Corvid Cloud support AI: where the LLM bill goes, and the six changes that cut it

$23,117 → $433 a month

98.1% lower at Anthropic list prices · quality: 76 of 81 checks pass after (78 of 81 before)

COMPANY	WORKLOAD	PRICES	PREPARED BY
Corvid Cloud	81 real model calls, scaled to 1,200,000 a month (assumption)	Public list prices read 11 Nov 2026	KH Labs · Khoa Tran

Executive summary

At 1,200,000 requests a month the support assistant costs $23,117. Six changes bring it to $433, a saving of $22,684 a month. Quality: 76 of 81 checks pass after (78 of 81 before), 2 fewer than before, every miss listed. Three changes do most of the work: trim prompt and rag context ($13,410), cap output and reasoning ($5,000) and route easy calls to a cheaper model ($3,689).

LEVER	SAVES / MONTH	OF THE BILL	BILL AFTER
1. Trim prompt and RAG context	$13,410	58.0%	$9,707
2. Cap output and reasoning	$5,000	21.6%	$4,707
3. Route easy calls to a cheaper model	$3,689	16.0%	$1,018
4. Prompt caching	$567	2.5%	$451
5. Dedupe repeated calls	$10	under 0.1%	$441
6. Batch the nightly job	$8	under 0.1%	$433
Total	**$22,684**	**98.1%**	**$433**

Savings in the order they were applied. Each is measured on what the previous levers left.

Cost per request falls from $0.019264 to $0.000361. At that rate the $1,900 fee is covered after about 100,512 requests.

Quality: 76 of 81 checks pass after, 78 of 81 before. The 5 remaining failures are listed in full; 2 of them also fail on the baseline.

Baseline shape. The baseline is a typical unoptimized stack: one big reasoning model on every call, a 3,073-token system prompt, 10 retrieved chunks, no output caps. A leaner stack starts from a lower bill; the free 48-hour AI check sizes the saving to yours before you pay.

Baseline: what the bill is made of

On your bill the baseline is your last full invoice month, reconciled line by line against a usage export, and we report the variance. Here the baseline is the API `usage` field of each of the 81 baseline calls, priced at Claude Sonnet 5.5 list prices ($2.00 per million input tokens, $10.00 per million output). The table re-adds those calls and checks the sum against the analysis.

TASK	CALLS	INPUT TOKENS	OUTPUT TOKENS	COST, SAMPLE	PER MONTH
Support answers (RAG)	37	161,440	24,550	$0.5684	$8,420
Ticket classifier	24	76,978	15,532	$0.3093	$4,582
Ticket summaries	12	39,973	12,979	$0.2097	$3,107
Nightly account digest	8	32,800	40,739	$0.4730	$7,007
Total	**81**	**311,191**	**93,800**	**$1.5604**	**$23,117**

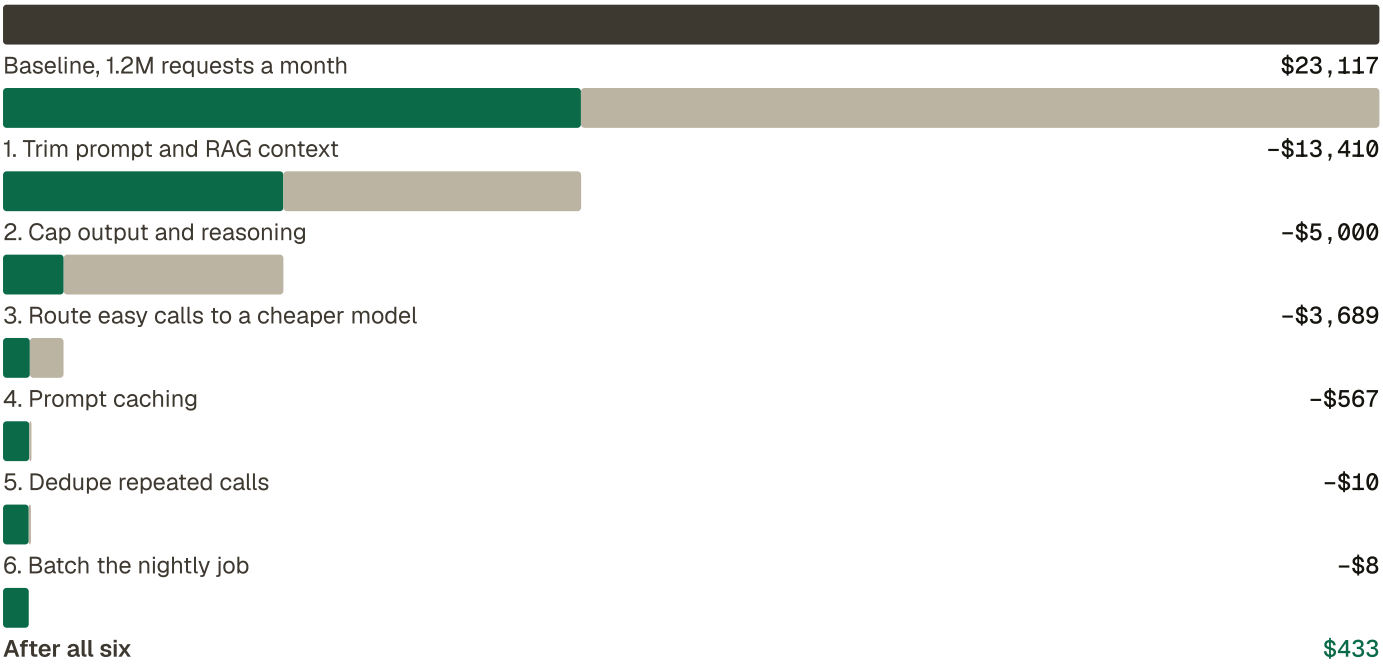
Baseline calls re-added from runs/baseline.json

- `Re-added total $23,116.77 against the analysis figure $23,116.77: variance $0.00.`

Per month = sample cost × 1,200,000 / 81. The volume is an assumption (assumption: the 81-request mix repeats at 1.2M requests a month (about 10 percent of volume is per-account nightly digests)). The nightly digest is 8 of 81 calls but 30% of the cost: its average output was 5,092 tokens.

Savings by lever

Levers were applied one after another on the same 81 requests. Each section shows the money, one measured request before and after, the pull request that makes the change, and how to undo it.



Monthly bill after each lever, Anthropic list prices. Grey is what that lever removed; green is what is still paid.

1 Trim prompt and RAG context

$13,410 / month
58.0% of the original bill

Shorter system prompt with every rule still in it, and 3 reranked chunks instead of the top 10. Bill before this lever $23,117, after $9,707 a month.

Measured on request R074: “How do I reset 2FA for a teammate who lost their phone?”

	SETUP	INPUT TOK.	OUTPUT TOK.	COST / CALL
Before	System prompt 3,073 tokens + 10 retrieved chunks `claude-sonnet-5-5-high`	4,371	481	$0.013552
After	System prompt 1,792 tokens (same rules, no filler) + 3 reranked chunks `claude-sonnet-5-5-high`	2,266	341	$0.007942

PR 1 · Trim the system prompt and rerank context `support/answer.py`

```
   def answer(question):
-      chunks = search(question, top_k=10)            # naive top-10
-      system = SYSTEM_PROMPT                         # 3,073 tokens: persona, greetings, repeated
rules
+      chunks = rerank(search(question, top_k=20))[:3]  # reranked top-3
+      system = POLICY_PACK + RULES                   # 1,792 tokens: every rule still in, filler
removed
       return llm.chat(model=BIG, system=system, user=render(chunks, question))
```

Harness: pipeline.build() baseline vs trim branch; prompts.BASE_SYSTEM vs prompts.OPT_QA_SYSTEM; retrieval.naive_top_k(q, 10) vs rerank_top_k(q, 3).

Risk. A rule or fact drops out of the shorter context. All-gold-chunks-in-context fell from 94.4% (top-10) to 88.9% (top-3). **Rollback.** Feature flag per prompt version; revert restores the old prompt and top-k in one deploy.

2 Cap output and reasoning

$5,000 / month
21.6% of the original bill

Terse output format, max_tokens caps, and a low reasoning budget on the big model. Bill before this lever $9,707, after $4,707 a month.

Measured on request R060: “Lost my phone, locked out of the owner account because of 2FA.”

	SETUP	INPUT TOK.	OUTPUT TOK.	COST / CALL
Before	Asked to explain its reasoning, then give a label `claude-sonnet-5-5-high`	486	424	$0.005212
After	One label, max 12 tokens, low reasoning budget `claude-sonnet-5-5-high`	454	5	$0.000958

PR 2 · Cap output and reasoning `llm/client.py`

```
   MAX_TOKENS = {"qa": 320, "classify": 12, "summarize": 220, "report": 800}
   def chat(task, model, system, user):
-      return client.create(model=model, system=system, messages=user)
+      return client.create(model=model, system=system, messages=user,
+                           max_tokens=MAX_TOKENS[task],
+                           reasoning_effort="low" if model == BIG else None,
+                           response_format=TERSE_JSON[task])   # e.g. one label for the
classifier
```

Harness: pipeline.MAX_TOKENS and max_tokens_for(); run_experiment.one() sets reasoning_effort=low on the big model from the caps stage on.

Risk. Truncated answers. In this run one big-model digest hit its 800-token cap at the caps stage (that stage was not shipped). **Rollback.** Caps are per task in one dict; raise or remove a cap without other changes.

3 Route easy calls to a cheaper model

$3,689 / month
16.0% of the original bill

Classifier, summaries and nightly digests go to the small model; routine support questions too. Escalation re-runs doubtful answers on the big model. Bill before this lever $4,707, after $1,018 a month.

Measured on request R077: “Ticket T-7105 | Customer: Aisha (Business)”

	SETUP	INPUT TOK.	OUTPUT TOK.	COST / CALL
Before	Big model: claude-sonnet-5-5-high	290	126	$0.00184
After	Small model: gemini-3.1-flash-lite	184	52	$0.000044

PR 3 · Route easy calls to the small model, escalate doubtful ones `llm/router.py`

```
+ SENSITIVE = ["refund", "delete", "deletion", "gdpr", "privacy", "security", ...]
+ def route(task, question):
+     if task in ("classify", "summarize", "report"): return SMALL
+     if any(k in question.lower() for k in SENSITIVE): return BIG
+     if needs_arithmetic(question): return BIG           # two or more numbers, 'how much', %
+     return SMALL
+
+ def answer(question):
+     out = chat(route("qa", question), ...)
+     if not out.confident or not out.cite or set(out.cite) - retrieved:
+         out = chat(BIG, ...)                            # escalation re-run
+     return out
```

Harness: router.route() and router.should_escalate() (v2 rules), used by run_experiment.run_v2().

Risk. The small model misses. The escalation rule re-ran 4 answers on the big model; 7 numeric or sensitive questions went straight to it. **Rollback.** Router returns BIG for everything when the flag is off.

4 Prompt caching

$567 / month
2.5% of the original bill

The stable prefix (policy pack, label definitions) is billed at the cached-input rate once it is warm. Bill before this lever $1,018, after $451 a month.

Applied alone to the original bill it would save $6,564 a month; it reads small here because earlier levers already removed most of the tokens it acts on.

Measured on request R074: “How do I reset 2FA for a teammate who lost their phone?”

	SETUP	INPUT TOK.	OUTPUT TOK.	COST / CALL
Before	Every token billed at the normal input rate `gemini-3.1-flash-lite`	1,527	48	$0.000177
After	1,172 of 1,527 prompt tokens are the stable prefix, billed at the cached rate `gemini-3.1-flash-lite`	1,527	48	$0.000073

PR 4 · Cache the stable prefix `llm/client.py`

```
-       system=system,
+       system=[{"type": "text", "text": system,          # policy pack + label definitions
+                "cache_control": {"type": "ephemeral"}}],  # prefix must not change between calls
```

Harness: cost.cache_split() on measured prefix sizes (runs/calibration.json); hit rate 95% and minimum 1,024 tokens are assumptions.

Risk. Lower hit rate than the 95% assumed (cache TTL, low traffic hours); a prefix that changes per call caches nothing. **Rollback.** Remove cache_control; prices fall back to normal input rates, answers unchanged.

---

5 Dedupe repeated calls

$10 / month
under 0.1% of the original bill

Exact-match and normalized-text cache: a repeated question is answered without a model call. Bill before this lever $451, after $441 a month.

Applied alone to the original bill it would save $1,718 a month; it reads small here because earlier levers already removed most of the tokens it acts on.

Measured on request R026: “What happens if my card payment fails?”

	SETUP	INPUT TOK.	OUTPUT TOK.	COST / CALL
Before	A model call for a question we already answered `gemini-3.1-flash-lite`	1,558	95	$0.000097
After	normalized cache hit on R022: "What happens when my card payment fails"	0	0	$0.

PR 5 · Answer repeated questions from a response cache `support/answer.py`

```
  def answer(question):
+       hit = cache.get("qa", question)          # exact, then normalized text
+       if hit: return hit
        out = ...
+       cache.put("qa", question, out)           # invalidated when the KB changes
```

Harness: cache.ResponseCache (exact + normalized keys), replayed over the 81 requests.

Risk. A cached answer goes stale after a policy or KB change. **Rollback.** Flush on KB deploy; TTL; flag off.

---

6 Batch the nightly job

$8 / month
under 0.1% of the original bill

The nightly digest is not urgent, so it runs through the Batch API at half price. Bill before this lever $441, after $433 a month.

Applied alone to the original bill it would save $3,504 a month; it reads small here because earlier levers already removed most of the tokens it acts on.

Measured on request R009: “Nightly digest: Fjord Analytics”

	SETUP	INPUT TOK.	OUTPUT TOK.	COST / CALL
Before	Synchronous call, nobody is waiting for it at 02:00 `gemini-3.1-flash-lite`	913	98	$0.00014
After	Same call through the Batch API (50 percent off, results within 24 h) `gemini-3.1-flash-lite`	913	98	$0.00007

PR 6 · Run the nightly digest through the Batch API — `jobs/nightly_digest.py`

```
- for account in accounts:
-     digest = llm.chat(task="report", ...)
+ batch = client.messages.batches.create(requests=[
+     {"custom_id": a.id, "params": digest_params(a)} for a in accounts])
+ # results within 24 h at half price; collected by the 06:00 job
```

Harness: batch factor 0.5 applied to the digest calls in analyze.py (pricing.json batch_factor).

Risk. Digests arrive later (within 24 h, not at once). **Rollback.** Switch the job back to synchronous calls.

Each PR is shown in the shape it would land in your repository. In this sample the change was made in our test harness; the caption under each diff names the file. Caching, dedupe and batch were priced on measured tokens rather than run as separate model calls.

Same tokens, three price lists

The run used one model pair through a proxy. To show the answer does not hinge on a vendor, the same measured token counts are priced as each provider's big and small tier. The big and small tier for each provider is our mapping to its commercial models.

PROVIDER	BIG / SMALL TIER PRICED	BEFORE	AFTER	SAVED
Anthropic	Claude Sonnet 5.5 / Claude Haiku 5.5 (prompts up to 100k)	$23,117	$433	98.1%
OpenAI	gpt-6.1-sol / gpt-6-luna	$23,117	$425	98.2%
Google	gemini-4-carbon / gemini-3.1-flash-lite	$23,117	$524	97.7%

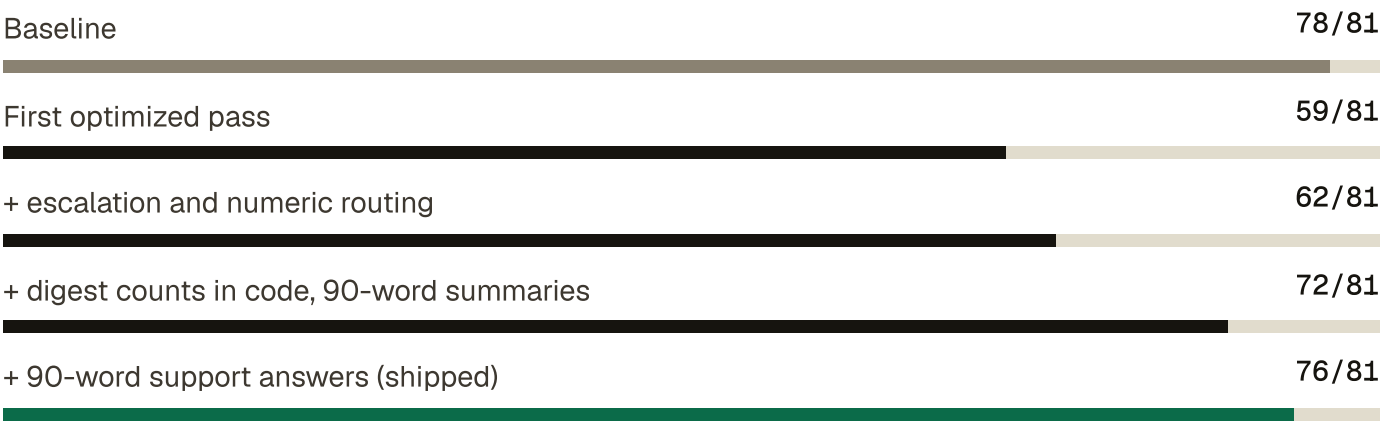
Google's Gemini 4 Carbon list prices rise on 2026-12-01; the table uses the prices valid on 2026-11-11.

Quality eval

Nothing ships that fails the eval. The classifier is scored by exact label match; everything else by an LLM judge (gemini-4-argon-high) with a written rubric (correct, complete, policy-safe), the full knowledge base, must-have facts and a must-not list.

TASK	BASELINE	AFTER	CHANGE
Support answers (RAG)	36/37	35/37	-1
Ticket classifier	23/24	23/24	0
Ticket summaries	12/12	10/12	-2
Nightly account digest	7/8	8/8	+1
All checks	**78/81 (96.3%)**	**76/81 (93.8%)**	**-2**

How the pass rate moved while tuning



Eval method. All 81 requests serve as both tuning set and eval, and the last two fixes came from reading eval failures, so 76/81 is an in-sample score; the first optimized pass (59/81) is the closest to a blind score. On your traffic we split before tuning: 80% to tune on, 20% reserved as a test set and scored once, at the end.

Every check that still fails

ID	TASK	REQUEST	BASELINE	JUDGE NOTE ON THE FINAL ANSWER
R019	Ticket summaries	Ticket T-7108 | Customer: Hiro (Business) Hiro: We need our data in the EU region for a new German client. Can you move …	pass	Missed specifying EU region and creating a new EU workspace.
R031	Support answers (RAG)	My invoice has the wrong company name. Can you reissue last month's?	pass	Did not offer to check with the Billing team about reissuing last month's invoice.
R045	Support answers (RAG)	How do I set up a webhook and verify it's really from you?	fail	Omitted Settings > Integrations > Webhooks and falsely claimed setup location wasn't in the policy.
R049	Ticket summaries	Ticket T-7109 | Customer: Dana (Team) Dana: Charged $1,740 on the 1st but we only have 60 seats at $29. Should be $1,740…	pass	Missing the 'no action' part of the status.
R056	Ticket classifier	Is there a way to switch the Kanban board to dark mode?	fail	exact label mismatch

Risks and rollback

Each lever ships behind its own flag, so any one can be turned off without touching the others. The eval runs on every PR and nightly for the 30 days after.

LEVER	WHAT CAN GO WRONG	ROLLBACK
Trim prompt and RAG context	A rule or fact drops out of the shorter context. All-gold-chunks-in-context fell from 94.4% (top-10) to 88.9% (top-3).	Feature flag per prompt version; revert restores the old prompt and top-k in one deploy.
Cap output and reasoning	Truncated answers. In this run one big-model digest hit its 800-token cap at the caps stage (that stage was not shipped).	Caps are per task in one dict; raise or remove a cap without other changes.
Route easy calls to a cheaper model	The small model misses. The escalation rule re-ran 4 answers on the big model; 7 numeric or sensitive questions went straight to it.	Router returns BIG for everything when the flag is off.
Prompt caching	Lower hit rate than the 95% assumed (cache TTL, low traffic hours); a prefix that changes per call caches nothing.	Remove cache_control; prices fall back to normal input rates, answers unchanged.
Dedupe repeated calls	A cached answer goes stale after a policy or KB change.	Flush on KB deploy; TTL; flag off.
Batch the nightly job	Digests arrive later (within 24 h, not at once).	Switch the job back to synchronous calls.

30-day verification plan

Savings are measured per request, so a busier or quieter month does not change the answer. The baseline cost per request is frozen before any PR merges.

`Day 0`	Freeze the baseline: last full invoice month, reconciled with the usage export, divided by its request count. Sample: $0.019264 per request.

Days 1-3	Merge the PRs behind flags. Turn them on for 10% of traffic; compare cost per request and eval pass rate against the other 90%.
Days 4-7	Full traffic once the reserved test set scores within the agreed margin of the baseline. Any lever that fails is switched off and listed in the report.
Days 8-30	Daily cost per request from the usage export, nightly eval on the reserved test set, an alert if either drifts.
Day 30	Reconcile the invoice. Verified saving = (baseline cost per request − new cost per request) × requests actually served. Sample: ($0.019264 − $0.000361) × 1,200,000 = $22,684.

The guarantee

$1,900

FIXED FEE · ABOUT ONE WEEK

If verified 30-day savings are below the $1,900 fee, we refund the difference. Verified saving = (old cost per request − new cost per request) × requests served. You keep the PRs, the eval and the report either way. This fits teams spending at least $5,000 a month on model APIs; below that the fee can exceed what there is to save.

- ✓ Shipped PRs, one per lever, each behind a flag
- ✓ A before and after report reconciled to your invoice
- ✓ An eval built from your own traffic, with a reserved test split
- ✓ A ranked fix list with dollars per fix

Get this report on your own bill

Send your AI bill export. The free 48-hour AI check tells you which of the six levers apply and a dollar range, before you pay anything. Or check your bill free in

your browser with the LLM Bill Checker. Get a free AI check: khlabs.dev/#check · linkedin.com/in/khoa-tran-khlabs · khoatran.mma26@gmail.com

Or write to khoatran.mma26@gmail.com. Download this report as a PDF.

Method

- **Workload.** 81 requests: 37 support questions over a 32-chunk knowledge base (8 are repeats or rephrasings), 24 ticket classifications with gold labels, 12 ticket summaries, 8 nightly digests.
- **Models.** Big claude-sonnet-5-5-high, small gemini-3.1-flash-lite, judge gemini-4-argon-high, through a local OpenAI-compatible proxy. Tokens from each call's usage field.
- **Caching** assumes 95% of a stable prefix of at least 1,024 tokens is served from cache; TTL and storage fees are reconciled against your invoice when we run it on your bill. **Dedupe** found 3 exact and 5 normalized repeats in 81. **Batch** is priced at half.
- **Rigor.** Every stage is a real run at temperature 0, scored by exact label match and by one LLM judge from the same family as the small model; the judge is a measurement, not ground truth. On your traffic we add repeated runs per stage and the reserved test split.
- **Sources.** public/data/results.json (engine/analyze.py), runs/*.json, pricing.json. This page: engine/build_full_report.py, which re-adds the baseline and refuses to build if it does not match.