สยามชำนาญกิจ
SIAM CHAMNAN KIT

Microservices

Somkiat Puisungnoen

Somkiat | Home

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

Intro

Software Craftsmanship

Software Practitioner at สยามชำนาญกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post | Photo/Video | Live Video | Life Event

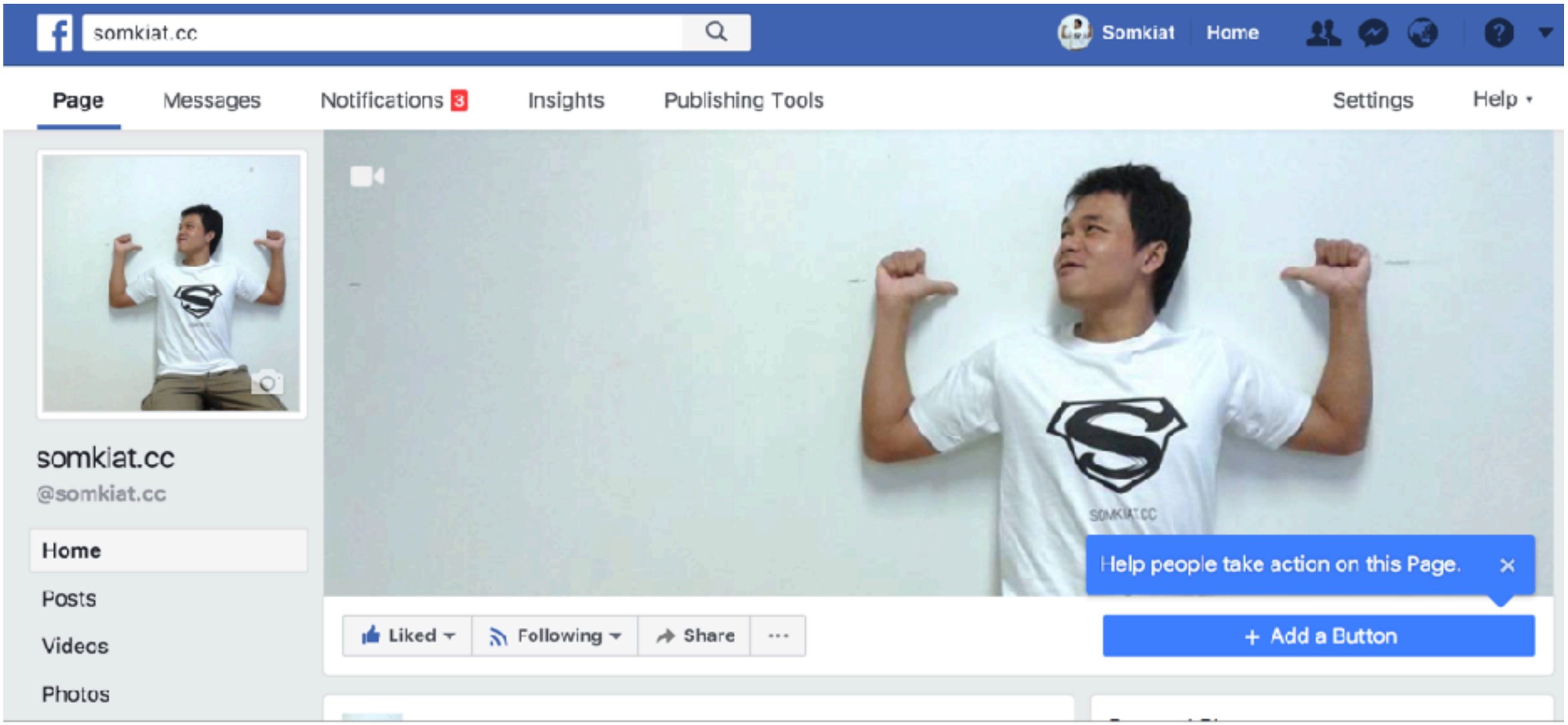
What's on your mind?

Public | Post

Somkiat Puisungnoen

15 mins · Bangkok ·

Java and Bigdata



Agenda

Cloud Native Application

Microservices and DevOps

The architecture of Microservices

How to model Microservices

Integrating multiple Microservices

Agenda

Testing and Developing Microservices

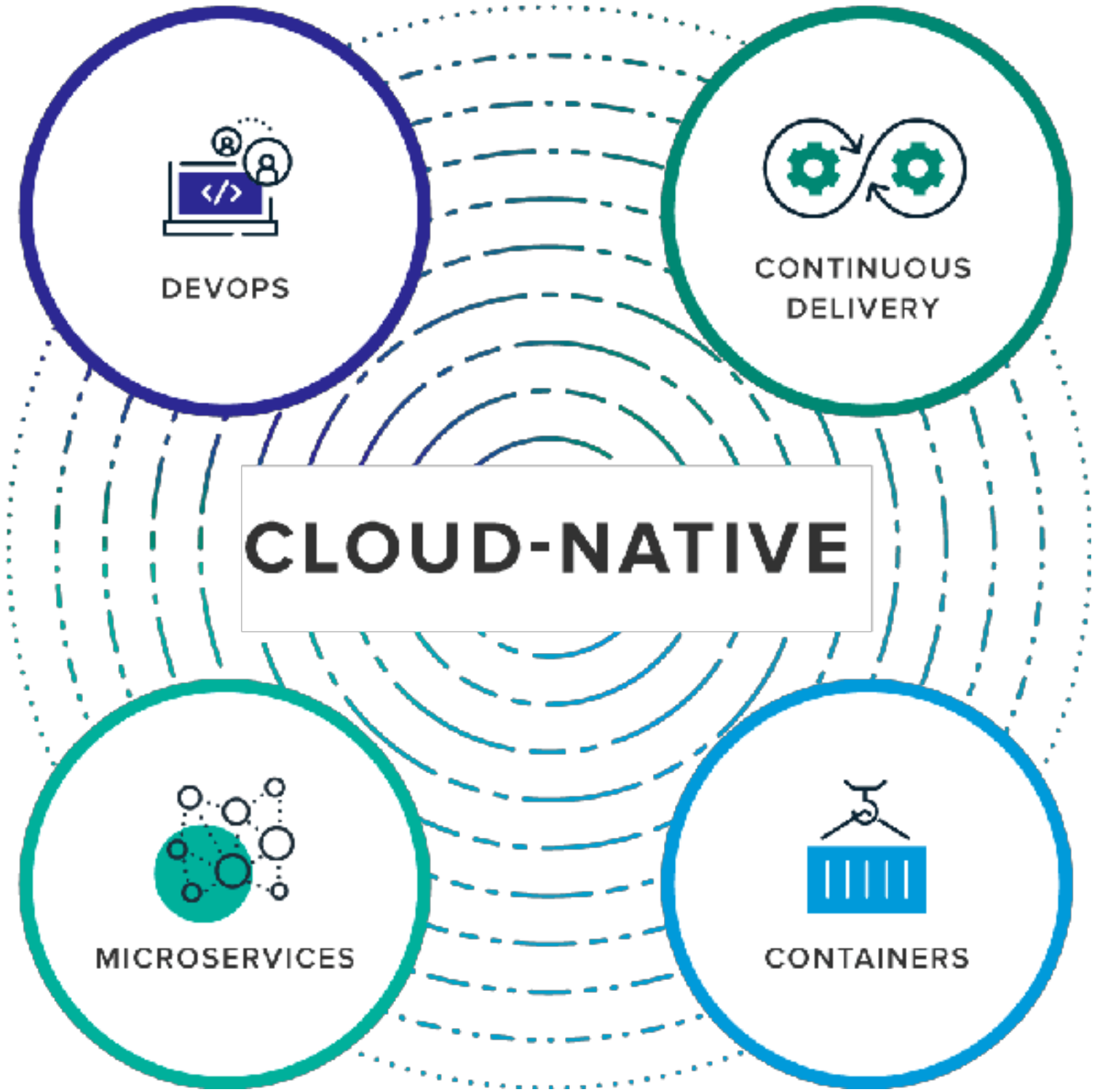
Deploying Microservices

Monitoring Microservices

Scaling up your Microservices

Workshop

https://github.com/up1/course-microservice



https://pivotal.io/cloud-native

Evolution of Architecture

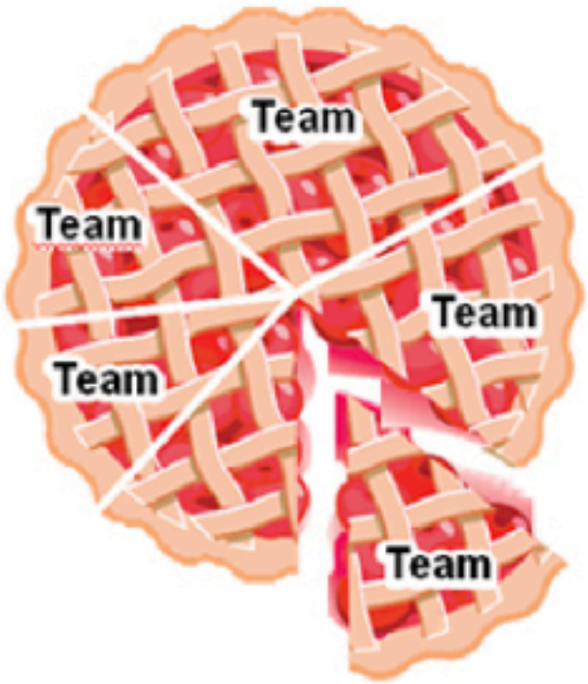
1990s and earlier

Pre-SOA (monolithic)
Tight coupling

For a monolith to change, all must agree on each change. Each change has unanticipated effects requiring careful testing beforehand.

2000s

Traditional SOA
Looser coupling



Elements in SOA are developed more autonomously but must be coordinated with others to fit into the overall design.

2010s

Microservices
Decoupled

Developers can create and activate new microservices without prior coordination with others. Their adherence to MSA principles makes continuous delivery of new or modified services possible.

Microservices

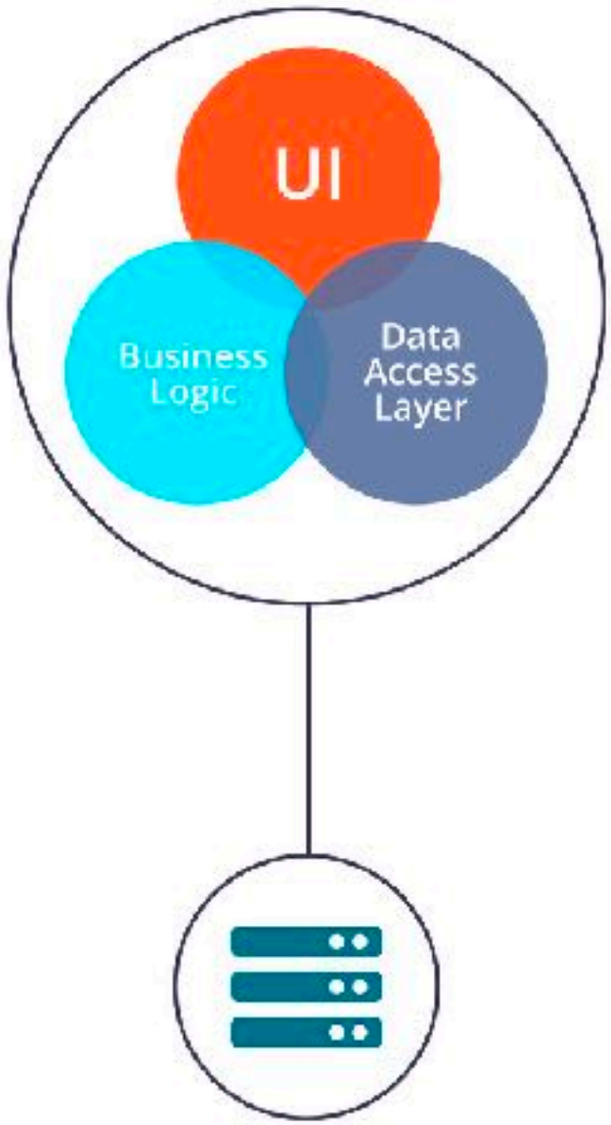
Microservices

Small, Do one thing (Bounded Context)

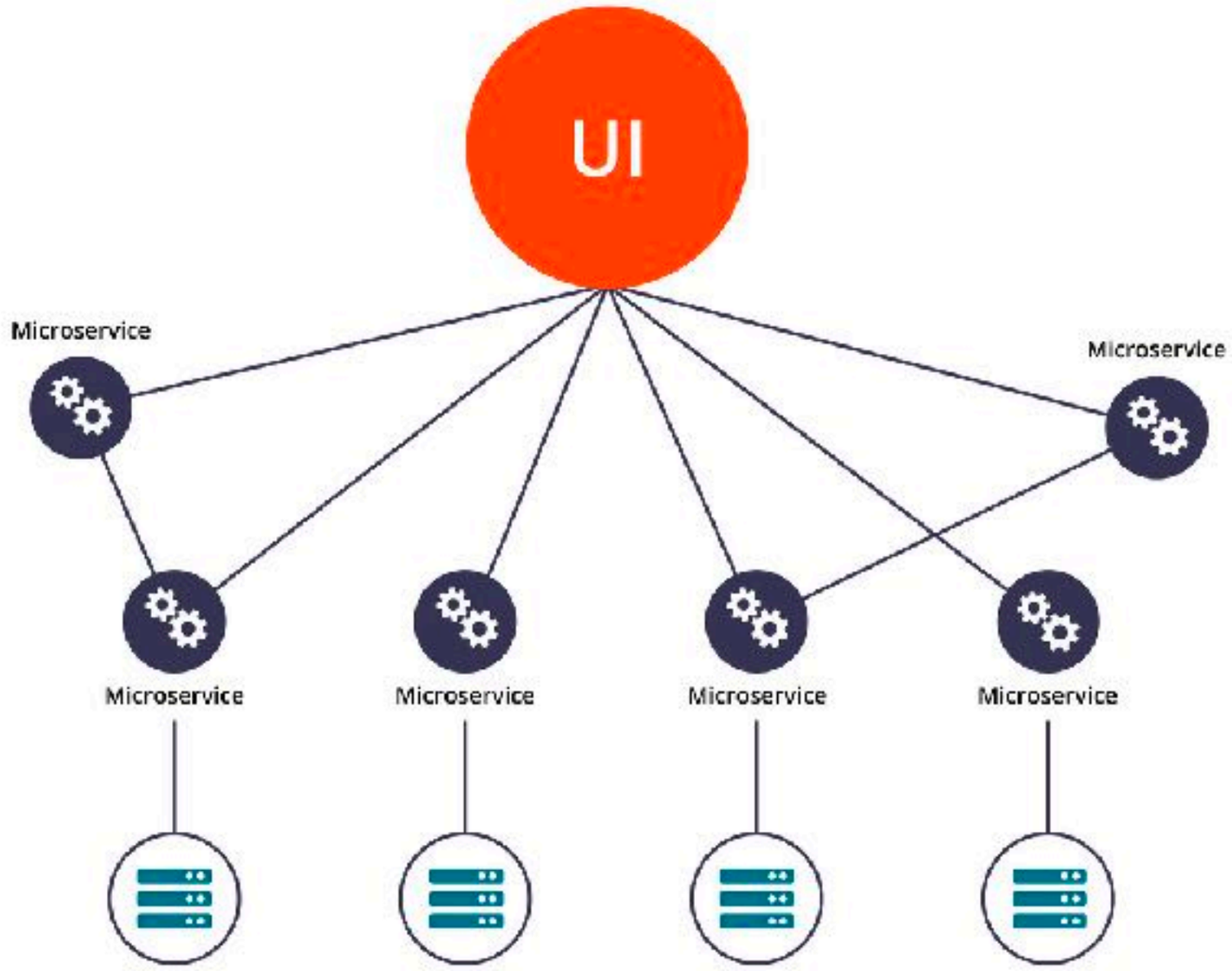
Modular

Easy to **deploy**

Scale independently

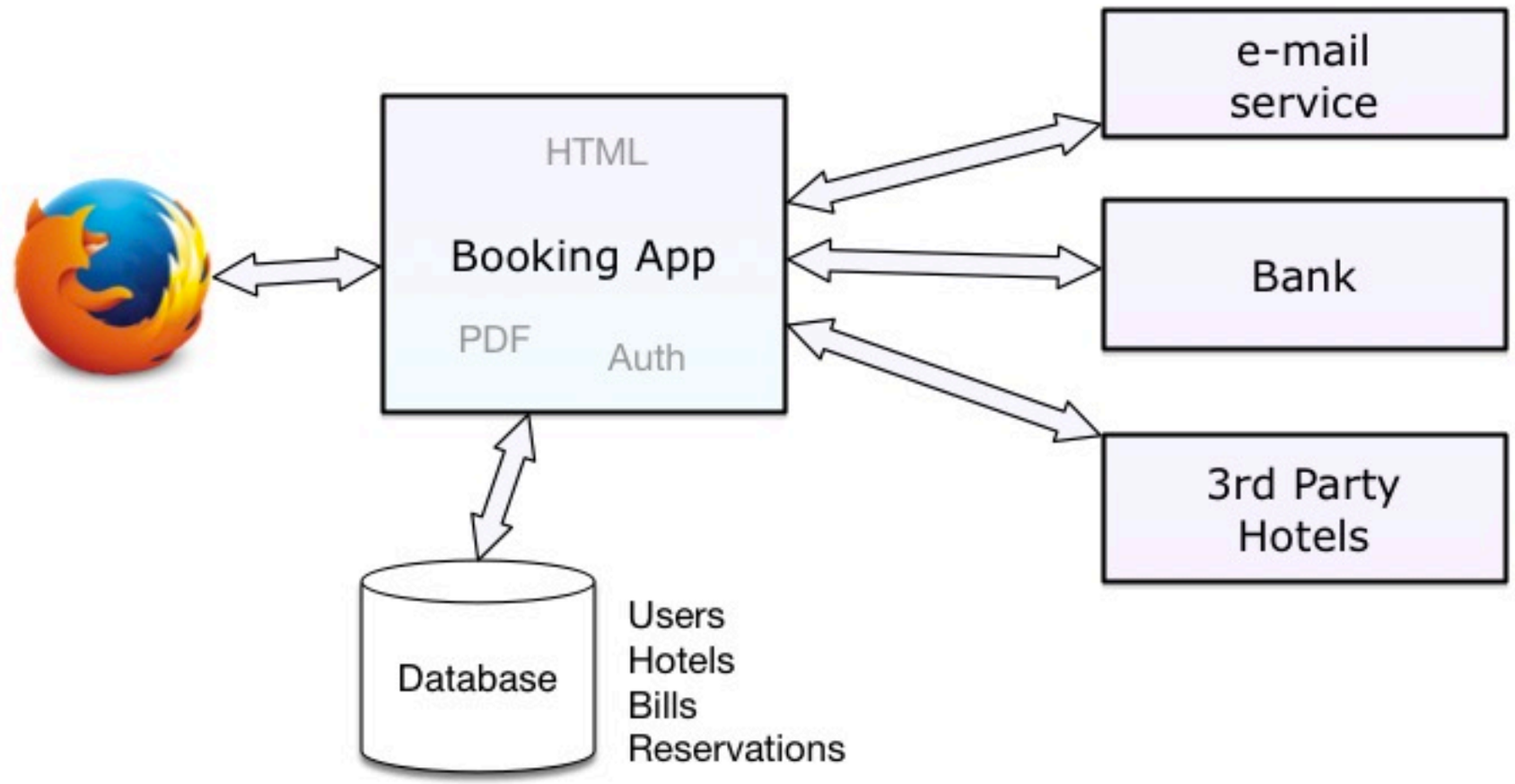


Monolithic Architecture

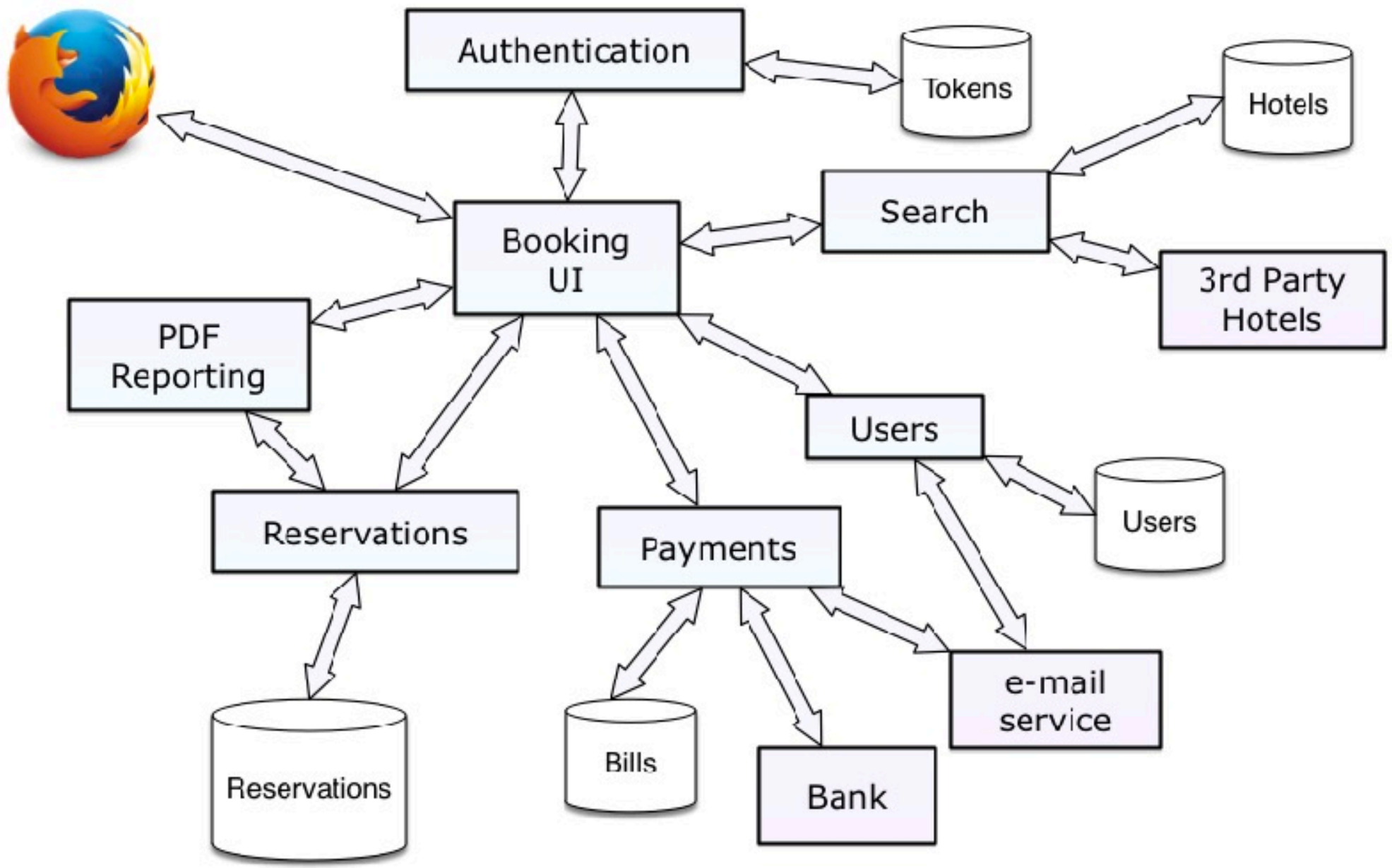


Microservice Architecture

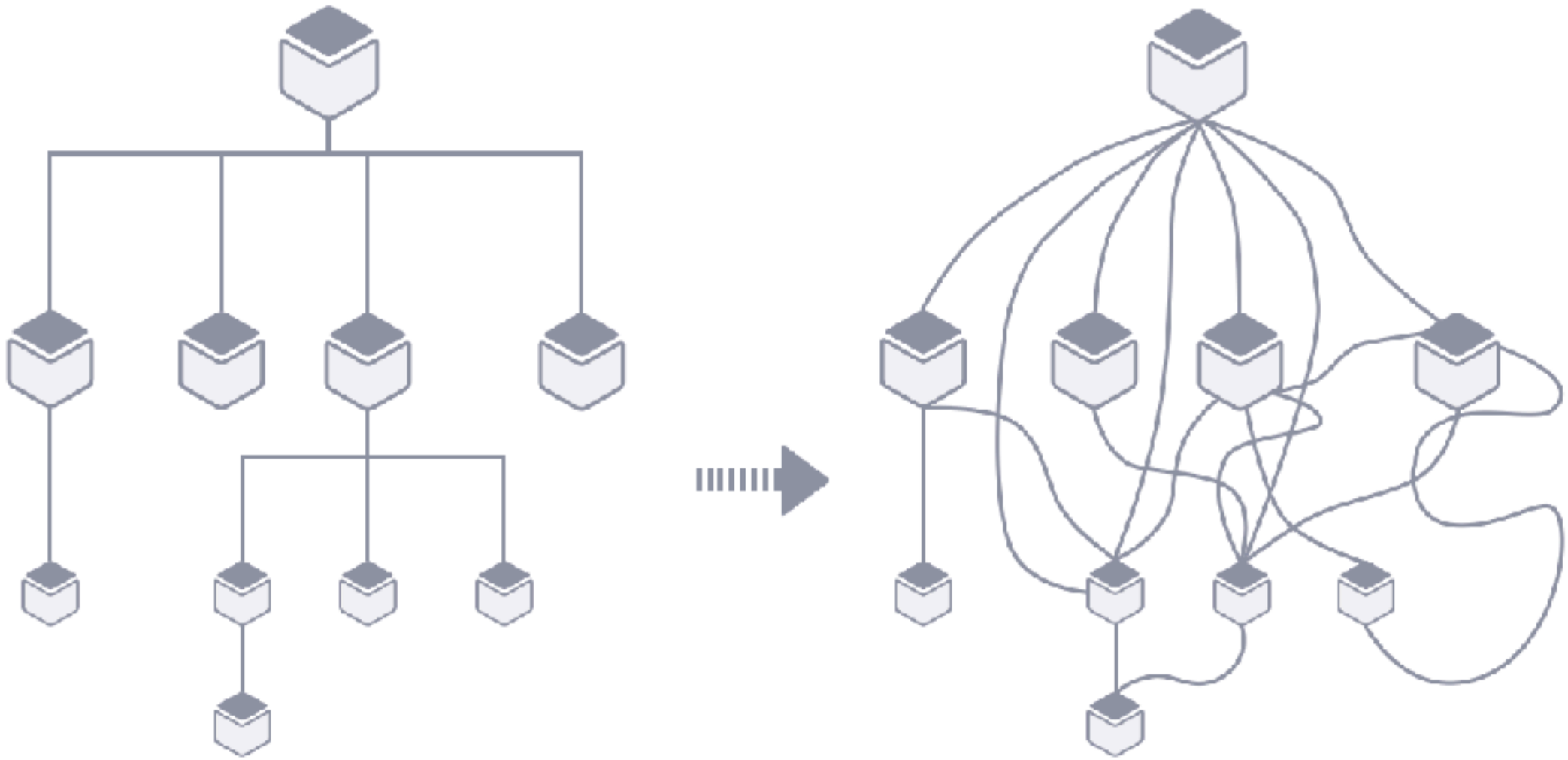
Monolithic



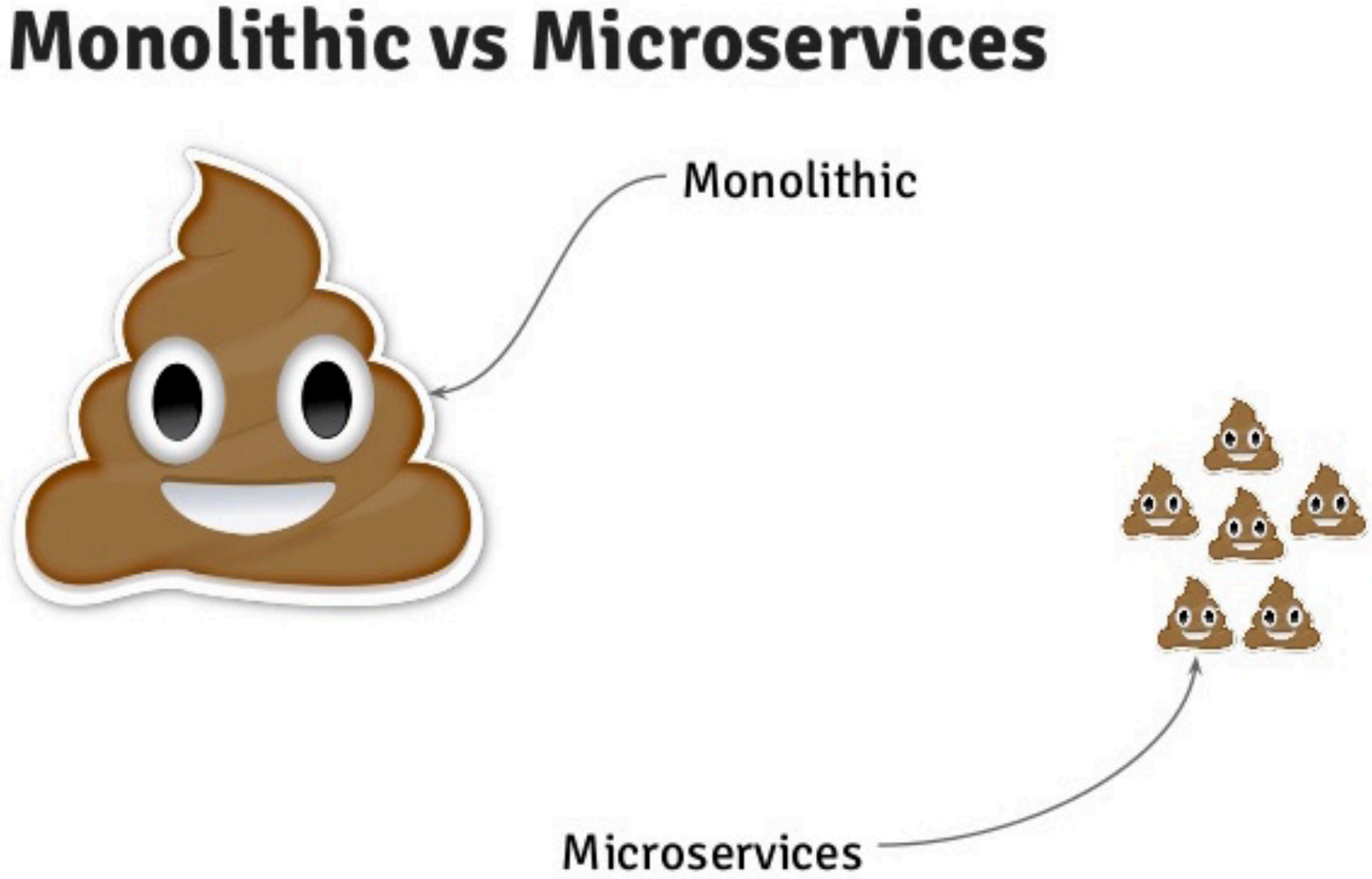
Microservices



Microservices spaghetti



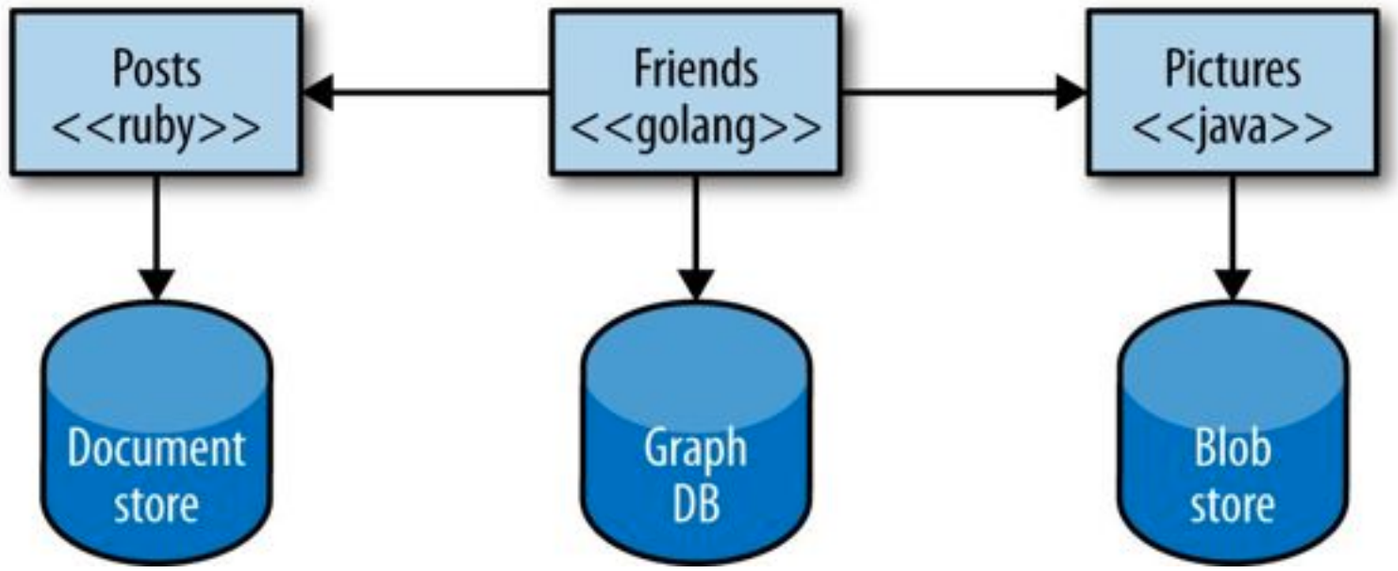
Microservices spaghetti



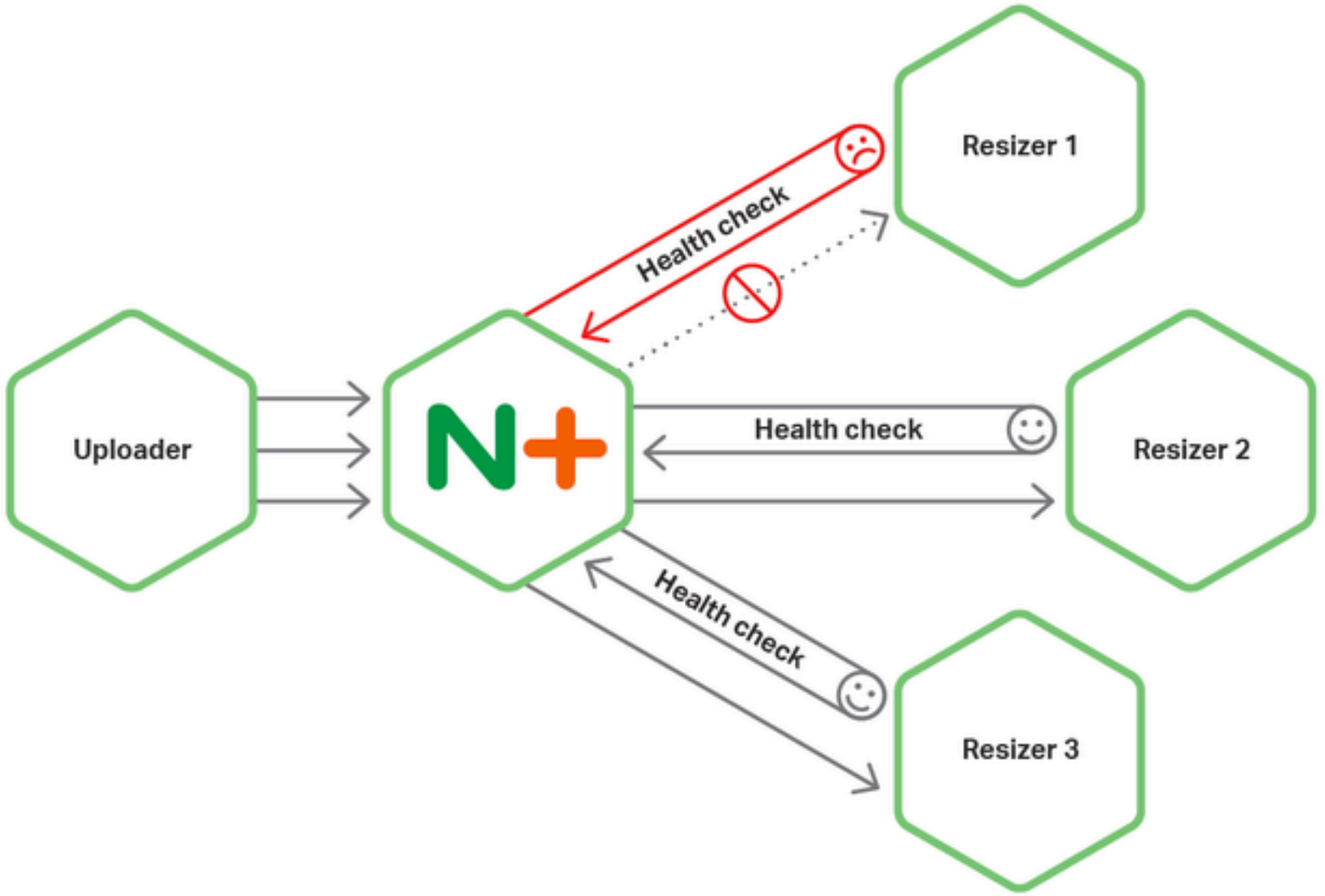
Key Benefits

1. Technology heterogeneity

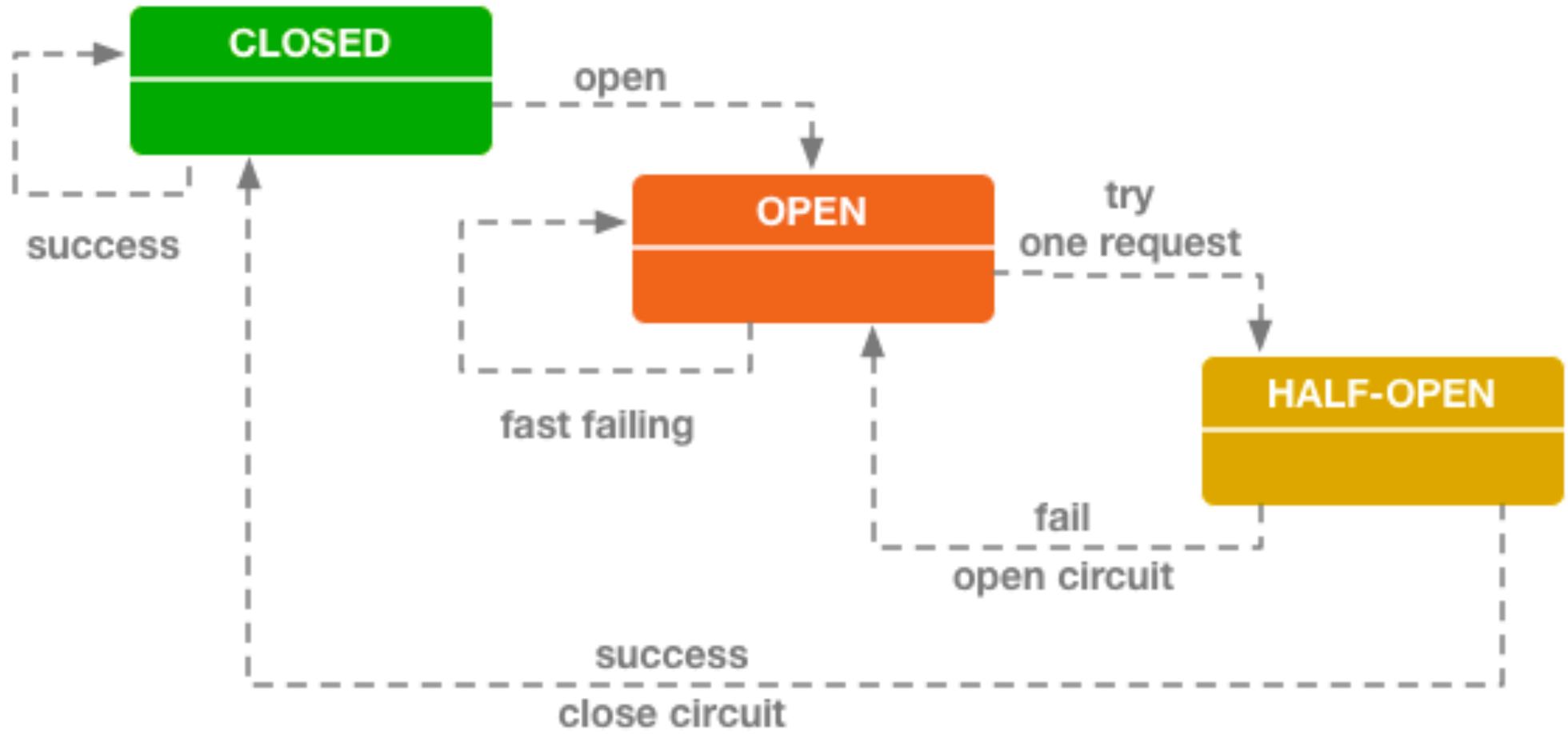
The right tool for each job



2. Resilience

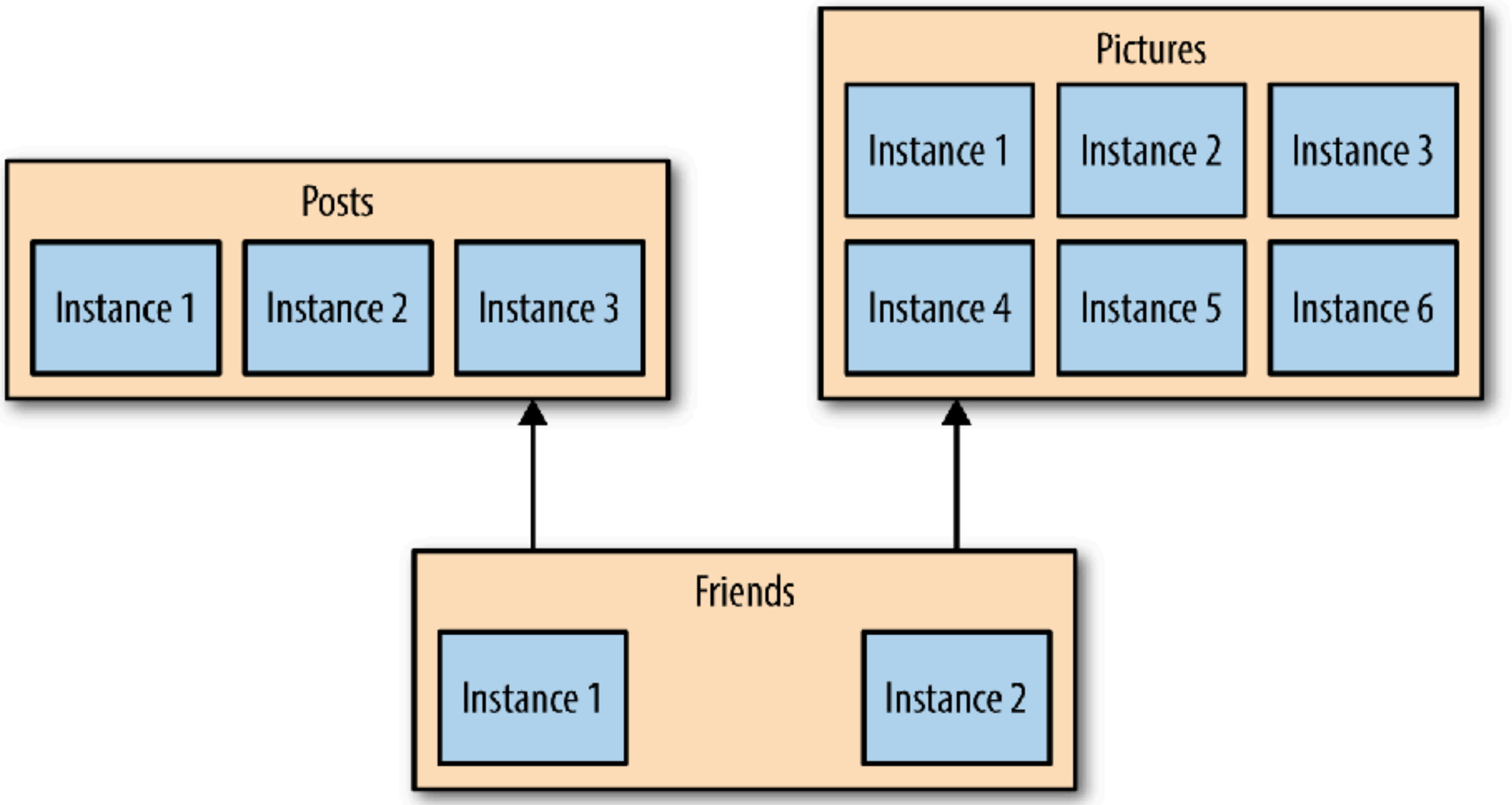


Circuit Breaker



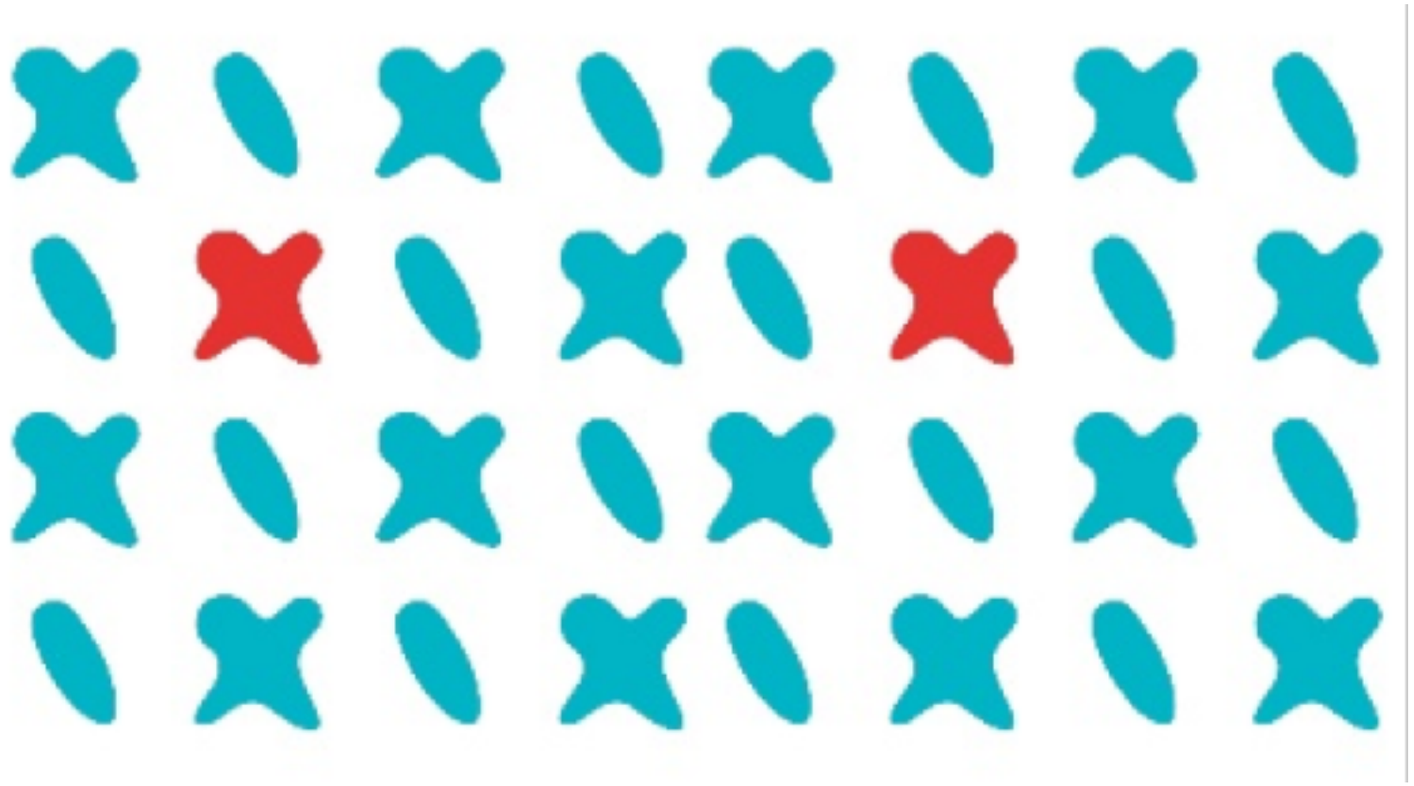
Circuit Breaker State Diagram

3. Scaling



4. Ease of deployment

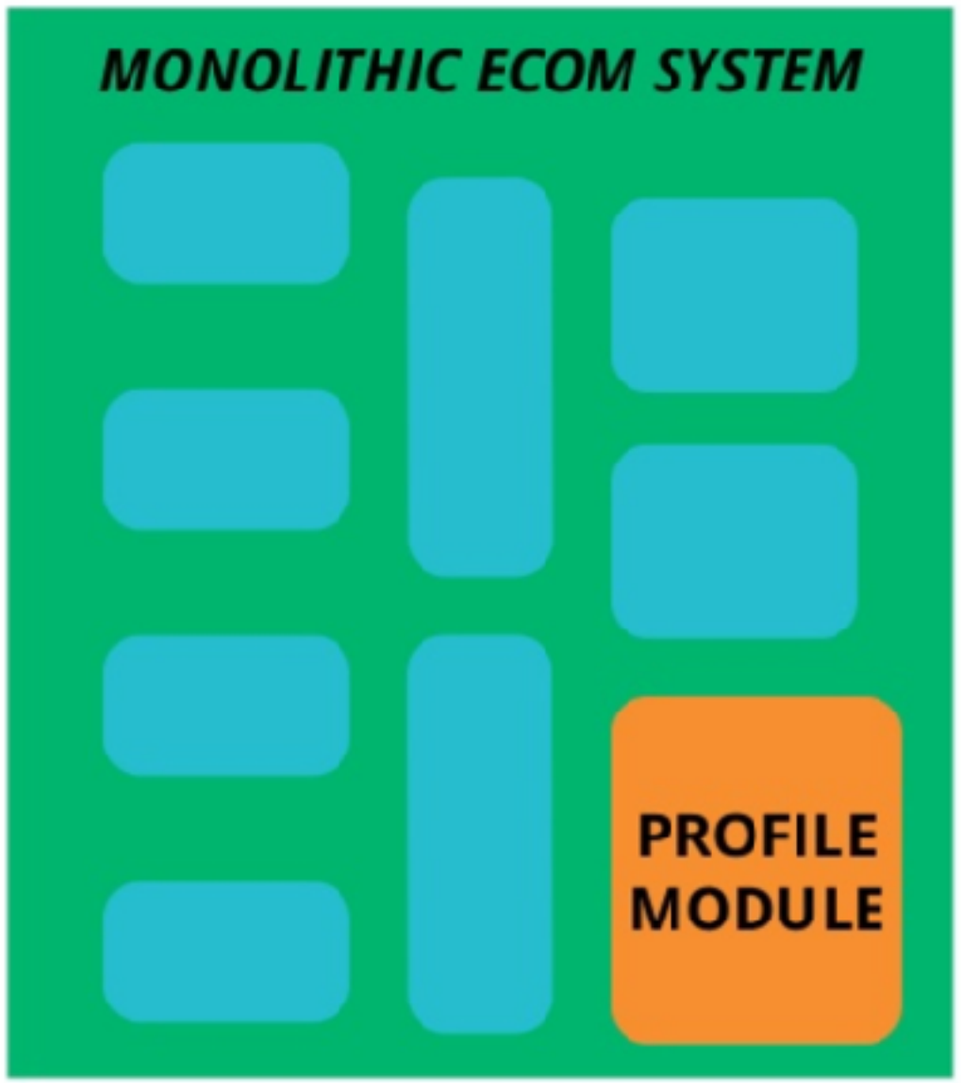
Deploys are faster, independent and problems can be isolated more easily



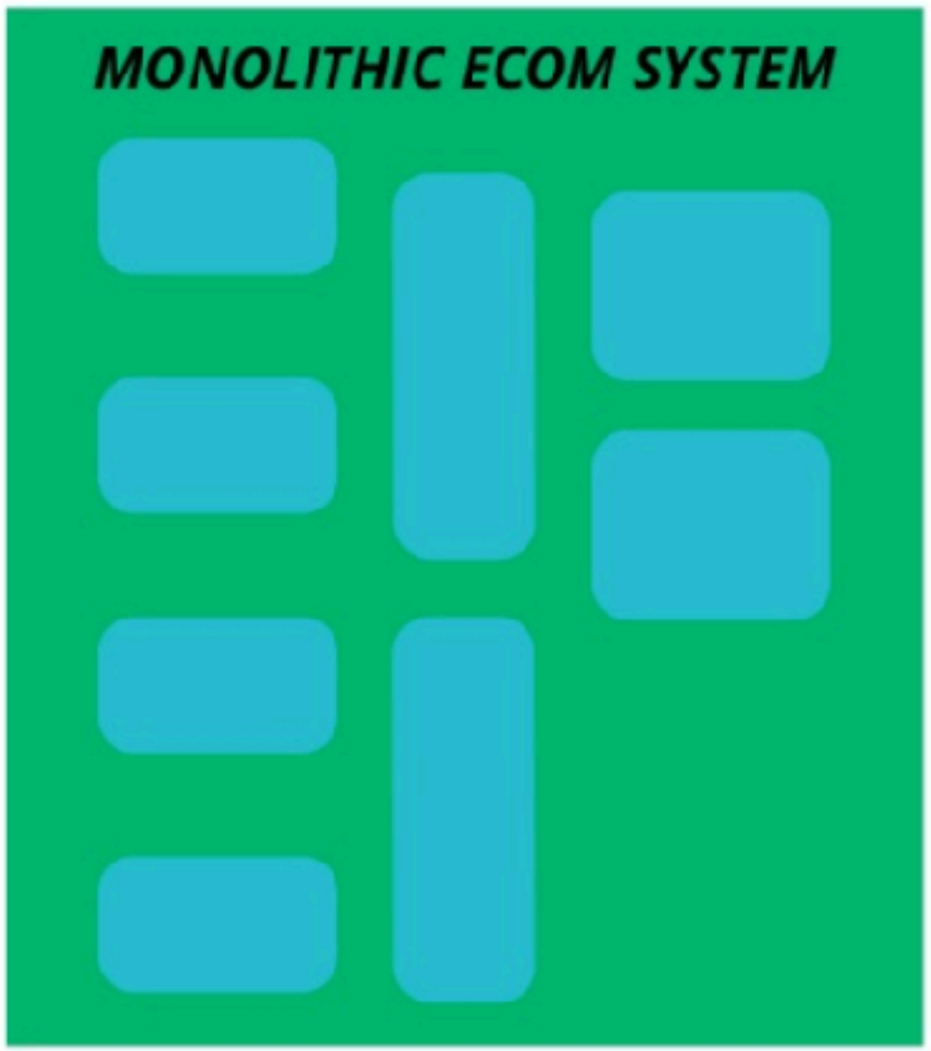
5. Organization alignment

Small teams and smaller codebases

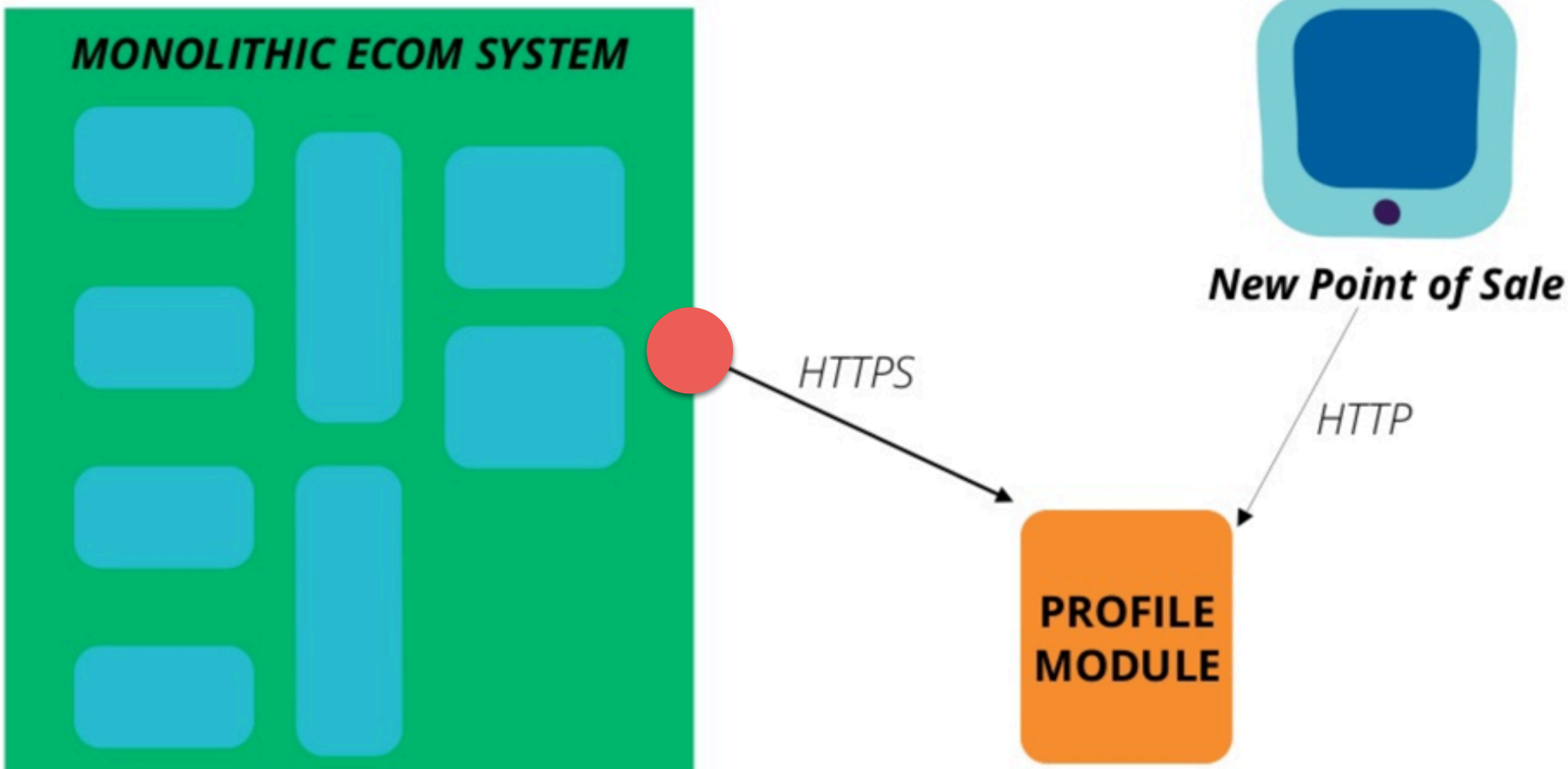
6. Composability and replaceability



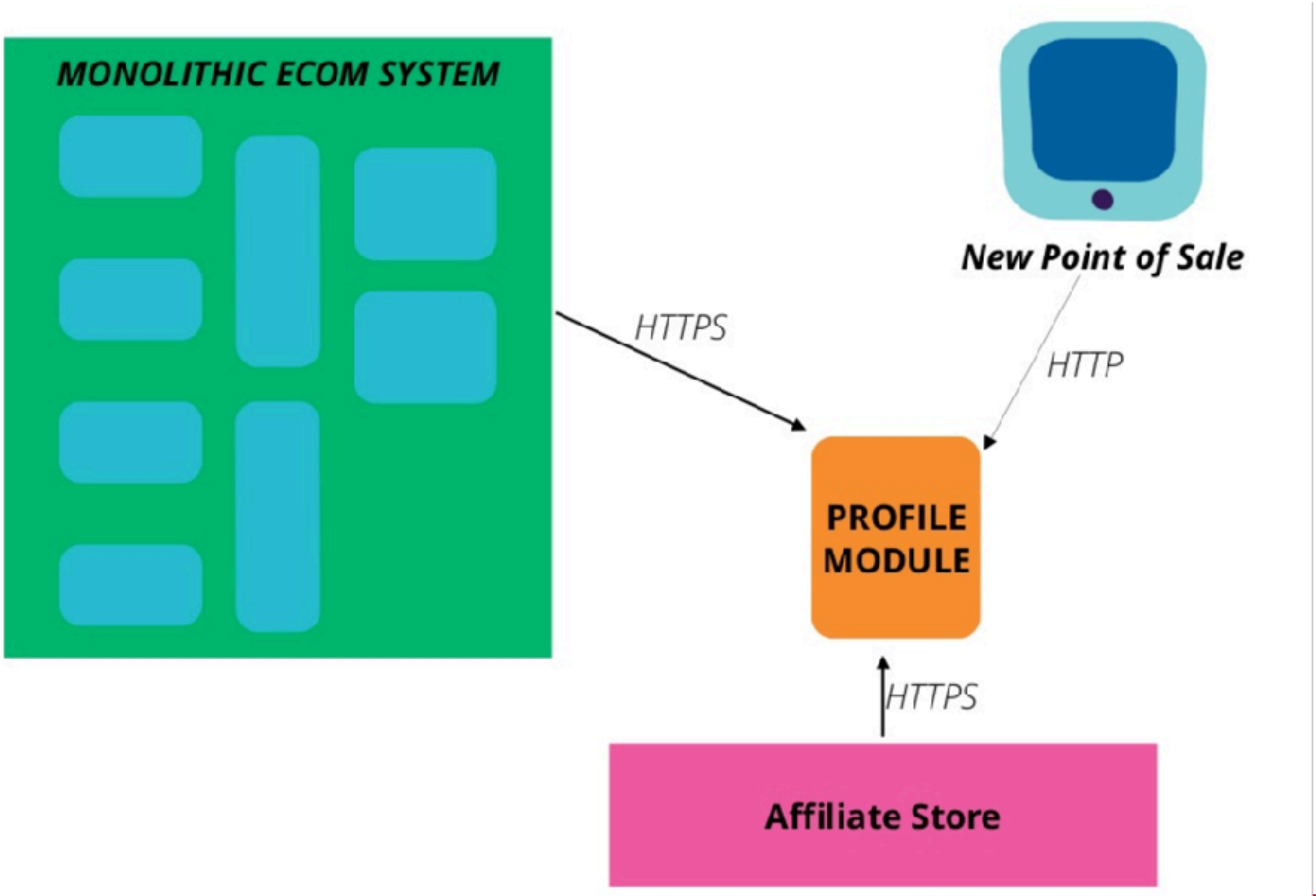
6. Composability and replaceability



6. Composability and replaceability



6. Composability and replaceability



Characteristics

1. Responsible for a single capability

Types of capabilities

Business capability
Technical capability

2. Individually deployable

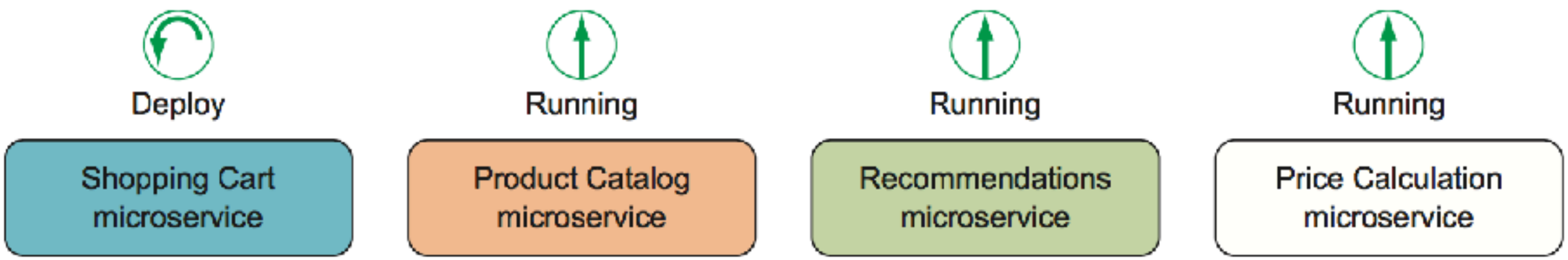


Figure 1.2 Other microservices continue to run while the Shopping Cart microservice is being deployed.

3. Consists of one or more processes

Problematic process boundary. Microservices should run in separate processes to avoid coupling.

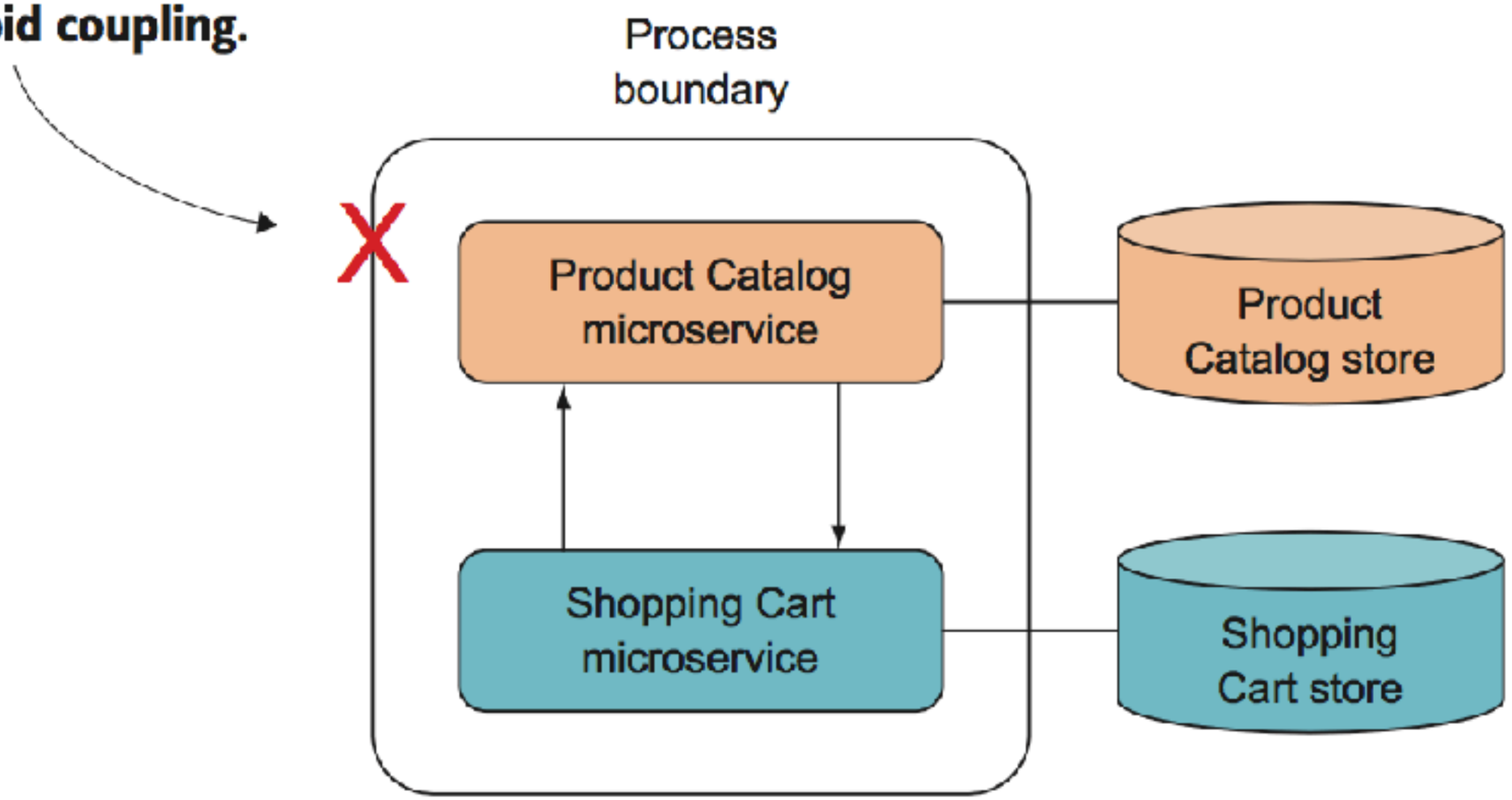


Figure 1.3 Running more than one microservice within a process leads to high coupling.

4. Own data store

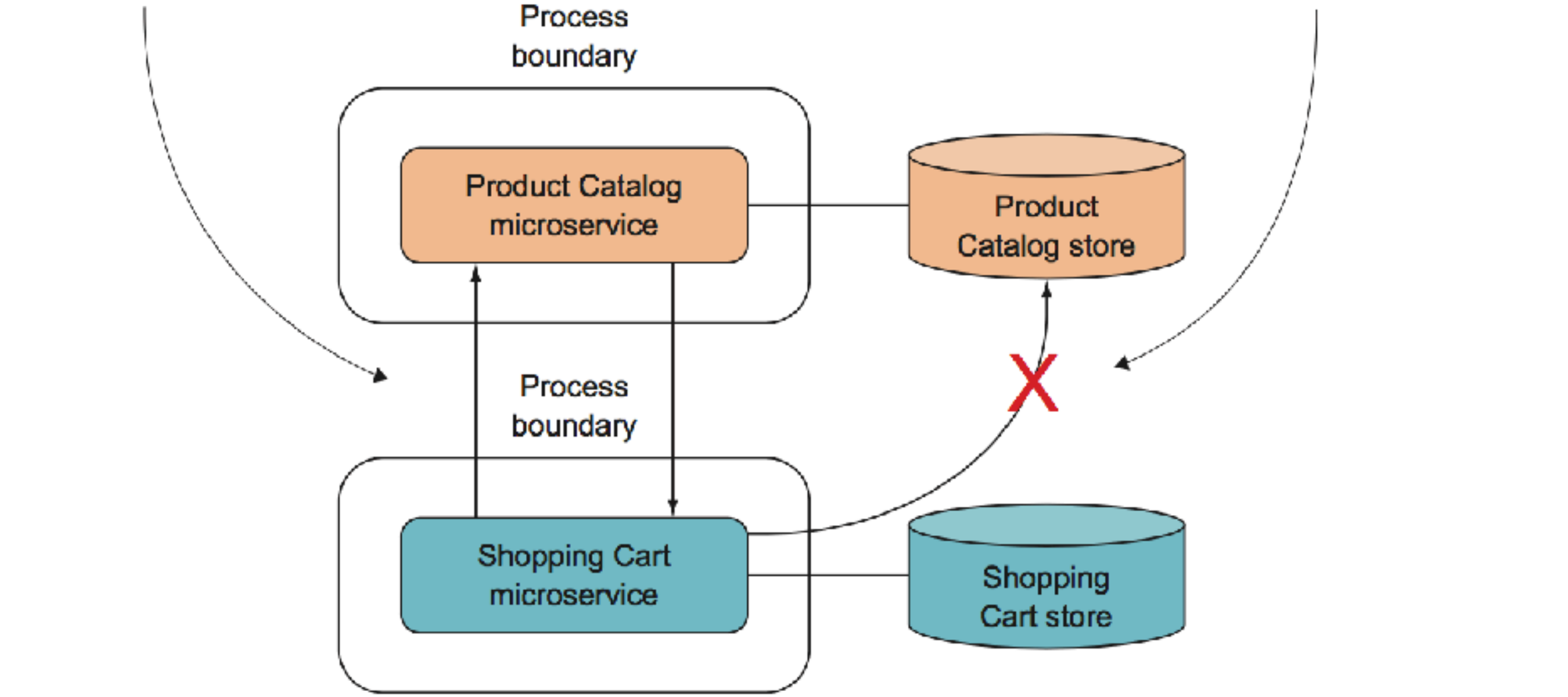
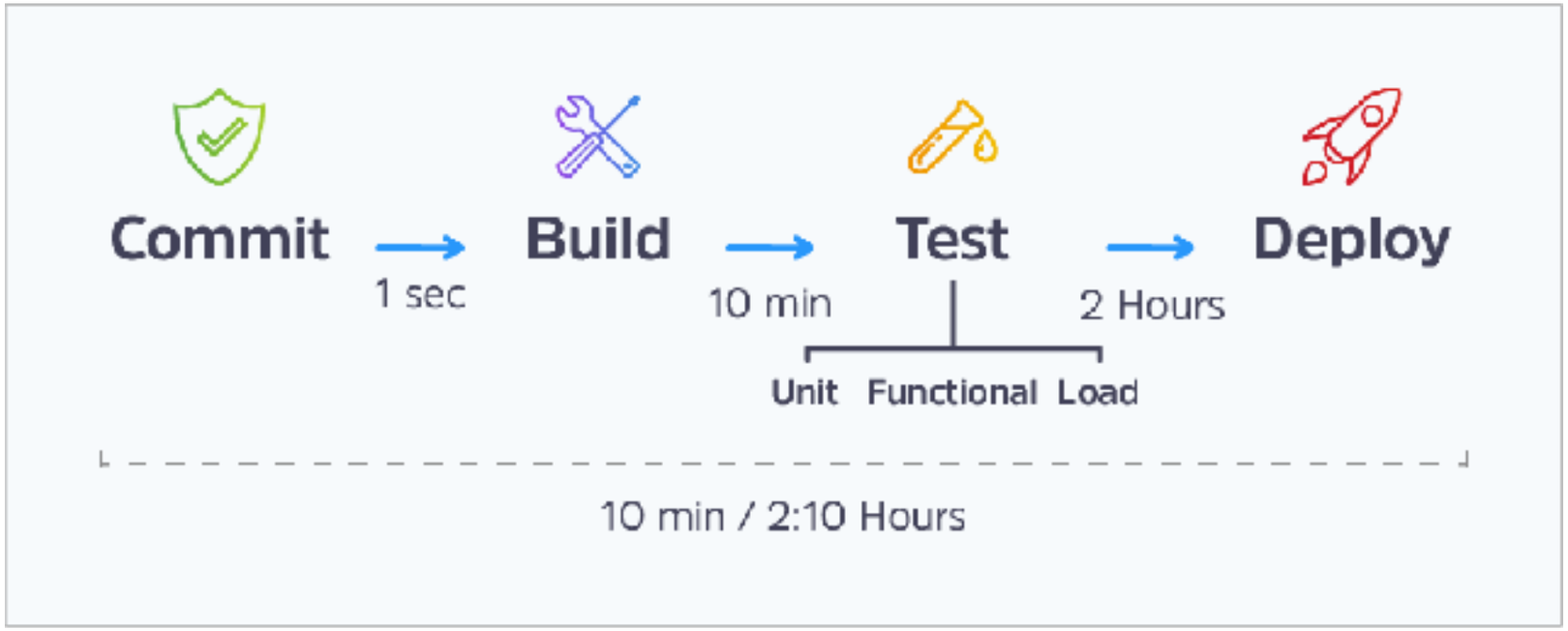


Figure 1.4 One microservice can't access another's data store.

5. Small team can maintain

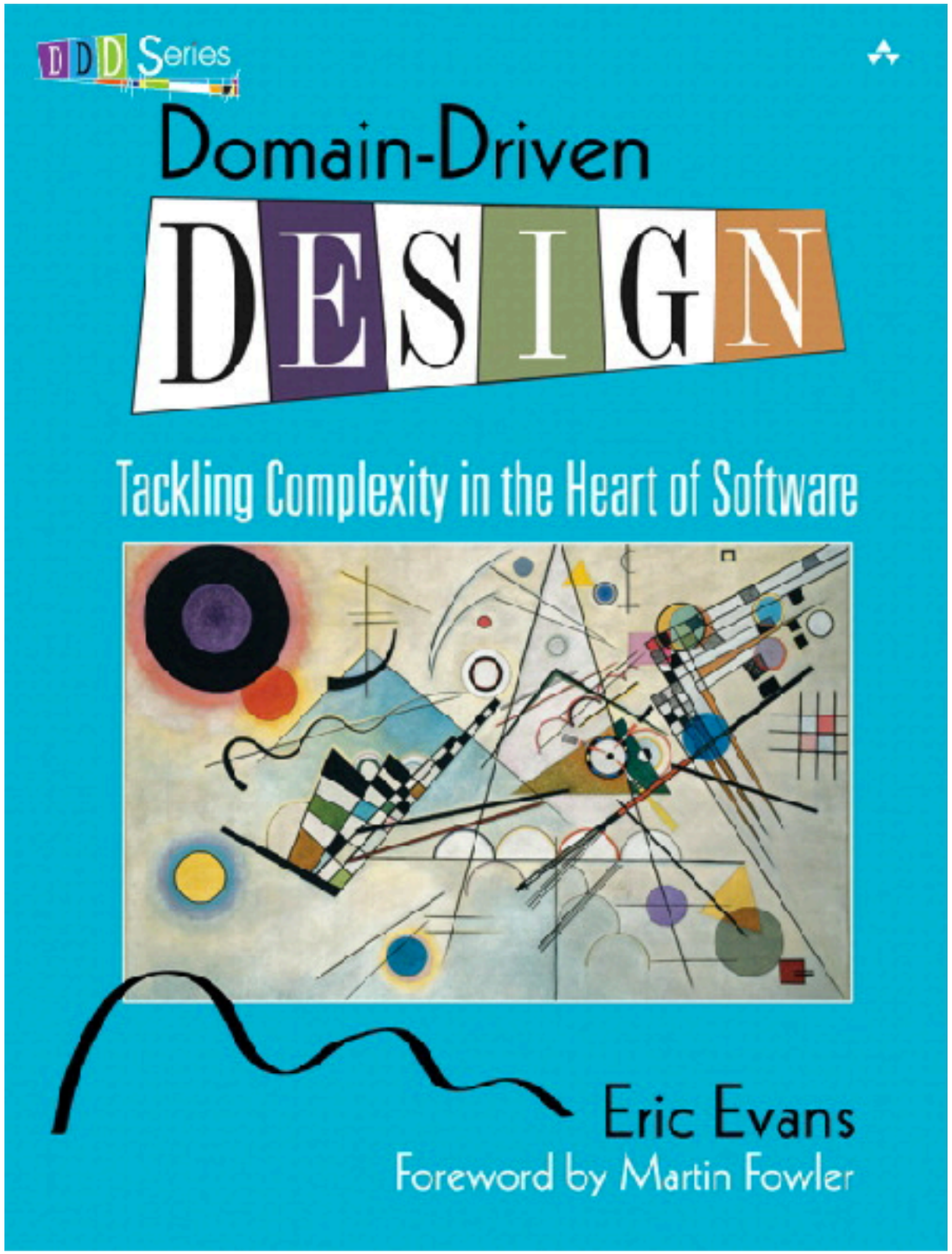


6. Replaceable

Microservices must own its domain **data** and **logic** under **autonomous** lifecycle

Challenges with Microservices ?

1. How to define the boundaries of each microservices ?



Premature splitting is the root of all **evil**.

Every time you make the decision to split out a new microservice, there's a **risk** of ending up with a bloated app.

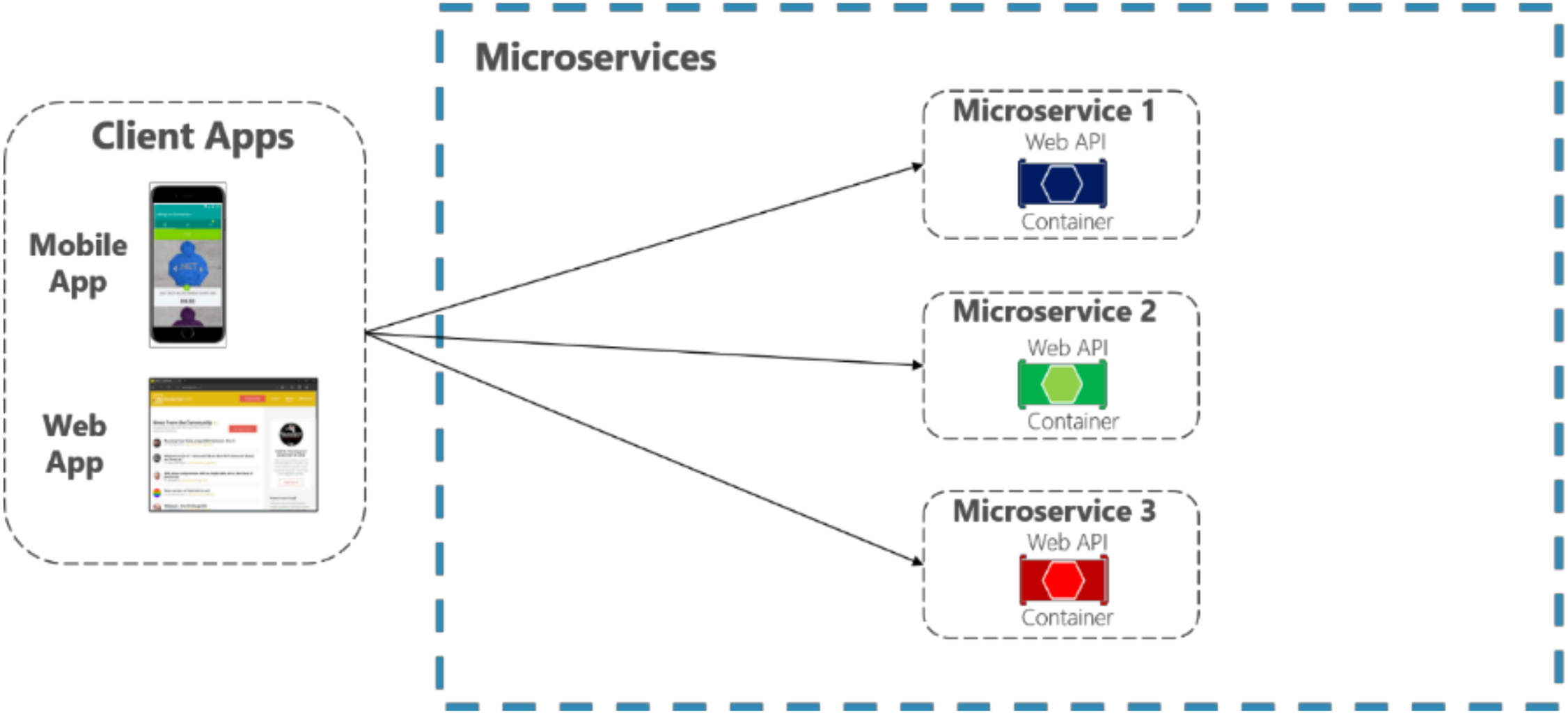
2. How to create queries that retrieve data from several microservices ?

Popular solutions

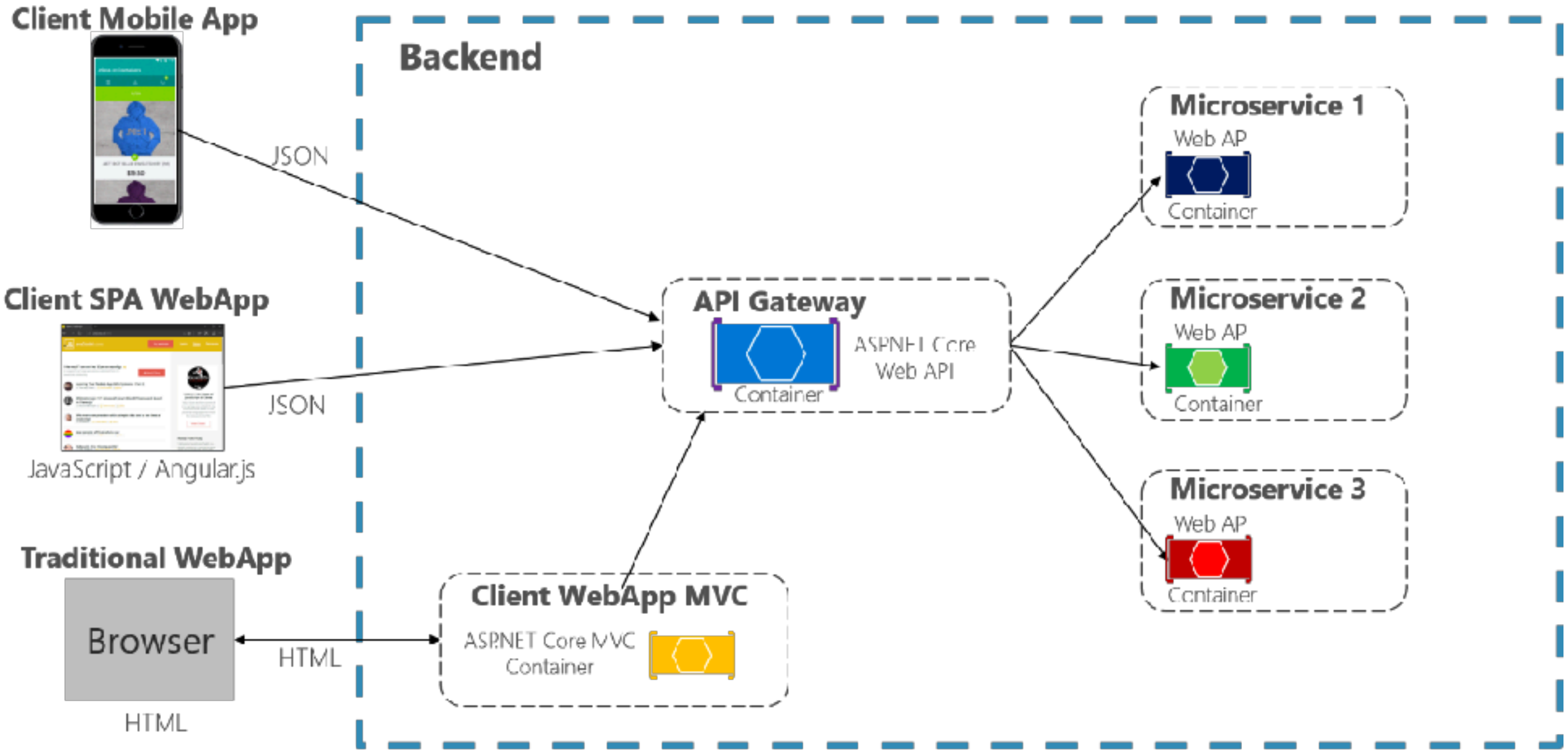
API Gateway
CQRS with query/read tables
Cold data in centralize database

Direct Client-To-Microservice communication

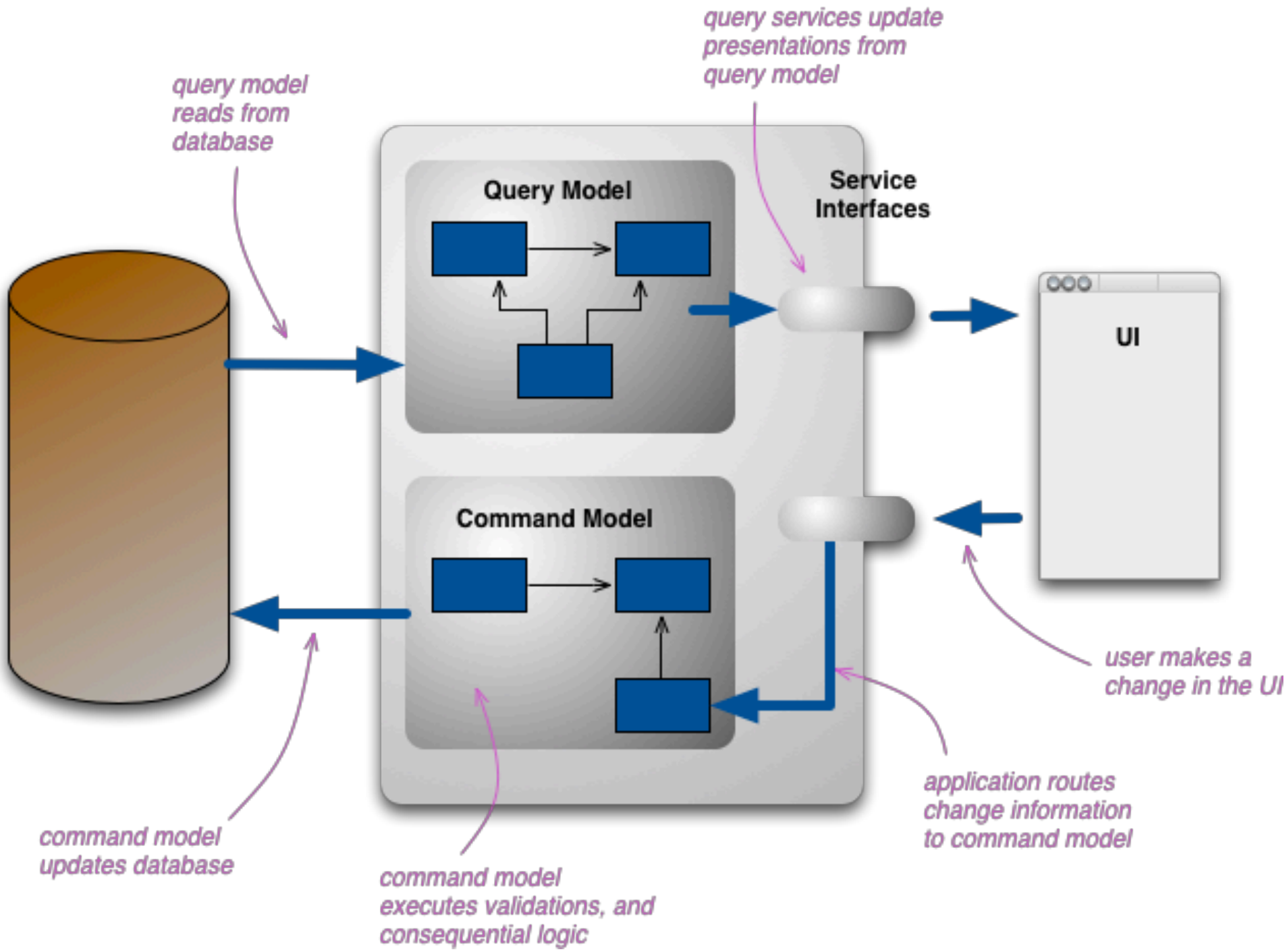
Architecture



Using the **API Gateway Service**

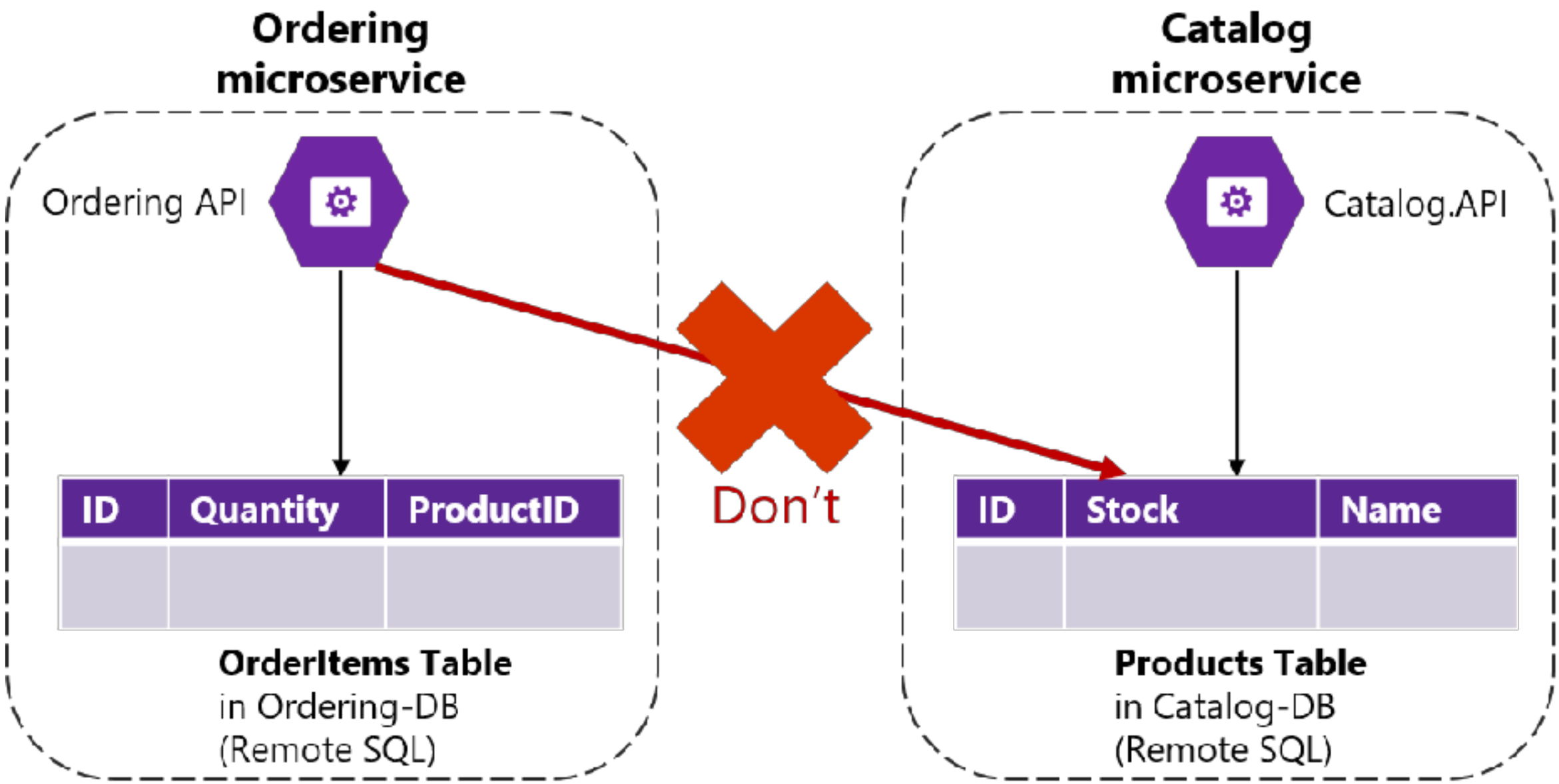


Command Query Responsibility Segregation



https://martinfowler.com/bliki/CQRS.html

3. How to achieve consistency across multiple microservices ?



Databases are private per microservice

CAP Theorem

Consistency

Availability

Partition Tolerance

http://robertgreiner.com/2014/08/cap-theorem-revisited/

4. How to design communication across microservices boundaries ?

Protocols

HTTP and REST
AMQP (Advance Message Queuing Protocol)
Messaging

Communication

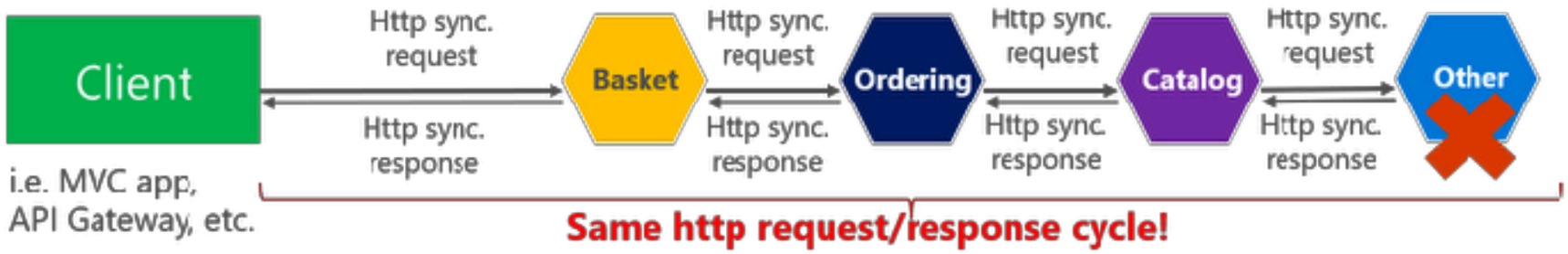
Request-Response model
Observer model

Communication

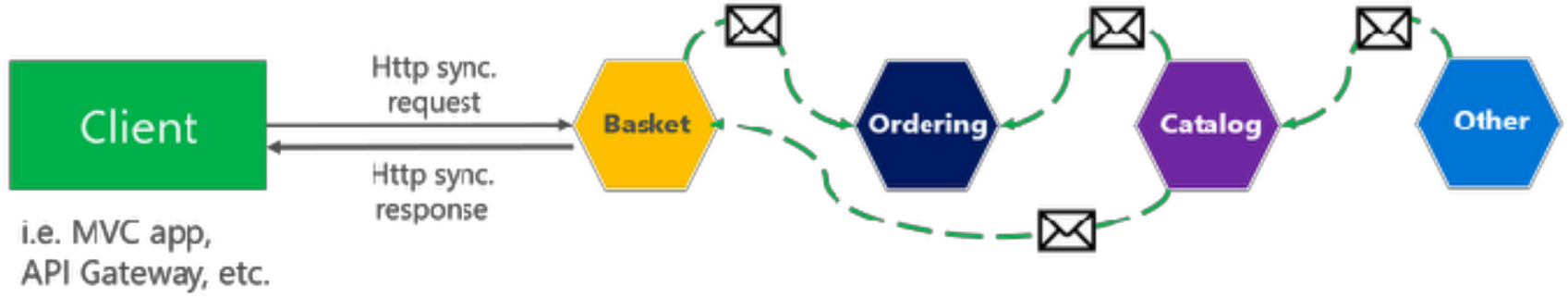
Synchronous vs. async communication across microservices

Anti-pattern

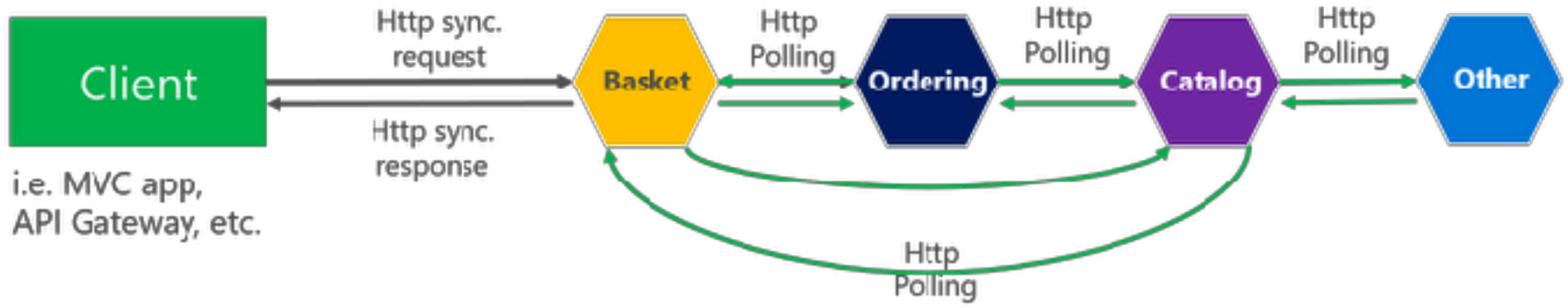
Synchronous
all req./resp. cycle



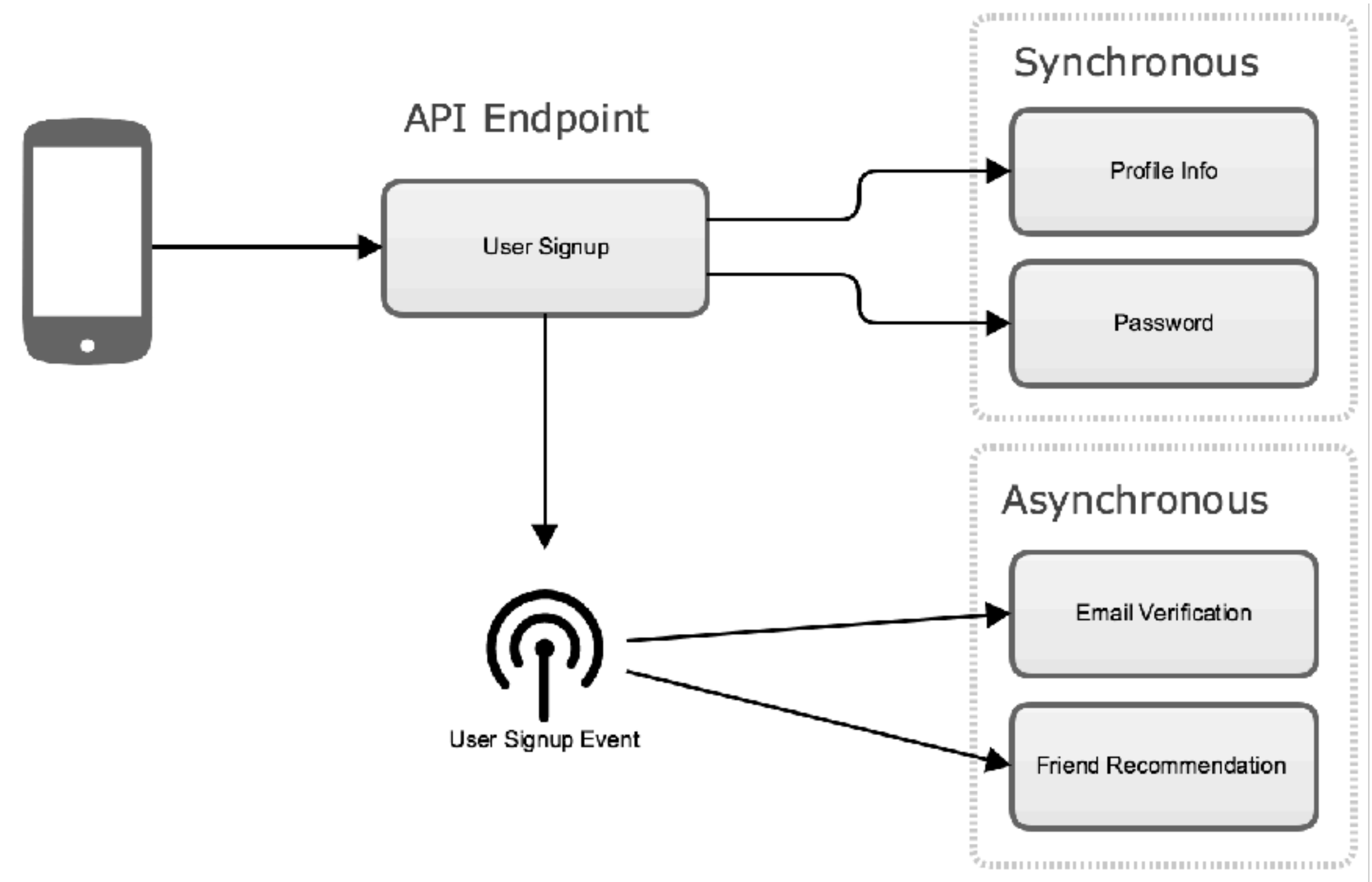
Asynchronous
Comm. across
internal microservices
(EventBus: i.e. **AMPQ**)



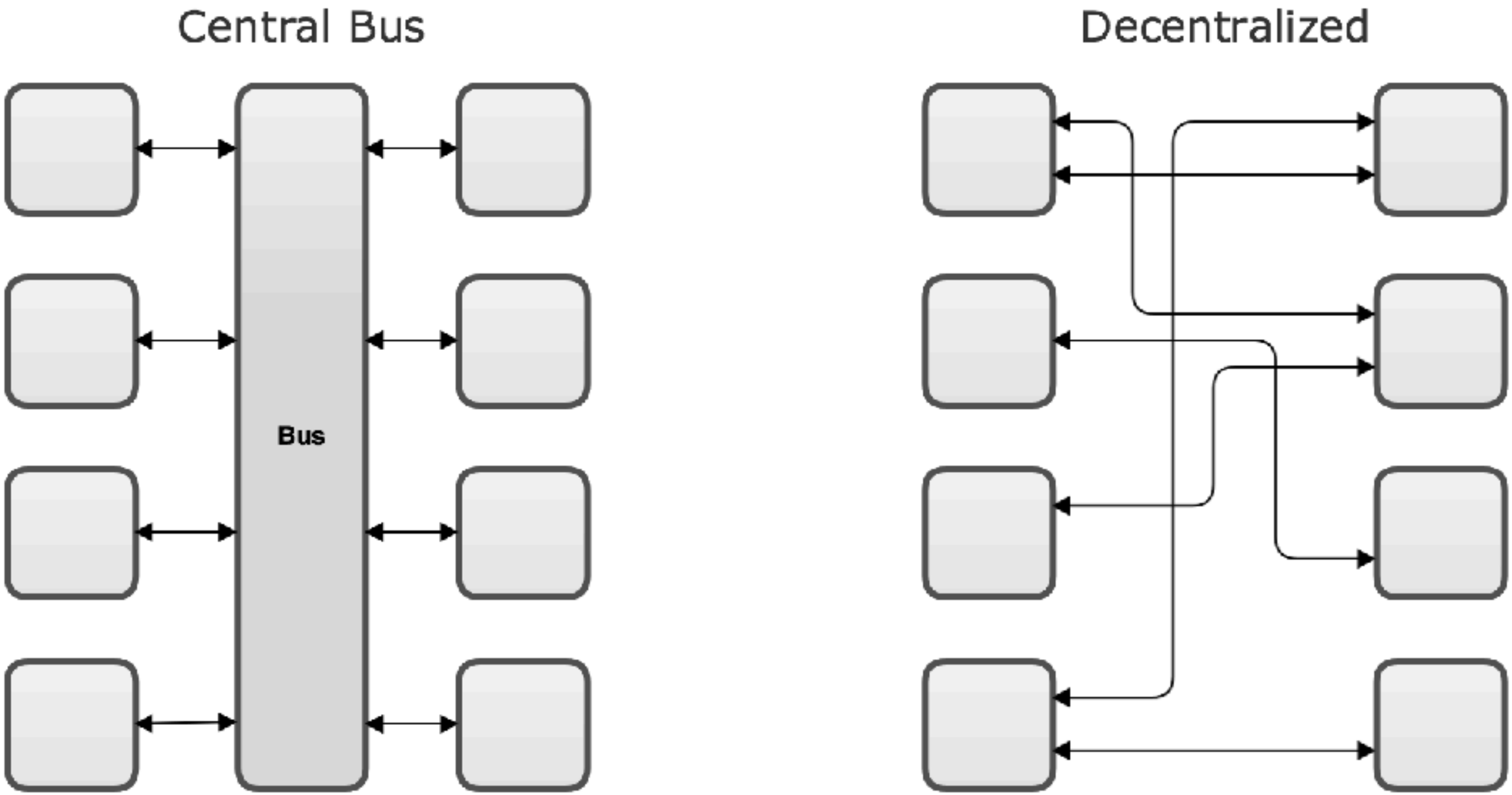
"Asynchronous"
Comm. across
internal microservices
(Polling: **Http**)



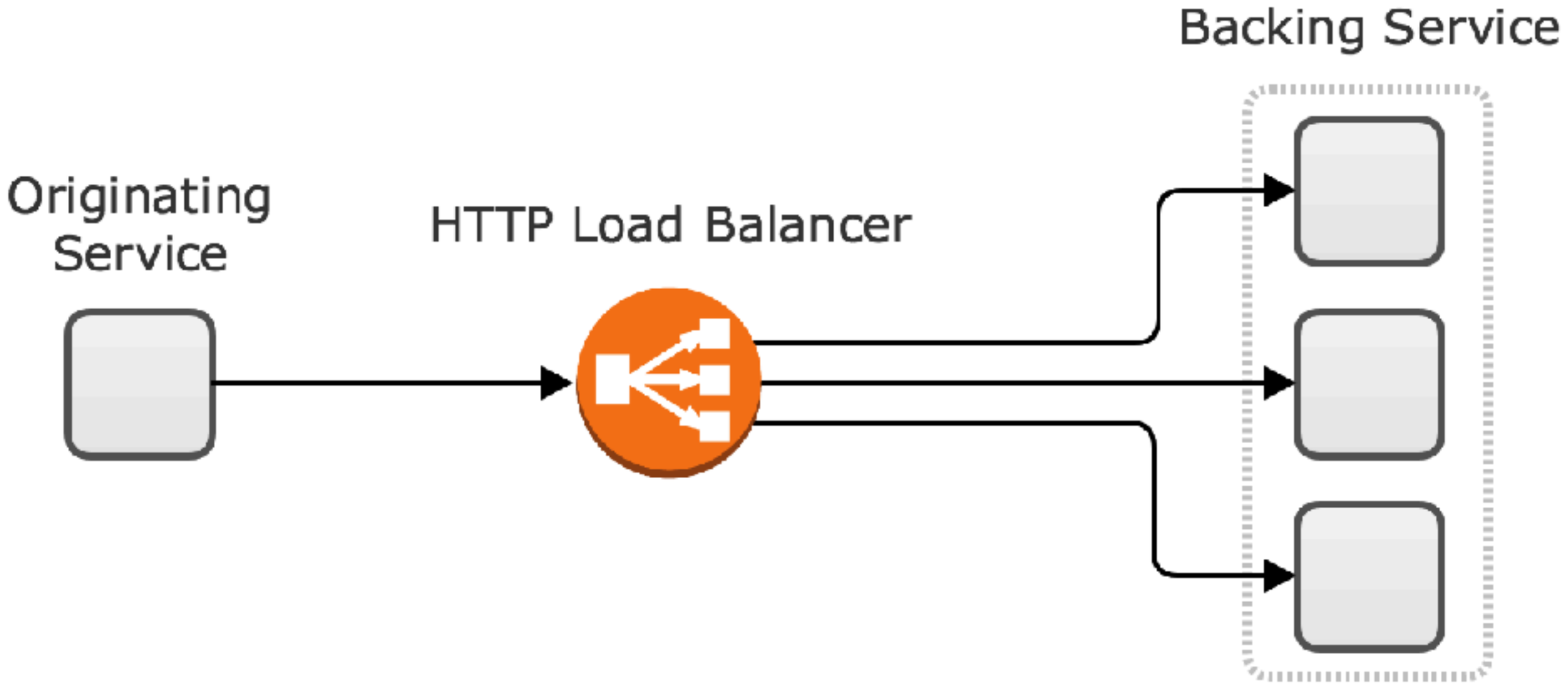
Communication



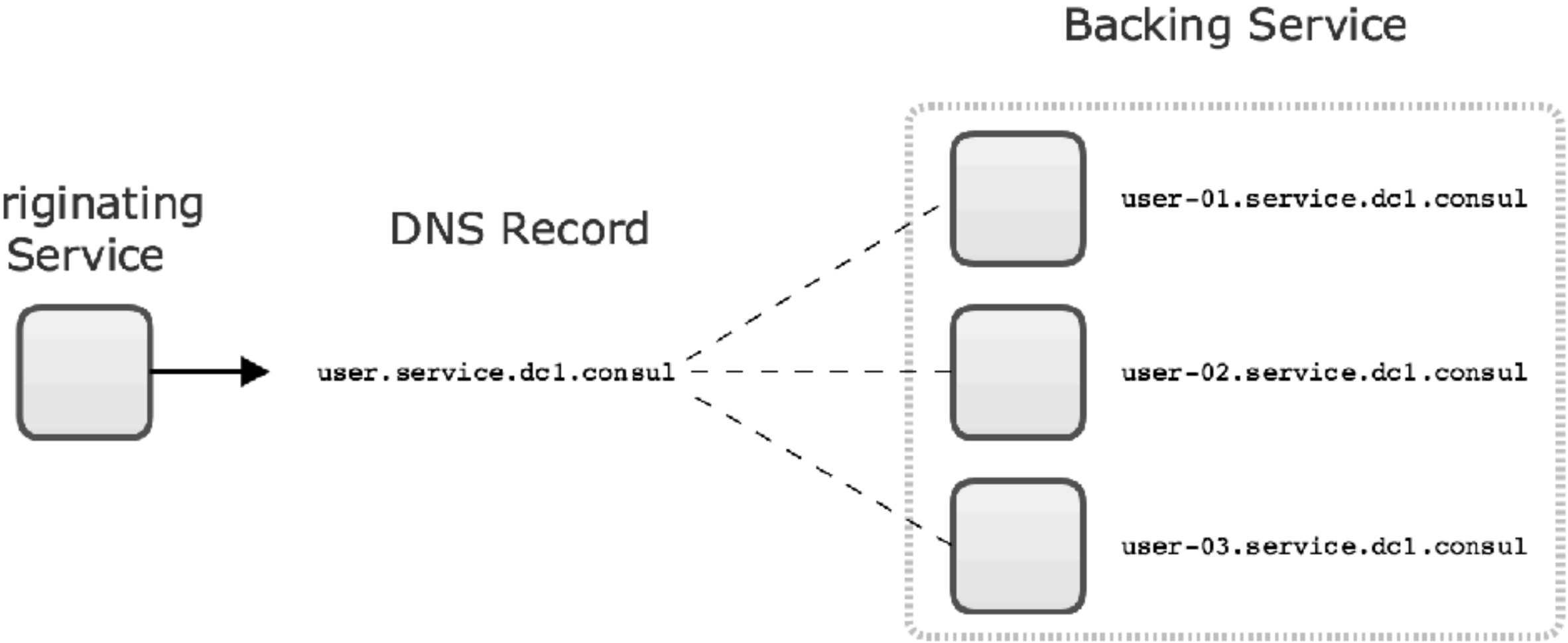
Anti-pattern :: centralize bus service



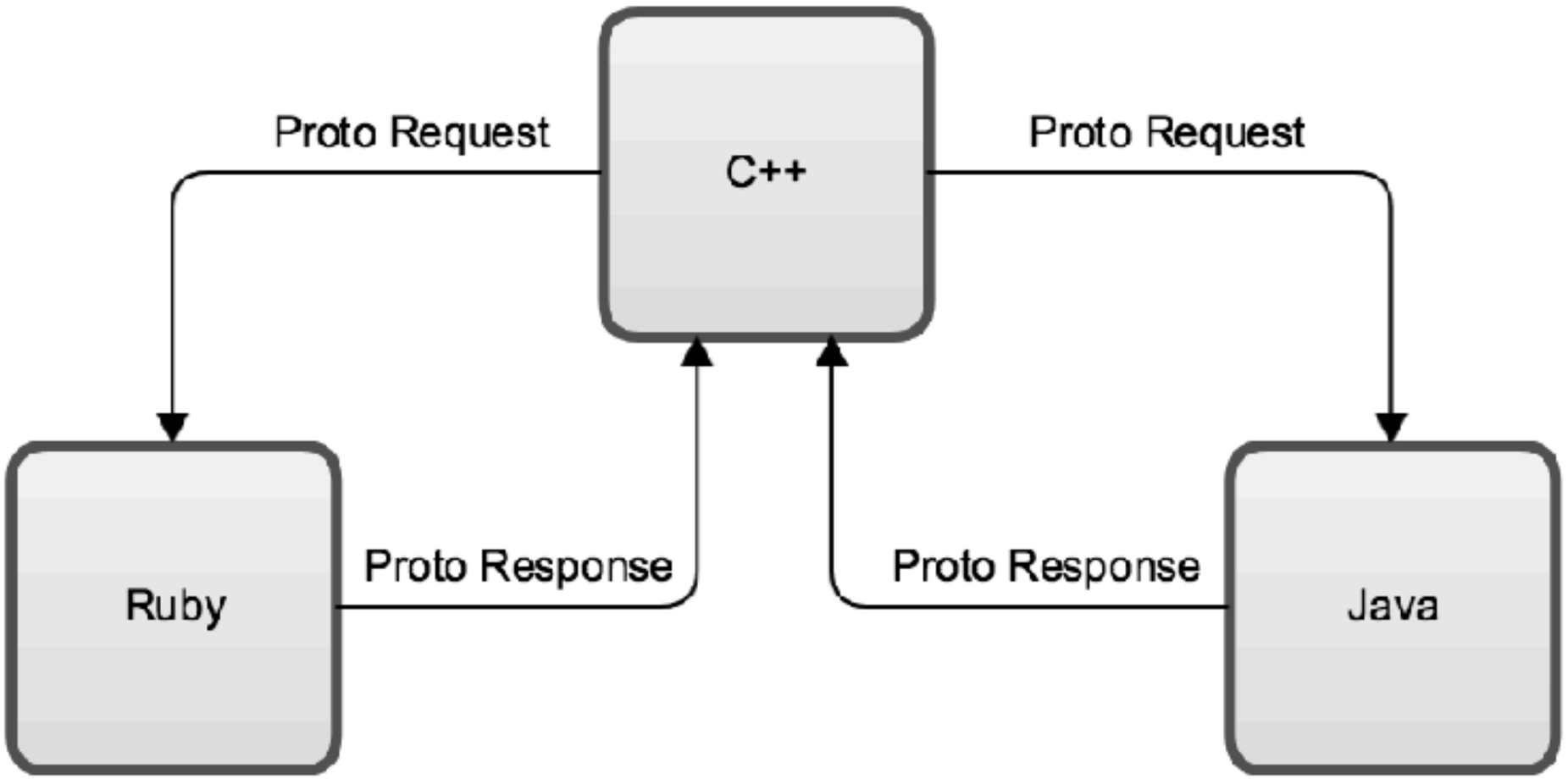
Request-response model



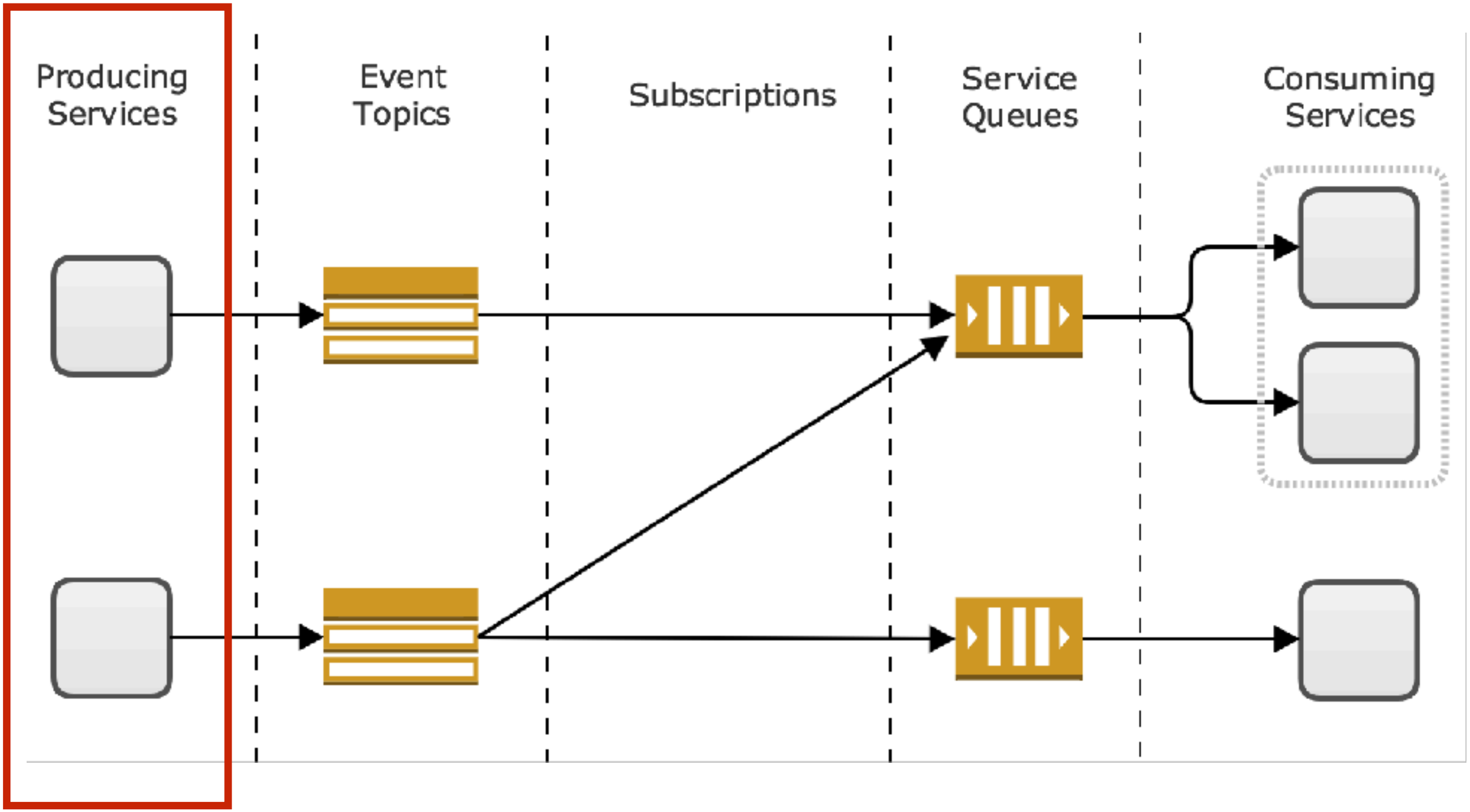
Request-response model



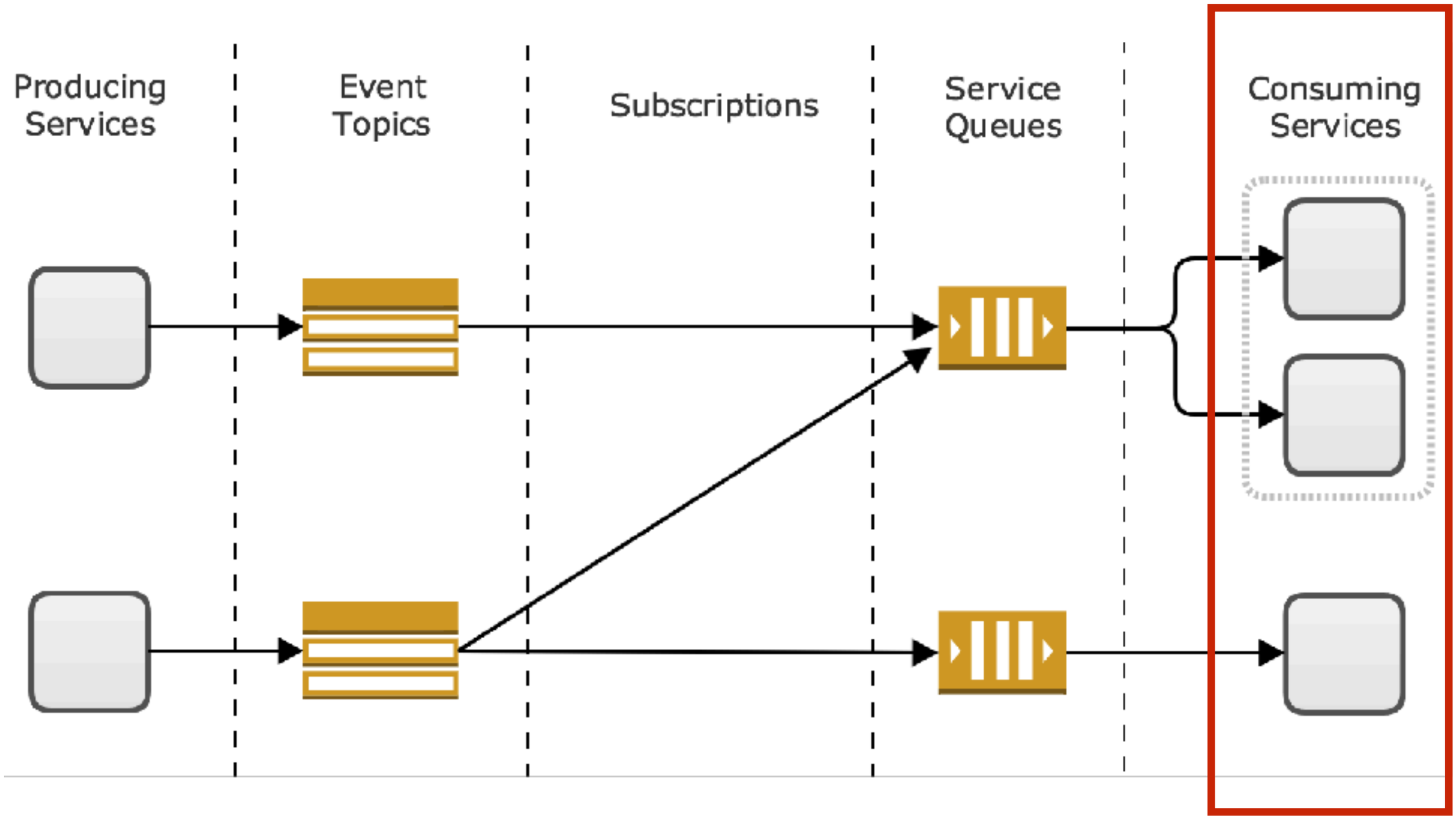
Request-response model



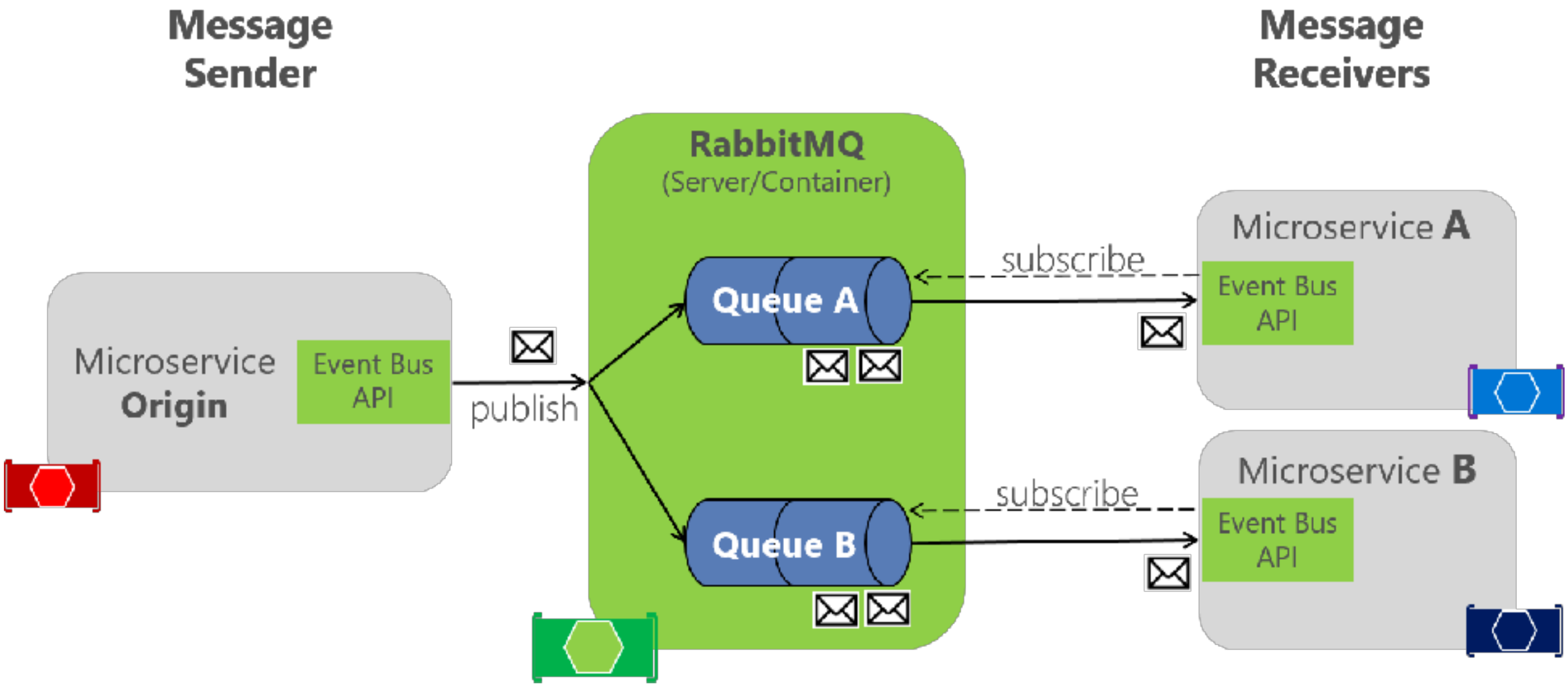
Observer model



Observer model

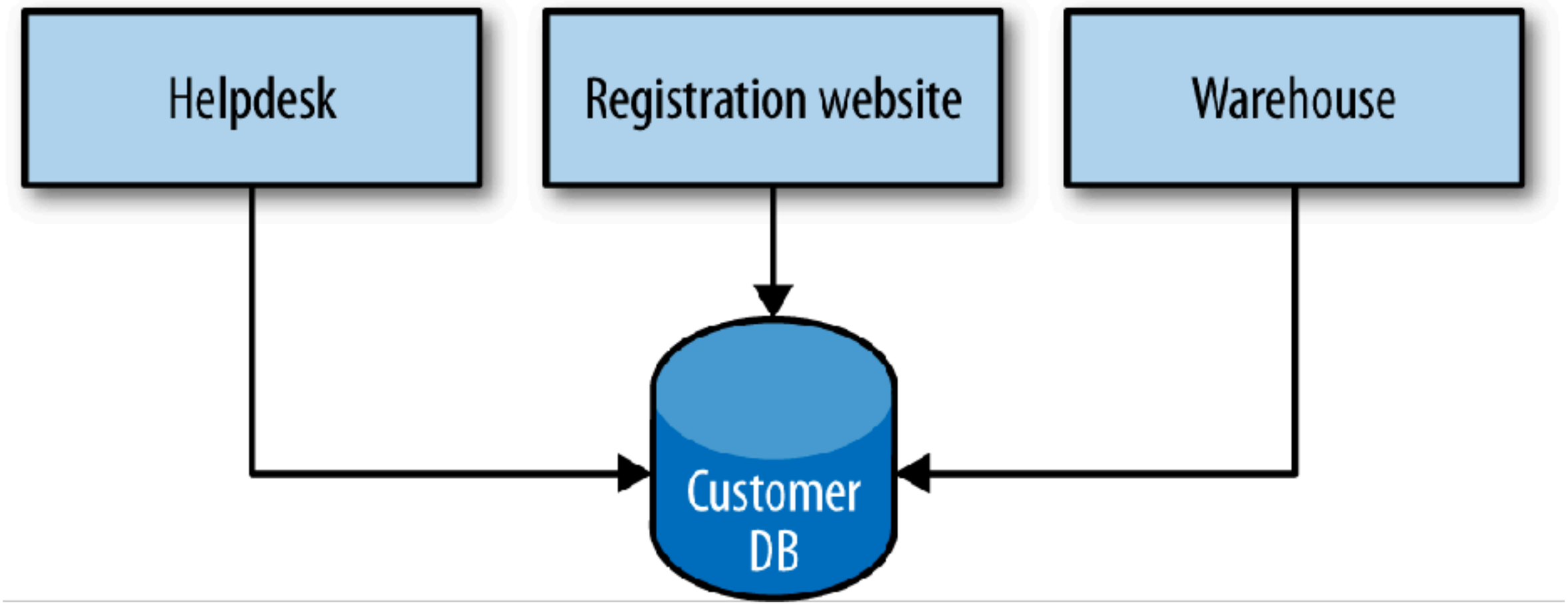


Observer model

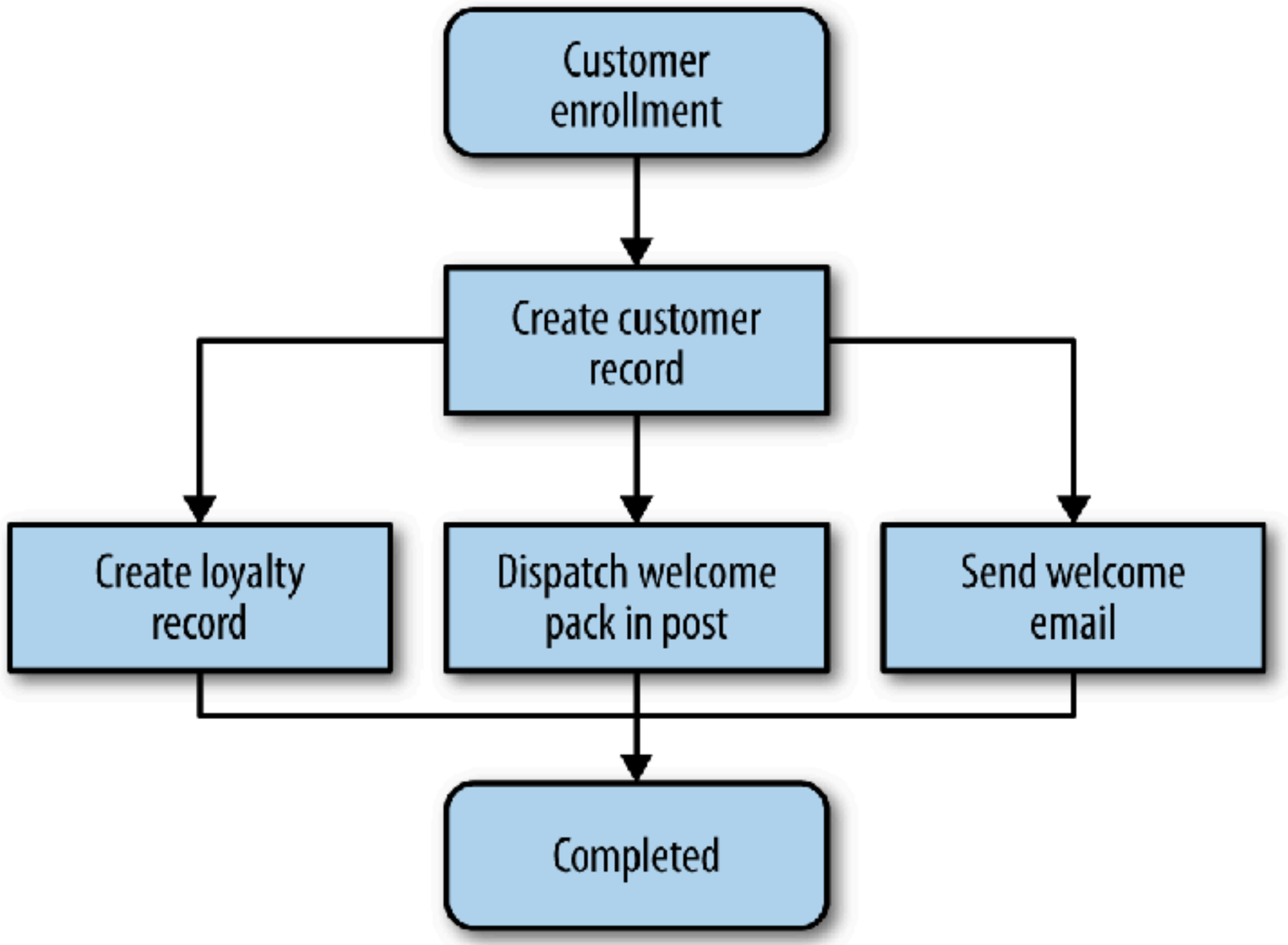


Services Integration

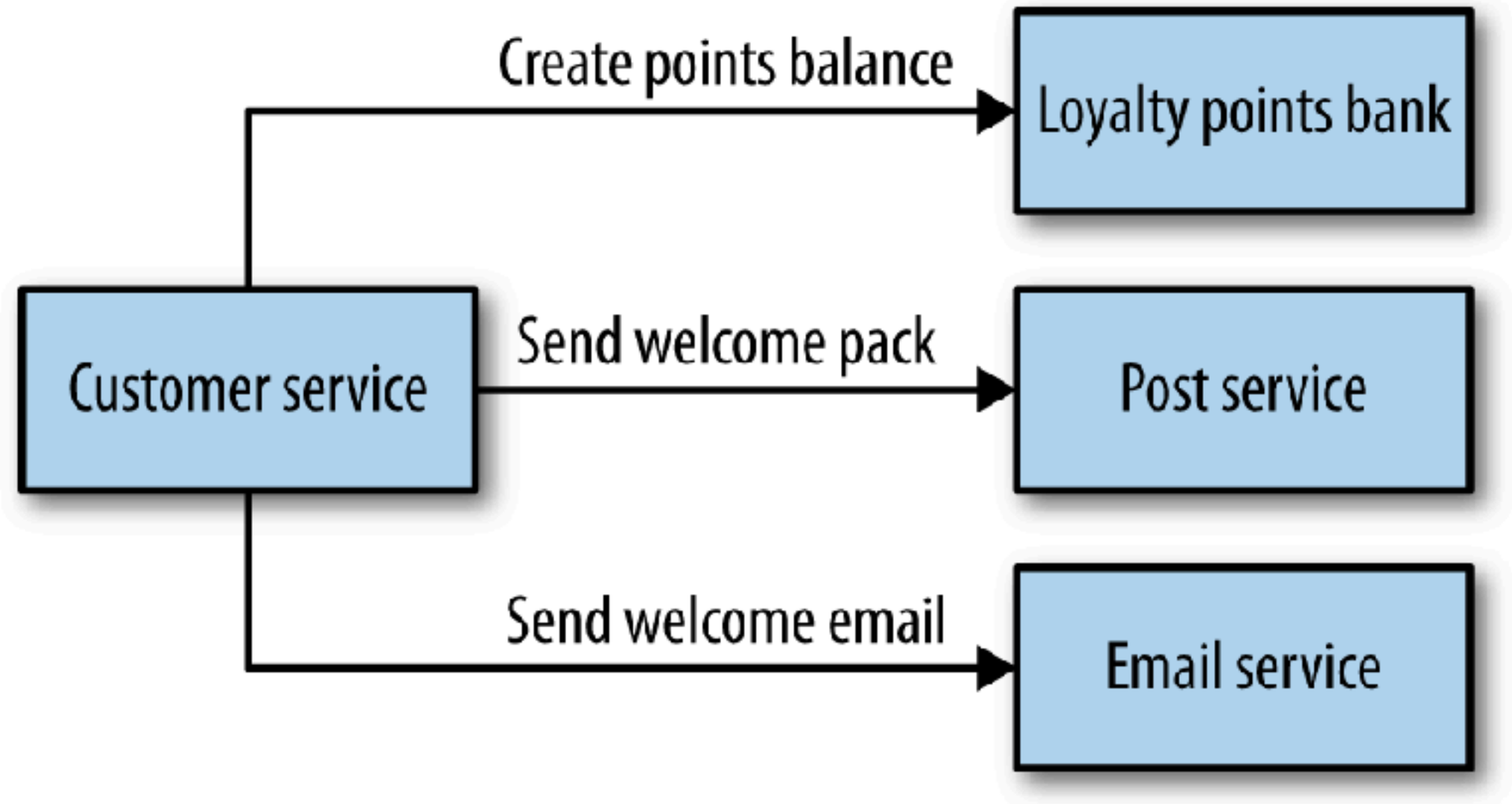
Shared database



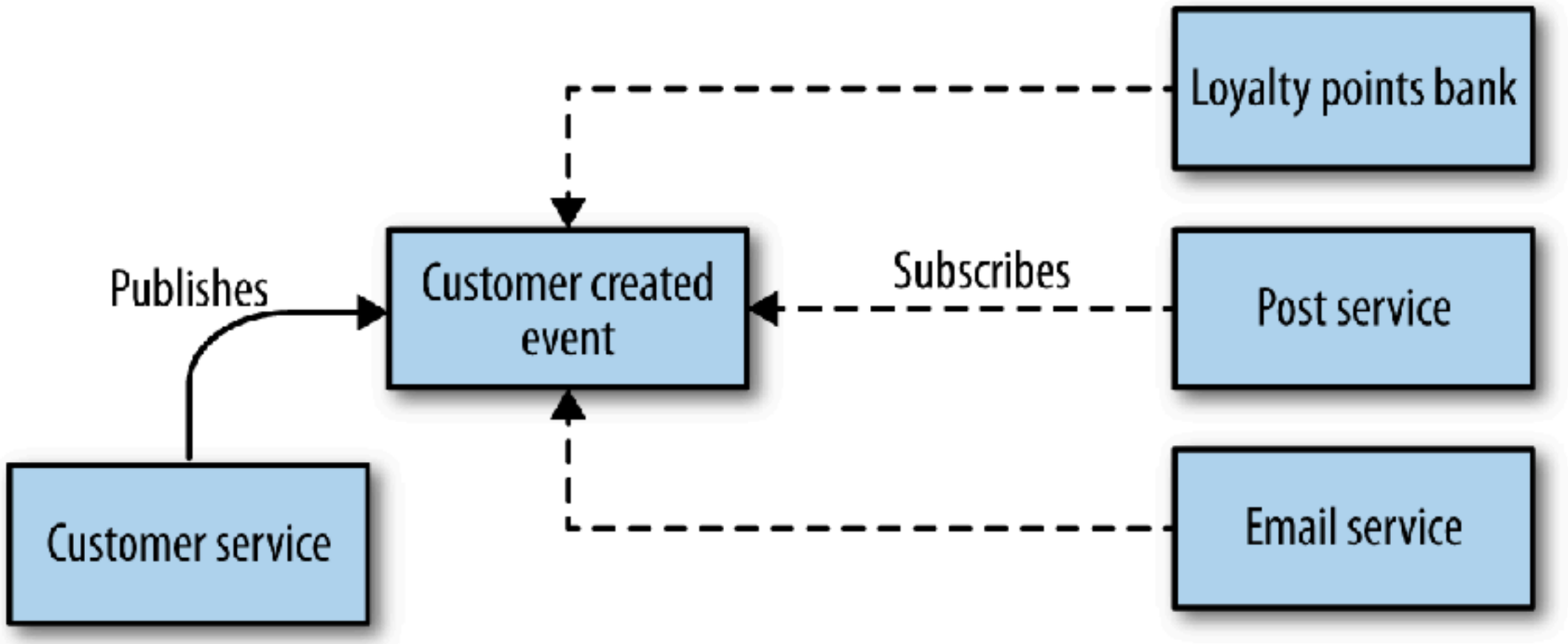
Orchestration vs Choreography



Orchestration

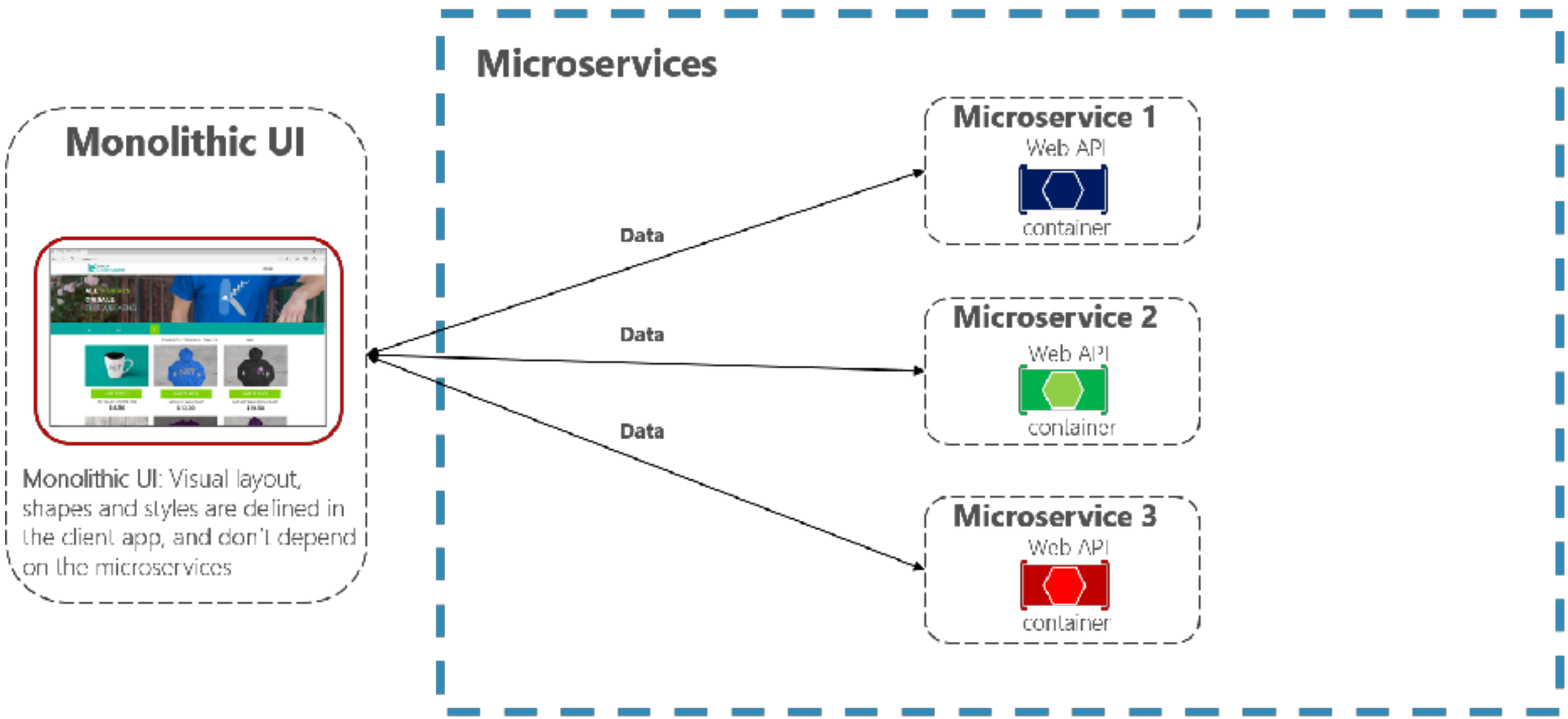


Choreography

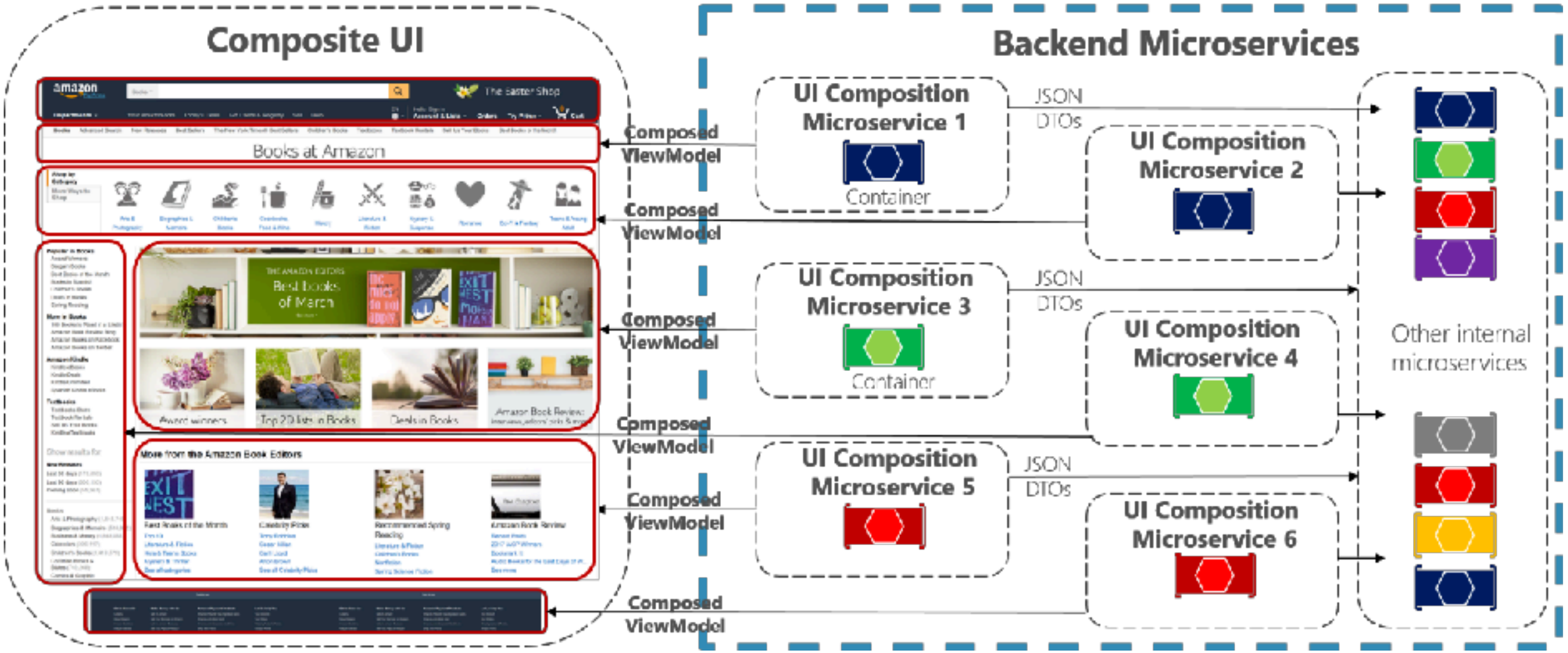


Integrate with User Interface

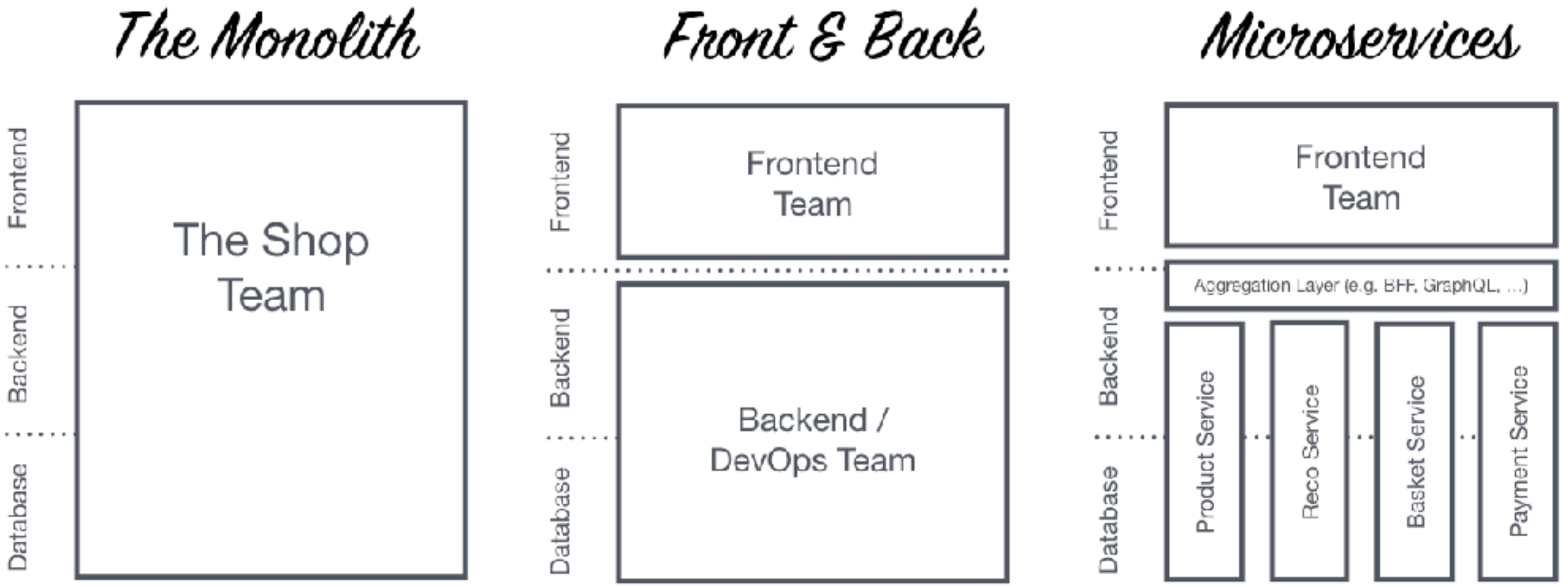
Monolithic UI consuming microservices



Composite UI *generated* by microservices

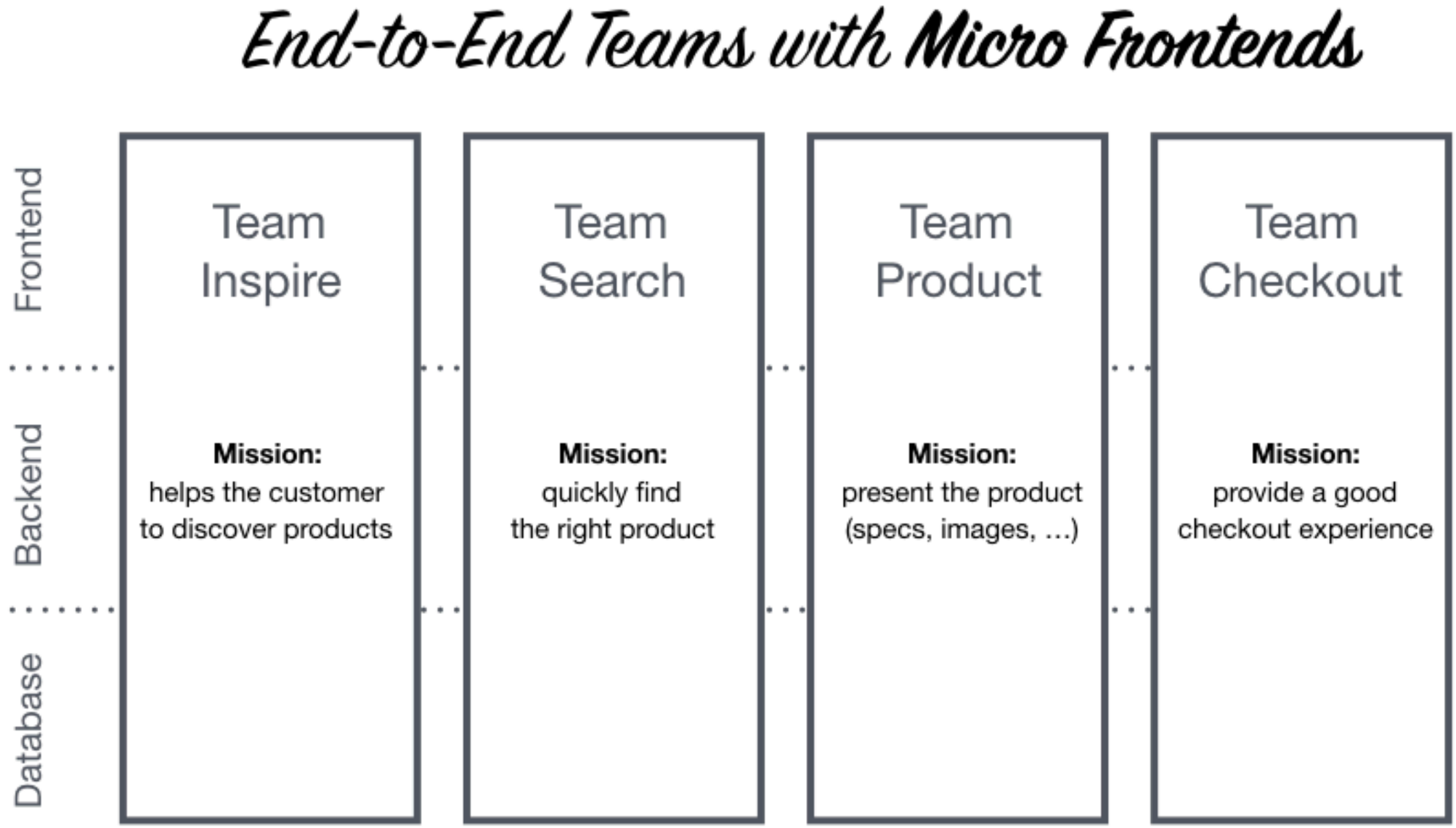


Monolith frontend

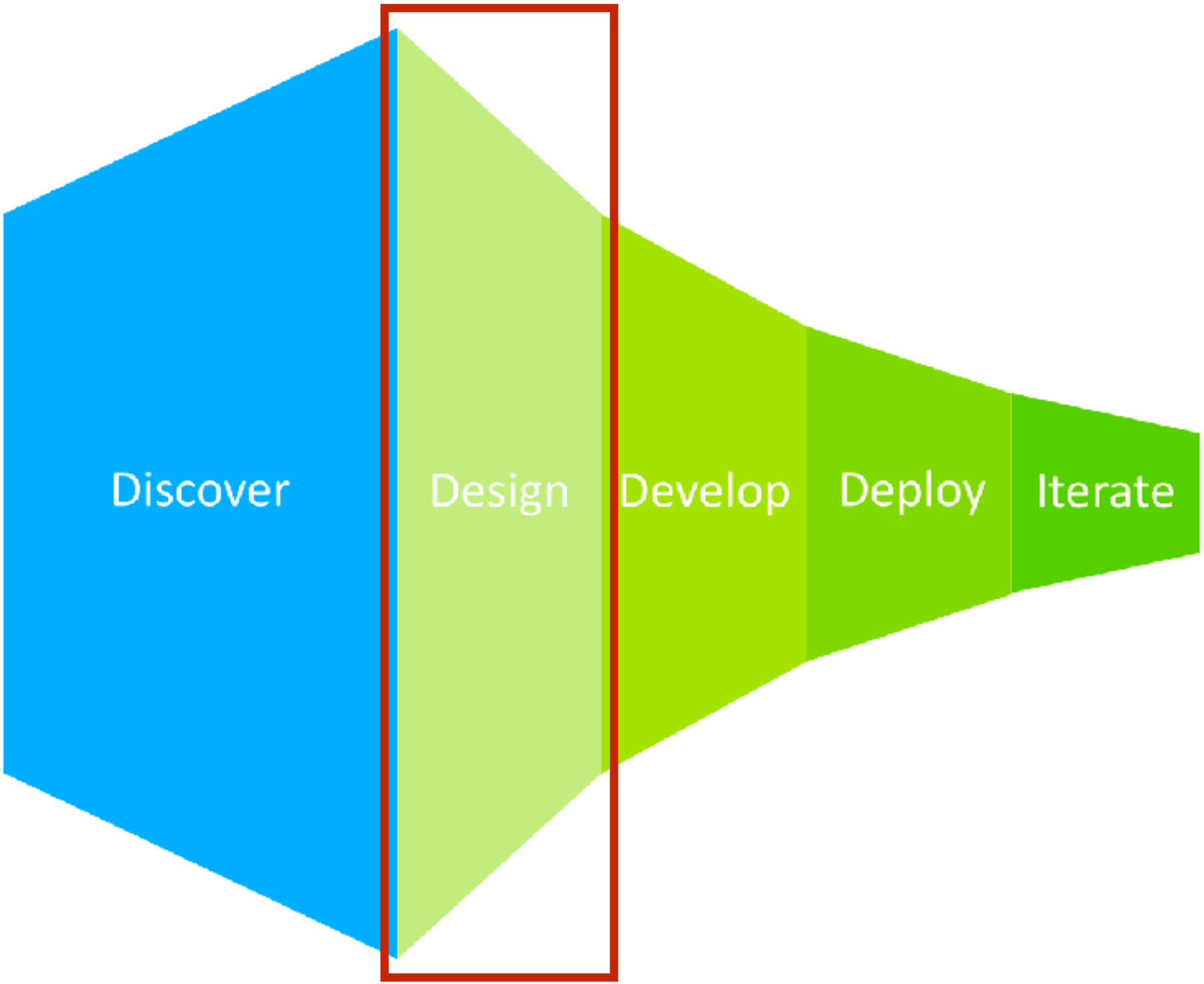


https://micro-frontends.org/

Micro frontend



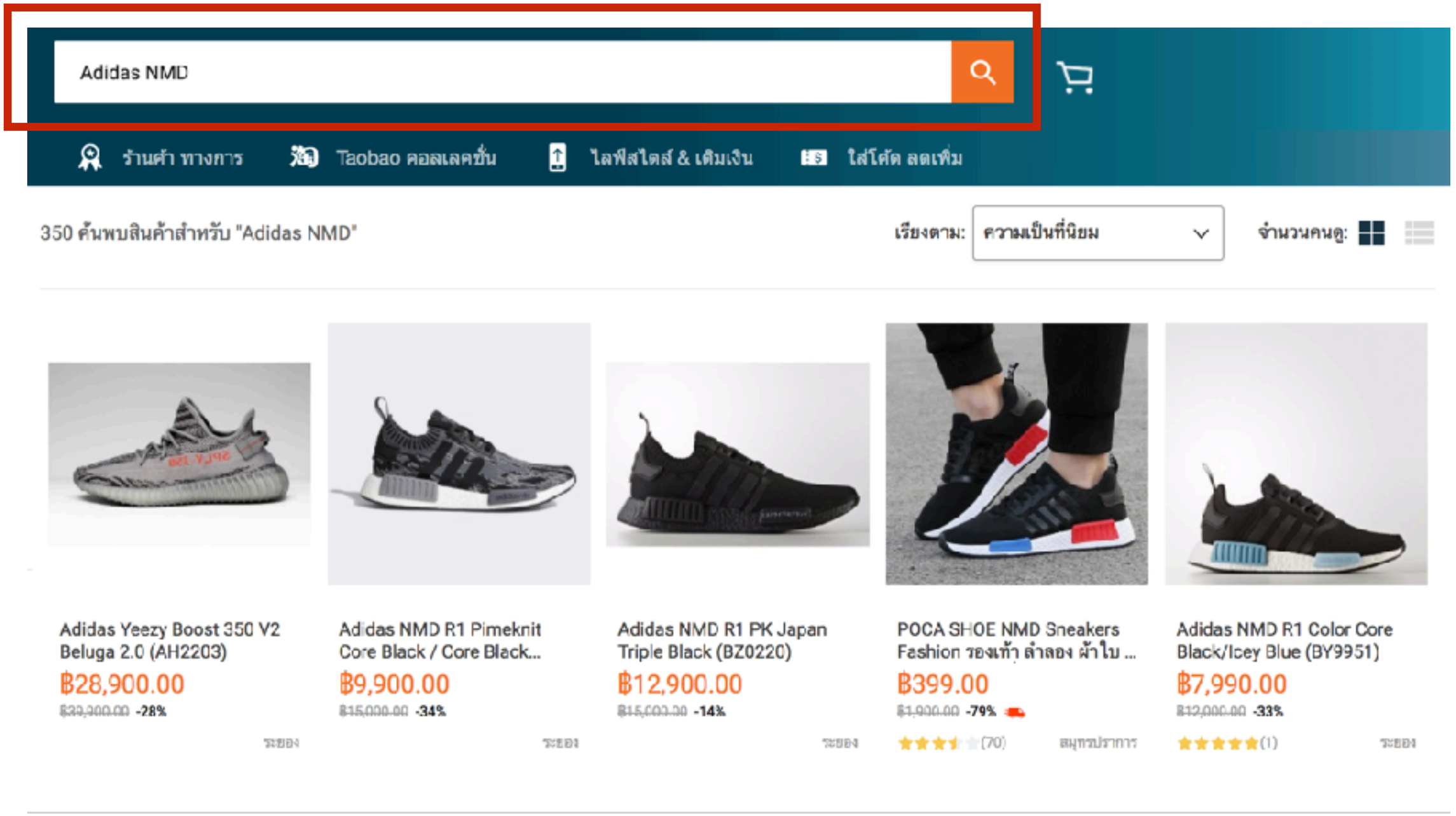
https://micro-frontends.org/



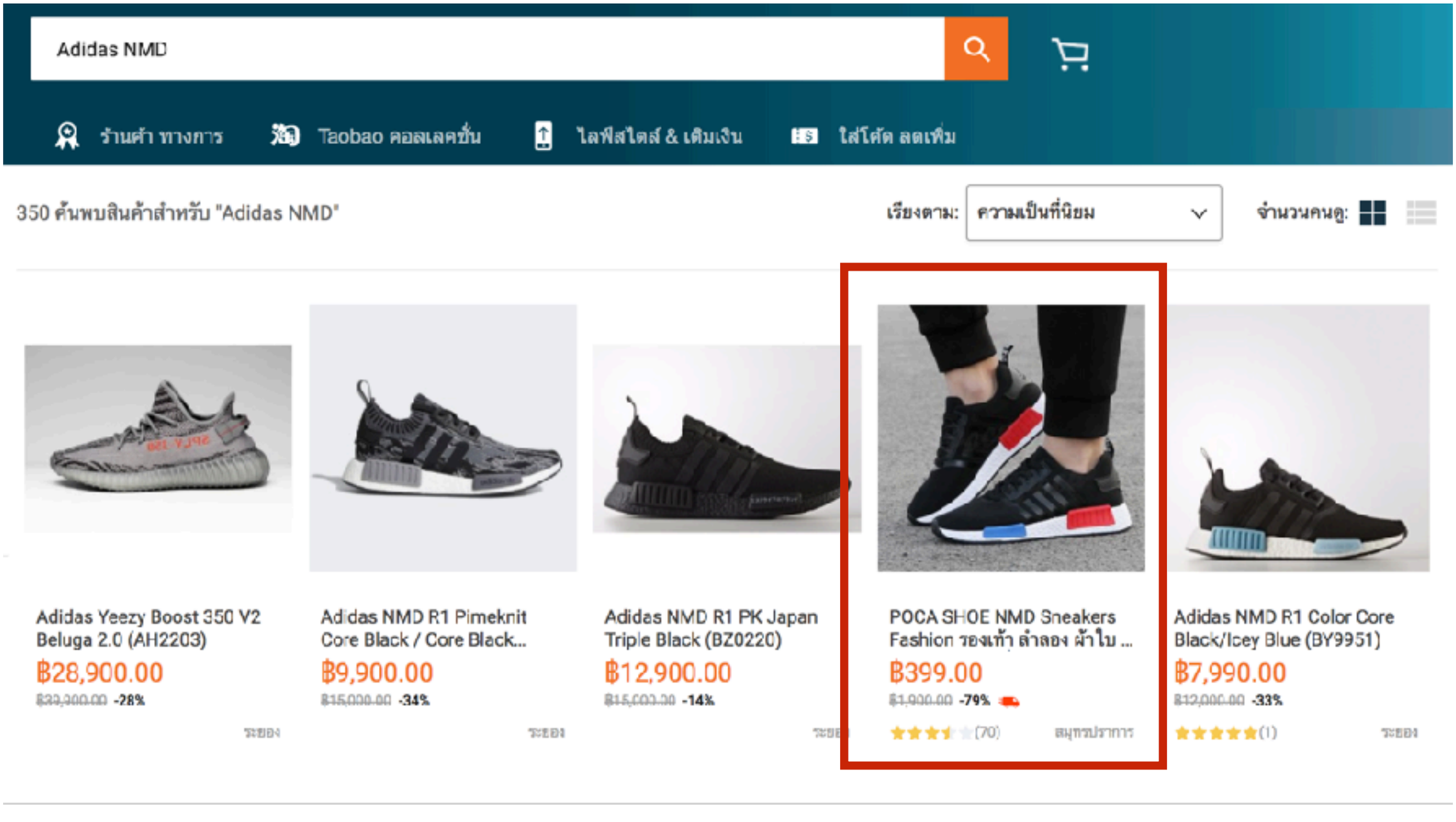
Let’s workshop with Design

E-commerce system

1. Search product by name



2. Choose a product



3. Show product detail

POCA SHOE NMD Sneakers Fashion รองเท้า ลำลอง ผ้าใบ ผู้หญิง-ผู้ชาย แฟชั่น ราคาถูกสวยๆ Sport Unisex รุ่น PSN-Black/White

(70) แสดงความคิดเห็น

ยี่ห้อ Poca Shoes | เพิ่มเติม สุภาพบุรุษ จาก Poca Shoes

2 Weeks Warranty by Seller เพิ่มเติม

- สวมใส่สบาย

เพิ่มเติม

เลือก ขนาด

ขนาด

ขนาด

399 บาท

ราคาปกติ ~~1,900 บาท,~~
ประหยัดทันที **79%**
ราคาโปรโมชั่นสามารถใช้ได้กับ 25/2/2018

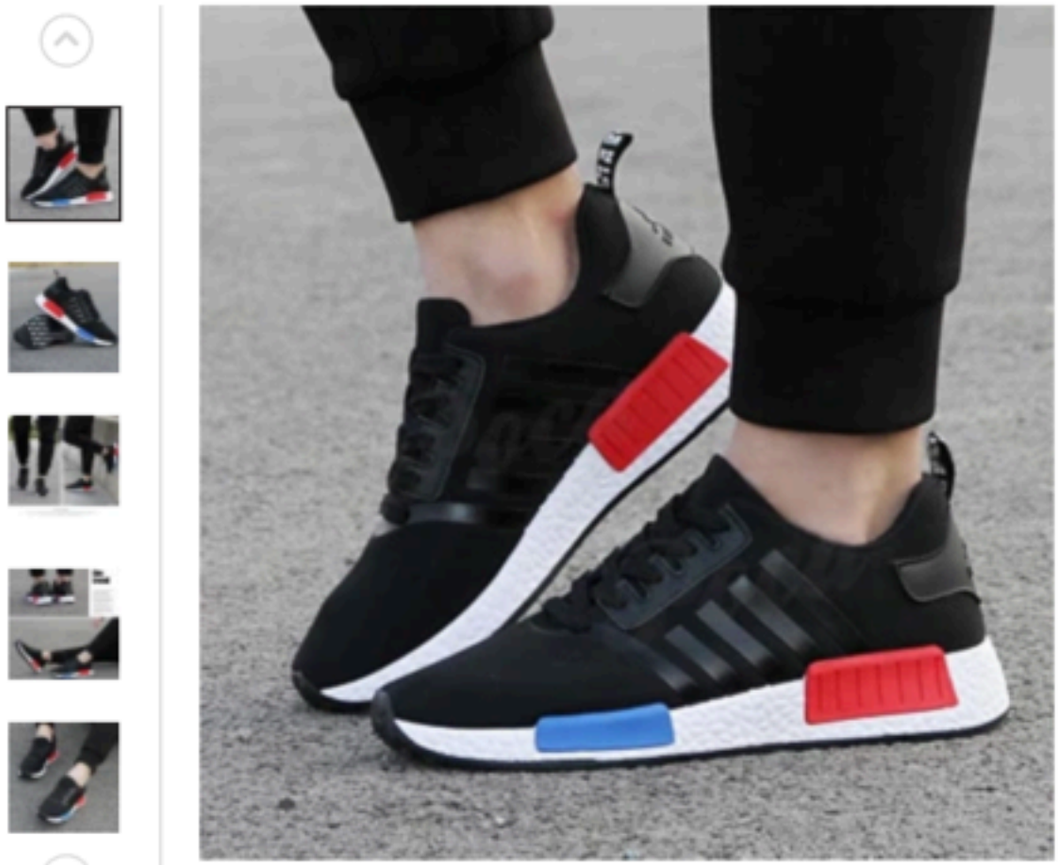
ใส่ตะกร้า วิธีการสั่งซื้อ

4. Add product to basket

POCA SHOE NMD Sneakers Fashion รองเท้า ลำลอง ผ้าใบ ผู้หญิง-ผู้ชาย แฟชั่น ราคาถูกสวยๆ Sport Unisex รุ่น PSN-Black/White

(70) แสดงความคิดเห็น

ยี่ห้อ Poca Shoes | เพิ่มเติม สุภาพบุรุษ จาก Poca Shoes



2 Weeks Warranty by Seller เพิ่มเติม

- สวมใส่สบาย

เพิ่มเติม

เลือก ขนาด

ขนาด

ขนาด ไม่มี

399 บาท

ราคาปกติ ~~1,900 บาท,~~
ประหยัดทันที **79%**
ราคาโปรโมชั่นสามารถใช้ได้กับ 25/2/2018

ใส่ตะกร้า

วิธีการสั่งซื้อ

5. Show data in basket

✓ สินค้า 1 ชิ้น ได้ถูกเพิ่มเข้าไปยังตะกร้าสินค้าของคุณ

POCA SHOE NMD Sneakers Fashion รองเท้า ลำลอง ผ้าใบ ผู้หญิง-ผู้ชาย แฟชั่น ราคาถูกสวยๆ Sport Unisex รุ่น PSN-Black/White

ไซส์: EU:40

Poca Shoes

399 บาท

~~1,900 บาท~~ 79% ปิด

ตะกร้าสินค้าของคุณ (1 สินค้า)

มูลค่าสินค้า: 399 บาท

ยอดสุทธิ ราคารวมภาษีมูลค่าเพิ่ม (ถ้ามี): **399 บาท**

เลือกซื้อสินค้าต่อ

ชำระค่าสินค้า

People Who Bought This Item Also Bought

กางเกงสแลคขายาว Hopper Progress ผ้ายืด ทรงเข้ารูป

~~900 บาท~~

67% ปิด

299 บาท

6. Checkout

✓ สินค้า 1 ชิ้น ได้ถูกเพิ่มเข้าไปยังตะกร้าสินค้าของคุณ

POCA SHOE NMD Sneakers Fashion รองเท้า ลำลอง ผ้าใบ ผู้หญิง-ผู้ชาย แฟชั่น ราคาถูกสวยๆ Sport Unisex รุ่น PSN-Black/White

ไซส์: EU:40

Poca Shoes

399 บาท

~~1,900 บาท~~ 79% ปิด

ตะกร้าสินค้าของคุณ (1 สินค้า)

มูลค่าสินค้า: 399 บาท

ยอดสุทธิ ราคารวมภาษีมูลค่าเพิ่ม (ถ้ามี): **399 บาท**

เลือกซื้อสินค้าต่อ

ชำระค่าสินค้า

People Who Bought This Item Also Bought

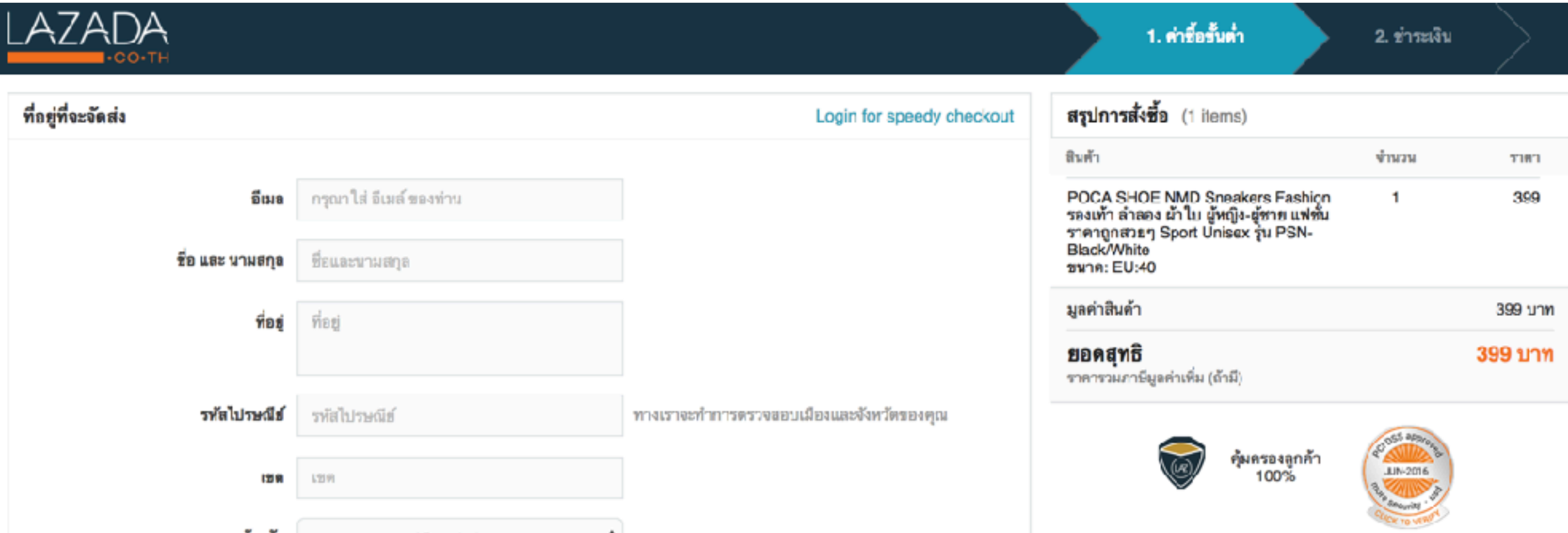
กางเกงสแลคขายาว Hopper Progress ผ้ายืด ทรงเข้ารูป

~~900 บาท~~

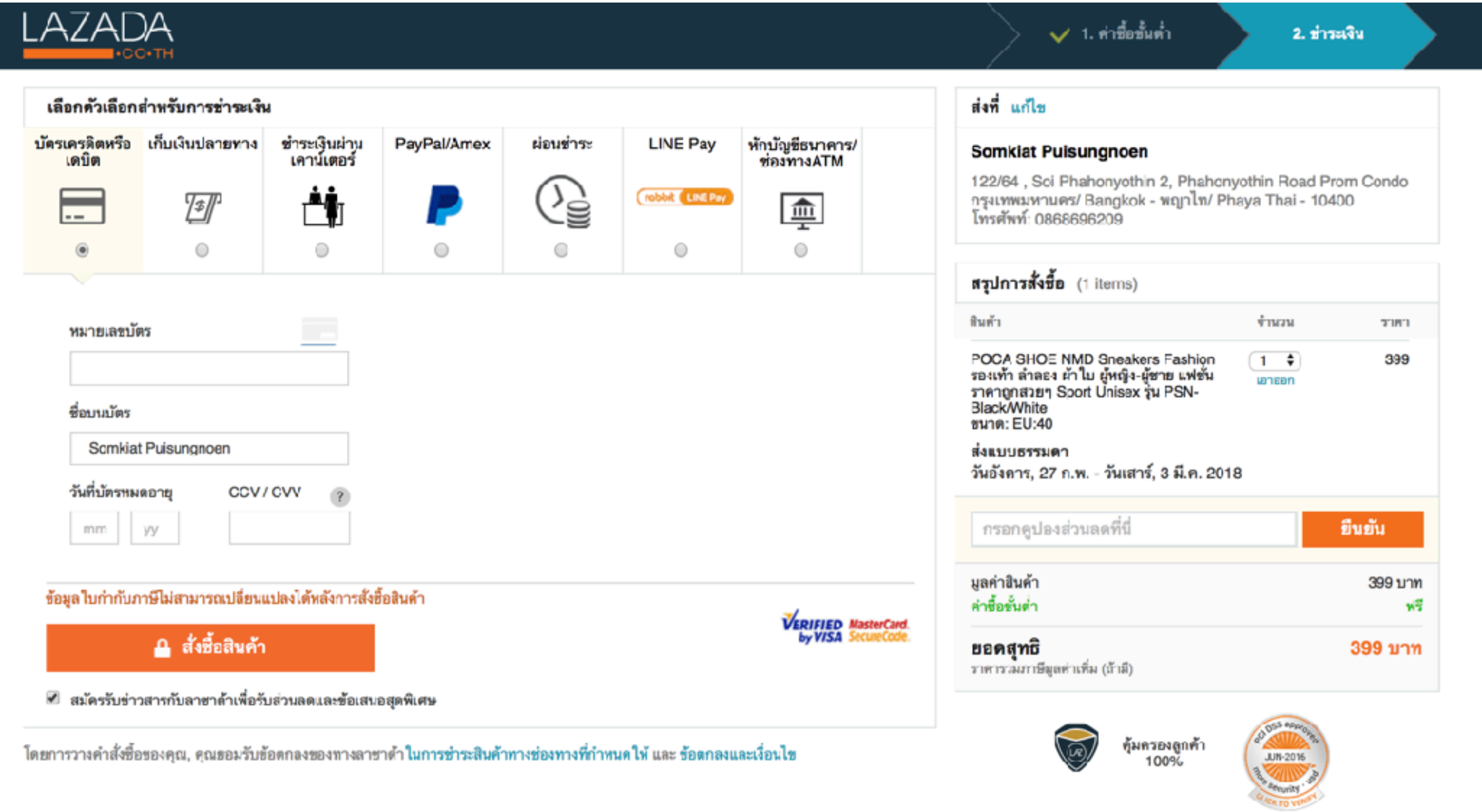
67% ปิด

299 บาท

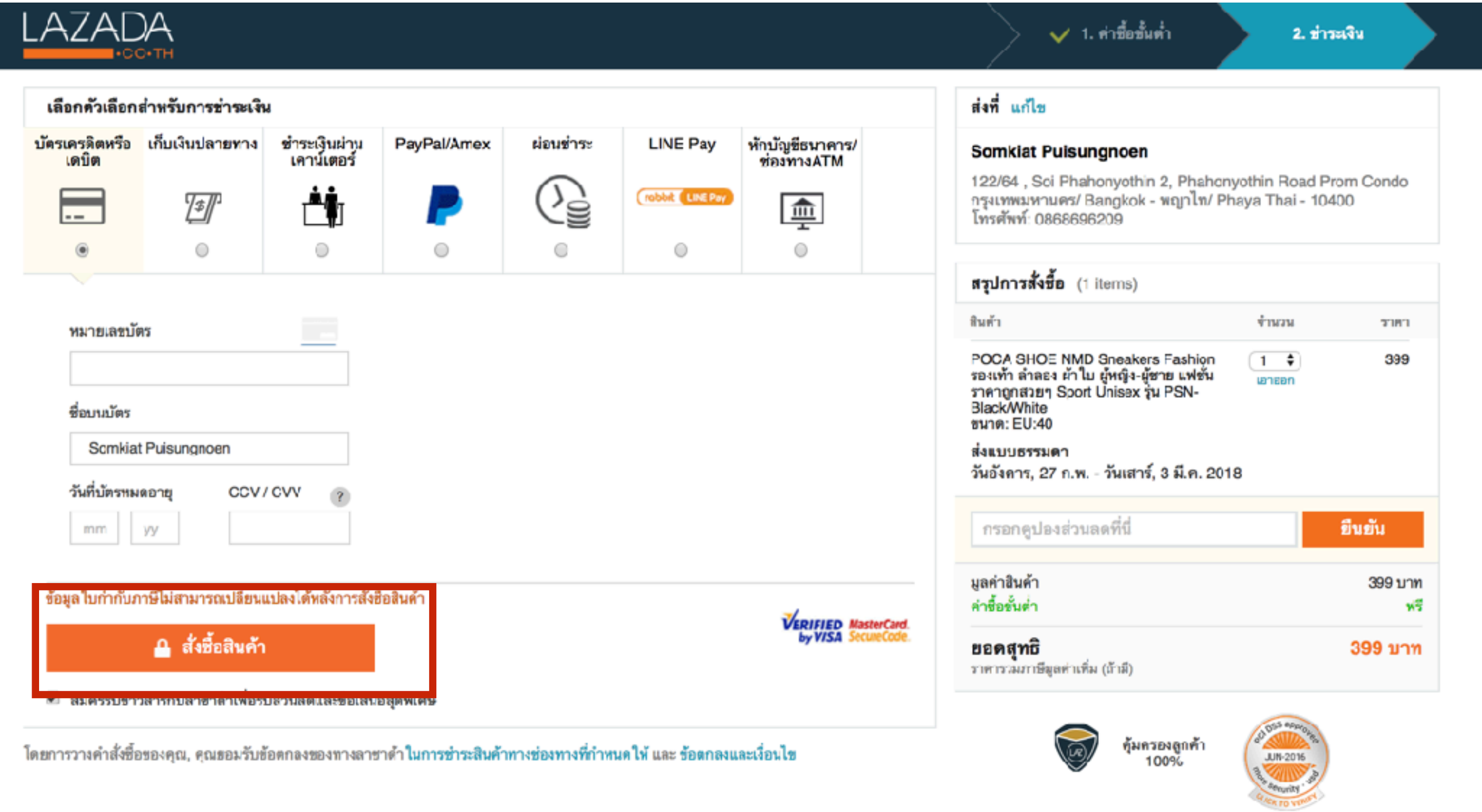
7. Shipping



8. Payment



9. Confirm to order



10. Summary

LAZADA .CO.TH

Pay@All by counterservice

ใบแจ้งการชำระเงิน(PaySlip)

Counter Service Co., Ltd.

เลขที่ใบแจ้งสินค้า/Invoice No:	**3779254692**
ผู้ชำระเงิน/Payer:	Somkiat Puisungnoen
วันที่ทำรายการ / Transaction Date:	25/02/2018 23:33
กำหนดชำระเงิน / Expired Date:	**27/02/2018 23:33**
เพื่อเข้าบัญชี / Payee:	www.lazada.co.th Tel: **020180000**
รายละเอียด / Detail:	Lazada

806010855864737

จำนวนเงินที่ชำระ / Amount:

399.00 บาท /THB

* ไม่รวมค่าธรรมเนียมของเคาน์เตอร์เซอร์วิส
(Excluding service fees at Counter Service)

คลิกปุ่ม **"Print"** พิมพ์ใบแจ้งการชำระเงิน
หรือ
จด **"รหัส 15 หลักใต้บาร์โค้ด"** เพื่อนำไปชำระเงินที่
Press "Print" button or write down paycode 15 digits for pay in cash at counter service(7-11)

Back to merchant | Print

Try to design system

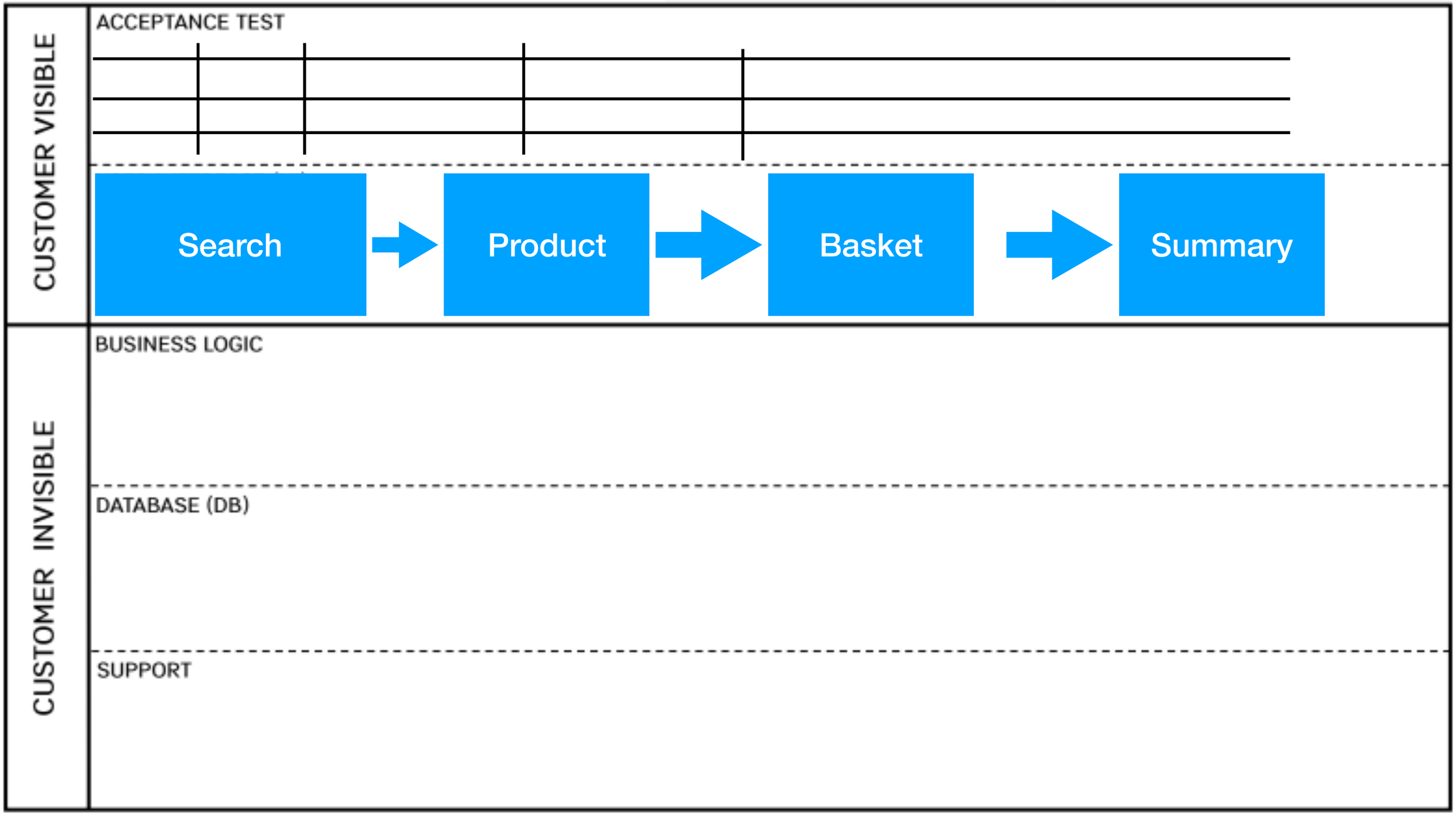
A-DAPT Blueprint

THEME:	EPIC:	FEATURE:	STORY:
DESIGNED BY:	DATE:	NOTE:	

CUSTOMER VISIBLE	ACCEPTANCE TEST
	USER INTERFACE (UI)
CUSTOMER INVISIBLE	BUSINESS LOGIC
	DATABASE (DB)
	SUPPORT

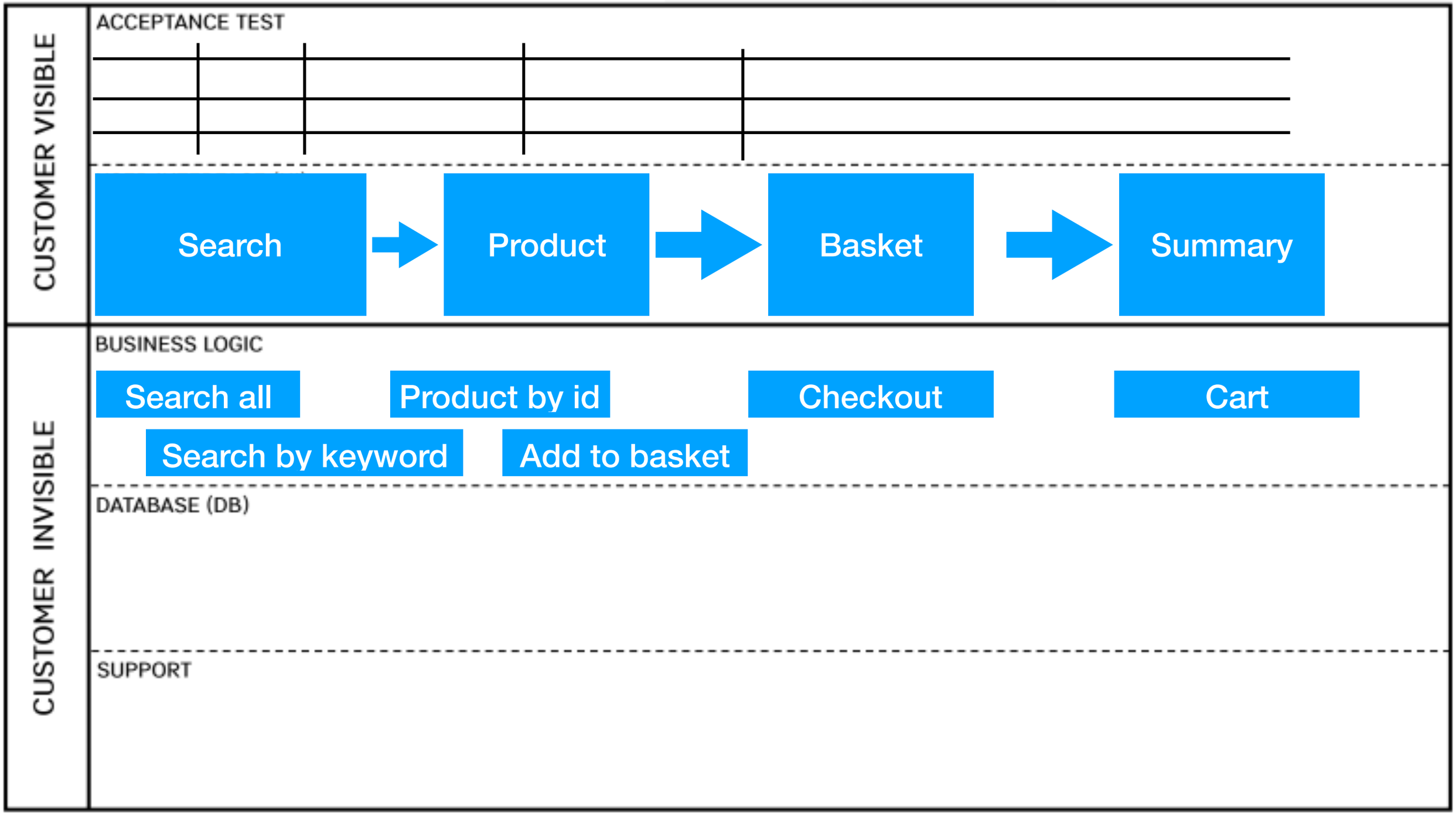
A-DAPT Blueprint

THEME:	EPIC:	FEATURE:	STORY:
DESIGNED BY:	DATE:	NOTE:	



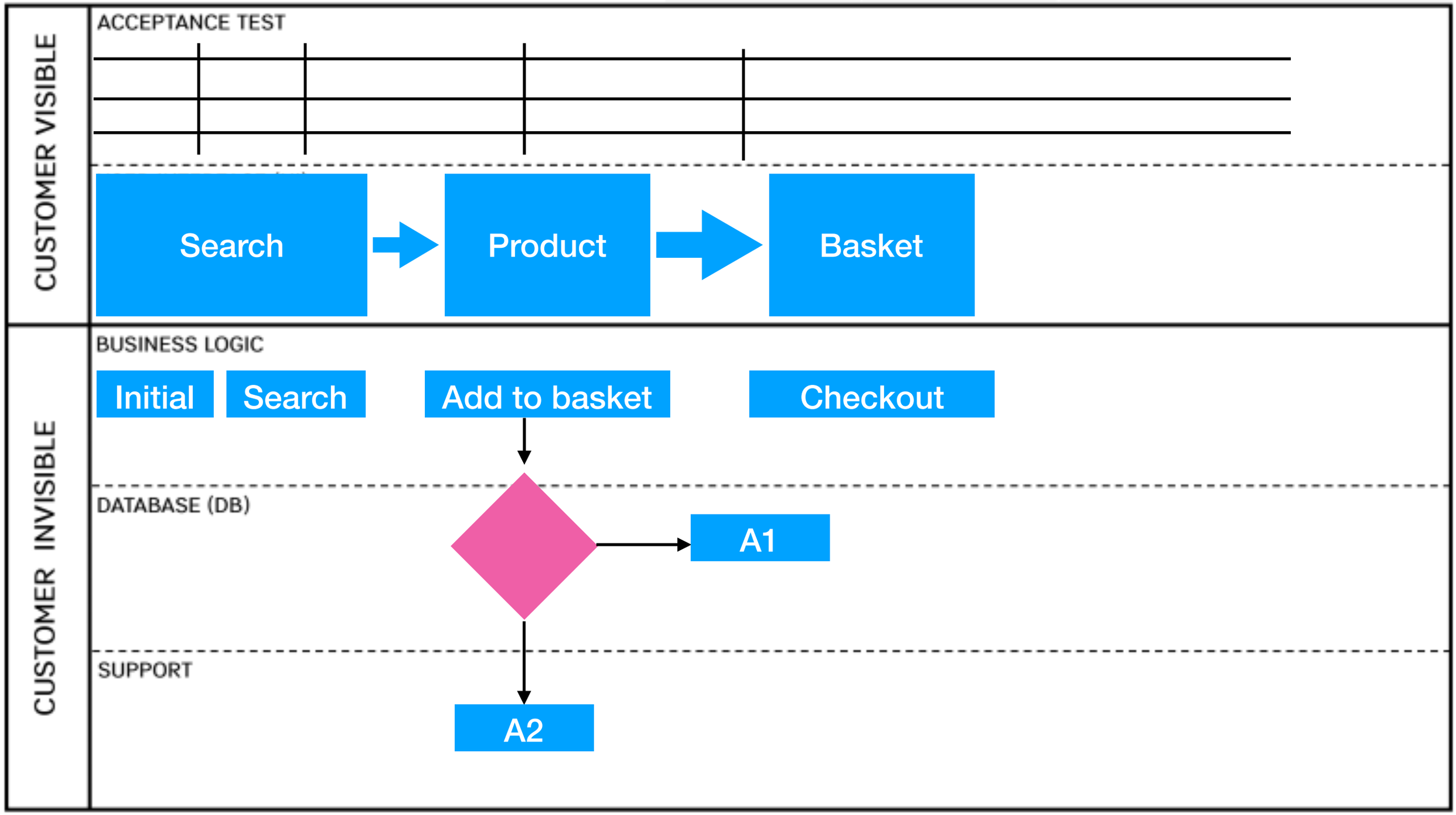
A-DAPT Blueprint

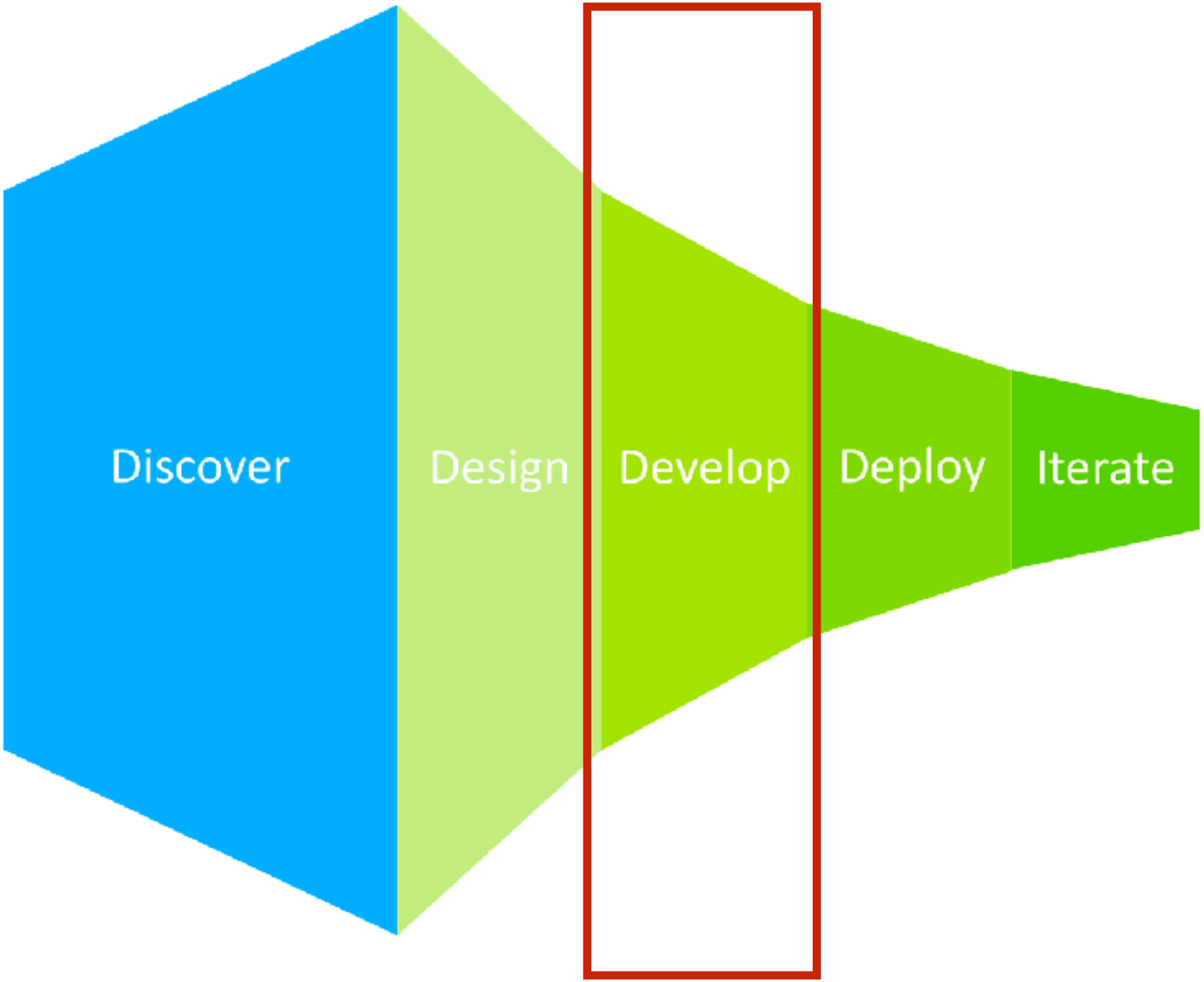
THEME:	EPIC:	FEATURE:	STORY:
DESIGNED BY:	DATE:	NOTE:	



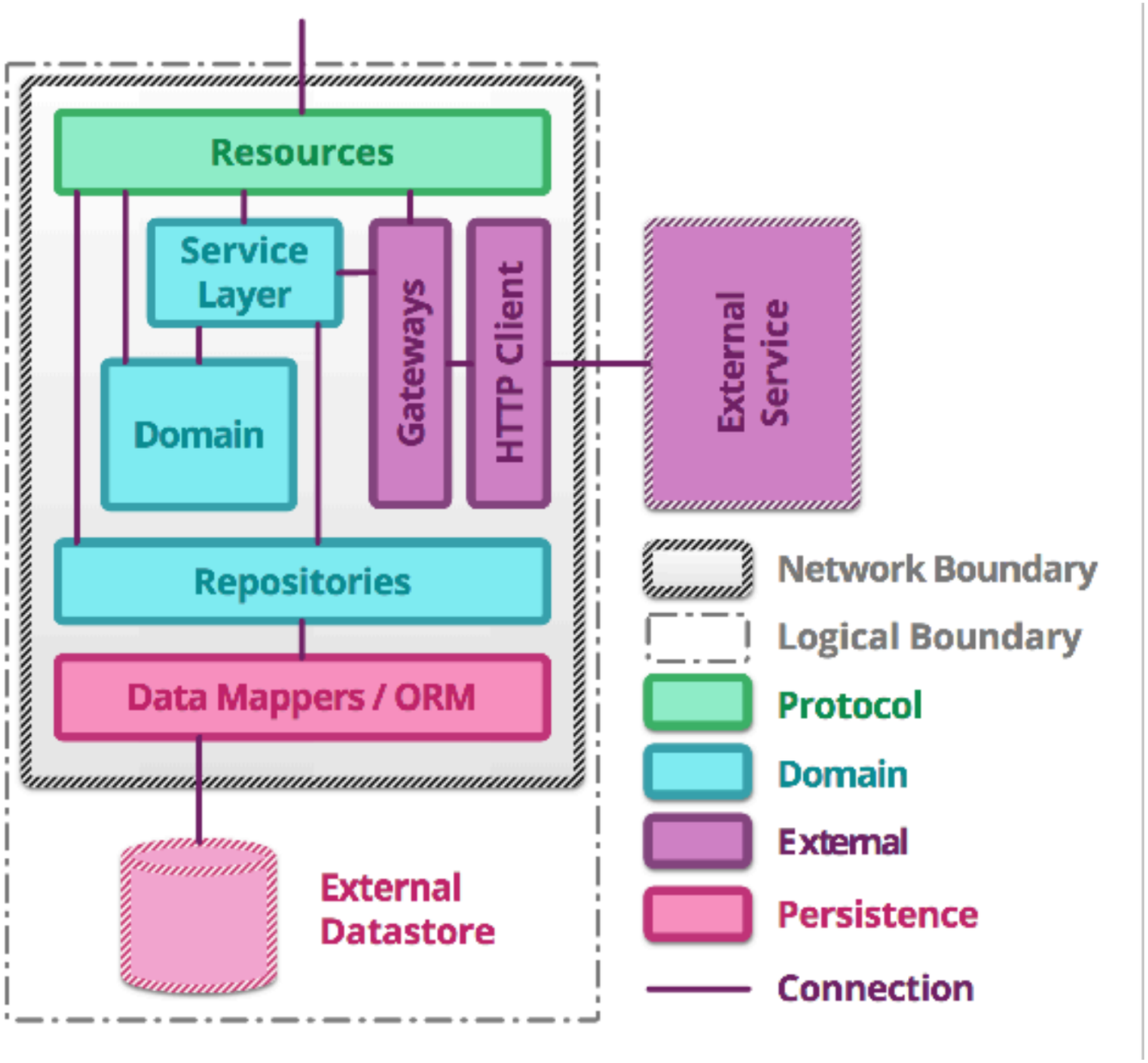
A-DAPT Blueprint

THEME:	EPIC:	FEATURE:	STORY:
DESIGNED BY:	DATE:	NOTE:	



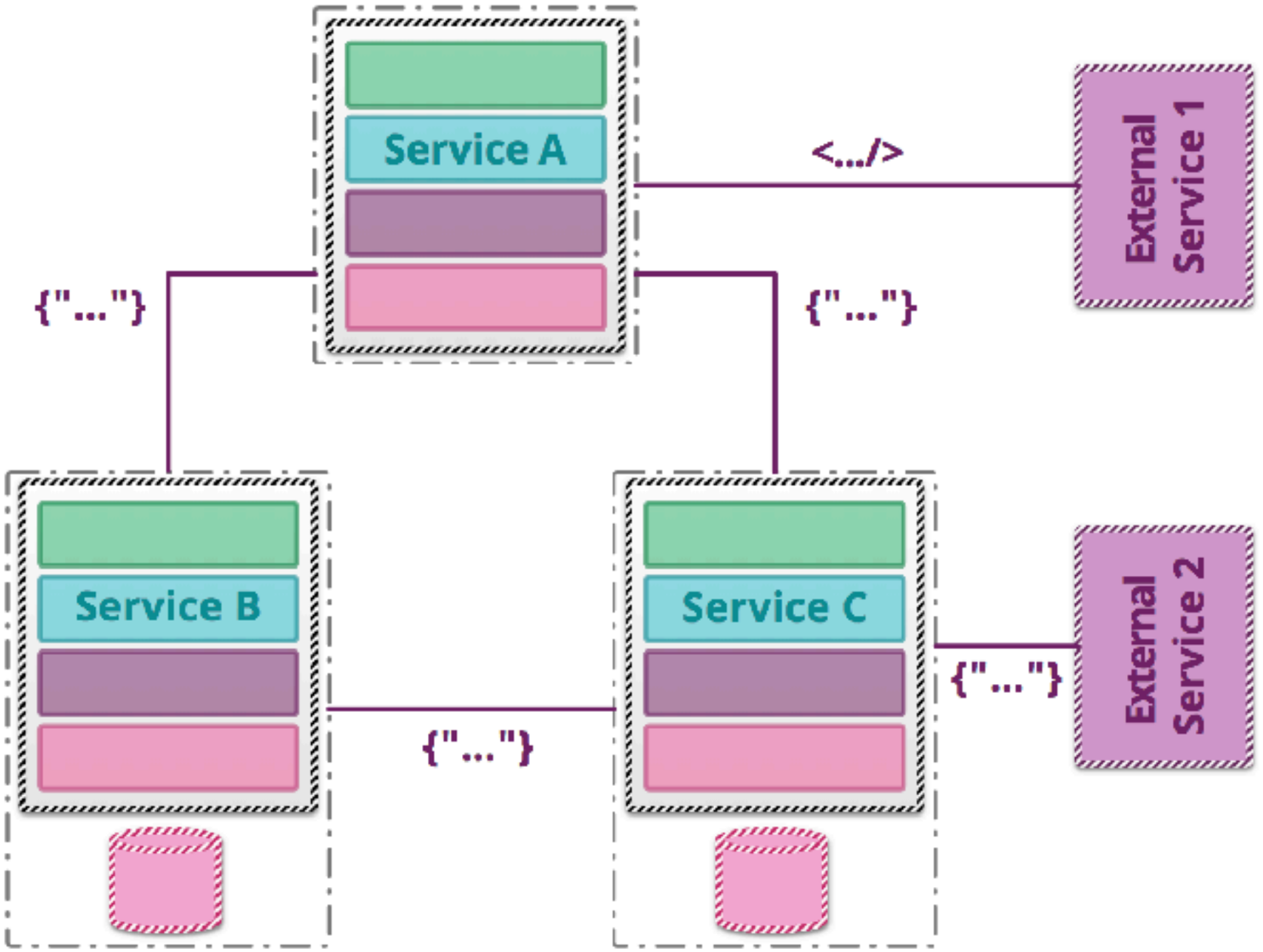


Service structure



https://martinfowler.com/articles/microservice-testing

Multiple services



https://martinfowler.com/articles/microservice-testing

Twelve-Factor App

Twelve-Factor App

https://12factor.net/

Initial to build app for Heroku

Principles to cloud and container native app

Twelve-Factor App

https://12factor.net/

Build process => compile, linking and **testing**

1. Codebase

Codebase must be tracked in version control and will have many deploy

2. Dependencies

Dependencies are explicitly declared and isolated

Use dependency management tools to shared libraries

3. Configuration

Store configuration in the environment

Add configuration in environment variables or config files

4. Backing services

Treat backing services as an attached resource

Should easy to deploy and change

5. Build, Release, Run

Always have a build and deploy strategy

Build strategies for repeated builds, versioning of running system and rollback

6. Processes

Execute the application as a stateless process

7. Port Binding

Expose services via port bindings

8. Concurrency

Scale out with the process model

9. Disposability

Quick application startup and shutdown times
Graceful shutdown

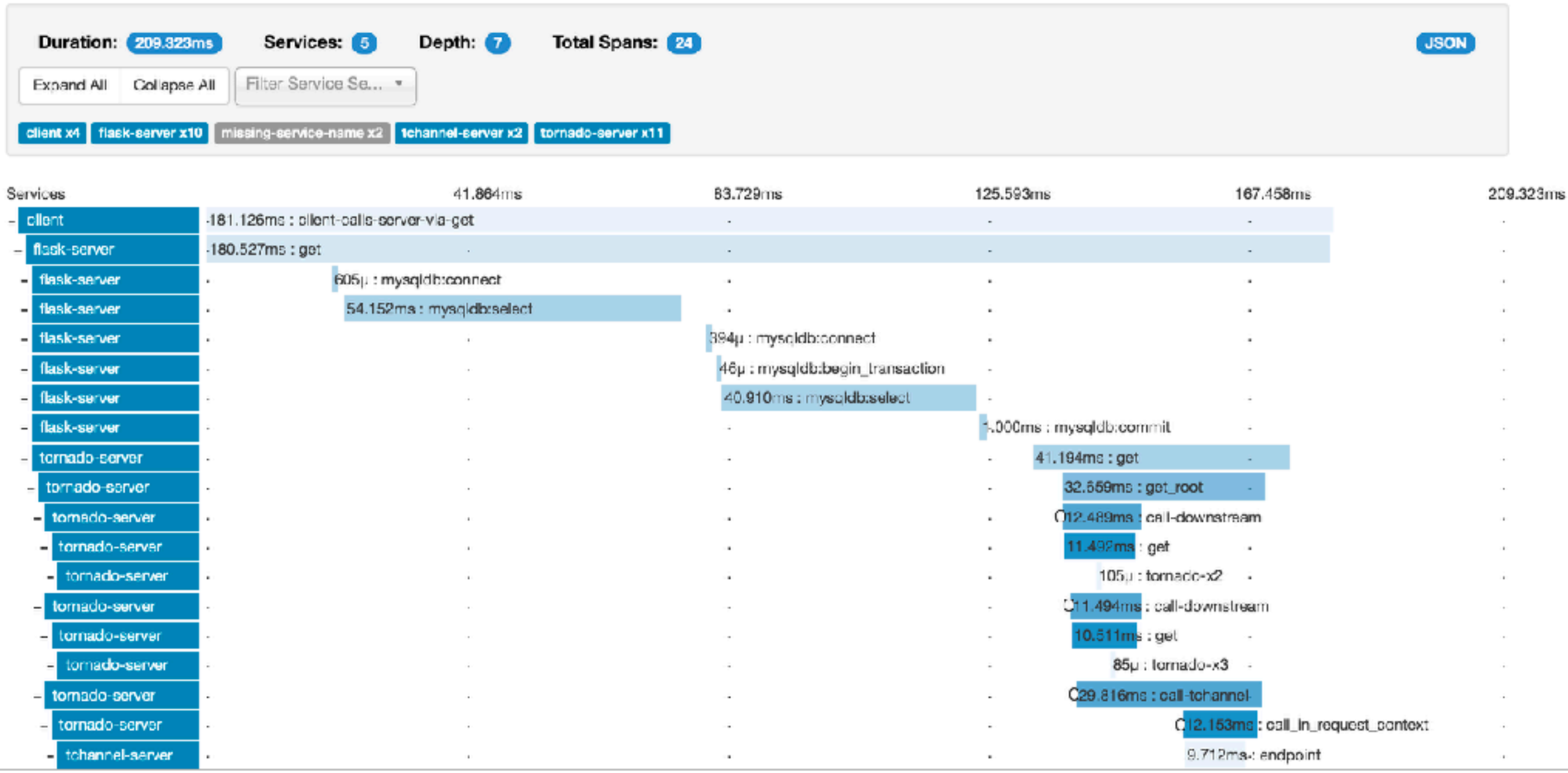
10. Dev/prod parity

Application is treated the same way in dev, staging and production

11. Log management

Treated as an event stream

Logging/Tracing

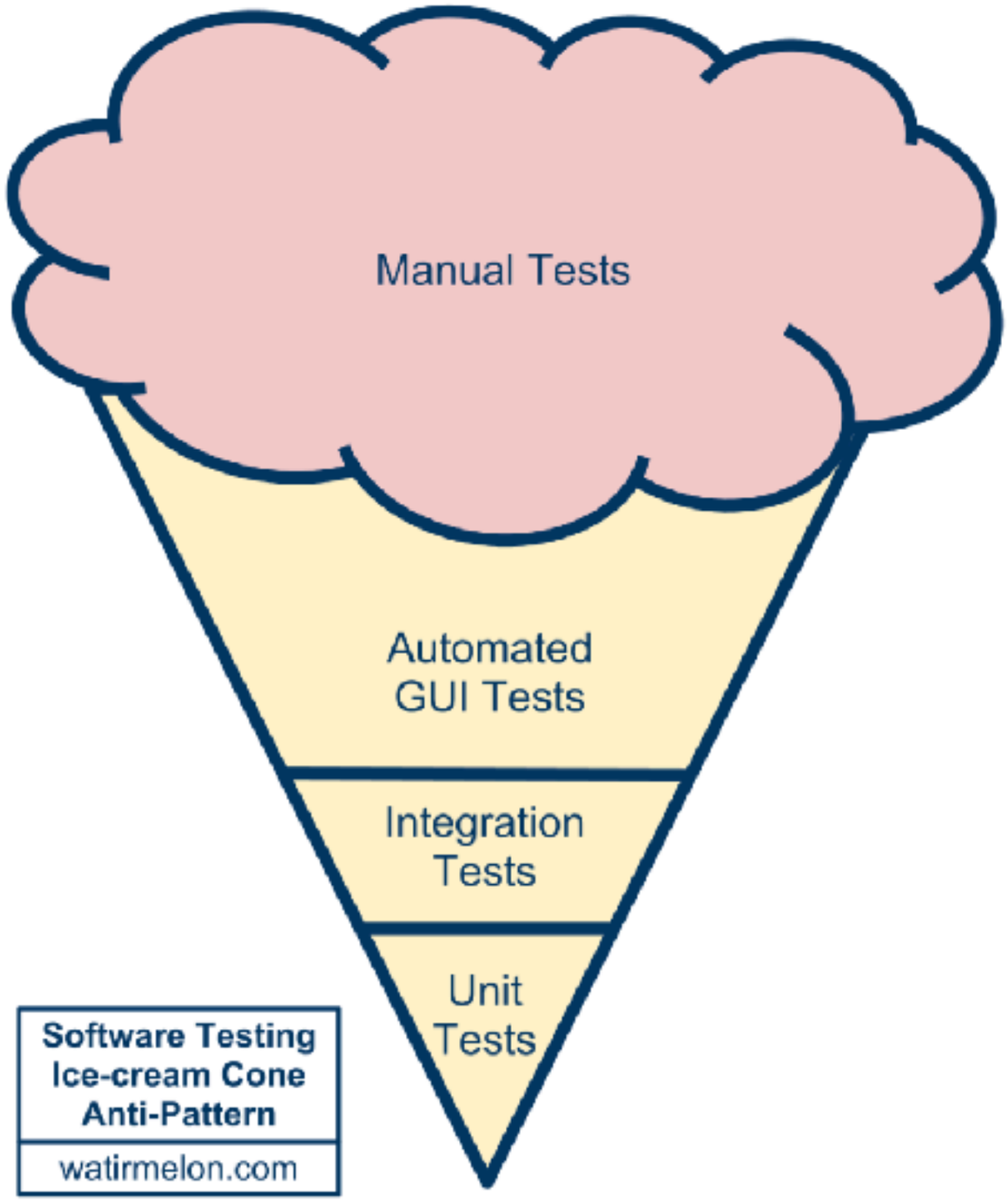


https://zipkin.io/

12. Admin tasks

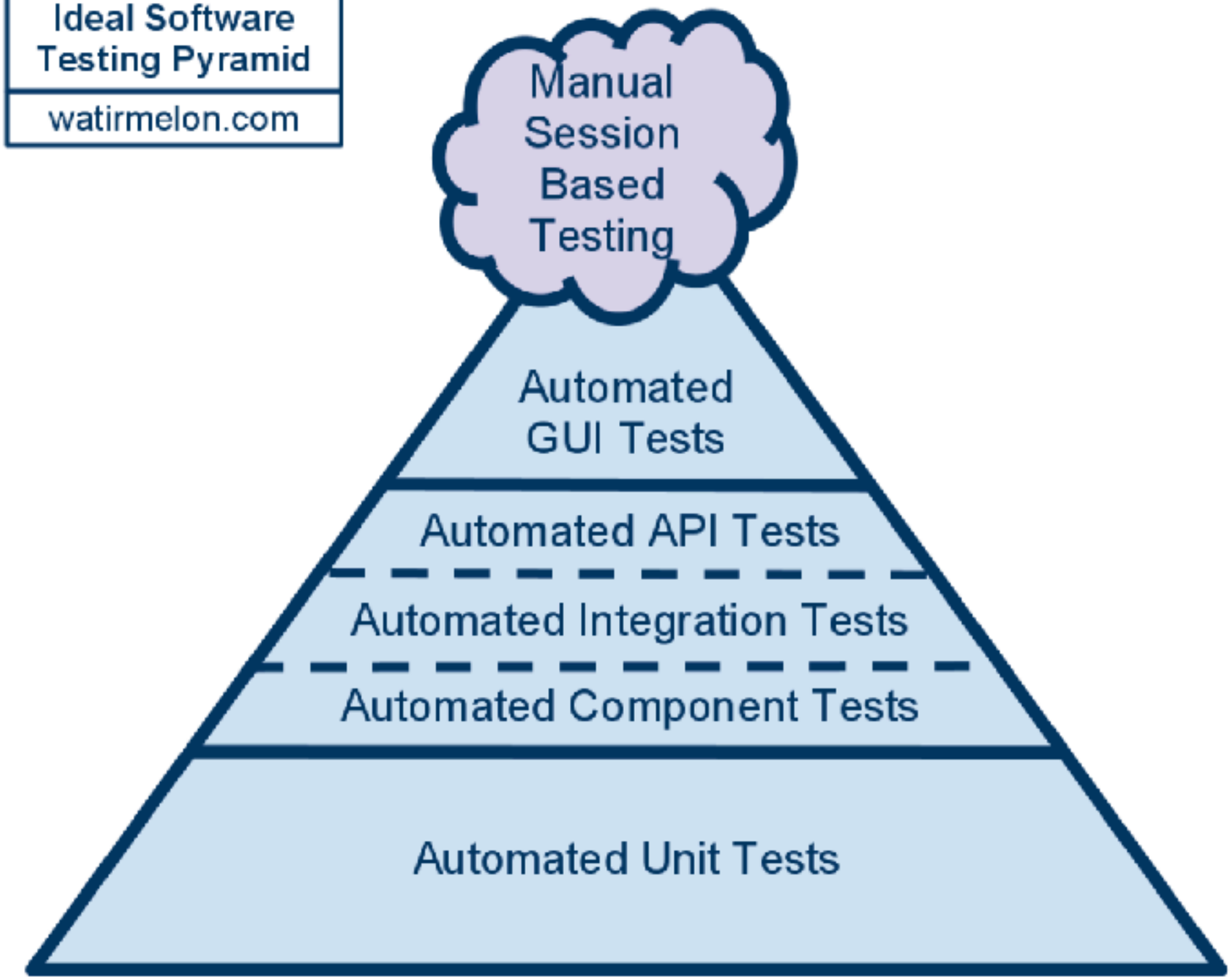
Treated the same way like the rest of the application

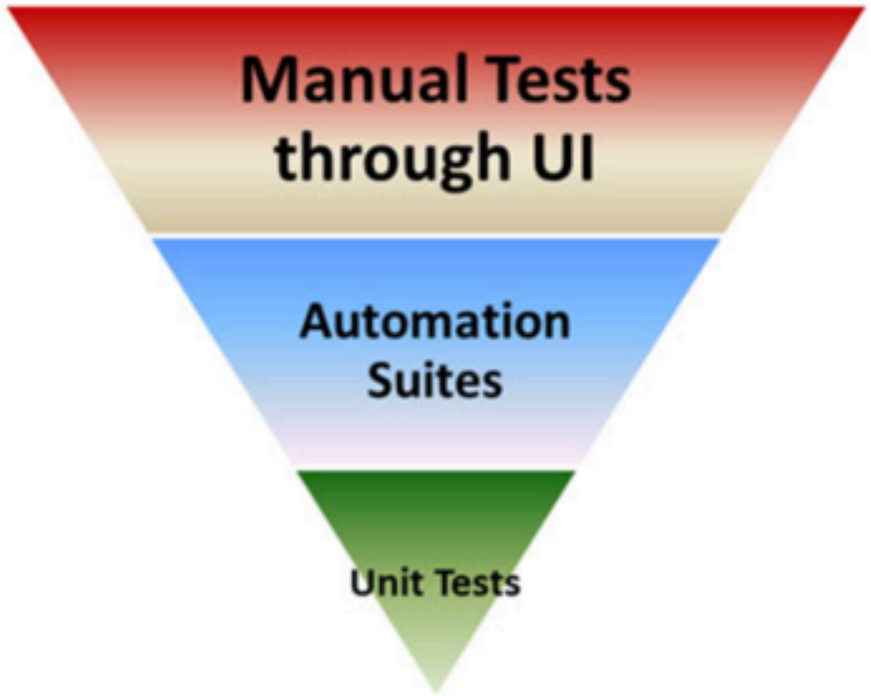
Microservice Testing



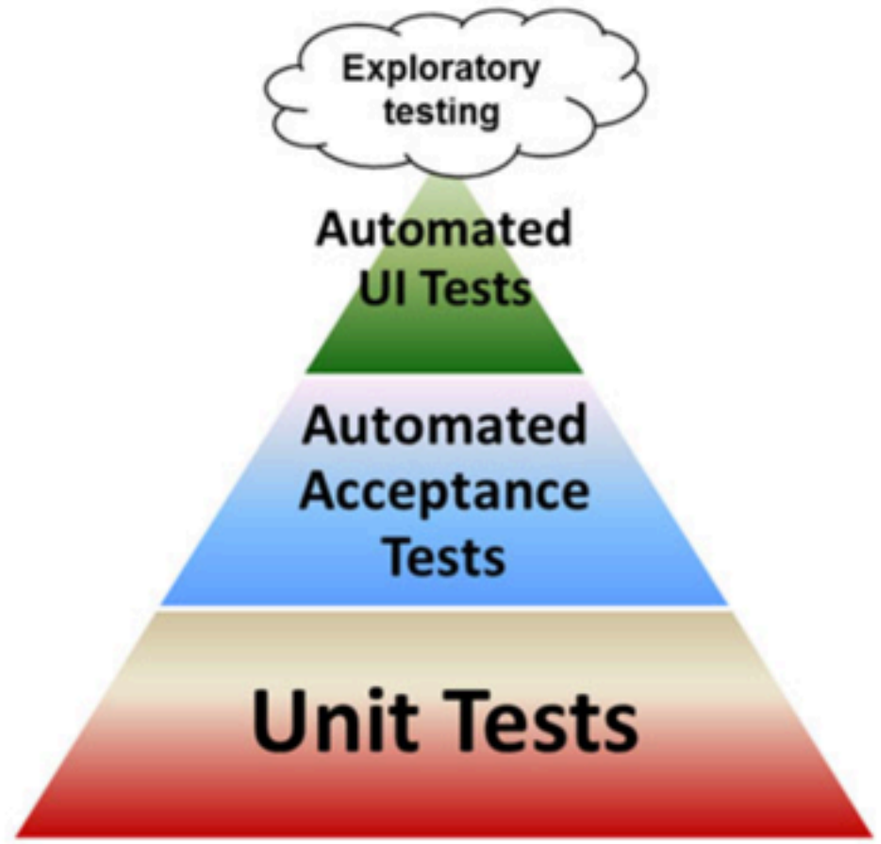
Ideal Software Testing Pyramid

watirmelon.com

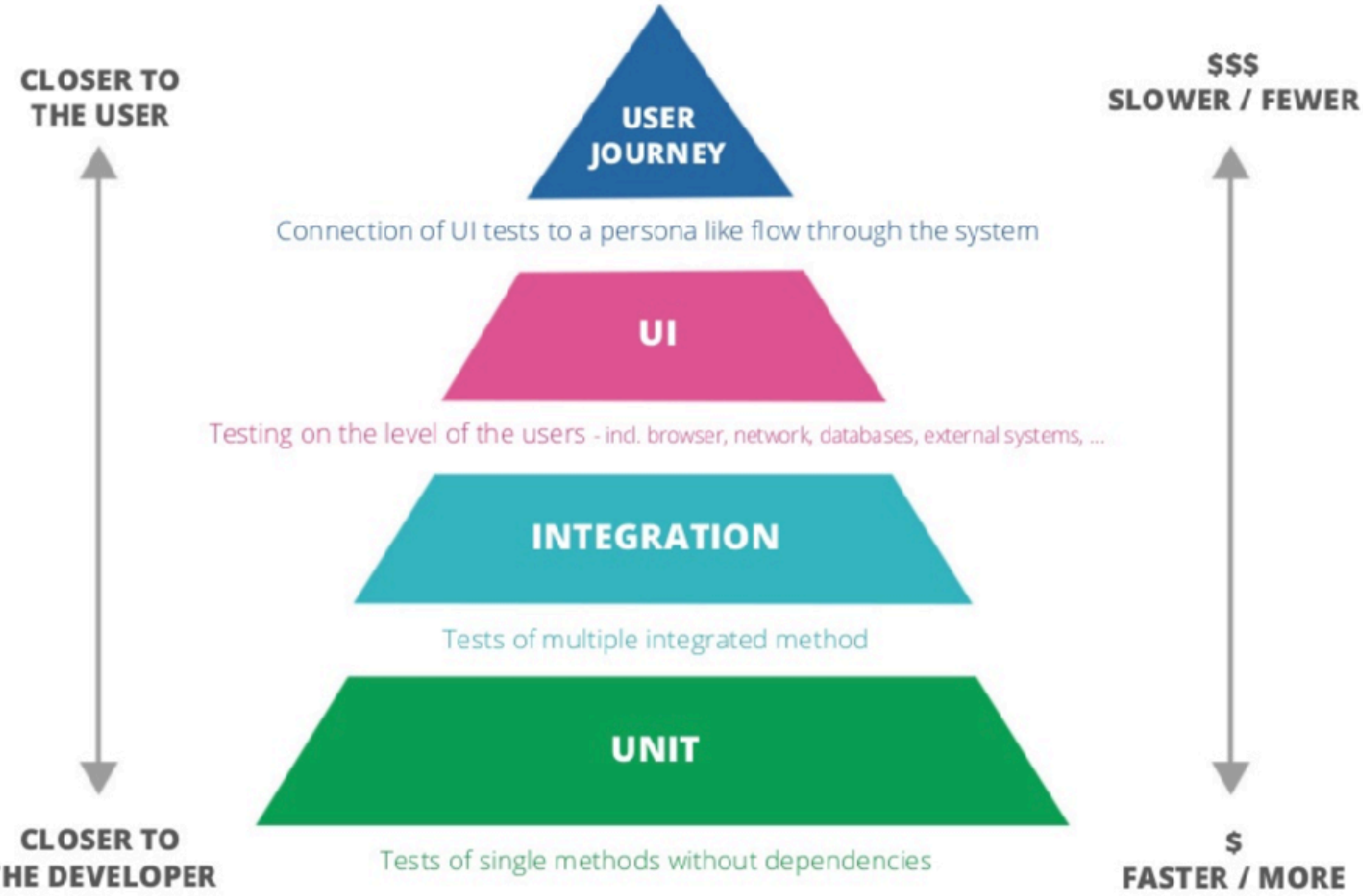




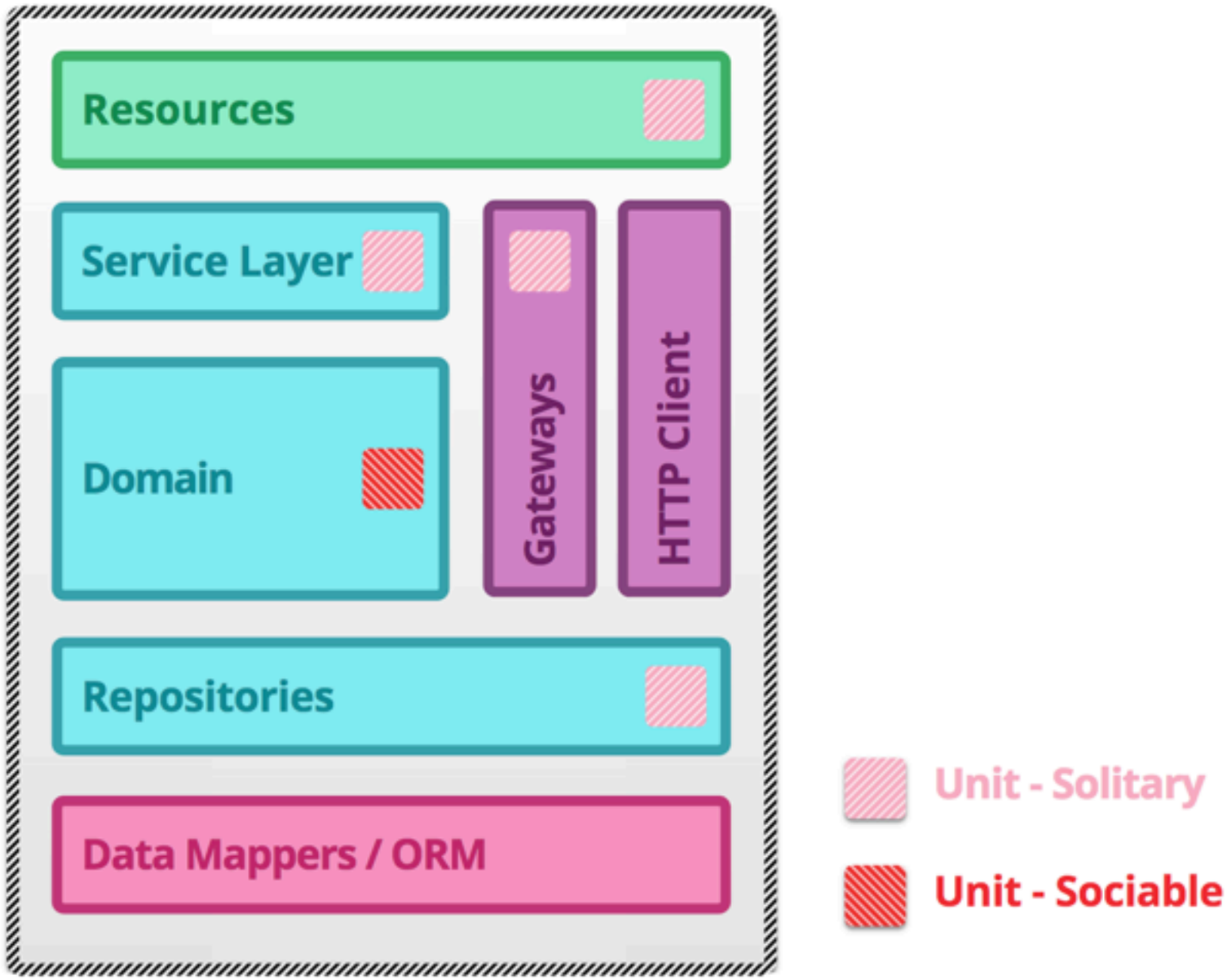
Traditional
(find bugs)



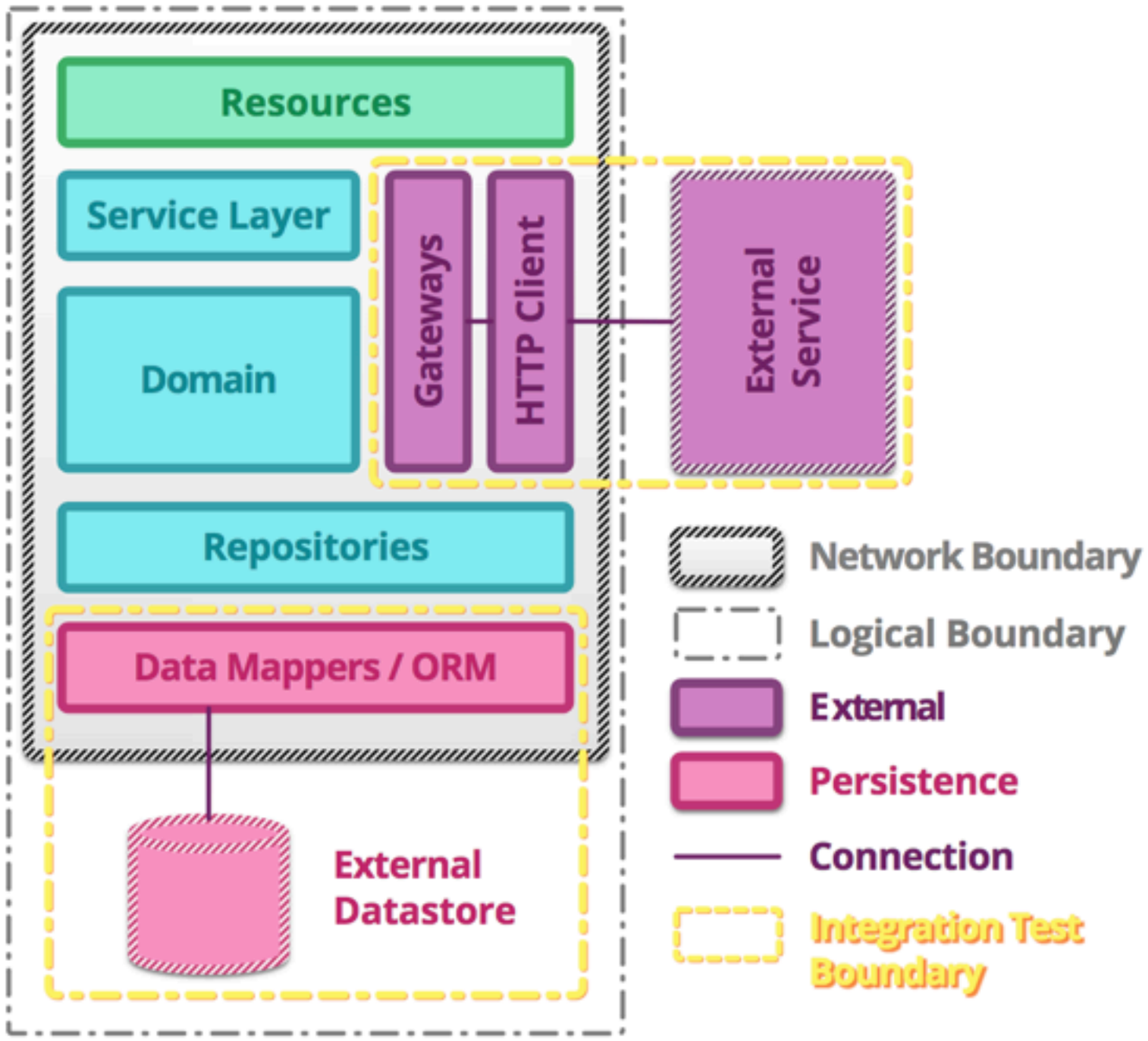
Agile
(prevent bugs)



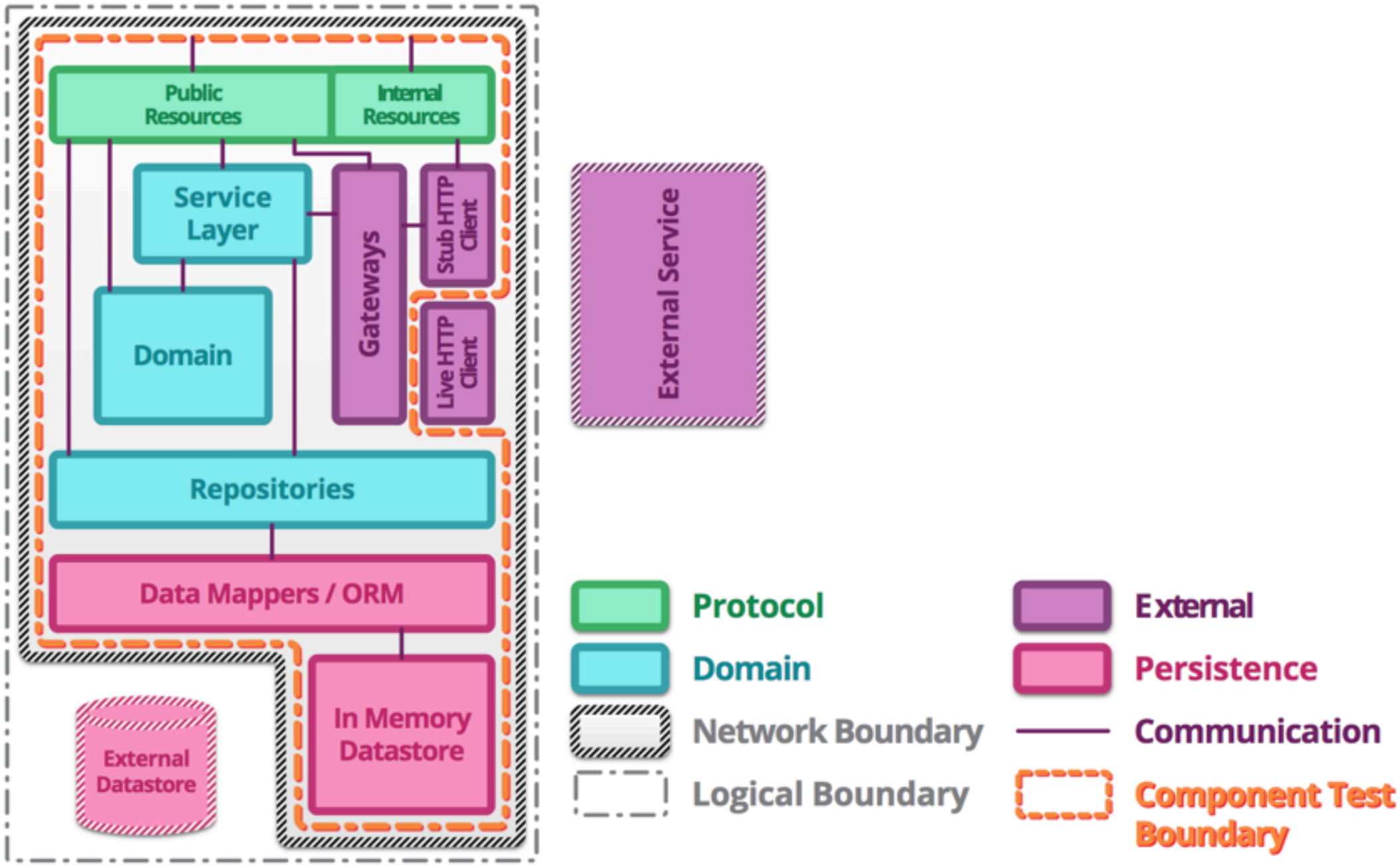
Unit testing



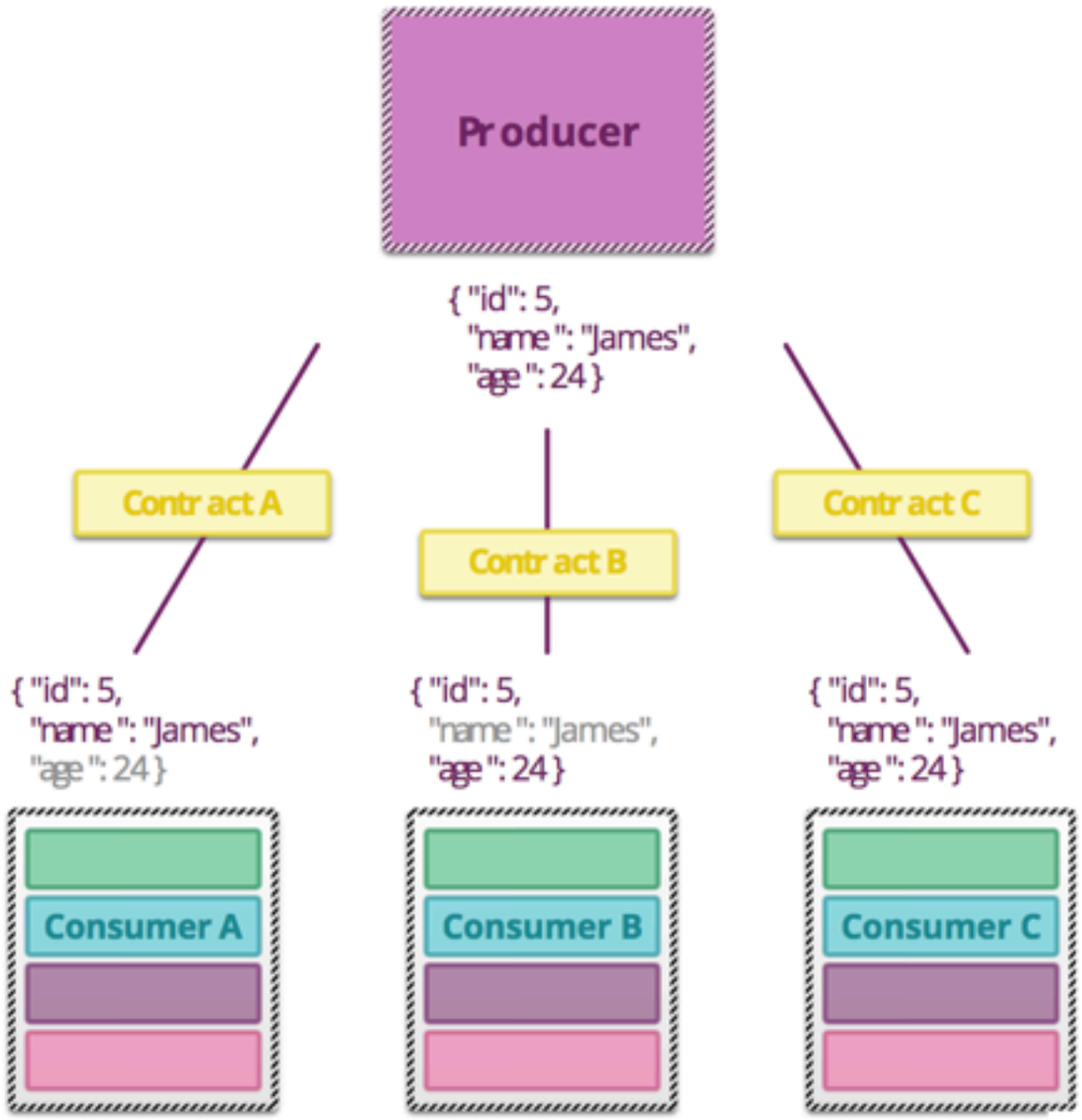
Integration testing



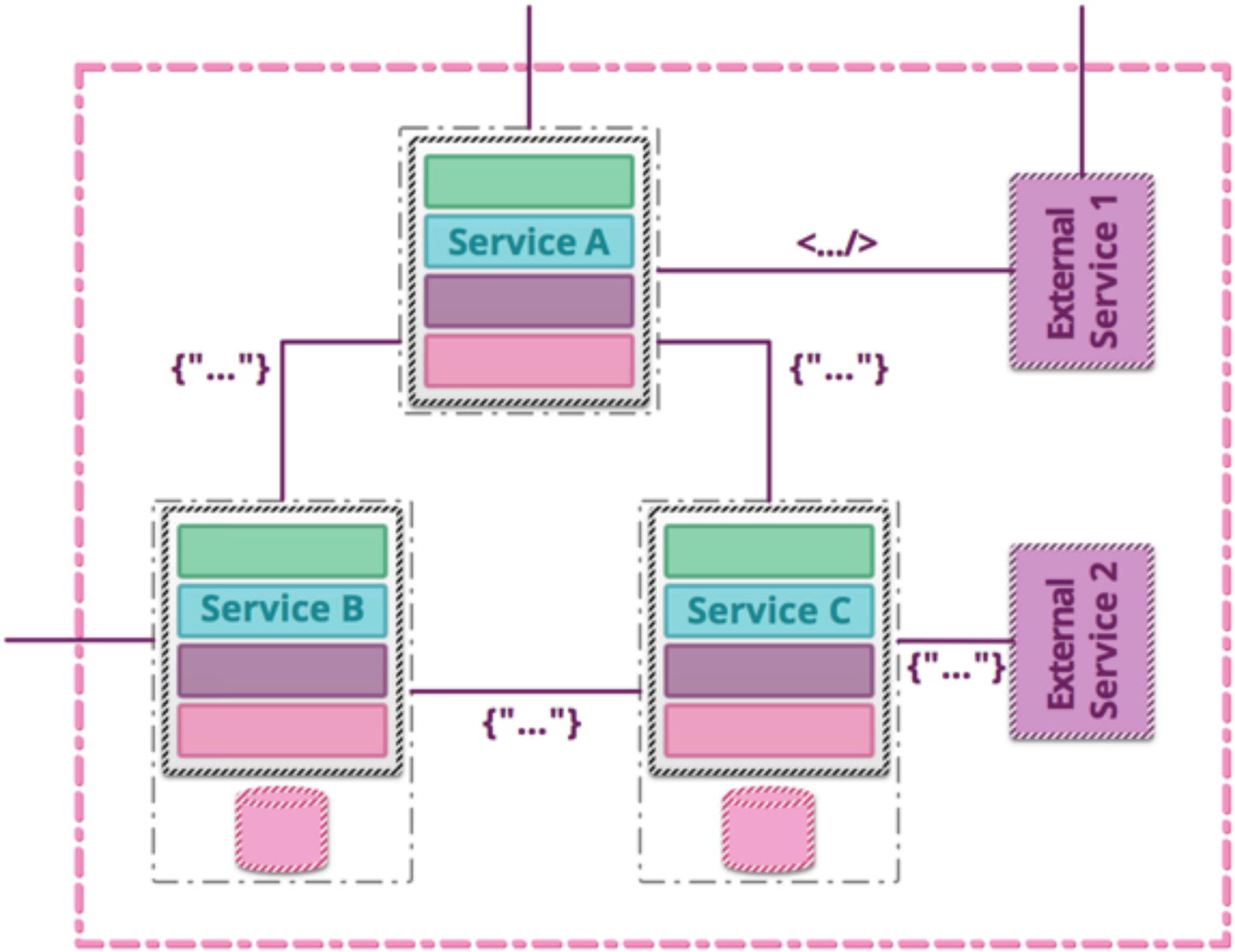
Component testing



Contract testing



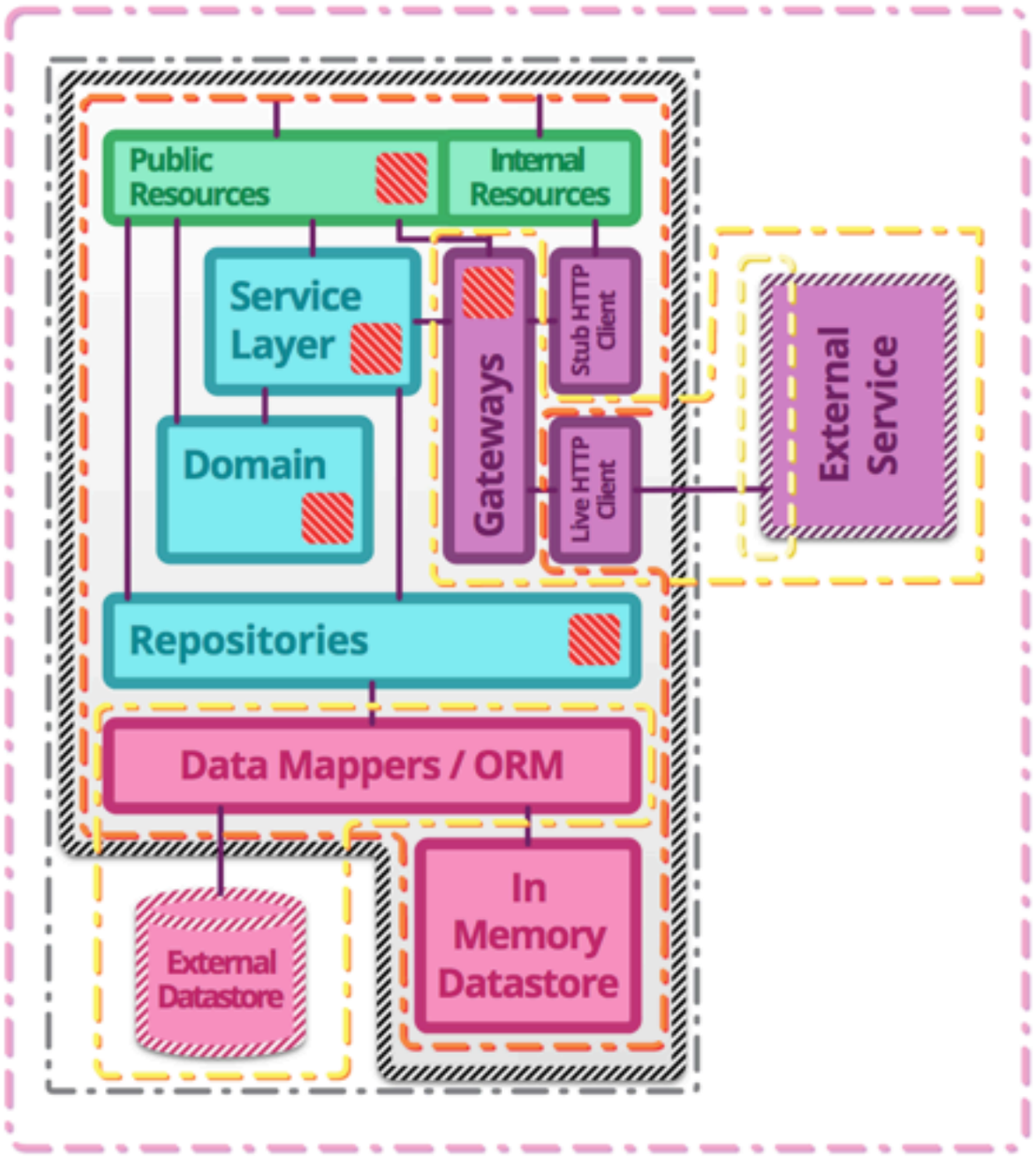
End-to-End testing



Summary

Unit tests : exercise the smallest pieces of testable software in the application to determine whether they behave as expected.

Integration tests : verify the communication paths and interactions between components to detect interface defects.



Component tests : limit the scope of the exercised software to a portion of the system under test, manipulating the system through internal code interfaces and using test doubles to isolate the code under test from other components.

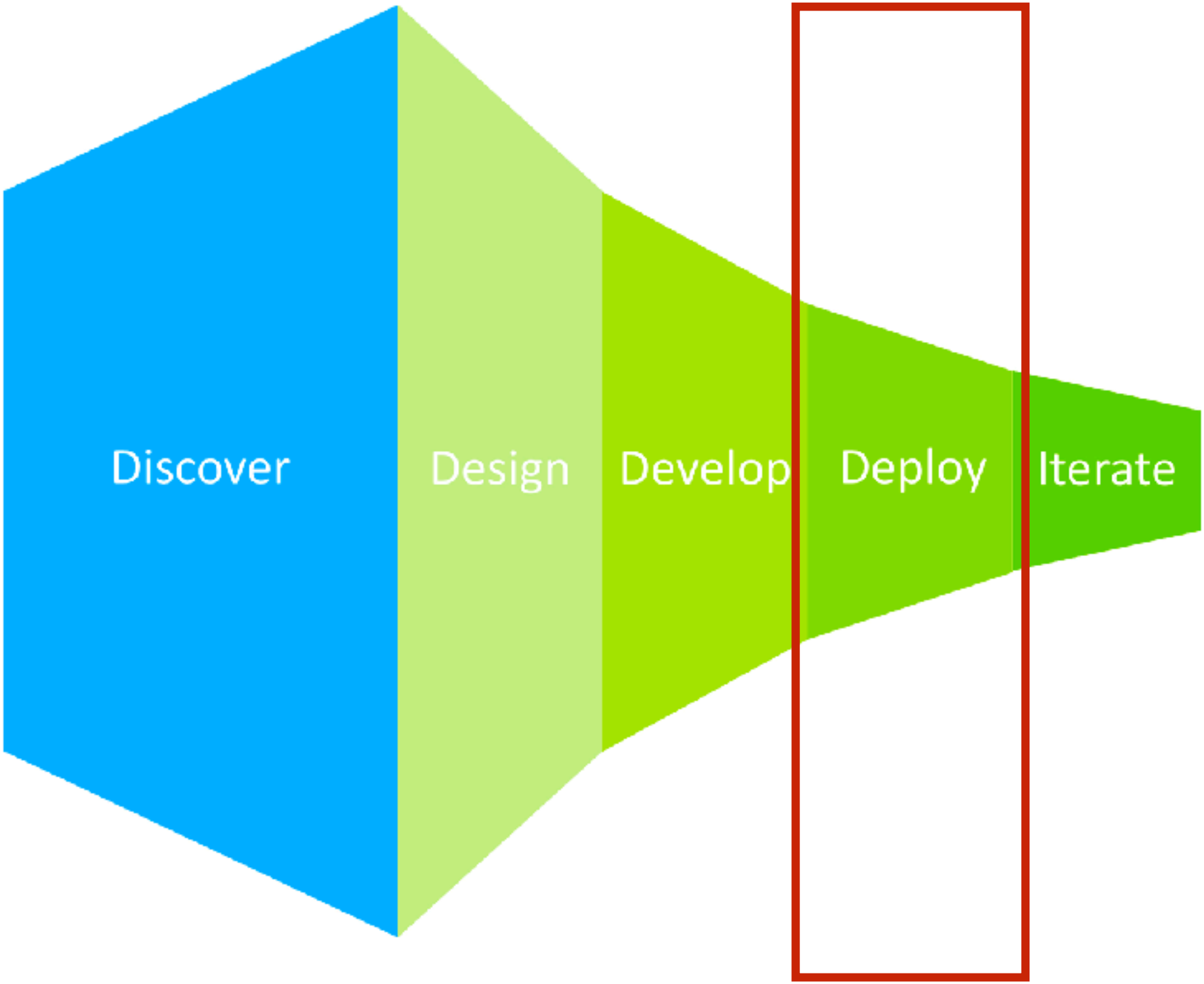
Contract tests : verify interactions at the boundary of an external service asserting that it meets the contract expected by a consuming service.

End-to-end tests : verify that a system meets external requirements and achieves its goals, testing the entire system, from end to end.

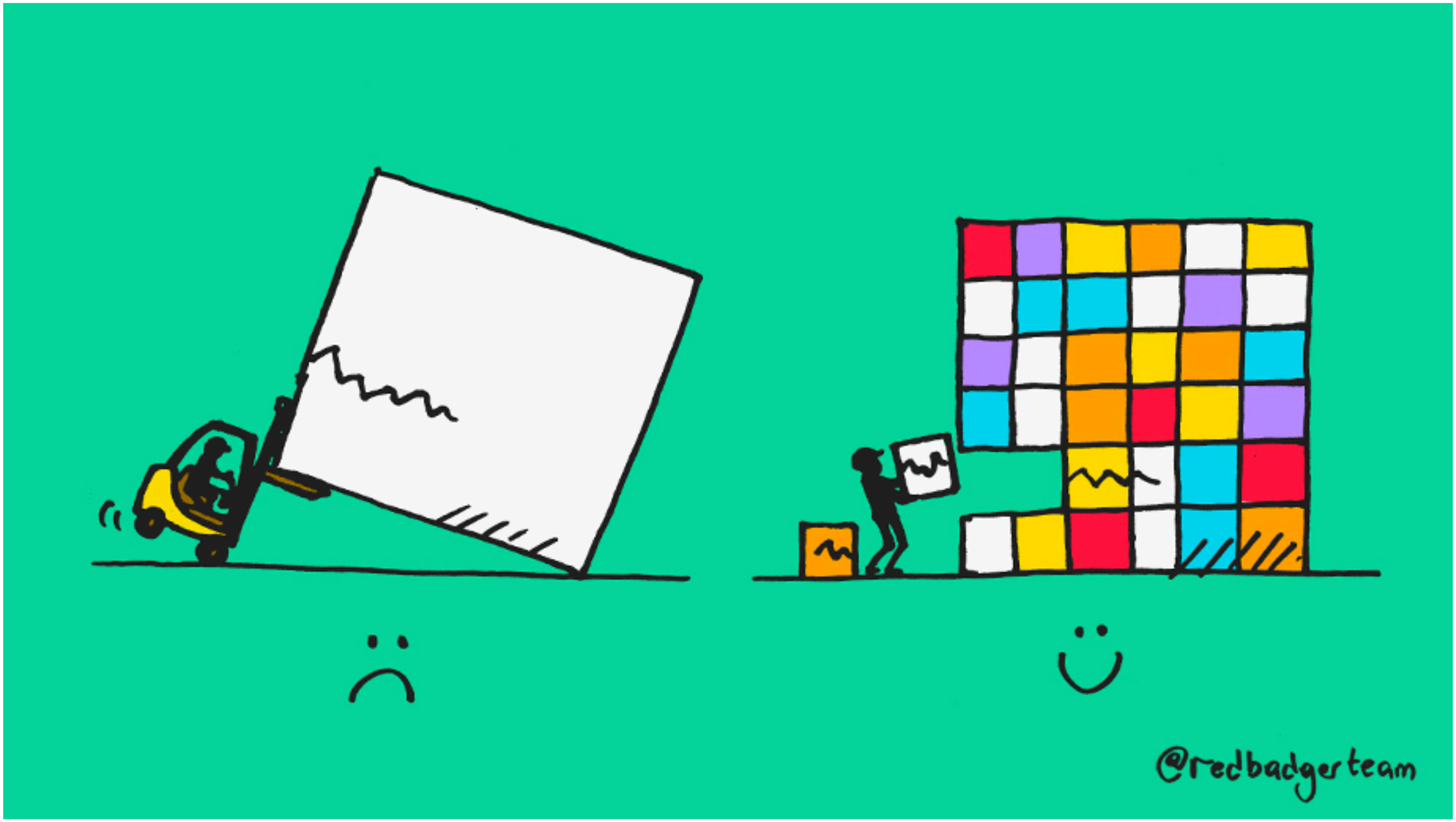
What is your testing strategy ?

Performance testing ?

Security testing ?



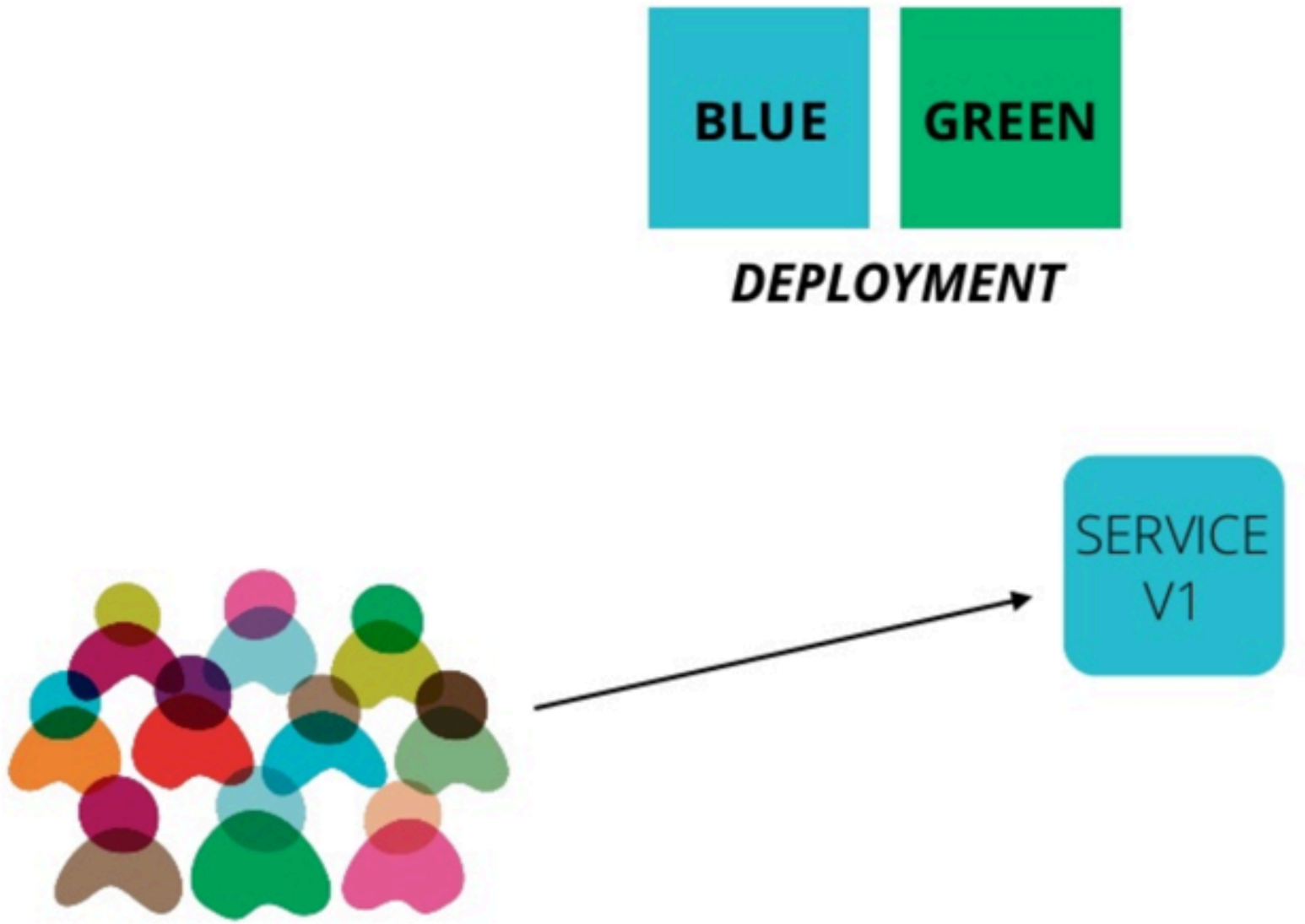
Deployment



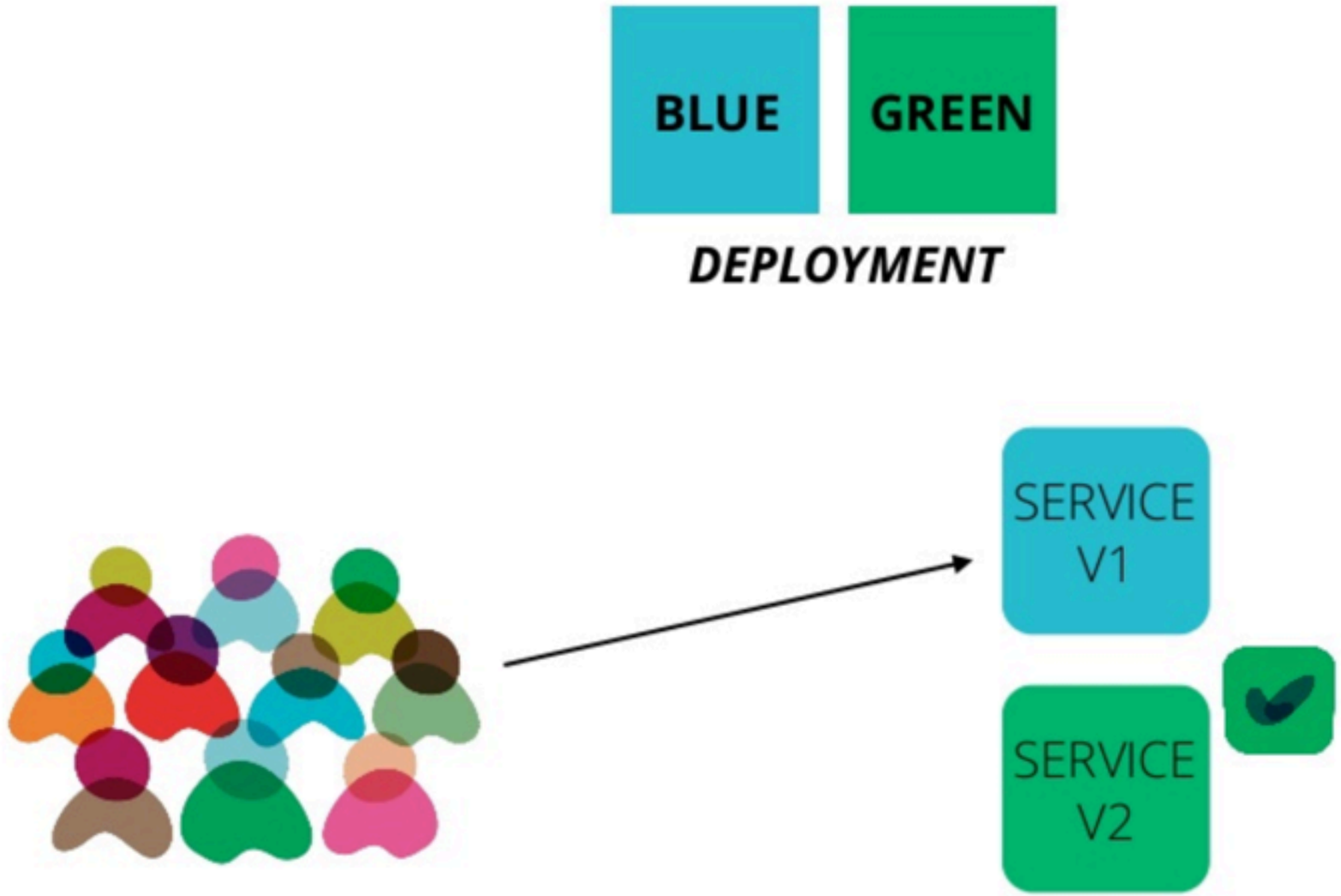
Deploy vs Release

Blue Green Deployment

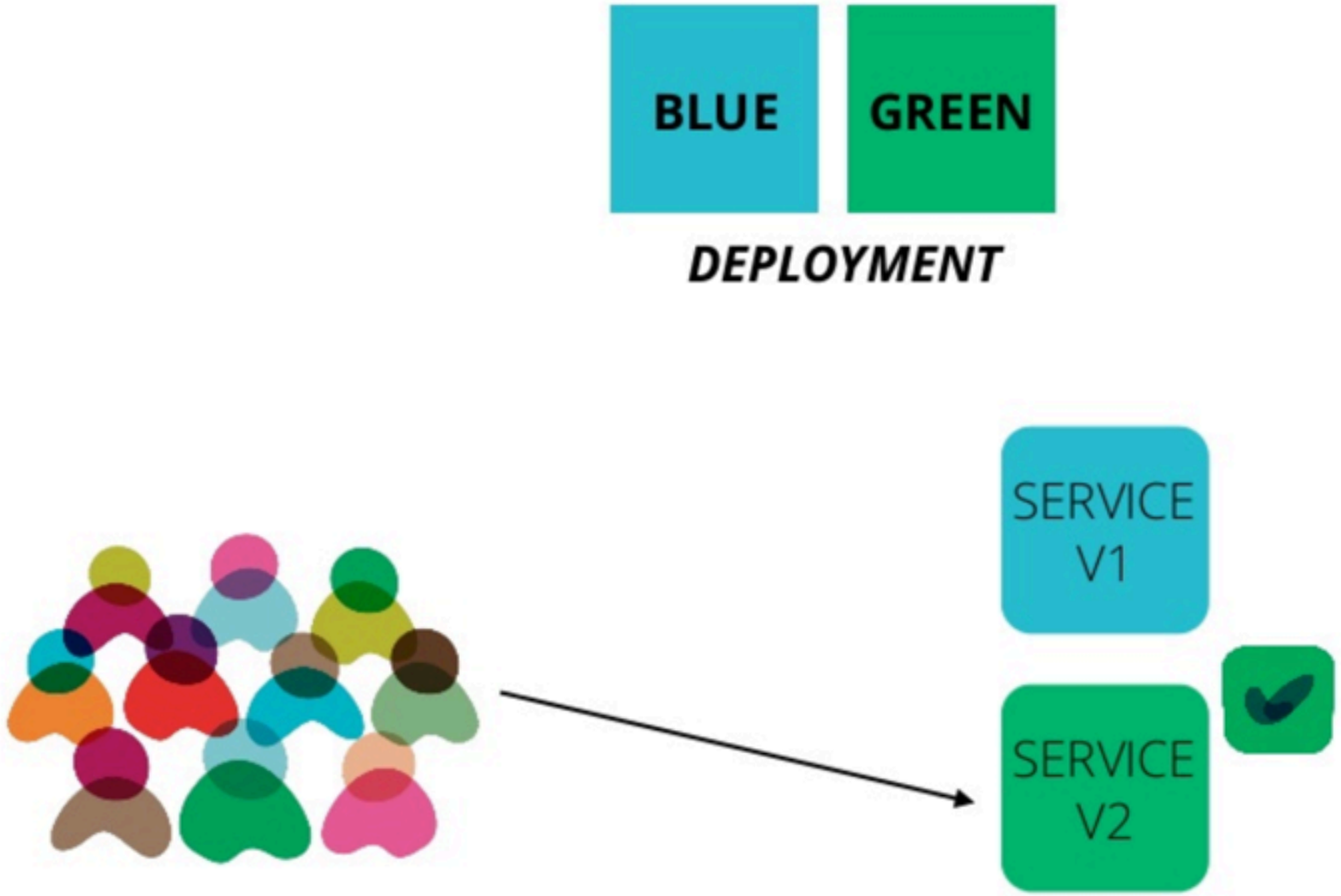
Blue Green Deployment



Blue Green Deployment



Blue Green Deployment

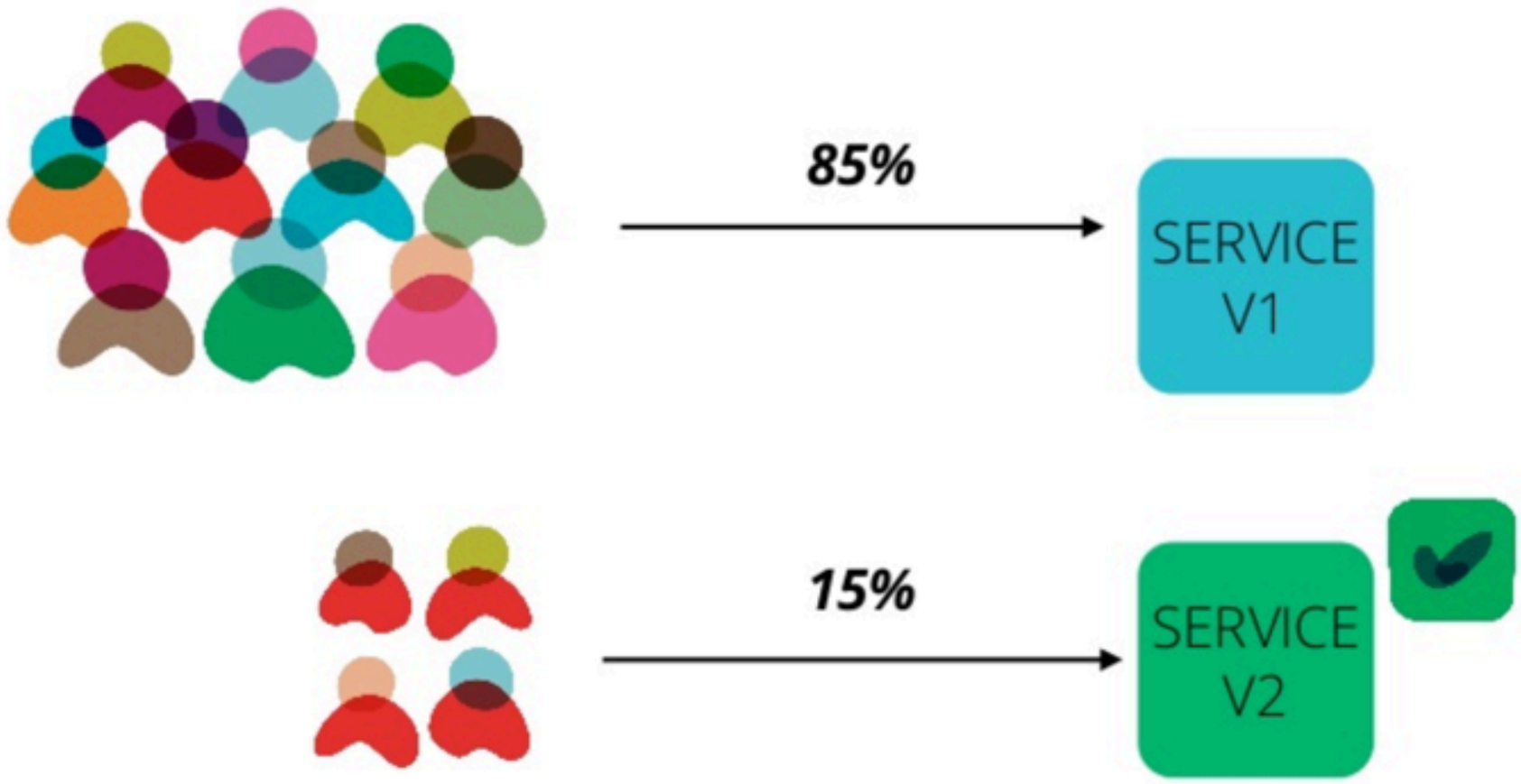


Canary Release

Canary Release

Canary Release

Canary Release



Mean Time to Recover (MTTR)

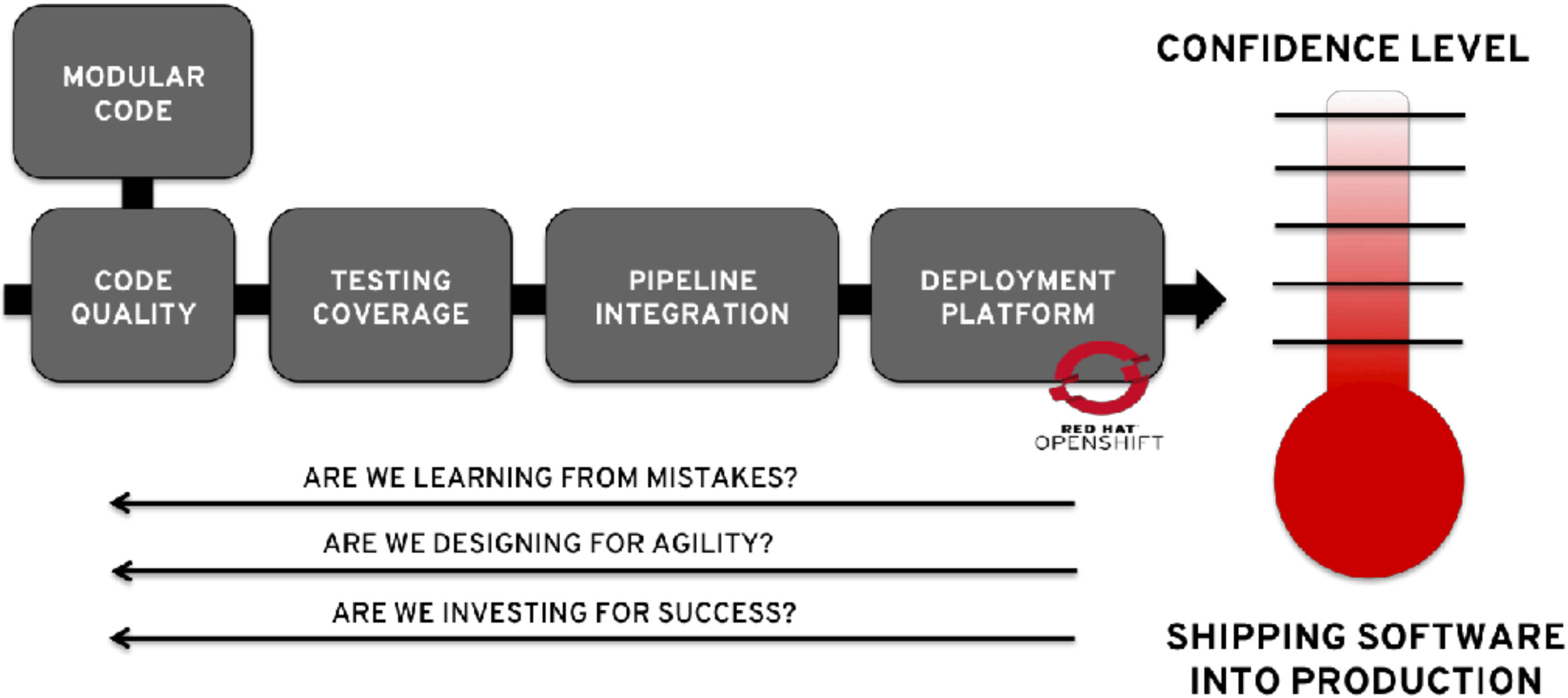
Mean Time to Recover (MTTR)

Tests are very important to reduce amount of defects in your systems. However, it's important to acknowledge that bugs will always happen in production.

Mean Time to Recover (MTTR)

How **fast to recover** from them will help determining our success !

What can i measure ?

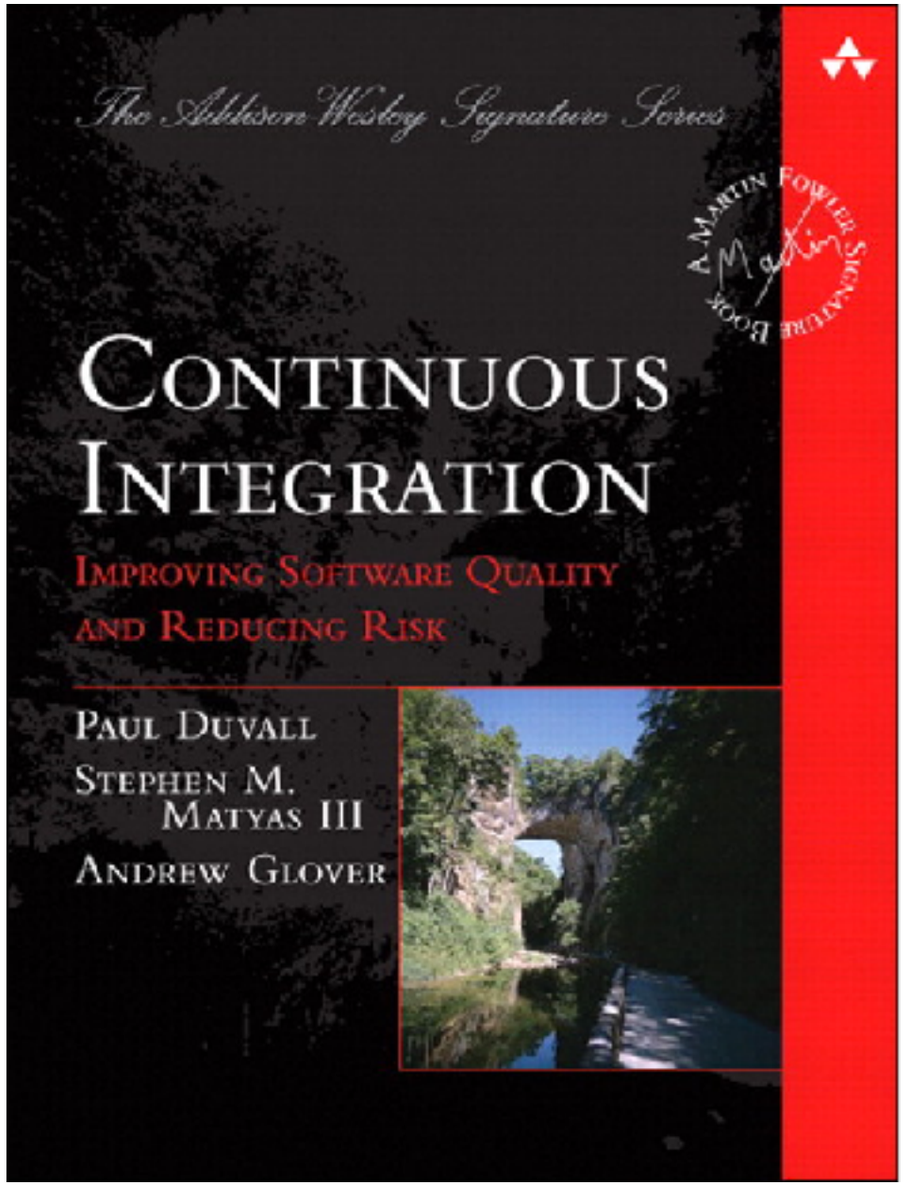


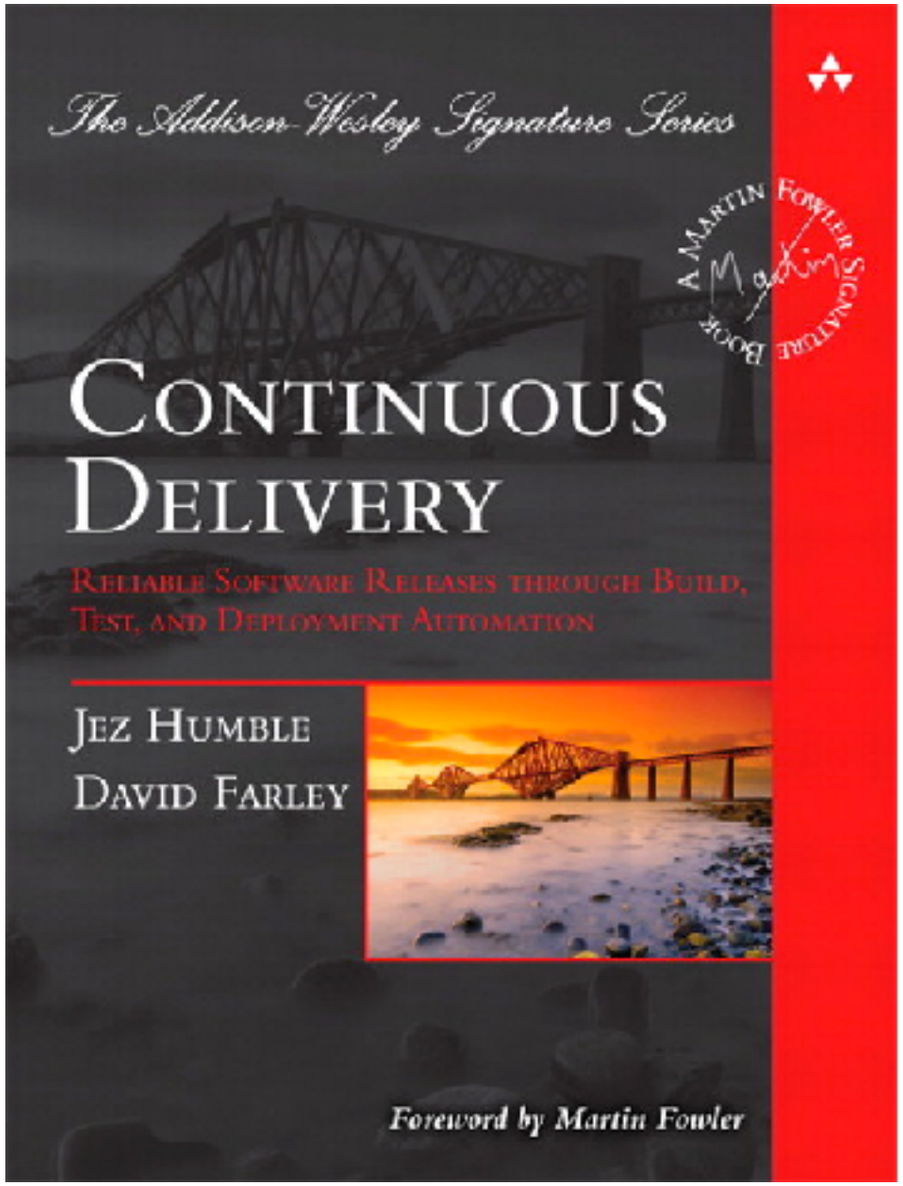
Start with Continuous Integration Continuous Delivery

“Behind every successful agile project, there is a

Continuous Integration Server”

Improve quality and reduce risk

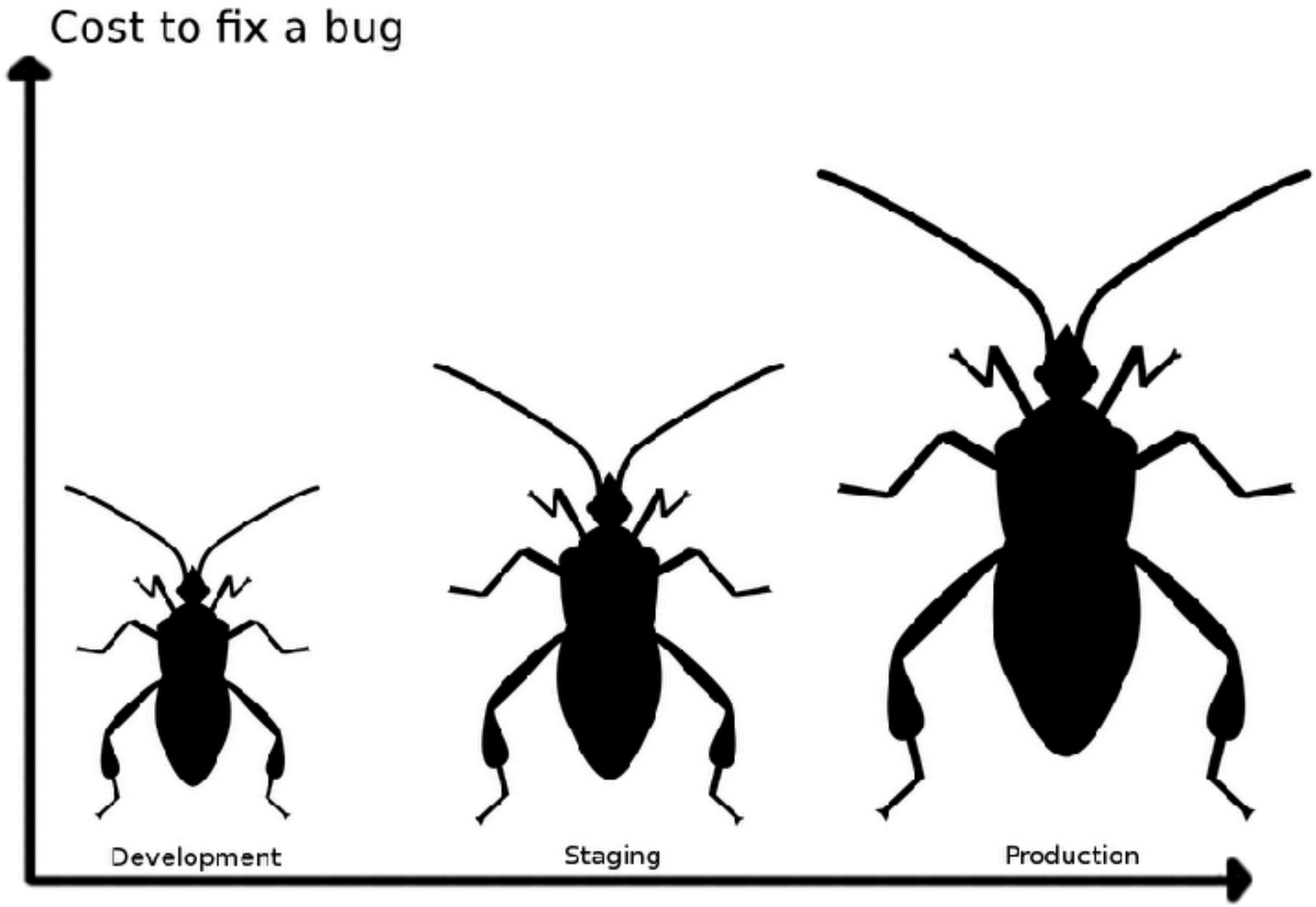


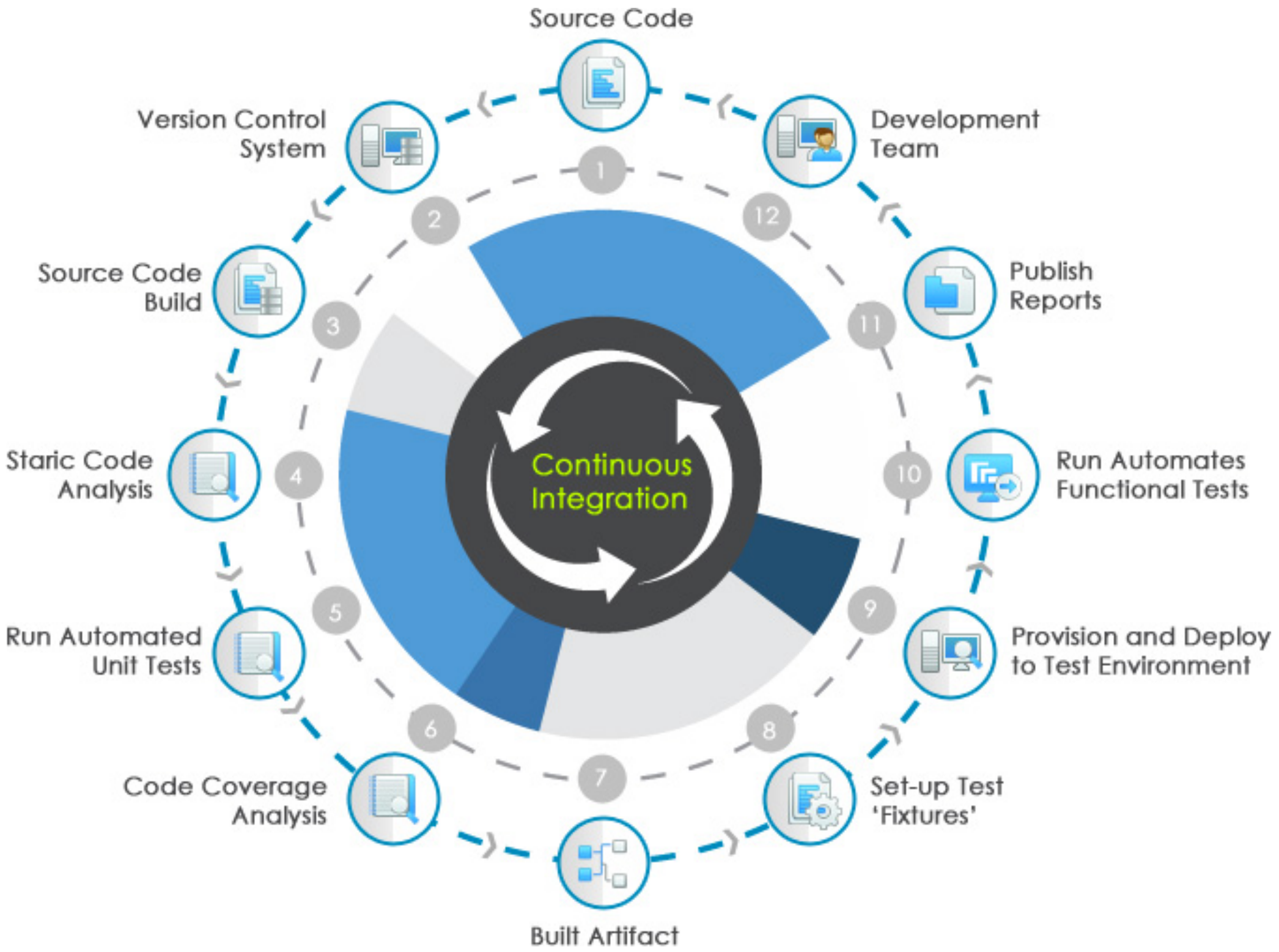


The cost of integration

1. Merging the code
2. Duplicate changes
3. Test again again !!
4. Fixing bugs
5. Impact on stability

The cost of integration





Jenkins

Bamboo

TeamCity

go

Visual Studio Team Foundation Server

Hudson

Travis

wercker

circleci

Jenkins

Bamboo

TeamCity

Visual Studio Team Foundation Server

Hudson

Travis

wercker

circleci

CI is about what people do not about what tools they use

Continuous Integration

Discipline to **integrate frequently**

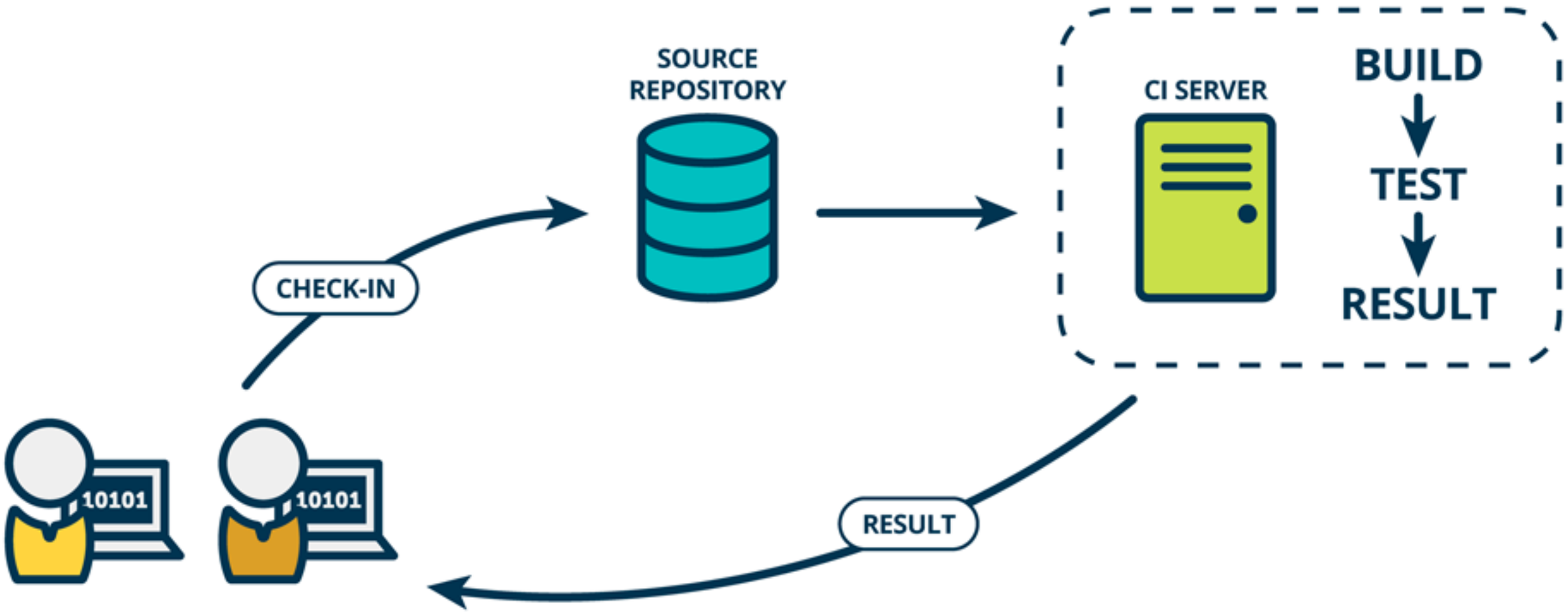
Continuous Integration

Strive to make **small change**

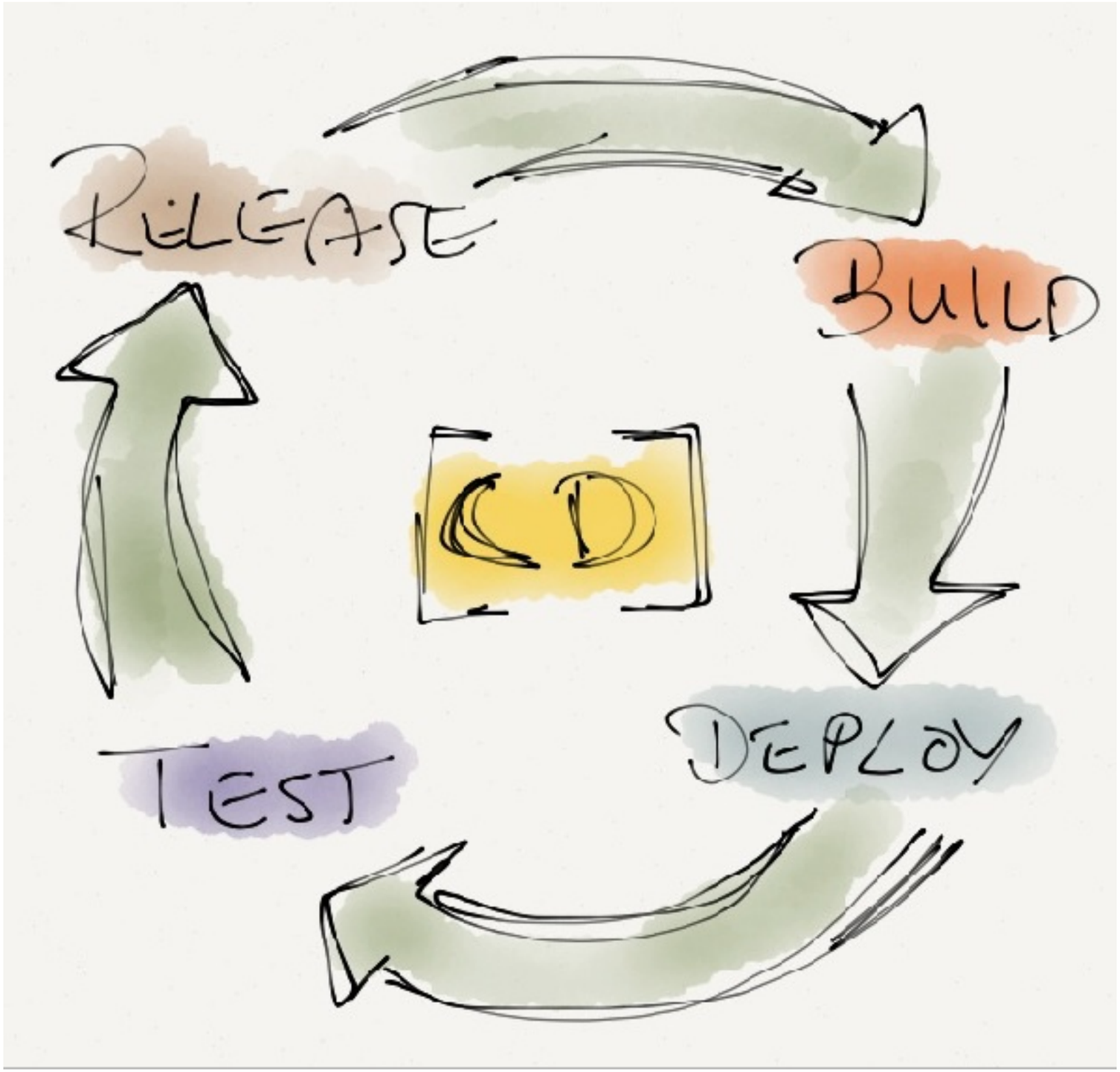
Continuous Integration

Strive for **fast feedback**

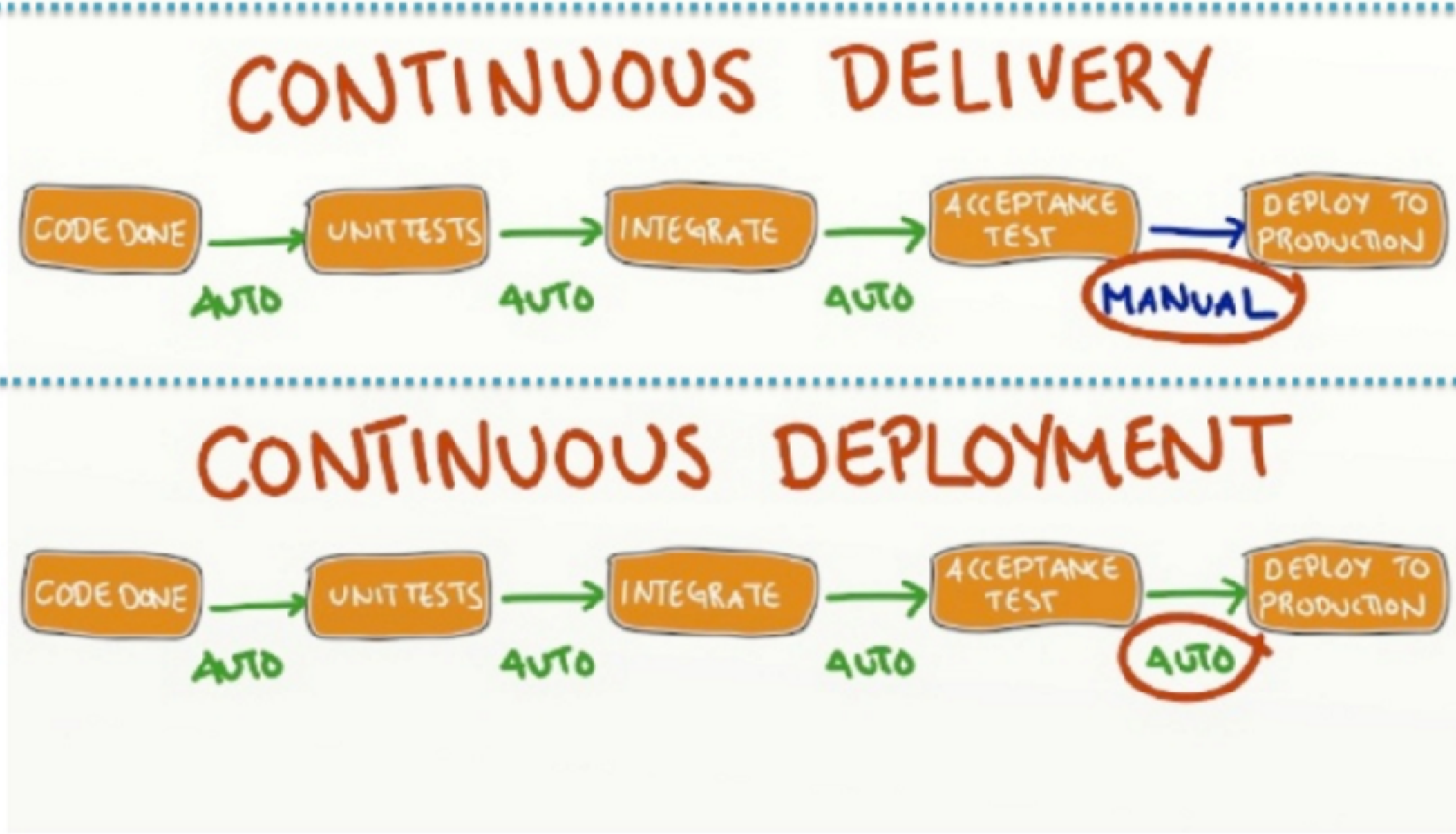
Continuous Integration



CD ?

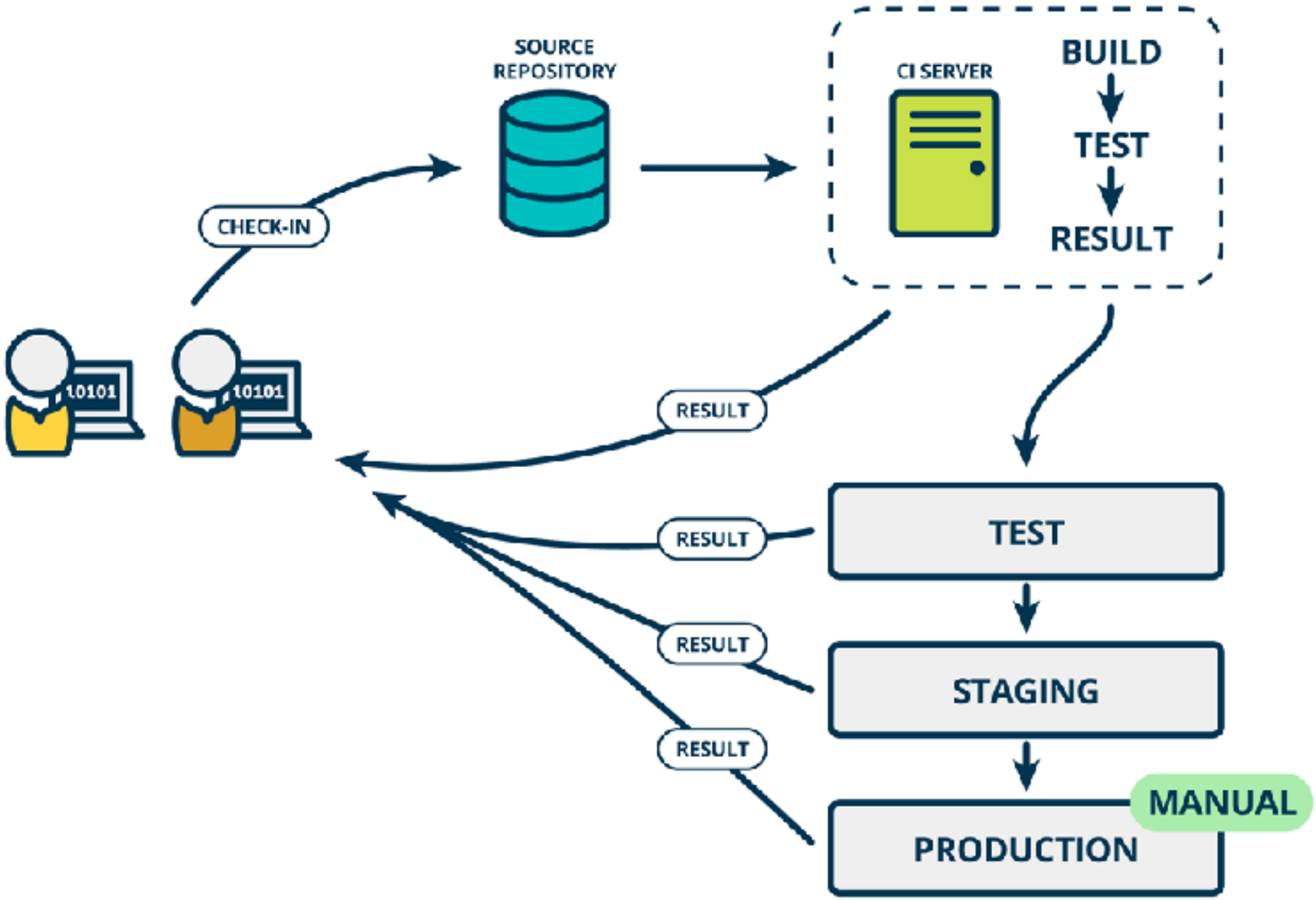


CD ?

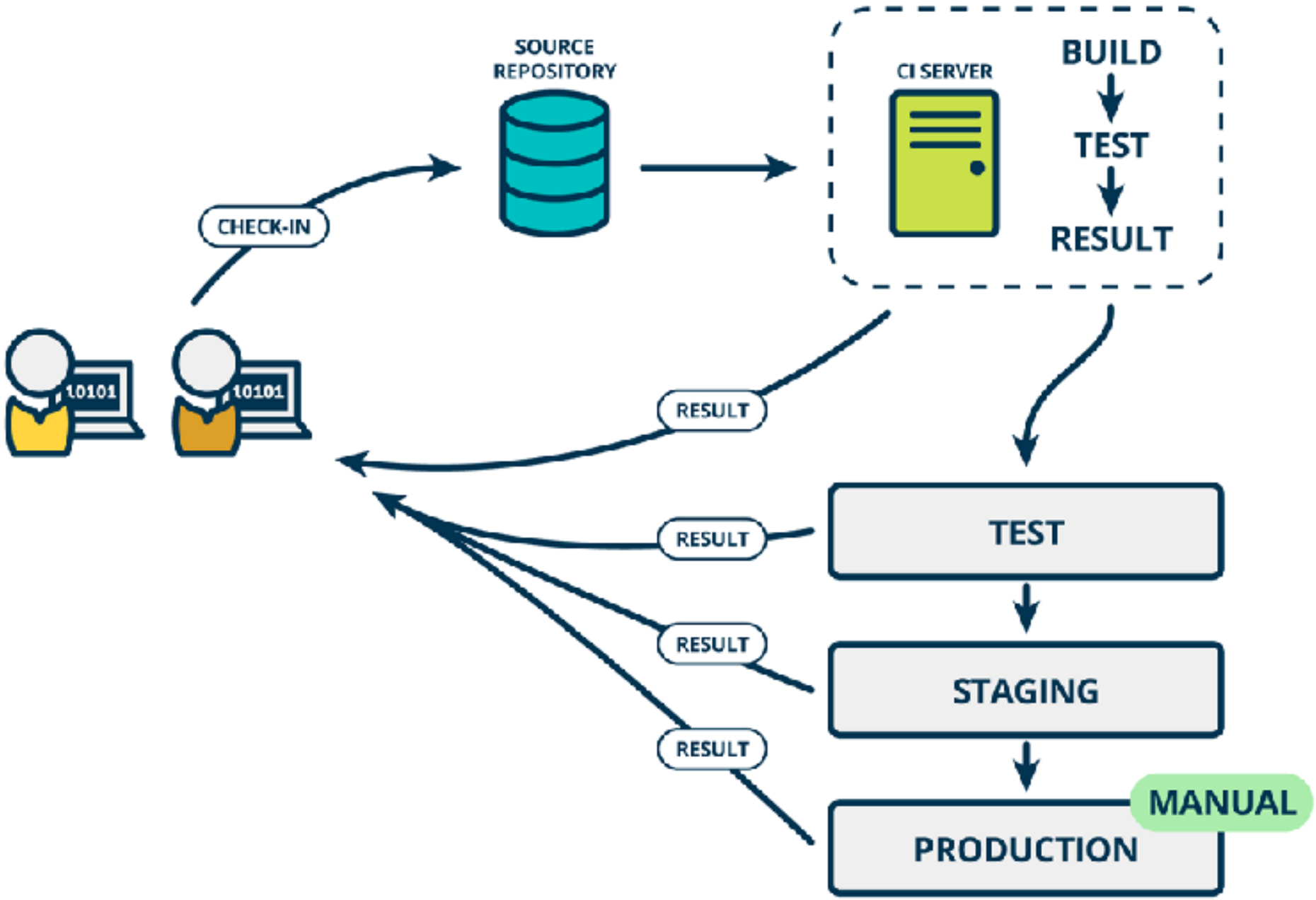


http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment

Continuous Delivery



Rise of DevOps



Continuous Integration

is a Software development **practices**

Practice 1

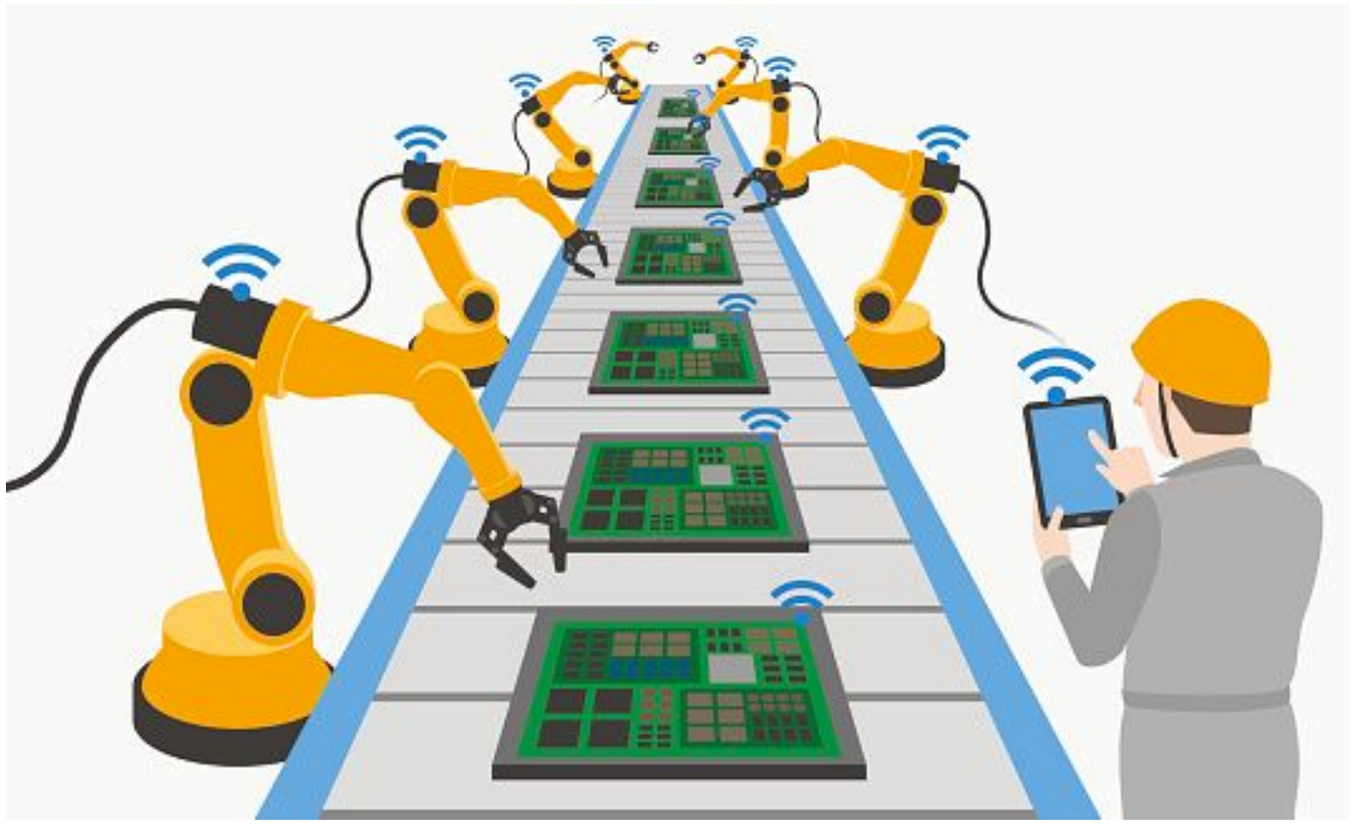
Maintain a single source repository

In general, you should store in source control everything you need to build anything

Practice 2

Automated the build

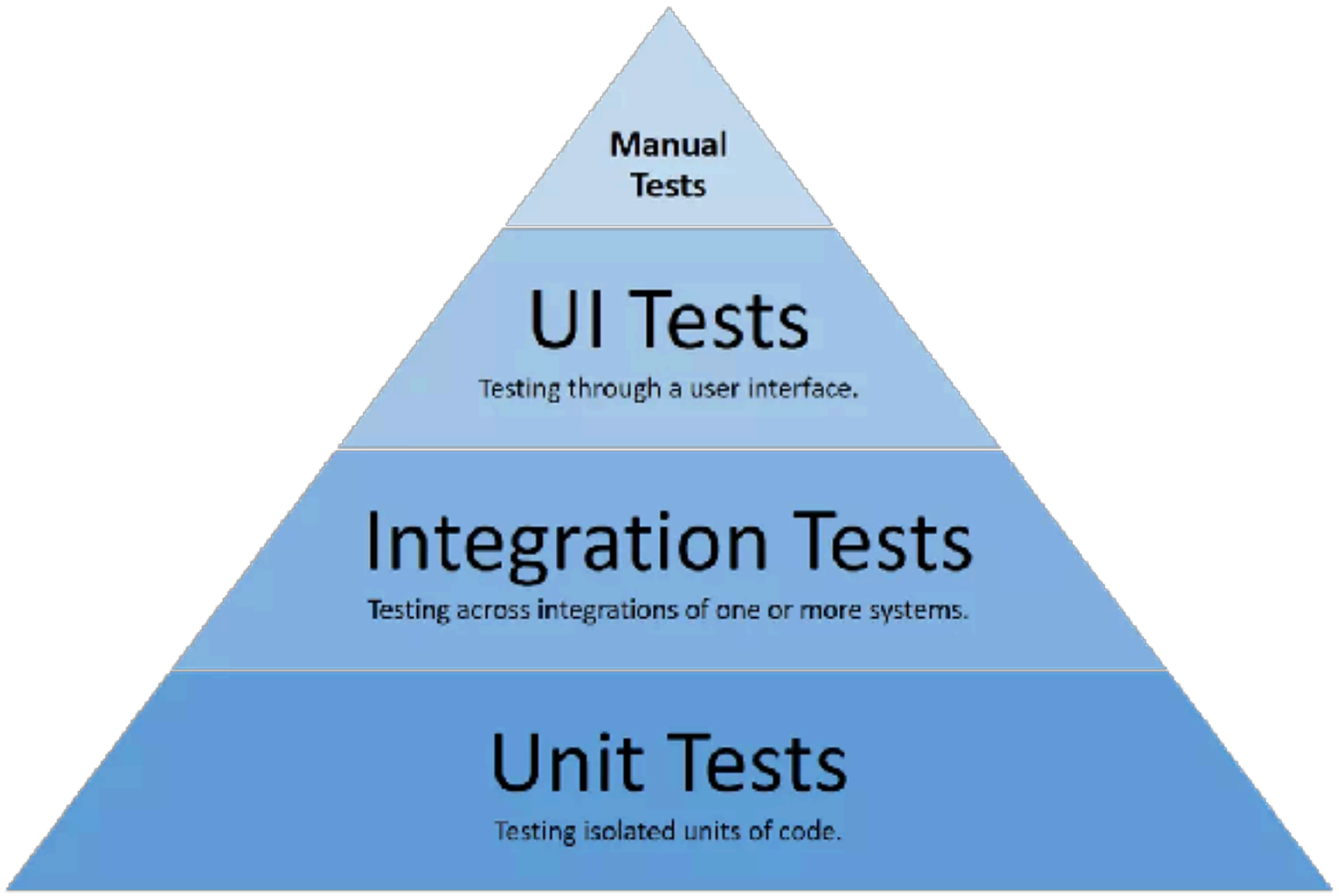
Automated environment for builds



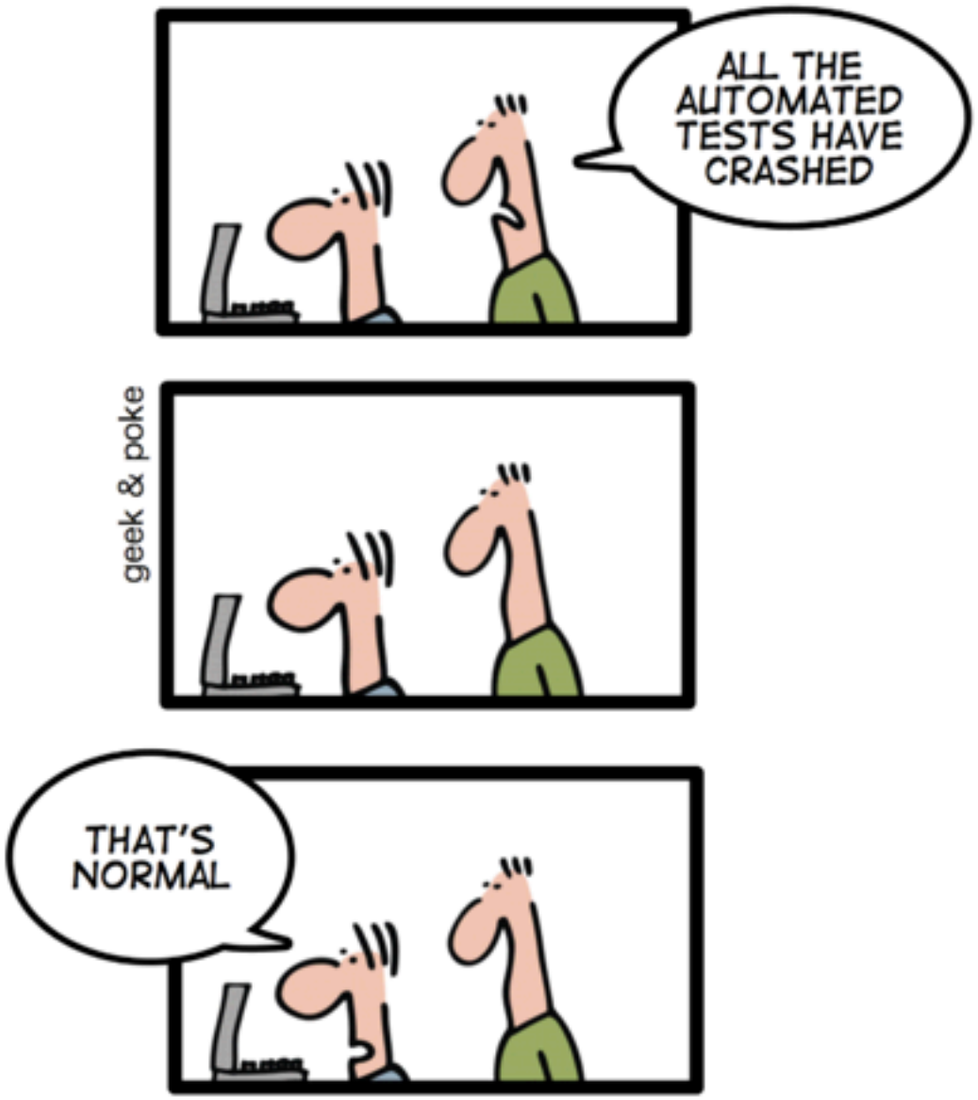
Practice 3

Make your build **self-testing**

Build process => compile, linking and **testing**



TODAY: CONTINUOUS INTEGRATION GIVES YOU THE COMFORTING FEELING TO KNOW THAT EVERYTHING IS NORMAL

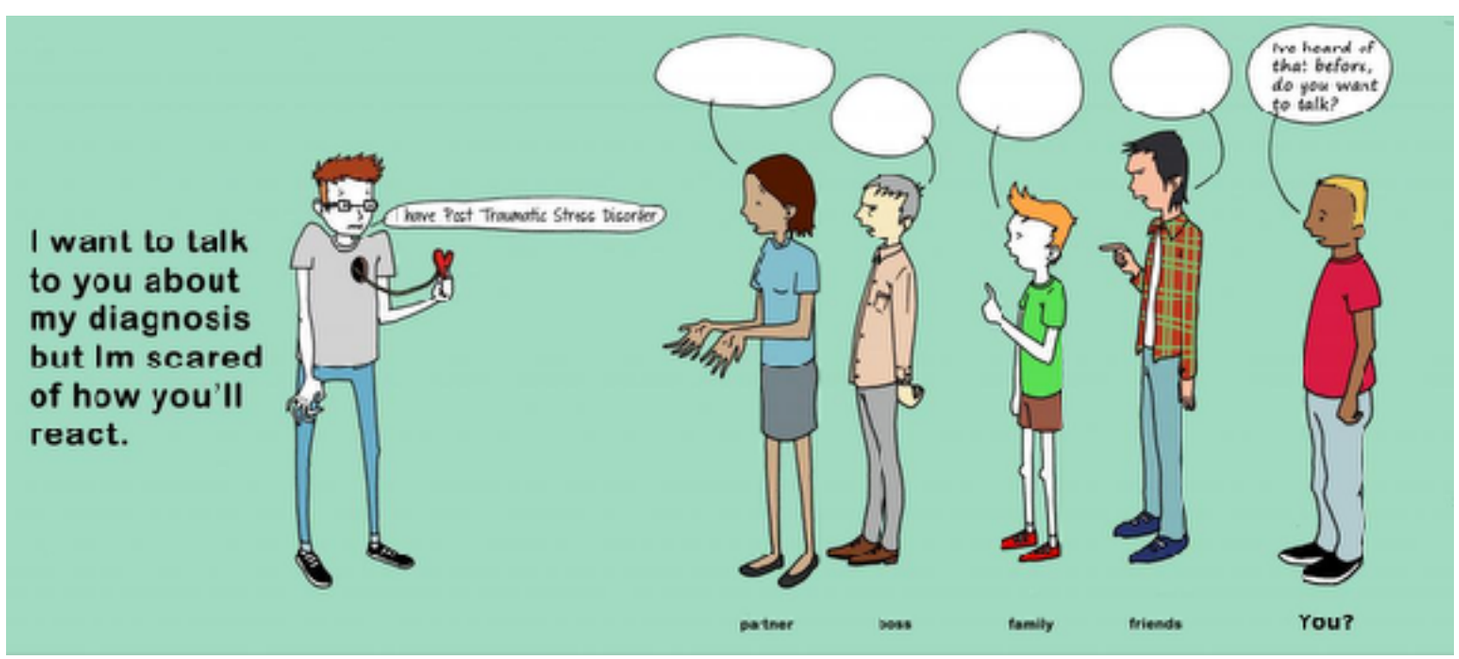


Practice 4

Everyone **commits** to the mainline everyday

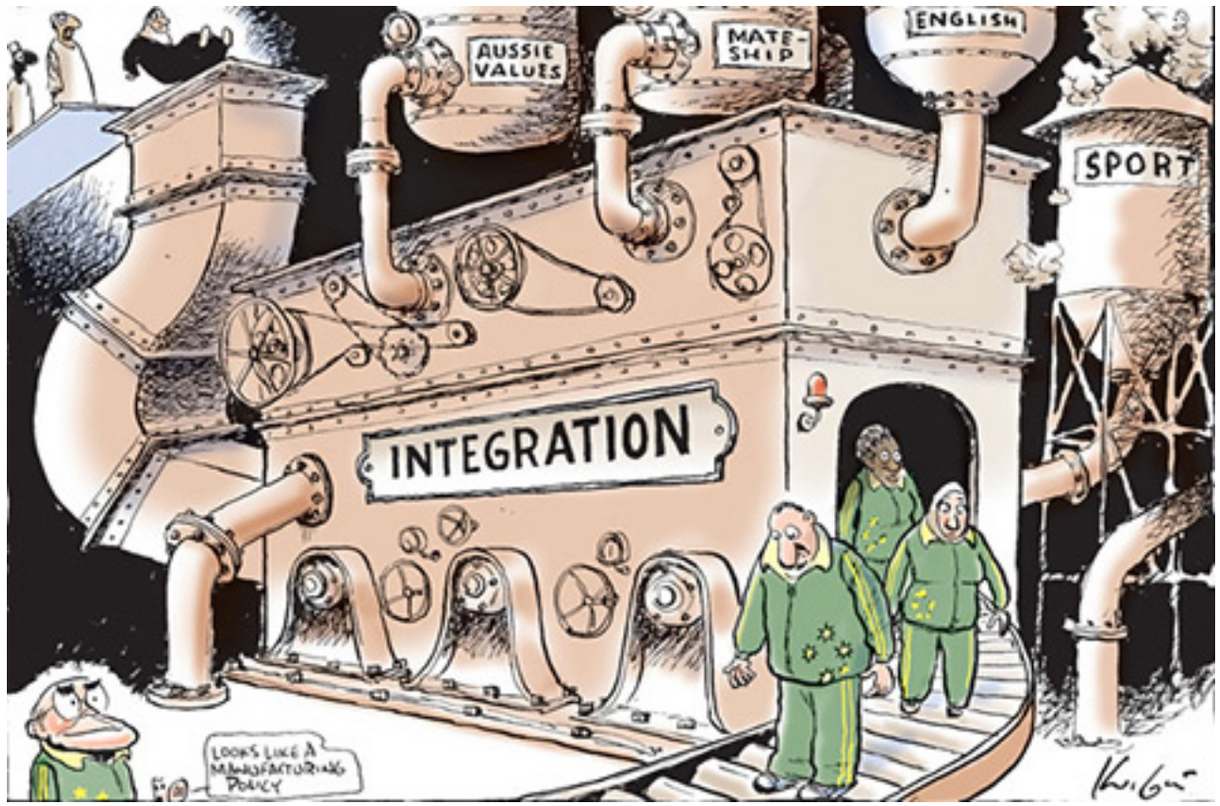
Integration is about **communication**

Integration allows developers to **tell** other developers



Practice 5

Every commits should build the mainline on an **Integration machine**



Nightly build is not enough for Continuous Integration

Practice 6

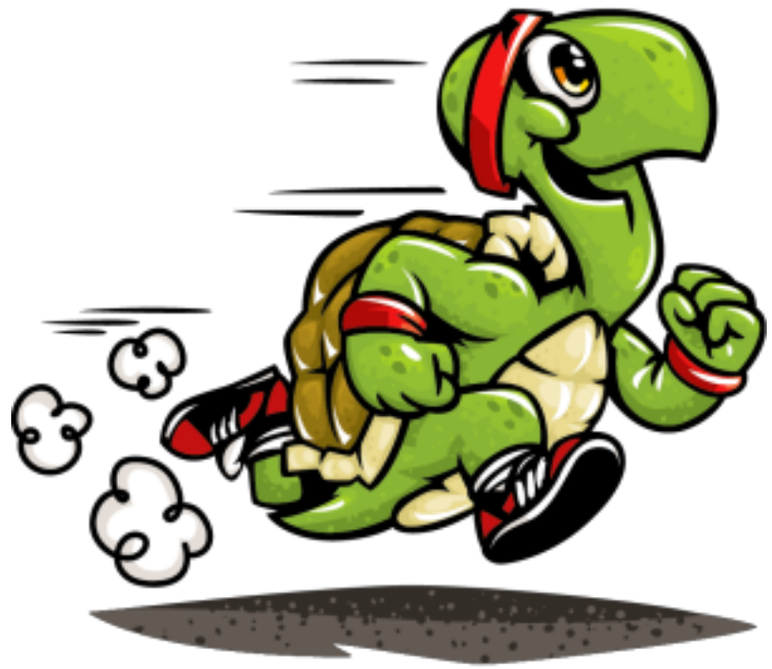
Fix broken builds **immediately**

"Nobody has a higher priority task than fixing the build"

Practice 7

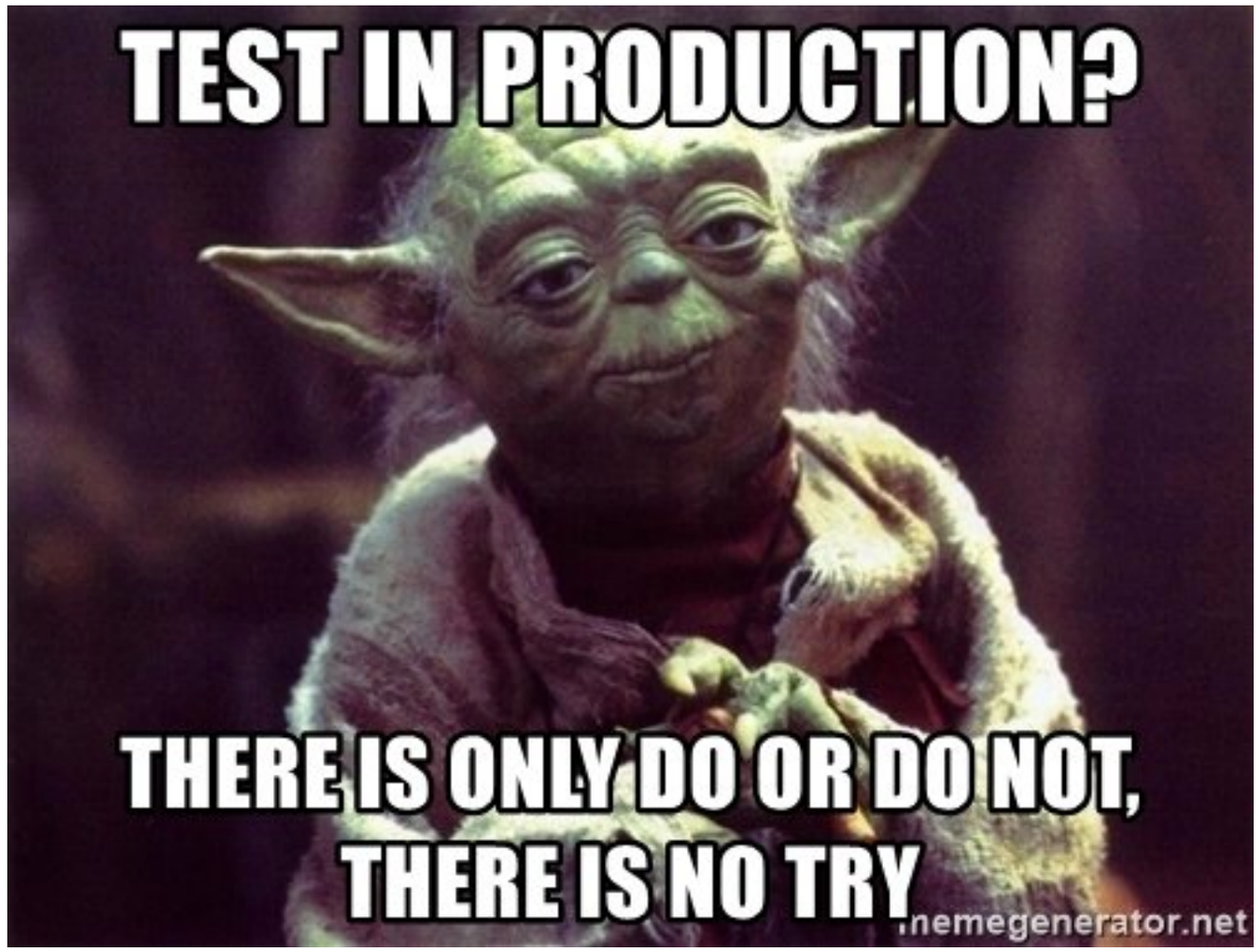
Keep the build **fast**

Continuous Integration is to provide rapid feedback



Practice 8

Test in clone of the **Production** environment



Practice 9

Make it easy for anyone to get
the latest executable

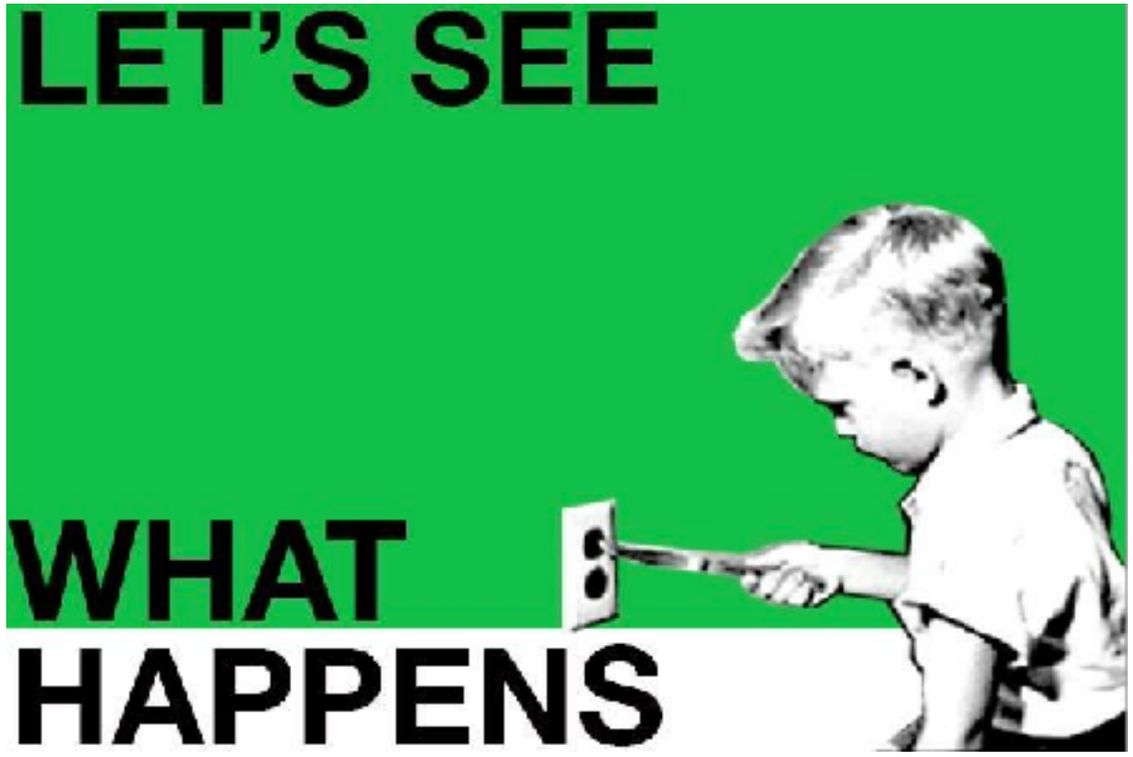
Make sure well known place where people can find

Practice 10

Everyone can see what's happening

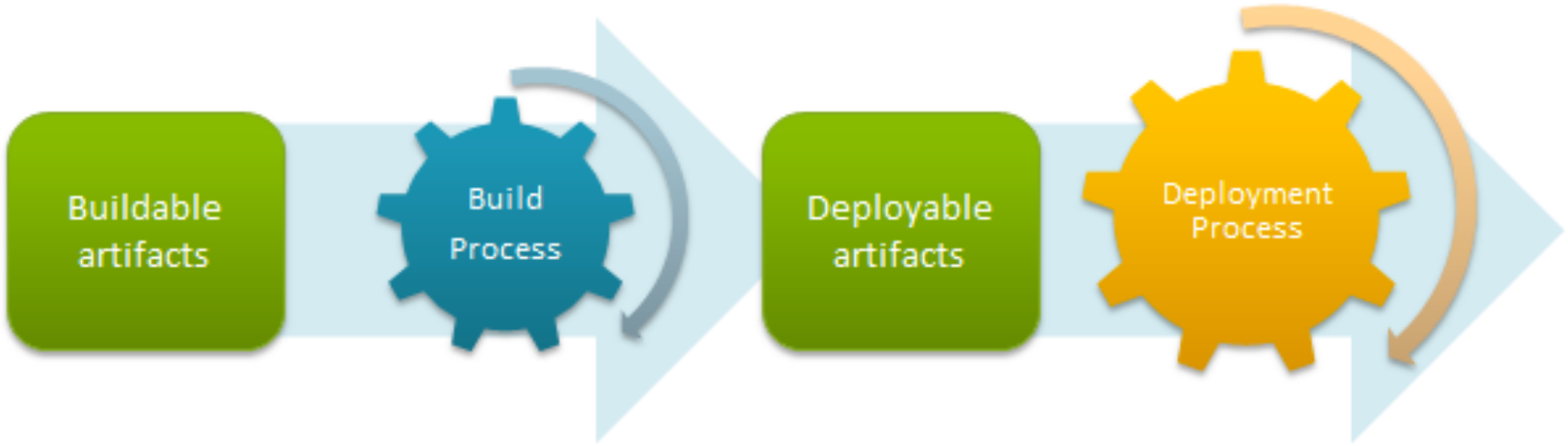
Easier to see the state of the system and changes

Show the good information



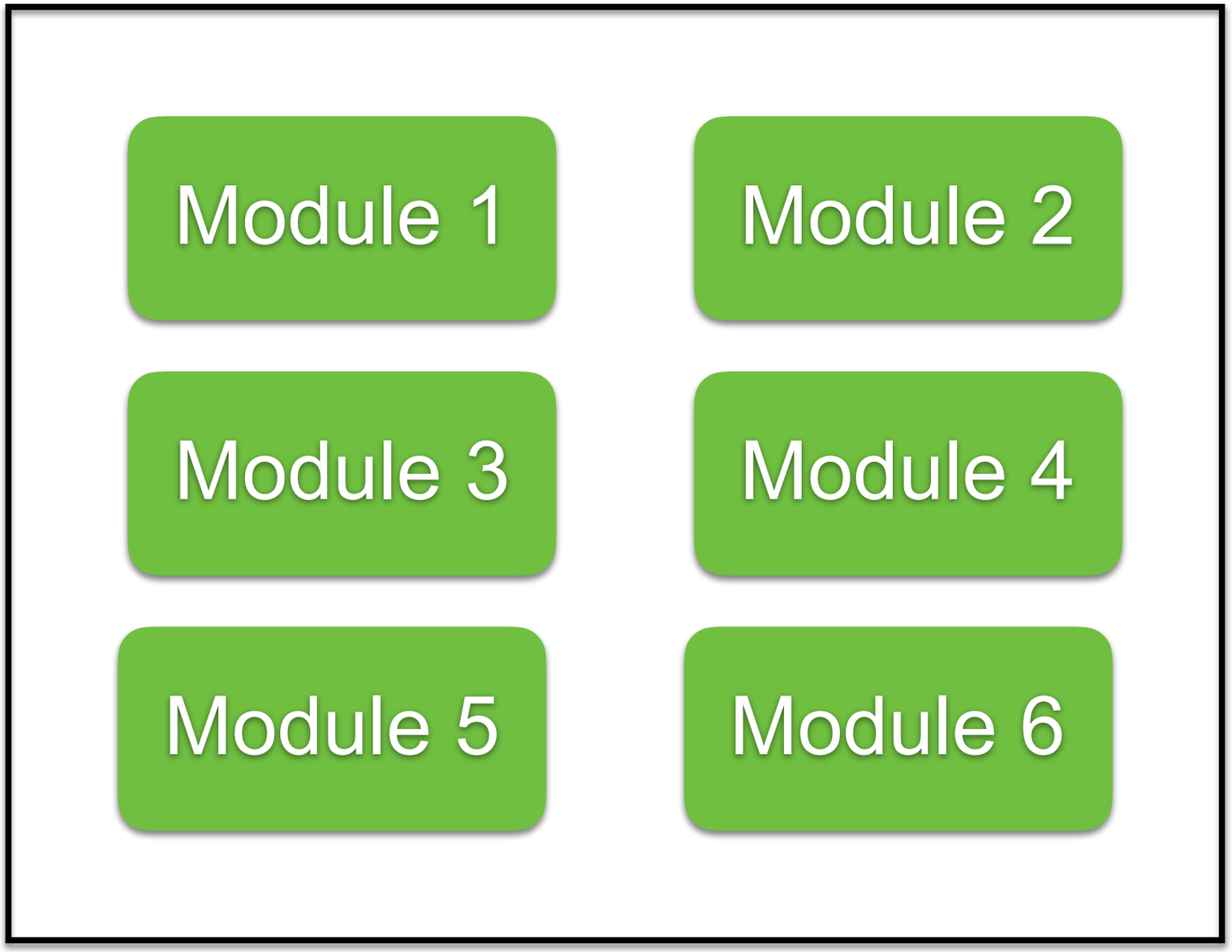
Practice 11

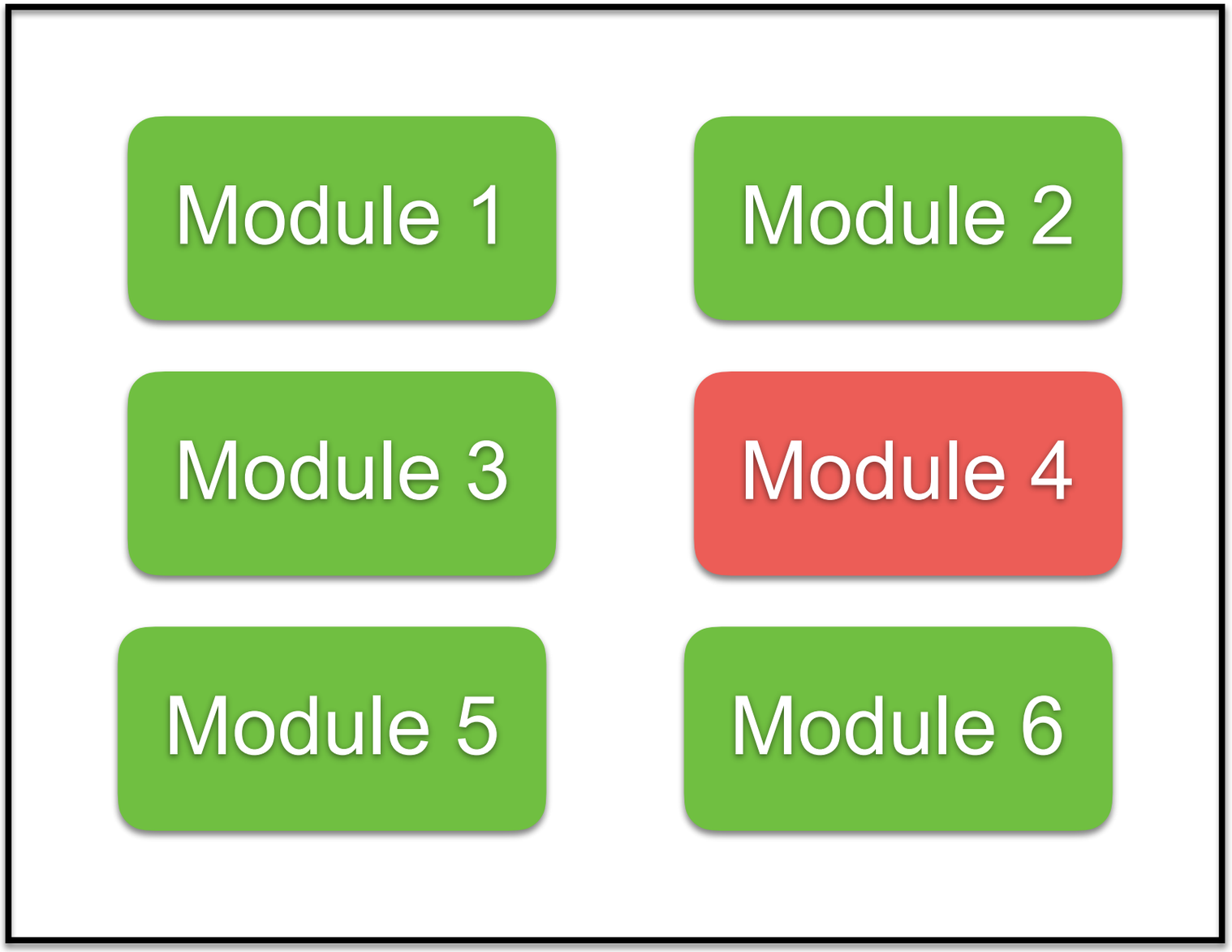
Automated deployment

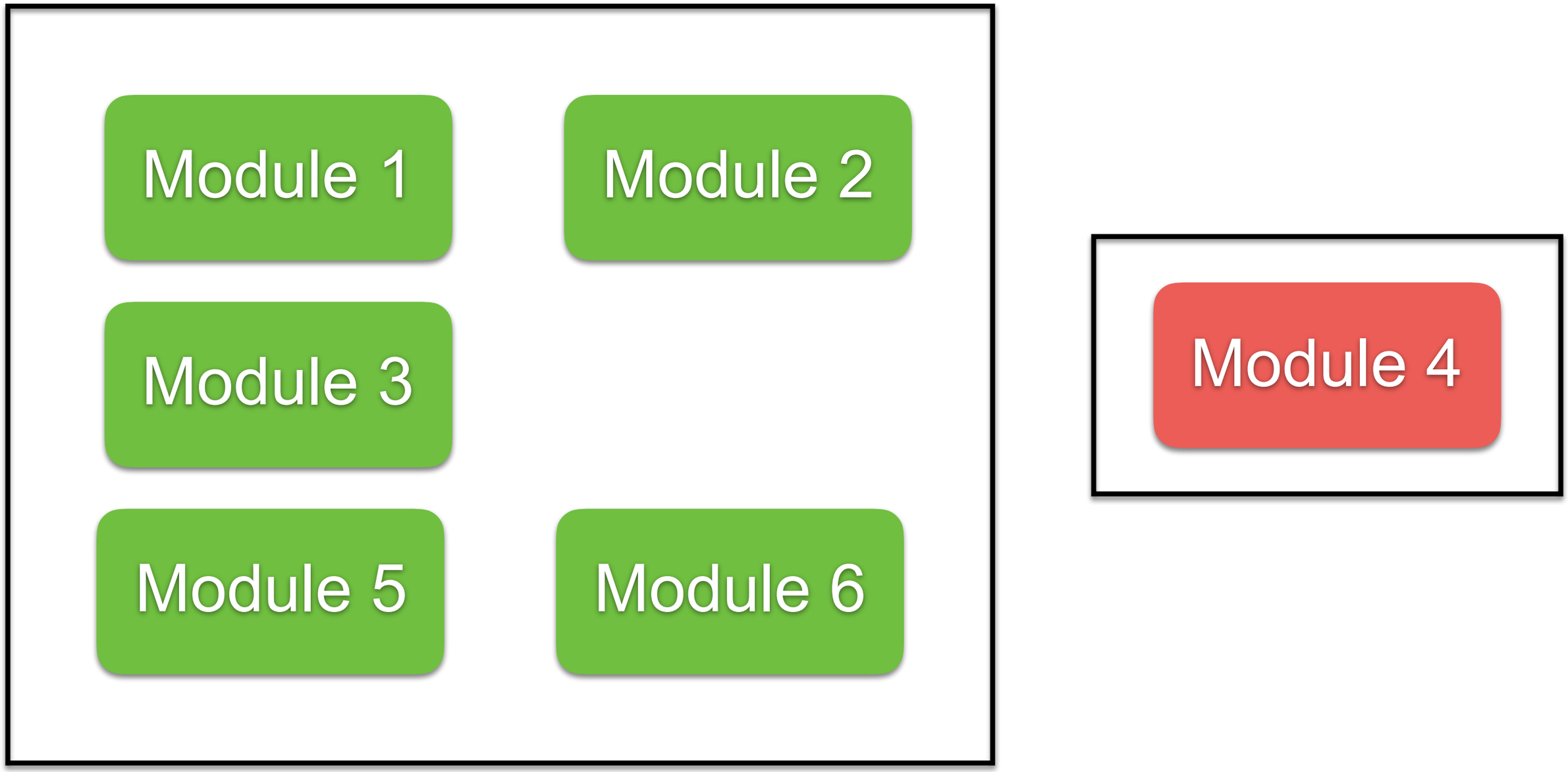


Summary

Let’s start with good monolith







Don’ t start with separate into small services

Key success factors

Monitoring and health check of service and infra

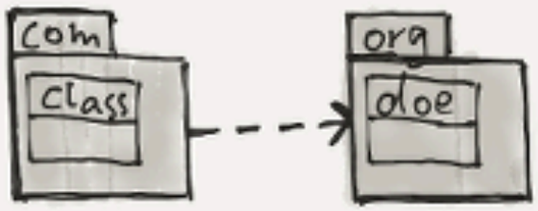
Scalable infrastructure for the services

Security design

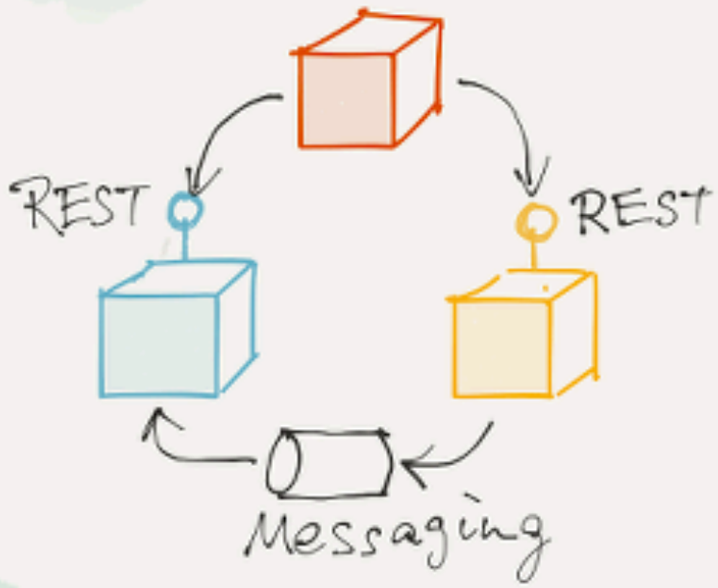
Rapid application delivery

CI/CD practices and infrastructure (DevOps)

Architecture



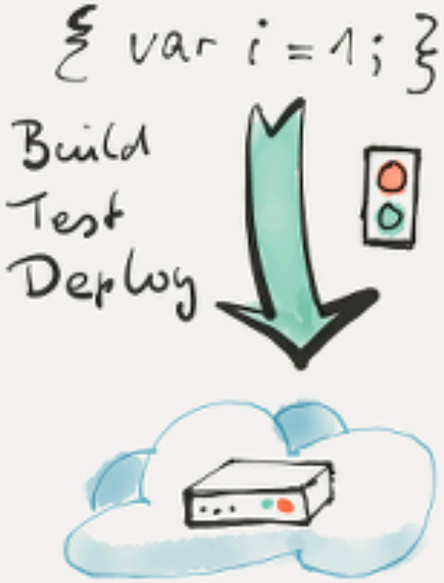
Microservices



People & Teams

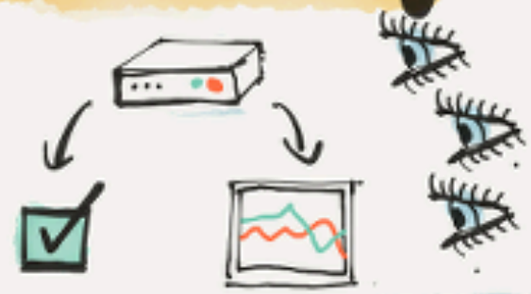
Deployment

Continous Delivery



Infrastructure

Monitoring



Microservices pitfalls

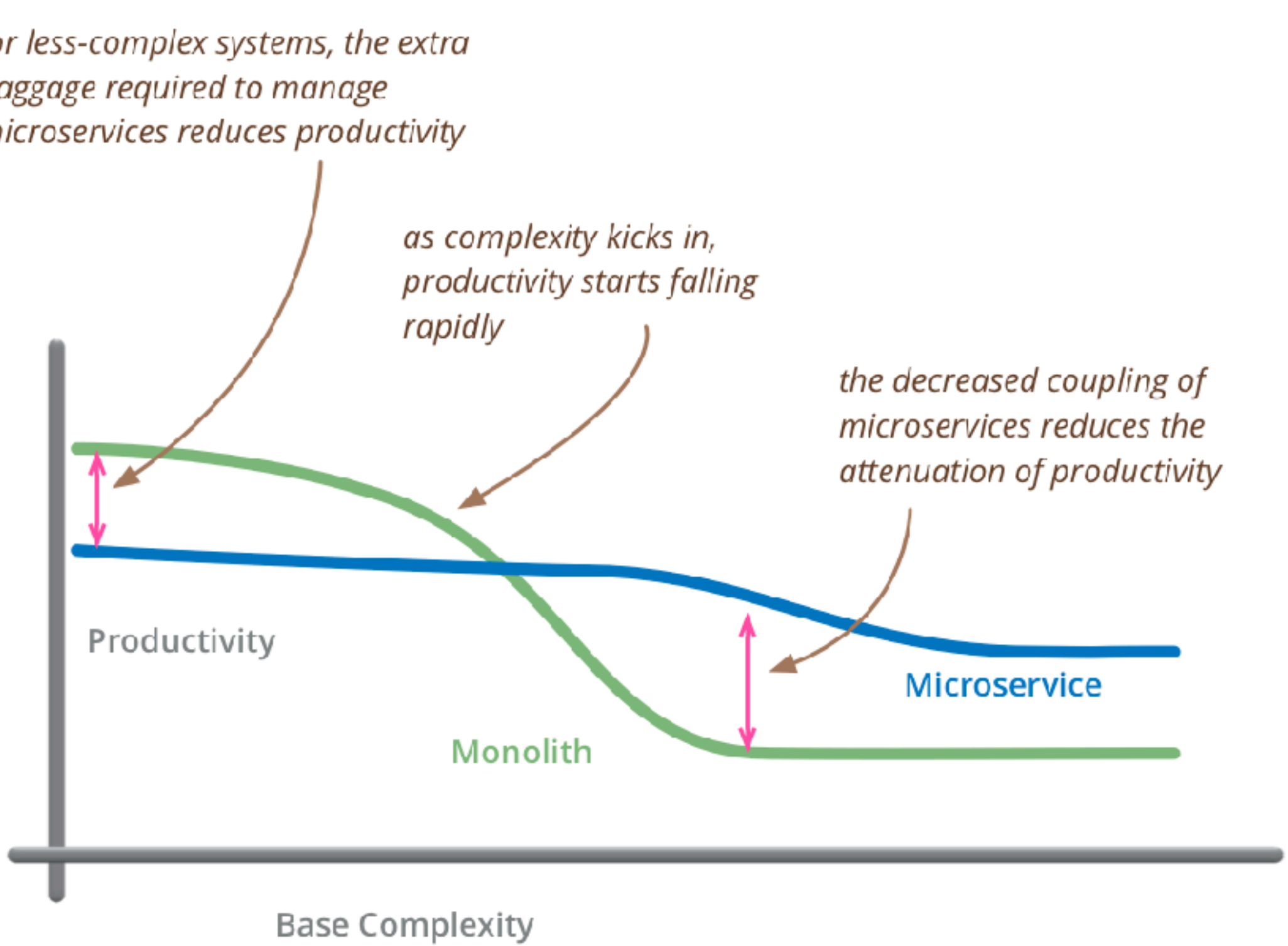
Microservices pitfalls

More/Low splitting
More network interaction
Data storing and sharing
Compatibility issues
Testing
Operation & Monitoring

Microservices pitfalls

More/Low splitting
More network interaction
Data storing and sharing
Compatibility issues
Testing
Operation & Monitoring

Productivity ?



but remember the skill of the team will outweigh any monolith/microservice choice