Railway Deployment Best Practices - Muscles AI Fitness

Optimized Configuration for Railway

This document outlines the best practices implemented to ensure successful deployment on Railway.

Key Improvements

1. **Memory Optimization**
   - Set `NODE_OPTIONS=--max-old-space-size=512` to respect Railway's memory constraints
   - Optimized Vite build process with chunk splitting and minification
   - Removed source maps in production builds
   - Added production pruning of dev dependencies
2. **Build Process Improvements**
   - Enforced Node.js v20 usage via `.nvmrc` and package.json engine field
   - Set consistent pnpm configuration via `.npmrc`
   - Optimized build command to use production mode explicitly
   - Added chunk splitting for better loading performance
3. **Deployment Configuration**
   - Reduced healthcheck timeout from 300s to 120s for faster failure detection
   - Added healthcheck interval configuration
   - Changed restart policy from "always" to "on-failure" with max retries
   - Added engine constraints to ensure compatible Node.js version

4. **Package Management**
   - Enforced pnpm as the package manager
   - Added hoisted node linking for better compatibility
   - Locked dependency resolution to highest mode
   - Set up auto peer dependency installation
5. **Vite Optimizations**
   - Environment-based configuration for development vs production
   - Improved chunk splitting strategy
   - Disabled sourcemaps in production
   - Set sensible asset inlining limits
   - Configured proper port binding for Railway

Railway.json Configuration

```
{
  "$schema": "https://railway.app/railway.schema.json",
  "build": {
    "builder": "nixpacks",
    "buildCommand": "pnpm install --frozen-lockfile && pnpm
build:prod"
  },
  "deploy": {
    "startCommand": "pnpm start",
    "healthcheckPath": "/",
    "healthcheckTimeout": 120,
    "healthcheckInterval": 10,
    "restartPolicyType": "on-failure",
    "restartPolicyMaxRetries": 5
  },
  "environments": {
    "production": {
      "variables": {
        "NODE_ENV": "production",
        "PORT": "3000"
      }
    }
  }
}
```

Nixpacks.toml Configuration

```
[phases.setup]
nixPkgs = ['nodejs_20', 'pnpm']
aptPkgs = ['build-essential']

[phases.install]
cmds = [
  'pnpm config set node-linker hoisted',
  'pnpm install --frozen-lockfile'
]
cacheDirectories = ['.pnpm-store']

[phases.build]
cmds = ['pnpm build:prod']
cacheDirectories = ['node_modules/.cache']

[start]
cmd = 'pnpm start'
dependsOn = ['build']

[variables]
NODE_ENV = 'production'
PORT = '3000'
NODE_OPTIONS = '--max-old-space-size=512'
```

Package.json Scripts

```
"scripts": {
  "dev": "vite",
  "build": "tsc -b && vite build",
  "build:prod": "NODE_ENV=production tsc -b && vite build",
  "lint":
"eslint . --ext ts,tsx --report-unused-disable-directives --max-
warnings 0",
  "preview": "vite preview --host 0.0.0.0 --port ${PORT:-3000}",
  "start": "vite preview --host 0.0.0.0 --port ${PORT:-3000}",
  "type-check": "tsc --noEmit",
  "postinstall": "[ \"$NODE_ENV\" = \"production\" ] && pnpm prune
--prod || echo 'Skipping production pruning'"
}
```

Troubleshooting Common Issues

Memory-Related Errors

- If encountering OOM (Out of Memory) errors, check for:
- Large dependencies that can be optimized
- Memory leaks in the application
- Build process consuming too much memory

Build Failures

- Ensure the build command is executable in a clean environment
- Verify that all dependencies are properly declared in package.json
- Check for type errors that might be failing the TypeScript compilation

Runtime Errors

- Inspect the application logs for runtime errors
- Ensure all required environment variables are properly set
- Verify that the start command correctly serves the built application

References

- Railway Documentation
- Nixpacks Documentation
- Vite Deployment Guide
- pnpm Best Practices

Document created: 2025-08-18