



Introduction to Microservices

UQ Architecture - April 2024



Damian MacLennan

Consultant CTO and Software Trainer

damian@damianm.com

damianm.com

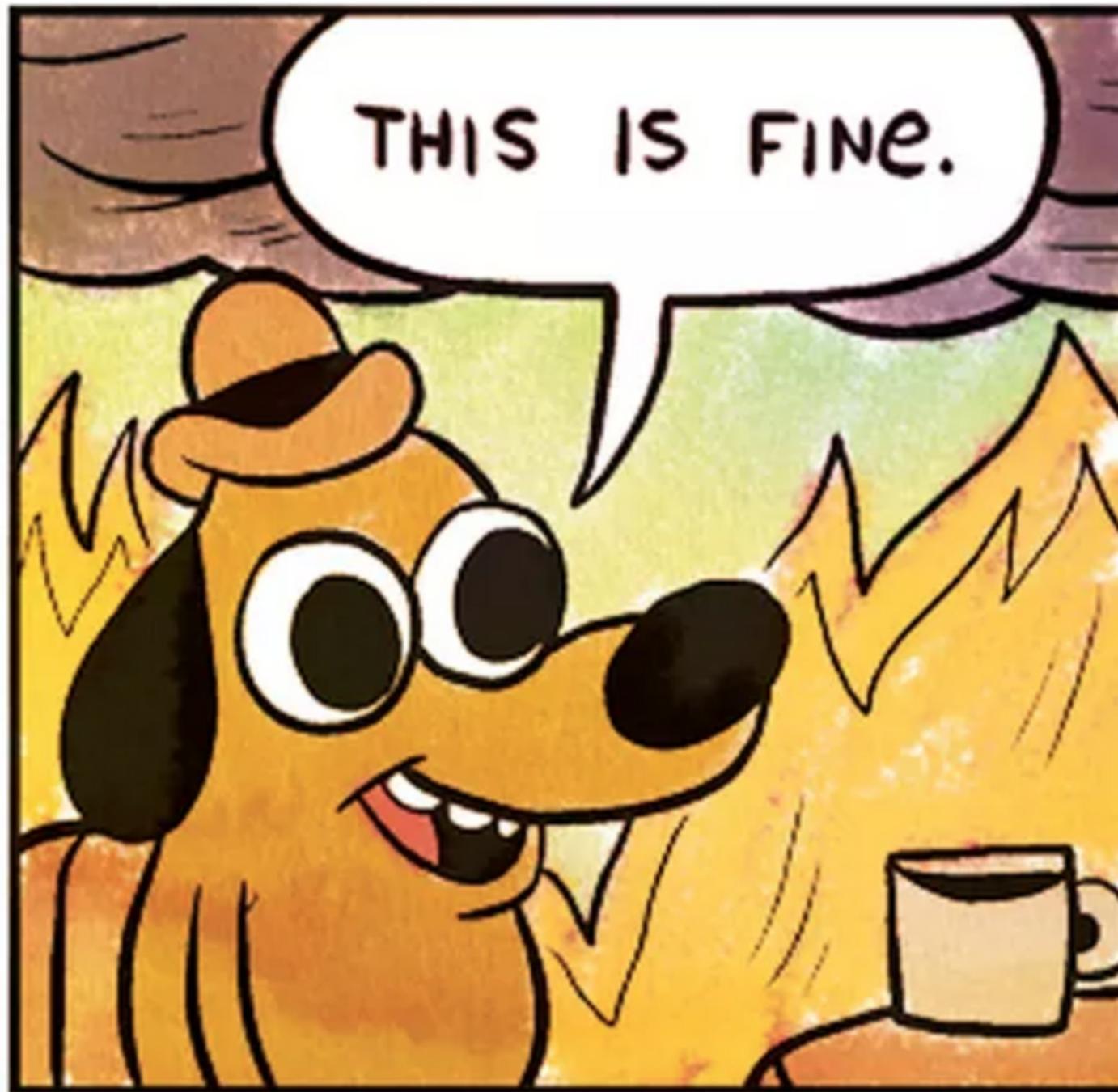
stackmechanics.com



Agenda

- 1 Background
- 2 History
- 3 Where things go wrong
- 4 Modelling
- 5 Concepts
- 6 Conclusions

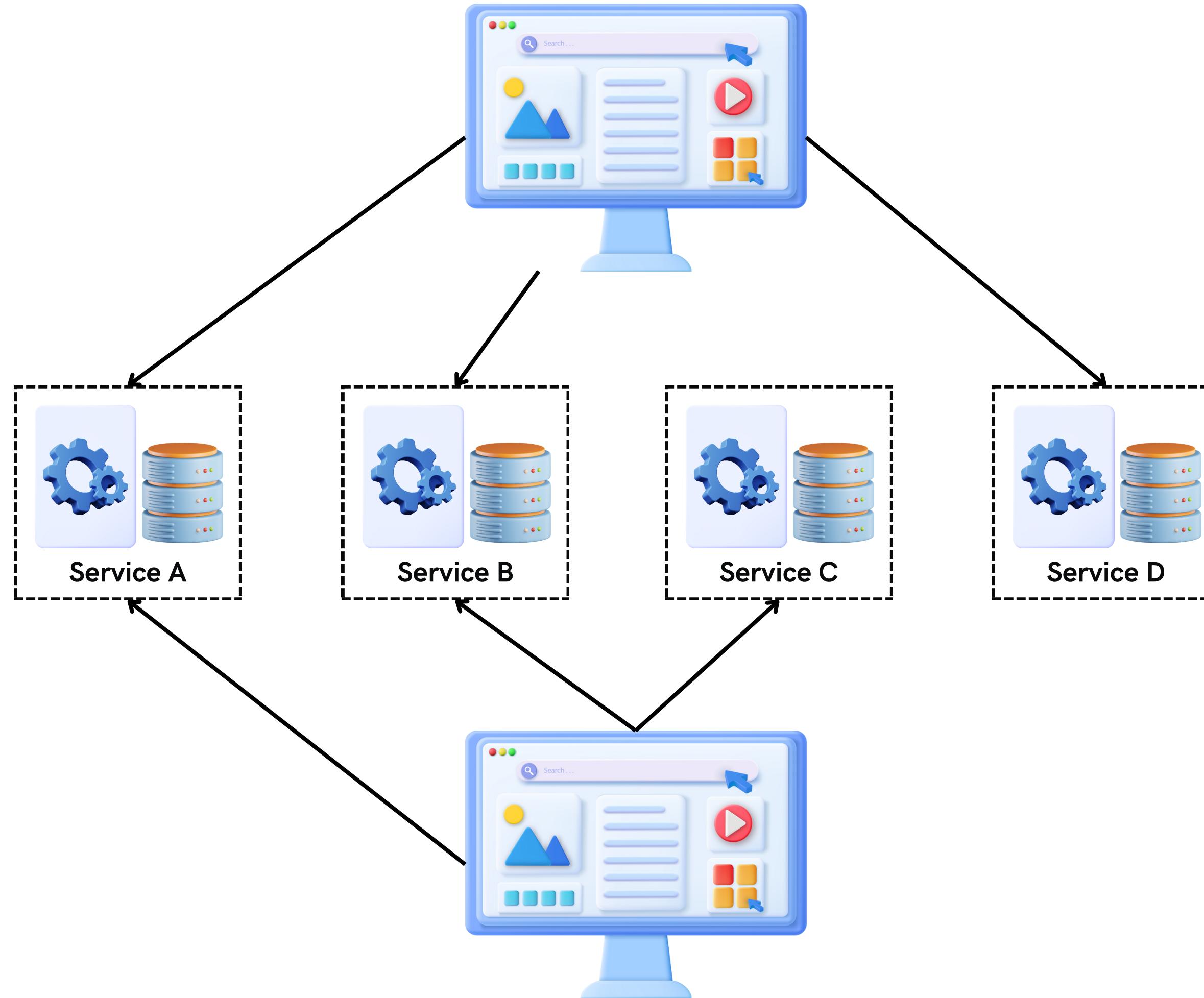
Background

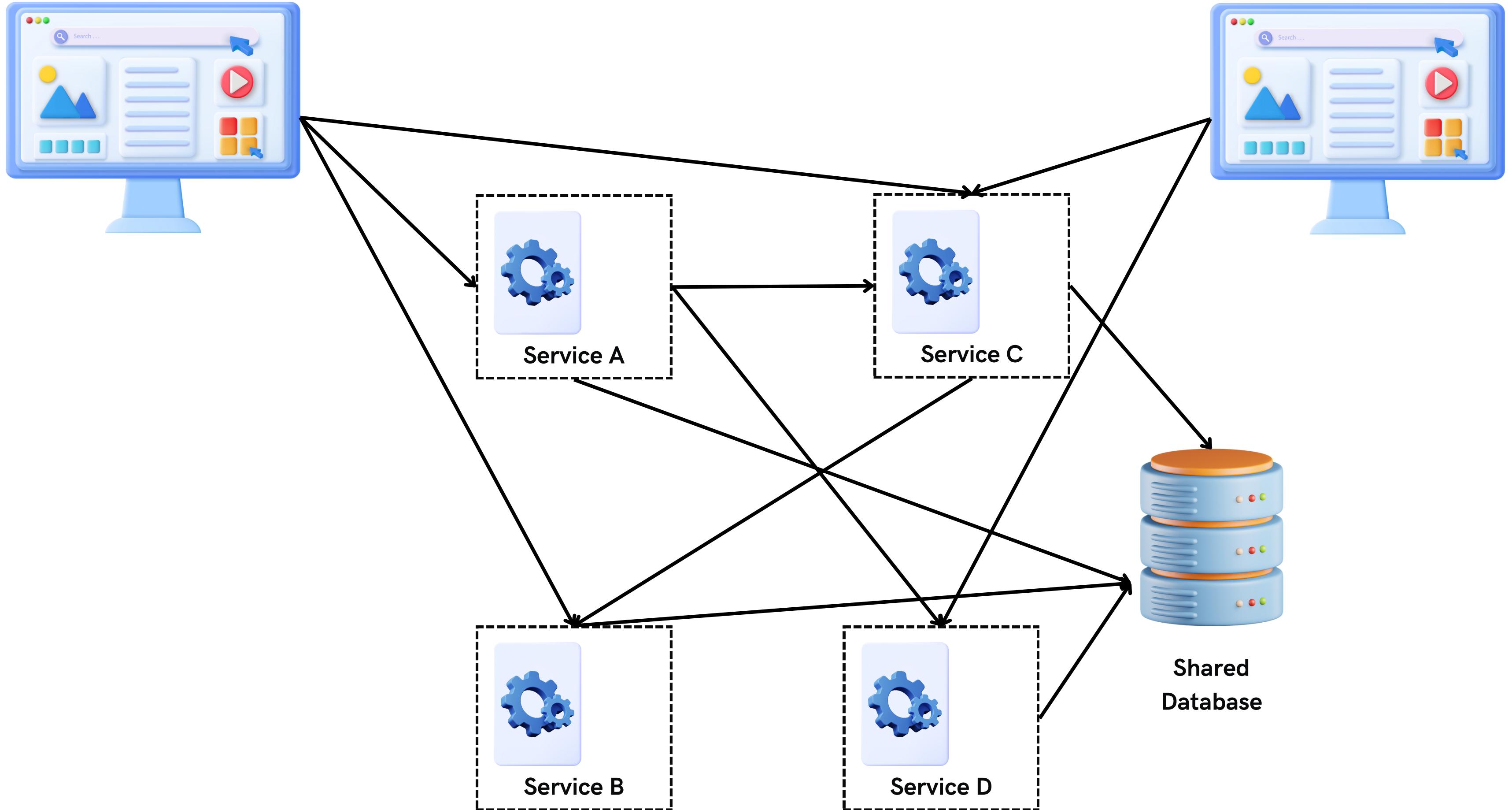


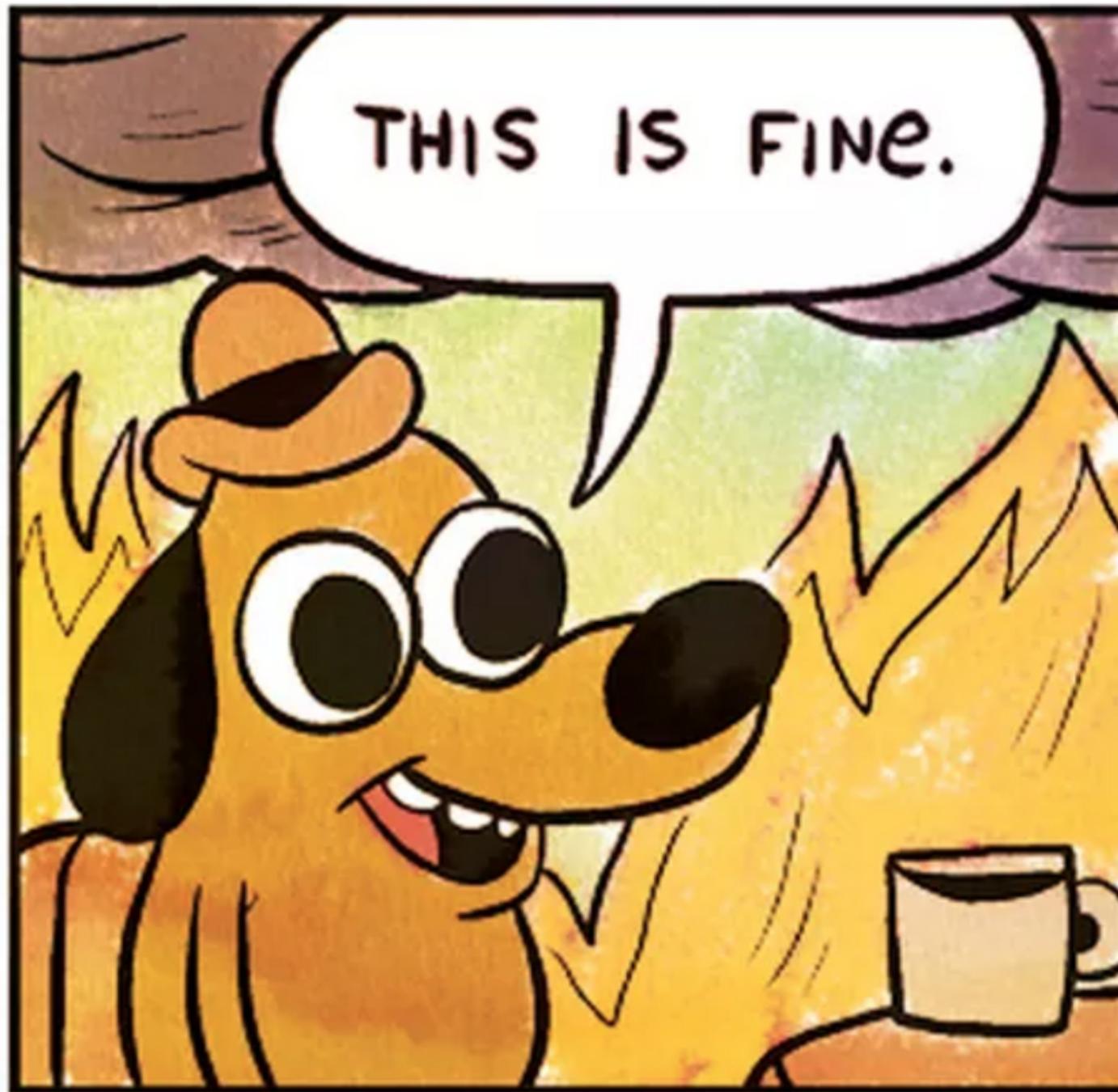
A Brief History of Microservices

SOA - Service Oriented Architecture



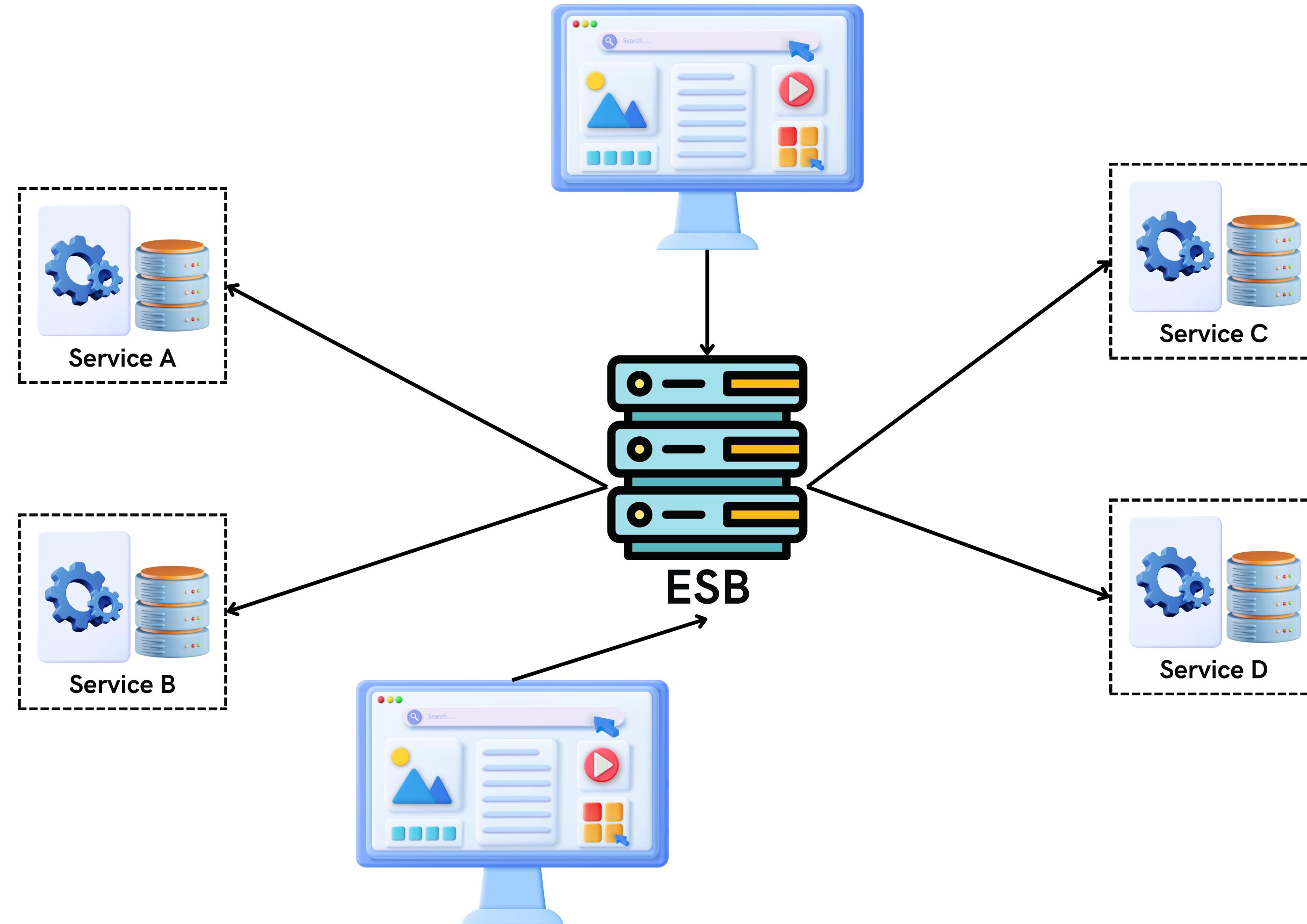


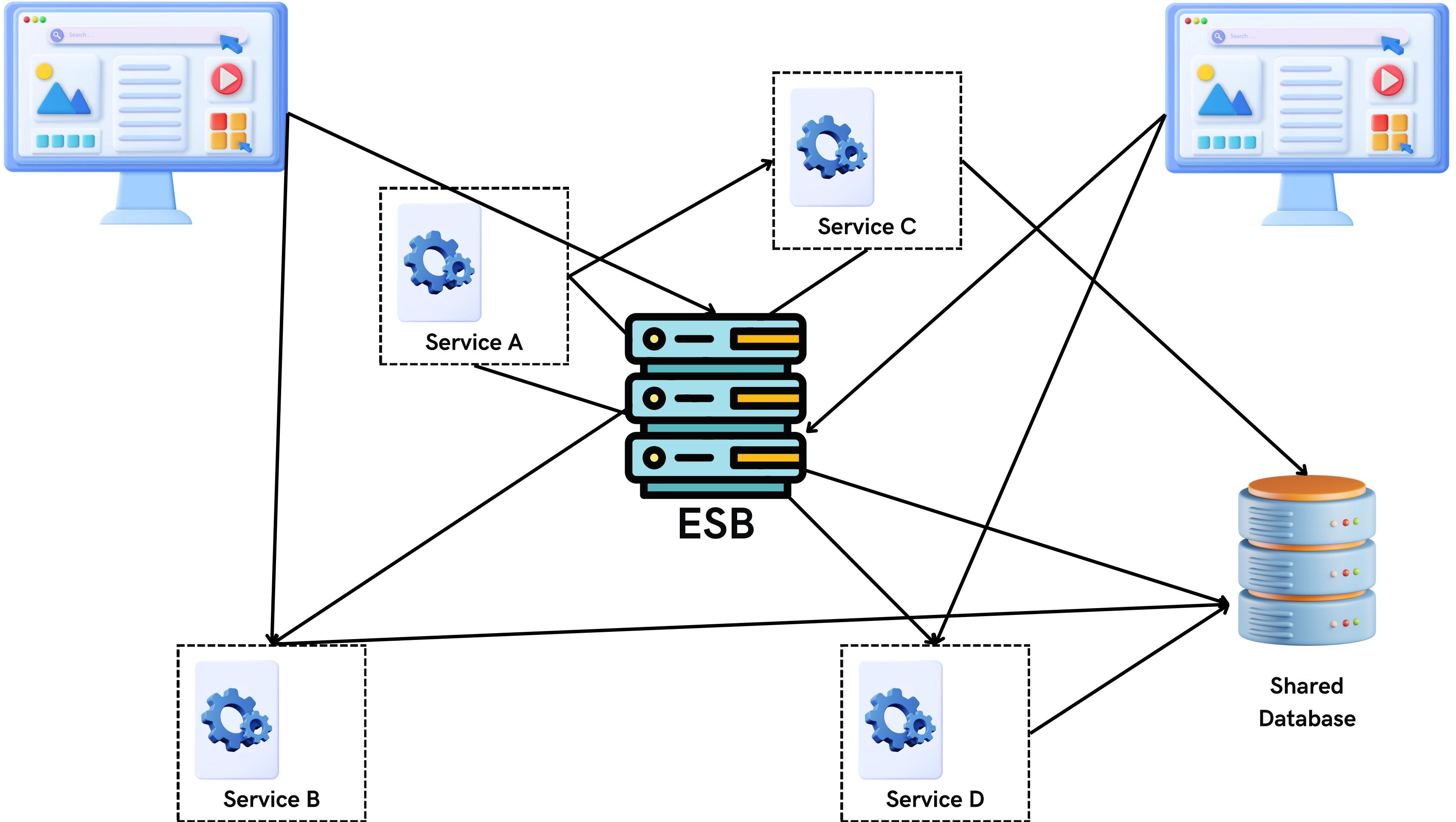


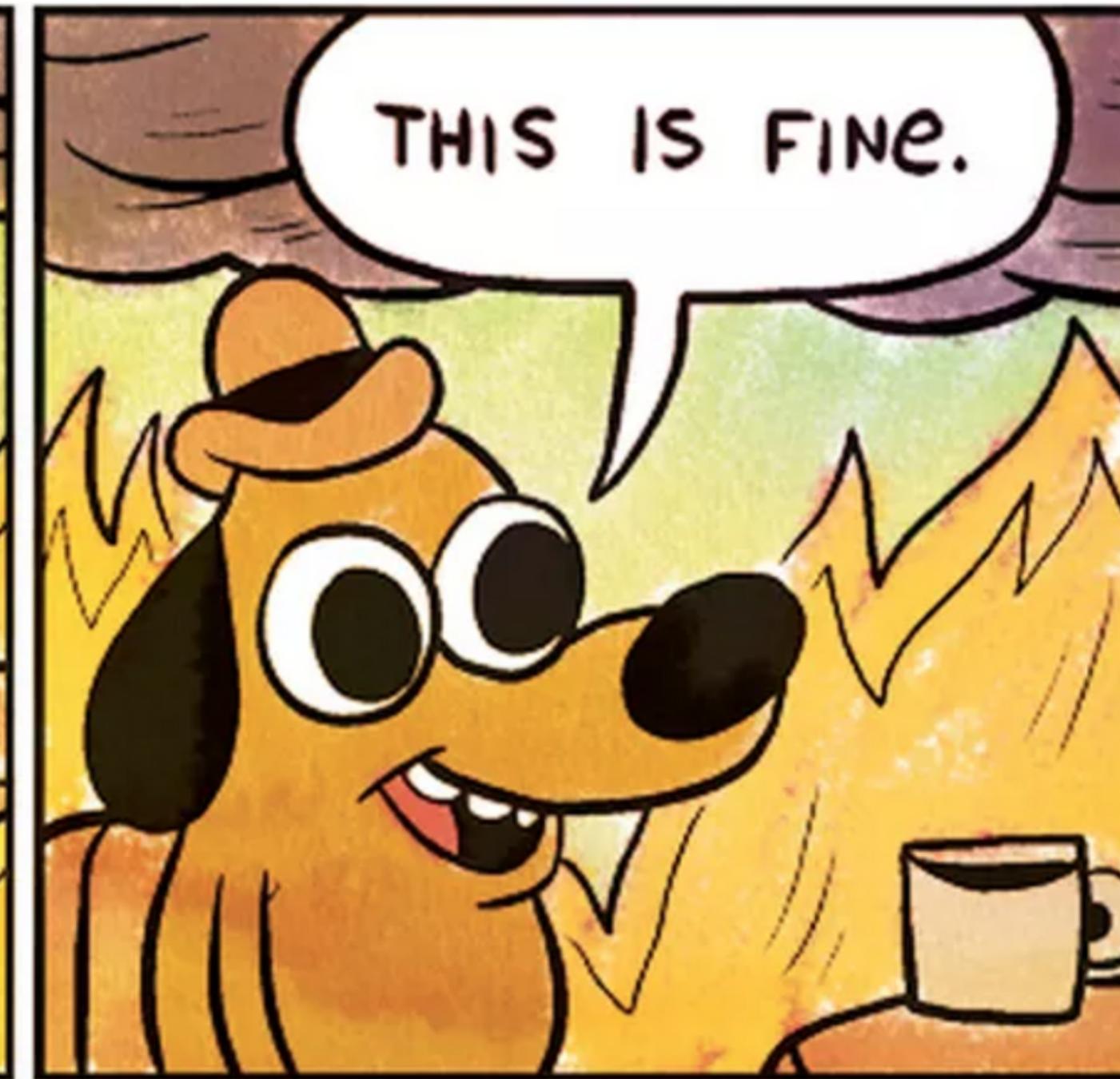


ESB - Enterprise Service Bus



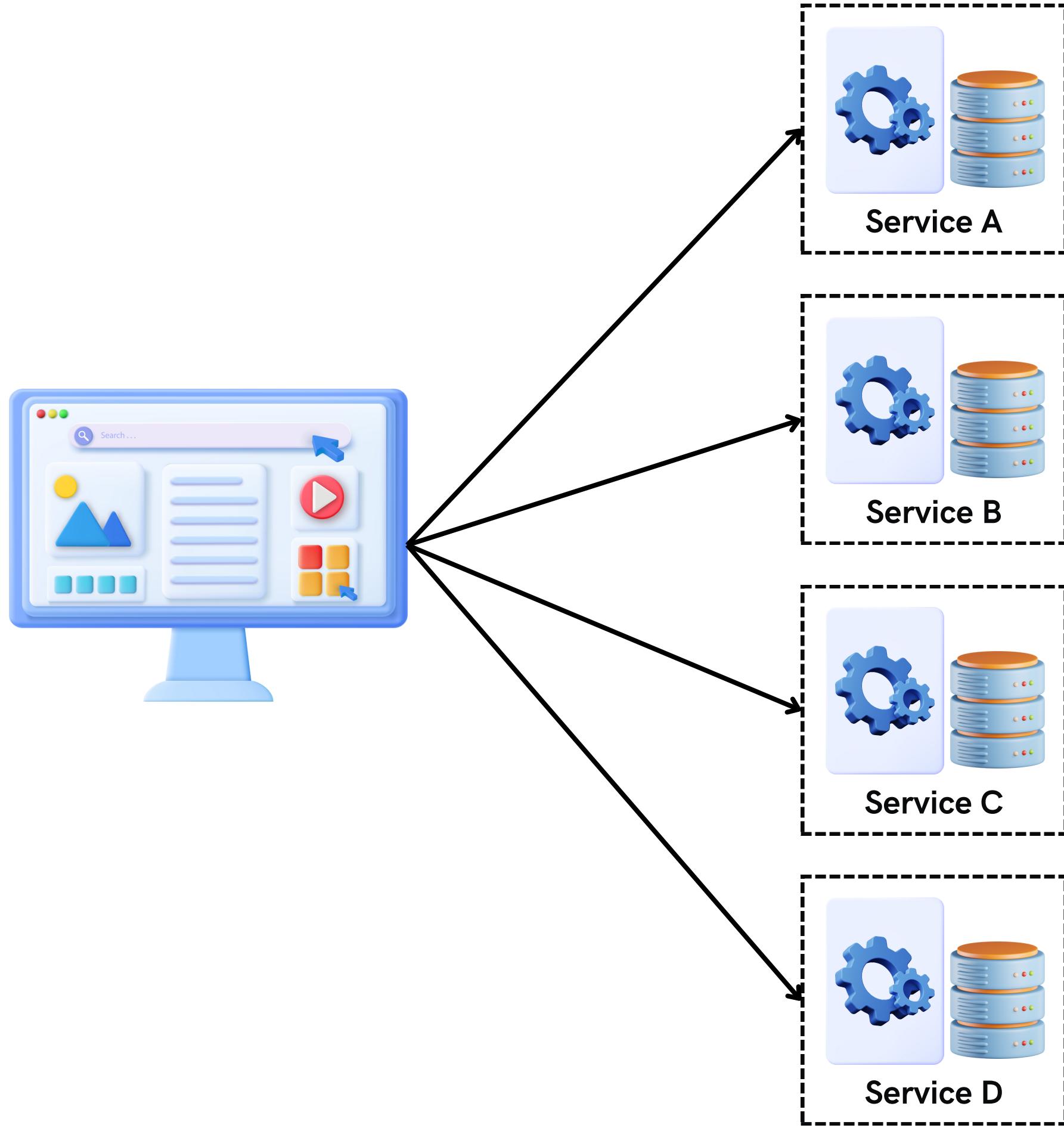


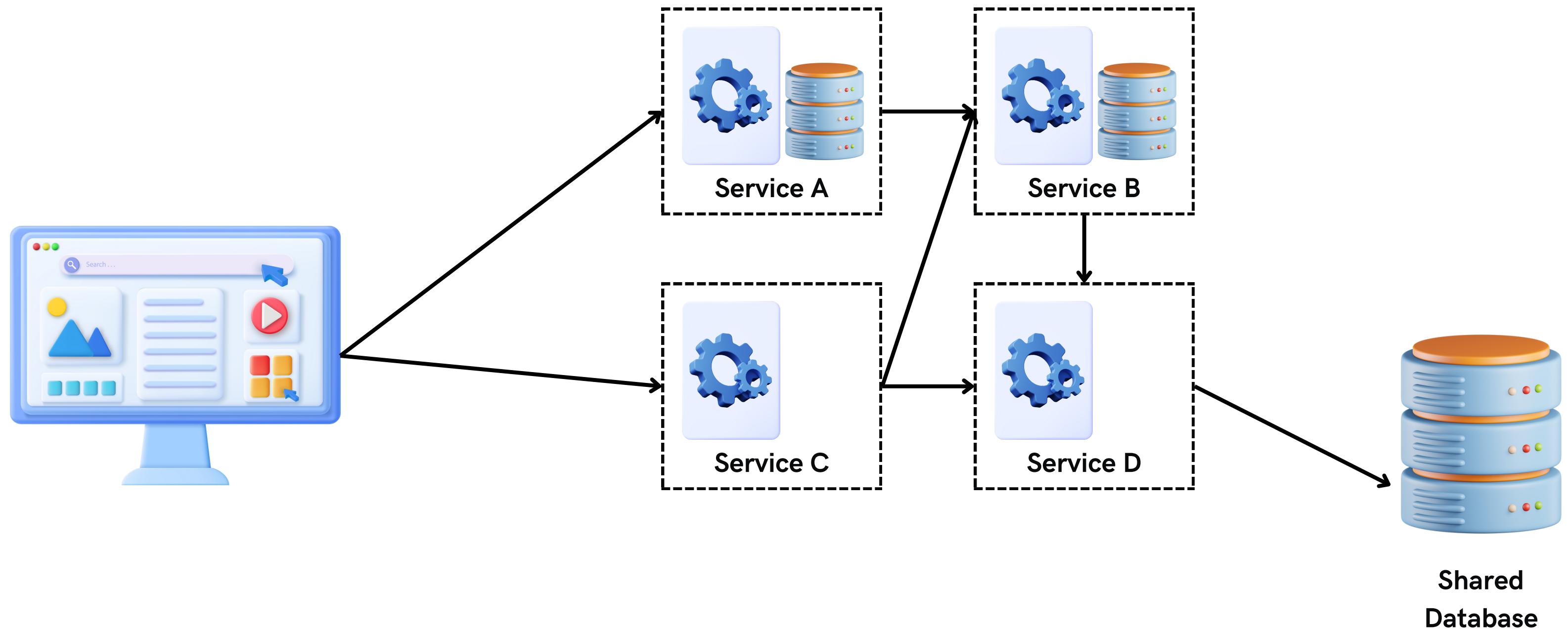


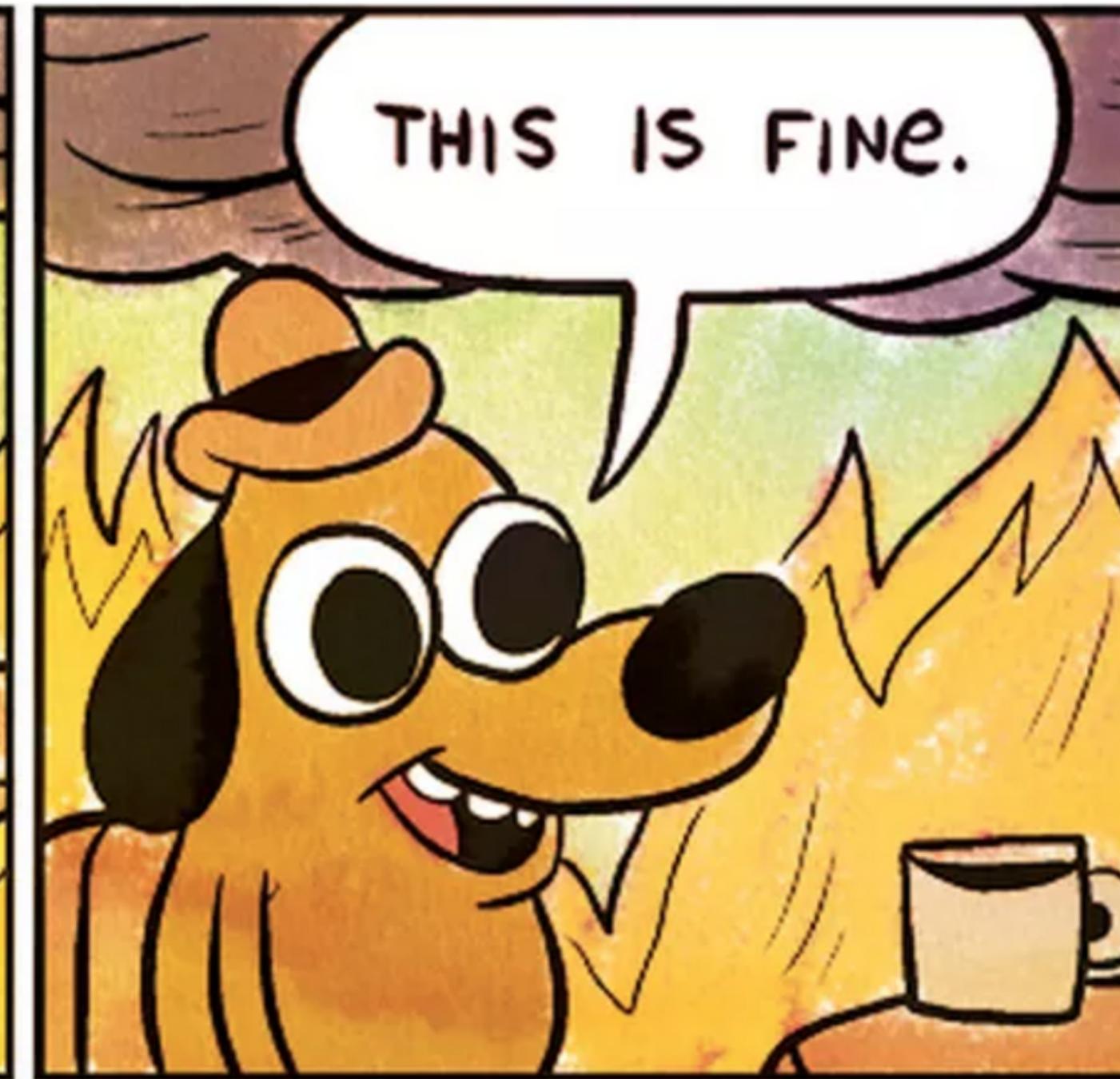


REST based Microservices



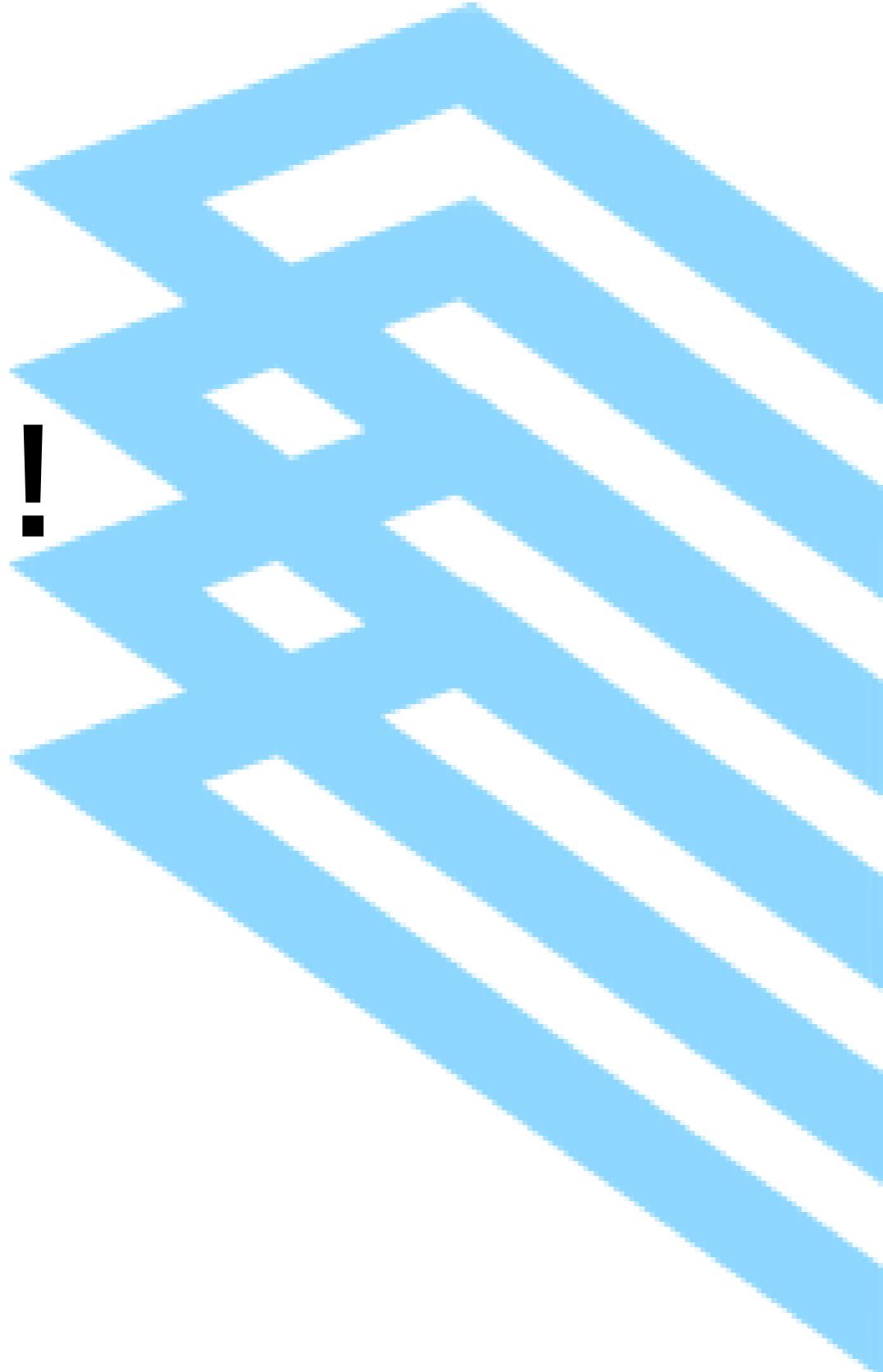






Where does it all go wrong?

Getting caught up in the tech!



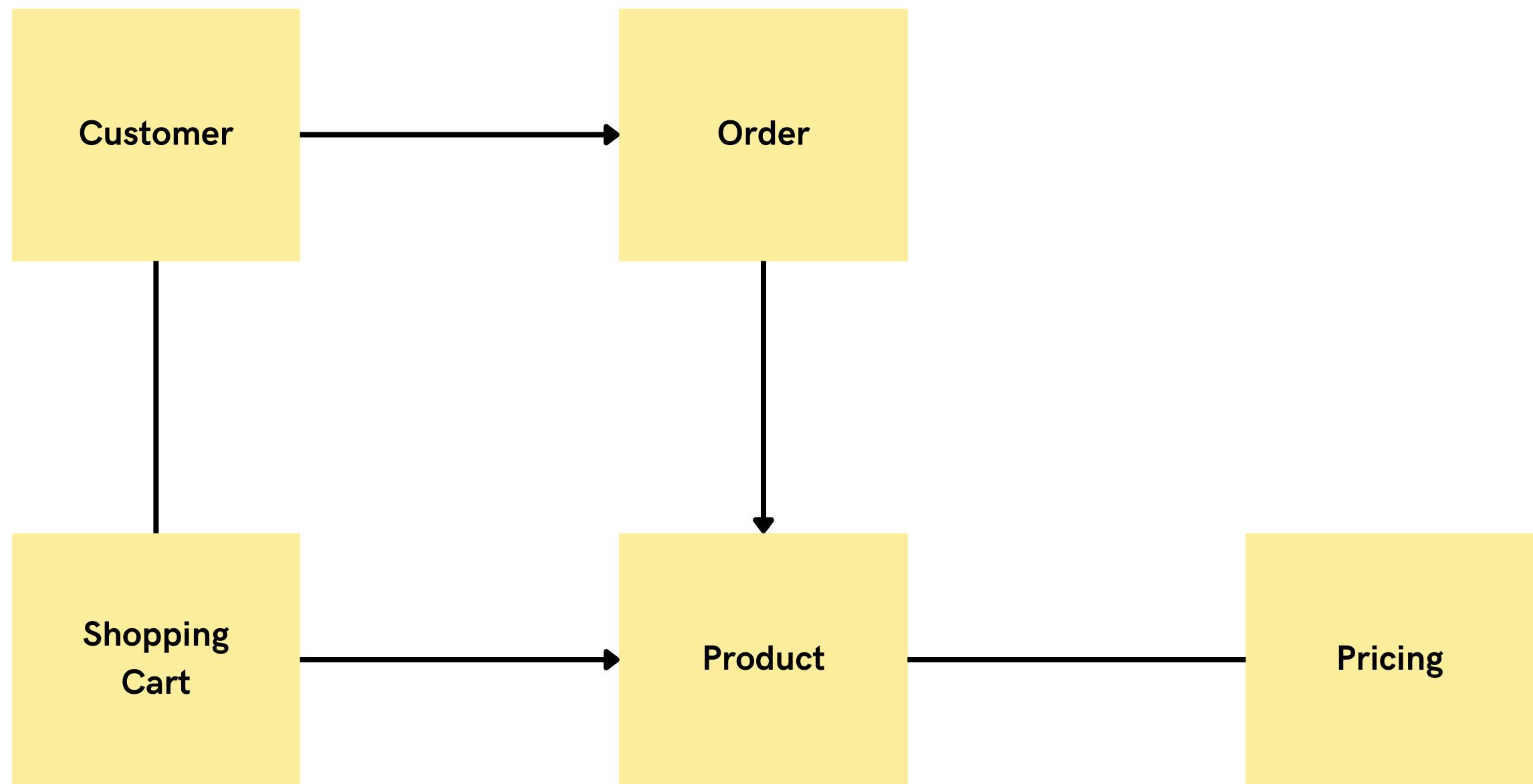


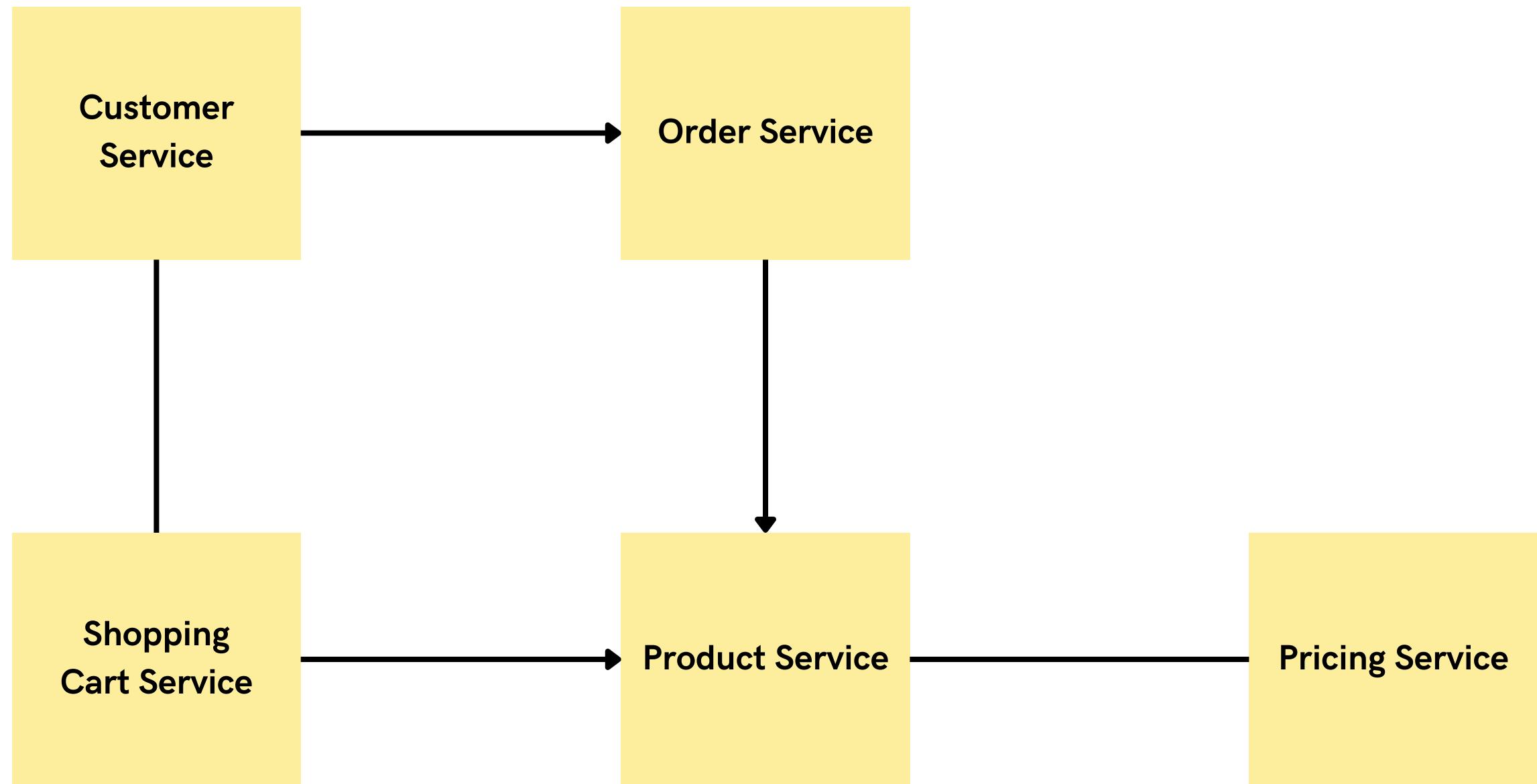
**If you haven't shipped one service. How
will you ship 20?**

Modelling

Entity Service Anti-Pattern





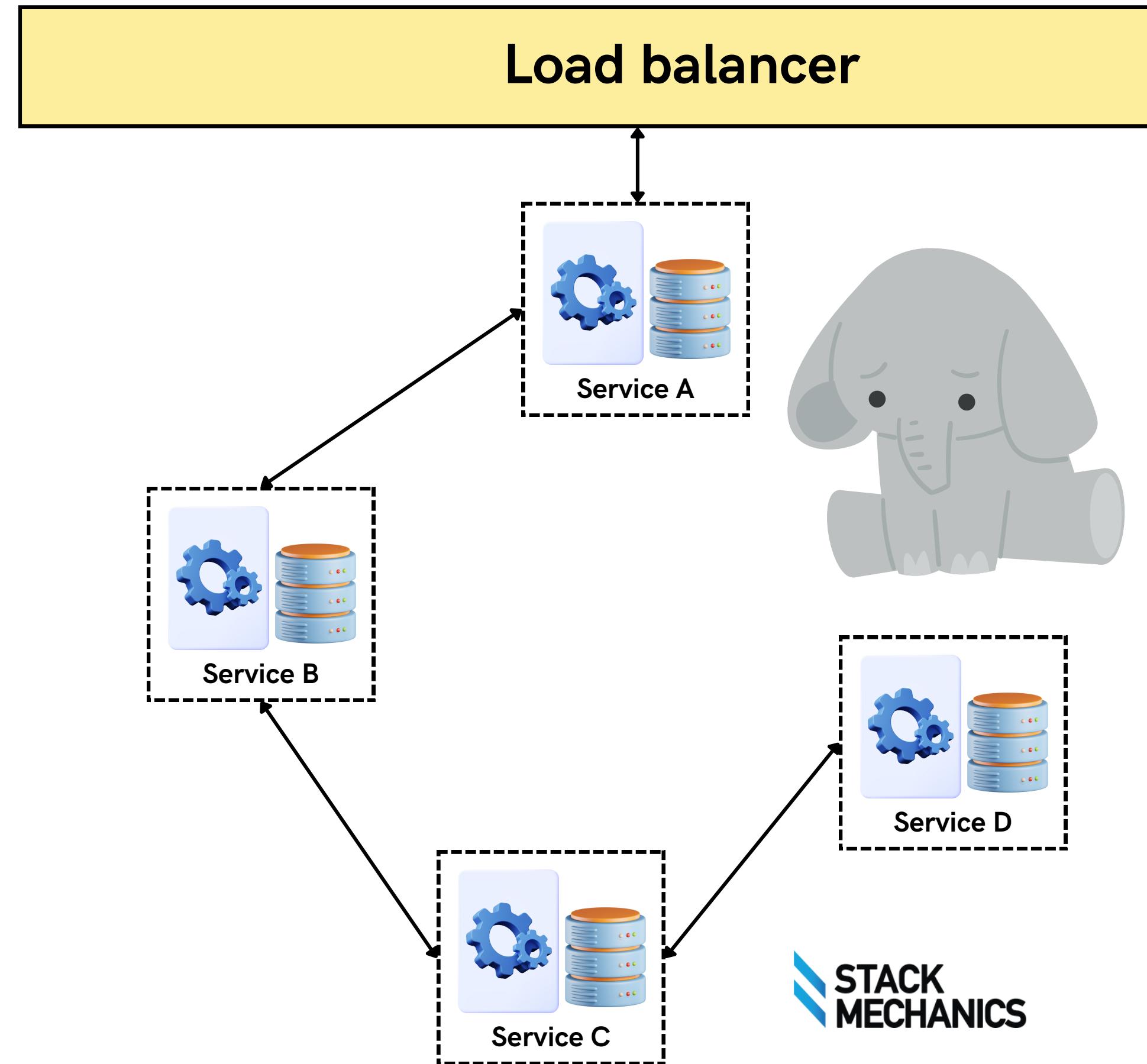


Symptoms



The problem with synchronous calls

- Introduces potential chains of failure
- Deployments are hard
- Failures become multipliers
- Latency is a multiplier
- Scaling is really hard
- Having these patterns makes you lazy



Domain Driven Design

Bounded Contexts

It's about people and behaviour

A Bad Restaurant

A Good Restaurant

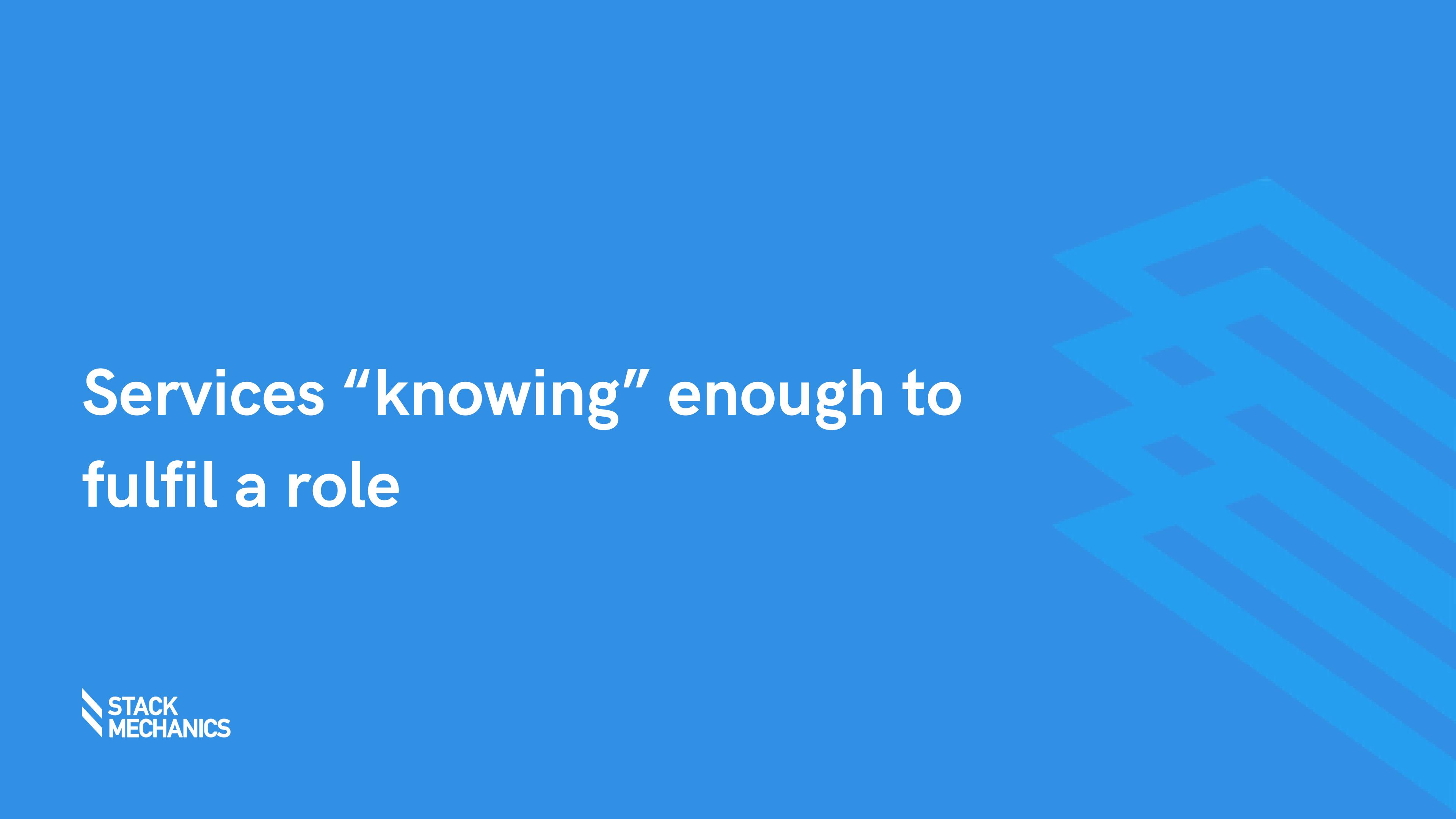
What can we learn from this?

Chains of requests don't scale

People that have the information to do
their job are more effective

**Software boundaries that follow these
principles will work better**

Synchronous vs Asynchronous



Services “knowing” enough to
fulfil a role

Human Shaped Microservices

**Look at the “Who” and “How”
rather than the “Thing”.**

What would people do?

How would they communicate?



Basic Messaging Concepts

The building blocks

Commands

A one-way message that
says **do this thing**

Send this
email

Generate a
PDF

Call the
payment
gateway

Events - publish and subscribe

A message that announces
this just happened.

A payment succeeded

We have a
new order

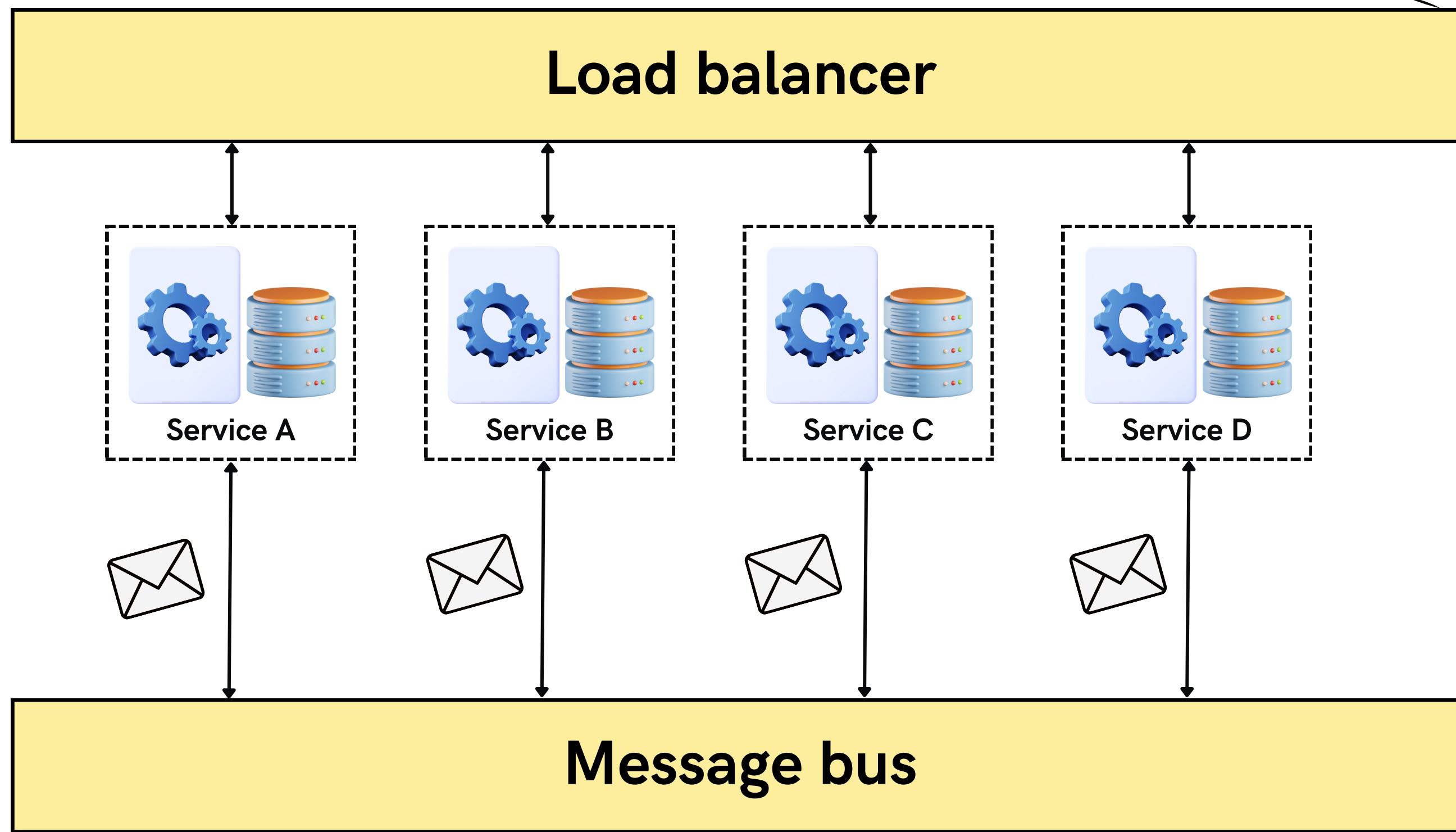
A new customer
signed up

Publish and subscribe

Any service that **cares** about this information, can listen for these messages and get a copy



Mullet Microservices



Loose coupling and domain design

- Hammers and design constraints
- Events and commands are about behaviour, and behaviour is how we model rich domains
- Thinking in terms of events (and commands) will help you model your domain better



Some conclusions

- Learn from history
- Microservices and SOA have the same roots (and the same pitfalls)
- Modelling is critical
- Look at people, behaviour, and processes in your modelling



**Never forget, delivering
value is the goal.**





Questions?

