

Project information

Created on
February 23, 2026

Name	Last update
 db	2 weeks ago
 load_testing	4 hours ago
 nginx	2 months ago
 service	4 hours ago
 .dockerignore	2 months ago
 .gitignore	2 months ago
 README.md	1 minute ago
 compose.yaml	2 weeks ago
 init.sh	2 months ago

 [README.md](#)

DSy26 Game of Life

A full-stack distributed single-page implementation of Conway's Game of Life with user authentication, persistent worlds, and load balancing.

Quick Start

```
# First time – generates .env with a random JWT secret, sets up the database and starts the docker container
bash ./init.sh

# Every subsequent time
docker compose up
```

Access the app at: <http://localhost:3000>

How to Play

Standard Conway's Game of Life:

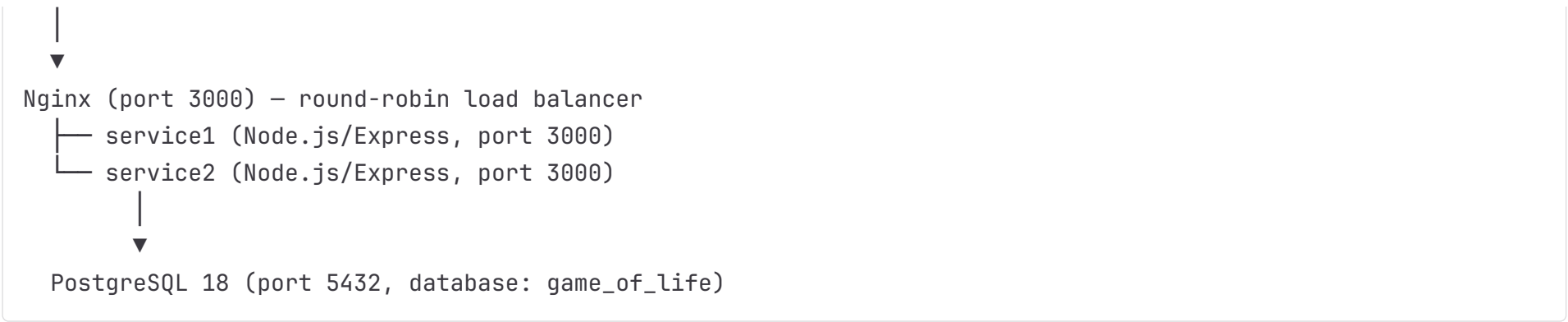
- A live cell with 2 or 3 neighbors survives
- A dead cell with exactly 3 neighbors becomes alive
- All other cells die or remain dead

Use the righthand sidebar to manage and create your worlds, then start drawing with left-click. Use right-click to clear cells.

You can control the simulation with the buttons at the bottom of the screen, letting it run automatically or manually advancing time steps.

Architecture

Browser



Both service instances are stateless and share the same PostgreSQL database, so either can handle any request.

Components

Nginx (nginx/)

Reverse proxy and load balancer. Distributes traffic between the two backend instances using round-robin. Marks a backend as down for 5 seconds after a single failed request and routes traffic to the healthy instance.

Service (service/)

Node.js/Express backend that also serves the frontend. Responsibilities:

- Serves `src/index.html` and static assets
- REST API for authentication and user/world management
- Stores cell data as packed bits (8 cells per byte) in PostgreSQL

Database (db/)

PostgreSQL schema initialized via `db/init.sql` . Three tables:

- `users` : username + Argon2-hashed password
- `refresh_tokens` : token string + expiry, linked to user
- `worlds` : name, dimensions, cell data (BYTEA), linked to user

Frontend (service/src/)

Vanilla JavaScript, no framework:

- `model/auth.js` — authentication state and API calls
- `model/main.js` — world loading, saving, and management
- `model/game_logic.js` — Conway's Game of Life logic
- `view/main.js` — canvas rendering and UI event handling

Libraries

Layer	Library	Version	Purpose
Backend	express	^5.2.1	HTTP server and routing
Backend	pg	^8.19.0	PostgreSQL client
Backend	argon2	^0.44.0	Password hashing
Backend	jsonwebtoken	^9.0.3	JWT signing and verification
Backend	cookie-parser	^1.4.7	Cookie middleware
Backend	helmet	^8.1.0	Security headers
Infrastructure	PostgreSQL	18	Persistent storage
Infrastructure	Nginx	latest	Load balancing
Infrastructure	Node.js	24-alpine	Runtime

API Endpoints

Method	Path	Description
--------	------	-------------

Method	Path	Description
POST	/api/register	Create account
POST	/api/login	Login, sets cookies
POST	/api/logout	Clear session cookies
POST	/api/refresh	Use refresh token to get new access token
GET	/api/me	Current user info
DELETE	/api/me	Delete account
GET	/api/worlds	List worlds
GET	/api/worlds/:id	Load world with cell data
POST	/api/worlds	Create world
PUT	/api/worlds/:id	Save world state
DELETE	/api/worlds/:id	Delete world

Authentication

JWT-based with two token types stored as httpOnly cookies:

- **Access token:** 15-minute expiry, required for all protected endpoints
- **Refresh token:** 7-day expiry, used to obtain a new access token

Passwords are hashed with Argon2. The JWT secret is generated once by `init.sh` and stored in `.env`.

World Storage

Cell data is packed as bits (1 bit per cell) before sending to the server and storing in the database. This means a 512×512 world requires 32 KB of storage. Worlds are limited to dimensions between 1×1 and 512×512, and each user can have at most 5 worlds.

Configuration

All configuration is in `.env` (created by `init.sh`):

Variable	Default	Description
JWT_SECRET	random	128-char hex secret for signing JWTs
PORT	3000	Express server port
DB_HOST	db	PostgreSQL hostname
POSTGRES_DB	game_of_life	Database name
POSTGRES_USER	postgres	Database user
POSTGRES_PASSWORD	postgres	Database password

Load Testing

```
cd load_testing
bash run.sh
```

Uses the [hey](#) HTTP benchmarking tool.