The Simplicity Code: Designing Intelligent Architecture Without the Buzzwords

Trade buzzword stacks for operational clarity, simpler boundaries, and faster change


It's YOUR fault!!
Did you even test this?!
I'm so sick of this code!

We’re shipping more complexity than capability

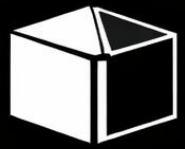
Kubernetes, CQRS, event streaming, service meshes: great tools... wrong default

Clarity is an availability feature

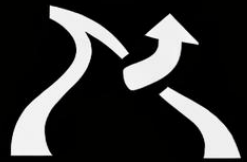
The Simplicity Code: Agenda

The 3 Filters:

2 AM Test



Half-Rule



Primary Path First

Then, Apply to 4 Decisions:

Boundaries

Data

Cloud

Scaling


Ugh... I've gotta fix the bug!
BEEP! BEEP!
12
3
9
2:07
URGENT!
BUG ALERT!

The Rubric

Detect: Do we know something’s wrong quickly?

The Rubric

Localize: Can we pinpoint where and why?

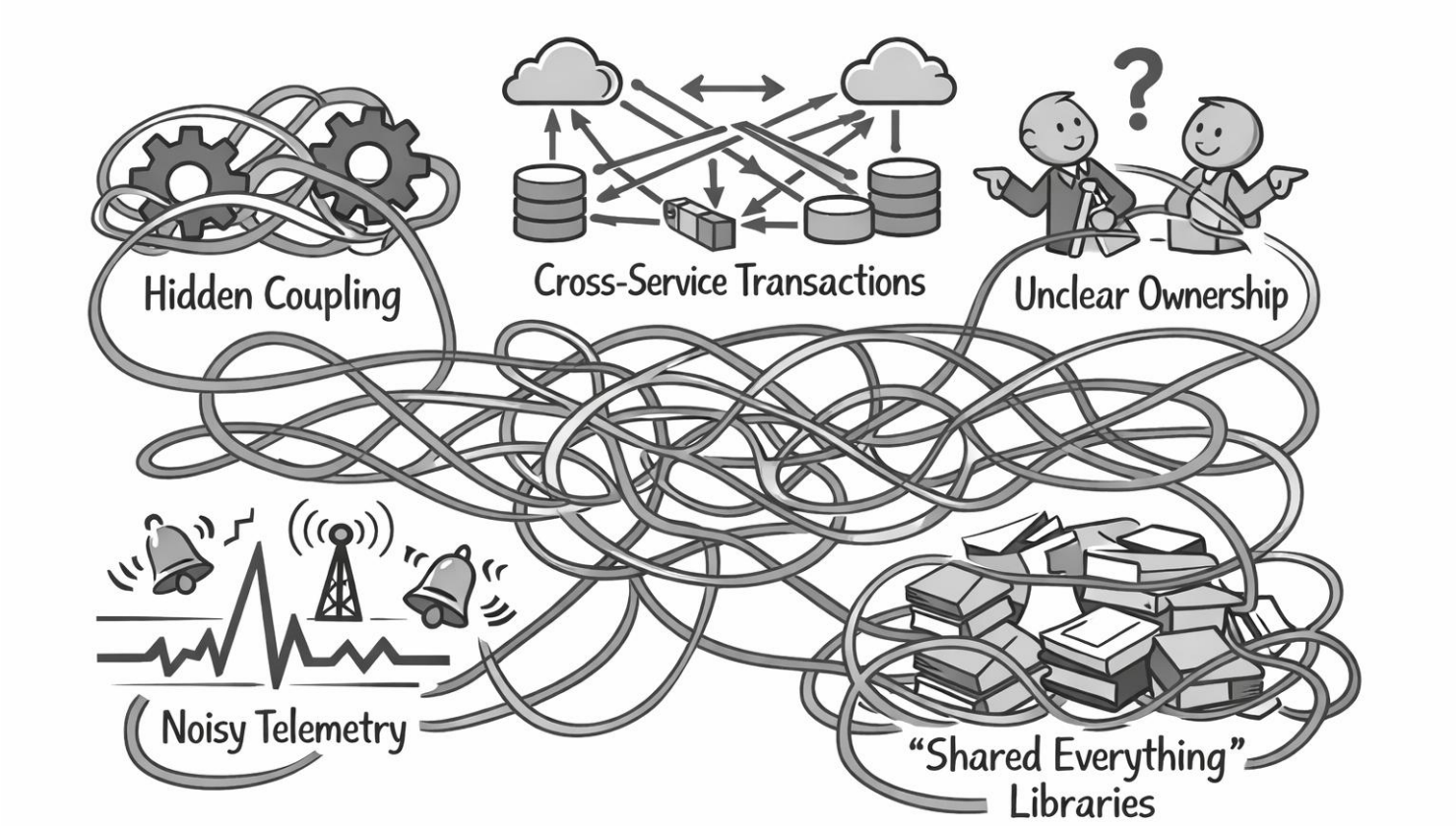
The Rubric

Change: Can we modify with confidence?

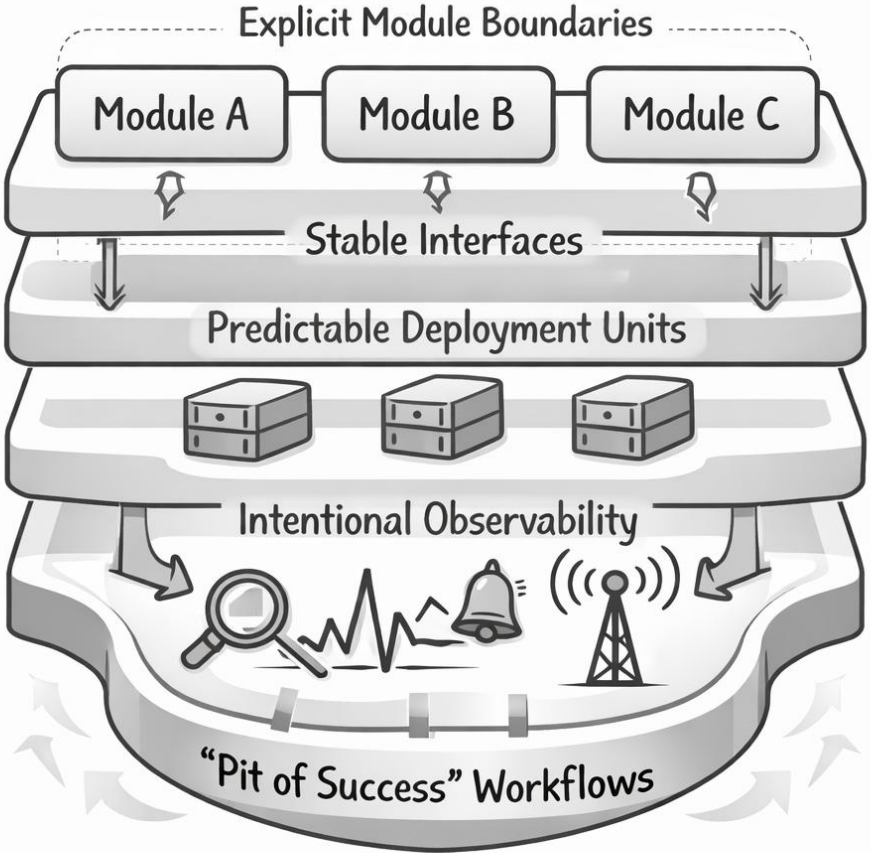
The Rubric

Validate: Can we prove it’s fixed?

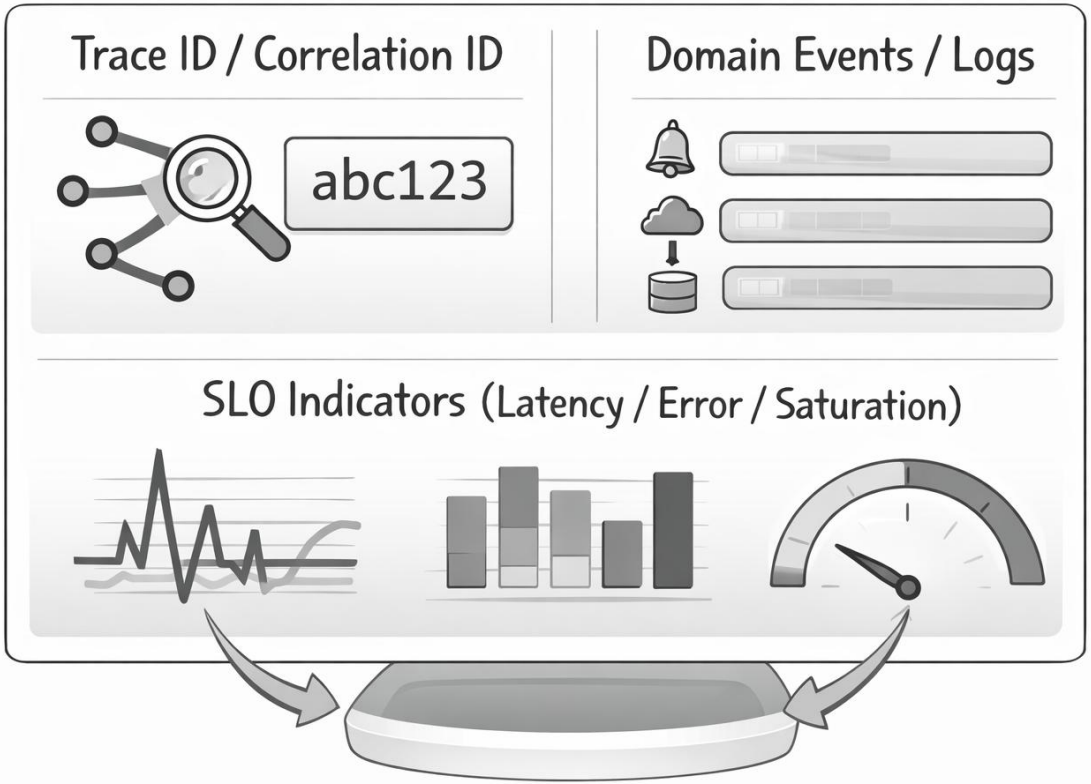
Architecture choices that fail the 2 AM Test



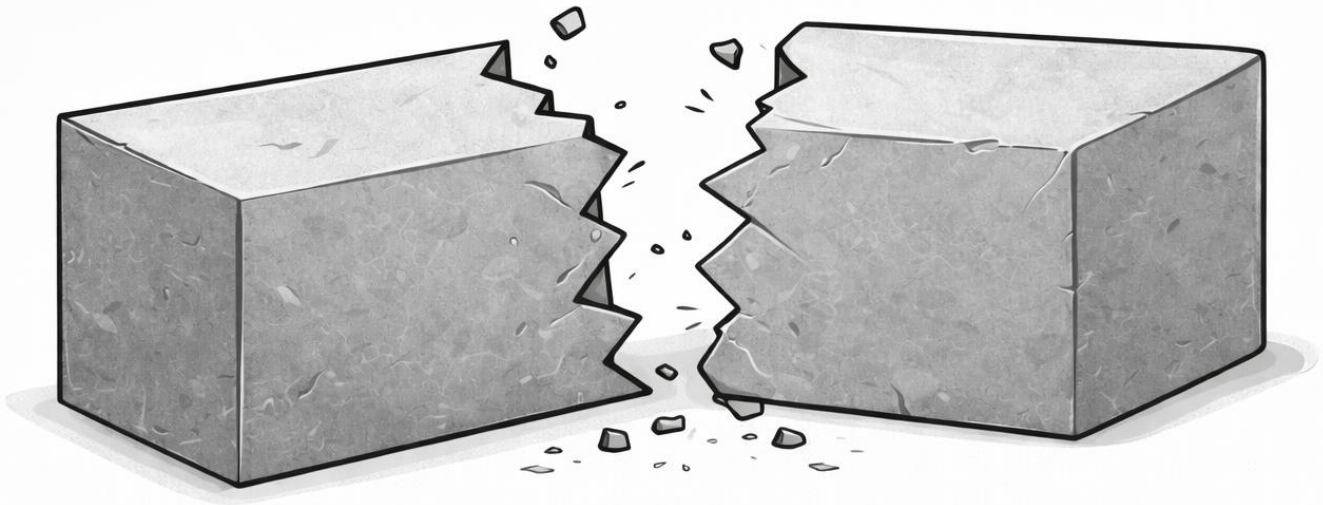
What passes the 2 AM Test



Practical instrumentation: signal over volume



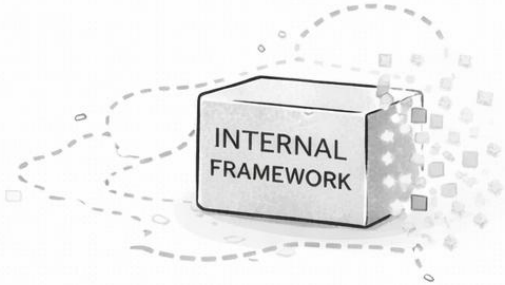
The Half-Rule



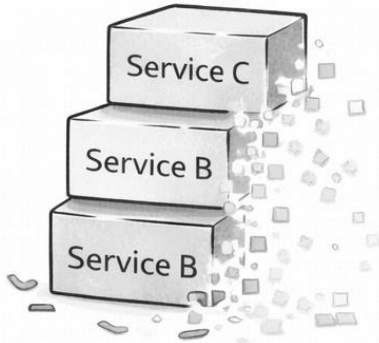
Build half of what you think you need;
treat the rest as hypothesis.

Every abstraction spends budget:

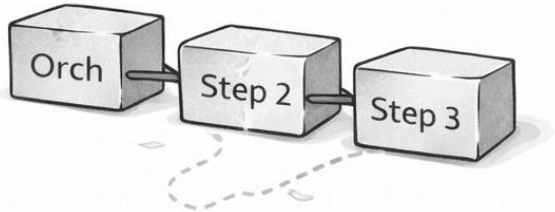
cognition, ops, latency, tooling, coordination.



Kill “frameworks” that exist only internally.

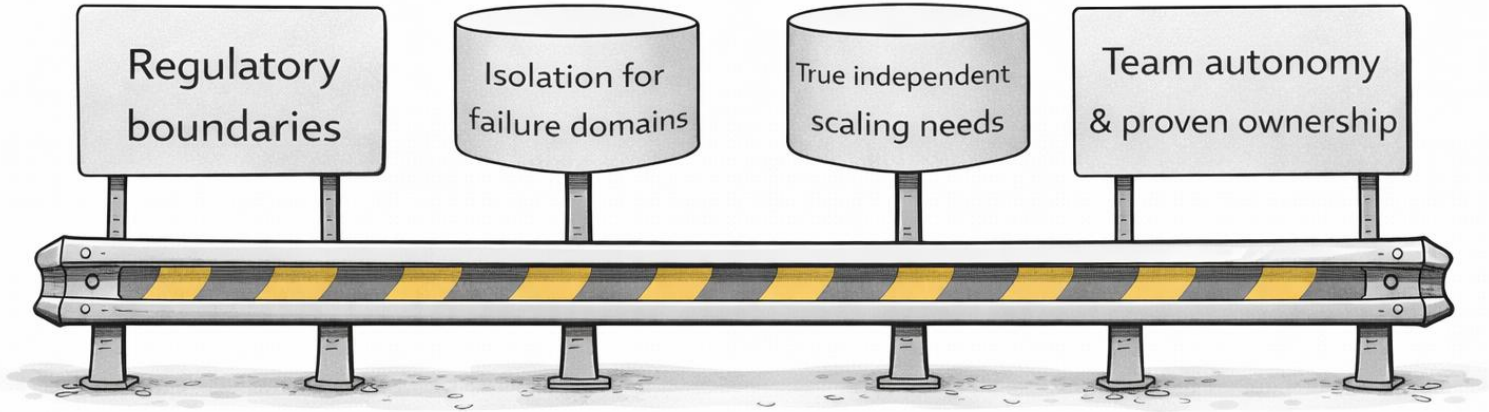


Collapse service count when boundaries are unclear.



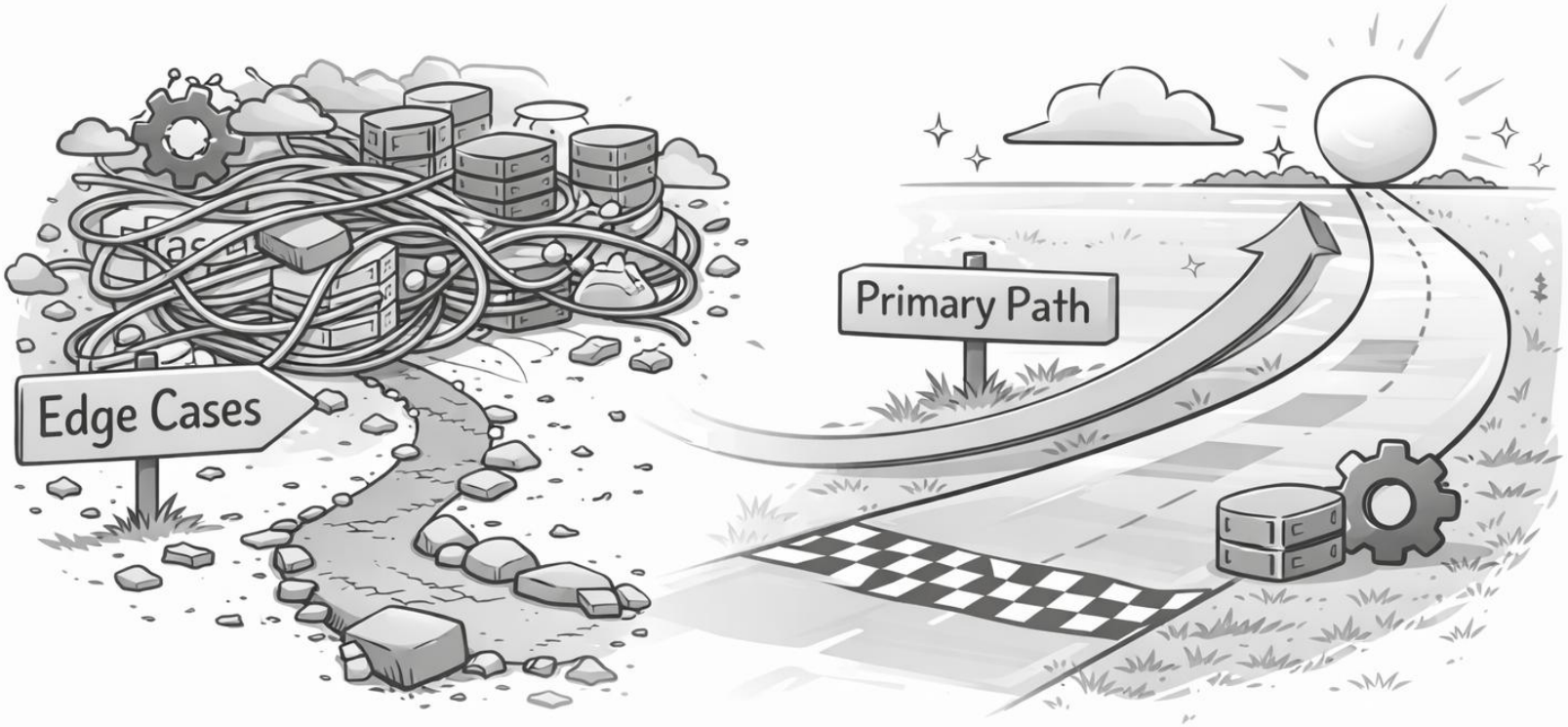
Replace orchestration with sequencing in-process when load doesnt justify distribution.

Subtraction Tactics.



When ***not*** to subtract.

Primary Path First



Make the 80% path boring, resilient,
and fast to understand.

Contain *Edge-Case* Complexity.

The Simplicity Code:

The 3 Filters:

2 AM Test

Half-Rule

Primary Path First

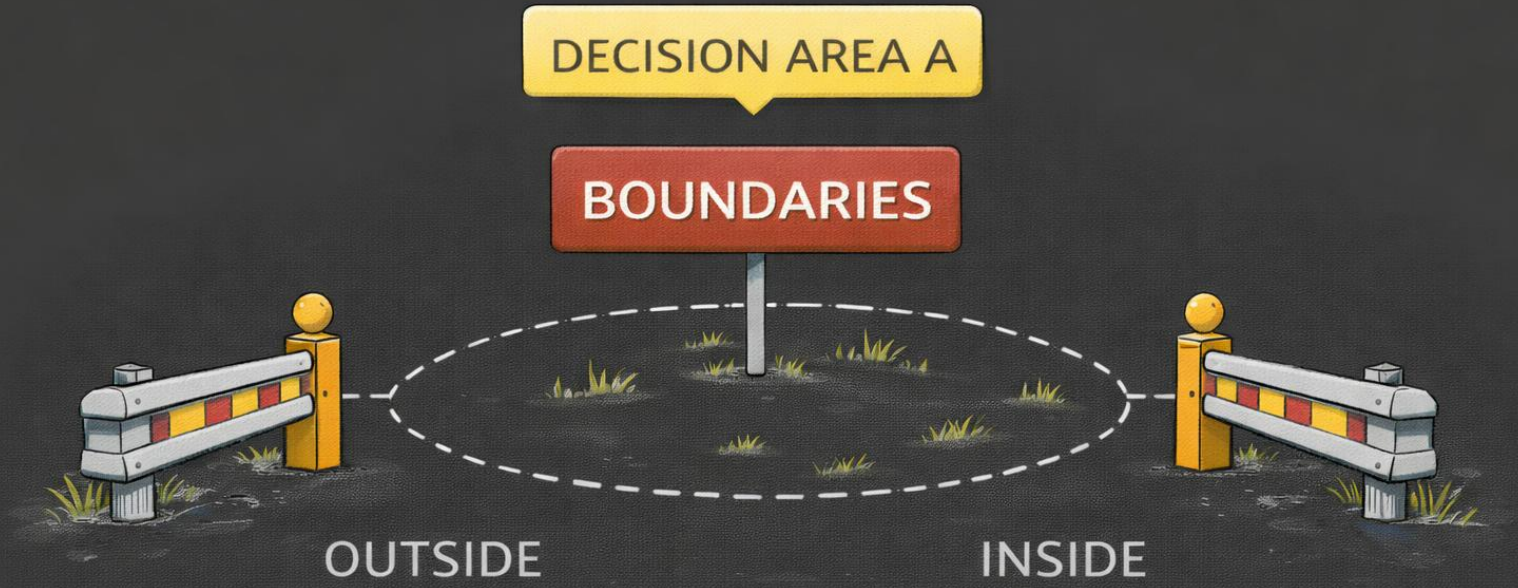
Then, Apply to 4 Decisions:

Boundaries

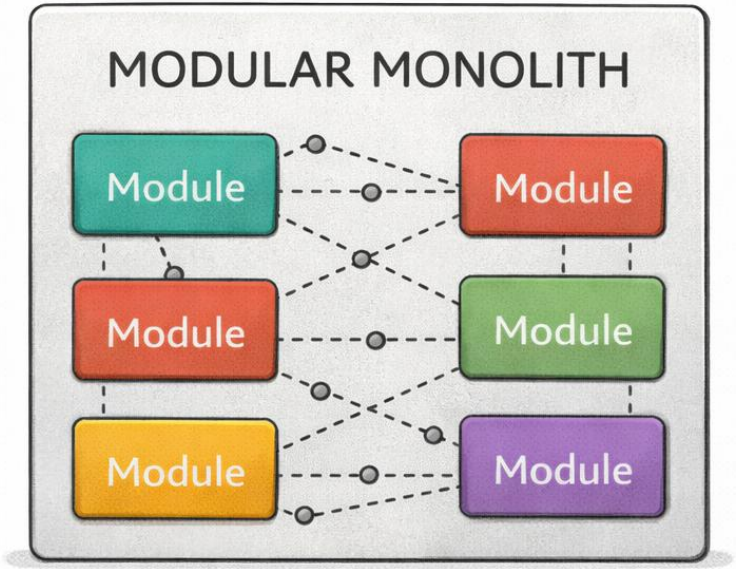
Data

Cloud

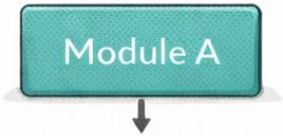
Scaling



Decision Area: *Boundaries*.

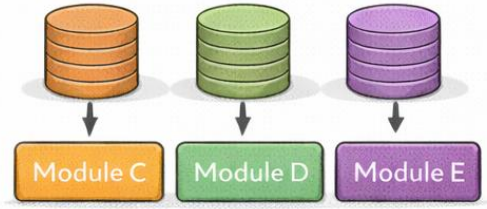


Microservices are a deployment choice, *not* a design starting point.

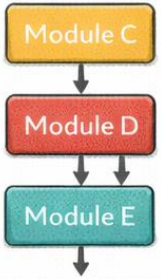


Explicit module API

Internal encapsulation

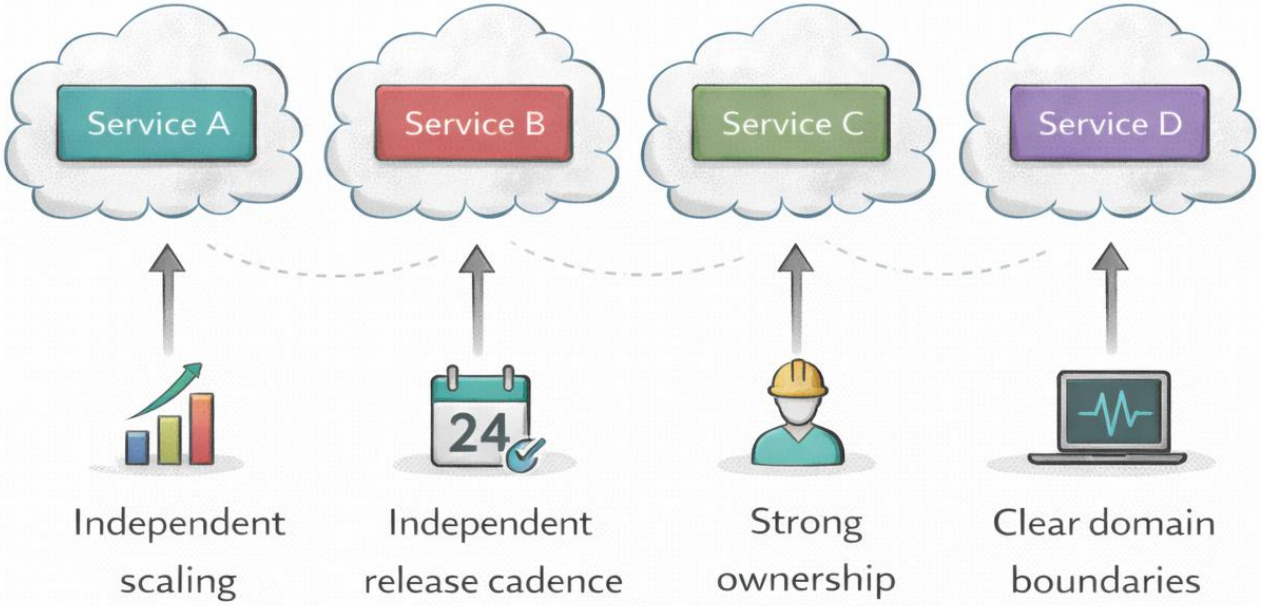


Separate data ownership (if feasible)

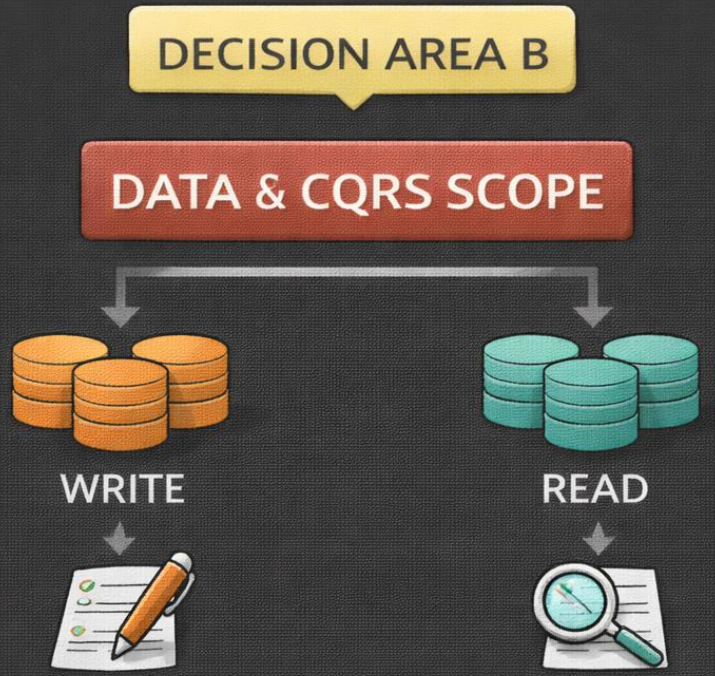


Clear dependency direction

Modular monolith: what it means (*practically*).

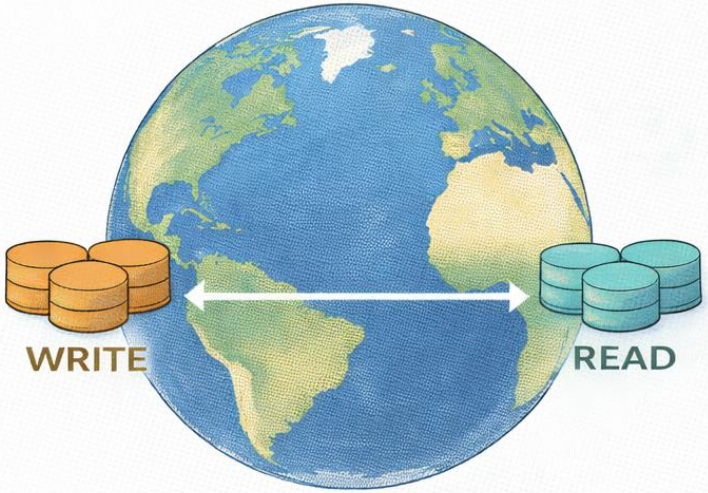


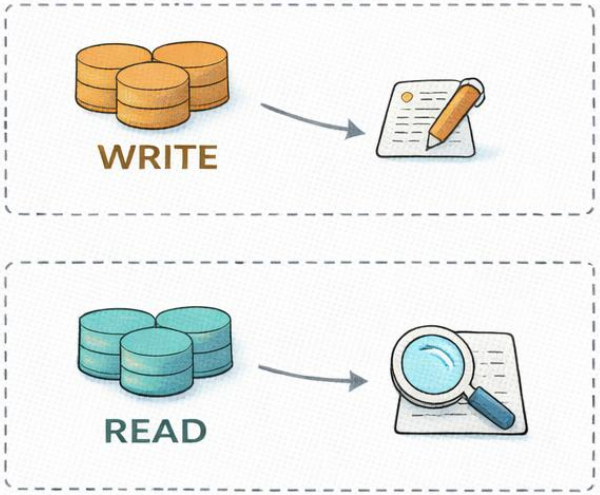
Microservices: earned, not *declared*.



Decision Area: *Data and CQRS scope.*

CQRS: stop making it global





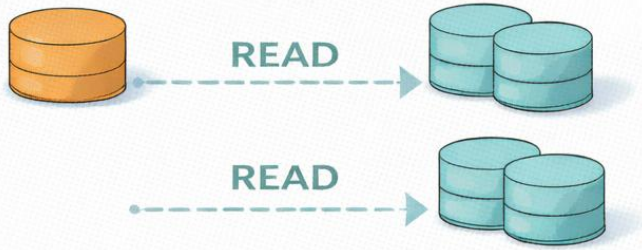
CQRS is powerful when localized;
harmful when it becomes an ideology.

evolutionary steps

Start

single model,
simple read optimizations,
cache where proven.

Evolve

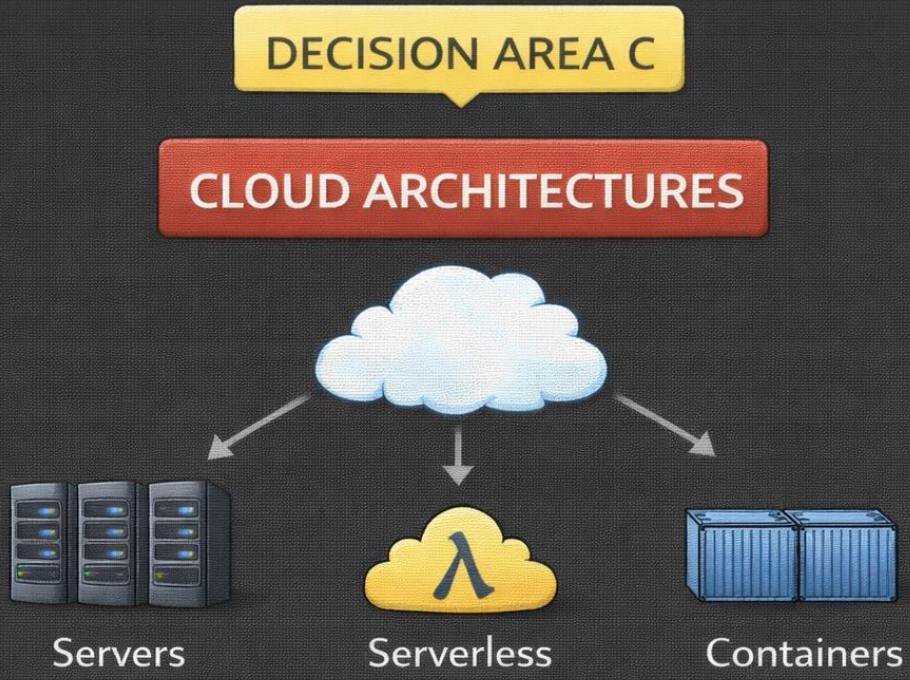


targeted read models
on the hottest queries.

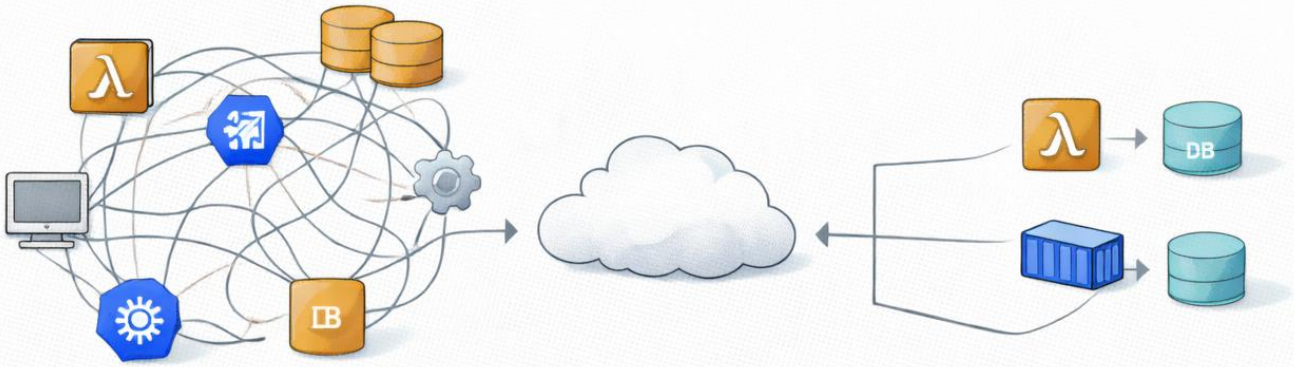
A simpler default.

boundary lines

Consistency boundaries you can explain at 2 AM.

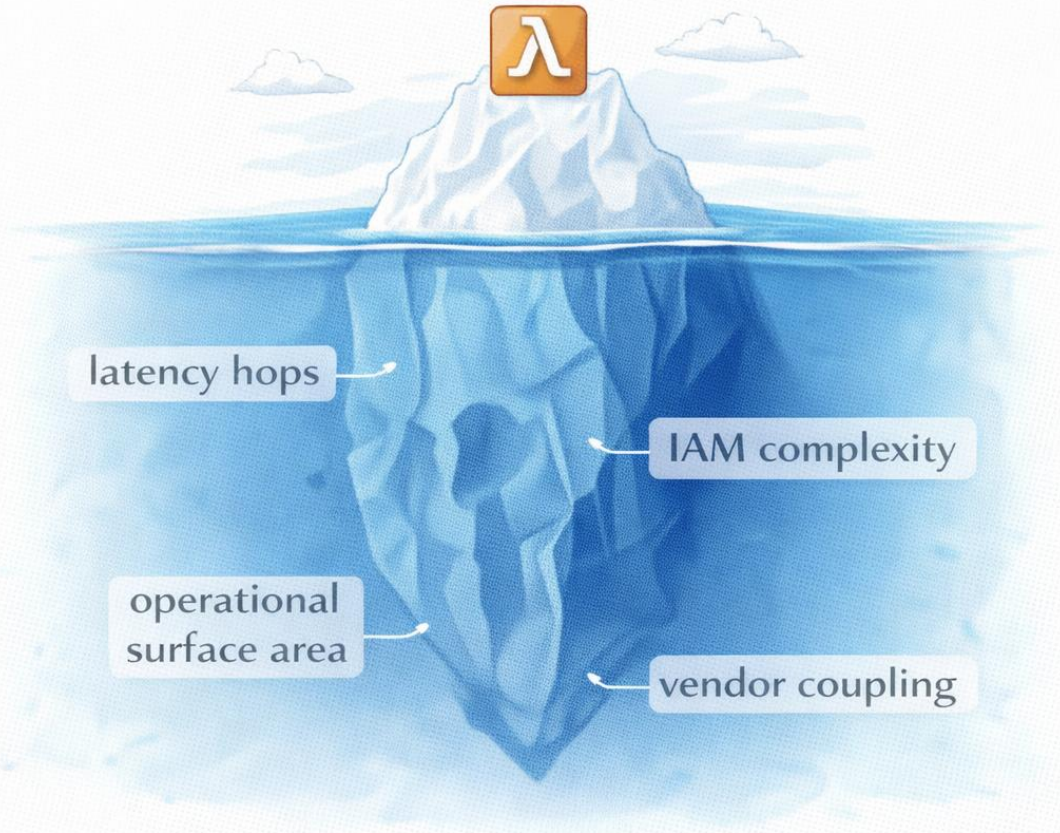


Decision Area: *Cloud architectures.*

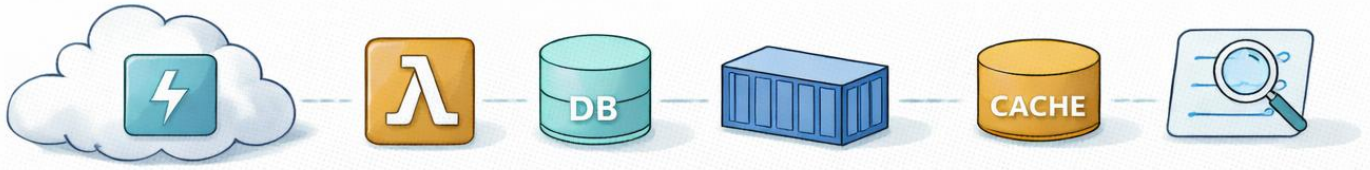


Use cloud to remove toil, not to multiply moving parts.

Cloud simplicity.



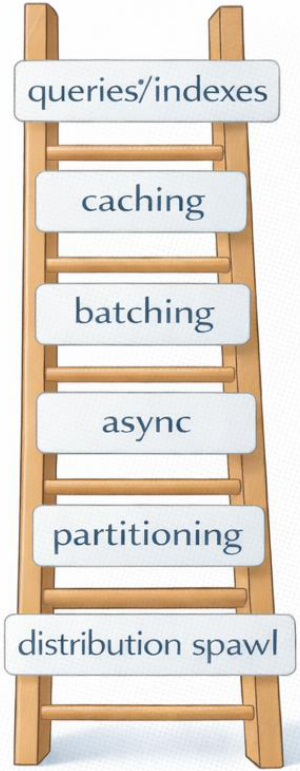
‘Managed service’ is not free.



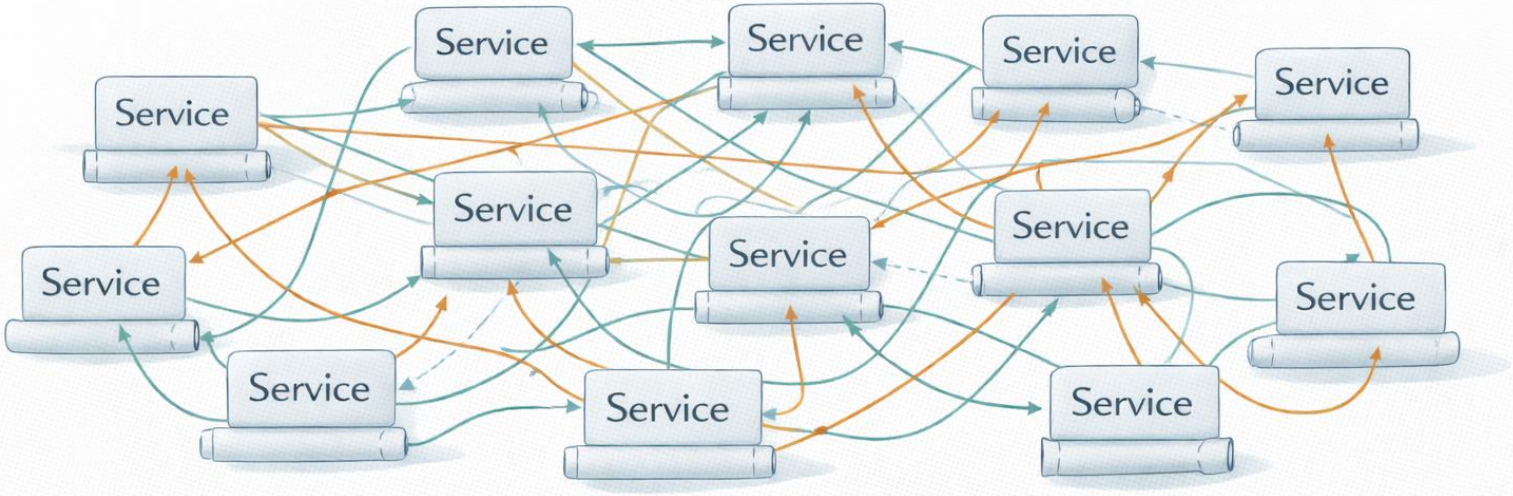
Add services only when a constraint forces it.

A sane default reference stack.

Decision Area: Scaling smarter.



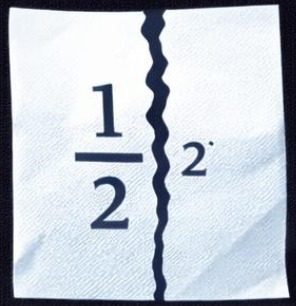
Scaling smarter, not bigger



We broke it into 20 services and got slower.

The anti-pattern: scaling by decomposition

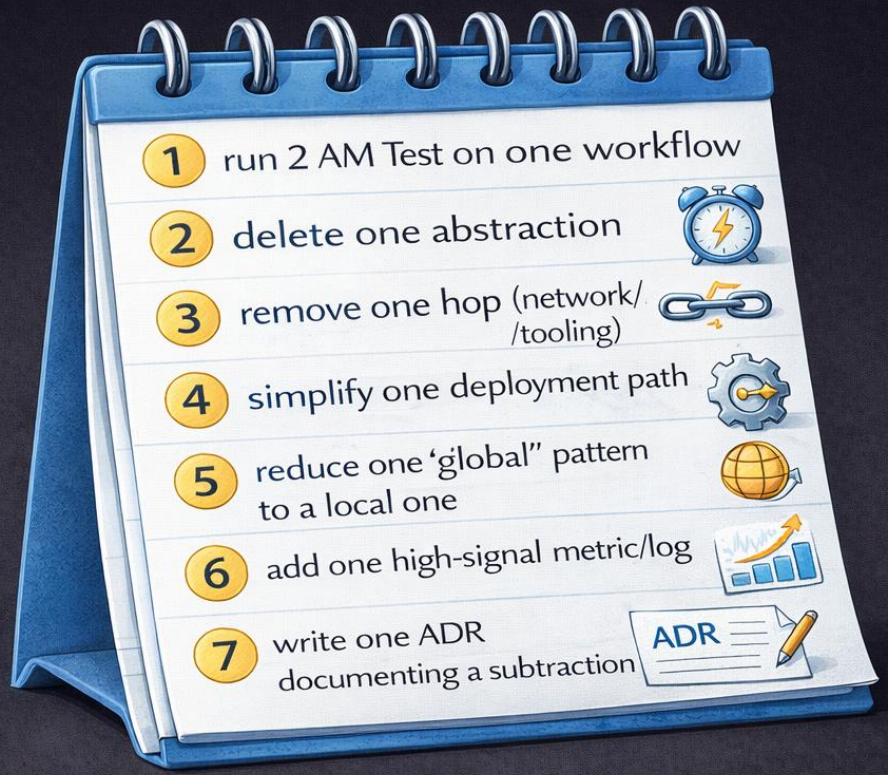
2 AM Test



Half-Rule

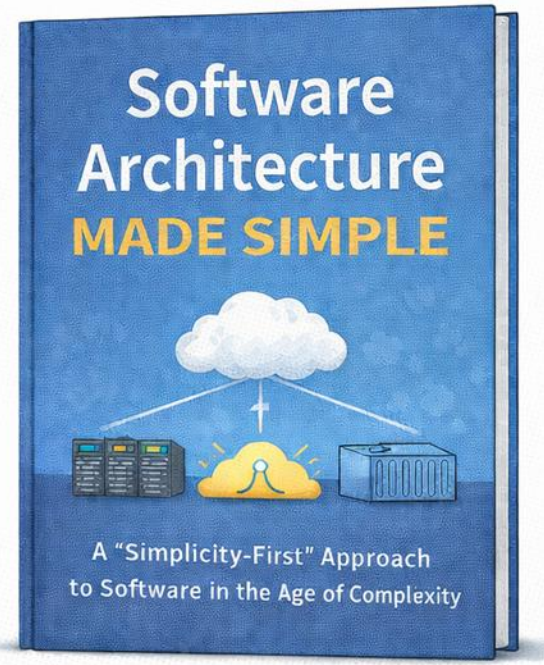
Primary Path First

The Simplicity Code



A 7-day architecture reset.

Coming Fall 2026!



How do software developers say thank you?

With a heartfelt commit.

```
# Thank you!
> git commit

❤️ [main ba5e704] Thank you so
   much for attending!
 1 file changed, 2 insertions(+)
```