Retour d'expérience sur la mise en place d'une solution de répartition de charge entièrement libre.

Gauthier Catteau

Ingénieur Systèmes et Réseaux
Responsable Infrastructure Système de l'Académie de Lille.
gauthier.catteau@ac-lille.fr

6 décembre 2011
Journée Thématique Min2rien
`http://www.min2rien.fr/`

This work is licensed under the *Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License*.

Lignes directrices

État des lieux

Nous avons 2 technologies de répartition de charges différentes qui ne nous donnent pas entière satisfaction.

1. 2 Cisco CSS 11501S-K9 *en place depuis 5 ans*.
2. 2 Foundry 4G-SSL *en place depuis 2 ans*.

Ces 4 boitiers sont toujours utilisés car la migration des ip virtuelles et des règles prend du temps. La technologie Foundry étant davantage orientée L4 que L7, la migration depuis les CSS en est d'autant plus complexe. Bien que cette migration ait été bien préparée nous l'avons interrompue car le résultat n'était pas satisfaisant.

Réflexions

Nous avons donc recherché une solution :

1. plus proche du système et des applications.
2. avec une configuration simple que l'on peut commenter.
3. facile à deployer.
4. robuste et redondante.

Lignes directrices

Choix de la solution

Solution basée sur plusieurs logiciels :

1. Une distribution Linux
2. Haproxy
3. Nginx
4. Heartbeat3
5. Mercurial

Présentation d'Haproxy

Avantages d'Haproxy

Le choix s'est porté sur Haproxy pour les raisons suivantes :

- Performant et rapide à mettre en place.
- Gestion fine des acls.
- Plugin Nagios pour superviser l'ensemble des ressources en une requête.
- Bonne documentation sur `http://code.google.com/p/haproxy-docs/wiki/` .
- Facile à débugger (possibilité de journalisation des entêtes http, cookies...).
- Journalisation native vers rsyslog.

Inconvénients

- Pas de support ssl.

Présentation d'Haproxy

Avantages d'Haproxy

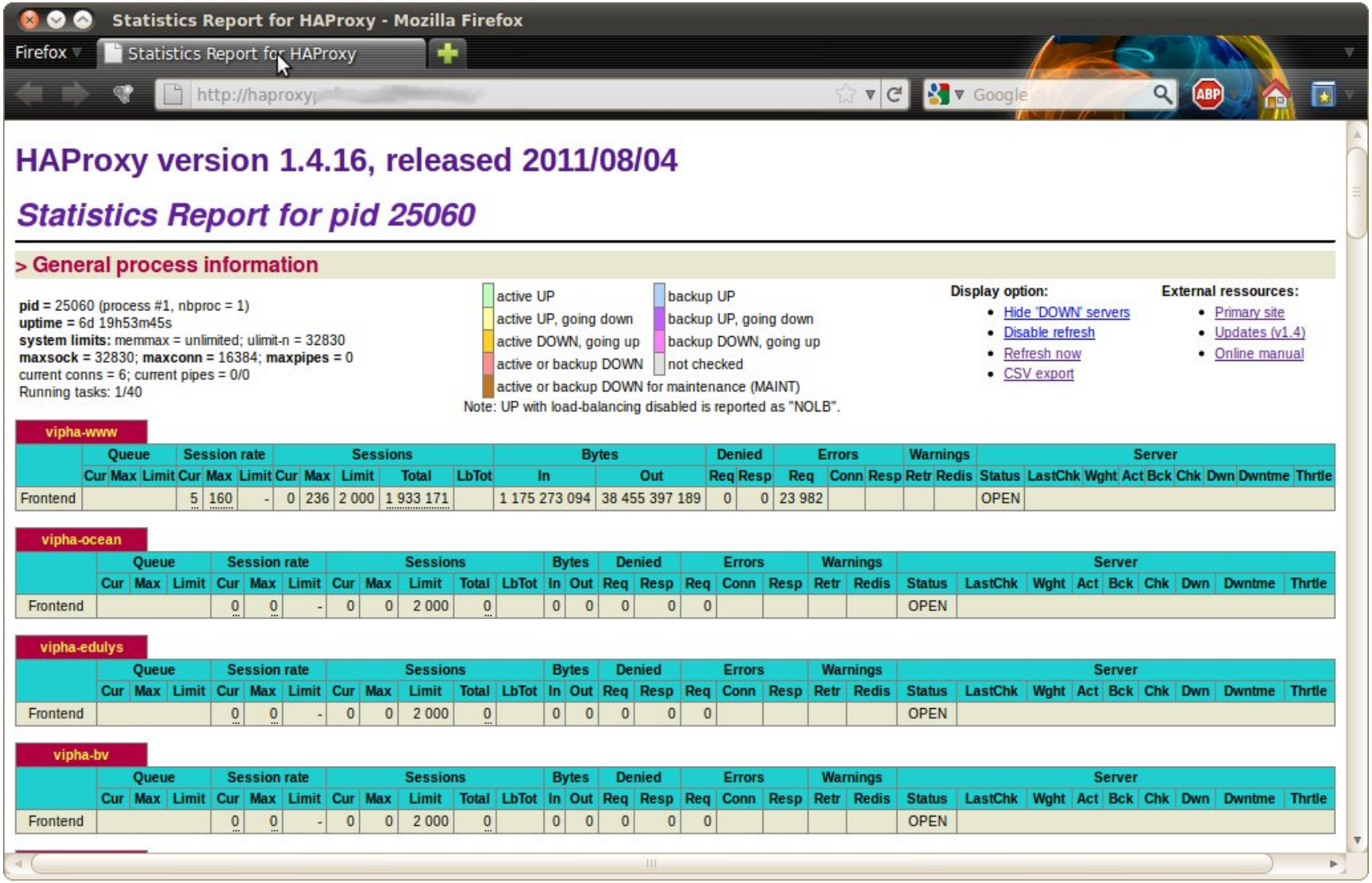
Le choix s'est porté sur Haproxy pour les raisons suivantes :

- Performant et rapide à mettre en place.
- Gestion fine des acls.
- Plugin Nagios pour superviser l'ensemble des ressources en une requête.
- Bonne documentation sur `http://code.google.com/p/haproxy-docs/wiki/` .
- Facile à débugger (possibilité de journalisation des entêtes http, cookies...).
- Journalisation native vers rsyslog.

Inconvénients

- Pas de support ssl.

Capture d'écran des stats Haproxy



Configuration

Le fichier de configuration se décompose en 4 parties :

1. La configuration globale et les paramétrages par défaut.
2. Les **frontend**s qui définissent le comportement des vips.
3. Les **backend**s qui définissent la répartition des services qui tournent sur un ou plusieurs serveurs.
4. Les **listen**s, ports en écoute pour d'autres services.

La configuration globale

la partie globale

```
1 global
      maxconn 16384 # Nombre total de connexions simultanées.
3     user haproxy
      group haproxy
5     daemon
      stats socket /var/run/haproxy.stats mode 600
```

les paramètres appliqués à tous les backends et frontends.

```
defaults
    log   global
    log 127.0.0.1  local1 info  # On envoie les logs sur local1 en udp.
    option dontlognull
    retries 3           # 3 tentatives pour valider une erreur de connexion.
    option redispatch   # On redispatche en cas de perte d'un serveur.
    maxconn 2000        # Nombre par défaut de connexions simultanées.
    timeout client 300s # Temps maximum de réponse du client.
    timeout server 300s # Temps maximum de réponse du serveur.
    timeout queue 60s   # Mise en attente maximum en cas de saturation.
    timeout connect 4s  # Temps de connexion, pas de raison de changer cette valeur.
    timeout http-request 5s  # Une requête complète ne doit pas être longue.
    default-server inter 20s # intervalle de test de 20 secondes
```

La configuration globale

la partie globale

```
global
    maxconn 16384 # Nombre total de connexions simultanées.
    user haproxy
    group haproxy
    daemon
    stats socket /var/run/haproxy.stats mode 600
```

les paramètres appliqués à tous les backends et frontends.

```
10  defaults
        log   global
12      log 127.0.0.1  local1 info  # On envoie les logs sur local1 en udp.
        option dontlognull
14      retries 3           # 3 tentatives pour valider une erreur de connexion.
        option redispatch   # On redispatche en cas de perte d'un serveur.
16      maxconn 2000        # Nombre par défaut de connexions simultanées.
        timeout client 300s # Temps maximum de réponse du client.
18      timeout server 300s # Temps maximum de réponse du serveur.
        timeout queue 60s   # Mise en attente maximum en cas de saturation.
20      timeout connect 4s  # Temps de connexion, pas de raison de changer cette valeur.
        timeout http-request 5s  # Une requête complète ne doit pas être longue.
22      default-server inter 20s # intervalle de test de 20 secondes
```

Le Frontend 1/3

Exemple de Frontend

```
30  frontend vipha-site2
        # On précise ici le type de log.
32      mode  http
        option httplog clf
34      # Les différents ports d'écoute.
        bind 192.168.0.201:81
36      bind 192.168.0.201:8443
        # Les headers seront ajoutés dans les logs.
38      # Le dernier chiffre correspond à la taille de la capture.
        capture request header User-Agent len 64
40      capture request header Referrer len 30
        capture request header X-Forwarded-For len 15
42      # Il est possible de loguer les cookies également.
        capture cookie PHPSESSION len 32
```

.../...

Le Frontend 2/3

Exemple de Frontend : partie acl

```
44      acl url_livret path_beg          /livret
        acl url_avsi path_beg            /avsi
46      acl url_portail_site2 path_dir      /portail_site2
        acl url_racine path              /
48       # Ici l'acl est positionnée sur un header (hdr) host.
        acl host_site2 hdr(host) -i site2
50      acl missing_cl hdr_cnt(Content-length) eq 0
```

.../...

Le Frontend 3/3

Exemple de Frontend : répartition et redirection

```
52      # On bloque ici la methode TRACE ou un POST vide :
        block if METH_TRACE || METH_POST missing_cl
54      # Règle de réécriture :
        reqrep ([^\ ]*)\ /portail/(.*) \1\ /portail_site2/\2
56      # Réécriture de https://site2/* en https://site2.mondomaine.fr/* :
        redirect code 301 prefix https://site2.mondomaine.fr if host_site2}
58      redirect location https://site2.mondomaine.fr/portail/ code 301 if url_racine
        use_backend webint if url_portail_site2
60      use_backend webint if url_livret url_avsi
        # Les url qui n'appartiennent à aucune règle sont envoyées sur le backend par
            défaut.
62      default_backend back2 # Règle par défaut. Facultatif.
```

Le Backend 1/3

Exemple de répartition roundrobin graçe à l'ajout d'un cookie.

```
100 backend webint
        mode http
102     # La répartition peut se faire de différentes façons (source, static-rr, uri ...)
            .
        balance roundrobin
104     # Ici le cookie idwebint est positionné sur le navigateur pour connaître le
            serveur à joindre.
        cookie IDWEBINT insert indirect domain .mondomaine.fr
106     # On ferme les connexions http.
        option httpclose
108     # On ajoute le X-Forwarded-For dans l'entête http.
        option forwardfor
110     option httpchk HEAD /test.html HTTP/1.0
        server webint1 192.168.0.3:80 cookie webint1 check inter 8s weight 40
112     server webint2 192.168.0.4:80 cookie webint2 check weight 60
        server webbackup 192.168.0.5:80 check backup
114     # Webbackup n'est utilisé que si webint1 et 2 ne répondent plus.
```

Le Backend 2/3

Exemple de répartition roundrobin grâce à un cookie existant.

```
120 backend appli_eval
        mode http
122     balance roundrobin
        option httpclose
124     option forwardfor
        option httpchk GET /appli_eval/check-appli-url.php HTTP/1.0
126     # Ici nous capturons le cookie PHPSESS_APPLI_EVAL pour définir la répartition de
            charge.
        appsession PHPSESS_APPLI_EVAL len 52 timeout 3h prefix request-learn
128     server webphp1 192.168.1.1:80 check
        server webphp2 192.168.1.2:80 check
```

Le Backend 3/3

Exemple de répartition sur un hash d'une entête http.

```
140 backend appli-ent-3
        mode http
142     # ici nous utilisons l'entête X-Real-IP pour définir la répartition de charge.
        # L'entête est positionnée par Nginx (voir plus loin)
144     balance hdr(X-Real-IP)
        option httpclose
146     option forwardfor
        option httpchk GET /ent3/check-appli-url.php HTTP/1.0
148     server ent-pers1 192.168.1.8:80 check
        server ent-pers2 192.168.1.9:80 check
```

Le Listen

Exemple de Listen.

La directive **Listen** permet de créer un service de répartition allégé sans acl ou de mettre en place les statistiques.

```
listen pop
        mode tcp
        bind 192.168.0.200:110
        balance roundrobin
        option tcplog
        server proxypop1 192.168.0.41:110 check
        server proxypop2 192.168.0.42:110 check

listen stats
        mode    http
        bind 192.168.0.5:8888
        bind 192.168.0.6:8888
        stats refresh 60s
        stats uri /haproxy-status
```

Présentation de Nginx

Haproxy ne faisant pas de chiffrement SSL, il est nécessaire d'ajouter un outil pour cela. Notre choix s'est porté vers Nginx mais d'autres choix existent : stunnel, stud ...

Avantages de Nginx

- Intégré aux distributions linux.
- Performant et rapide à mettre en place.
- Gère bien ssl.
- Modulaire.
- Permet une réécriture des url plus fine que haproxy

Inconvénients

- Pas de support rsyslog en natif.
- Gestion des passphrases avec expect.

Présentation de Nginx

Haproxy ne faisant pas de chiffrement SSL, il est nécessaire d'ajouter un outil pour cela. Notre choix s'est porté vers Nginx mais d'autres choix existent : stunnel, stud ...

Avantages de Nginx

- Intégré aux distributions linux.
- Performant et rapide à mettre en place.
- Gère bien ssl.
- Modulaire.
- Permet une réécriture des url plus fine que haproxy

Inconvénients

- Pas de support rsyslog en natif.
- Gestion des passphrases avec expect.

Configuration de Nginx

Nous n'utilisons que la partie chiffrement ssl de nginx. La configuration est très simple.

Configuration ssl

```
server {
    listen                192.168.0.200:443;
    ssl                   on;
    server_name           site1.mondomaine.fr;
    ssl_certificate       /etc/ssl/certs/site1.mondomaine.fr.pem;
    ssl_certificate_key   /etc/ssl/private/site1.mondomaine.fr.key;
    location / {
       proxy_pass         http://192.168.0.200:8443;
       proxy_set_header X-Real-IP $remote_addr;
       proxy_set_header X-Forwarded-Host $host;
       proxy_set_header X-Forwarded-Server $host;
       proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
       proxy_headers_hash_max_size 1024;
       proxy_headers_hash_bucket_size 128;
    }
}
```

Configuration de Nginx

Ici Nginx écoute sur le port 80 pour rediriger les flux vers le https.

Configuration redirection http->https avec Nginx

```
20  server {
        listen                  192.168.0.200:80;
22      ssl                     off;
        server_name             site1.mondomaine.fr;
24      rewrite ^(.*) https://$server_name$1 permanent;
    }
```

Configuration redirection http->https avec Haproxy

```
frontend vipha-redirect-site1
    mode    http
    option  httplog clf
    bind 192.168.0.200:80
    acl url_site1 hdr(host) -i site1.mondomaine.fr
    acl url_site1 hdr(host) -i site1
    redirect code 301 prefix https://site1.mondomaine.fr if url_site1
```

Configuration de Nginx

Ici Nginx écoute sur le port 80 pour rediriger les flux vers le https.

Configuration redirection http->https avec Nginx

```
server {
   listen                192.168.0.200:80;
   ssl                   off;
   server_name           site1.mondomaine.fr;
   rewrite ^(.*) https://$server_name$1 permanent;
}
```

Configuration redirection http->https avec Haproxy

```
frontend vipha-redirect-site1
    mode    http
    option  httplog clf
    bind 192.168.0.200:80
    acl url_site1 hdr(host) -i site1.mondomaine.fr
    acl url_site1 hdr(host) -i site1
    redirect code 301 prefix https://site1.mondomaine.fr if url_site1
```

Oui mais est-ce performant ?

Performance de Nginx et stud sur un bi-pro quad-coeur en 32 et 64bits

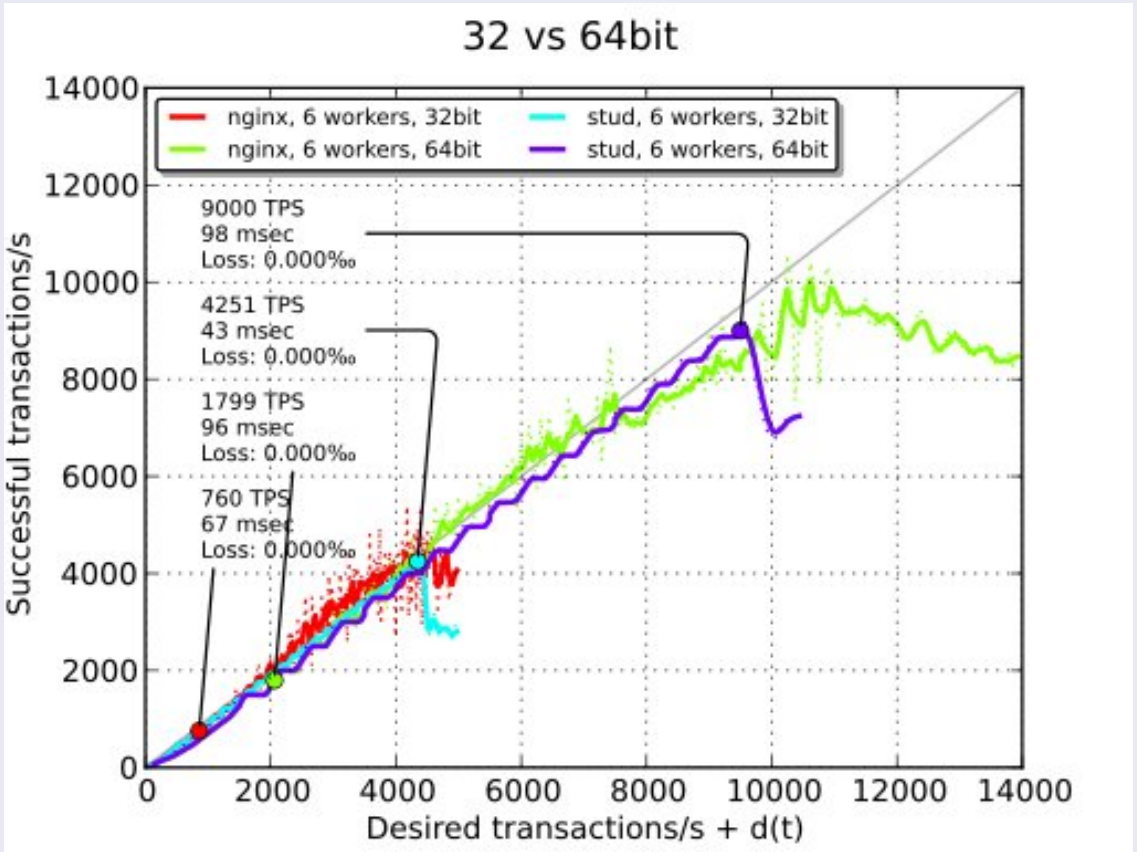


FIGURE: Performance ssl de Nginx

Script de lancement Nginx

Remplacement du script de lancement de nginx par ce script expect

```
#!/usr/bin/expect
set commande $argv
set timeout 60
set fp [open "/chemin/vers/la/pass/phrase" r]
set pass [read $fp]
spawn /etc/init.d/debian-nginx $commande

while 1 {
expect {
        "Enter PEM pass phrase:" {
        send -- "$pass"
                    }
            eof exit
    }
}
interact
```

Assurer la redondance de la solution

Heartbeat3

- Nous avons choisi ici Heartbeat 3. Sa mise en place et son administration sont relativement simples.
- Il fonctionne aussi bien sur des serveurs virtuels *Vmware,OpenVZ* que sur des serveurs rééls.
- Il existe d'autres outils comme keepalive qui peuvent également rendre le même service.

Répartition des vips sur 2 noeuds

```
haproxysrv1:~# crm status
============
Last updated: Thu Sep  1 13:48:32 2011
Stack: Heartbeat
Current DC: haproxysrv2 (298f7e5f-7531-4541-a1db-ea81c7c7004b) - partition with
     quorum
Version: 1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b
2 Nodes configured, unknown expected votes
12 Resources configured.
============

Online: [ haproxysrv1 haproxysrv2 ]
```

Assurer la redondance de la solution

Heartbeat3

- Nous avons choisi ici Heartbeat 3. Sa mise en place et son administration sont relativement simples.

Répartition des vips sur 2 noeuds

```
haproxysrv1:~# crm status
============
Last updated: Thu Sep  1 13:48:32 2011
Stack: Heartbeat
Current DC: haproxysrv2 (298f7e5f-7531-4541-a1db-ea81c7c7004b) - partition with
     quorum
Version: 1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b
2 Nodes configured, unknown expected votes
12 Resources configured.
============

Online: [ haproxysrv1 haproxysrv2 ]

 vipha-site1_192.168.0.200    (ocf::heartbeat:IPaddr2):       Started haproxysrv2
 vipha-site2_192.168.0.201    (ocf::heartbeat:IPaddr2):       Started haproxysrv1
 vipha-site3_192.168.0.202    (ocf::heartbeat:IPaddr2):       Started haproxysrv2
 vipha-site4_192.168.0.203    (ocf::heartbeat:IPaddr2):       Started haproxysrv2
 ....
```

Commandes de bases

Configuration d'une nouvelle ressource :

```
crm configure primitive vipha-site1_192.168.0.200 ocf:heartbeat:IPaddr2 params ip=
     192.168.0.200 cidr_netmask=24 op monitor interval=30s
```

Arrêt et relance d'une ressource

```
crm resource stop vipha-site1_192.168.0.200
crm resource start vipha-site1_192.168.0.200
```

Déplacement d'une ressource sur un autre noeud

```
crm resource move vipha-site1_192.168.0.200 haproxy2
```

Commandes de bases

Suppression d'une ressource

```
crm configure delete vipha-site1_192.168.0.200
```

Arrêt d'un noeud

```
crm node standby
```

Les vips basculent automatiquement sur l'autre noeud.

Redémarrage d'un noeud

```
crm node online
```

Les vips retrouvent leur noeud initial.

Mercurial

Un outil de synchronisation des configurations est nécessaire pour Haproxy et Nginx.

Nous (l'équipe système) avons pris l'habitude de gérer les fichiers de confs avec Mercurial.

Avantages de Mercurial

- Rapidité d'exécution et capacité à gérer de gros projets.
- Utilisation qui ne nécessite pas de serveur centralisé.
- Fonctionnement complètement distribué.

Mercurial

Un outil de synchronisation des configurations est nécessaire pour Haproxy et Nginx.

Nous (l'équipe système) avons pris l'habitude de gérer les fichiers de confs avec Mercurial.

Avantages de Mercurial

- Rapidité d'exécution et capacité à gérer de gros projets.
- Utilisation qui ne nécessite pas de serveur centralisé.
- Fonctionnement complètement distribué.

Script de synchro

Nous avons donc écrit un script qui se base sur mercurial pour propager les configurations et faire un suivi des modifications

```
haproxy:/etc/haproxy# ./synchro.sh --help
usage: synchro.sh -h
  -h|--help           Affiche cette aide
  -U|--update         Lance un hg update sur les 2 serveurs
  -u|--user           Utilisateur qui commit
  -c|--commit         Commit les modifications et pousse sur l'autre serveur
  -m|--message        Message du Commit
  -r|--reload         Recharge la conf sur les 2 serveurs
  -l|--last           Affiche le dernier commit et les dernières modifications
  -a|--auto           Commit, update, push et reload
================================================================================
Exemples:
        Pousse la conf sur les 2 haproxy et la recharge:
                        synchro.sh -a -m "Mon message" -u MonNom
        Pousse la conf sur les 2 haproxy sans recharger haproxy:
                        synchro.sh -c -m "Mon message" -u MonNom
        Compare la conf sur les 2 haproxy:
                        synchro.sh -l
```

Script de synchro

Exemples de commandes et leurs résultats.

```
haproxysrv1:/etc/haproxy# ./synchro.sh -l
Etat des modifications sur haproxysrv1
changeset:   119:01703303042b
tag:         tip
user:        jeanpierre
date:        Tue Aug 30 14:56:39 2011 +0200
summary:     installation de ovale

-------------------------------------
-------------------------------------
Etat des modifications sur haproxysrv2
changeset:   119:01703303042b
tag:         tip
user:        jeanpierre
date:        Tue Aug 30 14:56:39 2011 +0200
summary:     installation de ovale

-------------------------------------
-------------------------------------
```

Script de synchro

Exemples de commandes et leurs résultats.

```
haproxysrv1:/etc/haproxy# ./synchro.sh -a -m "Ajout commentaire" -u gauthier

Je mets a jour les fichiers sur haproxysrv1
0 files updated, 0 files merged, 0 files removed, 0 files unresolved
-------------------------------------
Je commit mes modifications sur haproxysrv1
-------------------------------------

Je pousse la conf sur haproxysrv2
pushing to ssh://haproxysrv2//etc/haproxy
searching for changes
remote: adding changesets
remote: adding manifests
remote: adding file changes
remote: added 1 changesets with 1 changes to 1 files
-------------------------------------
Je mets a jour les fichiers sur haproxysrv2
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
-------------------------------------
Je relance haproxy sur haproxysrv2
Reloading haproxy: haproxy.
-------------------------------------
```

Conclusion

Cette solution nous donne maintenant entière satisfaction. Sa mise en place a, bien sûr, pris un peu de temps, c'est pourquoi nous souhaitons partager notre expérience.

Des questions ?