

Gérer les accès PostgreSQL avec LDAP et ldap2pg

Alexandre Pereira

Le besoin de Bridoulou

Bridoulou veut gérer simplement les accès à sa base PostgreSQL :

- les comptes et groupes sont maintenus dans LDAP ;
- les droits sur les données sont gérés dans PostgreSQL ;
- l'équipe DBA veut éviter les créations et révocations manuelles répétitives.

Les briques et leurs rôles

Brique	Rôle dans le scénario
LDAP	comptes et groupes
<code>pg_hba.conf</code>	méthode d'authentification à la connexion
Tableau d'habilitation (csv)	correspondance groupes / schémas / droits
gomplate	génération du YAML depuis le modèle et le CSV
ldap2pg	synchronisation des rôles et privilèges PostgreSQL

LDAP authentifie. ldap2pg aligne les rôles et les droits.

Authentifier un compte LDAP depuis PostgreSQL

Extrait de `pg_hba.conf` :

```
# Réseau des postes Bridoulou : authentication des comptes via LDAP
host      all      all      10.42.0.0/16      ldap ldapserver=ldap.bridoulou.local ldapport
```

PostgreSQL interroge LDAP pour vérifier le mot de passe fourni à la connexion.

Authentification et gestion des rôles

- `pg_hba.conf` décide comment PostgreSQL vérifie l'identité.
- `ldap2pg` crée et met à jour les rôles PostgreSQL et leurs appartenances aux groupes.
- Les privilèges PostgreSQL déterminent l'accès aux bases, schémas et objets.

Une authentification LDAP réussie ne donne pas automatiquement de droits sur les données.

Ce que ldap2pg apporte

ldap2pg rapproche deux états :

1. l'état désiré, décrit par la configuration;
2. l'état constaté dans PostgreSQL.

Il peut ensuite gérer les rôles, leurs appartenances et les privilèges déclarés.

On inspecte le dry-run avant d'appliquer avec `--real`.

Les groupes de l'annuaire Bridoulou

Deux groupes LDAP illustrent le modèle :

- `bdl_writers` : Alice, Bob ;
- `bdl_readers` : Charly, Sophie.

Observer les groupes et leurs membres

```
ldapsearch -x -LLL -H ldaps://ldap.bridoulou.local \\  
-D 'cn=Administrateur,cn=Users,dc=bridoulou,dc=local' -W \\  
-b 'cn=Users,dc=bridoulou,dc=local' \\  
'(|(cn=bdl_readers)(cn=bdl_writers))' \\  
cn member
```

ldapsearch affiche le nom des groupes et le DN de leurs membres.

Décrire les habilitations dans un CSV

```
schema;role;privilege  
atelier_demo;bdl_readers;lecture  
atelier_demo;bdl_writers;ecriture
```

Le CSV relie chaque groupe à un schéma et à un profil de privilèges.

Du modèle au YAML avec Gomplate

```
gomplate --file=config/ldap2pg.yaml.tpl  
--datasource=habilitations=file://config/habilitations.csv  
--out=config/ldap2pg.yaml
```

Gomplate injecte les données du CSV et les paramètres LDAP dans le modèle.

Lire la configuration rendue

Avant l'exécution, contrôler :

- la recherche LDAP et le filtre des groupes ;
- la liste des rôles PostgreSQL gérés ;
- les règles de privilèges issues du CSV.

Simuler les changements

```
ldap2pg --config generated/ldap2pg.yaml
```

Le mode par défaut est le dry-run : il présente les changements proposés sans les appliquer.

Appliquer puis vérifier

Après validation du dry-run :

```
ldap2pg --config generated/ldap2pg.yaml --real
```

Puis vérifier les rôles et les droits :

```
psql -U postgres -d atelier -c '\du'  
psql -U postgres -d atelier -c '\dp atelier_demo.*'
```

À retenir

- `pg_hba.conf` contrôle l'authentification ;
- LDAP porte les identités et appartenances ;
- CSV + Gomplate produisent la configuration déclarative ;
- `ldap2pg` synchronise les rôles et privilèges PostgreSQL ;
- dry-run, contrôle, puis application.

Questions / discussion