Synchroniser des rôles Postgres depuis un annuaire LDAP

ldap2pg

DALIBO
L'expertise PostgreSQL

PGSession 9 - 17 novembre 2017

Étienne BERSAC @Dalibo

ldap2pg

Synchroniser des rôles Postgres depuis un annuaire LDAP

PGSession 9 - 17 novembre 2017

ldap2pg
Synchroniser des rôles Postgres depuis un annuaire LDAP

DATE : PGSession 9 - 17 novembre 2017

SOMMAIRE

Dans un premier temps, on abordera la problématique, l'état des lieux.

Ensuite, on verra l'outil ldap2pg lui-même : ce qui a guidé sa conception, ses fonctionnalités.

Enfin on jouera une démonstration en direct !

POSTGRES & LDAP

- industrialisation des rôles
- pg_hba.conf
- création des rôles ?

À quoi sert LDAP aujourd'hui ?

LDAP dans PostgreSQL : intégration dans le pg_hba.conf mais quid des rôles ?

Dans la documentation :

the user must already exist in the database before LDAP can be used for authentication.

Créer les rôles avec les bonnes options, quid des privilèges ?

ÉTAT DES LIEUX

- pg-ldap-sync
- script ad-hoc

pg-ldap-sync : projet ruby, peu maintenu depuis 2011. Apporte l'idée de fichier YML. Pensé avec les anciens concepts `USER`/`GROUP` de Postgres (pré 8.1). On peut faire seulement deux requêtes LDAP : une pour les utilisateurs, une pour les groupes. En outre, pg-ldap-sync masque les requêtes d'instrospection de rôles : on se retrouve à injecter des bouts de WHERE dans le YAML, difficilement testable dans psql. Pas d'`ALTER`.

script ad-hoc : demande beaucoup de temps et de compétences pour éprouver le script, l'adapter aux nouvelles situations.

BESOINS

- simple
- souple
- fiable

simple : par opposition au script ad-hoc : permettre au DBA de se concentrer sur le besoin fonctionnel et pas sur la complexité de LDAP et de la synchronisation. Mais aussi : peu de dépendances, pas d'abstractions.

souple : pg-ldap-sync manque de souplesse tandis que le script ad-hoc apporte énormément de souplesse.

fiable : les rôles, c'est sensible. On veut un outils bien testé, portable. Fonctionnellement on veut aussi tester sans toucher, avoir des informations claires sur ce qui est fait/à faire.

LDAP2PG

- portable, standards
- ldap2pg.yml
- dry run et check mode
- afficher requêtes SQL et recherches LDAP

portable : dépendances : pyldap (du projet OpenLDAP), psycopg2 et pyyaml

ldap2pg.yml : on reprend l'idée d'un fichier YAML expressif. Le fichier permet de rédiger directement les requêtes SQL, qu'on peut tester dans psql.

fiable : par défaut en dry run. le check-mode permet de remonter les problèmes de synchronisation. Chaque requête SQL est loggué, même en dry run.

bonus : les requêtes LDAP sont affichée dans les logs au format commande `ldapsearch` pour aider au débug. ldap2pg est un bon compagnon de psql et ldapsearch.

ÉTAPES

- inventaires Postgres et LDAP
- comparaison
- synchronisation

Comment se passe la synchronisation des rôles ?

1. faire l'inventaire des rôles dans Postgres et des rôles à créer.
2. comparer les deux inventaires.
3. générer des requêtes `DROP`, `ALTER` et `CREATE`.

INVENTAIRE LDAP

- requêtes LDAP
- extraction d'information, condition
- options et hiérarchie

Comment faire l'inventaire des rôles à créer depuis un annuaire ?

1. on exécute les requêtes (c'est là qu'on log l'équivalent ldapsearch)
2. on applique des règles d'extraction d'information définie par l'utilisateur dans le fichier YAML.

On peut extraire depuis une entrée de l'annuaire : nom de rôle et appartenance (rôle parent ou enfant). Dans le fichier YML, on peut définir des options de CREATE ROLE.

EXEMPLE

Remarquer :

- la requêtes LDAP dans `ldap`, avec la sémantique `ldapsearch`
- les règle d'extraction dans `roles`.
- `*_attribute` indique un référence à un attribut de chaque entrée.
- Expliquer les boucles : ldap2pg va extraire chaque valeur de l'attribut de chaque entrée de l'annuaire.
- noter la création explicite des `member` avec l'option `LOGIN`.

Dans cet exemple : on crée un groupe de DBA. L'option `SUPERUSER` est sur le groupe. L'option `LOGIN` est sur les membres.

ACL

- inspect, grant, revoke
- gestion par base, par schéma
- SQL brut

Fonctionalité jumelle et indépendante de la synchronisation de rôles.

Toujours le tryptique inspection, creation/suppression. Pas d'ALTER dans le cas des ACL.

`ldap2pg` aide à appliquer les ACL à plusieurs bases et plusieurs schémas.

`ldap2pg` ne propose pas d'ACL prédéfinie. Son utilité est uniquement de simplifier la glue introspection / synchronisation. Inspecter les ACL demande une bonne connaissance des catalogues systèmes. C'est pourtant là que c'est intéressant, ça permet d'auditer qui a droit à quoi dans un cluster.

On peut utiliser ldap2pg pour synchroniser les ACL et pas les rôles et vice-versa.

AUDIT

- comparaison des inventaires
- requiert une introspection parfaite des ACL
- intégration avec la supervision

Avec le mode audit, `ldap2pg` se limite à comparer l'inventaire de l'instance et l'inventaire de l'annuaire. C'est le mode par défaut.

Pour les ACL, l'inventaire est une opération délicate. Certaines ACL étant implicites. C'est également délicat avec le role et le schéma public.

Le mode contrôle facilite l'intégration dans une sonde de supervision ou dans un outil d'audit, un peu comme l'outil `diff` qui retourne un code d'erreur en cas de différences.

DÉMO !

- Présenter les trois services : `docker-compose ps`
- présenter l'annuaire, les infos qu'on veut retrouver dans Postgres : http ://ldapadmin.ldap2pg.docker
- Présenter l'existant dans l'instance : `psql -l SELECT * FROM information_schema.enabled_roles`
- Montrer la connexion à l'annuaire, standard : `ldapwhoami -x -w ${LDAPPASSWORD}`
- Présenter ldap2pg : `ldap2pg --version ldap2pg --help`
- Rédiger ldap2pg.yml
- Lancer par défaut : `ldap2pg ldap2pg --verbose ldap2pg --check ldap2pg --real ldap2pg --check`
- Vérifier l'existence des rôles : `psql -U alan postgres psql -U oscar postgres`

MERCI ! QUESTIONS ?