INDEPENDENT HANDS-ON PROJECT

Private ECS Fargate Application Platform

Immutable delivery, private tasks, autoscaling and observable deployments

Portfolio track	Independent Hands-on Engineering
Evidence level	Terraform fmt, init and validate passed
Author	Saurabh Dindokar

Evidence boundary: This is a public-safe personal implementation lab. It is not client production work, and no live deployment is claimed.

Overview

A Terraform platform for an ECS Fargate service behind an Application Load Balancer, with ECR image controls, private tasks, CloudWatch logs, Container Insights, rolling deployments and target-tracking autoscaling.

Engineering Problem

Containerized workloads need a repeatable platform that separates public ingress from private runtime tasks, gives deployment and service-health visibility, and avoids granting application permissions before the workload requires them.

Architecture

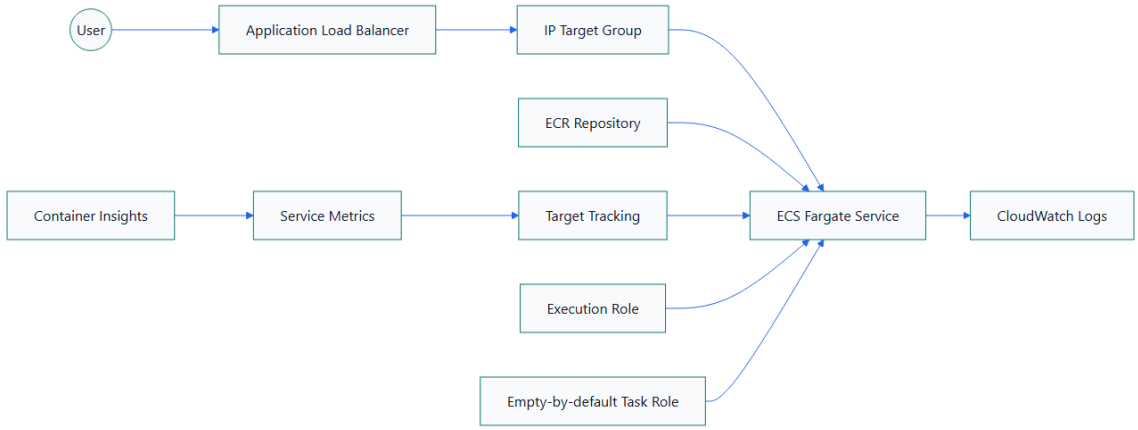

Figure 1. Private ECS Fargate Application Platform architecture

What I Implemented

- Created ECR lifecycle and image-scanning controls for application artifacts.
- Ran Fargate tasks without public IPs and allowed inbound traffic only from the ALB security group.
- Separated the ECS execution role from an application task role that has no workload permissions by default.
- Added CloudWatch log retention, ECS Container Insights, health checks and target-tracking autoscaling.

- Configured rolling deployment parameters so replacement tasks become healthy before old tasks are removed.

Important Technical Decisions

- The sample public Nginx image keeps infrastructure testing independent of application source code.
- A production release should use an immutable ECR digest.
- Application permissions are added only when a workload proves the need.
- HTTPS and hardened container settings are documented as production gates.

Security Controls

- Tasks are private and accept traffic only from the ALB.
- ECR scanning and lifecycle controls reduce unmanaged image accumulation.
- Execution and application roles are separated for least privilege.

Reliability and Operations

- ALB target health participates in deployment decisions.
- Target tracking adjusts task count based on service metrics.
- CloudWatch logs and Container Insights provide deployment and runtime evidence.

Cost and Cleanup Guardrails

- Fargate, ALB, NAT Gateway, CloudWatch ingestion and public IPv4 costs are identified.
- ECR lifecycle rules and bounded log retention control storage growth.
- Cleanup includes ALB, NAT, ENIs, ECR images and log groups.

Validation Evidence

The following local checks passed on 2026-08-13:

```
terraform fmt -check -recursive
terraform init -backend=false
terraform validate
```

Scope boundary: Format, initialization and validation passed on 2026-08-13. No plan or AWS deployment is claimed. The default Nginx image uses HTTP and a writable root filesystem; both are explicitly unsuitable as final production settings.

Delivery and Verification Runbook

- 1. Build and scan an immutable image
- 2. Publish to ECR
- 3. Register task definition
- 4. Roll out through ECS
- 5. Validate ALB health and service events
- 6. Review logs and scaling alarms

- 7. Rollback task definition if validation fails

Technology Stack

Terraform | Amazon ECS | AWS Fargate | Amazon ECR | Application Load Balancer | CloudWatch | Container Insights | Application Auto Scaling

Key Learnings

- Infrastructure evidence must distinguish code validation, plan review and live deployment.
- Security, reliability, cost and cleanup decisions should be documented before apply.
- A useful platform lab includes verification and rollback thinking, not only resource declarations.