COST ANALYSIS REPORT

CloudSave AI

AWS Infrastructure Cost Optimization

$692

Monthly Savings Identified

Generated: Feb 21, 2026
Input File: test-cdk-stack.ts
File Type: CDK
Services Analyzed: 6
Issues Found: 7

Executive Summary

SERVICES DETECTED	6
ISSUES FOUND	7
CURRENT MONTHLY COST	$998
OPTIMIZED MONTHLY COST	$306
MONTHLY SAVINGS	$692
ANNUAL SAVINGS	$8,304
SAVING PERCENTAGE	69.3%

Detected Services

Service	Detected Resources	Issues	Status
Lambda	Detected	1 issue	Issue
RDS	Detected	**None**	**OK**
ECS	Detected	2 issues	Issue
API Gateway	Detected	1 issue	Issue
S3	Detected	2 issues	Issue
NAT Gateway	Detected	1 issue	Issue

Cost Breakdown by Service

Service	Current/mo	Optimized/mo	Saving/mo	Annual
ECS	$666	**$183**	**$483**	**$5,796**
NAT Gateway	$111	**$37**	**$74**	**$888**
Lambda	$102	**$35**	**$67**	**$804**
API Gateway	$95	**$35**	**$60**	**$720**
S3	$24	**$16**	**$8**	**$96**
TOTAL	**$998**	**$306**	**$692**	**$8,304**

Issues & Recommendations

Over-Scaled ECS Service

HIGH

ECS service "appService" runs 5 tasks. Non-production environments typically only need 1 task for testing. Running 5 tasks multiplies Fargate costs without benefit.

CURRENT
5 tasks, 2048 CPU units, 16384MB
$555/mo

RECOMMENDED
1 task for non-production environments
$111/mo ’Save $444/mo

AI Analysis (Claude)
ECS Fargate charges per-second with a 1-minute minimum for vCPU and memory allocation, meaning 5 tasks at 2 vCPU (2048 CPU units) and 16GB RAM cost approximately $0.08052/hour per task ($0.04048 for vCPU + $0.004445/GB for memory). Running 5 tasks 24/7 in a non-production environment costs $555/month despite typically seeing <20% utilization outside business hours. Non-production workloads rarely require high availability or horizontal scaling, making this a classic over-provisioning scenario where development/staging teams replicate production architecture without adjusting for actual usage patterns. The 80% savings comes from eliminating 4 unnecessary tasks while maintaining functionality for testing purposes, as single-task failure in non-prod is acceptable with quick restart capabilities.

Risk: MODERATE · Apply during your established non-production maintenance window, ideally mid-week (Tuesday-Thursday) between 10 AM - 2 PM local time when developers are available to validate functionality post-change. Avoid Monday mornings (weekend code deployments settling) and Friday afternoons (reduced team availability for issue response). Ensure no active QA test cycles, load testing, or demo preparations are scheduled. Stage the change: development environment first, wait 24-48 hours monitoring stability, then staging. Never apply during sprint demos, customer UAT sessions, or within 3 days of production releases. If using blue/green deployments in non-prod, coordinate with your CI/CD pipeline to prevent automatic scaling during the transition window.

3 NAT Gateways Detected

HIGH

3 NAT Gateways are deployed, costing ~$37/month each. In non-production environments, a single NAT Gateway is sufficient. Each additional NAT Gateway adds ~$32-65/month in base + data transfer costs.

CURRENT
3 NAT Gateways
$111/mo

RECOMMENDED
1 NAT Gateway for non-production (or NAT instance for dev)
$37/mo ’Save $74/mo

AI Analysis (Claude)
NAT Gateways cost $0.045/hour ($32.40/month) plus $0.045/GB data processed, making them expensive for non-production environments. AWS bills per NAT Gateway per hour regardless of utilization, so 3 gateways cost $97.20/month in base charges alone before any data transfer. For dev/staging environments with minimal traffic and relaxed availability SLAs, consolidating to a single NAT Gateway or switching to t3.nano NAT instances ($3.80/month) eliminates redundant hourly charges while maintaining internet egress. The 66.7% savings comes primarily from removing 2 unnecessary NAT Gateway hourly charges, though you sacrifice multi-AZ high availability which is acceptable for non-production workloads. Production environments should retain multi-AZ NAT Gateways for resilience, but dev/staging/QA can safely operate with single-AZ NAT resources.

Risk: MODERATE · Apply during your lowest-traffic development window, typically weekday evenings 6-10 PM local time or weekends. Avoid implementing during active development sprints, load testing, or demo periods. Staging environments should be changed 2-3 days before production deployments to ensure adequate testing time. Dev environments can be modified immediately with team notification. Sequence changes as: sandbox ’ dev ’ QA ’ staging, waiting 24 hours between each tier. Never modify production NAT Gateway topology without formal change control and executive sign-off, as multi-AZ redundancy is critical for production availability SLAs.

Overprovisioned Memory

HIGH

Function "generateReportFunction" allocates 3072MB but typical workloads rarely exceed 1024MB. Over-allocating memory increases cost without performance benefit once CPU/network are no longer the bottleneck.

CURRENT	RECOMMENDED
3072MB memory	**1024MB with ARM64 (Graviton2)**
$102/mo	**$35/mo ’Save $67/mo**

AI Analysis (Claude)

Lambda pricing is per-GB-second with memory allocation directly controlling both memory AND CPU allocation. At 3072MB, you're paying for 3x the compute resources versus 1024MB, but if the function completes successfully at lower memory, you're wasting ~67% of allocated resources. AWS charges $0.0000166667 per GB-second, so 3072MB costs 3x more per execution than 1024MB for the same duration. Additionally, ARM64 (Graviton2) functions receive a 20% price reduction ($0.0000133334 per GB-second) compared to x86_64, compounding savings. The high severity indicates this function runs frequently (likely thousands of invocations monthly) and has been consistently under-utilizing its memory allocation, creating systematic waste across all executions.

Risk: MODERATE · Apply during your lowest-traffic window if this is a customer-facing function, typically weekday 2-5 AM in your primary region's timezone. For backend/async functions (SQS/EventBridge triggered), apply during business hours with active monitoring. Test in dev environment first, then staging for 24-48 hours before production. If using versioning/aliases with weighted routing, apply gradually (10%/50%/100% over 1-2 hours). Avoid applying during: month-end batch processing, marketing campaigns, or alongside other infrastructure changes. Check CloudWatch metrics for the past 30 days to identify true low-traffic periods specific to this function's invocation pattern.

NLB + API Gateway Anti-Pattern

HIGH

API "devApi" uses a Network Load Balancer with API Gateway. NLBs add ~$32/month+ and are unnecessary when API Gateway can directly integrate with ALB, Lambda, or HTTP endpoints via VPC Link with lower cost.

CURRENT	RECOMMENDED
REST API + NLB + VPC Link	**API Gateway HTTP API with direct Lambda or ALB integration**
$95/mo	**$35/mo ’Save $60/mo**

AI Analysis (Claude)

This architecture uses an overprovisioned networking stack where a Network Load Balancer ($22.50/month base + $0.006/LCU-hour) and VPC Link ($0.01/hour = $7.20/month) sit between API Gateway REST API and backend services. REST API charges $3.50/million requests plus data transfer, while the NLB adds unnecessary per-hour charges and LCU costs (billed on new connections, active connections, processed bytes, and new TLS connections). HTTP APIs cost

70% less than REST APIs ($1.00/million vs $3.50/million) and support native Lambda/ALB integrations without VPC Link, eliminating both the NLB infrastructure and the VPC Link hourly charges. The current setup was likely designed for private VPC resources but HTTP APIs now handle this natively through private integrations or direct ALB/Lambda targets, making the NLB an expensive middleman that adds latency without providing material benefit for API workloads.

Risk: NEEDS-REVIEW · Apply during your lowest-traffic maintenance window, typically weekday 2-5 AM UTC for US-based workloads or equivalent regional low-traffic periods. Avoid month-end, quarter-end, or known high-traffic events. For production environments, execute in staged rollout: dev environment first (validate for 48 hours), then staging (validate for 72 hours), finally production. Use weighted routing or canary deployment via Route 53 to gradually shift traffic from REST API to HTTP API over 1-2 week period. If running multi-region, migrate one region at a time with minimum 1-week monitoring intervals between regional deployments.

Fargate Task Over-Allocated

MEDIUM

Service "appService" allocates 2048 CPU units (2 vCPU) and 16384MB. Based on typical application patterns, this is over-provisioned. Right-sizing to 2 vCPU / 4GB reduces cost significantly.

CURRENT	RECOMMENDED
2048 CPU units, 16384MB memory **$111/mo**	**2048 CPU units (2 vCPU), 2048MB — measure then adjust** **$72/mo ’Save $39/mo**

AI Analysis (Claude)

Fargate pricing is based on per-second billing for vCPU and memory independently, with memory being significantly more expensive per GB than compute ($0.004445/GB-hour vs $0.04048/vCPU-hour in us-east-1). Your task is allocated 16GB RAM but only requires 2GB—a 8x over-allocation that wastes $72/month on unused memory alone. Fargate's 1:8 CPU-to-memory ratio flexibility allows 2 vCPU to pair with as little as 4GB, meaning you're paying for 14GB of RAM that never gets utilized. The 35% savings ($39/month) comes almost entirely from rightsizing memory while maintaining the same 2 vCPU allocation, which suggests your workload is compute-bound rather than memory-bound.

Risk: MODERATE · Deploy during your lowest traffic period, typically weekday 2-5 AM in your primary region's timezone, avoiding month-end/quarter-end processing windows. Roll out to development environment first for 48-72 hours of observation, then staging for 24 hours, finally production with canary deployment (10% ’ 50% ’ 100% over 2-hour window). If running batch workloads, time the change between job executions. For services behind ALB/NLB, leverage connection draining during task replacement to ensure zero dropped requests.

No Lifecycle Policy

MEDIUM

Bucket "appDataBucket" has no lifecycle policy. Without lifecycle rules, objects remain in S3 Standard indefinitely. Transitioning older objects to S3 Infrequent Access or Glacier can reduce storage costs by 40-80%.

CURRENT	RECOMMENDED
S3 Standard — no lifecycle transitions **$12/mo**	**Lifecycle: 30d ’S3-IA, 90d ’Glacier** **$8/mo ’Save $4/mo**

AI Analysis (Claude)

S3 Standard storage costs $0.023/GB-month for the first 50TB, while S3 Standard-IA costs $0.0125/GB-month (46% cheaper) and Glacier Flexible Retrieval costs $0.0036/GB-month (84% cheaper). Without lifecycle policies, all objects remain in S3 Standard indefinitely regardless of access patterns, incurring maximum storage costs. The pricing differential compounds over time—a 100GB bucket costs $27.60/year in Standard vs $15/year in S3-IA vs $4.32/year in Glacier. AWS bills storage daily based on average storage used during the month, so lifecycle transitions reduce costs starting the day objects move to cheaper tiers. The 34.8% savings suggests this bucket contains ~520GB with infrequent access patterns that align perfectly with S3-IA economics.

Risk: SAFE · Apply immediately during business hours—lifecycle policies take effect asynchronously with zero service interruption and do not impact object availability during transition. AWS evaluates lifecycle rules once daily at midnight UTC, so policies created today begin transitioning eligible objects tomorrow. No maintenance window required. Safe for all environments (dev/staging/prod) simultaneously since transitions are non-destructive and reversible. Avoid applying during active data migrations or backup windows to prevent S3 API throttling from simultaneous lifecycle evaluation and high-volume PUT operations. For production buckets serving real-time applications, validate access patterns over 7-14 days using S3 Storage Lens or access logs before implementing to ensure 30-day threshold aligns with actual usage.

Optimization Roadmap

Phase 1 — Quick Wins (< 1 hour)	$8/mo
1. No Lifecycle Policy (S3)	+$4/mo
2. No Lifecycle Policy (S3)	+$4/mo

Phase 2 — Medium Effort (1–4 hours)	$550/mo
1. Over-Scaled ECS Service (ECS)	+$444/mo
2. Overprovisioned Memory (Lambda)	+$67/mo
3. Fargate Task Over-Allocated (ECS)	+$39/mo

Phase 3 — Needs Planning (1+ days)	$134/mo
1. 3 NAT Gateways Detected (NAT Gateway)	+$74/mo
2. NLB + API Gateway Anti-Pattern (API Gateway)	+$60/mo

Disclaimer & Notes

Important: Review Before Applying

Estimates are based on static analysis of your infrastructure configuration files. Actual savings may vary based on:

- Actual usage patterns and traffic volumes
- Current AWS pricing in your region (prices change regularly)
- Reserved Instance or Savings Plan discounts you may already have
- Application-specific performance requirements
- Multi-region or compliance requirements

CloudSave AI by Prod Bois

This report was generated automatically by CloudSave AI using static infrastructure analysis. No real AWS API calls were made. All pricing is based on publicly available AWS pricing data.

Generated: Feb 21, 2026 · Version 1.0.0