Projet RabbitMQ

L'institut de physique nucléaire NGI souhate mettre en place un système de calcul distribué afin d'effectuer des opérations mathématiques complexes. Une consultation est donc menée auprès de fournisseurs potentiels. L'idée est de leur demander d'implémenter un système de calcul distribué simple pour juger de l'efficacité et des compétences de chaque fournisseur.

Dans ce projet, effecuté par groupe de 3 ou 4, on souhaite utiliser RabbitMQ afin d'implémenter la solution.

Pour cela, on pourra utiliser :
- Un serveur RabbitMQ installé en local
- Un serveur RabbitMQ via Docker
- Un serveur RabbitMQ déjà existant

Les langages de programmation à utiliser sont libres (JavaScript, Python, Java, C, C++,...), ainsi que les librairies AMQP utilisées. On pourrait même utiliser plusieurs langages de programmation selon le rôle.

Livrables attendus :
- L'adresse d'un dépôt Git **public** (GitHub, GitLab, Bitbucket,...) ou un fichier compressé (zip, rar, tar.gz,...) contenant le code source du projet
- Un document présentant le projet, les commandes à lancer pour le mettre en place et le tester, les explications sur le code, etc... Bref, mettre en avant l'implication de l'équipe et la qualité de son projet.

Note : Bien vérifier que tout est correct et complet pour mettre en place et lancer la solution (installation des dépendances, commandes,...). Le client final n'a pas envie de passer plusieurs jours à trouver comment lancer le projet 😊.

Barème de notation pour juger de la solution :
- Réponse au besoin initial : 25% - 5 points
- Implémentations de la dernière partie : 25% - 5 points
- Qualité du code : 25% - 5 points
- Documentation : 25% - 5 points

Début du projet

Le projet consiste à simuler un système de calcul distribué. On aura un serveur RabbitMQ qui va recevoir des requêtes de la part de clients, et qui va les distribuer à des workers. Les workers vont effectuer le calcul demandé, et renvoyer le résultat au serveur RabbitMQ.

Une demande de calcul est un message comprenant deux opérandes sous la forme d'un fichier JSON. Exemple :

```
{
  "n1": 5,
  "n2": 3
}
```

Ecrire un client qui va envoyer une requête de calcul toutes les 5 secondes au serveur en générant automatiquement une valeur pour `n1` et `n2`.

Le serveur RabbitMQ va recevoir ce message, et le distribuer à un worker parmi ceux disponibles. Le worker va effectuer la somme des deux opérandes, et renvoyer le résultat au serveur RabbitMQ dans une *queue*. Afin de simuler les calculs complexes prévus, le worker fera un *wait* de 5-15 secondes (aléatoire) avant de renvoyer le résultat.

Le résultat sera sous la forme d'un fichier JSON. Exemple :

```
{
  "n1": 5,
  "n2": 3,
  "op": "add"
  "result": 8
}
```

Ecrire un autre client qui va lire dans la *queue* les résultats des calculs, et les afficher à l'écran.

Amélioration du projet 1

On souhaite mettre en place 4 catégories de workers : *add*, *sub*, *mul*, *div*. Chaque worker ne traitera que les calculs de sa catégorie. On pourra par exemple paramétrer un worker à son démarrage en indiquant en paramètre de la commande l'opération qu'il doit effectuer.

Améliorer le client qui envoie les requêtes de calcul pour qu'il puisse envoyer des requêtes de calcul de type *add*, *sub*, *mul*, *div* de façon aléatoire, peut-être plus souvent que toutes les 5 secondes.

Amélioration du projet 2

On souhaite ajouter une opération de type *all* qui enverrait l'opération aux 4 types de workers.

Améliorer le client qui envoie les requêtes de calcul pour qu'il puisse envoyer des requêtes de calcul de type *add*, *sub*, *mul*, *div* et *all* de façon aléatoire.

Fin du projet

Améliorer le client ou les workers. Quelques idées mais il y en a bien d'autres - Laisser parler son imagination :

- Permettre à un utilisateur d'envoyer une opération de son choix au système
- Ajouter une interface graphique
- Ajouter une interface d'administration
- Faire en sorte que le client affiche le résultat des opérations que lui-même a soumises
- Mettre en place un fichier Docker Compose