Voici une démonstration très complète mettant en œuvre un projet web PHP avec Apache en utilisant `mod_proxy`, `mod_proxy_fcgi`, et `mod_proxy_balancer`. Le projet inclut une architecture réaliste où un serveur Apache agit comme un **reverse proxy** pour une application PHP exécutée sur plusieurs serveurs backend via PHP-FPM.

Projet : Plateforme de Gestion des Utilisateurs

Objectifs :

1. Utiliser Apache comme reverse proxy pour acheminer les requêtes HTTP vers une application PHP.
2. Distribuer la charge entre plusieurs instances PHP-FPM (load balancing).
3. Gérer des routes spécifiques pour une API et une interface utilisateur.
4. Configurer des pages personnalisées pour les erreurs.
5. Optimiser la performance avec le caching.

---

1. Architecture du Projet

Composants :

1. **Serveur Apache (Frontend)** : Reçoit les requêtes des utilisateurs.
2. **Serveurs Backend PHP-FPM** :
   - Instance 1 : `http://backend1.local`
   - Instance 2 : `http://backend2.local`

Flux des Requêtes :

1. L'utilisateur accède à `https://example.com`.
2. Apache envoie les requêtes PHP vers les instances PHP-FPM via `mod_proxy_fcgi`.
3. Le load balancing répartit les requêtes entre les serveurs backend.

---

2. Préparation du Serveur Apache

Modules Apache à Activer

```
sudo a2enmod proxy proxy_fcgi proxy_balancer ssl headers cache
sudo systemctl restart apache2
```

---

3. Configuration des Backends PHP-FPM

1. Installation de PHP-FPM sur les Serveurs Backend

Sur chaque backend (par exemple, `backend1.local` et `backend2.local`) :

```
sudo apt update
sudo apt install php-fpm
sudo systemctl enable php7.4-fpm
```

2. Application PHP

Créez l'application dans `/var/www/html/app/index.php` :

```
<?php
echo json_encode([
    "server" => $_SERVER['SERVER_ADDR'],
    "path" => $_SERVER['REQUEST_URI'],
    "timestamp" => date('Y-m-d H:i:s')
]);
```

Configurez PHP-FPM pour écouter les connexions sur un socket TCP :

```
sudo nano /etc/php/7.4/fpm/pool.d/www.conf
```

Modifiez :

```
listen = 0.0.0.0:9000
```

Redémarrez PHP-FPM :

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

---

4. Configuration du Serveur Apache (Reverse Proxy)

VirtualHost avec Load Balancing

Fichier `/etc/apache2/sites-available/example.com.conf` :

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

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

```
    # Activer le reverse proxy
    ProxyPreserveHost On

    # Configurer le cluster des serveurs backend
    <Proxy "balancer://phpcluster">
        BalancerMember fcgi://backend1.local:9000
        BalancerMember fcgi://backend2.local:9000
        ProxySet lbmethod=byrequests
    </Proxy>

    # Traiter les fichiers PHP via FastCGI
    ProxyPassMatch ^/(.*\.php)$ balancer://phpcluster/var/www/html/app/$1

    # Configurer les fichiers statiques
    Alias /static /var/www/html/app/static
    <Directory "/var/www/html/app/static">
        Require all granted
    </Directory>

    # Pages d'erreur personnalisées
    ErrorDocument 404 /static/errors/404.html
    ErrorDocument 500 /static/errors/500.html

    ErrorLog ${APACHE_LOG_DIR}/example.com_error.log
    CustomLog ${APACHE_LOG_DIR}/example.com_access.log combined
</VirtualHost>
```

Activez le site et rechargez Apache :

```
sudo a2ensite example.com.conf
sudo systemctl reload apache2
```

---

5. Gestion des Routes

API et Interface Utilisateur

Ajoutez des routes spécifiques dans l'application PHP :

```
<?php
$request_uri = $_SERVER['REQUEST_URI'];

if (strpos($request_uri, '/api') === 0) {
    header('Content-Type: application/json');
    echo json_encode(["message" => "Welcome to the API"]);
} elseif ($request_uri === '/') {
    echo "<h1>Welcome to the User Management Platform</h1>";
} else {
    http_response_code(404);
```

```
    echo "<h1>404 Not Found</h1>";
}
```

---

6. Tests Fonctionnels

1. Tester le Load Balancing

Accédez à `https://example.com` plusieurs fois et vérifiez que les réponses proviennent alternativement de `backend1.local` et `backend2.local`.

2. Tester les Routes

- Accédez à `https://example.com/api` pour voir la réponse JSON de l'API.
- Accédez à `https://example.com/unknown` pour vérifier la page d'erreur 404 personnalisée.

3. Tester les Performances

Utilisez `ab` (Apache Benchmark) pour tester le load balancing :

```
ab -n 100 -c 10 https://example.com/
```

---

7. Optimisations

Caching

Ajoutez un cache pour les fichiers statiques et les réponses :

```
CacheRoot /var/cache/apache2/proxy
CacheEnable disk /
CacheIgnoreNoLastMod On
CacheIgnoreCacheControl On
CacheDefaultExpire 3600
CacheHeader on
```

Compression

Activez la compression pour améliorer les performances :

```
<IfModule mod_deflate.c>
    AddOutputFilterByType DEFLATE text/html text/plain text/xml
application/json
</IfModule>
```

8. Bonnes Pratiques

1. **Sécuriser les Backends**
    - Utilisez des réseaux privés pour que les backends soient accessibles uniquement depuis Apache.
2. **Logs et Débogage**
    - Activez un niveau de logs plus élevé pour le débogage :

```
LogLevel proxy:trace2
```

3. **Testez les Règles**
    - Validez avec des outils comme `curl` :

```
curl -v https://example.com/api
```

9. Résumé

Ce projet montre comment configurer un serveur Apache comme **reverse proxy** pour une application PHP :

- `mod_proxy_fcgi` pour la communication avec PHP-FPM.
- `mod_proxy_balancer` pour répartir la charge entre plusieurs serveurs backend.
- Gestion des fichiers statiques, des routes, et des erreurs personnalisées.