**Syslog-NG**

|  |
| --- |
| **DESC** |
| Collecteur / forwarding de logs |
| **FILES** |
| /etc/syslog-ng/syslog-ng.conf |
| **QUICK START** |
| * Editer conf * Ecrire une source * Ecrire une destination * Ecrire un log statement qui fait la corrélation * Reload de syslog-ng |

**Caractéristiques :**

* Buffer disque : Stockages des logs sur DD si le serveur central ne répond plus ou si crash syslog ou autre et réémission dans l’ordre. (Paramètres buffer RAM aussi) : disk-buffer() au niveau de la destination.
* TLS pour l’échange de logs avec certificats client et serveur.
  + Sur serveur :
    - Son propre certificat
    - Certificat du CA qui a signé le certificat du client sur le serveur si besoin.
  + Sur client :
    - Le certificat du CA qui a signé le certificat serveur
    - Certificat client si besoin.

Voir site pour mettre certificats chez clients et serveur.

* Filtres pour les messages entrants.
* Parsing des messages entrants, notamment JSON, rewrinting, corrélation et agrégation de plusieurs logs entrants en un seul sortant.
* Supporte HDFS/Elastic search, Apache Kafka, AMPQ, STOMP, MongoDB, MSSQL…Graphite…
* Supporte logs JSON.
* Supporte IPv6.
* Flow-control (via le flag flow-control dans le log statement).
* Ne gère pas la high-availability : c’est à l’OS de le faire.
* Syslog-ng compatible avec logrotate.
* Possibilité de connaitre le nombre de logs droppés : via stats-freq() des options globales, stats-level() qui contrôle la verbosity et via la commande syslog-ng-ctl stats.
* Multi threads via une option globale.

**Processing d’un message :**

* Polling de la **source**
* #messages maxi rapatriés de la source : log-fetch-limit() propre a chaque source.
* Chaque connexion possède son control-window de taille log-iw-size() si le flow-control est activé dans les flags : un message lu = taille décroit de 1, un message envoyé = augmente la taille. Lorsque la taille atteint 0, on stop de lire la source (surcharge).

Log-iw-size() > max-connections() \* log-fetch-limit() pour que la window accepte tous les messages entrants lors de chaque poll.

* Processing du message :
  + Parsing automatique du message entrant au format syslog (via syslog-parser), erreur sinon (sauf si flag précisé) et remplissage des **macros**.

Si ne peut pas parser : tout le log est stocké dans ${MSG}

Si le flag no-parse est activé, rajout uniquement des headers syslog au message.

* + Si aucun timezone, on ajoute le timezone local ou le timezone spécifié.

Faire attention lors du forward du message à spécifier la bonne time-zone : le timestamp sera modifié en conséquence.

* + **Parsing** du message si besoin afin de Mettre à jour les macros correspondantes.
  + Puis **template** si besoin pour affecter les macros au message sortant dans un template spécifié (ex JSON).
  + Si le message (source) remplit la règle (**filtre**), il est forwardé (**destination**).
* Messages mis dans :
  + Output queue : 64 messages en RAM
  + Disk buffer si activé et si output queue pleine en DD : permet de supporter les crashs syslog-ng.
  + Overflow queue de taille log-fifo-size() qui doit être plus grand que le log-fetch-limit() si les deux pleines, et bien sûr >> log-iw-size() si flow-control. En RAM

Soft flow-control : On joue avec les paramètres des buffer RAM/DISK

Hard flow-control : control-flow dans les flags.

* Forwarding vers la destination

Chaque source et chaque destination à son propre buffer.

Parsing : on découpe le log entrant afin de crée des macros utilisables ensuite.

Template : on réorganise le log afin de le formater pour correspondre à la destination avec l’aide des macros.

**Ancien Syslog : RFC 3164 (Syslog legacy ou BSD syslog)**

1 K maxi, message contenant :

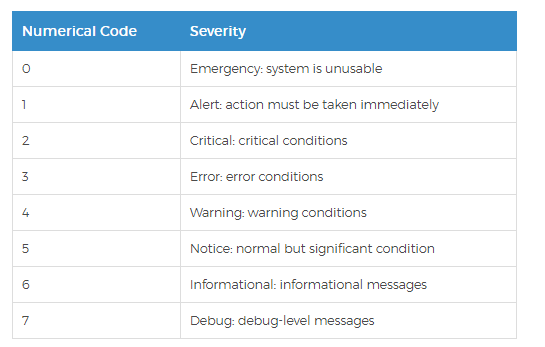
* PRI = 8 \* facility + severity
* HEADER : timestamp + hostname/IP
* MSG : Name of program/PID : msg

**New Syslog : RFC 5424 (IETF Syslog)**

Message contient :

* HEADER : Priority, version (1), ISOTIMESTAMP, HOSTNAME/IP, APP, PID, MSGID (ASCII format)
* STRUCTURED-DATA : meta-informations: [name="value"…] (UTF-8 format) Syslog-ng parse les datas structurées et on y a accès avec des macros.
* MSG (UTF-8 format)





**Quick syntaxe :**

Source s\_source {

Source\_driver(param , param) ;

} ;

Destination d\_destination {

Destination\_driver(param, param) ;

} ;

Log {

Source(s\_source) ;

Destination(d\_destination) ;

} ;

**Syntaxe générale du fichier de conf :**

@version : 3.8 #Obligatoire

@define name "value" #Accès à la variable avec `name`

@define allow-config-dups 1 #Autorisation de définir deux fois un objet avec le même nom.

#Le dernier prend la valeur finale.

@include "file" #idem qu’en C, \* marche .

**#Generality**

**#Syslog-ng –syntax-only** pour checker la validité du fichier de configuration.

# \_ ou – équivalents

# virgule ou espace équivalents dans les arguments d’une fonction.

# Nombres : 12 (base 10 ) 012 (octale) 0x12 (hexa).

#${MACRO} et $MACRO pareil

#pour utiliser @, utiliser @@

#Utilisé les mêmes drivers pour client et serveur.

#Source est en écoute sur un port uniquement si elle est définit dans un log statement.

* **Inline définition**: comme les classes anonymes en java qui ne fonctionnent qu’une fois : au lieu de source(s\_source) ; on utilise : source { internal() ;} ;
* **Plusieurs confs dans un même objet : fonction channel{} ;**



On pourra utiliser les résultats du parser dans les autres objets du log statement.

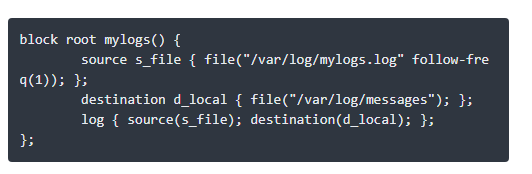
* **SCL : source configuration file = blocks = C++ templates = bloc de code réutilisable dans un fichier à part. On peut passer des arguments à un block.**

Appel avec block()

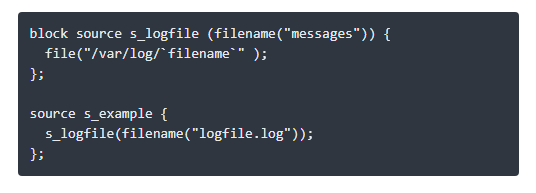




Utiliser root si plusieurs objets dans le block :



Passation d’arguments à un block avec valeur par défaut :



#${HOST} : Macro

**#Options**

Options{

Options(args) ;

} ;

|  |  |  |
| --- | --- | --- |
| **Option** | **Args** | **Comments** |
| Use\_dns(yes|no) ; |  | |
| Dns-cache(yes|no) ; |  | |
| Dns-cache-hosts("file") ; | Résolution DNS locale | |
| Dns-cache-size("#hostnames") ; |  | |
| File-template(template) ; | ré ordonnancement du message | |
| Flush-lines(100) ; | Attente de 100 messages avant de forwarder. | |
| Flush-timeout() ; | Temps avant que les messages soient forwarder. | |
| Keep-hostname() ; |  | |
| Keep-timestamp() ; |  | |
| Log-fifo-size() ; | Overflow queue | |
| Log-msg-size() ; | Taille maxi d’un log | |
| Normalize-hostnames() ; | Hostname reçus de DNS convertit en minuscules. | |
| Recv-time-zone() ; |  | |
| Send-time-zone() ; |  | |
| Stats-freq(600) ; | Statistiques paquets droppés. | |
| Threaded(yes|no) ; | Multi threads | |
| Time-zone() ; |  | |
| Ts-format(iso) ; | Conversion des timestamp | |

**#Sources**

Source s\_name {

source\_driver (arg1 , [arg2] , …) ;

source\_driver2(arg1 , [arg2] , …) ;

…

} ;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Source driver** | **Args** | | | **Comments** |
| Internal( [args] ) ; | Messages internes à Syslog-ng | | | |
|  | Host-override("new name") | | | Remplace ${HOST} du msg. |
|  | Tags("") | | | Labélisation du msg. |
|  | Use-fqdn(yes|no) | | | Long hostname | short |
| File("filename") ; | fichier | | | |
|  | Follow-freq(1) | | | Track toutes les 1 secondes  1.5 possible. |
|  | Flags([args]) | No-parse | | Syslog-ng par défaut parse le message en syslog quand il arrive. Ici, pas de parsing et rajout des headers syslog.  Message entrant accessible via ${MESSAGE} |
|  |  | kernel | | Evite les deadlocks : syslog-ng lit dans les messages systèmes donc les lock, toute comme certains démons systèmes.  Associé avec follow-freq(0) |
|  |  | Empty-lines | | Syslog-ng supprime les lignes vides, sauf si ce flag est mis. |
|  |  | No-multi-line | | Syslog-ng va transformer le message en un message d’une ligne. |
|  |  | Dont-store-legacy-msgdhr | | Par défaut, syslog-ng stocke le header original. |
|  |  | Syslog-protocol | | Les messages sont au formats new syslog sans header. |
|  |  | Validate-utf8 | | Ajout du BOM si absent. |
|  | Default-facility(String) | | | Si le log nen contient pas |
|  | Default-priority(String) | | |  |
|  | Encoding(UTF-8) | | |  |
|  | Keep-timestamp(yes|no) | | | On garde le timestamp du message original ou on le remplace. |
|  | Log-fetch-limit(10) | | | Nombre d’éléments du fichier rapatriés lors de chaque check |
|  | Log-iw-size(100) | | | Initial window de flow-control.  Doit être supérieur à log-fetch-limit(). |
|  | Log-msg-size(8192) | | | Tailles des messages maxi |
|  | Program-overide("string") | | | Remplace le contenu de ${PROGRAM} |
|  | Time-zone("TZ") | | |  |
| Network([args]) ; | RFC 3164  Par défaut 0.0.0.0 donc écoute sur toutes les interfaces sur TCP 601 | | | |
|  | Localip() | | | Ecoute sur un interface particulière. |
|  | Ip-protocol(4|6) | | | IPv6 |
|  | Ip("X.X.X.X") | | |  |
|  | Transport("udp|tcp") | | | Ecoute sur le port 514 pour udp |
|  | Port(X) | localport(X) | | | Port d’écoute. |
|  | Tls([args]) | Peer-verify("required-trusted") | |  |
|  |  | Key-file("path") | |  |
|  |  | Cert-file("path") | |  |
|  | Flags() encoding() keep-timestamp()log-fetch-limit() log-iw-size() log-msg-size() tag() time-zone() | | | Cf file() |
|  | Host-override("String") | | | Remplace ${HOST} |
|  | Keep-hostname(yes|no) | | | Si oui, on garde le hostname du message original, pas d’override. |
|  | Max-connections(10) | | |  |
|  | So-rcvbuf(0) | | | Taille de la socket de reception = Nombre de messages/secondes. |
|  | So-sndbuf(0) | | |  |
|  | Use-dns(yes|no|persist\_only) | | | Persist\_only : résolution locale.  Syslog-ng bloquant sur les résolutions DNS : attention DDOS |
| Nodejs() ; | @include "scl.conf" obligatoire  En interne utilise driver network() + processing JSON  Par défaut port 9003 sur toutes les adresses. | | | |
|  | Ip() | localip() | | |  |
|  | Port() | localport() | | |  |
| Pipe("filename") | Pareil que file() sauf que c’est read-write.  Voir docs pour options. | | | |
| Pacct() ; | Process accounting logs  Historiques cmds, users qui ont utilisés la cmd, CPU time du process, code d’exit… | | | |
| Program("path") ; | Start du programme et les logs sont lus de stdout | | | |
| Syslog() ; | RFC 5424 (IETF)  Options pareilles que network() | | | |
| System() ; | Messages systèmes | | | |
| Systemd-journal([args]) ; |  | | | |
| Tcp() tcp6() udp() udp6() ; | Obsolète | | | |
| Unix-stream("path" [args]) ; | Ecoute sur socket UNIX  Linux, TCP (SOCK\_STREAM) | | Certaines macros sont utilisables | |
| Unix-dgram("path" [args]) ; | Ecoute sur socket UNIX  BSD, UDP (SOCK\_DGRAM) | |

**#Destination**

Destination d\_name{

Destination-driver(param, param) ;

} ;

A noter que l’on peut disk-bufferisé les messages avant envoie pour éviter d’en perdre.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Destination driver** | **Args** | | | **Comments** |
| Ampq() ; |  | | | |
| Elasticsearch(  Index("")  Type("")  Cluster("")  ) ; | @module mod-java  @module "scl.conf"  Installation de dépendances requises (Cf site) | | | |
| Elasticsearch2(  Index("")  Type("")  Cluster("")  ) ; | @module mod-java  @module "scl.conf"  Utilisation de la REST API  Installation de dépendances requises (Cf site)  Envoie de messages JSON | | | |
|  | Client-mode("http") | | | http : REST API  … |
|  | Cluster-url("url") | | |  |
| File("path" [args]) ; | Utilisation de macros possible ici | | | |
|  | Create-dirs(yes|no) | | | Création si besoin. |
|  | Dir-owner() | | |  |
|  | Dir-perm() | | |  |
|  | Disk-buffer() |  | | Bufferise les messages sortants au cas ou failure |
|  |  | Reliable(yes|no) | | Impossible de perdre des messages si crash syslog, destination crash, ou reload/restart mais lent |
|  |  | Dir("") | | Emplacement disk buffer |
|  |  | Disk-buf-size() | |  |
|  |  | Mem-buf-length(10000) | | Si reliable(no)  10000 messages maxi dans le buffer |
|  |  | Mem-buf-size() | | Si reliable(yes) |
|  |  | Disk-buf-size() | |  |
|  |  | Qout-size() | | #messages dans le buffer d’output |
|  | Flags() | | |  |
|  | Flush-lines(10) | | | Attend 10 messages avant de forwarder |
|  | Log-fifo-size() | | | #messages dans la output queue |
|  | Overwrite-if-older(X) | | | Overwriting si dernière modif > X secondes |
|  | Owner() | | |  |
|  | Perm() | | |  |
|  | Template() | | | Envoyé le message dans un template != de syslog |
| Graphite(payload([args])) ; | Utilisation de graphite (metrics monitoring) | | | |
| Destination vers serveur http possible via java | | | | |
| http() ; | Destination vers serveur http sans java | | | |
| Kafka() ; | Apache kafka | | | |
| Mongodb([args]) ; |  | | | |
| Network() ; | RFC 3164  si dns activé dans les options, on peut mettre le nom de l’host | | | |
|  | Tls() | |  | |
| Pipe() ; |  | | | |
| Program() ; | Envoyé les logs à un programme. | | | |
| Pseudofile() ; | Ecrire dans /proc ou /sys par ex | | | |
| Smtp() ; | Génération d’email à partir de logs. | | | |
| Sql() ; |  | | | |
| Stomp() ; |  | | | |
| Syslog() ; | RFC 5424 | | | |
|  | Port() | destport() | | Port distant d’écoute | |
|  | Localport() | | Port local d’envoie | |
|  | Localip() | |  | |
|  | Ip() | |  | |
|  | Tls() | |  | |
| Unix-stream() ; |  | | | |
| Unix-dgram() ; |  | | | |
| Usertty() ; |  | | | |
| custom | Ecrire le SCL correspondance en C, Python… | | | |

**#Logs paths = log statement = correspondance sources/filtres/destination**

#Les logs statements peuvent être imbriqués.

Log{

Source(s\_name) ;

Filter(filtre) ;

Parser() ;

Rewrite() ;

Destination(d\_name) ;

Flags(flags) ;

} ;

**#Flags**

|  |  |  |
| --- | --- | --- |
|  | **Args** | **Comments** |
| Flags(args) ; |  | |
|  | final | Le process s’arrête ici. Les autres logs statement ne seront pas traités. |
|  | Flow-control | Evite la perte de message si la destination ne peut pas suivre |

**#Filtres**

Filter f\_filter {

Filter-type("expression")  [or|and|not|and not Filter-type("expression") ] ;

} ;

|  |  |  |
| --- | --- | --- |
| **Filter type** | **Args** | **Comments** |
| Host("") ; |  | |
| Match("word" [args]) ; | Value("MESSAGE") | Search in $MESSAGE |

**#Macro**

Hard macros dérivées d’une partie du message entrant, read only.

Soft macros : named value-pairs, générés à partir du log message ou custom.

|  |  |
| --- | --- |
| **Macro** | **Comments** |
| ${HOST} |  |
| ${DAY} |  |

**#Template : transformation du message**

Les templates utilisent les macros.

Template t\_template{

Template("")  <template-escape(yes|no)> ; #Template escape permet d’échapper les chars

$(function-name arg arg…)

} ;

Ou

Template t\_template "" #Si pas besoin du template escape

|  |  |  |
| --- | --- | --- |
| **template function** | **Args** | **Comments** |
| $(format-json args) |  | |
|  | --scope |  |
|  | --exclude |  |
|  | --key |  |
|  | --pair |  |
| $(graphite-output args) |  | |
|  |  |  |

**#Parsers**

Permet de MAJ les macros une fois le log parsé : toutes les clefs JSON ne retrouveront dans des macros par exemple.

Parser parser\_name{

Parser-name(args) ;

} ;

|  |  |  |
| --- | --- | --- |
| **Parser name** | **Args** | **Comments** |
| Json-parser() |  | |
|  | Marker() |  |
|  | Prefix("") | Permet d’affecter un préfixe aux macros générées. |

**Renommage des macros :**

Value-pairs(

Scope( //on Selectionne un groupe de Macro prédéfinies.

Nv\_pairs //name-value pairs : soft macros (sauf les macros .\*)

Core //=base=rfc3164

Syslog //=rfc5424

All\_macros //Toutes les hard-macros

Selected-macros //rfc 3164+ TAGS, SOURCEIP, SEQNUM

sdata //Toutes les structured-data du message nouveau-syslog

everything

)

Exlude("R\_\*") //Exclure certaines macros

Exclude(".SDATA")

Key("HOST" , "PROGRAM") //Macros spécifiques

Pair("MSGHDR" "$PROGRAM[$PID] : ") //Définition de nouvelles macros

)

**Modules :**

1 module = 1 ou plusieurs plugins = source/destination driver par exemple.

**Syslog-ng –version** : liste des modules available

**Syslog-ng –module-registry** : description des modules available

**Syslog-ng –defaut-modules** : Modules chargés au démarrage de syslog-ng

@module moduleName pour inclure un module.

@define autoload-compiled-modules 0 pour ne charger aucun modules automatiquement au démarrage.

**Paquets droppés :**

Possibilité de connaitre le nombre de logs droppés :

* Options globales :
  + stats-freq()
  + stats-level() qui contrôle la verbosity
* Commandes :
  + syslog-ng-ctl stats
  + syslog-ng-ctl stats --reset

**Exemple avec JSON :**



**Tip :**

* Avant de procéder à l’envoie du message, le stocké dans un fichier local (ou sur stdout) afin de vérifier ce qui va être envoyé et de vérifié que la réception marche.
* Syslog-ng –syntax-only pour checker le fichier de configuration.
* Utiliser tcpdump pour vérifier l’envoie et la réception de logs.
* Syslog-ng –Fevd : syslog-ng va runner en foreground et accès aux messages debug sur stdout.
* Connexion non sécurisée, puis authentification serveur puis authentification client dans l’ordre.
* Même driver et options dans la destination du client et la source du serveur.