Livre Blanc DALIBO #02

Industrialiser PostgreSQL

19.04

DALIBO
L'expertise PostgreSQL

Dalibo SCOP

https ://wwww.dalibo.com/

Industrialiser PostgreSQL

Livre Blanc DALIBO #02

TITRE : Industrialiser PostgreSQL
SOUS-TITRE : Livre Blanc DALIBO #02

REVISION : 19.04

Remerciements : Ce livre blanc est le fruit d’un travail collectif. Nous remercions chaleureusement ici toutes les personnes qui ont contribué directement ou indirectement à cet ouvrage.

À propos de DALIBO :

DALIBO est le spécialiste français de PostgreSQL. Nous proposons du support, de la formation et du conseil depuis 2005.

Retrouvez toutes nos livres blancs sur https ://dalibo.com/

Chers lectrices & lecteurs,

Nos livres blancs sont issus de plus de 12 ans d'études, d'expérience de terrain et de passion pour les logiciels libres. Pour Dalibo, l'utilisation de PostgreSQL n'est pas une marque d'opportunisme commercial, mais l'expression d'un engagement de longue date. Le choix de l'Open Source est aussi le choix de l'implication dans la communauté du logiciel.

Au-delà du contenu technique en lui-même, notre intention est de transmettre les valeurs qui animent et unissent les développeurs de PostgreSQL depuis toujours : partage, ouverture, transparence, créativité, dynamisme… Le but premier de nos livres blancs est de vous aider à mieux exploiter toute la puissance de PostgreSQL mais nous espérons également qu'ils vous inciteront à devenir un membre actif de la communauté en partageant à votre tour le savoir-faire que vous aurez acquis avec nous.

Nous mettons un point d'honneur à maintenir nos productions à jour, avec des informations précises et des exemples détaillés. Toutefois malgré nos efforts et nos multiples relectures, il est probable que ce document contienne des oublis, des coquilles, des imprécisions ou des erreurs. Si vous constatez un souci, n'hésitez pas à le signaler via l'adresse contact@dalibo.com !

Table des Matières

INDUSTRIALISER POSTGRESQL

LES DÉFIS DE LA GESTION DE ``L'ACTIF POSTGRESQL''

Les Directions des Systèmes d'Information (DSI) sont confrontés à plusieurs défis dans la gestion de leur "actif PostgreSQL" :

- à court terme, et dans des délais souvent contraints, elles doivent mettre à la disposition des applications un "service PostgreSQL" totalement exploitable, robuste et performant,
- à moyen terme, elles doivent disposer de moyens efficaces permettant de déployer rapidement le "socle PostgreSQL" sur de nouveaux environnements, tout en maîtrisant strictement les coûts associés à ce déploiement,
- à long terme, elles doivent s'assurer de la pérennité et du maintien en condition opérationnelle du service PostgreSQL, dans un contexte d'évolution très rapide de la "technologique PostgreSQL" (une version majeure du SGBD chaque année).

4 ENJEUX MAJEURS

Pour chacun de ces niveaux, il existe en effet un certain nombre d'écueils à éviter. Par exemple :

1. la mise à disposition rapide d'une **infrastructure incomplète** ou peu fiable ou pas assez performante donnerait, au delà des perturbations générées, une mauvaise image de la solution qui mettrait en péril l'adoption même de la solution,
2. **un délai trop long** de mise à disposition de l'infrastructure PostgreSQL pour les applications serait aussi préjudiciable :
    - le retour sur investissement des projets associés serait retardé d'autant,
    - et les DSI risqueraient que des utilisations cachées de PostgreSQL se développent hors de leur contrôle dans des directions métiers (le *"shadow PostgreSQL"*).
3. **un défaut de standardisation** de la solution PostgreSQL déployée induirait au fil du temps une prolifération d'environnements techniques hétérogènes dont l'administration serait nettement plus lourde et donc plus coûteuse, un défaut d'industrialisation de la solution PostgreSQL complexifierait également ses déploiements sur les serveurs, générant un surcoût préjudiciable et une potentiellement une frustration des clients de la solution,

4. **un défaut d'investissement** dans le maintien de l'infrastructure PostgreSQL à un bon niveau technique peut aussi se révéler être un mauvais calcul :
    - on constate en effet classiquement dans ce cas une dégradation progressive de la qualité du service rendu, même si elle n'est pas toujours clairement perceptible,
    - et exploiter d'anciennes versions de logiciels ne permet pas de tirer profit des améliorations (fonctionnelles, performance, etc) apportées.

Il apparaît donc clairement que l'industrialisation de la solution PostgreSQL est une étape stratégique dans la modernisation d'un Système d'Information.

LE CONCEPT DE ``SOCLE''

La réponse à ce triple défi consiste à bâtir, à déployer puis à maintenir ce qu’on peut appeler un “socle PostgreSQL”, qui constituera une brique essentielle de la composante “bases de données” du Système d’Information.

3 AXES DE TRAVAIL : OUTILS, PROCÉDURES, MANUELS

De manière très opérationnelle, bâtir un socle consiste pour l’essentiel à :

- Sélectionner, installer, paramétrer et mettre en œuvre le logiciel PostgreSQL et le catalogue d’outils associés nécessaires à une exploitation robuste,
- Développer les procédures d’exploitation permettant d’exploiter efficacement tous ces composants logiciels,
- Rédiger toute la documentation indispensable à l’exploitation de ces composants.

Ce socle ainsi bâti est donc constitué d’un ensemble cohérent de logiciels, procédures et documentations couvrant tous les besoins relatifs à la fourniture du “service PostgreSQL”. Ce socle a ensuite vocation à être déployé de manière standardisée sur tous les environnements qui le nécessite.

7 DOMAINES COUVERTS : DE L'INSTALLATION À LA HAUTE-DISPONIBILITÉ

Autour de ces 3 axes de travail, les sujets peuvent décomposés en 7 domaines fonctionnels :

1. Le domaine **Architecture** encadre l’utilisation du service et les choix stratégiques
2. Le domaine **Déploiement** assure l’homogénéité du parc d’instances
3. Le domaine **Maintenance** garantit la qualité du service
4. Le domaine **Sauvegarde** définit les actions mise en oeuvre pour assurer la pérennité des données, notamment le `Plan de Reprise d'Activité` (PRA)
5. Le domaine **Supervision** encadre l’usage des outils de monitoring
6. Le domaine **Haute-Disponibilité** avec comme objectif le maintien du service via un `Plan de Continuité d'Activté` (PCA)
7. Le domaine **Sécurité** spécifie les protocoles d’accès aux données

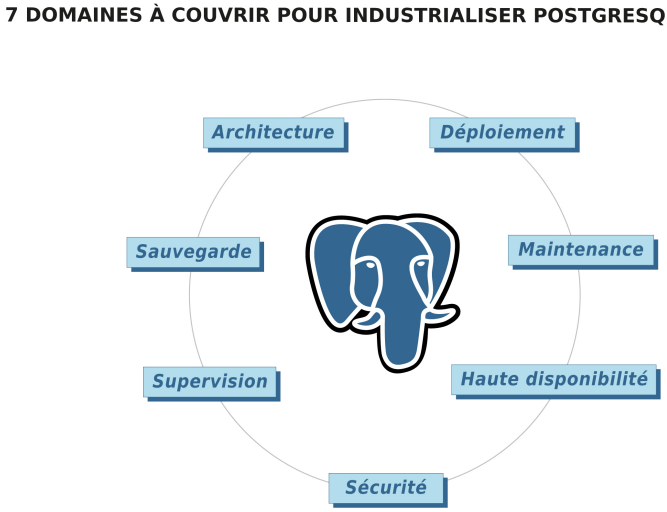

Fig. 1 : 7 domaines à couvrir pour industrialiser PostgreSQL

LE DOMAINE ARCHITECTURE

Avant de lancer la phase d'implémentation du socle, une étude doit être réalisée pour fixer les choix d'architecture qui devront guider sa mise en œuvre.

Cette phase d'architecture doit être formalisée par un "**Dossier d'Architecture Technique**" qui décrira dans le détail les choix d'architecture retenus.

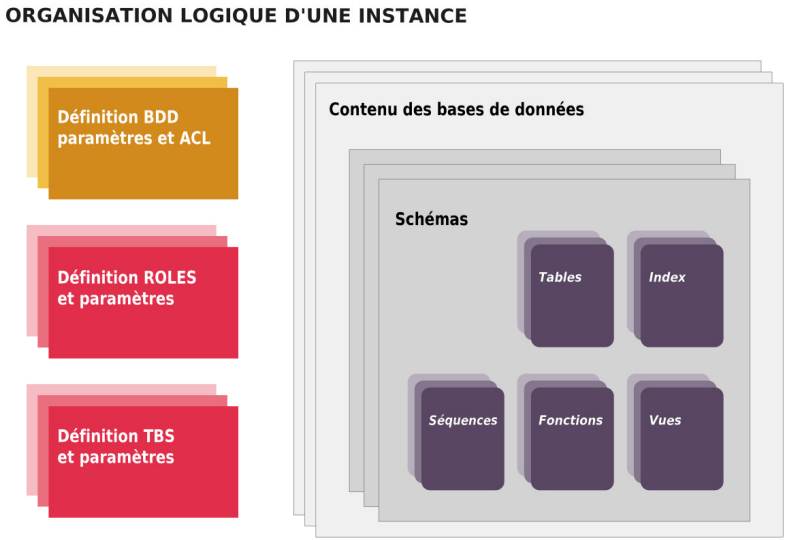

Fig. 2 : Structure d'une instance Postgres

Parmi les grands thèmes à traiter dans le cadre de l'étude d'architecture, citons :

- l'intégration des composants du socle sur les plateformes (types de serveur, stockage, ressources allouées,...),
- les relations entre les applications et les bases de données,
- les inter-relations entre les bases de données,
- l'évaluation de la Perte Maximale de Données Tolérable (PMDT), avec les moyens de répondre à cet objectif : sauvegarde/restauration, archivage des journaux de transactions, "Point In Time Recovery",
- les besoins en matière de sécurité (confidentialité, authentification, principes de gestion des droits,...),

- la quantification des flux entre les moteurs PostgreSQL et tous les composants externes (applications, serveur d'archivage, de sauvegarde, de réplication,...).

Mais cette étude doit également prendre en compte :

- les besoins en haute disponibilité, avec l'évaluation du Délai Maximal d'Indisponibilité Admissible (DMIA) et des moyens de répondre à cet objectif,
- les besoins en "scalabilité" (capacité à prendre en charge la croissance attendue tant des volumes de données que des charges d'accès).

Dès cette étape, la rédaction du "Cahier de Recette" permettra, à la fin de la phase de construction, de garantir la cohérence entre les livrables produits et les attentes identifiées.

Très peu de composants logiciels sont concernés par ce domaine. Néanmoins, des outils de migration tels que ora2pg (outil facilitant la migration des bases Oracle et MySQL) peuvent aider à la cartographie d'un parc de bases existantes candidates à une migration vers PostgreSQL. (Cf. le Livre Blanc "Migrer d'Oracle à PostgreSQL[1] ")

1. https://cloud.dalibo.com/p/Dalibo_Migrer_Oracle_PostgreSQL.latest.pdf

LE DOMAINE DÉPLOIEMENT

Comme le socle est destiné à être déployé sur potentiellement un grand nombre de serveurs, le processus d'installation et de configuration doit être industrialisé.

INSTALLATION DES LOGICIELS

Dans les environnements virtualisés, les déploiements peuvent s'effectuer au travers de "**templates de VM**" préalablement construites. Une approche alternative, valable pour tout type d'environnement technique, consiste à développer une **procédure d'installation complète** déployant un ensemble prédéfini de paquets, au format standard de l'OS utilisé (RPM, deb,...). Cette procédure d'installation doit naturellement inclure toutes les procédures et les outils nécessaires à la production.

La procédure d'installation et de déploiement doit être à la fois normative, pour répondre au besoin de standardisation, mais aussi assez souple pour couvrir les différents cas d'usage de PostgreSQL dans l'entreprise ou l'organisation.

Un **Manuel d'Installation** documente tout le processus d'installation.

GESTION DES INSTANCES, BASES ET UTILISATEURS

Une fois les logiciels installés, il faut pouvoir créer une instance, la configurer, créer la ou les bases de données et les utilisateurs nécessaires.

La **configuration des instances** doit, elle aussi, être standardisée sur l'ensemble du parc PostgreSQL. Néanmoins, certains paramètres peuvent être à valoriser en fonction de la taille de chaque instance (volumétrie, charge d'accès,...). La définition des règles de valorisation des "bons paramètres" conditionnera naturellement la qualité du "service PostgreSQL". Cette tâche essentielle requiert un bon niveau d'expertise sur le SGBD.

La gestion des instances, des bases de données et des utilisateurs impose de disposer, pour chacun de ces types d'objet, de procédures de création, de suppression et de modification. Il faut également prévoir une procédure de duplication de base de données. Toutes ces procédures doivent être documentées dans le "**Manuel d'Exploitation**".

LE DOMAINE MAINTENANCE

Maintenir le niveau de qualité du “Service PosgreSQL” impose de dépenser de l’énergie afin de lutter contre une forme d’entropie croissante naturelle. Concrètement, les équipes de production, et en particulier les DBA, doivent passer du temps pour analyser le fonctionnement des instances PostgreSQL et effectuer les tâches nécessaires au maintien de la qualité du service.

Pour limiter la charge de travail nécessaire pour ces tâches au quotidien, il est nécessaire de disposer :

- de bons outils d’analyse,
- de procédures d’intervention efficaces.

OUTILS D'ANALYSE

On peut classifier les **outils d’analyse** en deux catégories : ceux qui permettent d’avoir une vision en temps réel de l’activité et ceux qui permettent d’avoir une vision à posteriori de l’activité. La première catégorie est couverte par le Domaine Supervision décrit plus bas.

Parmi les outils qui permettent d’analyser le fonctionnement des instances PostgreSQL, citons simplement deux logiciels :

- pgBadger[2] , incontournable, exploite intensivement le contenu des traces des instances PostgreSQL, pour en restituer des informations de synthèse sous forme graphique,
- pgCluu[3] est un outil de collecte et d’analyse de données relatives au fonctionnement interne des instances.

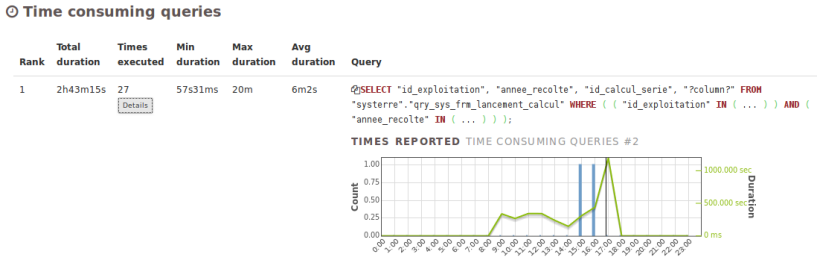

Fig. 3 : Analyse de requête SQL avec pgBadger

2. https://github.com/dalibo/pgbadger
3. http://pgcluu.darold.net/

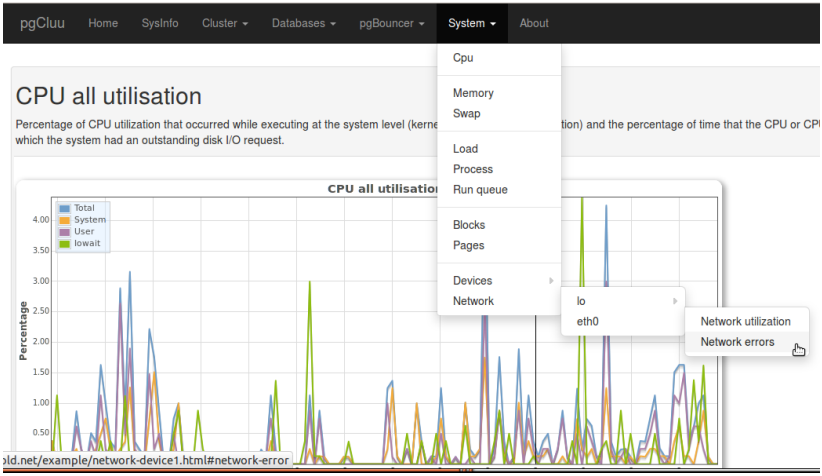

Fig. 4 : Charge sur les CPU avec pgCluu

Les outils sélectionnés doivent permettent des diagnostics fins en apportant une vision précise sur à minima : la consommation mémoire, les espaces disques, les I/O, le trafic SQL, l'utilisation des journaux de transaction.

PROCÉDURES

Les procédures liées à ce Domaine Maintenance doivent couvrir en particulier :

- la gestion opérationnelle des instances : démarrage, arrêt, rechargement de la configuration,
- la lutte contre la désorganisation de la base : défragmentation (vacuum), réindexation, réorganisation,...

Comme pour les autres procédures, celles-ci doivent être décrites dans le "Manuel d'Exploitation".

LE DOMAINE SAUVEGARDE ET RESTAURATION

Les sauvegardes/restaurations constituent bien sûr un élément clé pour garantir la pérennité du patrimoine des données de l'entreprise ou de l'organisation. Avoir un socle solide dans ce domaine est donc essentiel.

Or les fonctions offertes par le logiciel PostgreSQL dans ce domaine nécessitent d'être enrichies par des logiciels complémentaires, tels que les deux composants issus de la R&D Dalibo :

- pg_back[4] pour les sauvegardes logiques,
- pitrery[5] pour les sauvegardes physiques.

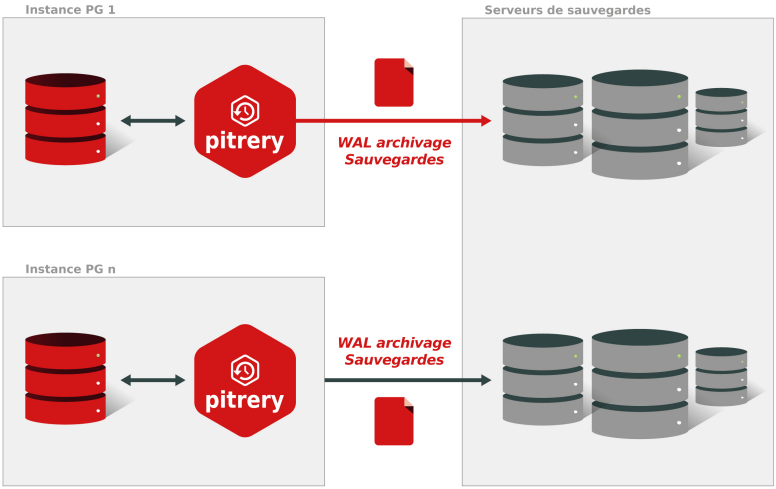

Fig. 5 : Exemples de sauvegarde avec Pitrery

Quels que soient les outils sélectionnés, au delà de la simple conservation des sauvegardes des instances ou des bases de données, il est important de nos jours de pouvoir mettre en place un archivage fiable des journaux de transaction afin d'être capable de mettre en œuvre, en cas de besoin, les techniques de `Point In Time Recovery` (PITR).

4. https://github.com/orgrim/pg_back
5. https://dalibo.github.io/pitrery/

PLUSIEURS NIVEAUX DE RÉSILIENCE

La solution globale retenue devra répondre aux objectifs de “Perte Maximale de Données Tolérable” (PMDT) définis dans le Domaine Architecture.

En matière de procédures, les besoins suivants doivent être couverts :

- sauvegarde logique (encapsulant pg_dump),
- restauration totale ou partielle à partir d’une sauvegarde logique,
- sauvegarde physique de l’instance, complète ou incrémentale,
- restauration de l’instance à un point dans le temps (PITR).

Ces procédures seront, elles aussi, documentées dans le Manuel d’Exploitation.

Il est également conseillé de compléter cette documentation par la rédaction d’un “Plan de Reprise d’Activité” qui décrit précisément le protocole à suivre pour reconstuire les bases de données et remettre en route le service Postgres après un incident majeur de production.

LE DOMAINE SUPERVISION

Pouvoir suivre en temps réel le fonctionnement du service PostgreSQL et disposer d'un système d'alerte en cas de dégradation ou de défaillance sont essentiels au maintien de la qualité du service.

En matière d'outils, deux grandes approches sont possibles. On peut intégrer les composants PostgreSQL dans une supervision plus large, voir unifiée du système de production. Mais on peut également bâtir une supervision dédiée à PostgreSQL.

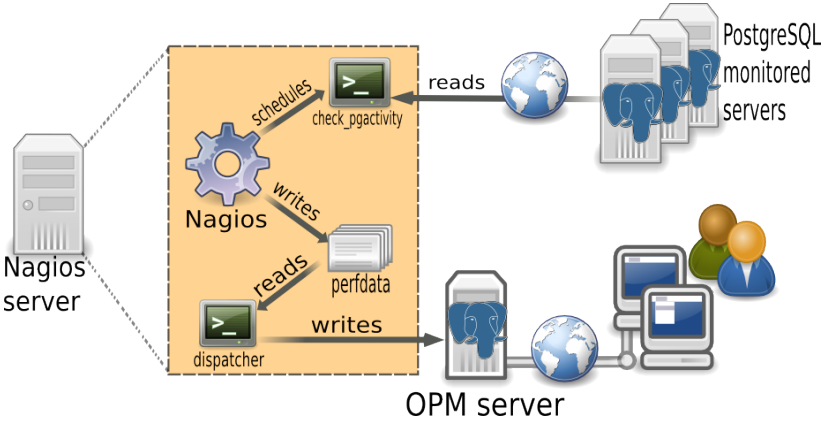

Fig. 6 : Architecture de supervision basée sur OPM

Dans le premier cas, on pourra utiliser des logiciels comme Nagios ou Zabbix et dans le second cas, des logiciels comme PoWA[6] (suivi du trafic en temps réel), OPM[7] ou TemBoard[8] (collecte et centralisation des statistiques d'activité du parc d'instances).

6. http://dalibo.github.io/powa/
7. http://opm.io
8. http://temboard.io/

ORGANISER LA COLLECTE D'INFORMATION

Dans quasiment tous les cas, il sera nécessaire de déployer sur les serveurs PostgreSQL un **agent de supervision**. Le déploiement ou la désactivation d'un agent de supervision, et l'activation d'une sonde PostgreSQL ou système doivent faire l'objet de procédures prêtes à l'emploi et documentées.

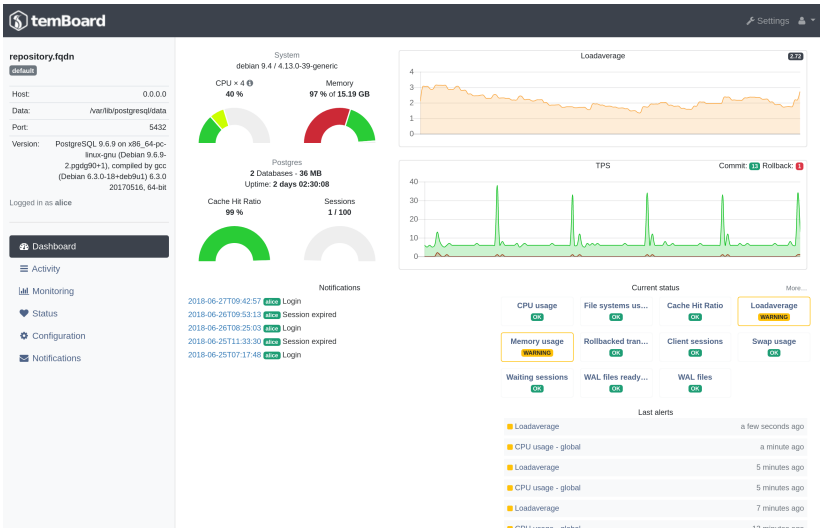

Fig. 7 : Tableau de bord en temps réel avec Temboard

Mais la sélection et le déploiement des outils n'est qu'une première étape de ce chantier. Il faut en effet définir également les métriques collectées et les seuils de déclenchement associés. Cette activité requiert elle aussi un bon niveau d'expertise en administration PostgreSQL.

La rédaction d'un "**Guide de Supervision**" permettra de formaliser ces choix et d'en garder la trace.

LE DOMAINE HAUTE-DISPONIBILITÉ

Le logiciel PostgreSQL offre de manière native un certain nombre de fonctionnalités : réplication physique, asynchrone et/ou synchrone, par les journaux de transaction ou au "fil de l'eau", avec une ou plusieurs instances secondaires, éventuellement en cascade, possibilité d'accéder en lecture seule aux serveurs secondaires, etc.

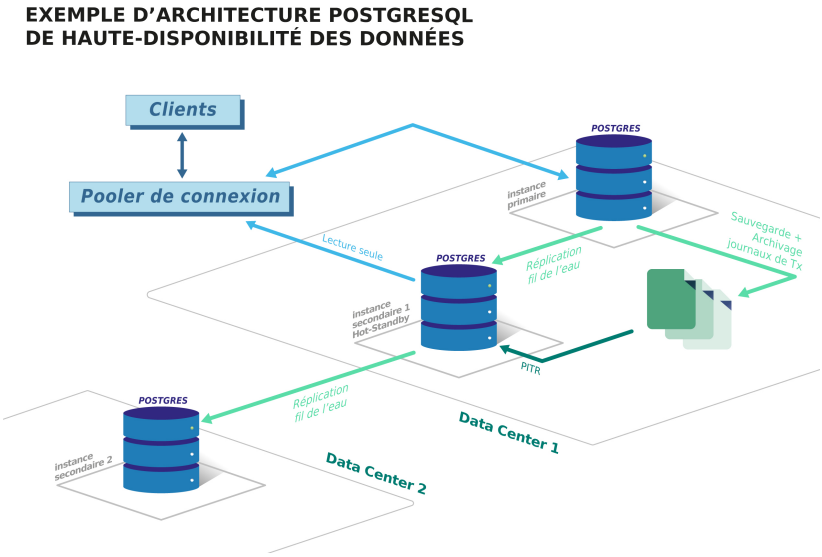

Fig. 8 : Exemple d'architecture de réplication avec PostgreSQL

La mise en œuvre de ces techniques permet de bâtir des architectures avec un bon niveau de haute disponibilité. Mais dans certains cas, il faut pouvoir aller plus loin. Typiquement, deux fonctionnalités complémentaires peuvent être souhaitées :

- du **pooling de connexion**, lorsque le nombre de clients accédant à l'instance est élevé,
- un dispositif de **bascule automatique** d'un serveur primaire sur un serveur secondaire en cas d'indisponibilité du serveur primaire (failover automatique).

Dans le premier cas, on pourra choisir un logiciel tel que pgBouncer[9] . Dans le second cas, on pourra tirer profit du logiciel PostgreSQL Automatic Failover[10] (PAF), basé sur

9. https://pgbouncer.github.io/
10. http://dalibo.github.io/PAF/

Pacemaker/Corosync et issu de la R&D Dalibo.

Un certain nombre de procédures doivent être développées et documentées, notamment :

- La mise en place d'un nœud secondaire Hot-Standby
- La bascule d'urgence (`FAIL OVER`)
- La bascule contrôlée (`SWITCH OVER`)
- Le retour à l'état stable

Le document "Plan de Continuité d'Activité" devra décrire les mécanismes retenus et en détailler la mise en œuvre.

LE DOMAINE SÉCURITÉ

Là encore, le logiciel PostgreSQL offre un certain nombre de fonctionnalités qui participent à la sécurisation de l'accès aux données.

L'authentification des connexions peut être contrôlée de multiples manières : mot de passe encrypté avec période de validité, Filtrage par les adresses réseau, Active Directory, Kerberos,... Et une fois connecté, l'utilisateur dispose des droits d'accès aux objets de la base de données qui lui auront été attribués (GRANT et DEFAULT PRIVILEGES).

Néanmoins, il peut être parfois nécessaire d'aller plus loin dans la sécurité.

En particulier, on peut vouloir renforcer la confidentialité des informations stockées en chiffrant certaines données sensibles dans la base de données. L'extension **pgcrypto** répond à ce type de besoin.

Par ailleurs, l'activation du module **Security-Enhanced PostgreSQL** (SE-PostgreSQL), qui offre une granularité et un traçage très fin des accès aux données, peut permettre la mise en place d'une politique de sécurité sophistiquée à l'image de SE-Linux.

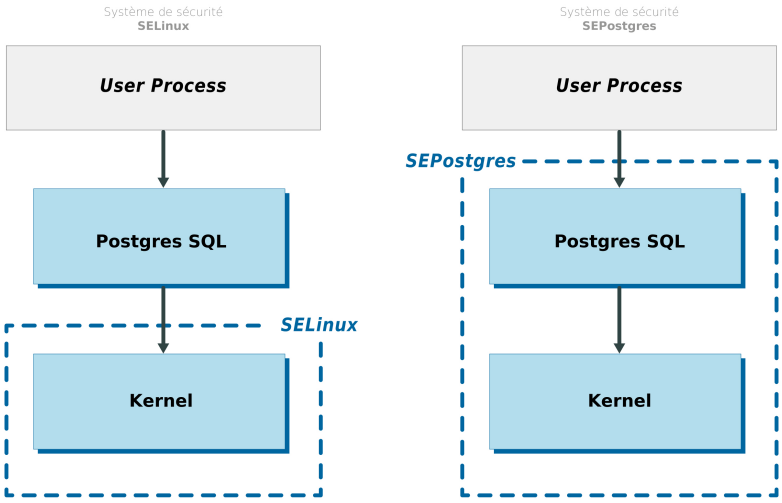

Fig. 9 : Sécurisation progressive de la stack applicative

En fonction des choix effectués en matière de sécurité, il faut prévoir le développement

et la documentation des procédures nécessaires, tels que :

- Confinement d'une instance,
- Fabrication de clés,
- Chiffrement/Déchiffrement des données.

La rédaction d'un "**Guide de Sécurisation**" permettra de lister et d'apprécier les risques identifiés, et de lister les solutions concrètes déployées pour mitiger ces risques. Ce document pourra servir de référence notamment auprès des autorités concernées par la sécurité (RSSI par exemple).

INTÉGRATION CONTINUE

Dans ce qui précède, nous avons défini ce que peut être le concept de socle et nous en avons précisé le contenu. L'enjeu ici n'est pas simplement de réussir l'investissement initial pour mettre en place le socle mais de **"faire vivre" ce socle** en incorporant progressivement les nouveautés et les améliorations produite par la communauté PostgreSQL.

Vous devez donc aligner le cylce de vie du socle sur celui de PostgreSQL. Bonne nouvelle ! Le cycle des versions de Postgres est à la fois simple, régulier et dynamique :

- Une version majeure est livrée chaque année
- Chaque version majeure est maintenue pendant 5 ans

Version PostgreSQL	Date de sortie	Supportée jusque	Supportée
10	01/10/2017	01/10/2022	Oui
9.6	01/09/2016	01/09/2021	Oui
9.5	01/01/2016	01/01/2021	Oui
9.4	01/12/2014	01/12/2019	Oui
9.3	01/09/2013	01/09/2018	Oui
9.2	01/09/2012	01/09/2017	Non
9.1	01/09/2011	01/09/2016	Non
9.0	01/09/2010	01/09/2015	Non

Fig. 10 : Cycle des versions de PostgreSQL

Par exemple : si vous avez développé un socle autour de PostgreSQL 9.6 en 2016 vous pouvez conserver cette version jusqu'en 2021. A l'inverse il peut être tentant d'intégrer chaque année la nouvelle version de Postgres dans le socle afin de fournir les toutes dernières avancées aux équipes projets…

Nous préconisons généralement une stratégie intermédaire basée sur des cycles de 3 ans qui garantit à la fois la stabilité des environnements et l'enrichissement continu du service PostgreSQL.

Notons également qu'il est possible de développer un **socle multi-versions** mais que cela implique un investissement très élevé en terme de maintenance.

Examinons maintenant quelles sont les différentes façons de développer puis de maintenir un socle. Trois types d'approches sont envisageables :

- faire appel à des ressources internes à l'entreprise ou l'organisation,
- sous-traiter le développement du socle,
- se procurer un socle déjà construit.

DÉVELOPPEMENT INTERNE

Développer puis maintenir un socle PostgreSQL tel qu'on a pu le définir, nécessite de mobiliser des ressources de plusieurs profils. Le profil DBA sera évidemment au cœur du dispositif. Mais il devra s'adjoindre également les services de profils architectes, administrateurs système et analystes d'exploitation.

Globalement, l'équipe ainsi constituée devra avoir un bon niveau d'expertise sur le logiciel PostgreSQL, mais aussi une bonne connaissance des outils périphériques et plus largement des technologies des Systèmes d'Information. Elle devra être capable de coder de manière fiable et de documenter les procédures d'exploitation.

Les charges de travail nécessaires au développement puis à la maintenance du socle sont conséquentes. Au delà de l'investissement initial en expérimentation, développement, test, etc., aboutissant à la première version du socle, il faut maintenir le socle en en gommant les éventuelles imperfections découvertes à l'usage, mais aussi en le mettant à niveau au fur et à mesure de l'arrivée de nouvelles versions des logiciels. Il y a donc un travail très régulier de veille technologique, de test et de qualification à poursuivre.

Les charges de travail que nous pouvons voir chez nos clients qui ont entrepris cette démarche sont souvent aux alentours d'une personne à temps plein pour la phase de développement pendant **9 à 12 mois** (suivant les fonctionnalités mises en œuvre), et d'à minima un **demi équivalent temps plein** par la suite pour le maintien en condition opérationnelle du socle.

La disponibilité et le niveau de compétence des ressources internes sont donc les facteurs clés de réussite de cette approche.

DÉVELOPPEMENT SOUS-TRAITÉ À L'EXTÉRIEUR

Sous-traiter le développement du socle peut permettre d'éviter de mobiliser de ressources internes et/ou de disposer plus rapidement de profils au niveau de compétence requis pour ces travaux. Il faut donc pour ce faire trouver sur le marché les profils à forte technicité à même de répondre à la demande.

Potentiellement séduisante à court terme, cette approche présente néanmoins un inconvénient sur le moyen terme quand il va s'agir de faire évoluer le socle. Le prestataire pourra, pendant un certain temps, effectuer le support des procédures qu'il aura rédigées. Mais, sauf achat d'une nouvelle prestation, il ne pourra prendre en charge les évolutions du socle.

SE PROCURER UN SOCLE DÉJÀ CONSTRUIT

Enfin, il est possible de se procurer sur le marché un socle déjà construit. C'est ce que propose par exemple Dalibo avec ses deux socles "Dalibo Essential PostgreSQL" et "Dalibo Advanced Postgres".

Ces solutions, construites uniquement à partir de composants open source, comprennent les logiciels, les procédures et la documentation associée permettant de bâtir une solution PostgreSQL solide et rapidement déployable en production. Elle intègre également le support de tous ces composants sur le moyen et le long terme.

Associés aux formations, ces socles constituent un moyen de rapidement mettre en place un service PostgreSQL. Mais ils répondent aussi à la nécessité de standardiser les environnements PostgreSQL déployés au sein du Système d'Information et enfin d'assurer la pérennité du socle sur le long terme.

ANNEXE A : RÉFÉRENCES

Quelques liens complémentaires :

Le Socle Dalibo Essential Postgres
https://cloud.dalibo.com/p/Dalibo_Socle_Essential_Postgres.pdf

Livre Blanc “Migrer d’Oracle à PostgreSQL” :
https://cloud.dalibo.com/p/Dalibo_Migrer_Oracle_PostgreSQL.latest.pdf

Présentation des logiciels issus de la R&D DALIBO :
https://cloud.dalibo.com/p/dalibo_presentation_rd_a4.pdf

Catalogue de Formation :
https://cloud.dalibo.com/p/dalibo_catalogue_formations_postgresql.pdf

Présentation du Support Technique :
https://cloud.dalibo.com/p/support_standard.pdf

SLA Standard :
https://cloud.dalibo.com/p/dalibo_sla-socle-essential.latest.pdf

Temboard : supervision, administration, dashboard et bien plus
https://www.youtube.com/watch?v=OZvw_IfMx54

ANNEXE B - DÉFINITIONS

Instance : désigne un serveur PostgreSQL. Une instance PostgreSQL peut contenir plusieurs bases de données. Il est possible d'héberger plusieurs instances sur un hôte Linux.