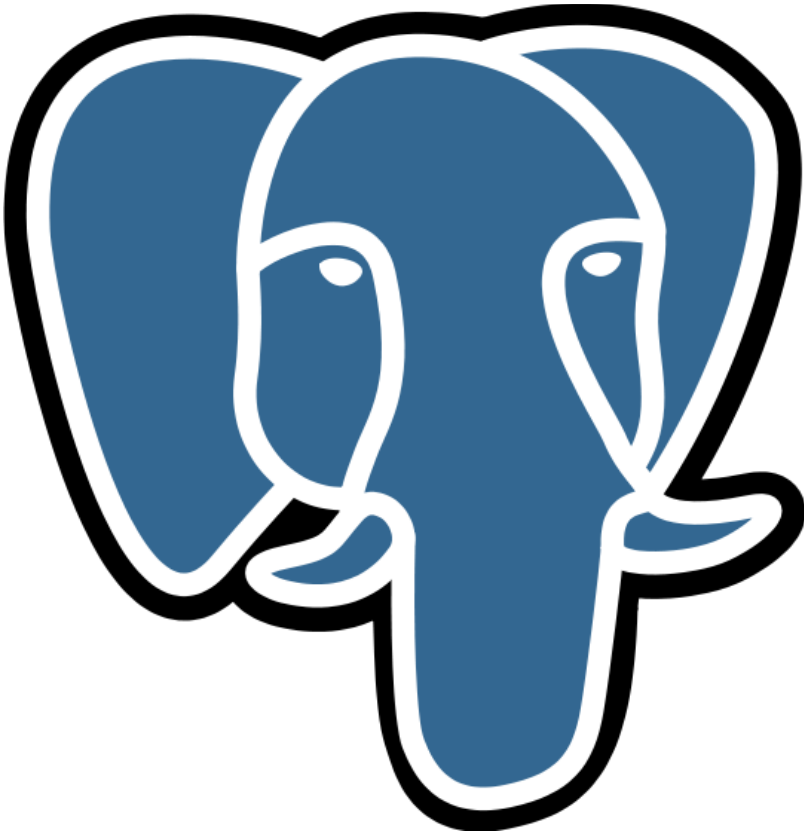
DALIBO

L'expertise PostgreSQL

Outils de sauvegarde

Table des matières

PostgreSQL : Outils de sauvegarde



1.1 Introduction

- 2 mécanismes de sauvegarde natifs et robustes.
- Industrialisation fastidieuse.
- Des outils existent !!

Nous avons vu le fonctionnement interne du mécanisme de sauvegarde physique. Celui-ci étant en place nativement dans le moteur PostgreSQL depuis de nombreuses versions, sa robustesse n'est plus à prouver, cependant, son industrialisation reste fastidieuse. Des outils tiers existent et

vont permettre de faciliter la gestion des sauvegardes, de leur mise en place jusqu'à la restauration. Dans ce module nous allons voir en détail certains de ces outils et étudier les critères qui vont nous permettre de choisir la meilleure solution selon notre contexte.

1.2 Au menu

- Présentation:
  - pg_back
  - Barman
  - pitrery
- Comment choisir ?

Lors de cette présentation, nous allons passer en revue les différents outils principaux de gestion de sauvegardes, leurs forces, le paramétrage, l'installation et l'exploitation.

1.3 Licence Creative Commons CC-BY-NC-SA

Vous êtes libres de redistribuer et/ou modifier cette création selon les conditions suivantes :

- Paternité
- Pas d'utilisation commerciale
- Partage des conditions initiales à l'identique

1.4 Définition du besoin - Contexte

- Sauvegarde locale (ex. NFS) ?
- Copie vers un serveur tiers (push) ?
- Sauvegarde distante initiée depuis un serveur tiers (pull) ?
- Ressources à disposition ?
- Accès SSH ?
- OS ?
- Sauvegardes physiques ? Logiques ?
- Version de PostgreSQL ?
- Politique de rétention ?

Où les sauvegardes doivent-elles être stockées ? Quelles ressources sont à disposition : serveur de sauvegarde dédié ? quelle puissance pour la compression ? De quel type d'accès aux serveurs de base de données dispose-t-on ? Quelle est la version du système d'exploitation ? Il est très important de se poser toutes ces questions, les réponses vont servir à établir le contexte et permettre de choisir l'outil et la méthode la plus appropriée.

2 pg_back

2.1 pg_back - Présentation

- Type de sauvegardes: **logiques** (pg_dump)
- Langage: **bash**
- Licence: **BSD** (libre)
- Type de stockage: **local**
- Planification: **crontab**
- OS: **Unix/Linux**
- Compression: **gzip**
- Versions compatibles: **toutes**
- Rétention: **durée** en jour

pg_back (https://github.com/orgrim/pg_back) est un outil écrit en bash par Nicolas Thauvin de Dalibo, également auteur de pitrery. Il s'agit d'un script assez complet permettant de gérer simplement des sauvegardes logiques (`pg_dump`, `pg_dumpall`), y compris au niveau de la rétention.

L'outil se veut très simple et facile à adapter, il peut donc être facilement modifié pour mettre en place une stratégie de sauvegarde logique plus complexe.

L'outil ne propose pas d'option pour restaurer les données, il faut pour cela utiliser les outils interne de PostgreSQL (`pg_restore`, `psql`).

3 pg_basebackup

3.1 pg_basebackup - Présentation

- Outil intégré à PostgreSQL
- Prévu pour créer une instance secondaire
- Permet les sauvegardes PITR

pg_basebackup (http://www.postgresql.org/docs/current/static/app-pgbasebackup.html) est une application cliente intégrée à PostgreSQL, au même titre que pg_dump ou pg_dumpall.

Elle est conçue pour permettre l'initialisation d'une instance secondaire en réplication. Cette opération consiste en une opération de sauvegarde PITR, pg_basebackup peut donc être utilisé pour faciliter l'opération de sauvegarde d'un instance.

3.2 pg_basebackup - Formats de sauvegarde

- `plain`
  - arborescence identique à l'instance sauvegardée
- `tar`
  - archive, permet la compression

Le format par défaut de la sauvegarde est `plain`, ce qui signifie que les fichiers seront créés tels quels dans le répertoire de destination (ou les répertoires en cas de tablespaces).

L'option `-F tar` active le format `tar`. pg_basebackup génère alors une archive `base.tar` pour le PGDATA de l'instance, puis une archive `<oid>.tar` par tablespace. Ce format permet l'activation de la compression `gzip` à l'aide de l'option `-z`. Le niveau de compression peut également être spécifié avec l'option `-Z <niveau>`.

3.3 pg_basebackup - Avantages

- Sauvegarde possible à partir d'un secondaire
- Transfert des WAL pendant la sauvegarde
- Débit configurable
- Relocalisation des tablespaces
- Écriture d'un `recovery.conf`
  - prévu pour la réplication

L'un des avantages de pg_basebackup est qu'il permet nativement de réaliser une sauvegarde à partir d'une instance secondaire.

Autre avantage, avec pg_basebackup, il est possible de récupérer les fichiers WAl nécessaires à la restauration de la sauvegarde sans passer par la commande d'archivage. Pour cela, il faut à l'exécution de la commande utiliser l'option `-x` (ou `-X fetch`) pour récupérer les WAL générés pendant la sauvegarde une fois celle-ci terminée, à condition qu'ils n'aient pas été recyclés entre-temps (ce qui peut nécessiter la configuration d'slot de réplication, ou d'un `wal_keep_segments` élevé avant la 9.4). Il est également possible de récupérer les WAL non pas en fin de sauvegarde mais en streaming pendant celle-ci, à l'aide de l'option `-X stream` - cela nécessite néanmoins l'utilisation d'un *wal sender* supplémentaire, le paramètre `max_wal_senders` doit donc être adapté en conséquence. Il convient néanmoins de noter que si l'archivage des WAL n'est pas également actif par ailleurs, la sauvegarde effectuée ne sera utilisée que pour restaurer l'instance telle qu'elle était au moment de la fin de la sauvegarde - il ne sera pas possible de réaliser une restauration de type *Point In Time Recovery*.

Le débit de la sauvegarde est configurable avec l'option `-r` pour éviter d'impacter excessivement l'instance - il convient de noter que si les fichiers WAL sont transférés en streaming (`-X stream`), ce flux ne sera pas affecté par la restriction de débit.

Il est également possible de relocaliser les éventuels tablespaces dans des répertoires distincts avec l'option `-T <ancien chemin>=<nouveau chemin>`, et de relocaliser le répertoire des fichiers WAL avec l'option `--xlogdir=<nouveau chemin>`.

L'option `-R` permet de demander à pg_basebackup de générer un fichier `recovery.conf` dans le répertoire PGDATA de la sauvegarde. Néanmoins, ce fichier ne contiendra que les paramètres nécessaires à l'activation de la réplication *streaming* (`standby_mode` et `primary_conninfo`), il n'est donc pas prévu en l'état pour effectuer une restauration.

3.4 pg_basebackup - Limitations

- Configuration de type réplication nécessaire
- Pas de configuration de l'archivage
- Pas d'association WAL archivés / sauvegarde
- Pas de gestion de la restauration
- Pas de politique de rétention

pg_basebackup étant conçu pour la mise en place d'une instance en réplication, l'instance principale nécessite d'être configurée en conséquence :

- `max_wal_senders` doit avoir une valeur supérieure à 0 pour permettre à pg_basebackup de se connecter (au moins 2 si on utilise le transfert des WAL par streaming) ;
- le fichier `pg_hba.conf` de l'instance principale doit être configuré pour autoriser les connexions de type `replication` depuis le serveur depuis lequel la sauvegarde est déclenchée.

Si la sauvegarde est effectuée à partir d'une instance secondaire, les aspects suivants doivent être pris en compte :

- l'instance secondaire doit être configurée en `host_standby` ;
- `max_wal_senders` doit avoir une valeur supérieure à 0 pour permettre à pg_basebackup de se connecter ;
- une attention particulière doit être apportée au fait que tous les fichiers WAL nécessaires à la restauration ont bien été archivés ;
- l'écriture complète des pages dans les WAL (paramètre `full_page_writes`) doit être activé.

L'archivage des fichiers WAL doit être configurée indépendamment à l'aide des commandes systèmes traditionnelles (`rsync`...).

Par ailleurs, la gestion des sauvegardes elles-mêmes n'est pas prévue dans l'outil. Notamment, pg_basebackup n'effectue pas de lien entre les WAL archivés et les sauvegardes effectuées, ne garde pas en mémoire les sauvegardes effectuées, ne permet pas de gérer de rétention.

L'outil ne permet pas non plus d'effectuer directement de restauration, il faut donc copier manuellement les fichiers à restaurer (par exemple avec `rsync`, ou `tar` si ce format a été sélectionné) vers le répertoire de destination de la restauration.

4 Barman

4.1 Barman - Présentation générale

- 2ndQuadrant IT
- Language: **python** 2.6/2.7
- OS: **Unix/Linux**
- Versions compatibles: **>= 8.3**
- License: **GPL3** (libre)
- Type d'interface: **CLI** (ligne de commande)

barman est un outil développé dans le langage python, compatible avec les version 2.6/2.7, compatible uniquement sur les environnements Linux/Unix. Il est principalement maintenu par la société 2ndQuadrant Italia et distribué sous license GPL3.

4.2 Barman - Diagrame

4.3 Barman - Sauvegardes

- Type de sauvegardes: **physiques/PITR** (à chaud)
- Type de stockage: **local** ou **pull** (rsync/ssh)
- Planification: **crontab**
- Méthode: **pg_start_backup() / rsync / pg_stop_backup()**
- Incrémentales: **rsync + hardlink**
- Compression des WAL

Barman gère uniquement des sauvegardes physiques. Il peut fonctionner soit en local (directement sur le serveur hébergeant l'instance à sauvegarder) pour un stockage des sauvegardes local, et peut aussi être exécuté depuis un serveur distant, déléguant ainsi l'ordonnancement, la compression et le stockage des données. Il est possible d'activer la déduplication de fichiers entre 2 sauvegardes.

La technique utilisée pour la prise de sauvegarde repose sur le mécanisme interne standard et historique : `pg_start_backup()`, copie des fichiers, `pg_stop_backup()`.

4.4 Barman - Sauvegardes (suite)

- Limitation du débit réseau lors des transferts
- Compression des données lors des transferts via le réseau
- Sauvegardes en parallèle via l'extension `pgespresso`
- Hook pre/post sauvegarde
- Hook pre/post WAL archivage
- Compression WAL: `gzip`, `bzip2`, `pigz`, `pbzip2`, etc..
- Pas de compression des données (sauf WAL)

Barman supporte la limitation du débit réseau lors du transfert des données sur un serveur tiers, ainsi que la compression des données à la volée le temps du transfert. Il peut être couplé à l'utilisation de l'extension `pgespresso`, permettant ainsi d'effectuer plusieurs sauvegardes en parallèle. Quatre niveaux de scripts ancrés (*hooks*) sont possibles :

- **avant la sauvegarde** ;
- **après la sauvegarde** ;
- **avant l'archivage d'un WAL** ;
- **après l'archivage d'un WAL**.

Attention, l'opération d'archivage citée ici est celle effectuée par Barman lorsqu'il déplace et compresse un WAL à partir du répertoire `incoming_wals/` vers le répertoire `wals/`, il ne s'agit pas de l'archivage au sens PostgreSQL.

4.5 Barman - Politique de rétention

- **Durée** (jour/semaine) et/ou **nombre** de sauvegardes.

La politique de rétention peut être exprimée soit en nombre de sauvegardes à conserver, soit en fenêtre de restauration : une semaine, 2 mois, etc.

4.6 Barman - Restauration

- Locale ou à distance
- Point dans le temps: date, identifiant de transaction, timeline ou point de restauration

La restauration d'une sauvegarde peut se faire soit localement, si les sauvegardes sont stockées en local, soit à distance, dans ce cas les données à restaurer seront transférées via SSH. Plusieurs types de point dans le temps peuvent être utilisés comme cible:

- La date
- Un identifiant de transaction
- timeline, en cas de divergence de timeline, `barman` peut restaurer les transactions issues d'une timeline précise.
- Un point de restauration créé par un appel préalable à la fonction `pg_create_restore_point_name()`

4.7 Barman - Installation

- Accéder au dépôt communautaire PGDG
- Installer le paquet `barman`

Barman est disponible sur le dépôt communautaire maintenu par la communauté PostgreSQL pour les systèmes d'exploitation disposant des gestionnaires de paquet APT (Debian, Ubuntu : http://apt.postgresql.org/pub/repos/apt/) ou YUM (RHEL, CentOS, Fedora : http://yum.postgresql.org/). Il est recommandé de manière générale de privilégier une installation à partir de paquets plutôt que par les sources, essentiellement pour des raisons de maintenance.

4.8 Barman - Utilisation

```
usage: barman [-h] [-v] [-c CONFIG] [-q] [-d] [-f {console}]
              {cron,list-server,show-
server,status,check,diagnose,backup,
              list-backup,show-backup,list-files,recover,delete,
              rebuild-xlogdb}

[...]

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
  -c CONFIG, --config CONFIG
                        uses a configuration file (defaults:
~/.barman.conf,
                        /etc/barman.conf,
/etc/barman/barman.conf)
  -q, --quiet           be quiet (default: False)
  -d, --debug           debug output (default: False)
  -f {console}, --format {console}
                        output format (default: console)
```

Barman propose différentes commandes pouvant être passées en argument afin de contrôler les actions. L'usage de ces différentes commandes sera détaillé ultérieurement.

L'option `-c` ou `--config` permet d'indiquer l'emplacement du fichier de configuration. L'option `-q`, ou `--quiet`, désactive l'envoi de messages sur la sortie standard. L'option `-f`, ou `--format`, n'offre pour le moment pas d'autre choix que `console`.

4.9 Barman - Configuration

- `/etc/barman.conf`
- Format `INI`
- Configuration générale dans la section `[barman]`
- Chaque instance à sauvegarder doit avoir sa propre section
- Un fichier de configuration / instance via la directive:

```
configuration_files_directory = /etc/barman.d
```

Le format de configuration `INI` permet de définir des sections, qui sont matérialisées sous la forme d'une ligne : `[nomdesection]`.

Barman s'attend à lire un fichier de configuration contenant la section `[barman]`, contenant les paramètres de configuration globaux, et une section par instance à sauvegarder, le nom de la section définissant ainsi le nom de l'instance. Pour des questions de lisibilité, il est possible de créer un fichier de configuration par instance à sauvegarder. Ce fichier doit alors se trouver (par défaut) dans le dossier `/etc/barman.d`. Le nom du fichier doit se terminer par `.conf` pour être pris en compte.

4.10 Barman - Configuration utilisateur

- Utilisateur système `barman`

L'utilisateur système `barman` est utilisé pour les connexions SSH, il faut donc penser à générer ses clefs RSA, les échanger et établir une première connexion avec les serveurs hébergeant les instances PostgreSQL à sauvegarder.

4.11 Barman - Configuration SSH

- Utilisateur `postgres` pour les serveurs PostgreSQL
- Utilisateur `barman` pour le serveur de sauvegardes
- Générer les clefs SSH (RSA) des utilisateurs système `postgres` (serveurs PG) et `barman` (serveur barman)
- Échanger les clefs SSH public entre les serveurs PostgreSQL et le serveur de sauvegarde
- Établir manuellement une première connexion SSH entre chaque machine

Dans le cadre de la mise en place de sauvegardes avec un stockage des données sur un serveur tiers, la plupart des outils et méthodes historiques de sauvegardes se reposent sur le protocole SSH et des outils tels que `rsync` pour assurer les transferts au travers du réseau. Afin d'automatiser ces transferts via le protocole SSH, il est impératif d'autoriser l'authentification SSH par clé, et d'échanger les clés publiques entre les différents serveurs hébergeant les instances PostgreSQL et le serveur de sauvegarde.

4.12 Barman - Configuration PostgreSQL

- Adapter la configuration de l'archivage dans le fichier `postgresql.conf` :

```
wal_level = 'hot_standby'
archive_mode = on
archive_command = 'rsync -a %p
barman@bkpsrv:<INCOMING_WALS_DIRECTORY>/%f'
```

La paramétrage de l'archivage des journaux de transaction reste classique, la directive `archive_command` doit faire appel directement à l'outil système en charge du transfert du fichier.

Attention à bien mettre le bon répertoire `<INCOMING_WALS_DIRECTORY>` pour l'emplacement de destination des WAL archivés, il s'agit du répertoire indiqué par la commande suivante :

```
barman show-server <instance>
```

4.13 Barman - Configuration globale

- Fichier `barman.conf`
- Section **`[barman]`** pour la configuration globale :

```
[barman]
barman_home = /var/lib/barman
barman_user = barman
log_file = /var/log/barman/barman.log
log_level = INFO
configuration_files_directory = /etc/barman.d
```

- **`barman_home`** : répertoire racine de travail de Barman, contenant les sauvegardes et les journaux de transactions archivés ;
- **`barman_user`** : utilisateur système ;
- **`log_file`** : fichier contenant les traces Barman ;
- **`configuration_files_directory`** : chemin vers le dossier d'inclusion des fichiers de configuration supplémentaires (défaut : `/etc/barman.d`) ;
- **`log_level`** : niveau de verbosité des traces, par défaut `INFO`.

4.14 Barman - Configuration sauvegardes

- Configuration globale des options de sauvegarde :

```
compression = gzip
reuse_backup = link
immediate_checkpoint = false
basebackup_retry_times = 0
basebackup_retry_sleep = 30
backup_options = exclusive_backup
```

- **compression** : méthode de compression des journaux de transaction - sont disponibles : gzip, bzip2, custom, laissant la possibilité d'utiliser l'utilitaire de compression de son choix (défaut : gzip) ;
- **reuse_backup** : paramétrage de la sauvegarde incrémentale (défaut : off)
  - link : déduplication de fichier basée sur des liens matériels (*hardlink*), économisant ainsi du temps et de l'espace disque ;
  - copy : copie locale des fichiers si identiques par rapport à la dernière sauvegarde, économisant ainsi du temps de transfert ;
  - off : sauvegarde complète.
- **immediate_checkpoint** : force la création immédiate d'un checkpoint impliquant une augmentation des écritures, le but étant de débuter la sauvegarde le plus rapidement possible (defaut : off) ;
- **basebackup_retry_times** : nombre de tentative d'écriture d'un fichier - utile pour relancer la copie d'un fichier en cas d'échec sans compromettre le déroulement global de la sauvegarde ;
- **basebackup_retry_sleep** : spécifié en secondes, il s'agit ici de l'intervalle de temps entre deux tentatives de copie d'un fichier en cas d'échec ;
- **backup_options** : exclusive_backup ou concurrent_backup, offre la possibilité d'activer la création de deux ou plusieurs sauvegardes de manière concurrente - nécessite l'installation de l'extension **pgespresso** (défaut : exclusive_backup).

4.15 Barman - Configuration réseau

- Possibilité de réduire la bande passante
- Et de compresser le trafic réseau

```
bandwidth_limit = 4000
network_compression = false
```

- **`bandwidth_limit`** : limitation de l'utilisation de la bande passante réseau lors du transfert de la sauvegarde, s'exprime en kbps (défaut : 0 soit pas de limitation) ;
- **`network_compression`** : activation de la compression à la volée des données lors du transfert réseau de la sauvegarde - utilisé à la sauvegarde ou lors d'une restauration (défaut : `false`).

4.16 Barman - Configuration rétention

- Configuration de la rétention en nombre de sauvegardes
- Et en « fenêtre de restauration », en jours, semaines ou mois
- Déclenchement d'une erreur en cas de sauvegarde trop ancienne

```
minimum_redundancy = 5
retention_policy = RECOVERY WINDOW OF 7 DAYS
last_backup_maximum_age = 2 DAYS
```

- **`minimum_redundancy`** : nombre minimum de sauvegardes à conserver - si n'est pas respecté, Barman empêchera la suppression (défaut : 0) ;
- **`retention_policy`** : définit la politique de rétention en s'exprimant soit en nombre de sauvegarde via la syntaxe `REDUNDANCY <valeur>`, soit en fenêtre de restauration via la syntaxe `RECOVERY OF <valeur> {DAYS | WEEKS | MONTHS}` (défaut : aucune rétention appliquée) ;
- **`last_backup_maximum_age`** : expression sous la forme `<value> {DAYS | WEEKS | MONTHS}`, définit l'âge maximal de la dernière sauvegarde - si celui-ci n'est pas respecté, lors de l'utilisation de la commande `barman check`, une erreur sera levée.

4.17 Barman - Configuration hooks

- Lancer des scripts avant ou après les sauvegardes
- Et avant ou après le traitement du WAL archivé par Barman

```
pre_backup_script = env | grep ^BARMAN
post_backup_script = env | grep ^BARMAN
pre_archive_script = env | grep ^BARMAN
post_archive_script = env | grep ^BARMAN
```

Barman offre la possibilité d'exécuter des commandes externes (scripts), avant et/ou après les opérations de sauvegarde, et les opérations d'archivage des journaux de transaction.

Attention, la notion d'archivage de journal de transaction dans ce contexte ne concerne pas l'archivage réalisé depuis l'instance PostgreSQL, qui copie les WAL dans un répertoire <incoming> sur le serveur Barman, mais bien l'opération de récupération du WAL depuis ce répertoire <incoming>.

4.18 Barman - Configuration par instance

Configuration par instance :

- `configuration_files_directory`
  - un fichier de configuration par instance
- Ou une section par instance

Après avoir vu les options globales, nous allons voir à présent les options spécifiques à chaque instance à sauvegarder.

Afin de conserver une certaine souplesse dans la gestion de la configuration Barman, il est recommandé de paramétrer la directive `configuration_files_directory` de la section `[barman]` afin de pouvoir charger d'autres fichiers de configuration, permettant ainsi d'isoler la section spécifique à chaque instance à sauvegarder dans son propre fichier de configuration.

4.19 Barman - Exemple configuration par instance

- Section spécifique par instance
- Permet d'adapter la configuration aux différentes instances

```
[pgsrv]
description = "PostgreSQL Instance pgsrv"
ssh_command = ssh postgres@pgsrv
conninfo = host=pgsrv user=postgres dbname=postgres
backup_directory =
basebackup_directory =
wals_directory =
incoming_wals_directory =
```

La première ligne définit le nom de la section, ce nom est important et doit être significatif car il sera utilisé lors des taches d'exploitation pour identifier l'instance cible. L'idéal est d'utiliser le nom d'hôte ou l'adresse IP du serveur si celui-ci n'héberge qu'une seule instance.

- **`description`** : chaîne de caractère servant de descriptif de l'instance ;
- **`ssh_command`** : commande shell utilisée pour établir la connexion `ssh` vers le serveur hébergeant l'instance à sauvegarder ;
- **`conninfo`**: chaîne de connexion PostgreSQL ;
- **`backup_directory`** : définit l'emplacement du dossier parent contenant les sauvegardes et les journaux de transaction (défaut : `<barman_home>/<nom_section_instance>`) ;
- **`basebackup_directory`** : emplacement du dossier contenant les sauvegardes (défaut : `<barman_home>/<nom_section_instance>/base`) ;
- **`wals_directory`** : emplacement du dossier contenant les journaux de transactions archivés (défaut : `<barman_home>/<nom_section_instance>/wals`) ;
- **`incoming_wals_directory`** : emplacement du dossier contenant les journaux de transactions en attente d'archivage (défaut : `<barman_home>/<nom_section_instance>/incoming`).

Tous les autres paramètres, à l'exception de `log_file` et `log_level`, peuvent être redéfinis pour chaque instance.

4.20 Barman - Vérification de la configuration

- La commande show-server montre la configuration :

```
$ sudo -u barman barman show-server {<instance> | all}
```

- La commande check effectue des tests pour la valider :

```
$ sudo -u barman barman check {<instance> | all}
```

La commande show-server permet de visualiser la configuration de Barman pour l'instance spécifiée, ou pour toutes les instances si le mot clé all est utilisé. C'est particulièrement utile lors d'un premier paramétrage, notamment pour récupérer la valeur assignée au paramètre incoming_wals_directory afin de configurer correctement la valeur du paramètre archive_command de l'instance à sauvegarder.

La commande check vérifie le bon paramétrage de Barman pour l'instance spécifiée, ou pour toutes les instances si le mot clé all est utilisé. Elle permet de s'assurer que les points clés sont fonctionnels, tels que l'accès SSH, l'archivage des journaux de transaction (archive_command, archive_mode...), la politique de rétention, la compression, etc. Il est possible d'utiliser l'option --nagios qui permet de formater la sortie de la commande check et de l'utiliser en tant que sonde Nagios.

Exemple de sortie de la commande show-server :

```
-bash-4.2$ barman show-server pgsrv
Server pgsrv:
        active: True
        archive_command: rsync %p
barman@bkpsrv:/var/lib/barman/pgsrv/incoming/%f
        archive_mode: on
        archived_count: 88
        backup_directory: /var/lib/barman/pgsrv
        backup_options: BackupOptions(['exclusive_backup'])
        bandwidth_limit: None
        basebackup_retry_sleep: 30
        basebackup_retry_times: 0
        basebackups_directory: /var/lib/barman/pgsrv/base
        compression: None
        config_file: /var/lib/pgsql/9.4/data/postgresql.conf
        conninfo: host=pgsrv user=postgres
        copy_method: rsync
        current_archived_wals_per_second: 0.00250231660903
        current_xlog: 0000000A0000000100000063
        custom_compression_filter: None
        custom_decompression_filter: None
```

```
        data_directory: /var/lib/pgsql/9.4/data
        description: PostgreSQL Instance pgsrv
        disabled: False
        failed_count: 16
        hba_file: /var/lib/pgsql/9.4/data/pg_hba.conf
        ident_file: /var/lib/pgsql/9.4/data/pg_ident.conf
        immediate_checkpoint: False
        incoming_wals_directory: /var/lib/barman/pgsrv/incoming
        is_archiving: True
        last_archived_time: 2015-10-29 19:31:25.152625+01:00
        last_archived_wal: 0000000A0000000100000062
        last_backup_maximum_age: None
        last_failed_time: 2015-10-29 19:30:58.923143+01:00
        last_failed_wal: 0000000A0000000100000060
        minimum_redundancy: 1
        network_compression: False
        pgespresso_installed: False
        post_archive_retry_script: None
        post_archive_script: None
        post_backup_retry_script: None
        post_backup_script: None
        pre_archive_retry_script: None
        pre_archive_script: None
        pre_backup_retry_script: None
        pre_backup_script: None
        recovery_options: RecoveryOptions([])
        retention_policy: REDUNDANCY 2
        retention_policy_mode: auto
        reuse_backup: None
        server_txt_version: 9.4.5
        ssh_command: ssh postgres@pgsrv
        stats_reset: 2015-10-29 09:46:00.034650+01:00
        tablespace_bandwidth_limit: None
        wal_level: hot_standby
        wal_retention_policy: MAIN
        wals_directory: /var/lib/barman/pgsrv/wals
```

Exemple de sortie de la commande check :

```
-bash-4.2$ barman check pgsrv
Server pgsrv:
        PostgreSQL: OK
        archive_mode: OK
        wal_level: OK
        archive_command: OK
        continuous archiving: OK
        directories: OK
        retention policy settings: OK
        backup maximum age: OK (no last_backup_maximum_age provided)
        compression settings: OK
        minimum redundancy requirements: OK (have 1 backups, expected at
least 1)
        ssh: OK (PostgreSQL server)
        not in recovery: OK
```

4.21 Barman - Statut

- La commande `status` affiche des informations détaillées
  - sur la configuration Barman
  - sur l'instance spécifiée

```
$ sudo -u barman barman status {<instance> | all}
```

La commande `status` retourne de manière détaillée le statut de l'instance spécifiée, ou de toutes si le mot-clé `all` est utilisé. Les informations renvoyées sont, entre autres :

- la description extraite du fichier de configuration de Barman ;
- la version de PostgreSQL ;
- si l'extension `pgespresso` est utilisée ;
- l'emplacement des données sur l'instance (`PGDATA`) ;
- la valeur de l'''archive_command'' ;
- des informations sur les journaux de transactions :
  - position courante
  - dernier segment archivé
- des informations sur les sauvegardes :
  - nombre de sauvegarde
  - ID de la première sauvegarde,
  - ID de la dernière sauvegarde,
  - politique de rétention.

Exemple de sortie de la commande :

```
-bash-4.2$ barman status pgsrv
Server pgsrv:
        Description: PostgreSQL Instance pgsrv
        Active: True
        Disabled: False
        PostgreSQL version: 9.4.5
        pgespresso extension: Not available
        PostgreSQL Data directory: /var/lib/pgsql/9.4/data
        PostgreSQL 'archive_command' setting: rsync %p
barman@bkpsrv:/var/lib/barman/pgsrv/incoming/%f
        Last archived WAL: 0000000A0000000100000062, at Thu Oct 29 19:31:25
2015
        Current WAL segment: 0000000A0000000100000063
```

```
        Failures of WAL archiver: 16 (0000000A0000000100000060 at Thu Oct 29
19:30:58 2015)
        Server WAL archiving rate: 8.98/hour
        Retention policies: enforced (mode: auto, retention: REDUNDANCY 2,
WAL retention: MAIN)
        No. of available backups: 1
        First available backup: 20151029T192840
        Last available backup: 20151029T192840
        Minimum redundancy requirements: satisfied (1/1)
```

4.22 Barman - Diagnostiquer

- La commande diagnose renvoie
  - les informations renvoyées par la commande status
  - des informations supplémentaires (sur le système par exemple)
  - au format json

```
$ sudo -u barman barman diagnose
```

La commande diagnose retourne les informations importantes concernant toutes les instances à sauvegarder, en donnant par exemple les versions de chacun des composants utilisés.

Elle reprend également les informations retournées par la commande status, le tout au format JSON.

4.23 Barman - Nouvelle sauvegarde

- Pour déclencher une nouvelle sauvegarde :

```
$ sudo -u barman barman backup {<instance> | all}
```

- Le détail de sauvegarde effectuée est affiché en sortie

La commande backup lance immédiatement une nouvelle sauvegarde, pour une seule instance si un identifiant est passé en argument, ou pour toutes les instances configurées si le mot clé all est utilisé.

Exemple de sortie de la commande :

```
-bash-4.2$ barman backup pgsrv
Starting backup for server pgsrv in
/var/lib/barman/pgsrv/base/20151029T193737
Backup start at xlog location: 1/64000028 (0000000A0000000100000064,
00000028)
Copying files.
Copy done.
Asking PostgreSQL server to finalize the backup.
Backup size: 203.0 MiB
Backup end at xlog location: 1/64000128 (0000000A0000000100000064, 00000128)
Backup completed
Processing xlog segments for pgsrv
        0000000A0000000100000063
        0000000A0000000100000064
        0000000A0000000100000064.00000028.backup
```

4.24 Barman - Lister les sauvegardes

- Pour lister les sauvegardes existantes :

```
$ sudo -u barman barman list-backup {<instance> | all}
```

- Affiche notamment la taille de la sauvegarde et des WAL associés

Liste les sauvegardes du catalogue, soit par instance, soit toutes si le mot clé `all` est passé en argument.

Exemple de sortie de la commande :

```
-bash-4.2$ barman list-backup pgsrv
pgsrv 20151029T193737 - Thu Oct 29 19:37:45 2015 - Size: 203.0 MiB - WAL
Size: 0 B (tablespaces: tmptbs:/tmp/tmptbs)
pgsrv 20151029T192840 - Thu Oct 29 19:28:48 2015 - Size: 203.0 MiB - WAL
Size: 48.0 MiB (tablespaces: tmptbs:/tmp/tmptbs)
```

4.25 Barman - Détail d'une sauvegarde

- show-backup affiche le détail d'une sauvegarde (taille...) :

```
$ sudo -u barman barman show-backup <instance> <ID-sauvegarde>
```

- `list-files` affiche le détail des fichiers d'une sauvegarde :

```
$ sudo -u barman barman list-files <instance> <ID-sauvegarde>
```

La commande show-backup affiche toutes les informations relatives à une sauvegarde en particulier, comme l'espace disque occupé, le nombre de journaux de transactions associés, etc.

La commande `list-files` permet quant à elle d'afficher la liste complète des fichiers contenus dans la sauvegarde.

Exemple de sortie de la commande show-backup :

```
-bash-4.2$ barman show-backup pgsrv 20151029T192840
Backup 20151029T192840:
  Server Name            : pgsrv
  Status                 : DONE
  PostgreSQL Version     : 90405
  PGDATA directory       : /var/lib/pgsql/9.4/data
  Tablespaces:
    tmptbs: /tmp/tmptbs (oid: 24583)

  Base backup information:
    Disk usage           : 203.0 MiB (219.0 MiB with WALs)
    Incremental size     : 203.0 MiB (-0.00%)
    Timeline             : 10
    Begin WAL            : 0000000A0000000100000061
    End WAL              : 0000000A0000000100000061
    WAL number           : 1
    Begin time           : 2015-10-29 19:28:40.409406+01:00
    End time             : 2015-10-29 19:28:48.044205+01:00
    Begin Offset         : 40
    End Offset           : 240
    Begin XLOG           : 1/61000028
    End XLOG             : 1/610000F0

  WAL information:
    No of files          : 3
    Disk usage           : 48.0 MiB
    WAL rate             : 37.67/hour
    Last available       : 0000000A0000000100000064

  Catalog information:
    Retention Policy     : VALID
```

```
    Previous Backup      : - (this is the oldest base backup)
    Next Backup          : 20151029T193737
```

4.26 Barman - Suppression d'une sauvegarde

- Pour supprimer manuellement une sauvegarde :

```
$ sudo -u barman barman delete <instance> <ID-sauvegarde>
```

- Renvoie une erreur si la redondance minimale ne le permet pas

La supression d'une sauvegarde nécessite de spécifier l'instance ciblée et l'identifiant de la sauvegarde à supprimer. Cet identifiant peut-être trouvé en utilisant la commande Barman `list-backup`.

Si le nombre de sauvegardes (après suppression) ne devait pas respecter le seuil défini par la directive `minimum_redundancy`, la suppression ne sera alors pas possible.

4.27 Barman - Tâches de maintenance

- La commande Barman `cron` déclenche la maintenance :
  - récupération des WAL archivés
  - compression
  - politique de rétention

```
$ sudo -u barman barman cron
```

- À planifier pour une exécution régulière
  - par exemple avec la `crontab` Linux

La commande `cron` permet d'exécuter les tâches de maintenance qui doivent être exécutées périodiquement, telles que l'archivage des journaux de transaction (déplacement du dossier `incoming_wals/` vers `wals/`), ou la compression. L'application de la politique de rétention est également faite dans ce cadre.

L'exécution de cette commande doit donc être planifiée via un cronjob, par

exemple toutes les minutes.

4.28 Barman - Restauration

Processus de restauration :

1. Copie/transfert de la sauvegarde
2. Copie/transfert des journaux de transaction
3. Génération du fichier `recovery.conf`
4. Copie/transfert des fichiers de configuration

Le processus de restauration géré par Barman reste classique, mais nécessite tout de même quelques points d'attention.

En particulier, les fichiers de configuration sauvegardés sont restaurés dans le dossier `$PGDATA`, or ce n'est potentiellement pas le bon emplacement selon le type d'installation / configuration de l'instance. Dans une installation basée sur les paquets Debian/Ubuntu par exemple, les fichiers de configuration se trouvent dans `/etc/postgresql/<version>/<instance>` et non dans le répertoire PGDATA. Il convient donc de penser à les supprimer du PGDATA s'ils n'ont rien à y faire avant de démarrer l'instance.

De même, la directive de configuration `archive_command` est passée à `false` par Barman. Une fois l'instance démarrée et fonctionnelle, il convient de modifier la valeur de ce paramètre pour réactiver l'archivage des journaux de transaction.

4.29 Barman - Options de restauration

- Locale ou à distance
- Cibles : timeline, date, ID de transaction ou point de restauration
- Déplacement des tablespaces

Au niveau de la restauration, Barman offre la possibilité de restaurer soit en local (sur le serveur où se trouvent les sauvegardes), ou à distance. Le cas le plus commun est une restauration à distance, car les sauvegardes sont généralement centralisées sur le serveur de sauvegarde d'où Barman est exécuté. Pour la restauration à distance, Barman s'appuie sur la couche SSH pour le transfert des données.

Barman supporte différents types de cibles dans le temps pour la restauration :

- **`timeline`** : via l'option `--target-tli`, lorsqu'une divergence de timeline a eu lieu, il est possible de restaurer et rejouer toutes les transactions d'une timeline particulière ;
- **`date`** : via l'option `--target-time` au format **YYYY-MM-DD HH:MM:SS.mmm**, spécifie une date limite précise dans le temps au delà de laquelle la procédure de restauration arrête de rejouer les transactions ;
- **`identifiant de transaction`** : via l'option `--target-xid`, restauration jusqu'à une transaction précise ;
- **`point de restauration`** : via l'option `--target-name`, restauration jusqu'à un point de restauration créé préalablement sur l'instance via l'appel à la fonction `pg_create_restore_point(nom)`.

Barman permet également de relocaliser un tablespace lors de la restauration. Ceci est utile lorsque l'on souhaite restaurer une sauvegarde sur un serveur différent, ne disposant pas des même points de montage des volumes que l'instance originelle.

4.30 Barman - Exemple de restauration à distance

- Exemple d'une restauration
  - déclenchée depuis le serveur Barman
  - avec un point dans le temps spécifié

```
$ sudo -u barman barman recover --remote-ssh-command "ssh
postgres@pgsrv" \
      --target-time "2015-09-02 14:15:00" \
      pgsrv 20150902T095027 /var/lib/pgsql/9.4/main
```

Dans cet exemple, nous souhaitons effectuer une restauration à distance via l'option `--remote-ssh-command`, prenant en argument “`ssh postgres@pgsrv`” correspondant à la commande SSH pour se connecter au serveur à restaurer.

L'option `--target-time` définit ici le point de restauration dans le temps comme étant la date “2015-09-02 14:15:00”.

Les 3 derniers arguments sont :

- l'identifiant de l'instance dans le fichier de configuration de Barman : `pgsrv` ;
- l'identifiant de la sauvegarde cible : 20150902T095027 ;
- et enfin le dossier PGDATA de l'instance à restaurer.

5 pitrery

5.1 pitrery - Présentation générale

- R&D Dalibo
- Langage : **bash**
- OS : **Unix/Linux**
- Versions compatibles : **>= 8.2**
- License : **BSD** (libre)
- Type d'interface : **CLI** (ligne de commande)

pitrery est un outil de gestion de sauvegarde physique et restauration PITR, écrit en bash, issu du labo R&D de Dalibo. Il est compatible avec tous les environnements Unix/Linux disposant de l'interpréteur de shell bash, et supporte toutes les version de PostgreSQL depuis la 8.2.

5.2 pitrery - Diagrame


5.3 pitrery - Sauvegardes

- Type de sauvegarde : **physiques/PITR**
- Type de stockage : **locale** ou **distant** (push) via `rsync/ssh`
- Planification : **crontab**
- Méthodes : **pg_start_backup() / rsync ou tar / pg_stop_backup()**
- Compression : **gzip**, **bzip2**, **pigz**, **pbzip2**, etc.
- Compression des WAL
- Scripts pre/post sauvegarde (hooks)
- Déduplication de fichier (`rsync` + lien matériel)

pitrery permet de gérer exclusivement des sauvegardes physiques. Il s'installe sur le même serveur que l'instance à sauvegarder, et est capable de stocker les sauvegardes sur un point de montage local ou de les pousser au travers d'une connexion SSH vers un serveur de sauvegarde tiers.

La méthode de sauvegarde interne utilisée se base sur les appels de fonction SQL `pg_start_backup()` et `pg_stop_backup()`, les données étant effectivement copiées entre ces deux appels. L'archivage des journaux de transactions doit être activé, et peut se faire en appelant un script interne à `pitrery`. Les sauvegardes et les journaux de transactions peuvent être compressés, avec possibilité de spécifier l'outil de compression à utiliser. pitrery propose également deux hooks (points d'ancrage) dans le code, offrant ainsi la possibilité d'exécuter des scripts externes avant ou après l'exécution d'une nouvelle sauvegarde.

Concernant le stockage des sauvegardes, il est possible d'activer la déduplication des fichiers entre deux sauvegardes. Ceci est possible en utilisant `rsync` comme format de sauvegarde - dans ce cas pitrery va tirer parti de la capacité de `rsync` à effectuer une sauvegarde différentielle combinée avec la création de liens matériels (hardlink), ce qui permet selon les cas d'économiser de l'espace disque et du temps de sauvegarde de manière conséquente.

La compression de la sauvegarde est également possible si l'on a choisi `tar` comme format de sauvegarde.

5.4 pitrery - Politique de rétention

- **Durée** (en jours)
- **Nombre** de sauvegardes

La politique de rétention des sauvegardes se définit en jours ou en nombre de sauvegardes.

La combinaison des deux paramétrages est possible. Dans ce cas, pitrery déclenchera la suppression d'une sauvegarde **si et seulement si** les deux rétentions sont dépassées.

Ainsi, si la durée spécifiée est de sept jours, et le nombre de sauvegardes est de trois, alors au moment de supprimer une sauvegarde antérieure à sept jours pitrery vérifiera qu'il reste bien au moins trois sauvegardes disponibles.

5.5 pitrery - Restauration

- À partir des données locales ou distantes
- Point dans le temps : date

Comme dans le cadre de la sauvegarde, lors de la restauration, pitrery est capable de transférer les données depuis un serveur tiers. Il est également possible de spécifier un point dans le temps pour effectuer une restauration de type PITR.

5.6 pitrery - Installation

- téléchargement depuis le site du projet
  - installation depuis les sources (tarball)
  - paquets RPM et DEB disponibles

L'installation est très simple.

Pour installer depuis les sources, il faut d'abord télécharger la dernière version stable depuis le site du projet (https://dalibo.github.io/pitrery/downloads.html), désarchiver le fichier, puis installer `pitrery` via la commande `make install`, à exécuter avec les droits root. Par défaut, les scripts pitrery sont installés dans le dossier `/usr/local/bin`, le fichier de configuration se trouve dans le dossier

`/usr/local/etc/pitrery`.

Exemple détaillé :

```
wget https://github.com/dalibo/pitrery/archive/v1.10.tar.gz
tar xvzf v1.10.tar.gz
cd pitrery-1.10
sudo make install
```

Des paquets RPM et DEB sont également disponibles dans la page de téléchargement.

5.7 pitrery - Utilisation

```
usage: pitrery [options] action [args]
options:
    -c file       Path to the configuration file
    -n            Show the command instead of executing it
    -V            Display the version and exit
    -?            Print help

actions:
    list
    backup
    restore
    purge
```

L'option `-c` permet de définir l'emplacement du fichier de configuration (par défaut, celui-ci se trouve ici sous `/usr/local/etc/pitrery/pitr.conf`). Le script `pitrery` étant un wrapper appelant d'autres scripts en fonction de l'action demandée, l'option `-n` permet d'afficher sur la sortie standard sans l'exécuter l'appel au script devant effectuer l'action. L'option `-V` affiche la version de l'outil et se termine. L'option `-?` permet d'afficher le message d'aide, et peut être combinée avec une commande pour afficher l'aide spécifique à la commande (par exemple, `pitrery restore -?`).

Les actions possibles sont les suivantes :

- **`list`** : pour lister le contenu du catalogue de sauvegarde ;
- **`backup`** : effectuer une sauvegarde ;
- **`restore`** : effectuer une restauration ;
- **`purge`** : supprimer les sauvegardes et journaux de transaction ne satisfaisant plus la politique de rétention.

5.8 pitrery - Configuration PostgreSQL

- Adapter l'archivage dans le fichier `postgresql.conf` :

```
archive_mode = on
wal_level = 'hot_standby'
archive_command = '/usr/local/bin/archive_xlog %p'
```

Comme pour Barman, il est nécessaire d'activer l'archivage des journaux de transactions en positionnant le paramètre `archive_mode` à on et en définissant un niveau d'enregistrement d'informations dans les journaux de transactions (`wal_level`) supérieur ou égal à `archive`.

pitrery fourni un script permettant de simplifier la configuration de la commande d'archivage. Pour l'utiliser, il faut configurer le paramètre `archive_command` pour qu'il appelle le script `archive_xlog`, suivi de l'argument %p qui correspond au dossier contenant les journaux de transactions de l'instance.

5.9 pitrery - Configuration SSH

- Stockage des sauvegardes sur un serveur distant
- Générer les clefs SSH (RSA) des utilisateurs système `postgres` sur chacun des serveurs (nœud PG et serveur de sauvegarde)
- Échanger les clefs SSH publiques entre les serveurs PostgreSQL et le serveur de sauvegarde
- Établir manuellement une première connexion SSH entre chaque serveur

Dans le cadre de la mise en place de sauvegardes avec un stockage des données sur un serveur tiers, la plupart des outils et méthodes historiques de sauvegardes se reposent sur le protocole SSH et les outils tels que `rsync` pour assurer les transferts au travers du réseau.

Afin d'automatiser ces transferts reposant sur le protocole SSH, il est impératif d'autoriser l'authentification SSH par clé et d'échanger les clés publiques des différents serveurs.

5.10 pitrery - Fichier de configuration

- emplacement par défaut: `/usr/local/etc/pitrery/pitr.conf`
- possibilité de spécifier un autre emplacement

Il est possible de spécifier un fichier de configuration spécifique à utiliser à l'appel d'un script. L'option à utiliser varie en fonction du script utilisé:

- `-c <fichier>` pour le script principal `pitrery` ;
- `-C <fichier>` pour les autres scripts.

5.11 pitrery - Configuration connexion PostgreSQL

- Options de connexion à l'instance :

```
PGPSQL="psql"
PGUSER="postgres"
PGPORT=5432
PGHOST="/var/run/postgresql"
PGDATABASE="postgres"
```

Ces paramètres sont utilisés par pitrery pour établir une connexion à l'instance afin d'exécuter les fonctions `pg_start_backup()` et `pg_stop_backup()` utilisées pour signaler à PostgreSQL qu'une sauvegarde PITR démarre et se termine.

- **`PGPSQL`** : chemin vers binaire du client `psql` ;
- **`PGUSER`** : super utilisateur PostgreSQL de connexion ;
- **`PGPORT`** : numéro de port d'écoute PostgreSQL ;
- **`PGHOST`** : hôte PostgreSQL (dossier contenant le socket UNIX de connexion ou l'adresse IP) ;
- **`PGDATABASE`** : nom de la base de données de connexion.

5.12 pitrery - Localisation

- Configurer les emplacements
- pour l'instance à sauvegarder :

```
PGDATA="/var/lib/pgsql/9.4/data"
PGXLOG=
```

- pour la destination des sauvegardes :

```
BACKUP_DIR="/var/lib/pitrery"
```

- pour la destination des WAL archivés :

```
ARCHIVE_DIR="$BACKUP_DIR/archived_xlog"
```

- **PGDATA** : répertoire contenant les données de l'instance PostgreSQL à sauvegarder ;
- **PGXLOG** : répertoire contenant les journaux de transaction PostgreSQL, laisser vide s'il se trouve à la racine du PGDATA ;
- **BACKUP_DIR** : répertoire destiné à contenir l'arborescence des sauvegardes PITR ;
- **ARCHIVE_DIR** : répertoire destiné à contenir les journaux de transaction archivés.

Il est possible d'utiliser des paramètres initialisés précédemment comme variables lors de l'initialisation des paramètres suivants.

5.13 pitrery - Configuration du mode de sauvegarde

- Paramètre de configuration STORAGE
- Deux modes possibles :
  - tar : sauvegarde complète, éventuellement compressée
  - rsync : sauvegarde complète ou différentielle, pas de compression

pitrery permet de spécifier deux modes de sauvegarde.

Le premier, `tar`, réalise la sauvegarde sous la forme d'une archive `tar` du répertoire PGDATA, éventuellement compressée, plus autant d'archives `tar` qu'il y a de tablespaces dans l'instance. Ce mode ne permet que les sauvegardes complètes.

Le second, `rsync`, indique à pitrery de copier les fichiers de l'instance à l'aide de la commande système idoine. Les fichiers dans le répertoire de la sauvegarde seront donc exactement les mêmes que ceux de l'instance :

- le contenu du répertoire PGDATA de l'instance est dans un sous-répertoire `pgdata` ;
- le contenu des différents tablespaces est dans un sous-répertoire `tblspc/<nom du tblspc>`.

Ce mode ne permet pas la compression, il consommera donc par défaut beaucoup plus de place que la sauvegarde `tar`. En revanche, ce mode permet la réalisation de sauvegarde différentielles, en réalisant des liens matériels (hardlink) des fichiers depuis une précédente sauvegarde et en utilisant la fonctionnalité de `rsync` basée sur les checksum pour ne sauvegarder que les fichiers ayant été modifiés. Si l'instance a eu peu de modifications depuis la précédente sauvegarde, ce mode peut donc faire économiser du temps de sauvegarde et de l'espace.

5.14 pitrery - Configuration de la rétention

- Configuration de la rétention en nombre de sauvegardes

```
PURGE_KEEP_COUNT=3
```

- Ou en jours

```
PURGE_OLDER_THAN=7
```

- Les deux paramètres peuvent être combinés

Le paramétrage de la rétention peut s'effectuer à deux niveaux :

- **`PURGE_KEEP_COUNT`** : nombre minimal de sauvegarde à conserver ;
- **`PURGE_OLDER_THAN`** : âge minimal des sauvegardes à conserver, exprimé en jours.

Si les deux paramètres sont utilisés de concert, une sauvegarde ne sera supprimée que si elle ne répond à aucune des deux rétentions. Ainsi, avec la configuration faite ici, une sauvegarde datant de plus de sept jours ne sera supprimée que s'il reste effectivement au moins trois autres sauvegardes.

5.15 pitrery - Configuration de l'archivage

- L'archivage peut être configuré pour :
  - archiver en local (montage NFS...)
  - archiver vers un serveur distant en SSH

```
ARCHIVE_LOCAL="no"
ARCHIVE_HOST=bkpsrv
ARCHIVE_USER=pitrery
ARCHIVE_COMPRESS="yes"
```

- Utilisé par la commande `archive_xlog`

Ce paramétrage est utilisé lors de l'appel au script `archive_xlog` par l'instance PostgreSQL :

- **`ARCHIVE_LOCAL`** : yes ou no, définit si les journaux sont archivés localement ou sur un serveur de sauvegarde distant (rsync/ssh) ;
- **`ARCHIVE_HOST`** : hôte stockant les journaux de transaction archivés, si `ARCHIVE_LOCAL = no` ;
- **`ARCHIVE_USER`** : utilisateur SSH de connexion vers le serveur distant pour le transfert des WAL, si `ARCHIVE_LOCAL = no` ;
- **`ARCHIVE_COMPRESS`** : compression des journaux de transaction lors de l'archivage.

5.16 pitrery - Configuration de la compression

- Configuration de la compression
  - des WAL archivés par la commande `archive_xlog`
  - des sauvegardes effectuées au format `tar`

```
COMPRESS_BIN=
COMPRESS_SUFFIX=
UNCOMPRESS_BIN=
BACKUP_COMPRESS_BIN=
BACKUP_COMPRESS_SUFFIX=
BACKUP_UNCOMPRESS_BIN=
```

Les paramètres spécifiés ici permettent de spécifier des commandes spécifiques pour compresser / décompresser les journaux de transactions

archivés et les sauvegardes :

- **COMPRESS_BIN** : chemin vers le binaire (+ ses options) utilisé pour la compression des fichiers WAL archivés (`gzip`, `bzip2` etc), `gzip -4` par défaut ;
- **COMPRESS_SUFFIX** : l'extension du fichier contenant les fichiers WAL archivés compressés, `gz` par défaut, utilisé pour la décompression ;
- **UNCOMPRESS_BIN** : chemin vers l'outil utilisé pour décompresser les fichiers WAL archivés, `gunzip` par défaut ;
- **BACKUP_COMPRESS_BIN** : chemin vers le binaire (+ ses options) utilisé pour la compression des sauvegardes (`gzip`, `bzip2` etc), `gzip -4` par défaut ;
- **BACKUP_COMPRESS_SUFFIX**: l'extension du fichier compressé de la sauvegarde ;
- **BACKUP_UNCOMPRESS_BIN**: chemin vers l'outil utilisé pour décompresser les sauvegardes, `gunzip` par défaut.

Il est possible d'utiliser des outils de compression multithread/multiprocess comme `pigz` ou `pbzip2`.

À noter que la compression de la sauvegarde n'est possible que si la méthode de stockage `tar` est utilisée.

5.17 pitrery - Configuration des traces

- Activer l'horodatage des traces :

```
LOG_TIMESTAMP="yes"
```

- Utiliser syslog pour les traces de l'archivage (`archive_xlog`)

```
SYSLOG="no"
SYSLOG_FACILITY="local0"
SYSLOG_IDENT="postgres"
```

Ces paramètres permettent d'activer la redirection des traces vers `syslog` pour les opérations d'archivage et de restauration des journaux de transaction. Les autres opérations sont écrites vers les sorties standards.

5.18 pitrery - Configuration de hooks

- Exécuter un script avant ou après une sauvegarde :

```
PRE_BACKUP_COMMAND=
POST_BACKUP_COMMAND=
```

Ces paramètres permettent de spécifier des commandes à exécuter avant ou après une sauvegarde :

- **PRE_BACKUP_COMMAND** : commande exécutée avant le début de la sauvegarde ;
- **POST_BACKUP_COMMAND** : commande exécutée après la fin de la sauvegarde.

5.19 pitrery - Effectuer une sauvegarde

- Pour déclencher une nouvelle sauvegarde :

```
$ sudo -u postgres /usr/local/bin/pitrery backup
```

- la plupart des paramètre peuvent être surchargés
- par exemple, l'option `-s` permet de spécifier un mode, `tar` ou `rsync`

L'exécution de la commande pitrery backup déclenche la création immédiate d'une nouvelle sauvegarde de l'instance. Toutes les commandes `pitrery` doivent être exécutées depuis un utilisateur système ayant un accès en lecture aux fichiers de données de l'instance PostgreSQL. En général on utilisera le compte de service PostgreSQL, par défaut `postgres`.

Exemple de sortie de la commande pour une sauvegarde `rsync` :

```
-bash-4.2$ pitrery backup
INFO: preparing directories in bkpsrv:/var/lib/pitrery/backups/pitr
INFO: listing tablespaces
INFO: starting the backup process
INFO: backing up PGDATA with rsync
INFO: preparing hardlinks from previous backup
INFO: transfering data from /var/lib/pgsql/9.4/data
INFO: backing up tablespace "tmptbs" with rsync
INFO: preparing hardlinks from previous backup
```

```
INFO: transfering data from /tmp/tmptbs
INFO: stopping the backup process
NOTICE:  pg_stop_backup complete, all required WAL segments have been
archived
INFO: copying the backup history file
INFO: copying the tablespaces list
INFO: backup directory is
bkpsrv:/var/lib/pitrery/backups/pitr/2015.10.29_22.08.11
INFO: done
```

On voit à la ligne suivante qu'il s'agit d'une sauvegarde différentielle, seuls les fichiers ayant été modifiés depuis la sauvegarde précédente ont réellement été transférés :

```
INFO: preparing hardlinks from previous backup
```

5.20 pitrery - Suppression des sauvegardes obsolètes

- Suppression des sauvegardes et archives ne satisfaisant plus la politique de sauvegarde :

```
$ sudo -u postgres /usr/local/bin/pitrery purge
```

- À déclencher systématiquement après une sauvegarde

La commande purge se charge d'appliquer la politique de rétention et de supprimer les sauvegardes les plus anciennes, ainsi que les journaux de transactions devenus obsolètes.

5.21 pitrery - Lister les sauvegardes

- Lister les sauvegardes présentes et leur taille :

```
$ sudo -u postgres /usr/local/bin/pitrery list
```

- l'option -v permet d'avoir plus de détail
  - mode de sauvegarde
  - date minimale utilisable pour la restauration
  - taille de chaque tablespace

La commande pitrery `list` énumère les sauvegardes présentes, indiquant l'emplacement, la taille, la date et l'heure de la création de chacune d'elles.

L'option -v permet d'afficher des informations plus détaillées, notamment la date minimale utilisable pour une restauration effectuée à partir de cette sauvegarde, ce qui est très pratique pour savoir précisément jusqu'à quel point dans le temps il est possible de remonter en utilisant les sauvegardes présentes.

Exemple de sortie de la commande :

```
-bash-4.2$ pitrery list
List of backups on bkpsrv
/var/lib/pitrery/backups/pitr/2015.10.29_22.09.52       36M       2015-10-29
22:09:52 CET
/var/lib/pitrery/backups/pitr/2015.10.29_22.17.15       240M      2015-10-29
22:17:15 CET
/var/lib/pitrery/backups/pitr/2015.10.29_22.28.48       240M      2015-10-29
22:28:48 CET
```

Avec l'option -v :

```
-bash-4.2$ pitrery list -v
List of backups on bkpsrv
--------------------------------------------------------------------------
Directory:
  /var/lib/pitrery/backups/pitr/2015.10.29_22.09.52
  space used: 36M
  storage: tar with gz compression
Minimum recovery target time:
  2015-10-29 22:09:52 CET
PGDATA:
  pg_default 202 MB
  pg_global 469 kB
Tablespaces:
```

```
  "tmptbs" /tmp/tmptbs (24583) 36 MB

------------------------------------------------------------------------
Directory:
  /var/lib/pitrery/backups/pitr/2015.10.29_22.17.15
  space used: 240M
  storage: rsync
Minimum recovery target time:
  2015-10-29 22:17:15 CET
PGDATA:
  pg_default 202 MB
  pg_global 469 kB
Tablespaces:
  "tmptbs" /tmp/tmptbs (24583) 36 MB

------------------------------------------------------------------------
Directory:
  /var/lib/pitrery/backups/pitr/2015.10.29_22.28.48
  space used: 240M
  storage: rsync
Minimum recovery target time:
  2015-10-29 22:28:48 CET
PGDATA:
  pg_default 202 MB
  pg_global 469 kB
Tablespaces:
  "tmptbs" /tmp/tmptbs (24583) 36 MB
```

5.22 pitrery - Planification

- Pas de planificateur intégré
  - le plus simple est d'utiliser `cron`
- Exemple :

```
00 00 * * * (/usr/local/bin/pitrery backup &&
/usr/local/bin/pitrery purge) \
    >> /var/log/postgresql/pitrery-$(date +\%Y-\%m-\%d).log 2>&1
```

La planification des sauvegardes et de la suppression des sauvegardes ne respectant plus la politique de rétention peut être faite par n'importe quel outil de planification de tâches, le plus connu étant `cron`.

L'exemple montre la planification d'une sauvegarde suivie de la suppression des sauvegardes obsolètes, via la crontab de l'utilisateur système `postgres`. La sauvegarde est effectuée tous les jours à minuit, et est suivie d'une purge seulement si elle a réussi. Les traces (stdout et stderr) sont redirigées vers un

fichier journalisé par jour.

5.23 pitrery - Restauration

- Effectuer une restauration :

```
$ sudo -u postgres /usr/local/bin/pitrery restore
```

- Nombreuses options à la restauration, notamment :
  - l'option `-D` permet de modifier la cible du PGDATA
  - l'option `-t` permet de modifier la cible d'un tablespace
  - l'option `-x` permet de modifier la cible du `pg_xlog`
  - l'option `-d` permet de spécifier une date de restauration

La restauration d'une sauvegarde se fait par l'appel à la commande pitrery `restore`. Plusieurs options peuvent être spécifiées, offrant notamment la possibilité de restaurer le PGDATA (option `-D`), les tablespaces (option `-t`) et le pg_xlog (option `-x`) dans des répertoires spécifiques. Il est également possible de restaurer à une date précise (option `-d`), à condition que la date spécifiée soit postérieure à la date minimale utilisable de la plus ancienne sauvegarde disponible.

Voici un exemple combinant ces différentes options :

```
-bash-4.2$ pitrery restore -d '2015-10-29 22:10:00' -D
/var/lib/pgsql/9.4/data2/ -x /var/lib/pgsql/9.4/pg_xlog2/ -t
tmptbs:/tmp/tmptbs2/
INFO: searching backup directory
INFO: searching for tablespaces information
INFO:
INFO: backup directory:
INFO:   /var/lib/pitrery/backups/pitr/2015.10.29_22.28.48
INFO:
INFO: destinations directories:
INFO:   PGDATA -> /var/lib/pgsql/9.4/data2/
INFO:   PGDATA/pg_xlog -> /var/lib/pgsql/9.4/pg_xlog2/
INFO:   tablespace "tmptbs" (24583) -> /tmp/tmptbs2/ (relocated: yes)
INFO:
INFO: recovery configuration:
INFO:   target owner of the restored files: postgres
INFO:   restore_command = '/usr/bin/restore_xlog -h bkpsrv -u pitrery -d
/var/lib/pitrery/archived_xlog %f %p'
INFO:   recovery_target_time = '2015-10-29 22:10:00'
INFO:
INFO: checking if /var/lib/pgsql/9.4/data2/ is empty
INFO: setting permissions of /var/lib/pgsql/9.4/data2/
INFO: checking if /var/lib/pgsql/9.4/pg_xlog2/ is empty
```

```
INFO: setting permissions of /var/lib/pgsql/9.4/pg_xlog2/
INFO: checking if /tmp/tmptbs2/ is empty
INFO: setting permissions of /tmp/tmptbs2/
INFO: transfering PGDATA to /var/lib/pgsql/9.4/data2/ with rsync
INFO: transfer of PGDATA successful
INFO: transfering PGDATA to /var/lib/pgsql/9.4/data2/ with rsync
INFO: transfer of tablespace "tmptbs" successful
INFO: creating symbolic link pg_xlog to /var/lib/pgsql/9.4/pg_xlog2/
INFO: preparing pg_xlog directory
INFO: preparing recovery.conf file
INFO: done
INFO:
INFO: please check directories and recovery.conf before starting the cluster
INFO: and do not forget to update the configuration of pitrery if needed
INFO:
```

Par cette commande, nous demandons à pitrery de :

- relocaliser le PGDATA dans `/var/lib/pgsql/9.4/data2/` ;
- relocaliser le pg_xlog dans `/var/lib/pgsql/9.4/pg_xlog2/` ;
- relocaliser le tablespace `tmptbs` dans `/tmp/tmpbbs2/` ;
- inscrire dans le `recovery.conf` que la restauration ne devra pas inclure les transactions au delà du 2015-10-29 à 22h10.

pitrery se charge de trouver automatiquement la sauvegarde la plus à jour permettant la restauration de l'instance à cette date précise.

6 Autres outils

6.1 Autres outils de l'écosystème

- De nombreux autres outils existent
- Moins importants que les précédents
  - répondant à des problématiques plus spécifiques
  - parfois anciens et potentiellement obsolètes
  - ou très récents et pas encore stables

Du fait du dynamisme du projet, l'écosystème des outils autour de PostgreSQL est très changeant.

En conséquence, on trouve de très nombreux projets autour du thème de la gestion des sauvegardes. Certains de ces projets répondaient à des problématiques très spécifiques (WAL-e, walmgr), d'autres sont assez anciens, pas forcément très actifs et souvent dépassés par des projets plus récents (pg_rman, pg_backman). On trouve néanmoins des nouveaux projets prometteurs (pgBackRest), néanmoins ils sont souvent trop jeunes pour être jugés stables.

6.2 pgBackMan - présentation

- Gestion de sauvegardes logiques
- Fonctionnalités limités
- Développement arrêté

pgBackMan (http://www.pgbackman.org/) est un outil d'archivage des sauvegardes produites via pg_dump et pg_dumpall. Son utilité est donc réduite à la gestion des sauvegardes logiques.

L'outil est très simple, voire limité. De plus son auteur semble inactif depuis plus d'un an. Dans le même registre, le projet pg_back est bien plus complet, et mieux maintenu.

6.3 walmgr - présentation

- Simplifie la réplication *Warm Standby*
- Membre de la suite Skytools
- Permet la gestion de sauvegardes PITR
  - manque de fonctionnalités
  - développement arrêté

walmgr fait partie de la suite Skytools (https://wiki.postgresql.org/wiki/SkyTools) développée par Skype. C'est outil conçu pour simplifier la gestion de la réplication Warm Standby et les sauvegardes PITR, en automatisant les opérations de mise en place et de bascule.

walmgr était un outil très utile entre 2007 et 2010. L'arrivée de la génération PostgreSQL 9.x avec la réplication en *streaming* et le *Hot Standby* a considérablement réduit l'intérêt de cette solution. Aujourd'hui le projet n'est plus actif, la documentation n'est plus disponible et l'avenir du logiciel est incertain.

6.4 WAL-E - présentation

- Gestion de sauvegardes PITR
- Focus sur le stockage en *cloud*
- Possibilités de restauration limitées
  - pas de génération du `recovery.conf`
  - pas de recréation des liens vers les tablespaces

WAL-E (https://github.com/wal-e/wal-e) est un programme très populaire pour gérer l'archivage continu des fichiers WAL et des sauvegardes à plat (*base backups*). De par sa conception, il est très adapté pour l'archivage des journaux de transactions vers des stockages *cloud*.

Il reste néanmoins assez limité dans les fonctionnalités offertes, notamment au niveau des possibilités de restauration. Des outils comme Barman ou pitrery sont donc généralement plus appropriés.

6.5 pg_rman - présentation

- Gestion de sauvegardes PITR
- Développé par NTT
- Peu de documentation
- Peu d'exemples d'utilisation connus

pg_rman (https://github.com/ossc-db/pg_rman), développé par NTT, est un outil directement inspiré par Oracle RMAN dont le but est d'automatiser la gestion des sauvegardes de type PITR. L'outil maintient un catalogue des sauvegardes disponibles et facilite leur restauration.

pg_rman est un projet actif mais peu dynamique. Même s'il existe une documentation minimale, il y a peu d'exemple d'utilisation en production. Il est plutôt conseillé d'utiliser des outils plus complets et plus pérennes comme pitrery ou Barman.

6.6 OmniPITR - présentation

- Gestion de l'archivage et du basebackup
- Mise en place de réplication par log-shipping
- Outil obsolète :
  - gestion du log-shipping en interne depuis PostgreSQL 9.0 (2010)
  - pas de génération de `recovery.conf`

OmniPITR, contrairement à ce qu'on pourrait déduire de son nom, ne permet pas de faire du PITR, mais plutôt de mettre en place de la réplication par log-shipping. Il est surtout utile pour les versions de PostgreSQL inférieures ou égales à la 8.4.

Il reste donc limité dans les fonctionnalités offertes au niveau de la restauration. Des outils comme Barman ou pitrery sont donc généralement plus appropriés.

6.7 pgBackRest - présentation

- Gestion de sauvegardes PITR
- Indépendant des commandes système
  - utilise un protocole dédié
- Sauvegardes complètes, différentielles ou incrémentales
- Gestion du multi-thread
- Projet récent (2014), non stabilisé

pgBackRest (https://github.com/pgmasters/backrest) est un outil de gestion de sauvegardes PITR écrit en perl, en cours de développement par David Steele de Crunchy Data.

Les principales fonctionnalités mises en avant sont :

- un protocole dédié pour le transfert / compression des données ;
- des opérations parallélisables en multi-thread ;
- la possibilité de réaliser des sauvegardes complètes, différentielles et incrémentielles ;
- la possibilité de réaliser l'archivage des WAL de façon asynchrone.

Le projet est actif et les fonctionnalités proposées sont intéressantes. Cet outil est néanmoins très récent, et n'a pas encore été stabilisé, il est donc préconçu de l'utiliser en environnement de production.

7 Conclusion

- Des outils pour vous aider!
- Pratiquer, pratiquer et pratiquer.
- Superviser les sauvegardes!

Nous venons de vous présenter des outils qui vont vous permettre de vous simplifier la tâche dans la mise en place d'une solution de sauvegarde fiable et robuste de vos instance PostgreSQL. Cependant, leur maîtrise passera par de la pratique, et en particulier, la pratique de la restauration. Le jour où la restauration d'une instance de production se présente, ce n'est généralement pas une situation confortable à cause du stress lié à une perte/corruption de données, interruption du service, etc. Autant maîtriser les outils qui vous permettront de sortir de ce mauvais pas.

N'oubliez pas également l'importance de la supervision des sauvegardes !