Cloud Cost Reality Check

Most teams can’t tell you what one customer costs to serve. They can tell you the monthly bill, and they can tell you it’s too high, and those two facts together produce a quarter of thrashing that saves eleven percent and breaks something.

The number that matters is cost per unit of the thing you sell. Once you have it, most decisions become obvious and most panics become unnecessary.

I have run this on infrastructure from a handful of instances up to three thousand. The pattern of where the money goes barely changes with scale. Only the size of the mistake does.

> Start with one number. Total infrastructure spend last month, divided by the number of active customers, or transactions, or whatever you actually charge for. Write it down before you read further. Most teams have never calculated it, and the reaction to seeing it is the useful part.

Know the number

OK	CON	N/A	
☐	☐	☐	Total monthly infrastructure spend, all providers, including the ones on someone’s card
☐	☐	☐	Cost per active user, or per transaction, or per whatever you invoice
☐	☐	☐	That figure tracked over the last twelve months, not just this month
☐	☐	☐	Direction of travel: is cost per unit rising or falling as you grow?
			—Falling is the whole point of leverage. Rising means you’re buying revenue.
☐	☐	☐	Gross margin including infrastructure, not just cost of goods as accounting defines it
☐	☐	☐	The cost of your single most expensive customer
☐	☐	☐	What a tenfold traffic increase would cost, calculated rather than assumed

Find the waste

In roughly the order I find money.

OK	CON	N/A	
☐	☐	☐	Non-production environments running production-sized, around the clock
☐	☐	☐	Environments nobody has logged into for 90 days
☐	☐	☐	Old snapshots, images and backups with no retention policy
☐	☐	☐	Detached volumes and unattached addresses
☐	☐	☐	Load balancers in front of nothing
☐	☐	☐	Over-provisioned instances chosen by guess and never revisited
☐	☐	☐	Logging everything at debug, retained for a year, in the expensive tier

- [ ] [ ] [ ] Data egress between zones or regions that a layout change would remove
  —Egress is the one that surprises people. It doesn’t show up as a resource you can see.
- [ ] [ ] [ ] Managed service tiers bought for a peak that happened once
- [ ] [ ] [ ] NAT gateway traffic that could have been a private endpoint
- [ ] [ ] [ ] Idle capacity on nights and weekends for a workload with business-hours traffic
- [ ] [ ] [ ] Duplicate observability tooling doing the same job

Structural costs

OK	CON	N/A	
☐	☐	☐	Storage class matches access pattern, and cold data is actually cold
☐	☐	☐	Retention policies exist and are enforced, not aspirational
☐	☐	☐	Commitments, reservations or savings plans match steady-state usage
☐	☐	☐	Those commitments have documented end dates and an owner
☐	☐	☐	Autoscaling scales down as well as up, and you have watched it do so
☐	☐	☐	Instance families are current generation
☐	☐	☐	Anything cache-able is cached, and the cache hit rate is known
☐	☐	☐	The database is sized for its working set rather than its total data

Attribution

You can’t manage what you can’t attribute.

OK	CON	N/A	
☐	☐	☐	Every resource carries an owner tag, a service tag and an environment tag
☐	☐	☐	Untagged spend is under five percent, and you know what it is
☐	☐	☐	Each team or product line can see its own spend
☐	☐	☐	Someone is accountable for the bill, by name
☐	☐	☐	Cost appears in a dashboard people actually open, not a monthly email
☐	☐	☐	There’s an alert on unusual growth, not just on a monthly total

Before you optimise

OK	CON	N/A	
☐	☐	☐	The cheapest engineering hour is one you don’t spend saving fifty dollars
☐	☐	☐	Rank every candidate saving by annual value against days of work
☐	☐	☐	Anything under one percent of the bill goes on a list, not on a sprint
☐	☐	☐	Confirm the workload isn’t about to change or be replaced

- ☐ ☐ ☐ Check that the saving doesn’t move cost into an engineer’s week forever
  —A saving that costs a day of toil every month isn’t a saving.
- ☐ ☐ ☐ Measure before, change one thing, measure after
- ☐ ☐ ☐ Write down what you expected to save and compare it to what you did

The questions that actually change the number

OK	CON	N/A	
☐	☐	☐	Is anything running that no customer would miss?
☐	☐	☐	Is any of this architecture sized for a scale you haven’t reached?
☐	☐	☐	What is the single largest line, and does its size make sense to a person outside the team?
☐	☐	☐	If the bill had to halve in ninety days, what would you do, and why are you not doing the top item now?
☐	☐	☐	What would the bill be if you designed this today, knowing what you know?

> **On an acquisition:** CodeProvenance.ai looks for the cost patterns in this document (uncapped token use, retries with no ceiling, idle reserved GPUs) in the target’s own code, and reads its cloud and model invoices in the buyout tier. Estimates show their inputs; the preflight does not promise a saving.