**TP 5 : Clustering**

Nous allons créer un cluster RabbitMQ avec Docker, nous allons utiliser Docker Compose pour orchestrer plusieurs conteneurs RabbitMQ.

**Étape 1 : Installer Docker et Docker Compose**

Assurez-vous que Docker et Docker Compose sont installés sur votre machine.

**Étape 2 : Créer le fichier docker-compose.yml**

Créez un fichier docker-compose.yml dans un répertoire de votre choix. Voici un exemple de fichier pour monter un cluster RabbitMQ avec trois nœuds :

version: '3'

services:

rabbitmq1:

image: rabbitmq:3-management

container\_name: rabbitmq1

hostname: rabbitmq1

environment:

- RABBITMQ\_ERLANG\_COOKIE=secretcookie

ports:

- "15672:15672"

- "5672:5672"

networks:

- rabbitmq\_network

rabbitmq2:

image: rabbitmq:3-management

container\_name: rabbitmq2

hostname: rabbitmq2

environment:

- RABBITMQ\_ERLANG\_COOKIE=secretcookie

networks:

- rabbitmq\_network

rabbitmq3:

image: rabbitmq:3-management

container\_name: rabbitmq3

hostname: rabbitmq3

environment:

- RABBITMQ\_ERLANG\_COOKIE=secretcookie

networks:

- rabbitmq\_network

networks:

rabbitmq\_network:

driver: bridge

**Étape 3 : Démarrer les conteneurs RabbitMQ**

Exécutez la commande suivante pour démarrer les conteneurs définis dans le fichier docker-compose.yml :

docker-compose up -d

**Étape 4 : Configurer le Clustering**

Une fois les conteneurs en cours d'exécution, vous devez joindre les nœuds RabbitMQ pour former un cluster.

Accédez au conteneur du premier nœud :

docker exec -it rabbitmq1 /bin/bash

Configurer le premier nœud (nœud principal) :

rabbitmqctl stop\_app

rabbitmqctl reset

rabbitmqctl start\_app

exit

Joindre les autres nœuds au cluster :

Répétez les étapes suivantes pour rabbitmq2 et rabbitmq3 :

docker exec -it rabbitmq2 /bin/bash

rabbitmqctl stop\_app

rabbitmqctl reset

rabbitmqctl join\_cluster rabbit@rabbitmq1

rabbitmqctl start\_app

exit

docker exec -it rabbitmq3 /bin/bash

rabbitmqctl stop\_app

rabbitmqctl reset

rabbitmqctl join\_cluster rabbit@rabbitmq1

rabbitmqctl start\_app

exit

**Étape 5 : Vérifier le Cluster**

Vérifiez que le cluster est correctement configuré en exécutant la commande suivante sur l'un des nœuds :

docker exec -it rabbitmq1 rabbitmqctl cluster\_status

Vous devriez voir les trois nœuds listés dans le cluster.

**Étape 6 : Configurer les Queues Mirroring :**

Pour configurer le mirroring des queues, utilisez la commande suivante pour définir une politique de haute disponibilité :

Accédez à l'un des nœuds (par exemple, rabbitmq1) :

docker exec -it rabbitmq1 /bin/bash

**Définir la politique de mirroring :**

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

**Tester votre cluster :**

**Étape 1 : Accéder à l'Interface de Gestion**

Accédez à l'interface de gestion RabbitMQ en ouvrant votre navigateur et en naviguant vers http://adresse\_ip:15672. Utilisez les identifiants de l’utilisaterur admin créé pour le nœud.

**Étape 2 : Vérifier l'état du cluster**

Connectez-vous à l'interface de gestion.

Allez dans l'onglet "Overview".

Vous devriez voir les trois nœuds listés dans la section "Nodes".

Étape 3 : Créer une queue avec mirroring

Dans l'interface de gestion, allez à l'onglet "Queues".

Cliquez sur "Add a new queue".

Donnez un nom à la queue (par exemple, test\_queue).

Cochez la case "Durable" pour s'assurer que la queue persiste.

Sous "Policy", assurez-vous que la politique de mirroring (ha-all) s'applique à la queue.

Cliquez sur "Add queue".

**Étape 4 : Publier des messages**

Sélectionnez la queue que vous avez créée (test\_queue).

Allez dans la section "Publish message".

Saisissez un message dans le champ "Payload".

Cliquez sur "Publish message".

Répétez cette opération pour envoyer plusieurs messages.

**Étape 5 : Consommer des messages**

Toujours dans l'onglet "Queues", sélectionnez la queue test\_queue.

Allez dans la section "Get messages".

Saisissez le nombre de messages à récupérer (par exemple, 5).

Cliquez sur "Get Message(s)".

Vous devriez voir les messages que vous avez publiés.

**Étape 6 : Tester la tolérance aux pannes**

Arrêtez l'un des nœuds pour simuler une panne :

docker stop rabbitmq2

Retournez à l'interface de gestion et vérifiez l'état du cluster. Les autres nœuds devraient continuer de fonctionner.

Publiez et consommez des messages pour vous assurer que le cluster continue de fonctionner sans le nœud arrêté.

Redémarrez le nœud arrêté :

docker start rabbitmq2

Vérifiez que le nœud a rejoint le cluster à nouveau en utilisant la commande :