2025/11/19 11:54 1/13 Configure Postgresql

Configure Postgresql

Déplacer le dossier des données de Postgresql



ATTENTION: il n'est finalement pas conseillé de déplacé la base de données sur le volume additionnel car ce dernier à des performances bien en dessous du disque principal du serveur. Le volume additionnel, type HIGH SPEED, affiche 42.91 IOPS en moyenne en lecture contre 5575.55 pour le disque principal!

- Installer les utilitaires : apt install rsync
- Vérifier l'emplacement actuel :
 - Accéder à Psql : psql
 - Lancer la requête : SHOW data directory;
 - Généralement le dossier est : /var/lib/postgresql/11/main
- Déplacer les données :
 - ∘ Stopper *Pgsql*: systemctl stop postgresql
 - Vérifier l'arrêt: systemetl status postgresql
 - Déplacer les données à l'aide de rsync pour préserver les droits (ATTENTION : noter l'absence de / final au 2 chemins) :
 - Tester avec l'options --dry-run : rsync -av /var/lib/postgresql /data --dry-run
 - Si tout semble OK, lancer le transfert --dry-run : rsync -av /var/lib/postgresql /data
 - Renommer l'ancien dossier: mv /var/lib/postgresql/11/main /var/lib/postgresql/11/main.bak
- Configurer le nouvel emplacement :
 - Éditer la config : vi /etc/postgresql/11/main/postgresql.conf
 - Modifier la valeur de la ligne : data directory = '/data/postgresql/11/main'
 - Redémarrer *Postgresql* et vérifier :
 - systemctl start postgresql
 - systemctl status postgresql
 - psql
 - SHOW data directory;
- Si tout est OK, supprimer l'ancien emplacement devenu inutile (garder /var/lib/postgresql qui est le home de l'utilisateur postgres) : cd /var/lib/postgresql/; rm -fR 11/

Configurer les utilisateurs pour Postgresql

- Utilisateurs :
 - postgres : superutilisateur ; psql.
 - admin : superutilisateur ; psql.
 - o geonatadmin : admin et propriétaire des bases geonature2db et gnatlas.
 - o geonatatlas : utilisateur de la base gnatlas.
 - telegraf: utilisateur permettant l'accès à la base postgres par l'utilitaire Telegraf (Docker).

- Créer pour l'utilisateur système admin le fichier d'initialisation : vi ~/.psqlrc
 - Y placer la commande d'activation du chronométrage des requêtes SQL :

\timing

- Se connecter au compte admin, et basculer sur le compte de l'utilisateur postgres : sudo -i u postgres
 - Créer l'utilisateur admin et son mot de passe à l'aide de psql: psql -c "CREATE ROLE admin WITH LOGIN PASSWORD '<mot-de-passe>';"
 - Attribuer les droits de super-utilisateur : psql c "ALTER USER admin WITH SUPERUSER CREATEDB CREATEROLE REPLICATION;"
 - Créer une base de données du nom de l'utilisateur pour qu'il puisse accéder à la commande psql : createdb -E UTF8 -O admin admin
 - Se déconnecter de l'utilisateur postgres : exit
- Depuis le compte de l'utilisateur admin :
 - Tester l'accès à Psql : psql
 - Puis en sortie : \q
 - Créer l'utilisateur Postgresql geonatadmin et la base de données de GeoNature :
 - Créer l'utilisateur geonatadmin pour GeoNature : psql -c "CREATE ROLE geonatadmin WITH LOGIN PASSWORD '<mot-de-passe>';"
 - Ajouter geonatadmin au rôle autorisant la lecture des fichiers pour la commande SQL COPY: GRANT pg_read_server_files T0 geonatadmin;
 - Le script d'installation créés la base de données de GeoNature (au cas ou la commande : createdb - E UTF8 - O geonatadmin geonature2db)
 - Créer l'utilisateur *Postgresgl geonatatlas* et la base de données de *GeoNature Atlas* :
 - Créer l'utilisateur geonatatlas pour GeoNature : psql -c "CREATE ROLE geonatatlas WITH LOGIN PASSWORD '<mot-de-passe>';"
 - Le script d'installation créés la base de données de GeoNature Atlas (au cas ou la commande : createdb - E UTF8 - O geonatatlas gnatlas)
 - Créer l'utilisateur Postgresql telegraf :
 - Créer l'utilisateur telegraf pour l'accès à la base postgres : psql -c "CREATE ROLE telegraf WITH LOGIN PASSWORD '<mot-de-passe>';"
 - avec accès :
 - à la base postgres seulement (mode basique), ajouter le droit de connexion à la base postgres avec : psql -c "GRANT CONNECT ON DATABASE postgres TO telegraf;"
 - à toutes les bases (mode avancé) : psql -c "ALTER USER telegraf SUPERUSER CONNECTION LIMIT 3;"
 - Créer, si nécessaire, l'utilisateur Postgreqsl gnreader qui a des droits d'accès lecture seule. Voir la documentation correspondante.

Rendre accessible le port 5432 depuis l'instance web-srv

 Créer un fichier qui surchargera postgresql.conf avec : vi /etc/postgresql/12/main/conf.d/01_override.conf pour ouvrir le port 5432 sur l'IP privé de l'instance db-srv avec la propriété :

```
listen_addresses = 'localhost,10.0.1.20,172.18.5.1'
```

- Modifier le fichier : vi /etc/postgresql/12/main/pg hba.conf pour :
 - ajouter l'accès depuis l'IP privé dans la section "IV4 local connections" :

```
# IPv4 local connections:
...
host all all 10.0.1.20/32
scram-sha-256
```

• ajouter les autorisations de l'instance Web à accéder en ajoutant en fin de fichier :

# GeoNature				
host geonature2db	geonatadmin	10.0.1.10/32		
scram-sha-256				
host gnatlas	geonatadmin	10.0.1.10/32		
scram-sha-256				
host gnatlas	geonatatlas	10.0.1.10/32		
scram-sha-256				
host postgres	telegraf	172.18.5.0/24		
scram-sha-256				

- redémarrer le service Postgresql pour prendre en compte les changements : systemctl restart postgresql
- Vérifier les ports ouverts :
 - ∘ ss -tunelp | grep 5432 doit afficher:

```
LISTEN 0
                               172.18.5.1:5432
                                                     0.0.0.0:*
tcp
                    128
uid:108 ino:18316865 sk:11 <->
                              10.0.1.20:5432
                                                     0.0.0.0:*
     LISTEN 0
                    128
uid:108 ino:498249 sk:12 <->
      LISTEN 0
                    128
                               127.0.0.1:5432
                                                     0.0.0.0:*
uid:108 ino:498248 sk:13 <->
tcp
      LISTEN 0
                    128
                                   [::1]:5432
                                                         [::]:*
uid:108 ino:498247 sk:14 v6only:1 <->
```

∘ netstat -anpt|grep :5432 doit afficher:

0	0 172.18.5.1:5432	0.0.0.0:*
29707	/postgres	
0	0 10.0.1.20:5432	0.0.0.0:*
31259/postgres		
0	0 127.0.0.1:5432	0.0.0.0:*
31259/postgres		
0	0 ::1:5432	*
31259	/postgres	
	29707 0 31259 0 31259 0	29707/postgres 0 0 10.0.1.20:5432 31259/postgres 0 0 127.0.0.1:5432 31259/postgres

- Tester depuis l'instance *web-srv* l'accès au port 5432 sur l'IP privé du VPN : telnet 10.0.1.20 5432
- Tester ensuite avec *DBeaver* en créant une nouvelle connexion avec tunnel *SSH* depuis l'instance *web-srv* vers l'instance *db-srv*. Cliquer sur le bouton "Tester la connexion".

Configurer le parefeu pour Postgresql

- Au préalable, voir la configuration du parefeu Firewalld dans la configuration commune.
- Si besoin, voici comment avec *Nftables* ajouter l'ouverture du port 5432 sur l'adresse 10.0.1.20
 - nft add rule ip filter INPUT ip daddr 10.0.1.20 tcp sport 1024-65535 tcp dport 5432 ct state new,established counter accept
 - nft add rule ip filter OUTPUT ip saddr 10.0.1.20 tcp sport 5432 tcp dport 1024-65535 ct state established counter accept

Optimiser la configuration de Postgresql vis à vis du serveur

Gestion des logs et des stats

• Commencer par modifier le fichier logrotate de Postgresql qui se chargera des rotations :

```
/var/log/postgresql/*.log {
    weekly
    rotate 10
    copytruncate
    delaycompress
    compress
    notifempty
    missingok
    su root root
}
```

• Nous allons surcharger de nouveaux paramètres de configuration de Postgresql (présents dans postgresql.conf) via le fichier 01_override.conf avec : vi /etc/postgresql/12/main/conf.d/01_override.conf

```
# General
shared_preload_libraries = 'pg_stat_statements,pg_prewarm'

# Log
log_destination = stderr
logging_collector = on
# Use logrotate for rotation. See: /etc/logrotate.d/postgresql-common
#log_rotation_age = 1d
#log_rotation_size = 100MB
# Log format for PgBadger. See:
https://pgbadger.darold.net/documentation.html#REQUIREMENT
log_line_prefix= '%t [%p]: db=%d,user=%u,app=%a,client=%h '
lc_messages = 'C.UTF-8'
log_autovacuum_min_duration = 0
log_checkpoints = on
log_connections = on
```

2025/11/19 11:54 5/13 Configure Postgresql

```
log_disconnections = on
log_lock_waits = on
log_temp_files = 0
log_error_verbosity = default

# Activity stats
track_activity_query_size = 20480
compute_query_id = on
track_io_timing = on
track_wal_io_timing = on
track_functions = 'pl'

# Extenssion pg_stat_statements config
pg_stat_statements.max = 10000
pg_stat_statements.track = all
```

Utilisation de PgTune

L'idée est d'adapter rapidement la configuration de Postgresql fournie par défaut aux caractéristiques de l'instance abritant le serveur Postgresql. Pour cela nous utiliserons le site https://pgtune.leopard.in.ua/.

- Se rendre sur : https://pgtune.leopard.in.ua/#/
 - Remplir le formulaire en fonction des caractéristiques de l'instance abritant la base de données.
 - Ex. SINP PACA:

```
# DB Version: 11
# OS Type: linux
# DB Type: web/data warehouse
# Total Memory (RAM): 15 GB
# CPUs num: 4
# Data Storage: ssd
```

■ Ex. SINP AURA:

```
# DB Version: 12
# OS Type: linux
# DB Type: web
# Total Memory (RAM): 15 GB
# CPUs num: 4
# Data Storage: ssd
```

- Copier la configuration générée
- **Note** : il est possible de comparer les résultats obtenus avec le type "data warehouse" et de réaliser un mixe...
- Se connecter à l'instance "db-srv" en tant qu'admin : ssh admin@db-<region>-sinp
 - ∘ Passer en root : sudo -i
 - Créer le fichier suivant : vi /etc/postgresql/15/main/conf.d/02_optimize.conf

• Y coller la configuration préalablement copier et enregistrer le fichier. Ex. :

```
# DB Version: 11
# OS Type: linux
# DB Type: web/data warehouse
# Total Memory (RAM): 15 GB
# CPUs num: 4
# Data Storage: ssd
max connections = 100
shared buffers = 3840MB
effective cache size = 11520MB
maintenance work mem = 1920MB
checkpoint_completion_target = 0.7
wal buffers = 16MB
default statistics target = 100
random page cost = 1.1
effective io concurrency = 200
work mem = 24MB
min wal size = 4GB
max wal size = 16GB
max_worker_processes = 4
max parallel workers per gather = 2
max parallel workers = 4
max parallel maintenance workers = 2
# ATTENTION : ne pas limiter
idle in transaction session timeout à 5mn10s car la mise à
jour de l'Atlas nécessite plus !
#idle in transaction session timeout=310000
```

- Redémarrer *Postgresql*: systemctl restart postgresql
- Vérifier la prise en compte d'une des nouvelles valeurs à l'aide la requête SQL suivante : SHOW <ma-variable>;

Utilisation de potgrestuner.pl

- Ressource : https://github.com/jfcoz/postgresqltuner
- Installer les dépendances : apt install libdbd-pg-perl libdbi-perl perl-modules
- Se connecter en tant qu'admin
- Se placer dans le dossier bin/ de l'admin : mkdir ~/bin/; cd ~/bin/
- Télécharger le script : wget 0 postgresqltuner.pl postgresqltuner.pl
- Donner les droits d'execution : chmod +x postgresqltuner.pl
- Recharger le shell : source ~/.bashrc
- Lancer le script (mettre un espace devant la commande pour éviter d'enregistrer le mot de passe dans l'historique) : postgresqltuner.pl —host=localhost —database=geonature2db —user=admin —password=<mot-de-passe>
- Etudier les consigne du script et modifier le fichier de config si nécessaire : vi /etc/postgresql/15/main/conf.d/02_optimize.conf

2025/11/19 11:54 7/13 Configure Postgresql

Configurer l'utilisation des Huges Pages

- Commencer par vérifier quel type de Huges Pages le sytème supporte : cat /proc/cpuinfo
 - Si pse est présent : indique le support des Huges Pages de 2MB
 - ∘ Si pdpe1gb est présent : indique le support des Huges Pages de 1GB
- Voir aussi la taille des Huges Pages avec le paramètre Hugepagesize dans la sortie de : cat /proc/meminfo
 - Si votre système supporte les pages de 1GB (pdpe1gb) et que Hugepagesize indique 2MB, vous pouvez changer cette valeur en suivant les indications de la page 34 de ce document PDF.
- Calculer le nombre de Huge Pages :
 - Récupérer le PID du processus principal de Postgresql : cat /data/postgresql/15/main/postmaster.pid ou cat /var/lib/postgresql/15/main/postmaster.pid c'est le nombre sur la première ligne.
 - Afficher le nombre de Ko max utilisé : grep ^VmPeak /proc/<pid-pgpostmaseter>/status, affiche par exemple VmPeak: 215424 kB
 - ∘ Calculer combien de Huge Pages de 2MB correspondent : 215424/1024/2 = 106
- Éditer le fichier /etc/sysctl.conf et ajouter le nombre de Huge Pages en l'augmentatn légèrement (ex. 110 pour 106) vm.nr_hugepages = 110
- Arrêter Postgresql: systemctl stop postgresql
- Actualiser le système : sysctl -p
- Éditer le fichier de surcharge de la conf de Postgresql : vi /etc/postgresql/15/main/conf.d/02_optimize.conf
 - o Ajouter : huge pages = on
 - Avec "on" s'il y a un problème avec les Huge Pages le serveur de démarrera pas. Sinon, utiliser "try".
- Afficher le nombre de Huge Pages reserved : grep ^Huge /proc/meminfo
 - ∘ Cela devrait afficher : HugePages_Rsvd: 0
- Démarrer Postgresql: systematl start postgresql
- Vérifier le statut car s'il y a un problème avec la conf le serveur ne démarrera pas : systemctl status postgresql
 - Si Postgresql ne veux pas démarrer à cause d'un manque de mémoire, augmenter le nombre de Huge Pages dans le fichier /etc/sysctl.conf en recommençant les étapes précédentes.
- Afficher le nombre de Huge Pages reserved : grep ^Huge /proc/meminfo
 - Cela devrait afficher un nombre supérieur à 0 : HugePages Rsvd: 64

Restaurer localement un dump de la base du serveur

- Nous partons du principe que l'utilisateur distant geonatadmin existe en local et que la base de données geonature2db n'existe pas en local ou peut être supprimée. Postgresql doit écouter en local (localhost) sur le port par défaut (= 5432). L'archive générée par pg_dump doit être au format directory (= "d").
 - Si nécessaire créer l'utilisateur geonatadmin localement avec la commande (changer le mot de passe) :

```
sudo -u postgres psql -c "CREATE ROLE geonatadmin WITH LOGIN
PASSWORD '<password-local>';"
```

Supprimer la base locale si elle existait préalablement :

```
sudo -u postgres dropdb --if-exists geonature2db
```

o En cas d'erreur ERREUR: la base de données « geonature2db » est en cours d'utilisation par d'autres utilisateurs : fermer les connexions à la base (DBeaver) ou utiliser la commande suivante pour forcer la fermeture de toutes les connexions à la base geonature2db :

```
sudo -u postgres psql -c "SELECT
pg_terminate_backend(pg_stat_activity.pid) FROM pg_stat_activity
WHERE pg stat activity.datname = 'geonature2db' AND pid <>
pg_backend_pid();"
```

• Afin de permette à l'utilisateur geonatadmin de recréer les extenssions dans la base, nous lui donnons temporairement (le temps de l'exécution de pg restore) des droits de super-utilisateur

```
sudo -u postgres psql -c "ALTER ROLE geonatadmin SUPERUSER;"
```

• Recréer une base de données vide (= avec le template template0) portant le même nom que la base distante (impossible de changer directement le nom avec pg restore):

```
sudo -u postgres createdb -T template0 geonature2db
```

 Donner les droits à l'utilisateur geonatadmin qui doit avoir les droits sur la base geonature2db:

```
sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE
geonature2db TO geonatadmin;"
```

- Si vous êtes connecté à Postgresql, par exemple via DBeaver, fermer vos connexions. Sinon, vous risquez d'avoir le message suivant : pg restore les emplacements de connexions restants sont réservés pour les connexions super-utilisateur non relatif à la réplication
- Si le fichier de sauvegarde récupéré est au format tar, il faut le désarchiver et donner les droits d'exécution:

```
cd /home/${USER}/tmp/ ; tar xvf
/home/${USER}/tmp/2021-07-27_geonature2db.tar ; chmod +x -R
/home/${USER}/tmp/2021-07-27 geonature2db/
```

• Restaurer la base à partir du dump distant :

```
sudo -u postgres pg_restore --host "localhost" --port "5432" -U
"geonatadmin" --jobs "8" --verbose --dbname "geonature2db"
"/home/\{USER\}/tmp/2021-07-27 geonature2db" 2>&1 | tee "./\{(date + ")^2\}
%m-%d")_pgrestore.log"
```

- Notes:
 - le dump est présent dans ~/tmp/2021-07-27 geonature2db. Changer

2025/11/19 11:54 9/13 Configure Postgresql

l'emplacement en fonction de votre machine.

- la commande précédente (à l'aide de tee) copie les éléments affichés à l'écran dans un fichier de log <date>_pgrestore.log.
- l'argument - format s'il n'est pas précisé est détecté automatiquement. Attention sinon à définir la bonne option: "c" (custom), "d" (directory), "t" (tar)
- pour l'option --jobs indiquer seulement 3/4 du nombre de processeurs (sinon une erreur "nombre max de client atteint" peut apparaître). Dans l'exemple, 8 pour 12 CPU. Pour afficher le nombre de processeur de votre machine : grep -c ^processor /proc/cpuinfo
- Retirer les droits de super-utilisateur à geonatadmin :

```
sudo -u postgres psql -c "ALTER ROLE geonatadmin NOSUPERUSER;"
```

• Renommer la base de données après avoir supprimé la base de données de destination. Exemple avec pour base de destination gn2_sinp_paca :

```
sudo -u postgres dropdb --if-exists gn2_sinp_paca
```

```
    sudo -u postgres psql -c "ALTER DATABASE geonature2db RENAME TO
gn2_sinp_paca;"
```

Mettre à jour Postgresql (Ex. v11 vers v15)

- Sur web s rv, activer la maintenance longue de GeoNature et l'Atlas.
- Se connecter sur db-srv en tant que "admin" puis passer en "root"
- Démarrer une session Screen : screen S upgrade-postgres l
- Réaliser le dump complet de la base : sudo -u postgres pg_dumpall > "/home/admin/backups/postgresql/2022-10-31 dumpall.dump"
 - $\circ\,$ Vérifier la taille du dump : du $\,$ -hs
 - /home/admin/backups/postgresql/2022-10-31 dumpall.dump
- Arrêter le service Postgresql: systemctl stop postgresql
 - Vérifier l'arrêt: systemetl status postgresql postgresql@11-main
- Installer la nouvelle version de Postgresql et Postgis: apt install -y postgresql-15postgis-3
- Si vous êtes sur Debian 11, le dépôt Postgresql par défaut est celui de la version 13. Il vous faudra installer PostgreSQL Apt Repository pour pouvoir installer les paquets debian de Postgresql 15.
- Copier le fichier de conf d'optimisation : cp /etc/postgresql/11/main/conf.d/01_optimizing.conf /etc/postgresql/15/main/conf.d/
- Réaliser le différentiel entre les 2 fichiers de conf et rechercher les paramètres non commentés différents : diff /etc/postgresql/11/main/postgresql.conf /etc/postgresql/15/main/postgresql.conf
 - Copier les paramètres suivant :

```
listen_addresses = 'localhost,10.0.1.20,172.18.5.1'
log_timezone = 'Europe/Paris'
datestyle = 'ISO, DMY'
timezone = 'Europe/Paris'
```

```
lc_messages = 'fr_FR.UTF-8'
lc monetary = 'fr FR.UTF-8'
lc numeric = 'fr FR.UTF-8'
lc time = 'fr FR.UTF-8'
default_text_search_config = 'pg_catalog.french'
```

• Réaliser le différentiel entre les 2 fichiers pg hba.conf : diff /etc/postgresql/11/main/pg hba.conf /etc/postgresql/15/main/pg hba.conf Copier les paramètres suivant :

```
host
        all
                        all
                                         10.0.1.20/32
scram-sha-256
# GeoNature
host
       geonature2db
                        geonatadmin
                                         10.0.1.10/32
scram-sha-256
                                         10.0.1.10/32
host
        gnatlas
                        geonatadmin
scram-sha-256
        gnatlas
                        geonatatlas
                                         10.0.1.10/32
host
scram-sha-256
# Si telegraf utilisé seulement avec un accès basique :
                         telegraf
                                         172.18.5.0/24
#host
         postgres
scram-sha-256
# Si telegraf utilisé seulement avec un accès avancés (toutes les
bases) :
host
       all
                        telegraf
                                        172.18.5.0/24
scram-sha-256
# GeoNature : access by gnreader (read only)
        geonature2db
                        gnreader
                                        10.0.1.20/32
host
scram-sha-256
        gnatlas
                                        10.0.1.20/32
host
                        gnreader
scram-sha-256
```

- Démarrer le service Postgresql : systemctl start postgresql
- Lancer la copie des données entre les 2 instances : sudo -u postgres pg_dumpall -p 5432 | sudo -u postgres psql -d postgres -p 5433
 - o Si la copie des données de l'Atlas échoue, vous pouvez passer à la suite puis régénérer la base de l'Atlas entièrement.
- Arrêter les 2 cluster et postgresql: systemctl stop postgresql postgresql@11-main postgresql@15-main
- Modifier les fichiers de conf des 2 version de Postgresql pour mettre le port 5432 à la plus récente :
 - o vi /etc/postgresql/11/main/postgresql.conf mettre port = 5433
 - vi /etc/postgresql/15/main/postgresql.conf mettre port = 5432
- Relancer les 2 versions de Postgresql: systemctl start postgresql postgresql@11main postgresql@15-main
 - Vérifier la bonne correspondance des ports et des versions : pg lsclusters

• ATTENTION :

- 1. pensez à mettre à jour le docker Borgmatic afin d'installer les outils clients de Postgresql dans la même version que celle de l'hôte.
- 2. la v14 a changer le type de cryptage par défaut des mots de passe. Voir ci-dessous pour

2025/11/19 11:54 11/13 Configure Postgresql

mettre à jour les mots de passes des rôles.

- Sur web-srv:
 - Redémarrer les services de GeoNature : sudo systematl restart geonature geonature-atlas taxhub usershub
 - o Vérifier le bon fonctionnement de GeoNature et de l'Atlas.
- Arrêter le service Postgresql correspondant à l'ancienne version : systemetl stop postgresql@11-main
- Vérifier le bon fonctionnement de GeoNature et de l'Atlas.
- Nettoyage :
 - Rapatrier en local sur un espace de stockage le dump de toutes les bases, puis le supprimer du serveur.
 - Désinstaller l'ancienne version de Postgresql après quelques temps : sudo apt remove
 --purge postgresql-11
 - Vérifier qu'il n'existe pas d'autres package ancien : apt list -a --installed postgresql* puis apt autoremove s'il faut supprimer des reliquats.
- Mettre à jour les références à la version de Postgresql :
 - ∘ sur web-srv et db-srv :
 - mettre à jour la version de Postgresql dans le fichier .env du Docker borgmatic du compte admin.
 - recompiler et relancer le container borgmatic : docker compose up -d -build borgmatic
 - sur *db-srv* seulement :
 - mettre à jour le fichier de config /opt/srvstatus/settings.ini : remplacer la version dans le nom du service postgresql.

Mise à jour des mots de passe des rôles

Le type de cryptage des mots de passe par défaut à changé (md5 → SCRAM-SHA-256) dans la version 14. Il est nécessaire de mettre à jour les mots de passe :

- Se connecter via le socket Unix avec un utilisateur superadmin (postresql ou votre compte) : psql
- Vérifier les mots de passe à mettre à jour :

```
SELECT rolname, rolpassword ~ '^SCRAM-SHA-256\$' AS has_upgraded
FROM pg_authid
WHERE rolcanlogin;
```

• Mettre les mots de passe à jour avec :

```
\password <login>
```

- Vérifier à nouveau que les mots de passe sont bien à jour avec la commande SQL précédente.
- Tester une connexion en vous connectant via :

```
psql -U <login> -h localhost
```

☐ Sauvegarder les bases de données

- Se connecter sur "db-srv" en tant qu'admin
- La mise en place de sauvegardes automatique des bases de données en local dans le dossier /home/admin/backups/postgresql/ avec copie distante sur Dropbox, passe par l'installation de scripts Bash et de Cron :
 - Créer le dossier ~/bin pour l'utilisateur admin mkdir ~/bin
 - Copier dedans le contenu des scripts Bash présents dans le dépôt Github sinp-pacasrv emplacement /db-srv/home/admin/bin/.
 - Installer en *root* dans le dossier /opt/bkp2dbx/ le contenu présents dans le dépôt Github sinp-paca-srv emplacement /shared/opt/bkp2dbx
 - Installer dans le dossier /etc/cron.d les cron présents dans le dépôt Github sinp-paca-srv emplacement /db-srv/etc/cron.d/. ATTENTION : bien retirer l'extension .cron. Les fichiers présents dans le dossier /etc/cron.d ne doivent pas contenir de point (.) ou de tiret (-) car sinon ils ne sont pas exécutés.
- Exemple de commande pour sauvegarder les bases GeoNature sans l'utilisation des scripts précédents :
 - S'il n'existe pas créer un dossier ~/backups/postgresql/ avec la commande : mkdir -p
 ~/backups/postgresql/
 - Création de la sauvegarde pour la base **geonature2db** :
 - Pour accélérer la sauvegarde et économiser de la place nous utiliserons le format de sauvegarde "directory" (paramètre - - format=d) qui permet une parallélisation sur plusieurs CPUs (paramètre - - jobs) et une compression (paramètre - compress 9):

```
export BACKUP_DIR="/home/${USER}/backups/postgresql"; \
export BACKUP_NAME="$(date +%F)_gonature2db"; \
export BACKUP_PATH="${BACKUP_DIR}/${BACKUP_NAME}"; \
mkdir -p "${BACKUP_DIR}"; \
pg_dump --file "${BACKUP_PATH}" --host "localhost" --port
"5432" --username "geonatadmin" --verbose --format=d --
jobs=$(grep -c ^processor /proc/cpuinfo) --compress 9
geonature2db; \
tar -cvf "${BACKUP_NAME}.tar" -C "${BACKUP_DIR}"
"${BACKUP_NAME}/"; \
cd "${BACKUP_DIR}"; \
rm -fR "${BACKUP_NAME}/";
find "${BACKUP_DIR}" -name "*_gonature2db.tar" -type f -mtime
+5 -exec rm -f {} \;
```

- Création de la sauvegarde pour la base gnatlas :
 - Elle ne contient que des tables étrangères et des vues matérialisées, nous l'exportons donc en SQL (paramètre - format=p) :

```
export BACKUP_DIR="/home/${USER}/backups/postgresql"; \
export BACKUP_NAME="$(date +%F)_gnatlas"; \
export BACKUP_PATH="${BACKUP_DIR}/${BACKUP_NAME}.sql"; \
pg_dump --file "${BACKUP_PATH}" --host "localhost" --port
"5432" --username "geonatatlas" --verbose --format=p gnatlas;
```

2025/11/19 11:54 13/13 Configure Postgresql

```
tar -cvf "${BACKUP_NAME}.tar" -C "${BACKUP_DIR}"
"${BACKUP_NAME}.sql" ; \
rm -f "${BACKUP_PATH}" ; \
find "${BACKUP_DIR}" -name "*_gnatlas.tar" -type f -mtime +365
-exec rm -f {} \;
```

 Pour envoyer ces sauvegardes sur Dropbox utilisé le script : /opt/bkp2dbx/bkp2dbx.sh "postgresql" "/home/admin/backups/postgresql/*.tar" "/home/admin/.dropbox_uploader"

□ Automatisation de la sauvegarde

- Se connecter sur "db-srv" en tant qu'admin
- Créer le dossier ~/bin/: mkdir ~/bin
- Copier dans ce dossier le contenu du dossier db-srv/home/admin/bin sur le serveur : scp r bin/* admin@db-<region>-sinp:~/bin/
- Créer un fichier ~/.pgpass (voir doc)qui contiendra les mots de passe pour accéder aux base de données : vi ~/.pgpass
 - Y stocker ceci:

```
# Format: hostname:port:database:username:password
localhost:5432:geonature2db:geonatadmin:<mot-de-passe>
localhost:5432:gnatlas:geonatatlas:<mot-de-passe>
```

- Donner les bons droits : chmod 600 ~/.pgpass
- S'il n'existe pas créer un dossier ~/backups/postgresql/ avec la commande : mkdir -p ~/backups/postgresql/
- Copier dans le dossier /etc/cron.d/ le fichier db-srv/etc/cron.d/backups sur le serveur : scp r backups admin@db-<region>-sinp:~/dwl/ puis sur le serveur : sudo -i ; mv /home/admin/dwl/backups /etc/cron.d/
- L'administrateur système doit recevoir un email après chaque exécution des scripts de sauvegarde.

From

https://wiki-sinp.cbn-alpin.fr/ - CBNA SINP

Permanent link:

https://wiki-sinp.cbn-alpin.fr/serveurs/installation/db-srv/postgresql-config?rev=1704991075

Last update: 2024/01/11 16:37

