Déploiement Grafana/Prometheus + exporters

Déploiement Grafana/Prometheus

Déploiement Prometheus + Grafana + Exporters - Guide Complet

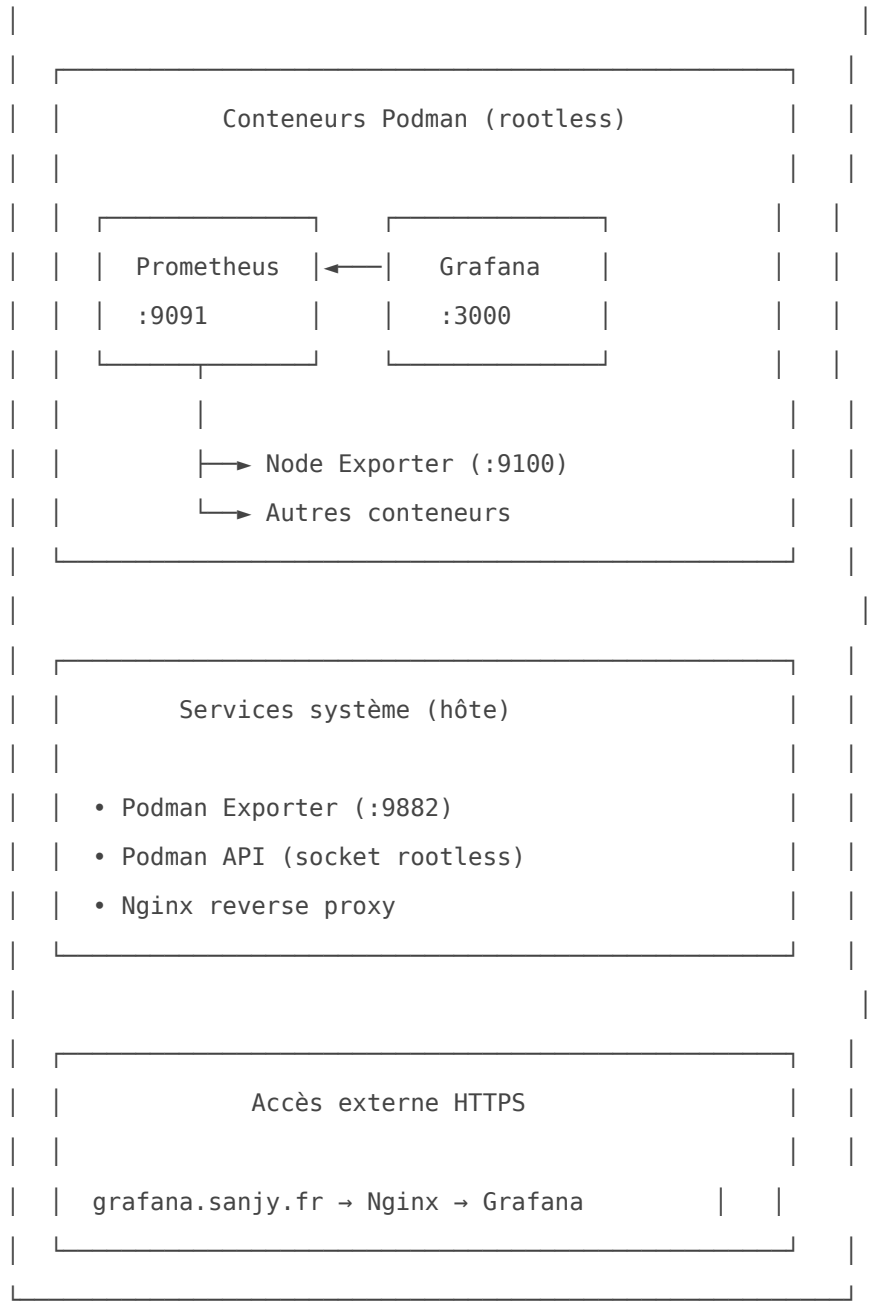
Date : 28 Novembre 2025
Environnement : AlmaLinux 10, Podman rootless
Infrastructure : VPS OVH avec monitoring complet des services

---

Table des matières

Architecture finale

```
┌─────────────────────────────────────────────────┐
│                VPS AlmaLinux 10                 │
```



Stack de monitoring déployée

Services déployés

Service	Type	Port	Description
Prometheus	Conteneur	9091	Collecte et stockage des métriques
Grafana	Conteneur	3000	Visualisation et dashboards

Service	Type	Port	Description
Node Exporter	Conteneur	9100	Métriques système (CPU, RAM, disque)
Podman Exporter	Systemd user	9882	Métriques des conteneurs Podman
Podman API	Systemd user	Socket	API Podman pour l'exporter
Nginx	Système	80/443	Reverse proxy HTTPS

Services monitorés

- Jenkins (conteneur)
- HashiCorp Vault (conteneur)
- BookStack (conteneur)
- Prometheus (auto-monitoring)
- Grafana (auto-monitoring)
- Système hôte (via Node Exporter)

Déploiement Prometheus & Grafana

Structure des fichiers

```
~/monitoring/
├── docker-compose.yml
├── prometheus/
│   ├── config/
│   │   └── prometheus.yaml
│   └── data/                    # Volume Podman
└── grafana/
    ├── data/                    # Volume Podman
    └── provisioning/
        └── datasources/
            └── prometheus.yml
```

Création de la structure

```
mkdir -p ~/monitoring/{prometheus/config,grafana/provisioning/datasources}
cd ~/monitoring
```

Configuration Prometheus

Fichier `prometheus/config/prometheus.yaml` :

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s

scrape_configs:
  # Prometheus auto-monitoring
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  # Node Exporter (système)
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['host.containers.internal:9100']

  # Podman Exporter (conteneurs)
  - job_name: 'podman_exporter'
    static_configs:
      - targets: ['host.containers.internal:9882']

  # Nginx Exporter (optionnel)
  - job_name: 'nginx'
    static_configs:
      - targets: ['host.containers.internal:9113']
```

Points clés :

- Utilisation de `host.containers.internal` au lieu de `localhost` pour accéder aux services de l'hôte depuis les conteneurs
- Port 9091 exposé sur l'hôte (9090 interne au conteneur)

Configuration datasource Grafana

Fichier `grafana/provisioning/datasources/prometheus.yml` :

```
apiVersion: 1

datasources:
  - name: Prometheus
    type: prometheus
    access: proxy
    url: http://prometheus:9090
    isDefault: true
    editable: false
```

Note : Utilisation de `prometheus:9090` (résolution DNS interne Docker/Podman entre conteneurs du même réseau).

Docker Compose

Fichier `docker-compose.yml` :

```
version: '3.8'

services:
  prometheus:
    image: docker.io/prom/prometheus:latest
    container_name: prometheus
    restart: unless-stopped
    volumes:
      - ./prometheus/config:/etc/prometheus:Z
      - prometheus_data:/prometheus:Z
    command:
      - '--config.file=/etc/prometheus/prometheus.yaml'
      - '--storage.tsdb.path=/prometheus'
      - '--storage.tsdb.retention.time=30d'
    ports:
      - "9091:9090"
    networks:
      - monitoring

  grafana:
    image: docker.io/grafana/grafana:latest
```

```
    container_name: grafana
    restart: unless-stopped
    user: "472"
    volumes:
      - grafana_data:/var/lib/grafana:Z
      - ./grafana/provisioning:/etc/grafana/provisioning:Z
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD= ******
      - GF_SERVER_ROOT_URL=https://grafana.sanjy.fr
      - GF_SERVER_DOMAIN=grafana.sanjy.fr
    ports:
      - "3000:3000"
    networks:
      - monitoring
    depends_on:
      - prometheus

  node-exporter:
    image: docker.io/prom/node-exporter:latest
    container_name: node-exporter
    restart: unless-stopped
    command:
      - '--path.rootfs=/host'
      - '--path.procfs=/host/proc'
      - '--path.sysfs=/host/sys'
      - '--collector.filesystem.mount-points-exclude=^/(sys|proc|dev|host|etc)($$|/)'
    volumes:
      - /:/host:ro,rslave
    ports:
      - "9100:9100"
    networks:
      - monitoring

volumes:
  prometheus_data:
  grafana_data:

networks:
  monitoring:
```

```
    driver: bridge
```

Points importants :

- `:Z` sur les volumes pour SELinux
- `user: "472"` pour Grafana (UID grafana)
- Port Prometheus changé en 9091 (9090 était utilisé par systemd)

Fix SELinux

```
# Appliquer les contextes SELinux
sudo chcon -R -t container_file_t ~/monitoring/

# Ou permanent
sudo semanage fcontext -a -t container_file_t "$HOME/monitoring(/.*)?"
sudo restorecon -Rv ~/monitoring/
```

Démarrage

```
cd ~/monitoring
podman-compose up -d

# Vérifier les logs
podman-compose logs -f

# Vérifier que tout tourne
podman ps
```

Accès

- **Prometheus :** `http://localhost:9091`
- **Grafana :** `http://localhost:3000` (admin / ChangeMe123!)

Configuration des exporters

Node Exporter (conteneur)

Déjà inclus dans docker-compose.yml.

Test :

```
curl http://localhost:9100/metrics | grep node_
```

Podman Exporter (service systemd user)

Pourquoi en service systemd et pas en conteneur ?

- Accès au socket Podman rootless difficile depuis un conteneur
- Plus simple et stable en service système
- Évite les problèmes de permissions complexes

La vérité vraie: j'ai juste pas réussi à faire marcher l'image prometheus-podman-exporter :D Si quelqu'un a déjà réussi à la faire marcher qu'il me dise comment svp.

Installation

```
# Installer depuis les dépôts AlmaLinux
sudo dnf install -y prometheus-podman-exporter
```

Service Podman API (prérequis)

Créer le service pour exposer l'API Podman :

```
mkdir -p ~/.config/systemd/user/

cat > ~/.config/systemd/user/podman-api.service << 'EOF'
[Unit]
Description=Podman API Service
Documentation=man:podman-system-service(1)
After=network.target

[Service]
Type=exec
KillMode=process
ExecStart=/usr/bin/podman system service --time=0

[Install]
WantedBy=default.target
EOF
```

```
# Activer et démarrer
systemctl --user daemon-reload
systemctl --user enable --now podman-api

# Vérifier
systemctl --user status podman-api

# Vérifier que le socket existe
ls -lah /run/user/$(id -u)/podman/podman.sock
```

Service Podman Exporter

```
cat > ~/.config/systemd/user/prometheus-podman-exporter.service << EOF
[Unit]
Description=Prometheus Podman Exporter
After=podman-api.service
Requires=podman-api.service

[Service]
Type=simple
Environment="CONTAINER_HOST=unix:///run/user/$(id -u)/podman/podman.sock"
ExecStart=/usr/bin/prometheus-podman-exporter
Restart=always
RestartSec=5

[Install]
WantedBy=default.target
EOF

# Activer et démarrer
systemctl --user daemon-reload
systemctl --user enable --now prometheus-podman-exporter

# Vérifier
systemctl --user status prometheus-podman-exporter
```

Test

```
# Métriques disponibles
curl http://localhost:9882/metrics | grep podman_container
```

```
# Vérifier dans Prometheus
curl -G http://localhost:9091/api/v1/query \
  --data-urlencode 'query=podman_container_info' | jq
```

Configuration Nginx reverse proxy

DNS OVH

Ajouter l'enregistrement A :

```
Type    Nom       Valeur          TTL
A       grafana   <IP_VPS>        3600
```

Certificat Let's Encrypt

Étendre le certificat existant :

```
sudo certbot certonly --nginx \
  -d sanjy.fr \
  -d www.sanjy.fr \
  -d vault.sanjy.fr \
  -d grafana.sanjy.fr \
  --expand
```

Configuration Nginx

Fichier `/etc/nginx/conf.d/grafana.conf` :

```
server {
    listen 80;
    server_name grafana.sanjy.fr;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
```

```
    server_name grafana.sanjy.fr;

    # SSL
    ssl_certificate /etc/letsencrypt/live/sanjy.fr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/sanjy.fr/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers on;

    # Headers sécurité
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;

    # Logs
    access_log /var/log/nginx/grafana-access.log;
    error_log /var/log/nginx/grafana-error.log;

    # Proxy vers Grafana
    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # WebSocket support
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";

        # Timeouts
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }
}
```

Test et activation

```
# Tester la configuration
sudo nginx -t

# Recharger Nginx
sudo systemctl reload nginx

# Fix SELinux si nécessaire
sudo setsebool -P httpd_can_network_connect 1

# Vérifier
curl -I https://grafana.sanjy.fr
```

Accès final

```
https://grafana.sanjy.fr
```

Dashboard Grafana

Connexion

- **URL :** `https://grafana.sanjy.fr`
- **Login :** `admin`
- **Password :** `password_de_zinzin!`

Dashboard "Services Status" (custom)

Création manuelle :

1. **Menu → Dashboards → New Dashboard**
2. **Add visualization**
3. **Select Prometheus datasource**

Query :

```
podman_container_info{name=~"jenkins|vault|bookstack"} * on(id) group_left(name)
podman_container_state
```

Configuration :

- **Visualization type :** Stat
- **Legend :** `{{name}}`
- **Color mode :** Background
- **Text mode :** Value and name

Value mappings :

- `1` → Text: `UP ✓` → Color: Green
- `0` → Text: `DOWN ✗` → Color: Red

Résultat : Briques vertes/rouges indiquant l'état de chaque service.

Dashboards communautaires (tentative)

IDs testés mais non fonctionnels :

- **1860** : Node Exporter Full (problème de compatibilité)
- **15798** : Podman Container Metrics (problème de labels)
- **19972** : Podman Exporter (problème de compatibilité)

Raison : Les dashboards communautaires attendent des labels ou métriques spécifiques qui ne correspondent pas exactement à la configuration.

Solution : Création de dashboards custom adaptés aux métriques disponibles.

Conclusion

Infrastructure monitoring complète déployée avec succès.

Points clés :

- Prometheus + Grafana en conteneurs Podman rootless
- Node Exporter pour métriques système
- Podman Exporter en service systemd user (solution stable pour rootless)
- Reverse proxy Nginx avec SSL Let's Encrypt
- Dashboard custom pour monitoring des services critiques
- Accès sécurisé via `https://grafana.sanjy.fr`

Stack complètement opérationnelle et prête pour la production.

Prochaines étapes possibles :

- Ajouter Alertmanager pour les notifications
- Configurer des alertes (services down, seuils CPU/RAM)
- Ajouter Nginx Exporter pour monitoring web
- Créer des dashboards additionnels
- Automatiser les backups Grafana/Prometheus

Toujours et encore des problèmes de zinzin + troubleshooting

Problèmes rencontrés et solutions

Problème 1 : Port 9090 déjà utilisé

Erreur :

```
Error: unable to start container: rootlessport listen tcp 0.0.0.0:9090: bind: address already in use
```

Cause : Le port 9090 était utilisé par systemd (service `websm`).

Solution :

```
# Vérifier qui utilise le port
sudo lsof -i :9090

# Changer le port de Prometheus dans docker-compose.yml
ports:
  - "9091:9090"  # Port hôte 9091, conteneur 9090
```

Problème 2 : Erreur `.containerenv` manquant

Le problème qu'il y avait avec cadvisor (au début je voulais l'utiliser à la place de l'image prometheus-podman-exporter)

Erreur :

```
crun: open `/home/almalinux/.local/share/containers/storage/overlay/.../merged/run/.containerenv`:
```

```
No such file or directory
```

Cause : Problème avec le storage overlay de Podman après des suppressions brutales de conteneurs.

Solutions testées :

1. **Ajout de tmpfs (non retenu - contournement) :**

```
   tmpfs:
     - /run
```

2. **Reset du storage Podman :**

```
   podman system prune -a --volumes
```

3. **Reboot du serveur:**

```
   sudo reboot
```

4. **J'ai dégagé cadvisor pour autre chose (solution finale) :**

Le reboot nettoie les mounts overlay résiduels et recrée un état propre.

Problème 3 : Prometheus ne trouve pas sa config

Erreur :

```
Error loading config: open /etc/prometheus/prometheus.yml: no such file or directory
```

Cause :

1. Fichier créé avec extension `.yaml` au lieu de `.yml`
2. Le volume mount ne fonctionnait pas correctement

Solutions :

```
# Vérifier que le fichier existe
ls -lah ~/monitoring/prometheus/config/prometheus.yaml


# Utiliser la bonne extension dans la command du podman-compose.yaml
```

```
command:
- '--config.file=/etc/prometheus/prometheus.yaml'
et non
command:
- '--config.file=/etc/prometheus/prometheus.yml'
```

Problème 4 : Podman Exporter - Permission denied sur socket

Erreur :

```
unable to connect to Podman socket: dial unix /run/user/1000/podman/podman.sock:
connect: permission denied
```

Causes multiples :

1. **Socket n'existe pas**
2. **Service en conteneur ne peut pas accéder au socket rootless**
3. **Plusieurs instances en conflit**

Diagnostic :

```
# Vérification que le socket existe
ls -lah /run/user/$(id -u)/podman/podman.sock

# Vérification des process en cours
ps aux | grep prometheus-podman-exporter

# Vérifier qui utilise le port
sudo lsof -i :9882
```

Solution finale : Service systemd user

```
# Désactiver toutes les instances
sudo systemctl disable --now prometheus-podman-exporter
systemctl --user disable --now prometheus-podman-exporter
sudo pkill -9 -f prometheus-podman-exporter

# Créer le service Podman API
systemctl --user enable --now podman-api
```

```
# Créer le service exporter
systemctl --user enable --now prometheus-podman-exporter

# Vérifier qu'il n'y a qu'une instance
ps aux | grep prometheus-podman-exporter | grep -v grep
```

Pourquoi cette solution ?

- Le service user a accès direct au socket rootless
- Pas de problème de mapping UID/permissions
- Plus stable que l'approche conteneur pour rootless

En fait du faire une bêtise à un moment et lancer podman et d'autres en sudo ça m'a mis dans la sauce.

Problème 5 : Prometheus ne peut pas scraper localhost depuis le conteneur

Erreur : Targets `podman_exporter` et `node_exporter` en état "down".

Cause : Prometheus (conteneur) ne peut pas atteindre `localhost:9882` (hôte).

Diagnostic :

```
# Depuis l'hôte, ça marche
curl http://localhost:9882/metrics

# Mais Prometheus (conteneur) ne voit pas localhost de l'hôte
```

Solution : Utiliser `host.containers.internal`

```
# Dans prometheus.yaml
- job_name: 'podman_exporter'
  static_configs:
    - targets: ['host.containers.internal:9882']

# Dans docker-compose.yml
prometheus:
  extra_hosts:
    - "host.containers.internal:host-gateway"
```

Résultat : Tous les targets passent à "up"

Problème 6 : Labels `name` manquants dans podman_container_state

Observation :

```
podman_container_state{id="abc123", ...} 1
```

Pas de label `name` !

Cause : Podman Exporter ne retourne pas toujours tous les labels dans toutes les métriques.

Solution : Utiliser podman_container_info et parser les infos

```
# Query qui contient tout
podman_container_info
```

Problème 7 : Dashboards communautaires vides

Cause : Les dashboards attendent des métriques ou labels spécifiques qui ne correspondent pas exactement.

Solution : Créer des dashboards custom adaptés aux métriques réellement disponibles.

Méthode :

1. Tester les queries dans Explore
2. Identifier les labels disponibles
3. Créer les panels manuellement
4. Adapter les value mappings

Commandes de maintenance

Gestion des services

```
# Status de tous les services
podman ps
systemctl --user status podman-api
systemctl --user status prometheus-podman-exporter

# Restart complet du stack
cd ~/monitoring
podman-compose restart

# Logs
podman-compose logs -f prometheus
podman-compose logs -f grafana
journalctl --user -u prometheus-podman-exporter -f
```

Vérification des métriques

```
# Node Exporter
curl http://localhost:9100/metrics | grep node_cpu

# Podman Exporter
curl http://localhost:9882/metrics | grep podman_container

# Prometheus targets
curl http://localhost:9091/api/v1/targets | jq '.data.activeTargets[] | {job: .labels.job,
health: .health}'

# Query spécifique
curl -G http://localhost:9091/api/v1/query \
  --data-urlencode 'query=podman_container_info' | jq
```

Reload configuration Prometheus

```
# Sans restart
podman exec prometheus kill -HUP 1

# Ou restart complet
podman restart prometheus
```

Ressources

Documentation

- **Prometheus :** https://prometheus.io/docs/
- **Grafana :** https://grafana.com/docs/
- **Node Exporter :** https://github.com/prometheus/node_exporter
- **Podman Exporter :** https://github.com/containers/prometheus-podman-exporter

Queries PromQL utiles que j'ai trouvées

Nombre de conteneurs :

```
count(podman_container_info)
```

Conteneurs par état :

```
sum by (state) (podman_container_state)
```

CPU usage par conteneur :

```
rate(podman_container_cpu_seconds_total[5m])
```

Memory usage :

```
podman_container_mem_usage_bytes
```

System CPU usage :

```
100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)
```

System memory usage :

```
(1 - (node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes)) * 100
```

Disk usage :

```
100 - ((node_filesystem_avail_bytes{mountpoint="/",fstype!="rootfs"} /
node_filesystem_size_bytes{mountpoint="/",fstype!="rootfs"}) * 100)
```

Configuration réseau complète

Ports utilisés :

```
80      HTTP (redirect HTTPS)
443     HTTPS (Nginx)
3000    Grafana (localhost)
8080    Nginx status (localhost)
9091    Prometheus (localhost)
9100    Node Exporter (localhost)
9113    Nginx Exporter (localhost, optionnel)
9882    Podman Exporter (localhost)
```

Firewall nftables :

```
# Monitoring (localhost only pas besoin de les rajouter dans nft)
tcp dport { 3000, 9091, 9100, 9882 } ip saddr 127.0.0.1 accept

# HTTPS public
tcp dport { 80, 443 } accept
```

Checklist de vérification

- ☐ Tous les conteneurs UP : `podman ps`
- ☐ Services systemd actifs : `systemctl --user status podman-api prometheus-podman-exporter`
- ☐ Socket Podman existe : `ls /run/user/$(id -u)/podman/podman.sock`
- ☐ Prometheus targets UP : `curl http://localhost:9091/api/v1/targets | jq`
- ☐ Grafana accessible : `https://grafana.sanjy.fr`
- ☐ Dashboard services fonctionne
- ☐ Certificat SSL valide
- ☐ Logs propres : `podman-compose logs --tail=50`
