

Cluster PostgreSQL à stockage partagé



ioguix

true

Cluster PostgreSQL à stockage partagé

TITRE : Cluster PostgreSQL à stockage partagé

SOUS-TITRE :

INTRO

PROGRAMME

- présentation
- points d'attentions
- démo

PRÉSENTATION

Principe de fonctionnement des clusters à stockage partagé.

Deux slides: très simple à expliquer !

FONCTIONNEMENT 1/2

- un stockage accessible depuis plusieurs serveurs
- les données de l'instance sont placées dans ce stockage
- un seul à la fois serveur peut :
 - accéder au stockage
 - démarrer PostgreSQL

FONCTIONNEMENT 2/2

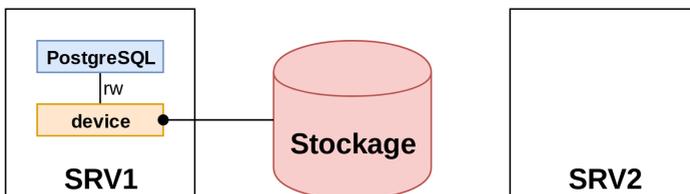
Cas d'une bascule sur incident:

- retirer l'accès au stockage à l'ancien serveur
- donner accès au stockage à un autre serveur
- démarrer PostgreSQL sur cet autre serveur

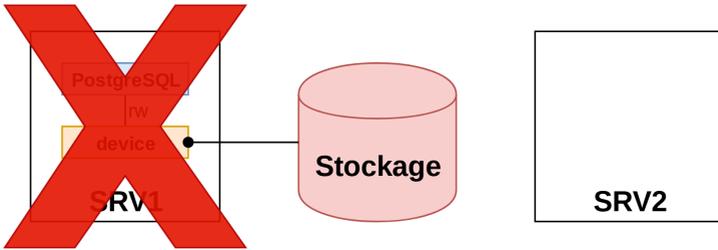
EXEMPLE

Cinématique simplifiée d'une bascule

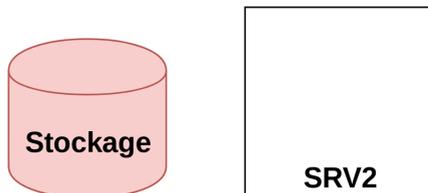
Cluster shared storage



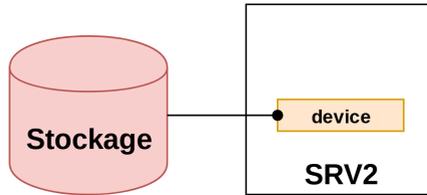
Cluster shared storage



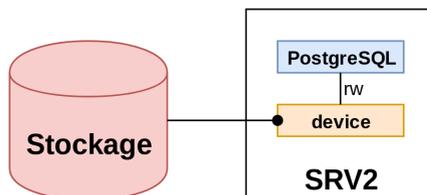
Cluster shared storage



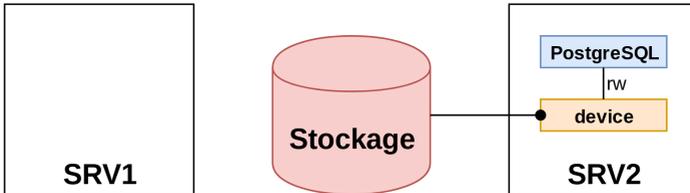
Cluster shared storage



Cluster shared storage



Cluster shared storage



BILAN

La HA, c'est pas si compliqué...

...

...mais c'est complexe.

POINTS D'ATTENTION

HA du stockage

Un SPoF évident: le stockage

...

Nécessite :

- un SAN répliqué
- un SAN en HA (multipath, RAID dans le SAN, etc)
- BONNE NOUVELLE: hors de notre périmètre !

RÉSERVATION DU STOCKAGE

Obligation d'empêcher plus d'un serveur à la fois à accéder aux données.

- le fencing nous protège déjà
- ...mais nous ne sommes jamais loin d'une erreur humaine
- solutions :
 - configuration SAN (eg. réservation)
 - LVM `systemid`

LVM SYSTEMID

Configuration du `systemid`:

```
root@srv1:~# cat /etc/lvm/lvmlocal.conf
global {
    system_id_source = "uname"
}
```

Voir `lvmsystemid(7)`.

EFFET DU SYSTEMID

```

root@srv1:~# vgchange -ay vg_san
  Cannot access VG vg_san with system ID srv2 with
  local system ID srv1.

root@srv1:~# vgchange --systemid srv1 vg_san
  Cannot access VG vg_san with system ID srv2
  local system ID srv1.

root@srv1:~# vgchange -y --systemid srv1 \
  --config 'local/extra_system_ids=["srv2"]' vg_san

```

FENCING

Plusieurs possibilités:

- fencing actif de serveur (droits système ?)
- coupure d'accès au SAN (droits SAN ?)
- sbd, watchdog et *poison pill* !

SBD

- partition dédiée sur un stockage partagé
- peut résider dans le même stockage que PostgreSQL
- redondance possible jusqu'à 3 disques
- quelques Mo pour des centaines de nœuds
- serveur en mode standby si plus d'accès au SBD
- reset watchdog si ça tourne mal
 - arrêt des ressources impossible
 - n'est plus capable de réagir assez vite...
- canal de communication entre les nœuds
 - self-fencing à la réception d'un *poison pill*

COLD STANDBY

Inacceptable ?

- mis en œuvre chez AWS:

The high-availability feature isn't a scaling solution for read-only scenarios ; you can't use a standby replica to serve read traffic.

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZ.html>

- possibilité de placer les applications sur les serveurs «dormants»

AVANTAGES

- hyper simple à mettre en œuvre
 - géré par les équipes système et SAN
 - pas besoin de réplication PostgreSQL
- indépendant de la configuration de PostgreSQL
- possibilité d'ajout des secondaires triviale

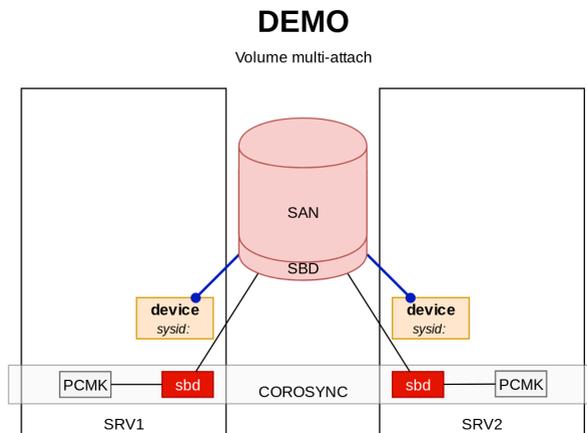
DÉSAVANTAGE

- Nécessite un stockage redondé et en HA

DÉMO

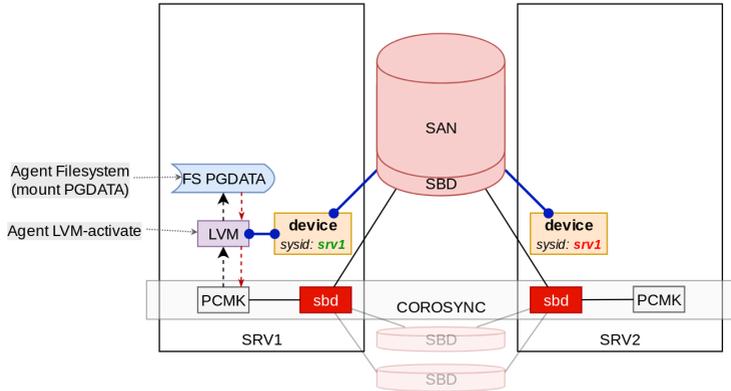
BRIQUES

- Debian 11
- Pacemaker avec `pcs`
- pas de fencing actif: SBD + watchdog
- stockage disponible sur tous les nœuds
 - utilisation de LVM `systemid`
 - 2 partitions: SBD et PostgreSQL
- lien de `/etc/postgresql/12/main` vers le SAN
- une IP suivant le positionnement de PostgreSQL
- une ressource `dummy` symbolisant une application
- vagrant disponible:
<https://github.com/ioguix/vagrant-pgsql-sbd>



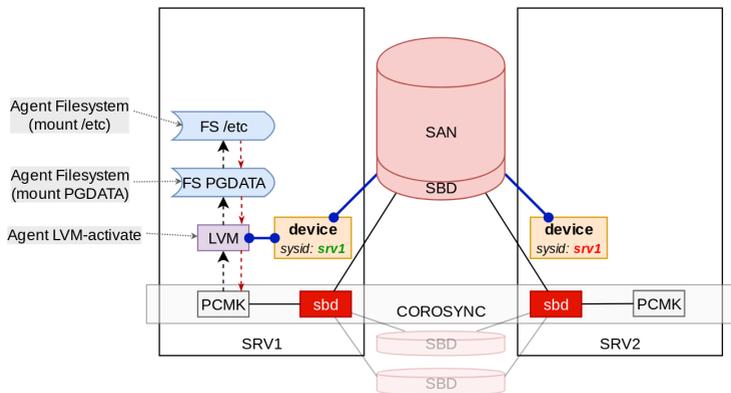
DEMO

Volume multi-attach



DEMO

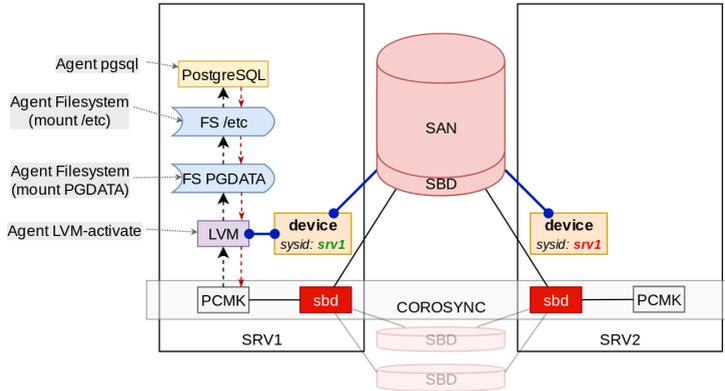
Volume multi-attach



Cluster PostgreSQL à stockage partagé

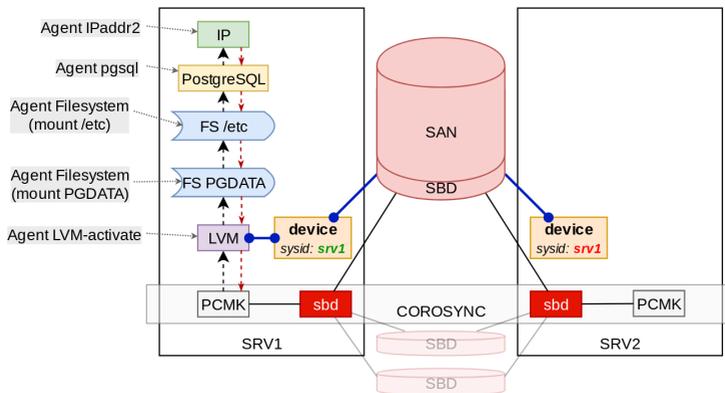
DEMO

Volume multi-attach



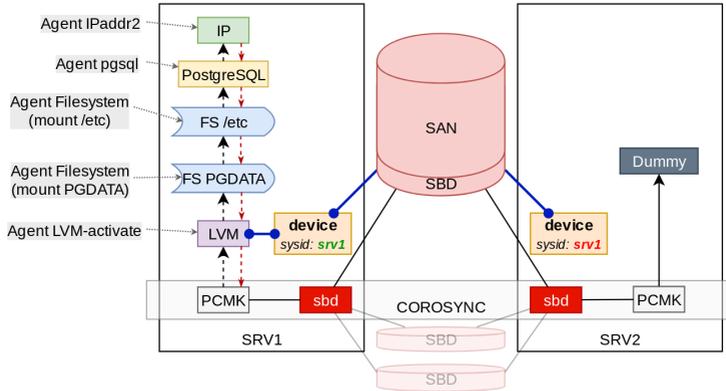
DEMO

Volume multi-attach



DEMO

Volume multi-attach



ACTIONS

- démarrage
- bascule
- standby
- boom

FIN

QUESTIONS