CloudFinOpsKit

Cloud Cost Review Checklist — Azure & AWS

A practical guide and monthly checklist for running a disciplined cloud cost review across Azure and AWS — and for standing up the FinOps operating model, personas and cadence that make those savings stick.

> Free template from the CloudFinOpsKit collection. The monthly checklist (Part B) takes 30–45 minutes to run and works for **Azure, AWS, or both**. Part A explains *how to think about FinOps* technically and organisationally so the numbers translate into real, defensible return on investment.

Part A — How to think about FinOps

Why FinOps, technically

FinOps (Cloud Financial Operations) is the discipline of bringing **engineering, finance and the business** together to get the most value from every pound spent in the cloud. In cloud terms it is the practice of making the **cost of an architectural decision visible at the moment the decision is made** — not three months later on an invoice.

The cloud changed two things that make this essential:

> **Spend is now a variable, engineering-driven decision.** A developer choosing a VM/instance SKU, a redundancy tier or a log-retention period is making a financial commitment in real time. Cost is an architectural attribute, like latency or availability.

> **Consumption is continuous and decentralised.** Hundreds of resources across many subscriptions and accounts change every day. Cost can only be controlled where it is created — at the team and workload level — not by a central gatekeeper after the fact.

FinOps answers this with a simple loop applied continuously: **Inform → Optimize → Operate.**

Phase	Question it answers	Azure & AWS mechanisms
Inform	What are we spending, on what, and who owns it?	Azure Cost Management + Billing / AWS Cost Explorer + CUR; tags & cost-allocation tags; Budgets (both); cost allocation, showback/chargeback
Optimize	Where is the waste and the better deal?	Rightsizing (Azure Advisor / AWS Compute Optimizer); Reservations & Savings Plans (both clouds); storage tiering; autoscale; decommissioning
Operate	How do we make good cost behaviour the default?	Azure Policy / AWS SCPs & Config; budget & anomaly alerts (Cost Management anomalies / AWS Cost Anomaly Detection); dashboards; the monthly review; the FinOps CAB

Correlating cost to your business structure

Savings are only credible when every pound of spend maps to a **business owner and a business outcome**. That mapping is built from your tagging taxonomy and your account/subscription hierarchy — the constructs differ by cloud but the intent is identical:

Azure construct	AWS equivalent	Business meaning it should carry
Management group	**Organization / OU**	Business unit / division / legal entity
Subscription	**Account**	Department, product line, or environment boundary
Resource group	**Tag / stack / application grouping**	A single application, service or workload
Tags (`CostCenter` , `Owner` , `Environment` , `Application` , `BusinessUnit`)	**Cost-allocation tags** (same keys, activated in Billing)	The cost-allocation keys that let finance roll spend up to a P&L line

When this is in place you can produce **showback** (every team sees its own spend) and, when the organisation is ready, **chargeback** (spend is billed back to the cost centre). This is what turns "the cloud bill" into "the cost of *running the customer portal*", which is the language the business and the CAB can actually act on.

> Rule of thumb: if you cannot answer *"which business capability does this resource serve and who pays for it?"* for any line item, your taxonomy — not your spend — is the first thing to fix. (On AWS, remember cost-allocation tags must be **activated** in the Billing console before they appear in Cost Explorer.)

The FinOps personas

FinOps is a team sport. These personas are the standing membership of the **FinOps CAB** (Cost Advisory Board) and the people accountable between meetings. Map each to a named individual in your organisation.

Persona	Owns	Contributes to the CAB
FinOps Lead / Practitioner	The operating model, the monthly review, the savings tracker	Chairs the CAB; presents trends, anomalies and the savings pipeline
Engineering / Platform leads	Architecture and rightsizing decisions; reacting to optimisation actions	Technical feasibility, effort vs saving, reliability trade-offs
Product / Application owners	The unit economics of their service (cost per customer / transaction)	Whether spend is justified by the value delivered; demand forecasts
Finance / FP&A	Budgets, forecasts, cost-centre mapping, ROI sign-off	Validates realised savings, reconciles to the P&L, sets budget targets
Procurement / Commitments	Reservations, Savings Plans, EA/MCA & EDP/PPA terms	Commitment strategy, coverage targets, renewal timing
Security / Governance	Policy guardrails, tagging compliance, allowed SKUs/regions	Confirms optimisation does not breach compliance or resilience
Executive sponsor	The mandate and the cross-department priority	Unblocks decisions; holds business units accountable for their spend

Constant collaboration — the FinOps CAB

The CAB is the heartbeat of the practice. It is **not** a once-a-quarter finance review; it is a short, regular, cross-department forum that keeps cost visible and turns findings into owned actions.

> **Cadence.** Monthly CAB (using the checklist in Part B), with a weekly async waste/anomaly sweep and a quarterly strategic review (commitments, budgets, targets).

Membership. All personas above. Cost is owned across **every department** — engineering, product, finance, procurement, security and the business units that consume the services.

Collaboration loop. Inform the room → agree optimisation actions with named owners and due dates → operate the guardrails so good behaviour becomes the default → report realised savings back to finance and the sponsor.

Anchor it to your critical infrastructure. Prioritise the conversation around the **technical infrastructure that supports your most critical solutions** — the platforms whose availability the business depends on. Optimisation there carries the most ROI *and* the most risk, so it needs engineering, security and finance in the room together.

Proving ROI — methodologies and frameworks to follow

Savings are only real once **finance has signed them off and they show up against a budget**. Use these methodologies consistently:

1. **Inform–Optimize–Operate** (the FinOps Framework) as the overall cadence.
2. **Realised vs projected savings tracking.** Every action moves from *projected* → *committed* → *realised*. Only realised, finance-validated savings count toward ROI.
3. **Unit economics.** Track **cost per business unit** (cost per customer, per transaction, per tenant). Falling total cost while units grow is the strongest ROI signal; it proves efficiency, not just cuts.
4. **Commitment-based discounting.** Reservations and Savings Plans for steady-state workloads on either cloud; target **60–80% coverage** with high utilisation.
5. **Rate vs usage optimisation.** Separate *paying less for the same thing* (rates: commitments, tiering, region) from *using less* (usage: rightsizing, autoscale, decommissioning). Report both.
6. **Avoid-vs-realise.** Distinguish **cost avoidance** (spend you prevented) from **realised savings** (spend you removed). Both matter; label them so the ROI story is honest.

ROI formula the CAB should report each month: **(realised savings + validated cost avoidance) ÷ cost of running the FinOps practice.** A healthy practice returns many multiples of its own cost.

Why it matters — the evidence

FinOps is not cost-cutting theatre; the difference between an organisation that runs a disciplined practice and one that does not shows up directly in the P&L, in delivery speed and in engineering culture. The figures below are widely reported across the industry (FinOps Foundation *State of FinOps*, the major cloud providers' Well-Architected cost guidance, and analyst research from Gartner and Flexera). Treat them as the order-of-magnitude case for doing this properly.

Organisations that run FinOps well vs those that don't

Dimension	Without a disciplined practice	With a mature FinOps practice
Wasted cloud spend	Industry surveys consistently report that roughly **30% of cloud spend is wasted** — idle, oversized, untagged or forgotten resources.	Mature practices routinely **recover 20–35% of spend** in the first year and hold it down as consumption grows.
Cost visibility	Spend surfaces weeks later on an invoice; **no clear owner** for most line items.	Near-real-time **showback/chargeback**; the large majority of spend is tagged and mapped to a business owner.
Forecast accuracy	Budgets blown without warning; finance and engineering work from different numbers.	Forecasts within a **few percent**; budget alerts catch overruns before month-end.
Commitment coverage	Steady-state workloads paid for **on-demand** — effectively a 20–60% premium.	**60–80% reservation/Savings Plan coverage** at high utilisation, locking in the discounted rate.
Engineering behaviour	Cost is "someone else's problem"; the same waste recurs every month.	Cost is an **architectural attribute** owned by the team that creates it; waste is engineered out at source.
Speed of decision	Optimisation stalls in finance-vs-engineering debate.	A standing **CAB** turns findings into owned actions in days, not quarters.

The headline most leaders care about: every **£1 invested** in a functioning FinOps practice typically returns **several pounds** in realised savings and validated cost avoidance — and unlike a one-off cost-cutting exercise, that return **compounds**, because the guardrails stop the waste coming back.

Real-world scenarios where it proves critical

These are representative of patterns seen repeatedly across Azure and AWS estates. They illustrate *why* the operating model — not just the tooling — is what delivers the result.

> **The runaway non-production estate.** A scale-up's dev/test subscriptions (and AWS sandbox accounts) quietly grew to rival production. A monthly waste sweep plus an auto-shutdown/scheduler policy on non-prod removed **~28% of that estate's spend in one cycle** — and the policy stopped it ever returning. *Lesson: a recurring cadence beats a one-off clean-up.*

Reservations bought blind — then bought right. A team bought three-year commitments against a workload they re-architected two months later, stranding it. Bringing **procurement and engineering into the same CAB** meant the next round was sized against the roadmap, lifting coverage to ~75% with near-full utilisation and avoiding a repeat write-off. *Lesson: commitments are a cross-department decision, not a finance one.*

The untraceable bill. A professional-services firm couldn't tell which client a third of its cloud spend served, so it couldn't bill it back. Fixing the **tagging taxonomy and cost allocation first** (activating cost-allocation tags on AWS, enforcing tag policy on Azure) turned an opaque invoice into per-client showback, recovered previously unbillable cost, and made margins defensible in front of the board. *Lesson: you cannot optimise — or charge back — what you cannot attribute.*

Protecting critical infrastructure under cost pressure. During a cost-down mandate, a blanket "shrink everything" instruction nearly downsized the platform behind a revenue-critical customer portal. Because **security and engineering sat in the CAB**, the cut was redirected to genuinely idle resources while the critical path kept its resilience. *Lesson: optimisation governed by the right people protects the business; optimisation done in a spreadsheet endangers it.*

Anomaly caught in days, not on the invoice. A mis-configured log ingestion (Azure Log Analytics / AWS CloudWatch Logs) started adding thousands per week. An anomaly alert reviewed in the weekly async sweep caught it within days — a problem that, without the cadence, would have surfaced only on the following month's bill. *Lesson: the operate phase is what converts visibility into prevented loss.*

The through-line in every scenario: tooling finds the number, but the **operating model — personas, cadence and the CAB — is what turns the number into delivered, defensible ROI**. Organisations that institutionalise that loop don't just spend less; they ship faster and make cloud cost a managed, predictable input to the business rather than a quarterly surprise.

Part B — The monthly review checklist

Print it, or copy it into your ticketing tool. Aim to complete the whole review in 30–45 minutes each month, in the FinOps CAB. Each item pairs the **Azure** and **AWS** action — run the column(s) for the cloud(s) you use.

Before the meeting (owner: FinOps lead)

[ ] Export last month's costs — **Azure:** Cost Management (CSV, grouped by service and resource group). **AWS:** Cost Explorer (or a CUR/Athena query), grouped by service and linked account.

- [ ] Pull the **budget vs actual** for every subscription/account and resource group.
- [ ] Run a **resource inventory** (Azure: `az resource list`; AWS: `aws resource-groups` / a Config aggregator or CUR query) and diff it against last month.
- [ ] Note any **alerts** that fired — budget alerts and anomaly alerts (Cost Management anomalies / AWS Cost Anomaly Detection) — since the last review.
- [ ] Pull **AI/token usage** per model & deployment (Azure OpenAI / AI Foundry metrics; Amazon Bedrock model-invocation metrics in CloudWatch) so AI spend enters the review as measured data, not a surprise.

1. Spend overview (5 min)

- [ ] Total spend this month vs last month — record the **% change**.
- [ ] Spend vs **budget** — are we over, under, or on track?
- [ ] Review **budget alert thresholds** (e.g. **50% / 80% / 100%** of budget) — which fired, and were the right people notified?
- [ ] **Forecast** for end of next month — any concern?

2. Top drivers (5 min)

- [ ] Review the **top 10 services/resources** by cost (per cloud).
- [ ] Flag anything that grew **>20%** month-over-month.
- [ ] Review any **anomaly alerts** (Azure Cost Management / AWS Cost Anomaly Detection) — root cause and action for each.
- [ ] Confirm each top driver is **expected** and has a clear owner.

3. Waste sweep (10 min)

- [ ] **Orphaned disks:** unattached managed disks (Azure) / unattached **EBS volumes** — state `available` (AWS).

[ ] **Idle IPs:** unassociated public IPs (Azure) / unassociated **Elastic IPs** (AWS).

[ ] **Stopped-but-billing compute:** stopped (not deallocated) VMs still paying for disks (Azure) / stopped **EC2** instances still paying for EBS (AWS).

[ ] **Idle managed services:** empty/under-used App Service plans (Azure) / idle **load balancers**, empty **Auto Scaling groups**, idle **NAT gateways** (AWS).

[ ] **Orphaned artifacts:** orphaned NICs, snapshots, old backups (Azure) / orphaned **ENIs**, unused **snapshots & AMIs** (AWS).

[ ] **Non-prod out of hours:** dev/test/sandbox resources running nights and weekends (both clouds) — schedule start/stop.

4. Optimization opportunities (10 min)

[ ] **Rightsizing:** any VM/SQL/AKS (Azure) or EC2/RDS/EKS (AWS) under 20% utilization? Use Azure Advisor / AWS Compute Optimizer.

[ ] **Reservations / Savings Plans:** coverage % and utilization on both clouds — buy, exchange, or cancel? Cover the steady-state baseline, not the peak.

[ ] **Storage tiering:** Azure Blob (Hot→Cool→Archive) / AWS S3 storage classes + lifecycle rules; and redundancy (Azure GRS→LRS / AWS cross-region replication) where not needed.

[ ] **Monitoring/logs:** Azure Log Analytics / Sentinel and AWS CloudWatch Logs — retention and ingestion volume.

5. AI & token spend (5 min)

AI is the fastest-growing line on many cloud bills, and it breaks classic cost checks — the money moves through **tokens**, not resources. Measure it, monitor the trend, and watch the token-level signals that dollar-only reviews miss. Applies to **Azure OpenAI / AI Foundry** and **Amazon Bedrock**.

[ ] **Measure — per model & deployment.** Confirm token usage and cost are captured for every deployment (Azure OpenAI / AI Foundry usage metrics; Bedrock model-invocation metrics in CloudWatch). Any deployment you can't see is unmanaged spend.

- [ ] **Monitor usage — the trend.** Tokens/day and spend/day per deployment; flag anything up **>20%** month-over-month, and any **new** model or deployment that appeared.
- [ ] **Output bloat.** Track the **output:input token ratio** — output tokens cost roughly **3–5× input**, so a rising ratio is a verbosity regression worth catching early.
- [ ] **Rate shift.** Compare blended **$/1K tokens** vs last month — a jump usually means a **model-mix change** (traffic moved to a pricier model).
- [ ] **Waste signals.** Low **prompt-cache hit rate** (you're re-sending context), missing `max_tokens` caps on production calls, and **idle/zombie deployments** or under-used **provisioned throughput (PTU on Azure / Provisioned Throughput on Bedrock)**.
- [ ] **Attribute it.** Is AI/token spend tagged and mapped to a **feature and owner**, or sitting in an unallocated bucket? Drive that bucket down like any other untagged spend.

6. Governance (5 min)

- [ ] **Untagged spend** — what % is missing `CostCenter` / `Owner` / `Environment`? (On AWS, confirm cost-allocation tags are **activated** in Billing.)
- [ ] Any new resources created **outside policy** — disallowed SKU/region/instance type (Azure Policy / AWS SCPs & Config)?
- [ ] **Budgets** still set correctly for new resource groups / accounts?

7. Actions & accountability (5 min)

- [ ] Log each action with an **owner** and a **due date**.
- [ ] Update the **savings tracker** (realized vs projected).
- [ ] Schedule any **decommissioning** approved this month.

Monthly scorecard

Metric	Last month	This month	Target
Total spend			
Spend vs budget			≤ 100%
Untagged spend %			< 5%
Reservation / Savings Plan coverage %			60–80%
AI / token spend (MoM change)			≤ +10%
Prompt-cache hit rate			≥ 50%
Realized savings (MTD)			

CloudFinOpsKit — Cloud Cost Review Checklist (Azure & AWS). Use and adapt freely for your own and client work.