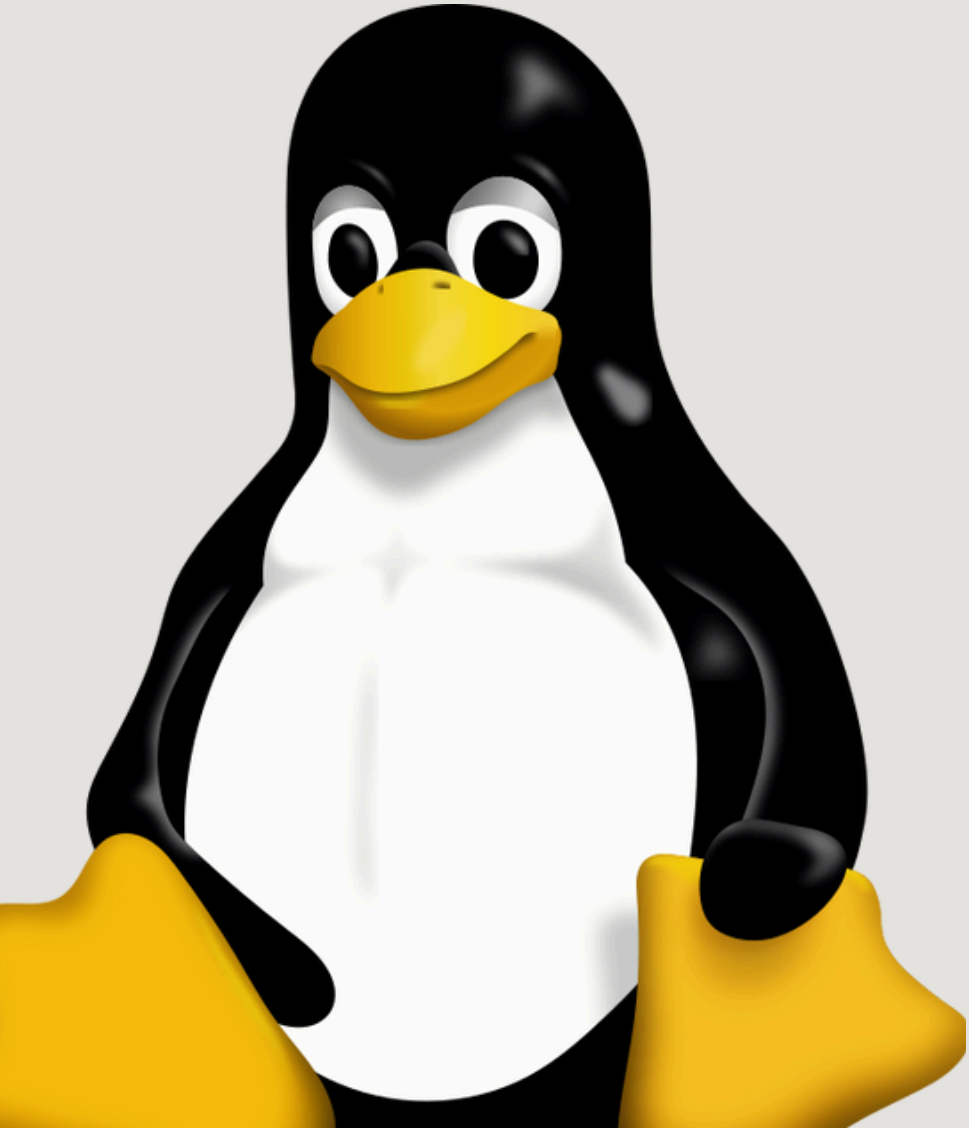
Rapport Linux

BEN MAAMAR Rayan
LANSARI Ayoub
MENE Hadi
NADHIF Zouhour
SAADANI Elias
TARFAYA Kheireddine



Sommaire

du rapport

1 Introduction page 2

2 NTP page 4

3 FireWall page 8

4 DNS page 9

5 DHCP page 11

6 Wazuh page 14

7 SSH page 21

8 RsysLog page 26

9 Docker page 28

10 Sauvegarde page 39

11 Conclusion page 42

1 Introduction

PRÉSENTATION DU PROJET

Ce projet a été conçu dans le cadre de la gestion d'un parc informatique complexe et la mise en place de mécanismes d'automatisation pour garantir la sécurité, la surveillance et la maintenance des services critiques. L'objectif principal était de centraliser la gestion de l'infrastructure, d'automatiser les tâches répétitives et d'assurer la fiabilité des services grâce à des politiques de sécurité et de supervision adaptées.

Tous les services ont été déployés et configurés sur des machines virtuelles, offrant ainsi une grande flexibilité pour tester, gérer et isoler les services tout en minimisant les risques d'impact sur l'infrastructure physique. Cette virtualisation nous a permis de simuler des scénarios réalistes tout en facilitant la gestion et le déploiement de nouvelles configurations.

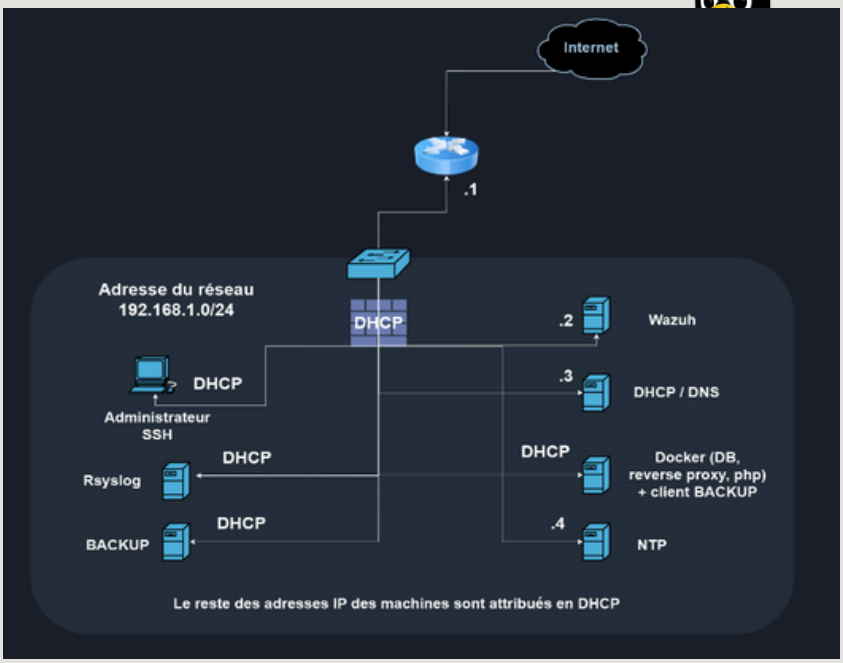
Chaque service a été mis en place selon les bonnes pratiques de gestion et de sécurité, avec un accent particulier mis sur l'automatisation des tâches administratives, la surveillance des services critiques et la mise en œuvre de règles de sécurité strictes. Grâce à des outils comme Wazuh et Docker, le projet visait non seulement à déployer et sécuriser l'infrastructure, mais aussi à faciliter les processus de supervision, de maintenance et de sauvegarde.

Le projet a été structuré en plusieurs étapes, allant de l'installation et la configuration des services jusqu'à l'automatisation des connexions sécurisées SSH et la supervision proactive des machines avec Wazuh. À travers ce projet, nous avons démontré la capacité à gérer efficacement une infrastructure réseau tout en minimisant les interventions manuelles par l'automatisation de tâches critiques.

PRÉSENTATION DU PARC INFORMATIQUE

Notre parc informatique virtuel se compose des services et serveurs suivants :

- NTP : Serveur de synchronisation de l'heure (IP : 192.168.1.4), garantissant une cohérence temporelle entre les différentes machines virtuelles du réseau.
- Firewall (iptables) : Configuration et application de politiques de sécurité pour contrôler les flux de données entrants et sortants.
- DNS : Serveur de noms de domaine pour la résolution des adresses IP.
- DHCP : Serveur de gestion dynamique de l'attribution des adresses IP.
- Wazuh : Outil de surveillance et de détection d'intrusions permettant de suivre en temps réel les événements de sécurité .
- SSH : Mise en place de connexions sécurisées sans mot de passe entre les différentes machines virtuelles du parc.
- RSYSLOG : Collecte centralisée des journaux système pour une analyse proactive des incidents.
- Docker : Déploiement et gestion d'applications via conteneurs pour simplifier la virtualisation.
- Sauvegarde : Implémentation de stratégies de sauvegarde pour garantir la protection des données critiques.



L'utilisation de machines virtuelles nous a permis de consolider la gestion de notre parc informatique tout en améliorant la sécurité, la flexibilité et la réactivité face aux événements de sécurité.

Le service NTP (Network Time Protocol) sert à synchroniser l'heure entre différents systèmes dans notre réseau. Cela va permettre à nos processus d'être dans la bonne horloge sachant que les processus et les transactions dépendent d'horodatages précis, comme les logs et la sécurité. Dans notre configuration, on a assigné l'adresse IP 192.168.1.4 au serveur NTP. Les autres machines du réseau vont interroger ce serveur pour obtenir l'heure correcte, ce qui va garantir une synchronisation horaire sur l'ensemble du réseau.

```
etudiant@debian:~$ sudo apt install ntp
[sudo] Mot de passe de etudiant :
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
ntp est déjà la version la plus récente (1:4.2.8p15+dfsg-1).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
etudiant@debian:~$
```

On va éditer le fichier de configuration NTP (/etc/ntp.conf) pour configurer les serveurs de temps auxquels votre serveur NTP va se synchroniser. Dans notre cas, on va choisir des serveurs publics (comme ceux de pool.ntp.org).

```
  GNU nano 5.4                                  /etc/ntp.conf *
# pick a different set every time it starts up.  Please consider joining the
# pool: <http://www.pool.ntp.org/join.html>
#pool 0.debian.pool.ntp.org iburst
#pool 1.debian.pool.ntp.org iburst
#pool 2.debian.pool.ntp.org iburst
#pool 3.debian.pool.ntp.org iburst
server 0.fr.pool.ntp.org
server 1.fr.pool.ntp.org
server 2.fr.pool.ntp.org
server 3.fr.pool.ntp.org

# Access control configuration; see /usr/share/doc/ntp-doc/html/accopt.html for
# details.  The web page <http://support.ntp.org/bin/view/Support/AccessRestrict
# might also be helpful.
#
# Note that "restrict" applies to both servers and clients, so a configuration
# that might be intended to block requests from certain clients could also end
# up blocking replies from your own upstream servers.
```

Autorisations: Dans le même fichier (/etc/ntp.conf), On va autoriser l'accès aux clients en ajoutant une ligne qui permet aux autres machines de notre réseau de se synchroniser avec ce serveur en utilisant la commande suivante : ***restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap***

```
# Needed for adding pool entries
restrict source notrap nomodify noquery


restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

Redémarrage du service NTP : Après avoir modifié la configuration, on redémarre le service NTP pour appliquer les changements : sudo systemctl restart ntp

On vérifie que le service est activé pour démarrer au lancement de la machine : sudo systemctl enable ntp

Synchronisation : On vérifie ensuite que le serveur est bien synchronisé avec un serveur de temps externe en utilisant la commande : ntpq -p

```
etudiant@ntp-server:~$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*176-137-36-37.a .GPS.            1 u   74  128  377   16.319   +0.923   6.160
+51-158-147-92.r 130.149.17.8     2 u   87  128  377   14.402   -0.437   2.830
+meshflow.net    192.150.70.56    2 u   15  128  377   15.165   +0.965   2.971
+time.cloudflare 10.19.13.133     3 u   82  128  377    5.294   +0.860   2.183
etudiant@ntp-server:~$
```

Ici on peut voir les 4 serveurs que j'ai rajouté, on a par exemple cloudflare et meshflow.net
remote : On est bien connecté à par exemple time.cloudflare
reach : 377 pour les deux serveurs, donc la connexion est stable.
delay : Latence de 5,294 ms pour le dernier serveur.
offset : La différence de temps avec le premier serveur est de -0,437 ms, et avec le second, c'est pratiquement synchronisé.
jitter : Le jitter est faible, ce qui indique que le délai de transmission est assez stable.

Pour vous montrer la configuration finale, voici une commande for fun que le prof m'a montré pour afficher l'ensemble des configurations que j'ai fait sans les commentaire, pour cela on va utiliser la commande suivante : grep -v '#' /etc/ntp.conf

```
Last login: Thu Oct 24 14:18:03 CEST 2024 from 10.0.2.2 on pts/1
etudiant@ntp-server:~$ grep -v '#' /etc/ntp.conf

driftfile /var/lib/ntp/ntp.drift

leapfile /usr/share/zoneinfo/leap-seconds.list


statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable



server 0.fr.pool.ntp.org iburst
server 1.fr.pool.ntp.org iburst
server 2.fr.pool.ntp.org iburst
server 3.fr.pool.ntp.org iburst


restrict -4 default kod notrap nomodify nopeer noquery limited
restrict -6 default kod notrap nomodify nopeer noquery limited

restrict 127.0.0.1
restrict ::1

restrict source notrap nomodify noquery


restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

Côté Client NTP :

Avant tout, on rappelle que le serveur NTP à pour adresse IP 192.168.1.4, il faut donc que sur tous les clients soit configuré avec l'IP du serveur.

On installe ntp avec la commande suivante : sudo apt install ntp

Ensuite, on va configurer le fichier de configuration NTP : /etc/ntp.conf pour définir le serveur NTP (la VM Serveur) auquel le client va se synchroniser. On va ajouter la ligne suivante <ip-du-serveur> par l'adresse IP de votre VM Serveur :

ce la va donner la ligne suivante : server <192.168.1.20> iburst

```
#pool 3.debian.pool.ntp.org iburst
server 192.168.1.4 iburst
```

Ensuite, on va redémarrer le service NTP : Redémarrez le service NTP pour prendre en compte la configuration :sudo systemctl restart ntp

On s'assure également que le service NTP est activé pour démarrer au démarrage de la machine :sudo systemctl enable ntp

```
etudiant@debian:~$ sudo systemctl enable ntp
Synchronizing state of ntp.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ntp
```

Vérifier la synchronisation : Pour ce faire, on utilise la commande suivante pour vérifier si le client est bien synchronisé avec le serveur NTP :

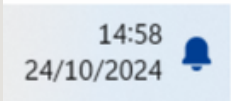
ntpq -p

```
etudiant@dhcp-server:~$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 192.168.1.4     176.137.36.37    2 u    1   64    1    2.299   -2.631   0.000
```

On peut voir dans l'encadré rouge l'adresse IP du serveur NTP, le client est donc bien synchronisé avec le serveur NTP.
remote : On est bien connecté à 192.168.1.4 (IP du serveur NTP configuré précédemment).
reach : 1 pour la connexion uniquement à ce serveur, donc la connexion est stable.
delay : Latence de 2.299 ms pour se connecter au serveur (relativement faible).
offset : La différence de temps avec le premier serveur est de -2,631 ms, c'est synchronisé.
jitter : Le jitter est faible, même nul (0.000) ce qui indique que le délai de transmission est parfaitement stable.

Maintenant sur le client on peut tester la date en utilisant la commande "date" et vérifier si c'est l'horloge est bien configuré avec l'heure d'aujourd'hui:

```
etudiant@dhcp-server:~$ date
jeu. 24 oct. 2024 14:57:51 CEST
etudiant@dhcp-server:~$
```



Commencez par effacer toutes les règles existantes pour commencer avec une configuration propre : sudo iptables -F

- Définir des politiques par défaut

Mettez en place des politiques par défaut pour autoriser et bloquer le trafic :

```
etudiant@debian:~$ sudo iptables -P INPUT DROP
etudiant@debian:~$ sudo iptables -P FORWARD DROP
etudiant@debian:~$ sudo iptables -P OUTPUT ACCEPT
```

- Autoriser le trafic local et le trafic SSH

Autorisez le trafic sur l'interface de loopback et les connexions SSH :

```
tudiant@debian:~$ sudo iptables -A INPUT -i lo -j ACCEPT
tudiant@debian:~$ sudo iptables -A OUTPUT -o lo -j ACCEPT
tudiant@debian:~$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

- Autoriser le trafic depuis le réseau local

Comme le pare-feu est sur le réseau 192.168.1.0/24, on doit permettre le trafic de ce réseau :

```
etudiant@debian:~$ sudo iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT
etudiant@debian:~$ sudo iptables -A OUTPUT -d 192.168.1.0/24 -j ACCEPT
```

- Autoriser le trafic ICMP (ping)

Si vous voulez que d'autres machines sur le réseau puissent "pinguer" votre pare-feu, vous pouvez permettre les requêtes ICMP :

```
etudiant@debian:~$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
```

Installation des paquets nécessaires au service :

sudo apt update

sudo apt install bind9 bind9utils bind9-doc

Fichiers de configuration :

Dans un premier temps, nous avons configuré le fichier named.conf.local qui permet de définir et de configurer les domaines spécifiques gérés localement par le serveur BIND, facilitant la gestion de la configuration DNS pour des zones spécifiques sans impacter la configuration générale du serveur. Dans notre cas nous avons ajouté la zone “groupe-5.com”. Nous l’avons mis en type master ce qui signifie qu’il s’agit du serveur DNS principal pour la zone. La ligne file “/etc/bind/db.groupe-5.com” spécifie le chemin vers le fichier de configuration de cette zone.

```
GNU nano 5.4                                    named.conf.local
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "groupe-5.com" {
    type master;
    file "/etc/bind/db.groupe-5.com";
};
```

Vérification des fichiers :

sudo named-checkconf

sudo named-checkzone groupe-5.com /etc/bind/db.groupe-5.com

Redémarrer et autoriser au démarrage :

sudo systemctl restart bind9

sudo systemctl enable bind9

```
GNU nano 5.4                                                    /etc/bind/db.groupe-5.com
$TTL 86400
@   IN  SOA ns1.example.com. admin.example.com. (
            2023102301 ; Serial
            3600       ; Refresh
            1800       ; Retry
            1209600    ; Expire
            86400 )    ; Minimum TTL

; Serveurs de noms
@   IN  NS  ns1.groupe-5.com.
@   IN  NS  ns2.groupe-5.com.

; Adresses IP des serveurs de noms
ns1 IN  A   192.168.1.3
ns2 IN  A   192.168.1.3

; Enregistrements A (Adresses IP)
dhcp-server IN  A   192.168.1.3
wazuh-server IN A   192.168.1.2
application IN A    192.168.1.26
```

Le service DNS BIND9 est actif, ce qui est confirmé par le statut en cours d'exécution. Les logs indiquent des tentatives de résolution de noms pour divers domaines. Les messages "network unreachable resolving" font référence à des requêtes pour des domaines externes (liées à la configuration initiale), mais cela n'affecte pas la résolution des noms locaux dans notre configuration.

```
etudiant@debian:/etc/bind$ sudo systemctl status bind9.service
● named.service - BIND Domain Name Server
     Loaded: loaded (/lib/systemd/system/named.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2024-10-24 09:14:19 CEST; 8min ago
       Docs: man:named(8)
   Main PID: 2159 (named)
      Tasks: 6 (limit: 2323)
     Memory: 16.0M
        CPU: 94ms
     CGroup: /system.slice/named.service
             └─2159 /usr/sbin/named -f -u bind

oct. 24 09:14:19 debian named[2159]: network unreachable resolving './DNSKEY/IN': 2001:dc3::35#53
oct. 24 09:14:19 debian named[2159]: network unreachable resolving './NS/IN': 2001:dc3::35#53
oct. 24 09:14:19 debian named[2159]: network unreachable resolving './DNSKEY/IN': 2001:500:12::d0d#53
oct. 24 09:14:19 debian named[2159]: network unreachable resolving './NS/IN': 2001:500:12::d0d#53
oct. 24 09:14:19 debian named[2159]: network unreachable resolving './DNSKEY/IN': 2001:503:c27::2:30#53
oct. 24 09:14:19 debian named[2159]: network unreachable resolving './NS/IN': 2001:503:c27::2:30#53
oct. 24 09:14:19 debian named[2159]: network unreachable resolving './DNSKEY/IN': 2001:7fd::1#53
oct. 24 09:14:19 debian named[2159]: network unreachable resolving './NS/IN': 2001:7fd::1#53
oct. 24 09:14:19 debian named[2159]: managed-keys-zone: Key 20326 for zone . is now trusted (acceptance
oct. 24 09:14:19 debian named[2159]: resolver priming query complete
```

Ce fichier spécifie les serveurs DNS et les domaines de recherche pour le serveur :

- Domaine : groupe-5.com
- Serveur DNS primaire : 192.168.1.3 (serveur DNS local)
- Serveur DNS secondaire : 8.8.8.8 (serveur DNS public de Google)

```
  GNU nano 5.4                                  /etc/resolv.conf
domain groupe-5.com
search groupe-5.com
nameserver 192.168.1.3
nameserver 8.8.8.8
nameserver 8.8.8.8
```

La commande nslookup montre que le serveur DNS est capable de résoudre le nom de domaine application.groupe-5.com vers l'adresse IP 192.168.1.26. Cette résolution valide la configuration correcte du service DNS et confirme que le serveur est opérationnel pour la résolution de noms au sein du domaine groupe-5.com.

```
etudiant@dhcp-server:/etc/bind$ nslookup application.groupe-5.com
Server:         192.168.1.3
Address:        192.168.1.3#53

Name:   application.groupe-5.com
Address: 192.168.1.26
```

Ce fichier est la configuration principale du DHCP. Voici les paramètres essentiels définis :

- Nom de domaine : groupe-5.com
- Serveurs de noms de domaine : ns1.groupe-5.com et ns2.groupe-5.com
- Durée de bail :
  - Bail par défaut : 6000 secondes
  - Bail maximum : 7200 secondes
- Sous-réseau 192.168.1.0/24 :
  - Plage d'adresses IP : 192.168.1.10 à 192.168.1.50
  - Passerelle par défaut : 192.168.1.1
  - Adresse de diffusion : 192.168.1.255
  - Serveurs DNS : 192.168.1.3 et 8.8.8.8

En outre, des adresses IP statiques sont attribuées à des machines spécifiques (comme wazuh-server avec l'adresse 192.168.1.2 et dhcp-server avec l'adresse 192.168.1.3), ce qui garantit une configuration réseau cohérente et fiable pour des services critiques.

```
GNU nano 5.4                    /etc/dhcp/dhcpd.conf
# dhcpd.conf
#
# Sample configuration file for ISC dhcpd
#

# Définition le domaine
option domain-name "groupe-5.com";
option domain-name-servers ns1.groupe-5.com, ns2.groupe-5.com;

# Configuration du bail par défaut
default-lease-time 6000;
max-lease-time 7200;

# Autorisation du démarrage du DHCP
authoritative;

# Configuration du sous-réseau
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.10 192.168.1.50;  # Plage d'adresses IP à attribuer
    option routers 192.168.1.1;         # Passerelle par défaut
    option subnet-mask 255.255.255.0;   # Masque de sous-réseau
    option broadcast-address 192.168.1.255; # Adresse de diffusion
    option domain-name-servers 192.168.1.3, 8.8.8.8; # Serveurs DNS
}

# Adresses statiques
host wazuh-server {
    hardware ethernet 08:00:27:0c:e7:c6;
    fixed-address 192.168.1.2;
}

host dhcp-server {
    hardware ethernet 08:00:27:fb:e8:7a;
    fixed-address 192.168.1.3;
}

host ntp-server {
    hardware ethernet 08:00:27:75:01:54;
    fixed-address 192.168.1.4;
}
```

Dans ce fichier, nous avons configuré l'interface réseau utilisée par le service DHCP :

- INTERFACESv4="enp0s8" : Spécifie que le service DHCP écoutera les requêtes sur l'interface enp0s8, correspondant à l'interface configurée pour le sous-réseau 192.168.1.0/24.

```
  GNU nano 5.4                    /etc/default/isc-dhcp-server
# Defaults for isc-dhcp-server (sourced by /etc/init.d/isc-dhcp-server)

# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
#DHCPDv4_CONF=/etc/dhcp/dhcpd.conf
#DHCPDv6_CONF=/etc/dhcp/dhcpd6.conf

# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPDv4_PID=/var/run/dhcpd.pid
#DHCPDv6_PID=/var/run/dhcpd6.pid

# Additional options to start dhcpd with.
#       Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
#       Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACESv4="enp0s8"
INTERFACESv6=""
```

Le service DHCP est actif (en cours d'exécution). Le log du service présente différentes transactions DHCP :

- DHCPDISCOVER et DHCPOFFER : Le serveur DHCP reçoit des requêtes de découverte d'adresses IP (DISCOVER) et envoie des offres (OFFER) en réponse.
- DHCPREQUEST et DHCPACK : Les clients acceptent une adresse IP proposée (REQUEST) et le serveur accorde cette adresse (ACK), ce qui complète le processus de configuration IP dynamique pour le client.

Ces messages montrent le bon fonctionnement du serveur DHCP.

```
etudiant@dhcp-server:/etc/bind$ sudo systemctl status isc-dhcp-server.service
● isc-dhcp-server.service - LSB: DHCP server
     Loaded: loaded (/etc/init.d/isc-dhcp-server; generated)
     Active: active (running) since Thu 2024-10-24 14:02:05 CEST; 1h 43min ago
       Docs: man:systemd-sysv-generator(8)
    Process: 4835 ExecStart=/etc/init.d/isc-dhcp-server start (code=exited, status=0/SUCCESS)
      Tasks: 4 (limit: 2323)
     Memory: 4.8M
        CPU: 63ms
     CGroup: /system.slice/isc-dhcp-server.service
             └─4850 /usr/sbin/dhcpd -4 -q -cf /etc/dhcp/dhcpd.conf enp0s8

oct. 24 15:36:31 dhcp-server dhcpd[4850]: DHCPDISCOVER from 60:45:cb:21:8a:46 (DESKTOP-M4KC24M) via enp0s8
oct. 24 15:36:31 dhcp-server dhcpd[4850]: DHCPOFFER on 192.168.1.27 to 60:45:cb:21:8a:46 (DESKTOP-M4KC24M) via enp0s8
oct. 24 15:36:31 dhcp-server dhcpd[4850]: reuse_lease: lease age 38 (secs) under 25% threshold, reply with unaltered, existing lease for 192.168.1.27
oct. 24 15:36:31 dhcp-server dhcpd[4850]: DHCPREQUEST for 192.168.1.27 (192.168.1.3) from 60:45:cb:21:8a:46 (DESKTOP-M4KC24M) via enp0s8
oct. 24 15:36:31 dhcp-server dhcpd[4850]: DHCPACK on 192.168.1.27 to 60:45:cb:21:8a:46 (DESKTOP-M4KC24M) via enp0s8
oct. 24 15:44:49 dhcp-server dhcpd[4850]: DHCPRELEASE of 192.168.1.26 from 08:00:27:23:6e:e0 (appli-web) via enp0s8 (found)
oct. 24 15:45:03 dhcp-server dhcpd[4850]: DHCPDISCOVER from 08:00:27:23:6e:e0 via enp0s8
oct. 24 15:45:04 dhcp-server dhcpd[4850]: DHCPOFFER on 192.168.1.26 to 08:00:27:23:6e:e0 (appli-web) via enp0s8
oct. 24 15:45:04 dhcp-server dhcpd[4850]: DHCPREQUEST for 192.168.1.26 (192.168.1.3) from 08:00:27:23:6e:e0 (appli-web) via enp0s8
oct. 24 15:45:04 dhcp-server dhcpd[4850]: DHCPACK on 192.168.1.26 to 08:00:27:23:6e:e0 (appli-web) via enp0s8
```

Les interfaces réseau configurées sur le serveur, avec leurs adresses IP et leurs paramètres associés.
Voici les détails concernant chacune des interfaces :

- Interface eth0 : Cette interface est configurée pour obtenir une adresse IP dynamique attribuée par le DHCP de l'école, qui est 10.0.2.15 dans ce cas. Nous avons choisi de maintenir une connexion sur ce réseau externe pour permettre des installations supplémentaires et pour garantir un accès à Internet en cas de besoin.
- Interface eth1 : Cette interface est utilisée pour le réseau local de notre parc informatique virtuel, avec une adresse IP fixe, 192.168.1.2, attribuée par notre propre serveur DHCP (voir le fichier de configuration du DHCP). Cette interface est dédiée aux services internes du réseau local et permet de gérer les clients du sous-réseau 192.168.1.0/24.

Ces deux interfaces permettent de séparer les connexions réseau : eth0 pour un accès externe, et eth1 pour le réseau local, assurant ainsi une bonne isolation et gestion des services pour nos clients internes.

```
etudiant@wazuh-server:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN grou
p 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: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
 UP group default qlen 1000
    link/ether 08:00:27:df:89:5b brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
       valid_lft 79530sec preferred_lft 79530sec
    inet6 fe80::a00:27ff:fedf:895b/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
 UP group default qlen 1000
    link/ether 08:00:27:0c:e7:c6 brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    inet 192.168.1.2/24 brd 192.168.1.255 scope global dynamic eth1
       valid_lft 4221sec preferred_lft 4221sec
    inet6 fe80::a00:27ff:fe0c:e7c6/64 scope link
       valid_lft forever preferred_lft forever
```

- Installation de Wazuh

curl -sO https://packages.wazuh.com/4.7/wazuh-install.sh && sudo bash ./wazuh-install.sh.

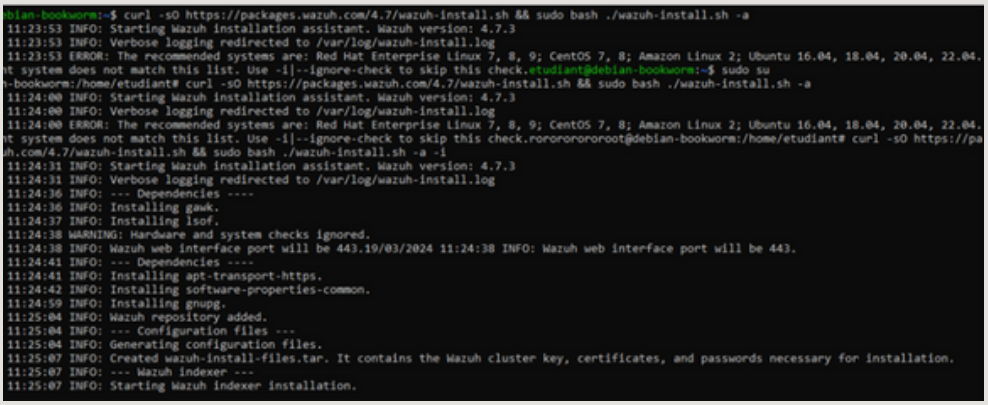


Figure 2 - Installation de Wazuh

- User et mot de passe

admin
30+k1THU7R7xTFe1G8qqQt+TgEPdj9bY

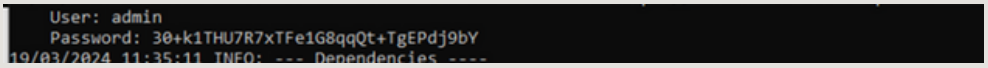


Figure 3 - Login Wazuh

- Les règles personnalisées wazuh doivent être configurées dans le fichier :

/var/ossec/etc/rules/local_rules.xml
Wazuh est accessible depuis le navigateur sur le port 443 :

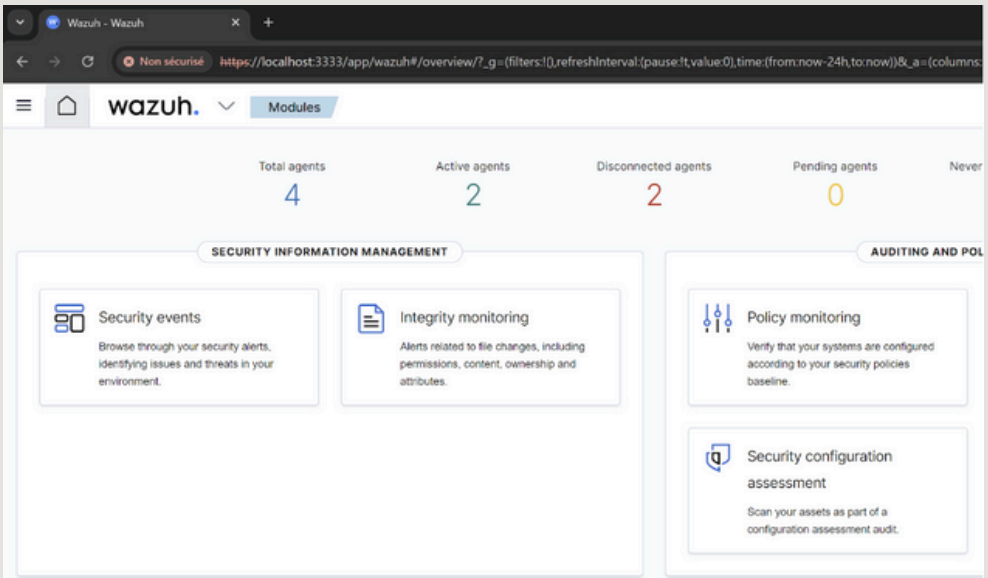


Figure 4 - Interface Wazuh

- Ajout d’un agent depuis l’interface web Wazuh (du serveur)

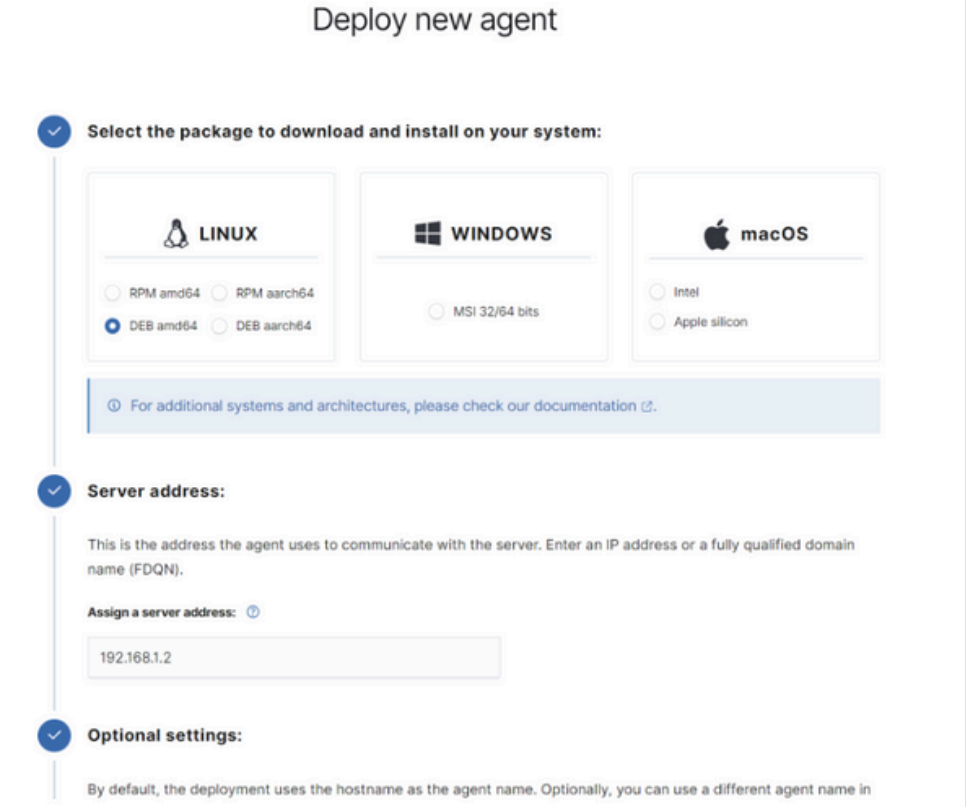


Figure 5 - Ajout d'un agent sur Wazuh

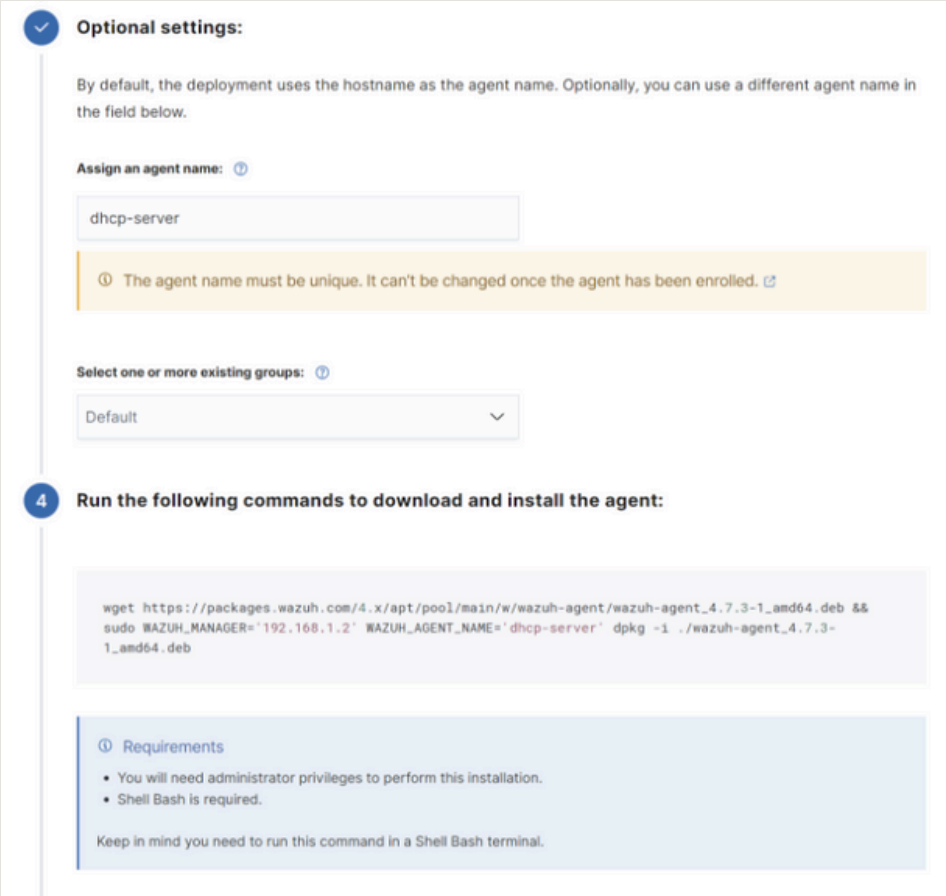


Figure 6 - Ajout d'un agent sur Wazuh 2

- Sur l'agent, machine sur laquelle on fait une surveillance

On copie colle la commande suivante :

```
wget https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/wazuh-agent_4.7.3-1_amd64.deb &&
sudo WAZUH_MANAGER='192.168.1.2' WAZUH_AGENT_NAME='dhcp-server' dpkg -i ./wazuh-agent_4.7.3-
1_amd64.deb
```

Figure 7 - Commande à exécuter sur l'agent pour configuration de celui-ci

Le nom de l'agent doit correspondre au hostname.
On exécute les commandes suivantes sur l'agent :

· sudo systemctl daemon-reload
· sudo systemctl enable wazuh-agent
· sudo systemctl start wazuh-agent
·wget https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/wazuh-agent_4.7.3-1_amd64.deb && sudo WAZUH_MANAGER='172.16.110.41' WAZUH_AGENT_NAME='serveur-prod-hadi' dpkg -i ./wazuh-agent_4.7.3-1_amd64.deb

- Notre agent a été ajouté, on peut le voir dans l'interface wazuh :

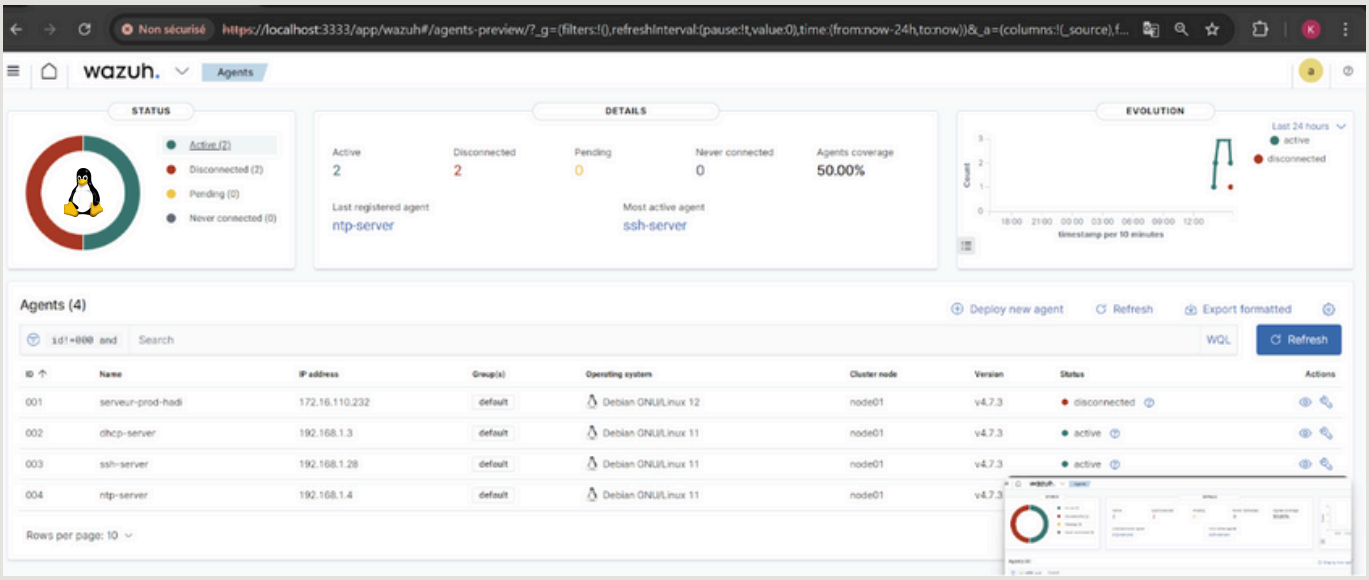


Figure 8 - Agent ajouté, visible sur Wazuh

- Dans le fichier , nous avons ajouté les lignes suivantes /var/ossec/etc/ossec.conf du manager :

```
  <localfile>
    <log_format>apache</log_format>
    <location>/var/log/apache2/access.log</location>
  </localfile>

<localfile>
    <log_format>syslog</log_format>
    <location>/var/log/auth.log</location>
</localfile>
```

Cela va permettre au serveur OSSEC de surveiller les fichiers en question, en l'occurrence ici il pourra surveiller le fichier access.log d'apache.

- Détection d'exécution d'un programme suspicieux

Afin de retrouver si un utilisateur exécute des commandes malicieuses on ajoute la ligne suivante dans le fichier /var/ossec/etc/ossec.conf du manager :
<list>etc/lists/suspicious-programs</list>

```
etudiant@debian-bookworm:~$ sudo cat /var/ossec/etc/lists/audit-keys
audit-wazuh-w:write
audit-wazuh-r:read
audit-wazuh-a:attribute
audit-wazuh-x:execute
audit-wazuh-c:command
```

Ajout de la règle pour détection de commande suspicieuse dans le fichier /var/ossec/etc/rules/local_rules.xml du manager :

```
<group name="audit">
  <rule id="100210" level="12">
    <if_sid>80792</if_sid>
                <list field="audit.command" lookup="match_key_value"
check_value="red">etc/lists/suspicious-programs</list>
                <description>Audit: Highly Suspicious Command executed:
$(audit.exe)</description>
    <group>audit_command,</group>
  </rule>
</group>
```

On a par la suite installé Auditd sur l'agent, qui permet de loger toutes les commandes exécutées sur un système. On a ensuite ajouté les lignes présentes dans la capture suivante dans le fichier /etc/audit/audit.rules :

```
root@serveur-prod-hadi:/home/etudiant# cat /etc/audit/audit.rules
## This file is automatically generated from /etc/audit/rules.d
-D
-b 8192
-f 1
--backlog_wait_time 60000

-a exit,always -F auid=1000 -F egid!=994 -F auid!=-1 -F arch=b32 -S execve -k audit-wazuh-c
-a exit,always -F auid=1000 -F egid!=994 -F auid!=-1 -F arch=b64 -S execve -k audit-wazuh-c
root@serveur-prod-hadi:/home/etudiant#
```

Listage des règles :

```
root@serveur-prod-hadi:/home/etudiant# sudo auditctl -l
-a always,exit -F arch=b32 -S execve -F auid=1000 -F egid!=994 -F auid!=-1 -F key=audit-wazuh-c
-a always,exit -F arch=b64 -S execve -F auid=1000 -F egid!=994 -F auid!=-1 -F key=audit-wazuh-c
root@serveur-prod-hadi:/home/etudiant#
```

Ajout sur l'agent sur le fichier /var/ossec/etc/ossec.conf :

```
<localfile>
  <log_format>audit</log_format>
  <location>/var/log/audit/audit.log</location>
</localfile>
```

Nous avons par la suite redémarré l'agent Wazuh : systemctl restart wazuh-agent

- Visualisation des alertes dans Wazuh

Pour retrouver nos alertes aller dans : Modules > Security events > Level 12 or above alerts.

Mar 26, 2024 @ 14:53:49.023	001	serveur-prod-hadi	Audit: Highly Suspicious Command executed: /usr/bin/nc	12	100210

Table JSON Rule

@timestamp	2024-03-26T13:53:49.023Z
_id	Ye0Je44Ben9hyzta6-kB
agent.id	001
agent.ip	172.16.110.247
agent.name	serveur-prod-hadi
data.audit.arch	c000003e
data.audit.auid	1000
data.audit.command	nc
data.audit.cwd	/home/etudiant
data.audit.egid	0
data.audit.euid	0
data.audit.exe	/usr/bin/nc
data.audit.execve.a0	nc
data.audit.execve.a1	172.16.110.41
data.audit.execve.a2	8080

On peut voir que l'alerte est remontée, une alerte de niveau 12. Wazuh a détecté une exécution de la commande nc qu'on a lancé sur l'agent, on retrouve aussi les différents arguments de la commande dans les champs data.audit.execve.a0.

- Configuration envoi d'alerte de niveau 12 par mail

Wazuh comprend une fonctionnalité d'envoi par mail, on a décidé de mettre en place cette fonctionnalité. On a mis en place un serveur SMTP, et configuré le mail dans la configuration du wazuh-manager /var/ossec/etc/ossec.conf :

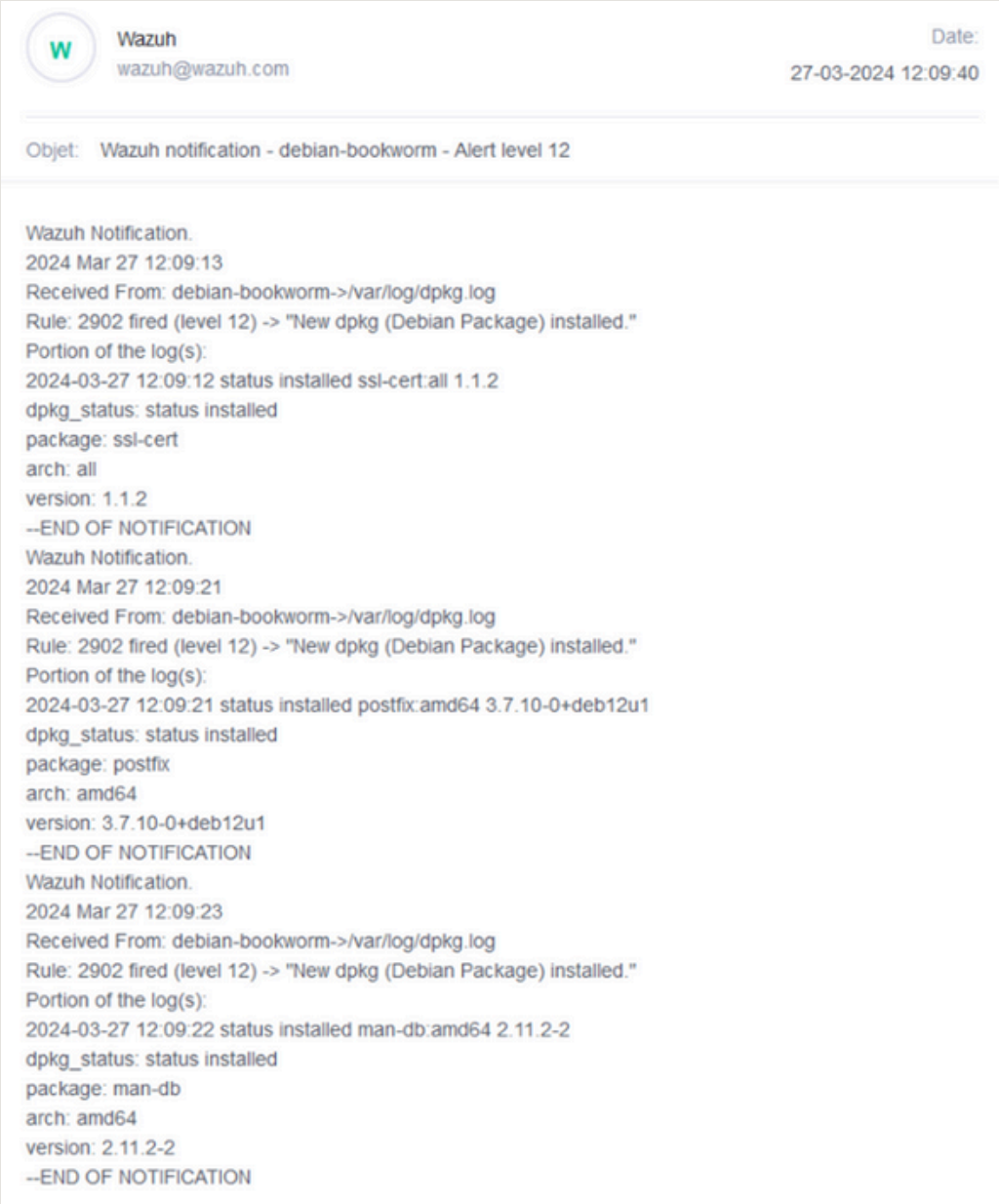
```
<ossec_config>
  <global>
        <jsonout_output>yes</jsonout_output>
        <alerts_log>yes</alerts_log>
        <logall>no</logall>
        <logall_json>no</logall_json>
        <email_notification>yes</email_notification>
        <smtp_server>localhost</smtp_server>
        <email_from>wazuh@wazuh.com</email_from>
        <email_to>xajise3006@sentrau.com</email_to>  //Ajout de notre adresse mail
(venant de tempmail)
        <email_maxperhour>12</email_maxperhour>
        <email_log_source>alerts.log</email_log_source>
        <agents_disconnection_time>10m</agents_disconnection_time>
        <agents_disconnection_alert_time>0</agents_disconnection_alert_time>
  </global>
```

Ensuite nous avons ajouté le niveau d'alerte souhaité, on a mis les alertes de niveau 12 dans le même fichier de configuration :

```
<alerts>
        <log_alert_level>3</log_alert_level>
        <email_alert_level>12</email_alert_level>
 </alerts>
```

- Visualisation des alertes par mail

Mail reçu après avoir installé un paquetage, on avait configurer les alertes au niveau 12 pour les installations de nouveaux paquets :



Wazuh
wazuh@wazuh.com

Date:
27-03-2024 12:09:40

Objet: Wazuh notification - debian-bookworm - Alert level 12

Wazuh Notification.
2024 Mar 27 12:09:13
Received From: debian-bookworm->/var/log/dpkg.log
Rule: 2902 fired (level 12) -> "New dpkg (Debian Package) installed."
Portion of the log(s):
2024-03-27 12:09:12 status installed ssl-cert:all 1.1.2
dpkg_status: status installed
package: ssl-cert
arch: all
version: 1.1.2
--END OF NOTIFICATION
Wazuh Notification.
2024 Mar 27 12:09:21
Received From: debian-bookworm->/var/log/dpkg.log
Rule: 2902 fired (level 12) -> "New dpkg (Debian Package) installed."
Portion of the log(s):
2024-03-27 12:09:21 status installed postfix:amd64 3.7.10-0+deb12u1
dpkg_status: status installed
package: postfix
arch: amd64
version: 3.7.10-0+deb12u1
--END OF NOTIFICATION
Wazuh Notification.
2024 Mar 27 12:09:23
Received From: debian-bookworm->/var/log/dpkg.log
Rule: 2902 fired (level 12) -> "New dpkg (Debian Package) installed."
Portion of the log(s):
2024-03-27 12:09:22 status installed man-db:amd64 2.11.2-2
dpkg_status: status installed
package: man-db
arch: amd64
version: 2.11.2-2
--END OF NOTIFICATION

- Génération des clés RSA

Afin de permettre des connexions sécurisées et sans mot de passe entre les machines, une paire de clés SSH a été générée à l'aide de la commande suivante :

```
etudiant@ssh-server:~$ ssh-keygen -t rsa -b 4096
```

Cette commande crée une paire de clés RSA de 4096 bits. La clé privée est stockée dans un fichier (id_rsa par défaut) et reste sur la machine locale. La clé publique correspondante (id_rsa.pub) sera partagée avec les autres machines pour permettre une authentification automatique.
Clé privée : Reste sur la machine de contrôle.
Clé publique : Partagée avec les autres machines.

```
+---[RSA 4096]----+
|      .oo...   .+  |
|      .=o    o + o |
|      .+o. . o o .|
|     ...+.. ...    |
|    o ...S.o....   |
|   E o  ..+.o...   |
|    . .    .*..o . |
|   o +.   .. =o o  |
|    +o+o..oo.....  |
+----[SHA256]-----+
```

https://matias.me/nsfw/

- Attribution de la clé publique sur les machines cibles

Pour autoriser la machine de contrôle à se connecter sans mot de passe aux autres machines, la clé publique a été copiée sur chacune d'elles à l'aide de la commande :

```
etudiant@ssh-server:~$ ssh-copy-id -p 2222 etudiant@172.16.9.206
```

Cette commande copie la clé publique vers le répertoire /home/etudiant/.ssh de l'utilisateur distant. Une fois la clé en place, une connexion SSH pourra être établie sans qu'un mot de passe ne soit requis.

```
etudiant@ssh-server:~$ ssh-copy-id -p 2222 etudiant@172.16.9.208
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/etudiant/.ssh/id_rsa.pub"
The authenticity of host '[172.16.9.208]:2222 ([172.16.9.208]:2222)' can't be established.
ECDSA key fingerprint is SHA256:BNqTcbNyvuiJG1AnkQKcjkaNnxJElKQ1WHkDG/IIbu0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
etudiant@172.16.9.208's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh -p '2222' 'etudiant@172.16.9.208'"
and check to make sure that only the key(s) you wanted were added.
```

- Exemple de connexion sans mot de passe

Une fois les clés correctement configurées, une connexion SSH peut être réalisée à l'aide de la commande suivante :

```
etudiant@ssh-server:~$ ssh -p 2222 etudiant@172.16.9.210
Linux dhcp-server 5.10.0-33-amd64 #1 SMP Debian 5.10.226-1 (2024-10-03) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Oct 24 13:50:50 2024 from 10.0.2.2
etudiant@dhcp-server:~$ exit
déconnexion
Connection to 172.16.9.210 closed.
etudiant@ssh-server:~$
```

- Vérification du statut du service SSH sur les machines

Cette commande SSH a pour objectif de vérifier l'état du service SSH sur une machine distante comme par exemple avec l'adresse IP 192.168.1.2 (Wazuh).

- Vérification du statut du service SSH sur les machines

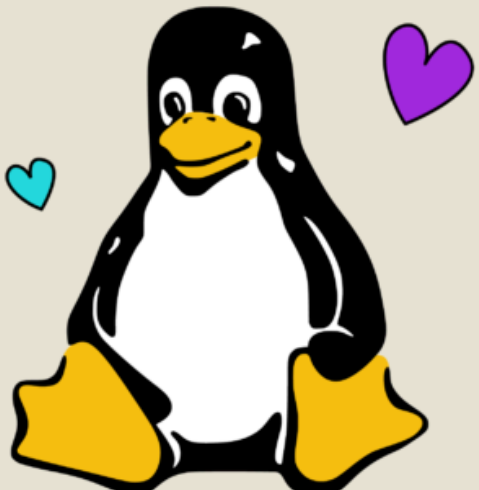
Cette commande SSH a pour objectif de vérifier l'état du service SSH sur une machine distante comme par exemple avec l'adresse IP 192.168.1.2 (Wazuh).

```
etudiant@ssh-server:~$ ssh -o ConnectTimeout=5 etudiant@192.168.1.2 "systemctl is-active ssh" 2>/dev/null
active
```

Dans un premier temps, nous avons réalisé ce script de base conçu pour automatiser la vérification de l'état du service SSH sur plusieurs machines distantes.
Script de base : Vérification de l'état des services

```
etudiant@ssh-server:~$ grep -v '#' verifieetat.sh
TARGET_MACHINES=("192.168.1.2" "192.168.1.3" "192.168.1.16" "192.168.1.35" "192.168.1.4")
REMOTE_USER="etudiant"
SERVICE="ssh"

for TARGET in "${TARGET_MACHINES[@]}"; do
    echo "Connecting to $TARGET..."
    ssh -o ConnectTimeout=5 ${REMOTE_USER}@${TARGET} "systemctl is-active $SERVICE" 2>/dev/null
done
```

```
etudiant@ssh-server:~$ etudiant@ssh-server:~$ ./verifieetat.sh
Connecting to 192.168.1.2...
active
Connecting to 192.168.1.3...
active
Connecting to 192.168.1.35...
The authenticity of host '192.168.1.35 (192.168.1.35)' can't be established.
ECDSA key fingerprint is SHA256:BNqTcbNyvuiJG1AnkQKcjkaNnxJElKQ1WHkDG/IIbu0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
active
Connecting to 192.168.1.4...
The authenticity of host '192.168.1.4 (192.168.1.4)' can't be established.
ECDSA key fingerprint is SHA256:BNqTcbNyvuiJG1AnkQKcjkaNnxJElKQ1WHkDG/IIbu0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
active
etudiant@ssh-server:~$
```



- Présentation du parc informatique

Les services principaux présents dans le parc informatique et leurs adresses sont les suivants :

  - Wazuh : Surveillance de la sécurité.
  - DHCP/DNS : Attribution dynamique d'adresses IP et résolution des noms de domaine.
  - Rsyslog : Collecte des journaux systèmes pour l'analyse.
  - NTP : Synchronisation de l'heure des systèmes.
  - SSH : Connexion sécurisée et administration des machines.
  - Docker : Stockage DB, reverse Proxy, PHP

Pour vérifier le statut de plusieurs services essentiels (comme wazuh-agent, ntp, et rsyslog), le script a été amélioré pour gérer plusieurs services et afficher un résultat en couleur pour une meilleure lisibilité.

Script amélioré : Vérification de l'état des services

```
etudiant@ssh-server:~$ grep -v '#' check_services.sh

TARGET_MACHINES=(
    "172.16.9.186:2222"
    "172.16.9.210:2222"
    "172.16.9.205:2222"
    "172.16.9.208:2222"
    "172.16.9.206:2222"
)
REMOTE_USER="etudiant"

SERVICES=("wazuh-agent" "ntp" "rsyslog")

print_ok() {
    echo -e "\e[32m[OK]\e[0m"
}

print_fail() {
    echo -e "\e[31m[NON OK]\e[0m"
}

for TARGET in "${TARGET_MACHINES[@]}"; do
    IFS=':' read -r IP PORT <<< "$TARGET"

    echo -e "\n\033[1mConnexion à la machine $IP sur le port $PORT...\033[0m"

    ssh -q -o ConnectTimeout=5 -p "$PORT" ${REMOTE_USER}@"$IP" exit
    if [ $? -ne 0 ]; then
        echo -e "\e[31mImpossible de se connecter à $IP sur le port $PORT.\e[0m"
        continue
    fi

    for SERVICE in "${SERVICES[@]}"; do
        echo -n "Vérification du service $SERVICE sur $IP : "

        STATUS=$(ssh -q -o ConnectTimeout=5 -p "$PORT" ${REMOTE_USER}@"$IP" "systemctl is-active $SERVICE 2
>/dev/null")

        if [ "$STATUS" == "active" ]; then
            print_ok
        else
            print_fail
        fi
    done
done

echo -e "\n\033[1;34mVérification des services terminée.\033[0m"
```

Ce script se connecte à chaque machine cible via SSH, vérifie l'état de chaque service spécifié, et affiche le résultat en vert (OK) ou en rouge (NON OK). Il utilise des codes de couleur pour améliorer l'affichage.

Ce script se connecte à chaque machine cible via SSH, vérifie l'état de chaque service spécifié, et affiche le résultat en vert (OK) ou en rouge (NON OK). Il utilise des codes de couleur pour améliorer l'affichage.

- Résultats et affichage
  - Le script affiche pour chaque machine :
  - Une tentative de connexion.
  - La vérification du statut des services.
  - Une couleur verte (OK) si le service est actif, et rouge (NON OK) s'il est inactif ou si la connexion échoue.

```
etudiant@ssh-server:~$ etudiant@ssh-server:~$ ./check_services.sh

Connexion à la machine 172.16.9.186 sur le port 2222...
Vérification du service wazuh-agent sur 172.16.9.186 : [NON OK]
Vérification du service ntp sur 172.16.9.186 : [OK]
Vérification du service rsyslog sur 172.16.9.186 : [OK]

Connexion à la machine 172.16.9.210 sur le port 2222...
Vérification du service wazuh-agent sur 172.16.9.210 : [OK]
Vérification du service ntp sur 172.16.9.210 : [OK]
Vérification du service rsyslog sur 172.16.9.210 : [OK]

Connexion à la machine 172.16.9.205 sur le port 2222...
Vérification du service wazuh-agent sur 172.16.9.205 : [OK]
Vérification du service ntp sur 172.16.9.205 : [OK]
Vérification du service rsyslog sur 172.16.9.205 : [OK]

Connexion à la machine 172.16.9.208 sur le port 2222...
Vérification du service wazuh-agent sur 172.16.9.208 : [OK]
Vérification du service ntp sur 172.16.9.208 : [OK]
Vérification du service rsyslog sur 172.16.9.208 : [OK]

Connexion à la machine 172.16.9.206 sur le port 2222...
Vérification du service wazuh-agent sur 172.16.9.206 : [NON OK]
Vérification du service ntp sur 172.16.9.206 : [NON OK]
Vérification du service rsyslog sur 172.16.9.206 : [OK]

Vérification des services terminée.
etudiant@ssh-server:~$
```

- Conclusion

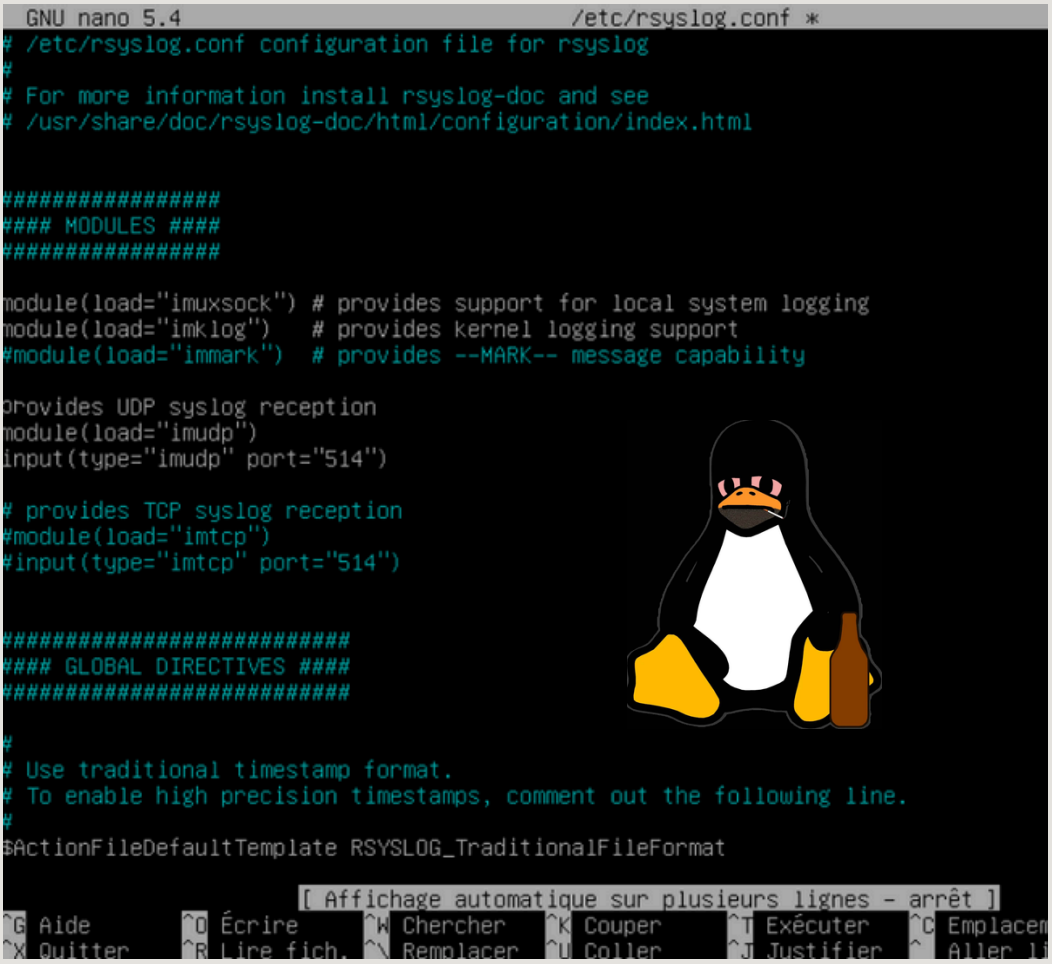
L'ensemble des procédures effectuées permet une administration sécurisée, automatique et efficace des machines dans le parc informatique. Grâce à l'utilisation de clés RSA et d'un script Bash amélioré, nous avons centralisé et automatisé la vérification des services critiques, ce qui simplifie la gestion et le diagnostic des systèmes.

Notre serveur Rsyslog sera un point unique de centralisation, auquel vos clients pourront envoyer leurs logs. Le protocole syslog a été conçu pour surveiller les périphériques réseaux et systèmes afin d'envoyer des messages de notification en cas de dysfonctionnement. Il peut vous envoyer aussi des alertes pour les événements pré notifiés, et surveiller les activités suspectes via les différents journaux de monitoring.

Tout d'abord, on installe rsyslog : sudo apt install rsyslog

Pour configurer notre serveur, il suffit d'éditer le fichier /etc/rsyslog.conf et de décommenter les lignes suivantes pour activer la collecte de logs sur le port UDP 514, qui est le port par défaut :

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



Ensuite, on redémarre le service rsyslog avec la commande suivante : service rsyslog restart.

Mais pour recevoir des logs, il faut qu'ils soient envoyés. Nous allons maintenant configurer notre client Linux, ici le fichier par défaut de Wazuh Server (172.16.9.186) pour envoyer ses logs à notre serveur de centralisation.

Pour ce faire, éditez le fichier par défaut /etc/rsyslog.d/50-default.conf :

```
  GNU nano 7.2                    /etc/rsyslog.d/50-default.conf
auth,authpriv.*         @172.16.9.205
```

Nous voulons enregistrer les logs d'authentification (auth, authpriv) et les transmettre à la machine 172.16.9.205, qui est notre serveur.
On voit ici les tentatives de connexion avec la remontée des logs : tail -f /var/log/auth.log :

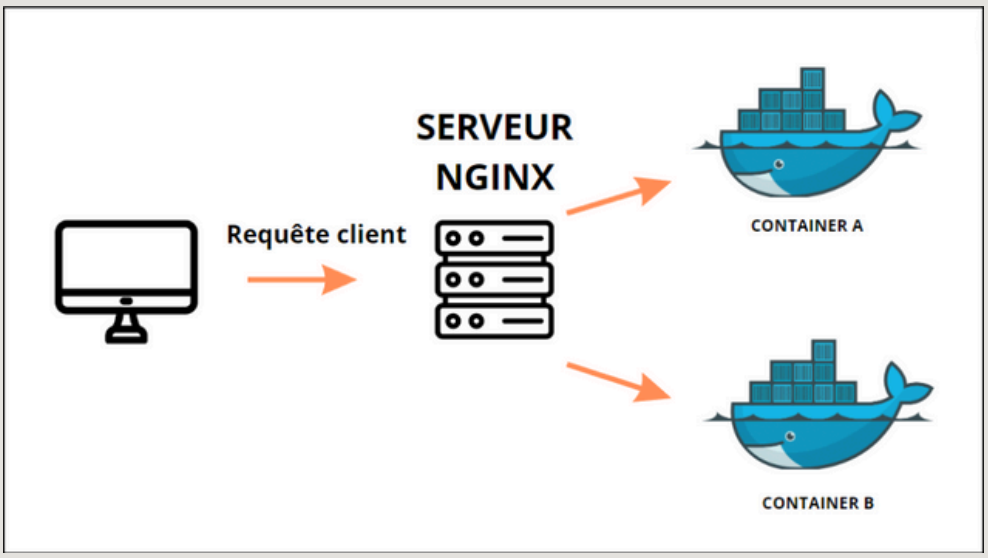
```
etudiant@elias:~$ sudo ssh etudiant@172.16.9.186 -p 2222
etudiant@172.16.9.186's password:
Linux wazuh-server 6.1.0-9-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.27-1 (2023-05-08) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Oct 24 15:34:56 2024 from 10.0.2.2
etudiant@wazuh-server:~$ sudo tail -f /var/log/auth.log
2024-10-24T15:43:43.323354+02:00 wazuh-server sshd[5668]: Disconnected from user etudiant 10.0.2.2 p
ort 53575
2024-10-24T15:43:43.326086+02:00 wazuh-server sshd[5662]: pam_unix(sshd:session): session closed for
 user etudiant
2024-10-24T15:43:43.334146+02:00 wazuh-server systemd-logind[433]: Session 73 logged out. Waiting fo
r processes to exit.
2024-10-24T15:43:43.342171+02:00 wazuh-server systemd-logind[433]: Removed session 73.
2024-10-24T15:44:40.610392+02:00 wazuh-server sshd[5671]: Accepted password for etudiant from 10.0.2
.2 port 56051 ssh2
2024-10-24T15:44:40.618031+02:00 wazuh-server sshd[5671]: pam_unix(sshd:session): session opened for
 user etudiant(uid=1000) by (uid=0)
2024-10-24T15:44:40.650329+02:00 wazuh-server systemd-logind[433]: New session 74 of user etudiant.
2024-10-24T15:44:40.727322+02:00 wazuh-server sshd[5671]: pam_env(sshd:session): deprecated reading
of user environment enabled
2024-10-24T15:44:46.228398+02:00 wazuh-server sudo: etudiant : TTY=pts/3 ; PWD=/home/etudiant ; USER
=root ; COMMAND=/usr/bin/tail -f /var/log/auth.log
2024-10-24T15:44:46.233848+02:00 wazuh-server sudo: pam_unix(sudo:session): session opened for user
root(uid=0) by etudiant(uid=1000)
```

En plus de l'application web en PHP et la base de données mysql nous avons ajouté un reverse proxy, pour ce faire nous avons déployé deux versions de notre application dans la
même VM mais sur deux ports différents.

Fichiers de l'application :

```
etudiant@appli-web:~$ sudo ls Docker2
docker-compose.yml  docker-compose.yml.bak  docker-compose.yml.bak2  Dockerfilephp  HTML  HTML2  init.sql  init.sql.bak  init.sql.save  nginx.conf  php
etudiant@appli-web:~$ sudo ls Docker2/HTML/
Accueil.php  activation.php  auth_code.php  changepassword.php  connexion_database.php  css  deconnexion.php  forgotpassword.php  images  index.php  info.php  js  signin.php  signup.php  src
etudiant@appli-web:~$
```

Schéma de l'infrastructure :



Nous avons modifié le fichier docker-compose.yml :

```
version: '3'

services:



  # Service MySQL

  mysql:
```

```
    image: mysql:latest

    container_name: mysql-container1

    environment:

    MYSQL_ROOT_PASSWORD: vitrygtr

    MYSQL_DATABASE: data

    MYSQL_USER: data_admin

    MYSQL_PASSWORD: vitrygtr



    networks:
    - loadbalancing

    volumes :

    - ./init.sql:/docker-entrypoint-initdb.d/init.sql



# Service MySQL second instance


# Service PHPMyAdmin

phpmyadmin:

    image: phpmyadmin/phpmyadmin:latest

    container_name: phpmyadmin-container

    environment:

    PMA_HOST: mysql

    PMA_PORT: 3306

    MYSQL_ROOT_PASSWORD: vitrygtr
```

```
    depends_on:

    - mysql



    networks:
    - loadbalancing

# Service PHP with Apache

php:

    build:

    context: .

    dockerfile: Dockerfilephp

    container_name: php-container1

    volumes:

    - ./HTML:/var/www/html

    - ./php/php.ini:/usr/local/etc/php/php.ini


    depends_on:

    - mysql

    networks:
    - loadbalancing

# Service PHP with Apache second instance

php2:

    build:

    context: .

    dockerfile: Dockerfilephp

    container_name: php-container2
```

```
    volumes:

    - ./HTML2:/var/www/html

    - ./php/php.ini:/usr/local/etc/php/php.ini


    depends_on:

    - mysql

    networks:
    - loadbalancing

# Service Apache

apache:

    image: httpd:latest

    container_name: apache-container1

    volumes:

    - ./HTML:/usr/local/apache2/htdocs/


    depends_on:

    - php

    networks:
    - loadbalancing

# Service Apache second instance

apache2:

    image: httpd:latest

    container_name: apache-container2

    volumes:

    - ./HTML:/usr/local/apache2/htdocs/
```

```
      depends_on:

      - php2
      networks:
      - loadbalancing


  # Nginx reverse proxy

  nginx:

      image: nginx:latest

      container_name: nginx-container

      volumes:

      - ./nginx.conf:/etc/nginx/nginx.conf

      ports:

      - "80:80"

      networks:
      - loadbalancing
      depends_on:
      - php
      - php2

networks:
  loadbalancing:
```

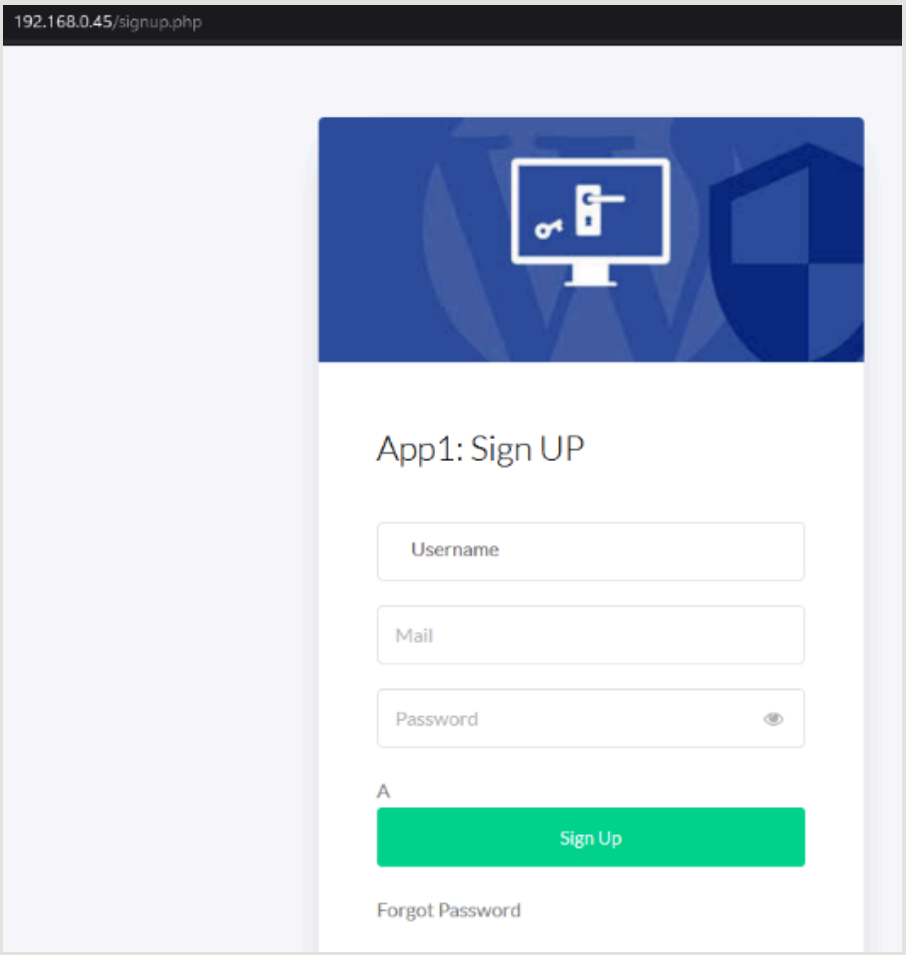
Comme vous pouvez le voir, nous avons dupliqué les services PHP, Apache afin d'effectuer un load balancing avec Nginx. On a ajouté ce dernier sous forme de container, il pointe sur le port 80 de notre machine hôte. Nous avons aussi configuré le fichier de configuration *nginx.conf* :

```
worker_processes  1;

events {
    worker_connections  1024;
}
```

```
http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;



    access_log  /var/log/nginx/access.log;
    error_log   /var/log/nginx/error.log;

    sendfile        on;
    keepalive_timeout  65;

    upstream pool {
        server php:80;
        server php2:80;
    }


    server {
        listen 80;
        location / {
            proxy_pass http://pool/;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
}
```

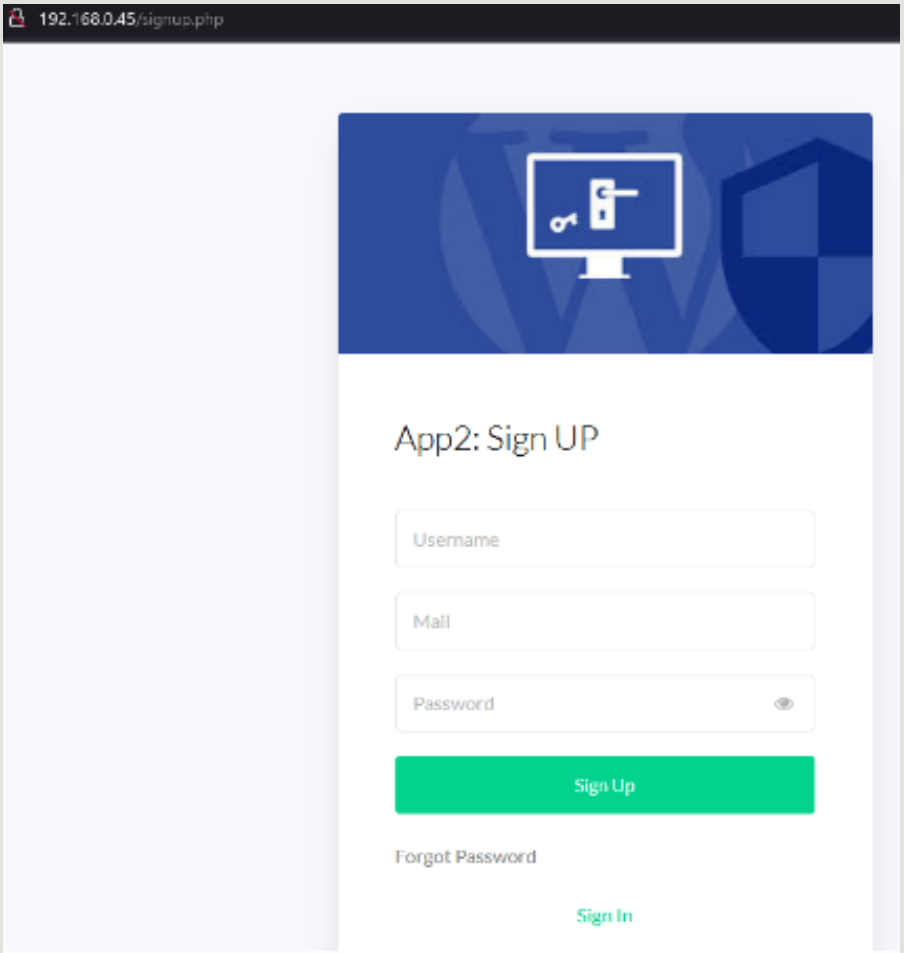
Maintenant nous avons testé que le load balancing a bien lieu, pour ce faire nous avons modifié le fichier signup.php de chaque container afin de vérifier si celui-ci fait bien le load balancing. Dans le container A nous avons mis l'inscription “site A” et sur le container B, “site B”. Bien sûr, ceci est fait dans un cadre de “test”, dans la réalité les deux sites ont le même code :

Site A :



Site B :

La première requête s'est faite sur le container A, une fois que nous rafraîchissons la page, nous obtenons la page *signup.php* du container B.



Nous avons aussi vérifié les logs dans les containers avec les commandes :

- sudo docker logs php-container1 & sudo docker logs php-container2.

Tout d'abord nous avons générer des requêtes vers la page signup.php avec l'aide de la commande :

- ab -n 1000 -c 100 http://192.168.1.130/signup.php

```
root@osboxes:/home/osboxes/Docker2# ab -n 1000 -c 100 http://192.168.0.45/signup.php
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.0.45 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        nginx/1.25.4
Server Hostname:        192.168.0.45
Server Port:            80

Document Path:          /signup.php
Document Length:        3032 bytes
```

Vérification dans le php-container1 :

```
root@osboxes:/home/osboxes/Docker2# sudo docker logs php-container1
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.19.0.6. Set the 'ServerName' directiv
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.19.0.6. Set the 'ServerName' directiv
[Sat Mar 30 17:31:41.653944 2024] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.57 (Debian) PHP/8.3.4 configured -- resuming normal o
[Sat Mar 30 17:31:41.654149 2024] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
172.19.0.9 - - [30/Mar/2024:17:31:58 +0000] "GET /signin.php?username=test&password=Vitrygtr99 HTTP/1.0" 200 1566 "http://192.168.0.45/s
0) Gecko/20100101 Firefox/124.0"
172.19.0.9 - - [30/Mar/2024:17:31:59 +0000] "GET /images/logo.jpg HTTP/1.0" 200 4754 "http://192.168.0.45/signin.php?username=test&passw
:124.0) Gecko/20100101 Firefox/124.0"
172.19.0.9 - - [30/Mar/2024:17:32:04 +0000] "GET / HTTP/1.0" 200 250 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0) Gecko/2010
172.19.0.9 - - [30/Mar/2024:17:32:10 +0000] "GET /signup.php HTTP/1.0" 200 1666 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0)
172.19.0.9 - - [30/Mar/2024:17:32:14 +0000] "GET /signup.php HTTP/1.0" 200 1463 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0)
172.19.0.9 - - [30/Mar/2024:17:32:20 +0000] "GET /signup.php HTTP/1.0" 200 1463 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0)
172.19.0.9 - - [30/Mar/2024:17:35:26 +0000] "GET /signup.php HTTP/1.0" 200 1463 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0)
172.19.0.9 - - [30/Mar/2024:17:35:30 +0000] "GET /signup.php HTTP/1.0" 200 1463 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0)
172.19.0.9 - - [30/Mar/2024:17:35:38 +0000] "GET /signup.php HTTP/1.0" 200 1463 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0)
172.19.0.9 - - [30/Mar/2024:17:35:42 +0000] "GET /signup.php HTTP/1.0" 200 1463 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:124.0)
172.19.0.9 - - [30/Mar/2024:17:37:13 +0000] "GET /favicon.ico HTTP/1.0" 404 454 "http://192.168.0.45/signup.php" "Mozilla/5.0 (Windows N
```

Vérification dans le php-container2 :

```
root@osboxes:/home/osboxes/Docker2# sudo docker logs php-container2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.19.0.5. Set th
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.19.0.5. Set th
[Sat Mar 30 17:31:42.937310 2024] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.57 (Debian) PHP/8.3.4 configur
[Sat Mar 30 17:31:42.941551 2024] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
172.19.0.9 - - [30/Mar/2024:17:31:58 +0000] "GET /css/style.css HTTP/1.0" 200 29023 "http://192.168.0.45/signin.p
124.0) Gecko/20100101 Firefox/124.0"
172.19.0.9 - - [30/Mar/2024:17:32:01 +0000] "GET / HTTP/1.0" 200 250 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x6
172.19.0.9 - - [30/Mar/2024:17:32:13 +0000] "GET /signup.php HTTP/1.0" 200 1657 "-" "Mozilla/5.0 (Windows NT 10.0
172.19.0.9 - - [30/Mar/2024:17:32:16 +0000] "GET /signup.php HTTP/1.0" 200 1456 "-" "Mozilla/5.0 (Windows NT 10.0
172.19.0.9 - - [30/Mar/2024:17:33:45 +0000] "GET / HTTP/1.0" 200 250 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x6
172.19.0.9 - - [30/Mar/2024:17:35:33 +0000] "GET /signup.php HTTP/1.0" 200 1456 "-" "Mozilla/5.0 (Windows NT 10.0
172.19.0.9 - - [30/Mar/2024:17:35:41 +0000] "GET /signup.php HTTP/1.0" 200 1456 "-" "Mozilla/5.0 (Windows NT 10.0
172.19.0.9 - - [30/Mar/2024:17:37:13 +0000] "GET /signup.php HTTP/1.0" 200 1456 "-" "Mozilla/5.0 (Windows NT 10.0
```

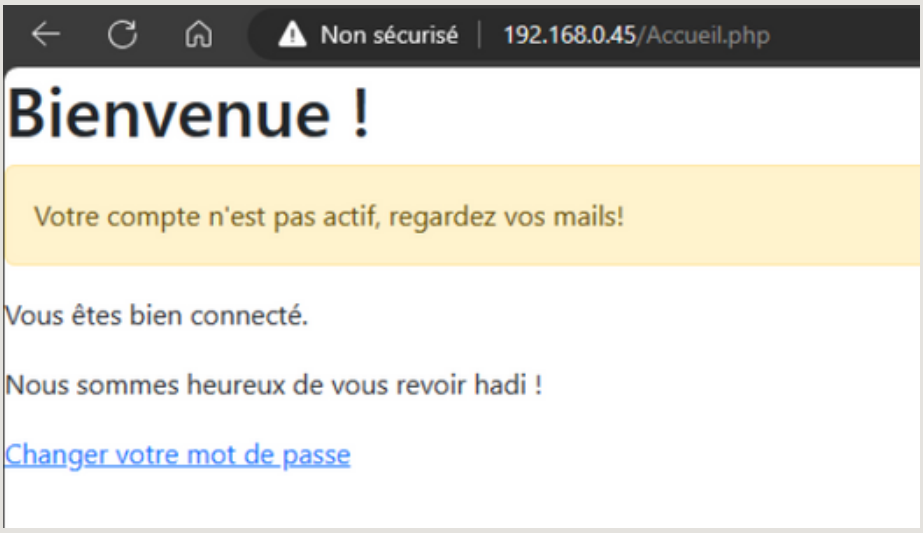
Pour conclure, Nginx nous offre la possibilité de répartir de manière efficace la charge du trafic entre plusieurs conteneurs en cas de fortes influences dans notre réseau. Pour assurer la stabilité et la robustesse de nos services web dans des environnements dynamiques et exigeants, Nginx devient un élément essentiel de notre architecture.

Problème des sessions PHP propre au Load Balancing

Après la mise en place du load balancing, nous nous sommes rendus compte que les sessions (cookies) utilisateurs disparaissaient entre temps (chaque reload de la page accueil.php) .

Sources :
https://www.freekb.net/Article?id=3682
https://autoize.com/php-session-storage-for-load-balanced-applications/

Après connexion :



Le site créé une session (cookies) avec notre username

Après rafraichissement de la page :

Ce problème est dû au fait que les cookies de sessions sont stockés localement sur un conteneur.

```
root@osboxes:/home/osboxes/Docker2/HTML# sudo docker exec php-container1 ls -l /tmp
total 4
-rw------- 1 www-data www-data 18 Mar 30 23:10 sess_425d3c7f68594bf92618be0ffb8e1343
-rw------- 1 www-data www-data  0 Mar 30 23:13 sess_dhlind309oh2p1c5nrv7l770td
root@osboxes:/home/osboxes/Docker2/HTML# sudo docker exec php-container2 ls -l /tmp
total 8
-rw------- 1 www-data www-data 18 Mar 30 23:10 sess_425d3c7f68594bf92618be0ffb8e1343
-rw------- 1 www-data www-data 18 Mar 30 23:13 sess_dhlind309oh2p1c5nrv7l770td
```

Comme les cookies sont stockés sur le conteneur qui les a créés, l'autre ne les reconnait pas puisqu'ils n'existent pas dans son répertoire de stockage.
Ainsi après rafraichissement de la page et voilà le reverse proxy Nginx nous redirige vers l'autre conteneur qui ne reconnaît pas notre session d'où la différence entre la page Accueil.php

La solution serait de partager le répertoire de stockage des sessions entre les deux conteneurs avec un volume commun sur docker-compose.

```
root@osboxes:/home/osboxes/Docker2# mkdir /usr/local/docker/php/sessions
root@osboxes:/home/osboxes/Docker2# mkdir /usr/local/docker/php/sessions
1777 /usr/local/dmkdir: cannot create directory '/usr/local/docker/php/sessions': File exists
root@osboxes:/home/osboxes/Docker2# chmod 1777 /usr/local/docker/php/sessions
```

Nous modifions le fichier docker-compose.yml pour ajouter le volume partagé entre les deux conteneurs PHP afin d'avoir les cookies des utilisateurs en commun:

```
# Service PHP with Apache
  php:
    build:
      context: .
      dockerfile: Dockerfilephp
    container_name: php-container1
    volumes:
      - ./HTML:/var/www/html
      - ./php/php.ini:/usr/local/etc/php/php.ini
      - /usr/local/docker/php/sessions:/usr/local/sessions
    depends_on:
      - mysql
    networks:
      - loadbalancing
# Service PHP with Apache second instance
  php2:
    build:
      context: .
      dockerfile: Dockerfilephp
    container_name: php-container2
    volumes:
      - ./HTML2:/var/www/html
      - ./php/php.ini:/usr/local/etc/php/php.ini
      - /usr/local/docker/php/sessions:/usr/local/sessions
    depends_on:
```

Nous modifions le fichier php.ini pour qu'il utilise le répertoire /usr/local/sessions pour stocker les sessions sur les deux conteneurs (volume partagé avec le répertoire /usr/local/docker/php/sessions du hôte) .

```
; where MODE is the octal representation of the mode. Note that this
; does not overwrite the process's umask.
; https://php.net/session.save-path
; configuration commune aux deux conteneurs
session.save_path = "/usr/local/sessions/"

; Whether to use strict session mode.
```

Les sessions sont maintenant partagées entre les deux conteneurs.
Problème Résolu !
Il nous reste plus qu'a rebuild les conteneurs.

Pour la sauvegarde nous allons utiliser Urbackup qui est un logiciel open-source qui fonctionne sous forme de serveur-client et qui permet de faire des sauvegardes régulières d'un répertoire/fichier .

Dans notre cas à nous, nous voulons sauvegarder le répertoire des fichiers de l'application docker pour pouvoir revenir en arrière en cas de problème.
Afin d'installer le logiciel dans notre machine serveur nous suivons les directives du développeur qui sont ici : https://www.urbackup.org/debianserverinstall.html

Nous exécutons ces commandes :

$ wget https://hndl.urbackup.org/Server/2.5.33/urbackup-server_2.5.33_amd64.deb
$ sudo dpkg -i urbackup-server_2.5.33_amd64.deb
$ sudo apt install -f

Nous vérifions que le service est lancé côté serveur :

```
etudiant@debian:~$ sudo service urbackupsrv start
[sudo] Mot de passe de etudiant :
etudiant@debian:~$ sudo service urbackupsrv status
● urbackupsrv.service - LSB: Server for doing backups
     Loaded: loaded (/etc/init.d/urbackupsrv; generated)
     Active: active (running) since Sun 2024-10-27 16:30:47 CET; 6min ago
       Docs: man:systemd-sysv-generator(8)
    Process: 385 ExecStart=/etc/init.d/urbackupsrv start (code=exited, status=0/SUCCESS
      Tasks: 19 (limit: 2323)
     Memory: 30.1M
        CPU: 243ms
     CGroup: /system.slice/urbackupsrv.service
             └─463 /usr/bin/urbackupsrv run --config /etc/default/urbackupsrv --daemon -

oct. 27 16:30:46 debian systemd[1]: Starting LSB: Server for doing backups...
oct. 27 16:30:47 debian systemd[1]: Started LSB: Server for doing backups.
```

Maintenant nous installons le client sur la machine docker:

$ TF=$(mktemp) && wget "https://hndl.urbackup.org/Client/2.5.25/UrBackup%20Client%20Linux%202.5.25.sh" -O $TF && sudo sh $TF; rm -f $TF

Si l'installation s'est bien passée alors nous avons des outils qui nous permettent de configurer la sauvegarde côté client.

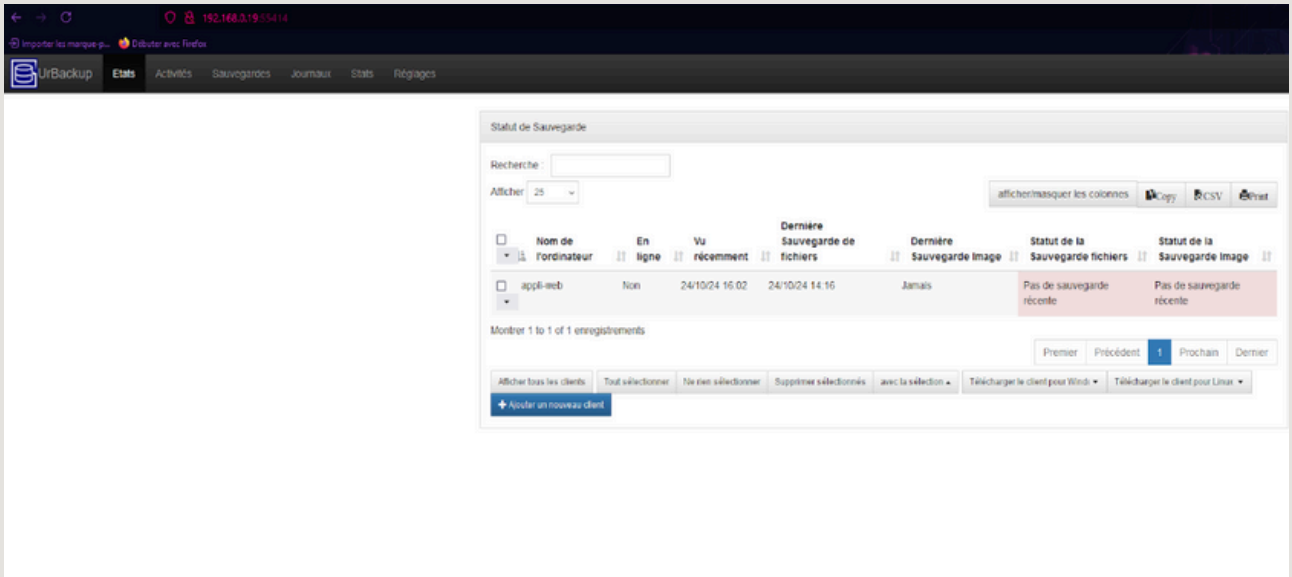
Le service client est-il démarré ?

```
etudiant@appli-web:~$ sudo systemctl status urbackup*
● urbackupclientbackend.service - UrBackup Client backend
     Loaded: loaded (/lib/systemd/system/urbackupclientbackend.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-10-27 17:15:22 CET; 4min 8s ago
   Main PID: 292 (urbackupclientb)
      Tasks: 13 (limit: 2322)
     Memory: 12.9M
        CPU: 1.799s
     CGroup: /system.slice/urbackupclientbackend.service
             └─292 /usr/local/sbin/urbackupclientbackend --config /etc/default/urbackupclient --no-consoletime

oct. 27 17:16:42 appli-web urbackupclientbackend[292]: ERROR: device-mapper: create ioctl on root-98d1f8b1f435-a31725acca864
oct. 27 17:16:42 appli-web urbackupclientbackend[292]: ERROR: Command failed.
oct. 27 17:16:42 appli-web urbackupclientbackend[292]: ERROR: Creating shadowcopy of "/dev/mapper/root-98d1f8b1f435" failed.
oct. 27 17:16:42 appli-web urbackupclientbackend[292]: ERROR: Creating shadow copy failed. See client log file for details.
oct. 27 17:16:43 appli-web urbackupclientbackend[292]: ERROR: Error in CQuery::Execute - database or disk is full  Stmt: [IN
oct. 27 17:16:43 appli-web urbackupclientbackend[292]: ERROR: Error in CQuery::Execute - database or disk is full  Stmt: [EN
oct. 27 17:16:43 appli-web urbackupclientbackend[292]: ERROR: SQL: database or disk is full Stmt: [END;]
oct. 27 17:16:43 appli-web urbackupclientbackend[292]: ERROR: SQL: database or disk is full Stmt: [INSERT INTO logs (ttime)
oct. 27 17:18:47 appli-web urbackupclientbackend[292]: ERROR: Error in CQuery::Execute - database or disk is full  Stmt: [EN
oct. 27 17:18:47 appli-web urbackupclientbackend[292]: ERROR: SQL: database or disk is full Stmt: [END;]
etudiant@appli-web:~$ |
```

Nous configurons le client de sauvegarde pour qu'il sauvegarde le répertoire /home/etudiant/Docker2.

```
etudiant@appli-web:~$ sudo /usr/local/bin/urbackupclientctl add-backupdir -d /home/etudiant/Dock
etudiant@appli-web:~$ sudo /usr/local/bin/urbackupclientctl list-backupdirs
PATH                   NAME    FLAGS
---------------------- ------- ------------------------------------------
/home/etudiant/Docker2 Docker2 follow_symlinks,symlinks_optional,share_hashes
etudiant@appli-web:~$ sudo /usr/local/bin/urbackupclientctl status
{
"capability_bits": 65536,
"finished_processes": [{
"process_id": 2,
"success": true
}
,{
"process_id": 3,
"success": false
}
],
"internet_connected": false,
"internet_status": "connected_local",
"last_backup_time": 1729772178,
"running_processes": [{
"action": "INCR",
"eta_ms": 0,
"percent_done": -1,
"process_id": 1,
"server_status_id": 1,
"speed_bpms": 0
}
],
"servers": [{
"internet_connection": false,
"name": "192.168.0.19"
}
],
"time_since_last_lan_connection": 26564
}

etudiant@appli-web:~$ sudo service urbackupclientctl status
```

Côté interface web:



Côté serveur de sauvegarde nous pouvons plusieurs sauvegardes de ce répertoire ce qui montre que nos actions ont été concluantes. Urbackup est un outil très puissant qui offre plein de possibilités de sauvegarde selon les besoins d'une infrastructure.

```
root@debian:/home/etudiant# ls /backup/
appli-web  clients  urbackup_tmp_files
root@debian:/home/etudiant# ls /backup/appli-web/
241024-1202_Image_C  241024-1222_Image_C  241024-1416  241024-1437_Image_C  241024-1452_Image_C  241024-15
root@debian:/home/etudiant# ls -lah /backup/appli-web/
total 40K
drwxr-x--- 10 urbackup urbackup 4,0K 27 oct.  17:16 .
drwxr-xr-x  5 urbackup urbackup 4,0K 27 oct.  17:00 ..
drwxr-x---  2 urbackup urbackup 4,0K 24 oct.  12:02 241024-1202_Image_C
drwxr-x---  2 urbackup urbackup 4,0K 24 oct.  12:22 241024-1222_Image_C
drwxr-x---  4 urbackup urbackup 4,0K 24 oct.  14:17 241024-1416
drwxr-x---  2 urbackup urbackup 4,0K 24 oct.  14:37 241024-1437_Image_C
drwxr-x---  2 urbackup urbackup 4,0K 24 oct.  14:52 241024-1452_Image_C
drwxr-x---  2 urbackup urbackup 4,0K 24 oct.  15:37 241024-1537_Image_C
drwxr-x---  3 urbackup urbackup 4,0K 27 oct.  17:16 241027-1716
drwxr-x---  2 urbackup urbackup 4,0K 27 oct.  17:16 241027-1716_Image_C
lrwxrwxrwx  1 urbackup urbackup   30 27 oct.  17:16 current -> /backup//appli-web/241027-1716
root@debian:/home/etudiant#
```

11 Conclusion

Ce projet de gestion des services et d'automatisation des tâches sur un parc informatique virtuel a permis de mettre en œuvre une infrastructure fiable et sécurisée tout en minimisant les interventions manuelles. Grâce à l'utilisation de machines virtuelles, nous avons pu créer un environnement flexible pour déployer, tester et configurer des services critiques tels que NTP, DNS, DHCP, et Wazuh, tout en assurant un haut niveau de sécurité à travers la configuration d'un firewall (iptables) et des connexions SSH sécurisées.

L'automatisation des tâches a été un aspect essentiel du projet, facilitant la gestion de l'infrastructure et réduisant les erreurs humaines. Les outils de surveillance comme Wazuh pourrait nous permettre de détecter rapidement des comportements anormaux et de réagir efficacement grâce à des alertes automatiques.

Ce projet a également démontré l'importance de la centralisation des journaux système (RSYSLOG), de la conteneurisation des applications (Docker), et de la mise en place de stratégies de sauvegarde pour protéger les données. Ces pratiques ont renforcé la stabilité et la sécurité de l'ensemble du parc informatique virtuel.

En conclusion, ce projet a permis de développer une expertise dans la gestion d'une infrastructure réseau complexe, d'automatiser les tâches de manière proactive, et de garantir une sécurité optimale à travers des pratiques modernes de déploiement et de surveillance. Ces compétences sont désormais transférables à d'autres environnements, qu'ils soient physiques ou virtuels, avec une approche axée sur la résilience, la sécurité et l'efficacité.