ALB PUBLIC AWS WAF

IDEO LAB

JANUARY 2026

1) Créer un Target Group (pour l'EC2)

AWS Console → EC2 > **Target Groups** > **Create target group**

- Type: **Instances**
- Protocol: **HTTP**
- Port: **80**

  (recommandé : ALB termine TLS, et parle en HTTP vers Nginx. Simple.)
- Health check:
  - Protocol: **HTTP**
  - Path: `/healthz/` (idéal) ou `/` (si tu n'as pas de endpoint health)
  - Success codes: `200-399`

Ensuite:

- **Register targets** → sélectionne ton EC2 → Add → Create

✅ Point de contrôle :

- Dans le Target Group, onglet **Targets** : l'instance doit passer **healthy**.
- Si "unhealthy" : c'est souvent SG/Nginx/healthcheck path.

2) Créer (ou valider) le certificat TLS dans ACM

AWS Console → **Certificate Manager (ACM)**

- Request a certificate
- Domain: `ideolab.com` et/ou `www.ideolab.com` (selon ton DNS)
- Validation: **DNS validation** (le plus simple)
- Ajoute les enregistrements DNS demandés

✅ Point de contrôle :

- Certificat en statut **Issued**

3) Créer l'ALB (Application Load Balancer)

AWS Console → EC2 > **Load Balancers** > **Create Load Balancer** > **Application Load Balancer**

Basic

- Scheme: **Internet-facing**
- IP type: IPv4
- VPC: celui de l'EC2
- Subnets: **2 AZ minimum** (obligatoire pour ALB)

Security Group (ALB)

- Crée un SG dédié, ex: `sg-alb-ideolab`
- Inbound:
  - TCP 443 from `0.0.0.0/0`
  - (optionnel) TCP 80 from `0.0.0.0/0` pour redirection vers 443

Listeners

- Listener 443 HTTPS:
  - Certificat: **ACM Issued**
  - Default action: Forward to **Target Group** créé en étape 1
- (Optionnel) Listener 80 HTTP:
  - Action: Redirect to HTTPS:443

4) Basculer le DNS (sans couper)

Route 53 (ou ton DNS provider)

- Mets `A/AAAA` (alias) ou `CNAME` vers le **DNS name de l'ALB**
  - `www` → ALB
  - racine `@` → Alias Route53 vers ALB (si Route53), sinon ALIAS/ANAME selon provider

✅ Point de contrôle (depuis ton poste) :

- Résolution DNS :

```bash
nslookup www.tondomaine.com
```

- Test HTTP :

```bash
curl -I https://www.tondomaine.com
```

Tant que tu ne verrouilles pas le SG EC2, tu peux revenir en arrière facilement.

5) Verrouiller l'EC2 (le vrai "bouclier")

C'est ici que tu élimines tcptrack / SYN flood sur l'EC2.

Dans le **Security Group de l'EC2** :

- Supprime "HTTPS 443 from 0.0.0.0/0"
- Ajoute à la place :
  - TCP 80 (ou 443 si tu gardes TLS interne) **source = SG de l'ALB** (`sg-alb-ideolab`)
- SSH:
  - seulement ton IP (ou mieux : SSM)

✅ Point de contrôle :

- Depuis Internet : ton site doit marcher via ALB
- Depuis une IP random : tu ne dois plus toucher directement l'EC2 en 443 (normal)

➡️ À ce moment-là, `tcptrack` sur l'EC2 devient calme, car l'EC2 ne reçoit plus Internet.

6) Ajouter AWS WAF (après stabilisation)

AWS Console → **WAF & Shield > Web ACLs > Create web ACL**

- Resource type: **Regional**
- Association: attacher à ton **ALB**

Règles recommandées (ordre) :

1. **AWS Managed Rules – Common Rule Set**
2. **Known bad inputs**
3. **Rate-based rule**
    - seuil de départ raisonnable : ex **2000 req / 5 min / IP** (à ajuster)
4. (Optionnel) Geo match si tu veux réduire la surface

✅ Point de contrôle :

- WAF logs / metrics : tu vois ce qui est bloqué
- Aucun faux positif majeur

ACTIONS SUR AWS

Paramètres - *immuable*

Choisissez un type de cible et l'équilibreur de charge et l'écouteur achemineront le trafic vers votre cible. Ces paramètres ne peuvent pas être modifiés après la création du groupe cible.«,

Type de cible

Indiquez le type de ressource que vous souhaitez cibler. Seul le type de ressource sélectionné peut être enregistré auprès de ce groupe cible.

- (●) **Instances**
  Supporte l'équilibrage de charge entre les instances d'un VPC. Intégrez les groupes Auto Scaling ou les services ECS pour une gestion automatique.
  Convient pour: ALB NLB GWLB
- ( ) **Adresses IP**
  Supporte l'équilibrage de charge entre le VPC et les ressources sur site. Facilite l'acheminement vers les adresses IP et les interfaces réseau de la même instance. Supporte les cibles IPv6.
  Convient pour: ALB NLB GWLB
- ( ) **Fonction Lambda**
  Supporte l'équilibrage de charge sur une seule fonction Lambda. ALB requis comme source de trafic.
  Convient pour: ALB
- ( ) **Application Load Balancer**
  Permet l'utilisation d'adresses IP statiques et de PrivateLink avec un Application Load Balancer. NLB requis comme source de trafic.
  Convient pour: NLB

Nom du groupe cible

Le nom doit être unique par région et par compte AWS.

tg-ideolab-https

Accepte :a-z, A-Z, 0-9, et tiret (-). Impossible de commencer ou de terminer par un tiret. 1- 32 caractères au total; Nombre: 16/32

Protocole

Protocole de communication entre l'équilibreur de charge et les cibles.

HTTPS ▼

Port

Numéro de port où les cibles reçoivent le trafic. Peut être contourné pour des cibles individuelles lors de l'inscription.

443

1-65535

Type d'adresse IP

Seules les cibles avec le type d'adresse IP indiqué peuvent être enregistrées dans ce groupe cible.

Détails du groupe cible

Nom	Type de cible	Protocole : Port	Version du protocole
tg-ideolab-https	Instance	HTTPS: 443	HTTP1

VPC	Type d'adresse IP
vpc-054c6890742b0ea50	IPv4

Détails de la surveillance de l'état

Protocole de vérification de l'état	Chemin de vérification de l'état	Port de vérification de l'état	Intervalle
HTTPS	/	traffic-port	30 secondes

Expiration	Seuil de bonne santé	Seuil de défectuosité	Codes de réussite
5 secondes	5	2	200-399

Étape 2 : Enregistrer les cibles

Modifier

Cibles (1)

ID d'instance	Nom	Port	Zone
i-0911840f16a5b65c1	IDEOLAB-1	443	eu-central-1a

Annuler | Précédent | Créer un groupe cible

Groupe cible créé avec succès : **tg-ideolab-https**. La détection des anomalies est automatiquement appliquée à toutes les cibles enregistrées. Les résultats peuvent être consultés dans l'onglet **Cibles**.

Donnez votre avis

tg-ideolab-https

Actions

Détails

arn:aws:elasticloadbalancing:eu-central-1:782198888023:targetgroup/tg-ideolab-https/a7116badc9dedf5f

Type de cible	Protocole : Port	Version du protocole	VPC
Instance	HTTPS: 443	HTTP1	vpc-054c6890742b0ea50

Type d'adresse IP	Équilibreur de charge
IPv4	Aucun associé

1	0	0	1	0	0
Total des cibles	Sain	Non sain	Non utilisé	Initial	Drainage
	0 Anormal				

▶ **Répartition des cibles par zone de disponibilité (AZ)**

Sélectionnez les valeurs de ce tableau pour voir les filtres correspondants appliqués au tableau Cibles enregistrées ci-dessous.

Cibles | **Surveillance** | **Surveillances de l'état** | **Attributs** | **Balises**

Cibles enregistrées (1) Infos

Gestion des anomalies : **Non applicable**

Annuler l'enregistrement | Enregistrer les cibles

Les groupes cibles acheminent les demandes vers des cibles individuelles enregistrées à l'aide du protocole et du numéro de port que vous spécifiez. Les surveillances de l'état sont effectuées sur toutes les cibles enregistrées conformément aux paramètres de la surveillance de l'état du groupe cible. La détection des anomalies est automatiquement appliquée aux groupes cibles HTTP/HTTPS avec au moins trois cibles saines.

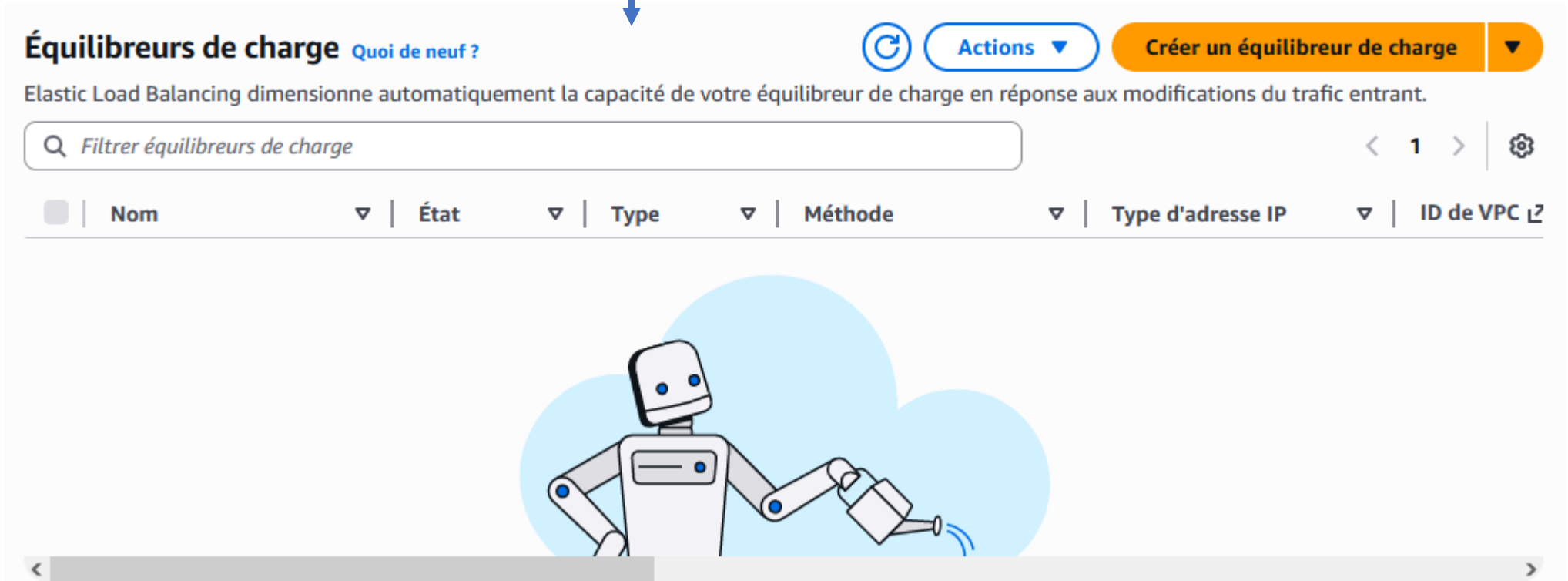
Filtrer cibles

1

CREATION D’UN ALB

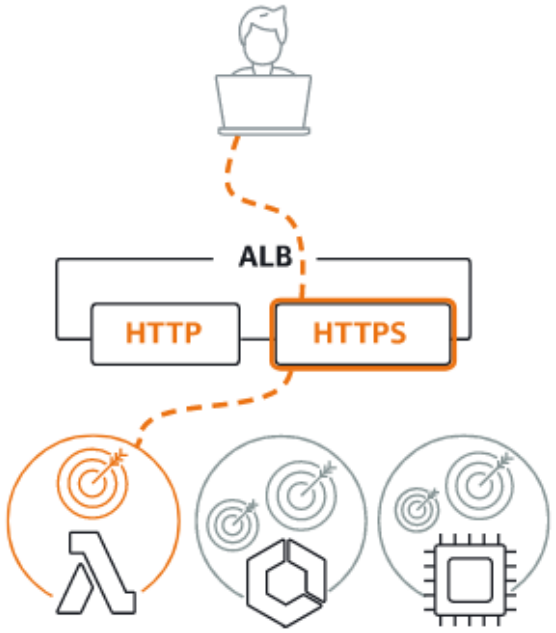
Équilibrage de charge

- Équilibreurs de charge
- Groupes cibles
- Trust Stores



Types d'équilibreurs de charge

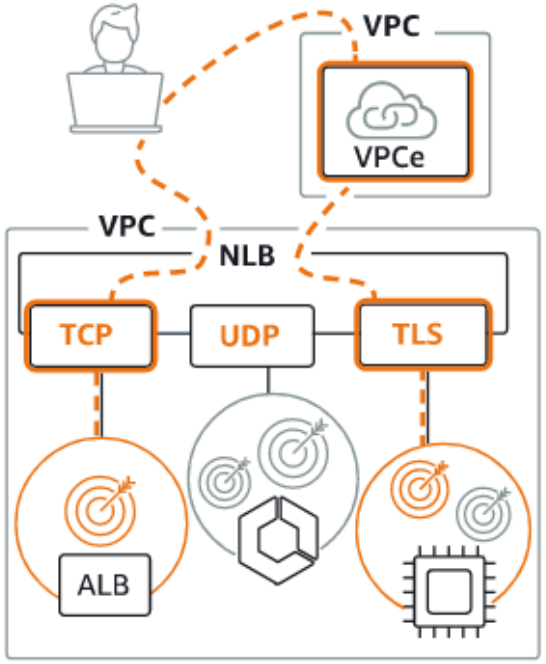
Application Load Balancer Infos



Choisissez un Application Load Balancer quand vous avez besoin d'un ensemble de fonctions flexible pour vos applications Web avec un trafic HTTP et HTTPS. En opérant au niveau des demandes, les Application Load Balancers fournissent des fonctions avancées de routage et de visibilité ciblant les architectures d'application, y compris les microservices et les conteneurs.

Créer

Network Load Balancer Infos



Choisissez un Network Load Balancer quand vous avez besoin de performances très élevées, de déchargement TLS à grande échelle, d'un déploiement de certificat centralisé, de la prise en charge d'UDP et d'adresses IP statiques pour vos applications. En opérant au niveau de la connexion, les Network Load Balancers sont capables de traiter des millions de demandes par seconde en toute sécurité tout en assurant des latences ultra-faibles.

Créer

Équilibreur de charge de passerelle Infos

Choisissez un Équilibreur de charge de passerelle lorsque vous devez déployer et gérer une flotte d'appliances virtuelles tiers prenant en charge GENEVE. Ces appliances vous permettent d'améliorer la sécurité, la conformité et les contrôles de politique.

Créer

Créer un Application Load Balancer Infos

L'Application Load Balancer répartit le trafic HTTP et HTTPS entrant sur plusieurs cibles, telles que des instances Amazon EC2, des microservices et des conteneurs, en fonction des attributs de la demande. Lorsque l'équilibreur de charge reçoit une demande de connexion, il évalue les règles de l'écouteur par ordre de priorité pour déterminer quelle règle appliquer et, le cas échéant, il sélectionne la cible de l'action de la règle dans le groupe de cibles.

▶ Fonctionnement des Application Load Balancers

Configuration de base

Nom de l'équilibreur de charge

Le nom doit être unique au sein de votre compte AWS et ne peut pas être modifié une fois l'équilibreur de charge créé.

32 caractères alphanumériques au maximum, y compris les traits d'union, sont autorisés mais le nom ne doit pas commencer ou se terminer par un trait d'union.

Méthode | Infos

Le schéma ne peut pas être modifié après la création de l'équilibreur de charge.

- (•) **Accessible sur Internet**
  - Sert le trafic connecté à Internet.
  - Dispose d'adresses IP publiques.
  - Le nom DNS est résolu en adresses IP publiques.
  - Nécessite un sous-réseau public.
- ( ) **Interne**
  - Sert le trafic interne.
  - Dispose d'adresses IP privées.
  - Le nom DNS est résolu en adresses IP privées.
  - Compatible avec les types d'adresses IP **IPv4** et **Dualstack**.

Types d'adresse IP de l'équilibreur de charge | Infos

Sélectionnez le type d'adresse IP frontale à attribuer à l'équilibreur de charge. Le VPC et les sous-réseaux mappés à cet équilibreur de charge doivent inclure les types d'adresses IP sélectionnés. Les adresses IPv4 publiques entraînent des frais supplémentaires.

- (•) **IPv4**
  Inclut uniquement les adresses IPv4.
- ( ) **Dualstack**
  Inclut les adresses IPv6 et IPv4.
- ( ) **Dualstack sans IPv4 public**
  Inclut une adresse IPv6 publique et des adresses IPv4 et IPv6 privées. Compatible uniquement avec les équilibreurs de charge **accessibles sur Internet**.

BLOCAGE ANTI FLOOD

SAVE CONFIG AND REVERT

- 1.1 Sauvegarde iptables (rollback en 1 commande)
  - sudo iptables-save > /root/iptables.pre_flood.$(date +%F_%H%M%S).rules

- Pour revenir EXACTEMENT à l'état initial :
  - sudo iptables-restore < /root/iptables.pre_flood.YYYY-MM-DD_HHMMSS.rules

- 1.2 Sauvegarde sysctl (rollback kernel)sudo sysctl -a > /root/sysctl.pre_flood.$(date +%F_%H%M%S).txt

2.1 Activer la protection SYN (immédiat)

bash Copier le code

```
sudo sysctl -w net.ipv4.tcp_syncookies=1
sudo sysctl -w net.ipv4.tcp_max_syn_backlog=8192
sudo sysctl -w net.ipv4.tcp_synack_retries=2
sudo sysctl -w net.core.somaxconn=4096
```

2.2 Activer TCP SYNPROXY côté kernel (si dispo) — option très efficace

Vérifie si le module existe :

bash Copier le code

```
sudo modprobe nf_synproxy_core 2>/dev/null && echo OK || echo "nf_synproxy_cor
```

👉 Si dispo, on peut faire un SYNPROXY **réversible** (je te le donne plus bas), mais d'abord on reste sur le "safe".

2.3 Vérifier que Nginx (ou ton reverse-proxy) n'est pas le goulot

bash Copier le code

```
sudo ss -lntp | grep ':443'
sudo systemctl status nginx --no-pager 2>/dev/null || true
sudo nginx -T 2>/dev/null | head
```

ACTIVER PROTECTION SYN

- sudo sysctl -w net.ipv4.tcp_syncookies=1
- sudo sysctl -w net.ipv4.tcp_max_syn_backlog=8192
- sudo sysctl -w net.ipv4.tcp_synack_retries=2
- sudo sysctl -w net.core.somaxconn=4096

Rollback (si tu veux annuler cette étape)

bash Copier le code

```
sudo sysctl -w net.ipv4.tcp_max_syn_backlog=128
sudo sysctl -w net.ipv4.tcp_synack_retries=5
sudo sysctl -w net.core.somaxconn=128
```

Vérifie l'effet (1 commande)

bash Copier le code

```
sudo ss -ant state syn-recv '( sport = :443 )' | wc -l
```

ÉTAPE 2 (SAFE, ciblée, réversible)

👉 Limiter UNIQUEMENT les SYN vers 443 au niveau firewall sans toucher au reste.

⚙️ Action (1 seule commande logique, nftables)

bash

Copier le code

```
sudo nft add table inet synlimit 2>/dev/null || true
sudo nft 'add chain inet synlimit input { type filter hook input priority -150
sudo nft add rule inet synlimit input ct state established,related accept
sudo nft add rule inet synlimit input tcp dport 443 tcp flags syn ct state new
sudo nft add rule inet synlimit input tcp dport 443 tcp flags syn ct state new
```

Ce que ça fait (simplement) :

- laisse passer les connexions existantes
- accepte jusqu'à **200 SYN/s** (burst 400)
- **drop uniquement l'excès**
- HTTPS normal OK

Limiter UNIQUEMENT les SYN vers 443 au niveau firewall

- sudo nft add table inet synlimit 2>/dev/null || true
- sudo nft 'add chain inet synlimit input { type filter hook input priority -150; policy accept; }' 2>/dev/null || true
- sudo nft add rule inet synlimit input ct state established,related accept
- sudo nft add rule inet synlimit input tcp dport 443 tcp flags syn ct state new limit rate 200/second burst 400 packets accept
- sudo nft add rule inet synlimit input tcp dport 443 tcp flags syn ct state new drop

🔄 **ROLLBACK IMMÉDIAT (1 commande)**

bash — Copier le code

```
sudo nft delete table inet synlimit
```

ÉTAPE UNIQUE : on recrée correctement table+chain, puis on ajoute 2 règles SYN (safe)

Copie/colle ce bloc entier :

bash Copier le code

```
# 0) Nettoyage si un truc partiel existe
sudo nft delete table inet synlimit 2>/dev/null || true

# 1) Création table + chain hook INPUT
sudo nft -f - <<'NFT'
table inet synlimit {
  chain input {
    type filter hook input priority -150; policy accept;

    ct state established,related accept

    tcp dport 443 tcp flags syn ct state new limit rate 200/second burst 400 p
    tcp dport 443 tcp flags syn ct state new drop
  }
}
NFT
```

Rollback immédiat (1 commande)

bash Copier le code

```
sudo nft delete table inet synlimit
```

NEXT STEPS

1) Rétention + rotation “pro”

- ✅ Commande Django `cleanup_accesslog` (30j AccessLog, 180j TrashIP non-banned)
- ✅ Tâche planifiée quotidienne
- Option “safe” : ne jamais purger les `TrashIP.is_banned=True` (ou purger au bout de 365j).

2) Stats utiles (pas juste des graphes jolis)

À ajouter à ton dashboard admin “Stats” :

- **Hits/jour** (200 only) + **unique IP/jour**
- **Top 20 pages** (paths) + **top referers**
- **Perf** : avg + p95 `duration_ms`
- **Erreurs** : 4xx/5xx par jour (mais sans polluer AccessLog si tu veux, on peut compter côté middleware)
- **Bots / scanners** : top IP 404, top “sensitive hits”, % d’IPs bannies

👉 Et un bouton “BAN iptables” / “UNBAN” depuis la table poubelle (tu l’as déjà, on améliore l’UX).

3) Auto-ban intelligent (le vrai gain)

Aujourd'hui tu stockes les 404 répétitifs. Étape suivante :

- Règles d'auto-ban :
  - `>= X 404 / 10 min` OU `>= Y sensitive paths` OU `>= Z req/s`
- Ban "temporaire" (TTL) : `ban_until` + un cron `unban_expired`
- Whitelist : ta/tes IP, monitoring, Googlebot/Bingbot (facultatif)

4) Migration iptables → nftables (recommandé)

Si tu commences à bannir beaucoup d'IPs, iptables va devenir pénible.

- nftables **set** (hash:set) = ban/unban O(1), ultra rapide
- Tu gardes l'admin identique, seul le "backend firewall" change.

5) "Evidence mode" (super utile)

Ajouter une table `FirewallActionLog` :

- qui a cliqué (user admin)
- action (ban/unban/check)
- commande exécutée
- rc/stdout/stderr
- timestamp
- ip concernée

  ➡️ Ça te donne audit + rollback fiable.