FREE COST CALCULATOR

The SaaS Infrastructure Cost Calculator

A worksheet for working out what your infrastructure really costs per customer, which line items are worth attacking, and how much of the bill is avoidable, for founders and engineering leads staring at a cloud invoice that keeps growing.

Cloud bills grow for structural reasons, not mysterious ones. Environments outlive the projects that created them. Log retention gets set once and never revisited. An instance sized for a launch peak runs at low utilization for two years. Nobody is wrong at any individual step, and the total ends up disconnected from the work it supports. This calculator gives you the arithmetic to see where that happened and what the fix is worth.

Do the arithmetic before you do any optimization. Almost every engineering team can name a technical improvement they would like to make. Very few can say what it is worth per month, which is why infrastructure cost work is so frequently deprioritized in favor of features that have a number attached.

The eight line items

Break the bill into these categories before you touch anything. Almost every cloud invoice can be mapped onto them, and the mapping itself usually reveals a surprise or two, because the categories that dominate are rarely the ones people assume.

COST CATEGORIES, WHAT DRIVES EACH ONE, AND THE LEVER THAT USUALLY MOVES IT.

Line item	What it covers	Primary driver	Usual lever
Compute	Servers, containers, functions, and the orchestration behind them	Provisioned capacity, not used capacity	Rightsizing, autoscaling, commitment discounts, non-production shutdown
Managed data services	Relational databases, caches, search, queues	Instance size, replicas, high availability configuration	Rightsizing, consolidating small instances, connection pooling
Storage	Block volumes, object storage, snapshots, archives	Volume, retention, and storage class	Lifecycle rules, deleting orphaned volumes and snapshots
Data transfer	Egress to the internet and between regions or zones	Traffic volume and architecture topology	CDN in front, keeping chatty services in one zone, compression
Observability	Logs, metrics, traces, error tracking, session tools	Ingest volume and cardinality, then retention	Sampling, dropping low-value fields, tiered retention
Build and delivery	CI minutes, container registries, artifact storage	Pipeline frequency and duration	Caching layers, parallelism tuning, pruning old images
Third party platform services	Email, payments infrastructure, authentication, feature flags, APIs	Seats, events, or requests	Plan alignment, removing duplicated tools
Support and licenses	Vendor support tiers, commercial licenses, reserved commitments	Contract terms	Right-tiering, annual renegotiation, removing unused seats

THE TWO CATEGORIES THAT SURPRISE PEOPLE

Observability and data transfer are the line items most often larger than expected, because both scale with something other than customer count. Log volume scales with code verbosity and traffic, and egress scales with architecture decisions made years earlier. Neither shows up as a decision anyone made. Both are usually reducible by a large fraction without losing anything anyone would miss.

The formulas

Total and unit cost

Total monthly infrastructure cost = compute + managed data services + storage + data transfer + observability + build and delivery + third party platform services + support and licenses.

Cost per customer = total monthly infrastructure cost divided by active paying accounts.

Infrastructure ratio = total monthly infrastructure cost divided by monthly recurring revenue, expressed as a percentage. This is the number to track over time, because it tells you whether your unit economics are improving or deteriorating as you grow, which a raw dollar total never will.

Fixed versus variable

Variable cost per customer = (total cost this month minus total cost in a month with materially fewer customers) divided by the difference in customer count. Everything left over is your fixed base. This split is what tells you whether growth improves your margin or erodes it, and it is the single most useful thing this worksheet produces.

The value of a fix

Annual value of an optimization = monthly saving times 12. **Payback period in months** = engineering hours required times fully loaded hourly cost, divided by the monthly saving. Anything with a payback period under about three months is straightforward to justify. Anything over twelve needs a reason beyond cost, such as reliability or reduced operational load.

Worked example

All figures below are illustrative round numbers chosen to make the arithmetic legible. They are not benchmarks, market rates, or vendor prices. Substitute your own from your actual invoice.

A business-to-business product with **500 active paying accounts** and **150,000 dollars of monthly recurring revenue** maps its invoice as follows.

ILLUSTRATIVE MONTHLY INFRASTRUCTURE SPEND. HYPOTHETICAL FIGURES.

Line item	Monthly	Share of total
Compute	$12,000	39%
Managed data services	$6,000	19%
Observability	$4,000	13%
Third party platform services	$3,000	10%
Data transfer	$2,500	8%
Storage and snapshots	$1,500	5%
Backups and disaster recovery	$1,200	4%
Build and delivery	$800	2%
Total	$31,000	100%

Cost per customer = 31,000 divided by 500 = **62 dollars per account per month**. **Infrastructure ratio** = 31,000 divided by 150,000 = **20.7 percent of revenue**. That ratio is the headline number, and it is the one to put on a slide.

Now find the avoidable portion

The team runs four checks and finds four things. Non-production environments run around the clock but are used roughly fifty hours a week. Application servers average low utilization because they were sized for a launch that is now two years old. Logs are retained at full fidelity for a long window, when the vast majority of queries only ever look at the last two weeks. Media assets are served directly from object storage rather than through a distribution network, so every request pays egress.

ILLUSTRATIVE SAVINGS FROM FOUR ORDINARY CHANGES. HYPOTHETICAL FIGURES FOR ARITHMETIC ONLY.

Change	Line item	Monthly saving	Rough effort	Payback
Shut down non-production outside working hours	Compute	$1,800	1 day	Under a month
Rightsize over-provisioned application instances	Compute	$2,400	3 days	Under a month
Drop debug logs and tier retention after 14 days	Observability	$1,600	2 days	Under a month
Serve media through a distribution network	Data transfer	$1,000	2 days	About a month
Delete orphaned volumes and old snapshots	Storage	$400	Half a day	Immediate

Total identified saving is 7,200 dollars a month, which is 86,400 dollars a year against roughly eight and a half days of engineering effort. The new total is 23,800 dollars, cost per customer falls from 62 to about 48 dollars, and the infrastructure ratio falls from 20.7 percent to about 15.9 percent. None of those five changes required an architectural rewrite, and none of them touched a line of product code.

> **WHY THIS PATTERN REPEATS**
>
> The largest savings in most infrastructure reviews come from capacity that is provisioned and not used, and from data that is retained and not read. Both accumulate quietly because there is no moment at which anyone decides to waste money. Set a recurring review, quarterly is usually enough, and the accumulation stops being a surprise.

Your worksheet

Compute

servers, containers, functions

Managed data services

databases, caches, search, queues

Observability

logs, metrics, traces, error tracking

Third party platform services

Data transfer

egress and cross-zone

Storage and snapshots

Backups and disaster recovery

Build and delivery

CI, registries, artifacts

Support and licenses

TOTAL monthly infrastructure cost

add the above

Active paying accounts

Monthly recurring revenue

Cost per customer

total divided by accounts

Infrastructure ratio

total divided by MRR, as a percentage

Largest single line item

start here

Estimated non-production share of compute

often 20 to 40 percent

Untagged or unattributed spend

if this is large, that is finding one

The review checklist

Work top to bottom. The early items are cheap and frequently the largest.

- [ ] Tag or label every resource with an owner, an environment, and a service, and find out what percentage of spend is unattributable.
  Untagged resources are the ones nobody will defend and nobody will delete. Getting attribution above about 95 percent changes every conversation that follows.
- [ ] List every environment that exists and confirm someone still needs it.
  Demo environments, an old staging setup, a proof of concept from a project that shipped or was cancelled. This is the most common source of pure waste.
- [ ] Schedule non-production environments to shut down outside working hours.
  Nights and weekends is roughly two thirds of the week. For anything billed by the hour, this is close to free money.
- [ ] Compare provisioned capacity against actual utilization over a full month, including your peak.
  Size for peak plus headroom, not for peak plus fear. Look at the utilization distribution rather than the maximum.
- [ ] Find orphaned resources: unattached volumes, old snapshots, idle load balancers, reserved addresses, abandoned buckets.
  These accumulate at a steady rate and never announce themselves. A quarterly sweep keeps the total near zero.
- [ ] Apply lifecycle rules to object storage so old data moves to cheaper classes or expires.
  Especially for logs, exports, user uploads from deleted accounts, and generated artifacts.
- [ ] Measure log volume by service and find the top three talkers.
  It is common for a small number of services, or one debug statement in a hot path, to account for the majority of ingest.

- [ ] Set retention by value rather than uniformly.
  Ask how far back people actually query. Most investigation happens within days; compliance needs are a separate and much smaller data set.
- [ ] Check high cardinality metrics and traces, which are often billed differently from volume.
  A label containing a user identifier or a request identifier can multiply cost enormously without adding investigative value.
- [ ] Put a distribution network in front of anything served repeatedly to the internet.
  This reduces egress and improves user-facing latency at the same time, which makes it easy to justify twice.
- [ ] Check for cross-zone and cross-region traffic between services that talk constantly.
  An accidental topology decision can produce a permanent recurring charge that no code change ever explains.
- [ ] Review commitment and reservation coverage against your stable baseline only.
  Commit to the floor of your usage, never to the peak. A commitment on capacity you later remove is worse than no commitment.
- [ ] Audit third party tool subscriptions for unused seats, overlapping products, and plan tiers you have outgrown or under-grown.
  Two tools doing the same job is common after any period of fast hiring.
- [ ] Put the infrastructure ratio on a dashboard someone looks at monthly.
  A single tracked ratio does more for cost discipline than any one-off cleanup project, because it makes the drift visible while it is still small.

How to prioritize what you found

1. **Sort by monthly saving divided by engineering days.**
   This ranks the work honestly and usually puts scheduling and cleanup above anything architectural.
2. **Do everything with a payback under a month before anything else.**
   These fund the rest of the effort and build the credibility to ask for time for the larger items.
3. **Separate cost reduction from cost avoidance.**
   Turning something off reduces the bill. Preventing next year's growth is real but will never show up as a decrease, so label it clearly or you will be accused of achieving nothing.
4. **Do not trade away reliability to save a modest amount.**
   Removing a replica, cutting retention below your investigative need, or eliminating headroom can cost far more in one incident than it saves in a year.
5. **Re-run the whole worksheet quarterly.**
   Infrastructure cost is not a project. It is a slow leak, and the only durable fix is a recurring look.

YOUR NEXT STEP

Get a second pair of eyes on the invoice and the architecture behind it

The largest savings usually sit where cost and architecture meet: an environment that should not exist, a data layer sized for a load profile that changed, an observability setup collecting things nobody queries. An infrastructure consultation reviews your spend against your actual workload, separates the quick reductions from the structural ones, and gives you a sequenced plan with the arithmetic attached to each item.

Book an infrastructure consultation at saastechsuite.com

Important. This is a general information worksheet for internal planning. Every figure in the worked example is hypothetical and included only to demonstrate the arithmetic; none of it represents vendor pricing, market rates, benchmarks, or research findings, and no particular saving is promised or implied. Your results depend on your workload, architecture, and contracts. This is not financial, accounting, or tax advice. Test any infrastructure change in a non-production environment first, confirm you have working backups, and review capacity or retention reductions with the engineers responsible for reliability before applying them.