CHAPTER 6 / 3 MINUTE BRIEF

Built-In Tools

Read the full chapter → / Read this brief online →

The main idea

It is possible to open three perfectly useful tools and receive what look like contradictory instructions. A machine-sizing report recommends a smaller instance, a budget alert says the project is spending too much, and the autoscaler wants to create more instances. Before we schedule a meeting to determine which tool has failed, it would help to know what each one was asked.

The sizing report summarizes at least a week of one machine's use, the alert compares charges that typically arrive within a day with a monthly threshold, and the autoscaler responds to the last few minutes of CPU. All three could be right. The difficult part is deciding whether today's load is temporary, whether the proposed smaller machine can handle the work we have promised, and whether somebody has already agreed to pay for it.

Explore this section in the full chapter →

Keep inventory, use, and cost distinguishable

When I look at a service, I want to move among three views without losing track of what each tells me. Inventory shows what exists and where; telemetry shows what it did during an interval; billing shows what was charged under the commercial arrangement. A resource that disappeared yesterday may be absent from today's inventory and still account for a substantial part of this month's bill. A workload that stopped running may have left billed resources behind, while an apparently unused resource may still be needed for recovery.

The distinction seems obvious until someone joins the three views, drops the records that do not match, and presents the result as a complete account. We need the unmatched records precisely because they may explain what we have failed to understand.

Explore this section in the full chapter →

A recommendation is the beginning of an experiment

Before I accepted or dismissed one, I would want to connect the recommendation and its evidence date to the workload's next important event. Has the application changed? Is a migration about to end? Does the proposed smaller configuration affect a recovery obligation? I would also ask Finance what the reduction means under the commercial arrangement. Freeing capacity inside an existing commitment can be worthwhile even when it does not lower this month's cash expenditure.

For a stable stateless service, a representative subset may give us a straightforward experiment: agree on completion and latency criteria, make the change through the operating team's process, compare with the baseline, and preserve a feasible route back. A data service may need more patience, because ordinary traffic does not exercise the maintenance or recovery work that justified the allocation.

Suppose a smaller configuration serves requests successfully but cannot complete a required rebuild. We have discovered that the proposed saving changes the service being supplied. The business may be willing to accept that change, but I would not present it as an equivalent service for less money.

This is also why I want a useful explanation when a recommendation is deferred. "Awaiting a representative peak" gives us a next step, as does "retained for this recovery obligation" or "superseded by the planned migration." "Engineering refused" mostly tells me that we have run out of patience with each other. The capacity team's insistence deserves no more authority than Engineering's refusal unless somebody can explain the work.

Explore this section in the full chapter →

Alerts, caps, and the person who receives them

An Azure budget notifies people when actual or forecast cost crosses a threshold; in Microsoft's words, "Resources aren't affected, and your consumption isn't stopped." At subscription and resource-group scopes, an action group can pass the notification to automa-

tion such as a function or webhook, which needs its own design: which work may stop, who authorizes it, and how the service resumes.[1]

Cost data typically arrives 8 to 24 hours after use, budgets are evaluated every 24 hours, and email follows within about an hour, so an overrun can be nearly two days old when someone reads about it. An application that needs an immediate limit has to enforce it in its own request or admission path, reconcile it with billing, and decide what happens when the limit is reached.

Explore this section in the full chapter →

[1] Microsoft Learn, “Tutorial - Create and manage budgets”, checked October 4, 2026. Notification without stopping consumption; data and evaluation timing; actual cost including purchases, with usage-only filters; action groups at subscription and resource-group scopes.