CloudPlexo
Modernize and secure all processes.

AWS Case Study: Re-architecting Infrastructure from Monolithic to Microservices for Convexity

Background

About Convexity:
Convexity Technology Limited stands as a prominent blockchain consultancy firm, not only in Nigeria but also across Africa. At the forefront of their offerings is their groundbreaking solution known as the ‘Convexity Humanitarian Aid Transfer Solution.’ This innovation is designed to establish comprehensive visibility and transparency in the distribution of humanitarian aid throughout Nigeria and Africa.

This solution seamlessly connects all stakeholders involved in the aid distribution chain, spanning from donors to beneficiaries, leveraging the power of blockchain technology to ensure efficiency and accountability.

The Convexity Humanitarian Aid Transfer Solution (CHATS) represents the flagship product of Convexity. With a dedicated product team boasting over two decades of collective experience in both humanitarian and financial sectors, CHATS is purpose-built to tackle distribution challenges associated with Cash & Voucher Assistance (CVA) in Africa. By harnessing blockchain technology, CHATS aims to combat fraud in aid administration and enhance the welfare of vulnerable individuals and households.

The Convexity team is actively developing a digital cash and voucher transfer platform underpinned by blockchain technology, facilitating the seamless movement of funds between donors, NGOs, and other relevant parties. Blockchain's inherent immutability and transparency empower donors and auditors to monitor aid distribution processes closely.

CHATS ensures that all aid recipients possess a wallet account linked to a verified individual, offering multiple channels for aid redemption, including USSD, SMS vouchers, QR code paper vouchers, and NFC cards for beneficiaries without internet access or smartphones. Moreover, aid distribution is meticulously geo-fenced and mapped to prevent unauthorized disbursements, thereby providing donors with real-time visibility into their funded projects.

Crucially, the CHATS system is fortified with a non-custodial smart contract cryptocurrency fund management system, allowing donors to intervene in case of

suspected malpractice during the disbursement process by pausing or retracting funds. This proactive measure further solidifies trust and accountability within the aid ecosystem, fostering impactful humanitarian initiatives across Nigeria and Africa.

Challenges

1. Scalability Issues: The monolithic architecture made it difficult to scale individual components independently, leading to resource inefficiencies and performance bottlenecks.
2. Deployment Bottlenecks: Any change or update required the entire application to be redeployed, increasing downtime and the risk of introducing bugs.
3. Limited Agility: The tightly coupled components slowed down the development process and hindered the ability to innovate rapidly.
4. Resource Management: Inefficient resource utilization resulted in higher operational costs.

Objectives

1. Improve Scalability: Enable independent scaling of different services to better handle load variations and optimize resource usage.
2. Enhance Agility: Allow for faster deployment and more frequent updates with minimal downtime.
3. Optimize Resource Utilization: Reduce operational costs by leveraging AWS's flexible infrastructure and services.
4. Increase Reliability: Ensure high availability and fault tolerance for critical financial services.

Solution

Convexity partnered with CloudPlexo expertise on AWS to re-architect their monolithic application into a microservices architecture leveraging CloudPlexo’s DevOps service. The solution involved the following key steps:

1. Assessment and Planning

CloudPlexo AWS Solution Architects conducted a thorough assessment of Convexity's existing infrastructure. A detailed plan was developed to transition to a microservices architecture, considering the following aspects:

- Identifying and defining microservices boundaries.
- Selecting appropriate AWS services for hosting and managing microservices.
- Establishing a robust CI/CD pipeline for continuous integration and deployment.

2. Service Decomposition

The monolithic application was decomposed into discrete, independently deployable microservices. Key functionalities such as user authentication, analytics, reporting, and notification services were separated into individual microservices.

3. Containerization with Amazon ECS

Each microservice was containerized using Docker. Amazon Elastic Container Service (ECS) was chosen to orchestrate and manage these containers, providing:

- Scalability and load balancing.
- Simplified management of containerized applications.
- Integration with other AWS services.

4. API Gateway and Service Mesh

Amazon API Gateway was implemented to provide a unified entry point for all microservices, enabling:

- Traffic management.
- Authorization and access control.
- Monitoring and logging.

AWS App Mesh was used to manage the communication between microservices, ensuring:

- Secure and reliable service-to-service communication.
- Observability and tracing.

5. Data Management with Amazon RDS and DynamoDB

Data storage was re-architected to use a combination of Amazon RDS for relational data and Amazon DynamoDB for high-performance NoSQL needs. This setup provided:

- Scalability and high availability.
- Optimized query performance.
- Cost-effective storage solutions.

6. CI/CD Pipeline with Github Actions

An automated CI/CD pipeline was established using Github Actions for Continuous integration and delivery.

- Automated testing and deployment.
- Reduced deployment times and errors.

7. Monitoring and Logging with Amazon CloudWatch

Amazon CloudWatch was implemented to monitor application performance and health. This included:

- Real-time monitoring and alerts.
- Detailed logging and metrics collection.
- Insights for proactive troubleshooting and optimization.

Results

Scalability: Convexity can now independently scale each microservice based on demand, significantly improving resource utilization and performance.

Deployment Efficiency: Deployment times were reduced by 70%, with the ability to deploy updates to individual microservices without affecting the entire application.

Agility: Development cycles became faster and more efficient, allowing Convexity to roll out new features and updates more frequently.

Cost Optimization: The move to AWS and microservices architecture resulted in a 30% reduction in operational costs due to better resource management and scalability.

Reliability: The new architecture provided enhanced fault tolerance and high availability, ensuring consistent service delivery to Convexity's clients.

Conclusion

Re-architecting from a monolithic to a microservices architecture on AWS enabled Convexity to overcome its scalability, deployment, and resource management challenges. This transformation not only improved their operational efficiency and reduced costs but also empowered them to innovate rapidly and deliver a superior user experience to their clients.

AWS Services Used:

- Amazon ECS
- Amazon RDS for MySQL
- Amazon ECR
- Amazon X-ray
- AWS App Mesh
- Amazon CloudWatch

Keywords: AWS, Microservices, Financial Services, Cloud Computing, Scalability, CI/CD, Containers, API Gateway, Service Mesh, Data Management, Monitoring