Installation de Borg

Ressources

- Borg Backup
- Borgomatic
- Docker Borgomatic
- Tuto mise en place de Borg

Mise en place du dépôt Borg

Créer un utilisateur "backups"

- Se connecter au serveur "bkp-srv" en tant qu'admin : `ssh admin@bkp-<region>-sinp`
- Passer en root : `sudo -i`
- Créer l'utilisateur "backups" sans mot de passe (connexion par clé SSH) avec son dossier */home/backups* : `useradd backups --create-home --home-dir /home/backups/ --shell /bin/bash`
- Créer le dossier qui contiendra les infos concernant SSH : `mkdir /home/backups/.ssh ; chmod 700 /home/backups/.ssh`
- Créer le fichier qui contiendra les clés SSH autorisées : `touch /home/backups/.ssh/authorized_keys ; chmod 600 /home/backups/.ssh/authorized_keys`
- Attribuer la propriété du dossier et de son contenu à l'utilisateur *backups* : `chown -R backups: /home/backups/.ssh`

Autoriser les machines distantes

Afin d'autoriser les machines accédant à l'utilsateur *admin* sur "bkp-srv" à accéder à l'utilisateur *backups*, nous allons copier le contenu du fichier */home/admin/.ssh/authorized_keys* :

- En tant que *root* sur "*bkp-srv*" : `cat /home/admin/.ssh/authorized_keys >> /home/backups/.ssh/authorized_keys`
- Tester la connexion depuis votre machine locale (aucun mot de passe ne doit être demandé) : `ssh backups@bkp-<region>-sinp`

Installer Borg sur "bkp-srv"

Sur l'instance "bkp-srv" hébergeant les dépôts Borg, nous installerons Borg avec Pip afin d'accéder aux dernières versions disponibles rapidement :

- Se connecter au serveur "bkp-srv" en tant qu'admin : `ssh admin@bkp-<region>-sinp`
- Passer en root : `sudo -i`
- En tant que *root*, installer les paquets système nécessaire :

    - Debian 11/12 avec Pyfuse3 : `apt install python3 python3-pip python3-virtualenv openssl libssl-dev libacl1-dev libacl1 build-essential pkg-config fuse3 libfuse3-dev`
    - Debian 10 avec llfuse: `apt install python3 python3-pip python3-virtualenv openssl libssl-dev libacl1-dev libacl1 build-essential pkg-config fuse libfuse-dev`
- En tant que *backups* (ou *root*), installer l'environnement virtuel : `virtualenv --python=python3 borg-env`
- Activer l'environnement : `source borg-env/bin/activate`
- Installer les dépendances : `pip install -U pip setuptools wheel`
- Installer Borg Backup avec le support pour Fuse, si le container utilise :
    - Debian 11/12 avec pyfuse3 : `pip install borgbackup[pyfuse3]`
    - Debian 10 avec llfuse : `pip install borgbackup[llfuse]`
- Sortir de l'environnement virtuel : `deactivate`
- **Notes** : il peut être intéressant d'installer *borg* sur l'utilisateur *root* afin de pouvoir accéder au dossier accessible uniquement par *root* sur le dépôt "db-srv". C'est le cas des fichiers de sauvegarde des bases de données.

Accéder à Borg sans activer l'environnement virtuel

Pour accéder à Borg sans activer l'environnement virtuel :

- Créer un dossier *~/bin* avec : `mkdir ~/bin`
- Modifier les fichiers :
    - ***~/.profile*** : ajouter le code ci-dessous au début du fichier *~/.profile* afin d'autoriser l'accès aux exécutable du dossier *~/bin* lors d'un accès par SSH avec shell interactif : `vi ~/.profile`
    - ***~/.bashrc*** : ajouter le code ci-dessous au début du fichier *~/.bashrc* avant la ligne de commentaire *# If not running interactively, don't do anything*. Sinon, l'accès à la commande *borg* via ssh avec un shell non interactif (ex. container *borgmatic*) ne fonctionnera pas !
    - Remplacer le code existant de création du dossier bin de l'utilisateur par celui-ci :

```
# Set PATH here to include user's private bin for SSH login
# It's necessary for using Borg !
if [[ -d "${HOME}bin" && ":$PATH:" != *":${HOME}bin:"* ]] ; then
    PATH="${HOME}bin:${PATH}"
fi
```

    - Pour root, ajouter le code ci-dessous :

```
# Set PATH here to include user's private bin for SSH login
# It's necessary for using Borg !
if [[ -d "/root/bin" && ":$PATH:" != *":/root/bin:"* ]] ; then
    PATH="/root/bin:${PATH}"
fi
```

- Vérifier que l'environnement virtuelle est bien désactivé : `deactivate`
- Recharger l'environnement : `source ~/.bashrc`

- Ajouter le lien symbolique vers l'executable de Borg : `ln -s ~/borg-env/bin/borg ~/bin/borg`
- Tester avec la version de Borg : `borg --version`

Mettre jour Borg

- En tant que *backups*, activer l'environnement : `source borg-env/bin/activate`
- Pour mettre à jour Borg :
  - Debian 11 avec pyfuse3 : `pip install -U borgbackup[pyfuse3]`
  - Debian 10 avec llfuse : `pip install -U borgbackup[llfuse]`

Sur les machines à sauvegarder

Préparer la connexion SSH

Le dépôt Borg étant distant (hébergé sur l'instance "bkp-srv"), il est nécessaire de créer un clé SSH pour l'utilisateur *admin* (qui héberge les containers Docker). La clé publique générée sera ensuite copier ans le fichier *~/.ssh/authorized_keys* de l'utilisateur *backups* de l'instance "*bkp-srv*" afin d'autoriser l'accès au dépôt depuis l'instance à sauvegarder.

- Se connecter à l'instance concernée : `ssh admin@<instance>-<region>-sinp`
- Générer une clé SSH : `ssh-keygen`
  - Accepter l'emplacement de stockage de la clef par défaut
  - Laisser une phrase de passe vide pour faciliter l'automatisation des sauvegardes
- Copier la clé publique sur le serveur "bkp-srv" dans le fichier *authorized_keys* de l'utilisateur *backups* afin de permettre une connexion SSH sans demande de mot de passe . L'utilisateur *backups* n'ayant pas de mot de passe, il faut passer par l'utilisateur *geonat* :
  - Copier de la clé publique : `scp -P <port-ssh-bkp-srv> ~/.ssh/id_rsa.pub geonat@bkp-<region>-sinp:~/.ssh/id_rsa.<instance>-srv.pub`
  - Se connecter au serveur "bkp-srv" avec l'utilisateur *admin* : `ssh admin@bkp-<region>-sinp`
  - Passer en root : `sudo -i`
  - Ajouter la clé publique de l'instance à sauvegarder au fichier */home/backups/.ssh/authorized_keys* de l'utilisateur *backups* : `cat /home/geonat/.ssh/id_rsa.<instance>-srv.pub >> /home/backups/.ssh/authorized_keys`
  - Tester la connexion SSH depuis un shell de l'utilisateur *admin* de l'instance à sauvegarder, aucun mot de passe ne doit être demandé : `ssh -p <port-ssh-bkp-srv> backups@bkp-<region>-sinp`
  - Si la tentative de connexion s'est déroulé sans demande de mot de passe et avec succès, supprimer le fichier de la clé public de l'instance : `rm -f /home/geonat/.ssh/id_rsa.<instance>-srv.pub`
- Sur l'instance à sauvegarder et pour l'utilisateur admin, créer un fichier config qui contiendra un paramètre maintenant la connexion SSH : `touch ~/.ssh/config; chmod 644 ~/.ssh/config ; vi ~/.ssh/config`
  - Y ajouter le contenu suivant :

```
Host *
```

```
    ServerAliveInterval 240
```

- Tester à nouveau la connexion SSH depuis un shell de l'utilisateur *admin* de l'instance à sauvegarder, aucun mot de passe ne doit être demandé : `ssh -p <port-ssh-bkp-srv> backups@bkp-<region>-sinp`
- Créer le dossier qui contiendra les informations *SSH* utilisées par le docker *Borgmatic* : `mkdir -p ~/docker/borgmatic/data/.ssh`
- Copier le dossier *~/.ssh* de l'utilisateur *admin* dans le dossier *data* de Borgmatic : `sudo cp -r ~/.ssh/ ~/docker/borgmatic/data/`
    - l'associer à l'utilisateur *root* car c'est l'utilisateur *root* qui l'utilisera dans le container : `sudo chown root: -R ~/docker/borgmatic/data/.ssh`
    - **ATTENTION** : Le docker *Borgmatic* se connectera à l'instance "*bkp-srv*" avec les infos contenues dans le dossier `~/docker/borgmatic/data/.ssh`.

Installer le container de sauvegarde

- Au préalable, mettre à jour les fichiers Docker pour l'instance concernée.
- Réaliser l'installation du container Docker Borgmatic

Préparer la sauvegarde des bases de données

- Afin de pouvoir accéder au port 5432 sur l'IP "gateaway" du container Docker, il faut modifier le fichier : `vi postgresql.conf`
    - Ajouter l'IP **172.18.0.1** au paramètre *listen_addresses*
- Il faut ensuite configurer l'accès aux bases à sauvegarder depuis l'IP du container Borgmatic. Le container créé a une IP privée au format *192.168.xxx.xxx*, nous ajoutons donc avec `/16` un ensemble d'IP privé qui devrait convenir dans la plupart des cas. Si nécessaire vérifier l'IP obtenu par votre container à l'aide de Portainer par exemple. Pour éditer la config de Postgresql : `vi pg_hba.conf`
    - Ajouter :

```
host    geonature2db    geonatadmin     172.18.5.1/16
scram-sha-256
host    gnatlas         geonatadmin     172.18.5.1/16
scram-sha-256
# Voir s'il n'est pas possible d'utiliser l'utilisateur
geonatadmin à la place de postgres...
host    gnatlas         postgres        172.18.5.1/16
trust
host    geonature2db    postgres        172.18.5.1/16
trust
```

- Il faut aussi s'assurer que le service Systemd de Postgresql démarre toujours après celui de Docker si l'on veut que Postgresql écoute sur l'IP 172.18.5.1. Pour cela surcharger le service de Postgresq: `systemctl edit postgresql@.service`
    - Ajouter le contenu suivant :

```
[Unit]
```

```
After=docker.service
```

    - Vérifier la présence du fichier : `vi /etc/systemd/system/postgresql@.service.d/override.conf`

Initialiser des dépôts

- Générer en local un UUID pour la *passphrase* de la clé de cryptage des dépôts local et distant de l'instance concernée : `uuid`
    - Stocker cette *passphrase* dans Keepass
- Sur l'instance *bkp-srv*, en tant qu'utilisateur `backups`, créer le dossier qui contiendra les dépôts : `sudo -i ; su - backups ; mkdir ~/<instance>-srv`
- Se connecter à l'instance concernée : `ssh admin@<instance>-<region>-sinp`
    - Modifier le fichier *~/docker/borgmatic/.env* de l'instance en ajoutant sa *passphrase* comme valeur du paramètre `BORG_PASSPHRASE` : `vi ~/docker/borgmatic/.env`
- Accéder au container *borgmatic* pour initialiser le dépôt de l'instance : `docker exec -it borgmatic /bin/bash`
- Établissez une première connexion au serveur du dépôt distant via SSH pour enregistrer le serveur distant : `ssh -p <port-ssh-bkp-srv> backups@<bkp-srv-private-ip>`
- À l'aide de *Borgmatic*, initialiser les dépôts de l'instance (présent dans le fichier de config de *Borgmatic*). Borgmatic se charge d'exécuter les commandes *Borg* : `borgmatic init --encryption repokey-blake2`
    - Si besoin, vous pouvez réinitialiser les dépôts en suivant la procédure "Réinitialiser un dépôt" décrite ci-dessous.
- Pour voir les infos concernant un dépôt :
    - Dépôt de "db-srv" : `borg info ssh://backups@10.0.1.30:<ssh-port-bkp-srv>/home/backups/db-srv/`
    - Dépôt de "web-srv" : `borg info ssh://backups@10.0.1.30:<ssh-port-bkp-srv>/home/backups/web-srv/`
    - Le dépôt local à chaque instance : `borg info /mnt/borg-repository`
- Export les clés des dépôts et les stocker dans Keepass :
    - Instance "db-srv" :
        - Dépôt distant : `borg key export ssh://backups@10.0.1.30:<ssh-posrt-bkp-srv>/home/backups/db-srv/ /root/.config/borg/db-srv-dist.repo-conf-file.txt ; cat /root/.config/borg/db-srv-dist.repo-conf-file.txt`
        - Dépôt local : `borg key export /mnt/borg-repository /root/.config/borg/db-srv-local.repo-conf-file.txt ; cat /root/.config/borg/db-srv-local.repo-conf-file.txt`
    - Instance "web-srv" :
        - Dépôt distant : `borg key export ssh://backups@10.0.1.30:<ssh-posrt-bkp-srv>/home/backups/web-srv/ /root/.config/borg/web-srv-dist.repo-conf-file.txt ; cat /root/.config/borg/web-srv-dist.repo-conf-file.txt`
        - Dépôt local : `borg key export /mnt/borg-repository /root/.config/borg/web-srv-local.repo-conf-file.txt ; cat /root/.config/borg/web-srv-local.repo-conf-file.txt`
- Si l'instance *bkp-srv* comprend un espace disque supplémentaire monté sur `/data`, il peut être intéressant de déplacer via un lien symbolique les dépôts Borg créés :
    - Créer le dossier qui contiendra les dépôts : `mkdir /data/borg-repos`

    - Donner les droits à l'utilisateur `backups` avec : `chown backups: /data/borg-repos`
    - Déplacer les dépôts : `mv /home/backups/*-srv /data/borg-repos/`
    - En tant qu'utilisateur *backups* : `ln -s /data/borg-repos/web-srv /home/backups/web-srv ; ln -s /data/borg-repos/db-srv /home/backups/db-srv`
    - Tester l'accès distant depuis un container *borgmatic* : `borg info ssh://backups@10.0.1.30:<ssh-posrt-bkp-srv>/home/backups/web-srv/`
- Tester une sauvegarde manuellement pour vérifier que tout fonctionne : `borgmatic --verbosity 2 --stats --files`

Réinitialiser un dépôt

S'il s'avérait nécessaire de réinitialiser un dépôt, la démarche à suivre est la suivante :

- Vérifier les archives actuellement présentes dans les dépôts : `borgmatic list`
- Supprimer un dépôt : `borgmatic borg delete <repo>`
    - Ex. : `borgmatic borg delete /mnt/borg-repository/`
    - ATTENTION : normalement la commande doit demander confirmation. Si cela ne fonctionne pas essayer la commande sans préciser le dépôt : `borgmatic borg delete`. La commande devrait demander confirmation de la suppression de chaque dépôt.
- Initialiser à nouveau les dépôt : `borgmatic init --encryption repokey-blake2`
- Vérifier à nouveau les dépôt : `borgmatic list`
- Récupérer et stocker dans Keepass les nouvelles clés des dépôts (voir la section "Initialisation des dépôts").

Tester le container de sauvegarde

Accéder au container *borgmatic* pour vérifier et tester son contenu : `docker exec -it borgmatic /bin/bash`

Tester l'envoi des emails

- À l'email adminsys : `echo "THIS IS A TEST EMAIL sended to adminsys at $(date "+%F %H:%M")" | mail -s "[${HOSTNAME}] Test email" adminsys@<domaine-sinp>`
- À un utilisateur système *root* : `echo "THIS IS A TEST EMAIL sended to root at $(date "+%F %H:%M")" | mail -s "[${HOSTNAME}] Test email" root`

Tester l'envoi de message Telegram

- Envoyer un message : `telegram-send "Test at `date`"`

Tester Borgmatic

- Tester les fichiers de configuration de *Borgmatic* avec : `borgmatic config validate` (anciennement `validate-borgmatic-config`)
- **ATTENTION** : il est maintenant nécessaire d'initialiser les dépôts avant de lancer la commande de test ci-dessous. Voir la section "Initialisation des dépôts".
- Tester une sauvegarde manuellement pour vérifier que tout fonctionne : `borgmatic --verbosity 2 --stats --files`

Tester le cron de Borgmatic

- Éditer le fichier */etc/cron.d/borgmatic* : `vi /etc/cron.d/borgmatic`
- Vérifier l'heure de lancement par défaut. Chaque instance devrait avoir une heure différente de sauvegarde. Il vaut mieux éviter de sauvegarder sur les dépôts distants hébergés sur le serveur *bkp-srv* au même moment.
- Modifier l'heure de lancement par défaut, pour que la sauvegarde démarre seulement quelques minutes plus tard
- **ATTENTION** : NE PAS Enregistrer les changement avec *Crontab*. *Crontab* et les fichiers présent dans le dossier *cron.d* sont gérés indépendamment. Si vous rajouter le fichier présent dans */ect/cron.d/* à *Crontab* 2 processus distincts seront lancés. Du coup, *Borg* affichera des erreurs liées au fichier *lock* des dépôts.
- Si vous avez correctement configurer l'envoie de message via Telegram, vous devriez recevoir des messages indiquant le démarrage de la sauvegarde à l'heure que vous avez indiqué. Sinon, vous devriez pouvoir voir les logs sur l'interface de Portainer.

Tester le dump Postgresql

- Contexte : tester manuellement le dump lancé par Borgmatic dans le container (si Postgresql est installé sur votre instance).
- Lancer le dump. Ex. :

```
PGPASSWORD=<password-geonatadmin> pg_dump --verbose --no-password --clean --if-exists --host 172.18.5.1 --port 5432 --username geonatadmin --format custom geonature2db > /root/geonature2db
```

- ATTENTION : dans les logs du container, le dump est lancer avec la variable d'env PGPASSWORD qui est non visible !
- Pour réaliser le dump de la base, il faut un utilisateur avec des droits de super utilisateurs, sinon il y a des risques que le dump échoue car l'utilisateur n'aura pas les droits suffisant sur certaines tables ou autres éléments de la base.

Utiliser Telegram

Configurer Telegram pour l'utiliser avec `Ntfy` ou le script `telegram-send` et envoyer des alertes à l'aide de Borgmatic. Ntfy nécessite d'autoriser manuellement un bot après chaque build/lancement d'un container. Ce n'est pas très pratique. Il vaut mieux utiliser le script `telegram-send` à la place.

- Installer Telegram sur votre Smartphone et/ou machine locale
- Créer un nouveau Bot d'alerte en envoyant un message au contact BotFather : `/newbot`
  - Indiquer son nom : `<region> SINP Alert Bot`

    - Indiquer comment l'appeler : `<region>SinpAlertBot`
    - Une fois créé, accéder à la liste de vos bot : `/mybots`
        - Choisir le bot `<region>SinpAlertBot`
        - Cliquer sur "API Token" pour en créer un nouveau, le copier dans le presse-papier et l'enregistrer sur Keepass.

Configurer Telegram avec le script ''telegram-send''

- Le script `telegram-send` peut être utilisé pour envoyer des messages directement à un utilisateur ou à un canal (=group)
- Si vous souhaitez créer un "Canal" pour que le bot y envoie les messages :
    - Dans l'interface de Telegram, via le menu, choisir "Nouveau canal"
        - Indiquer le nom du canal : "<REGION> SINP ALERT"
        - Indiquer la description : "Message d'infos et d'alertes du SINP <REGION>."
        - Cliquer sur "*Créer*"
        - Choisir un canal de type : *privé*
    - Ajouter le bot `<region>SinpAlertBot` au canal
        - Le promouvoir en tant qu'admin et lui donner seulement les droits de publier des messages
- Pour récupérer l'ID du canal (=group) et ou simplement l'ID d'un utilisateur, transférer un message envoyé sur le canal ou à l'utilisateur au bot getidsbot.
    - Il affiche les infos concernant le message dont les id qui vous intéresse : le votre ou celui du canal.
        - Afin que tout le monde reçoive bien les notification, utiliser l'ID du Canal.
    - Créer des entrées dans Keepass pour ces infos et renseigner les entrées *TELEGRAM_BOT_TOKEN* et *TELEGRAM_GROUP_ID* du fichier *.env*
- Redémarrer le container pour prendre en compte les changements du *.env* : `docker-compose down ; docker-compose up -d`
- Tester l'envoie de message :
    - Se connecter au container *borgmatic* : `docker exec -it borgmatic /bin/bash`
    - Envoyer un message : `telegram-send "Test at `date`"`

Configurer Telegram avec Ntfy

- Sur l'instance de serveur que vous souhaiter sauvegarder, se connecter au container *borgmatic* : `docker exec -it borgmatic /bin/bash`
    - **ATTENTION** : cette manipulation est à refaire à chaque nouvelle construction du container !
    - Lancer la commande : `ntfy -b telegram send "Telegram configured for ntfy"`
        - Coller le token précédemment copier lors de la création du bot
            - Copier le mot de passe qui s'affiche
        - Sur Telegram, ajouter votre bot en contact et le démarrer avec : `/start`
            - Coller le mot de passe précédemment copié généré par Ntfy
            - Un message de succès devrait s'afficher : *Telegram configured for ntfy*
    - Tester l'envoie de message sur votre bot : `ntfy -b telegram -t Borgmatic send "Borgmatic: test at `date` on ${HOSTNAME} !"`

- Si tout fonctionne correctement, un message devrait être reçu sur votre Bot dans l'interface de Telegram.

Restaurer une sauvegarde

Préparer la restauration

- Se connecter à l'instance concernée : `ssh admin@<instance>-<region>-sinp`
- Se placer dans le dossier *~/docker/borgomatic* avec : `cd ~/docker/borgomatic`
- Arrêter le container de sauvegarde : `docker compose down`
- Sur l'hôte, créer un dossier temporaire qui contiendra les fichiers ou dossier à restaurer : `sudo rm -fR /tmp/restore ; mkdir /tmp/restore`
- Lancer un shell interactif avec le dossier précédemment créé lié à l'hôte : `docker compose -f docker-compose.yml -f docker-compose.restore.yml run -v /tmp/restore:/tmp/restore borgmatic`
- Vous pouvez alors utiliser les commandes borgmatic ou Borg

Récupérer des fichiers et/ou dossiers avec Borg

- Se connecter à l'instance "*bkp-srv*" en tant que root
- Liste les sauvegardes de l'instance :
  - *db-srv* : `borg list /home/backups/db-srv`
  - *web-srv* : `borg list /home/backups/web-srv`
- Créer un dossier :
  - pour le point de montage du dépôt Borg : `mkdir /tmp/repo`
  - pour le dossier qui contiendra les fichiers à restaurer : `mkdir /tmp/restore`
- Utiliser Fuse pour monter la sauvegarde sur le dossier du point de montage */tmp/repo*: `borg mount <repo> /tmp/repo`
  - Ex. avec le dépôt local : `borg mount /home/backups/db-srv /tmp/repo`
  - Le dossier */tmp/repo* contient les dossiers des différentes sauvegardes. Ex. de dossier sauvegardé le 17 mai 2021 à 13h57 : `web-srv-2021-05-17T13:57:02`
- Copier les fichiers à restaurer sur l'hôte en les copiant depuis le point de montage vers le dossier lié l'hôte :
  - Ex. pour *web-srv* : `cp -r "/tmp/repo/web-srv-2021-05-17T13:57:02/mnt/source/etc/cron.d/" /tmp/restore/`
  - Ex. pour *db-srv* : `cp -r /tmp/repo/db-srv-2022-04-08T01\:07\:55/root/.borgmatic/postgresql_databases/172.18.5.1/geonature2db /tmp/restore/`
  - Vérifier las présence des fichiers à restaurer : `ls -al /tmp/restore`
- Donner les droits d'accès à l'utilisateur admin : `chmod 644 /tmp/restore/geonature2db`
- Démonter le point de montage et quitter le shell interactif : `borg umount /tmp/repo && exit`
- Récupérer sur votre machine locale l'archive de base de données : `scp admin@bkp-aura-sinp:/tmp/restore/geonature2db ./2022-04-08_geonature2db.custom`
  - Voir la restauration en local d'une base serveur

Récupérer des fichiers et/ou dossiers avec Borgmatic

- Se placer dans le dossier partagé avec l'hôte car la commande `borgmatic extract` extrait par défaut une archive ou un chemin particulier dans le dossier courant : `cd /tmp/restore`
    - Toutefois, il est possible d'utiliser l'option `--destination` pour préciser l'emplacement de destination.
- Lister les archives (=sauvegardes) disponibles : `borgmatic list`
    - ATTENTION : la commande ci-dessus listes les archives de tous les dépôts. Récupérer bien le nom de l'archive correspondant au dépôt que vous indiquerez dans la commande d'extraction.
- Extraire l'archive "<archive-name>" du dépôt <repo> dans le dossier partagé avec l'hôte : `borgmatic extract --repository <repo> --archive "<archive-name>" --destination /tmp/restore`
    - Ex. : `borgmatic extract --repository /mnt/borg-repository --archive "web-srv-2021-05-17T13:57:02" --destination /tmp/restore`
- Pour extraire seulement un dossier de l'archive utiliser l'option `--path` avec le chemin du dossier sans slash au début. Ex. : `borgmatic extract --repository /mnt/borg-repository --archive "web-srv-2021-05-17T13:57:02" --path "mnt/source/etc" --destination /tmp/restore`
- Note concernant la localisation des éléments sauvegardés dans le dossier d'extraction `/tmp/restore/` (si le dossier d'extraction est `/tmp/restore/`) :
    - les fichier ou dossiers systèmes sauvegardés se trouvent dans `/tmp/restore/mnt/…`
    - les sauvegardes des bases de données se trouvent dans `/tmp/restore/root/.borgmatic/postgresql_databases/172.18.0.1/..`
- Quitter le container : `exit`

Écraser les fichiers à restaurer par ceux provenant de la sauvegarde sur l'hôte

- Sur l'hôte, écraser les fichiers à restaurer par ceux provenant de la sauvegarde : `sudo cp -r /tmp/restore/cron.d/ /etc/`
    - Assurez vous que les dossiers et fichiers sont restaurés avec les bons droits d'accès et propriétaire
- Une fois les fichiers/dossiers restaurés vous pouvez supprimer le dossier de restauration et son contenu : `rm -fR /tmp/restore`
- **NOTE** : si nécessaire, il est possible de modifier le fichier *docker-compose.restore.yml* pour monter directement le volume source en écriture `${VOLUME_SOURCE}:/mnt/source` afin de remplacer les fichiers de l'hôte directement depuis le shell interactif.

Restaurer une base de données avec Borgmatic

- Lister les archives disponibles : `borgmatic list`
- Restaurer une base de données directement. ATTENTION : Pour que cela fonctionne, il est nécessaire d'avoir utiliser une super utilisateur dans la config Borgmatic du dump des bases :

```
 borgmatic restore --repository <repo> --archive <archive> --database
<db-name>
```

    - Ex. :

```
 borgmatic restore --repository /mnt/borg-repository/ --archive "db-srv-2021-05-17T01:00:02" --database geonature2db
```

    - Si la commande échoue, récupérer dans la console la ligne de commande qui s'affiche, supprimer l'option `--no-password` et lancer la commande manuellement. Des erreurs plus détaillées s'afficheront.
- Quitter le container : `exit`

Extraire l'archive de la base et la récupérer localement

- Sur l'hôte, copier le fichier : `sudo cp /tmp/restore/root/.borgmatic/postgresql_databases/172.18.0.1/geonature2db "/tmp/$(date +'%Y-%m-%d')_geonature2db.custom"`
- Changer les droits : `sudo chown admin: "/tmp/$(date +'%Y-%m-%d')_geonature2db.custom"`
- Depuis votre machine locale récupérer l'archive : `scp admin@db-<region>-sinp:/tmp/$(date +'%Y-%m-%d')_geonature2db.custom ./`

Finaliser la restauration

- Penser à relancer le service *Borgmatic* : `docker compose up -d`
- Au cas ou *Borg* échoue à créer/acquérir un verrou : `borg break-lock <repo>`

Cas particulier de la base GeoNature

- Commencer par extraire une archive dans le dossier `/tmp/restore` comme indiqué ci-dessus.
    - Ex. :

```
 borgmatic extract --repository ssh://backups@10.0.1.30:<port-ssh-bkp-srv>/home/backups/db-srv/ --archive "db-srv-2021-05-19T16:09:01" --destination /tmp/restore
```

- Supprimer les connexions à la base existante :

```
 psql -U postgres -h 172.18.0.1 -d gnatlas -c "SELECT pg_terminate_backend(pg_stat_activity.pid) FROM pg_stat_activity WHERE pg_stat_activity.datname = 'geonature2db' ; "
```

- Supprimer la base :

```
 psql -U postgres -h 172.18.0.1 -d gnatlas -c "DROP DATABASE geonature2db;"
```

- Recréer une base vide :

```
 psql -U postgres -h 172.18.0.1 -d gnatlas -c "CREATE DATABASE geonature2db;"
```

- Restaurer la base :

```
 pg_restore --if-exists --exit-on-error --clean --dbname geonature2db --host 172.18.0.1 --port 5432 --username postgres /tmp/restore/root/.borgmatic/postgresql_databases/172.18.0.1/geonature2db
```

Si nécessaire, il est aussi possible de manipuler les éléments à restaurer :

- Créer la liste des éléments à restaurer :

```
 pg_restore --list /tmp/restore/root/.borgmatic/postgresql_databases/172.18.0.1/geonature2db > gn2.list
```

  - Éditer le fichier : `vi gn2.list`
  - Vous pouvez par exemple commenter les lignes mentionnant `EXTENSION` puis enregistrer et quitter.
- Restaurer la base avec cette liste :

```
 pg_restore --if-exists --exit-on-error --clean --dbname geonature2db --host 172.18.0.1 --port 5432 --username postgres --use-list gn2.list /tmp/restore/root/.borgmatic/postgresql_databases/172.18.0.1/geonature2db
```

Cas particulier de la base GeoNature Atlas

En fonction du format de sauvegarde de la base et de l'utilisateur qui l'effectue, la restauration de la base de données ne pourra pas se faire avec la commande `borgmatic restore`. La base gnatlas contient des vues matérialisées qui posent problèmes car elles sont basées sur des tables "étrangères" (FDW). Or, les informations du Foreign Data Wrapper ne sont pas correctement restaurées (utilisateur, mot de passe) par *pg_restore* quand la sauvegarde est au format "directory". Avec la version 13+ de Postgresql ce comportement devrait être amélioré. Du coup, la restauration par *borgmatic* ne fonctionne pas et celle par *pg_restore* échoue si l'on ne supprime pas les requêtes de rafraîchissement des VM car les tables FDW ne contiennent pas de données et le rafraîchissement des vues échoue.

- Suivre les étapes précédentes visant à extraire une archive dans le dossier /tmp/restore à l'aide de Borgmatic.
- Le dossier de la sauvegarde de la base *gnatlas* à fournir à *pg_restore* se trouvera dans : `/tmp/restore/root/.borgmatic/postgresql_databases/172.18.0.1/gnatlas`
- Si le format "**plain**" a été utilisé dans le fichier de config de *Borgmatic* :
  - Il est simplement nécessaire d'utiliser *Psql* avec l'utilisateur *postgres* :

```
 psql -U postgres -h 172.18.0.1 -p 5432 -d gnatlas -f "/tmp/restore/root/.borgmatic/postgresql_databases/172.18.0.1/gnatlas"
```

- Si le format "**custom**" ou "**directory**" (qui fonctionne pour *gnatlas*) a été utilisé dans le fichier de config de *Borgmatic*, nous pouvons utiliser *pg_restore* :
    - Créer un fichier qui contient la liste des requêtes a exécuter par pg_restore. Nous supprimerons de la liste le rafraichissement des vues :

```
 pg_restore --list
/tmp/restore/root/.borgmatic/postgresql_databases/172.18.0.1/gnatl
as | grep -v "MATERIALIZED VIEW DATA" > without_refresh.list
```

    - Lancer la restauration de la base en executant seulement les requêtes listés dans le fichier `without_refresh.list` :

```
 pg_restore --if-exists --exit-on-error --clean --dbname gnatlas -
-host 172.18.0.1 --port 5432 --username postgres --use-list
without_refresh.list
/tmp/restore/root/.borgmatic/postgresql_databases/172.18.0.1/gnatl
as
```

    - Après avoir restaurer la base, il est nécessaire de reconfigurer le mapping utilisateur du FDW puis de peupler les vues matérialisées avec les données :
        - Sur l'instance "db-srv", lancer les requêtes suivantes :
            - Ajouter l'utilisateur distant au mapping de l'utilisateur "geonatadmin" utilisant la base "gnatlas" :

```
sudo -u postgres -s psql -d gnatlas -c "ALTER USER
MAPPING FOR geonatadmin SERVER geonaturedbserver OPTIONS
(ADD user 'geonatadmin');"
```

            - Ajouter le mot de passe de l'utilisateur distant :

```
sudo -u postgres -s psql -d gnatlas -c "ALTER USER
MAPPING FOR geonatadmin SERVER geonaturedbserver OPTIONS
(ADD password '<pwd>');"
```

            - Lancer ensuite le rafraichissement des vues :

```
 psql -h localhost -U geonatadmin -d gnatlas -f
/home/geonat/data/atlas/data/sql/02_refresh_mv_data.sql
```

Si besoin, pour voir ce qui existe dans la base concernant FDW:

- Se connecter à Postgresql en tant que *postgres* :

```
 sudo -u postgres -s psql -d gnatlas
```

- Lister les serveurs FDW :

```
SELECT * FROM pg_foreign_server;
```

- Lister le USER MAPPING :

```
SELECT um.*,rolname FROM pg_user_mapping um JOIN pg_roles r ON r.oid =
```

```
umuser JOIN pg_foreign_server fs ON fs.oid = umserver;
```

- Sortir : `exit`

Borg commandes utiles

- Supprimer le cache :

```
borg delete --cache-only <repo>
```

- Au cas ou Borg échoue à créer/acquérir un verrou :

```
borg break-lock <repo>
```