Module mineur

Prometheus & Grafana : c'est quoi ?

Prometheus
→ c'est la "base de données" pour les **métriques** (CPU, RAM, latence, nombre de requêtes).
Il va "scraper" (= interroger régulièrement) les services qui exposent des chiffres :

- `/metrics` de ton app backend (métriques applicatives)
- `/metrics` de cadvisor (métriques Docker/containers)

Il les stocke dans une base temporelle.

Grafana
→ c'est l'**interface web** qui se connecte à Prometheus.
Il permet de visualiser ces métriques en **dashboards** (graphiques, jauges, alertes).

Comment ça s'articule

Imagine un pipeline :

```
Backend / Services → (expose /metrics) → Prometheus → Grafana
                   + cAdvisor (stats containers)
```

- Ton **backend** expose un endpoint `/metrics` (grâce à `prom-client`).
- **cAdvisor** expose les métriques Docker (CPU, RAM, disque, réseau par conteneur).
- **Prometheus** va chercher (scrape) ces métriques toutes les 15s.
- **Grafana** lit les données de Prometheus et affiche des graphes (req/s, erreurs, latence, CPU…).

Lancer le module

Après un `make up` ou `docker compose up -d` :

- **Prometheus** (rien d'interessant à voir): http://localhost:9090
  → onglet *Status* → *Targets* : liste des services surveillés.
  → onglet *Alerts* : règles d'alerte.

- **Grafana** : http://localhost:3000
    - user = `admin`
    - password = `admin` (ou celui que tu as défini)

Voir le dashboard sur la page http://localhost:3000 : menu / bashboard / ft_transcendence

Metric observables :

Santé & disponibilité

- **Scrape health** : montre si Prometheus arrive à joindre chaque service (`up = 1`).
  À lire : une série par service (gateway, svc-auth, svc-chat, …). Si **0**, le service est down ou `/metrics` est inaccessible.
- **Uptime (process)** : durée depuis le dernier démarrage du process.
  À lire : si ça repart à 0 → redémarrage du service.

Ressources (par conteneur — cAdvisor)

- **Container CPU** : charge CPU de chaque conteneur (pics = charge, boucles, fuites).
- **Container Memory (working set)** : mémoire réellement utilisée. Une montée continue sans retour = suspicion de fuite.
- **Container restarts (24h)** : redémarrages récents.
- **Container liveness (last seen)** : dernier “signal de vie” du conteneur.

Trafic & performance HTTP (app/gateway)

- **Request rate (req/s)** : activité de l’API (intéressant pour voir les pics de charge).
- **Error rate (5xx)** : stabilité perçue par les clients.
- **p95 latency** : latence “quasi pire cas” — si ça monte, il y a engorgement (I/O, DB, CPU…).
- **Avg response time** : tendance générale (moins sensible que p95).

WebSocket (temps réel)

- **WS connections actives** : nombre de connexions simultanées (souvent côté **gateway**).
- **WS messages/s** : messages entrants/sortants (utile pour le chat et le jeu).
- **WS erreurs** : trames invalides, déconnexions anormales (à surveiller sous charge).

Métrique métier (exemple)

- **Compteur de visites** : total/variation (preuve bout-en-bout : proxy → gateway → service → stockage).
- **Alertmanager** : http://localhost:9093
  → reçoit et affiche les alertes Prometheus.
  cAdvisor : http://localhost:8081
  → métriques Docker par conteneur.

User et mots de passe

- **Grafana** :
    - User : `admin`
    - Password : `admin` (modifiable dans `.env`).

Comment tester chaque service

1) Prometheus

Vérifier qu'il tourne :

```
curl http://localhost:9090/-/healthy
```

✅ Réponse `Healthy`.

Vérifier les cibles scrappées :
Navigateur → http://localhost:9090/targets
→ tu dois voir `backend`, `cadvisor`, `prometheus`, `grafana`.

2) Grafana

Navigateur → http://localhost:3000
→ login `admin/admin`.

Puis :

- Menu gauche → **Connections / Data sources** → vérifier que Prometheus est configuré.

- Menu gauche → **Dashboards** → ouvrir le dashboard `ft_transcendence`.

3) cAdvisor

Navigateur → http://localhost:8081
→ voir la liste des conteneurs et leurs stats.

4) Alertmanager

Navigateur → http://localhost:9093
→ liste des alertes actives (BackendDown, HighErrorRate…).

Tester avec des requêtes HTTP

1. **Générer du trafic** :

```
for i in {1..200}; do curl -s -o /dev/null
http://localhost:8080/api/health; done
```

→ ça va faire 200 requêtes sur ton backend.

2. **Vérifier dans Prometheus** :

   http://localhost:9090/graph?g0.expr=sum(rate(http_request_duration_seconds_count[1m]))
   → tu dois voir les requêtes/s.

3. **Vérifier dans Grafana** :
   Dashboard `ft_transcendence` → tu dois voir le taux de requêtes, latence p95, erreurs 5xx, CPU/Mem.

Diagnostic rapide

Prometheus ne marche pas

```
docker compose ps prometheus
docker compose logs -f prometheus
```

- Vérifie `prometheus.yml` (scrape targets).

Grafana ne marche pas

```
docker compose ps grafana
docker compose logs -f grafana
```

- Vérifie la datasource Prometheus.

Pas de métriques backend

Vérifie que `/metrics` renvoie bien quelque chose :

```
curl http://localhost:8000/metrics
```

📌 Résumé en 3 tests rapides :

```
curl http://localhost:9090/-/healthy             # Prometheus OK
curl http://localhost:8000/metrics | head -20    # Backend expose
bien ses métriques
open http://localhost:3000                       # Grafana OK (login
admin/admin)
```