AronBA / **loom**

Code Issues Pull requests Agents Actions Projects Wiki Security and quality

Watch 0

Vibe coded logging dashboard

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

tbocek finale abgabe aaca8dc · 2 minutes ago

backend	finale abgabe	2 minutes ago
frontend	finale abgabe	2 minutes ago
nginx	fix bugs	41 minutes ago
scripts	finale abgabe	2 minutes ago
.gitignore	add jwt , add realtime update	last week
README.md	fix readme again	29 minutes ago
docker-compose.yml	fix bugs	41 minutes ago

README

Loom - Distributed Log Aggregation System

A distributed log aggregation system with a Spring Boot backend and Angular frontend. This system is designed for high availability with automated failover and health monitoring.

Architecture

- **Nginx**: Load balancer and reverse proxy with automated failover logic.
- **Backend**: Dual Spring Boot instances (`backend-1` , `backend-2`) for horizontal scaling.
- **Frontend**: Angular application served via Nginx.
- **Database**: PostgreSQL for persistent log storage.

Prerequisites

- Docker
- Docker Compose

How to Run

The entire stack is containerized and pre-configured for high availability.

1. **Start the application:**

```
docker-compose up -d --build
```

2. **Verify health status:**

```
docker-compose ps
```

   Wait until `loom-backend-1` and `loom-backend-2` show as `(healthy)`.

3. **Access the Application:**
   - **Frontend Dashboard**: http://localhost
   - **Backend API**: http://localhost/api
   - **Health Check**: http://localhost/api/actuator/health

4. **Stop the application:**

```
docker-compose down
```

High Availability & Failover

The system is configured to handle backend crashes gracefully. Nginx monitors the health of the backends and will automatically fail over if one instance becomes unresponsive.

Simulating a Crash

To test the failover logic:

1. Start a load test (see below).
2. Kill one of the backend instances:

```
docker kill loom-backend-1
```

3. Observe that the system remains operational via `backend-2`.

Testing

Load Testing

A load test script is included to verify performance and failover resilience:

```
./scripts/loadtest.sh
```

Note: Requires `hey` to be installed.

Log Generation

To generate sample logs for the dashboard:

```
python3 scripts/generate_random_logs.py
```

Authentication & Security

The system implements a secure, OWASP-compliant authentication flow using JWTs and Refresh Tokens stored in **HttpOnly Cookies**.

Key Features

- **HttpOnly Cookies**: Prevents XSS attacks by ensuring tokens cannot be accessed via JavaScript.
- **Token Rotation**: Every time a refresh token is used, it is deleted and a new one is issued. This detects and mitigates token theft.
- **Stateless Access Tokens**: Short-lived JWTs for performance.
- **Stateful Refresh Tokens**: Stored in the database for control over user sessions.

Flow Logic

1. **Login**: User provides credentials -> Backend validates -> Backend issues an Access Token (JWT) and a Refresh Token (UUID) -> Both are sent to the browser as `Set-Cookie` headers.
2. **Authorized Requests**: The browser automatically includes the cookies in every request to `/api`. The backend validates the JWT.
3. **Token Refresh**: When the JWT expires, the frontend (via an interceptor) calls `/api/auth/refresh`. The backend verifies the refresh token, deletes it, and issues a brand-new pair of tokens.
4. **Logout**: The backend deletes the refresh token from the database and instructs the browser to clear the cookies.

Relevant Files

- **Logic & Controller**: AuthController.java
- **Token Management**: RefreshTokenService.java
- **JWT Utilities**: JwtUtils.java
- **Security Configuration**: WebSecurityConfig.java

Development

- **Backend**: Located in `backend/`. Uses Spring Boot, Spring Security (JWT), and Flyway.
- **Frontend**: Located in `frontend/`. Angular application with Tailwind CSS.
- **Database**: PostgreSQL 18.

Default User

- **Username**: `testuser`
- **Password**: `password`

Releases

No releases published
Create a new release

Deployments 1

github-pages 3 months ago

Packages

No packages published
Publish your first package

Contributors

No contributors

Languages

Java 40.5% **HTML** 27.3% **TypeScript** 22.4% **Python** 6.2% **Shell** 1.9% **Dockerfile** 0.7% **Other** 1.0%