RabbitMQ for OpenVMS Integrity

Brett Cameron (brett.cameron@hp.com), John Apps (john.apps@hp.com) December 2011

Introduction

Thank you for your interest in this port of RabbitMQ (http://www.rabbitmq.org), an implementation of the Advanced Message Queuing Protocol (AMQP) specification (see http://www.amqp.org).

RabbitMQ provides a robust and flexible messaging platform that has been designed from the ground up to interoperate with other messaging systems. It is the leading (and arguably the most popular) implementation of AMQP, the open standard for business messaging, and, through adapters, supports protocols such as XMPP, SMTP, STOMP, and HTTP for lightweight web messaging.

The RabbitMQ server is written in Erlang (www.erlang.org) and is built on the Open Telecom Platform (OTP) framework for clustering and failover. The authors believe that the high availability and functional characteristics of Erlang/OTP make it ideally suited to use in an OpenVMS environment, and by inference the same may be said for RabbitMQ.

The following three sections, “What is AMQP”, “Why AMQP”, and “Scope of AMQP” are taken from the AMQP Specification:

- **What is AMQP?**

  The Advanced Message Queuing Protocol (AMQP) enables full functional interoperability between conforming clients and messaging middleware servers (also called "brokers").

  It is our aim that, through AMQP, messaging middleware capabilities may ultimately be driven into the network itself, and that through the pervasive availability of messaging middleware, new kinds of useful applications may be developed.

- **Why AMQP**

  Our goal is to enable the development and industry-wide use of standardized messaging middleware technology that will lower the cost of enterprise and systems integration and provide industrial-grade integration services to a broad audience.

- **Scope of AMQP**

  To enable complete interoperability for messaging middleware requires that both the networking protocol and the semantics of the broker services are sufficiently specified.

  AMQP, therefore, defines both the network protocol and the broker services through:

  - *A defined set of messaging capabilities called the "Advanced Message Queuing Protocol Model" (AMQP Model).*

    The AMQP Model consists of a set of components that route and store messages within the broker, plus a set of rules for wiring these components together.

- *A network wire-level protocol, AMQP, that lets client applications talk to the broker and interact with the AMQP Model it implements.*

One can partially imply the semantics of the server from the AMQP protocol specifications but we believe that an explicit description of these semantics helps the understanding of the protocol.

This release of RabbitMQ for OpenVMS is based on the RabbitMQ 2.7.0 distribution. New features in this release include order preservation of messages re-queued for a consumer, clients accept a new "amqp" URI scheme for connections, and significant performance improvements. Please refer to http://lists.rabbitmq.com/pipermail/rabbitmq-discuss/2011-November/016069.html for details of these and other new features and improvements included in the 2.7.0 release.

Requirements

The kit you are receiving has been compiled and built using the operating system and product versions listed below. While it is highly likely that you will have no problems installing and using the kit on systems running higher product versions of the products listed, we cannot say for sure that you will be so lucky if your system is running older versions. If you require a kit for a different configuration, we will do what we can to oblige. Note that the UnZip utility is required in order to unpack the ZIP kit.

- OpenVMS 8.3 or higher (IA64 only)
- HP TCP/IP Services V5.6 or higher (if you wish to perform network operations)

  It has not been verified whether the kit works with the MultiNet TCP/IP stack, but there is a good chance that it will.
- Erlang R13A for OpenVMS (the `ERLANG-R13A-VMS-I64-05.ZIP` kit, which includes HP SSL V1.4 support, and other components required by RabbitMQ).
- UnZip 5.42 (or similar) for OpenVMS (required to unpack the ZIP kit and can be found on the OpenVMS Freeware CD)
- HP SSL V1.4 for OpenVMS is required if you intend to use any RabbitMQ plugins that require SSL. Note that HP SSL V1.3 for OpenVMS is not supported by this release.
- Java 1.5 or higher (optional). RabbitMQ provides a Java client that may be used to test the RabbitMQ installation and to develop platform-neutral AMQP applications. The Java client is available at http://www.rabbitmq.com/java-client.html.
- An ODS-5 file system with approximately 9000 blocks of free disk space (you should not attempt to install or use the software on an ODS-2 file system).

In addition to the above requirements, it is assumed that the reader has a good knowledge of OpenVMS and software development in the OpenVMS environment.

Known issues

1. RabbitMQ uses the OTP distributed database, Mnesia, to reliably and persistently replicate session state across all nodes in a cluster. There is an issue if the RabbitMQ broker is shutdown uncleanly or crashes that the database may be left in an unknown or corrupted state. Care should be taken to ensure that the broker is shutdown correctly; however if the database files do become corrupted, it may be necessary to delete the contents of `rabbitmq$root:[mnesia]` and its sub-directories before you can restart

the broker (although typically the broker will manage this situation). Note that this is an issue with current port of Erlang to OpenVMS, as opposed a defect in RabbitMQ.

2. There is a chance that the broker may fail when operating under sustained extreme load. The authors believe this problem is related to the operation of the Erlang memory management algorithms on OpenVMS, as opposed to an issue with RabbitMQ. This matter will hopefully be addressed in future releases of Erlang for OpenVMS; however in the interim, the authors recommend running RabbitMQ with a relatively small memory footprint by setting the RabbitMQ parameter `vm_memory_high_watermark` to a low value (this parameter is set to 0.15 in the configuration file supplied with the kit) and setting the parameter `memory_alarms` to "`true`". These settings will have some impact on the performance of the broker, but will significantly improve stability.

3. A number of plug-ins are shipped with this release; however not all of these plug-ins will work correctly with Erlang 13A and consequently no facility is provided on OpenVMS to unpack and utilize these plug-ins. It is hoped that a later version of Erlang for OpenVMS will be available in the not too distant future.

Contents of the kit

The kit is currently provided as a ZIP file (created using Zip 2.3 for OpenVMS). The command `unzip -l RABBITMQ-VMS-I64-270.ZIP` run against the ZIP file kit will provide a complete listing of the contents of the ZIP file. The listing is not included here as it is considerable in length.

Required privileges and quotas

The following authorized privileges are required in order to start (and stop) the RabbitMQ broker process:

- `TMPMBX`
- `BYPASS`
- `SYSPRV`
- `DETACH`

Depending on the level of usage, the RabbitMQ broker process can consume considerable resources, particularly buffered I/O, file handles, and memory. The authors typically operate with quota settings similar to the following, which should be adequate for most purposes.

```
Maxjobs:          0  Fillm:        512  Bytlm:       128000
Maxacctjobs:      0  Shrfillm:      0  Pbytlm:           0
Maxdetach:        0  BIOlm:       150  JTquota:       4096
Prclm:           50  DIOlm:       150  WSdef:         4096
Prio:             4  ASTlm:       300  WSquo:         8192
Queprio:          4  TQElm:       100  WSextent:     16384
CPU:         (none)  Enqlm:      4000  Pgflquo:     700000
```

Installation and setup

The kit is currently provided as a ZIP file (created using Zip 2.3 for OpenVMS), which allows individual users to install the software under their own accounts, if so desired.

Unpacking and installing the ZIP file kit is very straightforward. After copying the supplied ZIP file (RABBITMQ-VMS-I64-270.ZIP) to a suitable location, unpack the contents of the file using the unzip command:

```
$ unzip RABBITMQ-VMS-I64-270.ZIP
```

After unpacking the kit you will find a [.RabbitMQ270] directory below your current directory that contains all the files that were extracted by unzip. Note that you must install the software onto an ODS5-formatted disk.

To complete the installation it is necessary to define the concealed logical name rabbitmq$root to point to the top-level [.RabbitMQ270] directory and to modify the command procedures rabbitmq$startup.com and rabbitmq$run.com to specify the correct hostname and any desired run-time options.

The following notes describe the necessary post-installation steps:

1. Set default to [.RabbitMQ270.sbin] and edit the start-up command procedure rabbitmq$startup.com. Modify the definition of logical name rabbitmq$root as required to point to the top level [.RabbitMQ270] directory and save your changes. For example, your definition of rabbitmq$root may appear as follows:

   ```
   $ DEFINE/SYSTEM/EXEC/TRANS=CONC RABBITMQ$ROOT DKA800:[USER.BIGGLES.RABBITMQ270.]
   ```

2. Edit the command procedure rabbitmq$run.com. Modify the value of the variable RABBITMQ_NODE_IP_ADDRESS to specify the TCP/IP for your server, and save your changes.

   You may wish to change the values of some of the other environment variables; however care should be taken when specifying directories – these must be specified using UNIX-style syntax. The default values for all modifiable environment variables (other than RABBITMQ_NODE_IP_ADDRESS) are shown below (note the use of UNIX syntax for directory paths):

   ```
   $ RABBITMQ_MNESIA_DIR = "/rabbitmq$root/mnesia"
   $ RABBITMQ_LOGS = "/rabbitmq$root/log"
   $ RABBITMQ_SASL_LOGS = "/rabbitmq$root/log"
   $ RABBITMQ_NODENAME = "rabbit"
   $ RABBITMQ_NODE_PORT = 5672
   ```

   The value of the variable RABBITMQ_NODE_PORT specifies the port number to be used by the RabbitMQ broker. Unless you are planning on running multiple instances of the broker, the authors would recommend keeping the default AMQP port number (5672).

3. Edit the command procedure rabbitmqctl.com. Modify the value of the environment variable RABBITMQ_NODENAME to specify the correct node-name (the name after the "@" symbol), and save your changes.

4. Review the RMS file attributes of the file rabbitmq.config and verify that the record format is "stream-LF":

   ```
   $ analyze/rms rabbitmq.config
   .
   .
   .
   RMS FILE ATTRIBUTES

           File Organization: sequential
           Record Format: stream-LF
           Record Attributes:  carriage-return
           Maximum Record Size: 0
           Longest Record: 74
           Blocks Allocated: 16, Default Extend Size: 0
   ```

```
        End-of-File VBN: 2, Offset: %X'0103'
        File Monitoring: disabled
        Global Buffer Count  pre-V8.3:          0
        Global Buffer Count post-V8.3:          0
        Global Buffer Flags post-V8.3:       none
```

If the record format is not `stream-LF`, Erlang will not read the configuration file correctly and RabbitMQ will fail to start. You can set the attributes on the file to stream-LF using the following command:

```
$ set file/attr=(rfm:stmlf) rabbitmq.config
```

5. Ensure that Erlang is installed and running. Specifically, ensure that the logical names `erlang$root` and `erlang$top` are defined system-wide and that the Erlang portmapper daemon (`epmd$server`) is running.

   You can verify that the Erlang portmapper daemon is running as follows:

```
$ show system/proc=epmd*
OpenVMS V8.3-1H1  on node CCIN02    8-DEC-2011 19:30:49.99   Uptime  132 04:05:37
  Pid    Process Name    State  Pri      I/O       CPU       Page flts  Pages
000056D0 EPMD$SERVER     LEF      5  1006793   0 00:00:00.16      1127    375
```

6. Set default to the `[.sbin]` directory and execute the command procedure `rabbitmq$startup.com` to define the necessary logical names and to start the RabbitMQ broker process.

Optionally include the procedures `rabbitmq$root:[sbin]rabbitmq$startup.com` and `rabbitmq$root:[sbin]rabbitmq$shutdown.com` in the OpenVMS system start-up and shutdown procedures, respectively. Be sure to place execute the RabbitMQ start-up procedure after Erlang has been started, and ensure that the RabbitMQ shutdown procedure is executed before shutting down Erlang.

Testing the installation

The following steps assume you are familiar with Java on OpenVMS and that you have set up your process environment appropriately. Java 1.5 or higher is required in order to run the RabbitMQ examples described below.

1. Download to your OpenVMS system the V2.7.0 RabbitMQ binary Java Client distribution from http://www.rabbitmq.com/java-client.html (download the ZIP file).
2. Unzip the ZIP file into the directory of your choice. You should now have a directory called `[.RABBITMQ-JAVA-CLIENT-BIN-2_7_0]`. Set default into this directory.
3. Define `JAVA$CLASSPATH` to point at the libraries (jar files) from the binary kit that you unzipped in the previous step. For example:

```
$ define java$classpath -
    DISK$USER:[BIGGLES.RABBITMQ-JAVA-CLIENT-BIN-2_7_0]rabbitmq-client.jar", -
    DISK$USER:[BIGGLES.RABBITMQ-JAVA-CLIENT-BIN-2_7_0]commons-io-1_2.jar", -
    DISK$USER:[BIGGLES.RABBITMQ-JAVA-CLIENT-BIN-2_7_0]commons-cli-1_1.jar", -
    DISK$USER:[BIGGLES.RABBITMQ-JAVA-CLIENT-BIN-2_7_0]junit.jar, -
    DISK$USER:[BIGGLES.RABBITMQ-JAVA-CLIENT-BIN-2_7_0]rabbitmq-client-tests.jar, -
    []
```

   You might wish to place this class-path information in a command procedure for future use.

4. Open a second OpenVMS session. Set default into the `[.RABBITMQ-JAVA-CLIENT-BIN-2_7_0]` directory, and repeat Step 3 to above to define the class-path for this session.

5. In the first window, enter the following command, replacing 10.0.0.1 with the TCP/IP address of your server:

```
$ java com.rabbitmq.examples.SimpleProducer 10.0.0.1
```

In the second window, after setting up the Java environment enter the following command with the correct TCP/IP address:

```
$ java com.rabbitmq.examples.SimpleConsumer 10.0.0.1
Message: the time is Mon Dec 06 05:31:53 NZDT 2010
```

A message similar to the one shown above should be displayed containing the current date and time.

Note that you may use the logical name `TCPIP$INET_HOSTADDR` and the lexical function `f$trnlnm()` to obtain the TCP/IP address of your server:

```
$ java com.rabbitmq.examples.SimpleProducer "''F$TRNLNM("TCPIP$INET_HOSTADDR")'"
```

Recommended reading

Before getting too carried away, be sure to read (or at least browse) some of the tutorials and other excellent documentation available on the RabbitMQ web site (http://www.rabbitmq.org). You might also wish to join the RabbitMQ mailing lists (http://lists.rabbitmq.com/cgi-bin/mailman/listinfo), which will provide you with direct access to the RabbitMQ developers, and a thriving, knowledgeable, and helpful community of other RabbitMQ users.

Appendix

Tools

Over time the authors will update this document with tools they have found useful in the course of their work. The authors welcome any suggestions and will include them in future versions of the document.

- Zip and UnZip for OpenVMS may be obtained from the OpenVMS freeware CD at http://h71000.www7.hp.com/openvms/freeware/.