Guide de Déploiement - Serveur Web Centralisé PROTOLAB

Date : 28 décembre 2025 **Version** : 1.0 **Auteur** : Claude Code **Projet** : Portfolio Protolab V4.7

Table des Matières

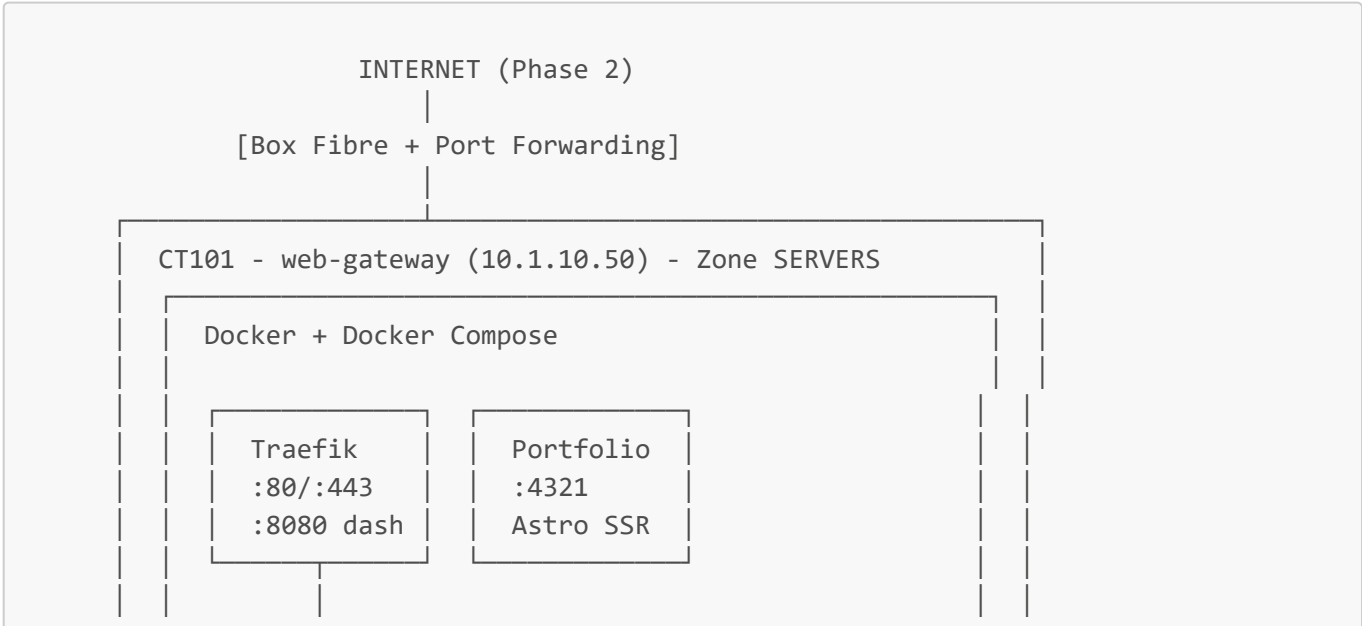
Résumé

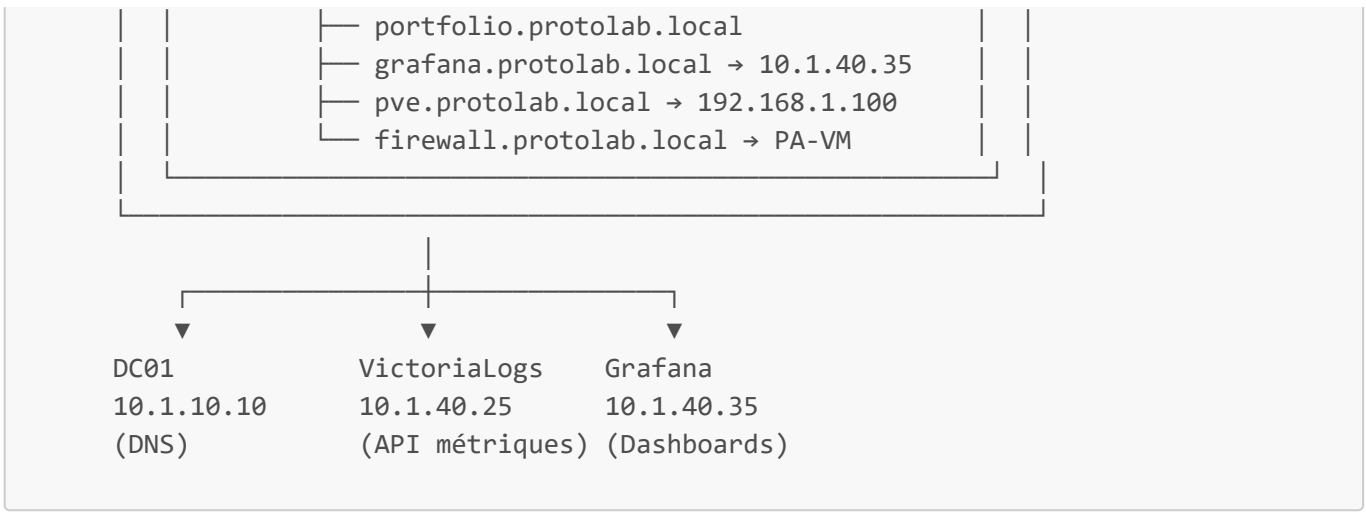
Ce guide détaille le déploiement d'un serveur web centralisé (CT LXC) hébergeant le portfolio V4.7 et servant de reverse proxy pour tous les services web de l'infrastructure PROTOLAB.

Phase 1 (Immédiat) : Déploiement local sur réseau `protolab.local`, accessible via VPN GlobalProtect.

Phase 2 (Janvier 2025) : Exposition Internet via port forwarding box fibre + domaine.

Architecture Cible





Prérequis

- [ ] Accès SSH à Proxmox (192.168.1.100)
- [ ] Template Ubuntu 22.04 disponible sur Proxmox
- [ ] Accès à l'interface PA-VM (https://192.168.1.37 ou 192.168.1.254)
- [ ] Accès RDP/PowerShell à DC01 (10.1.10.10)
- [ ] VPN GlobalProtect fonctionnel (pour tests à distance)

Phase 1 - Création du CT LXC "web-gateway"

Spécifications du Conteneur

Paramètre	Valeur
CTID	101
Hostname	web-gateway
Template	ubuntu-22.04-standard
CPU	2 vCPU
RAM	2048 MB
Swap	512 MB
Disk	20 GB (local-lvm)
Réseau	vmbr1 (SERVERS)
IP	10.1.10.50/24
Gateway	10.1.10.254 (PA-VM)
DNS	10.1.10.10 (DC01)
Features	nesting=1, keyctl=1

Étape 1.1 : Télécharger le Template Ubuntu 22.04

```
# SSH vers Proxmox
ssh root@192.168.1.100

# Vérifier si le template existe
pveam available | grep ubuntu-22.04

# Télécharger le template si absent
pveam download local ubuntu-22.04-standard_22.04-1_amd64.tar.zst
```

Étape 1.2 : Créer le Conteneur LXC

```
pct create 101 local:vztmpl/ubuntu-22.04-standard_22.04-1_amd64.tar.zst \
  --hostname web-gateway \
  --cores 2 \
  --memory 2048 \
  --swap 512 \
  --rootfs local-lvm:20 \
  --net0 name=eth0,bridge=vmbr1,ip=10.1.10.50/24,gw=10.1.10.254 \
  --nameserver 10.1.10.10 \
  --searchdomain protolab.local \
  --features nesting=1,keyctl=1 \
  --unprivileged 1 \
  --start 1
```

Vérification :

```
pct list | grep 101
pct status 101
```

Étape 1.3 : Configuration Initiale du CT

```
# Entrer dans le conteneur
pct enter 101

# Mise à jour du système
apt update && apt upgrade -y

# Installer les paquets essentiels
apt install -y \
  ca-certificates \
  curl \
  gnupg \
  lsb-release \
  git \
  htop \
```

```
  nano \
  wget

# Vérifier la connectivité réseau
ping -c 3 10.1.10.10    # DC01
ping -c 3 10.1.40.25    # VictoriaMetrics
ping -c 3 1.1.1.1       # Internet
```

Étape 1.4 : Installer Docker

```
# Installation Docker via script officiel
curl -fsSL https://get.docker.com | sh

# Activer Docker au démarrage
systemctl enable docker
systemctl start docker

# Vérifier Docker
docker --version
docker run hello-world

# Installer Docker Compose plugin
apt install -y docker-compose-plugin

# Vérifier Docker Compose
docker compose version
```

Étape 1.5 : Créer l'Arborescence des Projets

```
# Créer la structure de dossiers
mkdir -p /opt/docker/{traefik,portfolio,cloudflared}
mkdir -p /opt/docker/traefik/{dynamic,logs}

# Vérifier
tree /opt/docker -L 2
```

Résultat attendu :

```
/opt/docker/
├── traefik/
│   ├── dynamic/
│   └── logs/
├── portfolio/
└── cloudflared/
```

Phase 2 - Installation Traefik

Étape 2.1 : Créer le Fichier docker-compose.yml

```
cd /opt/docker/traefik
nano docker-compose.yml
```

Contenu :

```
version: '3.8'

services:
  traefik:
    image: traefik:v3.0
    container_name: traefik
    restart: unless-stopped
    security_opt:
      - no-new-privileges:true
    ports:
      - "80:80"
      - "443:443"
      - "8080:8080"  # Dashboard (interne uniquement)
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ./traefik.yml:/etc/traefik/traefik.yml:ro
      - ./dynamic:/etc/traefik/dynamic:ro
      - ./acme.json:/acme.json
      - ./logs:/var/log/traefik
    networks:
      - traefik-public
    labels:
      - "traefik.enable=true"
      # Dashboard interne
      - "traefik.http.routers.traefik-
dashboard.rule=Host(`traefik.protolab.local`)"
      - "traefik.http.routers.traefik-dashboard.service=api@internal"
      - "traefik.http.routers.traefik-dashboard.entrypoints=web"

networks:
  traefik-public:
    name: traefik-public
    driver: bridge
```

Étape 2.2 : Créer le Fichier traefik.yml

```
nano traefik.yml
```

Contenu :

```
api:
  dashboard: true
  insecure: true  # Dashboard sur :8080 (interne)

entryPoints:
  web:
    address: ":80"
  websecure:
    address: ":443"

providers:
  docker:
    endpoint: "unix:///var/run/docker.sock"
    exposedByDefault: false
    network: traefik-public
  file:
    directory: /etc/traefik/dynamic
    watch: true

log:
  level: INFO
  filePath: /var/log/traefik/traefik.log

accessLog:
  filePath: /var/log/traefik/access.log
```

Étape 2.3 : Créer les Fichiers de Configuration Dynamique

Routes vers les services internes :

```
nano dynamic/internal-services.yml
```

Contenu :

```
http:
  routers:
    # Grafana
    grafana-router:
      rule: "Host(`grafana.protolab.local`)"
      service: grafana-service
      entryPoints:
        - web

    # Proxmox (HTTPS backend)
    proxmox-router:
      rule: "Host(`pve.protolab.local`)"
```

```
      service: proxmox-service
      entryPoints:
        - web

    # Palo Alto (HTTPS backend)
    paloalto-router:
      rule: "Host(`firewall.protolab.local`)"
      service: paloalto-service
      entryPoints:
        - web

  services:
    grafana-service:
      loadBalancer:
        servers:
          - url: "http://10.1.40.35:3000"

    proxmox-service:
      loadBalancer:
        servers:
          - url: "https://192.168.1.100:8006"
        serversTransport: insecure-transport

    paloalto-service:
      loadBalancer:
        servers:
          - url: "https://10.1.10.254:443"
        serversTransport: insecure-transport

  serversTransports:
    insecure-transport:
      insecureSkipVerify: true
```

Middlewares (sécurité) :

```
nano dynamic/middlewares.yml
```

Contenu :

```
http:
  middlewares:
    # Security headers
    secure-headers:
      headers:
        frameDeny: true
        browserXssFilter: true
        contentTypeNosniff: true
        stsSeconds: 31536000
        stsIncludeSubdomains: true
```

```
        stsPreload: true
        customFrameOptionsValue: "SAMEORIGIN"

    # Rate limiting
    rate-limit:
      rateLimit:
        average: 100
        burst: 50
```

Étape 2.4 : Initialiser acme.json

```
touch acme.json
chmod 600 acme.json
```

Étape 2.5 : Démarrer Traefik

```
# Démarrer Traefik
docker compose up -d

# Vérifier les logs
docker compose logs -f

# Vérifier le statut
docker compose ps
```

Vérification :

- Dashboard : http://10.1.10.50:8080 (ou http://traefik.protolab.local:8080)

Phase 3 - Déploiement Portfolio

Étape 3.1 : Cloner le Repository

```
cd /opt/docker/portfolio

# Cloner depuis GitHub
git clone https://github.com/AdrienNewman/portfolio-protolab.git .

# Vérifier les fichiers
ls -la
```

Étape 3.2 : Créer le Dockerfile SSR

Le Dockerfile actuel utilise Nginx statique, mais l'API LiveLab nécessite Node.js. Créer un nouveau fichier :

```
nano Dockerfile.ssr
```

Contenu :

```
# Build stage
FROM node:20-alpine AS builder

WORKDIR /app

# Copier package files
COPY package*.json ./

# Installer les dépendances
RUN npm ci

# Copier le code source
COPY . .

# Build Astro en mode SSR
RUN npm run build

# Production stage - Node.js runtime
FROM node:20-alpine AS runtime

WORKDIR /app

# Copier uniquement les fichiers nécessaires
COPY --from=builder /app/dist ./dist
COPY --from=builder /app/node_modules ./node_modules
COPY --from=builder /app/package.json ./

# Variables d'environnement
ENV HOST=0.0.0.0
ENV PORT=4321
ENV NODE_ENV=production

# Exposer le port
EXPOSE 4321

# Healthcheck
HEALTHCHECK --interval=30s --timeout=3s --start-period=40s \
  CMD node -e "require('http').get('http://localhost:4321/api/lab-status.json',
(r) => {process.exit(r.statusCode === 200 ? 0 : 1)})"

# Démarrer le serveur Node
CMD ["node", "./dist/server/entry.mjs"]
```

Étape 3.3 : Créer le Fichier .env

```
nano .env
```

Contenu :

```
NODE_ENV=production
VICTORIA_METRICS_URL=http://10.1.40.25:8428
VICTORIA_LOGS_URL=http://10.1.40.25:9428
```

Étape 3.4 : Créer le docker-compose.yml

```
nano docker-compose.yml
```

Contenu :

```
version: '3.8'

services:
  portfolio:
    build:
      context: .
      dockerfile: Dockerfile.ssr
    container_name: protolab-portfolio
    restart: unless-stopped
    env_file:
      - .env
    networks:
      - traefik-public
    labels:
      - "traefik.enable=true"
      # Route interne (protolab.local)
      - "traefik.http.routers.portfolio-
internal.rule=Host(`portfolio.protolab.local`)"
      - "traefik.http.routers.portfolio-internal.entrypoints=web"
      - "traefik.http.services.portfolio.loadbalancer.server.port=4321"
      # Middlewares
      - "traefik.http.routers.portfolio-internal.middlewares=secure-headers@file"

networks:
  traefik-public:
    external: true
```

Étape 3.5 : Build et Démarrage

```
# Build l'image Docker
docker compose build

# Démarrer le conteneur
docker compose up -d

# Vérifier les logs
docker compose logs -f portfolio

# Tester l'API
curl http://localhost:4321/api/lab-status.json
```

Vérification :

- Portfolio : http://10.1.10.50 ou http://portfolio.protolab.local
- API LiveLab : http://portfolio.protolab.local/api/lab-status.json

---

Phase 4 - Configuration Firewall PA-VM

Étape 4.1 : Créer les Objets Réseau

Via CLI PA-VM :

```
# SSH vers PA-VM
ssh admin@192.168.1.37

# Entrer en mode configuration
configure

# Créer les objets adresses
set address web-gateway ip-netmask 10.1.10.50/32
set address victorialogs ip-netmask 10.1.40.25/32
set address grafana ip-netmask 10.1.40.35/32

# Créer les objets services
set service victoria-api protocol tcp port 8428
set service grafana-web protocol tcp port 3000
```

Étape 4.2 : Créer les Règles de Sécurité

```
# Règle : Portfolio vers VictoriaMetrics
set rulebase security rules portfolio-to-victoria \
  from SERVERS \
  to INFRA \
  source web-gateway \
  destination victorialogs \
  application any \
```

```
  service victoria-api \
  action allow

# Règle : Portfolio vers Grafana
set rulebase security rules portfolio-to-grafana \
  from SERVERS \
  to INFRA \
  source web-gateway \
  destination grafana \
  application any \
  service grafana-web \
  action allow

# Commit les changements
commit
```

Via GUI PA-VM :

1. Accéder à https://192.168.1.37
2. Policies > Security
3. Add rule :
   - **Name** : portfolio-to-victoria
   - **Source Zone** : SERVERS
   - **Destination Zone** : INFRA
   - **Source Address** : web-gateway
   - **Destination Address** : victorialogs
   - **Service** : victoria-api (TCP/8428)
   - **Action** : Allow
4. Répéter pour Grafana

Étape 4.3 : Vérifier les Règles

```
# Dans PA-VM CLI
show running security-policy-match

# Tester depuis web-gateway
pct enter 101
curl -I http://10.1.40.25:8428/api/v1/query?query=up
curl -I http://10.1.40.35:3000
```

Phase 5 - Configuration DNS

Étape 5.1 : Ajouter les Enregistrements DNS sur DC01

Via PowerShell sur DC01 :

```
# Enregistrement A pour web-gateway
Add-DnsServerResourceRecordA -ZoneName "protolab.local" -Name "web-gateway" -
IPv4Address "10.1.10.50"

# Enregistrements CNAME
Add-DnsServerResourceRecordCName -ZoneName "protolab.local" -Name "portfolio" -
HostNameAlias "web-gateway.protolab.local"
Add-DnsServerResourceRecordCName -ZoneName "protolab.local" -Name "traefik" -
HostNameAlias "web-gateway.protolab.local"
Add-DnsServerResourceRecordCName -ZoneName "protolab.local" -Name "grafana" -
HostNameAlias "web-gateway.protolab.local"
Add-DnsServerResourceRecordCName -ZoneName "protolab.local" -Name "pve" -
HostNameAlias "web-gateway.protolab.local"
Add-DnsServerResourceRecordCName -ZoneName "protolab.local" -Name "firewall" -
HostNameAlias "web-gateway.protolab.local"

# Vérifier les enregistrements
Get-DnsServerResourceRecord -ZoneName "protolab.local" | Where-Object {$_.HostName
-like "*gateway*" -or $_.HostName -eq "portfolio"}
```

Étape 5.2 : Tester la Résolution DNS

Depuis un poste Windows du domaine :

```
nslookup web-gateway.protolab.local
nslookup portfolio.protolab.local
nslookup grafana.protolab.local
```

Résultat attendu :

```
Server:  dc01.protolab.local
Address:  10.1.10.10

Name:    web-gateway.protolab.local
Address:  10.1.10.50
```

Phase 6 - Tests et Validation

Checklist de Validation

Infrastructure

- [ ] CT101 démarré et accessible (SSH)
- [ ] Docker fonctionnel (`docker ps`)
- [ ] Réseau OK (ping DC01, VictoriaMetrics, Internet)

Traefik

- [ ] Conteneur Traefik en cours d'exécution
- [ ] Dashboard accessible : http://traefik.protolab.local:8080
- [ ] Logs Traefik sans erreurs

Portfolio

- [ ] Build Docker réussi
- [ ] Conteneur portfolio en cours d'exécution
- [ ] API accessible : http://portfolio.protolab.local/api/lab-status.json
- [ ] Données LiveLab affichées correctement

Services Internes (Reverse Proxy)

- [ ] Grafana : http://grafana.protolab.local
- [ ] Proxmox : http://pve.protolab.local
- [ ] Palo Alto : http://firewall.protolab.local

DNS

- [ ] Résolution DNS web-gateway OK
- [ ] Résolution DNS portfolio OK
- [ ] Tous les CNAME résolus correctement

Tests Fonctionnels

Test 1 : Accès Portfolio

```
# Depuis votre poste
curl -I http://portfolio.protolab.local
```

Attendu : HTTP/1.1 200 OK

Test 2 : API LiveLab

```
curl http://portfolio.protolab.local/api/lab-status.json | jq
```

Attendu : JSON avec métriques Proxmox

Test 3 : Reverse Proxy Grafana

```
curl -I http://grafana.protolab.local
```

Attendu : HTTP/1.1 200 OK (redirection Grafana)

Test 4 : Via VPN GlobalProtect

- Connecter le VPN
- Ouvrir http://portfolio.protolab.local dans le navigateur
- Vérifier l'affichage du LiveLab

Phase 7 - Migration Future (Box Fibre)

Quand la Box Fibre sera installée (Fin Janvier 2025)

Option A : Avec Domaine Personnalisé

Prérequis :

- Domaine acheté (ex: adriennewman.com)
- Ajouté à Cloudflare (NS pointés)

Étape 7.1 : Configurer Port Forwarding sur la Box

Port Externe	Port Interne	Destination IP
80	80	10.1.10.50
443	443	10.1.10.50

Étape 7.2 : Ajouter Règle NAT sur PA-VM

```
# CLI PA-VM
configure

set rulebase nat rules web-inbound \
  from OUTSIDE \
  to OUTSIDE \
  source any \
  destination 192.168.1.254 \
  service service-http \
  destination-translation translated-address web-gateway

commit
```

Étape 7.3 : Configurer DNS Cloudflare

1. Aller sur https://dash.cloudflare.com
2. Sélectionner votre domaine
3. DNS > Records > Add record :
    - **Type** : A
    - **Name** : portfolio (ou @)
    - **IPv4** : [Votre IP publique]
    - **Proxy** : Activé (orange cloud)

Étape 7.4 : Mettre à Jour docker-compose.yml Portfolio

```
labels:
  - "traefik.http.routers.portfolio-
external.rule=Host(`portfolio.adriennewman.com`)"
  - "traefik.http.routers.portfolio-external.entrypoints=websecure"
  - "traefik.http.routers.portfolio-external.tls=true"
```

```
docker compose up -d
```

Option B : Avec DuckDNS (Gratuit)

Étape 7.1 : Créer un compte DuckDNS

1. Aller sur https://www.duckdns.org
2. Se connecter avec Google/GitHub
3. Créer un sous-domaine : `protolab.duckdns.org`
4. Récupérer le token

Étape 7.2 : Installer le client DuckDNS

```
# Sur CT101 web-gateway
cd /opt/docker
mkdir duckdns
cd duckdns

# Créer le script de mise à jour
nano duck.sh
```

Contenu duck.sh :

```
#!/bin/bash
echo url="https://www.duckdns.org/update?domains=protolab&token=VOTRE_TOKEN&ip=" |
curl -k -o ~/duckdns/duck.log -K -
```

```
chmod +x duck.sh

# Ajouter au crontab
crontab -e
```

Ajouter :

```
*/5 * * * * /opt/docker/duckdns/duck.sh >/dev/null 2>&1
```

Étape 7.3 : Configurer Traefik pour Let's Encrypt

Modifier `traefik.yml` :

```
certificatesResolvers:
  letsencrypt:
    acme:
      email: votre@email.com
      storage: /acme.json
      httpChallenge:
        entryPoint: web
```

Dépannage

Problème : Conteneur Portfolio ne démarre pas

Symptômes :

```
docker compose ps
# Status: Restarting
```

Solution :

```
# Voir les logs détaillés
docker compose logs portfolio

# Vérifier les variables d'environnement
docker compose exec portfolio env | grep VICTORIA

# Tester manuellement
docker compose exec portfolio node ./dist/server/entry.mjs
```

Problème : API LiveLab retourne 502

Cause : VictoriaMetrics inaccessible

Solution :

```
# Tester depuis le conteneur
docker compose exec portfolio sh
wget -O- http://10.1.40.25:8428/api/v1/query?query=up
```

```
# Vérifier les règles PA-VM
# Voir Phase 4
```

Problème : DNS ne résout pas

Symptômes :

```
nslookup portfolio.protolab.local
# Server failed
```

Solution :

```
# Sur DC01
# Vérifier le serveur DNS
Get-DnsServerResourceRecord -ZoneName "protolab.local"

# Redémarrer le service DNS
Restart-Service DNS

# Vider le cache DNS client
ipconfig /flushdns
```

Problème : Traefik ne route pas correctement

Solution :

```
# Vérifier les logs Traefik
docker logs traefik

# Vérifier les labels des conteneurs
docker inspect protolab-portfolio | grep traefik

# Forcer le rechargement
docker compose restart traefik
```

Problème : Certificat SSL auto-signé (Phase 2)

Normal : Let's Encrypt ne peut pas émettre de certificat sans domaine public.

Solution temporaire :

- Utiliser HTTP uniquement en interne
- Accepter le certificat auto-signé dans le navigateur
- Attendre la Phase 2 avec domaine public

Ressources Supplémentaires

Commandes Utiles

Surveillance des conteneurs :

```
# Logs en temps réel
docker compose logs -f

# Statistiques
docker stats

# Espace disque
df -h
docker system df
```

Sauvegarde :

```
# Sauvegarder les volumes
docker run --rm -v traefik_acme:/data -v $(pwd):/backup ubuntu tar czf
/backup/acme-backup.tar.gz /data

# Sauvegarder la config
tar czf /root/web-gateway-backup.tar.gz /opt/docker
```

Liens de Référence

- Documentation Traefik : https://doc.traefik.io/traefik/
- Documentation Astro SSR : https://docs.astro.build/en/guides/server-side-rendering/
- Documentation Docker Compose : https://docs.docker.com/compose/
- VictoriaMetrics API : https://docs.victoriametrics.com/

Changelog

Version	Date	Modifications
1.0	28/12/2025	Version initiale - Déploiement local
1.1	À venir	Ajout Cloudflare Tunnel ou Port Forwarding

Fin du guide de déploiement