[t]

Rad Link: Serverless URL Shortener

Custom SSL Termination on AWS Academy

RazaSoft Tech | Semester Project | Dec 2025

1. Project Abstract

Rad Link is a production-grade URL shortening service designed to differentiate itself through "Radical Transparency." It features a **Stupidity Dashboard**—a real-time visualization of system-wide usage, OS distribution wars, and geographic traffic heatmaps. The project demonstrates how to overcome the strict constraints of the AWS Academy Learner Lab environment by implementing a hybrid serverless architecture.

2. The Architecture Workaround

A major challenge in AWS Academy is the restriction on CloudFront and AWS Certificate Manager (ACM) for custom domains. To achieve secure HTTPS termination for `api.razasoft.tech`, we engineered a custom proxy layer.

Infrastructure Flow:

- **Entry Point:** An **EC2 t2.micro** instance acts as the public-facing gateway.
- **SSL Layer:** We utilized **Certbot** to generate and auto-renew Let's Encrypt certificates directly on the EC2 instance, bypassing ACM entirely.
- **Reverse Proxy:** An Nginx server block accepts HTTPS traffic and reverse-proxies it to the internal AWS HTTP API Gateway URL.
- **Result:** Seamless SSL termination with zero cost for custom certificates.

3. Backend Engineering

The core logic resides in a **Monolithic Node.js 20.x Lambda**. A monolith pattern was chosen over microservices to reduce cold starts and simplify the shared database connection logic.

Atomic Transactions: Updating analytics is complex. When a link is clicked, three distinct entities must update: the Link's specific count, the User's aggregate count, and the Global System dashboard. We use DynamoDB `TransactWriteItems` to ensure these updates happen atomically—either all metrics update, or none do, ensuring data consistency.

4. DynamoDB Single Table Design

To support the "Mega-Sync" dashboard (loading all data in one request), we used overloaded keys in a single table.

Data Entity	Key Structure
Link Meta	PK: `LINK#<id>` SK: `META`
Daily Stats	PK: `LINK#<id>` SK: `STATS#<date>`
Global State	PK: `SYSTEM#GLOBAL` SK: `STATS#TOTAL`

5. Flutter Frontend Architecture

The frontend is a reactive web application built with Flutter 3.x and **Riverpod** for state management. It features a custom "Cyber-Glass" aesthetic optimized for web performance.

Optimistic UI Implementation: To mask network latency, the app implements Optimistic UI. When a user generates a link:

1. The UI immediately adds the link to the dashboard list locally.
2. The user perceives 0ms latency.
3. The API call processes in the background.
4. If the Lambda returns an error (e.g., WAF block), the state rolls back and notifies the user.

6. Security & Auth

- **Cognito & SRP:** We use Amazon Cognito User Pools with the Secure Remote Password (SRP) protocol. Passwords never travel over the network.
- **JWT Authorization:** The EC2 proxy passes headers intact to API Gateway, which validates the JSON Web Token (JWT) signature before invoking Lambda.
- **WAF:** AWS Web Application Firewall is deployed to rate-limit aggressive IPs and prevent bot spam.

7. Dashboard Visualization

The "Stupidity Dashboard" aggregates data from the `SYSTEM#GLOBAL` partition.

[Insert Dashboard Screenshot Here]

Showing Real-Time OS Distribution

8. Conclusion

Rad Link proves that robust, production-grade architectures can be built even within restrictive environments. By leveraging EC2 for SSL termination and DynamoDB for massive scale, the system achieves high performance with minimal idle costs.