

# Infrastructure à mettre en place

Client ⇒ DNS ⇒ HAProxy (LXC) ⇒ oauth2-proxy ⇒ Applications internes

HAProxy dispose d'un certificat signé par ADCS.

Les serveurs sont accessibles :

- en http
- en https avec un certificat signé par ADCS

## Gestion des certificats avec SNA

Qu'est-ce qu'un SAN ?

Le SAN (Subject Alternative Name) est une extension d'un certificat SSL/TLS qui permet d'indiquer plusieurs noms DNS (ou adresses IP) comme étant valides pour un même certificat. C'est devenu obligatoire dans les certificats et certains navigateur comme Chrome/Firefox ne lisent plus le CN seul.

Pour qu'un certificat soit valide pour un nom DNS, ce nom DNS doit apparaître dans subjectAltName.

Rôle du le SAN :

Un certificat peut contenir :

- un CN (Common Name) → ex. CN=guac.lab.local
- plusieurs SAN DNS → ex :
  - DNS:guac.lab.local
  - DNS:[www.guac.lab.local](https://www.guac.lab.local)
  - DNS:guacamole.lab.local

- certificat = 1 application = 1 FQDN

Application	FQDN	Certificat nécessaire
Guacamole	guac.lab.local	guac.lab.local.pem
Grafana	grafana.lab.local	grafana.lab.local.pem
Portainer	portainer.lab.local	portainer.lab.local.pem

HAProxy sélectionne automatiquement le certificat grâce au SNA :

- Quand un navigateur appelle <https://grafana.lab.local>
- ⇒ Il envoie dans la poignée TLS : SNA = grafana.lab.local
- HAProxy :
  - lit le **SNA**
  - scanne le dossier **/etc/haproxy/certs/**
  - choisit le **.pem** correspondant
  - sert le bon certificat

Cela permet :

- une gestion simple
- un remplacement facile
- moins de risques d'exposition croisée
- une rotation par certificat

## Installer HAProxy

- Créer un conteneur LXC
- Mettre à jour

```
apt update && apt upgrade -y
```

- installer haproxy

```
apt install haproxy -y
```

- vérifier l'installation

```
haproxy -v
```

- Activer HAProxy au démarrage

```
systemctl enable haproxy
systemctl start haproxy
```

- Afficher le statut du service :

```
systemctl status haproxy
```

## Utiliser des certificats signés par une CA Microsoft

### Obtenir les certificats intermédiaires ADCS

- se connecter à <http://serveur-ADCS/certsrv>
- cliquez sur **Télécharger un certificat d'autorité de certification, une chaîne de certificats ou une liste de révocation des certificats** :

Services de certificats *Microsoft* Active Directory -- 0870019Y-SRV-WIN-SIO-CA-CA Accueil

---

**Bienvenue !**

Utilisez ce site Web pour demander un certificat pour votre navigateur Web, votre programme client de messagerie électronique ou un autre programme. En utilisant un certificat, vous pouvez confirmer votre identité aux personnes avec lesquelles vous communiquez sur le Web, signer et chiffrer des messages et, selon le type de certificat que vous demandez, effectuer d'autres tâches sécurisées.

Vous pouvez également utiliser ce site Web pour télécharger un certificat d'autorité de certification, une chaîne de certificats ou une liste de révocation des certificats, ou vous pouvez afficher le statut d'une requête en attente.

Pour obtenir plus d'informations sur les Services de certificats Active Directory, voir [Documentation sur les Services de certificats Active Directory](#).

**Sélectionnez une tâche :**

- [Demander un certificat](#)
- [Afficher le statut d'une requête de certificat en attente](#)
- [Télécharger un certificat d'autorité de certification, une chaîne de certificats ou une liste de révocation des certificats](#)**

- cliquez sur **Télécharger la chaîne de certificats d'autorité de certification** au format **Base 64** pour obtenir le fichier **certnew.p7b** :

Services de certificats *Microsoft* Active Directory -- 0870019Y-SRV-WIN-SIO-CA-CA Accueil

---

**Télécharger un certificat d'autorité de certification, une chaîne de certificats ou la liste de révocation des certificats**

Pour approuver les certificats approuvés par l'autorité de certification, [Installer ce certificat d'autorité de certification](#)

Pour sélectionner un certificat d'autorité de certification, une chaîne de certificats ou une liste de révocation des certificats, sélectionnez un certificat et une méthode de chiffrement.

**Certificat de l'autorité de certification :**

Actuel [0870019Y-SRV-WIN-SIO-CA-CA] ▲

**méthode de codage :**

DER

Base 64

[Installer un certificat d'autorité de certification](#)

[Télécharger un certificat de l'autorité de certification](#)

**[Télécharger la chaîne de certificats d'autorité de certification](#)**

[Télécharger la dernière Liste de révocation des certificats de base](#)

[Télécharger la dernière Liste de révocation des certificats delta](#)

- sous Linux extraire uniquement les intermédiaires

```
openssl pkcs7 -in certnew.p7b -print_certs -out chain-all.pem
```

- Puis extraire les intermédiaires :

La racine a généralement un Subject = Issuer Les intermédiaires ont Subject != Issuer

Tu peux les séparer manuellement, ou avec un script si tu veux (je peux t'en fournir un).

- certificats signés par la CA Microsoft
- utilisable pour les serveurs Linux (conteneurs LXC)
- sans avoir de warnings SSL dans les navigateurs Windows

Lien : \* [Gérer des certificats pour serveurs Debian avec la CA de Microsoft](#)

Le conteneur LXC ne peut pas demander automatiquement un certificat à la CA Microsoft.

## générer une clé + CSR dans le LXC

- générer la clé privée

```
cd /root
openssl genrsa -out guac.lab.local.key 2048
```

Crée un fichier san.cnf pour ajouter un SAN (recommandé) avec ce contenu :

```
[ req ]
default_bits = 2048
prompt = no
default_md = sha256
req_extensions = req_ext
distinguished_name = dn

[ dn ]
CN = guac.lab.local

[ req_ext ]
subjectAltName = @alt_names

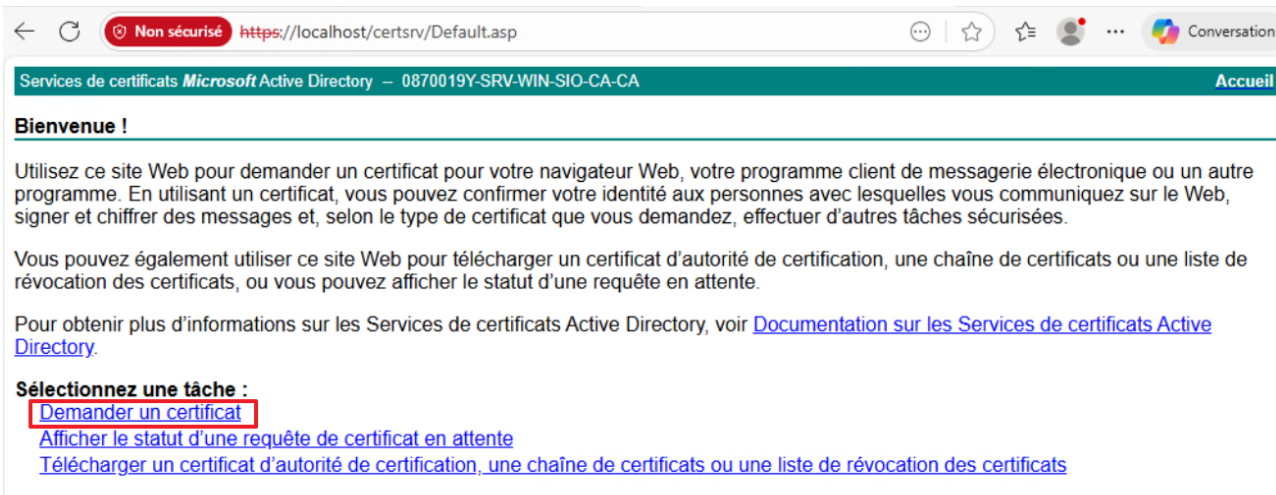
[ alt_names ]
DNS.1 = guac.lab.local
```

- générer le CSR

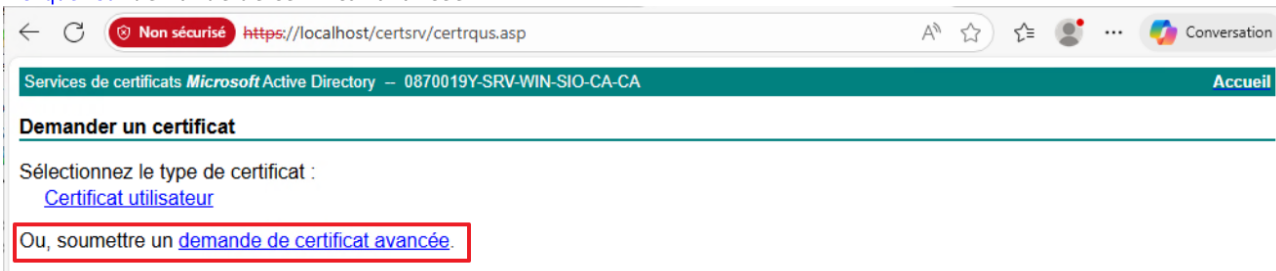
```
openssl req -new -key guac.lab.local.key \
-out guac.lab.local.csr \
-config san.cnf
```

## soumettre la CSR à la CA Microsoft

- Depuis un PC Windows membre du domaine, accédez à `http:<ADCS_SERVER>/certsrv` \* Cliquez sur **Demander un certificat**



\* Cliquez sur demande de certificat avancée :



Request a certificate => advanced certificate request submit a certificate request by using a base-64 encoded file Colle le contenu de guac.lab.local.csr Sélectionne un template adapté, par exemple : Web Server Computer ou un template personnalisé activé pour SAN ⚠ Important : dans la console CA, le template doit autoriser : "Allow private key to be exported" -> pas nécessaire "Supply in the request" -> obligatoire pour les SAN Télécharge ensuite : le certificat au format Base64 la chaîne "CA certificate" (Root CA + éventuellement la subCA) Tu obtiendras un fichier .cer. □ Étape 3 — Convertir le certificat Microsoft en PEM Dans le LXC : Copie le certificat : scp user@windows:/path/to/guac.lab.local.cer /root/ Convertis-le si nécessaire : Shellopenssl x509 -in guac.lab.local.cer -out guac.lab.local.crtAfficher plus de lignes Copie également le certificat de l'autorité racine (Root CA), depuis Windows : certutil -ca.cert rootCA.cer Puis convertis/le mets en .crt : Shellopenssl x509 -in rootCA.cer -out rootCA.crtAfficher plus de lignes (si tu as une CA intermédiaire : fais pareil) □ Étape 4 — Construire le fichier .pem pour HAProxy HAProxy doit avoir un seul fichier .pem contenant : la clé privée le certificat serveur la chaîne CA (intermédiaires + racine) Exemple : Shellcat guac.lab.local.crt guac.lab.local.key rootCA.crt > \ /etc/haproxy/certs/guac.lab.local.pemAfficher plus de lignes Donne des permissions sécurisées : Shellchmod 600 /etc/haproxy/certs/guac.lab.local.pemAfficher plus de lignes □ Étape 5 — Recharger HAProxy haproxy -c -f /etc/haproxy/haproxy.cfg systemctl reload haproxy Navigue ensuite vers : https://guac.lab.local Tu devrais obtenir aucune alerte SSL sur les PC du domaine Windows, car la CA Microsoft est déjà dans le magasin "Trusted Root Certification Authorities". □ Répéter pour chaque application Pour : grafana.lab.local portainer.lab.local vault.lab.local Tu répètes simplement : génération clé + CSR soumission à la CA Microsoft création du .pem final dépôt dans /etc/haproxy/certs/ HAProxy détectera automatiquement le bon certificat grâce au SNI. □ Bonus : Template conseillé dans ADCS pour reverse-proxy Dans l'outil "Certification Authority", crée un template basé sur Web Server, configure : Subject -> "Supply in the request" (obligatoire pour SAN) Extensions -> "Server Authentication" Validity : 2 ans Private key -> "Key usage: Digital Signature + Key Encipherment" Key length : 2048 ou 4096 bits C'est LE bon template pour HAProxy. □ Résumé simple ÉtapeAction1Générer clé + CSR dans le LXC2Envoyer la CSR à la CA Microsoft3Télécharger certificat + CA4Assembler .pem pour HAProxy5Reload HAProxy □ Tous les PC Windows du domaine font automatiquement confiance aux certificats émis par la CA Microsoft : zéro alerte SSL. ===== Installer le certificat ===== ===== récupérer le certificat signé ===== assembler le .crt + .key + chaîne CA ===== déposer le .pem dans /etc/haproxy/certs ===== Installer oauth2-proxy dans le conteneur HAProxy ===== Lien : https://github.com/oauth2-proxy/oauth2-proxy \* créer le dossier <code> mkdir -p /opt/oauth2-proxy </code> \* Télécharger le binaire officiel oauth2-proxy <code> cd /opt/oauth2-proxy wget https://github.com/oauth2-proxy/oauth2-proxy/releases/download/v7.13.0/oauth2-proxy-v7.13.0.linux-amd64.tar.gz tar -xvf oauth2-proxy-v7.13.0.linux-amd64.tar.gz mv oauth2-proxy-v7.13.0.linux-amd64/oauth2-proxy /usr/local/bin/ chmod +x /usr/local/bin/oauth2-proxy </code> \* Créer l'utilisateur système dédié (recommandé) <code> useradd -system -no-create-home -shell /usr/sbin/nologin oauth2-proxy </code> \* Créer le dossier de configuration <code> mkdir -p /etc/oauth2-proxy chown oauth2-proxy:oauth2-proxy /etc/oauth2-proxy </code> Ce dossier permet de stocker le fichier oauth2-proxy.cfg. \* Générer le cookie secret Un cookie secret 32 octets base64 URL-safe est obligatoire. <code> python3 - «'EOF' import os,base64; print(base64.urlsafe\_b64encode(os.urandom(32)).decode()) EOF </code> Noter la valeur générée qui sera nécessaire pour le service systemd : \* HG4Us2GnS8ZrJoM8ZSDnxxxxAtrcxWlxxxx8IP8AyXQ = \* Créer un service systemd pour oauth2-proxy pour une application

Exemple pour l'application Guacamole (à adapter avec ton clientid / clientsecret / redirect URL) : <code> nano /etc/systemd/system/oauth2-proxy-guacamole.service </code> Contenu : <code> [Unit] Description=oauth2-proxy (Guacamole) After=network-online.target [Service] User=oauth2-proxy Group=oauth2-proxy ExecStart=/usr/local/bin/oauth2-proxy \ -provider=msentraid \ -client-id=<APPIDGUAC> \ -client-secret=<APPSECRETGUAC> \ -redirect-url=https://guac.lab.local/oauth2/callback \ -email-domain=\* \ -cookie-secret=<TONCOOKIESECRET> \ -cookie-secure=true \ -cookie-samesite=None \ -cookie-name=guacoauth \ -set-xauthrequest \ -pass-access-token \ -http-address=127.0.0.1:4181 \ -upstream=http://10.xxx.yyy.zzz:8080/guacamole Restart=always RestartSec=3 [Install]

WantedBy=multi-user.target </code> \* Sauvegardez puis activez : <code> # systemctl daemon-reload # systemctl enable -now oauth2-proxy-guacamole Created symlink '/etc/systemd/system/multi-user.target.wants/oauth2-proxy-guacamole.service' → '/etc/systemd/system/oauth2-proxy-guacamole.service'. </code> ☐ Toutes les options du binaire (cookies, upstream, OIDC provider, scopes) suivent exactement les paramètres décrits dans la documentation oauth2-proxy. ☐ Le provider msenraid est officiellement supporté. \* Vérifier le fonctionnement <code> systemctl status oauth2-proxy-guacamole journalctl -u oauth2-proxy-guacamole -f </code> ===== Ajouter HAProxy en frontal ===== \* Ajoutez dans le fichier de configuration principal d'HAProxy **/etc/haproxy/haproxy.cfg** les lignes nécessaires. Ce fichier contient 4 grandes sections : \* global \* defaults \* frontend ... (entrée du trafic depuis Internet ou ton LAN) \* backend ... (cible vers laquelle HAProxy envoie les requêtes) Il faut ajouter : \* Un frontend qui écoute sur le port 443 (HTTPS) \* Un backend par application protégée via oauth2-proxy (ici Guacamole) Il faut configurer dans HAProxy (LXC), un backend pointant vers l'instance oauth2-proxy en rajoutant les lignes suivantes: <code> # ===== # FRONTEND HTTPS (entrant) # ===== frontend fehttps bind \*:443 ssl crt /etc/haproxy/certs/ # Détection du domaine utilisé acl hostguac hdr(host) -i sioguacamole.0870019y.lan # Routage vers le backend correspondant usebackend bkoauth2guac if hostguac defaultbackend bkdefault # ===== # BACKEND GUACAMOLE via oauth2-proxy # ===== backend bkoauth2guac http-request set-header X-Forwarded-Proto https http-request set-header X-Forwarded-Host %[req.hdr(host)] server oauth2\_guac 127.0.0.1:4181 check # Backend par défaut (si domaine inconnu) backend bk\_default errorfile 503 /etc/haproxy/errors/503.http </code> nouveau contenu du fichier **/etc/haproxy/haproxy.cfg** <code> global log /dev/log local0 log /dev/log local1 notice chroot /var/lib/haproxy stats socket /run/haproxy/admin.sock mode 660 level admin stats timeout 30s user haproxy group haproxy daemon # Default SSL material locations ca-base /etc/ssl/certs crt-base /etc/ssl/private # See: https://ssl-config.mozilla.org/#server=haproxy&server-version=2.0.3&config=intermediate ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GC> ssl-default-bind-ciphersuites TLSAES128GCM\_SHA256:TLSAES256GCM\_SHA384:TLSCHACHA20POLY1305\_SHA256 ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets defaults log global mode http option httplog option dontlognull timeout connect 5000 timeout client 50000 timeout server 50000 errorfile 400 /etc/haproxy/errors/400.http errorfile 403 /etc/haproxy/errors/403.http errorfile 408 /etc/haproxy/errors/408.http errorfile 500 /etc/haproxy/errors/500.http errorfile 502 /etc/haproxy/errors/502.http errorfile 503 /etc/haproxy/errors/503.http errorfile 504 /etc/haproxy/errors/504.http # ===== # FRONTEND HTTPS (entrant) # ===== frontend fe\_https bind \*:443 ssl crt /etc/haproxy/certs/ # Détection du domaine utilisé acl host\_guac hdr(host) -i sioguacamole.0870019y.lan # Routage vers le backend correspondant usebackend bkoauth2guac if hostguac defaultbackend bkdefault # ===== # BACKEND GUACAMOLE via oauth2-proxy # ===== backend bkoauth2guac http-request set-header X-Forwarded-Proto https http-request set-header X-Forwarded-Host %[req.hdr(host)] server oauth2\_guac 127.0.0.1:4181 check # Backend par défaut (si domaine inconnu) backend bk\_default errorfile 503 /etc/haproxy/errors/503.http </code> ===== Commentaires ===== \* Le frontend reçoit les connexions des utilisateurs (port 443) \* Le backend indique où HAProxy doit envoyer la requête \* → ici vers oauth2-proxy, qui tourne en local sur 127.0.0.1:4181 dans ton LXC. oauth2-proxy va ensuite : \* vérifier si l'utilisateur est authentifié \* sinon rediriger vers Microsoft Entra \* si oui → transmettre au serveur interne (Guacamole) grâce au paramètre -upstream mis dans le service systemd ===== tester & redémarrer ===== \* Vérifier la syntaxe des fichiers : <code> haproxy -c -f /etc/haproxy/haproxy.cfg </code> \* vérifier si tout est OK : <code> systemctl restart haproxy </code> ===== Test de fonctionnement ===== \* Ouvre un navigateur et visite <https://siopguacamole.0870019y.lan> Tu dois voir : \* Redirection vers Microsoft Entra ID \* Connexion \* Retour dans Guacamole (derrière oauth2-proxy) Pour vérifier en ligne de commande : <code> curl -I https://guac.lab.local </code> Il devrait y avoir une réponse du proxy. ☐ Résultat Tu as maintenant : B. Variante au proxy : placer oauth2-proxy entre HAProxy et Guacamole, et déléguer l'auth au proxy (OIDC/SAML) avec Entra ID, puis passer des en-têtes vers Guacamole. C'est utile si tu veux factoriser l'auth pour plusieurs apps derrière le même HAProxy. (Côté Entra, l'intégration la plus robuste est en OIDC avec l'IDP Microsoft Entra ID). [oauth2-pro....github.io], [oauth2-pro....github.io] Tip HA : si tu veux de la haute dispo du frontal, tu pourras ultérieurement doubler ce LXC et ajouter Keepalived/VRRP. [forum.proxmox.com] ===== Présentation de l'architecture SSO au niveau du proxy via oauth2-proxy (+ Entra ID) ===== \* centraliser l'authentification sur HAProxy au lieu de le faire dans chaque app), \* oauth2-proxy sait s'intégrer avec Microsoft Entra ID (OIDC) ; \* HAProxy fait appliquer l'auth (mode "forward-auth"/vérif de cookie) et injecter des en-têtes (X-Email, X-User, etc.) en amont de Guacamole. \* oauth2-proxy supporte OIDC/Entra ID nativement, avec gestion des groupes/claims

Choix de OIDC ici plutôt que SAML car l'intégration oauth2-proxy avec Entra ID est officielle, documentée et maintenue côté OIDC (group overage, multi-tenant, etc.).

- o lien : [https://oauth2-proxy.github.io/oauth2-proxy/configuration/providers/ms\\_entra\\_id/](https://oauth2-proxy.github.io/oauth2-proxy/configuration/providers/ms_entra_id/)

HAProxy (LXC) en frontal → oauth2-proxy (OIDC Microsoft Entra ID) → Applications internes (Guacamole, Grafana, ...) Cette approche centralise l'authentification côté proxy, factorise les règles (groupes, accès), et décharge les applis. Elle s'appuie sur le provider Microsoft Entra ID officiel d'oauth2-proxy (OIDC), bien documenté et maintenu. Lien : <https://github.com/ahucate/pfsense-haproxy> Objectifs SSO commun via Microsoft Entra ID (OIDC) pour plusieurs applis. 1 conteneur LXC Proxmox hébergeant HAProxy (terminaison TLS) et \*\*une ou plusieurs instances d'oauth2-proxy. Routage SNI : un FQDN par appli (ex. guac.lab.local, grafana.lab.local). Isolement & lisibilité : 1 instance oauth2-proxy par appli (recommandé pour débiter), chacune avec son App Registration Entra.

From:

/ - **Les cours du BTS SIO**

Permanent link:

[/doku.php/reseau/cloud/haproxy?rev=1768848405](https://reseau:cloud:haproxy/doku.php/reseau/cloud/haproxy?rev=1768848405)

Last update: **2026/01/19 19:46**

