ALOHA LOAD BALANCER
METHODE DE CONTROLE DE VITALITE

« APP NOTES » #0013 — LISTE DES CHECKS DANS L'ALOHA

Ce document a pour vocation de lister les principaux checks disponibles dans la solution ALOHA pour s'assurer de l'état de santé des serveurs réels et donc éviter les dysfonctionnements potentiels.

OBJECTIF

Configurer l'Aloha pour contrôler l'état de vitalité des serveurs applicatifs dans une ferme donnée.

COMPLEXITE

VERSIONS CONCERNEES

V 3.x et ultérieures

CHANGELOG

2012-01-02: Mise à jour pour a version ALOHA 5.5 et supérieur

2011-03-28: version initiale

CHECKS DISPONIBLES POUR LA REPARTITION DE CHARGE DE NIVEAU 4

option tcpcheck

```
option tcpcheck  [interval <seconds>] [timeout <seconds>]
                 [source <ip>] [port <int>]
```

Cette méthode de test permet de vérifier si une connexion TCP est ouverte sur les adresses IP des serveurs réels.

Avec :

- ***interval:*** fréquence des vérifications exprimée en secondes (10 secondes par défaut).
- ***timeout:*** délai au bout duquel on considère un échec si la connexion n'a pas été établie (3 secondes par défaut).
- ***source:*** force l'IP source qui doit être utilisée.
- ***port:*** force le port de destination (par défaut, le port des serveurs réels est utilisé s'il existe).

option httpcheck

```
option httpcheck    [interval <seconds>] [timeout <seconds>] [source <ip>]
                    [port <int>] [uri <uri>] [statuscode <int>]
```

Cette méthode de test, permet de vérifier la performance d'une requête HTTP de type « GET » sur les adresses IP des serveurs réels.

Avec :

- ***interval:*** fréquence des vérifications exprimée en secondes (10 secondes par défaut).
- ***timeout:*** délai au bout duquel on considère un échec en l'absence d'une réponse du serveur (3 secondes par défaut).
- ***source:*** force l'IP source qui doit être utilisée.
- ***port:*** force le port de destination (par défaut, le port des serveurs réels est utilisé s'il existe).
- ***uri:*** uri demandée (par défaut, il s'agit de la racine "/").
- ***statuscode:*** code d'état attendu (par défaut, on observe le code 200 : OK Requête traitée avec succès).

option arpcheck

```
option arpcheck   [interval <seconds>] [timeout <seconds>]
                  [source <ip>] [iface <name>]
```

Cette méthode de test permet d'émettre un « arp-whoas » sur les adresses IP des serveurs réels.

Avec :

- ***interval:*** fréquence des vérifications exprimée en secondes (10 secondes par défaut).
- ***timeout:*** délai au bout duquel on considère un échec en l'absence d'une réponse du serveur (3 secondes par défaut).
- ***source:*** force l'IP source qui doit être utilisée.
- ***iface:*** nom de l'interface réseau.

option icmpcheck

```
option icmpcheck [interval <seconds>] [timeout <seconds>] [source <ip>]
```

Cette méthode de test permet d'émettre un « icmp echo » sur les adresses IP des serveurs réels.

Avec :

- ***interval:*** fréquence des vérifications exprimée en secondes (10 secondes par défaut).
- ***timeout:*** délai au bout duquel on considère un échec en l'absence d'une réponse du serveur (3 secondes par défaut).
- ***source:*** force l'IP source qui doit être utilisée.

CHECKS DISPONIBLES POUR LA REPARTITION DE CHARGE DE NIVEAU 7

Adresse IP et Port

Par défaut, le check est envoyé sur l'adresse IP et le port du serveur configuré pour le load-balancing. Il est possible de surcharger ces informations avec les paramètres **addr** et **port**.

```
serveur server1 10.0.0.1:443 check port 80
```

Dans l'exemple ci-dessus, la répartition de charge est faite sur le port 443, mais le health check est exécuté sur le port 80.

check chiffré (SSL)

(Aloha 5.5 et supérieur uniquement)

Par défaut, les checks sont envoyés en clair. Si l'ALOHA est configuré pour se connecter au server en SSL, alors le healthcheck sera chiffré.

Dans certains cas, il peut être utile de forcer le chiffrement du check, notamment lors du load-balancing de services SSL en mode TCP combiné avec un health check niveau 7 (paramètre **check-ssl**):

```
option httpchk
serveur server1 10.0.0.1:443 check check-ssl
```

check tcp

Par défaut, lorsqu'aucune option de check n'est spécifiée, le contrôle de vitalité des serveurs consiste uniquement à essayer d'établir une connexion TCP.

Si le paramètre check-ssl est présent, le check se transforme en un CONNECT ssl.

option httpcheck

```
option httpchk
option httpchk <uri>
option httpchk <method> <uri>
option httpchk <method> <uri> <version>
```

Cette méthode de test, permet de vérifier via le protocole HTTP « l'état de santé » des serveurs.

Lorsque l'option "httpchk" est spécifiée, une requête HTTP complète est envoyée une fois la connexion TCP établie. Des réponses 2xx ou 3xx sont considérées comme valides, alors que toutes les autres indiquent une défaillance du serveur y compris l'absence de réponse.

Avec :

- ***method:*** est une méthode HTTP optionnelle utilisée lors des requêtes. Lorsqu'elle n'est pas précisée, la méthode « OPTIONS » est alors utilisée, car elle requière peu de ressources sur les serveurs et son filtrage dans les logs est aisé.

- ***uri:*** uri référencée dans les requêtes HTTP. La valeur par défaut correspond à la racine "/" qui est accessible par défaut sur n'importe quel serveur. Toutefois, n'importe quelle URI peut convenir et les chaînes de requêtes sont autorisées.

- ***version:*** est la chaîne optionnelle de version HTTP. La valeur par défaut est "HTTP/1.0", mais certains serveurs peuvent se comporter de façon incorrecte avec le protocole HTTP 1.0. Il est possible alors de se tourner vers la version HTTP/1.1.

IMPORTANT :

Le port et l'intervalle sont précisés dans la configuration du serveur.

Cette option ne requiert pas nécessairement un backend HTTP, et peut également fonctionner avec des backends TCP. Cette particularité est utile pour vérifier de simples scripts au travers de ports dédiés et utilisant le démon inetd.

option mysql-check

```
option mysql-check
```

Cette méthode de test, permet de vérifier via une initialisation de connexion MySQL « l'état de santé » des serveurs.

La vérification consiste à analyser les paquets transmis lors d'une initialisation de connexion avec un serveur MySQL. C'est un test de base, mais utile, qui ne produit pas de logs sur le serveur. Toutefois, il ne vérifie pas la présence de bases de données, ni leur cohérence, ni même les permissions d'accès. Pour ce faire, il est possible d'utiliser un contrôle externe avec « xinetd » par exemple.

option smtpchk

```
option smtpchk
option smtpchk <hello> <domain>
```

Cette méthode de test, permet de vérifier via le protocole SMTP « l'état de santé » des serveurs.

Avec :

- ***hello:*** est un argument optionnel qui utilise la commande « HELLO ». Il peut s'agir soit d'un « HELO » pour du SMTP, soit d'un « EHLO » pour du ESMTP.

- ***domain:*** est le nom de domaine à présenter au serveur. Il ne peut être spécifié que si la commande « HELLO » a été spécifiée. Par défaut, "localhost" est utilisé.

Lorsque "option smtpchk" est configurée, les contrôles de vitalité consisteront à suivre les connexions TCP effectuées par une commande SMTP. Par défaut, la commande est "HELO localhost". Le code retourné par le serveur est alors analysé et s'il commence par un "2", il sera considéré comme actif. Toutes les autres réponses, ainsi que l'absence de réponse constituent une erreur et indiquent que le serveur est indisponible.

Ce test est destiné à être utilisé avec les serveurs SMTP ou relais SMTP.

option ssl-hello-chk

```
option ssl-hello-chk
```

Lorsque certains protocoles basés sur du SSL sont relayés en mode TCP par le biais d'HAProxy, il est possible de vérifier que le serveur communique correctement en SSL au lieu de simplement vérifier qu'il accepte les connexions TCP. Quand l'option « option ssl-hello-chk » est configurée, un message « client hello » au format SSLv3 est envoyé une fois que la connexion est établie avec le serveur. La réponse est analysée afin d'y trouver un « server hello » correspondant.

Le serveur est considéré comme actif lorsque la réponse du serveur répond bien à l' « hello ».