Atelier interne

HA Linux et PostgreSQL

DALIBO
L'expertise PostgreSQL

0.1

HA Linux et PostgreSQL

Atelier interne

TITRE : HA Linux et PostgreSQL
SOUS-TITRE : Atelier interne

REVISION: 0.1
DATE: 24 septembre 2020

Table des Matières

INSTALLATION

PRÉREQUIS

- Trois VM CentOS 7 ;
- Noms d'hôte ajustés (utilisé par le cluster) ;
- Fichiers `/etc/hosts` à jour ;
- Connexion `root` par ssh sans mot de passe entre les nœuds (optionnel).

Changer le nom des serveurs en `hanode1`, `hanode2` et `hanode3`. Voici la commande pour `hanode1`:

```
hostnamectl set-hostname hanode1
```

Ces noms sont utilisés par le cluster et par PAF pour désigné chaque nœud et standby.

Pour accélérer et simplifier la résolution de nom, mettre à jour le fichier `/etc/hosts` sur l'ensemble des serveurs. Eg. :

```
cat << _EOF_ >> /etc/hosts
10.20.60.50 hanode1
10.20.60.51 hanode2
10.20.60.52 hanode3
_EOF_
```

Optionnellement, activer le firewall et autoriser postgres et la haute disponibilité :

```
systemctl --quiet --now enable firewalld
firewall-cmd --quiet --permanent --add-service=high-availability
firewall-cmd --quiet --permanent --add-service=postgresql
firewall-cmd --quiet --reload
```

Optionnellement, échanger les clés pour permettre de se connecter sans mot de passe avec `root`. Nous verrons dans la partie administration du document que ça peut être utile pour simplifier la collecte d'information sur l'état d'un cluster.

INSTALLATION DES PAQUETS

- Paquets essentiels :
  - corosync : communication entre les nœuds ;
  - pacemaker : orchestration du cluster ;
- pcs : administration simplifiée du cluster.

```
yum install -y pacemaker pcs
```

Les éléments essentiels à l'installation d'un cluster HA Linux sont :

- corosync : communication entre les nœuds
- pacemaker : orchestration du cluster

L'outil pcs va également être installé, car il facilite beaucoup la configuration et l'administration du cluster.

Pour installer pacemaker et pcs, lancer la commande suivante sur tous les nœuds :

```
$ yum install -y pacemaker pcs
```

Cette commande installe un grand nombre de dépendances de pacemaker, dont `corosync` et les agents de gestion des ressources. Le paquet `pacemaker-cli` est également installé et contient les commandes `crm_*` pour interagir avec le cluster.

Pour la simplicité de cet atelier, nous n'installons pas d'agent de fencing. Le cluster créé repose uniquement sur le quorum et le self-fencing grâce au watchdog.

PCSD

- Activer au démarrage et démarrer `pcsd` ;
- Configurer le mot de passe de l'utilisateur `hacluster` ;
- Authentifier tous les nœuds entre eux ; ~~bash pcs cluster auth <nodes 1..n> -u ~~
- Initialiser le cluster ; ~~bash pcs cluster setup --name <nodes 1..n>~~

Démarrer le démon `pcsd` sur chaque nœud :

```
systemctl enable pcsd
systemctl start pcsd
```

Configurer un mot de passe pour l'utilisateur `hacluster` sur chaque nœud :

```
echo "h@L1nuxD3mo" |  passwd --stdin hacluster
```

Authentifier les membres du cluster entre eux depuis un des nœuds avec `pcs` :

```
pcs cluster auth hanode1 hanode2 hanode3 -u hacluster -p "h@L1nuxD3mo"
```

La commande doit retourner:

```
hanode1: Authorized
hanode2: Authorized
hanode3: Authorized
```

Initialiser le cluster depuis un des nœuds :

```
pcs cluster setup --name cluster_tp hanode1 hanode2 hanode3
```

La commande retourne:

```
Destroying cluster on nodes: hanode1, hanode2, hanode3...
hanode1: Stopping Cluster (pacemaker)...
hanode3: Stopping Cluster (pacemaker)...
hanode2: Stopping Cluster (pacemaker)...
hanode1: Successfully destroyed cluster
hanode2: Successfully destroyed cluster
hanode3: Successfully destroyed cluster

Sending 'pacemaker_remote authkey' to 'hanode1', 'hanode2', 'hanode3'
hanode1: successful distribution of the file 'pacemaker_remote authkey'
hanode2: successful distribution of the file 'pacemaker_remote authkey'
hanode3: successful distribution of the file 'pacemaker_remote authkey'
Sending cluster config files to the nodes...
hanode1: Succeeded
hanode2: Succeeded
hanode3: Succeeded

Synchronizing pcsd certificates on nodes hanode1, hanode2, hanode3...
hanode1: Success
hanode2: Success
hanode3: Success
Restarting pcsd on the nodes in order to reload the certificates...
hanode1: Success
hanode2: Success
hanode3: Success
```

Les fichiers de configuration sont créés et synchronisés par `pcsd`. Ces fichiers doivent **toujours** être identiques sur tous les nœuds. Dans la mesure du possible, ne les éditez pas manuellement. Vérifier que la configuration de corosync est identique partout en lançant sur chaque nœud :

```
md5sum /etc/corosync/corosync.conf
```

Note : Pour activer les traces, il est possible de modifier le paramètre `PCMK_debug` du fichier `/etc/syscondig/pacemaker`. Elles seront écrites dans `/var/log/cluster/corosync.log`.

DÉMARRAGE DU CLUSTER

- Désactiver corosync et pacemaker au démarrage
- Démarrer le cluster
- Commandes pcs :

```
pcs cluster disable  --all
pcs cluster start --all
pcs status --full
```

Il est déconseillé de démarrer automatiquement Corosync et Pacemaker au démarrage du serveur afin de maîtriser pleinement l'instant où un nœud isolé est réintroduit dans le cluster.

Désactiver Corosync et Pacemaker au démarrage du serveur en lançant la commande suivante sur un des nœuds :

```
$ pcs cluster disable --all

hanode1: Cluster Disabled
hanode2: Cluster Disabled
hanode3: Cluster Disabled
```

Note : il est aussi possible d'utiliser `systemctl disable corosync pacemaker` sur chaque nœud.

Démarrer le cluster :

```
$ pcs cluster start --all
hanode1: Starting Cluster (corosync)...
hanode2: Starting Cluster (corosync)...
hanode3: Starting Cluster (corosync)...
hanode1: Starting Cluster (pacemaker)...
hanode2: Starting Cluster (pacemaker)...
hanode3: Starting Cluster (pacemaker)...
```

Vérifier le statut du cluster :

```
$ pcs status --full
Cluster name: cluster_tp

WARNINGS:
No stonith devices and stonith-enabled is not false

Stack: corosync
Current DC: hanode3 (3) (version 1.1.21-4.el7-f14e36fd43) - partition with quorum
Last updated: Wed Sep 16 14:35:12 2020
Last change: Wed Sep 16 14:35:10 2020 by hacluster via crmd on hanode3

3 nodes configured
0 resources configured

Online: [ hanode1 (1) hanode2 (2) hanode3 (3) ]

No resources


Node Attributes:
* Node hanode1 (1):
* Node hanode2 (2):
* Node hanode3 (3):

Migration Summary:
* Node hanode1 (1):
* Node hanode3 (3):
* Node hanode2 (2):

Fencing History:

PCSD Status:
  hanode1: Online
  hanode2: Online
  hanode3: Online

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
```

```
pcsd: active/enabled
```

Note : il est aussi possible d'utiliser `systemctl status pacemaker corosync`.

Ajouter les paramètres suivants à la configuration :

- `resource-stickiness` : ajoute un score supplémentaire aux ressources sur le nœud où elles se situent. Cela évite qu'une ressource ne se déplace entre des nœuds qui ont le même score.
- `migration-threshold` : contrôle le nombre de fois que le cluster va essayer de ré-animer une ressource sur un même nœud avant de la déplacer vers une autre.

```
pcs resource defaults migration-threshold=5
pcs resource defaults resource-stickiness=10
```

Note : la configuration du cluster peut être exportée sous forme de commandes `pcs` avec la commande: `pcs config export pcs-commands`.

INSTALLATION DE POSTGRESQL

- Installer PostgreSQL ;
- Mettre en place la réplication :
  - répliquer depuis une vIP ;
  - empêcher la réplication d'un nœud sur lui-même ;
  - `application_name` **doit** contenir le hostname du serveur ;
  - pas de slots de réplication ;
- Stopper PostgreSQL partout un fois tout en œuvre de marche.

L'installation de PostgreSQL et de la réplication se fait classiquement.

Quelques pré-requis et conseils à respecter :

- il faut empêcher un nœud de répliquer sur lui-même dans le `pg_hba.conf`. Ce n'est pas un pré-requis de PAF mais une conséquence du mode d'accès au primaire via une vIP. Le cluster démarre toujours les ressources de type *promotable* en mode standby, avant de faire un vote pour élire un primaire. Si pour une raison ou une autre la vIP est toujours disponible, une instance pourrait répliquer vers elle-même. De même lors d'un demote par exemple.
- PAF **impose** que le paramètre `application_name` du `primary_conninfo` soit valorisé avec le hostname de l'hôte. Cela lui permet de reconnaître les nœuds en réplication dans `pg_stat_replication` lors de ses contrôles. PAF vérifie ce paramétrage et lève une erreur de configuration si cette règle n'est pas respectée, menant (par défaut) à l'isolation du nœud fautif.

- les slots de réplication ne sont pas supportés, car ils peuvent être un risque pour le service en cas de décrochage prolongé d'une standby. Cela pourrait changer avec PostgreSQL 13 et l'ajout du paramètre GUC `max_slot_wal_keep_size`.

Sur les trois serveurs :

```
yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg
yum install -y postgresql12-server
```

Créer et configurer l'instance sur le serveur primaire :

```
/usr/pgsql-12/bin/postgresql-12-setup initdb

cat << _EOF_ >> /var/lib/pgsql/12/data/postgresql.conf
listen_addresses = '*'
primary_conninfo = 'host=10.20.60.55 port=5432 user=repli application_name=hanode1'
wal_keep_segments = 100
_EOF_

cat << _EOF_ >> /var/lib/pgsql/12/data/pg_hba.conf
host    replication     all             $(hostname -s)       reject
host    replication     all             10.20.60.55/32       reject
host    replication     all             127.0.0.0/8          reject
host    replication     all             10.20.60.50/32          md5
host    replication     all             10.20.60.51/32          md5
host    replication     all             10.20.60.52/32          md5
host    all             all             10.0.0.0/8              md5
_EOF_

systemctl start postgresql-12

sudo -iu postgres createuser --login --pwprompt --replication repli

cat << _EOF_ > ~postgres/.pgpass
10.20.60.55:5432:replication:repli:pwdrepli
_EOF_

chown postgres: ~postgres/.pgpass
chmod 600 ~postgres/.pgpass

cp ~postgres/12/data/pg_hba.conf /var/lib/pgsql/12/
```

```
cp ~postgres/12/data/postgresql.conf /var/lib/pgsql/12/
```

Note: il est recommandé de placer tout paramètre propre à chaque instance à l'extérieur du PGDATA afin de simplifier les étapes de reconstruction d'une instance (ici primary_conninfo et le fichier pg_hba.conf)

Créer une vIP sur le serveur primaire (attention à l'interface utilisée):

```
ip addr add 10.20.60.55/32 dev eth1
```

Sur les serveurs secondaires :

```
cat << _EOF_ > ~postgres/.pgpass
10.20.60.55:5432:replication:repli:pwdrepli
_EOF_

chown postgres: ~postgres/.pgpass
chmod 600 ~postgres/.pgpass
sudo -iu postgres  pg_basebackup --host 10.20.60.55              \
                                 --username repli                \
                                 --pgdata /var/lib/pgsql/12/data \
                                 --progress

sed -e "s/\(primary_conninfo = .* application_name=\).*'/\1$(hostname -s)'/" \
    -i /var/lib/pgsql/12/data/postgresql.conf

sed -e "s/\(hanode1\)\(\s*reject\)/$(hostname -s)\2/" \
    -i /var/lib/pgsql/12/data/pg_hba.conf

sudo -iu postgres touch /var/lib/pgsql/12/data/standby.signal

sudo -iu postgres cp ~postgres/12/data/pg_hba.conf /var/lib/pgsql/12/
sudo -iu postgres cp ~postgres/12/data/postgresql.conf /var/lib/pgsql/12/

systemctl start postgresql-12
```

Sur le serveur primaire :

```
sudo -iu postgres psql -xc "TABLE pg_stat_replication"
```

Arrêter PostgreSQL sur les trois serveurs, de préférence en commençant par la primaire, et s'assurer que le service n'est pas activé au démarrage du serveur :

```
systemctl stop postgresql-12
systemctl disable postgresql-12
```

Supprimer la vIP sur le serveur primaire :

```
ip addr del 10.20.60.55/32 dev eth1
```

FENCING

- Utilisation d'un watchtdog ;
- Compte à rebours avant l'arrêt du serveur, réinitialisé par Pacemaker aussi longtemps que possible ;
- Permet également d'isoler un autre nœud en cas de problème.

Pour la simplicité de cet atelier, nous créons un cluster reposant sur le quorum et le self-fencing grâce au watchdog. Le cluster obtenu est fiable. Il est néanmoins recommandé de configurer au moins un agent de fencing actif afin de rendre le cluster plus réactif.

Vérifier la présence d'un watchdog pré-existant. Eg.:

```
$ dmesg | grep Dog
[    4.090358] i6300esb: Intel 6300ESB WatchDog Timer Driver v0.05

# et/ou

$ ls /dev/watchdog*
```

Si aucun device watchdog n'existe, chargez le module `softdog` manuellement et configurer le système pour qu'il le fasse au démarrage:

```
modprobe softdog
echo softdog > /etc/modules-load.d/watchdog.conf
```

Note: il est recommandé d'utiliser un watchdog matériel et non un watchdog dépendant de la disponibilité et stabilité du système d'exploitation et du matériel.

Le watchdog doit être armé avant que le cluster soit démarré. Ce denrier est géré par la daemon `sbd`.

Stopper le cluster :

```
pcs cluster stop --all
```

Installer `sbd` («storage based death») et activer le service :

```
yum install -y sbd
pcs stonith sbd enable
```

Note: la commande pcs peut être remplacée par : `systemctl enable sbd`

Vérifier la configuration de sbd :

```
$ grep "SBD_PACEMAKER\|SBD_WATCHDOG_DEV" /etc/sysconfig/sbd
SBD_PACEMAKER=yes
SBD_WATCHDOG_DEV=/dev/watchdog
```

Redémarrer le cluster :

```
pcs cluster start --all
```

Consulter le statut et les propriétés :

```
$ pcs status  | tail -n 6

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
  sbd: active/enabled

$ pcs property | grep have-watchdog
 have-watchdog: true
```

Modifier la propriété qui gère le timeout du watchdog :

```
pcs property set stonith-watchdog-timeout=10s
```

Test : `sbd` ne ré-arme par le watchdog à temps :

```
killall -9 sbd
```

Test : isoler un autre nœud

```
pcs stonith fence hanode3
```

Note : alternativement, il est possible d'utiliser la commande `stonith_admin -F hanode2`.

Redémarrer le clusterware sur les noeuds qui ont été isolés avant de poursuivre.

RESOURCE POSTGRESQL

- Installer l'agent PAF ;
- validation hors-ligne de la configuration Pacemaker avant application ;
- Ressources pour PostgreSQL, gestion des rôles `Master/Slave` et vIP ;
- Contrainte de localisation de la vIP avec l'instance primaire ;
- Contraintes d'ordre :
  - Ajouter la vIP après avoir démarré l'instance primaire ;
  - Retirer la vIP après avoir arrêté l'instance primaire.

Installer l'agent PAF (disponible sur le dépôt PGDG) sur tous les nœuds :

```
yum install -y resource-agents-paf
```

Note: les paquets sont aussi disponibles sur le dépôt git de PAF : git

Effectuer les modifications dans un fichier `pgsqld.xml` avant de les appliquer au cluster.

```
pcs cluster cib pgsqld.xml
```

1. créer une ressource `pgsqld`

```
pcs -f pgsqld.xml resource create pgsqld ocf:heartbeat:pgsqlms \
    bindir=/usr/pgsql-12/bin pgdata=/var/lib/pgsql/12/data       \
    op start timeout=60s                                         \
    op stop timeout=60s                                          \
    op promote timeout=30s                                       \
    op demote timeout=120s                                       \
    op monitor interval=15s timeout=10s role="Master"            \
    op monitor interval=16s timeout=10s role="Slave"             \
    op notify timeout=60s
```

Nous précisons le timeout de chaque action. L'intervalle de vérification des états au sein du cluster est de 15s pour le primaire, 16s pour le secondaire.

2. créer la ressource `pgsqld-clone` responsable des clones de `pgsqld`

```
pcs -f pgsqld.xml resource master pgsqld-clone pgsqld notify=true
```

Nous ne précisons ici que l'option `notify` qui est **essentielle** à PAF et dont la valeur par défaut est `false`. Elle permet d'activer les actions `notify` avant et après chaque action sur la ressource. Nous laissons les autres options à leur valeur par défaut.

3. créer la ressource `pgsql-prim-ip` gérant l'adresse `10.20.60.55` sur l'interface réseau eth1 :

```
pcs -f pgsqld.xml resource create pgsql-prim-ip ocf:heartbeat:IPaddr2 \
    ip=10.20.60.55 cidr_netmask=24                                    \
    op monitor interval=10s
```

4. ajouter une colocation obligatoire entre l'instance primaire de `pgsqld-clone` et `pgsql-prim-ip`

```
pcs -f pgsqld.xml constraint \
    colocation add pgsql-prim-ip with master pgsqld-clone INFINITY
```

5. ajouter une contrainte asymétrique pour promouvoir `pgsqld-clone` avant de démarrer `pgsql-prim-ip`

```
pcs -f pgsqld.xml constraint order promote pgsqld-clone \
    then start pgsql-prim-ip symmetrical=false kind=Mandatory
```

6. ajouter une contrainte asymétrique pour rétrograder `pgsqld-clone` avant d'arrêter `pgsql-prim-ip`

```
pcs -f pgsqld.xml constraint order demote pgsqld-clone \
    then stop pgsql-prim-ip symmetrical=false kind=Mandatory
```

7. vérifier la configuration créée

```
pcs cluster verify -V pgsqld.xml
crm_simulate -S -x pgsqld.xml
```

Ce qui devrait indiquer:

```
Current cluster status:
Online: [ hanode1 hanode2 hanode3 ]

 Master/Slave Set: pgsqld-clone [pgsqld]
     Stopped: [ hanode1 hanode2 hanode3 ]
 pgsql-prim-ip  (ocf::heartbeat:IPaddr2):   Stopped

Transition Summary:
 * Start      pgsqld:0     ( hanode1 )
 * Start      pgsqld:1     ( hanode2 )
 * Start      pgsqld:2     ( hanode3 )

Executing cluster transition:
 * Resource action: pgsqld:0        monitor on hanode1
 * Resource action: pgsqld:1        monitor on hanode2
 * Resource action: pgsqld:2        monitor on hanode3
```

```
 * Pseudo action:   pgsqld-clone_pre_notify_start_0
 * Resource action: pgsql-prim-ip monitor on hanode3
 * Resource action: pgsql-prim-ip monitor on hanode2
 * Resource action: pgsql-prim-ip monitor on hanode1
 * Pseudo action:   pgsqld-clone_confirmed-pre_notify_start_0
 * Pseudo action:   pgsqld-clone_start_0
 * Resource action: pgsqld:0        start on hanode1
 * Resource action: pgsqld:1        start on hanode2
 * Resource action: pgsqld:2        start on hanode3
 * Pseudo action:   pgsqld-clone_running_0
 * Pseudo action:   pgsqld-clone_post_notify_running_0
 * Resource action: pgsqld:0        notify on hanode1
 * Resource action: pgsqld:1        notify on hanode2
 * Resource action: pgsqld:2        notify on hanode3
 * Pseudo action:   pgsqld-clone_confirmed-post_notify_running_0
 * Resource action: pgsqld:0        monitor=16000 on hanode1
 * Resource action: pgsqld:1        monitor=16000 on hanode2
 * Resource action: pgsqld:2        monitor=16000 on hanode3

Revised cluster status:
Online: [ hanode1 hanode2 hanode3 ]

 Master/Slave Set: pgsqld-clone [pgsqld]
     Slaves: [ hanode1 hanode2 hanode3 ]
 pgsql-prim-ip  (ocf::heartbeat:IPaddr2):   Stopped
```

Nous pouvons maintenant appliquer ces modifications au cluster:

```
pcs cluster cib-push pgsqld.xml
```

MAINTENANCE

NOTIONS IMPORTANTES

- CIB ;
- Scores ;
- DC (Designated Controller).

Pacemaker dispose d'une configuration qui décrit l'état souhaité du cluster. L'état des ressources est obtenu grâce à l'action `monitor` des agents et aux scores. Une ressource est démarrée sur le nœud où elle obtient le score le plus important. Les scores sont calculés dynamiquement en tenant compte de différentes contraintes: `stickiness`, `master_score`, score de localisation, de colocation, et autres règles dynamiques.

Les informations de configuration et de scores sont stockées dans un fichier XML, la CIB (Cluster Information Base). Il est présent et historisé sur tous les nœuds.

Si le résultat de ces actions est en contradiction avec l'état souhaité du cluster, pacemaker essaie de remettre le cluster dans un état compatible avec la configuration. Si une ressource est dans un état inconnu, il essaie de la redémarrer, puis de l'arrêter, avant d'isoler le nœud qui l'abrite en dernier recours.

Les décisions et actions sont calculées par le processus `pengine`. Ce processus est interrogé depuis un seul nœud, le DC (Designated Controller), qui orchestre alors l'exécution des actions. C'est dans ses logs que l'on trouvera les transitions créées par le cluster pour changer son état.

- les décisions du cluster sont tracées sur le DC ;
- les scores et l'état des ressources sont la base pour comprendre les décisions du cluster et l'effet de certaines commandes ;
- la configuration et l'état du cluster à un instant T sont écrits dans la CIB.

ETAT DU CLUSTER

- `pcs status --full`

```
$ pcs status --full

Cluster name: cluster_tp
Stack: corosync
Current DC: hanode3 (3) (version 1.1.21-4.el7-f14e36fd43) - partition with quorum
Last updated: Fri Sep 18 09:05:20 2020
Last change: Fri Sep 18 08:40:59 2020 by root via cibadmin on hanode1

3 nodes configured
4 resources configured

Online: [ hanode1 (1) hanode2 (2) hanode3 (3) ]

Full list of resources:

 Master/Slave Set: pgsqld-clone [pgsqld]
     pgsqld (ocf::heartbeat:pgsqlms):   Slave hanode3
     pgsqld (ocf::heartbeat:pgsqlms):   Slave hanode1
     pgsqld (ocf::heartbeat:pgsqlms):   Master hanode2
     Masters: [ hanode2 ]
     Slaves: [ hanode1 hanode3 ]
 pgsql-prim-ip  (ocf::heartbeat:IPaddr2):   Started hanode2

Node Attributes:
* Node hanode1 (1):
    + master-pgsqld                    : 1000
* Node hanode2 (2):
    + master-pgsqld                    : 1001
* Node hanode3 (3):
    + master-pgsqld                    : 990

Migration Summary:
* Node hanode3 (3):
* Node hanode1 (1):
* Node hanode2 (2):
```

```
Failed Resource Actions:

Failed Fencing Actions:

Fencing History:

PCSD Status:
  hanode1: Online
  hanode2: Online
  hanode3: Online

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
  sbd: active/enabled
```

Informations importantes :

- `Current DC : hanode2 (2)` : le DC (Designated Controller) est le nœud qui décide des transitions à effectuer au cours de la vie du cluster. C'est lui qui exécute le `pengine` et c'est dans les traces de ce nœud qu'il faut chercher pour trouver ;
- `Online: [ hanode1 (1) hanode2 (2) hanode3 (3) ]` : liste des nœuds disponibles ;
- statut et emplacement des ressources ;
- attributs `master-pgsqld` : C'est le score utilisé par PAF pour déterminer sur quel nœud la ressource va être promue :
  - 1 : est le score lors du premier démarrage avant passage du `monitor` ;
  - 1001 : est le score du primaire ;
  - -1000 : est le score réservé aux instances non connectées à la primaire, cela peut arriver brièvement suite à un changement de statut.

SWITCHOVER

- Deux solutions :
  - désigner un nœud cible ;
  - interdire la primaire de rester sur le nœud où elle est placée.
- Penser à nettoyer les contraintes temporaires.

La première méthode consiste à désigner la cible où placer la ressource `pgsql-clone`.

```
pcs resource move --master pgsqld-clone hanode3
```

Pour réaliser cette bascule, pacemaker doit faire en sorte que le score de la ressource `pgsql-clone:master` ait une valeur supérieure sur le nœud cible par rapport aux autres nœuds. Pour arriver à ce résultat, pacemaker pose une contrainte de localisation sur le nœud cible avec un score `+INFINITY`.

```
$ pcs constraint location show resource pgsqld-clone
Location Constraints:
  Resource: pgsqld-clone
    Enabled on: hanode3 (score:INFINITY) (role: Master)

$ crm_simulate -sL|grep 'promotion score'
pgsqld:1 promotion score on hanode3: INFINITY
pgsqld:0 promotion score on hanode1: 1000
pgsqld:2 promotion score on hanode2: 990
```

Il faut penser à retirer cette contrainte de localisation, sinon elle risque d'être oubliée et de provoquer des bascules inattendues à l'avenir.

```
$ pcs resource clear pgsqld-clone
$ crm_simulate -sL|grep 'promotion score'
pgsqld:1 promotion score on hanode3: 1021
pgsqld:0 promotion score on hanode1: 1000
pgsqld:2 promotion score on hanode2: 990
```

La seconde méthode consiste à interdire la primaire de rester sur le nœud ou elle est placée.

```
$ pcs resource ban pgsqld-clone --master
Warning: Creating location constraint cli-ban-pgsqld-clone-on-hanode3
with a score of -INFINITY for resource pgsqld-clone on node hanode3.
This will prevent pgsqld-clone from being promoted on hanode3 until the
constraint is removed. This will be the case even if hanode3 is the
last node in the cluster.
```

Pour réaliser cette bascule, pacemaker doit faire en sorte que le score de la ressource `pgsql-clone:master` soit inférieur sur le nœud où elle est placée par rapport à celui de tous les autres nœuds. Pour arriver à ce résultat, pacemaker pose une contrainte de localisation sur le nœud courant avec un score `-INFINITY`.

```
$ pcs constraint location show resource pgsqld-clone
Location Constraints:
  Resource: pgsqld-clone
    Disabled on: hanode3 (score:-INFINITY) (role: Master)
$ crm_simulate -sL|grep 'promotion score'
pgsqld:0 promotion score on hanode1: 1021
pgsqld:2 promotion score on hanode2: 1000
pgsqld:1 promotion score on hanode3: -INFINITY
```

Il faut penser à retirer cette contrainte de localisation, sinon elle risque d'être oubliée et de provoquer des bascules inattendues à l'avenir.

```
$ pcs resource clear pgsqld-clone
$ crm_simulate -sL|grep 'promotion score'
pgsqld:0 promotion score on hanode1: 1021
pgsqld:2 promotion score on hanode2: 1000
pgsqld:1 promotion score on hanode3: 990
```

OBTENIR ET LIRE LES TRACES

- Deux solutions :
  - `crm_report` (nécessite un ssh `root` entre les nœuds).
  - manuellement :
    - trace : `/var/log/messages` et `/var/log/cluster/corosync.log` ;
    - toutes les cib et diff : `/var/lib/pacemaker/cib` et `/var/lib/pacemaker/pengine`.
- Méthode :
  - Trouver le DC (Designated Controler) ;
  - Identifier la transition ;
  - Identifier l'ID des CIB utiles et les analyser.

Pacemaker dispose de l'outil `crm_report` permettant de regrouper l'ensemble des traces nécessaires pour faire une analyse.

```
$ crm_report -f "2020-09-17 08:25:00" -t "2020-09-17 09:00:00" -d mon_rapport
[...]
$  ls -al mon_rapport
```

```
total 76
drwxr-xr-x. 5 root root   141 Sep 17 09:11 .
dr-xr-x---. 5 root root   238 Sep 17 09:11 ..
-rw-r--r--. 1 root root  1693 Sep 17 09:11 analysis.txt
-rw-r--r--. 1 root root 46835 Sep 17 09:11 collector
-rw-r--r--. 1 root root  1154 Sep 17 09:11 crm_mon.txt
-rw-r--r--. 1 root root   276 Sep 17 09:11 .env
drwxr-xr-x. 2 root root  4096 Sep 17 09:14 hanode1
drwxr-xr-x. 3 root root  4096 Sep 17 09:17 hanode2
drwxr-xr-x. 2 root root  4096 Sep 17 09:11 hanode3
-rw-r--r--. 1 root root    93 Sep 17 09:11 report.summary
```

Note : sans le -d, l'outil génère une archive compressée.

Le contenu du fichier crm_mon.txt permet de connaître le DC.

```
$ grep "Current DC" crm_mon.txt
Current DC: hanode2 (version 1.1.21-4.el7-f14e36fd43) - partition with quorum
```

Il faut donc chercher dans les traces du nœud hanode2.

```
$ ls -al mon_rapport/hanode2
total 1256
drwxr-xr-x. 3 root root   4096 Sep 17 09:17 .
drwxr-xr-x. 5 root root    141 Sep 17 09:35 ..
-rw-r--r--. 1 root root     34 Sep 17 09:11 analysis.txt
-rw-r--r--. 1 root root   3390 Sep 17 09:11 cib.txt
-rw-------. 1 root root   8097 Sep 17 09:11 cib.xml
-rw-------. 1 root root  24871 Sep 17 09:11 cib.xml.live
lrwxrwxrwx. 1 root root     20 Sep 17 09:11 cluster-log.txt -> messages.extract.txt
-rw-r--r--. 1 root root 658534 Sep 17 09:11 corosync-blackbox-live.txt
-rw-------. 1 root root    437 Sep 17 09:11 corosync.conf
-rw-r--r--. 1 root root  15239 Sep 17 09:11 corosync.dump
-rw-r--r--. 1 root root    545 Sep 17 09:11 corosync.quorum
lrwxrwxrwx. 1 root root     14 Sep 17 09:11 crm_mon.txt -> ../crm_mon.txt
-rw-r--r--. 1 root root      8 Sep 17 09:11 DC
-rw-r--r--. 1 root root      0 Sep 17 09:11 events.txt
-rw-r--r--. 1 root root 438264 Sep 17 09:11 journal.log
-rw-r--r--. 1 root root     25 Sep 17 09:11 members.txt
-rw-------. 1 root root  32079 Sep 17 09:11 messages.extract.txt
drwxr-xr-x. 2 root root    253 Sep 17 09:16 pengine
-rw-r--r--. 1 root root     33 Sep 17 09:11 permissions.txt
```

```
-rw-r--r--. 1 root root   6750 Sep 17 09:11 report.out
-rw-r--r--. 1 root root      8 Sep 17 09:11 RUNNING
-rw-r--r--. 1 root root  20964 Sep 17 09:11 sysinfo.txt
-rw-r--r--. 1 root root  30971 Sep 17 09:11 sysstats.txt
```

Chercher la transition qui nous intéresse dans : `messages.extract.txt`

```
Sep 17 08:58:44 hanode2

pengine[4149]: warning: Processing failed probe of pgsqld:0 on hanode3: unknown error
pengine[4149]:  notice: If it is not possible for pgsqld:0 to run on hanode3,
+++ see the resource-discovery option for location constraints
pengine[4149]: warning: Processing failed probe of pgsqld:1 on hanode2: unknown error
pengine[4149]:  notice: If it is not possible for pgsqld:1 to run on hanode2, see
+++ the resource-discovery option for location constraints
pengine[4149]:  notice:  * Demote     pgsqld:1            (   Master -> Slave hanode2
pengine[4149]:  notice:  * Promote    pgsqld:2            (   Slave -> Master hanode1
pengine[4149]:  notice:  * Move       pgsql-prim-ip      (        hanode2 -> hanode1 )
pengine[4149]:  notice: Calculated transition 12, saving inputs in
+++ /var/lib/pacemaker/pengine/pe-input-63.bz2
```

Les traces nous apprennent que la ressource `pgsqld-clone:master` doit basculer du nœud `hanode2` vers le nœud `hanode1`. Elle est accompagnée par la vIP. L'état du cluster est décrit dans l'extraction de la cib appelée `pe-input-63.bz2`. Ce fichier est présent dans le rapport généré dans le répertoire `hanode2/pengine`

```
$ bzip2 -d pe-input-63.bz2
$ crm_simulate -Ss -x pe-input-63 | grep "promotion score"
pgsqld:2 promotion score on hanode1: 1000
pgsqld:0 promotion score on hanode3: 990
pgsqld:1 promotion score on hanode2: -INFINITY
```

Il apparaît que le score de la ressource `pgsqld` est `-INFINITY`. Il y a une contrainte de localisation de bannissement sur `hanode2`.

```
$ grep -e "pgsqld-clone.*hanode2.*score" pe-input-63
<rsc_location id="cli-ban-pgsqld-clone-on-hanode2" rsc="pgsqld-clone"
+++ role="Master" node="hanode2" score="-INFINITY"/>
```

REMONTER LE CLUSTER APRÈS UN FAILOVER

- Identifier la source du problème et le résoudre ;
- Sauvegarder l'instance si nécessaire ;
- Mettre la ressource PostgreSQL en mode *unmanaged* ;
- Remonter la réplication ;
- Stopper PostgreSQL ;
- Démarrer le cluster.

Faire planter l'instance primaire :

```
pkill -9 -F /var/lib/pgsql/12/data/postmaster.pid
```

Consulter le statut du cluster :

```
$ pcs status

Cluster name: cluster_tp
Stack: corosync
Current DC: hanode3 (version 1.1.21-4.el7-f14e36fd43) - partition with quorum
Last updated: Fri Sep 18 09:10:20 2020
Last change: Fri Sep 18 09:10:18 2020 by root via crm_attribute on hanode2

3 nodes configured
4 resources configured

Online: [ hanode1 hanode2 hanode3 ]

Full list of resources:

 Master/Slave Set: pgsqld-clone [pgsqld]
     Masters: [ hanode2 ]
     Slaves: [ hanode1 hanode3 ]
 pgsql-prim-ip  (ocf::heartbeat:IPaddr2):   Started hanode2

Failed Resource Actions:
* pgsqld_monitor_15000 on hanode2 'master (failed)' (9): call=22,
+++ status=complete, exitreason='Instance "pgsqld" controldata indicates a
+++ running primary instance, the instance has probably crashed',
    last-rc-change='Fri Sep 18 09:10:10 2020', queued=0ms, exec=0ms

Failed Fencing Actions:
```

```
Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
  sbd: active/enabled
```

Après quelques secondes, la ressource pgsqld a été relancée sur le nœud `hanode2`. Un message concernant cette erreur apparaît dans la rubrique *Failed Resource Actions*.

Consulter le failcount de la ressource `pgsqld` et la valeur du paramètre `migration-threshold`.

```
$ pcs resource failcount show pgsqld

Failcounts for resource 'pgsqld'
  hanode2: 1

$ pcs resource defaults | grep migration-threshold

migration-threshold=5
```

Le compteur d'erreur de la ressource `pgsqld` a été incrémenté de un. Une fois atteint un nombre d'échec égal à `migration-threshold`, la ressource est arrêtée et basculée ailleurs. Si l'arrêt n'est pas possible le nœud est isolé.

La valeur du compteur d'erreur n'est pas remise à zéro par le cluster, sauf à configurer le paramètre `failure-timeout`. Il est cependant préférable de superviser le cluster, d'analyser l'origine du problème et réinitialiser le compteur manuellement (`pcs resource refresh <nom> --node <nœud>`). Dans tous les cas, ne pas laisser le compteur en l'état, c'est une bombe à retardement.

Baisser le `migration-threshold` à 2 spécifiquement pour la ressource `pgsqld` (surcharge la valeur par défaut positionnée précédemment à cinq) et tuer une nouvelle fois le processus `postmaster`.

```
pcs resource update pgsqld meta migration-threshold=2
pkill -9 -F /var/lib/pgsql/12/data/postmaster.pid
```

Consulter le statut du cluster :

```
Cluster name: cluster_tp
Stack: corosync
Current DC: hanode3 (version 1.1.21-4.el7-f14e36fd43) - partition with quorum
Last updated: Fri Sep 18 09:15:53 2020
Last change: Fri Sep 18 09:15:45 2020 by root via crm_attribute on hanode1
```

```
3 nodes configured
4 resources configured

Online: [ hanode1 hanode3 ]
OFFLINE: [ hanode2 ]

Full list of resources:

 Master/Slave Set: pgsqld-clone [pgsqld]
      Masters: [ hanode1 ]
      Slaves: [ hanode3 ]
      Stopped: [ hanode2 ]
 pgsql-prim-ip  (ocf::heartbeat:IPaddr2):       Started hanode1

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
  sbd: active/enabled
```

On voit que la ressource `pgsqld` est stoppée sur le nœud `hanode2` (le noeud a été isolé) et a été promue sur `hanode1`.

Passer la ressource en mode non maintenu :

```
$ pcs resource unmanage pgsqld-clone
$ pcs resource show pgsqld-clone
 Master: pgsqld-clone
  Meta Attrs: notify=true
  Resource: pgsqld (class=ocf provider=heartbeat type=pgsqlms)
   Attributes: bindir=/usr/pgsql-12/bin pgdata=/var/lib/pgsql/12/data
   Meta Attrs: is-managed=false migration-threshold=2
   Operations: demote interval=0s timeout=120s (pgsqld-demote-interval-0s)
               methods interval=0s timeout=5 (pgsqld-methods-interval-0s)
               monitor interval=15s role=Master timeout=10s (pgsqld-monitor-interval-1
               monitor interval=16s role=Slave timeout=10s (pgsqld-monitor-interval-16
               notify interval=0s timeout=60s (pgsqld-notify-interval-0s)
               promote interval=0s timeout=30s (pgsqld-promote-interval-0s)
               reload interval=0s timeout=20 (pgsqld-reload-interval-0s)
               start interval=0s timeout=60s (pgsqld-start-interval-0s)
```

```
                stop interval=0s timeout=60s (pgsqld-stop-interval-0s)
```

Le méta-attribut `is-managed` de `pgsqld` est positionné à `false`. Le cluster ne réagira plus aux événements sur la ressource.

Recréer la standby :

```
cp /var/lib/pgsql/12/data/*.conf /var/lib/pgsql/12
rm -Rf /var/lib/pgsql/12/data/*
sudo -iu postgres  pg_basebackup --host 10.20.60.55 \
                                 --username repli \
                                 --pgdata /var/lib/pgsql/12/data \
                                 --progress
sudo -iu postgres touch /var/lib/pgsql/12/data/standby.signal

cp /var/lib/pgsql/12/pg_hba.conf /var/lib/pgsql/12/data
cp /var/lib/pgsql/12/postgresql.conf /var/lib/pgsql/12/data

systemctl start postgresql-12
systemctl status postgresql-12
```

Vérifier l'état de la réplication et stopper PostgreSQL :

```
systemctl stop postgresql-12
```

Démarrer le cluster sur le nœud qui a été isolé :

```
pcs cluster start
```

Nettoyer le compteur d'erreur de la ressource `pgsqld` sur le nœud qui avait planté (ici `hanode2`) :

```
$ pcs resource refresh pgsqld --node=hanode2
Cleaned up pgsqld:0 on hanode2
Cleaned up pgsqld:1 on hanode2
Cleaned up pgsqld:2 on hanode2

  * Configuration prevents cluster from stopping or starting unmanaged 'pgsqld-clone'
..Waiting for 1 reply from the CRMd. OK
```

Supprimer le méta-attribut `is-managed` et afficher le statut du cluster :

```
$ pcs resource manage pgsqld-clone
$ pcs status

Cluster name: cluster_tp
```

```
Stack: corosync
Current DC: hanode3 (version 1.1.21-4.el7-f14e36fd43) - partition with quorum
Last updated: Fri Sep 18 09:23:38 2020
Last change: Fri Sep 18 09:23:36 2020 by root via crm_attribute on hanode1

3 nodes configured
4 resources configured

Online: [ hanode1 hanode2 hanode3 ]

Full list of resources:

 Master/Slave Set: pgsqld-clone [pgsqld]
     Masters: [ hanode1 ]
     Slaves: [ hanode2 hanode3 ]
 pgsql-prim-ip  (ocf::heartbeat:IPaddr2):   Started hanode1

Failed Resource Actions:

Failed Fencing Actions:

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
  sbd: active/enabled
```

SORTIR POSTGRESQL DU CLUSTER

- Passer les ressources suivantes au statut *unmanaged* :
  - PostgreSQL ;
  - vIP ;
- Arrêter Pacemaker si nécessaire ;
- Contrôler le statut de la vIP et de la réplication nœuds.

Passer la ressource PostgreSQL et la VIP au statut non maintenu par le cluster.

```
pcs resource unmanage pgsqld
pcs resource unmanage pgsql-prim-ip
```

Vérifier le statut des ressources :

```
$ pcs status

Cluster name: cluster_tp
Stack: corosync
Current DC: hanode3 (version 1.1.21-4.el7-f14e36fd43) - partition with quorum
Last updated: Fri Sep 18 09:25:35 2020
Last change: Fri Sep 18 09:25:16 2020 by root via cibadmin on hanode2

3 nodes configured
4 resources configured

Online: [ hanode1 hanode2 hanode3 ]

Full list of resources:

 Master/Slave Set: pgsqld-clone [pgsqld]
     pgsqld (ocf::heartbeat:pgsqlms):   Slave hanode3 (unmanaged)
     pgsqld (ocf::heartbeat:pgsqlms):   Master hanode1 (unmanaged)
     pgsqld (ocf::heartbeat:pgsqlms):   Slave hanode2 (unmanaged)
 pgsql-prim-ip  (ocf::heartbeat:IPaddr2):   Started hanode1 (unmanaged)

Failed Resource Actions:

Failed Fencing Actions:

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
  sbd: active/enabled
```

Stopper Pacemaker si nécessaire (eg. aucune autre ressource):

```
pcs cluster stop --all
```

Vérification des ressources :

```
$ ps -fu postgres
UID        PID  PPID  C STIME TTY          TIME CMD
postgres  4160     1  0 13:28 ?        00:00:00 /usr/pgsql-12/bin/postgres -D /var/lib
```

```
postgres  4161  4160  0 13:28 ?        00:00:00 postgres: logger
postgres  4163  4160  0 13:28 ?        00:00:00 postgres: checkpointer
postgres  4164  4160  0 13:28 ?        00:00:00 postgres: background writer
postgres  4165  4160  0 13:28 ?        00:00:00 postgres: stats collector
postgres  6232  4160  0 13:44 ?        00:00:00 postgres: walwriter
postgres  6233  4160  0 13:44 ?        00:00:00 postgres: autovacuum launcher
postgres  6234  4160  0 13:44 ?        00:00:00 postgres: logical replication launcher
postgres  9554  4160  0 13:51 ?        00:00:00 postgres: walsender repli 10.20.60.52(
postgres 11631  4160  0 13:56 ?        00:00:00 postgres: walsender repli 10.20.60.50(

$ ip add | grep 10.20.60.55
    inet 10.20.60.55/24 brd 10.20.60.255 scope global secondary eth1
```