TP 8 : Base de données résiliente

Frédéric Fort - Université de Lille
frederic.fort@univ-lille.fr

2025

Introduction

Dans ce TP, vous allez déployer une infrastructure résiliente de base de données en utilisant PostgreSQL. Cette architecture utilisera 1 *serveur primaire*, accessible en lecture/écriture, ainsi que plusieurs *réplicateurs*, accessibles en lecture uniquement. Cette infrastructure offre un équilibre raisonnable entre cohérence et résilience. Les modifications ne pouvant venir que d'un seul serveur, la base de donnée ne peut à aucun moment être dans un état incohérent. Cependant, les réplicateurs peuvent répondre aux requêtes lecture seule qui typiquement sont les plus fréquentes et complexes. Cette architecture fonctionne de plus très bien avec la répartition de charge intégrée de Kubernetes.

Temps prévu pour ce TP : 1 séance

Quelques liens utiles pour ce TP :

— La documentation officielle de Kubernetes
— La documentation des StatefulSet de Kubernetes
— La documentation officielle de PostgreSQL
— La documentation de pg_basebackup
— La documentation de pgBackRest

Note pour le rendu

Pour faciliter la préparation et notation du rendu, il vous sera demandé d'utiliser un format unifié pour le rendu de ce TP. Créez un dossier `tp8` situé à la racine du dépôt de rendu. Les fichiers pour ce TP doivent tous se situer dans ce dossier.

Si vous avez rencontré des problèmes durant le TP, vous pouvez indiquer vos problèmes et présenter votre diagnostique ainsi que les opérations faites pour y remédier Un processus de réflexion correct et approfondi sera noté favorablement.

1 Configuration de base

Comme indiqué ci-dessus, l'architecture désirée devra comporter plusieurs serveurs, organisés en pods. Un pod aura la fonction de *primaire*, capable de répondre à toutes les requêtes. Afin de répartir la charge et afin de prévoir d'éventuelles défaillances, plusieurs *réplicateurs* disponibles en lecture seule utiliseront la fonctionnalité WAL Streaming Replication afin d'avoir toujours une version à jour. Il sera nécessaire d'exposer deux services, l'un permettant les requêtes en lecture seule, l'autre permettant les requêtes en lecture/écriture.

Stockez tous les fichiers nécessaires pour cette configuration dans un dossier `Q1`. Ajoutez dans le README les commandes à exécuter.

Votre configuration doit être résiliente à la défaillance des réplicateurs. Que ça soit en supprimant les pods ou en supprimant la VM, des remplaçant doivent être déployés et ceux-ci doivent être automatiquement configurés.

2 Restauration de primaire

Dans la configuration précédente, une défaillance du primaire n'est pas prise en compte. Mettez en place une solution permettant d'obtenir un nouveau primaire en cas de défaillance de celui-ci. Des possibilités sont :

— Promotion d'un réplicateur en primaire ;

— Restauration du primaire à partir d'un backup.

Stockez tous les fichiers nécessaires pour cette configuration dans un dossier `Q2`. Ajoutez dans le README les commandes à exécuter.

3 Bonus : Restauration de primaire et réplicateur

Dans les cas précédents, il était supposé qu'il existait toujours au moins un pod lors de la restauration de pods. Dans certains cas catastrophiques, il se peut qu'aucun pod ne soit disponible. Dans ce genre de situation, il est nécessaire d'utiliser un backup externe afin de bootstrap le cluster.

Mettez en place une procédure d'archivage utilisant pgBackRest. Stockez tous les fichiers nécessaires pour cette configuration dans un dossier `Q3`. Ajoutez dans le README les commandes à exécuter.