Les hôtes virtuels

> Bow ties are cool.
>
> Le onzième docteur

L'hébergement virtuel est une technique permettant d'héberger plusieurs sites Web sur un seul serveur Web. Ces sites sont différenciés soit par des noms de domaines différents, soit par des adresses IP ou des ports différents.

Cette fonctionnalité est bien entendue supportée sous Apache, qui a été l'un des premiers serveurs Web à la mettre en œuvre, et par Nginx.

Plusieurs techniques peuvent être utilisée pour réaliser ces hébergements virtuels :

- *Les hôtes virtuels par adresse IP* : les requêtes sont alors triées en fonction de l'adresse IP qu'elles utilisent. Chaque hôte virtuel dispose de sa propre adresse IP et répond alors aux requêtes concernant son adresse.
- *Les hôtes virtuels par port* : les requêtes sont alors triées en fonction du port qu'elles utilisent. Chaque hôte virtuel dispose de son propre port et répond alors aux requêtes concernant son adresse. Cette technique n'est quasiment jamais utilisée, car utiliser un port différent des ports standards (80 et 443) nécessite de le spécifier dans l'adresse du site (`http://example.org:8081/foo/bar`, par exemple).
- *Les hôtes virtuels par nom* : cette technique repose sur l'évolution apportée par la norme HTTP 1.1, qui prévoit d'ajouter un champ « `Host` » dans les en-têtes HTTP. Ce champ est alors utilisé pour identifier l'hôte virtuel qui est concerné par une requête donnée.
- *Les hôtes virtuels dynamiques* : qui ne possèdent pas de configuration propre, mais qui sont déterminés à partir de l'URL de la requête. Il est alors possible en utilisant quelques règles de décrire le comportement d'un ensemble d'hôtes virtuels.

1 Apache

1.1 Généralités

D'une manière générale, toutes les directives concernant un hôte virtuel particulier sont placées dans des conteneurs `<VirtualHost>`. Toute directive placée dans un tel conteneur ne s'applique donc qu'à l'hôte virtuel correspondant.

Il est également important de noter que les directives placées dans le contexte de configuration globale sont héritées dans les hôtes virtuels, sauf si elles y sont surchargées.

Avant de définir des hôtes virtuels, il est important de définir des directives `Listen`, indiquant sur quelles adresses IP et sur quels ports Apache doit attendre des requêtes (fichier `/etc/apache2/ports.conf` sur Debian).

```
Listen 80
```

=> Apache écoutera sur le port 80 de toutes les adresses IP du serveur (comportement par défaut avec la configuration par défaut de la plupart des distributions).

Il est possible de définir plusieurs directives `Listen`, comme cela est indiqué dans les exemples ci-dessous :

```
Listen 192.168.1.100:80
Listen 192.168.1.101:1337
Listen 4242
```

=> Apache écoutera sur le port 80 de l'adresse 192.168.1.100, sur le port 1337 de l'adresse 192.168.1.101 et sur le port 4242 de toutes les adresses IP du serveur.

1.2 Syntaxe de la directive `<VirtualHost>`

La syntaxe est :

```
<VirtualHost adresse IP[:port] [adresse IP[:port]] ...>
```

On peut par exemple écrire :

```
# Sans spécification d'adresse IP ni de port d'écoute
<VirtualHost *>
<VirtualHost *:*>
# Spécification du port
<VirtualHost *:80>
# Spécification de l'adresse IP et port
<VirtualHost 192.168.1.100:80>
# Spécification de l'adresse IP mais pas du port
<VirtualHost 192.168.1.100>
<VirtualHost 192.168.1.100:*>
```

On peut spécifier plusieurs adresses IP ou plusieurs ports pour un seul hôte virtuel. Par exemple :

```
<VirtualHost 192.168.1.100:80 192.168.1.100:1337 192.168.1.101>
  ...
</VirtualHost>
```

Dans le cas de l'utilisation d'IPv6, il faudra mettre l'adresse IP entre crochets quand on la spécifie :

```
Listen [2001:db8:a::42]:80
<VirtualHost [2001:db8:a::42]:80>
```

1.3 Hôtes virtuels par adresse IP

Cette méthode était la méthode la plus utilisée avant la possibilité de créer des hôtes virtuels par nom.

Les hôtes virtuels par adresse IP doivent être chacun associé à une adresse IP unique.

On s'assurera que la directive `Listen` d'Apache est positionnée comme suit afin d'écouter sur toutes les IPs :

```
Listen 80
```

NB : la commande `ip`, contenue dans le paquet `iproute2`, est plus puissante et plus simple à utiliser qu'`ifconfig`, dont l'usage est aujourd'hui obsolète, comme le montre sa page de man en version anglaise [1]. Oubliez `ifconfig` si jamais c'est que vous êtes habituée à utiliser, préférez la commande `ip`. Pour connaître quelques commandes utiles sur `ifconfig` et `ip`, consultez la page https://p5r.uk/blog/2010/ifconfig-ip-comparison.html.

On ajoute des adresses IP à une interface réseau :

```
ip addr add 192.168.1.100 dev eth0
ip addr add 192.168.1.101 dev eth0
```

1. https://linux.die.net/man/8/ifconfig

La configuration de ce type d'hôte se fait par une directive conteneur `<VirtualHost *adresse_IP*>`, placée après les directives du contexte de configuration globale.

La configuration d'un hôte virtuel se fait généralement dans un fichier différent de celui la configuration générale, qui comporte alors une directive `Include repertoire_du_fichier/*` pour prendre en compte l'hôte virtuel. Dans Debian, comme vu dans le TP 1, les fichiers des hôtes virtuels sont à placer dans le répertoire `/etc/apache2/sites-available/`, avec un lien symbolique de ces fichiers dans le dossier `/etc/apache2/sites-enabled/` (créé avec la commande `a2ensite`).

Il est ainsi possible, pour continuer l'exemple précédent, de faire les définitions suivantes :

```
<VirtualHost 192.168.1.100>
  DocumentRoot /var/www/vhost1
</VirtualHost>
<VirtualHost 192.168.1.101>
  DocumentRoot /var/www/vhost2
</VirtualHost>
```

Rappel : `DocumentRoot` permet de définir la racine des documents distribués par Apache (le répertoire où chercher les fichiers).

Pour ce type d'hôte virtuel, le nom de domaine (défini par la directive `ServerName`) n'est alors d'aucune importance, étant donné qu'Apache se base uniquement sur l'adresse IP pour déterminer quel hôte virtuel est concerné par une requête.

Cependant, si ces adresses IP ne sont pas résolvables en un nom de domaine, soit via une interrogation DNS, soit via le fichier `/etc/hosts`, Apache affichera une erreur au redémarrage ou à la vérification de la syntaxe (`apachectl -t`), mais celle-ci n'est pas une erreur bloquante.

Exemple d'erreur :

```
Name or service not known: AH00549: Failed to resolve server name \
    for 192.168.1.100 (check DNS) -- or specify an explicit ServerName
Name or service not known: AH00549: Failed to resolve server name \
    for 192.168.1.101 (check DNS) -- or specify an explicit ServerName
```

On peut mettre dans `/etc/hosts`, pour éviter cette erreur :

```
192.168.1.100 dalek.example.org
192.168.1.101 cybermen.example.org
```

Ou alors configurer la directive `ServerName` dans la configuration des hôtes virtuels :

```
<VirtualHost 192.168.1.100>
  DocumentRoot /var/www/vhost1
  ServerName dalek.example.org
</VirtualHost>
<VirtualHost 192.168.1.101>
  DocumentRoot /var/www/vhost2
  ServerName cybermen.example.org
</VirtualHost>
```

Mais c'est bien l'adresse IP associée au nom de domaine qui est déterminante dans le choix de l'hôte virtuel.

Bien entendu, pour que cette technique de serveurs virtuels par adresse IP soit mise en place, il faut que le FAI fournisse plusieurs IPs (ce qui ne pose pas de souci pour l'IPv6).

Si le serveur Web est contacté par une requête comportant une adresse IP ne correspondant à aucun hôte virtuel, c'est le serveur principal qui répondra à la requête. Vous pouvez tester ce comportement simplement en désactivant tous vos hôtes virtuels et en faisant une requête sur votre serveur web : le serveur vous répondra bien. C'est ce site par défaut qui est appelé *serveur principal*.

Pour changer ce comportement, il est alors possible de définir un hôte virtuel par défaut permettant de mieux traiter l'erreur rencontrée. Cet hôte virtuel par défaut se définit au moyen de la directive `<VirtualHost>` associé à la valeur spéciale `_default_` : `<VirtualHost _default_>`.

1.4 Hôtes virtuels par port

Tout comme on peut créer des hôtes virtuels en discriminant sur l'adresse IP utilisée, on peut utiliser le port utilisé pour créer différents Hôtes virtuels.

La directive `Listen` sera positionnée comme suit :

```
Listen 80
Listen 1337

<VirtualHost *:80>
  ServerName dalek.example.org
  DocumentRoot /var/www/vhost1
</VirtualHost>
<VirtualHost *:1337>
  ServerName cybermen.example.org
  DocumentRoot /var/www/vhost2
</VirtualHost>
```

Là encore, la directive `ServerName` ne sert à rien.

Si le serveur Web est contacté par une requête comportant un port ne correspondant à aucun hôte virtuel (mais défini dans `Listen`), c'est le serveur principal qui répondra à la requête. Pour changer ce comportement, il est alors possible de définir un hôte virtuel par défaut permettant de mieux traiter l'erreur rencontrée. Cet hôte virtuel par défaut se définit au moyen de la directive `<VirtualHost>` associé au caractère `*` pour l'adresse IP et éventuellement le port : `<VirtualHost *>` ou `<VirtualHost *:*>`.

1.5 Hôtes virtuels par nom

Cette fonctionnalité repose sur les « nouvelles » fonctionnalités proposées par HTTP 1.1, et en particulier le support du champ `Host`.

Apache 2.2 uniquement

Pour activer le support d'hôtes virtuels par nom, il faut utiliser la directive `NameVirtualHost` en lui passant en paramètre l'adresse IP du serveur (si le serveur a plusieurs addresses, on mettra plusieurs directives `NameVirtualHost`).

```
NameVirtualHost 192.168.1.102
```

Il ne peut y avoir qu'une seule directive `NameVirtualHost` pour une adresse IP donnée, mais plusieurs hôtes virtuels peuvent être définis pour une adresse IP donnée. La directive `NameVirtualHost` est obsolète depuis la version 2.4 d'Apache et n'est donc plus nécessaire sur Apache 2.4.

L'hôte virtuel souhaité est déterminé par le champ `Host` de l'en-tête HTTP de la requête.

Si aucune correspondance n'est trouvée pour l'hôte demandé par le client, c'est le premier hôte virtuel (dans l'ordre de lecture des fichiers de configuration) défini pour cette adresseIP qui est utilisé. On appelle pour cette raison celui-ci *hôte virtuel primaire*.

Voici des exemples de définition :

```
<VirtualHost 192.168.1.102>
  ServerName slitheen.example.org
  DocumentRoot /var/www/vhost1
</VirtualHost>
<VirtualHost 192.168.1.102>
  ServerName sontarans.example.org
  ServerAlias dalek.example.org cybermen.example.org
  ServerAlias *.enemies.example.org
  DocumentRoot /var/www/vhost2
</VirtualHost>
```

Pour définir des noms de domaine supplémentaires en plus de celui défini par la directive `ServerName` on utilisera la directive `ServerAlias` qui peut prendre un ou plusieurs noms de domaine en argument et supporte les caractères génériques comme `*`.

En vertu de la règle énoncée ci-dessus, le premier des hôtes virtuels définis est généralement conçu pour traiter les cas d'erreur [2]. Dans l'exemple ci-dessus, ces cas arriveront pour toute requête formulée auprès de l'adresse 192.168.1.102 comportant un champ `Host` ne correspondant à aucun hôte virtuel défini.

1.6 Hôtes virtuels dynamiques

Pour les serveurs Web hébergeant des centaines ou des milliers de sites Web différents, les techniques présentées ci-dessus ne sont pas toujours suffisantes [3].

Il est alors possible de définir des hôtes virtuels dynamiques, dont le support est assuré par le module `mod_vhost_alias`. Ces hôtes reposent alors sur la disponibilité de 4 directives :

- Pour les hôtes dynamiques par nom : `VirtualDocumentRoot`, indiquant un `DocumentRoot` construit à partir de l'URL de la requête, et `VirtualScriptAlias`, indiquant comment construire le nom du répertoire contenant des scripts CGI (voir la directive `ScriptAlias`) à partir de l'URL de la requête.
- Pour les hôtes dynamiques par adresse : `VirtualDocumentRootIP`, indiquant comment construire `DocumentRoot` à partir de l'adresse IP de la requête, et `VirtualScriptAliasIP`, indiquant comment construire le nom du répertoire contenant des scripts CGI (voir la directive `ScriptAlias`) à partir de l'adresse IP de la requête.

Avec ces directives, il n'est alors plus nécessaire de créer des directives `<VirtualHost>` pour chaque hôte virtuel à décrire. Chacune des 4 directives utilise un ensemble de spécificateurs permettant d'extraire des éléments de l'URL de la requête pour déterminer `DocumentRoot` et `ScriptAlias` (voir la documentation [4]).

Ainsi, par exemple, avec la configuration suivante :

```
UseCanonicalName Off
VirtualDocumentRoot /usr/local/apache/vhosts/%0
```

2. À votre avis, pourquoi le fichier par défaut `000-default.conf` est-il nommé ainsi ? Gagné : pour être le premier hôte virtuel dans l'ordre de lecture des fichiers de configuration.
3. Imagine-t-on un hébergement mutualisé où chaque hôte virtuel serait ajouté manuellement ?
4. http://httpd.apache.org/docs/2.4/mod/mod_vhost_alias.html

Une requête sur `http://www.example.org/dir/file.html` sera satisfaite avec le fichier `/usr/local/apache/vhosts/www.example.org/dir/file.html`.

Notez la directive `UseCanonicalName` paramétrée à `Off` : elle sert à indiquer à Apache qu'il va devoir construire ses URL *auto-identifiante* — c'est à dire les URL qui font référence au serveur lui-même — à partir des informations fournies par le client et est nécessaire pour utiliser des hôtes virtuels dynamiques.

L'inconvénient de cette technique est que tous les hôtes virtuels partageront la même configuration. Or, on ne configure pas de la même façon l'hôte virtuel pour un Wordpress, un Nextcloud… et encore moins pour un Gitlab !

1.7 Quelques conseils à propos des hôtes virtuels…

— Toujours utiliser une adresse IP (ou un astérisque `*`) dans les directives `<VirtualHost>` et jamais de nom de machine. En effet, un problème de résolution DNS peut alors empêcher Apache de démarrer.
— Toujours définir une directive `ServerName` dans tous les hôtes virtuels et donc ne pas se fier au DNS inverse pour connaître le nom de serveur d'un hôte virtuel.
— Les hôtes virtuels par adresse et par nom sont indépendants. Il faut donc s'arranger dans leur définition pour qu'il n'y ait pas de conflit.
— En production, toujours définir un hôte virtuel par défaut. Si on ne peut pas utiliser `_default_`, on s'assurera qu'un hôte virtuel est celui par défaut en nommant son fichier de configuration de façon à ce qu'il soit lu en premier par Apache. Le nom de ce fichier est généralement préfixé par `000-` car les fichiers sont lus dans l'ordre alphabétique.

2 Nginx

2.1 Hôtes virtuels par adresse IP et / ou port

Ici, point de directive `Listen` dans le contexte `main` (correspondant au contexte global), mais une ou plusieurs directives `listen` dans les contextes `server`, suivie d'une IP et/ou d'un port. On pourra omettre le port utilisé, auquel cas le port 80 sera utilisé. Une IPv6 sera encore une fois mise entre crochets.

```
server {
    listen 80;
    listen 192.168.1.100;
    listen 192.168.1.101:80;
    listen [2001:db8:a::42]:80;

    ...
}
```

Si la directive `listen` n'est pas présente dans un contexte `server`, le port 80 sera utilisé par défaut, sur toutes les interfaces (équivalent à `listen 80;`).

Si une requête arrive sur un couple adresse IP/port non défini dans un hôte virtuel, la connexion ne se fera tout simplement pas : il n'y a pas de serveur principal comme sur Apache.

2.2 Hôtes virtuels par nom

On utilisera la directive `server_name`.

```
server {
    listen 80;
    server_name example.org www.example.org *.example.org www.example.*;

    ...
}
```

Pour définir un hôte virtuel par défaut, au cas où aucun hôte virtuel ne correspond à la requête, on utilisera l'argument optionnel `default_server` dans la directive `listen`, sinon c'est l'ordre d'apparition des configurations qui s'appliquera (comme pour Apache) :

```
server {
    listen 80 default_server;

    ...
}
```

2.3 Hôtes virtuels dynamiques

Il n'y a pas de module équivalent à celui d'Apache, mais on peut aisément se débrouiller avec une regex et une capture :

```
server {
    listen 80;
    server_name   ~^(www\.)?(?<domain>.+)$;
    root       /var/www/$domain/public/;
}
```

Source : http://syshero.org/post/68729802960/nginx-dynamically-configured-mass-virtual-hosting