how to deploy a project made with mern frontend in react and backend with express js on vercel I am a beginners help me

Here's the beginner-friendly way to deploy a **MERN-style project (React frontend + Express backend)** on **Vercel**. I'll show **two options**:

- **Option A (recommended for beginners):** Deploy **frontend on Vercel** + deploy **backend also on Vercel** (as serverless).
- **Option B:** Deploy **frontend on Vercel** + deploy **backend somewhere else** (Render/Railway/Fly), if your backend needs "always-on server" behavior.

Vercel runs your backend as **serverless functions** (not a permanently running server), so some things like WebSockets need extra thought. Vercel

---

Option A: Frontend + Express backend both on Vercel (most common)

0) Typical folder structure

Example:

```perl
my-project/
  client/   (React)
  server/   (Express)
```

Push the whole project to **GitHub** first (easiest workflow).

---

1) Deploy the Express backend to Vercel

Step 1: Make Express work in serverless mode

In Vercel serverless, **don't use** `app.listen(...)`.

Inside your **server** folder, create this structure:

```pgsql
```

```
server/
  api/
    index.js
  vercel.json
  package.json
```

server/api/index.js (example)

```js
const express = require("express");
const cors = require("cors");

const app = express();

app.use(cors({
  origin: ["http://localhost:5173", "https://YOUR-FRONTEND.vercel.app"],
  credentials: true,
}));
app.use(express.json());

// test route
app.get("/api/health", (req, res) => {
  res.json({ ok: true });
});

// your routes...
// app.use("/api/users", require("./routes/users"));

module.exports = app;
```

Step 2: Add `vercel.json`

server/vercel.json

```json
{
  "version": 2,
  "rewrites": [{ "source": "/(.*)", "destination": "/api" }]
}
```

This is the official/simple approach: route everything to your Express function. Vercel

Step 3: Environment variables (MongoDB, JWT, etc.)

In Vercel dashboard → your backend project → **Settings** → **Environment Variables**
Add things like:

- `MONGODB_URI`
- `JWT_SECRET`
- `NODE_ENV=production`

Vercel supports env vars at project/team level, and you can also pull them locally with `vercel env pull`. Vercel

Step 4: Deploy backend

You can deploy either way:

A) Using Vercel Dashboard

- New Project → Import GitHub repo
- Set **Root Directory** = `server`
- Deploy

B) Using CLI

From the `server` folder:

```bash
npm i -g vercel
vercel login
vercel
```

(Vercel CLI + deploy flow is also described in their Express guide.) Vercel

After deploying, you'll get a backend URL like:
`https://YOUR-BACKEND.vercel.app`

Test it:
`https://YOUR-BACKEND.vercel.app/api/health`

---

2) Deploy the React frontend to Vercel

Step 1: Use an environment variable for API base URL

In React, don't hardcode localhost in production.

Example (Vite):

```js
const API = import.meta.env.VITE_API_URL;
// fetch(`${API}/api/health`)
```

Example (Create React App):

```js
const API = process.env.REACT_APP_API_URL;
```

Step 2: Deploy frontend

In Vercel:

- New Project → Import same repo
- Set **Root Directory** = `client`
- Vercel auto-detects React settings; CRA is supported directly. Vercel

Step 3: Add frontend env var

Frontend project → Settings → Environment Variables:

- Vite: `VITE_API_URL = https://YOUR-BACKEND.vercel.app`
- CRA: `REACT_APP_API_URL = https://YOUR-BACKEND.vercel.app`

Redeploy frontend after adding env vars.

Common beginner issues (quick fixes)

1) CORS error

Make sure your backend `cors()` allows your **frontend Vercel domain** (and localhost for dev).

2) Works locally, fails on Vercel

Usually:

- missing env vars in Vercel
- using `app.listen()` (should be removed)
- connecting to MongoDB without serverless-friendly connection handling (serverless can open many connections) Vercel

3) 404 for API routes

Confirm:

- backend deployed from the correct **Root Directory** (`server`)
- `server/vercel.json` exists
- your Express routes match what you're calling (e.g., `/api/...`)

Option B (simpler backend): Deploy backend elsewhere

If your backend needs:

- WebSockets
- long-running jobs
- always-on server behavior

...deploy the backend to Render/Railway/Fly and keep React on Vercel. Vercel itself notes serverless limitations like execution constraints & websocket concerns. Vercel

---

If you tell me your setup (CRA or Vite?) and your folder structure (`client/server` or something else), I'll give you the exact build settings + exact file paths for your project.