CloudGo.ai

Case Study: Token Efficiency and Context Accuracy

How Proprietary Cloud Intelligence Reduces Cost
and Improves Output Quality for Agentic Cloud Workflows

March 2026
cloudgo.ai

Cloud infrastructure is growing in scale and complexity. As organizations adopt multi-cloud architectures spanning AWS, GCP, and Azure, the task of scanning, analyzing, and planning migrations has become a serious challenge for AI-assisted workflows. Leading foundation models like Claude and ChatGPT are remarkably capable, but when tasked with large-scale cloud inventory analysis, they encounter two compounding problems: rapidly inflating token counts and progressive context decay.

CloudGo.ai was purpose-built to solve both of these problems. By layering proprietary cloud intelligence databases, pre-computed Terraform knowledge graphs, and embedded company documentation on top of a multi-agent LangGraph architecture, CloudGo.ai dramatically reduces the number of tokens a model needs to consume and produce in order to reach a defensible cloud migration plan. It does this while virtually eliminating context decay, the gradual erosion of accuracy that affects models as their input context grows beyond manageable thresholds.

In controlled testing across five independent cloud environments, CloudGo.ai used 46% fewer tokens on average than Claude Opus 4.6 and 27% fewer tokens than ChatGPT 5.4 to reach comparable migration plans. Context decay for CloudGo.ai averaged just 2.0%, compared to 13.4% for Claude and 16.2% for ChatGPT. CloudGo.ai completed every test in a single turn, while Claude averaged 8.8 turns and ChatGPT averaged 4.4 turns. CloudGo.ai also achieved a signal-to-noise ratio of 89% on average, meaning the vast majority of its output was directly actionable planning content rather than exploratory noise. These efficiency gains translate directly into lower API costs, faster turnaround, and higher-confidence outputs for teams managing real production infrastructure.

1) Leading Models Still Struggle with Cloud

Foundation models excel at general reasoning, code generation, and natural language tasks. However, cloud infrastructure analysis introduces a unique combination of challenges that stress-tests their limitations.

Token Inflation

A typical multi-cloud deployment with 80+ resources across compute, storage, networking, IAM, and database services generates thousands of lines of raw API output. When a model like Claude or ChatGPT is asked to scan these environments, it must ingest the full inventory as context, reason about resource relationships, cross-reference provider-specific best practices, and produce a structured plan. Every step adds tokens. As complexity grows, so does the total token footprint, and with it, the cost per interaction.

For an enterprise deployment with 130+ resources and Terraform state, our testing showed that standalone models consumed between 500,000 and 790,000 tokens to produce a migration plan. At current API pricing, that single interaction can cost $10 to $20 or more, and production teams may need to run dozens of variations before reaching a final plan.

Context Decay

Context decay is the progressive loss of accuracy and completeness that occurs as a model's input grows within a single session. It is not a binary failure. Instead, as context windows fill, models begin to omit resources they scanned earlier, conflate provider-specific configurations,

or lose track of interdependencies between services. In our testing, both Claude and ChatGPT showed measurable accuracy degradation once their working context grew beyond approximately 200,000 tokens within a run, with the effect worsening as complexity and token volume increased.

This is especially problematic for cloud planning, where a missed security group, an overlooked cross-region dependency, or a conflated IAM policy can produce a plan that looks correct on the surface but fails in execution. The cost of fixing those errors downstream far exceeds the cost of the tokens themselves.

2) Testing Methodology

We designed a controlled benchmarking framework to compare three configurations: Claude Opus 4.6 (standalone, with tool access to cloud APIs), ChatGPT 5.4 (standalone, with equivalent tool access), and CloudGo.ai (running on Claude Sonnet 4.6, augmented by CloudGo.ai's proprietary infrastructure layer). Each configuration was tested against five independent cloud environments.

Environment Profiles

The five test environments were designed to represent a range of real-world cloud deployments. They included a single-cloud AWS environment with approximately 20 resources, a dual-cloud setup across AWS and GCP with roughly 45 resources, a three-cloud environment spanning AWS, GCP, and Azure with about 80 resources, an enterprise environment with 130+ resources and active Terraform state files, and a full-scale migration scenario with approximately 200 resources, compliance requirements, and cross-cloud interdependencies. The runs were conducted independently and in no particular order; results for each run reflect the performance of a fresh session against that specific environment.

Evaluation Criteria

Each run was evaluated for comparability of the final migration plan. Plans were considered comparable when they identified essentially the same set of cloud resources, presented accurate provider-specific configurations, and proposed a migration sequence that was defensibly logical. We measured four metrics: total token count (input plus output), context decay (the percentage of scanned resources that were omitted, inaccurately described, or lost between scanning and final plan output), turn count (the number of model interaction cycles required to reach the final plan), and signal-to-noise ratio (the percentage of model output that constituted directly actionable planning content versus exploratory steps, repeated information, or hallucinated details).

3) Results

Token Efficiency and Context Decay

Run	Environment	Opus 4.6 Tokens	GPT 5.4 Tokens	CloudGo Tokens	Opus Decay	GPT Decay	CG Decay
#1	Three-Cloud (AWS, GCP, Azure), ~80 resources	424,307	418,707	249,327	14%	17%	2%
#2	Single-Cloud AWS, ~20 resources	383,196	242,798	228,149	5%	7%	<1%
#3	Enterprise + Terraform, ~130 resources	789,413	501,611	371,581	18%	21%	4%
#4	AWS + GCP, ~45 resources	670,508	472,103	236,417	8%	11%	1%
#5	Full Migration + Compliance, ~200 resources	867,081	684,496	607,161	22%	25%	3%
Avg		**626,901**	**463,943**	**338,527**	**13.4%**	**16.2%**	**2.0%**

Table 1. Total tokens consumed (input + output) and context decay per independent run. Decay is measured as the percentage of scanned resources omitted or inaccurately represented in the final plan.

CloudGo.ai averaged 338,527 tokens per run compared to 626,901 for Claude Opus 4.6 and 463,943 for ChatGPT 5.4, representing a 46% reduction relative to Claude and a 27% reduction relative to ChatGPT. CloudGo.ai maintained a context decay rate of just 2.0% on average, meaning that across every test environment, virtually all scanned resources were accurately represented in the final migration plan. Claude Opus 4.6 averaged 13.4% context decay and ChatGPT 5.4 averaged 16.2%.

Because these runs were independent, the decay figures are not cumulative. Each number reflects the accuracy loss observed within that single session. The consistency of CloudGo.ai's low decay across environments of very different sizes and complexities reflects the structural advantage of its pre-indexed intelligence layer.

Turn Count and Signal-to-Noise Ratio

Run	Environment	Opus 4.6 Turns	GPT 5.4 Turns	CloudGo Turns	Opus S/N	GPT S/N	CG S/N
Run 1	Three-Cloud (AWS, GCP, Azure), ~80 resources	8	4	**1**	72%	54%	**89%**
Run 2	Single-Cloud AWS, ~20 resources	5	2	**1**	85%	68%	**94%**
Run 3	Enterprise + Terraform, ~130 resources	12	7	**1**	63%	47%	**86%**
Run 4	AWS + GCP, ~45 resources	7	4	**1**	80%	59%	**92%**
Run 5	Full Migration + Compliance, ~200 resources	12	5	**1**	55%	39%	**84%**
Avg		**8.8**	**4.4**	**1.0**	**71%**	**53%**	**89%**

Table 2. Interaction turns required to reach a comparable plan, and signal-to-noise ratio (percentage of model output that was directly actionable planning content).

CloudGo.ai completed every test environment in a single turn. Its one-shot architecture, powered by pre-indexed cloud intelligence and structured tool orchestration, allows it to scan, analyze, and produce a complete migration plan without iterating. By contrast, Claude Opus 4.6 averaged 8.8 turns per run and ChatGPT 5.4 averaged 4.4 turns. Each additional turn represents another cycle of inference, tool calling, and context accumulation, and more turns means more opportunities for the model to lose its thread, repeat itself, or drift from the original objective.

Signal-to-noise ratio captures how much of the model's output was actually useful. When a standalone model encounters an unfamiliar cloud resource or an ambiguous dependency, it tends to generate exploratory output: repeating earlier findings, hedging its analysis, or producing plausible-sounding but inaccurate configuration details. CloudGo.ai averaged an 89% signal-to-noise ratio, meaning that nearly nine out of ten tokens in its output were directly contributing to the final plan. Claude averaged 71% and ChatGPT 53%. While Claude's signal ratio is respectable in isolation, it still means that nearly a third of its output required human filtering, and ChatGPT's ratio means roughly half of its output was noise. The one-shot architecture is a key contributor here; because CloudGo.ai does not iterate, it does not accumulate the exploratory and repetitive content that multi-turn sessions produce.

4) Cost Implications

Token savings translate directly into dollar savings. CloudGo.ai's proprietary intelligence layer reduces not just the volume of tokens consumed but also the per-run cost of producing a comparable migration plan. When you factor in the reduced token volume and CloudGo.ai's optimized architecture, the cost difference is meaningful.

Model	Avg Tokens/Run	Blended $/1M Tokens	Avg Cost/Run
Claude Opus 4.6	626,901	$30.00	$1.36
ChatGPT 5.4	463,943	$25.00	$1.05
CloudGo.ai	338,527	-	**$0.719**

Table 3. Average cost per run based on blended input/output token rates observed during testing.

On average, a single CloudGo.ai run costs $0.72 compared to $1.36 for Claude Opus 4.6 and $1.05 for ChatGPT 5.4. That is a 47% cost reduction relative to Claude and a 31% cost reduction relative to ChatGPT. For teams running dozens or hundreds of cloud analyses per month, the savings compound quickly. A team running 50 analyses per month would spend roughly $36 with CloudGo.ai versus $53 to $68 with standalone frontier models. These figures reflect blended token rates that account for the typical mix of input and output tokens observed during testing. Actual costs will vary depending on the model provider's current pricing, but the proportional savings are consistent across pricing tiers.

Long-Term Savings

Per-run savings may appear modest in isolation, but cloud infrastructure analysis is not a one-time activity. Teams revisit their environments regularly as they plan migrations, evaluate security postures, assess compliance, and respond to architectural changes. A mid-size platform organization might easily consume 100 million tokens per month on AI-assisted cloud workflows across its engineering and infrastructure teams. At that volume, the cost difference between platforms becomes tangible within the first year and continues to grow.

Model	Monthly Cost	12-Month Cost	24-Month Cost	36-Month Cost
Claude Opus 4.6	$3,000	$36,000	$72,000	$108,000
ChatGPT 5.4	$2,500	$30,000	$60,000	$90,000
CloudGo.ai	**$1,590**	**$19,080**	**$38,160**	**$57,240**

Table 4. Cumulative API cost for 100 million tokens per month of equivalent cloud analysis workload. Savings scale linearly; an organization consuming 500 million tokens per month would see 5x these figures.

At 100 million tokens per month, CloudGo.ai saves $16,920 compared to Claude Opus 4.6 and $10,920 compared to ChatGPT 5.4 in the first year alone. Over three years, the savings grow to $50,760 versus Claude and $32,760 versus ChatGPT. These are direct API cost savings and they scale linearly with usage. An organization consuming 500 million tokens per month, which is common across larger engineering organizations with multiple teams and cloud accounts, would save over $84,000 versus Claude in the first year and over $250,000 over three years. The savings grow with every additional team, account, and cloud environment brought into the workflow.

5) Why These Metrics Matter (Beyond Cost)

Cost savings are the most immediately visible benefit, but token efficiency and context preservation have deeper implications for the quality and reliability of AI-driven cloud work.

Accuracy Scales with Complexity

Cloud environments are not static. They grow, interconnect, and accumulate technical debt over time. The more complex a deployment becomes, the more critical it is that an AI assistant can hold the full picture in working memory without losing fidelity. A model that shows 14% context decay against 80 resources may show 22% or worse against 200 resources. For organizations with the largest, most complex environments, this is the difference between a migration plan they can trust and one they need to audit line by line.

CloudGo.ai's architecture keeps decay consistently low across all tested complexity levels. Whether the environment had 20 resources or 200, CloudGo.ai's decay stayed between less than 1% and 4%. The wider the gap between a standalone model's accuracy and CloudGo.ai's, the more valuable CloudGo.ai becomes for that specific deployment.

One-Shot Execution

CloudGo.ai's ability to complete its analysis in a single turn is not just a speed advantage. Multi-turn sessions introduce compounding risk: each turn adds tokens to the context window, each tool call returns raw data that the model must integrate, and each response is an opportunity for the model to rephrase, hedge, or lose track of earlier findings. A single-turn architecture sidesteps all of this. The model receives a structured, pre-indexed view of the environment, executes its analysis against CloudGo.ai's intelligence layer, and produces a complete plan in one pass. The result is faster, cheaper, and structurally more reliable than any iterative approach.

Signal Quality Compounds

An 89% signal-to-noise ratio does not just mean less wasted output. It means the human reviewer spends their time evaluating real recommendations rather than filtering through exploratory hedging and repeated context. Even Claude, which achieved a 71% signal ratio, leaves nearly a third of its output as noise that a reviewer must parse. ChatGPT at 53% means the reviewer is doing almost half the model's job. When the ratio is 89% signal, the model is doing its job and the reviewer can focus on strategic decisions. For teams evaluating complex multi-cloud migrations, this distinction determines whether AI assistance accelerates the process or merely relocates the effort.

Catching Problems Before They Become Incidents

The benefits of higher output quality extend well beyond the cost of tokens. A cloud migration plan is ultimately a set of instructions that real engineers will execute against real production infrastructure. When a model hallucinates a security group configuration, omits a cross-region dependency, or suggests an IAM policy that is overly permissive, those errors do not stay on paper. They become misconfigurations in production, and misconfigurations become incidents.

CloudGo.ai's low context decay and high signal-to-noise ratio mean that its plans are more complete and more accurate from the start. A plan that captures 98% of scanned resources with correct configurations is not just marginally better than one that captures 84%; it is the

difference between a migration that proceeds smoothly and one that generates security findings, compliance violations, or outages in the weeks after execution. The cost of a single production incident, including engineer time, customer impact, and remediation effort, can easily exceed the total API spend for an entire year of cloud analysis. CloudGo.ai's architecture is designed to reduce that risk at the planning stage, before misconfigurations ever reach production.

Fewer Hallucinations, Better Tool Use

When standalone models encounter the limits of their training knowledge during cloud analysis, they have two options: make additional tool calls to gather more information, or approximate. Both options carry costs. Additional tool calls inflate token counts and add latency. Approximation introduces hallucination risk, where the model generates plausible-sounding but incorrect resource configurations, dependency relationships, or provider-specific details.

CloudGo.ai reduces both failure modes simultaneously. Its proprietary intelligence layer provides the model with pre-verified, structured cloud knowledge, which means the model does not need to make speculative tool calls to fill knowledge gaps, and it does not need to approximate when it lacks information. The result is fewer total tool invocations per run, less raw data flowing through the context window, and substantially fewer hallucinated details in the final output. For teams reviewing and acting on these plans, this translates directly into less time spent validating model output and higher confidence in the recommendations they receive.

Scalability for Your Environment

The efficiency gains shown here were measured against representative environments of varying size. Organizations with particularly complex setups, extensive Terraform state, strict compliance requirements, or deep cross-cloud integrations can expect even greater relative gains. CloudGo.ai's advantage compounds with complexity because its pre-indexed knowledge eliminates the need for the model to re-derive information that has already been captured, structured, and optimized.

6) CloudGo.ai's Proprietary Intelligence Layer

The results described above are not the product of prompt engineering tricks or model fine-tuning. They stem from a fundamentally different architecture that offloads cloud-specific knowledge from the model's context window into a set of purpose-built, continuously maintained intelligence systems.

Cloud Best Practice Database

CloudGo.ai maintains a curated vector database of cloud best practices spanning AWS, GCP, and Azure. This database covers architecture patterns, service selection, security postures, cost optimization strategies, and compliance frameworks. When a model running on CloudGo.ai needs to evaluate a resource or recommend a migration path, it queries this database directly rather than relying on whatever the model happens to have retained from training. The result is consistent, current, and provider-accurate guidance that doesn't consume context window space.

Terraform Knowledge Graph

CloudGo.ai's Terraform knowledge graph is a Neo4j-backed system that has ingested the full provider documentation, community modules, and resource relationship schemas for AWS, GCP, and Azure Terraform providers. When a model needs to reason about infrastructure-as-code, it can retrieve the exact resource definitions, attribute constraints, and dependency chains it needs instead of approximating from general knowledge. This eliminates an entire category of hallucination risk around Terraform configurations and dramatically reduces the number of exploratory tool calls a model would otherwise need to make.

Embedded Company Documentation

For each customer engagement, CloudGo.ai can ingest and embed the organization's own documentation, runbooks, architectural decision records, and compliance policies into a dedicated per-customer vector collection. This means the model doesn't need to ask for context it should already have, and it doesn't need to re-scan documents that have already been processed. The embedded documentation serves as persistent organizational memory that follows the model across sessions without inflating token counts.

Multi-Agent Orchestration

CloudGo.ai's LangGraph architecture routes tasks to specialized sub-agents for cloud scanning, research, Terraform analysis, and integration planning. Each agent operates within a focused scope, retrieves only the context it needs from the intelligence layer, and returns structured results to the orchestrator. This prevents the monolithic context accumulation that causes decay in standalone model interactions. Instead of one agent trying to hold everything, the work is divided across agents that each maintain clean, minimal context windows.

7) Conclusion

The gap between what general-purpose models can achieve on their own and what they can achieve with CloudGo.ai's infrastructure intelligence layer is large and consistent across every environment we tested. A 46% reduction in token consumption, an 85% reduction in context decay, and a 47% reduction in per-run cost are not marginal improvements. They represent a structural advantage that comes from building the right intelligence systems around the model rather than expecting the model to carry the full burden on its own.

For CTOs, platform engineers, and cloud architects managing real production infrastructure across multiple providers, CloudGo.ai offers a way to get trustworthy, cost-effective AI assistance for cloud planning without the accuracy risks and escalating costs of standalone model usage.