Mise en place cadvisor grafana promotheus docker

Aujourdhui nous allons voir comment monitorer nos différents docker ce qui est très important en prod

- **cAdvisor (conteneurs Docker)**
  - Rôle : expose CPU, RAM, disque, réseau pour chaque conteneur Docker (nginx, wordpress, db01, db02, db03…).
  - Utile pour : voir quel conteneur sature, quels sont les pics de charge, repérer un conteneur qui fuit en mémoire.

Prometheus permettra de lire les stats et Grafana servira juste à les afficher via une interface web

Il faut rajouter cela dans notre compose.yaml

```yaml
cadvisor:
  image: gcr.io/cadvisor/cadvisor:latest
  container_name: cadvisor
  ports:
    - "8080:8080"
  volumes:
    - /:/rootfs:ro
    - /var/run:/var/run:rw
    - /sys:/sys:ro
    - /var/lib/docker/:/var/lib/docker:ro
  restart: unless-stopped

prometheus:
  image: prom/prometheus:latest
  container_name: prometheus
  volumes:
    - ./prometheus.yml:/etc/prometheus/prometheus.yml:ro
  ports:
    -"9090:9090"
  depends_on:
    - cadvisor
  restart: unless-stopped
grafana:
  image: grafana/grafana:latest
  container_name: grafana
  ports:
    - "3000:3000"
  depends_on:
    - prometheus
  restart: unless-stopped
```

Dans le même dossier, crée prometheus.yml

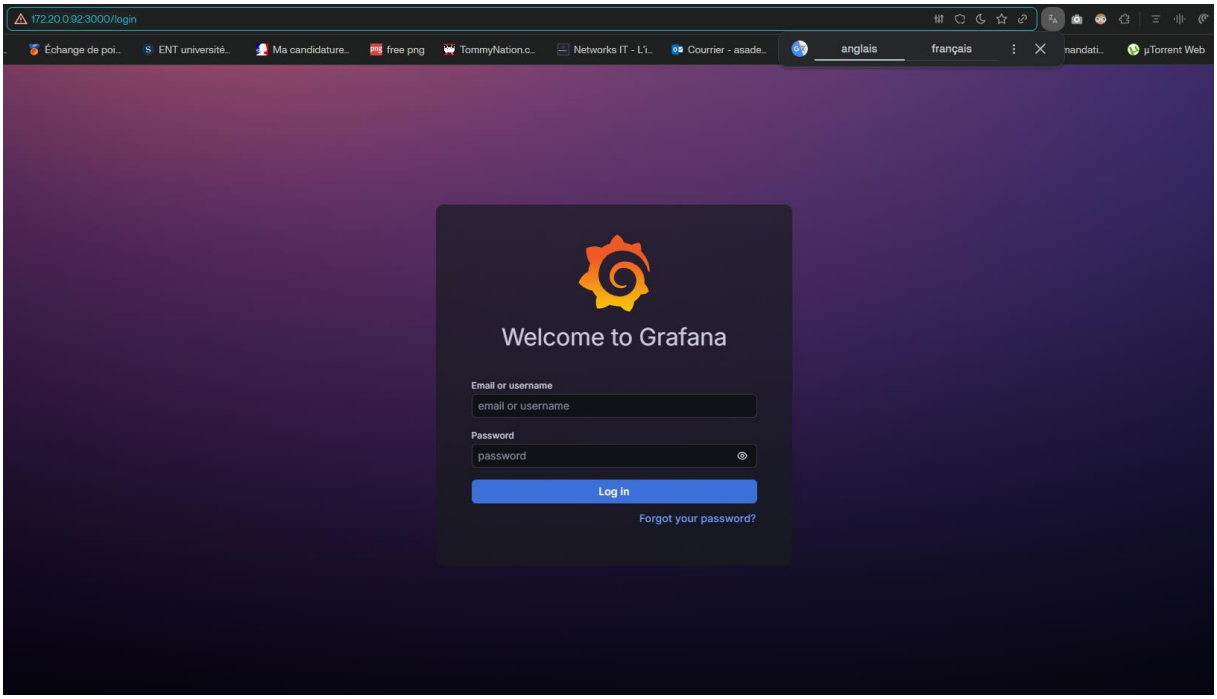
```
 GNU nano 8.4                                          prometheus.yml
global:
  scrape_interval: 10s

scrape_configs:
  - job_name: 'cadvisor'
    static_configs:
      - targets: ['cadvisor:8080']
```

Ici, Prometheus va juste scraper cAdvisor toutes les 10 secondes et récupérer les métriques de **tous** les conteneurs Docker du host

Ensuite on lance via docker compose up -d

Grafana

http://172.20.0.92:3000/login



Admin / admin

Une fois connecté à Grafana, tu dois juste 1) ajouter Prometheus comme datasource, 2) importer un dashboard cAdvisor.

1. Ajouter Prometheus comme datasource

Dans Grafana (UI récente) :

- Va dans **Connections → Data sources → Add data source**.
- Choisis **Prometheus**.
- Dans **URL**, mets http://prometheus:9090 (vu que Grafana et Prometheus sont dans le même compose).
- Laisse le reste par défaut, clique sur **Save & test** → tu dois voir un message « Data source is working ».

2. Importer un dashboard cAdvisor

- Dans le menu à gauche, clique sur **Dashboards → Import**.
- Dans le champ « Import via grafana.com ID », mets par exemple un ID de dashboard cAdvisor (par ex. un dashboard « cAdvisor exporter » de Grafana, type 14282 ou 19792).
- Clique sur **Load**.
- En bas, choisis la datasource **Prometheus** que tu viens de créer, puis **Import**.

Tu devrais voir directement les graphes pour tous tes conteneurs (CPU, mémoire, réseau, etc.).

Voici la doc :

Mise en place du monitoring Docker – Débogage et résolution

Contexte

Dans le cadre du projet d'infrastructure Docker (cluster Galera + WordPress + Nginx), nous avons mis en place une stack de monitoring pour visualiser les métriques de tous les conteneurs en temps réel.

Stack utilisée : Prometheus + Grafana + cAdvisor

Problèmes rencontrés et étapes de débogage

1. Erreur de parsing du fichier `prometheus.yml`

Problème : Au premier démarrage, Prometheus retournait l'erreur suivante dans ses logs :

```
text
parsing YAML file /etc/prometheus/prometheus.yml:
yaml: line 6: mapping values are not allowed in this context
```

Cause : Le fichier `prometheus.yml` contenait une erreur de syntaxe YAML : un espace parasite avant le `:` de `job_name` et une mauvaise indentation de `static_configs`.

Résolution : Correction du fichier avec une indentation stricte (espaces uniquement, pas de tabulations) :

```
text
global:
  scrape_interval: 10s

scrape_configs:
  - job_name: 'cadvisor'
    static_configs:
      - targets: ['cadvisor:8080']
```

---

2. Dashboard Grafana vide (No data)

Problème : Après import d'un dashboard cAdvisor dans Grafana, tous les panels affichaient « No data » malgré un Prometheus fonctionnel.

Diagnostic : Vérification en plusieurs étapes :

- `docker logs prometheus` → pas d'erreur, Prometheus démarre correctement.
- `curl http://127.0.0.1:8080/metrics` → cAdvisor répond bien et expose des métriques.
- UI Prometheus → **Status** → **Targets** : job `cadvisor` en état **UP**.
- Requête `container_cpu_usage_seconds_total` dans Prometheus → retourne bien des séries.

Cause identifiée : Les variables du dashboard Grafana (`$host`, `$container`) n'avaient pas de datasource configurée (champ `Select data source` vide), donc elles ne pouvaient pas résoudre leurs valeurs.

Résolution :

- Variable `host` : datasource → **Prometheus**, query → `label_values(container_cpu_usage_seconds_total, instance)`.
- Variable `container` : datasource → **Prometheus**, query → `label_values(container_memory_usage_bytes{name!=""}, name)`.

Le filtre `name!=""` est important : il exclut les cgroups système (`/system.slice/...`) et ne garde que les vrais conteneurs Docker nommés.

---

3. Requêtes PromQL des panels incompatibles

Problème : Même après correction des variables, certains panels restaient vides car les requêtes PromQL utilisaient le filtre `instance=~"$host"` alors que la variable `host` n'était pas affichée dans le dashboard.

Résolution : Suppression du filtre `$host` dans toutes les requêtes des panels, en ne gardant que le filtre `$container` :

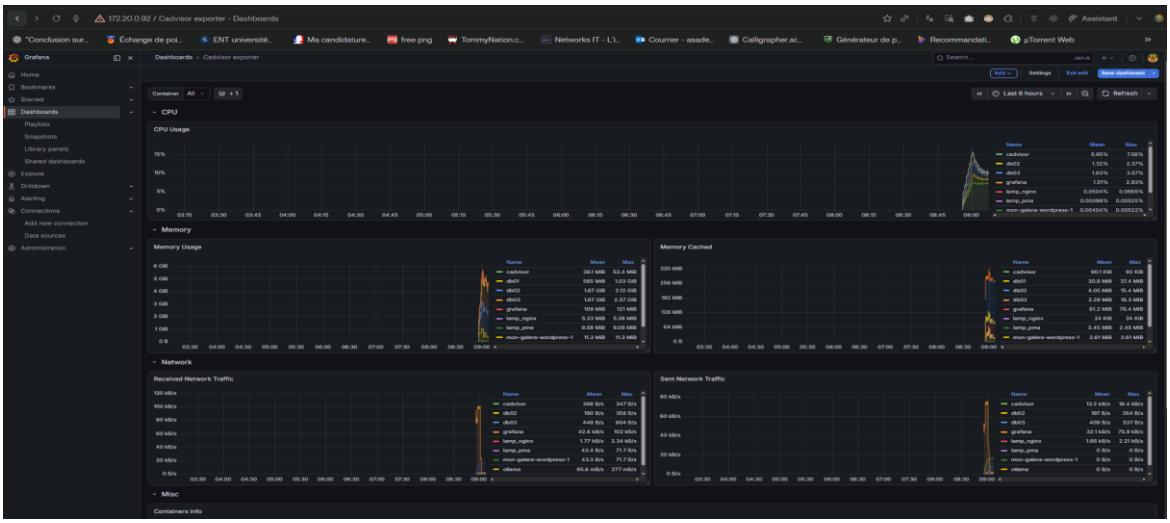
Panel	Requête corrigée
CPU Usage	`sum(rate(container_cpu_usage_seconds_total{name=~"$container", name=~".+"}[5m])) by (name) * 100`
Memory Usage	`container_memory_usage_bytes{name=~"$container", name=~".+"}`
Memory Cached	`container_memory_cache{name=~"$container", name=~".+"}`
Network RX	`sum(rate(container_network_receive_bytes_total{name=~"$container", name=~".+"}[5m])) by (name)`
Network TX	`sum(rate(container_network_transmit_bytes_total{name=~"$container", name=~".+"}[5m])) by (name)`

Résultat

La stack de monitoring est pleinement opérationnelle. Le dashboard Grafana affiche en temps réel :

- **CPU** : utilisation par conteneur avec moyenne et pic.
- **Mémoire** : RAM utilisée et cache par conteneur.
- **Réseau** : trafic entrant et sortant par conteneur.

Tous les conteneurs de l'infrastructure sont visibles : `db01`, `db02`, `db03`, `lamp_nginx`, `lamp_pma`, `mon-galera-wordpress-1`, `grafana`, `cadvisor`, etc.



Mysql exporter

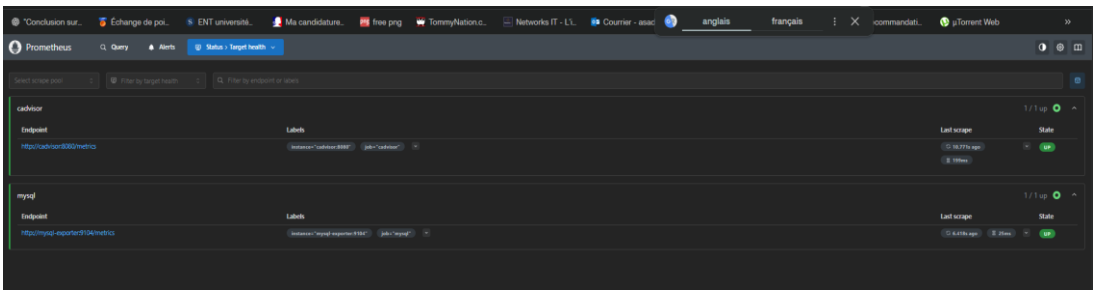
Pour exporter les metriques de nos différentes bases de données on va devoir utiliser l'image mysql exporter qui se connectera à nos serveurs et recuperera les metriques pour les envoyer à promotheus et grafana les recuperera depuis ce dernier pour afficher le tout proprement

On commence par ajouter le conteneur dans notre compose.yaml

```yaml
  mysql-exporter:
    image: prom/mysqld-exporter:v0.14.0
    container_name: mysql-exporter
    environment:
      - DATA_SOURCE_NAME=exporter:exporterpass@(db01:3306)/
    ports:
      - "9104:9104"
    restart: unless-stopped
```

On fait un compose up de tout

Maintenant dans promotheus je vois ma nouvelle target



Maintenant il faut réaliser ces étapes pour intégrer le nouveau dashboard

Maintenant importe un dashboard MariaDB dans Grafana :

1. Va dans **Dashboards → Import**.
2. Dans le champ ID, mets **7362** (MySQL Overview dashboard).
3. Clique sur **Load**.
4. Sélectionne ta datasource **Prometheus**.
5. **Import**.

Voici la doc :

Intégration du MySQL Exporter – Débogage et résolution

Contexte

Après avoir mis en place le monitoring des conteneurs Docker via cAdvisor, nous avons ajouté le **mysqld-exporter** pour récupérer les métriques des serveurs MariaDB/Galera (connexions, requêtes, InnoDB, statut du cluster, etc.).

Problèmes rencontrés et étapes de débogage

1. Crash de db01 – `safe_to_bootstrap: 0`

Problème : Avant de pouvoir créer l'utilisateur exporter dans MariaDB, db01 bouclait en restart permanent avec l'erreur suivante dans `/var/lib/mysql/mysqld.err` :

```
text
```

```
It may not be safe to bootstrap the cluster from this node.
To force cluster bootstrap with this node, edit the grastate.dat file manually
and set safe_to_bootstrap to 1
```

Cause : Tous les nœuds du cluster Galera s'étaient arrêtés en même temps. Galera avait marqué db01 comme n'étant pas le dernier nœud à quitter le cluster, donc il refusait de bootstrapper.

Résolution :

1. Identifier le bon nom du volume Docker :

   ```
   bash
   docker inspect db01 | grep -A5 "Mounts"
   # → Volume : mon-galera_db_data_01
   # → Path : /var/lib/docker/volumes/mon-galera_db_data_01/_data
   ```

2. Modifier manuellement `grastate.dat` :

   ```
   bash
   sed -i 's/safe_to_bootstrap: 0/safe_to_bootstrap: 1/' \
     /var/lib/docker/volumes/mon-galera_db_data_01/_data/grastate.dat
   ```

3. Redémarrer db01 :

   ```
   bash
   docker restart db01
   ```

   db01 a ensuite bootstrappé correctement avec `--wsrep-new-cluster` et s'est synchronisé.

---

2. Création de l'utilisateur exporter dans MariaDB

Une fois db01 stable, création d'un utilisateur dédié avec les droits minimum nécessaires pour l'exporter :

```
sql
CREATE USER 'exporter'@'%' IDENTIFIED BY 'exporterpass';
GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO 'exporter'@'%';
FLUSH PRIVILEGES;
```

---

3. Incompatibilité de la version mysqld-exporter avec `DATA_SOURCE_NAME`

Problème : La version `latest` (0.19.0) de `prom/mysqld-exporter` ne reconnaît plus la variable d'environnement `DATA_SOURCE_NAME` et retournait :

```
text
failed to validate config: no user specified in section or parent
```

Cause : Les nouvelles versions de `mysqld-exporter` utilisent un fichier de configuration `.my.cnf` au lieu de la variable d'environnement `DATA_SOURCE_NAME`.

Résolution : Utilisation de la version `v0.14.0` qui supporte encore `DATA_SOURCE_NAME` :

```
text
mysql-exporter:
 image: prom/mysqld-exporter:v0.14.0
 container_name: mysql-exporter
 environment:
  - DATA_SOURCE_NAME=exporter:exporterpass@(db01:3306)/
 ports:
  - "9104:9104"
 restart: unless-stopped
```

---

4. Job `mysql` absent dans Prometheus Targets

Problème : Après démarrage du `mysql-exporter`, le job `mysql` n'apparaissait pas dans **Status → Targets** de Prometheus.

Cause : Le `prometheus.yml` n'avait pas été rechargé après ajout du job.

Résolution : Ajout du job dans `prometheus.yml` puis redémarrage de Prometheus :

```
text
- job_name: 'mysql'
 static_configs:
  - targets: ['mysql-exporter:9104']
bash
docker compose restart prometheus
```

---

5. Dashboard Grafana MySQL Overview – Variable `$host` non résolue

Problème : Le dashboard MySQL Overview (ID 7362) affichait « No data » sur tous les panels car la variable `$host` n'avait pas de datasource configurée.

Résolution : Dans **Settings → Variables → host** du dashboard :

- **Data source** → Prometheus
- **Query** :

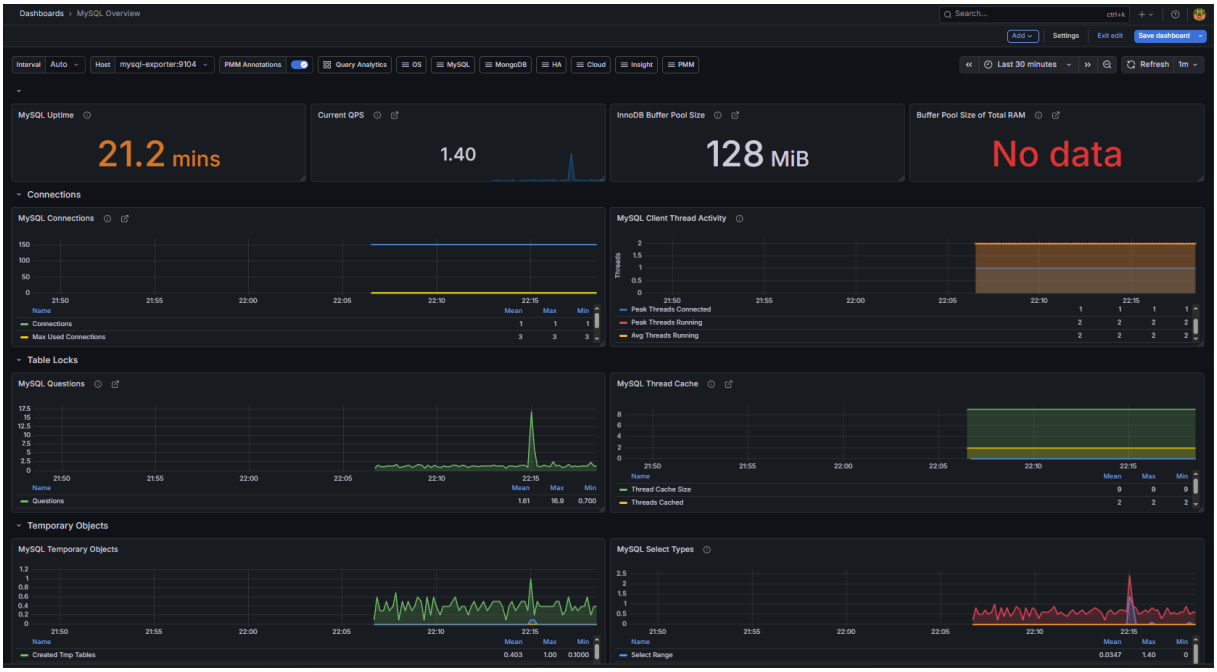
  ```
  text
  label_values(mysql_global_status_uptime, instance)
  ```

- Preview → `mysql-exporter:9104` apparaît.
- **Apply → Save dashboard**.

Résultat

Le dashboard MySQL Overview est pleinement opérationnel. Il affiche en temps réel les métriques de la base de données :

- **Uptime** du serveur MariaDB.
- **Connexions actives** et historique.
- **Requêtes par seconde** (SELECT, INSERT, UPDATE, DELETE).
- **Métriques InnoDB** (buffer pool, I/O, locks).
- **Statut de réplication** Galera.



Et promotheus.yml

```yaml
global:
  scrape_interval: 10s

scrape_configs:
  - job_name: 'cadvisor'
    static_configs:
      - targets: ['cadvisor:8080']
  - job_name: 'mysql'
    static_configs:
      - targets: ['mysql-exporter:9104']
```

Ensuite sur db01

CREATE USER 'exporter'@'%' IDENTIFIED BY 'exporterpass';

GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO 'exporter'@'%';

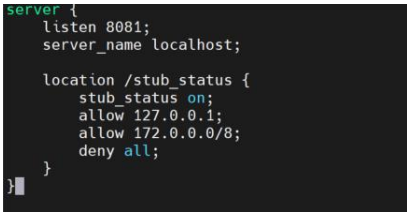
FLUSH PRIVILEGES;

EXIT;

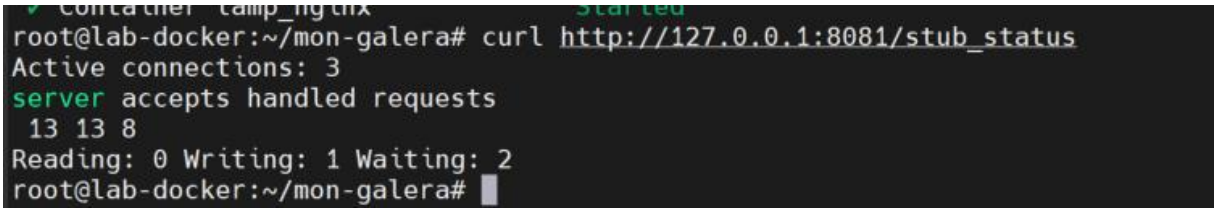
Log Nginx

Comme pour mysql on aura besoin d'un container séparer pour récupérer les metriques les envoyer à Promotheus et les afficher via grafana

Dans le fichier de conf il faut rajouter cela



Ça permettra d'avoir acces aux stats nginx



Il faut au préalable exposer le port 8081 regle de DNAT pour qu'il pointe vers 8081

Voila le bloc nginx exporter dans compose

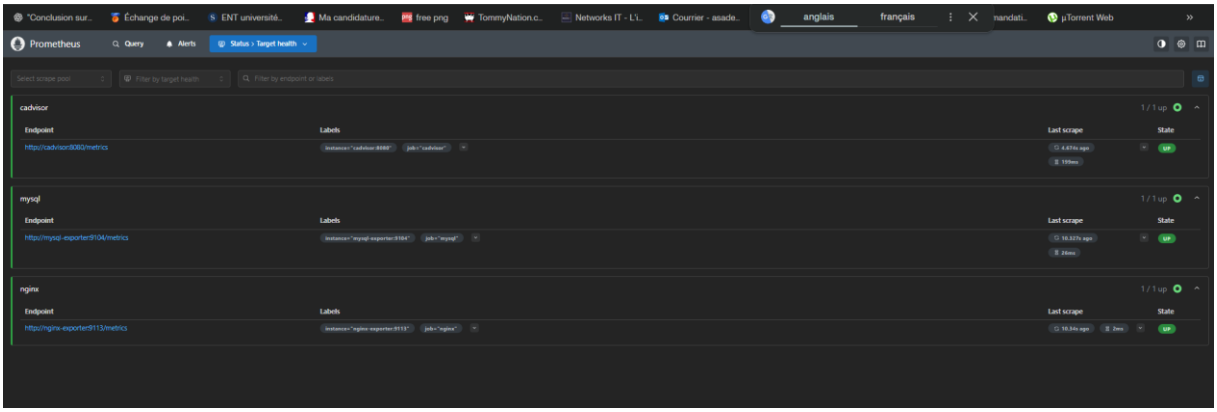
Et dans prometheus.yml

```
  - job_name: 'nginx'
    static_configs:
      - targets: ['nginx-exporter:9113']
```

docker compose up -d nginx-exporter

docker compose restart prometheus

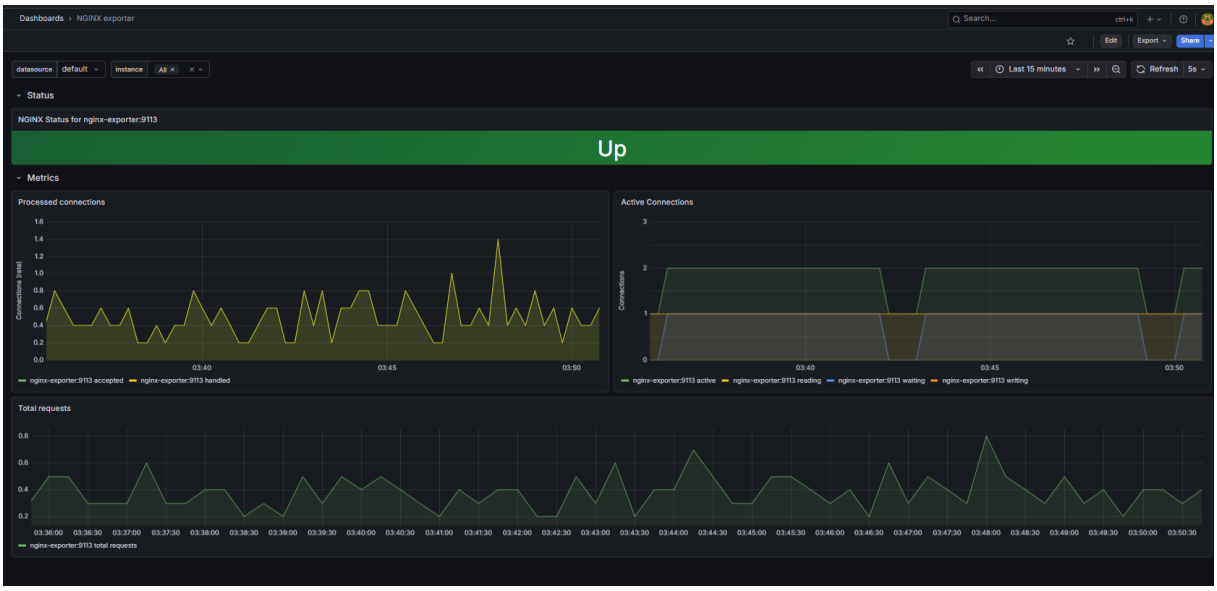
Je le vois maintenant sur promotheus et up



Maintenant importe un dashboard Nginx dans Grafana :

1. **Dashboards → Import**.
2. ID : **12708** (Nginx Prometheus Exporter dashboard).
3. **Load** → sélectionne datasource **Prometheus** → **Import**.

Directement apres l'import je vois tout


Dashboards › NGINX exporter
Search...
ctrl+k
Edit
Export
Share
datasource
default
instance
All
Last 15 minutes
Refresh
5s
Status
NGINX Status for nginx-exporter:9113
Up
Metrics
Processed connections
Connections (rate)
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
03:40
03:45
03:50
nginx-exporter:9113 accepted
nginx-exporter:9113 handled
Active Connections
Connections
3
2
1
0
03:40
03:45
03:50
nginx-exporter:9113 active
nginx-exporter:9113 reading
nginx-exporter:9113 waiting
nginx-exporter:9113 writing
Total requests
0.8
0.6
0.4
0.2
03:36:00
03:36:30
03:37:00
03:37:30
03:38:00
03:38:30
03:39:00
03:39:30
03:40:00
03:40:30
03:41:00
03:41:30
03:42:00
03:42:30
03:43:00
03:43:30
03:44:00
03:44:30
03:45:00
03:45:30
03:46:00
03:46:30
03:47:00
03:47:30
03:48:00
03:48:30
03:49:00
03:49:30
03:50:00
03:50:30
nginx-exporter:9113 total requests