

ft_transcendence

Information générale et architecture

Démarrage rapide

```
make up    # Lance tout (dev)
make down  # Arrête tout
make clean # arrete tout et supprime les volumes
make logs  # Voir les logs
make restart # arrete tout, supprime les volume , et relance
```

URLs importantes :

- App : <https://localhost:8443> (<http://localhost:8080> est redirigée vers la précédente)
- Websocket : <https://localhost:8443/ws-test.html>
- Grafana : <http://localhost:3000> (admin/admin)
- Prometheus : <http://localhost:9090>
- Kibana : <http://localhost:5601>
- Alertmanager : <http://localhost:9093>

Structure du projet

```
ft_transcendence/
├── backend/          # Node.js/TypeScript (Fastify) — modules = microservices logiques
│   └── src/modules/{auth,chat,game,tournament,visits}/http.ts
├── frontend/         # React + Vite + TypeScript
├── proxy/            # Nginx (reverse proxy + TLS)
├── monitoring/       # Observabilité (Prometheus, Grafana, Alertmanager, cAdvisor, ELK)
│   ├── grafana/ (provisioning + dashboards)
│   ├── prometheus/ (scrape + rules)
│   ├── alertmanager/
│   └── elk/ (elasticsearch, logstash, kibana)
├── scripts/          # testeurs, charge, init ELK...
└── Doc/              # Documentation (ce fichier)
```

- Backend en microservice avec une porte d'entrée 'gateway'

Avantages techniques

- **Scalabilité** : Architecture microservices containerisée
- **Sécurité** : SSL/TLS natif, reverse proxy, isolation des services
- **Monitoring** : Observabilité complète (métriques, logs, alertes)

Technologies utilisées

- **Frontend** : React, TypeScript, Vite, TailwindCSS
- **Backend** : Node.js, Express, TypeScript, WebSocket
- **Base de données** : PostgreSQL
- **Monitoring** : Prometheus, Grafana, ELK Stack
- **Infrastructure** : Docker, Nginx, SSL

Containerisation

Frontend Interface utilisateur (React + Vite)	frontend
Nginx	proxy
Backend API REST et WebSocket (Node.js + TypeScript) Microservices SQLite (intégrée au service, pas de conteneur séparé).	gateway chat auth game tournament visits (pour test)
Monitoring Reverse proxy avec SSL/TLS	grafana prometheus cadvisor alertmanager elasticsearch logstash kibana

POUR GARANTIR L'ARCHITECTURE, NE TOUCHEZ PAS À

- `docker-compose.yml` - Architecture figée
- `monitoring/` - Prometheus/Grafana configurés
- `elk/` - Stack de logs configurée
- `*/Dockerfile` → bases images, users, ports exposés (sécurité/CI)
- `proxy/nginx.conf` - Routage configuré
- `proxy/certs/`, `proxy/entrypoint.sh` → génération/chargement certs & bootstrap proxy

Mise en place d'un testeur

`./scripts/testeur.sh`

Utile pour voir si on a pas casse toute l'archi

- Proxy & Gateway (Nginx)
- API (via Gateway)
- API – Pings par service (via Gateway)
- WebSockets
- Prometheus / Grafana
- ELK