How to Manually Add Rabbitmq Cluster

Contents

Introduction

This document describes how to manually add RabbitMQ to a cluster if the cluster is broken.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

The information in this document is based on these software and hardware versions:

- Minimum 2 RabbitMQ server
- One Load Balancer

Problem

This article guides you on how to verify the RabbitMQ cluster and manually add those instance to the cluster. CloudCenter provides a wizard to configure High Availability (HA) for RabbitMQ however, in quite a few instance it says that the HA is successfully configured after it exits the wizard but the RabbitMQ cluster is not formed properly.

Verify if RabbitMQ is Out of Cluster

Step 1. Log in to all RabbitMQ server with the use of CLI console.

Step 2. Verify if RabbitMQ server runs on all the instances.

```
#ps -ef | grep rabbit
```

```
Output:
```

```
rabbitmq  1677     1  0 14:47 ?        00:00:00 /usr/lib/erlang/erts-6.4/bin/epmd -daemon
root      1973     1  0 14:47 ?        00:00:00 /bin/sh /etc/rc.d/init.d/rabbitmq-server start
root      2000  1973  0 14:47 ?        00:00:00 /bin/bash -c ulimit -S -c 0 >/dev/null 2>&1 ;
/usr/sbin/rabbitmq-server
root      2003  2000  0 14:47 ?        00:00:00 su rabbitmq -s /bin/sh -c
/usr/lib/rabbitmq/bin/rabbitmq-server
rabbitmq  2026  2003  0 14:47 ?        00:00:05 /usr/lib/erlang/erts-6.4/bin/beam -W w -K true -
A30 -P 1048576 -- -root /usr/lib/erlang -progname erl -- -home /var/lib/rabbitmq -- -pa
/usr/lib/rabbitmq/lib/rabbitmq_server-3.5.1/sbin/../ebin -noshell -noinput -s rabbit boot -sname
rabbit@ip-172-31-32-101 -boot start_sasl -config /etc/rabbitmq/rabbitmq -kernel
inet_default_connect_options [{nodelay,true}] -sasl errlog_type error -sasl sasl_error_logger
false -rabbit error_logger {file,"/var/log/rabbitmq/rabbit@ip-172-31-32-101.log"} -rabbit
sasl_error_logger {file,"/var/log/rabbitmq/rabbit@ip-172-31-32-101-sasl.log"} -rabbit
enabled_plugins_file "/etc/rabbitmq/enabled_plugins" -rabbit plugins_dir
"/usr/lib/rabbitmq/lib/rabbitmq_server-3.5.1/sbin/../plugins" -rabbit plugins_expand_dir
"/var/lib/rabbitmq/mnesia/rabbit@ip-172-31-32-101-plugins-expand" -os_mon start_cpu_sup false -
os_mon start_disksup false -os_mon start_memsup false -mnesia dir
"/var/lib/rabbitmq/mnesia/rabbit@ip-172-31-32-101" -kernel inet_dist_listen_min 25672 -kernel
inet_dist_listen_max 25672
rabbitmq  2242  2026  0 14:47 ?        00:00:00 inet_gethost 4
rabbitmq  2243  2242  0 14:47 ?        00:00:00 inet_gethost 4
root      2602  2588  0 15:04 pts/0    00:00:00 grep --color=auto rabbit
OR
#/sbin/service rabbitmq-server status

Status of node 'rabbit@ip-172-31-32-101' ...
[{pid,2026},
 {running_applications,
    [{rabbitmq_management,"RabbitMQ Management Console","3.5.1"},
     {rabbitmq_web_dispatch,"RabbitMQ Web Dispatcher","3.5.1"},
     {webmachine,"webmachine","1.10.3-rmq3.5.1-gite9359c7"},
     {mochiweb,"MochiMedia Web Server","2.7.0-rmq3.5.1-git680dba8"},
     {rabbitmq_management_agent,"RabbitMQ Management Agent","3.5.1"},
     {rabbit,"RabbitMQ","3.5.1"},

----- Text omitted for brevity
```

Step 3. Verify the cluster status of all the instance with these commands:

```
[root@ip-172-31-32-101 ~]# rabbitmqctl cluster_status
Cluster status of node 'rabbit@ip-172-31-32-101' ...
[{nodes,[{disc,['rabbit@ip-172-31-32-101']}]},
 {running_nodes,['rabbit@ip-172-31-32-101']},
 {cluster_name,<<"rabbit@ip-172-31-32-101.us-east-2.compute.internal">>},
 {partitions,[]}]
```

In this output, you can identify that there is only one node that runs in the cluster.

Solution

In this scenario, you add **rabbit@ip-172-31-32-101** to your cluster **rabbit@ip-172-31-45-110.us-east-2.compute.internal**

Step 1. In order to link the second RabbitMQ server, you need to ensure that the application is stopped and joins the cluster.

Step 2. Switch to RabbitMQ2 server and stop the application.

```
[root@ip-172-31-32-101 ~]# rabbitmqctl cluster_status
```

```
Cluster status of node 'rabbit@ip-172-31-32-101' ...
[{nodes,[{disc,['rabbit@ip-172-31-32-101']}]},
 {running_nodes,['rabbit@ip-172-31-32-101']},
 {cluster_name,<<"rabbit@ip-172-31-32-101.us-east-2.compute.internal">>},
 {partitions,[]}]
```

Step 3. Join the RabbitMQ2 server to the RabbitMQ1 cluster.

```
[root@ip-172-31-32-101 ~]# rabbitmqctl cluster_status
Cluster status of node 'rabbit@ip-172-31-32-101' ...
[{nodes,[{disc,['rabbit@ip-172-31-32-101']}]},
 {running_nodes,['rabbit@ip-172-31-32-101']},
 {cluster_name,<<"rabbit@ip-172-31-32-101.us-east-2.compute.internal">>},
 {partitions,[]}]
```

Step 4. Start the rabbitmq2 application.

```
[root@ip-172-31-32-101 ~]# rabbitmqctl cluster_status
Cluster status of node 'rabbit@ip-172-31-32-101' ...
[{nodes,[{disc,['rabbit@ip-172-31-32-101']}]},
 {running_nodes,['rabbit@ip-172-31-32-101']},
 {cluster_name,<<"rabbit@ip-172-31-32-101.us-east-2.compute.internal">>},
 {partitions,[]}]
```

You can see that the two nodes are joined in a cluster when you run the **cluster_status** command on either of the nodes.

```
[root@ip-172-31-32-101 ~]# rabbitmqctl cluster_status
Cluster status of node 'rabbit@ip-172-31-32-101' ...
[{nodes,[{disc,['rabbit@ip-172-31-32-101']}]},
 {running_nodes,['rabbit@ip-172-31-32-101']},
 {cluster_name,<<"rabbit@ip-172-31-32-101.us-east-2.compute.internal">>},
 {partitions,[]}]
```