Haute disponibilité d'un serveur FTP

Activité 2 – Intégration de la solution dans le cluster

L'objectif est ici d'automatiser la bascule sur le serveur esclave en cas de défaillance du serveur maître ainsi que le *failback*.

Les principes de l'intégration de la solution au cluster sont les suivants :

- Une ressource pour chacun des services FTP et DRBD doit être créée.
- Le service FTP sera activé/désactivé par Pacemaker.
- Le service DRBD sera activé sur les deux nœuds mais avec un rôle différent (primary/secondary – voir rappel ci-dessous) avec une préférence en tant que primaire pour le nœud *intralabXX* .
- Lorsque le service DRBD démarre sur le nœud qui joue le rôle de maître, ce dernier doit être en primary et monter le disque virtuel sur /home ==> il faut déclarer une nouvelle ressource de type « système de fichier » (*ocf:heartbeat:Filesystem).*
- La ressource FTP devra être activée sur le même nœud que l'adresse IP virtuelle et la ressource Web. De même, cette ressource ne devra être activée que sur le nœud sur lequel DRBD est lancé en tant que maître et après que le système de fichier soit monté.
- Le service DRDB doit démarrer avant celui de FTP.

La ressource DRBD doit donc être configurée en mode actif/actif avec la fonction "clone" qui, rappelons-le, admet 3 types :

- Le clone « **anonymous » (clone par défaut)** : les ressources sont configurées de manière identique sur chaque nœud. Si un nœud est indisponible, il n'y a pas de bascule de cette ressource puisqu'elle est déjà fonctionnelle et identique sur l'autre nœud.
- Le clone « **globally-unique=true »** : contrairement au cas précédent, les ressources ne sont pas identiques d'un nœud à l'autre. Elles peuvent par exemple, avoir des données différentes.
- Le « **multi-state »** : c'est une ressource qui peut être dans 2 états différents : « master » ou « slave ». **C'est ce type de clone que nous allons utiliser.**

Vous disposez de la documentation suivante :

- **document 1** : création et configuration des ressources DRBD et Système de fichiers ;
- **document 2** : création de la ressource relative au service FTP ;
- **document 3** : note relative au split brain ;.
- **document 4** : récapitulatif des commandes

Sur chacun des deux serveurs, la partition */dev/sda2* est celle que DRBD duplique (dans l'exemple). Comme nous faisons le choix (arbitraire) d'enregistrer les métadonnées sur cette partition, elle ne doit pas être formatée (ou si elle l'est, elle devra être remise à zéro et toutes les données seront perdues).
Il est possible d'utiliser une autre partition (non formatée ou remise à zéro) pour les métadonnées.

Travail à faire

1. Configuration des ressources DRBD et FileSystem (aide dans le document 1)

Q1. Créez les ressources DRBD et FileSystem.

```
root@intralabXXintralabXX:~# crm configure
crm(live)configure# primitive drbd0 ocf:linbit:drbd params drbd_resource=resftp op monitor role=Master interval=60s timeout=30s op monitor role=Slave interval=60s timeout=30s
crm(live)configure# ms ms-drbd0 drbd0 meta clone-max=2 clone-node-max=1 master-max=1 master-node-max=1 globally-unique=false notify=true
```

```
crm(live)configure# primitive fs0 ocf:heartbeat:Filesystem params fstype=ext4 directory=/home device=/dev/drbd0
crm(live)configure# colocation fs0-on-ms-drbd0 inf: fs0 ms-drbd0:Master
crm(live)configure# order ms-drbd0-before-fs0 mandatory: ms-drbd0:promote fs0:start
crm(live)configure# location ms-drbd0-master-on-intralabar ms-drbd0 rule role=master inf: #uname eq intralabar
crm(live)configure# commit
```

Q2. Tester et complétez le tableau de tests permettant de vérifier que la solution est opérationnelle.

Actions à effectuer	Résultats attendus	Résultats obtenus	Statut
Sur le serveur maître, vérifier que les ressources sont bien activées (crm status, ifconfig, ps, tests à partir de Filezilla)	Ressources activées	Ressources activées	OK
Tester l'accès en écriture sur la partition (touch /home/user/Documents/essai1)	« Home » accessible	« Home » accessible	OK
Mettre le serveur maître en « standby » (maintenance)	Les ressources sont arrêtées sur le serveur maître et activées sur le serveur esclave (crm status)	Les ressources migrent bien sur le serveur esclave	OK
Vérifier sur le serveur esclave la présence du fichier « essai1 »	essai1 a bien été dupliqué	essai1 dupliqué	OK
Tester sur le serveur esclave l'accès en écriture sur la partition (touch /home/user/Documents/essai2)	« Home » accessible	« Home » accessible	OK
Mettre le serveur maître « online »	Les ressources sont de nouveau activées sur le serveur maître (auto-failback)	Les ressources sont de nouveau activées sur le serveur maître	OK
Vérifier sur le serveur maître la présence du fichier « essai2 »	essai2 a bien été dupliqué	essai2 dupliqué	OK

2. Configuration de la ressource FTP

Lors des différents tests, notamment s'ils portent sur l'arrêt d'une ressource mise en erreur :

- ne pas oublier de supprimer toutes les erreurs via la commande : ***crm resource cleanup <id_resource>*** ;
- vérifier l'ensemble des ressources par la commande : ***crm_resource -P***

Q3. Créez la ressource « serviceFTP ». On pourra éventuellement surveiller cette ressource toutes les 30s.

```
root@intralabXX:~# crm configure primitive serviceFTP lsb:proftpd op monitor interval=30s
```

Q4. Procédez aux ajustements nécessaires, compte tenu notamment des contraintes de positionnement de la ressource et d'ordre de démarrage (elle ne devra être activée que sur le nœud sur lequel DRBD est lancé en tant que maître et après que le système de fichier soit monté).

root@intralabXX:~# crm configure
crm(live)configure#colocation serviceFTP-on-ms-drbd0 inf: serviceFTP ms-drbd0:Master
crm(live)configure#order serviceFTP-after-fs0 inf: fs0 serviceFTP
crm(live)configure#commit

De manière à ce que le service FTP soit activé sur le même serveur que les ressources IPFailover et serviceWeb, il suffit d'intégrer la ressource serviceFTP au "group" "servIntralab créé dans le coté Labo précédent.

Pour cela il est possible de passer par la commande :
root@intralabXX:~# crm configure edit servIntralab
Puis se mettre en mode insertion (saisir "i") et ajouter "serviceFTP" à la ligne "group servIntralab IPFailover serviceWeb", puis enregistrer et quitter (via ":wq").

Q5. Testez et complétez le tableau de test permettant de vérifier que la solution est opérationnelle.

Les tests doivent se faire à partir d'un client FTP sur l'adresse IP virtuelle pour vérifier l'écriture dans la partition dupliquée.

Actions à effectuer	Résultats attendus	Résultats obtenus	Statut
Sur le serveur maître, vérifier que les ressources sont bien activées (crm status, ifconfig, ps, tests à partir de Filezilla).	Ressources activées.	Ressources activées.	OK
Mettre le serveur maître en « standby ».	Les ressources sont arrêtées sur le serveur maître et activées sur le serveur esclave (crm status).	Les ressources migrent bien sur le serveur esclave.	OK
Tester l'accès en écriture au serveur FTP.	Tests à partir de Filezilla concluants.	Tests à partir de Filezilla concluants.	OK
Mettre le serveur maître « online ».	Les ressources sont de nouveau activées sur le serveur maître (auto-failback).	Les ressources sont de nouveau activées sur le serveur maître.	OK
Tester l'accès en écriture au serveur FTP.	Tests à partir de Filezilla concluants.	Tests à partir de Filezilla concluants.	OK

Remarque : d'autres tests peuvent être réalisés comme :
- ➔ le redémarrage du nœud maître ;
- ➔ l'arrêt d'un service (avec empêchement et sans empêchement de redémarrage) ;
- ➔ la déconnexion d'un câble réseau.

3. Sauvegarde et restauration

Q6. Sauvegardez votre configuration

crm configure save /root/sauveha.conf

Q7. Testez une restauration

À partir de « crm configure edit », supprimer des ressources et/ou des contraintes puis tester une restauration/mise à jour (voire, après arrêt de corosync, déplacer ou renommer le fichier cib.xml).

crm configure load update /root/sauveha.conf

À noter que la configuration global devrait ressembler à cela : root@intralabXX:~# crm configure show

```
node 169221774: intralabXX \
        attributes standby=off
node 169221777: hdintralabXX
primitive IPFailover IPaddr2 \
        params ip=10.22.100.220 cidr_netmask=16 nic=eth0 iflabel=VIP \
        op monitor interval=10s timeout=20s \
        op start timeout=30s interval=0 \
        op stop timeout=40s interval=0
primitive drbd0 ocf:linbit:drbd \
        params drbd_resource=resftp \
        op monitor role=Master interval=60s timeout=30s \
        op monitor role=Slave interval=60s timeout=30s
primitive fs0 Filesystem \
        params fstype=ext4 directory="/home" device="/dev/drbd0"
primitive serviceFTP lsb:proftpd \
        op monitor interval=10s
primitive serviceMySQL mysql \
        params socket="/var/run/mysqld/mysqld.sock"
primitive serviceWeb apache \
        params configfile="/etc/apache2/apache2.conf" \
        op start timeout=40s interval=0 \
        op stop timeout=60s interval=0 \
        op monitor interval=60s
group servIntralab IPFailover serviceWeb serviceFTP \
        meta migration-threshold=5
ms ms-drbd0 drbd0 \
         meta clone-max=2 clone-node-max=1 master-max=1 master-node-max=1 globally-unique=false
notify=true
clone cServiceMySQL serviceMySQL
location cli-prefer-IPFailover servIntralab role=Started inf: intralabXX
colocation fs0-on-ms-drbd0 inf: fs0 ms-drbd0:Master
order ms-drbd0-before-fs0 Mandatory: ms-drbd0:promote fs0:start
location ms-drbd0-master-on-intralabar ms-drbd0 \
        rule $role=master #uname eq intralabar
order serviceFTP-after-fs0 inf: fs0 serviceFTP
colocation serviceFTP-on-ms-drbd0 inf: serviceFTP ms-drbd0:Master
property cib-bootstrap-options: \
        have-watchdog=false \
        dc-version=1.1.14-70404b0 \
        cluster-infrastructure=corosync \
        cluster-name=debian \
        stonith-enabled=false \
        no-quorum-policy=ignore \
        last-lrm-refresh=1474674185
```

Documents

Document 1 : création et configuration des ressources DRBD et Système de fichiers

Il est préférable que les commandes qui suivent soient validées/activées en même temps car elles sont étroitement liées. On travaille donc en mode « live »

```
root@intralabXX:~# crm configure
```

Étape 1 : On crée tout d'abord une ressource appelée ***drbd0*** qui démarrera sur le maître et, sur l'esclave, la ressource *resftp* (définie dans le fichier */etc/drbd.d/ftp.res) :*

```
crm(live)configure# primitive drbd0 ocf:linbit:drbd params drbd_resource=resftp op monitor role=Master interval=60s timeout=30s op monitor role=Slave interval=60s timeout=30s
```

On remarque que le script "drbd" est ici disponible dans l'espace de nom "ocf:linbit" (et non "ocf:heartbeat" comme nous l'avons vu jusqu'à présent).

Étape 2 : La primitive ressource nommée ***drbd0*** est intégrée ici dans un objet plus complexe "ms" (pour master-slave) nommé ***ms-drbd0*** *:*

```
crm(live)configure# ms ms-drbd0 drbd0 meta clone-max=2 clone-node-max=1 master-max=1 master-node-max=1 globally-unique=false notify=true
```

Explication :

- clone-max=2 : il ne peut y avoir que 2 ressources démarrées en même temps (= nombre de nœuds) ;
- clone-node-max=1 : une ressource par nœud au maximum ;
- master-max=1 : une seule ressource peut être activée en tant que maître ;
- master-node-max=1 : une seule ressource sur le même nœud ;
- globally-unique=false : les ressources sont identiques d'un nœud à l'autre ;
- notify=true : le cluster informe les nœuds du changement d'état de chacune des ressources.

Étape 3 : on configure ensuite une ressource ***fs0*** qui permet de monter le système de fichiers qui contiendra les données partagées :

```
crm(live)configure# primitive fs0 ocf:heartbeat:Filesystem params fstype=ext4 directory=/home device=/dev/drbd0
```

Étape 4 : il est nécessaire maintenant de **gérer l'exécution des ressources** (préférence d'exécution, ordre dans lequel les ressources seront exécutées, etc). En effet, comme nous l'avons déjà observé, si les ressources ne sont pas groupées, elles vont être activées sur des nœuds différents de manière à optimiser le cluster.

Une alternative à la commande ***group*** (vue dans une précédente séance) est l'utilisation les contraintes ***colocation*** et ***order*** : le principe est le même, on va lier des ressources entre elles et créer un ordre de démarrage mais ces deux commandes permettent une gestion beaucoup plus fine des contraintes.

Trois types de contraintes sont disponibles :

- **contrainte locative avec la commande *location*** : définit une préférence ou une obligation d'exécution d'une ressource sur un nœud (c'est ce type de contrainte qui est écrit automatiquement lorsque l'on migre une ressource sur un nœud – voir première séance) ;
- **contrainte colocative avec la commande *colocation*** : oblige une ressource à fonctionner (ou pas) sur le même nœud qu'une autre ressource (l'obligation peut être une interdiction suivant la valeur de la pondération – voir ci-après) ;
- **contrainte d'ordre avec la commande *order*** : ordonne le démarrage et l'arrêt des ressources.

La syntaxe générale des commandes est la suivante :

location <id_contrainte> <id_ressource> [**rule role**=rôle] <score>: **#uname eq** <noeud>

colocation <id_contrainte> <score> : <id_ressource_dependante[:rôle]> <id_ressource_necessaire[:rôle]>

order <id_contrainte> <score> **:** <id_ressource1 :action1> <id_ressource2 :action2>

Avec

Rôle : started, slave ou master
action : start, stop, promote, demote (l'action1 est à réaliser sur la ressource 1 avant de réaliser l'action2 sur la ressource2)

Le cluster prend en compte **les préférences d'exécution** sur un nœud ou l'autre suivant un **mécanisme de pondération des ressources (*score*).** Le score peut avoir une valeur positive ou négative :

- *+ l'infini* (ou *mandatory*) : avec *location,* obligation d'exécution de la ressource sur le nœud (valeur notée *« inf »* dans les commandes) et avec *colocation*, obligation d'exécution des ressources ensemble.
- *Valeur comprise entre 0 et + l'infini* : préférence d'exécution de la ressource
- *Valeur comprise entre - l'infini et 0* : préférence de non-exécution de la ressource
- *Valeur = 0* correspond au mot clé *advisory*
- *- l'infinie* : avec *location,* interdiction d'exécution de la ressource sur le nœud (valeur notée « *-inf »* dans les commandes)

Il est possible de connaître la syntaxe d'une commande via la commande : ***crm configure help nom_commande*** (par exemple *crm configure help order*)

Le système de fichiers doit être monté sur le nœud où la ressource *ms-drbd0* est lancé en tant que maître, et seulement après que drbd0 ait été promu maître :

```
crm(live)configure# colocation fs0-on-ms-drbd0 inf: fs0 ms-drbd0:Master
```

La commande *colocation* permet donc de forcer deux (ou plus) ressources à cohabiter sur le même nœud : ici, fs0 et ms-drbd0 quand il est lancé en maître.

Contrairement à la commande *group*, cette dernière contrainte ne définit pas d'ordre de démarrage (et d'arrêt) des ressources. Il faut lui adjoindre la commande *order*.

Le système de fichiers doit être monté après que *ms-drbd0* ait été promu maître :

```
crm(live)configure# order ms-drbd0-before-fs0 mandatory: ms-drbd0:promote fs0:start
```

La commande *order* permet donc de définir un ordre dans les ressources. Ici, fs0 n'est démarré qu'une fois la ressource ms-drbd0 promue maître.

Les commandes *colocation* et *order* peuvent encore être affinées avec l'usage des parenthèses pour les ressources qui n'ont pas de liens entre elles.

Par exemple :
colocation id inf:(A B) C

Les ressources A et B sont indépendantes.
L'arrêt de la ressource A **ou** celle de la ressource B n'impactera pas la ressource C.
Pour que le ressource C s'arrête et migre avec A et/ou B, il est nécessaire que les ressources A et B s'arrêtent simultanément. Par contre, si la ressource C s'arrête, les ressources A et B seront coupées.

order id inf:(A B) C
Les ressources A et B peuvent démarrer en parallèle et C démarrera après le démarrage d'une des deux ressources.

Enfin, on déclare que ***ms-drbd0*** sera exécutée préférentiellement en tant que maître sur *intralabar* :

```
crm(live)configure# location ms-drbd0-master-on-intralabar ms-drbd0 rule role=master inf: #uname eq intralabar
```

Dans un cluster à deux nœuds une valeur de pondération (score) comprise entre 0 et + l'infini serait équivalente à la valeur « inf ». La donne change à partir de 3 nœuds car on pourra préférer créer un ordre de priorité pour chacun de nœuds (par exemple par rapport aux capacités physiques des serveurs).

On peut maintenant valider tout en étant vigilant sur les erreurs éventuelles :

```
crm(live)configure# commit
```

crm_mon :

```
Last updated: Sun Jul 25 14:24:54 2016          Last change: Sun Jul 24 14:23:35 2016 by root via cibadmin on intralabXX
Stack: corosync
Current DC: intralabXX (version 1.1.14-70404b0) - partition with quorum
2 nodes and 7 resources configured

Online: [ hdintralabXX intralabXX ]

 Resource Group: servIntralab
     IPFailover (ocf::heartbeat:IPaddr2):        Started intralabXX
     serviceHTTP        (ocf::heartbeat:apache):     Started intralabXX
 Clone Set: cServiceMySQL [serviceMySQL]
     Started: [ hdintralabXX intralabXX ]
 Master/Slave Set: ms-drbd0 [drbd0]
     Masters: [ intralabXX ]
     Slaves: [ hdintralabXX ]
fs0     (ocf::heartbeat:Filesystem):     Started intralabXX
```

Document 2 : création de la ressource relative au service FTP

La création de la ressource relative au service FTP pose problème avec le script "ocf". Lors d'une bascule, pacemaker considère que le service n'est pas démarré et crée des erreurs sur la ressource. Pour que ça fonctionne, il est nécessaire de redémarrer le service et de faire un cleanup sur la ressource avec la commande ***crm resource cleanup <id_resource>*** ;

Nous allons donc utiliser le script de démarrage "proftpd" de type LSB présent dans /etc/init.d.
Ce script est bien conforme à la spécification LSB (il répond au minimum *à une commande start, stop et status)* :
http://refspecs.linux-foundation.org/LSB_3.2.0/LSB-Core-generic/LSB-Core-generic/iniscrptact.html.

La commande est très simple :

```
crm configure primitive serviceFTP lsb:proftpd
```

Il est possible d'ajouter un certain nombre d'options (comme celles relatives à la surveillance des ressources et aux *timeout* de démarrage et d'arrêt).

La ressource FTP ne devra être activée que sur le nœud sur lequel DRBD est lancé en tant que maître et après que le système de fichier ne soit monté.

Document 3 : note sur le split brain

Un split brain se produit lorsque deux parties d'un cluster d'ordinateurs sont déconnectées, chaque partie croyant que l'autre ne fonctionne plus. Le terme est une analogie médicale du syndrome split-brain (littéralement « dédoublement du cerveau ») (source : http://fr.wikipedia.org/wiki/Split-brain).

Chaque nœud va alors démarrer les services (les deux nœuds se déclarent primary), ce qui va éventuellement provoquer une incohérence des données quand les différents nœuds montent et écrivent sur un système de fichiers de façon concurrente.
DRBD va détecter les deux primary au moment où la connectivité entre les deux nœuds est rétablie : le démon stoppe alors la connexion et arrête la réplication, ce qui est visible dans les logs :
Split-Brain detected, dropping connection!

Il est possible de configurer DRBD pour reprendre la main automatiquement et décider quelles données garder (instructions after-sb-*pri dans la section net) mais il est préférable d'intervenir manuellement afin d'estimer les « dégâts » et décider quel est le nœud dont les données sont les plus actuelles et quel est le nœud dont les données doivent être discréditées.

Comment éviter le split brain

- Assurer au maximum la liaison entre les éléments du cluster : on peut utiliser deux liens réseau différents ou faire du bonding – agrégation de liens.
- S'assurer que l'autre nœud est inactif : le Stonith : « Shoot The Other Node In The Head » est une méthode d'isolation d'un nœud qui ne répond plus. Le nœud jugé hors service sera donc, en général, arrêté électriquement (via IPMI – Interface de Gestion Intelligente du Matériel par exemple). *Nous ne le mettrons pas en œuvre dans ce Coté Labo mais ceci est indispensable en production.*

Comment résoudre le split brain ?

Il faut supprimer les données sur le nœud le moins actif et resynchroniser. En règle générale les commandes suivantes suffisent :
Sur les deux nœuds : *umount /dev/drbd0* et *drbdadm disconnect resftp*
Sur le nœud le moins actif : *drbdadm secondary resftp* ***et*** *drbdadm -- --discard-my-data connect resftp*
Sur le nœud à partir duquel on réplique les données : *drbdadm connect resftp*

À la fin de la synchronisation on monte DRBD : *mount /dev/drbd0 /home*

Document 4 : récapitulatif des commandes courantes

Vérifier que Corosync est lancé :
```
/etc/init.d/corosync status
```

Voir l'état d'un cluster :
```
crm_mon
```
ou
```
crm_mon -1
```
ou
```
crm status
```

Accéder à l'interface de configuration du Cluster :
```
crm
```

crm(live)# help

This is the CRM command line interface program.

Available commands:

cib	manage shadow CIBs
resource	resources management
node	nodes management
options	user preferences
configure	CRM Cluster configuration
ra	resource agents information center
status	show Cluster status
quit,bye,exit	exit the program
help	show help
end,cd,up	go back one level

Si on saisit ***crm(live)# nom_commande help***, on a les différentes possibilités d'arguments après la commande et ainsi de suite...

Éditer/Modifier une configuration
```
crm configure edit
```
cela vous placera dans une instance de « vim » pour effectuer une modification qui sera appliquée à la sauvegarde.

Éditer/Modifier directement une ressource
```
crm configure edit <id_ressource>
```
cela vous placera dans une instance de « vim » pour effectuer une modification qui sera appliquée à la sauvegarde.

Modifier la configuration du Cluster en entrant directement dans la mode de configuration :
```
crm configure edit
```

Voir la configuration
```
crm(config)configure# show
```
Équivalent de crm configure show

Vérifier sa configuration
```
crm(config)configure# verify
```
Équivalent de crm configure verify

Stopper une ressource
```
crm resource stop <id_ressource>
```

Démarrer une ressource
crm resource start <id_ressource>

Supprimer une ressource
crm configure delete <id_ressource>

Si la ressource est en cours d'utilisation, il vaut mieux la stopper avant de la supprimer.

Déplacer une ressource
crm resource move <id_ressource> <noeud>

Annuler le déplacement
crm resource unmove <id_ressource>

Préférence du noeud
crm configure location <nouvel_id_ressource> <id_ressource> <score>: <nœud>

La ressource <id_ressource> préférera tourner sur le nœud <nœud>.

Liste des ressources OCF
crm ra list ocf heartbeat

Connaître la liste complète des paramètres d'une primitive,
crm ra info ocf:heartbeat:nom_primitive.

Vous obtenez la liste des timeout par défaut, des paramètres et de leurs valeurs par défaut, les paramètres obligatoires et facultatifs.

Mettre un poste en maintenance
crm node standby [<nœud>]

Sortir un poste de maintenance
crm node online [<nœud>]

Effacer les erreurs sur une ressource
crm resource cleanup <id_resource>

Cloner une ressource (de type « anonymous »)
crm configure clone <nom_clone> <id_resource>

Sauvegarder une configuration
crm configure save /root/sauveha.conf
crm configure save xml /root/sauveha.conf.xml # Version XML

Restaurer une configuration (et remplacer la configuration actuelle)
crm configure load replace /root/sauveha.conf

Mettre à jour une configuration
crm configure load update /root/sauveha.conf