

Performance Comparison of AWS Elastic Beanstalk vs AWS Lambda for a Responsive Equestrian Competition Management Web Application

By Siobhan Kelly Stephens – 1800708
BSc (Hons) Computing (Software Engineering)

1. Introduction

Equestrian competition days create short, intense demand spikes where slow admin workflows and inconsistent system responsiveness cause real delays for competitors. This project delivers a responsive cloud-based equestrian web application and evaluates two functionally equivalent AWS backend deployments, Elastic Beanstalk vs Lambda + API Gateway, to assess responsiveness under intermittent, event-driven usage.

2. Aims and Objectives

Aim: Build a responsive cloud-based competition entry system and evaluate performance across AWS Elastic Beanstalk vs AWS Lambda + API Gateway.

Objectives:

1. Review research on performance and evaluation metrics.
2. Design and implement a responsive full-stack application.
3. Deploy in a cloud environment with repeatable releases.
4. Evaluate frontend and backend performance across both deployments.

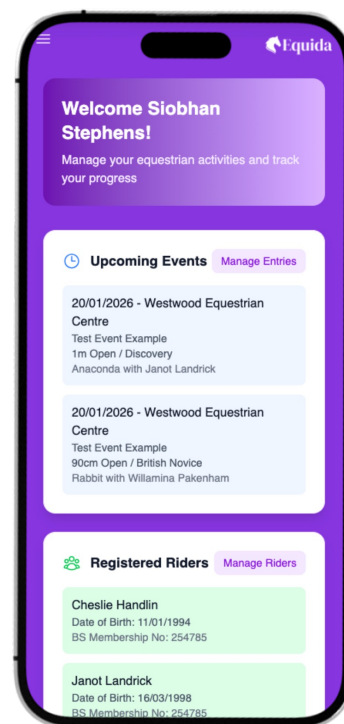
3. Research

Serverless can reduce cost for spiky demand but risks cold-start and tail latency [1], [2]. Performance should be evaluated using both API response time distributions and user-perceived frontend metrics, because each captures different behaviour.

4. Design & Methodology

React frontend, AWS Cognito auth, shared PostgreSQL DB on RDS. Two equivalent backends: Express on Elastic Beanstalk vs Lambda behind API Gateway. **Fair comparison:** shared core logic + same database, thin adapters only.

Method: identical API routes tested with repeatable Postman runs and frontend audited with Lighthouse



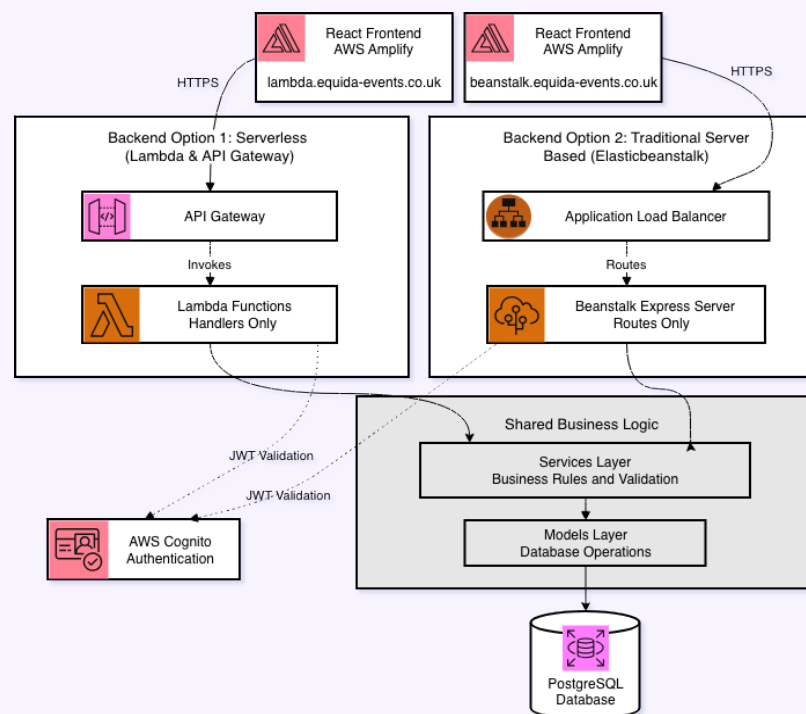
5. Implementation

Frontend: React SPA deployed as two parallel Amplify apps to keep the UI identical while switching backend via VITE_API_URL.

Auth: AWS Cognito (JWT + group roles), with route guards in the UI and server-side enforcement.

Data: Shared PostgreSQL on AWS RDS; core entities cover users, riders, horses, events, venues, classes, entries and results.

Backends (equivalent): Express API on Elastic Beanstalk + ALB vs Lambda + API Gateway. Shared core module (backend/lib/) keeps behaviour consistent; only thin adapters differ.



6. Results and Findings

Backend (Postman, 6 routes): 30 runs per route. Lambda includes 25 warm + 5 cold to reflect intermittent usage.

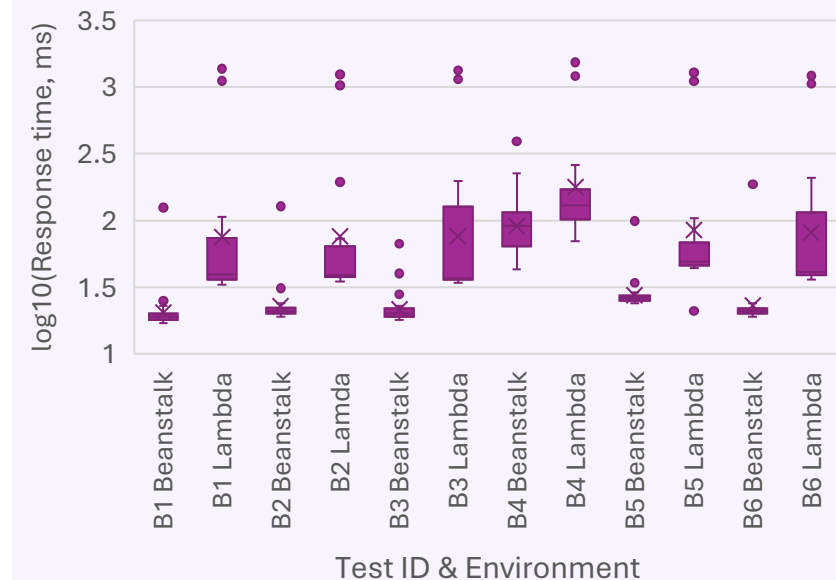
p95: Lambda 1244–1313ms vs Beanstalk 24–201ms

p99: Lambda 1275–1478ms vs Beanstalk 59–344ms

Heavy tail in Lambda consistent with cold-start and platform.

Frontend (Lighthouse, 4 pages): No statistically significant differences for most metrics; one CLS difference was small and still within “good” thresholds.

Beanstalk vs Lambda API Latency by Endpoint
Test B1-B6 (log10 response time in ms)



7. Recommendations and Conclusions

Conclusion: The serverless backend produced **higher variance and heavier tail latency** under intermittent usage, while Beanstalk remained more stable. Frontend Lighthouse scores stayed consistently strong, suggesting the UI design reduced visible impact in synthetic audits.

Recommendation:

Pick **Elastic Beanstalk** when predictable responsiveness matters across all actions (lower p95/p99 and tighter distributions).

Pick **Lambda + API Gateway** for spiky demand and cost, alignment but consider cold-start overhead and caching for suitable read-heavy routes.

8. References

- [1] R. Aapakallio-Autio, ‘From Elastic Beanstalk to Lambda: A comparative case study on the AWS tools’, Aalto University, Jul. 2021.
- [2] Gary Arora, Akash Tayal, and Rakinder Sembhi, ‘Determining the Total Cost of Ownership: Comparing Serverless and Server-based Technologies’. Deloitte, Jul. 2021.