

Facilitating DevOps with Application Modernization & Architecture Reboot

The Client

We have a client who won Gartner® Peer Insights™ Customers' Choice for content marketing platforms. Their platform helps bring brands and their customers closer through high-quality relevant content. Companies can also use the platform to produce and disseminate their own content to improve new customer acquisition, build brand loyalty and sustain interest in what the company does, and how it does it. They also maintain a digital publication which offers interesting and original content for public consumption on topics related to content marketing. Our client also owns a marketing analytics software that helps marketers compute ROI and improve outcomes on 18 integrated digital and social channels.

Project Goals

Client's IT infrastructure needed to be migrated and modernized. In addition, they were facing challenges in deploying, maintaining, and supporting its existing applications. The company needed to revamp its infrastructure to facilitate automation and DevOps for ongoing upgrades.

The Solution

AALC engineers team worked with client's technical leadership and the Product owner team to develop a modernized business solution which included the following steps: The application was containerized—i.e. a form of virtualization wherein the application is run in isolated spaces called containers. Containers share OS (unlike Virtual Machines). Everything the application needs to run, i.e. libraries, configuration files, etc., is encapsulated in its container. As a result, the containerized app can run on different types of infrastructure, from bare metal to VMs without any need to refactor it to the particular environment.





Client's application was packaged and bundled as a Docker image before being deployed in Kubernetes (aka, K8s) clusters. This will enable better management, such as rollouts/rollbacks, service discovery, load balancing, storage orchestration, and self-healing (This refers to automatically replacing and restarting containers that fail) and rescheduling containers when nodes die.



Automation and version control are at the heart of all DevOps practices, and to enable this we adopted GitHub as our repository for code and configuration files. These files are also used to implement CI/CD. GitHub Actions would be used to generate Java (JAR) files, which will be packaged as Docker images along with dependent libraries and set up for automated deployment to AWS Auto Scaling Groups on the client's cloud. (WAR/JAR files are digitally signed to signify the origin of source code, making it easy for developers to identify version, test, and deploy the web application).



To save time that would otherwise be spent in resource management, we templated setting up of resources using AWS CloudFormation an Infrastructure as Code service that takes care of provisioning and configuring resources, so you don't have to individually do it each time. CloudFormation eliminates manual errors, speeds up, and simplifies application development, scaling, and application integration.



As part of the testing protocol, we implemented Selenium, an open source testing tool to synchronize and orchestrate tests as per pre-set triggers, which were defined in the CI/CD delivery pipeline.



To ensure security was given its just due, we implemented OWASP, an open-source security testing tool. OWASP helps identify diverse vulnerabilities before fresh releases went into production. Monitoring and security go hand in hand, and to enable thorough security, we enabled automated alerts by setting up GitLab with Slack. This would automatically share the status of CI/CD deployments and monitor for failures, reducing the impact on the business. CloudWatch metrics and logs were also used to monitor the system.



From a cost-optimization perspective Lambda functions were written to scale the environment up and down



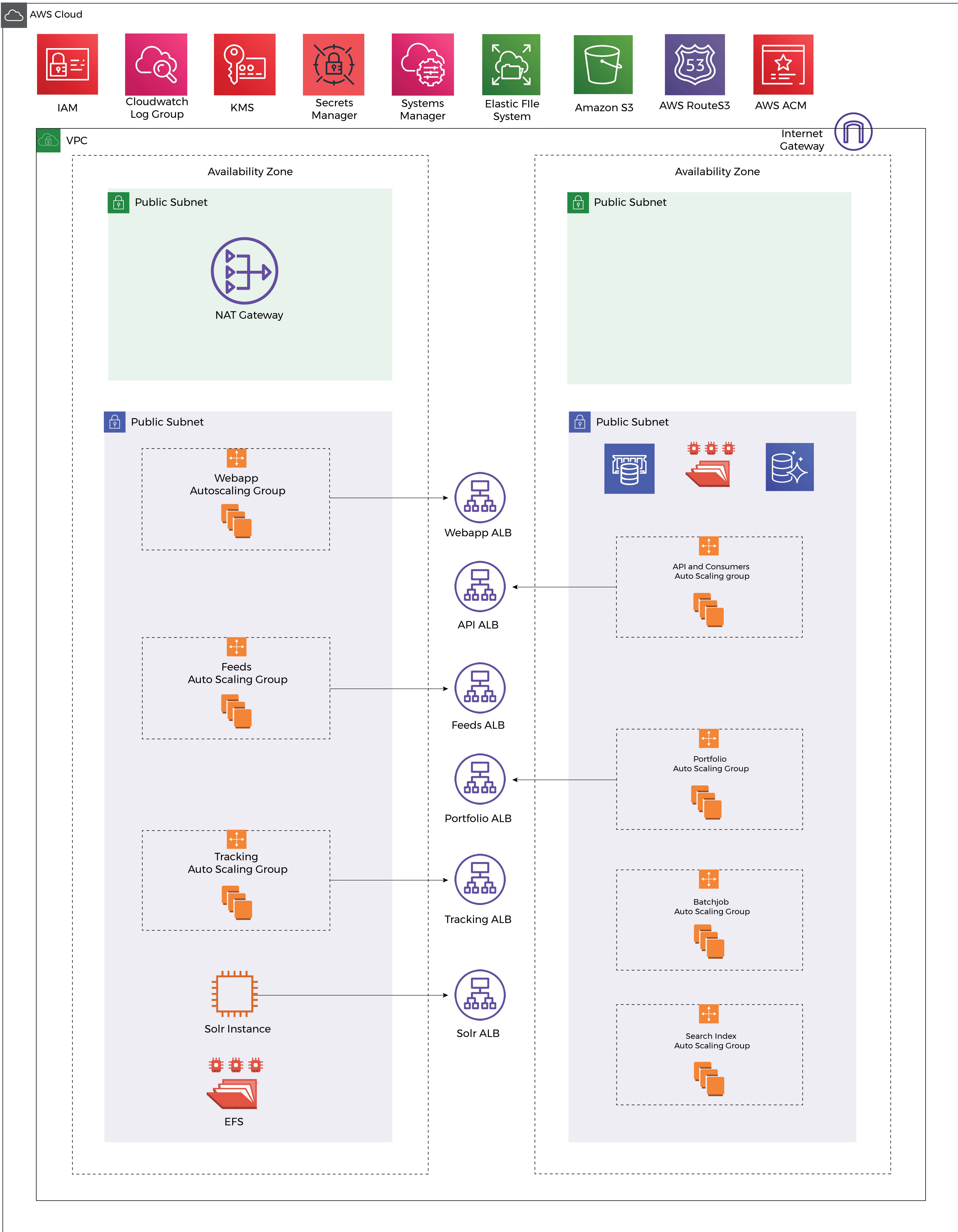
Dashboard was set up for operations and monitoring.



Architecture

Standard 3 Tier architecture was used to save information on clients securely in the Cloud. Clients can log in with appropriate IAM security, Plus Group Rules and Security.

Within AWS architecture, we utilized Amazon’s Elastic Load Balancers combined with Auto Scaling Groups to balance the traffic across multiple instances and automate the creation/termination of instances across multiple Availability Zones. A number of components were also split out from the current infrastructure to make it more modular and fault-tolerant, while assuring redundancy across several geographic locations.



Tech Stack

- Amazon Simple Storage Service (Amazon S3)- S3 is an object storage service. To store the images formatted by the openross application (a fast,open, and dynamic open source image resizer) S3 buckets were used.
- Amazon Elastic File System (EFS) - EFS is a regional service that stores data across multiple Availability Zones ensuring high availability and durability.
- Amazon CloudWatch Log Group - Amazon CloudWatch Logs were used to monitor, store, and access application logs.
- AWS Application Load Balancer - ALB was used for distributing incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones.
- AWS Route53 Hosted Zones - Route 53, highly available and scalable DNS web service, was used for routing internet traffic to client's applications.
- AWS Secrets Manager - All of client's application secrets or properties are stored in AWS Secrets Manager.
- AWS Key Management Service - an AWS managed service was used to create, encrypt, and control master keys used to encrypt data.
- Amazon ElastiCache (Redis) - Amazon ElastiCache is a Redis-compatible in-memory data store service. It was used for session management in a client's web application.
- AWS Certificate Manager - It is a service that lets you easily provision, manage, and deploy public and private SecureSockets Layer/Transport Layer Security (SSL/TLS) certificates for use with AWS services and internal connected resources.
- AWS System Manager Session Manager - Session Manager is a fully managed AWS Systems Manager capability that lets users manage their EC2 instances, on-premises instances, and virtual machines (VMs) through an interactive one-click browser-based shell or through the AWS CLI.
- Amazon Identity and Access Management – It was implemented to manage access to AWS services and resources securely.

Application Modernization: Steps & Benefits

The Data Layer - AWS Relational Database Service (RDS) was used for the data layer in conjunction with AWS Aurora, a MySQL-compatible Relational Database Management Service which was designed from the ground up to run on AWS.

Deployment - Application Instance creation, launch, and updating is managed using AMI (AmazonMachine Image.) In order to update AMI i.e., modifying WAR/JAR files present in the AMI, AMI bakery is used. It also provides support for deploying it into the respective auto-scaling group. AMI deployment is done in a blue-green fashion where a new instance will be launched and only after launch will the older or previous instance will be terminated. AWS Lambda function has been set up in the environment. The Lambda function is responsible for scaling up or scaling down autoscaling groups. It can be scheduled to scale up or scale down at a specific rate.

Version Control

Git VCS was used to manage the versioning of the infrastructure. a Git repository, with all of the relevant setup & configuration scripts, along with any relevant documentation, provides detailed information on how to recreate the infrastructure from scratch. This, combined with CloudFormation, enables the automated creation of almost all architected AWS services.

Migration Benefits



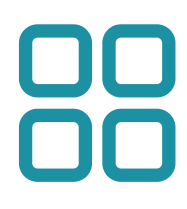
Significant improvement in Uptime, which can be monitored with 3rd party tools and confirmed. Implementation of the new pipeline helps to reduce the time of the deployment from 11 mins to 5 mins.

No single point of failure

Ability to innovate - Being on AWS opens up an array of new services and technologies, from new storage engines like Redshift or services like Amazon Machine Learning or Lambda, earlier implementation was taking hours to send notification during data processing & involved manual efforts to reboot k8s pods. AAIC automated this manual task using a lambda service which reduced the resolution time to 15 mins.

Integration to other systems like Wordpress and third party APIs are simple with this devops solution.

Cost Optimization

-  With the deployment and output of AWS Migration Evaluator, our client was able to see detailed reports of their inventory.
-  Utilization of resources was analyzed, and this helped in right-sizing the company's AWS solutions. Some of their existing resources were over-provisioned and right-sizing their resources resulted in significant cost savings.
-  Approx 30-35% reduction in the cost by selecting appropriate instances and deleting unused resources in AWS. Earlier the pods requested cpu and memory was higher which was resulting in high configuration spot instances, optimized solution given by AAIC resulted in this cost reduction.

**BOOK A MEETING WITH
OUR ARCHITECT**



ABOUT AAIC

We are automation experts, with a majority(> 60%) of our workforce AWS-certified. We assist you in applying intelligence to the Cloud and DevOps, as our name suggests.

Our AWS certified experts create high-performing cloud apps by utilizing intelligent components and smart integrations to accelerate your digital transformation journey.

Copyright © 2022 AppliedAIConsulting



+91-9923354746



connect@appliedaiconsulting.com



www.appliedaiconsulting.com



DevOps
In-a-box

Fastrack your DevOps
implementation



Applied AI
Intelligence Delivered