🚂 Railway Deployment Guide - Workigom Backend

🔴 Critical Issue Identified

Your deployment was failing because Railway was using the **wrong Dockerfile**. The logs showed Railway was trying to build with the root-level `Dockerfile` (which is for the frontend with nginx) instead of the `backend/Dockerfile`.

The Error

```
ERROR: failed to solve: failed to calculate checksum of ref: "/nginx.conf": not found
```

This happened because Railway was:
1. Looking at the root directory (not the `backend` directory)
2. Using the frontend Dockerfile
3. Trying to find `nginx.conf` which doesn't exist in the backend

✅ Solution

The fix is simple but **CRITICAL**: You must configure Railway to use the `backend` directory as the root directory.

📋 Step-by-Step Deployment Instructions

Step 1: Prerequisites

Before deploying, ensure you have:
- ✅ A Railway account
- ✅ A PostgreSQL database created in Railway (or external database)
- ✅ Your GitHub repository connected to Railway

Step 2: Create New Service

1. **Log in to Railway** → https://railway.app/
2. **Click "New Project"** → "Deploy from GitHub repo"
3. **Select your repository** (`workigom`)
4. **Click "Add variables"** to configure environment variables later

Step 3: ⚠️ CRITICAL - Configure Root Directory

This is the most important step that was missing:

1. Go to your Railway project
2. Click on your **backend service**
3. Go to **Settings** tab

4. Find **“Root Directory”** setting
5. Set it to: `backend`
6. Click **“Save”** or apply the change

Without this step, Railway will use the wrong Dockerfile and fail!

Step 4: Verify Builder Settings

In the same Settings page:

1. Scroll to **“Builder”** section
2. Ensure it’s set to **“Dockerfile”**
3. The `railway.toml` file will handle the rest

Step 5: Configure Environment Variables

Click on **“Variables”** tab and add the following:

Required Variables:

```
DATABASE_URL=postgresql://user:password@host:5432/database
JWT_SECRET=your-super-secret-jwt-key-here
```

Optional Variables (with defaults):

```
NODE_ENV=production
PORT=3001
```

How to Generate JWT_SECRET:

Run this command in your terminal:

```
openssl rand -base64 32
```

Or use this Node.js command:

```
node -e "console.log(require('crypto').randomBytes(32).toString('base64'))"
```

How to Get DATABASE_URL:

Option 1: Using Railway PostgreSQL
1. In Railway, create a new PostgreSQL database
2. Railway will automatically provide the `DATABASE_URL`
3. You can link it to your backend service

Option 2: Using External Database

```
postgresql://username:password@host:port/database_name
```

Step 6: Deploy

1. Once the root directory is set to `backend`, Railway will automatically trigger a new deployment
2. Watch the deployment logs in the **“Deployments”** tab

3. Look for successful build messages

Step 7: Verify Deployment

Check Health Endpoint:

Once deployed, visit:

```
https://your-railway-domain/api/health
```

You should see:

```
{
  "status": "OK",
  "timestamp": "...",
  "uptime": ...,
  "database": "connected"
}
```

Check Deployment Logs:

Look for these success messages:

```
🚀 Starting Workigom Backend...
📦 Running Prisma migrations...
✅ Starting server...
🚀 Workigom Backend API Server
📍 Environment: production
🌐 Server running on: http://localhost:3001
```

🔍 Troubleshooting

Issue 1: "nginx.conf not found" Error

Problem: Railway is using the wrong Dockerfile (frontend instead of backend)

Solution:
- Set Root Directory to `backend` in Railway Settings
- Redeploy

Issue 2: "Cannot find module" errors

Problem: Dependencies not installed correctly

Solution:
1. Check that `package.json` is in the backend directory
2. Verify the Dockerfile is running `npm ci`
3. Check build logs for installation errors

Issue 3: Database Connection Errors

Problem: `DATABASE_URL` not set or incorrect

Solution:
1. Verify `DATABASE_URL` is set in Railway Variables

2. Check the connection string format: `postgresql://user:password@host:port/database`
3. Ensure the database is accessible from Railway

Issue 4: "/app/dist not found" Error

Problem: TypeScript compilation failed during build

Solution:
1. Check build logs for TypeScript errors
2. Verify `tsconfig.json` has `"outDir": "./dist"`
3. Ensure all source files are in the `src` directory
4. Check that the Dockerfile runs `npm run build`

Issue 5: Migrations Fail

Problem: Prisma migrations not applying

Solution:
1. Check `DATABASE_URL` is correct
2. Verify `prisma/schema.prisma` exists
3. Check deployment logs for migration errors
4. Manually run migrations if needed: `bash`
```
   railway run npx prisma migrate deploy
```

Issue 6: Port Binding Issues

Problem: Server can't bind to port

Solution:
1. Ensure you're using `process.env.PORT` in your code
2. Railway automatically sets the `PORT` variable
3. Don't hardcode port numbers

📁 File Structure Verification

Ensure your backend directory has this structure:

```
backend/
├── Dockerfile              ✅ Backend Dockerfile
├── railway.toml            ✅ Railway configuration
├── package.json            ✅ Dependencies
├── tsconfig.json           ✅ TypeScript config
├── prisma/
│   └── schema.prisma        ✅ Database schema
├── src/
│   ├── server.ts            ✅ Entry point
│   ├── app.ts               ✅ Express app
│   ├── config/              ✅ Configuration
│   ├── controllers/         ✅ Route handlers
│   ├── middleware/          ✅ Middleware
│   ├── routes/              ✅ API routes
│   └── utils/               ✅ Utilities
└── .env.example            ✅ Example env vars
```

🎯 Quick Checklist

Before deploying, verify:

- [ ] Root Directory is set to `backend` in Railway Settings
- [ ] Builder is set to `Dockerfile`
- [ ] `DATABASE_URL` environment variable is set
- [ ] `JWT_SECRET` environment variable is set
- [ ] PostgreSQL database is created and accessible
- [ ] `railway.toml` file exists in the backend directory
- [ ] `Dockerfile` exists in the backend directory
- [ ] All source code is in `src/` directory
- [ ] `package.json` has `build` script: `"tsc"`

🔄 Redeployment Process

If you need to redeploy:

1. **Trigger Redeploy:**
   - Push to your GitHub repository, OR
   - Click "Redeploy" in Railway dashboard

2. **Watch Logs:**
   - Go to "Deployments" tab
   - Click on the latest deployment
   - Watch real-time logs

3. **Verify Success:**
   - Check for "✅ Starting server…" message
   - Test the health endpoint
   - Test your API endpoints

📞 Support

If you encounter issues:

1. **Check Railway Logs:** Most issues show up in deployment logs
2. **Verify Configuration:** Double-check all settings mentioned above
3. **Test Locally:** Ensure the Docker build works locally:
   ```
   bash
      cd backend
      docker build -t workigom-backend .
      docker run -p 3001:3001 --env-file .env workigom-backend
   ```

🎉 Success Indicators

Your deployment is successful when you see:

✅ Build completes without errors
✅ Health check endpoint returns 200 OK
✅ Database connection is established
✅ API endpoints are accessible
✅ No error logs in Railway console

🔐 Security Reminders

- ⚠️ Never commit `.env` files to Git
- ⚠️ Use strong, random JWT secrets
- ⚠️ Rotate secrets regularly
- ⚠️ Use Railway's private networking for database connections when possible

---

📚 Additional Resources

- Railway Documentation (https://docs.railway.app/)
- Prisma Deployment Docs (https://www.prisma.io/docs/guides/deployment)
- Docker Best Practices (https://docs.docker.com/develop/dev-best-practices/)

Last Updated: October 23, 2025
Version: 2.0 - Fixed Root Directory Issue