Deploying an Express API to Vercel: A Step-by-Step Guide for Students

This blog post explains how to deploy an Express API to Vercel. While Vercel isn't ideally suited for long-running Express APIs, this guide shows how it can be done for specific use cases. Remember to review Vercel's documentation on limitations before proceeding.

I. Understanding the Limitations

Vercel is primarily designed for serverless functions and applications like Next.js. Deploying an Express API presents challenges because Express typically involves long-lived connections, which aren't optimal for Vercel's serverless architecture. Vercel's documentation clearly outlines these limitations. You should carefully consider if Vercel is the right platform for your Express API; it's generally better suited for applications that don't require persistent connections (like websockets or server-sent events).

II. Setting up Your Express API

This section assumes you already have a basic Express API built. If not, you'll need to create one first. Numerous tutorials are available online to help you build a simple Express API. The example uses TypeScript but the concepts are applicable to JavaScript projects.

Project Structure: Organize your project so your Express app's main file (e.g., `app.ts` or `app.js`) is easily importable.

Exporting the App: Your main file must export the Express app instance. This allows Vercel's deployment process to access and run your API.

III. Preparing for Vercel Deployment

This section details the modifications needed to make your Express API compatible with Vercel's serverless environment.

API Folder: Create an `API` folder to encapsulate your Express app. Within this folder, create an `index.ts` (or `index.js`) file that imports and exports your Express app.

Public Folder: Create a `public` folder. Even if your API doesn't serve static files, Vercel expects this folder to exist. A `.gitkeep` file can be added to ensure Git tracks this empty folder.

Vercel Configuration (`vercel.json`): This file is crucial for configuring how Vercel handles your API. You'll need to add rewrite rules to redirect all incoming requests to your `API` folder's entry point. The configuration will look similar to this (adjust paths as needed):

```json
{
  "rewrites": [
    {
      "source": "/(.)",
      "destination": "/api"
    }
  ],
  "builds": [
    {
      "src": "vercel-build",
      "use": "@vercel/static"
    }
  ],
  "functions": {
    "api": {
      "runtime": "nodejs16.x",  //Or your node version
      "handler": "api/index.js" //Or api/index.ts
    }
  }
}

```

Build Command Override: Add a `vercel.build` property to your `vercel.json` that essentially performs a no-op. This prevents Vercel's default build process from interfering with your existing API setup. A simple `echo "Hello"` command will suffice.

IV. Deployment to Vercel

The final step involves deploying your prepared API to Vercel.

Vercel CLI: Use the Vercel CLI (`vercel`) to create and deploy your project. Follow the CLI prompts to link your project to your Vercel account and specify the appropriate settings. Ensure the build command is overridden as described in the previous section.

Verification: After deployment, test your API to confirm it functions correctly. All requests should be routed to the entry point defined in the `vercel.json` file, which in turn launches your Express app.

V. Conclusion

Deploying an Express API to Vercel is possible, but not ideal for applications requiring long-lived connections. This guide provides a detailed walkthrough. Always consult Vercel's official documentation for the most accurate and up-to-date information. Remember to carefully consider the limitations before proceeding.