🚀 Comprehensive Render.com Deployment Guide for Workigom

📋 Table of Contents

1. Prerequisites
2. Project Structure Overview
3. Backend Deployment
4. Frontend Deployment
5. Database Setup
6. Environment Variables
7. Post-Deployment Steps
8. Troubleshooting
9. Common Issues
10. Performance Optimization

Prerequisites

Before deploying to Render.com, ensure you have:

- ✅ A GitHub account with the Workigom repository
- ✅ A Render.com account (free tier is sufficient to start)
- ✅ Repository structure with:
- `backend/` directory (Node.js + Express + TypeScript)
- `src-frontend/` directory (React + Vite)
- ✅ PostgreSQL database (will be created on Render)

Project Structure Overview

```
workigom/
├── backend/              # Backend API service
│   ├── src/
│   │   ├── config/       # Database and app configuration
│   │   ├── controllers/  # Route controllers
│   │   ├── middleware/   # Authentication & validation
│   │   ├── models/       # Database models
│   │   ├── routes/       # API routes
│   │   └── server.ts     # Main entry point
│   ├── package.json
│   ├── tsconfig.json
│   ├── .env.example
│   └── Dockerfile
│
├── src-frontend/         # Frontend React application
│   ├── components/       # React components
│   ├── contexts/         # React contexts (Auth, etc.)
│   ├── pages/            # Page components
│   ├── types/            # TypeScript types
│   ├── package.json
│   ├── vite.config.ts
│   └── .env.example
│
└── RENDER_DEPLOYMENT_GUIDE.md  # This file
```

Backend Deployment

Step 1: Create a New Web Service

1. **Log in to Render.com** and click **“New +” → “Web Service”**

2. **Connect Your Repository**
   - Select “Connect GitHub” or “Connect GitLab”
   - Choose the `volkanakbulut73/Workigom` repository
   - Grant necessary permissions

3. **Configure the Backend Service**

Basic Settings:

`Name: workigom-backend`
`Region: Oregon (US West) or closest to your users`
`Branch: main`

⚠️ CRITICAL: Root Directory

`Root Directory: backend/`

> **Note:** This tells Render where your backend code lives. The `/` at the end is important!

Runtime:

`Runtime: Node`

Build Command:
`bash`
`npm install && npm run build`

Start Command:
`bash`
`npm start`

Instance Type:
`Free (or Starter if you need better performance)`

1. **Advanced Settings** (click “Advanced”)

Auto-Deploy:
`Yes (enabled) - deploys automatically on git push`

Health Check Path:
`/health`

Step 2: Configure Backend Environment Variables

Click on **“Environment”** tab and add these variables:

Required Database Variables

```
DATABASE_URL=<Will be automatically filled when you link PostgreSQL>
# Or if manual:
# DATABASE_URL=postgresql://username:password@host:5432/database_name

# Alternative individual database variables (if not using DATABASE_URL):
DB_HOST=<your-postgres-hostname>
DB_PORT=5432
DB_NAME=workigom
DB_USER=<your-db-username>
DB_PASSWORD=<your-db-password>
```

Application Configuration

```
NODE_ENV=production
PORT=3000

# JWT Configuration (IMPORTANT: Generate strong secrets!)
JWT_SECRET=<generate-a-strong-random-string-at-least-64-characters>
JWT_EXPIRES_IN=7d

# CORS Configuration
CORS_ORIGIN=https://your-frontend-url.onrender.com
# Or for multiple origins:
# CORS_ORIGIN=https://your-frontend-url.onrender.com,https://workigom.vercel.app
```

Optional but Recommended

```
# API Rate Limiting
RATE_LIMIT_WINDOW_MS=900000
RATE_LIMIT_MAX_REQUESTS=100

# File Upload Configuration
MAX_FILE_SIZE=5242880
UPLOAD_DIR=/tmp/uploads

# Logging
LOG_LEVEL=info
```

How to Generate Secure JWT_SECRET

Run this command locally:

```
node -e "console.log(require('crypto').randomBytes(64).toString('hex'))"
```

Or use an online generator (from a trusted source).

Frontend Deployment

Step 1: Create a Static Site

1. **Click “New +” → “Static Site”**
2. **Connect the Same Repository**
   - Select the `volkanakbulut73/Workigom` repository
3. **Configure the Frontend Service**

Basic Settings:
`Name: workigom-frontend`
`Branch: main`

⚠️ CRITICAL: Root Directory
`Root Directory: src-frontend/`

> **Note:** This is where your React/Vite app lives!

Build Command:
`bash`
`npm install && npm run build`

Publish Directory:
`bash`
`dist`

> **Note:** Vite builds to `dist/` by default. If you use Create React App, use `build` instead.

Step 2: Configure Frontend Environment Variables

Click on **“Environment”** tab:

```
# Backend API URL (CRITICAL!)
VITE_BACKEND_URL=https://workigom-backend.onrender.com

# Alternative names your app might use:
VITE_API_URL=https://workigom-backend.onrender.com
REACT_APP_API_URL=https://workigom-backend.onrender.com

# Application Environment
NODE_ENV=production
VITE_NODE_ENV=production

# Optional: Google OAuth Configuration
VITE_GOOGLE_CLIENT_ID=<your-google-oauth-client-id>
```

⚠️ **IMPORTANT:**
- Vite requires environment variables to be prefixed with `VITE_`
- The backend URL must match your backend service’s URL
- No trailing slash in the URL

Database Setup

Step 1: Create PostgreSQL Database

1. **Click “New +” → “PostgreSQL”**
2. **Configure Database:**
   ```
   Name: workigom-db
   Database: workigom
   User: workigom_user
   Region: Same as your backend service
   Plan: Free (Starter for production)
   ```
3. **Note the Connection Details:**
   - Internal Database URL (for backend)
   - External Database URL (for local development/migration)

Step 2: Link Database to Backend

1. Go to your **backend service**
2. Click on **“Environment”** tab
3. Add a new environment variable:
   ```
   bash
   DATABASE_URL=<paste-internal-database-url-here>
   ```

Step 3: Run Database Migrations

Option A: Automatic migrations on deployment

Update your backend’s start command:

```
npm run migrate && npm start
```

Option B: Manual migration using Render Shell

1. Go to your backend service
2. Click on **“Shell”** tab
3. Run:
   ```
   bash
   npm run migrate
   # or
   npm run typeorm migration:run
   ```

Step 4: Seed the Database

1. **Open the Render Shell** for your backend service
2. **Run the seed endpoint:**
   ```
   bash
   curl -X POST http://localhost:3000/api/seed/run \
     -H "Content-Type: application/json" \
     -H "X-Admin-Key: your-admin-secret-key"
   ```

Or visit:
`https://workigom-backend.onrender.com/api/seed/run`

1. **Test Users Created:**
   - **Admin:** admin@workigom.com / Admin123!
   - **Employer:** employer@workigom.com / Employer123!
   - **Helper:** helper@workigom.com / Helper123!

Environment Variables

Complete Backend `.env` Example

```
# Database
DATABASE_URL=postgresql://user:password@host:5432/workigom

# Server
NODE_ENV=production
PORT=3000

# JWT
JWT_SECRET=your-super-secret-jwt-key-min-64-chars-recommended
JWT_EXPIRES_IN=7d

# CORS
CORS_ORIGIN=https://workigom-frontend.onrender.com

# Admin
ADMIN_SECRET_KEY=your-admin-secret-for-seeding

# Rate Limiting
RATE_LIMIT_WINDOW_MS=900000
RATE_LIMIT_MAX_REQUESTS=100

# File Uploads
MAX_FILE_SIZE=5242880
UPLOAD_DIR=/tmp/uploads
```

Complete Frontend `.env` Example

```
# API Configuration
VITE_BACKEND_URL=https://workigom-backend.onrender.com
VITE_API_URL=https://workigom-backend.onrender.com

# Environment
NODE_ENV=production
VITE_NODE_ENV=production

# OAuth (Optional)
VITE_GOOGLE_CLIENT_ID=your-google-oauth-client-id
```

Post-Deployment Steps

1. Update CORS Configuration

Once your frontend is deployed, update the backend's `CORS_ORIGIN`:

```
CORS_ORIGIN=https://workigom-frontend.onrender.com
```

Redeploy the backend for changes to take effect.

2. Verify Health Endpoints

Backend Health Check:

```
curl https://workigom-backend.onrender.com/health
```

Expected response:

```
{
  "status": "ok",
  "timestamp": "2024-11-01T12:00:00.000Z",
  "database": "connected"
}
```

Frontend Check:

```
curl https://workigom-frontend.onrender.com
```

Should return HTML of your React app.

3. Test Authentication Flow

1. Visit `https://workigom-frontend.onrender.com`
2. Try logging in with test user: `admin@workigom.com` / `Admin123!`
3. Verify API calls are working (check Network tab in DevTools)

4. Monitor Logs

Backend Logs:
1. Go to backend service on Render
2. Click **“Logs”** tab
3. Monitor for errors

Common Log Messages to Look For:
- ✅ “Server is running on port 3000”
- ✅ “Database connected successfully”
- ❌ “CORS error” → Update CORS_ORIGIN
- ❌ “Database connection failed” → Check DATABASE_URL
- ❌ “JWT error” → Verify JWT_SECRET is set

Troubleshooting

Issue 1: Build Failed - “Cannot find module”

Cause: Dependencies not installed correctly

Solution:

```
# In backend/ or src-frontend/
npm install
npm run build
```

Check that `package.json` and `package-lock.json` are committed to git.

Issue 2: CORS Error in Browser Console

Error Message:

```
Access to XMLHttpRequest at 'https://workigom-backend.onrender.com/api/auth/login'
from origin 'https://workigom-frontend.onrender.com' has been blocked by CORS policy
```

Solution:

1. **Update backend environment variable:**
   `bash`
   `CORS_ORIGIN=https://workigom-frontend.onrender.com`
2. **Or in backend code** (`backend/src/config/cors.ts`):
   `typescript`
   `const corsOptions = {`
   `origin: [`
   `'https://workigom-frontend.onrender.com',`
   `'http://localhost:5173', // for local development`
   `],`
   `credentials: true,`
   `};`
3. **Redeploy backend** after making changes

Issue 3: Frontend Shows “Network Error” or “ERR_CONNECTION_REFUSED”

Cause: Frontend cannot reach the backend

Solutions:

1. **Verify VITE_BACKEND_URL is correct:**
   `bash`
   `# In frontend environment variables`
   `VITE_BACKEND_URL=https://workigom-backend.onrender.com`
2. **Check backend is running:**
   `bash`
   `curl https://workigom-backend.onrender.com/health`
3. **Verify frontend code uses the environment variable:**
   `typescript`
   `// In frontend code`
   `const API_URL = import.meta.env.VITE_BACKEND_URL || 'http://localhost:3000';`
4. **Rebuild and redeploy frontend** after updating environment variables

---

Issue 4: 401 Unauthorized on Login

Causes & Solutions:

A. JWT_SECRET mismatch or missing:

```
# Verify JWT_SECRET is set in backend environment
JWT_SECRET=your-secret-key-here
```

B. Credentials not being sent:

```
// Frontend API calls should include credentials
fetch(url, {
  credentials: 'include',
  headers: {
    'Content-Type': 'application/json',
  },
});
```

C. CORS credentials not allowed:

```
// Backend CORS config
app.use(cors({
  origin: process.env.CORS_ORIGIN,
  credentials: true, // Must be true!
}));
```

Issue 5: Database Connection Failed

Error in logs:

```
Error: connect ECONNREFUSED
```

Solutions:

1. **Verify DATABASE_URL format:**
   `bash`
   `# Correct format:`
   `DATABASE_URL=postgresql://username:password@host:port/database`

2. **Check database service is running:**
   - Go to your PostgreSQL database on Render
   - Verify it shows “Available”

3. **Use Internal Database URL:**
   - For backend service, use the Internal Database URL
   - External URL is for connecting from outside Render

4. **Check connection pooling:**
   `typescript`
   `// backend/src/config/database.ts`
   `const pool = new Pool({`
   `connectionString: process.env.DATABASE_URL,`
   `ssl: {`
   `rejectUnauthorized: false // Required for Render`

```
    }
  });
```

Issue 6: Service Keeps Spinning Down (Free Tier)

Cause: Render free tier spins down services after 15 minutes of inactivity

Solutions:

A. Keep-Alive Ping Service:

```
// Add to backend
setInterval(async () => {
  console.log('Keep-alive ping');
}, 14 * 60 * 1000); // Every 14 minutes
```

B. External Monitoring Service:
- Use https://uptimerobot.com (free)
- Ping your backend health endpoint every 5 minutes
- Configure: `https://workigom-backend.onrender.com/health`

C. Upgrade to Paid Tier:
- Starter plan ($7/month) keeps services always running
- Better performance and no spin-down delays

Issue 7: “Failed to load resource: net::ERR_CONNECTION_REFUSED”

Cause: Backend URL in frontend code is incorrect or backend is down

Debug Steps:

1. **Check browser console for the exact URL being called:**
   `Failed to load resource: https://workigom-backend.onrender.com/api/auth/login`
2. **Manually test that URL:**
   `bash`
   `curl https://workigom-backend.onrender.com/api/auth/login`
3. **If backend is down, check Render logs:**
   - Go to backend service → Logs tab
   - Look for startup errors
4. **Common fixes:**
   `bash`
   `# Frontend .env`
   `VITE_BACKEND_URL=https://workigom-backend.onrender.com`
   `# No trailing slash!`
   `# No /api at the end!`

Common Issues

⚠️ Issue: “Vite environment variable not loaded”

Problem:

```
console.log(import.meta.env.VITE_BACKEND_URL); // undefined
```

Solution:

1. **Ensure variable starts with `VITE_`:**
   ```bash
   # ✅ Correct
   VITE_BACKEND_URL=https://backend.onrender.com

# ❌ Wrong (won’t be exposed to client)
BACKEND_URL=https://backend.onrender.com
```

1. **Rebuild frontend after adding variables:**
   - Render → Frontend Service → Manual Deploy → Deploy Latest Commit

2. **Verify in build logs:**
   `Building with environment variables:`
   `VITE_BACKEND_URL=https://workigom-backend.onrender.com`

⚠️ Issue: “Root directory not found”

Error in Render logs:

```
Root directory 'backend' not found
```

Solution:

1. **Verify directory structure in GitHub:**
   ```
   workigom/
   ├── backend/
   │   ├── package.json
   │   └── src/
   └── src-frontend/
       ├── package.json
       └── src/
   ```

2. **Update Root Directory setting:**
   `Backend: backend/`
   `Frontend: src-frontend/`
   (Include the trailing slash!)

3. **Check branch is correct:**
   - Most repos use `main` branch
   - Some older repos use `master`

⚠️ Issue: TypeScript Build Errors

Error:

```
TS2307: Cannot find module '@/components/...'
```

Solution:

1. **Check `tsconfig.json` paths:**
   `json`
   `{`
   `"compilerOptions": {`
   `"baseUrl": ".",`
   `"paths": {`
   `"@/*": ["src/*"]`
   `}`
   `}`
   `}`

2. **Verify Vite config (for frontend):**
   ```typescript
   // vite.config.ts
   import path from 'path';

export default defineConfig({
resolve: {
alias: {
'@': path.resolve(__dirname, './src'),
},
},
});
```

1. **Ensure all dependencies are installed:**
   `bash`
   `npm install --save-dev @types/node @types/react`

Performance Optimization

1. Enable Compression

Backend:

```
import compression from 'compression';
app.use(compression());
```

2. Optimize Build Size

Frontend:

```
// vite.config.ts
export default defineConfig({
  build: {
    rollupOptions: {
      output: {
        manualChunks: {
          vendor: ['react', 'react-dom'],
          router: ['react-router-dom'],
        },
      },
    },
  },
});
```

3. Database Connection Pooling

Backend:

```
const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  max: 20, // Maximum pool size
  idleTimeoutMillis: 30000,
  connectionTimeoutMillis: 2000,
});
```

4. Add Caching Headers

Backend:

```
app.use((req, res, next) => {
  if (req.url.includes('/api/')) {
    res.set('Cache-Control', 'no-store');
  } else {
    res.set('Cache-Control', 'public, max-age=31536000');
  }
  next();
});
```

Quick Reference Card

🎯 Essential URLs After Deployment

```
# Backend Service
Backend URL: https://workigom-backend.onrender.com
Health Check: https://workigom-backend.onrender.com/health
API Base: https://workigom-backend.onrender.com/api

# Frontend Service
Frontend URL: https://workigom-frontend.onrender.com

# Database
Database Type: PostgreSQL
Connection: Internal Database URL (from Render dashboard)
```

🔑 Critical Environment Variables Checklist

Backend:
- [ ] DATABASE_URL
- [ ] JWT_SECRET
- [ ] CORS_ORIGIN
- [ ] NODE_ENV=production
- [ ] PORT=3000

Frontend:
- [ ] VITE_BACKEND_URL
- [ ] NODE_ENV=production

📝 Deployment Checklist

- [ ] Backend service created with correct root directory (`backend/`)
- [ ] Frontend service created with correct root directory (`src-frontend/`)
- [ ] PostgreSQL database created
- [ ] DATABASE_URL linked to backend
- [ ] JWT_SECRET generated and set
- [ ] CORS_ORIGIN set to frontend URL
- [ ] VITE_BACKEND_URL set to backend URL
- [ ] Database migrations run successfully
- [ ] Database seeded with test users
- [ ] Health endpoint returns 200 OK
- [ ] Login works with test credentials
- [ ] Logs show no errors

Support & Resources

- **Render Documentation:** https://render.com/docs
- **GitHub Repository:** https://github.com/volkanakbulut73/Workigom
- **Render Community Forum:** https://community.render.com

Deployment Summary

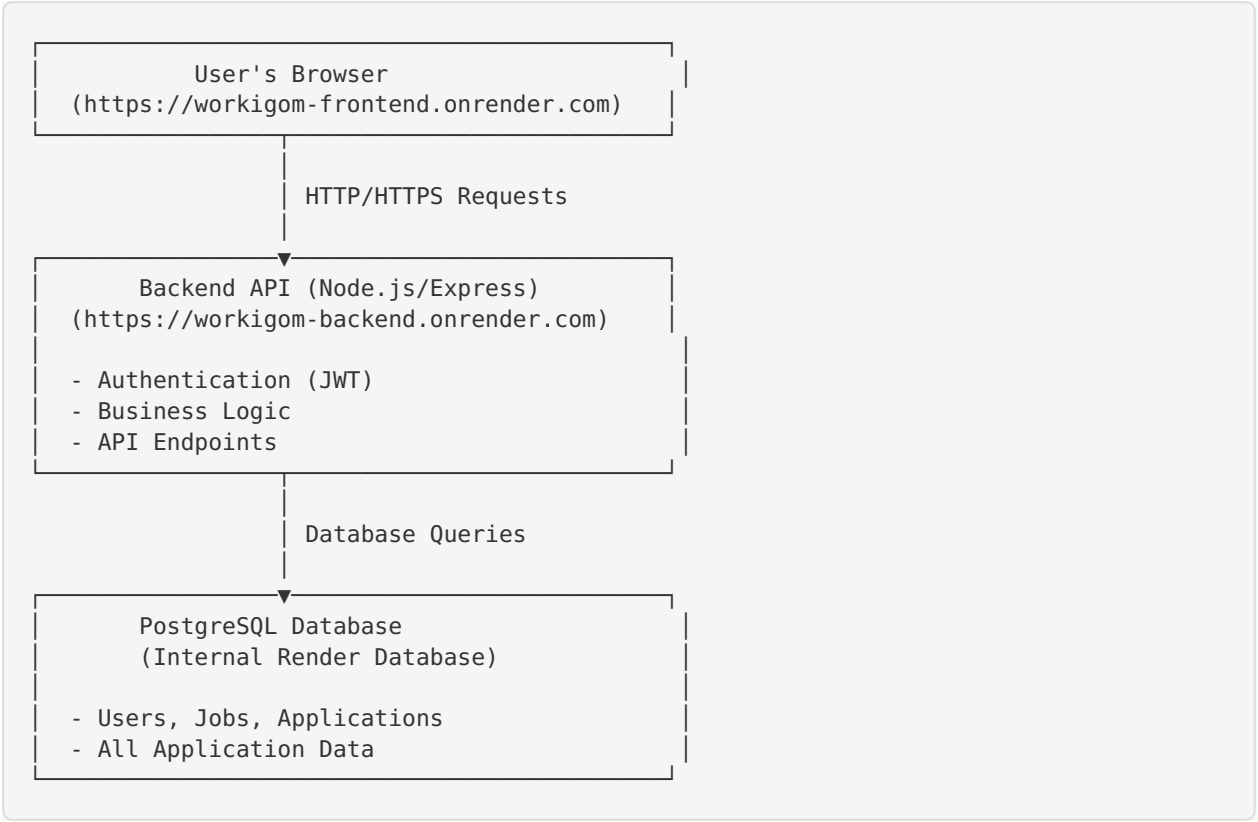
What We've Deployed:

1. **Backend Web Service** (Node.js + Express + TypeScript)
   - Handles API requests
   - JWT authentication
   - Database operations
   - Running on: `https://workigom-backend.onrender.com`

2. **Frontend Static Site** (React + Vite)
   - User interface
   - Connects to backend API
   - Running on: `https://workigom-frontend.onrender.com`

3. **PostgreSQL Database**
   - Stores all application data
   - Users, jobs, applications, etc.
   - Managed by Render

Architecture Diagram:



🎉 Congratulations!

You've successfully deployed Workigom to Render.com!

Your application is now live and accessible to users worldwide.

Next Steps:
1. Share your frontend URL with users
2. Monitor logs for any issues
3. Set up custom domain (optional)
4. Configure SSL certificates (automatic with Render)
5. Set up monitoring and alerts

Last Updated: November 1, 2024
Version: 1.0
Author: Workigom Development Team