

AWS FinOps in Practice: How We Cut Multi-Account Cloud Spend by 64%

64% sustained multi-account AWS spend reduction, with zero production downtime.



When enterprise cloud infrastructure scales across distributed teams, cost visibility can become a challenge.

Across a 10-account AWS Organizations and AWS Control Tower environment, monthly infrastructure spend increased by 17% in 30 days. Database services were consuming 65% of the total monthly cloud invoice, driving unsustainable cloud burn.

The bigger issue was that 100% of the spend was untagged in AWS Cost Explorer. Engineering teams did not have clear reasons for cost changes. Finance had no visibility into cost drivers. Unused cloud resources were causing operational bloat and security risks.

If you manage enterprise cloud platforms, then cloud financial management challenges (FinOps) are unavoidable. Best practice would have been to design a proper FinOps strategy directly into the architecture from the beginning. Realistically, tight timelines and stakeholder demands prioritize the completed solution over financial optimizations.

Below is the forensic audit, infrastructure rightsizing, and cost optimization playbook we implemented to deliver a sustained 64% reduction in AWS spend, with zero production downtime. The strategy acts as a starting point for implementing FinOps in an existing cloud environment.

1. Cloud Cost Forensics: Tag-Free Discovery and Resource Inventory

When cost-allocation tags are missing, native AWS billing dashboards are unable to track spend. To establish full financial observability without interrupting daily operations, we engineered an automated, read-only multi-account audit pipeline using the AWS CLI and cross-account IAM/SSO role assumptions to map live resources to billing usage types across active regions.

The forensic analysis isolated three major cost drivers:

- Overprovisioned database tiers: A mission-critical PostgreSQL RDS production instance provisioned with high-cost io1 storage, 3000 Provisioned IOPS, large allocated storage, and legacy engine version extended-support surcharges. This provision accounted for close to a quarter of total organization spend.
- Abandoned legacy workloads: An unmonitored testing environment running an idle 3-node Amazon DocumentDB cluster (db.r6g.large) and supporting EC2 instances in an alternative region accounted for 16% of total spend.
- Redundant network infrastructure: Duplicated VPC Interface Endpoints (AWS PrivateLink) deployed across dev, staging, and production VPCs in linked accounts racked up continuous hourly base rates without proportional need.

2. Surgical Decommissioning with Zero Data Risk

Decommissioning cloud infrastructure is simple. Executing the practice safely in an enterprise environment requires disciplined engineering gates and strict risk mitigation.

To protect business continuity and maintain compliance, we designed a multi-stage decommissioning workflow:

- Pre-change state preservation: Create checksum-verified offsite database exports and tagged EBS root-volume snapshots to establish strict rollback checkpoints.
- Automated dry-run validation: Executed live dry-run termination checks across compute instances, security groups, and Elastic Network Interfaces (ENIs) to detect hidden dependencies.
- Controlled decommissioning: Decommissioned the 3-node DocumentDB cluster, phased out unneeded compute, and released unattached Elastic IPs (EIPs) to eliminate recurring idle charges.

3. Production Rightsizing and Infrastructure Modernization

For active workloads requiring high availability, the objective shifted to cloud architecture optimization. We built an idempotent automation pipeline with pre-flight health checks and staged rollouts.

- Compute and storage rightsizing: Re-architected storage tiers to eliminate unneeded Provisioned IOPS and downscaled primary database compute instances to match actual production workload curves.
- Database modernization: Upgraded core database engines to supported versions, immediately eliminating recurring legacy extended-support penalties.
- Automated health and uptime verification: Validated connectivity end-to-end, testing database reconnect latency, automated stack failovers, and HTTP 200 health checks across all public endpoints.

Measurable Business and FinOps Outcomes

- 64% sustained cloud cost reduction: Slashed overall monthly AWS burn within two billing cycles, maintaining this baseline through subsequent quarters.
- >60% database line-item savings: Rightsized overprovisioned storage tiers, removed idle clusters, and modernized database engines.
- Zero production downtime: 100% service availability maintained throughout all migrations, snapshotting, and resizing events.
- Multi-account FinOps governance: Activated mandatory AWS cost-allocation tags and established resource lifecycle tracking to prevent future cloud bloat.

Strategic Takeaway for Tech Executives and Engineering Leaders

Cloud cost optimization is not a matter of blindly cutting servers. It is an engineering discipline combining cloud architecture, operational risk management, and systems observability.

Fixing runaway costs requires an immediate surgical intervention, but protecting margin over the long run requires implementing AWS Well-Architected practices: automated budget thresholds, consolidated networking topologies, and disciplined lifecycle management.

Is your organization struggling with untagged AWS sprawl, runaway database costs, or overprovisioned infrastructure? Contact Aikware to discuss cloud architecture, FinOps strategy, and automated infrastructure remediation. support@aikware.com