E5 - BTS Services Informatiques aux Organisations (SIO)
parcours SISR

C4 – VÉRIFICATION DES CONDITIONS DE LA CONTINUITÉ D'UN SERVICE INFORMATIQUE

Présenté par:

MECHRAFI-LACROIX Steven

Année académique : 2024/2026

Documentation réalisée dans le cadre du BTS SIO par Steven MECHRAFI-LACROIX, année académique 2024/2026.

Objectifs du document

Ce document a pour objectif de présenter la **vérification des conditions de la continuité d'un service informatique** à travers la mise en place d'un **load balancer HAProxy**. La répartition de charge permet d'assurer la disponibilité et la continuité des services même en cas de défaillance d'un serveur.

Il aborde notamment :

1. **Le principe de la continuité de service** : disponibilité, tolérance aux pannes et répartition de charge.
2. **La configuration d'un load balancer HAProxy** : répartition des requêtes entre plusieurs serveurs pour garantir la continuité des services.

Principe de la continuité de service

La **continuité d'un service informatique** désigne sa capacité à rester disponible et fonctionnel malgré les incidents (panne matérielle, surcharge, maintenance). Elle repose sur plusieurs leviers :

- **La haute disponibilité** : éviter qu'une panne unique n'interrompe le service (suppression du point unique de défaillance, ou *SPOF*).
- **La répartition de charge** (*load balancing*) : distribuer les requêtes entre plusieurs serveurs pour éviter la surcharge et améliorer les performances.
- **La redondance** : disposer de plusieurs serveurs capables d'assurer le même service, de sorte que si l'un tombe, les autres prennent le relais.

Le **load balancer** est l'élément clé de ce dispositif : il se place en frontal des serveurs et oriente les requêtes des clients vers les serveurs disponibles.

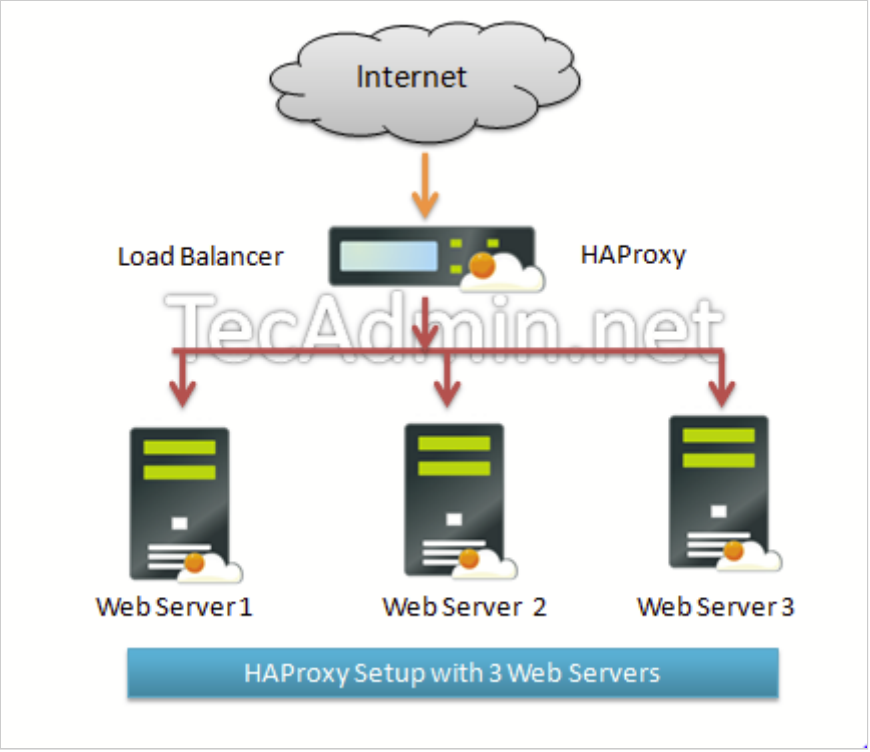
Configuration d'un load balancer HAProxy

HAProxy (*High Availability Proxy*) est une solution open source de répartition de charge et de proxy, très utilisée pour assurer la haute disponibilité des services web et applicatifs. Elle est réputée pour sa fiabilité et ses performances.

Rôle de HAProxy

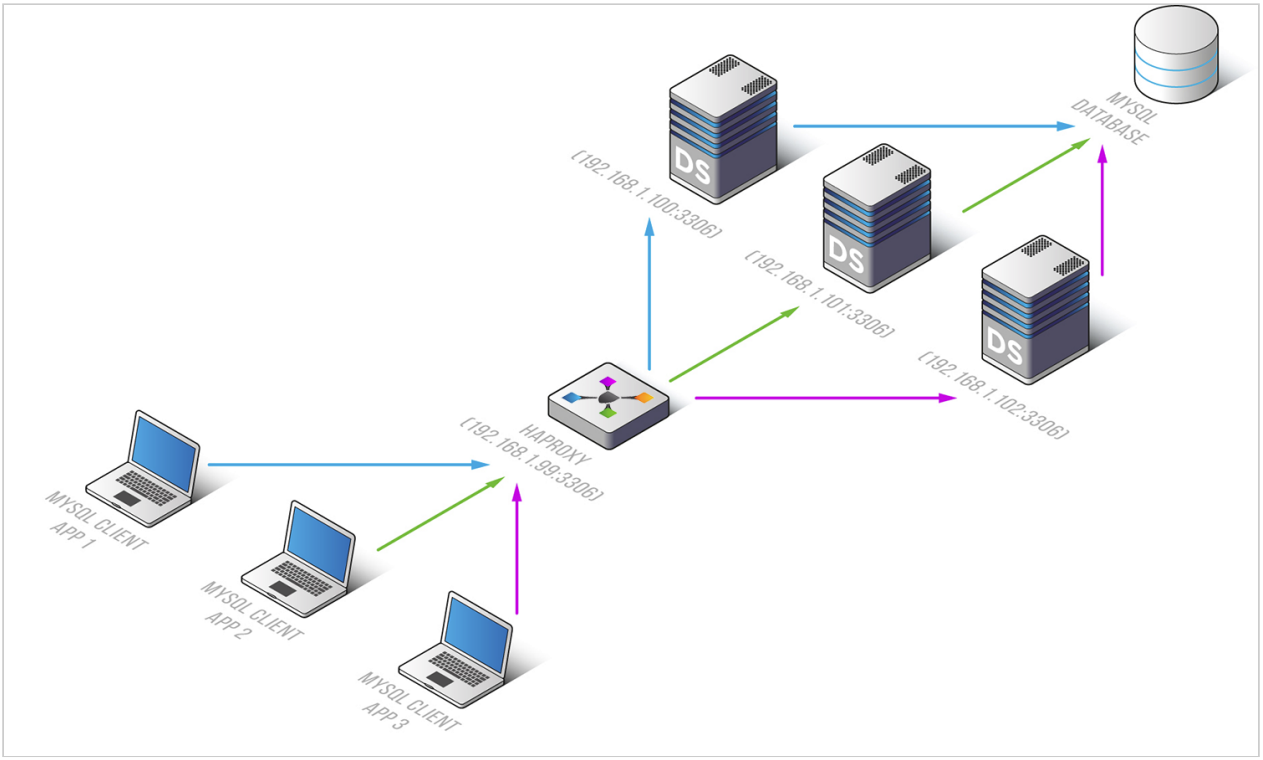
HAProxy reçoit l'ensemble des requêtes entrantes et les répartit entre plusieurs serveurs en arrière-plan (*backend*). Si l'un des serveurs devient indisponible, HAProxy le détecte et cesse de lui envoyer du trafic, garantissant ainsi la continuité du service.

Le schéma ci-dessous illustre un déploiement classique : les requêtes provenant d'Internet passent par le load balancer HAProxy, qui les répartit entre trois serveurs web.



Mise en place de HAProxy en load balancer devant trois serveurs web.

Le même principe s'applique à d'autres services, comme une base de données. Dans l'exemple suivant, plusieurs clients MySQL se connectent à HAProxy, qui répartit les connexions entre plusieurs serveurs avant l'accès à la base de données.



Répartition des connexions de clients MySQL par HAProxy vers plusieurs serveurs (192.168.1.100 à 192.168.1.102) avant accès à la base de données.

Principaux éléments de configuration

La configuration de HAProxy s'effectue dans le fichier `haproxy.cfg`, organisé en plusieurs sections :

- **global** : paramètres généraux (utilisateur, journalisation, limites).
- **defaults** : paramètres par défaut appliqués aux autres sections (timeouts, mode de fonctionnement).
- **frontend** : définit le point d'entrée (adresse IP et port d'écoute) sur lequel les clients se connectent.
- **backend** : liste les serveurs réels (*backend servers*) entre lesquels la charge est répartie.

Algorithmes de répartition

HAProxy propose plusieurs algorithmes pour distribuer les requêtes :

Algorithme	Principe
roundrobin	Distribution à tour de rôle entre les serveurs
leastconn	Envoi vers le serveur ayant le moins de connexions actives

Algorithme	Principe
source	Le même client est toujours dirigé vers le même serveur

Vérification de la disponibilité (health checks)

HAProxy effectue des **contrôles de santé** (*health checks*) réguliers sur chaque serveur du backend. Un serveur qui ne répond plus est automatiquement retiré de la répartition, puis réintégré dès qu'il redevient disponible. C'est ce mécanisme qui assure concrètement la **continuité du service** : la panne d'un serveur reste transparente pour les utilisateurs.

Vérification des conditions de continuité

Une fois HAProxy en place, il convient de **vérifier** que la continuité de service est bien assurée :

- **Test de répartition** : envoyer plusieurs requêtes et vérifier qu'elles sont bien distribuées entre les serveurs.
- **Test de bascule (failover)** : arrêter volontairement un serveur backend et vérifier que le service reste accessible via les serveurs restants.
- **Supervision** : consulter les statistiques de HAProxy (page de stats) pour suivre l'état des serveurs et le trafic.
- **Journalisation** : analyser les logs pour détecter d'éventuelles anomalies.

Synthèse

La mise en place d'un load balancer **HAProxy** permet de garantir la **continuité d'un service informatique** en répartissant la charge entre plusieurs serveurs et en retirant automatiquement de la rotation tout serveur défaillant. Associée à la redondance des serveurs, cette architecture supprime les points uniques de défaillance et assure une disponibilité élevée.

La **vérification régulière** (tests de bascule, supervision, journalisation) confirme que les conditions de continuité sont effectivement réunies et maintenues dans le temps.