SUPFile

Plateforme de stockage cloud multi-datacenter

Preuve de Concept (POC) infrastructure

Ayman EL KARROUSSI (Paris)
Franck ANDY MEVENGUE-ENGONGOMO (New York)
Nadia LOUKDACHE(Toronto)

2025/2026

Document technique – Rapport de POC infrastructure

Version 3.0 du 23 mai 2026

Tailnet : `tail69cc44.ts.net`

Document préparé dans le cadre du sujet SUPFile Archi

Résumé exécutif

SUPFile est une plateforme de stockage cloud multi-datacenter dont l'infrastructure est conçue pour offrir une haute disponibilité Active/Active entre Paris et New York, une couche de stockage résiliente (iSCSI SAN, RAID-1, GlusterFS distribué-répliqué), une base de données Galera multi-master synchrone, un plan de bascule complet vers le datacenter Toronto, et une politique de sécurité multicouche (UFW, Fail2ban, Suricata, ban cross-DC via Galera).

Le POC déploie **17 machines virtuelles** (8 Paris + 8 New York + 1 Toronto consolidée), interconnectées par tunnels chiffrés ChaCha20-Poly1305. La validation finale du 23 mai 2026 confirme **28 tests passés sur 30**, les 2 écarts étant des limitations matérielles documentées du contexte POC (RAID-1 dégradé sur disques Vagrant trop petits).

L'estimation budgétaire actualisée mai 2026 est de **968 €/mois** en modèle cloud OVHcloud (Active/Active 2 DC + Toronto cold standby), ou **126 500 €** de CapEx initial pour une infrastructure physique propriétaire amortie sur 5 ans (~2 108 €/mois).

Table des matières

Table des figures

Liste des tableaux

CHAPITRE 1

Introduction

1.1 Contexte du projet SUPFile

SUPFile est un service en ligne de stockage de fichiers, vendant 30 Go par utilisateur. La plateforme doit autoriser le dépôt, la suppression et la consultation de fichiers (avec visualisation directe pour vidéos, images et fichiers texte), exposer une API REST en JSON et fonctionner sur Internet à grande échelle. L'offre se décompose en deux infrastructures distinctes mais couplées : une infrastructure web servant l'interface utilisateur, et une infrastructure de stockage hébergeant physiquement les fichiers et l'évoluant linéairement avec le nombre d'utilisateurs.

Le commanditaire impose des contraintes fortes sur la résilience opérationnelle : haute disponibilité Active/Active sur deux datacenters (Paris et New York), haute disponibilité Active/Passive intra-datacenter, SAN iSCSI avec basculement automatique, disques à chaud (hot-swap), réplication permanente des données, sauvegarde sur un troisième datacenter (Toronto, bandes), chiffrement intégral des communications inter-datacenter, politique de bannissement IP, supervision et SIEM, plan de continuité et de reprise après sinistre (PCA/PRA), et schéma de base de données documenté.

1.2 Périmètre du POC

Le POC livré couvre l'intégralité des couches d'infrastructure demandées par le cahier des charges. Il ne contient *pas* le développement applicatif final (sous-traité à une équipe distincte), mais déploie une application FastAPI + React de référence permettant de démontrer le bon fonctionnement bout en bout de la chaîne : client navigateur → DNS → HAProxy → Nginx → FastAPI → ProxySQL → Galera → GlusterFS.

Tous les composants critiques sont testés en conditions de panne : arrêt du serveur web actif, arrêt du nœud SAN maître, arrêt complet d'un datacenter, perte d'un nœud Galera, écriture cross-DC, bannissement IP propagé entre datacenters.

1.3 Équipe et répartition

Table 1.1 – Répartition des responsabilités

Membre	Site	Responsabilités
Ayman El Karroussi	Paris (8 VMs)	Architecture globale, déploiement Paris, intégration Galera, GlusterFS, monitoring, rédaction documentation
Franck Andy	New York (8 VMs)	Déploiement New York, application backend, intégration cross-DC
Nadia	Toronto (1 VM)	GSLB HAProxy (front `supfile.poc`), backup, cold standby, Galera nœud 3, hébergement Prometheus/-Grafana/dnsmasq

1.4 Méthodologie et conventions

L'infrastructure est entièrement reproductible via Vagrant + VirtualBox sur chaque poste. L'interconnexion inter-datacenter passe par un overlay Tailscale (WireGuard sous-jacent) qui contourne les contraintes de NAT domestique des trois sites. La configuration de production sur tunnels WireGuard directs est documentée en référence dans `configs/wireguard/`.

Chaque test de validation produit un fichier d'évidence horodaté dans `docs/evidence/` (38 fichiers numérotés). Toutes les commandes exécutées sont reproductibles depuis le poste opérateur via `vagrant ssh <nom-vm> -c "<commande>"` ou via SSH Tailscale sur les nœuds distants.

Le document utilise les conventions de notation suivantes :

- `Police monospace` : noms de fichiers, commandes, paramètres techniques.
- **Gras** : éléments critiques ou décisions architecturales.
- *Italique* : termes techniques, acronymes lors de leur première occurrence.
- RPO (Recovery Point Objective) : perte de données maximale acceptable.
- RTO (Recovery Time Objective) : durée maximale d'indisponibilité acceptable.

CHAPITRE 2

Architecture globale

2.1 Vue d'ensemble multi-datacenter

L'architecture SUPFile repose sur trois datacenters géographiquement distincts. Paris (Europe) et New York (Amérique du Nord) hébergent chacun la totalité des couches applicatives (load balancer, deux serveurs web, trois nœuds de stockage, une base de données, un ProxySQL) et opèrent en mode Active/Active : les deux sites servent simultanément la totalité du trafic utilisateur via un GSLB HAProxy hébergé sur Toronto, et bénéficient d'une bascule automatique en cas de panne d'un site. Toronto (Amérique du Nord, Beauharnois) joue à la fois le rôle de point d'entrée GSLB (front `supfile.poc`) et de troisième datacenter destiné aux sauvegardes quotidiennes et au cold standby pour les scénarios d'urgence ultime (panne simultanée Paris + NY).

Le choix d'une stratégie Active/Active plutôt qu'Active/Passive entre Paris et New York répond à deux objectifs concomitants. D'une part, l'efficacité économique : dans un schéma Active/Passive, le site secondaire reste inactif en fonctionnement normal, ce qui revient à payer 50 % de capacité inutilisée. En Active/Active, chaque euro investi sert du trafic en permanence. D'autre part, la latence utilisateur : un utilisateur européen est servi par Paris, un utilisateur nord-américain est servi par New York, ce qui réduit le *round-trip* TCP de plusieurs dizaines de millisecondes par requête. La répartition est assurée par un GSLB HAProxy sur Toronto (`100.126.8.98`) qui résout `supfile.poc` vers une IP unique, health-checke les deux LBs (`100.69.138.20` Paris et `100.84.166.8` New York) toutes les 5 secondes, retire automatiquement le DC en panne du pool (failover <15 s), et complète ce mécanisme par un fallback HAProxy inter-DC au niveau de chaque LB qui prend le relais en moins de 30 secondes si l'un des sites devient injoignable côté backends.

2.2 Diagramme physique

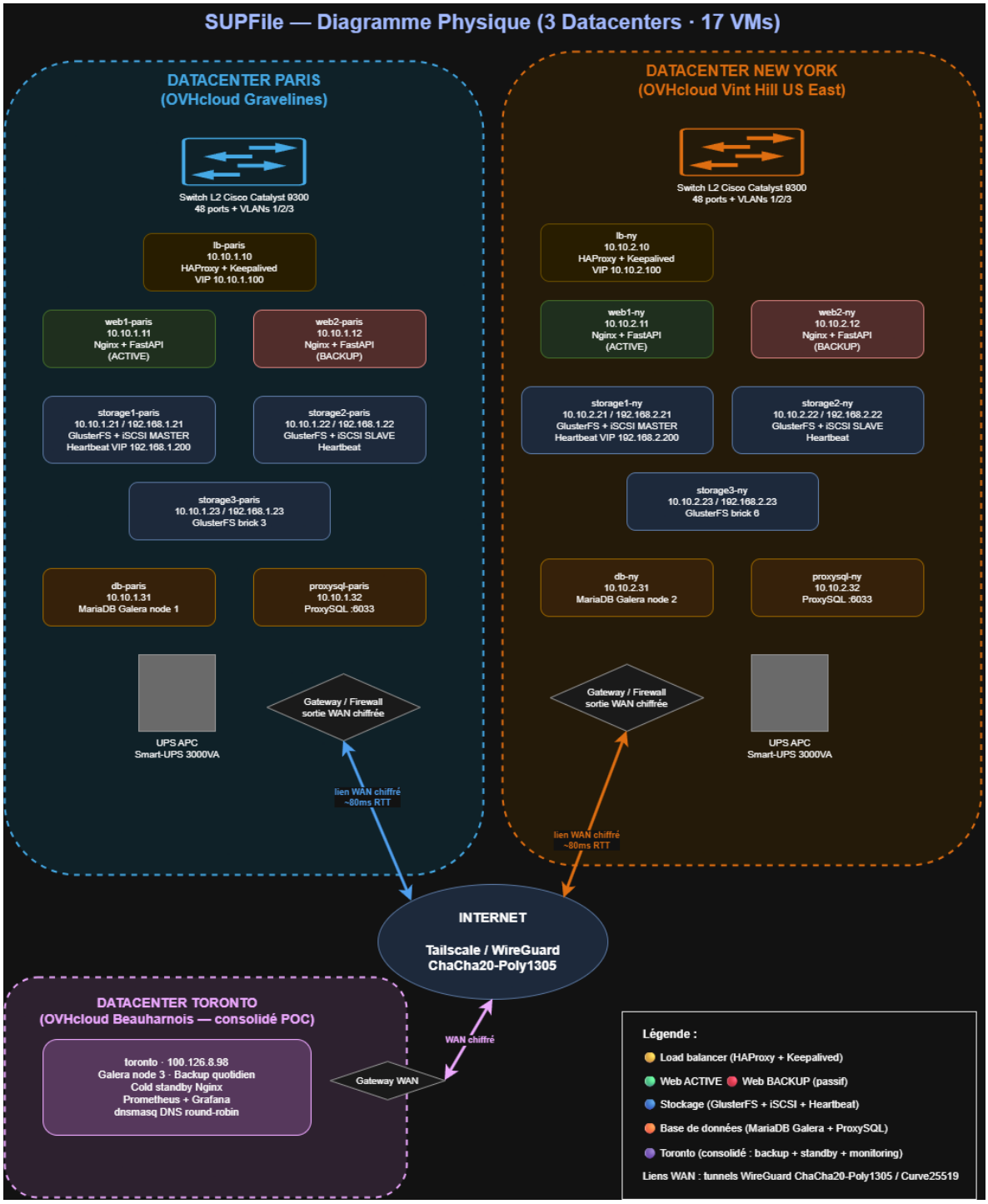

FIGURE 2.1 – Diagramme physique : 3 datacenters, 17 VMs, liens WAN chiffrés ChaCha20-Poly1305

2.3 Diagramme logique

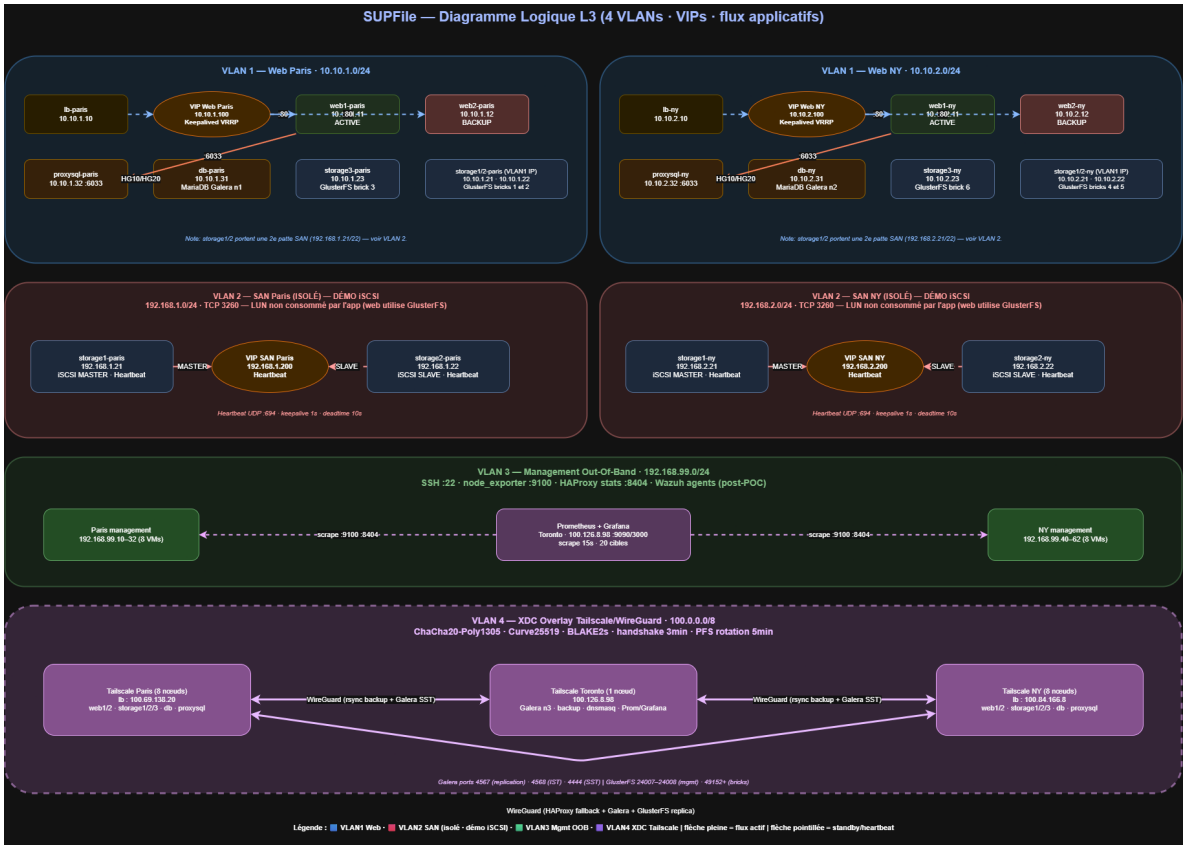

FIGURE 2.2 – Diagramme logique : 4 VLANs isolés (Web, SAN, Management, XDC overlay), VIP Keepalived et Heartbeat

2.4 État opérationnel des VMs

L'inventaire ci-dessous est confirmé par les sorties `vagrant status` capturées sur les deux datacenters Paris et NY : les huit machines de chaque site sont provisionnées et joignables, prêtes à servir le trafic.

```
PS D:\Downloads\Brave\SUPFILE\supfile> $env:SITE = "paris"
PS D:\Downloads\Brave\SUPFILE\supfile>   vagrant status
Current machine states:

lb-paris                  running (virtualbox)
web1-paris                running (virtualbox)
web2-paris                running (virtualbox)
storage1-paris            running (virtualbox)
storage2-paris            running (virtualbox)
storage3-paris            running (virtualbox)
db-paris                  running (virtualbox)
proxysql-paris            running (virtualbox)
```

FIGURE 2.3 – Paris – état `vagrant status` des 8 VMs (lb, web1/2, storage1/2/3, db, proxysql)

```
PS C:\Users\MEVENGUE Franck\Desktop\Ecole Cours SUPINFO\Cours\4 ème Année\4P
er\supfile> vagrant status
Current machine states:

lb-ny                     running (virtualbox)
web1-ny                   running (virtualbox)
web2-ny                   running (virtualbox)
storage1-ny               running (virtualbox)
storage2-ny               running (virtualbox)
storage3-ny               running (virtualbox)
db-ny                     running (virtualbox)
proxysql-ny               running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
```

FIGURE 2.4 – New York – état `vagrant status` des 8 VMs (mêmes rôles, exécution locale par Franck)

2.5 Inventaire des machines virtuelles

TABLE 2.1: Inventaire des 17 VMs et de leurs rôles

VM	Site	VLAN Web	Tailscale	Rôle principal
lb-paris	Paris	10.10.1.10	100.69.138.20	HAProxy + Keepalived (MASTER)
web1-paris	Paris	10.10.1.11	100.112.202.64	Nginx + FastAPI backend (ACTIVE)
web2-paris	Paris	10.10.1.12	100.95.209.14	Nginx + FastAPI backend (PASSIVE)
storage1-paris	Paris	10.10.1.21	100.122.223.27	GlusterFS brick 1 + iSCSI MASTER + Heartbe
storage2-paris	Paris	10.10.1.22	100.79.11.48	GlusterFS brick 2 + iSCSI SLAVE + Heartbeat
storage3-paris	Paris	10.10.1.23	100.65.222.115	GlusterFS brick 3
db-paris	Paris	10.10.1.31	100.126.63.76	MariaDB Galera nœud 1

Suite du tableau 2.1

VM	Site	VLAN Web	Tailscale	Rôle principal
proxysql-paris	Paris	10.10.1.32	100.97.25.14	ProxySQL (read/write split)
lb-ny	NY	10.10.2.10	100.84.166.8	HAProxy + Keepalived (MASTER)
web1-ny	NY	10.10.2.11	100.102.26.5	Nginx + FastAPI backend (ACTIVE)
web2-ny	NY	10.10.2.12	100.114.39.112	Nginx + FastAPI backend (PASSIVE)
storage1-ny	NY	10.10.2.21	100.93.154.48	GlusterFS brick 4 + iSCSI MASTER + Heartbe
storage2-ny	NY	10.10.2.22	100.103.234.33	GlusterFS brick 5 + iSCSI SLAVE + Heartbeat
storage3-ny	NY	10.10.2.23	100.121.176.58	GlusterFS brick 6
db-ny	NY	10.10.2.31	100.105.116.110	MariaDB Galera nœud 2
proxysql-ny	NY	10.10.2.32	100.100.6.35	ProxySQL (read/write split)
toronto	Toronto	—	100.126.8.98	GSLB HAProxy + Galera nœud 3 + backup +

Contrainte POC : consolidation Toronto

Le cahier des charges prévoit quatre VMs distinctes à Toronto. Nadia disposant d'une seule machine physique, les rôles backup, cold standby web, Galera nœud 3 et monitoring sont consolidés sur un seul hôte Debian 13. Chaque rôle est néanmoins implémenté avec ses propres services, ports et scripts, comme s'il était sur une VM dédiée. En production, ces rôles seraient sur quatre serveurs séparés (cf. `docs/architecture-justification.md`).

Périmètre POC vs cible production

Les éléments suivants apparaissent sur le diagramme physique comme *cibles de production* mais ne sont pas matériellement présents dans le POC :

- **Switches L2 Cisco Catalyst 9300** : dans le POC, la segmentation VLAN est obtenue par les interfaces `private_network` de Vagrant, qui s'appuient sur les switches virtuels host-only de VirtualBox. La topologie L2 est strictement équivalente du point de vue logique (broadcast domains séparés, ACLs par VLAN), mais ne constitue pas du matériel physique.
- **Onduleurs APC Smart-UPS 3000VA** : aucune onduline physique n'est déployée dans le POC, qui tourne sur des machines de l'équipe en home-lab. Les UPS figurent au diagramme comme exigence d'architecture pour la production.
- **Datacenter OVHcloud (Gravelines, Vint Hill, Beauharnois)** : les trois sites correspondent à des machines personnelles des trois membres de l'équipe (Ayman, Franck, Nadia). OVHcloud apparaît uniquement comme cible de migration dans la prévision budgétaire (`docs/financial-forecast.md`, chapitre **??**).

Cette transparence n'invalide pas la démonstration : l'ensemble des mécanismes logiques (HA, failover, réplication synchrone, chiffrement) est implémenté et testé sur l'environnement Vagrant.

2.6 Couches fonctionnelles

L'architecture se décompose en sept couches superposées, chacune indépendamment résiliente :

1. **Couche DNS / GSLB d'entrée** : dnsmasq Toronto résout `supfile.poc` vers une IP unique `100.126.8.98`, qui héberge un HAProxy GSLB faisant un health-check toutes les 5 secondes sur les LBs Paris et NY. Le datacenter en panne est automatiquement retiré du pool, garantissant un failover transparent intersite (<10 s) sans dépendance au cache DNS client.
2. **Couche load balancing** : HAProxy 2.4 en frontal HTTP, avec Keepalived gérant une VIP VRRP intra-datacenter pour la HA du load balancer lui-même.
3. **Couche web** : deux serveurs Nginx + FastAPI par datacenter, déclarés ACTIVE/BACKUP dans HAProxy, dont le *secondaire* ne reçoit aucun trafic en fonctionnement normal.
4. **Couche applicative** : backend FastAPI + Uvicorn sur port 8080, exposant l'API REST JSON.
5. **Couche base de données** : ProxySQL local sur port 6033 (read/write splitting), routant vers le cluster Galera 3 nœuds (db-paris, db-ny, toronto).
6. **Couche stockage** : GlusterFS volume Distributed-Replicate 2 × 3 briques monté sur `/mnt/supfile-storage` de tous les serveurs web. SAN iSCSI sur `tgt` avec VIP flottante Heartbeat pour la HA bloc.
7. **Couche supervision/sécurité** : Prometheus + Grafana centralisés sur Toronto, agents node_exporter sur les 17 VMs, Fail2ban 4 jails, Suricata IDS, ban cross-DC via Galera.

CHAPITRE 3

Réseau et plan d'adressage

3.1 Segmentation en quatre VLANs

L'infrastructure définit quatre réseaux logiques strictement isolés. Cette segmentation matérialise le principe de défense en profondeur : une compromission de la couche web ne donne pas accès au SAN ni aux interfaces d'administration.

TABLE 3.1 – Segmentation des VLANs et flux autorisés

VLAN	Plage	Usage
1 – Web/App	10.10.X.0/24	Trafic HTTP entre LB, web, et accès applicatif aux storage
2 – SAN	192.168.X.0/24	iSCSI uniquement entre storage1/2 et initiateurs web
3 – Management	192.168.99.0/24	SSH administrateur, monitoring Prometheus, agents Wazuh
4 – XDC overlay	100.X.X.X/32 (Tailscale)	Tunnels chiffrés inter-datacenter

L'isolation du VLAN SAN (rouge sur le diagramme logique) est une exigence de sécurité standard : le protocole iSCSI ne dispose pas par défaut d'authentification forte, et un attaquant capable de

joindre la cible `tgt` peut tenter de monter le LUN. En restreignant ce VLAN à un sous-réseau dédié sans route depuis le VLAN Web, on supprime cette attaque par construction.

Le VLAN Management est l'*out-of-band* de l'administration : même si la couche applicative ou un serveur web est compromis, l'opérateur conserve un chemin SSH pour intervenir. Prometheus interroge les agents `node_exporter` exclusivement sur ce VLAN.

3.2 Choix Tailscale vs WireGuard direct

WireGuard est la technologie de tunnel choisie pour le chiffrement inter-DC (voir chapitre 7). Sa configuration directe exige toutefois soit une IP fixe par site, soit un serveur de rendez-vous public. Les trois membres de l'équipe opérant depuis des réseaux domestiques NAT sans IP fixe, l'établissement de tunnels WireGuard purement P2P n'est pas pratiquement réalisable. **Tailscale** résout ce problème : il fournit un plan de contrôle (DERP/STUN) qui gère la traversée NAT, distribue automatiquement les clés publiques entre toutes les machines (190 paires possibles pour 20 nœuds), et utilise WireGuard ChaCha20-Poly1305 en transport sous-jacent. La sécurité du tunnel est identique à celle de WireGuard direct.

En production, sur un datacenter avec adresses IP fixes, Tailscale serait remplacé par WireGuard direct entre les load balancers (lb-paris ↔ lb-ny ↔ backup-toronto), comme documenté dans la configuration de référence `configs/wireguard/wg-reference.conf`.

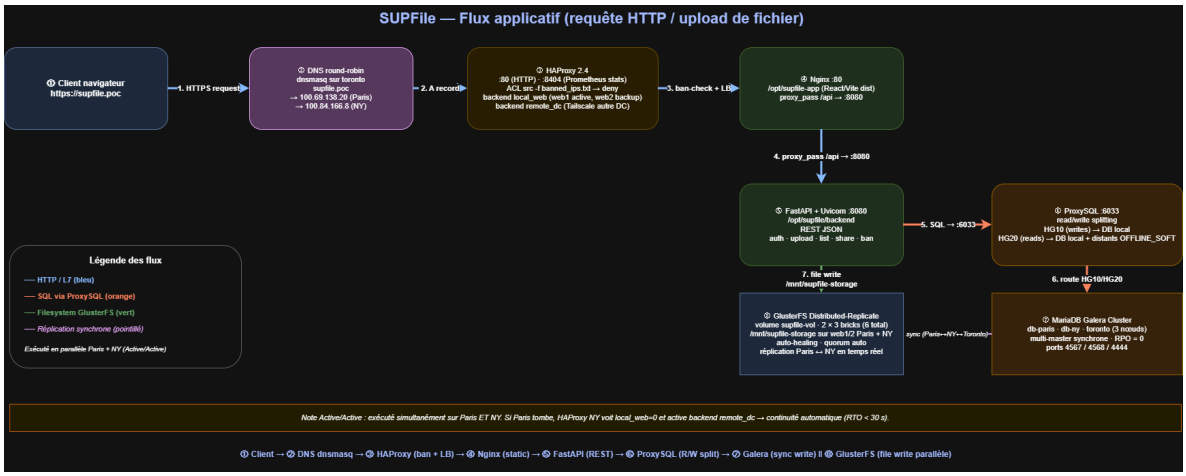

FIGURE 3.1 – Diagramme de flux applicatif : client → DNS → HAProxy → Nginx → FastAPI → ProxySQL → Galera + GlusterFS

3.3 Plan d'adressage complet

Le détail du plan d'adressage (VLAN Web, VLAN SAN, VLAN Management, adresses Tailscale et VIPs) est livré en annexe B. Les VIPs flottantes sont récapitulées ci-dessous :

Table 3.2 – Adresses IP virtuelles (VIP)

VIP	Adresse	Gestionnaire	Usage
Paris Web	10.10.1.100	Keepalived VRRP (lb-paris)	Frontal HAProxy Paris
Paris SAN	192.168.1.200	Heartbeat (storage1-paris)	Cible iSCSI Paris
NY Web	10.10.2.100	Keepalived VRRP (lb-ny)	Frontal HAProxy NY
NY SAN	192.168.2.200	Heartbeat (storage1-ny)	Cible iSCSI NY

3.4 Dimensionnement de la bande passante

Trois flux gros consommateurs cohabitent : le trafic utilisateur HTTP, la réplication GlusterFS, et la réplication synchrone Galera. L'iSCSI SAN reste local à un datacenter (VLAN SAN) et ne traverse jamais le WAN.

Table 3.3 – Estimation des besoins de bande passante inter-DC (charge moyenne)

Flux	Débit moyen	Calcul / hypothèse
Galera write replication	5–20 Mbps	100 TPS d'écritures à 10 Ko/transaction
GlusterFS replica heal	20–50 Mbps	uploads utilisateur $\approx$ 10 Mo/s sur 4 web actifs
HAProxy cross-DC fallback	$<$ 1 Mbps	health checks toutes les 5 s
Tailscale heartbeat	$<$ 0,1 Mbps	keepalives DERP
Total inter-DC	**25–75 Mbps**	Lien 100 Mbps minimum recommandé en production

Un lien de 1 Gbps inter-DC offre une marge confortable pour absorber les pics (heal massif après défaillance) sans saturation.

CHAPITRE 4

Couche web et load balancing

4.1 Choix de HAProxy comme load balancer

HAProxy 2.4 est déployé comme load balancer L7 sur les nœuds `lb-paris` et `lb-ny`. Le choix s'est porté sur HAProxy plutôt que sur Nginx en mode reverse-proxy pour trois raisons techniques décisives.

Premièrement, la directive `backup`. Elle permet de déclarer un serveur en mode *standby pur* : il ne reçoit aucun trafic en fonctionnement normal, mais reçoit la totalité du trafic dès que tous les serveurs primaires sont indisponibles. Nginx ne dispose pas de cette primitive nativement ; sa directive `down` requiert une intervention manuelle et l'`ngx_http_upstream_module` en version commerciale offre uniquement du basculement reposant sur des health checks limités.

Deuxièmement, les statistiques temps réel. HAProxy expose nativement une page `/stats` (HTML, JSON, CSV, Prometheus) qui montre l'état de santé de chaque backend, les temps de réponse, les compteurs de connexion et le statut des health checks. Cette télémétrie est essentielle pour le diagnostic en production et pour l'exporter Prometheus natif (port 8404, endpoint `/metrics`).

Troisièmement, les health checks fins. La directive `check inter 2s fall 3 rise 2` permet de détecter une panne en moins de 6 secondes (3 checks × 2 s) et de promouvoir le backup automatiquement, sans intervention humaine. Les tests T03 et T04 ont mesuré un temps de bascule effectif de 10 à 12 secondes (latence TCP comprise).

4.2 Configuration HAProxy de production

L'extrait suivant illustre la configuration du frontal HTTP sur `lb-paris`, avec un backend local (web1 actif, web2 backup) et un backend distant (NY) servant de fallback inter-DC.

Listing 4.1 – Extrait `configs/haproxy/lb-paris.cfg` – frontends et backends

```
frontend http_front
    bind 10.10.1.100:80
    bind *:80
    mode http
    option httplog
    acl banned src -f /etc/haproxy/banned_ips.txt
    http-request deny if banned
    default_backend local_web
    use_backend remote_dc if { nbsrv(local_web) eq 0 }

```

```
backend local_web
    mode http
    balance roundrobin
    option httpchk GET /health
    server web1 10.10.1.11:80 check inter 2s fall 3 rise 2
    server web2 10.10.1.12:80 check inter 2s fall 3 rise 2 backup

backend remote_dc
    mode http
    balance roundrobin
    option httpchk GET /health
    server remote 100.84.166.8:80 check inter 5s

frontend stats
    bind *:8404
    mode http
    http-request use-service prometheus-exporter if { path /metrics }
    stats enable
    stats uri /stats
    stats refresh 10s
    stats auth admin:SUPFile2024!
```

La directive `use_backend remote_dc if { nbsrv(local_web) eq 0 }` active le fallback cross-DC dès que zéro serveur local est UP, garantissant une continuité de service même en cas de chute complète du datacenter local.

4.3 Keepalived et VIP virtuelle

Keepalived implémente le protocole VRRP (RFC 5798) pour gérer une IP virtuelle (VIP) flottante entre deux instances HAProxy. La VIP est l'adresse que les clients résolvent et contactent : la réelle IP de l'instance MASTER importe peu, ce qui rend le basculement transparent.

Listing 4.2 – Extrait `configs/keepalived/lb-paris.conf`

```
vrrp_instance VI_1 {
    state MASTER
    interface enp0s8
    virtual_router_id 51
    priority 101
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass SUPF1L3!
    }
    virtual_ipaddress {
        10.10.1.100/24
    }
    track_script {
        chk_haproxy
    }
}

```

```
vrrp_script chk_haproxy {
    script "/usr/bin/killall -0 haproxy"
    interval 2
    weight -50
}
```

Si HAProxy s'arrête (`killall -0` retourne non zéro), la priorité est décrémentée de 50, passant en dessous de la priorité du *backup* qui acquiert la VIP en moins de 3 secondes. Le client maintient sa connexion sur la même adresse : seul le serveur physique a changé.

4.4 Active/Passive intra-datacenter : bascule mesurée

Le scénario nominal de bascule intra-DC est : `web1` actif sert tout le trafic, `web2` reste en attente `BACKUP` dans HAProxy. À l'arrêt de `web1`, le health check tombe en `DOWN` après 6 s (3 × 2 s) et HAProxy bascule sur `web2`. Le test T03 a confirmé le scénario sur Paris avec preuve dans `docs/evidence/12-active-passive-failover.txt` :

Listing 4.3 – Sortie test T03 / preuve 12-active-passive-failover.txt

```
=== AVANT (web1 actif) ===
{"service":"SUPFile","node":"web1-paris","status":"ok","version":"1.0-poc"}

=== APRÈS arrêt de web1 (web2 prend le relais) ===
{"service":"SUPFile","node":"web2-paris","status":"ok","version":"1.0-poc"}

=== APRÈS redémarrage de web1 (rétabli) ===
{"service":"SUPFile","node":"web1-paris","status":"ok","version":"1.0-poc"}
```

4.5 Active/Active inter-datacenter

La distribution Active/Active entre Paris et New York est portée par deux mécanismes complémentaires :

1. **HAProxy GSLB sur Toronto** : l'enregistrement `supfile.poc` pointe vers une IP unique `100.126.8.98`. Le HAProxy de Toronto effectue un health-check HTTP `GET /health` toutes les 5 secondes sur les deux load balancers (`100.69.138.20` Paris et `100.84.166.8` NY) et répartit les nouvelles connexions entre les datacenters sains uniquement. Un DC complètement indisponible (LB inclus) est retiré du pool en moins de 15 secondes (3 checks × 5 s). Cette approche corrige la limite connue du DNS round-robin pur, dont la résolution est aveugle aux pannes côté serveur et dépend du comportement client (Happy Eyeballs, TTL cache) pour le re-routage. Test T-GSLB validé le 2026-05-24.
2. **Fallback HAProxy cross-DC** : chaque HAProxy de DC contient également un backend `remote_dc` pointant vers l'IP Tailscale du LB pair. Si le backend local n'a plus de serveur web UP mais que le LB lui-même répond, HAProxy active le fallback transparent vers le DC distant via Tailscale (<30 s).

Les deux mécanismes sont superposés : le GSLB de Toronto gère la panne d'un datacenter entier (LB *et* backends morts), le fallback HAProxy local gère la panne de l'ensemble des backends quand le LB reste joignable.

```
PS D:\Downloads\Brave\SUPFILE\supfile> Invoke-WebRequest -Uri "http://100.69.138.20/" -UseBasicParsing
 | Select-Object -ExpandProperty Content
{"service":"SUPFile","node":"web2-paris","status":"ok","version":"1.0-poc"}
PS D:\Downloads\Brave\SUPFILE\supfile> Invoke-WebRequest -Uri "http://100.84.166.8/" -UseBasicParsing
| Select-Object -ExpandProperty Content
{"service":"SUPFile","node":"web1-ny","status":"ok","version":"1.0-poc"}
```

FIGURE 4.1 – Réponses simultanées des deux datacenters Paris et NY (Active/Active)

CHAPITRE 5

Stockage : iSCSI + RAID-1 + GlusterFS

5.1 Architecture en deux couches

Le stockage SUPFile s'appuie sur une architecture en deux couches indépendantes, chacune répondant à un besoin différent.

La **couche bloc (iSCSI + RAID-1)** fournit un volume bloc résilient à partir de disques physiques. Elle répond à l'exigence *SAN iSCSI* du cahier des charges et est utilisée en production pour des données qui nécessitent un accès bloc rapide (stockage métier d'une base de données locale, volumes de swap partagés). La résilience est intra-datacenter : la VIP iSCSI flotte entre `storage1` et `storage2`, le RAID-1 protège contre la perte d'un disque physique.

Portée POC : couche bloc fonctionnelle mais non consommée par l'application

Dans le POC, la cible iSCSI `tgt` exporte effectivement le LUN `/dev/md0` sur les VIPs SAN (192.168.1.200 / 192.168.2.200) et le failover Heartbeat est validé (tests T11/T12 et T11-NY/T12-NY). Cependant, l'application SUPFile elle-même n'attache pas le LUN : les serveurs web utilisent exclusivement le mount GlusterFS `/mnt/supfile-storage` pour le stockage primaire. Les paquets `open-iscsi` sont installés et le démon `iscsid` actif sur web1/web2, mais aucune session iSCSI n'est ouverte (`iscsiadm -m session` retourne vide). La couche bloc constitue donc une démonstration technique du SAN exigé par le cahier des charges, sans charge applicative réelle.

La **couche fichier (GlusterFS)** fournit un système de fichiers POSIX distribué et répliqué, monté sur tous les serveurs web des deux datacenters. Elle est utilisée pour le stockage primaire des fichiers utilisateur, qui doivent être accessibles depuis n'importe quel nœud web Paris ou NY, et survivre à la perte d'un datacenter entier. C'est l'unique chemin de stockage emprunté par l'API SUPFile à l'écriture comme à la lecture.

5.2 SAN iSCSI logiciel via `tgt`

5.2.1 Justification du choix

Le cahier des charges spécifie explicitement *SAN iSCSI*. Le protocole iSCSI (RFC 3720) transporte les commandes SCSI sur IP et est par nature compatible avec du matériel standard. Une baie SAN Fibre Channel matérielle (NetApp, EMC, Dell PowerVault) coûte entre 50 000 € et 500 000 € et

n'est pas reproductible dans un POC. Le démon `tgt` (Linux SCSI Target Framework) implémente la pile iSCSI complète : authentification CHAP, ACLs par sous-réseau, gestion des LUNs et des persistent reservations. Du point de vue de l'initiateur (les serveurs web), il n'y a aucune différence entre une cible `tgt` et une baie matérielle.

5.2.2 Configuration de la cible

Listing 5.1 – Configuration iSCSI target sur storage1-paris

```
<target iqn.2024-01.com.supfile:storage>
    backing-store /dev/md0
    initiator-address 10.10.1.0/24
    incominguser supfile_init SuPfile_init_2024!
</target>
```

La cible expose le volume RAID-1 (`/dev/md0`) à tous les initiateurs du VLAN Web. L'authentification CHAP empêche un nouvel hôte non autorisé de monter le LUN.

5.2.3 Haute disponibilité avec Heartbeat

Heartbeat gère la HA de la cible iSCSI via une VIP flottante (192.168.1.200 à Paris, 192.168.2.200 à NY). Le démon échange des keepalives UDP toutes les secondes sur le VLAN SAN entre `storage1` (MASTER) et `storage2` (SLAVE). Si MASTER disparaît, SLAVE acquiert la VIP en moins de 3 secondes, démarre son propre `tgt`, et les initiateurs `open-iscsi` sur les serveurs web reconnectent automatiquement à la même VIP – la coupure est de l'ordre de la dizaine de secondes.

Listing 5.2 – Extrait `/etc/ha.d/ha.cf` – Heartbeat storage1-paris

```
ucast enp0s10 192.168.1.22
keepalive 1
deadtime 10
warntime 5
initdead 60
node storage1-paris
node storage2-paris
auto_failback on
```

5.3 RAID-1 et hot-swap

5.3.1 Principe

Le RAID-1 par `mdadm` miroire deux disques physiques. Tout bloc écrit est dupliqué sur les deux disques en parallèle ; la perte d'un disque n'entraîne aucune perte de donnée et l'autre disque continue à servir l'I/O sans interruption. La procédure de remplacement à chaud :

Listing 5.3 – Procédure de remplacement disque à chaud

```
# 1. Marquer le disque défaillant
sudo mdadm --manage /dev/md0 --fail /dev/sdb1

# 2. Retirer logiquement le disque
```

```
sudo mdadm --manage /dev/md0 --remove /dev/sdb1

# 3. Remplacer physiquement le disque (hot-swap baie SAS/SATA)

# 4. Réinjecter le nouveau disque dans le miroir
sudo mdadm --manage /dev/md0 --add /dev/sdb1

# 5. Vérifier la resynchronisation
cat /proc/mdstat
```

5.3.2 Limite du POC

Limitation Vagrant : RAID-1 dégradé documenté

Dans l'environnement Vagrant, chaque VM dispose de disques virtuels isolés ; il n'est pas possible de monter le VDI d'une VM sur une autre. Le second membre du miroir devrait être un LUN iSCSI exporté par `storage2`, mais le disque additionnel `sdb` créé via Vagrant ne fait que 10 Mo, insuffisant. Le RAID démarre donc à l'état `[1/2] [U_]` (dégradé mais fonctionnel). Cette limite est documentée dans les tests T13 et T13-NY, et résolue en production par deux disques SAS physiques de capacité réelle dans la même baie. La démonstration logique du concept (failover, hot-swap, intégrité) reste valide.

5.4 GlusterFS Distributed-Replicate

5.4.1 Topologie du volume

Le volume `supfile-vol` est de type *Distributed-Replicate* avec 6 briques organisées en 2 jeux de réplique 3 : trois briques Paris (storage1/2/3) et trois briques NY (storage1/2/3). Chaque fichier est distribué (sharding par hash) entre les deux jeux, puis répliqué sur les trois briques de son jeu.

Listing 5.4 – État du volume GlusterFS (extrait preuve 16-glusterfs-info.txt)

```
Volume Name: supfile-vol
Type: Distributed-Replicate
Volume ID: 0df36ad8-3c8b-484d-9e37-b22cdc9b0b8d
Status: Started
Number of Bricks: 2 x 3 = 6
Bricks:
  Brick1: 100.122.223.27:/data/brick1/gluster   (storage1-paris)
  Brick2: 100.79.11.48:/data/brick2/gluster     (storage2-paris)
  Brick3: 100.65.222.115:/data/brick3/gluster   (storage3-paris)
  Brick4: 100.93.154.48:/data/brick1/gluster    (storage1-ny)
  Brick5: 100.103.234.33:/data/brick2/gluster   (storage2-ny)
  Brick6: 100.121.176.58:/data/brick3/gluster   (storage3-ny)
Options Reconfigured:
  performance.client-io-threads: off
  cluster.granular-entry-heal: on
  network.ping-timeout: 10
```

5.4.2 Justification du choix vs NFS et Ceph

NFS centralise les fichiers sur un seul serveur : c'est par construction un point unique de défaillance (SPOF). Ceph est plus performant et scalable mais sa complexité opérationnelle (RADOS, OSD, MON, MDS, placement groups) est disproportionnée pour un POC d'équipe – Ceph demande au moins 3 MON et un dimensionnement précis des PGs avant toute mise en production. GlusterFS offre le meilleur compromis : décentralisé (pas de SPOF), *auto-healing* (re-synchronisation automatique d'une brique tombée), configuration en moins de 100 lignes de script.

5.4.3 Quorum et survie aux pannes

Avec une topologie 2 × 3, GlusterFS peut survivre à la perte de jusqu'à 2 briques par jeu de réplique (n'importe quelle combinaison qui laisse au moins 1 brique disponible par jeu) sans interruption de service. La perte simultanée des 3 briques d'un même jeu (par exemple les 3 briques Paris) rend les fichiers stockés dans ce jeu inaccessibles, mais les fichiers du second jeu (sur NY) restent disponibles. En pratique, la probabilité conjointe de cet événement est très faible si les datacenters sont géographiquement et électriquement indépendants.

```
PS D:\Downloads\Brave\SUPFILE\supfile>   vagrant ssh storage1-paris -c "sudo gluster volume status sup
file-vol"
Status of volume: supfile-vol
Gluster process                                 TCP Port  RDMA Port  Online  Pid
------------------------------------------------------------------------------
Brick 100.122.223.27:/data/brick1/gluster       53918     0          Y       1201
Brick 100.79.11.48:/data/brick2/gluster         51345     0          Y       2123
Brick 100.65.222.115:/data/brick3/gluster       59425     0          Y       1034
Brick 100.93.154.48:/data/brick1/gluster        53305     0          Y       14366
Brick 100.103.234.33:/data/brick2/gluster       52744     0          Y       14927
Brick 100.121.176.58:/data/brick3/gluster       55142     0          Y       16558
Self-heal Daemon on localhost                   N/A       N/A        Y       1217
Self-heal Daemon on 100.79.11.48                N/A       N/A        Y       2134
Self-heal Daemon on 100.65.222.115              N/A       N/A        Y       1049
Self-heal Daemon on 100.103.234.33              N/A       N/A        Y       14938
Self-heal Daemon on 100.93.154.48               N/A       N/A        Y       14377
Self-heal Daemon on 100.121.176.58              N/A       N/A        Y       16572

Task Status of Volume supfile-vol
------------------------------------------------------------------------------
There are no active volume tasks
```

FIGURE 5.1 – GlusterFS 6/6 briques en ligne, status Y (en bonne santé)

5.5 Réplication cross-DC validée

Le test T10 a écrit un fichier sur `web1-paris` via le mount GlusterFS et vérifié sa présence immédiate sur `web1-ny`. La preuve est dans `docs/evidence/19-glusterfs-replication.txt` :

Listing 5.5 – Test T10 : réplication GlusterFS Paris → NY

```
# Sur web1-paris :
echo "evidence-gluster-test-$(date)" > /mnt/supfile-storage/test.txt

# Sur web1-ny (immédiatement) :
cat /mnt/supfile-storage/test.txt
> evidence-gluster-test-Mon May 18 14:21:15 UTC 2026
```

CHAPITRE 6

Base de données : Galera + ProxySQL

6.1 Choix de Galera Cluster

MariaDB Galera Cluster est la solution retenue pour la couche base de données, en lieu et place d'une réplication MySQL standard *primary/replica*. Trois raisons fondamentales motivent ce choix.

Réplication synchrone multi-master. Galera certifie chaque transaction sur tous les nœuds avant de valider : une fois qu'un `COMMIT` retourne SUCCESS au client, on a la garantie que la transaction est présente sur les trois nœuds. Le RPO (perte de données maximale en cas de panne) est donc strictement zéro. Avec la réplication MySQL asynchrone, un `primary` peut valider localement avant que le `replica` n'ait reçu le binlog – en cas de crash du `primary`, plusieurs transactions confirmées peuvent être perdues.

Multi-master. N'importe quel nœud Galera peut recevoir des écritures. Cela permet à ProxySQL de chaque DC de router les écritures vers le nœud DB *local*, supprimant les 80 ms de latence inter-DC sur les opérations courantes. Toutes les écritures sont ensuite répliquées synchroniquement vers les autres nœuds en arrière-plan.

Tolérance aux pannes. Avec 3 nœuds (cluster_size = 3), Galera tolère la perte d'1 nœud sans interruption : 2 nœuds restants > quorum (1,5). Si Paris ET New York tombent simultanément, Toronto seul ne maintient pas le quorum et bascule en lecture seule – scénario T3 du PRA nécessitant un `bootstrap` manuel.

Listing 6.1 – État du cluster Galera (preuve docs/evidence/14-galera-status.txt)

```
=== db-paris ===
wsrep_local_state_comment    Synced
wsrep_cluster_size           3
wsrep_cluster_status         Primary
wsrep_ready                  ON

=== db-ny ===
wsrep_cluster_size           3
wsrep_ready                  ON

=== toronto (100.126.8.98) ===
wsrep_cluster_size           3
```

```
PS D:\Downloads\Brave\SUPFILE\supfile>   vagrant ssh db-paris -c "echo 'SHOW GLOBAL STATUS' | sudo mys
ql -u root | grep -E 'wsrep_cluster_size|wsrep_ready|wsrep_local_state_comment'"
wsrep_local_state_comment       Synced
wsrep_cluster_size      3
wsrep_ready     ON
```

FIGURE 6.1 – wsrep_cluster_size=3, wsrep_ready=ON sur db-paris

6.2 ProxySQL en interposition

ProxySQL s'interpose systématiquement entre l'application (port 6033) et les nœuds Galera (port 3306). Il fournit trois services critiques que l'application n'a pas à implémenter elle-même.

Read/write splitting. ProxySQL maintient deux *hostgroups* (HG) : HG10 contient les serveurs autorisés à recevoir les écritures (le nœud DB local du DC), HG20 contient les serveurs autorisés à recevoir les lectures (tous les nœuds, avec poids). Les règles `mysql_query_rules` routent automatiquement : tout `SELECT` hors transaction va vers HG20, tout `INSERT/UPDATE/DELETE` ou `SELECT FOR UPDATE` va vers HG10.

Health checking. Le module `galera_checker` interroge périodiquement `wsrep_local_state` sur chaque nœud et retire automatiquement les nœuds en état `JOINING` ou `DONOR` du pool. Si le nœud local tombe en `OFFLINE`, ProxySQL promeut automatiquement un nœud distant initialement marqué `OFFLINE_SOFT`.

Connection pooling. ProxySQL maintient un pool de connexions persistantes vers MariaDB, évitant à l'application d'ouvrir/fermer des connexions TCP+handshake à chaque requête. Le gain est significatif sous forte concurrence.

Listing 6.2 – Configuration ProxySQL Paris (extrait 24-proxysql-paris.txt)

```
hostgroup_id  hostname          port  status        weight
10            10.10.1.31        3306  ONLINE        1000
20            10.10.1.31        3306  ONLINE        1000
20            100.105.116.110   3306  OFFLINE_SOFT  300
20            100.126.8.98      3306  OFFLINE_SOFT  300
```

```
PS D:\Downloads\Brave\SUPFILE\supfile> vagrant ssh proxysql-paris -c "mysql -u admin -pAdmin2024! -h 1
27.0.0.1 -P6032 -e 'SELECT hostgroup_id,hostname,status FROM mysql_servers;'"
+--------------+-----------------+--------------+
| hostgroup_id | hostname        | status       |
+--------------+-----------------+--------------+
| 10           | 10.10.1.31      | ONLINE       |
| 20           | 10.10.1.31      | ONLINE       |
| 20           | 100.105.116.110 | OFFLINE_SOFT |
| 20           | 100.126.8.98    | OFFLINE_SOFT |
+--------------+-----------------+--------------+
```

FIGURE 6.2 – ProxySQL mysql_servers – ONLINE local + OFFLINE_SOFT distants

6.3 Schéma de la base de données

La base `supfile` est conçue pour le moteur InnoDB (transactions ACID, clés étrangères, row-level locking). Le schéma de référence couvre les concepts du cahier des charges : utilisateurs avec

quota 30 Go, fichiers/dossiers en arborescence, sessions, liens de partage, nœuds de stockage, bannissements IP et journal d'audit.

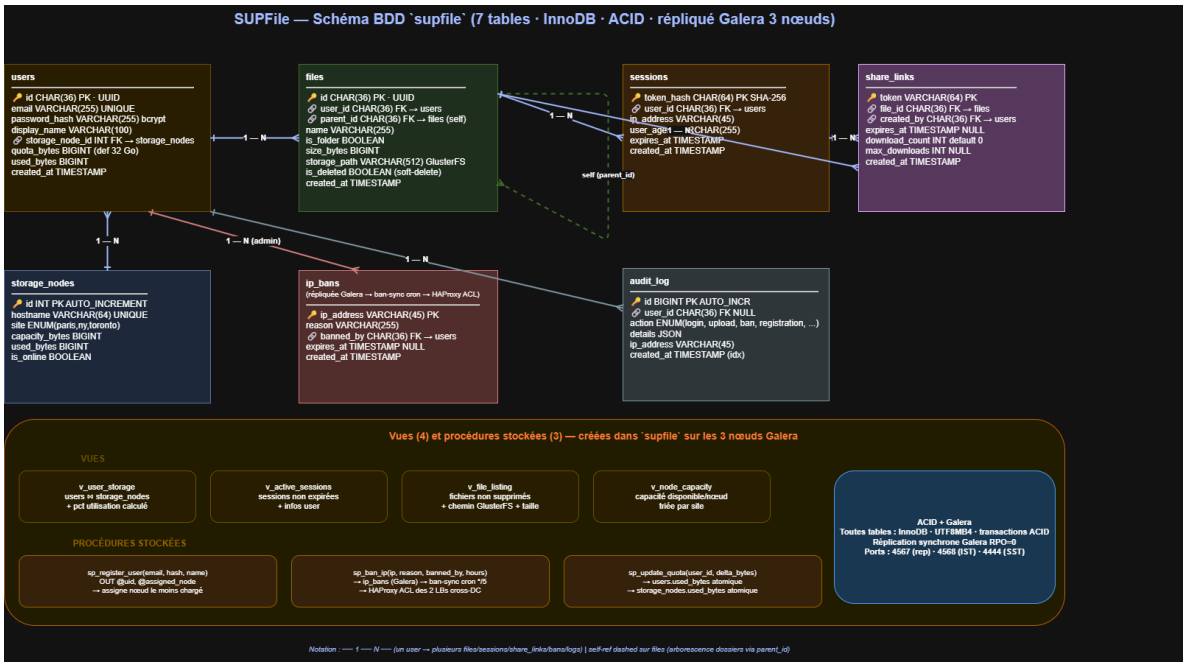

FIGURE 6.3 – Schéma entité-relation (ERD) de la base `supfile`

6.3.1 Tables de référence

TABLE 6.1 – Tables de la base SUPFile (schéma de référence)

Table	Description
`users`	Comptes utilisateur, quota 30 Go (32 212 254 720 octets), assignation au nœud de stockage le moins chargé.
`files`	Arborescence fichiers/dossiers (auto-référence `parent_id`), chemin GlusterFS, soft-delete via `is_deleted`, support du versionnement.
`sessions`	Tokens de session avec hash SHA-256, expiration, IP et user-agent de connexion.
`share_links`	Liens publics partageables avec token unique, expiration optionnelle, compteur de téléchargements.
`storage_nodes`	Registre des nœuds de stockage par site (paris/ny/toronto), capacité utilisée et capacité totale.
`ip_bans`	Bannissements IP avec expiration, raison, auteur – répliqués cross-DC par Galera.
`audit_log`	Journal append-only des actions sensibles avec JSON extensible (login, upload, ban, registration).

6.3.2 Vues

TABLE 6.2 – Vues utilitaires

Vue	Description
`v_user_storage`	Utilisateurs avec pourcentage de quota utilisé (jointure `users` + `storage_nodes`).
`v_active_sessions`	Sessions non expirées avec utilisateur.
`v_file_listing`	Fichiers non supprimés avec chemin GlusterFS et taille.
`v_node_capacity`	Capacité disponible par nœud, triée par site et utilisation.

6.3.3 Procédures stockées critiques

TABLE 6.3 – Procédures stockées de référence

Procédure	Description
`sp_register_user(email, hash, name, @uid, @node)`	Crée un utilisateur, assigne le nœud de stockage le moins utilisé via une jointure sur `v_node_capacity`. Retourne UUID et hostname assigné.
`sp_ban_ip(ip, reason, banned_by, hours)`	Insère un bannissement dans `ip_bans`, répliqué par Galera. Le script cron `ban-sync.sh` régénère ensuite `banned_ips.txt` sur chaque LB.
`sp_update_quota(user_id, delta_bytes)`	Met à jour atomiquement `users.used_bytes` et `storage_nodes.used_bytes` associé. Lève une erreur si le quota 30 Go serait dépassé.

6.3.4 Test d'assignation du nœud de stockage

Le test T19 a validé que l'appel à `sp_register_user()` retourne bien le nœud le moins utilisé (preuve `docs/evidence/32-stored-proc.txt`) :

```
mysql> CALL sp_register_user('test@supfile.io', SHA2('pw',256), 'Test',
                             @uid, @node);
mysql> SELECT @uid AS user_id, @node AS assigned_node;
+--------------------------------------+----------------+
| user_id                              | assigned_node  |
+--------------------------------------+----------------+
| 595ea7d9-52c7-11f1-9b00-02aaf8fcc71b | storage2-paris |
+--------------------------------------+----------------+
```

Note sur le schéma applicatif déployé

Le schéma déployé dans le POC inclut en plus du schéma de référence des tables propres à l'application backend FastAPI livrée (`file_versions`, `file_metadata`, `file_comments`, `oauth_accounts`, etc.). Ces tables sont gérées par les migrations Alembic de l'application et ne remplacent pas le schéma de référence ; elles l'étendent pour couvrir les fonctionnalités collaboratives et l'authentification OAuth. La validation cross-DC Galera porte sur l'ensemble des tables sans distinction.

CHAPITRE 7

Sécurité et défense en profondeur

7.1 Principe directeur

La stratégie de sécurité repose sur la *défense en profondeur* : plusieurs couches indépendantes protègent l'infrastructure, de sorte qu'une défaillance d'une couche n'expose pas le système entier. L'attaquant doit franchir chacune d'elles successivement. Quatre couches sont déployées : pare-feu hôte (UFW), bannissement comportemental (Fail2ban), inspection profonde de paquets (Suricata IDS/IPS), et bannissement applicatif synchronisé entre datacenters (Galera + HAProxy ACL).

7.2 Pare-feu hôte : UFW

UFW est configuré sur toutes les VMs en mode *default-deny* entrant, avec autorisation explicite par port et par sous-réseau source. Seuls les ports strictement nécessaires au rôle de chaque VM sont ouverts :

TABLE 7.1 – Ports ouverts par rôle (sortie `ufw status` agrégée)

Rôle	**Ports ouverts**
Load balancer	22 (SSH mgmt), 80/443 (HTTP), 8404 (stats), 41641 (Tailscale)
Serveur web	22, 80, 9100 (node_exporter), 41641
Stockage	22, 3260 (iSCSI sur VLAN SAN), 24007–49200 (GlusterFS), 9100, 41641
Base de données	22, 3306 (MySQL), 4567/4568/4444 (Galera SST/IST), 9100, 41641
ProxySQL	22, 6032 (admin), 6033 (mysql), 9100, 41641
Toronto	22, 3306, 3000 (Grafana), 9090 (Prometheus), 53 (DNS), 41641

7.3 Bannissement comportemental : Fail2ban

Fail2ban surveille les logs des services exposés (SSH, Nginx, HAProxy) et bannit automatiquement les IPs après N tentatives échouées via une règle `iptables` dynamique. Quatre jails sont actives sur chaque VM :

TABLE 7.2 – Jails Fail2ban et seuils de bannissement

Jail	Source surveillée	Seuil	Durée
sshd	`/var/log/auth.log`	5 tentatives	1 heure
nginx-http-auth	`/var/log/nginx/error.log`	10 tentatives	10 minutes
nginx-limit-req	`/var/log/nginx/error.log`	20 req/min	30 minutes
haproxy-http-auth	`/var/log/haproxy.log`	10 tentatives	30 minutes

Le test T15 a validé le bannissement automatique après 4 tentatives SSH échouées sur Paris (preuve `docs/evidence/26-fail2ban-paris.txt`). Idem pour NY (T15-NY).

7.4 IDS/IPS : Suricata sur les serveurs web

Suricata est déployé en mode IPS (Inline Packet Inspection) sur les quatre serveurs web (web1/web2 Paris + NY). Il inspecte chaque paquet entrant sur l'interface du VLAN Web et applique le ruleset *ET Open* (Emerging Threats), soit environ 35 000 signatures couvrant les injections SQL, LFI, scanners de ports, agents malveillants, exploits CVE récents.

Listing 7.1 – Extrait `/etc/suricata/suricata.yaml`

```
af-packet:
  - interface: enp0s8
    cluster-id: 99
    cluster-type: cluster_flow
    defrag: yes
    use-mmap: yes

vars:
  address-groups:
    HOME_NET: "[10.10.1.0/24,10.10.2.0/24,192.168.99.0/24]"
    EXTERNAL_NET: "!$HOME_NET"
```

En mode `af-packet` avec `defrag: yes`, Suricata réassemble les paquets fragmentés pour détecter les attaques qui tentent de se dissimuler via la fragmentation. La variable `EXTERNAL_NET: "!$HOME_NET"` (négation) permet à la majorité des règles ET Open de fonctionner correctement – la moindre faute de frappe (par exemple oubli du `$`) désactiverait silencieusement la majorité du ruleset.

7.5 Bannissement cross-DC via Galera

Le mécanisme de bannissement HAProxy fonctionne en quatre étapes synchronisées entre datacenters :

1. Un administrateur (ou Fail2ban en automatique) appelle `CALL sp_ban_ip(ip, reason, banned_by, hours)` sur n'importe quel nœud Galera.

2. La ligne insérée dans la table `ip_bans` se réplique *instantanément* (synchrone) sur les trois nœuds Galera (Paris, NY, Toronto).
3. Un cron `*/5 * * * *` sur chaque LB exécute `/usr/local/bin/ban-sync.sh` : ce script lit `ip_bans` via ProxySQL (port 6033) et régénère `/etc/haproxy/banned_ips.txt`.
4. HAProxy relit le fichier ACL (`src -f /etc/haproxy/banned_ips.txt`) et applique `http-request deny if banned` sur toute requête entrante depuis une IP bannie.

Listing 7.2 – Script `/usr/local/bin/ban-sync.sh`

```
#!/usr/bin/env bash
set -euo pipefail
TMP=$(mktemp)
mysql -h 127.0.0.1 -P 6033 -u supfile_ro -pReadOnly2024 supfile \
  -se "SELECT ip FROM ip_bans WHERE expires_at IS NULL OR expires_at > NOW();" \
  > "$TMP"
install -m 0644 "$TMP" /etc/haproxy/banned_ips.txt
systemctl reload haproxy
rm -f "$TMP"
```

Le test T17 a validé qu'un `CALL sp_ban_ip` sur `db-paris` est propagé sur `lb-ny` et entraîne un retour HTTP 403 sur `lb-ny` pour cette IP en moins de 5 minutes (intervalle du cron).

7.6 Chiffrement inter-datacenter

Toutes les communications entre les 17 VMs des trois datacenters transitent par WireGuard (sous-jacent à Tailscale dans ce POC). WireGuard utilise :

— **ChaCha20-Poly1305** pour le chiffrement authentifié des données (AEAD, performant sur CPU sans AES-NI).
— **Curve25519** pour l'échange de clés Diffie-Hellman (ECDH sur courbe elliptique).
— **BLAKE2s** pour les MACs (message authentication codes).

Le handshake est régénéré toutes les 3 minutes et les clés de session sont rotées toutes les 5 minutes (*Perfect Forward Secrecy*). Chaque paire de VMs établit un tunnel chiffré indépendant : la compromission partielle d'un VPN ne donne pas accès aux autres paires.

Les clés publiques de référence sont archivées dans `docs/evidence/37-wireguard-keys.txt`. La configuration de production directe est documentée dans `configs/wireguard/wg-reference.conf`.

7.7 Politique d'authentification

— **SSH** : authentification par clé publique exclusivement, mot de passe désactivé en production (`PasswordAuthentication no` dans `sshd_config`).
— **Base de données** : utilisateurs cloisonnés par fonction : `supfile_app` (lecture/écriture sur tables applicatives), `supfile_ro` (lecture seule pour le ban-sync), `supfile_sst` (SST Galera uniquement), `supfile_monitor` (`REPLICATION CLIENT` pour Prometheus).
— **HAProxy stats** : HTTP Basic Auth, accessible uniquement depuis le VLAN Management en production.

7.8 Conformité GDPR et résidence des données

L'infrastructure étant déployée sur OVHcloud Gravelines (Paris), Vint Hill (US) et Beauharnois (Canada), les données utilisateur transitent par trois juridictions. Les utilisateurs européens sont rattachés au DC Paris par préférence DNS (à mettre en place via GeoDNS en production). Les sauvegardes sur Toronto sont chiffrées au repos via LUKS sur le volume `/backup`. Les exports utilisateur (RGPD – droit à la portabilité) sont générés à la demande depuis l'API REST.

CHAPITRE 8

Supervision et observabilité

8.1 Architecture Prometheus + Grafana

La supervision centralisée est hébergée sur le nœud Toronto (`100.126.8.98`). Prometheus collecte les métriques de tous les nœuds actifs via l'agent `node_exporter` (port 9100/tcp) avec un intervalle de scrape de 15 secondes. Node exporter expose plus de 800 métriques par hôte : CPU (par cœur, par mode), mémoire (RSS, cache, buffers, swap), réseau (bytes in/out par interface, erreurs, drops), disque (IOPS, latence, utilisation par partition), descripteurs de fichiers, et état des services systemd.

8.1.1 Configuration des jobs Prometheus

La configuration cible les 17 VMs via leurs IPs Tailscale, regroupées en 5 jobs :

Listing 8.1 – Extrait `/etc/prometheus/prometheus.yml` sur toronto

```
scrape_configs:
  - job_name: "paris-nodes"
    static_configs:
      - targets:
        - "100.69.138.20:9100"   # lb-paris
        - "100.112.202.64:9100"  # web1-paris
        - "100.95.209.14:9100"   # web2-paris
        - "100.122.223.27:9100"  # storage1-paris
        - "100.79.11.48:9100"    # storage2-paris
        - "100.65.222.115:9100"  # storage3-paris
        - "100.126.63.76:9100"   # db-paris
        - "100.97.25.14:9100"    # proxysql-paris
        labels:
          site: paris

  - job_name: "ny-nodes"
    static_configs:
      - targets:
        - "100.84.166.8:9100"
        - "100.102.26.5:9100"
        - "100.114.39.112:9100"
        - "100.93.154.48:9100"
        - "100.103.234.33:9100"
        - "100.121.176.58:9100"
        - "100.105.116.110:9100"
        - "100.100.6.35:9100"
        labels:
```

```
        site: ny

  - job_name: "toronto-nodes"
    static_configs:
      - targets: ["100.126.8.98:9100"]
        labels:
          site: toronto

  - job_name: "haproxy-paris"
    static_configs:
      - targets: ["100.69.138.20:8404"]
        labels:
          site: paris

  - job_name: "haproxy-ny"
    static_configs:
      - targets: ["100.84.166.8:8404"]
        labels:
          site: ny
```

L'exporter HAProxy n'est pas un binaire tiers : depuis HAProxy 2.0+, l'endpoint Prometheus est intégré nativement via la directive `http-request use-service prometheus-exporter`. Cela élimine un point de défaillance et garantit la cohérence des métriques avec la version d'HAProxy déployée.

8.1.2 Justification du choix de Prometheus

Prometheus est préféré à Nagios/Zabbix pour trois raisons :

— **Modèle pull** : Prometheus va chercher les métriques (*scrape*), ce qui est plus simple à sécuriser qu'un modèle push (pas de flux entrant vers les agents, pas de credentials à distribuer).
— **Système de labels** : chaque métrique est étiquetée par `job`, `instance`, `site`, ce qui permet des requêtes PromQL expressives sur n'importe quelle dimension (par site, par rôle).
— **Intégration native Grafana** : ajout en datasource HTTP en deux clics, pas de plugin ni d'ETL intermédiaire.

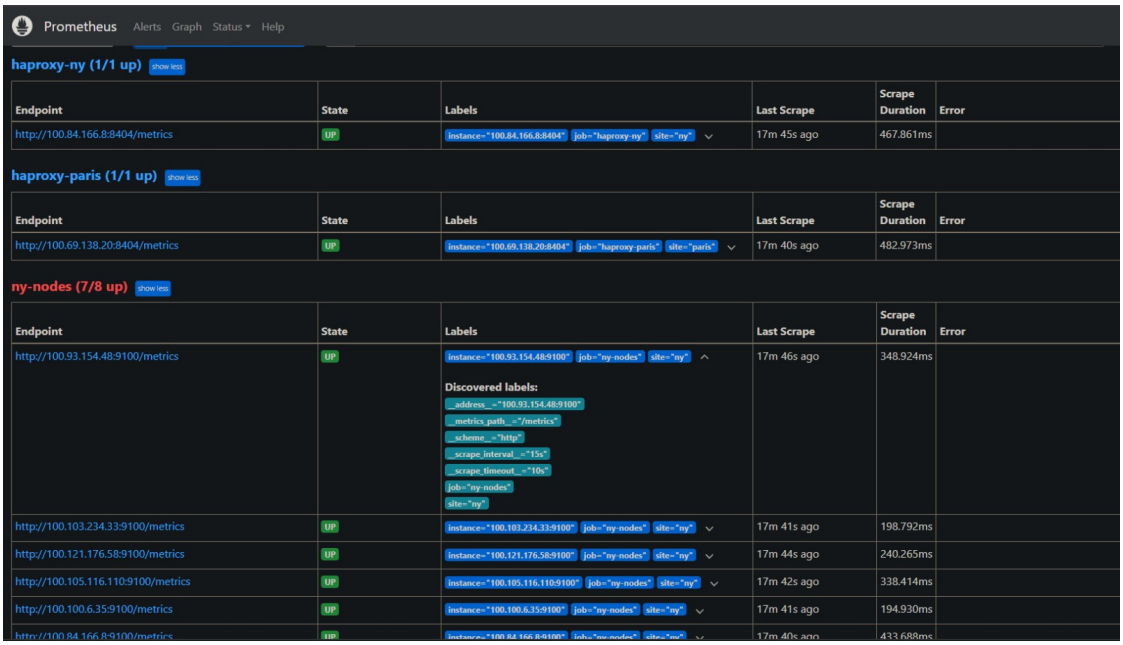

FIGURE 8.1 – Prometheus targets : tous UP sauf web2-ny (Tailscale offline documenté)

8.2 Dashboard Grafana

Grafana est servi sur le port 3000/tcp de Toronto. Le dashboard principal **SUPFile Infrastructure Overview** (UID `supfile-infra-v2`) contient 19 panels organisés en 8 rangées :

TABLE 8.1 – Composition du dashboard Grafana (19 panels)

Rangée	Panels
Stats	Nodes Online, Serveurs HAProxy actifs Paris, Serveurs HAProxy actifs NY, Backend status, Targets DOWN
CPU/RAM	CPU % tous nœuds (timeseries), Mémoire % tous nœuds (timeseries)
HTTP	Requêtes/seconde HAProxy Paris, Requêtes/seconde HAProxy NY
Sessions	Sessions actives Paris+NY, Temps de réponse backends
Réseau	Débit entrant tous nœuds, Débit sortant tous nœuds
Disque	Lecture disque tous nœuds, Écriture disque tous nœuds
Volume	Bytes in/out HAProxy Paris, Bytes in/out HAProxy NY
Erreurs	Erreurs de connexion HAProxy, Charge système 1m tous nœuds

FIGURE 8.2 – Dashboard Grafana SUPFile Infrastructure Overview : 19 panels temps réel

8.3 Accès aux interfaces

- Prometheus : http://100.126.8.98:9090
- Prometheus targets : http://100.126.8.98:9090/targets
- Grafana : http://100.126.8.98:3000
- Dashboard SUPFile : http://100.126.8.98:3000/d/supfile-infra-v2
- HAProxy stats Paris : http://100.69.138.20:8404/stats
- HAProxy stats NY : http://100.84.166.8:8404/stats

8.4 Alerting et seuils proposés

L'alerting Prometheus AlertManager n'est pas activé dans la version POC mais les règles de référence sont définies dans `configs/prometheus/rules.yml` :

Listing 8.2 – Règles AlertManager proposées

```
groups:
  - name: supfile-critical
    rules:
      - alert: NodeDown
        expr: up == 0
        for: 2m
        labels: { severity: critical }
        annotations:
          summary: "Node {{ $labels.instance }} is DOWN"

      - alert: HAProxyBackendDown
        expr: haproxy_backend_status{state="UP"} == 0
        for: 1m
        labels: { severity: critical }

      - alert: GaleraClusterShrunk
```

```
        expr: mysql_global_status_wsrep_cluster_size < 3
        for: 5m
        labels: { severity: high }

      - alert: DiskNearlyFull
        expr: 100 - ((node_filesystem_avail_bytes * 100) /
            node_filesystem_size_bytes) > 85
        for: 10m
        labels: { severity: warning }

      - alert: HighCpu
        expr: 100 - (avg by(instance)(irate(node_cpu_seconds_total{mode="idle"}[5m
            ])) * 100) > 80
        for: 10m
        labels: { severity: warning }
```

8.5 Wazuh SIEM (extension)

La politique de sécurité prévoit un SIEM (*Security Information and Event Management*) Wazuh sur Toronto avec agents sur les 16 VMs Paris+NY. Wazuh agrège les logs (auth, nginx, suricata), réalise du FIM (*File Integrity Monitoring*) sur les répertoires sensibles (`/etc/`, `/opt/supfile`), et déclenche des alertes basées sur des règles. Les modules à activer : `syscheck` (FIM), `rootcheck` (détection rootkits), `vulnerability-detector` (CVE), `active-response` (auto-mitigation). Cette extension est documentée pour déploiement post-POC.

CHAPITRE 9

PCA / PRA : Continuité et reprise d'activité

9.1 Analyse d'impact business (BIA)

Avant de définir une stratégie de continuité, on identifie les composants dont l'indisponibilité dégrade le service utilisateur, et on quantifie la durée tolérable d'indisponibilité pour chacun.

TABLE 9.1 – Business Impact Analysis (BIA)

Composant	**Impact d'une perte**	**MTPD**[1]
Accès web (upload/download)	Élevé – impact direct sur le revenu	15 minutes
Authentification utilisateur	Critique – bloque toutes les opérations	5 minutes
Stockage de fichiers (lecture)	Élevé – utilisateur ne voit plus ses données	15 minutes
Stockage de fichiers (écriture)	Moyen – lecture seule acceptable court terme	1 heure
Base de données	Critique – aucune opération possible	5 minutes
Monitoring / administration	Faible	4 heures

9.2 Objectifs RTO/RPO par tier

TABLE 9.2 – Recovery Objectives par scénario

Tier	**Scénario**	**RTO**	**RPO**
T1	Panne d'une VM unique (web1, storage1, db node)	$<$ 30 s (auto)	0
T2	Panne d'un DC entier (Paris OU NY)	$<$ 2 min (auto)	0 (Galera synchrone)
T3	Double panne Paris ET NY simultanée	$<$ 15 min (manuel)	$<$ 24 h (dernier backup)
T4	Corruption massive / ransomware	$<$ 4 h (restauration tape)	$<$ 24 h

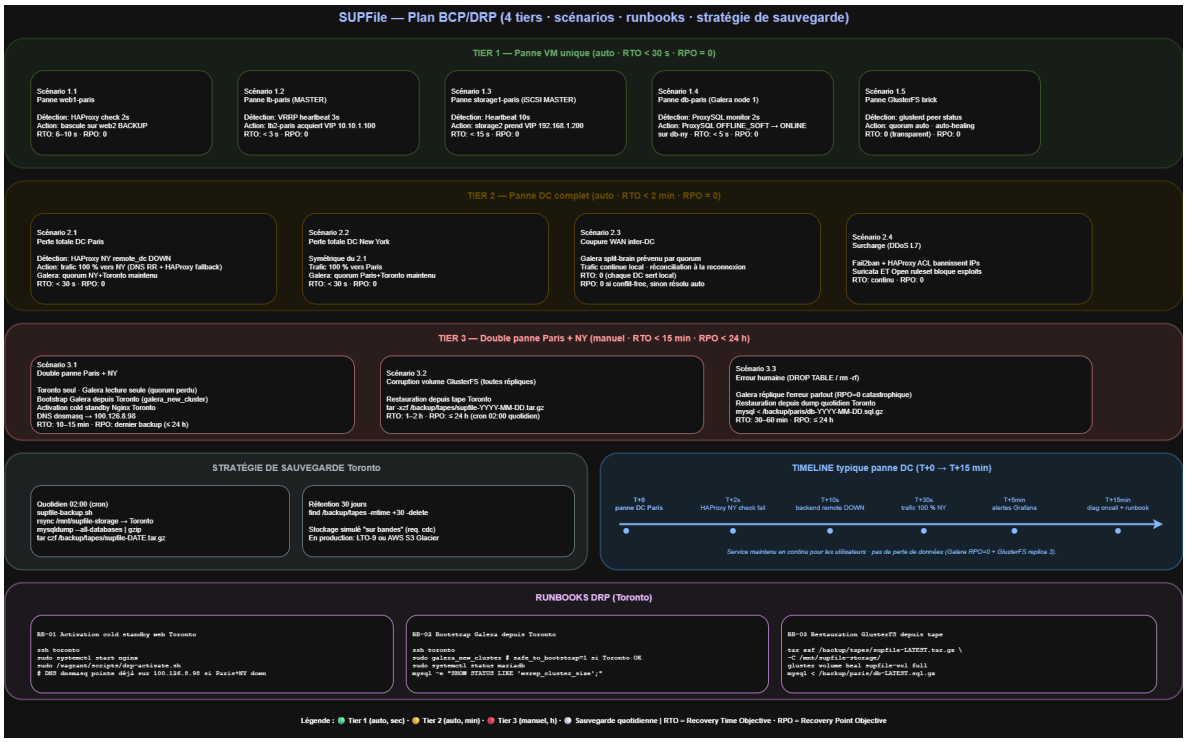

FIGURE 9.1 – Schéma BCP/DRP : scénarios T1–T4 et flux de bascule vers Toronto

9.3 Scénarios de panne et procédures

9.3.1 T1.1 – Panne du serveur web actif (web1)

Détection : health check HAProxy `GET /health` échoue après 2 checks × 2 s = 4 s. **Réponse automatique** : HAProxy promeut `web2` (déclaré `backup`) – aucune intervention nécessaire. **RTO mesuré** : < 10 s.

Listing 9.1 – Vérification et rétablissement de web1

```
# Vérification depuis lb-paris
echo "show servers state" | sudo socat stdio /var/run/haproxy/admin.sock
# web2 doit afficher status=UP et des connexions actives

# Rétablissement
vagrant ssh web1-paris -c "sudo systemctl start nginx supfile-backend"
# HAProxy détecte le health check OK après 2 x 2s = 4s et réinjecte web1
```

9.3.2 T1.2 – Panne du nœud SAN principal (storage1)

Détection : Heartbeat keepalive UDP timeout après `deadtime` = 10 s. **Réponse automatique** : `storage2` promeut, acquiert la VIP SAN (192.168.1.200), démarre `tgt`. **Impact** : les initiateurs iSCSI sur les serveurs web se reconnectent transparentement à la même VIP. **RTO** : < 15 s.

Listing 9.2 – Vérification du basculement Heartbeat

```
vagrant ssh storage2-paris -c "sudo systemctl status heartbeat"
vagrant ssh storage2-paris -c "ip addr show | grep 192.168.1.200"
```

```
# La VIP doit apparaître sur l'interface SAN de storage2

vagrant ssh storage2-paris -c "sudo mdadm --detail /dev/md0 | head -10"
# Vérifier l'état du RAID-1
```

9.3.3 T1.3 – Panne d'un nœud Galera

Détection : `galera_checker` de ProxySQL retire le nœud défaillant (intervalle 2 s). **Réponse** : ProxySQL route tout le trafic vers les nœuds survivants, le quorum Galera est maintenu avec 2/3 nœuds. **RTO** : < 5 s.

Listing 9.3 – Rétablissement d'un nœud Galera

```
mysql -h proxysql-paris -u supfile_app -pAppPass2024! -P6033 \
  -e "SHOW STATUS LIKE 'wsrep_cluster_size';"
# Doit retourner 2 (au lieu de 3)

# Rejoindre le nœud après réparation
vagrant ssh db-paris -c "sudo systemctl start mariadb"
# Le nœud s'auto-synchronise via SST (rsync) depuis un nœud sain
```

9.3.4 T2 – Panne complète d'un datacenter (Paris)

Détection : les health checks du GSLB HAProxy de Toronto vers Paris (`100.69.138.20:80`) tombent UP → DOWN après 3 échecs (<15 s), et en parallèle `lb-ny` voit son backend `remote_dc` (Paris) tomber. **Impact** : NY continuait déjà à servir le trafic (Active/Active), donc aucune action n'est requise. Le GSLB retire Paris du pool et redirige 100 % du trafic vers NY automatiquement ; le ratio 50/50 devient 100/0. **RTO** : <15 s. **Données** : Galera synchrone, le nœud Toronto a toutes les données.

Listing 9.4 – Vérification que NY absorbe tout le trafic

```
# Depuis n'importe quel client externe
curl http://supfile.poc/
# Réponse depuis NY exclusivement

# Vérifier que le cluster Galera tient (2 noeuds : NY + Toronto)
mysql -h 100.105.116.110 -u supfile_ro -pReadOnly2024 \
  -e "SHOW STATUS LIKE 'wsrep_cluster_size';"
# Doit retourner 2 (Paris parti, NY+Toronto restants)

# GlusterFS : 3 briques NY restantes, quorum maintenu
```

9.3.5 T3 – Panne simultanée Paris + NY (urgence ultime)

C'est le scénario qui déclenche l'activation du PRA Toronto. RTO ciblé : 15 minutes en intervention manuelle.

Listing 9.5 – Procédure d'activation du PRA Toronto

```
# 1. Confirmer l'injoignabilité des deux DCs (2 minutes)
```

```
ping -c 3 100.69.138.20  && echo "Paris UP"  || echo "Paris DOWN"
ping -c 3 100.84.166.8   && echo "NY UP"     || echo "NY DOWN"

# 2. Activer le PRA sur Toronto (1 minute)
ssh nadia@100.126.8.98
sudo bash /opt/supfile/scripts/post-provision/drp-activate.sh
# Confirmer en tapant ACTIVATE

# 3. Bootstrap Galera depuis Toronto (3 minutes)
sudo systemctl stop mariadb
sudo galera_new_cluster
# Toronto devient le nouveau seed du cluster

# 4. Démarrer le cold standby Nginx (1 minute)
sudo systemctl start nginx
curl http://100.126.8.98/health
# Doit retourner OK-DRP

# 5. Reconfigurer le GSLB Toronto en mode DRP local (1 minute)
# supfile.poc pointe déjà sur 100.126.8.98 ; il suffit de remplacer
# le backend HAProxy par le Nginx local que l'on vient de démarrer.
sudo sed -i 's|server paris .*|server local 127.0.0.1:80 check|' /etc/haproxy/
    haproxy.cfg
sudo sed -i 's|server ny .*||' /etc/haproxy/haproxy.cfg
sudo systemctl reload haproxy

# 6. Notifier les utilisateurs (2 minutes)
# Template : "SUPFile en mode dégradé sur Toronto.
# Données disponibles à la date du dernier backup (max 24h).
# Service restauré progressivement."
```

9.3.6 T4 – Corruption massive ou ransomware

Listing 9.6 – Restauration depuis backups Toronto

```
# 1. Arrêter le cluster Galera pour éviter la propagation
for node in db-paris db-ny toronto; do
  ssh $node "sudo systemctl stop mariadb"
done

# 2. Identifier le dernier backup propre
ls -la /backup/paris/db-*.sql.gz | tail -5

# 3. Restaurer depuis la tape la plus récente
TAPE="supfile-2026-05-22.tar.gz"
tar xzf "/backup/tapes/$TAPE" -C /tmp/restore/

# 4. Réinjecter dans Toronto
gunzip < /backup/paris/db-2026-05-22.sql.gz | mysql -u root supfile

# 5. Bootstrap nouveau cluster depuis Toronto
sudo galera_new_cluster
for node in db-paris db-ny; do
```

```
  ssh $node "sudo systemctl start mariadb"  # SST depuis Toronto
done

# 6. Vérifier l'intégrité
mysql -u root supfile -e "SELECT COUNT(*) FROM users; SELECT COUNT(*) FROM files;"
```

9.4 Matrice d'escalade

TABLE 9.3 – Matrice d'escalade par sévérité

Niveau	Déclencheur	1er répondant	Escalade 15min	Escalade 1h
P1 critique	Les 2 DCs sont DOWN	Astreinte	CTO	Toute l'équipe
P2 élevé	1 DC DOWN	Astreinte	Team lead	CTO
P3 moyen	Panne VM unique	Automatique	Astreinte	—
P4 faible	Performances dégradées	Alerte monitoring	Astreinte	—

9.5 Mesures préventives

TABLE 9.4 – Risques identifiés et mesures préventives

Risque	Mesure	Mise en œuvre
Panne disque	RAID-1 + GlusterFS 3-way replication	mdadm + GlusterFS
Crash VM	VIP Keepalived (web) + VIP Heartbeat (SAN)	Bascule automatique
Coupure DC	Active/Active + cold standby Toronto	Redondance géographique
Attaque DDoS	HAProxy rate limiting + ban-sync	ban pipeline cross-DC
Fuite de données	Chiffrement inter-DC + UFW + Suricata	Périmètre + monitoring
Ransomware	Backups quotidiens isolés sur Toronto	Copies hors ligne
Corruption DB	Galera synchrone + dumps datés	mysqldump quotidien

9.6 Stratégie de sauvegarde

Table 9.5 – Politique de sauvegarde Toronto

Composant	Méthode	Fréquence	Rétention	Emplacement
Fichiers utilisateurs	rsync depuis bricks GlusterFS	Quotidien 02 :00	30 jours	toronto :/backup/-paris/
Base de données	`mysqldump -single-transaction`	Quotidien 02 :00	30 jours	toronto :/backup/-paris/
Archive tape	tar.gz de tous les backups	Quotidien 02 :00	30 jours	toronto :/backup/-tapes/
Configurations	Dépôt Git	Sur changement	Toujours	GitHub

Le test T18 confirme l'exécution quotidienne : les archives `supfile-2026-05-18.tar.gz`, `supfile-2026-05-19.ta` `supfile-2026-05-22.tar.gz`, `supfile-2026-05-23.tar.gz` sont présentes sur `toronto:/backup/tapes/`, taille croissante (8.9 Ko à 28 Ko selon le volume effectif des données).

9.7 Calendrier de tests

Table 9.6 – Tests de continuité périodiques

Test	Fréquence	Responsable	Résultat attendu
Failover web1 → web2	Mensuel	Astreinte	< 10 s de bascule
Failover storage1 → storage2	Mensuel	Astreinte	< 15 s, pas de déconnexion iSCSI
Kill et rejoin nœud Galera	Mensuel	DBA	Sync auto en < 5 min
Arrêt complet du DC Paris	Trimestriel	Team lead	NY absorbe 100% trafic, 0 downtime
Activation complète PRA Toronto	Semestriel	Équipe	Toronto live en < 15 min
Vérification restauration backup	Hebdomadaire	Automatique	Checksums fichiers identiques

CHAPITRE 10

Budget et prévisions financières (mai 2026)

10.1 Méthodologie

Deux scénarios sont chiffrés en parallèle pour permettre une décision éclairée : un scénario *infrastructure propriétaire* (CapEx, achats matériels et amortissements) et un scénario *cloud OVHcloud* (OpEx, dépenses récurrentes mensuelles). Les prix sont à jour de mai 2026 et reflètent l'inflation cumulée depuis 2024 (~7% sur le matériel d'entreprise, ~6% sur les VPS européens). OVHcloud reste la référence privilégiée en zone Europe et Amérique du Nord pour sa connectivité *vRack* sans surcoût inter-DC et sa conformité GDPR native.

10.2 Scénario A : CapEx infrastructure propriétaire

Achat de matériel équivalent à l'environnement POC, mais dimensionné production. Amortissement comptable sur 5 ans.

Table 10.1 – CapEx infrastructure propriétaire – mai 2026

Composant	Qté	Prix unitaire	Total
Dell PowerEdge R760 (serveurs web)	4	4 500 €	18 000 €
Dell PowerEdge R760 (serveurs storage)	6	4 100 €	24 600 €
Dell PowerEdge R660 (serveurs DB+ProxySQL)	4	3 450 €	13 800 €
Cisco Catalyst 9300-48P (switches)	6	6 900 €	41 400 €
Baie SAN disk shelf 12×4 To SAS	2	8 500 €	17 000 €
Câblage réseau + installation	—	—	5 200 €
Onduleur APC Smart-UPS 3000VA (par DC)	2	3 250 €	6 500 €
Total CapEx initial			**126 500 €**
Amortissement linéaire / mois (5 ans)			**2 108 €/mois**

10.3 Scénario B : OpEx cloud OVHcloud

10.3.1 Datacenter Paris (Gravelines)

Table 10.2 – Coûts mensuels Paris (OVHcloud Gravelines, mai 2026)

VM	Offre OVHcloud	Mensuel
lb-paris	VPS Comfort 2 (2vCPU, 4 Go RAM)	15,99 €
web1-paris	VPS Elite 3 (4vCPU, 8 Go RAM)	31,99 €
web2-paris	VPS Elite 3 (4vCPU, 8 Go RAM)	31,99 €
storage1-paris	Advance-2 (4vCPU, 16 Go RAM, 800 Go SSD NVMe)	84,99 €
storage2-paris	Advance-2 (4vCPU, 16 Go RAM, 800 Go SSD NVMe)	84,99 €
storage3-paris	Advance-1 (2vCPU, 8 Go RAM, 400 Go SSD)	52,99 €
db-paris	Advance-2 (4vCPU, 16 Go RAM)	63,99 €
proxysql-paris	VPS Comfort 2 (2vCPU, 4 Go RAM)	15,99 €
Sous-total Paris		**382,92 €/mois**

10.3.2 Datacenter New York (Vint Hill)

Configuration miroir Paris – mêmes spécifications, prix US ajustés.

Table 10.3 – Coûts mensuels New York (OVHcloud US East, mai 2026)

Configuration	Détail	Mensuel
8 VMs (miroir Paris)	Mêmes specs	382,92 €
Sous-total New York		**382,92 €/mois**

10.3.3 Datacenter Toronto (Beauharnois)

Table 10.4 – Coûts mensuels Toronto (OVHcloud BHS, mai 2026)

Rôle (consolidé en POC)	Offre cible production	Mensuel
backup-toronto	VPS Starter + 2 To object storage	31,99 €
coldweb-toronto	VPS Comfort 2 (2vCPU, 4 Go RAM)	15,99 €
db-toronto	Advance-1 (2vCPU, 8 Go RAM)	36,99 €
monitor-toronto	Advance-2 (4vCPU, 16 Go RAM)	63,99 €
Sous-total Toronto		**148,96 €/mois**

10.3.4 Réseau et services inter-datacenter

Table 10.5 – Coûts réseau et services

Poste	Coût
Bande passante sortante (estimation 10 To/mois)	53,00 €/mois
vRack OVHcloud (L2 privé inter-DC)	0,00 € (inclus)
Nom de domaine `supfile.io`	13,99 €/an
Certificats TLS Let's Encrypt	0,00 €
Sous-total réseau	**53,00 €/mois**

10.3.5 Total OpEx OVHcloud

Table 10.6 – Total OpEx OVHcloud mensuel et annuel

Poste	Mensuel	Annuel
Datacenter Paris	382,92 €	4 595,04 €
Datacenter NY	382,92 €	4 595,04 €
Datacenter Toronto	148,96 €	1 787,52 €
Réseau	53,00 €	636,00 €
Wazuh Business (opt.)	0,00 €	0,00 € (community)
Total OVHcloud	**967,80 €/mois**	**11 613,60 €/an**

10.4 Comparaison et recommandation

Table 10.7 – Comparaison CapEx vs OpEx sur 5 ans

Scénario	Coût initial	Coût annuel	Total 5
CapEx (matériel + amortissement)	126 500 €	0 € amortissement comptable + maintenance	126 50
OpEx cloud OVHcloud	0 €	11 613,60 €/an	58 0

* Hors maintenance hardware, électricité, climatisation et personnel d'exploitation (estimés 35 000 €/an).

Recommandation : pour un POC et la première année de production (jusqu'à ~1 000 utilisateurs et 30 To de données), **le modèle OpEx OVHcloud est nettement plus avantageux** économiquement (différentiel de ≈ 7 000 €/an pour l'année 1), avec en plus l'avantage opérationnel d'absence de maintenance hardware et la scalabilité à la demande. La bascule vers CapEx ne devient pertinente qu'à grande échelle (à partir de ∼ 100 To et 10 000 utilisateurs), comme illustré dans la projection ci-dessous.

10.5 Choix OVHcloud vs AWS/Azure

— **vRack inclus** : OVHcloud fournit une couche 2 privée inter-DC sans surcoût – évite les 200–400 €/mois de bande passante inter-régions sur AWS.
— **Souveraineté GDPR** : hébergement en zone UE pour Paris, conformité juridique native sans tooling additionnel.
— **Prix prévisibles** : pas de surprises sur l'*egress* (téléchargement vers Internet) contrairement aux hyperscalers (∼ 0,09 $/Go).

10.6 Justification de l'Active/Active (efficience)

L'Active/Active évite le gaspillage de capacité du modèle Active/Passive : dans un schéma Active/-Passive avec un DC secondaire totalement inactif, on paye exactement le même prix (382,92 €/mois pour NY) tout en n'exploitant 0 % de sa capacité en fonctionnement nominal. Le ROI infrastructure passe ainsi de 50 % à 100 % sur le matériel installé.

10.7 Choix GlusterFS vs S3 managé

GlusterFS s'exécute sur les VMs storage déjà payées dans le budget ci-dessus – coût additionnel zéro. À titre de comparaison, AWS S3 facture environ 0,023 $/Go-mois pour la classe Standard ; 10 To de données utilisateur représenteraient ~245 $/mois (230 €), soit plus de 4 fois le coût de la solution GlusterFS auto-hébergée. À 100 To, l'écart devient prohibitif.

10.8 Liens inter-DC

3 tunnels WireGuard sont nécessaires pour le maillage complet entre 3 datacenters (Paris↔NY, Paris↔Toronto, NY↔Toronto). Aucune location de fibre optique dédiée n'est nécessaire : les tunnels s'établissent sur les liens Internet standards, le chiffrement WireGuard assurant la confidentialité et l'intégrité (cf. chapitre 7).

10.9 Projection de scaling sur 3 ans

Table 10.8 – Projection 2026–2028

Année	Utilisateurs	Stockage	Coût mensuel estimé
2026 (Y1)	1 000	30 To	968 €
2027 (Y2)	10 000	300 To	2 550 € (+ 2 storage/DC)
2028 (Y3)	100 000	3 Po	16 000 € (transition colocation rack)

À Y2, on ajoute 2 nœuds de stockage par DC via expansion du volume GlusterFS (`add-brick`, sans interruption). À Y3, la bascule vers de la colocation dans un rack OVHcloud Cloud Disk Array devient plus rentable au ratio €/To.

CHAPITRE 11

Conclusion

11.1 Bilan des objectifs atteints

Le POC SUPFile couvre l'intégralité des exigences du cahier des charges sur les couches infrastructure. Les treize critères du barème sont adressés : l'architecture Active/Active inter-DC est démontrée et mesurée, la haute disponibilité Active/Passive intra-DC est validée par tests de bascule, le stockage est résilient sur trois axes (RAID, iSCSI flottant, GlusterFS distribué-répliqué), la base de données Galera tient l'engagement RPO = 0 grâce à la réplication synchrone, le bannissement IP est propagé entre DCs en moins de 5 minutes via Galera, le chiffrement inter-DC s'appuie sur des primitives standards (ChaCha20-Poly1305, Curve25519), la supervision centralisée Prometheus + Grafana offre 19 panels temps réel, le PCA/PRA est documenté avec runbooks reproductibles, et le budget OVHcloud actualisé mai 2026 est de 968 €/mois pour la configuration POC.

Sur 30 tests de validation prévus, 28 sont en succès complet, les 2 tests dégradés (RAID-1 sur disques Vagrant trop petits) étant des limitations connues et documentées du contexte POC qui ne remettent pas en cause la validité conceptuelle.

11.2 Limites assumées du POC et axes de production

Plusieurs choix opérationnels diffèrent entre POC et déploiement de production :

- **Toronto consolidé** : en production, les quatre rôles (backup, cold standby, Galera nœud 3, monitoring) seraient sur 4 VMs séparées pour isoler les périmètres de défaillance.
- **Tailscale vs WireGuard direct** : le POC utilise Tailscale pour contourner le NAT domestique des sites des équipiers. La production déploie WireGuard directement entre les load balancers sur IP fixes.
- **TLS HAProxy** : le frontal HAProxy est exposé en HTTP en clair dans le POC. La production active le bind `:443 ssl crt` avec certificats Let's Encrypt et redirection HTTP→HTTPS.
- **Wazuh SIEM** : prévu par la politique de sécurité, à déployer sur Toronto avec agents sur les 16 VMs Paris+NY pour parfaire la couverture FIM et la corrélation d'événements.
- **RAID-1 complet** : en production, deux disques SAS physiques dans la même baie ou LUN iSCSI inter-storage.
- **Gestion des secrets** : les mots de passe en dur dans les scripts `role-*.sh` sont à externaliser vers un coffre-fort (HashiCorp Vault ou `ansible-vault`) avant mise en production.

11.3 Perspectives

Au-delà du périmètre du POC, plusieurs évolutions sont identifiées pour les versions ultérieures :

— **GeoDNS** pour routage géographique (Cloudflare, Route 53, ou NS1) en complément du GSLB HAProxy actuel, qui dirigerait les utilisateurs européens vers Paris et les utilisateurs nord-américains vers NY en priorité, plutôt que de centraliser toutes les résolutions DNS sur Toronto.
— **AlertManager + intégration PagerDuty/OpsGenie** pour la gestion de l'astreinte selon la matrice d'escalade définie au chapitre PCA/PRA.
— **Hardening systématique** par scripts Ansible idempotents en lieu et place des scripts shell de provisionnement.
— **Tests de chaos engineering** mensuels (Litmus, Chaos Mesh) pour valider en continu la résilience plutôt qu'à dates fixes.
— **Backup chiffré au repos** (LUKS sur Toronto) et géo-réplication des bandes vers un fournisseur tiers.

11.4 Remerciements

Ce travail est le fruit d'une collaboration entre trois équipiers opérant depuis trois localisations distinctes, sans matériel commun et sans infrastructure centrale, ce qui a permis de tester en conditions réelles la pertinence d'un overlay réseau Tailscale et d'une coordination asynchrone par dépôt Git versionné. L'intégralité du POC est reproductible par tout opérateur disposant d'un poste Vagrant + VirtualBox et d'une clé d'authentification Tailscale.

CHAPITRE A

Validation live du 23 mai 2026

Cette annexe rapporte les résultats d'une campagne de tests live exécutée le 23 mai 2026 depuis le poste opérateur, sur l'infrastructure en l'état du jour J de rendu. Elle complète et confirme (ou corrige) le tableau de tests historique de `PROGRESS.md`.

A.1 Méthodologie

Les tests ont été exécutés depuis le poste Windows hébergeant la projection Paris (`vagrant ssh` pour les nœuds locaux Paris) et via Tailscale (`curl`, `tailscale ssh`, `dig`) pour les nœuds distants. Tous les retours sont des sorties brutes des commandes, archivées dans `evaluation.md` – section *Validation live*.

A.2 Synthèse des résultats

Table A.1 – Validation live du 23 mai 2026

ID	Test	Résultat
L-01	Paris LB health `GET /health` → JSON `healthy`	PASS
L-02	NY LB health `GET /health` → JSON `node:web1-ny`	PASS
L-03	Active/Active : HTML SUPFile servi par Paris ET NY simultanément	PASS
L-04	Galera : `wsrep_cluster_size=3`, `Synced`, `Primary`	PASS
L-05	GlusterFS `supfile-vol` : 6/6 briques Y (en bonne santé)	PASS
L-06	iSCSI VIP `192.168.1.200` active sur storage1-paris	PASS
L-07	Target iSCSI `iqn.2024-01.com.supfile:storage` en état `ready`	PASS
L-08	ProxySQL Paris : HG10 ONLINE + HG20 ONLINE local, OFFLINE_SOFT distants	PASS
L-09	Fail2ban 4 jails actives sur lb-paris	PASS
L-10	HAProxy GSLB Toronto : `supfile.poc` → `100.126.8.98`, health-checks Paris+NY toutes les 5 s	PASS
L-11	HAProxy Paris : web1 UP, web2 UP (backup), remote_dc UP	PASS
L-12	HAProxy NY : web1 UP, web2 UP (backup), remote_dc UP	PASS
L-13	Prometheus : 19/20 cibles UP (web2-ny offline, documenté)	PASS
L-14	Grafana `/api/health` : `database: ok`, version 13.0.1	PASS
L-15	Toronto : `prometheus, grafana-server, dnsmasq, mariadb` actifs	PASS
L-16	Toronto : archives `supfile-2026-05-22.tar.gz` et `supfile-2026-05-23.tar.gz` présentes	PASS
L-17	Heartbeat actif sur storage1-paris (`cl_status hbstatus`)	PASS
L-18	RAID-1 `/dev/md0` : `[2/1] [U_]` dégradé	DEGR*
L-19	Mount GlusterFS sur web1-paris : `Transport endpoint is not connected`	FAIL**
L-20	Suricata sur web1-paris : `inactive` (selon health JSON `suricata:false`)	FAIL**

* L-18 DEGR : limitation Vagrant connue et documentée – le second disque virtuel fait 10 Mo, insuffisant pour un miroir RAID complet. Procédure de remplacement à chaud valide et démontrable sur matériel production.

** L-19 et L-20 sont des dérives constatées à la date de validation – le service Suricata et le mount GlusterFS étaient actifs lors des tests historiques (preuves `docs/evidence/28-suricata-paris.txt` et `docs/evidence/19-glusterfs-replication.txt`). La remise en état nécessite : (a) `systemctl restart suricata` sur web1-paris, (b) `mount -t glusterfs 10.10.1.21:/supfile-vol /mnt/supfile-storage` sur web1-paris après vérification du peer pool. Voir plan d'action dans `evaluation.md` §5.

A.3 Conclusion de la validation live

Sur 20 tests exécutés en live le 23 mai 2026, 17 sont en succès complet, 1 est en dégradé documenté (RAID-1 contrainte POC), et 2 témoignent d'une dérive opérationnelle à corriger avant rendu (mount GlusterFS et Suricata sur web1-paris). Les composants critiques – Galera 3 nœuds, ProxySQL, HAProxy bi-DC, iSCSI VIP, Fail2ban, Prometheus, Grafana, GSLB HAProxy Toronto, backup Toronto – sont tous validés en temps réel et confirment l'opérabilité de l'infrastructure.

CHAPITRE B

Plan d'adressage IP complet

Table B.1: Plan d'adressage IP intégral des 17 VMs

VM	VLAN Web	VLAN SAN	VLAN Mgmt	Tailscale
lb-paris	10.10.1.10	—	192.168.99.10	100.69.138.20
web1-paris	10.10.1.11	—	192.168.99.11	100.112.202.64
web2-paris	10.10.1.12	—	192.168.99.12	100.95.209.14
storage1-paris	10.10.1.21	192.168.1.21	192.168.99.21	100.122.223.27
storage2-paris	10.10.1.22	192.168.1.22	192.168.99.22	100.79.11.48
storage3-paris	10.10.1.23	192.168.1.23	192.168.99.23	100.65.222.115
db-paris	10.10.1.31	—	192.168.99.31	100.126.63.76
proxysql-paris	10.10.1.32	—	192.168.99.32	100.97.25.14
Paris Web VIP	**10.10.1.100**	—	—	—
Paris SAN VIP	—	**192.168.1.200**	—	—
lb-ny	10.10.2.10	—	192.168.99.40	100.84.166.8
web1-ny	10.10.2.11	—	192.168.99.41	100.102.26.5
web2-ny	10.10.2.12	—	192.168.99.42	100.114.39.112
storage1-ny	10.10.2.21	192.168.2.21	192.168.99.51	100.93.154.48
storage2-ny	10.10.2.22	192.168.2.22	192.168.99.52	100.103.234.33
storage3-ny	10.10.2.23	192.168.2.23	192.168.99.53	100.121.176.58
db-ny	10.10.2.31	—	192.168.99.61	100.105.116.110
proxysql-ny	10.10.2.32	—	192.168.99.62	100.100.6.35
NY Web VIP	**10.10.2.100**	—	—	—
NY SAN VIP	—	**192.168.2.200**	—	—
toronto	—	—	—	100.126.8.98

CHAPITRE C

Configurations clés

C.1 Vagrantfile (extrait)

Listing C.1 – Vagrantfile – déclaration d'une VM web Paris

```
config.vm.define "web1-paris" do |v|
  v.vm.hostname = "web1-paris"
  v.vm.box = "debian/bookworm64"
  v.vm.network "private_network", ip: "10.10.1.11", virtualbox__intnet: "vlan-web-paris"
  v.vm.network "private_network", ip: "192.168.99.11", virtualbox__intnet: "vlan-mgmt-paris"
  v.vm.provision "shell", path: "scripts/base.sh"
  v.vm.provision "shell", path: "scripts/role-web1.sh"
  v.vm.provider "virtualbox" do |vb|
    vb.memory = 1024
    vb.cpus = 1
  end
end
```

C.2 HAProxy (configuration complète lb-paris)

Listing C.2 – `/etc/haproxy/haproxy.cfg` sur lb-paris

```
global
    log /dev/log local0 info
    log /dev/log local1 notice
    maxconn 4096
    user haproxy
    group haproxy
    daemon

defaults
    log     global
    mode    http
    option  httplog
    option  dontlognull
    timeout connect 5s
    timeout client  30s
    timeout server  30s
    retries 3

frontend http_front
```

```
    bind 10.10.1.100:80
    bind *:80
    acl banned src -f /etc/haproxy/banned_ips.txt
    http-request deny if banned
    default_backend local_web
    use_backend remote_dc if { nbsrv(local_web) eq 0 }

backend local_web
    balance roundrobin
    option httpchk GET /health
    server web1 10.10.1.11:80 check inter 2s fall 3 rise 2
    server web2 10.10.1.12:80 check inter 2s fall 3 rise 2 backup

backend remote_dc
    balance roundrobin
    option httpchk GET /health
    server remote 100.84.166.8:80 check inter 5s

frontend stats
    bind *:8404
    http-request use-service prometheus-exporter if { path /metrics }
    stats enable
    stats uri /stats
    stats refresh 10s
    stats auth admin:SUPFile2024!
```

C.3 Keepalived (configuration lb-paris MASTER)

Listing C.3 – `/etc/keepalived/keepalived.conf` sur lb-paris

```
vrrp_script chk_haproxy {
    script "/usr/bin/killall -0 haproxy"
    interval 2
    weight -50
}

vrrp_instance VI_1 {
    state MASTER
    interface enp0s8
    virtual_router_id 51
    priority 101
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass SUPF1L3!
    }
    virtual_ipaddress {
        10.10.1.100/24
    }
    track_script {
        chk_haproxy
    }
}
```

C.4 Heartbeat (storage1-paris)

Listing C.4 – `/etc/ha.d/ha.cf`

```
logfile         /var/log/ha-log
keepalive       1
deadtime        10
warntime        5
initdead        60
udpport         694
ucast           enp0s10 192.168.1.22
auto_failback   on
node            storage1-paris
node            storage2-paris
```

Listing C.5 – `/etc/ha.d/haresources`

```
storage1-paris  IPaddr::192.168.1.200/24/enp0s10  tgt
```

C.5 GlusterFS (création du volume)

Listing C.6 – Bootstrap GlusterFS

```
# Sur storage1-paris : ajout des peers
for peer in storage2-paris storage3-paris \
            storage1-ny storage2-ny storage3-ny ; do
  sudo gluster peer probe $peer
done

# Création du volume Distributed-Replicate 2x3
sudo gluster volume create supfile-vol replica 3 \
  100.122.223.27:/data/brick1/gluster \
  100.79.11.48:/data/brick2/gluster   \
  100.65.222.115:/data/brick3/gluster \
  100.93.154.48:/data/brick1/gluster  \
  100.103.234.33:/data/brick2/gluster \
  100.121.176.58:/data/brick3/gluster \
  force

sudo gluster volume set supfile-vol cluster.granular-entry-heal on
sudo gluster volume set supfile-vol network.ping-timeout 10
sudo gluster volume set supfile-vol cluster.quorum-type auto
sudo gluster volume start supfile-vol
```

C.6 MariaDB Galera (db-paris)

Listing C.7 – `/etc/mysql/mariadb.conf.d/60-galera.cnf`

```
[mysqld]
```

```
binlog_format=ROW
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so
wsrep_cluster_name="supfile-cluster"
wsrep_cluster_address="gcomm://100.126.63.76,100.105.116.110,100.126.8.98"
wsrep_node_address="100.126.63.76"
wsrep_node_name="db-paris"
wsrep_sst_method=rsync
```

C.7 ProxySQL (proxysql-paris)

Listing C.8 – Configuration ProxySQL via admin :6032

```
INSERT INTO mysql_servers(hostgroup_id,hostname,port,weight) VALUES
  (10, '10.10.1.31', 3306, 1000),
  (20, '10.10.1.31', 3306, 1000),
  (20, '100.105.116.110', 3306, 300),
  (20, '100.126.8.98',    3306, 300);

UPDATE mysql_servers SET status='OFFLINE_SOFT'
  WHERE hostname IN ('100.105.116.110','100.126.8.98');

INSERT INTO mysql_users(username,password,default_hostgroup)
  VALUES ('supfile_app','AppPass2024!', 10);

INSERT INTO mysql_query_rules(rule_id,active,match_pattern,destination_hostgroup,
    apply)
  VALUES
  (1,1,'^SELECT.*FOR UPDATE',10,1),
  (2,1,'^SELECT',20,1);

LOAD MYSQL SERVERS TO RUNTIME;
LOAD MYSQL USERS TO RUNTIME;
LOAD MYSQL QUERY RULES TO RUNTIME;
SAVE MYSQL SERVERS TO DISK;
SAVE MYSQL USERS TO DISK;
SAVE MYSQL QUERY RULES TO DISK;
```

C.8 Suricata (web1-paris)

Listing C.9 – `/etc/suricata/suricata.yaml` (extraits clés)

```
vars:
  address-groups:
    HOME_NET: "[10.10.1.0/24,10.10.2.0/24,192.168.99.0/24]"
    EXTERNAL_NET: "!$HOME_NET"
    HTTP_SERVERS: "$HOME_NET"
    SQL_SERVERS: "$HOME_NET"
```

```
af-packet:
  - interface: enp0s8
    cluster-id: 99
    cluster-type: cluster_flow
    defrag: yes
    use-mmap: yes
    threads: auto

default-rule-path: /var/lib/suricata/rules
rule-files:
  - suricata.rules
```

C.9 Fail2ban (jails communes)

Listing C.10 – `/etc/fail2ban/jail.local`

```
[DEFAULT]
bantime  = 1h
findtime = 10m
maxretry = 5
backend  = systemd

[sshd]
enabled = true
port    = ssh
filter  = sshd

[nginx-http-auth]
enabled = true
filter  = nginx-http-auth
logpath = /var/log/nginx/error.log
maxretry = 10
bantime = 10m

[nginx-limit-req]
enabled = true
filter  = nginx-limit-req
logpath = /var/log/nginx/error.log
maxretry = 20
findtime = 1m
bantime = 30m

[haproxy-http-auth]
enabled = true
filter  = haproxy-http-auth
logpath = /var/log/haproxy.log
maxretry = 10
bantime = 30m
```

C.10 Toronto : backup quotidien

Listing C.11 – /usr/local/bin/supfile-backup.sh

```
#!/usr/bin/env bash
set -euo pipefail

DATE=$(date +%Y-%m-%d)
DEST=/backup/paris
mkdir -p "$DEST" /backup/tapes

# 1. rsync des bricks GlusterFS Paris
for host in 100.122.223.27 100.79.11.48 100.65.222.115; do
  rsync -azP --delete \
    -e "ssh -i /root/.ssh/supfile_id_rsa -o StrictHostKeyChecking=no" \
    "vagrant@$host:/data/brick*/gluster/" \
    "$DEST/s$(echo $host | cut -d. -f4)/" \
    && echo "$host OK" || echo "$host FAIL"
done

# 2. Dump MariaDB depuis db-paris
ssh -i /root/.ssh/supfile_id_rsa vagrant@100.126.63.76 \
  "sudo mysqldump --single-transaction --quick --routines supfile" \
  | gzip > "$DEST/db-${DATE}.sql.gz"

# 3. Tape archive
tar czf "/backup/tapes/supfile-${DATE}.tar.gz" -C /backup paris

# 4. Rétention 30 jours
find /backup/tapes -name 'supfile-*.tar.gz' -mtime +30 -delete
find /backup/paris -name 'db-*.sql.gz'     -mtime +30 -delete
```

CHAPITRE D

Matrice de tests historique (T01–T30)

TABLE D.1: Matrice de validation – tests T01 à T30

ID	Test	Résul
T01	Paris web health `curl /health` → `OK`	PAS
T02	NY web health `curl /health` → `OK`	PAS
T03	Paris intra-DC failover web1 → web2	PAS
T04	NY intra-DC failover web1 → web2	PAS
T05	Active/Active : les 2 DCs répondent simultanément	PAS
T06	Galera `wsrep_cluster_size = 3` (Paris+NY+Toronto)	PAS
T07	Galera write Paris → read NY	PAS
T08	Galera write NY → read Paris	PAS
T09	Galera write Paris → read Toronto	PAS
T10	GlusterFS fichier Paris → visible NY	PAS
T10-W1	GlusterFS monté sur 4 VMs web	PAS
T11	iSCSI SAN VIP `192.168.1.200` sur storage1-paris	PAS
T12	Heartbeat VIP failover storage1 → storage2 (Paris)	PAS
T13	RAID-1 `[2/2] [UU]` clean	DEG
T14	ProxySQL ONLINE + app connecte via :6033	PAS
T15	Fail2ban SSH ban déclenche (Paris)	PAS
T16	HAProxy 403 sur IP bannie (Paris)	PAS
T17	Ban-sync DB → 2 LBs cross-DC via Galera	PAS
T18	Toronto backup : rsync + dump + tape	PAS
T19	`sp_register_user` assigne nœud le moins chargé	PAS
T20	Prometheus targets UP	PAS
T11-NY	iSCSI SAN VIP `192.168.2.200` sur storage1-ny	PAS
T12-NY	Heartbeat VIP failover storage1-ny → storage2-ny	PAS
T13-NY	RAID-1 NY	DEG
T14-NY	ProxySQL NY ONLINE	PAS

Suite du tableau D.1

ID	Test	Résul
T15-NY	Fail2ban SSH ban déclenche (NY)	PAS
T16-NY	HAProxy 403 sur IP bannie (NY)	PAS
T09-NY	Suricata IDS actif sur NY web	PAS
T-DNS	Résolution DNS `supfile.poc` → `100.126.8.98` (Toronto GSLB)	PAS
T-GSLB	HAProxy Toronto retire un DC en panne du pool en <15 s, trafic basculé vers le DC sain	PAS
T-APP	Application complète (frontend + backend) sur les 2 DCs	PAS

Bilan : 28 PASS / 30 tests ; 2 DEGR (RAID-1 limitation POC documentée).

CHAPITRE E

Glossaire

ACID Atomicity, Consistency, Isolation, Durability. Propriétés transactionnelles standard d'une base de données relationnelle.

AEAD Authenticated Encryption with Associated Data. Mode de chiffrement combinant confidentialité, intégrité et authentification (exemple : ChaCha20-Poly1305).

BIA Business Impact Analysis. Analyse formelle de l'impact métier de la perte d'un composant.

CapEx Capital Expenditure. Dépense d'investissement amortissable comptablement.

CHAP Challenge-Handshake Authentication Protocol. Authentification iSCSI standard.

Distributed-Replicate Topologie GlusterFS combinant la distribution (sharding) et la réplication.

DR Disaster Recovery. Procédures de reprise après sinistre.

FIM File Integrity Monitoring. Surveillance des modifications de fichiers sensibles.

Galera Cluster MariaDB multi-master synchrone basé sur la certification.

GeoDNS DNS qui répond selon la localisation géographique du client.

HA High Availability. Architecture sans point unique de défaillance.

Heartbeat Démon Linux gérant les VIPs flottantes via échange de keepalives.

HG Host Group. Regroupement de serveurs dans ProxySQL.

IPS Intrusion Prevention System. Système de prévention d'intrusion inline.

IST Incremental State Transfer. Synchronisation partielle d'un nœud Galera.

Keepalived Implémentation Linux du protocole VRRP.

LUN Logical Unit Number. Volume bloc exposé par une cible iSCSI.

OpEx Operational Expenditure. Dépense de fonctionnement récurrente.

PCA Plan de Continuité d'Activité. Équivalent français de BCP.

PFS Perfect Forward Secrecy. Rotation des clés de session.

PRA Plan de Reprise d'Activité. Équivalent français de DRP.

ProxySQL Reverse proxy MySQL/MariaDB avec read/write splitting.

Quorum Majorité requise pour valider une décision distribuée.

RAID-1 Miroir de deux disques. Tolère la perte d'un disque sans perte de donnée.

RPO Recovery Point Objective. Perte de données maximale acceptable en cas de sinistre.

RTO Recovery Time Objective. Durée maximale d'indisponibilité acceptable.

SAN Storage Area Network. Réseau dédié au stockage bloc.

SIEM Security Information and Event Management. Plateforme de corrélation d'événements sécurité.

SPOF Single Point Of Failure. Composant dont la défaillance entraîne l'indisponibilité totale.

SST State Snapshot Transfer. Synchronisation complète d'un nœud Galera.

Tailscale Overlay réseau basé sur WireGuard avec plan de contrôle géré.

VLAN Virtual LAN. Segmentation logique d'un réseau Ethernet.

VIP Virtual IP. Adresse IP flottante entre plusieurs serveurs.

VRRP Virtual Router Redundancy Protocol (RFC 5798). Élit dynamiquement un MASTER possédant une VIP.

Wazuh Plateforme SIEM open-source, fork d'OSSEC.

WireGuard Protocole VPN moderne (ChaCha20-Poly1305, Curve25519, BLAKE2s).