Survivor Pool – DevOps Documentation

1. Overview

This document explains how the CI/CD pipelines and Docker setup work for the Survivor Pool project. It's intended for DevOps engineers and contributors who manage deployment and automation.

2. GitHub Actions Workflows

a) CI Pipeline – .github/workflows/ci.yml

Purpose: Validate that both backend and frontend build successfully on every push.

- Backend job:
  - Runs in Survivor_Pool/Backend
  - Installs dependencies with npm ci (or npm install if lockfile missing)
  - Runs npm run build
- Frontend job:
  - Runs in Survivor_Pool/Survivor
  - Installs dependencies with yarn install
  - Builds with yarn build

b) Frontend Deploy – .github/workflows/deploy-frontend-pages.yml

Purpose: Deploy frontend to GitHub Pages on every push to main.

Steps:
1. Checkout repo
2. Setup Node.js with Yarn cache
3. Inject backend URL into .env
4. Build frontend with yarn build
5. Add SPA fallback (404.html)
6. Upload Survivor_Pool/Survivor/dist to GitHub Pages

Secrets needed:
- VITE_BACKEND_URL: Backend Render URL

c) Backend Deploy to Render – .github/workflows/cd-backend-render.yml

Purpose: Trigger a new deployment of backend service on Render when code changes.

Steps:
- Call Render Deploy Hook stored in RENDER_DEPLOY_HOOK secret.

Secrets needed:
- RENDER_DEPLOY_HOOK

d) Backend Docker Build – .github/workflows/cd-backend-ghcr.yml

Purpose: Build backend Docker image and push to GitHub Container Registry (GHCR).

Steps:
- Checkout repo
- Log in to GHCR
- Build Dockerfile
- Push image

3. Docker Setup

a) Backend Dockerfile

```
FROM node:20-alpine AS builder
WORKDIR /app

COPY Survivor_Pool/Backend/package*.json ./
RUN npm ci

COPY Survivor_Pool/Backend/ ./
RUN npx prisma generate --schema=src/prisma/schema.prisma || true
RUN npm run build

FROM node:20-alpine AS runner
WORKDIR /app
ENV NODE_ENV=production
ENV PORT=3000

COPY --from=builder /app/node_modules ./node_modules
RUN npm prune --omit=dev

COPY --from=builder /app/dist ./dist

EXPOSE 3000
CMD ["node", "dist/app.js"]
```

b) .dockerignore

```
node_modules
dist
.git
.github
```

.env
*.log

4. Environment Variables (Secrets)

Variable	Where	Purpose
VITE_BACKEND_URL	GitHub Actions	Inject backend URL into frontend build
RENDER_DEPLOY_HOOK	GitHub Actions	Trigger Render redeploy of backend
DATABASE_URL	Render Backend	Prisma DB connection string (MySQL/MariaDB)
JEB_API_KEY	Render Backend	Auth key for external API calls

5. Workflow Summary

Push to main →
1. CI checks build (backend + frontend)
2. Backend deploy workflow triggers Render redeploy
3. Frontend deploy workflow builds React app and pushes to GitHub Pages
4. (Optional) GHCR workflow builds and publishes backend Docker image

With this setup, deployment is fully automated: one git push updates both backend and frontend.