The Shovel

HP

Overview

- Allows to move messages reliably and continually from a queue (a *source*) in one broker to an exchange in another broker

- Can configure a number of *shovels*, which do just that and start automatically when the broker starts.

- Use RabbitMQ Erlang client

HP

Shovel plugin

- The primary advantages of a shovel are
  - Loose coupling (it is possible to move messages from an AMQP 1.0 broker to RabbitMQ or vice versa)
  - WAN-friendly
  - Highly tailorable

Shovel is a minimalistic yet flexible tool in the distributed messaging toolkit that can accommodate a number of use cases.

HP

The Shovel plugin

- defines (and runs) an Erlang client application for each shovel.
- A shovel is a simple pump.
- Each shovel:
  - *connects* to the source broker and the destination broker,
  - *consumes* messages from the queue,
  - *re-publishes* each message to the destination broker (using, by default, the original exchange name and routing_key).

HP

The shovel configuration

- The shovel configuration allows each of these processes to be tailored.
- There is no requirement to run the shovel on the same broker (or cluster) as its source or destination; the shovel can be entirely separate.

```
rabbitmq-plugins enable rabbitmq_shovel
```

- There are two distinct ways to define shovels:
  - static shovels
  - dynamic shovels

HP

The shovel configuration

Static Shovels

Defined in the broker configuration file.

Require a restart of the hosting broker to change.

Slightly more general: any queues, exchanges or bindings can be declared manually at startup.

Dynamic Shovels

Defined in the broker's parameters.

Can be created and deleted at any time.

Slightly more opinionated: the queues, exchanges and bindings used by the shovel will be declared automatically.

HP

Static Shovels

- an Erlang term and consists of a single shovels clause:

```
{rabbitmq_shovel, [ {shovels, [ {shovel_name, [ ... ]}, ... ]} ]}
```

```
{shovel_name, [ {sources, [ ... ]}
              , {destinations, [ ... ]}
              , {queue, queue_name}
              , {prefetch_count, count}
              , {ack_mode, a_mode}
              , {publish_properties, [ ... ]}
              , {publish_fields, [ ... ]}
              , {reconnect_delay, reconn_delay}
              ]}
```

Static Shovels

- Sources and destinations :
  - Both of these clauses are mandatory.

```
{brokers, broker_list}
```

```
[ "amqp://fred:secret@host1.domain/my_vhost"
, "amqp://john:secret@host2.domain/my_vhost"
]
```

the *declaration_list* is a list of AMQP methods (in the style of the Erlang client) which can be sent to the broker after connection and before shovelling.

```
{declarations, [ 'queue.declare'
               , {'queue.bind', [ {exchange, <<"my_exchange">>}
                                , {queue,    <<>>}
                                ]}
               ]}
```

Static Shovels

- queue : This clause is mandatory

```
{queue, <<"my_work_queue">>}
```

This queue must exist. Use the resource **declarations** to create the queue (or ensure it exists) first. If *queue_name* is <<>> (the empty binary string) the *most recently declared queue* in declarations is used. This allows anonymous queues to be declared and used.

HP

Example Configuration

```
{rabbitmq_shovel,
    [ {shovels, [ {my_first_shovel,
                    [ {sources,
                        [ {brokers, [ "amqp://fred:secret@host1.domain/my_vhost"
                                    , "amqp://john:secret@host2.domain/my_vhost"
                                    ]}
                        , {declarations, [ {'exchange.declare',
                                            [ {exchange, <<"my_fanout">>}
                                            , {type, <<"fanout">>}
                                            , durable
                                            ]}
                                         , {'queue.declare',
                                            [{arguments,
                                               [{<<"x-message-ttl">>, long, 60000}]}]}
                                         , {'queue.bind',
                                            [ {exchange, <<"my_direct">>}
                                            , {queue,    <<>>}
                                            ]}
                                         ]}
                        ]}
```

HP

Example Configuration...

```
    , {destinations,
        [ {broker, "amqp://"}
        , {declarations, [ {'exchange.declare',
                            [ {exchange, <<"my_direct">>}
                            , {type, <<"direct">>}
                            , durable
                            ]}
                         ]}
        ]}
    , {queue, <<>>}
    , {prefetch_count, 10}
    , {ack_mode, on_confirm}
    , {publish_properties, [ {delivery_mode, 2} ]}
    , {add_forward_headers, true}
    , {publish_fields, [ {exchange, <<"my_direct">>}
                       , {routing_key, <<"from_shovel">>}
                       ]}
    , {reconnect_delay, 5}
    ]}
]}
```

HP

Dynamic Shovels

- Information about dynamic shovels is stored in the RabbitMQ database, along with users, permissions, queues, etc.
- Every shovel is defined by a corresponding named **parameter**.

Example - Dynamic

rabbitmqctl	rabbitmqctl set_parameter shovel my-shovel \ '{"src-uri": "amqp://", "src-queue": "my-queue", \ "dest-uri": "amqp://remote-server", "dest-queue": "another-queue"}'
rabbitmqctl (Windows)	rabbitmqctl set_parameter shovel my-shovel ^ "{""src-uri"":""amqp://"", ""src-queue"": ""my-queue"", ^ ""dest-uri"": ""amqp://remote-server"", ""dest-queue"": ""another-queue""}"
HTTP API	PUT /api/parameters/shovel/%2f/my-shovel {"value":{"src-uri": "amqp://", "src-queue": "my-queue", "dest-uri": "amqp://remote-server", "dest-queue": "another-queue"}}
Web UI	Navigate to Admin > Shovel Management > Add a new shovel. Enter "my-shovel" next to Name, "amqp://" and "my-queue" next to Source, and "amqp://remote-server" and "another-queue" next Destination Expiry. Click Add shovel.

HP

Shovels between clusters

- Shovels can fail over to different nodes in the source or destination clusters
- Dynamic shovels are automatically defined on all nodes of the hosting cluster on which the shovel plugin is enabled.
- Static shovels should be defined in the configuration file for all nodes of the hosting cluster on which the shovel plugin is enabled

HP

Monitoring shovels

- two ways of discovering the status of shovels
  - Use shovel management
  - Direct query

HP

Use shovel management

- Shovel status reported on the **Management plugin** user interface
  - enable the rabbitmq_shovel_management plugin
- information about configured shovels will automatically appear in the management API and UI.

HP

Direct query

```
$ rabbitmqctl eval 'rabbit_shovel_status:status().'
```

- *Name* is the shovel name,
- *Type* is either static or dynamic,
- *Status* is the current shovel state,and
- *Timestamp* is the time when the shovel *entered* this state.

Lab : Shovel