

Installation d'un espace de Pré-Production pour GeoNature

Dépendances nécessaires

- Se connecter à l'utilisateur geonat de bkp-srv :

```
ssh geonat@bkp-<region>-sinp
```

- Installer les dépendances (NVM, Python...) de GeoNature en suivant [la documentation](#).

Procédure rapatriement de la prod

- Se connecter à l'utilisateur geonat de bkp-srv :

```
ssh geonat@bkp-<region>-sinp
```

- Récupérer l'installation des outils GeoNature de web-srv (supprimer l'option --dry-run si tout est OK) :

```
rsync -e "ssh -p <port-ssh-web-srv>" -av --exclude .cache/ --exclude venv/ --exclude venv.save/ geonat@web-<region>-sinp:/home/geonat/www/ /home/geonat/www/ --dry-run
```

- Récupérer les données de db-srv (supprimer l'option --dry-run si tout est OK) :

```
rsync -av -e "ssh -p <port-ssh-db-srv>" --exclude "raw/*" geonat@db-<region>-sinp:/home/geonat/data/ /home/geonat/data --dry-run
```

- Recréer les liens symboliques de la racine du home de geonat :

```
ln -s www/atlas atlas
ln -s www/geonature geonature
ln -s www/taxhub taxhub
ln -s www/usershub usershub
```

Procédure : installation via Git

- Cloner le dépôt avec l'URL en https : `git clone https://github.com/PnX-SI/GeoNature.git geonature`
- Se placer dans le dossier du dépôt cloné : `cd geonature`
- Récupérer le fichier config/settings.ini avec : `scp geonat@web-<region>-sinp:~/geonature/config/settings.ini ./config/`
 - Adapter son contenu : `vi config/settings.ini`
- Récupérer le fichier config/geonature_config.toml avec : `scp geonat@web-`

- <region>-sinp:~/geonature/config/geonature_config.toml ./config/
 - Adapter son contenu : vi config/geonature_config.toml
- Réinstaller le venv du backend :
 - cd install/
 - ./01_install_backend.sh
 - Arrêter le script lors de la demande de mot de passe pour l'installation des scripts SystemD : CTRL+C
 - Installer le module Sentry :
 - source ../backend/venv/bin/activate
 - pip install sentry_sdk
 - deactivate
- Copier les composants "custom" présents dans frontend/src/custom/components/
 - scp -r geonat@web-<region>-sinp:~/geonature/frontend/src/custom/components/frontend/src/custom/
 - Si besoin restaurer les fichiers .sample : git restore *.sample
- Copier les images "custom" présentes dans frontend/src/custom/images/
 - scp -r geonat@web-<region>-sinp:~/geonature/frontend/src/custom/images frontend/src/custom/
- Copier le fichier *favicon.ico* :
 - scp -r geonat@web-<region>-sinp:~/geonature/frontend/src/favicon.ico frontend/src/
- Penser à reporter les éventuelles modifications du fichier frontend/src/assets/i18n/fr.json :
 - scp geonat@web-<region>-sinp:~/geonature/frontend/src/assets/i18n/fr.json ./frontend/src/assets/i18n/
- Récupérer le fichier *environ* :
 - scp geonat@web-<region>-sinp:~/geonature/envIRON ./
 - Adapter le fichier *environ* aux caractéristiques du serveur de la pré-prod : vi ./envIRON
- Recréer les liens symboliques présents dans frontend/src/external_assets/ et dans external_modules/
- Réinstaller les modules pre-existant :

```
source backend/venv/bin/activate
pip list | grep modules
pip install -e /home/geonat/www/modules/<module>
pip list | grep modules
```

- Essayer de réinstaller le frontend à l'aide du script :
 - cd install/
 - ./04_install_frontend.sh
- Sinon, une alternative et la réinstallation manuelle du frontend :
 - Réinstaller Npm et Node à l'aide de Nvm :
 - cd frontend/
 - nvm use
 - Réinstaller les paquets Npm :
 - cd frontend/
 - npm install
 - Reconstruire le frontend à l'aide d'Angular :

- cd frontend/
- npm run build
- Corriger/Créer un fichier Logrotate :
 - vi /etc/logrotate.d/geonature
- Corriger/Créer le service SystemD

Reconfigurer GeoNature

- [Installer/Mettre à jour les dépendances de GeoNature](#).
- **NOTES** : il est nécessaire que Gunicorn écoute sur l'IP 0.0.0.0 car Nginx étant dans un container il ne peut accéder à l'IP 127.0.0.1. Le parefeu se charge de rendre inaccessibles le port 8000 de GeoNature ainsi que ceux d'UsersHub et TaxHub.
- Reconfigurer GeoNature :
 - Changer les paramètres suivants dans *geonature_config.toml* :

```
vi ~/geonature/config/geonature_config.toml
```

```
SQLALCHEMY_DATABASE_URI = "postgresql://geonatadmin:<mot-de-passe-geonatadmin-sur-bkp-srv>@127.0.0.1:5432/geonature2db"
URL_APPLICATION = "https://gnpp.silene.eu"
API_ENDPOINT = "https://gnpp.silene.eu/api"
API_TAXHUB = "https://thpp.silene.eu/api"

# Remplacer par une clé aléatoire complexe
SECRET_KEY = "<utiliser-uuid-pour-générer-une-clé>"

appName = "<nom-du-site> - Pré-Prod"

# Set Sentry DSN
SENTRY_DSN = "<nouveau-sentry-DSN-généré-dans-app-monitor>"

[USERSHUB]
URL_USERSHUB = "https://uhpp.<sinp-domaine>"

[ACCOUNT_MANAGEMENT]
VALIDATOR_EMAIL = [
    "admins@<sinp-domaine>",
]

[FRONTEND]
# Durée de vie du cache des stats de la page d'accueil
# 86 400 = 1 jour
STAT_BLOC_TTL = 86400

[PERMISSION_MANAGEMENT]
VALIDATOR_EMAIL = [
    "admins@<sinp-domaine>",
]
```

- Changer les paramètres suivants dans *settings.ini* :

```
vi ~/geonature/config/settings.ini
```

```
#MODE=dev #Voir s'il faut utiliser git pour l'installation et donc le mode "dev"...  
my_url=https://gnpp.silene.eu/  
db_host=127.0.0.1  
user_pg_pass=<mot-de-passe-geonatadmin-sur-bkp-srv>
```

- Réinstaller le venv et Node pour GeoNature :

```
cd ~/geonature/frontend  
nvm install  
cd ~/geonature/backend  
rm -fR venv  
cd ~/geonature/install  
./01_install_backend.sh
```

- Réinstaller le frontend :

```
cd ~/geonature/install  
./04_install_frontend.sh
```

- Si erreur An unhandled exception occurred: Cannot find module '@angular-builders/custom-webpack/package.json :

```
cd ~/geonature/frontend/  
nvm use  
npm install  
npm run build
```

- Créer un dossier pour les logs :

```
mkdir /var/log/geonature/  
mv /var/log/usershub.log /var/log/geonature/  
chown -R geonat: /var/log/geonature/
```

- Créer un fichier logrotate :

```
vi /etc/logrotate.d/geonature
```

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

```
}
```

- Surcharger le service SystemD :

```
systemctl edit geonature
```

```
[Unit]
Description=GeoNature - PreProd
After=docker.service
StartLimitIntervalSec=60min
StartLimitBurst=25

[Service]
Restart=on-failure
RestartSec=2min
StandardOutput=append:/var/log/geonature/geonature.log
```

- Créer un fichier environ :

```
vi ~/geonature/environ
```

```
GUNICORN_HOST=0.0.0.0
GUNICORN_NUM_WORKERS=9
GUNICORN_TIMEOUT=300
```

Reconfigurer l'Atlas

- Changer les paramètres suivants dans *config.py* :

```
vi ~/atlas/atlas/configuration/config.py
```

```
database_connection = "postgresql://geonatlas:<mot-de-passe-
utilisateur-geonatlas>@127.0.0.1:5432/gnatlas"
NOM_APPLICATION = "Silene Nature - Pré-Prod"
SECRET_KEY = '<utiliser-uuid-pour-générer-une-clé>'
```

- Changer les paramètres suivants dans *settings.ini* :

```
vi ~/www/atlas/atlas/configuration/settings.ini
```

```
db_host=0.0.0.0
user_pg_pass=<mot-de-passe-utilisateur-geonatlas>
owner_atlas_pass=<mot-de-passe-utilisateur-geonatadmin>

db_source_host=127.0.0.1
atlas_source_pass=<mot-de-passe-utilisateur-geonatadmin>
```

- Réinstaller le *venv* et *Node* pour *GeoNature-Atlas* :

```
cd ~/atlas/atlas/static/
```

```
nvm install
cd ~/atlas/
rm -fR venv
./install_app.sh
```

- Créer un dossier pour les logs :

```
mkdir /var/log/geonature-atlas/
mv /var/log/usershub.log /var/log/geonature-atlas/
chown -R geonat: /var/log/geonature-atlas/
```

- Créer un fichier logrotate :

```
vi /etc/logrotate.d/geonature-atlas

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

- Surcharger le service SystemD :

```
systemctl edit geonature-atlas
```

```
[Unit]
Description=Atlas - PreProd
After=docker.service
StartLimitIntervalSec=60min
StartLimitBurst=25

[Service]
Restart=on-failure
RestartSec=2min
StandardOutput=append:/var/log/geonature-atlas/geonature-atlas.log
```

- Créer un fichier environ :

```
vi ~/atlas/environ

GUNICORN_HOST=0.0.0.0
GUNICORN_NUM_WORKERS=4
GUNICORN_TIMEOUT=90
```

Reconfigurer UsersHub

- Changer les paramètres suivants dans *config.py* :

```
vi ~/usershub/config/config.py
```

```
SQLALCHEMY_DATABASE_URI = "postgresql://geonatadmin:<mot-de-passe-geonatadmin-sur-bkp-srv>@127.0.0.1:5432/geonature2db"  
URL_APPLICATION = 'https://uhpp.silene.eu'  
SECRET_KEY = '<utiliser-uuid-pour-générer-une-clé>'
```

- Changer les paramètres suivants dans *settings.ini* :

```
vi ~/usershub/config/settings.ini
```

```
db_host=0.0.0.0  
user_pg_pass=<mot-de-passe-utilisateur-geonatadmin>  
url_application=https://uhpp.silene.eu
```

- Réinstaller le *venv* et *Node* pour *UsersHub* :

```
cd ~/usershub/app/static/  
nvm install  
cd ~/usershub/  
rm -fR venv  
./install_app.sh
```

- Créer un dossier pour les logs :

```
mkdir /var/log/usershub/  
mv /var/log/usershub.log /var/log/usershub/  
chown -R geonat: /var/log/usershub/
```

- Créer un fichier logrotate :

```
vi /etc/logrotate.d/usershub
```

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

- Surcharger le service SystemD :

```
systemctl edit usershub.service
```

```
[Unit]
Description=UsersHub - PreProd
After=docker.service
StartLimitIntervalSec=60min
StartLimitBurst=25

[Service]
Restart=on-failure
RestartSec=2min
StandardOutput=append:/var/log/usershub/usershub.log
```

- Créer un fichier environ :

```
vi ~/usershub/envIRON
```

```
GUNICORN_HOST=0.0.0.0
```

Reconfigurer TaxHub

- Changer les paramètres suivants dans *config.py* :

```
vi ~/taxhub/apptax/config.py
```

```
DEBUG=True
SQLALCHEMY_DATABASE_URI = "postgresql://geonatadmin:<mot-de-passe-geonatadmin-sur-bkp-srv>@127.0.0.1:5432/geonature2db"
SECRET_KEY = '<utiliser-uuid-pour-générer-une-clé>'
```

- Changer les paramètres suivants dans *settings.ini* :

```
vi ~/taxhub/settings.ini
```

```
db_host=0.0.0.0
user_pg_pass=<mot-de-passe-utilisateur-geonatadmin>
```

- Réinstaller le *venv* et *Node* pour *TaxHub* :

```
cd ~/taxhub/static/
nvm install
cd ~/taxhub/
rm -fR venv
./install_app.sh
```

- Créer un dossier pour les logs :

```
mkdir /var/log/taxhub/
mv /var/log/usershub.log /var/log/taxhub/
```

```
chown -R geonat: /var/log/taxhub/
```

- Créer un fichier logrotate :

```
vi /etc/logrotate.d/taxhub
```

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

- Surcharger le service SystemD :

```
systemctl edit taxhub
```

```
[Unit]
Description=TaxHub - PreProd
After=docker.service
StartLimitIntervalSec=60min
StartLimitBurst=25

[Service]
Restart=on-failure
RestartSec=2min
StandardOutput=append:/var/log/taxhub/taxhub.log
```

- Créer un fichier environ :

```
vi ~/taxhub/envIRON
```

```
GUNICORN_HOST=0.0.0.0
```

Création du container Docker hébergeant Postgresql & Nginx

Activation de l'IPv6 sur Docker et la stack proxy

- Afin d'avoir un support complet d'IPv6 sur la pré-prod, il est nécessaire [d'activer le support d'IPv6 par Docker](#).

Création de la stack preprod

- Ajouter l'utilisateur *geonat* au groupe docker :

```
sudo usermod -aG docker geonat
```

- Se déconnecter et se reconnecter au compte *geonat* et vérifier la prise en compte du groupe avec :

```
id
```

- Tester Docker en tant qu'utilisateur *geonat* avec :

```
docker run hello-world
```

- Si le message d'erreur docker: Got permission denied while trying to connect to the Docker daemon socket apparait, redémarrer la machine pour activer la prise en compte de Docker.
- Créer le dossier qui hébergera la stack GeoNature Docker Compose :

```
mkdir -p ~/docker/preprod
```

- Se placer dedans et récupérer les fichiers disponible sur [le dépôt Github sinp-paca-srv](#) ou sur [le dépôt Github sinp-aura-srv](#).
- Copier le fichier `.env.sample` en `.env` :

```
cp .env.sample .env
```

- Configurer le fichier `.env`
- Lancer la stack avec :

```
docker compose up -d
```

- Vérifier le bon fonctionnement avec *Portainer*.

Accéder au shell PsqI depuis le container Postgres

- Assurer vous que le container *preprod-postgres* est bien dans l'état *running* (via Portainer ou en ligne de commande).
- Accéder à l'intérieur du container :

```
docker exec -it preprod-postgres bash
```

- Accéder à *PsqI* avec :

```
psql -U ${POSTGRES_USER} ${POSTGRES_DB}
```

- Dans le container, l'utilisateur `${POSTGRES_USER}` et la base `${POSTGRES_DB}` remplace l'utilisateur par défaut *postgres*. Il est donc nécessaire de se connecter avec ces arguments !

Installer le client Postgresql sur l'hôte

- Vérifier la présence du dépôt de paquets deb pour Postgresql dans :
/etc/apt/sources.list.d/
- **Vérifier la version de Postgresql utilisé par le container Docker Postgresql de la pré-prod.**
 - Utiliser la même version pour l'installation du paquet postgresql-client. Ex. ici la v15 :
apt install postgresql-client-15
- Afin de pouvoir utiliser la commande psql sur l'hôte mais également le script install_db.sh de l'Atlas, il est nécessaire de :
 - Créer un utilisateur postgres sur l'hôte :

```
adduser --system --no-create-home postgres
```

- S'assurer que le fichier docker-compose.yml de la préprod lançant la base Postgres créé bien un mapping des sockets sur /run/postgresql dans les volumes (- /run/postgresql/:/var/run/postgresql).
- Il semble aussi nécessaire de modifier le mapping utilisateur des Foreign Data Tables en indiquant que le mot de passe n'est pas requis :

```
sudo -u postgres -s psql -d $db_name -c "CREATE USER MAPPING FOR $owner_atlas SERVER geonaturedbserver OPTIONS (user '$atlas_source_user', password_required 'false');" &>> log/install_db.log
```

- Le problème c'est que ce mécanisme ne marche pas en PROD où le mot de passe est bien requis ! Il faudrait chercher l'origine de la différence de fonctionnement...
 - En attendant, le plus simple semble de corriger le script install_db.sh sur la Préprod si besoin.

Stocker les logs Postgresql sur l'hôte

- Créer un dossier /var/log/postgresql:

```
mkdir /var/log/postgresql/  
chown root:999 /var/log/postgresql/  
chmod 774 /var/log/postgresql/
```

- Sur l'hôte l'utilisateur du container créant le socket correspond à l'utilisateur dont l'id vaut 999 (systemd-coredump | systemd-timesync). ☐ Il faudrait voir à utiliser l'utilisateur système "postgres"...
- Assurez vous que le fichier docker-compose.yml de la préprod lançant la base Postgres créé bien un mapping du dossier suivant- /var/log/postgresql/:/var/log/postgresql
- Vérifier également que Postgresql dans le container est bien lancé avec les paramètres de config suivant :

```
# Log  
#log_destination = stderr  
log_directory = '/var/log/postgresql'  
log_filename = 'postgresql-%a.log'  
log_file_mode = 0600  
log_truncate_on_rotation = on  
log_rotation_age = 1440
```

```
log_rotation_size = 0
```

- TODO : Voir comment forcer l'id de l'utilisateur "postgres" créant les fichiers de log : dans le container. Par défaut il s'agit de "postgres" avec un id 999, il semblerait...

Remplacer la base par la dernière version sauvegardée

- Se connecter en *root* sur *bkp-srv* :

```
ssh admin@bkp-<region>-sinp ; sudo -i
```

- Monter le dépôt Borg de *db-srv* :

```
borg mount /home/backups/db-srv /tmp/repo
```

- Saisir la *passphrase* de *db-srv*.
- Extraire le dump de la base de GeoNature et le copier dans l'espace de restauration du container *preprod-postgres-restore* avec :

- Pour *geonature2db* :

```
cp -r /tmp/repo/db-srv-<date-et-heure>/root/.borgmatic/postgresql_databases/172.18.5.1/geonature2db /home/geonat/docker/preprod/postgres/restore/<date>_geonature2db.custom
```

- Pour *gnatlas* :

```
cp -r /tmp/repo/db-srv-<date-et-heure>/root/.borgmatic/postgresql_databases/172.18.5.1/gnatlas /home/geonat/docker/preprod/postgres/restore/<date>_gnatlas.custom
```

- **ATTENTION** : le nom du fichier du dump récupéré doit contenir le nom de la base de donnée *geonature2db* ou *gnatlas*. Cette information est récupérée par le script de restauration pour effectuer certaines tâches spécifiques.
- Donner les droits à *geonat* sur les dumps :

```
chown geonat: /home/geonat/docker/preprod/postgres/restore/*.custom
```

- Démonter le dépôt :

```
cd ; borg umount /tmp/repo
```

- Se connecter en *geonat* sur *bkp-srv* :
 - Se placer dans le dossier de la stack *preprod* :

```
cd ~/docker/preprod/
```

- Restauration de **GeoNature** :
 - Lancer la restauration de la base de donnée (**ATTENTION**. Ex. avec *geonature2db* : cela supprimera la base en place) :

```
docker compose run --rm preprod-postgres-restore
/restore/restore.sh -d "2023-06-14_geonature2db.custom"
```

- Restauration de **GeoNature Atlas** :
 - Lancer la restauration de la base de donnée (**ATTENTION**. Ex. avec *gnatlas* : cela supprimera la base en place) :

```
docker compose run --rm preprod-postgres-restore
/restore/restore.sh -d "2023-06-23_gnatlas.custom"
```

- **ATTENTION** : La restauration précédente va bloquer sur le rafraîchissement de la VM "observations" pour une raison encore inconnue. Stopper la restauration avec CTRL+C. Les tables et les VMs sont malgré tout générés dans la base. Il suffit donc de rafraîchir l'ensemble des VMs pour finaliser la restauration.
- Rafraîchir les VMs grâce au script *gnatlas_refresh_all.sql* :

```
docker compose run --rm preprod-postgres-restore psql -U
geonatadmin -e -d gnatlas -f /restore/gnatlas_refresh_all.sql
```

- Notes : si **le fichier de la base de données prend trop de place** dans */home/geonat/docker/preprod/postgresql/restore/*, il est possible de le monter directement au bon endroit avec le paramètre `--volume`
 - Ex. :

```
docker compose run --volume
/data/tmp/restore/2025-03-20_geonature2db.custom:/restore
/2025-03-20_geonature2db.custom --rm preprod-postgres-
restore /restore/restore.sh -d
"2025-03-20_geonature2db.custom"
```

Docker Compose, Services SystemD et Nginx Proxy

- Pour fonctionner la pré-prod nécessite d'avoir démarré dans l'ordre suivant :
 1. le serveur PostgreSQL *preprod-postgres* de la stack Docker Compose "preprod"
 2. le fonctionnement sur l'hôte des services SystemD de GeoNature :

```
systemctl status geonature
systemctl status usershub
systemctl status taxhub
systemctl status geonature-atlas
```

3. le serveur Nginx *preprod-nginx* de la stack Docker Compose "preprod"

Installer le dépôt `sinp-<region>-data`

- Se connecter en `geonat` sur `bkp-srv` :

```
ssh geonat@bkp-<region>-sinp
```

- Deux solutions pour créer le dossier `~/data` et son contenu :
 - Copier directement le dossier `~/data` de production présent sur le serveur `db-srv` puis modifier les fichiers de config si nécessaire.
 - Cloner le dépôt dans un dossier `~/data/` ce qui implique ensuite de copier tous les fichiers de config. Ex. :

```
git clone --recursive https://github.com/cbn-alpin/sinp-aura-data.git data
```

- Se placer dans le dossier du dépôt :

```
cd ~/data/
```

- Installer Pipx :

```
sudo apt install pipx
```

- Configurer les chemins d'accès :

```
pipx ensurepath
```

- Relancer le terminal pour prendre en compte les changements ou essayer :

```
source ~/.bashrc
```

- Vérifier que Pipx est fonctionnel :

```
pipx --version
```

- Installer Pipenv :

```
pipx install pipenv
```

- Vérifier que Pipenv est fonctionnel :

```
pipenv --version
```

- Installer les dépendances de `import-parser` avec : `cd ~/data/import-parser/; pipenv install`
- Installer les dépendances de `gn2pg` avec : `cd ~/data/gn2pg/; pipenv install`
 - Si nécessaire, supprimer le dossier `~/gn2pg` si ce n'est pas un lien `ls -al ~/gn2pg` puis si c'est un dossier `rm -fr ~/gn2pg`
 - Recréer un lien symbolique : `ln -s ~/data/gn2pg/config ~/gn2pg`
- Copier TOUS les fichiers de config depuis le serveur `db-srv`

From:

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

Permanent link:

<https://sinp-wiki.cbn-alpin.fr/serveurs/installation/bkp-srv/pre-prod-geonature?rev=1742472245>

Last update: **2025/03/20 12:04**

