

Haute disponibilité Apache avec Heartbeat



Table des matières

1. Installation et configuration.....	2
a. Liaison série :	2
b. Installation de heartbeat :.....	2
b1. Le fichier ha.cf.....	2
b2. haresources.....	3
b3. authkeys.....	3
2. Test.....	4

Nous allons voir ici comment installer le produit HeartBeat, qui permet de créer des clusters de haute disponibilité sous GNU/Linux. Ici nous allons tenter de rendre hautement disponible le service de serveur web Apache2. Ainsi si le serveur principal tombe, le second traitera automatiquement les requêtes de clients en toute transparence.

1. Installation et configuration

a. Liaison série :

Comme son nom l'indique, Heartbeat utilise des « battements de coeur » permanents entre les différents nœuds du cluster. Si l'un de ces battements ne reçoit pas de réponse, c'est à ce moment qu'il déclenchera son processus de haute disponibilité. Pour faire passer ce battement, nous utiliserons une liaison fiable, un câble série de type null-modem. En effet l'utilisation d'une liaison ethernet pour les battements de coeur n'est pas des plus fiables, surtout si elle traverse divers éléments du réseau.

Nous allons donc nous assurer de la bonne transmission des données via le câble série entre les deux postes :

Sur le premier nœud :

```
cat < /dev/ttyS0
```

Sur le second nœud :

```
echo test > /dev/ttyS0
```

Normalement le premier nœud devrait voir le message « test » s'afficher.

b. Installation de heartbeat :

Sous GNU/Linux Debian :

```
apt-get install heartbeat
```

Les fichiers sont installés dans le répertoire `/etc/heartbeat` si les fichiers n'existent pas, il suffit de les créer :)

b1. Le fichier ha.cf

Le fichier principal de configuration se nomme `ha.cf` voici son contenu commenté

```
#Fichier de configuration de heartbeat  
  
#Ajout des info dans le syslog  
logfacility local0
```

```
#Deux secondes entre chaque battement
keepalive 2
#Dix secondes sans battement, machine déclaré hs
deadtime 10
#Six seconde sans battement entrainealerte dans les logs
warntime 6
#delais de demarrage augmenté de 60 sec si l'un des noeud est long a
demarrer pour que le second ne croit pas qu'il est HS
initdead 60
#Port UDP utilisé pour les battements
udpport 694
#lien réseau utilisé pour les battements
bcast eth0
#lien série utilisé pour les battements
serial /dev/ttyS0
#quand le maitre remonte de son crash, il reprend le contrôle principal
auto_failback on
#les differents noeud du réseau (uname -n)
node nico.test.local
node dracofeu
```

b2. haresources

Ce fichiers défini dans un premier temps le maitre du cluster, ainsi que l'adresse ip virtuel que l'on va utilisé pour le cluster, cette même adresse IP virtuelle sera automatiquement montée sur le noeud de secour si le premier tombe, assurant ainsi une total transparence de la panne.

Voici un exemple de fichier haresources :

```
serveur-miatre IPaddr2 ::192.168.1.236/24/eth0:0 apache2
MailTo ::root
```

Ici nous indiquons que le nom du serveur maitre, puis nous faisons appel au script `ipaddr2` qui nous permettra de monter un adresse ip virutel sur le machine, ayant pour adresse 192.168.1.236 avec un masque de 24 bits et qui aura pour alias `eth0:0`. Puis enfin nous lui précisons que nous voulons de la haute dispo sur `apache2`.

`MailTo ::<user>` permet d'envoyer un mail à `<user>` l'alertant du changement d'état du cluster.

b3. authkeys

Ce fichier permet de définir le type d'algorithme permettant d'authentifier les battements de coeurs entre les noeuds. Nous pouvons utiliser les types suivants : `sha1`, `crc` et `md5`.

Ici nous utiliserons le simple `crc` pour la vérification de l'intégrité des données qui circulent :

```
auth 1
1 crc
```

Protégeons ses fichiers en lui attribuant un *chmod 600 authkeys*

2. Test

Tout d'abord nous allons désactiver la gestion du démarrage et de l'extinction du service apache par le system. En effet désormais c'est heartbeat qui le lancera pour nous, comme défini dans le fichier *haresources*.

Utilisons la commande qui va bien : *update-rc.d -f apache2 remove*

Voilà maintenant nous pouvons relancer sur nos noeuds le service heartbeat. Un petit `tail` dans le `syslog` vous permettra de vérifier que le noeud est créé et que les différentes machines du noeuds communiquent.

Vous pouvez également faire un *ps aux |grep apache* sur les noeuds esclave afin de vous assurer que le service *apache2* ne tourne pas. Normal car c'est heartbeat qui s'en chargera en cas de plantage du premier noeud.

Maintenant vient le test le plus parlant, nous allons sur chacun de nos noeuds créer une page html toute simple contenant uniquement le nom de la machine sur laquelle elle est hébergée. Maintenant si nous ouvrons un navigateur web et que nous le pointons vers l'ip virtuel du noeud (192.168.1.236 dans notre cas), nous devrions voir une page contenant le nom de notre noeud maître. Maintenant, débranchez la prise RJ45 de votre serveur maître, puis actualisez le navigateur; le nom du noeud esclave est apparu. Autrement dit la bascule a bien été effectuée, notre cluster de haute disponibilité est opérationnel.