Voici une démonstration complète et détaillée pour comparer les performances de **HTTP/2** et **HTTP/1.1**.

Objectif

Comparer HTTP/2 et HTTP/1.1 en termes de performance pour télécharger plusieurs fichiers.

Pré-requis

1. **Serveur avec Apache installé** (Ubuntu, Debian, etc.).
    - Si Apache n'est pas installé :

```
sudo apt update
sudo apt install apache2 openssl
```

2. **Accès root ou sudo** pour configurer Apache et générer des certificats SSL.
3. **Deux sous-domaines ou un domaine avec deux noms distincts** (par exemple : `http2.example.com` et `http1.example.com`) configurés pour pointer vers l'IP de votre serveur. Si vous testez localement, vous pouvez utiliser `/etc/hosts`.
4. **Modules Apache nécessaires :** Activez les modules requis pour HTTP/2 et SSL :

```
sudo a2enmod ssl
sudo a2enmod http2
sudo systemctl restart apache2
```

Étapes détaillées

1. Créez les répertoires des sites

Créez des répertoires pour héberger les fichiers des deux sites :

```
sudo mkdir -p /var/www/http2.example.com
sudo mkdir -p /var/www/http1.example.com
```

2. Créez des fichiers lourds

Générez 50 fichiers de 1 Mo chacun dans les deux répertoires :

```
sudo bash -c 'for i in {1..50}; do base64 /dev/urandom | head -c 1048576 >
/var/www/http2.example.com/file$i.txt; done'
```

```
sudo bash -c 'for i in {1..50}; do base64 /dev/urandom | head -c 1048576 >
/var/www/http2.example.com/file$i.txt; done'
```

---

3. Générez un certificat SSL auto-signé

Créez un certificat SSL auto-signé pour vos sites :

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
-keyout /etc/apache2/ssl/example.key \
-out /etc/apache2/ssl/example.crt \
-subj "/C=US/ST=State/L=City/O=Organization/CN=*.example.com"
```

---

4. Configurez les VirtualHosts

Créez deux fichiers de configuration dans `/etc/apache2/sites-available/` :

HTTP/2 VirtualHost : `/etc/apache2/sites-available/http2.example.com.conf`

```
<VirtualHost *:443>
    ServerName http2.example.com
    DocumentRoot /var/www/http2.example.com

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

    Protocols h2 http/1.1

    ErrorLog ${APACHE_LOG_DIR}/http2-error.log
    CustomLog ${APACHE_LOG_DIR}/http2-access.log combined
</VirtualHost>
```

HTTP/1.1 VirtualHost : `/etc/apache2/sites-available/http1.example.com.conf`

```
<VirtualHost *:443>
    ServerName http1.example.com
    DocumentRoot /var/www/http1.example.com

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

    Protocols http/1.1
```

```
    ErrorLog ${APACHE_LOG_DIR}/http1-error.log
    CustomLog ${APACHE_LOG_DIR}/http1-access.log combined
</VirtualHost>
```

---

5. Activez les sites et redémarrez Apache

Activez les deux sites et redémarrez Apache :

```
sudo a2ensite http2.example.com.conf
sudo a2ensite http1.example.com.conf
sudo systemctl reload apache2
```

---

6. Configurez le fichier `/etc/hosts` (si vous testez en local)

Ajoutez les lignes suivantes pour que vos sous-domaines pointent vers `localhost` :

```
127.0.0.1 http2.example.com
127.0.0.1 http1.example.com
```

Tester les performances

1. Vérifiez les protocoles

- Testez si HTTP/2 est activé :

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

  Vous devriez voir `HTTP/2` dans la réponse.

- Testez si HTTP/1.1 est activé :

  ```
  curl -I https://http1.example.com
  ```

  Vous devriez voir `HTTP/1.1` dans la réponse.

---

2. Mesurez les temps de téléchargement

- sudo apt install build-essential git

- git clone https://github.com/wg/wrk.git
- cd wrk
- make
- sudo cp wrk /usr/local/bin
- wrk -t12 -c100 -d30s --latency https://http2.example.com/file1.txt
- wrk -t12 -c100 -d30s --latency https://http1.example.com/file1.txt

-t12 : Utilise 12 threads. -c100 : Simule 100 connexions ouvertes. -d30s : Test pendant 30 secondes.

Résultats attendus

1. **HTTP/2 :**
    - Les fichiers sont téléchargés plus rapidement grâce au multiplexage.
    - Une seule connexion TCP est utilisée pour télécharger tous les fichiers simultanément.
2. **HTTP/1.1 :**
    - Les fichiers sont téléchargés séquentiellement, ce qui prend plus de temps.
    - Chaque fichier nécessite une connexion ou une gestion de pipeline.

Analyse des différences

- **Multiplexage :** HTTP/2 télécharge plusieurs fichiers sur une même connexion, réduisant les délais liés à la latence.
- **Compression des en-têtes :** HTTP/2 compresse les en-têtes des requêtes, ce qui réduit la taille des données transmises.
- **Latence réduite :** HTTP/2 est conçu pour minimiser l'impact de la latence réseau.