Railway Deployment Guide

W.D. Gann Trading Application - Railway Deployment

This guide provides step-by-step instructions for deploying the W.D. Gann Trading Application to Railway.

Prerequisites

Before deploying to Railway, ensure you have:

1. A Railway account (https://railway.app/)
2. The Railway CLI (https://docs.railway.app/develop/cli) installed (optional but recommended)
3. A GitHub account with this repository pushed to GitHub
4. Git installed locally

Repository Setup

1. Push to GitHub

Ensure your code is pushed to a GitHub repository:

```
git remote add origin https://github.com/YOUR_USERNAME/YOUR_REPO_NAME.git
git branch -M main
git push -u origin main
```

Railway Deployment Steps

Option 1: Deploy via Railway Dashboard (Recommended for First-Time Users)

Step 1: Create a New Project

1. Log in to Railway Dashboard (https://railway.app/dashboard)
2. Click **“New Project”**
3. Select **“Deploy from GitHub repo”**
4. Authorize Railway to access your GitHub account (if not already done)
5. Select your repository from the list

Step 2: Configure Build Settings

Railway will automatically detect the project settings from `nixpacks.toml`. The configuration includes:

- **Node.js Version**: 20.x (pinned via nixpacks.toml)
- **Package Manager**: pnpm 8.x

- **Build Command**: `pnpm build` (automatically detected)
- **Start Command**: `pnpm start` (automatically detected)

Step 3: Set Environment Variables

In the Railway dashboard, navigate to your project settings and add the following environment variables:

Required Environment Variables:

Variable	Value	Description
`NODE_ENV`	`production`	Sets the application to production mode
`PORT`	Auto-set by Railway	Railway automatically sets this
`HOST`	`0.0.0.0`	Allows external connections
`ALLOWED_ORIGINS`	`https://your-app.up.railway.app`	**IMPORTANT**: Replace with your actual Railway URL

Optional Environment Variables:

Variable	Value	Description
`LOG_LEVEL`	`info`	Logging level (error, warn, info, debug, trace)

Example:

```
NODE_ENV=production
HOST=0.0.0.0
ALLOWED_ORIGINS=https://gann-trading-app-production.up.railway.app
LOG_LEVEL=info
```

> ⚠️ **CRITICAL**: After your first deployment, Railway will assign a URL to your application. You **MUST** update the `ALLOWED_ORIGINS` environment variable with this URL for CORS to work correctly.

Step 4: Deploy

1. Click **“Deploy Now”** or wait for Railway to automatically deploy
2. Railway will:
   - Install dependencies using `pnpm install --frozen-lockfile`
   - Build the frontend using `vite build`
   - Start the server using `tsx server/index.ts`
3. Monitor the build logs in the Railway dashboard
4. Once deployment is complete, Railway will provide a public URL

Step 5: Update CORS Configuration

1. Copy your Railway deployment URL (e.g., `https://gann-trading-app-production.up.railway.app`)

2. Go to **Variables** in your Railway project
3. Update `ALLOWED_ORIGINS` to your deployment URL
4. Railway will automatically redeploy with the updated configuration

---

Option 2: Deploy via Railway CLI (Advanced Users)

Step 1: Install Railway CLI

```
# macOS/Linux
curl -fsSL https://railway.app/install.sh | sh

# Windows (PowerShell)
iwr https://railway.app/install.ps1 | iex
```

Step 2: Login to Railway

```
railway login
```

Step 3: Initialize Railway Project

```
# Navigate to your project directory
cd /path/to/gann-trading-app

# Link to Railway
railway link
```

Step 4: Set Environment Variables

```
railway variables set NODE_ENV=production
railway variables set HOST=0.0.0.0
railway variables set ALLOWED_ORIGINS=https://your-app.up.railway.app
railway variables set LOG_LEVEL=info
```

Step 5: Deploy

```
railway up
```

Post-Deployment

1. Verify Deployment

Once deployed, verify your application:

1. **Health Check**: Visit `https://your-app.up.railway.app/api/health`
   - Should return: `{"status":"ok","timestamp":"..."}`

2. **Application**: Visit `https://your-app.up.railway.app`
   - Should load the W.D. Gann Trading Platform interface

3. **API Endpoint**: The tRPC API is available at `/api/trpc`

2. Update Domain (Optional)

If you want to use a custom domain:

1. Go to Railway project settings
2. Navigate to **"Domains"**
3. Add your custom domain
4. Update your DNS records as instructed
5. Update `ALLOWED_ORIGINS` environment variable with your custom domain

3. Monitor Logs

View real-time logs in Railway:

```
# Via CLI
railway logs

# Via Dashboard
Navigate to your project → "Deployments" tab → Click on a deployment
```

Configuration Files

The following files configure Railway deployment:

`nixpacks.toml`

```toml
[phases.setup]
nixPkgs = ["nodejs_20", "pnpm"]

[phases.install]
cmds = ["pnpm install --frozen-lockfile"]

[phases.build]
cmds = ["pnpm build"]

[start]
cmd = "pnpm start"

[variables]
NODE_ENV = "production"
```

`package.json` (Key Scripts)

```
{
  "scripts": {
    "build": "pnpm run clean && vite build",
    "start": "NODE_ENV=production tsx server/index.ts"
  },
  "engines": {
    "node": ">=20.0.0",
    "pnpm": ">=8.0.0"
  }
}
```

Troubleshooting

Build Failures

Issue: Build fails with dependency errors

```
# Solution: Clear Railway cache
railway run pnpm install --force
railway up
```

Issue: TypeScript errors during build

```
# Solution: Run type check locally
pnpm typecheck
# Fix any errors before redeploying
```

Runtime Errors

Issue: “CORS policy” errors in browser console

```
Solution:
1. Check ALLOWED_ORIGINS environment variable
2. Ensure it matches your Railway deployment URL exactly
3. Include the protocol (https://)
4. No trailing slash
```

Issue: “Cannot find module” errors

```
Solution:
1. Ensure all dependencies are in package.json
2. Run: pnpm install
3. Commit package.json and pnpm-lock.yaml
4. Redeploy
```

Issue: Application not responding

```
Solution:
1. Check Railway logs for errors
2. Verify PORT is not hardcoded (Railway sets it automatically)
3. Ensure HOST=0.0.0.0 in environment variables
```

Database/API Issues

Issue: Yahoo Finance API not working

```
Note: yahoo-finance2 works without an API key by default.
If rate limited, consider:
1. Implementing caching
2. Reducing API call frequency
3. Adding a premium API key if available
```

Environment-Specific Considerations

Development vs Production

Aspect	Development	Production (Railway)
Node Version	20.x	20.x (pinned via nixpacks)
Package Manager	pnpm	pnpm
Server Mode	Watch mode (`tsx watch`)	Production mode (`tsx`)
CORS	Wide open (`*`)	Restricted to deployment URL
Source Maps	Enabled	Enabled
Static Files	Vite dev server	Served from dist/public

Performance Optimization

1. Chunking Large Bundles

The build currently generates a large JavaScript bundle (>500KB). Consider:

```
// vite.config.ts
export default defineConfig({
  build: {
    rollupOptions: {
      output: {
        manualChunks: {
          'react-vendor': ['react', 'react-dom'],
          'charts': ['recharts'],
          'trpc': ['@trpc/client', '@trpc/react-query'],
        }
      }
    }
  }
});
```

2. Caching Strategy

Consider implementing Redis or similar caching for market data:

```
# Add Redis service in Railway
railway add redis

# Update environment variables
railway variables set REDIS_URL=${{REDIS.REDIS_URL}}
```

Continuous Deployment

Railway automatically deploys when you push to your main branch:

```
# Make changes
git add .
git commit -m "Your changes"
git push origin main

# Railway automatically detects and deploys
```

To disable automatic deployments:
1. Go to Railway project settings
2. Navigate to **“Settings” → “Deployments”**
3. Disable **“Automatic Deployments”**

Scaling Considerations

Horizontal Scaling

Railway supports horizontal scaling:
1. Navigate to project settings
2. Go to **“Settings” → “Resources”**
3. Adjust replica count

Note: For stateless applications like this one, horizontal scaling works out of the box.

Vertical Scaling

Adjust resources allocated to your application:

1. Navigate to project settings
2. Go to **“Settings” → “Resources”**
3. Adjust CPU and memory limits

Security Best Practices

1. Environment Variables

- **Never commit** `.env` files to version control
- Use Railway’s environment variable management
- Rotate sensitive credentials regularly

2. CORS Configuration

- Always set `ALLOWED_ORIGINS` to your specific domain
- Never use `*` in production
- Update when domain changes

3. Dependencies

```
# Regularly check for vulnerabilities
pnpm audit

# Update dependencies
pnpm update
```

Maintenance

Updating the Application

```
# Pull latest changes
git pull origin main

# Install dependencies
pnpm install

# Test locally
pnpm dev

# Build and verify
pnpm build
pnpm start

# Commit and push
git add .
git commit -m "Update: description"
git push origin main
```

Monitoring

1. **Railway Dashboard**: Real-time metrics and logs
2. **Health Check Endpoint**: `https://your-app.up.railway.app/api/health`
3. **Browser Console**: Check for client-side errors

Cost Estimates

Railway pricing (as of deployment):

- **Free Tier**: $5 of usage per month
- **Pro Plan**: $5/month + usage
- Typical usage for this app: ~$0-10/month depending on traffic

Estimate Breakdown:
- **Build Time**: ~1-2 minutes per deployment
- **Runtime**: Minimal (single process, low memory)
- **Bandwidth**: Depends on traffic

Support and Resources

- Railway Documentation (https://docs.railway.app/)
- Railway Discord (https://discord.gg/railway)
- Project Repository Issues (https://github.com/YOUR_USERNAME/YOUR_REPO_NAME/issues)
- W.D. Gann Trading App Docs (./README.md)

Quick Reference

Essential Commands

```
# Local development
pnpm dev

# Build for production
pnpm build

# Start production server (locally)
pnpm start

# Type checking
pnpm typecheck

# Railway CLI
railway login
railway link
railway logs
railway up
railway variables
```

Essential URLs

- **Railway Dashboard**: https://railway.app/dashboard
- **Health Check**: `https://your-app.up.railway.app/api/health`
- **tRPC API**: `https://your-app.up.railway.app/api/trpc`
- **Application**: `https://your-app.up.railway.app`

Conclusion

Your W.D. Gann Trading Application is now successfully deployed on Railway! 🎉

The application is:
- ✅ Running on Node.js 20
- ✅ Using pnpm for dependency management
- ✅ Automatically building and deploying from GitHub
- ✅ Configured with proper CORS and security settings
- ✅ Serving the React frontend and tRPC API
- ✅ Ready for production use

For issues or questions, please refer to the troubleshooting section or reach out via the support channels listed above.

Last Updated: October 27, 2025
Version: 1.0.0
Deployment Platform: Railway