Haute disponibilité d'un service Web dynamique

Activité 4 – Intégration de la solution dans le cluster

Problématique : comment gérer la défaillance du serveur maître ?

Si le serveur maître n'est plus disponible, *Corosync* bascule ses services sur le serveur esclave sur lequel MySQL est démarré et qui acceptera donc que l'application y accède en écriture... Jusque là, il n'y a pas de souci.

Le problème est que nous sommes, en natif, dans une configuration Maître-Esclave. Le master sera de nouveau fonctionnel à un moment donné et nous lui avons donné une préférence : les services vont basculer automatiquement, or ce dernier n'est pas configuré pour récupérer les données éventuellement écrites sur le SLAVE.
Deux solutions sont possibles :

Première solution non automatisée avec le service MySQL et non géré par Pacemaker

Cette solution peut s'avérer très lourde à mettre en place, d'autant plus qu'il faut considérer deux cas :

- **cas n°1** : on s'est aperçu immédiatement que le serveur esclave a pris le relais ==> on le transforme en maître en repérant le log binaire et la position du jeu de commandes dans le log de manière à ce que, quand le maître soit de nouveau opérationnel, on le transforme momentanément en esclave pour qu'il puisse récupérer les données utiles ;
- **cas n°2** : on ne s'est pas aperçu immédiatement que le serveur esclave a pris le relais et des données ont été écrites dans la base de données avant que l'on récupère le log binaire et la position du jeu de commandes dans le log. Il faut récupérer l'intégralité de la base de données contenant les données actuelles et la restaurer sur le maître avant de relancer le service.

Dans les deux cas, en dehors de la lourdeur de la solution, *l'interruption de services que celle-ci engendre peut ne pas être acceptable.*

Deuxième solution : configurer la réplication MySQL en multi maître

Le serveur de secours doit aussi être configuré comme étant maître au niveau de MySQL. Et chacun sera aussi l'esclave de l'autre.
Cela revient en fait à réaliser **deux réplications maître-esclave**.
Ainsi chaque serveur pourra, à chaque instant, prendre la relève de l'autre avec des données actuelles.

C'est cette solution que nous allons configurer car non seulement elle est simple à mettre en œuvre, n'engendre pas d'interruption de service mais c'est la seule possible (avec MySQL) pour une répartition de charge (que nous allons effectuer dans un prochain Coté Labo).

L'objectif opérationnel de cette quatrième activité est donc d'intégrer la solution dans le cluster de manière à ce que ce dernier gère automatiquement une éventuelle défaillance.

Vous disposez de la documentation suivante :

- **document 1** : configuration de la réplication MySQL en multi maître
- **document 2** : intégration du service MySQL au cluster.

Travail à faire

Q1. Transformez l'architecture maître-esclave en architecture multi maître afin de ne pas perdre de données lorsque le serveur maître sera de nouveau opérationnel après une défaillance. Écrivez ci-dessous la procédure.

Q2. Testez la solution.

Q3. Intégrez cette solution au cluster.

Q4. Complétez le tableau ci-dessous permettant de vérifier l'opérationnalité de la solution.

Actions à effectuer	Résultats attendus	Résultats obtenus	Statut (OK ou non OK)
Les ressources sont sur le nœud serv1 : Saisie d'une fiche de frais à partir de l'application			
On stoppe le nœud serv1.			
Les ressources sont sur le nœud serv2: Saisie d'une autre fiche de frais à partir de l'application			
On redémarre le nœud serv1			

Document 1 : Configuration de la réplication MySQL en multi maître

Les étapes sont les suivantes :

- ajouter dans chaque fichier de configuration ce qui lui manque (pour le nœud actuellement maître tout ce qui concerne la configuration d'un esclave et pour le nœud actuellement esclave tout ce qui concerne la configuration d'un maître) ;
- créer sur le « nouveau maître », l'utilisateur MySQL avec les droits adéquats ;
- redémarrer MySQL ;
- repérer sur chaque nœud le log binaire et la position du jeu de commandes dans le log ;
- saisir sur chaque nœud les commandes *change master to...* et *start slave* (dans notre cas, ces commandes ne sont à saisir que sur le nœud actuellement maître pour le transformer en esclave puisque l'autre nœud est déjà configuré en esclave) ;

il ne faut surtout pas oublier d'ajouter l'option l*og-slave-updates* sur chaque serveur. Le serveur « serv2 » va se baser sur le **log-bin** du serveur « serv1 » pour répliquer les données. Par défaut, le serveur « serv2 » ne « log » pas dans ses **log-bin** les données de réplication. Or, ceci est indispensable de manière à ce « serv2 » sache qu'un événement qu'il a déjà exécuté lui est revenu pour qu'il ne l'exécute pas deux fois (et vice-versa) ; c'est ce que permet cette option.

Pour tester la configuration, sur les 2 serveurs : *show slave status \G;*
Slave_IO_Running et Slave_SQL_Running doivent être à Yes.

Remarque : **dans le cas d'une configuration pour réaliser une répartition de charges,** deux autres lignes doivent être ajoutées dans chaque fichier de configuration pour éviter les conflits d'insertion menant la réplication à l'échec car lors de l'écriture simultanée sur les deux maîtres, il pourrait y avoir des collisions au niveau des clés auto incrémentées.
L'astuce est de configurer chaque serveur pour qu'il génère des clefs primaires qui n'entrent pas en conflit. Et cela est possible grâce aux variables *auto-increment-increment* et *auto-increment-offset*.

- ***auto_increment_increment*** contrôle l'incrémentation entre les valeurs successives des clés (le pas d'incrémentation).
- ***auto_increment_offset*** détermine le point de départ des valeurs des colonnes de type AUTO_INCREMENT.

Ainsi, pour une configuration à N maîtres, mettre :

- auto_increment_increment = N (sur chaque maître) ;
- auto_increment_offset = 1,2, ... , N (chaque maître aura une valeur unique)

On aura l'occasion d'y revenir.

Document 2 : Intégration de la réplication dans le cluster

Nous intégrerons la deuxième solution.

On utilise ici la ressource d'heartbeat ocf:heartbeat:mysql et la possibilité de configurer une ressource en **mode actif/actif.**

Une ressource configurée en actif/actif est démarrée sur chaque nœud. Pour passer une ressource en mode actif/actif il faut utiliser la fonction “clone”. Il y a 3 types de clones :

- Le clone « **anonymous » (clone par défaut)** : les ressources sont configurées de manière identique sur chaque nœud. Si un nœud est indisponible, il n’y a pas de bascule de cette ressource puisqu’elle est déjà fonctionnelle et identique sur l’autre nœud. **Ce type de clone permet d'intégrer notre solution dans le Cluster.**
- Le clone « **globally-unique=true »** : contrairement au cas précédent, les ressources ne sont pas identiques d’un nœud à l’autre. Elles peuvent par exemple, avoir des données différentes.
- Le “**multi-state**” : c’est une ressource qui peut être dans 2 états différents : “master” ou “slave”. Ce type de clone est celui que nous utiliserons dans le prochain Coté Labo (http://www.reseaucerta.org/hd-serveur-ftp).

Le principe est de **créer une ressource** de manière classique (tous les paramètres par défaut de la ressource *ocf:Corosync:mysql* conviennent **à l'exception du chemin du socket qui doit être positionné à "/var/run/mysqld/mysqld.sock"**) puis de **la cloner** avec la commande :

```
crm configure clone <nom_clone> <id_resource>
```

Par défaut, il s'agit d'un clone de type « anonymous », aucune option supplémentaire n'est donc nécessaire.

Après configuration, la commande crm_mon donne :

```
Last updated: Fri Jul 22 17:31:29 2016          Last change: Fri Jul 22 17:17:24 2016 by root via cibadmin on serv1
Stack: corosync
Current DC: serv2 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 4 resources configured

Online: [ serv2 serv1 ]

 Resource Group: servIntralab
     IPFailover (ocf::heartbeat:IPaddr2):               Started serv1
     serviceWeb       (ocf::heartbeat:apache):        Started serv1
 Clone Set: cServiceMySQL [serviceMySQL]
     Started: [ serv2 serv1 ]
```

Pacemaker ne désactivera le service MySQL sur aucun des deux nœuds.