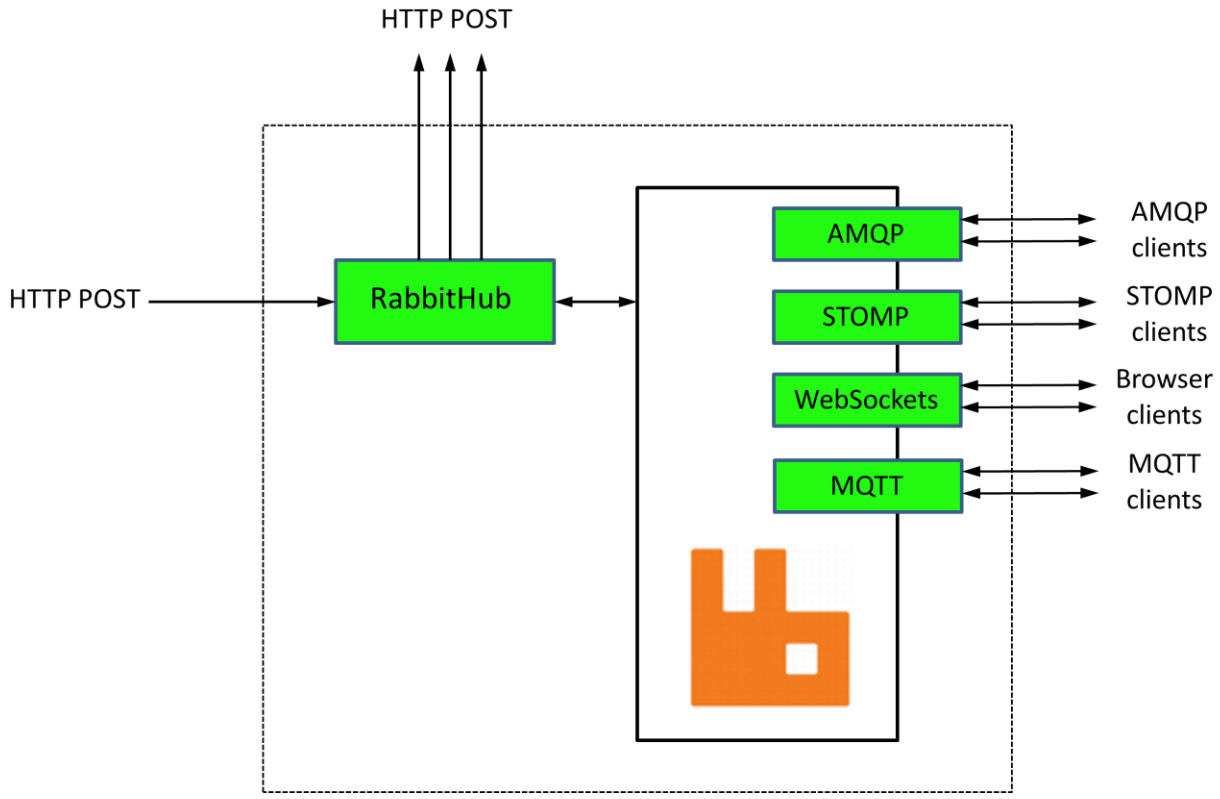
RabbitHub

Overview

RabbitHub is a RabbitMQ plugin-based implementation of PubSubHubBub (a simple web-hook-based pub/sub protocol) that provides an HTTP-based interface to RabbitMQ. It gives every AMQP exchange and queue hosted by the broker two URL's: one to use for delivering messages to the exchange or queue, and one to use to subscribe to messages forwarded on by the exchange or queue. Subscriptions supply a callback URL to RabbitHub that is used to deliver messages via HTTP POST. While the initial aim of the plugin was support for PubSubHubBub, the plugin provides a generally useful, easy to use, and efficient means of interacting with RabbitMQ in a RESTful manner. When used in combination with other plugins such as those for STOMP, WebSockets, and MQTT, the result is an extremely powerful and flexible messaging solution. Messages may be published via one protocol and consumed by another (or indeed by multiple protocols when messages are published to fanout or topic exchanges).



RabbitHub was originally written by Tony Garnock-Jones (one of the original members of the RabbitMQ team) in 2009; however RabbitMQ has evolved considerably since this time, and it has been necessary to update the code in order for it to work with current RabbitMQ versions. The updated code and

additional notes can be found at https://github.com/tonyg/rabbithub, and the binary distribution (which also happens to include the sources) can be found here.

Requirements

- Erlang 14B or higher (preferably Erlang 15B)
- RabbitMQ 2.8.2 or higher

Installation

The plugin is easily installed and configured according to the following steps:

1. Copy the binary distribution (`rabbithub.ez`) to the RabbitMQ plugins directory. Assuming a standard Ubuntu distribution and RabbitMQ version 2.8.4, this may be done as follows:

   ```
   sudo cp rabbithub.ez \
   /usr/lib/rabbitmq/lib/rabbitmq_server-2.8.4/plugins/
   ```

2. Enable the plugin:

   ```
   sudo rabbitmq-plugins enable rabbithub
   ```

3. Restart RabbitMQ:

   ```
   sudo /etc/init.d/rabbitmq-server restart
   ```

Note that by default the RabbitHub plugin will use TCP/IP port 55670. This may be changed by editing `rabbitmq.config` and restarting RabbitMQ (see http://www.rabbitmq.com/mochiweb.html for details). As currently implemented, the port used by the RabbitHub plugin (55670) clashes with that used by RabbitMQ-Web-Stomp, so be wary of enabling both plugins! We will see about fixing this shortly (there are probably one or two other related things that need to be tidied up also).

Usage

The plugin is extremely simple to use. The following basic operations are supported (illustrated using a few simple cURL commands):

- **Create queue**

  The following `PUT` command will create a new queue named "`foo`". Note that the plugin checks that the supplied credentials are valid and that the user has permission to perform the requested operation.

  ```
  $ curl -v -X PUT http://guest:guest@localhost:55670/endpoint/q/foo
  ```

- **Delete queue**

  The following `DELETE` command may be used to delete the queue "`foo`".

  ```
  $ curl -v -X DELETE http://guest:guest@localhost:55670/endpoint/q/foo
  ```

- **Create exchange**

  The following `PUT` command will create a new exchange named "`junk`".

```
$ curl -X PUT http://guest:guest@localhost:55670/endpoint/x/junk
```

Note that the exchange type will by default be "fanout". If you wish to create another type of exchange then you can do so by specifying the type in the query string. For example, appending the following query string to the above URL would cause a "direct" exchange to be created:

```
amqp.exchange_type=direct
```

- **Delete exchange**

  The following `DELETE` command may be used to delete the exchange "`junk`".

  ```
  $ curl -X DELETE http://guest:guest@localhost:55670/endpoint/x/junk
  ```

- **Subscribe to an exchange**

  A `POST` request of the following format may be used to subscribe to an exchange:

  ```
  curl -vd \
  "hub.mode=subscribe&hub.callback=http://10.1.1.8:4567/sub1&hub.topic=foo&h
  ub.verify=sync&hub.verify=async&hub.lease_seconds=86400" \
  http://guest:guest@localhost:55670/subscribe/x/amq.direct
  ```

  For a detailed description of the various query string name-value pairs the reader should refer to the PubSubHubBub specification (see http://code.google.com/p/pubsubhubbub/); however the key points are hopefully summarised in the following table:

Parameter	Description
`hub.callback`	The URL to which RabbitHub should post each message to as it arrives
`hub.topic`	A filter for selecting a subset of messages
`hub.verify`	The subscription verification mode for this request (the value may be either "`sync`" or "`async`"). Refer to the PubSubHubBub specification for additional details.
`hub.lease`	Subscriber-provided lease duration in seconds. After this time, the subscription will be terminated. The default lease is approximately 30 days, and the maximum lease is approximately 1000 years. Refer to the PubSubHubBub specification for additional information.

  When you subscribe to an exchange, a "pseudo queue" will be created for the subscription, and this queue will be deleted when the lease expires. Note that these pseudo queues tend to confuse the RabbitMQ management GUI, but otherwise cause no problems. Subscriptions to queues (as opposed to exchanges) do not create pseudo-queues; however subscriptions to queues facilitate only point-to-point messaging (a single message is sent to a single recipient). Subscriptions to fan-out or topic exchanges may be used to send single messages to multiple recipients

- **Subscribe to a queue**

  A `POST` request of the following format may be used to subscribe to a queue. The various query string parameters are as described previously.

  ```
  curl -vd \
  "hub.mode=subscribe&hub.callback=http://10.1.1.8:4567/sub2&hub.topic=foo&h
  ub.verify=sync&hub.lease_seconds=86400" \
  http://guest:guest@localhost:55670/subscribe/q/foo
  ```

As with subscribing to an exchange, upon receipt of the subscription request, RabbitHub will send a token to the specified callback URL, and the callback URL must respond with the same token in the body of the response. If the callback URL fails to respond correctly, the subscription will not be accepted.

- **Unsubscribe**

  Unsubscribing from an existing subscription can be achieved by issuing a request of the following format, replacing *<token>* with the token generated by the corresponding "subscribe" request. The binding and any other resources associated with the subscription will be deleted, and the subscription token will be invalidated.

```
curl -vd
"hub.mode=unsubscribe&hub.callback=http://10.1.1.8:4567/sub2&hub.topic=foo
&hub.verify=sync&<token>"
http://guest:guest@localhost:55670/subscribe/x/amq.direct
```

- **Send a message to an exchange**

  The following `POST` request will publish the message "`test message`" to the "`amq.direct`" exchange with routing key "`foo`". Note that from the perspective of the PubSubHubBub protocol, the routing key equates to the message topic.

```
curl -v -d "test message" \
http://guest:guest@localhost:55670/endpoint/x/amq.direct?hub.topic=foo
```

  When fanout or topic exchanges are used, all subscribers will be POSTed a copy of the message using their nominated callback URL's.

- **Send a message to a queue**

  The following `POST` request will publish the message "`another test message`" to a queue named "`foo`". This is achieved using the default exchange (the queue name is the routing key). Note that the query string "`hub.topic=rk`" is pretty much superfluous, as messages are published to the default exchange and the queue ("foo") is – like all other queues – bound to the default exchange.

```
$ curl -d "another test message" \
http://guest:guest@localhost:55670/endpoint/q/foo?hub.topic=rk
```

General format of RabbitHub URL's

You will notice that URL's are basically of two types:

1. Endpoint URL's for delivering messages (and issuing HTTP `PUT` and `DELETE` commands)
   - http://some.host.com:55670/endpoint/x/some_exchange_name
   - http://some.host.com:55670/endpoint/q/some_queue_name
2. Subscription URL's
   - http://some.host.com:55670/subscribe/x/some_exchange_name
   - http://some.host.com:55670/subscribe/q/some_queue_name