Support de cours : Automatiser la Maintenance de PostgreSQL - Introduction

La maintenance d'une base de données PostgreSQL est essentielle pour garantir sa performance, sa fiabilité et sa durabilité. L'automatisation des tâches de maintenance permet de réduire les interventions manuelles, prévenir les erreurs humaines et optimiser le temps d'administration. Ce document aborde les pratiques clés pour automatiser la maintenance de PostgreSQL.

1. Surveillance et Planification

1.1 Paramétrage des logs d'activité et des erreurs

La configuration des logs est cruciale pour surveiller et diagnostiquer les performances de PostgreSQL.

- **Paramètres importants à configurer dans le fichier postgresql.conf**
  - logging_collector = on : Active la collecte des logs.
  - log_directory = 'pg_log' : Spécifie le répertoire des logs.
  - log_filename = 'postgresql-%Y-%m-%d.log' : Définit un format de nom pour les fichiers de log.
  - log_statement = 'all' : Enregistre toutes les instructions SQL exécutées.
  - log_min_duration_statement : Enregistre les requêtes dont le temps d'exécution dépasse un certain seuil.

1.2 Utilisation de pg_stat_* pour le monitoring

Les vues du catalogue pg_stat_* fournissent des informations détaillées sur l'activité de la base de données.

1.2.1 pg_stat_activity

Cette vue permet de surveiller les connexions actives et les requêtes en cours.

- **Exemple de requête pour afficher les sessions actives :**

```
SELECT pid, usename, application_name, state, query, now() - pg_stat_activity.query_start AS duration
FROM pg_stat_activity WHERE state = 'active';
```

  - **Explications :** Cette requête affiche les processus en cours d'exécution, avec le nom de l'utilisateur, l'application utilisée et la durée d'exécution de la requête.
  - **Cas d'usage :** Identifier les requêtes longues ou bloquantes.

1.2.2 pg_stat_database

Cette vue fournit des statistiques agrégées par base de données, telles que le nombre de transactions et d'erreurs.

- **Exemple de requête pour afficher les statistiques des bases de données :**

```
SELECT datname, numbackends, xact_commit, xact_rollback, blks_read, blks_hit FROM pg_stat_database;
```

  - **Explications :** Cette requête montre le nombre de connexions actives (numbackends), les transactions validées (xact_commit), les transactions annulées (xact_rollback), et le ratio lecture sur disque vs lecture en cache.
  - **Cas d'usage :** Mesurer l'efficacité du cache et détecter les bases qui présentent un nombre élevé de rollbacks.

1.2.3 pg_stat_user_tables

Cette vue fournit des informations détaillées sur les lectures, écritures et analyses des tables.

- **Exemple de requête pour suivre l'activité sur les tables utilisateurs**

```
SELECT relname, seq_scan, seq_tup_read, idx_scan, n_tup_ins,
n_tup_upd, n_tup_del FROM pg_stat_user_tables
```

ORDER BY seq_scan DESC;

  - **Explications :** Cette requête affiche le nombre de scans séquentiels (seq_scan), le nombre de lectures de tuples (seq_tup_read), le nombre d'utilisations d'index (idx_scan), et le nombre de lignes insérées, mises à jour ou supprimées.
  - **Cas d'usage :** Identifier les tables fréquemment accédées sans indexation efficace.

1.2.4 pg_stat_replication

Cette vue est utile dans le cadre de la réplication pour surveiller l'état des réplications.

- **Exemple de requête pour vérifier l'état de la réplication :**

```
SELECT pid, client_addr, state, sent_lsn, write_lsn, flush_lsn, replay_lsn
FROM pg_stat_replication;
```

  - **Explications :** Cette requête affiche les informations sur les processus de réplication, y compris l'adresse du client et les positions LSN (Log Sequence Number) pour le suivi de l'avancement.
  - **Cas d'usage :** Surveiller le décalage entre le serveur principal et les répliques.

1.2.5 pg_stat_progress_vacuum

Cette vue montre l'avancement des processus de VACUUM en cours.

- **Exemple de requête pour afficher la progression de VACUUM :**

```
SELECT relname, phase, heap_blks_total, heap_blks_scanned,
heap_blks_vacuumed FROM pg_stat_progress_vacuum;
```

  - **Explications :** Cette requête indique le nom de la table, la phase actuelle de l'opération (scan, vacuum, etc.), et le nombre de blocs analysés ou nettoyés.
  - **Cas d'usage :** Suivre la progression et l'impact des tâches de maintenance VACUUM sur les performances.

2. Tâches de Maintenance

2.1 Analyse et Optimisation des Tables

L'optimisation des tables améliore les performances des requêtes et réduit la fragmentation.

- **ANALYZE :**
  - Met à jour les statistiques des tables pour aider le planificateur de requêtes.
  - Commande : ANALYZE [nom_table].
- **VACUUM :**
  - Libère l'espace occupé par les lignes supprimées ou mises à jour.
  - Commande : VACUUM [nom_table].
  - **VACUUM FULL :** Réorganise complètement la table et libère l'espace disque.
  - Attention : VACUUM FULL est bloquant et doit être planifié lors des périodes de faible activité.

2.2 Planification de Tâches via Cron (Linux) ou pgAgent

- **Cron (Linux) :** Utilisé pour planifier l'exécution de scripts shell à des horaires réguliers.
    - Exemple de tâche planifiée pour VACUUM hebdomadaire :

```
0 2 * * 7 /usr/local/bin/pg_vacuum.sh >> /var/log/pg_maintenance.log 2>&1
```

- **pgAgent :** Une extension PostgreSQL pour gérer des tâches planifiées à l'aide de requêtes SQL ou de commandes système.
    - Avantage : Intégré directement dans PostgreSQL, avec une gestion centralisée via pgAdmin.

3. Automatisation des Sauvegardes et Nettoyage

3.1 Scripts pour Backups Automatiques

Les sauvegardes régulières sont essentielles pour prévenir la perte de données en cas de panne.

- **Sauvegarde avec pg_dump :**

```
#!/bin/bash
BACKUP_DIR="/path/to/backup"
DATE=$(date +"%Y-%m-%d")
pg_dump -U postgres -F c -b -v -f "$BACKUP_DIR/db_backup_$DATE.dump" nom_base
```

- **Sauvegarde complète avec pg_basebackup :** Utilisé pour des sauvegardes complètes du cluster PostgreSQL.

3.2 Rotation des Fichiers de Logs

La rotation des fichiers de logs évite une accumulation excessive de fichiers volumineux.

- **Exemple de script pour supprimer les anciens fichiers de log :**

```
#!/bin/bash

LOG_DIR="/path/to/logs"

find "$LOG_DIR" -type f -name "*.log" -mtime +7 -exec rm {} \;
```

  - Ce script supprime les fichiers de log de plus de 7 jours.

Utilisation de logrotate :

- Outil natif Linux pour automatiser la rotation et la suppression des logs.
- Exemple de configuration :

```
/var/log/postgresql/*.log {
    daily
    missingok
    rotate 7
    compress
    delaycompress
    notifempty
    create 640 postgres adm
}
```

4. Plan de Reprise d'Activité (PRA)

Le RTO et le RPO sont les gardiens du temps et des données. Sans eux, votre Plan de Reprise d'Activité (PRA) ne répondra pas à vos enjeux et objectifs efficacement. Mais qu'est-ce que c'est et comment les calculer ?

▪ Pourquoi le RTO et le RPO?

Le RTO et le RPO sont proches de zéro. Si pour vous, l'ensemble de vos activités sont critiques et les enjeux trop importants pour absorber un arrêt, un Plan de Continuité d'Activité sera le plus adapté.

Le RTO et le RPO interviennent dans le PRA lors de la définition du périmètre de criticité de votre système d'information. Une fois les applications et activités critiques référencées, il convient de définir le RTO et le RPO pour définir les objectifs de la reprise du service.

▪ Qu'est-ce que le RTO ?

Le RTO ou Recovery Time Objective est la durée d'interruption maximale admissible.

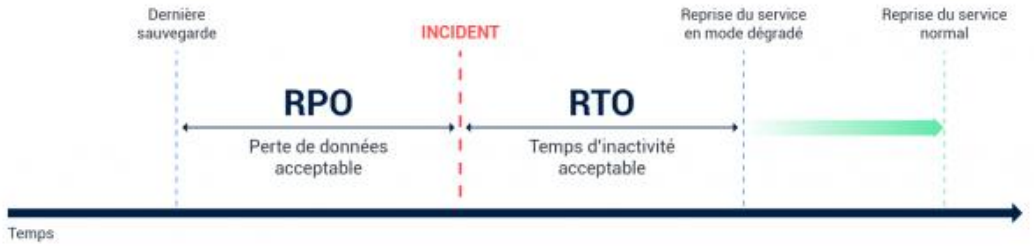
C'est le temps entre l'incident et la remise en route opérationnelle du système (serveur, réseau, applications…).

Par exemple, pour un RTO de deux heures, si l'incident se produit à 9h, le système sera de nouveau opérationnel à 11h maximum.

- **Qu'est-ce que le RPO ?**

Le RPO ou Recovery Point Objective est la perte de données maximale admissible. C'est le volume de données que vous êtes prêt à perdre entre la dernière sauvegarde et l'interruption de service. Les données comprises entre la dernière sauvegarde et l'interruption seront donc perdues.

Pour réduire au maximum le RPO, il est courant d'automatiser et de régulariser les sauvegardes. Mais des sauvegardes plus fréquentes entraînent des moyens techniques plus importants notamment sur le volume de stockage à disposition sur l'infrastructure et le réseau nécessaire.



Quelle est la différence entre les deux ?

Ce sont les notions et sollicitations qui différent entre les deux. Le RPO est généralement technique et matériel.

Le RTO, lui, implique beaucoup plus de ressources et est un plan plus large. Il sollicite toute la chaîne de l'entreprise :

- *Côté technique, l'équipe réseau vérifiera que l'accès au réseau est disponible. Ensuite, l'équipe système remettra en place les serveurs et les données. Et enfin, l'équipe développement/applicatif garantira le bon fonctionnement des applications. Cette chaîne technique doit être automatisée et/ou pilotée afin d'éviter des allers-retours entre les équipes.*
- *Côté marketing, l'équipe communication devra prévenir les clients de l'incident et du plan d'actions prévu.*
- *Suivant votre entreprise, d'autres services peuvent impactés et donc sollicités.*

L’organisation doit être bien définie et testée. Le RTO est ainsi beaucoup plus contrôlé.

Il est possible de créer un RTO et un RPO propre à chaque service/applicatif de l’entreprise. Chacun ayant ses propres enjeux et criticités, cette pratique peut être adaptée.

Il faut également savoir que ce n’est pas parce qu’il y a des sauvegardes régulières qu’on peut s’engager sur un RTO court. Et inversement, ce n’est pas parce que vous avez un excellent RPO que le RTO sera respecté en un claquement de doigts. Pour respecter le RTO, il faut des moyens et du temps.

Comment les calculer ?

Pour calculer le RTO et le RPO, il faut prendre en compte les potentiels impacts d'une indisponibilité. Une coupure aura deux impacts sur l'entreprise :

- *Impact direct comprenant la perte du chiffre d'affaires depuis l'incident, les salaires des employés, l'infrastructure à mettre en place pour restaurer le service et les interventions des prestataires externes.*
- *Impact indirect comprenant l'image de marque, la réputation de l'entreprise qui pourra entraîner la perte de chiffre d'affaires sur le moyen et long terme.*

Outre la notion de temps, les coûts sont très importants dans le calcul du RTO et du RPO. Plus le temps est court, plus le PRA sera onéreux. Ce calcul est un savant mélange entre vos moyens et la criticité de vos activités.

La définition du RTO RPO, s'accompagne de la définition des :

- *Moyens techniques, humains, organisationnels et financiers disponibles*
- *Technologies et méthodologies à utiliser*

Best practices

Il est important de se poser les bonnes questions pour ne rien négliger. On se dit souvent, ce n'est pas grave si cette application ne fonctionne pas une journée. Oui, mais si vos données n'ont pas été sauvegardées depuis une semaine ? Eh bien, acceptez-vous de tout perdre ? Comment réagiront vos clients et vos collaborateurs ?

Voici une liste de questions que j'estime importantes à se poser :

- *Quels sont les impacts sur la vie de mon entreprise si j'ai une interruption de service ?*
- *Acceptez-vous de perdre vos données ?*
- *Aujourd'hui, qu'avez-vous en place ?*
- *Avez-vous la garantie que l'existant est efficace ?*
- *Qu'est-ce qui se passe en cas de sinistre ?*
- *Quels sont mes objectifs pour pallier un sinistre ?*

Autre point, n'hésitez pas à vous faire accompagner par un prestataire dont c'est le métier. Sur l'ensemble du projet ou uniquement pour une partie, il est toujours intéressant d'avoir un regard extérieur à son organisation.

Conclusion

L'automatisation de la maintenance de PostgreSQL implique la mise en place de stratégies efficaces pour surveiller, optimiser et protéger les données. En utilisant des outils tels que pg_stat_*, cron, pgAgent, et des scripts adaptés, il est possible de garantir un fonctionnement fluide de la base de données tout en réduisant les risques d'erreur humaine. Une planification rigoureuse et des sauvegardes régulières sont indispensables pour assurer la pérennité des systèmes.