Below is a **complete, from-scratch guide** for the MT Copy-Trading App, covering **every step**, **file**, and **command** needed to build, test, deploy, and integrate CI/CD—all tailored so a non-technical person can follow along without missing anything.

This document is structured into major sections with clear **Why**, **Where**, **How**, and **Who** descriptions. Every code file is shown in its final, working form, with notes on changes made and problems solved. At the end, you'll find a link to Dillinger for direct PDF conversion.

Summary

We built a full-stack Copy-Trading App featuring a **Node.js + Express** backend connected to **MongoDB Atlas**, a **React** frontend created with Create React App, hosted on **Render** (backend) and **Vercel** (frontend), and set up a basic **GitHub Actions** CI pipeline. Along the way, we tackled DNS issues with SRV URIs, fixed JSON syntax errors, configured environment variables properly, and implemented JWT-based authentication and protected routes. Roles covered include **DevOps Engineer**, **Developer**, and **Tester**.

---

1. Prerequisites & Environment Setup

Why

Ensure consistent tools and access for development, testing, and deployment.

Where

On your local machine (Windows/macOS/Linux).

How

1. **Install Node.js & npm** from the official site cite turn0search0 .
2. **Install Git** and configure your GitHub account (`git config --global user.name` , `git config --global user.email`).
3. **Install VS Code** or another code editor.
4. **Sign up** for:
   - **GitHub** (repo hosting)
   - **MongoDB Atlas** (database)
   - **Postman** (API testing)
   - **Render** (backend hosting) cite turn0search4
   - **Vercel** (frontend hosting) cite turn0search5

Who

- **DevOps Engineer**: Installs global tools and creates cloud accounts.
- **Developer**: Verifies versions with `node -v` , `npm -v` , `git --version` .
- **Tester**: Confirms Postman and browser are ready.

---

2. Project Initialization & GitHub Setup

Why

Separate frontend and backend, enable version control and collaborative work.

Where

Inside a new parent folder named `MT`.

How

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

Who

- **Developer**: Runs commands and confirms `.git/config`.
- **DevOps Engineer**: Creates the `MT` repo on GitHub.

---

3. Backend Development

3.1 Folder Structure

```
backend/
├── .env
├── package.json
├── testMongo.js
└── src/
    ├── server.js
    ├── app.js
    ├── routes/
    │   ├── authRoutes.js
    │   └── tradeRoutes.js
    ├── controllers/
    │   ├── authController.js
    │   └── tradeController.js
    └── middleware/
        └── authMiddleware.js
```

3.2 `.env`

```
MONGO_URI=mongodb://admin:Admin.123@ac-bpojkkd-shard-00-
00.jcdaigc.mongodb.net:27017,ac-bpojkkd-shard-00-01.jcdaigc.mongodb.net:27017,ac-
bpojkkd-shard-00-02.jcdaigc.mongodb.net:27017/?replicaSet=atlas-lkjfx2-shard-
0&ssl=true&authSource=admin&retryWrites=true&w=majority&appName=Cluster0
JWT_SECRET=mySuperSecretKey
```

> ***Solved**: DNS error with SRV string → switched to standard `mongodb://` URI* *cite turn0search3 .*

3.3 `package.json`

```
{
  "name": "copy-trading-backend",
  "version": "1.0.0",
  "main": "src/server.js",
  "scripts": {
    "start": "node src/server.js",
    "test": "echo \"No tests yet – placeholder\" && exit 0"
  },
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.4.5",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.2",
    "mongodb": "^6.16.0",
    "mongoose": "^7.6.1"
  }
}
```

> ***Fixed**: Missing comma between `"start"` and `"test"` scripts caused JSON parse errors* *cite turn0search0* .

3.4 `testMongo.js`

```
require('dotenv').config();
const { MongoClient, ServerApiVersion } = require('mongodb');

const uri = process.env.MONGO_URI;
const client = new MongoClient(uri, {
  serverApi: { version: ServerApiVersion.v1, strict: true, deprecationErrors: true }
});

async function run() {
  try {
    await client.connect();
    await client.db("admin").command({ ping: 1 });
    console.log("  Connected to MongoDB!");
  } catch (error) {
    console.error("  Mongo error:", error);
  } finally {
    await client.close();
  }
}
run();
```

3.5 `src/server.js`

```
const app = require('./app');
const mongoose = require('mongoose');
require('dotenv').config();
```

```
mongoose.connect(process.env.MONGO_URI)
  .then(() => {
    console.log('✅ MongoDB connected');
    app.listen(process.env.PORT || 5000, () =>
      console.log(`🚀 Server running on port ${process.env.PORT || 5000}`)
    );
  })
  .catch(err => console.error('❌ MongoDB error:', err));
```

3.6 `src/app.js`

```
const express = require('express');
const cors = require('cors');
require('dotenv').config();
const authRoutes = require('./routes/authRoutes');
const tradeRoutes = require('./routes/tradeRoutes');

const app = express();
app.use(cors());
app.use(express.json());

app.use('/api/auth', authRoutes);
app.use('/api/trades', tradeRoutes);

app.get('/', (req, res) => res.send('Backend is working!'));
app.get('/health', (req, res) => res.status(200).send('Healthy'));

module.exports = app;
```

3.7 `src/routes/authRoutes.js`

```
const router = require('express').Router();
const { registerUser, loginUser } = require('../controllers/authController');

router.post('/register', registerUser);
router.post('/login', loginUser);
router.get('/test', (req, res) => res.send('Auth OK!'));

module.exports = router;
```

3.8 `src/controllers/authController.js`

```
const jwt = require('jsonwebtoken');
const bcrypt = require('bcryptjs');

let users = [{ id: 1, email: 'user@example.com', password:
'$2a$10$dummyHashedPassword' }];

exports.registerUser = async (req, res) => {
  const { email, password } = req.body;
  if (users.find(u => u.email === email)) return res.status(400).json({ error: 'User
```

```
exists' });
  const hashed = await bcrypt.hash(password, 10);
  users.push({ id: users.length + 1, email, password: hashed });
  res.status(201).json({ message: 'Registered' });
};

exports.loginUser = async (req, res) => {
  const { email, password } = req.body;
  const user = users.find(u => u.email === email);
  if (!user || !await bcrypt.compare(password, user.password)) {
    return res.status(401).json({ error: 'Invalid credentials' });
  }
  const token = jwt.sign({ id: user.id }, process.env.JWT_SECRET, { expiresIn: '1h'
});
  res.json({ token });
};
```

> *Uses JWT for stateless auth cite turn0search8.*

3.9 `src/routes/tradeRoutes.js`

```
const router = require('express').Router();
const { createTrade, getTrades, deleteTrade, updateTrade } =
require('../controllers/tradeController');
const auth = require('../middleware/authMiddleware');

router.get('/', getTrades);
router.post('/', auth, createTrade);
router.delete('/:id', auth, deleteTrade);
router.put('/:id', auth, updateTrade);

module.exports = router;
```

3.10 `src/controllers/tradeController.js`

```
let trades = [
  { id: 1, symbol: 'AAPL', action: 'buy', price: 150 },
  { id: 2, symbol: 'GOOGL', action: 'sell', price: 2800 },
];

exports.getTrades = (req, res) => res.json(trades);

exports.createTrade = (req, res) => {
  const { symbol, action, price } = req.body;
  if (!symbol || !action || !price) return res.status(400).json({ error: 'Missing
fields' });
  const newTrade = { id: trades.length + 1, symbol, action, price };
  trades.push(newTrade);
  res.status(201).json(newTrade);
};

exports.deleteTrade = (req, res) => {
```

```
  const id = +req.params.id;
  const len = trades.length;
  trades = trades.filter(t => t.id !== id);
  if (trades.length === len) return res.status(404).json({ error: 'Not found' });
  res.json({ message: 'Deleted' });
};

exports.updateTrade = (req, res) => {
  const id = +req.params.id, { symbol, action, price } = req.body;
  const trade = trades.find(t => t.id === id);
  if (!trade) return res.status(404).json({ error: 'Not found' });
  Object.assign(trade, { symbol, action, price });
  res.json({ message: 'Updated', trade });
};
```

3.11 `src/middleware/authMiddleware.js`

```
const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {
  const h = req.headers.authorization;
  if (!h) return res.status(401).json({ error: 'No token' });
  const token = h.split(' ')[1];
  jwt.verify(token, process.env.JWT_SECRET, (err, decoded) => {
    if (err) return res.status(401).json({ error: 'Invalid token' });
    req.userId = decoded.id;
    next();
  });
};
```

---

4. Local Backend Testing

```
cd MT/backend
npm install
node testMongo.js               # expect ✅ Connected to MongoDB!
npm start                       # expect ✅ Server running on port 5000
curl http://localhost:5000/health  # expect Healthy
```

---

5. Frontend Development

5.1 Setup with CRA

```
cd MT
npx create-react-app frontend 【cite】turn0search2【
cd frontend
npm install axios react-router-dom@6 【cite】turn0search7【
```

5.2 `frontend/.env.local`

```
REACT_APP_API_URL=http://localhost:5000
```

> *Must begin with `REACT_APP_` and **restart** server after edits cite turn0search7.*

5.3 `src/api.js`

```
import axios from 'axios';
console.log('API URL:', process.env.REACT_APP_API_URL);
const api = axios.create({ baseURL: process.env.REACT_APP_API_URL });
api.interceptors.request.use(cfg => {
  const t = localStorage.getItem('token');
  if (t) cfg.headers.Authorization = `Bearer ${t}`;
  return cfg;
});
export default api;
```

5.4 `src/TradeApp.js`

```
import React, { useState, useEffect } from 'react';
import api from './api';

export default function TradeApp() {
  const [trades, setTrades] = useState([]);
  const [error, setError] = useState('');
  const [newTrade, setNewTrade] = useState({ symbol: '', action: '', price: '' });

  useEffect(() => { fetchTrades(); }, []);

  function fetchTrades() {
    api.get('/api/trades').then(r => setTrades(r.data)).catch(() => setError('Load
failed'));
  }
  function handleChange(e) { setNewTrade({ ...newTrade, [e.target.name]:
e.target.value }); }
  function handleSubmit(e) {
    e.preventDefault();
    api.post('/api/trades', newTrade).then(r => {
      setTrades(prev => [...prev, r.data]);
      setNewTrade({ symbol: '', action: '', price: '' });
    }).catch(() => setError('Add failed'));
  }
  function handleDelete(id) {
    api.delete(`/api/trades/${id}`).then(fetchTrades).catch(() => setError('Delete
failed'));
  }

  return (
    <div style={{ padding: 20 }}>
      <h2> Copy Trading</h2>
      {error && <p style={{ color: 'red' }}>{error}</p>}
      <form onSubmit={handleSubmit}>
```

```
        <input name="symbol" placeholder="Symbol" value={newTrade.symbol} onChange=
{handleChange} required />
        <input name="action" placeholder="Action" value={newTrade.action} onChange=
{handleChange} required />
        <input name="price" placeholder="Price" type="number" value={newTrade.price}
onChange={handleChange} required />
        <button type="submit">Add</button>
      </form>
      <ul>
        {trades.map(t => (
          <li key={t.id}>{t.symbol} {t.action} @ {t.price} <button onClick={() =>
handleDelete(t.id)}></button></li>
        ))}
      </ul>
    </div>
  );
}
```

5.5 `src/pages/Login.js`

```
import React, { useState } from 'react';
import api from '../api';
import { useNavigate } from 'react-router-dom';

export default function Login() {
  const [c, setC] = useState({ email: '', password: '' });
  const [e, setE] = useState('');
  const nav = useNavigate();

  const handle = async ev => {
    ev.preventDefault();
    try {
      const { data } = await api.post('/api/auth/login', c);
      localStorage.setItem('token', data.token);
      nav('/');
    } catch {
      setE('Invalid credentials');
    }
  };

  return (
    <form onSubmit={handle}>
      {e && <p style={{ color: 'red' }}>{e}</p>}
      <input name="email" placeholder="Email" onChange={x => setC({ ...c, email:
x.target.value })} required />
      <input name="password" type="password" placeholder="Password" onChange={x =>
setC({ ...c, password: x.target.value })} required />
      <button type="submit">Login</button>
    </form>
  );
}
```

5.6 `src/pages/Register.js`

```
import React, { useState } from 'react';
import api from '../api';
import { useNavigate } from 'react-router-dom';

export default function Register() {
  const [c, setC] = useState({ email: '', password: '' });
  const [m, setM] = useState('');
  const nav = useNavigate();

  const handle = async ev => {
    ev.preventDefault();
    try {
      await api.post('/api/auth/register', c);
      setM('Registered! Redirecting...');
      setTimeout(() => nav('/login'), 1500);
    } catch {
      setM('Registration failed');
    }
  };

  return (
    <form onSubmit={handle}>
      {m && <p>{m}</p>}
      <input name="email" placeholder="Email" onChange={x => setC({ ...c, email:
x.target.value })} required />
      <input name="password" type="password" placeholder="Password" onChange={x =>
setC({ ...c, password: x.target.value })} required />
      <button type="submit">Register</button>
    </form>
  );
}
```

5.7 `src/components/PrivateRoute.js`

```
import React from 'react';
import { Navigate } from 'react-router-dom';

export default function PrivateRoute({ children }) {
  return localStorage.getItem('token') ? children : <Navigate to="/login" />;
}
```

5.8 `src/App.js`

```
import React from 'react';
import { BrowserRouter, Routes, Route, Link } from 'react-router-dom';
import Login from './pages/Login';
import Register from './pages/Register';
import PrivateRoute from './components/PrivateRoute';
import TradeApp from './TradeApp';
```

```
export default function App() {
  const logout = () => {
    localStorage.removeItem('token');
    window.location.href = '/login';
  };

  return (
    <BrowserRouter>
      <nav>
        <Link to="/">Home</Link> | <Link to="/register">Register</Link> | <Link
to="/login">Login</Link> | <button onClick={logout}>Logout</button>
      </nav>
      <Routes>
        <Route path="/login" element={<Login />} />
        <Route path="/register" element={<Register />} />
        <Route path="/" element={<PrivateRoute><TradeApp /></PrivateRoute>} />
      </Routes>
    </BrowserRouter>
  );
}
```

---

6. Local Integration Testing

```
# Backend
cd MT/backend
npm start

# Frontend (in a new terminal)
cd MT/frontend
npm start
```

- Visit `http://localhost:3000` → Register, Login, and manage trades.
- Open DevTools console to confirm `API URL: http://localhost:5000` cite turn0search7 .

---

7. Deployment

7.1 Backend on Render

1. In Render dashboard click **New** → **Web Service**.
2. Connect GitHub `MT`, set **Root Dir** to `backend`, **Build** `npm install`, **Start** `npm start`.
3. Add **Env Vars**:
   - `MONGO_URI` (Atlas URI)
   - `JWT_SECRET` cite turn0search4
4. Deploy and test at `https://copy-trading-backend-ksfs.onrender.com/health`.

7.2 Frontend on Vercel

1. In Vercel, click **New Project**, import `MT`, set **Root Dir** to `frontend`.
2. Build **npm install && npm run build**, output `build`.

3. Add Env Var `REACT_APP_API_URL=https://copy-trading-backend-ksfs.onrender.com` cite turn0search24 .
4. Deploy and visit your Vercel URL.

Future Work

- Add **unit & integration tests** (Mocha, Jest).
- Set up **CI for frontend** (GitHub Actions + Vercel CLI).
- Integrate **Sentry** for error monitoring.
- **Containerize** with Docker & compose.
- Enhance **security**: rate limiting, input validation, HTTPS enforcement.
- Implement **JWT refresh tokens** and **role-based access**.

Convert to PDF

Copy this entire content into Dillinger, then click **Export** → **PDF**.

Congratulations! You now have a self-contained guide to build, run, and deploy your full-stack Copy-Trading App, step by step.