HTTPD Load Balancer

Michaël Mathieu

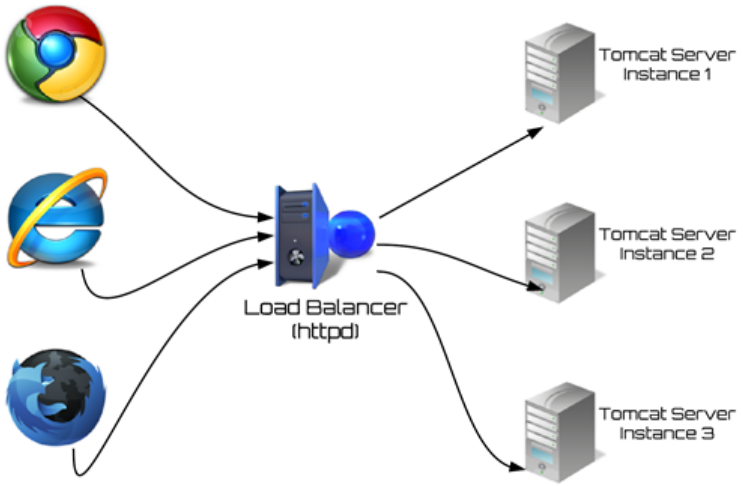
17 septembre 2020

Résumé

Ce labo va nous permettre de mettre en place le concept de performance et de disponibilité au sein d'un serveur web (Apache HTTPD).

⚠Pour réaliser ce labo, il vous faut au minimum 2 ordinateurs. Vous pouvez réaliser ce labo **seul ou à 2 maximum**.

1 Architecture mise en place



2 Environnement technique

Pour ce labo, nous allons dans un premier temps utiliser uniquement le serveur web HTTPD d'Apache.

Dans un deuxième temps, nous utiliserons une application web minimaliste développée en PHP.

Ce labo a été testé avec la version `17.0 devserver` d'*easyPHP* (`http://www.easyphp.org/`).

3 Activation du load balancer

Nous allons commencer par configurer la machine qui fera office de **load balancer**.

La documentation de référence pour configurer le load balancer est [1]. Deux autres sites ont été utiles à la configuration [3] et [2].

⚠il est inutile de configurer ainsi les autres machines du clusters !

Dans le fichier `easyPHP/eds-binaries/httpserver/apache.../conf/httpd.conf`, nous devons :

1. Nous assurer que les modules suivants sont chargés au démarrage du serveur :
   - `LoadModule lbmethod_bybusyness_module modules/mod_lbmethod_bybusyness.so`
   - `LoadModule lbmethod_byrequests_module modules/mod_lbmethod_byrequests.so`
   - `LoadModule lbmethod_bytraffic_module modules/mod_lbmethod_bytraffic.so`
   - `LoadModule lbmethod_heartbeat_module modules/mod_lbmethod_heartbeat.so`
   - `LoadModule proxy_module modules/mod_proxy.so`
   - `LoadModule proxy_balancer_module modules/mod_proxy_balancer.so`
   - `LoadModule proxy_http_module modules/mod_proxy_http.so`
   - `LoadModule slotmem_shm_module modules/mod_slotmem_shm.so`
2. Ajouter en fin de fichier, la configuration suivante :

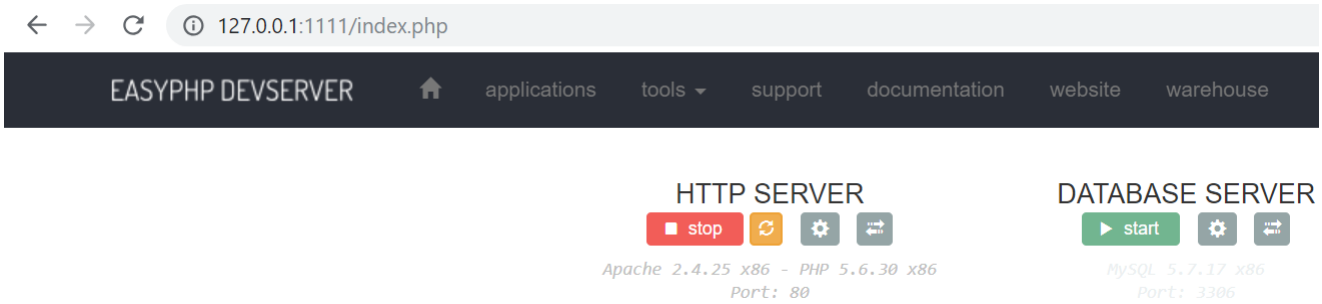
```
#Load Balancer Configuration
<IfModule proxy_balancer_module>
  <Location "/balancer-manager">
    SetHandler balancer-manager
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
# unsecure (instead of previous line): Allow from all
  </Location>

  <Proxy balancer://mybalancer>
    BalancerMember http://server1:8080 loadfactor=1
    BalancerMember http://server2:8080 loadfactor=1
    BalancerMember http://localhost:80 loadfactor=1
    ProxySet lbmethod=byrequests
  </Proxy>

  ProxyPass /balance balancer://mybalancer/test-lb
</IfModule>
```

Avant de démarrer ce serveur, ouvrez le fichier de logs (`easyPHP/eds-binaries/httpserver/apache.../logs/error.log`), effacez son contenu (afin de ne pas être pollué par d'anciens logs), puis démarrez le serveur.

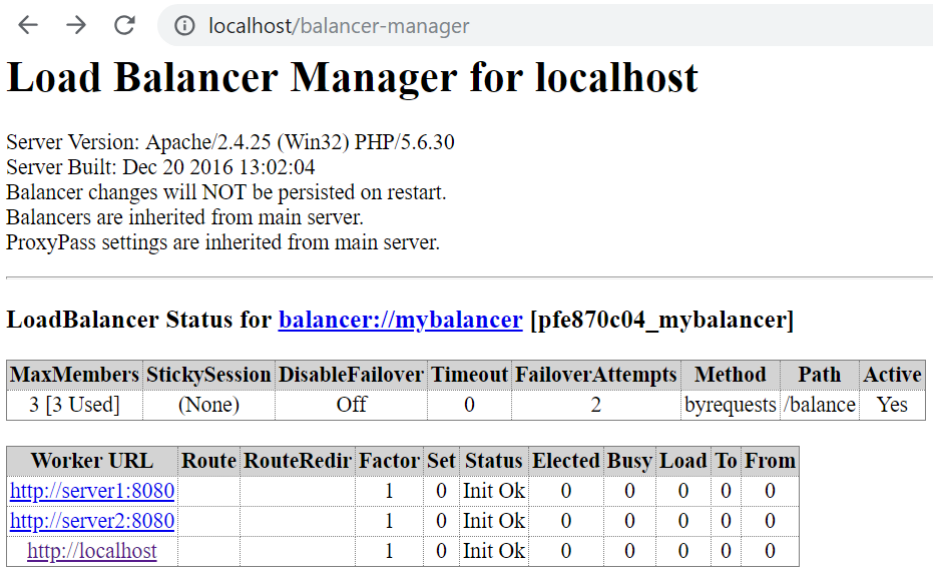
Assurez–vous qu'il n'y a aucune erreur et que le serveur est démarré :



Explication de la configuration que nous avons mis en fin de fichier :

1. Chargement de l'interface du `balancer-manager` qui permet de monitorer l'activité du load balancer
2. Cette interface n'est disponible que depuis l'interface locale (127.0.0.1)
3. Un cluster `mybalancer` est créé
4. Dans ce cluster, les serveurs pouvant traiter des requêtes sont `server1`, `server2`, `localhost`
5. Le choix des serveurs est basé selon la stratégie "*par nombre de requêtes*" (voir `https://httpd.apache.org/docs/trunk/mod/mod_lbmethod_byrequests.html`)
6. Sur les serveurs, l'application est déployée sous `http://x.x.x.x/test-lb`
7. L'adresse du load balancer que les clients doivent atteindre afin de profiter du mécanisme de load balancing est `http://x.x.x.x/balance`

Vous pouvez vous connectez sur `http://localhost/balancer-manager` afin de voir l'interface du `balancer-manager`.



Observez le comportement du load balancer lorsque vous faites plusieurs appels à `http://x.x.x.x/balance`.

Il existe 3 autres stratégies pour choisir quel serveur du cluster traitera la demande :
— `bytraffic`
(`https://httpd.apache.org/docs/trunk/mod/mod_lbmethod_bytraffic.html`)
— `bybusyness`
(`https://httpd.apache.org/docs/trunk/mod/mod_lbmethod_bybusyness.html`)
— `heartbeat`
(`https://httpd.apache.org/docs/trunk/mod/mod_lbmethod_heartbeat.html`)

Quelles différences notez–vous entre ces différentes stratégies ? Si vous deviez mettre en ligne un site marchand, laquelle de ces stratégie metteriez–vous en place et pourquoi ?

Essayez d'ajouter comme membres du cluster des machines qui n'existent pas sur votre réseau. Que se passe–t–il ?

4 Sticky session

Si nous n'implémentons pas la réplication de session avec un outil comme *Memcached* (`http://php.net/manual/fr/book.memcached.php`), ce qui est généralement le cas pour des raisons de performances, nous devons nous assurer, à partir du moment qu'une session est créée, que toutes les futures requêtes d'un utilisateur soient traitées par le même serveur (sinon l'utilisateur devra se reconnecter sur chaque serveur traitant une de ses requêtes afin qu'une session soit créée pour lui).

Dans le cas d'une application PHP, nous allons indiquer au load balancer de se baser sur le cookie `PHPSESSID` pour la sticky session
(voir `https://httpd.apache.org/docs/trunk/mod/mod_proxy.html`).

Si nous reprenons l'exemple de configuration précédent, nous aurions :

```
<Proxy balancer://mybalancer>
  ...
  ProxySet stickysession=PHPSESSID
</Proxy>
```

Après avoir redémarré le serveur, vous pouvez tester le mécanisme de *sticky session* avec le code PHP suivant :

test-lb/index.php

```
<?php
  session_start();

  echo "<html><body>";
  echo "served by " . $_SERVER["REMOTE_ADDR"] . "<br>";

  if (isset($_SESSION['data'])) {
    echo $_SESSION['data'];
  } else {
    echo "NEW SESSION CREATED";
    $_SESSION['data'] = "CONNECTED";
  }
?>
    <br><br>
    <a href="logout.php">Delete session</a>
  </body>
</html>
```

test-lb/logout.php

```
<?php
  session_start();

  echo "<html><body>";
  echo "served by " . $_SERVER["REMOTE_ADDR"] . "<br>";

  //remove PHPSESSID from browser
  if (isset($_COOKIE[session_name()])) {
    setcookie(session_name(), "", time() - 3600, "/" );
  }

  //clear session from globals
  $_SESSION = array();

  //clear session from disk
  session_destroy();
?>
    LOGGED OUT<br>
    <br>
    <a href="index.php">Create session</a>
  </body>
</html>
```

Références

[1] Apache. Module mod_proxy_balancer. https://httpd.apache.org/docs/trunk/mod/mod_proxy_balancer.html.

[2] Satish. Load balancing with apache mod_proxy_balancer. https://satishkumars.wordpress.com/2012/03/07/how-to-load-balance-tomcat-7-with-apache-2-2/.

[3] Rackspace Support. Simple load balancing with apache. https://support.rackspace.com/how-to/simple-load-balancing-with-apache/.