Apache 2 : sous-domaines (serveurs virtuels)

Les hôtes virtuels du serveur web Apache permettent d'héberger plusieurs domaines sur la même IP. Le champ *Host:* de la requête HTTP détermine l'hôte virtuel qui devra traiter la requête.

Sous-domaines par nom

Pour chaque domaine, on crée un fichier /etc/apache2/sites-available/domaineX.com

en local : sous-domaines du localhost

Exemple : sous-domaine doc.localhost

Accessible localement sur le PC lui-même (localhost) ; Cela permet de tester le serveur. Pour cela,

- déclarer doc.localhost dans le fichier hosts
- créer le fichier/etc/apache2/sites-available/doc.localhost

Éditer le fichier `etc/hosts` du PC :

```
sudo geany /etc/hosts
```

et lui ajouter la ligne :

hosts

```
127.0.0.1    doc.localhost
```

créer le fichier `/etc/apache2/sites-available/doc.localhost` :

```
sudo geany /etc/apache2/sites-available/doc.localhost
```

et lui donner ce contenu :

doc.localhost

```
ServerAdmin administrateur@domaine.fr
UseCanonicalName Off
LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

<VirtualHost *:80>
    ServerName    doc.localhost
    DocumentRoot    /home/<USER>/www/doc
```

```
</VirtualHost>
```

sur le web : sous-domaines d'un domaine

Pour créer doc.mondomaine.com, sous-domaine de mondomaine.com :

doc.mondomaine.com

```
ServerAdmin administrateur@domaine.fr
UseCanonicalName Off
LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

<VirtualHost *:80>
    ServerName    doc.mondomaine.com
    DocumentRoot    /home/<USER>/www/doc
</VirtualHost>
```

<VirtualHost *:80>

```
ServerName  localhost
ServerAlias *.localhost
```

DocumentRoot /home/nicolas/www

```
VirtualDocumentRoot /home/nicolas/www/%-2
```

</VirtualHost>

#<VirtualHost *:80> # ServerName cloud.localhost # DocumentRoot /home/nicolas/www/cloud #</VirtualHost>

Préparation de l'environnement

Voici un exemple d'organisation :

```
mondomaine.tld/
|-- config
|   |-- auth
|   `-- ssl
|-- logs
|   |-- access.log
|   `-- error.log
`-- www
    |-- subdir1
    |-- subdir2
```

```
|-- subdir3
`-- subdir4
```

- config → Contiendra toute la configuration “magique”:
  - auth → liens symboliques vers les sous-domaines (subdir*) nécessitant une authentification.
  - ssl → liens symboliques vers les sous-domaines (subdir*) nécessitant un accès SSL.

2. logs → logs d'accès et logs d'erreur pour le domaine concerné
3. www → racine de l'arborescence web où se trouveront nos pages et scripts éventuels et les sous-domaines (subdir*).

Configuration du domaine mondomaine.com

Pour ce domaine, nous créons les 2 configurations suivantes dans le fichier /etc/apache2/sites-available/mondomaine.com.

Configuration HTTP du domaine mondomaine.com

```
<VirtualHost *:80>
        ServerName mondomaine.com
        ServerAlias *.mondomaine.com
        ServerAlias *.mondomaine.net
        ServerAlias *.mondomaine.fr
        ServerAlias *.mondomaine.eu
        ServerAlias *.mondomaine.info

        ServerAdmin webmaster@mondomaine.com

        DocumentRoot /var/www/mondomaine.com/www/
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/mondomaine.com/www/>
                Options FollowSymLinks
                AllowOverride none
                Order allow,deny
                allow from all
        </Directory>

########## Log Directives
        ErrorLog /var/www/mondomaine.com/logs/error.log
        CustomLog /var/www/mondomaine.com/logs/access.log
vhost_combined
</VirtualHost>
```

Configuration HTTPS du domaine mondomaine.com

```
<VirtualHost *:443>
```

```
        ServerName mondomaine.com
        ServerAlias *.mondomaine.com
        ServerAlias *.mondomaine.net
        ServerAlias *.mondomaine.fr
        ServerAlias *.mondomaine.eu
        ServerAlias *.mondomaine.info

        ServerAdmin webmaster@mondomaine.com

        DocumentRoot /var/www/mondomaine.com/www/
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/mondomaine.com/www/>
                Options FollowSymLinks
                AllowOverride none
                Order allow,deny
                allow from all
        </Directory>

########## Log Directives
        ErrorLog /var/www/mondomaine.com/logs/error.log
        CustomLog /var/www/mondomaine.com/logs/access.log
vhost_combined

########## SSL Directives
        SSLEngine on
        SSLCertificateFile
/var/www/mondomaine.com/config/mondomaine.com.pem
        SSLCertificateKeyFile
/var/www/mondomaine.com/config/mondomaine.com.pem
        SSLCACertificateFile /var/www/mondomaine.com/config/ca.crt
</VirtualHost>
```

Activation du domaine mondomaine.com

```
sudo a2ensite mondomaine.com
```

Redémarrage d'Apache

```
/etc/init.d/apache restart
```

Cas du vhost par défaut

Pour gérer le cas d'une requête à destination d'un domaine qui n'est pas hébergé, un vhost par défaut est défini. Il est défini en premier.

Créer un fichier /etc/apache2/site-available/0000-default en y ajoutant la configuration suivante :

```
NameVirtualHost *:80
NameVirtualHost *:443
<VirtualHost *:80>
        RewriteEngine on
        RewriteRule ^/(.*) http://www.mondomaine.com/$1 [R=301,L]
</VirtualHost>
```

On redirige tout le trafic vers le “vrai” domaine. Par exemple, toute requête vers http://ip_du_serveur, traitée par ce vhost, sera visiteur redirigée vers http://www.mondomaine.com/

On active le vhost :

```
a2ensite 0000-default
```

Règles de comportement

Nous souhaitons héberger le domaine mondomaine.com.

Nous devons rediriger mondomaine.com vers www.mondomaine.com.

Nous voulons pouvoir obtenir, le plus simplement possible, différents sous-domaines. Par exemple:
* blog.domaine.com doit être accessible en HTTP ou en SSL. L'authentification est gérée par l'application elle-même.
* webmail.domaine.com doit être obligatoirement en SSL. L'authentification est gérée par l'application elle-même.
* protected.domaine.com ne sera accessible qu'après authentification, mais l'accès n'est pas forcé en SSL.
* maitredumonde.domaine.com ne sera accessible qu'après authentification, mais l'accès est en plus forcé en SSL.

Comme nous avons également loué le domaine mondomaine.fr, il faut que les 2 pointent sur le même espace d'hébergement.

En ce qui concerne les alias de domaines (différentes extensions tld), il est nécessaire de modifier la configuration du vhost en ajoutant les directives ServerAlias. En effet, les domaines n’apparaissant ni en ServerName, ni en ServerAlias seront traités par le vhost par défaut.

Configuration d'hôtes virtuels sur Apache avec support automatique des sous-domaines, du SSL et de l'authentification

Utilisation des virtualhosts

Les Serveurs Virtuels font fonctionner un ou plusieurs serveurs Web [1] sur une même machine.

Les serveurs virtuels peuvent être :

- “par-IP” → une adresse IP est attribuée à chaque serveur Web
- ou “par-nom” [2] → plusieurs noms de domaine se côtoient sur des mêmes adresses IP.

Il s'agit du même serveur physique, mais c'est transparent pour l'utilisateur.

- Pour un hébergement virtuel par IP, l'adresse IP de la connexion détermine quel serveur virtuel doit répondre. Chaque serveur doit donc avoir une adresse IP différente.
- Pour un hébergement virtuel par nom, le serveur s’appuie sur les informations transmises par le client dans les en-têtes HTTP de ses requêtes.

Préférer l'hébergement virtuel par nom, plus simple :

- il suffit de configurer le serveur DNS pour que chaque domaine pointe sur l'adresse IP voulue
- et de configurer le serveur Apache pour qu'il reconnaisse ces domaines

Serveurs Virtuels par-Nom (Un ou plusieurs sites Web par adresse IP)

Les noms utilisés doivent être définis sur le serveur DNS et être liés à une adresse IP associée au serveur.

www.domain.tld

```
# Toutes les adresses IP doivent répondre aux requêtes sur les serveurs virtuels
NameVirtualHost *:80

# section VirtualHost (même argument que NameVirtualHost)
<VirtualHost *:80>
  ServerName www.domain.tld
  DocumentRoot /homme/www/www.domain.tld
  ServerAlias domain.tld *.domain.tld
    ...
</VirtualHost>
```

Directives obligatoires d'une section VirtualHost :

ServerName : nom du serveur
DocumentRoot : racine du serveur sur le système de fichiers Autres directives (facultatives) :

ServerAlias : autres noms permis pour accéder au même site Web (jokers * et ? autorisés)

Génération automatique des domaines : virtualhosts dynamiques

La génération automatique des domaines est utile si les <VirtualHost> sont nombreux et sensiblement les mêmes, par exemple :

```
NameVirtualHost 111.22.33.44
<VirtualHost 111.22.33.44>
  ServerName www.customer-1.com
  DocumentRoot /www/hosts/www.customer-1.com/docs
  ScriptAlias /cgi-bin/ /www/hosts/www.customer-1.com/cgi-bin
</VirtualHost>

<VirtualHost 111.22.33.44>
  ServerName www.customer-2.com
  DocumentRoot /www/hosts/www.customer-2.com/docs
  ScriptAlias /cgi-bin/ /www/hosts/www.customer-2.com/cgi-bin
</VirtualHost>

<VirtualHost 111.22.33.44>
  ServerName www.customer-N.com
  DocumentRoot /www/hosts/www.customer-N.com/docs
  ScriptAlias /cgi-bin/ /www/hosts/www.customer-N.com/cgi-bin
</VirtualHost>
```

L'idée de base est de remplacer la configuration <VirtualHost> statique par un mécanisme dynamique. Ainsi,

- Le fichier de configuration est plus petit
- Ajouter des serveurs virtuels revient à créer les répertoires appropriés dans le système de fichiers et les entrées dans le DNS sans reconfigurer ni redémarrer Apache.

Un hôte virtuel est défini par :

- son adresse IP
- et le contenu de l'entête Host: de la requête HTTP.

Pour cela, cette information est insérée dynamiquement dans le chemin d'accès du fichier utilisé pour satisfaire la demande. Il faut activer le module mod_vhost_alias par :

```
sudo a2enmod vhost_alias
```

Il faut modifier deux choses pour rendre l'hôte virtuel dynamique :

- UseCanonicalName Off → le nom du serveur est fourni par l'entête Host: de la requête. Si Apache ne trouve pas nom du serveur, alors la valeur configurée avec ServerName est utilisée
- DocumentRoot

Hôtes virtuels dynamiques simples

Cet extrait du fichier httpd.conf met en œuvre l'hôte virtuel décrit ci-dessus, mais d'une façon générique à l'aide mod_vhost_alias. Voir aussi une amélioration au paragraphe 2.4. Serveur virtuel basé sur l'IP plus efficace

```
# get the server name from the Host: header
UseCanonicalName Off

# this log format can be split per-virtual-host based on the first
field
LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

# include the server name in the filenames used to satisfy requests
VirtualDocumentRoot /www/hosts/%0/docs
VirtualScriptAlias /www/hosts/%0/cgi-bin
```

Le nom du serveur qui est inséré dans le nom de fichier est déduit de l'adresse IP de l'hôte virtuel.

Un serveur de pages d'accueil virtuel

C'est une adaptation du système ci-dessus pour créer des pages d'accueil. Des sous-chaînes du nom du serveur sont utilisées dans le nom du fichier de sorte que, par exemple les documents pour www.user.isp.com se trouvent dans /home/user/.

Voir aussi 2.6. Un serveur de pages d'accueil en utilisant le mod_rewrite

```
# all the preliminary stuff is the same as above, then

# include part of the server name in the filenames
VirtualDocumentRoot /www/hosts/%2/docs

# single cgi-bin directory
ScriptAlias /cgi-bin/ /www/std-cgi/
```

Plus d'un serveur virtuel sur le même serveur

Par exemple, une adresse IP pour les pages d'accueil des clients les et un autre pour les commerciaux avec :

```
UseCanonicalName Off

LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon

<Directory /www/commercial>
```

```
  Options FollowSymLinks
  AllowOverride All
</Directory>

<Directory /www/homepages>
  Options FollowSymLinks
  AllowOverride None
</Directory>

<VirtualHost 111.22.33.44>
  ServerName www.commercial.isp.com

  CustomLog logs/access_log.commercial vcommon

  VirtualDocumentRoot /www/commercial/%0/docs
  VirtualScriptAlias /www/commercial/%0/cgi-bin
</VirtualHost>

<VirtualHost 111.22.33.45>
  ServerName www.homepages.isp.com

  CustomLog logs/access_log.homepages vcommon

  VirtualDocumentRoot /www/homepages/%0/docs
  ScriptAlias /cgi-bin/ /www/std-cgi/
</VirtualHost>
```

<note>Si le premier bloc VirtualHost n'inclut pas une directive ServerName, le reverse DNS de l'adresse IP correspondante sera utilisé à la place. Si ce n'est pas le nom du serveur voulu, une entrée bidon (ServerName none.example.com) peut être ajoutée pour contourner ce comportement.</note>

Serveur virtuel basé sur l'IP plus efficace

Amélioration du paragraphe 2.1. Hôtes virtuels dynamiques simples. Pour éviter la recherche DNS pour transformer un nom en une adresse IP, on peut se baser sur les adresses IP elles-mêmes plutôt que sur les noms correspondants.

```
# get the server name from the reverse DNS of the IP address
UseCanonicalName DNS

# include the IP address in the logs so they may be split
LogFormat "%A %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

# include the IP address in the filenames
VirtualDocumentRootIP /www/hosts/%0/docs
VirtualScriptAliasIP /www/hosts/%0/cgi-bin
```

Hôtes virtuels dynamiques simples utilisant mod_rewrite

Cet extrait du fichier httpd.conf fait la même chose que le premier exemple (2.1. Hôtes virtuels dynamiques simples). La première moitié est très semblable à la partie correspondante ci-dessus, mais avec quelques changements pour la compatibilité descendante et pour faire fonctionner correctement la partie mod_rewrite, la seconde moitié configure mod_rewrite pour faire le travail.

Il y a une paire de trucs particulièrement astucieux : par défaut, mod_rewrite est exécuté avant les autres modules de conversion d'URI (mod_alias etc) : s'ils sont utilisés, mod_rewrite doit être configuré pour en tenir compte. De plus, il faut faire un peu de magie pour faire un équivalent per-dynamic-virtual-host du ScriptAlias.

```
# get the server name from the Host: header
UseCanonicalName Off

# splittable logs
LogFormat "%{Host}i %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

<Directory /www/hosts>
  # ExecCGI is needed here because we can't force
  # CGI execution in the way that ScriptAlias does
  Options FollowSymLinks ExecCGI
</Directory>

# now for the hard bit

RewriteEngine On

# a ServerName derived from a Host: header may be any case at all
RewriteMap lowercase int:tolower

## deal with normal documents first:
# allow Alias /icons/ to work - repeat for other aliases
RewriteCond %{REQUEST_URI} !^/icons/
# allow CGIs to work
RewriteCond %{REQUEST_URI} !^/cgi-bin/
# do the magic
RewriteRule ^/(.*)$ /www/hosts/${lowercase:%{SERVER_NAME}}/docs/$1

## and now deal with CGIs - we have to force a MIME type
RewriteCond %{REQUEST_URI} ^/cgi-bin/
RewriteRule ^/(.*)$ /www/hosts/${lowercase:%{SERVER_NAME}}/cgi-bin/$1
[T=application/x-httpd-cgi]

# that's it!
```

Un serveur de pages d'accueil en utilisant le mod_rewrite

Ceci fait la même chose que le second exemple (2.2. Un serveur de pages d'accueil virtuel).

```
RewriteEngine on

RewriteMap lowercase int:tolower

# allow CGIs to work
RewriteCond %{REQUEST_URI} !^/cgi-bin/

# check the hostname is right so that the RewriteRule works
RewriteCond ${lowercase:%{SERVER_NAME}} ^www\.[a-z-]+\.isp\.com$

# concatenate the virtual host name onto the start of the URI
# the [C] means do the next rewrite on the result of this one
RewriteRule ^(.+) ${lowercase:%{SERVER_NAME}}$1 [C]

# now create the real file name
RewriteRule ^www\.([a-z-]+)\.isp\.com/(.*) /home/$1/$2

# define the global CGI directory
ScriptAlias /cgi-bin/ /www/std-cgi/
```

Utilisation d'un fichier séparé de configuration de l'hôte virtuel

Ce dispositif utilise des fonctionnalités plus avancées de mod_rewrite pour lier un hôte virtuel à la racine des documents à partir d'un fichier de configuration séparé. Cela offre plus de souplesse, mais nécessite une configuration plus complexe.

Le fichier vhost.map contient quelque chose comme ceci:

```
www.customer-1.com /www/customers/1
www.customer-2.com /www/customers/2
# ...
www.customer-N.com /www/customers/N
```

Le fichier http.conf contient ceci:

```
RewriteEngine on

RewriteMap lowercase int:tolower

# define the map file
RewriteMap vhost txt:/www/conf/vhost.map

# deal with aliases as above
RewriteCond %{REQUEST_URI} !^/icons/
RewriteCond %{REQUEST_URI} !^/cgi-bin/
```

```
RewriteCond ${lowercase:%{SERVER_NAME}} ^(.+)$
# this does the file-based remap
RewriteCond ${vhost:%1} ^(/.*)$
RewriteRule ^/(.*)$ %1/docs/$1

RewriteCond %{REQUEST_URI} ^/cgi-bin/
RewriteCond ${lowercase:%{SERVER_NAME}} ^(.+)$
RewriteCond ${vhost:%1} ^(/.*)$
RewriteRule ^/(.*)$ %1/cgi-bin/$1
```

Références

- http://julien-pauli.developpez.com/tutoriels/apache/vhosts/
- http://httpd.apache.org/docs/2.2/fr/vhosts/examples.html
- http://archives.steinmetz.fr/tutoriels/gerer-vos-sous-domaines-automatiquement-sous-apache-et-bind.html
- http://wiki.goldzoneweb.info/virtualhost_dynamique
- http://hosting-project.thibaudsowa.com/3-configuration-apache

Exemples

Plusieurs serveurs virtuels par nom sur une seule adresse IP.

Votre serveur ne dispose que d'une seule adresse IP, et de nombreux alias (CNAMES) pointent vers cette adresse dans le DNS.

Pour l'exemple, www.example.com et www.example.org doivent tourner sur cette machine.

<note> La configuration de serveurs virtuels sous Apache ne provoque pas leur apparition magique dans la configuration du DNS.

Il faut que leurs noms soient définis dans le DNS, et qu'ils y soient résolus sur l'adresse IP du serveur, faute de quoi personne ne pourra visiter votre site Web.

Il est possible d'ajouter des entrées dans le fichier hosts pour tests locaux, mais qui ne fonctionneront que sur la machine possédant ces entrées. </note>

du serveur

```
# Apache doit écouter sur le port 80
Listen 80

# Toutes les adresses IP doivent répondre aux requêtes sur les
# serveurs virtuels
NameVirtualHost *:80

<VirtualHost *:80>
DocumentRoot /www/example.com
```

```
ServerName www.example1.com

# Autres directives ici

</VirtualHost>

<VirtualHost *:80>
DocumentRoot /www/example.org
ServerName www.example2.org

# Autres directives ici

</VirtualHost>
```

Les astérisques correspondent à toutes les adresses, si bien que le serveur principal ne répondra jamais à aucune requête.

Comme www.example.com se trouve en premier dans le fichier de configuration, il a la plus grande priorité et peut être vu comme serveur par défaut ou primaire ; ce qui signifie que toute requête reçue ne correspondant à aucune des directives ServerName sera servie par ce premier VirtualHost.

<note> Si vous le souhaitez, vous pouvez remplacer * par l'adresse IP du système. Dans ce cas, l'argument de VirtualHost doit correspondre à l'argument de NameVirtualHost :

```
NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40>
# etc ...
```

En général, il est commode d'utiliser * sur les systèmes dont l'adresse IP n'est pas constante - par exemple, pour des serveurs dont l'adresse IP est attribuée dynamiquement par le FAI, et où le DNS est géré au moyen d'un DNS dynamique quelconque. Comme * signifie n'importe quelle adresse, cette configuration fonctionne sans devoir être modifiée quand l'adresse IP du système est modifiée. </note>

La configuration ci-dessus est en pratique utilisée dans la plupart des cas pour les serveurs virtuels par nom.

En fait, le seul cas où cette configuration ne fonctionne pas est lorsque différents contenus doivent être servis en fonction de l'adresse IP et du port contactés par le client.

Serveurs virtuels par nom sur plus d'une seule adresse IP

<note>Toutes les techniques présentées ici peuvent être étendues à un plus grand nombre d'adresses IP.</note>

Le serveur a deux adresses IP. Sur l'une (172.20.30.40), le serveur “principal” server.domain.com doit répondre, et sur l'autre (172.20.30.50), deux serveurs virtuels (ou plus) répondront.

du serveur

```
Listen 80

# Serveur "principal" sur 172.20.30.40
ServerName server.domain.com
DocumentRoot /www/mainserver

# l'autre adresse
NameVirtualHost 172.20.30.50

<VirtualHost 172.20.30.50>
DocumentRoot /www/example.com
ServerName www.example.com

# D'autres directives ici ...

</VirtualHost>

<VirtualHost 172.20.30.50>
DocumentRoot /www/example.org
ServerName www.example.org

# D'autres directives ici ...

</VirtualHost>
```

Toute requête arrivant sur une autre adresse que 172.20.30.50 sera servie par le serveur principal.

Les requêtes vers 172.20.30.50 avec un nom de serveur inconnu, ou sans en-tête Host:, seront servies par www.example.com.

Servir le même contenu sur des adresses IP différentes (telle qu'une adresse interne et une externe)

La machine serveur dispose de deux adresses IP (192.168.1.1 et 172.20.30.40).

Cette machine est placée à la fois sur le réseau interne (l'Intranet) et le réseau externe (Internet).

Sur Internet, le nom server.example.com pointe vers l'adresse externe (172.20.30.40), mais sur le réseau interne, ce même nom pointe vers l'adresse interne (192.168.1.1).

Le serveur peut être configuré pour répondre de la même manière aux requêtes internes et externes, au moyen d'une seule section VirtualHost.

du serveur

```
NameVirtualHost 192.168.1.1
NameVirtualHost 172.20.30.40

<VirtualHost 192.168.1.1 172.20.30.40>
DocumentRoot /www/server1
ServerName server.example.com
ServerAlias server
</VirtualHost>
```

Ainsi, les requêtes en provenance de chacun des deux réseaux seront servies par le même VirtualHost.

<note>Sur le réseau interne, il est possible d'utiliser le nom raccourci server au lieu du nom complet server.example.com.

Notez également que dans l'exemple précédent, vous pouvez remplacer la liste des adresses IP par des * afin que le serveur réponde de la même manière sur toutes ses adresses.</note>

Servir différents sites sur différents ports

Vous disposez de plusieurs domaines pointant sur la même adresse IP et vous voulez également servir de multiples ports.

Vous y parviendrez en définissant les ports dans la directive “NameVirtualHost”.

Si vous tentez d'utiliser <VirtualHost name:port> sans directive NameVirtualHost name:port, ou tentez d'utiliser la directive Listen, votre configuration ne fonctionnera pas.

du serveur

```
Listen 80
Listen 8080

NameVirtualHost 172.20.30.40:80
NameVirtualHost 172.20.30.40:8080

<VirtualHost 172.20.30.40:80>
ServerName www.example.com
DocumentRoot /www/domain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
ServerName www.example.com
DocumentRoot /www/domain-8080
</VirtualHost>

<VirtualHost 172.20.30.40:80>
ServerName www.example.org
DocumentRoot /www/otherdomain-80
```

```
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
ServerName www.example.org
DocumentRoot /www/otherdomain-8080
</VirtualHost>
```

Hébergement virtuel basé sur IP

Le serveur dispose de deux adresses IP (172.20.30.40 et 172.20.30.50) correspondant respectivement aux noms www.example.com et www.example.org.

du serveur

```
Listen 80

<VirtualHost 172.20.30.40>
DocumentRoot /www/example.com
ServerName www.example1.com
</VirtualHost>

<VirtualHost 172.20.30.50>
DocumentRoot /www/example.org
ServerName www.example2.org
</VirtualHost>
```

Les requêtes provenant d'adresses non spécifiées dans l'une des directives <VirtualHost> (comme pour localhost par exemple) seront dirigées vers le serveur principal, s'il en existe un.

Hébergements virtuels mixtes basés sur les ports et sur les IP

Le serveur dispose de deux adresses IP (172.20.30.40 et 172.20.30.50) correspondant respectivement aux noms www.example.com et www.example.org.

Pour chacun d'eux, nous voulons un hébergement sur les ports 80 et 8080.

du serveur

```
Listen 172.20.30.40:80
Listen 172.20.30.40:8080
Listen 172.20.30.50:80
Listen 172.20.30.50:8080

<VirtualHost 172.20.30.40:80>
DocumentRoot /www/example.com-80
ServerName www.example.com
</VirtualHost>
```

```
<VirtualHost 172.20.30.40:8080>
DocumentRoot /www/example.com-8080
ServerName www.example.com
</VirtualHost>

<VirtualHost 172.20.30.50:80>
DocumentRoot /www/example.org-80
ServerName www.example.org
</VirtualHost>

<VirtualHost 172.20.30.50:8080>
DocumentRoot /www/example.org-8080
ServerName www.example.org
</VirtualHost>
```

Hébergements virtuels mixtes basé sur les noms et sur IP

Pour certaines adresses, des serveurs virtuels seront définis par nom, et pour d'autres, ils seront définis par IP.

du serveur

```
Listen 80

NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40>
DocumentRoot /www/example.com
ServerName www.example.com
</VirtualHost>

<VirtualHost 172.20.30.40>
DocumentRoot /www/example.org
ServerName www.example.org
</VirtualHost>

<VirtualHost 172.20.30.40>
DocumentRoot /www/example.net
ServerName www.example.net
</VirtualHost>

# "par-IP"
<VirtualHost 172.20.30.50>
DocumentRoot /www/example.edu
ServerName www.example.edu
</VirtualHost>

<VirtualHost 172.20.30.60>
```

```
DocumentRoot /www/example.gov
ServerName www.example.gov
</VirtualHost>
```

Utilisation simultanée de Virtual_host et de mod_proxy

L'exemple suivant montre comment une machine peut mandater un serveur virtuel fonctionnant sur le serveur d'une autre machine.

Dans cet exemple, un serveur virtuel de même nom est configuré sur une machine à l'adresse 192.168.111.2.

La directive ProxyPreserveHost On est employée pour permette au nom de domaine d'être préservé lors du transfert, au cas où plusieurs noms de domaines cohabitent sur une même machine.

```
<VirtualHost *:*>
ProxyPreserveHost On
ProxyPass / http://192.168.111.2
ProxyPassReverse / http://192.168.111.2/
ServerName hostname.example.com
</VirtualHost>
```

Utilisation de serveurs virtuels _default_

Serveurs virtuels _default_ pour tous les ports

Exemple de capture de toutes les requêtes émanant d'adresses IP ou de ports non connus, c'est-à-dire, d'un couple adresse/port non traité par aucun autre serveur virtuel.

du serveur

```
<VirtualHost _default_:*>
DocumentRoot /www/default
</VirtualHost>
```

L'utilisation d'un tel serveur virtuel avec un joker pour le port empêche de manière efficace qu'une requête n'atteigne le serveur principal.

Un serveur virtuel par défaut ne servira jamais une requête qui est envoyée vers un couple adresse/port utilisée par un serveur virtuel par nom.

Si la requête contient un en-tête Host: inconnu, ou si celui-ci est absent, elle sera toujours servie par le serveur virtuel primaire par nom (celui correspondant à ce couple adresse/port trouvé en premier dans le fichier de configuration).

Vous pouvez utiliser une directive AliasMatch ou RewriteRule afin de réécrire une requête pour

une unique page d'information (ou pour un script).

Serveurs virtuels _default_ pour des ports différents

La configuration est similaire à l'exemple précédent, mais le serveur écoute sur plusieurs ports et un second serveur virtuel _default_ pour le port 80 est ajouté.

du serveur

```
<VirtualHost _default_:80>
DocumentRoot /www/default80
# ...
</VirtualHost>

<VirtualHost _default_:*>
DocumentRoot /www/default
# ...
</VirtualHost>
```

Le serveur virtuel par défaut défini pour le port 80 (il doit impérativement être placé avant un autre serveur virtuel par défaut traitant tous les ports grâce au joker *) capture toutes les requêtes envoyées sur une adresse IP non spécifiée.

Le serveur principal n'est jamais utilisé pour servir une requête.

Serveurs virtuels _default_ pour un seul port

Nous voulons créer un serveur virtuel par défaut seulement pour le port 80.

du serveur

```
<VirtualHost _default_:80>
DocumentRoot /www/default
...
</VirtualHost>
```

Une requête vers une adresse non spécifiée sur le port 80 sera servie par le serveur virtuel par défaut, et toute autre requête vers une adresse et un port non spécifiés sera servie par le serveur principal.

Migration d'un serveur virtuel par nom en un serveur virtuel par IP

Le serveur virtuel par nom avec le nom de domaine www.example.org (de notre exemple par nom) devrait obtenir sa propre adresse IP.

Pendant la phase de migration, il est possible d'éviter les problèmes avec les noms de serveurs

et autres serveurs mandataires qui mémorisent les vielles adresses IP pour les serveurs virtuels par nom.

La solution est simple, car il suffit d'ajouter la nouvelle adresse IP (172.20.30.50) dans la directive VirtualHost.

du serveur

```
Listen 80
ServerName www.example.com
DocumentRoot /www/example.com

NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40 172.20.30.50>
DocumentRoot /www/example.org
ServerName www.example.org
# ...
</VirtualHost>

<VirtualHost 172.20.30.40>
DocumentRoot /www/example.net
ServerName www.example.net
ServerAlias *.example.net
# ...
</VirtualHost>
```

Le serveur virtuel peut maintenant être joint par la nouvelle adresse (comme un serveur virtuel par IP) et par l'ancienne adresse (comme un serveur virtuel par nom).

Utilisation de la directive ServerPath

Dans le cas où vous disposez de deux serveurs virtuels par nom, le client doit transmettre un en-tête Host: correct pour déterminer le serveur concerné.

Les vieux clients HTTP/1.0 n'envoient pas un tel en-tête et Apache n'a aucun indice pour connaître le serveur virtuel devant être joint (il sert la requête à partir d'un serveur virtuel primaire).

Dans un souci de préserver la compatibilité descendante, il suffit de créer un serveur virtuel primaire chargé de retourner une page contenant des liens dont les URLs auront un préfixe identifiant les serveurs virtuels par nom.

du serveur

```
NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40>
# Serveur virtuel primaire
```

```
DocumentRoot /www/subdomain
RewriteEngine On
RewriteRule ^/.* /www/subdomain/index.html
# ...
</VirtualHost>

<VirtualHost 172.20.30.40>
DocumentRoot /www/subdomain/sub1
ServerName www.sub1.domain.tld
ServerPath /sub1/
RewriteEngine On
RewriteRule ^(/sub1/.*) /www/subdomain$1
# ...
</VirtualHost>

<VirtualHost 172.20.30.40>
DocumentRoot /www/subdomain/sub2
ServerName www.sub2.domain.tld
ServerPath /sub2/
RewriteEngine On
RewriteRule ^(/sub2/.*) /www/subdomain$1
# ...
</VirtualHost>
```

À cause de la directive ServerPath, une requête sur une URL http://www.sub1.domain.tld/sub1/ est toujours servie par le serveur sub1-vhost.

Une requête sur une URL http://www.sub1.domain.tld/ n'est servie par le serveur sub1-vhost que si le client envoie un en-tête Host: correct.

Si aucun en-tête Host: n'est transmis, le serveur primaire sera utilisé.

Notez qu'il y a une singularité : une requête sur http://www.sub2.domain.tld/sub1/ est également servie par le serveur sub1-vhost si le client n'envoie pas d'en-tête Host:.

Les directives RewriteRule sont employées pour s'assurer que le client qui envoie un en-tête Host: correct puisse utiliser d'autres variantes d'URLs, c'est-à-dire avec ou sans préfixe d'URL.

Sous-domaines du site domaine.com (IP=1.2.3.4)

L'ordre a une importance.

Si *.domaine.com est déclaré avant toto.domaine.com, alors toto.domaine.com ne sera jamais atteint car *.domaine.com correspond toujours.

Fichier 1.2.3.4

```
NameVirtualHost 1.2.3.4:80
ServerAdmin webmaster@votre-nom-de-domaine.com
```

```
# virtual host par défaut
#   = celui qui sert quand un domaine
#   qui ne correspond a aucun servername ni alias
#   redirige vers l'ip
#   Le plus simple est d'y mettre :
#       une page d'erreur style ('Non existing VirtualHost')
#       et une redirection automatique sur le site principal
    <VirtualHost 1.2.3.4:80>
        ServerName bogus.domaine.com
        DocumentRoot /home/error
    </VirtualHost>

    <VirtualHost 1.2.3.4:80>
        ServerName tintin.domaine.com
        ServerAlias *.tintin.domaine.com
        DocumentRoot /home/sites/titin.domaine.com
        <Directory /home/sites/titin.domaine.com>
            Options Indexes FollowSymLinks MultiViews
            AllowOverride All
            Order allow,deny
            allow from all
        </Directory>
    </VirtualHost>

    <VirtualHost 1.2.3.4:80>
        ServerName toto.domaine.com
        ServerAlias *.toto.com
        DocumentRoot /home/sites/toto.domaine.com
        <Directory /home/sites/toto.domaine.com>
            Options Indexes FollowSymLinks MultiViews
            AllowOverride All
            Order allow,deny
            allow from all
        </Directory>
    </VirtualHost>

    <VirtualHost 1.2.3.4:80>
        ServerName domaine.com
        ServerAlias *.domaine.com
        DocumentRoot  /home/sites/domaine.com
        <Directory /home/sites/domaine.com>
            Options Indexes FollowSymLinks MultiViews
            AllowOverride All
            Order allow,deny
            allow from all
        </Directory>
    </VirtualHost>
```

Fichier toto.kwikwi.net

```
NameVirtualHost *

<VirtualHost *>
    ServerName toto.kwikwi.net
    DocumentRoot /home/sandra/public_html
</VirtualHost>
```

titi.kwikwi.net

```
NameVirtualHost *

<VirtualHost *>
    ServerName titi.kwikwi.net
    DocumentRoot /home/kerphi/public_html
</VirtualHost>
```

Sous-domaines automatiques,

Fichier default

```
NameVirtualHost *:80

<VirtualHost *:80>
    LogFormat "%V %h %l %u %t \"%r\" %>s %b \"%{Referer}i\"
\"%{User-Agent}i \" \"%{forensic-id}n\" %v" combinedvhost
    CustomLog "/var/log/apache2/access.log" combined
    CustomLog "| /usr/sbin/split-logfile" combinedvhost
    VirtualDocumentRoot /media/Reservoirs/www/%0
    VirtualScriptAlias /media/Reservoirs/www/%0/cgi-bin
</VirtualHost>
```

Faire

```
a2enmod vhost_alias
```

Sous-domaines automatiques

Pour donner deux noms au même répertoire, créer un lien symbolique :

```
ln -s /chemin/vers/www/alias_voulu /chemin/vers/www/répertoire_réel
```

Fichier default

```
<VirtualHost *:80>
    ServerAlias *.thibaudsowa.com
    VirtualDocumentRoot /chemin/vers/www/%0
</VirtualHost>
```

Faire

```
a2enmod vhost_alias
```

Références

Configuration d'hôtes virtuels sur Apache avec support automatique des sous-domaines, du SSL et de l'authentification

1)

comme company1.example.com and company2.example.com

2)

parfois également appelée host-based ou serveur virtuel non-IP