TaskBoard – Distributed Task Management System

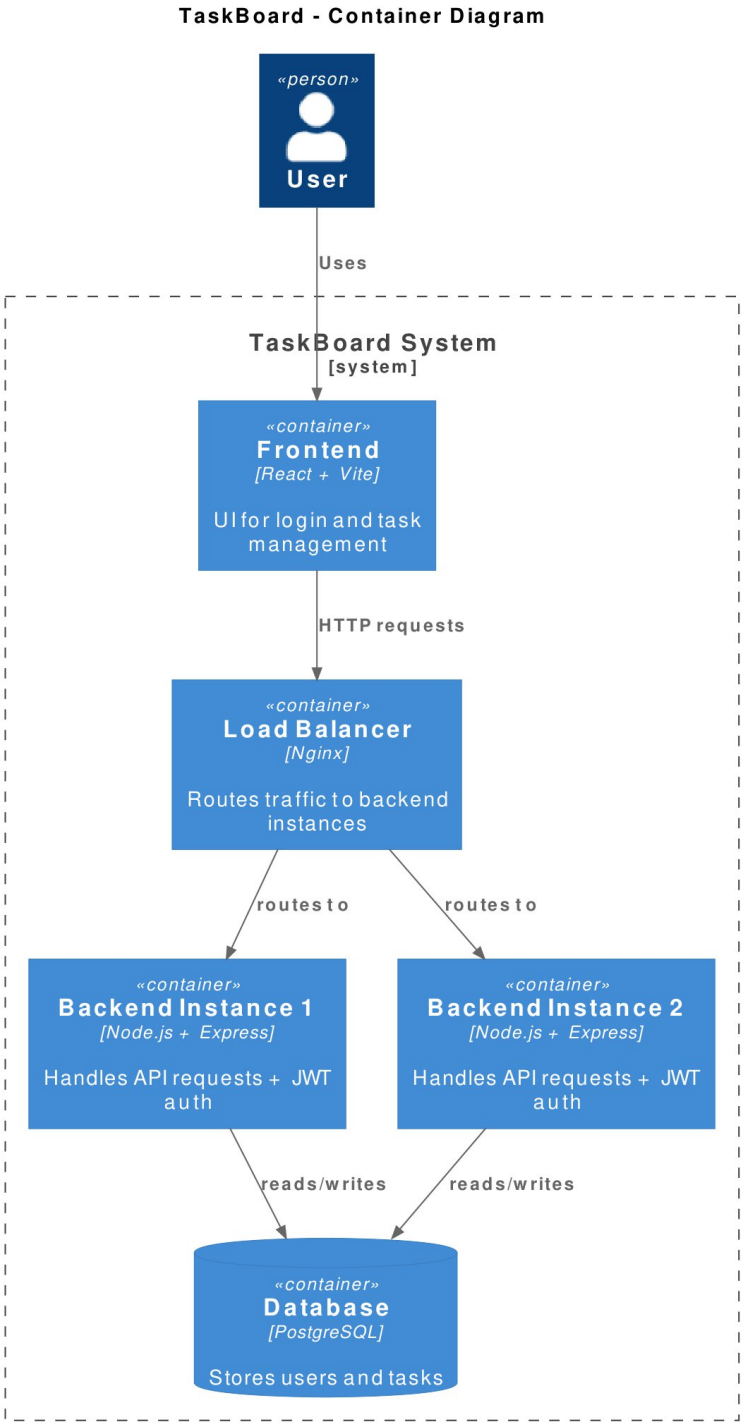
Challenge Task FS 2026

Goal

- Build a distributed web application from scratch
- Multiple backend instances
- Load balancing
- JWT authentication
- Persistent storage
- Containerized deployment
- Failover support

System Overview

- Frontend: React + Vite
- Backend: Node.js + Express
- Database: PostgreSQL
- Load Balancer: Nginx
- Containers: Docker Compose



Backend Design

- REST API
- JWT authentication
- Stateless backend
- /register
- /login
- /tasks

JWT Authentication

- User logs in
- Backend generates JWT token
- Token stored in localStorage
- Authorization header used
- Backend validates token
- Stateless authentication

Persistent Storage

- PostgreSQL database
- Stores users and tasks
- Shared state across instances
- Solves distributed state consistency problem

Load Balancing & Failover

- Nginx distributes requests
- Traffic routed to backend1 and backend2
- Failover supported
- System remains operational after backend failure

Containerization

- Docker Compose orchestration
- Single command startup
- frontend
- backend1
- backend2
- nginx
- db

Frontend Features

- Register/Login
- Persistent login
- Create tasks
- Mark tasks as done
- Protected API access
- Conditional UI rendering

Load Testing

- Tool: k6
- JWT-protected endpoint tested
- GET /api/tasks
- 20 and 200 virtual users tested

Load Test Results (200 VUs)

- Two backends: avg 227.84ms
- One backend: avg 383.15ms
- p95 latency increased under load
- System remained operational

Load Test Analysis

- Horizontal scaling works
- Latency increases under higher load
- Backend processing is bottleneck
- Database access contributes to latency
- Tail latency increases significantly

Demo