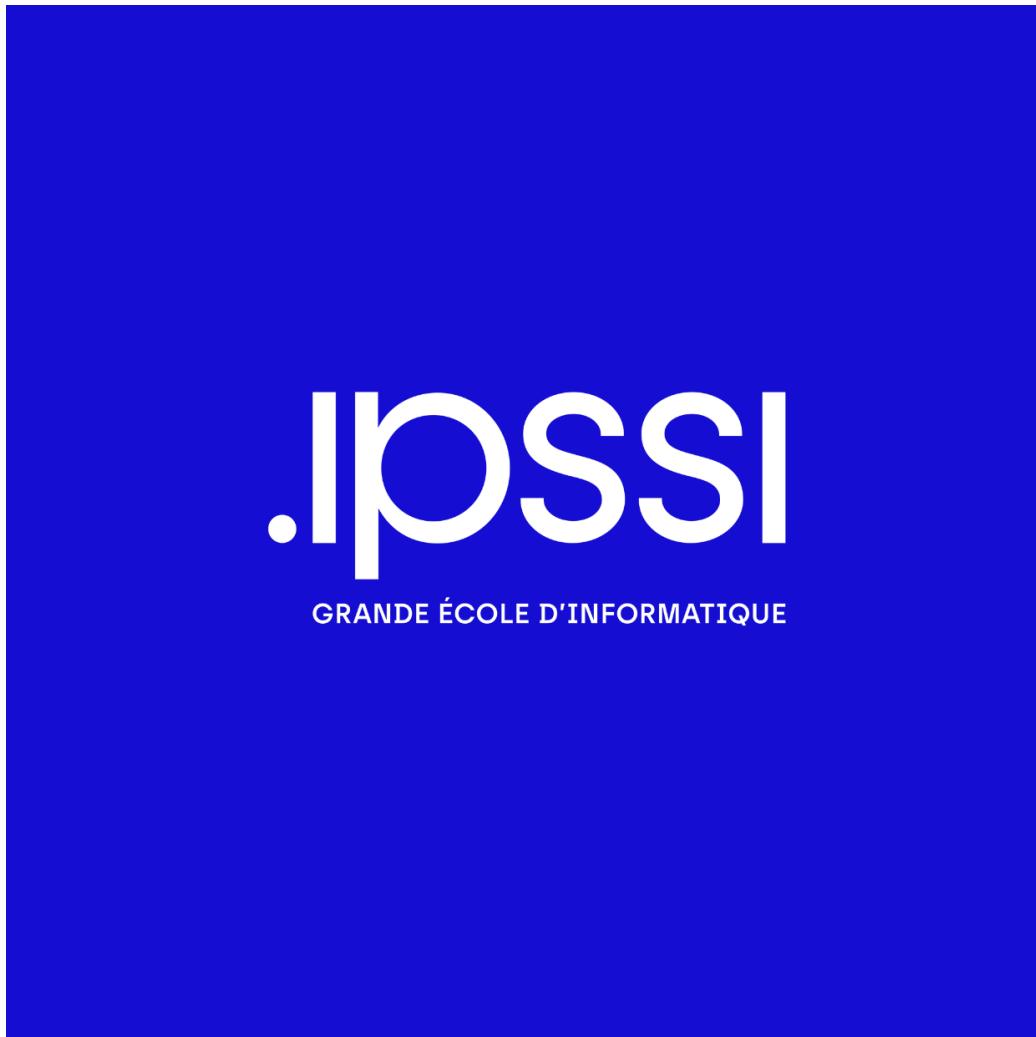


# Documentation Bankso

## (AWS)



# Sommaire

## Table des matières

Table des matières .....	2
Création des réseaux.....	3
Création du VPC .....	3
Création des sous-réseaux.....	3
Création de l'Internet Gateway .....	6
Création de la NAT Gateway .....	8
Machine d'administration (Sous-réseau Public) .....	9
Machines serveurs (Sous-réseau Privé).....	13
Machines admin (Sous-réseau Public).....	16
Vérification.....	18
Installation de Wireguard (VPN).....	20
5.SSH via VPN .....	24
Infrastructure Web .....	25
Prérequis.....	25
Installation et Mise en place de serveurs Web .....	26
Test.....	28
Installation d'un système de base de données .....	28
Installation d'un système SSL Offloader.....	30
Test.....	30
Installation et Mise en place d'une PKI.....	31
Création de la CA .....	31
Génération d'un certificat pour HAProxy .....	32
Test.....	33
Installation et Mise en place d'un système de détection d'intrusion .....	34
Test.....	35
Alternative Proxmox (Conteneurs LXC).....	35

# Création des réseaux

Nous allons mettre en place un réseau que nous segmenterons en plusieurs sous-réseaux, notamment :

**Le sous-réseau d'administration** (admin + load balancing), qui aura accès à Internet.

**Le sous-réseau de production** (serveurs web ), isolé pour renforcer la sécurité.

Cette segmentation respecte le principe de séparation des responsabilités, une bonne pratique en cybersécurité. En isolant les sous-réseaux, nous réduisons la surface d'attaque, améliorons la sécurité et facilitons la gestion du trafic réseau.

## Création du VPC

1. Aller sur AWS Management Console → VPC → Créer un VPC.
2. Nommer le VPC, choisir une plage CIDR (ex: 10.0.0.0/16).
3. Créer.

VPC > Vos VPC > Créer un VPC

---

### Créer un VPC Infos

Un VPC est une partie isolée du Cloud AWS remplie d'objets AWS, tels que des instances Amazon EC2.

**Paramètres VPC**

Ressources à créer Infos  
Créez uniquement la ressource VPC ou le VPC et d'autres ressources réseaux.

VPC uniquement  VPC et plus encore

**Identification de nom - facultatif**  
Crée une identification avec une clé du « Nom » et une valeur que vous spécifiez.  
tp-aws-vpc-public

**Bloc d'adresses CIDR IPv4 Infos**

Entrée manuelle CIDR IPv4  
 Bloc d'adresse CIDR IPv4 alloué à IPAM

**CIDR IPv4**  
10.0.0.0/16

La taille du bloc d'adresse CIDR doit être comprise entre /16 et /28.

**Bloc CIDR IPv6 Infos**

Aucun bloc d'adresses CIDR IPv6  
 Bloc d'adresse CIDR IPv6 alloué à IPAM  
 Bloc d'adresses CIDR IPv6 fourni par Amazon  
 CIDR IPv6 dont je suis propriétaire

# Création des sous-réseaux

## Sous-réseau Public

1. Aller sur **VPC** → **Sous-réseaux** → **Créer un sous-réseau**.
2. Sélectionner le VPC, donner un nom (public-subnet).

3. Spécifier une plage CIDR (ex: 10.0.1.0/24).
4. **Activer l'attribution automatique des IP publiques.**
5. **Créer.**

[VPC](#) > [Sous-réseaux](#) > Créer un sous-réseau

## Créer un sous-réseau Infos

### VPC

#### ID de VPC

Créez des sous-réseaux dans ce VPC.

vpc-0fded5e5175e3b5cb (tp-aws-vpc-public)



#### CIDR de VPC associés

#### CIDR IPv4

10.0.0.0/16

### Paramètres du sous-réseau

Précisez les blocs d'adresse CIDR et la zone de disponibilité pour le sous-réseau.

#### Sous-réseau 1 sur 1

##### Nom du sous-réseau (subnet)

Créez une balise avec une clé « Name » et une valeur à spécifier.

tp-aws-public-subnet1

Le nom peut comporter jusqu'à 256 caractères.

##### Zone de disponibilité Infos

Choisissez la zone dans laquelle votre sous-réseau résidera ou laissez Amazon en choisir une pour vous.

États-Unis (Virginie du Nord) / us-east-1a



##### Bloc d'adresse CIDR IPv4 VPC Infos

Choisissez le bloc d'adresse CIDR IPv4 du VPC pour le sous-réseau. L'adresse CIDR IPv4 du sous-réseau doit se trouver dans ce bloc.

10.0.0.0/16



##### Bloc d'adresse CIDR de sous-réseau IPv4

10.0.1.0/24

256 IPs



## Sous-réseau Privé

1. Même procédure mais avec un CIDR différent (ex: 10.0.2.0/24) et (ex: 10.0.3.0/24).
2. Ne pas activer l'attribution automatique des IP publiques.
3. **Créer.**

## Créer un sous-réseau Infos

### VPC

#### ID de VPC

Créez des sous-réseaux dans ce VPC.

vpc-0fded5e5175e3b5cb (tp-aws-vpc-public) ▾

#### CIDR de VPC associés

#### CIDR IPv4

10.0.0.0/16

### Paramètres du sous-réseau

Précisez les blocs d'adresse CIDR et la zone de disponibilité pour le sous-réseau.

#### Sous-réseau 1 sur 1

##### Nom du sous-réseau (subnet)

Créez une balise avec une clé « Name » et une valeur à spécifier.

tp-aws-prive-subnet3 ▾

Le nom peut comporter jusqu'à 256 caractères.

##### Zone de disponibilité Infos

Choisissez la zone dans laquelle votre sous-réseau résidera ou laissez Amazon en choisir une pour vous.

États-Unis (Virginie du Nord) / us-east-1a ▾

##### Bloc d'adresse CIDR IPv4 VPC Infos

Choisissez le bloc d'adresse CIDR IPv4 du VPC pour le sous-réseau. L'adresse CIDR IPv4 du sous-réseau doit se trouver dans ce bloc.

10.0.0.0/16 ▾

##### Bloc d'adresse CIDR de sous-réseau IPv4

10.0.3.0/24

256 IPs

◀ ▶ ⌂ ⌃

## Créer un sous-réseau Infos

### VPC

#### ID de VPC

Créez des sous-réseaux dans ce VPC.

vpc-0fded5e5175e3b5cb (tp-aws-vpc-public)



#### CIDR de VPC associés

#### CIDR IPv4

10.0.0.0/16

### Paramètres du sous-réseau

Précisez les blocs d'adresse CIDR et la zone de disponibilité pour le sous-réseau.

#### Sous-réseau 1 sur 1

##### Nom du sous-réseau (subnet)

Créez une balise avec une clé « Name » et une valeur à spécifier.

tp-aws-prive-subnet-2

Le nom peut comporter jusqu'à 256 caractères.

##### Zone de disponibilité Infos

Choisissez la zone dans laquelle votre sous-réseau résidera ou laissez Amazon en choisir une pour vous.

États-Unis (Virginie du Nord) / us-east-1a



##### Bloc d'adresse CIDR IPv4 VPC Infos

Choisissez le bloc d'adresse CIDR IPv4 du VPC pour le sous-réseau. L'adresse CIDR IPv4 du sous-réseau doit se trouver dans ce bloc.

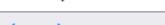
10.0.0.0/16



##### Bloc d'adresse CIDR de sous-réseau IPv4

10.0.2.0/24

256 IPs



## Création de l'Internet Gateway

### 1. Aller sur VPC → Internet Gateways → Créer une passerelle.

## Créer une passerelle Internet Infos

Une passerelle Internet est un routeur virtuel qui connecte un VPC à Internet. Pour créer une nouvelle passerelle Internet, spécifiez le nom de la pas

### Paramètres de passerelle Internet

#### Identification de nom

Crée une identification avec une clé du « Nom » et une valeur que vous spécifiez.

tp-aws-gateway

### 2. L'attacher au VPC.

## Attacher au VPC (igw-008a5f53e71c536a1) Infos

### VPC

Attachez une passerelle Internet à un VPC pour permettre au VPC de communiquer avec Internet. Spécifiez le VPC à attacher ci-dessous.

#### VPC disponibles

Attachez la passerelle Internet à ce VPC.



#### ▶ Commande AWS Command Line Interface

### 3. Aller sur VPC → Tables de routage → Créez une table de routage.

## Créez une table de routage Infos

Une table de routage spécifie comment les paquets sont transférés entre les sous-réseaux au sein de votre VPC, sur Internet et votre connexion VPN.

### Paramètres de la table de routage

#### Nom - facultatif

Créez une balise avec une clé « Name » et une valeur à spécifier.

#### VPC

VPC à utiliser pour cette table de routage.



### 4. Modifier la table de routage du sous-réseau public pour rediriger le trafic vers l'Internet Gateway (0.0.0.0/0 → Internet Gateway).

## Modifier des routes

Destination	Cible	Statut
10.0.0.0/16	local	<input checked="" type="checkbox"/> Actif
<input type="text" value="0.0.0.0/0"/>	<input type="text" value="local"/>	
<input type="text" value="0.0.0.0/0"/>	<input type="text" value="Passerelle Internet"/>	
<input type="text" value="0.0.0.0/0"/>	<input type="text" value="igw-008a5f53e71c536a1"/>	

[Ajouter une route](#)



## Modifier les associations de sous-réseau

Modifiez les sous-réseaux associés à cette table de routage.

### Sous-réseaux disponibles (1/3)

<input type="checkbox"/>	Nom	ID de sous-réseau	CIDR IPv4	CIDR IPv6	ID de la table de routage
<input type="checkbox"/>	tp-aws-prive-subnet-2	subnet-03c34795627b647a7	10.0.2.0/24	-	Principal (rtb-049e8cfcd416ecbe)
<input checked="" type="checkbox"/>	tp-aws-public-subnet1	subnet-0b165441a0be2b230	10.0.1.0/24	-	Principal (rtb-049e8cfcd416ecbe)
<input type="checkbox"/>	tp-aws-prive-subnet3	subnet-0851ced2d7dd413e3	10.0.3.0/24	-	Principal (rtb-049e8cfcd416ecbe)

### Sous-réseaux sélectionnés

Annuler

[Enregistrer les associations](#)

## Création de la NAT Gateway

1. Aller sur **VPC** → **NAT Gateway** → **Créer une NAT Gateway**.
2. Sélectionner le sous-réseau public et une adresse IP élastique.

VPC > Passerelles NAT > Créer une passerelle NAT

**Créer une passerelle NAT** Infos

Service géré de traduction d'adresses réseau (NAT) hautement disponible que les instances des sous-réseaux privés peuvent utiliser pour se connecter aux services d'autres VPC, réseaux sur site ou Internet.

**Paramètres de passerelle NAT**

**Nom - facultatif**  
Créez une balise avec une clé « Name » et une valeur à spécifier.  
  
Le nom peut comporter jusqu'à 256 caractères.

**Sous-réseau**  
Sélectionnez un sous-réseau où vous voulez créer la passerelle NAT.

**Type de connectivité**  
Sélectionnez un type de connectivité pour la passerelle NAT.  
 Publique  
 Privée

**ID d'allocation d'adresses IP Elastic** Infos  
Attribuez une adresse IP Elastic à la passerelle NAT.

3. Modifier la table de routage du sous-réseau privé pour que le trafic sortant passe par la NAT Gateway (0.0.0.0/0 → NAT Gateway).

VPC > Tables de routage > Créer une table de routage

**Créer une table de routage** Infos

Une table de routage spécifie comment les paquets sont transférés entre les sous-réseaux au sein de votre VPC, sur Internet et votre connexion VPN.

**Paramètres de la table de routage**

**Nom - facultatif**  
Créez une balise avec une clé « Name » et une valeur à spécifier.

**VPC**  
VPC à utiliser pour cette table de routage.

VPC > Tables de routage > rtb-080deb63057f66653 > Modifier des routes

### Modifier des routes

Destination	Cible	Statut
10.0.0.0/16	local	<input checked="" type="checkbox"/> Actif
	<input type="text" value="local"/> <input type="button" value="X"/>	<input type="button" value="X"/>
0.0.0.0/0	Passerelle NAT	-
	<input type="text" value="Passerelle NAT"/> <input type="button" value="X"/>	<input type="button" value="X"/>
	<input type="text" value="nat-053809619b73a2ba4"/> <input type="button" value="X"/>	<input type="button" value="X"/>
	<input type="button" value="Ajouter une route"/>	

## Modifier les associations de sous-réseau

Modifiez les sous-réseaux associés à cette table de routage.

### Sous-réseaux disponibles (2/3)

Filtrer les associations de sous-réseau			
	Nom	ID de sous-réseau	CIDR IPv4
<input checked="" type="checkbox"/>	tp-aws-prive-subnet-2	subnet-03c34795627b647a7	10.0.2.0/24
<input type="checkbox"/>	tp-aws-public-subnet1	subnet-0b165441a0be2b230	10.0.1.0/24
<input checked="" type="checkbox"/>	tp-aws-prive-subnet3	subnet-0851ced2d7dd413e3	10.0.3.0/24

### Sous-réseaux sélectionnés

subnet-03c34795627b647a7 / tp-aws-prive-subnet-2 X subnet-0851ced2d7dd413e3 / tp-aws-prive-subnet3 X

## Création des instances EC2

Nous allons déployer plusieurs services sur cette infrastructure pour accompagner l'évolution de l'entreprise. Cela inclut une machine d'administration, un HAProxy et des serveurs web.

### Machine d'administration (Sous-réseau Public)

1. Aller sur **EC2** → **Lancer une instance**.
2. Sélectionner Amazon Linux 2 ou Ubuntu.

#### Lancer une instance Informations

Amazon EC2 vous permet de créer des machines virtuelles, ou des instances, qui s'exécutent sur le Cloud AWS. Démarrer rapidement en suivant les étapes simples indiquées ci-dessous.

#### Nom et balises Informations

Nom

tp-aws-haproxy

Ajouter des balises supplémentaires

#### ▼ Images d'applications et de systèmes d'exploitation (Amazon Machine Image) Informations

Une AMI est un modèle contenant la configuration logicielle (système d'exploitation, serveur d'applications et applications) requise pour lancer votre instance. Parcourez ou recherchez des AMI si vous ne trouvez pas ce que vous recherchez ci-dessous.

Q Effectuer une recherche dans notre catalogue complet, qui comprend des milliers d'images d'applications et de systèmes d'exploitation

Récentes

Démarrage rapide



Explorer plus d'AMI

Y compris les AMI d'AWS, de Marketplace et de la communauté

### Amazon Machine Image (AMI)

AMI Amazon Linux 2023	Éligible à l'offre gratuite
ami-05b10e08d247fb927 (64 bits (x86), uefi-preferred) / ami-0f37c4a1ba152af46 (64 bits (Arm), uefi) Virtualisation: hvm ENA activé: true Type de périphérique racine: ebs	▼

#### Description

Amazon Linux 2023 est un système d'exploitation moderne basé sur Linux, à usage général et offrant cinq ans de support garanti. Optimisé pour AWS, il est conçu afin de fournir un environnement d'exécution sécurisé, stable et à hautes performances pour le développement et l'exécution de vos applications cloud.

Amazon Linux 2023 AMI 2023.6.20250218.2 x86\_64 HVM kernel-6.1

Architecture	Mode de démarrage	ID AMI	Nom d'utilisateur	ⓘ
64 bits (x86) ▼	uefi-preferred	ami-05b10e08d247fb927	ec2-user	Fournisseur vérifié

### ▼ Type d'instance [Informations](#) | [Obtenez des conseils](#)

#### Type d'instance

t2.micro	Éligible à l'offre gratuite
Famille: t2 1 vCPU 1 Gio Mémoire Génération actuelle: true	▼
À la demande Windows base tarification: 0.0162 USD per Hour	
À la demande Ubuntu Pro base tarification: 0.0134 USD per Hour	
À la demande SUSE base tarification: 0.0116 USD per Hour	
À la demande RHEL base tarification: 0.026 USD per Hour	À la demande Linux base tarification: 0.0116 USD per Hour

Toutes les générations

[Comparer les types d'instance](#)

3. Créer un pair de clé en pem pour se connecter en ssh .
4. Placer l'instance dans le sous-réseau public.
5. Activer l'IP publique.

### ▼ Paire de clés (connexion) [Informations](#)

Vous pouvez utiliser une paire de clés pour vous connecter en toute sécurité à votre instance. Assurez-vous d'avoir accès à la paire de clés sélectionnée avant de lancer l'instance.

#### Nom de la paire de clés - *obligatoire*

tp-aws-haproxy

[Créer une paire de clés](#)

### ▼ Paramètres réseau [Informations](#)

#### VPC - *obligatoire* | [Informations](#)

vpc-0fded5e175e3b5cb (tp-aws-vpc-public)  
10.0.0.0/16

↻

#### Sous-réseau | [Informations](#)

subnet-0b165441a0be2b230 tp-aws-public-subnet1  
VPC: vpc-0fded5e175e3b5cb Propriétaire: 590332306376 Zone de disponibilité: us-east-1a  
Type de zone: Zone de disponibilité Adresses IP disponibles: 250 CIDR: 10.0.1.0/24

↻

[Créer un nouveau sous-réseau](#) ↗

#### Attribuer automatiquement l'adresse IP publique | [Informations](#)

Activer

Des frais supplémentaires s'appliquent en cas de dépassement de la [limite de l'offre gratuite](#)

6. Ajouter une règle de sécurité pour permettre le SSH (22 depuis 0.0.0.0/0).
7. Ajouter une règle de sécurité pour permettre le HTTP (80 depuis 0.0.0.0/0).

Nom du groupe de sécurité - *obligatoire*

Ce groupe de sécurité sera ajouté à toutes les interfaces réseau. Le nom ne peut pas être modifié après la création du groupe de sécurité. La longueur maximale est de 255 caractères. Caractères valides : a-z, A-Z, 0-9, espaces et\_-:/@#= & ; ! \$\*

Description - *obligatoire* | [Informations](#)

Règles entrantes des groupes de sécurité

▼ Règle de groupe de sécurité 1 (TCP, 22, 0.0.0.0/0) [Supprimer](#)

Type   <a href="#">Informations</a> ssh	Protocole   <a href="#">Informations</a> TCP	Plage de ports   <a href="#">Informations</a> 22
Type de source   <a href="#">Informations</a> N'importe où	Source   <a href="#">Informations</a> <input type="text" value="Ajouter une adresse CIDR, une liste de préfixe"/> 0.0.0.0/0 <a href="#">X</a>	Description - <i>facultatif</i>   <a href="#">Informations</a> par exemple, SSH pour le bureau de l'administrateur

▼ Règle de groupe de sécurité 2 (TCP, 80, 0.0.0.0/0) [Supprimer](#)

Type   <a href="#">Informations</a> HTTP	Protocole   <a href="#">Informations</a> TCP	Plage de ports   <a href="#">Informations</a> 80
Type de source   <a href="#">Informations</a> N'importe où	Source   <a href="#">Informations</a> <input type="text" value="Ajouter une adresse CIDR, une liste de préfixe"/>	Description - <i>facultatif</i>   <a href="#">Informations</a> par exemple. SSH pour le bureau de l'administrateur

## 8. Ajouter un script pour installer nginx reverse proxy et son fichier configuration pour faire le loadbalancing

### Données utilisateur - *facultatif* | [Informations](#)

Chargez un fichier contenant vos données utilisateur ou saisissez-les dans le champ.

[↑ Choisir un fichier](#)

```
#!/bin/bash
yum install -y nginx
systemctl start nginx
systemctl enable nginx
echo "upstream backend {
    server 10.0.2.10;
    server 10.0.2.11;
}

server {
    listen 80;
    location / {
        proxy_pass http://backend;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
    }
}" | tee /etc/nginx/conf.d/load_balancer.conf
sudo systemctl restart nginx
```

## 9. Lancer et connecter vous en ssh a votre conteneur .

## Connectez-vous à l'instance

Connectez-vous à votre instance à i-04f0e15fa5e56a1f7 (tp-aws-haproxy) l'aide de l'une de ces options

EC2 Instance Connect	Session Manager	Client SSH	EC2 Serial Console
<p><b>ID d'instance</b></p> <p><input checked="" type="checkbox"/> i-04f0e15fa5e56a1f7 (tp-aws-haproxy)</p> <p>1. Ouvrez un client SSH.</p> <p>2. Recherchez votre fichier de clé privée. La clé utilisée pour lancer cette instance est tp-aws-haproxy.pem</p> <p>3. Exécutez, si nécessaire, cette commande pour vous assurer que votre clé n'est pas visible publiquement.</p> <p><input checked="" type="checkbox"/> chmod 400 "tp-aws-haproxy.pem"</p> <p>4. Connectez-vous à votre instance à l'aide de son IP publique :</p> <p><input checked="" type="checkbox"/> 50.19.71.183</p> <p>Exemple :</p> <p><input checked="" type="checkbox"/> ssh -i "tp-aws-haproxy.pem" ec2-user@50.19.71.183</p> <p><b>Remarque :</b> Dans la plupart des cas, le nom d'utilisateur deviné est correct. Cependant, lisez les instructions de sécurité et les paramètres de sécurité de l'instance pour vous assurer que vous utilisez les meilleures pratiques de sécurité.</p>			

```
[vand1@LAPTOP-EUJQQQAT] ~ ~/Downloads
$ ssh -i "tp-aws-haproxy.pem" ec2-user@50.19.71.183
The authenticity of host '50.19.71.183 (50.19.71.183)' can't be established.
ED25519 key fingerprint is SHA256:D2Ko/wD9BF+3N5q9hqQAusft7gZtoGCdcU94BCo9Lak.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '50.19.71.183' (ED25519) to the list of known hosts.

      #
      _###_
      Amazon Linux 2023
     _\###\
    \###|
    \###|
    \#/ __ https://aws.amazon.com/linux/amazon-linux-2023
   V~, '-->
   /_
  /_/
 /_m'

[ec2-user@ip-10-0-1-49 ~]$ sudo systemctl status nginx.service
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
   Active: active (running) since Fri 2025-02-28 10:18:17 UTC; 41s ago
     Process: 4567 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
     Process: 4579 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
     Process: 4615 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
 Main PID: 4661 (nginx)
    Tasks: 2 (limit: 1111)
   Memory: 2.5M
      CPU: 42ms
     CGroup: /system.slice/nginx.service
             └─4661 "nginx: master process /usr/sbin/nginx"
                  ├─4662 "nginx: worker process"

Feb 28 10:18:17 ip-10-0-1-49.ec2.internal systemd[1]: Starting nginx.service - The nginx HTTP and reverse proxy server...
Feb 28 10:18:17 ip-10-0-1-49.ec2.internal nginx[4579]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Feb 28 10:18:17 ip-10-0-1-49.ec2.internal nginx[4579]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Feb 28 10:18:17 ip-10-0-1-49.ec2.internal systemd[1]: Started nginx.service - The nginx HTTP and reverse proxy server.
[ec2-user@ip-10-0-1-49 ~]$
```

## Machines serveurs (Sous-réseau Privé)

1. Suivre la même procédure, mais placer les instances dans le sous-réseau privé et Désactiver l'IP publique. (Fait le même processus pour les deux machines et bien attribuer chacun dans le réseaux respectif)

☰ EC2 > Instances > Lancer une instance

### Lancer une instance Informations

Amazon EC2 vous permet de créer des machines virtuelles, ou des instances, qui s'exécutent sur le Cloud AWS. Démarrez rapidement en suivant les étapes simples indiquées ci-dessous.

#### Nom et balises Informations

Nom

tp-aws-web-2

Ajouter des balises supplémentaires

#### ▼ Images d'applications et de systèmes d'exploitation (Amazon Machine Image) Informations

Une AMI est un modèle contenant la configuration logicielle (système d'exploitation, serveur d'applications et applications) requise pour lancer votre instance. Parcourez ou recherchez des AMI si vous ne trouvez pas ce que vous recherchez ci-dessous.

🔍 Effectuer une recherche dans notre catalogue complet, qui comprend des milliers d'images d'applications et de systèmes d'exploitation

Récentes

Démarrage rapide



Explorer plus d'AMI

Y compris les AMI d'AWS, de Marketplace et de la communauté

#### ▼ Paire de clés (connexion) Informations

Vous pouvez utiliser une paire de clés pour vous connecter en toute sécurité à votre instance. Assurez-vous d'avoir accès à la paire de clés sélectionnée avant de lancer l'instance.

Nom de la paire de clés - *obligatoire*

tp-aws-web-2

⟳ Crée une paire de clés

#### ▼ Paramètres réseau Informations

VPC - *obligatoire* Informations

vpc-0fded5e5175e3b5cb (tp-aws-vpc-public)  
10.0.0.0/16



Sous-réseau Informations

subnet-0851ced2d7dd413e3 tp-aws-prive-subnet3  
VPC: vpc-0fded5e5175e3b5cb Propriétaire: 590332306376 Zone de disponibilité: us-east-1a  
Type de zone: Zone de disponibilité Adresses IP disponibles: 251 CIDR: 10.0.3.0/24



⟳ Crée un nouveau sous-réseau

Attribuer automatiquement l'adresse IP publique Informations

Désactiver

#### Amazon Machine Image (AMI)

AMI Amazon Linux 2023	Éligible à l'offre gratuite
ami-05b10e08d247fb927 (64 bits (x86), uefi-preferred) / ami-0f37c4a1ba152af46 (64 bits (Arm), uefi) Virtualisation: hvm ENA activé: true Type de périphérique racine: ebs	▼

#### Description

Amazon Linux 2023 est un système d'exploitation moderne basé sur Linux, à usage général et offrant cinq ans de support garanti. Optimisé pour AWS, il est conçu afin de fournir un environnement d'exécution sécurisé, stable et à hautes performances pour le développement et l'exécution de vos applications cloud.

Amazon Linux 2023 AMI 2023.6.20250218.2 x86\_64 HVM kernel-6.1

Architecture	Mode de démarrage	ID AMI	Nom d'utilisateur	Informations
64 bits (x86)	uefi-preferred	ami-05b10e08d247fb927	ec2-user	Fournisseur vérifié

#### ▼ Type d'instance [Informations](#) | [Obtenez des conseils](#)

##### Type d'instance

t2.micro	Éligible à l'offre gratuite
Famille: t2 1 vCPU 1 Gio Mémoire Génération actuelle: true	▼
À la demande Windows base tarification: 0.0162 USD per Hour	▼
À la demande Ubuntu Pro base tarification: 0.0134 USD per Hour	▼
À la demande SUSE base tarification: 0.0116 USD per Hour	▼
À la demande RHEL base tarification: 0.026 USD per Hour	À la demande Linux base tarification: 0.0116 USD per Hour

Toutes les générations

[Comparer les types d'instance](#)

#### ▼ Paire de clés (connexion) [Informations](#)

Vous pouvez utiliser une paire de clés pour vous connecter en toute sécurité à votre instance. Assurez-vous d'avoir accès à la paire de clés sélectionnée avant de lancer l'instance.

##### Nom de la paire de clés - *obligatoire*

tp-aws-web-1

[Créer une paire de clés](#)

#### ▼ Paramètres réseau [Informations](#)

##### VPC - *obligatoire* | [Informations](#)

vpc-0fded5e5175e3b5cb (tp-aws-vpc-public)  
10.0.0.0/16



##### Sous-réseau | [Informations](#)

subnet-03c34795627b647a7 tp-aws-prive-subnet-2  
VPC: vpc-0fded5e5175e3b5cb Propriétaire: 590332306376 Zone de disponibilité: us-east-1a  
Type de zone: Zone de disponibilité Adresses IP disponibles: 251 CIDR: 10.0.2.0/24

[Créer un nouveau sous-réseau](#)

##### Attribuer automatiquement l'adresse IP publique | [Informations](#)

Désactiver

2. Ajouter une règle de sécurité pour permettre uniquement SSH depuis le VPC.

3. Ajouter une règle de sécurité pour permettre uniquement HTTP depuis le VPC.

Nom du groupe de sécurité - *obligatoire*

Ce groupe de sécurité sera ajouté à toutes les interfaces réseau. Le nom ne peut pas être modifié après la création du groupe de sécurité. La longueur maximale est de 255 caractères. Caractères valides : a-z, A-Z, 0-9, espaces et \_- :/() #,@[]+= & ; ! \$\*

Description - *obligatoire* | Informations

Règles entrantes des groupes de sécurité

▼ Règle de groupe de sécurité 1 (TCP, 22, 0.0.0.0/0) Supprimer

Type   Informations	Protocole   Informations	Plage de ports   Informations
ssh	TCP	22
Type de source   Informations	Source   Informations	Description - <i>facultatif</i>   Informations
N'importe où	<input type="text" value="0.0.0.0"/> Ajouter une adresse CIDR, une liste de préfixe	par exemple, SSH pour le bureau de l'administrateur
	X	

▼ Règle de groupe de sécurité 2 (TCP, 80, 0.0.0.0/0) Supprimer

Type   Informations	Protocole   Informations	Plage de ports   Informations
HTTP	TCP	80
Type de source   Informations	Source   Informations	Description - <i>facultatif</i>   Informations
N'importe où	<input type="text" value="0.0.0.0"/> Ajouter une adresse CIDR, une liste de préfixe	par exemple, SSH pour le bureau de l'administrateur
	X	

#### 4. Ajouter un script pour installer httpd et modifier l'index.

##### Données utilisateur - *facultatif* | Informations

Chargez un fichier contenant vos données utilisateur ou saisissez-les dans le champ.

↑ Choisir un fichier

```
#!/bin/bash
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "<h1>10.0.2.1</h1>" | tee /var/www/html/index.html
systemctl restart httpd
```

5. Lancer
6. Depuis le conteneur admin accéder aux conteneurs web 1 et web 2 (bien penser à télécharger les deux paire de clé des serveurs sur le conteneur admin pour pouvoir accéder au conteneur)

## Connectez-vous à l'instance Informations

Connectez-vous à votre instance à i-0d8a466d11e007e27 (tp-aws-web-2) l'aide de l'une de ces options

EC2 Instance Connect
Session Manager
Client SSH
EC2 Serial Console

---

**ID d'instance**

[i-0d8a466d11e007e27 \(tp-aws-web-2\)](#)

1. Ouvrez un client SSH.

2. Recherchez votre fichier de clé privée. La clé utilisée pour lancer cette instance est tp-aws-web-2.pem

3. Exécuter, si nécessaire, cette commande pour vous assurer que votre clé n'est pas visible publiquement.  
 `chmod 400 "tp-aws-web-2.pem"`

4. Connectez-vous à votre instance à l'aide de son IP privée :  
 `10.0.3.120`

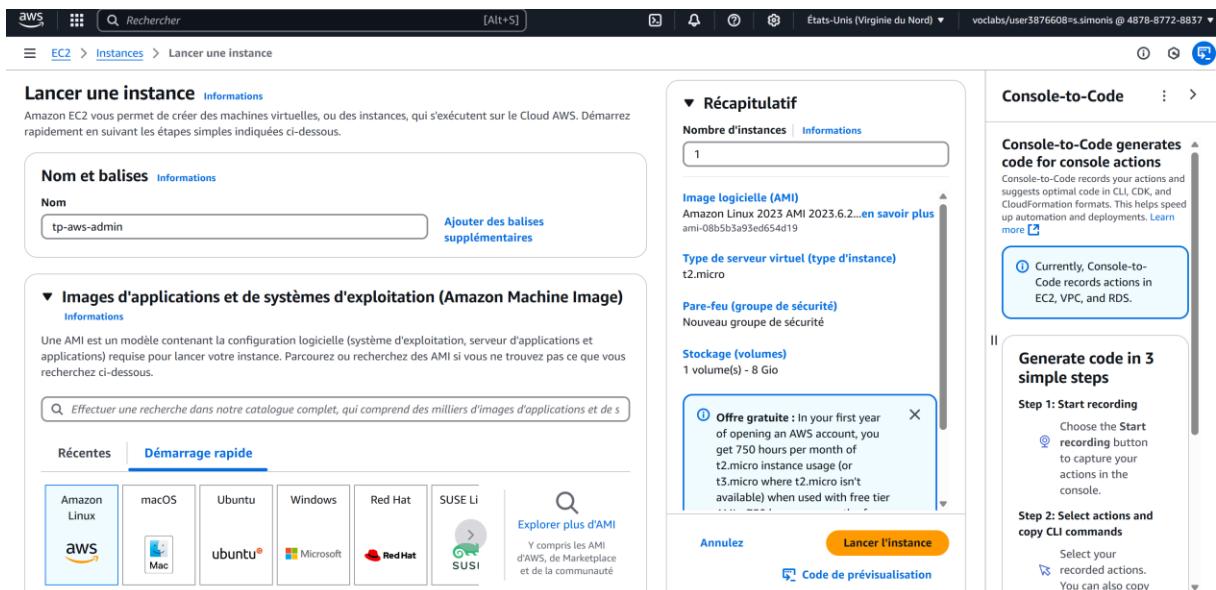
Exemple :

`ssh -i "tp-aws-web-2.pem" ec2-user@10.0.3.120`

ⓘ **Remarque :** Dans la plupart des cas, le nom d'utilisateur deviné est correct. Cependant, lisez les instructions de sécurité et vérifiez que l'utilisateur deviné est correct.

## Machines admin (Sous-réseau Public)

1. Suivre la même procédure, mais placer l'instances dans le sous-réseau public et activer l'IP publique.



The screenshot shows the AWS Cloud9 interface. On the left, there's a navigation bar with 'aws' logo, search bar, and 'Instances' selected. The main area has tabs for 'Lancer une instance' (selected), 'Images d'applications et de systèmes d'exploitation (Amazon Machine Image)', and 'Récapitulatif'. Under 'Lancer une instance', fields include 'Nom' (tp-aws-admin) and 'Image logicielle (AMI)' (Amazon Linux 2023 AMI 2023.6.2...). The 'Console-to-Code' sidebar on the right is open, showing steps for generating code, recording actions in EC2, VPC, and RDS, and a preview of the generated code.

## Amazon Machine Image (AMI)

AMI Amazon Linux 2023	Éligible à l'offre gratuite
ami-05b10e08d247fb927 (64 bits (x86), uefi-preferred) / ami-0f37c4a1ba152af46 (64 bits (Arm), uefi)	
Virtualisation: hvm ENA activé: true Type de périphérique racine: ebs	

### Description

Amazon Linux 2023 est un système d'exploitation moderne basé sur Linux, à usage général et offrant cinq ans de support garanti. Optimisé pour AWS, il est conçu afin de fournir un environnement d'exécution sécurisé, stable et à hautes performances pour le développement et l'exécution de vos applications cloud.

Amazon Linux 2023 AMI 2023.6.20250218.2 x86\_64 HVM kernel-6.1

Architecture	Mode de démarrage	ID AMI	Nom d'utilisateur	
64 bits (x86)	uefi-preferred	ami-05b10e08d247fb927	ec2-user	Fournisseur vérifié

### ▼ Type d'instance [Informations](#) | [Obtenez des conseils](#)

#### Type d'instance

t2.micro	Éligible à l'offre gratuite
Famille: t2 1 vCPU 1 Gio Mémoire Génération actuelle: true	
À la demande Windows base tarification: 0.0162 USD per Hour	
À la demande Ubuntu Pro base tarification: 0.0134 USD per Hour	
À la demande SUSE base tarification: 0.0116 USD per Hour	
À la demande RHEL base tarification: 0.026 USD per Hour	À la demande Linux base tarification: 0.0116 USD per Hour

Toutes les générations

[Comparer les types d'instance](#)

## Créer une paire de clés



### Nom de la paire de clés

Les paires de clés vous permettent de vous connecter à votre instance en toute sécurité.

Admin

La longueur maximale du nom est de 255 caractères ASCII. Il ne peut pas inclure d'espaces avant ou après.

### Type de paire de clés

RSA

Paire de clés privée et publique chiffrée  
RSA

ED25519

Paire de clés privée et publique chiffrée  
ED25519

### Format de fichier de clé privée

.pem

À utiliser avec OpenSSH

.ppk

À utiliser avec PuTTY

**⚠️** Lorsque vous y êtes invité, stockez la paire de clés dans un emplacement sécurisé et accessible sur votre ordinateur. **Vous en aurez besoin ultérieurement pour vous connecter à votre instance.** [En savoir plus](#)

[Annulez](#)

[Créer une paire de clés](#)

## ▼ Paramètres réseau [Informations](#)

### VPC - obligatoire | [Informations](#)

vpc-0fded5e5175e3b5cb (tp-aws-vpc-public)  
10.0.0.0/16



### Sous-réseau | [Informations](#)

subnet-0b165441a0be2b230 tp-aws-public-subnet1  
VPC: vpc-0fded5e5175e3b5cb Propriétaire: 590332306376 Zone de disponibilité: us-east-1  
Type de zone: Zone de disponibilité Adresses IP disponibles: 250 CIDR: 10.0.1.0/24

[Créer un nouveau sous-réseau](#)

### Attribuer automatiquement l'adresse IP publique | [Informations](#)

Activer



1. Ajouter une règle de sécurité pour permettre le SSH (22 depuis 0.0.0.0/0).
2. Ajouter une règle de sécurité pour permettre le UDP (51820 depuis 0.0.0.0/0).

sécurité. La longueur maximale est de 255 caractères. Caractères valides : a-z, A-Z, 0-9, espaces et .-:/() #,@[]+= & ; {} ! \$\*

### Description - obligatoire | [Informations](#)

launch-wizard-1 created 2025-03-17T14:09:33.286Z

### Règles entrantes des groupes de sécurité

- ▼ Règle de groupe de sécurité 1 (TCP, 22)

[Supprimer](#)

#### Type | [Informations](#)

ssh

#### Protocole | [Informations](#)

TCP

#### Plage de ports | [Informations](#)

22

#### Type de source | [Informations](#)

Personnalisé

#### Source | [Informations](#)

Ajouter une adresse CIDR, une

#### Description - facultatif | [Informations](#)

par exemple, SSH pour le bureau

- ▼ Règle de groupe de sécurité 2 (UDP, 51820)

[Supprimer](#)

#### Type | [Informations](#)

UDP personnalisé

#### Protocole | [Informations](#)

UDP

#### Plage de ports | [Informations](#)

51820

#### Type de source | [Informations](#)

Personnalisé

#### Source | [Informations](#)

Ajouter une adresse CIDR, une

#### Description - facultatif | [Informations](#)

par exemple, SSH pour le bureau

[Ajouter une règle de groupe de sécurité](#)

## Vérification

1. Modifications apportées aux fichiers de configuration de HAProxy :

```
[ec2-user@ip-10-0-1-49 ~]$ vim /etc/nginx/conf.d/load_balancer.conf
upstream backend {
    server 10.0.2.127;
    server 10.0.3.120;
}

server {
    listen 80;
    location / {
        proxy_pass http://backend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

```
[ec2-user@ip-10-0-1-49 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-1-49 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
   Active: active (running) since Fri 2025-02-28 11:08:52 UTC; 5s ago
     Process: 27686 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
     Process: 27687 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
     Process: 27688 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
      Main PID: 27689 (nginx)
        Tasks: 2 (limit: 1111)
       Memory: 2.5M
          CPU: 42ms
         CGroup: /system.slice/nginx.service
                 └─27689 "nginx: master process /usr/sbin/nginx"
                   ├─27690 "nginx: worker process"

Feb 28 11:08:52 ip-10-0-1-49.ec2.internal systemd[1]: Starting nginx.service - The nginx HTTP and reverse proxy server.
Feb 28 11:08:52 ip-10-0-1-49.ec2.internal nginx[27687]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Feb 28 11:08:52 ip-10-0-1-49.ec2.internal nginx[27687]: nginx: configuration file /etc/nginx/nginx.conf test is success
Feb 28 11:08:52 ip-10-0-1-49.ec2.internal systemd[1]: Started nginx.service - The nginx HTTP and reverse proxy server.
```

## 2. Modifications apportées aux fichiers de HTML des serveurs WEB :

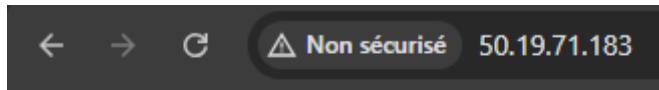
```
[ec2-user@ip-10-0-2-127 ~]$ cat /var/www/html/index.html
<h1>10.0.2.127</h1>
[ec2-user@ip-10-0-2-127 ~]$ sudo systemctl restart httpd
```

```
[ec2-user@ip-10-0-3-120 ~]$ sudo cat /var/www/html/index.html
<h1>10.0.3.120</h1>
[ec2-user@ip-10-0-3-120 ~]$ sudo systemctl restart httpd
```

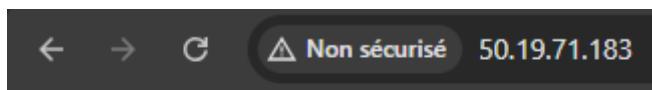
## 3. Rapport de Test :

```
[ec2-user@ip-10-0-1-49 ~]$ watch -n1 curl -s http://127.0.0.1
```

Every 1.0s: curl -s http://127.0.0.1	ip-10-0-1-49.ec2.internal: Fri Feb 28 11:14:18 2025
<h1>10.0.2.127</h1>	
Every 1.0s: curl -s http://127.0.0.1	ip-10-0-1-49.ec2.internal: Fri Feb 28 11:14:13 2025
<h1>10.0.3.120</h1>	



**10.0.2.127**



**10.0.3.120**

## Installation de Wireguard (VPN)

WireGuard est un VPN performant, reposant sur un chiffrement moderne avec l'algorithme ChaCha20, plus efficace que l'AES sur certains processeurs. Il se distingue par une configuration simple et claire, ne laisse aucun port ouvert lorsqu'il est inactif et utilise des clés cryptographiques modernes, offrant une sécurité supérieure à certains certificats. Nous l'utiliserons pour établir un VPN site-to-site, assurant la communication entre l'infrastructure locale (Proxmox) et Hyphydre (AWS)

### 1. Lancer et connecter vous en ssh a votre conteneur admin .

```
PS C:\Users\steve\Downloads> ssh -i "WOI.pem" ec2-user@98.80.139.132
The authenticity of host '98.80.139.132 (98.80.139.132)' can't be established
.
ED25519 key fingerprint is SHA256:35/LBPqNKEAEUZMHE9+orWgDwT+H6pFJshzGe5GT9nQ
.
This host key is known by the following other names/addresses:
  C:\Users\steve/.ssh/known_hosts:24: 3.215.179.154
  C:\Users\steve/.ssh/known_hosts:26: 44.200.221.226
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

### 2.1 Installer Wireguard sur la machine Admin AWS

```
[ec2-user@ip-10-0-20-154 ~]$ sudo yum update && sudo yum install wireguard -y
Last metadata expiration check: 3:13:27 ago on Mon Mar 17 11:11:09 2025.
=====
WARNING:
A newer release of "Amazon Linux" is available.

Available Versions:
Version 2023.6.20250303:
Run the following command to upgrade to 2023.6.20250303:
dnf upgrade --releasever=2023.6.20250303

Release notes:
https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.6.20250303.html
=====

Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 3:13:28 ago on Mon Mar 17 11:11:09 2025.
No match for argument: wireguard
Error: Unable to find a match: wireguard
[ec2-user@ip-10-0-20-154 ~]$ [ec2-user@ip-10-0-20-154 ~]$ sudo dnf install wireguard-tools -y
Last metadata expiration check: 3:14:22 ago on Mon Mar 17 11:11:09 2025.
Dependencies resolved.
=====
Package          Architecture Version      Repository      Size
=====
wireguard-tools x86_64       0.0.2-1     epel           10 k
=====
Total Download Size: 10 k
=====
[ec2-user@ip-10-0-20-154 ~]$
```

### 2.1 Génération des clés publiques et privées

.privatekey = clé privée du serveur AWS.

.publickey = clé publique du serveur AWS .

```
[ec2-user@ip-10-0-20-154 ~]$ wg genkey | tee privatekey | wg pubkey > publickey  
[ec2-user@ip-10-0-20-154 ~]$
```

Récupère les clés avec :

cat privatekey

cat publickey

### 3.1 Installer Wireguard sur la machine Admin Promox

```
root@lover:~# apt update && apt install wireguard -y
```

### 3.2 Génération des clés publiques et privées

.privatekey = clé privée du serveur Promox.

.publickey = clé publique du serveur Promox .

```
root@lover:~# wg genkey | tee privatekey | wg pubkey > publickey
```

Récupère les clés avec :

cat privatekey

cat publickey

### 4.1 Configuration de WireGuard sur l'instance AWS :

Sudo nano /etc/wireguard/wg0.conf

Édite le fichier de configuration sur AWS :

```
GNU nano 5.8                               /etc/wireguard/wg0.conf                         Modified  
[Interface]  
PrivateKey = 2P0Dc2yV3VyWL AqTeIS7HV oXnAvmOyW52X6oqPwqH1g=  
Address = 10.10.10.1/24  
ListenPort = 51820  
  
[Peer]  
PublicKey = UtYUplbDfT+17AUphPbVe8zUGp98Wg2476KheREbZVs=  
AllowedIPs = 10.10.10.2/32  
Endpoint = 192.168.118.136:51820  
PersistentKeepalive = 25
```

**Explication :**

- **Address** → IP WireGuard attribuée à AWS.
- **ListenPort** → Port d'écoute de WireGuard.
- **PrivateKey** → Clé privée du peer (AWS).
- **PublicKey** → Clé publique du peer (Proxmox).
- **AllowedIPs** → Sous-réseaux accessibles via ce VPN.

- **Endpoint** → Adresse publique de Proxmox.
- **PersistentKeepalive** → Maintient le tunnel actif (utile pour NAT).

#### Démarrer WireGuard sur AWS:

```
sudo systemctl enable wg-quick@wg0
```

```
sudo systemctl start wg-quick@wg0
```

```
[ec2-user@ip-10-0-20-154 ~]$ sudo systemctl enable wg-quick@wg0
sudo systemctl start wg-quick@wg0
Created symlink /etc/systemd/system/multi-user.target.wants/wg-quick@wg0.service → /usr/lib/systemd/system/wg-quick@.service.
[ec2-user@ip-10-0-20-154 ~]$
```

#### Ajouter du routage pour permettre la communication entre les sous-réseaux

```
[ec2-user@ip-10-0-20-154 ~]$ echo "net.ipv4.ip_forward = 1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.ip_forward = 1
```

```
[ec2-user@ip-10-0-20-154 ~]$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

#### 4.2 Configuration de WireGuard sur l'instance Promox:

```
root@lover:~# sudo nano /etc/wireguard/wg0.conf
```

Édite le fichier de configuration sur **Promox** :

```
GNU nano 7.2                               /etc/wireguard/wg0.conf *
[Interface]
PrivateKey = CE3InIERXoGXRm270s174ZFgJybCNbNnq6RzgG1/rm8=
Address = 10.10.10.2/24
ListenPort = 51820

[Peer]
PublicKey = Jd8r/6CjHbr3Ihcf/UJxYoD4S4HEoL97aXBVCiMU3A=
AllowedIPs = 10.10.10.1/32, 192.168.1.0/24
Endpoint = 98.80.139.132:51820
PersistentKeepalive = 25
```

#### Explication :

- **Address** → IP WireGuard attribuée à Promox.
- **ListenPort** → Port d'écoute de WireGuard.
- **PrivateKey** → Clé publique du peer (Proxmox).
- **PublicKey** → Clé publique du peer (AWS).
- **AllowedIPs** → Sous-réseaux accessibles via ce VPN.

- **Endpoint** → Adresse publique de AWS.
- **PersistentKeepalive** → Maintient le tunnel actif (utile pour NAT).

### Démarrer WireGuard sur Proxmox:

```
sudo systemctl enable wg-quick@wg0
```

```
sudo systemctl start wg-quick@wg0
```

```
root@lover:~# systemctl enable wg-quick@wg0
Created symlink /etc/systemd/system/multi-user.target.wants/wg-quick@wg0.service → /lib/systemd/system/wg-quick@.service.
root@lover:~# systemctl start wg-quick@wg0
```

```
sudo systemctl status wg-quick@wg0
```

```
root@lover:~# systemctl status wg-quick@wg0
● wg-quick@wg0.service - WireGuard via wg-quick(8) for wg0
   Loaded: loaded (/lib/systemd/system/wg-quick@.service; enabled; preset:>)
   Active: active (exited) since Tue 2025-03-18 14:49:18 CET; 12s ago
     Docs: man:wg-quick(8)
           man:wg(8)
           https://www.wireguard.com/
           https://www.wireguard.com/quickstart/
           https://git.zx2c4.com/wireguard-tools/about/src/man/wg-quick.8
           https://git.zx2c4.com/wireguard-tools/about/src/man/wg.8
   Process: 7359 ExecStart=/usr/bin/wg-quick up wg0 (code=exited, status=0/0)
   Main PID: 7359 (code=exited, status=0/SUCCESS)
     CPU: 41ms

Mar 18 14:49:18 lover systemd[1]: Starting wg-quick@wg0.service - WireGuard >
Mar 18 14:49:18 lover wg-quick[7359]: [#] ip link add wg0 type wireguard
Mar 18 14:49:18 lover wg-quick[7359]: [#] wg setconf wg0 /dev/fd/63
Mar 18 14:49:18 lover wg-quick[7359]: [#] ip -4 address add 10.10.10.2/24 dev
Mar 18 14:49:18 lover wg-quick[7359]: [#] ip link set mtu 1420 up dev wg0
Mar 18 14:49:18 lover wg-quick[7359]: [#] ip -4 route add 192.168.1.0/24 dev
Mar 18 14:49:18 lover systemd[1]: Finished wg-quick@wg0.service - WireGuard >
```

### Ajouter du routage pour permettre la communication entre les sous-réseaux

```
root@lover:~# echo "net.ipv4.ip_forward = 1" | tee -a /etc/sysctl.conf
net.ipv4.ip_forward = 1
root@lover:~# sudo sysctl -p
-bash: sudo: command not found
root@lover:~# sysctl -p
net.ipv4.ip_forward = 1
```

### Ajouter une route pour tout le réseau AWS :

```
sudo ip route add 192.168.1.0/24 via 10.10.10.1 dev wg0
```

Pour permettre la communication de l'infrastructure avec celle de l'hybride

```

root@lover:~# ping 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
64 bytes from 10.10.10.1: icmp_seq=1 ttl=127 time=82.2 ms
64 bytes from 10.10.10.1: icmp_seq=2 ttl=127 time=84.0 ms
^C
--- 10.10.10.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 82.183/83.108/84.034/0.925 ms
root@lover:~#

```

## 5.SSH via VPN

Nous allons modifier les règles du groupe de sécurité de la machine afin de restreindre l'accès SSH uniquement au réseau du VPN.

- 1 Aller dans EC2 → Groupes de sécurité → votre groupe de sécurité relier à votre machine  
→ modifier les règles entrantes
- 2 Modifier la règle de sécurité pour permettre la connexion SSH via VPN (22 depuis ex : 10.10.10.0/24).

The screenshot shows the AWS CloudWatch Metrics interface. A single log entry is displayed:

```

{
    "logEvent": {
        "version": "0",
        "id": "d4a2a2f0-4a2c-4a2c-4a2c-4a2c4a2c4a2c",
        "timestamp": "2023-09-18T10:00:00Z",
        "logStreamName": "lambda-logs",
        "logGroup": "/aws/lambda/lambda-logs",
        "message": "{'@version': '1', '@tima...'}"
    }
}

```

- 3 Relancer votre instance admin sur AWS

3. Essayer de vous connecter avec votre pc ou autre à l'instance

```

PS C:\Users\steve\Downloads> ssh -i "WOI.pem" ec2-user@98.80.139.132
ssh: connect to host 98.80.139.132 port 22: Connection timed out

```

Essayez de vous connecter à la machine d'administration de Proxmox.

**⚠️** Avant de vous connecter à la machine AWS, assurez-vous de bien récupérer la clé

PEM que vous avez téléchargée depuis AWS et de la copier sur la machine Proxmox.

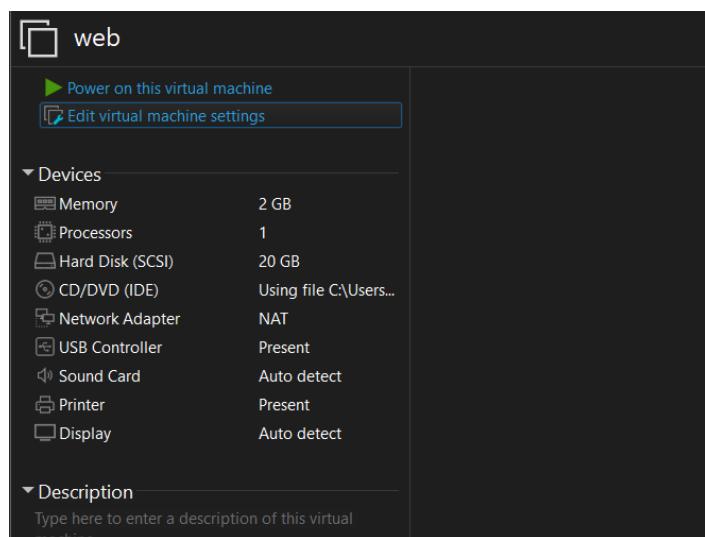
```
root@lover:~# ssh -i "MOI.pem" ec2-user@10.10.10.1
A newer release of "Amazon Linux" is available.
  Version 2023.6.20250303:
Run "/usr/bin/dnf check-release-update" for full release and version update info
      _#
     ~\_ #####_      Amazon Linux 2023
     ~~ \_#####\_
     ~~   \###|
     ~~     \#/ ___ https://aws.amazon.com/linux/amazon-linux-2023
     ~~       V~' '-->
     ~~         /
     ~~.~. /_/
     ~~ /_/
     _/m/`_
Last login: Mon Mar 17 15:17:49 2025 from 80.15.154.187
[ec2-user@ip-10-0-20-154 ~]$
```

## Infrastructure Web

### Prérequis

Afin de remplir les exigences du client, nous avons installé plusieurs serveurs qui auront chacun un rôle distinct. En effet, parmi ces serveurs, deux seront des serveurs web, un sera le serveur haproxy, un autre pour la PKI et le dernier sera le serveur de base de données.

Pour la mise en place de tous nos services nous avons choisis Debian comme OS. Voici un exemple de config pour nos machines :

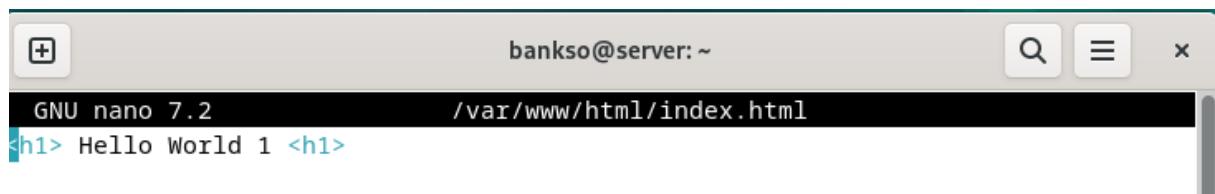


## Installation et Mise en place de serveurs Web

Une fois que notre machine Debian prête, nous allons installer notre service web c'est-à-dire Nginx :

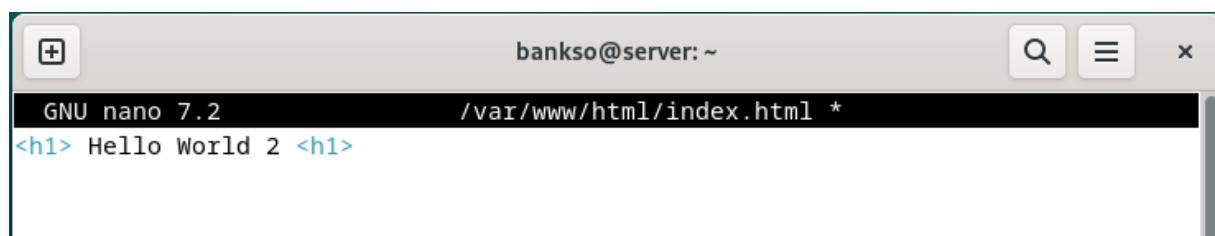
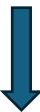
```
bankso@server:~$ sudo apt install nginx
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les NOUVEAUX paquets suivants seront installés :
  nginx
0 mis à jour, 1 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 528 ko dans les archives.
Après cette opération, 1 362 ko d'espace disque supplémentaires seront utilisés.
Réception de :1 http://deb.debian.org/debian bookworm/main amd64 nginx amd64 1.2
2.1-9+deb12u1 [528 kB]
528 ko réceptionnés en 0s (3 799 ko/s)
Sélection du paquet nginx précédemment désélectionné.
(Lecture de la base de données... 150086 fichiers et répertoires déjà installés.
)
Préparation du dépaquetage de .../nginx_1.22.1-9+deb12u1_amd64.deb ...
Dénpaquetage de nginx (1.22.1-9+deb12u1) ...
```

Après installation, nous modifierons son fichier html afin de faire la différence lors du load balancing



```
bankso@server: ~
GNU nano 7.2          /var/www/html/index.html
<h1> Hello World 1 <h1>
```

Nous ferons de même pour l'autre serveur



```
bankso@server: ~
GNU nano 7.2          /var/www/html/index.html *
<h1> Hello World 2 <h1>
```

Une fois cela fait nous passerons donc à l'installation de notre serveur pour le load balancing.  
Pour cela nous installerons le service haproxy :

```

bankso@haproxy:~$ sudo apt install haproxy
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Paquets suggérés :
  vim-haproxy haproxy-doc
Les NOUVEAUX paquets suivants seront installés :
  haproxy
0 mis à jour, 1 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 2 042 ko dans les archives.
Après cette opération, 4 324 ko d'espace disque supplémentaires seront utilisés
Réception de :1 http://deb.debian.org/debian bookworm/main amd64 haproxy amd64
  .6.12-1+deb12u1 [2 042 kB]
2 042 ko réceptionnés en 0s (6 435 ko/s)
Sélection du paquet haproxy précédemment désélectionné.
(Lecture de la base de données... 149993 fichiers et répertoires déjà installés
)
Préparation du dépaquetage de .../haproxy_2.6.12-1+deb12u1_amd64.deb ...
--2023-09-11 14:44:21-- /etc/haproxy/haproxy.cfg

```

Maintenant nous modifierons le fichier de conf haproxy.cfg afin de préciser la redondance sur les ip de nos serveurs web



```

bankso@haproxy: ~
/etc/haproxy/haproxy.cfg *

defaults
  log global
  mode http
  option httplog
  option dontlognull
  timeout connect 5000ms
  timeout client 50000ms
  timeout server 50000ms

frontend http_front
  bind *:80
  default_backend web_servers
#mes servers
backend web_servers
  balance roundrobin
  balance roundrobin
  server web1 192.168.2.128:80 check
  server web2 192.168.2.130:80 check

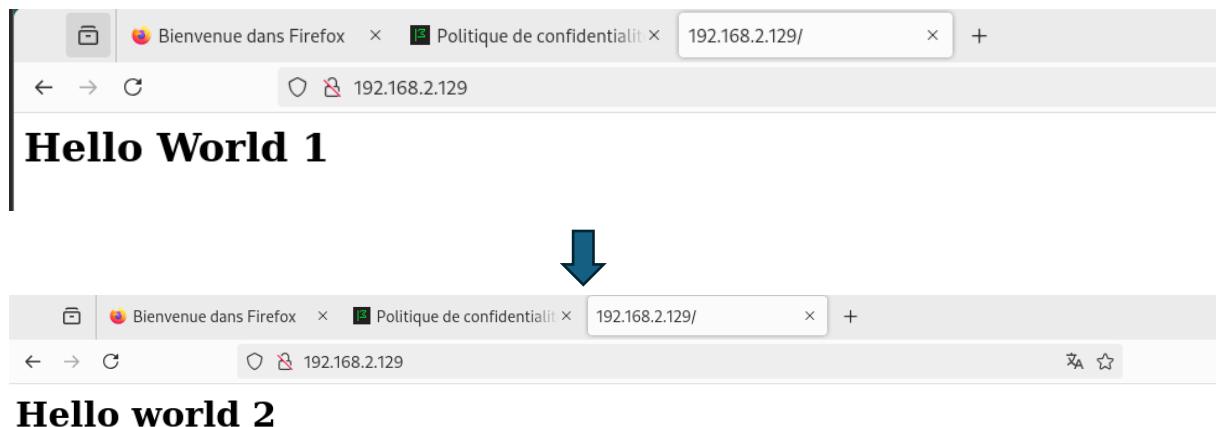
```

Nous rajouterons une partie dans le fichier de config qui nous servira à voir les stats de notre load balancing

```
#afficher les stats de mes servers
listen stats
    bind *:80
    stats enable
    stats uri /haproxy_stats
    stats refresh 10s
```

## Test

Pour tester notre load balancing nous mettrons l'adresse ip de notre serveur haproxy sur un navigateur web. Nous sommes censés voir les deux pages web distinctes sur la même adresse ip ce qui prouvera la redondance.



## Installation d'un système de base de données

Pour notre système de base de données, nous avons opté pour MariaDB pour ses performances et sa sécurité.

```
bankso@server:~$ sudo apt install mariadb-server -y
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  galera-4 gawk libcgifast-perl libcgipm-perl libconfiginifiles-perl
  libdbd-mariadb-perl libdbi-perl libfcgi-bin libfcgi-perl libfcgi0ldbl
  libhtml-template-perl libmariadb3 libsigsegv2 libtermreadkey-perl liburing2
  mariadb-client mariadb-client-core mariadb-common
  mariadb-plugin-provider-bzip2 mariadb-plugin-provider-lz4
  mariadb-plugin-provider-lzma mariadb-plugin-provider-lzo
  mariadb-plugin-provider-snappy mariadb-server-core mysql-common pv rsync
  socat
Paquets suggérés :
  gawk-doc libmldb-perl libnet-daemon-perl libsql-statement-perl
  libipc-sharedcache-perl mailx mariadb-test netcat-openbsd doc-base
  python3-braceexpand
Les NOUVEAUX paquets suivants seront installés :
  galera-4 gawk libcgifast-perl libcgipm-perl libconfiginifiles-perl
  libdbd-mariadb-perl libdbi-perl libfcgi-bin libfcgi-perl libfcgi0ldbl
```

```
bankso@server:~$ systemctl status mariadb
● mariadb.service - MariaDB 10.11.11 database server
    Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; preset: enabled)
    Active: active (running) since Thu 2025-03-20 22:38:44 CET; 14min ago
      Docs: man:mariadb(8)
             https://mariadb.com/kb/en/library/systemd/
        Main PID: 3669 (mariadb)
          Status: "Taking your SQL requests now..."
            Tasks: 9 (limit: 14792)
           Memory: 88.0M
              CPU: 1.507s
            CGroup: /system.slice/mariadb.service
                      └─3669 /usr/sbin/mariadb
```

Une fois que notre système de base de données a été fonctionnel, nous l'avons ensuite sécurisé

```
bankso@server:~$ sudo mysql_secure_installation
[sudo] Mot de passe de bankso :

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...
```

Une fois que tout cela a été fait, nous avons donc créé une base de donnée ainsi qu'un utilisateur test pour celle-ci

```
bankso@server:~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 41
Server version: 10.11.11-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE bankso;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> CREATE USER 'ezechiel'@'%' IDENTIFIED BY 'eze21'
-> GRANT ALL PRIVILEGES ON bankso.* TO 'ezechiel'@'%'
```

Voici le résultat de nos commandes

```
MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| bankso   |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0.001 sec)
```

```
MariaDB [(none)]> SELECT user, host FROM mysql.user;
+-----+-----+
| User      | Host     |
+-----+-----+
| ezechiel  | %       |
| mariadb.sys | localhost |
| mysql     | localhost |
| root      | localhost |
+-----+-----+
4 rows in set (0.002 sec)
```

# Installation d'un système SSL Offloader

Pour la mise en place de ce système nous avons utilisé notre serveur Haproxy où nous avons généré un certificat auto-signé.

```
bankso@haproxy:~$ sudo openssl req -x509 -newkey rsa:4096 -keyout /etc/haproxy/server.key -out /etc/haproxy/server.crt -days 365 -nodes
[sudo] Mot de passe de bankso :
-----  
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
-----  
Country Name (2 letter code) [AU]:fr  
State or Province Name (full name) [Some-State]:paris  
Locality Name (eg, city) []:paris  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:bankso  
Organizational Unit Name (eg, section) []:DSI  
Common Name (e.g. server FQDN or YOUR name) []:  
Email Address []:  
bankso@haproxy:~$ |
```

Nous avons ensuite modifié le fichier de conf de haproxy afin de lui préciser la redirection vers HTTPS et activer SSL par défaut

```
GNU nano 7.2                                     /etc/haproxy/haproxy.cfg *

ssl-default-bind-options no-sslv3
ssl-default-bind-ciphers ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256

defaults
    log global
    mode http
    option httplog
    option dontlognull
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms

# Afficher les stats de HAProxy
listen stats
    bind *:8080
    stats enable
    stats uri /haproxy_stats
    stats refresh 10s
    stats auth admin:password # Modifie ce mot de passe en prod

# Frontend HTTP --> Redirige vers HTTPS
frontend http_front
    bind *:80
    redirect scheme https if !{ ssl_fc }

# Frontend HTTPS avec SSL Offloading
frontend https_front
    bind *:443 ssl crt /etc/haproxy/haproxy.pem
    mode http
    option forwardfor
    http-request add-header X-Forwarded-Proto https
    default_backend web_servers

# Mes serveurs backend
backend web_servers
    balance roundrobin
    server web1 192.168.2.128:80 check
```

## Test

Une fois toutes nos configurations réussies nous avons testé si la redirection vers https fonctionnait comme il fallait :

```
bankso@haproxy:~$ curl -I https://192.168.2.129 --insecure
HTTP/1.1 200 OK
server: nginx/1.22.1
date: Thu, 20 Mar 2025 23:17:58 GMT
content-type: text/html
content-length: 24
last-modified: Thu, 20 Mar 2025 12:10:08 GMT
etag: "67dc05a0-18"
accept-ranges: bytes
```

## Installation et Mise en place d'une PKI

Comme demandé par le client, une autorité de certification devait être faites et nous avons choisi d'utilisé EasyRSA

```
[bankso@ph1:~] $ sudo apt update && sudo apt install easy-rsa -y
[sudo] Mot de passe de bankso :
Atteint :1 http://security.debian.org/debian-security bookworm-security InRelease
Atteint :2 http://deb.debian.org/debian bookworm InRelease
Atteint :3 http://deb.debian.org/debian bookworm-updates InRelease
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Tous les paquets sont à jour.
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  liblccid opencsc opensc-pkcs11 pccsd
Paquets suggérés :
  pmciautils
Les NOUVEAUX paquets suivants seront installés :
  easy-rsa liblccid opencsc-pkcs11 pccsd
0 mis à jour, 5 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 1 800 ko dans les archives.
Après cette opération, 5 657 ko d'espace disque supplémentaires seront utilisés.
Réception de :1 http://deb.debian.org/debian bookworm/main amd64 liblccid amd64 1.5.2-1 [367 kB]
Réception de :2 http://deb.debian.org/debian bookworm/main amd64 pccsd amd64 1.9.9-2 [89.7 kB]
Réception de :3 http://deb.debian.org/debian bookworm/main amd64 easy-rsa all 3.1.0-1 [54.8 kB]
Réception de :4 http://deb.debian.org/debian bookworm/main amd64 opencsc-pkcs11 amd64 0.23.0-0.3+deb12u2 [917 kB]
Réception de :5 http://deb.debian.org/debian bookworm/main amd64 opencsc amd64 0.23.0-0.3+deb12u2 [372 kB]
1 800 ko réceptionnés en 0s (13,1 Mo/s)
Sélection du paquet liblccid précédemment désélectionné.
(Lecture de la base de données... 151960 fichiers et répertoires déjà installés.)
```

## Création de la CA

## Génération d'un certificat pour HAProxy

Une fois que notre certificat a bien été créé, nous l'avons signé

```
bankso@pki:~/easy-rsa$ cd ~/easy-rsa
./easyrsa sign-req server haproxy-server
* Notice:
Using Easy-RSA configuration from: /home/bankso/easy-rsa/pki/vars

* Notice:
Using SSL: openssl OpenSSL 3.0.15 3 Sep 2024 (Library: OpenSSL 3.0.15 3 Sep 2024)

You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a server certificate for 825 days:

subject=
    commonName          = eze

Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes

Using configuration from /home/bankso/easy-rsa/pki/8c69468e/temp.870be25c
Enter pass phrase for /home/bankso/easy-rsa/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName      :ASN.1 12:'eze'
Certificate is to be certified until Jun 24 00:11:50 2027 GMT (825 days)

Write out database with 1 new entries
Database updated

* Notice:
Certificate created at: /home/bankso/easy-rsa/pki/issued/haproxy-server.crt
```

Une fois que notre certificat a été signé, nous avons transférer le certificat, la clé privée et le certificat de l'autorité de certification vers le serveur HAProxy

```

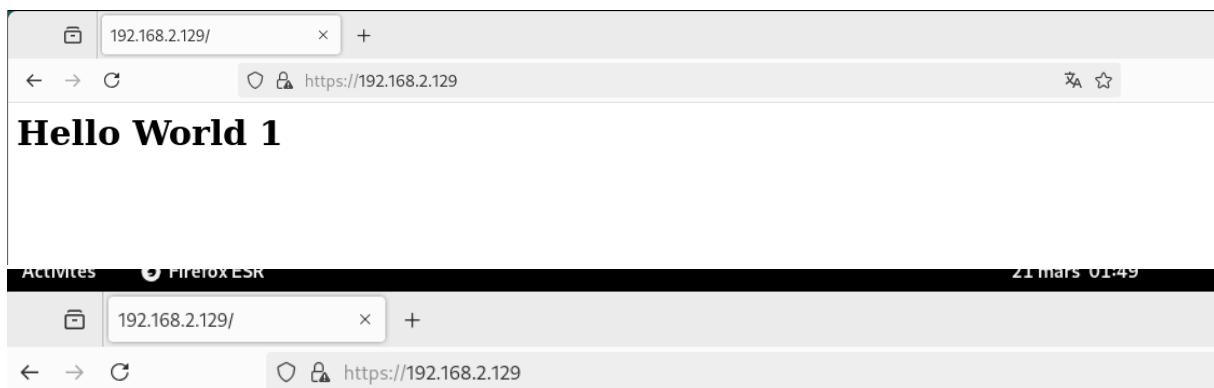
bankso@pki:~/easy-rsa$ scp ~/easy-rsa/pki/issued/haproxy-server.crt haproxy:/tmp/
scp ~/easy-rsa/pki/private/haproxy-server.key haproxy:/tmp/
scp ~/easy-rsa/pki/ca.crt haproxy:/tmp/

ssh -t haproxy "sudo mv /tmp/haproxy-server.crt /etc/haproxy/"
ssh -t haproxy "sudo mv /tmp/haproxy-server.key /etc/haproxy/"
ssh -t haproxy "sudo mv /tmp/ca.crt /etc/haproxy/"
The authenticity of host 'haproxy (192.168.2.129)' can't be established.
ED25519 key fingerprint is SHA256:hrCilmWRAsd00q5BcCHTPosUyijpnQ7OTxNwn4L8b0M.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'haproxy' (ED25519) to the list of known hosts.
bankso@haproxy's password:
haproxy-server.crt
bankso@haproxy's password:
haproxy-server.key
bankso@haproxy's password:
ca.crt
bankso@haproxy's password:
[sudo] Mot de passe de bankso :
Connection to haproxy closed.
bankso@haproxy's password:
[sudo] Mot de passe de bankso :
Connection to haproxy closed.
bankso@haproxy's password:
[sudo] Mot de passe de bankso :
Connection to haproxy closed.
bankso@pki:~/easy-rsa$ |

```

## Test

Apres la mise en place de tous nos certificats nous pouvons tester la connexion en https :



# Installation et Mise en place d'un système de détection d'intrusion

Pour cette partie nous avons choisi d'utiliser Suricata

```
bankso@haproxy:~$ sudo apt update && sudo apt install -y suricata
Atteint :1 http://security.debian.org/debian-security bookworm-security InRelease
Atteint :2 http://deb.debian.org/debian bookworm InRelease
Atteint :3 http://deb.debian.org/debian bookworm-updates InRelease
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Tous les paquets sont à jour.
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  libevent-core-2.1-7 libevent-pthreads-2.1-7 libhiredis0.14 libhttp2 libhyperscan5 libluajit-5.1-2
  libluajit-5.1-common libnet1 libnetfilter-log1 libnetfilter-queue1 libpcre3 python3-yaml suricata-update
Paquets suggérés :
  libtcmalloc-minimal4
Paquets recommandés :
  snort-rules-default
```

Une fois l'isntallation terminé, nous devrons vérifier notre interface réseau car nous en auront besoin pour nos configs. Pour cela nous ferons un **Ip a**

```
bankso@haproxy:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host noprefixroute
            valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:64:df:44 brd ff:ff:ff:ff:ff:ff
        altname enp2s1
        inet 192.168.2.129/24 brd 192.168.2.255 scope global dynamic noprefixroute ens33
            valid_lft 1021sec preferred_lft 1021sec
        inet6 fe80::20c:29ff:fe64:df44/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
bankso@haproxy:~$ |
```

Comme nous pouvons le voir ici, notre interface est ens33. Ensuite nous avons modifié le fichier de conf de Suricata

```
GNU nano 7.2                                     /etc/suricata/suricata.yaml *
# Linux high speed capture support
af-packet:
  - interface: ens33
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99
    # Default AF_PACKET cluster type. AF_PACKET can load balance per flow or per hash.
    # This is only supported for Linux kernel > 3.1
    # possible value are:
    # * cluster_flow: all packets of a given flow are sent to the same socket
    # * cluster_cpu: all packets treated in kernel by a CPU are sent to the same socket
    # * cluster_qm: all packets linked by network card to a RSS queue are sent to the same
    #   socket. Requires at least Linux 3.14.
    # * cluster_ebpf: eBPF file load balancing. See doc/userguide/capture-hardware/ebpf-xdp.rst for
    #   more info.
    # Recommended modes are cluster_flow on most boxes and cluster_cpu or cluster_qm on system
    # with capture card using RSS (requires cpu affinity tuning and system IRQ tuning)
    cluster-type: cluster_flow
    # In some fragmentation cases, the hash can not be computed. If "defrag" is set
    # to yes, the kernel will do the needed defragmentation before sending the packets.
    defrag: yes
    # To use the ring feature of AF_PACKET, set 'use-mmap' to yes
    #use-mmap: yes
```

Nous pouvons dès a présent voir que notre système est bien en marche et ne présente aucun probleme.

```

bankso@haproxy:~$ sudo nano /etc/suricata/suricata.yaml
bankso@haproxy:~$ sudo systemctl enable --now suricata
Synchronizing state of suricata.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable suricata
bankso@haproxy:~$ sudo systemctl status suricata
● suricata.service - Suricata IDS/IDP daemon
   Loaded: loaded (/lib/systemd/system/suricata.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-03-21 02:04:31 CET; 10s ago
       Docs: man:suricata(8)
              man:suricatasc(8)
              https://suricata-ids.org/docs/
   Process: 9300 ExecStart=/usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid
   Main PID: 9301 (Suricata-Main)
      Tasks: 7 (limit: 2241)
     Memory: 44.6M
        CPU: 443ms
      CGroup: /system.slice/suricata.service
              └─9301 /usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid

mars 21 02:04:31 haproxy systemd[1]: Starting suricata.service - Suricata IDS/IDP daemon...
mars 21 02:04:31 haproxy suricata[9300]: 21/3/2025 -- 02:04:31 - <Notice> - This is Suricata version 6.0.10 RELEASE run
mars 21 02:04:31 haproxy systemd[1]: suricata.service: Can't open PID file /run/suricata.pid (yet?) after start: No suc
mars 21 02:04:31 haproxy systemd[1]: Started suricata.service - Suricata IDS/IDP daemon.
[lines 1-18/18 (END)]

```

## Test

Pour tester notre détection d'intrusion, nous ferons un **sudo nmap -sS -p 22,80,443 192.168.2.129** qui nous permettra de scanner des ports sur notre serveur haproxy et par la suite ne testerons avec Suricata s'il détecte quelque chose

```

bankso@haproxy:~$ sudo nmap -sS -p 80,443,8080 192.168.2.129
Starting Nmap 7.93 ( https://nmap.org ) at 2025-03-21 02:23 CET
Nmap scan report for 192.168.2.129
Host is up (0.000048s latency).

PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
8080/tcp  open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds
bankso@haproxy:~$ curl http://testmyids.com
uid=0(root) gid=0(root) groups=0(root)
bankso@haproxy:~$ sudo tail -f /var/log/suricata/fast.log
03/21/2025-02:22:21.000635 [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 217.160.0.187:80 -> 192.168.2.129:59632
03/21/2025-02:23:31.125820 [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 217.160.0.187:80 -> 192.168.2.129:52414

```

Nous pouvons voir dans les logs de suricata qu'il a bel et bien détecté une activité

## Alternative Proxmox (Conteneurs LXC)

Pour automatiser cela nous avons mis en place des scripts Bash permettant l'installation de ces différentes machines.

```

#!/bin/bash

# Demande des variables avec validation
read -p "Adresse IP WEB : " web_ip
if [[ $web_ip =~ ^([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)$ ]]; then
  echo "adresse IP valide"
else
  exit 1
fi

read -p "Gateway : " gateway
if [[ $gateway =~ ^([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)$ ]]; then
  echo "Gateway invalide"
else
  exit 1
fi

read -p "VM ID : " id
if [[ $id =~ ^[0-9]+$ ]]; then
  echo "L'ID de la VM doit être un nombre"
else
  exit 1
fi

read -p "VM Haproxy ID : " haproxy_id
if [[ $haproxy_id =~ ^[0-9]+$ ]]; then
  echo "L'ID de la VM Haproxy doit être un nombre"
else
  exit 1
fi

# Mise à jour de Proxmox
echo "Mise à jour de Proxmox..."
proxmox update
proxmox

# Téléchargement de la template si elle n'existe pas déjà
echo "Téléchargement de la template Ubuntu..."
proxmox download local ubuntu-22.04-standard_22.04-1_amd64.tar.xz
proxmox

# Créditation du conteneur
echo "Création du conteneur..."
proxmox create "$id" local:virtmp/ubuntu-22.04-standard_22.04-1_amd64.tar.xz \
--hostname $id \
--storage local-lvm \
--password password \
--rootfs 2 \
--cpucores 1 \
--cpuunits 512 \
--net0 name=eth0,bridge=vmbr1,ip=$web_ip/24,gw="$gateway"

```

```

#!/bin/bash

# Demande des variables avec validation
read -p "Adresse IP Haproxy : " haproxy_ip
if [[ $haproxy_ip =~ ^([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)$ ]]; then
  echo "adresse IP invalide"
else
  exit 1
fi

read -p "Adresse IP WEB : " web_ip
if [[ $web_ip =~ ^([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)$ ]]; then
  echo "adresse IP invalide"
else
  exit 1
fi

read -p "Gateway : " gateway
if [[ $gateway =~ ^([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)$ ]]; then
  echo "Gateway invalide"
else
  exit 1
fi

read -p "VM ID : " id
if [[ $id =~ ^[0-9]+$ ]]; then
  echo "L'ID de la VM doit être un nombre"
else
  exit 1
fi

# Mise à jour de Proxmox
echo "Mise à jour de Proxmox..."
proxmox update
proxmox

# Téléchargement de la template si elle n'existe pas déjà
echo "Téléchargement de la template Ubuntu..."
proxmox download local ubuntu-22.04-standard_22.04-1_amd64.tar.xz
proxmox

# Créditation du conteneur
echo "Création du conteneur..."
proxmox create "$id" local:virtmp/ubuntu-22.04-standard_22.04-1_amd64.tar.xz \
--hostname haproxy \
--storage local-lvm \
--password password \
--rootfs 2 \
--cpuunits 512 \
--memory 512 \
--net0 name=eth0,bridge=vmbr1,ip="$haproxy_ip/24",gw="$gateway"

```

```

$ pksh
1 #!/bin/bash
2
3 set -e # Arrête le script en cas d'erreur
4 set -o pipefail
5
6 LOG_FILE="/var/log/deploy/pki-lxc.log"
7 exec > >(tee -a "$LOG_FILE") 2>&1 # Rediriger la sortie vers le log
8
9 echo "Début du déploiement de la PKI avec LXC... $(date)"
10
11 # === DEMANDE DES VARIABLES UTILISATEUR ===
12 read p "$0 du conteneur LXC : $CT_ID"
13 if [[ ! $CT_ID =~ [0-9]+[.][0-9]+ ]]; then
14     echo "L'ID doit être un nombre !"
15     exit 1
16 fi
17
18 read n "Nom du conteneur : $CT_NAME"
19 read a "Adresse IP du conteneur (ex: 192.168.2.50/32) : $IP"
20 if [[ ! $IP =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}([.][0-9]{1,3})? ]]; then
21     echo "Adresse IP invalide !"
22     exit 1
23 fi
24
25 read_p "Passerelle réseau (Gateway) : $GATEWAY"
26 if [[ ! $GATEWAY =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}([.][0-9]{1,3})? ]]; then
27     echo "Gateway invalide !"
28     exit 1
29 fi
30
31 read_p "Nom du serveur PKI : $SERVER_NAME"
32 read_p "Nom du client PKI : $CLIENT_NAME"
33
34 # === CONFIGURATION PAR DÉFAUT ===
35 TEMPLATE="/localtemp/pki/debian-12-standard_12-0_1_amd64.tar.zst" # Image LXC
36 BRIDGE="veth0" # Interface réseau
37 CASSHA_DUB="etl/essy-rsa"
38
39 # --- 1. CRÉATION DU CONTENEUR LXC ---
40 echo "Création du conteneur LXC..."
41 pvt create_lxc "$CT_ID" --name "$CLIENT_NAME"
42 -hostname "$CLIENT_NAME"
43 -net0 "name=eth0,bridge=$BRIDGE,ip=$IP,pmtu=$GATEWAY"
44 -storage local-lvm \

```

```
#!/bin/bash

1 set -e # Arrêter le script en cas d'erreur
2 set -o pipefail
3
4 LOGFILE="/var/log/deploy.ph1.lxc.log"
5 exec > >(tee -a "$LOGFILE") 2>&1 # Redirige la sortie vers le log
6 echo "Début du déploiement de la PI1 avec $(date)""
7
8 # --- DEMARRE DES VARIABLES UTILISATRICE ---
9 read -p "ID du conteneur LXC : " CT_ID
10 if ! [[ $CT_ID =~ ^[0-9]{1,}[0-9]{1,}$ ]]; then
11     echo "ID du conteneur invalide !"
12     exit 1
13 fi
14
15 read -p "Nom du conteneur : " CT_NAME
16 read -p "Adresse IP du conteneur (ex: 192.168.2.56/24) : " IP
17 if ! [[ $IP =~ ^([0-9]{1,}[0-9]{1,}).([0-9]{1,}[0-9]{1,}).([0-9]{1,}[0-9]{1,}).([0-9]{1,}[0-9]{1,})$ ]]; then
18     echo "Adresse IP invalide !"
19     exit 1
20 fi
21
22
23 read -p "Passerelle réseau (Gateway) : " GATEWAY
24 if ! [[ $GATEWAY =~ ^([0-9]{1,}[0-9]{1,}).([0-9]{1,}[0-9]{1,}).([0-9]{1,}[0-9]{1,}).([0-9]{1,}[0-9]{1,})$ ]]; then
25     echo "Gateway invalide !"
26     exit 1
27 fi
28
29 read -p "Nom du serveur PCE : " SERVER_NAME
30 read -p "Nom du client PUI : " CLIENT_NAME
31
32 # --- CONFIGURATION PAR DÉFAUT ---
33 TEMPLATE="/local/www/tmp/obmin-12-standard_12-0-1_and64.tar.zst" # Image LXC
34 INTERFACE="eth0" # Interface réseau
35 FAKE_IP="192.168.1.100"
36
37 # --- 1. CRÉATION DU CONTENEUR LXC ---
38 #echo "Création du conteneur LXC..." 
39 pvt create "$CT_ID" "STERPELATE" \
40 hostnames "$CT_NAME" \
41 netmask="255.255.255.0" bridge="$BRIDGE" ip="$IP" gw="$GATEWAY" \
42 storage="lvm" disk_size="10G" \
43 memory="1024" 
```

# Journalisation

A présent nous allons optimiser la sécurité de notre base de données en éditant le fichier /etc/my.cnf avec sudo vim /etc/my.cnf :

```
#  
# This group is read both by the client and the server  
# use it for options that affect everything  
#[client-server]  
  
#  
# include all files from the config directory  
#!includedir /etc/my.cnf.d  
  
#Désactiver les root distantes  
#bind-address = 127.0.0.1  
  
#Chiffrer les connexions avec un certificat préalablement créé et signé par notre CA  
[mysqld]  
ssl-ca=/var/lib/mysql/ca.crt  
ssl-cert=/var/lib/mysql/bdd.crt  
ssl-key=/var/lib/mysql/bdd.key
```

Puis on redémarre le service

Afin d'automatiser la sauvegarde des données, se sert du script ci-dessous :

```
#!/bin/bash
DATE=$(date +"%Y-%m-%d")
mysqldump -u root -p'BanksP' --all-databases > /backup/mysql_backup_${DATE}.sql
```

Editer grâce à la commande sudo vim bddsave.sh puis on l' ajoute à **crontab** pour exécuter chaque nuit avec crontab -e et dans ce fichier on renseigne la ligne suivante : 0 2 \* \* \* ~/bddsave.sh

NB : Si crontab n'est pas installé, procéder comme suit :

- pour l'installer sudo yum install cronie -y
- puis pour démarrer le service : sudo systemctl start crond
- puis pour le lancer au démarrage : sudo systemctl enable crond

NB : A faire sur chacun des serveurs web.

## 11. Centralisation et Forwarding des journaux (Jouranalisation)

Nous avons choisi d'utiliser le service Rsyslog. Sur notre serveur Haproxy ou un serveur de logs dédié, les étapes de configuration sont les suivantes :

- On installe le service, on le démarre et on le configure pour qu'il se lance au démarrage :

```
sudo yum install rsyslog -y
```

```
sudo systemctl start rsyslog
```

```
sudo systemctl enable rsyslog
```

- On édite le fichier de configuration avec la commande sudo vim /etc/rsyslog.conf comme suit

```
# Activer la réception des logs via UDP et TCP
module(load="imudp")
input(type="imudp" port="514")

module(load="imtcp")
input(type="imtcp" port="514")

$template RemoteLogs,"/var/log/remote/%HOSTNAME%/%PROGRAMNAME%.log"
*.?RemoteLogs
& stop
```

- Puis on redémarre le service
- Ensuite on installe et démarre aussi le service au niveau des serveurs web
- Puis on édite le fichier de configuration de rsyslog sur les serveurs web comme suit :

```
# rsyslog configuration file
.* @10.0.4.26:514 # UDP
.* @@10.0.4.26:514 # TCP
```

10.0.4.26 est l'adresse du serveur de logs

- Puis on redémarre le service
- Pour forcer le Forwarding des logs Nginx il faut dans le fichier de configuration Nginx /etc/nginx/nginx.conf ajouter les lignes : access\_log syslog:server=10.0.4.26:514 ; et error\_log syslog:server=10.0.4.26:514; puis on redémarre le service
- Pour forcer le Forwarding des logs Mariadb il faut éditer le fichier de configuration /etc/my.cnf comme suit:

```
#
# This group is read both by the client and the server
# use it for options that affect everything
#
[client-server]

#
# include all files from the config directory
#
!includedir /etc/my.cnf.d

# Désactiver les root distantes
# bind-address = 127.0.0.1

# Chiffrer les connexions avec un certificat préalablement créé et signé par notre CA
[mysqld]
ssl-ca=/var/lib/mysql/ca.crt
ssl-cert=/var/lib/mysql/bdd.crt
ssl-key=/var/lib/mysql/bdd.key

# Forwarding des logs vers le serveur
log_error = syslog

# Journalisation
general_log = 1
general_log_file = /var/log/mariadb/general.log

# Journalisation des requêtes suspectes
slow_query_log = ON
slow_query_log_file = /var/log/mariadb/slow.log
long_query_time = 1
log_queries_not_using_indexes = ON
```

NB : Il faudra créer les fichiers general.log et slow.log au préalable