Introduction to Microservices

Nanda

Architectural style

This is not new!

The old new thing...

SOA

service oriented architecture

- Modernized version of SOA

New world:

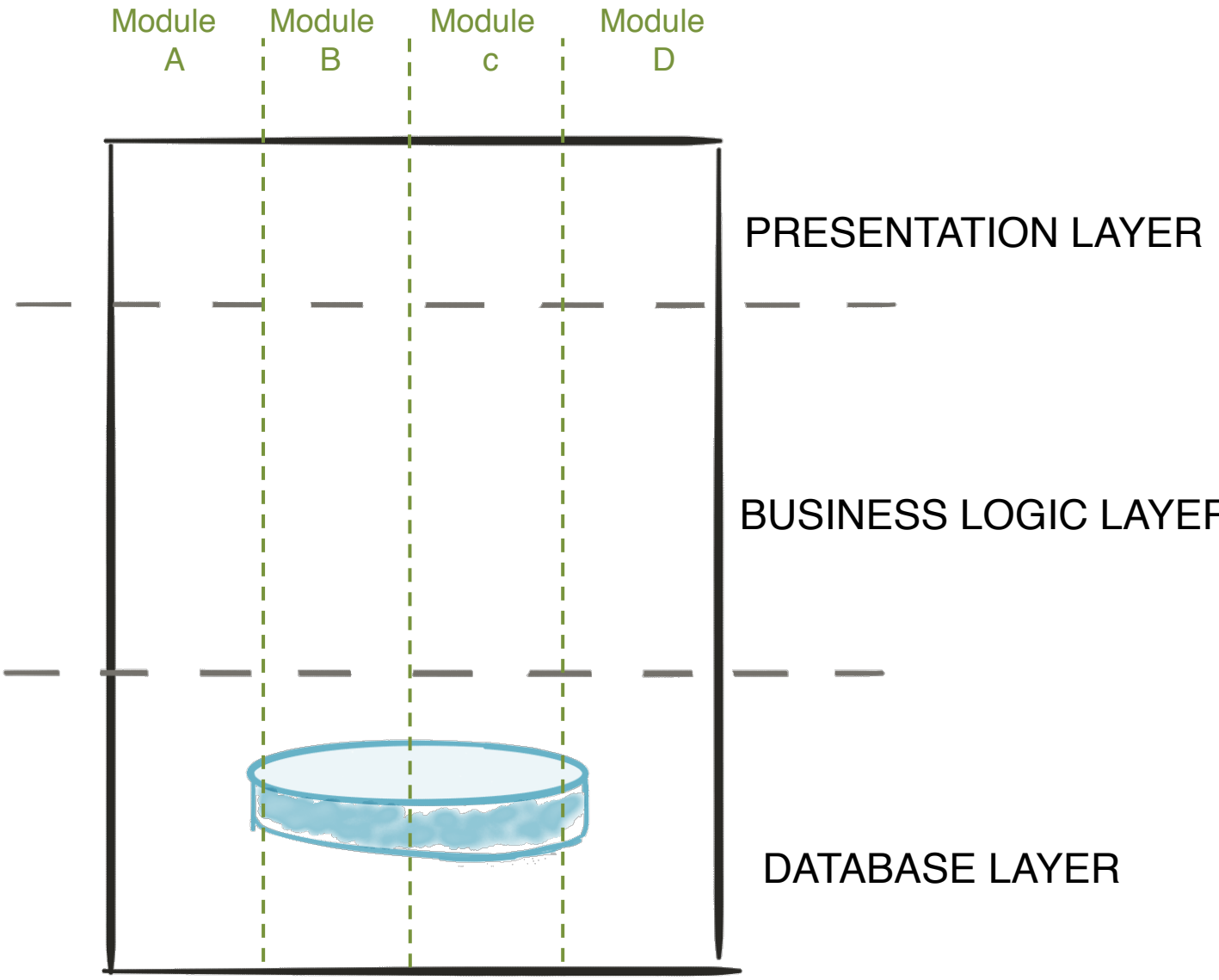
- Speed of delivery
- Scalability
- Innovation / experimentation
- Cloud / devops

VS

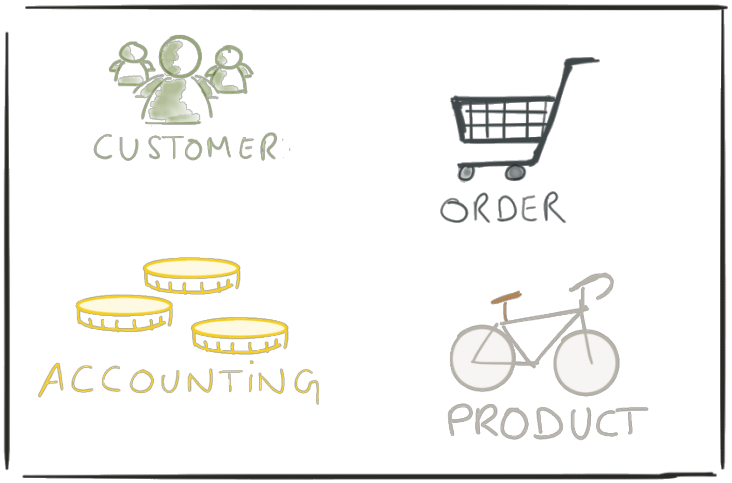
monolith

microservices

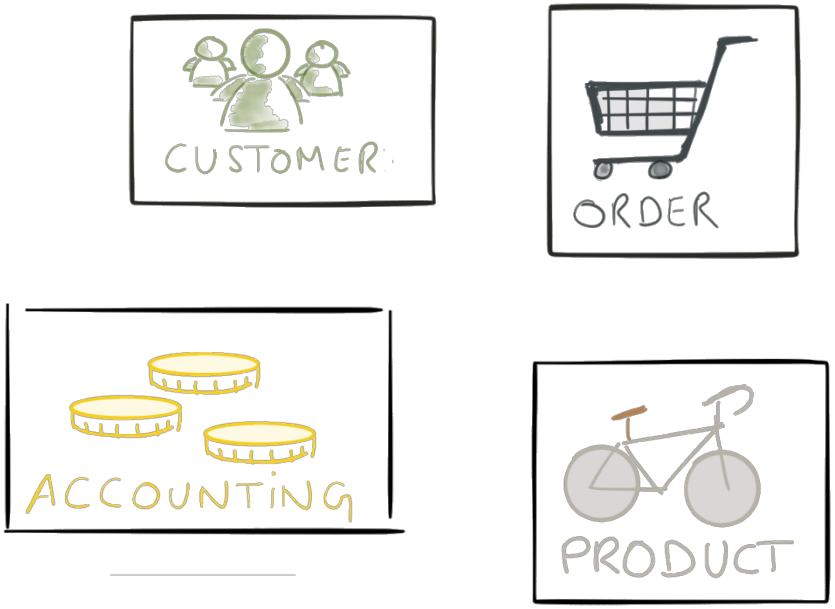
A monolith


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

VS

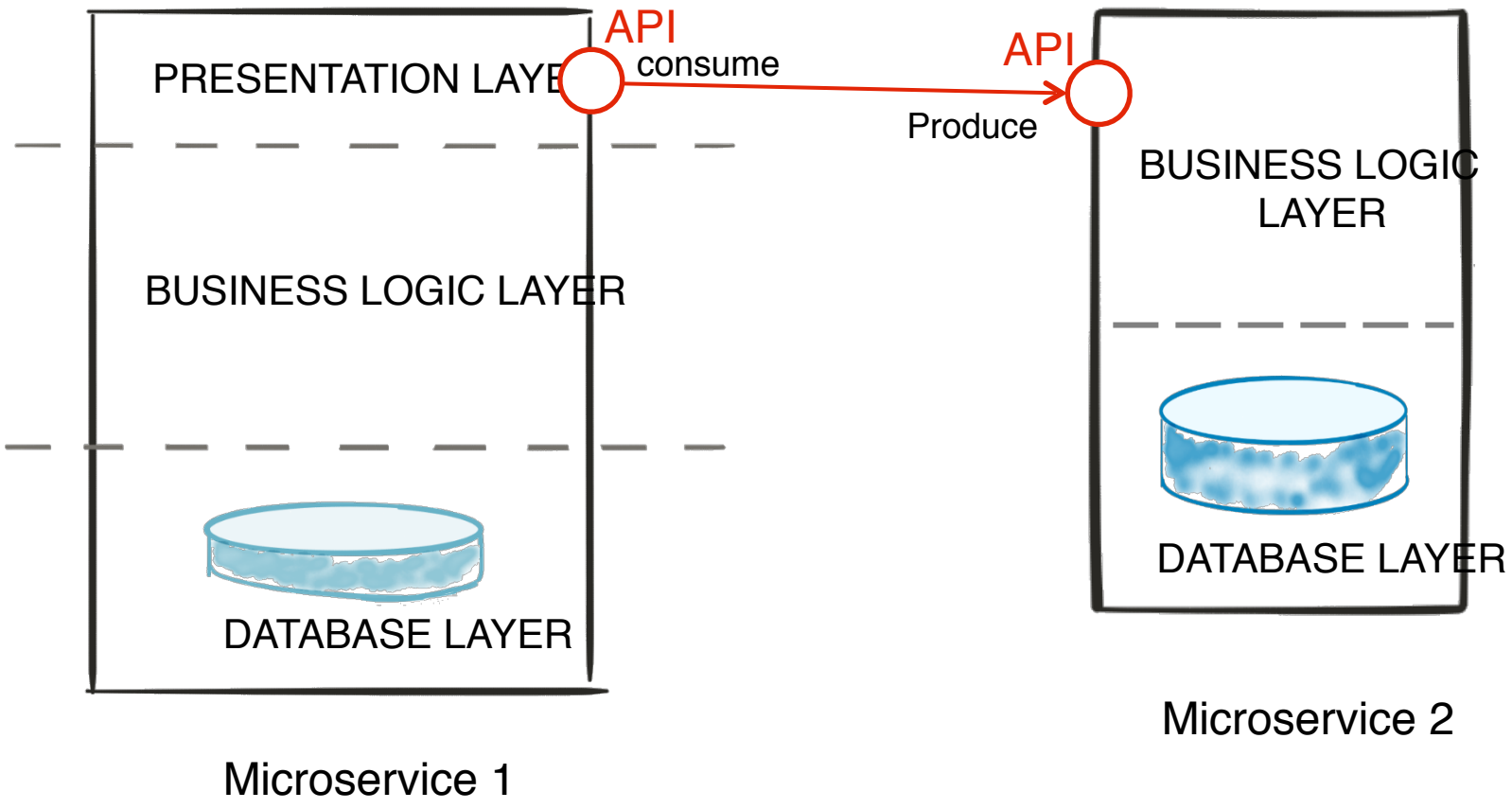


monolith



microservices

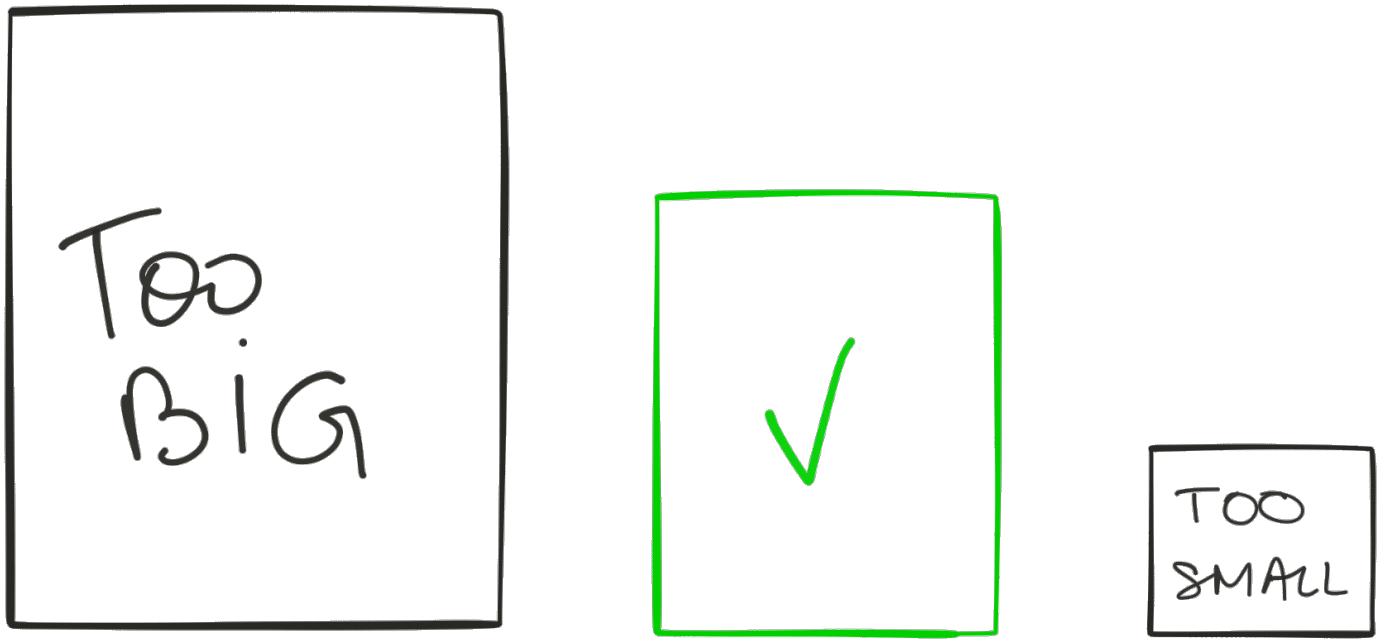
microservices



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

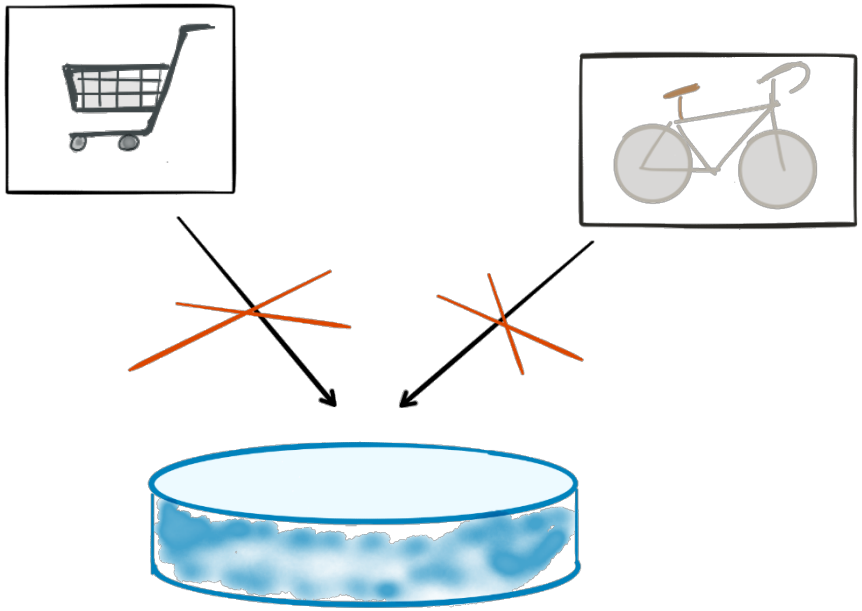
UI

dba

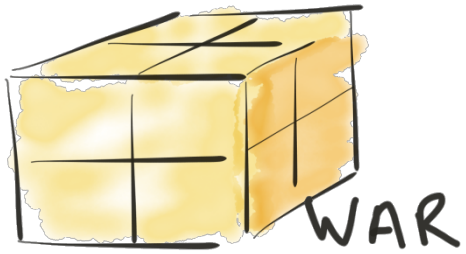
SERVER

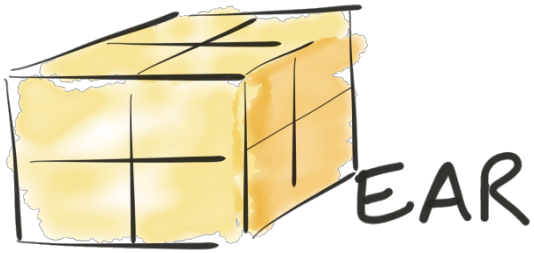
microservices

autonomous



autonomous

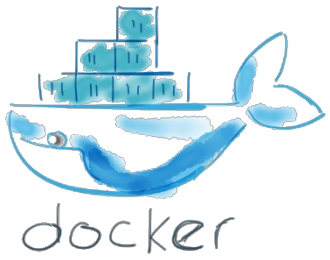




✓

- Libraries
- http listener

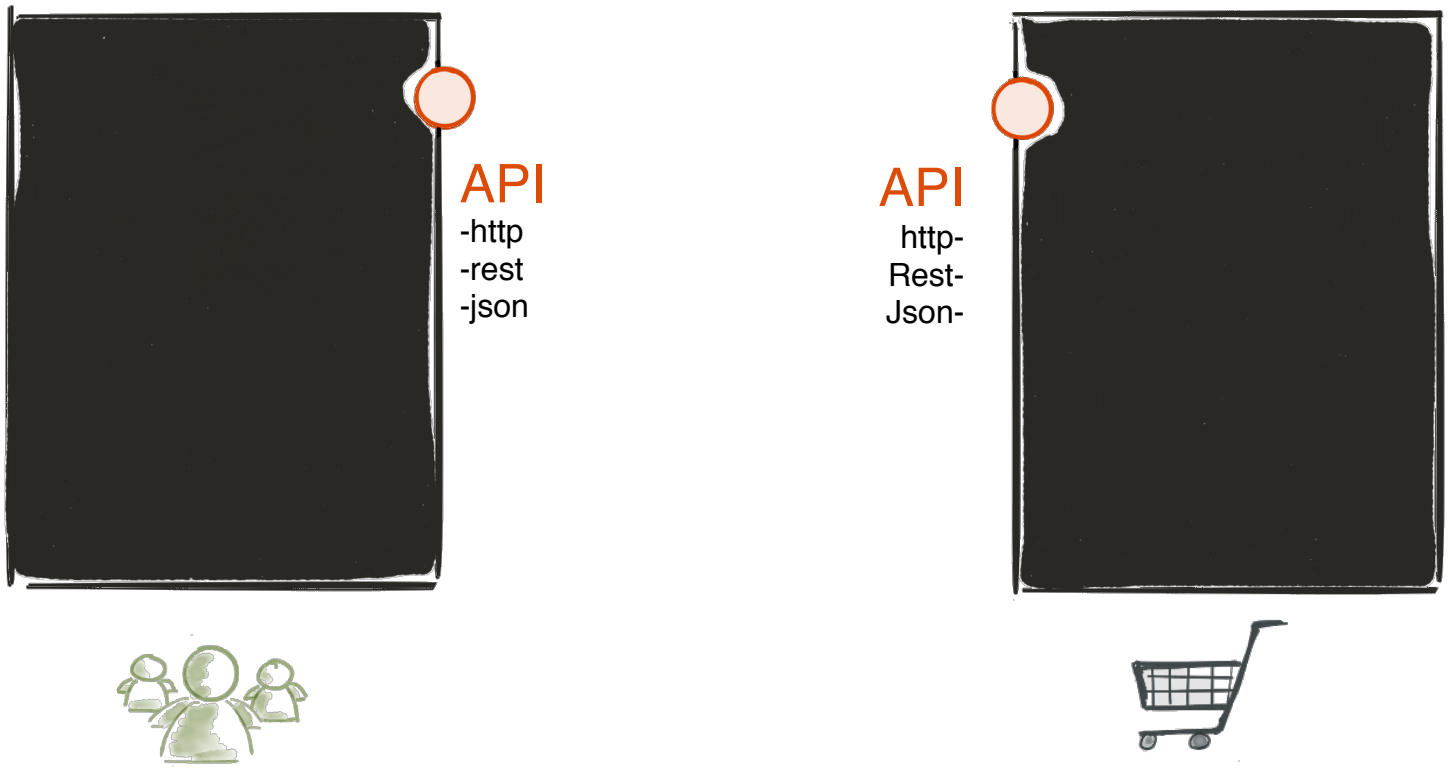
✓



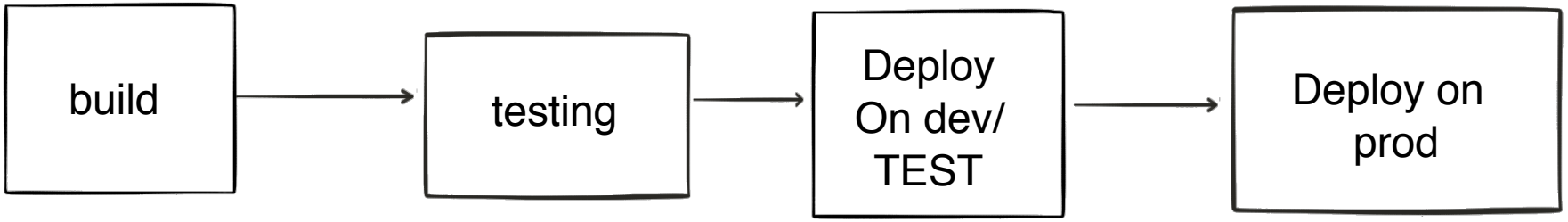
monolith

microservices

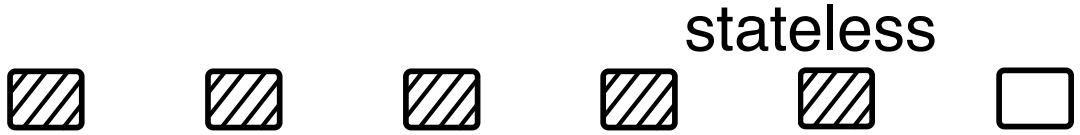
hide implementation details

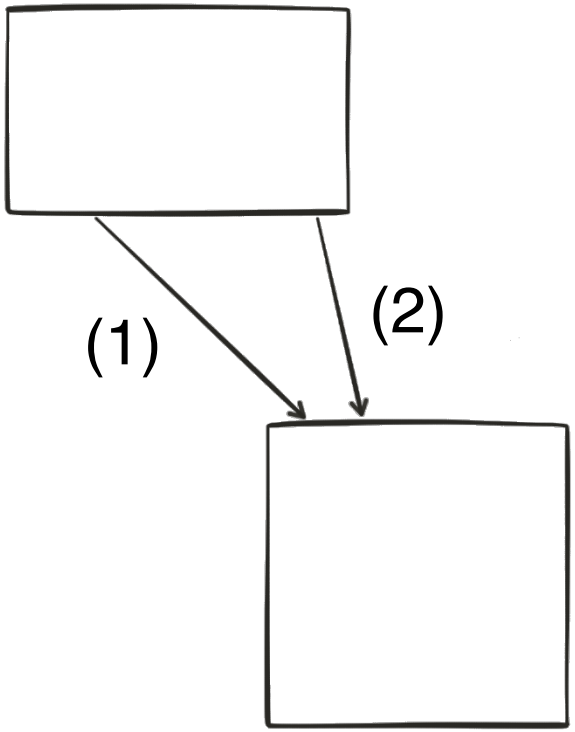


automation



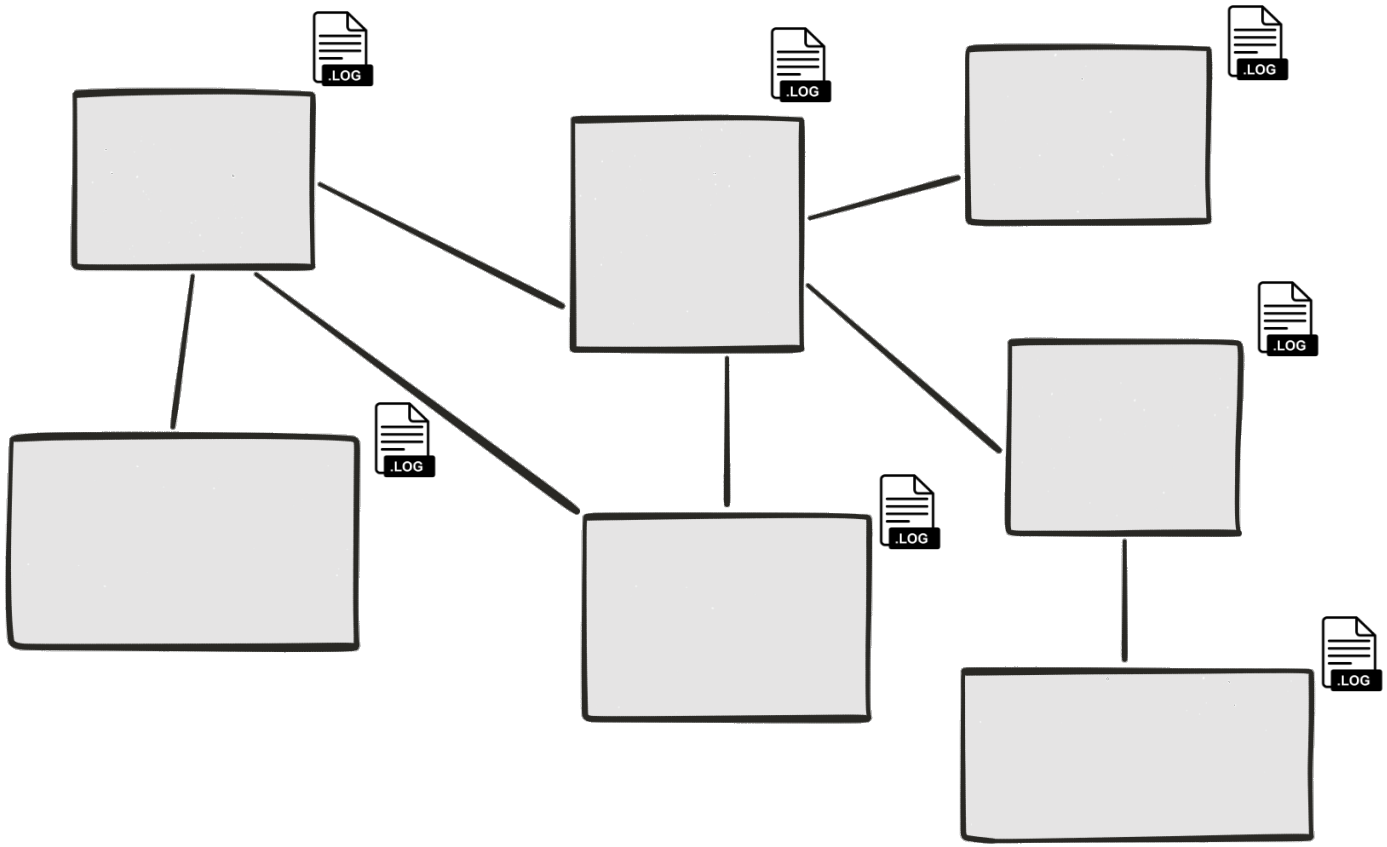
- Continuous integration
- Continuous deployment


stateless


(1)
(2)

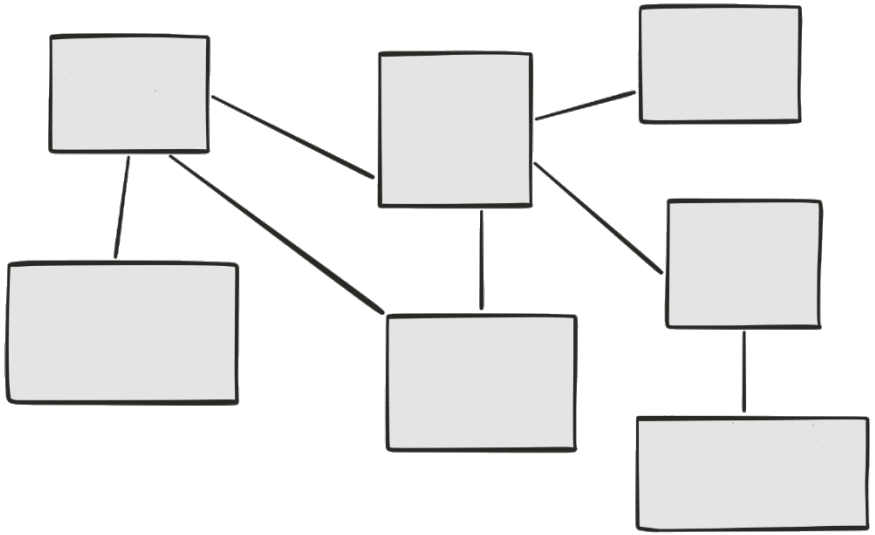
Highly observable

Logs



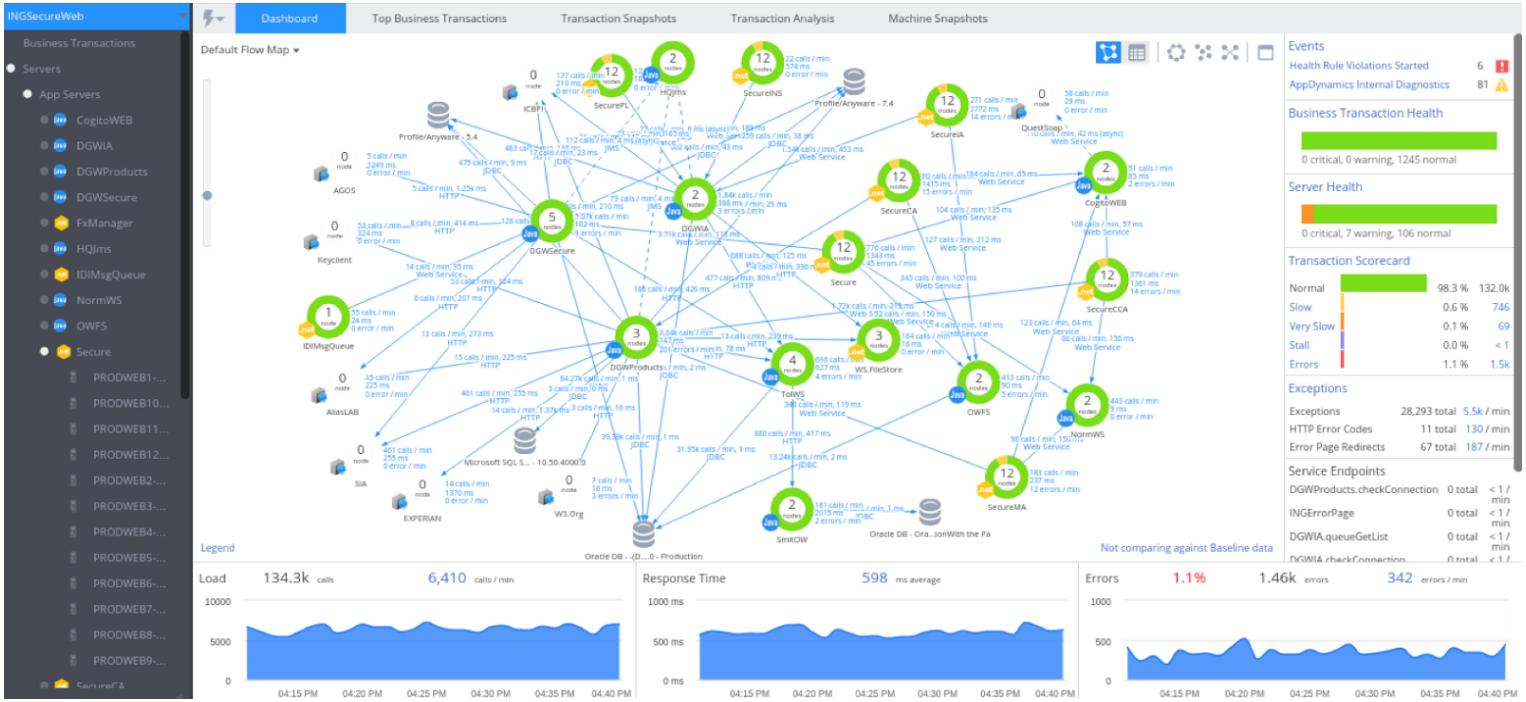
Highly observable

Centralized logging



Highly observable

monitoring



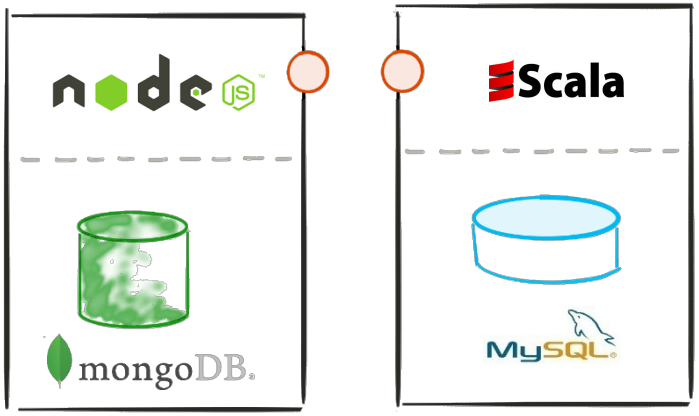
AppDynamics

principles

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

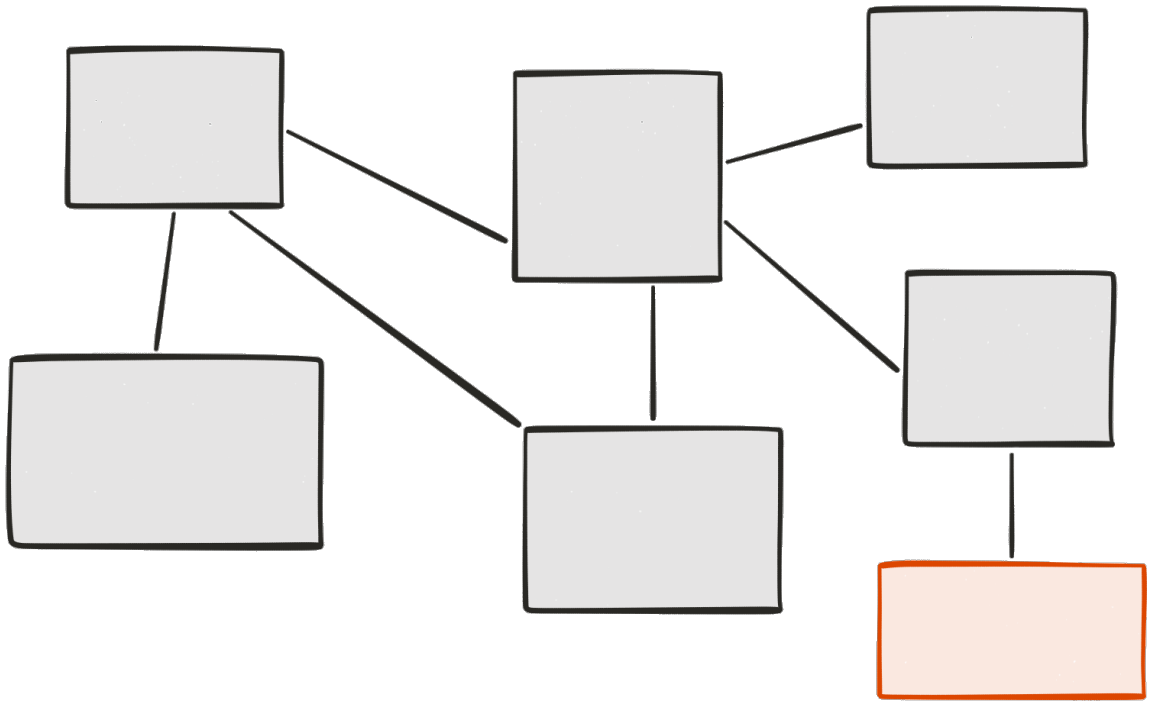
Advantages

Polyglot architecture



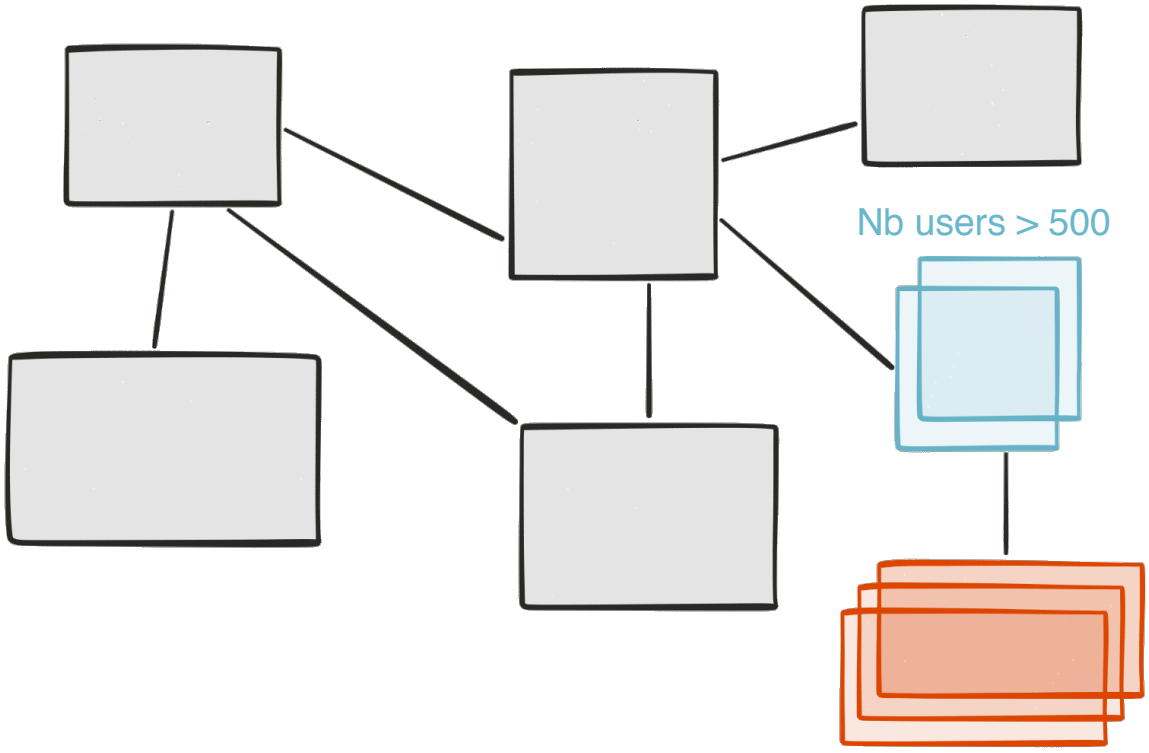
- The right technology for the job
- reduce technical debt

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

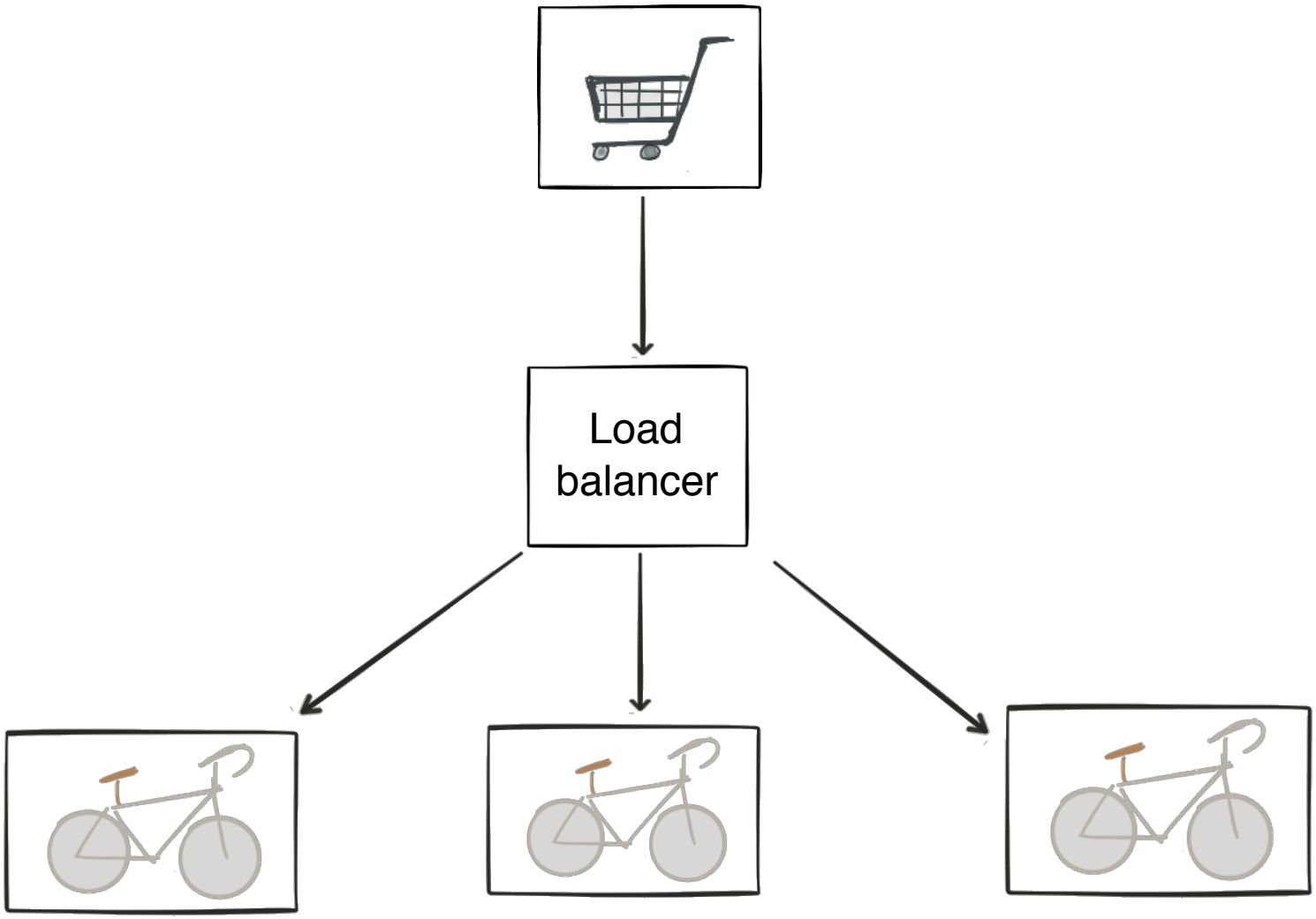
Conclusion:

- The Microservices architecture is more complex
Than a monolith.

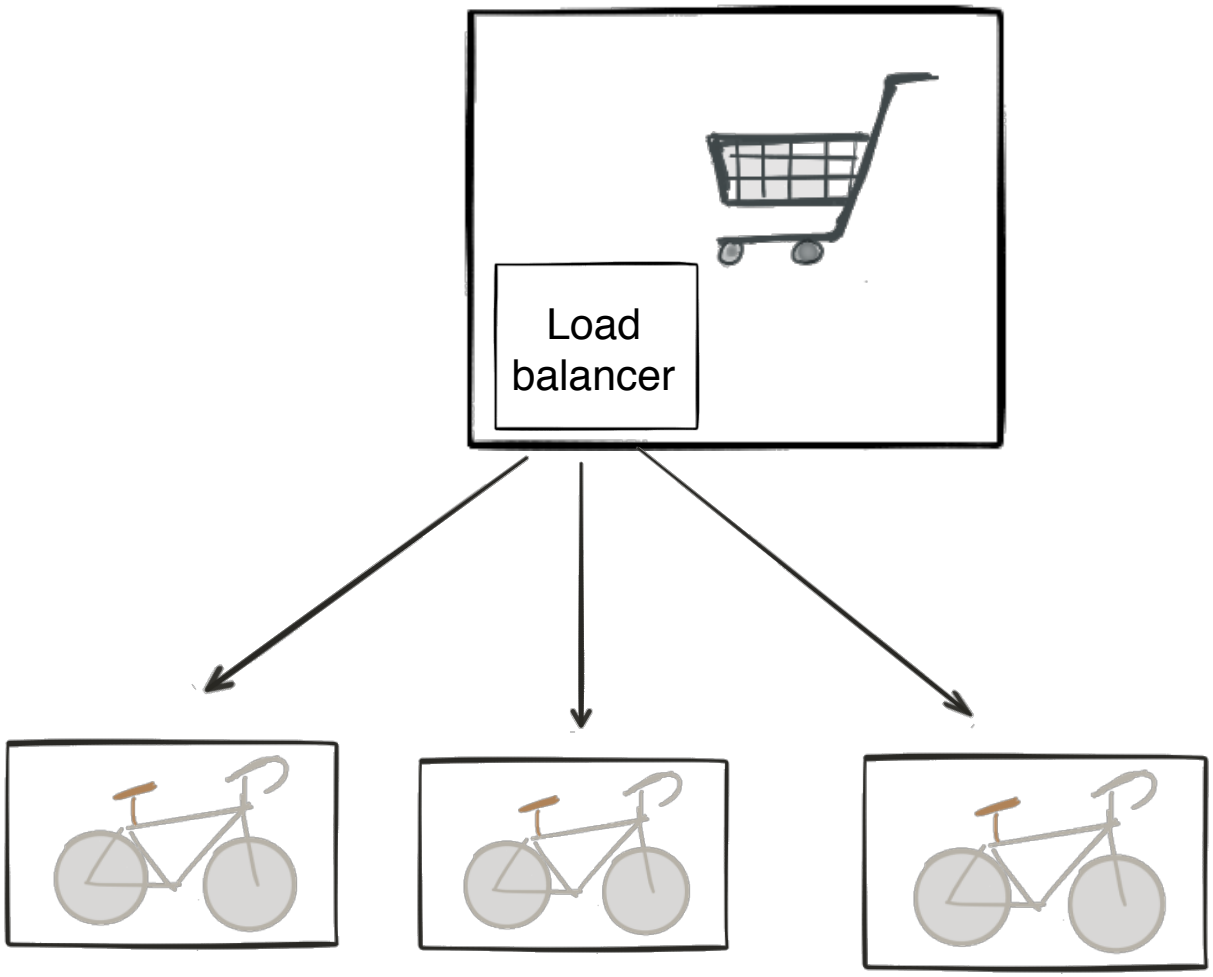
- This the cost of growing and scaling easily

Microservices ecosystem

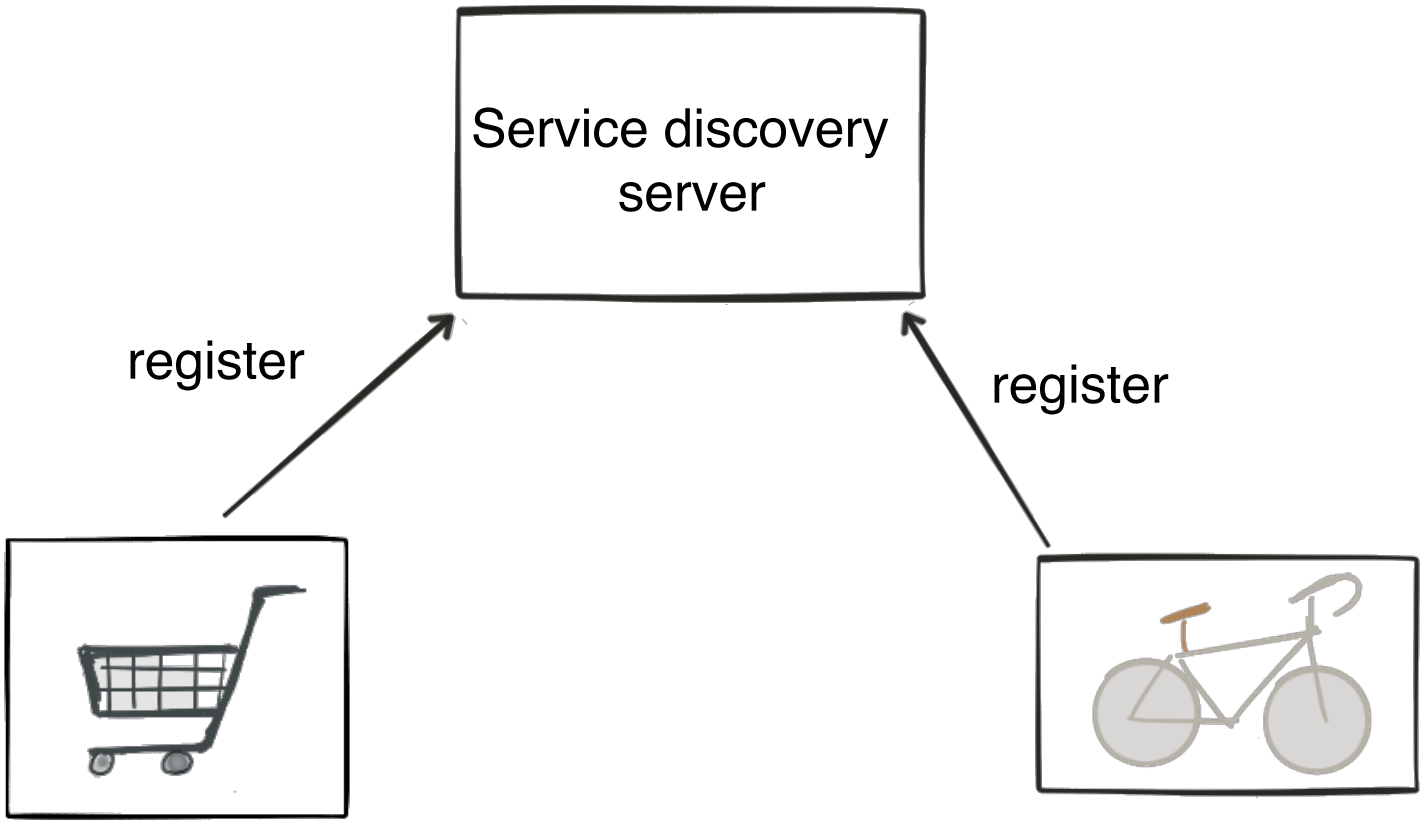
Load balancer



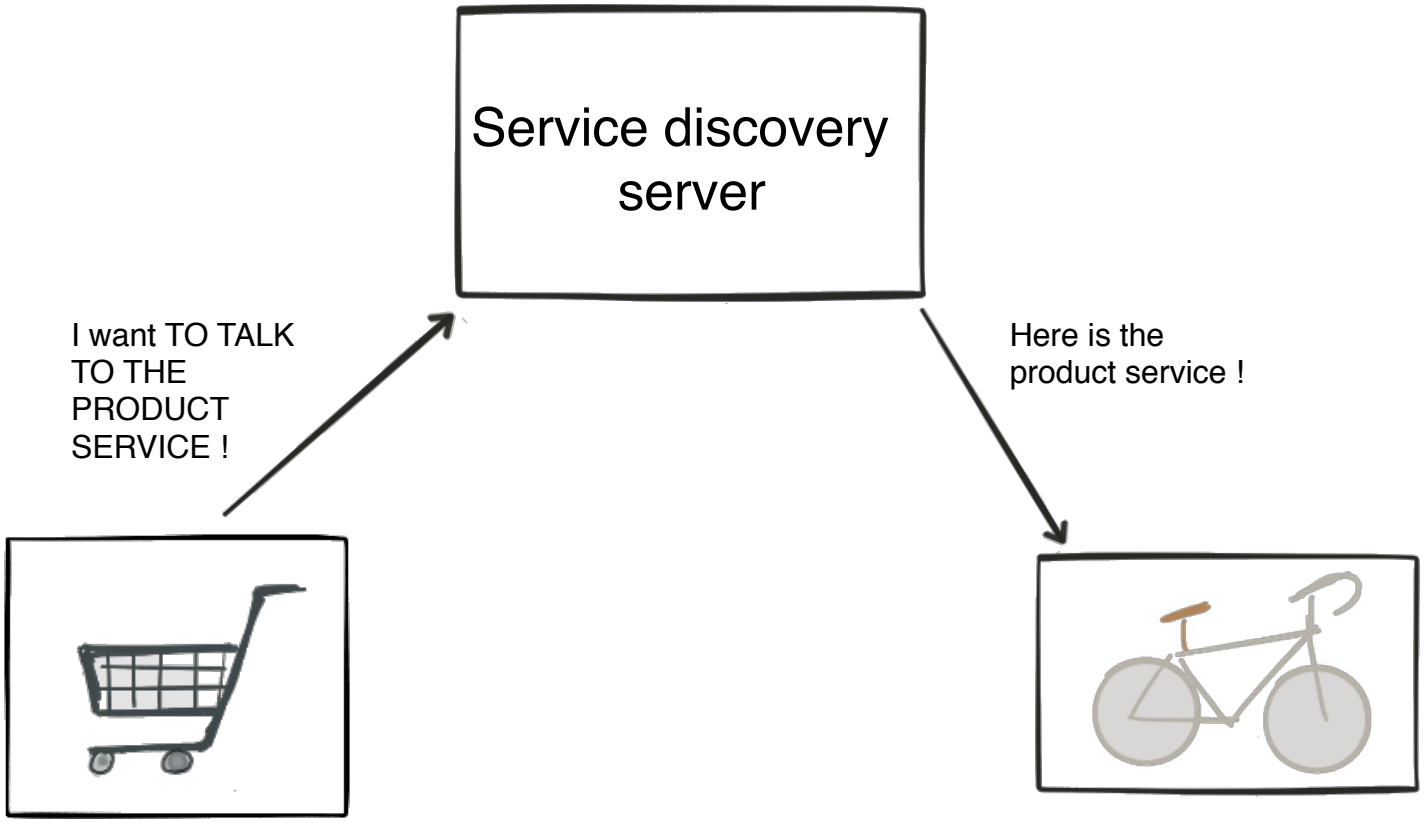
Load balancer (client side)



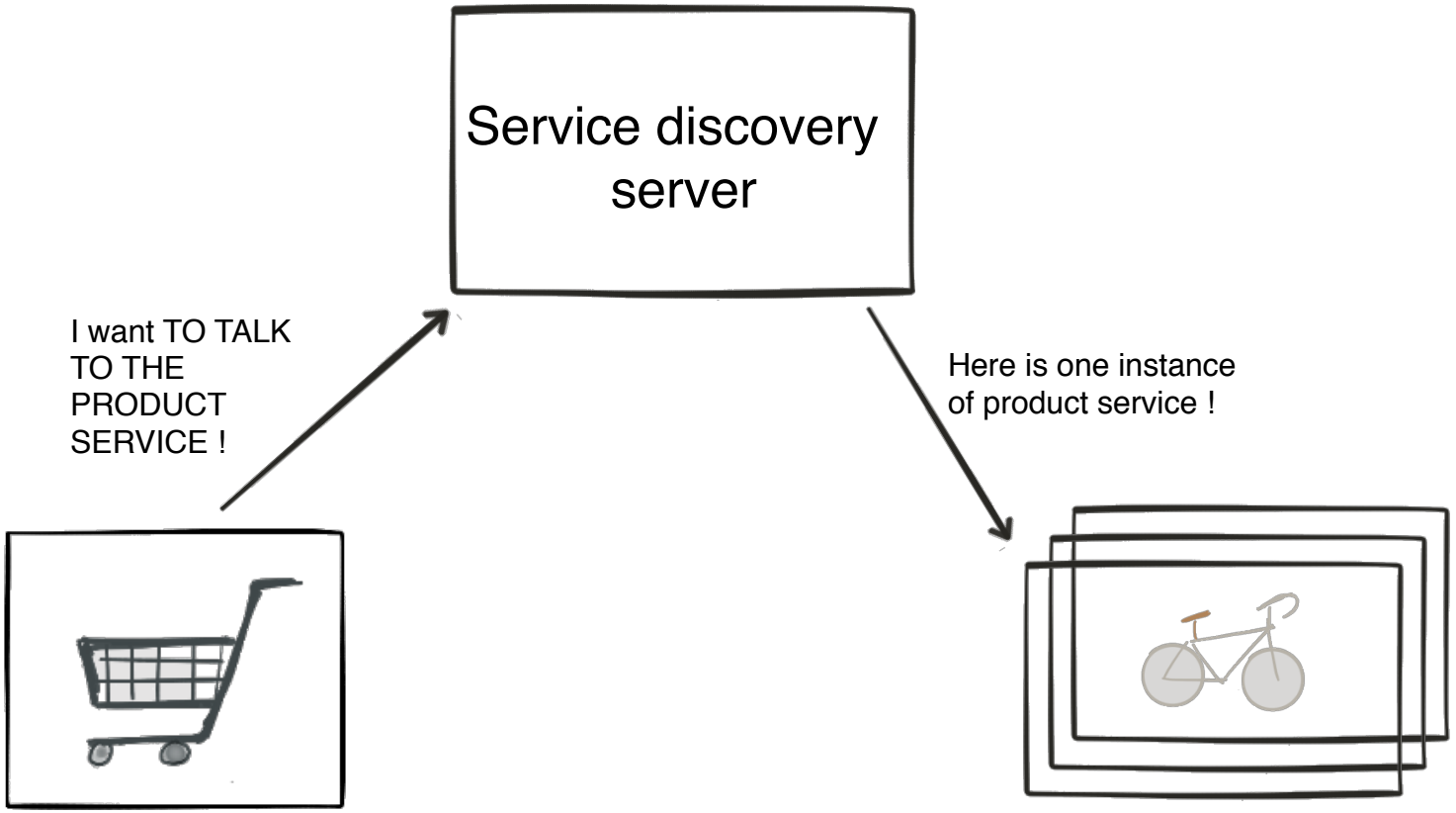
Service discovery



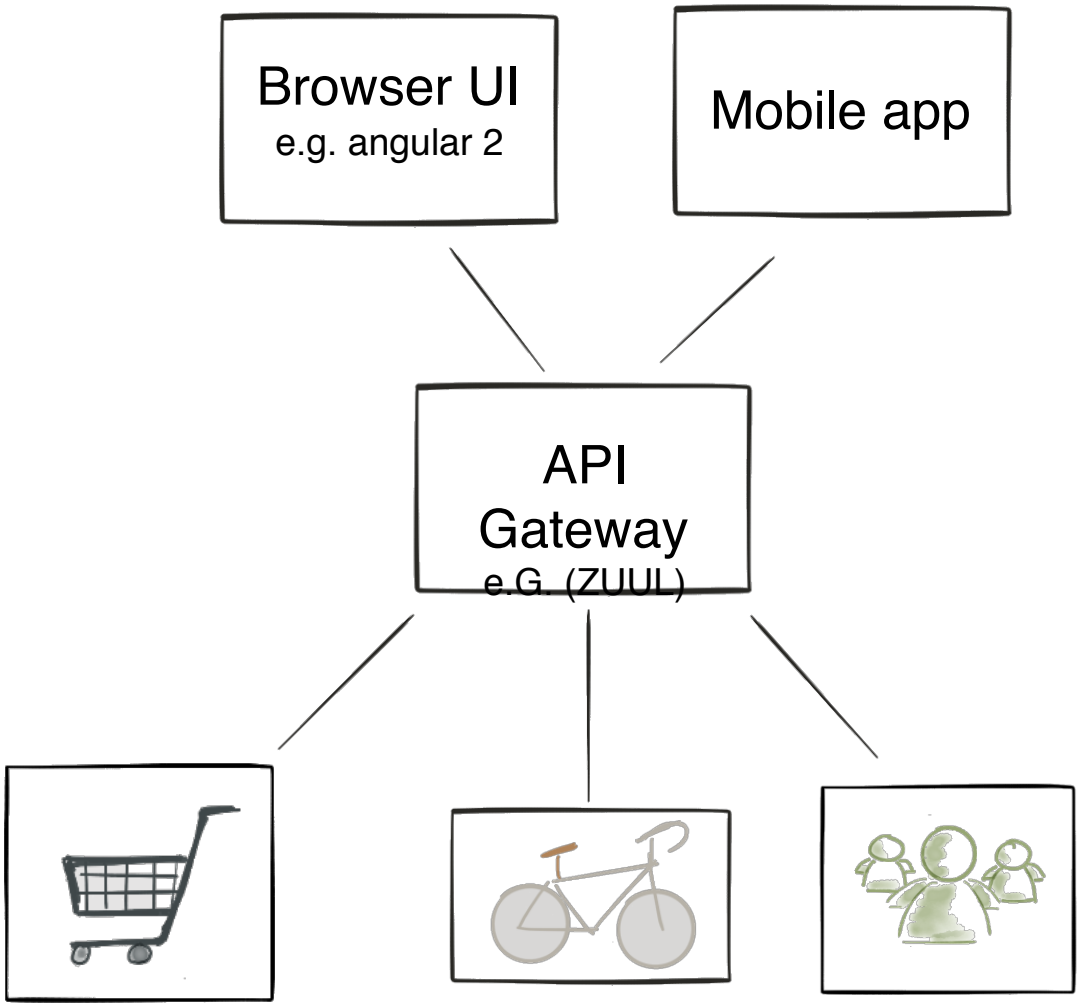
Service discovery



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

Types of Microservices

Self-contained microservices

Self-contained microservices also are packaged into a single fat JAR file, but these also include an embedded framework with optional compatible third-party libraries like *Spring Boot* and *Wildfly Swarm*.

In-container microservices

Lastly, in-container microservices package an entire Java EE Container and its service implementation in a Docker image.

The container provides verified implementations through standard APIs. This gives your developer the leeway to focus directly on business functionality.

Java Frameworks for Microservices

Spring Boot. Easily one of the best Java microservices framework, Spring Boot integrates optimally with other supporting languages.

Spark Framework. Spark is a is a free and open-source software Web Application and domain-specific language written in Java. It runs on an embedded Jetty web server by default but can be configured to run on other web servers.

Swagger. This framework with the coolest name helps your developers document API. Swagger also gives you a development portal, which allows users to test your APIs.

Dropwizard. Dropwizard pulls together mature and stable Java libraries in lightweight packages that you can use for your own applications.

Play Framework. Play Framework offers an easier way to build, create and deploy Web applications using Scala and Java. Play Framework also has one of the biggest communities out of all microservices frameworks.

Java Frameworks for Microservices

Jersey. This open source framework supports JAX-RS APIs in Java and is very easy to use.

Restlet. Restlet helps developers create fast and scalable Web APIs that adhere to the RESTful architecture pattern. It has good routing and filtering, and is available for many major frameworks .

Restx. This lightweight, modular, feature rich, incredibly fast open source Java REST framework is a great framework for microservices.

THANKS !