Université Hassan 1er

Ecole Nationale des Sciences Appliquées de Berrechid

Département de mathématique et informatique

Filière : Ingénierie des Systèmes d'Information et BIG DATA

Module : Grandes masses de données sur Cloud

Semestre : S7

Rapport du projet :

Réalisé par :

- MOUSSAOUI Othman
- SALIME MEHDI

Encadré par :

- Pr.KARIM Lamia

Année universitaire : 2022-202

Table de matière :

Introduction :

Les brokers de messages sont des applications intermédiaires qui jouent un rôle clé dans la communication fiable et stable entre différents services. Ils peuvent stocker les demandes entrantes dans une file d'attente et les servir une par une aux services récepteurs. En découplant vos services de cette façon, vous les rendez plus évolutifs et performants.

Pour notre projet, nous avons utilisé RabbitMQ comme solution de gestion de messages pour faciliter la communication entre différentes parties de notre application. Nous avons mis en place un système de files d'attente pour stocker les demandes entrantes et les traiter de manière asynchrone, ce qui a permis d'améliorer la scalabilité et la performance de notre application.

Grâce à RabbitMQ, nous avons également pu mettre en place un système de publication-abonnement qui nous a permis de diffuser facilement des mises à jour à différents subscribers de manière efficace.

En utilisant RabbitMQ, nous avons pu construire une application plus robuste et évolutive, capable de gérer des volumes importants de trafic de manière fiable. Nous sommes convaincus que cet outil a joué un rôle clé dans la réussite de notre projet.

Qu'est-ce que RabbitMq ?

RabbitMQ est un broker de messages open source populaire qui stocke et transmet des messages asynchrones entre deux ou plusieurs services selon des règles prédéfinies. C'est un logiciel intermédiaire qui garantit que vos systèmes sont plus fiables, évolutifs et toujours disponibles. Par exemple, RabbitMQ peut être utilisé pour réduire la charge système en déléguant des tâches lourdes à un autre service qui est inactif au moment donné.

Pour mieux comprendre les spécificités de RabbitMQ, familiarisons-nous avec quelques-uns de ses concepts les plus importants :

- **Producteur** - application cliente qui crée des messages et les livre à un exchange.
- **Consommateur** - une application qui se connecte à la file d'attente et s'abonne aux messages à traiter.
- **Exchange** - agent de routage de messages qui est responsable de recevoir les messages d'un producteur et de les pousser vers une ou plusieurs files d'attente en fonction des règles définies par les liens.
- **Liens** - règles que les échanges utilisent pour acheminer les messages vers les files d'attente.
- **File d'attente** - grand tampon qui stocke les messages et est lié à au moins un exchange. Les consommateurs et les producteurs ont une relation de n-à-1 avec une file d'attente.
- **Message** - BLOB binaire de données qui peuvent être utiles pour l'application consommatrice, y compris les déclencheurs de processus, l'état des tâches, les messages de texte simples, etc.
- **Hôte virtuel** - environnement virtuel qui isole différents producteurs, consommateurs, échanges, files d'attente et autres objets sur un serveur RabbitMQ.

Comment fonctionne RabbitMq ?

RabbitMQ permet de découpler différents services en gérant leur communication à travers les messages qu'ils envoient l'un à l'autre. RabbitMQ est basé sur le protocole Advanced Message Queuing (AMQP) qui donne aux développeurs beaucoup de flexibilité pour définir des entités et des schémas de routage au niveau de l'application.

Examinons un cycle de travail standard de RabbitMQ. Une application productrice publie d'abord un message sur un certain type d'échange. L'échange est lié à une ou plusieurs files d'attente. Lorsque l'échange reçoit un message, il prend en compte différents attributs de

message et achemine le message vers une ou plusieurs files d'attente en conséquence. Le message reste alors dans une file d'attente jusqu'à ce qu'il soit pris par un consommateur.

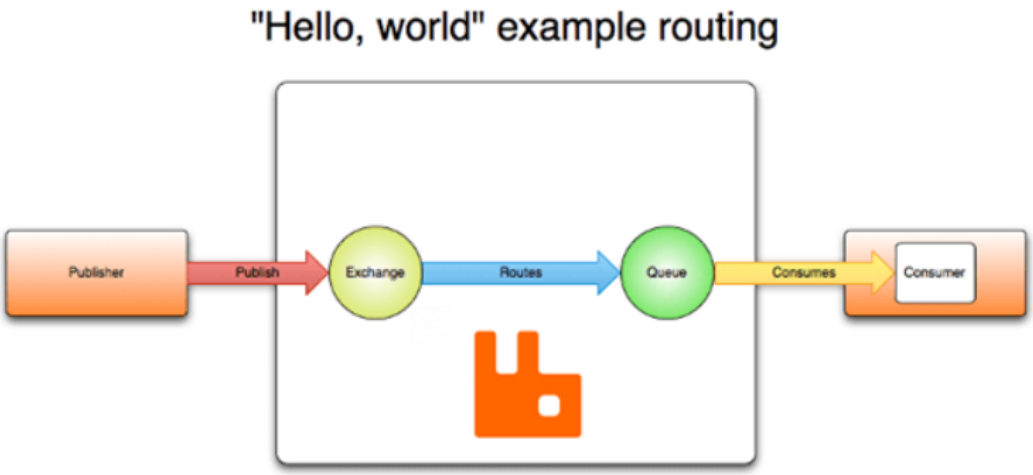


Figure1 : un schéma de routage de message simple

Types d'échanges RabbitMQ :

Un producteur ne peut envoyer des messages qu'à un échange qui est responsable de pousser des messages vers différentes files d'attente à l'aide d'attributs d'en-tête, de liaisons et de clés de routage. Dans RabbitMQ, il existe quatre types distincts d'échanges qui acheminent les messages selon différents paramètres et configurations de liaison.

- **Échange direct :**

Avec un échange direct, un message est acheminé vers une ou plusieurs files d'attente avec une clé de liaison qui correspond exactement à la clé de routage du message. La clé de routage est ajoutée à l'en-tête du message par l'application productrice. Si le message ne correspond à aucune clé de liaison, il est jeté.

Par exemple, vous pouvez utiliser un échange direct pour répartir les messages entre plusieurs consommateurs de manière à la ronde et utiliser plusieurs travailleurs pour la même tâche.

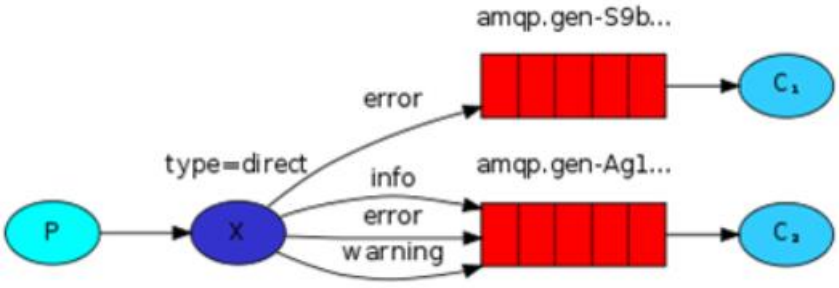


Figure 2 : sur l'image ci-dessus, vous pouvez voir un producteur (P) connecté à un échange direct (X) qui possède quatre liaisons de correspondance directe ("error", "info", "warning") vers deux files d'attente utilisées chacune par un consommateur différent (C).

- **Échange de sujet :**

Avec un échange de sujet, un message est acheminé vers une ou plusieurs files d'attente avec un modèle de liaison générique qui correspond à la clé de routage du message. Une liste de mots dans un modèle de routage sont délimités par des points, un astérisque (*) représente un seul mot, tandis qu'un dièse (#) représente zéro ou plusieurs mots.

Les échanges de sujet sont très flexibles et ont de nombreux cas d'utilisation pour les problèmes où plusieurs applications choisissent sélectivement quel type de messages elles souhaitent consommer.

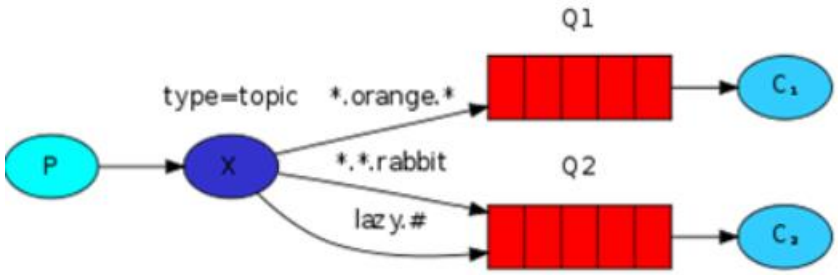


Figure3 : sur l'image ci-dessus, vous pouvez voir un producteur (P) connecté à un échange de sujet (X) qui possède trois liaisons ("orange.", "..rabbit", "lazy.#") vers deux files d'attente, chacune utilisée par un consommateur différent (C).

- **Échange Fanout :**

Avec un échange Fanout, un message est simplement poussé vers toutes les files d'attente liées à l'échange Fanout, indépendamment des clés de routage ou du matching de motif. L'échange Fanout est idéal lorsque vous souhaitez toujours diffuser certains messages vers les mêmes consommateurs multiples.

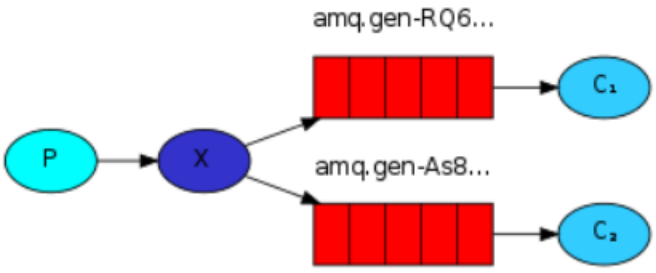


Figure 4 : Sur l'image ci-dessus, vous pouvez voir un producteur (P) connecté à un échange direct (X) qui possède deux liaisons directes vers deux files d'attente, chacune utilisée par un consommateur différent (C).

- **Échange d'en-têtes :**

Avec un échange d'en-têtes, un message est acheminé en utilisant des attributs d'en-tête de message au lieu des clés de routage. Vous pouvez utiliser plus d'une clé d'en-tête - non seulement des chaînes de caractères, mais également des entiers, des hachages, etc. - pour faire correspondre les messages avec les files d'attente.

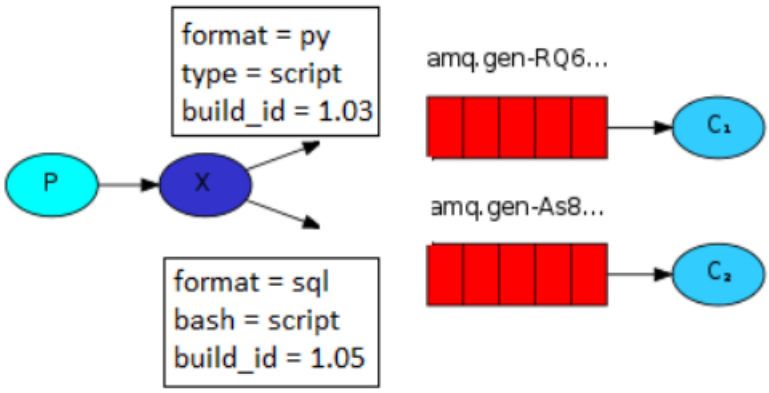


Figure 5 : Sur l'image ci-dessus, vous pouvez voir un producteur (P) connecté à un échange de message (X) qui possède deux liaisons avec un en-tête de message unique vers deux files d'attente, chacune utilisée par un consommateur différent (C).

Avantages de l'utilisation de RabbitMq :

En dehors de la réception de réponses rapides des serveurs Web, voici quelques autres avantages de l'utilisation de RabbitMQ :

- Déploiement distribué- Vous pouvez facilement déployer le logiciel léger en grappes fédérées sur plusieurs zones, offrant ainsi une haute disponibilité et une meilleure capacité de traitement.
- Outils de développement et greffons - Il offre une large gamme d'outils de développement pour les langages populaires et de greffons pour l'intégration continue et les métriques opérationnelles.
- Léger et facile à déployer - RabbitMQ nécessite un minimum de RAM ce qui en fait l'un des brokers de message les plus légers et qui peut être facilement déployé sur les nuages publics et privés.
- Messagerie asynchrone - Il prend en charge plusieurs types d'échange, l'acquittement de livraison, la mise en file d'attente de messages et les protocoles de messagerie.
- Fiable - RabbitMQ offre plusieurs fonctionnalités, notamment l'acquittement de livraison, la persistance et la haute disponibilité, qui aident les utilisateurs à échanger des performances contre de la fiabilité.

- Évolutif et flexible - RabbitMQ est principalement responsable de la communication, et le reste (maintien des consommateurs et des éditeurs) est géré par les développeurs.

Prend en charge plusieurs protocoles - Un autre avantage de RabbitMQ est qu'il permet aux utilisateurs d'envoyer des messages sur plusieurs protocoles de messagerie.

Protocole AMQP (Advanced Message Queuing Protocol) :

AMQP (Advanced Message Queuing Protocol) est un protocole de messagerie ouvert qui permet à différents systèmes de communiquer entre eux de manière fiable et asynchrone. Il est principalement utilisé pour la mise en file d'attente de messages et la diffusion de messages à un groupe de consommateurs.

RabbitMQ est une implémentation populaire d'un serveur de file d'attente de messages qui utilise le protocole AMQP.

Voici comment fonctionne le protocole AMQP lorsqu'il est utilisé avec RabbitMQ :

1. Un producteur envoie un message à une file d'attente de messages gérée par RabbitMQ.
2. RabbitMQ stocke le message dans la file d'attente jusqu'à ce qu'un consommateur soit disponible pour le traiter.
3. Lorsqu'un consommateur est disponible, RabbitMQ envoie le message à ce dernier.
4. Le consommateur traite le message et envoie une notification à RabbitMQ indiquant que le message a été traité.
5. RabbitMQ supprime le message de la file d'attente.

Le protocole AMQP utilise également des échanges et des liaisons pour acheminer les messages vers les files d'attente de manière plus flexible. Les échanges sont des points de terminaison qui reçoivent les messages du producteur et les acheminent vers les files d'attente en fonction de règles de routage configurables. Les liaisons sont utilisées pour lier les échanges aux files d'attente de manière à ce que les messages puissent être acheminés vers les files d'attente correctes.

En utilisant ces éléments de base, le protocole AMQP permet de mettre en place des architectures de messagerie flexibles et évolutives pour une grande variété d'applications.

L'installation de RabbitMq :

Dans cet article, nous allons vous montrer un guide étape par étape sur la façon d'installer et de configurer un serveur RabbitMQ sur Ubuntu 22.04 pour vous aider à démarrer et à utiliser cet excellent logiciel.

Pour suivre ce guide, vous devez répondre aux exigences suivantes :

- Une instance d'Ubuntu 22.04.
- Un utilisateur avec des privilèges sudo
- Python 3 avec le gestionnaire de paquets pip installé

Étape 1 : Installer le serveur RabbitMQ

Tout d'abord, installons les prérequis :

```
apt-get install curl gnupg apt-transport-https -y
```

Nous sommes maintenant prêts à ajouter des clés de signature de référentiel pour les référentiels RabbiMQ main, ErLang et RabbitMQ PackageCloud respectivement :

```
curl -1sLf "https://keys.openpgp.org/vks/v1/by-fingerprint/0A9AF2115F4687BD29803A206B73A36E6026DFCA" | sudo gpg --dearmor | sudo tee /usr/share/keyrings/com.rabbitmq.team.gpg > /dev/null
curl -1sLf "https://keyserver.ubuntu.com/pks/lookup?op=get&search=0xf77f1eda57ebb1cc" | sudo gpg --dearmor | sudo tee /usr/share/keyrings/net.launchpad.ppa.rabbitmq.erlang.gpg > /dev/null
curl -1sLf "https://packagecloud.io/rabbitmq/rabbitmq-server/gpgkey" | sudo gpg --dearmor | sudo tee /usr/share/keyrings/io.packagecloud.rabbitmq.gpg > /dev/null
```

Créez un nouveau fichier sur /etc/apt/sources.list.d/rabbitmq.listet ajoutez les référentiels suivants pour ErLang et RabbitMQ, respectivement, adaptés à la jammyversion Ubuntu 22.04 :

```
deb [signed-by=/usr/share/keyrings/net.launchpad.ppa.rabbitmq.erlang.gpg] http://ppa.launchpad.net/rabbitmq/rabbitmq-erlang/ubuntu jammy main
deb-src [signed-by=/usr/share/keyrings/net.launchpad.ppa.rabbitmq.erlang.gpg] http://ppa.launchpad.net/rabbitmq/rabbitmq-erlang/ubuntu jammy main
deb [signed-by=/usr/share/keyrings/io.packagecloud.rabbitmq.gpg] https://packagecloud.io/rabbitmq/rabbitmq-server/ubuntu/ jammy main
deb-src [signed-by=/usr/share/keyrings/io.packagecloud.rabbitmq.gpg] https://packagecloud.io/rabbitmq/rabbitmq-server/ubuntu/ jammy main
```

Enregistrez le fichier et vous êtes prêt à mettre à jour vos listes de référentiel :

```
apt-get update -y
```

Une fois vos listes de référentiels mises à jour, poursuivez l'installation des packages ErLang requis :

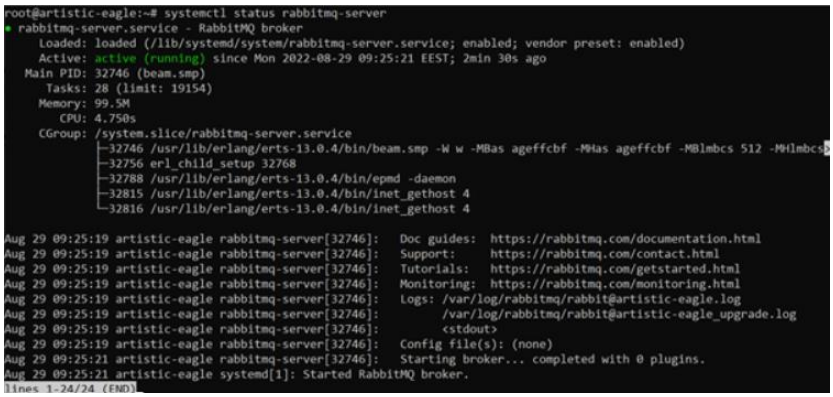
```
apt-get install -y erlang-base \
   erlang-asn1 erlang-crypto erlang-eldap erlang-ftp erlang-inets \
   erlang-mnesia erlang-os-mon erlang-parsetools erlang-public-key \
   erlang-runtime-tools erlang-snmp erlang-ssl \
   erlang-syntax-tools erlang-tftp erlang-tools erlang-xmerl
```

Enfin, nous pouvons installer le serveur RabbitMQ et ses dépendances :

```
apt-get install rabbitmq-server -y --fix-missing
```

Si tout s'est bien passé, vous devriez voir un processus rabbitmq-server en cours d'exécution :

```
systemctl status rabbitmq-server
```

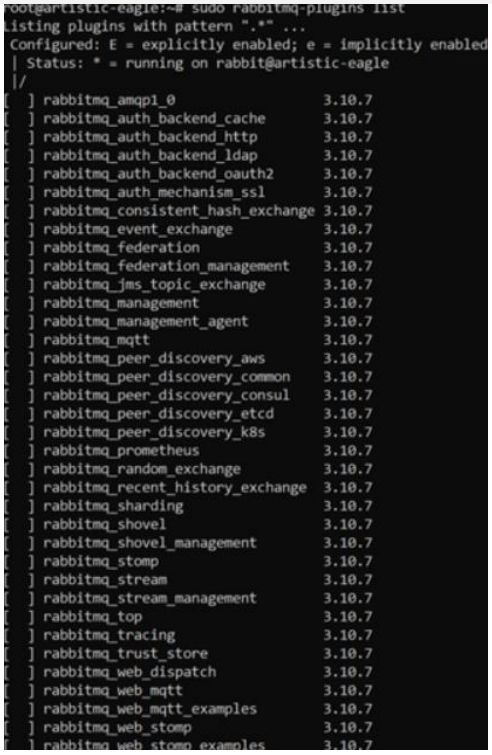


Félicitations, vous avez maintenant installé RabbitMQ avec succès ! Il est maintenant temps d'apprendre à l'utiliser.

Étape 2 : Activer la console de gestion RabbitMQ

RabbitMQ dispose d'un plug-in de console de gestion qui vous permet d'effectuer diverses tâches de gestion et de surveillance via une interface Web. Vous pouvez gérer les échanges, les files d'attente, les liaisons, les utilisateurs et d'autres objets RabbitMQ, ainsi que surveiller des éléments tels que l'utilisation de la mémoire, les débits de messages, les connexions et d'autres processus.
Pour vérifier la liste de tous les plugins RabbitMQ disponibles, exécutez la commande suivante :

```
rabbitmq-plugins list
```



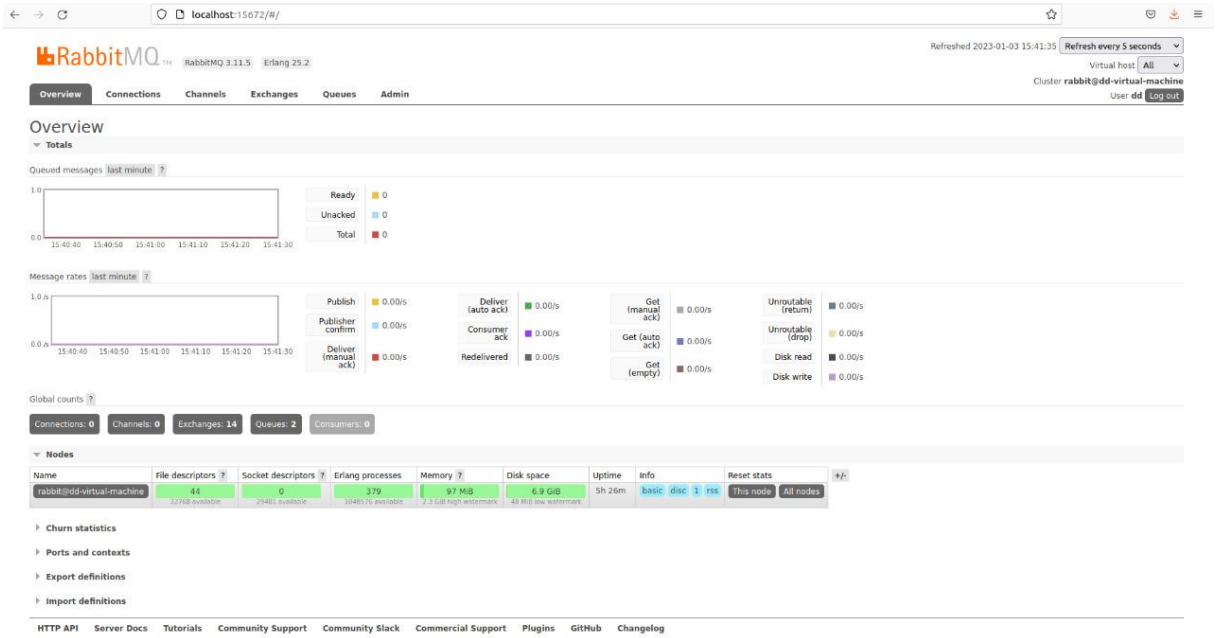
Comme vous pouvez le voir, tous les plugins sont actuellement désactivés. Vous pouvez activer le plugin de gestion RabbitMQ en utilisant la commande suivante :

```
rabbitmq-plugins enable rabbitmq_management
```

```
root@artistic-eagle:~# rabbitmq-plugins enable rabbitmq_management
Enabling plugins on node rabbit@artistic-eagle:
rabbitmq_management
The following plugins have been configured:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch
Applying plugin configuration to rabbit@artistic-eagle...
The following plugins have been enabled:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch

started 3 plugins.
```

Vous pouvez maintenant vous connecter à l'interface Web de RabbitMQ. Pour y accéder, ouvrez votre navigateur Web et tapez l'URL http://your-server-ip:15672:



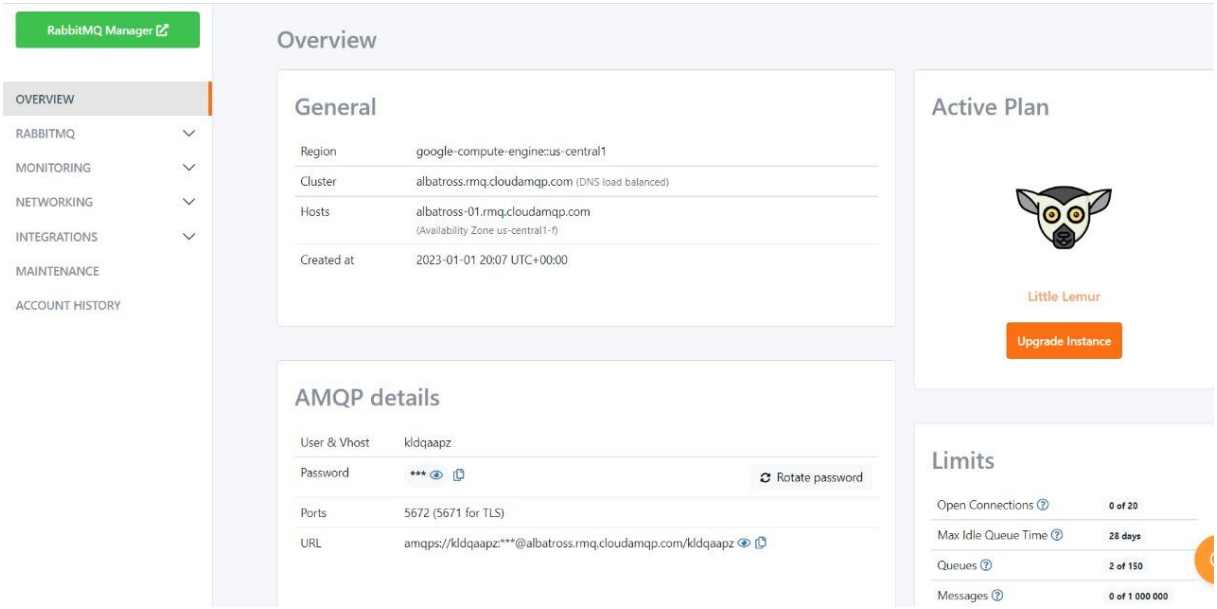
Cloud AMQPS :

CloudAMQP est un service de file d'attente de messages géré qui fournit un serveur RabbitMQ en tant que service. Il permet aux développeurs d'intégrer facilement la messagerie dans leurs applications sans avoir à gérer et à maintenir leur propre serveur RabbitMQ.

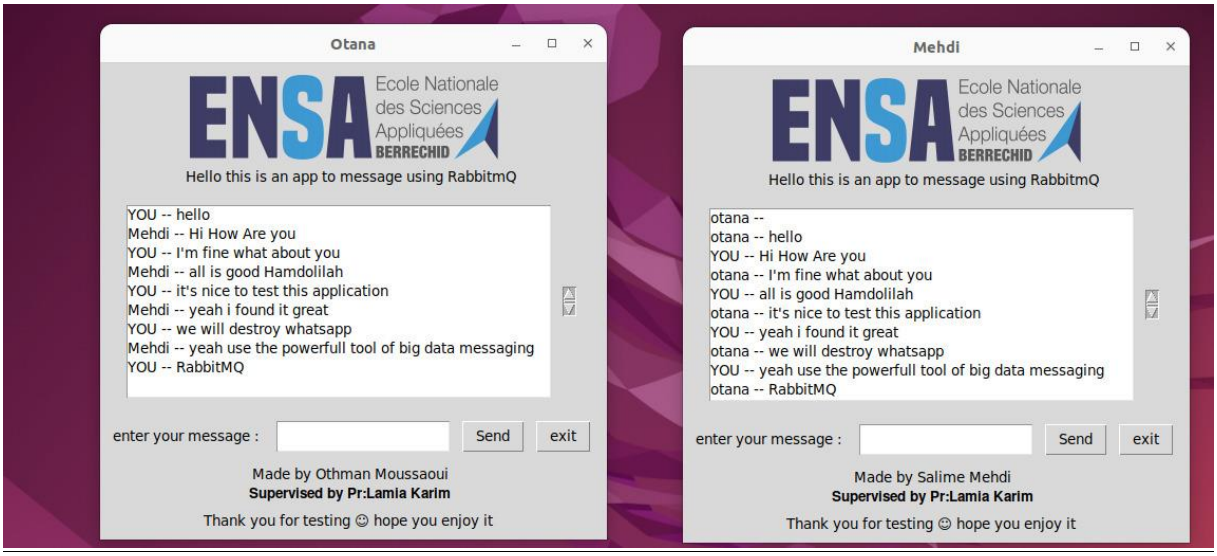
Avec CloudAMQP, vous pouvez créer une instance RabbitMQ et la configurer selon vos besoins spécifiques. Vous pouvez choisir la taille, l'emplacement et le type de plan de l'instance, ainsi que configurer des options telles que les hôtes virtuels, les utilisateurs et les autorisations. CloudAMQP fournit une console de gestion basée sur le web pour administrer

votre instance RabbitMQ, ainsi que diverses bibliothèques client pour différents langages de programmation pour faciliter l'intégration de la messagerie dans vos applications.

CloudAMQP est particulièrement utile pour les développeurs qui souhaitent utiliser RabbitMQ dans leurs applications, mais qui ne veulent pas gérer et maintenir un serveur RabbitMQ. C'est également une bonne option pour les applications qui nécessitent un haut niveau de fiabilité et de scalabilité, car CloudAMQP gère l'infrastructure et l'entretien du serveur pour vous.



Application du chat réalisé :



Les principales fonctions utilisées par le producteur et le consommateur :

```
import pika
import time
from threading import Thread
from tkinter import *
from tkinter import messagebox
from PIL import Image,ImageDraw,ImageGrab,ImageTk
url = "amqps://kldqaapz:lqFOJ_g3ElJAq3vpYvR56ajj1Rkkog0t@albatross.rmq.cloudamqp.com/kldqaapz"
params = pika.URLParameters(url)
params.socket_timeout = 5
#fonction of receiver
def receiver():
    def callback(ch, method, propreties, body):
        msg_list.insert(END, "Mehdi -- "+ body.decode()
    connection = pika.BlockingConnection(params)
    channel = connection.channel()
    channel.queue_declare(queue='task_queue2')
    if (channel.basic_consume(queue='task_queue2', on_message_callback= callback, auto_ack=True)):
        msg_list.insert(END,"welcome again this is the box of messages... ")
        time.sleep(2)
        messagebox.showinfo("boom", " You'r online!")
        msg_list.delete(0,END)
    channel.start_consuming()
    connection.close()
#function of send
def send():
    connection = pika.BlockingConnection(params)
```

```
#connection = pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
channel = connection.channel()
channel.queue_declare(queue='task_queue1')
live = entry_field.get()
msg_list.insert(END, "YOU -- " + live)
channel.basic_publish(exchange='', routing_key='task_queue1', body= live)
connection.close()
```

Conclusion :

RabbitMQ est un broker de messages open source qui permet aux applications de communiquer de manière fiable et asynchrone en utilisant des files d'attente et des échanges. Les producteurs sont des applications clientes qui créent et envoient des messages à un échange, qui est un agent de routage de messages. L'échange utilise des règles de liens pour acheminer les messages vers une ou plusieurs files d'attente, qui sont des tampons qui stockent les messages. Les consommateurs sont des applications qui se connectent à la file d'attente et s'abonnent aux messages à traiter. RabbitMQ est basé sur le protocole AMQP et permet aux développeurs de définir des entités et des schémas de routage au niveau de l'application. Il est utilisé pour améliorer la scalabilité et la performance des applications en découplant les services et en gérant leur communication à travers les messages.