the custom image to dockerhub, use an image stream on RHOS 4.2 to pull the image onto RHOS 4.2 cluster. ICP4i (Helm) deploy of the custom image. Connect tools on the lap top to the queue manager on ICP4i on RHOS 4.2 via TLS

Part 1 - Custom MQ Image on RHOS 3.1.1 with ICP4i (no TLS)

Load Official ICP4i MQAdvanced Server image to local docker repos

Windows (C:) > SHARE > ProductsTechnology > MQv9-9.1 > ICP4iMQAdv-image > images

<input type="checkbox"/> Name	Date modified	Type
 ibm-mqadvanced-server-integration_9.1.3.0-r4-amd64.tar	9/11/2019 1:14 AM	GZ File
 ibm-mq-oidc-registration_2.2.0-amd64.tar	9/11/2019 1:14 AM	GZ File
 icp4i-od-agent_1.0.0-amd64.tar	9/11/2019 1:14 AM	GZ File
 icp4i-od-collector_1.0.0-amd64.tar	9/11/2019 1:14 AM	GZ File

docker load < ibm-mqadvanced-server-integration_9.1.3.0-r4-amd64.tar.gz

```
Directory of C:\SHARE\ProductsTechnology\MQv9-9.1\ICP4iMQAdv-image\images



13/12/2019  07:02 PM  <DIR>          .
13/12/2019  07:02 PM  <DIR>          ..
09/11/2019  01:14 AM      161,053,184  ibm-mq-oidc-registration_2.2.0-amd64.tar.gz
09/11/2019  01:14 AM      942,485,504  ibm-mqadvanced-server-integration_9.1.3.0-r4-amd64.tar.gz
09/11/2019  01:14 AM      310,161,408  icp4i-od-agent_1.0.0-amd64.tar.gz
09/11/2019  01:14 AM      321,895,424  icp4i-od-collector_1.0.0-amd64.tar.gz
               4 File(s)  1,735,595,520 bytes
               2 Dir(s)  217,878,286,336 bytes free

C:\SHARE\ProductsTechnology\MQv9-9.1\ICP4iMQAdv-image\images>docker load < ibm-mqadvanced-server-integration_9.1.3.0-r4-amd64.tar.gz
c2d28e8b3d42a: Loading layer [=====] 7.68kB/7.68kB
84eaf14cef7b: Loading layer [=====] 6.144kB/6.144kB
dc025946c48b: Loading layer [=====] 806MB/806MB
084a5de0f8d3: Loading layer [=====] 2.56kB/2.56kB
a0cf9052d6e2: Loading layer [=====] 12.39MB/12.39MB
8c20517487b4: Loading layer [=====] 5.876MB/5.876MB
0199fa08053a: Loading layer [=====] 10.24kB/10.24kB
7f6264a97cf1: Loading layer [=====] 11.78kB/11.78kB
1355ce09db14: Loading layer [=====] 3.584kB/3.584kB
f9d44c67deb4: Loading layer [=====] 18.28MB/18.28MB
c4b465bf4f6a: Loading layer [=====] 7.68kB/7.68kB
79163b2a33c5: Loading layer [=====] 2.146MB/2.146MB
2c903f0b7155: Loading layer [=====] 3.072kB/3.072kB
87ba76dd2431: Loading layer [=====] 3.584kB/3.584kB
1c321fdb962: Loading layer [=====] 7.168kB/7.168kB
f561b484f28b: Loading layer [=====] 2.152MB/2.152MB
2b68e45b087e: Loading layer [=====] 7.168kB/7.168kB
a10f6d83153f: Loading layer [=====] 21.5kB/21.5kB
1cf819e9f0e: Loading layer [=====] 3.237MB/3.237MB
364112d1df0e: Loading layer [=====] 261.6kB/261.6kB
9f4789d1e0ed: Loading layer [=====] 3.072kB/3.072kB
07cdf8094239: Loading layer [=====] 3.497MB/3.497MB
6eb6a773bf24: Loading layer [=====] 3.738MB/3.738MB
ac7598f58291: Loading layer [=====] 3.738MB/3.738MB
Loaded image: ibm-mqadvanced-server-integration:9.1.3.0-r4-amd64

C:\SHARE\ProductsTechnology\MQv9-9.1\ICP4iMQAdv-image\images>
```

Docker File and MQSC script for custom image

(C:) > Users > DAVIDARNOLD > myDocker > ibm-mqadvanced-server-ivt

<input type="checkbox"/> Name	Date modified	Type
 dockerfile	12/12/2019 3:10 PM	File
 ivt.mqsc	12/12/2019 4:55 PM	MQSC File

Docker file – run an mqsc script

FROM ibm-mqadvanced-server-integration:9.1.3.0-r4-amd64

USER mqm

COPY ivt.mqsc /etc/mqm/

IVT.mqsc – enable clients to connect

```
DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCONN)
```

```
SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)
```

```
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE)  
ADOPTCTX(YES)
```

```
SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')
```

```
REFRESH SECURITY TYPE(CONNAUTH)
```

OR – if this is based off a non ICP4i mq image that used admin/passw0rd

* Connection authentication

```
DEFINE AUTHINFO('ORG.AUTHINFO') AUTHTYPE(IDPWOS) CHCKCLNT(REQDADM)  
CHCKLOCL(OPTIONAL) ADOPTCTX(YES) REPLACE
```

```
ALTER QMGR CONNAUTH('ORG.AUTHINFO')
```

```
REFRESH SECURITY(*) TYPE(CONNAUTH)
```

* Channels (Application + Admin)

```
DEFINE CHANNEL('IVT.SVRCONN') CHLTYPE(SVRCONN) REPLACE
```

```
DEFINE CHANNEL('ORG.APP.SVRCONN') CHLTYPE(SVRCONN) MCAUSER('app') REPLACE
```

* Channel authentication rules

```
SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS) DESCR('Back-stop rule -  
Blocks everyone') ACTION(REPLACE)
```

```
SET CHLAUTH('ORG.APP.SVRCONN') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(CHANNEL)  
CHCKCLNT(REQUIRED) DESCR('Allows connection via APP channel') ACTION(REPLACE)
```

```
SET CHLAUTH('IVT.SVRCONN') TYPE(BLOCKUSER) USERLIST('nobody') DESCR('Allows admins on  
ADMIN channel') ACTION(REPLACE)
```

```
SET CHLAUTH('IVT.SVRCONN') TYPE(USERMAP) CLNTUSER('admin') USERSRC(CHANNEL)  
DESCR('Allows admin user to connect via ADMIN channel') ACTION(REPLACE)
```

* Authority records

```
SET AUTHREC GROUP('mqclient') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
```



```
SET AUTHREC PROFILE('ORG.**') GROUP('mqclient') OBJTYPE(QUEUE)
AUTHADD(BROWSE,GET,INQ,PUT)
```

```
SET AUTHREC PROFILE('ORG.**') GROUP('mqclient') OBJTYPE(TOPIC) AUTHADD(PUB,SUB)
```

Docker build – Create image FROM ibm-mqadvanced-server

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ace11002mqc91intms1	1.0	ec39bb5df84f	2 days ago	1.59GB
ibmcom/mq	latest	268baf40303f	7 days ago	927MB
ibmcom/ace-mq	latest	0f5a883d8a95	4 weeks ago	2.51GB
ibmcom/ace-mqclient	latest	d71cee6dfd5f	4 weeks ago	1.94GB
ibmcom/ace	latest	d33c5a966d7b	4 weeks ago	1.63GB
ibm-mqadvanced-server-integration	9.1.3.0-r4-amd64	ee41039b9552	5 weeks ago	932MB

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt> docker build -t davexacom/ibm-  
mqadvanced-server-ivt:9.1.3.0-r4-amd64 .
```

```
C:\Users\DAVIDARNOLD\myDocker\mqsrvmqsc>docker build -t davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64 .  
Sending build context to Docker daemon 4.096kB  
Step 1/3 : FROM ibmcom/mq  
----> 268baf40303f  
Step 2/3 : USER mqm  
----> Using cache  
----> f976d4712a79  
Step 3/3 : COPY ivt.mqsc /etc/mqm/  
----> Using cache  
----> 0c686c880b92  
Successfully built 0c686c880b92  
Successfully tagged davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64
```

Perform Local testing on docker workstation

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
davexacom/ibm-mqadvanced-server-ivt	9.1.3.0-r4-amd64	0c686c880b92	3 days ago	927MB

```
docker run -e LICENSE=accept -e MQ_QMGR_NAME=QM1 -P davexacom/ibm-mqadvanced-server-  
ivt:9.1.3.0-r4-amd64
```

Start the container – docker run

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker run -e LICENSE=accept -e  
MQ_QMGR_NAME=QM1 -P davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64
```

```
2019-12-13T08:29:20.120Z CPU architecture: amd64
```

2019-12-13T08:29:20.121Z Linux kernel version: 4.9.184-linuxkit

2019-12-13T08:29:20.121Z Container runtime: docker

2019-12-13T08:29:20.121Z Base image: Red Hat Enterprise Linux Server 7.7 (Maipo)

Break

Starting MQSC for queue manager QM1.

1 : DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCONN)

AMQ8014I: IBM MQ channel created.

2 : SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)

AMQ8877I: IBM MQ channel authentication record set.

3 : ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS)
CHKCCLNT(NONE) ADOPTCTX(YES)

AMQ8567I: IBM MQ authentication information changed.

4 : SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')

AMQ8877I: IBM MQ channel authentication record set.

5 : REFRESH SECURITY TYPE(CONNAUTH)

AMQ8560I: IBM MQ security cache refreshed.

5 MQSC commands read.

No commands have a syntax error.

All valid MQSC commands were processed.2019-12-13T08:29:22.125Z

Metrics are disabled

2019-12-13T09:09:17.684Z AMQ8024I: IBM MQ channel initiator started.

2019-12-13T09:09:17.706Z AMQ5806I: Queued Publish/Subscribe Daemon started for queue manager QM1.

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS		NAMES		

b5e2c8034d8c davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64
"runmqintegrationser..." 26 seconds ago Up 25 seconds 0.0.0.0:32770->1414/tcp,
0.0.0.0:32769->9157/tcp, 0.0.0.0:32768->9443/tcp relaxed_lamport

[Connect MQ Explorer](#)

IBM MQ

Queue Managers

Add Queue Manager

Select the queue manager and connection method

Identify the queue manager to add and choose the connection method to use

Queue manager name:

Queue manager name:

Connection details

Host name or IP address:

Port number:

Server-connection channel:

Queue manager name:

☒ Enable user identification

☐ User identification compatibility mode

Userid:

Password

☒ No password

MQ Explorer - Navigator
MQ Explorer - Content

IBM MQ
Queue Managers
EAIQM1 on 'icp-console.apps.ocp.cloudnativekubernetes.com(30798)'
EAIQM2 on 'icp-console.apps.ocp.cloudnativekubernetes.com(32588)'
IDCQM1 on 'icp-console.apps.ocp.cloudnativekubernetes.com(30419)'
IDCQM2 on 'icp-proxy.apps.ocp.cloudnativekubernetes.com(31971)'
QM1 on 'localhost(32805)'
Queues
Topics
Subscriptions
Channels
Client Connections
Channel Authentication Records

Channels

Filter: Standard for Channels

Channel name	Channel type	Overall channel status
IVT.SVRCONN	Server-connection	Running

Access via RUNMQSC client

SET MQSERVER=IVT.SVRCONN/TCP/localhost(32770)

manual

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>runmqsc -c QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager QM1.

def ql(DA.Q)
  1 : def ql(DA.Q)
AMQ8006I: IBM MQ queue created.
end
  2 : end
2 command responses received.
```

scripted

1q-npwd.mqsc - Notepad

File Edit Format View Help

```
define ql(DA) replace
```

```
C:\temp>SET MQSERVER=IVT.SVRCONN/TCP/localhost(32805)
```

```
C:\temp>runmqsc -c QM1 < 1q-npwd.mqsc
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager QM1.
```

```
1 : define ql(DA) replace
AMQ8006I: IBM MQ queue created.
2 command responses received.
```

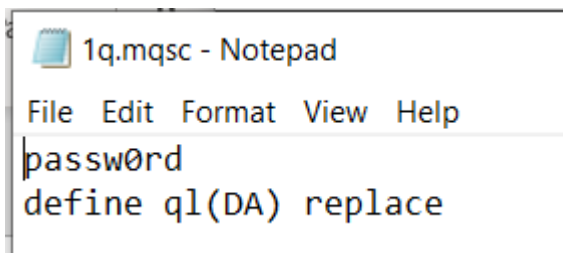
Note if you are using a queue manager where the username is password protected

Manual running of mqsc

```
C:\temp>runmqsc -c -u admin QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Enter password:
*****
Starting MQSC for queue manager QM1.

end
1 : end
0 command responses received.
```

If -u is on the command and we are piping in, runmqsc assumes the first line is the password.



```
C:\temp>runmqsc -c -u admin QM1 < 1q.mqsc
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Enter password:
Starting MQSC for queue manager QM1.

1 : define ql(DA) replace
AMQ8006I: IBM MQ queue created.
2 command responses received.
```

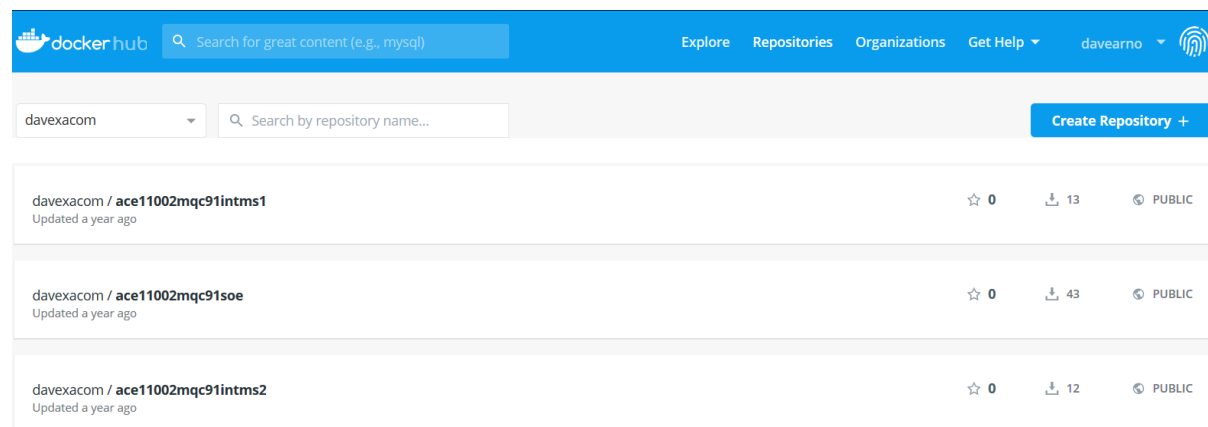
Push the new image to Dockerhub

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
davexacom/ibm-mqadvanced-server-ivt	9.1.3.0-r4-amd64	2237fea38967	17 minutes ago	932MB

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
davexacom/ibm-mqadvanced-server-ivt  9.1.3.0-r4-amd64  e269ad126613     5 minutes ago   932MB
```

<https://hub.docker.com/repositories>



docker push davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker push davexacom/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64
```

The push refers to repository [docker.io/davexacom/ibm-mqadvanced-server-ivt]

```
0d8459f27ecc: Pushing [=====>] 3.072kB
```

```
ac7598f58291: Pushing [=====>] 1.378MB/3.735MB
```

```
6eb6a773bf24: Pushing [=====>] 1.378MB/3.735MB
```

```
07cdf8094239: Pushing [=====>] 561.2kB/3.49MB
```

```
9f4789d1e0ed: Pushing [=====>] 3.072kB
```

```
364112d1df0e: Waiting
```

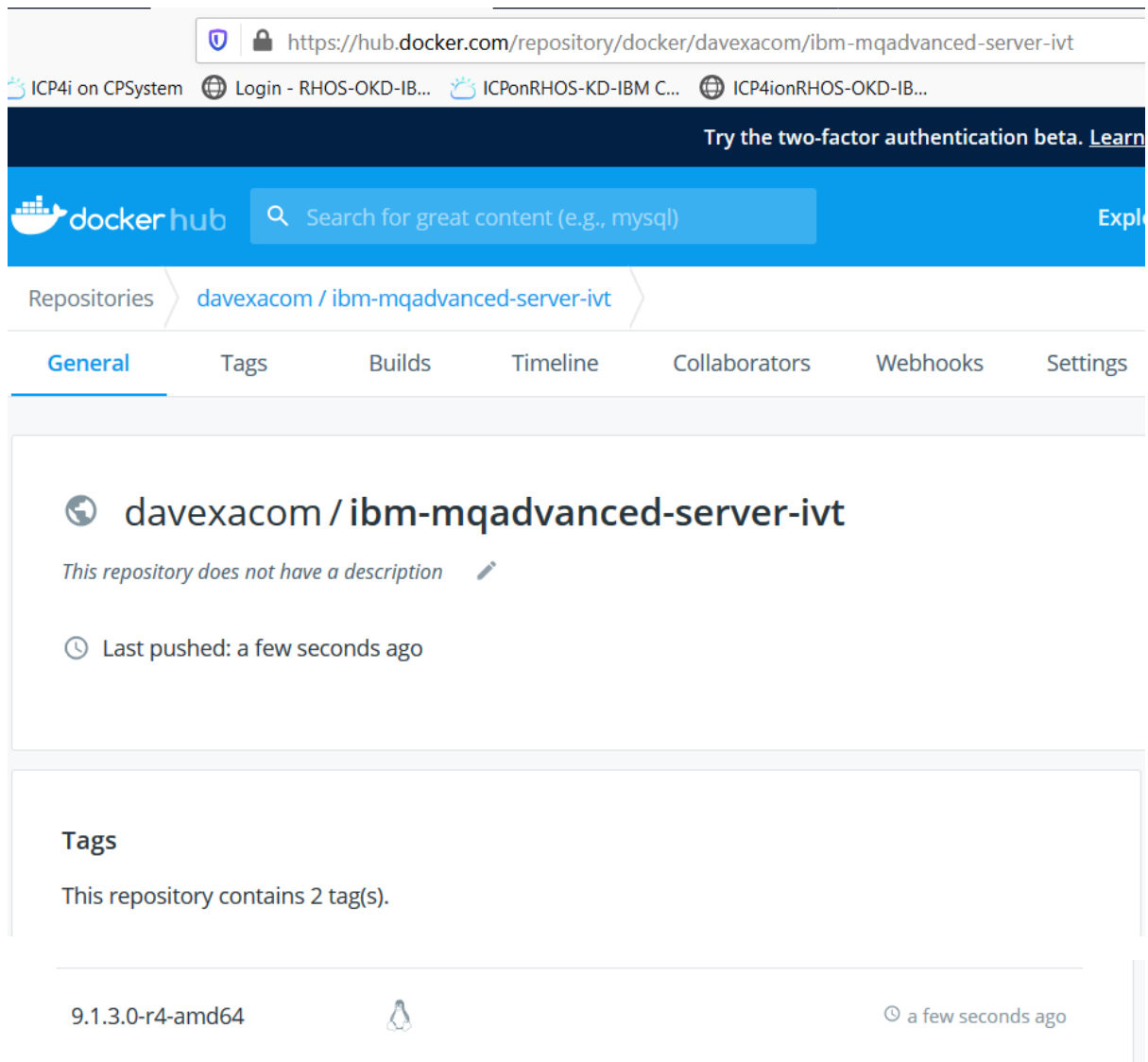
```
cd28e8b3d42a: Pushed
```

```
2705f79af6db: Layer already exists
```

```
ce459cc53899: Layer already exists 9.1.3.0-r4-
```

```
amd64: digest: sha256:040e8e0aca3307369b10cce8411f97e147a9a92bb7514619a16c2a78093c085f
size: 5963
```

<https://hub.docker.com/repository/docker/davexacom/ibm-mqadvanced-server-ivt>



The screenshot shows the Docker Hub interface for the repository `davexacom/ibm-mqadvanced-server-ivt`. The page includes a navigation bar with the Docker Hub logo and a search bar. Below the navigation bar, the repository name is displayed, followed by a description stating "This repository does not have a description". A clock icon indicates the repository was "Last pushed: a few seconds ago". The "Tags" section shows two tags: `9.1.3.0-r4-amd64` and `9.1.3.0-r1-amd64`, both pushed "a few seconds ago".


<https://hub.docker.com/repository/docker/davexacom/ibm-mqadvanced-server-ivt>


Try the two-factor authentication beta. [Learn](#)


docker hub Search for great content (e.g., mysql) Expl

Repositories > davexacom / ibm-mqadvanced-server-ivt >

General Tags Builds Timeline Collaborators Webhooks Settings


 **davexacom / ibm-mqadvanced-server-ivt**

This repository does not have a description 

 Last pushed: a few seconds ago

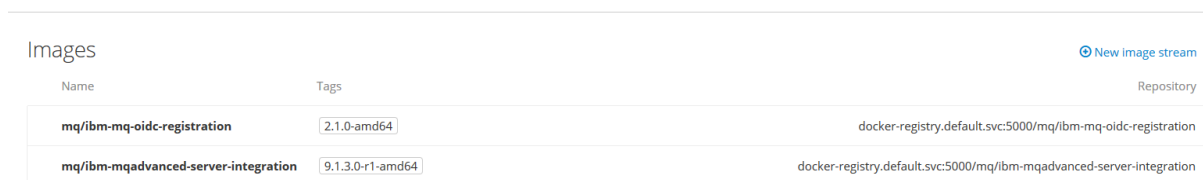
Tags

This repository contains 2 tag(s).

9.1.3.0-r4-amd64  a few seconds ago

Create a new Image Stream on RHOS

<https://registry-console-default.apps.ocp.cloudnativekube.com/registry#/images/mq>



The screenshot shows the RHOS registry console interface. It displays a table of image streams with columns for Name, Tags, and Repository. There are two image streams listed: `mq/ibm-mq-oidc-registration` and `mq/ibm-mqadvanced-server-integration`. A link to "New image stream" is visible in the top right corner.

Name	Tags	Repository
mq/ibm-mq-oidc-registration	2.1.0-amd64	docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration
mq/ibm-mqadvanced-server-integration	9.1.3.0-r1-amd64	docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-integration

[+ New image stream](#)

Repository

Docker.io/davexacom/ibm-mqadvanced-server-ivt

Create new image stream / or change image stream

Change image stream

Name

Project

Populate

Pull from

Tags

☐ Remote registry is insecure

mq/ibm-mqadvanced-server-ivt

1.0-amd64 9.1.3.0-r4-amd64

docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt

Image stream

Tags

Access Policy Images may only be pulled by specific users or groups

Pulling repository... docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt

Image count 2

To push an image to this image stream:

```
$ sudo docker tag myimage docker-registry-default.apps.ocp.cloudnativekubernetes.com/mq/ibm-mqadvanced-server-ivt:tag
$ sudo docker push docker-registry-default.apps.ocp.cloudnativekubernetes.com/mq/ibm-mqadvanced-server-ivt:tag
```

Pulling repository to use in helm charts

docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt

Add new ICP4i instance of MQ

MQ

You'll be deploying [IBM MQ](#) to provide this capability. Please work with your IBM Cloud Private cluster administrator to ensure the following dependencies are satisfied:


- Close

Continue

Configure Helm chart ibm-mqadvanced-server-integration-prod V 4.1.0

Overview

Configuration

 IBM Certified Container


IBM MQ queue manager for Cloud Pak for Integration

ibm-entitled-charts

[Licenses](#) | [Release Notes](#) | [Qualification](#)

IBM Certified Container

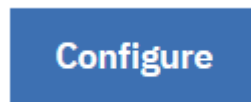
4.1.0

 IBM MQ Advanced

Introduction

This chart deploys a single IBM® MQ version 9.1.3 middleware that simplifies and accelerates the integration of multiple platforms. It uses message queues to facilitate messaging solution for cloud, mobile, Internet of Things.

Hit configure



Values for the charts

Cluster Hostname (for deployment)

icp-proxy.apps.ocp.cloudnativekube.com

MQ server image:

docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt

MQ Server image Tag:

9.1.3.0-r4

MQ Server OIDC registration image

docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration

MQ Server OIDC registration image tag:

2.1.0

Managed NFS name (from oc command):

managed-nfs-storage

TLS Secret

ibm-mq-tls-secret

IBM MQ queue manager for Cloud Pak for Integration. Edit these parameters for configuration.

Helm release name *

idcqm1p

Target namespace *

mq

Target Cluster *

local-cluster

License * 



I have read and agreed to the [License agreement](#)

icp-proxy.apps.ocp.cloudnativekube.com



Quick start

Required and recommended parameters to view and edit.

TLS

Configuration settings for TLS

Cluster hostname * 

icp-proxy.apps.ocp.cloudnativekube.com|

MQ server image:

[docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt](#)

MQ Server image Tag:

[9.1.3.0-r4](#)

Results in the following image and tag being used

[docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ict:9.1.3.0-r4-amd64](#)

Image	
Configuration settings for the container image	
Image repository *	Image tag *
docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt	9.1.3.0-r4
Image pull secret	Image pull policy (for all images) *
Enter value	IfNotPresent

MQ Server OIDC registration image

docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration

MQ Server OIDC registration image tag:

2.1.0

Single sign-on	
Configuration settings for single sign-on	
Registration image repository *	Registration image tag *
docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration	2.1.0

TLS Secret

ibm-mq-tls-secret

TLS	
Configuration settings for TLS	
<input checked="" type="checkbox"/> Generate Certificate	
Cluster hostname *	Secret name ?
icp-proxy.apps.ocp.cloudnativekubernetes.com	Ibm-mq-tls-secret

Managed NFS name (from oc command):

managed-nfs-storage

Persistence

Configuration settings for Persistent Volumes

☒ **Enable persistence ***

☒ **Use dynamic provisioning ***

Data PVC

Configuration settings for the main Persistent Volume Claim

Name *

data

Storage Class name 


managed-nfs-storage

Size *

2Gi

Queue manager

Configuration settings for the Queue Manager

Queue manager name 

IDCQM1

Metrics

Configuration settings for Prometheus metrics

☐ **Enable metrics *** 

Readiness probe

Configuration settings for the MQ readiness probe, which checks when the MQ listener is running

Initial delay (seconds)

30



Install

Use the little X to retain your populated chart for re-use if things go wrong.

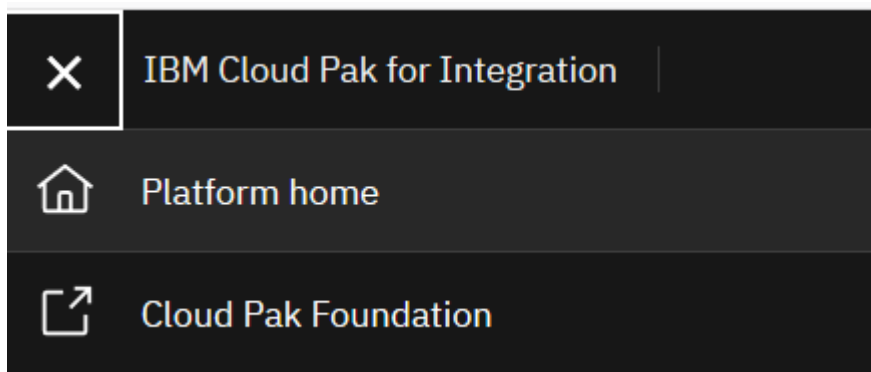


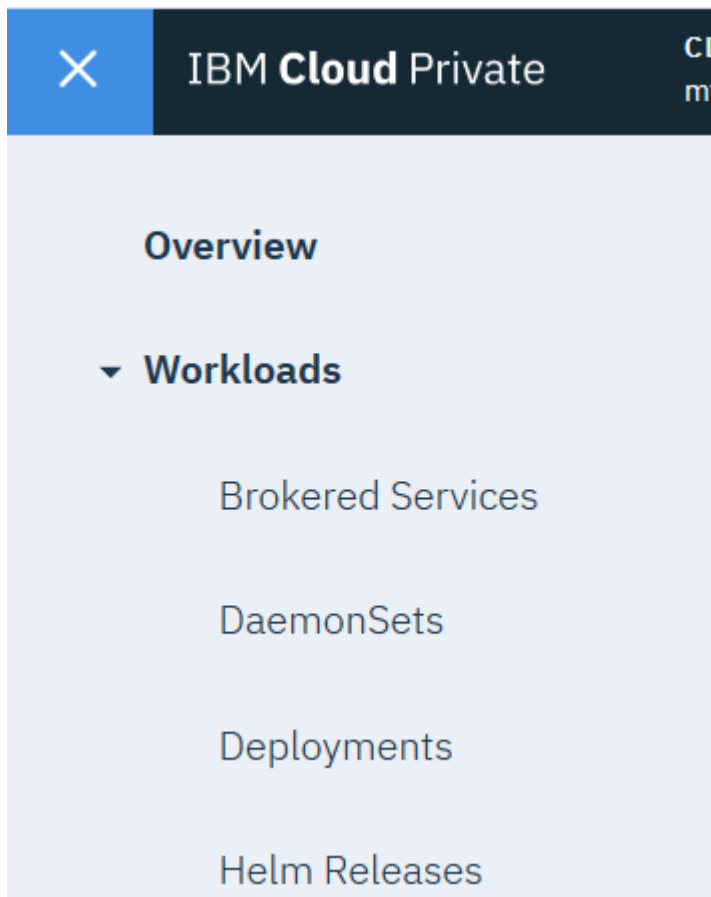
Installation started. For progress view your Helm releases.

View Helm Releases

Return to [Catalog](#)

Verifying the Helm release via ICP foundation





Helm Releases

<div><div></div><div>You are currently viewing only the helm releases of this cluster.</div></div>			
<div><div></div><div>idcq </div><div></div></div>			
Name ▲	Namespace	Status	Chart name
idcqm1p	mq	<div></div> Deployed	ibm-mqadvanced-server-integration-prod

g Started

ICP4i on CPSystem

Login - RHOS-OKD-IB...

ICPonRHOS-KD-IBM C...

ICP4ionRHOS-OKD-IB...

https://icp-console.apps.ocp.cloudnativekubernetes.com/catalog/instancedetails/mq/idcqm1p

...🏠

IBM Cloud Private

CLUSTER
mycluster

Create resource

Job

Name	COMPLETIONS	DURATION	Age
idcqm1p-ibm-mq-registration	1/1	7s	119s

Pod

Name	READY	Status	RESTARTS	Age
idcqm1p-ibm-mq-0	1/1	Running	0	118s
idcqm1p-ibm-mq-registration-fknmq	0/1	Completed	0	119s

Role

Name	Age
idcqm1p-ibm-mq	2m

https://icp-console.apps.ocp.cloudnativek8s.com/catalog/instancedetails/mq/idcqm1p

Started

ICP4i on CPSystem

Login - RHOS-OKD-IB...

ICPonRHOS-KD-IBM C...

ICP4ionRHOS-OKD-IB...

IBM Cloud Private

CLUSTER
mycluster

Create resource

Service

Name	TYPE	Cluster IP	External IP	Port(s)
idcqm1p-ibm-mq	NodePort	172.30.223.140	<none>	9443:31495/TCP,1414:32182/TCP

ServiceAccount

Name	SECRETS	Age
idcqm1p-ibm-mq	2	2m

StatefulSet

Name	Desired	Current	Age
idcqm1p-ibm-mq	1	1	119s

[StatefulSets](#) / [idcqm1p-ibm-mq](#)

idcqm1p-ibm-mq

Overview

[Events](#)

Search Events				
Type	Source	Count	Reason	Message
Normal	statefulset-controller	1	SuccessfulCreate	create Pod idcqm1p-ibm-mq-0 in StatefulSet idcqm1p-ibm-mq successful
items per page 20 1-1 of 1 items				

Pods					
<div><div></div> Search Pods</div>					
Name	Namespace	Status	Host IP	Pod IP	Ready
idcqm1p-ibm-mq-0	mq	Running	135.90.82.92	10.129.1.228	1/1
items per page 20 ▾ 1-1 of 1 items					

StatefulSets / idcqm1p-ibm-mq / idcqm1p-ibm-mq-0

idcqm1p-ibm-mq-0

Overview Containers Events

<div><div></div> Search Events</div>					
Type	Source	Count	Reason	Message	
Normal	kubelet icp4inodeme3.ocp.cloudnativekube.com	1	Started	Started container	
Normal	kubelet icp4inodeme3.ocp.cloudnativekube.com	1	Pulled	Container image "docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-ivt:9.1.3.0-r4-amd64" already present on machine	
Normal	kubelet icp4inodeme3.ocp.cloudnativekube.com	1	Created	Created container	

Starting the MQ Web Console

IBM Cloud Private

CLUSTER
mycluster

Create resource

Catalog

Helm Releases

You are currently viewing only the helm releases of this cluster.

idc

Name ↕	Namespace	Status	Chart name	Current version	Available version	Updated	
idcab1	mq	<div></div> Deployed	ibm-mqadvanced-server-integration-prod	4.0.0	5.0.0 ▲	December 13, 2019 11:17am	Launch ▼
idcqm1p	mq	<div></div> Deployed	ibm-mqadvanced-server-integration-prod	4.1.0	5.0.0 ▲	December 16, 2019 02:51pm	Launch ▲
oidcclient-watcher	kube-system	<div></div> Deployed	oidcclient-watcher	3.2.1906-rhel	Up To Date	December 4, 2019	Launch ▼ console-https



Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to icp-proxy.apps.ocp.cloudnativekube.com. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

[Learn more...](#)

Go Back (Recommended)

Advanced...

icp-proxy.apps.ocp.cloudnativekube.com:31495 uses an invalid security certificate.

The certificate is not trusted because it is self-signed.

Error code: [MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT](#)

[View Certificate](#)

Go Back (Recommended)

Accept the Risk and Continue

Add Widget - Channels

Started

ICP4j on CPSystem

Login - RHOS-OKD-IB...

ICPonRHOS-KD-IBM C...

ICP4ionRHOS-OKD-IB...

Tab 1

+

Local Queue Managers

↺ ✕

Channels on IDCQM1

Search


▲ Name	Status
IDCQM1	● Running

Search



▲ Name	Type
IVT.SVRCONN	Server-connection

Connecting MQ Explorer


Add remote queue manager


 IBM MQ Explorer (Installation1)

File Edit Window Help

 MQ Explorer - Navigator 

▼  IBM MQ

▼  Queue Managers

 Add Queue Manager

Select the queue manager and connection method


Identify the queue manager to add and choose the connection method to use

Queue manager name:

How do you want to connect to this queue manager?

☒ Connect directly

This option creates a new connection to the queue manager (recommended)

 Add Queue Manager

Specify new connection details

Provide details of the connection you want to set up

Queue manager name:

Connection details

Host name or IP address:

Port number:

Server-connection channel:

Add Queue Manager

Specify user identification details

Provide a userid name and password

Queue manager name:

IDCQM1

☒ Enable user identification

☐ User identification compatibility mode

Userid:

mqm

Password

☒ No password

MQ Explorer - Navigator

- IBM MQ
 - Queue Managers
 - EAIQM1 on 'icp-console.apps.ocp.cloudnativekube.com(30798)'
 - EAIQM2 on 'icp-console.apps.ocp.cloudnativekube.com(32588)'
 - IDCQM1 on 'icp-console.apps.ocp.cloudnativekube.com(30419)'
 - IDCQM1 on 'icp-proxy.apps.ocp.cloudnativekube.com(32182)'
 - Queues
 - Topics
 - Subscriptions
 - Channels

MQ Explorer - Content

Channels

Filter: Standard for Channels

Channel name	Channel type	Overall channel status
IVT.SVRCONN	Server-connection	Running

Using Runmqsc in client mode

SET MQSERVER=IVT.SVRCONN/TCP/icp-proxy.apps.ocp.cloudnativekube.com(32182)

Runmqsc -c IDCQM1

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>SET MQSERVER=IVT.SVRCONN/TCP/icp-proxy.apps.ocp.cloudnativekube.com(32182)

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>runmqsc -c IDCQM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager IDCQM1.

def ql(manual.q)
  1 : def ql(manual.q)
AMQ8006I: IBM MQ queue created.
end
  2 : end
2 command responses received.
```

```
C:\temp>type 1q-npwd.mqsc
define ql(DA) replace

C:\temp>runmqsc -c IDCQM1 < 1q-npwd.mqsc
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager IDCQM1.


      1 : define ql(DA) replace
AMQ8006I: IBM MQ queue created.
2 command responses received.
```

Part 2 - Custom MQ Image on RHOS 3.11 and 4.2 with ICP4i - TLS

With iCP4i on RHOS 4.2 for clients that are running outside the RHOS cluster to connect to queue managers TLS protocol must be used.

clients must use TLS to connect now.

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.1.0/com.ibm.mq.mcpak.doc/cc_conn_qm_openshift.htm

"You need an [OpenShift Route](#) to connect an application to an IBM® MQ queue manager from outside a Red Hat OpenShift cluster. You must enable TLS on your IBM MQ queue manager and client application, because [Server Name Indication](#) (SNI) is only available in the TLS protocol. The OpenShift Container Platform (OCP) Router uses SNI for routing requests to the IBM MQ queue manager. The required configuration of the OpenShift Route depends on the SNI behavior of your client application."

Documentation from the Helm chart with annotations

Configuring MQ objects

You have the following major options for configuring the MQ queue manager itself:

1. ~~Use the MQ web console interactively~~
2. Create a new image layer with your configuration **for remote administration baked-in via the supply of an mqsc file with TLS enabled SVRCONN definitions for admin and app.baked-in.**
3. ~~Configure remote administration over messaging, and use existing tools, such as runmqsc, MQ Explorer or the MQ Command Server.~~

The REST administrative API is not currently supported.

Configuring MQ using a new image layer

You can create a new container image layer, on top of the IBM MQ Advanced base image. You can add MQSC files to define MQ objects such as queues and topics, and place these files into `/etc/mqm` in your image. When the MQ pod starts, it will run any MQSC files found in this directory (in sorted order).

Example Dockerfile and MQSC script for creating a new image

In this example you will create a Dockerfile that creates two users:

- `admin` - Administrator user which is a member of the `mqm` group
- `app` - Client application user which is a member of the `mqclient` group. (You will also create this group)

You will also create a MQSC Script file called `config.mqsc` that will be run automatically when your container starts. This script will do the following:

- Create default local queues for my applications
- Create channels for use by the `admin` and `app` users
- Configure security to allow use of the channels by remote applications
- Create authority records to allow members of the `mqclient` group to access the Queue Manager and the default local queues.

First create a file called `config.mqsc`. This the MQSC file that will be run when an MQ container starts. It should contain the following:

```
* Create Local Queues that my application(s) can use.
DEFINE QLOCAL('EXAMPLE.QUEUE.1') REPLACE
DEFINE QLOCAL('EXAMPLE.QUEUE.2') REPLACE

* Create a Dead Letter Queue for undeliverable messages and set the Queue
Manager to use it.
DEFINE QLOCAL('EXAMPLE.DEAD.LETTER.QUEUE') REPLACE
ALTER QMGR DEADQ('EXAMPLE.DEAD.LETTER.QUEUE')

* Set ADOPTCTX to YES so we use the same userid passed for authentication
as the one for authorization and refresh the security configuration
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS)
ADOPTCTX(YES)
REFRESH SECURITY(*) TYPE(CONNAUTH)

* Create a entry channel for the Admin user and Application user
* add the TLS configuration values on channels here
DEFINE CHANNEL('EXAMP.ADMIN.SVRCONN') CHLTYPE(SVRCONN) REPLACE
DEFINE CHANNEL('EXAMP.APP.SVRCONN') CHLTYPE(SVRCONN) MCAUSER('app') REPLACE

REFRESH SECURITY TYPE(SSL)

* Set Channel authentication rules to only allow access through the two
channels we created and only allow admin users to connect through
EXAMPLE.ADMIN.SVRCONN
SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS)
DESCR('Back-stop rule - Blocks everyone') ACTION(REPLACE)
SET CHLAUTH('EXAMP.APP.SVRCONN') TYPE(ADDRESSMAP) ADDRESS('*')
USERSRC(CHANNEL) DESCR('Allows connection via APP channel') ACTION(REPLACE)
SET CHLAUTH('EXAMP.ADMIN.SVRCONN') TYPE(ADDRESSMAP) ADDRESS('*')
USERSRC(CHANNEL) DESCR('Allows connection via ADMIN channel')
ACTION(REPLACE)
SET CHLAUTH('EXAMP.ADMIN.SVRCONN') TYPE(BLOCKUSER) USERLIST('nobody')
DESCR('Allows admins on ADMIN channel') ACTION(REPLACE)

* Set Authority records to allow the members of the mqclient group to
connect to the Queue Manager and access the Local Queues which start with
"EXAMPLE."
SET AUTHREC OBJTYPE(QMGR) GROUP('mqclient') AUTHADD(CONNECT,INQ)
SET AUTHREC PROFILE('EXAMPLE.**') OBJTYPE(QUEUE) GROUP('mqclient')
AUTHADD(INQ,PUT,GET,BROWSE)
```

Next create a `Dockerfile` that expands on the MQ Advanced Server image to create the users and groups. It should contain the following, replacing `<IMAGE NAME>` with the MQ image you want to use as a base:

```
FROM <IMAGE NAME>
# Add the admin user as a member of the mqm group and set their password
USER root
RUN useradd admin -G mqm \
    && echo admin:passw0rd | chpasswd \
# Create the mqclient group
    && groupadd mqclient \
```

```
# Create the app user as a member of the mqclient group and set their
password
    && useradd app -G mqclient \
    && echo app:passw0rd | chpasswd
# Copy the configuration script to /etc/mqm where it will be picked up
automatically
USER mqm
COPY config.mqsc /etc/mqm/
```

Finally, build and push the image to your registry.

You can then use the new image when you deploy MQ into your cluster. You will find that once you have run the image you will be able to see your new default objects and users.

JSON log output

By default, the MQ container output is in JSON format, to better integrate with log aggregation services. On the command line, you can use utilities like 'jq' to format this output, for example:

```
kubectl logs foo-ibm-mq-0 | jq -r '.ibm_datetime + " " + .message'
```

Supplying certificates to be used for TLS

Create a matched set of keys and certs "locally" and test that your MQ Explorer (client) can connect to a local queue manager over TLS.

Use the server side keys and certs as input to the following.

The `pki.trust` and `pki.keys` allow you to supply details of Kubernetes secrets that contain TLS certificates. By doing so the TLS certificates will be imported into the container at runtime and MQ will be configured to use them. You can supply both certificates which contain only a public key and certificates that contain both public and private keys.

If you supply invalid files or invalid YAML objects then the container will terminate with an appropriate termination message. The next 2 sections will detail the requirements for supplying each type of certificate.

Supplying certificates which contain the public and private keys

When supplying a Kubernetes secret that contains a certificate files for both the public and private key you must ensure that the secret contains a file that ends in `.crt` and a file that ends in `.key` named the same. For example: `tls.crt` and `tls.key`. The extension of the file denotes whether the file is the public key (`.crt`) or the private key (`.key`) and must be correct. If your certificate has been issued by a Certificate Authority, then the certificate from the CA must be included as a separate file with the `.crt` extension. For example: `ca.crt`.

The format of the YAML objects for `pki.keys` value is as follows:

```
- name: mykey
  secret:
    secretName: mykeysecret
    items:
      - tls.key
      - tls.crt
```



```
- ca.crt
```

or alternatively in a single line you can supply the following: `- name: mykey, secret: {secretName: mykeysecret, items: [tls.key, tls.crt, ca.crt]}`

`name` must be set to a lowercase alphanumeric value and will be used as the label for the certificate in the keystore and queue manager.

`secret.secretName` must match the name of a Kubernetes secret that contains the TLS certificates you wish to import

`secret.items` must list the TLS certificate files contained in `secret.secretName` you want to import.

To supply the YAML objects when deploying via Helm you should use the following: `--set pki.keys[0].name=mykey,pki.keys[0].secret.secretName=mykeysecret,pki.keys[0].secret.items[0]=tls.key,pki.keys[0].secret.items[1]=tls.crt,pki.keys[0].secret.items[2]=ca.crt`

If you supply multiple YAML objects then the queue manager will use the first object chosen by the label name alphabetically. For example if you supply the following labels: `alabel`, `blabel` and `clabel`. The queue manager will use the certificate with the label `alabel` for its identity. This can be changed at runtime by executing the MQSC command: `ALTER QMGR CERTLABL('<new label>')`.

Supplying certificates which contain only the public key

When supplying a Kubernetes secret that contains a certificate file with only the public key you must ensure that the secret contains files that have the extension `.crt`. For example: `tls.crt` and `ca.crt`.

The format of the YAML objects for `pki.trust` value is as follows:

```
- secret:
  secretName: mycertificate
  items:
    - tls.crt
```

or alternatively in a single line you can supply the following: `- secret: {secretName: mycertificate, items: [tls.crt]}`

`secret.secretName` must match the name of a Kubernetes secret that contains the TLS certificates you wish to add.

`secret.items` must list the TLS certificate files contained in `secret.secretName` you want to add.

To supply the YAML objects when deploying via Helm you should use the following: `--set pki.trust[0].secret.secretName=mycertificate,pki.trust[0].secret.items[0]=tls.crt`

If you supply multiple YAML objects then all of the certificates specified will be added into the queue manager and MQ Console Truststores.

Creating TLS key and certs set – In the local “Admin” environment

Points of note

Clarification of “extract”/“add” versus “export”/“import”

SSL uses public/private keys to provide a flexible encryption scheme that can be setup at the time of the secure transaction. When a certificate is created, it contains both the public and private keys.

The "extract" and "add" functions deal with ONLY the public keys. That is, the "extract" gets the public key of a certificate from a database and the "add" puts the public key into a database. No passwords are required because the private key is not obtained.

The "export" and "import" functions deal with BOTH the public and private keys for a certificate. Passwords are required due to the private key.

Summary of steps

Step 1: Client: Create TLS client key database

Step 2: Client: Create certificate

Step 3: Client: Extract the public TLS client certificate and copy it to the TLS server side

Step 4: Server: Create TLS server key database

Step 5: Server: Create certificate

Step 6: Server: Add the TLS client certificate to the Queue Manager's key database.

Step 7: Server: Extract the public TLS server certificate and copy it to the TLS client side

Step 8: Client: Add the TLS client certificate to the Client's key database.

Step 9: Server: Run MQSC commands for TLS server side queue manager SVRCONN

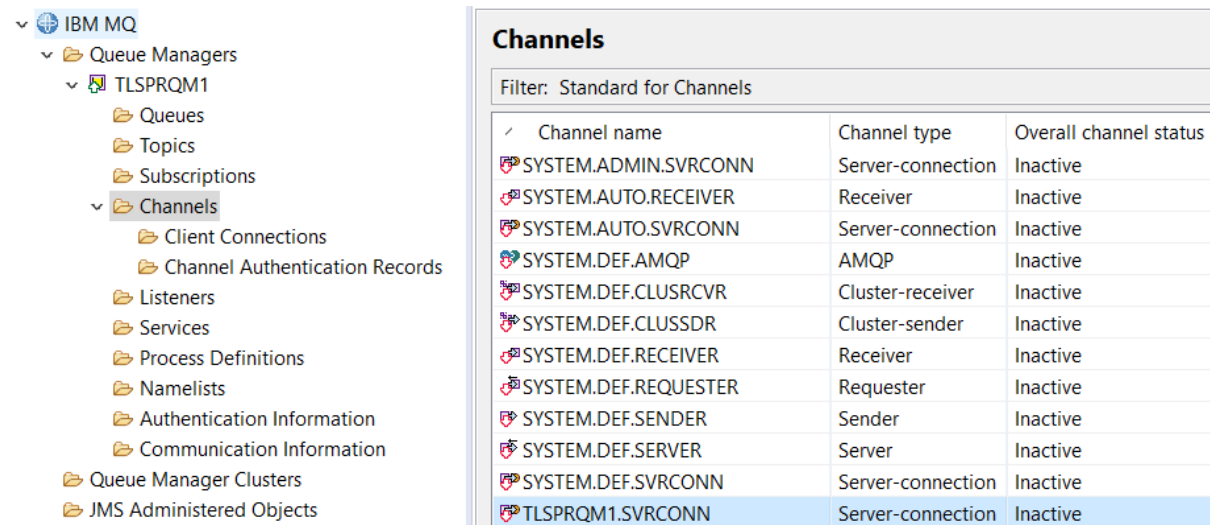
Step 10: Client: Verifying the TLS set up with MQCERTCK

Step 11: Client: Use runmqsc in client mode to connect to the QMGR and administer it

Local Environment

Queue Manager Name – TLSPRQM1 (TLS enabled Production Queue Manager 1)

Listening on port - 2414



The screenshot shows the IBM MQ console interface. On the left, a tree view displays the hierarchy: IBM MQ > Queue Managers > TLSPRQM1 > Channels. The 'Channels' folder is selected. On the right, a table titled 'Channels' displays the list of channels. The table has three columns: 'Channel name', 'Channel type', and 'Overall channel status'. The filter is set to 'Standard for Channels'. The table lists 15 channels, all of which are 'Inactive'. The channel 'TLSPRQM1.SVRCONN' is highlighted in blue at the bottom of the list.

Channel name	Channel type	Overall channel status
SYSTEM.ADMIN.SVRCONN	Server-connection	Inactive
SYSTEM.AUTO.RECEIVER	Receiver	Inactive
SYSTEM.AUTO.SVRCONN	Server-connection	Inactive
SYSTEM.DEF.AMQP	AMQP	Inactive
SYSTEM.DEF.CLUSRCVR	Cluster-receiver	Inactive
SYSTEM.DEF.CLUSSDR	Cluster-sender	Inactive
SYSTEM.DEF.RECEIVER	Receiver	Inactive
SYSTEM.DEF.REQUESTER	Requester	Inactive
SYSTEM.DEF.SENDER	Sender	Inactive
SYSTEM.DEF.SERVER	Server	Inactive
SYSTEM.DEF.SVRCONN	Server-connection	Inactive
TLSPRQM1.SVRCONN	Server-connection	Inactive

Runmqsc TLSPRQM1 to create SVRCONN to use

Note: on windows This should have been MCAUSER('MUSR_MQADMIN') rather than mqm

```
DEFINE CHANNEL(TLSPRQM1.SVRCONN) CHLTYPE(SVRCONN)
```

```
SET CHLAUTH(TLSPRQM1.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)
```

```
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE)  
ADOPTCTX(YES)
```

```
*SET CHLAUTH(TLSPRQM1.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')
```

```
SET CHLAUTH(TLSPRQM1.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*)  
MCAUSER('MUSR_MQADMIN')
```

```
REFRESH SECURITY TYPE(CONNAUTH)
```

Step 1: Client: Create SSL client key database

Windows command prompt running as administrator

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\windows\system32>
```

The default directory for MQ 9.1 in this machine is:

```
cd C:\Program Files\IBM\MQ\bin
```

If you have multiple installations of MQ versions set the correct installation

```
setmqenv -n Installation1
```

```
C:\Program Files\IBM\MQ\bin>setmqenv -n Installation1
C:\Program Files\IBM\MQ\bin>
```

This PC > Windows (C:) > Program Files > IBM > MQ > bin		
SS	<input type="checkbox"/>	Name
	<input checked="" type="checkbox"/>	runmqckm
	<input type="checkbox"/>	runmqdlq
		Date modified
		2/07/2019 12:00 AM
		2/07/2019 12:00 AM

Create TLS client key database

```
runmqckm -keydb -create -db "C:\Program Files\IBM\MQ\arnold.kdb" -pw clientpass -type cms -
expire 365 -stash
```

```
C:\Program Files\IBM\MQ\bin>runmqckm -keydb -create -db "C:\Program Files\IBM\MQ\arnold.kdb" -pw clientpass -type cms -e
xpire 365 -stash
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
C:\Program Files\IBM\MQ\bin>
```

Files are created by default in c:\program files\ibm\mq

Windows (C:) > Program Files > IBM > MQ			
	<input type="checkbox"/>	Name	Date modified
	<input checked="" type="checkbox"/>	arnold.kdb	18/12/2019 4:34 PM
	<input checked="" type="checkbox"/>	arnold.rdb	18/12/2019 4:34 PM
	<input checked="" type="checkbox"/>	arnold.sth	18/12/2019 4:34 PM
		Type	
		KDB File	
		RDB File	
		STH File	

Step 2: Client: Create certificate

```
Administrator: Command Prompt

C:\Program Files\IBM\MQ\bin>cd ..

C:\Program Files\IBM\MQ>dir arnold.*
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

Directory of C:\Program Files\IBM\MQ

18/12/2019  04:34 PM                88 arnold.kdb
18/12/2019  04:34 PM                80 arnold.rdb
18/12/2019  04:34 PM               193 arnold.sth
```

Create certificate

```
runmqckm -cert -create -db "C:\program files\ibm\mq\arnold.kdb" -pw clientpass -label
ibmmqarnold -dn "CN=arnold,OU=Support,O=IBM,ST=NC,C=" -expire 365
```

```
C:\Program Files\IBM\MQ>runmqckm -cert -create -db "C:\program files\ibm\mq\arnold.kdb" -pw clientpass -label ibmmqarnol
d -dn "CN=arnold,OU=Support,O=IBM,ST=NC,C=" -expire 365
5724-H72 (C) Copyright IBM Corp. 1994, 2019.

C:\Program Files\IBM\MQ>
```

List the certificate

List the certificate:

```
runmqckm -cert -list -db "c:\program files\ibm\mq\arnold.kdb" -pw clientpass
```

```
C:\Program Files\IBM\MQ>runmqckm -cert -list -db "c:\program files\ibm\mq\arnold.kdb" -pw clientpass
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Certificates in database c:\program files\ibm\mq\arnold.kdb:
    ibmmqarnold
```

Dir the files – note change in size of arnold.kdb

```
C:\Program Files\IBM\MQ>dir arnold.*
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

Directory of C:\Program Files\IBM\MQ

18/12/2019  04:56 PM           5,088 arnold.kdb
18/12/2019  04:56 PM                80 arnold.rdb
18/12/2019  04:34 PM               193 arnold.sth
```

Step 3: Client: Extract the public SSL client certificate and copy it to the SSL server side
Extract the public TLS client certificate for copy to the TLS server side

Extract the certificate:

```
runmqckm -cert -extract -db "c:\Program Files\IBM\MQ\arnold.kdb" -pw clientpass -label  
ibmmqarnold -target arnold.crt -format ascii
```

```
C:\Program Files\IBM\MQ>runmqckm -cert -extract -db "c:\Program Files\IBM\MQ\arnold.kdb" -pw clientpass -label ibmmqarnold -target arnold.crt -format ascii  
5724-H72 (C) Copyright IBM Corp. 1994, 2019.  
C:\Program Files\IBM\MQ>
```

Note: The file "arnold.crt" is created in the current directory

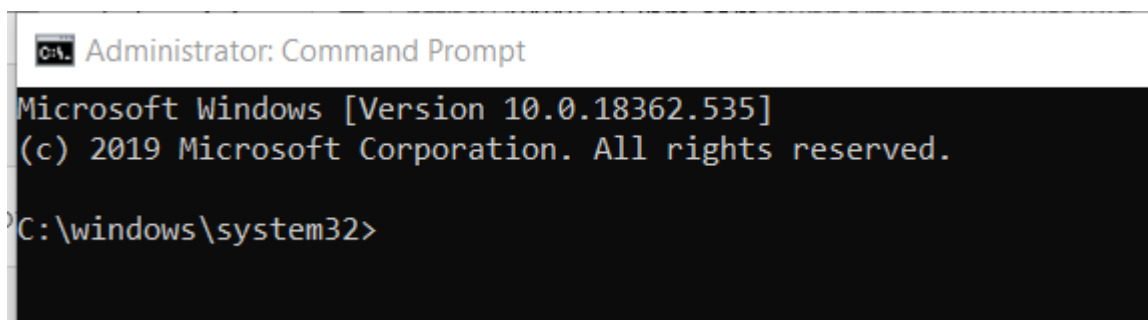
```
Directory of C:\Program Files\IBM\MQ  
  
18/12/2019  05:03 PM                836 arnold.crt  
18/12/2019  04:56 PM            5,088 arnold.kdb  
18/12/2019  04:56 PM                80 arnold.rdb  
18/12/2019  04:34 PM                193 arnold.sth
```

Windows (C:) > Program Files > IBM > MQ			
	Name	Date modified	Type
<input checked="" type="checkbox"/>	arnold	18/12/2019 5:03 PM	Security Certificate
<input type="checkbox"/>	arnold.kdb	18/12/2019 4:56 PM	KDB File
<input type="checkbox"/>	arnold.rdb	18/12/2019 4:56 PM	RDB File
<input type="checkbox"/>	arnold.sth	18/12/2019 4:34 PM	STH File

Arnold.crt is our client side public TLS certificate extracted from the client database that can be used by the server side queue manager.

Step 4: Server: Create SSL server key database Step 5: Server: Create certificate

Windows command prompt running as administrator



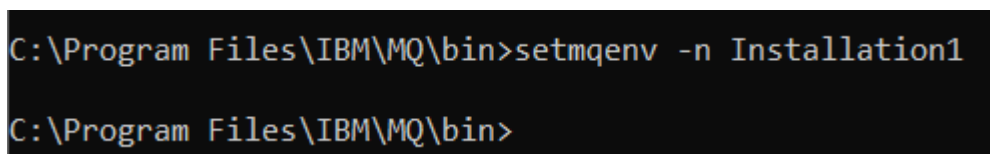
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\windows\system32>
```

The default directory for MQ 9.1 in this machine is:

```
cd C:\Program Files\IBM\MQ\bin
```

If you have multiple installations of MQ versions set the correct installation

```
setmqenv -n Installation1
```



```
C:\Program Files\IBM\MQ\bin>setmqenv -n Installation1
C:\Program Files\IBM\MQ\bin>
```

Change to the directory where the Queue Manager key database will be located:

```
Cd C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl
```

(linux cd /var/mqm/qmgrs/ TLSPRQM1/ssl/)


```

C:\Program Files\IBM\MQ>cd C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl

C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>dir
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

Directory of C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl

18/12/2019  04:12 PM    <DIR>          .
18/12/2019  04:12 PM    <DIR>          ..
               0 File(s)                0 bytes

```

Create SSL server key database

```

runmqckm -keydb -create -db "C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.kdb" -
pw serverpass -type cms -expire 365 -stash

```

```

C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>runmqckm -keydb -create -db "C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.
kdb" -pw serverpass -type cms -expire 365 -stash
5724-H72 (C) Copyright IBM Corp. 1994, 2019.

C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>

```

These are the files that are created: dir C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl

```

C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>dir C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

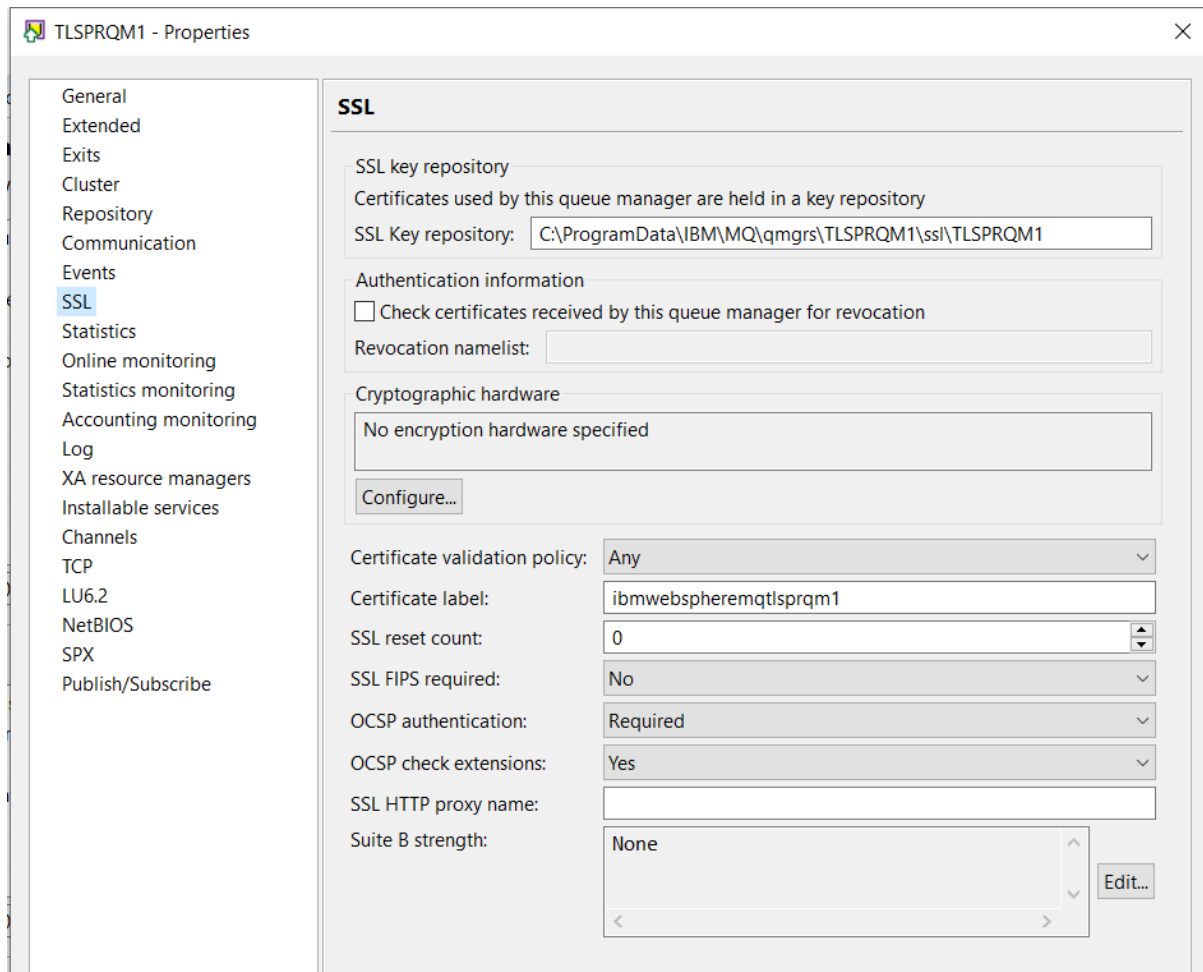
Directory of C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl

18/12/2019  06:41 PM    <DIR>          .
18/12/2019  06:41 PM    <DIR>          ..
18/12/2019  06:41 PM                88 TLSPRQM1.kdb
18/12/2019  06:41 PM                80 TLSPRQM1.rdb
18/12/2019  06:41 PM               193 TLSPRQM1.sth

```

Step 5: Server: Create certificate

Important: There is a default certificate label on the Queue Manager side.



If you choose not to use the default `ibmwebspheremq<lower case queue manager name>` you will need to change this value in the queue manager.

In this example we will use a non default. "`ibmmq<lower case queue manager name>`" i.e. **`ibmmqtlsprqm1`**

Therefore I will change the queue manager's cert label

TLSPRQM1 - Properties

<ul style="list-style-type: none"> General Extended Exits Cluster Repository Communication Events SSL Statistics Online monitoring Statistics monitoring Accounting monitoring Log XA resource managers Installable services Channels TCP LU6.2 	<h3>SSL</h3> <p>SSL key repository</p> <p>Certificates used by this queue manager are held in a key repository</p> <p>SSL Key repository: <input type="text" value="C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1"/></p> <p>Authentication information</p> <p><input type="checkbox"/> Check certificates received by this queue manager for revocation</p> <p>Revocation namelist: <input type="text"/></p> <p>Cryptographic hardware</p> <p><input type="text" value="No encryption hardware specified"/></p> <p><input type="button" value="Configure..."/></p> <p>Certificate validation policy: <input type="text" value="Any"/></p> <p>Certificate label: <input type="text" value="ibmmqtlsprqm1"/></p>
--	--

TLS server certificate setup

Create the certificate:

```
runmqckm -cert -create -db "C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.kdb" -pw
serverpass -label ibmmqtlsprqm1 -dn "CN=TLSPRQM1,OU=Support,O=IBM,ST=NC,C=" -expire 365
```

```
C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>runmqckm -cert -create -db "C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.kdb" -pw serverpass -label ibmmqtlsprqm1 -dn "CN=TLSPRQM1,OU=Support,O=IBM,ST=NC,C=" -expire 365
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>
```

List the certificate:

```
runmqckm -cert -list -db "C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.kdb" -pw
serverpass
```

```
C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>runmqckm -cert -list -db "C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.kdb" -pw serverpass
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Certificates in database C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.kdb:
    ibmmqtlsprqm1
```

Notice that the file size for the kdb file was increased

```
dir C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl
```

```
C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>dir C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

Directory of C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl

18/12/2019  06:41 PM    <DIR>          .
18/12/2019  06:41 PM    <DIR>          ..
18/12/2019  06:48 PM             5,088 TLSPRQM1.kdb
18/12/2019  06:48 PM              80 TLSPRQM1.rdb
18/12/2019  06:41 PM             193 TLSPRQM1.sth
```

Step 6: Server: Add the SSL client certificate to the Queue Manager's key database.

Add the SSL client certificate **arnold.crt** to the Queue Manager's key database.

Change to the directory where the queue manager key database is located:

(linux `cd /var/mqm/qmgrs/TLSPRQM1/ssl/`)

Cd `C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl`

Notice that the file `rivera.crt` is in the directory: `ls -l -rw----- 1 rivera mqm 5080 2014-03-04 11:43 QM_71SSL.kdb -rw----- 1 rivera mqm 80 2014-03-04 11:43 QM_71SSL.rdb -rw----- 1 rivera mqm 129 2014-03-04 11:37 QM_71SSL.sth -rw----- 1 rivera mqm 776 2014-03-04 11:50 rivera.crt (new file)`

Add the client certificate `arnold.crt`:

Windows (C:) > Program Files > IBM > MQ			
	Name	Date modified	Type
<input checked="" type="checkbox"/>	arnold	18/12/2019 5:03 PM	Security Certificate
<input type="checkbox"/>	arnold.kdb	18/12/2019 4:56 PM	KDB File
<input type="checkbox"/>	arnold.rdb	18/12/2019 4:56 PM	RDB File
<input type="checkbox"/>	arnold.sth	18/12/2019 4:34 PM	STH File

`runmqckm -cert -add -db "C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.kdb" -pw serverpass -label ibmmqarnold -file "c:\program files\ibm\mq\arnold.crt" -format ascii`

```
C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>runmqckm -cert -add -db "C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.kdb" -pw serverpass -label ibmmqarnold -file "c:\program files\ibm\mq\arnold.crt" -format ascii
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>
```

List the certificates:

`runmqckm -cert -list -db "C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.kdb" -pw serverpass`

```
C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>runmqckm -cert -list -db "C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.kdb" -pw serverpass
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Certificates in database C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.kdb:
ibmmqtlspqm1
ibmmqarnold
```

Notice the increase in size for the kdb file:

```

C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>dir
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

Directory of C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl

18/12/2019  06:41 PM    <DIR>          .
18/12/2019  06:41 PM    <DIR>          ..
18/12/2019  07:02 PM             10,088 TLSPRQM1.kdb
18/12/2019  07:02 PM              80 TLSPRQM1.rdb
18/12/2019  06:41 PM             193 TLSPRQM1.sth

```

Step 7: Server: Extract the public TLS server certificate and copy it to the TLS client side

Extract the public TLS server certificate to make it available to the TLS client side

Change to the directory where the queue manager key database is located:

(linux `cd /var/mqm/qmgrs/TLSPRQM1/ssl/`)

Cd C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl

```
runmqckm -cert -extract -db "C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.kdb" -pw
serverpass -label ibmmqtlspqm1 -target TLSPRQM1.crt -format ascii
```

The file "TLSPRQM1.crt" is created in the current directory

```

C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl>runmqckm -cert -extract -db "C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1.
kdb" -pw serverpass -label ibmmqtlspqm1 -target TLSPRQM1.crt -format ascii
5724-H72 (C) Copyright IBM Corp. 1994, 2019.

```

Dir C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl

```

C:\ProgramData\IBM\MQ\mqgrs\TLSPRQM1\ssl>dir
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

Directory of C:\ProgramData\IBM\MQ\mqgrs\TLSPRQM1\ssl

18/12/2019  07:12 PM    <DIR>          .
18/12/2019  07:12 PM    <DIR>          ..
18/12/2019  07:12 PM             842 TLSPRQM1.crt
18/12/2019  07:02 PM          10,088 TLSPRQM1.kdb
18/12/2019  07:02 PM             80 TLSPRQM1.rdb
18/12/2019  06:41 PM           193 TLSPRQM1.sth

```

Step 8: Client: Add the SSL client certificate to the Client's key database.

The MQ Queue Manager Signer Certificate, " TLSPRQM1.crt" needs to be made available to the host where the MQ Client is located.

Change directory to the local client side key database arnold.kdb

cd C:\Program Files\IBM\MQ

```

C:\Program Files\IBM\MQ>dir arnold.*
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

Directory of C:\Program Files\IBM\MQ

18/12/2019  05:03 PM             836 arnold.crt
18/12/2019  04:56 PM          5,088 arnold.kdb
18/12/2019  04:56 PM             80 arnold.rdb
18/12/2019  04:34 PM           193 arnold.sth

```

Client side key database

Windows (C:) > Program Files > IBM > MQ				
<input type="checkbox"/>	Name	Date modified	Type	Size
<input type="checkbox"/>	arnold	18/12/2019 5:03 PM	Security Certificate	1 KB
<input checked="" type="checkbox"/>	arnold.kdb	18/12/2019 4:56 PM	KDB File	5 KB
<input type="checkbox"/>	arnold.rdb	18/12/2019 4:56 PM	RDB File	1 KB
<input type="checkbox"/>	arnold.sth	18/12/2019 4:34 PM	STH File	1 KB

Add the queue manager certificate to client side key database

Queue manager certificate

Windows (C:) > ProgramData > IBM > MQ > qmgrs > TSPRQM1 > ssl

<input type="checkbox"/> Name	Date modified	Type
<input checked="" type="checkbox"/> TSPRQM1	18/12/2019 7:12 PM	Security Certificate
TSPRQM1.kdb	18/12/2019 7:02 PM	KDB File
TSPRQM1.rdb	18/12/2019 7:02 PM	RDB File
TSPRQM1.sth	18/12/2019 6:41 PM	STH File

```
runmqckm -cert -add -db "C:\Program Files\IBM\MQ\arnold.kdb" -pw clientpass -label
ibmmqtlspqm1 -file "c:\programData\ibm\mq\qmgrs\TSPRQM1\ssl\TSPRQM1.crt" -format ascii
```

```
C:\Program Files\IBM\MQ>runmqckm -cert -add -db "C:\Program Files\IBM\MQ\arnold.kdb" -pw clientpass -label ibmmqtlspqm
1 -file "c:\programData\ibm\mq\qmgrs\TSPRQM1\ssl\TSPRQM1.crt" -format ascii
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
C:\Program Files\IBM\MQ>
```

List the certificate

```
runmqckm -cert -list -db "C:\Program Files\IBM\MQ\arnold.kdb" -pw clientpass
```

```
C:\Program Files\IBM\MQ>runmqckm -cert -list -db "C:\Program Files\IBM\MQ\arnold.kdb" -pw clientpass
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Certificates in database C:\Program Files\IBM\MQ\arnold.kdb:
    ibmmqarnold
    ibmmqtlspqm1
```

Note the change in size of the arnold.kdb file

```
C:\Program Files\IBM\MQ>dir arnold.*
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

Directory of C:\Program Files\IBM\MQ

18/12/2019  05:03 PM                836 arnold.crt
18/12/2019  07:28 PM            10,088 arnold.kdb
18/12/2019  07:28 PM                80 arnold.rdb
18/12/2019  04:34 PM                193 arnold.sth
```

Step 9: Server: Run MQSC commands for TLS server side queue manager

GSKit will add the ".kdb" suffix for the key database. So don't add the kdb extension

You MUST provide the value for SSLKEYR as follows:

full path name of the key store MINUS the .kdb suffix (the .kdb is added at runtime):

Windows Full path name with suffix:

C:\ProgramData\IBM\MQ\qmgrs\TSPRQM1\ssl\TSPRQM1.kdb

Windows Full path name minus suffix: C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1

(Linux full path name minus suffix /var/mqm/qmgrs/ TLSPRQM1/ssl/ TLSPRQM1)

Watch out for single quotes when cut and pasting – you might have to type over them in the command after the paste.

Runmqsc TLSPRQM1

ALTER QMGR SSLKEYR('C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1')

ALTER QMGR SSLFIPS(NO)

```
ALTER QMGR SSLKEYR('C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1')
  2 : ALTER QMGR SSLKEYR('C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\ssl\TLSPRQM1')
AMQ8005I: IBM MQ queue manager changed.
ALTER QMGR SSLFIPS(NO)
  3 : ALTER QMGR SSLFIPS(NO)
AMQ8005I: IBM MQ queue manager changed.
```

DEFINE CHANNEL('TLSPRQM1.SVRCONN') CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCIPH(ANY_TLS12)
SSLCAUTH(REQUIRED) REPLACE

SET CHLAUTH(TLSPRQM1.SVRCONN) TYPE(BLOCKUSER) USERLIST('nobody')

REFRESH SECURITY TYPE(SSL)

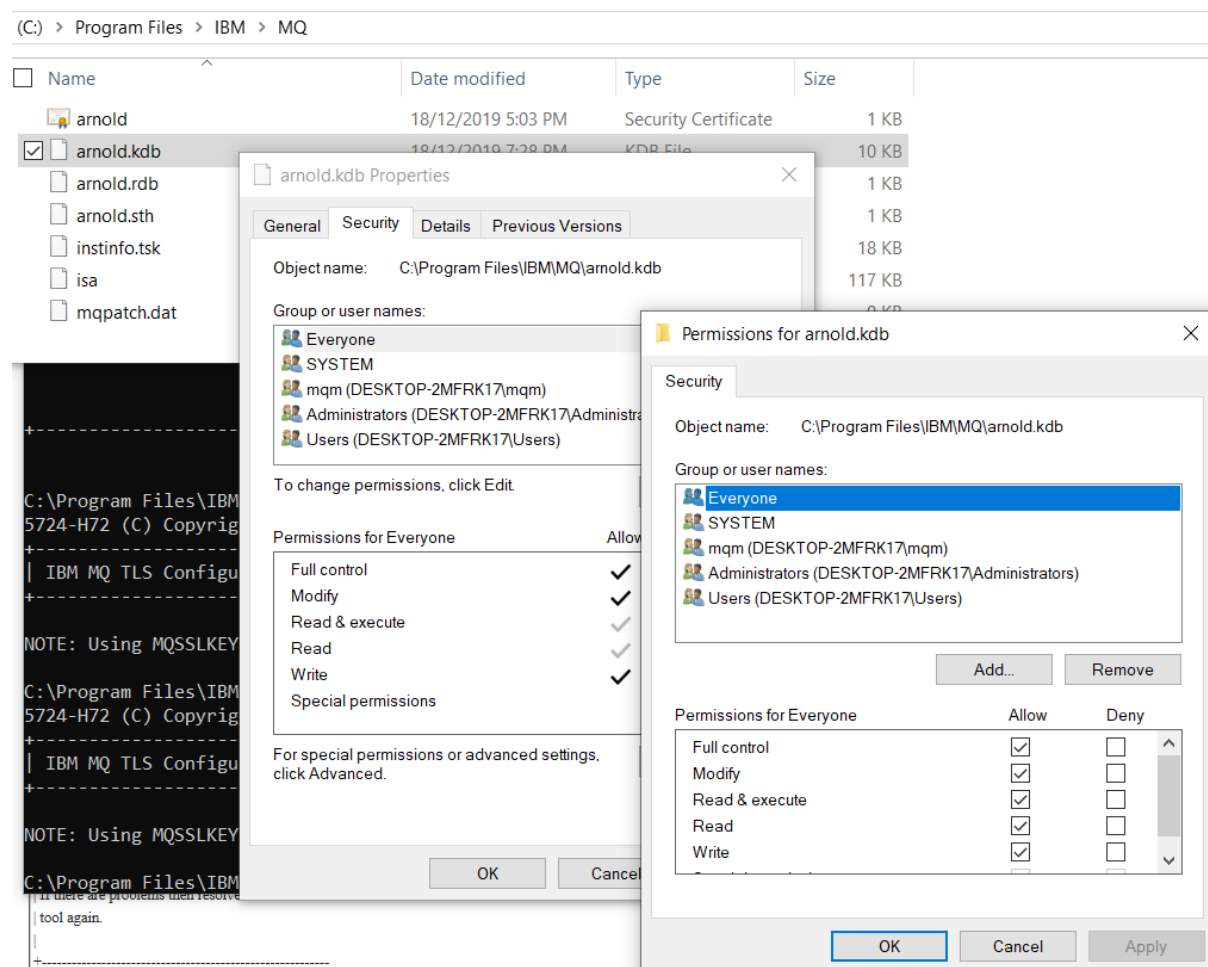
end

```
DEFINE CHANNEL('TLSPRQM1.SVRCONN') CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCIPH(ANY_TLS12) SSLCAUTH(REQUIRED) REPLACE
  7 : DEFINE CHANNEL('TLSPRQM1.SVRCONN') CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCIPH(ANY_TLS12) SSLCAUTH(REQUIRED) REPLACE
AMQ8014I: IBM MQ channel created.
REFRESH SECURITY TYPE(SSL)
  8 : REFRESH SECURITY TYPE(SSL)
AMQ8560I: IBM MQ security cache refreshed.
end
```


Step 10: Verifying the TLS set up with MQCERTCK

File read/write attributes on the .KDB

There might be a problem with read/write access to the files



Notes from an IBM tech note

Check the AMQERR01.LOG

AMQ9660: SSL key repository: password stash file absent or unusable. EXPLANATION: The SSL key repository cannot be used because MQ cannot obtain a password to access it. Reasons giving rise to this error include: (a) the key database file and password stash file are not present in the location configured for the key repository, (b) the key database file exists in the correct place but that no password stash file has been created for it, (c) the files are present in the correct place but the userid under which MQ is running does not have permission to read them, (d) one or both of the files are corrupt.

ACTION: Ensure that the key repository variable is set to where the key database file is. Ensure that a password stash file has been associated with the key database file in the same directory, and that the userid under which MQ is running has read access to both files. If both are already present and readable in the correct place, delete and recreate them. Restart the channel.

file permissions for the ssl files:

```
cd /var/mqm/qmgrs/TLSPRQM1/ssl ls -l -rw----- 1 davidarnold mqm 782 2014-03-04 12:12
TLSPRQM1.crt -rw----- 1 davidarnold mqm 10080 2014-03-04 11:51 TLSPRQM1.kdb -rw----- 1
arnold mqm 80 2014-03-04 11:51 TLSPRQM1.rdb -rw----- 1 davidarnold mqm 129 2014-03-04
11:37 TLSPRQM1.sth -rw----- 1 davidarnold mqm 776 2014-03-04 11:50 arnold.crt
```

Notice that the userid “davidarnold” is being used in this scenario. This user is a member of the “mqm” group, but the SSL files were created in such a way that other members of the group cannot read/write the files.

Change the permissions to allow the group “mqm” to read and write:

```
/var/mqm/qmgrs/TLSPRQM1/ssl chmod 660 *
```

```
riviera@veracruz: /var/mqm/qmgrs/TLSPRQM1/ssl ls -l -rw-rw---- 1 davidarnold mqm 782 2014-03-
04 12:12 TLSPRQM1.crt -rw-rw---- 1 davidarnold mqm 10080 2014-03-04 11:51 TLSPRQM1.kdb -rw-
rw---- 1 davidarnold mqm 80 2014-03-04 11:51 TLSPRQM1.rdb -rw-rw---- 1 davidarnold mqm 129
2014-03-04 11:37 TLSPRQM1.sth -rw-rw---- 1 davidarnold mqm 776 2014-03-04 11:50 arnold.crt
```

Using MQCERTCK

```
mqcertck <QMName> -clientkeyr <Client Key Repository> -clientchannel <Channel Name> -
clientusername <Client Username> -clientlabel <Client Certificate label> -clientport <Free
Port Number>
```

Set the environment variable for the client side key database arnold.kdb

set MQSSLKEYR=C:\Program Files\IBM\MQ\arnold

set the environment variable for MQSERVER to specific the client channel

Set MQSERVER=TLSPRQM1.SVRCONN/TCP/localhost(2414)

Try mqcertck with -clientuser mqm

```
C:\Program Files\IBM\MQ>mqcertck TLSPRQM1 -clientchannel TLSPRQM1.SVRCONN -clientuser
mqm -clientport 5857
```

5724-H72 (C) Copyright IBM Corp. 1994, 2019.

```
+-----
```

```
| IBM MQ TLS Configuration Test tool
```

```
+-----
```

NOTE: Using MQSSLKEYR environment variable

```
C:\Program Files\IBM\MQ>mqcertck TLSPRQM1 -clientchannel TLSPRQM1.SVRCONN -clientuser mqm -clientport 5857
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
+-----
| IBM MQ TLS Configuration Test tool
+-----

NOTE: Using MQSSLKEYR environment variable
C:\Program Files\IBM\MQ>
```

Try mqcertck with -clientlabel ibmmqtlspqm1

```
C:\Program Files\IBM\MQ>mqcertck TLSPQM1 -clientchannel TLSPQM1.SVRCONN -clientlabel  
ibmmqtlsprqm1 -clientport 5857
```

```
C:\Program Files\IBM\MQ>mqcertck TLSPQM1 -clientchannel TLSPQM1.SVRCONN -clientlabel ibmmqtlsprqm1 -clientport 5857  
5724-H72 (C) Copyright IBM Corp. 1994, 2019.  
+-----  
| IBM MQ TLS Configuration Test tool  
+-----  
  
NOTE: Using MQSSLKEYR environment variable  
  
C:\Program Files\IBM\MQ>
```

Step 11: Client: use runmqsc in client mode to connect to the TLSPRQM1 and administer it

Set the environment variable to point to the client's key database

Windows (C:) > Program Files > IBM > MQ			
	Name	Date modified	Type
	arnold	18/12/2019 5:03 PM	Security Certificate
<input checked="" type="checkbox"/>	arnold.kdb	18/12/2019 7:28 PM	KDB File
	arnold.rdb	18/12/2019 7:28 PM	RDB File
	arnold.sth	18/12/2019 4:34 PM	STH File

Windows Full path name with suffix: C:\Program Files\IBM\MQ\arnold.kdb

Windows Full path name minus suffix: C:\Program Files\IBM\MQ\arnold

```
set MQSSLKEYR=C:\Program Files\IBM\MQ\arnold
```

```
C:\Program Files\IBM\MQ>set MQSSLKEYR=C:\Program Files\IBM\MQ\arnold

C:\Program Files\IBM\MQ>echo %MQSSLKEYR%
C:\Program Files\IBM\MQ\arnold
```

Using MQSERVER for the client channel won't work

Set the environment variable for the MQSERVER

Set MQSERVER=TLSPRQM1.SVRCONN/TCP/localhost(2414)

```
C:\Program Files\IBM\MQ>Set MQSERVER= TLSPRQM1.SVRCONN/TCP/localhost(2414)

C:\Program Files\IBM\MQ>echo %MQSERVER%
TLSPRQM1.SVRCONN/TCP/localhost(2414)
```

Note that you cannot set the cipherspec for the client side channel to use.

If you runmqsc -c TLSPRQM1 it fails

```
C:\Program Files\IBM\MQ>runmqsc -c TLSPRQM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager TLSPRQM1.

AMQ9709E: SSL/TLS initialization error.

0 command responses received.

C:\Program Files\IBM\MQ>
```

In the logs

Windows (C:) > ProgramData > IBM > MQ > qmgrs > TLSPRQM1 > errors

<input type="checkbox"/>	Name	Date modified
<input checked="" type="checkbox"/>	AMQERR01	18/12/2019 11:08 PM

MQ9639E: Remote channel 'TLSPRQM1.SVRCONN' did not specify a CipherSpec.

EXPLANATION:

Remote channel 'TLSPRQM1.SVRCONN' did not specify a CipherSpec when the local channel expected one to be specified.

Using MQCCDT for the client channel

Notes:

Do we have to use MQCCDT file in order for runmqsc to client connect because the MQSERVER limits the details we can specify for the connection to the SVRCONN

A CCDT enables the specification of cipher specs etc to be specified on the client end

See <https://www-01.ibm.com/support/docview.wss?uid=swg27045974&aid=1>

A CCDT file always have at least one channel: SYSTEM.DEF.CLNTCONN # It is the default, but causes confusion. Do not use it.

We need a CLNTCONN channel something like the following

display CHANNEL(TLSPRQM1.SVRCONN) all AMQ8414: Display Channel details.

CHANNEL(TLSPRQM1.SVRCONN)

CHLTYPE(CLNTCONN) AFFINITY(PREFERRED) CERTLABL() CLNTWGHT(0)
COMPHDR(NONE) COMPMMSG(NONE)

CONNNAME(localhost(2414)) DEFRECON(NO) DESCR() HBINT(300)
KAINT(AUTO) LOCLADDR() MAXMSGL(4194304) MODENAME()
PASSWORD() QMNAME(TLSPRQM1) RCVDATA() RCVEXIT()
SCYDATA() SCYEXIT() SENDDATA() SENDEXIT() SHARECNV(10)

SSLCIPH(ANY_TLS12) SSLPEER() TPNAME() TRPTYPE(TCP) USERID()

Set up the CCDT

Create an client connection channel MQSC file

Make sure you add the **CERTLAB** to the CLNTCONN definition and retry

```
DEFINE CHANNEL(TLSPRQM1.SVRCONN) +
```

```
    CHLTYPE(CLNTCONN) +
```

```
    TRPTYPE(TCP) +
```

```
    CONNAME('localhost(2414)') +
```

```
    CERTLABL('ibmmqarnold') +
```

```
    QMNAME(TLSPRQM1) +
```

```
    SSLCIPH(ANY_TLS12) +
```

```
    REPLACE
```

```
C:\temp>dir TLSPRQM1CLTCONN.mqsc
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

Directory of C:\temp

19/12/2019  10:38 AM                197 TLSPRQM1CLTCONN.mqsc
               1 File(s)                197 bytes
               0 Dir(s)  211,153,235,968 bytes free

C:\temp>type TLSPRQM1CLTCONN.mqsc
DEFINE CHANNEL(TLSPRQM1.SVRCONN) +
    CHLTYPE(CLNTCONN) +
    TRPTYPE(TCP) +
    CONNAME('localhost(2414)') +
    CERTLABL('ibmmqarnold') +
    QMNAME(TLSPRQM1) +
    SSLCIPH(ANY_TLS12) +
    REPLACE
```

Runmqsc with the -n (no queue manager switch) to create a CCDT from the MQSC script

```
runmqsc -n < c:\temp\TLSPRQM1CLTCONN.MQSC
```

```

C:\temp>runmqsc -n < c:\temp\TLSPRQM1CLTCONN.MQSC
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting local MQSC for 'AMQCLCHL.TAB'.
      1 : DEFINE CHANNEL(TLSPRQM1.SVRCONN) +
        :     CHLTYPE(CLNTCONN) +
        :     TRPTYPE(TCP) +
        :     CONNAME('localhost(2414)') +
        :     QMNAME(TLSPRQM1) +
        :     SSLCIPH(ANY_TLS12) +
        :     REPLACE
AMQ8014I: IBM MQ channel created.
      :
      :
No commands have a syntax error.

```

If the default CCDT TAB name and default CCDT TAB directory are not used, we will need to set up the environment variable for path to AMQCLCHL.TAB

Because I have used all defaults the MQClient code will find and use the default AMQCLCHL.TAB

On Linux

```
export MQCHLTAB=/var/mqm/AMQCLCHL.TAB
```

```
echo $MQCHLTAB
```

On windows

```
set MQCHLTAB= C:\ProgramData\IBM\MQ\AMQCLCHL.TAB
```

```
echo %MQCHLTAB%
```

```
C:\ProgramData\IBM\MQ
```

Windows (C:) > ProgramData > IBM > MQ			
Name		Date modified	Type
<input type="checkbox"/>	AMQCLCHL.TAB	19/12/2019 10:04 AM	TAB File

```

C:\temp>cd \programData\ibm\mq
C:\ProgramData\IBM\MQ>type AMQCLCHL.TAB
AMQR          TLSPRQM1.SVRCONN  0  0  0
RQM1
                                2  p0  localhost(2414)
    <  Ü;
Ü; @
                                0  localhost(2414)

    ,  ö Ç
ANY_TLS12
                                0  2  2
    Ü; Ü;  0  0  0  ibmmqarnold  0
                                c].]
C:\ProgramData\IBM\MQ>

```

Check the MQSSLKEYR environment variable

```

C:\ProgramData\IBM\MQ>echo %MQSSLKEYR%
c:\program files\ibm\mq\arnold
C:\ProgramData\IBM\MQ>

```

Clear MQSERVER environment variable

```

C:\ProgramData\IBM\MQ>set MQSERVER=
C:\ProgramData\IBM\MQ>echo %MQSERVER%
%MQSERVER%
C:\ProgramData\IBM\MQ>

```

runmqsc -c TLSPRQM1

If you get any of the errors below check the logs C:\ProgramData\IBM\MQ\qmgrs\TLSPRQM1\errors

Goto the debug section in the document for help.

```

C:\ProgramData\IBM\MQ>runmqsc -c TLSPRQM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager TLSPRQM1.

AMQ9709E: SSL/TLS initialization error.

```

```

C:\temp>runmqsc -c TLSPRQM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager TLSPRQM1.

AMQ8135E: Not authorized.

```






You should get

```
C:\temp>runmqsc -c TLSPRQM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager TLSPRQM1.





dis QL(*)
  1 : dis QL(*)
AMQ8409I: Display Queue details.
      QUEUE(AMQ.5DF9F6E423C0A402)          TYPE(QLOCAL)
AMQ8409I: Display Queue details.
```

SUCCESS!!!!

SERVERside TLS files – end of exercise

Windows (C:) > ProgramData > IBM > MQ > qmgrs > TLSPRQM1 > ssl			
<input type="checkbox"/> Name	Date modified	Type	
 TLSPRQM1	18/12/2019 7:12 PM	Security Certificate	
 TLSPRQM1.kdb	18/12/2019 7:02 PM	KDB File	
 TLSPRQM1.rdb	18/12/2019 7:02 PM	RDB File	
 TLSPRQM1.sth	18/12/2019 6:41 PM	STH File	

CLIENTside TLS files – end of exercise

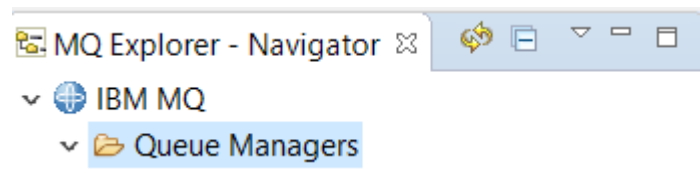
Windows (C:) > Program Files > IBM > MQ			
<input type="checkbox"/> Name	Date modified	Type	
 arnold	18/12/2019 5:03 PM	Security Certificate	
<input checked="" type="checkbox"/>  arnold.kdb	18/12/2019 7:28 PM	KDB File	
 arnold.rdb	18/12/2019 7:28 PM	RDB File	
 arnold.sth	18/12/2019 4:34 PM	STH File	

Step 12: Client: connect MQ Explore to queue manager via “Add remote Queue Manager”

What the blazes! .JKS

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.1.0/com.ibm.mq.adm.doc/q020430.htm

When you get to the SSL certificate repository details MQExplorer wants .JKS not .KDB



Add remote queue manager

Queue manager name:	<input type="text" value="TLSPRQM1"/>
How do you want to connect to this queue manager?	
<input checked="" type="radio"/> Connect directly	
This option creates a new connection to the queue manager (recommended)	

Queue manager name:	<input type="text" value="TLSPRQM1"/>
Connection details	
Host name or IP address:	<input type="text" value="localhost"/>
Port number:	<input type="text" value="2414"/>
Server-connection channel:	<input type="text" value="TLSPRQM1 SVRCONN"/>

Specify SSL certificate key repository details

Provide the location and password of a trusted store and optionally a personal certificate store

Queue manager name:

☒ Enable SSL key repositories

Trusted Certificate Store

Store name:

Password:

Personal Certificate Store

Store name:

Password:

Creating the custom MQ Image Layer for ICP4i on RHOS 4.2

Run docker images

```
C:\Users\DAVIDARNOLD\myDocker>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
davexacom/ibm-mqadvanced-server-ivt	9.1.3.0-r4-amd64	e269ad126613	3 days ago	932MB
ace11002mqc91intms1	1.0	ec39bb5df84f	8 days ago	1.59GB
ibmcom/mq	latest	268baf40303f	13 days ago	927MB
ibmcom/ace-mq	latest	0f5a883d8a95	5 weeks ago	2.51GB
ibmcom/ace-mqclient	latest	d71cee6dfd5f	5 weeks ago	1.94GB
ibmcom/ace	latest	d33c5a966d7b	5 weeks ago	1.63GB
ibm-mqadvanced-server-integration	9.1.3.0-r4-amd64	ee41039b9552	6 weeks ago	932MB

If you do not have an image for the IBM published ibm-mqadvanced-server-integration 9.1.3.0-r4-amd64 see the instructions in the RHOS 3.11 section for loading the image into the local repository

TLS enabled custom image layer – Build Files

```
md ibm-mqadvanced-server-tls
```

```
cd ibm-mqadvanced-server-tls
```

```
C:\Users\DAVIDARNOLD\myDocker>md ibm-mqadvanced-server-tls

C:\Users\DAVIDARNOLD\myDocker>cd ibm-mqadvanced-server-tls

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-tls>
```

Windows (C:) > Users > DAVIDARNOLD > myDocker > ibm-mqadvanced-server-tls			
<input type="checkbox"/>	Name	Date modified	Type
	dockerfile	20/12/2019 12:24 PM	File
	NOTLS.mqsc	20/12/2019 9:51 AM	MQSC File
	TLSPRQM1	18/12/2019 7:12 PM	Security Certificate
	TLSPRQM1.kdb	18/12/2019 7:02 PM	KDB File
	TLSPRQM1.mqsc	20/12/2019 10:48 AM	MQSC File
	TLSPRQM1.rdb	18/12/2019 7:02 PM	RDB File
	TLSPRQM1.sth	18/12/2019 6:41 PM	STH File

Docker file – run mqsc scripts and copy .KDB file to image

Notepad++ dockerfile

```
FROM ibm-mqadvanced-server-integration:9.1.3.0-r4-amd64
```

```
USER mqm
```

```
COPY TLSPRQM1.mqsc /etc/mqm/
```

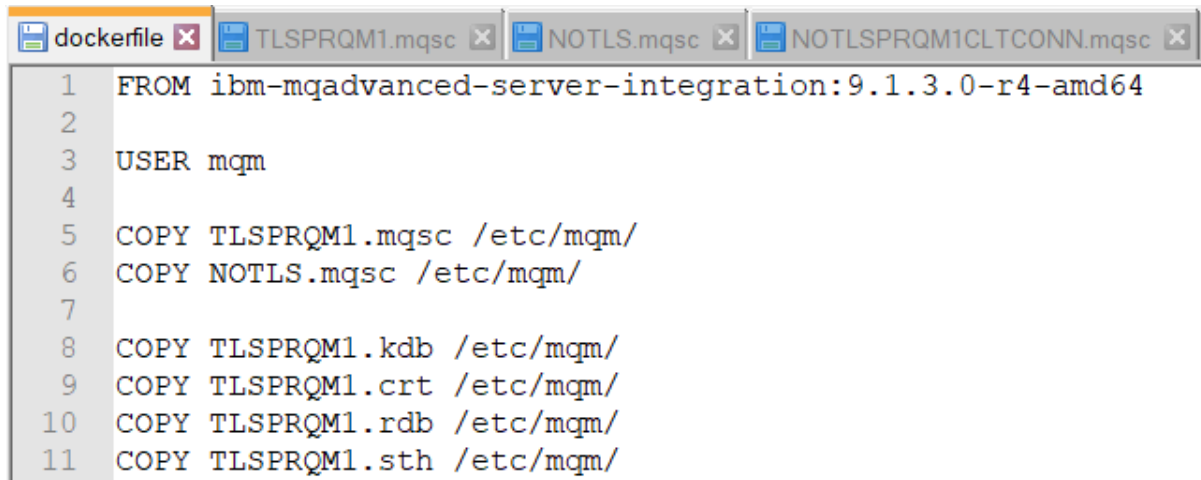
```
COPY NOTLS.mqsc /etc/mqm/
```

```
COPY TLSPRQM1.kdb /run/runmqserver/tls/
```

COPY TLSPRQM1.crt /etc/mqm/

COPY TLSPRQM1.rdb /etc/mqm/

COPY TLSPRQM1.sth /etc/mqm/



```
1 FROM ibm-mqadvanced-server-integration:9.1.3.0-r4-amd64
2
3 USER mqm
4
5 COPY TLSPRQM1.mqsc /etc/mqm/
6 COPY NOTLS.mqsc /etc/mqm/
7
8 COPY TLSPRQM1.kdb /etc/mqm/
9 COPY TLSPRQM1.crt /etc/mqm/
10 COPY TLSPRQM1.rdb /etc/mqm/
11 COPY TLSPRQM1.sth /etc/mqm/
```

TLSPRQM1.mqsc – enable TLS clients to connect
notes

MUSR_MQADMIN is windows version of mqm user in linux so should be changed for linux container deployment

The supplied base container sets this ('/run/runmqserver/tls/key') to target key.kdb

We will ALTER QMGR SSLKEYR to target TLSPRQM1.kdb

if you copy in the TLSPRQM1.kdb (and other key files) to the /run/runmqserver/tls directory this happens at start up so put it elsewhere /etc/mqm for example

*2019-12-19T23:42:50.123Z Failed to create PKCS#12 TrustStore: error running
"/opt/mqm/bin/runmqakm -keydb -create": /opt/mqm/bin/runmqakm: exit status 1 CTGSK3026W
The key file "/run/runmqserver/tls/trust.p12" does not exist or cannot be read.*

The supplied image does ALTER QMGR CERTLABL('')

We need to set CERTLABL('ibmmqtlspqm1')

The MSQC file:

DEFINE CHANNEL(TLSPRQM1.SVRCONN) CHLTYPE(SVRCONN) SSLCIPH(ANY_TLS12) REPLACE

SET CHLAUTH(TLSPRQM1.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)

ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE)
ADOPTCTX(YES)

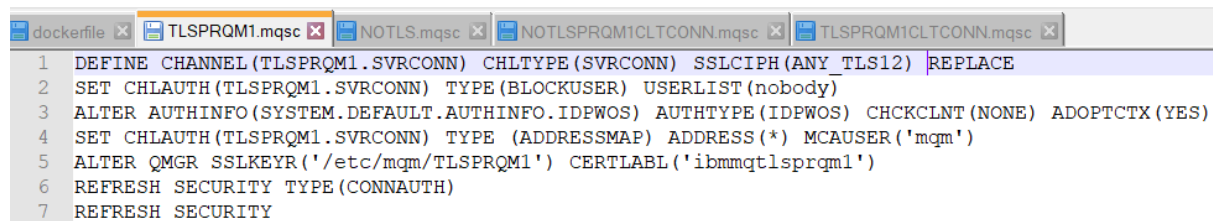
SET CHLAUTH(TLSPRQM1.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')

ALTER QMGR SSLKEYR('/etc/mqm/TLSPRQM1') CERTLABL('ibmmqtlspqm1')

REFRESH SECURITY TYPE(CONNAUTH)

REFRESH SECURITY

Notepad++ TLSPRQM1.mqsc



```
1 DEFINE CHANNEL(TLSPRQM1.SVRCONN) CHLTYPE(SVRCONN) SSLCIPH(ANY TLS12) REPLACE
2 SET CHLAUTH(TLSPRQM1.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)
3 ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE) ADOPTCTX(YES)
4 SET CHLAUTH(TLSPRQM1.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')
5 ALTER QMGR SSLKEYR('/etc/mqm/TLSPRQM1') CERTLABL('ibmmqtlspqm1')
6 REFRESH SECURITY TYPE(CONNAUTH)
7 REFRESH SECURITY
```

NOTLS.mqsc – add channel for in cluster clients to connect

DEFINE CHANNEL(IVT.SVRCONN) CHLTYPE(SVRCONN)

SET CHLAUTH(IVT.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)

ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE)
ADOPTCTX(YES)

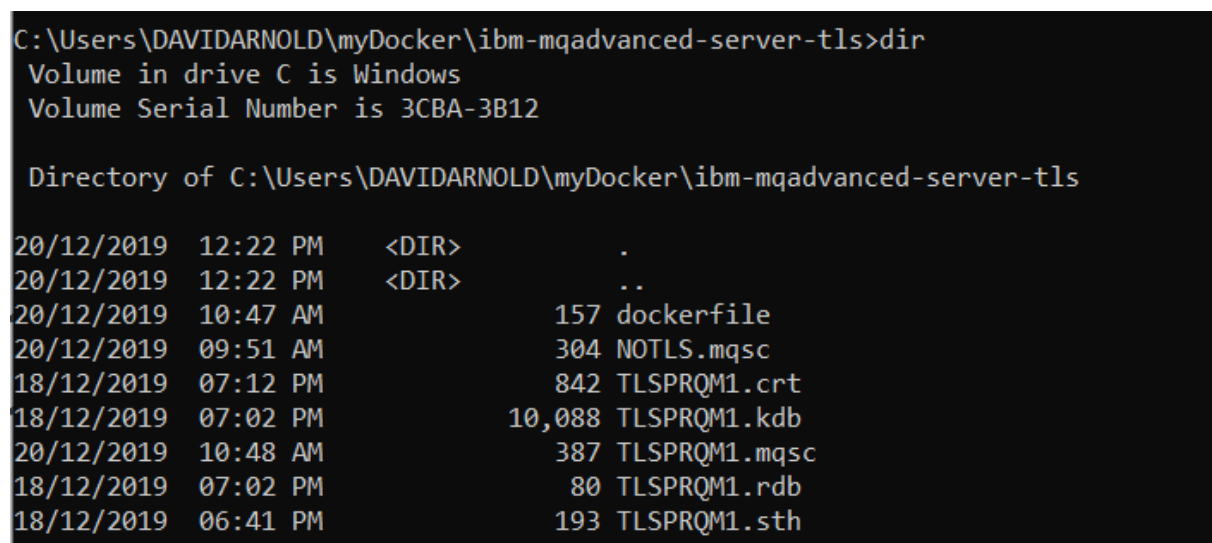
SET CHLAUTH(IVT.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')

REFRESH SECURITY TYPE(CONNAUTH)

Key database files and password stash files

The .KDB etc with server side and client side certs from the matched set of files created earlier

Docker build – Create image FROM ibm-mqadvanced-server



```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-tls>dir
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

Directory of C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-tls

20/12/2019  12:22 PM    <DIR>          .
20/12/2019  12:22 PM    <DIR>          ..
20/12/2019  10:47 AM             157 dockerfile
20/12/2019  09:51 AM             304 NOTLS.mqsc
18/12/2019  07:12 PM             842 TLSPRQM1.crt
18/12/2019  07:02 PM          10,088 TLSPRQM1.kdb
20/12/2019  10:48 AM             387 TLSPRQM1.mqsc
18/12/2019  07:02 PM              80 TLSPRQM1.rdb
18/12/2019  06:41 PM             193 TLSPRQM1.sth
```

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-tls> docker build -t davexacom/ibm-mqadvanced-server-tls:9.1.3.0-r4-amd64 .

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-tls>docker build -t davexacom/ibm-mqadvanced-server-tls:9.1.3.0-r4-amd64 .
Sending build context to Docker daemon 18.43kB
Step 1/8 : FROM ibm-mqadvanced-server-integration:9.1.3.0-r4-amd64
--> ee41039b9552
Step 2/8 : USER mqm
--> Using cache
--> 89c0bb4a3f90
Step 3/8 : COPY TLSPRQM1.mqsc /etc/mqm/
--> f4175b5e2b23
Step 4/8 : COPY NOTLS.mqsc /etc/mqm/
--> ada3d193149d
Step 5/8 : COPY TLSPRQM1.kdb /etc/mqm/
--> cfbf7115749c
Step 6/8 : COPY TLSPRQM1.crt /etc/mqm/
--> 8da14eae49e3
Step 7/8 : COPY TLSPRQM1.rdb /etc/mqm/
--> 75e9c882e224
Step 8/8 : COPY TLSPRQM1.sth /etc/mqm/
--> 0d0d475be96d
Successfully built 0d0d475be96d
Successfully tagged davexacom/ibm-mqadvanced-server-tls:9.1.3.0-r4-amd64
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to
build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and d
irectories.
```

Perform Local testing on docker workstation

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-tls>docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
davexacom/ibm-mqadvanced-server-tls	9.1.3.0-r4-amd64	f34a2f062408	2 minutes ago	932MB

Start the container – docker run

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-tls>docker run -e LICENSE=accept -e MQ_QMGR_NAME=TLSPRQM1 -P davexacom/ibm-mqadvanced-server-tls:9.1.3.0-r4-amd64

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-tls>docker run -e LICENSE=accept -e MQ_QMGR_NAME=TLSPRQM1 -P davexacom/ibm-mqadvanced-server-tls:9.1.3.0-r4-amd64
2019-12-19T06:30:19.945Z CPU architecture: amd64
2019-12-19T06:30:19.945Z Linux kernel version: 4.9.184-linuxkit
2019-12-19T06:30:19.945Z Container runtime: docker
2019-12-19T06:30:19.945Z Base image: Red Hat Enterprise Linux Server 7.7 (Maipo)
2019-12-19T06:30:19.947Z Running as user ID 888 () with primary group 888, and supplementary groups 0
2019-12-19T06:30:19.947Z Capabilities (bounding set): chown,dac_override,fowner,fsetid,kill,setgid,setuid,setpcap,net_bind_service,net_raw,sys_chroot,mknod,audit_write,setfcap
2019-12-19T06:30:19.947Z seccomp enforcing mode: filtering
2019-12-19T06:30:19.947Z Process security attributes: none
2019-12-19T06:30:19.948Z No volume detected. Persistent messages may be lost
2019-12-19T06:30:19.971Z Open Tracing is disabled
2019-12-19T06:30:19.979Z Using queue manager name: TLSPRQM1
2019-12-19T06:30:19.984Z Created directory structure under /var/mqm
2019-12-19T06:30:19.984Z Image created: 2019-11-06T12:02:58+0000
2019-12-19T06:30:19.984Z Image tag: ibm-mqadvanced-server-integration:9.1.3.0-r4-amd64
2019-12-19T06:30:19.994Z MQ version: 9.1.3.0
2019-12-19T06:30:19.995Z MQ level: p913-L190628_IT30047_P
2019-12-19T06:30:19.995Z MQ license: Production
2019-12-19T06:30:20.398Z Creating queue manager TLSPRQM1
```

Break

note that the default container image sets an SSLKEYR ssl key repository of /run/runmqserver/tls/key which means it expects a key.kdb file in that location to house the key database.

```
      : * Set the keystore location for the queue manager
1 : ALTER QMGR SSLKEYR('/run/runmqserver/tls/key')
AMQ8005I: IBM MQ queue manager changed.
2 : ALTER QMGR CERTLABL('')
AMQ8005I: IBM MQ queue manager changed.
3 : REFRESH SECURITY(*) TYPE(SSL)
AMQ8560I: IBM MQ security cache refreshed.
3 MQSC commands read.
```

Here are our TLS channel definitions being run

Starting MQSC for queue manager TLSPRQM1.

```
1 : DEFINE CHANNEL(TLSPRQM1.SVRCONN) CHLTYPE(SVRCONN) SSLCIPH(ANY_TLS12) REPLACE
AMQ8014I: IBM MQ channel created.
2 : SET CHLAUTH(TLSPRQM1.SVRCONN) TYPE(BLOCKUSER) USERLIST(nobody)
AMQ8877I: IBM MQ channel authentication record set.
3 : ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(NONE) ADOPTCTX(YES)
AMQ8567I: IBM MQ authentication information changed.
4 : SET CHLAUTH(TLSPRQM1.SVRCONN) TYPE (ADDRESSMAP) ADDRESS(*) MCAUSER('mqm')
AMQ8877I: IBM MQ channel authentication record set.
5 : ALTER QMGR SSLKEYR('/etc/mqm/TLSPRQM1') CERTLABL('ibmmqtlsprqm1')
AMQ8005I: IBM MQ queue manager changed.
6 : REFRESH SECURITY TYPE(CONNAUTH)
AMQ8560I: IBM MQ security cache refreshed.
7 : REFRESH SECURITY
AMQ8560I: IBM MQ security cache refreshed.
7 MQSC commands read.
```

```
2019-12-19T06:30:21.648Z Metrics are disabled
2019-12-19T06:30:21.471Z AMQ5051I: The queue manager task 'LOGGER-IO' has started.
2019-12-19T06:30:21.477Z AMQ7229I: 5 log records accessed on queue manager 'TLSPRQM1' during the log replay phase.
2019-12-19T06:30:21.477Z AMQ7230I: Log replay for queue manager 'TLSPRQM1' complete.
2019-12-19T06:30:21.478Z AMQ5051I: The queue manager task 'CHECKPOINT' has started.
2019-12-19T06:30:21.480Z AMQ7231I: 0 log records accessed on queue manager 'TLSPRQM1' during the recovery phase.
2019-12-19T06:30:21.480Z AMQ7232I: Transaction manager state recovered for queue manager 'TLSPRQM1'.
2019-12-19T06:30:21.488Z AMQ7233I: 0 out of 0 in-flight transactions resolved for queue manager 'TLSPRQM1'.
```

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-tls>docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS		NAMES
8a10fb9133ac	davexacom/ibm-mqadvanced-server-tls:9.1.3.0-r4-amd64	"runmqintegrationser..."	41 seconds ago
Up 40 seconds	0.0.0.0:32782->1414/tcp, 0.0.0.0:32781->9157/tcp, 0.0.0.0:32780->9443/tcp		lucid_haibt

Ports exposed 0.0.0.0:32782->1414/tcp, 0.0.0.0:32781->9157/tcp, 0.0.0.0:32780->9443/tcp

[Connect MQ Explorer](#)

Add remote queue manager



Add Queue Manager

Select the queue manager and connection method

Identify the queue manager to add and choose the connection method to use

Queue manager name:

How do you want to connect to this queue manager?



Connect directly

This option creates a new connection to the queue manager (recommended)

Add Queue Manager

Specify new connection details

Provide details of the connection you want to set up

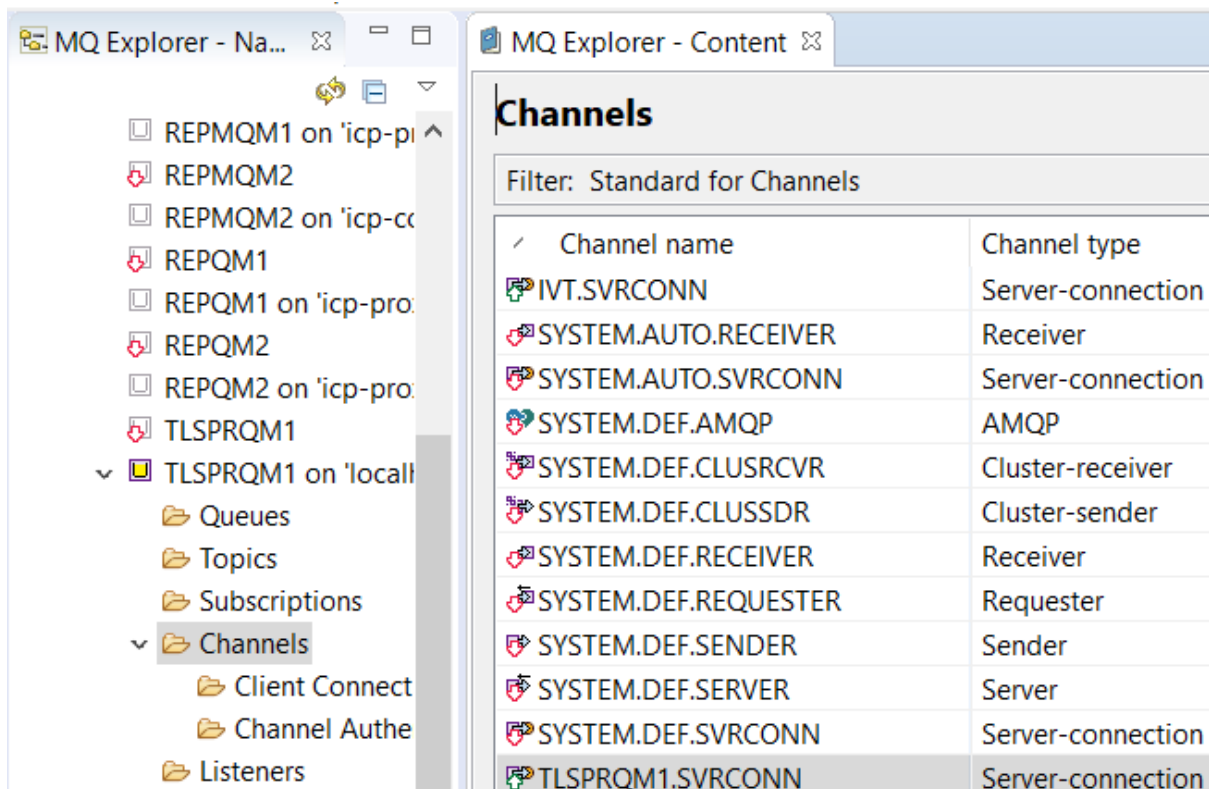
Queue manager name:	TLSPRQM1
Connection details	
Host name or IP address:	localhost
Port number:	32782
Server-connection channel:	IVT.SVRCONN

Add Queue Manager

Specify user identification details

Provide a userid name and password

Queue manager name:	TLSPRQM1
<input checked="" type="checkbox"/> Enable user identification	
<input type="checkbox"/> User identification compatibility mode	
Userid:	mqm
Password	
<input checked="" type="radio"/> No password	



Connect RUNMQSC via Non TLS client

Set MQSERVER=IVT.SVRCONN/TCP/localhost(32782)

```
C:\temp>Set MQSERVER=IVT.SVRCONN/TCP/localhost(32782)

C:\temp>runmqsc -c TLSPRQM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager TLSPRQM1.

dis ql(*)
1 : dis ql(*)
AMQ8409I: Display Queue details.
      QUEUE(AMQ.5DFC2484257A2902)                TYPE(QLOCAL)
AMQ8409I: Display Queue details
```

Connect RUNMQSC via TLS client

Set up the Client Channel Definition Table

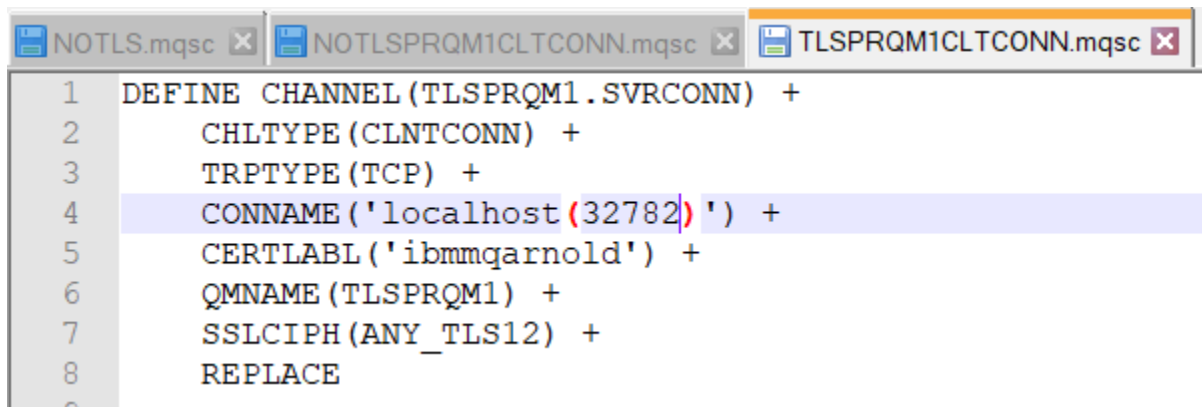
Create an client connection channel MQSC file

Make sure you correctly set the port number

DEFINE CHANNEL(TLSPRQM1.SVRCONN) +

CHLTYPE(CLNTCONN) +

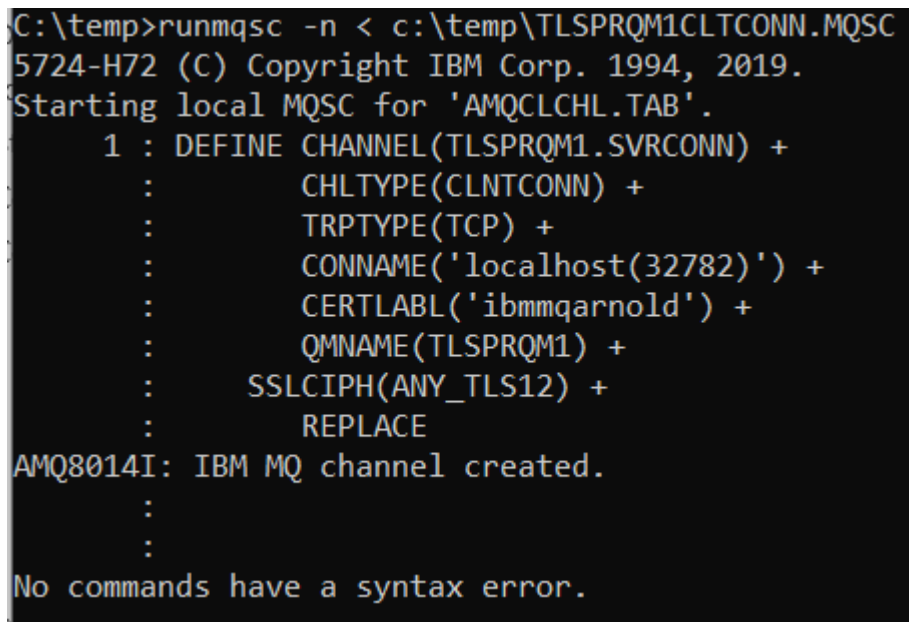
```
TRPTYPE(TCP) +  
CONNNAME('localhost(32782)') +  
CERTLABL('ibmmqarnold') +  
QMNAME(TLSPRQM1) +  
SSLCIPH(ANY_TLS12) +  
REPLACE
```



```
NOTLS.mqsc x NOTLSPRQM1CLTCONN.mqsc x TLSPRQM1CLTCONN.mqsc x  
1 DEFINE CHANNEL(TLSPRQM1.SVRCONN) +  
2   CHLTYPE(CLNTCONN) +  
3   TRPTYPE(TCP) +  
4   CONNNAME('localhost(32782)') +  
5   CERTLABL('ibmmqarnold') +  
6   QMNAME(TLSPRQM1) +  
7   SSLCIPH(ANY_TLS12) +  
8   REPLACE
```

Runmqsc with the -n (no queue manager switch) to create a CCDT from the MQSC script

runmqsc -n < c:\temp\TLSPRQM1CLTCONN.MQSC



```
C:\temp>runmqsc -n < c:\temp\TLSPRQM1CLTCONN.MQSC  
5724-H72 (C) Copyright IBM Corp. 1994, 2019.  
Starting local MQSC for 'AMQCLCHL.TAB'.  
1 : DEFINE CHANNEL(TLSPRQM1.SVRCONN) +  
  :   CHLTYPE(CLNTCONN) +  
  :   TRPTYPE(TCP) +  
  :   CONNNAME('localhost(32782)') +  
  :   CERTLABL('ibmmqarnold') +  
  :   QMNAME(TLSPRQM1) +  
  :   SSLCIPH(ANY_TLS12) +  
  :   REPLACE  
AMQ8014I: IBM MQ channel created.  
  :  
  :  
No commands have a syntax error.
```

Optionally set the CCDT environment variables

If the default CCDT TAB name and default CCDT TAB directory are not used, we will need to set up the environment variable for path to AMQCLCHL.TAB

Because I have used all defaults the MQClient code will find and use the default AMQCLCHL.TAB

dir c:\programdata\ibm\mq*.TAB

```
C:\temp>dir C:\ProgramData\IBM\MQ\*.TAB
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

Directory of C:\ProgramData\IBM\MQ

20/12/2019  11:08 AM                2,106 AMQCLCHL.TAB
              1 File(s)                2,106 bytes
              0 Dir(s)  215,347,277,824 bytes free
```

This is the default but for reference purposes here's how to set the environment variable.

On Linux

```
export MQCHLLIB=/var/mqm/AMQCLCHL.TAB
```

```
echo $MQCHLLIB
```

On windows

```
set MQCHLLIB=C:\ProgramData\IBM\MQ\AMQCLCHL.TAB
```

```
echo %MQCHLLIB%
```

```
set MQCHLTAB= AMQCLCHL.TAB
```

```
echo %MQCHLTAB%
```

Eyeball the .TAB file – check for host and port number

```
type c:\programdata\ibm\mq\AMQCLCHL.TAB
```

```
C:\temp>type c:\programdata\ibm\mq\AMQCLCHL.TAB
AMQR          TLSPRQM1.SVRCONN      @  @  @
RQM1
                                           localhost(32782)
2      p@
<  Ü;@
Ü; @
      localhost(32782)
00 ,@  @  ö Ç
ANY_TLS12
      2  2
Ü; Ü;  @  ê@  @  ibmmqarnold
      ]%"]
C:\temp>
```

Clear the MQSERVER environment variable so the CCDT is used

SET MQSERVER=

ECHO %MQSERVER%

Set the MQSSLKEYR environment variable for the client side
set MQSSLKEYR=C:\Program Files\IBM\MQ\arnold

```
C:\temp>set MQSSLKEYR=C:\Program Files\IBM\MQ\arnold

C:\temp>echo %MQSSLKEYR%
C:\Program Files\IBM\MQ\arnold

C:\temp>dir C:\"Program Files"\IBM\MQ\arnold.kdb
Volume in drive C is Windows
Volume Serial Number is 3CBA-3B12

Directory of C:\Program Files\IBM\MQ

18/12/2019  07:28 PM                10,088 arnold.kdb
               1 File(s)                10,088 bytes
               0 Dir(s)  215,355,838,464 bytes free
```

Double check MQSERVER is clear

Echo %MQSERVER%

```
C:\ProgramData\IBM\MQ>set MQSERVER=

C:\ProgramData\IBM\MQ>echo %MQSERVER%
%MQSERVER%

C:\ProgramData\IBM\MQ>
```

runmqsc -c TLSPRQM1

```
C:\temp>runmqsc -c TLSPRQM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager TLSPRQM1.
```

```
dis chl(TLSPRQM1.SVRCONN)
  1 : dis chl(TLSPRQM1.SVRCONN)
AMQ8414I: Display Channel details.
CHANNEL(TLSPRQM1.SVRCONN)          CHLTYPE(SVRCONN)
ALTDATE(2019-12-20)                ALTTIME(01.59.34)
CERTLABL( )                        COMPHDR(NONE)
COMPMSG(NONE)                      DESCR( )
DISCINT(0)                         HBINT(300)
KAINT(AUTO)                        MAXINST(999999999)
MAXINSTC(999999999)               MAXMSGL(4194304)
MCAUSER( )                         MONCHL(QMGR)
RCVDATA( )                         RCVEXIT( )
SCYDATA( )                         SCYEXIT( )
SENDDATA( )                       SENDEXIT( )
SHARECNV(10)                      SSLCAUTH(REQUIRED)
SSLCIPH(ANY_TLS12)                SSLPEER( )
TRPTYPE(TCP)
```

You are now connected to TLSPRQM1 in your container via TLS

Testing TLS custom MQ Image Layer on RHOS 3.11

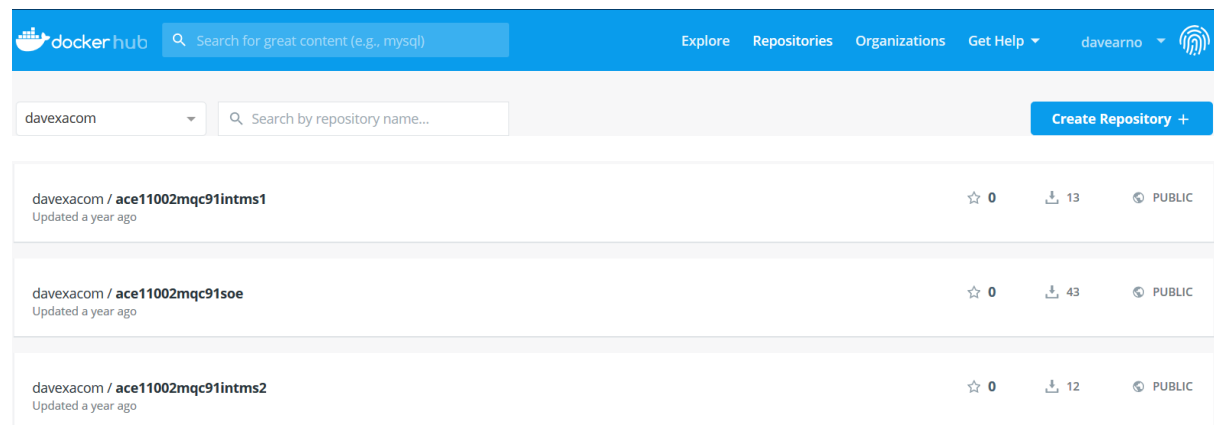
Pushing the custom image to docker hub

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-tls>docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
davexacom/ibm-mqadvanced-server-tls	9.1.3.0-r4-amd64	2237fea38967	17 minutes ago	932MB

```
C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker images
REPOSITORY          TAG                IMAGE ID           CREATED            SIZE
davexacom/ibm-mqadvanced-server-ivt  9.1.3.0-r4-amd64  e269ad126613      5 minutes ago     932MB
```

<https://hub.docker.com/repositories>



docker push davexacom/ibm-mqadvanced-server-tls:9.1.3.0-r4-amd64

C:\Users\DAVIDARNOLD\myDocker\ibm-mqadvanced-server-ivt>docker push davexacom/ibm-mqadvanced-server-tls:9.1.3.0-r4-amd64

The push refers to repository [docker.io/davexacom/ibm-mqadvanced-server-tls]

0d8459f27ecc: Pushing [=====>] 3.072kB

ac7598f58291: Pushing [=====>] 1.378MB/3.735MB

6eb6a773bf24: Pushing [=====>] 1.378MB/3.735MB

07cdf8094239: Pushing [=====>] 561.2kB/3.49MB

9f4789d1e0ed: Pushing [=====>] 3.072kB

364112d1df0e: Waiting

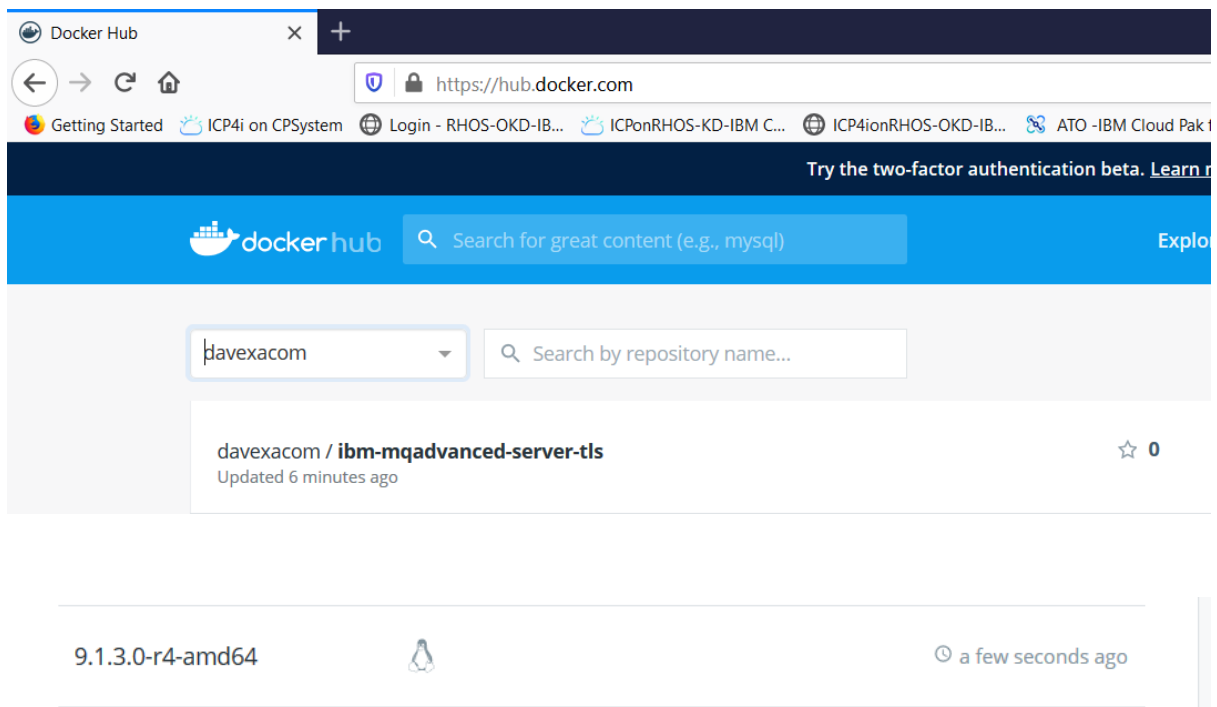
cd28e8b3d42a: Pushed

2705f79af6db: Layer already exists

ce459cc53899: Layer already exists 9.1.3.0-r4-

amd64: digest: sha256:040e8e0aca3307369b10cce8411f97e147a9a92bb7514619a16c2a78093c085f
size: 5963

<https://hub.docker.com/repository/docker/davexacom/ibm-mqadvanced-server-tls>



Create a new Image Stream on RHOS 3.11

<https://registry-console-default.apps.ocp.cloudnativekube.com/registry#/images/mq>

Images			New image stream
Name	Tags	Repository	
mq/ibm-mq-oidc-registration	2.1.0-amd64	docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration	
mq/ibm-mqadvanced-server-integration	9.1.3.0-r1-amd64	docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-integration	

[New image stream](#)

Repository

Docker.io/davexacom/ibm-mqadvanced-server-tls

Create new image stream

Create image stream

Name

ibm-mqadvanced-server-tls

Project

mq

Populate

Pull specific tags from another image repository

Pull from

docker.io/davexacom/ibm-mqadvanced-server-tls

Tags

9.1.3.0-r4-amd64

☐ Remote registry is insecure

Cancel

Create

mq/ibm-mqadvanced-server-tls

Show all image streams

Image Stream

Access Policy

Images may only be pulled by specific users or groups

Pulling repository

docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-tls

Image count

1

To push an image to this image stream:

```
$ sudo docker tag myimage docker-registry-default.apps.ocp.cloudnativekubernetes.com/mq/ibm-mqadvanced-server-tls:tag
$ sudo docker push docker-registry-default.apps.ocp.cloudnativekubernetes.com/mq/ibm-mqadvanced-server-tls:tag
```

Images

Tag	From
9.1.3.0-r4...	docker.io/davexacom/ibm-mqadvanced-server-tls:9.1.3.0-r4-amd64

Pulling repository to use in helm charts on 3.11

docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-tls

Add new ICP4i instance of MQ custom TLS layer on RHOS 3.11

<https://icp-proxy.apps.ocp.cloudnativekubernetes.com/integration>



idcab1

mqtest

[+ Add new instance](#)

You'll be deploying [IBM MQ](#) to provide this capability. Please work with your IBM Cloud Private cluster administrator to ensure the following dependencies are satisfied:

- a unique [namespace](#) must exist for the sole use of IBM MQ. Multiple instances can be deployed in the same namespace. This namespace must allow deployments that require a [security context constraint](#) of type : **ibm-anyuid-scc**.
- a [storage class](#) or [persistent volume](#) needs to be provided for persistent storage


Close

Continue

Configure Helm chart ibm-mqadvanced-server-integration-prod V 4.1.0

Overview

Configuration

 IBM Certified Container


IBM MQ queue manager for Cloud Pak for Integration

ibm-entitled-charts

[Licenses](#) | [Release Notes](#) | [Qualification](#)

IBM Certified Container

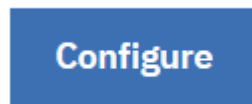
4.1.0

 IBM MQ Advanced

Introduction

This chart deploys a single IBM® MQ version 9.1.3 middleware that simplifies and accelerates the integration of multiple platforms. It uses message queues to facilitate messaging solution for cloud, mobile, Internet of Things.

Hit configure



Values for the charts

Cluster Hostname (for deployment)

icp-proxy.apps.ocp.cloudnativekube.com

MQ server image:

docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-tls

MQ Server image Tag:

9.1.3.0-r4

MQ Server OIDC registration image

docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration

MQ Server OIDC registration image tag:

2.1.0

Managed NFS name (from oc command):

managed-nfs-storage

TLS Secret

ibm-mq-tls-secret

ibm-mqadvanced-server-integration-prod V 4.1.0

Overview

Configuration

Pod Security Warning Your ICP cluster is running all namespaces Unrestricted (ibm-anyuid-hostpath-psp) by default. This could pose a security risk.

Configuration

IBM MQ queue manager for Cloud Pak for Integration. Edit these parameters for configuration.

Helm release name *

tlsprqm1p

Target namespace *

mq

Target Cluster *

local-cluster

License *ⁱ

☒ I have read and agreed to the [License agreement](#)

icp-proxy.apps.ocp.cloudnativekube.com



Quick start

Required and recommended parameters to view and edit.

TLS

Configuration settings for TLS

Cluster hostname *ⁱ

icp-proxy.apps.ocp.cloudnativekube.com|

MQ server image:


docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-tls

MQ Server image Tag:

9.1.3.0-r4

Results in the following image and tag being used

docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-tls:9.1.3.0-r4-amd64

Image	
Configuration settings for the container image	
Image repository *	Image tag * 
docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-tls	9.1.3.0-r4
Image pull secret	Image pull policy (for all images) *
Enter value	IfNotPresent

MQ Server OIDC registration image

docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration


MQ Server OIDC registration image tag:

2.1.0

Single sign-on	
Configuration settings for single sign-on	
Registration image repository *	Registration image tag *
docker-registry.default.svc:5000/mq/ibm-mq-oidc-registration	2.1.0

TLS Secret

ibm-mq-tls-secret

TLS	
Configuration settings for TLS	
<input checked="" type="checkbox"/> Generate Certificate	
Cluster hostname *	Secret name 
icp-proxy.apps.ocp.cloudnativekube.com	Ibm-mq-tls-secret

Managed NFS name (from oc command):

managed-nfs-storage

Persistence

Configuration settings for Persistent Volumes

☒ **Enable persistence ***

☒ **Use dynamic provisioning ***

Data PVC

Configuration settings for the main Persistent Volume Claim

Name *

data

Storage Class name 

managed-nfs-storage

Size *

2Gi

Queue manager


Configuration settings for the Queue Manager

Queue manager name 

TLSPRQM1

Metrics

Configuration settings for Prometheus metrics

☐ **Enable metrics *** 

Readiness probe

Configuration settings for the MQ readiness probe, which checks when the MQ listener is running

Initial delay (seconds)

30



Install

Use the little X to retain your populated chart for re-use if things go wrong.

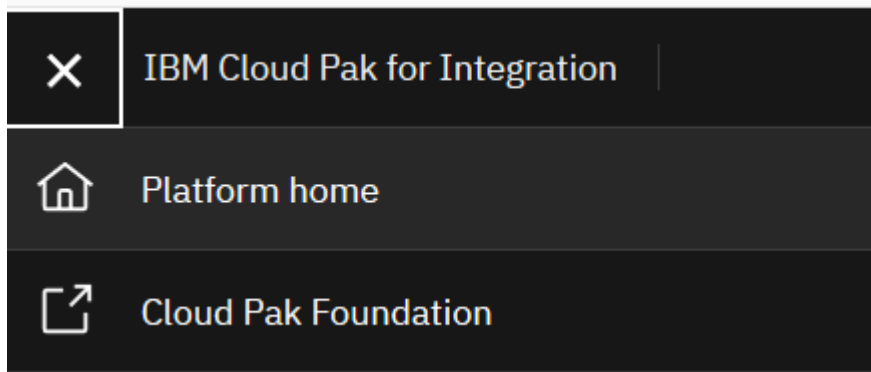


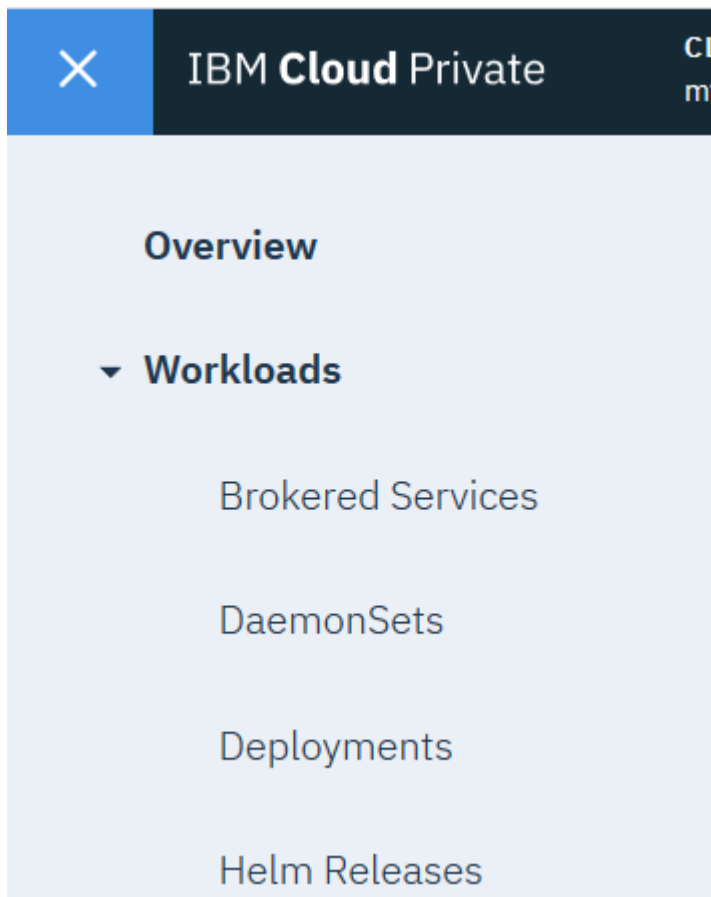
Installation started. For progress view your Helm releases.

View Helm Releases

Return to [Catalog](#)

Verifying the Helm release via ICP foundation





Helm Releases

<div><div></div><div>You are currently viewing only the helm releases of this cluster.</div></div>				
<div><div><div></div><div>tls</div><div></div></div></div>				
Name ^	Namespace	Status	Chart name	Current version
tlsprqm1p	mq	<div></div> Deployed	ibm-mqadvanced-server-integration-prod	4.1.0

Details and Upgrades

CHART NAME

ibm-mqadvanced-server-integration-prod

NAMESPACE

mq

CURRENT VERSION

4.1.0

Installed: December 20, 2019
→ [Release Notes](#)

AVAILABLE VERSION

5.0.0 ▲

Released: November 29, 2019
→ [Release Notes](#)

Job

Name	COMPLETIONS
tlsprqm1p-ibm-mq-registration	1/1

Service

Name	TYPE	Cluster IP	External IP	Port(s)
tlsprqm1p-ibm-mq	NodePort	172.30.190.198	<none>	9443:30218/TCP,1414:30182/TCP

StatefulSets / [tlsprqm1p-ibm-mq](#)

tlsprqm1p-ibm-mq

Overview

[Events](#)

Q Search Events

Type	Source	Count	Reason	Message
Normal	statefulset-controller	1	SuccessfulCreate	create Claim data-tlsprqm1p-ibm-mq-0 Pod tlsprqm1p-ibm-mq-0 in StatefulSet tlsprqm1p-ibm-mq success
Normal	statefulset-controller	1	SuccessfulCreate	create Pod tlsprqm1p-ibm-mq-0 in StatefulSet tlsprqm1p-ibm-mq successful

StatefulSets / [tlsprqm1p-ibm-mq](#) / [tlsprqm1p-ibm-mq-0](#)

tlsprqm1p-ibm-mq-0

Overview

[Containers](#)

Events

Name	Image	Port	State
qmgr	docker-registry.default.svc:5000/mq/ibm-mqadvanced-server-tls:9.1.3.0-r4-amd64	1414/TCP,9443/TCP,9157/TCP	Running

Configure client side CCDT for TLSPRQM1 on RHOS 3.11

Goto the Helm release's service definition

Service				
Name	TYPE	Cluster IP	External IP	Port(s)
tlsprqm1p-ibm-mq	NodePort	172.30.237.192	<none>	9443:32527/TCP,1414:32602/TCP

Click on qmgr 32602/TCP

Node port

console-https 32527/TCP

qmgr 32602/TCP

Copy the URL from the web browser link

icp-console.apps.ocp.cloudnativekube.com:32602

Update the MQSC file that builds the CCDT

```
dockerfile x TLSPRQM1.mqsc x NOTLS.mqsc x NOTLSPRQM1CLTCONN.mqsc x TLSPRQM1CLTCONN.mqsc x
1 DEFINE CHANNEL(TLSPRQM1.SVRCONN) +
2   CHLTYPE(CLNTCONN) +
3   TRPTYPE(TCP) +
4   CONNAME('icp-console.apps.ocp.cloudnativekube.com(32602)') +
5   CERTLABL('ibmmqarnold') +
6   QMNAME(TLSPRQM1) +
7   SSLCIPH(ANY_TLS12) +
8   REPLACE
```

runmqsc -n < c:\temp\TLSPRQM1CLTCONN.MQSC

```
C:\temp>runmqsc -n < c:\temp\TLSPRQM1CLTCONN.MQSC
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting local MQSC for 'AMQCLCHL.TAB'.
1 : DEFINE CHANNEL(TLSPRQM1.SVRCONN) +
:   CHLTYPE(CLNTCONN) +
:   TRPTYPE(TCP) +
:   CONNAME('icp-console.apps.ocp.cloudnativekube.com(32602)') +
:   CERTLABL('ibmmqarnold') +
:   QMNAME(TLSPRQM1) +
:   SSLCIPH(ANY_TLS12) +
:   REPLACE
AMQ8014I: IBM MQ channel created.
:
:
No commands have a syntax error.
```

Check the MQSERVER, MQSSLKEYR and MQCHLLIB MQCHLTAB environment variables are correct

MQSERVER,MQCHLLIB and MQCHLTAB can all be unset if defaults are being used.

MQSSLKEYR should have been set from the local docker testing

C:\temp\SET

```
LOGONSERVER=\\DESKTOP-2MFRK17
MQSSLKEYR=C:\Program Files\IBM\MQ\arnold
MQ_FILE_PATH=C:\Program Files\IBM\MQ
```

Starting the MQ Web Console – from the service link

<https://icp-console.apps.ocp.cloudnativekube.com:32527/>

Add widgets->Queues

Create a new queue called MYQ

Create a Queue

Queue name: *

Queue type:

☒ Local ☐ Remote ☐ Alias ☐ Model

C:\temp\runmqsc -c TLSPQM1

```
C:\temp>runmqsc -c TLSPQM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager TLSPQM1.

dis chl(TLSPQM1.SVRCONN)
  1 : dis chl(TLSPQM1.SVRCONN)
AMQ8414I: Display Channel details.
      CHANNEL(TLSPQM1.SVRCONN)          CHLTYPE(SVRCONN)
      ALTDATE(2019-12-20)                ALTTIME(02.50.41)
      CERTLABL( )                       COMPHDR(NONE)
      COMPMSG(NONE)                     DESCR( )
      DISCINT(0)                        HBINT(300)
      KAUTN(AUTO)                       MAXINST(999999999)
      MAXINSTC(999999999)               MAXMSGL(4194304)
      MCAUSER( )                       MONCHL(QMGR)
      RCVDATA( )                       RCVEXIT( )
      SCYDATA( )                       SCYEXIT( )
      SENDDATA( )                      SENDEXIT( )
      SHARECNV(10)                     SSLCAUTH(REQUIRED)
      SSLCIPH(ANY_TLS12)                SSLPEER( )
      TRPTYPE(TCP)

dis ql(MYQ)
  2 : dis ql(MYQ)
AMQ8409I: Display Queue details.
      QUEUE(MYQ)                        TYPE(LOCAL)
```

Success ! you are now TLS connected to a QMgr on RHOS 3.11

Using pre-configured collateral to connect

Custom Layer MQ Image for TLS

The custom layer MQ image with TLS enabled and the server side TLS files deployed

<https://hub.docker.com/repository/docker/davexacom/ibm-mqadvanced-server-tls>

The docker file and mq files used to create the image

<https://github.com/DAVEXACOM/Exploring-ICP4i-RHOS/tree/master/TLS%20-%20custom%20layer%20MQ%20Image/customImageLayerFiles>

Plus the server side TLS files it has deployed can be found here

<https://github.com/DAVEXACOM/Exploring-ICP4i-RHOS/tree/master/TLS%20-%20custom%20layer%20MQ%20Image/serverside%20TLS%20files>

Deploy an ICP4i MQ instance

Follow instructions from earlier in this section = "[Configure Helm Chart](#)"

CCDT

Client channel definition table and the MQSC file to create it.

AMQCLCHL.TAB from here and stick it in c:\programdata\ibm\mq (default location (windows example))

<https://github.com/DAVEXACOM/Exploring-ICP4i-RHOS/tree/master/TLS%20-%20custom%20layer%20MQ%20Image/CCDTfiles>

client side TLS files

The arnold.* client side files

in C:\Program Files\IBM\MQ (default location (windows example))

<https://github.com/DAVEXACOM/Exploring-ICP4i-RHOS/tree/master/TLS%20-%20custom%20layer%20MQ%20Image/clientside%20TLS%20files>

Set/Clear Environment Variables

Check the MQSERVER, MQSSLKEYR and MQCHLLIB MQCHLTAB environment variables are correct

MQSERVER, MQCHLLIB and MQCHLTAB can all be unset if defaults are being used.

MQSSLKEYR should have been set from the local docker testing

C:\temp\SET MQSSLKEYR=c:\Program Files\IBM\MQ\arnold

Connecting runmqsc in client mode

runmqsc -c TLSPRQM1

Testing TLS custom MQ Image Layer on RHOS 4.2

Pushing the custom image to docker hub

Appendix of useful-ish info and links

+++ Step 10: Client: Run sample client to test connection

WINDOWS: Run the C sample client "SSLSample.exe" (provided with the SupportPac MO04) on the Windows box

Download the following SupportPac. Notice that we are going to use the testing samples only, which work for 7.1 and 7.5. The GUI that is provided for the SSL Wizard function has not been updated to use the GSKit commands used by MQ 7.1 and 7.5.

<http://www-1.ibm.com/support/docview.wss?uid=swg24010367> MO04: WebSphere MQ SSL Wizard

Change the directory to the location where the SupportPac was downloaded: cd "C:\MQ-SupportPac\MO04 SSL Wizard" cd client_samples\bin

```
SSLSample.exe veracruz.x.com(1430) SSL.SVRCONN QM_71SSL NULL_SHA
"C:\Users\IBM_ADMIN\AppData\Local\VirtualStore\Program Files (x86)\IBM\WebSphere
MQ_2\rivera" Connecting to: Conname = veracruz.x.com(1430). SvrconnChannelName =
SSL.SVRCONN. QMgrName = QM_71SSL. SSLCiph = NULL_SHA. SSLKeyr =
C:\Users\IBM_ADMIN\AppData\Local\VirtualStore\Program Files (x86)\IBM\WebSphere
MQ_2\rivera. MQCONN ended with reason code 2393 Use "mqrc" to find out the short name for
the return code 2393.
```

```
C:\MQ-SupportPac\MO04 SSL Wizard\client_samples\bin> mqrc 2393
```

```
2393 0x00000959 MQRC_SSL_INITIALIZATION_ERROR
```

Page 16 of 19

+++ Step 11: Troubleshooting

UNIX: Let's take a look at the error log for the queue manager

```
cd /var/mqm/qmgrs/QM_71SSL/errors
```

```
tail AMQERR01.LOG
```

AMQ9660: SSL key repository: password stash file absent or unusable. EXPLANATION: The SSL key repository cannot be used because MQ cannot obtain a password to access it. Reasons giving rise to this error include: (a) the key database file and password stash file are not present in the location configured for the key repository, (b) the key database file exists in the correct place but that no password stash file has been created for it, (c) the files are present in the correct place but the userid under which MQ is running does not have permission to read them, (d) one or both of the files are corrupt.

ACTION: Ensure that the key repository variable is set to where the key database file is. Ensure that a password stash file has been associated with the key database file in the same directory, and that the userid under which MQ is running has read access to both files. If both are already present and readable in the correct place, delete and recreate them. Restart the channel.

Let's take a look at the file permissions for the ssl files:

```
cd /var/mqm/qmgrs/QM_71SSL/ssl ls -l -rw----- 1 rivera mqm 782 2014-03-04 12:12 QM_71SSL.crt
-rw----- 1 rivera mqm 10080 2014-03-04 11:51 QM_71SSL.kdb -rw----- 1 rivera mqm 80 2014-03-
```

```
04 11:51 QM_71SSL.rdb -rw----- 1 rivera mqm 129 2014-03-04 11:37 QM_71SSL.sth -rw----- 1
rivera mqm 776 2014-03-04 11:50 rivera.crt
```

Notice that the userid "rivera" is being used in this scenario. This user is a member of the "mqm" group, but the SSL files were created in such a way that other members of the group cannot read/write the files.

Page 17 of 19

Let's change the permissions to allow the group "mqm" to read and write:

```
rivera@veracruz: /var/mqm/qmgrs/QM_71SSL/ssl chmod 660 *
```

```
rivera@veracruz: /var/mqm/qmgrs/QM_71SSL/ssl ls -l -rw-rw---- 1 rivera mqm 782 2014-03-04
12:12 QM_71SSL.crt -rw-rw---- 1 rivera mqm 10080 2014-03-04 11:51 QM_71SSL.kdb -rw-rw---- 1
rivera mqm 80 2014-03-04 11:51 QM_71SSL.rdb -rw-rw---- 1 rivera mqm 129 2014-03-04 11:37
QM_71SSL.sth -rw-rw---- 1 rivera mqm 776 2014-03-04 11:50 rivera.crt
```

Windows: Let's try again:

```
C:\MQ-SupportPac\MO04 SSL Wizard\client_samples\bin> SSLSample.exe veracruz.x.com(1430)
SSL.SVRCONN QM_71SSL NULL_SHA "C:\Users\IBM_ADMIN\AppData\Local\VirtualStore\Program
Files (x86)\IBM\WebSphere MQ_2\rivera" Connecting to: Conname = veracruz.x.com(1430).
SvrconnChannelName = SSL.SVRCONN. QMgrName = QM_71SSL. SSLCiph = NULL_SHA. SSLKeyr =
C:\Users\IBM_ADMIN\AppData\Local\VirtualStore\Program Files (x86)\IBM\WebSphere
MQ_2\rivera. MQCONN ended with reason code 2035 The rc 2035 means: 2035 0x000007f3
MQRC_NOT_AUTHORIZED
```

Unix: Let's take a look at the error log of the queue manager.

```
AMQ9776: Channel was blocked by userid EXPLANATION: The inbound channel 'SSL.SVRCONN' was
blocked from address '9.80.3.88' because the active values of the channel were mapped to a userid
which should be blocked. The active values of the channel were 'MCAUSER(rivera) CLNTUSER(rivera)
SSLPEER(SERIALNUMBER=53:16:00:C7,CN=rivera,OU=Support,O=IBM,ST=NC,C=)'.
```

Page 18 of 19

ACTION: Contact the systems administrator, who should examine the channel authentication records to ensure that the correct settings have been configured. The ALTER QMGR CHLAUTH switch is used to control whether channel authentication records are used. The command DISPLAY CHLAUTH can be used to query the channel authentication records.

Let's display the channel authentication records:

```
runmqsc QM_71SSL DISPLAY CHLAUTH (*) 2 : DISPLAY CHLAUTH (*) AMQ8878: Display channel
authentication record details. CHLAUTH(SYSTEM.ADMIN.SVRCONN) TYPE(ADDRESSMAP)
ADDRESS(*) USERSRC(CHANNEL) AMQ8878: Display channel authentication record
details. CHLAUTH(SYSTEM.ADMIN.*) TYPE(BLOCKUSER) USERLIST(nobody) AMQ8878:
Display channel authentication record details. CHLAUTH(SYSTEM.*)
TYPE(ADDRESSMAP) ADDRESS(*) USERSRC(NOACCESS) AMQ8878: Display channel
authentication record details. CHLAUTH(*) TYPE(BLOCKUSER) USERLIST(nobody
,*MQADMIN)
```

The last record is a rule that indicates that the “*MQADMIN” is going to be blocked for all channels, and this notation indicates an MQ Administrator. Because the userid used in this example is “rivera” and belongs to the group “mqm” is an MQ Administrator, and thus it is blocked.

We have now the choice to create a rule to add rivera as a valid user, or to reduce the blacklist for SSL.SVRCONN, allowing administrators to use this channel. Let's try this 2nd option:

```
SET CHLAUTH(SSL.SVRCONN) TYPE(BLOCKUSER) USERLIST('nobody') end
```

Page 19 of 19

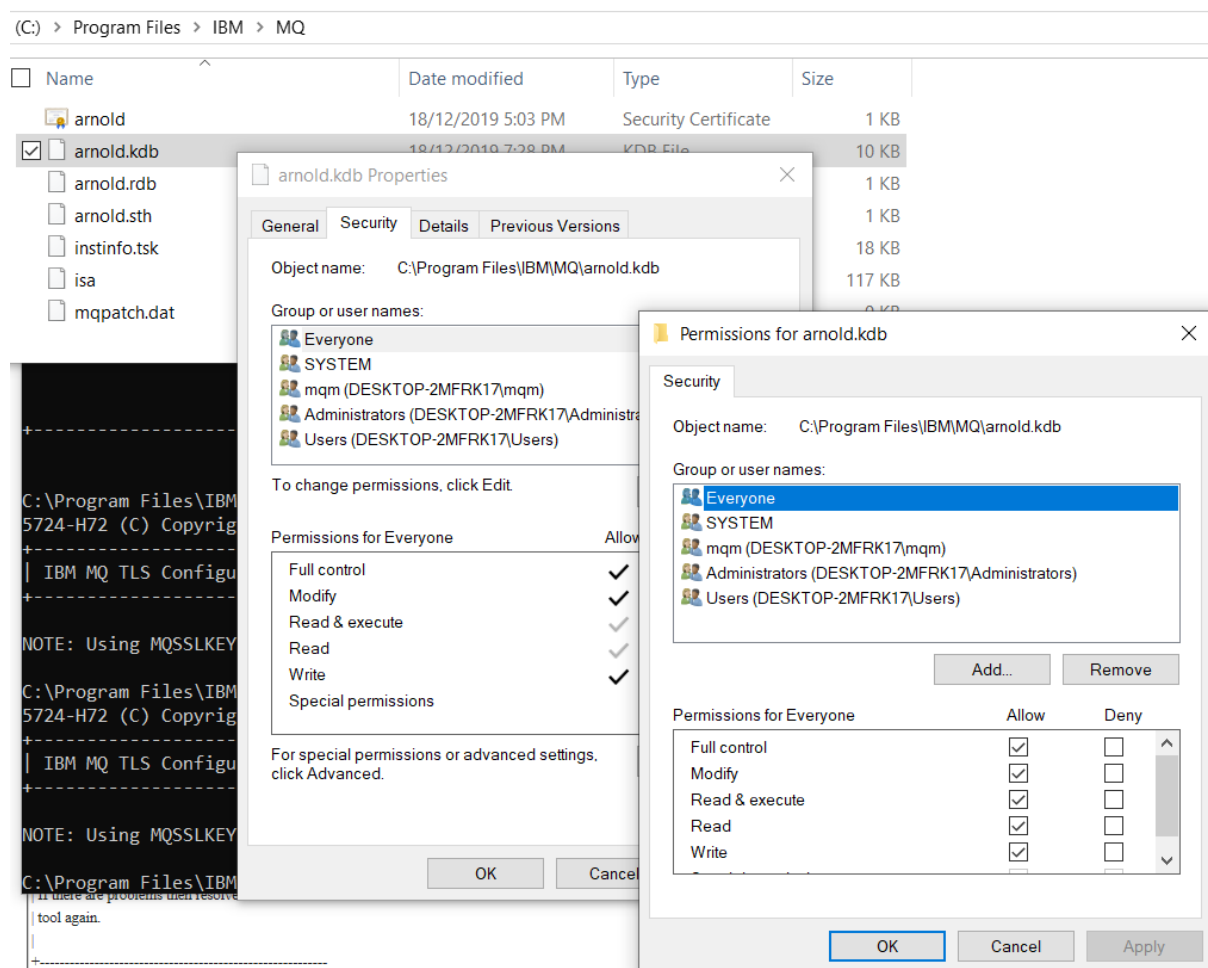
WINDOWS:

Let's try the sample again:

```
C:\MQ-SupportPac\MO04 SSL Wizard\client_samples\bin> SSLSample.exe veracruz.x.com(1430)
SSL.SVRCONN QM_71SSL NULL_SHA "C:\Users\IBM_ADMIN\AppData\Local\VirtualStore\Program
Files (x86)\IBM\WebSphere MQ_2\rivera" Connecting to: Conname = veracruz.x.com(1430).
SvrconnChannelName = SSL.SVRCONN. QMgrName = QM_71SSL. SSLCiph = NULL_SHA. SSLKeyr =
C:\Users\IBM_ADMIN\AppData\Local\VirtualStore\Program Files (x86)\IBM\WebSphere
MQ_2\rivera. Connection Successful!. SSLSample end
```

This time, the connection was successful. Yeah!!!

+++ end ++



https://www.mqtechconference.com/sessions_v2018/Using_runmqsc_and_dmpmqcfg_over_TLS_client.pdf

<http://runmqsc.blogspot.com/>

<https://www-01.ibm.com/support/docview.wss?uid=swg27045974&aid=1>

CCDT setting up

```
DEFINE CHANNEL(TLSPRQM1.SVRCONN) +  
    CHLTYPE(CLNTCONN) +  
    TRPTYPE(TCP) +  
    CONNAME('localhost(2414)') +  
    QMNAME(TLSPRQM1) +  
    SSLCIPH(ANY_TLS12)  
REPLACE
```

```
echo "run the runmqsc command to create the CCDT called AMQCLCHL.TAB in /var/mqm directory"  
runmqsc -n < /tmp/mq/mqsc/TLSPRQM1CLT.MQSC"
```

```
# will need to set up the environment variable for path to  
# AMQCLCHL.TAB if the default CCDT TAB name and default CCDT TAB directory are not used.  
# because I have used all defaults the MQClient code will find and use the AMQCLCHL.TAB  
# separate script package NPP-SET-MQENV can be used to export the MQCHLTAB env variable  
#echo "setting MQCHLTAB to var/mqm for the CCDT .TAB file"  
#  
#export MQCHLTAB=/var/mqm/AMQCLCHL.TAB  
#  
#echo $MQCHLTAB  
#  
#you can test simple client connection with amqsputc  
#putting msgs to APP qmgr RQ1s and they will be picked up by IIB  
#and put back to EQ1s which exist in GW qmgrs across the cluster  
#export MQSERVER=SYSTEM.DEF.SVRCONN/TCP/IPaddress  
#echo $MQSERVER  
#cd var/mqm/samp/bin  
#./amqsputc RQ1 GWQM1  
#export MQCHLTAB=/var/mqm/AMQCLCHL.TAB
```

```
#echo $MQCHLTAB  
#cd var/mqm/samp/bin  
#./amqsputc RQ1 GWQM1
```

Debugging

Lacking a certificate – didn't add the certlab to CLNTCONN in CCDT
from the logs

19/12/2019 10:15:11 - Process(29412.12) User(MUSR_MQADMIN) Program(amqrmppa.exe)

Host(DESKTOP-2MFRK17) Installation(Installation1)

VRMF(9.1.3.0) QMgr(TLSPRQM1)

Time(2019-12-18T23:15:11.742Z)

RemoteHost(127.0.0.1)

CommentInsert1(TLSPRQM1.SVRCONN)

CommentInsert2(gsk_attribute_get_cert_info)

CommentInsert3(kubernetes (127.0.0.1))

AMQ9637E: Channel is lacking a certificate.

EXPLANATION:

The channel is lacking a certificate to use for the SSL handshake. The channel name is 'TLSPRQM1.SVRCONN' (if '????' it is unknown at this stage in the SSL processing).

The remote host is 'kubernetes (127.0.0.1)'.

The channel did not start.

ACTION:

Make sure the appropriate certificates are correctly configured in the key repositories for both ends of the channel.

----- amqccisa.c : 8245 -----

19/12/2019 10:15:11 - Process(29412.12) User(MUSR_MQADMIN) Program(amqrmppa.exe)

Host(DESKTOP-2MFRK17) Installation(Installation1)

VRMF(9.1.3.0) QMgr(TLSPRQM1)
Time(2019-12-18T23:15:11.743Z)
CommentInsert1(TLSPRQM1.SVRCONN)
CommentInsert2(29412(39076))
CommentInsert3(kubernetes (127.0.0.1))

AMQ9999E: Channel 'TLSPRQM1.SVRCONN' to host 'kubernetes (127.0.0.1)' ended abnormally.

EXPLANATION:

The channel program running under process ID 29412(39076) for channel 'TLSPRQM1.SVRCONN' ended abnormally. The host name is 'kubernetes (127.0.0.1)'; in some cases the host name cannot be determined and so is shown as '????'.

ACTION:

Look at previous error messages for the channel program in the error logs to determine the cause of the failure. Note that this message can be excluded completely or suppressed by tuning the "ExcludeMessage" or "SuppressMessage" attributes under the "QMErrorLog" stanza in qm.ini. Further information can be found in the System Administration Guide.

CERTLABL

Certificate label for this channel to use.

The label identifies which personal certificate in the key repository is sent to the remote peer. If this attribute is blank, the certificate is determined by the queue manager **CERTLABL** parameter.

Add the CERTLAB to the CLNTCONN definition and retry

DEFINE CHANNEL(TLSPRQM1.SVRCONN) +

CHLTYPE(CLNTCONN) +

TRPTYPE(TCP) +

CONNNAME('localhost(2414)') +

CERTLABL('ibmmqarnold') +

QMNAME(TLSPRQM1) +

SSLCIPH(ANY_TLS12) +

REPLACE

runmqsc -n < c:\temp\TLSPRQM1CLTCONN.MQSC

Not authorized – used mqm instead of MUSR_MQADMIN in the chl auth rec
From the logs

----- amqrmrsa.c : 945 -----

19/12/2019 10:38:58 - Process(29412.13) User(MUSR_MQADMIN) Program(amqrmppa.exe)

Host(DESKTOP-2MFRK17) Installation(Installation1)

VRMF(9.1.3.0) QMgr(TLSPRQM1)

Time(2019-12-18T23:38:58.367Z)

CommentInsert1(mqm)

CommentInsert2(TLSPRQM1.SVRCONN)

CommentInsert3(TLSPRQM1)

AMQ9245W: Unable to obtain account details for channel MCA user ID.

EXPLANATION:

IBM MQ was unable to obtain the account details for MCA user ID 'mqm'. This user ID was the MCA user ID for channel 'TLSPRQM1.SVRCONN' on queue manager 'TLSPRQM1' and may have been defined in the channel definition, or supplied either by a channel exit or by a client.

ACTION:

Ensure that the user ID is correct and that it is defined on the Windows local system, the local domain or on a trusted domain. For a domain user ID, ensure that all necessary domain controllers are available.

----- cmqxrsrv.c : 2220 -----

19/12/2019 10:38:58 - Process(29412.13) User(MUSR_MQADMIN) Program(amqrmppa.exe)

Host(DESKTOP-2MFRK17) Installation(Installation1)

VRMF(9.1.3.0) QMgr(TLSPRQM1)

Time(2019-12-18T23:38:58.369Z)

ArithInsert1(2) ArithInsert2(2035)

CommentInsert1(mqm)

AMQ9557E: Queue Manager User ID initialization failed for 'mqm'.

EXPLANATION:

The call to initialize the User ID 'mqm' failed with CompCode 2 and Reason 2035. If an MQCSP block was used, the User ID in the MQCSP block was ''. If a userID flow was used, the User ID in the UID header was '' and any CHLAUTH rules applied prior to user adoption were evaluated case-sensitively against this value.

ACTION:

Correct the error and try again.

----- cmqxrsrv.c : 2524 -----

When I created the SVRCONN in the first place I had a CHLAUTH for MCAUSER('mqm')

This should have been MCAUSER('MUSR_MQADMIN')

Change it in MQExplorer



Channel	Address Map	User ID
SYSTEM.*		
SYSTEM.ADMIN.SVRCONN		
TLSPRQM1.SVRCONN		
TLSPRQM1.SVRCONN		

No CipherSpec – used MQSERVER instead of CCDT.

MQ9639E: Remote channel 'TLSPRQM1.SVRCONN' did not specify a CipherSpec.

EXPLANATION:

Remote channel 'TLSPRQM1.SVRCONN' did not specify a CipherSpec when the local channel expected one to be specified.

Do we have to use MQCCDT file in order for runmqsc to client connect because the MQSERVER limits the details we can specify for the connection to the SVRCONN

A CCDT enables the specification of cipher specs etc

Debugging in local container

```
C:\temp>runmqsc -c TLSPRQM1
5724-H72 (C) Copyright IBM Corp. 1994, 2019.
Starting MQSC for queue manager TLSPRQM1.

AMQ9709E: SSL/TLS initialization error.

0 command responses received.
```

Docker ps

Docker exec -it <container id> /bin/bash

```
C:\temp>docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
5edaadeb1fe0   davexacom/ibm-mqadvanced-server-tls:9.1.3.0-r4-amd64   "runmqintegrationser..." About an hour ago Up About an hour   0.0.0.0:32779->1414/tcp, 0.0.0.0:32778->9157/tcp, 0.0.0.0:32777->9443/tcp   silly_noether

C:\temp>docker exec -it 5edaadeb1fe0 /bin/bash
bash-4.2$ ls
bin boot dev etc home lib lib64 licenses media mnt opt proc root run sbin srv sys tmp usr var
bash-4.2$
```

Find the error logs for MQ

/var/mqm/qmgrs/TLSPRQM1/errors

```
bash-4.2$ ls /var/mqm/qmgrs/TLSPRQM1/errors
AMQERR01.json AMQERR01.LOG AMQERR02.json AMQERR02.LOG AMQERR03.json AMQERR03.LOG
```

tail /var/mqm/qmgrs/TLSPRQM1/errors/AMQERR01.json

"AMQ9660E: SSL key repository: password stash file absent or unusable."}

```
{"ibm_messageId":"AMQ9999E","ibm_arithInsert1":0,"ibm_arithInsert2":0,"ibm_commentInsert1":"
????","ibm_commentInsert2":"258","ibm_commentInsert3":"gateway
(172.17.0.1)","ibm_datetime":"2019-12-
20T01:05:05.311Z","ibm_serverName":"TLSPRQM1","type":"mq_log","host":"5edaadeb1fe0","logle
vel":"ERROR","module":"amqrmrsa.c:945","ibm_sequence":"1576803905_312716800","ibm_qmgrI
d":"TLSPRQM1_2019-12-
19_23.50.28","ibm_processId":"258","ibm_threadId":"9","ibm_version":"9.1.3.0","ibm_processNam
e":"amqrmpa","ibm_userName":"mqm","ibm_installationName":"Installation1","ibm_installationD
ir":"/opt/mqm","message":"AMQ9999E: Channel '????' to host 'gateway (172.17.0.1)' ended
abnormally."}
```

h