Import des données via GN2PG

Notes

- C'est le serveur *db-srv* qui se charge d'héberger le script Gn2Pg dans le dossier `/home/geonat/data/gn2pg`
  - Le serveur *bkp-srv* peut également héberger Gn2Pg pour l'espace de préprod.
- Un script Bash '`/home/geonat/data/gn2pg/bin/gn2pg_update.sh` encapsule l'appel de Gn2Pg
- Le script Bash a une dépendance à *Pipenv* qui se charge de télécharger et installer le paquet Python *gn2pg-client* pour nous
- Il est nécessaire d'installer *Pipenv* sur le système et de préparer le dossier `/home/geonat/data/gn2pg/gn2pg`, pour cela voir le fichier /home/geonat/data/gn2pg/README.md
- Un cron (`/etc/crond.d/gn2pg`) se charge de lancer Gn2Pg aux heures déterminées pour chaque config définie.
- Il est possible de vérifier si Gn2Pg est en cours de fonctionnement en filtrant les processus sur le mot-clé "gn2pg_update" : `ps -aux|grep gn2pg_update`
- Les fichiers de log de Gn2Pg sont consultables ici : `/home/geonat/data/gn2pg/log/`

Relancer un import GN2PG interrompu

- Connectez vous à la base de donnée
- Afficher la table : `gn2pg_lpo.data_json`
- Trier les données par ordre décroissant du champ `update_ts`. La date la plus récente devrait s'afficher en premier
  - La date `update_ts` et la valeur du champ `id_data` du dernier enregistrement devrait correspondre au dernier log de Gn2Pg : `2024-11-18 15:37:53,851 - DEBUG - store_postgresql:store_1_data - 43682513`
- Le champ "id_data" contient l'id_synthese à copier.
- Modifier ensuite le fichier `~/data/gn2pg/config/lpo_config.toml` sur le serveur hébergeant *Gn2Pg*.
  - Ajouter/Modifier le paramètre `filter_n_up_id_synthese` et lui associer l'id synthese précédemment copié comme valeur.
  - **ATTENTION** : penser une fois le téléchargement via Gn2PG terminé à commenter cette valeur.

Principes pour les données transmises

- Nous gardons l'ID synthèse du GeoNature de chaque fournisseur de données (pôle en AURA) en guise de `entity_source_pk_value`.
- Nous gardons la contrainte d'unicité entre `id_source` et `entité_source_pk_value`.
  - Avec la fourniture du « vrai » identifiant produit par le producteur au niveau du champs additionnel sous la clé « `source_id_data` ».
- Jusqu'à la version v1.6.9, pas d'utilisation de l'UUID car effets de bords qui s'avérent

problématiques selon les usages. Utilisation de l'UUID à partir de la version 1.8.0.
- Les champs `identifier` et `email` des données sur les utilisateurs devront avoir la valeur `NULL`. C'est important pour éviter tout conflit avec les utilisateurs créant des comptes directement depuis les interfaces. Pour éviter aussi des bugs au niveau de l'inscription et du renouvellement des mots de passe, il y a un index unique sur ces 2 champs. Enfin, même si cela crée des doublons, nous distinguerons les entrées dans la table `t_roles` pour les utilisateurs s'inscrivant au SINP vis à vis des entrées générées par GN2PG ou les scripts d'intégrations des données au format CSV.
    - **La fonction se chargeant d'insérer les utilisateurs dans la table `t_roles` de GeoNature a été modifiée pour insérer `NULL` dans le champ `email` et l'email dans le champ `additional_data` sous l'attribut `gn2pg_data.email`**.
- Le champ `meta_validation_date` de la synthèse du fournisseur utilisera l'alias `validation_date` soit : `s.meta_validation_date AS validation_date`
- Le champ `additional_data` de la synthèse du fournisseur utilisera l'alias `donnees_additionnelles` soit : `s.additional_data::text AS donnees_additionnelles`
- Les données à intégrer au champs additionnel en tant qu'attributs de l'objet principal sont :
    - « `source_nom` »
    - « `source_id_data` »

Mise à jour de l'installation de Gn2Pg

- Sur le dépôt Github *sinp-<region>-data* :
    - modifier la version du client *gn2pg-client* dans le fichier `gn2pg/Pipfile`.
        - Installer la nouvelle version du client avec : `pipenv install`
        - Si besoin, mettre à jour le fichier Pipfile.lock avec la commande : `pipenv lock`
        - Créer un nouveau commit avec ces modifications
    - corriger les fichier `gn2pg/data/sql/…to_synthese.sql` en :
        - comparant les modifications effectuées sur le fichier to_gnsynthese.sql du dépôt GN2PG depuis la précédente version utilisée.
    - Mettre à jour le fichier `README.md` si nécessaire
    - Corriger les fichiers `config/…_config.toml` si les paramètres de config ont évolués (voir check_conf.py, Changelog et realeases)
        - En local, les tester avec : `pipenv run gn2pg_cli --custom-script ./data/sql/<...>_to_synthese.sql <...>_config.toml`
    - Tester le script en local depuis le dossier `gn2pg/bin/` avec : `./gn2pg_update.sh -v -c <…>_config.toml`
- Sur le serveur *db-srv* ou *bkp-srv* (pré-production) :
    - Se placer dans le dossier ~/data/ avec : '' cd ~/data/''
    - Récupérer les changements : `git pull ; git submodule update`
    - Mettre à jour Pipenv :
        - afficher la version actuellement installée de Pipenv via Pipx : `pipx list`
        - mise à jour : `pipx upgrade pipenv`
        - vérifier la version de Pipenv : `pipenv --version`
    - Mettre à jour les dépendances des paquets Python : `pipenv sync`
    - Se placer dans le dossier de gn2pg : '' cd ~/data/gn2pg/''
    - Vérifier l'installation de Gn2Pg :
        - s'assurer que `~/.gn2pg/` est bien un lien vers `~/data/gn2pg/config/` avec : `ls -al ~/.gn2pg`
            - si nécessaire, supprimer le dossier et recréer le lien : `rm -fr ~/.gn2pg/;`

```
ln -s ~/data/gn2pg/config ~/.gn2pg
```

        - vérifier la version de Gn2pg : `pipenv run gn2pg_cli --version`
    - Vérifier/Ajuster les paramètres de config : `vi <…>_config.toml`
    - Si nécessaire, réinstaller les fichiers `…_to_synthese.sql` avec : `pipenv run gn2pg_cli --custom-script ./data/sql/<...>_to_synthese.sql <...>_config.toml`

Installer le Dashboard Gn2Pg

- Ressources : https://github.com/lpoaura/GN2PG/blob/main/docs/dashboard.rst
- Ex. : https://gn2pg.biodiversite-aura.net/
- Se connecter sur le serveur *web-srv* en tant que *geonat* avec : `ssh geonat@web-<region>-sinp`
- Créer les dossiers qui contiendront l'installation de Gn2Pg : `mkdir -p ~/www/gn2pg/public/`
- Copier dedans les fichiers suivant :

```
cp ~/www/geonature/frontend/src/favicon.ico ~/www/gn2pg/public/
cp ~/www/geonature/frontend/src/custom/favicon.ico ~/www/gn2pg/public/
cp ~/www/geonature/frontend/src/custom/images/logo-orb.png ~/www/gn2pg/public/
```

- Créer une page `index.html` qui contiendra les liens vers les Dashboards de Gn2Pg : `vi ~/www/gn2pg/public/index.html`
- Pour chaque schéma abritant les tables de Gn2Pg (ou *source*), il faut cloner le dépôt Gn2Pg. Ex. pour le SINP AURA (nous avons 2 *sources* : flavia, lpo) :
    - Se placer dans le dossier `gn2pg/` avec : `cd ~/www/gn2pg/`
    - Cloner une première fois le dépôt en tant que *lpo* avec : `git clone https://github.com/lpoaura/GN2PG.git lpo`
    - Cloner une seconde fois le dépôt en tant que *flavia* avec : `git clone https://github.com/lpoaura/GN2PG.git flavia`
- Pour chaque source, nous allons réaliser manuellement les étapes présentes dans `install/02_install_app.sh` (pour les adapter à notre infrastructure) :
    - se placer dans le dossier de la source : `cd ~/www/gn2pg/<source>/`
    - Copier le fichier settings.ini : `cp install/settings.ini.sample install/settings.ini`
    - Éditer et modifier le fichier settings.ini : `vi install/settings.ini`

```
GN2PG_CONFIG_NAME="<source>_config.toml"
APPLICATION_ROOT="/<source>"
SERVER_NAME="gn2pg.<domaine-sinp>"
GUNICORN_WORKERS=4
GUNICORN_TIMEOUT=30
GUNICORN_PORT=5050<num-increment-source>
DEBUG=true
```

        - Pour le numéro du port Gunicorn, partir de 50500 et ajouter 1 pour chaque source (50501, 50502…).
    - Copier le fichier de config gn2pg depuis le serveur *db-srv* : `scp -P <port-ssh-db-srv> geonat@db-<region>-`

`sinp:~/data/gn2pg/config/<source>_config.toml install/<source>_config.toml`
    - Modifier le fichier de config : `vi install/<source>_config.toml`
        - Vous pouvez garder seulement la section `[db]` et commenter les autres sections
- Créer un fichier `environ` avec : `vi environ`

```
GUNICORN_PROC_NAME=gn2pg-<source>
GUNICORN_PORT=5050<num-increment-source>
GUNICORN_LOG_FILE=/var/log/gn2pg/gn2pg-gunicorn-<source>.log
GUNICORN_LOG_LEVEL=debug
```

- Installer Poetry : `pipx install poetry`
- Installer les dépendances du Dashboard de Gn2pg avec :

```
poetry config virtualenvs.create true --local
poetry config virtualenvs.in-project true --local
poetry install --extras=dashboard
```

- Créer un fichier de service Sytemd : `vi /etc/systemd/system/gb2pg-<source>.service` (voir ci-dessous)
    - Prendre en compte le service créé : `systemctl daemon-reload`
    - Lancer le service : `systemctl start gn2pg-<source>.service`
- Si nécessaire, créer un fichier Logrotate (uniquement pour la première source) : `vi /etc/logrotate.d/gn2pg` (voir ci-dessous)
- Si nécessaire, créer un fichier de config Nginx : '' vi /etc/nginx/site-available/gn2pg.conf '' (voir ci-dessous)
    - Lors de la création du fichier :
        - Activer le nouveau fichier de config : `nginx_ensite gn2pg.conf`
        - Recharger Nginx : `nginx-reload`
        - Récupérer le certificat SSL : '' certbot -d gn2pg.<domaine-sinp>''
    - Pour les sources suivantes, il faut seulement compléter le fichier config en ajoutant une nouvelle "location" et son proxy.
        - Recharger Nginx : `nginx-reload`
- Tester l'accès au site en https
    - Consulter les logs en cas de problème : `ls /var/log/gn2pg/`

Fichier Nginx par défaut

```
server {
    listen [::]:80 ipv6only=on;
    listen 80;

    server_name gn2pg.<domaine-sinp>;
    root /home/geonat/www/gn2pg/public;

    satisfy any;
    allow <ip-v4-srv-web>;
    deny  all;
    auth_basic "Zone restreinte";
```

```
    auth_basic_user_file /etc/nginx/.htpasswd;

    location /<source> {
        proxy_set_header X-Forwarded-Host $host:$server_port;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_redirect off;
        proxy_buffering off;

        proxy_read_timeout 30s;
        proxy_connect_timeout 10s;
        proxy_pass http://127.0.0.1:<source-port>;# ATTENTION : ne rien
mettre après le port pour que le chemin complet soit passé à Gunicorn
    }
}
```

Fichier Logrotate de Gn2Pg

```
/var/log/gn2pg/*.log {
    su geonat geonat
    daily
    rotate 8
    size 100M
    create
    compress
    postrotate
    systemctl reload gn2pg || true
    endscript
}
```

Fichier Systemd de Gn2Pg

```
[Unit]
Description=GN2PG-<source>
After=network.target
#After=postgresql.service

[Service]
Type=simple
User=geonat
Group=geonat
WorkingDirectory=/home/geonat/www/gn2pg/<source>/
Environment=GUNICORN_PROC_NAME=gn2pg
Environment=GUNICORN_NUM_WORKERS=4
Environment=GUNICORN_PORT=5001
Environment=GUNICORN_TIMEOUT=30
Environment=GUNICORN_LOG_FILE=/var/log/gn2pg/gn2pg-gunicorn.log
```

```
Environment=GUNICORN_LOG_LEVEL=info
EnvironmentFile=-/home/geonat/www/gn2pg/<source>/environ
ExecStart=/home/geonat/www/gn2pg/<source>/.venv/bin/gunicorn
gn2pg.app.app:create_app() \
                --name "${GUNICORN_PROC_NAME}" --workers
"${GUNICORN_NUM_WORKERS}" \
                --bind "${GUNICORN_HOST}:${GUNICORN_PORT}" --
timeout="${GUNICORN_TIMEOUT}" \
                --log-file "${GUNICORN_LOG_FILE}" --log-level
"${GUNICORN_LOG_LEVEL}"
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutStartSec=10
TimeoutStopSec=5
PrivateTmp=true
StandardOutput=append:/var/log/gn2pg/gn2pg-<source>.log
StandardError=inherit

[Install]
WantedBy=multi-user.target
```

Exemple de contenu pour index.html

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Gn2Pg Dashboards Biodiv'AURA</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.cs
s" rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    <style>
      .bi {
        vertical-align: -.125em;
        fill: currentColor;
      }
    </style>
  </head>
  <body>
    <svg xmlns="http://www.w3.org/2000/svg" class="d-none">
      <symbol id="arrow-right-short" viewBox="0 0 16 16">
        <path fill-rule="evenodd" d="M4 8a.5.5 0 0 1 .5-.5h5.793L8.146
5.354a.5.5 0 1 1 .708-.708l3 3a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708-
.708L10.293 8.5H4.5A.5.5 0 0 1 4 8z"/>
      </symbol>
    </svg>
    <div class="container my-5">
      <div class="p-5 text-center bg-body-tertiary rounded-3">
```

```
        <img src="logo-orb.png" alt="ORB AURA"/>
        <img src="/lpo/img/src_gn2pg.png" />
        <h1 class="text-body-emphasis">Gn2Pg dashboards</h1>
        <p class="col-lg-8 mx-auto fs-5 text-muted">
          Accès aux interfaces des dashboards de Gn2Pg.
        </p>
        <div class="d-inline-flex gap-2 mb-5">
          <a class="d-inline-flex align-items-center btn btn-primary btn-lg
px-4 rounded-pill" href="/lpo">
            LPO
            <svg class="bi ms-2" width="24" height="24"><use
xlink:href="#arrow-right-short"/></svg>
          </a>
          <a class="d-inline-flex align-items-center btn btn-primary btn-lg
px-4 rounded-pill" href="/flavia">
            Flavia
            <svg class="bi ms-2" width="24" height="24"><use
xlink:href="#arrow-right-short"/></svg>
          </a>
        </div>
      </div>
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.m
in.js" integrity="sha384-
C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
crossorigin="anonymous"></script>
  </body>
</html>
```

Requêtes SQL utiles pour Gn2Pg

```
-- Nbre de données récupérées depuis une date donnée
SELECT COUNT(*)
FROM gn2pg_lpo.data_json
WHERE update_ts > '2025-06-30';

-- Nbre d'erreurs depuis une date donnée
SELECT COUNT(*)
FROM gn2pg_lpo.error_log
WHERE last_ts > '2025-06-22';

-- Principaux types d'erreur différents
SELECT DISTINCT  split_part(error, e'\n', 1)
FROM gn2pg_lpo.error_log;
```

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

Permanent link:
https://wiki-sinp.cbn-alpin.fr/database/sinp-aura/gn2pg?rev=1772704140

Last update: **2026/03/05 09:49**