NOMAS E-Commerce Platform - Deployment Guide for Render & Other Platforms

A complete guide to deploy your NOMAS e-commerce platform on Render, Heroku, Railway, Vercel, and other popular hosting platforms.

Table of Contents

Deployment on Render (Recommended)

Render is the easiest and most straightforward option for deploying Node.js applications with databases.

Prerequisites

- GitHub account with your project repository
- Render account (https://render.com)
- Credit card for billing (free tier available)

Step 1: Push Project to GitHub

```
# Initialize git repository (if not already done)
git init

# Add all files
git add .

# Commit
git commit -m "Initial commit - NOMAS e-commerce platform"

# Create repository on GitHub
# Then push:
git remote add origin https://github.com/yourusername/shopping_website.git
git branch -M main
git push -u origin main
```

Step 2: Create Render Account

1. Go to https://render.com
2. Click "Sign Up"
3. Sign up with GitHub (recommended)
4. Authorize Render to access your GitHub account

Step 3: Create PostgreSQL Database on Render

1. In Render dashboard, click "New +"
2. Select "PostgreSQL"
3. Fill in details:
    - **Name**: `nomas-db`

    - **Database**: `nomas_db`
    - **User**: `nomas_user`
    - **Region**: Choose closest to your location
    - **PostgreSQL Version**: 15 (latest)
4. Click "Create Database"
5. Wait for database to be created (2-3 minutes)
6. Copy the **Internal Database URL** (you'll need this)

Step 4: Create Web Service on Render

1. Click "New +"
2. Select "Web Service"
3. Connect your GitHub repository:
    - Click "Connect account" if needed
    - Select `shopping_website` repository
4. Fill in configuration:
    - **Name**: `nomas-app`
    - **Environment**: `Node`
    - **Region**: Same as database
    - **Branch**: `main`
    - **Build Command**: `npm install && npm run build`
    - **Start Command**: `npm run start`
5. Click "Advanced" and add environment variables:

```
DATABASE_URL=postgresql://nomas_user:PASSWORD@HOST:5432/nomas_db
JWT_SECRET=your-super-secret-key-min-32-characters-change-this
VITE_APP_TITLE=NOMAS
VITE_APP_LOGO=/logo.png
NODE_ENV=production
```

1. Click "Create Web Service"

2. Wait for deployment (5-10 minutes)

Step 5: Update Database URL

1. In Render dashboard, go to your PostgreSQL database
2. Copy the **Internal Database URL**
3. Go to your Web Service settings
4. Update `DATABASE_URL` environment variable with the correct URL

Step 6: Run Database Migrations

After deployment, you need to run migrations:

```
# SSH into Render (from your local machine)
# Or use Render Shell in dashboard

# Run migrations
npm run db:push
```

Step 7: Verify Deployment

1. Go to your Render web service
2. Click the URL to open your application
3. Test the following:
    - Homepage loads
    - Can log in
    - Can browse products
    - Can add to cart
    - Can checkout

Your app is now live! 🎉

Deployment on Railway

Railway is another excellent option with a generous free tier.

Prerequisites

- GitHub account
- Railway account (https://railway.app)

Step 1: Push to GitHub

Same as Render (see above)

Step 2: Create Railway Account

1. Go to https://railway.app
2. Sign up with GitHub
3. Authorize Railway

Step 3: Create New Project

1. Click "New Project"
2. Select "Deploy from GitHub repo"
3. Select your `shopping_website` repository
4. Click "Deploy Now"

Step 4: Add PostgreSQL Database

1. In Railway dashboard, click "Add"
2. Select "PostgreSQL"
3. Wait for database to be created

Step 5: Configure Environment Variables

1. In Railway, go to your project

2. Click on the Node.js service
3. Go to “Variables” tab
4. Add these variables:

```
DATABASE_URL=postgresql://postgres:PASSWORD@HOST:5432/railway
JWT_SECRET=your-super-secret-key-min-32-characters
VITE_APP_TITLE=NOMAS
NODE_ENV=production
```

1. Click “Deploy” to redeploy with new variables

Step 6: Run Migrations

```
# Use Railway CLI or SSH
railway run npm run db:push
```

Step 7: Access Your App

Railway will provide a URL. Click it to access your deployed app.

Deployment on Heroku

Heroku is being phased out but still works. Here’s how:

Prerequisites

- Heroku account (https://www.heroku.com)
- Heroku CLI installed

Step 1: Install Heroku CLI

```
# macOS
brew tap heroku/brew && brew install heroku

# Windows
# Download from https://devcenter.heroku.com/articles/heroku-cli

# Verify installation
heroku --version
```

Step 2: Login to Heroku

```
heroku login
```

Step 3: Create Heroku App

```
heroku create nomas-app
```

Step 4: Add PostgreSQL Database

```
heroku addons:create heroku-postgresql:hobby-dev
```

Step 5: Set Environment Variables

```
heroku config:set JWT_SECRET=your-super-secret-key-min-32-characters
heroku config:set VITE_APP_TITLE=NOMAS
heroku config:set NODE_ENV=production
```

Step 6: Deploy

```
git push heroku main
```

Step 7: Run Migrations

```
heroku run npm run db:push
```

Step 8: Open App

```
heroku open
```

Deployment on Vercel + Serverless

Vercel is best for frontend, but you can use it with a separate backend.

Frontend on Vercel

```
# Install Vercel CLI
npm install -g vercel

# Deploy
vercel

# Follow prompts and select your project
```

Backend on Render or Railway

Deploy backend separately using Render or Railway guides above.

Update Frontend API Endpoint

In `client/src/lib/trpc.ts`, update the API URL:

```
const apiUrl = process.env.VITE_API_URL || 'https://your-backend-url.com';
```

Deployment on DigitalOcean

DigitalOcean provides full control and is great for production.

Prerequisites

- DigitalOcean account
- SSH knowledge
- Domain name (optional but recommended)

Step 1: Create Droplet

1. Go to https://www.digitalocean.com
2. Click "Create" → "Droplet"
3. Select:
    - **Image**: Ubuntu 22.04 LTS
    - **Size**: $6/month (2GB RAM, 1 CPU)
    - **Region**: Closest to your users
    - **Authentication**: SSH key (recommended)
4. Click "Create Droplet"
5. Wait for droplet to be created

Step 2: SSH into Droplet

```
ssh root@your_droplet_ip
```

Step 3: Update System

```
apt update && apt upgrade -y
```

Step 4: Install Node.js

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
apt install -y nodejs
```

Step 5: Install PostgreSQL

```
apt install -y postgresql postgresql-contrib
```

Step 6: Create Database

```
sudo -u postgres psql

# In PostgreSQL prompt:
CREATE DATABASE nomas_db;
CREATE USER nomas_user WITH PASSWORD 'strong_password_here';
ALTER ROLE nomas_user SET client_encoding TO 'utf8';
ALTER ROLE nomas_user SET default_transaction_isolation TO 'read committed';
ALTER ROLE nomas_user SET default_transaction_deferrable TO on;
ALTER ROLE nomas_user SET timezone TO 'UTC';
GRANT ALL PRIVILEGES ON DATABASE nomas_db TO nomas_user;
\q
```

Step 7: Clone Project

```
cd /var/www
git clone https://github.com/yourusername/shopping_website.git
cd shopping_website
npm install
```

Step 8: Configure Environment

```
nano .env
```

Add:

```
DATABASE_URL=postgresql://nomas_user:password@localhost:5432/nomas_db
JWT_SECRET=your-secret-key
VITE_APP_TITLE=NOMAS
NODE_ENV=production
```

Step 9: Run Migrations

```
npm run db:push
```

Step 10: Build Application

```
npm run build
```

Step 11: Install Nginx

```
apt install -y nginx
```

Step 12: Configure Nginx

```
nano /etc/nginx/sites-available/nomas
```

Add:

```
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Enable site:

```
ln -s /etc/nginx/sites-available/nomas /etc/nginx/sites-enabled/
nginx -t
systemctl restart nginx
```

Step 13: Install PM2

```
npm install -g pm2
cd /var/www/shopping_website
pm2 start npm --name "nomas" -- start
pm2 startup
pm2 save
```

Step 14: Set Up SSL (Let’s Encrypt)

```
apt install -y certbot python3-certbot-nginx
certbot --nginx -d yourdomain.com -d www.yourdomain.com
```

Step 15: Verify

Visit `https://yourdomain.com` in your browser.

Deployment on AWS

AWS is powerful but more complex. Here’s a simplified guide.

Option 1: Elastic Beanstalk (Easiest)

1. Go to AWS Console
2. Search for “Elastic Beanstalk”
3. Click “Create application”
4. Fill in details:
    - **Application name**: nomas
    - **Environment**: Node.js
    - **Platform**: Node.js 18
5. Upload your code as ZIP
6. Add RDS PostgreSQL database
7. Set environment variables
8. Deploy

Option 2: EC2 + RDS (More Control)

1. Create EC2 instance (Ubuntu 22.04)
2. Create RDS PostgreSQL database

3. SSH into EC2
4. Follow DigitalOcean steps above (similar process)

Deployment on Netlify + Backend

Netlify is great for frontend, but backend needs separate hosting.

Deploy Frontend on Netlify

1. Go to https://netlify.com
2. Click "Add new site" → "Import an existing project"
3. Connect GitHub
4. Select repository
5. Build settings:
   - **Build command**: `npm run build`
   - **Publish directory**: `dist`
6. Click "Deploy"

Deploy Backend Separately

Use Render, Railway, or DigitalOcean (see above)

Connect Frontend to Backend

Update API endpoint in frontend code to point to your backend URL.

Post-Deployment Configuration

1. Set Up Custom Domain

On Render:

1. Go to Web Service settings
2. Click "Custom Domain"
3. Add your domain
4. Update DNS records at your domain provider

On Railway:

1. Go to project settings
2. Add custom domain
3. Update DNS

On DigitalOcean:

1. Update DNS at domain provider to point to droplet IP
2. Configure Nginx (already done above)

2. Set Up SSL Certificate

Most platforms (Render, Railway, Heroku) provide free SSL automatically.

For DigitalOcean:

```
certbot --nginx -d yourdomain.com
```

3. Configure Email Notifications

Add email configuration to `.env`:

```
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-email@gmail.com
SMTP_PASSWORD=your-app-password
```

4. Set Up Backups

Render/Railway:

- Automatic daily backups included

DigitalOcean:

```
# Create backup script
#!/bin/bash
pg_dump -U nomas_user nomas_db > /backups/nomas_$(date +%Y%m%d).sql
```

Monitoring & Maintenance

Monitor Application

Render/Railway:

- Built-in monitoring in dashboard
- View logs in real-time

DigitalOcean:

```
# SSH into droplet
pm2 logs nomas
pm2 status
```

Update Application

```
# Pull latest changes
git pull origin main

# Rebuild
npm run build

# Restart
pm2 restart nomas
# or
heroku deploy
# or push to Render/Railway
```

Database Maintenance

```
# Backup database
pg_dump -U nomas_user nomas_db > backup.sql

# Optimize database
VACUUM ANALYZE;
```

Troubleshooting

"Cannot connect to database"

Solution:

1. Verify DATABASE_URL is correct
2. Check database credentials
3. Ensure database is running
4. Check firewall rules allow connection

“Build failed”

Solution:

1. Check build logs
2. Verify all dependencies are installed
3. Run `npm install` locally to test
4. Check Node.js version compatibility

“Application crashes after deployment”

Solution:

1. Check application logs
2. Verify environment variables are set
3. Check database migrations ran successfully
4. Verify all required packages are in package.json

“Slow performance”

Solution:

1. Check server resources (CPU, RAM)
2. Enable database query caching
3. Optimize images
4. Use CDN for static files
5. Enable gzip compression

“SSL certificate errors”

Solution:

1. Verify domain DNS is configured correctly
2. Wait 24-48 hours for DNS propagation
3. Renew certificate:

```
certbot renew
```

Comparison Table

Platform	Cost	Ease	Database	SSL	Best For
Render	$7+/mo	⭐⭐⭐⭐⭐	Included	Free	Beginners
Railway	$5+/mo	⭐⭐⭐⭐⭐	Included	Free	Beginners
Heroku	$7+/mo	⭐⭐⭐⭐	Paid addon	Free	Quick deploy
DigitalOcean	$6+/mo	⭐⭐⭐	Self-managed	Free	Full control
AWS	$1+/mo	⭐⭐	Included	Free	Enterprise
Vercel	Free	⭐⭐⭐⭐⭐	Separate	Free	Frontend only

Quick Start Commands by Platform

Render

```
git push origin main
# Auto-deploys from GitHub
```

Railway

```
railway up
```

Heroku

```
git push heroku main
```

DigitalOcean

```
git pull origin main
npm run build
pm2 restart nomas
```

Security Checklist

- [ ] Set strong JWT_SECRET (min 32 characters)
- [ ] Enable HTTPS/SSL
- [ ] Configure firewall rules
- [ ] Set up database backups
- [ ] Enable application monitoring
- [ ] Configure rate limiting
- [ ] Set up error logging
- [ ] Enable CORS properly
- [ ] Rotate secrets regularly
- [ ] Monitor for security updates

Support & Resources

- **Render Docs**: https://render.com/docs
- **Railway Docs**: https://docs.railway.app
- **Heroku Docs**: https://devcenter.heroku.com

- **DigitalOcean Docs**: https://docs.digitalocean.com
- **AWS Docs**: https://docs.aws.amazon.com

Version: 1.0.0
Last Updated: November 2025
Recommended Platform: Render (easiest and most reliable)