AUTOSCALING IS AN AUTHORITY SYSTEM, NOT A CAPACITY SYSTEM

Every scaling policy delegates decision-making authority to the execution plane.
Most teams have never governed that delegation.

THE CAPACITY FRAMING IS WRONG

HOW MOST TEAMS FRAME IT

- CPU hits threshold
- Replicas scale up
- Problem solved

CAPACITY QUESTION

THE REAL QUESTION

What is this system permitted to do without asking a human?

That is a governance question.
Capacity is just the output.

AUTHORITY QUESTION

SILENT DELEGATION

No architect signs a document stating:

"The execution plane may increase application capacity by 500% without human review."

Yet that is exactly what many autoscaling policies authorize.

The delegation happened when the configuration was written.
The authority persisted long after the original decision-maker stopped thinking about it.

SCALING DIVERGENCE

FAILURE PATTERN — #152 OPERATIONAL AUTHORITY BOUNDARY

Autoscaling behavior remains technically correct relative to configuration while becoming operationally incorrect relative to current intent.

THE SCENARIO

LAUNCH	HPA target set at 70% CPU. Bottleneck: compute.
9 MONTHS LATER	Bottleneck shifts to database I/O. CPU stays low.
RESULT	Autoscaler holds steady. Latency spikes. CPU looks fine.

The workload changed. The authority logic did not.

THREE FAILURE MODES

01 — CEILING BLINDNESS

Max replicas derived from the original spec, not capacity headroom. Autoscaler hits the ceiling under load. Nobody knows if the ceiling is a safety boundary or an artifact.

02 — FLOOR DRIFT

Min replicas reflect day-one sizing. Application has grown 3×. During scale-down, the autoscaler holds at a floor that no longer matches minimum viable capacity.

03 — MULTI-CONTROLLER CONFLICT

HPA and VPA run on the same workload without coordination mode. VPA changes requests. HPA recalculates against modified values. Scheduler behavior becomes unpredictable.

AUTOSCALING AUTHORITY GOVERNANCE

Four properties a functioning autoscaling authority system requires:

01 — DECLARED INTENT

Configuration values documented as decisions — what load profile, what bottleneck, what was assumed.

02 — AUTHORITY OWNERSHIP

A named team or role accountable for this policy. If nobody owns it, nobody reviews it.

03 — CHANGE COUPLING

Workload changes trigger scaling policy review — not discovered six months later in an incident.

04 — BEHAVIORAL AUDIT

Periodic review of actual scaling events against declared intent. Not monitoring. Deliberate comparison.

AUTOSCALING AUTHORITY AUDIT

Three questions every autoscaling configuration in your estate should be able to answer:

Q1 **What workload model does this configuration assume?**

Not the current values — the load profile, bottleneck type, and traffic pattern used to derive them.

Q2 **When was this last validated against actual behavior?**

Not last modified. Last validated — someone compared scaling events against declared intent.

Q3 **Who is accountable if this damages the application under load?**

If the answer is nobody, authority ownership is undefined. The execution plane has no principal.

THE ARCHITECT'S VERDICT

Every autoscaling system either has explicit authority or it has defaults.

Defaults are simply authority decisions that survived long enough to become invisible.