M1 Informatique Réseaux et systèmes
TP9: Apache

Gaétan Richard
gaetan.richard@unicaen.fr

11/2017

Ce TP va vous permettre de configurer un serveur apache *from scratch*. Pour ne pas pourrir votre quota d'espace disque, vous pouvez éventuellement travailler dans le répertoire `/tmp/`. Rappelez-vous néanmoins que ce répertoire est nettoyé lors du redémarrage de la machine.

1 Compilation

1.1 Compilation

Pour commencer, récupérer le source de la version développement du serveur web d'Apache (*trunk*) via le *svn* (il peut être nécessaire de passer par HTTPS).

Lire la documentation d'installation (fichier `INSTALL`), réfléchir, réfléchir, compiler et installer (dans un sous-dossier de `/tmp`) le logiciel.

Modifiez le fichier `httpd.conf` pour avoir une configuration correcte puis démarrez le serveur à l'aide de l'outil adapté (**apachectl**).

Constatez que le serveur tourne effectivement (obtention de *"It works !"*). S'il ne tourne pas, corrigez de manière adéquate sa configuration.

1.2 Première analyse

Comme vous l'avez déjà remarqué, le fichier de configuration est le fichier `httpd.conf`. Il est assez volumineux et correctement commenté. Lorsque vous le modifierez, pensez à mettre des commentaires et à bien choisir l'emplacement de vos modifications afin de maintenir ce fichier un minimum lisible.

En parcourant ce fichier (et les fichiers inclus) et à l'aide de la documentation apache, trouvez les réponses aux questions suivantes. Vérifiez vos réponses en observant le programme en cours d'exécution :

— Combien de programmes sont lancés au démarrage du serveur ?
— Quel est l'utilisateur / groupe du programme principal du serveur ?
— Où sont situés les fichiers de logs et combien y en a-t-il ?
— À quoi correspond l'instruction `Require all granted` trouvée dans le fichier de configuration.

2 les modules

Les modules sont des extensions du serveur. Ces modules peuvent être soit directement dans le programme principal, soit externes. Regardez la liste des modules internes de votre serveur (**httpd -l**). Trouvez les modules disponibles et le moyen de les activer.

Quels sont les avantages et inconvénients de chacune des méthodes ?

2.1 Mod_info, mod_status

Testez les modules *mod_info* et *mod_status*. S'ils ne sont pas compilés, Vous pouvez regarder l'utilisation de **apxs**. Ces modules permettent d'avoir des informations sur le serveur.

2.2 Pages des utilisateurs Unix

Trouvez et Configurez le module adéquat afin que votre serveur serve les pages personnelles des utilisateurs (répertoire `public_html` , accessible depuis l'url `/~login/`. Cela ne devrait pas fonctionner pour d'autres répertoires que le vôtre. Les pages personnelles du département informatique ne fonctionnent pas selon ce mode.

2.3 Un peu de php

Nous voulons maintenant ajouter le support de *php5* sur notre serveur. Pour cela, nous allons tout simplement récupérer le code source de php et le compiler. N'oubliez pas de passez les options correctes au configure !

La compilation prend un peu de temps, vous pouvez continuer tranquillement le TP et revenir à la fin de cette partie dès que le compilation est finie.

Une fois php5 compilé, modifiez le fichier `http.conf` pour que les fichiers portant l'extension php soit pris en charge. Testez et validez le résultat.

3 Les fichiers `.ht*`

Ces fichiers servent de fichiers de configurations par répertoire pour le serveur.

Est-ce que ces fichiers sont en accès libre depuis l'extérieur ? Trouvez où se situe la ligne de configuration concernant ce comportement.

3.1 Modifications du comportement

Essayez maintenant de consulter à l'aide de votre serveur la page `~grichard`. Trouvez l'origine du problème et faites en sorte d'autoriser la modification de cette option.

À votre avis, quel doit être le comportement par défaut d'apache pour le listing des répertoires ?

Trouvez l'option permettant de modifier la page affiché lorsque le document demandé n'est pas présent (*erreur 404*). Indiquez également comment faire pour effectuer une redirection permanente ou temporaire à l'aide de directives. Quelle est la différence entre ces deux redirections ?

3.2 Restrictions d'accès

On souhaite maintenant mettre en place des restriction d'accès par mot de passe à certaines page web. En vous appuyant sur l'aide de **htpasswd**, mettez en place dans votre site web un dossier dont l'accès est protégé par mot de passe.

Comment ces mots de passes sont-ils stockés ? Que peut-on faire en cas de perte du mot de passe ?

4 CMS

Depuis quelques années, on assiste à un développement important des CMS (*Content Management Systems*) qui permettent de créer et de gérer de façon simple et dynamique un site web.

Ces systèmes reposent souvent sur l'utilisation de PHP pour fournir une interface d'administration et d'une base de donnée pour enregistrer le site.

Dans notre cas, nous n'avons pas mis en place de base de donnée, mais nous allons installer un CMS plus réduit qui n'en nécessite pas : **CMSimple**. Installez ce CMS dans un sous-répertoire de votre site et testez le.

5 Tout-en-un

Il est courant d'héberger plusieurs site web sur une même machine voir sur un même serveur. Pour cela, il existe un module permettant de configurer des serveurs virtuels. Compilez ce module et configurez le afin d'avoir deux serveurs :

— Un qui répond lorsque l'on consulte la machine par `machine.etu.info.unicaen.fr` ;

— Un quand on le consulte par l'intermédiaire de `localhost`.

Vous veillerez à bien configurer les fichiers de log et à vérifier le résultat obtenu. Que ce passe-t-il si on contacte le serveur par un autre moyen (en utilisant son adresse ip par exemple) ?

Pour le deuxième serveur, restreignez l'accès de ce serveur à votre seule machine en utilisant le module *authz_host*. Quels sont les avantages et inconvénients des fournisseurs d'authentification `host` et `ip`.

Vous remarquerez qu'il existe un grand nombre de modules permettant de mettre un large panel de restrictions d'accès. Jetez un coup d'œil à la documentation de ces modules, particulièrement celle de `mod_access_compat`.

6 Et pour finir, . . .ou pas

6.1 Un peu de réécriture

Dans de nombreux cas, il est souhaitable de pouvoir réécrire les adresses *à la volée* et de façon transparente pour l'utilisateur. Pour cela, il existe le module *mod_rewrite*. S'il n'est pas encore présent dans votre serveur, installez ce module.

Dans un premier temps, utilisez votre `.htaccess` pour effectuer localement un alias à l'aide de la réécriture. Faites ensuite une réécriture plus complexe qui renvoie les pages de la forme `~`*`login`*`/test/x/y.html` sur la page

`~`*`login`*`/toto.html?val1=x&val2=y`.

Au niveau du serveur, faites une règle qui permet d'accéder à votre espace web par l'intermédiaire d'adresses de la forme `http://`*`machine`*`/~toto`.

À l'aide de réécritures conditionnelles, mettez en place une page spéciale pour les utilisateurs de **links**.