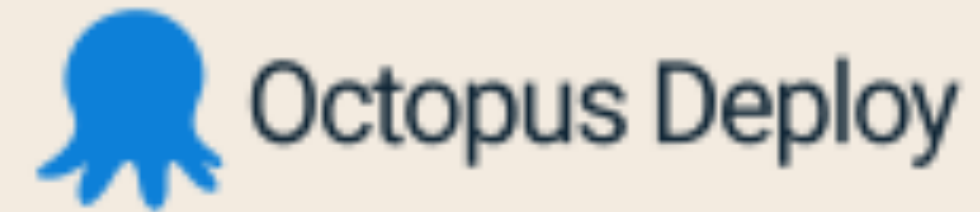


Human Shaped Microservices

A people centric approach to modelling systems

GOLD SPONSORS



SA SPONSORS



SILVER SPONSORS



WITH SUPPORT FROM



Hi! I'm Damian

- “Consultant CTO”
 - Long time consultant
 - CTO for a few software and hardware startups
 - Now Freelance
- I teach software architecture at Stack Mechanics
- I live in Brisbane
- I have a lot of hobbies
- Can find me as @damianm in most places.



Agenda

- Some background
- A brief history of microservices
- Where people go wrong
- Modelling concepts
- Domain Driven Design - The other DDD
- Human Shaped Microservices
- A side rant into cargo cult agile and why words have meaning
- Wrap it up with a couple of things you can take away

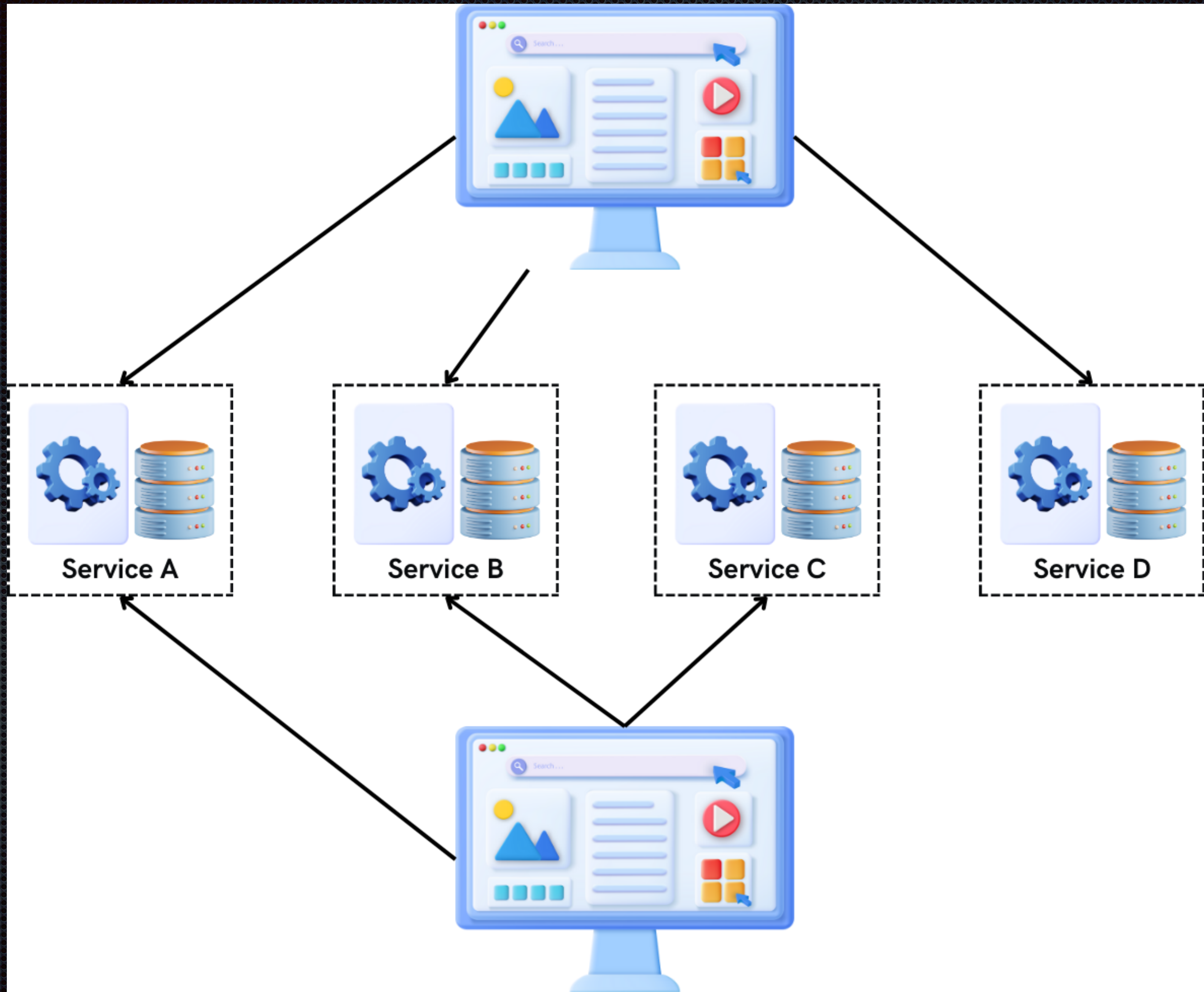
Background



Photo by [Shannon Kunkle](#) on [Unsplash](#)

A brief history of Microservices

Service Oriented Architecture



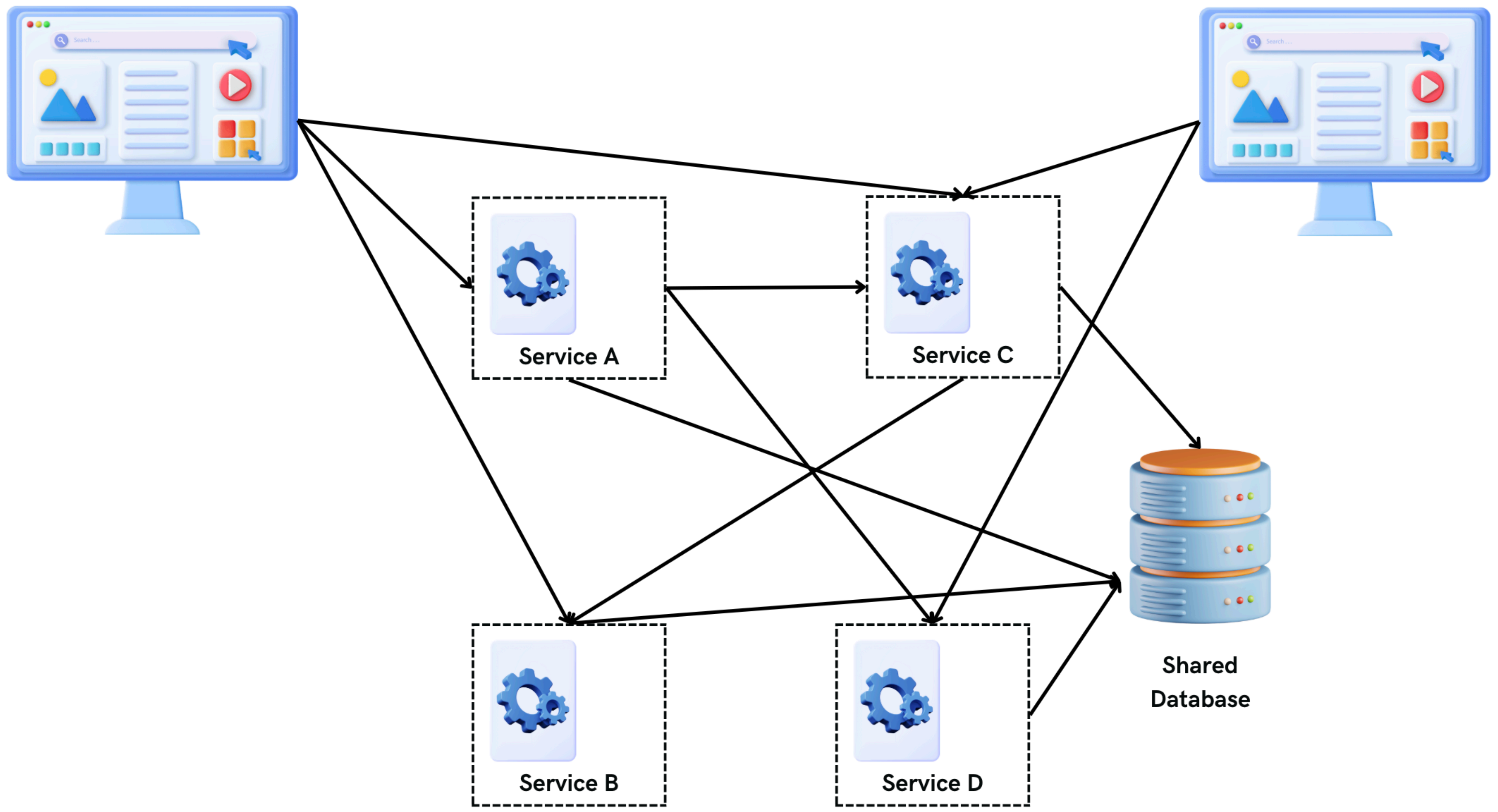
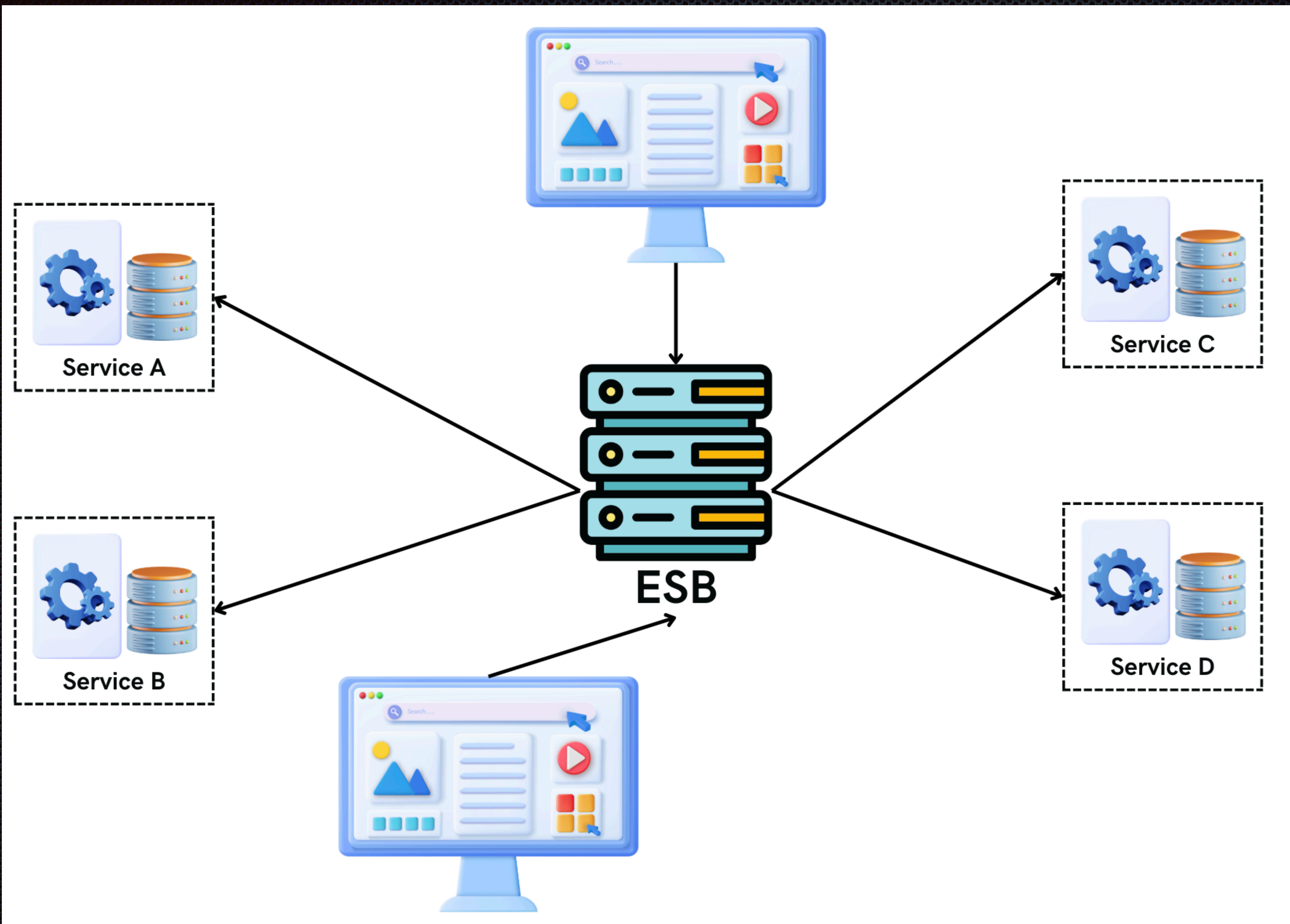




Photo by [Shannon Kunkle](#) on [Unsplash](#)

Enterprise Service Bus - ESB



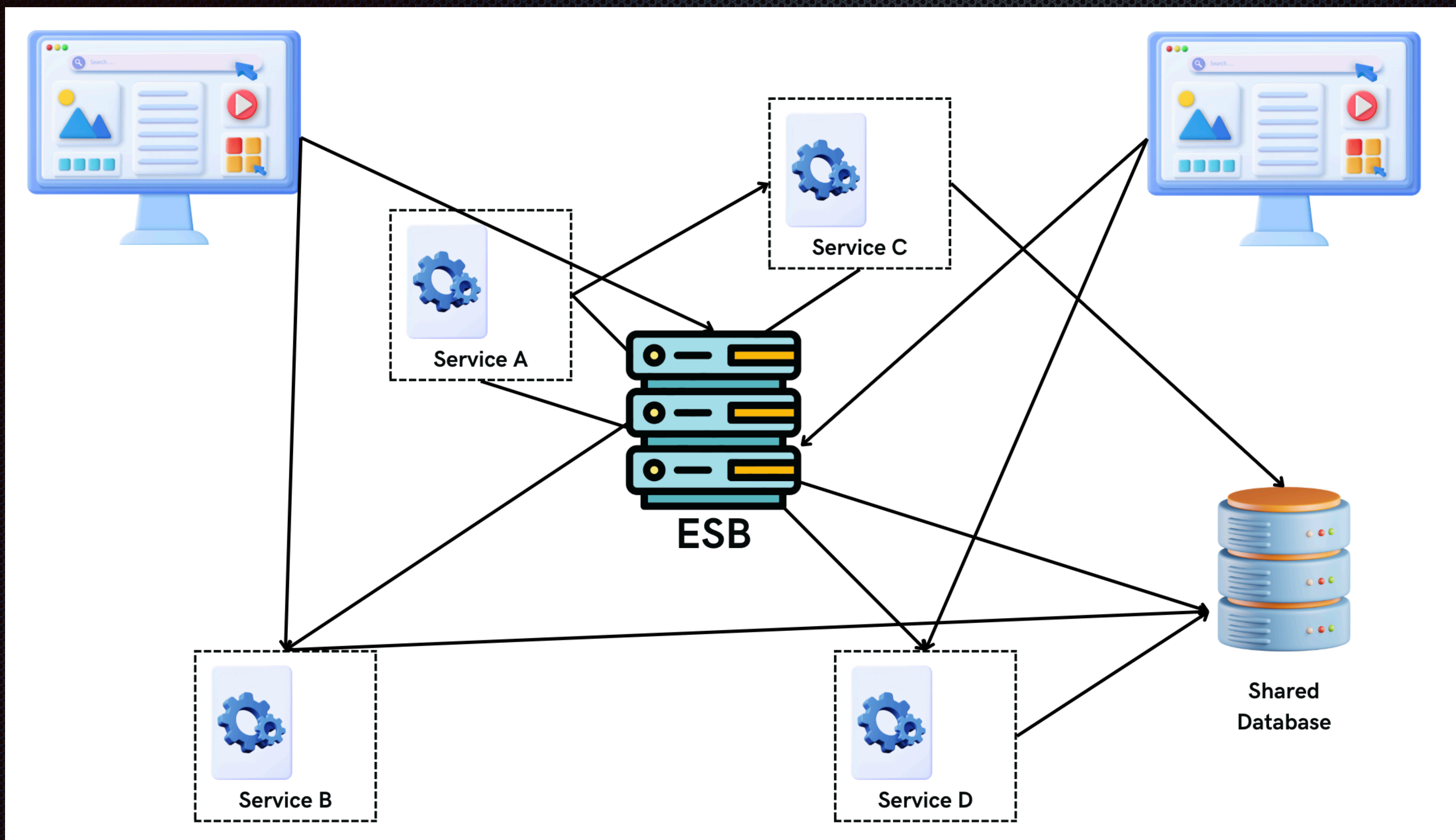
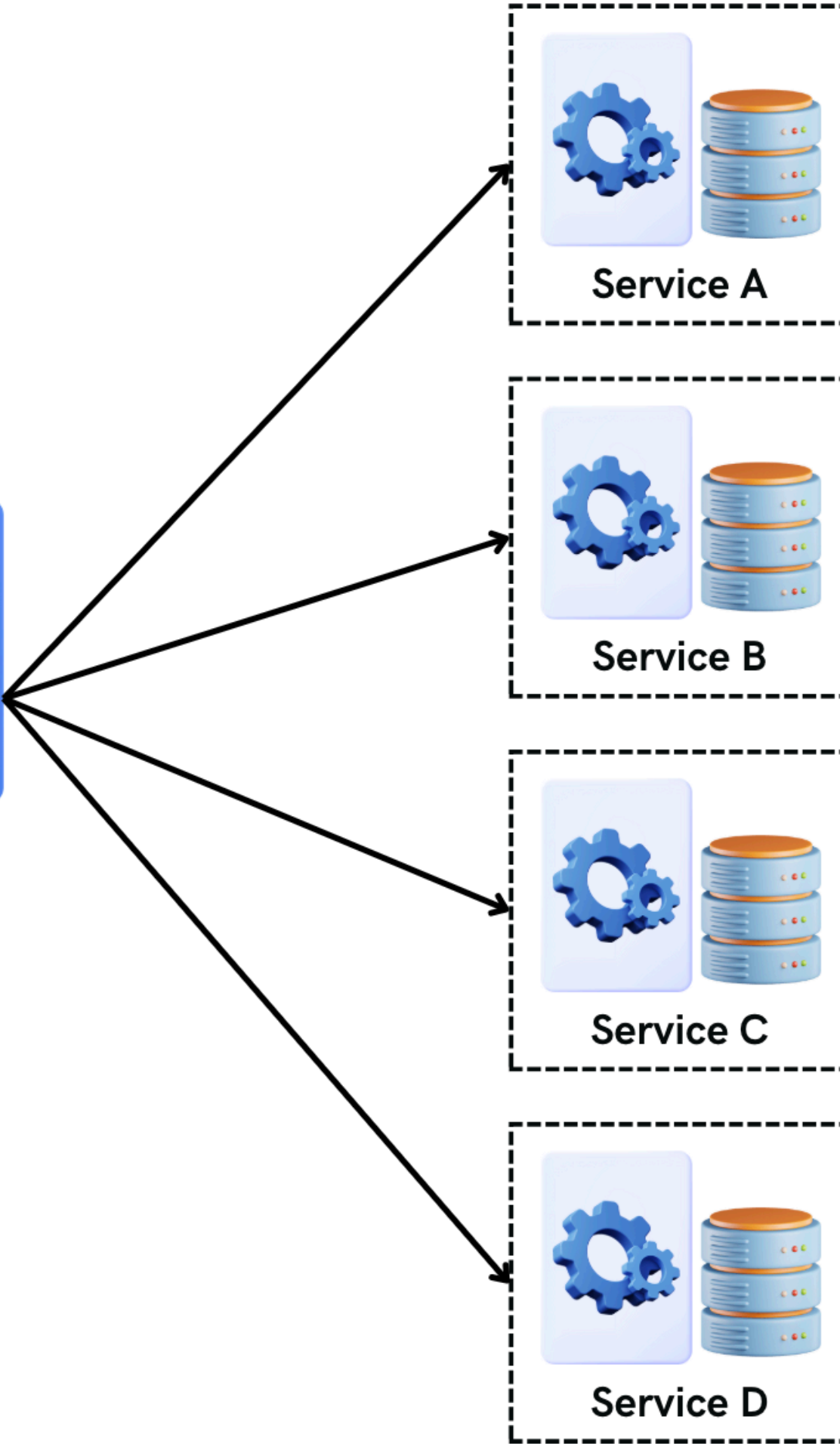
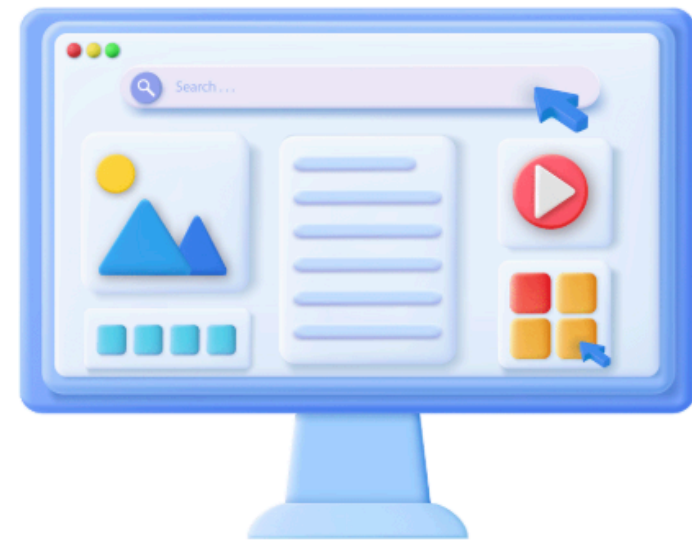




Photo by [Shannon Kunkle](#) on [Unsplash](#)

REST Based Microservices



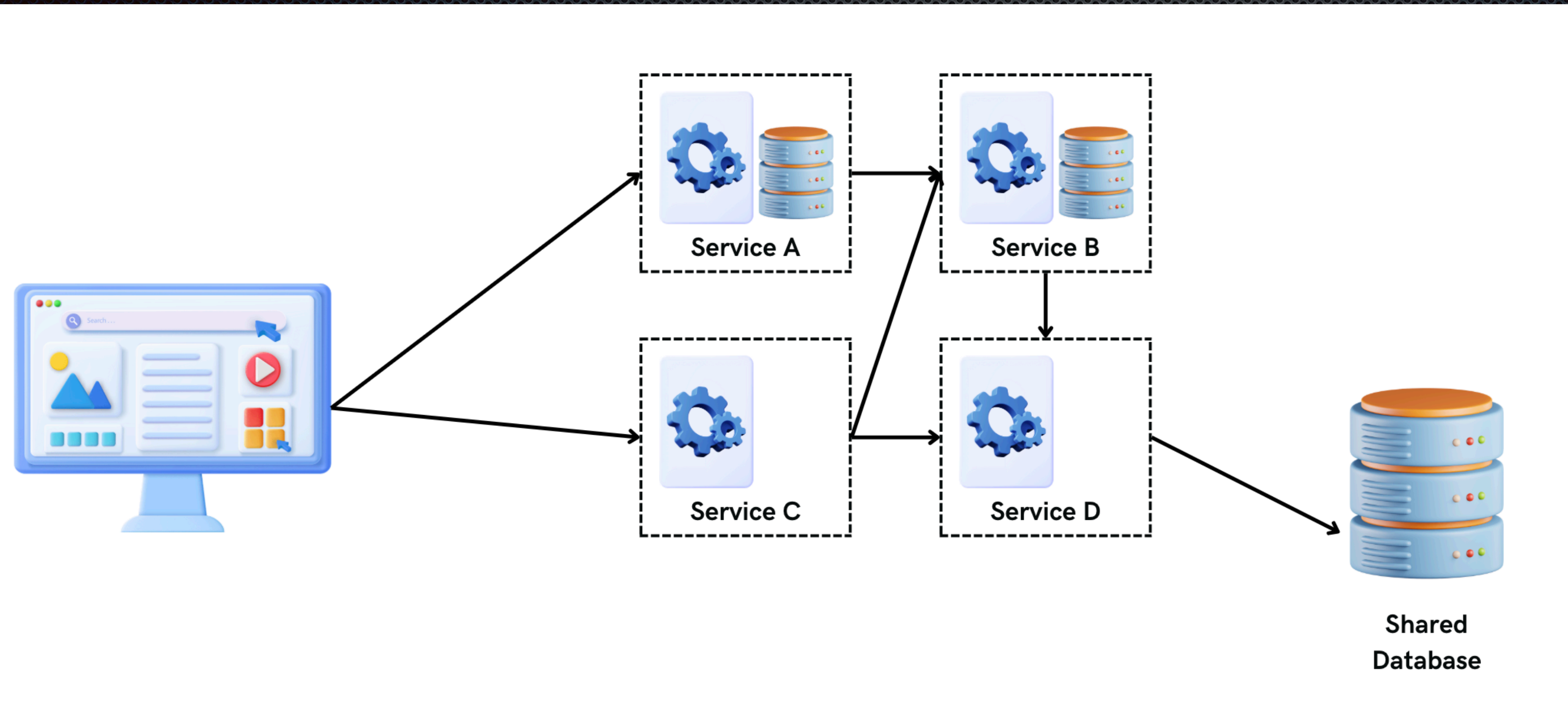




Photo by [Shannon Kunkle](#) on [Unsplash](#)

“Where do we keep going wrong?”

“Everything old is new again”

–Jonathan Swift

“Those who do not remember the past are condemned to repeat it”

–George Santayana

Modelling

The Entity Service Anti Pattern

AKA - We are really good at modelling data, but bad at modelling behaviour.

Failure Mode #1

A calling service has to make **multiple calls to multiple services** to collect all the data it needs.

Then decide how to join it.

Your responsibilities have just leaked all over the place!



Wim Deblauwe
@wimdeblauwe

Follow

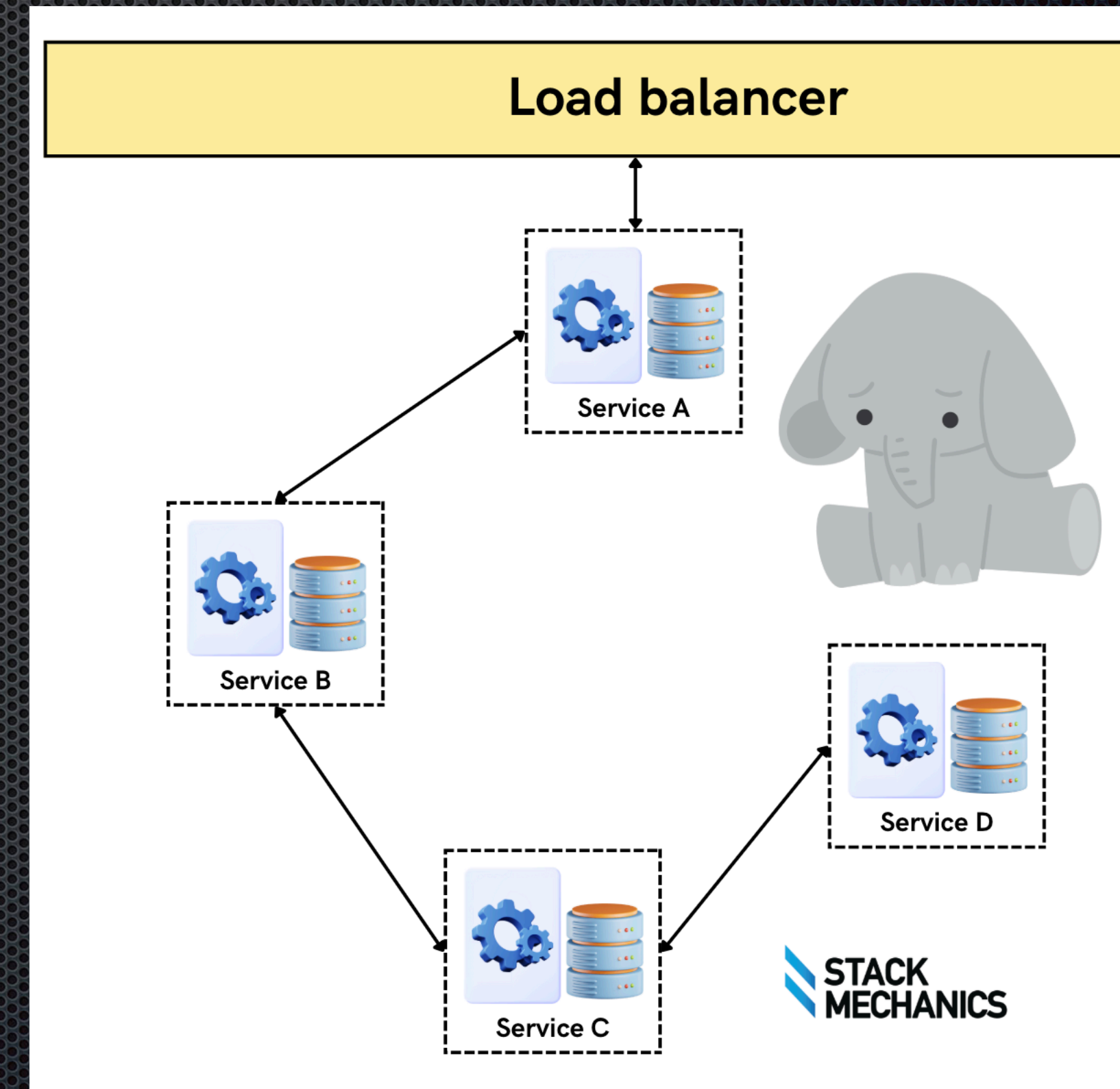


Microservices: what could have been a JOIN, is now a 1000 network calls

Failure Mode #2

The service has to orchestrate calls to other services.

This causes a synchronous chain of locks and all the latency and fault tolerance issues that we see.



Other Smells

- Overuse of Polly or other retry libraries
- Services trying to manage referential integrity or Entity Relationships over HTTP
- Overuse of service meshes
- Service locators in code

```
public interface IHttpServiceLocator
{
    string GetUrl(string environment, string entityType);
}
```

- ✦ This breaks so many of the rules we know about modelling.
- ✦ We're tightly coupled, but in a way that introduces latency and failure.
- ✦ None of the services have any autonomy or own any business process.

Domain Driven Design

Bounded Contexts

Story Time - A bad restaurant

Story Time - A good restaurant

Some Lessons

- Chains of requests where you have to wait for an answer doesn't work so well
- Letting people know what you want, and having them let you know when something is done, or has changed works a whole lot better
- People with the information to do their job are more effective
- Different teams need different information in different formats

“Software which follows these principles will work better.”

-Me

Human Shaped Microservices

Human Shaped Microservices

- ✦ Async over synchronous where possible
 - ✦ Sending commands
 - ✦ Publishing or receiving events
- ✦ Services “knowing” the right amount to fulfill a role or process

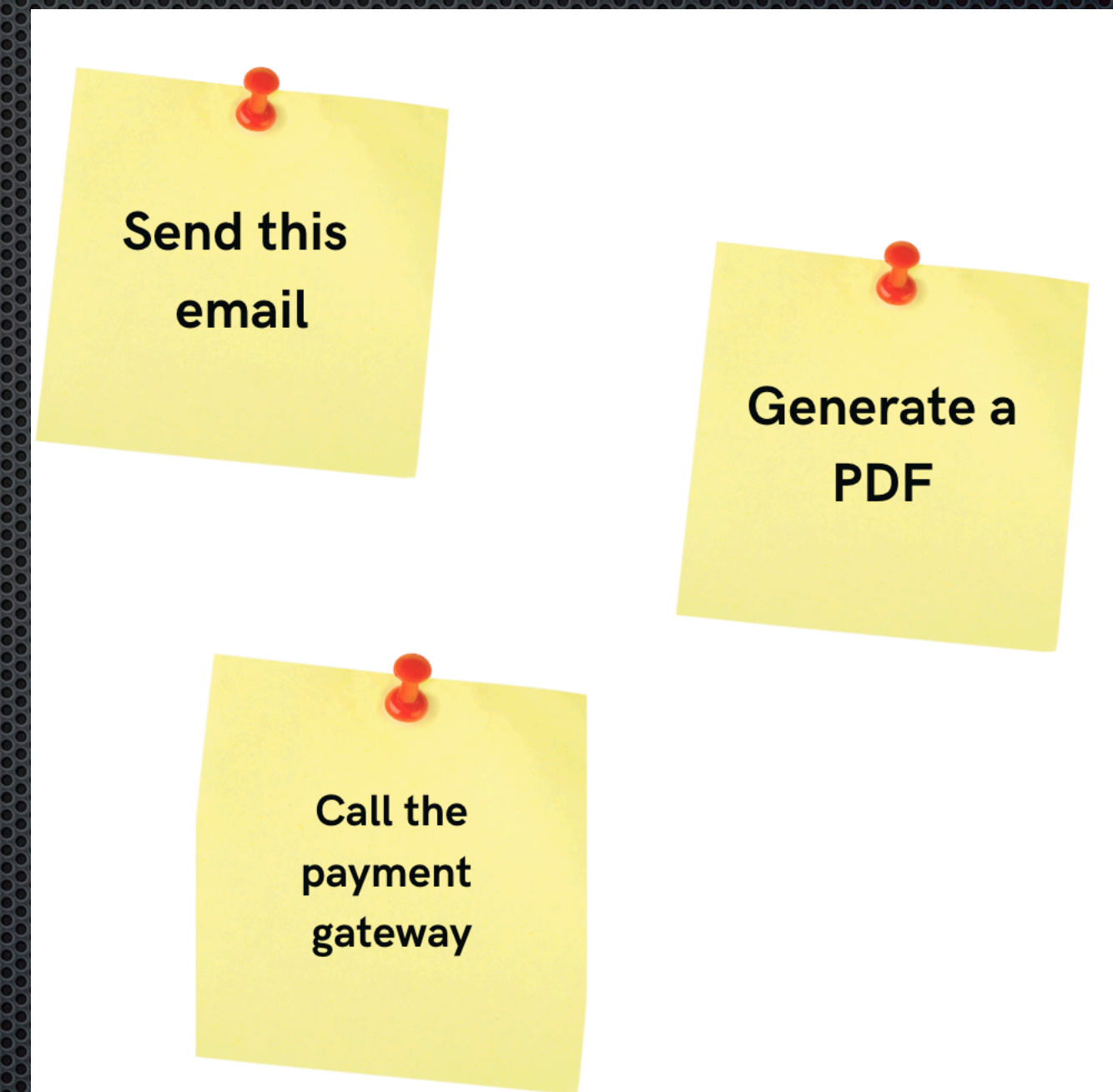
Modelling Principles

- If you were building this as a company, with people, what are the teams?
- What are the different roles?
- How do they communicate with each other?
- What roles can you put multiple people in to scale their work? How does work get assigned?

Some building blocks

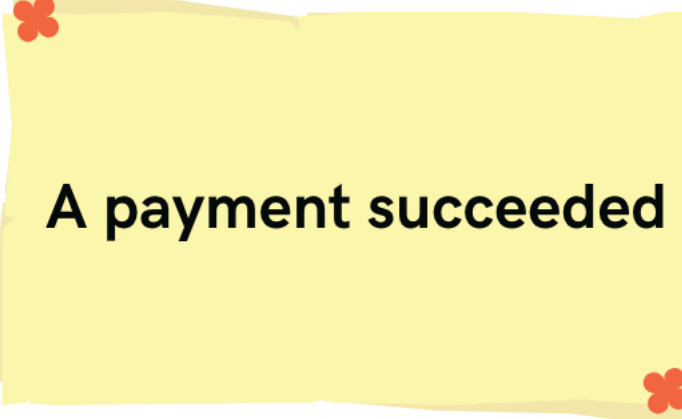
Commands

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




Events


- ✦ Events can be **published**, and they can be **subscribed to** (pubsub)
- ✦ A message that announces **this just happened.**
- ✦ Any service that cares about this information can listen for these messages and get a copy.



A payment succeeded



We have a
new order



A new customer
signed up

Bounded Contexts

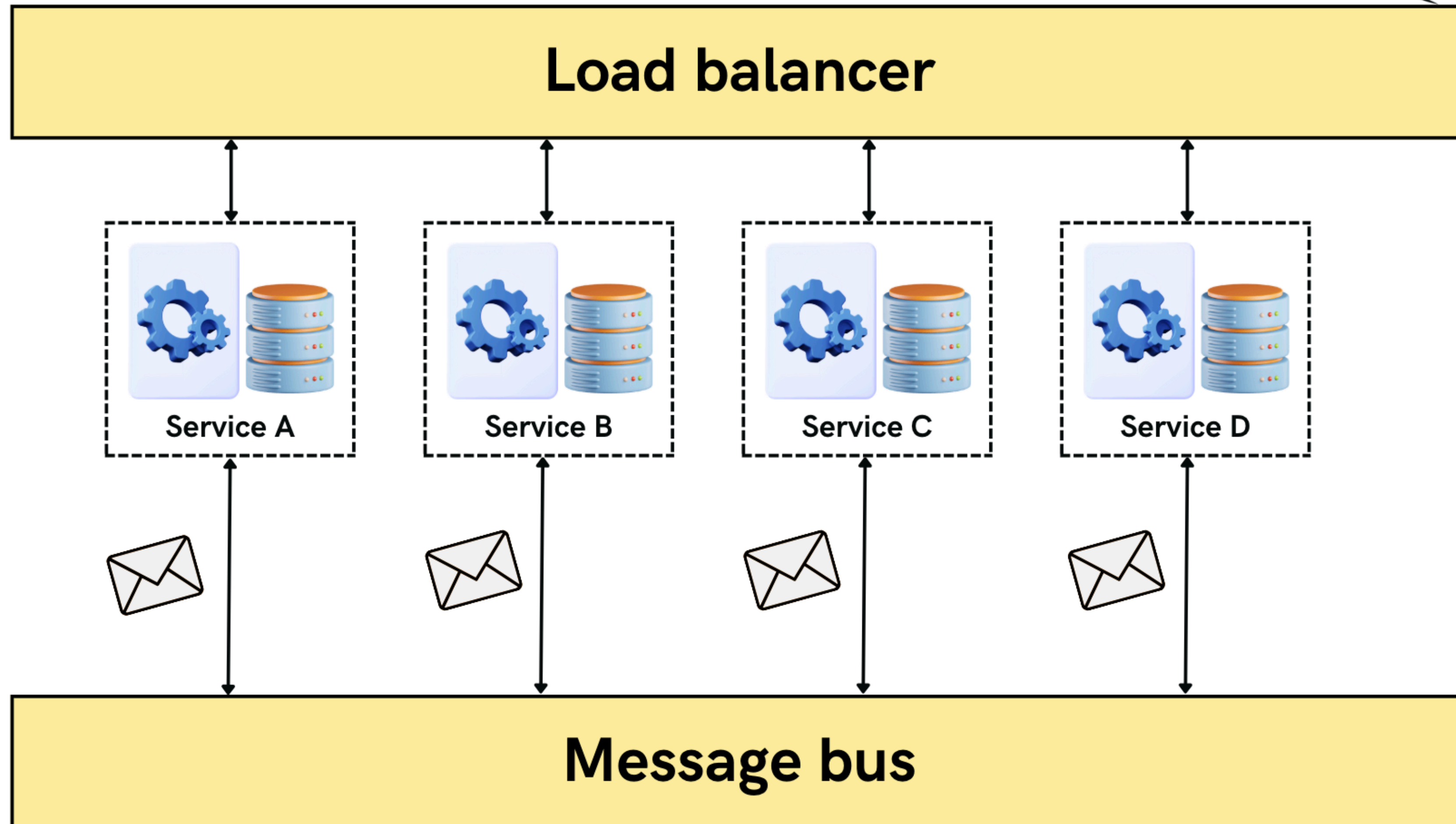
- ✦ This is the size and shape of your service.
- ✦ What does the “team” look like to own this business process or problem space. When would you split a team up into sub teams

Loose coupling and domain design

- ✦ Pick your hammer wisely
- ✦ Events and commands are about behaviour, and behaviour is how we model rich domains.
- ✦ Thinking in terms of events and commands will give you a better model, and a more fault tolerant architecture.



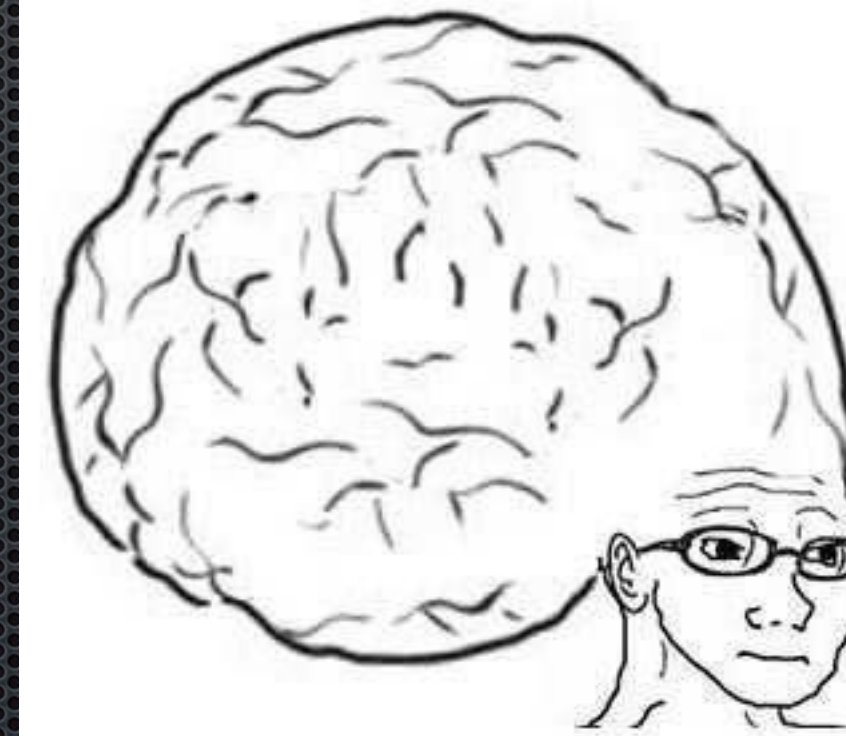
Mullet Microservices



Side Rant

- SOA was never just about SOAP
- Microservices were never about putting HTTP on every database table
- DDD isn't just the Repository Pattern
- Devops isn't just about hiring a team to be build engineers
- Agile isn't just about user stories and "Kanban" boards
- Words have meanings, and when we lose sight of this we all suffer

Agile then



"Let's get a customer to sit with the team while we iterate on a solution"

Agile now



"Your card is in the wrong Jira template"

Some Conclusions

- ✦ Learn from history
- ✦ Understand what problem you're really solving
- ✦ Modelling is critical
- ✦ Look at the people, the behaviours, and the processes in your modelling
- ✦ Talk to people!

Thank you!

