Services & Administration

- Nextcloud
- GITEA
- Installation et configuration d'un relais SMTP
- GLPI docker SSL + LDAP
- CheckMK

Nextcloud

Cette documentation détaille l'installation d'une instance Nextcloud All-In-One (AIO) optimisée. La particularité de cette installation réside dans l'utilisation d'un stockage déporté, d'un reverse-proxy Caddy et de certificats SSL locaux via mkcert.

Prérequis système

Avant de commencer, nous installons les outils de base nécessaires à la gestion des dépôts, du réseau et des conteneurs.

```
apt install curl git wget rsync docker-compose-plugin
```

Configuration du stockage dédié

Pour garantir les performances, nous dédions un disque entier (`/dev/sdb`) aux données.

```
fdisk -l
mkfs -t ext4 /dev/sdb
sudo mkdir -p /mnt/sdb
lsblk -no UUID /dev/sdb  # Note l'UUID
echo "UUID=xxx /mnt/sdb ext4 defaults 0 2" >> /etc/fstab
sudo mount -a
```

Docker (sur `/mnt/sdb`)

```
# Installation Docker
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update && sudo apt install docker-ce docker-ce-cli docker-compose-plugin
```

```
containerd.io

# Docker sur /mnt/sdb
sudo systemctl stop docker docker.socket
sudo mkdir -p /mnt/sdb/docker

sudo tee /etc/docker/daemon.json <<EOF
{
  "data-root": "/mnt/sdb/docker"
}
EOF

sudo rsync -aP /var/lib/docker/ /mnt/sdb/docker/
sudo systemctl start docker docker.socket
```

mkcert SSL

```
# Installation mkcert
wget $(curl -s https://api.github.com/repos/FiloSottile/mkcert/releases/latest | grep
browser_download_url | grep linux-amd64 | cut -d '"' -f 4)
sudo mv mkcert* /usr/local/bin/mkcert && sudo chmod +x /usr/local/bin/mkcert
mkcert -install
mkcert cloud.eyrode.lan

# Permissions
sudo mkdir -p /var/opt/mkcert
sudo mv cloud.eyrode.lan*.pem /var/opt/mkcert/
sudo chown -R 1000:1000 /var/opt/mkcert
sudo chmod 644 /var/opt/mkcert/*.pem
sudo chmod 600 /var/opt/mkcert/*-key.pem
```

Nextcloud AIO (HTTPS + Caddy)

```
sudo mkdir -p /var/opt/nextcloud/all-in-one
cd /var/opt/nextcloud/all-in-one
git clone https://github.com/nextcloud/all-in-one.git .
```

`compose.yaml` **FINAL** (copie-colle complet) :

```
name: nextcloud-aio
services:
  nextcloud-aio-mastercontainer:
    image: ghcr.io/nextcloud-releases/all-in-one:latest
    init: true
    restart: always
    container_name: nextcloud-aio-mastercontainer
    volumes:
      - nextcloud_aio_mastercontainer:/mnt/docker-aio-config
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - /var/opt/mkcert:/mnt/certs:ro
    ports:
      - 8080:8080
    environment:
      APACHE_PORT: 11000
      APACHE_IP_BINDING: 0.0.0.0
      NEXTCLOUD_DOMAIN: cloud.eyrode.lan
      SKIP_DOMAIN_VALIDATION: true
      SSL_CERTIFICATE_FILE: /mnt/certs/cloud.eyrode.lan.pem
      SSL_CERTIFICATE_KEY_FILE: /mnt/certs/cloud.eyrode.lan-key.pem

  caddy:
    image: caddy:alpine
    restart: always
    container_name: caddy
    volumes:
      - caddy_certs:/certs
      - caddy_config:/config
      - caddy_data:/data
      - caddy_sites:/srv
      - /var/opt/mkcert:/var/opt/mkcert:ro
    network_mode: host
    configs:
      - source: Caddyfile
        target: /etc/caddy/Caddyfile

configs:
  Caddyfile:
```

```
    content: |
      cloud.eyrode.lan {
        reverse_proxy localhost:11000
        tls /var/opt/mkcert/cloud.eyrode.lan.pem /var/opt/mkcert/cloud.eyrode.lan-key.pem
      }

volumes:
  nextcloud_aio_mastercontainer:
    name: nextcloud_aio_mastercontainer
  caddy_certs:
  caddy_config:
  caddy_data:
  caddy_sites:
```

Lancement :

```
docker compose up -d
```

Dépannage (Si nécessaire)

Port 443 bloqué :

```
sudo systemctl restart docker
docker system prune -af
docker compose up -d
```

Interface AIO : `https://cloud.eyrode.lan:8080` **Clique "Start all containers" → 5min** → tout vert

Accès Final

```
https://cloud.eyrode.lan           # Nextcloud (cadenas vert mkcert)
https://cloud.eyrode.lan:8080      # Interface AIO
```

Logs de diagnostic

```
docker compose logs nextcloud-aio-mastercontainer
docker compose logs caddy
docker ps  # Tous les conteneurs doivent être "Up"
```

□□ Récupération Mot de Passe Admin

Méthode 1 : Interface AIO (RECOMMANDÉE) https://cloud.eyrode.lan:8080 → "Access details" → Password admin affiché !

Méthode 2 : Reset CLI (syntaxe correcte)

```
docker exec -u www-data -e OC_PASS="MonSuperPass123!" nextcloud-aio-nextcloud php occ user:add
tanny --password-from-env --group=admin --display-name="Eythan"
```

GITEA

Documentation GITEA

Eythan RONCIER-LEMEE | 15/09/2025 **BTS SIO SISR | IIA Laval | TP-GITEA.pdf**

1. Notions

1.1. Debian

Système d'exploitation basé sur Linux, réputé pour sa stabilité et sa fiabilité. Il utilise la gestion de paquets **APT** et le format **.deb**.

1.2. Gitea

Service Git auto-hébergé, léger et performant, écrit en Go. Fournit une interface web similaire à GitHub avec gestion des tickets et pull requests.

1.3. Apache

Serveur web open source utilisé pour traiter les requêtes HTTP/HTTPS. Utilisé ici comme reverse proxy.

1.4. SSH (Secure Shell)

Protocole de communication sécurisé et chiffré pour l'administration à distance.

1.5. LVM (Logical Volume Manager)

Couche d'abstraction entre disques physiques et systèmes de fichiers, permettant le redimensionnement flexible des volumes logiques.

1.6. FHS (Filesystem Hierarchy Standard)

Norme définissant l'organisation des répertoires sur Linux (ex: `/etc` pour les confs, `/var` pour les données).

2. Installation de Debian

2.1. Réseaux

En l'absence de DHCP, configuration manuelle des paramètres :

- Adresse IP, Masque, Passerelle et DNS.
- Fichier de configuration : `/etc/network/interfaces`.

2.2. Nom de machine et Utilisateurs

- Modification du hostname dans `/etc/hostname` et `/etc/hosts`.
- Création de l'utilisateur root et d'un utilisateur standard.
- Utilisation de `adduser` et `usermod -aG sudo` pour les droits d'administration.

2.3. Partitionnement LVM manuel

Structure type mise en place :

- **sda1** : BIOS GRUB (2 Mo)
- **sda2** : BIOS UEFI (512 Mo)
- **sda3** : /boot (512 Mo - EXT4)
- **sda4** : Volume physique LVM (27,4 Go)

Volumes logiques (vg00) :

- `lv_root` (4 Go), `lv_var` (3 Go), `lv_var_log` (2 Go), `lv_tmp` (1 Go), `lv_home` (2 Go), `lv_swap` (512 Mo).

3. SSH

3.1. Installation et activation

```
apt update && apt install ssh
systemctl enable ssh
systemctl status ssh
```

3.2. Sécurisation par clés

1. Génération d'une paire de clés (ex: PuTTYgen).
2. Création du dossier `~/.ssh` (chmod 700).
3. Ajout de la clé publique dans `~/.ssh/authorized_keys` (chmod 600).
4. Désactivation de l'authentification par mot de passe dans `/etc/ssh/sshd_config` : `PasswordAuthentication no`.

4. MariaDB

4.1. Installation et Sécurisation

```
apt install mariadb-server
mariadb-secure-installation
```

4.2. Base de données pour Gitea

```
CREATE DATABASE giteaDB CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
CREATE USER 'gitea'@'localhost' IDENTIFIED BY 'votre_mot_de_passe';
GRANT ALL PRIVILEGES ON giteaDB.* TO 'gitea'@'localhost';
FLUSH PRIVILEGES;
```

5. Gitea

5.1. Compte de service

Création d'un utilisateur système dédié :

```
groupadd giteausers
adduser --system --shell /bin/bash --gecos 'Git Version Control' --group --disabled-password
--home /var/lib/gitea gitea
```

5.2. Installation du binaire

```
cd /usr/local/bin
wget -O gitea [https://dl.gitea.com/gitea/1.24.5/gitea-1.24.5-linux-
amd64](https://dl.gitea.com/gitea/1.24.5/gitea-1.24.5-linux-amd64)
chmod +x gitea
```

5.3. Service Systemd

Configuration du fichier `/etc/systemd/system/gitea.service` pour automatiser le lancement avec l'utilisateur `gitea`.

```
systemctl enable gitea
systemctl start gitea
```

6. Reverse Proxy - Apache

6.1. Modules et Virtual Host

Activation des modules proxy : `a2enmod proxy proxy_http ssl`.

Configuration du fichier `/etc/apache2/sites-available/gitea.conf` :

```
<VirtualHost *:443>
    ProxyPreserveHost On
    ProxyRequests Off
    AllowEncodedSlashes NoDecode
    ProxyPass / http://localhost:3000/ nocanon
</VirtualHost>
```

6.2. Mise en ligne

```
a2ensite gitea.conf
apache2ctl configtest
systemctl restart apache2
```

Installation et configuration d'un relais SMTP

Prérequis

Les prérequis sont assez simple il suffit de disposer:

- machine sous Debian en version 13
- accès réseau permettant de joindre le domaine du relais SMTP
- dépots à jour permettant la récupération des divers paquets.

Installation

1. Postfix : Le moteur d'envoi (MTA)

Explication

Postfix est un **MTA** (Mail Transfer Agent), c'est-à-dire l'agent de transfert de courrier. Son rôle est de gérer toute la logistique technique de l'envoi.

- **Sa mission :** Il réceptionne les messages envoyés par le système et les transporte vers un serveur distant appelé "Relayhost".
- **Sécurité :** Il s'occupe de l'authentification sécurisée (via SASL) pour prouver votre identité au serveur de sortie.
- **Conformité :** Il réécrit les adresses d'expédition pour s'assurer qu'elles sont valides et éviter que vos mails ne soient bloqués ou marqués comme SPAM.

Installation

Pas de grande complexité il suffit d'aller chercher le paquets dans le dépots définis après ou pendant l'installation de Debian :

```
sudo apt install postfix sasl2-bin libsasl2-modules
```

2. Mailutils : L'outil de rédaction (MUA)

Explication

Mailutils est un **MUA** (Mail User Agent), soit l'agent utilisateur. C'est l'interface qui permet à l'humain ou aux scripts de "parler" à Postfix.

- **Sa mission :** Il fournit les commandes nécessaires pour rédiger, envoyer et lire des courriels en ligne de commande.
- **Utilisation :** On utilise principalement sa commande `mail` pour tester la configuration en envoyant un message de test vers une adresse externe.
- **Outils inclus :** Il comprend divers utilitaires pour lister les messages ou manipuler les boîtes aux lettres.

Installation

```
sudo apt install mailutils
```

Dans la phase d'installation ces trois demandes seront formulés, je ne vais pas trop m'y attarder car je vais repasser sur la configuration dans la partie de la documentation correspondante :

- L'installateur propose plusieurs types de serveurs. Bien que la documentation préconise d'être vigilant si le système propose "Système satellite", c'est ce que je vais choisir dans ce cas.



Selon wiki ubuntu-fr, Un système satellite pour Postfix signifie simplement que les mails du système seront envoyés automatiquement à un serveur SMTP. Ceci vous permettra d'utiliser la commande mail (du package mailx) directement pour envoyer des emails.

Il pourra par la suite être utilisé pour envoyer des emails automatiquement comme par exemple des fichiers de logs grâce à cron.

- Cette seconde demande concerne le nom de domaine qui servira à qualifier les adresses sortantes sans domaine précis. Par défaut, l'installateur suggère le nom d'hôte de la machine



- Pour finir, l'installateur demande l'adresse du sserveur qui transmettra vos messages. On peut saisir une valeur temporaire, car nous configurerons précisement le relais avec une authentification dans les étapes suivantes



> Une fois l'installation terminée, nous procédons au remplacement de cette configuration automatique pour appliquer nos paramètres spécifiques

Configuration

> Pour la configuration il peut être plus simple de passer par le SSH !

Préparation des fichiers

Avant de commencer quoique ce soit il va falloir arrêter le service `postfix.service` :

```
systemctl stop postfix
```

Puis sauvegarder la configuration originale en cas de problème dans les prochaines étapes.

```
mv /etc/postfix/main.cf /etc/postfix/main.cf.bck
```

Prérequis

- Une machine sous Debian (version 13 testée).
- Un accès réseau vers le port du relais SMTP (généralement 465 ou 587).
- Les privilèges `sudo` ou un accès `root`.

1. Installation des paquets

Postfix (MTA) et SASL

Objectif : Installer le moteur de transfert et les modules nécessaires à l'authentification sécurisée.

```
sudo apt update
sudo apt install postfix sasl2-bin libsasl2-modules
```

2. Configuration du Relais (main.cf)

Paramétrage générique

Objectif : Configurer Postfix pour utiliser un hôte distant avec chiffrement fort.

```
# Sauvegarde de la configuration par défaut
mv /etc/postfix/main.cf /etc/postfix/main.cf.orig
```

Contenu suggéré pour `/etc/postfix/main.cf`

```
# Configuration du relais distant
relayhost = [<RELAY_HOST>]:<PORT>

# Sécurité TLS Client
smtp_tls_wrappermode = yes
smtp_tls_security_level = encrypt
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt

# Authentification SASL
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/smtp_sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_sasl_mechanism_filter = plain, login
```

```
# Réécriture d'identité
sender_canonical_maps = regexp:/etc/postfix/sender_canonical_maps
smtp_header_checks = regexp:/etc/postfix/smtp_header_check
```

3. Gestion de l'authentification et de l'identité

Création des tables de hachage

Objectif : Stocker les identifiants et définir les règles de réécriture d'adresses.

1. **Identifiants du relais :**

```
echo "[<RELAY_HOST>]:<PORT> <USER_AUTH>:<PASSWORD>" > /etc/postfix/smtp_sasl_passwd
chmod 600 /etc/postfix/smtp_sasl_passwd
postmap /etc/postfix/smtp_sasl_passwd
```

2. **Réécriture Canonique (Enveloppe):**

```
echo '/.+/ <EMAIL_VALIDE>' > /etc/postfix/sender_canonical_maps
postmap /etc/postfix/sender_canonical_maps
```

3. **Nettoyage des entêtes (Headers):** Création de `/etc/postfix/smtp_header_check`:

```
/^From:/          REPLACE From: "<NOM_AFFICHAGE>" <<EMAIL_VALIDE>>
/^Sender:/        IGNORE
/^Resent-Sender:/ IGNORE
```

Incident : Caractères non-ASCII (Espaces insécables)

Problématique : Erreur fatale au démarrage suite à un copier-coller depuis une documentation web.

Action de correction :

```
# Détection et remplacement des octets UTF-8 invisibles (\xc2\xa0) par des espaces standards
sed -i 's/\xc2\xa0/ /g' /etc/postfix/main.cf

# Vérification de la syntaxe
postfix check
postconf -n
```

Test final

```
echo "Corps du message" | mail -s "Sujet du Test" -a "From: <EMAIL_VALIDE>" <DESTINATAIRE>
tail -f /var/log/mail/mail.log
```

GLPI docker SSL + LDAP

1. INSTALLATION DOCKER & DOCKER-COMPOSE

```
# Mise à jour système
sudo apt update && sudo apt upgrade -y

# Installation Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Installation Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-
$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# Services + utilisateur
sudo systemctl enable --now docker
sudo usermod -aG docker $USER

# Vérification
docker --version
docker compose version
```

2. INSTALLATION MKCERT (SUR L'HÔTE)

```
# Dépendances
sudo apt install libnss3-tools wget -y

# Téléchargement mkcert
wget https://github.com/FiloSottile/mkcert/releases/latest/download/mkcert-v1.4.4-linux-amd64
sudo mv mkcert-v1.4.4-linux-amd64 /usr/local/bin/mkcert
sudo chmod +x /usr/local/bin/mkcert

# Installation CA locale
```

```
sudo mkcert -install

# Vérification
mkcert -version
```

3. CRÉATION DOSSIER & CERTIFICATS MKCERT

```
# Dossier GLPI
sudo mkdir -p /var/www/support.eyrode.lan
cd /var/www/support.eyrode.lan

# Génération certificats mkcert
mkcert support.eyrode.lan 172.16.82.2 localhost 127.0.0.1 ::1

# Vérification certificats
ls -la support.eyrode.lan*.pem
chmod 600 support.eyrode.lan*.pem
```

Fichiers créés :
support.eyrode.lan.pem ? Certificat public
support.eyrode.lan-key.pem ? Clé privée

5. CRÉATION DOCKER-COMPOSE.YML

```
cat > docker-compose.yml << 'EOF'
services:
  mariadb:
    image: mariadb
    container_name: mariadb
    hostname: mariadb
    volumes:
      - /var/lib/mysql:/var/lib/mysql
    env_file:
      - ./mariadb.env
    restart: always

  glpi:
    image: diouxx/glpi
    container_name: glpi
    hostname: glpi
```

```
    volumes:
      - /var/www/html/glpi/:/var/www/html/glpi
    environment:
      - TIMEZONE=Europe/Paris
    restart: always
  caddy:
    image: caddy:alpine
    container_name: caddy
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./Caddyfile:/etc/caddy/Caddyfile
      - ./support.eyrode.lan.pem:/etc/caddy/certs/support.eyrode.lan.pem:ro
      - ./support.eyrode.lan-key.pem:/etc/caddy/certs/support.eyrode.lan-key.pem:ro
      - caddy_data:/data
      - caddy_config:/config
    depends_on:
      - glpi
    restart: always

volumes:
  caddy_data:
  caddy_config:
EOF
```

6. CRÉATION MARIADB.ENV

```
cat > mariadb.env << 'EOF'
MYSQL_ROOT_PASSWORD=SuperRoot2026!
MYSQL_DATABASE=glpidb
MYSQL_USER=glpi
MYSQL_PASSWORD=GlpiSupport2026!
EOF
chmod 600 mariadb.env
```

7. CRÉATION CADDYFILE

```
cat > Caddyfile << 'EOF'
support.eyrode.lan {
    tls /etc/caddy/certs/support.eyrode.lan.pem /etc/caddy/certs/support.eyrode.lan-key.pem
    reverse_proxy glpi:80 {
        header_up Host {upstream_hostport}
        header_up X-Real-IP {remote_host}
        header_up X-Forwarded-Proto https
        header_up X-Forwarded-For {remote_host}
    }
}
EOF
```

8. PRÉPARATION VOLUMES PERSISTANTS

```
sudo mkdir -p /var/www/html/glpi /var/lib/mysql
sudo chown -R 33:33 /var/www/html/glpi/
sudo chmod -R 755 /var/www/html/glpi/
```

9. DÉPLOIEMENT (Lancement)

```
cd /var/www/support.eyrode.lan/

# Lancement
docker compose up -d

# Attente initialisation
sleep 30




# Vérification
docker compose ps
docker compose logs caddy
```

Logs attendus :
caddy | serving initial configuration
mariadb | 2026-03-05 21:30:00 0 [Note] /usr/sbin/mariadbd: ready for connections

glpi | Apache/2.4.x (Debian) Server started

10. CONFIGURATION INITIALE GLPI

Ouvrir : https://support.eyrode.lan
Assistant installation :Langue : Français

Base de données :

- **Hôte :** mariadb
- **Utilisateur :** root
- **Mot de passe :** SuperRoot2026!
- **Base :** glpidb

Continuer ? Login : glpi/glpi

11. VÉRIFICATIONS FINALES

```
# Statut services
docker compose ps  → 3/3 Up

# Test HTTPS mkcert (cadenas vert)
curl -k https://support.eyrode.lan  → HTML GLPI

# Test backend interne
docker exec caddy curl http://glpi:80  → Page login

# Base glpidb
docker exec mariadb mysql -u root -pSuperRoot2026! -e "SHOW TABLES FROM glpidb;" | wc -l  →
442

# Certificats mkcert
docker exec caddy ls -la /etc/caddy/certs/  → support.eyrode.lan.pem
```

12. SAUVEGARDE IMMÉDIATE

```
cd /var/www/support.eyrode.lan/

# Base + fichiers + mkcert
docker exec mariadb mysqldump -u root -pSuperRoot2026! glpidb > glpi-$(date +%Y%m%d).sql
tar czf glpi-complete-$(date +%Y%m%d).tar.gz docker-compose.yml mariadb.env Caddyfile
support.eyrode.lan*.pem -C /var/www/html/glpi .
```

```
ls -lh glpi-*.sql glpi-complete-*.tar.gz
```

COMMANDES UTILITAIRES

```
# Redémarrage rapide
docker compose restart caddy

# Mise à jour
docker compose pull && docker compose up -d

# Logs temps réel
docker compose logs -f

# Nettoyage
docker compose down && docker system prune -f
```

CheckMK

CheckMK.pdf

prérequis

- Debian 13 ou 12
- Google Chrome >= 131; Firefox >= 133; Microsoft Edge >= 131; Safari >= 18.1
- Seconde machine

Docker

Prérequis

Désinstallation des paquets conflictuels

```
apt remove $(dpkg --get-selections docker.io docker-compose docker-doc podman-docker
containerd runc | cut -f1)
```

Installation

```
# Add Docker's official GPG key:
apt update
apt install ca-certificates curl
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc
chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
tee /etc/apt/sources.list.d/docker.sources <<EOF
Types: deb
URIs: https://download.docker.com/linux/debian
Suites: $(. /etc/os-release && echo "$VERSION_CODENAME")
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
```

```
EOF

apt update

apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Vérification

```
systemctl status docker
systemctl start docker
docker run hello-world
```

CheckMK

Tout ce qui est expliqué plus bas peut ce retrouver sur ces sources :

- Documentation CheckMK

Déploiement du conteneur

```
mkdir -p checkmk
cd /var/opt/checkmk
touch docker-compose.yml
```

```
services:
  checkmk:
    image: "checkmk/check-mk-raw:2.4.0-latest"
    container_name: "bigbrother"
    environment:
      - CMK_PASSWORD=mypassword #Entrer un mot de passe
      - TZ=Europe/Paris
    volumes:
      - bigbrother:/omd/sites
    tmpfs:
      - /opt/omd/sites/cmk/tmp:uid=1000,gid=1000
    ports:
      - 8080:5000
```

```
      - 8000:8000
    restart: always

volumes:
  bigbrother:
```

Mise en place de mkcert

```
mkdir -p /var/opt/certif
cd /var/opt/certif
curl -sSL https://github.com/FiloSottile/mkcert/releases/latest/download/mkcert-v1.4.4-linux-amd64 -o /usr/local/bin/mkcert
chmod +x /usr/local/bin/mkcert
```

```
mkcert -install
mkcert bigbrother.eyrode.lan
ls -la
```

Ajout de caddy

docker compose complet

```
services:
  checkmk:
    image: "checkmk/check-mk-raw:2.4.0-latest"
    container_name: "bigbrother"
    environment:
      - CMK_PASSWORD=mypassword #Entrer un mot de passe
      - TZ=Europe/Paris
    volumes:
      - bigbrother:/omd/sites
    tmpfs:
      - /opt/omd/sites/cmk/tmp:uid=1000,gid=1000
    restart: always

  caddy:
    image: caddy:latest
    container_name: "caddy"
```

```
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /var/opt/certif:/certs:ro
      - ./Caddyfile:/etc/caddy/Caddyfile:ro
    depends_on:
      - checkmk
    restart: always

volumes:
  bigbrother:
```

CaddyFile

```
bigbrother.eyrode.lan {
    tls /certs/bigbrother.eyrode.lan.crt /certs/bigbrother.eyrode.lan.key
    reverse_proxy checkmk:5000
}
```