MISE EN PLACE ET CONFIGURATION D'UN SERVEUR HTTP

Objectifs :

Mettre en place un serveur web et compléter la maquette mise en place lors du TP01. Comprendre le fonctionnement du serveur Apache, ses fichiers de configuration, la gestion des sites et des modules du serveur.

Etapes facultatives si vous partez directement d'un template Debian-11 ou 12 :

- **Récupération d'un ISO Debian**

  Un ISO de Debian 12 est accessible dans le dossier __RESSOURCES du cluster ESXI.

- **Création d'une machine virtuelle sous vSphere**

  Nom du poste : XX_ServeurLAMP01 (XX : vos initiales)
  Mémoire vive allouée : 512 Mo
  Création disque dur virtuel : Thin Provision
  4,00 Go

  Configuration :
  Réseau : Carte 1 \ Internet Accessible
  Stockage : Lecteur CD/DVD : choisir l'ISO Debian 12
  Démarrage de la machine

- **Installation de Debian**

  Installez la distribution sur votre machine virtuelle.
  Configuration via DHCP
  Nom de la machine : www
  Utilisez le disque entier, tout dans une seule partition.
  N'installez **aucun service**, hormis SSH

- **Installation de quelques paquets utiles :**

```
apt update
apt install net-tools
apt install open-vm-tools
apt install host
```

- **Installation de APACHE**

```
apt update
« apt upgrade » à éviter si réseau peu rapide
apt install apache2
apt install ssl-cert
```

Liste des modules activés par l'installation, version de HTTP Apache 2 :

```
update-alternatives: utilisation de « /usr/bin/prename » pour fournir « /usr/bin
/rename » (rename) en mode automatique
Paramétrage de apache2.2-common (2.2.22-13+deb7u3) ...
Enabling site default.
Enabling module alias.
Enabling module autoindex.
Enabling module dir.
Enabling module env.
Enabling module mime.
Enabling module negotiation.
Enabling module setenvif.
Enabling module status.
Enabling module auth_basic.
Enabling module deflate.
Enabling module authz_default.
Enabling module authz_user.
Enabling module authz_groupfile.
Enabling module authn_file.
Enabling module authz_host.
Enabling module reqtimeout.
Paramétrage de apache2-mpm-worker (2.2.22-13+deb7u3) ...
[ ok ] Starting web server: apache2.
Paramétrage de apache2 (2.2.22-13+deb7u3) ...
Paramétrage de libswitch-perl (2.16-2) ...
root@www:~# _
```

- **Configuration IP statique**

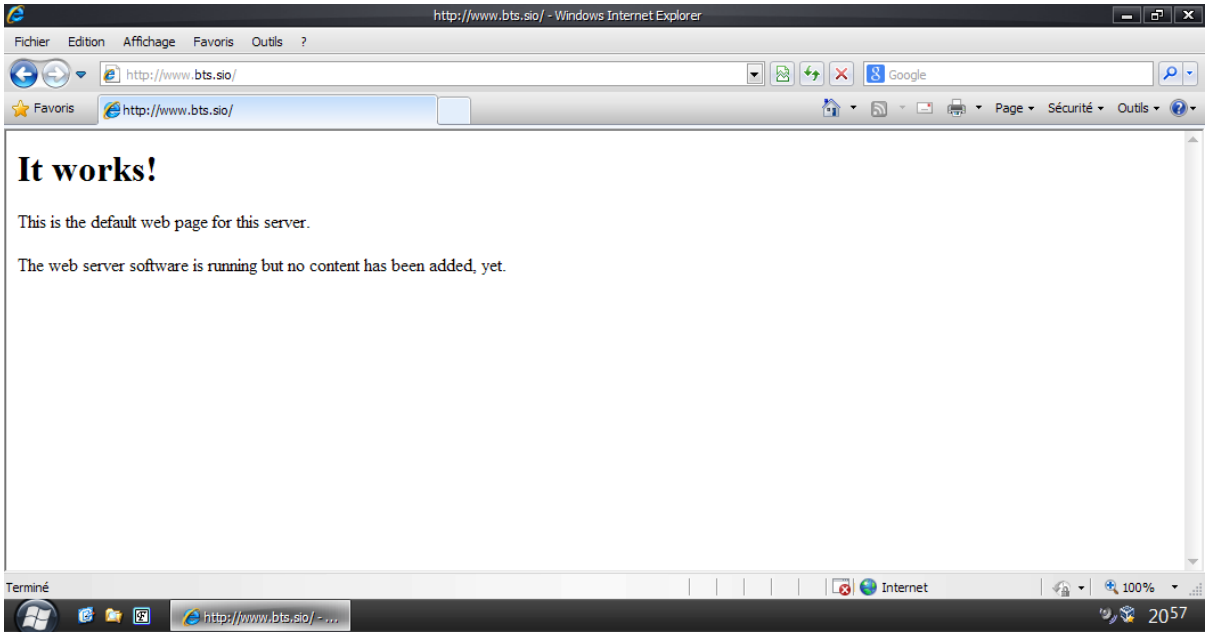
Configurez maintenant votre système en adressage IP statique
A adapter en fonction de votre VPL et du VLAN utilisé (exemple) :
IP : 192.168.xx.5/24
DNS1 : 192.168.xx.1
DNS2 : 192.168.xx.2
Domaine bts.sio

- **Vérification fonctionnement serveur WEB**

A l'aide d'un client (XP, 7, 10 ou Linux) faisant partie du domaine « bts.sio. », tester l'accès à la page web renvoyée par l'installation d'apache.

Soit en tapant l'IP du serveur WEB http://192.168.xx.5 soit, si tous vos serveurs sont bien configurés, directement à l'adresse http://www.bts.sio

Vous devriez accéder à ceci :



- **Configuration du serveur**

Après chaque modification de la configuration du serveur, et afin de tester les différentes fonctionnalités de ce dernier, il sera nécessaire de redémarrer le serveur Apache. Vous pouvez utiliser la commande :

```
systemctl restart apache2
```

Pour informations, voici les paramètres possibles pour la commande :

```
systemctl [start/stop/restart/reload/force-reload/status] apache2
```

Les commandes classiques (à utiliser pour une installation sur une distribution autre que Debian) d'arrêt et de redémarrage du processus apache2 sont les suivantes :

```
apache2ctl -k start
apache2ctl -k stop
apache2ctl -k graceful-stop
apache2ctl -k restart
apache2ctl -k graceful
```

Pour toute modification des fichiers de configuration, n'omettez pas de faire une sauvegarde du fichier au préalable afin de revenir à une configuration fonctionnelle en cas de besoin.

On utilisera la commande `apache2ctl -t` pour vérifier la syntaxe des fichiers de configuration.

La commande `apache2ctl -l` quant à elle permet de vérifier les modules qui ont été compilés avec le serveur. Apache étant un serveur modulaire, seules les fonctionnalités les plus courantes sont incluses dans le serveur de base. Les fonctionnalités étendues sont fournies à l'aide de modules qui peuvent être chargés dans Apache.

Le fichier de configuration principal d'Apache est `/etc/apache2/apache2.conf` ; la multitude de commentaires le rend peu lisible mais heureusement compréhensible.

Analysez votre fichier de configuration et essayez de comprendre l'ensemble des directives présentes.

Répondez ensuite aux questions suivantes :

1. Vérifiez si le numéro du processus linux du démon *apache2* contenu dans le fichier `apache.pid` correspond bien au premier processus apache2.
2. Combien y'a t'il de processus apache2 actuellement fonctionnel ?
3. Que se passe-t-il sur les numéros des processus apache2 après l'exécution d'une commande de redémarrage ? Vous vérifierez notamment si les commandes `apache2ctl -k restart` et `systemctl restart apache2` ont le même effet sur le serveur apache httpd.
4. Vérifiez dans le fichier le nom d'utilisateur et du groupe d'apache.
5. Que contiennent les répertoires
   - `/etc/apache2/mods-available`
   - `/etc/apache2/mods-enabled`
   - `/etc/apache2/sites-available`
   - `/etc/apache2/sites-enabled` ?

Nous allons maintenant utiliser la directive *DirectoryIndex* pour spécifier quel fichier doit être lu si aucun n'est défini par la requête http. Si le module `dir` n'a pas été activé au début, activez le module `dir` **à l'aide d'une des 3 méthodes suivantes :** la 1 étant la moins souhaitable, la 3 étant la plus souhaitable…

1. copiez les fichiers `dir.conf` et `dir.load` situés dans `/etc/apache2/mods-available/` dans `/etc/apache2/mods-enabled/`
2. faire un lien symbolique des `dir.conf` et `dir.load` situés dans `/etc/apache2/mods-available/` vers `/etc/apache2/mods-enabled/`
3. utiliser la commande apache2 « `a2enmod` » pour activer le module (la commande `a2dismod` permet de désactiver le module) : `a2enmod dir`

Modifier la ligne suivante dans votre fichier `dir.conf` puis redémarrer le serveur http :

```
DirectoryIndex index.htm index.html index.php
```

Consultez l'adresse http://www.bts.sio depuis votre client, puisque aucune page `index.htm` n'existe dans `/var/www/html/`, c'est le fichier `index.html` qui est lu.

Copier la page `index.html` contenue dans `/var/www/html/` vers `/var/www/html/index.htm`, modifier ensuite chaque page afin de les différencier.

Sans redémarrer le serveur, rechargez votre page web, effectuez un rafraîchissement de la page si vous ne remarquez pas de différence.

De manière générale, si on veut qu'une modification effectuée dans un des fichiers de `/etc/apache2/` soit pris en compte, il faut redémarrer le serveur web alors que si on fait une modification dans l'espace web `/var/www/html/`, il est inutile de redémarrer le serveur web pour qu'elle soit prise en compte.

Nous allons maintenant créer un espace web privé à notre utilisateur *user* (à créer s'il n'est pas existant). Cet espace sera accessible via l'url http://www.bts.sio/~user/

Commencez par créer un répertoire `public_html` dans le `homedir` de *user* (attention de bien faire cela depuis le compte *user* et non le compte *root*).

Pour passer sous le compte *user*, utilisez la commande suivante :

```
su user
```

Pour aller dans le `homedir` de user :

```
cd
```

Pour créer le répertoire :

```
mkdir public_html
```

Pour revenir au compte *root* :

```
exit
```

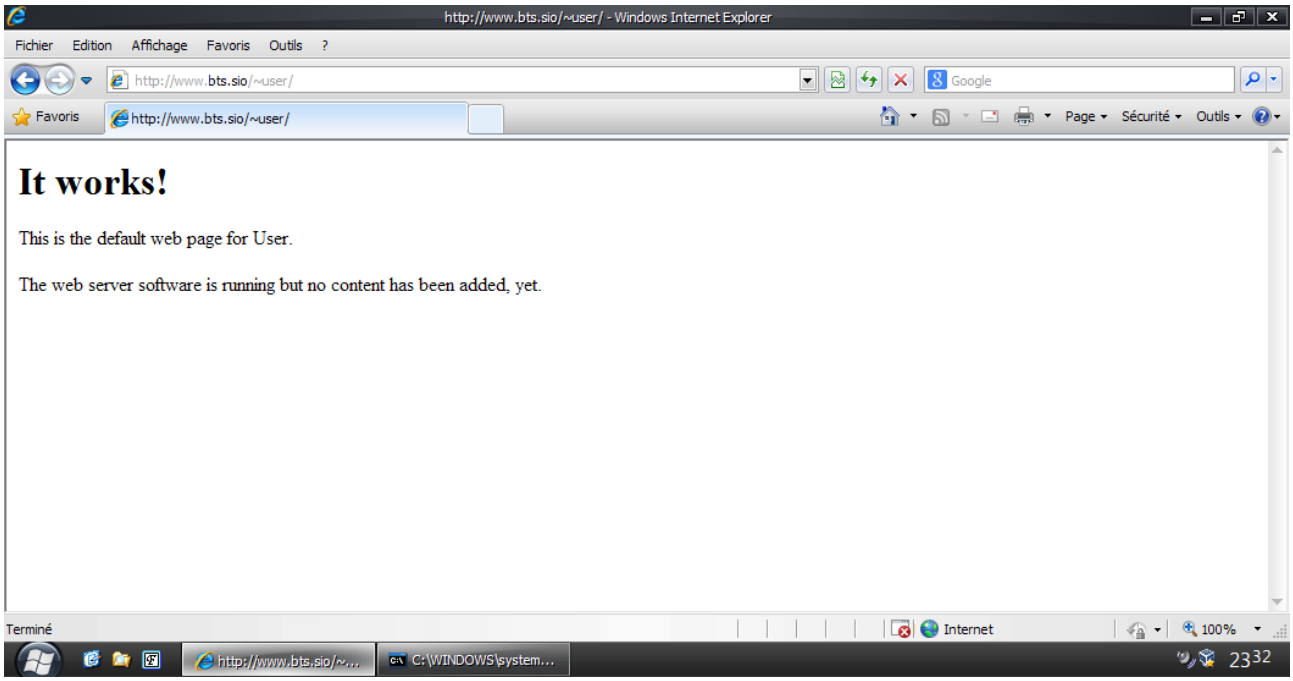
Revenez sous le compte *root*. Il vous reste à activer le module `userdir` (de la même manière que pour le module `dir`) et enfin de redémarrer le serveur http. Le paramétrage se fait dans le fichier `userdir.conf`

```
UserDir public_html
```

Copier la page `index.htm` contenue dans `/var/www/html/` vers `/home/user/public_html/index.htm`, modifier ensuite cette dernière afin de la différencier.

Affichez la page suivante pour vérifier si cela fonctionne :

http://www.bts.sio/~user/



- **Le protocole http**

Cet exercice a pour but de vous faire tester simplement les requêtes et réponses du protocole http. Exécutez depuis un client Windows ou Linux la commande suivante pour vous connecter à votre serveur http :

```
telnet www.bts.sio 80
```

Nous allons envoyer notre première requête http. Envoyez la commande suivante et vérifiez si vous obtenez bien la page web correspondante :

```
GET
```

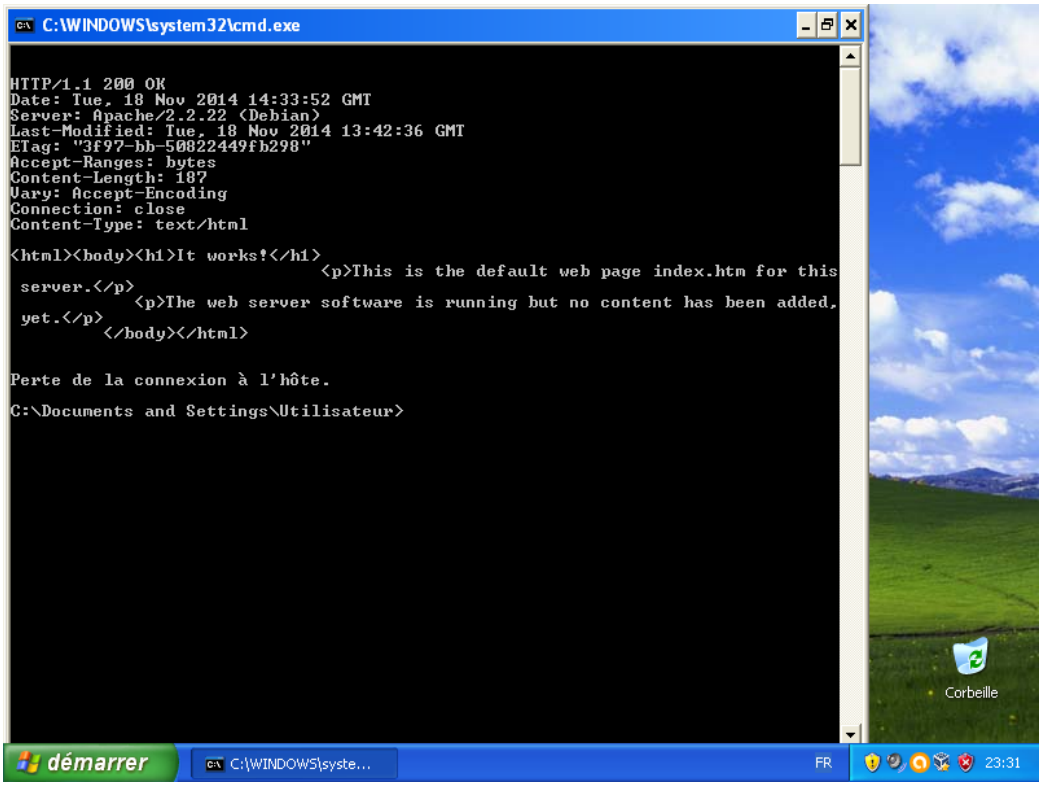
Comme la requête n'est pas valide nous devrions avoir une erreur. Mais le serveur nous retourne normalement la page web sans rien d'autre. Ecrivons maintenant une requête http valide la plus simple possible. Pour information, une requête doit avoir la forme suivante (les informations entre [] sont facultatives) :

```
<Méthode> <URI> HTTP/<Version>
[<Champ d'entête> : <Valeur>]
[<tab><Suite Valeur si >1024>]
Ligne_vide (CRLF)
```

La requête la plus simple est donc la suivante *(n'oubliez pas la ligne vide)* :

ligne 1 `GET / HTTP/1.0`
ligne 2

Vérifiez que vous obtenez bien quelque chose comme cela :

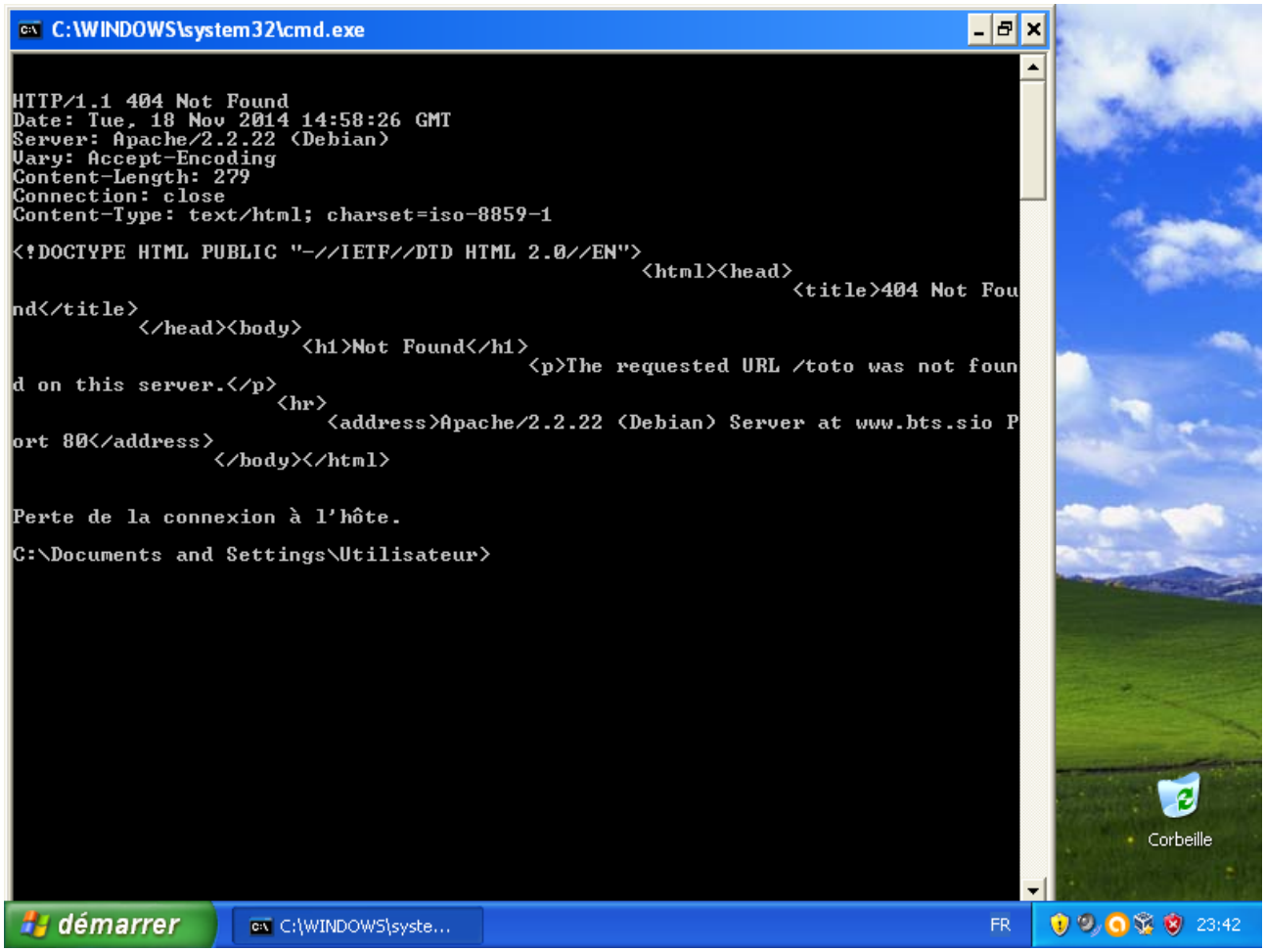


Le but est maintenant de tester les différents codes d'erreurs utilisés dans le protocole HTTP.

Commençons par l'erreur 404 : *page not found*. Ce code est retourné quand la page demandée n'existe pas. Pour vérifier cela, envoyer la requête suivante en vérifiant que vous n'avez pas le fichier *toto* à la racine de votre serveur :

```
GET /toto HTTP/1.0
```

Normalement vous devriez avoir le résultat suivant :



Faites la même requête avec votre navigateur web :

http://www.bts.sio/toto

C'est maintenant à vous d'écrire les requêtes pour observer les codes de statuts suivants. Attention, il ne suffit pas toujours de trouver une requête spécifique mais il faut que vous trouviez une requête pour une configuration serveur donnée. Vous indiquerez donc la configuration apache mise en place si celle-ci joue un rôle particulier.

- 200 : OK
- 403 : Forbidden
- 404 : Not Found
- 400 : Bad Request