MYDIGITALSCHOOL ANGERS — BTS SIO SISR

Mise en œuvre d'un Reverse Proxyavec Load Balancing (HAProxy)

Rapport de tests de failover

Rédacteur	Paul Barrault-Gautier
Établissement	MyDigitalSchool Angers
Formation	BTS SIO — option SISR (Session 2026)
Cadre	Travaux pratiques / Projet de formation
Date	Avril 2026
Version	1.0

1. Objectif et périmètre des tests

Ce rapport documente les tests de basculement (failover) et de load balancing réalisés sur l'infrastructure HAProxy mise en place chez Loste Tradi-France. L'objectif est de valider :

- La détection automatique de la panne d'un serveur back-end
- Le basculement du trafic vers les serveurs disponibles sans interruption perceptible
- La reprise automatique d'un serveur back-end après rétablissement
- La répartition effective de la charge entre les serveurs actifs
- La continuité de service lors d'une mise à jour sans coupure (rolling restart)

1.1 Environnement de test

Paramètre	Valeur
Date des tests	Avril 2026
Testeur	Paul Barrault-Gautier
Version HAProxy	2.8.x LTS
Outil de génération de charge	Apache Benchmark (ab) + curl en boucle
Serveurs back-end	web-back-01 / 02 / 03 (Debian 12, Nginx)
Health check configuré	GET /health — intervalle 2s — fall 3 — rise 2

2. Plan de tests

ID	Scénario	Résultat attendu	Statut
T-01	Arrêt brutal de web-back-01 (kill -9)	Trafic basculé sur 02 et 03, 0 erreur client	**PASS**
T-02	Arrêt du service Nginx sur web-back-02	Health check détecte la panne en < 10s	**PASS**
T-03	Panne réseau simulée (iptables DROP) sur web-back-03	Timeout détecté, basculement automatique	**PASS**
T-04	Panne simultanée de 2 back-ends sur 3	Service maintenu sur le back-end restant	**PASS**
T-05	Panne des 3 back-ends simultanément	Retour 503, reprise auto au retour du service	**PASS**
T-06	Reprise de web-back-01 après arrêt (T-01)	Réintégré automatiquement après 2 checks OK	**PASS**
T-07	Vérification de la répartition roundrobin (1000 req.)	~333 req/back-end (±5 %)	**PASS**
T-08	Reload HAProxy (sudo systemctl reload haproxy)	Zéro coupure, connexions en cours maintenues	**PASS**

3. Résultats détaillés

3.1 Test T-01 — Arrêt brutal d'un back-end

Procédure :

```
# Simuler un crash de web-back-01
ssh web-back-01 'sudo kill -9 $(pidof nginx)'
```

```
# Maintenir une charge continue depuis le client de test
while true; do curl -s -o /dev/null -w "%{http_code}\n" http://192.168.10.10/; sleep
0.2; done
```

Résultats :

- Délai de détection de la panne : 6 secondes (3 health checks échoués × 2s)
- Nombre de requêtes en erreur (code 5xx) reçues par le client : 0
- Le trafic a été redistribué automatiquement sur web-back-02 et web-back-03
- Les logs HAProxy indiquent le passage de web-back-01 en état DOWN

```
# Extrait de log HAProxy
[WARNING] Server be_app1/web-back-01 is DOWN, reason: Layer4 connection problem
          info: "Connection refused", check duration: 1ms. 2 active and 0 backup
servers left.
```

3.2 Test T-02 — Arrêt du service applicatif

Procédure :

```
ssh web-back-02 'sudo systemctl stop nginx'
```

Résultats :

- Health check HTTP GET /health : retour HTTP 502 (Nginx stoppé)
- Délai avant marquage DOWN : 3 checks échoués = 6 secondes
- Aucune interruption de service côté client
- Service repris automatiquement après redémarrage de Nginx sur web-back-02

```
# Reprise automatique
[WARNING] Server be_app1/web-back-02 is UP, reason: Layer7 check passed
          code: 200, check duration: 3ms. 3 active and 0 backup servers left.
```

3.3 Test T-07 — Vérification de la répartition de charge

Procédure :

```
# 1000 requêtes avec 10 threads simultanés
ab -n 1000 -c 10 http://192.168.10.10/
```

Résultats de répartition (relevé dans les access logs de chaque back-end) :

Serveur back-end	Requêtes reçues	% du total	Écart vs théorie
web-back-01	338	33,8 %	+0,8 % ✓
web-back-02	331	33,1 %	+0,1 % ✓
web-back-03	331	33,1 %	+0,1 % ✓
TOTAL	**1 000**	**100 %**	—

La répartition observée est conforme à la configuration roundrobin avec poids égaux (tolérance ≤ 5 %).

3.4 Test T-08 — Reload sans coupure

Procédure :

```
# Générer une charge continue
ab -n 5000 -c 20 http://192.168.10.10/ &

# Pendant la charge, recharger HAProxy
sudo systemctl reload haproxy
```

Résultats :

- Aucune requête échouée pendant le reload (0 erreur sur 5000 requêtes)
- HAProxy utilise un mécanisme de « zero-downtime reload » : le nouveau processus reprend les sockets avant que l'ancien ne se termine
- Temps de reload mesuré : < 200 ms

4. Métriques de performance

Métrique	Valeur mesurée	Seuil acceptable
Délai de détection de panne (health check)	6 s (3×2s)	< 15 s
Temps de basculement effectif (client)	< 1 s	< 5 s
Délai de réintégration d'un back-end (rise 2)	4 s (2×2s)	< 30 s
Taux d'erreur client lors des tests de panne	0 %	< 0,1 %
Débit max observé (ab, 10 threads)	1 250 req/s	> 500 req/s
Temps de reload sans coupure	< 200 ms	< 1 s

5. Conclusion et bilan

L'ensemble des 8 scénarios de test ont été validés avec succès. L'infrastructure HAProxy mise en place répond aux exigences de haute disponibilité et de répartition de charge définies pour Loste Tradi-France.

Points clés validés :

- Le mécanisme de health check détecte toute panne en moins de 10 secondes
- Le failover est transparent pour les clients (0 erreur 5xx lors des pannes simulées)
- La répartition roundrobin est homogène (écart < 1 % entre les back-ends)
- Le reload de configuration se fait sans interruption de service
- La reprise automatique des serveurs après rétablissement est fonctionnelle

5.1 Recommandations post-déploiement

- Mettre en place un monitoring HAProxy via Prometheus / haproxy_exporter + Grafana
- Configurer des alertes sur le passage en DOWN de tout serveur back-end
- Documenter et automatiser la procédure de mise à jour des certificats SSL
- Envisager un deuxième HAProxy en mode actif/passif (Keepalived + VIP) pour éliminer le SPOF
- Programmer des tests de failover trimestriels pour s'assurer de la robustesse continue