



AWS COST · SAMPLE DELIVERABLE

Cost Analysis and Savings Roadmap

Week 2 of 6: Find and rank · prepared for Seatlark, Inc.

Prepared by Charles Sieg · charles.sieg@vantalect.com · vantalect.com

May 21, 2026 · AWS Cost Analysis & Optimization: The Optimization (six weeks)

SAMPLE DELIVERABLE

This is a sample deliverable. Seatlark, Inc. is a fictional company: every account, resource, finding, figure, and person in this report is simulated to show the form and depth of a Vantalect engagement. AWS prices are list prices from the AWS Price List API retrieved on October 13 and 16, 2026, and AWS Business Support+ pricing is from the AWS Support pricing page on October 16, 2026.

Contents

Executive summary	2
Method	5
Deliverable 2: the waste register	8
Deliverable 3: the rightsizing plan	12
Storage and databases	18
Network	23
Logging	27
S3 lifecycle	30
Graviton	33
Deliverable 4: the commitment plan	35
Deliverable 6: the savings roadmap	38
The migration trajectory and cost avoidance	43
Appendix A: the registers	46
Appendix B: assumptions	46
Appendix C: price provenance	46

Executive summary

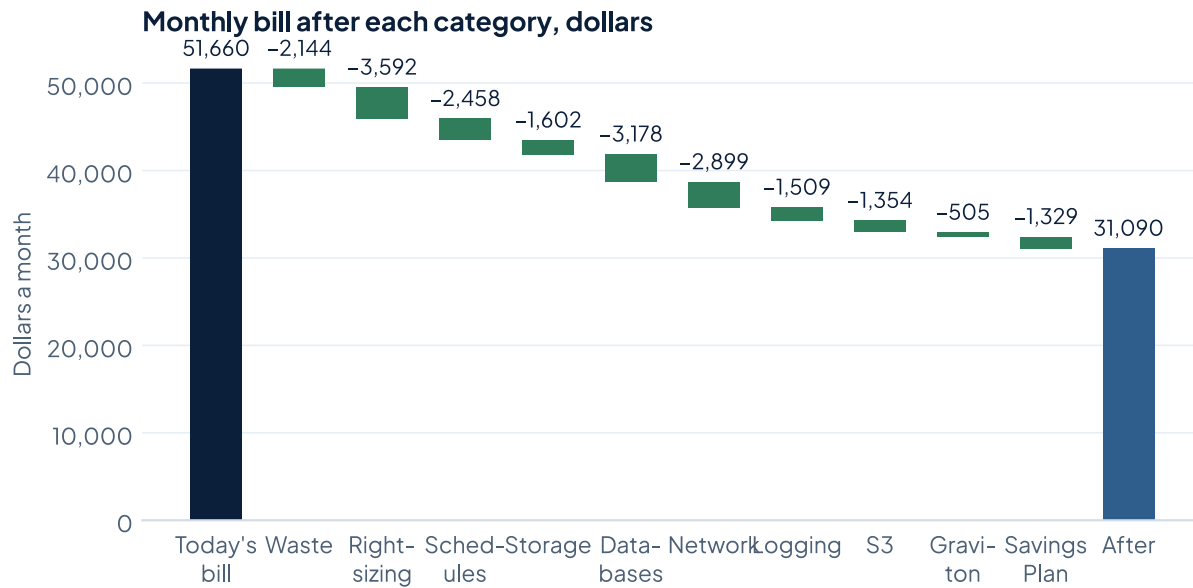
We found **\$246,839 a year** of savings in Seatlark’s AWS bill, 39.8% of the \$619,914 a year it runs at today. They come from 29 findings, priced as 30 changes on 244 resources, and carried out in full they take the monthly bill from **\$51,660 to \$31,090**. These are estimates from your own usage and AWS list prices; they become measured savings only in week 6, change by change, on your Cost and Usage Report.

The number to hold on to is 2.4 times. The Optimization’s second invoice, \$17,500, is issued only if the savings in place at the end of week six, measured on your Cost and Usage Report and annualized, are at least three times the \$35,000 fee: **\$105,000 a year**, 16.9% of your bill. The roadmap identifies 2.4 times that, and the 19 changes rated High confidence alone are 1.4 times it ([The guarantee](#)). The number is credible because of what Seatlark is: 11 months into a 24-month move, with the first waves lifted from the colocation data center at the size of the hardware they ran on and no commitment of any kind.

The headline numbers, all estimates at AWS list prices. Seatlark has no private pricing agreement.

Measure	Amount	What it is
Bill at the May run rate	\$51,660 a month	\$619,914 a year at list prices, including \$3,567 a month of AWS Business Support+
Identified savings	\$246,839 a year	\$20,570 a month across all 30 changes
After the roadmap	\$31,090 a month	Before the fall in Support, which is not counted (Business Support+ follows the bill)
High confidence	\$148,299 a year	19 changes, 60% of the identified savings
Proposed before the freeze	\$174,293 a year	23 changes in weeks 3 and 4, before the production freeze of June 4, 2026
Waiting on a test, a notice, or a build	\$72,545 a year	7 changes that cannot land before June 15, 2026 (The waves)
One-time charges	\$3,117.00	Glacier transition requests and the archive-tier minimum, in W-4, S3-1, and S3-2 (Owners, and what it costs to get there)

Three things shape how to read the number. **Nothing is counted twice:** the savings are an ordered stack, waste first and the commitment last, and every finding is the difference between two prices computed the same way ([Method](#)). **Identified is not in place:** 7 of the changes (\$72,545 a year) wait on a load test, a partner notice period, an application change, or an arm64 build, and nothing here says any change will be approved; each owner decides their own. **A bigger number sits beside this one and is not in it:** the remaining migration waves add \$71,000 a month, and what guardrails could keep out of that growth is not measured, not counted toward the gate, and not a saving on the bill ([The migration trajectory and cost avoidance](#)).



From today's bill to the bill after the roadmap, in the fixed order, dollars a month.

What is in the number

Identified savings by category, in the order the report prices them. E-2 has a staging part and a production part, so 29 findings are 30 changes.

Category	Findings	Changes	Resources	Per month	Per year	One-time cost
Rightsizing	R-1, R-2, R-3, R-4	4	21	\$3,592	\$43,099	\$0.00
Databases	D-1, D-2, D-3	3	4	\$3,178	\$38,135	\$0.00
Network	N-1, N-2, N-3, N-4	4	16	\$2,899	\$34,793	\$0.00
Schedules	S-1, S-2	2	17	\$2,458	\$29,492	\$0.00
Waste	W-1, W-2, W-3, W-4	4	31	\$2,144	\$25,733	\$0.00
Storage	E-1, E-2a, E-2b	3	95	\$1,602	\$19,224	\$0.00
Logging	L-1, L-2, L-3, L-4	4	9	\$1,509	\$18,112	\$0.00
S3 lifecycle	S3-1, S3-2, S3-3	3	3	\$1,354	\$16,245	\$3,117.00
Commitments	C-1	1	30	\$1,329	\$15,950	\$0.00
Graviton	G-1, G-2	2	18	\$505	\$6,055	\$0.00
Total		30	244	\$20,570	\$246,839	\$3,117.00

What happened this week

On May 18, 2026, Aaron Feld walked the on-sale calendar with us, and the production freeze of June 4, 2026 to June 14, 2026 went into the plan: every production change is scheduled for June 2, 2026 to June 3, 2026 or after June 14, 2026, and none in between. On May 19, 2026, Imani Brooks walked the search cluster and the catalog database, the two stateful systems whose changes need the most care ([E-2b: Search cluster io1 to gp3, production](#), [D-3: Catalog database: downsize and Graviton](#)). Today, May 21, 2026, the savings review puts every finding in front of its owner; the sponsor readout with Renee Castellano is tomorrow, May 22, 2026.

Why a capable team missed it. Nobody was careless: each wave was judged on whether the workload came up, not on what it cost. MGN launched every server at the size of its colo allocation, the landing zone stamps its defaults into every VPC and log group, migration leaves volumes, replication servers, and snapshots behind, and every commitment waited "until the migration completes," an instinct that is half right ([Deliverable 4: the commitment plan](#)).

What we need from you

1. **Tell us which findings an owner will not take on as written.** Every change has a named owner, and the owner decides; we rank by dollars, and owners know things we do not.
2. **Confirm the change windows:** production changes on June 2, 2026 and June 3, 2026, or after June 14, 2026.
3. **Look at the commitment plan.** Lydia Brandt owns the one purchase in this report: a one-year No Upfront Compute Savings Plan at \$5.00 an hour, saving \$15,950 a year ([Deliverable 4: the commitment plan](#)).
4. **Sign off the waste register before anything is deleted.** Besides the Savings Plan, the deletions are the only changes that cannot be undone, and each has a snapshot or a named way back ([Deliverable 2: the waste register](#)).

Method

The numbers come from one model of your estate, rebuilt from the resource inventory and twelve months of the Cost and Usage Report and priced hour by hour at AWS list prices. Every finding later in the report shows its own arithmetic.

The fixed order

A resource can appear in more than one finding: an instance is rightsized and then scheduled. Each finding is priced on the estate as the findings before it leave it, so the later one is worth what remains and no dollar is counted twice. The order never changes:

The order every finding is priced in. Waste goes first because it needs no design and shrinks what everything after is sized on; rightsizing precedes schedules, so hours come off the right size; Graviton is priced on the sizes rightsizing leaves; and the commitment goes last, on the 24-hour floor that survives every change before it. The bill starts at \$51,660 a month.

Step	Category	Findings	Bill after the step
1	Waste	W-1, W-2, W-3, W-4	\$49,515
2	Rightsizing	R-1, R-2, R-3, R-4	\$45,923
3	Schedules	S-1, S-2	\$43,466
4	Storage	E-1, E-2a, E-2b	\$41,864
5	Databases	D-1, D-2, D-3	\$38,686
6	Network	N-1, N-2, N-3, N-4	\$35,786
7	Logging	L-1, L-2, L-3, L-4	\$34,277
8	S3 lifecycle	S3-1, S3-2, S3-3	\$32,923
9	Graviton	G-1, G-2	\$32,419
10	Commitments	C-1	\$31,090

A saving is the difference between two prices

Each finding is a change applied to a copy of the estate, priced by the same code that priced the bill, so a saving is always the difference between two prices computed the same way. The simplest case is [R-1: Marketplace API tier to m5.xlarge](#): 6 API hosts moving from an m5.2xlarge at \$0.384 an hour to an m5.xlarge at \$0.192 an hour, for 730 hours: $6 \times (\$0.384 - \$0.192) \times 730 = \$840.96$ a month. Three more rules keep the stack honest. **A later finding is priced on the earlier ones' result:** [S-1: Schedules for development, partner non-production, and perf-test](#) schedules the partner non-production hosts at the size [R-4: Non-production search and partner hosts one size down](#) leaves, so it saves \$0.192 an hour for each hour off, not the larger rate. **Rates are list prices** for a 730-hour month. **Support follows the bill and is not counted** ([Business Support+](#)

[follows the bill](#)), and **the migration's own growth is never netted**: wave 6 lands in the coming weeks and raises the bill, and savings are measured change by change against the resources they touch.

S3 is priced on the whole organization

S3 Standard is priced in volume tiers across the organization, because the 16 accounts share one consolidated bill. Seatlark keeps 87,040 GB in Standard, so the first 51,200 GB cost \$0.023 a GB-month and the 35,840 GB above that cost \$0.022. Gigabytes that leave Standard come off the top of that stack: a gigabyte leaving the top tier saves the top tier's rate, and once the total falls below the first tier's width the rate saved rises to \$0.023. That is why the renditions in S3-3, priced last, save \$0.023 a GB-month where they would have saved \$0.022 priced first ([S3 lifecycle](#)).

Prices, rules, and ratings

Every price is a us-east-1 list price from the AWS Price List API retrieved on October 13, 2026 (Direct Connect, AWS DMS, and AWS Application Migration Service on October 16, 2026), and AWS Business Support+ is from the AWS Support pricing page on October 16, 2026; for this sample all are the dated retrieval, and a live report carries the date of its own analysis ([Appendix C: price provenance](#)). The rules that turn measurements into findings:

- **Waste**: a volume detached 60 or more days with no reads or writes; an MGN server at Cutover complete with no replication for 30 days; a load balancer with no healthy targets or bytes for 90 days; a snapshot with no source volume, AML, or policy.
- **Rightsizing**: in production, one size down when the smaller size's projected peak is under 60% CPU and memory fits, on 93 days of CloudWatch agent data; in non-production, when the 95th percentile is 14% or less. A window that missed an on-sale makes the finding Medium until a load test settles it.
- **IOPS** are provisioned at about twice the measured peak, never below the 3,000 IOPS gp3 includes. **Logs** keep what was queried: no group was queried past 30 days in 90 days of history.

Effort S is a day or less, M a few days with a rolling replacement, L needs an application change.

Risk Low reverses in minutes and touches nothing customers use; Medium touches production or is hard to reverse. **Confidence** High means the measurement covers the case that matters, such as the April on-sale; Medium means it does not, or the saving depends on how AWS bills a detail. Of the 30 changes, 19 are High (\$148,299 a year) and 11 Medium (\$98,540).

Where each change is made

Seatlark's infrastructure is AWS CloudFormation in four repositories (seatlark-landing-zone, marketplace-infra, partner-infra, data-content-infra), with Ansible roles for the lifted hosts. A finding that goes in as a pull request changes a resource one of them owns; a change ticket with a scripted runbook is a one-time operation or a hand-built resource. Nothing changes until the owner signs off.

Deliverable 2: the waste register

\$25,733 a year of Seatlark's bill pays for resources nobody uses: 4 findings covering 31 resources, W-1 through W-4. Almost all of it is what a migration leaves behind. Removing it needs no new design, only an owner's yes and a safe way back. The register is at the end of this section ([The register](#)); the CSV carries the same rows, docs/samples/seatlark/data/cost/waste-register.csv.

How every removal is done. The same five rules apply to every finding here. (1) The owner signs off first, and nothing is deleted on our word alone. (2) Tag the resource `sl:pending-removal=<date>` and wait a business day. (3) Keep a way back at the cost the finding states: a snapshot kept 30 days, an archived snapshot, or a source server that stays in the colo until the lease ends. (4) Prove it in the bill a week later; that is what week 6 counts. (5) If the resource is in code, delete it from the repository in the same pull request, so the next deploy does not recreate it.

The four findings

W-1: Unattached volumes from test cutovers

Owner: Beth Morrow. **Goes in as:** change ticket with a scripted runbook. **Lands:** week 3.

Confidence: High. **Effort:** S. **Risk:** Low. **Saves:** \$500 a month, \$6,000 a year.

What we saw. 10 gp2 volumes, 500 GB each, detached since the February test cutovers, billed \$0.10 a GB-month. Every volume has been detached 60 to 105 days; CloudWatch shows zero read and write operations; the owning test runbook is closed.

The change. Snapshot each volume once (kept 30 days), then delete the volume. **Rollback:** Restore from the 30-day snapshot.

The volumes sit in marketplace-dev and marketplace-staging, so the script runs once per account. A detached volume with no I/O cannot be missed by a running host, and the snapshot keeps the data.

BASH

```
# Per account, as the remediation role. approved-volumes.txt holds one volume ID per line, only
rows the owner signed off.
while read -r VOL; do
    SNAP=$(aws ec2 create-snapshot --volume-id "$VOL" --description "waste-register W-1" --query
SnapshotId --output text \
    --tag-specifications 'ResourceType=snapshot,Tags=[{Key=sl:register-id,Value=W-1},
{Key=sl:delete-after,Value=<yyyy-mm-dd>}]')
    aws ec2 wait snapshot-completed --snapshot-ids "$SNAP" && aws ec2 delete-volume --volume-id
"$VOL"
done < approved-volumes.txt
```

How it was priced. Unattached gp2 volumes: 5,000 GB × \$0.10 a GB-month = \$500.00. The saving is **\$500.00 a month**, \$6,000.00 a year.

W-2: MGN replication servers from completed waves

Owner: Beth Morrow. **Goes in as:** change ticket with a scripted runbook. **Lands:** week 3.

Confidence: High. **Effort:** S. **Risk:** Low. **Saves:** \$661 a month, \$7,927 a year.

What we saw. 14 t3.small replication servers and 14 gp3 staging disks for source servers whose waves finished months ago and were never marked archived. All 14 source servers show Cutover complete in MGN; none has replicated in 30 or more days.

The change. Finalize each cutover in MGN if it is not already, disconnect each source server (MGN then terminates its replication server and staging disk), and mark it archived. **Rollback:** Re-install the replication agent on the source server (the colo server is retained until the lease ends). The Cost and Usage Report shows no AWS Application Migration Service source-server charge for these servers in May: MGN bills a source server by the hour only while it replicates, and none of the 14 has replicated in 30 or more days (their colo sources were powered off after cutover). What still bills is the replication servers and their staging disks, which W-2 removes.

MGN removes the replication resources itself within 90 minutes of a disconnect.

BASH

```
# Source servers that finished cutover and were never archived (the owner signs off the list):
aws mgn describe-source-servers --region us-east-1 --filters isArchived=false \
    --query "items[?lifeCycle.state=='CUTOVER']".
[sourceServerID,sourceProperties.identificationHints.hostname]" --output text
# Each approved server, its cutover already finalized: disconnect (MGN deletes the replication
server and staging disks), then archive.
while read -r SERVER; do
    aws mgn disconnect-from-service --region us-east-1 --source-server-id "$SERVER"
    aws mgn mark-as-archived --region us-east-1 --source-server-id "$SERVER"
done < approved-source-servers.txt
```

What the Cost and Usage Report shows for the MGN fee. After a source server's 2,160 free replication hours, MGN bills \$0.042 for each hour it keeps replicating (AWS Price List API, October 16, 2026). May's Cost and Usage Report has no such charge for these 14 servers, because none has replicated in 30 or more days; it would be at most \$429 a month if their sources were powered back on before W-2 runs. W-2 is priced on the replication servers and staging disks, which are on the bill. The wave runbook should archive on cutover sign-off so the next wave leaves nothing behind (week 5's work, not counted here).

How it was priced. MGN replication servers (t3.small): 14 servers × \$0.0208 an hour × 730 hours = \$212.58; Staging disks (gp3): 5,600 GB × \$0.08 a GB-month = \$448.00. The saving is **\$660.58 a month**, \$7,926.91 a year.

W-3: Idle load balancers from on-sale rehearsals

Owner: Aaron Feld. **Goes in as:** change ticket with a scripted runbook. **Lands:** week 3.

Confidence: High. **Effort:** S. **Risk:** Low. **Saves:** \$33 a month, \$394 a year.

What we saw. Two internet-facing ALBs with no healthy targets and no requests in 90 days. Zero processed bytes and zero healthy targets over 90 days; no Route 53 record points at either.

The change. Delete both load balancers and their empty target groups. **Rollback:** Recreate from the marketplace-infra stack.

Neither load balancer has taken a request in 90 days, and both recreate from the marketplace-infra stack. Delete them (`aws elbv2 delete-load-balancer`, then their empty target groups) and remove both from the template in the same pull request.

How it was priced. Idle Application Load Balancers: 2 load balancers × \$0.0225 an hour × 730 hours = \$32.85. The saving is **\$32.85 a month**, \$394.20 a year.

W-4: Snapshot and AMI sprawl

Owner: Beth Morrow. **Goes in as:** change ticket with a scripted runbook. **Lands:** week 3.

Confidence: Medium. **Effort:** M. **Risk:** Low. **Saves:** \$951 a month, \$11,412 a year.

What we saw. 30 TB of snapshots and AMIs billed, most from six migration waves, with daily lifecycle policies that never expire. EBS snapshots are incremental, so deleting one frees only the blocks no other snapshot references: the estimate uses each snapshot's billed cost from the resource-level Cost and Usage Report lines, not its volume size.

The change. Delete orphaned conversion snapshots and test-cutover AMIs, set a 35-day expiry on the daily policy, archive the pre-cutover final snapshots for a year, delete 1.8 TB of RDS manual snapshots. **Rollback:** Snapshots in the archive tier restore on request (\$0.03 a GB); deleted snapshots are not recoverable, so each deletion has a signed-off line in the waste register.

A deleted snapshot is gone, so each deletion has a signed-off line in the register. Snapshots in the archive tier cost \$0.0125 a GB-month to hold and \$0.03 a GB to restore. **The honest limit:** EBS snapshots are incremental. Deleting one frees only the blocks no other snapshot references, so the estimate uses each set's billed gigabytes from the resource-level lines of the Cost and Usage Report, not the volume's size, and a deleted snapshot's blocks can turn out to be shared with a survivor. That is why this finding is rated Medium. The archive works the other way: an archived snapshot is stored as a full copy, so the pre-cutover set is priced on its full size in the archive tier, while the standard tier releases only its unique blocks.

The snapshots and AMIs, by set: snap-orphaned-mgn-conversion (5,000 GB billed in 210 snapshots, \$250 a month, deleted); snap-orphaned-test-cutover-amis (3,000 GB billed in 96 snapshots, \$150 a month, deleted); snap-lifecycle-no-expiry (5,000 GB billed in 3,100 snapshots, \$250 a month, trimmed to the 35-day policy); snap-golden-pre-cutover (4,000 GB billed in 48 snapshots, \$200 a month, archive tier, kept a year); snap-in-use-wave-amis (7,200 GB billed in 410 snapshots, \$360 a month, stays); snap-in-use-daily (4,000 GB billed in 700 snapshots, \$200 a month, stays). The sets that stay are the ones a running workload or a wave AMI still needs.

The daily policy gets an expiry so the pile cannot grow back (seatlark-landing-zone); the deletions (aws ec2 delete-snapshot, aws rds delete-db-snapshot) are scripted from the signed-off lists, and the archive move is one command:

YAML

```
Resources:
  DailySnapshots:
    Type: AWS::DLM::LifecyclePolicy
    Properties:
      Description: Daily EBS snapshots, kept 35 days
      State: ENABLED
      ExecutionRoleArn: !Sub arn:${AWS::Partition}:iam::${AWS::AccountId}:role/service-role/
AWSDataLifecycleManagerDefaultRole
    PolicyDetails:
      PolicyType: EBS_SNAPSHOT_MANAGEMENT
      ResourceTypes: [VOLUME]
      TargetTags: [{ Key: Backup, Value: daily }]
      Schedules:
        - Name: daily
          CreateRule: { Interval: 24, IntervalUnit: HOURS, Times: ["03:00"] }
          RetainRule: { Interval: 35, IntervalUnit: DAYS }
          CopyTags: true
```

BASH

```
while read -r SNAP; do aws ec2 modify-snapshot-tier --snapshot-id "$SNAP" --storage-tier
archive; done < golden-snapshots.txt # 90-day minimum
```

There is no one-time charge: the archive tier bills a 90-day minimum, and these snapshots stay there a year.

How it was priced. Orphaned conversion snapshots and test-cutover AMIs deleted: 8,000 GB billed × \$0.05 a GB-month = \$400.00; No-expiry daily policy trimmed to 35 days: 5,000 GB billed × \$0.05 a GB-month = \$250.00; Pre-cutover final snapshots leave the standard tier (their unique blocks): 4,000 GB billed × \$0.05 a GB-month = \$200.00; The same snapshots held in the archive tier as full copies: 5,600 GB full size × \$0.0125 a GB-month = –\$70.00; RDS manual snapshots deleted: 1,800 GB × \$0.095 a GB-month = \$171.00. The saving is **\$951.00 a month**, \$11,412.00 a year.

The register

ID	Finding	Owner	Confidence	Per month	Per year
W-1	Unattached volumes from test cutovers (10 resources, marketplace-dev, marketplace-staging)	Beth Morrow	High	\$500	\$6,000
W-2	MGN replication servers from completed waves (14 resources, sl-migration)	Beth Morrow	High	\$661	\$7,927
W-3	Idle load balancers from on-sale rehearsals (2 resources, marketplace-staging)	Aaron Feld	High	\$33	\$394
W-4	Snapshot and AMI sprawl (5 resources, sl-backup, sl-migration)	Beth Morrow	Medium	\$951	\$11,412
Total				\$2,144	\$25,733

Deliverable 3: the rightsizing plan

Compute is \$43,099 a year of the identified savings in four findings, R-1 through R-4, plus \$29,492 a year in two schedule findings, S-1 and S-2. All of it follows from one fact: MGN launched every server at the size of its colo allocation, and nobody has resized since. The plan sizes from measured use, with two stops that the evidence draws and that we state plainly: **the web tier keeps its size**, and **the ingestion workers wait for a load test**.

What the measurements say

The CloudWatch agent was installed in each wave, so 93 days of CPU and memory exist for every lifted host. Compute Optimizer agrees with them on every group but one: its default 14-day window missed the April on-sale, so for the ingestion workers it saw a quiet two weeks.

Halving an instance doubles its CPU use; memory is the share of the smaller size's RAM that the 95th-percentile use fills. Production search and the CI runners are shown for contrast and are not resized.

CPU and memory by group, and what each would run at on the next size down.

Group	Hosts	Size now	CPU p95	CPU peak	Memory p95	Smaller size: peak	Smaller size: memory
Marketplace web	6	m5.2xlarge	41%	41%	52%	82%	104% (m5.xlarge)
Marketplace API (R-1)	6	m5.2xlarge	9%	29%	24%	58%	48% (m5.xlarge)
Partner APIs (R-2)	6	m5.2xlarge	9%	21%	22%	42%	44% (m5.xlarge)
Listing ingestion (R-3)	4	m5.4xlarge	18%	33%	30%	66%	60% (m5.2xlarge)
Staging search (R-4)	2	r5.2xlarge	14%	40%	40%	80%	80% (r5.xlarge)
Partner non-production (R-4)	3	m5.2xlarge	7%	16%	20%	32%	40% (m5.xlarge)
Production search	3	r5.2xlarge	38%	55%	61%		
CI runners	3	m5.4xlarge	78%	92%	55%		

By group: the **API tier** (R-1) runs at 9% CPU at the 95th percentile and 29% at the April on-sale peak, and on an m5.xlarge that peak is 58%, under the 60% line, with memory filling 48% of the smaller size. The **partner APIs** (R-2) peak at 21% and would run at 42%. **The web tier keeps its size:** its on-sale peak is 41% on the current size and would be 82% on the next size down, and memory that would fill 104% of the smaller size, with no autoscaling yet to absorb the difference, so it stays on m5.2xlarge and is counted nowhere in this report. The **ingestion workers** (R-3) project to a 66% peak, above the line, and Compute Optimizer never saw an on-sale: two reasons for a load test, and why the finding is Medium. **Non-production** (R-4) is held to its 95th percentile, because the on-sale rehearsals there replay a fraction of production load and never approach the peak (the reason S-2 keeps staging running overnight is when they run, not how hard): staging search runs at 14% and the partner hosts at 7%, and both double without trouble. Production search peaks at 55% and the CI runners at 92% on their current size, so neither is a finding.

The four findings

Each production resize is a rolling replacement, one host at a time out of the load balancer, with the previous size one command away. The pull request changes the instance type in the marketplace-infra or partner-infra launch template, so the next deploy keeps it; this script does the day-of resize for one target group and is the same for R-1, R-2, and the load-balanced R-4 hosts:

BASH

```
# For a rollback, run it again with NEW set to the old type.
TG=<target-group-arn>; NEW=m5.xlarge
for ID in $(cat approved-hosts.txt); do
    aws elbv2 deregister-targets --target-group-arn "$TG" --targets Id="$ID"
    aws elbv2 wait target-deregistered --target-group-arn "$TG" --targets Id="$ID"
    aws ec2 stop-instances --instance-ids "$ID" > /dev/null && aws ec2 wait instance-stopped --
instance-ids "$ID"
    aws ec2 modify-instance-attribute --instance-id "$ID" --instance-type "{\"Value\": \"$NEW\"}"
    aws ec2 start-instances --instance-ids "$ID" > /dev/null && aws ec2 wait instance-status-ok
--instance-ids "$ID"
    aws elbv2 register-targets --target-group-arn "$TG" --targets Id="$ID"
    aws elbv2 wait target-in-service --target-group-arn "$TG" --targets Id="$ID"
done
```

R-1: Marketplace API tier to m5.xlarge

Owner: Aaron Feld. **Goes in as:** pull request in marketplace-infra. **Lands:** week 4. **Confidence:** High. **Effort:** M. **Risk:** Medium. **Saves:** \$841 a month, \$10,092 a year.

What we saw. 6 m5.2xlarge API hosts at 9% CPU at the 95th percentile and 29% at the April on-sale peak. The evidence is 93 days of CloudWatch agent memory and CPU metrics (the agent was installed in each wave), which include the April on-sale, and Compute Optimizer agrees; at the smaller size the on-sale peak would be near 58%.

The change. Replace the 6 API hosts with m5.xlarge, one at a time behind the load balancer, before the June freeze. **Rollback:** Instant: relaunch the previous launch template version.

Five of the six hosts stay in service throughout. Proposed for June 2, 2026 to June 3, 2026, before the freeze, so the on-sales that follow are the live test on the smaller size.

How it was priced. API hosts: m5.2xlarge to m5.xlarge: 6 hosts × (\$0.384 – \$0.192) an hour × 730 hours = \$840.96. The saving is **\$840.96 a month**, \$10,091.52 a year.

R-2: Partner APIs to m5.xlarge

Owner: Diego Saldana. **Goes in as:** pull request in partner-infra. **Lands:** week 4. **Confidence:** High. **Effort:** M. **Risk:** Medium. **Saves:** \$841 a month, \$10,092 a year.

What we saw. 6 m5.2xlarge partner API hosts at 9% CPU at the 95th percentile. 93 days of agent metrics show 21% at the busiest partner feed run.

The change. Replace with m5.xlarge, rolling, before the June freeze. **Rollback:** Instant: relaunch the previous launch template version.

Same script, between partner feed runs.

How it was priced. Partner API hosts: m5.2xlarge to m5.xlarge: 6 hosts × (\$0.384 – \$0.192) an hour × 730 hours = \$840.96. The saving is **\$840.96 a month**, \$10,091.52 a year.

R-3: Listing ingestion workers to m5.2xlarge

Owner: Imani Brooks. **Goes in as:** pull request in marketplace-infra. **Lands:** after the freeze.

Confidence: Medium. **Effort:** M. **Risk:** Medium. **Saves:** \$1,121 a month, \$13,455 a year.

What we saw. 4 m5.4xlarge ingestion workers at 18% CPU at the 95th percentile. The peak is 33%, but Compute Optimizer's default 14-day window missed the April on-sale entirely, so the recommendation is rated medium.

The change. Resize to m5.2xlarge after a load test that replays an on-sale. It waits on the load test, run as a perf-test exercise after the freeze, because Compute Optimizer never saw an on-sale. **Rollback:** Instant: relaunch the previous launch template version.

The one rightsizing finding we ask to be tested first. The load test replays an on-sale against m5.2xlarge workers in the perf-test account, and the resize goes ahead only if the replay keeps the peak under the line.

How it was priced. Ingestion workers: m5.4xlarge to m5.2xlarge: 4 hosts × (\$0.768 – \$0.384) an hour × 730 hours = \$1,121.28. The saving is **\$1,121.28 a month**, \$13,455.36 a year.

R-4: Non-production search and partner hosts one size down

Owner: Beth Morrow. **Goes in as:** pull request in marketplace-infra. **Lands:** week 3. **Confidence:** High. **Effort:** S. **Risk:** Low. **Saves:** \$788 a month, \$9,461 a year.

What we saw. 2 staging r5.2xlarge search nodes and 3 partner non-production m5.2xlarge hosts at under 15% CPU. Non-production peaks at 14% CPU at the 95th percentile; staging search holds a tenth of production's index.

The change. Resize the staging search nodes to r5.xlarge and the partner non-production hosts to m5.xlarge. **Rollback:** Instant: relaunch the previous launch template version.

Non-production, so the same script without the load-balancer steps where a host is not behind one.

How it was priced. Staging search nodes: r5.2xlarge to r5.xlarge: 2 hosts × (\$0.504 – \$0.252) an hour × 730 hours = \$367.92; Partner non-production hosts: m5.2xlarge to m5.xlarge: 3 hosts × (\$0.384 – \$0.192) an hour × 730 hours = \$420.48. The saving is **\$788.40 a month**, \$9,460.80 a year.

Schedules

Five development hosts, three partner non-production hosts, and the perf-test fleet run every hour of the month. Nobody uses them outside working hours, and the perf-test fleet is needed only for on-sale rehearsals. [Instance Scheduler on AWS](#) starts and stops tagged instances, and its schedules are code, a stack in marketplace-infra plus a tag on each host:

YAML

```
Parameters:
  ServiceInstanceScheduleServiceTokenARN: { Type: String }  # the hub stack's
ServiceInstanceScheduleServiceToken output
Resources:
  WeekdayTwelveHours:
    Type: Custom::ServiceInstanceSchedule
    Properties:
      ServiceToken: !Ref ServiceInstanceScheduleServiceTokenARN
      NoStackPrefix: 'True'
      Name: weekday-12h
      Timezone: America/New_York
      Periods:
        - Description: Weekdays, 07:00 to 18:59
          BeginTime: '07:00'
          EndTime: '18:59'
          WeekDays: mon-fri
```

BASH

```
aws ec2 create-tags --resources $(cat approved-hosts.txt) --tags
Key=Schedule,Value=weekday-12h  # remove the tag to roll back
```

S-1: Schedules for development, partner non-production, and perf-test

Owner: Beth Morrow. **Goes in as:** pull request in marketplace-infra. **Lands:** week 3. **Confidence:** High. **Effort:** S. **Risk:** Low. **Saves:** \$2,015 a month, \$24,183 a year.

What we saw. 5 development hosts, 3 partner non-production hosts, and 3 perf-test hosts running every hour of the month. Development and partner non-production traffic is zero outside working hours; the perf-test fleet is used for on-sale rehearsals a few times a month.

The change. Instance Scheduler: 12 hours a day on weekdays (260 of 730 hours); the perf-test fleet at zero and scaled out for rehearsals (96 hours a month in the plan). **Rollback:** Remove the schedule tag; the hosts start within minutes.

The perf-test fleet is not on a clock: it sits stopped (`aws ec2 stop-instances`), and the rehearsal checklist starts and stops it. The plan counts 96 hours a month for it.

How it was priced. Development hosts: m5.xlarge, off 470 of 730 hours: 5 hosts × \$0.192 an hour × 470 hours off = \$451.20; Partner non-production hosts: m5.xlarge, off 470 of 730 hours: 3 hosts × \$0.192 an hour × 470 hours off = \$270.72; Perf-test fleet: c5.4xlarge, off 634 of 730 hours: 3 hosts × \$0.68 an hour × 634 hours off = \$1,293.36. The saving is **\$2,015.28 a month**, \$24,183.36 a year.

S-2: Schedule for staging

Owner: Aaron Feld. **Goes in as:** pull request in marketplace-infra. **Lands:** week 3. **Confidence:** Medium. **Effort:** S. **Risk:** Low. **Saves:** \$442 a month, \$5,308 a year.

What we saw. 6 staging hosts running every hour of the month. Staging traffic is zero overnight except the on-sale rehearsals and partner acceptance tests in other time zones that the schedule would interrupt.

The change. Instance Scheduler: 6 AM to 10 PM Eastern on weekdays (about 347 of 730 hours).

Rollback: Remove the schedule tag.

The one schedule with a real cost to a team: staging also runs the overnight on-sale rehearsals and the partner acceptance tests that start in other time zones, which a 6 AM to 10 PM schedule would interrupt. That is why it is Medium confidence, and the owner may reasonably keep staging on a wider window. It is the same resource as above with BeginTime: '06:00' and EndTime: '21:59' under the name staging-6am-10pm.

How it was priced. Staging hosts: c5.xlarge, off 383.3 of 730 hours: 1 host × \$0.17 an hour × 383.3333 hours off = \$65.17; Staging hosts: m5.large, off 383.3 of 730 hours: 1 host × \$0.096 an hour × 383.3333 hours off = \$36.80; Staging hosts: m5.xlarge, off 383.3 of 730 hours: 2 hosts × \$0.192 an hour × 383.3333 hours off = \$147.20; Staging hosts: r5.xlarge, off 383.3 of 730 hours: 2 hosts × \$0.252 an hour × 383.3333 hours off = \$193.20. The saving is **\$442.37 a month**, \$5,308.40 a year.

The rightsizing plan

The rightsizing plan: the four rightsizing findings. The schedule findings, S-1 and S-2, are priced on the sizes this plan leaves.

ID	Accounts	Change	Owner	Confidence	Effort	Risk	Per year
R-1	marketplace-prod	Marketplace API tier to m5.xlarge (6 hosts)	Aaron Feld	High	M	Medium	\$10,092
R-2	partner-prod	Partner APIs to m5.xlarge (6 hosts)	Diego Saldana	High	M	Medium	\$10,092
R-3	marketplace-prod	Listing ingestion workers to m5.2xlarge (4 hosts)	Imani Brooks	Medium	M	Medium	\$13,455
R-4	marketplace-staging, partner-nonprod	Non-production search and partner hosts one size down (5 hosts)	Beth Morrow	High	S	Low	\$9,461
Total							\$43,099

Storage and databases

Storage and databases are \$57,359 a year in five findings, priced as six changes: E-1 and E-2 (as E-2a and E-2b) for volumes (\$19,224), D-1 through D-3 for databases (\$38,135).

Volumes

The estate holds 10 gp2 unattached volumes (5,000 GB, \$500 a month); 90 gp2 attached volumes (17,600 GB, \$1,760 a month); 14 gp3 attached volumes (5,600 GB, \$448 a month); 5 io1 attached volumes (5,000 GB, \$1,665 a month). Every gp2 volume was created by MGN; the unattached ones are W-1 and the io1 volumes are the search cluster.

gp3 lists at \$0.08 a GB-month against gp2's \$0.10, includes 3,000 IOPS and 125 MiB/s at any size, and changes online with Elastic Volumes: no restart, no detach. The search cluster's io1 volumes were provisioned to match the colo SAN's tier; their measured peak is 1,900 IOPS in production, including the reindex after the April on-sale, and 700 in staging.

E-1: gp2 to gp3 on attached volumes

Owner: Beth Morrow. **Goes in as:** change ticket with a scripted runbook. **Lands:** week 3.

Confidence: High. **Effort:** S. **Risk:** Low. **Saves:** \$352 a month, \$4,224 a year.

What we saw. 17,600 GB of attached gp2 volumes created by MGN, billed \$0.10 a GB-month. gp3 lists at \$0.08 a GB-month with a higher baseline than gp2 for volumes under 1 TB; no volume's measured peak is above 3,000 IOPS.

The change. Modify each volume to gp3 online with Elastic Volumes (3,000 IOPS and 125 MiB/s included); no instance restart. **Rollback:** Modify the volume back to gp2 (once in 6 hours). RDS storage is not a finding: RDS gp3 lists at the same \$0.115 a GB-month as gp2 in us-east-1.

One modification per volume is allowed every six hours, so a rollback to gp2 is possible but not instant. No volume's measured peak is above 3,000 IOPS, and gp3's baseline matches or beats gp2's under 1 TB.

BASH

```
while read -r VOL; do aws ec2 modify-volume --volume-id "$VOL" --volume-type gp3; done <
approved-gp2-volumes.txt
aws ec2 describe-volumes-modifications --filters Name=modification-
state,Values=modifying,optimizing \
--query 'VolumesModifications[][VolumeId,ModificationState,Progress]' --output text
```

How it was priced. Attached gp2 volumes to gp3: 17,600 GB × (\$0.10 – \$0.08) a GB-month = \$352.00. The saving is **\$352.00 a month**, \$4,224.00 a year.

E-2a: Search cluster io1 to gp3, staging

Owner: Imani Brooks. **Goes in as:** pull request in marketplace-infra. **Lands:** week 3. **Confidence:** High. **Effort:** S. **Risk:** Low. **Saves:** \$350 a month, \$4,200 a year.

What we saw. 2 io1 volumes, 1,000 GB at 2,000 provisioned IOPS each, provisioned to match the colo SAN's tier. Measured peak is 700 IOPS.

The change. Modify to gp3 at the included 3,000 IOPS. **Rollback:** Modify back to io1.

Staging moves to the 3,000 IOPS that gp3 includes, more than four times the measured peak.

How it was priced. Capacity: io1 to gp3: 2,000 GB × (\$0.125 – \$0.080) a GB-month = \$90.00; Provisioned IOPS dropped (gp3 includes 3,000): 4,000 IOPS × \$0.065 an IOPS-month = \$260.00. The saving is **\$350.00 a month**, \$4,200.00 a year.

E-2b: Search cluster io1 to gp3, production

Owner: Imani Brooks. **Goes in as:** pull request in marketplace-infra. **Lands:** week 4. **Confidence:** High. **Effort:** S. **Risk:** Low. **Saves:** \$900 a month, \$10,800 a year.

What we saw. 3 io1 volumes, 1,000 GB at 4,000 provisioned IOPS each (\$385 a volume-month). Measured peak is 1,900 IOPS, including the reindex after the April on-sale.

The change. Modify to gp3 at 4,000 IOPS (\$85 a volume-month), one volume at a time. **Rollback:** Modify back to io1.

Production moves to gp3 at 4,000 IOPS, about twice the measured peak (\$0.005 an IOPS-month above the included 3,000), one volume at a time with the cluster green between volumes. The matching Iops and Throughput properties in marketplace-infra keep the next deploy from reverting it.

BASH

```
aws ec2 modify-volume --volume-id <volume-id> --volume-type gp3 --iops 4000 --throughput 125
```

How it was priced. Capacity: io1 to gp3: 3,000 GB × (\$0.125 – \$0.080) a GB-month = \$135.00; Provisioned io1 IOPS dropped: 12,000 IOPS × \$0.065 an IOPS-month = \$780.00; gp3 IOPS above the included 3,000, kept at 4,000: 3,000 IOPS × \$0.005 an IOPS-month = –\$15.00. The saving is **\$900.00 a month**, \$10,800.00 a year.

Databases

Every database came over as Multi-AZ, at the size of its colo host. Non-production Multi-AZ doubles the instance and storage price for an availability nobody needs; the production databases are oversized. D-2 and D-3 halve their memory, which only the memory and buffer-cache evidence below can justify, and move them to Graviton classes; each card shows the two parts of the saving separately.

The five RDS instances, instance and storage together. The staging catalog database is not a finding: the team rehearses failover there before each cutover, so it keeps Multi-AZ.

Database	Engine	Class	Multi-AZ	CPU p95	Per month	Finding
catalog-prod (marketplace-prod)	postgres	db.r5.4xlarge	Yes	18%	\$3,725	D-3
partner-prod (partner-prod)	mysql	db.r5.2xlarge	Yes	9%	\$1,770	D-2
catalog-staging (marketplace-staging)	postgres	db.r5.xlarge	Yes	6%	\$1,006	None: keeps Multi-AZ
catalog-dev (marketplace-dev)	postgres	db.r5.xlarge	Yes	4%	\$1,075	D-1
partner-nonprod (partner-nonprod)	mysql	db.r5.large	Yes	5%	\$517	D-1

RDS storage is not a finding

RDS gp3 storage lists at \$0.115 a GB-month in us-east-1, the same as gp2's \$0.115, so switching the storage type saves nothing at these sizes.

D-1: Multi-AZ off on two non-production databases

Owner: Beth Morrow. **Goes in as:** pull request in marketplace-infra. **Lands:** week 3. **Confidence:** High. **Effort:** S. **Risk:** Low. **Saves:** \$796 a month, \$9,552 a year.

What we saw. 2 non-production databases running Multi-AZ at the size of their colo hosts. Nobody depends on non-production failover; Multi-AZ doubles the instance and storage price.

The change. Convert to Single-AZ (the staging catalog database keeps Multi-AZ because the team rehearses failover there before each cutover). **Rollback:** Convert back to Multi-AZ.

Nobody depends on non-production failover; converting back is the same command with `--multi-az`.

How it was priced. catalog-dev: instance, Multi-AZ to Single-AZ (db.r5.xlarge): 1 instance × (\$1.00 – \$0.50) an hour × 730 hours = \$365.00; catalog-dev: the standby's storage copy: 1,500 GB × \$0.115 a GB-month = \$172.50; partner-nonprod: instance, Multi-AZ to Single-AZ (db.r5.large): 1 instance × (\$0.48 – \$0.24) an hour × 730 hours = \$175.20; partner-nonprod: the standby's storage copy: 724 GB × \$0.115 a GB-month = \$83.26. The saving is **\$795.96 a month**, \$9,551.52 a year.

D-2: Partner database: downsize and Graviton

Owner: Diego Saldana. **Goes in as:** pull request in partner-infra. **Lands:** after the freeze. **Confidence:** Medium. **Effort:** M. **Risk:** Medium. **Saves:** \$775 a month, \$9,294 a year.

What we saw. db.r5.2xlarge MySQL Multi-AZ (64 GiB) at 9% CPU at the 95th percentile; over 30 days FreeableMemory never fell below 41 GiB, and the InnoDB buffer pool's working set is 18 GiB with a 99.8% hit ratio. This halves the memory, so it is a downsize first and a Graviton move second, and the saving is shown in those two parts; the 18 GiB working set fits in 32 GiB with room for connections. The contracts require ten business days' notice of a maintenance window.

The change. Restore a snapshot to db.r6g.xlarge and replay a week of partner traffic against it; if the buffer-pool hit ratio stays above 99% and p95 latency holds, modify to db.r6g.xlarge Multi-AZ in the announced window. It waits on a load test against a restored copy at the new class, then ten business days' notice of a maintenance window, which Seatlark Connect's partner contracts require. **Rollback:** Modify back to the previous class.

The partner database serves Seatlark Connect, whose contracts require ten business days' notice of a maintenance window, so the change cannot be scheduled before that notice has run. The load test against a restored copy is a gate, as it is for D-3: the notice goes out only after it passes. A class change on a Multi-AZ instance includes a failover.

How it was priced. partner-prod: downsize, db.r5.2xlarge to db.r5.xlarge, Multi-AZ: 1 instance × (\$1.92 – \$0.96) an hour × 730 hours = \$700.80; partner-prod: Graviton, db.r5.xlarge to db.r6g.xlarge, Multi-AZ: 1 instance × (\$0.960 – \$0.859) an hour × 730 hours = \$73.73. The saving is **\$774.53 a month**, \$9,294.36 a year.

D-3: Catalog database: downsize and Graviton

Owner: Imani Brooks. **Goes in as:** pull request in marketplace-infra. **Lands:** after the freeze.

Confidence: Medium. **Effort:** M. **Risk:** Medium. **Saves:** \$1,607 a month, \$19,290 a year.

What we saw. db.r5.4xlarge PostgreSQL Multi-AZ (128 GiB) at 18% CPU at the 95th percentile; over 30 days FreeableMemory never fell below 70 GiB, and the shared-buffer hit ratio is 99.6% on a hot set of 38 GiB. This halves the memory, so it is a downsize first and a Graviton move second, and the saving is shown in those two parts. db.r5.4xlarge PostgreSQL Multi-AZ lists at \$4.00 an hour; db.r6g.2xlarge at \$1.798.

The change. Modify to db.r6g.2xlarge Multi-AZ after a load test that replays an on-sale against a restored copy at the new class. It waits on a load test that replays an on-sale, then a maintenance window. **Rollback:** Modify back to the previous class.

The catalog database is the exchange's core. A load test replays an on-sale against a copy at the smaller class first, and the resize follows in a maintenance window.

How it was priced. catalog-prod: downsize, db.r5.4xlarge to db.r5.2xlarge, Multi-AZ: 1 instance × (\$4.00 – \$2.00) an hour × 730 hours = \$1,460.00; catalog-prod: Graviton, db.r5.2xlarge to db.r6g.2xlarge, Multi-AZ: 1 instance × (\$2.000 – \$1.798) an hour × 730 hours = \$147.46. The saving is **\$1,607.46 a month**, \$19,289.52 a year.

MySQL 8.0 leaves standard support on July 31, 2026

Both MySQL databases run 8.0. Amazon RDS ends standard support for MySQL 8.0 on July 31, 2026, and from August 1, 2026 RDS Extended Support bills \$0.10 a vCPU-hour in years one and two (\$0.20 in year three) on every 8.0 instance left, both copies of a Multi-AZ instance. Priced from the same AWS Price List table as every other finding, that is `partner-prod` \$584 a month (4 vCPUs, two copies, at db.r6g.xlarge; \$1,168 if D-2 slips and it is still db.r5.2xlarge), `partner-nonprod` \$146 a month (2 vCPUs, at db.r5.large) and : \$730 a month, \$8,760 a year. Upgrading both to MySQL 8.4 with an RDS blue/green deployment in D-2's June 27, 2026 window avoids it; Diego Saldana owns it. This is **cost avoidance: not on the bill today, not measured, and not counted toward the gate**, so it is not in the roadmap's total and adds nothing to the gate.

BASH

```
aws rds modify-db-instance --db-instance-identifier catalog-dev --no-multi-az --apply-immediately # D-1 (and partner-nonprod)
aws rds modify-db-instance --db-instance-identifier partner-prod --db-instance-class db.r6g.xlarge --apply-immediately # D-2, after the load test, in the announced window
aws rds modify-db-instance --db-instance-identifier catalog-prod --db-instance-class db.r6g.2xlarge --apply-immediately # D-3, after the load test
```

Network

Network is \$34,793 a year in four findings, N-1 through N-4. Three of them come from the landing zone's defaults and the shape of a hybrid estate: every VPC with its own internet egress (six of them) got three NAT gateways and no S3 gateway endpoint, and every page view on the exchange still calls the orders database in the colo across Direct Connect.

What the NAT gateways carry

The 18 NAT gateways at \$0.045 an hour and \$0.045 a GB processed. The CUR query below finds the gateways and what they cost; the bytes bound for S3 come from matching the flow logs' destinations to the S3 ranges AWS publishes in `ip-ranges.json`.

Account	Gateways	GB processed	Bound for S3	Share to S3	Hours per month	Data per month	Of which S3
marketplace-prod	3	10,963	6,644	61%	\$99	\$493	\$299
partner-prod	3	5,903	2,602	44%	\$99	\$266	\$117
perf-test	3	4,357	1,440	33%	\$99	\$196	\$65
marketplace-staging	3	2,952	1,138	39%	\$99	\$133	\$51
marketplace-dev	3	2,670	883	33%	\$99	\$120	\$40
partner-nonprod	3	1,827	604	33%	\$99	\$82	\$27
Total	18	28,672	13,312	46%	\$591	\$1,290	\$599

SQL

```
-- CUR 2.0 export in Athena: NAT gateway hours and bytes by account, for the baseline month.
SELECT line_item_usage_account_id AS account, line_item_usage_type AS usage_type,
       ROUND(SUM(line_item_usage_amount), 1) AS quantity, ROUND(SUM(line_item_unblended_cost),
2) AS cost
FROM cur2.seatlark_cur
WHERE billing_period = '2026-04' AND line_item_usage_type LIKE '%NatGateway%'
GROUP BY 1, 2 ORDER BY cost DESC;
```

N-1: S3 gateway endpoints in every VPC with a NAT gateway

Owner: Beth Morrow. **Goes in as:** pull request in seatlark-landing-zone. **Lands:** week 3.

Confidence: High. **Effort:** S. **Risk:** Low. **Saves:** \$599 a month, \$7,188 a year.

What we saw. None of the six VPCs with NAT gateways has an S3 gateway endpoint, so every byte to S3 leaves through a NAT gateway at \$0.045 a GB. 13 TB of the 28 TB the NAT gateways process a month is bound for S3.

The change. Add an S3 gateway endpoint (free) to each VPC's route tables. **Rollback:** Remove the endpoint; traffic returns to the NAT gateways.

A gateway endpoint adds a route for S3's address ranges to the route tables you name, and same-Region traffic to S3 stops using the NAT gateway. It is free, and it does not work across a Transit Gateway or from outside the VPC, which none of this traffic does. The change is a StackSet in seatlark-landing-zone on the private route tables of the 6 VPCs that run workloads:

YAML

```
S3GatewayEndpoint:
  Type: AWS::EC2::VPCEndpoint
  Properties:
    VpcEndpointType: Gateway
    ServiceName: !Sub com.amazonaws.${AWS::Region}.s3
    VpcId: !Ref Vpc
    RouteTableIds: [!Ref PrivateRouteTableA, !Ref PrivateRouteTableB, !Ref PrivateRouteTableC]
```

How it was priced. Bytes to S3 that leave the NAT gateways: 13,312 GB × \$0.045 a GB = \$599.04. The saving is **\$599.04 a month**, \$7,188.48 a year.

N-2: One NAT gateway per non-production VPC

Owner: Beth Morrow. **Goes in as:** pull request in seatlark-landing-zone. **Lands:** week 3.

Confidence: High. **Effort:** S. **Risk:** Low. **Saves:** \$263 a month, \$3,154 a year.

What we saw. The landing zone stamps three NAT gateways into every VPC with internet egress, production or not: 12 in four non-production VPCs. Non-production needs no availability across three zones.

The change. One NAT gateway per non-production VPC; production keeps three. **Rollback:** Re-add NAT gateways from the landing-zone template.

Production keeps three gateways. Where a host sits in a different zone from the gateway that stays, its traffic pays cross-AZ transfer to reach it, which the estimate does not net, so the pull request review checks each non-production VPC's zones first. The landing-zone template takes a gateway count (NatGatewayCount, 1 for non-production) behind a condition, and each private default route points at the surviving gateway with !If [ThreeNatGateways, !Ref NatGatewayB, !Ref NatGatewayA].

How it was priced. NAT gateways removed from non-production VPCs: 8 gateways × \$0.045 an hour × 730 hours = \$262.80. The saving is **\$262.80 a month**, \$3,153.60 a year.

The hybrid path to the colo

The orders and checkout services and their SQL Server database are still in the colo, and every page view calls them for order status and held inventory across Direct Connect. That chatter pays three ways:

What the traffic between AWS and the colo costs today. The Direct Connect ports are priced separately, in N-4.

Path	GB per month	Rate per GB	Charged	Per month
Transit Gateway data processed	40,000	\$0.020	once	\$800
Direct Connect data out toward the colo	28,000	\$0.020	once	\$560
Cross-AZ transfer, charged each way	34,350	\$0.010	each way	\$687

N-3: Cache the colo order-status calls

Owner: Aaron Feld. **Goes in as:** pull request in marketplace-infra. **Lands:** after the freeze.

Confidence: Medium. **Effort:** L. **Risk:** Medium. **Saves:** \$614 a month, \$7,369 a year.

What we saw. Every page view on the exchange still calls the orders database in colo across Direct Connect for order status and held inventory. Hybrid chatter pays \$0.02 a GB through Transit Gateway, \$0.02 a GB out over Direct Connect, and \$0.01 a GB each way across AZs; the estimate is a 30% cut.

The change. A short-TTL cache for order status and held inventory, plus in-AZ reads. It waits on an application change: a short-TTL cache for order status and held inventory. **Rollback:** Turn the cache off.

The largest effort in the report (L). This is an application change, not a configuration one: the exchange caches order status and held inventory for a short time to live and reads in the same zone where it can. Held inventory needs care, since a stale answer can sell a seat twice, so the time to live is the owner's number and the cache must fail open to the colo call. The estimate is a 30% cut in the three paths above; the measurement will say whether it held.

How it was priced. Transit Gateway data processed, down 30%: 12,000 GB × \$0.02 a GB = \$240.00; Direct Connect data out, down 30%: 8,400 GB × \$0.02 a GB = \$168.00; Cross-AZ transfer, down 30%, charged each way: 20,610 GB × \$0.01 a GB = \$206.10. The saving is **\$614.10 a month**, \$7,369.20 a year.

N-4: Second Direct Connect port to 1 Gbps

Owner: Owen Pradhan. **Goes in as:** pull request in seatlark-landing-zone. **Lands:** after the freeze.

Confidence: Medium. **Effort:** M. **Risk:** Medium. **Saves:** \$1,424 a month, \$17,082 a year.

What we saw. Two dedicated 10 Gbps ports ordered for wave 2's bulk copy; traffic peaks at 1.1 Gbps and runs about 380 Mbps at the 95th percentile. Of the 1.1 Gbps peak, 0.35 Gbps is the nightly backup copy to the colo; production traffic alone peaks at 0.75 Gbps.

The change. Replace the second port with a 1 Gbps port, and shape the nightly backup copy so that traffic on the failover port stays under 1 Gbps. It waits on a new 1 Gbps cross-connect, and a window clear of the wave 7 orders-database cutover, which uses the second port as its failover path. **Rollback:** Order the 10 Gbps port again (a new cross-connect).

The two dedicated 10 Gbps ports were ordered for wave 2's bulk copy. Since then traffic peaks at 1.1 Gbps and runs about 380 Mbps at the 95th percentile, so a 1 Gbps port does not carry the peak: as the failover path it would cap traffic below the 1.1 Gbps high. The peak is the nightly backup copy (0.35 Gbps) on top of production (0.75 Gbps at most), so the change includes shaping that copy: when traffic fails over to the 1 Gbps port, the backup job throttles to 200 Mbps or waits, and production keeps its headroom. Hosted 2 or 5 Gbps connections would avoid the shaping but are priced by the AWS Partner that provides them, not on the AWS Price List, so we did not price them. The risk is not capacity today: wave 7 moves the orders and checkout services and their SQL Server Enterprise database, and the second port is its failover path, so a 1 Gbps port has to be in service and tested before that cutover, and replacing it later means a new cross-connect. At the AWS Price List API rates of October 16, 2026 (\$2.25 an hour for 10 Gbps, \$0.30 for 1 Gbps), both ports cost \$3,285 a month and one of each \$1,862.

BASH

```
aws directconnect create-connection --location <location-code> --bandwidth 1Gbps --connection-  
name seatlark-dx-b-1g  
# then build the virtual interfaces on it and move the failover path before retiring the old  
port
```

How it was priced. Second port: 10 Gbps to 1 Gbps: 1 port × (\$2.25 – \$0.30) an hour × 730 hours = \$1,423.50. The saving is **\$1,423.50 a month**, \$17,082.00 a year.

Logging

Logging is \$18,112 a year in four findings, L-1 through L-4. Every log group was created with no expiry, so after 11 months they hold the logs of the whole migration; the lifted applications log at debug, the flow logs go to the most expensive destination, and the CloudWatch agent publishes a metric for every mount and every process.

The five log groups hold 22,528 GB and ingest 2,704 GB a month, at \$0.50 a GB ingested and \$0.03 a GB-month stored: /seatlark/marketplace/debug (1,000 GB a month, 9,800 GB stored, \$794 a month); /seatlark/vpc-flow (820 GB a month, 7,300 GB stored, \$629 a month); /seatlark/partner/debug (600 GB a month, 4,300 GB stored, \$429 a month); /seatlark/audit (184 GB a month, 568 GB stored, \$109 a month); /seatlark/application (100 GB a month, 560 GB stored, \$67 a month). None has a retention policy.

The four findings

L-1: CloudWatch Logs retention

Owner: Beth Morrow. **Goes in as:** pull request in seatlark-landing-zone. **Lands:** week 3.

Confidence: High. **Effort:** S. **Risk:** Low. **Saves:** \$584 a month, \$7,004 a year.

What we saw. Every log group was created with no expiry; after 11 months they hold 22 TB at \$0.03 a GB-month. No log group has been queried past 30 days in 90 days of Logs Insights history.

The change. Set retention in the account baseline: 30 days for application and flow logs, 90 days for audit groups; security and audit logs keep a year in S3 Glacier Instant Retrieval. **Rollback:** Raise the retention.

No log group has been queried past 30 days in 90 days of Logs Insights history. Application and flow logs keep 30 days and audit groups 90; security and audit logs also keep a year in S3 Glacier Instant Retrieval, which is what PCI DSS will ask of the payments wave. That S3 copy is small (about 184 GB a month) and left out of the estimate.

BASH

```
for G in $(aws logs describe-log-groups --log-group-name-prefix /seatlark \
  --query 'logGroups[?!retentionInDays && !contains(logGroupName, `audit`)].logGroupName' --
  output text); do
  aws logs put-retention-policy --log-group-name "$G" --retention-in-days 30
done
aws logs put-retention-policy --log-group-name /seatlark/audit --retention-in-days 90
```

How it was priced. Log storage no longer held (22 TB to the retention window): 19,456 GB × \$0.03 a GB-month = \$583.68. The saving is **\$583.68 a month**, \$7,004.16 a year.

L-2: Debug logging off

Owner: Aaron Feld. **Goes in as:** pull request in marketplace-infra. **Lands:** week 3. **Confidence:** Medium. **Effort:** S. **Risk:** Low. **Saves:** \$300 a month, \$3,600 a year.

What we saw. The lifted services log at debug; ingestion is \$0.50 a GB. 1,600 GB a month of debug ingestion across two services.

The change. Set the log level to info in the marketplace and partner services. **Rollback:** Set the level back.

The lifted services log at debug, which ingests 1,600 GB a month across the marketplace and partner services; at info they ingest 1,000. Services reload their level on their own schedules, which is why this is Medium confidence. The change is a role variable in marketplace-infra and partner-infra (app_log_level: info, was debug), applied with ansible-playbook site.yml --tags logging.

How it was priced. Debug ingestion avoided: 600 GB × \$0.50 a GB = \$300.00. The saving is **\$300.00 a month**, \$3,600.00 a year.

L-3: Flow logs to S3

Owner: Beth Morrow. **Goes in as:** pull request in seatlark-landing-zone. **Lands:** week 3.

Confidence: High. **Effort:** S. **Risk:** Low. **Saves:** \$205 a month, \$2,460 a year.

What we saw. Flow logs are delivered to CloudWatch Logs at the vended-logs rate (\$0.50 a GB). 820 GB a month of flow logs.

The change. Deliver to S3 (\$0.25 a GB) with a 30-day lifecycle. **Rollback:** Re-point the flow logs.

The same flow-log records go to S3 at the vended-log rate for S3 (\$0.25 a GB against \$0.50 to CloudWatch Logs) and expire after 30 days. Athena still queries them.

In seatlark-landing-zone, an AWS::EC2::FlowLog with LogDestinationType: s3 replaces the CloudWatch destination, and the bucket's lifecycle rule sets ExpirationInDays: 30.

How it was priced. Flow-log delivery: CloudWatch Logs to S3: 820 GB × (\$0.50 – \$0.25) a GB = \$205.00. The saving is **\$205.00 a month**, \$2,460.00 a year.

L-4: CloudWatch agent metrics

Owner: Beth Morrow. **Goes in as:** pull request in seatlark-landing-zone. **Lands:** week 4.

Confidence: High. **Effort:** S. **Risk:** Low. **Saves:** \$421 a month, \$5,047 a year.

What we saw. The CloudWatch agent publishes per-mount and per-process metrics at \$0.30 each. 2,400 custom metrics; 1,400 are per-mount and per-process series nobody alarms on.

The change. A shared agent configuration that keeps host-level metrics only. **Rollback:** Restore the previous agent configuration.

The CloudWatch agent publishes 2,400 custom metrics today. One shared configuration brings it to 1,000 by dropping 1,400 per-mount and per-process series that nobody alarms on. Ansible installs it (ansible.builtin.copy to /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json, then amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:... as a handler):

JSON

```
{
  "agent": { "metrics_collection_interval": 60 },
  "metrics": {
    "append_dimensions": { "InstanceId": "${aws:InstanceId}" },
    "aggregation_dimensions": [ ["InstanceId"] ],
    "metrics_collected": {
      "mem": { "measurement": [ "mem_used_percent" ] },
      "disk": { "measurement": [ "used_percent" ], "resources": [ "/" ], "drop_device": true }
    }
  }
}
```

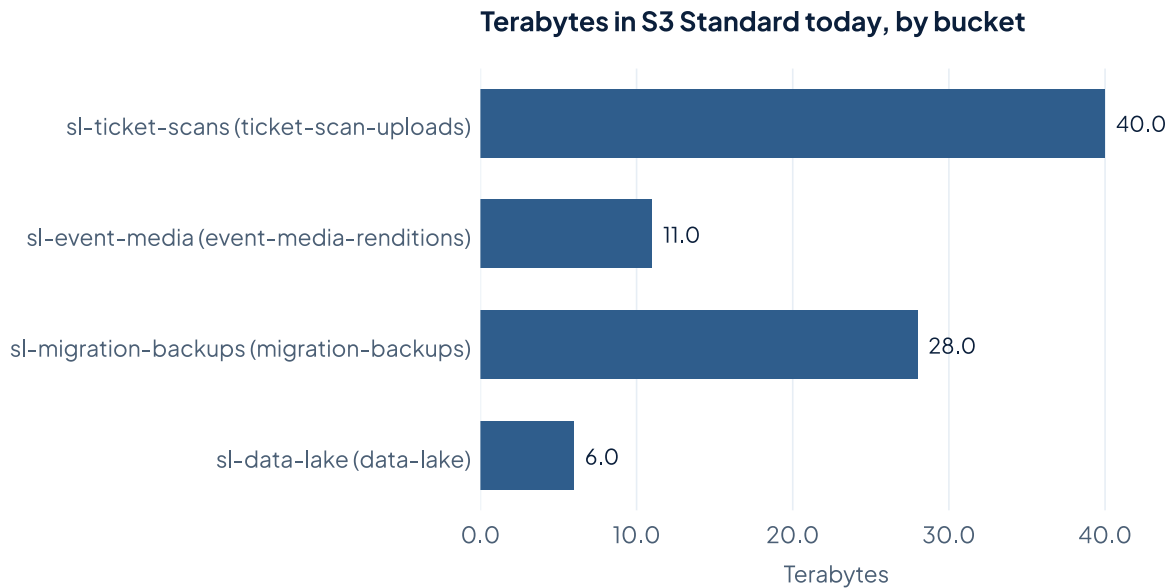
How it was priced. Custom metrics dropped: 1,400 metrics × \$0.30 a metric-month = \$420.00; PutMetricData requests for those metrics: 61,320 requests × \$0.00001 a request = \$0.61. The saving is **\$420.61 a month**, \$5,047.36 a year.

S3 lifecycle

S3 is \$16,245 a year in three findings, S3-1 through S3-3, and the category with the one-time charge worth reading: the transitions in S3-1 cost \$3,102 once (6.6 months of its own saving), and S3-2's 300,000 objects cost \$15.00. Seatlark keeps 85 TB in S3 Standard, in four buckets:

The four buckets, from S3 Inventory (age) and server access logs (reads).

Bucket	Data class	TB	Objects	Average object	Older than 90 days	Read a month from old objects
sl-ticket-scans	ticket-scan-uploads	40.0	238,615,385	175 KB	65%	40 GB
sl-event-media	event-media-renditions	11.0	12,000,000	920 KB	80%	300 GB
sl-migration-backups	migration-backups	28.0	300,000	95,000 KB	100%	0 GB
sl-data-lake	data-lake	6.0	4,000,000	1,500 KB	50%	20 GB



Terabytes in S3 Standard today. The ticket scans and the migration backups are 80% of it.

Three prices decide these findings. Standard is priced in tiers across the organization ([S3 is priced on the whole organization](#)). Glacier Instant Retrieval stores at \$0.004 a GB-month and Deep Archive at \$0.00099. And a lifecycle transition is a request that costs money per object: \$0.02 per 1,000 to Glacier Instant Retrieval, which on 155 million objects is \$3,102, once.

The three findings

S3-1: Ticket scans to Glacier Instant Retrieval

Owner: Kara Lindqvist. **Goes in as:** pull request in data-content-infra. **Lands:** week 4.

Confidence: Medium. **Effort:** S. **Risk:** Low. **Saves:** \$473 a month, \$5,674 a year.

What we saw. 26 TB of ticket PDFs, barcode images, and transfer proofs older than 120 days in S3 Standard at \$0.022 a GB-month. Sellers upload them, read heavily until the event and through the 120-day chargeback window, then almost never. About 155 million objects at \$0.02 per 1,000 transitions is a one-time charge; objects under 128 KB are billed as 128 KB, so the estimate carries a 6% allowance. Intelligent-Tiering was considered and rejected: its monitoring fee on 155 million objects would eat most of the saving.

The change. A lifecycle rule to Glacier Instant Retrieval at 120 days (\$0.004 a GB-month, millisecond reads for a rare dispute). **Rollback:** Copy the objects back to Standard with S3 Batch Operations (retrieval \$0.03 a GB); a lifecycle rule only moves objects toward colder classes.

Sellers upload ticket PDFs, barcode images, and transfer proofs, read heavily until the event and through the 120-day chargeback window, then almost never: the bucket is 65% older than 90 days, and those bytes see 40 GB of reads a month. A rare dispute reads in milliseconds at \$0.03 a GB retrieved. **Two honest costs.** The transition charge, about 155 million objects at \$0.02 per 1,000, is \$3,102 once, repaid in 6.6 months. And Glacier Instant Retrieval bills an object under 128 KB as 128 KB while S3's default does not transition such objects at all; the average ticket scan is 175 KB, so most clear it, and the estimate carries a 6% allowance on the bytes it moves.

Intelligent-Tiering was considered and rejected: its monitoring fee is \$0.0025 per 1,000 objects a month, so on the same 155 million objects it would cost \$388 a month, most of what the lifecycle rule saves.

YAML

```
# data-content-infra: sl-ticket-scans
LifecycleConfiguration:
  Rules:
    - Id: ticket-scans-to-glacier-ir
      Status: Enabled
      Transitions: [{ StorageClass: GLACIER_IR, TransitionInDays: 120 }]
```

How it was priced. Ticket scans, 26 TB: leaves Standard at the \$0.022 tier: 26,624 GB × \$0.022 a GB-month = \$585.73; Held in Glacier Instant Retrieval, with the 6% small-object allowance: 28,221.44 GB × \$0.004 a GB-month = -\$112.89. The saving is **\$472.84 a month**, \$5,674.11 a year.

S3-2: Migration backups to Deep Archive

Owner: Owen Pradhan. **Goes in as:** pull request in seatlark-landing-zone. **Lands:** week 3.

Confidence: High. **Effort:** S. **Risk:** Low. **Saves:** \$622 a month, \$7,462 a year.

What we saw. 28 TB of colo database dumps and VM exports in S3 Standard. Written once during the waves; read by nobody since.

The change. Move to S3 Glacier Deep Archive (180-day minimum), kept a year. **Rollback:** Restore on request (standard retrieval \$0.02 a GB, 12 hours).

The 28 TB of colo database dumps and VM exports were written once during the waves and read by nobody since. Deep Archive has a 180-day minimum and is kept a year; the transition is a one-time \$15.00 (300,000 objects at \$0.05 per 1,000), and a restore costs \$0.02 a GB at the standard tier and takes up to 12 hours. The rule is the same shape as S3-1 with `StorageClass: DEEP_ARCHIVE` and `TransitionInDays: 0`, and it does not expire: what happens after the year is the owner's decision. Priced after S3-1, so its gigabytes save two tier rates, as the arithmetic shows.

How it was priced. Migration backups, 28 TB: leaves Standard at the \$0.022 tier: 9,216 GB × \$0.022 a GB-month = \$202.75; Migration backups, 28 TB: leaves Standard at the \$0.023 tier: 19,456 GB × \$0.023 a GB-month = \$447.49; Held in Glacier Deep Archive: 28,672 GB × \$0.00099 a GB-month = −\$28.39. The saving is **\$621.85 a month**, \$7,462.26 a year.

S3–3: Regenerable image renditions

Owner: Diego Saldana. **Goes in as:** pull request in data-content-infra. **Lands:** week 4.

Confidence: High. **Effort:** S. **Risk:** Low. **Saves:** \$259 a month, \$3,109 a year.

What we saw. 11 TB of image renditions for past events. Past-event renditions are regenerable from the originals.

The change. Delete; the originals regenerate them on demand. **Rollback:** Regenerate from the originals.

Past-event renditions regenerate from the originals, which is why deleting them is safe in principle. But the event-media bucket still sees 300 GB of reads a month from objects older than 90 days, and a deleted object is gone. Before anything is deleted, find out who reads old renditions; the result of the query decides whether the change goes ahead.

SQL

```
-- S3 server access logs in Athena: who still reads past-event renditions?
SELECT referrer, requester, COUNT(*) AS gets, ROUND(SUM(bytesent) / 1e9, 1) AS gb_sent
FROM s3_access_logs.sl_event_media
WHERE operation = 'REST.GET.OBJECT' AND key LIKE 'renditions/%'
      AND parse_datetime(requestdatetime, 'dd/MMM/yyyy:HH:mm:ss Z') > current_timestamp - INTERVAL
      '30' DAY
GROUP BY 1, 2 ORDER BY gets DESC LIMIT 50;
```

How it was priced. Event renditions, 11 TB: leaves Standard at the \$0.023 tier: 11,264 GB × \$0.023 a GB-month = \$259.07. The saving is **\$259.07 a month**, \$3,108.86 a year.

Graviton

Graviton is \$6,055 a year in two findings, and the smallest category. Lift-and-shift AMIs are x86 images, so Graviton means rebuilding for arm64, not resizing. At list prices it saves 15% on the general-purpose family: an m5.xlarge at \$0.192 an hour against an m7g.xlarge at \$0.1632. It is a modest number, rated Medium, and priced last of the instance findings so it applies to the sizes rightsizing leaves.

The two findings

G-1: Marketplace web and API to Graviton

Owner: Aaron Feld. **Goes in as:** pull request in marketplace-infra. **Lands:** after the freeze.

Confidence: Medium. **Effort:** L. **Risk:** Medium. **Saves:** \$378 a month, \$4,541 a year.

What we saw. x86 AMIs lifted from the colo; the marketplace is Java on Amazon Corretto. At list prices Graviton saves 15% on the general-purpose family (m5.xlarge \$0.192 against m7g.xlarge \$0.1632).

The change. Rebuild as arm64 images with the July container replatform and move the web and API tiers to m7g. It waits on the July container replatform, which builds arm64 images. **Rollback:** Redeploy the x86 image.

The marketplace is Java on Amazon Corretto, which runs on arm64, and the move rides with the July container replatform instead of its own change window. The web tier moves at its current size and the API tier at the size [R-1: Marketplace API tier to m5.xlarge](#) leaves; the launch template takes an arm64 AMI (/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-arm64 from Systems Manager Parameter Store) and m7g types.

How it was priced. Web hosts: m5.2xlarge to m7g.2xlarge: 6 hosts × (\$0.3840 – \$0.3264) an hour × 730 hours = \$252.29; API hosts: m5.xlarge to m7g.xlarge: 6 hosts × (\$0.1920 – \$0.1632) an hour × 730 hours = \$126.14. The saving is **\$378.43 a month**, \$4,541.18 a year.

G-2: Partner APIs to Graviton

Owner: Diego Saldana. **Goes in as:** pull request in partner-infra. **Lands:** after the freeze.

Confidence: Medium. **Effort:** M. **Risk:** Medium. **Saves:** \$126 a month, \$1,514 a year.

What we saw. x86 AMIs; the partner APIs are Go and need only an arm64 build. The same 15% as G-1; modest and rated medium.

The change. Add an arm64 build to the pipeline, validate, and move to m7g.xlarge. It waits on an arm64 build of the partner APIs, validated in staging. **Rollback:** Redeploy the x86 image.

The partner APIs are Go and need only an arm64 build in the pipeline (G00S=linux G0ARCH=arm64 CGO_ENABLED=0 go build), validated in staging before production moves.

How it was priced. Partner API hosts: m5.xlarge to m7g.xlarge: 6 hosts × (\$0.1920 – \$0.1632) an hour × 730 hours = \$126.14. The saving is **\$126.14 a month**, \$1,513.73 a year.

Deliverable 4: the commitment plan

Seatlark holds no commitment of any kind, and the biggest single decision in the engagement is whether to keep waiting. We recommend one purchase: a **one-year No Upfront Compute Savings Plan at \$5.00 an hour (C-1)**, worth **\$15,950 a year**. Lydia Brandt, Vice President, Finance, owns the decision, and Seatlark's finance team buys it in the management account. Vantalect does not purchase commitments. It is the one purchase in the report that cannot be undone once its seven-day return window closes: past that window a Savings Plan cannot be canceled, which is why it is sized below the floor that will survive every remaining change.

Why waiting for the migration to finish is half right

The team's instinct is sound for most commitments. A three-year plan or a reserved instance sized today would strand money when the remaining waves change the estate: orders and checkout in wave 7 alone add \$22,000 a month, and the PCI zone, the data warehouse, and corporate IT follow. Database commitments wait for the catalog resize (D-3) and the orders database.

But a 24-hour floor will certainly remain: the web and API tiers, the search cluster, ingestion, the partner APIs, the CI runners, and shared services, 30 hosts that run every hour. After the changes proposed for weeks 3 and 4 that floor costs **\$11.63 an hour On-Demand**, and \$8.53 an hour at a one-year Compute plan's rates, 26.7% off. A **Compute** plan covers any instance family, size, or Region, and Fargate and Lambda, so the July container replatform, the Graviton moves, and the ingestion resize do not strand it.

How the size was found

The plan is sized last, on the estate after every change in place, so it never covers hours another finding has already removed. The floor, by workload:

The hosts that run every hour and will still be there after the remaining waves. The last two columns apply the roadmap's remaining changes, including the ones proposed for after the freeze (R-3, G-1, G-2): On-Demand falls to \$9.41 an hour and the plan rate to \$6.89.

Workload	Hosts	Size after weeks 3 and 4	On-Demand an hour	Plan rate an hour	Size after every change	Plan rate then
Marketplace web	6	m5.2xlarge	\$2.304	\$1.692	m7g.2xlarge	\$1.439
Marketplace API	6	m5.xlarge	\$1.152	\$0.846	m7g.xlarge	\$0.719
Listing ingestion	4	m5.4xlarge	\$3.072	\$2.256	m5.2xlarge	\$1.128
Partner APIs	6	m5.xlarge	\$1.152	\$0.846	m7g.xlarge	\$0.719
Production search	3	r5.2xlarge	\$1.512	\$1.095	r5.2xlarge	\$1.095
CI runners	3	m5.4xlarge	\$2.304	\$1.692	m5.4xlarge	\$1.692
Bastion	1	t3.medium	\$0.042	\$0.030	t3.medium	\$0.030
Tooling	1	m5.large	\$0.096	\$0.071	m5.large	\$0.071
Total	30		\$11.634	\$8.528		\$6.894

A commitment of \$5.00 an hour at the plan's rates pays for \$6.82 an hour of On-Demand usage, 59% of the floor: \$4,979.20 of On-Demand usage a month less the \$3,650.00 commitment saves \$1,329.20 (C-1: [Compute Savings Plan](#)). The remaining 41% of the floor, \$3,513 a month, stays On-Demand.

Commitment sizes against the floor after every roadmap change. The bold row is C-1. A size above the plan rate left after every change (\$6.89 an hour) pays for hours that are not there.

Commitment an hour	Covers (On-Demand)	Share of the floor	Saves a month	Plan used after every change	Unused a month then
\$3.00	\$4.09	35%	\$798	100%	\$0
\$4.00	\$5.46	47%	\$1,063	100%	\$0
\$5.00	\$6.82	59%	\$1,329	100%	\$0
\$6.00	\$8.19	70%	\$1,595	100%	\$0
\$7.00	\$9.55	82%	\$1,861	98%	\$78
\$8.00	\$10.91	94%	\$2,127	86%	\$808

At \$5.00 an hour the plan stays 100% used after every change, with 27% to spare before an hour goes unused. That margin is deliberate: the web tier has no autoscaling yet, and autoscaling would lower the always-on floor; the July replatform may move hosts to containers on a different footprint; and a plan that stays fully used if some changes are delayed or never made is worth more than the extra dollars of a larger one.

A three-year plan would save 49.1% against 26.7% for the same coverage, about \$1,118 a month more, and is the right buy at colo exit, when the end-state estate is known.

The finding

C-1: Compute Savings Plan

Owner: Lydia Brandt. **Goes in as:** change ticket with a scripted runbook. **Lands:** week 4.

Confidence: High. **Effort:** S. **Risk:** Low. **Saves:** \$1,329 a month, \$15,950 a year.

What we saw. No commitments of any kind; the 24-hour floor of \$11.63 an hour runs On-Demand. The floor after every change in place costs \$11.63 an hour On-Demand and \$8.53 at the plan's rates (26.7% off); after every deferred rightsizing and Graviton change it is still \$9.41 an hour On-Demand.

The change. A one-year No Upfront Compute Savings Plan at \$5.00 an hour, covering \$6.82 an hour of On-Demand usage (59% of the floor). **Rollback:** Return it within seven days of purchase, in the same calendar month (AWS allows returns of plans of \$100 an hour or less); after that it cannot be canceled, which is why it is sized below the floor that will survive every remaining change.

Risk. Final once bought, and low only in the sense that the plan is sized below even the floor that remains after every change. Confidence is High and effort is S: the saving does not depend on anything the other findings might fail to deliver. The way to buy it:

BASH

```
# 1. Find the offering for the type, term, and payment option. One year is 31536000 seconds.
aws savingsplans describe-savings-plans-offerings --region us-east-1 \
  --plan-types Compute --payment-options "No Upfront" --durations 31536000 --currencies USD \
  --query 'searchResults[.
{id:offeringId,type:planType,term:durationSeconds,payment:paymentOption}{'

# 2. Purchase in the management account, once the owner has approved it. It can be returned
within 7 days, in the same calendar month (plans of $100 an hour or less); after that it cannot
be canceled.
aws savingsplans create-savings-plan --region us-east-1 \
  --savings-plan-offering-id <offering-id> \
  --commitment 5.00 \
  --client-token sl-compute-sp-1yr \
  --tags '{"sl:approved-by": "<approver>", "sl:change-ticket": "<ticket-id>"}'

# 3. Check Cost Explorer's own recommendation before buying, and the plan's utilization after.
aws ce get-savings-plans-purchase-recommendation --region us-east-1 \
  --savings-plans-type COMPUTE_SP --term-in-years ONE_YEAR --payment-option NO_UPFRONT --
  lookback-period-in-days SIXTY_DAYS
```

Measured at week six, the plan is judged by its ARN: its fee, the usage it covered, and its utilization.

How it was priced. On-Demand usage the plan covers: \$6.8208 an hour × 730 hours = \$4,979.20; The hourly commitment, paid whether or not it is used: \$5.0000 an hour × 730 hours = – \$3,650.00. The saving is **\$1,329.20 a month**, \$15,950.43 a year.

Deliverable 6: the savings roadmap

The roadmap is every change in this report, ranked by annual savings, with its confidence, effort, risk, owner, and the week it would land. **30 changes are worth \$246,839 a year.** The ten largest are \$141,907 of it (57%), and the five largest \$89,961. Every change has a named owner, and the owner decides.

The waves

The waves are what the Optimization's weeks 3 and 4 would carry out, clear of the freeze, and what Seatlark's own team does afterward.

The three waves and what each carries. Week 5 (the week of June 8, 2026) is inside the production freeze and carries no production change.

Wave	Dates	What lands	Changes	Per year
Week 3	Week of May 25, 2026 (four working days: May 25, 2026 is Memorial Day)	Waste, schedules, non-production rightsizing and databases, storage conversion, S3 endpoints and NAT consolidation, log retention and level, flow logs, migration backups to Deep Archive	16	\$113,530
Week 4	Week of June 1, 2026, production changes on June 2, 2026 and June 3, 2026	Production rightsizing, the production search volumes, the agent metrics, the ticket-scan lifecycle, renditions after the access-log check, and the Savings Plan	7	\$60,764
After the freeze	June 15, 2026 or later	Changes that wait on a load test, a notice period, an application change, or an arm64 build	7	\$72,545

The last wave is \$72,545 a year that cannot land inside six weeks:

- **D-3, Catalog database: downsize and Graviton** (\$19,290 a year, Imani Brooks) waits on a load test that replays an on-sale, then a maintenance window.
- **N-4, Second Direct Connect port to 1 Gbps** (\$17,082 a year, Owen Pradhan) waits on a new 1 Gbps cross-connect, and a window clear of the wave 7 orders-database cutover, which uses the second port as its failover path.
- **R-3, Listing ingestion workers to m5.2xlarge** (\$13,455 a year, Imani Brooks) waits on the load test, run as a perf-test exercise after the freeze, because Compute Optimizer never saw an on-sale.
- **D-2, Partner database: downsize and Graviton** (\$9,294 a year, Diego Saldana) waits on a load test against a restored copy at the new class, then ten business days' notice of a maintenance window, which Seatlark Connect's partner contracts require.
- **N-3, Cache the colo order-status calls** (\$7,369 a year, Aaron Feld) waits on an application change: a short-TTL cache for order status and held inventory.
- **G-1, Marketplace web and API to Graviton** (\$4,541 a year, Aaron Feld) waits on the July container replatform, which builds arm64 images.
- **G-2, Partner APIs to Graviton** (\$1,514 a year, Diego Saldana) waits on an arm64 build of the partner APIs, validated in staging.

The 30 changes, ranked

Every change, ranked by annual savings. Rating is confidence, effort (S a day or less, M a few days, L needs an application change), and risk. How each goes in (pull request or change ticket) is in its section.

Rank	ID	Change	Owner	Lands	Rating	Per year	Running share
1	S-1	Schedules for development, partner non-production, and perf-test	Beth Morrow	Week 3	High, S, Low	\$24,183	10%
2	D-3	Catalog database: downsize and Graviton	Imani Brooks	After the freeze	Medium, M, Medium	\$19,290	18%
3	N-4	Second Direct Connect port to 1 Gbps	Owen Pradhan	After the freeze	Medium, M, Medium	\$17,082	25%
4	C-1	Compute Savings Plan	Lydia Brandt	Week 4	High, S, Low	\$15,950	31%
5	R-3	Listing ingestion workers to m5.2xlarge	Imani Brooks	After the freeze	Medium, M, Medium	\$13,455	36%
6	W-4	Snapshot and AMI sprawl	Beth Morrow	Week 3	Medium, M, Low	\$11,412	41%
7	E-2b	Search cluster io1 to gp3, production	Imani Brooks	Week 4	High, S, Low	\$10,800	45%
8	R-1	Marketplace API tier to m5.xlarge	Aaron Feld	Week 4	High, M, Medium	\$10,092	50%
9	R-2	Partner APIs to m5.xlarge	Diego Saldana	Week 4	High, M, Medium	\$10,092	54%
10	D-1	Multi-AZ off on two non-production databases	Beth Morrow	Week 3	High, S, Low	\$9,552	57%
11	R-4	Non-production search and partner hosts one size down	Beth Morrow	Week 3	High, S, Low	\$9,461	61%
12	D-2	Partner database: downsize and Graviton	Diego Saldana	After the freeze	Medium, M, Medium	\$9,294	65%
13	W-2	MGN replication servers from completed waves	Beth Morrow	Week 3	High, S, Low	\$7,927	68%
14	S3-2	Migration backups to Deep Archive	Owen Pradhan	Week 3	High, S, Low	\$7,462	71%
15	N-3	Cache the colo order-status calls	Aaron Feld	After the freeze	Medium, L, Medium	\$7,369	74%

Rank	ID	Change	Owner	Lands	Rating	Per year	Running share
16	N-1	S3 gateway endpoints in every VPC with a NAT gateway	Beth Morrow	Week 3	High, S, Low	\$7,188	77%
17	L-1	CloudWatch Logs retention	Beth Morrow	Week 3	High, S, Low	\$7,004	80%
18	W-1	Unattached volumes from test cutovers	Beth Morrow	Week 3	High, S, Low	\$6,000	82%
19	S3-1	Ticket scans to Glacier Instant Retrieval	Kara Lindqvist	Week 4	Medium, S, Low	\$5,674	85%
20	S-2	Schedule for staging	Aaron Feld	Week 3	Medium, S, Low	\$5,308	87%
21	L-4	CloudWatch agent metrics	Beth Morrow	Week 4	High, S, Low	\$5,047	89%
22	G-1	Marketplace web and API to Graviton	Aaron Feld	After the freeze	Medium, L, Medium	\$4,541	91%
23	E-1	gp2 to gp3 on attached volumes	Beth Morrow	Week 3	High, S, Low	\$4,224	93%
24	E-2a	Search cluster io1 to gp3, staging	Imani Brooks	Week 3	High, S, Low	\$4,200	94%
25	L-2	Debug logging off	Aaron Feld	Week 3	Medium, S, Low	\$3,600	96%
26	N-2	One NAT gateway per non-production VPC	Beth Morrow	Week 3	High, S, Low	\$3,154	97%
27	S3-3	Regenerable image renditions	Diego Saldana	Week 4	High, S, Low	\$3,109	98%
28	L-3	Flow logs to S3	Beth Morrow	Week 3	High, S, Low	\$2,460	99%
29	G-2	Partner APIs to Graviton	Diego Saldana	After the freeze	Medium, M, Medium	\$1,514	100%
30	W-3	Idle load balancers from on-sale rehearsals	Aaron Feld	Week 3	High, S, Low	\$394	100%
Total						\$246,839	100%

Owners, and what it costs to get there

Every change has a named owner. Beth Morrow (Manager, Site Reliability Engineering) owns 12 changes worth \$97,612 a year; Imani Brooks (Director, Inventory and Search) owns 4 changes worth \$47,745 a year; Aaron Feld (Director, Marketplace Engineering) owns 6 changes worth \$31,304 a year; Owen Pradhan (Vice President, Infrastructure) owns 2 changes worth \$24,544 a year; Diego Saldana (Director, Partner Engineering) owns 4 changes worth \$24,008 a year; Lydia Brandt (Vice President, Finance) owns 1 change worth \$15,950 a year; Kara Lindqvist (Manager, Fulfillment and Payments Engineering) owns 1 change worth \$5,674 a year. S3-1 carries a one-time charge of \$3,102.00, repaid in 6.6 months; S3-2 carries a one-time charge of \$15.00, repaid within a few days. Every other change costs an engineer's time and nothing else.

The order that matters

Waste goes first, because it needs no design and shrinks what everything after is sized on. Rightsizing and schedules come before the commitment, so the production resizes (R-1, R-2) land first and the plan is bought the day after. And the freeze is a wall: everything in production is done on June 2, 2026 and June 3, 2026 or after June 14, 2026, never between, and a change that cannot make the first window goes to the last wave. The engagement closes on June 18, 2026.

Business Support+ follows the bill

AWS Business Support+ is the greater of \$29 a month or a percentage of monthly AWS charges, measured before discounts or credits:

The tiers, from the AWS Support pricing page on October 16, 2026: \$0 to \$10,000, 9%; \$10,000 to \$80,000, 7%; \$80,000 to \$250,000, 5%; above \$250,000, 3%.

Seatlark pays \$3,566.51 a month, 6.9% of its bill. At the margin it sits in the 7% band, so each dollar saved lowers the fee by 7% of a dollar. With the 29 changes other than the Savings Plan in place the fee would be \$2,219.66, a fall of **\$1,346.85 a month (\$16,162 a year)**. That fall is real, and it is **not counted** in any number in this report or toward the gate: it is the AWS fee following the bill down, not a change anyone makes. We assume nothing about whether the Savings Plan's discount lowers the fee further.

The guarantee

Where this stands against the guarantee

The Optimization's second invoice, \$17,500, is issued only if the savings in place at the end of week six, measured change by change on Seatlark's Cost and Usage Report and annualized, are at least three times the \$35,000 fee: **\$105,000 a year**. This roadmap identifies \$246,839, 2.4 times the gate, and the 19 High-confidence changes alone are \$148,299, 1.4 times it.

That multiple is an estimate and not a measurement. What counts at week six is what is in place and visible on the bill: a change that is declined, deferred past the six weeks, or approved and later declined counts for nothing. The cost avoidance on the remaining migration waves ([The migration trajectory and cost avoidance](#)) is not measured, not counted toward the gate, and not a saving on the bill, and it is in no multiple.

The migration trajectory and cost avoidance

Not a saving

Everything in this section is not measured, not counted toward the gate, and not a saving on the bill. There is no "before" on the Cost and Usage Report for a workload that has not moved. It is a model of what Seatlark's own plan costs and what the patterns in this report could keep out of it. It is in no total, no multiple, and no part of the gate.

Seatlark's migration plan is its own, not ours: each remaining wave is priced the way the first six were sized, lifted at the colo allocation. At colo exit on June 30, 2027 the plan reaches **\$125,300 a month, \$1.5 million a year**, from \$49,600 in April. That matches the board's migration business case, which we read as the end-state annual run rate (assumption A1, to be confirmed); if it was instead the total spend over the next twelve months, the end state is higher and the avoidance below scales by 1.6 to 2.1 times.

The remaining waves in Seatlark's own plan. Wave 6 raises the bill during the engagement; that growth is never netted against the savings.

Wave	When	What moves	Adds per month	Run rate after
6	May 30, 2026 (during the engagement)	Partner reporting and the affiliate feed generator	\$2,600	\$54,300
7	July to August 2026	Orders and checkout, and the SQL Server Enterprise database (9 TB)	\$22,000	\$76,300
8	October to November 2026	Broker Desk and the remaining SSIS hosts	\$12,000	\$88,300
9	January to March 2027	Payments and fulfillment (the PCI zone)	\$18,000	\$106,300
10	March to May 2027	The data warehouse (14 TB)	\$14,000	\$120,300
11	May to June 2027	Corporate IT; colo exit June 30, 2027	\$5,000	\$125,300

What the same patterns could keep out of the remaining waves

Week 5 puts the patterns that found this report's savings into the landing zone and the wave runbook before the remaining waves move, so they land right and do not need fixing later. The method applies to each remaining wave the share of its planned cost that this engagement's findings cut from the first six, then a capture rate for how much the guardrails and runbook prevent without anyone remembering to. The remaining waves add \$71,000 a month.

The levers and the share of a wave's planned cost each could keep out. The totals are not measured, not counted toward the gate, and not a saving on the bill.

Lever, built in week 5	Share of a wave's planned cost	Per month on the remaining waves
The wave runbook sizes from 30-day vCenter 95th-percentile use, not allocation	7.0% to 10.0%	\$4,970 to \$7,100
Instance Scheduler on by default in non-production accounts; Multi-AZ only in production	2.0% to 4.0%	\$1,420 to \$2,840
An AWS Config rule and an SCP that refuse new gp2 and io1 volumes without an exception tag	1.0% to 2.0%	\$710 to \$1,420
Log-group retention set by the account baseline; agent metrics from a shared configuration	1.0% to 2.0%	\$710 to \$1,420
The VPC template: S3 and DynamoDB gateway endpoints, one NAT gateway in non-production	0.5% to 1.0%	\$355 to \$710
MGN cleanup in the cutover checklist (finalize, disconnect, archive)	0.5% to 1.0%	\$355 to \$710
Total	12% to 20%	\$8,520 to \$14,200

At those shares the avoidance at colo exit is **\$8,520 to \$14,200 a month, \$100,000 to \$170,000 a year** at the end-state run rate. It is not measured, not counted toward the gate, and not a saving on the bill. The shares are planning assumptions, not measurements.

Two larger numbers, named and not claimed

Two other items are larger than any lever above. Neither is in this report's totals, and neither is claimed as avoidance:

- 1. Retiring the Direct Connect ports at colo exit.** Two dedicated 10 Gbps ports cost up to \$3,285 a month (\$39,420 a year) and are already in Seatlark's plan to end with the colo. [N-4: Second Direct Connect port to 1 Gbps](#) is the only part of it this report prices, and only for a smaller second port in the meantime.
- 2. Moving the orders database off SQL Server Enterprise.** Wave 7 brings the SQL Server Enterprise database to AWS as it is. Moving it to a different engine, such as Aurora PostgreSQL, would change its licensing cost, but it is an application and licensing decision beyond the scope of this engagement. We do not size it.

Appendix A: the registers

Three CSV files carry the full detail behind this report. They sit under `docs/samples/seatlark/data/cost/` and are the registers as they stand at the end of the engagement: the Analysis columns are this report's, and the status columns are maintained by the later reports.

`waste-register.csv` (4 rows, Deliverable 2), `rightsizing-plan.csv` (4 rows, Deliverable 3), and `savings-roadmap.csv` (30 rows, every change, Deliverable 6). The columns are the finding's id, category, title, account, owner, savings, one-time cost, and ratings, then the status columns the later reports maintain.

Appendix B: assumptions

- **Estimates at list prices**, at the May run rate of \$51,660 a month, with no private pricing agreement. Week 6 replaces each with a measurement.
- **Support follows the bill** by about \$1,347 a month at full roadmap, and is not counted ([Business Support+ follows the bill](#)).
- **The honest limits** are in the findings: incremental snapshots (W-4), the MGN per-server fee, which May's bill does not carry (W-2), Compute Optimizer's 14-day window (R-3), the web tier that keeps its size, RDS gp3 saving nothing, the 128 KB S3 minimum (S3-1), and the shaping a 1 Gbps failover port needs (N-4).
- **The Savings Plan (C-1)** is a proposal until finance approves it, and **the migration's cost avoidance** ([The migration trajectory and cost avoidance](#)) is not measured, not counted toward the gate, and not a saving on the bill.

Appendix C: price provenance

Every price in the model and in this report is a us-east-1 AWS list price from the AWS Price List API, retrieved October 13, 2026, except Direct Connect, AWS DMS, and AWS Application Migration Service (October 16, 2026) and AWS Business Support+ (the AWS Support pricing page, October 16, 2026, in [Business Support+ follows the bill](#)). Savings Plans effective rates come from the AWS Savings Plans API (`describe-savings-plans-offering-rates`, Compute, Linux, shared tenancy, one and three years), retrieved October 13, 2026.

The prices behind the model.

Price	Value	Unit
EC2 m5.large / .xlarge / .2xlarge / .4xlarge	\$0.096 / \$0.192 / \$0.384 / \$0.768	an hour
EC2 r5.xlarge / r5.2xlarge / t3.small	\$0.252 / \$0.504 / \$0.0208	an hour
EC2 Graviton m7g.xlarge / m7g.2xlarge	\$0.1632 / \$0.3264	an hour
EBS gp2 / gp3 storage, gp3 IOPS above 3,000	\$0.10 / \$0.08 / \$0.005	a GB-month / a GB-month / an IOPS-month
EBS io1 storage / IOPS; snapshot standard / archive	\$0.125 / \$0.065 / \$0.05 / \$0.0125	a GB-month / an IOPS-month / a GB-month / a GB-month
Application Load Balancer; NAT gateway hour / GB	\$0.0225 / \$0.045 / \$0.045	an hour / an hour / a GB
Transit Gateway data; cross-AZ each way	\$0.02 / \$0.01	a GB
Direct Connect port 1 Gbps / 10 Gbps; data out	\$0.30 / \$2.25 / \$0.02	an hour / an hour / a GB
CloudWatch Logs ingest / storage; custom metric	\$0.50 / \$0.03 / \$0.30	a GB / a GB-month / a metric-month
Vended flow logs to CloudWatch Logs / to S3	\$0.50 / \$0.25	a GB
S3 Standard tiers (organization)	\$0.023 / \$0.022 / \$0.021	a GB-month
S3 Glacier Instant Retrieval / Deep Archive; transition to each	\$0.004 / \$0.00099 / \$0.02 / \$0.05	a GB-month / a GB-month / per 1,000 requests / per 1,000 requests
AWS MGN replication, after the free period	\$0.042	a source-server hour

© 2026 Vantalect, LLC. Sample deliverable; Seatlark, Inc. is fictional.

AWS, Amazon Web Services, AWS Graviton, and the AWS service names are trademarks of Amazon.com, Inc. or its affiliates. Ansible is a trademark of Red Hat, Inc. Elasticsearch is a trademark of Elasticsearch B.V. PostgreSQL is a registered trademark of the PostgreSQL Community Association of Canada. MySQL is a trademark of Oracle Corporation and its affiliates. Redis is a trademark of Redis Ltd. Microsoft and SQL Server are trademarks of the Microsoft group of companies. VMware and vCenter are trademarks of Broadcom Inc. NetApp is a trademark of NetApp, Inc. Java is a trademark of Oracle Corporation and its affiliates. Vantalect is not affiliated with or endorsed by any of them. Seatlark, Inc. is fictional.