Josh™ Application Cloud Migration: AWS to GCP Case Study

This document presents a comprehensive case study on the migration of the Josh application from AWS to GCP, a critical initiative undertaken in September 2023 and delivered in 10 weeks. This migration involved a substantial infrastructure scale, encompassing 20,000 vCPUs, a 64TB database, and 2PB of storage. The transition was driven by the need to optimize costs, enhance scalability, and reduce vendor lock-in, ultimately aiming for a more agile, cloud agnostic, and high performing platform for Josh's millions of users.

Quarkwiz

MIGRATING FROM

aws **TO**

Introduction to Josh and Migration Imperatives

Josh is a prominent short video platform in India, boasting over 1 million concurrent users and 100 million Daily Active Users (DAU). Since its launch in mid-2020 on Amazon Web Services (AWS), Josh experienced exponential growth, leading to a significant increase in AWS infrastructure usage. QuarkWiz led the cloud infrastructure migration for Verse, the parent company of Josh, transitioning from AWS to Google Cloud Platform (GCP), to better support its expanding needs and leverage the robust features offered by these cloud providers.

Challenges with AWS Infrastructure

The rapid adoption of AWS cloud-native services from 2021 to 2022, while facilitating initial growth, presented two primary challenges:

- Frequent issues with AWS PaaS offerings, where customers lacked adequate control and visibility.
- Mounting AWS expenses due to increased usage.

Technical Limitations

Beyond cost, the existing architecture faced significant technical hurdles:

- Limited Scalability: The current setup struggled to efficiently handle the ever-growing data loads and user traffic.
- Vendor Lock-In: Heavy reliance on AWS-native services created a rigid ecosystem, restricting flexibility and multi-cloud strategies.

The migration aimed to address these critical issues, transitioning Josh to a more resilient and cost-effective cloud environment.

© 2025 QuarkWiz, All rights reserved.

Desired Outcomes and Pre Migration Analysis

The migration initiative was driven by clear objectives designed to transform Josh's cloud infrastructure. The primary desired outcomes were:

Cost Optimization

To optimize existing AWS resources and achieve significant billing reductions.

Architectural Modernization

To rearchitect the application to a microservice based architecture, making it agile, cloud agnostic, and highly scalable.

Reduced Vendor Dependency

To move away from AWS Cloud-native services towards enterprise or open source self managed infrastructure.

Enhanced Performance

To improve scalability and performance, capable of handling increasing user traffic and data loads.

Improved Analytics Capabilities

To leverage advanced data analytics and machine learning for better user experience and content recommendations.

Before the migration, an extensive analysis was conducted to understand the existing infrastructure and its dependencies. Key findings included:

- Approximately 3,000 VMs (applications, ECS, EMR).
- 1,000+ application load balancers.
- Critical interconnect links between cloud platforms and data centers.
- Over 30 petabytes of storage data.
- Dependencies on cloud-native services: SQS, MSK, RDS, OpenSearch, etc.
- Applications deployed on VMs accessed via load balancers.
- Monitoring by standalone Prometheus and CloudWatch; no platform for EMR/batch services.
- Lack of central logging platform; production logs managed by ELK stack and CloudWatch.
- Dependence on Athena for big data querying.
- Utilization of Lambda for batch processing tasks.

© 2025 QuarkWiz, All rights reserved.

Google Cloud

Migration Readiness and Strategic Planning

The migration to GCP required a meticulous assessment of various infrastructure components, data, and services to ensure a seamless transition. Key limitations and readiness aspects were thoroughly evaluated to formulate a robust migration strategy.

1

Non Negotiable Zero Downtime

A critical business requirement was ensuring zero downtime during the migration process to maintain continuous service availability for users.

2

Uninterrupted Production Development

The migration process had to avoid disrupting ongoing production development, ensuring a smooth transition from development to deployment.

3

Open Source/GCP Managed Alternatives

Heavy dependencies on AWS managed services necessitated exploring open source alternatives or suitable GCP managed services for a seamless transition.

A detailed migration readiness assessment was conducted, covering several critical areas:

Infrastructure Assessment

Reviewing existing AWS infrastructure, including VMs, ASGs, load balancers, and managed services, to identify dependencies and configurations.

IAM Migration

Planning the migration of users, roles, and permissions from AWS to GCP to maintain access control.

Data Assessment

Assessing the volume (30 PB), types, and sensitivity of data, including transfer requirements and governance.

Service Compatibility

Determining compatibility of AWS managed services (MSK, OpenSearch, DynamoDB, RDS, Athena, Lambda) with GCP counterparts.

Big Data Solutions

Analyzing MapReduce applications and Athena architecture for migration and optimization on GCP.

Monitoring & Logging

Reviewing existing stacks and evaluating GCP tools like Stackdriver for migration strategy.

Connectivity & Networking

Assessing existing direct connections and evaluating GCP solutions like Dedicated/Partner Interconnect.

Risk Analysis

Identifying potential risks (downtime, data loss, compatibility) and developing mitigation strategies.

Team Readiness

Evaluating team skills for GCP services and tools, providing necessary training and resources.

© 2025 QuarkWiz, All rights reserved.

Google Cloud

Migration Strategy and Planning Phases

The migration strategy was structured around clear goals and a phased planning approach to ensure a controlled and successful transition.

1 Replicate and Rearchitect Infrastructure

Successfully replicate/re-architect the existing AWS infrastructure onto GCP for seamless operational continuity.

2 Efficient Data Migration (30 PB)

Efficiently migrate 30 petabytes of storage data from AWS to GCP, minimizing downtime and ensuring data integrity and security.

3 Seamless Managed Service Transition

Smoothly transition services like MSK, OpenSearch, DynamoDB, and RDS to equivalent/alternative/cloud-agnostic services on GCP, ensuring compatibility and performance.

4 Optimized Big Data Solutions

Implement optimized big data querying and processing on GCP, leveraging native offerings like BigQuery and Dataflow.

5 Comprehensive Monitoring and Logging

Establish robust monitoring and logging stacks on GCP using tools to maintain operational efficiency and address issues promptly.

The migration planning involved three distinct phases:

Discovery and Planning Phase

- Identify Stakeholders and gather requirements.
- Define project scope, success criteria, and current infrastructure assessment.
- Create a roadmap, migration strategies, and identify/mitigate risks.

Preparation and Execution Phase

- Procure required resources; perform compatibility and dependency checks.
- Conduct team member training on necessary technologies.
- Start with non-critical system components and develop testing processes.

Validation and Switchover Phase

- Validate system migration with UAT for end-to-end functionality.
- Prepare documentation with architecture diagrams and conduct knowledge transfer.
- Monitor performance and address post-migration issues.
- Implement phased go-live (1% to 5% offloading) and document rollback strategies.

© 2025 QuarkWiz, All rights reserved.

Implementation: Phases 1 and 2

The implementation of the migration was executed in two distinct phases, starting with comprehensive preparation and solution design, followed by the actual migration execution and optimization.

Phase 1: Preparation and Solution Design

Preparation Steps

- Selection of the optimal migration strategy.
- Setup of the cloud environment with necessary services and access controls.
- Deployment of infrastructure with High Availability (HA) and Disaster Recovery (DR) setup.
- Regular internal audits for security and compliance.
- Categorization and planning for data transfer.
- Multi-layered migration approach: Infrastructure optimization, transition from managed to self-managed services in AWS, and workload migration to GCP.

Solution Design and Re-architecture

- Assessment of current AWS architecture for optimization.
- Mapping AWS services to equivalent GCP services.
- Designing networking configurations for secure cross-cloud communication.
- Architecting HA and DR solutions on GCP.
- Implementing security controls and compliance measures.
- Optimizing architectures for scalability and performance on GCP.
- Analyzing cost implications and designing cost-effective solutions.
- Designing integration components for seamless system integration.
- Implementing monitoring and logging for proactive management.

© 2025 QuarkWiz, All rights reserved.

Phase 2: Migration Execution and Optimization

This phase focused on the actual data and application transfer, system integration, and comprehensive optimization for cost and performance.

1 Data Migration

Secure transfer of 30 PB data from AWS S3 to GCP GCS using tools like Data Migration Service (DMS) and Storage Transfer Service (STS). MirrorMaker for Kafka replication and RIOT for Redis data migration.

2 Application Migration

Smooth transition of applications from AWS to GCP Kubernetes (GKE) with minimal disruption, utilizing DNS TTL reduction, gradual traffic increase, and health checks.

3 System Integration

Configuration of integration components in GKE, establishment of CI/CD pipelines, and adoption of a canary approach with Global Traffic Manager (GTM) for gradual rollouts and quick rollbacks.

4 Optimization & Modernization

Adoption of a mix of On-Demand, CUD, and Preemptible instances for cost-performance balance. Significant shift from AWS cloud-native to open-source/GCP native services. Containerization of applications using Docker and deployment on GKE.

Movement from Cloud Native to Open Source

1 Database

- Migrate from RDS to CloudSQL / MySQL
- DynamoDB to Cassandra.

2 Analytics

- Transition from ElasticSearch to OpenSearch
- Athena to Trino
- QuickSight to ClickHouse.

3 Messaging

- Move from Kinesis to Strimz Kafka
- Firehose to RabbitMQ
- SQS to an Open Source Queue System
- MSK to Kafka.

4 Computing & Monitoring

- Replace Lambda with Python APIs
- CloudWatch with Prometheus
- SageMaker with a Ray Cluster
- EMR with Spark.

5 Networking

- Swap Load Balancers for HA Proxy/GLB.

6 Caching & Streaming

- Migrate ElastiCache to Redis
- EMR to Spark for streaming workloads.

Google Cloud

Key Components, Monitoring, and Challenges

The successful migration hinged on the strategic use of specific GCP components, robust monitoring, and proactive management of both technical and organizational challenges.

Core GCP Components Used

- Compute: Google Compute Engine (GCE)
- Containerization: Google Kubernetes Engine (GKE)
- Networking: Global Load Balancers, Cloud DNS, VPC, VPC Peering, Hub Network
- Data Services: GCS Object Storage, BigQuery
- Monitoring: Prometheus, Grafana, Metric Explorer
- Logging: Cloud Logging

Database & Pipeline Ecosystem

- Databases: MySQL, PostgreSQL, Cassandra, Redis, ElasticSearch, Obelix
- Pipelines: Kafka, Kafka REST, Lambda (Python-based APIs), Airflow, BigQuery, Spark, GCS Bucket
- APIs: Developed using Java, Python, Go, Rust, Nginx frameworks

Monitoring and Observability

A comprehensive monitoring and observability stack was implemented to ensure operational efficiency and rapid incident resolution.

- Prometheus and Grafana for real-time metrics and dashboards.
- Opsgenie integration for alerting and incident management.
- BigQuery and ClickHouse for detailed analytics and historical data.

Technical Challenges and Solutions

- Monolithic Architecture: Re-architected applications for GCP compatibility.
- Service Mapping: Carefully mapped AWS services to GCP equivalents.
- IAM Differences: Addressed varying IAM policies between AWS and GCP.
- Network Configuration: Leveraged GCP VPCs, firewalls, and routes for robust connectivity.
- Data Migration: Utilized GCP Data Transfer Service for large-scale data migration, minimizing downtime and cost.
- Application Dependencies: Implemented cloud-agnostic solutions and re-architected AWS-specific applications.
- Operational Changes: Streamlined CI/CD pipelines to adapt to new workflows.
- Security & Compliance: Maintained strict security, data integrity, and compliance throughout.

Organizational Challenges and Solutions

Managing organizational change was critical. This involved establishing governance policies, defining roles, managing stakeholder expectations, and providing extensive training for employees to adapt to GCP services and tools.

© 2025 QuarkWiz, All rights reserved.

Results, Benefits, and Lessons Learned

The migration project concluded successfully in 10 weeks, delivering substantial improvements in performance, cost efficiency, and overall business outcomes.

Performance Improvements

35%

Reduction in overall Josh application platform response time.

99.9999%

Increased availability of the Josh application.

60%

Reduction in total incident counts.

Predictive scaling through the modern architecture resulted in better handling of traffic spikes, alongside enhanced visibility and insights from improved monitoring and alerting systems.

Cost Savings

70%

Overall cost reduction in VM costs due to auto-scaling and modern architecture.

40%

Overall cost reduction in Object Storage through better tiering and compression.

66%

Savings in managed services by migrating from cloud-native to open-source/GCP native solutions.

These savings demonstrate the financial viability and long-term benefits of the migration strategy.

Business Outcomes

- Better performance translated into an improved user experience, leading to reduced user attrition and increased acquisitions.
- Approximately 70% reduction in BAU tasks, significantly enhancing productivity for both developers and operations teams.
- Modern architecture enabled faster release cycles for services (90% faster with Canary and Traffic routing).

Lessons Learned

Rearchitecture, while initially challenging, can yield tremendous mid - to long - term benefits and is a key driver for migration success.

- Always assess current infrastructure and dependencies thoroughly.
- Break down larger problems into smaller, manageable tasks.
- Rigorous testing and retesting are crucial before the final switchover.
- Accurate capacity planning is vital for project success.
- Leveraging historical data greatly assists in planning and execution.

The successful migration to GCP Kubernetes has positioned Josh for future growth, offering a robust foundation for scalable, efficient, and reliable application deployment, with plans to further leverage GCP services for predictive analytics and personalized user experiences.

© 2025 QuarkWiz, All rights reserved.

THANK YOU

www.quarkwiz.com

© 2025 QuarkWiz, All rights reserved.