|  |
| --- |
|  |
| Squid - Proxy |
| Guide du débutant |

|  |
| --- |
| Adil BOUZIT  [Date] |

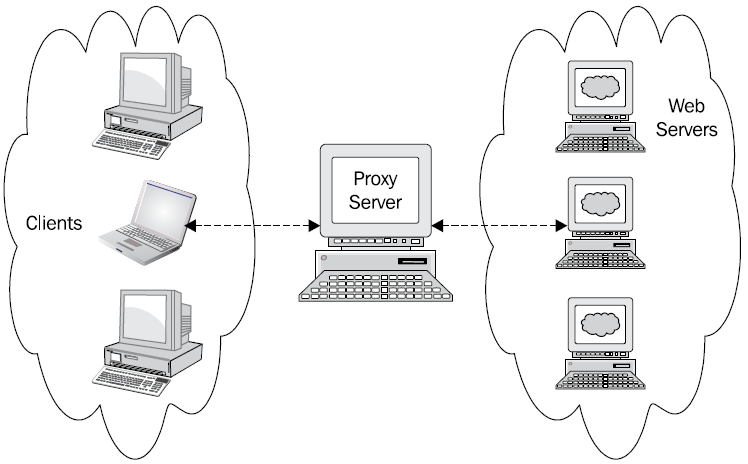
Proxy & Reverse Proxy

Qu’est-ce qu’un proxy ?

Un proxy est un système informatique, un intermédiaire situé entre un client et un serveur. Dans sa forme simple, le proxy facilite la communication entre un client et serveur cible sans modifier les requêtes cliente ou réponses serveur. La mise en place d’un serveur proxy empêche donc le client de communiquer directement avec le serveur cible.

Lorsque l’on initie une requête vers un serveur cible, le serveur proxy détourne la connexion et se présente au serveur cible comme étant lui-même le client, requêtant le serveur en son nom (celui du client), si une réponse est reçue, le serveur proxy nous la retourne, donnant le sentiment d’avoir communiqué directement avec le serveur cible.

Dans sa forme avancée, le serveur proxy peut filtrer les requêtes selon des règles définies permettant ou non la communication au moyen d’une validation. Les règles sont généralement basées selon les adresses ip des client ou d’un serveur, d’un protocole, type de document etc



Parfois, un serveur proxy peut modifier les requêtes ou réponses, voir même, conserver localement les réponses du serveur pour pouvoir les renvoyer à d’autres clients, ce procédé s’appelle la mise en cache ou « caching », c’est une technique populaire utilisé par les proxy pour sauvegarder de la bande passante, participer à la réduction de charge des serveurs et améliorer l’expérience utilisateur. Ils sont communément utilisés pour accomplir les tâches suivantes :

* Réduire la bande passante
* Améliorer l’expérience utilisateur en réduisant le temps de chargement de page
* Renforcer les règles de sécurité
* Monitorer le trafic utilisateur
* Anonymiser un utilisateur sur internet
* Filtre les requêtes ou réponse en utilisant un service de détection anti-virus
* Équilibrer les charges de trafic entre de multiples connexions internet

Qu’est-ce qu’un reverse proxy ?

Le reverse proxy ou « reverse proxying » est une technique de stockage local de réponse ou ressources d’un serveur web, ainsi, les requêtes peuvent être satisfaite depuis la copie locale sur le serveur proxy, parfois même sans avoir à contacter le serveur. Le proxy ou le web cache vérifie si les copies locales sont toujours valides avant de distribuer les copies

La durée de vie des copies stockées est calculée depuis une en-tête http additionnelle reçue par le serveur web. Utilisant les en-têtes HTTP, un serveur web peut contrôler ce qu’il distribue comme document/réponse pouvant être mis ou non en cache par le serveur proxy.

La mise en cache est communément utilisée :

* Réduire l’utilisation de la bande passante : Un large nombre de page web statique comme les fichiers CSS et JavaScript, les images, vidéos etc peuvent être mise en cache du moment qu’ils ne changent pas fréquemment et constituent la majeure partie des réponse d’un serveur web.
* Par les FAI pour réduire le temps moyen de chargement d’une page pour améliorer l’expérience de leurs client sur leurs réseaux commutés ou à bande large.
* Pour réduire la charge de travail d’un serveur web occupé en distribuant les pages statiques etc depuis les contenu copié en cache du serveur proxy.

Squid comme proxy

Il existe plusieurs façons d’installer Squid :

1. Télécharger le code source et le compiler
2. Télécharger la version squid précompilé, disponible sur les divers distributions (exemple : apt-get install squid)

Installer Squid depuis son code source

L’installation de squid depuis son code source se fait en 3 étapes :

1. Sélectionner les fonctionnalités et les paramètres spécifiques à notre OS
2. Compiler le code source pour générer des fichiers exécutables
3. Placer les fichiers exécutables et autres fichiers dans les destinations désignée par Squid afin qu’il fonctionne correctement

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

On utilisera le code source de la version 6 :

wget <http://www.squid-cache.org/Versions/v6/squid-6.10.tar.gz>

Compiler Squid :

Est un processus de compilation de fichiers critiques contenant du C/C++ dans son code source et génère des exécutables. Compiler Squid peut être fait en quelques étapes. Nous avons besoin pour cela d’un compilateur ANSI C/C++. Si vous n’en disposez pas, installez en un.

Pourquoi compiler Squid

Compiler Squid peut être une tâche ardue, comparé à l’installer depuis le package de binaires, cependant, il est recommandé de compiler squid depuis son code source plutôt que d’utiliser des fichiers binaires précompilés, voici ici quelques avantages à la compilation :

* On peut activer des fonctionnalités supplémentaires qui peuvent ne pas être activée dans le package précompilé et/ou désactiver les fonctionnalités dont nous n’avons pas besoin pour un environnement particulier
* « ./configure » recherche des fonctionnalités sur le système et les active/désactive en conséquence, tandis que le package binaire précompilé disposent de fonctionnalités détectées pour le système sur lequel les sources ont été compilées
* En utilisant « ./configure » on peut personnalisé entièrement l’installation de Squid

Pour résumé, la compilation offre beaucoup d’avantages sur l’installation via les packages de binaires qui lui aussi possèdent ses propres avantages comme une installation rapide.

Configure ou system check

« Configure » ou « System Check » est la première étape du processus de compilation et s'effectue en exécutant « ./configure » à partir de la ligne de commande.

Ce programme sonde le système et s'assure que les paquets requis sont installés. Il vérifie aussi les capacités du système et collecte des informations sur son architecture et les paramètres par défaut, tels que les descripteurs de fichiers disponibles, etc.

Après le lancement de la commande « ./configure », le programme va générer les fichiers « Makefile » requis pour la compilation.

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

L'exécution de « ./configure » sans paramètre utilisera les valeurs par défaut. Pour modifier les paramètres par défaut ou désactiver certaines fonctionnalité etc nous devons passer des options à « ./configure »

la commande suivante pour voir les options disponibles ainsi qu'une brève description :

« ./configure --help | less »

Les options :

--prefix : est utilisé pour identifier les différentes version et prévenir des interférences ou confusions entre celles-ci, il est conseillé de les installer dans des répertoires distincts .

Exemple : Pour installer la version 6.10 de Squid on peut utiliser le répertoire /opt/squid/6.10, la commande configure correspondante sera ./configure –prefix=/opt/squid/3.1.10/

Squid offre plus de contrôle sur l’emplacement des types de fichiers. Leurs emplacement peut être définit à l’aide d’options tels que --bindir , --sbindir , etc , pour plus de détails consulter la page d’aide de « configure » .

--enable-gnuregex : Des expressions régulières seront utilisées pour construire les ACL dans squid, si l’installation se fait sur Linux ou Unix, nous n’avons pas besoin d’activer l’option.

--disable-inline : Squid a beaucoup de code qui peut être intégré, ce qui est bon pour une utilisation en production. Mais le code en ligne prend plus de temps à compiler et est utile lorsque nous avons besoin de compiler une source une seule fois pour configurer Squid. Cette option est destinée à être utilisée lorsque nous avons besoin de compiler à plusieurs reprises.

--disable-optimizations : Par défaut Squid est compilé avec des optimisations du compilateur qui se traduisent par de meilleurs performances. Cette option doit être utilisée pour résoudre un problème/tester les versions car cela réduit le temps de compilation.

L’option --disable-inline est automatiquement utilisée.

--enable-storeio : Les performances de Squid dépendent des performances d'E/S (écriture par seconde) du disque lorsque la mise en cache du disque est activée. Plus Squid peut lire/écrire rapidement des fichiers à partir du cache , moins il faudra de temps pour satisfaire une requête.

Différentes techniques de stockage permettent d'optimiser les perf, en fonction du type de trafic/utilisation. On peut utiliser l’option pour construire Squid avec le support de différents modules d'entrées/sorties. Consultez le répertoire src/fs/ dans le code source de Squid pour connaître les modules d'entrées-sorties disponibles.

« ./configure --enable-storeio=ufs,aufs,coss,diskd,null »

--enable-removal-policies : En utilisant la mise en cache sur disque, on demande à Squid d'utiliser un espace disque spécifique pour la mise en cache. A un moment, Squid aura besoin de plus d'espace pour mettre en cache les documents, il doit alors décider quels anciens documents doivent être supprimés/purgés du cache pour libérer de l'espace. Il existe plusieurs politiques de purge de documents pour tirer le meilleur parti de la mise en cache.

Les politiques sont basées sur les structures de données de type "tas" et "liste". La structure

de données en liste est activée par défaut.

Veuillez consulter le répertoire src/repl/ dans le code source de Squid pour connaître les politiques de suppression disponibles.

« ./configure --enable-removal-policies=heap,lru »

--enable-icmp : Permet de déterminer la distance par rapport aux autres pairs de cache et aux serveurs distants afin d'estimer la latence approximative. L’option n'est utile que s'il existe d'autres pairs de cache dans le réseau.

--enable-delay-pools : Squid utilise des pools de retard pour limiter/contrôler la bande passante utilisée par un client ou groupe de clients. Ses pools sont comme des seaux qui laissent échapper des données (trafic web) vers les clients, remplis à un rythme contrôlé. Ils sont utiles lorsque nous devons contrôler la bande passante utilisée.

--enable-esi : Cette option permet à Squid d'utiliser les Edge Side Includes. Si l’option est activée, Squid ignore complètement les en-têtes cache-control des clients. L’option ne doit être utilisée que lorsque Squid est utilisé en mode accélérateur.

--enable-useragent-log : Permet d'enregistrer les en-têtes de l'agent utilisateur dans les requêtes HTTP des clients.

--enable-referer-log : Si nous activons cette option, Squid pourra écrire un champ d'en-tête referer à partir des requêtes HTTP.

--enable-cachemgr-hostname : Cache Manager (cachemgr.cgi) est un utilitaire CGI qui permet de gérer le cache de Squid et de visualiser les statistiques du cache à l'aide d'une interface web. Le nom d'hôte pour accéder au gestionnaire de cache peut être défini à l'aide de cette option. Par défaut, nous pouvons accéder à l'interface web du gestionnaire de cache en utilisant localhost ou l'adresse IP du serveur Squid.

« ./configure --enable-cachemgr-hostname=squidproxy.example.com »

--enable-arp-acl : Permet de créer des listes de contrôle d'accès basées sur les adresses MAC (ou Ethernet). Cette fonction est désactivée par défaut. Si l’on veut contrôler l'accès des clients sur la base de l'Ethernet nous devons activer cette fonctionnalité.

--enable-ssl : Lorsqu’il est configuré en proxy inverse, il peut mettre fin aux connexions SSL initiées par les clients et les gérer pour le compte du serveur web dorsal en amont signifiant qu’il n'aura pas à effectuer de travail sur SSL. Dans ce cas, la communication entre Squid et le serveur web dorsal sera purement HTTP, mais les clients y verront toujours une connexion sécurisée avec le serveur web.

--enable-cache-digests : Les résumés de cache sont un moyen pour Squid de partager des informations avec les serveurs Squid voisins.

--enable-default-err-language : Quand Squid rencontre une erreur qui doit être communiquée au client, Squid utilise des pages par défaut pour afficher ces erreurs. Les pages d'erreur sont disponibles dans les langues locales. L’option permet de spécifier la langue par défaut pour toutes les pages d'erreur. La langue par défaut des pages d'erreur est l'anglais.

« ./configure --enable-default-err-language=Spanish »

--enable-err-languages : Par défaut, Squid prend en charge toutes les langues disponibles. Si nous ne voulons construire Squid qu'avec les langues familières, nous pouvons utiliser cette option. Consultez le répertoire errors/ dans le code source de Squid pour connaître les langues disponibles.

« ./configure --enable-err-languages='English French German' »

--enable-ipfw-transparent : IPFIREWALL (IPFW) est un pare-feu pour système FreeBSD maintenu par sa communauté. Cette option est utile pour configurer le Serveur Proxy Transparent sur les systèmes avec IPFW. Si notre système n'a pas IPFW alors on évite d'utiliser cette option car Squid ne se compilera pas. Le comportement par défaut est l'auto-détection.

--enable-ipf-transparent : IPFilter (IPF) est un pare-feu pour de nombreux systèmes Unix. Il est fourni par NetBSD, Solaris, etc. Si notre système dispose d'IPF, nous devons activer cette option pour pouvoir configurer Squid en mode transparent. L'activation de cette option en l'absence d'IPF sur le système entraînera des erreurs de compilation.

--enable-pf-transparent : Packet Filter (PF) est un pare-feu développé à l'origine pour OpenBSD. L’option est utile sur les systèmes où PF est installé , pour obtenir le mode Proxy transparent. N'activez pas cette option si PF n'est pas installé.

--enable-linux-netfliter : Netfilter est le cadre de filtrage des paquets dans les noyaux Linux des séries 2.4.x et 2.6.x. Cette option est utile pour activer la prise en charge du proxy transparent sur les systèmes Linux.

--enable-follow-x-forwarded-for : Lorsqu'une requête HTTP est transmise par un proxy, celui-ci écrit des informations sur lui-même et sur le client destinataire de la requête, dans les entêtes HTTP. Cette option permet à Squid d'essayer de rechercher l'adresse IP du client d'origine pour lequel la requête a été transmise par un ou plusieurs serveurs proxy.

--enable-default-hostsfile Cette option permet de sélectionner l'emplacement par défaut du fichier hosts. Sur la plupart des OS il se trouve dans le répertoire /etc/hosts.

« ./configure --enable-default-hostsfile=/some/other/location/hosts »

--enable-auth : Active la prise en charge des schémas d'authentification. Cette option est utilisée que pour activer la prise en charge globale de l'authentification et la liste des schémas d'authentification n'est pas transmise.

Le schéma d'authentification est activé avec --enable-auth-AUTHENTICATION\_SCHEME où AUTHENTICATION\_SCHEME est le nom du schéma d'authentification.

Par défaut, tous les schémas d'authentification sont activés et les aides à l'authentification correspondantes sont construites lors de la compilation. Les aides à l'authentification sont des programmes externes qui peuvent authentifier les clients à l'aide de divers mécanismes l'authentification, par rapport à différentes bases de données d'utilisateurs.

--enable-auth-basic : Active la prise en charge d'un schéma d'authentification de base et établit la liste des aides spécifiées. Si la liste des aides n'est pas fournie, cette option activera toutes les aides possibles. Une liste des aides disponibles se situe dans le répertoire helpers/basic\_auth/ dans le code source de Squid.

Pour désactiver ce schéma d'authentification, nous utilisons --disable- auth-basic.

« ./configure --enable-auth-basic=PAM,NCSA,LDAP »

Si nous voulons activer cette option mais que nous ne voulons pas construire d'aides, nous devons utiliser "none" au lieu d'une liste d'aides.

« ./configure --enable-auth-basic=none »

Auparavant, cette option était connue sous le nom de --enable-basic-auth-helpers.

La liste des aides est transmise de la même manière.

« ./configure --enable-basic-auth-helpers=PAM,NCSA,LDAP »

--enable-auth-ntlm : Permet la prise en charge le schéma d'authentification NTLM. Les aides disponibles pour ce schéma se trouvent dans le répertoire helpers/ntlm\_auth/ du code source de Squid.

Pour désactiver la prise en charge de ce schéma, utilisez l'option --disable-auth-ntlm.

« ./configure --enable-auth-ntlm=smb\_lm,no\_check »

--enable-auth-negotiate : Active le schéma d'authentification négocié. Les détails et la syntaxe sont similaires à ceux de l'option de schéma d'authentification ci-dessus.

« ./configure --enable-auth-negotiate=kerberos »

--enable-ntlm-fail-open : Si cette option est activée et qu'une aide échoue lors de l'authentification d'un utilisateur, elle peut toujours permettre à Squid d'authentifier l'utilisateur. Cette option doit être utilisée avec précaution car elle peut conduire à des erreurs d'authentification. aux failles de sécurité.

--enable-external-acl-helpers : Squid supporte les ACLs externes en utilisant des aides. Si nous voulons utiliser des ACLs externes, nous devrions envisager d'utiliser cette option. Nous pouvons également utiliser cette option pendant l'apprentissage. Une liste d'ACLs externes. Les aides ACL doivent être passées pour construire des aides spécifiques.

Par défaut Squid construit toutes les aides disponibles. Une liste des aides ACL externes disponibles se trouve dans le répertoire helpers/external\_acl/ de son code source.

« ./configure --enable-external-acl-helpers=unix\_group,ldap\_group »

--disable-http-violations : Squid dispose d'options de configuration lui permettant la violation des normes du protocole HTTP, en remplaçant les champs d'en-tête dans les requêtes/réponses HTTP. La manipulation des en-têtes va à l'encontre des normes HTTP standard. L’option désactive la prise en charge de toutes violations du protocole HTTP.

--disable-wccp : Désactive la prise en charge du protocole de communication de cache Web (WCCP) de Cisco. Le WCCP permet la communication entre les caches, ce qui contribue à la localisation du trafic. Par défaut, la prise en charge du WCCP est activée.

--disable-wccpv2 : Désactive la prise en charge le protocole WCCP v2 de Cisco (activée par défaut) qui dispose d'un support intégré pour le loadbalancing, la scalabilité, des mécanismes de tolérance aux pannes et d'assurance de service.

--disable-snmp : Dans les versions 3.x de Squid, le protocole SNMP est activé par défaut. SNMP est très populaire parmi les administrateurs de systèmes pour la surveillance des serveurs et des périphériques de réseau.

--disable-htcp : Le protocole HTCP (Hypertext Caching Protocol) peut être utilisé par Squid pour envoyer et recevoir des résumés de cache aux caches voisins. Cette option désactive la prise en charge du protocole HTCP.

--disable-ident-lookups : Empêche Squid d'effectuer des recherches d'identité ou d'identifier un nom d'utilisateur pour chaque connexion. Désactiver cette fonction peut empêcher une attaque DDOS de la part d'un client malveillant.

--disable-internal-dns : Squid possède sa propre implémentation DNS et est capable de construire des requêtes DNS, pour l’utiliser nous ne devons pas le désactiver.

--disable-translation : Par défaut, Squid essaie de présenter les pages d'erreur et de manuel dans la langue locale. Pour que cela ne se produise pas, nous utilisons cette option.

--disable-auto-locale : Sur la base des en-têtes d’une requête cliente, Squid tente de fournir des pages d'erreur localisées. L’option permet de désactiver la localisation automatique. La balise error\_directory dans le fichier de configuration de Squid doit être configurée si nous utilisons cette option.

--disable-unlinkd : unlinkd est un processus externe utilisé pour effectuer des appels système de type "unlink". Cette option désactive le support de unlinkd dans Squid. Désactiver unlinkd n'est pas une bonne idée car l'appel système unlink peut bloquer un processus pendant un temps considérable, ce qui peut entraîner un retard dans les réponses.

--with-default-user : Normalement, nous ne voulons pas exécuter Squid en tant qu'utilisateur root afin d'éviter tout risque de sécurité. Par défaut, Squid s'exécute sous l'utilisateur « nobody ». Si nous avons installé Squid à partir d'un binaire pré-compilé, Squid peut s'exécuter en tant qu'utilisateur « squid »ou « proxy » en fonction de l’OS utilisé. Cette option permet de définir l'utilisateur par défaut pour l'exécution de Squid.

L'exemple suivant montre comment utiliser cette option :

« ./configure --with-default-user=squid »

--with-logdir : Par défaut, Squid écrit tous les journaux et rapports d'erreur dans des fichiers désignés dans ${prefix}/var/ logs/.

Cet emplacement est différent de celui utilisé par tous les autres processus et daemons pour écrire leurs journaux. Afin d'accéder rapidement aux journaux de Squid, nous pourrions vouloir les placer dans le répertoire de journaux système par défaut, qui est /var/log/ dans la plupart des systèmes Linux.

Voir Démarrer avec Squid l'exemple suivant de la syntaxe pour y parvenir :

« ./configure --with-logdir=/var/log/squid/ »

--with-pidfile : L'emplacement du PID de Squid est ${prefix}/var/run/squid.pid, ce qui n'est pas l'emplacement standard du système pour le stockage des fichiers PID.

Sur la plupart des systèmes Linux les fichiers PID sont stockés dans /var/run/. Il est donc possible de modifier l'emplacement par défaut du fichier PID à l'aide de l'option suivante :

« ./configure --with-pidfile=/var/run/squid.pid »

--with-aufs-threads : Permet de spécifier le nombre de threads à utiliser lorsque le système de stockage AUFS est utilisé pour gérer les répertoires du cache. Si cette option n'est pas utilisée, Squid calcule automatiquement le nombre de threads à utiliser :

« ./configure --with-aufs-threads=12 »

--without-pthreads : Les anciennes versions de Squid ne supportaient pas les threads POSIX. Maintenant, Squid est construit avec le support pthreads par défaut, donc, si nous ne voulons pas activer le support pthreads, nous devrons le désactiver explicitement.

--with-openssl : Utilisée pour construire Squid avec le support d’OpenSSL en spécifiant le chemin d’installation d’OpenSSL s’il n’est pas installé à l’emplacement par défaut

« ./configure --with-openssl=/opt/openssl/ »

--with-large-files : En cas de trafic important, les fichiers journaux de Squid (en particulier le journal d'accès) augmentent rapidement et, à long terme, la taille du fichier peut devenir assez importante. Nous pouvons utiliser cette option pour activer la prise en charge des fichiers journaux volumineux.

Pour de meilleures performance, il est recommandé de changer de fichier de log fréquemment plutôt de concentrer les logs dans un seul fichier plus large.

--with-filedescriptors : Les OS utilisent des descripteurs de fichiers (essentiellement des nombres entiers) pour suivre les fichiers et les sockets ouverts. Par défaut, le nombre de descripteurs de fichiers qu'un utilisateur peut utiliser est limité (normalement 1024). Une fois que Squid a accepté des connexions qui ont consommé tous les descripteurs de fichiers disponibles pour l'utilisateur Squid, il ne peut plus accepter d'autres connexions, sauf si certains des descripteurs de fichiers sont libérés.

En cas de forte charge, Squid se retrouve fréquemment à court de descripteurs de fichiers.

Nous pouvons utiliser l'option suivante pour résoudre le problème de pénurie de

descripteurs de fichiers :

« ./configure --with-filedescriptors=8192 »

Un héros en herbe – Le descripteur de fichiers !

Déterminez le nombre maximum de descripteurs de fichiers disponibles pour votre utilisateur. Notez également les commandes qui permettront de fixer la limite maximale de descripteurs de fichiers disponibles à 8192.

Solution : Pour vérifier le nombre de descripteurs de fichiers disponibles, utilisez la commande suivante : « ulimit -n »

Pour fixer la limite des descripteurs de fichiers à 8192, nous devons ajouter les lignes suivantes au fichier /etc/security/limits.conf :

username hard nofile 8192

username soft nofile 8192

L'action précédente ne peut être effectuée qu'avec les privilèges de root ou de super utilisateur.

Passer à l’action – Lancer notre première commande

Maintenant que nous avons jeté un bref coup d'œil à plusieurs des options disponibles, nous pouvons présenter les options pour l'environnement pour lequel nous construisons Squid.

Avant toute chose, nous créons notre utilisateur « squid », ainsi fait, nous sommes prêts à lancer la commande « ./configure » avec les options suivantes :

« ./configure --prefix=/opt/squid/ --with-logdir=/var/log/squid/ --with-pidfile=/var/run/squid.pid --enable-storeio=ufs,aufs --enable-removal-policies=lru,heap --enable-icmp --enable-useragent-log --enable-referer-log --enable-cache-digests --with-large-files --with-default-user=squid »

Explorer Squid - ${prefix}/

Jetons un œil sur les fichiers et répertoires générés durant l’installation. Utilisons la commande tree pour afficher le répertoire de Squid en arborescence : « tree ${prefix} | less »

${prefix} est le répertoire utilisé pour l’installation avec l’option « --prefix » ajouté à « ./configure ». En pratique, notre commande sera : « tree /opt/squid | less » .

/bin

Ce répertoire contient les programmes pouvant être exécutés ou lancés par les utilisateur.

/bin/squidclient

squidclient est un client HTTP avec des fonctions avancées permettant la manipulation de requêtes HTTP à des fins de test du serveur Squid. Lancez le pour voir les options disponibles.

/etc

**C’est le répertoire contenant tous les fichiers de configuration de Squid.**

/etc/squid.conf

C’est le fichier de configuration de Squid, squid.conf est généré durant l’installation, il dispose de la configuration minimale requise pour permettre l’utilisation de Squid. Pour apporter une nouvelle configuration à Squid, c’est ce fichier que nous devons modifier.

/etc/squid.conf.default

Squid génère ce fichier de configuration par défaut, ainsi, nous disposons de la configuration minimale par défaut si nous décidons un retour à zéro de la configuration.

/etc/squid.conf.documented

Ce fichier est une version entièrement documentée de squid.conf contenant des centaines de commentaires, nous devons toujours nous référer à ce document lorsque l’on veut chercher et ajouter des tags de configuration correspondant à notre version de squid.

/libexec

Répertoire contenant des « programmes aidants » construits durant la compilation de Squid.

/libexec/cachemgr.cgi

Ce programme CGI fourni une interface web pour gérer le cache de Squid : Cache Manager.

/sbin

Répertoire contenant des programmes pouvant être exécutés par un utilisateur root/sudo.

/sbin/squid

C’est le programme Squid, qui est généralement exécuté comme un daemon (processus).

/share

Répertoire contenant les modèles de page d’erreur, documentation et autres, utilisé par Squid.

/share/errors

Répertoire contenant les modèles de page d’erreur localisées.

/share/icons

Répertoire contenant un nombre d’images utilisé pour le FTP ou le Gopher Directory Listing

/share/man

C’est l’endroit où se situent les pages du manuel de Squid, Squidclient, les programmes aidants construits pendant la compilation ou des pages d’aides pouvant être lu avec « man »

Pour lire la page « ${prefix}/share/man/man8/squid.8 » on lancera la commande suivante :

**man /opt/squid/share/man/man8/squid.8**

/var

Un endroit où sont placé des fichiers temporaires de Squid lorsqu’il est en cours d’exécution.

/var/cache

Répertoire par défaut qui est utilisé pour stocker les documents web en cache.

/var/logs

C’est le répertoire par défaut des fichiers logs, comme cache.log, acces.log etc) de Squid.

Sommaire

Nous avons appris que les serveurs proxy et la mise en cache web en général et la façon dont ils peuvent être utiles, spécifiquement pour économiser de la bande passante et améliorer l’expérience utilisateur. Nous avons ensuite exploré le répertoire d’installation de Squid qui est un serveur proxy puissant pour la mise en cache. Les éléments suivants sont des points importants que nous avons appris dans ce chapitre :

* Les différentes façons d'utiliser Squid pour la production ou le développement
* Signification des différentes options de configuration
* Compilation du code source de Squid
* Installer Squid à partir du code source et du paquetage binaire
* Avantages et inconvénients de la compilation du code source de Squid

Nous avons également discuté de la structure des répertoires et des fichiers générés par Squid pendant l'installation.

Maintenant que nous savons comment installer Squid, nous sommes prêts à apprendre comment configurer Squid en fonction des besoins d'un environnement réseau donné. Nous verrons cela à l'aide de quelques exemples dans le chapitre suivant.

Configurer Squid

Dans ce chapitre, nous aborderons les points suivants :

* Présentation rapide de Squid
* Syntaxe du fichier de configuration
* Port HTTP, la directive de configuration la plus importante
* Listes de contrôle d'accès (ACL)
* Contrôle de l'accès aux différents composants de Squid
* Pairs de cache ou voisins
* Mise en cache des documents Web dans la mémoire principale et sur le disque dur
* Réglage de Squid pour améliorer les économies de bande passante et réduire la latence
* Modification des en-têtes HTTP accompagnant les requêtes et les réponses
* Configuration de Squid pour utiliser les serveurs DNS
* Quelques directives relatives à la journalisation
* Autres directives de configuration importantes ou couramment utilisées

Commençons donc par le commencement.

Démarrage rapide

Avant que l’on explore le fichier de configuration en détail, jetons un œil sur la configuration minimale dont nous avons besoin pour commencer. Positionnez vous dans le fichier de configuration localisé dans ${prexif}/etc/squid.conf car nous allons effectuer des changements et ajouts pour mettre en place rapidement un serveur proxy.

On ajoute les lignes de commande suivantes en haut de notre fichier de configuration *(assurez-vous de remplacer l’IP par celle de la machine utilisée pour naviguer internet)* :

cache\_dir ufs /opt/squid/var/cache/ 500 16 256

acl ma\_machine src 192.0.0.0 #Remplacer l’addresse IP par la votre

http\_access allow ma\_machine

Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

Maintenant, nous avons besoin de créer les répertoires de caches. On peut le faire en utilisant la commande suivante : « sudo ${prefix}/squid/sbin/squid -z »

Nous sommes prêts à démarrer notre serveur proxy, et ceci peut être fait en lançant la commande suivante : « sudo ${prefix}/squid/sbin/squid »

Squid va commencer à écouter sur le port 3128 (port par défaut) – sur toutes les interfaces réseau de notre machine et nous pouvons configurer votre navigateur pour qu’il utilise squid comme serveur proxy http ayant pour hôte l’IP du serveur sur le port 3128.

Une image contenant texte, capture d’écran, Police, Logiciel multimédia

Description générée automatiquement

Une fois votre navigateur configuré, essayé d’aller sur « http://www.exemple.com » et observez le message que reçu. Le message affiché est un accès refusé envoyé par votre Squid !

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Syntaxe du fichier de configuration

Le fichier de configuration de Squid peut être trouvé dans /etc/squid/squid.conf, /usr/local/squid/etc/squid.conf ou ${prefix}/etc/squid.conf où ${prefix} est la valeur de l’option --prefix passée avec la commande ./configure avant la compilation.

Dans les récentes versions de Squid, une version documentée du fichier de configuration, connu sous le nom de squid.conf.documented, peut être trouvé au même endroit que squid.conf . Nous aborderons certaines directives du fichier de configuration.

Référer vous à : <http://www.squid-cache.org/Doc/config/>

La syntaxe du fichier de configuration est similaire à beaucoup d’autres programmes pour Linux/Unix. Généralement, il y a quelques lignes de commentaires contenant des informations utiles à la compréhension de chaque directives utilisées dans le fichier de configuration et ce, même pour ceux qui ne sont pas familiers avec la configuration d’application via un fichier. Normalement, nous avons seulement besoin de lire les commentaires et utiliser l’option appropriée pour chaque options disponible pour une directive spécifique.

Les lignes commençant par le caractère # sont traités comme des commentaires et sont complètement ignorées par Squid lors de l’analyse du fichier de configuration. Additionnellement, chaque ligne vide sont aussi ignorées.

Voyons un extrait issu du fichier de configuration documenté (squid.conf.documented) :

Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

Dans cet extrait, à la première ligne, le nom de la directive est mentionné : « cache\_effective\_user » par un TAG, les lignes suivantes le tag fournissent de brèves informations à propos de l’utilisation de la directive. La dernière ligne montre la valeur par défaut de la directive, si aucune n’est spécifiée.

Types de directives

Directive à valeur unique

Comme indiquée, ses directives ne prennent qu’une seule valeur, ses directives ne devraient pas être appelées plusieurs fois dans le fichier de configuration car la dernière occurrence de la directive écrasera les précédentes déclaration. Par exemple :

|  |
| --- |
| logfile\_rotate 10  # Few lines containing other configuration directives  logfile\_rotate 5 |

Dans ce cas-là, cinq rotations de logfiles seront effectuées lorsque nous déclencherons la rotation des logfiles par Squid.

Directives à valeur booléennes ou à alternance

Ses directives ont, elles aussi, des valeurs uniques mais celles-ci sont généralement utilisée pour faire alterner une fonctionnalité entre « on » et « off »

|  |
| --- |
| query\_icmp on  log\_icp\_queries off  url\_rewrite\_bypass off |

On utilise ses directives lorsque l’on a besoin de changer des comportements par défaut.

Directives avec le temps comme valeur

Ses directives, comme nommé, prennent le temps comme valeur unitaire. Squid comprend les mots « seconds », « minutes » et « hours» etc, ces mots sont associés à des valeurs numérique spécifiant les valeurs de temps.

|  |
| --- |
| request\_timeout 3 hours  persistent\_request\_timeout 2 minutes |

Directives dont les valeurs sont la taille d’un fichier ou de la mémoire

Les valeurs données à ces directives sont généralement précédées d'unités de taille de fichier ou de mémoire telles que des octets, des Ko, des Mo ou des Go. Par exemple :

|  |
| --- |
| reply\_body\_max\_size 10 MB  cache\_mem 512 MB  maximum\_object\_in\_memory 8192 KB |

D’autres directives ?

Si vous ouvrez le fichier de configuration documenté de squid, vous trouverez d’autres types de directives prenant en charge des valeurs particulières, ouvrez donc le fichier pour savoir quels types de directives sont acceptées par Squid, mais aussi la syntaxe de leurs valeurs.

La directive HTTP port

Cette directive permet de spécifier le port d’écoute de connexions clientes. Par défaut, le port est 3128. Voyons maintenant les façons de spécifier le port dans le fichier squid.conf :

* Dans sa forme simple, on spécifie le port sur lequel on veut que Squid écoute :

http\_port 8080

* On peut aussi spécifier l’adresse IP combiné avec un port sur lequel on veut que Squid soit en écoute. On utilise cette approche lorsque l’on dispose de plusieurs interfaces sur notre machine et que l’on désire que Squid écouter uniquement sur une interface :

http\_port 192.0.2.25 :3128

* Une autre où l’on spécifie http\_port en utilisant le nom d’hôte combiné au port :

http\_port myproxy.exemple.com :8080

* Un autre aspect de cette directive est qu’elle puisse prendre des valeurs multiples sur des lignes séparées, Ces lignes vont déclencher l’écoute de Squid sur 3 adresses/port différentes, comme dans l’exemple suivant :

http\_port 192.168.0.25:8080

http\_port lan1.exemple.com :3128

http\_port lan2.exemple.com :8081

* Dans les version récentes de squid, nous pouvons aussi spécifier le mode opératoire de la directive comme « intercept », « tproxy », « accel » etc
* Le mode « Intercept » va supporter l’interception de requête sans avoir besoin de configurer les machines clientes :

http\_port 3128 intercept

* le mode tproxy va permettre l’activation du proxy transparent linux afin d’usurper les connexions entrantes à l’aide de l’adresse IP du client :

http\_port 8080 tproxy

* Le mode accélérateur est bien pour écouter sur le port 80, ce mode ne peut pas être utilisé tel quel, nous devons spécifier au moins un site web que à accélérer :

http\_port 80 accel defaultsite=website.example.com

*Note : Activer intercept ou tproxy désactive n’importe quel mécanisme d’authentification configuré.*

Controller l’accès au serveur proxy

Le contrôle d’accès HTTP

Les ACL identifient les requêtes en se basant sur différentes règles. Elles sont combinées avec des directives de contrôles d’accès pour autoriser/refuse un accès. http\_acces est utilisé pour accorder l’accès aux opérations HTTP par l’intermédiaire de Squid. Voyons voir la syntaxe :

http\_access allow|deny [!]ACL\_NAME

En utilisant http\_access, on peut autoriser/refuser l’accès à une opération http à travers Squid. « ACL\_NAME » est le nom de l’ACL pour laquelle l’accès doit être autorisé ou refusé. Si un bang ( ! ) est préfixé devant ACL\_NAME alors l’accès sera autorisé ou refusé pour toutes les requête qui ne sont pas identifiées par ACL\_NAME.

Combiner les ACL avec http\_access

Lorsque l’on a de multiples règles d’accès, à la réception d’une requête, Squid va lire les règles inscrite de haut en bas dans le fichier et ce jusqu’à ce qu’une action soit déterminée par allow ou deny. Si l’on a plusieurs ACL dans une seule règle alors la requête sera comparée avec toutes les ACL de gauche à droite et Squid arrête le traitement de la règle dès qu'il rencontre une ACL qui ne peut pas identifier la requête. Une règle d'accès comportant plusieurs ACL n'aboutit à une action définie que si la requête est identifiée par toutes les ACL utilisées dans la règle.

acl my\_home\_machine src 192.0.2.21

acl my\_lab\_machine src 198.51.100.86

http\_access allow my\_home\_machine

http\_access allow my\_lab\_machine

Ses règles et ACL dans ce code vont permettre l’accès au serveur proxy Squid aux hôtes 192.0.2.21 et 192.51.100.86. Elles peuvent aussi être écrite de cette manière :

acl my\_machines src 192.0.2.21 198.51.100.86

http\_access allow my\_machines

Le comportement par autorise l'accès à tous clients d'un réseau local et à refuser l'accès aux autres. Si nous voulons que des clients (en dehors du réseau local) puissent utiliser le serveur proxy, nous devons ajouter des règles d'accès supplémentaires pour les autoriser :

http\_access deny all

Le paramètre all est un élément représentant **toutes** les adresses IP. Cette ligne refusera l'accès à tout. Comme elle vient après toutes les autres règles d'accès, les requêtes de clients inconnus seront refusées.

La directive http\_reply\_access

http reply est une réponse du serveur web correspondante à la requête initiée par le client. En utilisant cette directive on peut contrôler l’accès aux réponses reçues :

http\_reply\_access allow|deny [!]ACL\_NAME

Cette directive remplace partiellement les permissions donnée par http\_access, en exemple :

acl my\_machine src 192.0.2.21

http\_access allow my\_machine

http\_reply\_access deny my\_machine

Nous avons autorisé http\_access à l'hôte 192.0.2.21 mais elle ne pourra pas accéder correctement aux sites web car elle n'est pas autorisée à recevoir de réponses. L'hôte peut seulement envoyer des requêtes au serveur proxy. Cette directive est normalement utilisée pour refuser l'accès à des types de contenu tels que l'audio, la vidéo etc

ICP access

Directive utilisée pour contrôler l'accès aux requêtes par caches voisins utilisant le protocole ICP (Internet Cache Protocol), autorise/refuse l'accès au port ICP. Son comportement par défaut est le refus de toutes requêtes ICP.

icp\_access allow|deny [!]ACL\_NAME

HTCP access

Permet de contrôler les réponses de Squid à certaines requêtes HTCP. Son comportement par défaut est de les refuser.

htcp\_access allow|deny [!]ACL\_NAME

HTCP CLR access

Les caches voisins peuvent demander la purge ou la suppression d'objets de cache sous la forme de requêtes HTCP CLR. La directive htcp\_clr\_access peut être utilisée pour accorder l'accès à la purge aux seuls pairs de cache de confiance.

Htcp\_clr\_access allow|deny [!]ACL\_NAME

Miss access

Cette directive est utilisée pour spécifier le cache parent que tous les pairs ou clients du cache peuvent utiliser. Lorsqu'un homologue ou client du cache tente de récupérer un contenu à l'aide de notre serveur mandataire, la demande peut aboutir à un MISS (non présent dans le cache) ou à un HIT (peut être satisfait à partir de notre cache). En général, un MISS est récupéré par notre serveur pour le compte d'un client ou d'un homologue. Si nous ne voulons pas que nos clients ou nos pairs récupèrent du contenu en utilisant notre proxy, nous pouvons utiliser la directive miss\_access, comme indiqué :

acl bad\_clients src 192.0.2.0/24

miss\_access deny bad\_clients

miss\_access allow all

Ce code ne permettra pas aux bad\_clients d'utiliser notre serveur proxy comme proxy parent. Le comportement par défaut est d'autoriser tous les clients qui passent la règle http\_access à utiliser le serveur proxy comme parent.