Context-Aware Cloud Migration

How Synthesized Infrastructure Intelligence Transforms Multi-Cloud Strategy

A CloudGo.ai Case Study: GCP-to-AWS Migration with [Redacted] Analytics

1. Introduction

Cloud migrations fail not from lack of technical knowledge, but from lack of organizational context. The gap between generic migration playbooks and executable migration plans is filled by tribal knowledge, scattered documentation, and institutional memory that rarely surfaces in planning sessions. Engineering teams routinely discover, mid-migration, that a committed use discount makes their timeline financially ruinous, that a key customer contract limits what truly needs to move, or that a past incident postmortem already documented why a particular approach will fail.

This case study demonstrates a fundamental thesis: migration recommendations change dramatically when infrastructure telemetry, cost commitments, compliance constraints, and operational history are synthesized alongside cloud best practices. We examine a GCP-to-AWS migration where full-context analysis reduced critical-path scope by approximately 70% and compressed the compliance timeline from seven months to eight weeks.

The subject company, [Redacted] Analytics, faced an enterprise customer mandate requiring AWS infrastructure by Q3. A generic migration assessment recommended full platform replatforming, including a complex BigQuery-to-Redshift migration. CloudGo.ai's context-aware analysis revealed that the customer accessed only 20% of the platform, that $180K in GCP commitments would be forfeited by full migration, and that the riskiest workflow (BQML model migration) could be eliminated from the critical path entirely.

1.1 [Redacted]’s Core Issue

Enterprise cloud migrations operate under competing constraints: compliance deadlines, cost preservation, operational continuity, and team capability limitations. Migration planning tools typically address one dimension in isolation. Infrastructure mapping tools catalog what exists. Cost estimation tools project future spend. Service equivalency matrices suggest target architectures. None synthesize across organizational boundaries to surface the constraints that determine whether a migration plan is feasible.

The result: technically sound migration plans that ignore financial commitments, understaffed skill gaps, or contractual nuances that determine actual feasibility. A plan that recommends migrating BigQuery to Redshift is incomplete if it doesn't account for

the $180K in committed use discounts that would be forfeited, the team's lack of Redshift experience, or the fact that the customer driving the migration never touches the analytics warehouse.

1.2 Success Criteria

Successful migration planning in this context required:

- Actionable phased plan: A concrete execution roadmap that satisfies the compliance deadline without unnecessary scope expansion.
- Financial preservation: Identification of existing cloud commitments and strategies to avoid sunk cost forfeiture.
- Scope differentiation: Clear separation between workflows on the critical path versus those that can be deferred.
- Risk mitigation: Recommendations informed by operational history, not just theoretical failure modes.
- Executable outputs: Concrete next steps including infrastructure-as-code scaffolding and CLI validation commands.

1.3 What CloudGo.ai Achieved

CloudGo.ai's context-aware analysis delivered the following outcomes:

- Identified that only approximately 20% of the platform required migration to satisfy customer compliance requirements.
- Preserved $180K in GCP committed use discounts by recommending hybrid architecture over full replatforming.
- Eliminated the highest risk workflow (BQML to SageMaker migration) from the critical path entirely.
- Surfaced hidden timeline dependencies (partner webhook IP whitelisting, GDPR processor change notifications) that would have caused deadline slippage.
- Delivered an execution-ready action plan with Terraform module references and CLI commands for immediate validation.

2. Case Context

2.1 Company Profile: [Redacted] Analytics

[Redacted] Analytics is a mid-size B2B data analytics platform serving enterprise customers with real-time dashboards and ML-powered forecasting. Their customer base spans mid-market and enterprise segments, with one customer who will be referred to as Hartfell Partners, representing approximately 35% of annual recurring revenue.

[Redacted]'s technical footprint on Google Cloud Platform included:

1. *Compute:* GKE cluster with 12 nodes running mixed workloads for core application services.
2. *Database:* Cloud SQL (PostgreSQL) for transactional data, with 60% of queries serving read replicas for the dashboard layer.
3. *Analytics:* BigQuery as the primary data warehouse with three years of historical data. Cost breakdown showed approximately $54.8K/month, with 80% attributed to internal analytics workloads and 20% serving customer-facing dashboards.
4. *Machine Learning:* Four BQML models for demand forecasting, retrained weekly via scheduled SQL, with approximately 50,000 inference calls per day. The Architecture Decision Records revealed that BQML was specifically chosen because the team lacked ML Ops expertise; the models run with zero pipeline overhead, a simplicity that would be lost in a SageMaker migration.
5. *Storage:* Cloud Storage with multi-region configuration: approximately 520 TB in US regions and 120 TB in EU regions for GDPR-regulated customer data.
6. *Messaging:* Pub/Sub for event streaming between services; Cloud Functions for webhook processing. Approximately 90% of Pub/Sub traffic was internal service-to-service communication.
7. *External Integrations:* Three data partners pushing to Pub/Sub via IP-whitelisted endpoints.

2.2 The Migration Prompt

Hartfell Partners, the largest customer, mandated that all vendors operate on AWS infrastructure by Q3 to support their FedRAMP authorization pursuit. [Redacted]'s CEO committed to the timeline without engineering consultation. Engineering leadership was particularly concerned about BigQuery-to-Redshift migration complexity and BQML model portability.

The request to CloudGo.ai was framed as follows:

> *We are [Redacted Company], a mid-size B2B analytics platform currently running our entire stack on GCP. Our largest enterprise customer, Hartfell Partners (about 35% of ARR), just informed us they're requiring all vendors to operate on AWS infrastructure by Q3 due to their new security and compliance framework; they're pursuing FedRAMP authorization and want supply chain alignment.*
>
> *We need to migrate from GCP to AWS. I've attached our cloud scan and usage information as well as connected our company documentation. Our CEO committed to the timeline without consulting engineering. We have about 7 months. What's our migration strategy? We're worried about the BigQuery to Redshift migration especially; we have 3 years of historical data and ML models running on BQML.*

2.3 The Generic Response: What Standard Analysis Recommends

Before examining CloudGo.ai's context-aware analysis, it is instructive to understand what a standard migration assessment would recommend based on the prompt alone. Without access to organizational documentation, usage patterns, or financial commitments, the recommendation follows a predictable pattern: full lift-and-shift migration with a seven-month timeline.

A generic assessment would recommend:

- Complete platform replatforming: GKE to EKS, Cloud SQL to RDS, BigQuery to Redshift, Cloud Storage to S3, Pub/Sub to SNS/SQS and Kinesis
- A seven-month migration timeline with all workflows treated as critical path
- BQML to SageMaker migration requiring ML Ops pipeline buildout from scratch
- Full data warehouse migration including three years of historical data

The generic assessment would prescribe a standard seven-month playbook: Month 1 for mobilization and landing zone setup, Month 2 for data plane prototyping and BigQuery export testing, Month 3 for historical backfill at scale, Month 4 for analytics query migration and performance tuning, Months 5-6 for BQML to SageMaker migration requiring ML Ops pipeline buildout, and Month 7 for cutover and stabilization.

This recommendation is technically correct for the stated requirement. It is also organizationally naive, financially costly, and operationally dangerous. The generic approach fails to account for constraints that exist but were not surfaced in the initial prompt: the customer contract that limits what Hartfell has access to, the $180K in committed use discounts that would be forfeited, the team skills gap that makes SageMaker migration high-risk, and the incident history that mandates staged rollouts.

3. Secure, Contextual Data Collection

The prompt above, while detailed, omits the constraints that determine whether a migration plan is pragmatically executable. CloudGo.ai's differentiation lies in its ability to gather and synthesize context that traditional migration planning cannot access. This context comes from two sources: cloud infrastructure connections and company documentation integration.

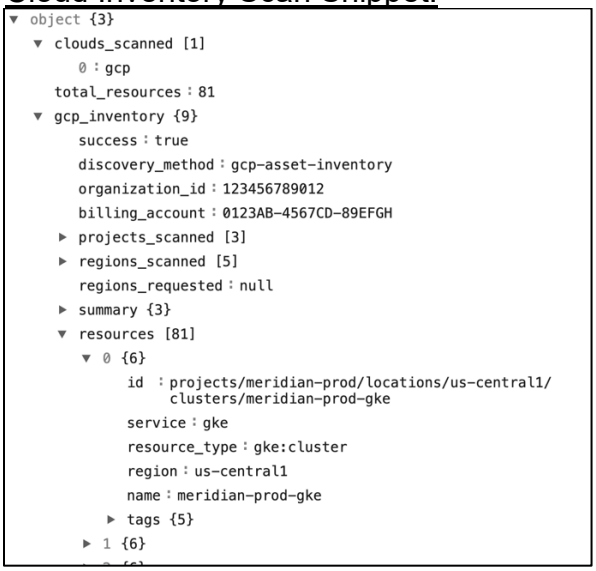
3.1 Cloud Infrastructure Connection

CloudGo.ai connects to customer cloud environments via read-only CLI credentials (GCP gcloud, AWS CLI, Azure CLI). This connection enables infrastructure inventory scans, usage telemetry retrieval, cost and billing data access, and IAM/security posture assessment.

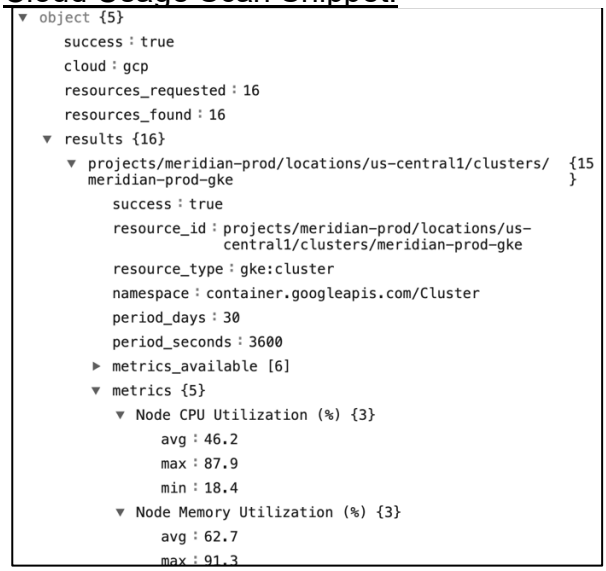
For [Redacted], the GCP scan surfaced the following critical information:

Data Point	Finding
BigQuery Cost Breakdown	~$54.8K/month; 80% internal analytics, 20% customer-facing
BQML Inference Volume	~50K calls/day with low error rates
Storage Footprint	520 TB US-region, 120 TB EU-region
Compute Commitments	$180K remaining on committed use discounts (18 months)

Cloud Inventory Scan Snippet:



Cloud Usage Scan Snippet:

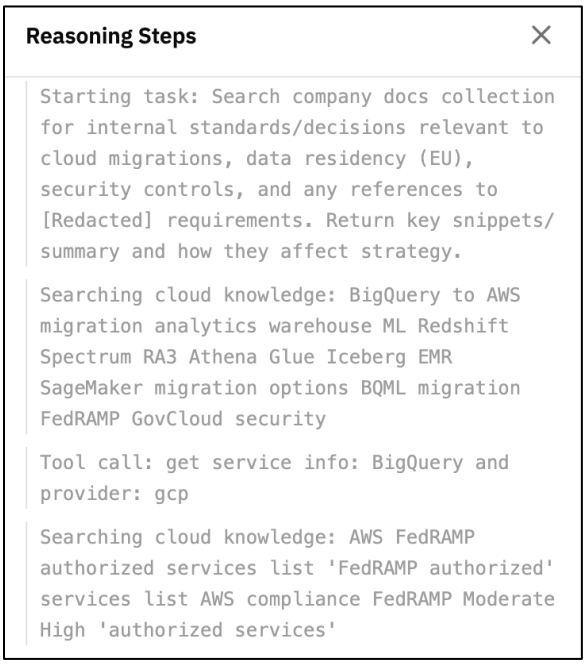


3.2 Company Documentation Integration

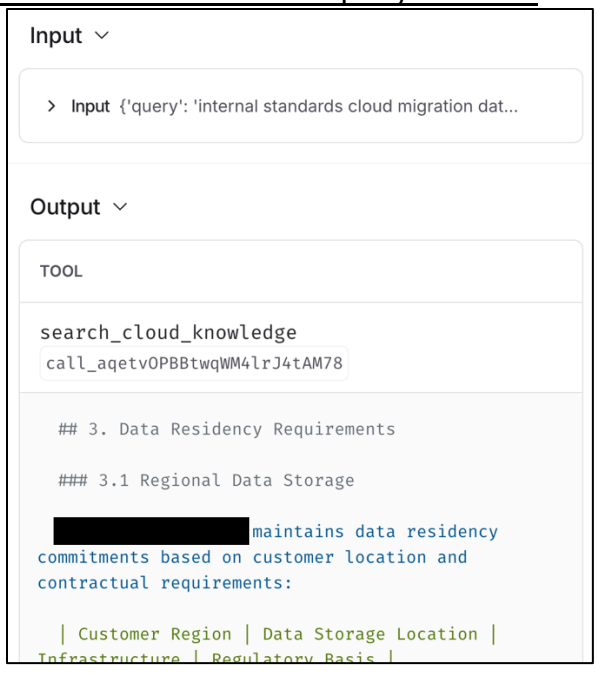
CloudGo.ai ingests customer-provided documentation into a secure, private document store. Documents are indexed for semantic retrieval during analysis. Each customer's documentation exists in a private collection with no cross-tenant data access. For [Redacted], the following documents were ingested and queried:

Document	Key Information Retrieved
Customer Contract (Hartfell)	Hartfell accesses only the reporting API and dashboard frontend; no direct access to analytics pipeline or ML services
GCP Billing Summary	18-month committed use discount with $180K remaining value; early termination forfeits commitment
Architecture Decision Records	BQML selected because team lacked MLOps expertise; retraining runs via scheduled SQL with zero pipeline overhead
Team Skills Assessment	Two engineers hold AWS certifications; no Redshift experience; strong BigQuery/SQL proficiency
Data Governance Policy	EU customer data must remain in EU regions; GDPR DPA specifies GCP as the data processor
Vendor Integration Docs	Three data partners use IP-whitelisted webhooks; endpoint changes require 4-6 week security review per partner
Incident Postmortem (February)	Previous rushed migration caused 3-day outage; postmortem mandated staged rollouts with parallel running

CloudGo.ai Reasoning Steps Snippet:

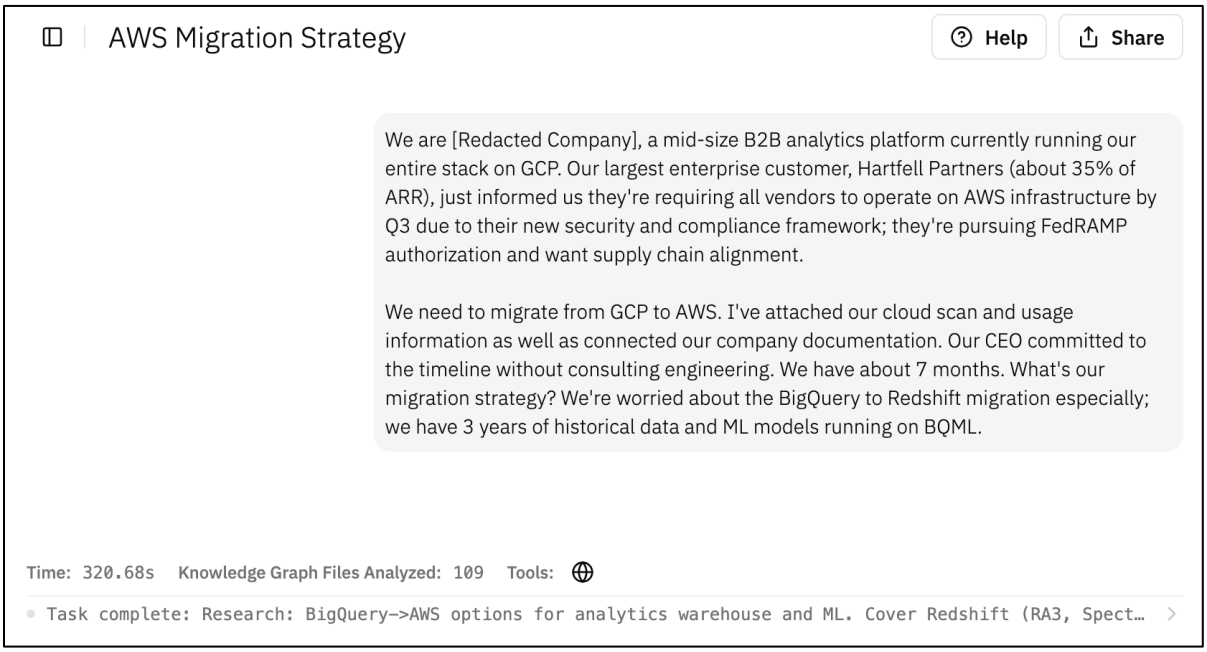


CloudGo.ai Locates Company Context:



4. Synthesis: How CloudGo.ai Reasons Over Multi-Source Context

The same natural language prompt is now executed with the GCP CLI connection active (cloud scan and usage data available), the company documentation collection linked (Hartfell contract, billing records, ADRs, governance policy, incident history), and CloudGo.ai's full tooling stack available for research and execution planning.



4.1 CloudGo.ai's Unique Tooling Stack

CloudGo.ai brings multiple resources to bear on the augmented prompt: our curated cloud knowledge base contains curated documentation for AWS, Azure, and GCP including Well-Architected Frameworks, service FAQs and pricing documentation, migration whitepapers, and Cloud Adoption Framework patterns. Semantic search enables retrieval of relevant guidance based on the specific migration pattern.

CloudGo.ai also executes read-only CLI commands against connected cloud environments, retrieving real-time infrastructure state, usage metrics, and cost data. This validates assumptions in the migration plan against actual resource configurations. We further maintain a searchable index of Terraform provider documentation and module configurations. This enables generation of infrastructure-as-code scaffolding for target architecture, with integrated validation using industry-standard security tools including Checkov, TFLint, and OPA/Conftest.

4.2 Synthesis in Action: CloudGo.ai’s Reasoning Process

CloudGo.ai recognizes this as a multi-cloud migration requiring both research and infrastructure analysis and begins by scanning the current GCP environment. The

infrastructure scan returns the full inventory: GKE cluster configuration, Cloud SQL instances, BigQuery datasets with storage and query volumes, GCS buckets with regional distribution, and Pub/Sub topics with subscriber counts. A follow-up query retrieves usage telemetry: BigQuery cost attribution (80% internal, 20% customer-facing), BQML inference volumes, and the critical discovery of $180K in unused committed use discounts.

After querying the company documentation collection, the search surfaces the Hartfell contract (revealing limited service scope), the incident postmortem (recommending staged migrations), and the vendor integration guide (exposing the 4-6 week webhook lead time). CloudGo.ai then queries its cloud knowledge base for GCP-to-AWS migration patterns, retrieving AWS prescriptive guidance on BigQuery-to-Redshift migration, S3/Athena lakehouse patterns, and FedRAMP compliance considerations.

The findings are then synthesized across all sources, identifying that Hartfell’s requirement can be satisfied by migrating only the customer-facing services, while the analytics platform (BigQuery, BQML) can remain on GCP without violating the compliance mandate. This synthesis required connecting multiple data points that no single source contained: the Hartfell contract revealed limited service scope, the API usage logs confirmed this pattern quantitatively, the GCP billing data showed the financial cost of premature migration, the team skills assessment revealed Redshift as an unfamiliar technology, and the incident postmortem provided organizational context for why staged approaches were necessary.

The key insight emerged from the intersection of contractual, financial, and operational data. Hartfell's FedRAMP pursuit required supply chain alignment, but their actual service consumption was limited to the reporting API and dashboard frontend. These services could be migrated to AWS within weeks, satisfying the compliance requirement while preserving the 80% of infrastructure that Hartfell never touches. The $180K in committed use discounts would remain intact. The team could gain AWS operational experience on lower-risk workloads before attempting a warehouse migration. The February incident postmortem's recommendation for staged rollouts would be honored.

5. CloudGo.ai's Comprehensive Action Plan

5.1 Strategic Recommendation

CloudGo.ai's synthesized recommendation diverged sharply from the generic assessment. The core insight: [Redacted] should not attempt full replatforming by Q3. Instead, the strategy called for migrating only the Hartfell-facing surface area (reporting API, dashboard frontend) to AWS while maintaining BigQuery and BQML on GCP through the deadline. These systems are not accessed by Hartfell, and migration would be high-effort and high-risk with no compliance benefit.

You should **not attempt a "big bang" full-platform replatform to AWS by Q3**. The winning strategy is:

1. **Meet The Cleint's "AWS infrastructure" requirement with a customer-specific AWS boundary first** (their externally accessed endpoints + identity + logging/audit trail + network controls), while
2. **Keeping your highest-risk / highest-effort systems (BigQuery warehouse + BQML) running on GCP during the deadline**, and
3. **Running a controlled, phased data-plane migration afterward** (or in parallel only where it's safe), so you don't blow up reliability for the rest of your customer base.

This aligns with your own internal docs: Hartfell traffic is concentrated on a small set of externally accessed endpoints, and they **don't have EU residency requirements**; internal docs also explicitly recommend migrating the Reporting API + Dashboard first and *not* making BigQuery the critical path for Hartfell.

What your scan/usage says (why BigQuery shouldn't be the critical path)

From your usage export:

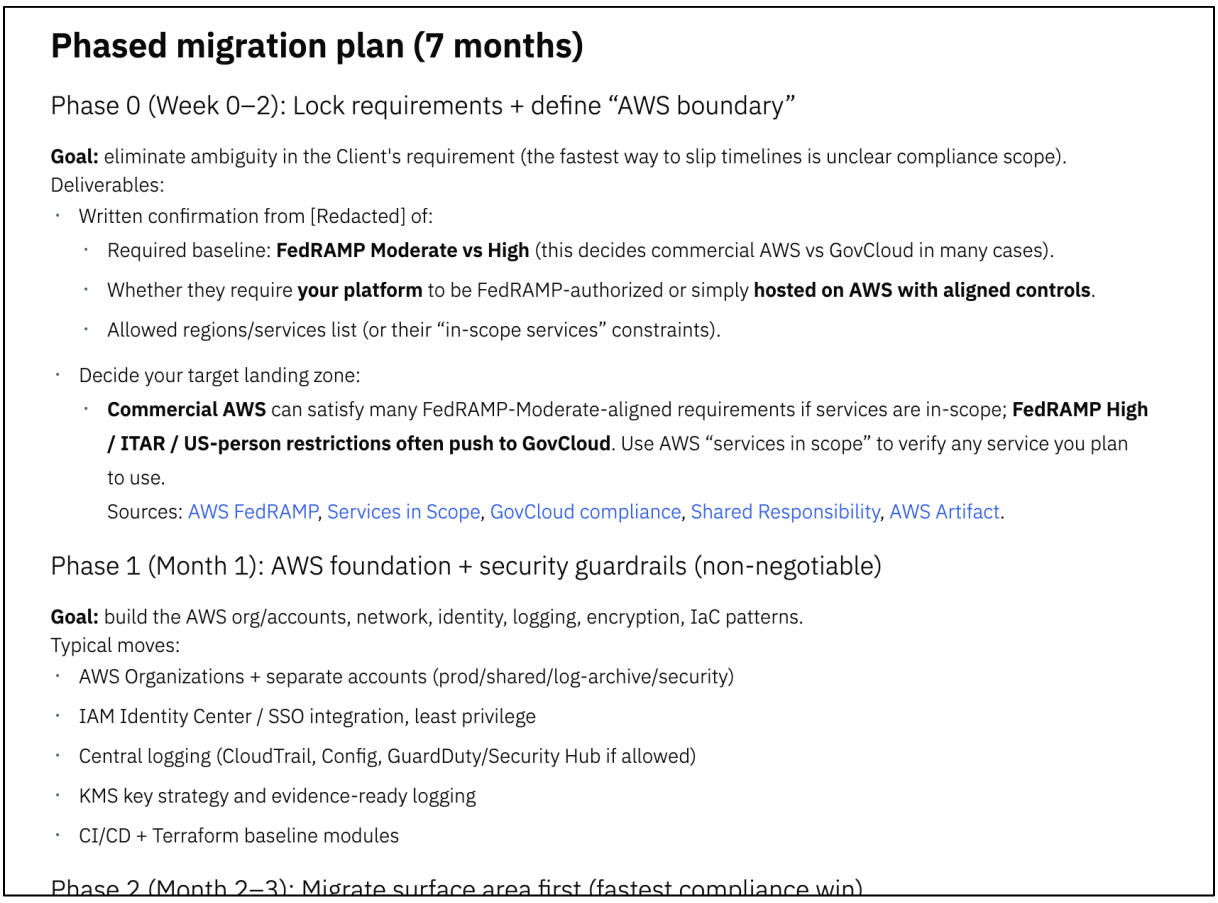
- **BigQuery DW cost:** ~$54.8k/mo; **~80% internal analytics**, **~20% customer dashboards**.
  → Optimization + selective migration yields more value than rushing a full Redshift rebuild.
- **BQML models:** weekly retrain; **~50k inference calls/day** with low error rate.
  → Rebuilding MLOps during an already-compressed migration is pure risk.
- **GCS raw ingest is massive:** ~**520 TB** US + **120 TB** EU.
  → Moving the lake + re-cataloging + governance is months by itself.
- **Externally sensitive integrations:** partner webhooks + allowlists; docs say **4–6 weeks security review per partner** and recommend **dual endpoints** during transitions.

So: **deadline success = migrate the externally visible/control-plane surface first**, then tackle the warehouse with a deliberate path.

The recommendation called for establishing hybrid connectivity via Cloud Interconnect and Direct Connect for backend communication during transition, then revisiting full migration in 18 months when GCP commitments expire and the team has accumulated AWS operational experience.

5.2 Phased Execution Plan

CloudGo.ai delivered a structured phased approach spanning the seven-month timeline while compressing the critical compliance path to approximately eight weeks.



Phased migration plan (7 months)

Phase 0 (Week 0–2): Lock requirements + define "AWS boundary"

Goal: eliminate ambiguity in the Client's requirement (the fastest way to slip timelines is unclear compliance scope).
Deliverables:

- Written confirmation from [Redacted] of:
  - Required baseline: **FedRAMP Moderate vs High** (this decides commercial AWS vs GovCloud in many cases).
  - Whether they require **your platform** to be FedRAMP-authorized or simply **hosted on AWS with aligned controls**.
  - Allowed regions/services list (or their "in-scope services" constraints).
- Decide your target landing zone:
  - **Commercial AWS** can satisfy many FedRAMP-Moderate-aligned requirements if services are in-scope; **FedRAMP High / ITAR / US-person restrictions often push to GovCloud**. Use AWS "services in scope" to verify any service you plan to use.
    Sources: AWS FedRAMP, Services in Scope, GovCloud compliance, Shared Responsibility, AWS Artifact.

Phase 1 (Month 1): AWS foundation + security guardrails (non-negotiable)

Goal: build the AWS org/accounts, network, identity, logging, encryption, IaC patterns.
Typical moves:

- AWS Organizations + separate accounts (prod/shared/log-archive/security)
- IAM Identity Center / SSO integration, least privilege
- Central logging (CloudTrail, Config, GuardDuty/Security Hub if allowed)
- KMS key strategy and evidence-ready logging
- CI/CD + Terraform baseline modules

Phase 2 (Month 2–3): Migrate surface area first (fastest compliance win)

Phase 0 (Weeks 1-2): Lock Hartfell requirements and define the AWS boundary scope. Obtain written confirmation from Hartfell regarding FedRAMP Moderate versus High requirements, whether they require [Redacted]'s platform to be FedRAMP-authorized or simply hosted on AWS with aligned controls (and allowed regions and services). This definition drives subsequent architectural decisions: commercial AWS can satisfy many FedRAMP-Moderate-aligned requirements if services are in-scope, while FedRAMP High or ITAR restrictions often push toward GovCloud.

Phase 1 (Month 1): AWS landing zone and security guardrails. Build the AWS organization and accounts, network, identity, logging, encryption, and IaC patterns. This includes multi-account strategy with prod, staging, shared services, and security/logging accounts, IAM Identity Center for least-privilege access, central logging via CloudTrail organization trail, and KMS key strategy with evidence-ready logging for compliance.

Phase 2 (Months 2-3): Migrate Hartfell-facing services. Stand up AWS ingress (ALB/WAF), route Hartfell traffic first via IP allowlists, move reporting and dashboard services to AWS compute using ECS/Fargate or EKS. Deploy dual endpoints for partner

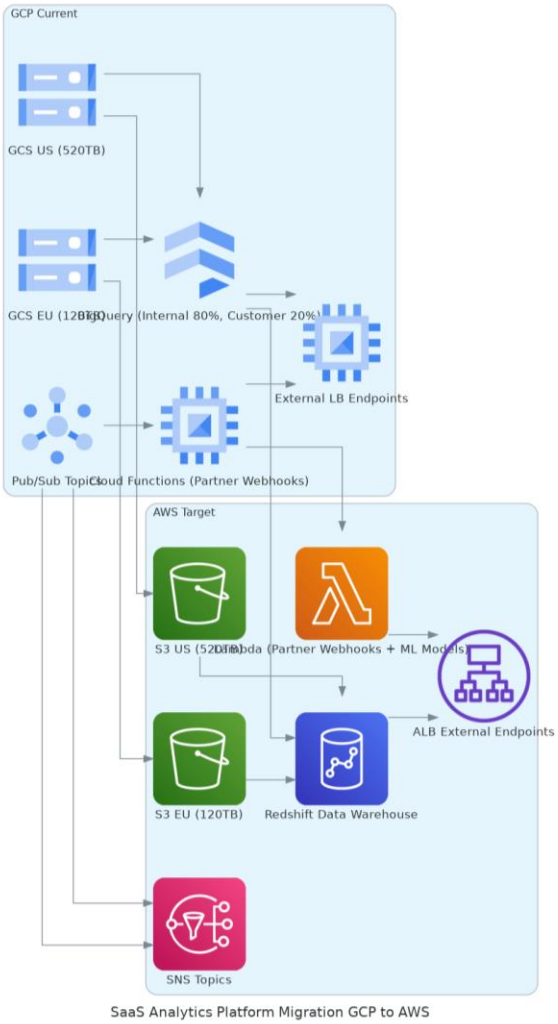
webhooks until allowlists are updated, coordinating the 4-6 week security review per partner that the vendor integration documentation requires. This is the critical path for Hartfell compliance.

Phase 3 (Months 3-5): Data plane bridge. Stand up S3 and Glue Data Catalog; start landing new raw data to S3 via dual-write. Optionally expose an AWS-queryable view of data using Athena/Iceberg for incremental workloads. Keep BigQuery as the primary warehouse.

Phase 4 (Months 4-6): Optional warehouse migration track. Given time constraints, CloudGo.ai recommended the lakehouse-first approach (Athena/Glue/Iceberg with targeted Redshift for high-concurrency dashboard marts) over full Redshift migration.

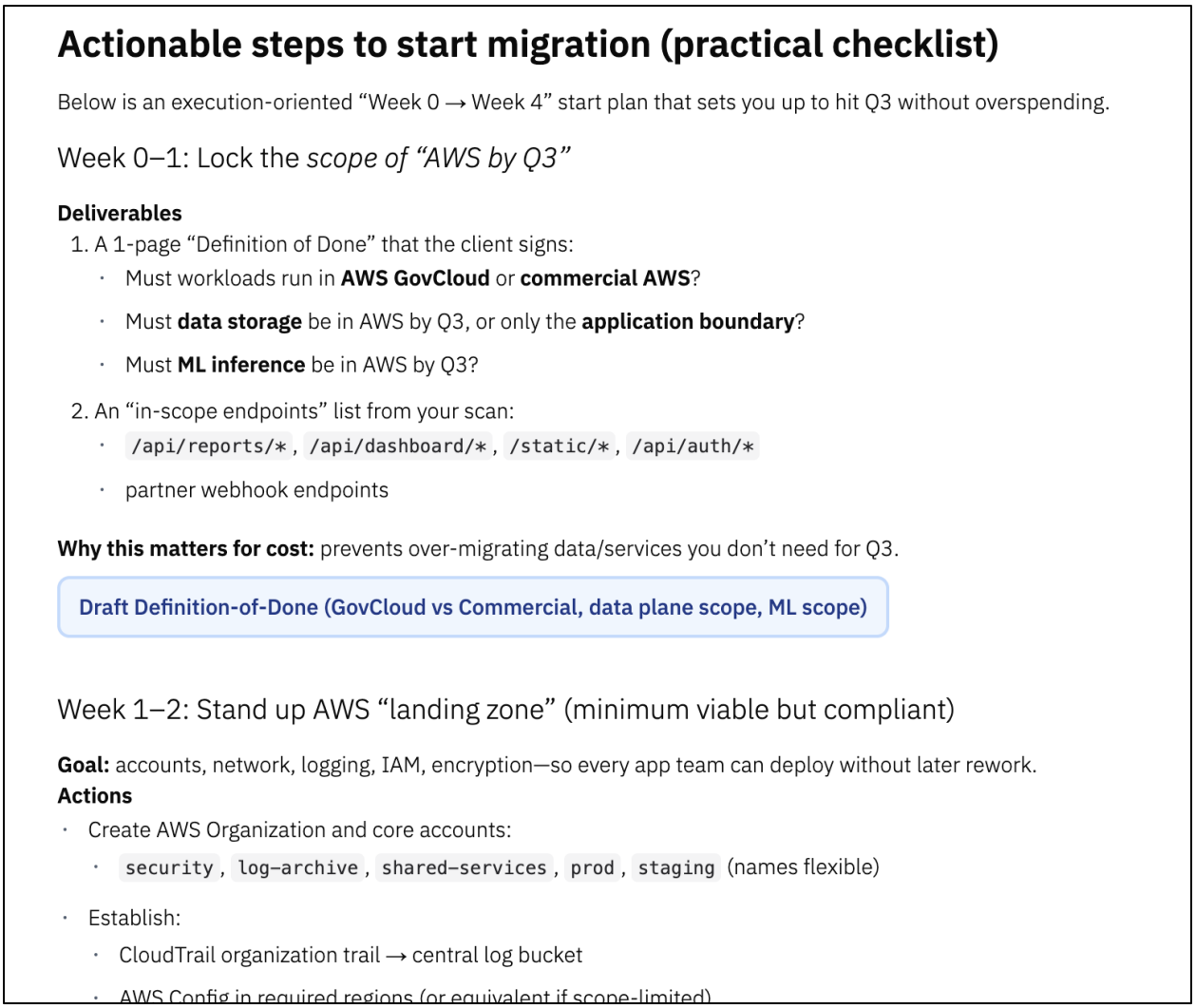
Phase 5 (Months 6-7): Cutover and stabilization. Cut over Hartfell traffic fully to AWS endpoints; maintain rollback capability for at least one full reporting cycle.

Hybrid architecture diagram generated by CloudGo.ai:



5.3 Actionable Outputs: From Strategy to Execution

CloudGo.ai delivered concrete next steps that [Redacted]'s engineering team could execute immediately. Beyond the phased migration plan, CloudGo.ai provided detailed cost analysis, infrastructure blueprints, and security guardrails—transforming strategic recommendations into implementation-ready artifacts.



Actionable steps to start migration (practical checklist)

Below is an execution-oriented "Week 0 → Week 4" start plan that sets you up to hit Q3 without overspending.

Week 0–1: Lock the *scope of "AWS by Q3"*

Deliverables

1. A 1-page "Definition of Done" that the client signs:
   - Must workloads run in **AWS GovCloud** or **commercial AWS**?
   - Must **data storage** be in AWS by Q3, or only the **application boundary**?
   - Must **ML inference** be in AWS by Q3?
2. An "in-scope endpoints" list from your scan:
   - `/api/reports/*`, `/api/dashboard/*`, `/static/*`, `/api/auth/*`
   - partner webhook endpoints

Why this matters for cost: prevents over-migrating data/services you don't need for Q3.

Draft Definition-of-Done (GovCloud vs Commercial, data plane scope, ML scope)

Week 1–2: Stand up AWS "landing zone" (minimum viable but compliant)

Goal: accounts, network, logging, IAM, encryption—so every app team can deploy without later rework.
Actions

- Create AWS Organization and core accounts:
  - `security`, `log-archive`, `shared-services`, `prod`, `staging` (names flexible)
- Establish:
  - CloudTrail organization trail → central log bucket
  - AWS Config in required regions (or equivalent if scope-limited)

Cost-Conscious Migration Analysis

CloudGo.ai's plan explicitly addressed cost optimization across six dimensions. First, it avoided a "double-spend cliff" on the data warehouse by migrating only the 20% of customer-critical workloads rather than duplicating the entire BigQuery environment in Redshift. Second, it preserved the $180K in sunk GCP commitments by keeping compute-heavy workloads running to burn down the commitment while AWS

infrastructure was established. Third, it minimized egress and bulk-transfer costs by deferring the 640 TB raw lake migration until after schema and query semantics were validated on AWS. Fourth, it avoided building new MLOps infrastructure under time pressure by keeping BQML operational until the platform boundary stabilized. Fifth, it reduced “rush tax” rework by establishing security and compliance guardrails in the AWS landing zone before migrating workloads. Sixth, it protected against incident-driven costs by maintaining dual endpoints during partner integration cutover.

Sample Snippet: AWS Landing Zone Blueprint

CloudGo.ai generated a minimum-viable, FedRAMP-aligned landing zone design. The blueprint specified an AWS Organizations structure with Security, Platform, Workloads, and Sandbox organizational units. The Security OU contained dedicated accounts for log archival (immutable CloudTrail/Config storage) and security tooling (delegated admin for Security Hub, GuardDuty, and Config aggregation). The Platform OU housed network infrastructure (Transit Gateway, shared endpoints, egress controls) and shared services (DNS, CI/CD, artifact repositories). The Workloads OU separated production and staging environments with mirrored guardrails. The blueprint included IAM Identity Center integration for workforce identity, centralized CloudTrail organization trails with SSE-KMS encryption, AWS Config aggregation for compliance visibility, and VPC baseline patterns with cost-conscious NAT Gateway placement.

OU / Account Map ([Redacted] AWS Landing Zone)

Below is a concrete, implementable **AWS Organizations** layout plus **starter SCP guardrails**. It’s intentionally minimal (so you can move fast) but locks the big foot-guns: regions, logging, and public S3 exposure.

Recommended OUs and accounts

OU	Account	Purpose	Notes
Security	`log-archive`	Central immutable storage for CloudTrail/Config logs	No workloads. Tight access.
Security	`security-tooling`	Security services delegated admin (Security Hub, GuardDuty), Config aggregator, IAM Access Analyzer	Good place for audit dashboards.
Platform	`network`	Central networking primitives (TGW, shared endpoints, egress inspection)	Optional if you’re small at first, but recommended.
Platform	`shared-services`	DNS/Route53 zones, shared CI/CD, container registry, artifacts	Keep minimal initially.
Workloads	`prod`	All production application + data plane	Consider splitting later (`prod-app` , `prod-data`) if needed.
Workloads	`staging`	Pre-prod, performance testing, integration	Mirror prod guardrails.
Sandbox	`sandbox`	Developer experimentation	Strong SCP restrictions + budget alarms.

Sample Snippet: Security Control Policies

CloudGo.ai provided production-ready Service Control Policy (SCP) JSON for three critical guardrails. The first SCP restricted resource creation to approved AWS regions (supporting either commercial AWS or GovCloud depending on Hartfell's final FedRAMP requirements), preventing cost sprawl and compliance drift. The second SCP protected audit infrastructure by denying actions that would disable CloudTrail or AWS Config, maintaining the evidence trail required for FedRAMP authorization. The third SCP enforced S3 public access blocks and denied bucket policies that could expose data publicly. CloudGo.ai recommended a staged rollout: attach log protection to the organization root immediately, then roll region restrictions through Sandbox, Staging, and Production sequentially after validating global-service exceptions.

SCP 1 — Allowed Regions (deny everything except your approved regions)

What it does: prevents teams from creating resources outside your allowed AWS regions (critical for compliance + cost sprawl control).
How it works: deny actions when `aws:RequestedRegion` is not in an allowlist. AWS provides this pattern as an example in the Organizations docs (SCP examples).
Customize this list based on Hartfell requirements:

- Commercial AWS typical: `us-east-1`, `us-west-2` (and maybe `us-east-2`)
- GovCloud typical: `us-gov-west-1`, `us-gov-east-1`

> *Replace the region list below with your chosen set.*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAllOutsideApprovedRegions",
      "Effect": "Deny",
      "NotAction": [
        "iam:*",
        "organizations:*",
        "route53:*",
        "cloudfront:*",
        "support:*",
        "budgets:*",
        "ce:*",
        "cur:*",
        "sts:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:RequestedRegion": [
            "us-east-1",
            "us-west-2"
          ]
        }
      }
```

5.4 Intercloud Cost Considerations

CloudGo.ai also provided detailed egress cost projections for the GCP-to-AWS data transfer. For bulk migration traffic, GCP egress is the primary cost driver; AWS internet ingress is typically $0/GB. CloudGo.ai generated cost range tables based on [Redacted]'s source regions and transfer volumes, enabling precise budgeting for each migration phase.

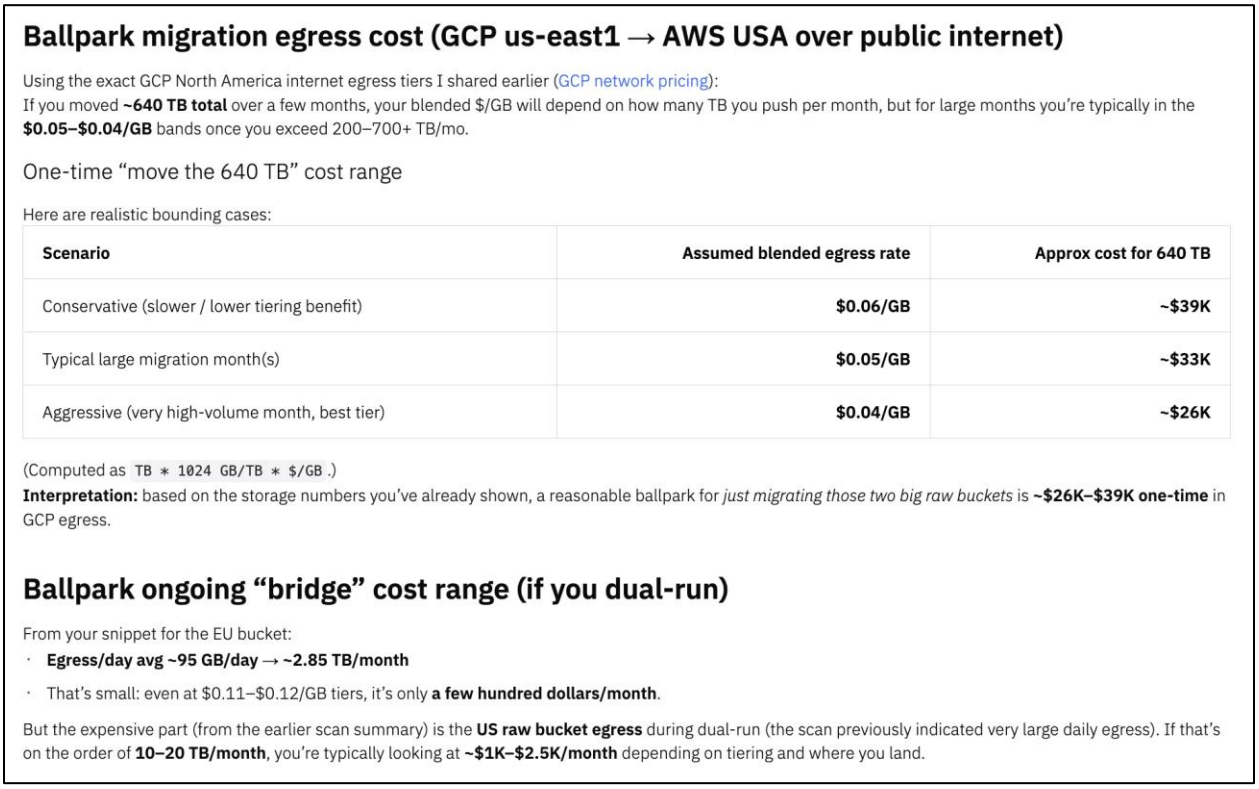
Cost range table (public internet transfer)

These are **planning ranges** that match how GCP internet egress is typically priced by geography/tier (confirm exact rows for your source regions on the pricing page). AWS ingress is included for completeness and is generally $0.

Flow / scenario	Primary billed side	Unit cost range (USD / GB)	Cost per 1 TB (USD)	Notes	Sources
GCP → AWS (same-continent) e.g., GCP US → AWS US	GCP egress	**$0.08–$0.12/GB**	**$80–$120/TB**	Common case if you keep US workloads in US. Exact rate depends on GCP region + tiering.	GCP Internet egress
GCP → AWS (intercontinental) e.g., GCP EU → AWS US (or vice versa)	GCP egress	**$0.12–$0.20/GB**	**$120–$200/TB**	Intercontinental moves are meaningfully more expensive; avoid this with EU-in-EU and US-in-US cutovers.	GCP Internet egress
GCP APAC → AWS (regional/APAC)	GCP egress	**$0.12–$0.19/GB**	**$120–$190/TB**	APAC rates often trend higher than US/EU. Validate per source region.	GCP Internet egress
GCP → AWS (Australia / South America)	GCP egress	**$0.18–$0.28/GB**	**$180–$280/TB**	Higher-cost geographies; plan careful staging and minimize backfills.	GCP Internet egress
Internet → AWS (ingress)	AWS	**$0.00/GB** (typical)	**$0/TB**	AWS generally does not charge for data transferred *into* AWS from the internet.	AWS key principles, AWS CUR transfers

The egress cost analysis revealed significant variation by geography: same-continent transfers (GCP US to AWS US) cost $80–$120 per TB, while intercontinental transfers could reach $120–$200 per TB. Higher-cost regions like Australia and South America ranged from $180–$280 per TB. This analysis drove a key recommendation: keep EU data flowing to AWS EU regions and US data to AWS US regions, avoiding the intercontinental premium entirely.

Applying these rates to [Redacted]'s actual storage footprint, CloudGo.ai generated concrete cost bounds for both bulk migration and ongoing dual-run scenarios:



Ballpark migration egress cost (GCP us-east1 → AWS USA over public internet)

Using the exact GCP North America internet egress tiers I shared earlier (GCP network pricing):
If you moved **~640 TB total** over a few months, your blended $/GB will depend on how many TB you push per month, but for large months you're typically in the **$0.05–$0.04/GB** bands once you exceed 200–700+ TB/mo.

One-time "move the 640 TB" cost range

Here are realistic bounding cases:

Scenario	Assumed blended egress rate	Approx cost for 640 TB
Conservative (slower / lower tiering benefit)	**$0.06/GB**	**~$39K**
Typical large migration month(s)	**$0.05/GB**	**~$33K**
Aggressive (very high-volume month, best tier)	**$0.04/GB**	**~$26K**

(Computed as `TB * 1024 GB/TB * $/GB`.)
Interpretation: based on the storage numbers you've already shown, a reasonable ballpark for *just migrating those two big raw buckets* is **~$26K–$39K one-time** in GCP egress.

Ballpark ongoing "bridge" cost range (if you dual-run)

From your snippet for the EU bucket:

- **Egress/day avg ~95 GB/day → ~2.85 TB/month**
- That's small: even at $0.11–$0.12/GB tiers, it's only **a few hundred dollars/month**.

But the expensive part (from the earlier scan summary) is the **US raw bucket egress** during dual-run (the scan previously indicated very large daily egress). If that's on the order of **10–20 TB/month**, you're typically looking at **~$1K–$2.5K/month** depending on tiering and where you land.

The EU raw bucket showed average egress of approximately 95 GB/day (~2.85 TB/month); small enough that even at premium tier pricing, the cost remained in the hundreds of dollars monthly. The US raw bucket, with significantly higher daily egress volumes in the 10–20 TB/month range during active dual-run periods, projected to $1,000–$2,500/month depending on tiering. These bridge costs reinforced the recommendation to minimize the dual-run window through careful phase sequencing rather than indefinite parallel operation.

CloudGo.ai also mapped these costs to migration phases. Phases 1–2 (landing zone and Hartfell endpoints) would incur minimal transfer costs—primarily application artifacts and small database snapshots in the low terabyte range. Phase 3 (data-plane bridge with dual-write to S3) would generate medium transfer volumes as customer-facing marts were exported incrementally. Phase 4 (warehouse migration track) represented the highest cost sensitivity, potentially moving tens to hundreds of terabytes monthly. Phase 5 (historical backfill) could reach petabyte-scale transfers if not carefully staged.

The analysis recommended explicit monthly egress budget caps per phase: Phase 3 capped at 20 TB/month (approximately $1,600–$2,400 in egress costs), Phase 4 at 100

TB/month unless business requirements demanded more, and Phase 5 allocated as a controlled one-time migration window. By measuring actual export sizes before committing to transfers—pulling BigQuery export sizes and GCS object inventory by prefix—[Redacted] could pay egress only on data that truly needed to move, avoiding the common mistake of transferring data that would later require re-transfer after schema changes.

How these costs relate to your migration phases

Below maps cost drivers to the phases we discussed (Month 1–7). I'm tying it to the plan's key idea: **migrate the app boundary first, then "marts-first" data, then bulk history later**.

Phase (from plan)	What data moves GCP → AWS	Typical monthly transfer magnitude	Expected cost sensitivity	How to keep it cost-conscious
Phase 1–2 (Month 1–3): Landing zone + edge + Thornfield endpoints	Mostly **application artifacts** (images, configs), small DB snapshots, limited logs	**Low (GBs → low TBs)**	Low	Keep services chatty within one cloud (don't create cross-cloud hot paths); move only what Thornfield needs.
Phase 3 (Month 3–5): Data-plane bridge (dual-write landing to S3)	Incremental exports / CDC of **customer-facing marts** and new raw landing	**Medium (TBs → tens of TBs/ month)**	Medium	Export only the marts powering `/api/reports/*` and `/api/dashboard/*` first; avoid bulk historical backfill yet.
Phase 4 (Month 4–6): Warehouse migration track	Larger curated datasets, dimension tables, and some history needed for dashboards	**Medium–High (tens → hundreds TB/ month)**	High	Prefer "query-in-place" (S3/Athena/ Spectrum) for cold data; materialize only hot tables into Redshift. Regionalize (EU→EU, US→US) to avoid intercontinental rates.
Phase 5 (Month 6–7): Cutover + backfill/ decommission planning	Potentially **largest wave**: multi-year history, archival lake copies	**High (hundreds TB → PB-scale total)**	Very High	Schedule backfills after schema/query parity; compress/partition; run in controlled windows; avoid copying data twice; consider private connectivity if sustained.

The combination of cost analysis, infrastructure blueprints, security policies, and egress budgeting transformed CloudGo.ai's strategic recommendation into an execution-ready package. [Redacted]'s engineering team could begin implementation immediately: the landing zone blueprint provided the AWS foundation, the SCPs established compliance guardrails from day one, the egress budget caps prevented runaway transfer costs, and the cost-conscious sequencing ensured the $180K GCP commitment would be preserved while AWS operational experience accumulated.

6. Conclusion

6.1 The Transformation in Cloud Recommendations

The migration recommendation changed fundamentally when organizational context was incorporated. The following table summarizes the contrast:

Without Context	With Context
Full migration to AWS	Partial migration; hybrid architecture
BigQuery → Redshift (6-month effort)	Keep BigQuery; Hartfell doesn't use it
BQML → SageMaker (MLOps buildout)	Defer; team lacks expertise and it's not customer-facing
7-month timeline for everything	8-week timeline for Hartfell; 18-month optional full migration
Forfeit $180K GCP commitment	Preserve commitment; migrate when it expires
Single-cloud target state	Intentional hybrid as the right architecture

Cloud migrations fail when planning operates in isolation from financial, contractual, and operational reality. The information required for sound migration decisions exists within every organization. It resides in billing systems, contract repositories, incident databases, and the tribal knowledge of engineers who have lived through previous platform changes. The challenge is not generating migration recommendations. The challenge is gathering and synthesizing the context that makes recommendations executable.

CloudGo.ai's contribution is not superior reasoning over the same inputs; it is the ability to gather and synthesize inputs that traditional planning tools cannot access. The Hartfell contract was not attached to the migration prompt, the $180K commitment was not mentioned, the February incident postmortem was not referenced, but all or any of the three determined whether the generic migration plan would succeed or fail.

No single document or data source contained the optimal recommendation. The insight that Hartfell only uses two services required both the customer contract and API usage logs. The insight that BigQuery should not be migrated required the GCP billing data, the team skills assessment, and understanding of BQML operational complexity. The insight that an 18-month hybrid approach was financially optimal required committed use discount terms combined with migration cost modeling. The insight that partner webhooks needed six-week lead time came from the vendor integration documentation.

The insight that staged migrations were organizationally mandated came from the February incident postmortem.

This pattern of distributed, cross-functional knowledge is not unique to [Redacted]. Every organization accumulates constraints across billing systems, contract repositories, incident databases, architecture decision records, and the institutional memory of engineers who have navigated previous migrations. These constraints determine whether a migration succeeds or fails, yet they rarely surface in planning sessions because they exist in different systems, owned by different teams, documented in different formats.

6.2 A Different Kind of Migration Planning

A generic LLM, given the same prompt, would produce technically correct but organizationally naive guidance: migrate to Redshift, use AWS DMS, here is a six-month timeline. CloudGo.ai connects the dots and produces actionable guidance: you do not need to migrate BigQuery at all because Hartfell never touches it, you would forfeit $180K in commitments, and your team would struggle with SageMaker. Migrate only what you must, run hybrid for 18 months, and revisit when the math changes.

For teams facing multi-cloud migrations, compliance deadlines, or infrastructure optimization decisions, the difference between generic and context-aware guidance is the difference between a plan that looks correct and a plan that genuinely works.

The [Redacted] case illustrates a broader truth about cloud strategy: the constraint that determines success or failure is rarely technical. It is financial (the $180K commitment), contractual (what Hartfell has access to), organizational (the team's skill gaps), or historical (the lessons encoded in past incident postmortems). Tools that address only the technical dimension produce recommendations that are correct in the abstract but fail in the concrete reality of organizational execution.

Context-aware migration planning does not require omniscience. It requires the ability to gather information from where it lives, synthesize across organizational boundaries, and reason about constraints that traditional planning tools ignore. The result is not a different answer to the same question, but a different question entirely: not 'how do we migrate everything to AWS in seven months?' but 'what is the minimum migration that satisfies our compliance requirement while preserving our financial commitments and avoiding our highest risk workflows?' The answer to that question is actionable. The answer to the first question is a plan that looks impressive in a slide deck and fails in production.