MIGRATING LEGACY UNDERWRITING SYSTEM TO CLOUD NATIVE ARCHITECTURE FOR SCALABILITY AND AUTOMATION

CAPABILITIES SHOWN

- Advisory Services
- Cloud Migration
- Digital Transformation
- Cloud Engineering
- Observability & Monitoring

ABOUT THE COMPANY

We are an ISO 9001:2008, 9001:27001, 20000-1:2018, CMMI Level 3, EDWOSB providing superior, affordable and innovative business management and information technology services to federal and private sector clients nation-wide. We specialize in Software Development, Business Intelligence (BI), Data Management, Data Governance, Cyber Security, Data Quality, Master Data Management, Advanced Data Analytics and Cloud Services.

13454 Sunrise Valley Dr. Suite 240, Herndon, VA 20171 +1-571-222-464 info@navitastech.com

ABOUT THE CUSTOMER – SECONDARY MORTGAGE COMPANY

The Financial Company is a leading source of mortgage financing in all markets and at all times. They ensure the availability of affordable mortgage loans. The financing solutions They develop make sustainable homeownership and workforce rental housing a reality for millions of people.

MIGRATION OF LEGACY UNDERWRITING SYSTEM TO CLOUD NATIVE ARCHITECTURE FOR SCALABILITY AND AUTOMATION FOR CUSTOMERS

Navatis tech partnered with the secondary mortgage provider to build the industry-leading underwriting system that helps lenders efficiently complete credit risk assessments to establish a home loan's eligibility for sale and delivery with easy-to-use, powerful tools.

Key Challenge

- Migrate Legacy application to a modern cloud native application
- Manage and deploy changes to the solution rapidly and significantly reduce deployment error rate
- Legacy tech stack with security vulnerabilities and scalability issues

SOLUTION

Clients Underwriting system is a high throughput, low latency transaction processing use case where the transaction arrival rate is over 100 TPS. The application was rearchitected to follow cloud native principles using a microservices architecture. Microservices are built using Spring Boot and are deployed to containers ECS/Fargate. A group of microservices bounded by a common business context form a domain and each domain has its ECS cluster. Lambda functions are used for batch processing use cases and event streaming using Kinesis. Each microservice has DynamoDB as the data store and the flow between microservices is managed by a centralized orchestrator.

SERVICES USED

- Elastic Container Service (ECS):
  - Used to deploy microservices as containers.
  - Utilized Fargate for serverless container deployment.
- Lambda:
  - Employed for batch processing and event streaming using Amazon Kinesis.

- DynamoDB:
  - Used as the data store for each microservice.
  - Also used for storing data for containerized workloads.
- CloudWatch:
  - Set up alerts for monitoring application health, performance, availability, and other metrics.
  - Integrated CloudWatch with Splunk for log monitoring.
- Amazon Kinesis :
  - Used for event streaming and real-time data processing.
- AWS Inspector & GuardDuty:
  - Employed for security monitoring and threat detection.
- Amazon S3:
  - Used as a landing zone for data from different sources.
  - Enabled cross-region replication for data redundancy.
  - Utilized for storing container images, data storage, backups, and more.
- AWS Config:
  - Utilized along with CloudTrail and CloudWatch for detecting drift from baselines.
- AWS API Gateway:
  - Fronted services with API Gateway for external access and interaction.
- Amazon Aurora (RDS):
  - Used for database workloads.
- AWS Identity and Access Management (IAM):
  - Implemented granular access controls and permissions.
  - Used for federation with ADFS for console access.
- AWS Secrets Manager:
  - Employed for centralized secret management.
- Amazon EKS (Elastic Kubernetes Service):
- AWS Transfer for SFTP:
  - Utilized for secure data transfer to AWS.
- Amazon ECR (Elastic Container Registry):
  - Used to store versioned container images.
- Terraform:
  - Utilized for Infrastructure as Code (IaC) deployments.
- Ansible:
  - Used for OS-level and application-related configuration management.
- AWS Backup:
  - Employed for backup and recovery purposes.

THE BENEFITS

- **Transformed the application to a highly available and scalable application:**
  - Achieved 4-9s availability for the application through multi-region active/standby architecture.
  - Reduced Recovery Point Objective (RPO) to 15 minutes and Recovery Time Objective (RTO) to 40 minutes.
  - Implemented multi-AZ deployment for ECS clusters and Aurora RDS instances.
  - Utilized AWS Elastic Beanstalk for application hosting with auto-scaling across 4 availability zones.
- **Managed and deployed changes rapidly and reduced deployment error rates:**
  - Implemented full automation using AWS CodePipeline and Jenkins.
  - Reduced deployment error rates by automating unit tests, static code analysis, and security scans.
  - Utilized Infrastructure as Code (IaC) with Terraform for consistent and error-free infrastructure provisioning.
- **Improved DevOps practices and unlocked business agility and innovation:**
  - Implemented DevOps practices by automating CI/CD pipelines for continuous integration and delivery.
  - Adopted AWS ECS/Fargate for containerized workloads, enabling rapid deployment and scaling.
  - Leveraged AWS Lambda functions for batch processing and event streaming, enhancing flexibility.
  - Utilized serverless technologies to reduce operational overhead and enable rapid innovation.
- **Provided a safe, repeatable, and easy means to release features:**
  - Utilized AWS CodePipeline for automated release management, ensuring consistency and repeatability.
  - Enabled feature flags and dynamic configuration using Apigee API Gateway for controlled feature releases.
  - Reduced time-to-market for new features by automating testing, validation, and deployment processes.