BTS Services informatiques aux organisations - SISR
Session 2026

E4 – Support et mise à disposition de services informatiques
Coefficient 4

DESCRIPTION DE LA REALISATION PROFESSIONNELLE

NOM et prénom du candidat : Liana MEGAZZINI

Contexte de la réalisation professionnelle

Dans le cadre de la réalisation de mon portfolio pour l'épreuve du BTS SIO, j'ai été amenée à mettre en place une solution de supervision au sein de mon infrastructure de maquettage virtualisée.

Afin de garantir la disponibilité et les performances des services de l'infrastructure, il était nécessaire de mettre en place un outil de supervision centralisé, capable de collecter et visualiser des métriques en temps réel.

Le projet consistait à déployer une solution de supervision basée sur Grafana et Prometheus, en instrumentant les différents serveurs de l'infrastructure (Linux et Windows) avec des exporters dédiés.

Plusieurs contraintes ont été identifiées :
- Assurer la compatibilité entre les versions des exporters et de Prometheus
- Gérer les règles de pare-feu pour autoriser le scraping Prometheus
- Configurer correctement les datasources et dashboards Grafana
- Adapter les requêtes PromQL aux métriques disponibles selon les versions d'exporters

L'objectif était de mettre en place une solution complète de supervision, fonctionnelle et conforme à une architecture professionnelle.

Intitulé de la réalisation professionnelle :
Mise en place d'une solution de supervision avec Grafana et Prometheus

Période de réalisation : Lieu :
Modalité : ✓ Individuelle □ En équipe

Principale(s) activité(s) concernée(s) :
- **Mettre à disposition des utilisateurs un service informatique :**
  - Réaliser les tests d'intégration et d'acceptation d'un service
  - Déployer un service
- **Gérer le patrimoine informatique :**
  - Recenser et identifier les ressources numériques
  - Déployer un service
  - Vérifier les conditions de la continuité d'un service informatique
  - Vérifier le respect des règles d'utilisation des ressources numériques

Conditions de réalisation :
- **Ressources présentes :**
  - Infrastructure virtualisée sous ESXi
  - VM Grafana (Debian 12)
  - VM DNS-Bind (Debian 12)
  - VM WordPress (Debian 12)
  - Serveur Veeam (Windows Server)
  - Réseau interne segmentté en VLANs
- **Résultats attendus :**
  - Mise en place d'une solution de supervision fonctionnelle
  - Collecte des métriques système (CPU, RAM, réseau, services)

- Visualisation centralisée via des dashboards Grafana
- Supervision des VMs Linux et Windows

- **Durée de réalisation**
  - 30 Mars 2026 – 31 Mars 2026

Modalités d'accès à cette réalisation professionnelle.
URL : https://pllm.fr Mot de passe : sioBTSsisr26

Partie 1 – Procédure de mise en œuvre.

Outils et technologies utilisés :

- Grafana
- Prometheus
- Node Exporter (Linux)
- Windows Exporter (Windows)
- Debian 12
- UFW (pare-feu Linux)

Organisation du projet (travail autonome) :

- Travail réalisé en autonomie dans un environnement de maquettage
- Recherches de documentations techniques (Grafana, Prometheus, exporters)
- Tests et validation progressive des configurations

Phases de mise en œuvre :

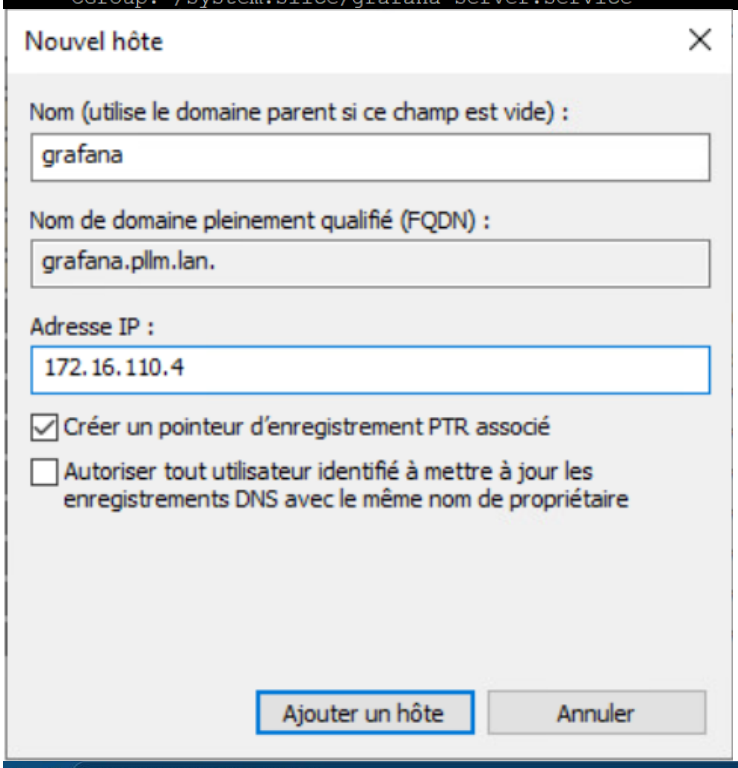
1. Installation et configuration de Grafana

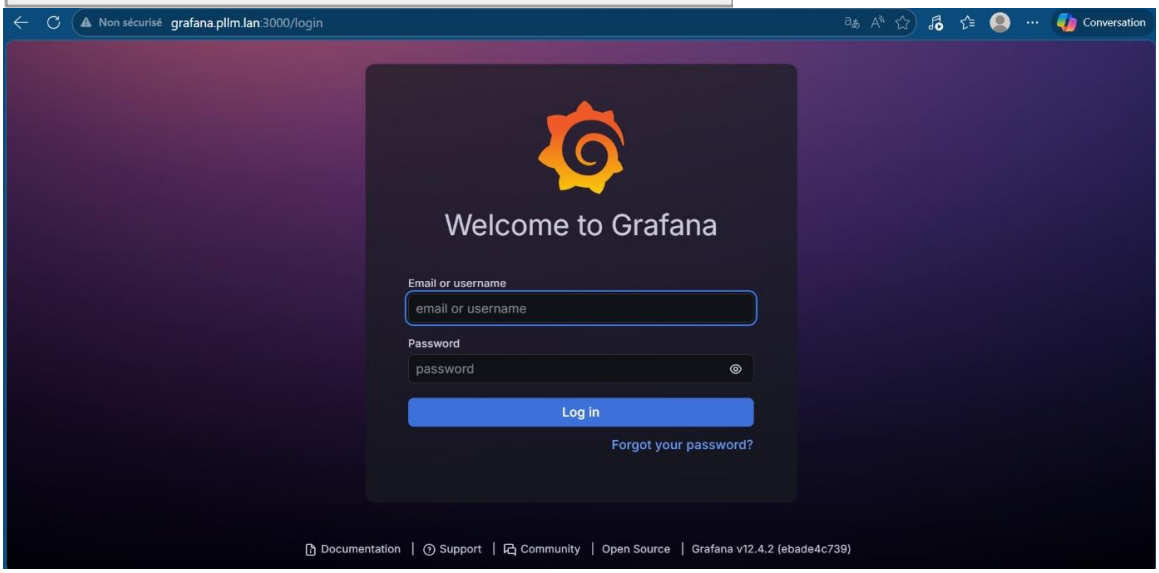
J'ai commencé par installer Grafana sur une VM Debian 12 dédiée à la supervision. Pour garantir la fraicheur des paquets, j'ai ajouté le dépôt officiel de Grafana plutôt qu'un paquet .deb manuel :

```
  GNU nano 7.2
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug ens192
iface ens192 inet static
        address 172.16.110.4/24
        gateway 172.16.110.1
        dns-nameservers 172.16.110.2 172.16.130.3
        dns-domain pllm.lan
```

```
sudo mkdir -p /etc/apt/keyrings
wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee
/etc/apt/keyrings/grafana.gpg > /dev/null
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com stable
main" | sudo tee /etc/apt/sources.list.d/grafana.list
sudo apt update && sudo apt install -y grafana
sudo systemctl enable --now grafana-server
```

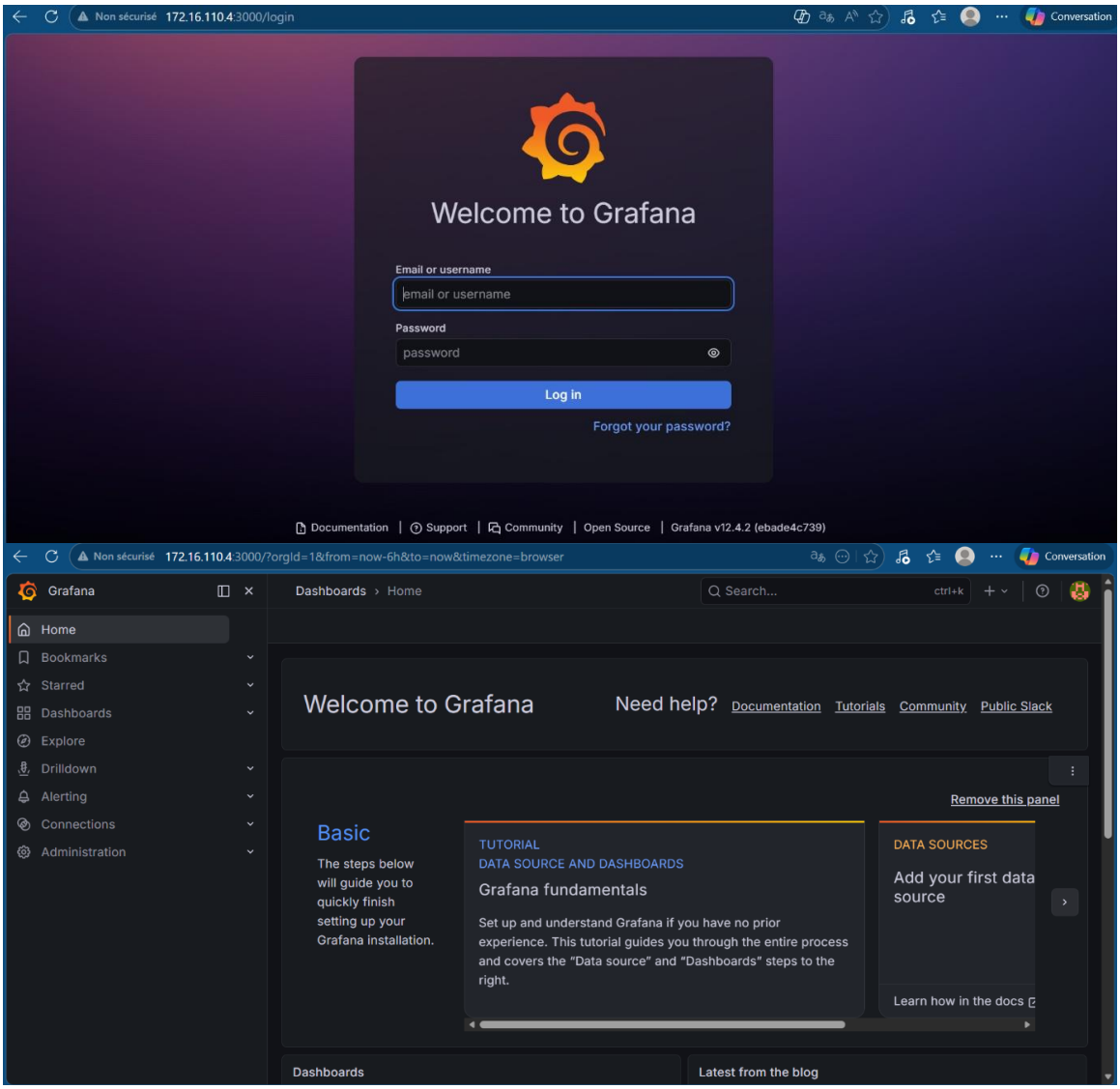
```
root@grafana:~# grafana-server -v
Version 12.4.2 (commit: ebade4c739e1aface4ce094934ad85374887a680, branch: releas
e-12.4.2)
```





Grafana est ensuite accessible via http://grafana.pllm.lan:3000 (identifiants par défaut : admin/admin). Lors de la première connexion, j'ai rencontré une difficulté : Grafana refusait le mot de passe admin/admin. Le problème venait du fait que la base SQLite avait été initialisée lors d'un premier démarrage avec un mot de passe différent. La solution a été de réinitialiser le mot de passe via la CLI :

```
sudo grafana-cli admin reset-admin-password admin
```



2. Installation et configuration de Prometheus

Grafana seul ne collecte pas de métriques, il a besoin d'une datasource. J'ai installé Prometheus sur la même VM en téléchargeant le binaire officiel depuis GitHub et en créant un service systemd dédié :

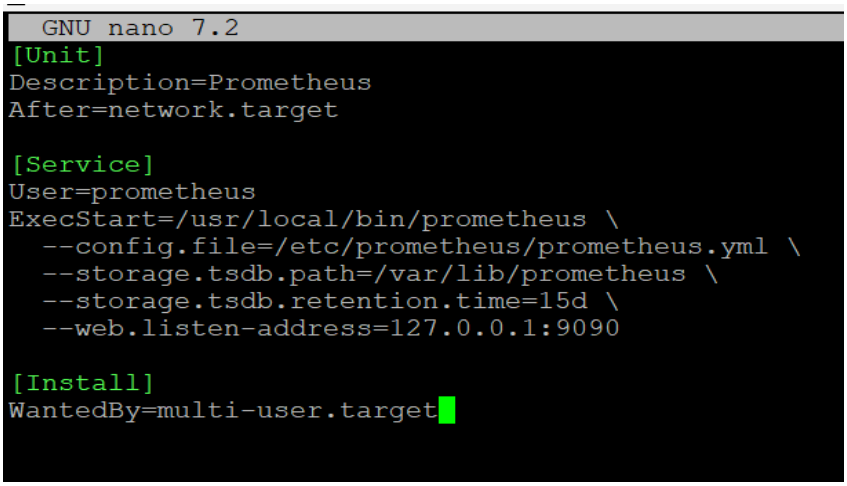
```
sudo useradd --no-create-home --shell /bin/false prometheus
sudo mkdir /etc/prometheus /var/lib/prometheus
sudo chown -R prometheus:prometheus /etc/prometheus /var/lib/prometheus
```

Le fichier de configuration /etc/prometheus/prometheus.yml définit les cibles à superviser (scrape_configs). Prometheus a été configuré pour écouter uniquement en local (127.0.0.1:9090) afin de ne pas exposer son interface sur le réseau, Grafana étant sur la même machine.

```
root@grafana:/tmp# wget -qO- https://api.github.com/repos/prometheus/prometheus/releases/latest | grep browser_download_url
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.darwin-amd64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.darwin-arm64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.dragonfly-amd64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.freebsd-386.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.freebsd-amd64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.freebsd-arm64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.freebsd-armv6.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.freebsd-armv7.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.illumos-amd64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-386.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-amd64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-arm64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-armv5.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-armv6.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-armv7.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-mips.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-mips64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-mips64le.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-mipsle.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-ppc64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-ppc64le.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-riscv64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.linux-s390x.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.netbsd-386.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.netbsd-amd64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.netbsd-arm64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.netbsd-armv6.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.netbsd-armv7.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.windows-386.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.windows-386.zip"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.windows-amd64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.windows-amd64.zip"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.windows-arm64.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-3.10.0.windows-arm64.zip"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/prometheus-web-ui-3.10.0.tar.gz"
    "browser_download_url": "https://github.com/prometheus/prometheus/releases/download/v3.10.0/sha256sums.txt"
```

```
  GNU nano 7.2
global:
  scrape_interval: 15s
  evaluation_interval: 15s

scrape_configs:
  - job_name: "supervision"
    static_configs:
      - targets: ["localhost:9100"]

  - job_name: "dns-bind"
    static_configs:
      - targets: ["172.16.130.3:9100"]

  - job_name: "wordpress"
    static_configs:
      - targets: ["172.16.130.2:9100"]
      - targets: ["172.16.130.2:9104"]

  - job_name: "windows"
    static_configs:
      - targets: ["172.16.110.2:9182"]
```

```
root@grafana:/tmp/prometheus-3.10.0.linux-amd64# promtool check config /etc/prometheus/prometheus.yml
Checking /etc/prometheus/prometheus.yml
 SUCCESS: /etc/prometheus/prometheus.yml is valid prometheus config file syntax
```



3. Supervision de la VM Windows (Serveur Veeam)

Pour superviser le serveur Windows Veeam (IP : 172.16.140.2), j'ai installé windows_exporter v0.31.5 via un fichier .msi téléchargé depuis le dépôt officiel Prometheus Community. L'installeur crée automatiquement un service Windows qui écoute sur le port 9182.

J'ai ensuite créé une règle de pare-feu Windows pour autoriser uniquement la VM Grafana (172.16.110.4) à accéder au port 9182 :

```
New-NetFirewallRule -DisplayName "Prometheus windows_exporter" -Direction Inbound
-Protocol TCP -LocalPort 9182 -RemoteAddress 172.16.110.4 -Action Allow
```

Plusieurs problèmes ont été rencontrés et résolus lors de cette étape :

- Erreur de timeout Prometheus (context deadline exceeded) : causée par une mauvaise IP renseignée dans prometheus.yml (172.16.110.2 au lieu de 172.16.140.2). Corrigé en mettant à jour l'IP et en rechargant Prometheus via kill -HUP.
- Certains collecteurs windows_exporter (logical_disk) retournaient une valeur 0 (collecteur en échec). Les métriques disponibles ont été vérifiées via curl et les dashboards adaptés en conséquence.

4. Supervision des VMs Linux (DNS Bind)

Pour les VMs Linux, j'ai installé node_exporter sur la VM DNS Bind (IP : 172.16.130.3). L'installation suit le même principe : téléchargement du binaire, création d'un utilisateur dédié et mise en place d'un service systemd :

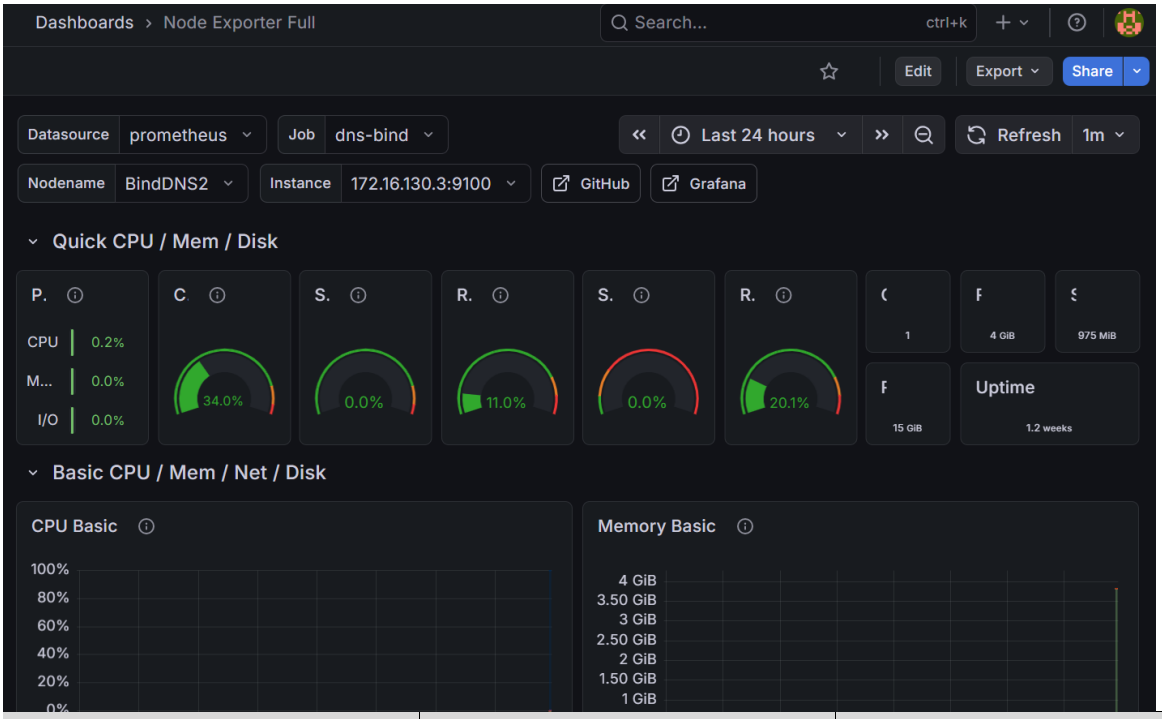
```
sudo useradd --no-create-home --shell /bin/false node_exporter
sudo cp node_exporter-1.8.2.linux-amd64/node_exporter /usr/local/bin/
sudo systemctl enable --now node_exporter
```

Le pare-feu UFW a été configuré pour n'autoriser que la VM supervision à scraper le port 9100 :

```
sudo ufw allow from 172.16.110.4 to any port 9100
```

5. Configuration des datasources et dashboards Grafana

Une fois Prometheus opérationnel, j'ai configuré la datasource dans Grafana (Connections → Data sources → Add new data source → Prometheus, URL : http://localhost:9090). J'ai ensuite importé des dashboards communautaires via leurs IDs :



Cible	Dashboard	ID
Linux (node_exporter)	Node Exporter Full	1860
Windows (windows_exporter)	Windows Exporter	14694

Plusieurs difficultés ont été rencontrées lors de la configuration des dashboards :

- Erreur datasource not found : la variable ${DS_PROMETHEUS} dans le JSON du dashboard ne se résolvait pas correctement. Résolu en remplaçant manuellement la variable par l'UID réel de la datasource dans le JSON du dashboard.
- Variable $server vide : la datasource de la variable était configurée sur la base Grafana interne au lieu de Prometheus. Corrigé en sélectionnant Prometheus comme datasource de la variable et en configurant le type Label values sur le label instance.
- Panels mémoire en N/A : le dashboard utilisait les métriques windows_cs_* et windows_os_* qui ne sont plus disponibles dans les versions récentes de windows_exporter. Les requêtes PromQL ont été adaptées pour utiliser windows_memory_physical_total_bytes et windows_memory_available_bytes.

6. Architecture

Partie 2 – Validation.

- Vérification que Prometheus scrape correctement toutes les targets :
  `curl "http://localhost:9090/api/v1/query?query=up"`
  → Toutes les targets retournent une value à 1 (UP)

- Vérification des métriques disponibles sur chaque exporter :
  `curl http://172.16.140.2:9182/metrics | head -20`
  → Les métriques windows_cpu, windows_memory, windows_service remontent correctement

- Vérification de l'affichage dans Grafana :
  → Dashboard Windows : panels CPU Load, CPU Usage et Services by state fonctionnels
  → Dashboard Linux : métriques système (CPU, RAM, réseau) visibles pour la VM DNS Bind

Ces tests confirment que la solution de supervision est fonctionnelle et conforme aux objectifs.

Partie 3 – Veille technologique.

D'autres solutions auraient pu être envisagées :
- Zabbix : solution de supervision tout-en-un, plus simple à déployer mais moins flexible pour la visualisation
- Nagios : solution historique, robuste mais interface vieillissante et configuration complexe
- Datadog : solution cloud SaaS, très complète mais payante et dépendante d'un tiers
- Stack ELK (Elasticsearch, Logstash, Kibana) : orientée logs plutôt que métriques

Ces solutions ont été écartées en raison :
- de la volonté d'utiliser une stack open-source largement adoptée en entreprise
- de la flexibilité de Grafana pour la création de dashboards personnalisés
- de la puissance de Prometheus pour la collecte et le stockage de métriques temporelles
- de la grande communauté et des nombreux exporters disponibles pour tous types de services

Le choix de Grafana et Prometheus permet de reproduire une architecture de supervision réaliste utilisée en entreprise, tout en restant complètement maîtrisable et gratuite.