🚀 Workigom Render Deployment Guide

This comprehensive guide will walk you through deploying the Workigom application (backend + frontend) on Render.com.

📋 Table of Contents

Prerequisites

Before you begin, ensure you have:

- ✅ A Render.com (https://render.com) account (free tier is sufficient)
- ✅ A GitHub account with the Workigom repository
- ✅ The repository pushed to GitHub: `https://github.com/volkanakbulut73/workigom`
- ✅ Basic understanding of environment variables and deployment concepts

---

Project Structure

```
workigom/
├── backend/                 # Node.js + Express + TypeScript API
│   ├── src/
│   ├── prisma/
│   │   └── schema.prisma   # Database schema
│   ├── package.json
│   └── .env.example
├── src-frontend/           # React + TypeScript + Vite frontend
│   ├── package.json
│   └── vite.config.ts
└── render.yaml            # Render blueprint configuration
```

Step 1: Connect GitHub Repository

1.1 Login to Render

1. Go to https://render.com (https://render.com)
2. Click **"Get Started"** or **"Sign In"**
3. Sign in with your GitHub account

1.2 Authorize Render

1. When prompted, authorize Render to access your GitHub repositories
2. You can grant access to all repositories or select specific ones
3. Make sure `volkanakbulut73/workigom` is accessible

Step 2: Deploy Using Blueprint

Render Blueprints allow you to deploy multiple services (backend, frontend, database) with a single click using the `render.yaml` file.

2.1 Create Blueprint

1. Go to your Render Dashboard: https://dashboard.render.com/ (https://dashboard.render.com/)
2. Click **"New +"** → **"Blueprint"**
3. Select your GitHub repository: `volkanakbulut73/workigom`
4. Render will automatically detect the `render.yaml` file
5. Review the services that will be created:
   - **workigom-backend** (Web Service)
   - **workigom-frontend** (Static Site)
   - **workigom-db** (PostgreSQL Database)
6. Click **"Apply"** to start deployment

2.2 Wait for Deployment

The deployment process will:
- Create the PostgreSQL database first
- Deploy the backend service (installs dependencies, generates Prisma client, builds TypeScript)
- Deploy the frontend service (installs dependencies, builds with Vite)

This process typically takes **5-10 minutes** for the first deployment.

Step 3: Configure Environment Variables

While the blueprint configures most environment variables automatically, you should verify and customize them.

3.1 Backend Environment Variables

Go to your backend service: **Dashboard → workigom-backend → Environment**

Required Variables (Auto-configured):

```
NODE_ENV=production
PORT=10000
DATABASE_URL=<automatically linked from database>
JWT_SECRET=<automatically generated>
JWT_REFRESH_SECRET=<automatically generated>
```

Verify/Update These:

```
JWT_EXPIRES_IN=7d
JWT_REFRESH_EXPIRES_IN=30d
CORS_ORIGIN=https://workigom-frontend.onrender.com,https://workigom.vercel.app
RATE_LIMIT_WINDOW_MS=900000
RATE_LIMIT_MAX_REQUESTS=100
MAX_FILE_SIZE=5242880
ALLOWED_FILE_TYPES=image/jpeg,image/png,image/jpg,application/pdf
```

> **Important:** Update `CORS_ORIGIN` to include your actual frontend URL once deployed.

3.2 Frontend Environment Variables

Go to your frontend service: **Dashboard → workigom-frontend → Environment**

```
VITE_BACKEND_URL=https://workigom-backend.onrender.com
```

> **Important:** Update this with your actual backend URL once it's deployed.

3.3 Save Changes

After updating environment variables:
1. Click **"Save Changes"**
2. The service will automatically redeploy

Step 4: Run Database Migrations

After the backend is deployed and the database is connected, you need to run Prisma migrations.

4.1 Access Backend Shell

1. Go to **Dashboard → workigom-backend**
2. Click on **"Shell"** tab (or **"Connect"** → **"Shell"**)
3. This opens a terminal connected to your backend service

4.2 Run Migration Command

In the shell, run:

```
npx prisma migrate deploy
```

This command will:
- Apply all pending migrations to your database

- Create all necessary tables
- Set up relationships and indexes

4.3 Verify Migration

Check the output for successful migration messages. You should see:

```
✓ Migration applied successfully
```

Step 5: Verify Deployment

5.1 Check Backend Health

1. Find your backend URL: `https://workigom-backend.onrender.com`
2. Test the health endpoint: `https://workigom-backend.onrender.com/api/health`
3. You should see a JSON response:
   ```
   json
      {
        "status": "ok",
        "timestamp": "2024-11-01T...",
        "environment": "production"
      }
   ```

5.2 Check Frontend

1. Find your frontend URL: `https://workigom-frontend.onrender.com`
2. Open it in a browser
3. Verify that the page loads correctly
4. Check browser console for any errors

5.3 Check Logs

If something isn’t working:

Backend Logs:
- Dashboard → workigom-backend → Logs

Frontend Build Logs:
- Dashboard → workigom-frontend → Logs

Database Logs:
- Dashboard → workigom-db → Logs

Step 6: Seed Database (Optional)

If you want to populate the database with test data:

6.1 Via Backend Shell

Access the backend shell and run:

```
npm run prisma:seed
```

6.2 Via API Endpoint (If Available)

If your backend has a seed endpoint:

```
curl -X POST https://workigom-backend.onrender.com/api/seed \
  -H "Content-Type: application/json"
```

6.3 Via Prisma Studio (Development)

For a GUI-based approach:

1. In backend shell, run:
   `bash`
   `   npx prisma studio`
2. This opens a web interface to manage your database

Troubleshooting

Problem: Backend Won’t Start

Symptoms:
- Backend service shows “Build succeeded, deploy failed”
- Logs show database connection errors

Solutions:
1. Verify `DATABASE_URL` is correctly set
2. Check if database is fully provisioned (can take a few minutes)
3. Ensure Prisma migrations have been run
4. Restart the backend service

Problem: Frontend Can’t Connect to Backend

Symptoms:
- Frontend loads but shows connection errors
- Browser console shows CORS errors or network errors

Solutions:
1. Verify `VITE_BACKEND_URL` in frontend environment variables
2. Check `CORS_ORIGIN` in backend includes your frontend URL
3. Make sure backend is actually running (check health endpoint)
4. Rebuild frontend after updating `VITE_BACKEND_URL`

Problem: Database Connection Failed

Symptoms:
- Backend logs show “Connection refused” or “Database not found”

Solutions:
1. Wait a few minutes - database provisioning can be slow
2. Check database status in Render dashboard

3. Verify `DATABASE_URL` format is correct
4. Ensure database and backend are in the same region

Problem: Build Failures

Backend Build Issues:

```
# Check these in backend shell:
node --version        # Should be v18+ or v20+
npm --version
npm list @prisma/client
```

Frontend Build Issues:

```
# Check Node.js version (should be 18+)
# Verify all dependencies are correctly installed
npm install
npm run build
```

Problem: Free Tier Limitations

Render free tier has limitations:
- **Spin Down:** Services sleep after 15 minutes of inactivity
- **Cold Start:** First request after sleep takes ~30 seconds
- **Build Minutes:** Limited build minutes per month
- **Bandwidth:** Limited bandwidth

Workarounds:
1. Keep services warm with periodic pings
2. Upgrade to paid plan for always-on services
3. Use external monitoring services

Manual Deployment (Alternative)

If you prefer not to use Blueprint, you can deploy services manually:

Manual Backend Deployment

1. **Create Web Service:**
   - Dashboard → New + → Web Service
   - Connect repository: `volkanakbulut73/workigom`
   - Root Directory: `backend`
   - Build Command: `npm install && npx prisma generate && npm run build`
   - Start Command: `npm run start`

2. **Add Environment Variables:**
   - Copy all variables from Step 3.1

3. **Create Database:**
   - Dashboard → New + → PostgreSQL
   - Name: `workigom-db`

   - Plan: Free
   - Copy connection string

4. **Link Database:**
   - Go to backend service → Environment
   - Set `DATABASE_URL` to the database connection string

Manual Frontend Deployment

1. **Create Static Site:**
   - Dashboard → New + → Static Site
   - Connect repository: `volkanakbulut73/workigom`
   - Root Directory: `src-frontend`
   - Build Command: `npm install && npm run build`
   - Publish Directory: `dist`

2. **Add Environment Variables:**
   - `VITE_BACKEND_URL`: Your backend URL

Post-Deployment Configuration

Update CORS Origins

After both services are deployed, update the `CORS_ORIGIN` environment variable:

```
CORS_ORIGIN=https://workigom-frontend.onrender.com,https://your-actual-frontend-url.com
```

Set Up Custom Domain (Optional)

1. Go to your service → Settings → Custom Domain
2. Add your domain
3. Configure DNS records as instructed
4. Update CORS settings accordingly

Enable Auto-Deploy

By default, services auto-deploy on git push. You can disable this:
1. Service → Settings → Auto-Deploy
2. Toggle on/off as needed

Environment Variables Quick Reference

Backend (.env)

```
NODE_ENV=production
PORT=10000
DATABASE_URL=<from-render-database>
JWT_SECRET=<auto-generated>
JWT_EXPIRES_IN=7d
JWT_REFRESH_SECRET=<auto-generated>
JWT_REFRESH_EXPIRES_IN=30d
CORS_ORIGIN=https://workigom-frontend.onrender.com
RATE_LIMIT_WINDOW_MS=900000
RATE_LIMIT_MAX_REQUESTS=100
MAX_FILE_SIZE=5242880
ALLOWED_FILE_TYPES=image/jpeg,image/png,image/jpg,application/pdf
```

Frontend (.env)

```
VITE_BACKEND_URL=https://workigom-backend.onrender.com
```

Deployment Checklist

Use this checklist to ensure everything is configured correctly:

- [ ] GitHub repository is connected to Render
- [ ] Blueprint deployed (or services created manually)
- [ ] Backend service is running
- [ ] Frontend service is running
- [ ] PostgreSQL database is provisioned
- [ ] Backend environment variables are set
- [ ] Frontend environment variables are set
- [ ] Database migrations have been run (`npx prisma migrate deploy`)
- [ ] Backend health endpoint responds: `/api/health`
- [ ] Frontend loads in browser
- [ ] Frontend can communicate with backend
- [ ] CORS is properly configured
- [ ] (Optional) Database is seeded with test data
- [ ] (Optional) Custom domain is configured

Monitoring and Maintenance

Check Service Health

Regularly monitor your services:
- **Backend:** https://workigom-backend.onrender.com/api/health

- **Frontend:** https://workigom-frontend.onrender.com
- **Logs:** Check Render dashboard for errors

Database Backups

Free tier PostgreSQL includes:
- Automatic daily backups (retained for 7 days)
- Manual backups can be triggered from dashboard

Scaling

If your app grows:
1. Upgrade to paid plans for better performance
2. Consider horizontal scaling for backend
3. Use CDN for frontend static assets

Additional Resources

- **Render Documentation:** https://render.com/docs
- **Render Node.js Guide:** https://render.com/docs/deploy-node-express-app
- **Render Static Site Guide:** https://render.com/docs/deploy-static-site
- **Prisma Deployment:** https://www.prisma.io/docs/guides/deployment/deployment-guides/deploying-to-render
- **Workigom Backend Docs:** `backend/README.md`
- **Workigom Frontend Docs:** `src-frontend/README.md`

Support

If you encounter issues not covered in this guide:

1. **Check Render Status:** https://status.render.com
2. **Render Community:** https://community.render.com
3. **Render Support:** https://render.com/support
4. **Project GitHub Issues:** https://github.com/volkanakbulut73/workigom/issues

---

Summary

You’ve successfully deployed Workigom on Render! 🎉

Your deployed services:
- **Backend API:** https://workigom-backend.onrender.com
- **Frontend App:** https://workigom-frontend.onrender.com
- **Database:** Managed PostgreSQL instance

Next steps:
1. Test all functionality thoroughly
2. Monitor logs for any errors

3. Consider setting up monitoring/alerts
4. Plan for production data migration
5. Configure custom domains if needed

Last Updated: November 1, 2024
Version: 1.0.0
Author: Workigom Team