Rabbit MQ setup and MaxScale Integration

Massimiliano Pinto, Markus Makela, Timofey Turenko

Last Updated: 4rd September 2014

Contents

Document History

Date	Comment
2nd September 2014	Initial Version
3rd September 2014	Added logging triggers example
4th September 2014	Repositories setup instruction added

Introduction

A step by step guide helps installing a RabbitMQ server and testing it before MaxScale integration.

New plugin filter and a message consumer application need to be compiled and linked with an external C library, RabbitMQ-c, that provides AMQP protocol integration.
Custom configuration, with TCP/IP and Queue parameters, is also detailed here.
The software install setup provides RPM packaging and traditional download and compilation steps.

Step 1 - Get the RabbitMQ binaries

On Centos 6.5 using fedora / RHEL rpm get the rpm from http://www.rabbitmq.com/

```
 rabbitmq-server-3.3.4-1.noarch.rpm
```

Please note, before installing RabbitMQ, you must install Erlang.

Example:

```
# yum install erlang
...

Package erlang-R14B-04.3.el6.x86_64 already installed and latest version
```

Step 2 - Install and Start the Server

```
# rpm -i rabbitmq-server-3.3.4-1.noarch.rpm

# /etc/init.d/rabbitmq-server start
```

rabbitmqctl is a command line tool for managing a RabbitMQ broker. It performs all actions by connecting to one of the broker's nodes.

```
# rabbitmqctl list_queues

# rabbitmqctl list_queues | list_exchanges| cluster_status | list_bindings
| list_connections | list_consumers | status
[root@maxscale-02 MaxScale]# rabbitmqctl status
```

```
Status of node 'rabbit@maxscale-02' ...
[{pid,12251},
 {running_applications,[{rabbit,"RabbitMQ","3.3.4"},
                        {os_mon,"CPO  CXC 138 46","2.2.7"},
                        {xmerl,"XML parser","1.2.10"},
                        {mnesia,"MNESIA  CXC 138 12","4.5"},
                        {sasl,"SASL  CXC 138 11","2.1.10"},
                        {stdlib,"ERTS  CXC 138 10","1.17.5"},
                        {kernel,"ERTS  CXC 138 10","2.14.5"}]},
 {os,{unix,linux}},
 {erlang_version,"Erlang R14B04 (erts-5.8.5) [source] [64-bit] [smp:2:2] [rq:2]
[async-threads:30] [kernel-poll:true]\n"},
...
 {listeners,[{clustering,25672,"::"},{amqp,5672,"::"}]},
...
...done.
```

```
[root@maxscale-02 MaxScale]# rabbitmqctl list_bindings
Listing bindings ...
x1      exchange        q1      queue   k1      []
...done.
```

Interaction with the server may require stop & reset at some point:

```
# rabbitmqctl stop_app
```

```
# rabbitmqctl reset
```

```
# rabbitmqctl start_app
```

Step 3 - Install and test the client libraries

The selected library for MaxScale integration is:

https://github.com/alanxz/rabbitmq-c

(1) manual software compilation task

```
# mkdir /packages/

# cd /packages

# git clone  https://github.com/alanxz/rabbitmq-c.git
# cd rabbitmq-c
# cmake -DCMAKE_INSTALL_PREFIX=/packages/setup/rabbitmq-c
# cmake --build .
# make
# make install
```

Please note, the **/packages/setup/rabbitmq-c/lib64** path will be used for **LIBS** and **/packages/setup/rabbitmq-c/include** for **INCLUDES** when compiling MaxScale new components.

(2) setup the EPEL repository

On Centos 6.5 setup EPEL repository.

```
# cd /packages/
# wget
http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
# sudo rpm -Uvh epel-release-6*.rpm
```

Install the software:

```
# yum install librabbitmq.x86_64
```

you might also like to install:

```
librabbitmq-tools.x86_64, librabbitmq-devel.x86_64
```

Please note you may also install the rabbitmq server from that repository:

```
# yum install rabbitmq-server
```

(3) basic tests with library

The required library librabbitmq-c is now installed and we continue with basic operations with amqp_* tools, located in /packages/rabbitmq-c//build/examples, testing client server interaction.

Please note, those example applications may not be included in the RPM library packages.

- Test 1 - create the exchange

```
[root@maxscale-02 examples]# pwd
```

```
/packages/rabbitmq-c/examples
```

```
[root@maxscale-02 examples]# ./amqp_exchange_declare
Usage: amqp_exchange_declare host port exchange exchangetype

[root@maxscale-02 examples]# ./amqp_exchange_declare 127.0.0.1 5672 foo direct
```

- Test 2 - Listen to exchange with selected binding key

```
[root@maxscale-02 examples]# ./amqp_listen
Usage: amqp_listen host port exchange bindingkey

[root@maxscale-02 examples]# ./amqp_listen 127.0.0.1 5672 foo k1 &
```

- Test 3 - Send a message …

```
[root@maxscale-02 examples]# ./amqp_sendstring
Usage: amqp_sendstring host port exchange routingkey messagebody

[root@maxscale-02 examples]# ./amqp_sendstring 127.0.0.1 5672 foo k1 “This is a new message”
```

... and watch the listener output

```
Delivery 1, exchange foo routingkey k1
Content-type: text/plain
```

Step 4 - MaxScale integration with librabbitmq-c

A new filter (mqfilter.c) is implemented in order to send messages to the rabbitmq server and a message consumer (rabbitmq_consumer/consumer.c) program will get messages and store them into a MySQL/MariaDB database.There are two ways:

If the librabbitmq-c library is manually compiled it may be necessary to update **server/modules/filterMakefile** adding

to CFLAGS:
`-I/rabbitmq-include-dir`

- to LDFLAGS
`-L/packages/rabbitmq-c/rabbitmq-c/librabbitmq`

and also changing LIBRARY_DIRS and INCLUDE_DIRS in **rabbitmq_consumer/buildconfig.inc**

Please note, Message Queue Consumer (consumer.c) also needs to compiled with MySQL/MariaDB client libraries, this example may help:

`buildconfig.inc:`

```
LIBRARY_DIRS :=-L/packages/mariadb_client-2.0.0-Linux/lib/mariadb
-L/packages/setup/rabbitmq-c/lib64

INCLUDE_DIRS :=-I/usr/include -I/packages/mariadb_client-2.0.0-Linux/include/mariadb
-I/packages/setup/rabbitmq-c/include
```

The message queue consumer must be built as a separate task, it's not built as part of MaxScale build system.

```
# cd rabbitmq_consumer
# make
```

Another setup is the RPM via SkySQL repository.

Put following into /etc/yum.repos.d/skysql.repo

```
[skysql]
name = skysql
baseurl =http://jenkins.engskysql.com/repo/Z3/centos6.5_x86_64/
```

```
gpgcheck=1
```

Download and Import GPG key:

```
wget http://jenkins.engskysql.com/repo/MariaDBManager-GPG-KEY.public
```

```
rpm --import MariaDBManager-GPG-KEY.public
```

Setup MariaDB repository.
Put following into /etc/yum.repos.d/mariadb.repo:

```
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/5.5/centos6-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB

gpgcheck=1
```

and install the software

```
# yum install rabbitmq-message-consumer
```

Step 5 - Configure new applications

(1) The new filter need to be configured in maxScale.cnf

```
[Test Service]
type=service
router=readconnroute
router_options=slave
servers=server1,server2,server3,server5,server4
user=massi
passwd=massi
```

```
enable_root_user=0
filters=MQ

[MQ]
type=filter
module=mqfilter
exchange=x1
key=k1
queue=q1
hostname=127.0.0.1
port=5672

# log all incoming queries
logging_trigger=all
```

Logging triggers define whether to log all or a subset of the incoming queries using these options:

```
# log only some elements or all
logging_trigger=[all,source,schema,object]

# Whether to log only SELECT, UPDATE, INSERT and DELETE queries or all possible
queries
logging_log_all=true|false



# Log only when any of the trigger parameters match or only if all parameters
match
logging_strict=true|false

# specify objects
logging_object=mytable,another_table

# specify logged users
logging_source_user=testuser,testuser
```

```
# specify source addresses
logging_source_host=127.0.0.1,192.168.10.14

# specify schemas
logging_schema=employees,orders,catalog
```

Example:

```
logging_trigger=object,schema,source
logging_strict=false
logging_log_all=false
logging_object=my1
logging_schema=test
logging_source_user=maxtest
```

The logging result is:

- if user maxtest does something, it's logged
- and all queries in test schema are logged
- anything targeting my1 table is logged
- SELECT NOW(), SELECT MD5(“xyz)” are not logged

Please note.

- if we want to log only the user ‘maxtest’ accessing the schema ‘test’ with target ‘my1’ the option `logging_strict` must be set to TRUE
- if we want to include those selects without schema name the option `logging_log_all` must be set to TRUE

The mqfilter logs into the MaxScale TRACE log informations about the matched logging triggers and the message delivering:

```
2014 09/03 06:22:04   Trigger is TRG_SOURCE: user: testuser = testuser
2014 09/03 06:22:04   Trigger is TRG_SCHEMA: test = test
2014 09/03 06:22:04   Trigger is TRG_OBJECT: test.t1 = t1
2014 09/03 06:22:04   Routing message to: 127.0.0.1:5672 / as guest/guest, exchange:
x1<direct> key:k1 queue:q1
```

The consumer application needs to be configured as well:

```
# more consumer.cnf

#
#The options for the consumer are:
#hostname     RabbitMQ hostname
#port         RabbitMQ port
#vhost        RabbitMQ virtual host
#user         RabbitMQ username
#passwd       RabbitMQ password
#queue        Name of the queue to use
#dbserver     SQL server name
#dbport       SQL server port
#dbname       Name of the database to use
#dbuser       SQL server username
#dbpasswd     SQL server password
#logfile      Message log filename

[consumer]
hostname=127.0.0.1
port=5672
vhost=/
user=guest
passwd=guest
queue=q1
dbserver=127.0.0.1
```

```
dbport=3308
dbname=mqpairs
dbuser=xxx
dbpasswd=yyy
```

We may probably need to modify LD_LIBRARY_PATH before launching ‘consumer’:

```
# export
LD_LIBRARY_PATH=/packages/rabbitmq-c/rabbitmq-c/librabbitmq:/packages/mariadb_client-2.0.
0-Linux/lib/mariadb:/usr/lib64
```

and finally we can launch it:

```
# ./consumer
```

If the consumer.cnf file is not in the same directory as the binary file is, you can provide the location of the folder that it is in by passing it the -c flag followed by the path:

```
# ./consumer -c path/to/file
```

and start maxScale as well

Step 6 - Test the filter and check collected data

assuming maxScale and message consumer successfully running let's connect to the service with active mqfilter:

```
[root@maxscale-02 MaxScale]#  mysql -h 127.0.0.1 -P 4506 -uxxx -pyyy
...

MariaDB [(none)]> select RAND(3), RAND(5);
+--------------------+---------------------+
| RAND(3)            | RAND(5)             |
+--------------------+---------------------+
| 0.9057697559760601 | 0.40613597483014313 |
+--------------------+---------------------+
1 row in set (0.01 sec)

...
MariaDB [(none)]> select RAND(3544), RAND(11);
```

we can check the consumer output in the terminal where it was started:

```
-------------------------------------------------------------
Received: 1409671452|select @@version_comment limit ?
Received: 1409671452|Columns: 1
...
Received: 1409671477|select RAND(?), RAND(?)
Received: 1409671477|Columns: 2
```

We query now the database for the content collected so far:

```
MariaDB [(none)]> use mqpairs;

Database changed
```

```
MariaDB [mqpairs]> select * from pairs;

+----------------------------------+----------------------------------+------------+--
------------------+---------------------+---------+
| tag                              | query                            | reply      |
date_in             | date_out            | counter |
+----------------------------------+----------------------------------+------------+--
------------------+---------------------+---------+
| 006c006d006e006f007000710072007374  | select @@version_comment limit ? | Columns: 1 |
2014-09-02 11:14:51 | 2014-09-02 11:26:38 |       3 |
| 007500760077007800790079007a007b007c7d  | SELECT DATABASE()                | Columns: 1 |
2014-09-02 11:14:56 | 2014-09-02 11:27:06 |       3 |
| 007e007f008000810082008300840085 86  | show databases                   | Columns: 1 |
2014-09-02 11:14:56 | 2014-09-02 11:27:06 |       3 |
| 008700880089008a008b008c008d008e8f  | show tables                      | Columns: 1 |
2014-09-02 11:14:56 | 2014-09-02 11:27:06 |       3 |
| 009000910092009300940095009600979 98  | select * from mqpairs.pairs      | Columns: 6 |
2014-09-02 11:15:00 | 2014-09-02 11:27:00 |      12 |
| 00fc00fd00fe00ff0100010101020103104 | select NOW()                     | Columns: 1 |
2014-09-02 11:24:23 | 2014-09-02 11:24:23 |       1 |
| 0105010601070108010901 0a010b010c10d | select RAND(?), RAND(?)          | Columns: 2 |
2014-09-02 11:24:37 | 2014-09-02 11:24:37 |       1 |
+----------------------------------+----------------------------------+------------+--
------------------+---------------------+---------+
7 rows in set (0.01 sec)

MariaDB [mqpairs]>
```

The filter send queries to the RabbitMQ server in the canonical format, i.e `select RAND(?), RAND(?)`

The queries Message Queue Consumer application gets from the server are stored with a counter that quickly shows how many times that normalized query was received:

```
| 0105010601070108010901 0a010b010c10d | select RAND(?), RAND(?)          | Columns: 2 |
2014-09-02 11:24:37 | 2014-09-02 11:29:15 |       3 |
```