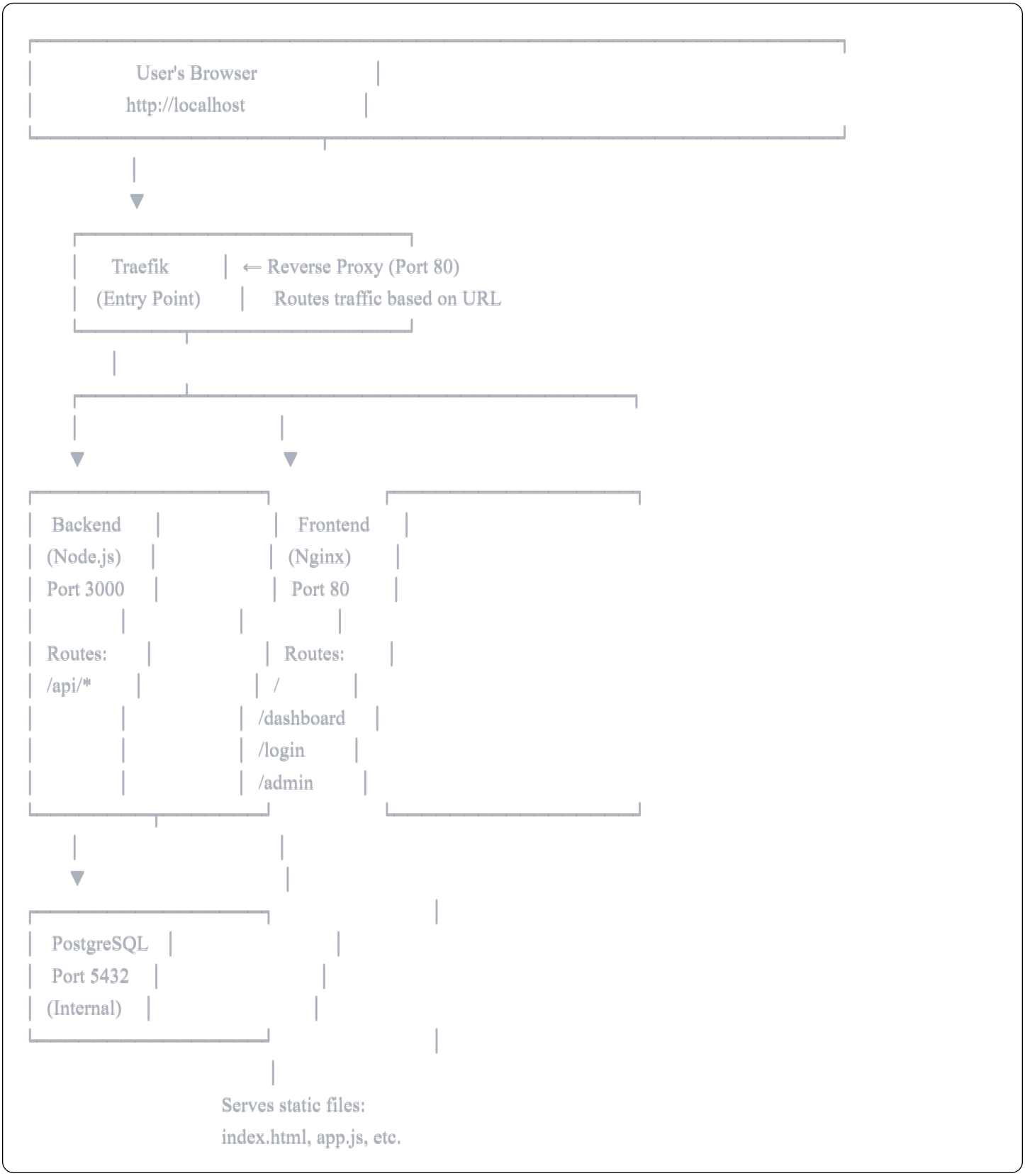
Architecture Overview - Traefik + Nginx Explained

🏗️ System Architecture



🎯 How It Works

Request Flow

1. User visits `http://localhost/`

```
Browser → Traefik → Frontend (Nginx) → Returns index.html + React app
```

2. User clicks "Login" (stays in React app)

```
React Router handles /login → No server request
```

3. User submits login form

```
React → Axios POST http://localhost/api/auth/login
    → Traefik sees /api → Routes to Backend
    → Backend processes → Returns JWT token
```

4. User creates project

```
React → Axios POST http://localhost/api/projects
    → Traefik sees /api → Routes to Backend
    → Backend → PostgreSQL → Returns data
```

🔧 Component Roles

Traefik (Reverse Proxy)

Purpose: Smart router that directs traffic

What it does:

- Listens on port 80
- Reads Docker labels from containers
- Routes `/api/*` → Backend container
- Routes everything else → Frontend container
- No configuration files needed (uses labels)

Why we use it:

- ✅ Automatic service discovery
- ✅ Zero-downtime deployments
- ✅ Load balancing (if multiple instances)
- ✅ Easy HTTPS with Let's Encrypt

Nginx (Static File Server)

Purpose: Serves React build files

What it does:

- Runs inside frontend container
- Serves `/usr/share/nginx/html` (React build output)
- Handles SPA routing (all routes → index.html)
- Gzip compression
- Static file caching

Why we use it:

- ✅ Fast static file serving
- ✅ Production-ready
- ✅ Low resource usage
- ✅ Battle-tested

Backend (Node.js + Express)

Purpose: REST API server

What it does:

- Handles all `/api/*` requests
- Authenticates users (JWT)
- Manages database operations
- Calls AI provider APIs
- Returns JSON responses

Frontend (React + Vite)

Purpose: User interface

What it does:

- Single Page Application (SPA)
- Client-side routing (React Router)
- Makes API calls to backend
- Renders UI components
- Manages local state

📊 Port Mapping

Service	Internal Port	External Port	Access
Traefik	80	80	http://localhost
Traefik Dashboard	8080	8080	http://localhost:8080
Backend	3000	-	Via Traefik only
Frontend (Nginx)	80	-	Via Traefik only
PostgreSQL	5432	-	Internal only

Key Point: Only Traefik is exposed. Everything else is internal to Docker network.

🔀 Routing Rules (Traefik Labels)

Backend Labels

```yaml
labels:
  - "traefik.enable=true"
  - "traefik.http.routers.backend.rule=Host(`localhost`) && PathPrefix(`/api`)"
  - "traefik.http.services.backend.loadbalancer.server.port=3000"
```

Meaning:

- Enable Traefik routing for this container
- Match requests to `localhost` with path starting with `/api`
- Forward to container's port 3000

Frontend Labels

```yaml
labels:
  - "traefik.enable=true"
  - "traefik.http.routers.frontend.rule=Host(`localhost`)"
  - "traefik.http.services.frontend.loadbalancer.server.port=80"
```

Meaning:

- Enable Traefik routing for this container
- Match all requests to `localhost` (catch-all)
- Forward to container's port 80 (Nginx)
- Lower priority than backend (PathPrefix is more specific)

🌐 URL Examples

Frontend URLs (Handled by React Router)

```
http://localhost/              → Frontend (index.html)
http://localhost/login         → Frontend (index.html)
http://localhost/dashboard     → Frontend (index.html)
http://localhost/project/1     → Frontend (index.html)
http://localhost/admin         → Frontend (index.html)
```

All these return the same `index.html`, then React Router takes over.

Backend API URLs

```
http://localhost/api/health          → Backend
http://localhost/api/auth/login      → Backend
http://localhost/api/projects        → Backend
http://localhost/api/chats/1/messages → Backend
http://localhost/api/providers/keys  → Backend
```

All `/api/*` requests go to the backend.

🔄 Development vs Production

Development Mode

Frontend:

```bash
cd frontend
npm run dev  # Vite dev server on port 5173
```

- Hot module replacement
- Source maps
- Fast refresh

Backend:

```bash
cd backend
npm run dev  # Nodemon on port 3000
```

- Auto-restart on changes
- Direct database connection

Access:

- Frontend: http://localhost:5173
- Backend: http://localhost:3000
- (No Traefik in development)

Production Mode (Docker)

```bash
docker-compose up -d
```

Frontend:

- Built with Vite (`npm run build`)
- Static files served by Nginx
- Gzipped, cached, optimized

Backend:

- Runs with `npm start`
- Production mode (`NODE_ENV=production`)

Access:

- Everything: http://localhost (via Traefik)

🚀 Scaling Options

Multiple Backend Instances

```yaml
backend:
  deploy:
    replicas: 3
```

Traefik automatically load balances between them.

Multiple Frontend Instances

Not needed - Nginx serves static files efficiently.

Database Replication

Can add read replicas for PostgreSQL if needed.

🔐 Security Layers

1. Network Isolation

All containers on private `ai-platform` network. Only Traefik exposed to host.

2. Traefik (Edge)

- Can add rate limiting
- Can add IP whitelist
- Can add authentication middleware

3. Backend (API)

- JWT authentication
- Admin role checks
- Input validation
- SQL injection prevention (parameterized queries)

4. Database

- Not exposed to host
- Only backend can connect
- User permissions

📝 Nginx Configuration Explained

```nginx
server {
    listen 80;
    root /usr/share/nginx/html;

    # Try to serve file, fallback to index.html for SPA routing
    location / {
        try_files $uri $uri/ /index.html;
    }

    # Cache static assets
    location ~* \.(js|css|png|jpg|svg)$ {
        expires 1y;
    }
}
```

Why `try_files $uri $uri/ /index.html`?

React Router uses client-side routing. When user visits `/dashboard`:

1. Nginx checks if file `/dashboard` exists → No
2. Nginx checks if directory `/dashboard/` exists → No
3. Nginx serves `/index.html` → React app loads → React Router shows Dashboard

Without this, refreshing on `/dashboard` would give 404.

🧪 Testing the Architecture

Test Traefik Routing

```bash
# Should return React app HTML
curl http://localhost/

# Should return API JSON
curl http://localhost/api/health

# Traefik dashboard
open http://localhost:8080
```

Test Backend Directly (Development)

```bash
# If backend running outside Docker
curl http://localhost:3000/api/health
```

Test Frontend Directly (Development)

```bash
# If frontend running with Vite
open http://localhost:5173
```

🐛 Common Issues

"Cannot reach backend from frontend"

Problem: Frontend tries to call `http://backend:3000/api`

Solution: Use relative URLs `/api` not absolute URLs. Traefik handles routing.

```javascript
// ❌ Wrong
axios.get('http://backend:3000/api/health')

// ✅ Correct
axios.get('/api/health')
```

"404 on page refresh"

Problem: Nginx not configured for SPA routing

Solution: Check nginx.conf has `try_files $uri $uri/ /index.html;`

"Traefik shows no routes"

Problem: Labels not applied

Solution:

```bash
# Check Traefik dashboard
open http://localhost:8080

# Restart services
docker-compose restart
```

📊 Summary

Component	Technology	Purpose	Port
Traefik	Go	Reverse proxy & router	80
Nginx	C	Static file server	80 (internal)
Backend	Node.js	REST API	3000 (internal)
Frontend	React	UI	Built files
PostgreSQL	SQL	Database	5432 (internal)

Key Takeaway:

- Traefik = Smart router (reads Docker labels)
- Nginx = Fast file server (inside frontend container)
- They work together, not in competition!