REPUBLIQUE DEMOCRATIQUE DU CONGO

Ministère de l'enseignement supérieur et universitaire

FACULTE DE SCIENCE ET TECHNOLOGIE

UNIVERSITE DE KINSHASA

MENTION MATHEMATIQUE, STATISTIQUE ET INFORMATIQUE
B.P 190 KINSHASA

PROJET DU SYSTEME D'EXPLOITATION

L2 LMD

MISE EN PLACE D'UN SERVEUR WEB APACHE SECURISE AVEC TLS (HTTPS)

GROUPE N° 01

- **FERUZI KABONGO MATHIEU**
- **ELONGO IKWA WILLIAM**
- **MADIMBA MUTOMBO ERNEST**
- **KAMUKENJI KABEYA PREGANA**
- **NZOLO MONGA-NZOLO BENJAMIN**
- **MASUNDA KUMBI FAITH**
- **BUABUA TSHIMAKINDA CHADRACK**
- **BATUPENDI KALOMBO GLORIA**
- **KIANZA GILAPHE CHRISTELLA**
- **TAKONGO LWAMBA CHRISTELLE**

Prof. Dr Kasengedia Motumbe Pierre

Collaborateurs :

- ***Doctorant Junior Kaningini***
- ***Fipa Bukusu & Ferdinand Djungu & All***

ANNEE ACADEMIQUE 2025-2026

Résumé

Ce projet avait pour objectif principal la mise en place, la configuration et la compréhension approfondie d'un serveur web basé sur **Apache HTTP Server** dans un environnement Linux. Il s'inscrit dans une démarche pédagogique visant à maîtriser les fondements de l'administration système et des services web, en mettant l'accent sur la structure interne du serveur, son fonctionnement et ses mécanismes de configuration.

Dans un premier temps, le projet a consisté à étudier **l'architecture globale d'Apache**, notamment l'organisation de ses fichiers et répertoires essentiels. Cette analyse a permis de comprendre le rôle de chaque composant, tels que les fichiers de configuration principaux, les dossiers dédiés aux modules, aux sites web, ainsi qu'aux journaux (**logs**). L'objectif était d'identifier clairement la fonction de chaque élément et son interaction avec le reste du système.

Ensuite, une attention particulière a été portée à la configuration du serveur web. Cela inclut la gestion des ports d'écoute, la définition des paramètres globaux du serveur, ainsi que l'utilisation des hôtes virtuels afin d'héberger et administrer un ou plusieurs sites web de manière structurée. Cette étape a permis de mettre en pratique les notions théoriques liées au fonctionnement d'Apache et de comprendre comment le serveur interprète et applique les directives de configuration.

Par ailleurs, le projet a intégré l'étude des **permissions et des propriétaires de fichiers sous Linux**, indispensables au bon fonctionnement et à la sécurisation du serveur web. La maîtrise des droits d'accès, combinée à la gestion correcte des liens symboliques, a permis d'assurer une organisation cohérente des fichiers tout en respectant les contraintes de sécurité imposées par le système d'exploitation.

Enfin, ce travail a permis de développer une vision globale et cohérente du rôle d'Apache dans une infrastructure web. Il a renforcé la compréhension des interactions entre le système Linux et le serveur web, tout en consolidant les compétences pratiques nécessaires à l'administration, au dépannage et à la sécurisation d'un service web en environnement réel.

Abstract

This project aimed to design, configure, and develop an in-depth understanding of a web server based on **Apache HTTP Server** in a Linux environment. It is part of an educational approach intended to master the fundamentals of system administration and web services, with particular emphasis on the internal structure of the server, its operation, and its configuration mechanisms.

Initially, the project focused on the study of Apache's overall architecture, particularly the organization of its essential files and directories. This analysis made it possible to understand the role of each component, including the main configuration files, directories dedicated to modules, websites, and **log files**. The objective was to clearly identify the function of each element and its interaction with the rest of the system.

Subsequently, special attention was given to the configuration of the web server. This included the management of listening ports, the definition of global server parameters, and the use of virtual hosts to host and manage one or more websites in a structured manner. This phase enabled the practical application of theoretical concepts related to Apache's operation and clarified how the server interprets and applies configuration directives.

Furthermore, the project incorporated the study of **file permissions and ownership under Linux**, which are essential for the proper functioning and security of a web server. Mastery of access rights, combined with the correct management of symbolic links, ensured a coherent file organization while respecting the security constraints imposed by the operating system.

Finally, this work provided a comprehensive and coherent view of Apache's role within a web infrastructure. It strengthened the understanding of interactions between the Linux system and the web server, while consolidating the practical skills required for the administration, troubleshooting, and security of a web service in a real-world environment.

Introduction

Avec le développement croissant des technologies de l'information et de la communication, les services web occupent une place centrale dans le fonctionnement des systèmes informatiques modernes. Les serveurs web constituent ainsi un élément fondamental de l'infrastructure réseau, permettant la mise à disposition de contenus et de services accessibles à distance. Parmi les solutions existantes, **Apache HTTP Server** demeure l'un des serveurs web les plus utilisés en raison de sa robustesse, de sa flexibilité et de sa large communauté.

Dans le cadre du cours de **Système d'Exploitation**, ce projet vise à approfondir la compréhension pratique du fonctionnement d'un serveur web sous un environnement Linux. Il s'agit non seulement d'installer et de configurer Apache, mais également d'analyser son architecture interne, la gestion de ses fichiers de configuration et les mécanismes qui assurent son bon fonctionnement.

Le projet met un accent particulier sur la configuration des hôtes virtuels, la gestion des permissions et des droits d'accès au sein du système Linux, ainsi que sur la sécurisation des communications par le protocole **HTTPS (TLS).** Ces aspects sont essentiels pour garantir la fiabilité, la sécurité et la maintenabilité d'un serveur web en environnement réel.

Ainsi, ce travail constitue une introduction complète à l'administration d'un serveur Apache, permettant de relier les notions théoriques du cours aux réalités pratiques de l'administration système et de la sécurité informatique.

Problématique et Objectifs

1. Problématique

La mise en place d'un serveur web sous un système d'exploitation Linux constitue une tâche essentielle mais complexe pour tout administrateur système. Bien qu'Apache HTTP Server soit largement utilisé et documenté, sa configuration repose sur de nombreux fichiers, directives et mécanismes qui peuvent entraîner des erreurs de fonctionnement ou des failles de sécurité lorsqu'ils sont mal maîtrisés. Parmi les difficultés les plus courantes figurent la mauvaise gestion des permissions, les erreurs dans la configuration des hôtes virtuels, ainsi que l'absence ou la mauvaise implémentation du chiffrement des communications.

Dans un contexte académique, ces difficultés soulèvent la question suivante :

Comment installer, configurer et sécuriser efficacement un serveur Apache sous Linux tout en respectant les bonnes pratiques d'administration système et de sécurité informatique ?

2. Objectifs

- ❖ Objectif général

L'objectif général de ce projet est de mettre en place un serveur web Apache fonctionnel et sécurisé sous Linux, tout en comprenant son architecture, son fonctionnement et les mécanismes de configuration essentiels.

- ❖ Objectifs spécifiques

De manière plus précise, ce projet vise à :

- Installer et démarrer le serveur Apache HTTP Server sur un système Linux ;
- Analyser l'architecture d'Apache et le rôle de ses principaux fichiers et répertoires ;
- Configurer des hôtes virtuels pour l'hébergement de sites web ;
- Gérer correctement les permissions et les propriétaires des fichiers et répertoires ;
- Mettre en œuvre la sécurisation des communications à l'aide du protocole HTTPS ;
- Tester et valider le bon fonctionnement du serveur après configuration.

Méthodologie

Plan de la méthodologie

- Approche adoptée
- Environnement de travail
- Outils utilisés
- Étapes de réalisation

1. Approche adoptée

Le projet a été réalisé suivant une **approche progressive et séquentielle**, permettant de valider chaque étape avant de passer à la suivante.

Cette méthode repose sur une logique simple :
Installation du service → Configuration du serveur → Sécurisation des communications.
Chaque phase a été testée afin de s'assurer du bon fonctionnement du serveur avant d'engager l'étape suivante.

2. Environnement de travail

Le serveur a été déployé sur un système d'exploitation **Linux de type Ubuntu**, choisi pour sa stabilité et sa compatibilité native avec Apache HTTP Server. Toutes les opérations ont été effectuées en ligne de commande à l'aide du terminal, ce qui permet un meilleur contrôle des configurations et une traçabilité des actions réalisées.

3. Outils utilisés

Les principaux outils utilisés dans le cadre de ce projet sont :

- **Linux** : système d'exploitation du serveur ;
- **Apache HTTP Server** : serveur web utilisé pour l'hébergement ;
- **OpenSSL** : outil de génération et de gestion des certificats TLS ;
- **Commandes système Linux** : `apt`, `systemctl`, `a2enmod`, `a2ensite`, `openssl`, utilisées pour l'installation, la gestion des services et la configuration.

4. Étapes de réalisation

1. **Installation**

   Installation du serveur Apache à l'aide du gestionnaire de paquets, suivi de la vérification du bon fonctionnement du service et de l'accessibilité via le navigateur.

2. **Configuration**

   Analyse de l'architecture d'Apache et modification des fichiers de configuration principaux. Mise en place des Virtual Hosts, organisation des répertoires du site web et configuration des fichiers journaux.

3. **Sécurisation**

   Activation du module SSL, génération de certificats **TLS** avec `OpenSSL`, configuration du protocole HTTPS et mise en place de la redirection automatique du trafic HTTP vers HTTPS.

Conception du système

Architecture d'Apache et rôle des fichiers de configuration

Apache adopte une architecture basée sur des fichiers de configuration hiérarchisés, permettant une séparation claire entre la configuration globale, les modules et les sites web.

1. Répertoire principal de configuration /etc/apache2

Sous les systèmes Debian/Ubuntu, l'ensemble de la configuration d'Apache est centralisé dans le répertoire suivant :

```
cd /etc/apache2
ls -l
```

Ce répertoire contient les fichiers et sous-répertoires essentiels au fonctionnement du serveur. Parmi les plus importants, on distingue :

- `apache2.conf`
- `ports.conf`
- `envvars`
- `sites-available/`
- `sites-enabled/`
- `mods-available/`
- `mods-enabled/`
- `conf-available/`
- `conf-enabled/`

2. Fichier apache2.conf : configuration globale

Le fichier `apache2.conf` constitue le **fichier de configuration principal** du serveur Apache.

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

Il définit :

- les paramètres globaux du serveur,
- les règles de base de sécurité,
- les inclusions des autres fichiers de configuration.

Contrairement à une configuration monolithique, Apache utilise des directives `Include` pour charger dynamiquement les configurations complémentaires :

```
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf
IncludeOptional sites-enabled/*.conf
```

Ce mécanisme permet :

- Une configuration modulaire,
- Une meilleure lisibilité,
- Une administration simplifiée.

3. Fichier ports.conf : gestion des ports d'écoute

Le fichier `ports.conf` détermine les ports sur lesquels Apache écoute les requêtes entrantes.

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

Par défaut, on y trouve :

```
Listen 80
```

Après l'activation de TLS, ce fichier contiendra également :

```
Listen 443
```

Ce fichier joue un rôle central dans la configuration réseau du serveur et doit être cohérent avec les hôtes virtuels définis ultérieurement.

4. Répertoires sites-available et sites-enabled

Apache utilise un mécanisme de **sites disponibles / sites activés** pour gérer les hôtes virtuels.

```
ls /etc/apache2/sites-available
ls /etc/apache2/sites-enabled
```

- `sites-available` : contient les fichiers de configuration des sites web
- `sites-enabled` : contient des **liens symboliques** vers les sites actifs

L'activation d'un site se fait via la commande :

```
sudo a2ensite nom_du_site.conf
sudo systemctl reload apache2
```

La désactivation s'effectue avec :

```
sudo a2dissite nom_du_site.conf
sudo systemctl reload apache2
```

Ce mécanisme repose sur l'utilisation de **liens symboliques**, ce qui permet d'activer ou désactiver un site

sans modifier directement les fichiers.

5. Répertoires mods-available et mods-enabled

Apache fonctionne de manière modulaire. Les modules permettent d'étendre les fonctionnalités du serveur (SSL, réécriture d'URL, PHP, etc.).

```
ls /etc/apache2/mods-available
ls /etc/apache2/mods-enabled
```

- `mods-available` : contient tous les modules installés
- `mods-enabled` : contient les modules effectivement chargés

L'activation d'un module se fait par :

```
sudo a2enmod nom_du_module
sudo systemctl restart apache2
```

Exemple :

```
sudo a2enmod ssl
```

6. Répertoire /var/www : racine des sites web

Le contenu des sites web est stocké par défaut dans :

```
ls /var/www
```

Le site par défaut utilise le répertoire :

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

Ce répertoire doit être correctement configuré au niveau :

- Des permissions,
- Du propriétaire,
- Des directives Apache (`DocumentRoot`).

7. Journaux Apache : /var/log/apache2

Les fichiers journaux sont essentiels pour le diagnostic et la sécurité.

```
ls /var/log/apache2
```

Les principaux fichiers sont :

- `access.log` : requêtes reçues
- `error.log` : erreurs du serveur

Consultation en temps réel :

```
tail -f /var/log/apache2/error.log
```

Implémentation et Configuration

1. Installation du serveur web Apache

1.1 Mise à jour du système

Avant toute installation de service, il est recommandé de mettre à jour la liste des paquets afin de garantir l'installation des versions stables et corrigées.

```
sudo apt update
sudo apt upgrade -y
```

Cette opération permet :

- De synchroniser la base de données des paquets,
- D'éviter les conflits de dépendances,
- D'améliorer la stabilité et la sécurité du système.

1.2 Installation d'Apache HTTP Server

L'installation du serveur Apache est effectuée à l'aide du gestionnaire de paquets du système.

```
sudo apt install apache2 -y
```

Cette commande installe :

- Le serveur Apache,
- Les fichiers de configuration par défaut,
- Les scripts de gestion du service,
- L'arborescence standard d'Apache.

1.3 Vérification de l'état du service Apache

Une fois l'installation terminée, il est nécessaire de vérifier que le service Apache est correctement démarré.

```
sudo systemctl status apache2
```

Un service fonctionnel doit apparaître avec l'état :

```
active (running)
```

Si nécessaire, Apache peut être démarré ou redémarré manuellement :

```
sudo systemctl start apache2
sudo systemctl restart apache2
```

1.4 Vérification du fonctionnement via le navigateur

Apache installe par défaut une page web de test. Pour vérifier le bon fonctionnement du serveur, il suffit d'accéder à l'adresse suivante depuis un navigateur :

`http://localhost`

ou, depuis une autre machine :

`http://<adresse_IP_du_serveur>`

L'apparition de la page **"Apache2 Ubuntu Default Page"** confirme que :

- le serveur est opérationnel,
- Apache écoute correctement sur le port HTTP,
- la communication réseau est fonctionnelle.

1.5 Ports d'écoute par défaut

Par défaut, Apache écoute sur le port **80 (HTTP)**. Cette configuration est définie dans le fichier :

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

Ce point sera étudié en détail dans les étapes suivantes, notamment lors de la configuration avancée et de la mise en place du protocole HTTPS (TLS).

2. Création et configuration d'un Virtual Host

L'objectif est de configurer un **hôte virtuel** (Virtual Host) pour permettre à Apache de servir un site web spécifique. Cette configuration permet de :

- Séparer plusieurs sites sur un même serveur,
- Définir des logs et répertoires propres à chaque site,
- Faciliter la gestion et la sécurisation des sites web.

2.1 Création du répertoire du site

Chaque site web doit disposer d'un **répertoire dédié** dans lequel seront placés ses fichiers HTML, CSS, JS, etc. Par exemple, pour un site nommé monprojet :

```
sudo mkdir -p /var/www/monprojet
sudo chown -R $USER:$USER /var/www/monprojet
sudo chmod -R 755 /var/www/monprojet
```

- `mkdir -p` : crée le répertoire et tous ses parents si nécessaire
- `chown` : change le propriétaire pour l'utilisateur actuel
- `chmod 755` : donne les permissions nécessaires pour que le serveur puisse lire les fichiers

Création d'une page test :

```
nano /var/www/monprojet/index.html
```

Contenu exemple :

```
<!DOCTYPE html>
<html>
<head>
    <title>Mon Projet Apache</title>
</head>
<body>
    <h1>Site en fonctionnement</h1>
</body>
</html>
```

2.2 Création du fichier de configuration du Virtual Host

Dans `/etc/apache2/sites-available/`,créer un fichier `monprojet.conf` :

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

Contenu type pour HTTP :

```
<VirtualHost *:80>
    ServerAdmin admin@monprojet.com
    ServerName monprojet.local
    DocumentRoot /var/www/monprojet
    ErrorLog ${APACHE_LOG_DIR}/monprojet_error.log
    CustomLog ${APACHE_LOG_DIR}/monprojet_access.log combined
</VirtualHost>
```

Explications :

- `ServerAdmin` : email de l'administrateur du site
- `ServerName` : nom de domaine ou nom local du site
- `DocumentRoot` : chemin vers les fichiers du site

- `ErrorLog et CustomLog` : logs dédiés au site
- `<VirtualHost *:80>` : écoute sur le port HTTP par défaut

2.3 Activation du Virtual Host

Pour activer le site, utiliser les commandes :

```
sudo a2ensite monprojet.conf
sudo systemctl reload apache2
```

- `a2ensite` : crée un **lien symbolique** dans `sites-enabled`
- `systemctl reload` : applique la nouvelle configuration sans interrompre le service

2.4 Modification du fichier hosts (test local)

Pour tester le site en local, modifier `/etc/hosts` pour mapper le nom du site à l'adresse locale :

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

Ajouter la ligne :

```
127.0.0.1 monprojet.local
```

Ensuite, accéder à :

`http://monprojet.local`

Si la page s'affiche correctement, le Virtual Host fonctionne.

2.5 Désactivation ou suppression d'un site

Pour désactiver un site :

```
sudo a2dissite monprojet.conf
sudo systemctl reload apache2
```

Pour supprimer un site définitivement, supprimer le fichier et le répertoire associé :

```
sudo rm /etc/apache2/sites-available/monprojet.conf
sudo rm -r /var/www/monprojet
```

3. Sécurisation du serveur avec TLS (HTTPS)

L'objectif est de **chiffrer les communications** entre le serveur Apache et les clients, afin d'assurer :

- La confidentialité des données,
- L'intégrité des informations échangées,
- L'authenticité du serveur via **un certificat TLS**.

Cette étape repose sur le module SSL d'Apache et l'utilisation de certificats auto-signés ou fournis par une autorité de certification (CA).

3.1 Activation du module SSL d'Apache

Apache utilise des modules pour ajouter des fonctionnalités.

Le module SSL doit être activé :

```
sudo a2enmod ssl
sudo systemctl restart apache2
```

Vérification que le module est actif :

```
apache2ctl -M | grep ssl
```

On doit obtenir :

```
ssl_module (shared)
```

3.2 Création d'un certificat auto-signé

Pour un environnement de test ou local, on peut générer un certificat auto-signé. Les fichiers seront stockés dans `/etc/ssl/.`

```
sudo mkdir -p /etc/ssl/monprojet
cd /etc/ssl/monprojet
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout monprojet.key -out monprojet.crt
```

Explications :

- `-x509` : type de certificat auto-signé
- `-nodes` : ne pas chiffrer la clé privée (facilite Apache)
- `-days 365` : validité du certificat (1 an)

- `-newkey rsa:2048` : **clé RSA** de **2048 bits**
- `-keyout et -out` : chemins du fichier clé et du certificat

Lors de la commande, `OpenSSL` demande des informations pour le certificat : pays, ville, organisation, nom du serveur (`ServerName`), etc.

3.3 Configuration du Virtual Host pour HTTPS

Créer un nouveau fichier dans `/etc/apache2/sites-available/` : `monprojet-ssl.conf`

```
sudo nano /etc/apache2/sites-available/monprojet-ssl.conf
```

Contenu type :

```
<VirtualHost *:443>
    ServerAdmin admin@monprojet.com
    ServerName monprojet.local
    DocumentRoot /var/www/monprojet

    ErrorLog ${APACHE_LOG_DIR}/monprojet_error.log
    CustomLog ${APACHE_LOG_DIR}/monprojet_access.log combined

    SSLEngine on
    SSLCertificateFile /etc/ssl/monprojet/monprojet.crt
    SSLCertificateKeyFile /etc/ssl/monprojet/monprojet.key
</VirtualHost>
```

Explications :

- `SSLEngine on` : active le chiffrement **SSL/TLS**
- `SSLCertificateFile` : chemin vers le certificat public
- `SSLCertificateKeyFile` : chemin vers la clé privée

3.4 Activation du Virtual Host HTTPS

```
sudo a2ensite monprojet-ssl.conf
sudo systemctl reload apache2
```

Vérification :

```
sudo apache2ctl configtest
```

On doit obtenir :

```
Syntax OK
```

Ensuite, tester l'accès via le navigateur :

`https://monprojet.local`

Remarque : le navigateur peut signaler un avertissement pour certificat auto-signé, ce qui est normal en environnement de test.

3.5 Redirection HTTP → HTTPS (optionnelle mais recommandée)

Pour forcer le trafic sécurisé, modifier le fichier HTTP du Virtual Host (`monprojet.conf`) :

```
<VirtualHost *:80>
    ServerName monprojet.local
    Redirect permanent / https://monprojet.local/
</VirtualHost>
```

Puis recharger Apache :

```
sudo systemctl reload apache2
```

Ainsi, toute requête HTTP est automatiquement redirigée vers HTTPS.

4. Sécurisation finale et bonnes pratiques

Cette étape vise à **renforcer la sécurité** du serveur Apache, à optimiser les permissions et à garantir une configuration robuste pour un environnement de production. Elle inclut la gestion des droits sur les fichiers, la configuration des journaux, et l'optimisation des paramètres TLS.

4.1 Gestion des permissions et propriétaires

Pour qu'Apache fonctionne correctement tout en respectant les règles de sécurité, il est essentiel de vérifier les permissions des fichiers et répertoires :

```
sudo chown -R www-data:www-data /var/www/monprojet
sudo chmod -R 755 /var/www/monprojet
```

Explications :

- `www-data` : utilisateur et groupe sous lequel Apache s'exécute
- `chown` : assure que le serveur a la propriété des fichiers
- `chmod 755` : lecture et exécution pour tous, écriture pour le propriétaire uniquement

Cette configuration limite les risques liés à la modification non autorisée des fichiers web.

4.2 Sécurisation des fichiers de configuration

Les fichiers de configuration d'Apache doivent également être protégés :

```
sudo chmod 644 /etc/apache2/sites-available/*.conf
sudo chmod 600 /etc/ssl/monprojet/*.key
```

- **644** : lecture pour tous, écriture uniquement pour le propriétaire
- **600** : lecture/écriture uniquement pour le propriétaire (clé privée TLS)

4.3 Configuration des journaux

Apache permet de configurer des logs distincts pour chaque site. Pour renforcer la sécurité et la traçabilité :

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

Vérifier que :

```
ErrorLog ${APACHE_LOG_DIR}/monprojet_error.log
CustomLog ${APACHE_LOG_DIR}/monprojet_access.log combined
```

Bonnes pratiques :

- Rotation régulière des logs (`logrotate`)
- Permissions restreintes sur les fichiers de **logs**

4.4 Optimisation TLS

Pour un environnement de production, il est recommandé d'appliquer des paramètres TLS sécurisés :

1. Désactiver les anciens protocoles (**SSLv2, SSLv3, TLS 1.0, TLS 1.1)**
2. Activer uniquement **TLS 1.2** et **1.3**
3. Forcer des suites de chiffrement robustes

Exemple dans le Virtual Host SSL :

```
SSLProtocol TLSv1.2 TLSv1.3
SSLCipherSuite HIGH:!aNULL:!MD5
SSLHonorCipherOrder on
```

Puis recharger Apache :

```
sudo systemctl reload apache2
```

Tests et Résultats

1. Tests réalisés

Plusieurs tests ont été effectués afin de vérifier le bon fonctionnement du serveur Apache après chaque phase de configuration.

- **Test du service Apache**

  Le service Apache a été vérifié afin de s'assurer qu'il est actif et fonctionnel :

```
sudo systemctl status apache2
```

Le service est correctement démarré et fonctionne sans erreur.

- **Test d'accès HTTP**

  L'accès au serveur via le protocole HTTP a été testé à l'aide d'un navigateur web :

```
http://localhost
```

Le site configuré s'affiche correctement, confirmant le bon fonctionnement du Virtual Host.

- **Test de configuration Apache**

  La validité de la configuration a été vérifiée après chaque modification :

```
sudo apache2ctl configtest
```

Le résultat obtenu est :

```
Syntax OK
```

- **Test d'accès HTTPS (TLS)**

  Le site a été testé via le protocole HTTPS :

```
https://monprojet.local
```

Le chiffrement TLS est fonctionnel. Un avertissement du navigateur peut apparaître en raison de l'utilisation d'un certificat auto-signé, ce qui est normal en environnement de test.

- **Test des ports d'écoute**

  La disponibilité des ports HTTP et HTTPS a été vérifiée :

```
sudo ss -tulnp | grep apache2
```

Les ports 80 et 443 sont bien ouverts et utilisés par Apache.

2. Résultats obtenus

Les résultats obtenus à l'issue des tests montrent que :

- Le serveur Apache est correctement installé et opérationnel ;
- Les Virtual Hosts sont fonctionnels et permettent l'hébergement du site web ;
- Les permissions et les propriétaires des fichiers sont correctement configurés ;
- la communication HTTPS est active et sécurisée à l'aide du protocole TLS ;
- la redirection HTTP vers HTTPS fonctionne conformément à la configuration définie.

Analyse critique

Lors de la mise en place et de la configuration du serveur Apache, plusieurs difficultés ont été rencontrées :

1. **Gestion des permissions et propriétaires** : Apache ne pouvait pas lire certains fichiers ou répertoires à cause de droits mal configurés. La correction a nécessité l'attribution du propriétaire **www-data** et des permissions adaptées (**755** pour les répertoires, **644** pour les fichiers, **600** pour les clés **TLS**).
2. **Configuration des Virtual Hosts** : Des erreurs dans le `ServerName`, les liens symboliques ou les ports provoquaient des conflits et empêchaient certains sites de fonctionner. La solution a été de vérifier systématiquement les chemins et de recharger Apache après chaque modification.
3. **Activation du SSL et gestion des certificats** : La génération de certificats auto-signés et leur configuration ont causé des avertissements et des erreurs de chemin. Ces problèmes ont été résolus en respectant strictement la structure des fichiers et en testant chaque étape.
4. **Redirection HTTP → HTTPS** : Une mauvaise configuration entraînait parfois des boucles de redirection. La solution a été de séparer clairement les Virtual Hosts HTTP et HTTPS et de valider la configuration avec `apache2ctl configtest`.

Ces difficultés ont permis de renforcer la maîtrise des aspects essentiels de l'administration d'Apache et des bonnes pratiques de sécurisation d'un serveur web.

Conclusion

En suivant ces étapes, le projet couvre l'ensemble du **cycle de vie d'un serveur web Apache** :

1. **Installation et mise en service initiale,**
2. **Compréhension de l'architecture des fichiers et modules,**
3. **Configuration des Virtual Hosts pour l'hébergement de sites,**
4. **Sécurisation via TLS/HTTPS,**
5. **Renforcement de la sécurité et bonnes pratiques de maintenance.**

Bibliographie

1. **Apache Software Foundation, *Apache HTTP Server Documentation*.**
   Disponible sur : https://httpd.apache.org/docs/
2. **Ubuntu Documentation, *Apache HTTP Server Guide*.**
   Disponible sur : https://help.ubuntu.com/
3. **Linux Manual Pages, *apache2*, *systemctl*, *chmod*, *chown*, *openssl*.**
   Disponible via la commande `man` sous Linux.
4. **OpenSSL Project, *OpenSSL Documentation*.**
   Disponible sur : https://www.openssl.org/docs/
5. **Nemeth, E., Snyder, G., Hein, T. R., & Whaley, B., *UNIX and Linux System Administration Handbook*, Prentice Hall.**