Deploying Your E-commerce Application on Render.com

This guide walks you through the process of deploying your e-commerce application on Render.com using the Docker-based approach.

Prerequisites

- A Render.com account (Sign up here)
- Your code pushed to a GitHub or GitLab repository
- Docker installed locally for testing (optional)

Step 1: Prepare Your Repository

Ensure your repository has the following structure:

```
project-root/
├── server/                 # Backend code + seed data
│   ├── data/               # JSON seed files
│   └── app.js              # Main app file
├── client/                 # Frontend code
├── docker/                 # Docker configuration
│   ├── Dockerfile.client
│   ├── Dockerfile.server
│   └── nginx.conf
├── docker-compose.yml      # For local testing
└── render.yaml             # Render configuration
```

Step 2: Create a MongoDB Database on Render

1. Sign in to your Render.com dashboard
2. Click on **New** and select **PostgreSQL**
3. Wait - we actually need MongoDB for this project, but Render doesn't offer managed MongoDB directly
4. We'll need to use an external MongoDB service like MongoDB Atlas

Setting up MongoDB Atlas (Alternative)

1. Create a free account on MongoDB Atlas
2. Create a new cluster (M0 Free Tier is sufficient for testing)
3. Set up a database user with appropriate credentials
4. Configure network access to allow connections from Render.com (whitelist `0.0.0.0/0` for now)
5. Get your connection string, which will look like:

```
mongodb+srv://username:password@cluster.mongodb.net/ecommerce?retryWrites=true&w=majority
```

Step 3: Deploy Your Backend API Service

1. In your Render.com dashboard, click on **New** and select **Web Service**
2. Connect your GitHub or GitLab repository
3. Configure the service:
   - **Name**: `ecommerce-api`
   - **Environment**: Docker
   - **Dockerfile Path**: docker/Dockerfile.server
   - **Docker Context Directory**: . (root of your repo)
   - **Region**: Choose the one closest to your target audience
   - **Branch**: main (or your deployment branch)
   - **Plan**: Starter or higher based on your needs
4. Add environment variables:
   - `NODE_ENV`: production
   - `PORT`: 3000 (Render automatically assigns an internal port)
   - `JWT_SECRET`: [generate a secure random string]
   - `SESSION_SECRET`: [generate a secure random string]
   - `MONGODB_URI`: [your MongoDB Atlas connection string]

- `SEED_DATABASE`: true (for first deployment, set to false after data is seeded)
- `CLIENT_URL`: [leave blank for now, we'll update after frontend deployment]

5. Click **Create Web Service**

Step 4: Deploy Your Frontend Service

1. In your Render.com dashboard, click on **New** and select **Web Service**
2. Connect your GitHub or GitLab repository (same as before)
3. Configure the service:
   - **Name**: `ecommerce-client`
   - **Environment**: Docker
   - **Dockerfile Path**: docker/Dockerfile.client
   - **Docker Context Directory**: . (root of your repo)
   - **Region**: Same as your backend service
   - **Branch**: main (or your deployment branch)
   - **Plan**: Starter or higher based on your needs
4. Add environment variables:
   - `REACT_APP_API_URL`: [URL of your backend service, e.g., https://ecommerce-api.onrender.com]
   - `REACT_APP_NODE_ENV`: production
5. Click **Create Web Service**

Step 5: Update Cross-Service References

1. Wait for both services to deploy successfully
2. Go to your backend service (ecommerce-api) in the Render dashboard
3. Update the `CLIENT_URL` environment variable with the URL of your frontend service
4. Click **Save Changes** and wait for the service to redeploy

Step 6: Initialize the Database

For the first deployment, we've set `SEED_DATABASE=true` to populate the database with initial data. Once the seeding is complete, you should:

1. Go to your backend service (ecommerce-api)
2. Change the `SEED_DATABASE` environment variable to `false`
3. Save changes to prevent reseeding on future deployments

Step 7: Verify Your Deployment

1. Visit your frontend URL (https://ecommerce-client.onrender.com)
2. Test key functionality:
   - Browse products
   - Create a user account
   - Add items to cart
   - Complete a checkout process
   - Leave a review

Step 8: Set Up Custom Domain (Optional)

If you have a custom domain:

1. Go to your frontend service settings
2. Click on **Custom Domain**
3. Follow the instructions to set up your domain with Render

Step 9: Monitoring and Maintenance

1. Set up alerts in Render for both services
2. Monitor performance and scale up if needed
3. Set up regular backups for your MongoDB database

Troubleshooting

Service Fails to Deploy

- Check the build logs for specific errors
- Ensure your Dockerfiles are correctly configured
- Verify that all required environment variables are set

Database Connection Issues

- Check if your MongoDB URI is correct
- Ensure that the IP whitelist in MongoDB Atlas includes Render's IPs

Frontend Cannot Connect to Backend

- Verify the `REACT_APP_API_URL` is correctly set
- Check CORS configuration in your backend code

Next Steps

- Set up CI/CD for automatic deployments
- Implement SSL for all services
- Configure rate limiting and security features
- Set up analytics to track user behavior

For more help, refer to Render's documentation or join their community Discord server.