(RabbitMQ)

SSL/TLS enabled RabbitMQ Setup Doc

Table of Content

Change Control

Version	Date	Author	Description
1.0	07-Dec-2023	Bharat Gupta	Doc initialization

Issues

S.R. No.	Date	Owner	Description	Resolution
1.0				

Pending Tasks

S.R. No.	Owner	Description	Resolution

Overview

This document provides an overview of the RabbitMQ SSL/TLS-enabled server setup, offering detailed guidance on configuring RabbitMQ for secure communication.

RabbitMQ is an open-source enterprise message broker that is free to use. It implements the Advanced Message Queueing Protocol and is developed in Erlang (AMQP). Client libraries are available in all major programming languages.

QA Team

Engineer - Bharat Gupta
Team - QA NS

RabbitMQ Server Setup

Below are the steps for creating SSL/TLS enabled RabbitMQ Server Setup -

1. Create a Directory named RabbitMQ.
2. Create a .sh file named RabbitMQ.sh in RabbitMQ directory.
3. Paste below shell snippet in the .sh file.

```
#!/bin/bash

username="bharat"
password="bharat"

echo "################################### WELCOME TO RABBITMQ
INSTALLATION SCRIPT ##############################"

sleep 1s

echo "################################# INSTALLING DEPENDENCIES &
REQUIRED UTILITIES ##############################"

sudo apt update
sudo apt install -y vim htop telnet net-tools dnsutils openssl libssl-dev libudev-dev git
build-essential

echo "############################################ STARTED
INSTALLING RABBITMQ
##################################################"
```

```
sudo apt install -y
https://github.com/rabbitmq/erlang-rpm/releases/download/v19.3.6.2/erlang_19.3.6.2-1
_amd64.deb
sudo apt install -y
https://github.com/rabbitmq/rabbitmq-server/releases/download/v3.7.6/rabbitmq-serve
r_3.7.6-1_all.deb

sudo systemctl start rabbitmq-server
sudo systemctl enable rabbitmq-server
sudo service rabbitmq-server status

sudo rabbitmq-plugins enable rabbitmq_management
sudo systemctl restart rabbitmq-server

echo "##################################### ADDING USER TO RABBITMQ
##########################"

sleep 1s

sudo rabbitmqctl add_user $username $password

echo "################################### SETTING RABBITMQ
PERMISSIONS ##################################"

sleep 1s

sudo rabbitmqctl set_permissions -p / $username ".*" ".*" ".*"

echo "################################# JUST FINAL SETUP PLEASE STAY
PATIENT ############################"

sudo rabbitmqctl set_user_tags $username administrator

echo "############################## RABBITMQ INSTALLED & CONFIGURED
SUCCESSFULLY #######################"
```

4. Save the file and give RabbitMQ.sh file execute permission.
   chmod +x RabbitMQ.sh
5. Execute the file
   ./RabbitMQ.sh
6. Run command "**sudo systemctl status rabbitmq-server.service**" to check the rabbitmq service is running.

Example -

```
cavisson@worker-1:~/RabbitMQ_ssl$ sudo systemctl status rabbitmq-server.service
● rabbitmq-server.service - RabbitMQ Messaging Server
     Loaded: loaded (/lib/systemd/system/rabbitmq-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-12-07 11:11:38 IST; 1h 31min ago
   Main PID: 3685036 (beam.smp)
     Status: "Initialized"
      Tasks: 91 (limit: 18981)
     Memory: 91.1M
     CGroup: /system.slice/rabbitmq-server.service
             ├─3685032 /bin/sh /usr/sbin/rabbitmq-server
             ├─3685036 /usr/lib/erlang/erts-10.6.4/bin/beam.smp -W w -A 64 -MBas ageffcbf -MHas ageffcbf -MBlmbcs 512 -MHlmbcs 512 -MMmcs 30 -P 1048576 -t 5000000 -stbt db -zdbbl 128000 -K true -- -root /usr/lib
             ├─3685338 erl_child_setup 65536
             ├─3703409 inet_gethost 4
             └─3703423 inet_gethost 4

Dec 07 11:11:22 worker-1 systemd[1]: Starting RabbitMQ Messaging Server...
Dec 07 11:11:38 worker-1 systemd[1]: rabbitmq-server.service: Supervising process 3685036 which is not our child. We'll most likely not notice when it exits.
Dec 07 11:11:38 worker-1 systemd[1]: Started RabbitMQ Messaging Server.
lines 1-17/17 (END)
```

7. Run command "**sudo rabbitmq-plugins list**" to see the list of plugins RabbitMQ supports.

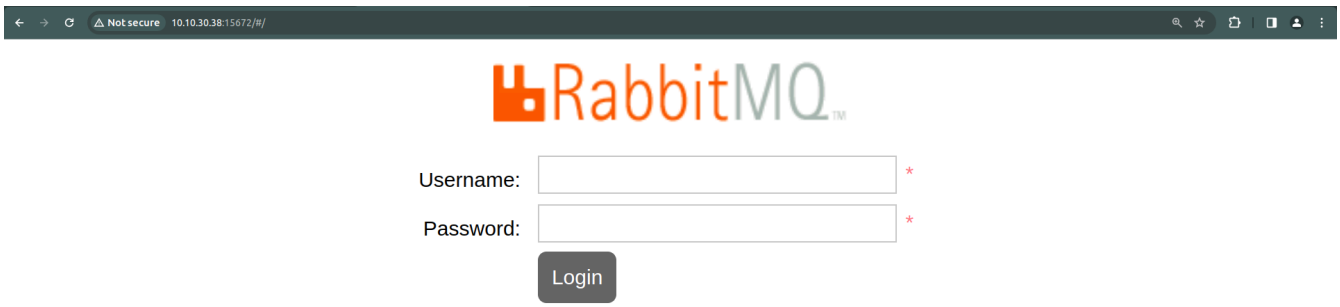
```
cavisson@worker-1:~/RabbitMQ_ssl$ sudo rabbitmq-plugins list
Listing plugins with pattern ".*" ...
 Configured: E = explicitly enabled; e = implicitly enabled
 | Status: * = running on rabbit@worker-1
 |/
[  ] rabbitmq_amqp1_0                  3.8.2
[  ] rabbitmq_auth_backend_cache       3.8.2
[  ] rabbitmq_auth_backend_http        3.8.2
[  ] rabbitmq_auth_backend_ldap        3.8.2
[  ] rabbitmq_auth_backend_oauth2      3.8.2
[  ] rabbitmq_auth_mechanism_ssl       3.8.2
[  ] rabbitmq_consistent_hash_exchange 3.8.2
[  ] rabbitmq_event_exchange           3.8.2
[  ] rabbitmq_federation               3.8.2
[  ] rabbitmq_federation_management    3.8.2
[  ] rabbitmq_jms_topic_exchange       3.8.2
[E*] rabbitmq_management               3.8.2
[e*] rabbitmq_management_agent         3.8.2
[  ] rabbitmq_mqtt                     3.8.2
[  ] rabbitmq_peer_discovery_aws       3.8.2
[  ] rabbitmq_peer_discovery_common    3.8.2
[  ] rabbitmq_peer_discovery_consul    3.8.2
[  ] rabbitmq_peer_discovery_etcd      3.8.2
[  ] rabbitmq_peer_discovery_k8s       3.8.2
[  ] rabbitmq_prometheus               3.8.2
[  ] rabbitmq_random_exchange          3.8.2
[  ] rabbitmq_recent_history_exchange  3.8.2
[  ] rabbitmq_sharding                 3.8.2
[  ] rabbitmq_shovel                   3.8.2
[  ] rabbitmq_shovel_management        3.8.2
[  ] rabbitmq_stomp                    3.8.2
[  ] rabbitmq_top                      3.8.2
[  ] rabbitmq_tracing                  3.8.2
[  ] rabbitmq_trust_store              3.8.2
[e*] rabbitmq_web_dispatch             3.8.2
[  ] rabbitmq_web_mqtt                 3.8.2
[  ] rabbitmq_web_mqtt_examples        3.8.2
[  ] rabbitmq_web_stomp                3.8.2
[  ] rabbitmq_web_stomp_examples       3.8.2
```

8. Enable RabbitMQ management plugin using "**sudo rabbitmq-plugins enable rabbitmq_management**"

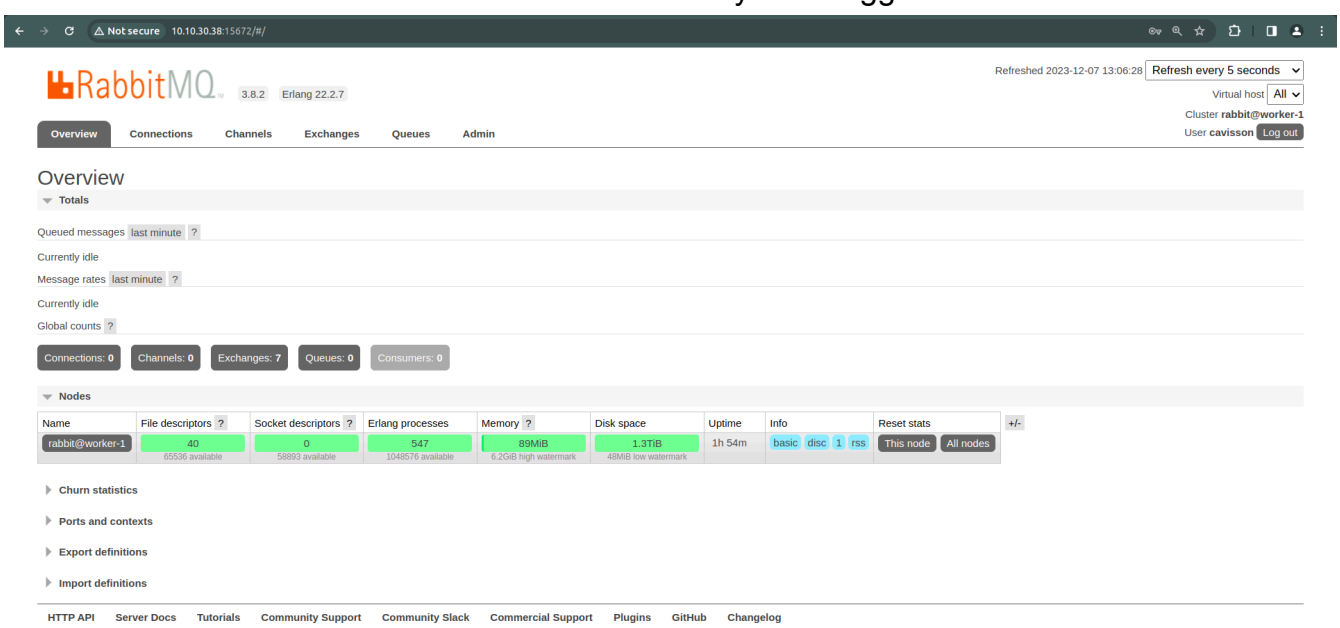
```
cavisson@worker-1:~/RabbitMQ_ssl$ sudo rabbitmq-plugins enable rabbitmq_management
Enabling plugins on node rabbit@worker-1:
rabbitmq_management
The following plugins have been configured:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch
Applying plugin configuration to rabbit@worker-1...
Plugin configuration unchanged.
```

9. We open the following URL in our preferred web browser to access RabbitMQ's administrative interface: http://Your_Server_IP:15672/

   You will see this login page:



   You will then see the administrator dashboard once you've logged in:



10. ***Generate self-signed certificates for client and server :-***

For enabling TLS/SSL, we need certificates that allow Certificate/Key pairs. This can be achieved with the help of tools like OpenSSL that generate self-signed certificates for a client and a server.

> Generating our own CA certificate.
We will now use OpenSSL to create all the required keys and certificates. Let's start creating the CA certificates. Make sure all certificates are placed in the below directory:
cd /etc/pki/tls

The step first is to make the root private key with the help of the below command:
sudo openssl genrsa -out RMQ-CA-Key.pem

Our next step is to sign the certificate.
sudo openssl req -new -key RMQ-CA-Key.pem -x509 -days 100 -out RMQ-CA-cert.pem

After running the above command, an interactive script will start and ask for various bits of information. We will up all the required details here.

An important thing to take care of while entering **Common-Name (CN) is the need to place a common name as a server IP(10.10.10.10)**. Once completed, this will generate a root CA certificate.

Now, we are going to create a server key and server certificate.

Please note: *We will need RMQ-CA-Key.pem and RMQ-CA-cert.pem later.*

> Generate a server key
sudo openssl genrsa -out RMQ-server-key.pem

> Generate a CSR (Certificate Signing Request):

sudo openssl req -new -key RMQ-server-key.pem -out RMQ-signingrequest.csr

When we run the above command, it will ask for various parameters (Country, State, ON, OU, etc.) **We also need to select a common name. This should preferably be the server's IP.**

> Generate the self-signed certificate using signing.csr, CA-Key.pem, and CA-cert.pem.

sudo openssl x509 -req -days 100 -in RMQ-signingrequest.csr -CA RMQ-CA-cert.pem -CAkey RMQ-CA-Key.pem -CAcreateserial -out RMQ-server-cert.pem

The above command creates a signed server certificate.

We must then concatenate both private and public certificates to create a .pem file:

sudo cat RMQ-server-key.pem RMQ-server-cert.pem > RMQ-serverpemkeyfile.pem

Please note: *Change the permissions of the files RMQ-server-key.pem, RMQ-server-cert.pem, and RMQ-serverpemkeyfile.pem to make them modifiable*

Finally, the required keys are generated. We will need to use them to set up TLS/SSL.

11. Enable TLS/SSL support in a RabbitMQ server

To enable TLS support in RabbitMQ, we need to create a configuration file at the location below with the name rabbitmq.conf:

cd /etc/rabbitmq

Then, paste the below configuration code into the file:

```
listeners.tcp.default = 5672
listeners.ssl.default = 5671
ssl_options.cacertfile = /etc/pki/tls/RMQ-CA-cert.pem
ssl_options.certfile = /etc/pki/tls/RMQ-server-cert.pem
ssl_options.keyfile = /etc/pki/tls/RMQ-server-key.pem
ssl_options.verify = verify_peer
ssl_options.fail_if_no_peer_cert = false

management.tcp.port = 15672
management.ssl.port = 15671
management.ssl.cacertfile = /etc/pki/tls/RMQ-CA-cert.pem
```

management.ssl.certfile = /etc/pki/tls/RMQ-server-cert.pem
management.ssl.keyfile = /etc/pki/tls/RMQ-server-key.pem
management.ssl.verify = verify_peer
management.ssl.fail_if_no_peer_cert = false

We then restart the service with the following command:

sudo systemctl restart rabbitmq-server.service
sudo systemctl status rabbitmq-server.service

After restarting to verify if TLS/SSL is set up successfully, we check our logs with the below command:

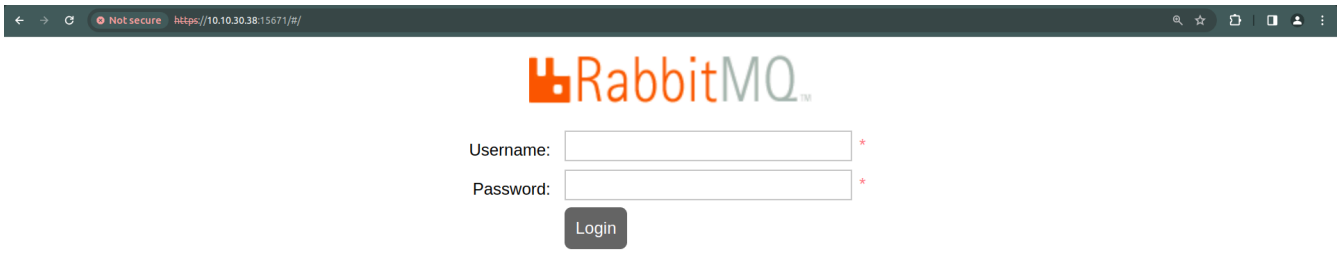
sudo cat /var/log/rabbitmq/rabbit@ip-10–10–10–10.log

```
2023-12-07 13:55:12.381 [info] <0.329.0> Running boot step delegate_sup defined by app rabbit
2023-12-07 13:55:12.382 [info] <0.329.0> Running boot step rabbit_memory_monitor defined by app rabbit
2023-12-07 13:55:12.383 [info] <0.329.0> Running boot step core_initialized defined by app rabbit
2023-12-07 13:55:12.383 [info] <0.329.0> Running boot step upgrade_queues defined by app rabbit
2023-12-07 13:55:12.406 [info] <0.329.0> Running boot step rabbit_connection_tracking defined by app rabbit
2023-12-07 13:55:12.407 [info] <0.329.0> Running boot step rabbit_connection_tracking_handler defined by app rabbit
2023-12-07 13:55:12.407 [info] <0.329.0> Running boot step rabbit_exchange_parameters defined by app rabbit
2023-12-07 13:55:12.407 [info] <0.329.0> Running boot step rabbit_mirror_queue_misc defined by app rabbit
2023-12-07 13:55:12.407 [info] <0.329.0> Running boot step rabbit_policies defined by app rabbit
2023-12-07 13:55:12.408 [info] <0.329.0> Running boot step rabbit_policy defined by app rabbit
2023-12-07 13:55:12.408 [info] <0.329.0> Running boot step rabbit_queue_location_validator defined by app rabbit
2023-12-07 13:55:12.408 [info] <0.329.0> Running boot step rabbit_quorum_memory_manager defined by app rabbit
2023-12-07 13:55:12.408 [info] <0.329.0> Running boot step rabbit_vhost_limit defined by app rabbit
2023-12-07 13:55:12.408 [info] <0.329.0> Running boot step rabbit_mgmt_reset_handler defined by app rabbitmq_management
2023-12-07 13:55:12.408 [info] <0.329.0> Running boot step rabbit_mgmt_db_handler defined by app rabbitmq_management_agent
2023-12-07 13:55:12.408 [info] <0.329.0> Management plugin: using rates mode 'basic'
2023-12-07 13:55:12.409 [info] <0.329.0> Running boot step recovery defined by app rabbit
2023-12-07 13:55:12.409 [info] <0.408.0> Making sure data directory '/var/lib/rabbitmq/mnesia/rabbit@worker-1/msg_stores/vhosts/628WB79CIFDYO9LJI6DKMI09L' for vhost '/' exists
2023-12-07 13:55:12.470 [info] <0.408.0> Starting message stores for vhost '/'
2023-12-07 13:55:12.471 [info] <0.412.0> Message store "628WB79CIFDYO9LJI6DKMI09L/msg_store_transient": using rabbit_msg_store_ets_index to provide index
2023-12-07 13:55:12.490 [info] <0.408.0> Started message store of type transient for vhost '/'
2023-12-07 13:55:12.490 [info] <0.415.0> Message store "628WB79CIFDYO9LJI6DKMI09L/msg_store_persistent": using rabbit_msg_store_ets_index to provide index
2023-12-07 13:55:12.492 [info] <0.408.0> Started message store of type persistent for vhost '/'
2023-12-07 13:55:12.495 [info] <0.329.0> Running boot step load_core_definitions defined by app rabbit
2023-12-07 13:55:12.495 [info] <0.329.0> Running boot step empty_db_check defined by app rabbit
2023-12-07 13:55:12.495 [info] <0.329.0> Running boot step rabbit_looking_glass defined by app rabbit
2023-12-07 13:55:12.495 [info] <0.329.0> Running boot step rabbit_core_metrics_gc defined by app rabbit
2023-12-07 13:55:12.496 [info] <0.329.0> Running boot step background_gc defined by app rabbit
2023-12-07 13:55:12.496 [info] <0.329.0> Running boot step connection_tracking defined by app rabbit
2023-12-07 13:55:12.496 [info] <0.329.0> Setting up a table for connection tracking on this node: 'tracked_connection_on_node_rabbit@worker-1'
2023-12-07 13:55:12.496 [info] <0.329.0> Setting up a table for per-vhost connection counting on this node: 'tracked_connection_per_vhost_on_node_rabbit@worker-1'
2023-12-07 13:55:12.496 [info] <0.329.0> Running boot step routing_ready defined by app rabbit
2023-12-07 13:55:12.496 [info] <0.329.0> Running boot step pre_flight defined by app rabbit
2023-12-07 13:55:12.496 [info] <0.329.0> Running boot step notify_cluster defined by app rabbit
2023-12-07 13:55:12.497 [info] <0.329.0> Running boot step networking defined by app rabbit
2023-12-07 13:55:12.498 [info] <0.452.0> started TCP listener on [::]:5672
2023-12-07 13:55:12.500 [info] <0.468.0> started TLS (SSL) listener on [::]:5671
2023-12-07 13:55:12.500 [info] <0.329.0> Running boot step cluster_name defined by app rabbit
2023-12-07 13:55:12.500 [info] <0.329.0> Running boot step direct_client defined by app rabbit
2023-12-07 13:55:12.524 [info] <0.518.0> Management plugin: HTTP (non-TLS) listener started on port 15672
2023-12-07 13:55:12.551 [info] <0.518.0> Management plugin: HTTPS listener started on port 15671
2023-12-07 13:55:12.551 [info] <0.728.0> Statistics database started.
2023-12-07 13:55:12.551 [info] <0.727.0> Starting worker pool 'management_worker_pool' with 3 processes in it
2023-12-07 13:55:13.734 [notice] <0.106.0> Changed loghwm of /var/log/rabbitmq/rabbit@worker-1.log to 50
2023-12-07 13:55:13.826 [info] <0.8.0> Server startup complete; 3 plugins started.
 * rabbitmq_management
 * rabbitmq_web_dispatch
 * rabbitmq_management_agent
```

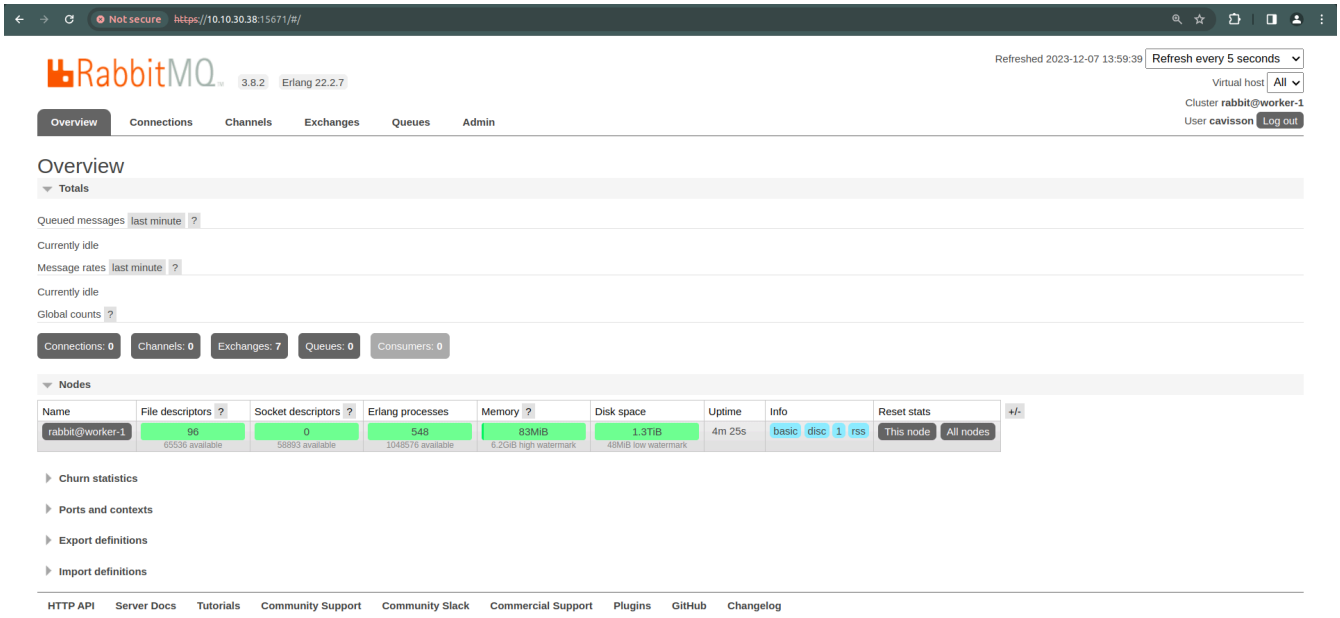
Looking closely at the code above, we can see that we have successfully created our RabbitMQ server with SSL/TLS enabled.

12. We open the following URL in our preferred web browser to access RabbitMQ's administrative interface: https://Your_Server_IP:15671/

You will see this login page:



You will then see the administrator dashboard once you've logged in:



RabbitMQ Server setup with docker compose

1. We can configure RabbitMQ to listen on both SSL and non-SSL ports simultaneously. Here docker-compose.yml file includes both configurations:

```
version: '3'

services:
  rabbitmq:
    image: "rabbitmq:3-management"
    ports:
      - "5672:5672"   # Non-SSL port
      - "5671:5671"   # SSL port
      - "15672:15672" # Non-SSL management console
      - "15671:15671" # SSL-enabled management console
    environment:
      RABBITMQ_DEFAULT_USER: "your_username"
      RABBITMQ_DEFAULT_PASS: "your_password"
      RABBITMQ_SSL_CACERTFILE: "/etc/rabbitmq/certs/ca_certificate.pem"
      RABBITMQ_SSL_CERTFILE: "/etc/rabbitmq/certs/server_certificate.pem"
      RABBITMQ_SSL_KEYFILE: "/etc/rabbitmq/certs/server_key.pem"
    volumes:
      - ./certs:/etc/rabbitmq/ssl
```

   This configuration opens both non-SSL ports (5672 and 15672) and SSL ports (5671 and 15671). Adjust the environment section and SSL certificate paths as necessary.

2. After configuring the docker-compose.yml file, run:

   sudo docker-compose up -d

3. To stop RabbitMQ, use:

   sudo docker-compose down

   Now, we can access both the non-SSL and SSL-enabled management consoles:

   Non-SSL: http://localhost:15672/
   SSL: https://localhost:15671/

   Additionally, We can use both non-SSL and SSL connections to RabbitMQ on ports 5672 and 5671, respectively, in our applications.

Remember to replace "your_username" and "your_password" with the desired username and password, and ensure that our SSL certificates are valid and properly configured.

Note :- *Make sure to replace "your_username" and "your_password" with the desired username and password. Also, create a "certs" directory in the same folder as your docker-compose.yml file, and place your SSL certificate files (ca_certificate.pem, server_certificate.pem, and server_key.pem) in that directory.*