B2-Act2-TP1 – La haute disponibilité (8h)

La haute disponibilité

La « haute disponibilité » (en anglais « high availability ») regroupe de nombreuses techniques et processus permettant de garantir un certain pourcentage de disponibilité d'un service.
Par exemple, un taux de 99 % de disponibilité assure une disponibilité d'environ 361 jours sur 364 alors qu'un taux de 99,5 % assure une disponibilité de plus de 363 jours sur 365.

La réalité économique fait que les organisations tendent de plus en plus vers des taux encore plus grands comme 99,9 % ou 99,99 % notamment sur certains services critiques. En effet, les conséquences d'une interruption de service sont innombrables et peuvent coûter très cher à tous points de vue.

Par exemple, sur le site http://www.zdnet.fr, on pouvait lire qu'une interruption de service de 40mn le 19 août 2013 aurait fait perdre à Amazon près de 5 millions de dollars.
(http://www.zdnet.fr/actualites/comme-google-amazon-a-subi-une-panne-informatique-39793254.htm)

La haute disponibilité, comment ça marche ?

Pour améliorer la haute disponibilité, il existe de nombreuses configurations possibles.

Dans une configuration très simple que nous allons découvrir, la haute disponibilité nécessite la présence d'un serveur secondaire, fonctionnant sous le même système d'exploitation et fournissant un accès aux services que l'on souhaite rendre « hautement » disponibles.

Ce second serveur configuré à l'identique, en règle générale services arrêtés, surveillera le premier en permanence. En cas de panne du serveur primaire, il la détectera et prendra la relève, devenant alors le nouveau serveur actif.

Un des algorithmes utilisés pour ce genre d'opérations est basé sur la tachycardie. Il est appelé « heartbeat », ou battements de cœur. Le serveur actif émet régulièrement des informations sur le réseau pour dire qu'il est vivant, pendant que l'autre écoute passivement.

Si plus aucune information n'arrive au serveur en écoute, celui-ci s'alarme.

Il prend alors le rôle de serveur actif (les services basculent sur ce serveur) et se met à son tour à émettre des battements de cœur.
Si l'autre serveur est restauré, il jouera, au moins dans un premier temps, le rôle de serveur en écoute.

N'importe quelle machine pourra ainsi tomber en panne sans que l'ensemble ne soit pénalisé.

Ces techniques seront mises en œuvre via deux outils à installer et configurer :

- **Corosync** qui permet de détecter la défaillance d'un poste grâce à un système de communication et de gérer le cluster en lui-même (on aurait pu tout aussi bien utiliser ici un autre outil comme « Heartbeat »).
- **Pacemaker** qui est un gestionnaire de ressources. Il est chargé de créer, démarrer, arrêter et superviser les ressources du cluster c'est à dire les services gérés en cluster et donc inclus dans la continuité de services.

Contexte

Le laboratoire pharmaceutique Galaxy-Swiss Bourdin (GSB) met à disposition des visiteurs médicaux une application Web sécurisée de gestion des frais de remboursement. Cette application nécessite :

- un serveur Web sécurisé (HTTPS, SSL/TLS) ;
- l'accès à une base de données relationnelle, éventuellement administrable par interface Web.

L'authentification des visiteurs pour l'accès au contenu est gérée par l'application à travers la base de données.

L'entreprise a choisi d'héberger en interne les serveurs exécutant l'application sur un serveur Linux.
Par mesure de simplification, les deux serveurs sont sur la même machine physique. À noter aussi que ce Côté Labo peut être réalisé même si le protocole HTTPS n'a pas été mis en œuvre.

GSB désire disposer d'un serveur de secours prêt à prendre le relais si le serveur principal venait à ne plus être opérationnel.

Les activités proposées ont pour objectif de construire, étape par étape, une solution totalement opérationnelle.

La première activité consiste à préparer les deux serveurs à l'identique en installant et en procédant à une première configuration des outils de haute disponibilité (fichier *haServiceWeb_activite1*).

La deuxième activité a pour objectif d'intégrer le service Web au **cluster** composé des 2 serveurs (fichier *haServiceWeb_activite2*).

La troisième activité consiste à configurer la réplication entre les deux serveurs selon une architecture de type maître-esclave. On ne se préoccupe pas ici des problématiques de Corosync et de Pacemaker (fichier *haServiceWeb_activite3*).

La quatrième activité a pour objectif d'obtenir une solution complètement opérationnelle en intégrant le service de base de données dans le cluster (fichier *haServiceWeb_activite4*).

Le schéma de l'infrastructure est donné en annexe, page suivante.

Source : Certa

Schéma de l'infrastructure – 1ère partie (activités 1 et 2) : cluster de serveurs web

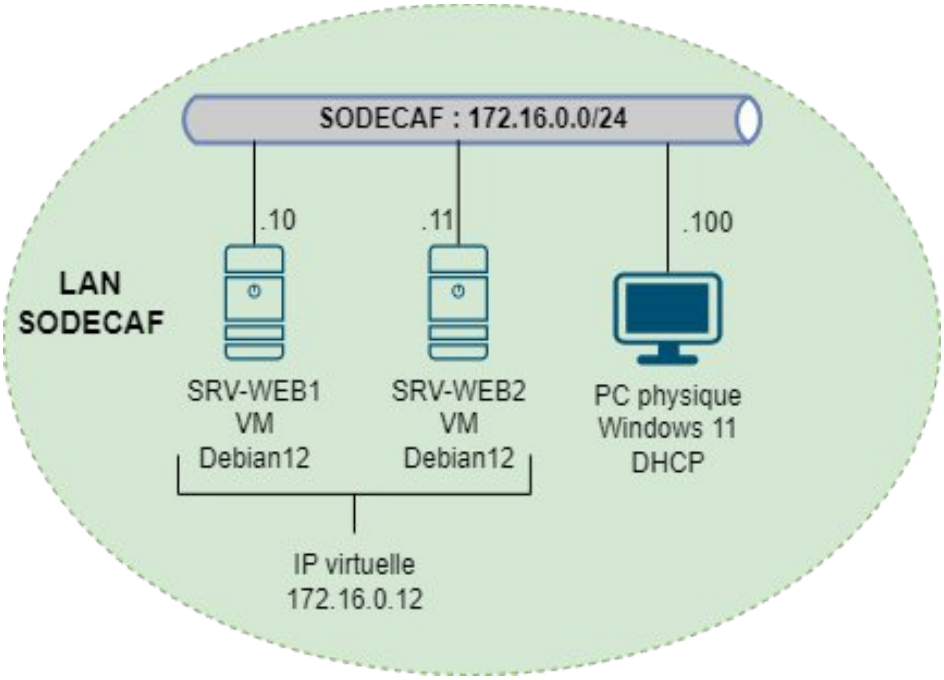

Schéma de l'infrastructure – 2ème partie (activités 3 et 4) : cluster de serveurs web + BDD MySQL