```
#################################################################
#                                                               #
#     Créateur : Arthur YANG - Responsable Documentation  #
#                         & Administratif       #

#     Date de création : 18/02/2025                         #
#                                                               #
#     Dernier modificateur : Arthur YANG            #
#     Date de modification : 18/02/2025             #
#                                                               #
#           Version actuelle : 1.0                  #
#                                                               #
#################################################################


---------------------- Configuration et sécurisation du serveur de
monitoring ----------------------

Sources : https://shape.host/resources/comment-installer-prometheus-sur-
debian-12


Actions effectuées dans le conteneur :

   => Connexion via root / mot de passe
   => apt install unzip
   => Installation de l'agent promtail

      Installation de Promtail
             => Téléchargement et installation :
            On se place dans le chemin : /usr/local/bin/
            On télécharge le fichier zip de Promtail : wget
https://github.com/grafana/loki/releases/download/v3.4.2/promtail-linux-
amd64.zip
            Nous pouvons ensuite l'unzip : unzip (nom du fichier)
            C'est un fichier executable que l'on a.

      => Configuration de Promtail :

      Nous devons ici, avoir un fichier nommé "promtail-config.yaml" pour
la configuration de Promtail.
      Nous pouvons télécharger un modèle pré-existant : wget
https://github.com/grafana/loki/blob/main/clients/cmd/promtail/promtail-
local-config.yaml ou le créer nous même.
      Cela télécharge un modèle que l'on peut renommer comme nous le
voulions.


=> Config .yaml personnalisé que nous effectuons :

-------------------------------------------------------------------------
-------------------

server:
```

```
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /var/lib/promtail/positions.yaml

clients:
  - url: http://192.168.7.133:3100/loki/api/v1/push  # Utilisation de
l'IP du serveur Loki

scrape_configs:
  - job_name: SRV-MONITORING-AARD-SYSLOG
    static_configs:
      - targets:
          - localhost
        labels:
          job: SYSLOG-AARD-MONITORING
          host: SRV-MONITORING-AARD
          stream: stdout
          __path__: /var/log/syslog

  - job_name: SRV-MONITORING-AARD-MAIL
    static_configs:
      - targets:
          - localhost
        labels:
          job: MAIL-AARD-MONITORING
          host: SRV-MONITORING-AARD
          stream: stdout
          __path__: /var/log/mail.log

  - job_name: SRV-MONITORING-AARD-AUTHLOG
    static_configs:
      - targets:
          - localhost
        labels:
          job: AUTHLOG-AARD-MONITORING
          host: SRV-MONITORING-AARD
          stream: stdout
          __path__: /var/log/auth.log

  - job_name: SRV-MONITORING-AARD-CRON
    static_configs:
      - targets:
          - localhost
        labels:
          job: CRON-AARD-MONITORING
          host: SRV-MONITORING-AARD
          stream: stdout
          __path__: /var/log/cron.log

--------------------------------------------------------------------------
-------------------
```

=> Créer un dossier /var/lib/promtail/ : mkdir -p /var/lib/promtail
=> chown -R monitoring:monitoring /var/lib/promtail
=> chmod -R 750 /var/lib/promtail

=> Nous mettons l'utilisateur créée plus tôt "monitoring" en tant que propriétaire et propriétaire groupe des fichiers de config de Loki.

=> chown -R monitoring:monitoring promtail-config.yaml promtail-linux-amd64
=> chmod 755 promtail-linux-amd64
=> chmod 640 promtail-config.yaml

=> Création du fichier : /etc/systemd/system/promtail.service

---

```
[Unit]
Description=Promtail Loki
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
User=monitoring
Group=monitoring
ExecStart=/usr/local/bin/promtail-linux-amd64 -
config.file=/usr/local/bin/promtail-config.yaml

SyslogIdentifier=monitoring
Restart=always

[Install]
WantedBy=multi-user.target
```

Commandes système et visualisation des logs en direct :

=> systemctl enable promtail.service
=> systemctl start promtail.service
=> systemctl status promtail.service
=> journalctl -f -u promtail.service

Normalement le service promtail est lancé.

Dans nos conteneurs LXC Debian, les logs semblent être dans le "journald" et non le syslog.

On installe "rsyslog" pour qu'ils puissent etre visible dans "syslog"

```
=> apt install rsyslog -y
=> systemctl enable rsyslog --now
=> tail -f /var/log/syslog
```

Pour que notre utilisateur "monitoring" puissent avoir les droits de lecture de logs, on va lui ajouter dans le groupe adm :

```
=> usermod -aG adm monitoring
=> systemctl restart promtail.service
```

En effet, Un problèmes d'accès aux fichiers /var/log/auth.log, /var/log/cron.log et /var/log/user.log par Grafana sera présent par le fait que l'on a paramétré le service promtail avec l'utilisateur "monitoring".

Faire de même avec le fichier "syslog" :

```
=> chown monitoring:root /var/log/syslog
=> chmod 640 /var/log/syslog
=> systemctl restart promtail.service
```

On peut rajouter :

```
=> chown monitoring:utmp /var/log/btmp
```

De manière générale, regarder les permissions des fichiers selon l'utilisateur qui utilise le service.

Changement du port par défaut pour Grafana et accès web : port 3000 vers (Voir revue de sécurité)

```
=> Faire le changement dans le fichier de configuration.
```

On va modifier quelques lignes pour la configuration des logs de grafana :

```
=> Dans le fichier /etc/grafana/grafana.ini

      => # Directory where grafana can store logs
         ;logs = /var/log/grafana

=> On décommente (enlève le point virgule) la ligne ci-dessus
```

Puis on relance le service, et on peut observer l'enregistrement des logs dans le syslog.

On va config l'envoi des mails depuis grafana, dans le fichier /etc/grafana/grafana.ini :

On utilise l'adresse : svg.wizardsndice@gmail.com et ne pas oublier de créer un mot de passe application

---

```
################################### SMTP / Emailing
##########################
[smtp]
enabled = true
host = smtp.gmail.com:587
user = svg.wizardsndice@gmail.com
password = XXXXX
;cert_file =
;key_file =
skip_verify = false
from_address = svg.wizardsndice@gmail.com
from_name = Grafana Monitoring
;ehlo_identity = dashboard.example.com
startTLS_policy = MandatoryStartTLS
# Enable trace propagation in e-mail headers, using the 'traceparent',
'tracestate' and (optionally) 'baggage' fields (defaults to false)
;enable_tracing = false
```

La config est faite, on redemarre le service grafana.
Je créée les libellés sur la boite gmail pour les mails de grafana.
Les mails arrivent sur la boite mail.

Maintenant on peut config les alertes sur GRAFANA.
Voir sur grafana les alertes et leur config.

On va configurer LDAP pour Grafana, afin de pouvoir s'authentifier avec les id ldap.

=> On créer le user pour link grafana et l'ad (grafanalink).
=> On importe les certifs ldaps sur le srv monitoring (MDP : Meme que le compte "grafanalink" que j'ai mis pour ouvrir les certifs) => Dans /usr/local/share/ca-certificates/

Dans le fichier grafana.ini, on a la config suivante :
On doit activer LDAP ici pour que sur le ui web LDAP apparaisse.

```
# The public facing domain name used to access grafana from a browser
domain = wnd.local

[auth.ldap]
enabled = true
config_file = /etc/grafana/ldap.toml
```

```
allow_sign_up = true
# prevent synchronizing ldap users organization roles
skip_org_role_sync = false
```

On peut restart le service, mais pour réellement config LDAPS sur le grafana, tout se passe dans le fichier : /etc/grafana/ldap.toml

---

```
# To troubleshoot and get more log info enable ldap debug logging in grafana.ini
# [log]
# filters = ldap:debug

[[servers]]
# Ldap server host (specify multiple hosts space separated)
host = "SRV-DC-AARD.wnd.local"
# Default port is 389 or 636 if use_ssl = true
port = 636
# Set to true if LDAP server should use an encrypted TLS connection (either with STARTTLS or LDAPS)
use_ssl = true
# If set to true, use LDAP with STARTTLS instead of LDAPS
start_tls = false
# The value of an accepted TLS cipher. By default, this value is empty. Example value: ["TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384"])
# For a complete list of supported ciphers and TLS versions, refer to: https://go.dev/src/crypto/tls/cipher_suites.go
# Starting with Grafana v11.0 only ciphers with ECDHE support are accepted for TLS 1.2 connections.
tls_ciphers = []
# This is the minimum TLS version allowed. By default, this value is empty. Accepted values are: TLS1.1 (only for Grafana v10.4 or older), TLS1.2, TLS1.3.
min_tls_version = ""
# set to true if you want to skip ssl cert validation
ssl_skip_verify = false
# set to the path to your root CA certificate or leave unset to use system defaults
root_ca_cert = "/usr/local/share/ca-certificates/cert_intermediate.crt"
# Authentication against LDAP servers requiring client certificates
# client_cert = "/path/to/client.crt"
# client_key = "/path/to/client.key"

# Search user bind dn
bind_dn = "cn=Grafana Link,ou=Utilisateurs,ou=WND,dc=wnd,dc=local"
# Search user bind password
# If the password contains # or ; you have to wrap it with triple quotes. Ex """#password;"""
```

```
# We recommend using variable expansion for the bind_password, for more
info https://grafana.com/docs/grafana/latest/setup-grafana/configure-
grafana/#var>
bind_password = '$__env{LDAP_BIND_PASSWORD}'

# Timeout in seconds (applies to each host specified in the 'host' entry
(space separated))
timeout = 10

# User search filter, for example "(cn=%s)" or "(sAMAccountName=%s)" or
"(uid=%s)"
search_filter = "(sAMAccountName=%s)"

# An array of base dns to search through
search_base_dns = ["dc=wnd,dc=local"]

## For Posix or LDAP setups that does not support member_of attribute you
can define the below settings
## Please check grafana LDAP docs for examples
#group_search_filter = "(&(objectClass=group)(member=%s))"
#group_search_base_dns = ["dc=wnd,dc=local"]
#group_search_filter_user_attribute = "distinguishedName"

# Specify names of the ldap attributes your ldap uses
[servers.attributes]
name = "givenName"
surname = "sn"
username = "sAMAccountName"
member_of = "memberOf"
email = "UserPrincipalName"

# Map ldap groups to grafana org roles
[[servers.group_mappings]]
group_dn = "cn=GL_Admins,ou=Services,ou=Groupes,ou=WND,dc=wnd,dc=local"
org_role = "Admin"
# To make user an instance admin  (Grafana Admin) uncomment line below
grafana_admin = true
# The Grafana organization database id, optional, if left out the default
org (id 1) will be used
# org_id = 1

[[servers.group_mappings]]
group_dn = "cn=GL_Admins,ou=Services,ou=Groupes,ou=WND,dc=wnd,dc=local"
org_role = "Editor"

#[[servers.group_mappings]]
# If you want to match all (or no ldap groups) then you can use wildcard
#group_dn = "*"
#org_role = "Viewer"

---------------------------------------------------------------------------
------------------
```

Sur le web ui, on essaye de config LDAPS :

=> On voit que c'est connecté sur le port 636 depuis le srv moni au srv dc
=> Le test ne marche pas, mais j'arrive a me connecter avec mon compte "arthury",

Maintenant je config dans le fichier /etc/grafana/ldap.toml
=> Le fait d'attribuer les bon groupes aux personnes, cacher le mdp dedans etc.. , en gros de la config
=> Les groupes sont fait, mais qu'avec un GL et non DL

Il faut que l'on cache le mot de passe qui lie le serveur DC avec le serveur monitoring, car dans la conf : /etc/grafana/ldap.toml, on voit qu'il est en clair (Le compte "grafana link" en gros).

On va donc créer une unité systemd avec variable cachée dans le fichier "override" du service GRAFANA.

=> mkdir -p /etc/systemd/system/grafana-server.service.d
=> nano /etc/systemd/system/grafana-server.service.d/override.conf

Puis on colle ceci dans le fichier :

=> [Service]
Environment="LDAP_BIND_PASSWORD=MDP"

=> On sauvegarde et quitte (Ctrl+O, Entrée, Ctrl+X)
=> On change les droits pour plus de sécurité : chmod root:root 600 /etc/systemd/system/grafana-server.service.d/override.conf

On recharge systemd + restart Grafana

=> systemctl daemon-reexec
=> systemctl restart grafana-server

On peut vérifier que tout est bon :

=> systemctl show grafana-server | grep LDAP_BIND_PASSWORD

On doit voir : Environment=LDAP_BIND_PASSWORD=MDP

Et dans ldap.toml, on doit bien avoir :

=> bind_password = '$__env{LDAP_BIND_PASSWORD}'

On peut restart encore une fois les services, mais maintenant on plus le MDP en clair dans le fichier CONF de LDAP venant de GRAFANA.
Le LDAPS doit fonctionner en se connectant avec les comptes ADMINS.

Ayant changé de serveur LDAP (AD => UCS); voici la nouvelle configuration du fichier ldap.toml :

```
# To troubleshoot and get more log info enable ldap debug logging in grafana.ini
# [log]
# filters = ldap:debug

[[servers]]
# Ldap server host (specify multiple hosts space separated)
host = "SRV-UCS-AARD.wnd.local"
# Default port is 389 or 636 if use_ssl = true
port = 636
# Set to true if LDAP server should use an encrypted TLS connection (either with STARTTLS or LDAPS)
use_ssl = true
# If set to true, use LDAP with STARTTLS instead of LDAPS
start_tls = false
# The value of an accepted TLS cipher. By default, this value is empty. Example value: ["TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384"])
# For a complete list of supported ciphers and TLS versions, refer to: https://go.dev/src/crypto/tls/cipher_suites.go
# Starting with Grafana v11.0 only ciphers with ECDHE support are accepted for TLS 1.2 connections.
tls_ciphers = []
# This is the minimum TLS version allowed. By default, this value is empty. Accepted values are: TLS1.1 (only for Grafana v10.4 or older), TLS1.2, TLS1.3.
min_tls_version = ""
# set to true if you want to skip ssl cert validation
ssl_skip_verify = false
# set to the path to your root CA certificate or leave unset to use system defaults
root_ca_cert = "/usr/local/share/ca-certificates/ucs-root-ca.crt"
# Authentication against LDAP servers requiring client certificates
# client_cert = "/path/to/client.crt"
# client_key = "/path/to/client.key"

# Search user bind dn
bind_dn = "cn=grafanalink,ou=Utilisateurs,ou=WND,dc=wnd,dc=local"
# Search user bind password
# If the password contains # or ; you have to wrap it with triple quotes. Ex """#password;"""
# We recommend using variable expansion for the bind_password, for more info https://grafana.com/docs/grafana/latest/setup-grafana/configure-grafana/#var>
bind_password = '$__env{LDAP_BIND_PASSWORD}'

# Timeout in seconds (applies to each host specified in the 'host' entry (space separated))
```

```
timeout = 10

# User search filter, for example "(cn=%s)" or "(sAMAccountName=%s)" or
"(uid=%s)"
search_filter = "(cn=%s)"

# An array of base dns to search through
search_base_dns = ["ou=Utilisateurs,ou=WND,dc=wnd,dc=local"]

## For Posix or LDAP setups that does not support member_of attribute you
can define the below settings
## Please check grafana LDAP docs for examples
#group_search_filter = "(&(objectClass=group)(member=%s))"
#group_search_base_dns = ["dc=wnd,dc=local"]
#group_search_filter_user_attribute = "distinguishedName"

# Specify names of the ldap attributes your ldap uses
[servers.attributes]
name = "givenName"
surname = "sn"
username = "cn"
member_of = "memberOf"
email = "mail"

# Map ldap groups to grafana org roles
[[servers.group_mappings]]
group_dn = "cn=GL_Admins,ou=Services,ou=Groupes,ou=WND,dc=wnd,dc=local"
org_role = "Admin"
# To make user an instance admin  (Grafana Admin) uncomment line below
grafana_admin = true
# The Grafana organization database id, optional, if left out the default
org (id 1) will be used
# org_id = 1

[[servers.group_mappings]]
group_dn = "cn=GL_Admins,ou=Services,ou=Groupes,ou=WND,dc=wnd,dc=local"
org_role = "Editor"

#[[servers.group_mappings]]
# If you want to match all (or no ldap groups) then you can use wildcard
#group_dn = "*"
#org_role = "Viewer"

---------------------------------------------------------------------------
------------------
```

Bien sur, il fallait importer le nouveau certificat root du nouveau serveur LDAP sur le serveur actuelle. Et update les certificats.

On va avant de faire les dashboards, installer Prometheus. Cela nous permettra de récupérer des métriques sur le Stormshield par exemple, et voir quelques autres serveurs.

=> Pour rester simple, on continuera d'utiliser l'utilisateur général "monitoring" créer exprès pour ce serveur. Pas besoin de créer un user pour Prometheus exclusivement.

=> Créer un dossier /var/lib/prometheus : mkdir -p /var/lib/prometheus
=> chown -R monitoring:monitoring /var/lib/prometheus
=> chmod -R 750 /var/lib/prometheus

=> mkdir /etc/prometheus
=> chown -R monitoring /etc/prometheus
=> chmod -R 750 /etc/prometheus

=> Installation de Prometheus

=> Téléchargement et installation :
On se place dans le chemin : /usr/local/bin/
On télécharge le fichier zip de Prometheus : wget https://github.com/prometheus/prometheus/releases/download/v3.2.1/prometheus-3.2.1.linux-amd64.tar.gz
Nous pouvons ensuite l'unzip : tar -xvf (nom du fichier)

=> Configuration de Prometheus :

Nous avons ici le répertoire Prometheus, on nous allons récupérer les applications "prometheus" et "promtool" dans /usr/local/bin

=> mv prometheus-3.2.1.linux-amd64/prometheus /usr/local/bin/
=> mv prometheus-3.2.1.linux-amd64/promtool /usr/local/bin/

Puis le fichier "yaml" dans /etc/prometheus

=> mv prometheus-3.2.1.linux-amd64/prometheus.yml /etc/prometheus

On peut supprimer le repertoire "prometheus-3.2.1.linux-amd64"

=> rm -r /usr/local/bin/prometheus-3.2.1.linux-amd64/

On donne les bon droits pour tout les fichiers et répertoires :

=> chown -R monitoring:monitoring /usr/local/bin/prometheus /usr/local/bin/promtool
=> chmod 755 /usr/local/bin/prometheus /usr/local/bin/promtool
=> chmod 640 /etc/prometheus/prometheus.yml
=> chown -R monitoring:monitoring /etc/prometheus/prometheus.yml

=> Création du fichier : nano /etc/systemd/system/prometheus.service

```
--------------------------------------------------------------------------------
------------------

[Unit]
Description=Prometheus Monitoring
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
User=monitoring
Group=monitoring
ExecStart=/usr/local/bin/prometheus \
  --config.file=/etc/prometheus/prometheus.yml \
  --storage.tsdb.path=/var/lib/prometheus \
  --web.listen-address=0.0.0.0:XXXX
  --storage.tsdb.retention.time=7d \
  --storage.tsdb.min-block-duration=2h \
  --storage.tsdb.max-block-duration=2h

SyslogIdentifier=monitoring
Restart=always

[Install]
WantedBy=multi-user.target

--------------------------------------------------------------------------------
------------------
```

On a changé ici le port d'écoute de Promotheus ici, dans le fichier du service. Ca ne se fait pas dans le .yml .
Et d'autres paramètres sont présent pour de l'optimisation.

Commandes système et visualisation des logs en direct :

```
            => systemctl daemon-reexec
            => systemctl daemon-reload
            => systemctl enable prometheus.service
            => systemctl start prometheus.service
            => systemctl status prometheus.service
            => journalctl -f -u prometheus.service
```

Normalement le service prometheus est lancé.

Prometheus est maintenant installé, il faut maintenant le config. Sur grafana ça fonctionne bien.

=> On autorisera si besoin, l'écoute du port PROMETHEUS vers d'autres serveurs. Même si il n'y a pas besoin.
=> Ce sera surtout les ports des "exporters" où il faudra autoriser sur le FW, depuis autre serveur surement.
=> Utilisation d'un port différent de celui par défaut pour PROMETHEUS. (Voir FW ou DOC de Sécu)

=> Utilisation d'un port différent de celui par défaut pour SNMP EXPORTER (Voir FW ou DOC de Sécu)

J'ai ajouté des lignes dans le fichier /etc/prometheus/prometheus.yml :

```
=> global:
     scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
       evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
       # scrape_timeout is set to the global default (10s).
       external_labels:
       environment: "production"
         instance: "SRV-MONITORING-AARD"
```

Ces lignes pourront être modifiées plus tard.

On va essayer de surveiller l'état du FW avec SNMP et Prometheus.

=> On active SNMP sur FW (SNMPV2C) voir config sur le FW
=> Autorisation du passage des infos snmp en autorisant le port SNMP et SNMPTRAP depuis le FW vers SRV MONITORING
=> Sur le SRV MONITORING : apt install snmp
=> On peut tester de récup des infos depuis le serveur via SNMP sur le FW : snmpwalk -v2c -c wnd -r 1 192.168.7.142

=> Dans /usr/local/bin/ , on télécharge SNMP Exporter qui va nous servir à récupérer les métriques de Stormshield ici.

```
      => wget https://github.com/prometheus/snmp_exporter/releases/download/v0.28.0/snmp_exporter-0.28.0.linux-amd64.tar.gz
      => tar -xvzf snmp_exporter-0.28.0.linux-amd64.tar.gz
      => rm snmp_exporter-0.28.0.linux-amd64.tar.gz
      => mv /usr/local/bin/snmp_exporter-0.28.0.linux-amd64/snmp_exporter /usr/local/bin/
      => mv /usr/local/bin/snmp_exporter-0.28.0.linux-amd64/snmp.yml /etc/prometheus
      => chown -R monitoring:monitoring /usr/local/bin/snmp_exporter
      => chmod 755 /usr/local/bin/snmp_exporter
      => chmod 640 /etc/prometheus/snmp.yml
      => chown -R monitoring:monitoring /etc/prometheus/snmp.yml
      => rm -r /usr/local/bin/snmp_exporter-0.28.0.linux-amd64/
```

=> Création du fichier : nano /etc/systemd/system/snmp_exporter.service

---

```
[Unit]
Description=Prometheus SNMP Exporter
```

```
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
User=monitoring
Group=monitoring
ExecStart=/usr/local/bin/snmp_exporter \
  --config.file=/etc/prometheus/snmp.yml \
  --web.listen-address=0.0.0.0:XXXX

SyslogIdentifier=monitoring
Restart=always

[Install]
WantedBy=multi-user.target
```

On a changé ici le port d'écoute de SNMP EXPORTER de Prometheus ici, dans le fichier du service. Ca ne se fait pas dans le .yml .

Commandes système et visualisation des logs en direct :

```
=> systemctl daemon-reexec
=> systemctl daemon-reload
=> systemctl enable snmp_exporter.service
=> systemctl start snmp_exporter.service
=> systemctl status snmp_exporter.service
=> journalctl -f -u snmp_exporter.service
```

On a ajouter le bloc "job_name" dans le fichier /etc/prometheus/prometheus.yml :

```
  - job_name: 'stormshield'
#   scrape_interval: 3000s
    static_configs:
      - targets:
          - 192.168.7.142   # IP du firewall SNS
    metrics_path: /snmp
    params:
      auth: [wnd_v2c]
      module: [stormshield]

    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target

      - source_labels: [__param_target]
        target_label: instance
```

```
      - target_label: __address__
        replacement: 192.168.7.134:XXXX
```

Création au cas ou, de l'objet PORT pour SNMP EXPORTER.

On personnalise le fichier /etc/prometheus/snmp.yml

```
auths:
  wnd_v2c:
    community: wnd
    version: 2

modules:
  stormshield:
    walk:
      - 1.3.6.1.2.1.1          # system
      - 1.3.6.1.2.1.2          # interfaces
      - 1.3.6.1.2.1.25         # host resources
      - 1.3.6.1.2.1.4          # IP
      - 1.3.6.1.2.1.5          # ICMP
      - 1.3.6.1.2.1.6          # TCP
      - 1.3.6.1.2.1.7          # UDP
      - 1.3.6.1.2.1.31         # ifXTable
      - 1.3.6.1.4.1.11256.1.1.1.1  # Stormshield system
      - 1.3.6.1.4.1.11256.1.1.7  # Sessions
      - 1.3.6.1.4.1.2021 #Infos
      - 1.3.6.1.2.1.1.3.0        # sysUpTimeInstance
      - 1.3.6.1.2.1.1.4.0        # sysContact
      - 1.3.6.1.2.1.1.5.0        # sysName
      - 1.3.6.1.4.1.2021.4       # Memory stats
      - 1.3.6.1.4.1.2021.10      # CPU Load
      - 1.3.6.1.4.1.2021.10.1.6.1
      - 1.3.6.1.4.1.2021.10.1.6.2
      - 1.3.6.1.4.1.2021.10.1.6.3
      - 1.3.6.1.2.1.25.1.1.0
      - 1.3.6.1.2.1.2.2.1.2
      - 1.3.6.1.2.1.25.3.3.1.2.196608

    metrics:
      - name: sysUpTimeInstance
        oid: 1.3.6.1.2.1.1.3.0
        type: gauge
        help: Uptime du systeme

      - name: sysContact
        oid: 1.3.6.1.2.1.1.4.0
        type: DisplayString
        help: Contact information
```

```
      - name: sysName
        oid: 1.3.6.1.2.1.1.5.0
        type: DisplayString
        help: Hostname du firewall

      - name: memTotalReal
        oid: 1.3.6.1.4.1.2021.4.5.0
        type: gauge
        help: Total RAM physique en KB

      - name: memAvailReal
        oid: 1.3.6.1.4.1.2021.4.6.0
        type: gauge
        help: RAM physique disponible en KB

      - name: memBuffer
        oid: 1.3.6.1.4.1.2021.4.14.0
        type: gauge
        help: RAM buffered en KB

      - name: cpuLoad1
        oid: 1.3.6.1.4.1.2021.10.1.3.1
        type: gauge
        help: Charge CPU sur 1 min

      - name: cpuLoad5
        oid: 1.3.6.1.4.1.2021.10.1.3.2
        type: gauge
        help: Charge CPU sur 5 min

      - name: cpuLoad15
        oid: 1.3.6.1.4.1.2021.10.1.3.3
        type: gauge
        help: Charge CPU sur 15 min

      - name: ListeIntStorm
        oid: 1.3.6.1.2.1.2.2.1.2
        type: DisplayString
        help: Liste des interfaces sur le Firewall

      - name: Utilisation_CPU_moyen
        oid: 1.3.6.1.2.1.25.3.3.1.2.196608
        type: gauge
        help: Utilisation moyenne du CPU (en %) sur 1m
-----------------------------------------------------------------
```

Maintenant le service se lance, donc SNMP EXPORTER fonctionne, mais il faut faire de la configuration maintenant.
(c'est pas facile et sur d'autres appareils)

Dont notamment pour récupérer des métriques. C'est de la config dans le snmp.yml.

Pour la configuration des dashboard dont ici pour le FW, pour un panneau, j'ai ajouter un module dans le menu "Transformations" pour dire que dès la requete en question est effectué, eh bas on remplace le résultat ici par le nom du FW.

=> Ce qui fait que c'est à 75% automatique mais que 25% ne l'est pas car si le nom du FW venait à changer (meme si ça reste quand meme rare), il faudrait aussi changer sur GRAFANA le nom que l'on veut afficher. Mais bon c'est le moyen le plus simple de le faire.

=> Config de beaucoup de dashboard et alerts, voir vidéo pour savoir comment faire rapidement.

=> Pour les alertes sur GRAFANA, je n'ai pas de règle pour chaque serveur de manière individuelle, car sinon ce serait vraiment trop long (jusqu'a 100 règles). Nous sommes en maquettes donc j'ai fait 2 règles individuelles + une générale pour chaque thème.

Install de ALERT MANAGER :

> Dans /usr/local/bin/ , on télécharge Alert Manager qui va nous servir à alerte si un problème survenait.

=> wget https://github.com/prometheus/alertmanager/releases/download/v0.28.1/alertmanager-0.28.1.linux-amd64.tar.gz
=> tar -xvzf alertmanager-0.28.1.linux-amd64.tar.gz
=> rm alertmanager-0.28.1.linux-amd64.tar.gz
=> mv /usr/local/bin/alertmanager-0.28.1.linux-amd64/alertmanager /usr/local/bin/
=> mv /usr/local/bin/alertmanager-0.28.1.linux-amd64/amtool /usr/local/bin/
=> mv /usr/local/bin/alertmanager-0.28.1.linux-amd64/alertmanager.yml /etc/prometheus
=> chown -R monitoring:monitoring /usr/local/bin/alertmanager /usr/local/bin/amtool
=> chmod 755 /usr/local/bin/alertmanager /usr/local/bin/amtool
=> chmod 640 /etc/prometheus/alertmanager.yml
=> chown -R monitoring:monitoring /etc/prometheus/alertmanager.yml
=> rm -r /usr/local/bin/alertmanager-0.28.1.linux-amd64/

=> Créer un dossier /var/lib/alertmanager : mkdir -p /var/lib/alertmanager
=> chown -R monitoring:monitoring /var/lib/alertmanager
=> chmod -R 750 /var/lib/alertmanager

On a modifier le bloc "alerting" dans le fichier /etc/prometheus/prometheus.yml :

```
# Alertmanager configuration
```

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
           - alertmanager:XXXX
```

=> Création du fichier : nano /etc/systemd/system/alertmanager.service

---

```
[Unit]
Description=Prometheus AlertManager
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
User=monitoring
Group=monitoring
ExecStart=/usr/local/bin/alertmanager \
  --config.file=/etc/prometheus/alertmanager.yml \
  --web.listen-address=0.0.0.0:XXXX \
  --storage.path=/var/lib/alertmanager

SyslogIdentifier=monitoring
Restart=always

[Install]
WantedBy=multi-user.target
```

Commandes système et visualisation des logs en direct :

```
=> systemctl daemon-reexec
=> systemctl daemon-reload
=> systemctl enable alertmanager.service
=> systemctl start alertmanager.service
=> systemctl status alertmanager.service
=> journalctl -f -u alertmanager.service
```

On a changé ici le port d'écoute de Alert Manager de Prometheus ici, dans le fichier du service. Ca ne se fait pas dans le .yml . (Voir DOC DE SECU, sur le FW ou le SERVEUR pour savoir le port)
Création au cas ou, de l'objet PORT pour Alert Manager.

Voila pour l'installation de AlertManager, maintenant il faudrait configurer des règles en créant un fichier "alert.rules.yml" par exemple avec les règles dedans et activer le bloc "rule_files" dans le prometheus.yml . (Voir chatgpt)

On va installer pour la fin, NODE EXPORTER pour récupérer des informations plus précises que avec SNMP, qui surveillera les serveurs directs et non les équipements réseaux.

=> Dans /usr/local/bin/ , on télécharge NODE Exporter qui va nous servir à récupérer les métriques du serveur de monitoring ici.

=> wget https://github.com/prometheus/node_exporter/releases/download/v1.9.0/node_exporter-1.9.0.linux-amd64.tar.gz
=> tar -xvzf node_exporter-1.9.0.linux-amd64.tar.gz
=> rm node_exporter-1.9.0.linux-amd64.tar.gz
=> mv /usr/local/bin/node_exporter-1.9.0.linux-amd64/node_exporter /usr/local/bin/
=> chown -R monitoring:monitoring /usr/local/bin/node_exporter
=> chmod 755 /usr/local/bin/node_exporter
=> rm -r /usr/local/bin/node_exporter-1.9.0.linux-amd64/

=> Création du fichier : nano /etc/systemd/system/node_exporter.service

---

```
[Unit]
Description=Prometheus NODE Exporter
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
User=monitoring
Group=monitoring
ExecStart=/usr/local/bin/node_exporter \
  --web.listen-address=0.0.0.0:XXXX \
  --collector.systemd \
  --collector.processes

SyslogIdentifier=monitoring
Restart=always

[Install]
WantedBy=multi-user.target
```

On a changé ici le port d'écoute de SNMP EXPORTER de Prometheus ici, dans le fichier du service. Ca ne se fait pas dans le .yml car il n'y pas de fichier .yml pour NODE Exporter .

Commandes système et visualisation des logs en direct :

```
=> systemctl daemon-reexec
=> systemctl daemon-reload
=> systemctl enable node_exporter.service
=> systemctl start node_exporter.service
=> systemctl status node_exporter.service
=> journalctl -f -u node_exporter.service
```

On a ajouter le bloc "job_name" dans le fichier /etc/prometheus/prometheus.yml :

```
  - job_name: 'NODE-SRV-MONITORING-AARD'
    static_configs:
      - targets: ['localhost:XXXX']

  - job_name: 'NODE-SRV-LOG-AARD'
    static_configs:
      - targets: ['192.168.7.133:XXXX']

  - job_name: 'NODE-SRV-WEB-AARD'
    static_configs:
      - targets: ['172.16.7.1:XXXX']

  - job_name: 'NODE-SRV-BASTION-AARD'
    static_configs:
      - targets: ['172.16.14.1:XXXX']

  - job_name: 'NODE-SRV-RPROXY-AARD'
    static_configs:
      - targets: ['172.16.7.129:XXXX']

  - job_name: 'NODE-TNAS-F4-423'
    static_configs:
      - targets: ['192.168.7.130:XXXX']

  - job_name: 'NODE-SRV-RDS-AARD'
    static_configs:
      - targets: ['192.168.7.132:XXXX']

  - job_name: 'NODE-SRV-BDD-PROD-AARD'
    static_configs:
      - targets: ['192.168.7.145:XXXX']

  - job_name: 'NODE-SRV-SVG-AARD'
    static_configs:
```

```
      - targets: ['192.168.7.131:XXXX']

  - job_name: 'NODE-SRV-BDD-PREPROD-AARD'
    static_configs:
      - targets: ['192.168.7.146:XXXX']

  - job_name: 'NODE-SRV-WEB-PREPROD-AARD'
    static_configs:
      - targets: ['172.16.7.2:XXXX']

  - job_name: 'PROMTAIL-VERSION-SRV-MONITORING-AARD'
    static_configs:
      - targets: ['localhost:9080']

  - job_name: 'NODE-SRV-UCS-AARD'
    static_configs:
      - targets: ['192.168.7.140:59051']
```

Création au cas ou, de l'objet PORT pour NODE EXPORTER.

Voila pour l'installation de NODE EXPORTER.

=> On installe node exporter sur toute les machines sauf la windows car celle-ci ne sera plus présente.

Je ne pense pas installé d'autres modules avec Prometheus autre que SNMP EXPORTER et ALERT MANAGER et NODE EXPORTER. Etant en maquette, il nous faut juste quelques exemples.

Je fais une règle pour autoriser l'envoi des infos pour les métriques de toutes les machines comme fait avec loki.

Pour configurer des DASHBOARDS ou utiliser grafana, une vidéo sera disponible en privé sur la chaine YT.

=> Ajout du MOTD personnalisé sur la VM (et toutes) du NAS.
/etc/motd

Wizards & Dice | Shell Access for Debian GNU/Linux

```
**************************************************************************
*****************************************
*                                        LEGAL NOTICE
* This system is for authorized use only. All activities on this system
may be * monitored and recorded. Unauthorized access or use is strictly
prohibited * and may result in disciplinary action, criminal prosecution,
or both. By * continuing to use and access this system, you consent to
such monitoring.*

**************************************************************************
*****************************************
```

=> Ajout des éléments pour le monitoring des machines et de leurs MAJS.

      => wget https://raw.githubusercontent.com/labmonkeys-space/apt-prometheus/main/script/apt-metrics.sh -O /usr/local/bin/apt-metrics.sh
      => chmod 750 /usr/local/bin/apt-metrics.sh
      => chown monitoring /usr/local/bin/apt-metrics.sh
      => Puis on change tout le fichier :

```
#!/bin/bash

OUT_FILE="/var/lib/node_exporter/textfile_collector/apt_updates.prom"

# Total updates
TOTAL_UPDATES=$(apt list --upgradeable 2>/dev/null | grep -v "Listing" | wc -l)

# Security updates (si 'unattended-upgrades' est installé)
SEC_UPDATES=$(apt list --upgradeable 2>/dev/null | grep security | wc -l)

# Reboot required (flag fichier)
if [ -f /var/run/reboot-required ]; then
  REBOOT_NEEDED=1
else
  REBOOT_NEEDED=0
fi

# Export en format Prometheus
echo "# HELP debian_apt_updates Nombre total de mises à jour APT" > "$OUT_FILE"
echo "# TYPE debian_apt_updates gauge" >> "$OUT_FILE"
echo "debian_apt_updates $TOTAL_UPDATES" >> "$OUT_FILE"

echo "# HELP debian_security_updates Nombre de mises à jour de sécurité" >> "$OUT_FILE"
echo "# TYPE debian_security_updates gauge" >> "$OUT_FILE"
echo "debian_security_updates $SEC_UPDATES" >> "$OUT_FILE"

echo "# HELP debian_reboot_required Système nécessite un redémarrage" >> "$OUT_FILE"
echo "# TYPE debian_reboot_required gauge" >> "$OUT_FILE"
echo "debian_reboot_required $REBOOT_NEEDED" >> "$OUT_FILE"
```

Dans le node_exporter.service

```
[Unit]
Description=Prometheus NODE Exporter
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
User=monitoring
Group=monitoring
ExecStart=/usr/local/bin/node_exporter --web.listen-address=0.0.0.0:XXXX
--collector.systemd --collector.processes --collector.textfile --
collector.textfile.directory=/var/lib/node_exporter/textfile_collector

SyslogIdentifier=monitoring
Restart=always

[Install]
WantedBy=multi-user.target
```

```
=> mkdir -p /var/lib/node_exporter/textfile_collector
=> chown -R monitoring:root /var/lib/node_exporter/textfile_collector
=> chmod -R 770 /var/lib/node_exporter/textfile_collector

            => systemctl daemon-reexec
            => systemctl daemon-reload
            => systemctl restart node_exporter.service
            => systemctl status node_exporter.service
```

Il faut maintenant faire un cron, pour executer le script "apt-metrics.sh" régulièrement :

```
=> crontab -e

30 7 * * * /usr/local/bin/apt-metrics.sh

=> systemctl restart cron.service
```

Maintenant il suffit sur grafana avec le dashboard, de regarder ce que nous voulons voir.

Passage en HTTPS pour le web GRAFANA :

1Créer le répertoire sur UCS (Vérifier les droits)

=> mkdir /etc/univention/ssl/SRV-MONITORING-AARD.wnd.local

2Générer la clé privée

=> openssl genrsa -out /etc/univention/ssl/SRV-MONITORING-AARD.wnd.local/private.key 2048

3Générer la demande de certificat (CSR)

=> openssl req -new -key /etc/univention/ssl/SRV-MONITORING-AARD.wnd.local/private.key -out /etc/univention/ssl/SRV-MONITORING-AARD.wnd.local/req.pem -subj "/CN=SRV-MONITORING-AARD.wnd.local"

4Signer avec la CA UCS

=> univention-certificate new -name "SRV-MONITORING-AARD.wnd.local" -days 3650

✅ Cela génère automatiquement :

=> /etc/univention/ssl/SRV-MONITORING-AARD.wnd.local/cert.pem

=> /etc/univention/ssl/SRV-MONITORING-AARD.wnd.local/private.key

=> signés par ucs-root-ca.

Sur le serveur monitoring / Grafana

=> mkdir /etc/grafana/certs
=> chown -R root:grafana /etc/grafana/certs
=> chmod -R 750 /etc/grafana/certs

5Copier les clés sur le serveur monitoring

Depuis l'UCS, faire :

=> scp /etc/univention/ssl/SRV-MONITORING-AARD.wnd.local/cert.pem root@192.168.7.134:/etc/grafana/certs/grafana.crt
=> scp /etc/univention/ssl/SRV-MONITORING-AARD.wnd.local/private.key root@192.168.7.134:/etc/grafana/certs/grafana.key

Sur serveur monitoring :

=> chgrp grafana /etc/grafana/certs/grafana.crt /etc/grafana/certs/grafana.key

6Configurer Grafana /etc/grafana/grafana.ini

Dans /etc/grafana/grafana.ini [server] :

=> protocol = https

=> http_port = 58272
=> cert_file = /etc/grafana/certs/grafana.crt
=> cert_key = /etc/grafana/certs/grafana.key

7 Redémarrer Grafana

=> systemctl restart grafana-server

Normalement il faut etre sur que le "ucs-root-ca.crt", soit présent sur le serveur /etc/local/share ... et de mettre l'exception à l'accès sur le site grafana pour du HTTPS sur Firefox dans "certificates" .... .

Maintenant il suffit sur grafana avec le dashboard, de regarder ce que nous voulons voir.

Mise en place de UFW :

=> apt install ufw

Politique par défaut

=> ufw default deny incoming
=> ufw default deny outgoing

Maintenant les règles précises :

=> ufw allow from 172.16.14.1 to any port 14714 proto tcp comment 'SSH depuis bastion'
=> ufw allow from 192.168.7.132 to any port 14714 proto tcp comment 'SSH depuis SRV-RDS-AARD'
=> ufw allow from 192.168.7.128/28 to any port 9080 proto tcp comment 'VUE WEB DES TARGETS PROMTAIL'
=> ufw allow from 192.168.7.128/28 to any port 47171 proto tcp comment 'VUE WEB DES ALERTMANAGER PROMETHEUS'
=> ufw allow from 192.168.7.128/28 to any port 58419 proto tcp comment 'VUE WEB DES METRIQUES SNMP PROMETHEUS'
=> ufw allow from 192.168.7.128/28 to any port 49080 proto tcp comment 'VUE WEB DES REQUETES PROMETHEUS'
=> ufw allow from 192.168.7.134 to any port 59051 proto tcp comment 'Prometheus avec node_exporter'
=> ufw allow from 192.168.7.128/28 to any port 58272 proto tcp comment 'HTTPS Grafana'
=> ufw allow out 59051/tcp comment 'Scraping des métriques sur infrastructure'
=> ufw allow out to 192.168.7.133 port 3100 proto tcp comment "Accès à Loki pour logs"
=> ufw allow out 80/tcp comment 'Sortie HTTP vers dépôts Linux'
=> ufw allow out 443/tcp comment 'Sortie HTTPS vers dépôts Linux'
=> ufw allow out to 192.168.7.140 port 53 proto udp comment 'Sortie DNS vers SRV-UCS-AARD'
=> ufw allow out 123/udp comment 'NTP'
=> ufw allow out 587/tcp comment 'Envoi de mails via GRAFANA'

=> ufw enabled

Pour ajouter la possibilité de PING, il faut modifier le fichier /etc/ufw/before.rules et ajouter :

```
# allow ICMP outbound (ping vers l'extérieur)
-A ufw-before-output -p icmp --icmp-type echo-request -j ACCEPT
-A ufw-before-output -p icmp --icmp-type destination-unreachable -j ACCEPT
-A ufw-before-output -p icmp --icmp-type time-exceeded -j ACCEPT
-A ufw-before-output -p icmp --icmp-type echo-reply -j ACCEPT
```

Voila pour UFW, il faudra voir si d'autres problèmes sont présent, comme ça on ajoute ou enleve des règles.

Installation lynis + fail2ban

=> Executer lynis dans syslog : lynis audit system | logger -t lynis

OU

=> Executer le script dans /usr/local/bin/lynis-syslog.sh

nano /usr/local/bin/lynis-syslog.sh

Script :

```
#!/bin/bash
lynis audit system | logger -t lynis
```

```
=> chmod 750 /usr/local/bin/lynis-syslog.sh
=> chown monitoring /usr/local/bin/lynis-syslog.sh
```

Pas de cron.

On peut voir depuis GRAFANA dans les logs, le résultat.

Fail2Ban :

```
=> apt install fail2ban -y
=> systemctl enable fail2ban
=> systemctl start fail2ban
```

On ne modifie jamais directement jail.conf. On crée un fichier jail.local :

```
=> cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
=> nano /etc/fail2ban/jail.local
```

```
[sshd]
enabled = true
port    = ssh
logpath = %(sshd_log)s
backend = %(sshd_backend)s
maxretry = 5
bantime = 500
findtime = 600
```

```
=> systemctl restart fail2ban
=> fail2ban-client status
=> fail2ban-client status sshd
```

Fail2Ban lit les logs /var/log/auth.log (par défaut) pour détecter les tentatives SSH.