Troubleshoot

? Problèmes rencontrés et solutions

1. Erreur "invalid_redirect_uri" avec port 30180

Symptôme : Keycloak génère des URLs avec `http://sso.sanjy.fr:30180` au lieu de `https://sso.sanjy.fr`

Cause : Keycloak ne prend pas en compte les headers du reverse proxy

Solution :

- Ajouter `KC_PROXY_HEADERS: xforwarded` dans la config Keycloak
- Ajouter `KC_HOSTNAME_URL: https://sso.sanjy.fr` pour forcer l'URL frontend
- S'assurer que Nginx envoie les headers `X-Forwarded-Proto: https` et `X-Forwarded-Port: 443`

2. Redirect URI en `localhost:3000`

Symptôme : L'URL de redirection OAuth contient `http://localhost:3000` au lieu de `https://grafana.sanjy.fr`

Cause : Grafana ne connaît pas son URL publique

Solutions :

- Ajouter `GF_SERVER_ROOT_URL=https://grafana.sanjy.fr` dans la config Grafana
- Ajouter `GF_SERVER_DOMAIN=grafana.sanjy.fr`
- **Retirer** `proxy_redirect off;` dans la config Nginx (empêche les réécritures)
- Redémarrer complètement Grafana avec `podman-compose down` puis `up`

3. Erreur "Page not found" sur le realm

Symptôme : Erreur 404 lors de l'accès aux endpoints Keycloak

Cause : Le nom du realm dans la config ne correspond pas au nom réel dans Keycloak

Solution :

- Vérifier le nom exact du realm dans l'interface Keycloak (en haut à gauche)
- S'assurer que ce nom est utilisé partout dans les configs Grafana
- Tester avec : `curl https://sso.sanjy.fr/realms/VOTRE_REALM/.well-known/openid-configuration`

4. Erreur "connection refused" sur TOKEN_URL

Symptôme : Grafana ne peut pas échanger le code OAuth contre un token

Cause : Grafana essaie d'accéder à `https://sso.sanjy.fr` depuis le conteneur mais ne peut pas atteindre l'IP publique

Solution :

- Créer un réseau Podman partagé entre Keycloak et Grafana
- Utiliser `http://keycloak:8080` pour `TOKEN_URL` et `API_URL` (communication inter-conteneurs)
- Garder `https://sso.sanjy.fr` uniquement pour `AUTH_URL` (utilisé par le navigateur)

Vérification de la communication :

```
# Tester depuis le conteneur Grafana
podman exec grafana curl http://keycloak:8080/realms/sanjy.fr/.well-known/openid-configuration
```

5. Données Keycloak non persistées

Symptôme : Le realm et les utilisateurs disparaissent après `podman-compose down/up`

Cause : Volume non persistant ou supprimé avec `podman-compose down -v`

Solution :

- Utiliser un volume nommé dans `podman-compose.yaml` :

```
volumes:
  - data:/opt/keycloak/data


volumes:
  data:
```

- **Ne jamais utiliser** `podman-compose down -v` (le `-v` supprime les volumes)
- Vérifier que le volume existe : `podman volume ls`

6. Permissions refusées sur bind mount

Symptôme : Erreur Java "Error while creating file" dans les logs Keycloak

Cause : User namespace mapping de Podman en rootless (UID sur l'hôte ≠ UID dans le conteneur)

Solution abandonnée : Bind mount avec `./data:/opt/keycloak/data:Z`

Solution retenue : Utiliser un volume nommé Podman qui gère automatiquement les permissions :

```
volumes:
  - data:/opt/keycloak/data
```

7. Préfixe réseau `keycloak_sso` au lieu de `sso`

Symptôme : Le réseau créé s'appelle `keycloak_sso` au lieu de `sso`

Cause : `podman-compose` préfixe automatiquement avec le nom du répertoire du projet

Solution : Forcer le nom exact dans la définition du réseau :

```
networks:
  sso:
    name: sso
    driver: bridge
```

? Vérifications finales

Vérifier Keycloak

```
# Endpoint OIDC du realm
curl https://sso.sanjy.fr/realms/sanjy.fr/.well-known/openid-configuration

# Doit retourner du JSON avec issuer, authorization_endpoint, etc.
```

Vérifier le réseau Podman

```
# Lister les réseaux
podman network ls | grep sso

# Inspecter le réseau
podman network inspect keycloak_sso
```

Vérifier la communication inter-conteneurs

```
# Depuis Grafana vers Keycloak
podman exec grafana curl http://keycloak:8080/realms/sanjy.fr/.well-known/openid-configuration

# Doit retourner du JSON (pas d'erreur de connexion)
```

Vérifier le volume Keycloak

```
# Lister les volumes
podman volume ls

# Inspecter le volume
podman volume inspect keycloak_data  # ou le nom de votre volume
```

Vérifier les logs

```
# Logs Keycloak
podman logs keycloak

# Logs Grafana (chercher "oauth", "keycloak")
podman logs grafana | grep -i oauth
```

? Points clés à retenir

URLs publiques vs URLs internes

- **AUTH_URL** : HTTPS public (`https://sso.sanjy.fr`) - utilisé par le navigateur
- **TOKEN_URL** et **API_URL** : HTTP interne (`http://keycloak:8080`) - communication serveur à serveur

Réseau Podman

- Les conteneurs sur le même réseau peuvent communiquer via leurs noms
- Utiliser `keycloak:8080` depuis Grafana au lieu de `localhost:30180`

Volumes Podman

- Les volumes nommés sont plus fiables que les bind mounts en rootless
- Ne jamais utiliser `podman-compose down -v` pour éviter de supprimer les données
- Backup : `podman volume export keycloak_data -o backup.tar`

Headers Nginx essentiels

```
proxy_set_header X-Forwarded-Proto https;
proxy_set_header X-Forwarded-Host $host;
proxy_set_header X-Forwarded-Port 443;
```

Ces headers sont essentiels pour que Keycloak génère les bonnes URLs.

Ne JAMAIS utiliser `proxy_redirect off;`

Cette directive empêche Nginx de réécrire les redirections, ce qui casse OAuth/OIDC.

Revision #3
Created 2025-12-02 01:56:14 UTC by Admin
Updated 2025-12-02 01:57:39 UTC by Admin