AnyCompany <> Railway

Migration to Railway

Eric Lim, Solutions Architect

Agenda

AnyCompany’s Current state

Pain points

Current AWS Architecture

Railway Demo

Railway Value Proposition

Migration Strategy

Questions & Next Steps

About AnyCompany

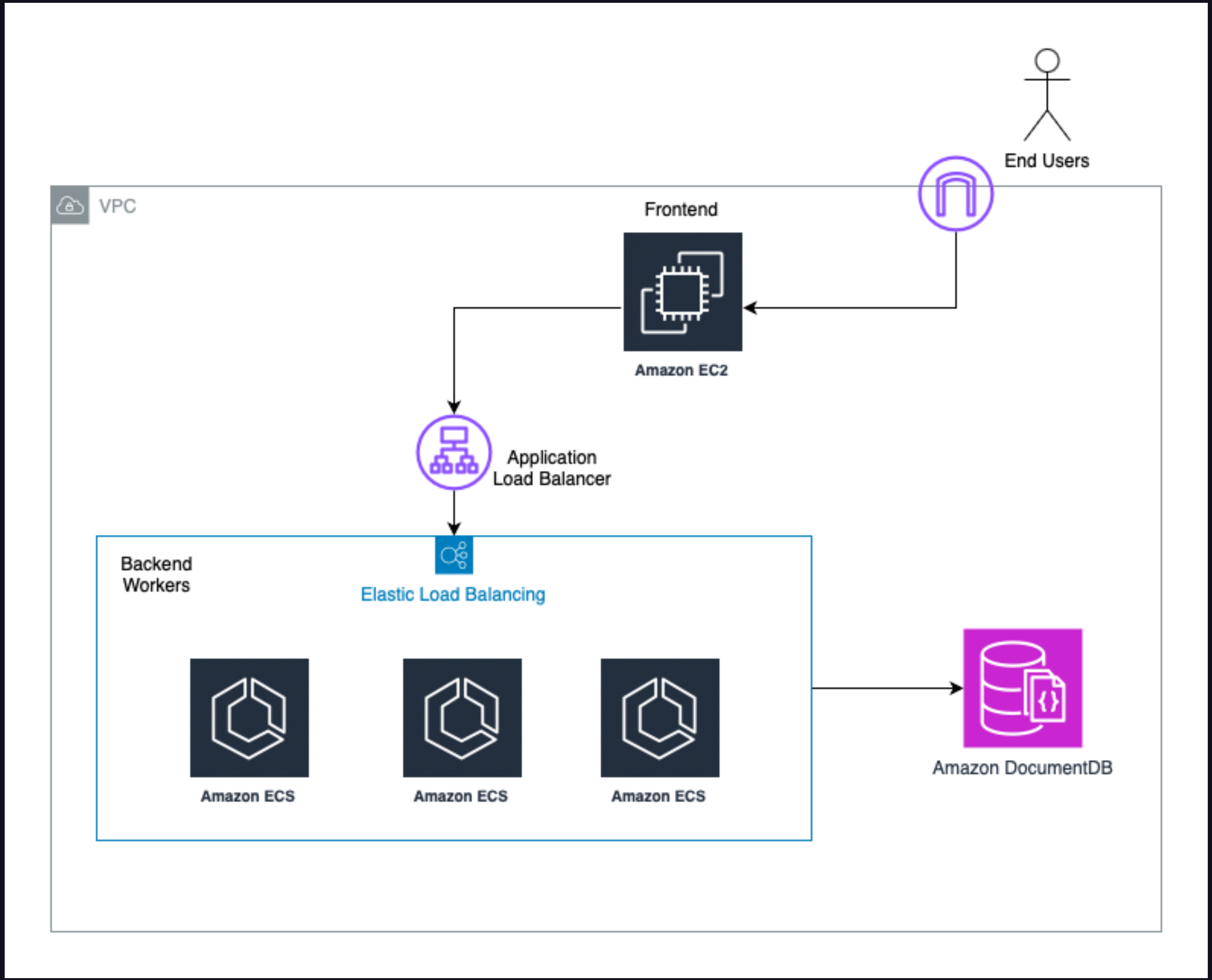
- DevTools SaaS company: Developer productivity, engineering metrics
- Startup/SMB - currently scaling out service.
- Mainly US customers (EDU), plans to go global
- Currently deployed on AWS
- Small team of developers
  - DevOps overhead consuming 40% of senior developer time
  - Complex AWS IAM/AWS Organizations management
  - Difficulty in hiring CloudOps Engineer & Onboarding new developers to infra

Pain points

- DevOps, CloudOps Complexity: consuming 40% of senior eng. time
    - IAM, AWS Organizations, AWS Account security been a blocker
    - Difficulty in onboarding new SWEs due to infra complexity
- Tech stack deeply with AWS specific services (ECS, DocumentDB, etc) - vendor-locked
- Difficulty managing IAC - Terraform version updates, state files, infra PR
- Scattered Observability - CloudWatch, XRay, Quicksight, Datadog, Wiz..
- Overall - Development to Deployment time is increased

Current AWS Architecture

- 3 tier web application structure
- DocumentDB for storing customer data
- ECS is contaner based workers, needs scale
- Frontend is for UI users, most usage from the API



Demo

Where Railway can help

- Build and Deploy fast
    - Railway intuitive UI, Infra change tracked automatically, no need for separate IaC pipeline. Overall easier learning curve & faster deployments
- Eliminate Cloud Vendor Lock-in
    - Use standard container deployments & platform-agnostic development practices. Standardization of service interfaces.
- Unified Observability Platform
    - Single-pane-of-glass for all monitoring. Integrated logging, metrics, tracing.
    - Built-in alerts & deployment tracking. Visibility across all users and services.
    - Faster MTTR(mean time to resolution)

Proposed Migration Strategy

1. Prep & Assessment
    - AWS usage, DB size, mapping AWS services to Railway equivalent, Application analysis, CI/CD
2. Migration & Testing
    1. Dev env. migration
    2. Application layer deployment migration + testing
    3. Data migration
        1. Database replica, data consistency, downtime req. backup. restore.
        2. Storage (file transfer testing, file integrity, backup, rollback if necessary)
    4. Testing (Load, stress, failover, backup&restore, performance benchmarks, error handling)
3. Production cutover (maintenance, rollback plan, final data sync)

Concluding thoughts

With Railway..

- Faster time to market (build!)
- Cost structure improvements
  - Eliminated: Terraform license, AWS support plans, third-party monitoring, DevOps training/hiring overhead
  - Reduced: Infra maintenance, Fewer specialized roles in company, lower operational complexity
- Risk reduction in infra - lesser moving parts
- Business agility

Thank you