Module W7a

Solutions de Haute Disponibilité de service

24.04

Table des matières

Sur ce document

Formation	Module W7a
Titre	Solutions de Haute Disponibilité de service
Révision	24.04
PDF	https://dali.bo/w7a_pdf
EPUB	https://dali.bo/w7a_epub
HTML	https://dali.bo/w7a_html
Slides	https://dali.bo/w7a_slides

Vous trouverez en ligne les différentes versions complètes de ce document.

Chers lectrices & lecteurs,

Nos formations PostgreSQL sont issues de nombreuses années 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 formations est de vous aider à mieux exploiter toute la puissance de PostgreSQL mais nous espérons également qu'elles 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 manuels 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 formation@dalibo.com[1] !

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 formations sur https://dalibo.com/formations

[1] mailto:formation@dalibo.com

Remerciements

Ce manuel de formation est une aventure collective qui se transmet au sein de notre société depuis des années. Nous remercions chaleureusement ici toutes les personnes qui ont contribué directement ou indirectement à cet ouvrage, notamment :

Jean-Paul Argudo, Alexandre Anriot, Carole Arnaud, Alexandre Baron, David Bidoc, Sharon Bonan, Franck Boudehen, Arnaud Bruniquel, Pierrick Chovelon, Damien Clochard, Christophe Courtois, Marc Cousin, Gilles Darold, Jehan-Guillaume de Rorthais, Ronan Dunklau, Vik Fearing, Stefan Fercot, Pierre Giraud, Nicolas Gollet, Dimitri Fontaine, Florent Jardin, Virginie Jourdan, Luc Lamarle, Denis Laxalde, Guillaume Lelarge, Alain Lesage, Benoit Lobréau, Jean-Louis Louër, Thibaut Madelaine, Adrien Nayrat, Alexandre Pereira, Flavie Perette, Robin Portigliatti, Thomas Reiss, Maël Rimbault, Julien Rouhaud, Stéphane Schildknecht, Julien Tachoires, Nicolas Thauvin, Be Hai Tran, Christophe Truffier, Cédric Villemain, Thibaud Walkowiak, Frédéric Yhuel.

Forme de ce manuel

Les versions PDF, EPUB ou HTML de ce document sont structurées autour des slides de nos formations. Le texte suivant chaque slide contient le cours et de nombreux détails qui ne peuvent être données à l'oral.

Licence Creative Commons CC-BY-NC-SA

Cela inclut les diapositives, les manuels eux-mêmes et les travaux pratiques. Cette formation peut également contenir quelques images et schémas dont la redistribution est soumise à des licences différentes qui sont alors précisées.

Marques déposées

PostgreSQL® Postgres® et le logo Slonik sont des marques déposées[3] par PostgreSQL Community Association of Canada.

Versions de PostgreSQL couvertes

Ce document ne couvre que les versions supportées de PostgreSQL au moment de sa rédaction, soit les versions 12 à 16.

Sur les versions précédentes susceptibles d'être encore rencontrées en production, seuls quelques points très importants sont évoqués, en plus éventuellement de quelques éléments historiques.

Sauf précision contraire, le système d'exploitation utilisé est Linux.

[3] https://www.postgresql.org/about/policies/trademarks/

1/ Haute disponibilité de service

1.1 PRÉAMBULE

- Quelle architecture pour l'instance primaire et ses secondaires ?
- Plusieurs architectures et solutions possibles
- *Shared Storage* : Pacemaker
- *Shared Nothing*
  - Pacemaker
  - Patroni

1.2 SHARED STORAGE

- Architecture simple
  - *Cold standby* matériel
- Disque partagé entre primaire et secours
  - accessible par **un seul** serveur
- Redondance du stockage nécessaire
  - SAN, DRBD…
- Alternative : créer/déplacer une machine virtuelle
- Pacemaker : possible

Une architecture de type *Shared Storage* repose essentiellement sur un disque partagé entre au moins deux serveurs. Les données de l'instance sont situées sur le disque partagé. En cas de panne sur le serveur actif, le ou les disques sont montés sur l'un des serveurs de secours et l'instance PostgreSQL y est démarrée.

L'avantage d'une telle architecture est son apparente simplicité. Les données étant habituellement situées dans un SAN, il est aisé en cas d'incident sur le serveur principal de monter les disques dans un serveur de secours et redémarrer PostgreSQL. La configuration de PostgreSQL reste au plus simple et l'espace occupé par les données de l'instance n'est pas dupliqué à travers plusieurs serveurs.

La haute disponibilité des données doit être prise en charge par une réplication au niveau disque (SAN, DRBD, etc).

La disponibilité du service dépend de la technologie utilisée : machine virtuelle, lame, serveur physique. Certaines technologies de virtualisation sont capables de « migrer » une machine virtuelle et ses disques en cas de panne sur un hyperviseur. D'autres cluster-wares (comme Pacemaker, rgmanager) sont capables de basculer le disque et le service sur un serveur tiers du cluster.

L'un des points essentiel de cette architecture est de s'assurer que le disque ne peut être disponible que sur un seul serveur à la fois à tout instant de façon excessivement stricte. La fiabilité des données en dépend.

1.3 SHARE NOTHING

- Plusieurs nœuds indépendants
  - *ie* plusieurs instances PostgreSQL
  - en réplication
- Notamment : Patroni

L'autre architecture de clustering est appelée *Share Nothing*. Dans une telle architecture, chaque nœud du cluster est totalement indépendant.

Dans le cadre de PostgreSQL, cela signifie que les données sont situées sur plusieurs serveurs grâce à la réplication interne de ce dernier. Les solutions que nous proposons par la suite sont toutes de type *Share Nothing*.

1.4 PATRONI

- Fiable : SDB + *watchdog*
- Ne gère que PostgreSQL
  - bascule
  - (re)construction
- 2 nœuds PostgreSQL ou plus par cluster Patroni
- Simple mais formation nécessaire
- 1 cluster DCS de 3 nœuds est recommandé
  - etcd (ou autre)
  - un DCS peut gérer N clusters Patroni
  - +watchdog

Patroni assure l'exploitation d'un cluster PostgreSQL en réplication et est capable d'effectuer une bascule automatique en cas d'incident sur l'instance primaire.

Le projet repose sur un DCS externe comme stockage distribué de sa configuration. etcd[1], basé sur le protocole Raft, est sans doute le plus simple, mais d'autres sont possibles.

Le DCS nécessite au moins 3 nœuds (donc 3 serveurs ou machines virtuelles) pour assurer son quorum propre, la sécurité et la viabilité de ses données. Néanmoins un même cluster DCS peut ensuite gérer plusieurs clusters Patroni.

Se reposer sur un DCS fiable permet à Patroni plus de robustesse et de se focaliser sur l'expertise PostgreSQL uniquement. Le mécanisme de modification atomique de la base de configuration distribuée permet notamment une élection fiable en cas d'incident.

Depuis la version 2.0, Patroni permet aussi d'utiliser le protocole Raft, sans utiliser de DCS externe. Nous n'avons aucun recul sur cette technologie pour le moment. Nous utiliserons Patroni avec etcd par la suite.

En plus de ce DCS fiable et d'un mécanisme d'élection ne laissant aucune place aux *race conditions*, Patroni supporte l'utilisation d'un *watchdog* matériel. Le couple DCS+*watchdog* permet donc d'éviter les situations de *split-brain*.

Depuis la version 2.0, Patroni offre la possibilité d'exécuter une action pré-promotion, capable d'annuler la promotion si nécessaire. Cette fonctionnalité a été ajoutée afin de permettre l'ajout d'une action de *fencing* de l'ancien primaire. Nous n'avons pour le moment aucun recul sur cette fonctionnalité. Il faut notamment étudier quelles informations sont accessibles depuis le callback pour décider d'un *fencing* ou non.

[1] https://etcd.io/

Patroni est un projet simple, rapide à déployer et prendre en main. Il est toutefois toujours hautement recommandé de former une équipe à son administration.

1.5 PACEMAKER

- Référence de la HA sous Linux
- Principaux contributeurs : RedHat et Suse
- Empaqueté et cohérent sur toutes les distributions principales
- Super fiable : quorum, *watchdog* et *fencing* supportés
- Gère PostgreSQL et tout autre ressource
- Cluster deux nœuds possible avec *fencing*

Pacemaker est la solution de haute disponibilité de référence sur les distributions Linux modernes. Plusieurs entreprises telle que RedHat, Suse ou Linbit investissent du temps de recherche et développement pour la maintenance et l'évolution du projet. RedHat et Suse supportent officiellement les clusters Pacemaker (souvent au travers de souscriptions complémentaires) en imposant certaines contraintes de robustesse à leurs clients (comme un *fencing* obligatoire).

Le projet est très complet, supporte plusieurs types de *fencing*, avec escalade possible, le quorum, l'utilisation de *watchdog*, le clustering étendu (déployé entre deux datacenters). Il est capable de mettre en haute disponibilité plusieurs dizaines de services tels que des bases de données, des serveurs HTTP, mail, des machines virtuelles, des containers, etc.

Cette souplesse a néanmoins un prix en matière d'apprentissage et de maintenance. Il est largement recommandé d'être formé à l'utilisation et la maintenance de Pacemaker avant de le mettre en production.

Concernant la gestion de PostgreSQL, nous recommandons l'utilisation de l'agent PAF[2] (*PostgreSQL Automatic Failover*). Son développement a été lancé au sein de Dalibo, et il est maintenu actuellement par Jehan-Guillaume de Rorthais. afin de corriger ou contourner les problèmes et limitations de l'agent officiel existant. Il est entré au sein de l'organisme ClusterLabs regroupant les différents projets liés à Pacemaker.

[2] https://clusterlabs.github.io/PAF/

1.6 ACCÈS AUX RESSOURCES

Comment l'application accède-t-elle à la bonne instance ?

- IP virtuelle
- Proxy
  - HAProxy & autres
- Intelligence dans l'application
  - ex: `target_session_attrs=read-write`

Il existe plusieurs solutions pour gérer l'accès des applications à l'instance principale.

La plus simple est l'utilisation d'une IP virtuelle. Cette solution est à privilégier avec Pacemaker, ce dernier gérant alors à la fois la localisation de l'IP virtuelle et de l'instance primaire. Cette solution est aussi envisageable avec Patroni, à condition d'utiliser un script en callback ou d'ajouter un service supplémentaire dédié à cette tâche. Cette solution n'est cependant pas totalement robuste avec Patroni.

Une autre solution revient à utiliser un tiers faisant office de proxy entre les clients et l'instance principale. HAProxy est une solution populaire, mais il en existe bien d'autres, propriétaires ou non. La solution doit être capable de déterminer où se trouve l'instance primaire au sein du cluster.

Enfin, la dernière solution consiste à intégrer l'intelligence nécessaire directement dans les couches applicatives. La chaîne de connexion à l'instance peut par exemple comporter plusieurs destinataires et désigner le rôle recherché (`target_session_attrs=read-write`). Il est aussi possible d'utiliser un gestionnaire de configuration centralisé aux applications tel que `confd`.

1.7 COMPARATIF DES SOLUTIONS ET CHOIX

Solution	RTO	RPO	Qui	Complexité	Serveurs
PITR	humain	> 1 min	DBA	1	1+
Réplication	humain	< 1 min	DBA	2	2+
Shared disk	< 1mn	0-1 min	SYS	5	2+
Patroni	> 5s	0-1 min	DBA	4	5+
PAF	> 5s	0-1 min	DBA/SYS	5	3 (2+)

- Humain = tâche effectué / déclenché manuellement
- Nombre de Serveurs dédiés aux sauvegardes et HAProxy nom comptés

1.7.1 Le jeu en vaut-il la chandelle ?

- Complexité et coût
- Souvent plus sûr de rester sur des procédures :
  - simples
  - manuelles
  - éprouvées
  - maîtrisées
- Patroni ou Pacemaker/PAF
  - reconnus
  - éprouvés
  - éviter le reste

La mise en œuvre de la haute disponibilité apporte un certain nombre de complications et complexifications à l'architecture. Avec une disponibilité exigée de 99,5% (soit 44 h d'indisponibilité par an !), nous déconseillons la mise en œuvre d'une telle architecture. Il est plus sage de reposer sur des procédures connues et éprouvées, outillées, avec une phase de prise de décision humaine optimisée pour être déclenchée le plus vite possible.

Une meilleure maîtrise de l'architecture évite souvent les appels d'astreintes impromptus et se substitue avantageusement à une bascule automatique beaucoup plus difficile à maîtriser et budgétiser.

Si une bascule automatisée est nécessaire, nous recommandons principalement l'utilisation de Patroni ou de Pacemaker/PAF, d'une part pour leur robustesse, d'autre part pour leur adoption importante au sein de la communauté.

Évitez les solutions exotiques, les systèmes de fichiers distribués ou les projets peu populaires ou peu maintenus qui vous exposent à des risques certains d'indisponibilité, voire de corruption (expériences plusieurs fois vécues au support Dalibo).

1.8 CONCLUSION

Points de vigilance

- La supervision
- Les sauvegardes
- La formation des équipes
- La documentation

Si vos instances sont critiques et nécessitent une haute disponibilité de service et/ou de données, rappelons qu'il est **vital** de porter un soin particulier à la **supervision** pour anticiper les incidents et aux **sauvegardes** pour pallier aux incidents majeurs.

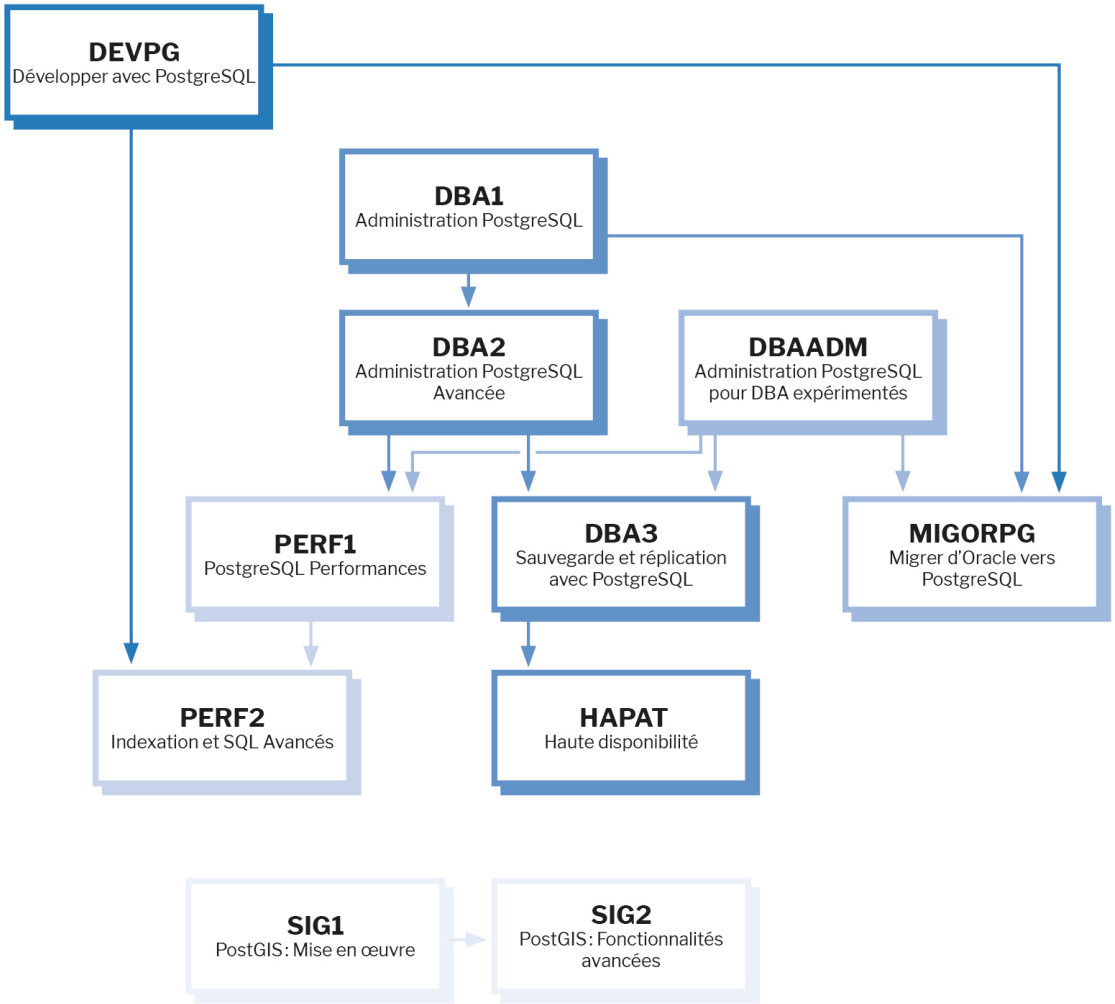
Enfin, la **formation des équipes** et **la documentation** à jour du montage utilisé sont indispensables.

Les formations Dalibo

Retrouvez nos formations et le calendrier sur https://dali.bo/formation

Pour toute information ou question, n'hésitez pas à nous écrire sur contact@dalibo.com.

Cursus des formations



Retrouvez nos formations dans leur dernière version :

- DBA1 : Administration PostgreSQL
  https://dali.bo/dba1
- DBA2 : Administration PostgreSQL avancé
  https://dali.bo/dba2
- DBA3 : Sauvegarde et réplication avec PostgreSQL
  https://dali.bo/dba3
- DEVPG : Développer avec PostgreSQL
  https://dali.bo/devpg
- PERF1 : PostgreSQL Performances
  https://dali.bo/perf1
- PERF2 : Indexation et SQL avancés
  https://dali.bo/perf2
- MIGORPG : Migrer d'Oracle à PostgreSQL
  https://dali.bo/migorpg
- HAPAT : Haute disponibilité avec PostgreSQL
  https://dali.bo/hapat

Les livres blancs

- Migrer d'Oracle à PostgreSQL
  https://dali.bo/dlb01
- Industrialiser PostgreSQL
  https://dali.bo/dlb02
- Bonnes pratiques de modélisation avec PostgreSQL
  https://dali.bo/dlb04
- Bonnes pratiques de développement avec PostgreSQL
  https://dali.bo/dlb05

Téléchargement gratuit

Les versions électroniques de nos publications sont disponibles gratuitement sous licence open source ou sous licence Creative Commons.