HP

Distributed RabbitMQ brokers

HP

Distributed RabbitMQ brokers

- Ways to make the RabbitMQ broker distributed:
  - Clustering
  - Federation
  - Shovel.

HP

Clustering

- Connects multiple machines together to form a single logical broker.
- Via Erlang message-passing
- All nodes in the cluster must have the same Erlang cookie.
- The network links between machines in a cluster **must** be reliable
- All machines in the cluster must run the same versions of RabbitMQ and Erlang.

HP

Clustering

- Virtual hosts, exchanges, users, and permissions are automatically mirrored.
- A client can connect to any node in a cluster .
- Use for high availability and increased throughput in a single location

HP

Federation

- Use to link brokers across the internet for pub/sub messaging and work queueing.
- Allows an exchange or queue on one broker to receive messages published to an exchange or queue on another.
- Types:
  - Federated exchanges
  - Federated queues

HP

The Shovel

- Similar to connecting brokers with federation
- Federation aims to provide opinionated distribution
- Simply consumes messages from a queue on one broker
  - forwards them to an exchange on another.
- To link brokers across the internet when you need more control than federation provides.

The Shovel

- Types:
  - Static
  - Dynamic

HP

Summary - DA

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.

Summary - DA

Federation / Shovel	Clustering
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 (AP) from the CAP theorem.	Chooses Consistency and Partition Tolerance (CP) 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.

Clustering

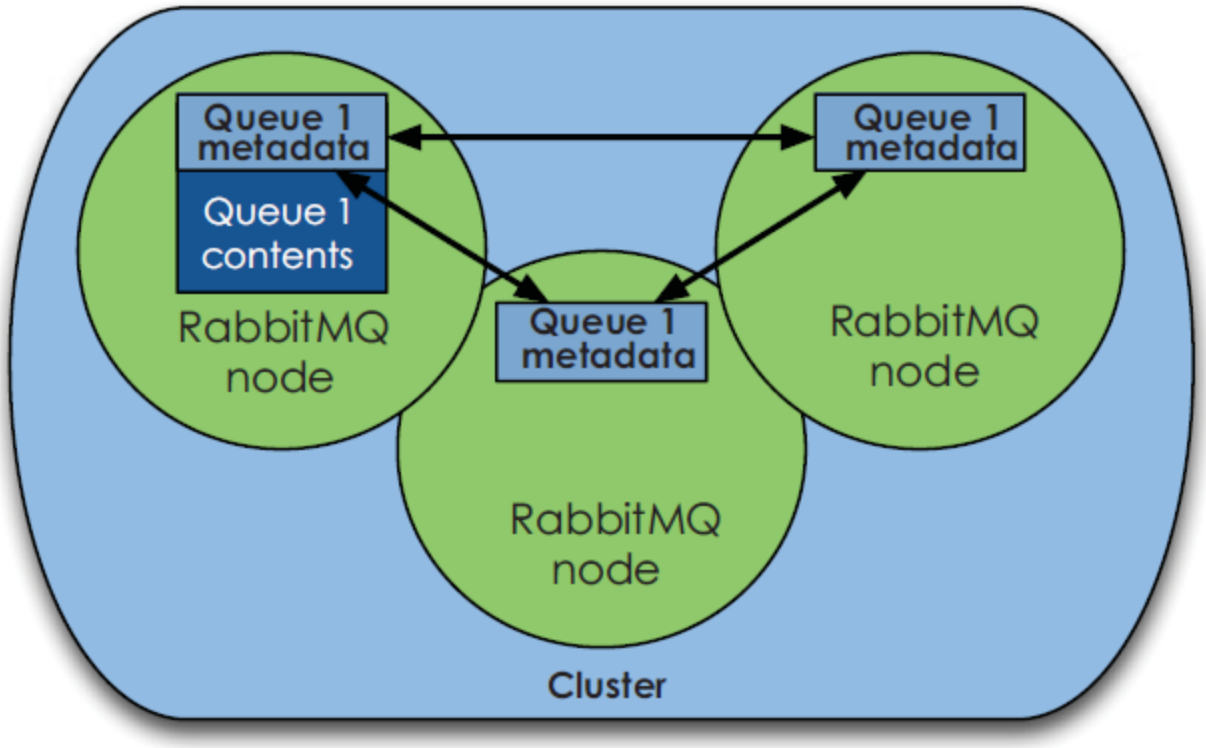
Clustering - RabbitMQ

- A RabbitMQ *broker* :

  A logical grouping of one or several Erlang *nodes*,
  - each running the RabbitMQ *application*
  - sharing users, virtual hosts, queues, exchanges, etc.
- Collection of such nodes → *cluster*.
- All data/state required is replicated across all nodes
- Provides reliability and scaling, with full ACID properties.
- Message queues, which by default reside on the node that created them
  - they are visible and reachable from all nodes.

HP

Clustering - RabbitMQ



HP

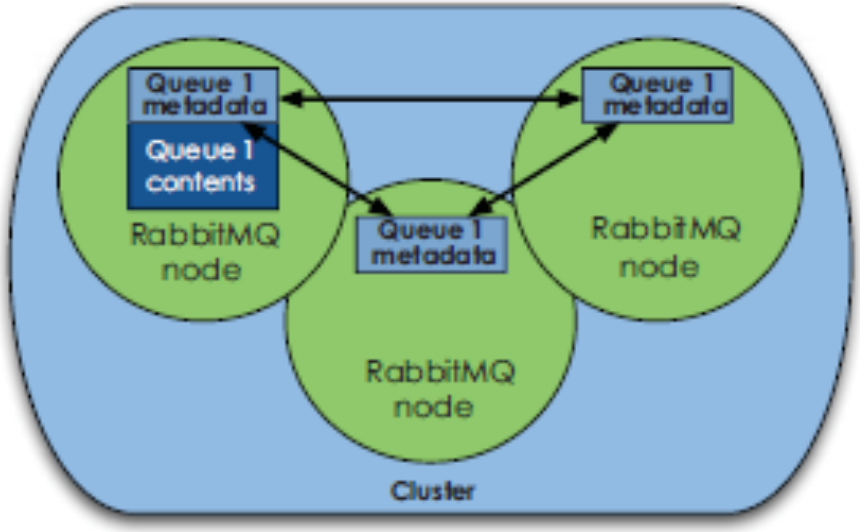
Clustering - RabbitMQ

- **does not tolerate network partitions well**, so it should not be used over a WAN.
- Better solutions for wan :
  The **shovel** or **federation** plugins
- The composition of a cluster can be altered dynamically.

HP

Clustering - RabbitMQ

- All RabbitMQ brokers start out as running on a single node.
- nodes can be joined into clusters, and subsequently turned back into individual brokers again.
- tolerate the failure of individual nodes.
- Nodes can be started and stopped at will



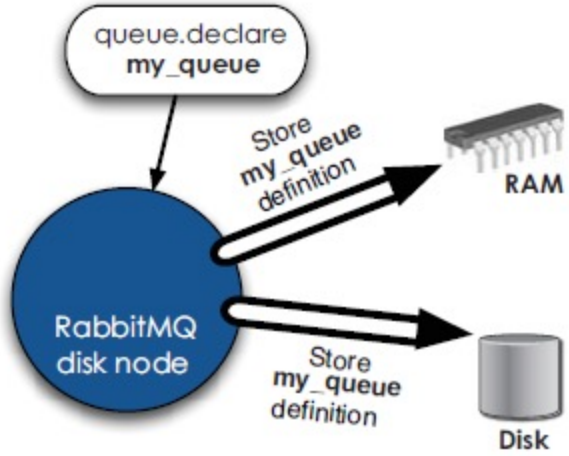
HP

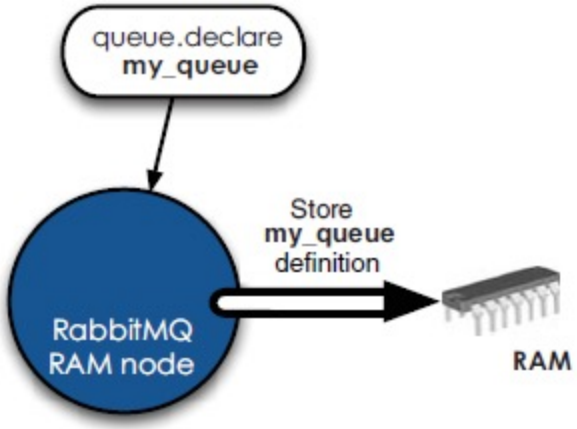
Clustering - RabbitMQ

A node can be :

a *disk node*

or a *RAM node*.





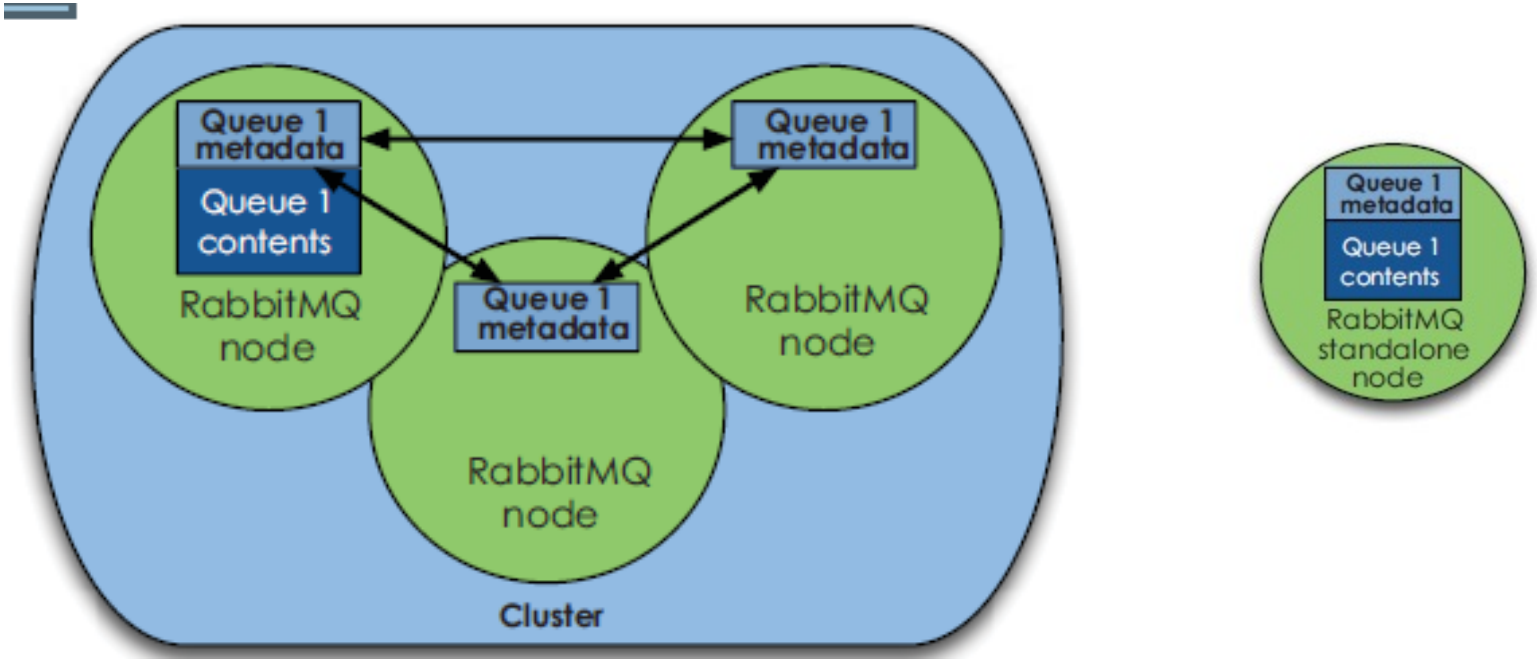
Cluster Configuration

- Cluster can formed in a number of ways:
  - Manually with rabbitmqctl
  - Declaratively by listing cluster nodes in **config file**
  - Declaratively with **rabbitmq-autocluster** (a plugin)
  - Declaratively with **rabbitmq-clusterer** (a plugin)
- The composition of a cluster can be altered dynamically.
- All RabbitMQ brokers start out as running on a single node.
  - These nodes can be joined into clusters, and subsequently turned back into individual brokers again.

HP

Clustering transcript

- Queue behavior in standalone and cluster configurations



HP

Clustering transcript

Erlang cookie

On Unix systems, the cookie will be typically located in

/var/lib/rabbitmq/.erlang.cookie or$HOME/.erlang.cookie.

On Windows, the locations are

C:\Users*Current User*\.erlang.cookie(%HOMEDRIVE% + %HOMEPATH%\.erlang.cookie) or C:\Documents and Settings*Current User*\.erlang.cookie, and C:\Windows\.erlang.cookie for RabbitMQ Windows service.

If Windows service is used, the cookie should be placed in both places.

As an alternative, you can insert the option "-setcookie *cookie*" in the erl call in the

rabbitmq-server and rabbitmqctl scripts.

HP Cluster- Starting independent nodes

Start RabbitMQ on all nodes in the normal way:

```
rabbit1$ rabbitmq-server -detached
rabbit2$ rabbitmq-server -detached
rabbit3$ rabbitmq-server -detached
```

Creates three *independent* RabbitMQ brokers

```
rabbit1$ rabbitmqctl cluster_status
Cluster status of node rabbit@rabbit1 ...
[{nodes,[{disc,[rabbit@rabbit1]}]},{running_nodes,[rabbit@rabbit1]}]
...done.
rabbit2$ rabbitmqctl cluster_status
Cluster status of node rabbit@rabbit2 ...
[{nodes,[{disc,[rabbit@rabbit2]}]},{running_nodes,[rabbit@rabbit2]}]
...done.
rabbit3$ rabbitmqctl cluster_status
Cluster status of node rabbit@rabbit3 ...
[{nodes,[{disc,[rabbit@rabbit3]}]},{running_nodes,[rabbit@rabbit3]}]
...done.
```

HP

Creating the cluster

```
rabbit2$ rabbitmqctl stop_app
Stopping node rabbit@rabbit2 ...done.
rabbit2$ rabbitmqctl join_cluster --ram rabbit@rabbit1
Clustering node rabbit@rabbit2 with [rabbit@rabbit1] ...done.
rabbit2$ rabbitmqctl start_app
Starting node rabbit@rabbit2 ...done.
```

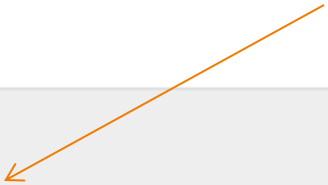
HP

Creating the cluster..

Verify the cluster status

```
rabbit1$ rabbitmqctl cluster_status
Cluster status of node rabbit@rabbit1 ...
[{nodes,[{disc,[rabbit@rabbit1]},{ram,[rabbit@rabbit2]}]},
 {running_nodes,[rabbit@rabbit2,rabbit@rabbit1]}]
...done.
rabbit2$ rabbitmqctl cluster_status
Cluster status of node rabbit@rabbit2 ...
[{nodes,[{disc,[rabbit@rabbit1]},{ram,[rabbit@rabbit2]}]},
 {running_nodes,[rabbit@rabbit1,rabbit@rabbit2]}]
...done.
```

rabbit@rabbit3 as a disk node



```
rabbit3$ rabbitmqctl stop_app
Stopping node rabbit@rabbit3 ...done.
rabbit3$ rabbitmqctl join_cluster rabbit@rabbit2
Clustering node rabbit@rabbit3 with rabbit@rabbit2 ...done.
rabbit3$ rabbitmqctl start_app
Starting node rabbit@rabbit3 ...done.
```

HP

Creating the cluster..

Verify the cluster by running the *cluster_status* command on any of the nodes:

```
rabbit1$ rabbitmqctl cluster_status
Cluster status of node rabbit@rabbit1 ...
[{nodes,[{disc,[rabbit@rabbit1,rabbit@rabbit3]},{ram,[rabbit@rabbit2]}]},
 {running_nodes,[rabbit@rabbit3,rabbit@rabbit2,rabbit@rabbit1]}]
...done.
rabbit2$ rabbitmqctl cluster_status
Cluster status of node rabbit@rabbit2 ...
[{nodes,[{disc,[rabbit@rabbit1,rabbit@rabbit3]},{ram,[rabbit@rabbit2]}]},
 {running_nodes,[rabbit@rabbit3,rabbit@rabbit1,rabbit@rabbit2]}]
...done.
rabbit3$ rabbitmqctl cluster_status
Cluster status of node rabbit@rabbit3 ...
[{nodes,[{disc,[rabbit@rabbit3,rabbit@rabbit1]},{ram,[rabbit@rabbit2]}]},
 {running_nodes,[rabbit@rabbit2,rabbit@rabbit1,rabbit@rabbit3]}]
...done.
```

HP

Changing node types

change the type of a node from ram to disk and vice versa.

```
rabbit2$ rabbitmqctl stop_app
Stopping node rabbit@rabbit2 ...done.
rabbit2$ rabbitmqctl change_cluster_node_type disc
Turning rabbit@rabbit2 into a disc node ...
...done.
Starting node rabbit@rabbit2 ...done.
rabbit3$ rabbitmqctl stop_app
Stopping node rabbit@rabbit3 ...done.
rabbit3$ rabbitmqctl change_cluster_node_type ram
Turning rabbit@rabbit3 into a ram node ...
rabbit3$ rabbitmqctl start_app
Starting node rabbit@rabbit3 ...done.
```

HP

Breaking up a cluster

```
rabbit3$ rabbitmqctl stop_app
Stopping node rabbit@rabbit3 ...done.
rabbit3$ rabbitmqctl reset
Resetting node rabbit@rabbit3 ...done.
rabbit3$ rabbitmqctl start_app
Starting node rabbit@rabbit3 ...done.
```

```
rabbit1$ rabbitmqctl cluster_status
Cluster status of node rabbit@rabbit1 ...
[{nodes,[{disc,[rabbit@rabbit1,rabbit@rabbit2]}]},
 {running_nodes,[rabbit@rabbit2,rabbit@rabbit1]}]
...done.
rabbit2$ rabbitmqctl cluster_status
Cluster status of node rabbit@rabbit2 ...
[{nodes,[{disc,[rabbit@rabbit1,rabbit@rabbit2]}]},
 {running_nodes,[rabbit@rabbit1,rabbit@rabbit2]}]
...done.
rabbit3$ rabbitmqctl cluster_status
Cluster status of node rabbit@rabbit3 ...
[{nodes,[{disc,[rabbit@rabbit3]}]},{running_nodes,[rabbit@rabbit3]}]
...done.
```

Lab - Clustering RabbitMQ