ESGIS
ECOLE SUPERIEURE DE GESTION D'INFORMATIQUE ET DES SCIENCES

Décembre 2025

Administration des Serveurs Web

Grace Fanta KOUNOU
Ingénieure en Informatique

Objectifs d'Apprentissage

✓ Déployer un serveur web Apache et Nginx

✓ Configurer des Virtual Hosts multiples

✓ Sécuriser avec SSL/TLS

✓ Mettre en place une architecture Apache+Nginx

✓ Monitorer et diagnostiquer les problèmes

✓ Administrer un serveur en production

Plan du cours

01
Fondation du web

02
Serveur Web Apache2

03
Serveur Web Nginx

04
Comparaison & Cas d'Usage

05
Sécurité des Serveurs Web

06
Monitoring, Logs & Dépannage

07
Travaux Pratiques

Fondation du Web

- Architecture Client-Serveur
- Protocole HTTP
- Protocole HTTPS
- Concepts Importants

Architecture Client-Serveur

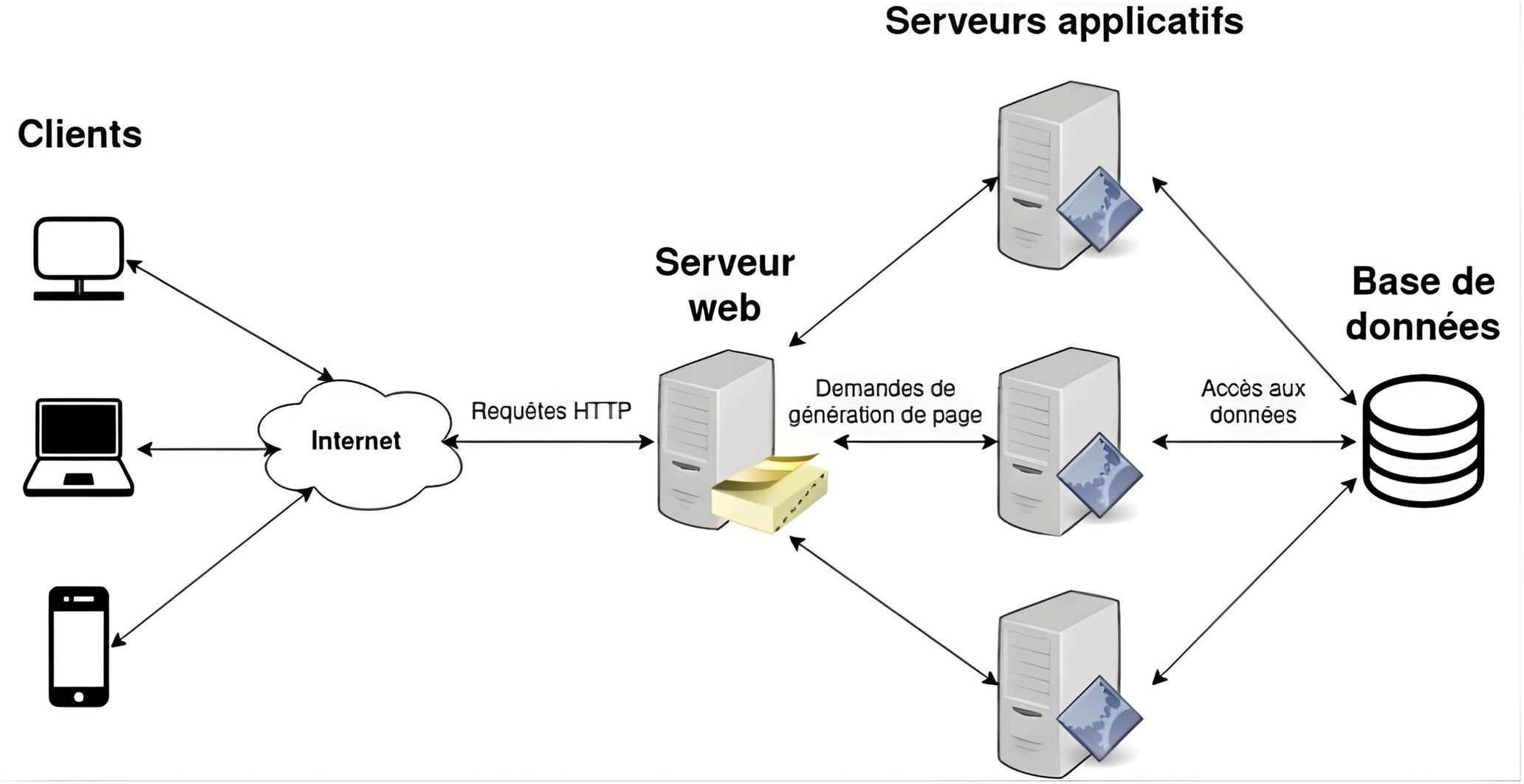
L'architecture client-serveur est le modèle de communication dominant sur Internet.

Elle repose sur deux rôles principaux :

Le Client : Une application (généralement un navigateur web tel que Chrome, Firefox, Safari) qui envoie une requête pour obtenir une ressource.

Le Serveur : Une machine physique ou virtuelle qui exécute un logiciel serveur (ex: web, base de données) et qui attend les requêtes.

Après traitement, le serveur renvoie une réponse (contenant la ressource demandée, comme une page HTML, une image, etc.) au client.


Serveurs applicatifs
Clients
Serveur web
Base de données
Internet
Requêtes HTTP
Demandes de génération de page
Accès aux données

Protocole HTTP

L'Hypertext Transfer Protocol (Protocole de Transfert Hypertexte), généralement abrégé HTTP, est un protocole standard pour la communication entre client et serveur.

Port par défaut : 80

Méthodes principales :

- **GET** : Récupérer une ressource
- **POST** : Envoyer des données au serveur
- **PUT** : Mettre à jour une ressource
- **DELETE** : Supprimer une ressource
- **HEAD** : Récupérer uniquement les en-têtes
- **OPTIONS** : Obtenir les méthodes supportées

Structure HTTP

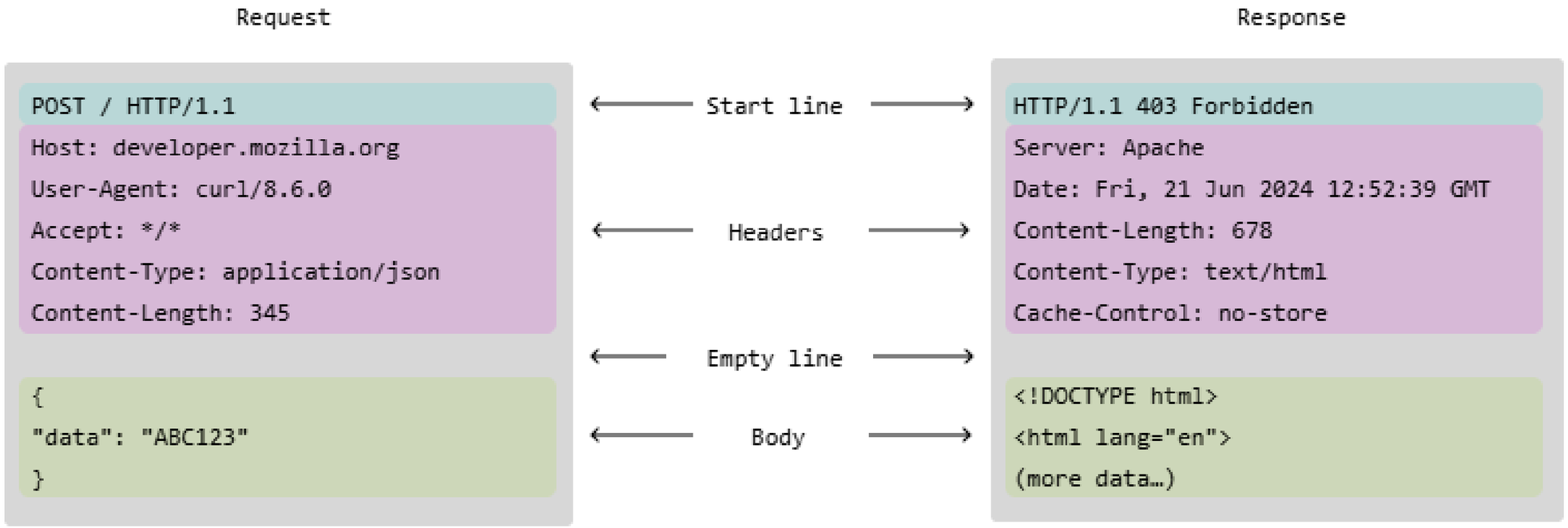
Toute communication HTTP se déroule en deux parties : la r**equête** envoyée par le client et la r**éponse** envoyée par le serveur.

Requête HTTP	Réponse HTTP
GET /index.html HTTP/1.1 Host: www.example.com User-Agent: Mozilla/5.0 Accept: text/html Accept-Language: fr-FR Connection: keep-alive	HTTP/1.1 200 OK Date: Tue, 21 Oct 2025 10:00:00 GMT Server: Apache/2.4.41 Content-Type: text/html; charset=UTF-8 Content-Length: 1234 <!DOCTYPE html> <html>...

Code de statut HTTP

Le code de statut est un nombre à trois chiffres dans la ligne de statut de la réponse. Il indique l'état du traitement de la requête.

Code	Catégorie	Signification
200	Succès	OK
301	Redirection	Moved Permanently
302	Redirection	Found
304	Redirection	Not Modified
400	Erreur Client	Bad Request
401	Erreur Client	Unauthorized
403	Erreur Client	Forbidden
404	Erreur Client	Not Found
500	Erreur Serveur	Internal Server Error
502	Erreur Serveur	Bad Gateway
503	Erreur Serveur	Service Unavailable


Request
Response
POST / HTTP/1.1
Host: developer.mozilla.org
User-Agent: curl/8.6.0
Accept: */*
Content-Type: application/json
Content-Length: 345
{
"data": "ABC123"
}
Start line
Headers
Empty line
Body
HTTP/1.1 403 Forbidden
Server: Apache
Date: Fri, 21 Jun 2024 12:52:39 GMT
Content-Length: 678
Content-Type: text/html
Cache-Control: no-store
<!DOCTYPE html>
<html lang="en">
(more data…)

Protocole HTTPS

Le protocole HTTPS (Hypertext Transfer Protocol Secure) est l'extension sécurisée de HTTP. Il utilise le protocole SSL/TLS (Secure Sockets Layer / Transport Layer Security) pour chiffrer la communication.

Avantages :

Confidentialité : Personne ne peut lire les données
Intégrité : Les données ne peuvent être modifiées sans détection
Authentification : Garantit que vous parlez au bon serveur

Caractéristiques :

Port par défaut : 443
Mise en œuvre : Nécessite un certificat SSL/TLS (souvent fourni par des autorités comme Let's Encrypt)


HTTP://
!
HTTPS://

Concepts Importants

- **Cookies** : Petits fichiers stockés côté client pour maintenir l'état entre les requêtes.

- **Sessions** : Ensemble de données spécifiques à un utilisateur (variables, état de connexion, contenu du panier) stocké temporairement sur le serveur.

- **Cache** : Stockage temporaire et rapide de données fréquemment demandées.

Serveur Web Apache2

- Historique & Philosophie
- Architecture & Fonctionnement
- Configuration & Arborescence
- Avantages & Inconvénients

Historique & Philosophie

Historique

Lancé en 1995, Apache HTTP Server est le serveur web le plus ancien et a dominé le marché pendant des décennies.
Il est maintenu par l'Apache Software Foundation (ASF).

Philosophie

Être un serveur robuste, modulaire et complet (feature-rich).
Sa conception est basée sur l'idée que presque toutes les fonctionnalités peuvent être ajoutées via des modules chargés dynamiquement.

Architecture & Fonctionnement

MPM Prefork

- Crée un processus enfant par connexion
- Très stable et compatible (idéal pour modules non thread-safe)
- Gourmand en mémoire
- Pas de parallélisme au niveau des threads

MPM Worker

- Utilise plusieurs processus enfants, chacun gérant plusieurs threads
- Plus efficace que Prefork en termes de mémoire
- Meilleur pour les charges élevées

MPM Event (moderne)

- Semblable à Worker mais optimise la gestion des connexions Keep-Alive
- Dédié un thread par connexion active seulement
- Recommandé pour la plupart des cas modernes

Configuration & Arborescence

L'architecture d'Apache2 est basée sur un fichier de configuration principal et un système très modulaire et distribué.

Eléments	Chemins
Paramètres fondamentaux du serveur	/etc/apache2/apache2.conf
Fichiers pour chaque site web	/etc/apache2/sites-available/
Fichiers pour chaque site web activé	/etc/apache2/sites-enabled/
Fichiers de configuration des modules	/etc/apache2/mods-available/ & /etc/apache2/mods-enabled/
Emplacement des fichiers web	/var/www/html

Avantages & Inconvénients

Des avantages et inconvénients de Apache :

Points Forts (Avantages)	Points Faibles (Inconvénients)
• Robustesse et Stabilité prouvée • Extrême Modularité (via les modules) • Configuration flexible (via .htaccess) • Très large communauté et documentation	• Moins performant pour servir des fichiers statiques et gérer beaucoup de connexions simultanées • Consommation de mémoire plus élevée, surtout avec MPM Prefork. • La flexibilité des .htaccess peut ralentir les performances.

Serveur Web Nginx

- Historique & Philosophie
- Architecture & Fonctionnement
- Configuration & Arborescence
- Avantages & Inconvénients

Historique & Philosophie

Historique

Créé en 2004 par Igor Sysoev en Russie pour résoudre le problème C10k (gérer 10 000 connexions concurrentes).
Il a rapidement gagné en popularité pour sa performance et sa légèreté.

Philosophie

Être rapide, léger et hautement concurrent en utilisant une architecture événementielle (event-driven).

Architecture & Fonctionnement

Master Process

- Il y a un seul processus maître (master process) qui gère les permissions et la configuration.

Worker Processes

- Plusieurs processus worker (workerprocesses) gèrent les connexions. Chaque worker peut gérer des milliers de connexions simultanées en utilisant une boucle d'événements unique (event loop).

Fonctionnement Non-Bloquant

- Lorsqu'une requête arrive, le worker ne reste pas bloqué à l'attendre (contrairement aux architectures basées sur les processus/threads par connexion) mais passe à l'événement suivant, gérant l'entrée/sortie de manière efficace.

Configuration & Arborescence

L'architecture de Nginx est plus centralisée et hiérarchique. Elle est optimisée pour la vitesse de lecture et la performance.

Eléments	Chemins
Paramètres fondamentaux du serveur	/etc/nginx/nginx.conf
Fichiers pour chaque site web	/etc/nginx/sites-available/
Fichiers pour chaque site web activé	/etc/nginx/sites-enabled/
Emplacement des fichiers web	/var/www/html

Avantages & Inconvénients

Des avantages et inconvénients de Nginx :

Points Forts (Avantages)	Points Faibles (Inconvénients)
• Performance supérieure pour le contenu statique • Faible consommation de mémoire (léger) • Excellent pour servir comme Reverse Proxy et Load Balancer • Architecture permettant une haute concurrence	• Moins de possibilités pour ajouter des fonctionnalités non-standard • La gestion de la configuration est plus centralisée ; pas de fonctionnalité .htaccess • La courbe d'apprentissage de la configuration légèrement plus raide pour les débutants.

Comparaison et Cas d'usage

- Quand utiliser Apache ?
- Quand utiliser Nginx ?
- Architecture de synergie

Quand utiliser Apache ?

À privilégier lorsque :

- Vous déployez un serveur monolithique unique utilisant PHP via mod_php.
- Vos sites web nécessitent une configuration dynamique, notamment l'usage de fichiers .htaccess.
- Vous travaillez dans des environnements Java utilisant le protocole AJP pour la communication avec les applications.

Quand utiliser Nginx ?

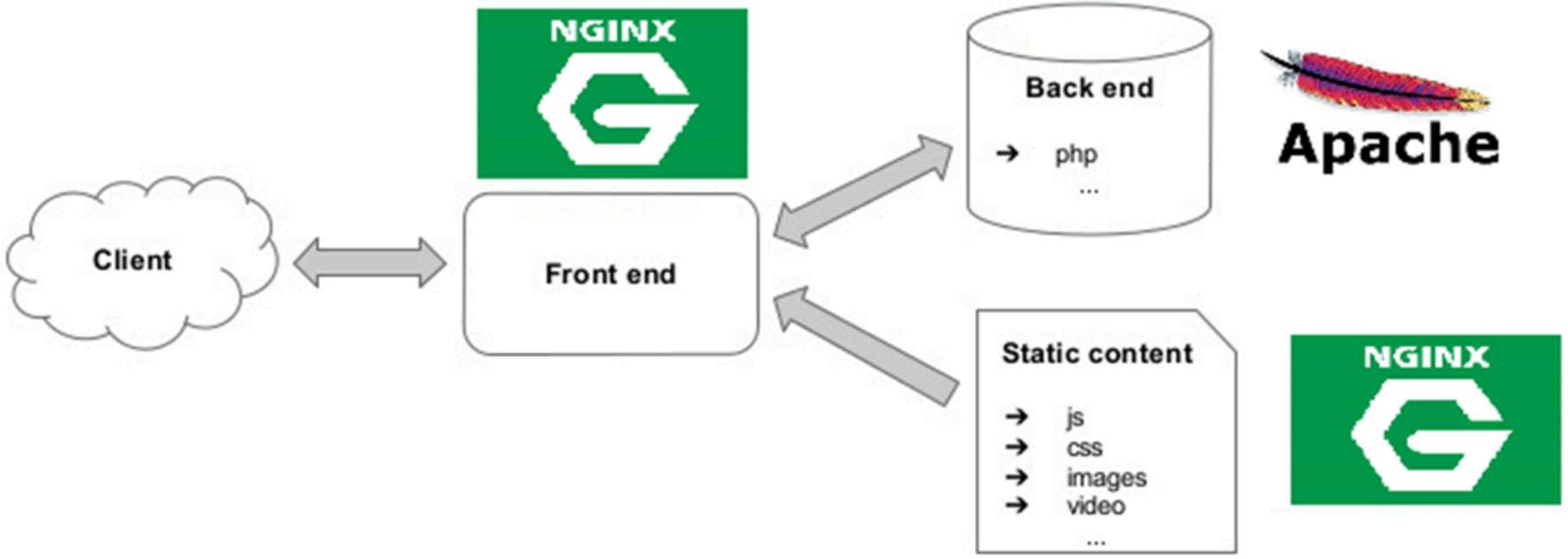
À privilégier lorsque :

- Vous l'utilisez comme reverse proxy, notamment en frontal d'Apache ou d'un serveur d'application.
- Vous avez besoin d'un load balancer performant pour distribuer la charge.
- Vous devez servir efficacement des fichiers statiques (images, CSS, JavaScript).
- Vous gérez des API à forte concurrence, nécessitant rapidité et faible latence.

Architecture de synergie

Cette combinaison permet de tirer parti des atouts de chaque serveur : rapidité et légèreté de Nginx en frontal, puissance d'interprétation d'Apache en backend.

- Nginx en reverse proxy en frontal pour :
  - Gérer efficacement les connexions rapides et simultanées.
  - Servir directement le contenu statique (images, CSS, JS) avec des performances optimisées.
- Apache2 en backend pour :
  - Traiter les requêtes dynamiques nécessitant une logique applicative (PHP, etc.).


NGINX
Client
Front end
Back end
→ php
...
Apache
Static content
→ js
→ css
→ images
→ video
...
NGINX

Sécurité des Serveurs Web

- Pourquoi la sécurité est cruciale ?
- Architecture & Fonctionnement
- Configuration & Arborescence
- Avantages & Inconvénients

Pourquoi la Sécurité ?

Protection des Données Sensibles

Informations personnelles, paiements, secrets d'entreprise

Continuité de Service

Éviter interruptions, attaques DDoS, ransomwares

Conformité Réglementaire

RGPD, PCI-DSS, ISO 27001, SOC 2

Confiance des Utilisateurs

84% abandonnent si le site n'est pas sécurisé

Menaces principales

Attaques par Injection

- SQL Injection : Accès base de données via formulaires
- XSS (Cross-Site Scripting) : Scripts malveillants injectés

Attaques par Déni de Service

- DDoS : Saturation du serveur
- HTTP Flood : Requêtes massives

Exploitation de Vulnérabilités

- Zero-day : Failles inconnues
- Mauvaises configurations : Permissions, ports ouverts

Malware et Backdoors

- Webshells : Contrôle à distance
- Ransomware : Chiffrement contre rançon

Principes Fondamentaux

Défense en Profondeur (Defense in Depth)

Multiples couches de sécurité

Principe du Moindre Privilège

Permissions minimales nécessaires seulement

Sécurité par Défaut (Secure by Default)

Configuration sécurisée dès l'installation

Mise à Jour Continue

Patches de sécurité appliqués rapidement

Chiffrement Systématique

HTTPS/TLS pour toutes les communications

Culture de la Sécurité

Pour les administrateurs

- Documenter toutes les configurations
- Utiliser des mots de passe forts et uniques
- Activer l'authentification multi-facteurs (2FA)
- Ne jamais stocker de mots de passe en clair

Pour les développeurs

- Valider et filtrer toutes les entrées utilisateur
- Utiliser des requêtes préparées (prepared statements)
- Échapper les sorties pour éviter le XSS
- Suivre les principes OWASP

Pour l'Organisation

- Plans de réponse aux incidents
- Sauvegardes régulières et testées
- Formation continue du personnel
- Audits de sécurité périodiques

Monitoring, Log et Dépannage

- Pourquoi le monitoring ?
- Niveaux de monitoring
- Types de logs essentiels
- Dépannage (Troubleshooting)

Pourquoi le monitoring ?

Détection Proactive des Problèmes

Identifier avant impact utilisateurs

Compréhension du Comportement du Système

Analyser performance, charge, tendances

Optimisation des Ressources

Dimensionnement optimal des serveurs

Sécurité et Détection d'Intrusion

Repérer les attaques en temps réel

Conformité et Audit

Traçabilité requise par les réglementations

Types de Logs

Access Logs (Journaux d'accès)

Qui a accédé ? À quoi ? Quand ? Résultat ?
→ Analyse trafic, pages populaires, détection bots

Error Logs (Journaux d'erreurs)

Qu'est-ce qui a mal fonctionné ? Pourquoi ?
→ Diagnostic bugs, erreurs de config

Security Logs (Journaux de sécurité)

Y a-t-il eu des tentatives d'intrusion ?
→ Scans vulnérabilités, brute force, accès non autorisés

Application Logs (Journaux applicatifs)

Que fait l'application en interne ?
→ Debug métier, traçabilité transactions

Dépannages fréquents

Quelques problèmes courants et les solutions recommandées pour les résoudre efficacement :

Problèmes courants	Solutions
Le serveur ne démarre pas	Vérifier syntaxe config, ports utilisés, logs
Erreur 403 Forbidden	Permissions fichiers, directives Require/deny
Erreur 404 Not Found	Vérifier DocumentRoot/root, fichier existe
Erreur 502 Bad Gateway (Nginx)	Backend accessible ? proxy_pass correct ?

Travaux Pratiques