Brette Morgane BTS SIO 2 **2024/2025**

Guide de configuration de Rsyslog dans un environnement Linux

Table des matières

I- Configuration

1. Détail de l'infrastructure de centralisation de logs

Dans le cadre de la sécurisation et de la supervision du système d'information de la M2L, une infrastructure de centralisation des logs est mise en place, basée sur le service rsyslog.

Actuellement, cette configuration est déployée dans un environnement de test, constitué :

-d'un serveur de centralisation (Debian) configuré pour recevoir les journaux système et ;

-d'un poste client Linux configuré pour envoyer ses logs.

Ce scénario permet de valider la solution avant un déploiement plus large sur l'ensemble du parc Linux de la M2L.

2. Objectifs

Les objectifs ici sont :

- d'installer et configurer Rsyslog sur le serveur Debian pour recevoir les logs d'une machine distante ;

- de configurer le client Debian pour envoyer ses logs vers le serveur ;

- de tester la connectivité et la réception des journaux et ;

- d'améliorer la configuration Rsyslog avec la mise en place d'un filtrage, d'une séparation, d'une compression, et d'une rotation des logs.

3. Topologie du réseau

Tout d'abord, nous devons nous assurer que les deux machines soient dans le même réseau.

De ce fait, il faut au préalable vérifier l'adresse ip de chaque machine afin de s'assurer qu'elles sont dans le même réseau, sinon la centralisation de logs ne pourra pas fonctionner. Pour ce faire, on utilise la commande « **ip a** ».

Serveur de centralisation de logs

```
root@debian:/home/a# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:ed:cc:8e brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.10.146/24 brd 192.168.10.255 scope global dynamic noprefixroute ens33
       valid_lft 1602sec preferred_lft 1602sec
    inet6 fe80::20c:29ff:feed:cc8e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Poste client

```
root@debian:/home/a# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:18:1b:9f brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.10.147/24 brd 192.168.10.255 scope global dynamic noprefixroute ens33
       valid_lft 1382sec preferred_lft 1382sec
    inet6 fe80::20c:29ff:fe18:1b9f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Nous constatons que le serveur, ainsi que la machine cliente sont dans le même réseau.

Il existe une étape supplémentaire afin de s'en assurer en utilisant la commande « **ping adresseipmachine** ».

Poste client

```
root@debian:/home/a# ping 192.168.10.146
PING 192.168.10.146 (192.168.10.146) 56(84) bytes of data.
64 bytes from 192.168.10.146: icmp_seq=1 ttl=64 time=0.663 ms
64 bytes from 192.168.10.146: icmp_seq=2 ttl=64 time=0.475 ms
```

La commande confirme qu'on parvient à communiquer avec succès le serveur, il y a donc bien une connectivité réseau entre le serveur de centralisation de logs, et le poste client.
Les deux adresses appartiennent bien au même sous-réseau 192.168.10.0/24.
En résumé, voici les adresses IP de nos machines :
→Serveur Rsyslog (Debian) dont l'adresse IP est 192.168.10.146/24.
→Client Debian dont l'adresse ip est 192.168.10.147/24.

II. Configurer le serveur Rsyslog

1. Installer Rsyslog

(voir Guide d'installation de Rsyslog sur les systèmes d'exploitation de type Unix (ici Linux-Debian))

2. Configuration du serveur Rsyslog pour accepter les logs distants

Etape 1 : Ouvrir le fichier de configuration de Rsyslog

On doit ouvrir le fichier de configuration principal de Rsyslog en utilisant la commande « **nano /etc/rsyslog.conf** »

```
root@debian:/home/elk# nano /etc/rsyslog.conf
```

→Il faut décommenter les lignes suivantes afin d'activer le module UDP[1] + module TCP[2], qui sont permettre de transmettre les logs.

```
# provides UDP syslog reception
#module(load="imudp")
#input(type="imudp" port="514")

# provides TCP syslog reception
#module(load="imtcp")
#input(type="imtcp" port="514")
```

Donnant ça

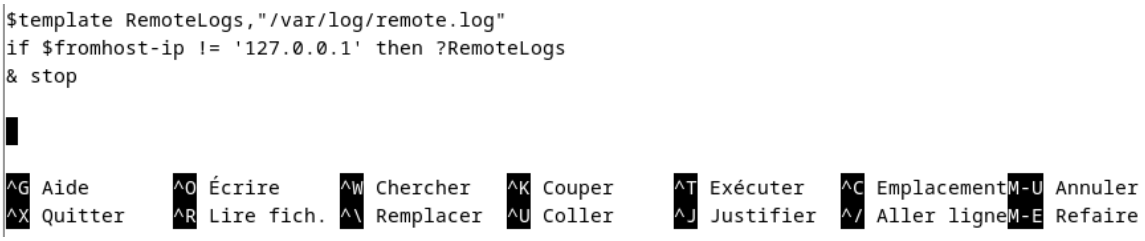
```
# provides UDP syslog reception
module(load="imudp")
input(type="imudp" port="514")

# provides TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")
```

[1] Protocole de transmission des données (*User Datagram Protocol*) sans connexion, rapide mais sans garantie que tous les paquets soient transmis.
[2] Protocole de transmission des données (*Transmission Control Protocol*) avec connexion, fiable mais plus lent que UDP, mais sûr de transmettre tous les paquets dans le bon ordre et sans perte.

→ Il faut aussi Toujours dans le fichier créer un fichier spécifique afin d'y stocker les logs distants en rajoutant à la fin du fichier le bloc
« **$template RemoteLogs,"/var/log/remote.log"**

if $fromhost-ip != '127.0.0.1' then ?RemoteLogs

& stop»



Détail de la commande :
* **$template RemoteLogs…** » définit le chemin du fichier où enregistré,
* **if $fromhost-ip != '127.0.0.1** cible uniquement les machines autres que le serveur lui-même
* **then ?RemoteLogs** utilise le template défini plus haut
***& stop** empêche d'enregistrer ces logs ailleurs (dans /var/log/syslog par exemple)

Puis :
→ sauvegarder : appuyer simultanément sur CTRL+X
→ Appuyer sur « o » pour sauvegarder le fichier modifier ,
→ Appuyer sur « ENTREE » pour valider et quitter

Maintenant il faut redémarrer le service Rsyslog avec la commande
« **systemctl restart rsyslog** »

```
root@debian:/home/elk# systemctl restart rsyslog
```

Etape 2 : Configurer le firewall

Nous devons configurer le pare-feu afin d'autoriser le trafic syslog.
Pour cela, nous devons vérifier que le pare-feu est installé en utilisant la commande
« **dpkg -l | grep ufw** »

```
root@debian:/home/elk# dpkg -l | grep  ufw
ii  ufw                          0.36.2-1                        all          program for managing a Netfilter firewall
```

Ici, la commande confirme que le pare-feu est bien installé
→sinon suivre procédure d'installation pour les paquets Rsyslog, car la procédure est identique.

Il faut rajouter les règles afin autoriser le trafic syslog avec la commande
« **ufw allow 514/udp** » + « **ufw allow 514/tcp** »

```
root@debian:/home/elk# ufw allow 514/udp
Rule added
Rule added (v6)
root@debian:/home/elk#  ufw allow 514/tcp
Rule added
Rule added (v6)
```

On vérifie que le pare-feu fonctionne avec la commande « **ufw status verbose** » permettant de voir si UFW activé, et quelles sont les règles appliquées

```
root@debian:/home/elk# ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip
root@debian:/home/elk# ufw allow 514/udp
Rule added
Rule added (v6)
root@debian:/home/elk#  ufw allow 514/tcp
Rule added
Rule added (v6)
```

La commande confirme que les port 514 UDP, et TCP sont ouverts.

3. Configurer la machine cliente

Etape : 1. Installer Rsyslog

(voir Guide d'installation de Rsyslog sur les systèmes d'exploitation de type Unix (ici Linux-Debian))

Etape 2 : Configurer Rsyslog

Maintenant nous devons configurer le poste client, afin qu'il envoie ses journaux au serveur de centralisation.

On doit ouvrir le fichier de configuration principal de Rsyslog avec la commande
« **nano /etc/rsyslog.conf** »

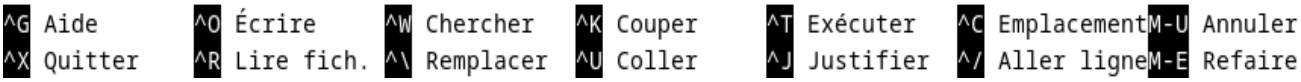
```
root@debian:/home/elk# nano /etc/rsyslog.conf
```

L'objectif est de permettre l'envoi de tous les logs au serveur Debian via UDP et TCP.
(voir 2. Configuration du serveur Rsyslog pour accepter les logs distants / Etape 1 : Ouvrir le fichier de configuration de Rsyslog)

Etape 3 : Rajout les lignes d'envoi

A la fin du fichier qu'on modifie, il faut rajouter les lignes suivantes
« ***.* @192.168.10.146:514** »
« ***.* @192.168.10.146:514** »
Sauvegarder puis quitter.

```
*.* @192.168.10.146:514
*.* @192.168.10.146:514
```



« ***.* @adresseip :514** » permettre l'envoi de tous les logs vers le serveur distant ayant l'adresse ip 192.168.10.146 via le port spécifique 514.

Si envoi via protocole UPD « *.* @192.168.10.133:514 »
Si envoi via protocole TCP « *.* @@192.168.10.133:514 »

Etape 4 : Redémarrer le service Rsyslog

Il faut redémarrer le service Rsyslog avec la commande
« **systemctl restart rsyslog** »

```
root@debian:/home/elk# systemctl restart rsyslog
```

Cette commande s'utilise après modification du fichier de configuration, afin d'appliquer les changements.
*__systemctl__ : outil de gestion des services sous systemd (utilisé par Debian, Ubuntu…) ;
*__restart__ : stoppe puis redémarre immédiatement le service et ;
*__rsyslog__ : c'est le démon responsable de la journalisation système (collecte des logs).

4. Vérification et validation

Etape 1 : Envoyer un message de test (client)

Sur machine cliente envoyer un message test avec la commande
« logger « mesagequ'onveutenvoyer » »

```
root@debian:/home/a# logger "Bonjour ça va ?"
```

Etape 2 : Vérifier la réception (serveur)

Sur le serveur vérifier qu'on a bien reçu le log dans « **/var/log/remote.log** » avec la commande « **tail -f /var/log/remote.log** »

```
2025-05-02T14:43:29+02:00 debian root: Bonjour ça va ?
```

Si le message de test s'affiche correctement dans le fichier de logs du serveur, cela confirme que la communication entre le client et le serveur fonctionne, et que la centralisation des journaux est opérationnelle.
La configuration de rsyslog entre un serveur Debian et une machine cliente Debian est donc réussie.

III- Améliorer Rsyslog

1. Filtrage des logs

Etape 1 : Isoler les logs

Dans cette étape, nous allons mettre en place un filtrage permettant d'isoler uniquement les messages de type "erreur" (niveau err) dans un fichier spécifique.
Cela permet de faciliter le diagnostic en regroupant les messages critiques dans un fichier unique, séparé du reste des journaux système.

Pour cela, nous allons créer un fichier de configuration personnalisé dans /etc/rsyslog.d/.

Étape 2 : Création du fichier de filtrage

Nous allons créer un fichier de configuration dédié à cette règle, ce qui permet de ne pas modifier directement le fichier principal /etc/rsyslog.conf et de garder une configuration plus propre.

Ouvre un nouveau fichier de configuration dans le répertoire /etc/rsyslog.d/

```
root@debian:/home/a# nano /etc/rsyslog.d/20-filter-errors.conf
```

Cette commande correspond à :
*/etc/rsyslog.d/ : répertoire des configurations personnalisées
*20-filter-errors.conf : nom du fichier

Puis, on doit y ajouter la règle dans le fichier
« ***.err /var/log/errors.log** »
Modifier, sauvegarder, quitter.

Étape 3 : Redémarrer Rsyslog et tester le filtrage

Redémarrer le service Rsyslog avec la commande
« systemctl restart rsyslog »

On envoie un message de test avec le bon niveau de gravité (ici err) depuis le poste client

```
root@debian:/home/a# logger -p user.err "Ceci est un message d'erreur de test"
```

Maintenant, on vérifie que le message a été enregistré dans /var/log/errors.log
« tail -f /var/log/errors.log »

```
root@debian:/home/a# tail -f /var/log/errors.log
2025-05-02T14:57:16+02:00 debian root: ceci ddsqdqsq
2025-05-02T14:57:16+02:00 debian root: ceci ddsqdqsq
2025-05-02T15:18:38+02:00 debian root: Ceci est un message d'erreur de test
2025-05-02T15:18:38+02:00 debian root: Ceci est un message d'erreur de test
```

⚠ Il est important de noter que tous les messages d'erreur ne sont pas de gravité « err ».
Voici un tableau rappelant les niveaux de gravité Syslog :

Niveau	Gravité	Description
0	emerg	Système inutilisable
1	alert	Action immédiate requise
2	crit	Conditions critiques
3	err	Erreurs
4	warning	Avertissements
5	notice	Conditions normales mais importantes
6	info	Informations
7	debug	Messages de débogage

2. Séparer les logs

C'est une étape essentielle afin de rendre l'analyse des évènements système plus claire, et organisée. Ici, nous allons séparer les logs selon les types de services (mail, authentification, cron, ...). L'essentiel étant d'éviter que tous les messages soient mélangés au sein d'un seul, et unique fichier.

Étape 1 : Comprendre la notion de "facility"

Chaque service système utilise une facility (code de classification) pour identifier le type de message qu'il envoie. Par exemple :

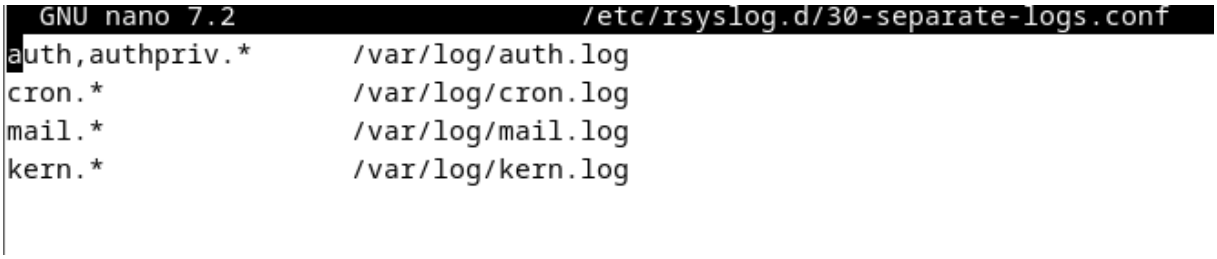
Facility	Service associé
Auth, authnpriv	Authentification (login, sudo, SSH, ...)
Cron	Tâches planifiées (cron jobs)
Mail	Services de messagerie
Kern	Noyau Linux
daemon	Services en arrière-plan

Étape 2 : Créer une règle de séparation

On peut créer un fichier dans /etc/rsyslog.d/ pour y mettre les règles de séparation.
Pour cela créer un fichier de configuration et le modifier en faisant
« **nano /etc/rsyslog.d/30-separate-logs.conf** »

```
root@debian:/home/a# nano /etc/rsyslog.d/30-separate-logs.conf
```

Puis, on y rajoute les règles suivantes



Sauvegarder, et quitter.
auth,authpriv. **/var/log/auth.log :** signifie que tous les messages de type « **auth** » iront dans **« /var/log/auth.log » ;**
cron. **/var/log/cron.log** : signifie que tous les logs « **cron** » iront dans **« /var/log/cron.log » ;**
mail. **/var/log/mail.log** : signifie que tous les logs « **mail** » iront dans **« /var/log/mail.log » et ;**
*** kern.*** **/var/log/kern.log** : signifie que tous les logs « **kern** » iront dans **« /var/log/kern.log ».**

Il faut redémarrer Rsyslog.

Étape 3 : Vérification

Une fois Rsyslog redémarré, il est nécessaire de générer des événements afin de tester si la centralisation et la séparation des logs fonctionnent correctement. On peut générer un événement, par exemple en ouvrant une session SSH, en exécutant une commande avec sudo, ou en provoquant une erreur volontaire.

→Sortir et repasser en mode « root »

Depuis le poste client

```
root@debian:/home/a# exit
exit
a@debian:~$ su root
Mot de passe :
root@debian:/home/a#
```

Il faut rentrer la commande pour afficher les logs d'authentification générer par nos tests dans« **tail -f /var/log/auth.log** » depuis le serveur de logs.

Depuis le serveur de centralisation de logs

```
2025-05-02T15:46:07+02:00 debian su[4585]: pam_unix(su:session): session opened for user root(uid=0)
by (uid=1000)
```

Test sur le serveur directement

```
2025-05-02T15:47:12.798622+02:00 debian su[5289]: pam_unix(su:session): session closed for user root
2025-05-02T15:47:15.541823+02:00 debian su[5336]: (to root) a on pts/0
2025-05-02T15:47:15.542107+02:00 debian su[5336]: pam_unix(su:session): session opened for user root
uid=0) by (uid=1000)
```

*__session closed for user root__ : Une session su en tant que root a été fermée.

*__(to root) au on pts/0__ : On est en train d'exécuter la commande su (ou su root) depuis un terminal (pts/0), dans le but de passer en utilisateur root.

3. Compresser les logs

L'objectif à travers la compression des logs est d'économiser de l'espace disque, et de conserver un historique propre et lisible. Pour se faire, on utilise « **logrotate** » qui est l'outil pour la gestion des logs sous Linux, donc sous Debian.

Étape 1 : Vérifie que logrotate est installé

On utilise la commande « **dpkg -t | grep logrotate** »

```
root@debian:/home/a# dpkg -l | grep logrotate
ii  logrotate                            3.21.0-1                        amd64        Log rota
tion utility
```

Il est déjà installé ici.

Étape 2 : Crée une règle personnalisée

On utilise la commande « **nano /etc/logrotate.d/errors-log** »

```
root@debian:/home/a# nano /etc/logrotate.d/errors-log
```

Puis, on colle la configuration suivante dedans :

/var/log/errors.log {

  weekly

  rotate 4

  compress

  missingok

  notifempty

  create 0640 syslog adm

}

Qui veut dire :
*weekly : rotation des logs chaque semaine ;
*rotate 4 : on garde les 4 dernières versions ;
*compress : on compresse les anciens fichiers en .gz ;
*notifempty : on ne fait rien si le fichier est vide ;
*missingok : on ignore si le fichier est absent et ;
*create : on recrée le fichier vide avec les bons droits.

Étape 3 : Tester manuellement la rotation

On peut forcer un test de rotation avec
« **logrotate -f /etc/logrotate.d/errors-log** »

```
root@debian:/home/a# logrotate -f /etc/logrotate.d/errors-log
```

Puis, on regarde avec « **ls -lh /var/log/errors.log*** »

```
root@debian:/home/a# ls -lh /var/log/errors.log*
-rw-r----- 1 a    adm   0  2 mai   15:57 /var/log/errors.log
-rw-r----- 1 root adm 121  2 mai   15:18 /var/log/errors.log.1.gz
```

On y voit le nouveau fichier vide « **errors.log** » et l'ancienne version compressée
« **errors.log.1.gz** ».

ls -lh /var/log/errors.log* qui signifie :
***ls** : lister le contenu d'un dossier ;
* **-l** : format long (affiche les permissions, propriétaires, taille, date, ...) ;
* **-h** Human-readable (taille en Ko/Mo au lieu d'octets bruts) et ;
/var/log/errors.log : on liste tous les fichiers dont le nom commence par « errors.log ».

4. Rotation des logs

Pour la rotation des logs, nous utilisons toujours Logrotate.

Étape 1 : Modifier le fichier de configuration personnalisé créer en amont

On utilise la commande « nano /etc/logrotate.d/errors-log »

```
root@debian:/home/a# nano /etc/logrotate.d/errors-log
```

```
  GNU nano 7.2
/var/log/errors.log {
    weekly
    size 50k
    rotate 4
    compress
    missingok
    notifempty
    create 0640 root adm
}
```

Nous rajoutons « **Size 50k** » (rotation quand le fichier atteinte 50 Ko) comme critère de rotation, en plus de « **weekly** » donc Logrotate déclenchera automatiquement la rotation si l'un de ces 2 critères est atteint.

Etape 2 : Forcer une rotation

On utilise la commande « **logrotate -f /etc/logrotate.d/errors-log** »

```
root@debian:/home/a# logrotate -f /etc/logrotate.d/errors-log
```

Si tout est bon, on verras dans « **/var/log** » en faisant la commande « **ls -lh /var/log/errors.log*** » 2 fichiers comme en amont.

```
root@debian:/home/a# ls -lh /var/log/errors.log*
-rw-r----- 1 a    adm   0  2 mai   15:57 /var/log/errors.log
-rw-r----- 1 root adm 121  2 mai   15:18 /var/log/errors.log.1.gz
```

De ce fait, le fichier « **errors.log** » est réinitialisé (vide), et l'ancien fichier est archivé et compressé, on peut l'ouvrir en faisant « **zcat /var/log/errors.log.1.gz** »

```
root@debian:/home/a# zcat /var/log/errors.log.1.gz
2025-05-02T14:57:16+02:00 debian root: ceci ddsqdqsq
2025-05-02T14:57:16+02:00 debian root: ceci ddsqdqsq
2025-05-02T15:18:38+02:00 debian root: Ceci est un message d'erreur de test
2025-05-02T15:18:38+02:00 debian root: Ceci est un message d'erreur de test
```

Si on ouvre l'autre, ce dernier n'affiche donc rien, et est bien vide.

```
root@debian:/home/a# cat /var/log/errors.log
root@debian:/home/a#
```