Case Study | DevOps implementation in Fintech

Overview

Alpha fortress LLC is a cross-border payment company focused on paying out transactions in Africa. The platform developed in Roby on rails and utilizing a custom blockchain and multiple cryptocurrencies to facilitate fast and secure payments processing. The customer-facing brand name of N2Xpress is **the largest** remittance services provider in Canada.

The system process over 300 transactions a minute with a total daily value of one million USD. The API backed is utilized by multiple cross-border companies like Smallworld, XpressMoney, ResourceBridge LLC, Globex, TravelX and the list is increasing. The unique ability to pay into Africa mobile wallets and bank accounts makes them one of the most reliable and fastest payout providers in the continent

Tool Set

Cloud - AWS.
Git - BitBucket.
Build - Jenkins.
AWS ECS - Container hosting.
AWS Secrets manager - Secret manager.
AWS EC2 - Virtual Instances.
AWS SQS - Queue.
AWS ALB + Global Accelerator.
AWS Elastic file system.

About Calance

Calance is an IT Services firm operating in the United States, Canada and India. We provide Consulting, Application Development & Systems Integration, Managed Services, IT Staffing and pre-built Products & Solutions. With a long history of success in IT Services, we help clients tackle many of today's technology challenges. For more information, please contact us at info@calance.com Visit our website at www.calanceus.com

Challenges

The development was done in-house using Ruby on rails (v2.3) and the server requirements for running the monolith application were very high, being a financial company downtime of the application is not acceptable. The standard approach to put the application under a load balancer to multiple servers was not possible as this could introduce race conditions where the same transaction is paid out multiple times. The wallets for crypto currencies were hosted on machines and communicated over open APIs introducing security concerns. The company was paying more than thirteen to fifteen thousand USD to AWS. This was a cost that needs to be reduced.

The request from the customer was

1. Increase the reliability of the services.
2. Have a zero-downtime infrastructure with failover & blue-green deployments.
3. Reduce costing of AWS
4. Improve on security where applicable.

Our Solution

We introduced multiple solutions for meeting the challenges of the client.

Increase reliability of the service/Application

We split the application into multiple services and introduces a RabbitMQ for queuing services. All inter-application services communicate using dedicated channels on RabbitMQ. For handling the issue of race conditions, we set up timed single-use signatures computed using API Secret and API secret, using this method the correspondents utilizing the services encrypted the payload using the secret key. This protected the application from replay attacks as well as accidental/malicious API requests hitting the platform. Now even if multiple services are up one message on the queue will only be processed by one instance of the application**.**

Have Zero downtime

With the introduction of the micro services and the single-use API post, we were able to deploy the services to Amazon Container Services and have a self-healing infrastructure. This also enabled us to have blue-green deployments which in turn enabled us to provide zero downtime for application upgrades. We also leveraged vercel.com to host the front-end code of the application. To increase the speed of the API responses we introduced AWS Global Accelerator.

Reduce cost

We moved the application from EC2 instances to ECS services, the only service that is now running on a dedicated host is RabbitMQ. The rest of the services we containerized including the cryptocurrency wallet services and the blockchain was mounted on S3 using FUSE filesystem. By updating the application and moving to container technologies we were able to bring down the AWS costs from fifteen thousand to three thousand.

Results

By making the changes in the application and introducing new technologies were able to meet all the client's expectation. There are many improvements that the client saw with the new architecture other than the requirements mentioned. Lower response times to API requests have enabled the client to accept and process even more transactions than before. The cost of implementing the changes will be re-couped by the client with 12 months of going live with the new architecture just from the reduced AWS spends alone.