Gestion de la memoire cache d'un serveur web

Nom : Grari - Prénom : Mohammed Achraf - Matricule : 000509902

Nom : Suau - Prenom : Thomas - Matricule : 000580972

Nom : Massondi - Prenom : Williame - Matricule : 000568610

Nom :Nguemayem Kaze - Prenom : Lydie Nadia - Matricule : 000509851

Nom : Meghadio Gonzeu - Prenom : Cedric Jordan - Matricule : 000564114

Contents

1 Introduction à NGINX

Dans l'optique de croissance de CoursParticuliers.com sur le web et des outils numériques nous pensons utile de vous fournir des détails techniques sur le fonctionnement de NGINX bien que nous n'utiliserons que la partie proxy cache et serveur de cet outil.

L'utilisation la plus fréquente de NGINX est de le configurer comme un serveur Web classique pour servir des fichiers statiques et comme un proxy pour les requêtes dynamiques typiquement acheminées en utilisant une interface FastCGI vers un ou des serveurs applicatifs avec un mécanisme de répartition de charge. Il assure plusieurs tâches importantes pour que le site fonctionne bien et soit sécurisé.

- Proxy inversé avec mise en cache : NGINX peut être utilisé pour faire en sorte que les demandes d'un utilisateur soient envoyées à différents serveurs web, puis stocker en mémoire les résultats pour que les futures demandes soient traitées plus rapidement.
- IPv6 : NGINX peut comprendre et gérer les nouvelles adresses IP, qui vont remplacer les anciennes adresses IPv4 et permettent de connecter plus d'appareils à internet.
- Équilibrage de charge : NGINX peut s'assurer que les demandes des utilisateurs sont distribuées de manière égales entre plusieurs serveurs, pour éviter que certains ne soient surchargés et que le temps de réponse soit trop élevé.
- Support FastCGI avec mise en cache : NGINX peut communiquer avec des applications web pour récupérer des informations plus rapidement, puis stocker ces informations en mémoire pour que les futures demandes soient traitées plus rapidement.
- WebSockets : NGINX peut permettre une communication en temps réel entre un utilisateur et un serveur, ce qui est utile pour les chats en ligne, les jeux en ligne, etc.
- Gestion des fichiers statiques, des fichiers d'index et de l'indexation automatique : NGINX peut gérer différents types de fichiers, tels que les images, les pages web et les feuilles de style, et peut également aider à organiser les fichiers dans des dossiers pour que les utilisateurs puissent les trouver plus facilement. Cela peut se faire via l'usage de l'argument `location`.
- TLS/SSL avec SNI : NGINX peut protéger les informations des utilisateurs en utilisant des protocoles de chiffrement pour que les données soient illisibles par les personnes mal intentionnées, et il peut également gérer plusieurs noms de domaine sur un même serveur pour économiser de l'espace et faciliter la maintenance.

NGINX est plus performant qu'Apache pour le traitement des requêtes de contenu statique. Il peut également servir de nombreux clients en même temps lors d'une charge élevée, ce qui en fait un excellent choix pour un site à fort trafic grâce à un système de worker. Là où Apache crée un processus par connexion, NGINX lance une série de workers qui vont chacun être capable de gérer de multiples connexions d'une manière non bloquante.

La configuration de NGINX est morcelée en plusieurs fichiers pour une plus grande liberté de configuration pour chaque composants du serveur et se trouve dans le dossier `/etc/nginx`.

1. `nginx.conf`, contient la configuration générale ;
2. `conf.d/*.conf` contient la configuration des différents serveurs ;
3. `sites-available/*.conf` contient les fichiers de configurations de tous nos serveurs web NGINX et `sites-enabled/*.conf` doit contenir les fichiers de configuration des sites actifs sur le serveur.

Suivant l'installation NGINX semble marcher sur tous les OS sans distinction.

2 Instalation NGINX

Il faut git clone notre repo à l'adresse suivante : https://github.com/tsua0002/PROJET_SYS_ADMIN__Q2_2023

Puis suivre la partie installation du `README`.

```
git clone https://github.com/tsua0002/PROJET_SYS_ADMIN__Q2_2023
```

```
cd PROJET_SYS_ADMIN__Q2_2023
```

```
chmod +x launch.sh
```

```
./launch.sh
```

Notre serveur a comme Système d'exploitation : Linux.
L'IP hôte a été configuré sur `0.0.0.0` avec notre nom de domaine `coursparticuliers.com` afin de tester notre configuration. En production, il nous suffira de remplacer `0.0.0.0` par l'IP du serveur.

Pour plus de détails sur les configurations et le script de lancement, veuillez vous référez au commentaire dans les fichiers suivants **launch.sh**, **nginx.conf** et **coursparticuliers.com**.

Le fichier **launch.sh** contient à la fois l'installation de NGNIX et son lancement.

3 Détails de notre configuration

3.1 Notre configuration globale nginx.conf

Le serveur NGINX a été configuré pour héberger un site Web à l'aide du protocole HTTP sur le **port** 80. La directive "listen" a été utilisée pour définir le port et le protocole pour les connexions IPv4 et IPv6. La directive "server_name" a été utilisée pour spécifier le nom de domaine pour le site Web. Le répertoire racine pour le site Web a été défini avec la directive "root".

De plus, la directive "location" a été utilisée pour définir la durée de validité des fichiers en cache à 30 jours pour améliorer les performances du site.

La configuration SSL a également été mise en place avec les protocoles TLSv1 à TLSv1.3 pris en charge et les ciphers préférés du serveur configurés avec la directive "ssl_prefer_server_ciphers".

Les journaux d'accès et d'erreur ont été définis pour le serveur NGINX avec les directives "access_log" et "error_log", respectivement. La compression gzip a été activée pour compresser les données transférées entre le serveur et les clients, réduisant ainsi le temps de réponse.

Enfin, les configurations virtuelles de NGINX ont été incluses à l'aide de la directive "include" pour faciliter la gestion des sites Web multiples sur le serveur.

En plus, de la configuration globale `/etc/nginx/nginx.conf` nous avons configuré le site web spécifiquement via `/etc/nginx/site-available/coursparticuliers.com`.

3.2 Nos choix de configuration pour le site web coursparticuliers.com

Pour le caching, notre problème initial, nous avons opté pour 30 jours. Notre site n'ayant pas pour vocation a être mis à jour régulièrement, nous pensons que ce temps de caching chez le client peut augmenter grandement l'efficacité et la réactivité du site. Cela afin d'éviter de futurs désagrément liés à un trafic trop important.

Le bloc de configuration de la directive "location /" définit le comportement du serveur pour les requêtes qui commencent par "/". Le serveur tentera de servir le fichier demandé et, s'il ne peut pas le trouver, cherchera

un répertoire portant le même nom que le fichier et tentera de servir le fichier "index.html" ou "index.htm" qu'il contient. Si le fichier ou le répertoire n'existe pas, le serveur renverra une erreur 404.

Nous avons également fait le choix de ne pas implémenter de serveur proxy par soucis d'économie financière. (Dans le scénario l'accent à été mis sur le coût financier comme un critère important)

4 Commandes utiles

- **Arrêter NGINX** : systemctl stop nginx
- **Vérifier l'état de NGINX** : systemctl status nginx
- **Redémarrer NGINX** : systemctl restart nginx
- **Démarrer NGINX** : systemctl start nginx

5 Maintenances futures

Fichier de config principal du site /etc/ng inx/sites-enabled/coursparticuliers.com

Si des ralentissements sont de nouveaux présents sur le site alors nous avons deux solutions :

1. Augmenter le temps de caching `expires` dans `location`
2. Augmenter la gestion du nombre de connexion entrantes autorisées dans `worker_connections`.

Afin d'aider au mieux la maintenance future du site internet vous pourrez consulter cette référence qui présente différentes stratégies de caching qui peuvent être implémentée par la suite : Stratégies de cache pour le web.