SAE S2.03

Installation de service réseaux

Partie 1 : Création du serveur web en local

Table des matières

Introduction 4
Étape 1 : Déploiement du serveur web sur le port par défaut (Port 80) 5
1. Vérification des conflits et lancement du serveur Apache 5
2. Analyse des fichiers de configuration (ports.conf et 000-default.conf) 6
3. Déploiement de votre site personnel 7
4. Test d'affichage du site dans le navigateur 8
Étape 2 : Analyse du trafic réseau (Trames HTTP) 9
1. Lancement de Wireshark 9
2. Génération et filtrage du trafic 10
3. Analyse et commentaires des trames HTTP 11
Étape 3 : Modification de la configuration (Passage au port 8080) 12
1. Modification des fichiers de configuration d'Apache 12
A-Modifier le fichier des ports d'écoute 12
B-Modifier le fichier du site par défaut 12
2. Redémarrage du serveur et test 14
3. Analyse des trames HTTP modifiées 15
Étape 4 : Personnalisation de la configuration (Retour au port 80) 16
1. Rétablissement de l'écoute sur le port 80 16
2. Création d'un fichier de configuration personnalisé 16
3. Activation du nouveau site et désactivation de l'ancien 17
4. Redémarrage et vérification finale 17
Étape 5 : Intégration du principe de DNS 18
1. Modification du fichier "hosts" (l'annuaire local) 18
2. Configuration d'Apache (ServerName) 20
3. Test final dans le navigateur 21
Étape 6 : Configuration d'un serveur Nginx 22
1. Le piège du conflit de ports 22
2. Installation de Nginx 22
3. Modification du port d'écoute de Nginx 22
4. Démarrage et test de cohabitation 23

Étape 7 : Requêtes HTTP manuelles avec Telnet 24
1. Connexion au serveur web via Telnet 24
2. Envoi de la requête HTTP manuelle 24
3. Analyse de la réponse du serveur 25
Étape 8 : Serveur web Apache sous Windows 26
1. Téléchargement des outils et lancement d'Apache 26
2. Déploiement de votre site personnel 27
3. Test d'affichage du site 28
Étape 9 : Analyse des méthodes GET et POST 29
1. Préparation : Création d'une page avec un formulaire 29
2. Capture et observation de la méthode GET 30
3. Capture et observation de la méthode POST 31
Conclusion 32

Introduction

Bienvenue dans ce manuel d'utilisation dédié à la mise en place et à la configuration d'un serveur web local.

Ce document a été spécialement conçu pour vous guider pas à pas, de la préparation de votre environnement de travail jusqu'à l'analyse avancée du trafic réseau. Il s'adresse tout particulièrement aux personnes novices n'ayant aucune expérience préalable dans l'administration système ou la configuration de serveurs. Chaque étape y est minutieusement détaillée, expliquée et illustrée par des captures d'écran pour vous garantir une progression fluide et sans blocage.

Objectifs de ce manuel : L'objectif principal de ce guide est de démystifier le fonctionnement d'un serveur web. Tout au long de ces pages, vous allez apprendre à transformer une simple machine en un serveur capable d'héberger et d'afficher un site internet.

À l'issue de ce tutoriel, vous serez capable de :

- **Préparer un environnement de travail** en utilisant une machine virtuelle.
- **Déployer un site web** localement à l'aide du serveur web Apache.
- **Comprendre les échanges de données** entre un navigateur (le client) et votre serveur en capturant et analysant des trames HTTP avec le logiciel Wireshark.
- **Manipuler la configuration** de votre serveur web (comme modifier le port d'écoute ou personnaliser les fichiers de configuration).

Prenez le temps de lire attentivement chaque instruction, n'hésitez pas à vous référer aux images, et bonne configuration !

Étape 1 : Déploiement du serveur web sur le port par défaut (Port 80)

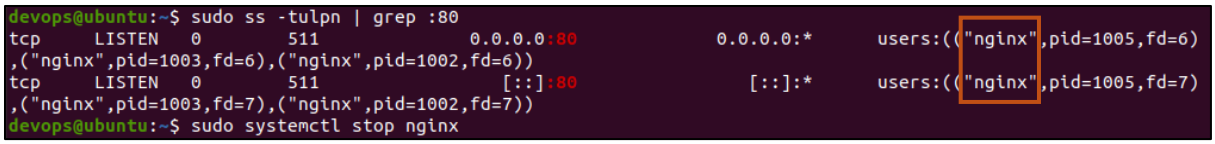
1. Vérification des conflits et lancement du serveur Apache

Le logiciel serveur web le plus utilisé sous Linux s'appelle Apache (ou `apache2`). Cependant, avant de le lancer, une bonne pratique d'administrateur réseau consiste à vérifier qu'aucun autre serveur (comme `nginx` ou un processus fantôme) ne monopolise déjà le port 80. S'il y a un conflit, Apache refusera de démarrer.

1. Ouvrez le terminal de votre machine virtuelle.
2. Tapez la commande suivante pour vérifier si un service écoute déjà sur le port 80 :

```
sudo ss -tulpn | grep :80
```

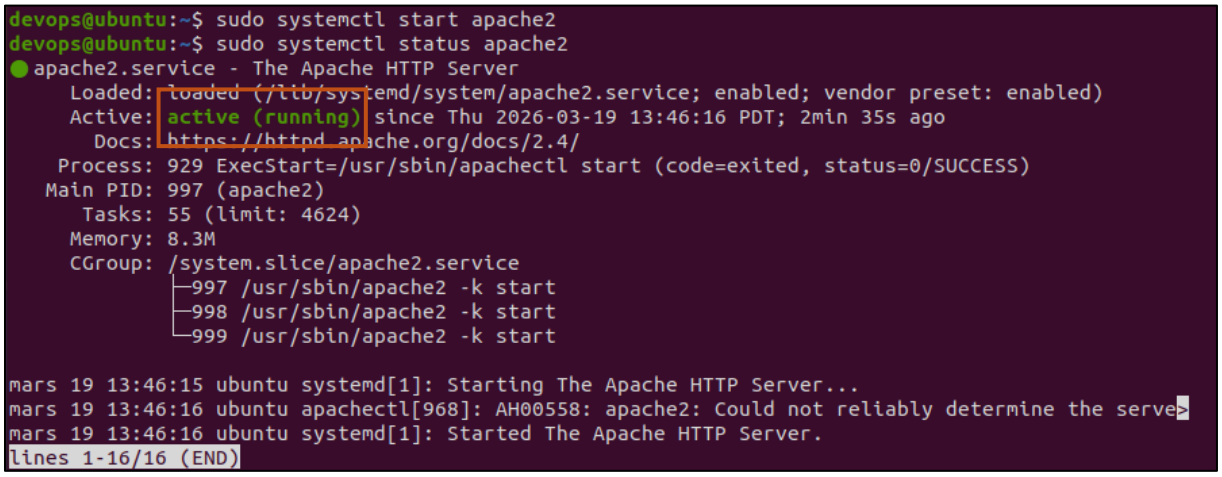
(Remarque : La commande `sudo` vous donne les droits d'administrateur. Il vous sera demandé de taper le mot de passe de l'utilisateur).



- **Cas n°1 : La commande ne renvoie rien.** Parfait, le port 80 est libre !
- **Cas n°2 : Un service (ex: `nginx`) apparaît dans le résultat.** Il faut l'arrêter pour libérer la place, via la commande sudo systemctl stop nginx (remplacez "`nginx`" par le nom du service affiché).

3. Maintenant que la voie est libre, démarrez Apache et vérifiez son état :

```
sudo systemctl start apache2
sudo systemctl status apache2
```



Si le serveur fonctionne correctement, vous devriez voir un point vert et la mention "`active (running)`". Pour quitter cet affichage, appuyez simplement sur la touche q de votre clavier.

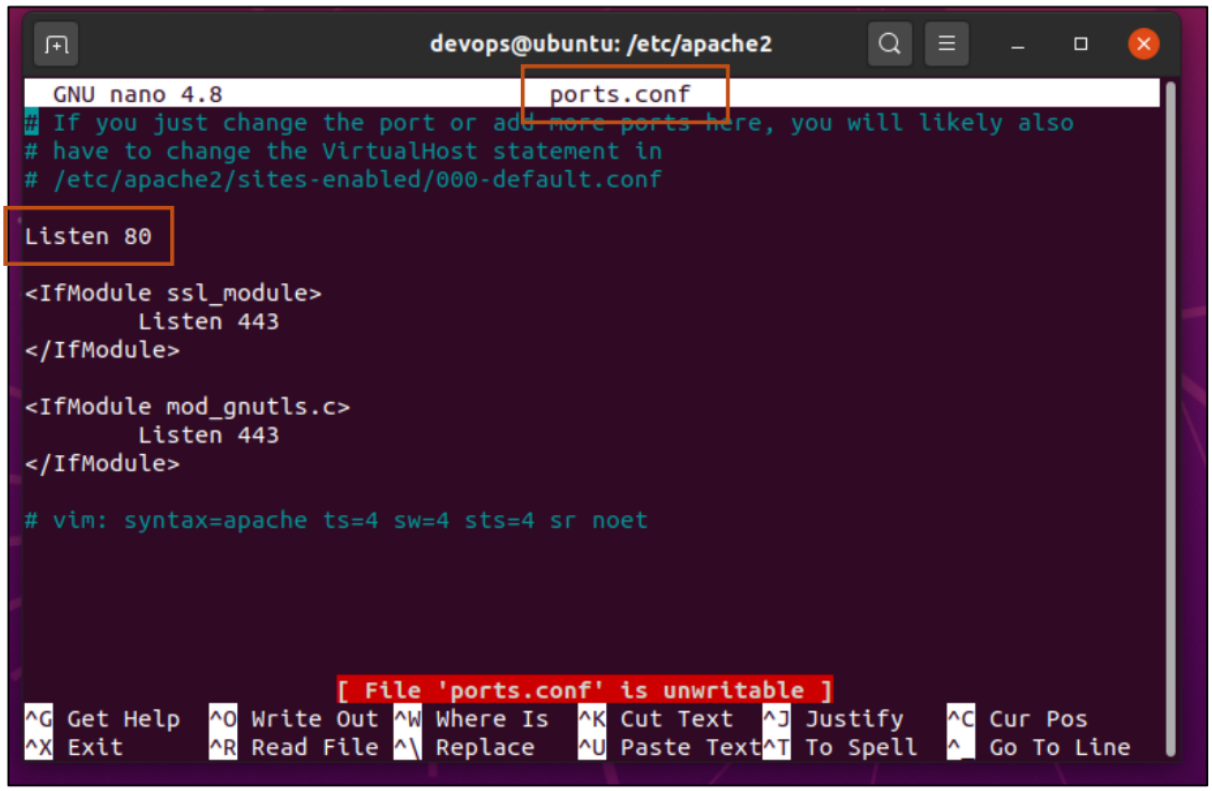
2. Analyse des fichiers de configuration (ports.conf et 000-default.conf)

Pour qu'Apache affiche votre site, il a besoin de deux informations cruciales : savoir sur quel port "écouter" globalement, et savoir quel dossier correspond à ce port. Cela se passe dans deux fichiers distincts.

A. **L'écoute globale** : Le fichier `ports.conf` Par défaut, le trafic web non sécurisé (HTTP) transite par le port 80. Nous allons vérifier qu'Apache est bien configuré pour écouter ce port. Tapez la commande suivante :

```
cat /etc/apache2/ports.conf
```

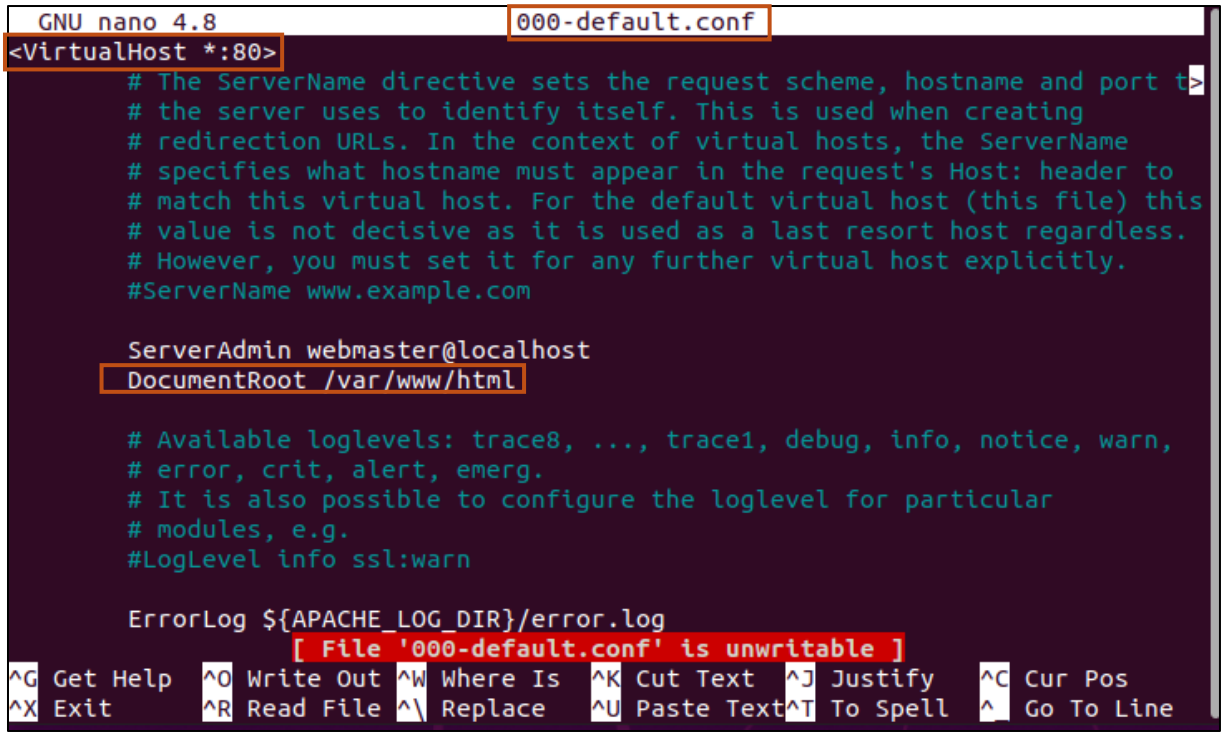
(La commande cat permet de lire le contenu d'un fichier directement dans le terminal).



Dans le texte affiché, repérez la ligne `Listen 80`. C'est la directive principale qui ordonne au serveur d'ouvrir ses "oreilles" sur le port 80.

B. **L'hôte virtuel** : Le fichier `000-default.conf` Maintenant qu'Apache écoute sur le port 80, il faut lui dire quoi faire lorsqu'une requête arrive sur ce port. C'est le rôle du fichier de l'hôte virtuel par défaut. Regardons son contenu :

```
cat /etc/apache2/sites-available/000-default.conf
```



Dans ce fichier, repérez ces deux lignes très importantes :

- `<VirtualHost *:80>` : Cela indique à Apache d'appliquer la configuration suivante pour les requêtes entrant sur le port 80.
- `DocumentRoot /var/www/html` : C'est le chemin du dossier racine où le serveur va chercher les fichiers de votre site web à afficher.

3. Déploiement de votre site personnel

Maintenant que nous savons qu'Apache va chercher les pages web dans le dossier `/var/www/html`, nous devons y placer les fichiers de votre site personnel (celui réalisé lors de votre TP de Web du Semestre 1).

Par défaut, le dossier /var/www/html appartient à l'administrateur. Nous allons copier les fichiers de votre site web vers ce dossier. Si votre site se trouve par exemple dans un dossier `MonSiteWeb` sur votre bureau, utilisez la commande suivante pour copier tout son contenu (remplacez le chemin par le vôtre) :

```
sudo cp -r /chemin/vers/votre/dossier/MonSiteWeb/* /var/www/html/
```

Vérifions que les fichiers sont bien au bon endroit en listant le contenu du répertoire cible :

```
ls /var/www/html/
```



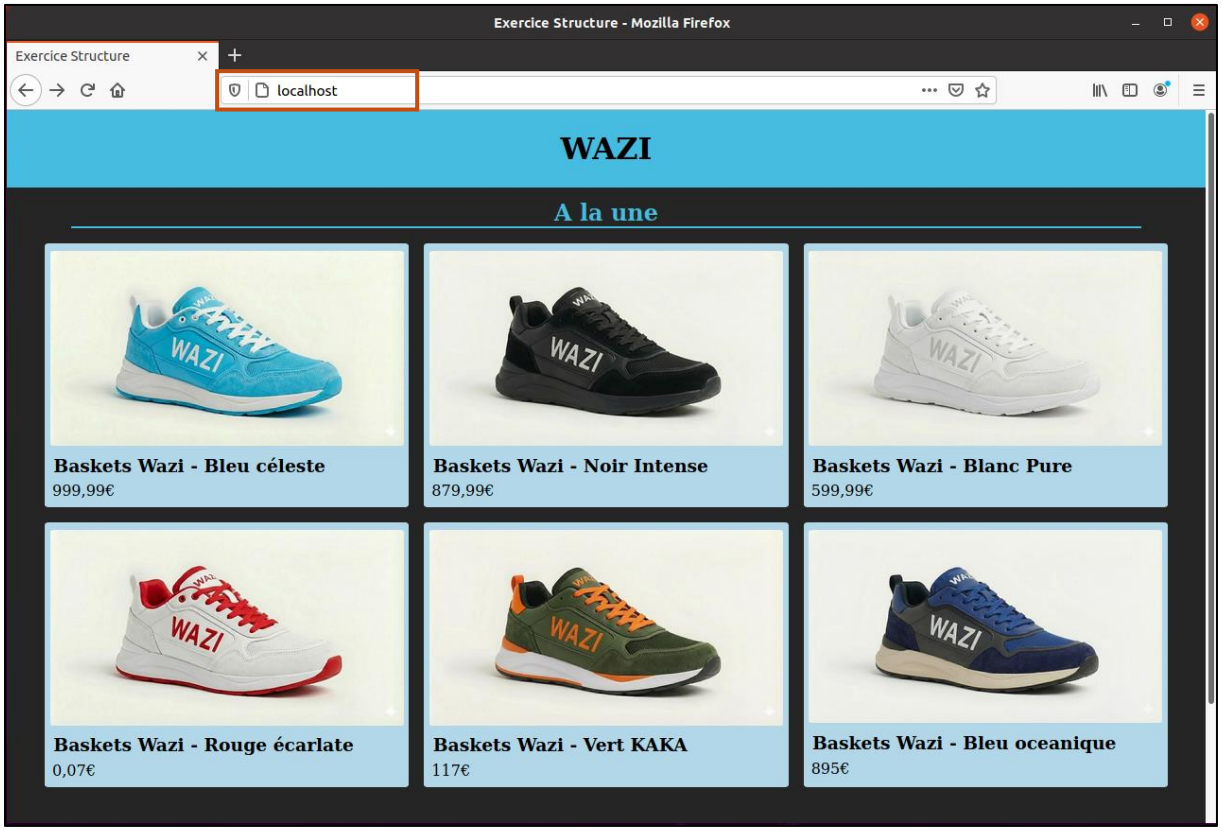
Vous devriez voir apparaître votre fichier `index.html` (la page d'accueil de votre site) ainsi que vos autres fichiers (CSS, images, etc.).

4. Test d'affichage du site dans le navigateur

Tout est prêt ! Il ne reste plus qu'à vérifier que le serveur fait bien son travail.

1. Ouvrez le navigateur web intégré à votre machine virtuelle (par exemple Firefox).
2. Dans la barre d'adresse, tapez simplement : `http://localhost` (ou `http://127.0.0.1` qui est l'adresse IP locale de votre machine).
3. Appuyez sur Entrée.

Votre site web personnel doit s'afficher ! Apache a reçu votre demande sur le port 80, a vérifié qu'aucun autre serveur ne bloquait le passage, a cherché le fichier `index.html` dans le dossier par défaut, et l'a renvoyé à votre navigateur.



Étape 2 : Analyse du trafic réseau (Trames HTTP)

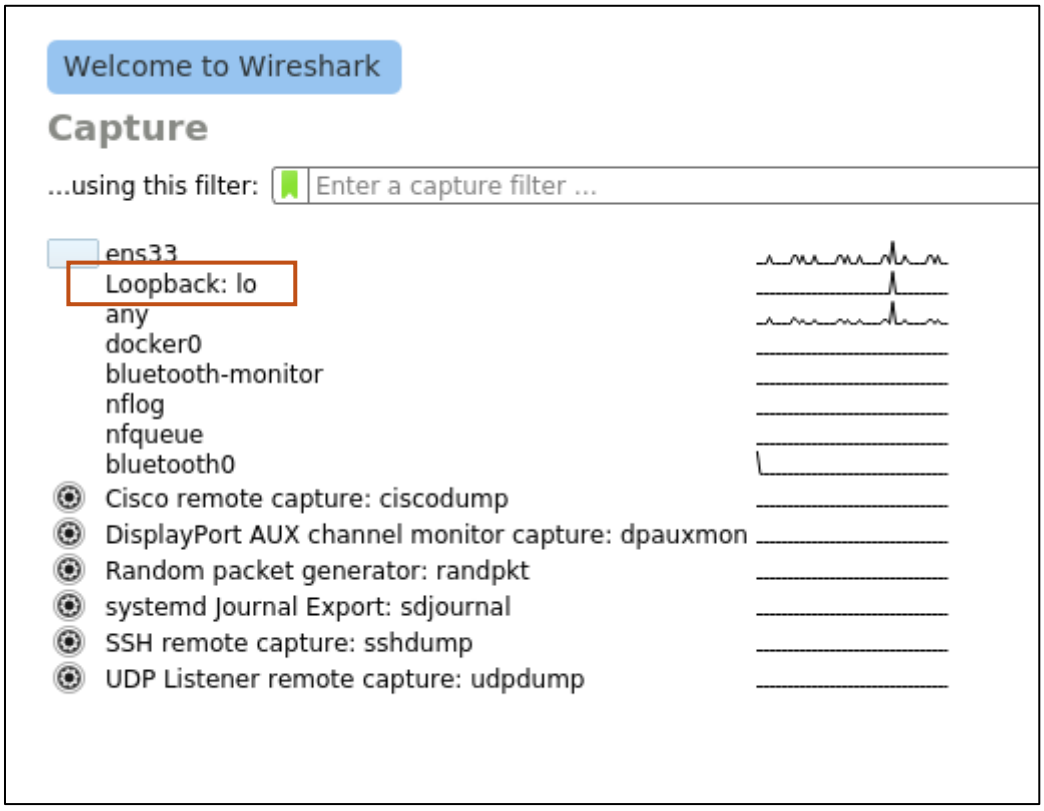
Lorsque vous tapez l'adresse d'un site web, des messages invisibles (appelés "trames" ou "paquets") sont échangés sur le réseau. Nous allons utiliser un outil d'analyse réseau appelé Wireshark pour capturer et lire ces messages.

1. Lancement de Wireshark

1. Retournez dans le terminal de votre machine virtuelle.
2. Lancez le logiciel Wireshark avec les droits d'administrateur en tapant la commande suivante :

```
sudo wireshark
```

3. L'interface graphique de Wireshark va s'ouvrir. Le logiciel vous demande d'abord sur quelle "carte réseau" (interface) il doit écouter.
4. Puisque notre client (le navigateur) et notre serveur (Apache) sont sur la *même* machine, les messages ne sortent pas sur le vrai réseau physique. Ils passent par une interface virtuelle locale. Double-cliquez sur l'interface nommée `Loopback` (souvent abrégée en `lo`). La capture commence immédiatement !



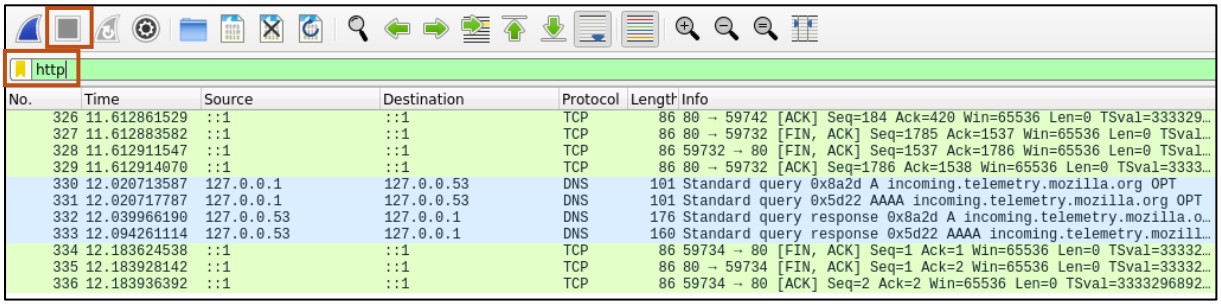
2. Génération et filtrage du trafic

Pour l'instant, il se peut que votre écran se remplisse de lignes de toutes les couleurs (c'est le "bruit de fond" de votre machine). Nous allons générer notre propre trafic web et isoler ce qui nous intéresse.

1. Laissez Wireshark tourner en arrière-plan et retournez sur votre navigateur web où votre site (`http://localhost`) est affiché.
2. **Actualisez la page** (touche F5 ou le bouton de rafraîchissement). Cela va forcer le navigateur à redemander la page au serveur.
3. Retournez vite sur Wireshark et cliquez sur le **carré rouge** en haut à gauche pour arrêter la capture.
4. Pour y voir plus clair, nous allons utiliser un filtre. Dans la barre de recherche verte en haut de Wireshark (où il est écrit "Apply a display filter"), tapez simplement en minuscules :

   `http`

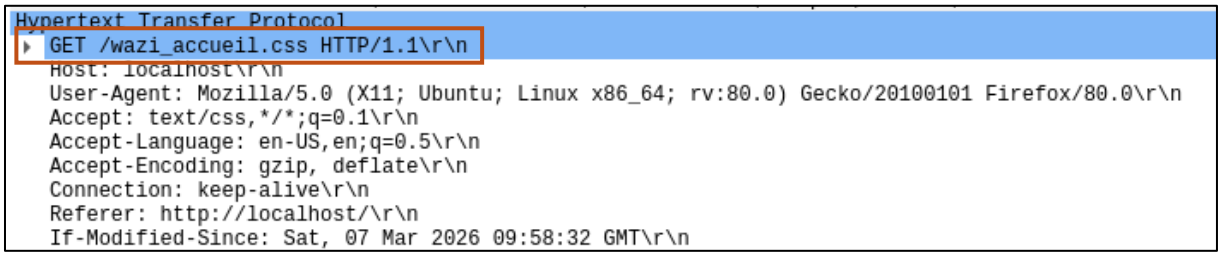
5. Appuyez sur Entrée. Wireshark ne vous affiche désormais plus que les trames liées au web !



3. Analyse et commentaires des trames HTTP

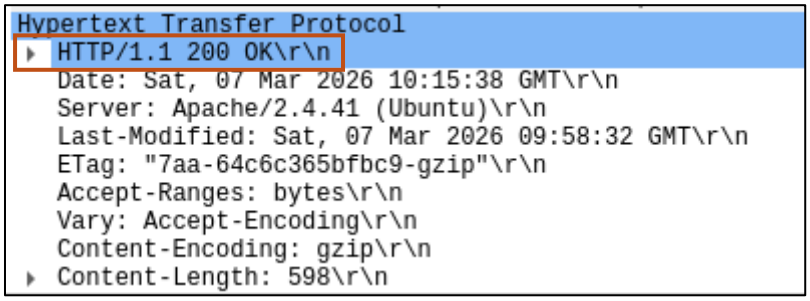
Dans les lignes filtrées, vous devriez voir deux lignes qui se suivent et qui sont particulièrement importantes. C'est la conversation classique entre un client et un serveur :

- **La requête (Le Client demande) :** Cherchez une ligne avec l'info `GET / HTTP/1.1`.



  - *Explication :* Cette trame est envoyée par le navigateur web vers le serveur. La méthode `GET` signifie littéralement "Donne-moi". Le client demande au serveur de lui envoyer le fichier situé à la racine (`/`), c'est-à-dire votre page d'accueil.

- **La réponse (Le Serveur répond) :** Cherchez la ligne suivante avec l'info `HTTP/1.1 200 OK`.



  - *Explication :* C'est la réponse d'Apache. Le code `200 OK` est le code standard du web pour dire "J'ai bien reçu ta demande, j'ai trouvé le fichier, et je te l'envoie !". Le contenu de votre site web est transporté juste après cette validation.

Étape 3 : Modification de la configuration (Passage au port 8080)

Jusqu'à présent, notre serveur Apache écoutait sur le port par défaut du trafic web (le port 80). Nous allons maintenant modifier sa configuration pour l'obliger à écouter sur un port différent : le **port 8080**. Cette manipulation est très utile lorsque plusieurs services tournent sur la même machine ou pour des raisons de sécurité.

1. Modification des fichiers de configuration d'Apache

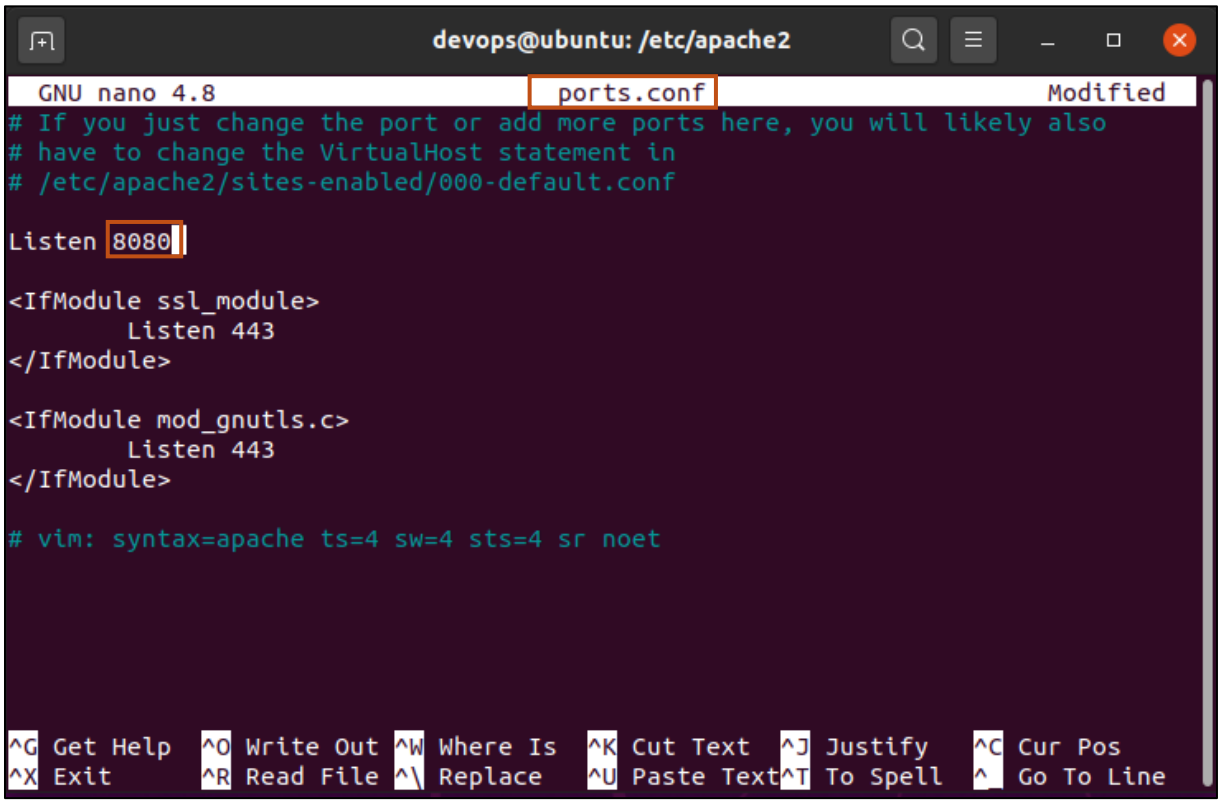
Pour changer le port d'écoute, nous devons modifier deux fichiers de configuration d'Apache. Comme ce sont des fichiers système, nous utiliserons l'éditeur de texte en ligne de commande `nano` avec les droits d'administrateur.

A-Modifier le fichier des ports d'écoute

Dans votre terminal, tapez la commande suivante :

```
sudo nano /etc/apache2/ports.conf
```

Dans ce fichier, cherchez la ligne `Listen 80`. Modifiez-la pour qu'elle devienne : `Listen 8080`. Pour sauvegarder et quitter l'éditeur `nano` : faites `Ctrl + O` (pour enregistrer), appuyez sur Entrée (pour valider le nom du fichier), puis `Ctrl + X` (pour quitter).

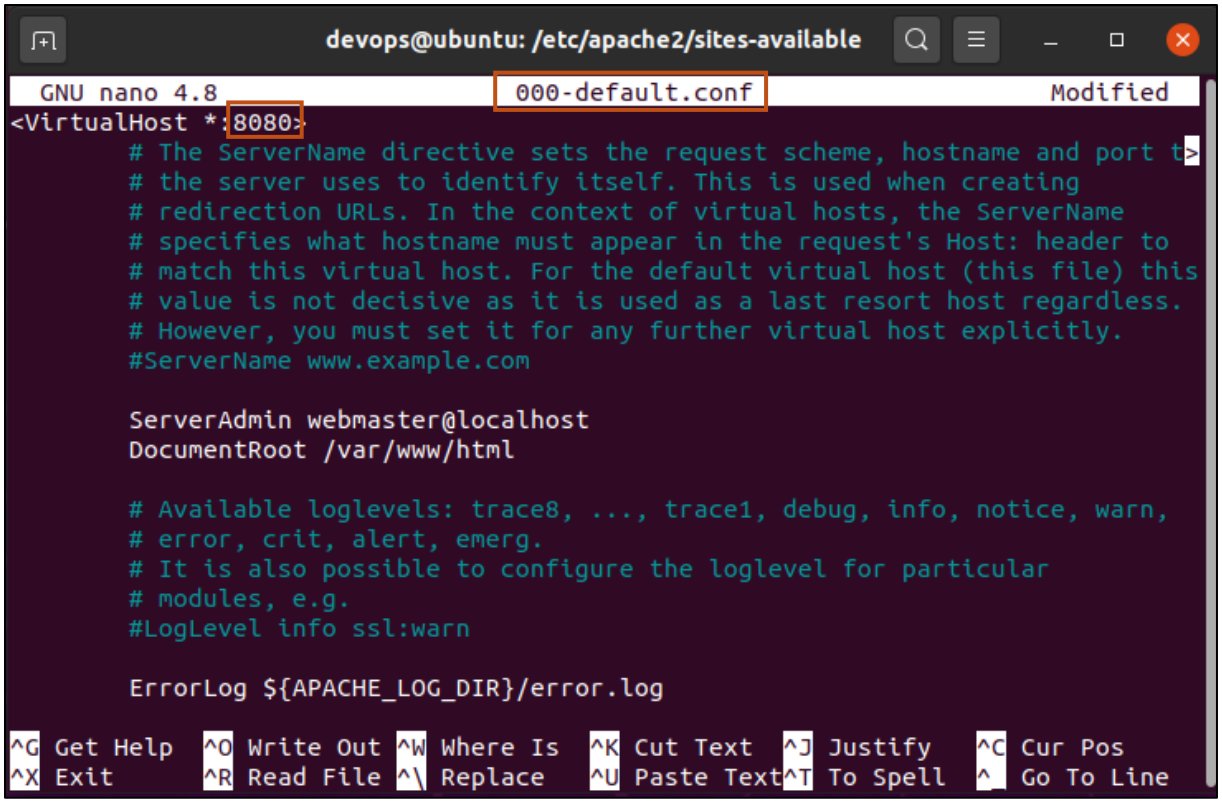


B-Modifier le fichier du site par défaut

Maintenant, nous devons dire à notre site web d'utiliser ce nouveau port. Ouvrez le fichier de configuration de notre site :

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

Tout en haut du fichier, changez la toute première balise `<VirtualHost *:80>` en `<VirtualHost *:8080>`. Sauvegardez et quittez de la même manière (`Ctrl + O`, `Entrée`, `Ctrl + X`).



2. Redémarrage du serveur et test

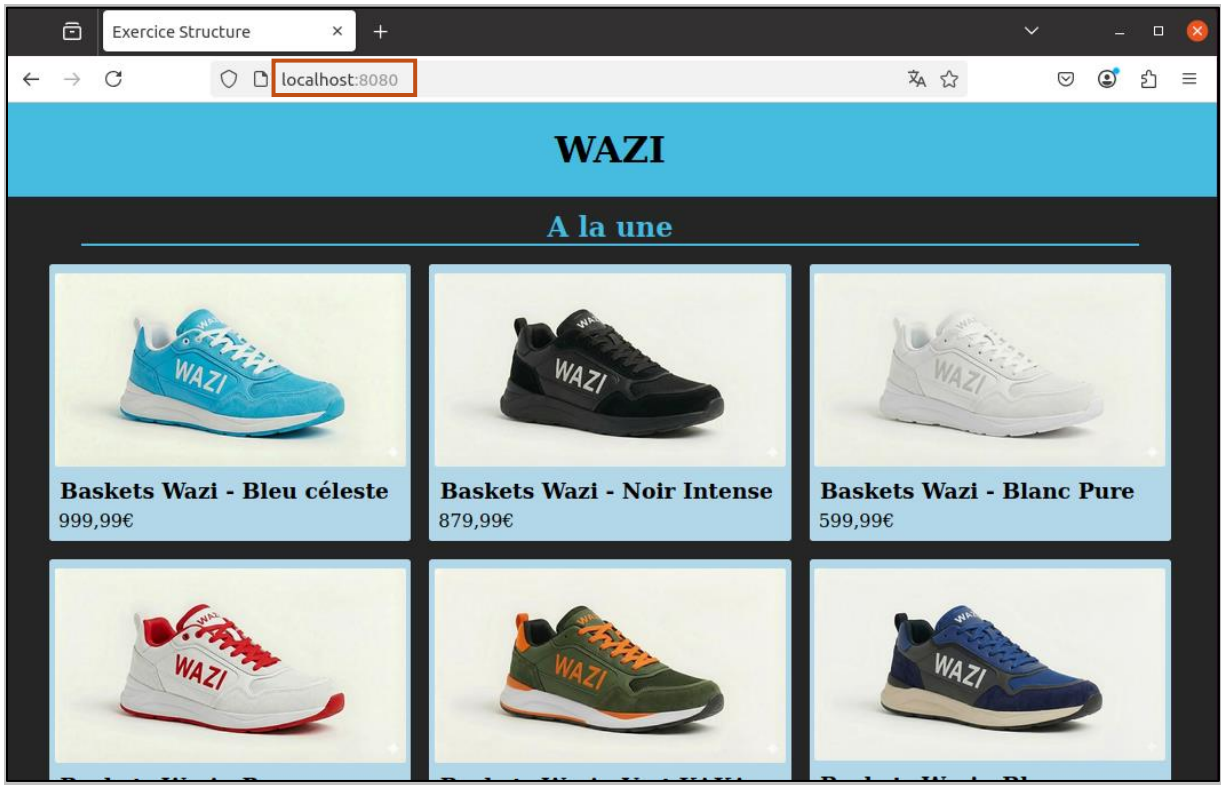
Pour qu'Apache prenne en compte ces nouveaux paramètres, il faut le redémarrer.

1. Tapez la commande suivante :

```
sudo systemctl restart apache2
```

2. Retournez dans votre navigateur web. Si vous actualisez la page `http://localhost`, vous devriez obtenir une erreur (car le serveur n'écoute plus sur ce port !).

3. Pour accéder à votre site, vous devez maintenant préciser le nouveau port dans l'adresse. Tapez : `http://localhost:8080`



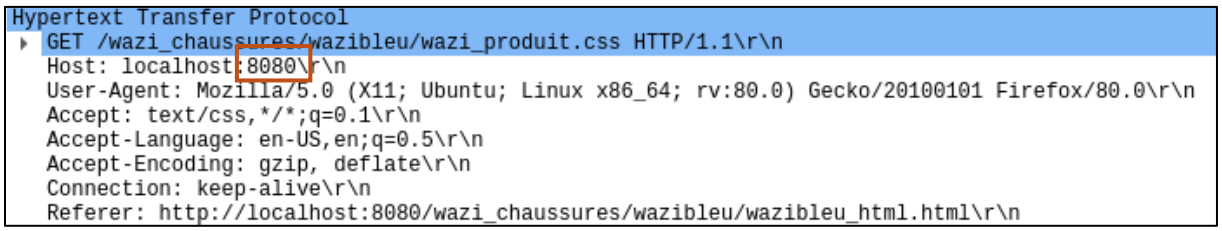
Votre site web personnel doit s'afficher à nouveau !

3. Analyse des trames HTTP modifiées

Maintenant que le port a changé, voyons comment cela se traduit sur le réseau.

1. Relancez une capture Wireshark (sur l'interface `Loopback` comme à l'étape 2).
2. Actualisez la page de votre navigateur (`http://localhost:8080`).
3. Arrêtez la capture dans Wireshark et appliquez à nouveau le filtre http.
4. Sélectionnez la requête `GET / HTTP/1.1`.

Commentaire sur les modifications : Dans le panneau des détails de Wireshark (Transmission Control Protocol), vous remarquerez que le "Destination Port" (Port de destination) lors de la requête du client n'est plus 80, mais **8080**. De même, le "Source Port" (Port source) lors de la réponse du serveur (`200 OK`) est maintenant **8080**. Le contenu des trames HTTP (le texte de la page web) reste identique, c'est uniquement la "porte d'entrée" du serveur qui a changé !



Étape 4 : Personnalisation de la configuration (Retour au port 80)

Maintenant que nous avons compris comment modifier le port d'écoute, nous allons remettre notre serveur sur le port standard (80). Cependant, pour des raisons d'organisation et de propreté, nous n'allons plus utiliser le fichier `000-default.conf`, mais créer le nôtre !

1. Rétablissement de l'écoute sur le port 80

La première chose à faire est de dire à Apache de réécouter sur le port 80.

1. Ouvrez à nouveau le fichier des ports :

```
sudo nano /etc/apache2/ports.conf
```

2. Remplacez `Listen 8080` par `Listen 80`.
3. Sauvegardez et quittez (`Ctrl + O`, `Entrée`, `Ctrl + X`).

2. Création d'un fichier de configuration personnalisé

Plutôt que de partir de zéro, nous allons copier le fichier par défaut pour créer notre propre fichier que nous appellerons `monsite.conf`.

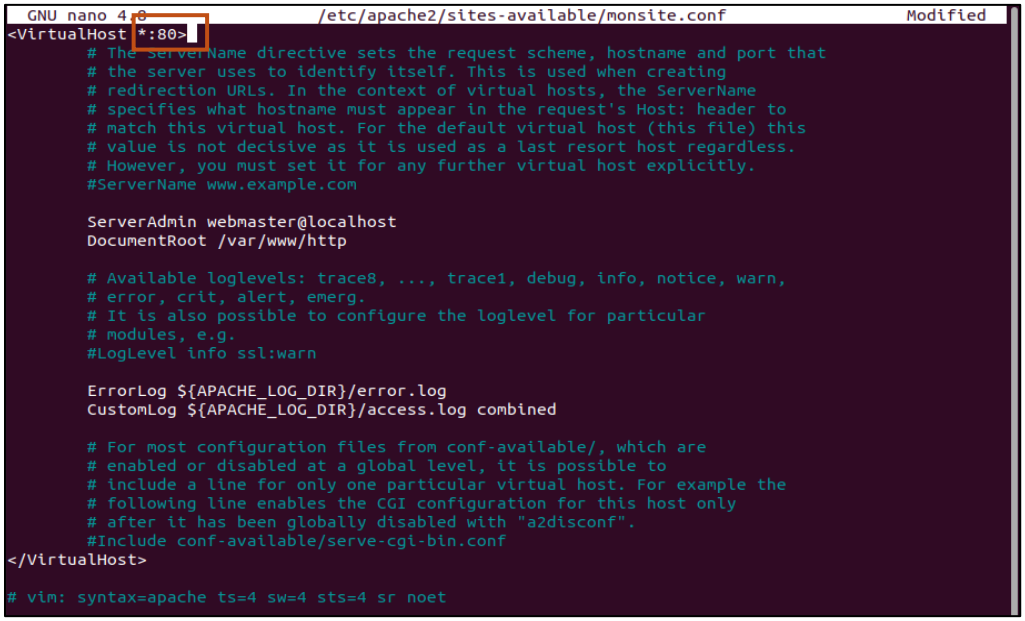
1. Tapez la commande suivante pour copier le fichier :

```
sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/monsite.conf
```

2. Ouvrez maintenant votre nouveau fichier pour le modifier :

```
sudo nano /etc/apache2/sites-available/monsite.conf
```

3. Assurez-vous que la première ligne est bien configurée pour le port 80 : `<VirtualHost *:80>`. Si elle est encore sur 8080, modifiez-la.
4. Sauvegardez et quittez.



3. Activation du nouveau site et désactivation de l'ancien

Apache possède des commandes très pratiques pour activer ou désactiver des sites sans avoir à supprimer les fichiers.

1. Nous allons d'abord désactiver le site par défaut avec la commande `a2dissite` :

```
sudo a2dissite 000-default.conf
```

2. Ensuite, nous allons activer notre nouveau site personnalisé avec la commande a2ensite :

```
sudo a2ensite monsite.conf
```

```
devops@ubuntu:/etc/apache2$ sudo a2dissite 000-default.conf
Site 000-default disabled.
To activate the new configuration, you need to run:
  systemctl reload apache2
devops@ubuntu:/etc/apache2$ sudo a2ensite monsite.conf
Enabling site monsite.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

Le terminal va vous indiquer que les modifications ont bien été prises en compte, mais qu'il faut recharger Apache pour qu'elles soient effectives.

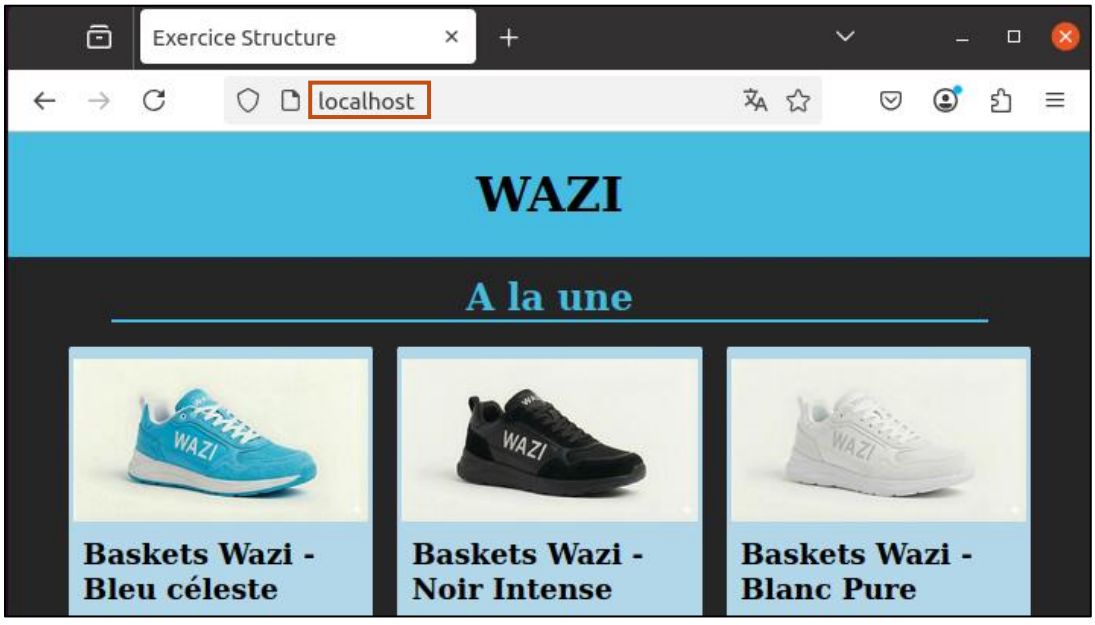
4. Redémarrage et vérification finale

Il ne reste plus qu'à appliquer les changements et vérifier que tout fonctionne comme prévu.

1. Rechargez la configuration d'Apache :

```
sudo systemctl reload apache2
```

2. Ouvrez votre navigateur web et tapez l'adresse standard : `http://localhost` (sans préciser de port, ce qui correspond au port 80 par défaut).



Si votre site s'affiche correctement, félicitations ! Votre serveur web utilise désormais votre propre fichier de configuration personnalisé sur le port standard.

Étape 5 : Intégration du principe de DNS

Jusqu'à présent, pour accéder à notre site web, nous devions taper l'adresse IP locale de la machine (`127.0.0.1`) ou le mot-clé localhost. Cependant, sur Internet, personne ne tape d'adresses IP pour visiter un site. On utilise des noms de domaine (comme `monsupersite.fr`), et c'est le système **DNS** (Domain Name System) qui se charge de faire la traduction entre ce nom et l'adresse IP du serveur.

Dans cette étape bonus, nous allons simuler ce fonctionnement en créant notre propre nom de domaine local.

1. Modification du fichier "hosts" (l'annuaire local)

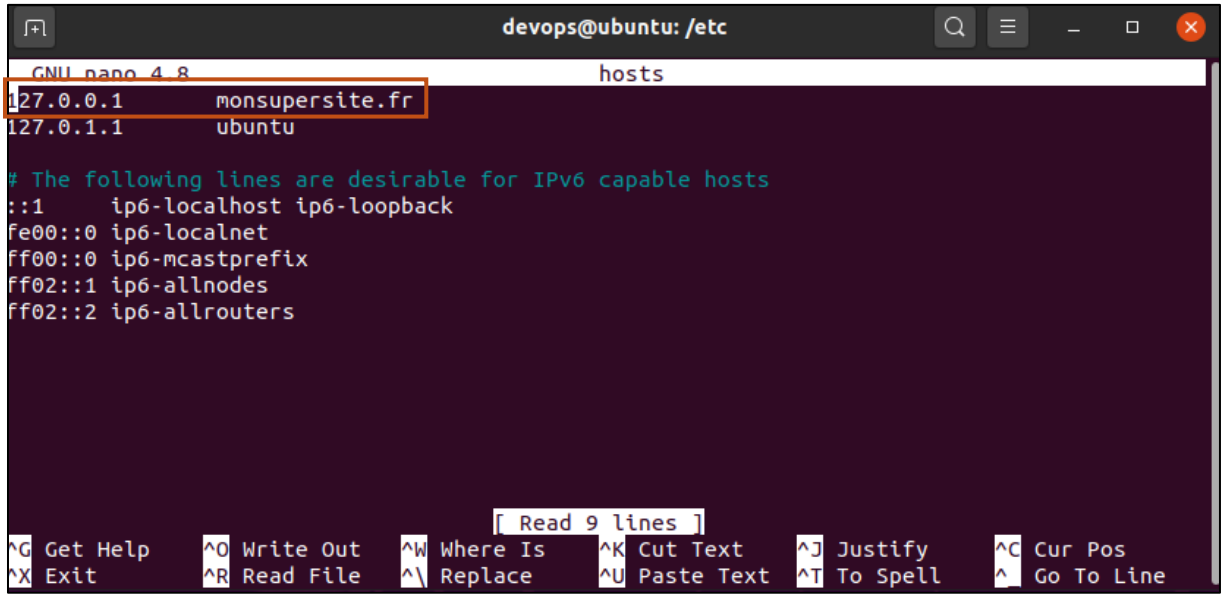
Avant d'interroger les serveurs DNS mondiaux, votre ordinateur consulte toujours un petit fichier local appelé `hosts`. Il agit comme un annuaire téléphonique personnel pour votre machine. Nous allons y ajouter notre faux nom de domaine.

1. Ouvrez le terminal et éditez le fichier `hosts` avec les droits d'administrateur :

```
sudo nano /etc/hosts
```

2. Dans ce fichier, vous devriez voir des lignes existantes. Allez tout en bas ou juste en dessous de la ligne `127.0.0.1 localhost`, et ajoutez l'association suivante (vous pouvez choisir le nom de domaine que vous souhaitez, par exemple `monsupersite.fr`) :

```
127.0.0.1     monsupersite.fr
```



3. Sauvegardez et quittez l'éditeur (`Ctrl + O`, `Entrée`, `Ctrl + X`).

Désormais, votre machine sait que si vous demandez `monsupersite.fr`, elle doit s'adresser à elle-même (`127.0.0.1`).

2. Configuration d'Apache (ServerName)

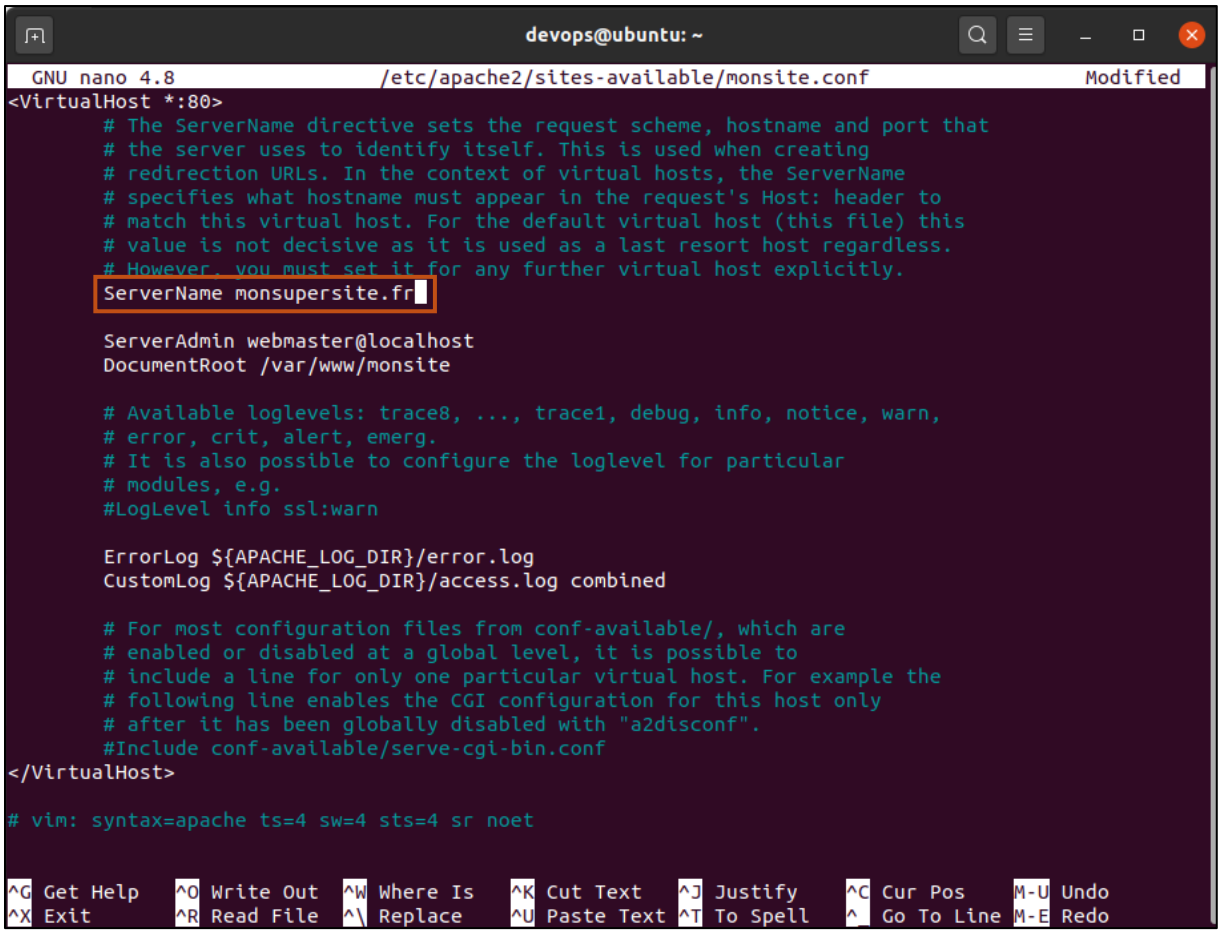
Bien que l'étape précédente suffise souvent si vous n'avez qu'un seul site, la bonne pratique est de dire à Apache qu'il est officiellement responsable de ce nom de domaine.

1. Réouvrez le fichier de configuration de votre site que nous avons créé à l'étape 4 :

   `sudo nano /etc/apache2/sites-available/monsite.conf`

2. En dessous de la balise `<VirtualHost *:80>`, ajoutez la ligne `ServerName` suivie de votre nom de domaine :

```
<VirtualHost *:80>
    ServerName monsuper.fr
    DocumentRoot /var/www/html
...
```



3. Sauvegardez et quittez (`Ctrl + O`, `Entrée`, `Ctrl + X`).

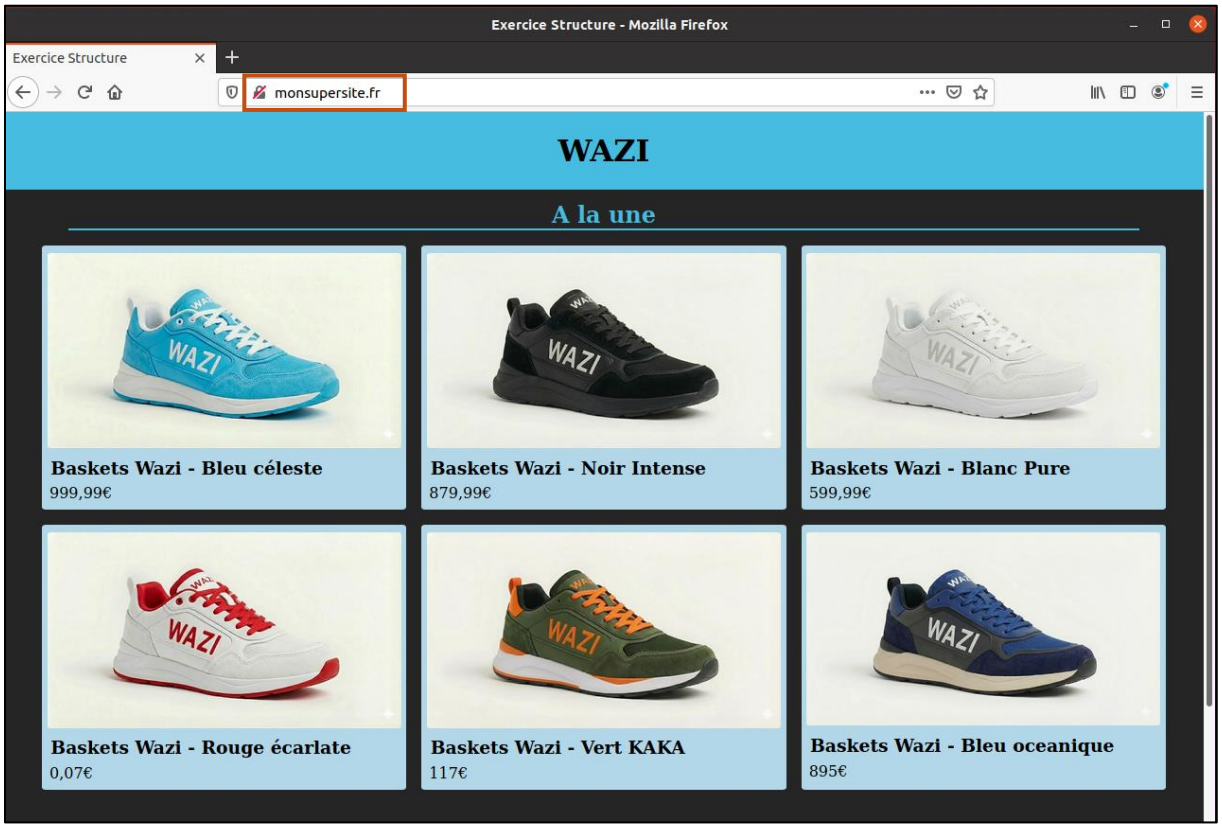
4. Rechargez Apache pour qu'il prenne en compte cette nouveauté :

   `sudo systemctl reload apache2`

3. Test final dans le navigateur

C'est le moment de vérité !

1. Ouvrez le navigateur web de votre machine virtuelle.
2. Dans la barre d'adresse, tapez votre tout nouveau nom de domaine : `http://www.mon-super-site.local`
3. Appuyez sur Entrée. Votre site s'affiche !



Grâce au fichier `hosts`, votre navigateur a traduit le nom en `127.0.0.1`. La requête a été envoyée sur le port 80. Apache l'a reçue, a vu que la requête demandait `monsupersite.fr`, a regardé dans `monsite.conf`, et a renvoyé les fichiers correspondants.

Étape 6 : Configuration d'un serveur Nginx

Dans le monde professionnel, Apache n'est pas le seul serveur web existant. **Nginx** (prononcé "Engine-X") est un autre serveur extrêmement populaire, réputé pour sa légèreté et sa rapidité. Dans cette étape, nous allons faire cohabiter les deux serveurs sur la même machine !

1. Le piège du conflit de ports

Avant de commencer, il faut comprendre une règle fondamentale : **deux services différents ne peuvent pas écouter sur le même port en même temps**. Actuellement, Apache utilise le port 80. Si nous installons Nginx tel quel, il essaiera aussi d'utiliser le port 80 et affichera une erreur. Pour qu'ils cohabitent, nous devrons assigner un nouveau port à Nginx (par exemple le port 8088).

2. Installation de Nginx

1. Ouvrez votre terminal.
2. Mettez à jour la liste des paquets disponibles (toujours une bonne pratique avant une installation) :

```
sudo apt update
```

3. Installez le serveur Nginx avec la commande suivante :

```
sudo apt install nginx
```

(Appuyez sur 'O' ou 'Y' si le système vous demande de confirmer l'installation).

3. Modification du port d'écoute de Nginx

Dès la fin de l'installation, Nginx a probablement tenté de démarrer et a échoué à cause d'Apache. Nous allons corriger son fichier de configuration.

1. Ouvrez le fichier de configuration par défaut de Nginx avec l'éditeur `nano` :

```
sudo nano /etc/nginx/sites-available/default
```

2. Cherchez les lignes suivantes au début du fichier :

```
listen 80 default_server;
listen [::]:80 default_server;
```

3. Modifiez le `80` par `8088` sur ces deux lignes pour que Nginx écoute sur ce nouveau port :

```
listen 8088 default_server;
listen [::]:8088 default_server;
```

```
server {
        listen 8088 default_server;
        listen [::]:8088 default_server;

        # SSL configuration
        #
        # listen 443 ssl default_server;
```

4. Sauvegardez et quittez l'éditeur (`Ctrl + O`, `Entrée`, `Ctrl + X`).

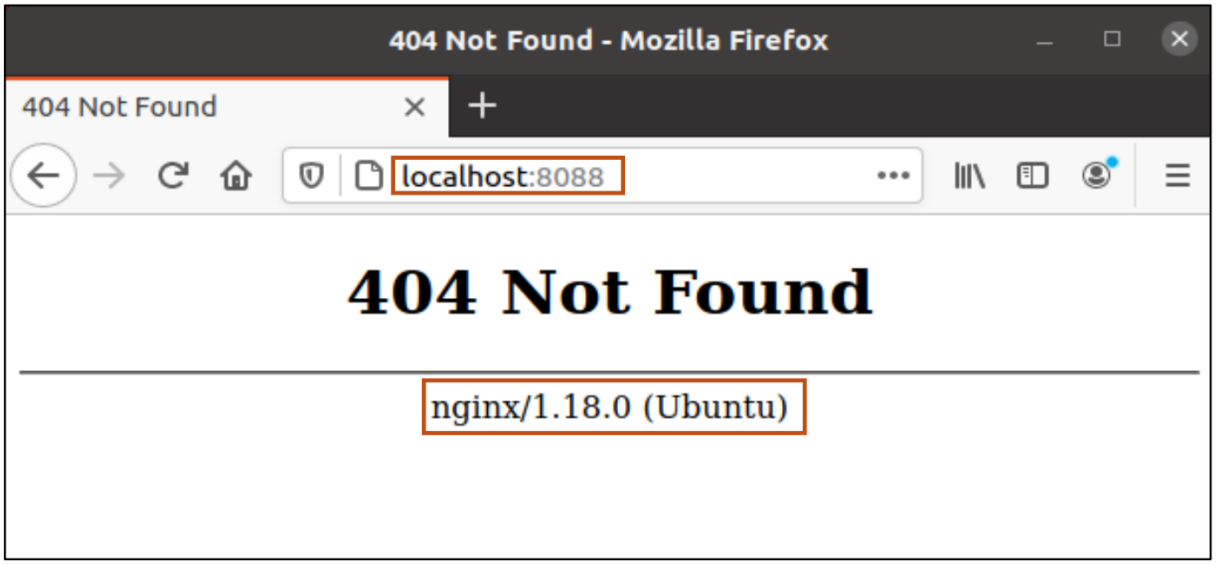
4. Démarrage et test de cohabitation

Maintenant que chaque serveur a sa propre "porte d'entrée" (Apache sur le 80, Nginx sur le 8088), nous pouvons démarrer Nginx en toute sécurité.

1. Redémarrez le service Nginx pour appliquer la modification :

```
sudo systemctl restart nginx
```

2. Vérifiez que tout fonctionne en ouvrant votre navigateur web.
3. Tapez d'abord `http://localhost` : Vous devriez voir votre site personnel (C'est **Apache** qui répond sur le port 80).
4. Ouvrez un nouvel onglet et tapez `http://localhost:8088` : Vous devriez voir la page d'accueil par défaut "Welcome to nginx!" (C'est **Nginx** qui répond sur le port 8088).



Félicitations, vous avez maintenant deux serveurs web distincts qui tournent simultanément sur votre machine !

Étape 7 : Requêtes HTTP manuelles avec Telnet

Dans cette étape, nous allons nous passer de notre navigateur web (comme Firefox ou Chrome) et discuter directement avec notre serveur Apache en lui parlant dans sa langue : le protocole HTTP. Pour cela, nous allons utiliser un outil en ligne de commande très ancien mais redoutable : **Telnet**.

1. Connexion au serveur web via Telnet

Telnet permet d'ouvrir une connexion brute vers un port spécifique d'une machine. Nous allons nous connecter au port 80 de notre propre machine, là où Apache écoute nos requêtes.

1. Ouvrez votre terminal.
2. (Optionnel) Si la commande Telnet n'est pas trouvée, installez l'outil avec la commande :

```
sudo apt install telnet
```

3. Tapez la commande suivante pour établir la connexion avec votre serveur local sur le port 80 :

```
telnet localhost 80
```

Le terminal va vous répondre `Trying 127.0.0.1`... puis `Connected to localhost.`. Cela signifie que la connexion TCP est établie et qu'Apache vous écoute. Il attend maintenant votre requête HTTP.

2. Envoi de la requête HTTP manuelle

C'est ici qu'il faut être précis. Vous allez taper exactement ce qu'un navigateur web envoie en arrière-plan. Attention, avec Telnet, vous ne pouvez pas utiliser la touche "Retour arrière" pour corriger une faute de frappe, il faut taper d'une traite !

1. Tapez attentivement la ligne suivante (qui demande la page d'accueil) et appuyez sur **Entrée** :

```
GET / HTTP/1.1
```

2. Tapez ensuite la ligne suivante (qui précise à quel hôte on s'adresse, obligatoire en HTTP/1.1) et appuyez sur **Entrée** :

```
Host: localhost
```

3. **Étape cruciale :** Appuyez une **deuxième fois sur Entrée** (sur une ligne vide). C'est le signal standard du protocole HTTP pour dire au serveur : "J'ai fini de t'envoyer mes en-têtes, tu peux me répondre".

3. Analyse de la réponse du serveur

Si vous avez bien respecté la syntaxe et le double appui sur la touche Entrée, le terminal va soudainement se remplir de texte !

Le serveur Apache vient de vous répondre. Vous pourrez y observer deux parties distinctes :

- **Les en-têtes de réponse :** Les premières lignes générées par Apache, commençant par le fameux `HTTP/1.1 200 OK`, suivi d'informations sur le serveur, la date, et le type de contenu.
- **Le corps de la réponse :** Juste en dessous, vous verrez apparaître tout le code HTML brut de la page d'accueil de votre site personnel (les balises `<html>`, `<head>`, `<body>`, etc.).

```
devops@ubuntu:/$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.1
Host: google.fr

HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Tue, 10 Mar 2026 16:24:48 GMT
Content-Type: text/html
Content-Length: 1962
Last-Modified: Tue, 03 Mar 2026 16:06:59 GMT
Connection: keep-alive
ETag: "69a70723-7aa"
Accept-Ranges: bytes

<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="./wazi_accueil.css">
    <title>Exercice Structure</title>
  </head>
  <body>
        <header>
                <h1>WAZI</h1>
        </header>
        <main id="Produit">

                <h2>A la une</h2>
                <div id="produit">
                <section>
                        <h3>Baskets Wazi - Bleu céleste</h3>
                        <a href="wazi_chaussures/wazibleuclair/wazibleuclair_html.html">
                                <img src="wazi_chaussures/wazibleuclair/wazibleuclair.jpg" alt="sabat" />
                        </a>
                        <ul>
                                <li>999,99€</li>
                        </ul>
```

Telnet fermera ensuite la connexion automatiquement (Connection closed by foreign host). Vous venez de faire manuellement le travail d'un navigateur web !

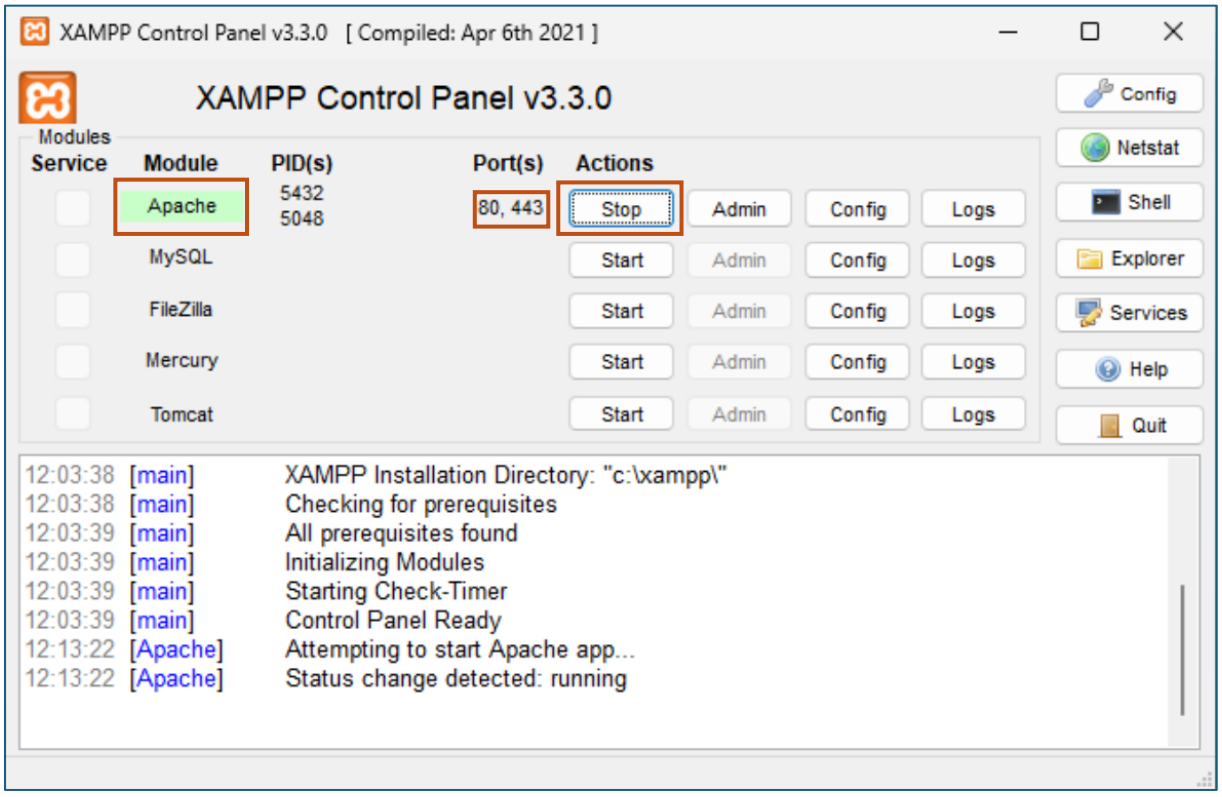
Étape 8 : Serveur web Apache sous Windows

Jusqu'à présent, nous avons travaillé exclusivement sur un système d'exploitation Linux. Cependant, il est tout à fait possible de faire tourner un serveur web Apache sous Windows. Pour cette étape, vous devrez utiliser la machine virtuelle Windows issue de votre SAE du premier semestre.

1. Téléchargement des outils et lancement d'Apache

Sous Windows, l'installation d'Apache se fait généralement via un utilitaire regroupant plusieurs outils web. Le plus connu pour les débutants s'appelle **XAMPP**.

1. Depuis votre machine virtuelle Windows connecté à Internet, rendez-vous sur le site officiel d'Apache Friends (https://www.apachefriends.org).
2. Téléchargez la version de **XAMPP pour Windows**.
3. Suivez les étapes de l'assistant d'installation (vous pouvez laisser les paramètres par défaut, le dossier d'installation standard sera `C:\xampp`).
4. À la fin de l'installation, ouvrez le **XAMPP Control Panel** (le panneau de contrôle).
5. Sur la ligne correspondant à "Apache", cliquez sur le bouton **Start**. Si le nom "Apache" se surligne en vert et que des numéros de ports apparaissent (ex: 80, 443), votre serveur est lancé !



2. Déploiement de votre site personnel

Sous Linux, nous avons vu que le dossier racine du serveur web (le *DocumentRoot*) était `/var/www/html`. Sous Windows avec XAMPP, ce dossier par défaut s'appelle `htdocs`.

1. Ouvrez l'explorateur de fichiers Windows.
2. Naviguez jusqu'au dossier d'installation d'Apache : `C:\xampp\htdocs`.
3. Ce dossier contient déjà des fichiers par défaut. Vous pouvez les supprimer pour faire de la place.
4. Copiez les fichiers de votre site web (depuis votre clé USB) et collez-les directement dans ce dossier `C:\xampp\htdocs`.

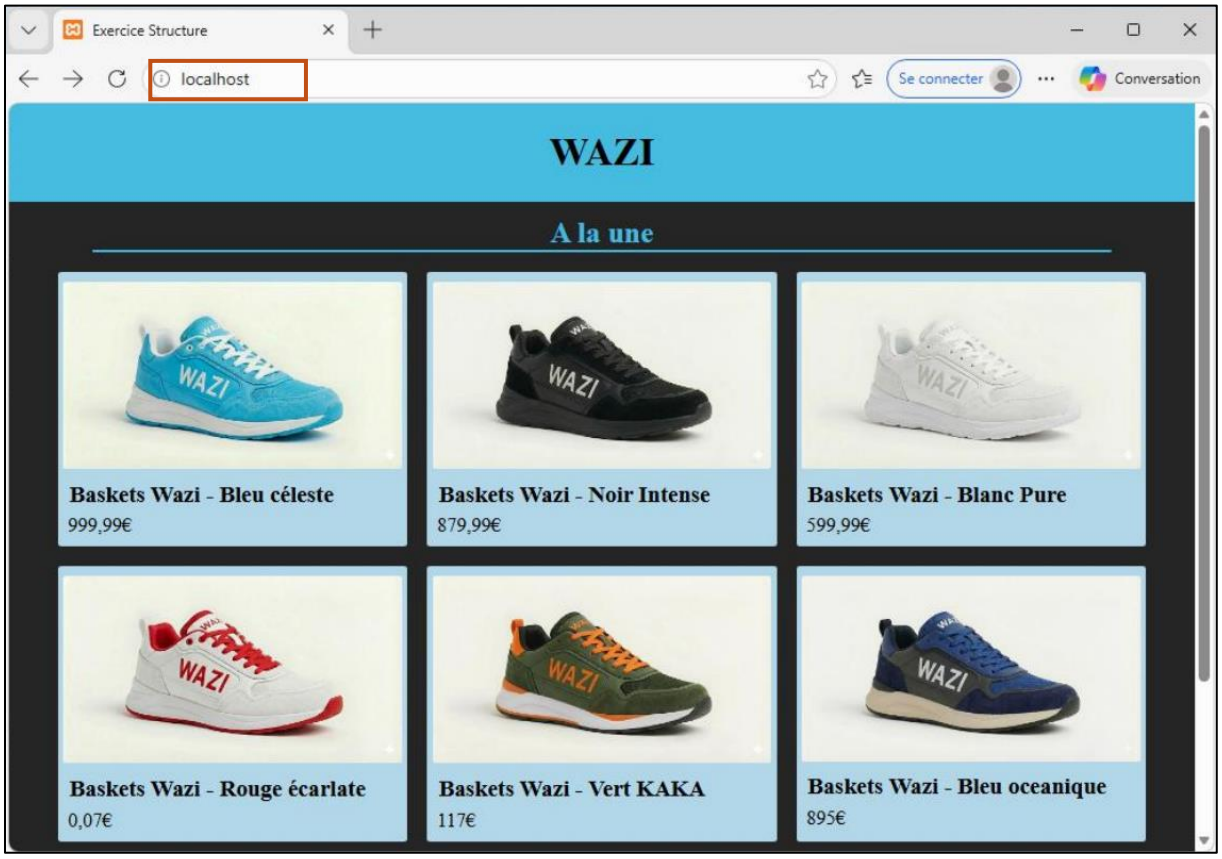
```
    Répertoire : C:\xampp\htdocs


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        20/03/2026     12:19                image_produit
d-----        20/03/2026     12:19                wazi_achat
d-----        20/03/2026     12:19                wazi_chaussures
-a----        16/12/2025     13:34           1962 index.html
-a----        16/12/2025     13:33           1933 wazi_accueil.css
```

3. Test d'affichage du site

Il ne reste plus qu'à vérifier que le serveur Apache sous Windows réagit correctement à vos requêtes.

1. Ouvrez le navigateur web de votre machine virtuelle Windows (Edge, Firefox ou Chrome).
2. Dans la barre d'adresse, tapez simplement : `http://localhost`
3. Appuyez sur Entrée.



Votre site web doit s'afficher ! Le principe reste exactement le même que sous Linux : le navigateur envoie une requête GET sur le port 80 de la machine locale, et le processus Apache (qui tourne maintenant sous Windows) lit les fichiers dans `htdocs` pour vous les renvoyer.

Étape 9 : Analyse des méthodes GET et POST

Jusqu'ici, notre navigateur s'est contenté de "demander" des pages au serveur. C'est ce qu'on appelle la méthode **GET** (obtenir). Mais que se passe-t-il lorsque vous remplissez un formulaire de connexion ou envoyez un message ? Le navigateur doit envoyer des données au serveur : il utilise alors la méthode **POST** (publier/envoyer).

Nous allons observer la différence entre ces deux méthodes directement dans les trames réseau avec Wireshark.

1. Préparation : Création d'une page avec un formulaire

Pour tester la méthode POST, nous avons besoin d'un formulaire. Nous allons créer une petite page web de test directement sur notre serveur Apache (sous Linux).

1. Ouvrez votre terminal et créez un nouveau fichier nommé formulaire.html dans le dossier de votre site web :

```
sudo nano /var/www/html/formulaire.html
```

2. Copiez-collez (ou tapez) le code HTML suivant, qui crée un formulaire basique envoyant des données en POST :

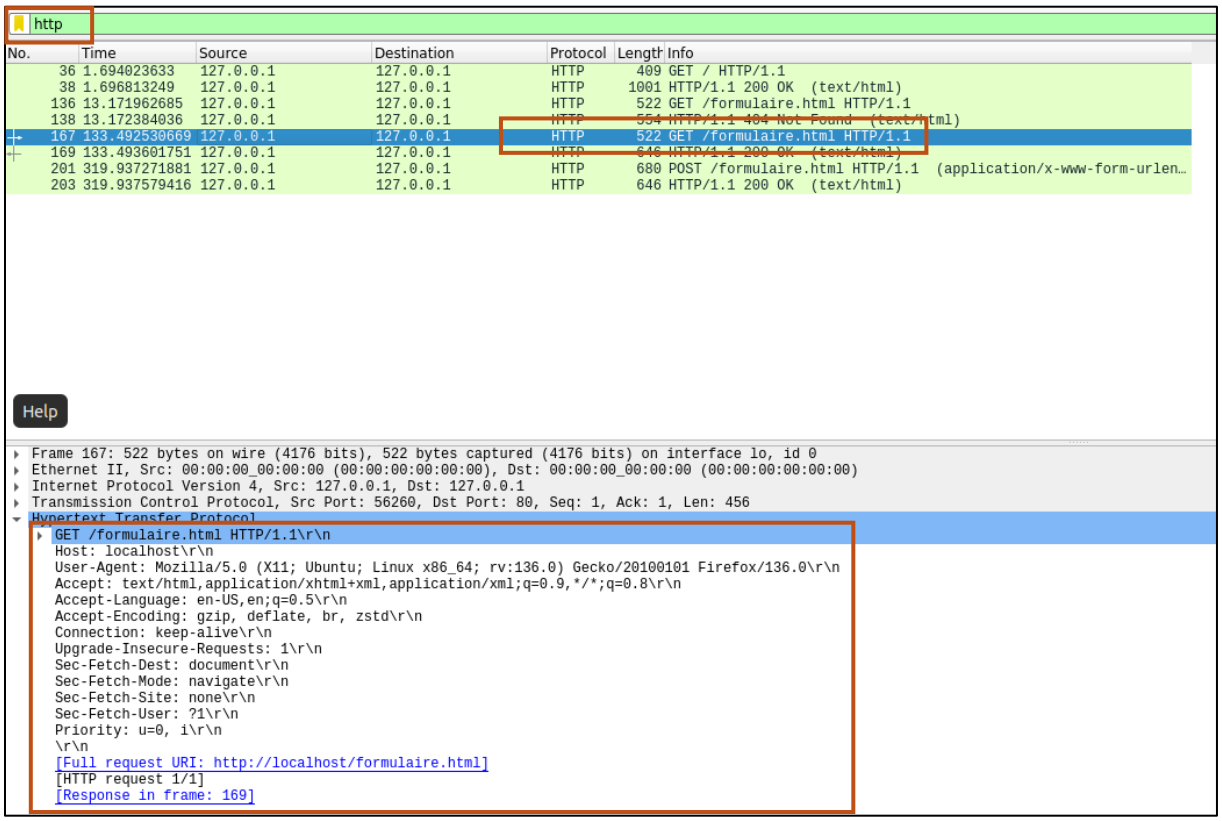
```
<!DOCTYPE html>
<html>
<head><title>Test GET et POST</title></head>
<body>
    <h2>Mon formulaire de test</h2>
    <form action="/formulaire.html" method="POST">
        <label for="prenom">Votre prénom :</label>
        <input type="text" id="prenom" name="prenom"><br><br>
        <input type="submit" value="Envoyer au serveur">
    </form>
</body>
</html>
```

3. Sauvegardez et quittez (`Ctrl + O`, `Entrée`, `Ctrl + X`).

2. Capture et observation de la méthode GET

Nous allons d'abord capturer le moment où le navigateur demande à afficher la page du formulaire.

1. Lancez Wireshark (commande `sudo wireshark`) et démarrez une capture sur l'interface `Loopback` (ou `lo`).
2. Ouvrez votre navigateur web et tapez l'adresse de votre nouvelle page : `http://local-host/formulaire.html`
3. Retournez dans Wireshark, arrêtez la capture et appliquez le filtre `http`.
4. **Analyse :** Cliquez sur la ligne contenant `GET /formulaire.html HTTP/1.1`.

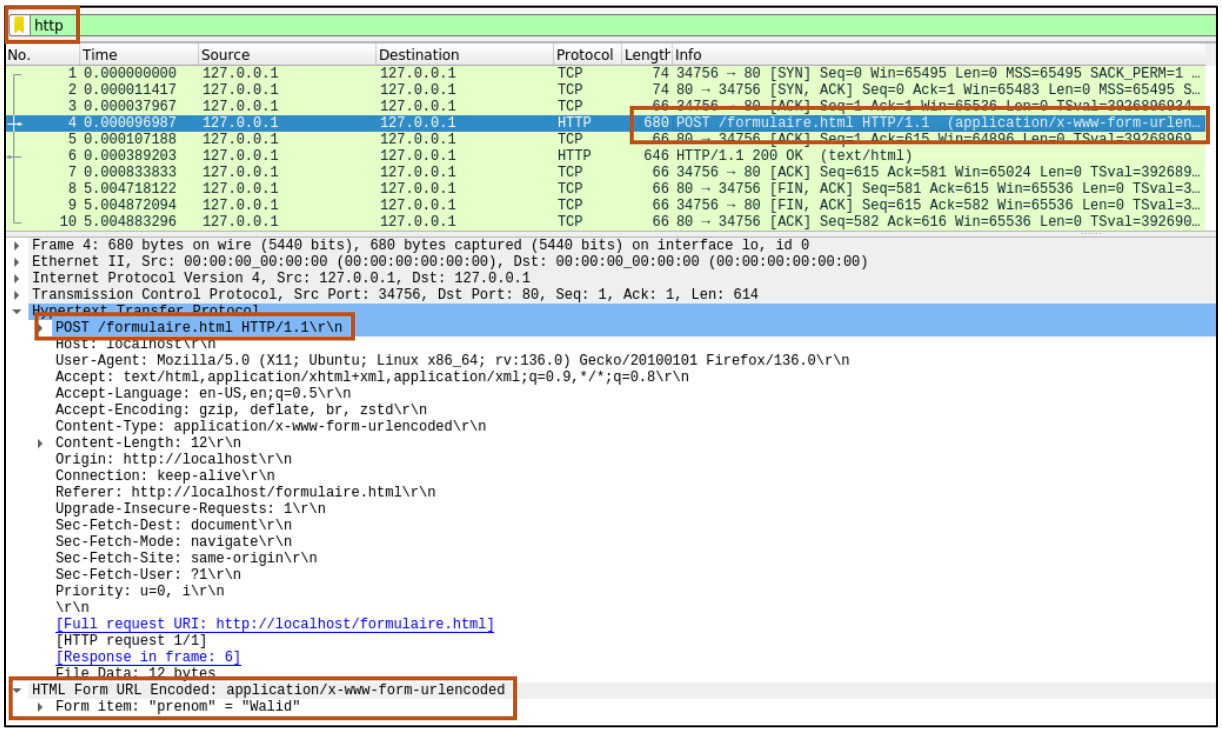


   - *Explication :* Cette trame est une requête `GET`. Le client dit simplement "Envoie-moi le fichier formulaire.html". Aucune donnée personnelle n'est envoyée par le client dans cette requête, il ne fait que demander l'affichage.

3. Capture et observation de la méthode POST

Maintenant, nous allons remplir le formulaire et l'envoyer au serveur.

1. Relancez une nouvelle capture dans Wireshark (toujours sur `Loopback`).
2. Retournez dans votre navigateur (qui affiche le formulaire). Tapez un prénom dans la case (par exemple "Alice" ou "Bob") et cliquez sur le bouton **"Envoyer au serveur"**.
3. Retournez dans Wireshark, arrêtez la capture et appliquez le filtre `http`.
4. **Analyse :** Cette fois-ci, cherchez la ligne contenant `POST /formulaire.html HTTP/1.1`. Cliquez dessus.



- *Explication :* Le navigateur n'a pas utilisé `GET`, mais `POST`. Regardez dans le panneau du bas de Wireshark : tout en bas des détails, vous devriez voir une section nommée **HTML Form URL Encoded**. Si vous la déroulez, vous verrez la donnée exacte que vous avez tapée (ex: `Form item: "prenom" = "Walid"`).

  La méthode `POST` permet de "cacher" les données envoyées à l'intérieur du corps de la requête `HTTP` (le *body*), contrairement à la méthode `GET` qui ne fait que demander une ressource.

Conclusion

Félicitations, vous êtes arrivé au terme de ce manuel d'utilisation !

À travers ces différentes étapes, vous avez pu découvrir qu'un serveur web n'est pas une "boîte noire" complexe, mais un service logique que l'on peut configurer et analyser. Vous êtes désormais capable de :

- Déployer un site web localement avec **Apache** ou **Nginx**.
- Modifier les fichiers de configuration pour gérer les ports d'écoute (80, 8080) et les noms de domaine locaux (DNS).
- Analyser en profondeur les communications réseau entre un client et un serveur grâce à **Wireshark**, en comprenant la différence fondamentale entre les méthodes **GET** et **POST**.
- Dialoguer manuellement avec un serveur en utilisant le protocole HTTP brut via **Telnet**.

Votre serveur web est maintenant parfaitement configuré et fonctionnel en local. La prochaine étape logique de votre apprentissage sera de l'ouvrir sur l'extérieur en le connectant à un véritable réseau composé de routeurs et d'autres machines clientes.