Distributed RabbitMQ brokers

AMQP and the other messaging protocols supported by RabbitMQ via plug-ins (e.g. STOMP), are (of course) inherently distributed - it is quite common for applications from multiple machines to connect to a single broker, even across the internet.

Sometimes however it is necessary or desirable to make the RabbitMQ broker itself distributed. There are three ways in which to accomplish that: with clustering, with federation, and using the shovel. This page explains the purpose of each approach.

Note that you do not need to pick a single approach - you can connect clusters together with federation, or the shovel, or both.

Clustering

Clustering connects multiple machines together to form a single logical broker. Communication is via Erlang message-passing, so all nodes in the cluster must have the same Erlang cookie. The network links between machines in a cluster must be reliable, and all machines in the cluster must run the same versions of RabbitMQ and Erlang.

Virtual hosts, exchanges, users, and permissions are automatically mirrored across all nodes in a cluster. Queues may be located on a single node, or **mirrored across multiple nodes**. A client connecting to any node in a cluster can see all queues in the cluster, even if they are not located on that node.

Typically you would use clustering for high availability and increased throughput, with machines in a single location.

Federation

Federation allows an exchange or queue on one broker to receive messages published to an exchange or queue on another (the brokers may be individual machines, or clusters). Communication is via AMQP (with optional SSL), so for two exchanges or queues to federate they must be granted appropriate users and permissions.

Federated exchanges are connected with one way point-to-point links. By default, messages will only be forwarded over a federation link once, but this can be increased to allow for more complex routing topologies. Some messages may not be forwarded over the link; if a message would not be routed to a queue after reaching the federated exchange, it will not be forwarded in the first place.

Federated queues are similarly connected with one way point-to-point links. Messages will be moved between federated queues an arbitrary number of times to follow the consumers.

Typically you would use federation to link brokers across the internet for pub/sub messaging and work queueing.

The Shovel

Connecting brokers with **the shovel** is conceptually similar to connecting them with federation. However, the shovel works at a lower level.

Whereas federation aims to provide opinionated distribution of exchanges and queues, the shovel simply consumes messages from a queue on one broker, and forwards them to an exchange on another.

Typically you would use the shovel to link brokers across the internet when you need more control than federation provides.

Dynamic shovels can also be useful for moving messages around in an ad-hoc manner on a single broker.

Summary

Federation / Shovel	Clustering
Brokers are logically separate and may have different owners.	A cluster forms a single logical broker.
Brokers can run different versions of RabbitMQ and Erlang.	Nodes must run the same version of RabbitMQ, and frequently Erlang.
Brokers can be connected via unreliable WAN links. Communication is via AMQP (optionally secured by SSL), requiring appropriate users and permissions to be set up.	Brokers must be connected via reliable LAN links. Communication is via Erlang internode messaging, requiring a shared Erlang cookie.
Brokers can be connected in whatever topology you arrange. Links can be one- or two-way.	All nodes connect to all other nodes in both directions.
Chooses Availability and Partition Tolerance from the **CAP theorem**.	Chooses Consistency and Availability (or optionally **Consistency and Partition Tolerance**) from the CAP theorem.
Some exchanges in a broker may be federated while some may be local.	Clustering is all-or-nothing.
A client connecting to any broker can only see queues in that broker.	A client connecting to any node can see queues on all nodes.

Copyright © 2014 Pivotal Software, Inc. All rights reserved | Privacy Policy