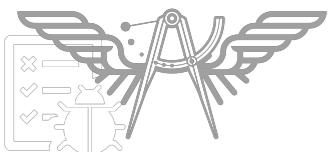


Design Services/Microservices





Somkiat Puisungnoen

Somkiat Puisungnoen

Update Info 1

View Activity Log 10+

...
Timeline About Friends 3,138 Photos More

When did you work at Opendream?
... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro
Software Craftsmanship

Software Practitioner at สยามชัมนาภิกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ⚙️

Java and Bigdata





Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

@somkiat.cc

Home

Posts

Videos

Photos



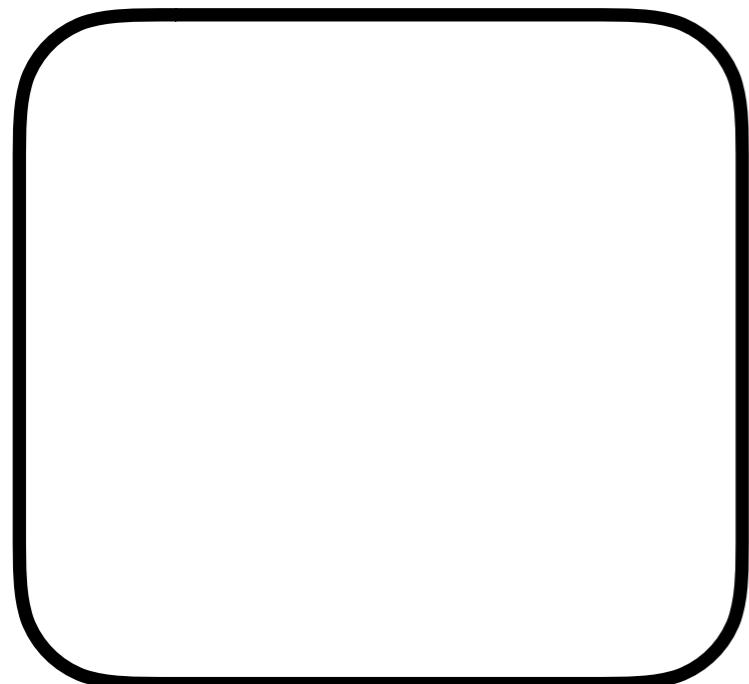
**[https://github.com/up1/
course_microservice_kmitl_2019](https://github.com/up1/course_microservice_kmitl_2019)**



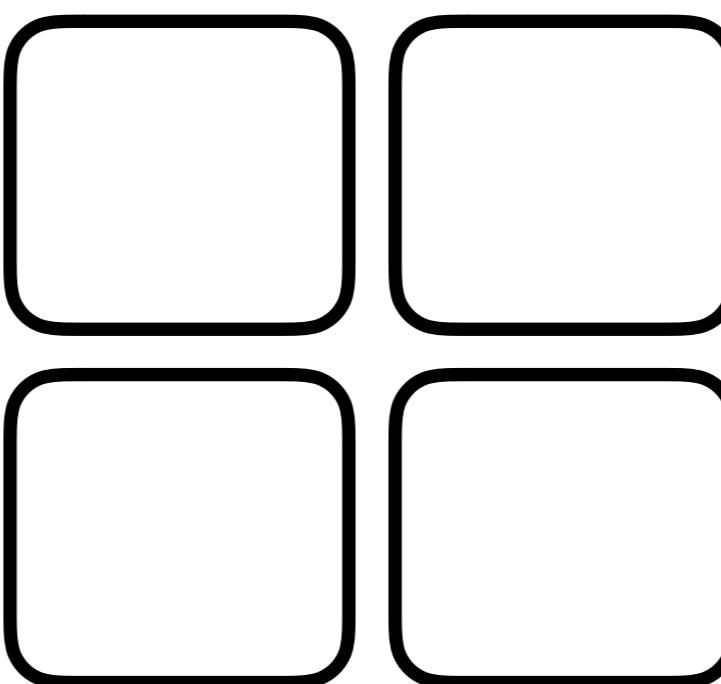
Let's start



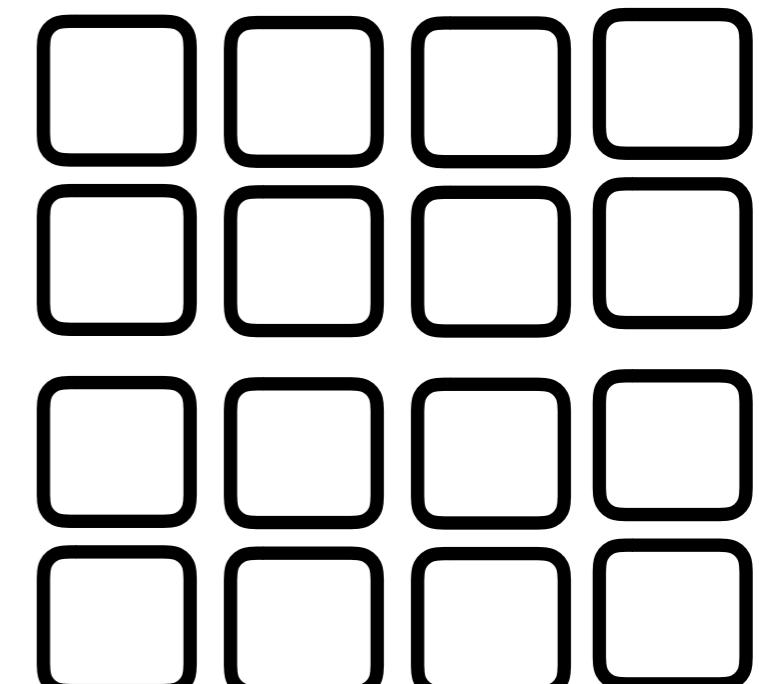
Monolith vs SOA vs Microservices



Monolithic



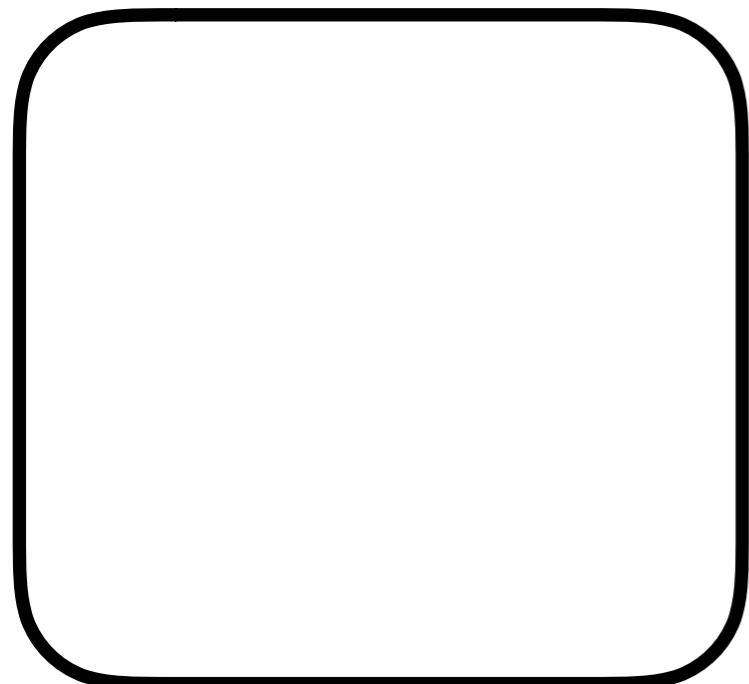
SOA



Microservices

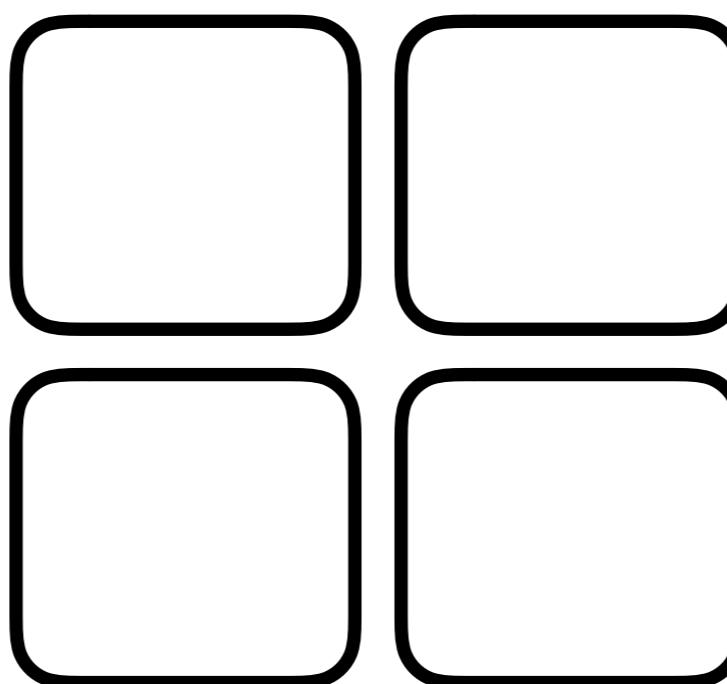


Monolith vs SOA vs Microservices



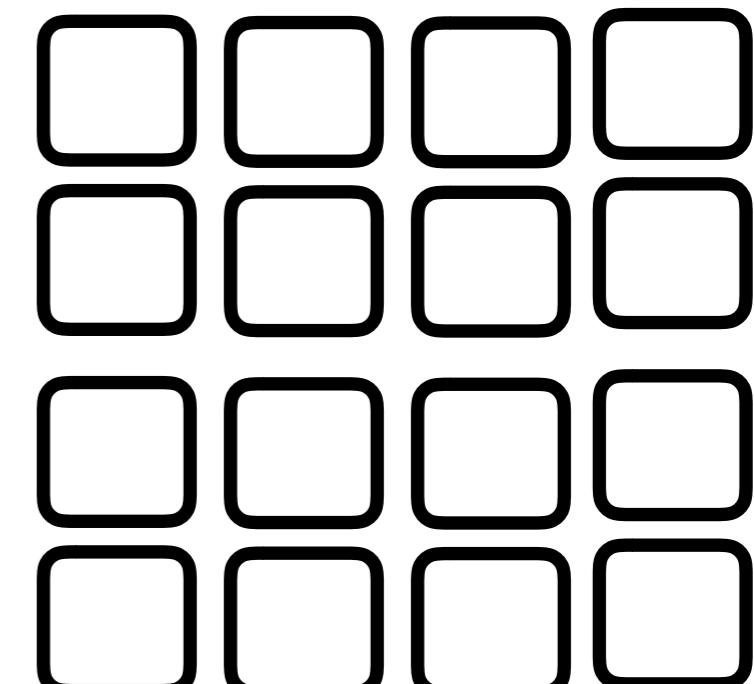
Monolithic

Single unit



SOA

Coarse-grained



Microservices

Fine-grained



SOA vs Microservices



SOA

Business
service

Enterprise
service

Application
service

Infrastructure
service

Business
Users

Share services
Team

Application
Development
Team

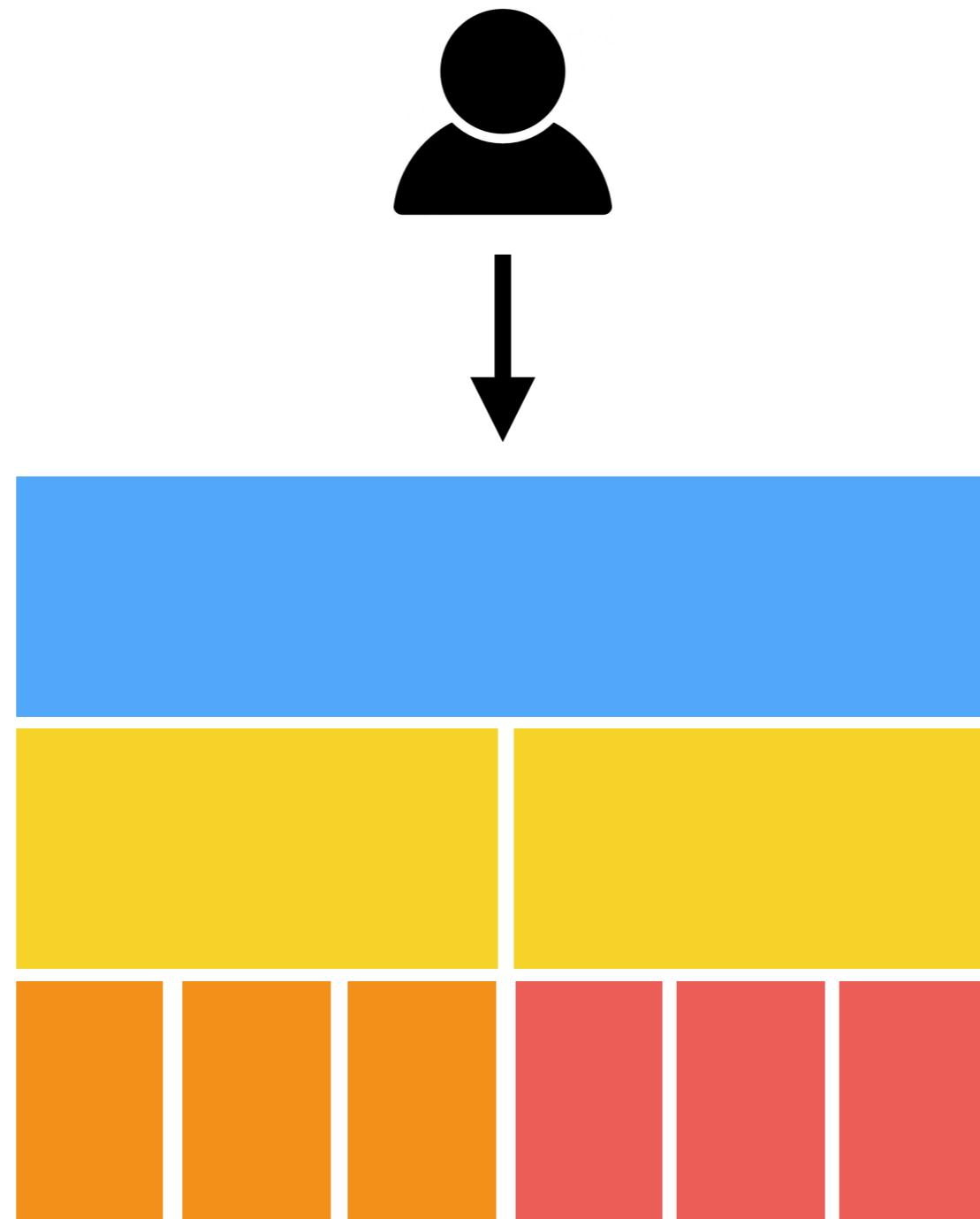
Infrastructure
Services Team



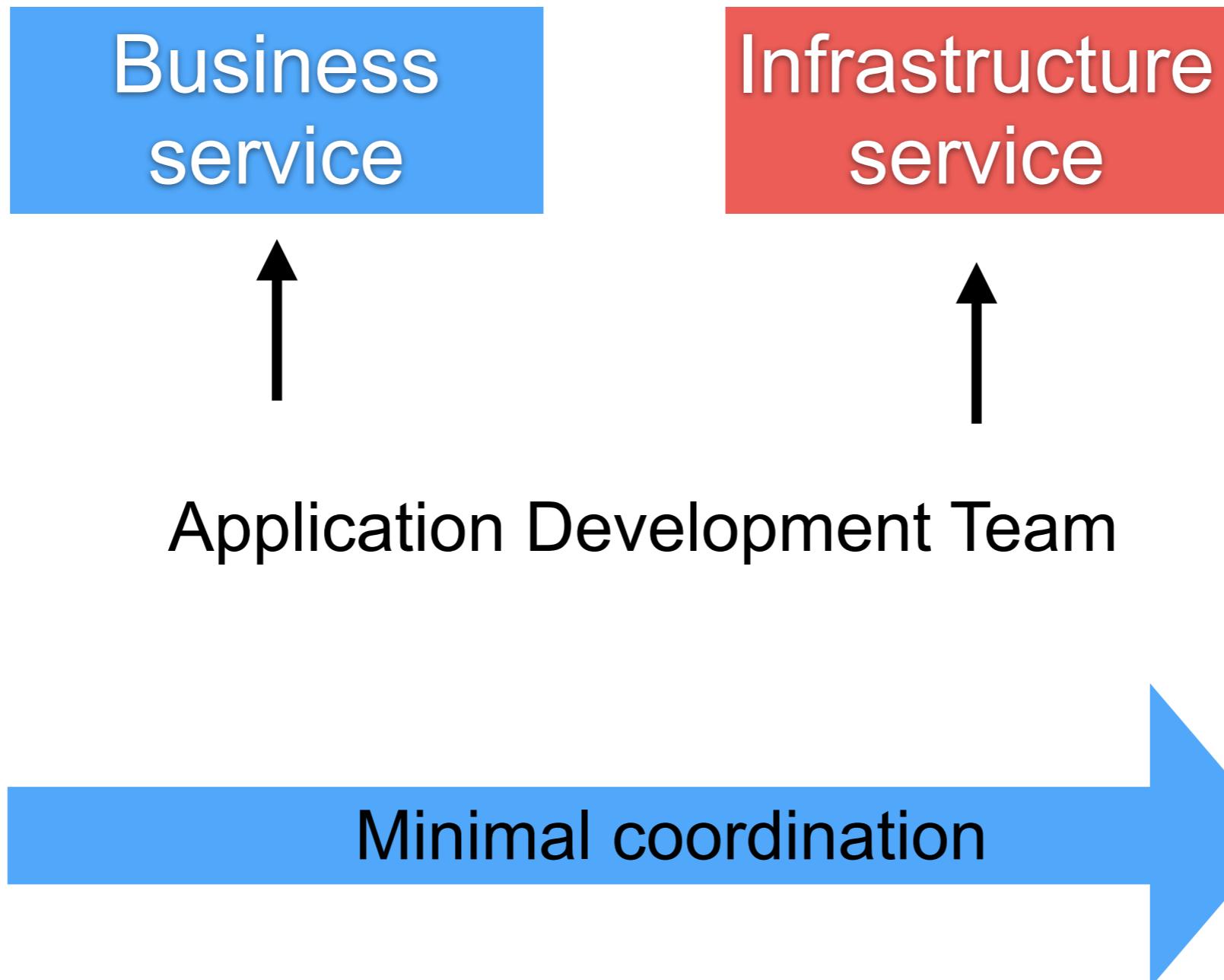
Coordination across all teams



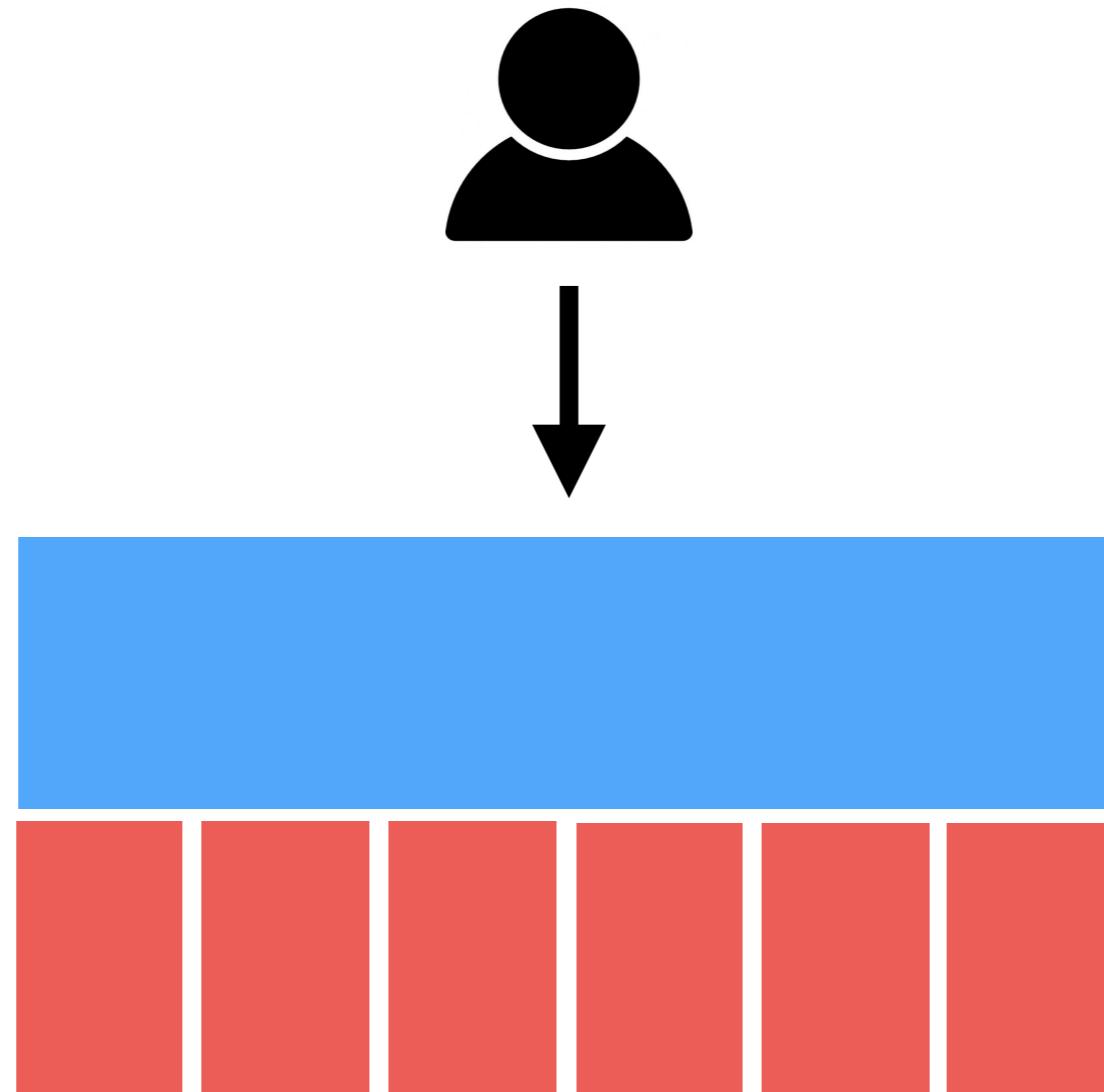
SOA



Microservices



Microservices

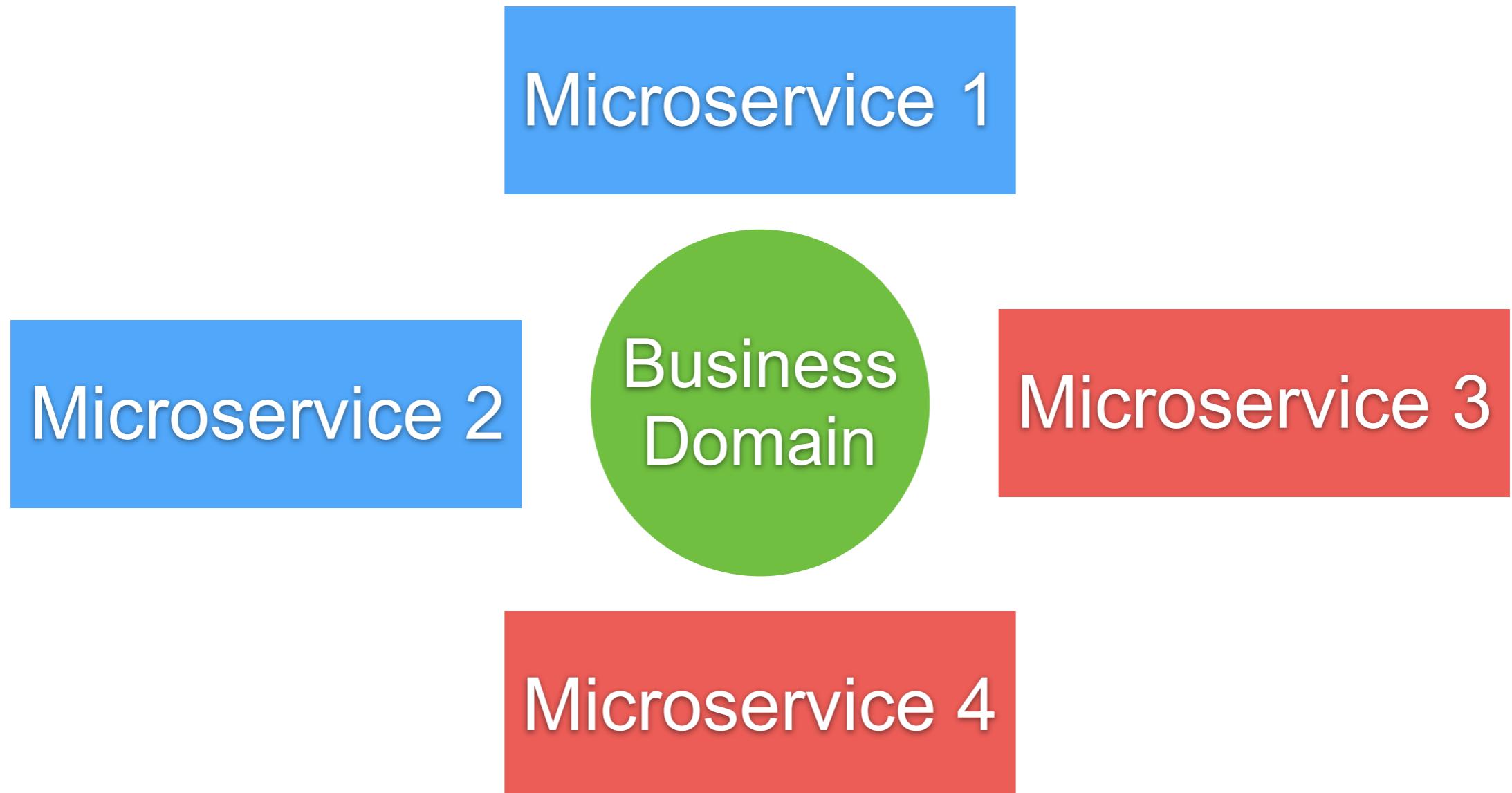


Microservices ?

Architectural style
Collection of small autonomous services
Modelling around a business domain



Model around a business domain



Autonomous Service ?

Culture of automation

High observability

Hide internal implementation details

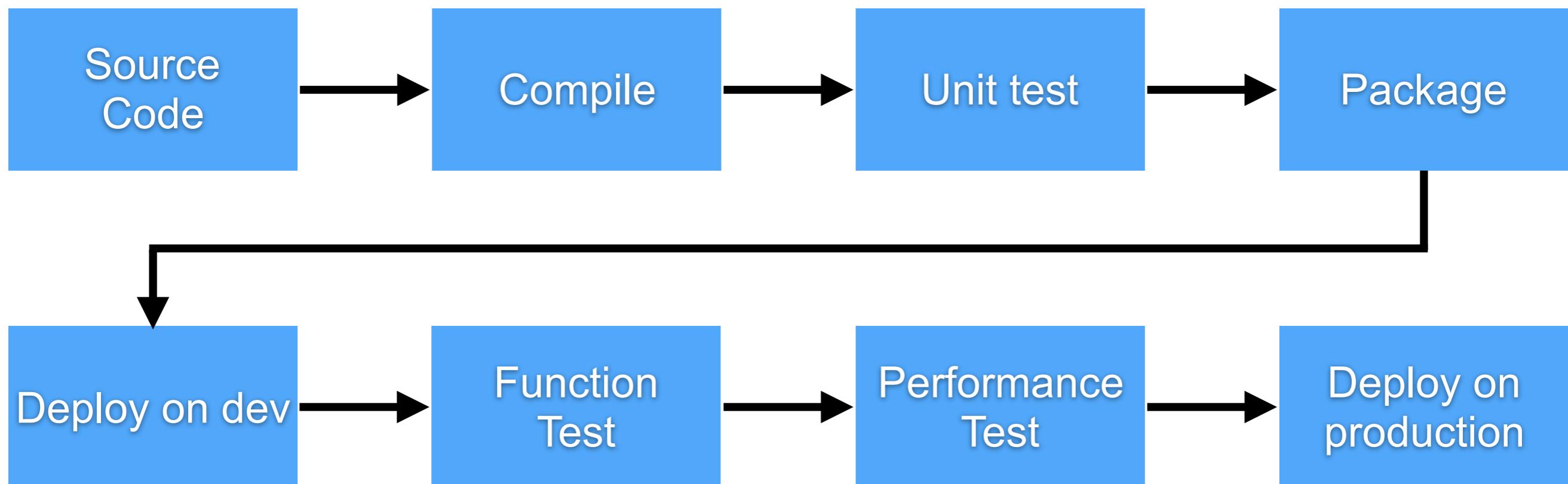
Isolate failure

Deploy independently

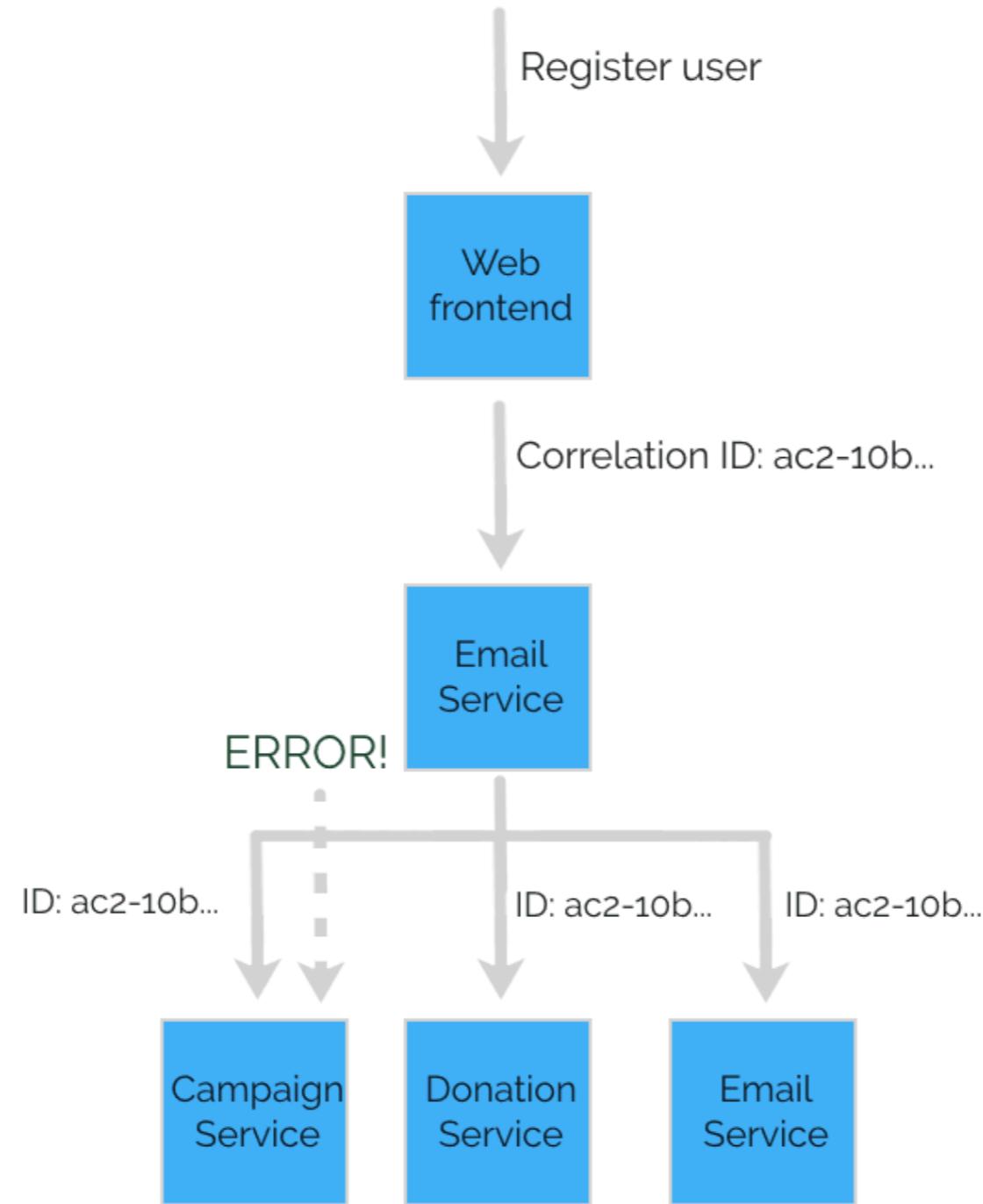
Decentralize all the things



Culture of automation



Highly observability



Highly observability

Centralize logging
Distributed tracing
Metrics



Hide internal implementation

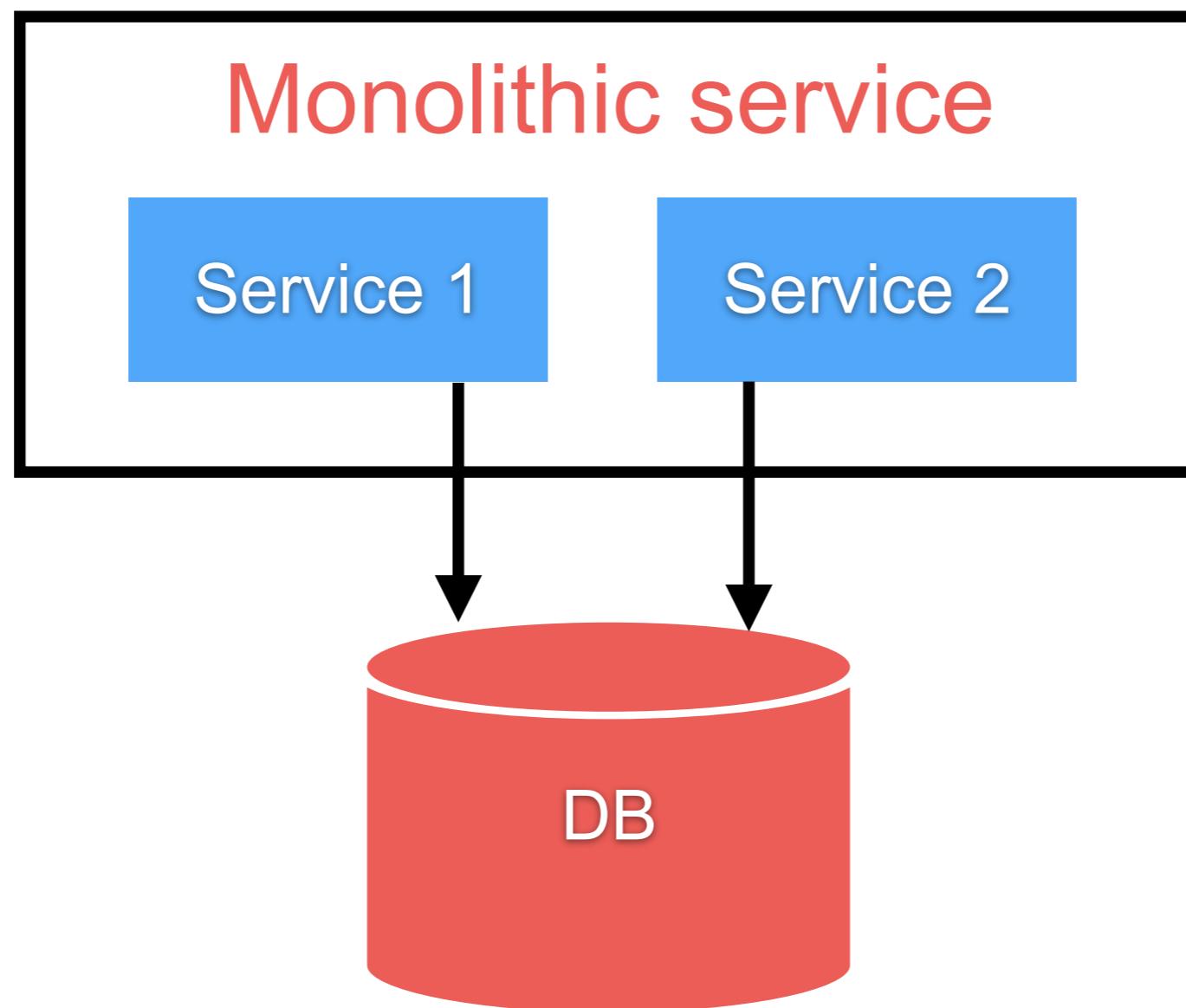
**Maximize the ability of one service to evolve
independently of others**

**Services should hide their database
Avoid tight coupling**



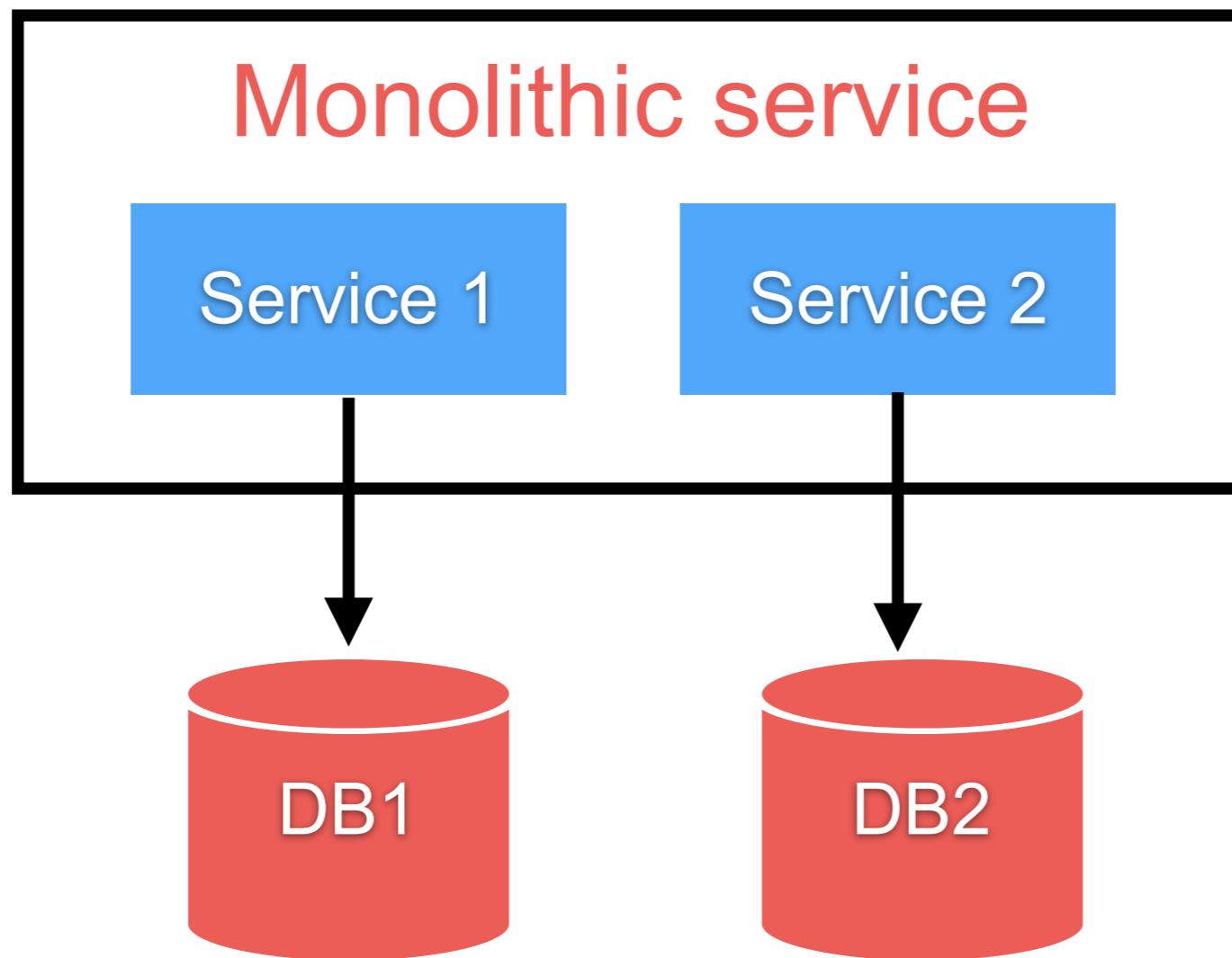
Hide internal implementation

Example with Database



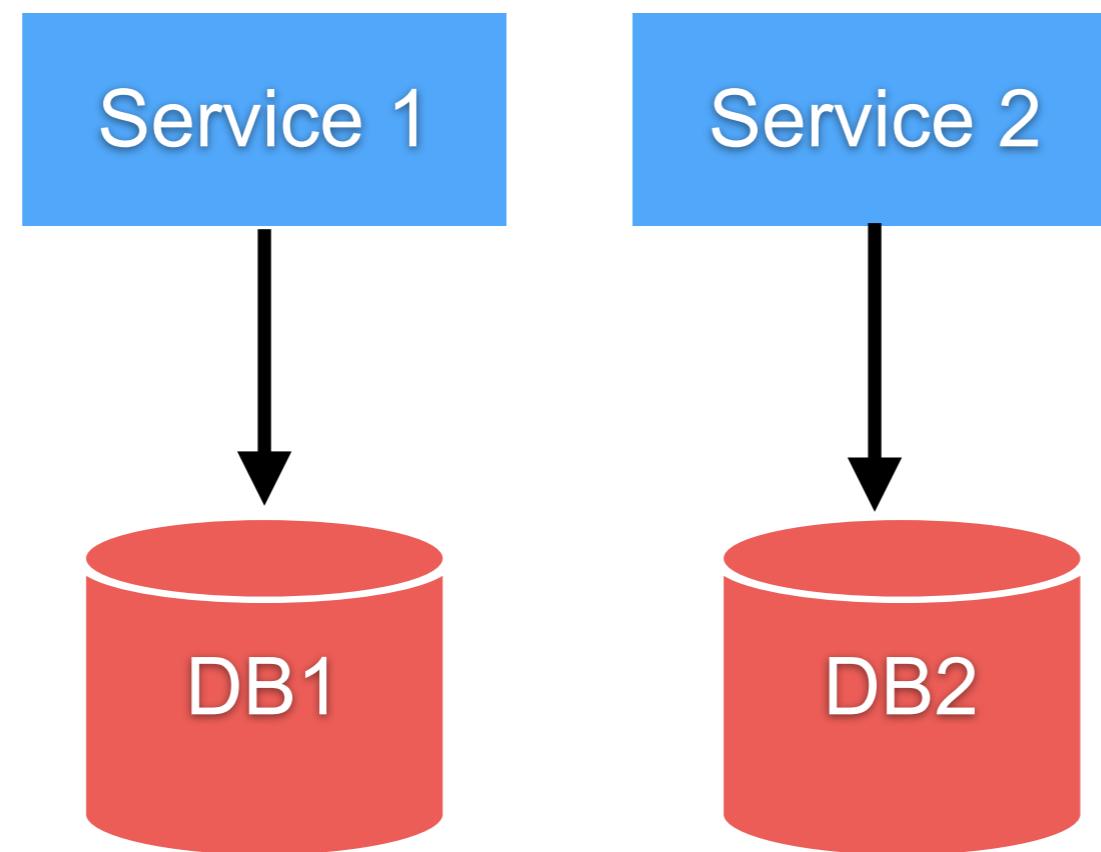
Hide internal implementation

Split database schema



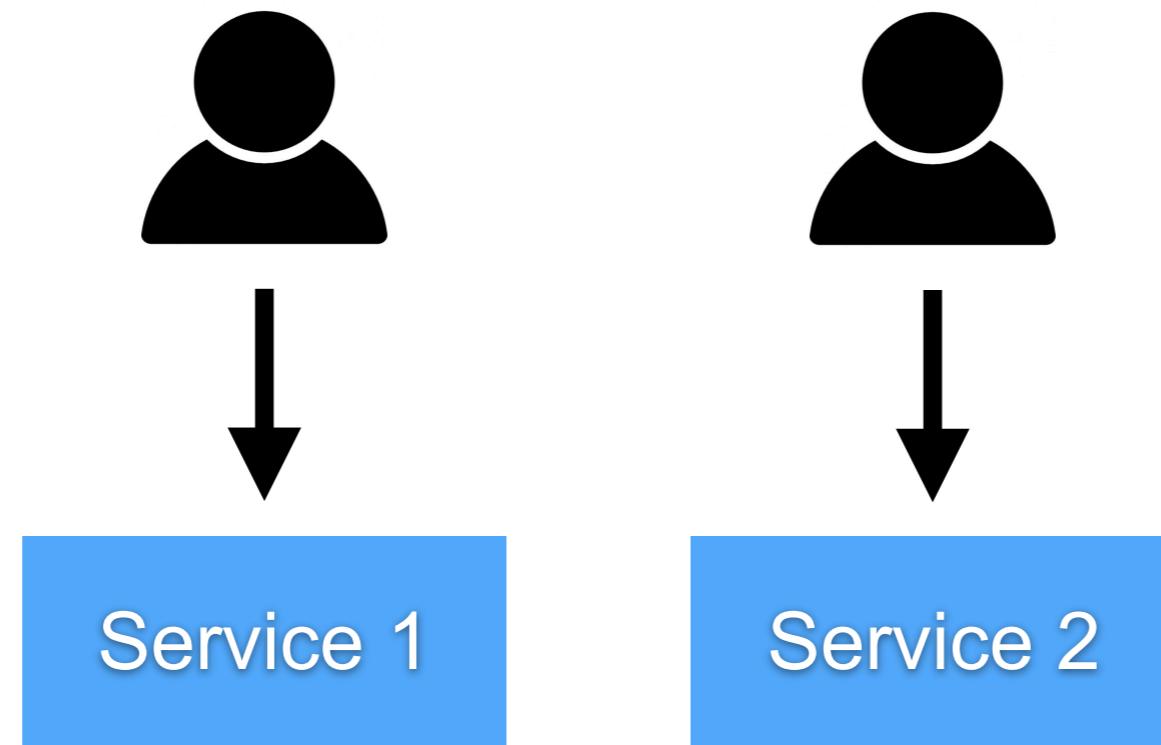
Hide internal implementation

Split application into micro services



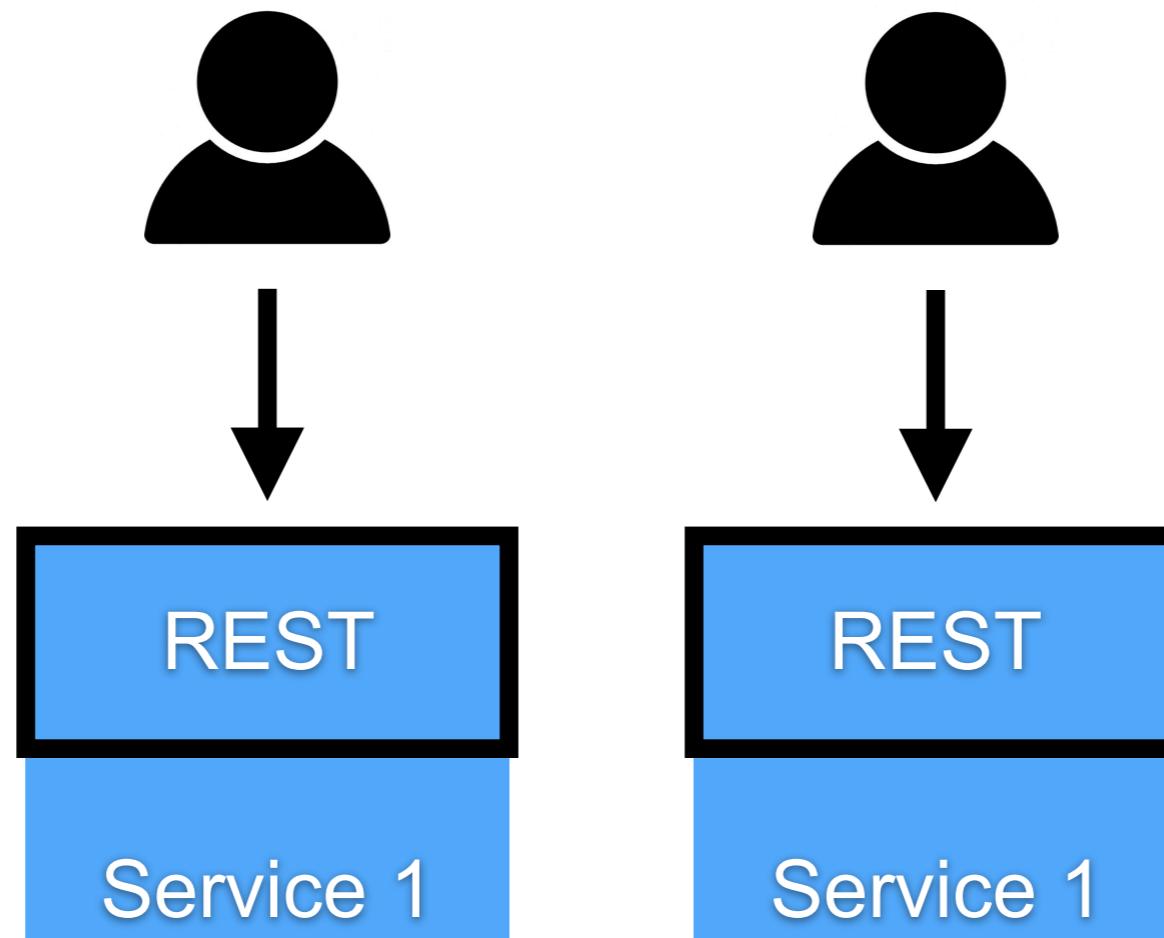
Hide internal implementation

Using specific technology to access services



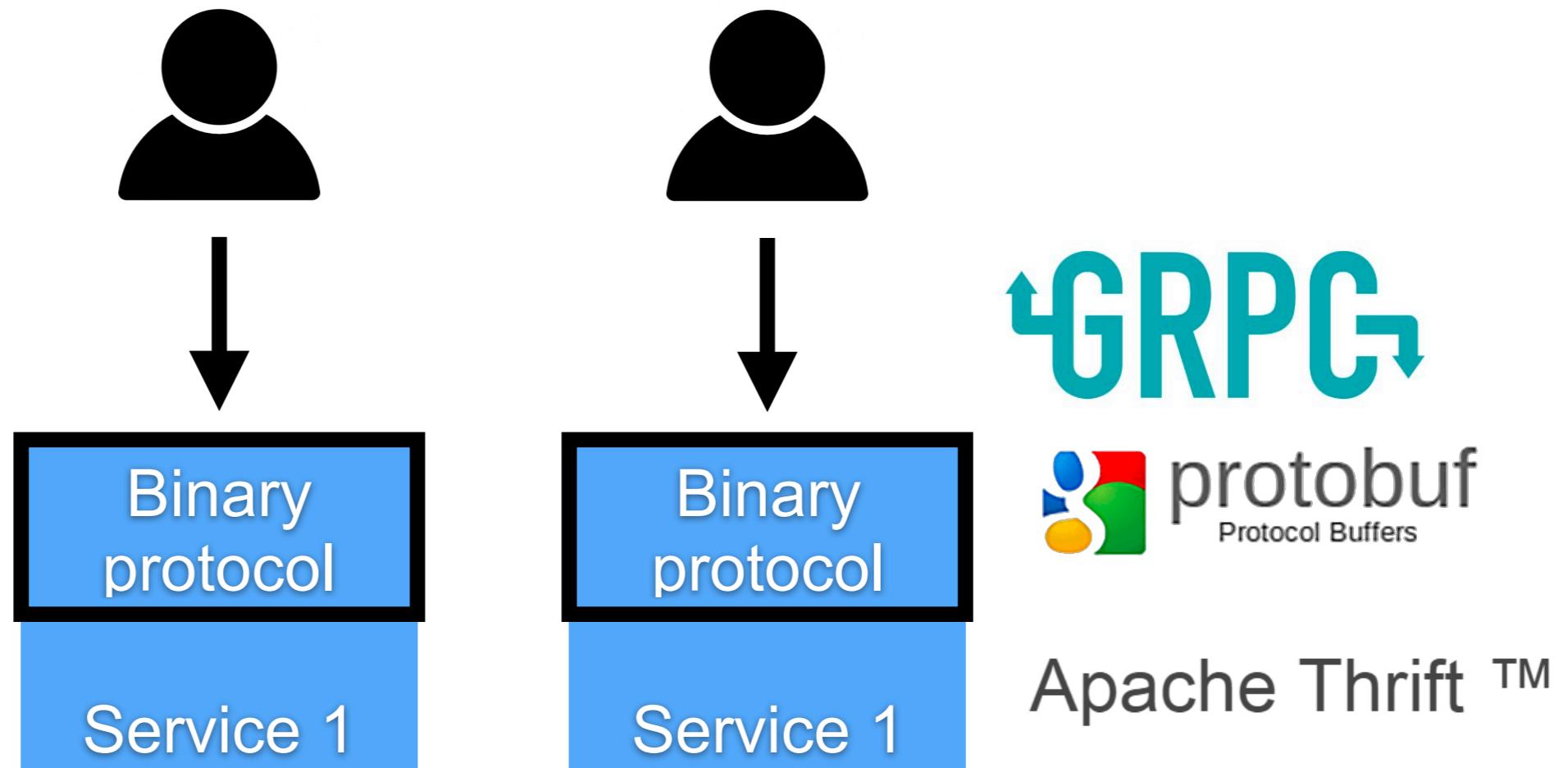
Hide internal implementation

Using specific technology to access services



Hide internal implementation

Using specific technology to access services



Isolated failure

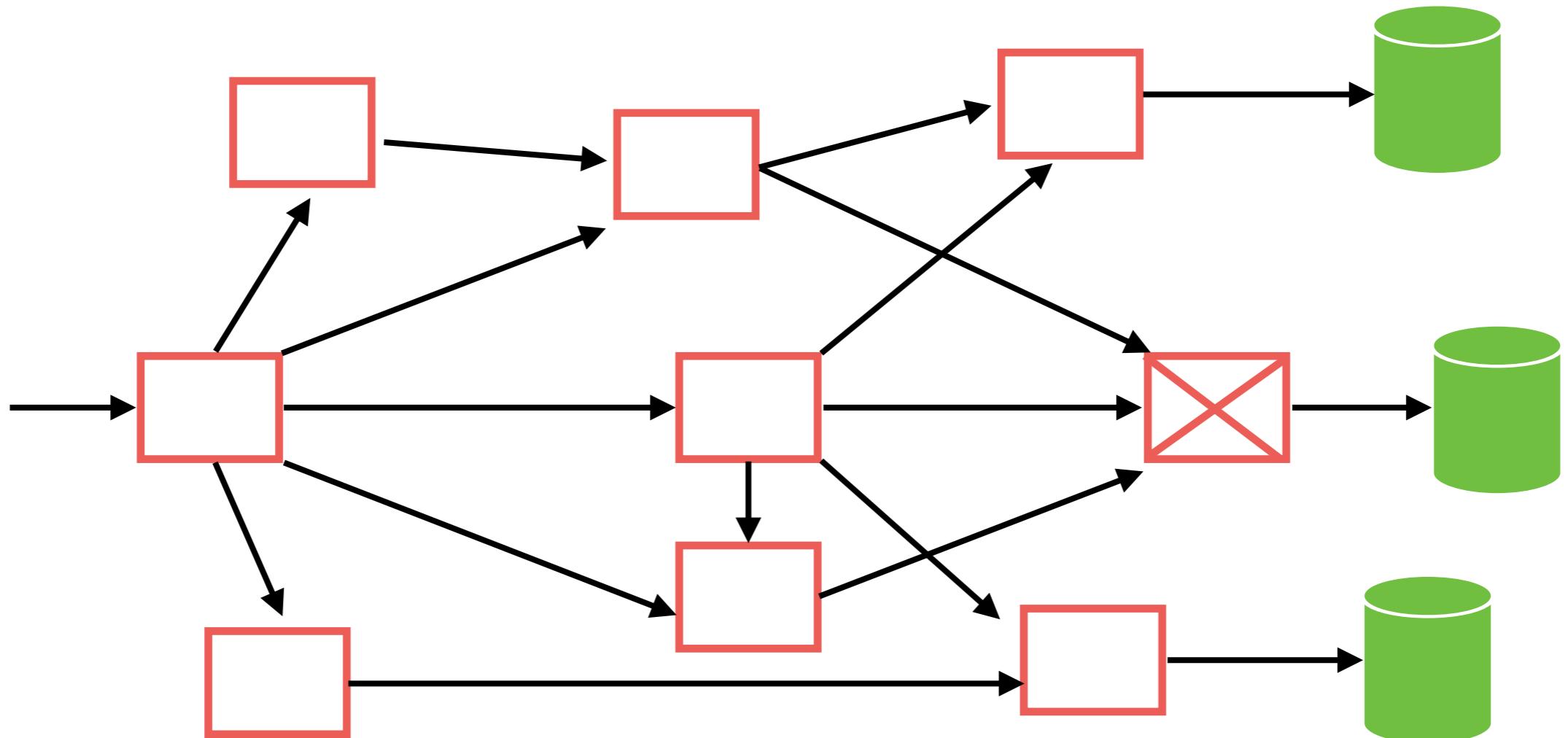
Need more resilient than monolithic system

Plan for failures in all parts of system

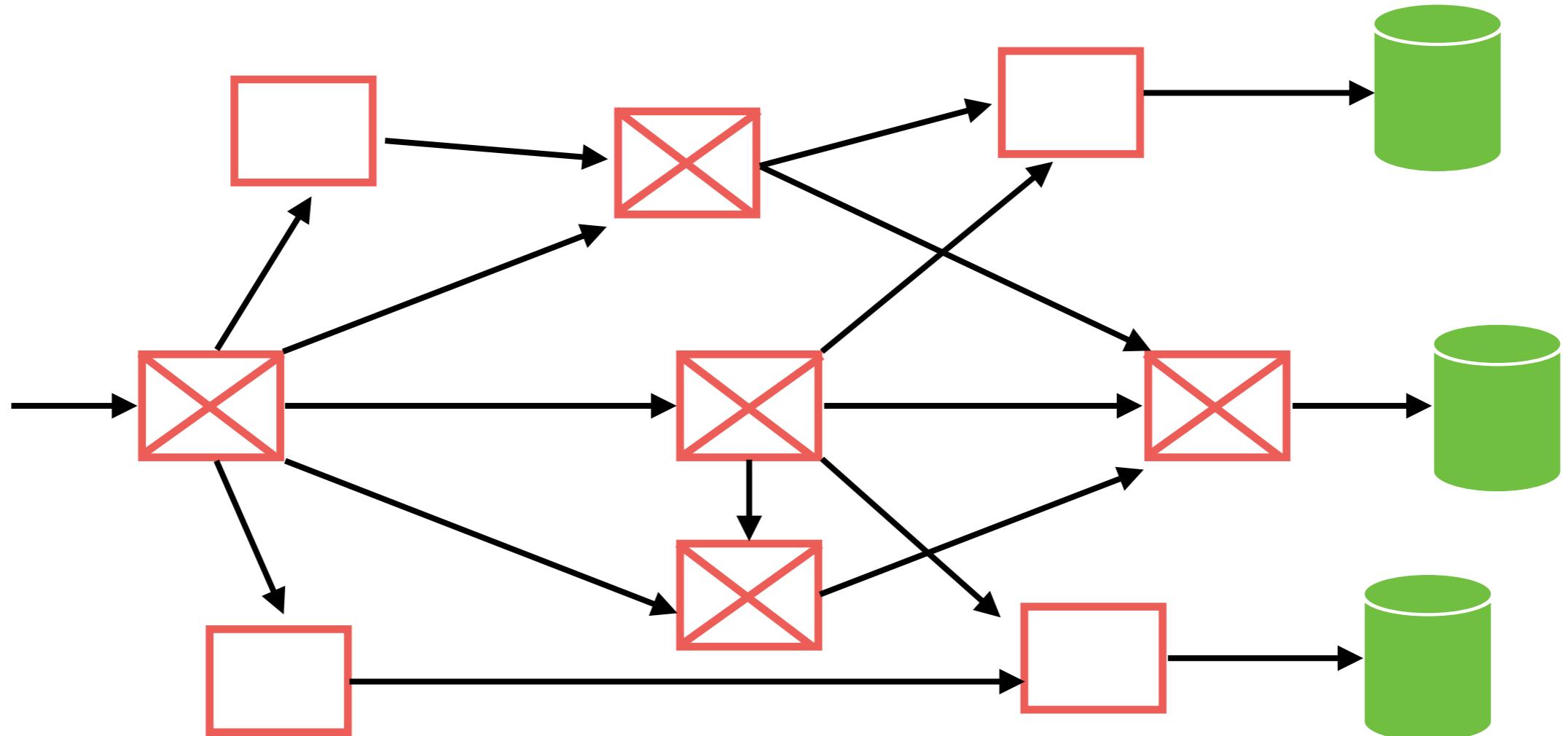
Reduce cascading failure of service



Failure !!



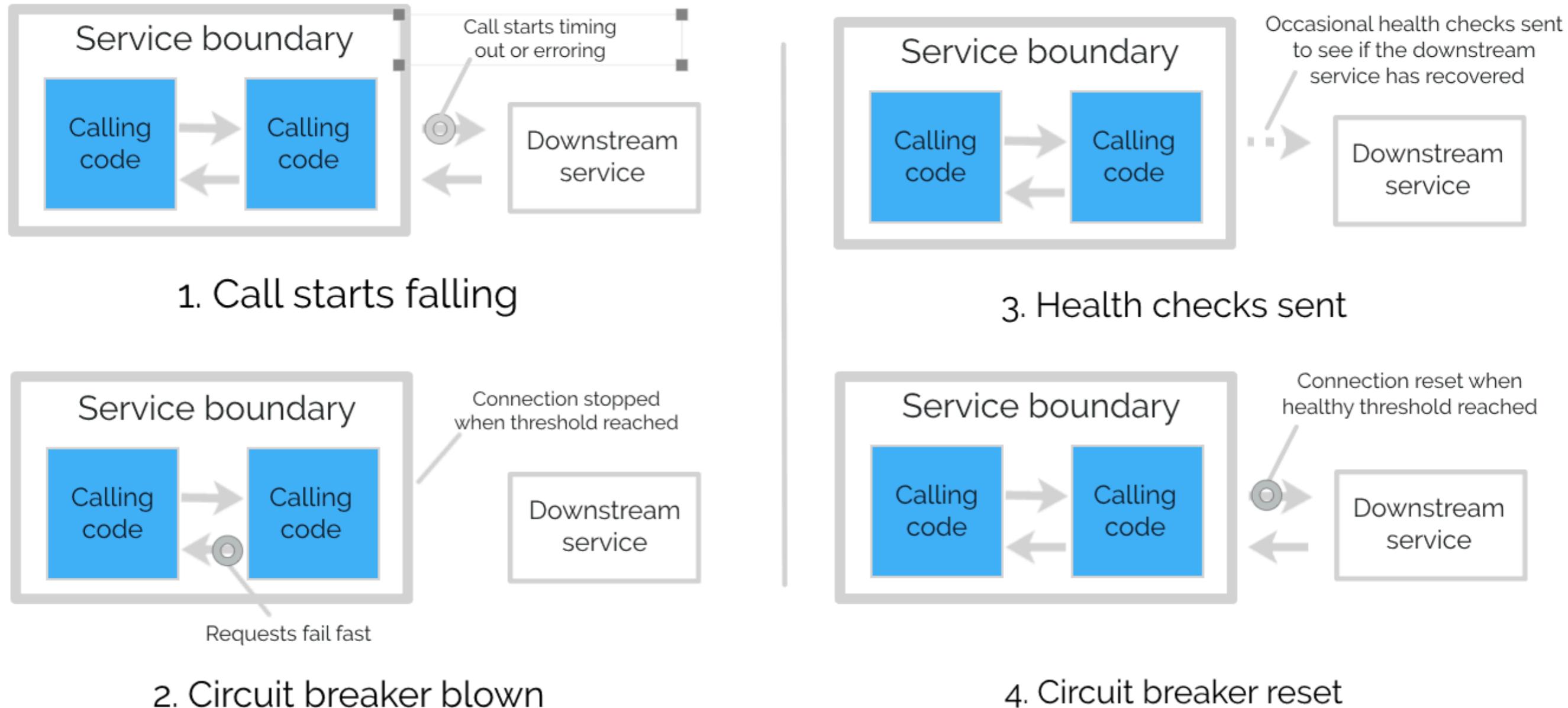
Cascading Failure !!



Circuit Breaker



Circuit Breaker

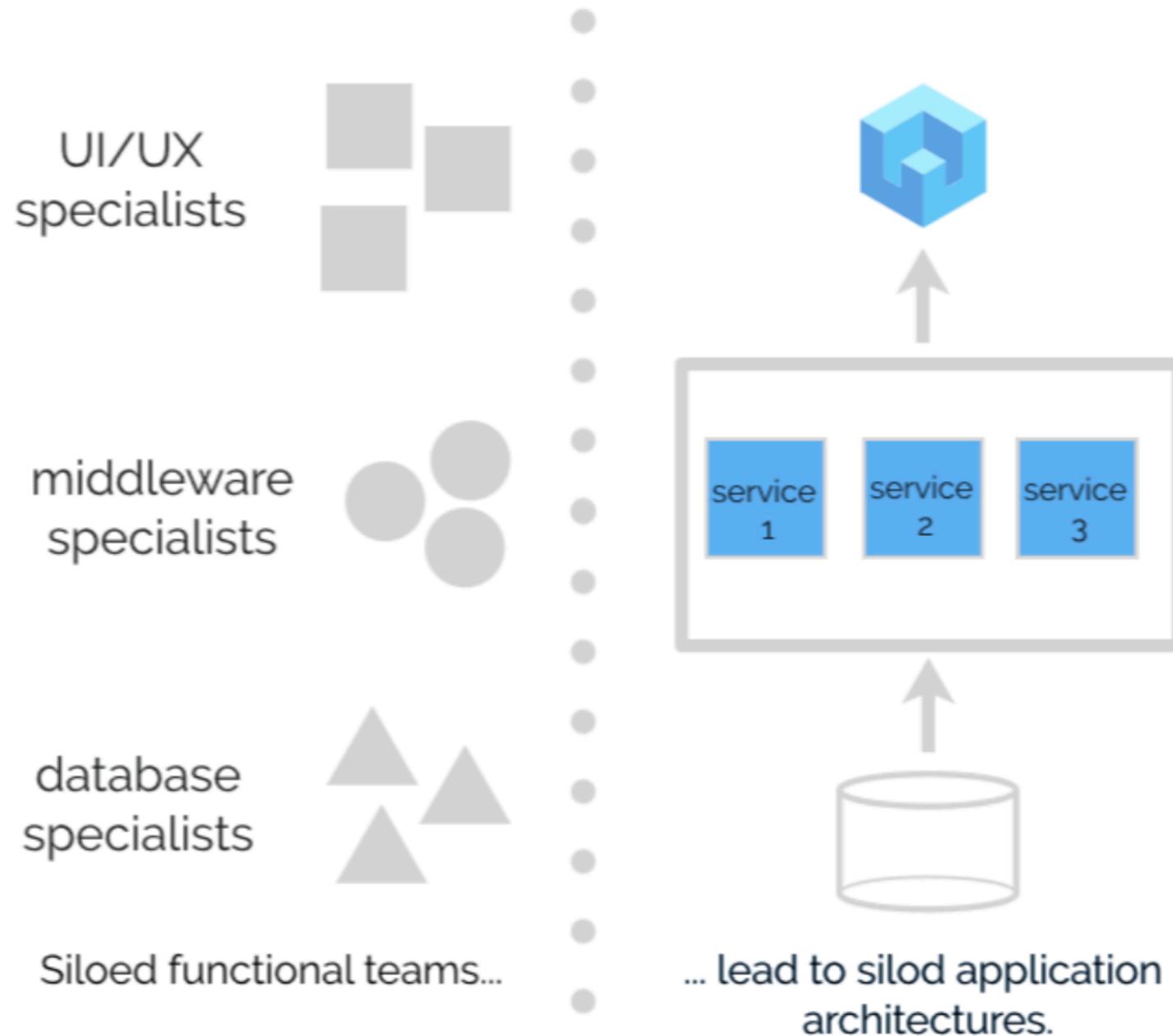


Decentralize all the things

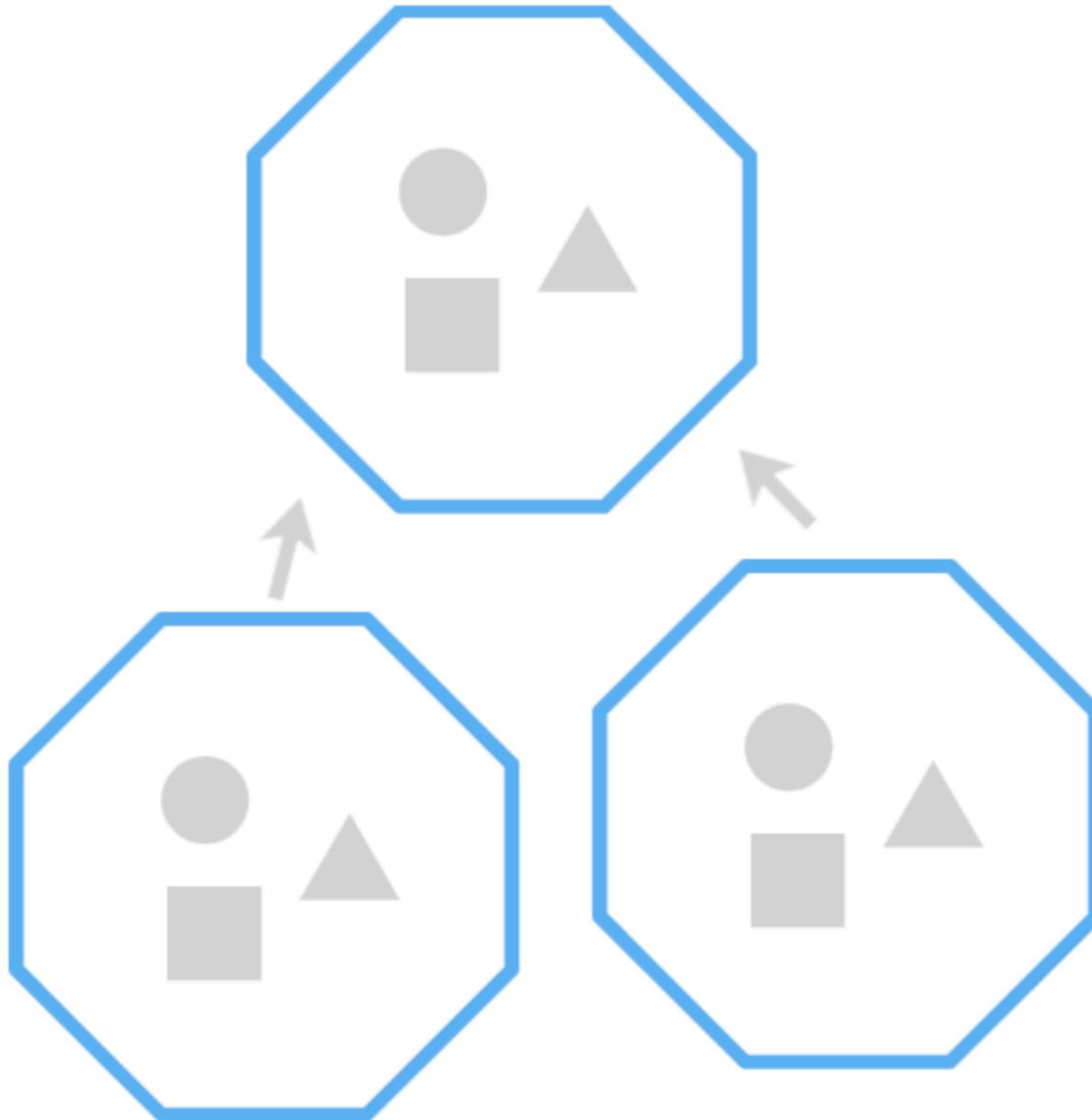
Need self-service
Cross-functional team
Team own their services



Traditional structure



Cross-functional team



Workshop

Design your services



1. Understand requirements

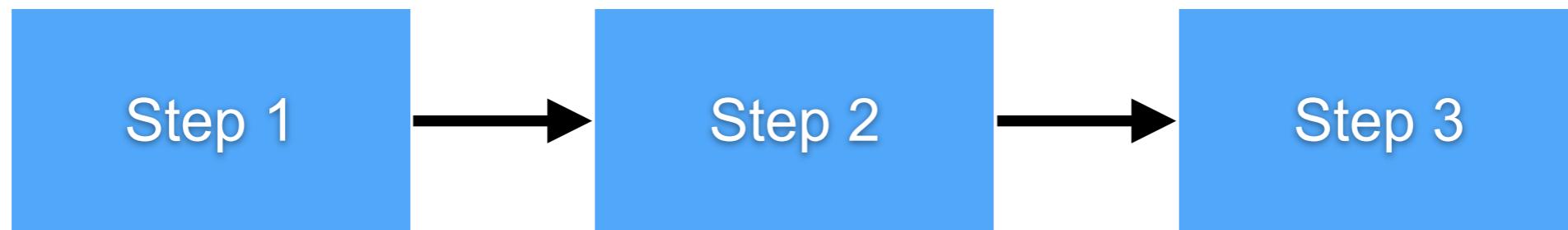
Listen

Ask question

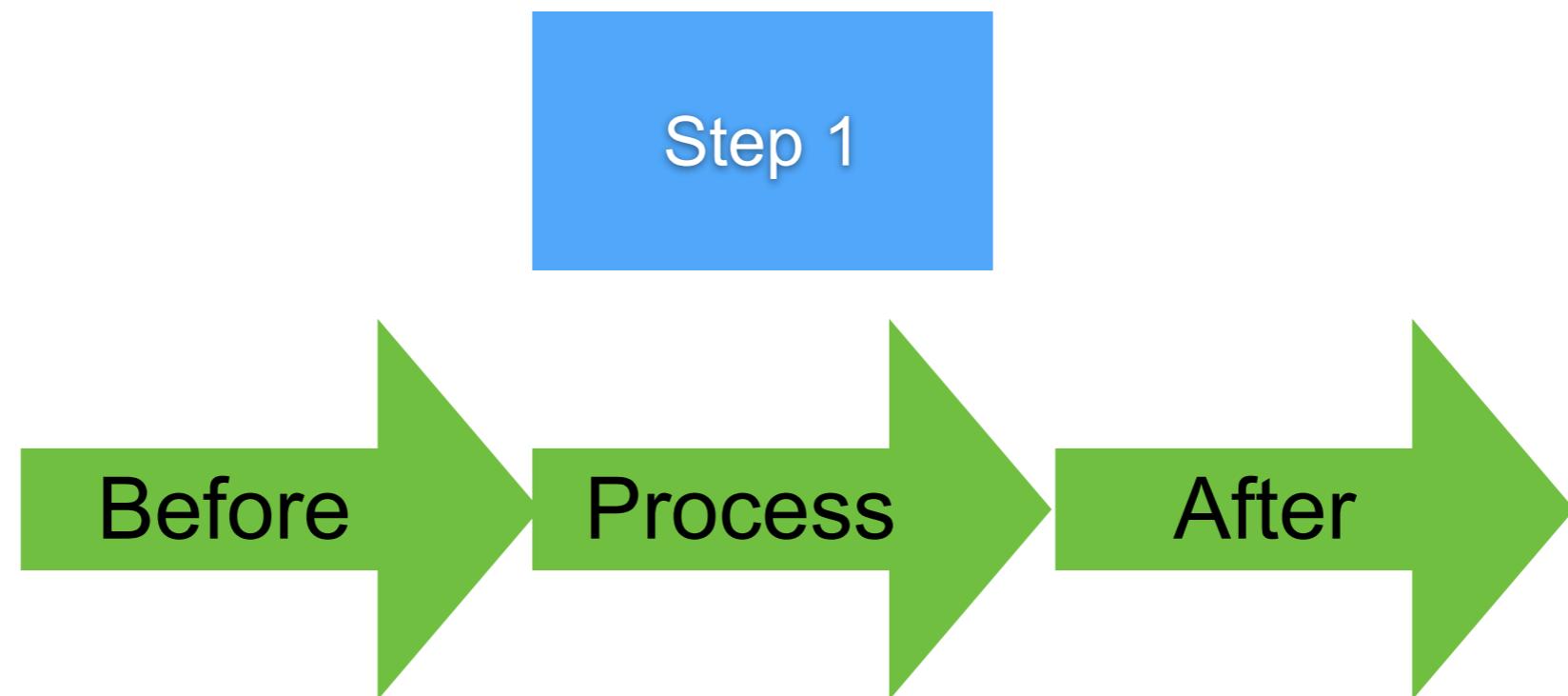
With scenario and example



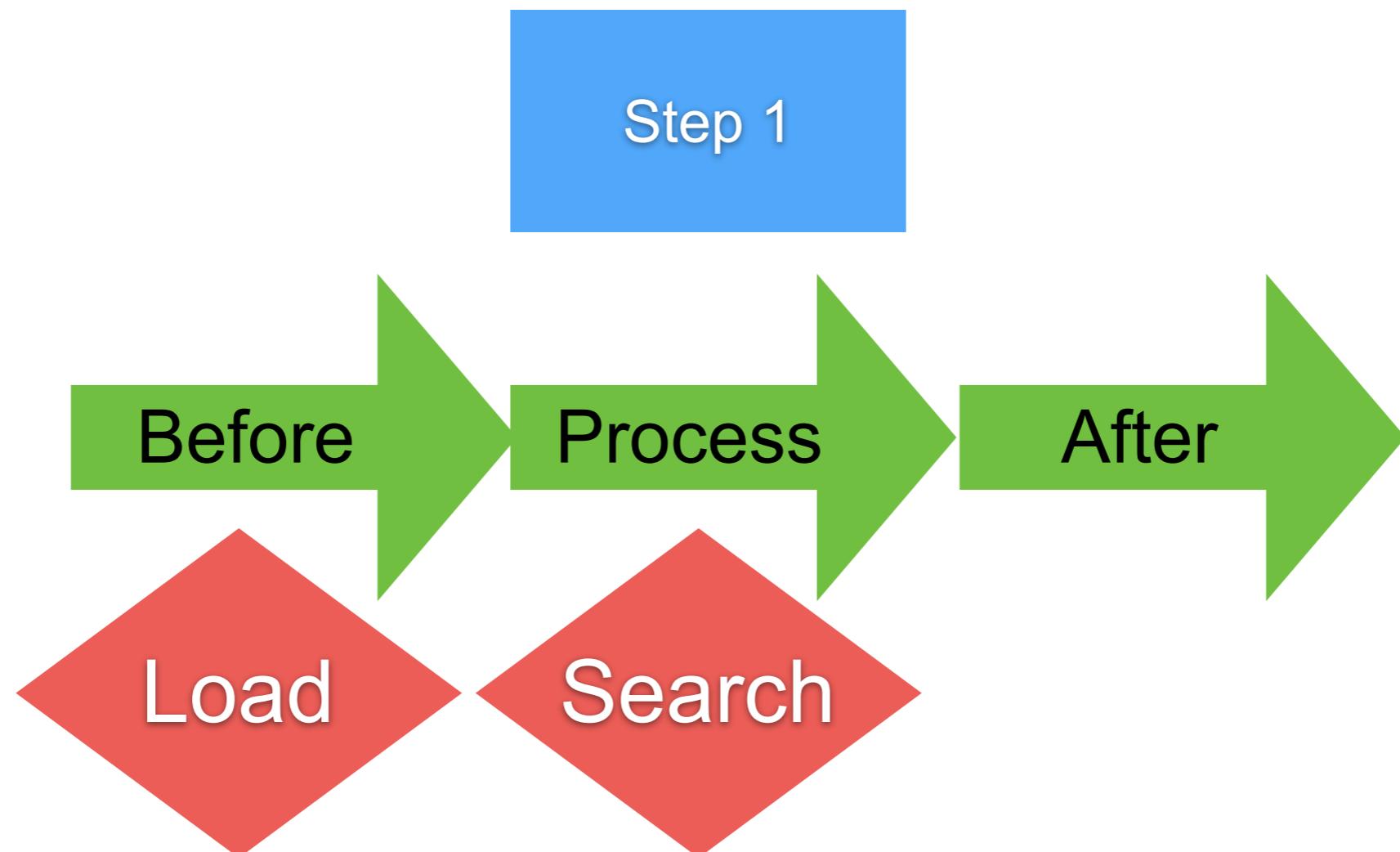
2. User flow/process



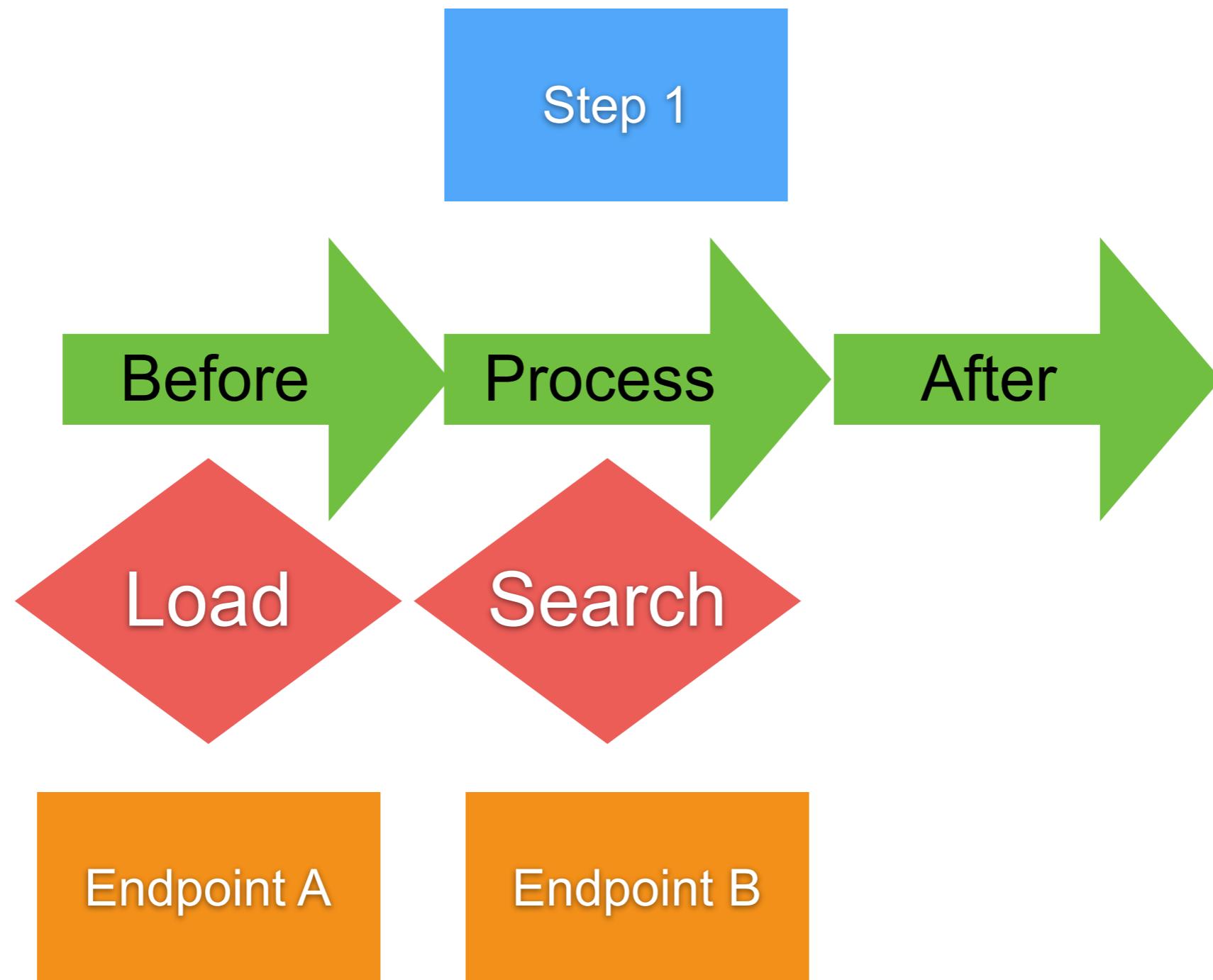
3. Define events/action



3. Define events/action



4. Define endpoint in each events

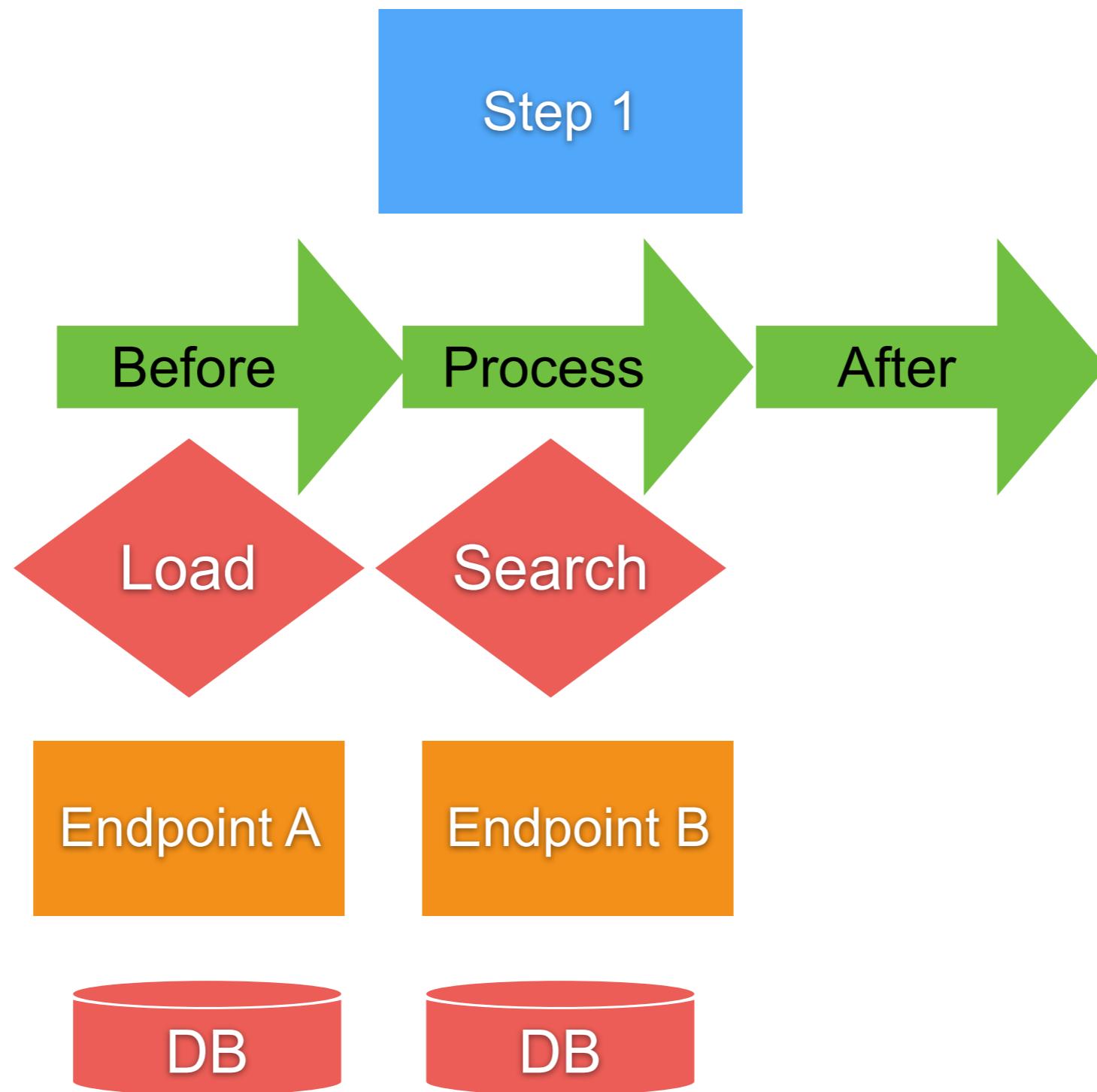


Endpoint details

Property	Example
Name	GetUserById
HTTP method	GET
Path	/user/<id>
Request	JSON format
Response	JSON format



5. Define database/data store



Start workshop



Homework

