**Phần 1: Cài đặt và cấu hình RabbitMQ on Centos 6**

Website tham khảo cài đặt : <https://www.rabbitmq.com/install-rpm.html>

<https://gist.github.com/ravibhure/92e780ecc850cd5ab0ab>

https://kipalog.com/posts/Tim-hieu-RabbitMQ---Phan-1

Server1 : 192.168.254.1 – Hostname : rabbitmq01

Server2 : 192.168.254.2 – Hostname : rabbitmq02

Hướng dẫn cài RabbitMQ offline:

Trước khi cài đặt nên đặt hostname 2 server như yêu cầu là rabbitmq01 và rabbitmq02

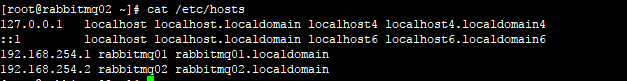
Chỉnh hostname có thể phân giải được tên miền trong file /etc/hosts bằng cách

# vi /etc/hosts

Thêm 2 dòng sau vào file host

192.168.254.1 rabbitmq01 rabbitmq01.localdomain

192.168.254.2 rabbitmq02 rabbitmq02.localdomain



Bước 1 : Chuẩn bị các gói cần thiết để cài đặt

* Download gói [rabbitmq-server-3.6.10-1.el6.noarch.rpm](https://bintray.com/rabbitmq/rabbitmq-server-rpm/download_file?file_path=rabbitmq-server-3.6.10-1.el6.noarch.rpm)

<https://bintray.com/rabbitmq/rabbitmq-server-rpm/download_file?file_path=rabbitmq-server-3.6.10-1.el6.noarch.rpm>

* Download esl-erlang\_16.b.3-1-centos-6\_amd64.rpm (RB16B03)

<https://packages.erlang-solutions.com/erlang/esl-erlang/FLAVOUR_1_general/esl-erlang_16.b.3-1~centos~6_amd64.rpm>

* Download rabbitmq-release-signing-key.asc

<https://bintray.com/user/downloadSubjectPublicKey?username=rabbitmq>

* Download epel-release-6-8.noarch.rpm

<http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm>

* Download remi-release-6.rpm

<http://rpms.famillecollet.com/enterprise/remi-release-6.rpm>

* Các gói phụ thuộc

+ unixODBC-2.2.14-14.el6.x86\_64.rpm

<http://rpm.pbone.net/index.php3/stat/4/idpl/27828741/dir/centos_6/com/unixODBC-2.2.14-14.el6.x86_64.rpm.html>

+ wxBase-2.8.12-1.el6.centos.x86\_64.rpm

<https://centos.pkgs.org/6/epel-x86_64/wxBase-2.8.12-1.el6.x86_64.rpm.html>

+ wxGTK-2.8.12-1.el6.centos.x86\_64.rpm

<https://centos.pkgs.org/6/centos-extras-x86_64/wxGTK-2.8.12-1.el6.centos.x86_64.rpm.html>

+ wxGTK-gl-2.8.12-1.el6.centos.x86\_64.rpm

<https://centos.pkgs.org/6/centos-extras-x86_64/wxGTK-gl-2.8.12-1.el6.centos.x86_64.rpm.html>

Bước 2 : Tiến hành cài đặt Erlang/OTP:

* Trước khi cài đặt RabbitMQ phải cài đặt Erlang/OTP, có 2 cách cài Erlang :

Cách 1 : Cài đặt qua EPEL : ( Cách này cần internet)

1. Download the EPEL RPM.

#wget <http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm>

#wget <http://rpms.famillecollet.com/enterprise/remi-release-6.rpm>

1. Install the EPEL RPM.

#rpm -ivh epel-release-6\*.rpm

#rpm –ivh remi-release-6.rpm

1. Install Erlang via "yum"

#yum install erlang

Cách 2 : Cài đặt qua Erlang Solution RPM ( cách này cài offline)

1. Import Erlang Solutions Certificate

#rpm --import <http://binaries.erlang-solutions.com/debian/erlang_solutions.asc> (lệnh này cần internet )

Or

#Wget <http://binaries.erlang-solutions.com/debian/erlang_solutions.asc>

#Gpg --import erlang\_solutions.asc



1. Cài đặt các gói đi kèm

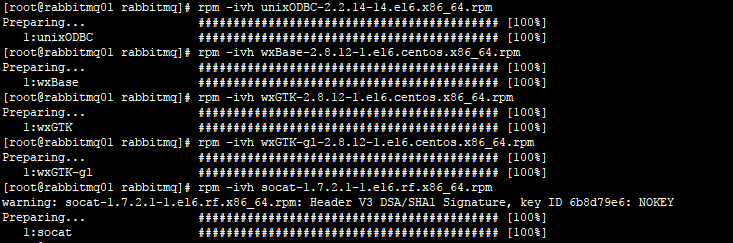
#rpm –ivh unixODBC-2.2.14-14.el6.x86\_64.rpm

#rpm –ivh wxBase-2.8.12-1.el6.centos.x86\_64.rpm

#rpm –ivh wxGTK-2.8.12-1.el6.centos.x86\_64.rpm

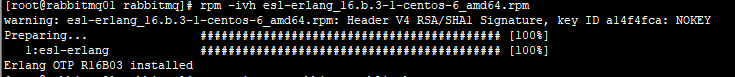
#rpm –ivh wxGTK-gl-2.8.12-1.el6.centos.x86\_64.rpm

#rpm –ivh socat-1.7.2.1-1.el6.rf.x86\_64.rpm



1. Cài đặt Erlang OTP R16B03

#rpm –ivh esl-erlang\_16.b.3-1-centos-6\_amd64.rpm



Bước 3 : Cài đặt RabbitMQ : [rabbitmq-server-3.6.10-1.el6.noarch.rpm](https://bintray.com/rabbitmq/rabbitmq-server-rpm/download_file?file_path=rabbitmq-server-3.6.10-1.el6.noarch.rpm)

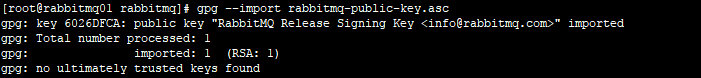
**Nếu Erlang RPM cài từ EPEL thì gõ lệnh này**

#yum install rabbitmq-server-3.0.1-1.noarch.rpm

### Nếu Erlang RPM cài từ Erlang Solutions thì làm theo cách này

* Import rabbitmq-public-key.asc

#gpg --import rabbitmq-public-key.asc



* Cài đặt [rabbitmq-server-3.6.10-1.el6.noarch.rpm](https://bintray.com/rabbitmq/rabbitmq-server-rpm/download_file?file_path=rabbitmq-server-3.6.10-1.el6.noarch.rpm)

#rpm --nodeps –ivh rabbitmq-server-3.6.10-1.el6.noarch.rpm

Bước 4 : Start rabbitmq :

#service rabbitmq-server start



Cấu hình rabbit khởi động cùng windows

#chkconfig rabbitmq-server on

Bước 5 : Mở port firewall nếu có firewall chạn các port sau:

* 4369: [epmd](http://erlang.org/doc/man/epmd.html), a peer discovery service used by RabbitMQ nodes and CLI tools
* 5672, 5671: used by AMQP 0-9-1 and 1.0 clients without and with TLS
* 25672: used by Erlang distribution for inter-node and CLI tools communication and is allocated from a dynamic range (limited to a single port by default, computed as AMQP port + 20000). See [networking guide](https://www.rabbitmq.com/networking.html) for details.
* 15672: [HTTP API](https://www.rabbitmq.com/management.html) clients and [rabbitmqadmin](https://www.rabbitmq.com/management-cli.html) (only if the [management plugin](https://www.rabbitmq.com/management.html) is enabled)
* 61613, 61614: [STOMP clients](https://stomp.github.io/stomp-specification-1.2.html) without and with TLS (only if the [STOMP plugin](https://www.rabbitmq.com/stomp.html) is enabled)
* 1883, 8883: ([MQTT clients](http://mqtt.org/) without and with TLS, if the [MQTT plugin](https://www.rabbitmq.com/mqtt.html) is enabled
* 15674: STOMP-over-WebSockets clients (only if the [Web STOMP plugin](https://www.rabbitmq.com/web-stomp.html) is enabled)
* 15675: MQTT-over-WebSockets clients (only if the [Web MQTT plugin](https://www.rabbitmq.com/web-mqtt.html) is enabled)

Dùng lệnh “vi” để thêm các port cần mở như trên vào iptables

# vi /etc/sysconfig/iptables

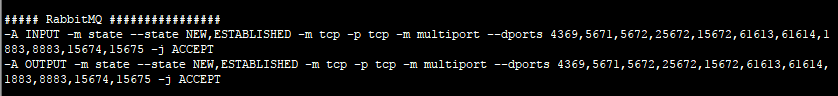
Thêm đoạn sao vào file iptables

# vi /etc/sysconfig/iptables

##### RabbitMQ ################

-A INPUT -m state --state NEW,ESTABLISHED -m tcp -p tcp -m multiport –dports 4369,5671,5672,25672,15672,61613,61614,1883,8883,15674,15675 -j ACCEPT

-A OUTPUT -m state --state NEW,ESTABLISHED -m tcp -p tcp -m multiport --dports 4369,5671,5672,25672,15672,61613,61614,1883,8883,15674,15675 -j ACCEPT



Bước 6 : tạo user admin cho đăng nhập Web Management Console

$ sudo rabbitmqctl add\_user admin password

$ sudo rabbitmqctl set\_user\_tags admin administrator

$ sudo rabbitmqctl set\_permissions -p / admin ".\*" ".\*" ".\*"

Để enable web management của RabbitMQ gõ lệnh như sau :

*Tham khảo : https://www.rabbitmq.com/management.html*

The management plugin is included in the RabbitMQ distribution. To enable it, use [rabbitmq-plugins](man/rabbitmq-plugins.1.man.html):

#rabbitmq-plugins enable rabbitmq\_management

* The Web UI is located at: http://*server-name*:15672/ or *localhost:15672*

Mặc định user/pass là : guest/guest

* The Web UI uses an HTTP API provided by the same plugin. Said API's documentation can be accessed at http://*server-host*:15672/api/ or our [latest HTTP API documentation here](https://rawcdn.githack.com/rabbitmq/rabbitmq-management/rabbitmq_v3_6_10/priv/www/api/index.html)).
* Download [rabbitmqadmin](management-cli.html) at: http://*server-name*:15672/cli/

NB: The port for RabbitMQ versions prior to 3.0 is 55672.

NOTE : tạo user rabbit cho một group

rabbitmqctl add\_user ivrmq ivrmq123

rabbitmqctl set\_user\_tags ivrmq administrator

rabbitmqctl add\_vhost IVR

rabbitmqctl set\_permissions -p IVR ivrmq ".\*" ".\*" ".\*"

**PHẦN 2 : CẤU HÌNH RABBITMQ CLUSTER AND HIGH AVAILABILITY (HA)**

**Tài liệu tham khảo :**

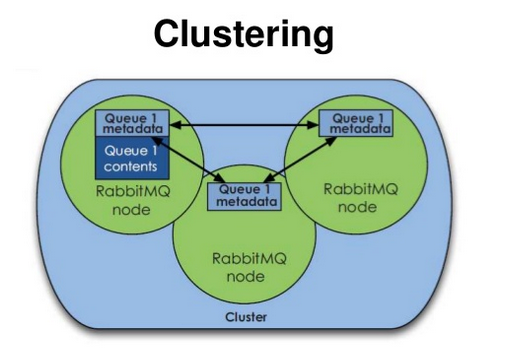
* [**https://www.rabbitmq.com/clustering.html**](https://www.rabbitmq.com/clustering.html)
* [**https://www.rabbitmq.com/ha.html**](https://www.rabbitmq.com/ha.html)
* [**https://kuntalchandra.wordpress.com/2016/03/21/rabbitmq-cluster-setup-and-mirrored-queue/**](https://kuntalchandra.wordpress.com/2016/03/21/rabbitmq-cluster-setup-and-mirrored-queue/)
* [**https://www.youtube.com/watch?v=X9dv-2TFeV8**](https://www.youtube.com/watch?v=X9dv-2TFeV8)
* [**https://kipalog.com/posts/Tim-hieu-RabbitMQ---Phan-2**](https://kipalog.com/posts/Tim-hieu-RabbitMQ---Phan-2)
* [**http://liemnhiep.blogspot.com/2016/09/rabbitmq.html**](http://liemnhiep.blogspot.com/2016/09/rabbitmq.html)

# Cluster là gì

Cluster là nhóm các thành phần mà hoạt động cùng với nhau để cung cấp một dịch vụ nào đó. Thành phần ở đây gọi là một node. Mỗi node này là một process hoạt động. Thường thì node được đồng nhất với một server do mỗi node thường được cài đặt trên một server riêng rẽ ( để tránh bị chết chùm ). Khái niệm cluster này xuất hiện trong rất nhiều kiến trúc như galera, mysql, redis... Mục đích sử dụng cluster là để load balancing, high availibility (đảm bảo hệ thống vẫn hoạt động khi có sự cố), scale hệ thống.

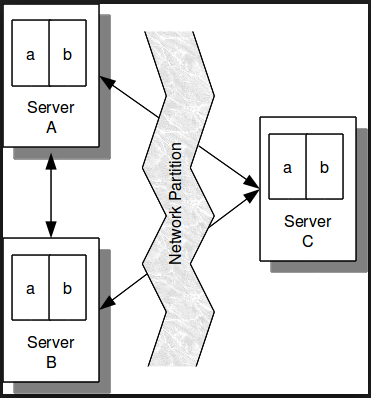
# Cluster trong rabbitmq

Trong rabbitmq, một cluster là một nhóm các erlang node làm việc cùng với nhau. Mỗi erlang node có một rabbitmq application hoạt động và cùng chia sẻ tài nguyên: user, vhost, queue, exchange...

[[](https://s3-ap-southeast-1.amazonaws.com/kipalog.com/Screenshot%20from%202015-06-01%2023:26:39.png_w0mj6sxzay)](https://s3-ap-southeast-1.amazonaws.com/kipalog.com/Screenshot%20from%202015-06-01%2023%3A26%3A39.png_w0mj6sxzay)

# Một số đặc điểm cần chú ý

* Metadata của một node được replicate đến các node còn lại trong cluster ngoại trừ queue. Queue được tạo ra trên node nào thì vẫn nằm trên node đó, không có replicate gì hết nhưng bạn hoàn toàn có thể nhìn thấy một queue tạo ra trên một node khi truy xuất qua các node còn lại do đó đối với client một cluster rabbitmq chẳng khác gì một single rabbitmq.
* Vì queue không được replicate nên bản thân cluster rabbitmq chưa cung cấp high availibility (HA). Bạn vẫn cần cấu hình thêm chút nữa nhưng cluster là tiền đề để rabbitmq có thể thực hiện được HA.
* Một node có thể là disc node (mặc định) hoặc ram node
* Như trong tài liệu của rabbitmq có khẳng định, rabbitmq không xử lý tốt network partition nên không khuyến khích sử dụng cluster của rabbitmq trên WAN. Có thể bạn sẽ thắc mắc network partition là gì ? Một network partition hay còn gọi là split brain là tình huống rất hay gặp trong hệ cluster.

[[](https://s3-ap-southeast-1.amazonaws.com/kipalog.com/Screenshot%20from%202015-06-01%2023:29:21.png_bwnuenk169)](https://s3-ap-southeast-1.amazonaws.com/kipalog.com/Screenshot%20from%202015-06-01%2023%3A29%3A21.png_bwnuenk169)

Khi network partition xảy ra, hệ cluster bị chia đôi và mỗi phần không thể liên lạc được với phần còn lại nên đâm ra bản thân từng partition lại cứ ngỡ nó là toàn bộ cluster. Vấn đề chính là ở đây. Mỗi partition khi đó sẽ không đồng bộ được với phần còn lại nên chúng ta sẽ có hai tập dữ liệu riêng biệt trên mỗi partition. Nguyên do dẫn đến network partition thường là network không ổn định hoặc quá tải trên node (Khi quá tải do CPU hoặc IO bị nghẽn network của server thường rất chập chờn ). LAN network so với WAN network thì thường ổn định hơn nhiều nên rabbitmq cluster thích hợp khi các node được liên kết với nhau qua LAN network. Về network partition, tôi sẽ trình bày trong một phần khác.

* Clustering rabbitmq chỉ là một trong ba cách cấu hình hệ phân tán rabbitmq. Clustering phù hợp cho môi trường LAN network còn với môi trường WAN network thì các mô hình như federation hay shovel lại được khuyến khích. Trong trường hợp của tôi, tôi không sử dụng rabbitmq trong môi trường WAN nên hai mô hình shovel và federation tôi không trình bày.

# Điều kiện để thiết lập clustering

* Tất cả các node phải cùng erlang version và rabbitmq version
* Các node liên kết qua LAN network
* Tất cả các node chia sẻ cùng một erlang cookie

Trong mô hình cluster, các node sử dụng phương thức trao đổi giữa của erlang. Khi sử dụng phương thức này, hai erlang node chỉ nói chuyện được với nhau khi có cùng erlang cookie. Erlang cookie chỉ là một chuỗi ký tự. Khi startup một rabbitmq server lần đầu tiên, mặc định một erlang cookie ngẫu nhiên được sinh ra nằm trong */var/lib/rabbitmq/.erlang.cookie*

**Thực hiện cấu hình Cluster RabbitMQ**

Đầu tiên, chúng ta cần chọn một node, sau đó copy erlang cookie của node đó sang các node còn lại.

Vì node name của rabbitmq có dạng rabbit@hostname nên bạn cần chuẩn bị sẵn hostname cho mỗi node. Hostname là bất cứ string nào bạn muốn nhưng tốt nhất đừng sử dụng các ký tự đặc biệt trong này. Sau đó đưa vào /etc/hosts của cả hai node:

192.168.254.1 rabbitmq01 rabbitmq01.localdomain *### gọi là node 1*

192.168.254.2 rabbitmq02 rabbitmq02.localdomain *### gọi là node 2*

**Copy erlang cookie từ rabbitmq01 sang rabbitmq02**  
[root@rabbitmq01 ~]#scp –r /var/lib/rabbitmq/.erlang.cookie 192.168.254.2:/var/lib/rabbitmq/

Sau đó nhập pass truy cập vào 192.168.254.2 để copy

Trên server 2 :

Set quyền ưu tiên :

[root@rabbitmq02 ~]# chown rabbitmq:rabbitmq /var/lib/rabbitmq/.erlang.cookie

[root@rabbitmq02 ~]# chmod 400 /var/lib/rabbitmq/.erlang.cookie

Xong reboot lại node2

**Bước 1:**  
Khởi động rabbitmq-server trên mỗi node.  
Trên mỗi node, bạn có thể chạy

service rabbitmq-server start  
hoặc  
rabbitmq-server –detached

rabbitmq01$ *rabbitmq-server -detached*

rabbitmq02$ *rabbitmq-server -detached*

Trong cách thứ hai, bạn có thể sẽ gặp một warning:  
**Warning: PID file not written; -detached was passed.**  
Đừng quá lo lắng. Trong manual của rabbitmq-server có cho biết khi chạy với tham số -detached server process sẽ hoạt động ở chế độ background và điều đó khiến cho pid của process không được ghi vào pid file.

**Bước 2:**  
Chọn một node làm khởi điểm sau đó các node còn lại sẽ join với node khởi điểm để hình thành lên cluster.

Trước khi thực hiện, chúng ta thử xem cluster status của từng node trước khi join với nhau. Trên mỗi node, bạn thực hiện lệnh  
rabbitmqctl cluster\_status

Và đây là kết quả:

Trên server1

[root@rabbitmq01 ~]# rabbitmqctl cluster\_status

Cluster status of node rabbit@rabbitmq01

[{nodes,[{disc,[rabbit@rabbitmq01]}]},

{running\_nodes,[ rabbit@rabbitmq01]},

{cluster\_name,<<"rabbit@rabbitmq01">>},

{partitions,[]},

{alarms,[rabbit@rabbitmq01,[]}]}]

Trên server2

[root@rabbitmq02 ~]# rabbitmqctl cluster\_status

Cluster status of node rabbit@rabbitmq02

[{nodes,[{disc,[rabbit@rabbitmq02]}]},

{running\_nodes,[ rabbit@rabbitmq02]},

{cluster\_name,<<"rabbit@rabbitmq02">>},

{partitions,[]},

{alarms,[rabbit@rabbitmq02,[]}]}]

Tạo cluster bằng cách join node 2 với node 1:

[root@rabbitmq02 ~]*# rabbitmqctl stop\_app*

Stopping node rabbit@rabbitmq02 ...

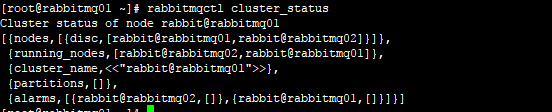
[root@rabbitmq02 ~]*# rabbitmqctl join\_cluster rabbit@rabbitmq01*

Clustering node rabbit@rabbitmq02 with rabbit@rabbitmq01 ...

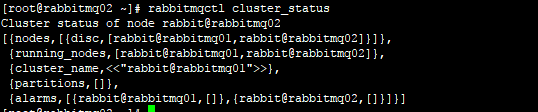
[root@rabbitmq02 ~]*# rabbitmqctl start\_app*

Starting node rabbit@rabbitmq02 ...

**Bước 3:**  
Xem cluster status trên server1



Xem cluster status trên server2



Như vậy đã thực hiện tạo xong Cluster RabbitMQ

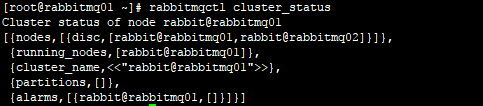
**Bước 4: thực hiện test cluster rabbitmq**

* Thực hiện stop rabbimq trên server 2 sau đó khởi động lại xem kết quả :

Trên server 2 : thực hiện stop rabbitmq



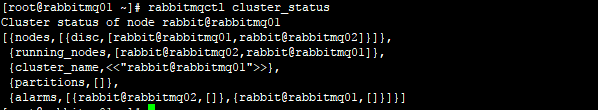
Xem cluster status trên server 1 đã thấy node 2 bị remove chỉ còn mỗi node 1 hoạt động



Trên server 2 : thực hiện start rabbitmq



Trên server 1: xem cluster\_status trên server 1 thấy node 2 đã quay trở lại



Như vậy, ngay sau khi được start trở lại, các node sẽ tự động tham gia vào cluster và running luôn.

Trong các trường hợp có sự cố nghiêm trọng như toàn bộ các node đều down lần lượt hoặc tất cả đều down đồng thời thì quy trình start cluster lại hơi khác một chút. Chúng ta đi vào từng trường hợp một.

**Trường hợp thứ nhất:** Tình huống xảy ra khi bạn cần restart cluster để upgrade cho rabbitmq hoặc erlang. Sau khi node 1, node 2 được bạn stop thì thảm họa xảy ra với node còn lại. Node còn lại bị down ngoài ý muốn. Trong trường hợp này việc khởi động lại cluster đòi hỏi thứ tự: Node cuối cùng bị down phải là node đầu tiên được start. Giả sử các node bị down theo thứ tự: node 3 -> node 1 -> node 2. Sau đó tôi cố gắng start các node 3 hoặc node 1 đầu tiên. Tôi sẽ không thành công. Rabbitmq để lại vài dòng log sau:

This cluster node was shut down while other nodes were still running.

To avoid losing data, you should start the other nodes first, then

start this one. To force this node to start, first invoke

"rabbitmqctl force\_boot". If you do so, any changes made on other

cluster nodes after this one was shut down may be lost.

Để khởi động được cluster, bạn chỉ cần tuân theo nguyên tắc, start node 2 đầu tiên. Với các node sau, thứ tự không quan trọng. Bạn có thể dùng thứ tự node 2 - > node 1 -> node 3 hoặc node2 -> node 3 -> node1.

**Trường hợp thứ hai:** Cũng giống trường hợp một nhưng đáng tiếc là node 2 bị sự cố quá nghiêm trọng không thể phục hồi được. Vậy là node cuối cùng không thể boot được. Lúc này bạn phải ép một node không phải node down cuối cùng làm node khởi điểm

[root@rabbitmq01 root]*# rabbitmqctl force\_boot*

Forcing boot for Mnesia dir /var/lib/rabbitmq/mnesia/rabbit@rabbitmq01 ...

[root@rabbitmq01 root]*# service rabbitmq-server start*

Starting rabbitmq-server: SUCCESS

rabbitmq-server.

Sau đó bạn khởi động lại các node kế tiếp.

**Trường hợp thứ ba:** Khủng khiếp hơn ! Bạn chẳng làm gì nhưng cụm server mà chứa rabbitmq cluster bị crash đột ngột. Lúc này thì bạn chẳng thể biết node nào down trước hay down sau cả. Cách xử lý giống hệt trường hợp thứ hai

Bước 5 : Cấu hình HA ( High Availability ) cho RabbitMQ

### Cấu hình Mirroring

Cấu hình Queue mirroring được thực hiện thông qua **policy**. Policie có thể thay đổi mọi thời điểm, đồng nghĩa với việc có thể tạo một non-mirrored queue sau đó biến chúng thành mirrored queue hoặc ngược lại.

Để tạo mirrored queue, ta tạo 1 policy để match và sử dụng key ha-mode và ha-params để set policy. Dưới đây là một vài ví dụ.

Policy match những queue với tên bắt đầu bằng "ha." sẽ mirror sang tất các node trong cluster:

rabbitmqctl set\_policy ha-all "^ha\." '{"ha-mode":"all"}'



Policy match những queue với tên bắt đầu bằng "two." sẽ mirror sang 2 node bất kỳ trong cluster với option automatic synchronization:

rabbitmqctl set\_policy ha-two "^two\." '{"ha-mode":"exactly","ha-params":2,"ha-sync-mode":"automatic"}'

Policy match những queue với tên bắt đầu bằng "nodes." sẽ mirror các node được chỉ định cụ thể trong cluster:

rabbitmqctl set\_policy ha-nodes "^nodes\." '{"ha-mode":"nodes","ha-params":["nodeA", "nodeB"]}'

### Rời một node khỏi cluster

**Cách 1:** Để cho bản thân node đó quên rằng nó đã từng ở trong cluster. Giả sử tôi muốn tách node 2 khỏi cluster hoàn toàn.

[root@rabbitmq02 root]*# rabbitmqctl stop\_app*

Stopping node rabbit@rabbitmq02 ...

[root@rabbitmq02 root]*# rabbitmqctl reset*

Resetting node rabbit@rabbitmq02 ...

[root@rabbitmq02 root]*# rabbitmqctl start\_app*

Starting node rabbit@rabbitmq02 ...

[root@rabbitmq02 root]*# rabbitmqctl cluster\_status*

Cluster status of node rabbit@rabbitmq02 ...

[{nodes,[{disc,[rabbit@rabbitmq02]}]},

{running\_nodes,[rabbit@rabbitmq02]},

{cluster\_name,<<"rabbit@rabbitmq02">>},

{partitions,[]}]

Để reset thành công, bạn không được config cluster trong file cấu hình.  
Reset đồng thời sẽ xóa mọi data của node 2 như vhost, user, exchange, queue...

**Cách 2:** Làm cho các node còn lại trong cluster hắt hủi node cần được tách khỏi cluster :(

[root@rabbitmq02 root]# rabbitmqctl stop\_app  
Stopping node rabbit@rabbitmq02 ...

[root@rabbitmq01 root]# rabbitmqctl forget\_cluster\_node rabbit@rabbitmq02  
Removing node rabbit@rabbitmq02 from cluster ...

Lúc này các node còn lại trong cluster đều đã không coi node 2 nằm trong cluster nhưng node2 vẫn không chịu chấp nhận thực tế phũ phàng đó. Nếu bạn start\_app node 2

Error: {error,{inconsistent\_cluster,"Node rabbit@rabbit2 thinks it's clustered with node rabbit@rabbitmq01, but rabbit@rabbitmq01 disagrees"}}

Để node 2 hoạt động được bình thường, bạn phải làm nó quên đi nó từng thuộc về cluster.

[root@rabbitmq02 root]*# rabbitmqctl reset*

Resetting node rabbit@rabbitmq02 ...

[root@rabbitmq02 root]*# rabbitmqctl start\_app*

Starting node rabbit@rabbitmq02 ...

[root@rabbitmq02 root]*# rabbitmqctl cluster\_status*

Cluster status of node rabbit@rabbitmq02 ...

[{nodes,[{disc,[rabbit@rabbitmq02]}]},

{running\_nodes,[rabbit@rabbitmq02]},

{cluster\_name,<<"rabbit@rabbitmq02">>},

{partitions,[]}]

### Thêm một node vào cluster

[root@rabbitmq03 root]*# rabbitmqctl stop\_app*

Stopping node rabbit@rabbitmq03 ...

[root@rabbitmq03 root]*# rabbitmqctl join\_cluster rabbit@rabbitmq01*

Clustering node rabbit@rabbitmq03 with rabbit@rabbitmq01 ...

[root@rabbitmq03 root]*# rabbitmqctl start\_app*

Starting node rabbit@rabbitmq03 ...

[root@rabbitmq03 root]# roorabbitmqctl cluster\_status

Cluster status of node rabbit@rabbitmq03 ...

[{nodes,[{disc,[rabbit@rabbitmq01,rabbit@rabbitmq02,rabbit@rabbitmq03]}]},

{running\_nodes,[rabbit@rabbitmq01,rabbit@rabbitmq03,rabbit@rabbitmq02]},

{cluster\_name,<<"rabbit@rabbitmq01">>},

{partitions,[]}]

Nếu cố join\_cluster từ một running node, bạn sẽ gặp:

Error: mnesia\_unexpectedly\_running

### Thêm một RAM node

**So sánh RAM node với disc node**  
Sự khác biệt lớn nhất là ram node chỉ giữ metadata của nó trong memory còn bản thân các queue data vẫn lưu xuống disk. Sự khác biệt này cho phép ram node ít tạo ra các hoạt động IO hơn nên performance tốt hơn disc node. Một cluster hoàn toàn chỉ có ram node thì rất có nguy cơ mất metadata. Giải pháp an toàn hơn cả là trộn lẫn ram node và disc node. Trong cluster, phần metadata được replicate giữa các node (disc node lưu metadata trên disk) nên sẽ không lo mất sạch metadata.

[root@rabbitmq02 root]*# rabbitmqctl stop\_app*

Stopping node rabbit@rabbitmq02 ...

[root@rabbitmq02 root]*# rabbitmqctl join\_cluster --ram rabbit@rabbitmq01*

Clustering node rabbit@rabbitmq02 with rabbit@rabbitmq01 ...

[root@rabbitmq02 root]*# rabbitmqctl start\_app*

Starting node rabbit@rabbitmq02 ...

[root@rabbitmq02 root]*# rabbitmqctl cluster\_status*

Cluster status of node rabbit@rabbitmq02 ...

[{nodes,[{disc,[rabbit@rabbitmq03,rabbit@rabbitmq01]},{ram,[rabbit@rabbitmq02]}]},

{running\_nodes,[rabbit@rabbitmq01,rabbit@rabbitmq03,rabbit@rabbitmq02]},

{cluster\_name,<<"rabbit@rabbitmq01">>},

{partitions,[]}]

### Thay đổi node type

[root@rabbitmq02 root]*# rabbitmqctl stop\_app*

Stopping node rabbit@rabbitmq02 ...

[root@rabbitmq02 root]*# rabbitmqctl change\_cluster\_node\_type disc*

Turning rabbit@rabbitmq02 into a disc node ...

[root@rabbitmq02 root]*# rabbitmqctl start\_app*

Starting node rabbit@rabbitmq02 ...

[root@rabbitmq02 root]*# rabbitmqctl cluster\_status*

Cluster status of node rabbit@rabbitmq02 ...

[{nodes,[{disc,[rabbit@rabbitmq01,rabbit@rabbitmq02,rabbit@rabbitmq03]}]},

{running\_nodes,[rabbit@rabbitmq01,rabbit@rabbitmq03,rabbit@rabbitmq02]},

{cluster\_name,<<"rabbit@rabbitmq01">>},

{partitions,[]}]

[root@rabbitmq02 root]*# rabbitmqctl stop\_app*

Stopping node rabbit@rabbitmq02 ...

[root@rabbitmq02 root]*# rabbitmqctl change\_cluster\_node\_type ram*

Turning rabbit@rabbitmq02 into a ram node ...

[root@rabbitmq02 root]*# rabbitmqctl start\_app*

Starting node rabbit@rabbitmq02 ...

[root@rabbitmq02 root]*# rabbitmqctl cluster\_status*

Cluster status of node rabbit@rabbitmq02 ...

[{nodes,[{disc,[rabbit@rabbitmq03,rabbit@rabbitmq01]},{ram,[rabbit@rabbitmq02]}]},

{running\_nodes,[rabbit@rabbitmq01,rabbit@rabbitmq03,rabbit@rabbitmq02]},

{cluster\_name,<<"rabbit@rabbitmq01">>},

{partitions,[]}]

**Vấn đề timeout**

Bản thân client sẽ luôn giữ kết nối đến rabbitmq. Sẽ không có timeout nếu như bạn kết nối trực tiếp nhưng khi qua một proxy thì vấn đề xuất hiện. Proxy sẽ không giữ kết nối liên tục giữa client và backend nên trong quá trình sử dụng bạn có thể thấy hiện tượng client bị mất kết nối sau một quãng thời gian không sử dụng. Đáng tiếc rabbitmq client không có cơ chế reconnect lại.

Một linux client có cơ chế tự động gửi lại keep-alive packet để duy trì kết nối nhưng quãng thời gian này quá lâu. cat /proc/sys/net/ipv4/tcp\_keepalive\_time trả về giá trị 7200 nghĩa là cứ sau 2 tiếng mới có một cú gửi keep-alived. Muốn proxy duy trì kết nối thì keep-alived packet phải được gửi trước khi timeout của proxy kết thúc. Trong trường hợp của tôi proxy là haproxy. Tôi điều chỉnh chút ít về cấu hình. Tôi bổ sung ba dòng sau vào cụm backend rabbitmq

timeout client 3h

timeout server 3h

option clitcpka