

LOGBOOK - Architecture de supervision et conteneurisation pour une application BI

Conception et déploiement d'une infrastructure sécurisée sous Ubuntu Server.

Orchestration d'une pile logicielle (Apache, MariaDB) via Docker Compose et mise en place d'un monitoring complet (Prometheus, Grafana, Node Exporter).

Sécurisation des flux via SSL/TLS et gestion de l'interopérabilité réseau Windows/Linux.

1 - Infrastructure & réseau

Mise en place d'un environnement serveur sous Ubuntu Server 24.04 LTS sur VirtualBox ;

Paramétrage d'une interface en mode "Accès par pont" (Bridge) pour permettre la communication bidirectionnelle entre l'hôte Windows et le serveur Linux ;

Configuration d'une adresse IP statique (Netplan) pour garantir la persistance des services.

2 - Services natifs & sécurité

Installation et configuration initiale d'Apache2 et MariaDB en local ;

Mise en place d'un pare-feu (UFW) restrictif et configuration du protocole HTTPS via la génération de certificats SSL (OpenSSL) pour chiffrer les échanges ;

Configuration de Samba pour le transfert de fichiers sécurisé entre les environnements Windows et Linux.

3 - Modernisation via la conteneurisation

Migration des services applicatifs vers Docker pour assurer l'isolation et la portabilité ;

Création d'un fichier `docker-compose.yml` pour automatiser le déploiement simultané du serveur web, de la base de données et des outils de monitoring ;

Utilisation de volumes Docker pour séparer les données critiques (sites web, bases SQL) du cycle de vie des conteneurs

4 - Supervision & observabilité (LGTM)

Déploiement de **Prometheus** pour l'agrégation des métriques et de **Node Exporter** pour la surveillance en temps réel des ressources matérielles du serveur (CPU, RAM, Disque) ;

Installation de **Grafana** pour la visualisation et création de tableaux de bord dynamiques pour piloter l'infrastructure graphiquement depuis un poste distant.

5.1 - Sécurisation avancée & segmentation réseau

Cloisonnement réseau (Network Segmentation) → création de réseaux Docker isolés (frontend et backend) pour limiter la surface d'attaque ;

Configuration de la DB MariaDB et du moteur Prometheus en mode "privé", les rendant isolés et inaccessibles depuis l'extérieur du serveur tout en maintenant la communication inter-conteneurs ;

Suppression des mappages de ports non essentiels sur l'hôte, garantissant que seuls les ports d'entrée (8080, 3000) sont exposés.

5.2 - Centralisation des flux & Reverse Proxy (Nginx)

Déploiement d'un conteneur Nginx agissant comme Reverse Proxy pour centraliser tous les flux sur le port standard 80 ;

Configuration de Nginx pour rediriger le trafic dynamiquement : Le trafic racine (/) vers le serveur Apache & le trafic de supervision (/monitoring/) vers l'interface Grafana.

Ajustement des variables d'environnement Grafana (`ROOT_URL`) pour assurer la cohérence des liens internes lors de l'exposition derrière un sous-répertoire.

6 - Validation & Tests de conformité

Utilisation d'outils de diagnostic réseau (curl, ss) à l'intérieur des conteneurs pour valider le flux de données entre le serveur Web et la DB ;

Simulation d'un scan de ports depuis le poste client (Windows) via Test-NetConnection pour confirmer l'étanchéité de la base de données face au réseau local ;

Test de la chaîne de métriques complète, de l'extraction par Node Exporter à la visualisation sur Grafana, validant la fiabilité des indicateurs de performance.

The image shows two browser screenshots. The top one is at 'my-app-bi.local/' displaying 'welcome to my BI app' with server status 'GOOD' and database 'decision_db'. The bottom one is at '192.168.1.100:8080' displaying 'App BI working on Docker !' and 'i'm working'.

```
project_bt docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
0cecf1e0c93e   grafana/grafana:latest              "/run.sh"               31 minutes ago Up 31 minutes 0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp   grafana_container
85bcb6d8605e   mariadb:latest                       "docker-entrypoint.s..." 31 minutes ago Up 31 minutes 0.0.0.0:3306->3306/tcp, [::]:3306->3306/tcp   mariadb_container
ac10ed76df67   httpd:latest                         "httpd-foreground"      31 minutes ago Up 31 minutes 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp       apache_container
07283d97a4aa   prom/prometheus:latest              "/bin/prometheus --c..." 31 minutes ago Up 31 minutes 0.0.0.0:9090->9090/tcp, [::]:9090->9090/tcp   prometheus_container
27c398dac03a   prom/node-exporter:latest           "/bin/node_exporter ..." 31 minutes ago Up 31 minutes 0.0.0.0:9100->9100/tcp, [::]:9100->9100/tcp   node_exporter_container
```

```
nano docker-compose.yml
docker-compose.yml
version: '3.6'

services:
  db-bit:
    image: mariadb:latest
    container_name: mariadb_container
    environment:
      MYSQL_ROOT_PASSWORD: adminantoine
      MYSQL_DATABASE: decivision_db
    ports:
      - "3306:3306"

  # Service Web
  web-bit:
    image: httpd:latest
    container_name: apache_container
    ports:
      - "8080:80"
    volumes:
      - ./website:/usr/local/apache2/htdocs/

  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus_container
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    ports:
      - "9090:9090"

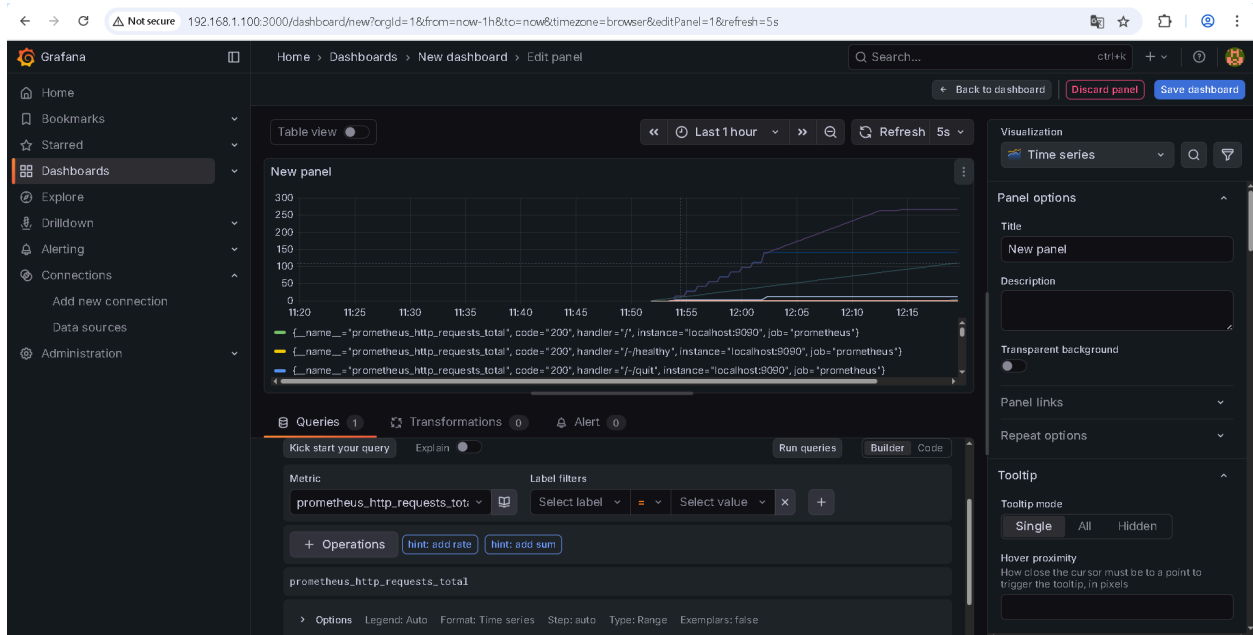
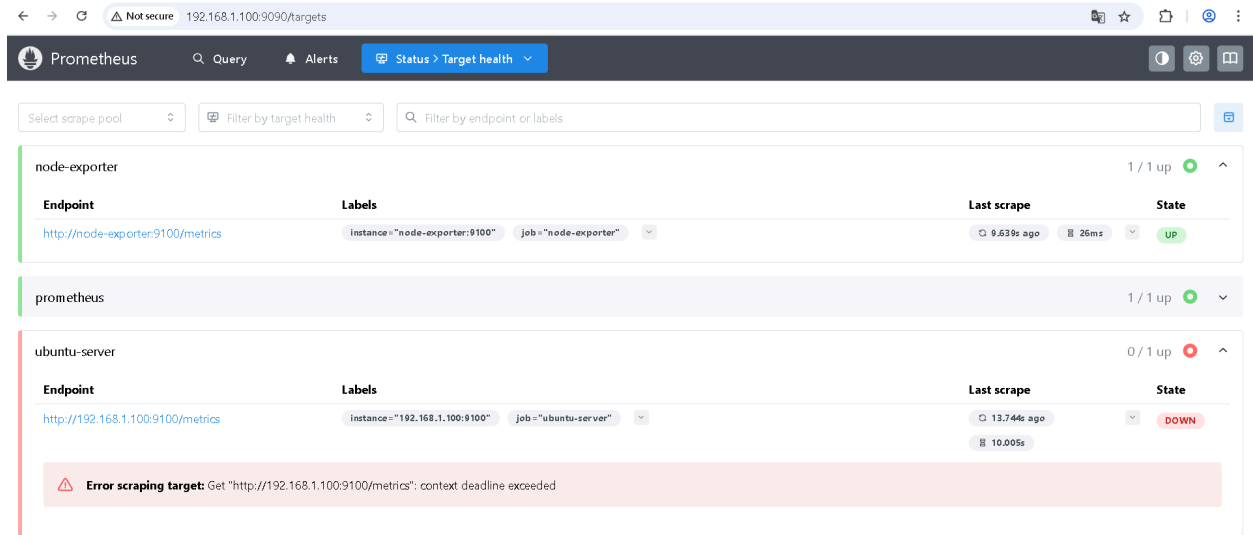
  grafana:
    image: grafana/grafana:latest
    container_name: grafana_container
    ports:
      - "3000:3000"
    environment:
      GF_SECURITY_ADMIN_PASSWORD=admingrafana
    depends_on:
      - prometheus

  node-exporter:
    image: prom/node-exporter:latest
    container_name: node_exporter_container
    restart: unless-stopped
    volumes:
      - /proc:/host/proc:ro
      - /sys:/host/sys:ro
      - /:/rootfs:ro
    command:
      - --path.procfs=/host/proc
      - --path.sysfs=/host/sys
      - --collector.filesystem.ignored-mount-points='^/(sys|proc|dev|host|etc)($|/)'
```

```
nano prometheus.yml
prometheus.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'ubuntu-server'
    static_configs:
      - targets: ['192.168.1.100:9100'] # On surveillera l'hôte via Node Exporter plus tard
  - job_name: 'node-exporter'
    static_configs:
      - targets: ['node-exporter:9100']
```




```

networks:
  frontend:
  backend:

services:
  db-bi:
    image: mariadb:latest
    container_name: mariadb_container
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: adminantoine
      MYSQL_DATABASE: decivision_db
    networks:
      - backend

  web-bi:
    image: httpd:latest
    container_name: apache_container
    restart: always
    ports:
      - "8080:80"
    volumes:
      - ./website:/usr/local/apache2/htdocs/
    networks:
      - frontend
      - backend

# PROMETHEUS (ISOLÉ)
prometheus:
  image: prom/prometheus:latest
  container_name: prometheus_container
  restart: always
  volumes:
    - ./prometheus.yml:/etc/prometheus/prometheus.yml
  networks:
    - backend

grafana:
  image: grafana/grafana:latest
  container_name: grafana_container
  restart: always
  ports:
    - "3000:3000"
  environment:
    - GF_SECURITY_ADMIN_PASSWORD=admingrafana
  depends_on:
    - prometheus
  networks:
    - frontend
    - backend

node-exporter:
  image: prom/node-exporter:latest
  container_name: node_exporter_container
  restart: unless-stopped
  volumes:
    - /proc:/host/proc:ro
    - /sys:/host/sys:ro
    - /:/rootfs:ro
  command:
    - '--path.procfs=/host/proc'
    - '--path.sysfs=/host/sys'
    - '--collector.filesystem.ignored-mount-points=^/(sys|proc|dev|host|etc)($$|/)'
  networks:
    - backend

```

Administrator: Windows PowerShell

```
PS C:\WINDOWS\system32> Test-NetConnection -ComputerName 192.168.1.100 -Port 3306
WARNING: TCP connect to (192.168.1.100 : 3306) failed
```

```
ComputerName      : 192.168.1.100
RemoteAddress     : 192.168.1.100
RemotePort        : 3306
InterfaceAlias    : Wi-Fi
SourceAddress     : 192.168.1.125
PingSucceeded     : True
PingReplyDetails (RTT) : 94 ms
TcpTestSucceeded  : False
```

```
→ project_bi docker-compose ps
      Name                    Command                                State      Ports
-----
apache_container            httpd-foreground                      Up         0.0.0.0:8080->80/tcp,:::8080->80/tcp
grafana_container          /run.sh                              Up         0.0.0.0:3000->3000/tcp,:::3000->3000/tcp
mariadb_container         docker-entrypoint.sh mariadb         Up         3306/tcp
node_exporter_container    /bin/node_exporter --path. ...       Up         9100/tcp
prometheus_container       /bin/prometheus --config.f ...       Up         9090/tcp
```

```
→ project_bi nano docker-compose.yml
→ project_bi
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT   MEM %     NET I/O       BLOCK I/O   PIDS
bfc22e8c6c2f  grafana_container      0.80%    88.2MiB / 1.906GiB  4.52%    42.8MB / 16.7MB 12.1MB / 87MB 13
55d6c36f7926  mariadb_container     0.02%   102.4MiB / 1.906GiB  5.25%    12.4kB / 613B   25MB / 24.2MB 8
0780faa8270a  node_exporter_container 0.00%    8.332MiB / 1.906GiB  0.43%    139kB / 2.36MB  0B / 0B       5
fda637742ca3  apache_container      0.00%    41.74MiB / 1.906GiB  2.14%    21.3MB / 422kB  35.9MB / 36.7MB 82
d162ea51ff25  prometheus_container   0.23%    76.22MiB / 1.906GiB  3.91%    2.44MB / 326kB  49.8MB / 2.94MB 8
```