Haute disponibilité d'un service Web dynamique

Activité 2 - Configuration des ressources « IPFailover » et « serviceWeb ».

La haute disponibilité sous-entend que plusieurs machines seront utilisées pour répondre à un même service. Seulement, chaque machine a normalement une adresse IP différente sur le réseau, ce qui est problématique puisque le client ne connaît qu'une seule adresse IP et/ou le nom d'hôte pleinement qualifié (qui n'est associé qu'à une seule adresse IP).

Problématique : comment alors faire en sorte que toutes les machines en haute disponibilité répondent à la même adresse ?

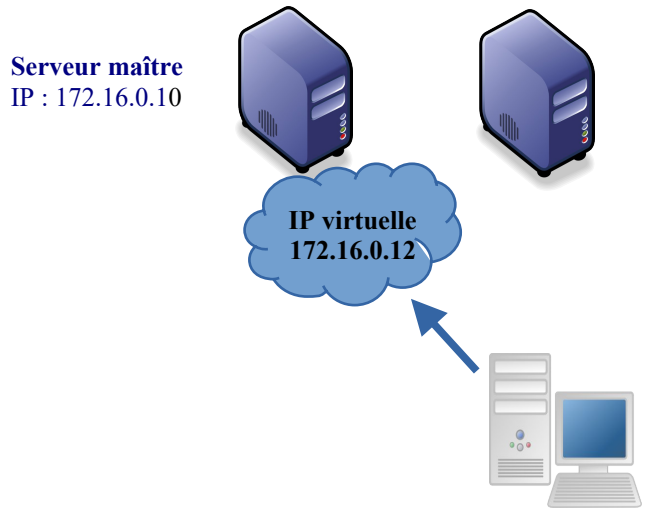
Il faut mettre en place une adresse IP virtuelle, c'est-à-dire une adresse IP qui ne sera pas toujours reliée à la même machine physique. L'adresse IP virtuelle est en fait attribuée au cluster c'est à dire au groupe de machines participant à la haute disponibilité. L'utilisateur ne connaîtra que cette adresse, et sera en fait redirigé vers l'un ou l'autre des serveurs par un mécanisme réseau.

Pour cela, plusieurs solutions sont possibles. Dans notre cas, nous n'utilisons que 2 machines en haute disponibilité (une active et une passive) ; une seule d'entre elles fonctionne pour répondre aux requêtes, on peut lui demander de reconnaître 2 adresses IP au lieu d'une : la sienne et l'adresse virtuelle. **Toute requête est adressée à l'adresse IP virtuelle.**

Lorsqu'elle tombe en panne, la machine passive prend la main, lance ses services et se met à répondre à son tour aux requêtes à destination de l'IP virtuelle.

Exemple d'un fonctionnement en cas de serveur maître disponible :

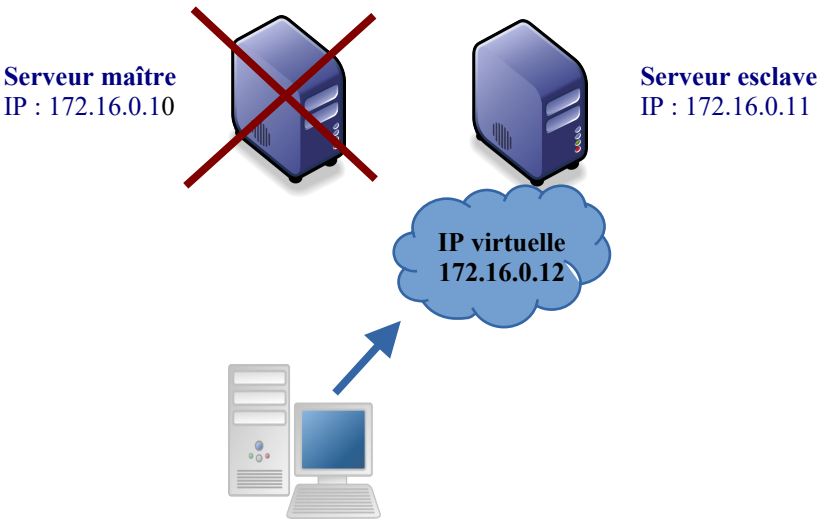
Lors de son lancement et après configuration, Corosync, via Pacemaker, activera l'IP virtuelle 172.16.0.12 pour le serveur considéré comme « actif » à un instant donné.



À noter qu'il n'y a pas que l'IP virtuelle qui sera activée sur un serveur et désactivée sur l'autre mais aussi les services (appelés ressources) que l'on a décidé de mettre en haute disponibilité (comme HTTP).

Si le serveur maître n'est plus disponible :

En cas de défaillance du serveur primaire, c'est le serveur secondaire qui sera accessible à l'adresse 172.16.0.12.



Lorsque la première machine est restaurée, deux options peuvent être envisagées :

- elle pourra faire à son tour office de secours ;
- elle pourra forcer l'autre machine à redevenir passive, et prendre sa place actuelle.

C'est l'adresse IP virtuelle qui doit « apparaître » dans les fichiers de configuration des services (par exemple, dans la configuration du serveur DNS).

L'objectif opérationnel de cette deuxième activité est de configurer l'adresse IP virtuelle et le service Web en haute disponibilité.

Vous disposez de la documentation suivante :

- **Document 1** : création et configuration des ressources.
- **Document 2** : récapitulatif des commandes courantes.

Travail à faire

Le serveur maître a pour nom d'hôte « interne » « serv1 » et le serveur secondaire « serv2 ».

Configuration du Failover IP

Q1. En vous servant du document 1, créez la ressource permettant le « failover IP ». Vous créerez un moniteur pour cette ressource avec un test de vie de la ressource de 10 secondes, un timeout de démarrage de 30 secondes et un timeout d'arrêt de 40 secondes.

Q2. Sur quel nœud la ressource a-t-elle été lancée ? Justifiez et vérifiez.

Q3. Migrez éventuellement la ressource (si elle n'y était pas déjà) sur le nœud primaire ; vérifiez ensuite que la ressource est bien activée sur ce nœud et que la préférence a été donnée à ce nœud.

Q4. Testez l'accès au service Web à partir de l'adresse IP virtuelle.

Q5. Quel autre service est obligatoirement impacté par le changement d'adresse IP ? Procédez aux ajouts/modifications nécessaires.

Q6. Testez le « failover IP » de deux manières (mise du nœud en standby et extinction du serveur). Y a-t-il une différence ?

Configuration de la ressource HTTP

Q7. A l'aide du document 1, créez la ressource « serviceWeb » et vérifiez sur quel nœud elle démarre.

Q8. Vérifiez que le service HTTP soit bien démarré sur le serveur spécifié et non activé sur l'autre.

Q9. A l'aide du document 1, groupez les ressources de manière à ce qu'elles soient actives sur le même nœud.

Q10. Quels tests pourriez-vous effectuer pour vérifier la solution mise en place ?

Documentation

Document 1 : création et configuration des ressources

Après avoir installé les services sur chaque nœud du cluster, il est nécessaire de configurer le comportement de ceux que l'on désire mettre en haute disponibilité.

Un service est une ressource au sens de Pacemaker. La commande ***crm*** qui va permettre de configurer les ressources dans le cluster est implémentée par Pacemaker ; quel que soit le nœud sur lequel la commande est exécutée, la configuration **se propage sur chaque nœud automatiquement en modifiant le fichier *cib.xml***

Comment créer une ressource ?

La création d'une ressource s'effectue avec une entrée nommée « primitive » dans la CIB. On trouvera au minimum dans la commande :
crm configure primitive <nom de la ressource> <espace de nom:nom du script>. Par exemple :

- crm configure primitive serviceWeb **ocf:heartbeat:**apache
- crm configure primitive serviceWeb **lsb**:apache

Cette « primitive » fait appel à un **script d'application correspondant au service** à mettre en haute disponibilité.

Le script du service peut être :

- **De type OCF** (http://linux-ha.org/wiki/OCF_Resource_Agent) qui est un script développé pour Pacemaker : ils sont, pour la plupart, dans le répertoire ***/usr/lib/ocf/resource.d/heartbeat/* ou *****/usr/lib/ocf/resource.d/pacemaker*** ; la liste des scripts est obtenue avec les commandes ***crm ra list ocf heartbeat*** et ***crm ra list ocf pacemaker.*** Dans la commande, l'espace de nom est dans ce cas ocf:heartbeat ou ocf:pacemaker. OCF est une classe de ressources qui propose un certain nombre de ressources intitulées Resources Agents, définies par le projet Open Cluster Framework. Pour rappel, le but d'OCF est que toute solution de clustering libre ou non s'appuie sur ses recommandations pour fonctionner.
- **De type LSB** : c'est un script de démarrage sur Linux présent dans ***/etc/init.d*** ; pour qu'il puisse être utilisé avec Pacemaker, ce script doit être conforme à la spécification LSB :

http://refspecs.linux-foundation.org/LSB_3.2.0/LSB-Core-generic/LSB-Core-generic/iniscrptact.html.
Le type ***lsb:service*** utilise au minimum les capacités *start*, *stop*, *status* des scripts de démarrage système de /etc/init.d. Pour en vérifier la compatibilité minimum, les scripts *doivent répondre à une commande start, stop et status.*
*L*e lien http://linux-ha.org/wiki/LSB_Resource_Agents aide à la vérification de la compatibilité totale. Dans la mesure du possible, il est conseillé d'utiliser les scripts OCF ou d'écrire (s'il n'existe pas) un script OCF basé sur le script d'initialisation existant. D'autant plus que pour certains services, les scripts OCF offrent des finesses supplémentaires de configuration.

On indique ensuite éventuellement (dans la commande ***crm configure primitive***) des paramètres propres à la ressource et des options de monitoring (voir plus loin les configurations spécifiques).

La configuration passe donc par la commande crm que l'on peut utiliser directement en ligne de commande (voir « configuration du failover IP ») ou en interactif (voir « configuration de la ressource « serviceWeb »).

Cette commande se propage sur tous les nœuds (ce qui est très pratique) et modifie bien évidemment le fichier XML sur chaque nœud. **Attention**, il est très dangereux de modifier ce fichier à la main (surtout quand Corosync est lancé) ; il faut utiliser à la place la commande ***crm configure edit. Si, en dernier recours, vous devez passer par la modification du fichier à la main, il est nécessaire de sauvegarder le fichier, arrêter le service « corosync » et opérer la même modification sur les deux nœuds.***

Configuration du failover d'IP

Corosync est lancé en démon (ce qui est visible avec la commande netstat) et il se charge (via Pacemaker) d'activer l'adresse IP virtuelle sur le serveur qui joue le rôle de maître ainsi que le lancement des services (ressources au sens de Pacemaker) mis en haute disponibilité.

Le failover d'IP (en anglais, fail-over se traduit par « passer outre la panne ») est le basculement de l'adresse IP (virtuelle) du serveur maître vers le serveur esclave si le premier venait à défaillir .

Première étape : attribution de l'adresse IP virtuelle

À minima, on crée une ressource nommée « *IPFailover* » (le nom est libre) qui va créer une adresse IP virtuelle 172.16.0.12/24 (d'autres options sont disponibles).

root@serv1:~# crm configure primitive IPFailover ocf:heartbeat:IPaddr2 params ip=172.16.0.12 cidr_netmask=24 nic=eth0 iflabel=VIP

En jaune = à adapter à votre cas

- **Primitive :** argument pour ajouter une primitive regroupant plusieurs valeurs indiquant au Cluster quels scripts utiliser pour la ressource, où le trouver et à quel standard il correspond.
- **Ocf** : classe de la ressource (ça pourrait donc aussi être lsb)
- **heartbeat**: fournisseur de la ressource
- **IPaddr2** : ressource gérant les adresses IPv4 virtuelles ==> le script appelé
- **Params** : déclaration des paramètres nécessaires à la ressource
- **IPFaillover** : le nom de la ressource (il est évidemment libre... mais doit être suffisamment « parlant »),
- **IPaddr2** : le script appelé
- **params :** suivent les différents paramètres à appliquer
- **ip=172.16.0.12 :** nom et valeurs du paramètre « ip »
- ***cidr_netmask=24*** *: masque de sous-réseau en notation CIDR*
- ***nic=eth0*** *: carte réseau sur laquelle est appliquée l'adresse IP virtuelle*
- **iflabel=VIP :** permet de donner un label à la carte réseau virtuelle. Sans ce label, la VIP n'est pas visible avec la commande ***ifconfig*** mais seulement avec la commande ***ip addr show***.

Remarques :

- pour connaître tous les paramètres de la primitive IPaddr2 (ainsi que ses options par défaut comme tout ce qui concerne le monitoring de la ressource) : **crm ra info ocf:heartbeat:IPaddr2 ;**
- pour visualiser en direct l'évolution de la configuration, il est intéressant de réaliser la commande sur un nœud et de laisser tourner un *« **crm_mon** »* sur l'autre (attendre un peu que la propagation soit effective) ;
- en cas d'erreur de syntaxe dans la commande, le système peut demander s'il va quand même enregistrer en terminant son retour d'erreur par « Do you still want to commit? » : il faut évidemment répondre par « n » ;
- chaque script de la classe ocf permet la supervision de la ressource via l'option monitor (voir ci-après) et/ou la promotion d'une ressource dans le cadre d'un Cluster actif/actif (comme nous allons le voir dans le une séance ultérieure).

Avec *crm_mon*, on peut constater qu'une ressource est configurée :

```
Last updated: Thu Jul 21 11:25:50 2016      Last change: Thu Jul 21 11:25:20 2016 by root via cibadmin on serv1
Stack: corosync
Current DC: serv1 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 1 resource configured

Online: [ serv2 serv1 ]

IPFailover      (ocf::heartbeat:IPaddr2):       Started serv1
```

Et nous pouvons constater aussi que l'adresse virtuelle démarrera sur l'un ou l'autre nœud de manière aléatoire (ici *Started serv1).*
Pour vérifier l'adresse IP attribuée on utilise la commande « ***ip addr show*** » (attention, **pas** ifconfig si le paramètre iflabel n'a pas été défini) :

```
root@serv1:~# ip addr show
1: lo: ...
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 36:61:30:63:30:30 brd ff:ff:ff:ff:ff:ff
    inet 172.16.0.10/24 brd 172.16.0.255 scope global eth0
       valid_lft forever preferred_lft forever
   inet 172.16.0.12/24 brd 172.26.0.255 scope global secondary eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::3461:30ff:fe63:3030/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: ...
```

Si le paramètre « iflabel » a été défini :

```
root@serv1:~# ifconfig
eth0   Link encap:Ethernet  HWaddr 36:61:30:63:30:30
         inet adr: 172.16.0.10  Bcast: 172.16.0.255  Masque:255.255.255.0
       ...

eth0:VIP  Link encap:Ethernet  HWaddr 36:61:30:63:30:30
         inet adr:172.16.0.12  Bcast: 172.16.0.255  Masque:255.255.255.0
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

Pour éditer/modifier une ressource : ***crm configure edit <id_ressource>*** ; cela vous placera dans une instance de « *vim* » pour effectuer une modification. À la sauvegarde, la modification est directement appliquée. Par exemple, si l'on veut ajouter à la ressource configurée précédemment des options de monitoring : ***crm configure edit IPFailover :***

```
primitive IPFailover ocf:heartbeat:IPaddr2 \
       params ip="172.16.0.12" cidr_netmask="255.255.255.0" nic="eth0" op monitor interval="30s" timeout="20s"
```

« op » définit des options et « monitor » est l'action de monitoring du pacemaker : toutes les 30 secondes, Pacemaker va appeler le script OCF avec comme paramètre monitor. Avant le timeout de 20 secondes, on devra avoir un code retour à 0 pour que Pacemaker considère la ressource comme fonctionnelle. Sinon, il arrêtera/relancera la ressource ou effectuera une bascule (selon le paramétrage des autres options).

Remarque : Pour éditer/modifier **l'ensemble** d'une configuration : ***crm configure edit.***

Deuxième étape : définir une préférence de nœud primaire pour l'adresse virtuelle

Il s'agit en fait de définir une **contrainte « locative »** qui va définir une préférence d'une ressource sur un nœud : **le plus simple est, en fait, de laisser Pacemaker « écrire » lui même cette contrainte.** On migre la ressource sur le nœud que l'on veut et Pacemaker comprend qu'on a une préférence pour ce nœud : ***crm resource move IPFailover serv1***

Et on constate : crm configure show

```
...
location cli-prefer-IPFailover IPFailover role=Started inf: serv1
...
```

À partir de ce moment là, dès que le nœud serv1 est actif, la ressource migrera vers ce nœud (**auto failback**).

Cette situation n'est peut-être pas la situation idéale car il est souvent préférable de revenir à un nœud de manière manuelle après avoir vérifié que toutes les conditions favorables sont réunies et qu'il n'y aura pas de perte de données.
Dans ce cas, il faut utiliser le paramètre « resource-stickiness » ***crm configure property default-resource-stickiness=100*** qui empêche la ressource de retourner sur la machine élue par défaut après que celle-ci ait défailli et soit revenue en ligne. La ressource devra être migrée manuellement.

Et dans certains cas, on ne veut privilégier aucun serveur par rapport à d'autres (par exemple, quand les machines maître et esclave sont strictement identiques, notamment en terme de capacités).

Test du failover d'IP

Plusieurs possibilités pour tester la solution mise en place dont :

- arrêter le serveur ou le service ;
- mettre un nœud en maintenance : ***crm node standby*** sur le nœud que l'on veut rendre inactif (***crm node online*** pour le remettre actif).

Par exemple cette commande sur le nœud *serv1* a pour effet de faire basculer l'adresse IP virtuelle sur le nœud *serv2* :

```
Last updated: Thu Jul 21 11:42:08 2016          Last change: Thu Jul 21 11:41:49 2016 by root via
crm_attribute on serv1
Stack: corosync
Current DC: serv1 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 1 resource configured

Node serv1: standby
Online: [ serv2 ]

IPFailover      (ocf::heartbeat:IPaddr2):       Started serv2
```

Un ***crm node online*** sur le nœud que l'on veut rendre actif doit permettre de récupérer l'adresse IP sur *intralab* car nous avons défini la préférence ainsi :

```
Last updated: Thu Jul 21 11:43:35 2016          Last change: Thu Jul 21 11:43:32 2016 by root via
crm_attribute on serv1
Stack: corosync
Current DC: serv1 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 1 resource configured

Online: [ serv2 serv1 ]

IPFailover      (ocf::heartbeat:IPaddr2):       Started serv1
```

Rappel : en production, il est parfois préférable de maîtriser le retour au serveur maître.

Configuration de la ressource « serviceWeb »

Le service Web comme tout service est lancé automatiquement par Linux au démarrage de la machine. **C'est Pacemaker qui va se charger de désactiver sur le serveur esclave le service**. Ainsi pour voir si un service est actif ou non, on ne peut pas utiliser la commande */etc/init.d/nom_du_service status* ou *service nom_du_service status* qui renverra un statut actif. Il est possible d'utiliser commandes nmap, netstat ou de tester l'accès au service via le client correspondant.

Nous allons configurer cette ressource en « interactif » car cela procure plusieurs avantages dont :

- l'utilisation de la touche *tabulation* qui peut aider à trouver la « bonne » commande ou le « bon » argument (la double tabulation listera les possibilités) ;
- l'utilisation de la commande *help* qui liste les commande possibles ;
- la possibilité de configurer plusieurs ressources avec leurs propriétés avant de valider le tout (par un *commit*).

root@serv1:~# crm configure

```
crm(live)configure# primitive serviceWeb lsb:apache2 op monitor interval=60s op start interval=0 timeout=60s op stop interval=0 timeout=60s
crm(live)configure# commit
crm(live)configure# quit
```

On crée une ressource nommée « serviceWeb ».

Chaque option (en règle générale, chaque action à effectuer) est ensuite précédée du mot clé « op ».
On crée un moniteur pour cette ressource avec un test de la « vie » de la ressource toutes les 60 secondes (op monitor interval=60s) et on spécifie en plus un timeout de démarrage de 40 secondes (op start timeout=40s) et un timeout d'arrêt de 40 secondes (op stop timeout=40s).

On valide la ressource par « *commit* » et on sort de la commande interactive par « *quit* » ou « *exit* »

N'hésitez pas à utiliser la tabulation et la double tabulation...

Remarque : avant de valider la ressource, on peut la consulter :

```
crm(live)configure# show
...
primitive serviceWeb lsb:apache2 \
        op monitor interval="60s"
        op start interval="0" timeout="60s" \
        op stop interval="0" timeout="60s" \
...
```

À la validation (commit), un « Warning » peut apparaître :
WARNING: serviceWeb: specified timeout 40s for stop is smaller than the advised 60s
Vous pouvez l'ignorer ou... en tenir compte. Les développeurs ont estimé qu'un délai de 60s pour arrêter ce service était raisonnable. Si vous pensez qu'effectivement, l'arrêt du service Web prendra plus de 40 secondes, il vaut mieux augmenter le délai.

Sur la console crm_mon, la nouvelle ressource apparaît bien. :

Last updated: Thu Jul 21 12:04:42 2016 Last change: Thu Jul 21 12:04:37 2016 by root via cibadmin on serv1
Stack: corosync
Current DC: serv1 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 2 resources configured

Online: [serv2 serv1]

IPFailover (ocf::heartbeat:IPaddr2): Started serv1
serviceWeb (lsb:apache2): Started serv2

On constate avec la commande ***nmap localhost -p 80*** que le service est "open" sur serv2 et "closed" sur serv1. Par ailleurs l'accès à l'application est toujours effectif.

Et, comme on peut le voir, la ressource *serviceWeb* n'est pas démarrée sur le même nœud que celle concernant l'IP virtuelle. **Par défaut Pacemaker répartit les ressources entre les membres du cluster.**

Pour que l'adresse IP virtuelle et le service « serviceWeb » soient sur le même nœud, il existe plusieurs possibilités comme le groupement des ressources (solution la plus simple) :

root@serv1:~# crm configure
crm(live)configure# *group servweb IPFailover serviceWeb meta migration-threshold="5"*
crm(live)configure# *commit*
crm(live)configure# *quit*

On crée un groupe nommé « servweb » composé des ressources « IPFailover » et « serviceWeb » qui seront toujours démarrées dans l'ordre des entrées du groupe (arrêtées en sens inverse).
On spécifie aussi le nombre d'incidents tolérés avant de migrer la ressource définitivement vers un autre nœud ("meta migration-threshold=5"). Ainsi, au bout de cinq incidents, le groupe de ressources sera définitivement migré vers un autre nœud ; si une ressource échoue au démarrage, l'ensemble des ressources du groupe sera démarré sur un autre nœud.

==> **La ressource « serviceWeb » migre immédiatement sur le même nœud que celui de « IPFailover ».**

Last updated: Thu Jul 21 12:08:46 2016 Last change: Thu Jul 21 12:08:41 2016 by root via cibadmin on serv1
Stack: corosync
Current DC: serv1 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 2 resources configured

Online: [serv2 serv1]

Resource Group: servIntralab
IPFailover (ocf::heartbeat:IPaddr2): Started serv1
serviceWeb (lsb:apache2): Started serv1

Nous étudierons, dans les prochaines séances, d'autres possibilités de configuration plus compliquées mais permettant des paramétrages plus fins.

Test de la solution

Il est nécessaire de tester la survenue de 2 problèmes :

1. **Le serveur est hors service** : dans ce cas, il suffit de l'éteindre ou de mettre le nœud en standby et de s'assurer que les services ont bien migré sur l'autre serveur et sont opérationnels.

2. **Seul le service Web n'est plus actif**.

Si le service est arrêté par la commande classique ***service apache2 stop***, pacemaker essaye de réactiver le service et théoriquement y arrive : on se retrouve avec un service qui est apparemment stoppé mais qui est en fait actif ==> il n'y a donc pas de migration de ressources. **Pour tester un arrêt définitif du service** de manière à ce que l'ensemble des ressources migrent sur l'autre nœud, il faut s'arranger pour que le service ne puisse pas redémarrer : on peut pas exemple écrire une directive qui n'existe pas dans */etc/apache2/apache2.conf* et on stoppe le service. Il ne faudra pas oublier de supprimer cette directive puis de redémarrer le service.

Quand le cluster bascule sur l'autre nœud suite à un certain nombre d'erreurs sur une ressource, le nœud qui a rencontré les erreurs ne pourra plus héberger les ressources tant que toutes les erreurs n'auront pas été effacées par la commande suivante : ***crm resource cleanup <id_resource>***

Document 2 : 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