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

#     Date de création : 07/02/2025                         #
#                                                           #
#     Dernier modificateur : Arthur YANG            #
#     Date de modification : 07/02/2025             #
#                                                           #
#           Version actuelle : 1.0                  #
#                                                           #
#################################################################


----------------------- Déploiement du serveur de monitoring ------------
-----------

Sources : https://grafana.com/docs/grafana/latest/setup-
grafana/installation/debian/
            https://grafana.com/docs/grafana/latest/getting-
started/build-first-dashboard/

Création du conteneur LXC :
   => Nom : SRV-MONITORING-AARD
   => OS : Debian-12-standard_12.7-1_amd64.tar.zst
   => Stockage : local -> local_zfs
   => Mot de passe root conteneur : Voir Bitwarden

Configuration réseau :
   => Interface : in - VLAN 20 (servers)
   => IP : 192.168.7.134
   => Masque : 255.255.255.240
   => Passerelle : 192.168.7.142
   => DNS : 192.168.7.129


Allocation des ressources :
   => CPU : 1 cœur
   => RAM : 512 Mb
   => Stockage : 16 Gb

Actions effectuées dans le conteneur :

   => Connexion via root / mot de passe
   => apt update
   => apt upgrade
   => Création utilisateur "monitoring" et groupe "monitoring" / mdp :
Voir Bitwarden, si besoin plus tard via adduser.



Actions pour Grafana et son installation (avec le compte en root) :
```

Remarques :

- Choix d'installer Grafana Enterprise (gratuit et similaire à Grafana OSS)
- Nous installons à partir du dépôt APT, car si nous installons depuis le dépôt APT, Grafana est automatiquement mis à jour lorsque nous exécutons "apt-get update".

=> Installez les packages requis : apt-get install -y apt-transport-https software-properties-common wget
=> Import de la clé GPG Grafana :
=> mkdir -p /etc/apt/keyrings/
=> wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | tee /etc/apt/keyrings/grafana.gpg > /dev/null
=> Ajout d'un dépot pour les versions stables : echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com stable main" | tee -a /etc/apt/sources.list.d/grafana.list
=> Ajout d'un dépot pour les versions en beta : echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com beta main" | tee -a /etc/apt/sources.list.d/grafana.list
=> Exécution de la commande suivante pour mettre à jour la liste des paquets disponibles : sudo apt-get update
=> Pour installer Grafana Enterprise : apt-get install grafana-enterprise
=> Démarrer le serveur Grafana : service grafana-server start
=> Pour vérifier que le service fonctionne : service grafana-server status
=> Pour configurer le serveur Grafana afin qu'il démarre au démarrage : systemctl enable grafana-server
=> vérifier que le service est démarrer au démarrage : systemctl is-enabled grafana-server
=> Remarque : le port par défaut de Grafana est 3000, vérifier que Grafana écoute bien sur le port 3000 : ss -tulnp | grep 3000
=> Tester l'accès HTTP avec curl : curl -I http://localhost:3000 OU dans une barre de recherche : http://@IP:3000
=> Facultatif (sécurité), modifier les paramètres du serveur Grafana via fichier de configuration : /etc/grafana/grafana.ini

Sur la page de connexion (http://192.168.7.134:XXXX(VOIR NOUVEAU PORT)) depuis un poste, nous pouvons entrer par défaut "admin" pour le nom d'utilisateur et le mot de passe (voir bitwarden).

Cela nous authentifie, et nous arrivons sur la page Web de Grafana.

Nous pouvons aller dans "ajouter votre première source de donnée" pour configurer Grafana avec Loki.

=> URL : http://192.168.7.133:XXXX (Voir FW ou document pour récupérer le port)
=> Save & test pour voir si la communication du SRV-MONITORING-AARD communique avec SRV-LOG-AARD

Mise en place d'un DASHBOARD pour tester :

=> Choix de la database LOKI
=> Ensuite, c'est effectuer des requêtes. Soit des filename, job etc ... et le choix du chemin (var/log/svglog) etc ...

```
#################################################################
#                                                               #
#     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;"""
```

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))
```

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

[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)

```
      - 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
```

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:
```

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

#     Date de création : 07/02/2025                         #
#                                                           #
#     Dernier modificateur : Arthur YANG            #
#     Date de modification : 07/02/2025             #
#                                                           #
#           Version actuelle : 1.0                  #
#                                                           #
#################################################################


-----------------------  Déploiement du serveur web  --------------------
---

Sources : https://www.it-connect.fr/installation-de-wordpress-sous-linux/
           https://www.it-connect.fr/installer-un-serveur-lamp-linux-
apache-mariadb-php-sous-debian-11/

Création du conteneur LXC :
   => Nom : SRV-WEB-AARD
   => OS : Debian-12-standard_12.7-1_amd64.tar.zst
   => Stockage : local -> local_zfs
   => Mot de passe root conteneur : Voir Bitwarden

Configuration réseau :
   => Interface : DMZ publique - VLAN 100 (WEB)
   => IP : 172.16.7.1
   => Masque : 255.255.255.128
   => Passerelle : 172.16.7.126
   => DNS : 1.1.1.1 (Zone DMZ publique)

Allocation des ressources :
   => CPU : 1 cœur
   => RAM : 512 Mb
   => Stockage : 8 Gb


Actions effectuées dans le conteneur :

   => Connexion via root / mot de passe
   => apt update
   => apt upgrade
   => apt install curl
   => apt install unzip
   => apt install mariadb-client
   => Création utilisateur "web" et groupe "web" / mdp : voir Bitwarden
si besoin plus tard via adduser.
```

Actions pour Apache et Wordpress et leur installation (avec le compte en root) :

Remarques :
- Choix d'installer MySQL ou MariaDB sur le serveur de BDD.
- la commande "a2enmod" qui sert à activer un module. A l'inverse, la commande "a2dismod" sert à désactiver un module.
- Le fichier de configuration d'Apache 2 est le suivant :

=> /etc/apache2/apache2.conf

Dans un premier temps, il peut servir à configurer Apache pour ne pas afficher le numéro de version sur les pages d'erreurs. Même si cette option est gérable aussi dans le fichier "/etc/apache2/conf-enabled/security.conf", c'est au choix.

- Pour la configuration qui concerne PHP, le fichier de configuration est différent : "/etc/php/7.4/apache2/php.ini"

A. L'archive d'installation de WordPress

Connectez-vous sur votre serveur Linux en SSH afin de télécharger l'archive ZIP qui contient les sources de WordPress.

Positionnez-vous dans le dossier "/tmp" et téléchargez la dernière version de WordPress :

```
=> cd /tmp
=> wget https://wordpress.org/latest.zip
```

B. Le serveur Web (Apache)

Installer Apache :

```
=> apt-get install -y apache2
```

Pour qu'Apache démarre automatiquement en même temps que Debian, saisissez la commande ci-dessous (même si normalement c'est déjà le cas) :

```
=> systemctl enable apache2
```

Activation de quelques modules d'Apache qui sont indispensables, notamment pour faire tourner un site Internet :

```
=> a2enmod rewrite
=> a2enmod headers
=> a2enmod deflate
=> a2enmod ssl
```

Après avoir activé ou désactivé un module, ou modifié la configuration d'Apache, il faut redémarrer le service apache2 :

```
=> systemctl restart apache2
```

C. Installer PHP

PHP va venir se greffer sur notre serveur Apache, comme une extension, afin de pouvoir traiter les scripts intégrés aux pages ".php". Afin d'y aller progressivement, installons le paquet "php" en lui-même :

```
=> apt-get install -y php
```

On peut voir que cette commande va installer une multitude de paquets.

Avant d'aller plus loin, nous allons installer quelques paquets supplémentaires pour compléter l'installation de PHP sur notre serveur. Par exemple, pour permettre les interactions entre PHP et notre instance MariaDB/MySQL.

```
=> apt-get install -y php-pdo php-mysql php-zip php-gd php-mbstring php-curl php-xml php-pear php-bcmath
```

D. Décompresser l'archive WordPress à la racine du site

Nous allons utiliser le site par défaut d'Apache, qui a pour racine "/var/www/html" afin de stocker les données de notre site WordPress. Au préalable, on supprime la page d'index créée par défaut par Apache :

```
=> rm /var/www/html/index.html
```

On décompresse l'archive dans "/var/www/html" grâce à la commande suivante (en étant positionné dans le dossier où l'on a téléchargé le fichier latest.zip) :

```
=>  unzip latest.zip -d /var/www/html
```

Le dossier WordPress apparaitra donc dans "/var/www/html" qui est le dossier où sont stockées les pages web par défaut.

Déplacez-nous dans le dossier "/var/www/html" et exécutons la commande ci-dessous pour déplacer tout le contenu du dossier "wordpress" à la racine de notre site :

```
=>  mv wordpress/* /var/www/html/
```

Puisque le dossier "wordpress" ne sert plus à rien, on va le supprimer :

```
=>  rm wordpress/ -Rf
```

Enfin, on termine en donnant les droits à l'utilisateur "www-data" (correspondant à Apache) sur tous les fichiers de notre site, de manière récursive :

=> chown -R www-data:www-data /var/www/html/

On obtient une liste de fichiers et dossiers. Au niveau des droits et pour des raisons de sécurité, nous devons avoir 755 sur les dossiers et 644 sur les fichiers. Ce qui est le cas par défaut si nous n'avons pas fait de modifications. En aucun cas nous ne devons poser des droits "777" sur un dossier ou un fichier.

Si vous avez un doute ou que vous pensez avoir modifié les droits, vous pouvez rectifier la situation.

Pour les fichiers, exécutez cette commande :

=> find /var/www/html/ -type f -exec chmod 644 {} \;

Pour les dossiers, exécutez cette commande :

=> find /var/www/html/ -type d -exec chmod 755 {} \;

E. Installation de WordPress

Pour la première fois, nous allons nous connecter sur l'interface web WordPress dans le but d'effectuer l'installation. Pour cela, il faut se rendre sur "http://IP-SERVEUR" (Pour nous c'est : http://172.16.7.1)

La première étape consiste à choisir la langue du site et de l'interface de WordPress.

Ensuite il faut entrer les informations de la base de données :

=> Nom de la base de données : ce sera "wp202502_wnd"
=> Identifiant : le nom de l'utilisateur qui a les droits sur la base de données, en l'occurrence "adminwp202502_wnd"
=> Mot de passe : le mot de passe de cet utilisateur : Voir Bitwarden
=> Adresse de la base de données : notre serveur web et la BDD ne sont pas sur le même serveur, donc nous c'est : "192.168.7.145:PORTS" (on précise les ports que nous avons changés lors du déploiement du serveur BDD)
=> Préfixe des tables : chaque table de la base de données WordPress aura un préfixe. Par défaut, ce préfixe est "wp" donc par exemple la table des utilisateurs sera nommée "wp_users". On met pour nous "wp_wnd_".

WordPress va tester de se connecter à notre base de données et si cela fonctionne, un bouton "Lancer l'installation" va s'afficher. Cliquons dessus.

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

#     Date de création : 19/02/2025                         #
#                                                               #
#     Dernier modificateur : Arthur YANG            #
#     Date de modification : 19/02/2025             #
#                                                               #
#           Version actuelle : 1.0                  #
#                                                               #
#################################################################


----------------------- Configuration et sécurisation du serveur web prod
-----------------------

Sources : https://artheodoc.wordpress.com/wp-content/uploads/2022/01/1-
serveur_web_apache_debian_11_pour_wordpress_nom_de_domaine_-https.pdf
           https://les-enovateurs.com/debian-vps-apache-mysql-php-
wordpress


Actions effectuées dans la VM :

   => Connexion via root / mot de passe
   => apt update
   => apt upgrade


Installation de l'agent promtail (en root) :

      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 :
```

=> Hébergeur SMTP : smtp.gmail.com
=> Cryptage : TLS
=> Port : 587
=> Authentification activé
=> ID SMTP : svg.wizardsndice@gmail.com
=> Mdp smtp : Le mot de passe d'application créer dans les paramètres de la boite mail SVG

Sauvegarder ensuite, puis nous pouvons tester l'envoi de mail depuis les plugins installés, tout fonctionne normalement.
Les plugins étant gratuit, ne donnent pas vraiment d'informations sur les logs dans le rapport, néanmoins ils pourraient être utile et plus largement détaillés avec une version payante.

Nous configurons aussi des filtres et étiquettes pour la réception de ces mails sur la boite GMAIL SVG. Selon si c'est de DRYN ou AARD en fonction de l'ip source pour le moment.

Nous avons donc maintenant pour WordPress le monitoring des logs sur WordPress directement. Il faudrait faire des configurations plus "avancées" pour pouvoir avoir ces logs sur GRAFANA, mais demande plus d'investigation. Néanmoins les logs d'apache sont bien dispo sur GRAFANA.

Après cette phase de config des logs / monitoring, on peut passer au configs suivantes.

Nous allons ajouter la possibilité de pouvoir se connecter avec des identifiant ldap sur le wordpress.
Pour cela, nous installons le plugin "Active Directory / LDAP Intergration".

Il faut aussi ajouter l'extension PHP LDAP.

=> Étape 1
Pour Ubuntu/Debian, la commande d'installation serait sudo apt-get -y install php-Idap.
Recherchez extension=php_ldap.so dans le fichier /etc/php/8.2/apache2/php.ini. Décommentez cette ligne, si elle n'est pas présente, ajoutez cette ligne dans le fichier et enregistrez le fichier
Dans notre cas, on ajoute cette ligne.

Étape 3
Redémarrez le serveur. Ensuite, actualisez la page de configuration du plugin LDAP/AD

On peut passer ensuite a la config :

=> On ouvre les ports 389 du SRV DC vers le serveur WEBPREPROD

=> On créer un compte LDAP pour la liaison : wordpress.wp (MDP sur bitwarden)
=> Service Account Username : wordpress.wp@wnd.local
=> Teste la liaison : OK
=> Search Base : dc=wnd,dc=local
=> SAMAccountName = wnd.local\user
=> On peut tester avec nos user : OK donc config de base FINI

Après on peut personnaliser les rôles etc ...
=> faire du role mapping (Le mappage de rôles permet d'attribuer un rôle WordPress en fonction du groupe LDAP auquel appartient un utilisateur sur le serveur LDAP).
=> Néanmoins, comme nous avons la version gratuite, nous pourrons pas le faire.
=> Donc par défaut nos comptes LDAP sont des abonnées du WP. Nous pourrons modifier nos rôles a nous une fois connecté une fois sur WP pour etre admin.

Dans le menu "Attribute Mapping",

=> Infos sur cette partie : https://faq.miniorange.com/knowledgebase/configure-attribute-mapping-in-ldap/
=> nous laissons "username@email_domain" dans la premiere case, rien dans la 2eme et les autres rien car abonnement gratuit

Ensuite dans le dernier menu, nous autorisons tous ce que nous pouvons jusqu'aux options au abonnés.

Nous pouvons tester le ldap sur la page d'accueil de connexion => Ca fonctionne
Mais nous sommes juste lecteur, donc avec le compte "superadmin", nous pouvons modifier le role et mettre admin dans le menu des comptes.
Il faut obligatoirment mettre un mail aux users, j'ai inventé pour le moment.

Voila pour le LDAP intégré sur Wordpress.

Nous pouvons suivre les recommendés de WP dans "l'état de santé du site" dans le menu "tableau de bord" qui permet de savoir au niveau conf perf,sécu ce que nous pouvons faire.
Au niveau des conf performances, nous mettre en place les modules PHP importants qui ne sont pas forcément présent mais qui servent pour faire tourner le site.

Les modules nécessaires sont :
=> imagick
=> intl

Nous installons donc ces modules sur le serveur web :
=> apt install php-imagick -y
=> apt install php-intl -y

Puis on restart le service apache2.
Les modules sont installés et serviront pour plus tard.

Il faut aussi voir la mise en cache des pages web/wordpress.
(Elles permettent l'upgrade de la vitesse et perf du site en servant des pages statiques au client au lieu de tout le temps générer dynamiquement à chaque visite)

Pour cela, nous installons un plugin : "W3 Super Cache"
Après l'installation, il va falloir autoriser plus de droits sur le serveur WEB et notamment le fichier wp-config.php (Actuellement en 400)

=> chmod 600 /var/www/html/wp-config.php
=> chmod -R 777 /var/www/html/wp-content
Attention : Ce 2ème réglage est temporaire. Une fois WP Super Cache configuré, il faudra remettre des permissions plus sécurisées.

Puis restart du service apache2, rafraichir la page WP sur le web.
Maintenant on repasse à des droits plus restrictifs :

=> chmod -R 755 /var/www/html/wp-content

Apres restart encore du service apache2 et page admin web de WP
On voit que la page réglage est enlevé au final apres nos parametrages.
L'installation est ainsi finit.
Maintenant il faut continuer à configurer ce plugin. Mais nous le ferons dans un autre temps.

Voila pour les reco de WP mais il pourrait en avoir d'autres bien sur.

Pour mettre en place Woocommerce, il nous faudrais les prérequis suivant :

=> Un nom de domaine (Pas encore obtenue / wizardsndice.fr)
=> Hébergement Web (obtenue)
=> Téléchargement de WordPress (obtenue)
=> Installation d'un certificat SSL (obtenue, c'est le reverse proxy qui s'occupe du certificat SSL)

Donc comme nous n'avons pas tout sous la main pour l'instant, nous installons et configurons au maximum WooCommerce pour le moment. Cela fonctionne quand meme, meme sans le nom de domaine. c'est juste que la config à 100% ne sera pas faite.

Pour cela, nous installons le plugin Woocommerce sur WP :
=> Puis nous commençons l'installation avec l'assistance
=> Nous mettons l'adresse wizardsndice@outlook.com
=> Vêtements et accessoires
=> On laisse cocher les fonctionnalités sauf pour l'IA

Maintenant c'est que de la config qu'il faut faire, par exemple :

=> Il y a parfois des configs vraiment obligatoire, par exemple le fait que le site soit juste privé, il faut configuré le module jetpack de WooCommerce etc …

=> On supprime le plugin MailPoet qui pour l'instant ne servira dans les extensions

Nous allons juste faire en sorte que des gens peuvent s'authentifier / créer des comptes.

=> Il faut juste dans les paramètres/réglages de WooCommerce => puis le menu 'comptes et condidentialité' , on peut activer la création de compte.

=> Néanmoins, la création fonctionne mais lorsque nous nous déconnectons de l'utilisateur et que nous voulons nous reconnecter avec, celui-ci ne fonctionne plus alors qu'il est toujours bel et bien présent dans la BDD. Je ne sais pas l'erreur actuelle mais elle est peut être lié au fait que n'avons pas fini à l'heure actuelle, de configurer WooCommerce ou Wordpress de manière générale.

Mais voila, a partir de la et comme tout a l'heure, il faut juste continuer à configurer WooCommerce. Donc on a un déploiement et une configuration vraiment légère.

=> Il faudrait voir plus tard comment enlever la page d'authentification de Wordpress d'origine pour éviter que client accède dessus. (surement enlever le wp-admin.php)

Avant de passer aux fichiers de configs apache ou php sur le serveur directement, nous devrons installer un autre plugin qui permettra la redirection de la page "wp-login.php" vers la page web classique, pour permettre l'authentification dessus et pas depuis la page "wp-login.php"

De même pour la page "wp-admin" qui les redirige autre part si ils n'ont pas le droit ou veulent y acceder directement.

Donc cette partie qui serait plus dans la partie sécu, sera à pensé lorsque tout sera mis en place sur les 2 sites.

On repasse maintenant coté serveur pour faire un tour de la configuration APACHE/PHP :

=> Vérification des modules Apache activés

=> a2enmod rewrite headers expires

=> systemctl restart apache2

- rewrite : Nécessaire pour les permaliens WordPress.
- headers : Utile pour la sécurité et les réglages de cache.
- expires : Aide à la gestion du cache des ressources statiques.

Après cela, on va faire en sorte d'utiliser un virtualhost différent de celui-par défaut

```
      Celui qu'on utilise est le 000...
            => on peut le voir avec la commande : ls -l
/etc/apache2/sites-enabled/

Il sera pas si différent mais au moins on aura un "personnalisé"
      on copie le virtualhost de base vers un autre fichier :
            => cp /etc/apache2/sites-available/000-default.conf
/etc/apache2/sites-available/wizardsndice.conf

Maintenant on peut le personnaliser "wizardsndice.conf" :

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

<VirtualHost *:80>
    ServerName wizardsndice.fr
    ServerAlias 172.16.7.1 wizardsndice.wnd.local
    ServerAdmin svg.wizardsndice@gmail.com
    DocumentRoot /var/www/html

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    <Directory /var/www/html>
        AllowOverride All
        Require all granted
    </Directory>

    # Empêcher l’exécution de scripts PHP dans /uploads et /wp-includes
    <Directory "/var/www/html/wp-content/uploads">
        Require all denied
    </Directory>

    <Directory "/var/www/html/wp-includes">
        Require all denied
    </Directory>

    # Bloquer l’accès aux fichiers sensibles
    <FilesMatch "(wp-config.php|xmlrpc.php|.htaccess|.htpasswd)">
        Require all denied
    </FilesMatch>

    # Empêcher l'affichage des fichiers d'un dossier (évite de voir la
liste des fichiers)
    Options -Indexes

    # Headers utiles pour la sécurité et la compatibilité avec le reverse
proxy
      Header set X-Forwarded-Proto "https"
      Header set X-Frame-Options "SAMEORIGIN"
      Header set X-XSS-Protection "1; mode=block"
      Header set X-Content-Type-Options "nosniff"
      Header set Content-Security-Policy "default-src 'self'"
```

```
</VirtualHost>
```

Ensuite on regarde la config de "/var/www/html/.htaccess" si elle est bien configuré :

```
      => nano /var/www/html/.htaccess
```

Il faut qu'elle soit comme ceci :

```
# BEGIN WordPress
# Les directives (lignes) entre     BEGIN WordPress     et     END
WordPress     sont g  n  r  es
# dynamiquement, et doivent   tre modifi  es uniquement via les filtres
WordPress.
# Toute modification des directives situ  es entre ces marqueurs sera
surcharg  e.
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
RewriteBase /
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [L]
</IfModule>

# END WordPress
```

Puis, on s'assure que le fichier a les bonnes permissions :

```
      => sudo chown www-data:www-data /var/www/html/.htaccess
      => sudo chmod 644 /var/www/html/.htaccess
```

Enfin on restart le service apache et charge le nouvelle hote virtuel "wizardsndice.conf" :

```
      => a2ensite wizardsndice.conf (se mettre dans le bon répertoire)
      => systemctl restart apache2
      => a2dissite 000-default.conf
```

On peut aussi vérif quels sites sont activés avec : => apachectl -S

Normalement cela utilise maintenant notre config créer plus tot sur le site web.

Autre configurations que l'on va faire meme si on pourrait le mettre dans la partie "Sécu" :

=> Cacher les informations de version d'Apache et Debian sur les pages d'erreur
=> Éditer le fichier /etc/apache2/conf-available/security.conf :
=> Ajoute ou modifie ces lignes :
=> ServerTokens Prod
=> ServerSignature Off
=> on restart apache

Pour finir la partie configuration (qui pourrait être encore plus longue), on personnaliser le fichier /etc/apache2/apache2.conf :

=> Timeout 30
=> Ligne à ajouter :
=> <IfModule mod_deflate.c>
AddOutputFilterByType DEFLATE text/plain text/css text/javascript application/javascript text/xml application/xml application/json
</IfModule>
=> Modif de la ligne : <FilesMatch "^\.ht"> PAR <FilesMatch "(^\.ht|\.git|\.env|composer.json|composer.lock)">

=> Puis dans cette partie :

```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>
```

AVEC

```
<Directory /var/www/>
    Options -Indexes +FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>
```

=> La suite dans ce fichier de conf pourra se faire dans la partie "Sécu" avec les .htaccess. etc ... et plus si besoin.

Voila pour la partie configuration. Comme dit plus tot, il pourrait y avoir encore plein de configuration possible, mais ceci demanderait plus de temps et de reflexion.

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.

```
WantedBy=multi-user.target

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

=> mkdir -p /var/lib/node_exporter/textfile_collector
=> chown -R web: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

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

-------------------------------------------------------------------------------
-----------------------------------------------
=> crontab -e

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

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

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 192.168.7.134 to any port 59051 proto tcp comment
'Prometheus avec node_exporter'
=> ufw allow from 172.16.7.129 to any port 80 proto tcp comment 'HTTP
depuis PROXY'
=> ufw allow from 172.16.7.129 to any port 443 proto tcp comment 'HTTPS
depuis PROXY'
=> ufw allow from 192.168.7.132 to any port 80 proto tcp comment 'HTTP
depuis SRV-RDS-AARD'
```

```
=> ufw allow from 192.168.7.132 to any port 443 proto tcp comment 'HTTPS
depuis SRV-RDS-AARD'
=> 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 1.1.1.1 port 53 proto udp comment 'Sortie DNS vers
1.1.1.1'
=> ufw allow out 123/udp comment 'NTP'
=> ufw allow out 587/tcp comment 'Envoi de mails via WORDPRESS'
=> ufw allow out to 192.168.7.145 port 29590 proto tcp comment 'Requetes
vers BDD'
=> ufw enable

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 web /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 :

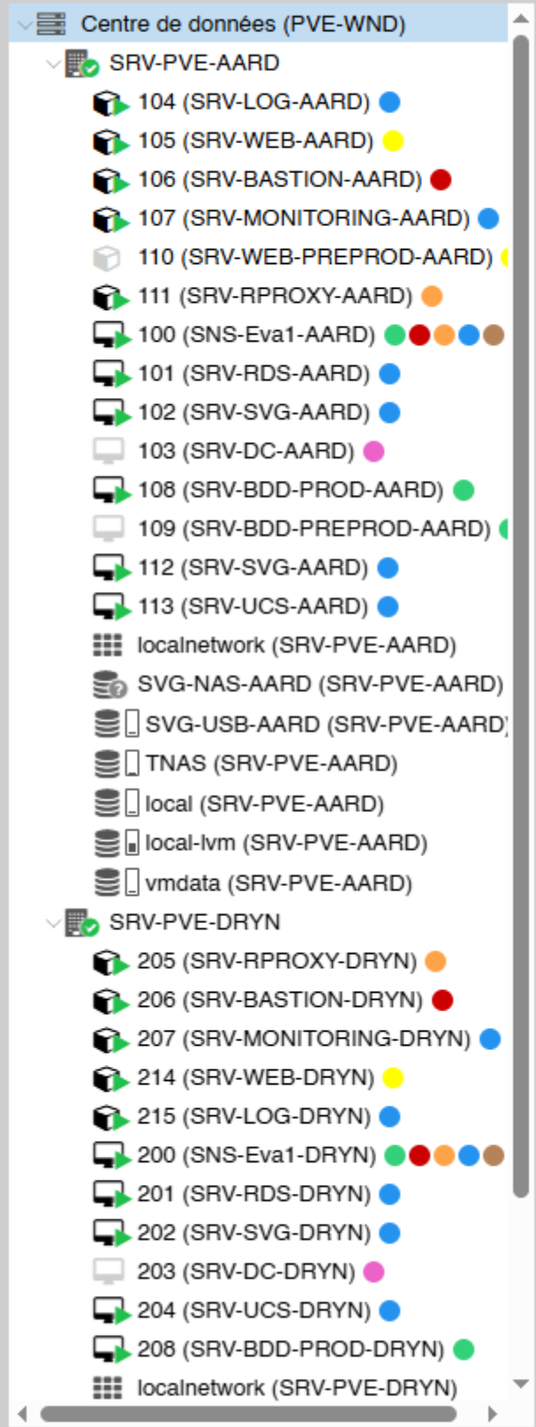
```
=> cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
=> nano /etc/fail2ban/jail.local
```

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

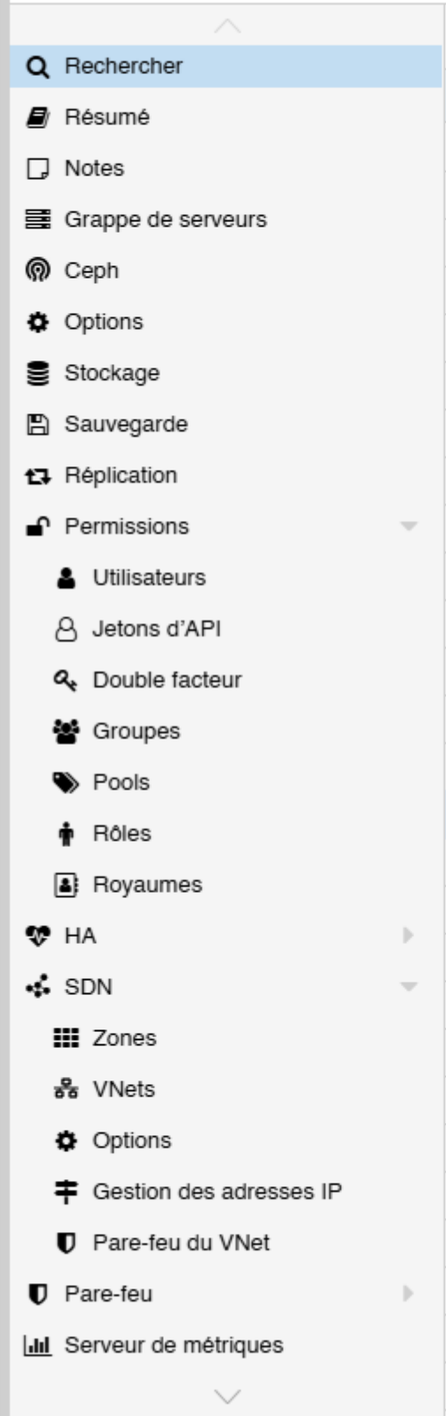
```
=> systemctl restart fail2ban
=> fail2ban-client status
=> fail2ban-client status sshd
```

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

Vue serveur



Centre de données



Type	Description	Utilisation …	Utilisation …	Utilisation …	Durée de fon…	Utilisation …	Utilisation …
lxc	104 (SRV-LOG-AARD)	20.5 %	56.6 %	0.6% of 1 …	4 jours 02:54…	0.2% of 4 …	0.9 %
lxc	105 (SRV-WEB-AARD)	19.6 %	24.1 %	0.2% of 1 …	2 jours 04:19…	0.1% of 4 …	0.8 %
lxc	106 (SRV-BASTION-AARD)	89.2 %	33.3 %	0.6% of 2 …	3 jours 20:55…	0.3% of 4 …	2.1 %
lxc	107 (SRV-MONITORING-AA…	39.5 %	16.4 %	1.7% of 1 …	4 jours 02:52…	0.4% of 4 …	1.1 %
lxc	110 (SRV-WEB-PREPROD-…				-		
lxc	111 (SRV-RPROXY-AARD)	59.2 %	25.9 %	0.4% of 1 …	4 jours 02:51…	0.1% of 4 …	0.4 %
lxc	205 (SRV-RPROXY-DRYN)	45.6 %	27.5 %	0.5% of 1 …	4 jours 02:43…	0.1% of 4 …	0.4 %
lxc	206 (SRV-BASTION-DRYN)	88.5 %	29.7 %	0.2% of 2 …	4 jours 02:43…	0.1% of 4 …	1.9 %
lxc	207 (SRV-MONITORING-DR…	37.9 %	14.6 %	1.8% of 1 …	4 jours 02:42…	0.4% of 4 …	0.9 %
lxc	214 (SRV-WEB-DRYN)	19.5 %	8.7 %	0.2% of 1 …	2 jours 04:18…	0.0% of 4 …	0.3 %
lxc	215 (SRV-LOG-DRYN)	33.6 %	25.0 %	0.6% of 1 …	4 jours 02:42…	0.1% of 4 …	0.4 %
node	SRV-PVE-AARD	6.6 %	54.7 %	11.7% of 4 …	4 jours 02:56…		
node	SRV-PVE-DRYN	0.7 %	50.5 %	8.5% of 4 …	4 jours 02:47…		
qemu	100 (SNS-Eva1-AARD)	0.0 %	71.9 %	19.9% of 1 …	4 jours 02:55…	5.0% of 4 …	4.6 %
qemu	101 (SRV-RDS-AARD)	0.0 %	23.9 %	0.9% of 4 …	4 jours 02:55…	0.9% of 4 …	6.1 %
qemu	102 (SRV-SVG-AARD)	0.0 %	19.4 %	2.2% of 2 …	4 jours 02:55…	1.1% of 4 …	2.5 %
qemu	103 (SRV-DC-AARD)				-		
qemu	108 (SRV-BDD-PROD-AARD)	0.0 %	51.1 %	1.4% of 2 …	2 jours 04:25…	0.7% of 4 …	3.3 %
qemu	109 (SRV-BDD-PREPROD-…				-		
qemu	112 (SRV-SVG-AARD)	0.0 %	41.6 %	4.5% of 2 …	4 jours 02:51…	2.2% of 4 …	5.4 %
qemu	113 (SRV-UCS-AARD)	0.0 %	78.0 %	2.7% of 2 …	3 jours 20:57…	1.3% of 4 …	10.0 %
qemu	200 (SNS-Eva1-DRYN)	0.0 %	48.7 %	27.1% of 1 …	4 jours 02:46…	6.8% of 4 …	3.1 %
qemu	201 (SRV-RDS-DRYN)	0.0 %	9.4 %	0.6% of 2 …	4 jours 02:46…	0.3% of 4 …	2.4 %
qemu	202 (SRV-SVG-DRYN)	0.0 %	92.1 %	2.4% of 2 …	4 jours 02:43…	1.2% of 4 …	11.9 %
qemu	203 (SRV-DC-DRYN)				-		
qemu	204 (SRV-UCS-DRYN)	0.0 %	85.1 %	0.8% of 2 …	3 jours 20:57…	0.4% of 4 …	5.5 %
qemu	208 (SRV-BDD-PROD-DRYN)	0.0 %	45.5 %	1.0% of 2 …	2 jours 04:19…	0.5% of 4 …	2.9 %

Search or jump to... ctrl+k

Arthur YANG
arthury

Profile
Notification history
Change password
Sign out

Dashboards

Create and manage dashboards to visualize your data

Search for dashboards and folders

Filter by tag | Starred | Sort

Name	Tags
Filtres AARD	
MONITORING SUR LES LOGS	
MONITORING SUR LES METRIQUES	
Règles sur les métriques	
APT Monitoring	
SNS-Eva1-AARD	FW, Prometheus
SRV-BASTION-AARD	linux
SRV-BDD-PREPROD-AARD	linux
SRV-BDD-PROD-AARD	linux
SRV-LOG-AARD	linux
SRV-MONITORING-AARD	linux
SRV-RDS-AARD	linux
SRV-RPROXY-AARD	linux
SRV-RPROXY-AARD-METRIQUES-2	Prometheus, StarsL.cn, node_exporter

Datasource Prometheus | Job NODE-SRV-LOG-AARD | Host 192.168.7.133:59051 | GitHub | Grafana | Last 24 hours | Cancel | 5s

Quick CPU / Mem / Disk

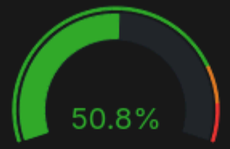
Pressure

CPU	24.0%
Mem	0.0%
I/O	0.4%

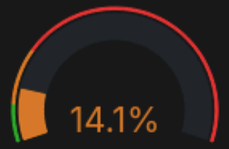
CPU Busy

Sys Load

RAM Used



SWAP Used

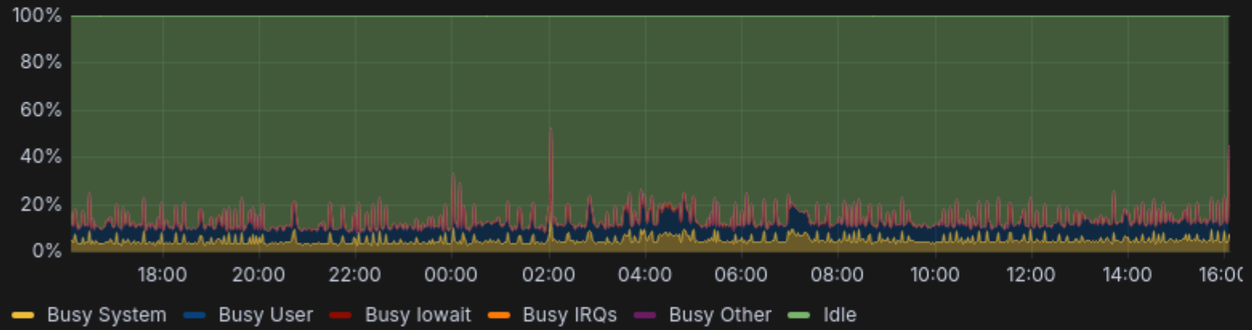


Root FS Used

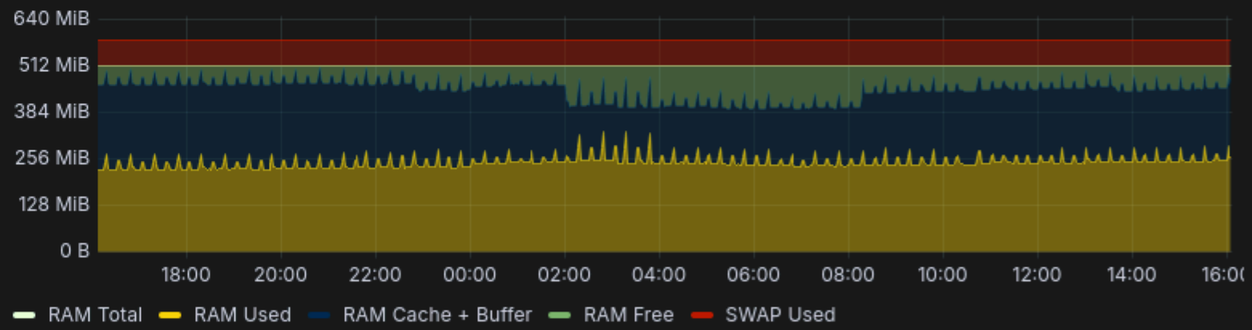
CPU ...	1
Uptime	4.1 days
Root...	16 GiB
RAM ...	512 MiB
SWA...	512 MiB

Basic CPU / Mem / Net / Disk

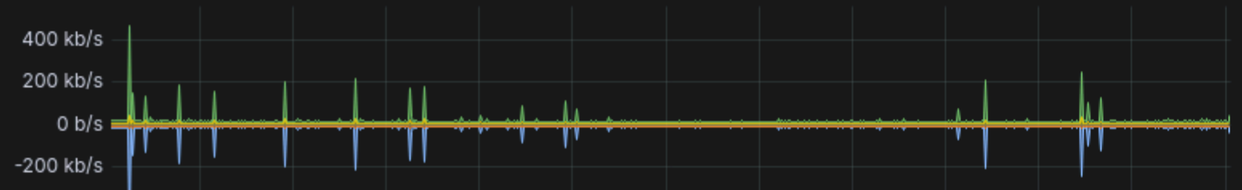
CPU Basic



Memory Basic



Network Traffic Basic



Disk Space Used Basic