UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

APACHE PERFECTIONNEMENT

Sommaire

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Apache HTTPD 2.4 : rappels et nouveautés

Apache HTTPD 2.4 : rappels et nouveautés

Rappels :

- **Apache HTTP Server (HTTPD)** est l'un des serveurs web les plus populaires.
- Il repose sur une architecture modulaire.
- Supporte des fonctionnalités comme :
  - **Proxying** (mod_proxy, mod_proxy_http, mod_proxy_ajp, etc.)
  - **SSL/TLS** (via mod_ssl).
  - **Virtual hosts** (configuration de plusieurs sites sur une seule machine).
- Version 2.4 apporte des améliorations par rapport à 2.2 en termes de performance, sécurité et flexibilité.

Apache HTTPD 2.4 : rappels et nouveautés

Nouveautés d'Apache HTTPD 2.4 (comparé à 2.2) :

MPM améliorés :

- Nouveau MPM **Event**, optimisé pour les connexions persistantes et HTTP/2.
- Amélioration des MPM Worker et Prefork.
- **Authentification unifiée** :
  - Nouvelle architecture avec `mod_authz_core`, permettant des directives comme `Require` pour centraliser la gestion d'accès.
  - Authentification via base de données avec `mod_authn_dbd`.
- **Configuration conditionnelle** :
  - Support des directives `If`, `ElseIf`, `Else` pour une configuration plus dynamique.
  - Exemple :

```
<If "%{HTTP_HOST} == 'example.com'">
  DocumentRoot "/var/www/example"
</If>
```

Apache HTTPD 2.4 : rappels et nouveautés

Nouveautés d'Apache HTTPD 2.4 (comparé à 2.2) :

- **Modules proxy améliorés** :
  - Support natif pour WebSocket via `mod_proxy_wstunnel`.
  - Meilleure gestion des backends via `mod_proxy_balancer`.
- **HTTP/2** :
  - Ajout du module `mod_http2` pour le protocole HTTP/2.
- **Amélioration des performances** :
  - Réduction de la consommation mémoire grâce à des optimisations dans le traitement des connexions.
- **Directive IncludeOptional** :
  - Permet d'inclure des fichiers de configuration optionnels, évitant les erreurs si un fichier est manquant.

Apache HTTPD 2.4 : rappels et nouveautés

Compilation, Installation et Test Initial :

1. Pré-requis :

- Vérifier les dépendances :
  - **APR et APR-util** (Apache Portable Runtime).
  - OpenSSL pour SSL/TLS.
  - PCRE (Perl Compatible Regular Expressions) pour `mod_rewrite` et autres modules.

Apache HTTPD 2.4 : rappels et nouveautés

Compilation, Installation et Test Initial :

2. Télécharger les sources :

```
wget https://downloads.apache.org/httpd/httpd-2.4.x.tar.gz
```

3. Télécharger les dépendances APR :

```
wget https://downloads.apache.org/apr/apr-1.x.tar.gz
wget https://downloads.apache.org/apr/apr-util-1.x.tar.gz
```

Décompressez APR et APR-util, puis placez-les dans le répertoire des sources d'Apache (sous `srclib/`) :

```
tar -xzvf apr-1.x.tar.gz
tar -xzvf apr-util-1.x.tar.gz
mv apr-1.x httpd-2.4.x/srclib/apr
mv apr-util-1.x httpd-2.4.x/srclib/apr-util
```

Apache HTTPD 2.4 : rappels et nouveautés

Compilation, Installation et Test Initial :

4. Décompression des sources Apache HTTPD :

```
tar -xzvf httpd-2.4.x.tar.gz
cd httpd-2.4.x
```

Apache HTTPD 2.4 : rappels et nouveautés

Compilation, Installation et Test Initial :

5. Configuration des options de compilation :

```
./configure --enable-so --enable-ssl --enable-mods-shared=all --with-mpm=event --with-pcre --enable-rewrite
```

- `--enable-so` : Activer les modules dynamiques.
- `--enable-ssl` : Inclure le support SSL/TLS.
- `--enable-mods-shared=all` : Compiler tous les modules sous forme dynamique.
- `--with-mpm=event` : Choisir le MPM Event comme module de traitement par défaut.

Apache HTTPD 2.4 : rappels et nouveautés

Compilation, Installation et Test Initial :

6. Compilation et installation :

```
make
sudo make install
```

Apache HTTPD 2.4 : rappels et nouveautés

Compilation, Installation et Test Initial :

7. Test initial :

1. Vérifiez la version installée :

```
/usr/local/apache2/bin/httpd -v
```

2. Démarrez le serveur Apache :

```
/usr/local/apache2/bin/apachectl start
```

3. Vérifiez le processus Apache :

```
ps aux | grep httpd
```

4. Testez dans un navigateur :
   - Accédez à `http://localhost`.
   - Si tout est correctement configuré, la page par défaut d'Apache apparaîtra.

Apache HTTPD 2.4 : rappels et nouveautés

Structure par défaut de l'installation :

- **Fichiers de configuration** :
  - `/usr/local/apache2/conf/httpd.conf` : fichier principal.
  - `/usr/local/apache2/conf/extra/` : configurations supplémentaires (SSL, virtual hosts, MPM).
- **Répertoires** :
  - `/usr/local/apache2/htdocs/` : emplacement des fichiers web par défaut.
  - `/usr/local/apache2/logs/` : logs d'erreurs et d'accès.

Apache HTTPD 2.4 : rappels et nouveautés

Installation complète d'Apache avec apt

1. Mettre à jour les paquets

Avant toute installation, mettez à jour les informations des paquets pour garantir que vous obtenez la dernière version disponible :

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

2. Installer Apache

Installez Apache HTTP Server avec la commande suivante :

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

Apache HTTPD 2.4 : rappels et nouveautés

Installation complète d'Apache avec apt

3. Vérifier l'installation

Vérifiez qu'Apache est bien installé et en cours d'exécution :

```
sudo systemctl status apache2
```

Vous devriez voir une sortie indiquant que le service est `active (running)`.

Apache HTTPD 2.4 : rappels et nouveautés

Installation complète d'Apache avec apt

4. Tester Apache

1. **Dans un navigateur** : Accédez à votre serveur à l'adresse suivante :
   - Si c'est en local : http://localhost ou http://127.0.0.1.
   - Si c'est une machine distante : http://<adresse IP>.
2. Vous devriez voir une page intitulée **"Apache2 Ubuntu Default Page"**.

Apache HTTPD 2.4 : rappels et nouveautés

Installation complète d'Apache avec apt

5. Configurer le pare-feu

Si vous utilisez un pare-feu (comme UFW), ouvrez le port 80 (HTTP) et 443 (HTTPS) pour permettre l'accès au serveur :

```
sudo ufw allow 'Apache Full'
sudo ufw enable
sudo ufw status
```

Apache HTTPD 2.4 : rappels et nouveautés

Installation complète d'Apache avec apt

6. Activer les modules nécessaires

Apache inclut plusieurs modules utiles. Voici les commandes pour activer les modules les plus courants :

1. **Module SSL** (pour HTTPS) :

```
sudo a2enmod ssl
```

2. **Module Rewrite** (pour les réécritures d'URL comme dans WordPress) :

```
sudo a2enmod rewrite
```

3. Redémarrez Apache pour appliquer les changements :

```
sudo systemctl restart apache2
```

Apache HTTPD 2.4 : rappels et nouveautés

Installation complète d'Apache avec apt

Outils et emplacements importants

- **Fichiers de configuration des sites** :
  - **Disponibles** : `/etc/apache2/sites-available/`
  - **Activés** : `/etc/apache2/sites-enabled/`
- **Modules disponibles** : `/etc/apache2/mods-available/`
- **Répertoire des journaux** : `/var/log/apache2/`

Apache HTTPD 2.4 : rappels et nouveautés

Installation complète d'Apache avec apt

Résumé des commandes principales

- **Installer Apache** : `sudo apt install apache2`
- **Activer un module** : `sudo a2enmod <module>`
- **Activer un site** : `sudo a2ensite <site>`
- **Vérifier la configuration** : `sudo apache2ctl configtest`
- **Redémarrer Apache** : `sudo systemctl restart apache2`

Apache HTTPD 2.4 : rappels et nouveautés

Comparatif dans les installations

Méthode	Avantages	Inconvénients	Cas d'usage
Gestionnaire de paquets	Facile, stable, rapide	Peu de personnalisation	Serveurs de production standard
Compilation (sources)	Personnalisation, flexibilité	Maintenance manuelle, complexe	Besoins spécifiques, environnements avancés
Conteneurs Docker	Isolation, portabilité	Complexité initiale	Microservices, CI/CD, Cloud-native
Outils d'automatisation	Automatisation, réplicabilité	Nécessite des outils tiers	Environnements à grande échelle
Services Cloud préconfigurés	Simplicité, haute disponibilité	Coût élevé, dépendance	Startups, mise à l'échelle rapide

Apache HTTPD 2.4 : rappels et nouveautés

Comparatif dans les installations

La meilleure façon de faire

Cas généraux : Utilisation du gestionnaire de paquets

- Simple, rapide et stable.
- Idéal pour des environnements de production nécessitant peu de personnalisation.

Cas spécifiques : Compilation des sources

- Permet de répondre à des besoins particuliers (versions spécifiques, optimisations).
- Nécessite un bon niveau en administration système.

Environnements modernes : Docker et Automatisation

- Docker pour des environnements flexibles, réplicables et isolés.
- Automatisation (Ansible, Terraform) pour les infrastructures complexes à grande échelle.

Apache HTTPD 2.4 : rappels et nouveautés

Configuration générale du serveur:

La localisation des dossiers principaux d'**Apache HTTPD** dépend de la méthode d'installation et du système d'exploitation.

1. Installation par les gestionnaires de paquets

Si Apache est installé via un gestionnaire de paquets (comme `apt` ou `yum`), les fichiers principaux se trouvent généralement sous `/etc/` pour la configuration et `/var/www/` pour les fichiers web.

Debian/Ubuntu

- **Fichiers de configuration** : Localisés sous `/etc/apache2/`.
  - `/etc/apache2/apache2.conf` : Fichier principal de configuration.
  - `/etc/apache2/sites-available/` : Configurations des Virtual Hosts disponibles.
  - `/etc/apache2/sites-enabled/` : Liens symboliques vers les Virtual Hosts activés.
  - `/etc/apache2/mods-available/` : Modules disponibles.
  - `/etc/apache2/mods-enabled/` : Liens symboliques vers les modules activés.
- **Répertoire des sites web** : `/var/www/`
  - Par défaut, `/var/www/html` est le répertoire racine pour les fichiers accessibles par le serveur.

Apache HTTPD 2.4 : rappels et nouveautés

Configuration générale du serveur:

La localisation des dossiers principaux d'**Apache HTTPD** dépend de la méthode d'installation et du système d'exploitation.

1. Installation par les gestionnaires de paquets

CentOS/RHEL

- **Fichiers de configuration** : Localisés sous `/etc/httpd/`.
  - `/etc/httpd/conf/httpd.conf` : Fichier principal de configuration.
  - `/etc/httpd/conf.d/` : Configurations additionnelles (ex. SSL, Virtual Hosts).
  - `/etc/httpd/conf.modules.d/` : Liste des modules disponibles ou chargés.
- **Répertoire des sites web** : `/var/www/`
  - Par défaut, `/var/www/html` est utilisé comme racine des sites.

Apache HTTPD 2.4 : rappels et nouveautés

Configuration générale du serveur:

2. Installation à partir des sources

Si Apache est installé à partir des **sources**, l'organisation par défaut diffère. En général, les fichiers principaux sont placés sous `/usr/local/`, sauf si des options spécifiques sont définies lors de la compilation.

Répertoires typiques :

- **Configuration** : `/usr/local/apache2/conf/`
  - `/usr/local/apache2/conf/httpd.conf` : Fichier principal de configuration.
- **Modules** : `/usr/local/apache2/modules/`
- **Logs** : `/usr/local/apache2/logs/`
- **Répertoire des sites web** : `/usr/local/apache2/htdocs/`

Apache HTTPD 2.4 : rappels et nouveautés

Configuration générale du serveur:

3. Comparaison des répertoires principaux

Installation	Configuration	Modules	Sites web
Debian/Ubuntu	`/etc/apache2/`	`/etc/apache2/mods-available/`	`/var/www/html/`
CentOS/RHEL	`/etc/httpd/`	`/etc/httpd/modules/`	`/var/www/html/`
Compilation (sources)	`/usr/local/apache2/conf/`	`/usr/local/apache2/modules/`	`/usr/local/apache2/htdocs/`

Apache HTTPD 2.4 : rappels et nouveautés

Choisir le bon MPM (Multi-Processing Module) :

- Le choix du MPM et la gestion des limites sont cruciaux pour optimiser les performances d'Apache en fonction de la charge de travail et des ressources système.
- Apache HTTPD offre plusieurs MPMs. Chaque MPM a ses propres caractéristiques qui influencent les performances et l'utilisation des ressources

Apache HTTPD 2.4 : rappels et nouveautés

Gérer la charge et les limites :

MPM	Description	Avantages	Limites
prefork	Basé sur des processus, chaque requête est traitée par un processus distinct.	- Stable. - Compatible avec les modules non thread-safe. - Simple à configurer.	- Consomme beaucoup de mémoire. - Performances limitées pour les charges élevées.
worker	Combinaison de processus et threads. Un processus peut traiter plusieurs requêtes via des threads.	- Performances élevées. - Meilleure utilisation des ressources. - Moins de mémoire utilisée.	- Modules non thread-safe incompatibles. - Configuration plus complexe.

Apache HTTPD 2.4 : rappels et nouveautés

Gérer la charge et les limites :

MPM	Description	Avantages	Limites
event	Extension de worker. Optimisé pour les connexions persistantes (Keep-Alive).	- Très performant pour les connexions Keep-Alive. - Faible consommation des ressources.	- Modules non thread-safe incompatibles. - Configuration avancée requise.

Apache HTTPD 2.4 : rappels et nouveautés

Choisir le bon MPM (Multi-Processing Module) :

2. Choisir le bon MPM

Critères de choix :

1. **Charge prévue :**
   - **Faible charge ou faible trafic** : `prefork` est suffisant.
   - **Charge moyenne à élevée** : `worker` ou `event` est plus adapté.
   - **Connexions persistantes (Keep-Alive)** : `event` est recommandé.
2. **Compatibilité des modules :**
   - **Modules non thread-safe** (par exemple certaines extensions PHP) : Utilisez `prefork`.
   - **Modules thread-safe** : Préférez `worker` ou `event`.

Apache HTTPD 2.4 : rappels et nouveautés

Choisir le bon MPM (Multi-Processing Module) :

3. Configurer les limites pour chaque MPM

a. Configuration du MPM `prefork`

Dans le fichier de configuration Apache (`httpd.conf` ou un fichier spécifique selon votre système), ajustez les paramètres suivants :

```
<IfModule mpm_prefork_module>
    StartServers          5         # Nombre initial de processus.
    MinSpareServers       5         # Processus minimum en attente.
    MaxSpareServers      10         # Processus maximum en attente.
    MaxRequestWorkers    150        # Limite maximale de processus simultanés.
    MaxConnectionsPerChild 0        # Nombre de requêtes par processus (0 = illimité).
</IfModule>
```

Apache HTTPD 2.4 : rappels et nouveautés

Choisir le bon MPM (Multi-Processing Module) :

3. Configurer les limites pour chaque MPM

b. Configuration du MPM `worker`

Les paramètres à ajuster pour `worker` :

```
<IfModule mpm_worker_module>
    StartServers          2       # Processus de démarrage.
    MinSpareThreads      25       # Threads minimum en attente.
    MaxSpareThreads      75       # Threads maximum en attente.
    ThreadLimit          64       # Limite de threads par processus.
    ThreadsPerChild      25       # Threads gérés par chaque processus.
    MaxRequestWorkers    150      # Nombre total maximum de threads (processeurs * threads).
    MaxConnectionsPerChild 0      # Nombre de requêtes avant de recréer un processus.
</IfModule>
```

Apache HTTPD 2.4 : rappels et nouveautés

Choisir le bon MPM (Multi-Processing Module) :

3. Configurer les limites pour chaque MPM

c. Configuration du MPM `event`

Les paramètres pour `event` sont similaires à ceux de `worker`, avec des optimisations pour les connexions Keep-Alive :

```
<IfModule mpm_event_module>
    StartServers          2
    MinSpareThreads      25
    MaxSpareThreads      75
    ThreadLimit          64
    ThreadsPerChild      25
    MaxRequestWorkers   150
    MaxConnectionsPerChild 0
</IfModule>
```

Apache HTTPD 2.4 : rappels et nouveautés

Choisir le bon MPM (Multi-Processing Module) :

4. Gérer la charge : Limites et optimisations

a. MaxRequestWorkers

- Définit le nombre maximum de requêtes simultanées que le serveur peut traiter.
- Adaptez ce paramètre en fonction de la mémoire disponible et de la charge prévue.

b. Timeout

- Contrôle le temps maximum qu'Apache attend pour qu'une requête soit traitée.
- Réduisez cette valeur pour éviter des connexions bloquantes :

```
Timeout 60
```

Apache HTTPD 2.4 : rappels et nouveautés

Choisir le bon MPM (Multi-Processing Module) :

4. Gérer la charge : Limites et optimisations

c. Keep-Alive

- Permet aux clients de réutiliser une connexion TCP pour plusieurs requêtes.
- Réglez les paramètres pour optimiser les connexions persistantes :

```
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5
```

d. Gestion de la mémoire

- Surveillez l'utilisation de la mémoire avec des outils comme `top` ou `htop`.
- Augmentez les limites système si nécessaire :
  - Exemple pour augmenter les fichiers ouverts :

```
ulimit -n 4096
```

Apache HTTPD 2.4 : rappels et nouveautés

Choisir le bon MPM (Multi-Processing Module) :

4. Gérer la charge : Limites et optimisations

e. Surveillance et monitoring

- Utilisez des outils comme **Prometheus**, **Nagios**, ou **Grafana** pour surveiller la charge et les performances.
- Analysez les fichiers de logs pour détecter les goulots d'étranglement :

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

Apache HTTPD 2.4 : rappels et nouveautés

Choisir le bon MPM (Multi-Processing Module) :

5. Tests et ajustements

1. **Testez vos paramètres avec des outils de charge** :
   - Exemple : **Apache Benchmark** (ab) :

     ```
     ab -n 1000 -c 100 http://localhost/
     ```

   - Cela permet de tester comment Apache gère un grand nombre de requêtes simultanées.
2. **Ajustez les paramètres** :
   - Si Apache atteint la limite de `MaxRequestWorkers`, vous devrez augmenter cette valeur ou optimiser l'application.

Apache HTTPD 2.4 : rappels et nouveautés

Choisir le bon MPM (Multi-Processing Module) :

6. Résumé : Quelle stratégie adopter ?

Scénario	MPM recommandé	Stratégie
Site avec faible trafic	`prefork`	Paramètres par défaut, Keep-Alive activé.
Site avec trafic modéré/élevé	`worker`	Ajustez `ThreadsPerChild` et `MaxRequestWorkers` pour gérer la charge.
API ou connexions Keep-Alive	`event`	Optimisez Keep-Alive et les threads pour maximiser les performances avec des connexions longues.

Apache HTTPD 2.4 : rappels et nouveautés

Choisir le bon MPM (Multi-Processing Module) :

2. Choisir le bon MPM

Critères de choix :

3. **Ressources disponibles :**
   - **Mémoire limitée** : Évitez `prefork` et optez pour `worker` ou `event`.
   - **Environnements virtualisés/cloud** : Préférez `worker` ou `event`.
4. **Type d'application :**
   - Applications lourdes en PHP : `prefork`.
   - API ou sites modernes : `worker` ou `event`.

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

- Apache HTTPD est un serveur modulaire, ce qui signifie que ses fonctionnalités peuvent être étendues ou réduites en activant ou désactivant des modules.
- Le choix des modules à activer dépend de vos besoins spécifiques, comme la sécurité, les performances, ou les fonctionnalités nécessaires pour votre application.

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

1. Gestion des modules dans Apache

b. Activer un module

- Sur **Ubuntu/Debian** :

```
sudo a2enmod module_name
```

Exemple : Activer le module `rewrite` :

```
sudo a2enmod rewrite
sudo systemctl reload apache2
```

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

1. Gestion des modules dans Apache

- Sur **CentOS/RHEL** :
  - Modifiez le fichier principal de configuration (`httpd.conf`) ou les fichiers sous `/etc/httpd/conf.modules.d/`.
  - Exemple :

```
LoadModule rewrite_module modules/mod_rewrite.so
```

  - Rechargez la configuration :

```
sudo systemctl reload httpd
```

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

1. Gestion des modules dans Apache

c. Désactiver un module

- Sur **Ubuntu/Debian** :

```
sudo a2dismod module_name
sudo systemctl reload apache2
```

- Sur **CentOS/RHEL** :
  - Commentez ou supprimez la ligne `LoadModule` correspondante.
    ``

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

2. Modules essentiels et leurs rôles : Module base

Module	Description
`mod_core`	Module de base, chargé automatiquement.
`mod_mpm_*`	Multi-Processing Modules (choisir `prefork`, `worker` ou `event`).
`mod_log_config`	Permet de configurer les journaux (access logs, error logs).
`mod_status`	Fournit une page de statut pour surveiller les performances et connexions actives.
`mod_alias`	Permet de créer des alias d'URL (par exemple, rediriger `/docs` vers un dossier spécifique).

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

2. Modules essentiels et leurs rôles : Modules de performance

Module	Description
`mod_deflate`	Compression des réponses HTTP pour économiser de la bande passante.
`mod_headers`	Permet de modifier les en-têtes HTTP.
`mod_cache`	Met en cache les contenus pour accélérer les réponses.
`mod_expires`	Définit les politiques de cache basées sur la durée de vie des ressources.

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

2. Modules essentiels et leurs rôles : Modules de sécurité

Module	Description
`mod_ssl`	Ajoute le support HTTPS (SSL/TLS).
`mod_security`	Améliore la sécurité en bloquant les requêtes malveillantes (WAF - Web Application Firewall).
`mod_auth_basic`	Authentification basique par mot de passe.
`mod_auth_digest`	Authentification plus sécurisée que `mod_auth_basic`.
`mod_evasive`	Protège contre les attaques DDoS ou brute-force en limitant les connexions répétées.

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

2. Modules essentiels et leurs rôles : Redirections et réécritures

Module	Description
`mod_rewrite`	Permet la réécriture des URL (très utilisé pour les sites dynamiques et les SEO).
`mod_redirect`	Configure des redirections HTTP simples.

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

2. Modules essentiels et leurs rôles : Applications et contenu

Module	Description
`mod_cgi`	Exécute des scripts CGI.
`mod_php`	Intègre le support PHP directement dans Apache (souvent remplacé par PHP-FPM).
`mod_proxy`	Configure un proxy ou une passerelle pour d'autres serveurs.
`mod_proxy_http`	Support pour le proxy HTTP (souvent utilisé avec `mod_proxy`).
`mod_proxy_fcgi`	Permet la communication avec PHP-FPM via FastCGI.
`mod_wsgi`	Permet d'exécuter des applications Python (par exemple, Django).

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

2. Modules essentiels et leurs rôles : Modules de diagnostic

Module	Description
`mod_info`	Fournit des informations sur la configuration actuelle d'Apache.
`mod_status`	Fournit des statistiques en temps réel sur le serveur (ex. requêtes actives).

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

3. Quels modules activer selon les besoins

a. Serveur web classique (HTML/CSS/JS uniquement)

- Modules recommandés :
  - `mod_deflate`
  - `mod_headers`
  - `mod_alias`
  - `mod_expires`
  - `mod_status`

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

3. Quels modules activer selon les besoins

b. Serveur pour applications dynamiques (PHP, Python, etc.)

- Modules recommandés :
  - `mod_php` ou `mod_proxy_fcgi` (pour PHP-FPM)
  - `mod_wsgi` (pour Python)
  - `mod_rewrite`
  - `mod_deflate`
  - `mod_headers`

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

3. Quels modules activer selon les besoins

c. Serveur sécurisé avec HTTPS

- Modules recommandés :
  - `mod_ssl`
  - `mod_security`
  - `mod_evasive`
  - `mod_headers`
  - `mod_rewrite`

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

3. Quels modules activer selon les besoins

d. Serveur proxy/reverse proxy

- Modules recommandés :
  - `mod_proxy`
  - `mod_proxy_http`
  - `mod_proxy_fcgi`
  - `mod_cache`
  - `mod_headers`

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

4. Désactiver les modules inutiles

Pour des raisons de performance et de sécurité, désactivez les modules non utilisés. Par exemple :

- Modules CGI si vous n'exécutez pas de scripts CGI :

```
sudo a2dismod cgi
sudo systemctl reload apache2
```

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

5. Vérification après activation/désactivation

- **Tester la configuration Apache :**

```
apache2ctl configtest
```

Exemple de sortie si tout est correct :

```
Syntax OK
```

- **Vérifier les modules activés :**

```
apache2ctl -M
```

Apache HTTPD 2.4 : rappels et nouveautés

Chargement des modules et modules à activer :

Résumé

1. **Modules essentiels** : `mod_ssl`, `mod_rewrite`, `mod_headers`, `mod_status`.
2. **Modules à activer selon les besoins spécifiques** :
   - Performance : `mod_deflate`, `mod_expires`, `mod_cache`.
   - Sécurité : `mod_security`, `mod_evasive`.
   - Applications dynamiques : `mod_php`, `mod_wsgi`, `mod_proxy_fcgi`.
3. Désactivez les modules inutiles pour améliorer la sécurité et les performances.

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

- Apache HTTPD 2.4 introduit des fonctionnalités permettant une gestion plus fine des configurations grâce à de nouveaux **types de contextes**.
- Ces contextes permettent de personnaliser les comportements selon des conditions ou des environnements spécifiques.
- Ces contextes sont une manière de structurer la configuration du serveur en fonction de l'endroit ou du niveau où les directives s'appliquent.

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

1. Contextes standards existants dans Apache

a. Serveur global (`server config`)

- **Description** :
  - Les directives s'appliquent à l'ensemble du serveur.
  - Configuré directement dans le fichier principal (`httpd.conf` ou `apache2.conf`).
- **Exemple** :

```
ServerRoot "/etc/httpd"
Listen 80
```

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

b. Virtual Host (`virtual host`)

- **Description** :
  - Les directives s'appliquent à un hôte virtuel spécifique (une configuration pour chaque domaine ou sous-domaine).
  - Définies dans les fichiers comme `/etc/apache2/sites-available/example.conf`.
- **Exemple** :

```
<VirtualHost *:80>
    ServerName www.example.com
    DocumentRoot /var/www/example
</VirtualHost>
```

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

c. Répertoires (`directory`)

- **Description** :
  - Les directives s'appliquent à des répertoires ou chemins spécifiques du système de fichiers.
- **Exemple** :

```
<Directory "/var/www/html">
    AllowOverride All
    Require all granted
</Directory>
```

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

d. Emplacement URL (`location`)

- **Description** :
  - Les directives s'appliquent à des parties spécifiques de l'URL.
- **Exemple** :

```
<Location "/admin">
    Require ip 192.168.1.0/24
</Location>
```

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

e. Fichiers (`files`)

- **Description** :
  - Les directives s'appliquent à des fichiers spécifiques.
- **Exemple** :

```
<Files "secret.html">
    Require all denied
</Files>
```

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

f. Module spécifique (`ifmodule`)

- **Description** :
  - Les directives ne s'appliquent que si un module spécifique est activé.
- **Exemple** :

```
<IfModule mod_ssl.c>
    SSLEngine On
</IfModule>
```

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

2. Nouveaux types ou usages de contextes dans Apache

Avec les versions modernes d'Apache HTTPD (2.4+), de nouveaux types de contextes ou des fonctionnalités avancées ont émergé.

a. Contexte conditionnel (`if`)

- **Description** :
  - Permet de conditionner les directives en fonction d'expressions dynamiques (comme des variables d'environnement, des conditions IP, etc.).
- **Exemple** :

```
<If "%{HTTP_HOST} == 'example.com'">
    DocumentRoot "/var/www/example"
</If>
```

- **Utilité** :
  - Flexibilité accrue dans les configurations.
  - Simplifie les scénarios complexes comme la gestion de plusieurs domaines ou configurations spécifiques.

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

2. Nouveaux types ou usages de contextes dans Apache

Avec les versions modernes d'Apache HTTPD (2.4+), de nouveaux types de contextes ou des fonctionnalités avancées ont émergé.

b. Contexte utilisateur (`userdir`)

- **Description** :
  - Définit les règles pour les répertoires personnels des utilisateurs.
- **Exemple** :

```
<IfModule mod_userdir.c>
    UserDir public_html
    <Directory "/home/*/public_html">
        AllowOverride FileInfo AuthConfig
        Require all granted
    </Directory>
</IfModule>
```

- **Utilité** :
  - Permet aux utilisateurs du système d'avoir leurs propres sites web accessibles via `/~username`.

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

2. Nouveaux types ou usages de contextes dans Apache

Avec les versions modernes d'Apache HTTPD (2.4+), de nouveaux types de contextes ou des fonctionnalités avancées ont émergé.

c. Contexte fichier spécifique (`filesmatch`)

- **Description** :
  - S'applique aux fichiers correspondant à des motifs spécifiques (par exemple, via regex).
- **Exemple** :

```
<FilesMatch "\.(txt|log)$">
    Require all denied
</FilesMatch>
```

- **Utilité** :
  - Utile pour protéger certains types de fichiers sensibles (comme `.env`, `.htpasswd`, etc.).

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

2. Nouveaux types ou usages de contextes dans Apache

Avec les versions modernes d'Apache HTTPD (2.4+), de nouveaux types de contextes ou des fonctionnalités avancées ont émergé.

d. Contexte dynamique pour les proxys (`proxy`)

- **Description** :
  - Utilisé pour gérer des directives spécifiques aux connexions proxy ou reverse proxy.
- **Exemple** :

```
<Proxy "http://backend.example.com">
    Require ip 192.168.0.0/24
</Proxy>
```

- **Utilité** :
  - Permet un contrôle précis des connexions aux proxys.

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

2. Nouveaux types ou usages de contextes dans Apache

e. Contexte dynamique par moteur de règles (`ifdefine` et `ifenv`)

- **Description** :
  - Conditionne les directives en fonction des variables définies ou de l'environnement.
- **Exemples** :
  - **IfDefine** :

```
<IfDefine DEBUG>
    LogLevel debug
</IfDefine>
```

    - Utilisé pour activer/désactiver des configurations spécifiques en ligne de commande :

```
apache2ctl -D DEBUG
```

  - **IfEnv** :

```
<IfEnv TEST_ENV>
    DocumentRoot "/var/www/test"
</IfEnv>
```

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

2. Nouveaux types ou usages de contextes dans Apache

f. Contexte par balises HTML intégrées (`htaccess inline`)

- **Description** :
  - Directives placées directement dans un fichier `.htaccess`.
- **Exemple** :

```
RewriteEngine On
RewriteRule ^about$ /about.html [L]
```

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

3. Bonnes pratiques dans l'utilisation des contextes

1. **Minimisez l'utilisation des fichiers `.htaccess`** :
   - Préférez la configuration dans les fichiers de Virtual Host ou dans le fichier principal (`httpd.conf`).
   - `.htaccess` ajoute une surcharge, car Apache le charge à chaque requête.
2. **Utilisez des directives conditionnelles (`<If>` et `<IfModule>`)** pour les configurations dynamiques ou spécifiques.

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

3. Bonnes pratiques dans l'utilisation des contextes

3. **Protégez les fichiers sensibles avec des contextes `FilesMatch`** :
   - Exemple pour protéger `.htpasswd` :

```
<FilesMatch "^\.ht">
    Require all denied
</FilesMatch>
```

4. **Planifiez et organisez vos Virtual Hosts** :
   - Placez un fichier distinct par domaine dans `/etc/apache2/sites-available/` ou `/etc/httpd/conf.d/`.

Apache HTTPD 2.4 : rappels et nouveautés

Nouveaux types de contextes dans Apache HTTPD 2.4

Résumé des nouveaux types de contextes

Contexte	Usage principal
`<If>`	Conditions dynamiques basées sur des variables ou expressions (ex. domaine, IP).
`<FilesMatch>`	S'applique aux fichiers correspondant à un motif spécifique (regex).
`<Proxy>`	Contrôle des proxys et reverse-proxys.
`<IfDefine>`	Activé/désactivé en fonction de variables définies au démarrage d'Apache.
`<IfEnv>`	Activé/désactivé en fonction des variables d'environnement.

Apache HTTPD 2.4 : rappels et nouveautés

Panorama des modules d'Apache 2.4

Les modules étendent les fonctionnalités d'Apache HTTPD et permettent de répondre à des besoins spécifiques en matière de sécurité, performances, gestion des contenus, proxy, etc. Nous allons faire un petit panorama des modules par catégorie.

Apache HTTPD 2.4 : rappels et nouveautés

Panorama des modules d'Apache 2.4

1. Modules de base

Ces modules sont nécessaires au fonctionnement principal du serveur Apache.

Module	Description
`core`	Gère les fonctionnalités principales (traitement des requêtes, gestion des connexions).
`mpm_prefork`	Multi-Processing Module basé sur des processus (mode non threadé).
`mpm_worker`	Multi-Processing Module basé sur des threads et processus combinés.
`mpm_event`	Extension de `worker`, optimisé pour les connexions persistantes (Keep-Alive).
`mod_log_config`	Permet la configuration des journaux d'accès et d'erreurs.
`mod_version`	Active des directives en fonction de la version d'Apache.

Apache HTTPD 2.4 : rappels et nouveautés

Panorama des modules d'Apache 2.4 : Modules de sécurité

2. Modules de sécurité

Module	Description
`mod_ssl`	Ajoute le support HTTPS via SSL/TLS.
`mod_auth_basic`	Permet l'authentification basique par mot de passe.
`mod_auth_digest`	Authentification plus sécurisée que `mod_auth_basic`.
`mod_authz_host`	Contrôle l'accès basé sur les adresses IP.
`mod_security`	Module tiers pour un pare-feu d'application web (WAF).
`mod_evasive`	Protège contre les attaques DoS/DDoS.
`mod_headers`	Ajoute ou modifie les en-têtes HTTP (utile pour CSP, CORS, etc.).
`mod_remoteip`	Remplace l'adresse IP client par celle envoyée dans un en-tête (proxy).

Apache HTTPD 2.4 : rappels et nouveautés

Panorama des modules d'Apache 2.4 : Modules de contenus

Module	Description
`mod_dir`	Permet de définir le fichier par défaut d'un répertoire (exemple : `index.html`).
`mod_autoindex`	Génère automatiquement une liste de fichiers pour les répertoires sans index.
`mod_alias`	Crée des alias d'URL (exemple : `/docs` vers un autre répertoire).
`mod_negotiation`	Permet la négociation de contenu (choix du fichier basé sur les préférences du client).
`mod_mime`	Détermine le type MIME des fichiers en fonction de leur extension.
`mod_rewrite`	Permet la réécriture dynamique des URL.
`mod_deflate`	Comprime les réponses HTTP pour réduire l'utilisation de la bande passante.

Apache HTTPD 2.4 : rappels et nouveautés

Panorama des modules d'Apache 2.4 : Modules de proxy et reverse proxy

Module	Description
`mod_proxy`	Active les fonctionnalités de proxy de base.
`mod_proxy_http`	Permet le proxying des requêtes HTTP.
`mod_proxy_fcgi`	Gère les connexions avec des applications FastCGI (exemple : PHP-FPM).
`mod_proxy_balancer`	Gère l'équilibrage de charge entre plusieurs serveurs backend.
`mod_proxy_wstunnel`	Supporte les connexions WebSocket via un proxy.
`mod_proxy_ftp`	Permet le proxying des connexions FTP.

Apache HTTPD 2.4 : rappels et nouveautés

Panorama des modules d'Apache 2.4 : Modules pour les applications dynamiques

Module	Description
`mod_cgi`	Exécute des scripts CGI traditionnels.
`mod_php`	Intègre PHP directement dans Apache (souvent remplacé par PHP-FPM).
`mod_perl`	Permet d'exécuter des scripts Perl.
`mod_wsgi`	Exécute des applications Python (exemple : Django).
`mod_lua`	Permet d'intégrer des scripts Lua dans Apache.

Apache HTTPD 2.4 : rappels et nouveautés

Panorama des modules d'Apache 2.4 : Modules conditionnels

Module	Description
`mod_status`	Fournit une page de statut avec des informations sur les performances et connexions.
`mod_info`	Affiche la configuration complète d'Apache pour diagnostic.
`mod_watchdog`	Fournit un mécanisme pour surveiller les services d'arrière-plan d'Apache.

Apache HTTPD 2.4 : rappels et nouveautés

Panorama des modules d'Apache 2.4 : Modules de diagnostic

Module	Description
`mod_if`	Permet d'utiliser des conditions dynamiques basées sur des variables.
`mod_define`	Définit des variables utilisées dans les directives conditionnelles.
`mod_env`	Permet de définir ou d'utiliser des variables d'environnement.

Apache HTTPD 2.4 : rappels et nouveautés

Panorama des modules d'Apache 2.4 : Modules spécifiques à HTTP/2

Module	Description
`mod_http2`	Ajoute le support HTTP/2, améliorant la vitesse et les performances des connexions.

Apache HTTPD 2.4 : rappels et nouveautés

Panorama des modules d'Apache 2.4 : Modules tiers populaires

Module	Description
`mod_pagespeed`	Optimise automatiquement les performances des sites (cache, minification, etc.).
`mod_security`	Fournit une protection WAF (pare-feu d'application web).
`mod_cluster`	Permet le clustering avec Apache Tomcat ou JBoss.

Apache HTTPD 2.4 : rappels et nouveautés

Panorama des modules d'Apache 2.4

Résumé des modules essentiels à activer selon les besoins

Type de serveur	Modules essentiels à activer
Serveur de base (HTTP)	`mod_dir`, `mod_alias`, `mod_mime`, `mod_deflate`, `mod_headers`, `mod_rewrite`
Serveur HTTPS	`mod_ssl`, `mod_headers`, `mod_deflate`, `mod_expires`
Proxy ou reverse proxy	`mod_proxy`, `mod_proxy_http`, `mod_proxy_fcgi`, `mod_proxy_balancer`
Applications dynamiques	`mod_php`, `mod_proxy_fcgi`, `mod_wsgi`, `mod_perl`
Serveur haute performance	`mod_http2`, `mod_deflate`, `mod_expires`, `mod_cache`
Sécurité avancée	`mod_security`, `mod_evasive`, `mod_headers`, `mod_remoteip`

Apache HTTPD 2.4 : rappels et nouveautés

Support du protocole HTTP/2

- Apache HTTPD 2.4 introduit le support natif du protocole **HTTP/2** via le module `mod_http2`.
- HTTP/2 est conçu pour améliorer les performances des sites web modernes.
- Le protocole **HTTP/2** est une amélioration significative de HTTP/1.1.
- Il améliore les performances, la latence et la gestion des connexions réseau.
- Apache HTTPD prend en charge HTTP/2 à partir de la version **2.4.17**, mais il nécessite une configuration spécifique.

Apache HTTPD 2.4 : rappels et nouveautés

Support du protocole HTTP/2

1. Avantages du protocole HTTP/2

- **Multiplexage** : Plusieurs requêtes et réponses peuvent être échangées simultanément sur une seule connexion TCP.
- **Compression des en-têtes** : Réduit la taille des en-têtes HTTP, ce qui améliore les performances.
- **Priorisation des ressources** : Permet de prioriser certains contenus, comme les fichiers CSS ou JavaScript critiques.
- **Connexions persistantes** : Réduit la surcharge en limitant l'ouverture de connexions multiples.
- **Performances accrues** : Idéal pour les sites modernes, notamment ceux avec beaucoup de fichiers (CSS, JS, images).

Apache HTTPD 2.4 : rappels et nouveautés

Support du protocole HTTP/2

2. Prérequis pour activer HTTP/2

1. **Version d'Apache** : HTTP/2 est pris en charge à partir de la version **2.4.17**.
   - Vérifiez votre version avec :

```
apache2 -v
```

2. **SSL/TLS activé** : HTTP/2 nécessite HTTPS pour être activé dans Apache.
3. **Module requis** : Le module `mod_http2` doit être activé.
4. **Version d'OpenSSL** : OpenSSL **1.0.2** ou supérieur est requis.

Apache HTTPD 2.4 : rappels et nouveautés

Support du protocole HTTP/2

3. Étapes pour activer HTTP/2

a. Vérifier et activer le module HTTP/2

- **Ubuntu/Debian** :
  Activez le module `http2` :

```
sudo a2enmod http2
sudo systemctl reload apache2
```

b. Modifier la configuration SSL pour activer HTTP/2

Ajoutez ou modifiez la directive `Protocols` dans le fichier de configuration du Virtual Host HTTPS.

Apache HTTPD 2.4 : rappels et nouveautés

Support du protocole HTTP/2

3. Étapes pour activer HTTP/2

b. Modifier la configuration SSL pour activer HTTP/2

Exemple de configuration dans `/etc/apache2/sites-available/default-ssl.conf` (Ubuntu) ou `/etc/httpd/conf.d/ssl.conf` (CentOS) :

```
<VirtualHost *:443>
    ServerName example.com
    DocumentRoot /var/www/html

    SSLEngine On
    SSLCertificateFile /etc/apache2/ssl/example.crt
    SSLCertificateKeyFile /etc/apache2/ssl/example.key

    # Activer HTTP/2
    Protocols h2 h2c http/1.1

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Apache HTTPD 2.4 : rappels et nouveautés

Support du protocole HTTP/2

3. Étapes pour activer HTTP/2

c. Vérifier et activer MPM event

HTTP/2 fonctionne mieux avec le MPM **event**. Activez-le si ce n'est pas déjà fait :

- Désactivez le MPM prefork (si activé) :

```
sudo a2dismod mpm_prefork
```

- Activez le MPM event :

```
sudo a2enmod mpm_event
```

- Rechargez Apache :

```
sudo systemctl reload apache2
```

Apache HTTPD 2.4 : rappels et nouveautés

Support du protocole HTTP/2

4. Tester HTTP/2

a. Avec un navigateur

- Accédez à votre site via HTTPS (`https://example.com`).
- Ouvrez les outils de développement (F12), allez dans l'onglet **Network**, et vérifiez la colonne **Protocol**. Si HTTP/2 est activé, vous verrez `h2`.

b. Avec `curl`

Utilisez l'option `--http2` :

```
curl -I --http2 https://example.com
```

c. Avec un outil en ligne

Utilisez des outils comme HTTP/2 Test pour vérifier si HTTP/2 est activé sur votre site.

Apache HTTPD 2.4 : rappels et nouveautés

Support du protocole HTTP/2

5. Résolution des problèmes courants

1. **HTTP/2 non activé malgré la configuration**
   - Vérifiez que le module `http2` est chargé :

     ```
     apache2ctl -M | grep http2
     ```

     Vous devriez voir :

     ```
     http2_module (shared)
     ```

2. **Erreurs liées à OpenSSL**
   - Assurez-vous que votre version d'OpenSSL est **1.0.2** ou supérieure :

     ```
     openssl version
     ```

Apache HTTPD 2.4 : rappels et nouveautés

Support du protocole HTTP/2

5. Résolution des problèmes courants

3. Problème de compatibilité avec MPM prefork

- HTTP/2 ne fonctionne pas avec `mpm_prefork`. Utilisez `mpm_event` ou `mpm_worker`.

Apache HTTPD 2.4 : rappels et nouveautés

Support du protocole HTTP/2

6. Résumé des directives importantes pour HTTP/2

Directive	Description
`Protocols`	Définit les protocoles pris en charge (exemple : `h2 h2c http/1.1`).
`ProtocolsHonorOrder`	Priorise les protocoles dans l'ordre spécifié dans `Protocols`.

Apache HTTPD 2.4 : rappels et nouveautés

Support du protocole HTTP/2

7. Bénéfices pratiques de HTTP/2

1. **Sites à fort trafic** : HTTP/2 réduit la latence et améliore les temps de chargement.
2. **Applications modernes** : Idéal pour les sites avec beaucoup de fichiers (CSS, JS, images).
3. **Meilleures performances SEO** : Les moteurs de recherche favorisent les sites rapides et sécurisés.

Apache HTTPD 2.4 : rappels et nouveautés

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Héberger des applications PHP

Héberger des applications PHP

Héberger des applications PHP et faire cohabiter **PHP 5** et **PHP 7** sur le même serveur Apache peut être utile lorsque vous devez prendre en charge des applications héritées utilisant PHP 5 tout en supportant des applications modernes développées avec PHP 7 ou une version ultérieure.

Héberger des applications PHP

Faire cohabiter PHP5 et PHP7.

1. Concepts clés

1. **Apache et PHP-FPM** :
   - Apache peut exécuter plusieurs versions de PHP en utilisant **PHP-FPM** (FastCGI Process Manager).
   - Chaque version de PHP est configurée dans un pool distinct.
2. **Virtual Hosts** :
   - Vous configurez Apache pour assigner des versions spécifiques de PHP à des **Virtual Hosts** (ou des répertoires spécifiques).

Héberger des applications PHP

Faire cohabiter PHP5 et PHP7.

2. Installation des versions PHP

Installer PHP 5.6 et PHP 7.4 (ou autres versions)

Sur Ubuntu/Debian :

1. Installez les dépôts nécessaires :

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository ppa:ondrej/php
sudo apt update
```

2. Installez PHP 5.6 et PHP 7.4 avec leurs modules FPM :

```
sudo apt install php5.6 php5.6-fpm php7.4 php7.4-fpm
```

Héberger des applications PHP

Faire cohabiter PHP5 et PHP7.

3. Configurer PHP-FPM

Configurer les pools pour chaque version

Les pools PHP-FPM permettent de séparer les applications exécutées sous différentes versions de PHP.

1. **Fichier de configuration PHP-FPM pour PHP 5.6** :
   - Chemin : `/etc/php/5.6/fpm/pool.d/www.conf` (ou similaire selon l'OS).
   - Modifiez le port ou le socket pour éviter les conflits.

```
[www]
listen = /run/php/php5.6-fpm.sock
```

2. **Fichier de configuration PHP-FPM pour PHP 7.4** :
   - Chemin : `/etc/php/7.4/fpm/pool.d/www.conf`.
   - Exemple :

```
[www]
listen = /run/php/php7.4-fpm.sock
```

Héberger des applications PHP

Faire cohabiter PHP5 et PHP7.

3. Configurer PHP-FPM

Configurer les pools pour chaque version

3. Redémarrez les services PHP-FPM :

```
sudo systemctl restart php5.6-fpm
sudo systemctl restart php7.4-fpm
```

Héberger des applications PHP

Faire cohabiter PHP5 et PHP7.

4. Configurer Apache pour cohabiter PHP 5 et PHP 7

Activer les modules Apache nécessaires

Activez les modules requis pour PHP-FPM et FastCGI :

```
sudo a2enmod proxy_fcgi setenvif
sudo a2enmod actions
sudo systemctl reload apache2
```

Héberger des applications PHP

Faire cohabiter PHP5 et PHP7.

Configurer des Virtual Hosts

Créez un Virtual Host pour chaque version de PHP.

1. **Virtual Host pour PHP 5.6** :
   Fichier : `/etc/apache2/sites-available/php5.6.example.com.conf`.

```
<VirtualHost *:80>
    ServerName php5.6.example.com
    DocumentRoot /var/www/php5.6

    <Directory /var/www/php5.6>
        AllowOverride All
        Require all granted
    </Directory>

    <FilesMatch \.php$>
        SetHandler "proxy:unix:/run/php/php5.6-fpm.sock|fcgi://localhost"
    </FilesMatch>

    ErrorLog ${APACHE_LOG_DIR}/php5.6-error.log
    CustomLog ${APACHE_LOG_DIR}/php5.6-access.log combined
</VirtualHost>
```

Héberger des applications PHP

Faire cohabiter PHP5 et PHP7.

Configurer des Virtual Hosts

Créez un Virtual Host pour chaque version de PHP.

2. **Virtual Host pour PHP 7.4** :
   Fichier : `/etc/apache2/sites-available/php7.4.example.com.conf`.

```
<VirtualHost *:80>
    ServerName php7.4.example.com
    DocumentRoot /var/www/php7.4

    <Directory /var/www/php7.4>
        AllowOverride All
        Require all granted
    </Directory>

    <FilesMatch \.php$>
        SetHandler "proxy:unix:/run/php/php7.4-fpm.sock|fcgi://localhost"
    </FilesMatch>

    ErrorLog ${APACHE_LOG_DIR}/php7.4-error.log
    CustomLog ${APACHE_LOG_DIR}/php7.4-access.log combined
</VirtualHost>
```

Héberger des applications PHP

Faire cohabiter PHP5 et PHP7.

Configurer des Virtual Hosts

Créez un Virtual Host pour chaque version de PHP.

3. Activez les Virtual Hosts :

```
sudo a2ensite php5.6.example.com.conf
sudo a2ensite php7.4.example.com.conf
sudo systemctl reload apache2
```

Héberger des applications PHP

Faire cohabiter PHP5 et PHP7.

5. Tester la configuration

1. **Créer des répertoires de test**
   - Répertoire pour PHP 5.6 :

```
sudo mkdir -p /var/www/php5.6
echo "<?php phpinfo(); ?>" | sudo tee /var/www/php5.6/index.php
```

   - Répertoire pour PHP 7.4 :

```
sudo mkdir -p /var/www/php7.4
echo "<?php phpinfo(); ?>" | sudo tee /var/www/php7.4/index.php
```

Héberger des applications PHP

Faire cohabiter PHP5 et PHP7.

5. Tester la configuration

2. **Accéder aux Virtual Hosts**
    - Testez les URLs :
        - http://php5.6.example.com : Vous devriez voir la page PHP Info indiquant **PHP 5.6**.
        - http://php7.4.example.com : Vous devriez voir la page PHP Info indiquant **PHP 7.4**.

Héberger des applications PHP

Faire cohabiter PHP5 et PHP7.

6. Résolution des problèmes

- **Erreur 500 ou aucun contenu PHP exécuté :**
  - Vérifiez que les sockets PHP-FPM sont actifs :

```
sudo systemctl status php5.6-fpm
sudo systemctl status php7.4-fpm
```

  - Vérifiez les logs Apache pour plus de détails :

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

- **Conflits entre les versions PHP :**
  - Assurez-vous que chaque version utilise un socket ou un port distinct.

Héberger des applications PHP

Faire cohabiter PHP5 et PHP7.

Résumé

1. Installez plusieurs versions de PHP via PHP-FPM.
2. Configurez des pools PHP-FPM distincts pour chaque version.
3. Configurez Apache avec des Virtual Hosts assignés à chaque version PHP.
4. Testez les sites pour vérifier que chaque Virtual Host utilise la version PHP appropriée.

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

Ces technologies sont utilisées pour exécuter des applications dynamiques (comme des scripts PHP) sur des serveurs web tels qu'Apache.

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

1. Common Gateway Interface (CGI)

Description

- Le **Common Gateway Interface (CGI)** est une norme permettant à un serveur web de communiquer avec des scripts ou programmes externes pour générer des pages dynamiques.
- Fonctionnement :
  - Pour chaque requête, le serveur web exécute un nouveau processus pour le script.
  - Le script retourne une réponse au serveur qui est ensuite envoyée au client.

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

1. Common Gateway Interface (CGI)

Avantages

- Simplicité : Facile à implémenter.
- Compatibilité : Supporte de nombreux langages (Perl, Python, PHP, etc.).
- Isolation : Chaque requête étant un processus distinct, les erreurs affectent moins le reste du système.

Inconvénients

- Performance médiocre : Un nouveau processus est créé pour chaque requête, ce qui est coûteux en ressources.
- Non adapté à une charge élevée : Problèmes de scalabilité dans des environnements à fort trafic.

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

2. CGI Daemon (CGID)

Description

- **CGID** est une variante de CGI introduite dans Apache.
- Fonctionnement :
  - Les scripts CGI sont gérés par un daemon dédié au lieu d'être directement exécutés par le serveur web.
  - Cela réduit la surcharge liée à la création et destruction de processus à chaque requête.

Avantages

- Réduction de la charge serveur par rapport à CGI classique.
- Toujours isolé entre chaque requête.

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

2. CGI Daemon (CGID)

Inconvénients

- Toujours limité en termes de performance par rapport à des solutions modernes comme FastCGI.
- Légèrement plus complexe à configurer que CGI.

Cas d'usage

- Environnements où CGI est nécessaire mais avec une meilleure gestion des ressources.

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

3. FastCGI

Description

- **FastCGI** est une amélioration significative de CGI.
- Fonctionnement :
  - Au lieu de créer un nouveau processus pour chaque requête, un **processus persistant** est créé.
  - Ce processus persistant gère plusieurs requêtes, réduisant la surcharge.

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

3. FastCGI

Avantages

- Performance : Réutilisation des processus permet une gestion beaucoup plus rapide des requêtes.
- Support de plusieurs langages : Fonctionne avec PHP, Python, Ruby, etc.
- Scalabilité : Convient aux sites à fort trafic.
- Séparation : Les scripts peuvent être exécutés sur un serveur distant ou dédié.

Inconvénients

- Configuration plus complexe que CGI.
- Nécessite un module spécifique sur le serveur web (comme `mod_proxy_fcgi` pour Apache).

Cas d'usage

- Applications web dynamiques à fort trafic nécessitant de meilleures performances que CGI classique.

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

4. PHP-FPM (FastCGI Process Manager)

Description

- **PHP-FPM** est une implémentation de FastCGI spécialement conçue pour PHP.
- Fonctionnement :
  - Crée des processus persistants pour traiter les scripts PHP.
  - Offre des fonctionnalités avancées comme la gestion des pools, des limites de ressources, et des logs.

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

4. PHP-FPM (FastCGI Process Manager)

Avantages

- Performance : Plus rapide que CGI ou CGID.
- Gestion des ressources : Vous pouvez configurer des pools distincts avec des limites spécifiques (CPU, RAM) pour chaque application ou domaine.
- Fiabilité : Gère les scripts PHP de manière isolée pour minimiser les effets des erreurs.
- Flexibilité : Compatible avec plusieurs serveurs web (Apache, Nginx, etc.).

Inconvénients

- Limité à PHP.
- Configuration initiale légèrement complexe.

Cas d'usage

- Hébergement d'applications PHP modernes avec des besoins en performances et scalabilité.

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

Comparaison des technologies

Critère	CGI	CGID	FastCGI	PHP-FPM
Performance	Faible	Moyenne	Élevée	Très élevée
Compatibilité	Tous les langages	Tous les langages	Tous les langages	PHP uniquement
Scalabilité	Faible	Moyenne	Bonne	Excellente
Complexité	Simple	Moyenne	Moyenne	Moyenne à élevée
Usage recommandé	Sites très simples	Sites avec faible à modérée charge	Sites dynamiques ou API modernes	Applications PHP performantes

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

Exemple de configuration

FastCGI avec PHP-FPM dans Apache

1. **Activer les modules nécessaires**

```
sudo a2enmod proxy_fcgi setenvif
sudo systemctl restart apache2
```

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

2. **Configurer un Virtual Host**
   Fichier : `/etc/apache2/sites-available/example.conf`

```
<VirtualHost *:80>
    ServerName example.com
    DocumentRoot /var/www/html

    <Directory /var/www/html>
        AllowOverride All
        Require all granted
    </Directory>

    # Utiliser PHP-FPM via FastCGI
    <FilesMatch \.php$>
        SetHandler "proxy:unix:/run/php/php7.4-fpm.sock|fcgi://localhost"
    </FilesMatch>

    ErrorLog ${APACHE_LOG_DIR}/example-error.log
    CustomLog ${APACHE_LOG_DIR}/example-access.log combined
</VirtualHost>
```

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

3. **Redémarrer les services**

```
sudo systemctl restart php7.4-fpm
sudo systemctl reload apache2
```

4. **Tester**
   - Créez un fichier `info.php` :

```
<?php phpinfo(); ?>
```

   - Accédez à `http://example.com/info.php` et vérifiez que PHP fonctionne via FastCGI.

Héberger des applications PHP

CGI, CGID, Fast CGI et PHP-FPM

Résumé

1. **CGI** : Simple mais obsolète, adapté à des environnements avec peu de trafic.
2. **CGID** : Une légère amélioration de CGI pour des cas spécifiques.
3. **FastCGI** : Une solution moderne et performante pour exécuter des scripts dynamiques.
4. **PHP-FPM** : Implémentation de FastCGI optimisée pour PHP, offrant des performances et une scalabilité excellentes.

Héberger des applications PHP

Droits et identité dédiée, sessions

Lors de l'hébergement d'applications PHP avec Apache et PHP-FPM, il est important de :

1. Configurer des **droits et identités dédiées** pour isoler les applications.
2. Gérer les **sessions PHP** pour assurer la sécurité et le bon fonctionnement des utilisateurs.

Héberger des applications PHP

Droits et identité dédiée, sessions

1. Droits et identités dédiées

Chaque application hébergée doit fonctionner sous une identité utilisateur différente pour :

- Isoler les applications (éviter qu'une application compromise n'accède aux données d'une autre).
- Améliorer la sécurité globale du serveur.

Étapes pour configurer des identités dédiées

a. Créer des utilisateurs système pour chaque application

1. Créez un utilisateur système pour chaque application :

```
sudo adduser --system --no-create-home --group phpapp1
sudo adduser --system --no-create-home --group phpapp2
```

Héberger des applications PHP

Droits et identité dédiée, sessions

1. Droits et identités dédiées

Étapes pour configurer des identités dédiées

a. Créer des utilisateurs système pour chaque application

2. Attribuez les permissions des répertoires des applications :
   - Application 1 :

```
sudo chown -R phpapp1:phpapp1 /var/www/phpapp1
```

   - Application 2 :

```
sudo chown -R phpapp2:phpapp2 /var/www/phpapp2
```

Héberger des applications PHP

Droits et identité dédiée, sessions

1. Droits et identités dédiées

Étapes pour configurer des identités dédiées

b. Configurer PHP-FPM avec des pools dédiés

1. Modifiez le fichier de configuration des pools PHP-FPM pour chaque application :

- **Pour l'application 1** : `/etc/php/7.4/fpm/pool.d/phpapp1.conf`

```
[phpapp1]
user = phpapp1
group = phpapp1
listen = /run/php/phpapp1.sock
listen.owner = www-data
listen.group = www-data
listen.mode = 0660
```

Héberger des applications PHP

Droits et identité dédiée, sessions

1. Droits et identités dédiées

Étapes pour configurer des identités dédiées

b. Configurer PHP-FPM avec des pools dédiés

1. Modifiez le fichier de configuration des pools PHP-FPM pour chaque application :

- **Pour l'application 2** : `/etc/php/8.1/fpm/pool.d/phpapp2.conf`

```
[phpapp2]
user = phpapp2
group = phpapp2
listen = /run/php/phpapp2.sock
listen.owner = www-data
listen.group = www-data
listen.mode = 0660
```

Héberger des applications PHP

Droits et identité dédiée, sessions

1. Droits et identités dédiées

Étapes pour configurer des identités dédiées

b. Configurer PHP-FPM avec des pools dédiés

1. Modifiez le fichier de configuration des pools PHP-FPM pour chaque application :

- **Pour l'application 2** : `/etc/php/8.1/fpm/pool.d/phpapp2.conf`

```
[phpapp2]
user = phpapp2
group = phpapp2
listen = /run/php/phpapp2.sock
listen.owner = www-data
listen.group = www-data
listen.mode = 0660
```

2. Redémarrez PHP-FPM pour appliquer les changements :

```
sudo systemctl restart php7.4-fpm
sudo systemctl restart php8.1-fpm
```

Héberger des applications PHP

Droits et identité dédiée, sessions

1. Droits et identités dédiées

Étapes pour configurer des identités dédiées

c. Configurer Apache pour utiliser les pools dédiés

Modifiez les Virtual Hosts pour chaque application afin d'utiliser les sockets spécifiques :

- **Application 1** : `/etc/apache2/sites-available/phpapp1.conf`

```
<VirtualHost *:80>
    ServerName phpapp1.example.com
    DocumentRoot /var/www/phpapp1

    <Directory /var/www/phpapp1>
        AllowOverride All
        Require all granted
    </Directory>

    <FilesMatch \.php$>
        SetHandler "proxy:unix:/run/php/phpapp1.sock|fcgi://localhost"
    </FilesMatch>
</VirtualHost>
```

Héberger des applications PHP

Droits et identité dédiée, sessions

1. Droits et identités dédiées

Étapes pour configurer des identités dédiées

c. Configurer Apache pour utiliser les pools dédiés

Modifiez les Virtual Hosts pour chaque application afin d'utiliser les sockets spécifiques :

- **Application 2** : `/etc/apache2/sites-available/phpapp2.conf`

```
<VirtualHost *:80>
    ServerName phpapp2.example.com
    DocumentRoot /var/www/phpapp2

    <Directory /var/www/phpapp2>
        AllowOverride All
        Require all granted
    </Directory>

    <FilesMatch \.php$>
        SetHandler "proxy:unix:/run/php/phpapp2.sock|fcgi://localhost"
    </FilesMatch>
</VirtualHost>
```

Héberger des applications PHP

Droits et identité dédiée, sessions

1. Droits et identités dédiées

Étapes pour configurer des identités dédiées

c. Configurer Apache pour utiliser les pools dédiés

Activez les Virtual Hosts et redémarrez Apache :

```
sudo a2ensite phpapp1.conf phpapp2.conf
sudo systemctl reload apache2
```

Héberger des applications PHP

Droits et identité dédiée, sessions

2. Gérer les sessions PHP

Les sessions sont une fonctionnalité clé pour les applications web dynamiques. Une mauvaise gestion des sessions peut exposer les utilisateurs à des risques (vol de session, accès non autorisé).

a. Configurer un répertoire de sessions dédié pour chaque application

1. Créez des répertoires distincts pour stocker les fichiers de session :

```
sudo mkdir -p /var/lib/php/sessions/phpapp1
sudo mkdir -p /var/lib/php/sessions/phpapp2
```

2. Attribuez les permissions appropriées :

```
sudo chown -R phpapp1:phpapp1 /var/lib/php/sessions/phpapp1
sudo chown -R phpapp2:phpapp2 /var/lib/php/sessions/phpapp2
sudo chmod 700 /var/lib/php/sessions/phpapp1 /var/lib/php/sessions/phpapp2
```

Héberger des applications PHP

Droits et identité dédiée, sessions

2. Gérer les sessions PHP

Les sessions sont une fonctionnalité clé pour les applications web dynamiques. Une mauvaise gestion des sessions peut exposer les utilisateurs à des risques (vol de session, accès non autorisé).

b. Configurer PHP-FPM pour utiliser ces répertoires

- **Pour l'application 1** :
  Modifiez `/etc/php/7.4/fpm/pool.d/phpapp1.conf` :

```
php_value[session.save_path] = /var/lib/php/sessions/phpapp1
```

- **Pour l'application 2** :
  Modifiez `/etc/php/8.1/fpm/pool.d/phpapp2.conf` :

```
php_value[session.save_path] = /var/lib/php/sessions/phpapp2
```

Héberger des applications PHP

Droits et identité dédiée, sessions

3. Sécurisation supplémentaire des sessions

a. Configurer des options sécurisées pour les sessions PHP

Ajoutez ces directives dans les fichiers de configuration PHP (`php.ini`) :

```
session.cookie_httponly = 1      ; Empêche l'accès aux cookies via JavaScript.
session.cookie_secure = 1        ; Force l'utilisation de HTTPS pour les cookies.
session.use_strict_mode = 1      ; Rejette les sessions non valides.
session.use_only_cookies = 1     ; Interdit l'utilisation d'URL contenant des identifiants de session.
```

Héberger des applications PHP

Droits et identité dédiée, sessions

3. Sécurisation supplémentaire des sessions

b. Utiliser une session basée sur une base de données

Pour une application complexe, vous pouvez stocker les sessions dans une base de données sécurisée. Cela facilite la gestion et améliore la résilience des sessions.

Héberger des applications PHP

Droits et identité dédiée, sessions

Résumé de la configuration

1. **Identités dédiées** :
   - Chaque application a son propre utilisateur et groupe système.
   - Chaque application fonctionne dans un pool PHP-FPM dédié.
2. **Gestion des sessions** :
   - Les sessions sont stockées dans des répertoires séparés pour éviter toute interférence.
   - Les sessions sont sécurisées avec des options PHP appropriées.
3. **Résultat** :
   - Applications isolées en termes de droits et sessions.
   - Sécurité renforcée pour les utilisateurs et données.

Héberger des applications PHP

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Contrôle d'accès et authentification

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

Héberger des applications PHP et faire cohabiter PHP 5 et PHP 7 sur le même serveur Apache peut être utile lorsque vous devez prendre en charge des applications héritées utilisant PHP 5 tout en supportant des applications modernes développées avec PHP 7 ou une version ultérieure.

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

1. Introduction aux modules `mod_authz*`

Les modules `mod_authz_*` sont utilisés pour définir les règles d'autorisation dans Apache HTTPD. Ils remplacent les anciennes directives (`Allow`, `Deny`, `Order`) utilisées dans les versions antérieures à 2.4.

Module	Description
`mod_authz_host`	Autorisation basée sur des adresses IP ou des noms d'hôte.
`mod_authz_user`	Autorisation basée sur des utilisateurs définis.
`mod_authz_groupfile`	Autorisation basée sur des groupes d'utilisateurs définis dans un fichier.
`mod_authz_core`	Base pour toutes les règles d'autorisation, fournit les directives `Require`.
`mod_authz_owner`	Autorisation basée sur le propriétaire du fichier.

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

2. Syntaxe des règles d'autorisation dans Apache 2.4

Directive principale : `Require`

La directive `Require` détermine qui est autorisé à accéder à une ressource.

Syntaxe de base :

```
Require <type> <value>
```

- **`<type>`** : Spécifie le type de contrôle (utilisateur, IP, etc.).
- **`<value>`** : Les valeurs autorisées pour ce type.

Exemple :

```
Require ip 192.168.1.0/24
Require user admin
```

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

3. Contrôle d'accès avec `mod_authz_host`

Le module `mod_authz_host` contrôle l'accès en fonction des adresses IP ou des noms d'hôte.

Directives disponibles :

Directive	Description
`Require ip`	Autorise l'accès aux adresses IP spécifiées.
`Require host`	Autorise l'accès aux noms d'hôte spécifiés.

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

Exemples :

- **Autoriser uniquement une adresse IP spécifique** :

```
<Directory "/var/www/secure">
    Require ip 192.168.1.100
</Directory>
```

- **Autoriser une plage d'adresses IP** :

```
<Directory "/var/www/secure">
    Require ip 192.168.1.0/24
</Directory>
```

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

Exemples :

- **Autoriser un nom d'hôte** :

```
<Directory "/var/www/secure">
    Require host example.com
</Directory>
```

- **Interdire tout accès sauf une IP** :

```
<Directory "/var/www/secure">
    Require all denied
    Require ip 192.168.1.100
</Directory>
```

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

4. Contrôle d'accès avec `mod_authz_user`

Le module `mod_authz_user` permet de restreindre l'accès en fonction des utilisateurs authentifiés.

Directive disponible :

Directive	Description
`Require user`	Autorise l'accès uniquement aux utilisateurs spécifiés.

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

Exemples :

- **Autoriser un utilisateur spécifique** :

```
<Directory "/var/www/secure">
    Require user admin
</Directory>
```

- **Autoriser plusieurs utilisateurs** :

```
<Directory "/var/www/secure">
    Require user admin john alice
</Directory>
```

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

5. Contrôle d'accès avec `mod_authz_groupfile`

Le module `mod_authz_groupfile` gère l'accès basé sur des groupes d'utilisateurs définis dans un fichier.

Configuration :

1. **Créer un fichier de groupes** :
   Fichier : `/etc/apache2/groups`

```
admin: john alice
developers: mike sara
```

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

5. Contrôle d'accès avec `mod_authz_groupfile`

Configuration :

2. **Configurer Apache** :
   - Ajouter la directive pour spécifier le fichier des groupes :

```
AuthGroupFile /etc/apache2/groups
```

   - Autoriser un groupe spécifique :

```
<Directory "/var/www/secure">
    Require group admin
</Directory>
```

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

6. Contrôle d'accès avancé avec `mod_authz_core`

Le module `mod_authz_core` introduit des conditions logiques pour combiner des règles d'autorisation.

Opérateurs disponibles :

Opérateur	Description
`RequireAll`	Toutes les conditions doivent être vraies.
`RequireAny`	Au moins une des conditions doit être vraie.
`RequireNone`	Aucune des conditions ne doit être vraie.

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

6. Contrôle d'accès avancé avec `mod_authz_core`

Exemples :

- **Combinaison avec `RequireAll` :**
  - Autorise uniquement l'utilisateur `admin` depuis une IP spécifique.

```
<Directory "/var/www/secure">
    <RequireAll>
        Require user admin
        Require ip 192.168.1.100
    </RequireAll>
</Directory>
```

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

6. Contrôle d'accès avancé avec `mod_authz_core`

Exemples :

- **Combinaison avec `RequireAny` :**
  - Autorise soit l'utilisateur `admin`, soit les connexions depuis un réseau spécifique.

```
<Directory "/var/www/secure">
    <RequireAny>
        Require user admin
        Require ip 192.168.1.0/24
    </RequireAny>
</Directory>
```

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

6. Contrôle d'accès avancé avec `mod_authz_core`

Exemples :

- **Interdire tout sauf une IP spécifique** :

```
<Directory "/var/www/secure">
    <RequireNone>
        Require ip 10.0.0.0/8
    </RequireNone>
    Require ip 192.168.1.100
</Directory>
```

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

7. Meilleures pratiques

1. **Toujours tester la configuration** :
   Avant de recharger Apache, vérifiez la syntaxe :

   ```
   apachectl configtest
   ```

2. **Documenter les règles** :
   - Incluez des commentaires dans votre configuration pour expliquer les règles d'accès.
3. **Combiner avec SSL/TLS** :
   - Utilisez `mod_ssl` pour sécuriser les connexions lors de l'utilisation de `mod_authz_user` ou `mod_authz_groupfile`.
4. **Minimiser l'accès par défaut** :
   - Utilisez `Require all denied` comme base et ajoutez des exceptions pour limiter l'exposition.

Contrôle d'accès et authentification

Contrôle d'accès et authentification avec Apache : Modules mod_authz

8. Exemple complet : Contrôle d'accès avancé

Voici un exemple combinant plusieurs modules `mod_authz_*` :

```
<VirtualHost *:443>
    ServerName secure.example.com
    DocumentRoot "/var/www/secure"

    SSLEngine On
    SSLCertificateFile "/etc/ssl/certs/example.com.crt"
    SSLCertificateKeyFile "/etc/ssl/private/example.com.key"

    <Directory "/var/www/secure">
        # Interdire tout accès par défaut
        Require all denied

        # Autoriser les utilisateurs du groupe 'admin' depuis le réseau local
        <RequireAll>
            Require group admin
            Require ip 192.168.1.0/24
        </RequireAll>

        # Autoriser l'utilisateur 'john' quelle que soit son IP
        RequireAny user john
    </Directory>

    ErrorLog /var/log/apache2/secure-error.log
    CustomLog /var/log/apache2/secure-access.log combined
</VirtualHost>
```

Contrôle d'accès et authentification

Authentification LDAP avec `mod_authnz_ldap` dans Apache HTTPD

- Le module mod_authnz_ldap permet à Apache HTTPD d'authentifier les utilisateurs via un annuaire LDAP (Lightweight Directory Access Protocol).
- Cela est souvent utilisé pour intégrer un serveur web dans un environnement d'entreprise utilisant des annuaires comme Active Directory ou OpenLDAP.

Contrôle d'accès et authentification

Authentification LDAP avec `mod_authnz_ldap` dans Apache HTTPD

1. Prérequis

1. **Modules Apache nécessaires** :
   - `mod_ldap`
   - `mod_authnz_ldap`

   Vérifiez leur disponibilité et activez-les :

```
sudo a2enmod ldap authnz_ldap
sudo systemctl restart apache2
```

Contrôle d'accès et authentification

Authentification LDAP avec `mod_authnz_ldap` dans Apache HTTPD

2. **Serveur LDAP** :
   - Un serveur LDAP ou Active Directory doit être configuré et accessible.
3. **Informations LDAP nécessaires** :
   - URL du serveur LDAP (ex. : `ldap://ldap.example.com` ou `ldaps://ldap.example.com` pour une connexion sécurisée).
   - Base DN (ex. : `dc=example,dc=com`).
   - DN de liaison pour les recherches LDAP (ex. : `cn=admin,dc=example,dc=com`).
   - Attributs utilisateur (ex. : `uid` ou `sAMAccountName` dans Active Directory).

Contrôle d'accès et authentification

Authentification LDAP avec `mod_authnz_ldap` dans Apache HTTPD

2. Configurer Apache pour l'authentification LDAP

1. **Modifier le fichier de configuration du Virtual Host**
   Fichier : `/etc/apache2/sites-available/ldap-auth.conf` (Debian/Ubuntu) ou `/etc/httpd/conf.d/ldap-auth.conf` (CentOS/RHEL)

   Exemple de configuration :

```
<VirtualHost *:443>
    ServerName secure.example.com
    DocumentRoot "/var/www/secure"

    SSLEngine On
    SSLCertificateFile "/etc/ssl/certs/example.com.crt"
    SSLCertificateKeyFile "/etc/ssl/private/example.com.key"

    # Configuration LDAP
    <Directory "/var/www/secure">
        AuthType Basic
        AuthName "LDAP Authentication"
        AuthBasicProvider ldap
        AuthLDAPURL "ldap://ldap.example.com:389/dc=example,dc=com?uid?sub?(objectClass=person)"
        AuthLDAPBindDN "cn=admin,dc=example,dc=com"
        AuthLDAPBindPassword "admin_password"

        # Autoriser uniquement les utilisateurs authentifiés
        Require valid-user
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/ldap-auth-error.log
```

Contrôle d'accès et authentification

Authentification LDAP avec `mod_authnz_ldap` dans Apache HTTPD

3. Explications des directives LDAP

Directive	Description
`AuthType`	Définit le type d'authentification (ici, `Basic`).
`AuthName`	Message affiché dans la boîte de dialogue de connexion.
`AuthBasicProvider`	Définit le fournisseur d'authentification (ici, `ldap`).
`AuthLDAPURL`	URL du serveur LDAP, suivie du Base DN, de l'attribut utilisateur, et du filtre d'objets LDAP.
`AuthLDAPBindDN`	DN de l'utilisateur utilisé pour rechercher les entrées LDAP.
`AuthLDAPBindPassword`	Mot de passe du DN de liaison.
`Require valid-user`	Autorise tous les utilisateurs LDAP authentifiés.

Contrôle d'accès et authentification

Authentification LDAP avec `mod_authnz_ldap` dans Apache HTTPD

3. Explications des directives LDAP

Exemple d'URL LDAP dans `AuthLDAPURL` :

- LDAP classique :

```
ldap://ldap.example.com:389/dc=example,dc=com?uid?sub?(objectClass=person)
```

  - `dc=example,dc=com` : Base DN.
  - `uid` : Attribut utilisateur (peut être `sAMAccountName` dans Active Directory).
  - `sub` : Étendue de recherche (sous-arbre complet).
  - `(objectClass=person)` : Filtre optionnel pour restreindre les résultats.

- LDAP sécurisé (LDAPS) :

```
ldaps://ldap.example.com:636/dc=example,dc=com?uid?sub?(objectClass=person)
```

Contrôle d'accès et authentification

Authentification LDAP avec `mod_authnz_ldap` dans Apache HTTPD

3. Explications des directives LDAP

Exemple d'URL LDAP dans `AuthLDAPURL` :

- LDAP classique :

```
ldap://ldap.example.com:389/dc=example,dc=com?uid?sub?(objectClass=person)
```

  - `dc=example,dc=com` : Base DN.
  - `uid` : Attribut utilisateur (peut être `sAMAccountName` dans Active Directory).
  - `sub` : Étendue de recherche (sous-arbre complet).
  - `(objectClass=person)` : Filtre optionnel pour restreindre les résultats.

- LDAP sécurisé (LDAPS) :

```
ldaps://ldap.example.com:636/dc=example,dc=com?uid?sub?(objectClass=person)
```

Contrôle d'accès et authentification

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Redirection, réécriture d'adresses, filtres

Redirection, réécriture d'adresses, filtres

- La redirection consiste à informer le client (navigateur ou autre) que la ressource demandée se trouve à une autre adresse (URL). Cela se fait en renvoyant un code HTTP de redirection.
- La réécriture d'adresses (URL rewriting) modifie l'URL en interne sur le serveur sans que le client en soit informé. Elle est utilisée pour rendre les URLs plus propres ou pour masquer la structure réelle du site.
- Les filtres permettent de transformer ou d'ajouter des éléments aux réponses HTTP avant qu'elles ne soient envoyées au client

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

1. DocumentRoot

Définition :

- `DocumentRoot` est une directive dans Apache qui spécifie le répertoire racine où le serveur web cherche les fichiers à servir pour un domaine ou un sous-domaine donné.

Usage dans un VirtualHost :

- Chaque hôte virtuel (`VirtualHost`) peut avoir son propre `DocumentRoot`.
- Exemple :

```
<VirtualHost *:80>
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot "/var/www/html/example"
    <Directory "/var/www/html/example">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

1. DocumentRoot

- **ServerName** : Nom du domaine principal.
- **ServerAlias** : Autres noms de domaine pointant vers ce même site.
- **DocumentRoot** : Chemin où Apache recherche les fichiers pour ce site.
- **Directory** : Permet de spécifier les permissions et options pour le répertoire associé.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

1. DocumentRoot

Options courantes pour DocumentRoot :

- **Indexes** : Affiche une liste des fichiers si aucun fichier `index` n'est trouvé.
- **FollowSymLinks** : Autorise l'utilisation de liens symboliques.
- **AllowOverride** : Permet d'utiliser un fichier `.htaccess` pour modifier la configuration.

Cas pratique :

Si un fichier `index.html` est placé dans `/var/www/html/example`, il sera servi à l'URL `http://example.com`.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

2. Module mod_alias

Définition :

- `mod_alias` est un module d'Apache qui permet :
  - De créer des **alias** pour mapper une URL à un chemin différent sur le système de fichiers.
  - De configurer des **redirections simples** d'une URL vers une autre.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

2. Module mod_alias

Alias :

- Permet de servir des fichiers depuis un répertoire différent de `DocumentRoot`.
- Syntaxe :

```
Alias /url_path /system_path
```

- Exemple :

```
Alias /images "/var/www/assets/images"
<Directory "/var/www/assets/images">
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

  - **Alias /images** : Lorsque l'utilisateur accède à `http://example.com/images`, Apache sert les fichiers depuis `/var/www/assets/images`.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

2. Module mod_alias

Redirect :

- **Description** : Redirige une URL vers une nouvelle URL.
- **Syntaxe** :

```
Redirect [status] [URL_PATH] [TARGET_URL]
```

  - `status` : Le code HTTP (par défaut : `302`). Peut être `301`, `302`, ou d'autres codes de redirection.
  - `URL_PATH` : Le chemin relatif sur le serveur.
  - `TARGET_URL` : L'URL vers laquelle rediriger.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

2. Module mod_alias

Redirect :

- Redirige une URL vers une autre.
- Syntaxe :

```
Redirect [status_code] /source_path http://destination_url
```

- Exemple :

```
Redirect 301 /old-page.html http://example.com/new-page.html
```

  - Redirige de manière permanente (`301`) les requêtes pour `http://example.com/old-page.html` vers `http://example.com/new-page.html`.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

3. Exemples pratiques avec `mod_alias`

3.3. Redirection vers un autre site

Cas : Rediriger `/blog` vers un blog externe.

```
Redirect 302 /blog https://blog.example.com
```

Effet :

- `http://example.com/blog` → `https://blog.example.com`.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

2. Module mod_alias

RedirectMatch

- **Description** : Redirige les requêtes en utilisant des expressions régulières.
- **Syntaxe** :

```
RedirectMatch [status] REGEX TARGET_URL
```

  - `REGEX` : Expression régulière pour correspondre aux URLs.
  - `TARGET_URL` : L'URL cible.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

2. Module mod_alias

RedirectMatch :

- Redirection conditionnelle utilisant des expressions régulières.
- Exemple :

```
RedirectMatch 403 ^/private
```

  - Toute URL commençant par `/private` retourne une erreur `403 Forbidden`.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

3. Exemples pratiques avec `mod_alias`

3.1. Redirection permanente d'un domaine entier

Cas : Rediriger tout le trafic de `http://example.com` vers `https://newexample.com`.

```
<VirtualHost *:80>
    ServerName example.com
    Redirect 301 / https://newexample.com/
</VirtualHost>
```

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

3. Exemples pratiques avec `mod_alias`

3.2. Redirection d'une page spécifique

Cas : Rediriger `/old-page` vers `/new-page`.

```
Redirect 301 /old-page /new-page
```

Effet :

- `http://example.com/old-page` → `http://example.com/new-page`.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

3. Exemples pratiques avec `mod_alias`

3.3. Redirection vers un autre site

Cas : Rediriger `/blog` vers un blog externe.

```
Redirect 302 /blog https://blog.example.com
```

Effet :

- `http://example.com/blog` → `https://blog.example.com`.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

3. Exemples pratiques avec `mod_alias`

3.4. Redirection basée sur un pattern

Cas : Rediriger toutes les pages terminant par `.html` vers `.php`.

```
RedirectMatch 301 (.*)\.html$ $1.php
```

Effet :

- `http://example.com/page.html` → `http://example.com/page.php`.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

3. Exemples pratiques avec `mod_alias`

3.5. Redirection conditionnelle (par sous-domaine)

Cas : Rediriger `http://shop.example.com` vers une boutique externe.

```
<VirtualHost *:80>
    ServerName shop.example.com
    Redirect 301 / https://shopify.com/shop
</VirtualHost>
```

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

3. Exemples pratiques avec `mod_alias`

3.6. Redirection d'une section vers une nouvelle structure

Cas : Rediriger toutes les pages d'une section `/old-section` vers `/new-section`.

```
RedirectMatch 301 ^/old-section/(.*)$ /new-section/$1
```

Effet :

- `http://example.com/old-section/page` → `http://example.com/new-section/page`.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

4. Cas d'erreurs ou pages spéciales

4.1. Redirection d'une erreur 404 vers une page personnalisée

Cas : Rediriger toutes les erreurs 404 vers une page `/not-found`.

```
ErrorDocument 404 /not-found
Redirect 301 /not-found /custom-404-page
```

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

4. Cas d'erreurs ou pages spéciales

4.2. Blocage d'une URL avec un code d'erreur

Cas : Retourner une erreur 410 (Gone) pour une page supprimée.

```
Redirect 410 /deleted-page
```

Effet :

- Les utilisateurs recevront une erreur `410 Gone` pour cette page.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

5. Cas spécifiques : Gestion de HTTPS avec `mod_alias`

Redirection de HTTP vers HTTPS

Cas : Forcer tout le trafic HTTP à utiliser HTTPS.

```
<VirtualHost *:80>
    ServerName example.com
    Redirect 301 / https://example.com/
</VirtualHost>
```

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

Exemples pratiques

2. Redirection avec mod_alias :

Rediriger tout le contenu d'un ancien site vers un nouveau.

```
<VirtualHost *:80>
    ServerName oldsite.com
    Redirect 301 / http://newsite.com/
</VirtualHost>
```

- Toutes les requêtes vers `oldsite.com` sont redirigées vers `newsite.com`.

Redirection, réécriture d'adresses, filtres

DocumentRoot et le module mod_alias

Exemples pratiques

3. Alias pour des applications spécifiques :

Vous hébergez plusieurs applications sur le même serveur :

- Application principale : `/var/www/html`.
- Interface d'administration : `/var/www/admin`.

Configuration :

```
<VirtualHost *:80>
    ServerName example.com
    DocumentRoot "/var/www/html"

    Alias /admin "/var/www/admin"
    <Directory "/var/www/admin">
        Options FollowSymLinks
        Require all granted
    </Directory>
</VirtualHost>
```

Redirection, réécriture d'adresses, filtres

Notion de Répertoire Virtuel et d'Alias dans Apache

2. Alias

Définition :

Un **alias** est une directive utilisée pour définir un chemin alternatif pour les ressources sur le serveur.

Redirection, réécriture d'adresses, filtres

Notion de Répertoire Virtuel et d'Alias dans Apache

2. Alias

Cas d'Usage des Alias :

1. **Organiser les Ressources** :
   - Si les images, CSS et JavaScript sont stockés dans des répertoires différents, vous pouvez les servir via des chemins d'alias pour plus de clarté.

Exemple :

```
Alias /images "/var/www/media/images"
Alias /css "/var/www/media/styles"
Alias /js "/var/www/media/scripts"
<Directory "/var/www/media">
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

Redirection, réécriture d'adresses, filtres

Notion de Répertoire Virtuel et d'Alias dans Apache

2. Alias

Cas d'Usage des Alias :

1. **Organiser les Ressources** :
   - Si les images, CSS et JavaScript sont stockés dans des répertoires différents, vous pouvez les servir via des chemins d'alias pour plus de clarté.

Exemple :

```
Alias /images "/var/www/media/images"
Alias /css "/var/www/media/styles"
Alias /js "/var/www/media/scripts"
<Directory "/var/www/media">
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

Résultat :

- Accéder à `http://example.com/images` sert les fichiers de `/var/www/media/images`.
- Accéder à `http://example.com/css` sert les fichiers de `/var/www/media/styles`.

Redirection, réécriture d'adresses, filtres

Notion de Répertoire Virtuel et d'Alias dans Apache

2. Alias

Cas d'Usage des Alias :

2. **Héberger des Applications Séparées** :
   - Si vous avez une application d'administration et une application principale sur le même serveur.

   Exemple :

```
Alias /admin "/var/www/apps/admin"
Alias /app "/var/www/apps/main"
<Directory "/var/www/apps">
    Options Indexes FollowSymLinks
    Require all granted
</Directory>
```

   Résultat :
   - L'URL `http://example.com/admin` accède aux fichiers de `/var/www/apps/admin`.
   - L'URL `http://example.com/app` accède aux fichiers de `/var/www/apps/main`.
3. **Servir des Fichiers Hors de `DocumentRoot`** :
   - Les alias permettent de servir des fichiers qui ne sont pas situés dans le répertoire défini par `DocumentRoot`.

Redirection, réécriture d'adresses, filtres

Notion de Répertoire Virtuel et d'Alias dans Apache

3. Redirection vs Alias

Bien que les deux modifient l'interprétation des URL, ils ont des objectifs différents :

- **Alias** : Mappe une URL à un chemin physique sur le système de fichiers.
- **Redirect** : Envoie une réponse HTTP (comme une redirection 301) pour pointer vers une autre URL.

Exemple de Redirection :

```
Redirect 301 /old-url http://example.com/new-url
```

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

Le module `mod_rewrite` d'Apache est un outil puissant permettant de manipuler les URL pour :

- Les rendre plus lisibles.
- Mettre en place des redirections dynamiques.
- Masquer la structure interne du site Web.

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

1. Activation du Module `mod_rewrite`

Avant d'utiliser `mod_rewrite`, vous devez vous assurer qu'il est activé :

```
sudo a2enmod rewrite
sudo systemctl restart apache2
```

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

2. Syntaxe de Base des Règles de Ré-écriture

- Les règles se définissent dans :
  - La configuration du serveur (fichier `.conf`).
  - Un fichier `.htaccess` (si `AllowOverride` est activé).

Directives Principales

1. **RewriteEngine** :
   - Active ou désactive le module.

```
RewriteEngine On
```

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

2. Syntaxe de Base des Règles de Ré-écriture

Directives Principales

2. **RewriteRule** :
   - Définit une règle de ré-écriture pour transformer une URL entrante.

```
RewriteRule Pattern Substitution [Flags]
```

   - **Pattern** : Une expression régulière définissant les URL à réécrire.
   - **Substitution** : L'URL modifiée ou le chemin vers lequel la requête est redirigée.
   - **Flags** : Modificateurs influençant le comportement de la règle.
3. **RewriteCond** :
   - Ajoute des conditions pour qu'une règle soit appliquée.

```
RewriteCond TestString Condition
```

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

3. Exemples Pratiques

3.1 Réécriture Simple

Transformer une URL complexe en une URL plus lisible.

```
RewriteEngine On
RewriteRule ^product/([0-9]+)$ /product.php?id=$1 [L]
```

- **Explication** :
  - `^product/([0-9]+)$` : Correspond aux URL commençant par `product/` et suivies d'un nombre.
  - `/product.php?id=$1` : Redirige la requête vers `product.php` avec le nombre capturé comme paramètre `id`.
  - `[L]` : Indique que cette règle est la dernière si elle correspond.

Exemple :
URL entrée : `http://example.com/product/123`
Apache réécrit : `/product.php?id=123`

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

3.2 Redirection Permanente

Rediriger une ancienne URL vers une nouvelle.

```
RewriteEngine On
RewriteRule ^old-page.html$ /new-page.html [R=301,L]
```

- **Explication** :
  - `[R=301]` : Envoie une redirection HTTP permanente (301).

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

3.3 Forcer l'Utilisation de HTTPS

Rediriger automatiquement tout le trafic HTTP vers HTTPS.

```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule ^(.*)$ https://%{HTTP_HOST}/$1 [R=301,L]
```

- **RewriteCond %{HTTPS} off** : Vérifie si HTTPS n'est pas utilisé.
- **RewriteRule** : Redirige vers l'URL équivalente avec HTTPS.

Exemple :

URL entrée : `http://example.com/page`

Résultat : `https://example.com/page`

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

3.4 Forcer le WWW

Rediriger tout le trafic vers une version avec `www`.

```
RewriteEngine On
RewriteCond %{HTTP_HOST} !^www\.
RewriteRule ^(.*)$ https://www.%{HTTP_HOST}/$1 [R=301,L]
```

- **RewriteCond %{HTTP_HOST} !^www\.** : Vérifie que l'hôte ne commence pas par `www`.
- **RewriteRule** : Ajoute `www.` au début de l'URL.

Exemple :
URL entrée : `http://example.com`
Résultat : `http://www.example.com`

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

3.5 Redirection Basée sur l'Agent Utilisateur

Diriger les utilisateurs mobiles vers une version spécifique du site.

```
RewriteEngine On
RewriteCond %{HTTP_USER_AGENT} "Mobile|Android|iPhone"
RewriteRule ^(.*)$ /mobile/$1 [L]
```

- **RewriteCond %{HTTP_USER_AGENT}** : Vérifie si l'agent utilisateur correspond à des mots-clés mobiles.
- **RewriteRule** : Redirige vers le sous-répertoire `/mobile`.

Exemple :
URL entrée : `http://example.com/page` (via mobile)
Résultat : `http://example.com/mobile/page`

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

5. Flags (`[FLAGS]`)

- **Description** : Modifient le comportement des règles.
- **Syntaxe** : Les flags sont spécifiés entre crochets `[ ]`, séparés par des virgules si multiples.

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

5. Flags (`[FLAGS]`)

Flag	Description	Cas d'utilisation
`L`	Arrête le traitement des règles si celle-ci correspond.	Empêcher des règles supplémentaires d'être appliquées.
`R`	Effectue une redirection. Valeurs possibles : `R=301` (permanente), `R=302`.	Rediriger vers un autre domaine ou protocole.
`P`	Proxy : Passe la requête à un serveur backend.	Utilisé pour les proxys ou passerelles.
`QSA`	Ajoute les paramètres de la requête à la cible (Query String Append).	Conserver les paramètres GET d'une requête.
`NC`	Insensible à la casse (No Case).	Utilisé pour les correspondances insensibles à la casse.

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

Flag	Description	Cas d'utilisation
`NE`	Empêche l'encodage des caractères spéciaux (No Escape).	Utile pour les caractères réservés dans les URL.
`OR`	Combine les conditions avec un **OU** logique (par défaut, c'est `ET`).	Utilisé dans `RewriteCond`.
`T`	Change le type MIME de la réponse.	Servir un fichier avec un type MIME différent.
`G`	Retourne une erreur `410 Gone`.	Indiquer qu'une ressource a été supprimée.
`F`	Retourne une erreur `403 Forbidden`.	Bloquer l'accès à une ressource.
`C`	Lien conditionnel entre plusieurs règles (Chain).	Utilisé pour enchaîner des règles qui doivent être testées ensemble.

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

Flag	Description	Cas d'utilisation
`NE`	Empêche l'encodage des caractères spéciaux (No Escape).	Utile pour les caractères réservés dans les URL.
`OR`	Combine les conditions avec un **OU** logique (par défaut, c'est `ET`).	Utilisé dans `RewriteCond`.
`T`	Change le type MIME de la réponse.	Servir un fichier avec un type MIME différent.
`G`	Retourne une erreur `410 Gone`.	Indiquer qu'une ressource a été supprimée.
`F`	Retourne une erreur `403 Forbidden`.	Bloquer l'accès à une ressource.
`C`	Lien conditionnel entre plusieurs règles (Chain).	Utilisé pour enchaîner des règles qui doivent être testées ensemble.

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

Variables et conditions disponibles dans `RewriteCond`

1. Variables de serveur

Variable	Description
`%{HTTP_HOST}`	Le nom de domaine ou l'adresse IP du serveur.
`%{REQUEST_URI}`	L'URI de la requête.
`%{QUERY_STRING}`	Les paramètres GET.
`%{REMOTE_ADDR}`	Adresse IP de l'utilisateur.
`%{HTTP_USER_AGENT}`	L'agent utilisateur (navigateur).
`%{HTTPS}`	Indique si HTTPS est activé (`on` ou vide).

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

Variables et conditions disponibles dans `RewriteCond`

1. Variables de serveur

Exemple : Rediriger en fonction de l'adresse IP

```
RewriteCond %{REMOTE_ADDR} ^192\.168\.1\.100$
RewriteRule ^(.*)$ /blocked.html [L]
```

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

2. Expressions régulières

- **Syntaxe** : Les patterns utilisent des expressions régulières pour correspondre.
- **Caractères courants** :

Caractère	Signification
`^`	Début de l'URI.
`$`	Fin de l'URI.
`(.*)`	Capturer tout.
`[abc]`	Correspond à `a`, `b` ou `c`.
`\d`	Correspond à un chiffre.
`\w`	Correspond à un mot.

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

3. Conditions temporelles

- **Variables temporelles** :

Variable	Description
`%{TIME_HOUR}`	L'heure actuelle (00-23).
`%{TIME_MIN}`	Les minutes (00-59).
`%{TIME_SEC}`	Les secondes (00-59).

Exemple : Redirection en dehors des heures d'ouverture

```
RewriteCond %{TIME_HOUR} <09 [OR]
RewriteCond %{TIME_HOUR} >17
RewriteRule ^(.*)$ /closed.html [L]
```

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

2. Redirection conditionnelle

Objectif : Rediriger uniquement les utilisateurs mobiles.

```
RewriteCond %{HTTP_USER_AGENT} "Mobile" [NC]
RewriteRule ^(.*)$ /mobile/ [L]
```

3. Blocage d'accès à un fichier

Objectif : Bloquer l'accès à `config.php`.

```
RewriteRule ^config\.php$ - [F]
```

4. Servir une page multilingue

Objectif : Réécrire `/fr/contact` en `contact.php?lang=fr`.

```
RewriteRule ^([a-z]{2})/(.*)$ $2.php?lang=$1 [L,QSA]
```

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

4. Flags Courants dans mod_rewrite

Les flags modifient le comportement des règles. Quelques exemples importants :

4. **QSA (Query String Append)** :
   - Ajoute la chaîne de requête existante à la nouvelle URL.

```
RewriteRule ^search/(.*)$ /search.php?query=$1 [QSA,L]
```

5. **P (Proxy)** :
   - Active le mode proxy, utile pour rediriger vers des serveurs externes.

```
RewriteRule ^api/(.*)$ http://backend.example.com/$1 [P]
```

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

5. Bonnes Pratiques

1. **Tester les Règles** :
   - Utilisez des outils comme Regex101 pour valider vos expressions régulières.
2. **Utilisation du Fichier `.htaccess`** :
   - Activez `AllowOverride All` dans la configuration Apache pour permettre l'utilisation de fichiers `.htaccess` :

```
<Directory /var/www/html>
    AllowOverride All
</Directory>
```

3. **Minimiser le Nombre de Règles** :
   - Trop de règles peut ralentir le serveur.
4. **Débogage** :
   - Activez les journaux de débogage pour vérifier l'exécution des règles.

```
LogLevel alert rewrite:trace3
```

Redirection, réécriture d'adresses, filtres

Les Règles de Ré-écriture d'URL et le Module mod_rewrite dans Apache

5. Bonnes Pratiques

1. **Tester les Règles** :
   - Utilisez des outils comme Regex101 pour valider vos expressions régulières.
2. **Utilisation du Fichier `.htaccess`** :
   - Activez `AllowOverride All` dans la configuration Apache pour permettre l'utilisation de fichiers `.htaccess` :

```
<Directory /var/www/html>
    AllowOverride All
</Directory>
```

3. **Minimiser le Nombre de Règles** :
   - Trop de règles peut ralentir le serveur.
4. **Débogage** :
   - Activez les journaux de débogage pour vérifier l'exécution des règles.

```
LogLevel alert rewrite:trace3
```

Redirection, réécriture d'adresses, filtres

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Reverse Proxy et Cache

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

Apache peut fonctionner comme un **proxy** et un **reverse proxy**, offrant des fonctionnalités avancées pour rediriger, équilibrer et gérer les requêtes entre clients et serveurs backend.

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

1. Concepts de Proxy et Reverse Proxy

Proxy

Un proxy agit comme un intermédiaire entre le client et le serveur. Il est utilisé pour :

- Cacher l'identité du client.
- Filtrer ou journaliser les requêtes sortantes.
- Accélérer les requêtes avec du caching.

Reverse Proxy

Un reverse proxy agit comme un intermédiaire entre les clients et un ou plusieurs serveurs backend. Il est utilisé pour :

- Cacher l'identité des serveurs backend.
- Répartir la charge entre plusieurs serveurs (load balancing).
- Terminer le SSL.
- Fournir un point unique d'accès sécurisé pour les clients.

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

2. Modules Nécessaires

Apache utilise les modules suivants pour le proxy :

- `mod_proxy` : Fournit les fonctionnalités de base de proxy.
- `mod_proxy_http` : Proxy pour les requêtes HTTP.
- `mod_proxy_balancer` : Gestion du load balancing.
- `mod_proxy_fcgi` : Proxy pour les requêtes FastCGI.
- `mod_ssl` : Gestion des connexions HTTPS.

Activer les modules

Pour activer ces modules, utilisez les commandes :

```
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2enmod proxy_balancer
sudo a2enmod ssl
sudo systemctl restart apache2
```

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

3. Configuration de Base : Reverse Proxy HTTP

Objectif

Rediriger les requêtes d'un domaine vers un serveur backend situé à `http://backend.local`.

Configuration

Fichier `/etc/apache2/sites-available/reverse-proxy.conf` :

```
<VirtualHost *:80>
    ServerName example.com

    ProxyPreserveHost On
    ProxyPass / http://backend.local/
    ProxyPassReverse / http://backend.local/

    ErrorLog ${APACHE_LOG_DIR}/reverse_proxy_error.log
    CustomLog ${APACHE_LOG_DIR}/reverse_proxy_access.log combined
</VirtualHost>
```

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

3. Configuration de Base : Reverse Proxy HTTP

Objectif

Rediriger les requêtes d'un domaine vers un serveur backend situé à `http://backend.local`.

Explication

- **`ProxyPreserveHost On`** : Préserve l'en-tête `Host` de la requête d'origine.
- **`ProxyPass`** : Redirige les requêtes entrantes vers `http://backend.local/`.
- **`ProxyPassReverse`** : Modifie les en-têtes de réponse pour correspondre à l'URL du client.

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

4. Reverse Proxy avec HTTPS

Objectif

Proxy HTTPS vers un backend sécurisé situé à `https://secure-backend.local`.

Configuration

Fichier `/etc/apache2/sites-available/secure-reverse-proxy.conf` :

```
<VirtualHost *:443>
    ServerName secure.example.com

    SSLEngine On
    SSLCertificateFile /etc/ssl/certs/example.com.crt
    SSLCertificateKeyFile /etc/ssl/private/example.com.key

    ProxyPreserveHost On
    ProxyPass / https://secure-backend.local/
    ProxyPassReverse / https://secure-backend.local/

    ErrorLog ${APACHE_LOG_DIR}/secure_reverse_proxy_error.log
    CustomLog ${APACHE_LOG_DIR}/secure_reverse_proxy_access.log combined
</VirtualHost>
```

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

4. Reverse Proxy avec HTTPS

Objectif

Proxy HTTPS vers un backend sécurisé situé à `https://secure-backend.local`.

Explication

- **SSL** : Termination SSL sur Apache avec des certificats locaux.
- Le reverse proxy gère les connexions sécurisées.

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

5. Load Balancing avec Reverse Proxy

Objectif

Distribuer les requêtes entre plusieurs serveurs backend pour équilibrer la charge.

Configuration

Fichier `/etc/apache2/sites-available/load-balancer.conf` :

```
<Proxy "balancer://mycluster">
    BalancerMember http://backend1.local
    BalancerMember http://backend2.local
    ProxySet lbmethod=byrequests
</Proxy>

<VirtualHost *:80>
    ServerName example.com

    ProxyPreserveHost On
    ProxyPass / balancer://mycluster/
    ProxyPassReverse / balancer://mycluster/

    ErrorLog ${APACHE_LOG_DIR}/load_balancer_error.log
    CustomLog ${APACHE_LOG_DIR}/load_balancer_access.log combined
</VirtualHost>
```

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

5. Load Balancing avec Reverse Proxy

Objectif

Distribuer les requêtes entre plusieurs serveurs backend pour équilibrer la charge.

Explication

- `BalancerMember` : Déclare les serveurs backend.
- `ProxySet lbmethod=byrequests` : Utilise une méthode d'équilibrage basée sur le nombre de requêtes.
- Méthodes disponibles :
  - `byrequests` : Répartit en fonction du nombre de requêtes.
  - `bytraffic` : Répartit en fonction du trafic.

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

6. Reverse Proxy pour WebSocket

Objectif

Supporter les connexions WebSocket.

Configuration

```
<VirtualHost *:80>
    ServerName websocket.example.com

    ProxyPreserveHost On
    ProxyPass /ws/ ws://backend.local:8080/
    ProxyPassReverse /ws/ ws://backend.local:8080/

    ErrorLog ${APACHE_LOG_DIR}/websocket_proxy_error.log
    CustomLog ${APACHE_LOG_DIR}/websocket_proxy_access.log combined
</VirtualHost>
```

Explication

- `ws://` : Protocole WebSocket.
- `/ws/` : Redirige uniquement les requêtes vers WebSocket.

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

7. Gestion du Cache Proxy

Apache peut mettre en cache les réponses backend pour améliorer les performances.

Configuration

Ajoutez le cache dans le VirtualHost :

```
CacheQuickHandler off
CacheLock on
CacheRoot /var/cache/apache2/proxy
CacheEnable disk /
CacheIgnoreNoLastMod On
CacheIgnoreCacheControl On
```

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

8. Sécurisation du Reverse Proxy

1. Filtrage des Requêtes

Bloquez les requêtes indésirables :

```
<Proxy "http://backend.local">
    Require ip 192.168.1.0/24
</Proxy>
```

2. Limitation de la Taille

Limitez la taille des requêtes :

```
LimitRequestBody 10485760
```

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

9. Journaux pour Debugging

Activez les journaux de débogage pour le proxy :

```
LogLevel proxy:trace2
```

Consultez les logs Apache :

```
sudo tail -f /var/log/apache2/proxy.log
```

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

Cas Pratiques et Résultats

1. **Simple Reverse Proxy** :
   - Client : `http://example.com/page`
   - Backend : `http://backend.local/page`
2. **HTTPS Proxy** :
   - Client : `https://secure.example.com/api`
   - Backend : `https://secure-backend.local/api`
3. **Load Balancer** :
   - Client : `http://example.com/`
   - Backend 1 : `http://backend1.local/`
   - Backend 2 : `http://backend2.local/`
4. **WebSocket** :
   - Client : `ws://websocket.example.com/ws`
   - Backend : `ws://backend.local:8080/`

Reverse Proxy et Cache

Les fonctionnalités de proxy et de Reverse Proxy

Résumé

- Apache peut être configuré comme **proxy** et **reverse proxy** pour gérer des scénarios complexes.
- Les modules nécessaires incluent `mod_proxy`, `mod_proxy_http`, et `mod_proxy_balancer`.
- Les fonctionnalités incluent le routage HTTP/HTTPS, l'équilibrage de charge, le support WebSocket, et le caching.
- La sécurité et les performances peuvent être optimisées via des filtres et le caching.

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

Dans une architecture où plusieurs serveurs backend sont utilisés pour traiter les requêtes des utilisateurs, la **répartition de charge** (load balancing) et l'**affinité de session** (session stickiness) jouent un rôle clé pour garantir à la fois les performances et la cohérence des sessions.

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

1. Répartition de Charge

Définition

La répartition de charge consiste à distribuer les requêtes des utilisateurs entre plusieurs serveurs backend pour :

- **Améliorer les performances** : Éviter la surcharge d'un seul serveur.
- **Augmenter la disponibilité** : Si un serveur est défaillant, les autres prennent le relais.
- **Assurer l'évolutivité** : Ajouter des serveurs backend au cluster en fonction de la demande.

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

1. Répartition de Charge

Méthodes de Répartition de Charge dans Apache

Apache prend en charge plusieurs méthodes via le module `mod_proxy_balancer` :

Méthode	Description
`byrequests`	Équilibre le nombre de requêtes en les répartissant de manière égale entre les serveurs.
`bytraffic`	Équilibre en fonction de la quantité de trafic (en octets) envoyée à chaque serveur.
`bybusyness`	Envoie les requêtes aux serveurs les moins occupés.
`heartbeat`	Utilise des informations de santé des serveurs backend (requiert le module `mod_heartbeat`).

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

1. Répartition de Charge

Configuration de Base

Un exemple de configuration de répartition de charge avec `byrequests` :

```
<Proxy "balancer://mycluster">
    BalancerMember http://backend1.local
    BalancerMember http://backend2.local
    ProxySet lbmethod=byrequests
</Proxy>

<VirtualHost *:80>
    ServerName example.com

    ProxyPreserveHost On
    ProxyPass / balancer://mycluster/
    ProxyPassReverse / balancer://mycluster/

    ErrorLog ${APACHE_LOG_DIR}/load_balancer_error.log
    CustomLog ${APACHE_LOG_DIR}/load_balancer_access.log combined
</VirtualHost>
```

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

2. Affinité de Session (Session Stickiness)

Définition

L'affinité de session est une stratégie permettant de garantir qu'un utilisateur est toujours redirigé vers le même serveur backend pour une session donnée. Cela est essentiel lorsque :

- Les sessions utilisateur sont stockées en mémoire locale sur les serveurs backend.
- Une requête sur un autre serveur pourrait entraîner une perte de session.

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

2. Affinité de Session (Session Stickiness)

Mécanismes de l'Affinité de Session

Apache supporte l'affinité de session avec :

1. **Cookies de Session** :
   - Apache peut injecter un cookie unique pour identifier le serveur backend correspondant.
   - Le cookie est envoyé au client, qui le renvoie dans les requêtes suivantes.
2. **Paramètres d'URL** :
   - L'affinité est basée sur un identifiant stocké dans l'URL.
3. **Adresse IP Source** :
   - Apache dirige les requêtes provenant de la même IP vers le même backend.

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

2. Affinité de Session (Session Stickiness)

Configuration avec un Cookie

Ajoutez l'option `stickysession` pour utiliser les cookies :

```
<Proxy "balancer://mycluster">
    BalancerMember http://backend1.local
    BalancerMember http://backend2.local
    ProxySet lbmethod=byrequests stickysession=JSESSIONID
</Proxy>

<VirtualHost *:80>
    ServerName example.com

    ProxyPreserveHost On
    ProxyPass / balancer://mycluster/
    ProxyPassReverse / balancer://mycluster/

    ErrorLog ${APACHE_LOG_DIR}/load_balancer_error.log
    CustomLog ${APACHE_LOG_DIR}/load_balancer_access.log combined
</VirtualHost>
```

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

2. Affinité de Session (Session Stickiness)

Configuration avec un Cookie

- `stickysession=JSESSIONID` : Utilise le cookie `JSESSIONID` pour l'affinité.
- Si le cookie `JSESSIONID` n'est pas trouvé, Apache attribue un backend en fonction de la méthode de répartition de charge définie.

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

3. Avantages et Inconvénients

Avantages de l'Affinité de Session

1. **Simplicité** : Maintient les sessions utilisateur cohérentes sans changer l'infrastructure existante.
2. **Compatibilité** : Fonctionne bien avec des applications web classiques.
3. **Évolutivité Modérée** : Peut être combinée avec des stratégies de répartition.

Inconvénients

1. **Risque de Déséquilibre** : Certains serveurs peuvent être surchargés si les sessions ne sont pas distribuées équitablement.
2. **Tolérance aux Pannes** :
   - Si un serveur backend tombe en panne, les sessions associées sont perdues.
3. **Complexité de Mise à l'Échelle** :
   - Si les sessions sont liées à un serveur spécifique, il est difficile d'ajouter ou de retirer des serveurs

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

4. Améliorations Possibles

1. **Stockage de Session Centralisé** :
   - Utilisez un stockage partagé (par exemple, Redis, Memcached, ou une base de données) pour conserver les sessions.
   - Cela élimine le besoin d'affinité stricte.
2. **Réplication de Session** :
   - Configurez les serveurs backend pour répliquer les sessions entre eux.
   - Permet à tout serveur de traiter les requêtes.
3. **Monitoring des Backends** :
   - Utilisez le module `mod_status` pour surveiller la charge et l'état des backends.
   - Activez un système d'alerte pour détecter les surcharges ou les pannes.

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

5. Surveillance et Debugging

Surveiller les Backends

Activez le tableau de bord du balancer :

```
<Location "/balancer-manager">
    SetHandler balancer-manager
    Require ip 192.168.1.0/24
</Location>
```

Accédez à `http://example.com/balancer-manager` pour visualiser et ajuster manuellement les serveurs.

Augmenter la Verbosité des Logs

Ajoutez des logs détaillés pour déboguer la répartition de charge :

```
LogLevel proxy:trace2
```

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

6. Cas Pratique : API avec Load Balancer et Affinité

Supposons une API où les sessions utilisateur sont critiques.

Configuration

```
<Proxy "balancer://apicluster">
    BalancerMember http://api1.local route=api1
    BalancerMember http://api2.local route=api2
    ProxySet lbmethod=byrequests stickysession=SESSIONID
</Proxy>

<VirtualHost *:80>
    ServerName api.example.com

    ProxyPreserveHost On
    ProxyPass /api balancer://apicluster/
    ProxyPassReverse /api balancer://apicluster/

    ErrorLog ${APACHE_LOG_DIR}/api_error.log
    CustomLog ${APACHE_LOG_DIR}/api_access.log combined
</VirtualHost>
```

Reverse Proxy et Cache

Répartition de Charge et Affinité de Session avec Apache

6. Cas Pratique : API avec Load Balancer et Affinité

Supposons une API où les sessions utilisateur sont critiques.

Configuration

- Les requêtes incluent un cookie `SESSIONID` pour l'affinité.
- Les backends `api1.local` et `api2.local` sont équilibrés selon le nombre de requêtes.

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

Dans une architecture distribuée, Apache HTTPD peut gérer la disponibilité et la résilience des serveurs backend en surveillant leur état, en redirigeant le trafic en cas de panne (failover), et en utilisant un système de heartbeat pour vérifier la santé des serveurs.

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

1. Concepts Clés

Gestion de l'État des Serveurs

Apache peut surveiller l'état des serveurs backend pour déterminer :

- Si un serveur est en ligne ou hors ligne.
- La charge actuelle d'un serveur.

Failover

Le **failover** consiste à rediriger automatiquement les requêtes vers un serveur secondaire en cas de défaillance du serveur principal. Apache supporte cela nativement via `mod_proxy_balancer`.

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

1. Concepts Clés

Heartbeat

Le module `mod_heartbeat` permet à Apache de surveiller en temps réel l'état des serveurs backend à l'aide de messages de heartbeat (battements de cœur), qui indiquent si un serveur est opérationnel.

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

2. Modules Requis

Pour activer ces fonctionnalités, les modules suivants sont nécessaires :

- `mod_proxy` : Proxy et reverse proxy.
- `mod_proxy_balancer` : Load balancing et gestion de l'état des serveurs.
- `mod_heartbeat` : Surveillance de la santé des serveurs backend.
- `mod_lbmethod_heartbeat` : Méthode d'équilibrage basée sur le heartbeat.

Activer les Modules

```
sudo a2enmod proxy proxy_balancer heartbeat lbmethod_heartbeat
sudo systemctl restart apache2
```

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

3. Configuration : Gestion de l'État des Serveurs

1. Surveiller l'État des Serveurs avec le Balancer Manager

Ajoutez une interface de gestion pour surveiller les serveurs backend :

```
<Location "/balancer-manager">
    SetHandler balancer-manager
    Require ip 192.168.1.0/24
</Location>
```

Accédez à `http://example.com/balancer-manager` pour visualiser :

- L'état des serveurs backend.
- Les statistiques de charge.
- La possibilité de marquer un serveur comme "hors ligne".

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

3. Configuration : Gestion de l'État des Serveurs

2. Détection Automatique des Pannes

Apache détecte automatiquement les serveurs non réactifs et les désactive temporairement. Exemple de configuration avec `mod_proxy_balancer` :

```
<Proxy "balancer://mycluster">
    BalancerMember http://backend1.local retry=5
    BalancerMember http://backend2.local retry=5
    ProxySet lbmethod=byrequests
</Proxy>

<VirtualHost *:80>
    ServerName example.com

    ProxyPreserveHost On
    ProxyPass / balancer://mycluster/
    ProxyPassReverse / balancer://mycluster/

    ErrorLog ${APACHE_LOG_DIR}/failover_error.log
    CustomLog ${APACHE_LOG_DIR}/failover_access.log combined
</VirtualHost>
```

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

3. Configuration : Gestion de l'État des Serveurs

2. Détection Automatique des Pannes

- `retry=5` : Réessaie un serveur défaillant après 5 secondes.
- Les requêtes sont redirigées vers les serveurs encore en ligne.

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

4. Failover avec Serveur Principal et de Secours

Objectif

Configurer un serveur principal et un serveur de secours pour garantir la disponibilité.

Configuration

```
<Proxy "balancer://mycluster">
    BalancerMember http://primary-backend.local loadfactor=1
    BalancerMember http://secondary-backend.local status=+H loadfactor=10
    ProxySet lbmethod=byrequests
</Proxy>

<VirtualHost *:80>
    ServerName example.com

    ProxyPreserveHost On
    ProxyPass / balancer://mycluster/
    ProxyPassReverse / balancer://mycluster/

    ErrorLog ${APACHE_LOG_DIR}/failover_error.log
    CustomLog ${APACHE_LOG_DIR}/failover_access.log combined
</VirtualHost>
```

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

4. Failover avec Serveur Principal et de Secours

Objectif

Configurer un serveur principal et un serveur de secours pour garantir la disponibilité.

Configuration

- `loadfactor` : Priorité des serveurs (1 = haute priorité).
- `status=+H` : Définit le serveur comme de secours. Les requêtes y sont envoyées uniquement si le serveur principal est défaillant.

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

5. Utilisation de Heartbeat pour la Surveillance des Backends

1. Activer Heartbeat sur les Backends

Chaque serveur backend doit envoyer des messages de heartbeat pour signaler son état.
Configurez Apache sur chaque backend pour activer `mod_heartbeat` :

```
<IfModule mod_heartbeat.c>
    HeartbeatAddress 239.0.0.1:12345
    HeartbeatMaxServers 10
    HeartbeatStorage file:/var/run/heartbeat
</IfModule>
```

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

5. Utilisation de Heartbeat pour la Surveillance des Backends

2. Configurer Apache avec `mod_lbmethod_heartbeat`

Sur le serveur frontend Apache, configurez l'équilibrage basé sur le heartbeat :

```
<Proxy "balancer://mycluster">
    BalancerMember http://backend1.local
    BalancerMember http://backend2.local
    ProxySet lbmethod=heartbeat
</Proxy>

<VirtualHost *:80>
    ServerName example.com

    ProxyPreserveHost On
    ProxyPass / balancer://mycluster/
    ProxyPassReverse / balancer://mycluster/

    ErrorLog ${APACHE_LOG_DIR}/heartbeat_error.log
    CustomLog ${APACHE_LOG_DIR}/heartbeat_access.log combined
</VirtualHost>
```

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

5. Utilisation de Heartbeat pour la Surveillance des Backends

2. Configurer Apache avec `mod_lbmethod_heartbeat`

- Apache ajustera automatiquement la répartition de charge en fonction des rapports de santé des serveurs backend.

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

6. Résilience et Monitoring

Monitoring des Backends

Pour surveiller les backends et identifier les pannes, utilisez :

1. `balancer-manager` : Interface web pour voir l'état des serveurs.
2. **Logs Apache** : Activez des logs détaillés pour le proxy et le heartbeat :

```
LogLevel proxy:trace2 heartbeat:trace2
```

Résilience

- **Failover Automatique** : Redirige le trafic vers un serveur de secours en cas de panne.
- **Retry Automatique** : Réessaye les serveurs défaillants après un délai configurable.
- **Désactivation Manuelle** : Permet d'isoler un serveur pour maintenance via `balancer-manager`.

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

7. Bonnes Pratiques

1. **Redondance**
   - Ajoutez plusieurs serveurs backend pour garantir une haute disponibilité.
   - Configurez des serveurs de secours avec une priorité basse.
2. **Surveillance**
   - Intégrez des outils comme **Nagios** ou **Zabbix** pour surveiller les performances des backends.
3. **Optimisation des Temps de Requête**
   - Configurez des délais (`timeout`, `retry`) adaptés à votre infrastructure.
4. **Cohérence des Sessions**
   - Combinez failover avec une gestion centralisée des sessions (Redis, Memcached).

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

9. Résumé

Avec Apache HTTPD, la gestion de l'état des serveurs, le failover, et le heartbeat offrent des solutions robustes pour garantir la disponibilité et la résilience des services. Les fonctionnalités clés incluent :

- **Surveillance Automatique** : Détection des serveurs défaillants.
- **Failover** : Redirection transparente en cas de panne.
- **Heartbeat** : Vérification en temps réel de la santé des serveurs backend.

Reverse Proxy et Cache

Gestion de l'État des Serveurs, Failover et Heartbeat avec Apache

9. Résumé

Avec Apache HTTPD, la gestion de l'état des serveurs, le failover, et le heartbeat offrent des solutions robustes pour garantir la disponibilité et la résilience des services. Les fonctionnalités clés incluent :

- **Surveillance Automatique** : Détection des serveurs défaillants.
- **Failover** : Redirection transparente en cas de panne.
- **Heartbeat** : Vérification en temps réel de la santé des serveurs backend.

Reverse Proxy et Cache

Keepalived et la Haute Disponibilité du Reverse-Proxy avec Apache

- Dans les architectures réseau, **Keepalived** est un outil clé pour garantir la **haute disponibilité** (HA) des services critiques tels que les reverse-proxies.
- Associé à Apache, Keepalived permet de gérer automatiquement le basculement (failover) entre plusieurs serveurs en cas de défaillance.

Reverse Proxy et Cache

Keepalived et la Haute Disponibilité du Reverse-Proxy avec Apache

1. Concepts Clés

Haute Disponibilité (HA)

- Garantit la continuité du service même si un serveur tombe en panne.
- Implémente une redondance via plusieurs serveurs actifs et un mécanisme de basculement (failover).

Keepalived

- Utilise le protocole **VRRP** (Virtual Router Redundancy Protocol) pour offrir une IP virtuelle flottante (VIP) qui peut être associée dynamiquement à un serveur.
- Fonctionne avec une configuration **master-slave** ou **actif-actif** pour les serveurs.

IP Virtuelle (VIP)

- Une adresse IP virtuelle partagée entre plusieurs serveurs.
- Les clients se connectent à la VIP, et Keepalived assure que l'un des serveurs répond toujours.

Reverse Proxy et Cache

Keepalived et la Haute Disponibilité du Reverse-Proxy avec Apache

2. Cas d'Usage : Reverse Proxy Apache avec Keepalived

Objectif

- Configurer deux serveurs Apache comme reverse proxies avec Keepalived.
- Garantir que l'adresse IP virtuelle reste accessible même en cas de panne de l'un des serveurs.

Architecture

- **Serveurs** :
  - **Apache-1 (Master)** : 192.168.1.101
  - **Apache-2 (Backup)** : 192.168.1.102
- **VIP** : 192.168.1.200
- **Backends** :
  - Serveurs backend à équilibrer via Apache.

Reverse Proxy et Cache

Keepalived et la Haute Disponibilité du Reverse-Proxy avec Apache

3. Installation de Keepalived

Sur les deux serveurs Apache, installez **Keepalived** :

```
sudo apt update
sudo apt install keepalived
```

Reverse Proxy et Cache

Keepalived et la Haute Disponibilité du Reverse-Proxy avec Apache

4. Configuration de Keepalived

Sur le Serveur Master (Apache-1)

Éditez le fichier de configuration `/etc/keepalived/keepalived.conf` :

```
vrrp_instance VI_1 {
    state MASTER
    interface eth0  # Interface réseau associée
    virtual_router_id 51
    priority 100  # Priorité la plus élevée pour le master
    advert_int 1

    authentication {
        auth_type PASS
        auth_pass secretpass
    }

    virtual_ipaddress {
        192.168.1.200  # IP virtuelle
    }

    track_script {
        chk_apache
    }
}
# Script pour vérifier si Apache fonctionne
vrrp_script chk_apache {
    script "/usr/bin/systemctl is-active apache2"
    interval 2
```

Reverse Proxy et Cache

Keepalived et la Haute Disponibilité du Reverse-Proxy avec Apache

4. Configuration de Keepalived

Sur le Serveur Backup (Apache-2)

Éditez `/etc/keepalived/keepalived.conf` :

```
vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    virtual_router_id 51
    priority 90  # Priorité inférieure pour le backup
    advert_int 1

    authentication {
        auth_type PASS
        auth_pass secretpass
    }

    virtual_ipaddress {
        192.168.1.200
    }

    track_script {
        chk_apache
    }
}

vrrp_script chk_apache {
    script "/usr/bin/systemctl is-active apache2"
    interval 2
```

Reverse Proxy et Cache

Keepalived et la Haute Disponibilité du Reverse-Proxy avec Apache

4. Configuration de Keepalived

Sur le Serveur Backup (Apache-2)

Redémarrez Keepalived

Sur les deux serveurs :

```
sudo systemctl enable keepalived
sudo systemctl start keepalived
```

Reverse Proxy et Cache

Keepalived et la Haute Disponibilité du Reverse-Proxy avec Apache

5. Configuration d'Apache comme Reverse Proxy

Configurez Apache sur les deux serveurs pour agir comme reverse proxies.

VirtualHost Configuration

Éditez `/etc/apache2/sites-available/000-default.conf` :

```
<VirtualHost *:80>
    ServerName 192.168.1.200  # Correspond à la VIP

    ProxyPreserveHost On

    <Proxy "balancer://mycluster">
        BalancerMember http://backend1.local
        BalancerMember http://backend2.local
        ProxySet lbmethod=byrequests
    </Proxy>

    ProxyPass / balancer://mycluster/
    ProxyPassReverse / balancer://mycluster/

    ErrorLog ${APACHE_LOG_DIR}/proxy_error.log
    CustomLog ${APACHE_LOG_DIR}/proxy_access.log combined
</VirtualHost>
```

Redémarrez Apache sur les deux serveurs :

```
sudo systemctl restart apache2
```

Reverse Proxy et Cache

Keepalived et la Haute Disponibilité du Reverse-Proxy avec Apache

7. Tests

1. **Tester la VIP** :
   - Accédez à `http://192.168.1.200` dans un navigateur ou utilisez `curl` :

```
curl http://192.168.1.200
```

2. **Simuler une Panne du Master** :
   - Arrêtez Apache ou Keepalived sur le serveur Master (Apache-1) :

```
sudo systemctl stop apache2
```

   - Vérifiez que la VIP passe au serveur Backup (Apache-2).
3. **Retour du Master** :
   - Relancez Apache sur le Master et observez que la VIP revient automatiquement sur le Master :

```
sudo systemctl start apache2
```

Reverse Proxy et Cache

Keepalived et la Haute Disponibilité du Reverse-Proxy avec Apache

8. Surveillance et Debugging

1. **Logs de Keepalived** :
   - Vérifiez les logs pour identifier les problèmes :

```
sudo journalctl -u keepalived
```

2. **Surveillance Active** :
   - Intégrez Keepalived à un système de monitoring (Nagios, Prometheus).
3. **Verifier les États VRRP** :
   - Utilisez `ip` pour vérifier l'attribution de la VIP :

```
ip addr show
```

Reverse Proxy et Cache

Keepalived et la Haute Disponibilité du Reverse-Proxy avec Apache

9. Bonnes Pratiques

1. **Redondance Réseau** :
   - Configurez plusieurs interfaces réseau pour éviter un point de défaillance unique.
2. **Scripts de Santé** :
   - Personnalisez le script de vérification de santé pour inclure d'autres vérifications, comme la disponibilité des backends.
3. **Priorités Ajustées** :
   - Ajustez les priorités des serveurs en fonction de leurs capacités.
4. **Évitez les Conflits IP** :
   - Assurez-vous que la VIP n'est pas utilisée ailleurs dans le réseau.

Reverse Proxy et Cache

Keepalived et la Haute Disponibilité du Reverse-Proxy avec Apache

10. Résumé

Avec Keepalived et Apache :

- La **haute disponibilité** est assurée via une **VIP** flottante.
- Les **basculements automatiques** (failover) garantissent la continuité du service.
- Apache, configuré comme **reverse proxy**, assure l'équilibrage de charge entre les backends.

Reverse Proxy et Cache