C2H6Ethan / **DistributedChess**

Code Issues Pull requests Agents Actions Projects Wiki Security and quality

Watch 0

Fullstack Chess App with loadbalancing

0 stars **0** forks **0** watching Branches Activity
Tags

Public repository

1 Branch **0** Tags Go to file T Go to file Add file Code

C2H6Ethan updated versions 8b2ccf8 · now

backend	updated versions	now
engine	bump safe dependencies to latest	1 hour ago
frontend	updated versions	now
loadtest	add JWT refresh tokens, replace k6 ...	4 hours ago
.gitignore	added gitignore	3 months ago
architecture.md	add architecture.md	3 months ago
docker-compose.yml	updated versions	now
readme.md	update README: architecture diagr...	3 hours ago

README

Distributed Chess

A fault-tolerant multiplayer chess platform built as a microservice system. Designed to survive instance failure — during a live demo, killing one backend or engine container has no visible effect on active games.

Architecture

```
Browser
  └── Traefik (port 80, round-robin)
        ├── Go Referee ×2  ──── PostgreSQL (persistent state)
        │     └── C++ Engine ×2 (move validation, bot AI)
        └── React Frontend ×1 (nginx)
```

- **Traefik** load-balances across both Go replicas and strips the `/api` prefix
- **Go referees** are stateless — any replica can serve any request; all state lives in Postgres
- **C++ engines** are internal-only (not reachable from outside); Go round-robins across them via Docker DNS
- **PostgreSQL** uses a named volume (`pgdata`) so data survives container restarts

Tech Stack

Layer	Technology
Frontend	React 18 + Vite + Tailwind CSS, `react-chessboard` , `chess.js`
Load Balancer	Traefik v3.1
Backend	Go 1.23, `golang-jwt/jwt/v5` , `lib/pq` , `bcrypt`
Engine	C++, `cpp-httplib` v0.18.5, `nlohmann/json` v3.11.3
Database	PostgreSQL 15
Deployment	Docker Compose

Quick Start

Requires Docker. One command brings up the entire system including DB migration:

```
JWT_SECRET=your_secret docker compose up --build -d
```

- App: http://localhost
- `JWT_SECRET` is the only required secret — defaults to `changeme_in_production` if omitted (not safe for production)

JWT Authentication

All API endpoints except `/register` , `/login` , and `/refresh` require a `Bearer` token.

Flow:

1. `POST /register` or `POST /login` → returns `{ token, refresh_token }`
2. Include `Authorization: Bearer <token>` on every protected request
3. Access tokens expire after **15 minutes**
4. `POST /refresh` with `{ refresh_token }` → returns a new rotated token pair

OWASP compliance:

- Short-lived access tokens (15 min), long-lived refresh tokens (30 days)
- Refresh token rotation on every use — old token is immediately invalidated
- Reuse detection: concurrent refresh attempts with the same token get a `401` (RowsAffected check)
- Algorithm pinned to HS256 in `parseToken` ; any other algorithm is rejected

- Secret loaded from env var; server panics at startup if unset

Load Test

Tests `GET /games` — a JWT-protected endpoint — using hey.

```
brew install hey
./loadtest/refresh_test.sh
```

The script registers a user, verifies token refresh/rotation, then fires **500 requests at 50 concurrency** against the protected endpoint. On a local machine:

```
Requests/sec:  ~1576
p50:           19 ms
p95:           74 ms
p99:           92 ms
[200] 500/500  (0% errors)
```

Bottleneck is JWT verification + PostgreSQL round-trip (`resp wait ~25ms avg`). The two backend replicas and two engine replicas can be seen handling the load via the infra dashboard at `GET /api/stats` .

Failover Demo

With the system running, kill one backend replica and traffic continues uninterrupted:

```
docker compose ps                          # find a backend container name
docker stop distributedchess-backend-1    # kill it
# play a move in the browser — works fine, routed to backend-2
docker compose up -d                       # bring it back
```

Same works for engine replicas.

DB Schema

```
users(id, username, password_hash)
games(id, white_id, black_id, current_fen, status, ...)
moves(id, game_id, ply, uci, fen_after)
refresh_tokens(id, user_id, token_hash, expires_at)
```

Schema is created automatically on first boot via `CREATE TABLE IF NOT EXISTS` in `backend/db.go` .

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 2


C2H6Ethan


claude Claude

Languages

C++ 41.9% **JavaScript** 35.1% **Go** 19.7% **Shell** 1.4% **Dockerfile** 1.1% **CMake** 0.4% **Other** 0.4%

Suggested workflows

Based on your tech stack

SLSA Generic generator Configure

Generate SLSA3 provenance for your existing release workflows

C/C++ with Make Configure

Build and test a C/C++ project using Make.

Publish Node.js Package Configure

Publishes a Node.js package to npm.

More workflows Dismiss suggestions