CASE STUDY

Modernizing a Storyboard Application with AWS Cloud

Author:
Enis Avdičaušević, Senior Project Manager

At a Glance:

Comtrade 360 successfully modernized a legacy storyboard application for a leading animation studio, transforming it into a cloud-based solution that enhances scalability, collaboration, and maintainability while introducing advanced features to meet modern animation needs.

Industry:

Animation and Entertainment

Technology:

- AWS Cloud Services
- Docker
- Kubernetes
- React
- TypeScript
- Tailwind CSS
- Microservices Architecture
- AWS Elastic Kubernetes Service
- AWS Batch
- AWS RDS
- AWS EFS
- AWS CloudWatch

Introduction

A storyboard application is an important development tool within the animation industry as it lays down the base for telling a story. Yet, legacy client-server storyboard applications have a few key drawbacks, such as the inability to scale well or collaborate easily and heavy maintenance overhead. This case study discusses the rearchitecting of a legacy storyboard application into a more scalable cloud-based solution to serve multiple studios, support collaboration, improve maintainability, and develop new features.

Background

The legacy storyboard application was a desktop-based, client/server solution used by one of the largest animation studios. It was effective in its time, but it faced many challenges with growing demands in today's animation production process: limited scalability and performance, lack of real-time collaboration, high maintenance complexity, and inability to add new functionalities easily. All these limitations started to emerge when the studio expanded, working on more projects and collaborating with external partners.

Objectives

The objectives of the modernization project were as follows:

1. **Cloud Solution:** Transform the application and migrate it to a cloud platform to support simultaneous use by several users and studios, while also leveraging modern cloud-based architecture and practices.

2. **Scalability:** The solution must support scaling by handling different workloads and user demands efficiently.

3. **Collaboration:** Enable real-time collaboration that allows several users to work on the same project at the same time from anywhere.

4. **Support and Maintenance:** Provide simplified and efficient support and maintenance.

5. **Feature Improvements:** Add advanced drawing tools, integrations, and new functionalities that would suit modern animation needs.

Solution

The implementation phase of the modernization of the storyboard application involved multi-faceted activities aimed at changing the existing on-premises system into a modern, cloud-based solution. First and foremost, as part of the modernization, the application had to be migrated to cloud infrastructure. Among many choices, AWS was selected because of its comprehensive set of services and robust infrastructure.

The application was re-architected from a monolithic application into a microservices architecture. It was broken down into independent services, each handling a specific functionality. This decomposition allows easier management, scalability, and deployment.

Containerization was a crucial aspect of this transformation. Every microservice was wrapped in a Docker container and provided with a consistent runtime environment for development, testing, and production. Kubernetes clusters were managed by AWS Elastic Kubernetes Service, reducing the operational burden of maintaining Kubernetes infrastructure.
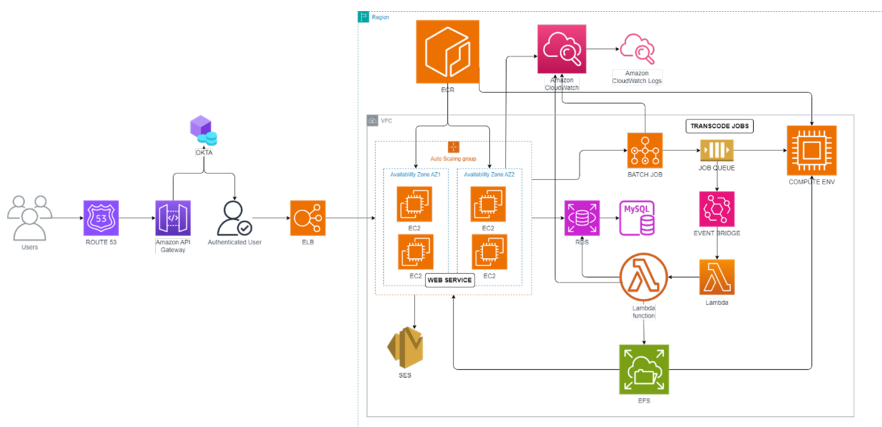
Ingress (ROUTE 53 / API Gateway / OKTA / ELB)

The backend API ingress is based on Route53 DNS, coupled with the AWS API gateway handling incoming user API requests and authenticating them against OKTA. Once authenticated, the incoming requests are load balanced (ELB) across web-service EC2 instances in multiple availability zones.

Web Service (EC2 / RDS)

The backend API service runs across multiple EC2 instances split over multiple availability zones. The service is responsible for handling all incoming requests and maintaining an RDS database where the object state and versioning information are kept. All service EC2 instances have R/W access to the same network file system (EFS) where the raw media object files are stored. The web service is also responsible for maintaining the job scheduler and monitoring job progress used for various media operation actions.

Job controller (Batch / Compute Env / Event Bridge / Lambda)

The job execution environment is based on AWS Batch, coupled with matching compute environments for different job types. The media operation actions are split into multiple jobs, which are scheduled across multiple job queues and executed on the matching compute environment. Each compute environment has direct R/W access to the EFS file system for media object access and output storage. Job execution is monitored using the AWS Event Bridge, which, on job completion, triggers a lambda function to perform object state updates in the RDS system and perform any remaining job cleanup actions.

Network File System (EFS)

The AWS EFS filesystem holds all raw media object files in a specific folder hierarchy. Multiple backend systems have direct access to it, which reduces the amount of data passing between those modules, allowing for easier scale-out and state recovery. The EFS is highly optimized for multi-client access, which matches our usage model.

Monitoring (CloudWatch)

The solution uses AWS CloudWatch to monitor all our backend modules and KPIs. We make use of the built-in metrics for AWS native services and additionally expose our service-specific metrics to CloudWatch. The main metrics and KPIs can be monitored with a set of CloudWatch dashboards with integrated alarm triggers for specific KPIs.

Frontend

For a storyboarding web application, the tech stack we used includes React for its component-based architecture and efficient rendering, TypeScript for type safety and improved developer experience, and Tailwind CSS for rapid and customizable styling. This combination ensures a scalable, maintainable, and highly interactive user interface, making it ideal for the dynamic needs of storyboarding.

Results

The modernization of the storyboard application using AWS cloud services brought considerable improvements. The application can dynamically scale to meet fluctuating workloads, ensuring optimal performance and cost efficiency. Real-time collaboration features increase workflow efficiency, allowing creative teams to collaborate on projects in a single workspace from different locations. Centralized management and advanced monitoring tools increase support efficiency by one-third, reducing response times and hence improving user satisfaction. Automated CI/CD pipelines and IaC reduce downtime, avoiding issues that had complicated the maintenance process before and increasing the speed at which updates and features could be released. In particular, the modular microservices architecture and event-driven architecture have boosted the speed of development and deployment of new features, keeping the application competitive and innovative. The new tech stack with modern tools and approaches allows faster development of new functionality and ideas.

Conclusion

The storyboard application was successfully modernized into a cloud-based solution, which underlines the importance of leveraging modern technologies to address the evolving needs of the animation industry. Modernization provided enhanced versatility with added scalability, support, and collaboration, along with many new features that could be used by many studios. Done right through careful planning, strategic technology selection, and user-centric design, the project not only revitalized the storyboard application but also set a base for further innovations and enhancements.

As a leader in application modernization, Comtrade 360 is ready to partner with you to transform your legacy systems into innovative, scalable solutions. Contact us today to learn how we can help you stay ahead in the competitive landscape of your industry.