**RabbitMQ SSL Implementation**

**Rabbit MQ SSL Document**:

**TABLE OF CONTENTS :**

1. Installation and Configuration Rabbit MQ and Erlang(On Bothe Nodes)……………………………2

1.1 AMI backup and exporting the queques**………………………………………………………………………....2**

1.2 Installation and configuration of rabbitMQ and erlang.........................................................2

2. Setting Up Oracle JDK Environment……………………………………………………………………………………………..2

3. Cluster Formation (2 Nodes)………………………………………………………………………………………………………….3

3.1 Setting Hostname on Instances……………………………………………………………………………………………….4

4. RabbitMQ Cluster Creation………………………………………………………………………………………………4

4.1 RabbitMQ Erlang cookie………………………………………………………………………………………….4

4.2 Joining the cluster node2 from node3……………………………………………………………………………………4

4.3 cluster status from Dev2-RabbitMQ-02…………………………………………………………………………………5

4.4 cluster status from Dev2-RabbitMQ-03…………………………………………………………………………………5

5. Steps for Cluster restart…………………………………………………………………………………………………………………6

6. Terminating RabbitMQ node………………………………………………………………………………………………………….8

7. Clustering Communication using SSL……………………………………………………………………………………………10

8. Rabbitmq cluster communication with ssl validation………………………………………………………11

1. **Installation and Configuration Rabbit MQ and Erlang(On Bothe Nodes):**

**1.1 AMI backup and exporting the queques:**

* We will do the activity after taking AMI backup and exporting the queques from rabbitmq nodes.

**Import export definitions**

It is possible to import and export configuration definitions. When you download the definitions, you get a JSON representation of your broker (your RabbitMQ settings). This can be used to restore exchanges, queues, virtual hosts, policies and users. This feature can be used as a backup. Every time you make a change in the config, you can keep the old settings just in case.

--------🡪exporting screenshots

* Firstly we need to uninstall the current old version of rabbitmq and erlang from the server.

**yum remove rabbitmq\* erlang \***

* 1. **Installation and configuration of rabbitMQ and erlang!:**
* In order to Install and configure RabbitMQ we can get rpm package from below url and run the below commands for installation for both nodes.

**# wget <https://www.rabbitmq.com/releases/erlang/erlang-18.3-1.el7.centos.x86_64.rpm>**

# **sudo rpm -Uvh erlang-18.3-1.el7.centos.x86\_64.rpm**

# **yum install socat**

**# wget** [**https://www.rabbitmq.com/releases/rabbitmq-server/v3.6.5/rabbitmq-server-3.6.5-1.noarch.rpm**](https://www.rabbitmq.com/releases/rabbitmq-server/v3.6.5/rabbitmq-server-3.6.5-1.noarch.rpm)

**# yum install rabbitmq-server-3.6.5-1.noarch.rpm**

**#** **sudo chkconfig rabbitmq-server on**

**# sudo /etc/init.d/rabbitmq-server start**

**# rabbitmqctl status**

1. **Setting Up Oracle JDK Environment :**

* Assuming JDK Rpm package is available in pwd (present working directory) run the below command for install JDK

**# yum localinstall jdk-8u45-linux-x64.rpm**

1. **Cluster Formation (2 Nodes):**

Steps to Form Cluster

**3.1 Setting Hostname on Instances:**

* For the cluster formation we should provide all node hostname in host file.

**For Node 2:**

* **[root@Dev2-RabbitMQ-02 ~]# vim /etc/hosts**

127.0.0.1 localhost localhost.localdomain

14.0.1.196 Dev2-RabbitMQ-02

14.0.1.197 Dev2-RabbitMQ-03

* **[root@Dev2-RabbitMQ-02 ~]# vim /etc/sysconfig/network**

NETWORKING=yes

HOSTNAME=Dev2-RabbitMQ-02

NOZEROCONF=yes

NETWORKING\_IPV6=no

IPV6INIT=no

IPV6\_ROUTER=no

IPV6\_AUTOCONF=no

IPV6FORWARDING=no

IPV6TO4INIT=no

IPV6\_CONTROL\_RADVD=no

**For Node 3**:

**[root@Dev2-RabbitMQ-03~]# vim /etc/hosts**

127.0.0.1 localhost localhost.localdomain

14.0.1.196 Dev2-RabbitMQ-02

14.0.1.197 Dev2-RabbitMQ-03

**[root@Dev2-RabbitMQ-03 ~]# vim /etc/sysconfig/network**

NETWORKING=yes

HOSTNAME=Dev2-RabbitMQ-03

NOZEROCONF=yes

NETWORKING\_IPV6=no

IPV6INIT=no

IPV6\_ROUTER=no

IPV6\_AUTOCONF=no

IPV6FORWARDING=no

IPV6TO4INIT=no

IPV6\_CONTROL\_RADVD=no

---------🡪>> ulimit values updation for all nodes

ulimit value need to setup for all cluster nodes like below

[root@Dev2-RabbitMQ-03 ~]# ulimit -S -n 4096

[root@Dev2-RabbitMQ-03 ~]# ulimit -H -n 65536

[root@Dev2-RabbitMQ-03 ~]# cat /etc/security/limits.conf

root hard nofile 65536

root soft nofile 65536

rabbitmq hard nofile 65536

rabbitmq soft nofile 65536

[root@Dev2-RabbitMQ-03 ~]# /etc/init.d/rabbitmq-server start

Starting rabbitmq-server: SUCCESS

rabbitmq-server.

[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl cluster\_status

Cluster status of node 'rabbit@Dev2-RabbitMQ-03' ...

[{nodes,[{disc,['rabbit@Dev2-RabbitMQ-02','rabbit@Dev2-RabbitMQ-03']}]},

{running\_nodes,['rabbit@Dev2-RabbitMQ-02','rabbit@Dev2-RabbitMQ-03']},

{cluster\_name,<<"Dev2-RabbitMQ-Cluster">>},

{partitions,[]},

{alarms,[{'rabbit@Dev2-RabbitMQ-02',[]},{'rabbit@Dev2-RabbitMQ-03',[]}]}]

[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl status

Status of node 'rabbit@Dev2-RabbitMQ-03' ...

[{pid,3059},

{running\_applications,

[{rabbitmq\_management,"RabbitMQ Management Console","3.6.5"},

{rabbitmq\_web\_dispatch,"RabbitMQ Web Dispatcher","3.6.5"},

{webmachine,"webmachine","1.10.3"},

{mochiweb,"MochiMedia Web Server","2.13.1"},

{rabbitmq\_management\_agent,"RabbitMQ Management Agent","3.6.5"},

{rabbit,"RabbitMQ","3.6.5"},

{amqp\_client,"RabbitMQ AMQP Client","3.6.5"},

{rabbit\_common,[],"3.6.5"},

{os\_mon,"CPO CXC 138 46","2.4"},

{ssl,"Erlang/OTP SSL application","7.3"},

{public\_key,"Public key infrastructure","1.1.1"},

{crypto,"CRYPTO","3.6.3"},

{asn1,"The Erlang ASN1 compiler version 4.0.2","4.0.2"},

{ranch,"Socket acceptor pool for TCP protocols.","1.2.1"},

{xmerl,"XML parser","1.3.10"},

{mnesia,"MNESIA CXC 138 12","4.13.3"},

{inets,"INETS CXC 138 49","6.2"},

{syntax\_tools,"Syntax tools","1.7"},

{rabbitmq\_auth\_mechanism\_ssl,

"RabbitMQ SSL authentication (SASL EXTERNAL)","3.6.5"},

{compiler,"ERTS CXC 138 10","6.0.3"},

{sasl,"SASL CXC 138 11","2.7"},

{stdlib,"ERTS CXC 138 10","2.8"},

{kernel,"ERTS CXC 138 10","4.2"}]},

{os,{unix,linux}},

{erlang\_version,

"Erlang/OTP 18 [erts-7.3] [source] [64-bit] [async-threads:64] [hipe] [kernel-poll:true]\n"},

{memory,

[{total,59991232},

{connection\_readers,0},

{connection\_writers,0},

{connection\_channels,0},

{connection\_other,2680},

{queue\_procs,2680},

{queue\_slave\_procs,0},

{plugins,479888},

{other\_proc,19383784},

{mnesia,69008},

{mgmt\_db,123904},

{msg\_index,39648},

{other\_ets,1442768},

{binary,33472},

{code,27832428},

{atom,1000601},

{other\_system,9580371}]},

{alarms,[]},

{listeners,[{clustering,25672,"::"},{amqp,5672,"::"},{'amqp/ssl',5671,"::"}]},

{vm\_memory\_high\_watermark,0.4},

{vm\_memory\_limit,1580181094},

{disk\_free\_limit,50000000},

{disk\_free,3549458432},

{file\_descriptors,

[{total\_limit,65436},-------------🡪

{total\_used,2},

{sockets\_limit,58890},

{sockets\_used,0}]},

{processes,[{limit,1048576},{used,206}]},

{run\_queue,0},

{uptime,14},

{kernel,{net\_ticktime,60}}]

#####3

1. **RabbitMQ Cluster Creation:**

Clusters are set up by re-configuring existing RabbitMQ nodes into a cluster configuration. Hence the first step is to start RabbitMQ on all nodes in the normal way:

**[root@Dev2-RabbitMQ-02 ~]# rabbitmq-server -detached**

**[root@Dev2-RabbitMQ-03 ~]# rabbitmq-server –detached**

**4.1.RabbitMQ Erlang cookie:**

RabbitMQ nodes and CLI tools (e.g. rabbitmqctl) use a cookie to determine whether they are allowed to communicate with each other. For two nodes to be able to communicate they must have the same shared secret called the Erlang cookie. The cookie is just a string of alphanumeric characters. It can be as long or short as you like. Every cluster node must have the same cookie.

Erlang VM will automatically create a random cookie file when the RabbitMQ server starts up. The easiest way to proceed is to allow one node to create the file, and then copy it to all the other nodes in the cluster

**Node02:**

**['rabbit@Dev2-RabbitMQ-02']# cat /var/lib/rabbitmq/.erlang.cookie**

**TAPJIJMUSLIDGUWMVTLB**

**Node 03:**

**['rabbit@Dev2-RabbitMQ-03']# cat /var/lib/rabbitmq/.erlang.cookie**

**TAPJIJMUSLIDGUWMVTLB**

* For cluster creation we should joining the Node 2 from node 3.

**4.2 Joining the cluster node2 from node3:**

* For cluster creation we should joining the Node 2 from node3 first we stop the RabbitMQ application to run below command .

**[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl stop\_app**

**Stopping node 'rabbit@Dev2-RabbitMQ-03' ...**

* In order to create a Clustering node 'rabbit@Dev2-RabbitMQ-03' with 'rabbit@Dev2-RabbitMQ-02' we should run the below command .

**[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl join\_cluster rabbit@Dev2-RabbitMQ-02**

* Now Clustering node 'rabbit@Dev2-RabbitMQ-03' with 'rabbit@Dev2-RabbitMQ-02' ...done.
* For starting of the rabbitMQ application in Node3 we should run below command .

**[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl start\_app**

**Starting node 'rabbit@Dev2-RabbitMQ-03' ...**

**4.3 cluster status from Dev2-RabbitMQ-02:**

* Now we should check the status of the RabbitMQ cluster by using the below command form Node2

**[root@Dev2-RabbitMQ-02 ~]# rabbitmqctl cluster\_status**

**Cluster status of node 'rabbit@Dev2-RabbitMQ-02' ...**

**[{nodes,[{disc,['rabbit@Dev2-RabbitMQ-02','rabbit@Dev2-RabbitMQ-03']}]},**

**{running\_nodes,['rabbit@Dev2-RabbitMQ-03','rabbit@Dev2-RabbitMQ-02']},**

**{cluster\_name,<<"Dev2-RabbitMQ-Cluster">>},**

**{partitions,[]},**

**{alarms,[{'rabbit@Dev2-RabbitMQ-03',[]},{'rabbit@Dev2-RabbitMQ-02',[]}]}]**

**4.4 cluster status from Dev2-RabbitMQ-03:**

* Now we should check the status of the RabbitMQ cluster by using the below command from Node3

**[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl cluster\_status**

**Cluster status of node 'rabbit@Dev2-RabbitMQ-03' ...**

**[{nodes,[{disc,['rabbit@Dev2-RabbitMQ-02','rabbit@Dev2-RabbitMQ-03']}]},**

**{running\_nodes,['rabbit@Dev2-RabbitMQ-02','rabbit@Dev2-RabbitMQ-03']},**

**{cluster\_name,<<"Dev2-RabbitMQ-Cluster">>},**

**{partitions,[]},**

**{alarms,[{'rabbit@Dev2-RabbitMQ-02',[]},{'rabbit@Dev2-RabbitMQ-03',[]}]}]**

1. **Steps for Cluster restart :**

**step1:**

* stop the rabbitmq application from node2

**[root@Dev2-RabbitMQ-02 ~]# rabbitmqctl stop\_app**

**Stopping node 'rabbit@Dev2-RabbitMQ-02' ...**

* stop the rabbitmq server from node2

**[root@Dev2-RabbitMQ-02 ~]# /etc/init.d/rabbitmq-server stop**

**Stopping rabbitmq-server:**

**rabbitmq-server.**

**[root@Dev2-RabbitMQ-02 ~]#**

* check the rabbitmq server status from node2

**[root@Dev2-RabbitMQ-02 ~]# /etc/init.d/rabbitmq-server status**

**Status of node 'rabbit@Dev2-RabbitMQ-02' ...**

**step2:**

* stop the rabbitmq application from node3

**[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl stop\_app**

**Stopping node 'rabbit@Dev2-RabbitMQ-03' ...**

* stop the rabbitmq server from node3

**[root@Dev2-RabbitMQ-03 ~]# /etc/init.d/rabbitmq-server stop**

**Stopping rabbitmq-server:**

**rabbitmq-server.**

**[root@Dev2-RabbitMQ-03 ~]#**

* check the rabbitmq server status from node3

**[root@Dev2-RabbitMQ-03 ~]# /etc/init.d/rabbitmq-server status**

**Status of node 'rabbit@Dev2-RabbitMQ-03' ...**

**step3:**

* Start the rabbitmq server from node3

**[root@Dev2-RabbitMQ-03 ~]# /etc/init.d/rabbitmq-server start**

**Starting rabbitmq-server: SUCCESS**

**rabbitmq-server.**

* Start the rabbitmq application from node3

**[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl start\_app**

**Starting node 'rabbit@Dev2-RabbitMQ-03' ...**

**step4:**

* Start the rabbitmq server from node2

**[root@Dev2-RabbitMQ-02 ~]# /etc/init.d/rabbitmq-server start**

**Starting rabbitmq-server: RabbitMQ is currently running**

**rabbitmq-server.**

* Start the rabbitmq application from node2

**[root@Dev2-RabbitMQ-02 ~]# rabbitmqctl start\_app**

**Starting node 'rabbit@Dev2-RabbitMQ-02' ...**

**step5.**

**Now check the cluster status on both nodes:**

**In Node02:**

* Now check the cluster status from node02 for that we can run the below command.

**[root@Dev2-RabbitMQ-02 ~]# rabbitmqctl cluster\_status**

**Cluster status of node 'rabbit@Dev2-RabbitMQ-02' ...**

**[{nodes,[{disc,['rabbit@Dev2-RabbitMQ-02','rabbit@Dev2-RabbitMQ-03']}]},**

**{running\_nodes,['rabbit@Dev2-RabbitMQ-03','rabbit@Dev2-RabbitMQ-02']},**

**{cluster\_name,<<"Dev2-RabbitMQ-Cluster">>},**

**{partitions,[]},**

**{alarms,[{'rabbit@Dev2-RabbitMQ-03',[]},{'rabbit@Dev2-RabbitMQ-02',[]}]}]**

* Here both node are up and running

**In node03**:

* Now we should check the cluster status from node03 for that we can run the below command .

**[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl cluster\_status**

**Cluster status of node 'rabbit@Dev2-RabbitMQ-03' ...**

**[{nodes,[{disc,['rabbit@Dev2-RabbitMQ-02','rabbit@Dev2-RabbitMQ-03']}]},**

**{running\_nodes,['rabbit@Dev2-RabbitMQ-02','rabbit@Dev2-RabbitMQ-03']},**

**{cluster\_name,<<"Dev2-RabbitMQ-Cluster">>},**

**{partitions,[]},**

**{alarms,[{'rabbit@Dev2-RabbitMQ-02',[]},{'rabbit@Dev2-RabbitMQ-03',[]}]}]**

**####################cluster setup done for two nodes ################**

**6.** **Terminating RabbitMQ node:**

* This command instructs the RabbitMQ node to terminate.
* we should check the cluster status from node02 for that we can run the below command

**[root@Dev2-RabbitMQ-02 ~]# rabbitmqctl cluster\_status**

**Cluster status of node 'rabbit@Dev2-RabbitMQ-02' ...**

**[{nodes,[{disc,['rabbit@Dev2-RabbitMQ-02','rabbit@Dev2-RabbitMQ-03']}]},**

**{running\_nodes,['rabbit@Dev2-RabbitMQ-03','rabbit@Dev2-RabbitMQ-02']},**

**{cluster\_name,<<"Dev2-RabbitMQ-Cluster">>},**

**{partitions,[]},**

**{alarms,[{'rabbit@Dev2-RabbitMQ-03',[]},{'rabbit@Dev2-RabbitMQ-02',[]}]}]**

**Step1**:

* stop the rabbitmq application from node2

**[root@Dev2-RabbitMQ-02 ~]# rabbitmqctl stop**

**Stopping and halting node 'rabbit@Dev2-RabbitMQ-02' ...**

**[root@Dev2-RabbitMQ-02 ~]# rabbitmqctl reset**

**Restting node 'rabbit@Dev2-RabbitMQ-02 … done.**

**[root@Dev2-RabbitMQ-02 ~]# rabbitmqctl force\_reset**

**Restting node 'rabbit@Dev2-RabbitMQ-02 … done.**

* Then check rabbitmq cluster status from node2

**[root@Dev2-RabbitMQ-02 ~]# rabbitmqctl cluster\_status**

**Cluster status of node 'rabbit@Dev2-RabbitMQ-02' ...**

**Error: unable to connect to node 'rabbit@Dev2-RabbitMQ-02': nodedown**

**DIAGNOSTICS FROM NODE02:**

**========================**

**attempted to contact: ['rabbit@Dev2-RabbitMQ-02']**

**rabbit@Dev2-RabbitMQ-02:**

**\* connected to epmd (port 4369) on Dev2-RabbitMQ-02**

**\* epmd reports: node 'rabbit' not running at all**

**no other nodes on Dev2-RabbitMQ-02**

**\* suggestion: start the node**

**current node details:**

**- node name: 'rabbitmq-cli-14@Dev2-RabbitMQ-02'**

**- home dir: /var/lib/rabbitmq**

**- cookie hash: 4VUANjzWKcXLbCOrW+U2RA==**

**[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl cluster\_status**

**Cluster status of node 'rabbit@Dev2-RabbitMQ-03' ...**

**Error: unable to connect to node 'rabbit@Dev2-RabbitMQ-03': nodedown**

**DIAGNOSTICS**

**attempted to contact: ['rabbit@Dev2-RabbitMQ-03']**

**rabbit@Dev2-RabbitMQ-03:**

**\* connected to epmd (port 4369) on Dev2-RabbitMQ-03**

**\* epmd reports: node 'rabbit' not running at all**

**no other nodes on Dev2-RabbitMQ-03**

**\* suggestion: start the node**

**current node details:**

**- node name: 'rabbitmq-cli-05@Dev2-RabbitMQ-03'**

**- home dir: /var/lib/rabbitmq**

**- cookie hash: 4VUANjzWKcXLbCOrW+U2RA==**

**Second RabbitMQ node to terminate:**

* stop the rabbitmq application from node3

**[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl stop**

**Stopping and halting node 'rabbit@Dev2-RabbitMQ-03' ...**

* Then check rabbitmq cluster status from node3

**[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl cluster\_status**

**Cluster status of node 'rabbit@Dev2-RabbitMQ-03' ...**

**Error: unable to connect to node 'rabbit@Dev2-RabbitMQ**-03': nodedown

**DIAGNOSTICS FROM NODE03:**

**========================**

**attempted to contact: ['rabbit@Dev2-RabbitMQ-03']**

**rabbit@Dev2-RabbitMQ-03:**

**\* connected to epmd (port 4369) on Dev2-RabbitMQ-03**

**\* epmd reports: node 'rabbit' not running at all**

**no other nodes on Dev2-RabbitMQ-03**

**\* suggestion: start the node**

**current node details:**

**- node name: 'rabbitmq-cli-16@Dev2-RabbitMQ-03'**

**- home dir: /var/lib/rabbitmq**

**- cookie hash: 4VUANjzWKcXLbCOrW+U2RA==**

**[root@Dev2-RabbitMQ-03 ~]# rabbitmqctl start**

**Error: could not recognise command**

Once cluster provisioned , we have to import the queues which we exported ealier as below

##### Import export definitions

It is possible to import and export configuration definitions. When you download the definitions, you get a JSON representation of your broker (your RabbitMQ settings). This can be used to restore exchanges, queues, virtual hosts, policies and users. This feature can be used as a backup. Every time you make a change in the config, you can keep the old settings just in case.

---🡪 import screenhosts

**####heading### ssl certification configuration with RMQ:**

**1.Ssl certificates creations for rabbitmq**

**-------------have to update-----------------------**

**2. configuration of ssl certs for rabbitmq**

**Once certificates are created then ssl configuration has to add in the rabbitmq configuration file for all nodes where ever required ssl communication.**

**[root@Dev2-RabbitMQ-02 ~]# cat /etc/rabbitmq/rabbitmq.config**

**%%[{rabbit, [{loopback\_users, []}]}].**

**[**

**{ssl, [{versions, ['tlsv1.2', 'tlsv1.1', tlsv1]}]},**

**{rabbit, [**

**{loopback\_users, []},**

**{ssl\_listeners, [5671] },**

**{ssl\_options, [{cacertfile,"/etc/testca/cacert.pem"},**

**{certfile,"/etc/server/cert.pem"},**

**{keyfile,"/etc/server/key.pem"},**

**{verify,verify\_peer},**

**{versions, ['tlsv1.2', 'tlsv1.1', tlsv1]},**

**{ciphers, ["ECDHE-ECDSA-AES256-GCM-SHA384","ECDHE-RSA-AES256-GCM-SHA384",**

**"ECDHE-ECDSA-AES256-SHA384","ECDHE-RSA-AES256-SHA384", "ECDHE-ECDSA-DES-CBC3-SHA",**

**"ECDH-ECDSA-AES256-GCM-SHA384","ECDH-RSA-AES256-GCM-SHA384","ECDH-ECDSA-AES256-SHA384",**

**"ECDH-RSA-AES256-SHA384","DHE-DSS-AES256-GCM-SHA384","DHE-DSS-AES256-SHA256",**

**"AES256-GCM-SHA384","AES256-SHA256","ECDHE-ECDSA-AES128-GCM-SHA256",**

**"ECDHE-RSA-AES128-GCM-SHA256","ECDHE-ECDSA-AES128-SHA256","ECDHE-RSA-AES128-SHA256",**

**"ECDH-ECDSA-AES128-GCM-SHA256","ECDH-RSA-AES128-GCM-SHA256","ECDH-ECDSA-AES128-SHA256",**

**"ECDH-RSA-AES128-SHA256","DHE-DSS-AES128-GCM-SHA256","DHE-DSS-AES128-SHA256",**

**"AES128-GCM-SHA256","AES128-SHA256","ECDHE-ECDSA-AES256-SHA",**

**"ECDHE-RSA-AES256-SHA","DHE-DSS-AES256-SHA","ECDH-ECDSA-AES256-SHA",**

**"ECDH-RSA-AES256-SHA","AES256-SHA","ECDHE-ECDSA-AES128-SHA",**

**"ECDHE-RSA-AES128-SHA","DHE-DSS-AES128-SHA","ECDH-ECDSA-AES128-SHA",**

**"ECDH-RSA-AES128-SHA","AES128-SHA"]},**

**{honor\_cipher\_order, true},**

**{fail\_if\_no\_peer\_cert,false}]}**

**]**

**}].**

**7.** **Clustering Communication using SSL:**

Sometimes is desirable to make the Erlang nodes talk to each other using SSL, and thus make the whole RabbitMQ cluster communication via SSL. To achieve that we need to make the Erlang distribution mechanism to use SSL. In this document we are going to review the steps to make this possible.

First we need to create the SSL certificate that's going to be used by the Erlang distribution mechanism. We assume you have done that already, otherwise follow the guide here. Once we have our certificates ready we need to concatenate the server certificate and key into one file.

* we have the files cert.pem key.pem we can do the following:

**cd /etc/server/**

**cat cert.pem key.pem > rabbit.pem**

* Then we have to tell Erlang where to find the ssl library during startup. We can create a variable like this:

**[root@Dev2-RabbitMQ-02 ~]# echo `erl -eval 'io:format("~p", [code:lib\_dir(ssl, ebin)]),halt().' -noshell`**

**"/usr/lib64/erlang/lib/ssl-7.3/ebin"**

**[root@Dev2-RabbitMQ-02 ~]#**

**export ERL\_SSL\_PATH=/usr/lib64/erlang/lib/ssl-7.3/ebin**

There first we find where Erlang has the ssl library, and then the variable ERL\_SSL\_PATH is set with the result from the first command without the double quotes.

By using the previous information now is time to craft the $RABBITMQ\_SERVER\_ADDITIONAL\_ERL\_ARGS environment variable so RabbitMQ is able to start Erlang using SSL for distribution. We do that by setting the proto\_dist argument to inet\_tls and then telling Erlang what certificate to use (in our case that's the rabbit.pem file we just created).

Finally we set secure renegotiation to true. Here's the whole command:

**[root@Dev2-RabbitMQ-02 ~]#**

**export RABBITMQ\_SERVER\_ADDITIONAL\_ERL\_ARGS="-pa ${ERL\_SSL\_PATH} \**

**-proto\_dist inet\_tls \**

**-ssl\_dist\_opt server\_certfile /etc/server/rabbit.pem \**

**-ssl\_dist\_opt server\_secure\_renegotiate true client\_secure\_renegotiate true"**

Keep in mind that every Erlang program that tries to communicate with our RabbitMQ server by using Erlang's distribution must now use SSL as well.

One of such programs is rabbitmqctl that we use for administering RabbitMQ. This means we have to do what we just did for $RABBITMQ\_SERVER\_ADDITIONAL\_ERL\_ARGS but this time for the environment variable RABBITMQ\_CTL\_ERL\_ARGS.

Now that we have this in place, it's just a matter of starting RabbitMQ as we usually do to get the Erlang distribution to use SSL for internode communication. Keep in mind that other nodes in the cluster that want to join our initial node must use the same certificate and the same RABBITMQ\_SERVER\_ADDITIONAL\_ERL\_ARGS arguments. The same applies for rabbitmqctl.

**8. Rabbitmq cluster communication with ssl validation :**

**[harish.k@Dev2-RabbitMQ-02 ~]$ telnet 14.0.1.197 5671**

**Trying 14.0.1.197...**

**Connected to 14.0.1.197.**

**Escape character is '^]'.**

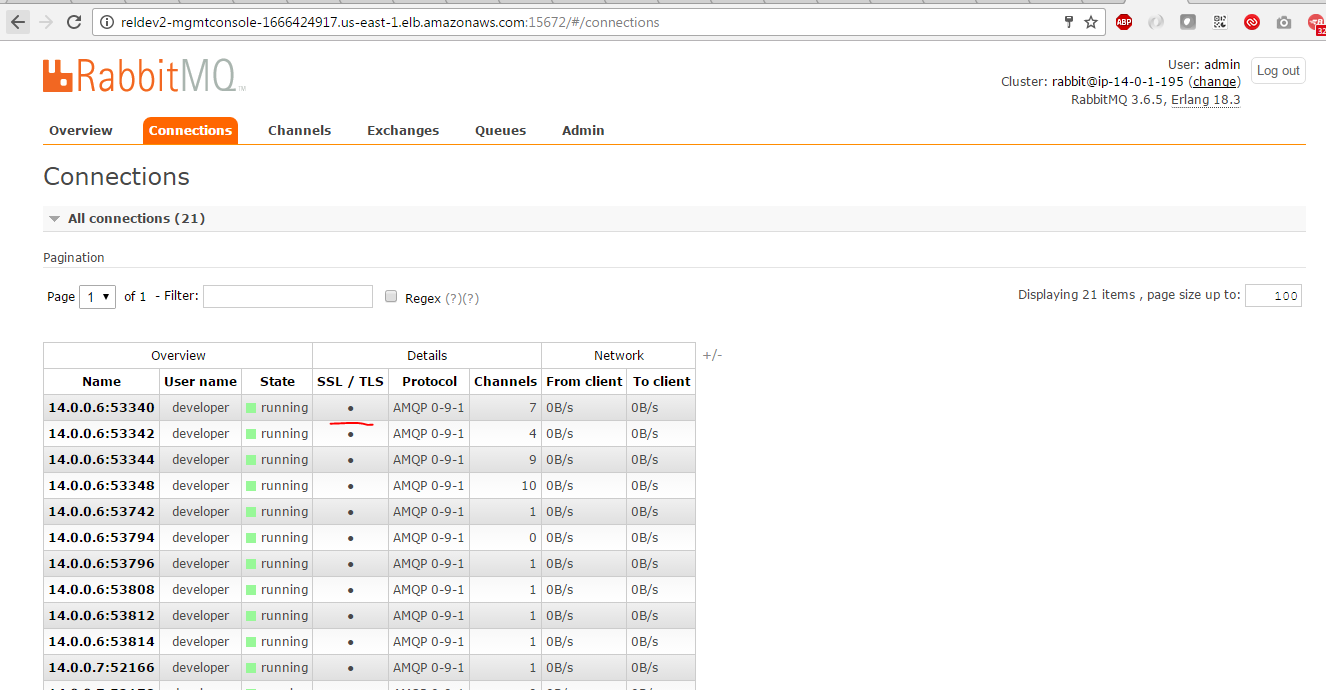
**[20:04:05] HaRish KunChe: [harish.k@Dev2-RabbitMQ-03 ~]$ telnet 14.0.1.196 5671**

**Trying 14.0.1.196...**

**Connected to 14.0.1.196.**

**Escape character is '^]'.**

**9. rabbitmq ssl verification:**

****