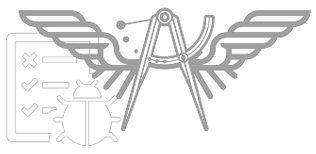
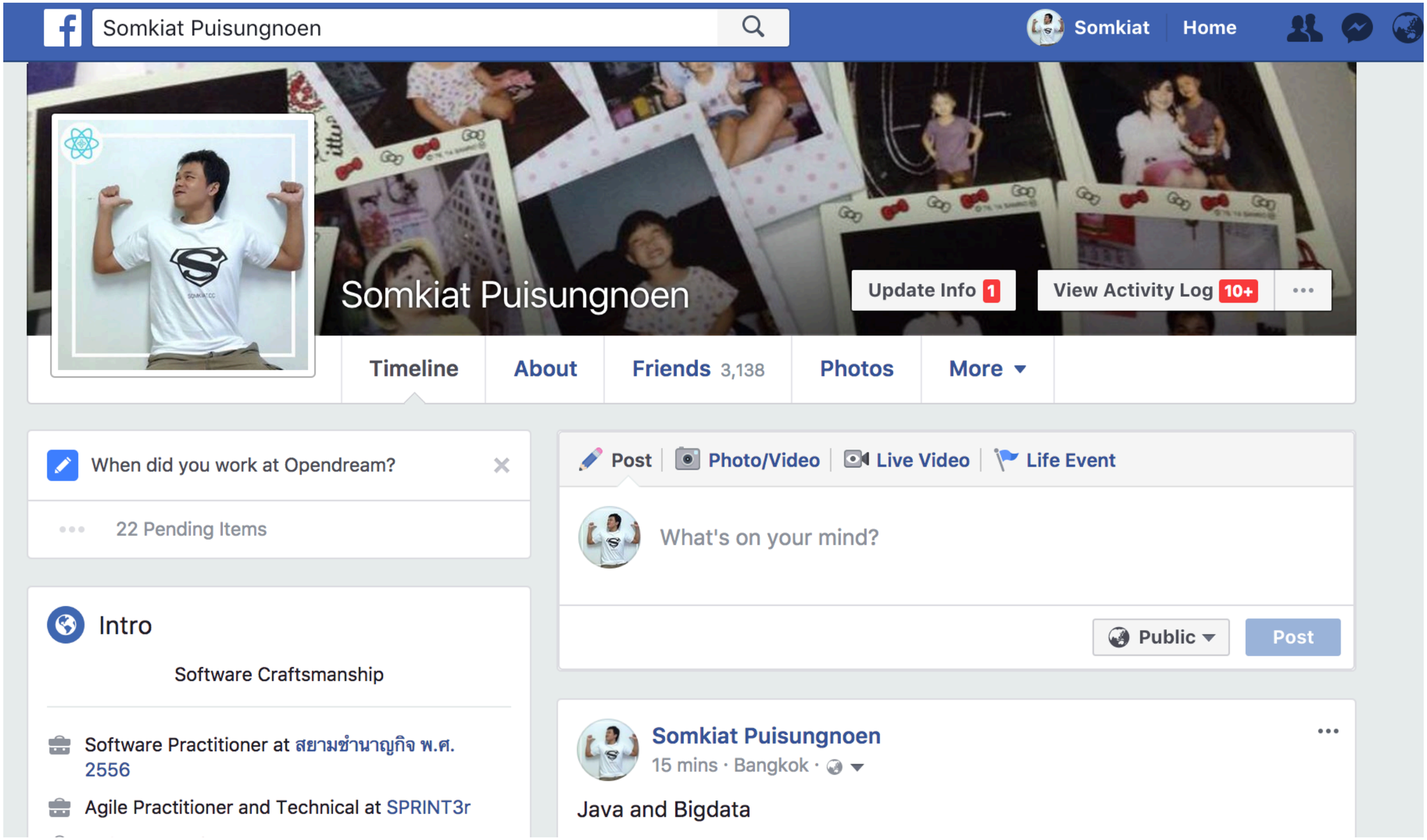
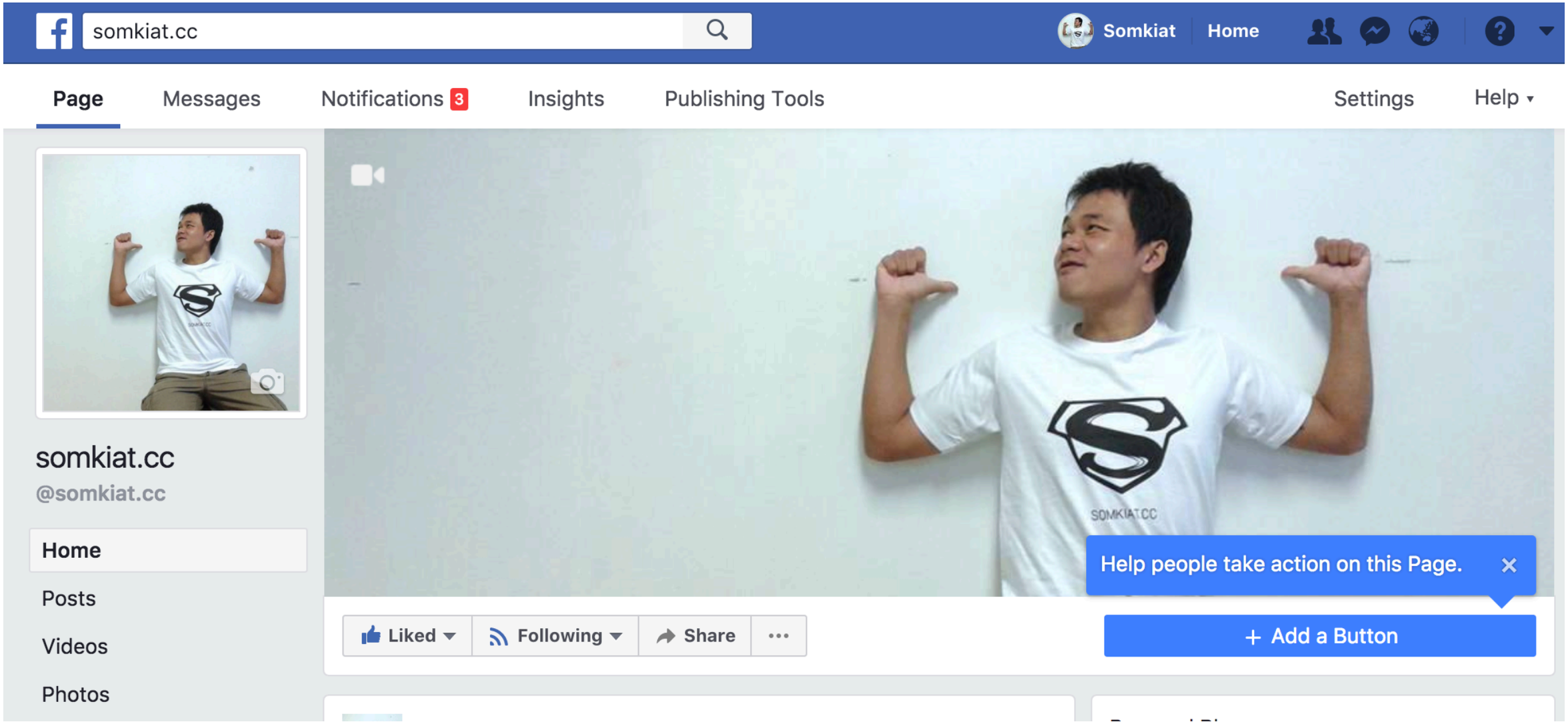
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Design Services/Microservices



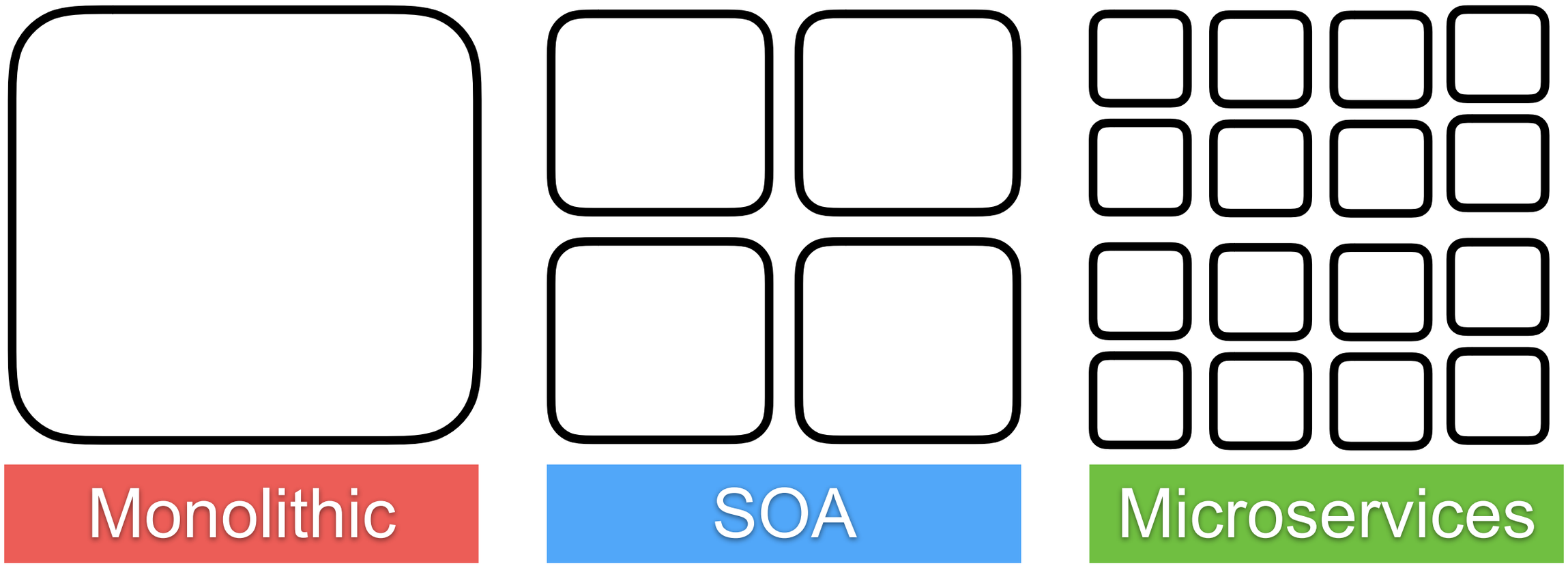




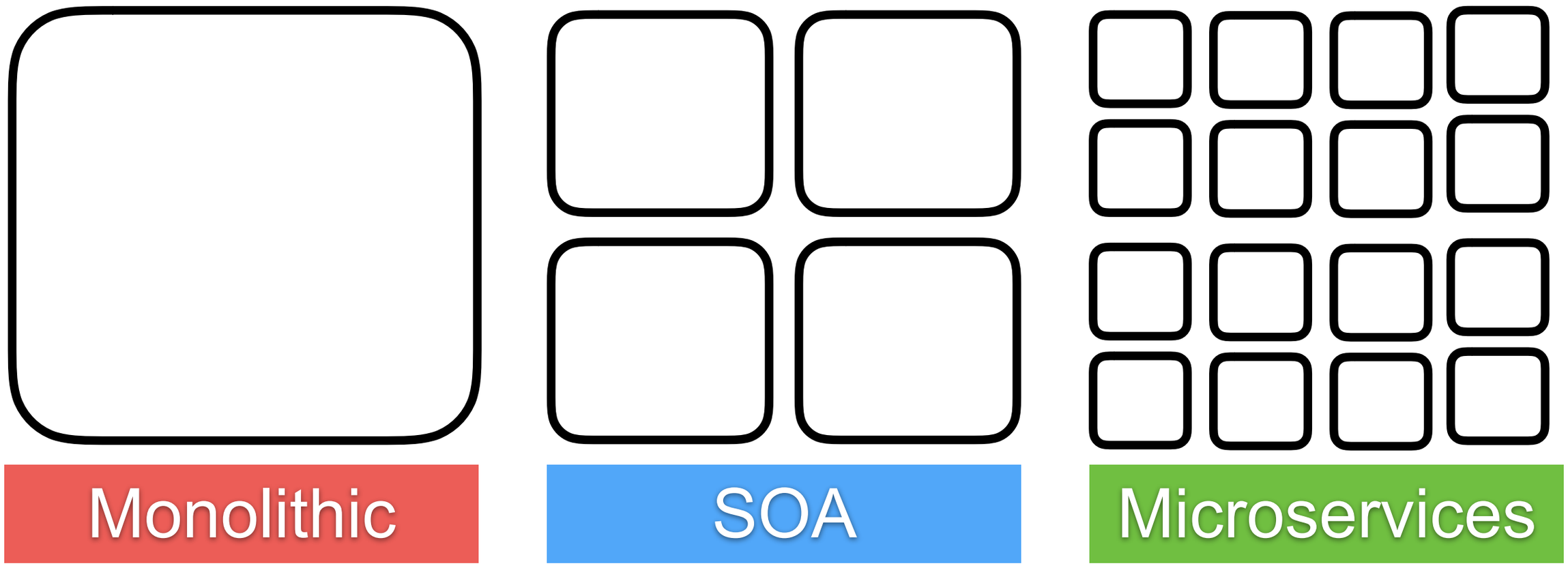
https://github.com/up1/course_microservice_kmitl_2019

Let’s start

Monolith vs SOA vs Microservies



Monolith vs SOA vs Microservies



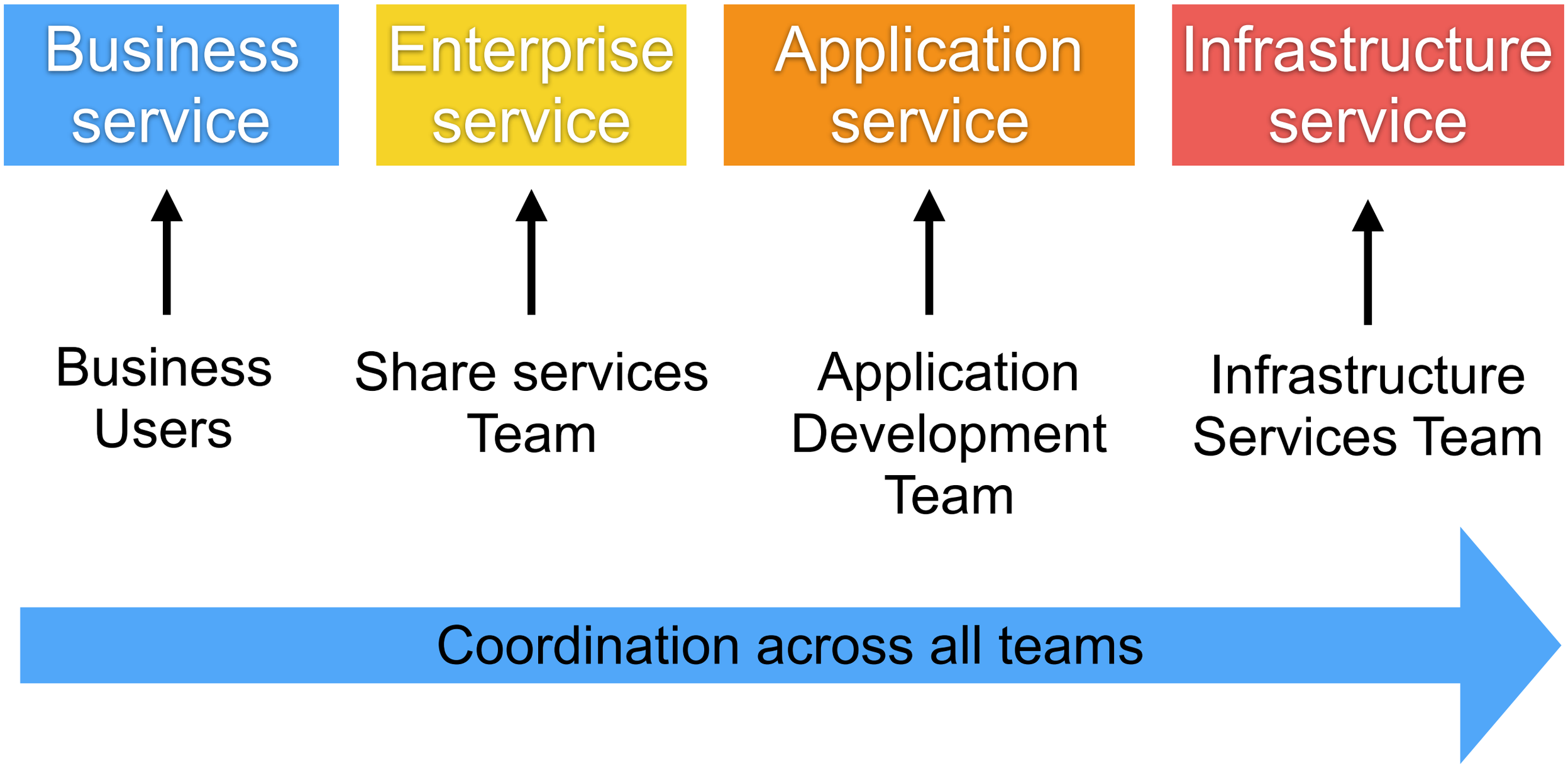
Single unit

Coarse-grained

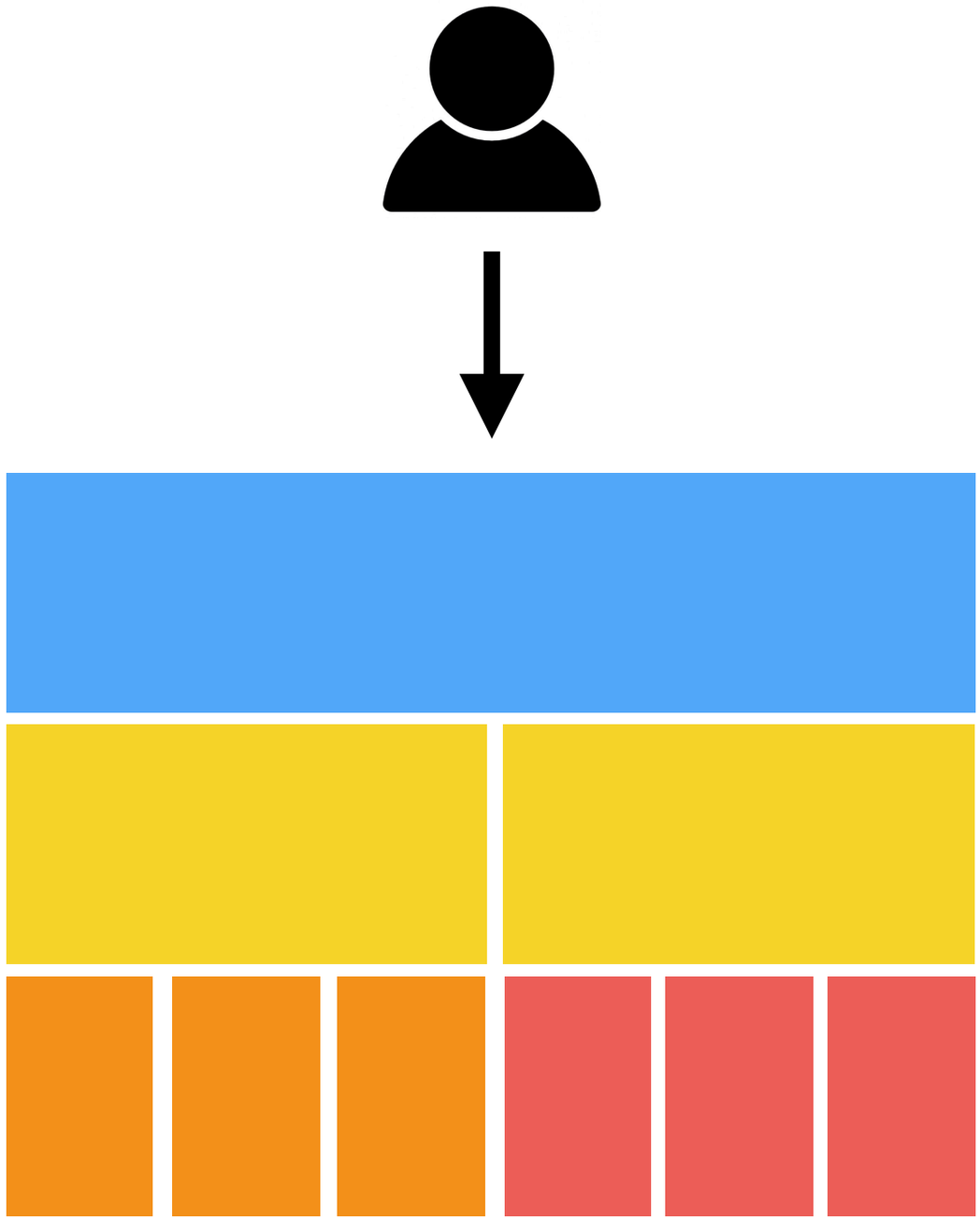
Fine-grained

SOA vs Microservices

SOA

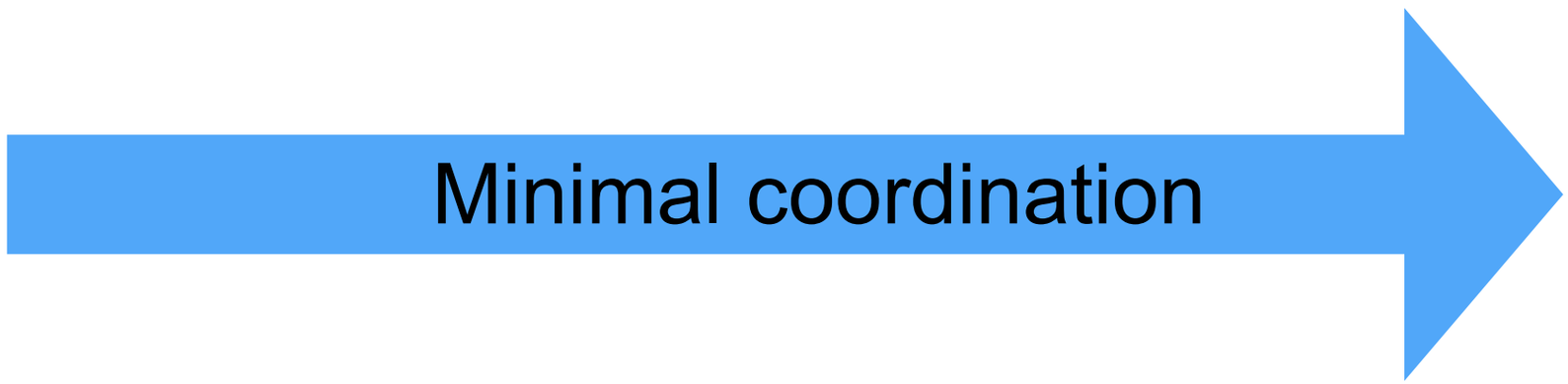


SOA

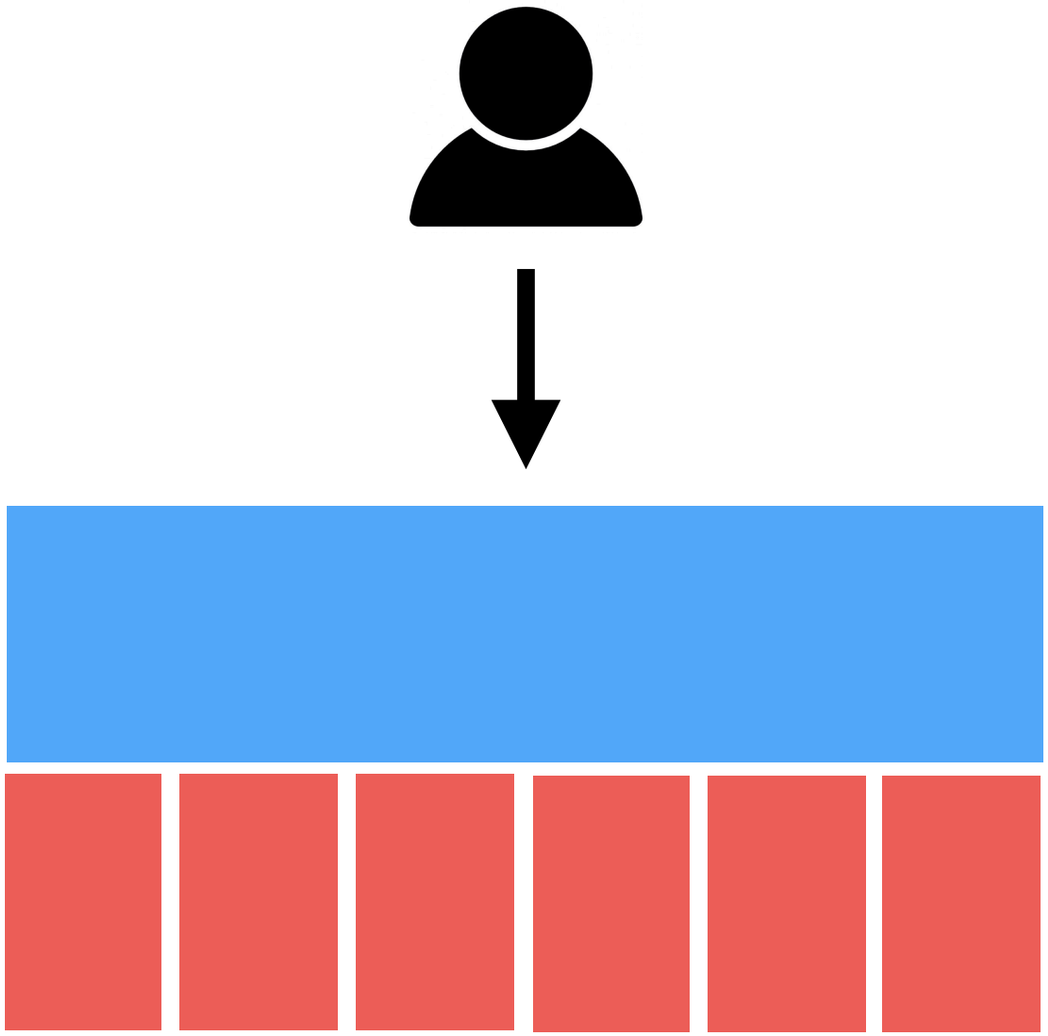


Microservices

Application Development Team



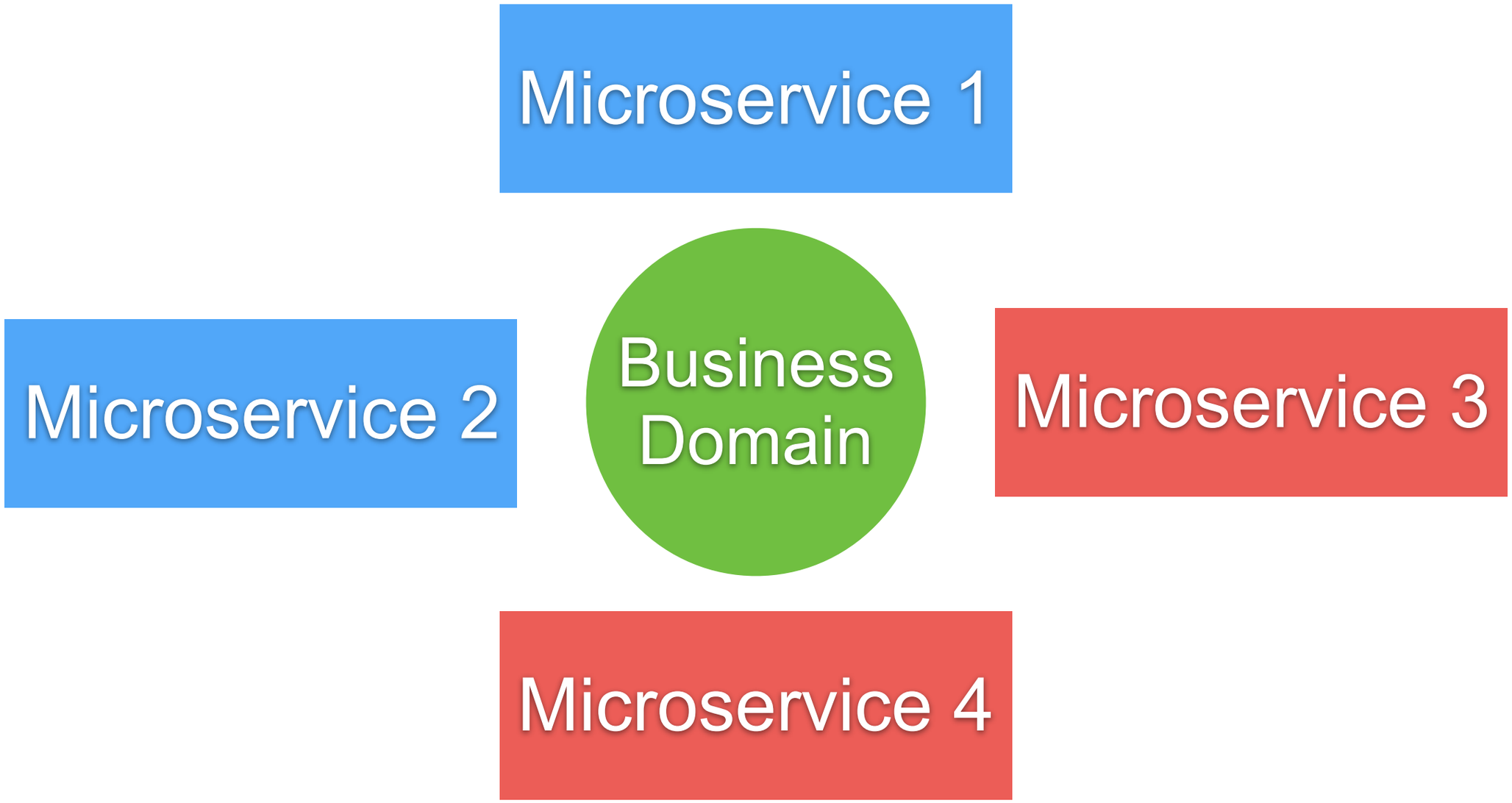
Microservices



Microservices ?

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

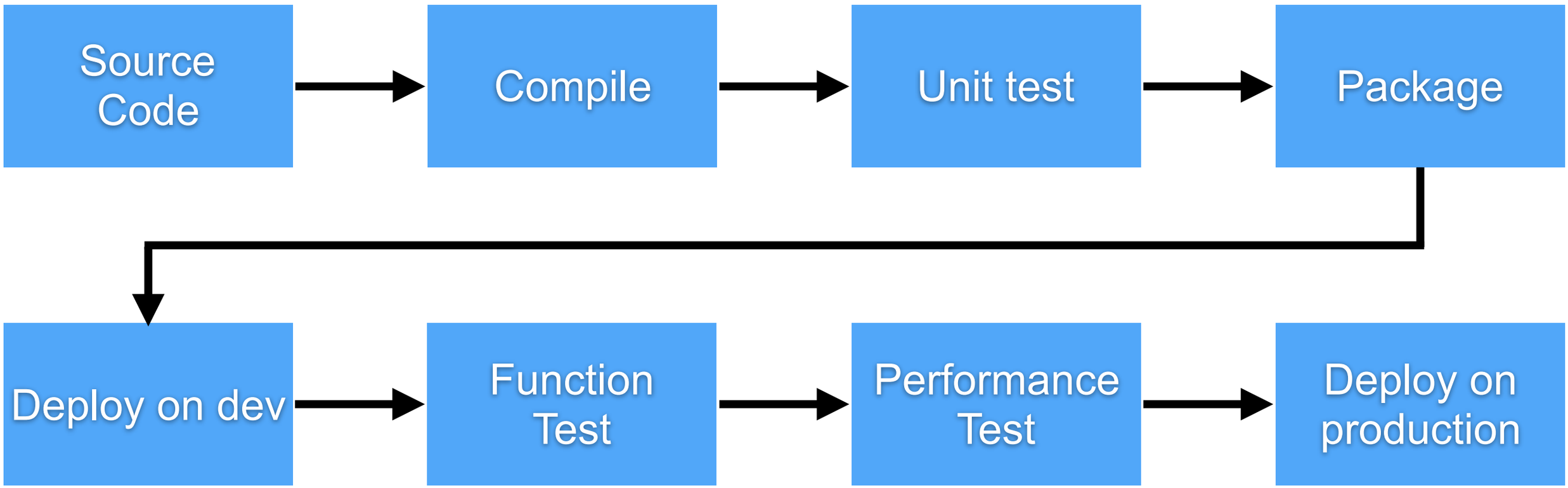
Model around a business domain



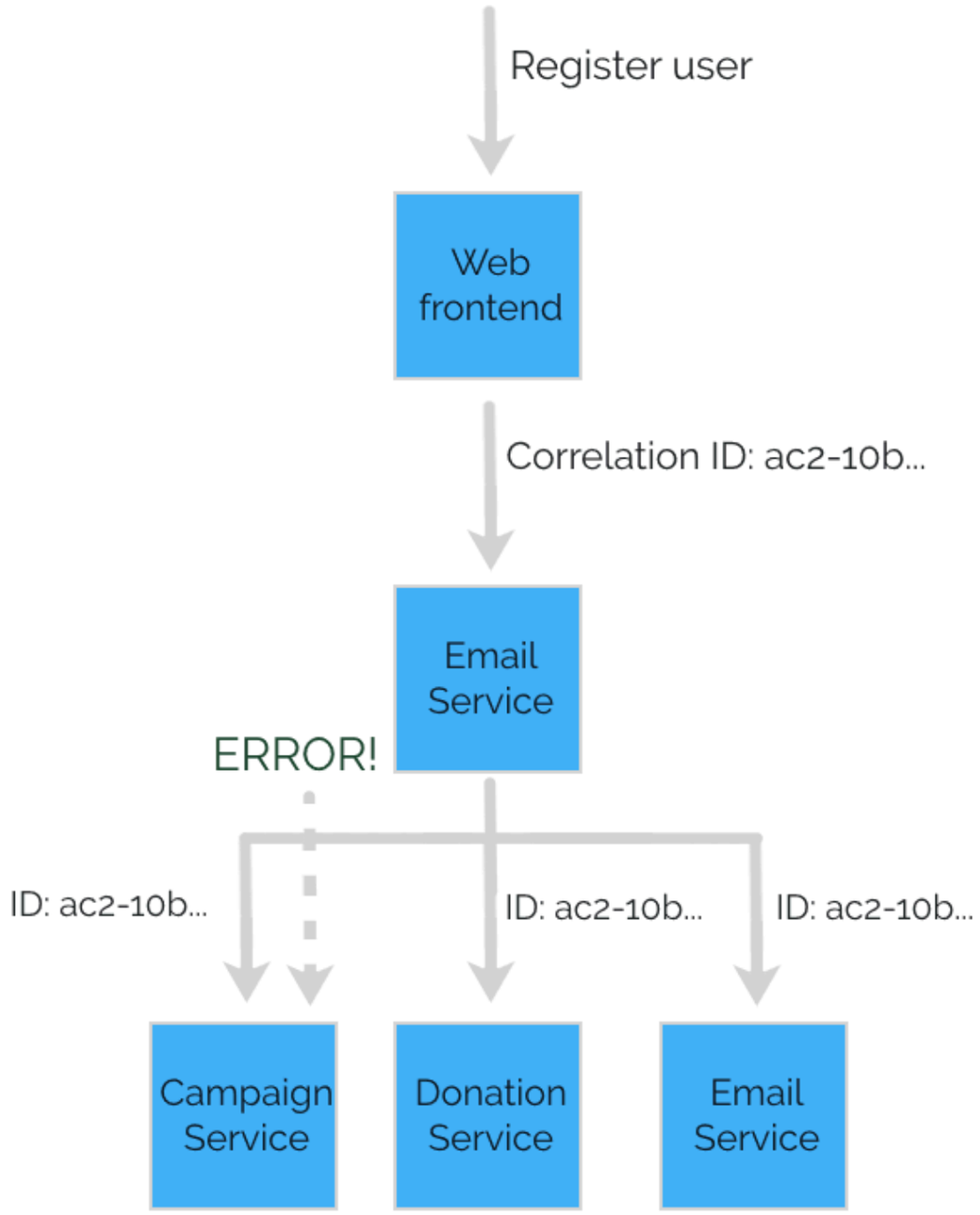
Autonomous Service ?

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

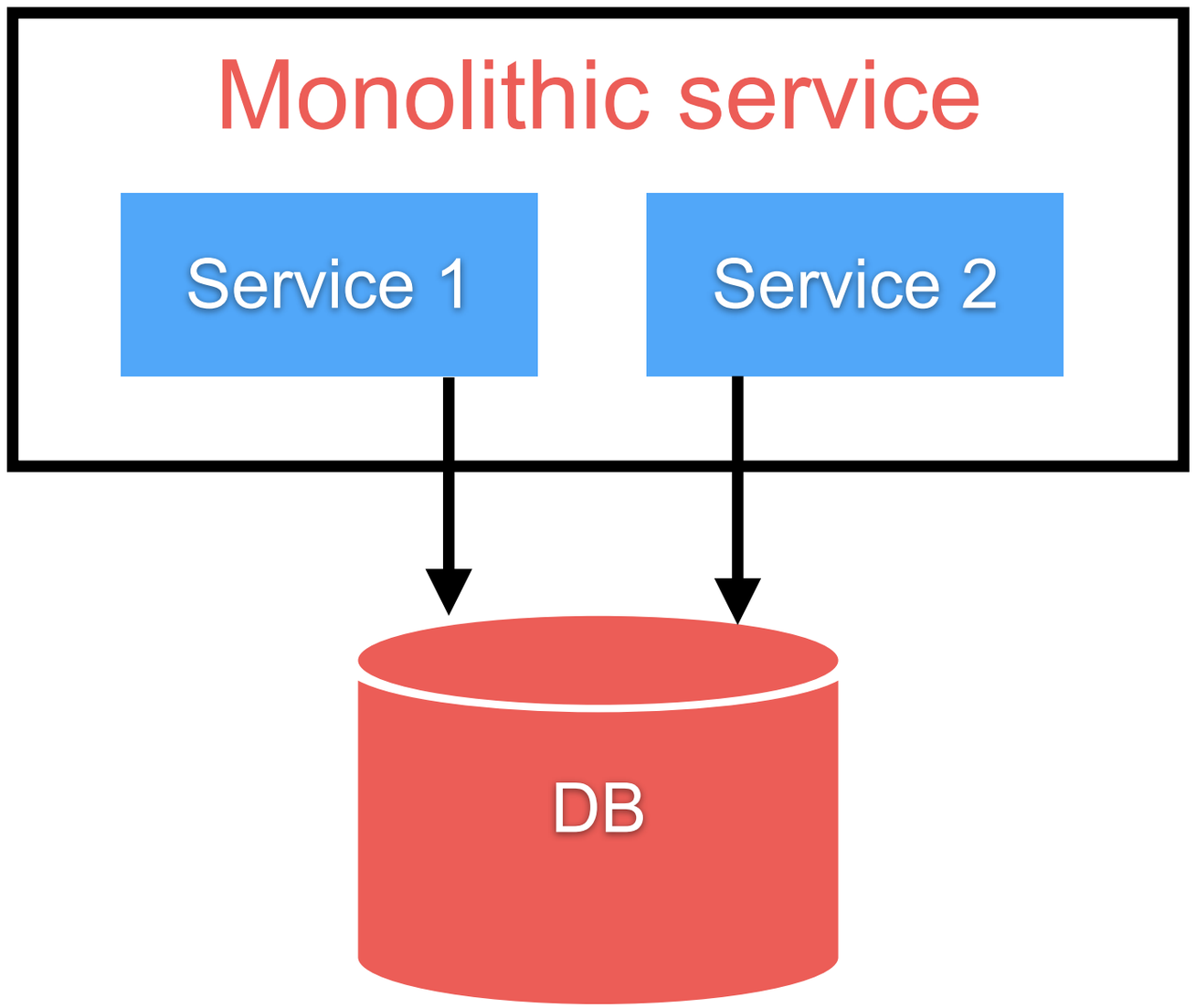
Maximize the ability of one service to evolve independently of others

Services should **hide their database**

Avoid **tight coupling**

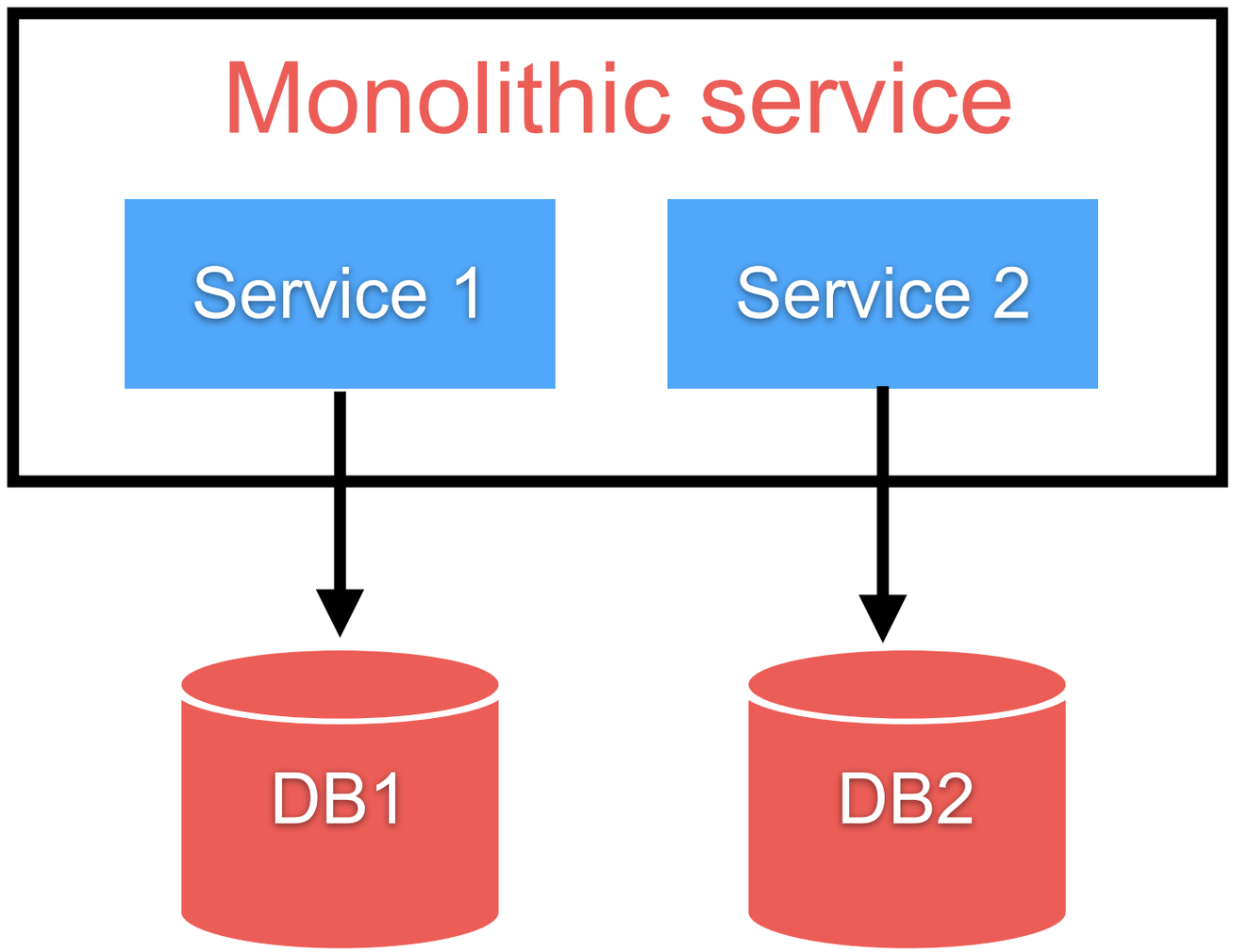
Hide internal implementation

Example with Database



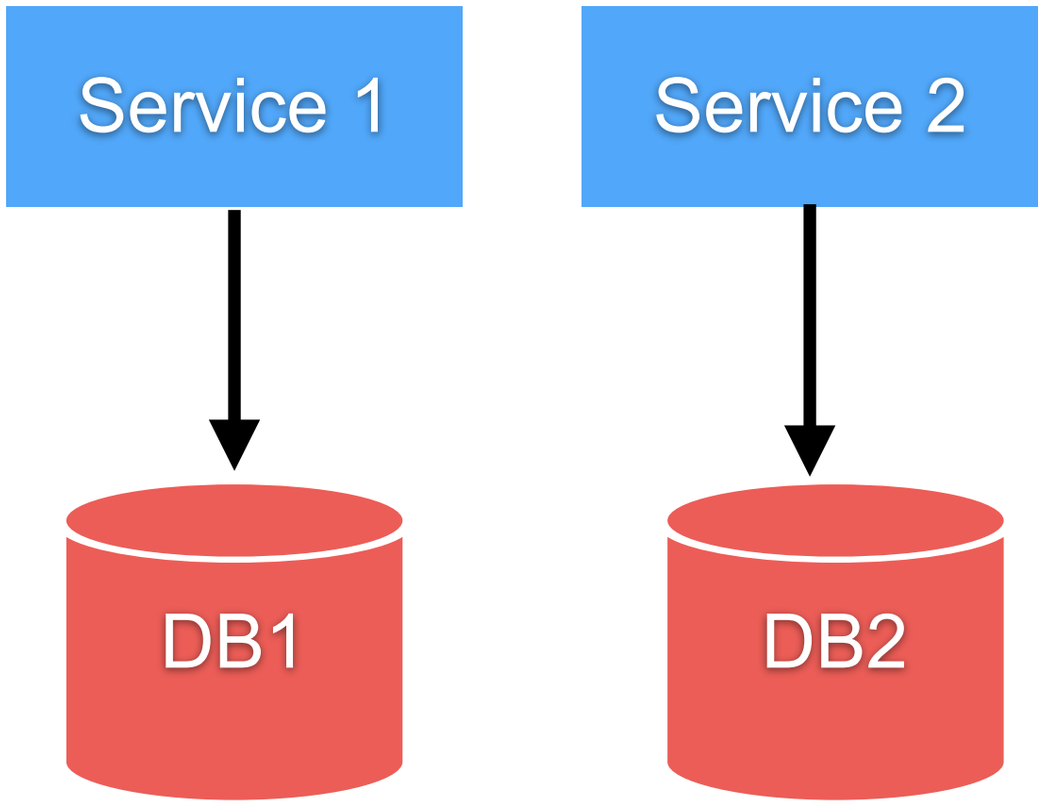
Hide internal implementation

Split database schema



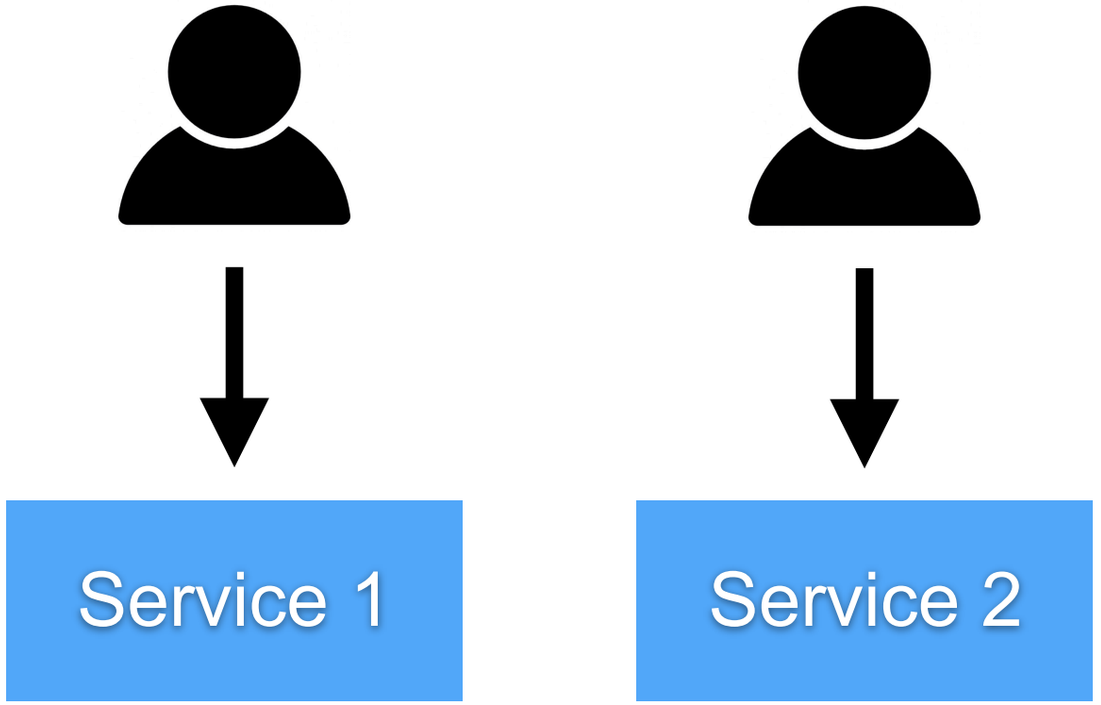
Hide internal implementation

Split application into micro services



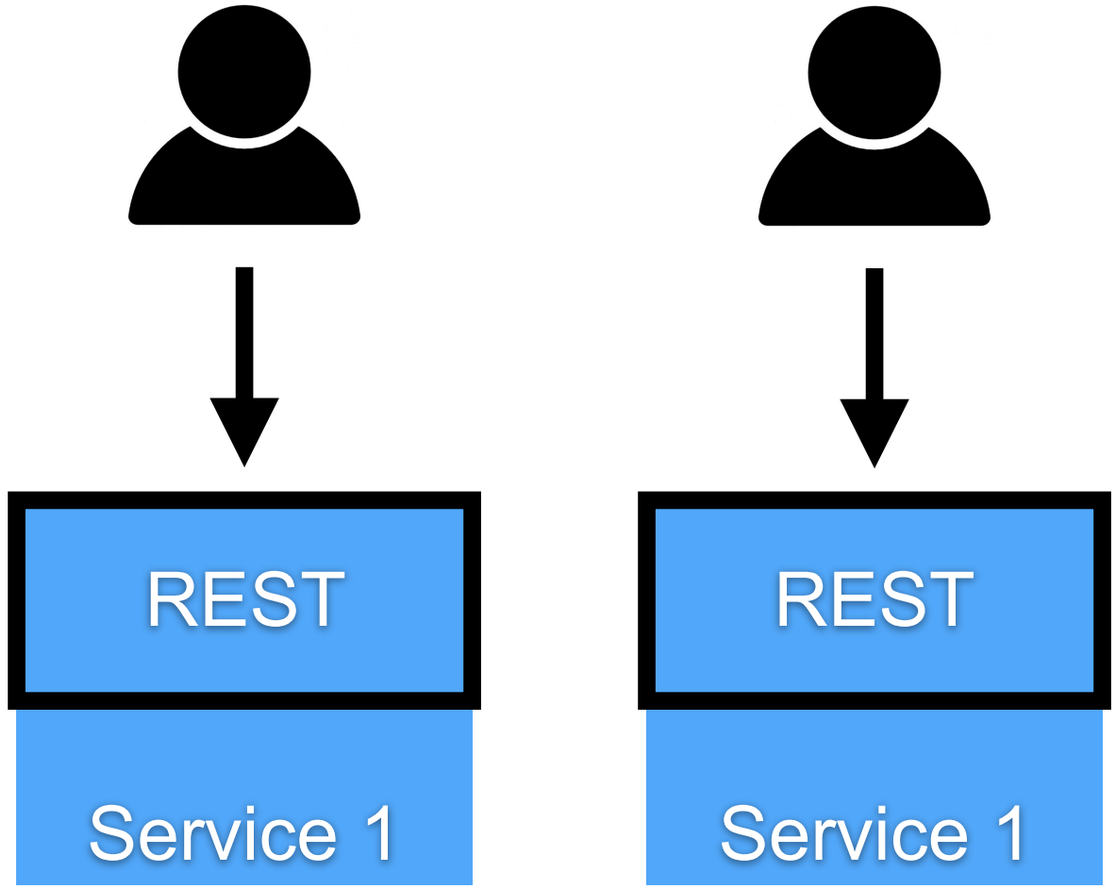
Hide internal implementation

Using specific technology to access services



Hide internal implementation

Using specific technology to access services



Hide internal implementation

Using specific technology to access services

Binary protocol

Service 1

Binary protocol

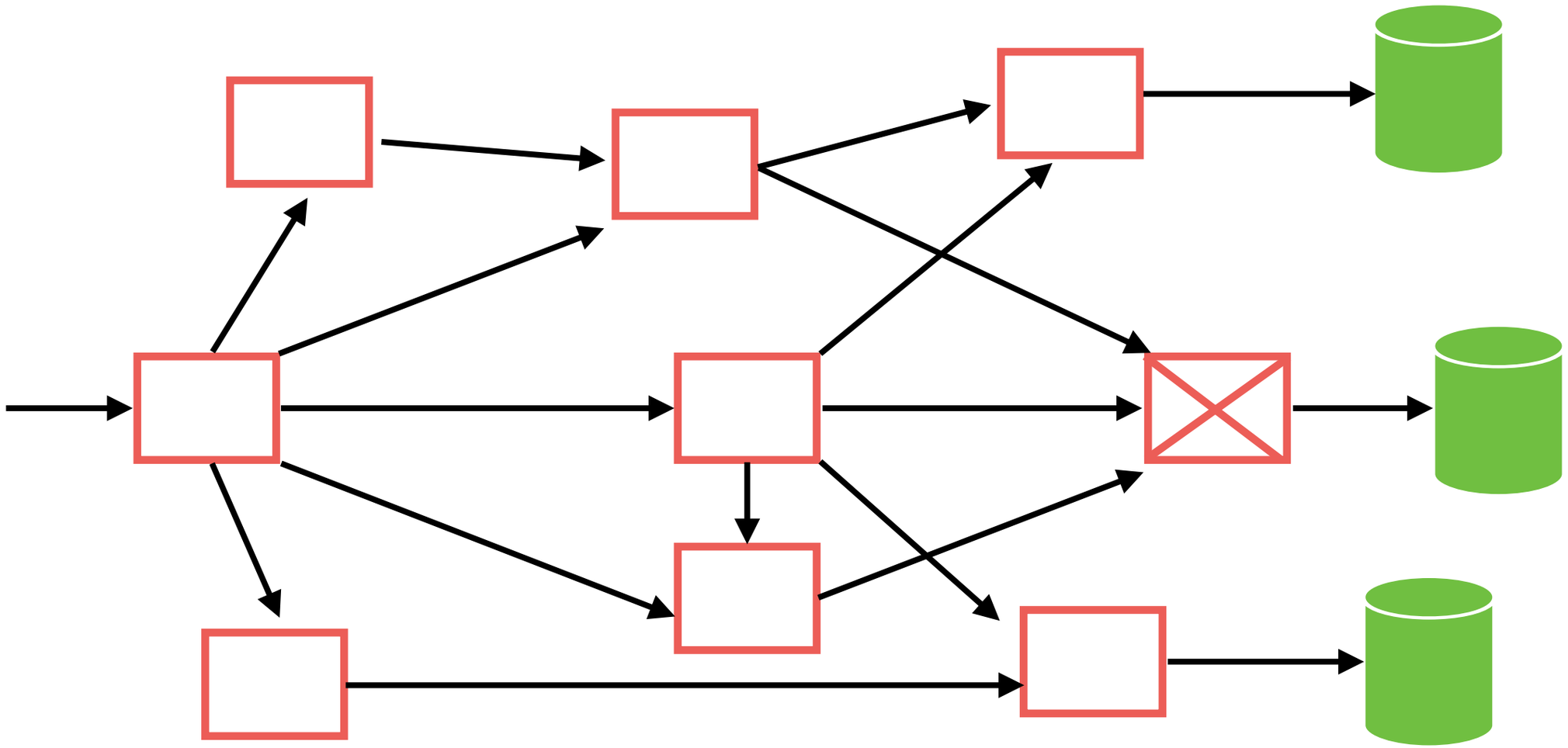
Service 1

Apache Thrift ™

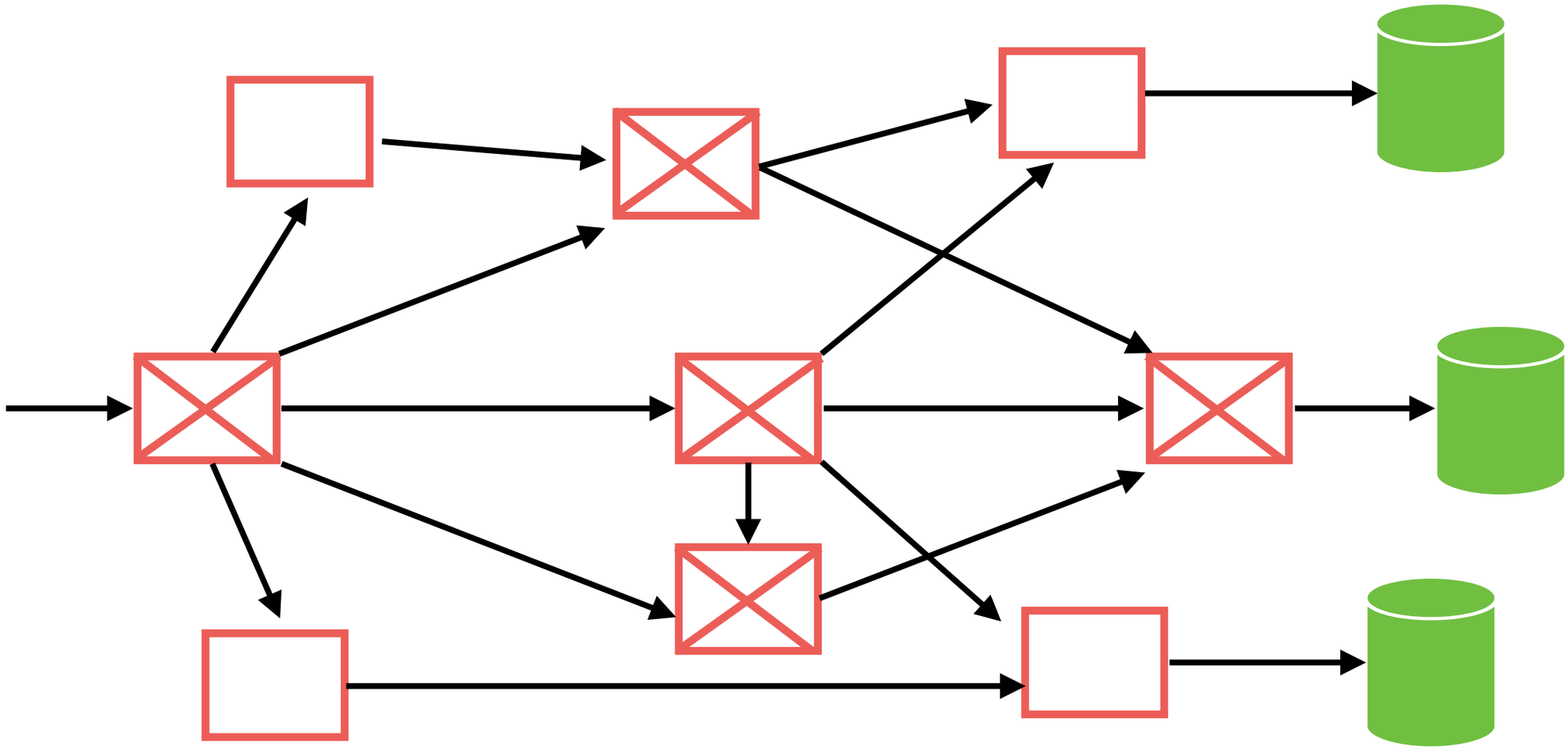
Isolated failure

Need more resilient than monolithic system
Plan for failures in all parts of system
Reduce cascading failure of service

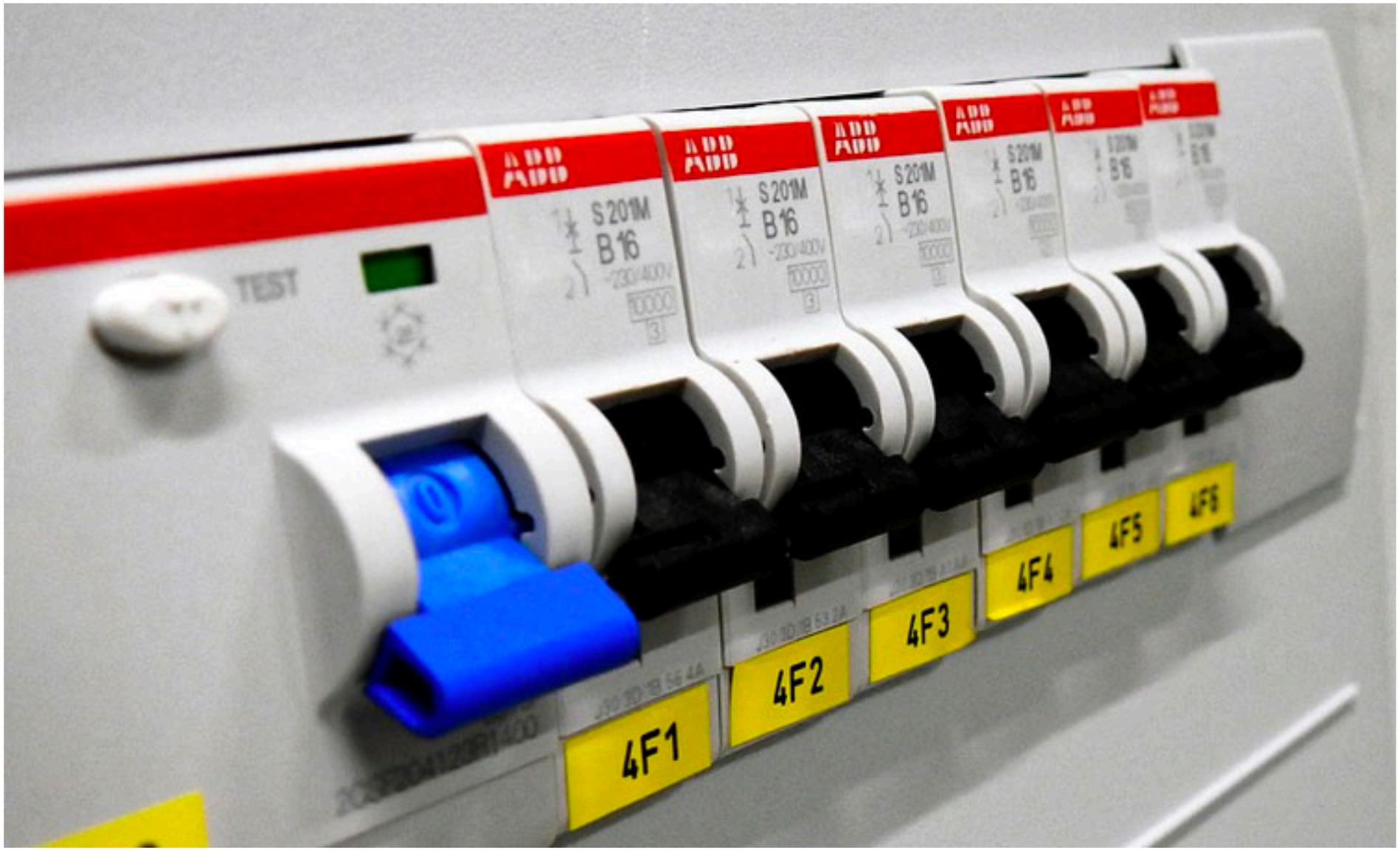
Failure !!



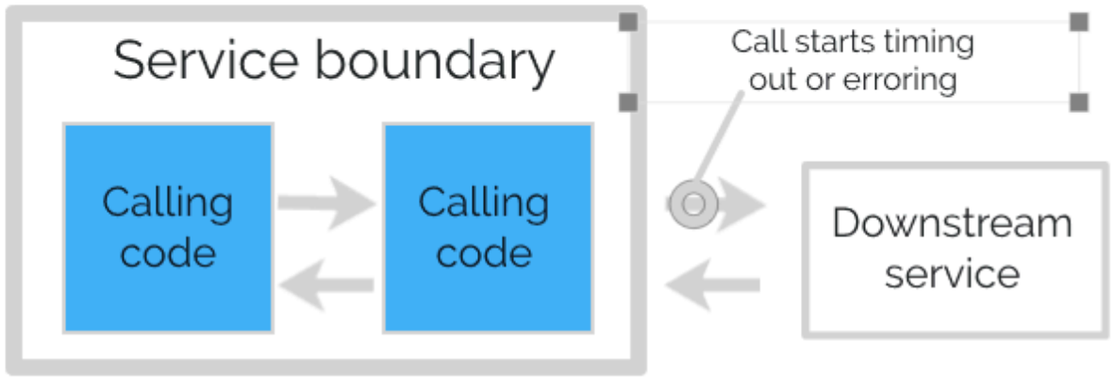
Cascading Failure !!



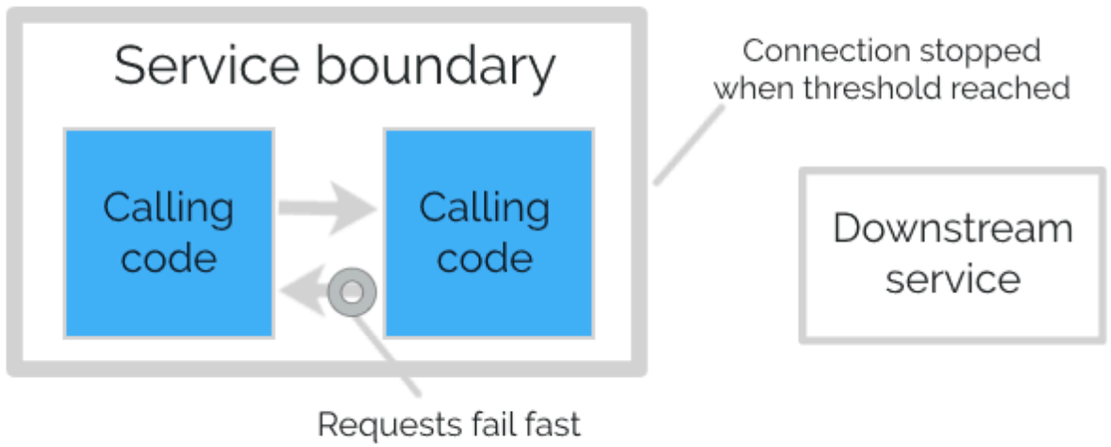
Circuit Breaker



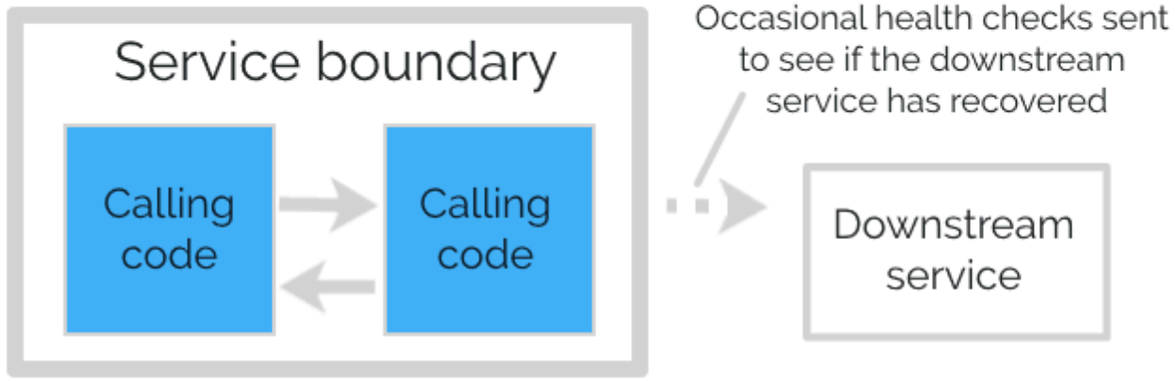
Circuit Breaker



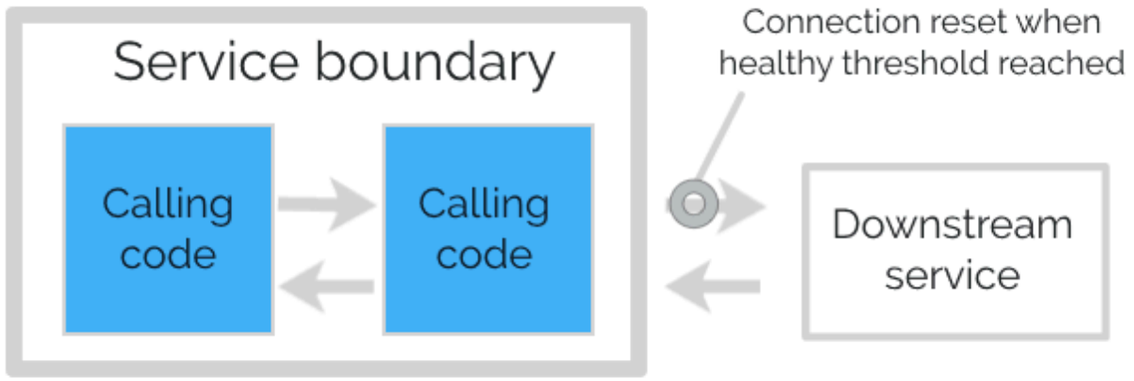
1. Call starts falling



2. Circuit breaker blown



3. Health checks sent

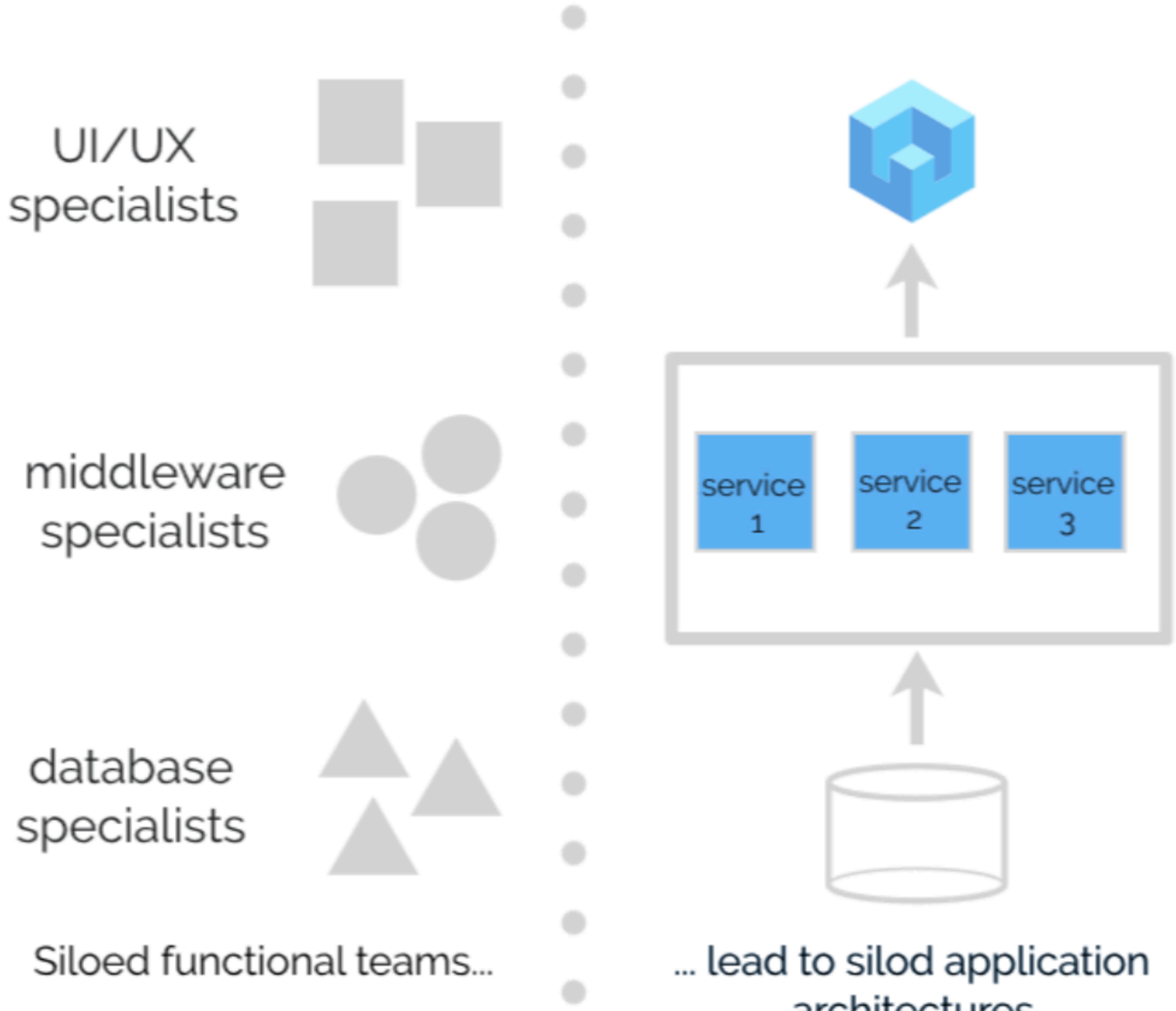


4. Circuit breaker reset

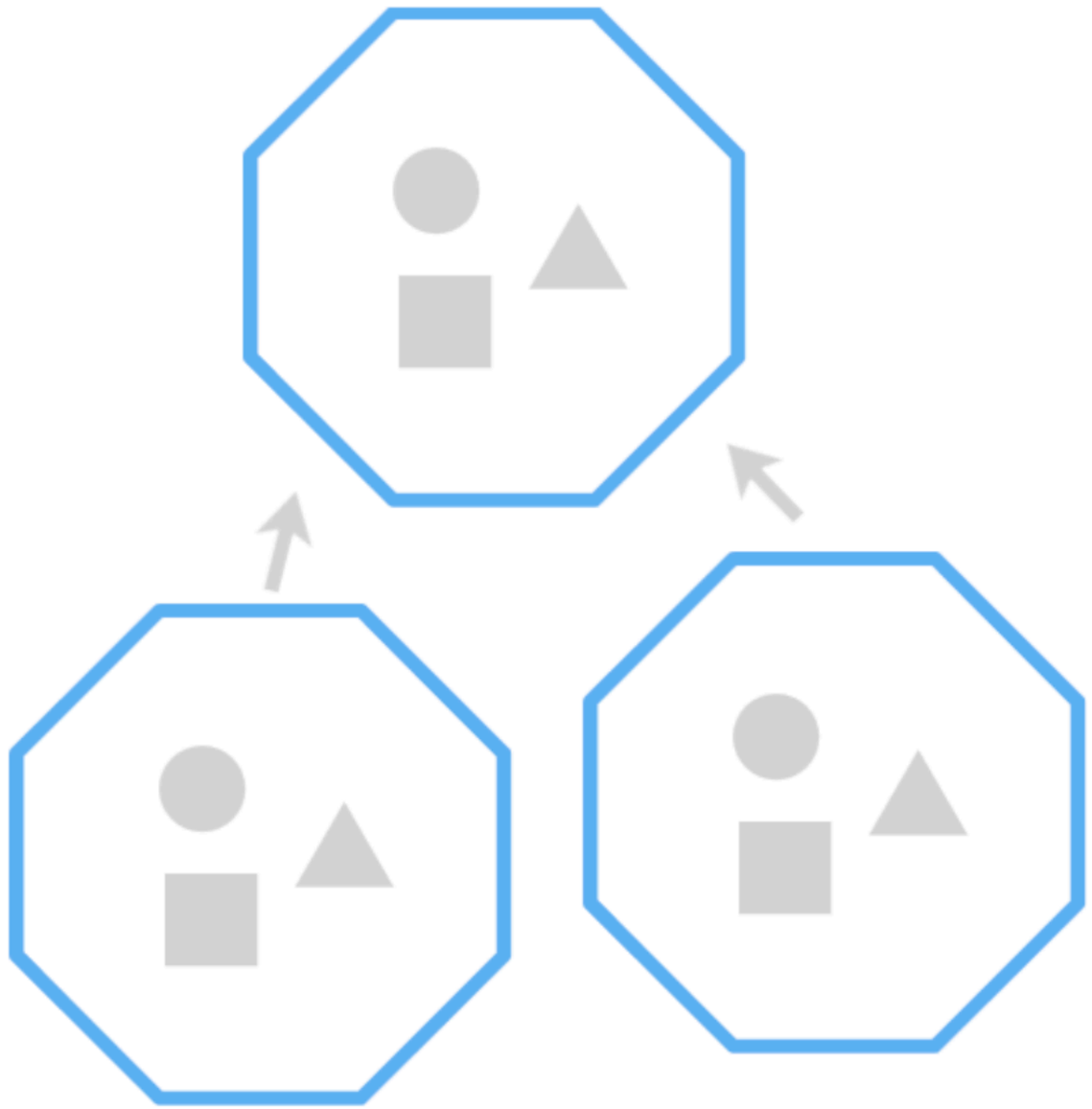
Decentralize all the things

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

Traditional structure



Cross-functional team



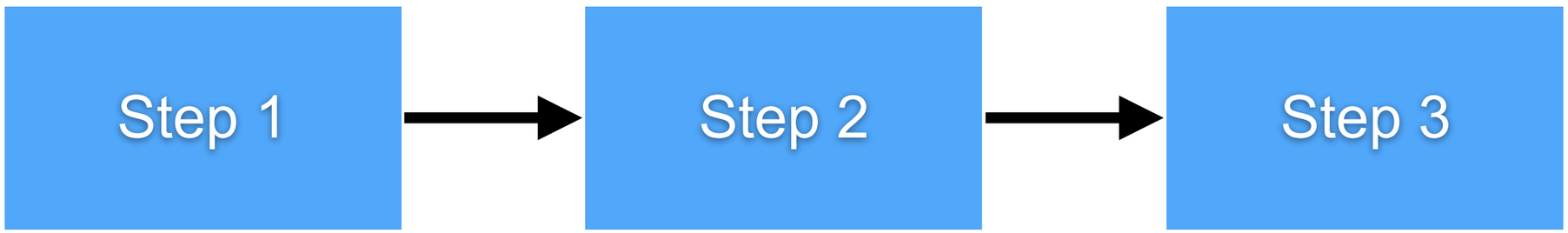
Workshop

Design your services

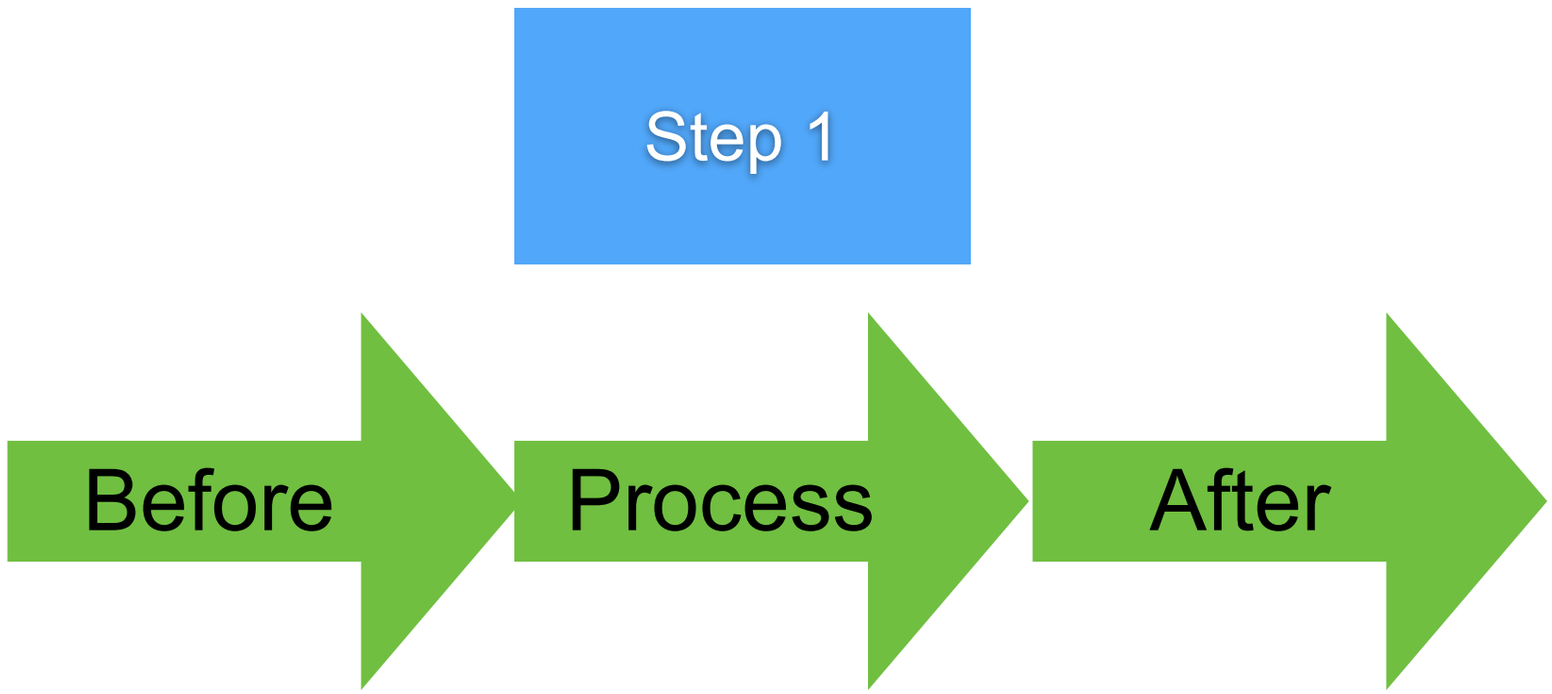
1. Understand requirements

Listen
Ask question
With scenario and example

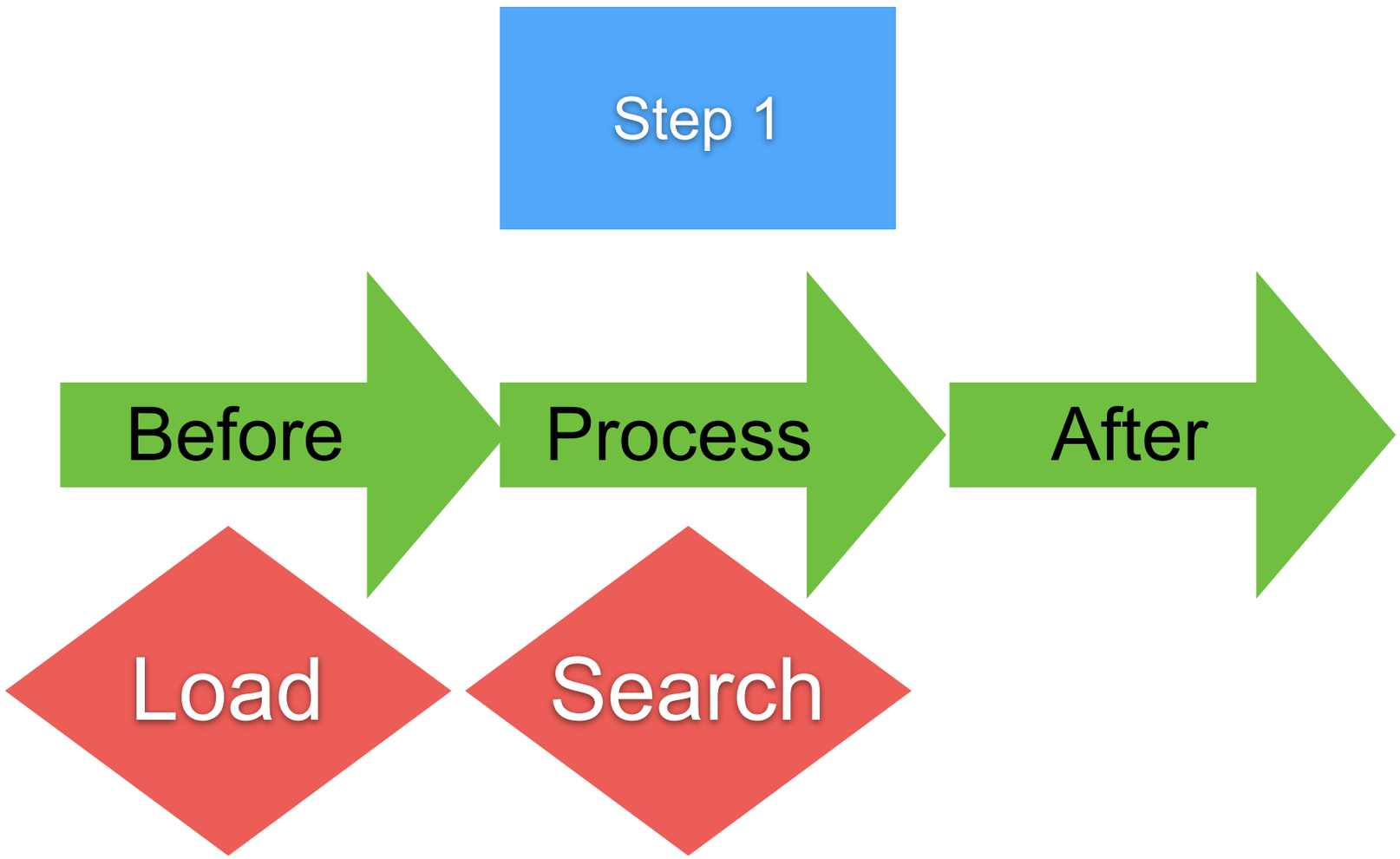
2. User flow/process



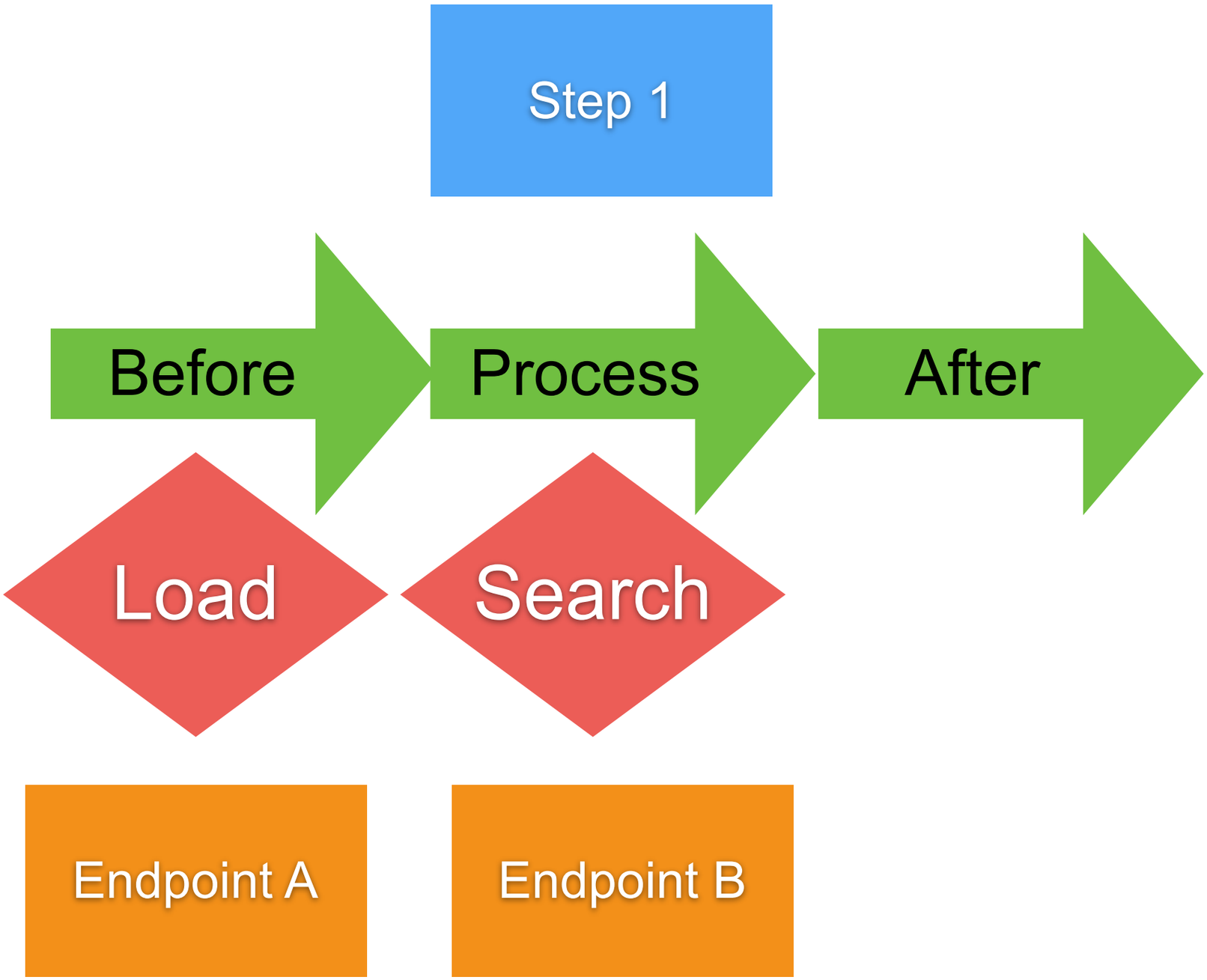
3. Define events/action



3. Define events/action



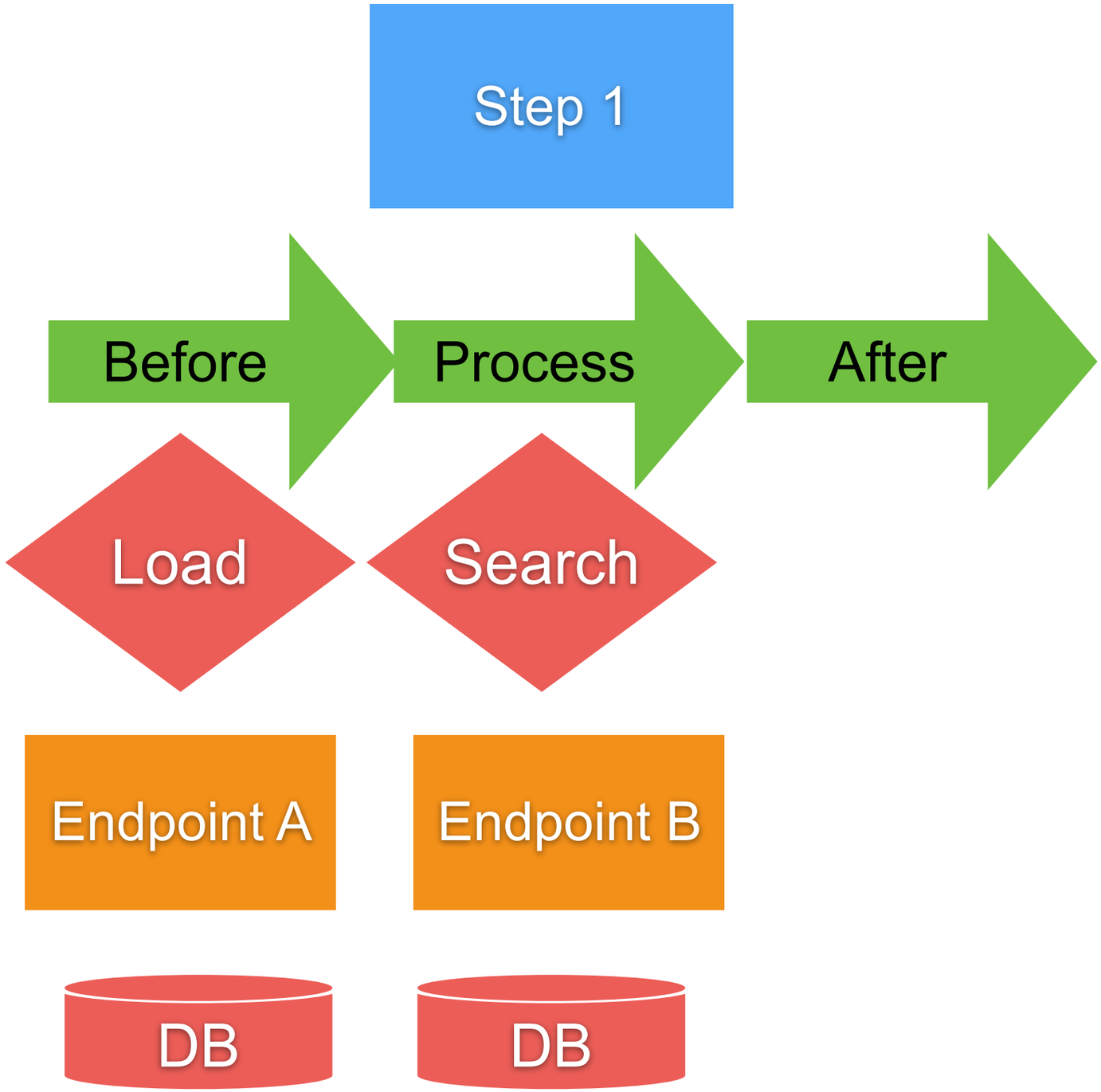
4. Define endpoint in each events



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