



Enterprise AWS Cloud Architecture & Web Security

Implementing a high-performance, multi-layered AWS application infrastructure with Next.js, Strapi CMS, AWS WAF security, and centralized operational logging for a real estate leader.

Overview

QBurst engineered and deployed a modern AWS cloud architecture for Bozzuto.com, powering a Next.js frontend and Strapi CMS through containerized microservices and multi-layered security controls.

- **Containerized Decoupled Architecture:** Hosted Next.js and Strapi CMS on Amazon ECS with AWS Fargate, managed via Application Load Balancers (ALB) for high availability and seamless scalability.
- **Comprehensive Web Security & Optimization:** Configured Amazon CloudFront and AWS WAF with managed protections against bot traffic, SQL injection, DDoS attacks, and custom rate limits, paired with Lambda-based image optimization.
- **Unified Visibility & Escalation Workflows:** Established multi-tier monitoring across ECS tasks, RDS memory/CPU limits, and UptimeRobot availability, providing structured incident triage and escalation from Tech Support to DevOps.



Client Profile

Bozzuto is a diversified real estate company specializing in the development, construction, acquisition, and management of high-end residential communities across the United States. With over 4,000 employees and more than 100,000 residences under management, the company is committed to creating vibrant, community-focused living experiences.

Multi-Layer Security and Operational Visibility

Delivering enterprise real estate web applications while protecting against public traffic vulnerabilities presented distinct operational requirements.

- **Application Availability & Capacity Risks:** Needed continuous monitoring of Next.js frontend and Strapi CMS container health to prevent capacity degradation during traffic spikes.
- **Public Web Vulnerabilities & Bot Traffic:** Required safeguarding public endpoints against automated bots, SQL injections, malicious IP addresses, and DDoS threats.
- **Content Delivery & Database Performance:** Demanded maintaining Strapi CMS database health and application caching without compromising image asset delivery speed.
- **Fragmented Operational Visibility:** Lacked unified cross-layer logging and alerting to detect frontend outages, ECS task count changes, or RDS memory pressure.

QBurst Solution

We designed and implemented a managed AWS cloud environment featuring containerization, automated CI/CD pipelines, and proactive threat inspection.

- **Containerized Workload Hosting:** Deployed Next.js and Strapi CMS on Amazon ECS with AWS Fargate, utilizing Application Load Balancers to decouple frontend and CMS operations.
- **Perimeter Security & Edge Delivery:** Managed DNS resolution via Amazon Route 53 and global asset distribution with Amazon CloudFront, protected by an AWS WAF Web ACL containing managed rules for bot control, IP reputation, and rate-based blocking.
- **Data Services & Serverless Image Optimization:** Deployed Strapi data layers on Amazon RDS with Amazon ElastiCache for Valkey caching and Amazon S3 object storage, using AWS Lambda and API Gateway for real-time image optimization.
- **Automated CI/CD & Preview Environments:** Implemented GitHub-based CI/CD workflows targeting Amazon ECR and ECS task definitions, leveraging AWS Amplify for frontend pre-release previews.

- **Centralized Logging & Analytics Pipelines:** Streamed container, ALB, CloudFront, and WAF logs to CloudWatch Logs and S3, routing CloudFront log data via Amazon Kinesis Data Firehose to Microsoft Clarity for user behavior insights.

Technical Highlights

- **Compute & Frontend:** Next.js, Strapi CMS, AWS Fargate, Amazon ECS, AWS Amplify.
- **Security & Edge:** AWS WAF, Amazon CloudFront, Amazon Route 53, Application Load Balancers.
- **Data & Image Processing:** Amazon RDS, Amazon ElastiCache for Valkey, Amazon S3, AWS Lambda, Amazon API Gateway.
- **DevOps & Monitoring:** Amazon ECR, GitHub Actions, Amazon CloudWatch Logs, Kinesis Data Firehose, UptimeRobot, Microsoft Clarity.

Scalable Availability and Significant Cost Savings

- **Proactive Capacity Guardrails:** Configured CloudWatch alarms to flag low-capacity risks (when frontend or Strapi running tasks drop to ≤ 3) and high-load thresholds (≥ 15 tasks).
- **Database Health Monitoring:** Set threshold alerts for RDS CPU utilization ($> 80\%$) and freeable memory ($\leq 2\text{GB}$) to prevent database query bottlenecks.
- **Automated Threat Mitigation:** Blocked malicious web traffic, scrapers, and exploit attempts at the edge using custom and managed AWS WAF rules.
- **Streamlined Support Escalation:** Provided Tech Support with defined CloudWatch log views and UptimeRobot email alerts, establishing a clear tier-2 escalation path to DevOps.