Introduction to Microservices

Architectural style

This is not new!

The old new thing...

SOA

service oriented architecture

- Modernized version of SOA

New world:

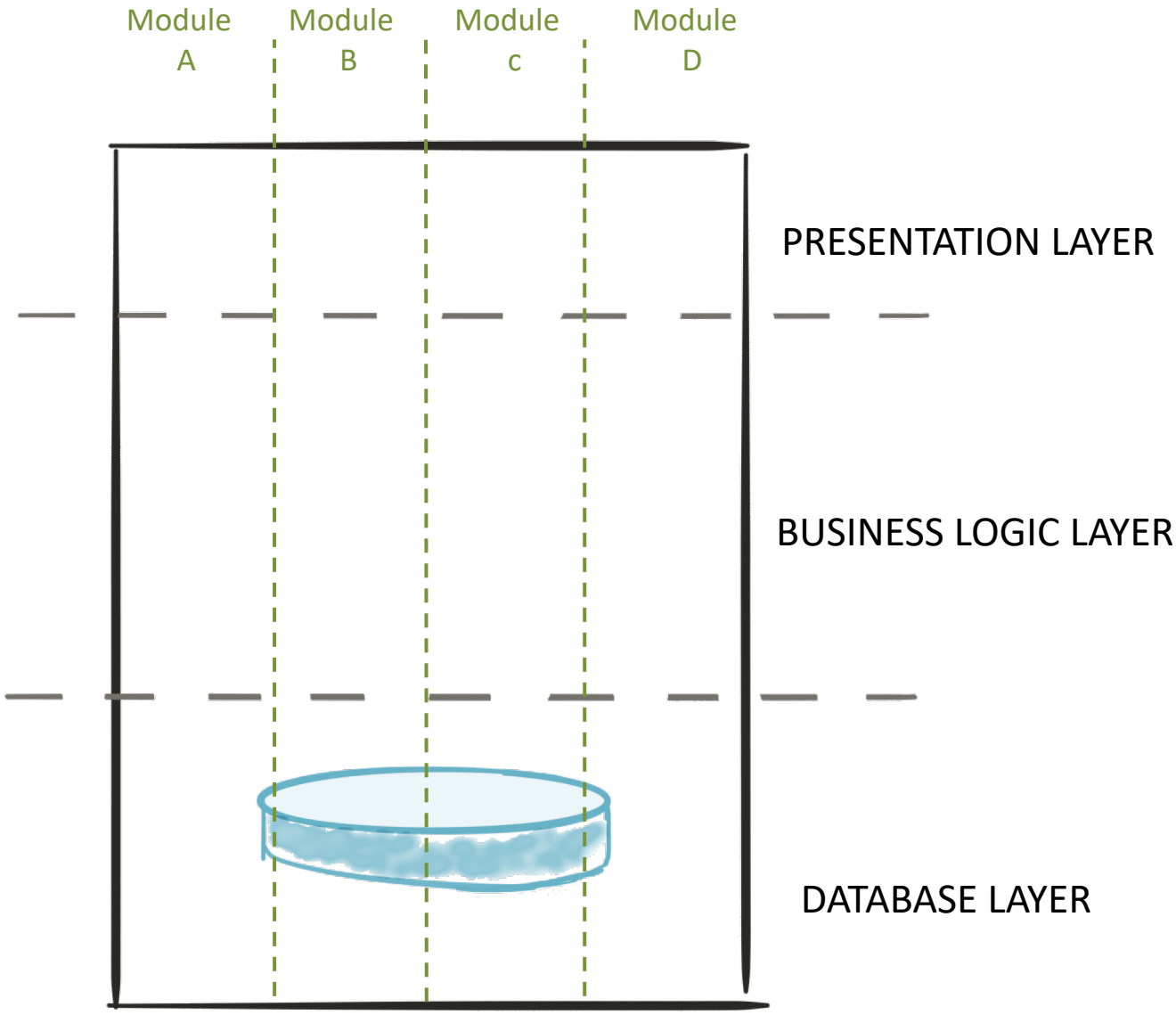
- Speed of delivery
- Scalability
- Innovation / experimentation
- Cloud / devops

VS

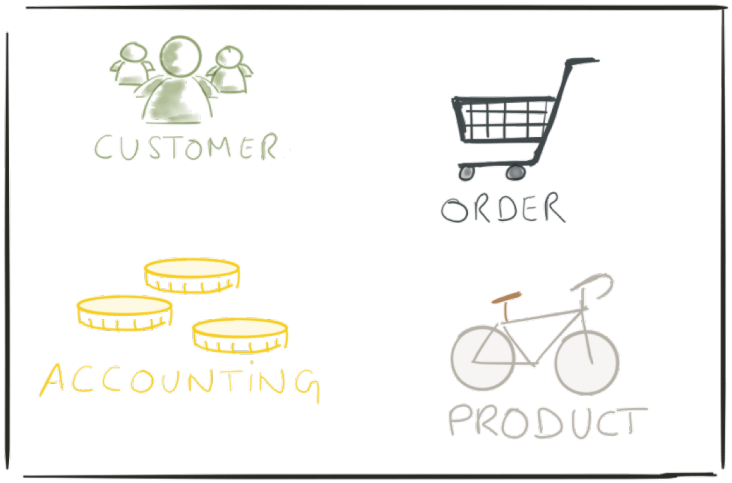
monolith

microservices

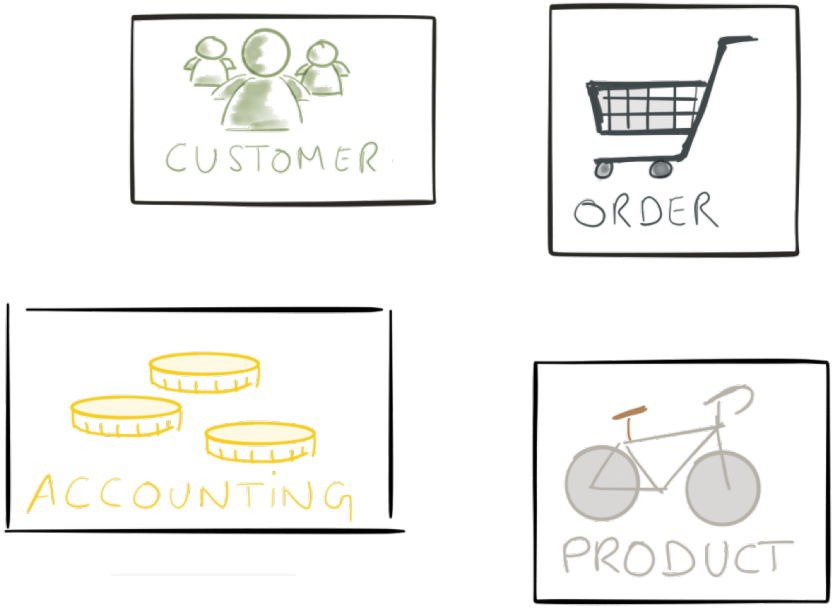
A monolith


Module A
Module B
Module c
Module D
PRESENTATION LAYER
BUSINESS LOGIC LAYER
DATABASE LAYER

VS

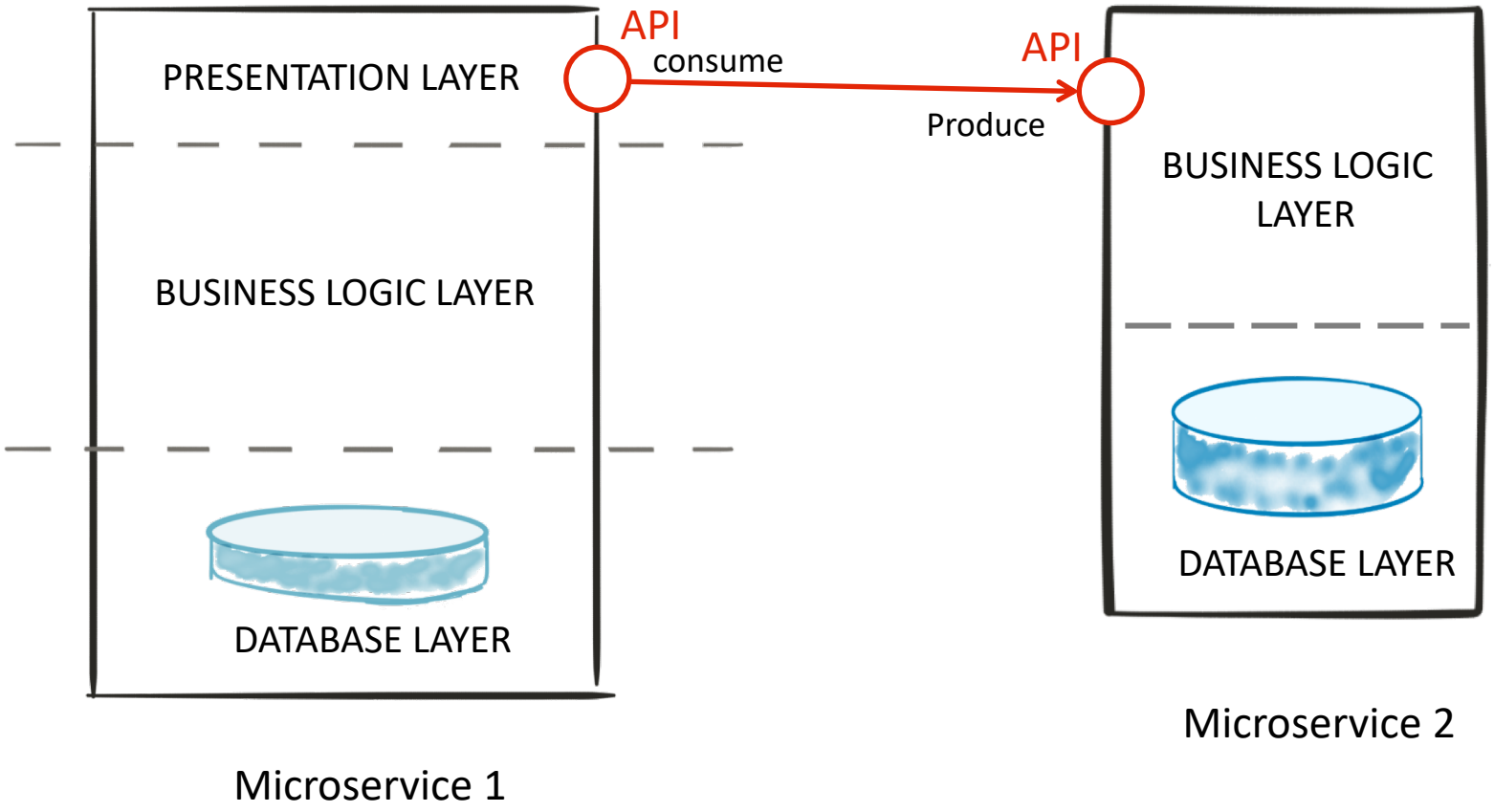


monolith



microservices

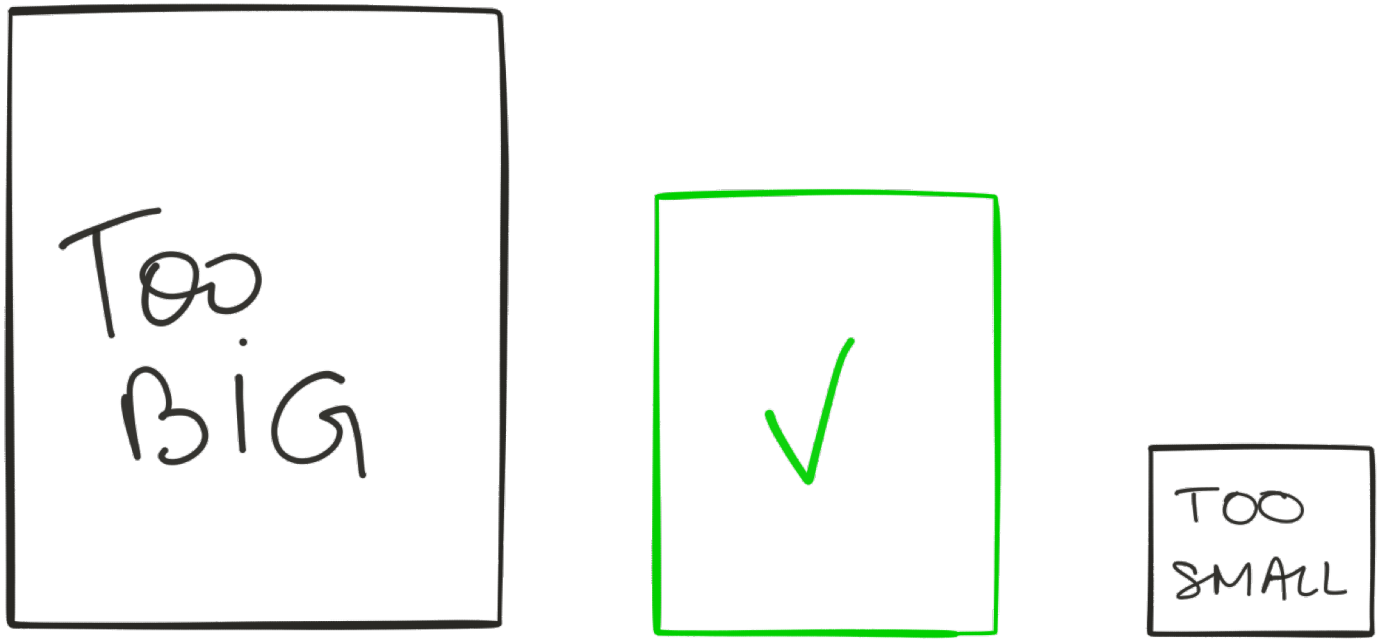
microservices



principles

- ☐ Modularity
- ☐ Autonomous
- ☐ hide implementation details
- ☐ automation
- ☐ Stateless
- ☐ highly observable

modularity



modularity

- ✓ ModeLled around business capability
  - Single responsibility
  - Single data domain
- ✓ Separation of concerns
- ✓ Low coupling
- ✓ Understandable by a person

Modularity (TEAM)

A product not a project

UI - team

SERVER - team

Dba - team

monolith

UI

dba

SERVER

UI

dba

SERVER

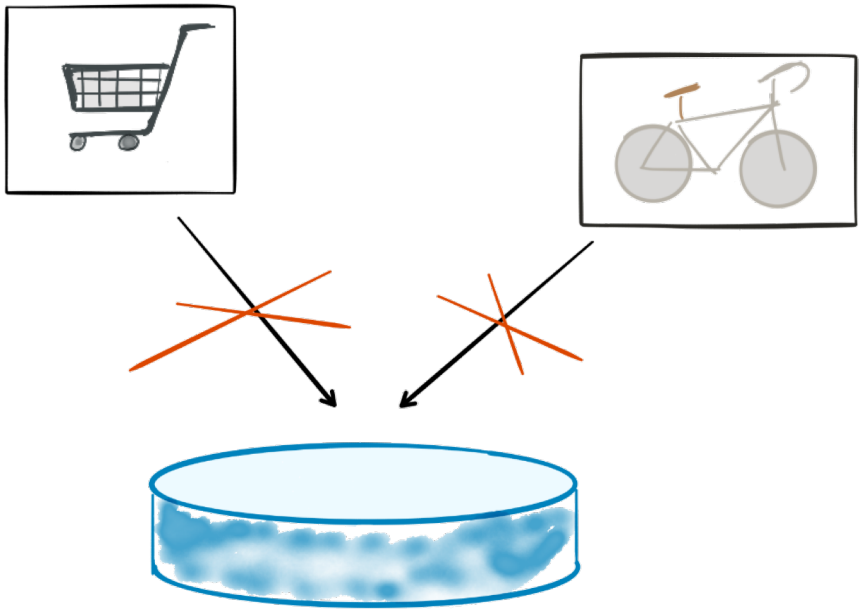
UI

dba

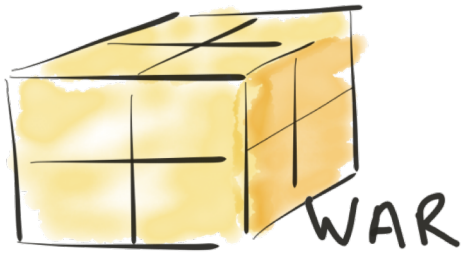
SERVER

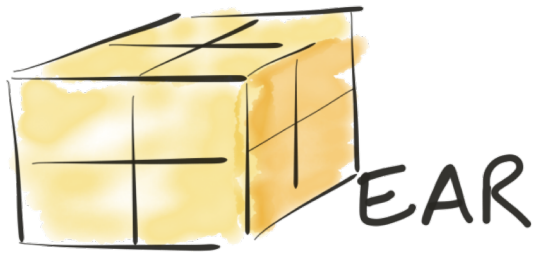
microservices

modularity



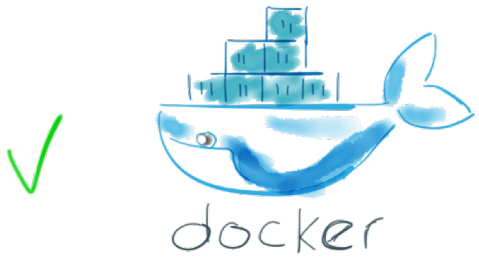
autonomous





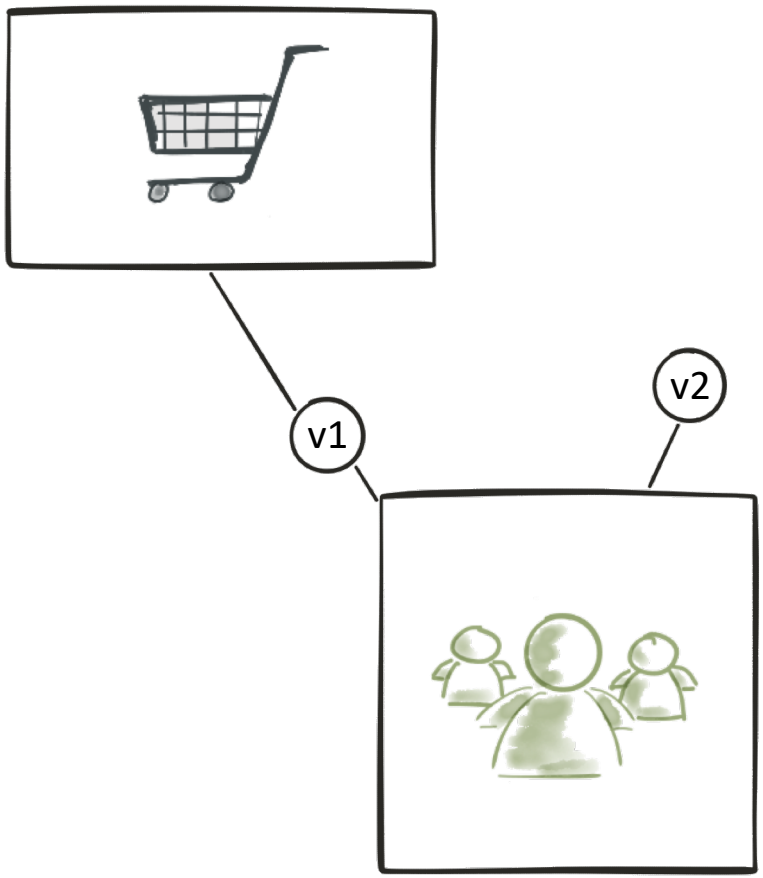
monolith

- Libraries
- http listener

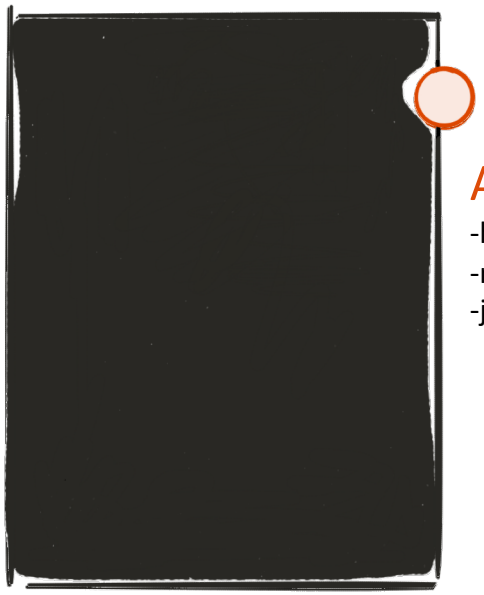


microservices

autonomous

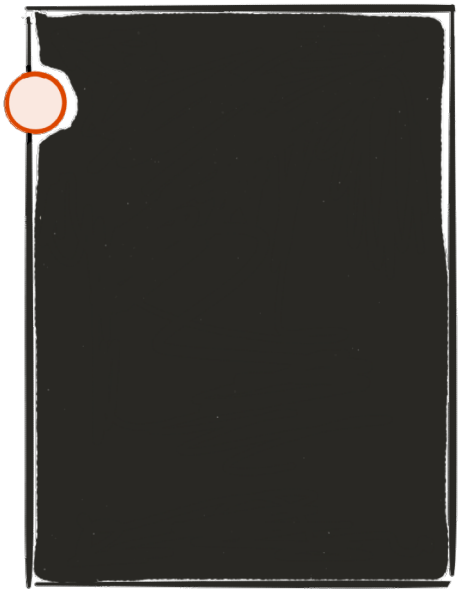


hide implementation details

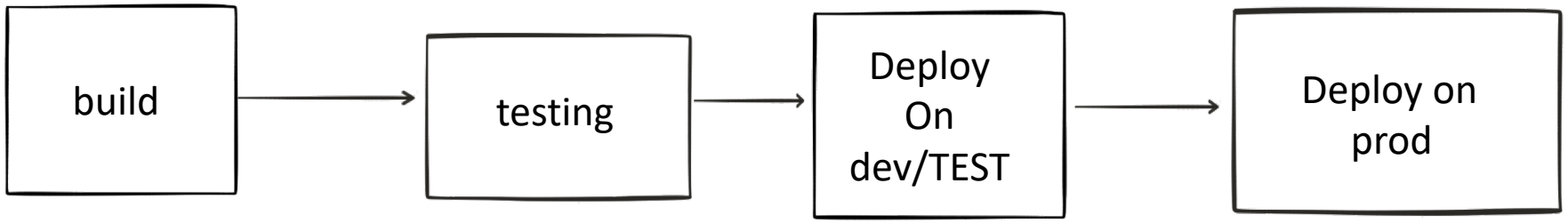


API
-http
-rest
-json

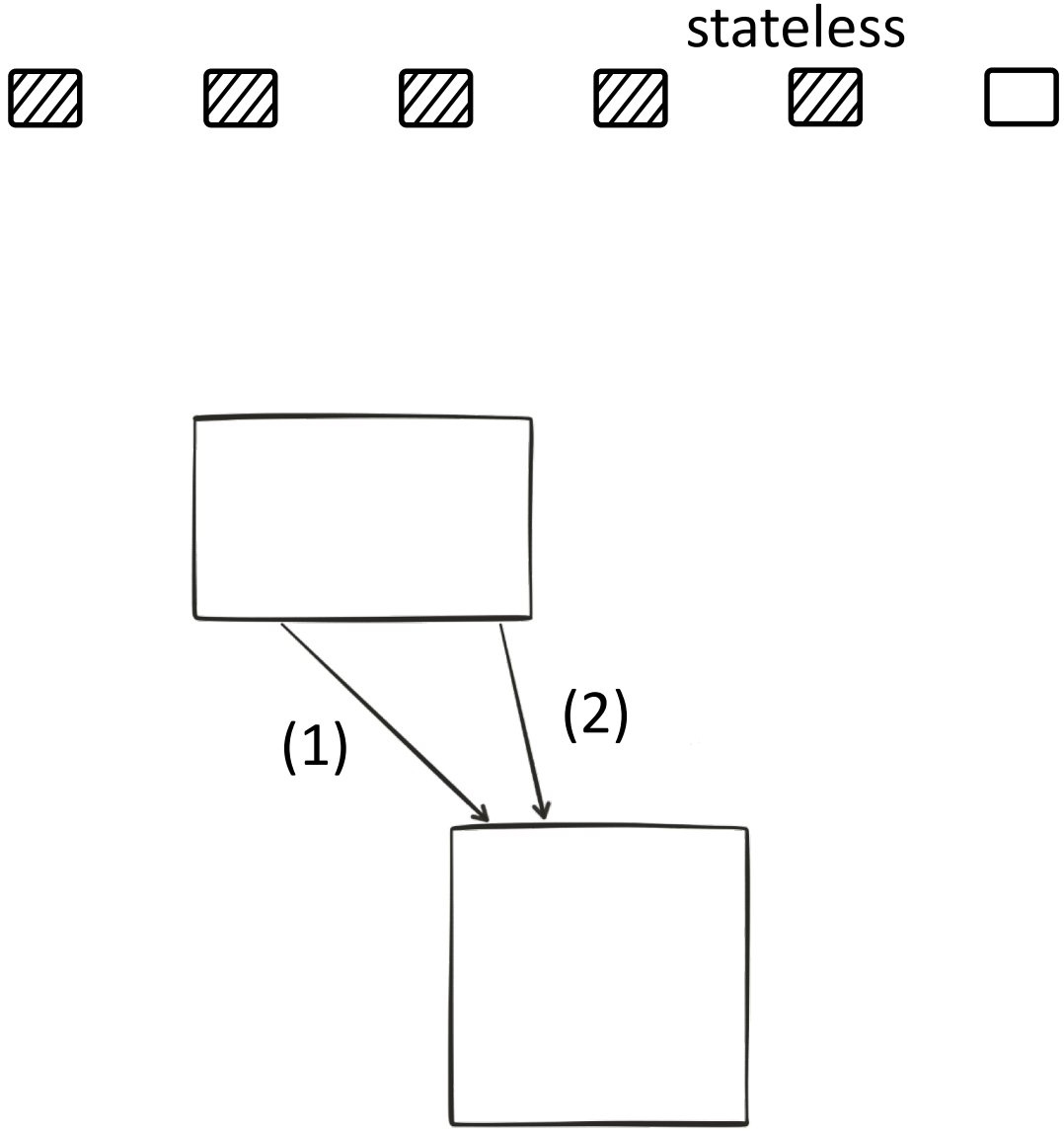
API
http-
Rest-
Json-



automation

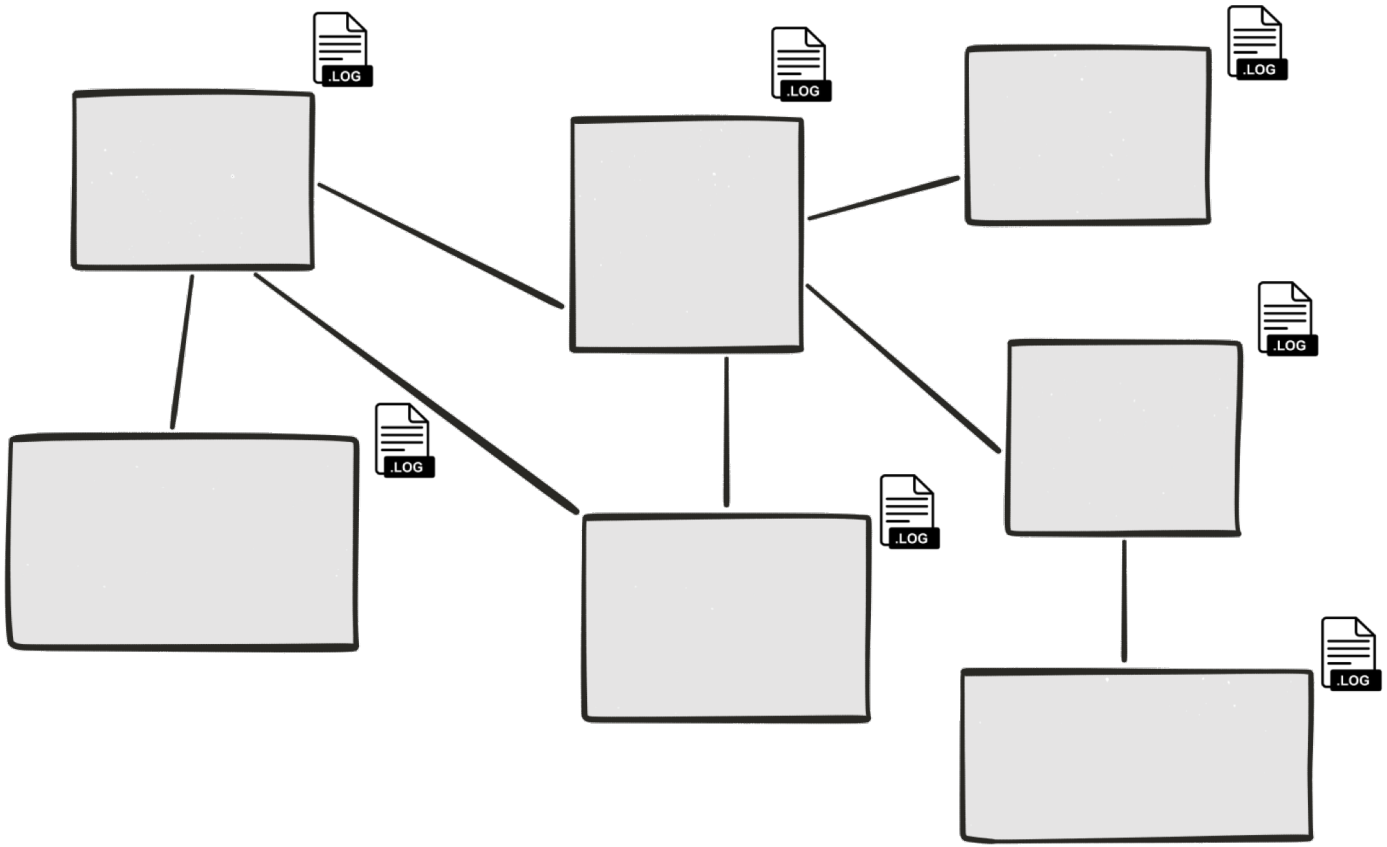


- Continuous integration
- Continuous deployment


stateless
(1)
(2)

Highly observable

Logs



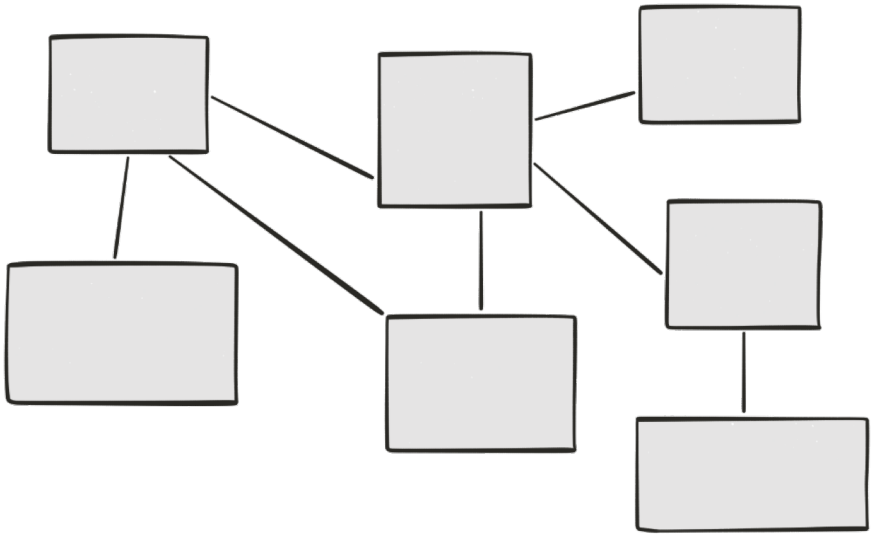
Highly observable

Centralized logging

elasticsearch.

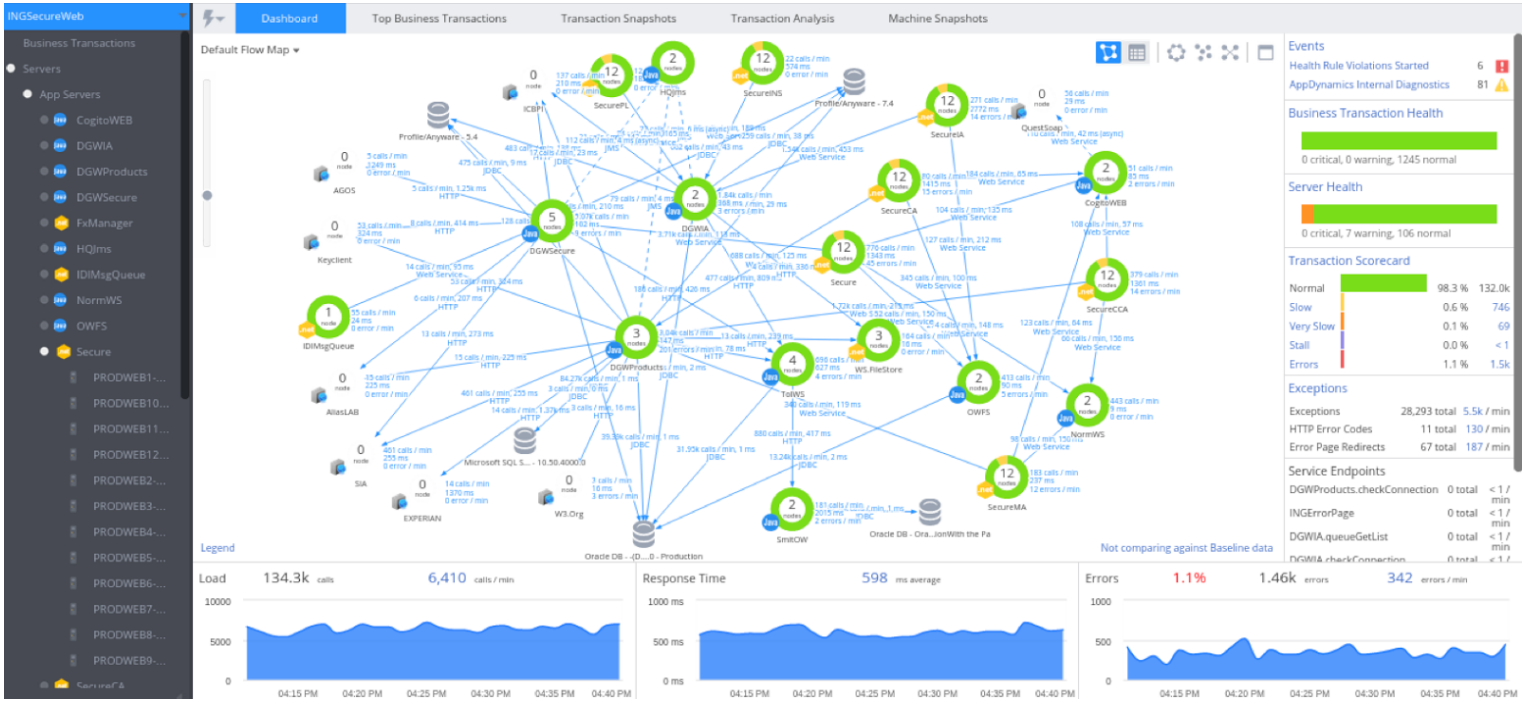
logstash

Kibana



Highly observable

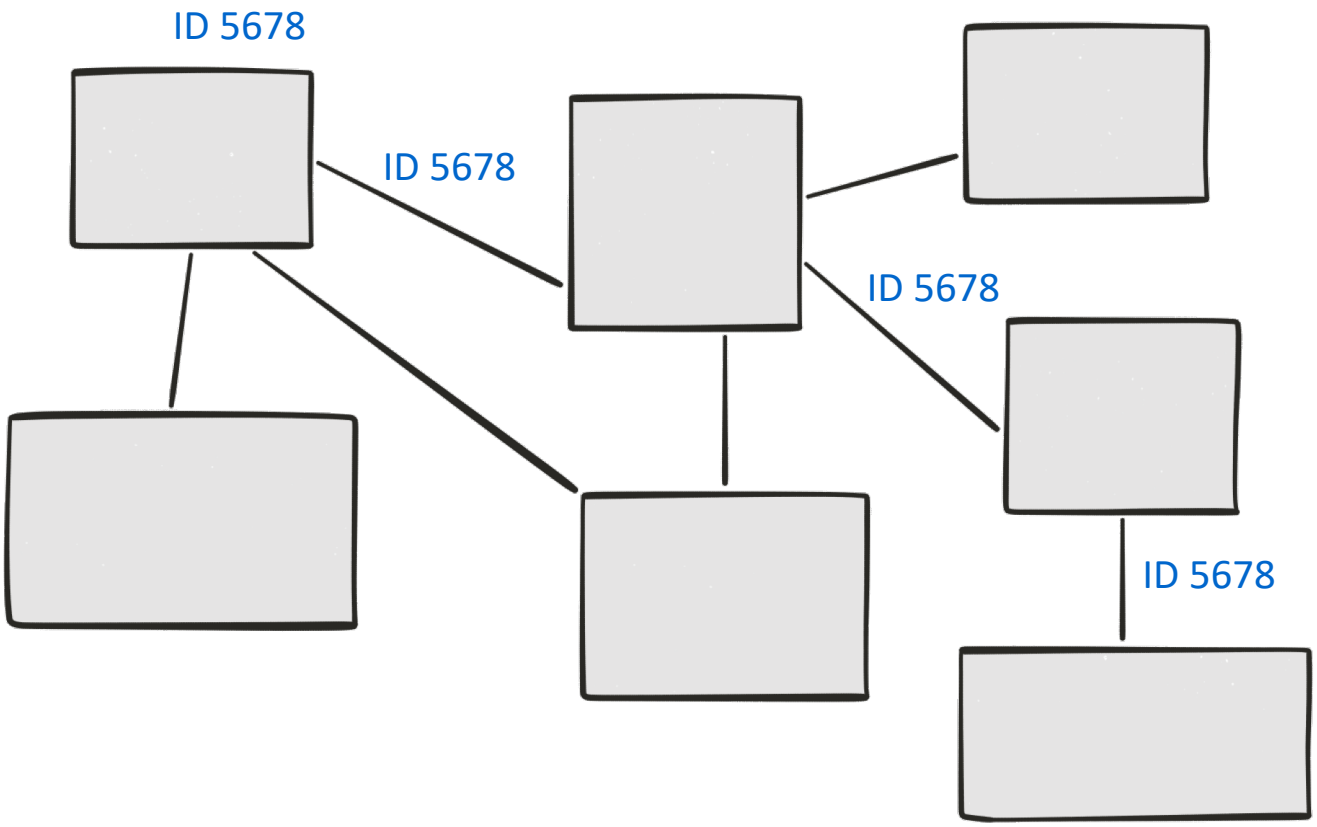
monitoring



AppDynamics

Highly observable

Correlation ids

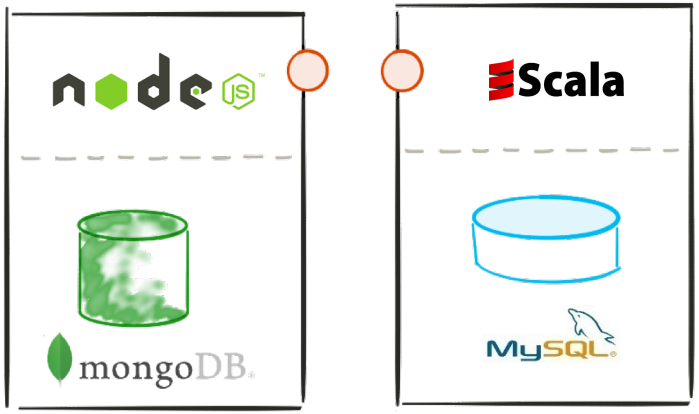


principles

- Modularity
- Autonomous
- hide implementation details
- Automation
- Stateless
- Highly observable

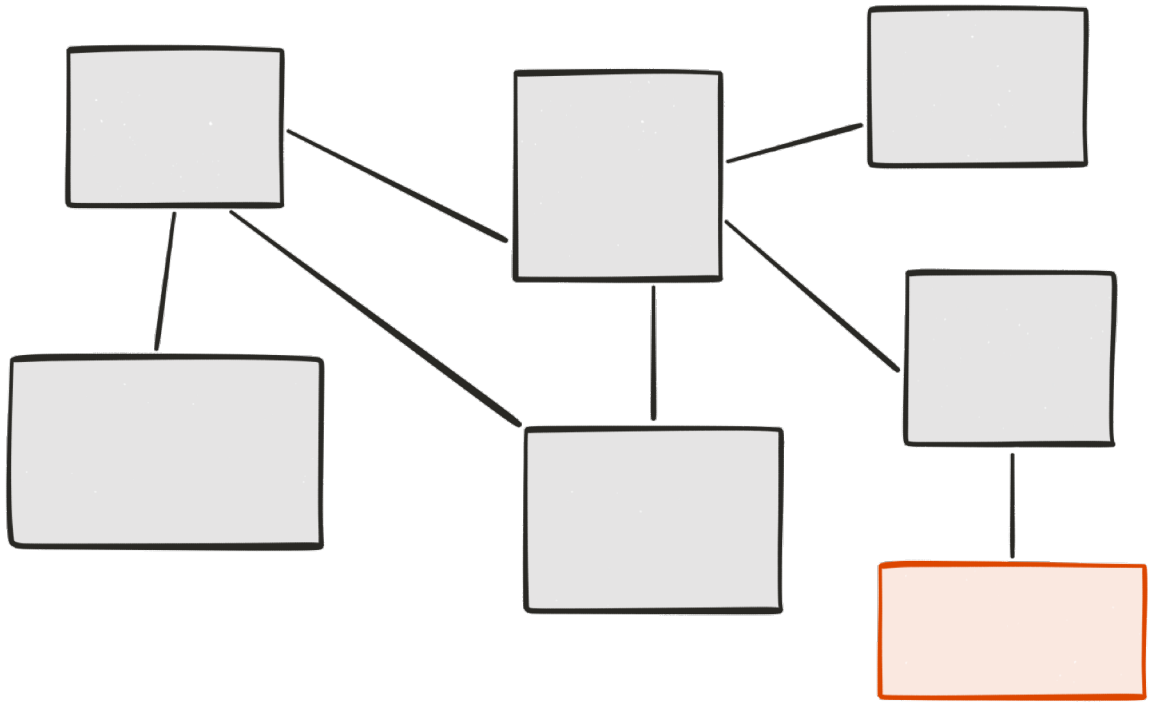
Advantages

Polyglot architecture



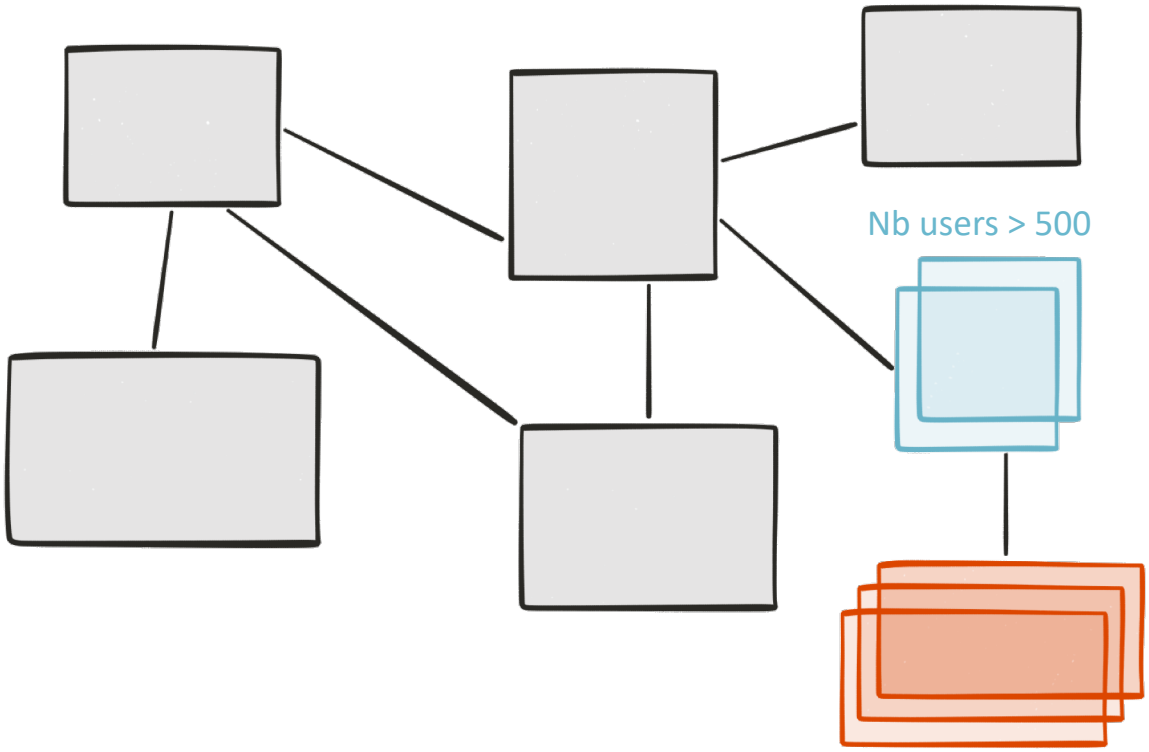
- The right technology for the job
- reduce technical debt

Evolutionary design



- Remove
- Add
- Replace
- Experimental microservice
- Grow at “no” cost

Selective scalability



Big vs small

- ✓ Smaller code base
- ✓ Simpler to develop / test / deploy / scale
- ✓ Start faster
- ✓ Easier for new developers

drawbacks

- Distributed system
  - Consistency
  - Transaction
  - Request travelling

- Slow (http)

- Requires an ecosystem

- Synchronous vs asynchronous

- Integration tests

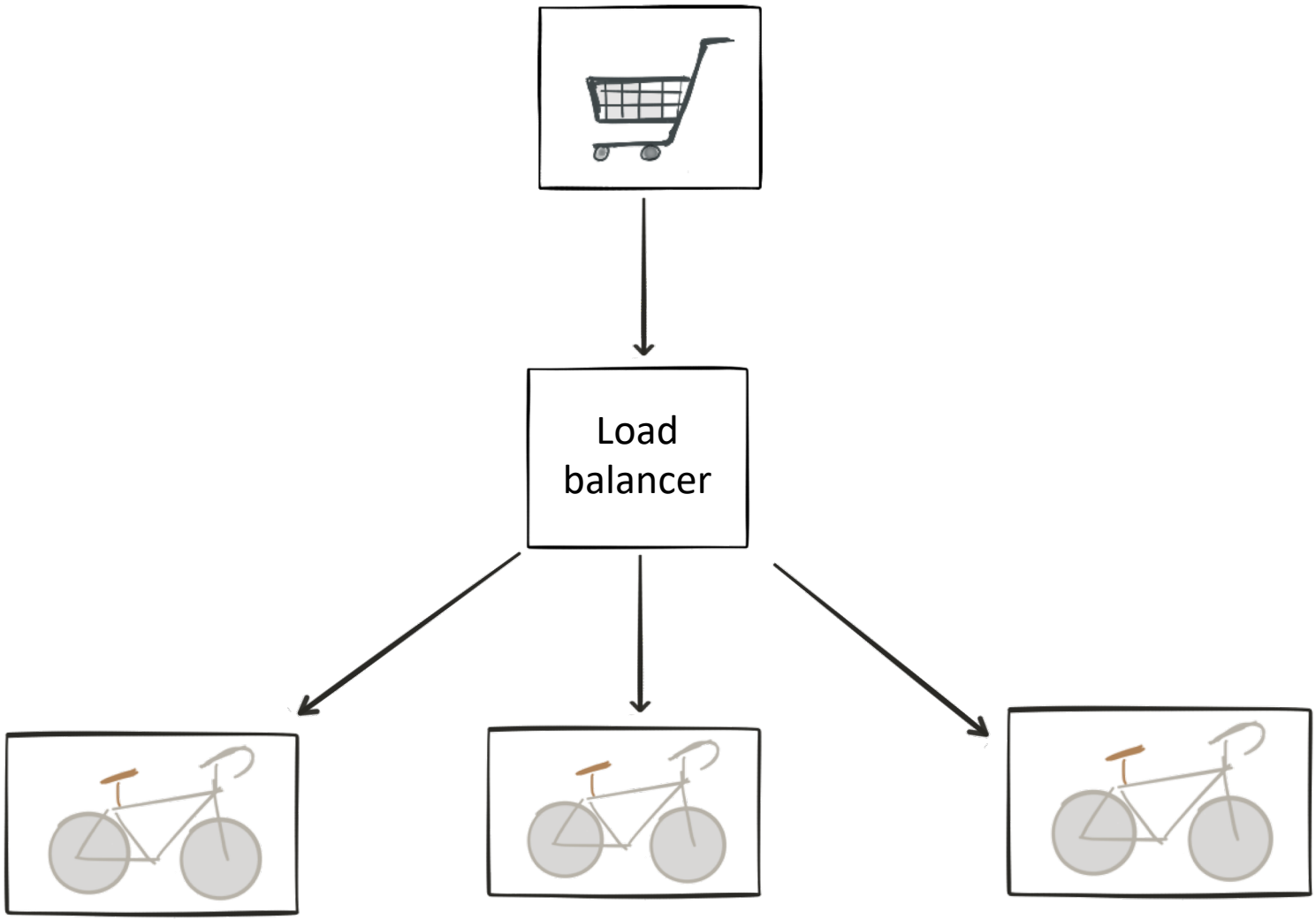
Conclusion:

- The Microservices architecture is more complex
Than a monolith.

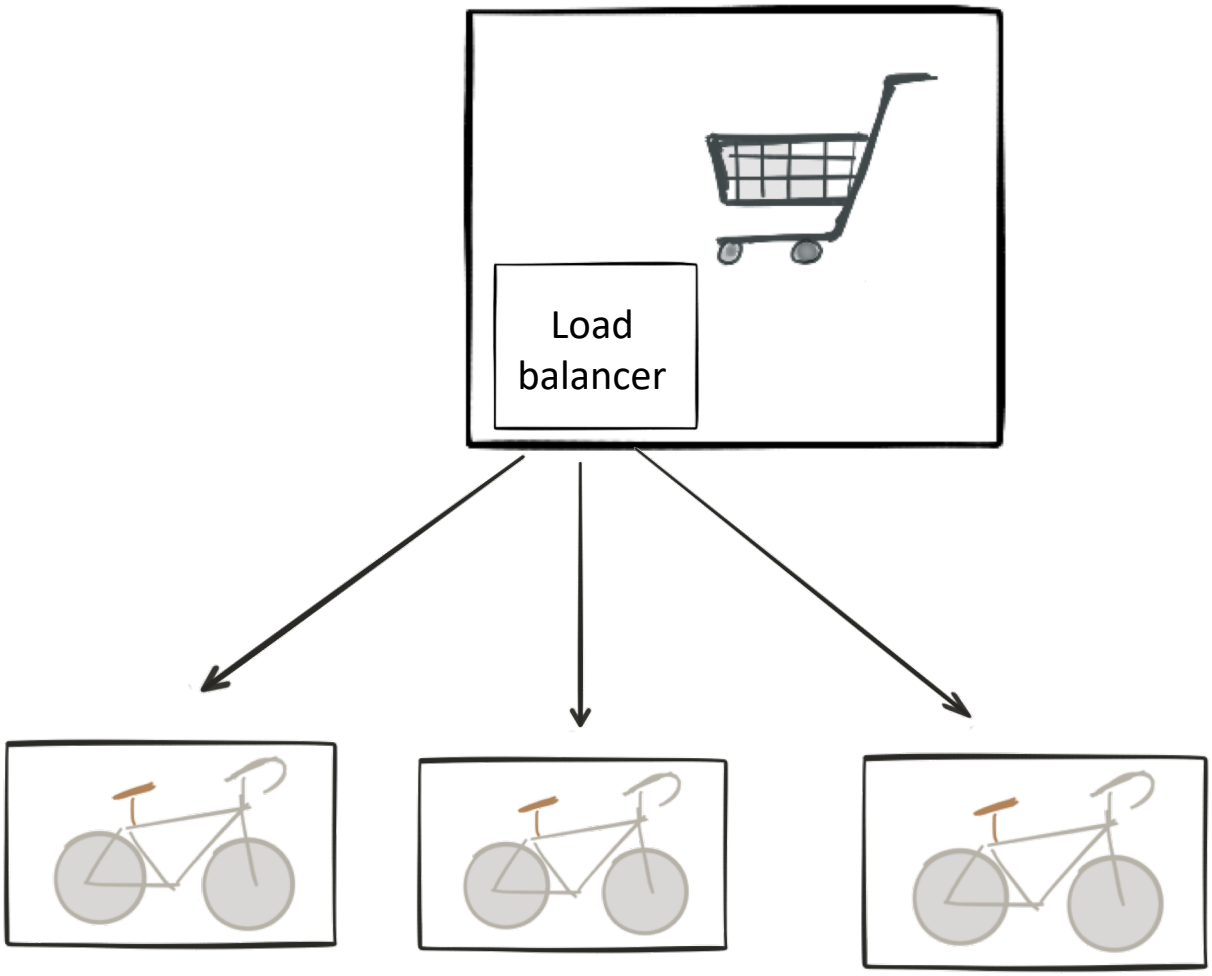
- This the cost of growing and scaling easily

Microservices ecosystem

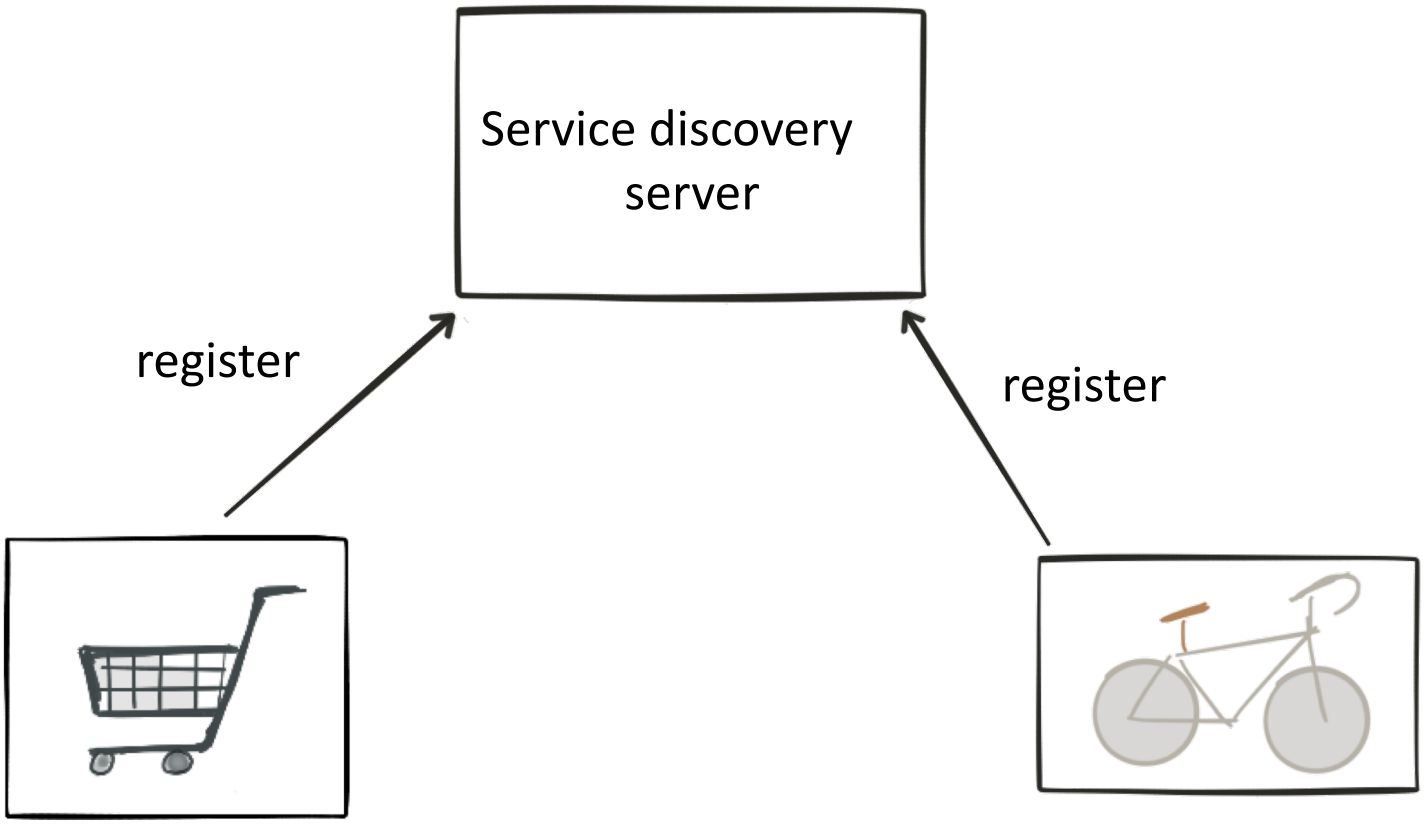
Load balancer



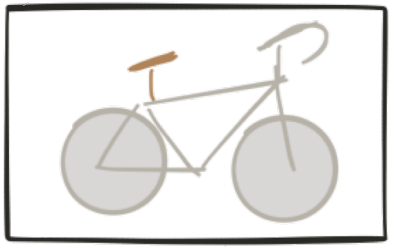
Load balancer (client side)



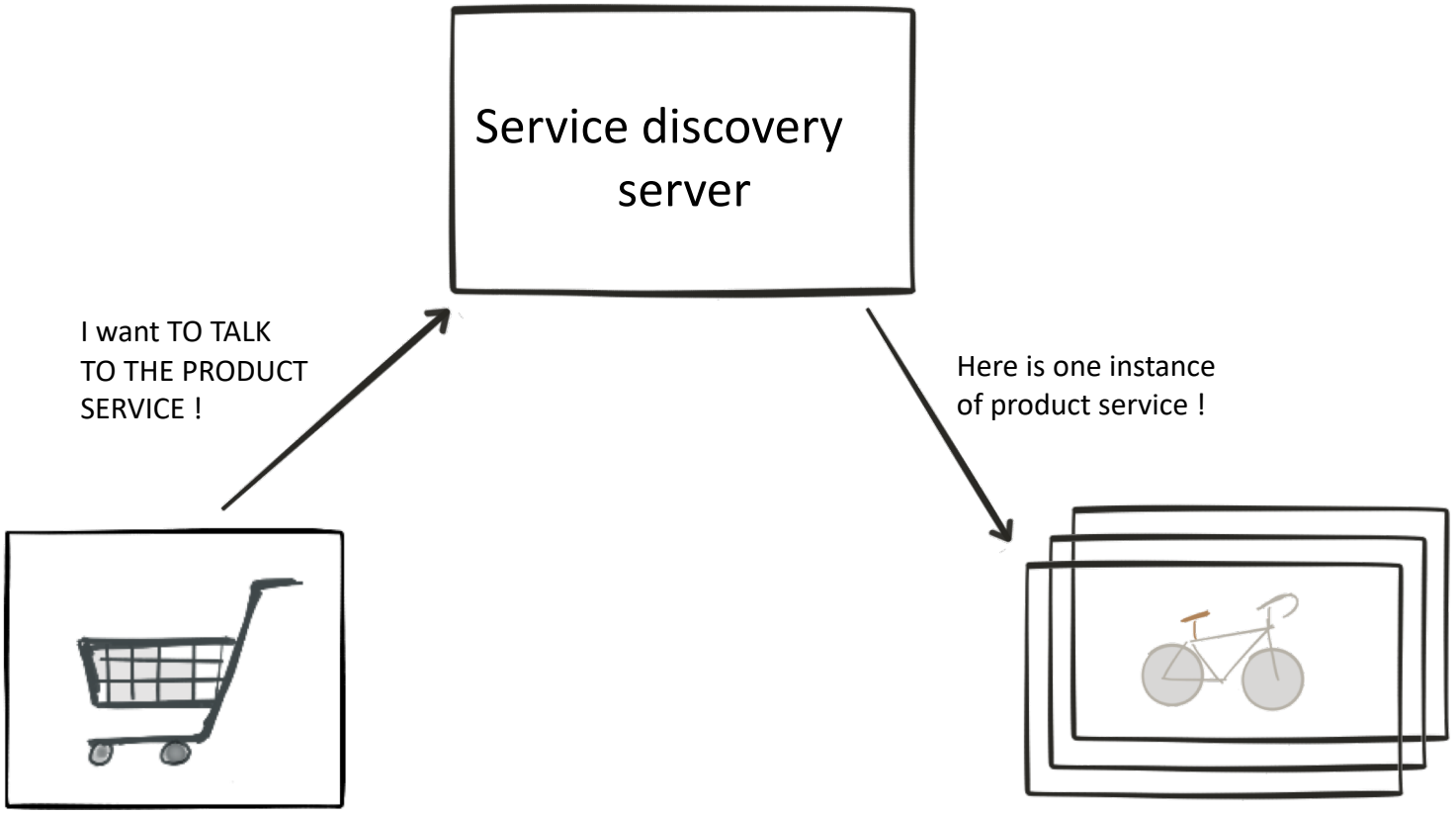
Service discovery



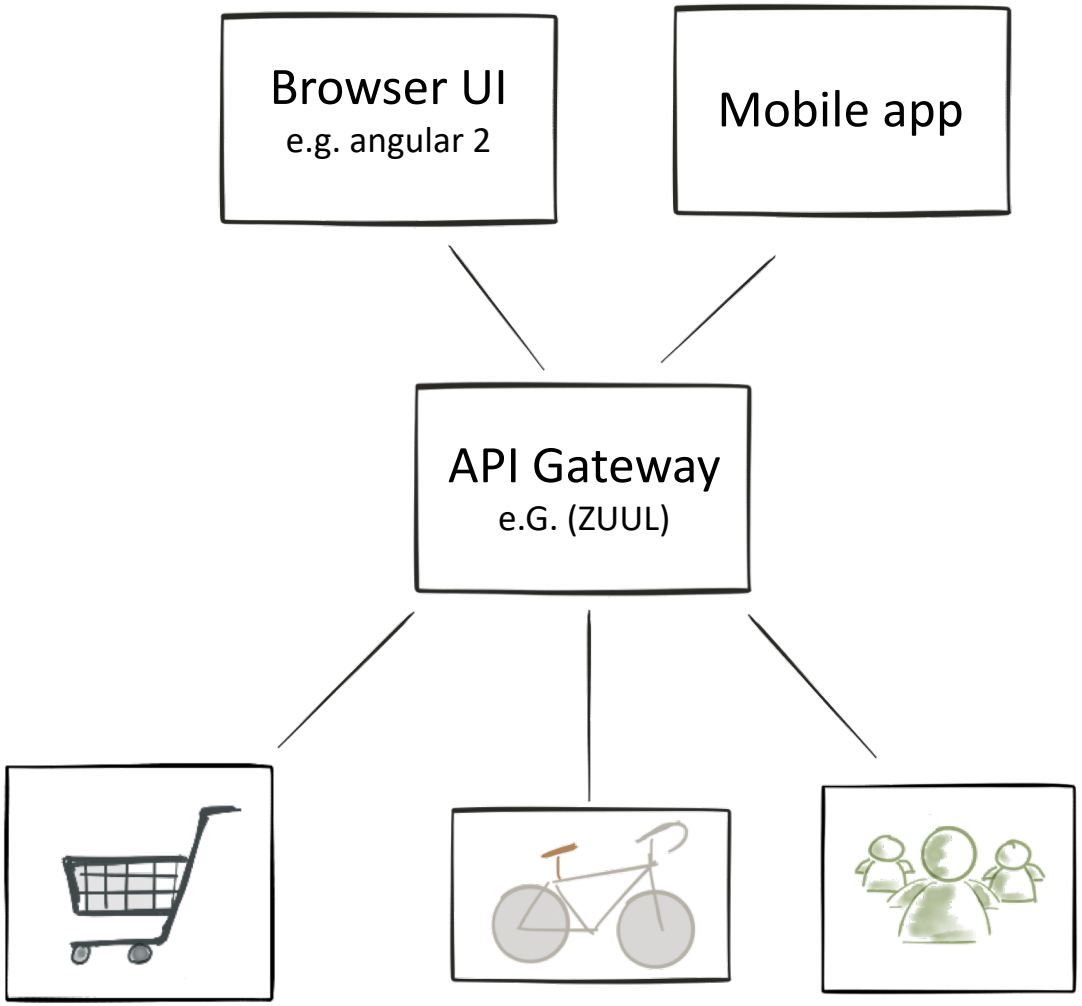
Service discovery



Service discovery (load balancing)



Api Gateway



This is not new!

The old new thing...

Principles

- Modularity
- Autonomous
- hide implementation details
- Automation
- Stateless
- Highly observable

advantages

- Polyglot architecture
- Evolutionary design
- Selective scalability
- Big vs small

drawbacks

- Distributed system
- Synchronous vs asynchronous
- Slow (http)
- Requires an ecosystem

ecosystem

- Load balancer
- Service discovery
- API gateway

THANKS !