BEYOND LOCALHOST

Deploying Voyager's Log to the Public Internet

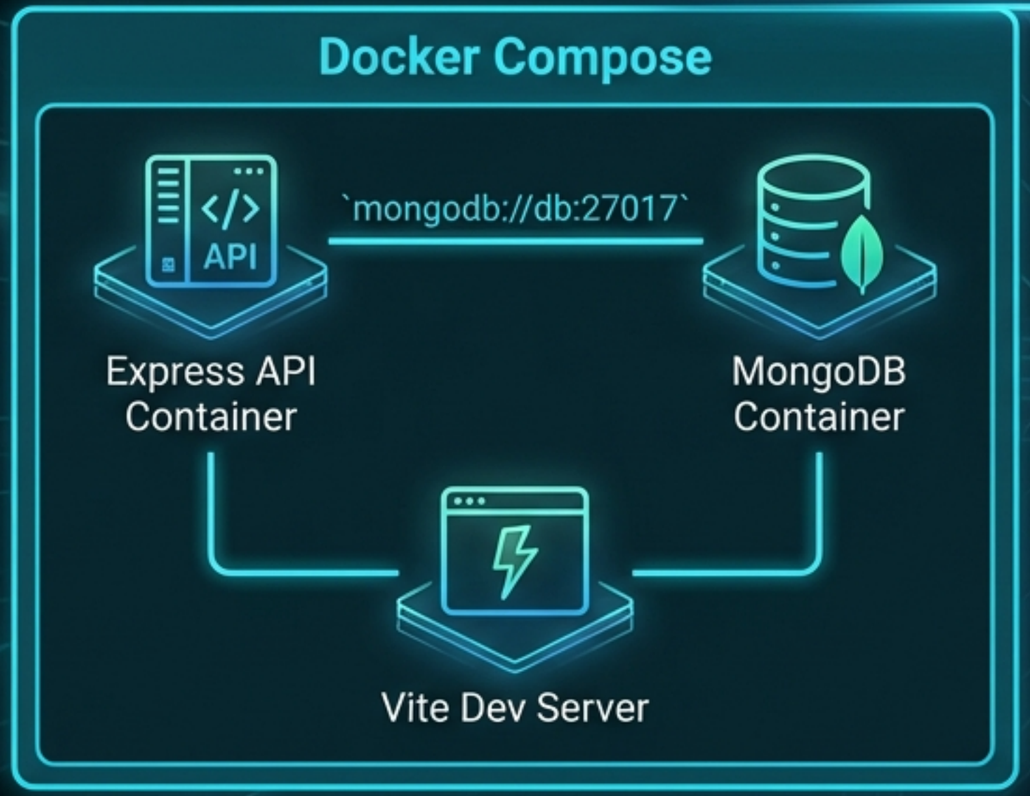
```
`SYSTEM: ONLINE`
`TARGET: HOSTED RUNTIME`
`OBJECTIVE: ARCHITECTURAL SYNTHESIS &
            OPERATIONAL BLUEPRINT`
```

p.s. keep learning

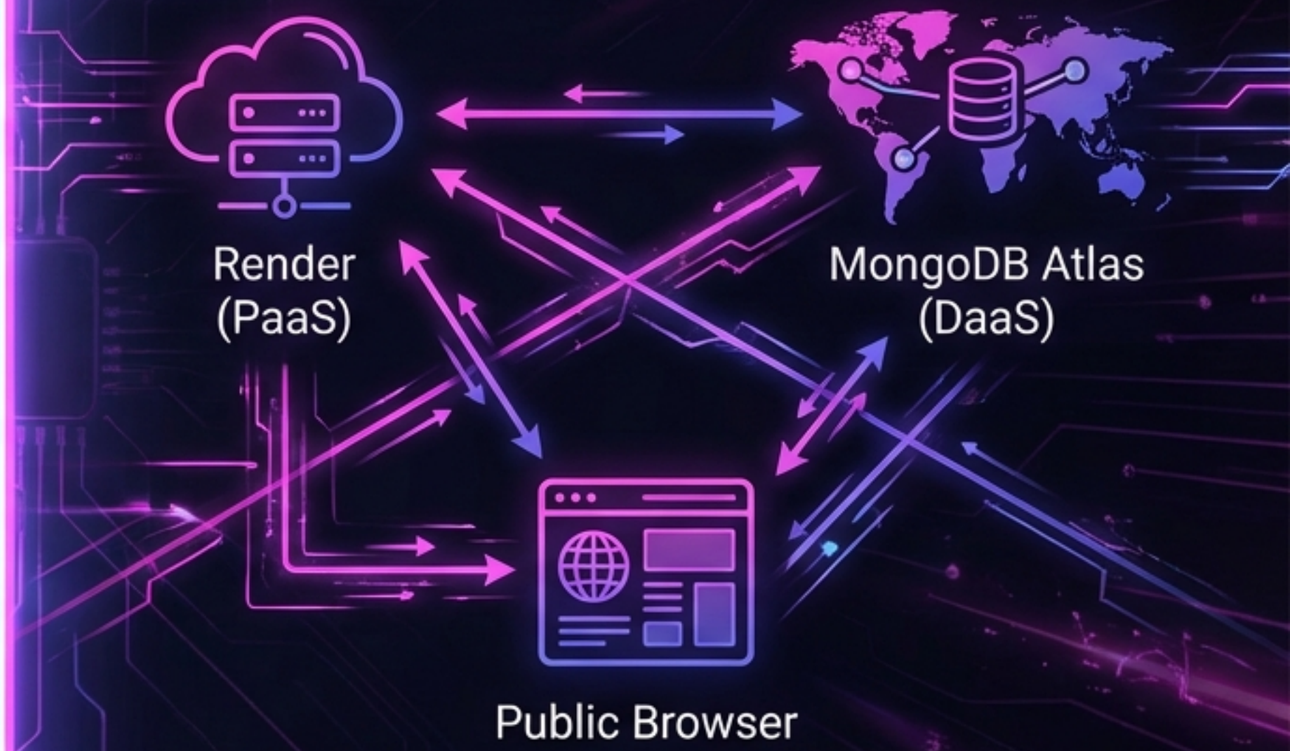
Local Autonomy

Distributed Reality

Trading the training wheels of Compose for real-world DNS and public routing.



Isolated bridge network.
Guaranteed startup order.



Public IP routing.
Platform-injected configuration.
Managed persistence.

p.s. keep learning

The Pre-Flight Refactor

Dynamic Port Binding

```
~~const PORT = 3000;~~
const PORT = process.env.PORT || 3000;
```

Dynamic DB Routing

```
~~mongoose.connect('mongodb://db:27017')~~
mongoose.connect(process.env.MONGO_URI)
```

Dynamic Secrets

```
~~secret: 'dev-only-secret'~~
secret: process.env.SESSION_SECRET
```

Production Static Serving

```
if (process.env.NODE_ENV === 'production') {
  app.use(express.static(distPath));
}
```

Teach Express to serve the compiled Vite app.

p.s. keep learning

The Multi-Stage Paradigm

The Dev Server Payload

- Vite Dev Server
- Frontend Source Dependencies
- Tailwind Build Tooling
- Raw Source Files

WARNING: We do NOT ship Vite dev servers to production.

The Production Image

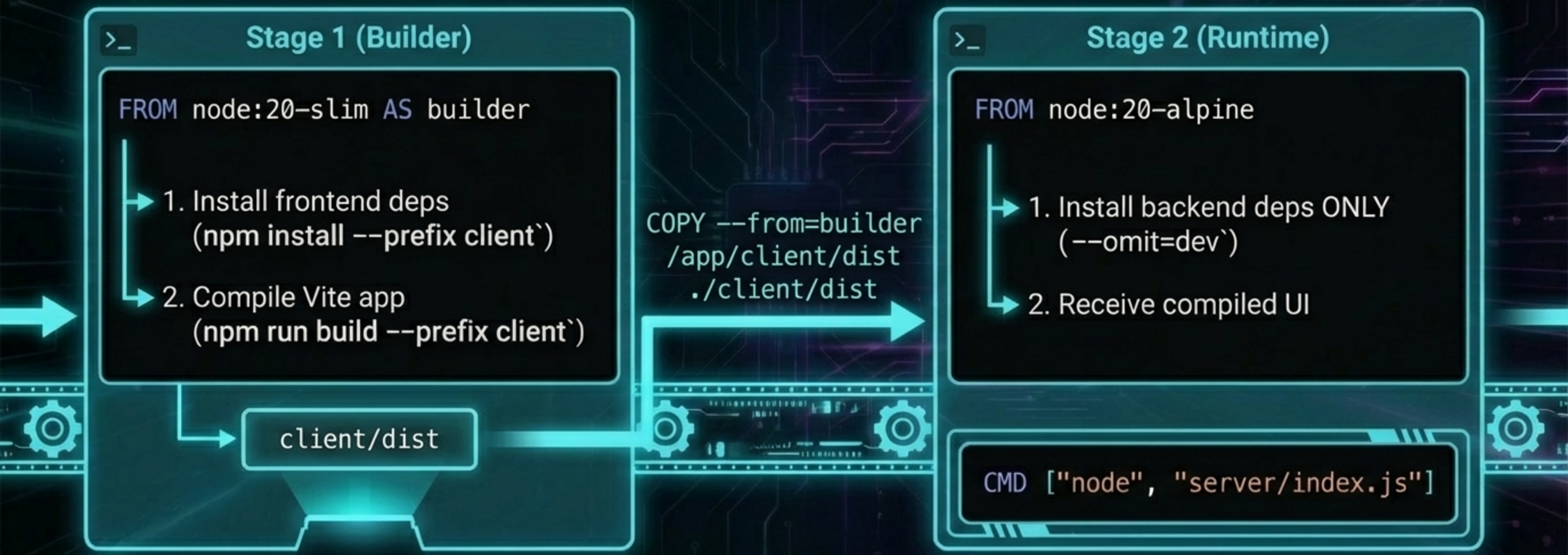
- Express Server Code
- Backend Runtime Dependencies
- Compiled Static Frontend (client/dist)

No extra passengers. Just appropriately trimmed runtime execution.

Locally, Vite is a dev server. In production, Vite is a build tool that gets out of the way.

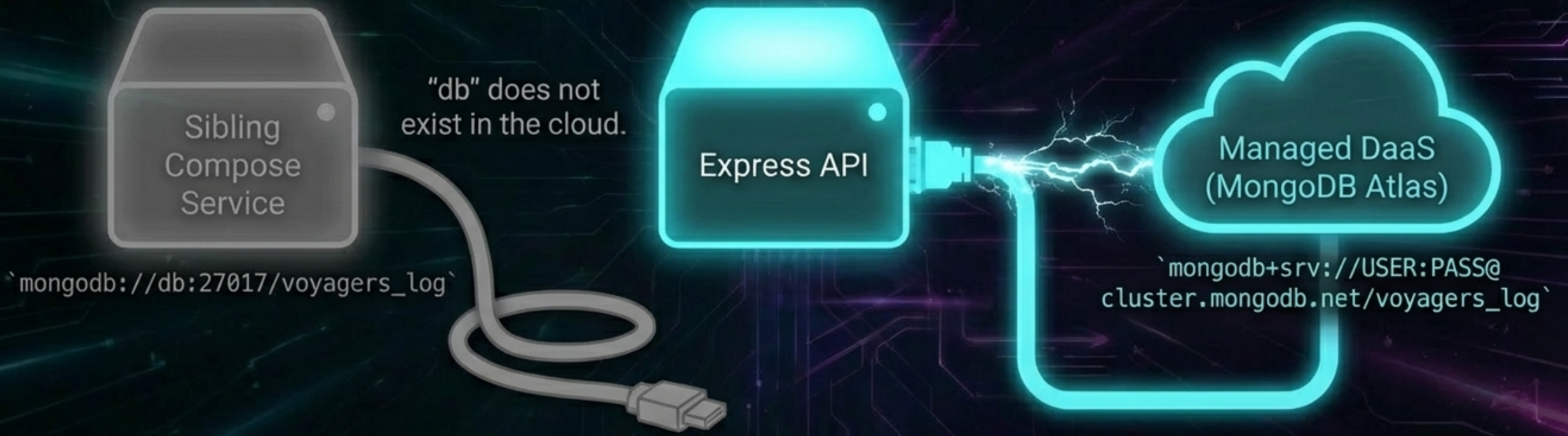
p.s. keep learning

The Docker Assembly Line



p.s. keep learning

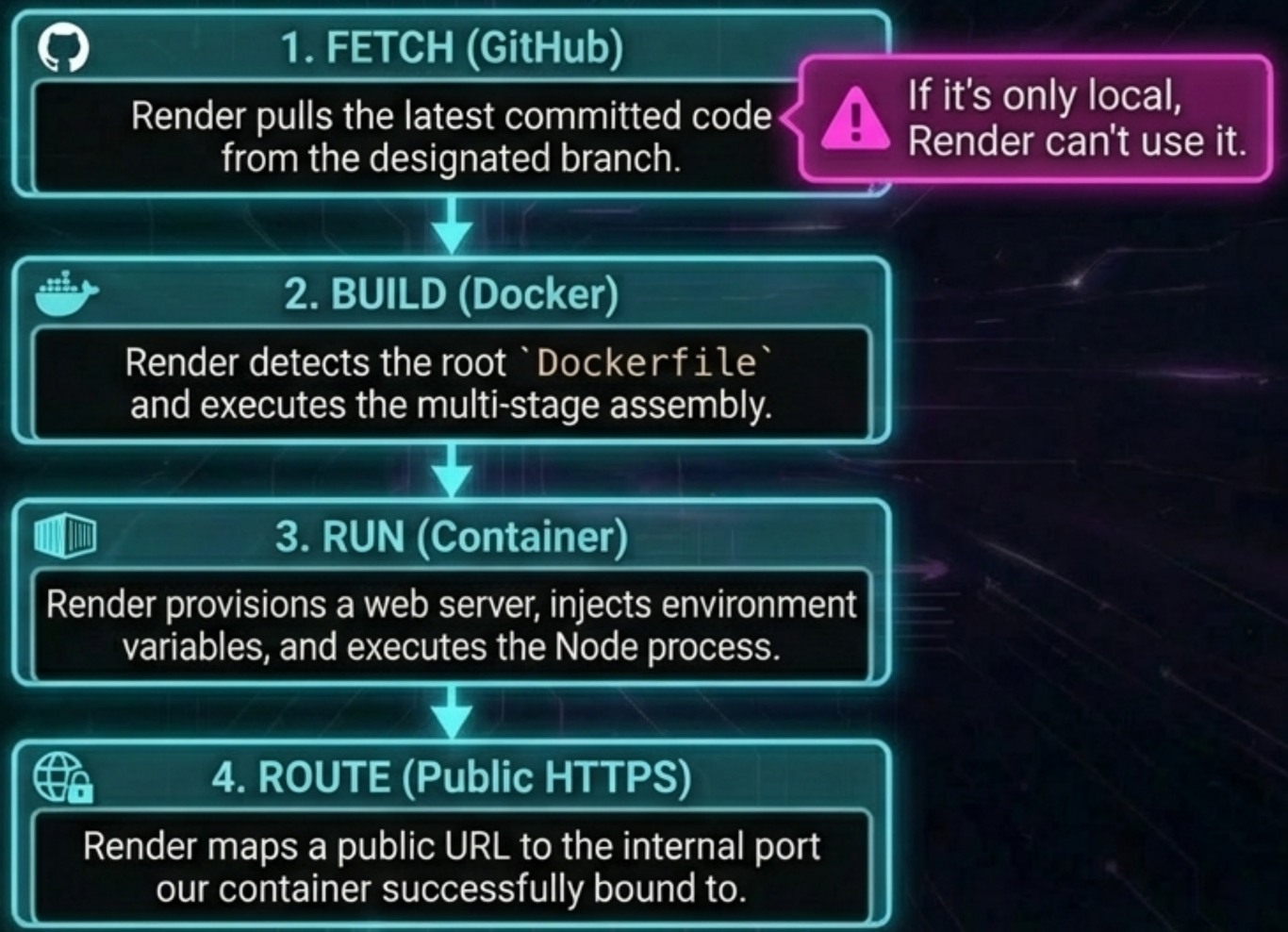
Precision Persistence: Atlas



One Clean Swap: The application code stays identical.
Only the environment variable (`process.env.MONGO_URI`) shifts.

The Hosted Runtime (Render)

Trading hardware management for managed execution.



Supplying the Environment

Committed secrets are dead.

Render Environment Settings

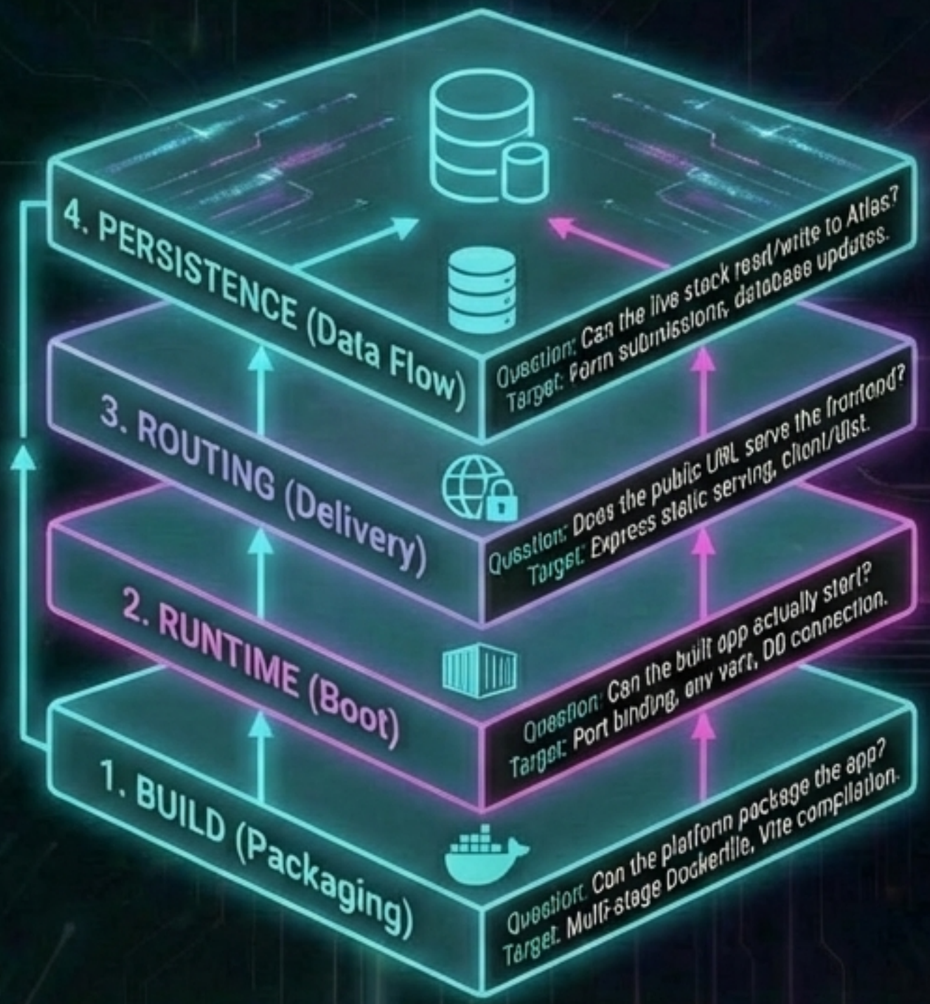
- `MONGO_URI` = Verified Atlas connection string
- `SESSION_SECRET` = Strong, unguessable production string
- `NODE_ENV` = production (Critical for Express static serving)

WARNING: Do NOT manually set `PORT`. The PaaS provides it dynamically.

The code stays static; the injected environment provides the reality.

The 4 Layers of Verification

Diagnosis requires discipline, not guessing.



p.s. keep

Layers 1 & 2: Build vs. Runtime

Build Phase (Layer 1)

> **Log Source:** Docker execution logs.
> **Success Signals:** `vite build`, `✓ modules transformed`, `dist/index.html generated`.
> **Failure Mode:** Code never reaches startup. Fails at `npm install` or file paths.

Runtime Phase (Layer 2)

> **Log Source:** Node.js process logs.
> **Success Signals:** `Connected to MongoDB`, `API listening on port 10000`.
> **Failure Mode:** Immediate crash loops. `MongooseServerSelectionError`, `EADDRINUSE`.

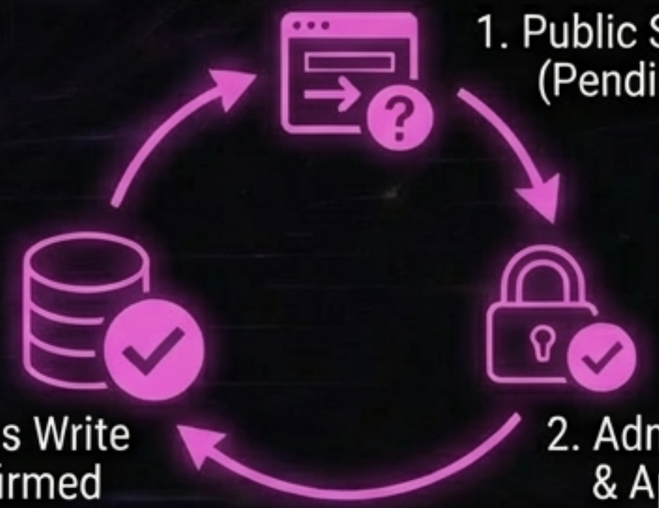
CRITICAL RULE: Build success is not app success. Do not hunt for runtime errors in the build feed.

Layers 3 & 4: Routing & Persistence

Layer 3: External Routing

Checkpoint: No ‘Cannot GET /’. The `NODE_ENV=production` logic successfully delivered `client/dist`. (Warning: A working homepage can still lie to you).

Layer 4: Distributed Persistence



Checkpoint: The ultimate proof. Data wasn’t just displayed; it traveled through the live API and was written to the managed database.

p.s keep

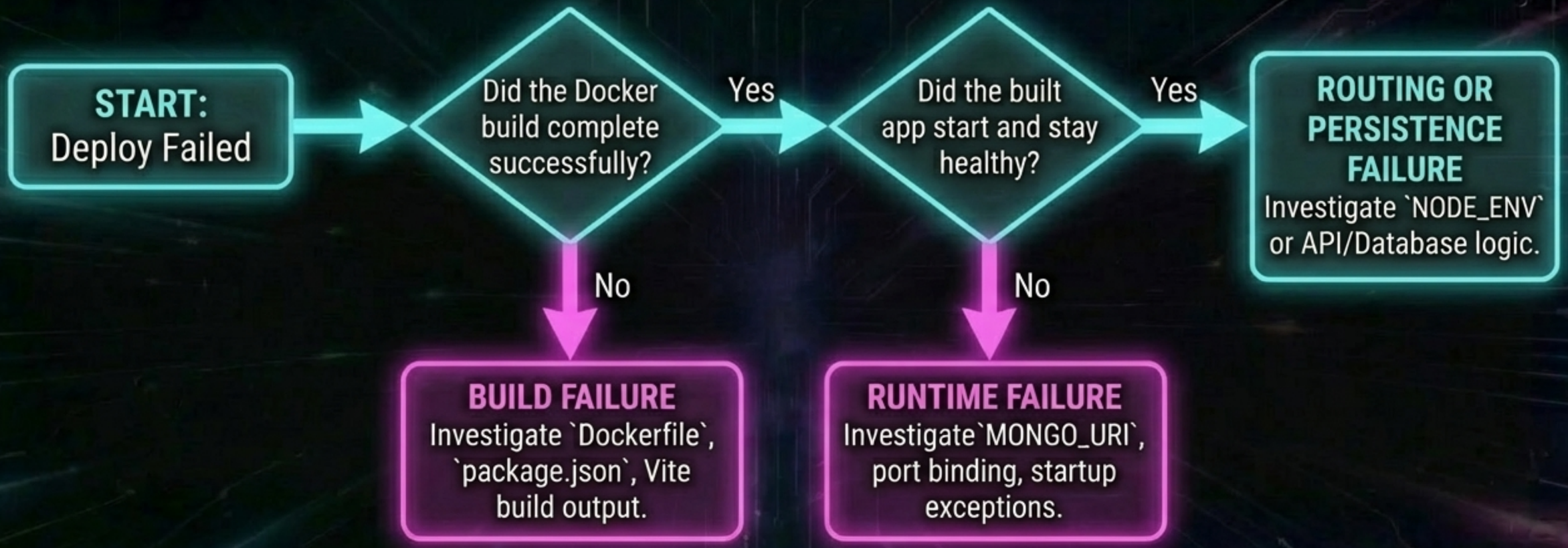
Platform Logs as Absolute Truth

The Myth:
"I'll run it locally to see why the cloud is broken."

The Reality:
The deployed app has a different port, a remote database, and a reverse proxy. Local success is not production proof.

Render's logs are the remote terminal you no longer have direct shell access to.
Silence is a clue: If there are no logs, the process never booted.

Diagnostic Triage Tree



First classify the failure. Then fix the layer that actually failed.
Do not solve the wrong problem.

Dry Dock Drills

Break the Build (Layer 1)

Action: Introduce a bad import in Vite.
Result: Deploy fails during `npm run build`. App never reaches startup.

Break Runtime Config (Layer 2)

Action: Mutate `MONGO_URI` in Render settings.
Result: Build succeeds. App crashes on startup trying to connect to Atlas.

Break Routing (Layer 3)

Action: Remove `NODE_ENV=production`.
Result: App boots. URL serves `Cannot GET /` because static path isn't triggered.

Break Persistence (Layer 4)

Action: Break Mongoose create operation.
Result: Homepage loads. UI acts normal until a submission throws a 500 API error.

The Shipping Manifest

	Local Assumptions	Production Realities
1	Port 3000 is the app port	The platform provides the runtime port.
2	The DB is at `db`	The DB is reached through `MONGO_URI`.
3	Vite serves the frontend	Express serves built frontend assets.
4	My local files are the source	The committed Git repo is the source.

The app didn't change. How it is built, configured, connected, and hosted changed.

Professor Solo