Hosting Guide: Node.js Express + Next.js on Vercel by SKILLHUB

Overview

This guide covers deploying a full-stack application with Node.js Express backend and Next.js frontend on Vercel.

● Node.js Express JavaScript Server Setup

1. Update the .env file

```
 # Use Live MongoDB URL
MONGO_URL=your_live_mongodb_connection_string

# Set up NODE_ENV
NODE_ENV=development
```

2. Cookie Security Configuration

Configure secure cookies based on environment:

```
 // In your server configuration
res.cookie('token', token,  {
  secure: process.env.NODE_ENV === 'production', // false in development, true in production
  httpOnly: true,
  sameSite: 'strict'
});
```

3. Export App Variable

In your index/server entry file:

```
 // index.js or server.js
const app = express();

// Your middleware and routes here

// For local development
```

```
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});

// Export for Vercel serverless functions
module.exports = app;
```

Next.js TypeScript Frontend Setup

1. Update the .env file

```
 # Environment setting
NEXT_PUBLIC_ENV=development

# API URLs
NEXT_PUBLIC_API_LIVE=https://your-server-url.vercel.app
NEXT_PUBLIC_API_LOCAL=http://localhost:5000
```

2. API Configuration

Ensure all API calls use environment variables with Redux Toolkit:

```
 // Example API configuration using Redux Toolkit
import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react';

export const authApi = createApi({
  reducerPath: "authApi",
  baseQuery: fetchBaseQuery({
    baseUrl: `${process.env.NEXT_PUBLIC_ENV === 'production'
      ? process.env.NEXT_PUBLIC_API_LIVE
      : process.env.NEXT_PUBLIC_API_LOCAL}/api/auth`,
    credentials: "include"
  }),
  endpoints: (builder) => ({
    // Define your endpoints here
    login: builder.mutation({
      query: (credentials) => ({
        url: '/login',
        method: 'POST',
        body: credentials
      })
    }),
    register: builder.mutation({
      query: (userData) => ({
        url: '/register',
        method: 'POST',
        body: userData
      })
    }),
    logout: builder.mutation({
      query: () => ({
        url: '/logout',
        method: 'POST'
      })
    })
  })
});

// Export hooks for usage in components
export const { useLoginMutation, useRegisterMutation, useLogoutMutation } = authApi;
```

🚀 Deployment Steps

Step 1: Push Code to GitHub

```
 # Make sure you're in the root folder (contains both client and server folders)
git add .
git commit -m "Ready for deployment"
git push origin master
```

Step 2: Deploy Server to Vercel

1. Go to Vercel Dashboard
2. Click "Add New Project"
3. Select your repository from GitHub
4. Change project name to identify it as the server (e.g., `myapp-server`)
5. Click "Edit" next to "Root Directory"
6. Select the server folder
7. Add environment variables:
   - `MONGO_URL`: Your live MongoDB connection string
   - `NODE_ENV`: `production`
8. Click "Deploy"

Step 3: Update Next.js .env

After server deployment, update the frontend .env:

```
NEXT_PUBLIC_API_URL=https://your-server-project-name.vercel.app
```

Step 4: Deploy Client to Vercel

1. Go to Vercel Dashboard
2. Click "Add New Project"
3. Select the same repository from GitHub
4. Change project name to identify it as the client (e.g., `myapp-client`)
5. Click "Edit" next to "Root Directory"
6. Select the client folder
7. Add environment variables:
    - `NEXT_PUBLIC_API_URL`: Your live server URL
8. Click "Deploy"

Step 5: Update Server CORS

In your server index file, update CORS settings:

```
const cors = require('cors');

app.use(cors({
  origin: process.env.NODE_ENV === 'production'
    ? 'https://your-client-project-name.vercel.app'
    : 'http://localhost:3000',
  credentials: true
}));
```

Step 6: Final Push

```
# Push the CORS updates
git add .
git commit -m "Update CORS for production"
git push origin main
```

✅ Testing

1. Wait for both deployments to complete
2. Visit your client URL: `https://your-client-project-name.vercel.app`
3. Test all functionality:
   - User authentication
   - API calls
   - Database operations
   - Cookie handling

🛠️ Troubleshooting

Common Issues:

- **CORS errors**: Double-check origin configuration
- **Database connection**: Verify MONGO_URL is correct
- **Environment variables**: Ensure they're properly set in Vercel
- **Build failures**: Check Vercel build logs for specific errors

Debugging Tips:

- Use Vercel Functions Logs to monitor server errors
- Check browser console for frontend issues
- Verify environment variables are being loaded correctly

📝 Checklist

- [ ] Server .env configured with live MONGO_URL
- [ ] NODE_ENV set to production
- [ ] Cookie security flags configured
- [ ] App variable exported in server entry
- [ ] Client .env uses live backend URL
- [ ] All API calls use environment variables
- [ ] Code pushed to GitHub from root folder
- [ ] Server deployed to Vercel with correct folder

- [ ] Client deployed to Vercel with correct folder
- [ ] CORS updated for production domain
- [ ] Final push completed
- [ ] Application tested in production

🎉 **Your full-stack application is now live!**