Here’s a comprehensive, step-by-step guide to everything we’ve done—from local setup through CI/CD—organized by **Why**, **Where**, **How**, and **Who**, with all the gotchas and solutions we encountered along the way. After the CI/CD section we’ll list future-work bullet points.

Summary

We built a full-stack Copy-Trading App with a Node/Express/MongoDB backend and a React frontend, deployed the backend on Render, the frontend on Vercel, and established a basic GitHub Actions CI for the backend. Major hurdles included JSON syntax errors in `package.json`, MongoDB SRV-DNS resolution failures, Axios misconfigurations, missing React components, and environment-variable pitfalls—each solved with targeted fixes. Roles: **DevOps Engineer** (infrastructure, deployment, CI), **Developer** (coding frontend/backend), **Tester** (manual API/UI validation).

---

1. Prerequisites & Environment Setup

Why

Ensure all team members have the same tools and accounts so development, testing, and deployment run smoothly.

Where

On your local machine (Windows, macOS, or Linux).

How

1. **Install Node.js & npm** from nodejs.org
2. **Install Git** and sign in to GitHub (`git config --global user.name/email`)
3. **Install VS Code** for editing code
4. **Create accounts on** GitHub, MongoDB Atlas, Postman, Render, and Vercel

Who

- **DevOps Engineer**: Installs and configures system-level tools; creates Render/Vercel projects.
- **Developer**: Verifies `node -v`, `npm -v`, `git --version`.
- **Tester**: Confirms Postman can launch and hit endpoints once they exist.

---

2. Repository Setup & Folder Structure

Why

Separate backend and frontend code for clarity, enable independent deployment, and enforce clean architecture.

Where

Inside a parent folder called `MT`.

How

```
cd ~/Documents
mkdir MT
cd MT
mkdir backend frontend
git init
git remote add origin https://github.com/talha0pse/MT.git
```

Who

- **DevOps Engineer**: Creates the GitHub repo and sets remotes.
- **Developer**: Clones `MT` and verifies `.git/config`.

---

3. Backend Development

Why

Provide a secure REST API for user auth and trade CRUD operations.

Where

`MT/backend`

How

- Create `.env` (with a non-SRV MongoDB URI).
- Write `src/server.js`, `src/app.js`, route/controllers, and JWT middleware.
- Use `testMongo.js` to verify Atlas connectivity.

```
// Example: server.js
const app = require('./app');
const mongoose = require('mongoose');
require('dotenv').config();

mongoose.connect(process.env.MONGO_URI, { useNewUrlParser: true, useUnifiedTopology:
true })
  .then(() => app.listen(process.env.PORT||5000, () => console.log('🚀 Server
running')))
  .catch(err => console.error(err));
```

Who

- **Developer**: Writes code.
- **DevOps Engineer**: Supplies Atlas connection string and sets `.env`.
- **Tester**: Runs `node testMongo.js` and `npm start`, checks console logs.

Problem: DNS Error with SRV String

We saw:

```
Error: querySrv ENOTFOUND _mongodb._tcp.cluster0.mongodb.net
```

Solution: Use the standard `mongodb://` URI from Atlas → fixed DNS resolution 【cite】turn2search1【】.

---

4. Local Backend Testing

Why

Verify that the API and DB connection work before integrating with the frontend.

Where

Terminal in `MT/backend`

How

```
npm install
node testMongo.js      # expect "✅ Pinged your deployment"
npm start              # expect "🚀 Server running on port 5000"
curl http://localhost:5000/health
```

Who

- **Tester**: Runs these commands and reports any errors.
- **Developer**: Fixes code/config as needed.

---

5. Frontend Development

Why

Provide a user interface to interact with our API.

Where

`MT/frontend`

How

1. Bootstrap with CRA:

   ```
   cd MT
   npx create-react-app frontend
   cd frontend
   npm install axios react-router-dom@6
   ```

2. Replace `src/api.js`, `src/App.js`, add `src/TradeApp.js`, `src/pages/Login.js`, `src/pages/Register.js`, and `src/components/PrivateRoute.js`.

```
// src/api.js
import axios from 'axios';
const api = axios.create({ baseURL: process.env.REACT_APP_API_URL });
api.interceptors.request.use(cfg => {
  const token = localStorage.getItem('token');
  if (token) cfg.headers.Authorization = `Bearer ${token}`;
  return cfg;
});
export default api;
```

Who

- **Developer**: Writes React components and config.
- **Tester**: Verifies UI appears, buttons trigger the right network calls.

Problems & Fixes

- **Missing Axios** → installed with `npm install axios`.
- **TradeApp import error** → created `src/TradeApp.js` and updated imports.
- **ERR_CONNECTION_REFUSED** → realized backend must be running locally (`npm start`).
- **Env-var misreads** → cleaned `.env.local` to a single line `REACT_APP_API_URL=http://localhost:5000` cite turn0search5 .

---

6. Local Integration Testing

Why

Ensure frontend and backend communicate properly.

Where

Browser at `localhost:3000` and `localhost:5000`

How

- Run **backend**: `cd MT/backend && npm start`
- Run **frontend**: `cd MT/frontend && npm start`
- Use browser DevTools to confirm:

```
API URL: http://localhost:5000        # via console.log in api.js
GET /api/trades → 200 OK (JSON array)
```

---

7. Deployment: Backend on Render

Why

Host the API online so remote clients can use it.

Where

Render Dashboard → New Web Service

How

1. Connect GitHub `MT` repo.
2. Service config:
    - **Root Directory:** `backend` cite turn0search12
    - **Build:** `npm install`
    - **Start:** `npm start`
3. Add environment vars **MONGO_URI** and **JWT_SECRET** (no PORT).
4. Deploy and verify at `https://copy-trading-backend-ksfs.onrender.com/health`.

Who

- **DevOps Engineer**: Configures Render, sets secrets.

- **Tester**: Hits `/health` and `/api/trades` remotely.

Gotcha: Internal Port vs Public URL

Render uses an internal port (e.g. 10000) but proxies HTTPS on 443—public requests must use the `https://...render.com` domain cite turn0search21 .

---

8. Deployment: Frontend on Vercel

Why

Host the UI so anyone can access the app.

Where

Vercel Dashboard → New Project

How

1. Import repo **talha0pse/MT**, set **Root Directory** to `frontend`.
2. Ensure **Framework Preset** = “Create React App”.
3. **Build Command:** `npm install && npm run build`
4. **Output Directory:** `build`
5. Add env var `REACT_APP_API_URL=https://copy-trading-backend-ksfs.onrender.com` cite turn0search4 .
6. Deploy and verify at `https://<your-vercel-url>/` .

Who

- **DevOps Engineer**: Sets env vars, triggers deploy.
- **Tester**: Verifies UI loads and fetches real data.

---

9. CI/CD: Backend with GitHub Actions

Why

Automatically verify that the backend installs and “tests” correctly on every push.

Where

`.github/workflows/backend-ci.yml` in `MT` repo

How

1. **Add placeholder test script** to `backend/package.json`:

   ```
   "scripts": {
     "start": "node src/server.js",
     "test":  "echo \"No tests yet – placeholder\" && exit 0"
   }
   ```

   Fix JSON syntax by adding a comma after the start script! cite turn0search10
2. **Create** `.github/workflows/backend-ci.yml`:

   ```
   name: Backend CI
   on: [push,pull_request,workflow_dispatch]
   ```

```yaml
jobs:
  build-and-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3                # cite turn0search2
      - uses: actions/setup-node@v3              # cite turn0search1
        with: node-version: '18'
      - run: npm ci
        working-directory: ./backend
      - run: npm test
        working-directory: ./backend
```

3. Commit & push.
4. Observe the GitHub Actions tab for **Backend CI** success.

Who

- **DevOps Engineer**: Writes the workflow, configures actions.
- **Developer**: Ensures placeholder tests pass, expands to real tests later.
- **Tester**: Confirms CI runs on push and PR.

Future Work (Post-CI/CD)

- Write real **unit** and **integration** tests (Mocha/Chai, Jest/RTL).
- Add **linting** and **security scans** (ESLint, Trivy).
- Implement **GitHub Actions** for frontend (build + deploy via Vercel CLI).
- Integrate **Sentry** for error monitoring in backend and UI.
- Containerize services with **Docker** and add **docker-compose** for local orchestration.
- Harden security: HTTPS enforcement, JWT refresh flow, rate limiting.

Key Pitfalls to Watch

- **JSON Syntax**: every property (e.g. in `package.json`) needs a comma between entries cite turn0search10.
- **Env-var Loading**: CRA only reads `REACT_APP_*` from `.env*` at startup—always restart after edits cite turn0search5.
- **Local vs. Remote URLs**: `localhost:5000` ≠ `render.com` domain—don't confuse internal ports cite turn0search21.
- **Missing Dependencies**: always run `npm install` after adding libs (e.g. `axios`, `react-router-dom`).
- **Route Protection**: ensure `PrivateRoute` wraps your main UI, or unauthenticated users will see empty pages cite turn0search7.
- **CI Cache**: use `npm ci` for reproducible installs; avoid `npm install` in CI for lockfile consistency cite turn0search19.