Configurer le service Web pour gérer des sites Web virtuels

Présentation

Vous allez modifier la configuration de base du service web Apache installé sur le serveur Linux Debian INTRALAB (172.16.0.100).

Ce document se base :

- sur un serveur Web ayant l'**adresse IP 172.16.0.100** et le nom **INTRALAB** ;
- sur un serveur DNS ayant l'**adresse IP 172.16.0.30** et le nom **LABANNU** Il faut **adapter** ces informations à votre propre configuration IP.

A partir d'un navigateur, vérifiez que vous accédez au site par défaut du serveur avec l'URL http://172.16.0.100.

En utilisant le serveur DNS que vous avez installé vous pouvez maintenant configurer votre serveur DNS afin d'utiliser l'URL **http://apache.gsb.local.**

Premiers tests

- Vérifiez la connectivité entre votre serveur Linux et l'ordinateur à partir duquel vous lancez votre navigateur, avec l'utilitaire Ping ;
- Vérifiez que votre serveur Linux est bien bien client du service DNS. Le fichier **/etc/resolv.conf** devrait ressembler à : <code> root@intralab:~# cat /etc/resolv.conf domain gsb.local search gsb.local nameserver 172.16.0.30 </code> Pour modifier la configuration DNS d'un client Linux, modifiez le fichier **/etc/resolv.conf** qui devrait ressembler à : root@clientlionux:~# cat /etc/resolv.conf domain gsb.local search gsb.local nameserver 172.16.0.30

A partir de votre serveur Linux, testez le bon fonctionnement de la résolution de nom avec nslookup :

```
root@intralab:~# nslookup intralab
Server:         172.16.0.30
Address:        172.16.0.30#53
Name:   intralab.gsb.local
Address: 172.16.0.100
```

- Résolution sur le domaine Internet :

```
root@intralab:~# nslookup www.google.fr
Réponse ne faisant pas autorité :
Nom :    www-cctld.l.google.com
Address:  173.194.66.94
Aliases:  www.google.fr
```

Si tout tout fonctionne, vous pouvez continuer.

- Renseigner dans votre serveur DNS un alias **www** avec comme IP celle de votre serveur Web **INTRALAB 175.16.0.100.**

Configuration du serveur

Les fichiers de configuration d'Apache sont dans le dossier **/etc/apache2** :

```
root@debianbtssio:~# ls /etc/apache2
apache2.conf  envvars     mods-available  ports.conf       sites-enabled
conf.d        httpd.conf  mods-enabled    magic          sites-available
```

La configuration du serveur proprement dit, des sites et des modules se fait dans de nombreux fichiers. Installez le paquet **tree** si nécessaire afin de visualiser l'arborescence des dossiers :

```
root@debianbtssio:~# tree /etc/apache2
```

Hôtes virtuels

Les hôtes virtuels sont régulièrement utilisés sur les serveurs Web :

- chaque serveur héberge plusieurs sites web : il faut donc des hôtes virtuels ;
- vous n'utiliserez que quelques directives de configuration.

Par défaut, le serveur répond à l'adresse **www.gsb.local**. L'installation d'Apache sous Debian propose un site par défaut pré-configuré. Ses paramètres sont dans le fichier **/etc/apache2/sites-enabled/000-default**.

Pour l'instant, n'importe quelle requête http qui arrive sur le port 80 de ce serveur sera dirigé vers ce site. Si l'on veut installer différents sites, il faut configurer Apache de façon à diriger les requêtes http vers le bon site.

Requêtes http

Pour comprendre le mécanisme d'hôtes virtuels, il faut bien comprendre comment sont constituées les requêtes http. Comme pour la plupart des services réseau TCP/IP, les échanges entre client et serveur se font en mode texte.

Rapide historique :

Au début du protocole http (version http 0.9) il n'y avait qu'une seule méthode (la méthode GET) permettant d'obtenir des pages au format uniquement texte. On peut le mettre en oeuvre et simuler une requête envoyée par un navigateur au serveur. A partir de votre serveur Linux (mais c'est également possible à partir d'une invite de commandes Windows), tapez la commande suivante :

```
telnet www.btssio.local 80
Trying 192.168.1.50...
Connected to www.btssio.local (192.168.1.50).
Escape character is '^]'.
```

Ensuite tapez : GET /

```
GET /
<html><body><h1>It works!</h1></body></html>
Connection closed by foreign host.
```

Vous obtenez la page puis la connexion est fermée. Avec la version 1.0, il y a des nouveautés :

- nouvelles méthodes : **HEAD** (informations sur la ressource) et surtout **POST** (envoyer des données au serveur)
- les requêtes peuvent contenir des paramètres (informations sur le navigateur, préférences du navigateur par exemple)
- les réponses peuvent aussi contenir des paramètres (et non plus seulement la ressource elle-même) :
    - code réponse : 200 pour une requête réussie, 404 pour une ressource inexistante, 403 pour un accès interdit
    - Date de création de la ressource, taille, date d'expiration, informations sur le serveur, etc.

Pour comprendre :

```
telnet www.gsb.local 80
Trying 192.168.1.50...
Connected to www.btssio.local (192.168.1.50).
Escape character is '^]'.
```

Commande à tapez ...	**Explications ...**
GET / HTTP/1.0	Requête du client indiquant la version du protocole
	Taper deux fois sur ENTREE
HTTP/1.1 200 OK	Version du protocole supporté par le serveur suivi du code status HTTP (200 = OK, la ressource est disponible dans la réponse)
Date: Sat, 05 May 2012 22:34:19 GMT	En-têtes associés à la réponse
Server: Apache/2.2.16 (Debian) Last-Modified: Sat, 06 Aug 2011 18:11:21 GMT ETag: “230dc-2d-48125c2558440” Accept-Ranges: bytes Content-Length: 45 Vary: Accept-Encoding Connection: close Content-Type: text/html	
<html><body> It works! </body></html>	La ressource elle-même
Connection closed by foreign host.	

Tous les détails sur le protocole HTTP 1.0 sont définis dans la RFC1945.

Ce protocole présentait néanmoins des limites, en particulier :

- un seul site par serveur
- pas de persistance des connexions : une connexion est ouverte à chaque requête, ce qui consomme des ressources alors que bien souvent un utilisateur consulte plusieurs pages sur le même site.

Le HTTP 1.1 (RFC 2616) apporte des améliorations et permet en particulier d'héberger plusieurs sites sur un même serveur. Dans la

requête, le navigateur doit donc intégrer le nom du site demandé sur le serveur :

```
telnet www.gsb.local 80
Trying 172.16.0.100...
Connected to 05 May 2012 (192.168.1.50).
Escape character is '^]'.
GET / HTTP/1.1
Host:  www.gsb.local
```

Remarquez que la connexion n'est pas fermée par le serveur.

Configuration de deux hôtes virtuels Vous allez configurer deux sites différents :

- site1.gsb.local et site2.gsb.local

Le site par défaut restera accessible. Dans un premier temps, **modifiez** votre serveur DNS pour faire la résolution de noms sur ces deux machines (créez **deux alias** pour www.gsb.local).

Dans le répertoire **/etc/apache2/sites-available**, créez le fichier nommé site1.gsb.local avec le contenu suivant :

```
<VirtualHost *:80>
        ServerName site1.gsb.local
        DocumentRoot /var/www/site1.gsb.local
       ErrorLog /var/log/apache2/error.log
        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn
        CustomLog /var/log/apache2/site1.gsb.local.access.log combined
</VirtualHost>
```

La directive **ServerName** identifie l'hôte virtuel. La directive **DocumentRoot** indique la racine où sont situés les fichiers du site web. Dans **/var/www**, créez le répertoire **site1.gsb.local** puis, dans ce répertoire, créez le fichier **index.html** ci-dessous :

```
<h1>Site1</h1>
```

Au démarrage, Apache tourne sous l'utilisateur **root** mais ensuite il se divise en sous-processus. Ce sont ces sous-processus qui servent les pages html. Ils tournent quant eux sous le compte de l'utilisateur **www-data** :

```
# ps aux | grep apache2
root      4203  0.0  1.0  13376  2784 ?        Ss   00:18   0:01 /usr/sbin/apache2 -k start
www-data  4204  0.0  0.7  13148  1988 ?        S    00:18   0:00 /usr/sbin/apache2 -k start
www-data  4205  0.0  1.2 234872  3160 ?        Sl   00:18   0:00 /usr/sbin/apache2 -k start
www-data  4210  0.0  1.1 234720  3076 ?        Sl   00:18   0:00 /usr/sbin/apache2 -k start
```

Il est donc conseillé d'adapter les droits d'accès :

```
intralab:~# cd /var/www
intralab:/var/www# ls -la
intralab:/var/www# chown www-data . -R
intralab:/var/www# chgrp www-data . -R
intralab:/var/www# ls -la
```

Il nous reste maintenant à **activer** le site et à recharger la configuration d'Apache :

```
# a2ensite site1.gsb.local
# /etc/init.d/apache2 reload
```

Dans votre navigateur, tapez **http://site1.gsb.local**. La page « Site1 » doit s'afficher. Vous pouvez faire de même avec le site 2 en adaptant le contenu des fichiers et les commandes.

Principales directives Voici les principales directives utilisées habituellement. Une chose importante à savoir est qu'une configuration réalisée à un certain niveau de l'arborescence des sites se propage à tous les fichiers et répertoires situés en dessous.

DirectoryIndex Par défaut, si un répertoire du site web ne contient pas de fichier **index.html**, n'importe qui peut lister son contenu :

```
# cd /var/www/site2.gsb.local
# mkdir rep
```

Maintenant dans le navigateur, si l'on va sur **http://site2.gsb.local/rep**, le contenu du répertoire est affiché.

Pour empêcher ceci, dans le fichier de configuration du site **/etc/apache2/sites-available/site2.gsb.local**, ajoutez juste avant la fin du fichier :

```
        <Location />
                Options -Indexes
        </Location>
</VirtualHost>
```

Ce qui signifie : à partir de la racine (/), l'affichage des répertoires est interdit (-). On recharge la configuration d'Apache, le résultat dans le navigateur est différent.

Redirection On est fréquemment amené à faire des redirections. L'utilisateur arrive à un endroit du site et il est redirigé soit vers une autre page soit vers un autre site.

Supposons que **Site2** soit momentanément indisponible. Nous voulons rediriger l'utilisateur vers **Site1**. Commençons par ceci :

```
        Redirect temp / http://site1.gsb.local/
</VirtualHost>
```

Après rechargement de la configuration, si on essaie d'aller sur **http://site2.gsb.local/rep** on est redirigé au même endroit sur l'autre URL **http://site1.gsb.local/rep** mais ce répertoire n'existe pas.

On peut faire mieux en disant que quel que soit l'endroit où l'utilisateur arrive sur le **site2** il est systématiquement redirigé à la racine du site1. On remplace l'instruction par :

```
RedirectMatch temp /* http://site1.btssio.local/
```

Vous pouvez observer la redirection ici :

```
$ telnet 172.16.0.100 80
Trying 172.16.0.100...
Connected to www.gsb.local (172.16.0.100).
Escape character is '^]'.
GET / HTTP/1.1
Host: site2.gsb.local

HTTP/1.1 302 Found
Date: Sun, 07 Mar 2010 00:34:11 GMT
Server: Apache/2.2.16 (Debian)
Location: http://site1.gsb.local/
Vary: Accept-Encoding
Content-Length: 291
Content-Type: text/html; charset=iso-8859-1
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://site1.gsb.local/">here</a>.</p>
<hr>
<address>Apache/2.2.16 (Debian) Server at site2.gsb.local Port 80</address>
</body></html>
Connection closed by foreign host.
```

Le serveur vous répond FOUND (302) mais vous indique un autre endroit (Location). C'est le navigateur qui utilise cet en-tête de la réponse pour y aller. Référence : http://httpd.apache.org/docs/2.2/mod/mod_alias.html

Alias L'alias est un mécanisme différent de la redirection. Il s'agit de la notion de répertoire virtuel. On veut par exemple donner accès à un répertoire sur le serveur situé ailleurs que dans le **DocumentRoot**. Vous verrez par la suite qu'un produit comme **Phpmyadmin** par exemple, ne s'installe pas dans **/var/www** mais ailleurs. Or, le serveur web ne peut pas renvoyer une ressource située en dehors de son **DocumentRoot**, sauf si on utilise un alias. Par exemple :

```
Alias /phpmyadmin /usr/share/phpmyadmin
```

Lorsque l'utilisateur demande **http://site1.gsb.local/phpmyadmin**, ce sont des fichiers situés dans **/usr/share/phpmyadmin** qui sont servis.

Référence : http://httpd.apache.org/docs/2.2/mod/mod_alias.html

Restriction sur IP Il arrive régulièrement, pour des raisons de sécurité, de réserver l'accès à certains sites ou parties de site à certaines

adresses IP.

Travaillons sur le **site1** et imaginons que celui-ci n'est accessible qu'au réseau IP **172.16.0.0**.

```
<Location />
        Order Deny,Allow
        Deny from All
        Allow from 172.16
</Location>
</VirtualHost>
```

Si vous essayez d'accéder au site1, vous aurez un **Forbidden** (status HTTP 403). Référence : http://httpd.apache.org/docs/2.2/mod/mod_authz_host.html

Accès par mot de passe Une autre manière de protéger l'accès à un site est de demander un mot de passe. Notez bien qu'en http, les mots de passes circulent en clair entre le navigateur et le serveur.

Sur le **site1**, supprimez la restriction sur l'IP. Puis créez le fichier de mot de passe des utilisateurs autorisés (ce sont donc des comptes indépendants des comptes Unix).

```
# cd /etc/apache2/
debianbtssio:/etc/apache2# mkdir passwd
debianbtssio:/etc/apache2# cd passwd
debianbtssio:/etc/apache2# htpasswd -c /etc/apache2/passwd/passwords util2
New password:
Re-type new password:
Adding password for user util2
```

Dans le fichier de configuration du **site1** :

```
        <Location />
                AuthType Basic
                AuthName "Acces protege"
                AuthBasicProvider file
                AuthUserFile /etc/apache2/passwd/passwords
                Require valid-user
        </Location>
</VirtualHost>
```

Lorsque l'on accède au site, on obtient une fenêtre d'authentification. Si ce n'est pas le cas, il faut probablement vider votre cache ou forcer le rechargement de la page avec CTRL+F5 : Référence : http://httpd.apache.org/docs/2.2/howto/auth.html

Étalonnage Comment répondre à des questions comme « combien de requêtes http mon serveur est-il capable de gérer ? » ou « est-ce que cette nouvelle configuration de mon serveur web est plus efficace ? ». La commande **ab** peut nous donner des éléments. Pour plus de détails consultez la documentation disponible avec la commande **man**. Lançez la commande ci-dessous qui va réaliser 100 000 requêtes http, 100 à la fois sur le serveur web local :

```
# ab -n 100000 -c 100 http://localhost/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
Benchmarking localhost (be patient)
Completed 10000 requests
...
Finished 100000 requests
```

Suivant votre configuration matérielle, ce travail prendra un temps plus ou moins long. Exemple de résultat :

```
Server Software:        Apache/2.2.16
Server Hostname:        localhost
Server Port:            80
Document Path:          /
Document Length:        45 bytes
Concurrency Level:      100
Time taken for tests:   404.186 seconds
Complete requests:      100000
Failed requests:        0
Write errors:           0
Total transferred:      35714280 bytes
```

```
HTML transferred:       4501800 bytes
Requests per second:    247.41 [#/sec] (mean)
Time per request:       404.186 [ms] (mean)
Time per request:       4.042 [ms] (mean, across all concurrent requests)
Transfer rate:          86.29 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        1  185  55.3    173     712
Processing:    75  218  70.8    198    1049
Waiting:       41  175  62.0    162     884
Total:        150  403  95.9    379    1446

Percentage of the requests served within a certain time (ms)
  50%    379
  66%    411
  75%    437
  80%    455
  90%    516
  95%    582
  98%    692
  99%    774
 100%   1446 (longest request)
```

On cherche bien sûr à avoir la plus grande valeur possible pour « **Requests per seconds** » et la plus faible pour « **Time per request** ». Mais ces outils sont à manipuler avec précaution car il faut pouvoir « comparer ce qui est comparable ». En effet, suivant le lien réseau qui vous sépare du serveur, la taille des pages demandées, la charge actuelle du serveur si celui-ci est en production, tous ces éléments et certainement d'autres peuvent grandement influencer les résultats.

Dépannage La commande **apache2ctl –t** permet de valider la syntaxe des fichiers de configuration.

Visionnez régulièrement les fichiers **/var/log/apache2** afin de voir l'historique des accès et des erreurs.

Poursuivre l'activité

J'ai terminé cette étape et je continue l'activité : Installer et configurer le service Web(contexte GSB)