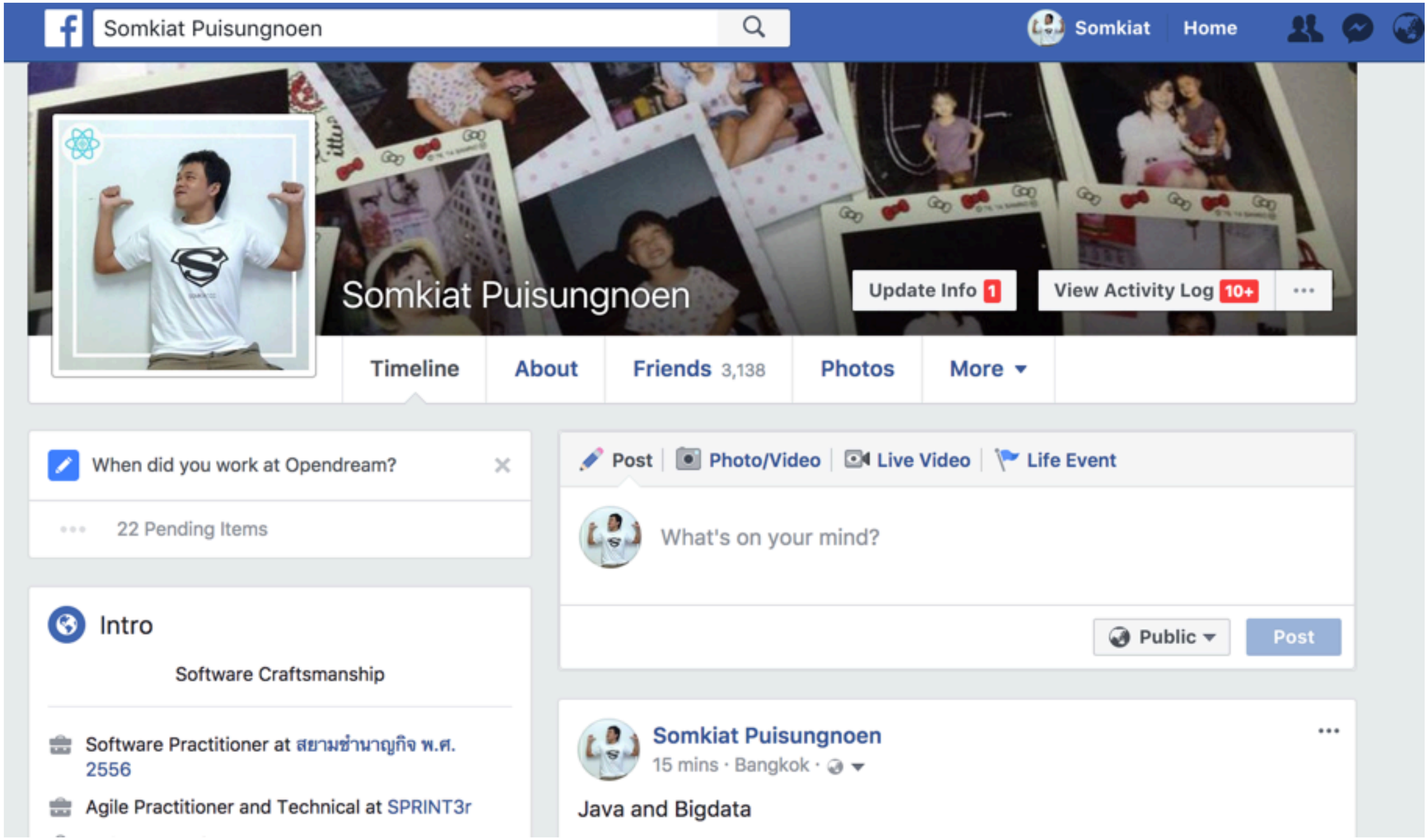
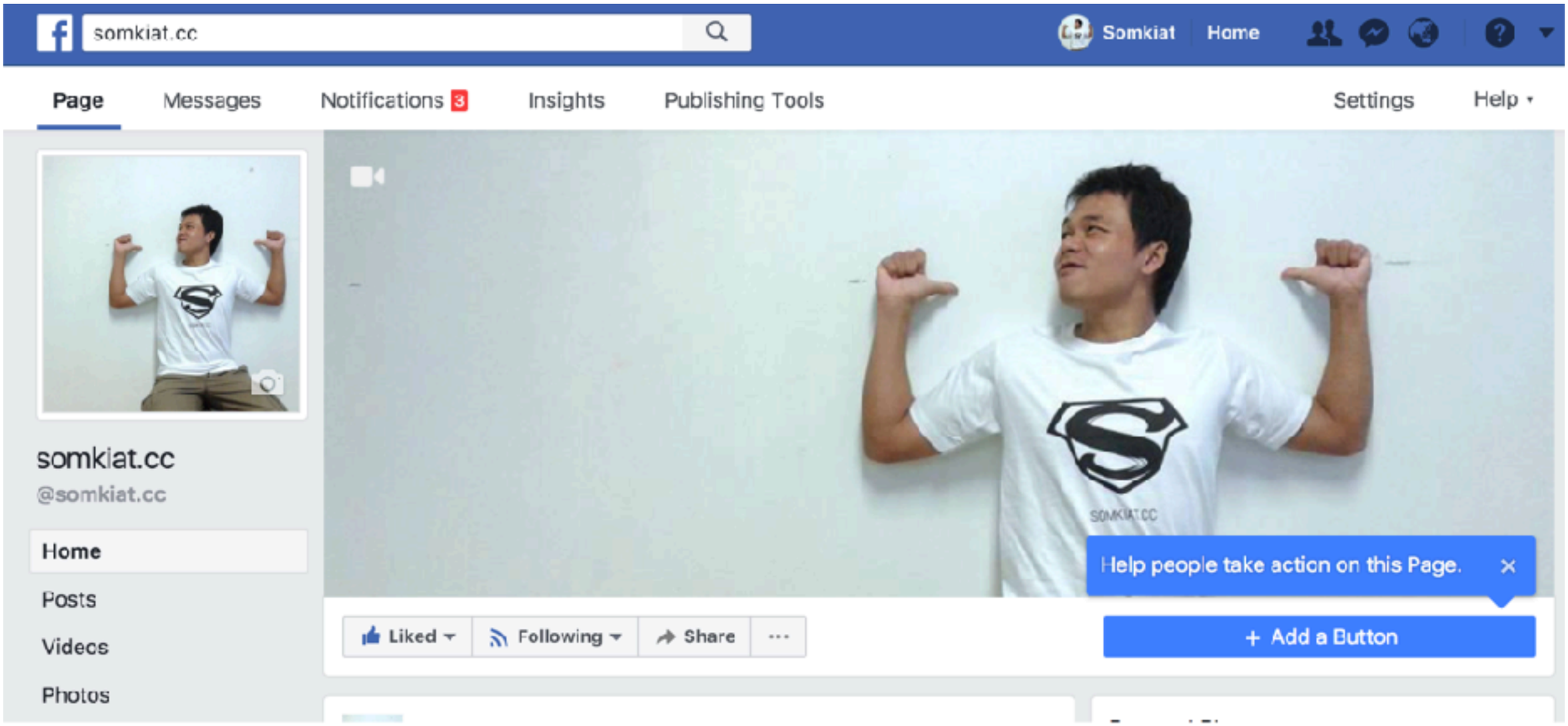
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Microservices

In Practices with Java Technology





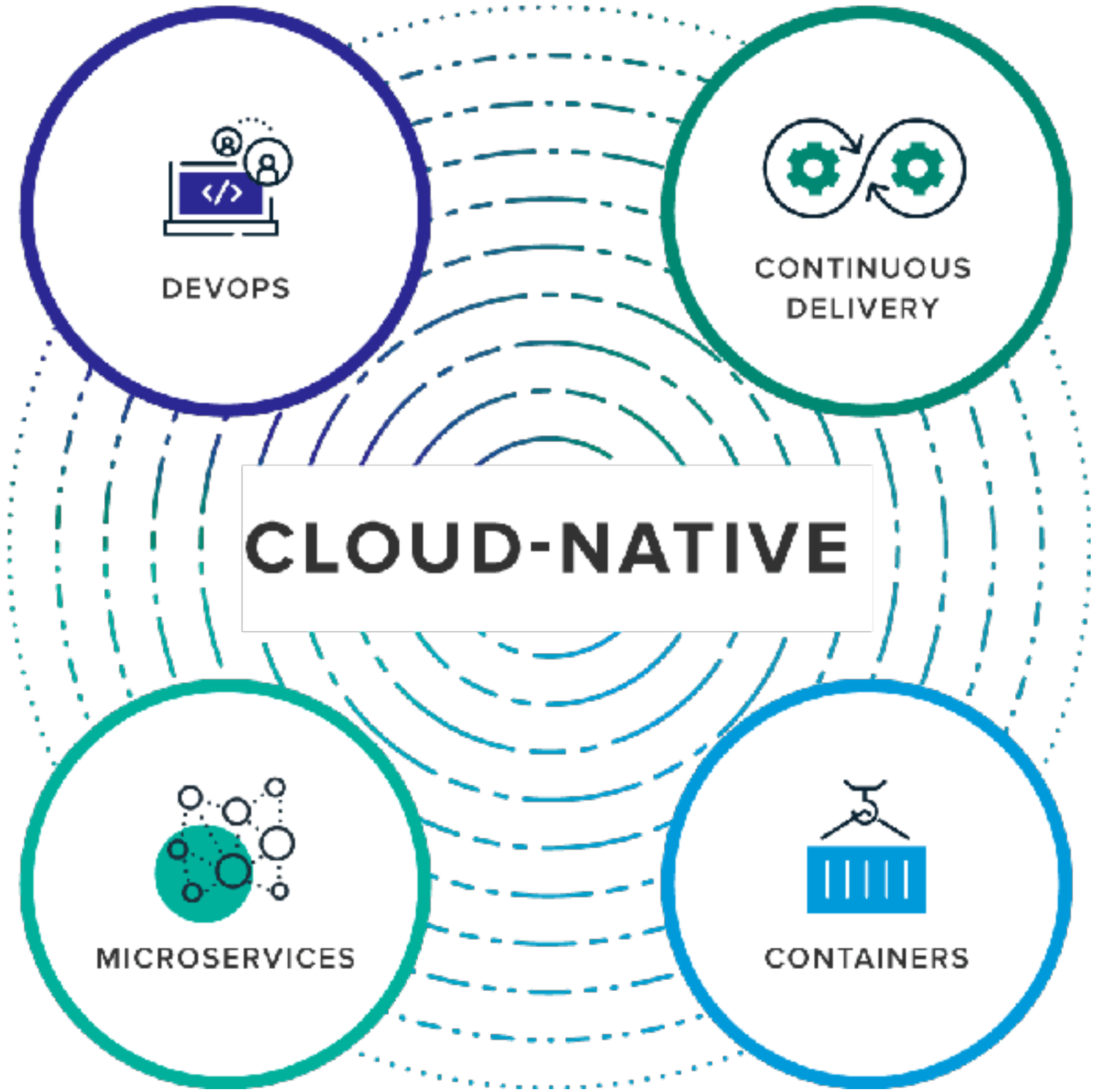
Agenda Day 1

1. Cloud Native Application
2. Microservices and DevOps
3. The architecture of Microservices
4. How to model Microservices
5. Integrating multiple Microservices
6. Developing Microservices with Java
7. Workshop

Agenda Day 2

1. Testing Microservices
2. Deploying Microservices
3. Maintaining healthy Microservices
4. Monitoring Microservices
5. Scaling up your Microservices
6. Workshop
7. Agile model

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



https://pivotal.io/cloud-native

Evolution of Architecture

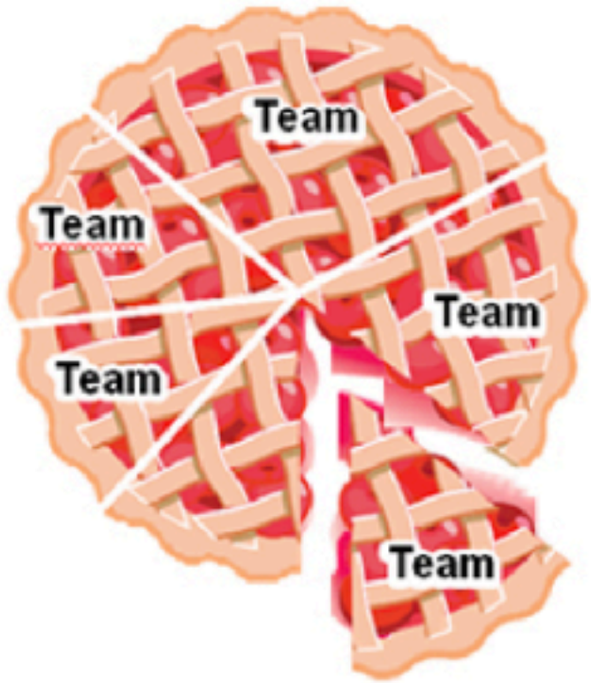
1990s and earlier

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



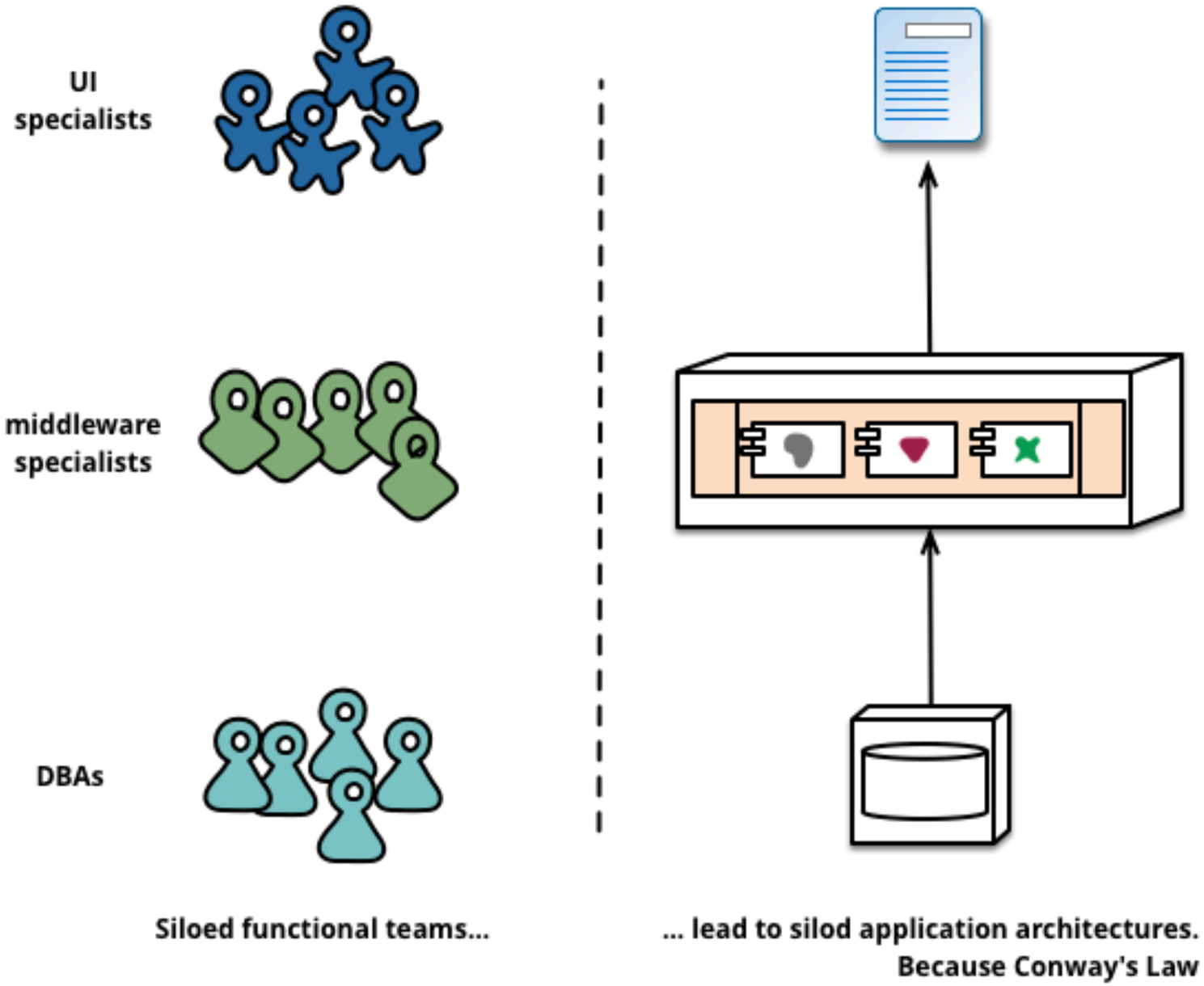
Elements in SOA are developed more autonomously but must be coordinated with others to fit into the overall design.

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.

Conway’s Law

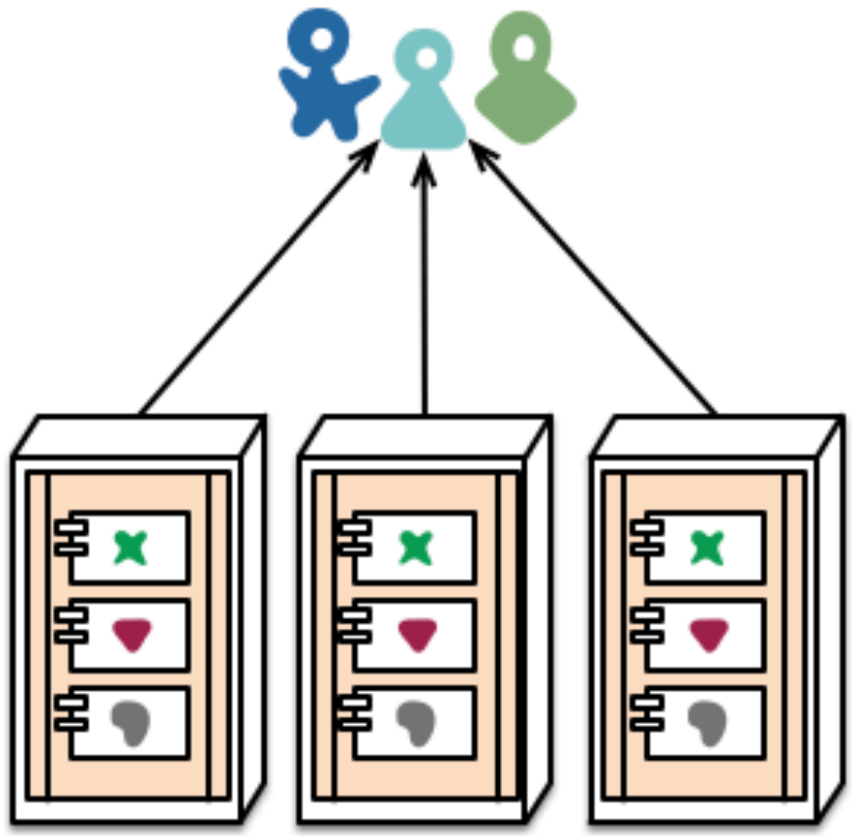


Microservices

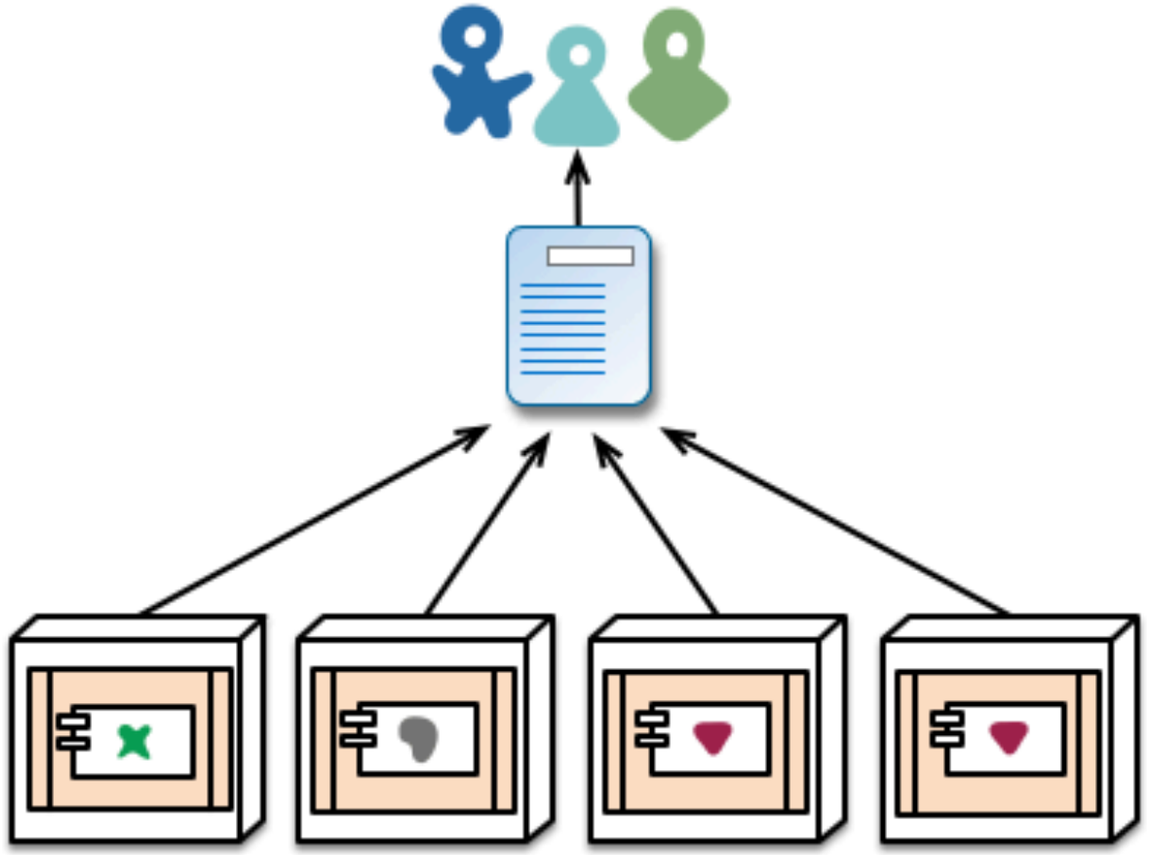
Microservices

Small, Do one thing
Modular
Easy to deploy
Scale independently

Conway's Law

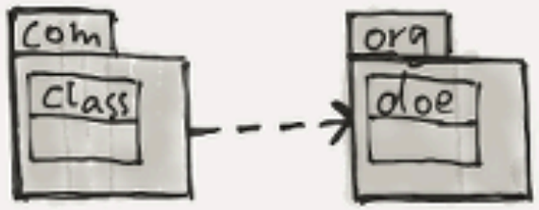


monolith - multiple modules in the same process

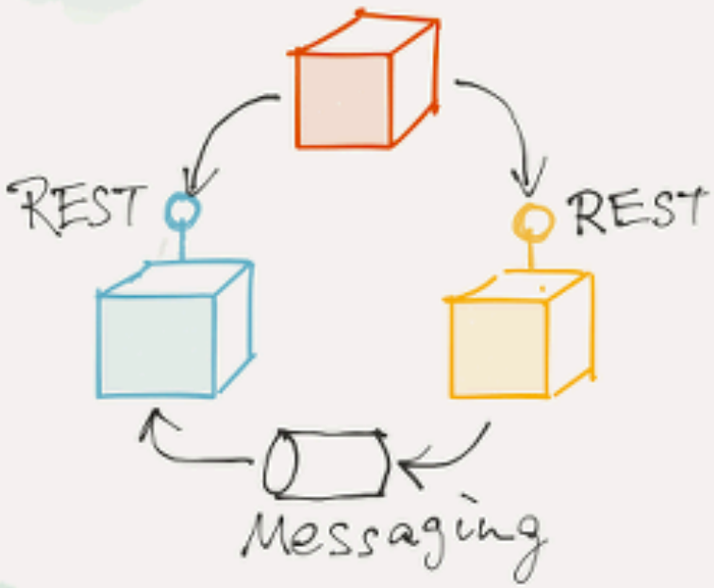


microservices - modules running in different processes

Architecture



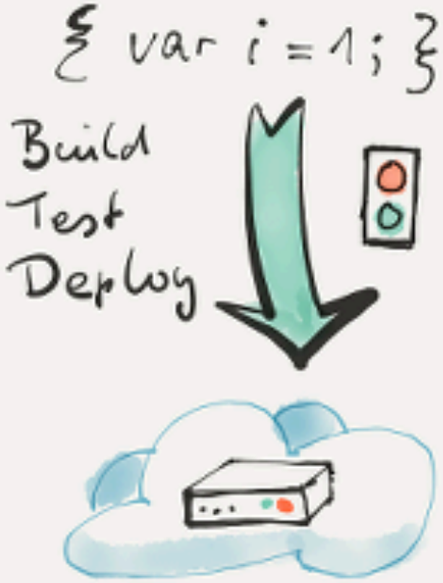
Microservices



People & Teams

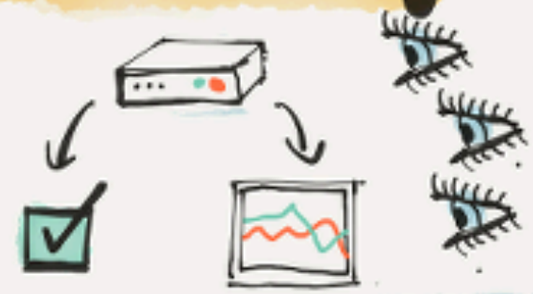
Deployment

Continous Delivery



Infrastructure

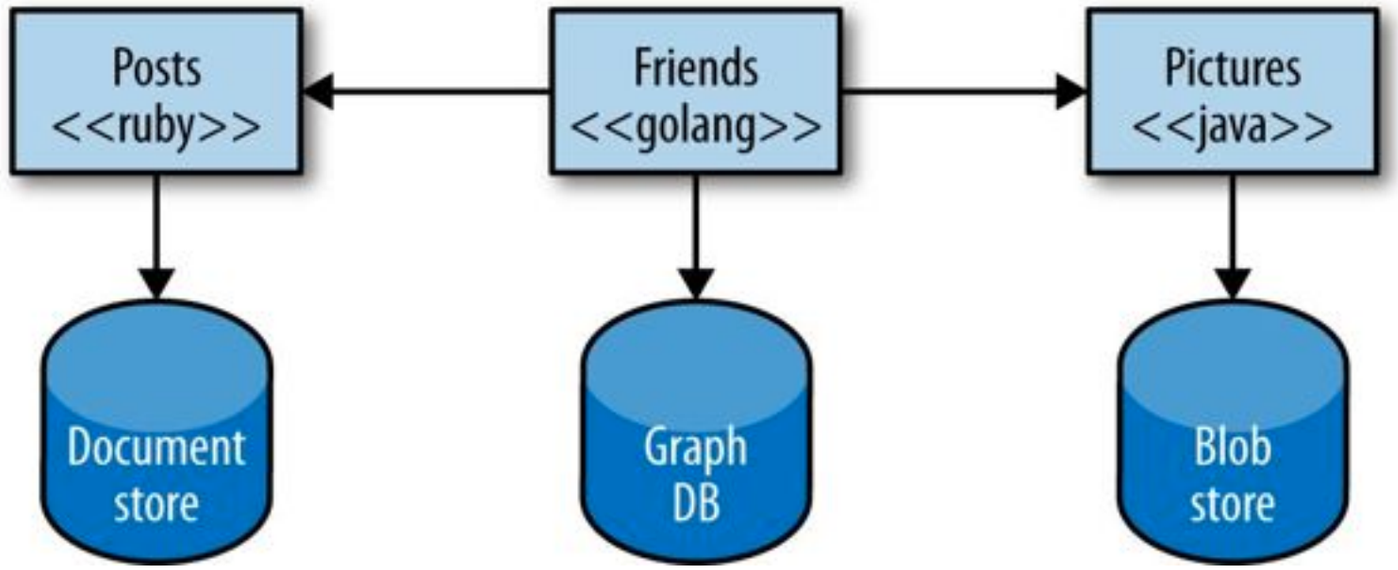
Monitoring



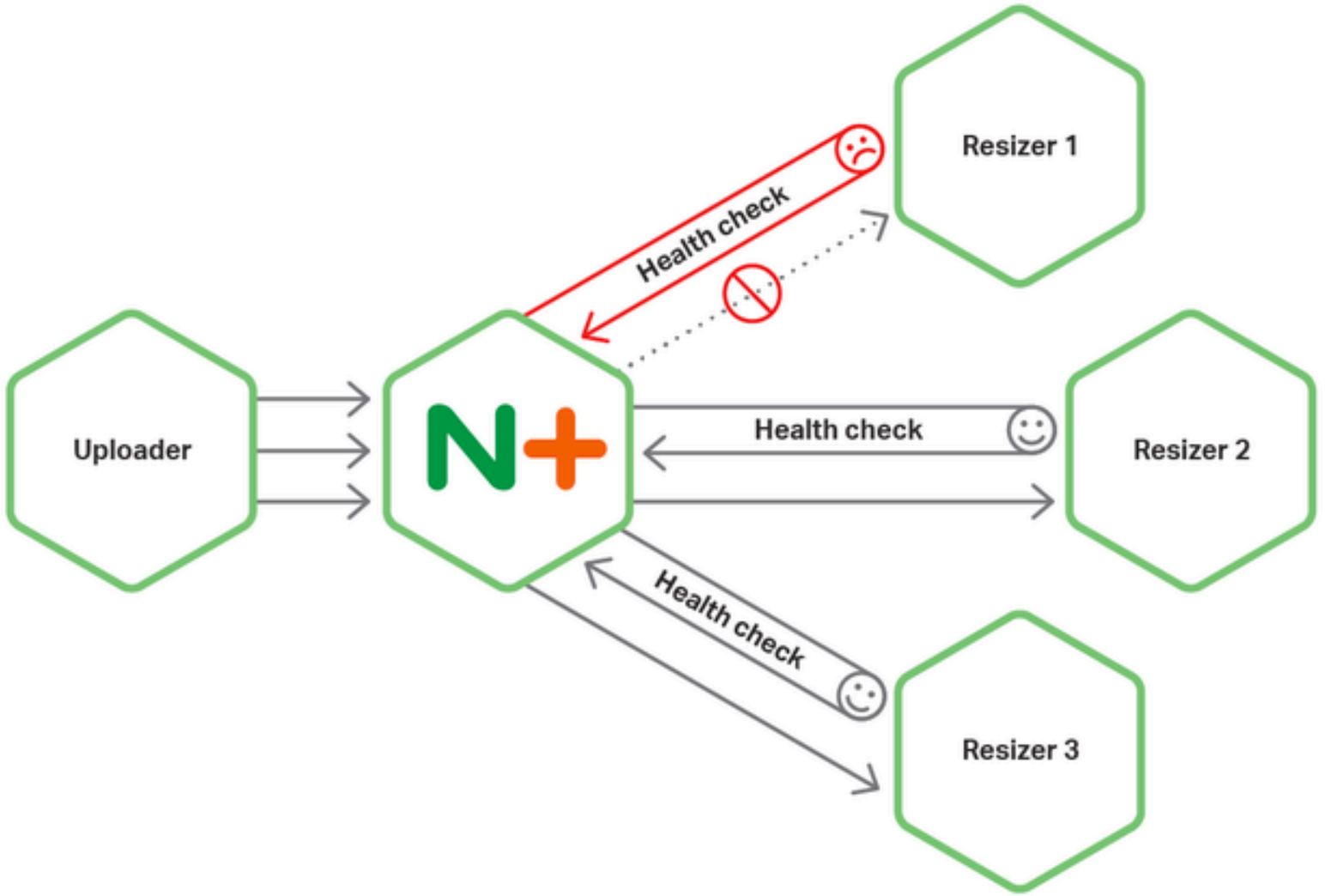
Key Benefits

1. Technology heterogeneity

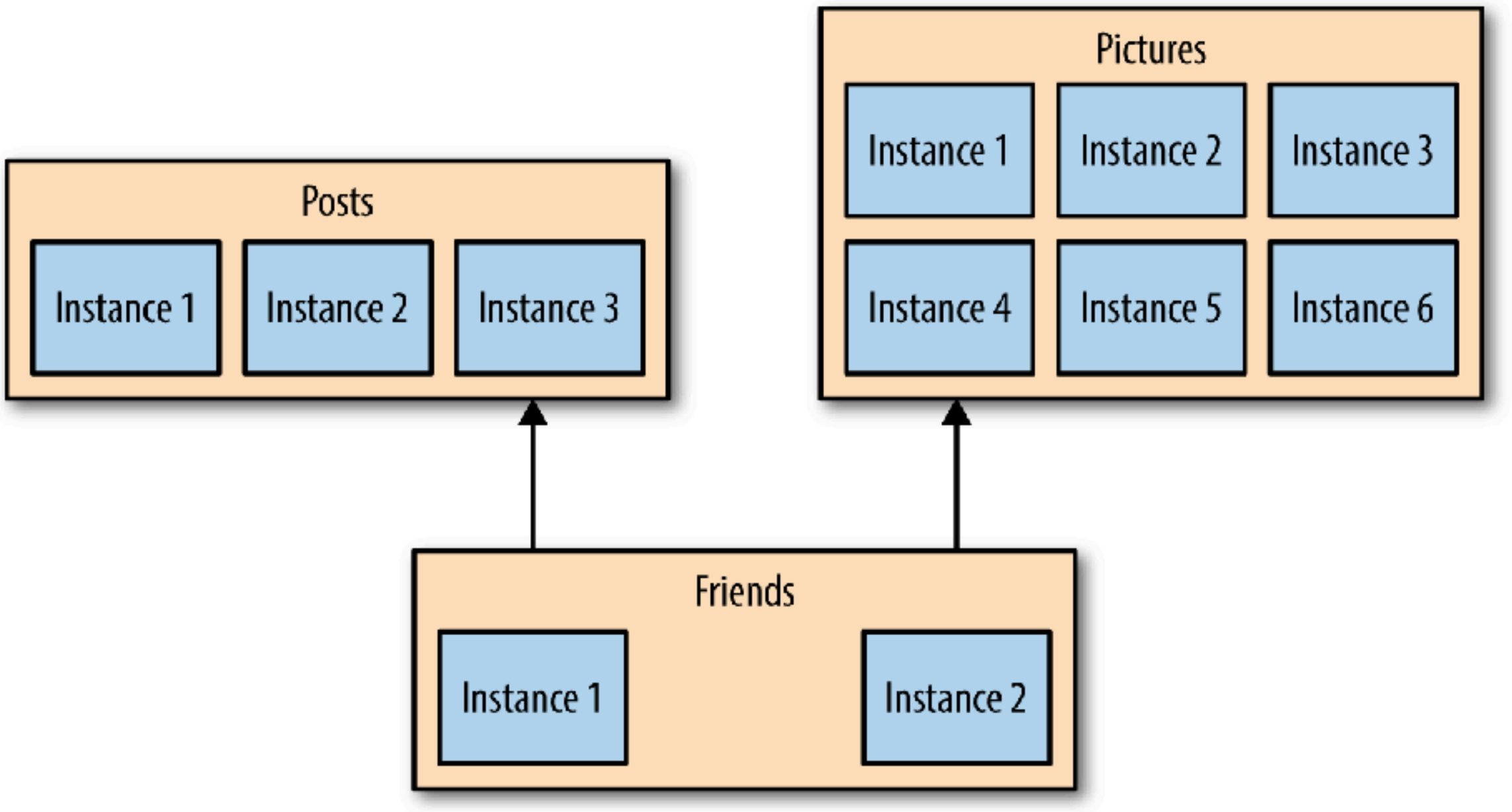
The right tool for each job



2. Resilience

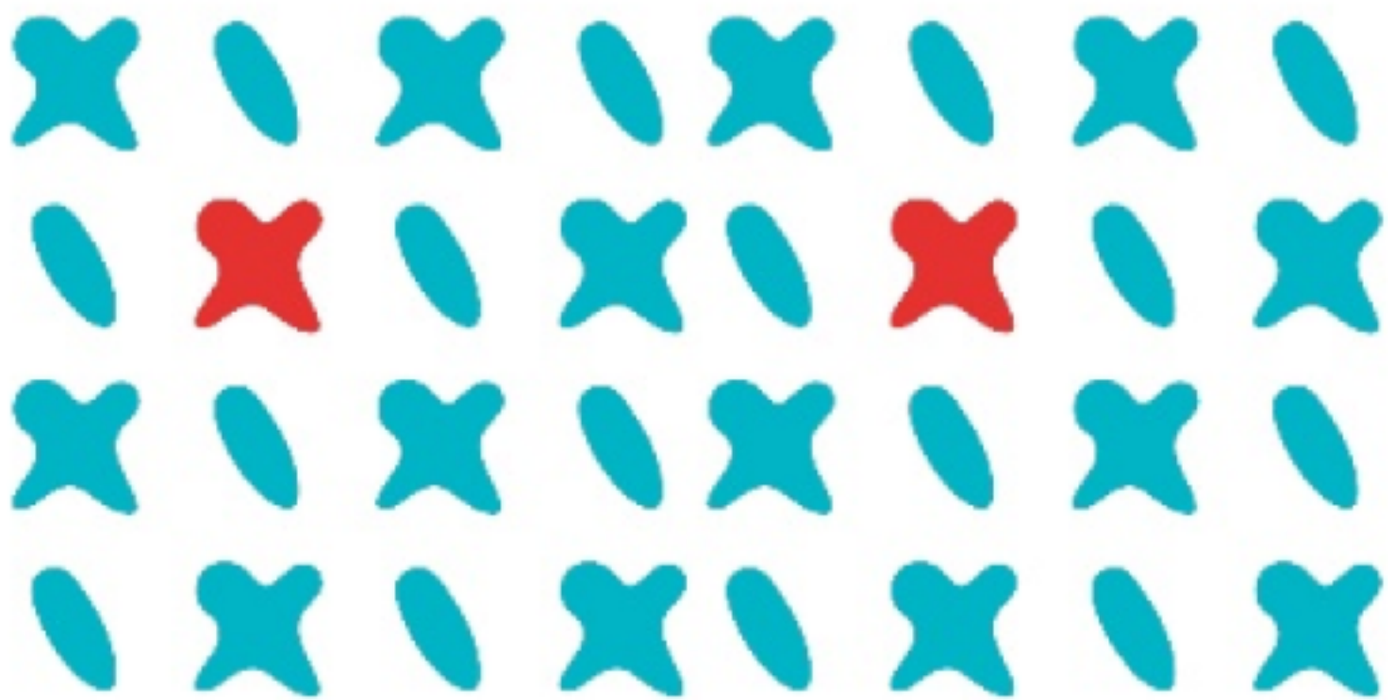


3. Scaling



4. Ease of deployment

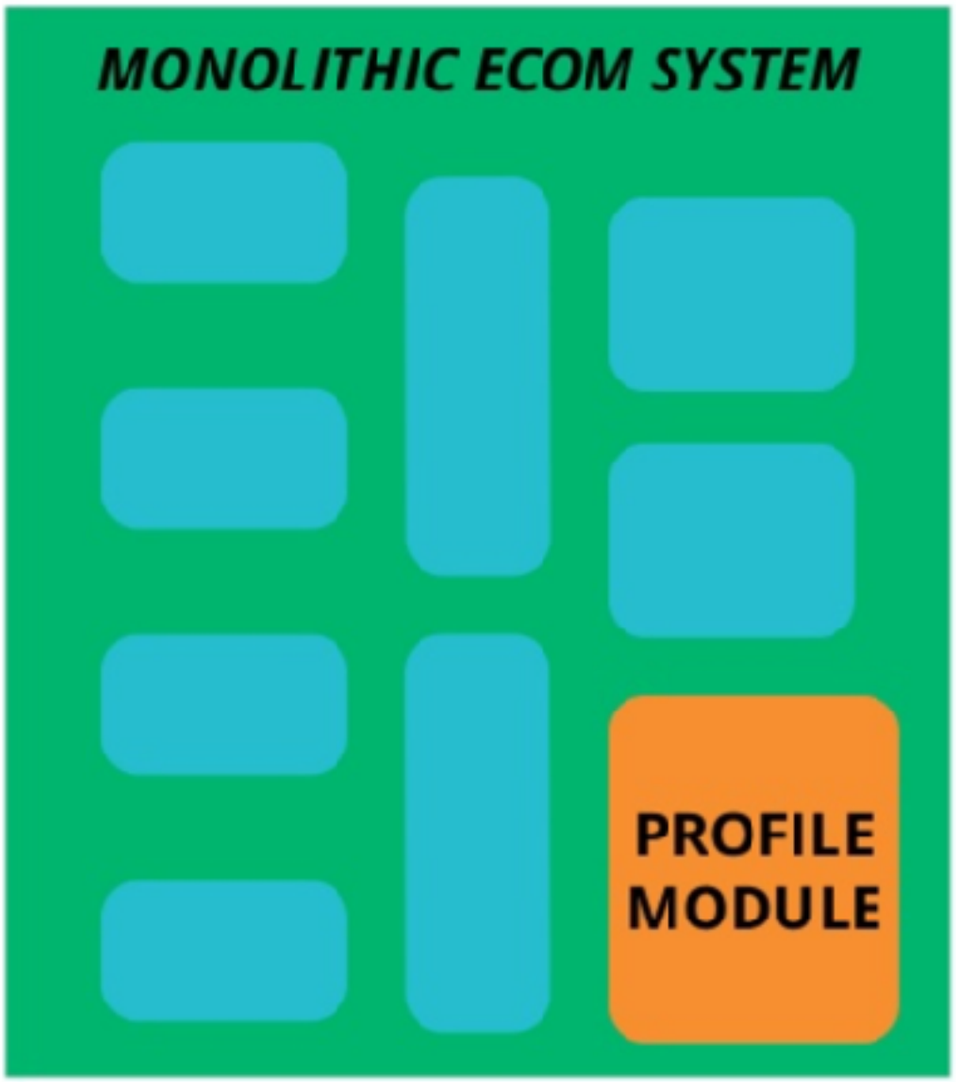
Deploys are faster, independent and problems can be isolated more easily



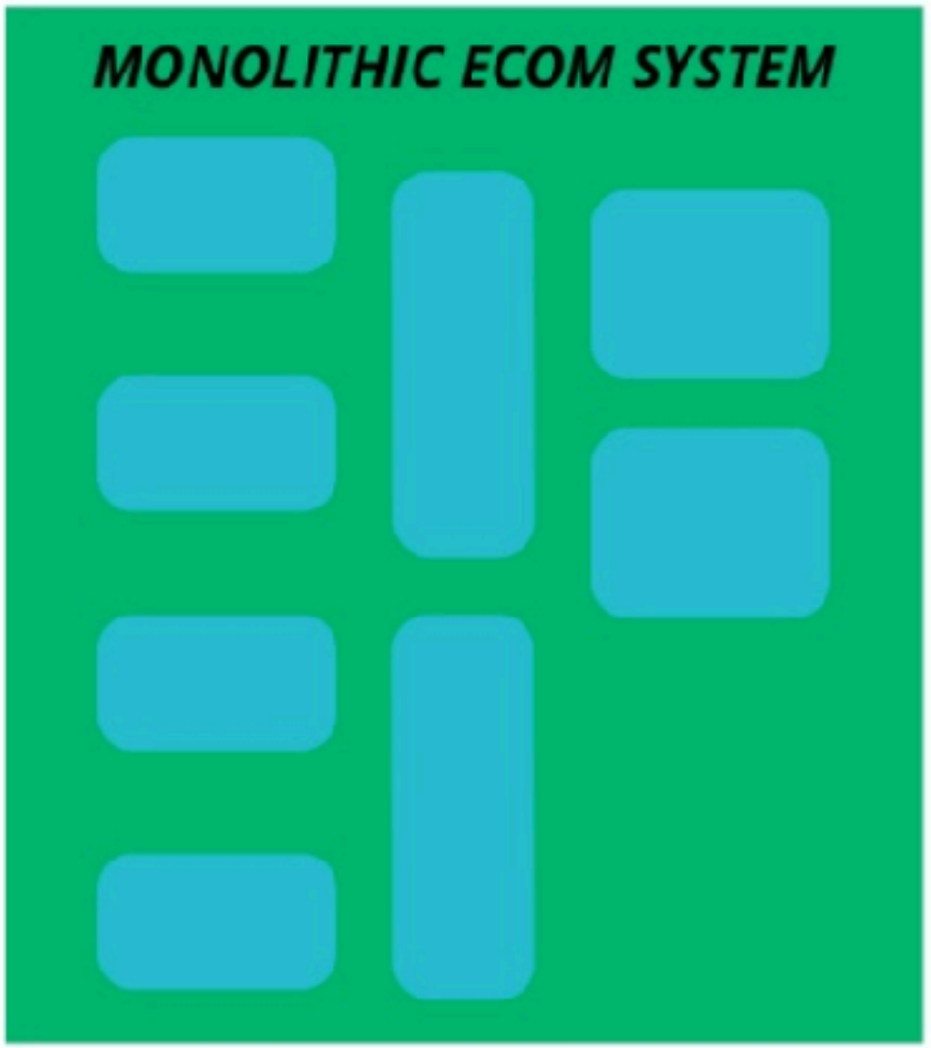
5. Organization alignment

Small teams and smaller codebases

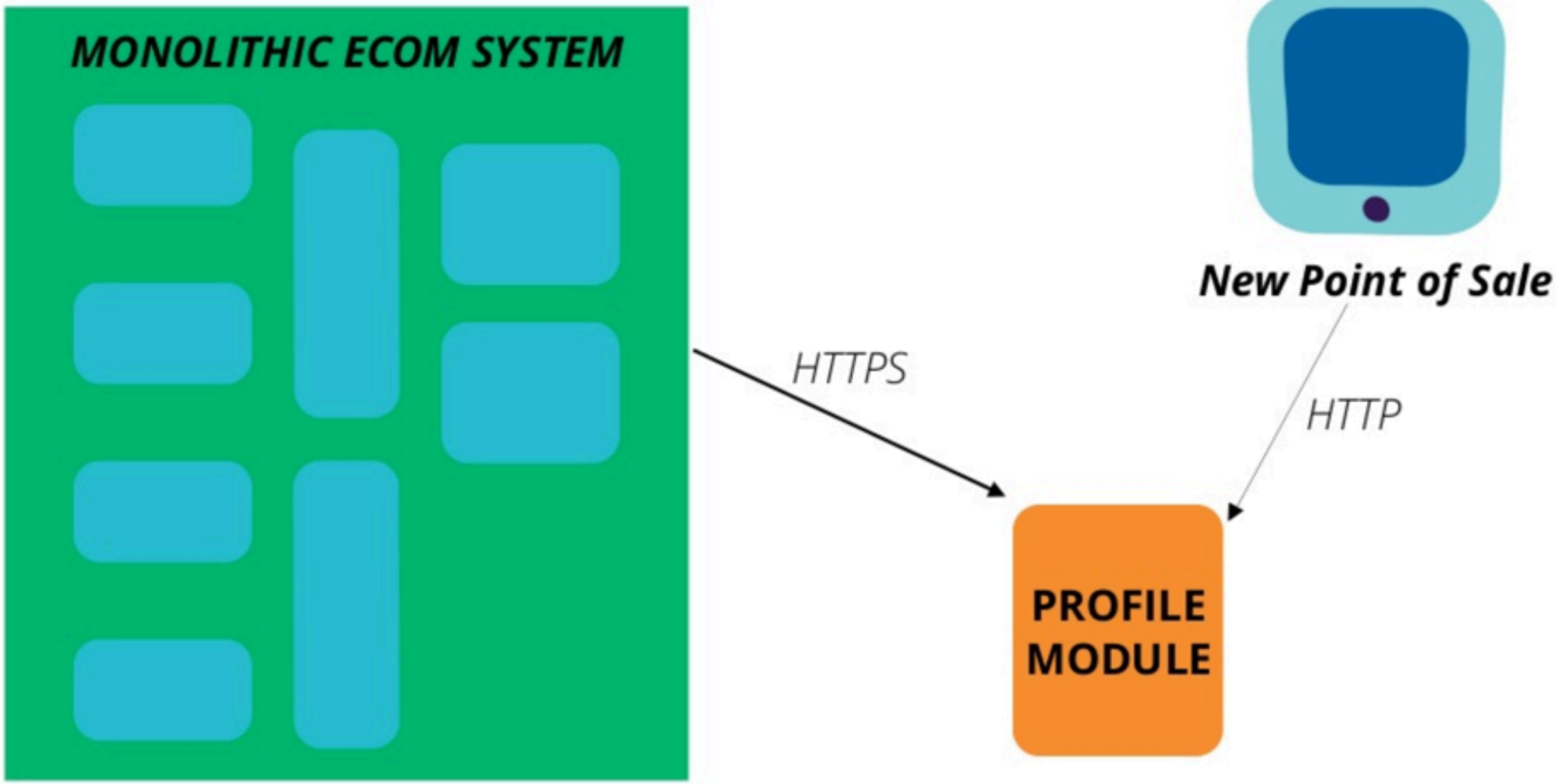
6. Composability and replaceability



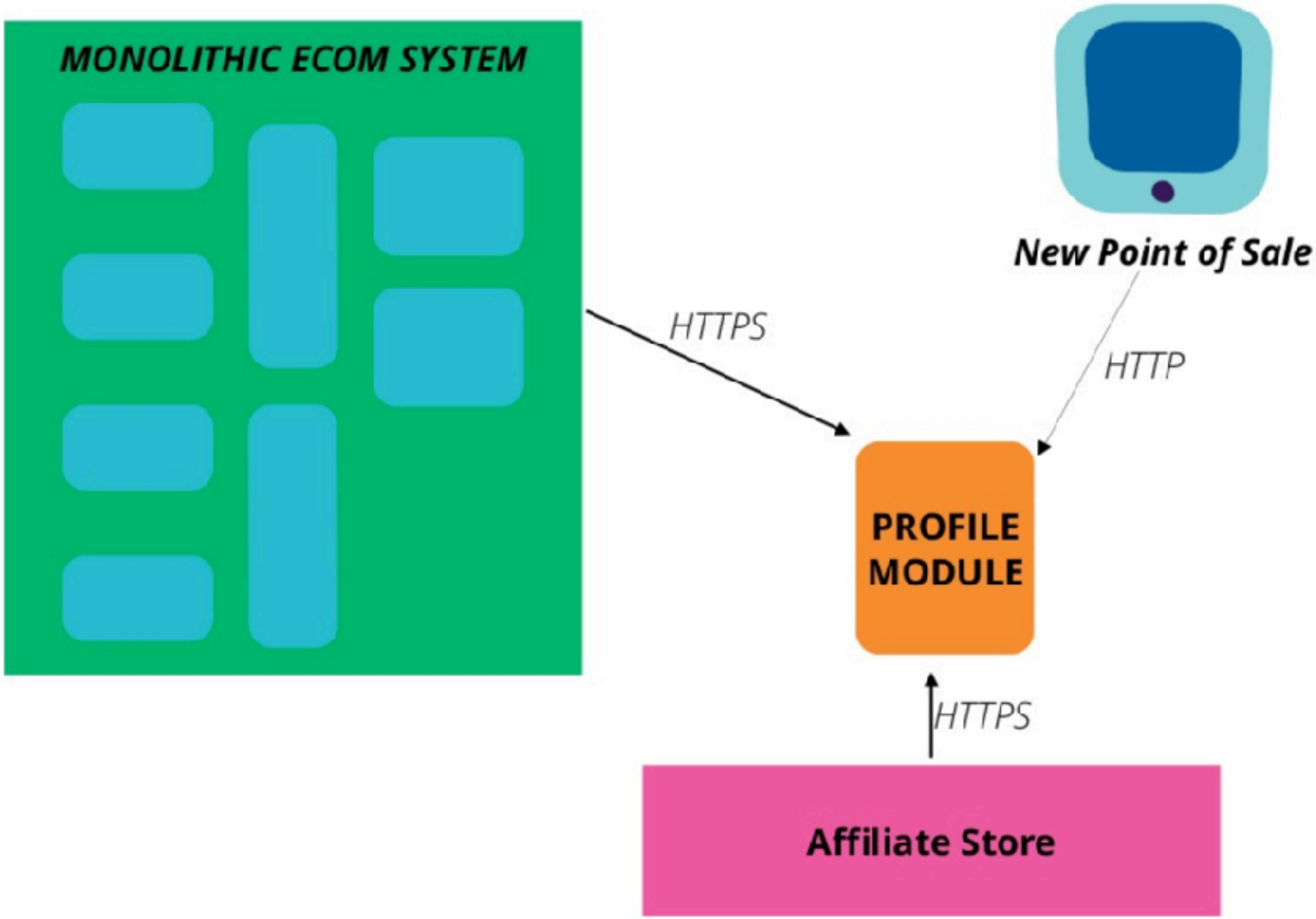
6. Composability and replaceability



6. Composability and replaceability



6. Composability and replaceability

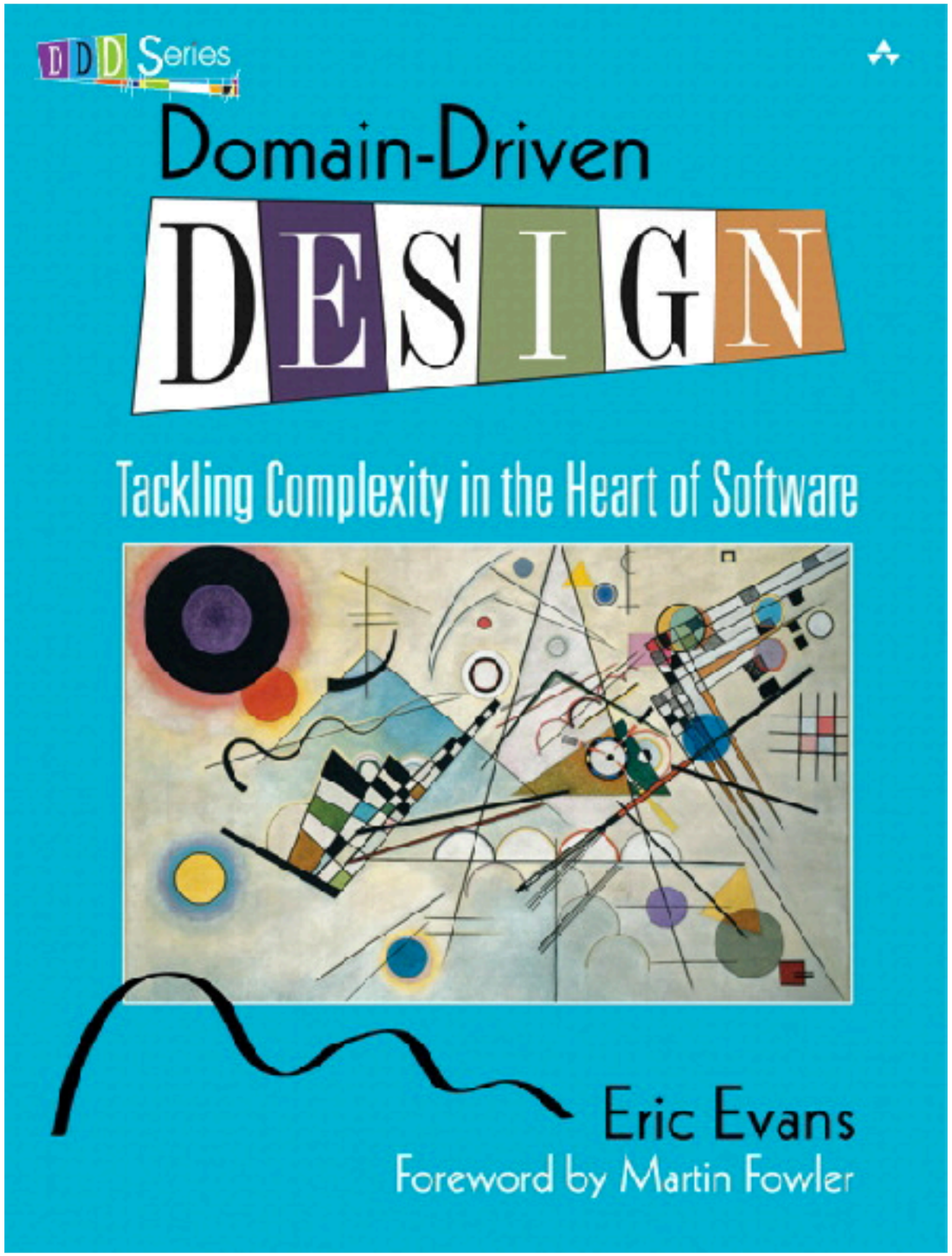


Characteristics

1. Responsible for a single capability

Types of capabilities

Business capability
Technical capability



2. Individually deployable

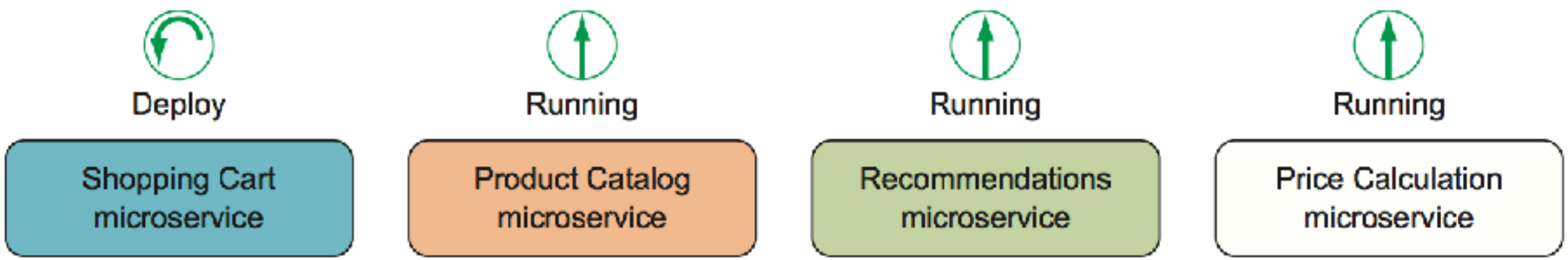


Figure 1.2 Other microservices continue to run while the Shopping Cart microservice is being deployed.

3. Consists of one or more processes

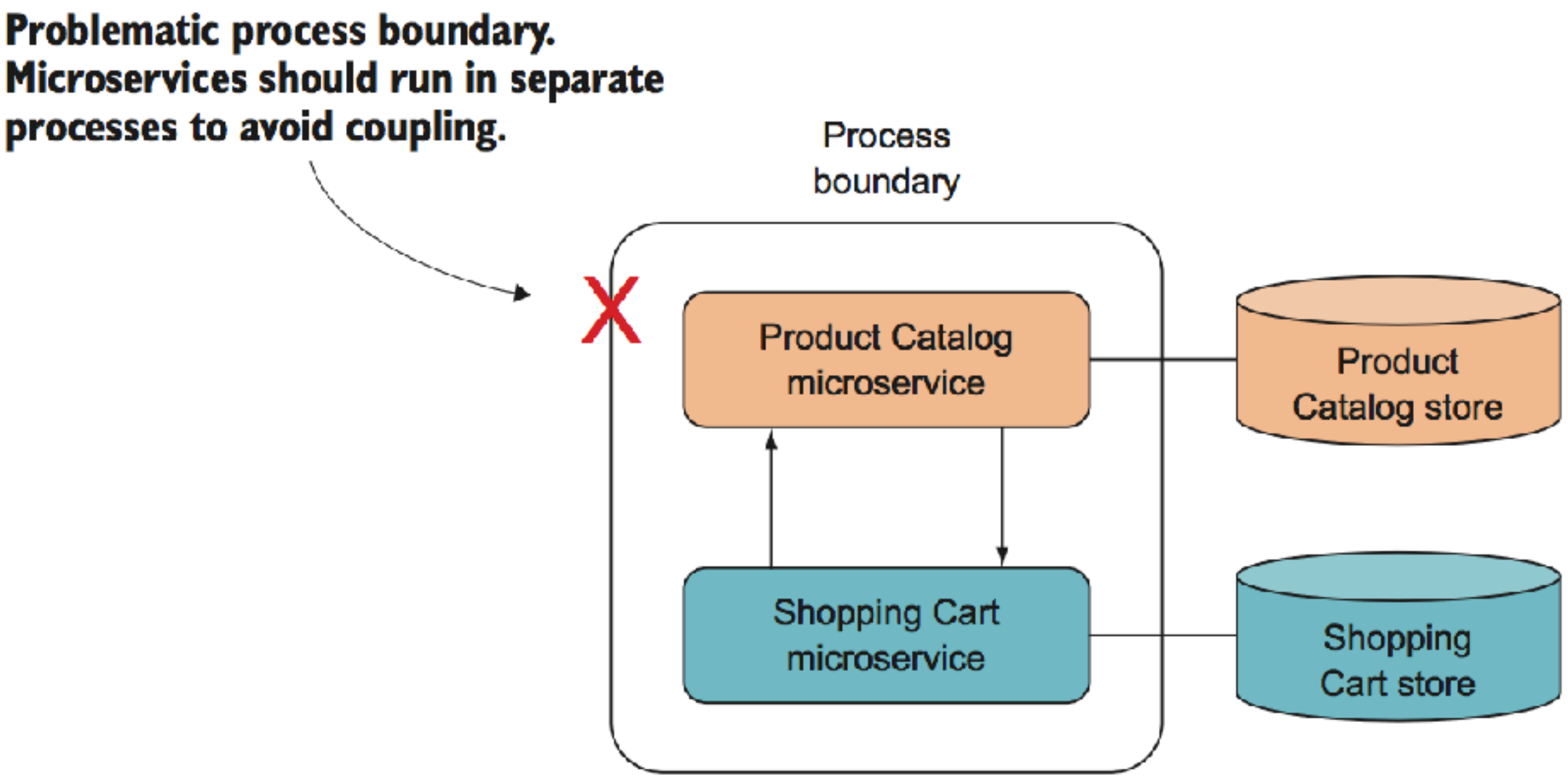


Figure 1.3 Running more than one microservice within a process leads to high coupling.

4. Own data store

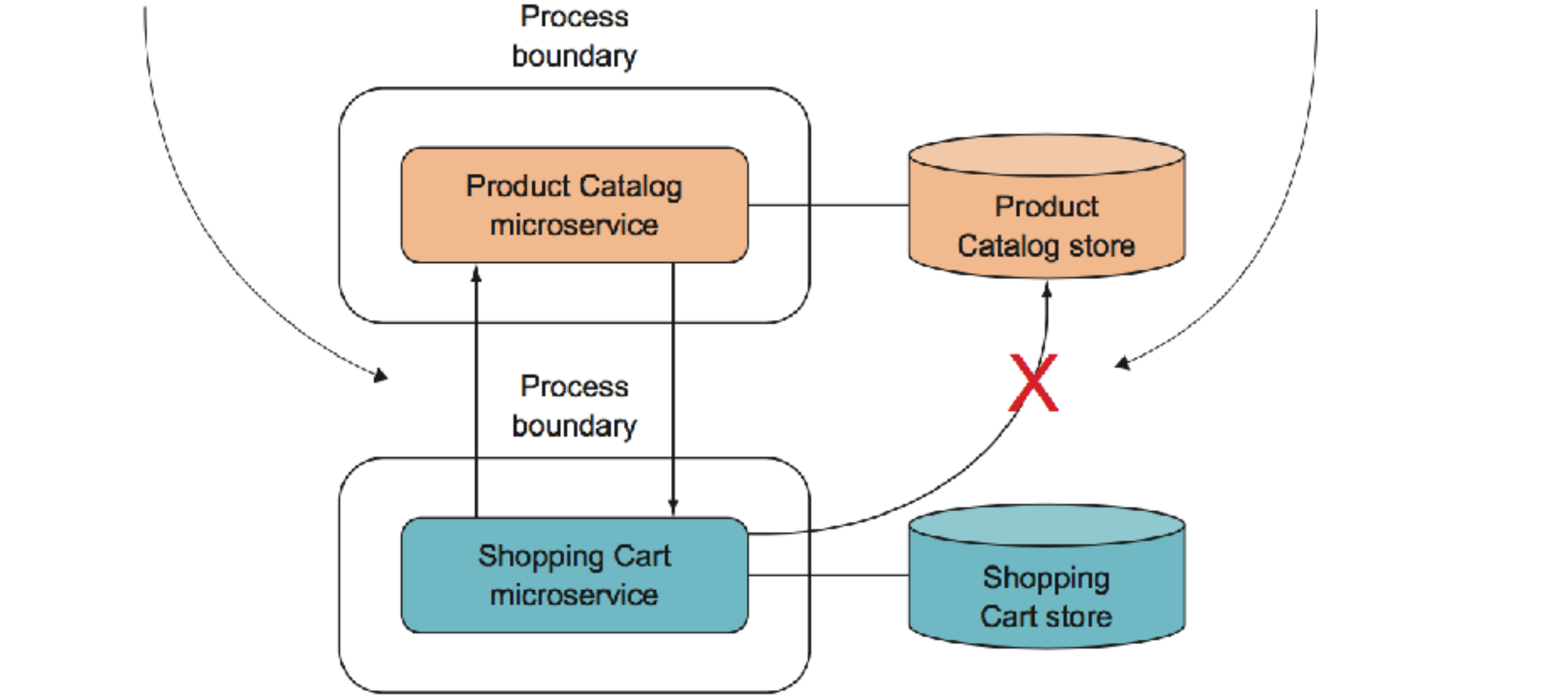


Figure 1.4 One microservice can't access another's data store.

5. Small team can maintain

6. Replaceable

Enabled system

Flexible
Scalable
Resilient

Challenges with Microservices ?

1. How to define the boundaries of each microservices ?

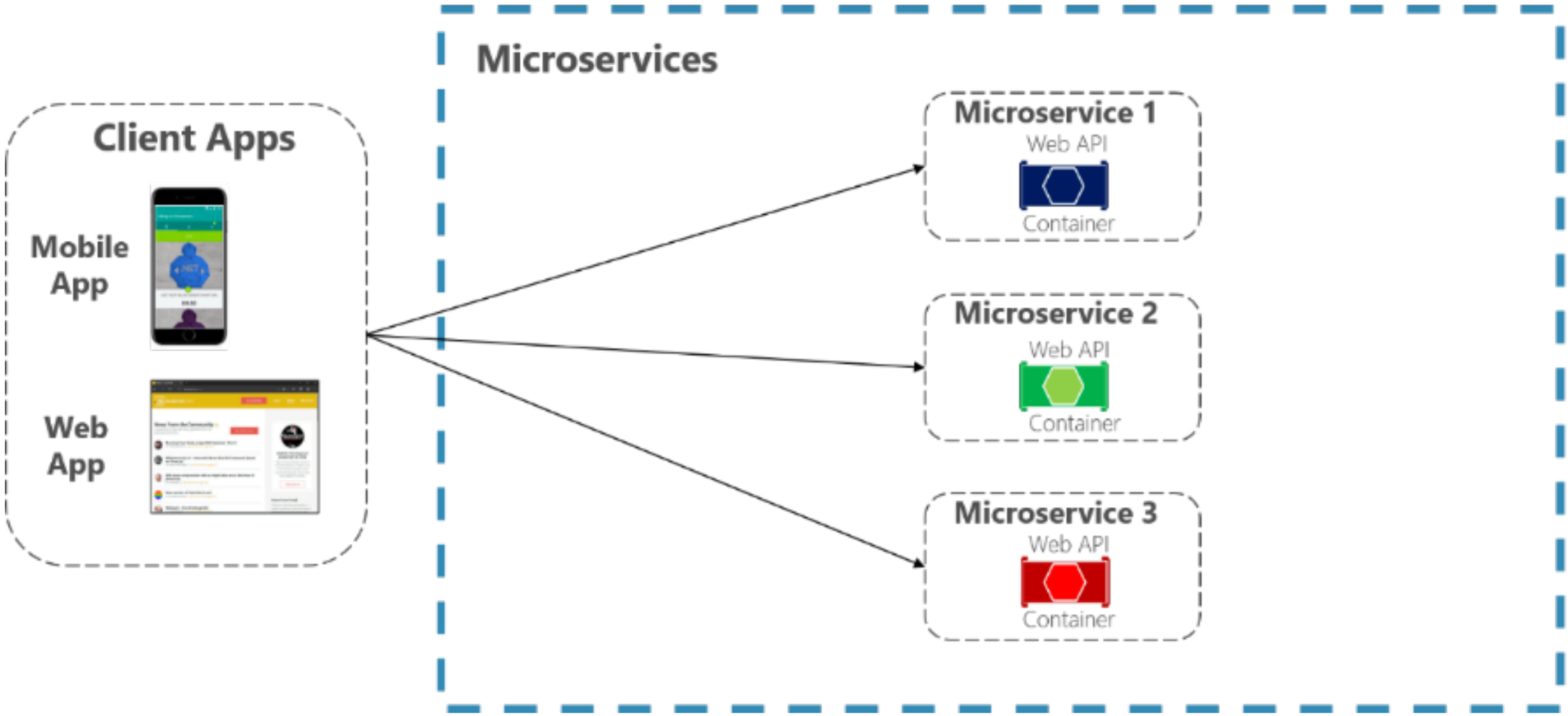
2. How to create queries that retrieve data from several microservices ?

Popular solutions

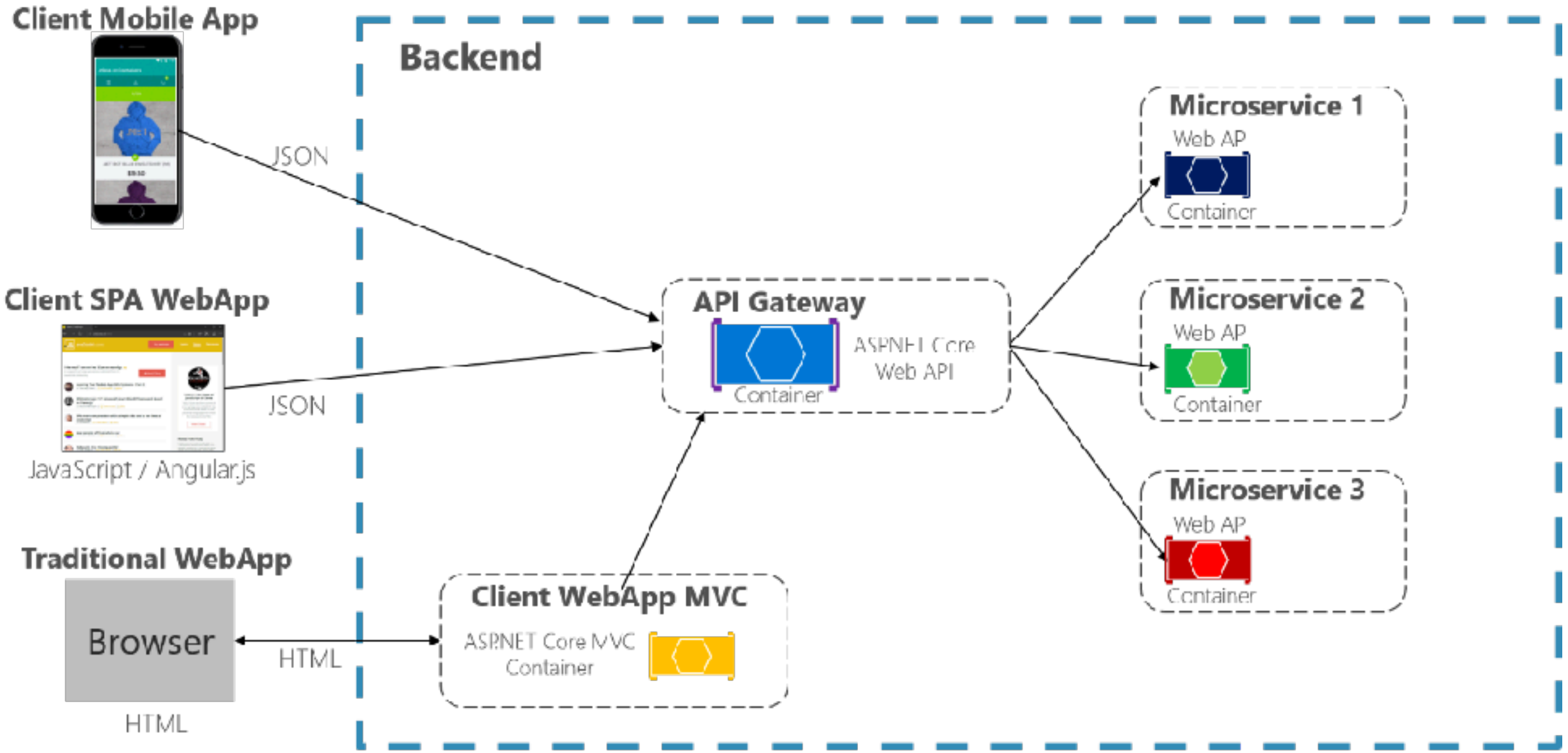
API Gateway
CORS with query/read tables
Cold data in centralize database

Direct Client-To-Microservice communication

Architecture

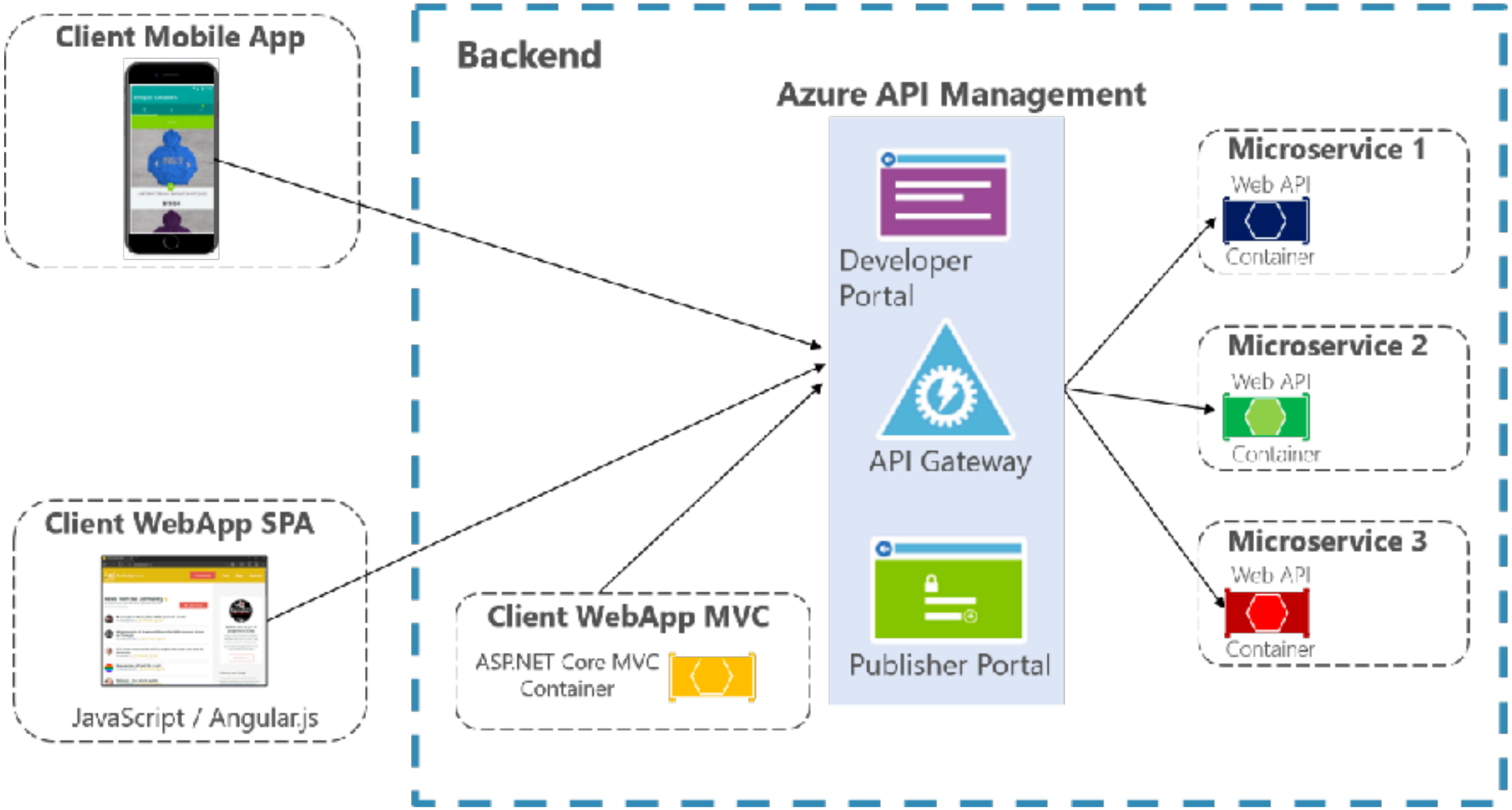


Using the **API Gateway Service**

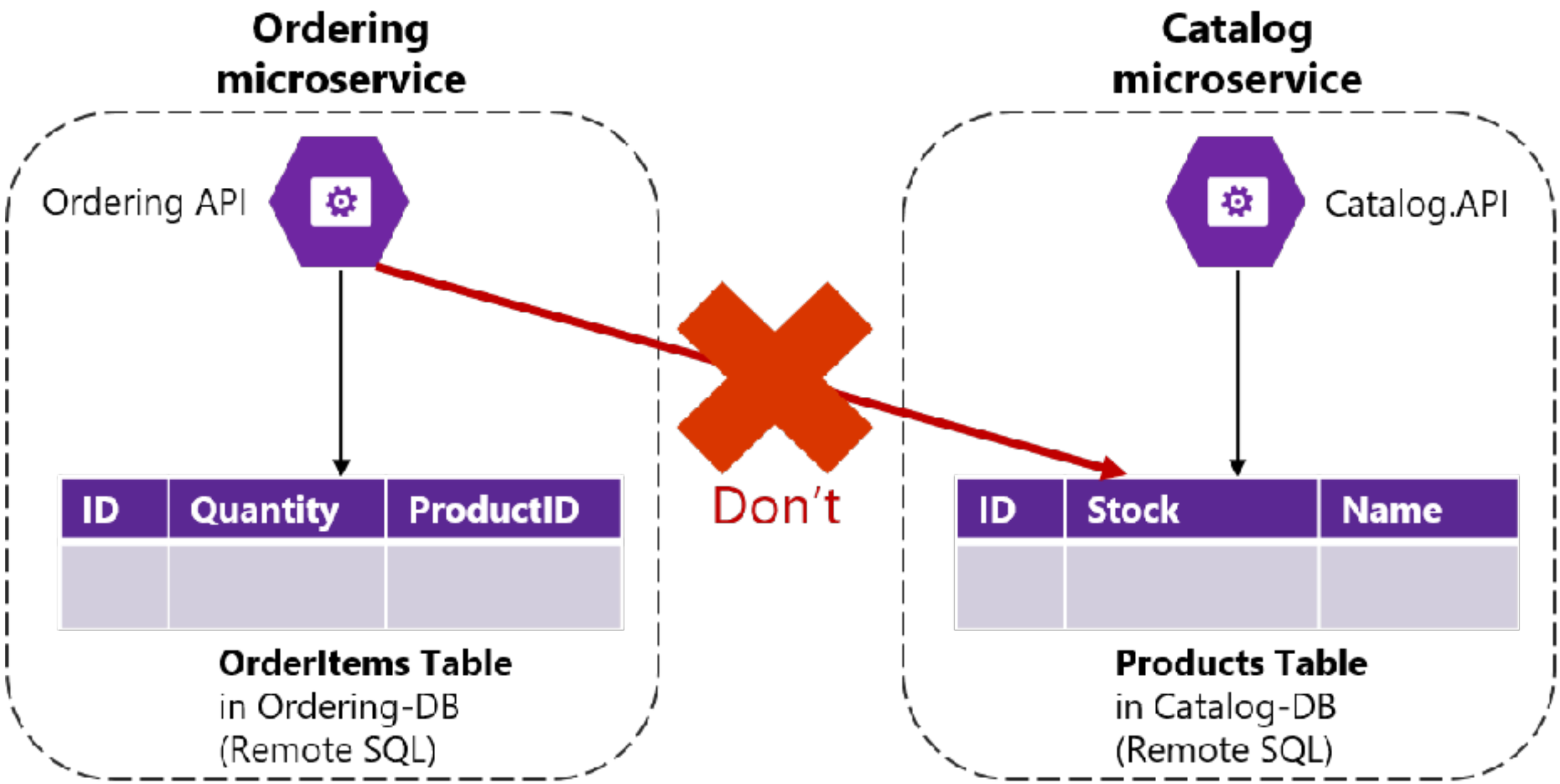


API Gateway with Azure API Management

Architecture



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
Messaging

Communication

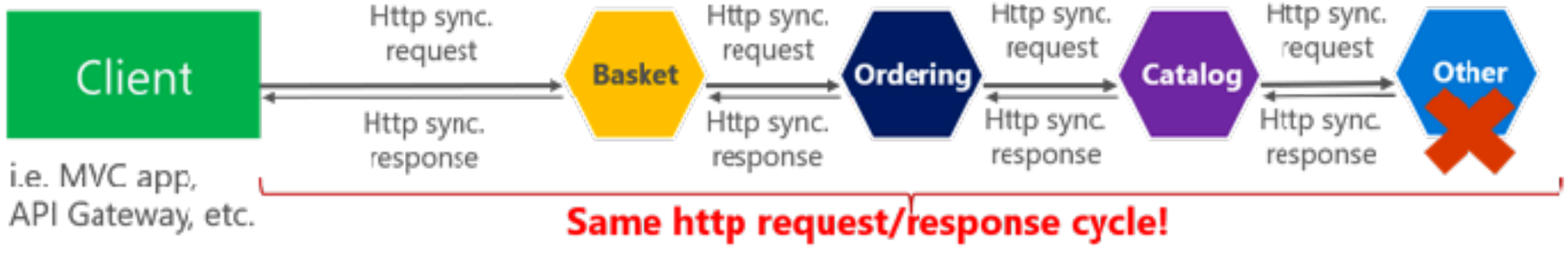
Request-Response model
Observer model

Communication

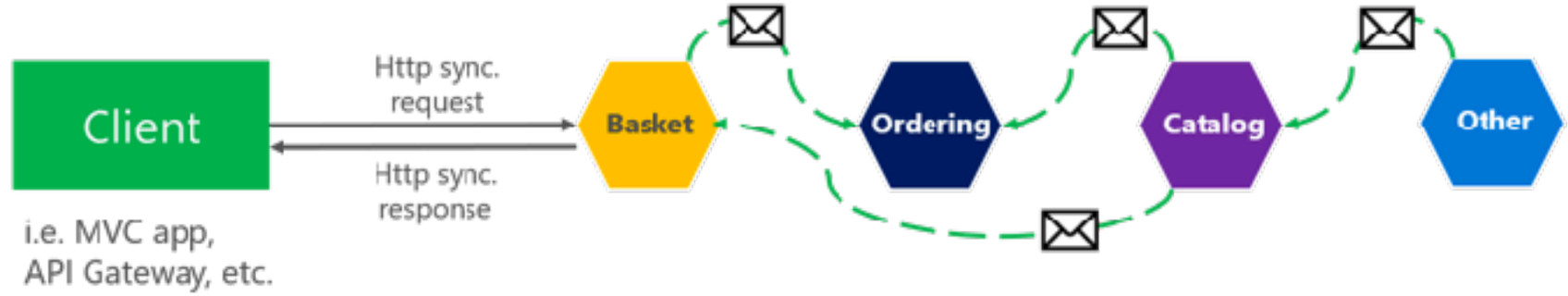
Synchronous vs. async communication across microservices

Anti-pattern

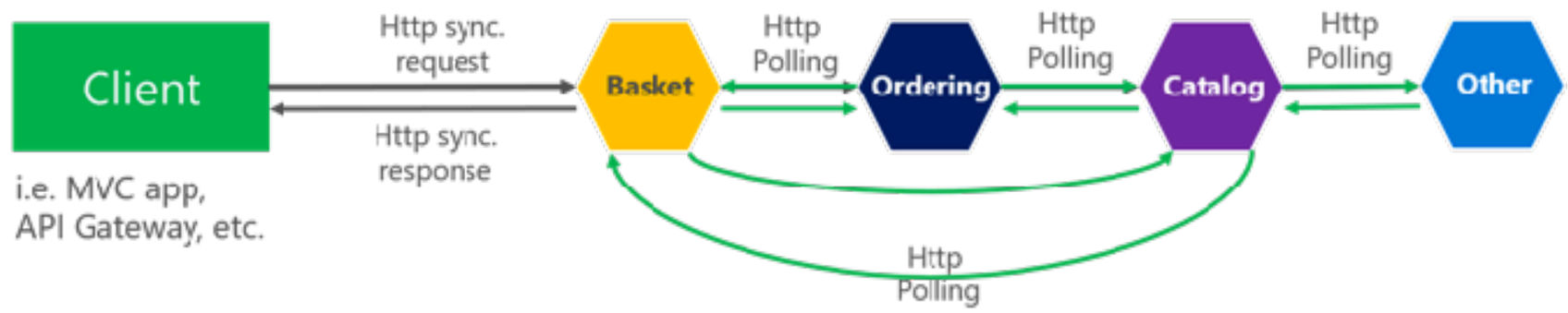
Synchronous
all req./resp. cycle



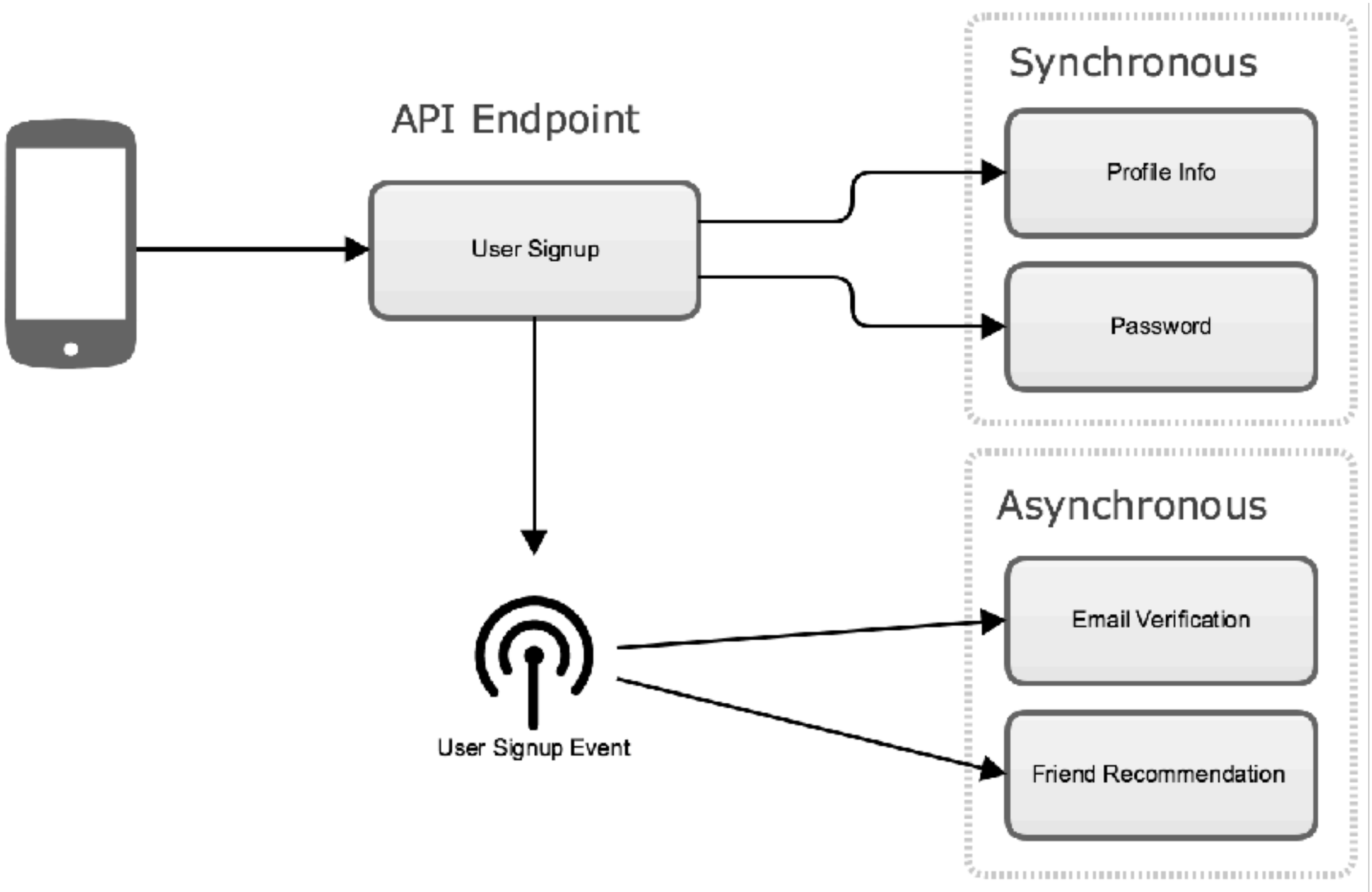
Asynchronous
Comm. across
internal microservices
(EventBus: i.e. **AMPQ**)



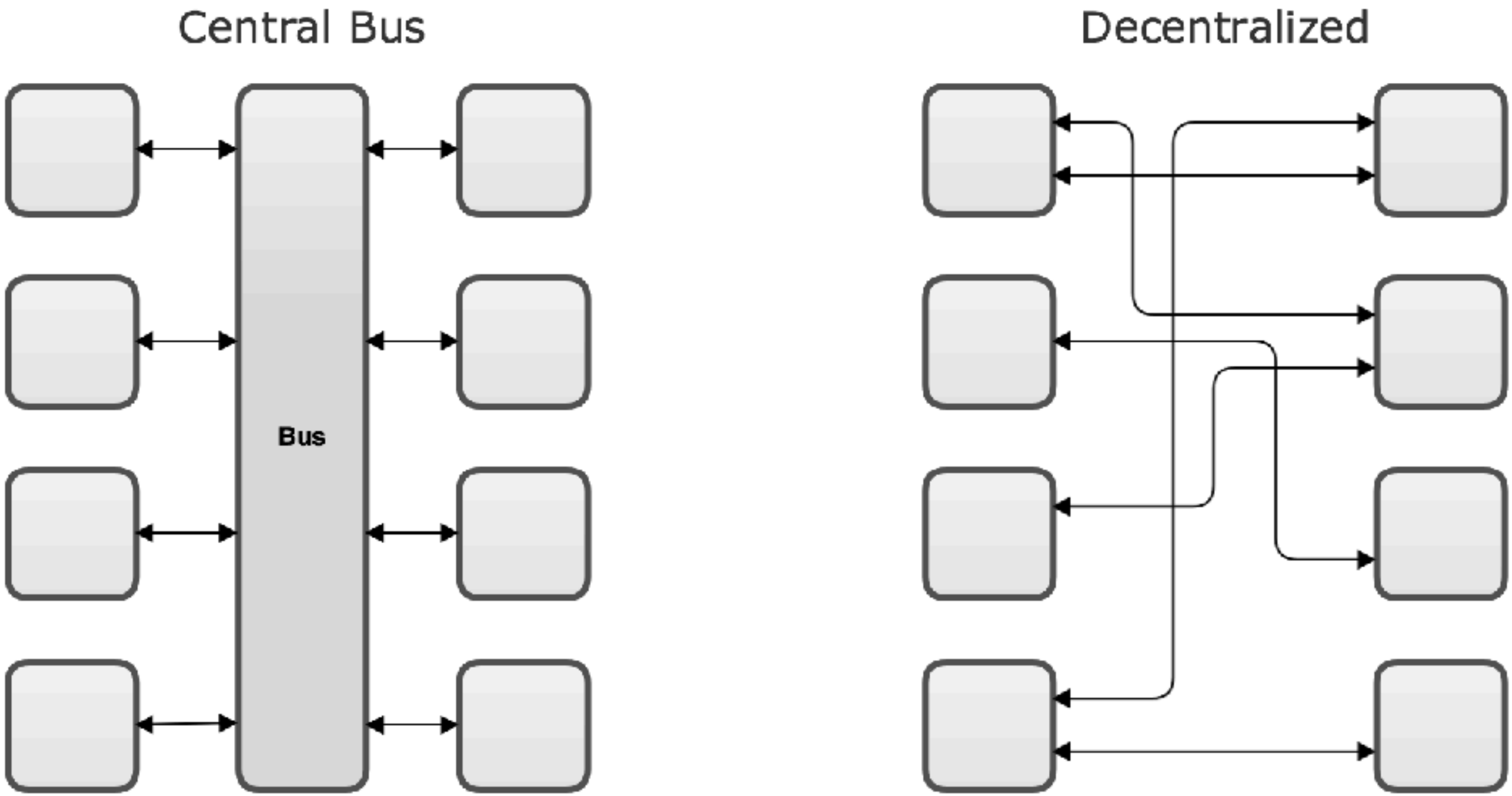
"Asynchronous"
Comm. across
internal microservices
(Polling: **Http**)



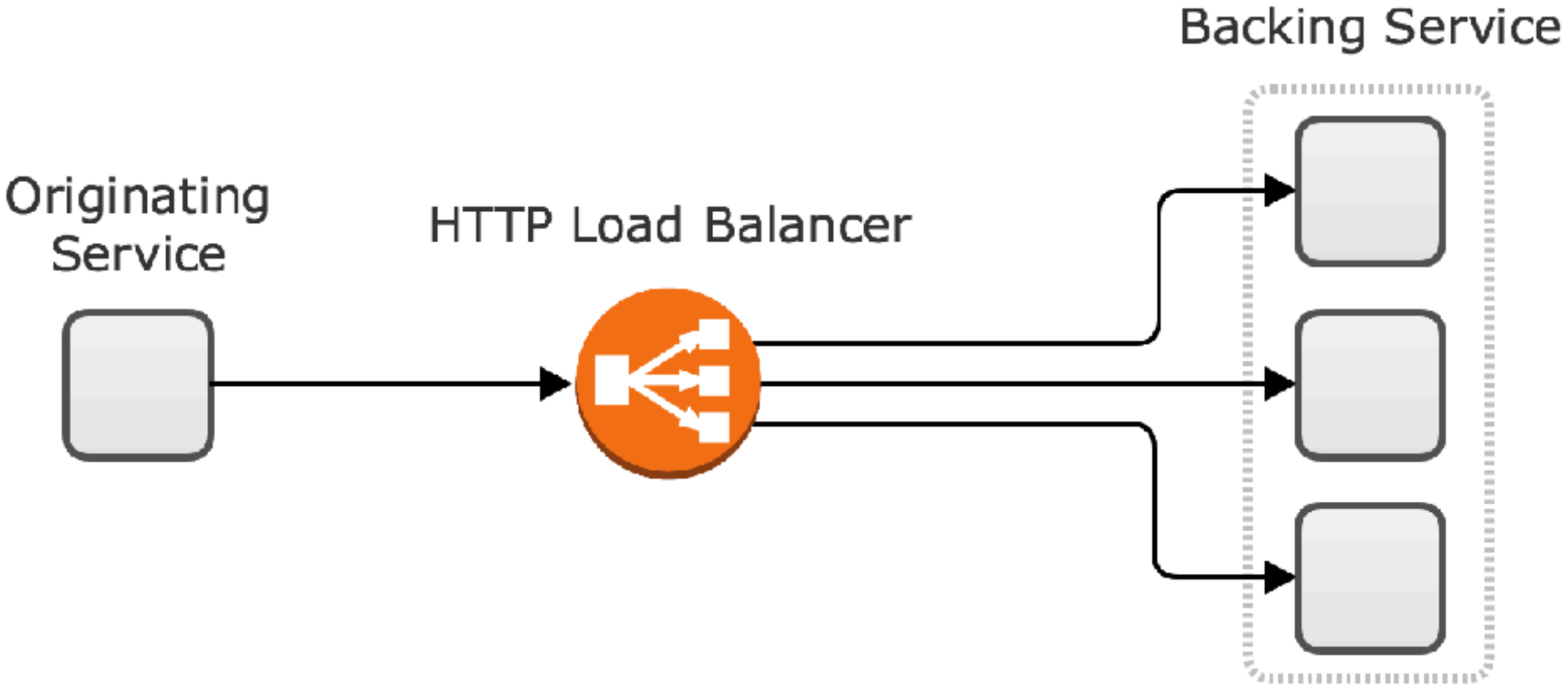
Communication



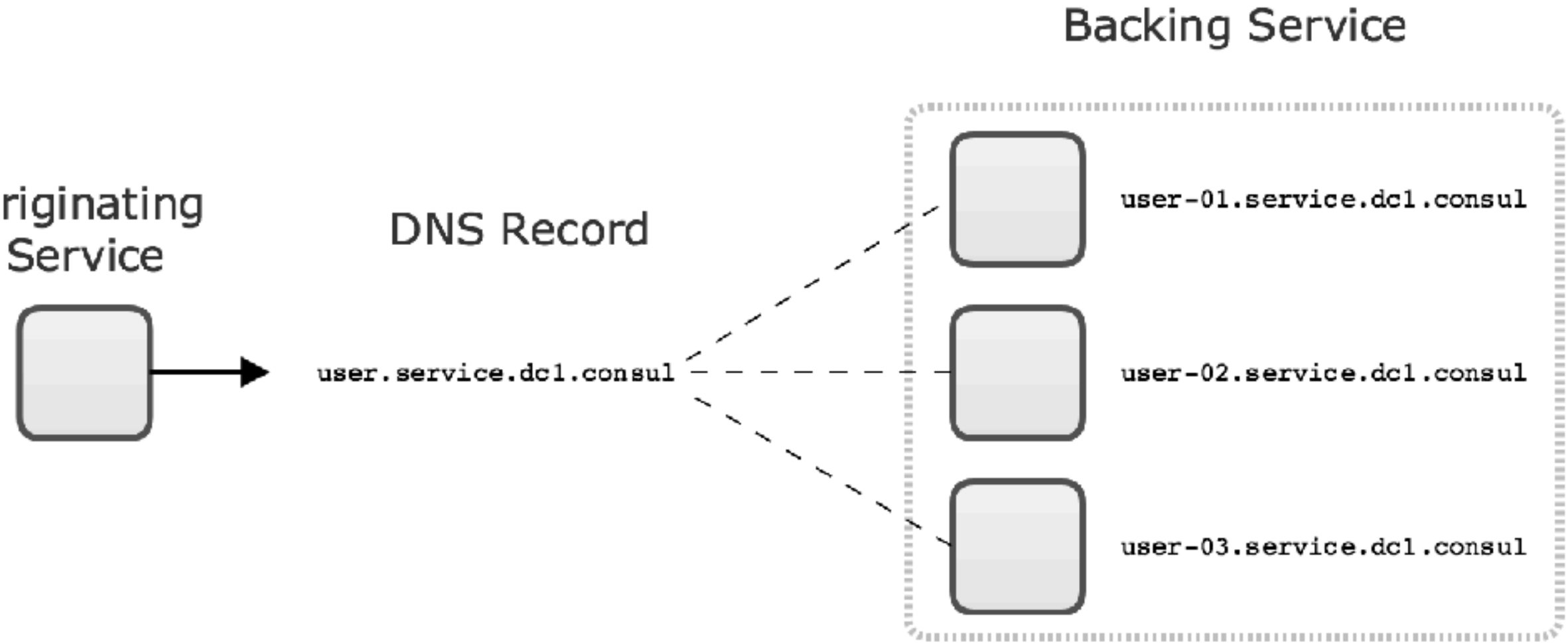
Anti-pattern :: centralize bus service



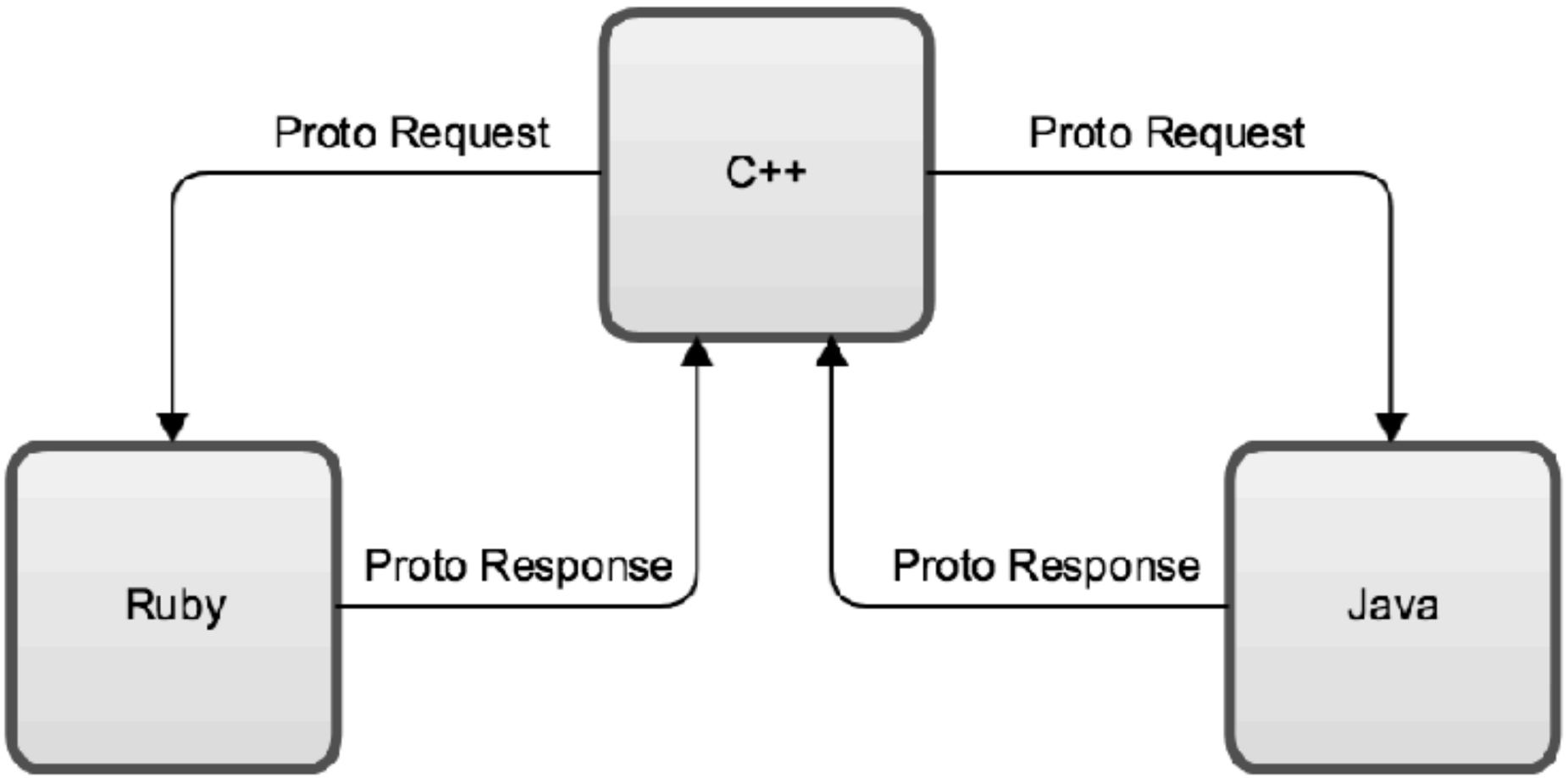
Request-response model



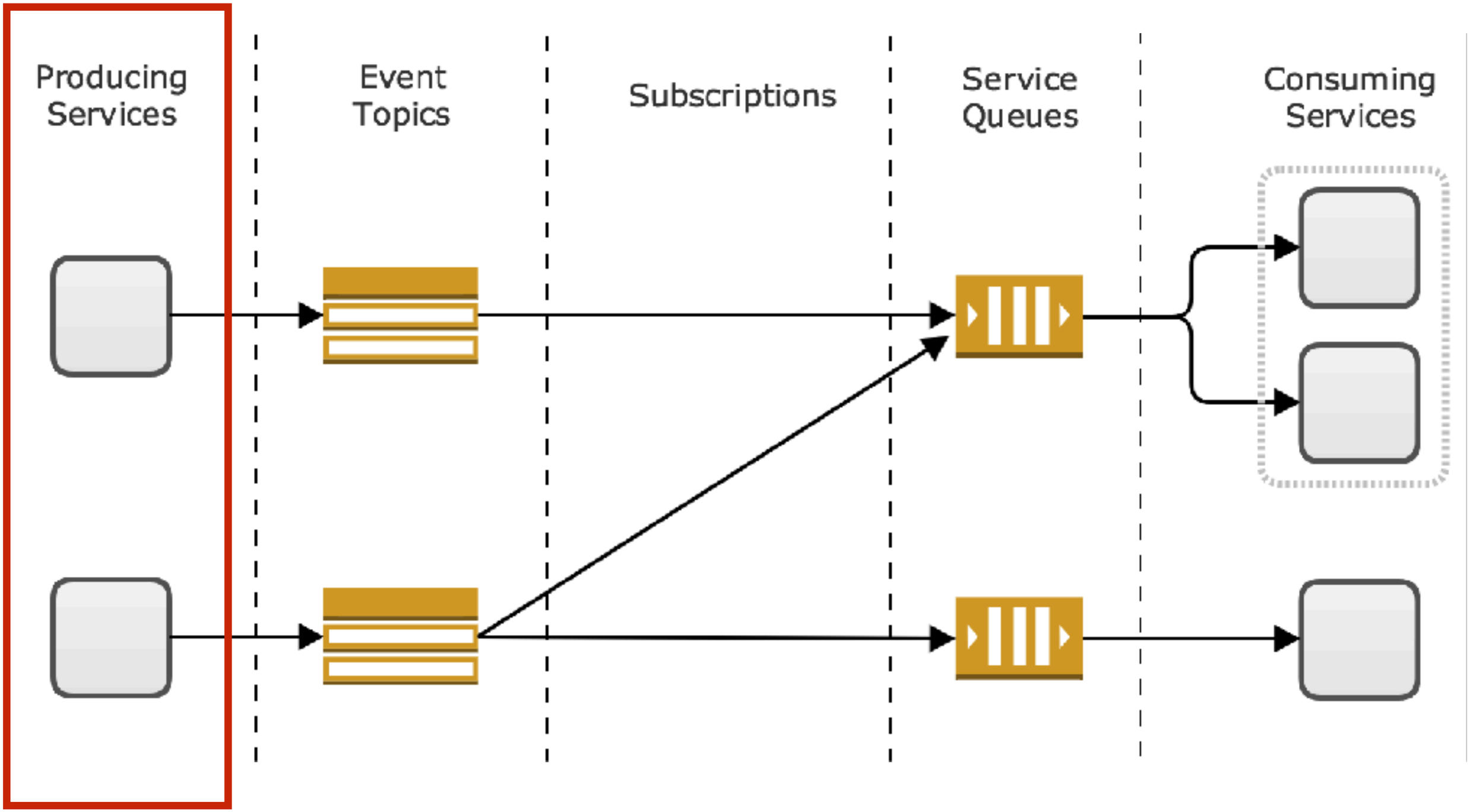
Request-response model



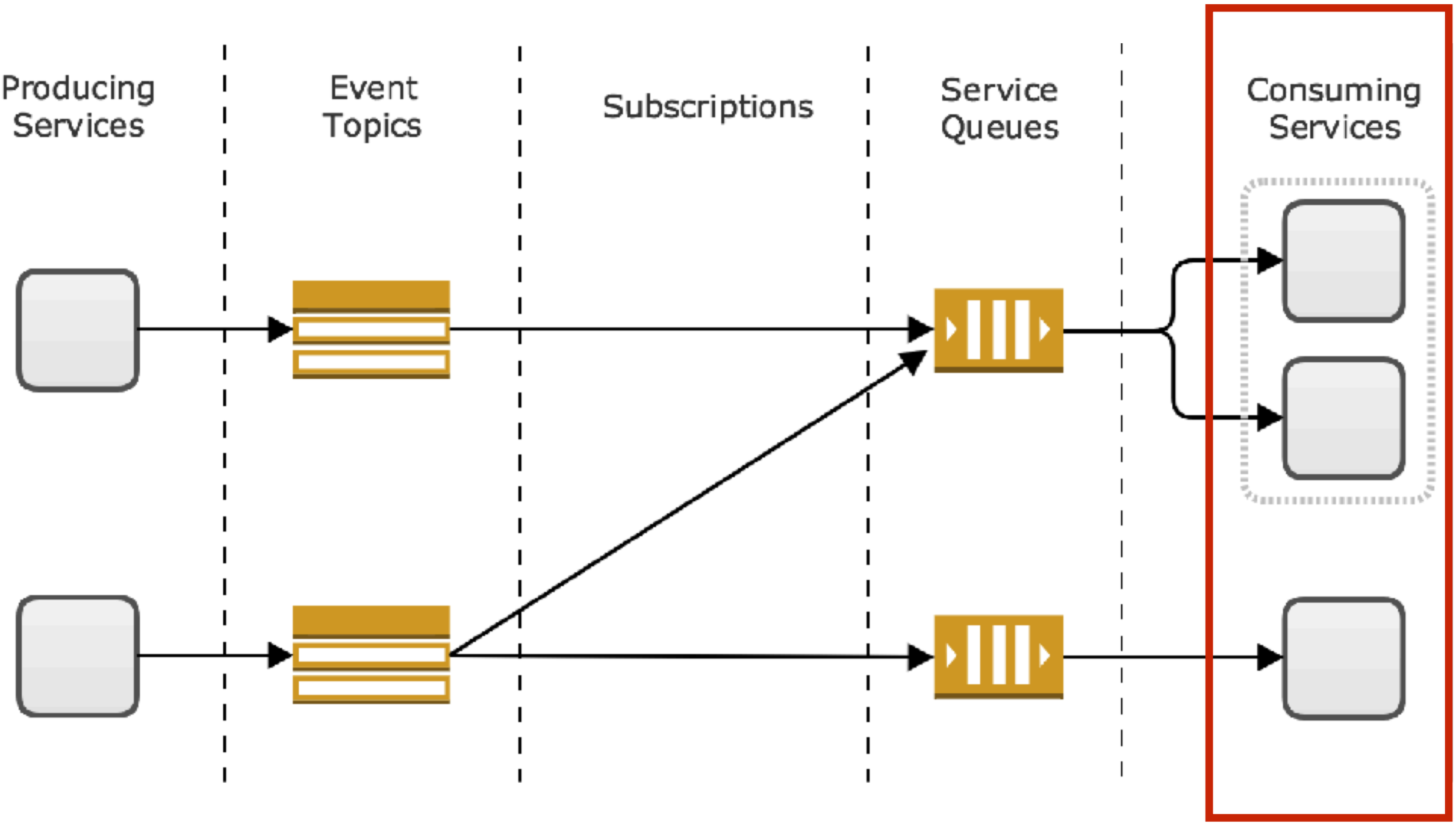
Request-response model



Observer model

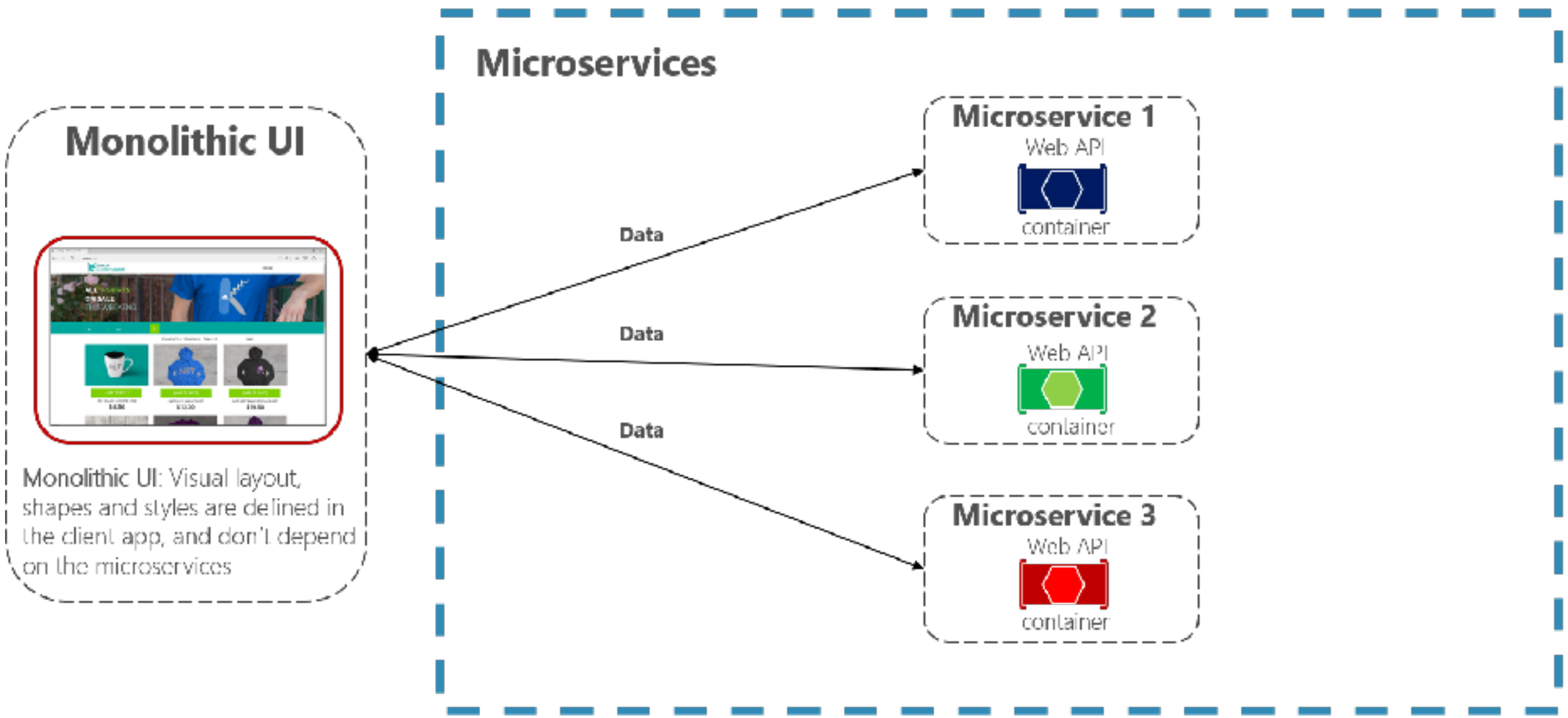


Observer model

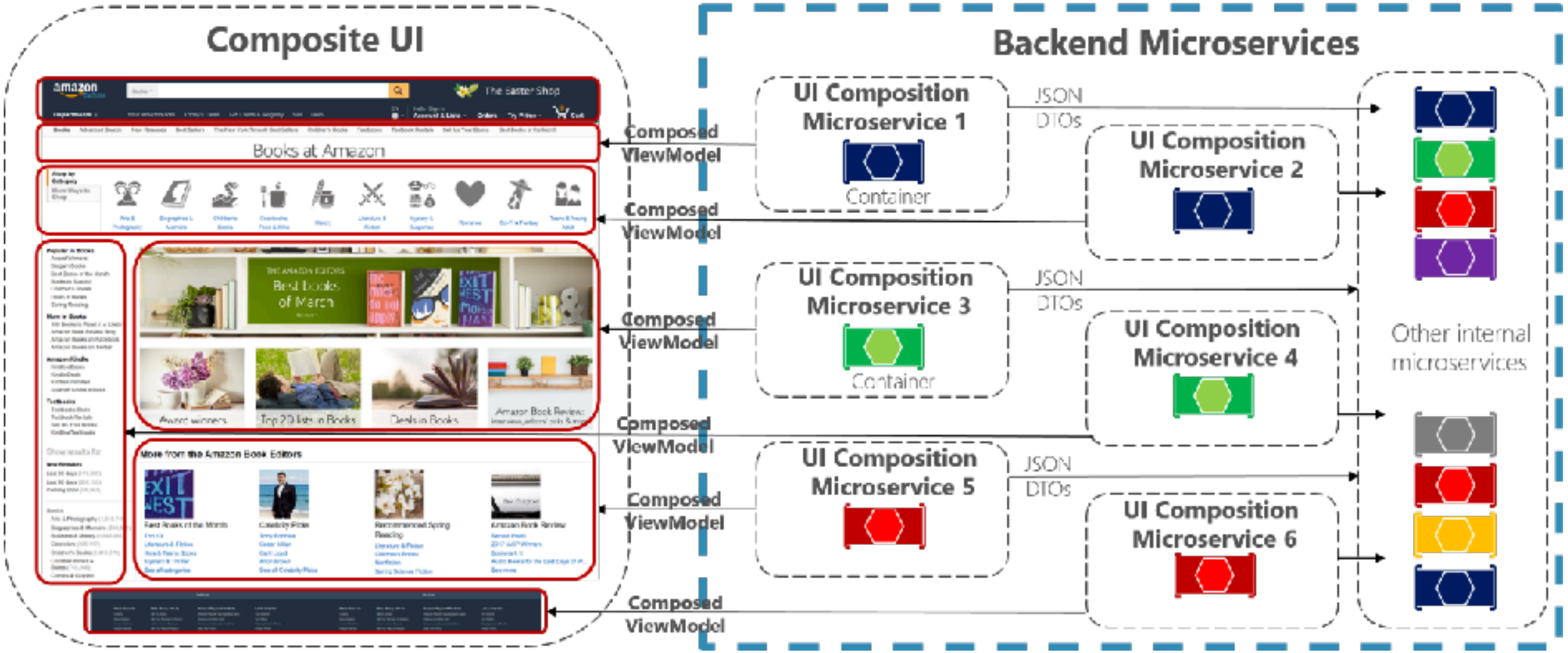


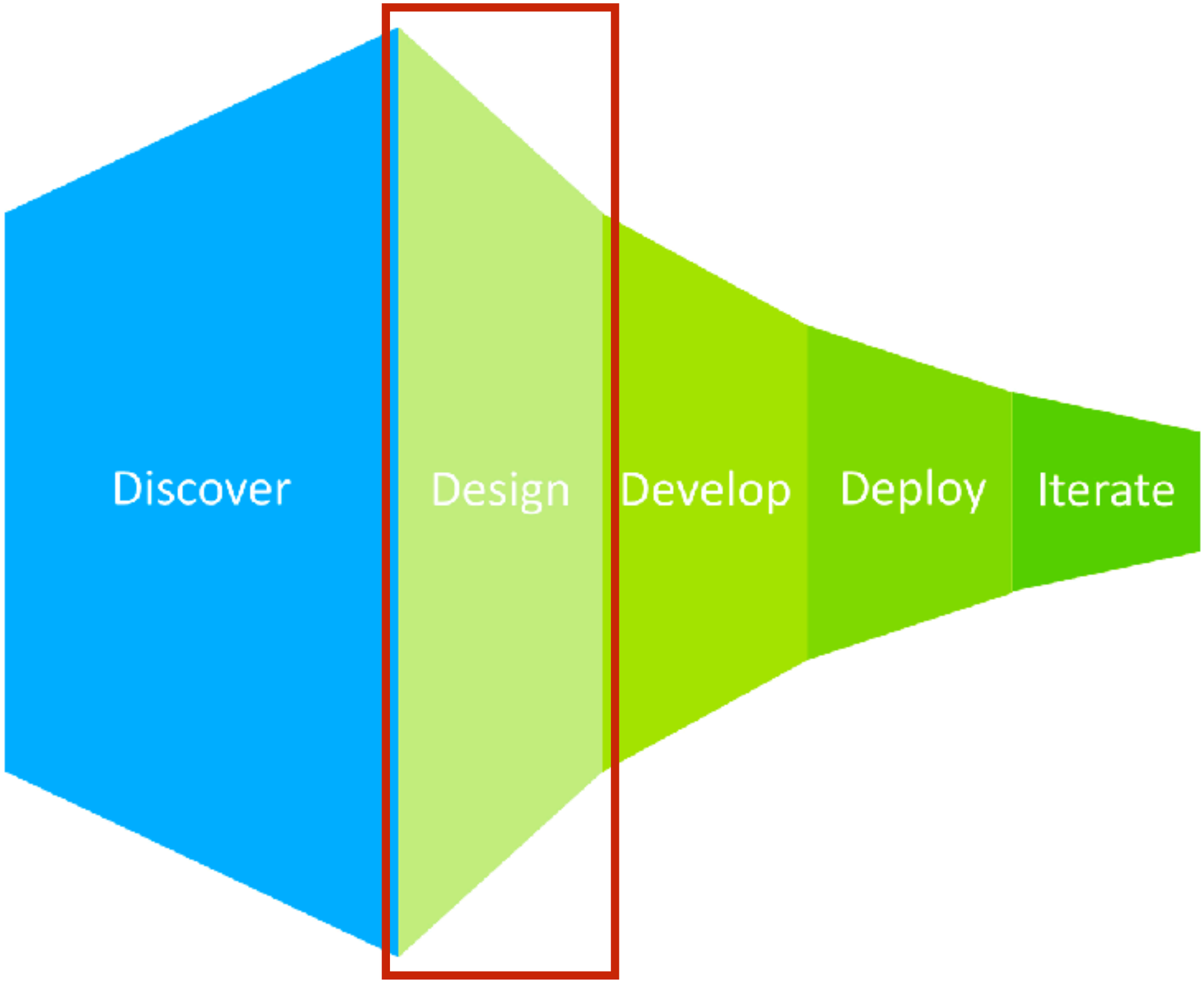
Integrate with User Interface

Monolithic UI consuming microservices



Composite UI *generated* by microservices

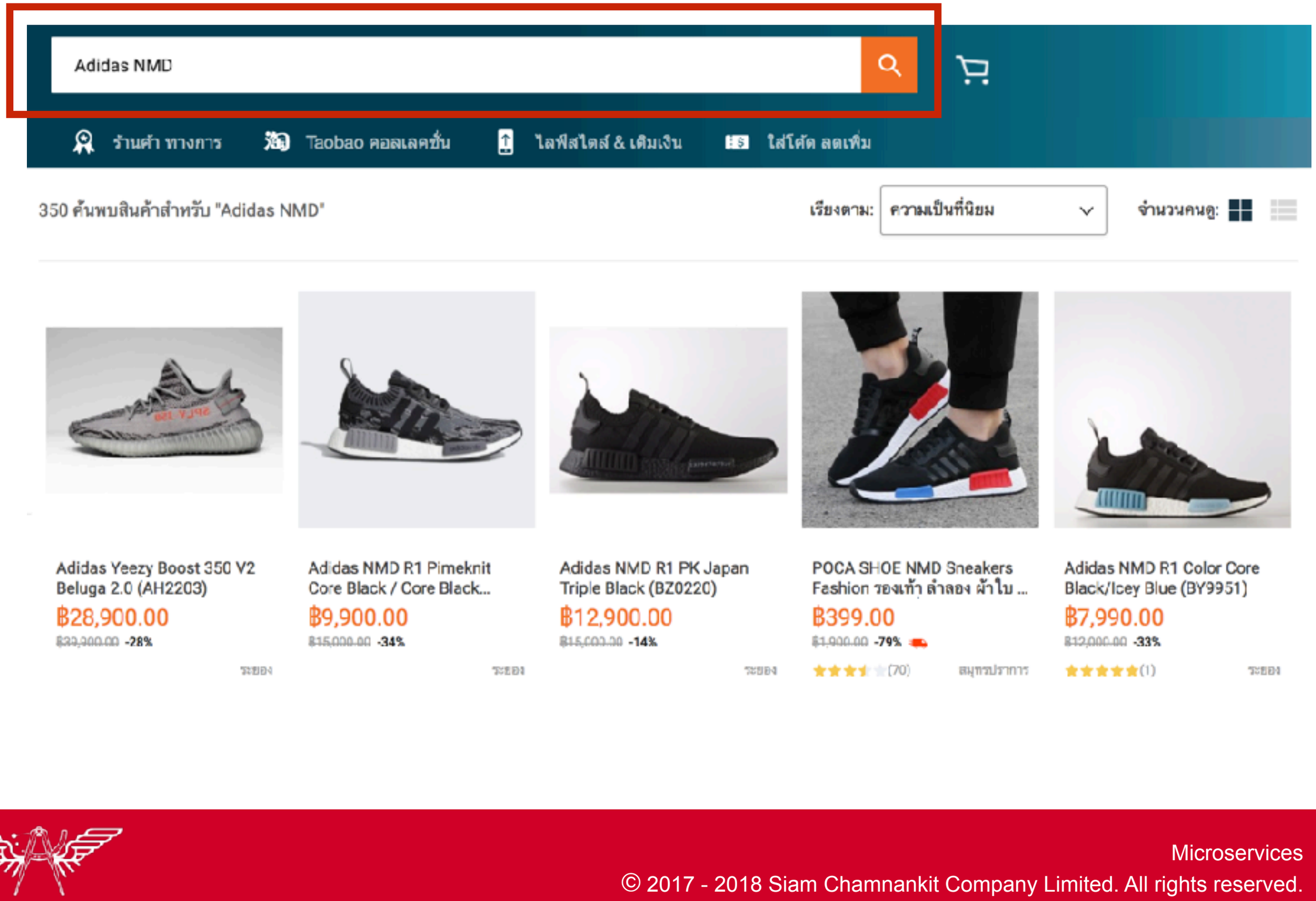




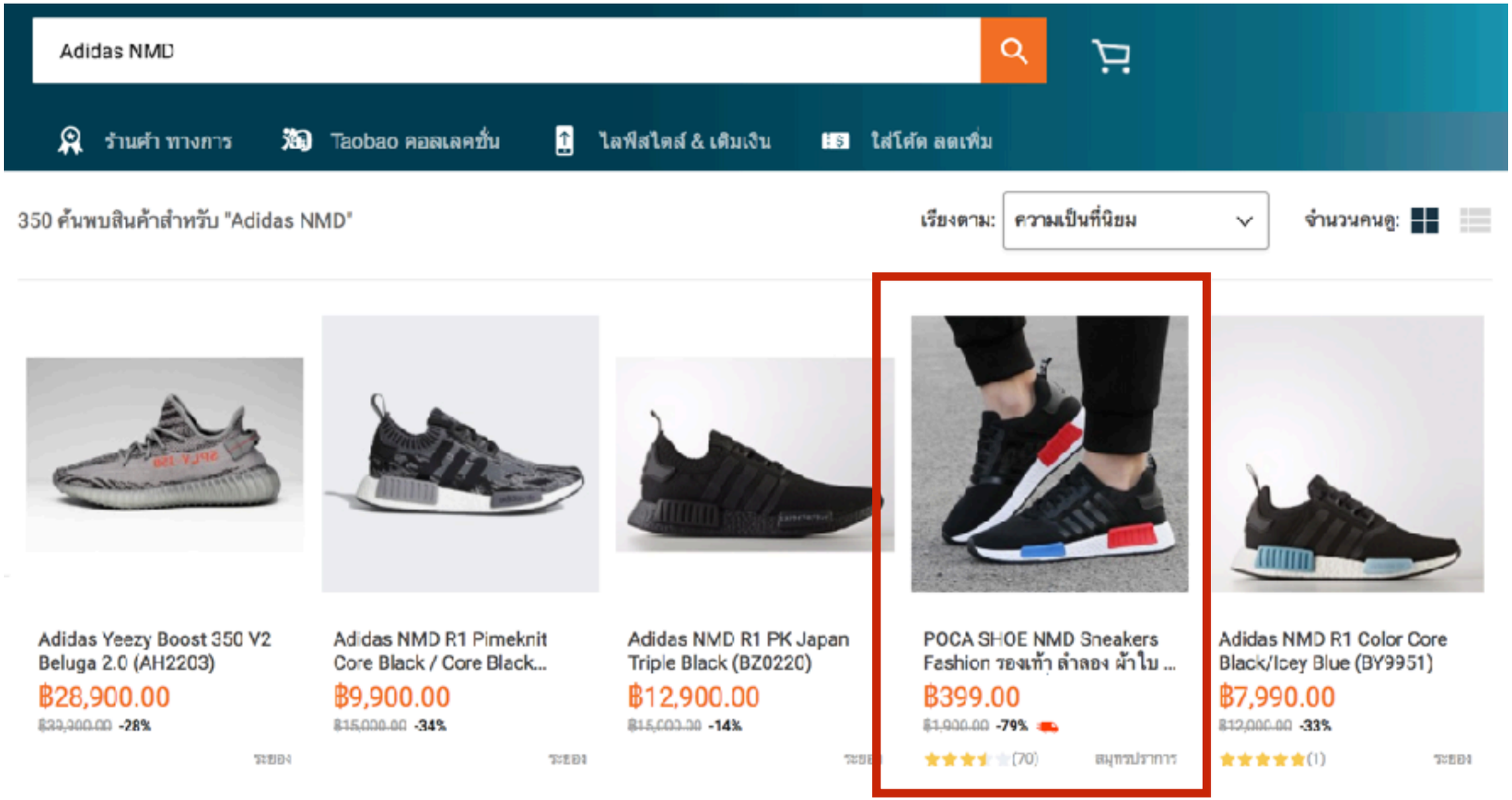
Let’s workshop with Design

E-commerce system

1. Search product by name



2. Choose a product

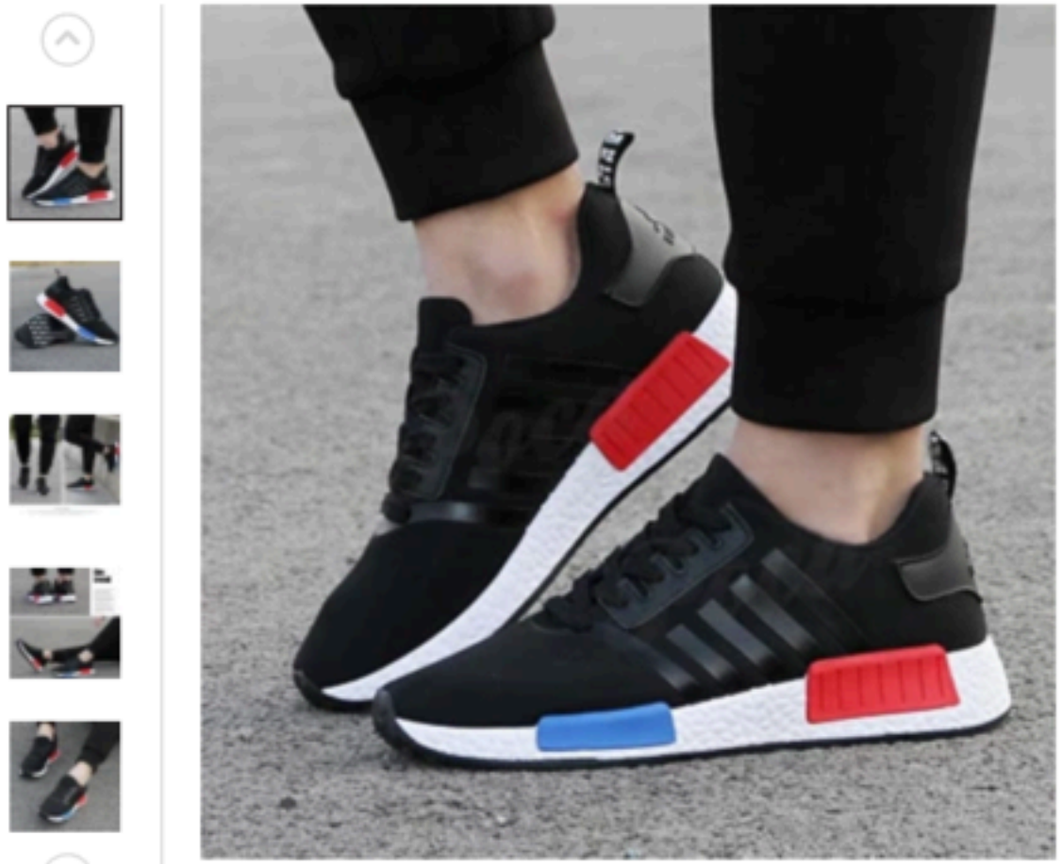


3. Show product detail

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 ไม่มี

399 บาท

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

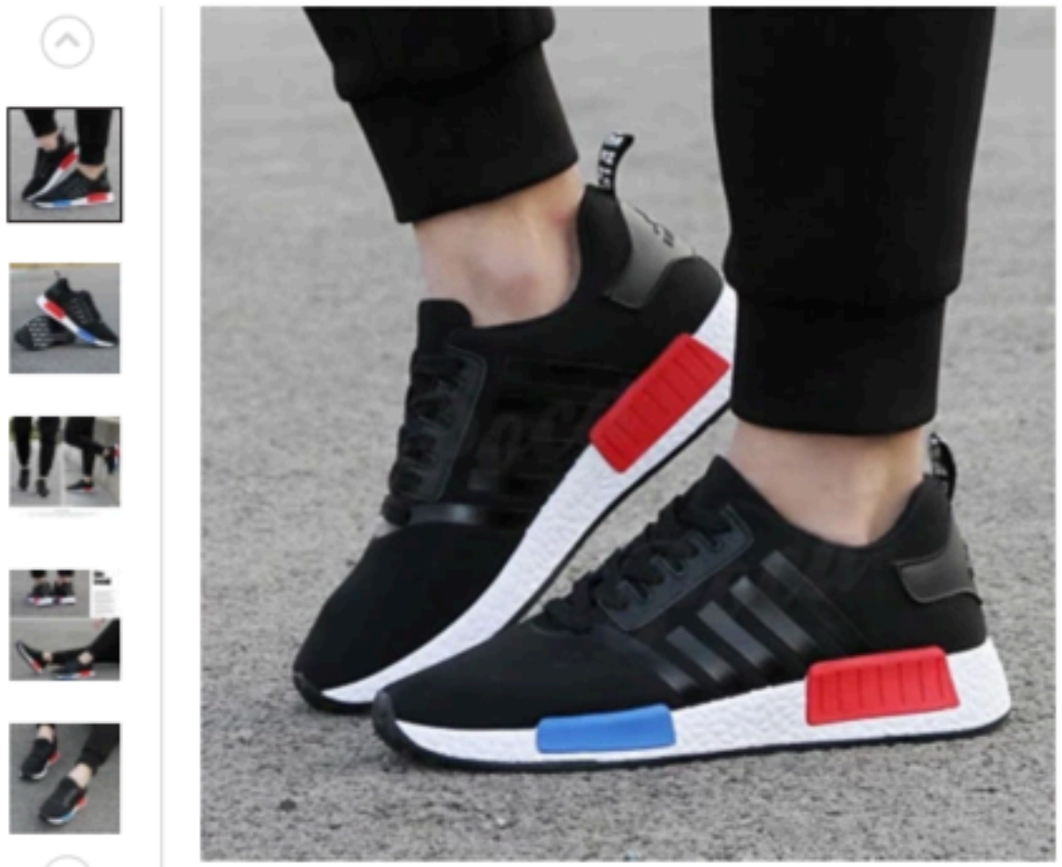
ใส่ตะกร้า วิธีการสั่งซื้อ

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

LAZADA .CO.TH

1. ค่าซื้อขั้นต่ำ | 2. ชำระเงิน

ที่อยู่ที่จะจัดส่ง Login for speedy checkout

อีเมล: กรุณาใส่ อีเมล์ ของท่าน

ชื่อ และ นามสกุล: ชื่อและนามสกุล

ที่อยู่: ที่อยู่

รหัสไปรษณีย์: รหัสไปรษณีย์ ทางเราจะทำการตรวจสอบเมืองและจังหวัดของคุณ

เขต: เขต

จังหวัด: กรุงเทพมหานคร/ Bangkok

โทรศัพท์มือถือ: +66 เบอร์โทรติดต่อ เพื่อให้มั่นใจว่าทำการจัดส่งได้

☐ ที่อยู่สำหรับใบเสร็จ/ใบกำกับภาษี - กรุณาเลือกและกรอกข้อมูลเพื่อทำการขอใบกำกับภาษี

ข้อมูลการส่งสินค้า

ส่งแบบธรรมดา: ฟรี

Get it วันอังคาร, 27 ก.พ. - วันจันทร์, 5 มี.ค. 2018

ดำเนินการต่อ

สรุปการสั่งซื้อ (1 items)

สินค้า	จำนวน	ราคา
POCA SHOE NMD Sneakers Fashion รองเท้า ลำลอง ผ้าใบ ผู้หญิง-ผู้ชาย แฟชั่น ราคาถูกสวยๆ Sport Unisex รุ่น PSN-Black/White ขนาด: EU:40	1	399

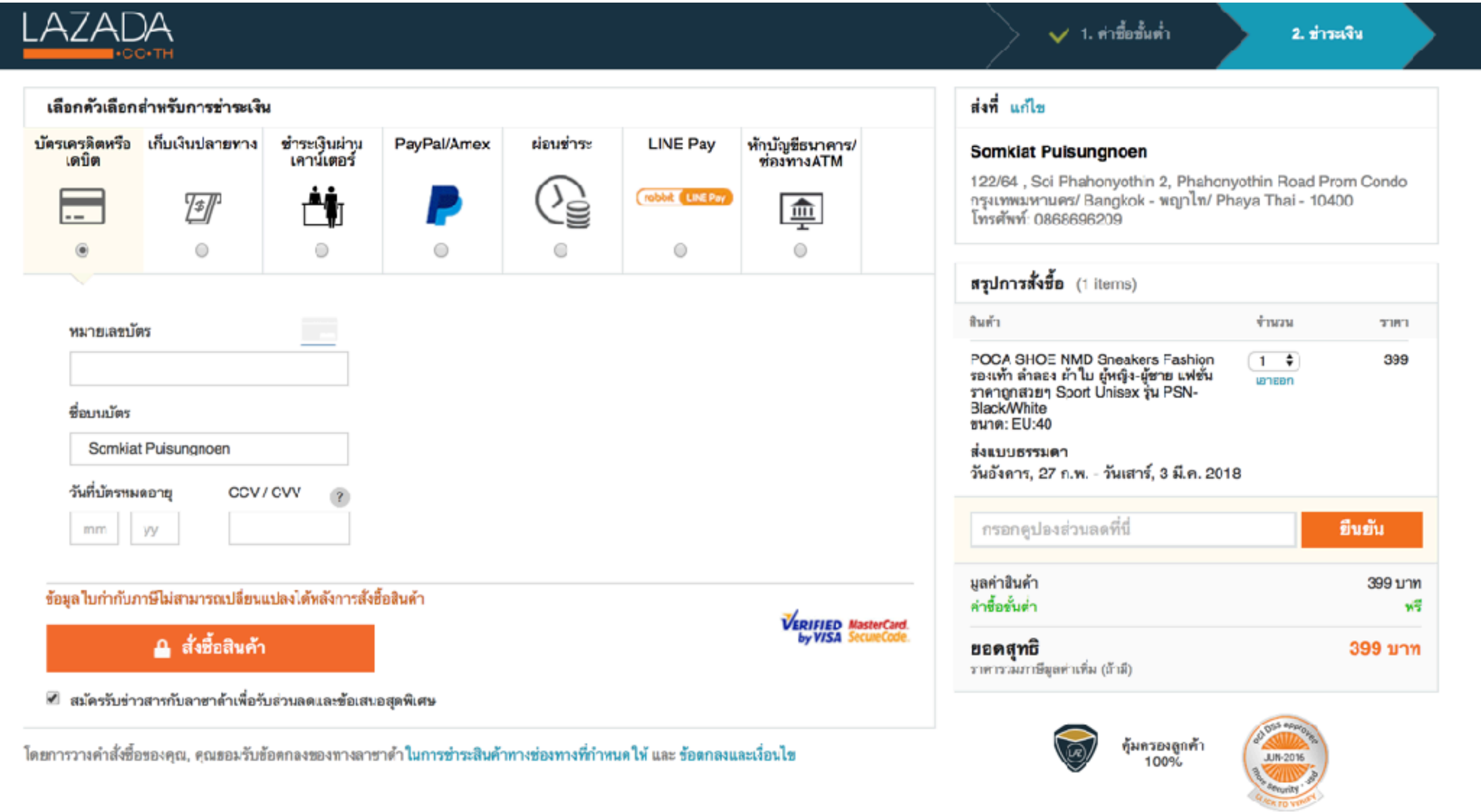
มูลค่าสินค้า 399 บาท

ยอดสุทธิ 399 บาท

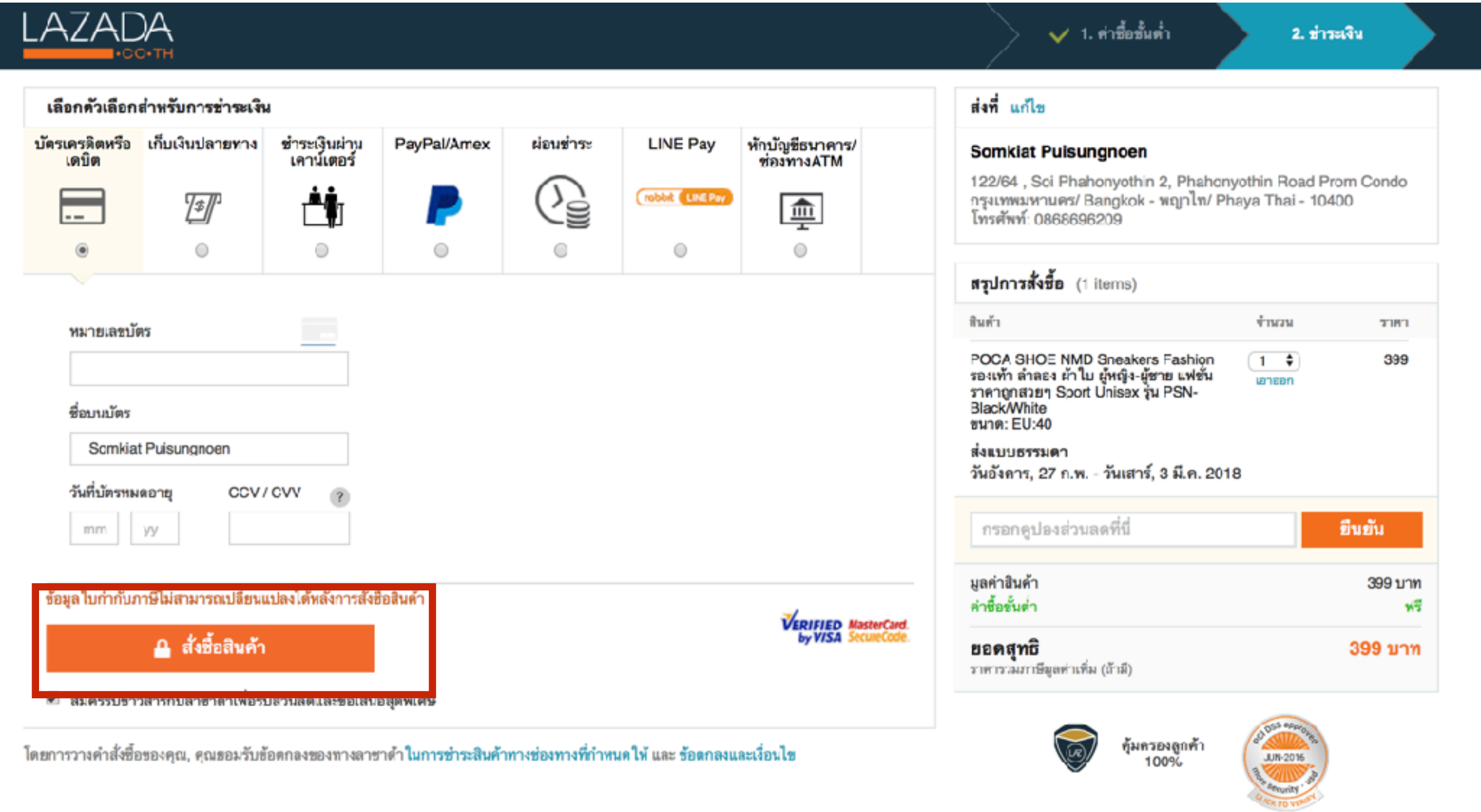
ราคารวมภาษีมูลค่าเพิ่ม (ถ้ามี)

คุ้มครองลูกค้า 100%

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:

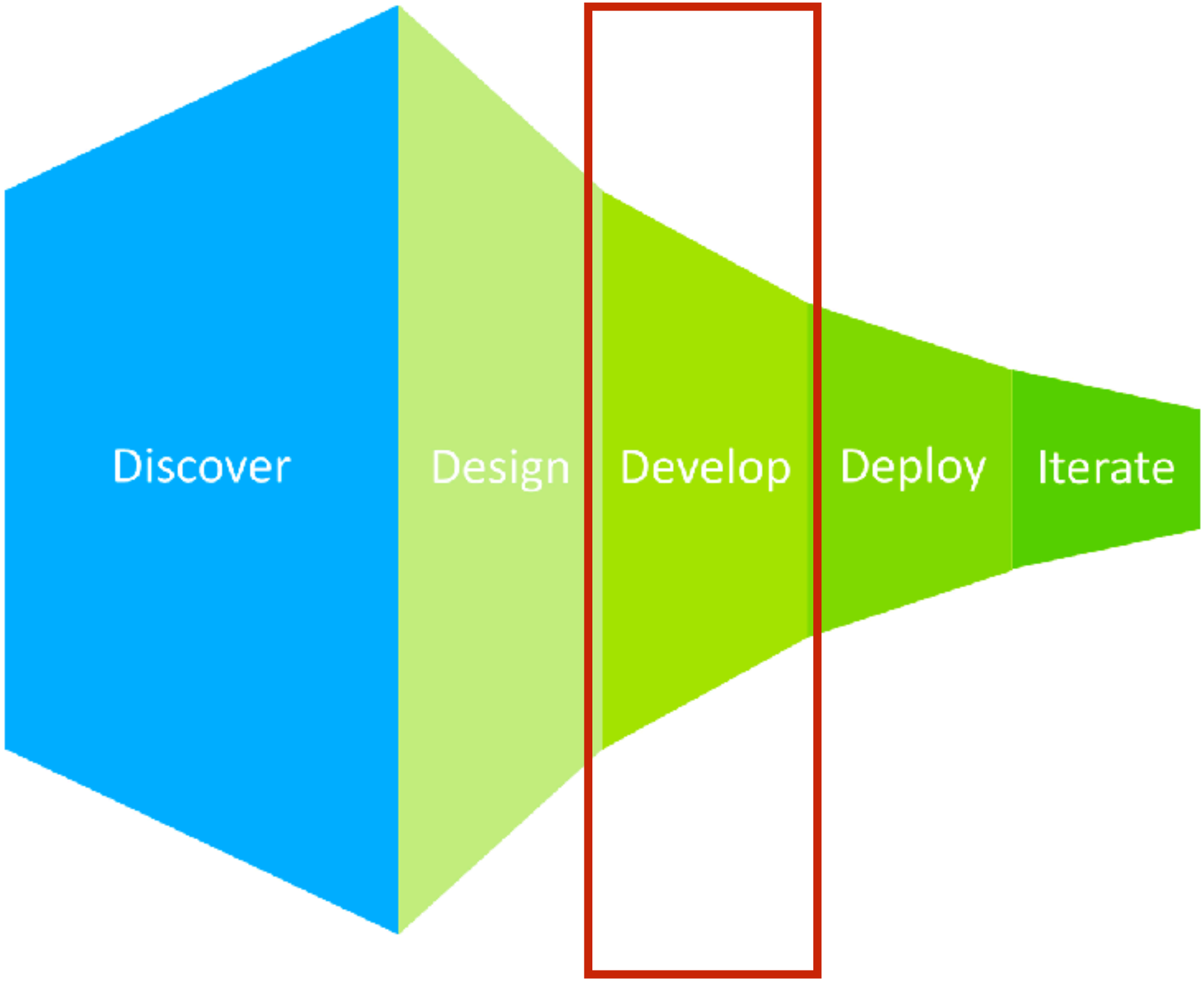
399.00 บาท /THB

* ไม่รวมค่าธรรมเนียมของเคาน์เตอร์เซอร์วิส
(Excluding service fees at Counter Service)

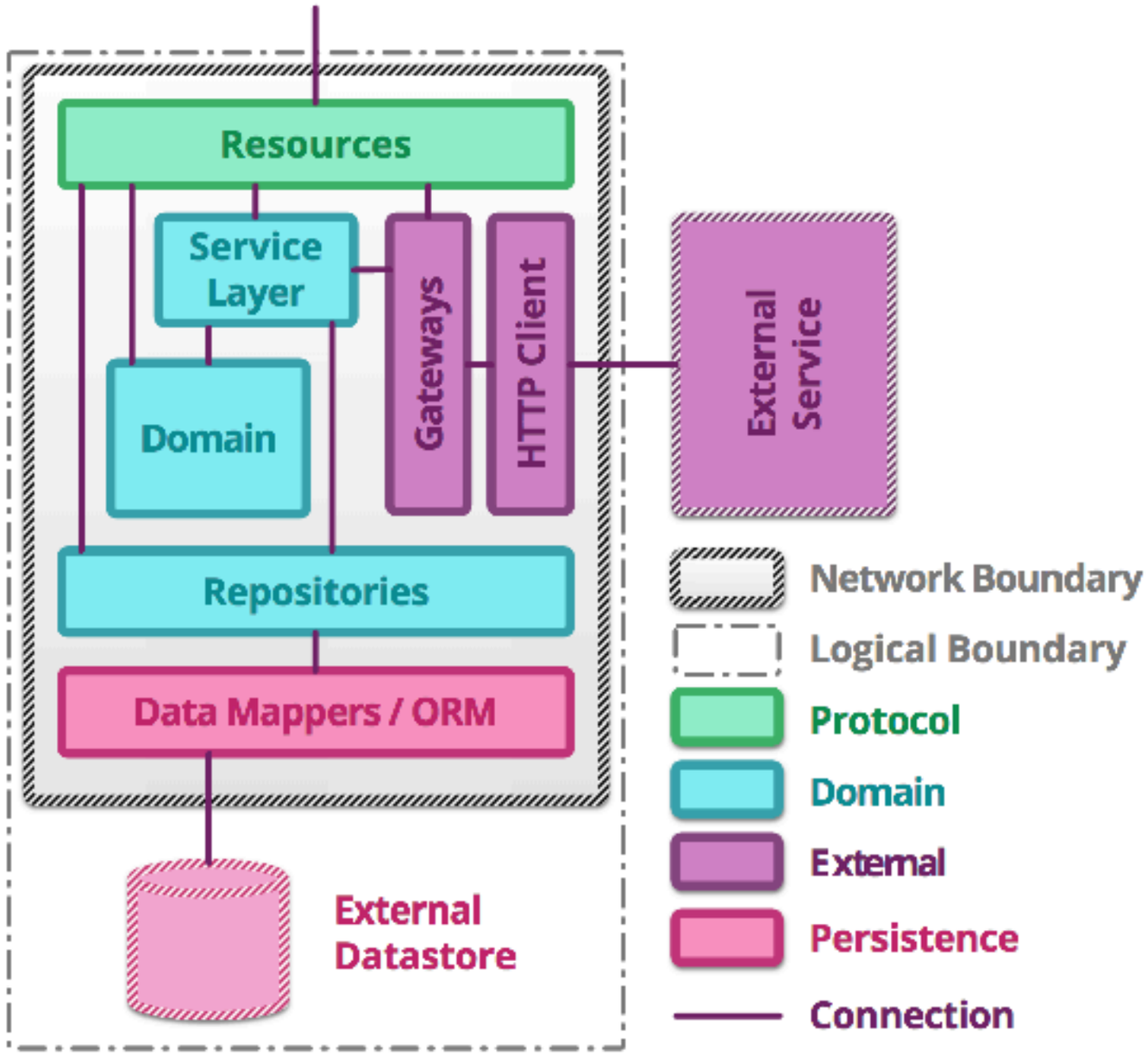
คลิกปุ่ม **"Print"** พิมพ์ใบแจ้งการชำระเงิน
หรือ
จด **"รหัส 15 หลักใต้บาร์โค้ด"** เพื่อนำไปชำระเงินที่
Press "Print" button or write down paycode 15 digits for pay in cash at counter service(7-11)

Back to merchant | Print

Try to design system

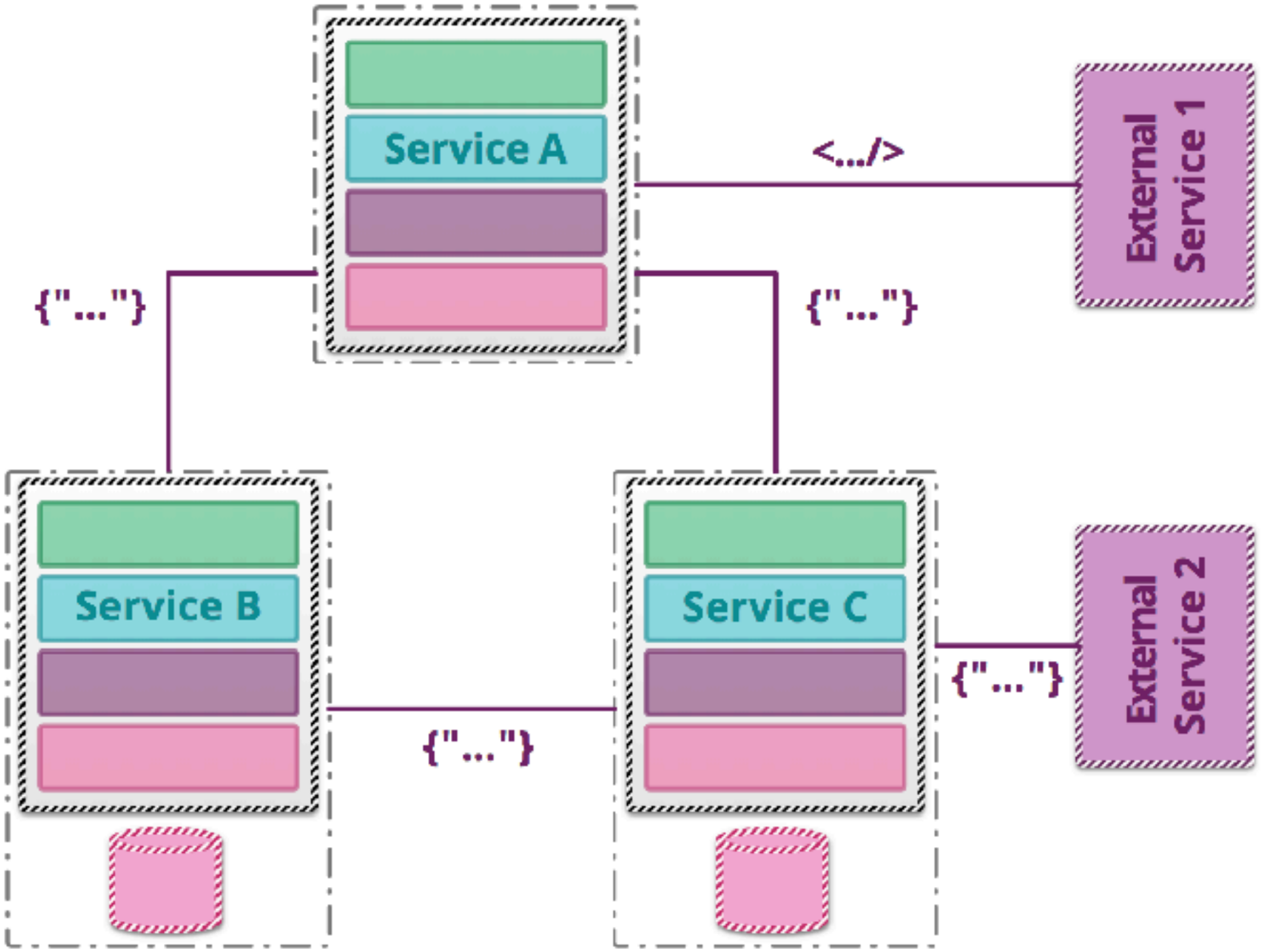


Service structure



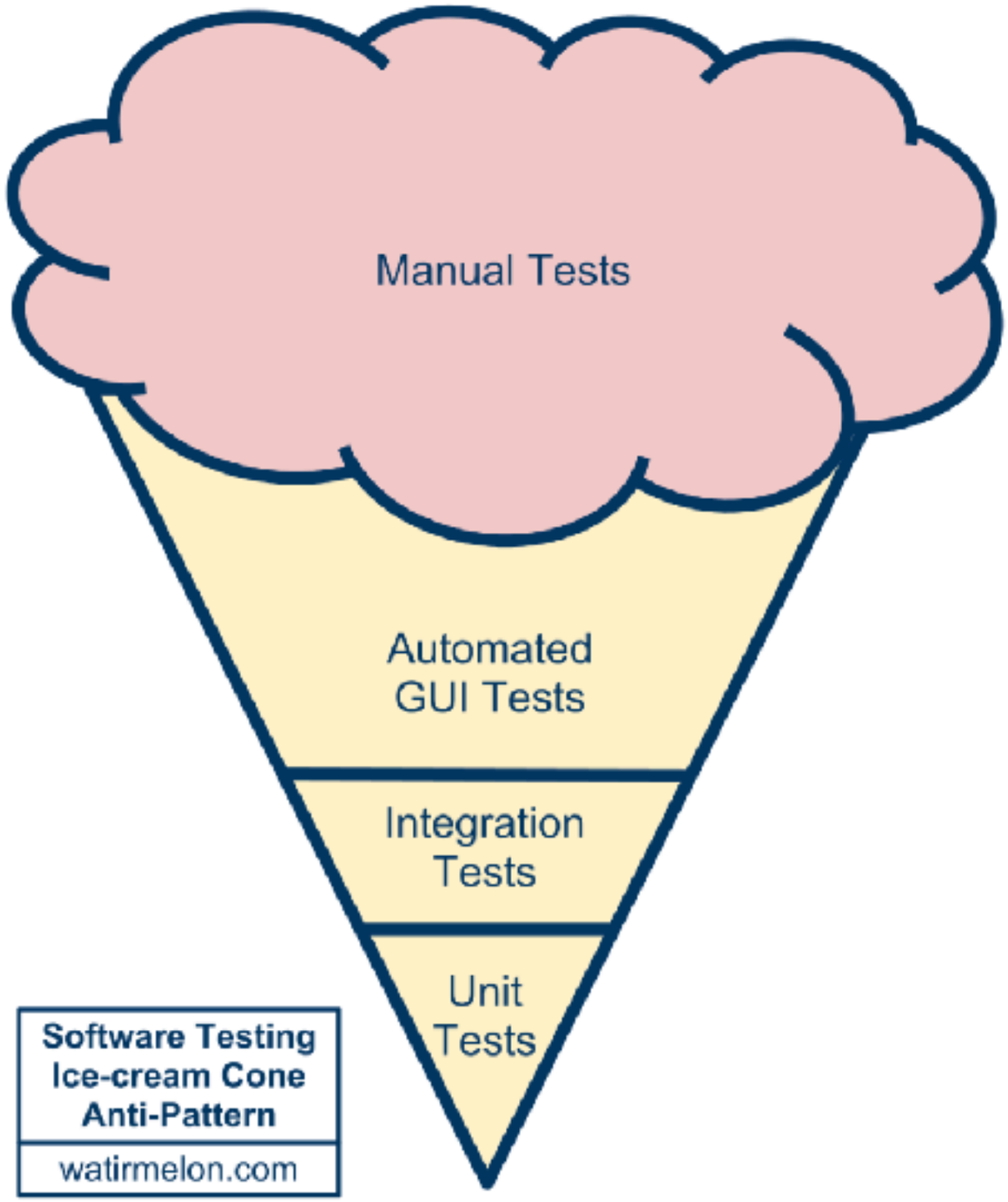
https://martinfowler.com/articles/microservice-testing

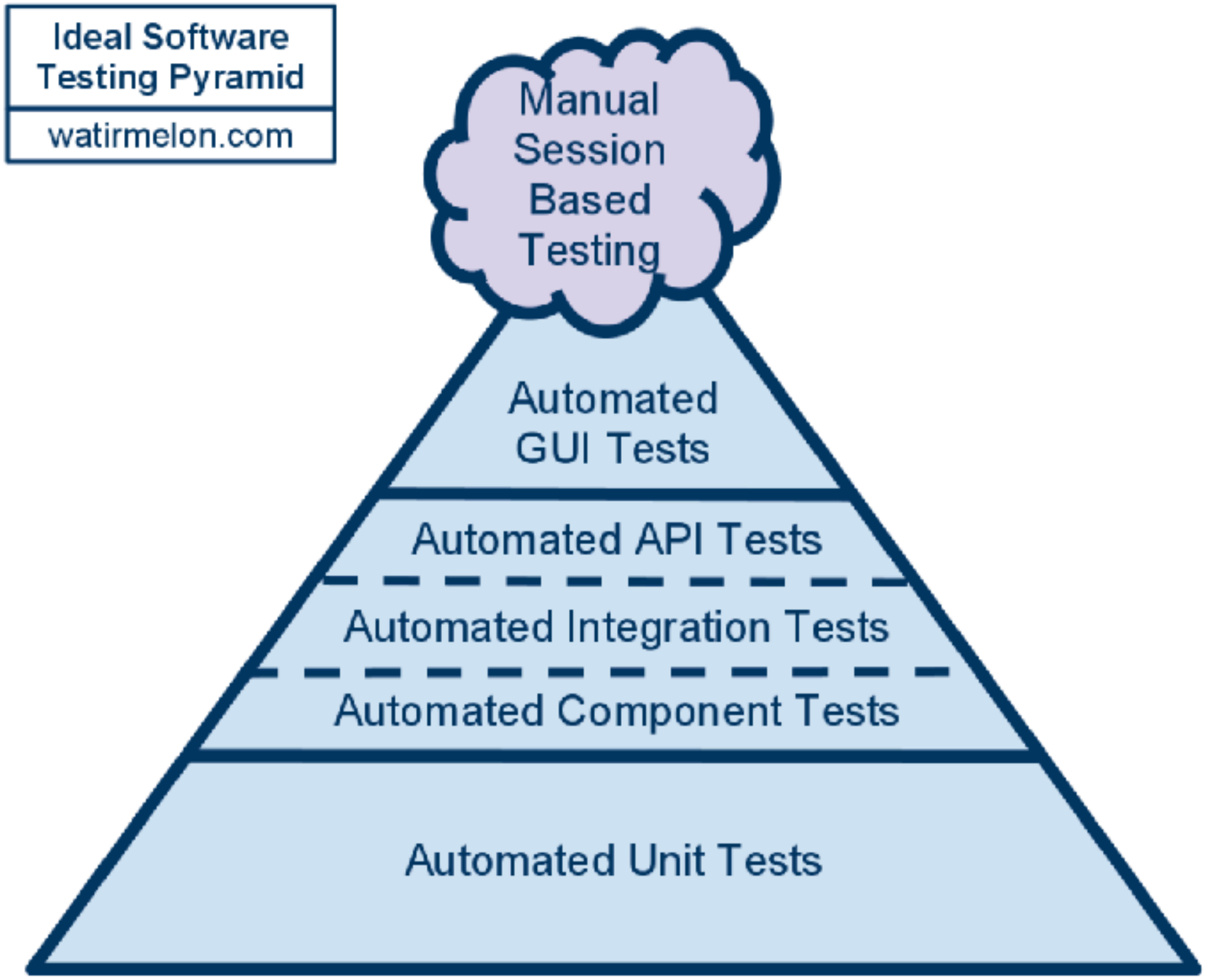
Multiple services

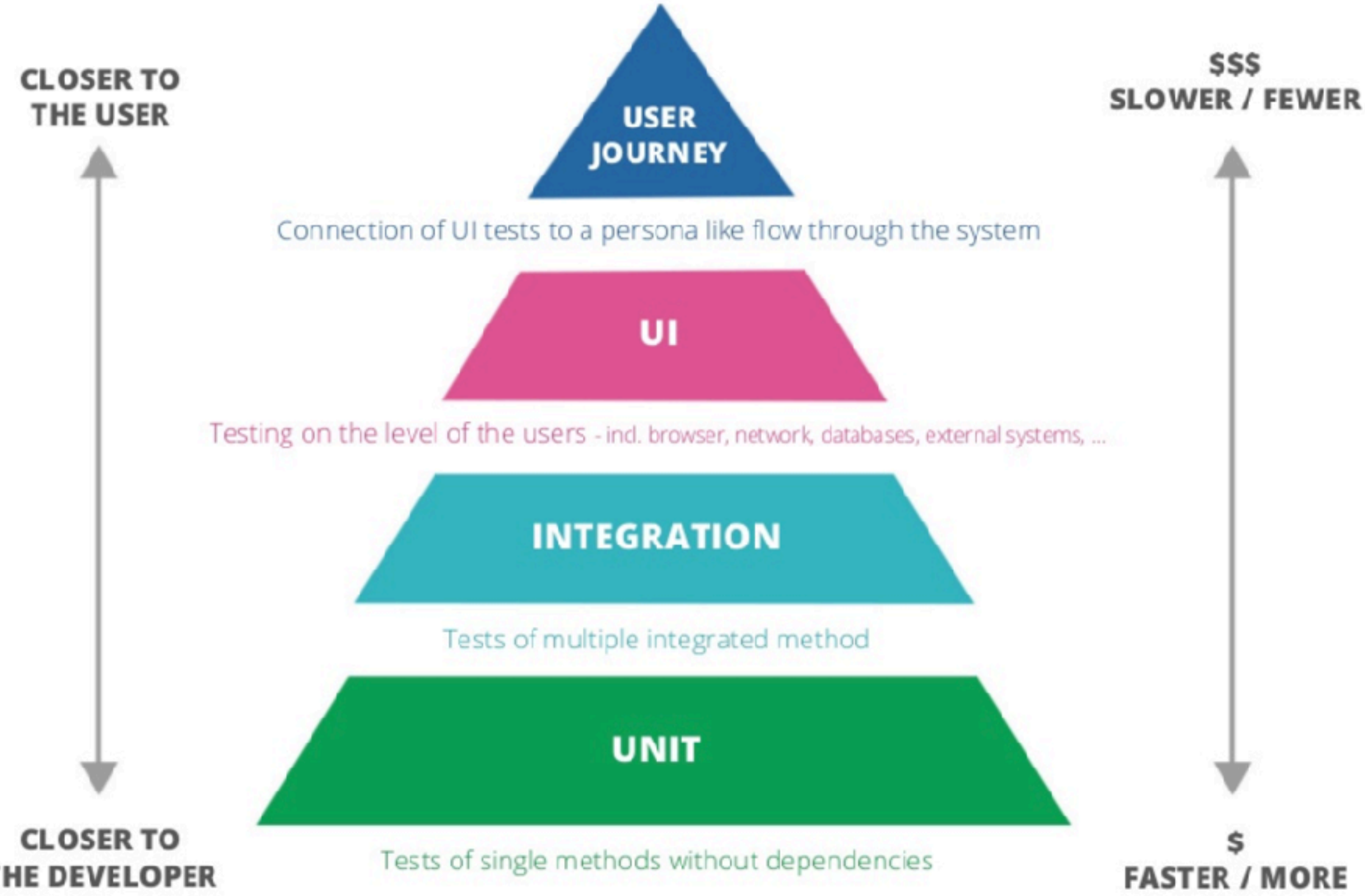


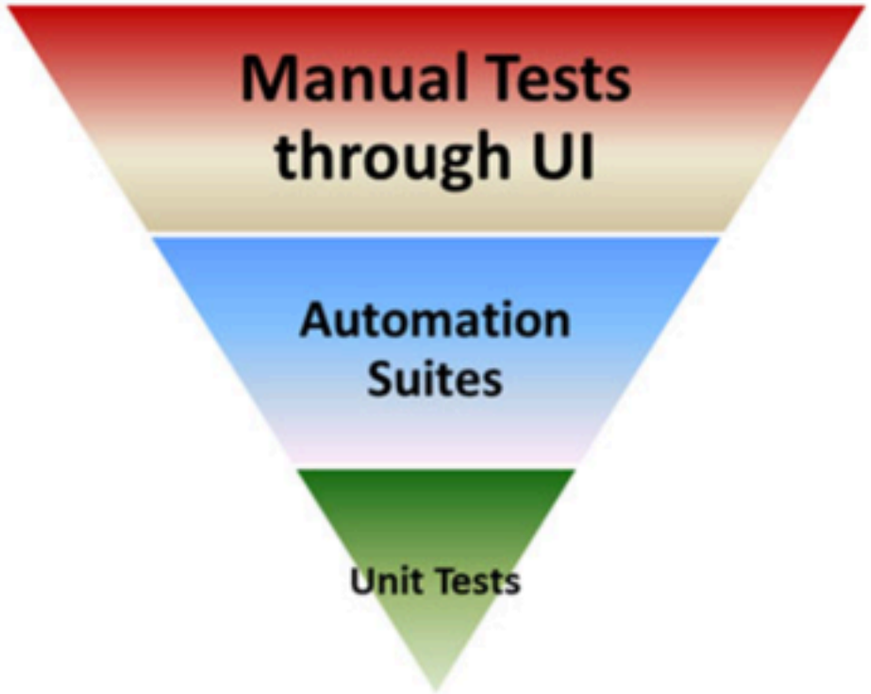
https://martinfowler.com/articles/microservice-testing

Microservice Testing

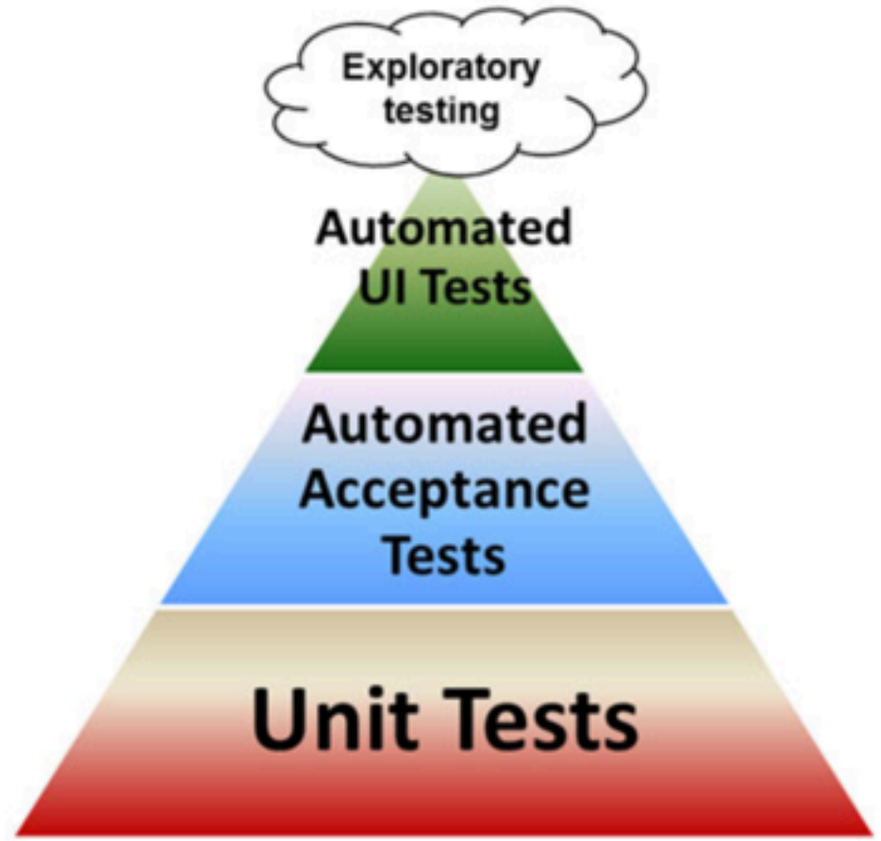




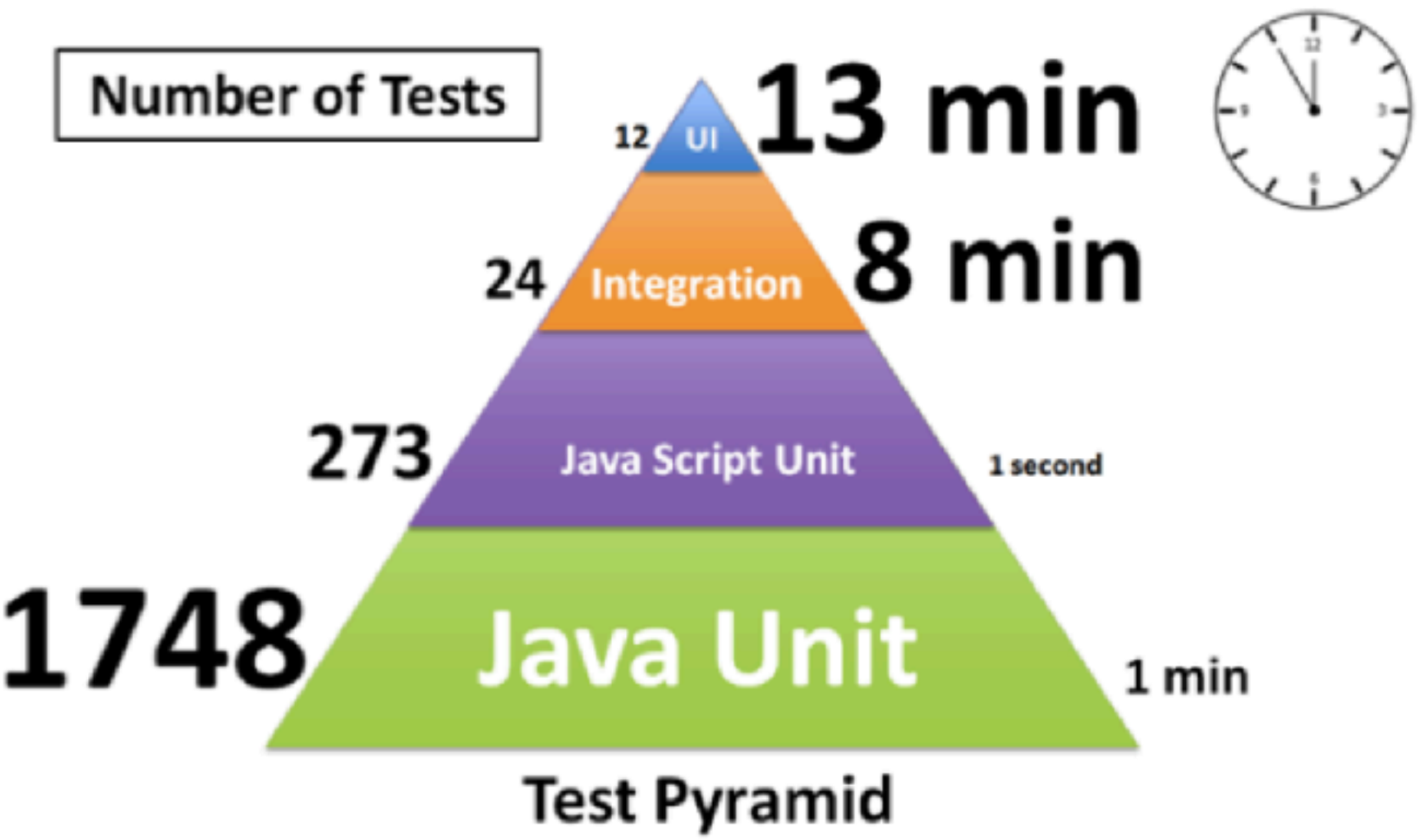




Traditional
(find bugs)

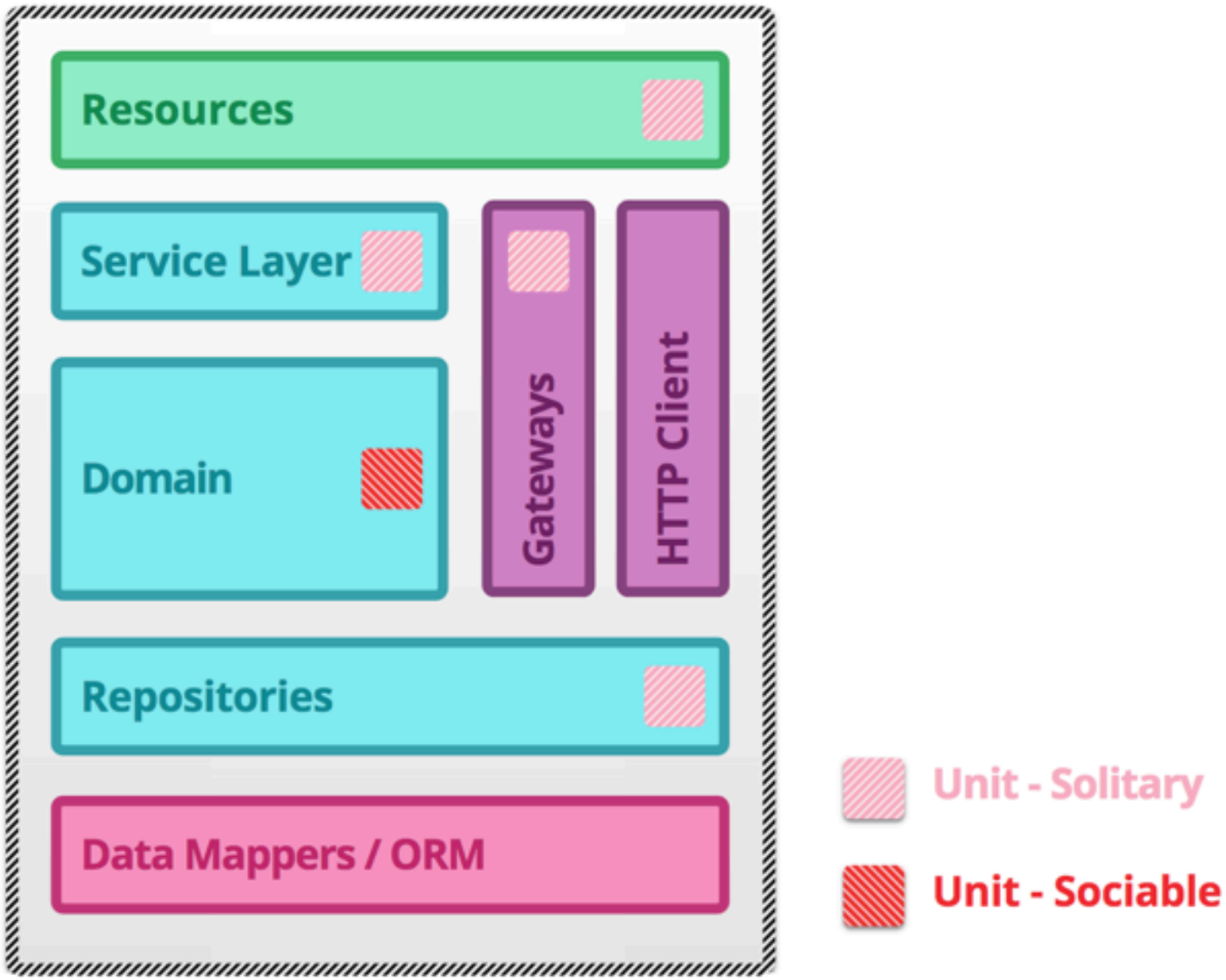


Agile
(prevent bugs)

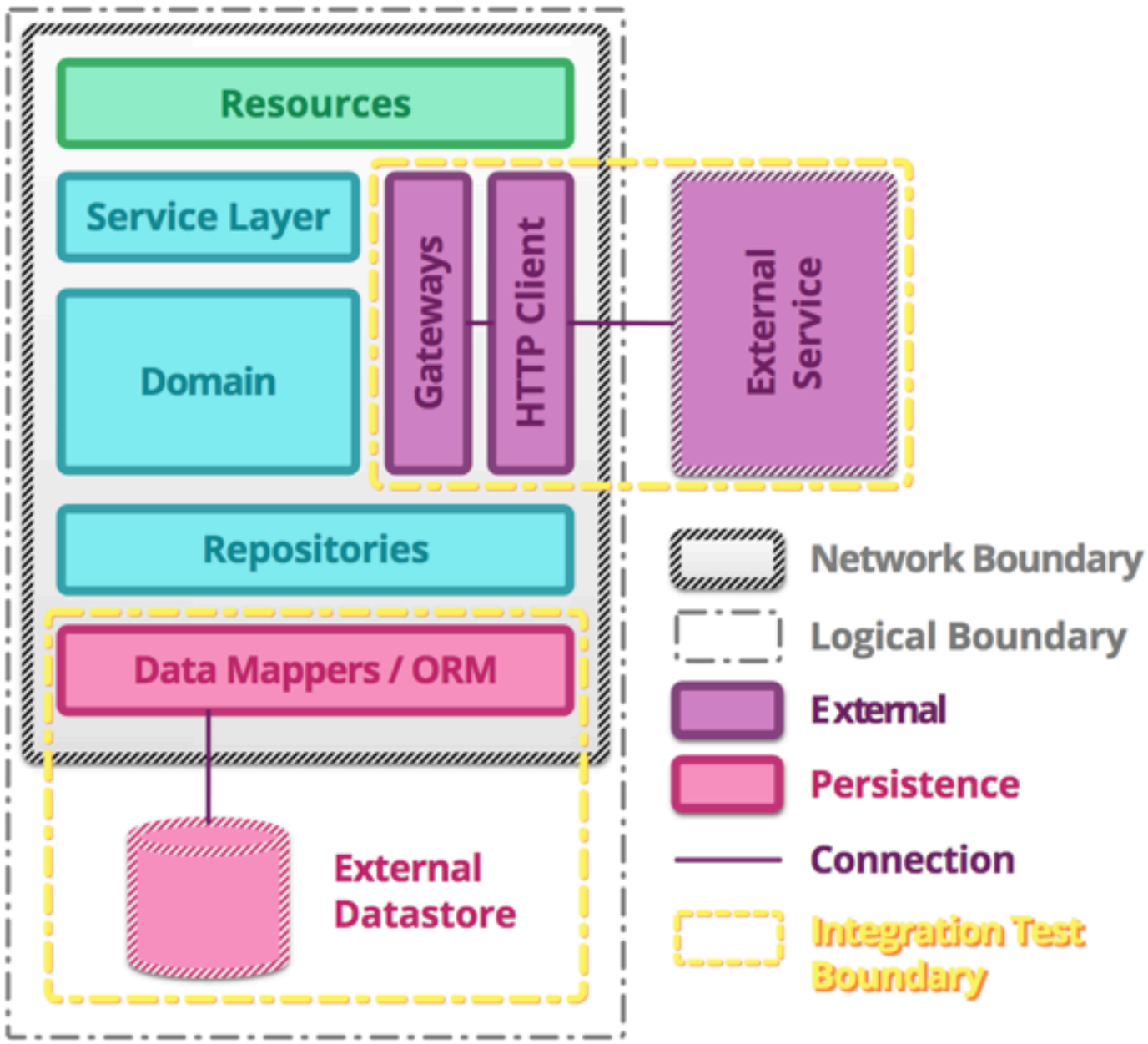


Test Pyramid

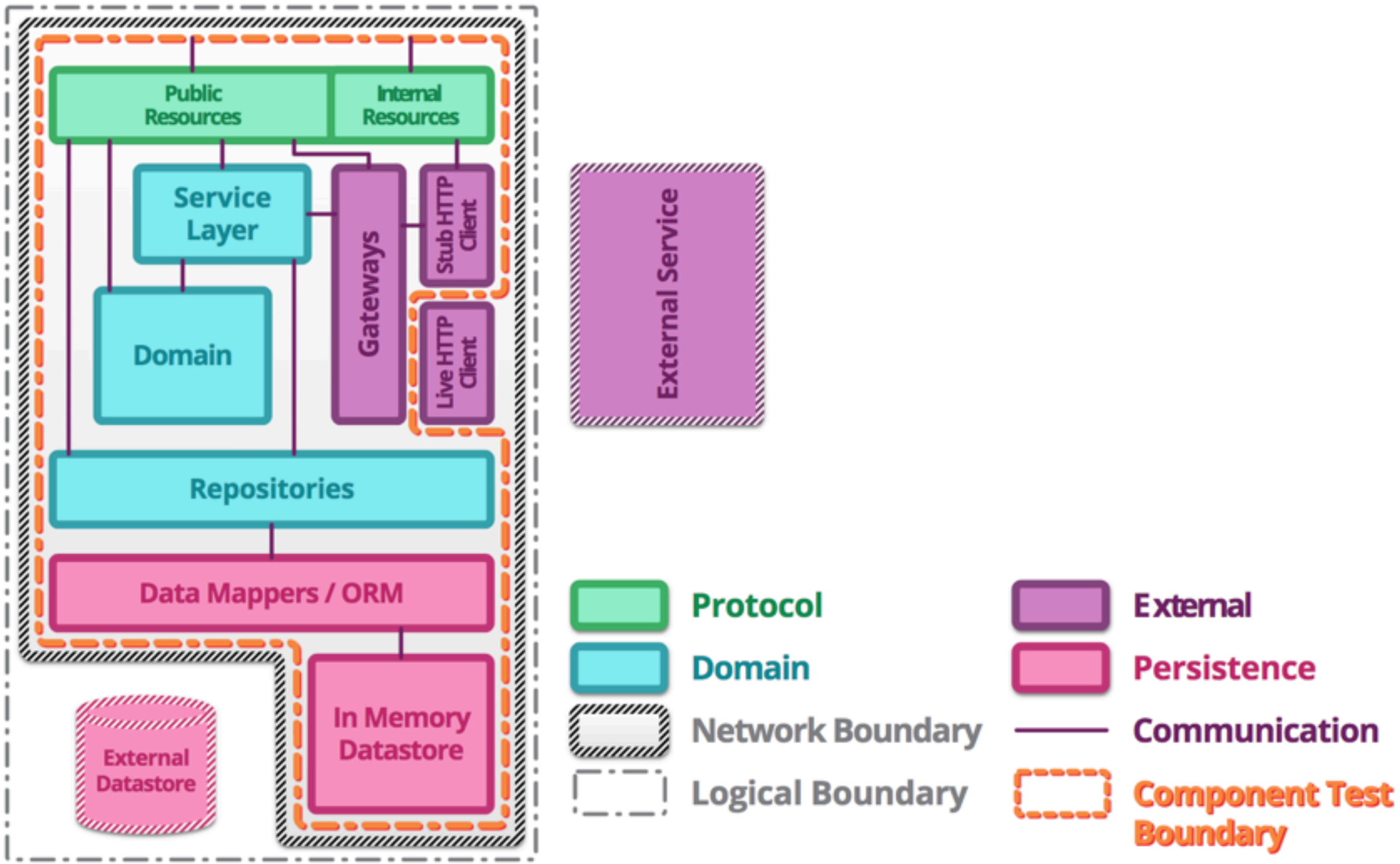
Unit testing



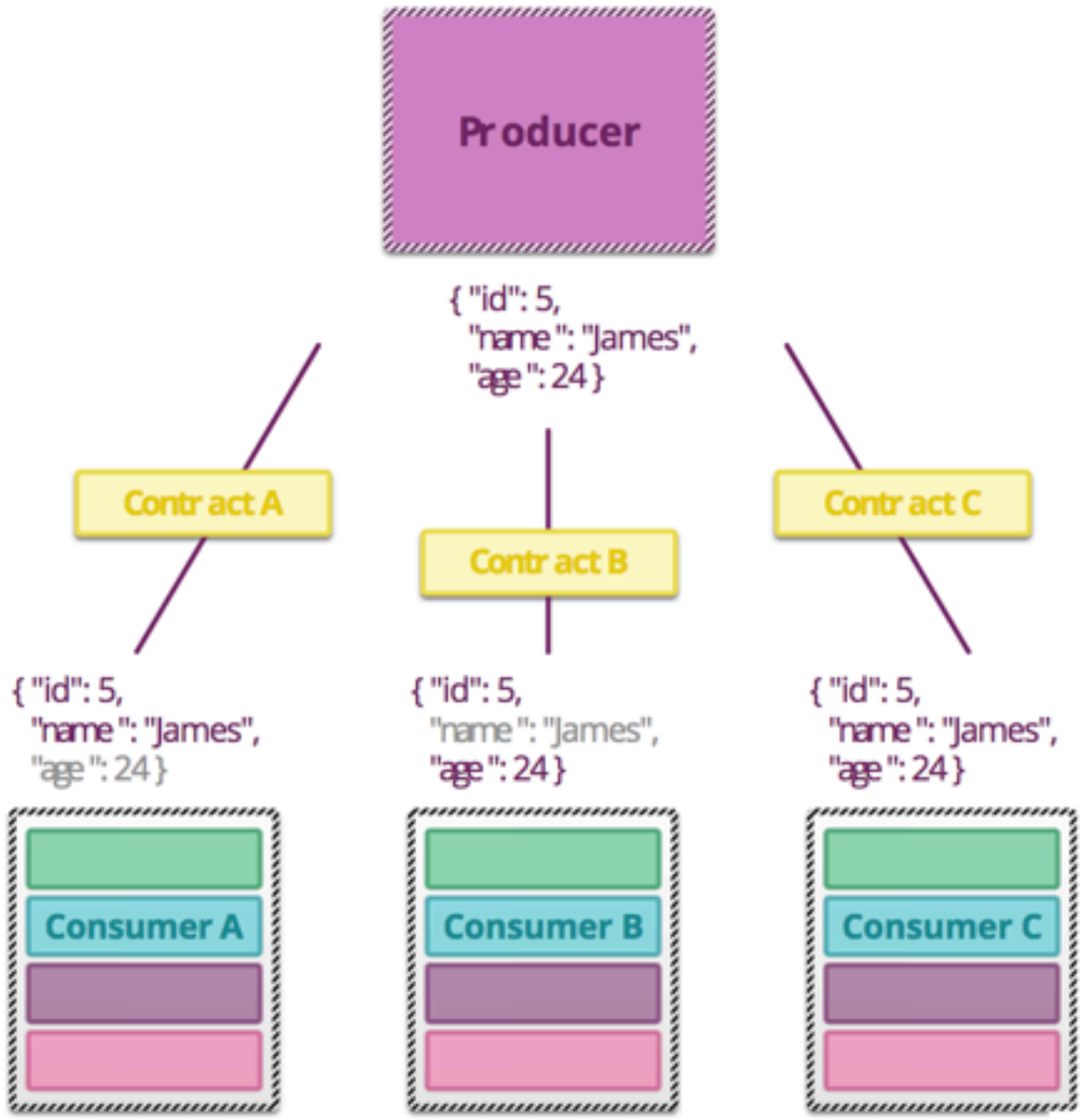
Integration testing



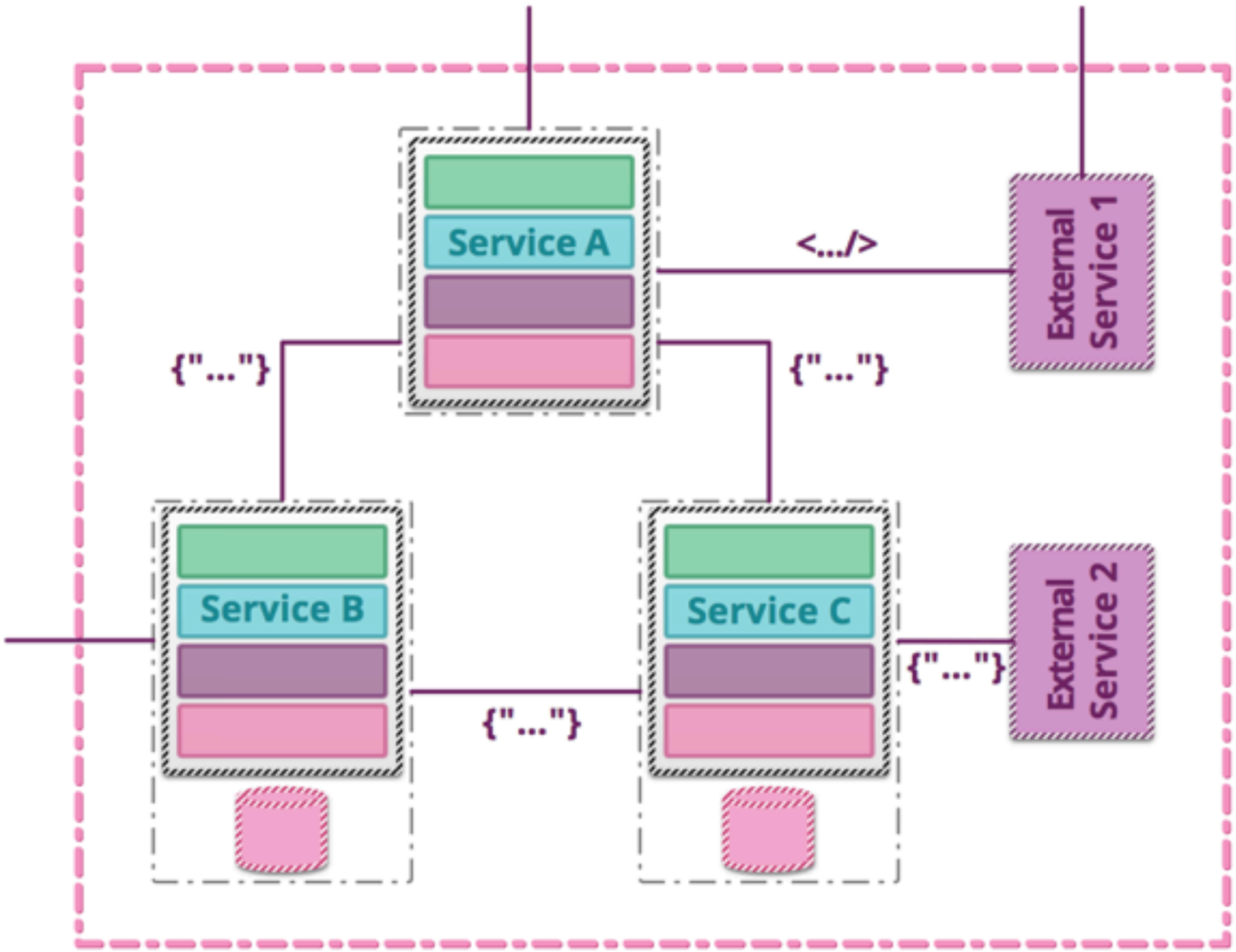
Component testing



Contract testing



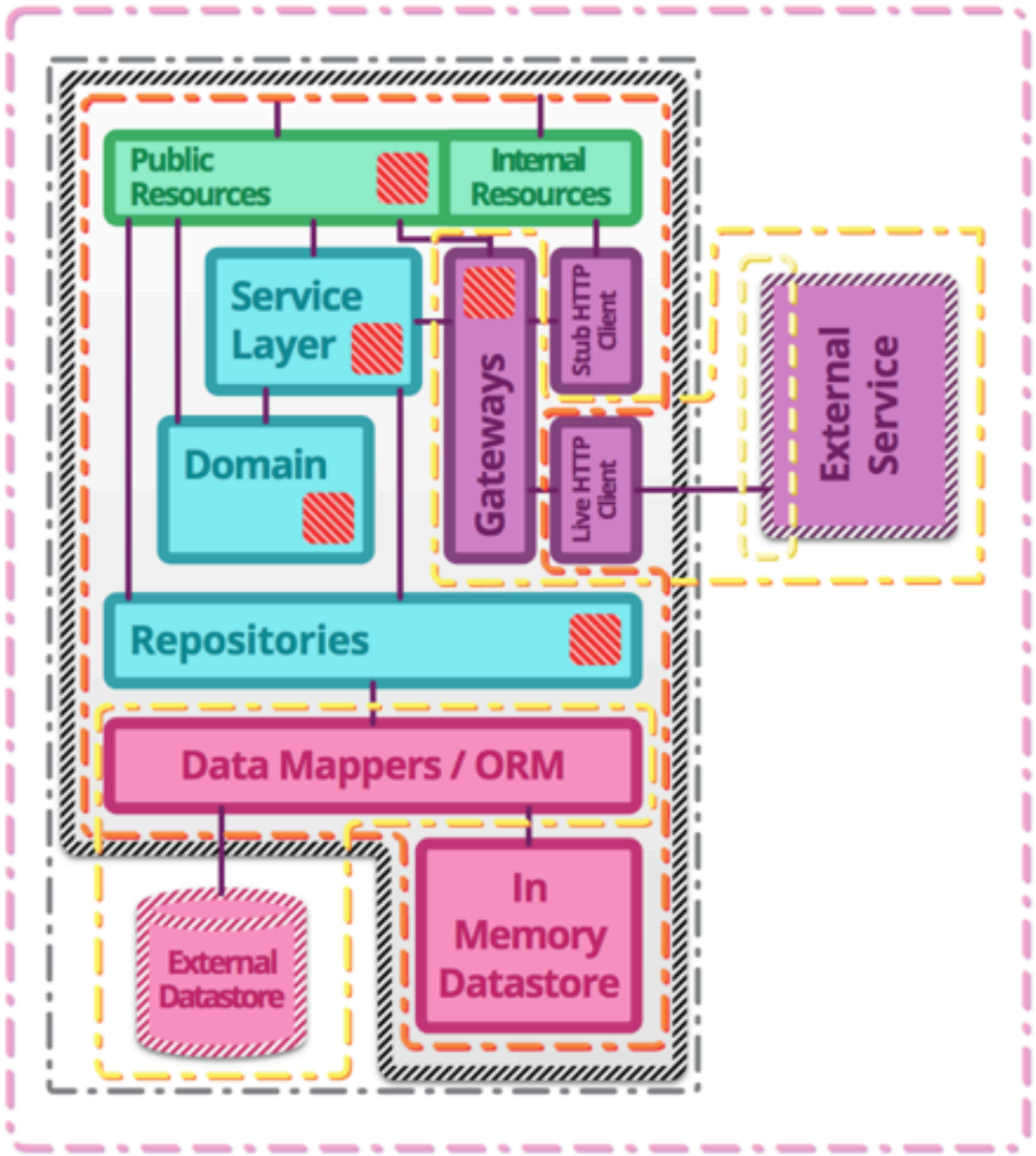
End-to-End testing



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.

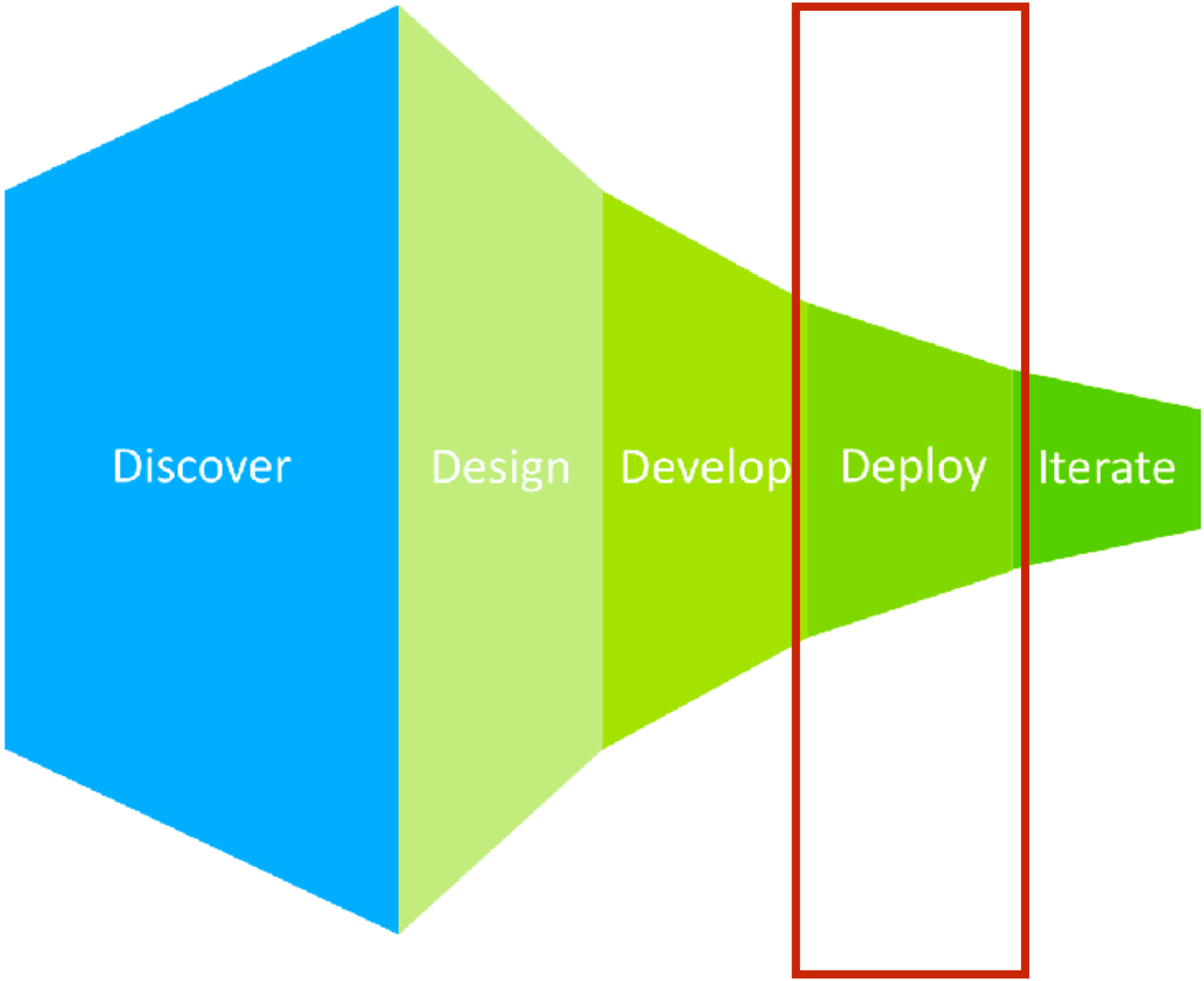


Component tests : limit the scope of the exercised software to a portion of the system under test, manipulating the system through internal code interfaces and using test doubles to isolate the code under test from other components.

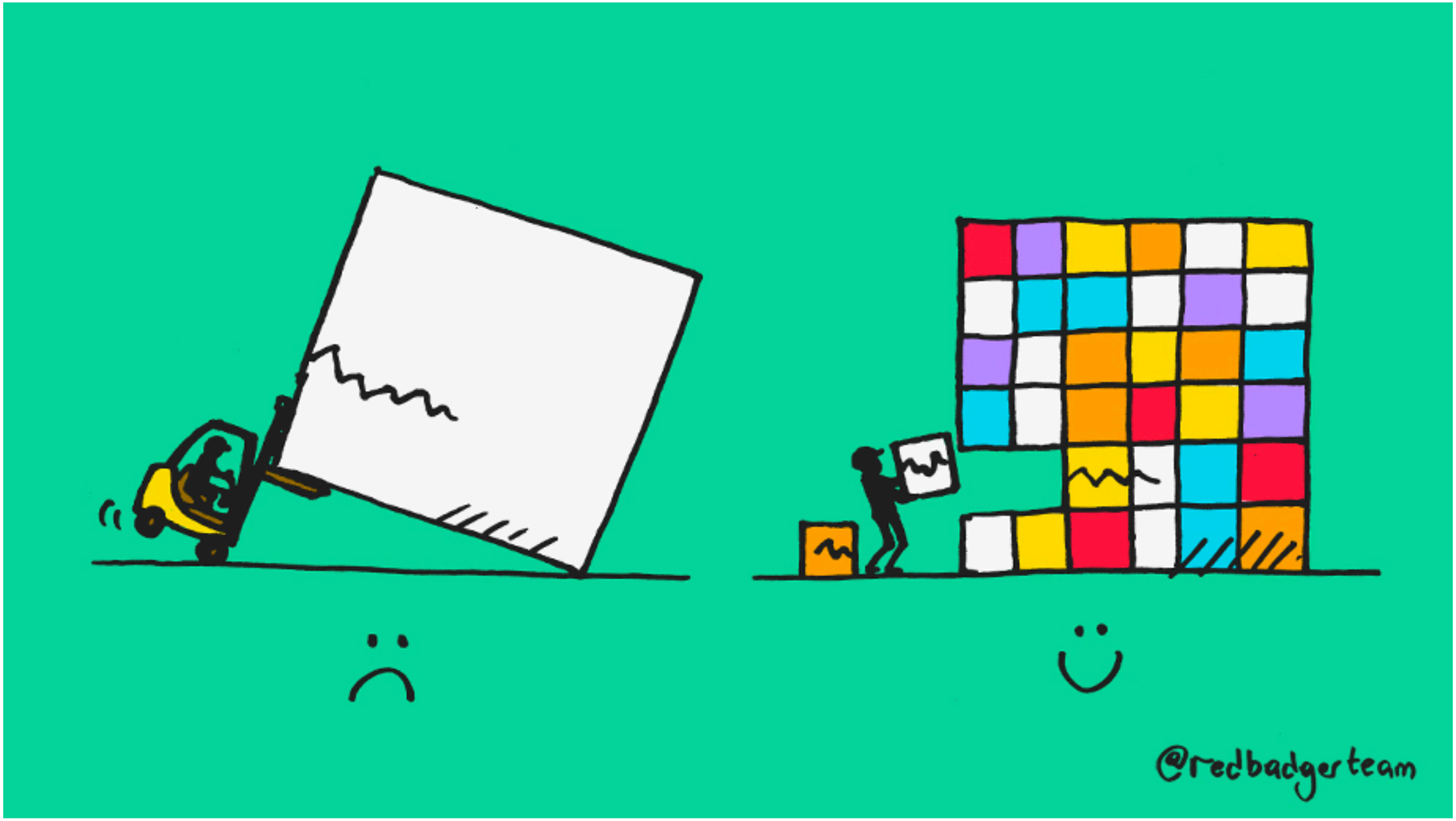
Contract tests : verify interactions at the boundary of an external service asserting that it meets the contract expected by a consuming service.

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 ?



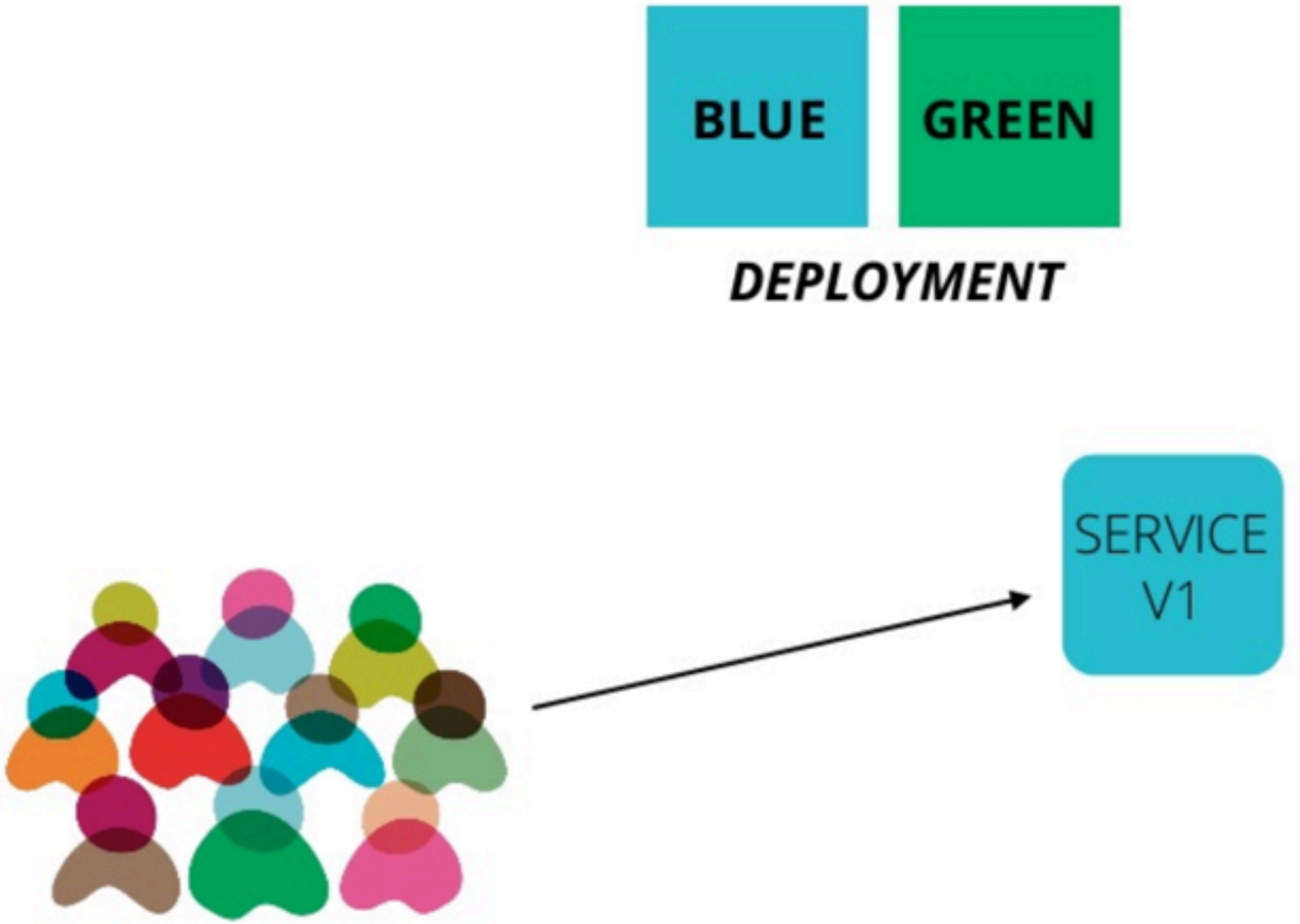
Deployment



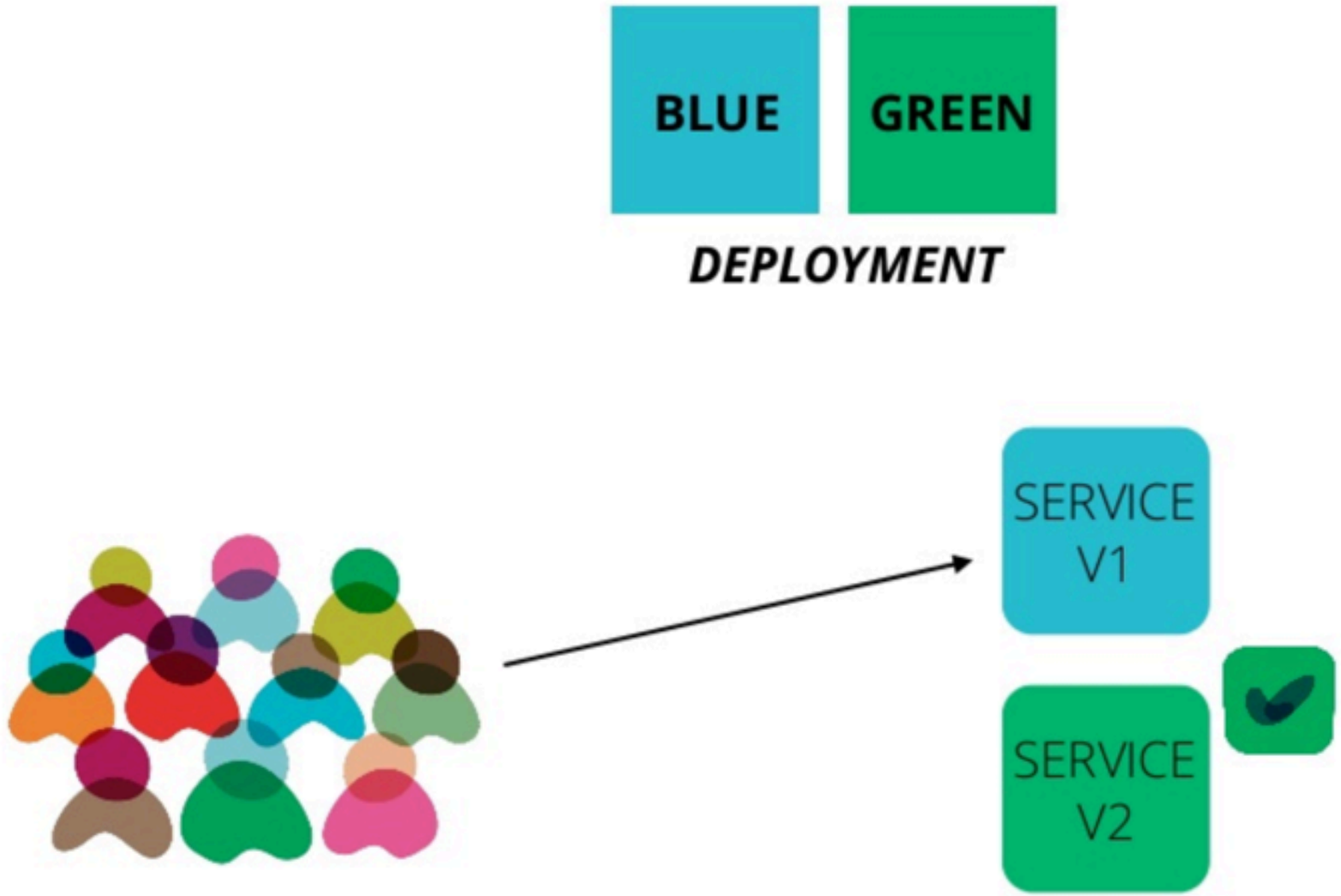
Deploy vs Release

Blue Green Deployment

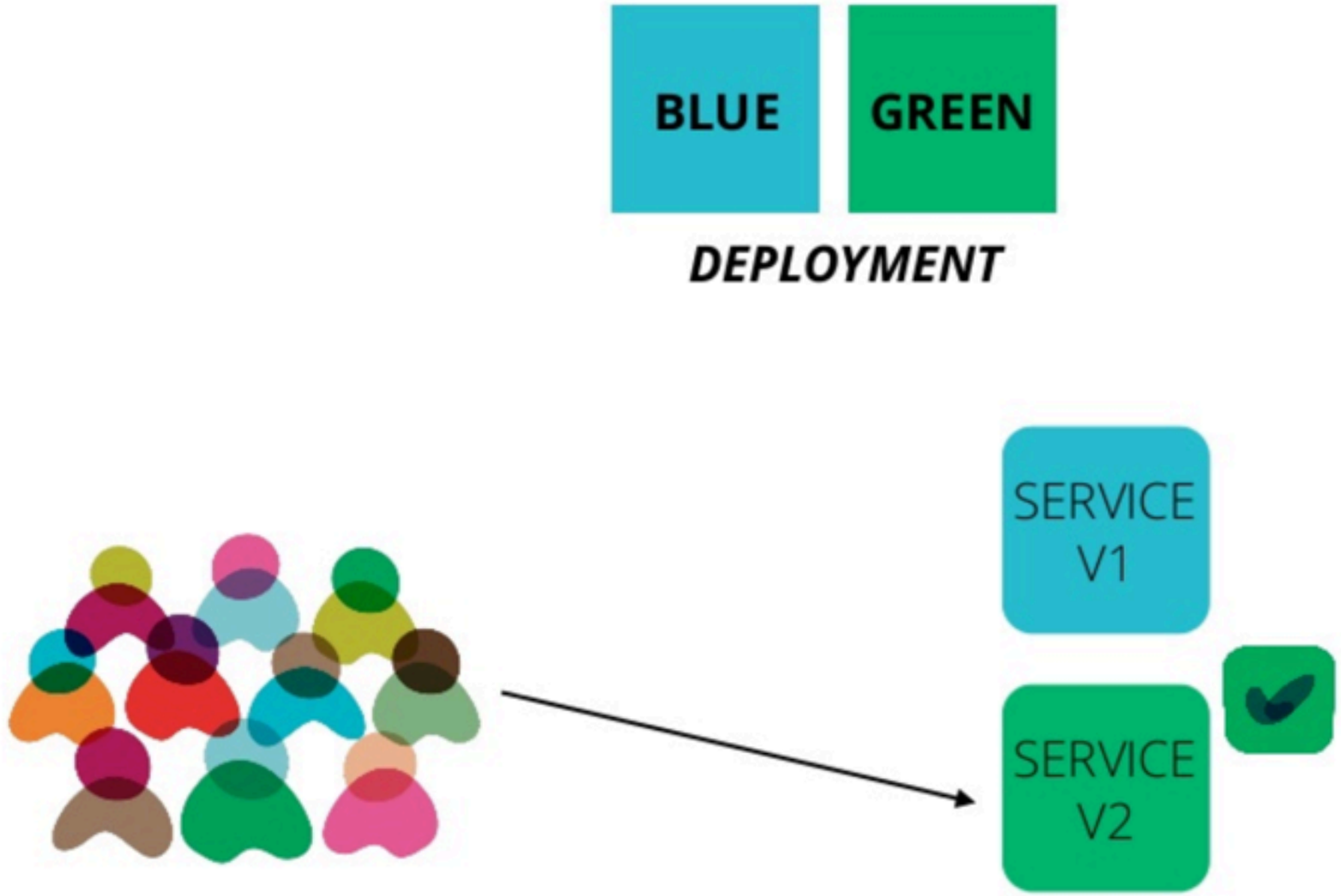
Blue Green Deployment



Blue Green Deployment



Blue Green Deployment

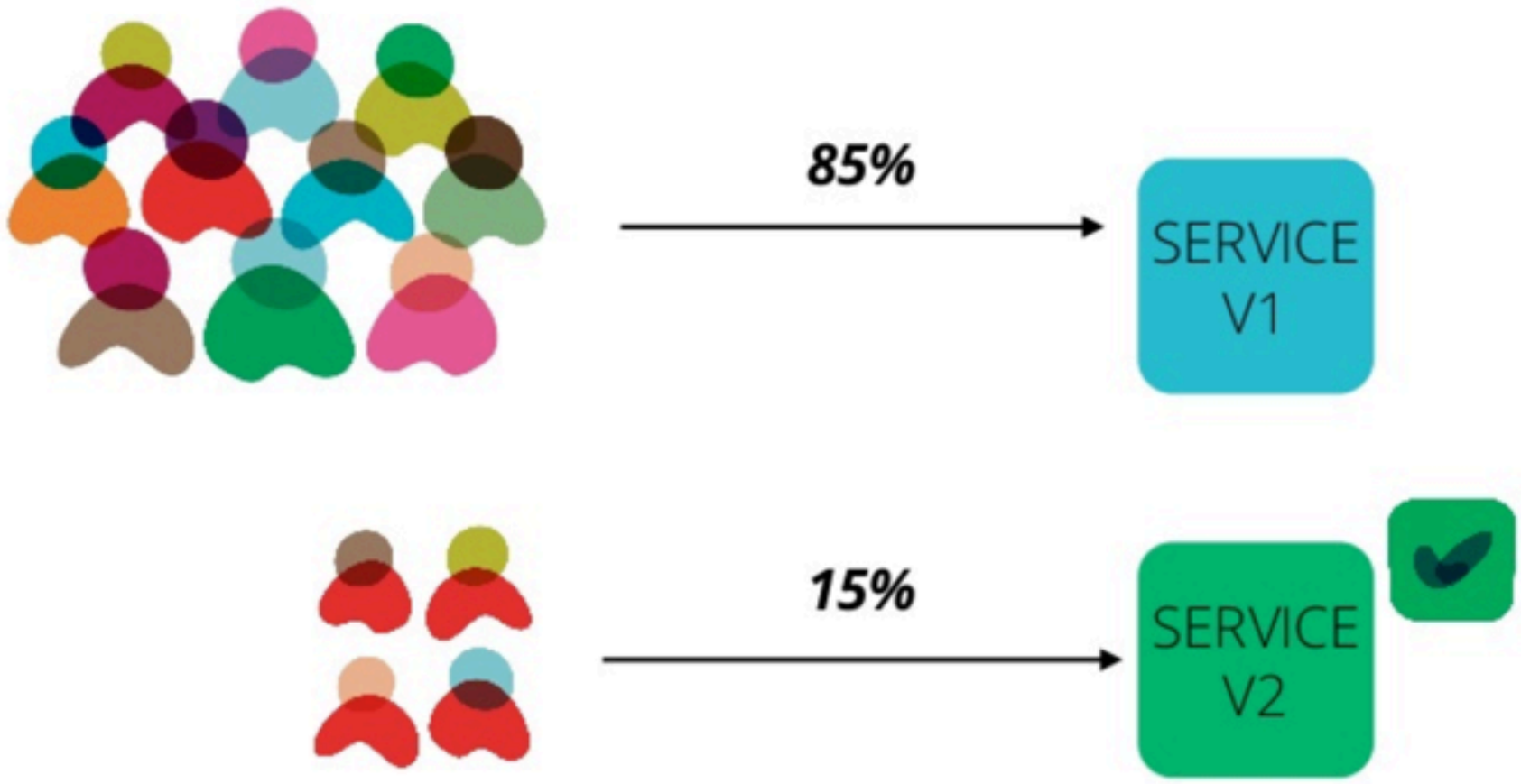


Canary Release

Canary Release

Canary Release

Canary Release



Mean Time to Recover (MTTR)

Mean Time to Recover (MTTR)

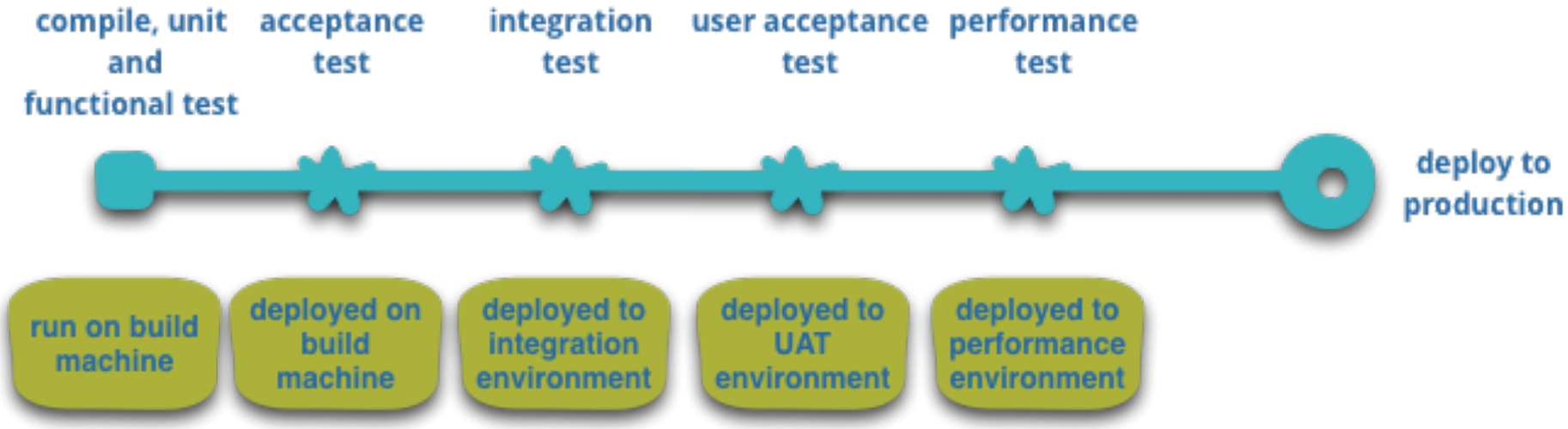
Tests are very important to reduce amount of defects in your systems. However, it's important to acknowledge that bugs will always happen in production.

Mean Time to Recover (MTTR)

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

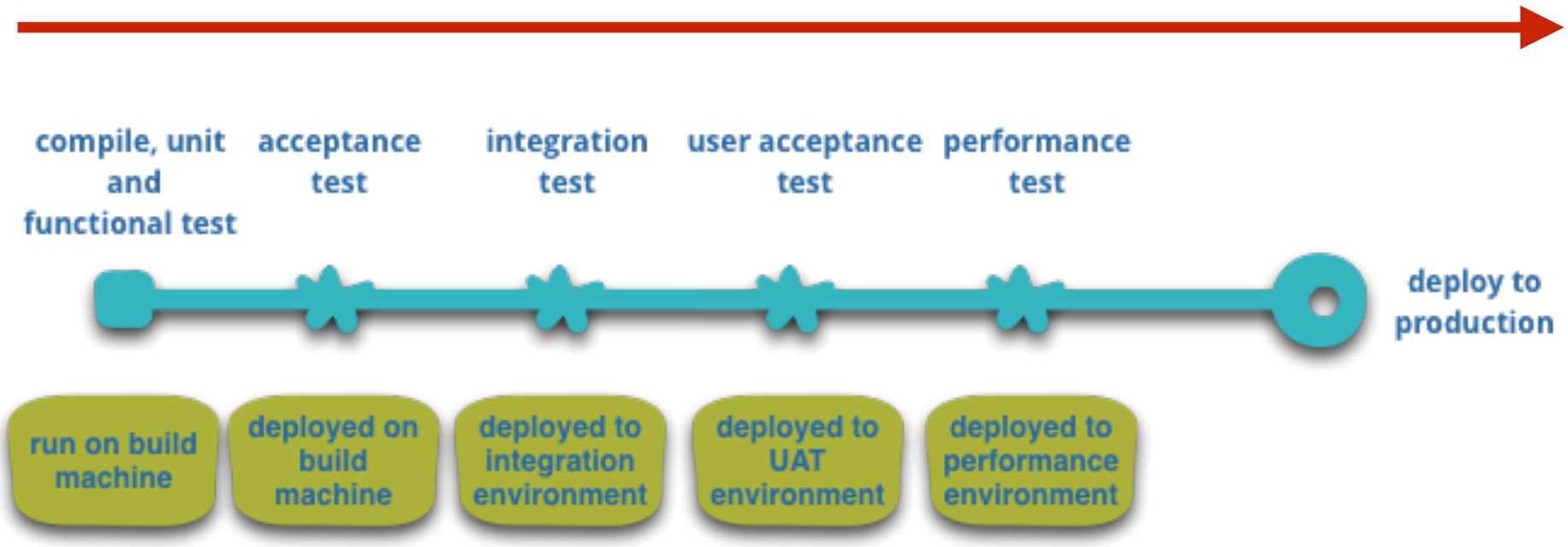
Current situation !!

Infrastructure Automation

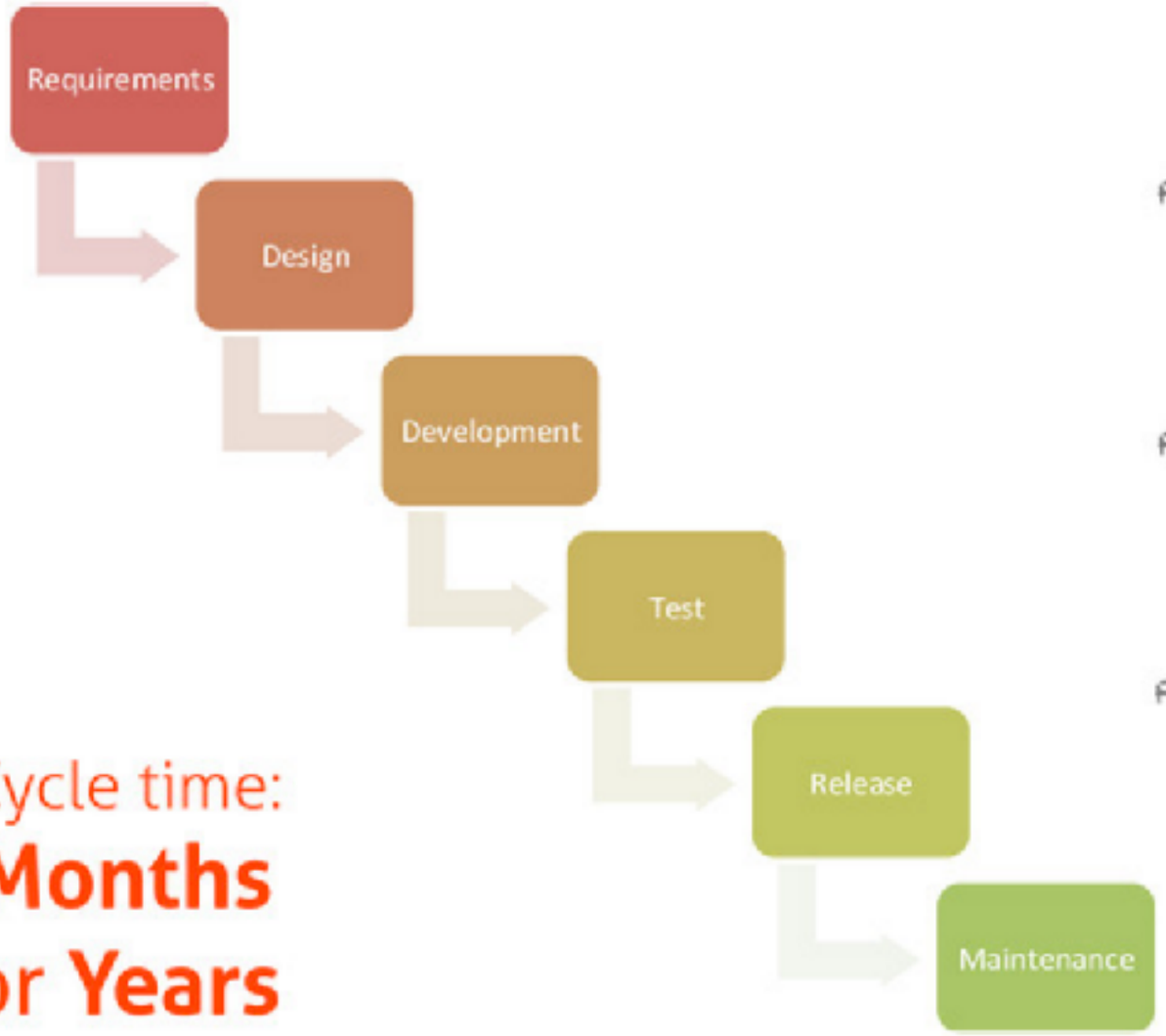


Infrastructure Automation

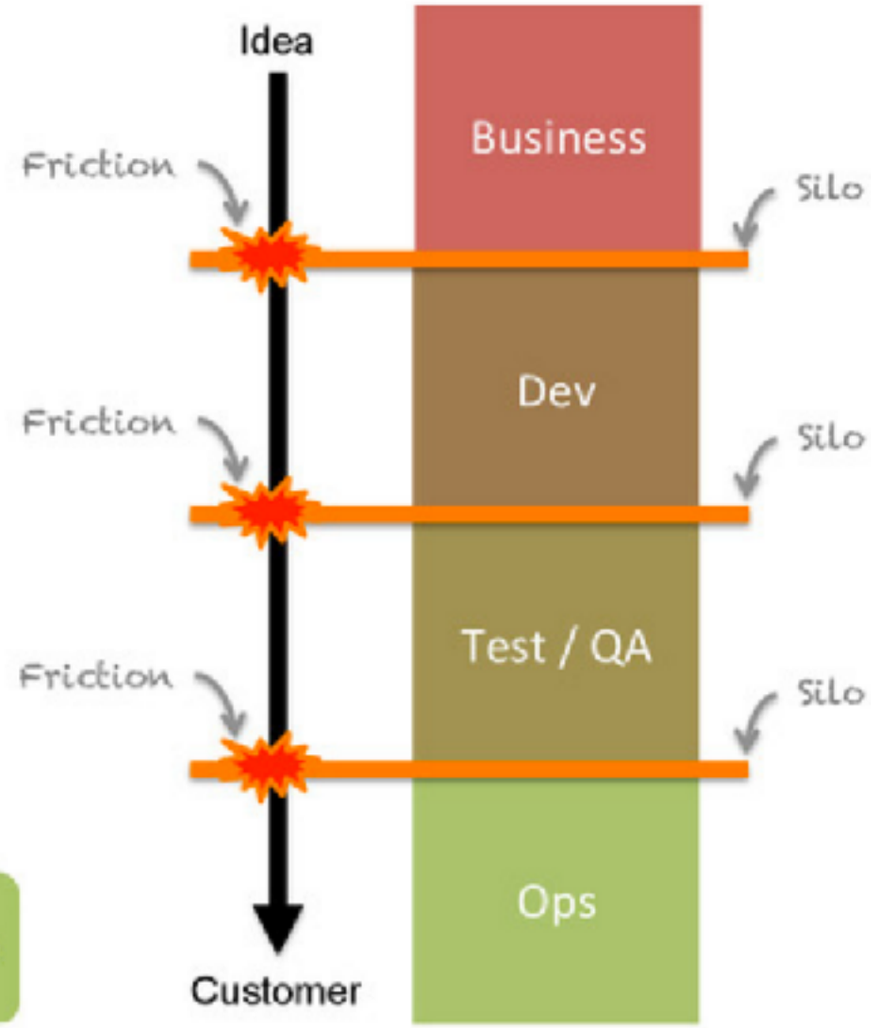
Lead time ?



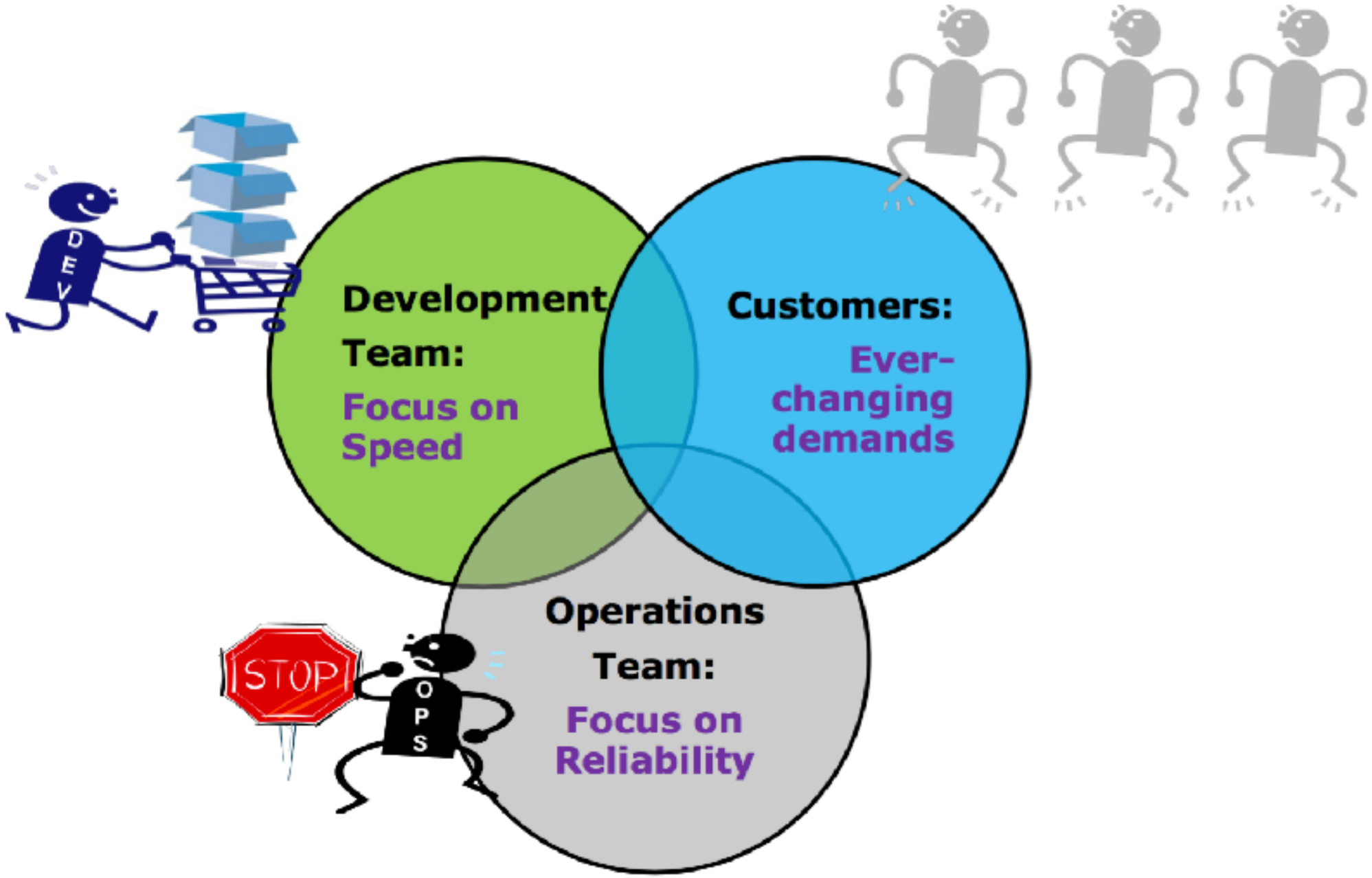
Traditional development



Cycle time:
Months
or **Years**

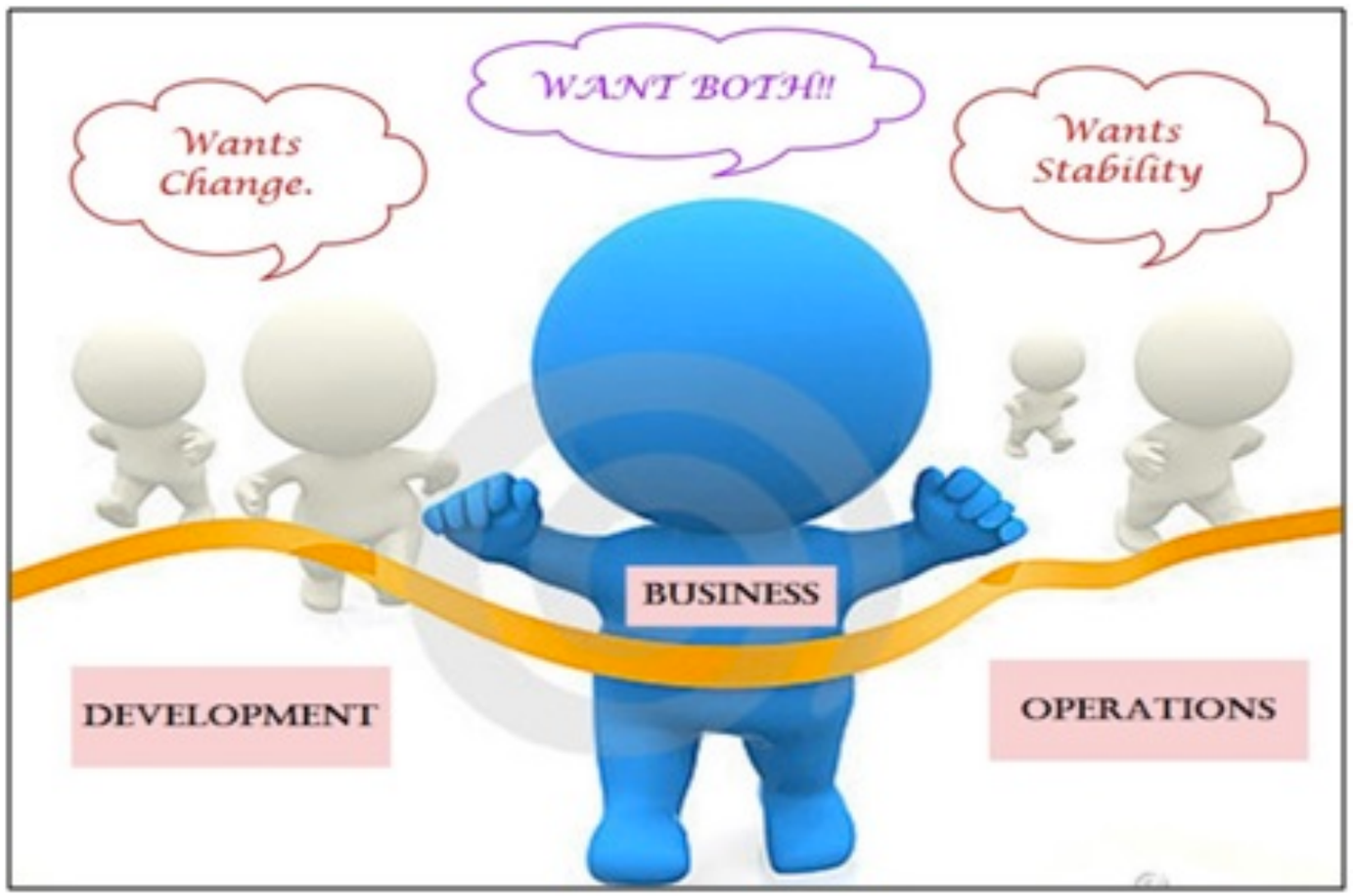


Conflict of Interest

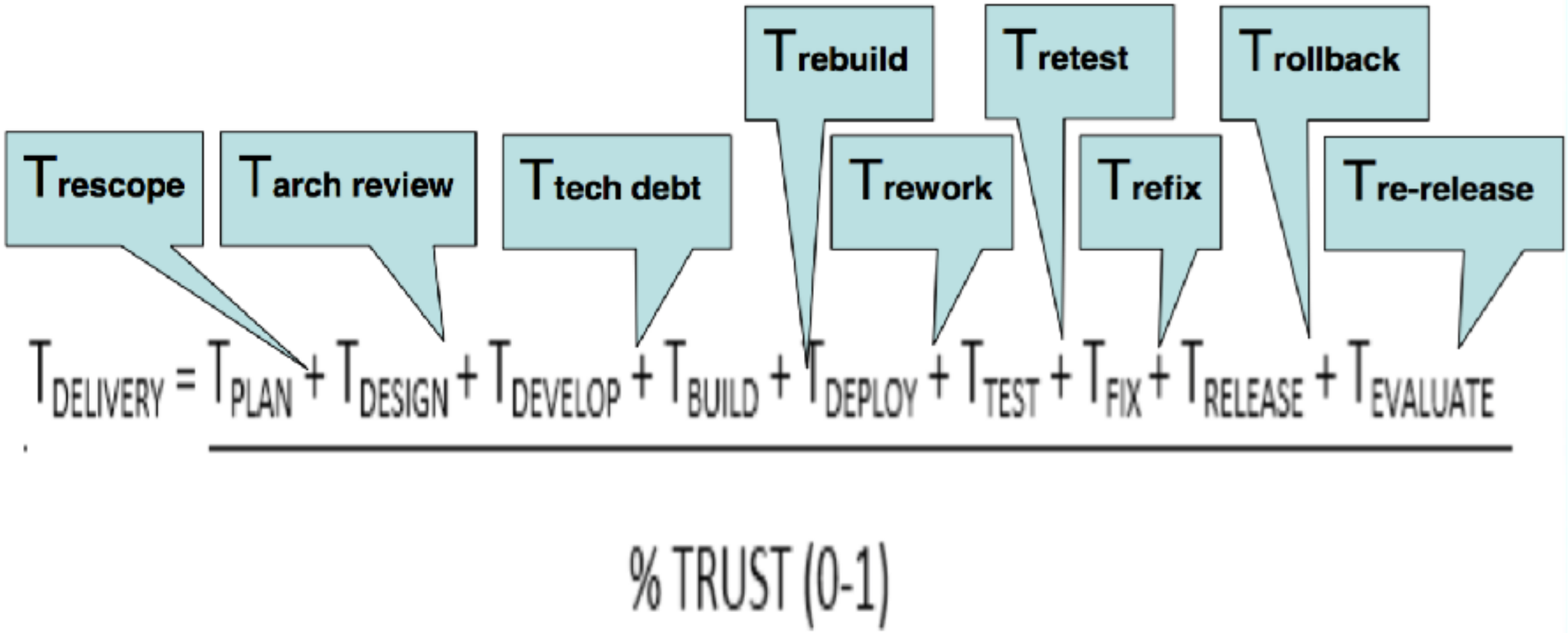


Conflict of Interest

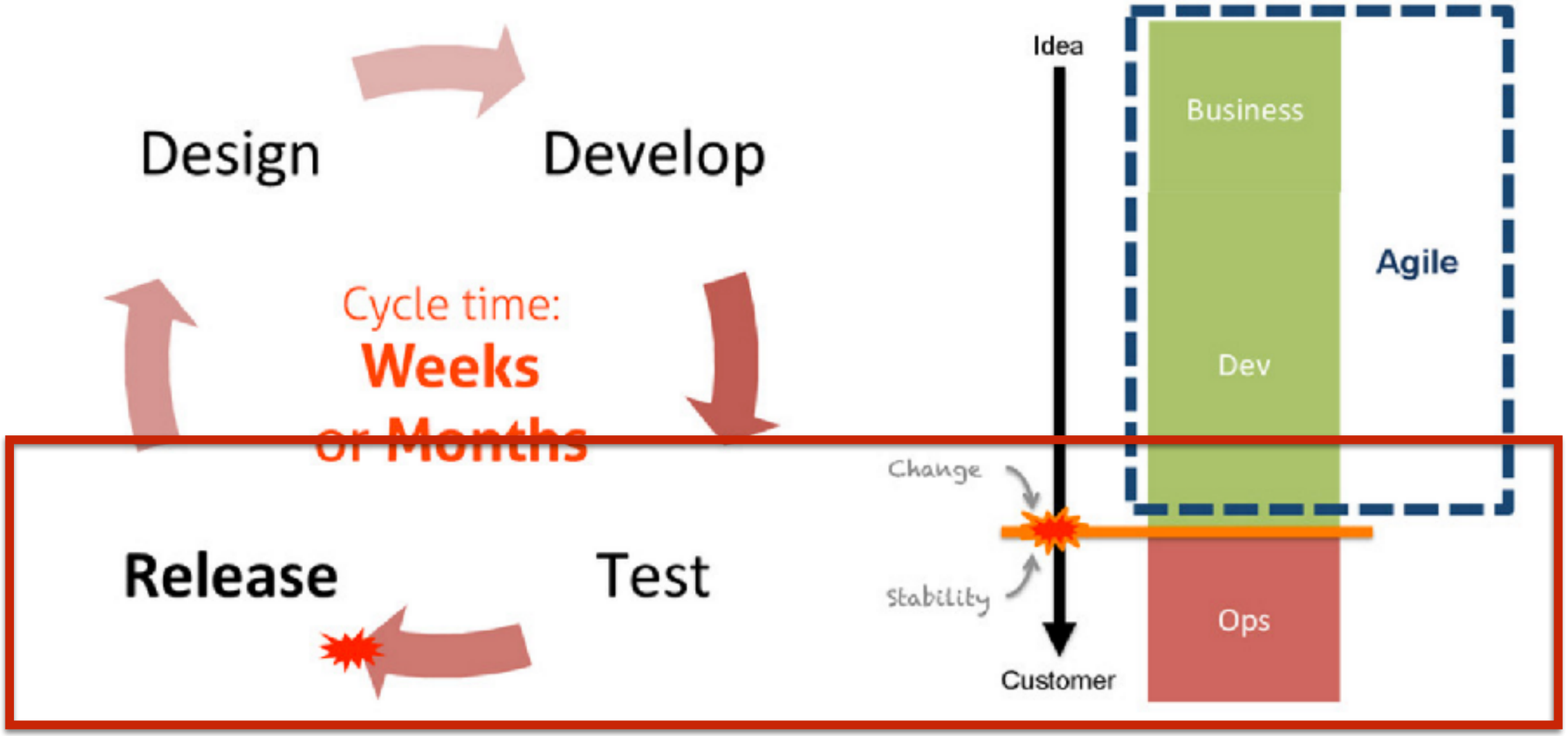
Conflict of Interest



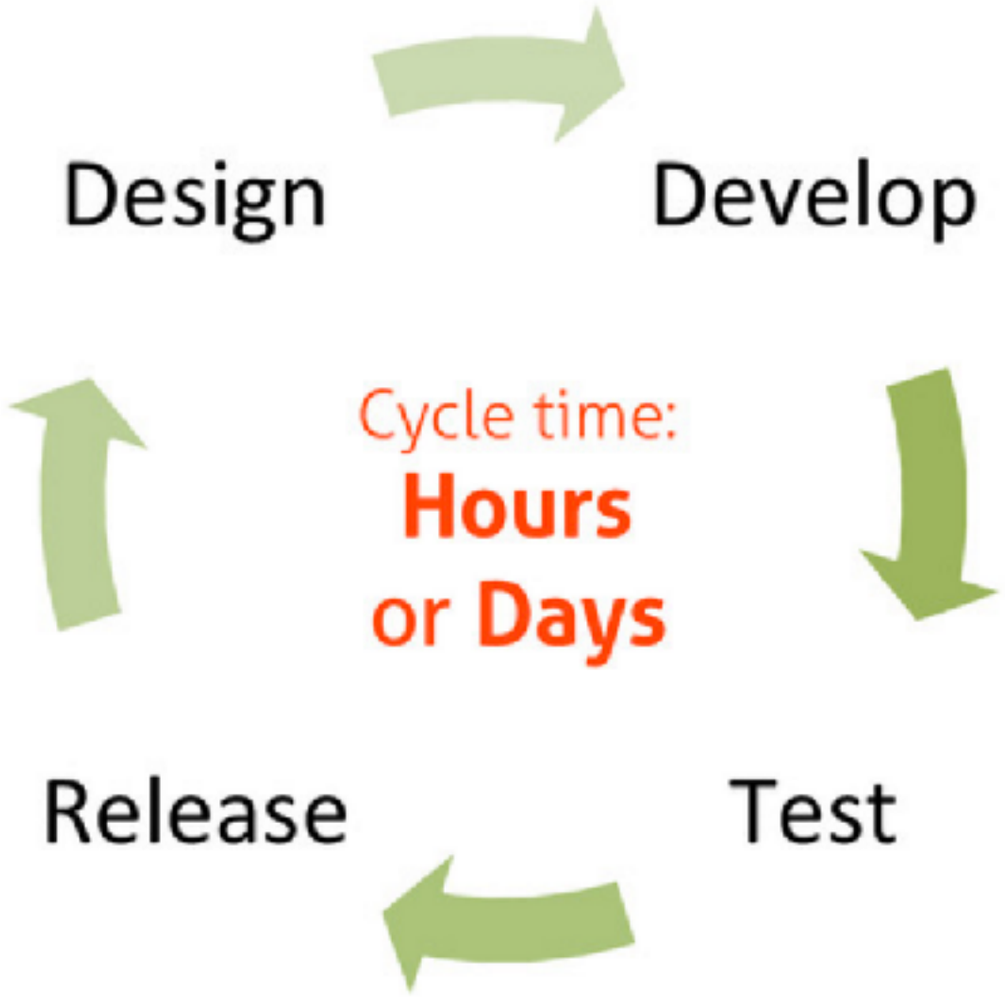
Low trust create extra steps

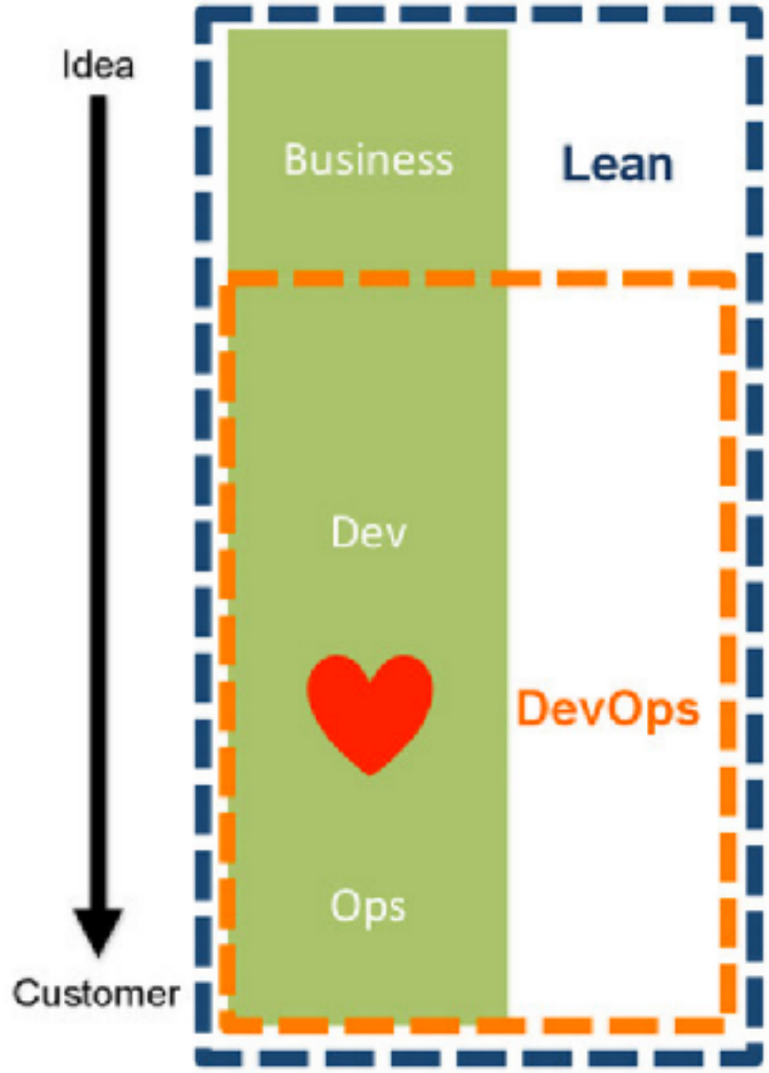


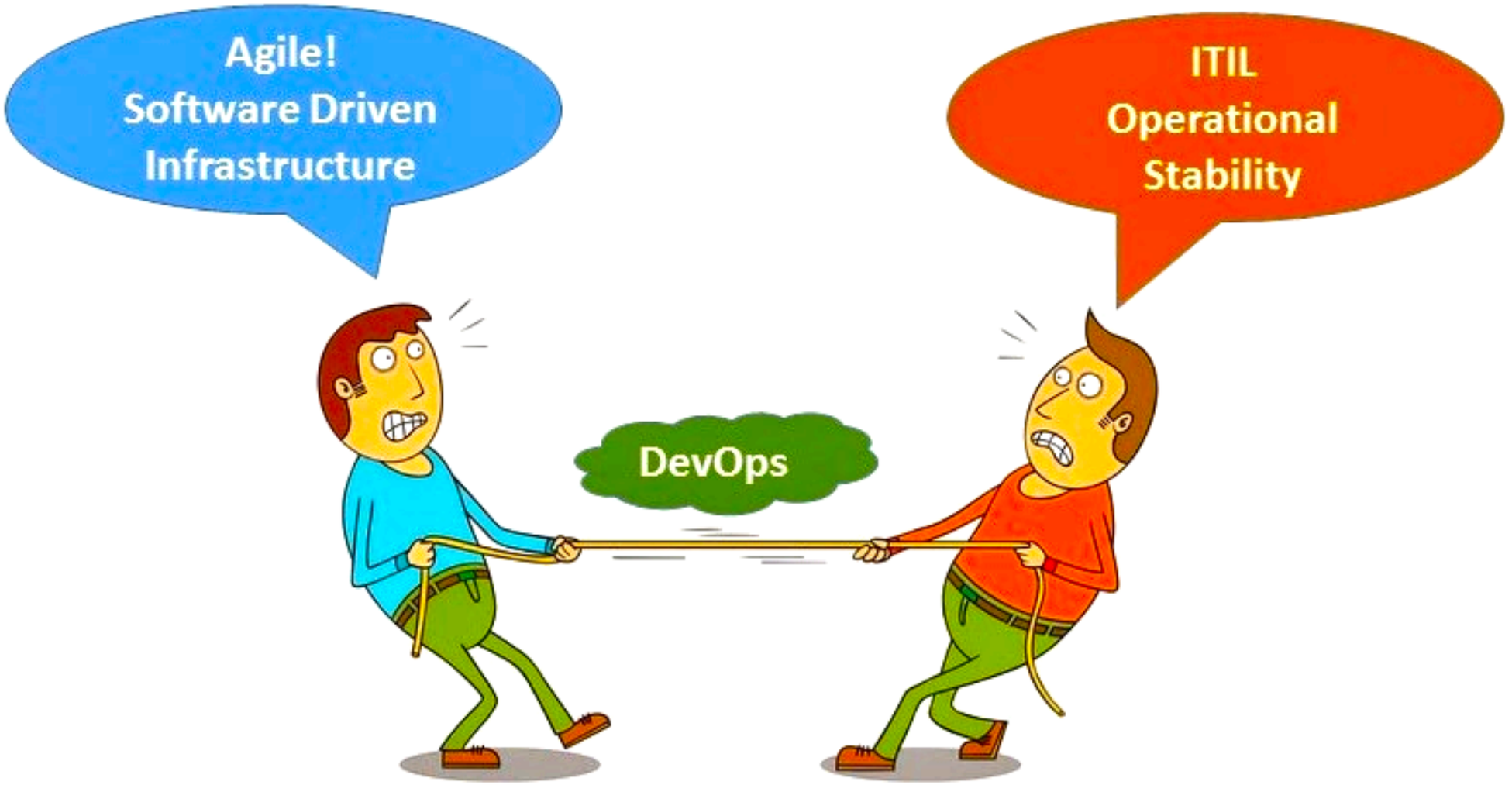
Iterative/Agile development

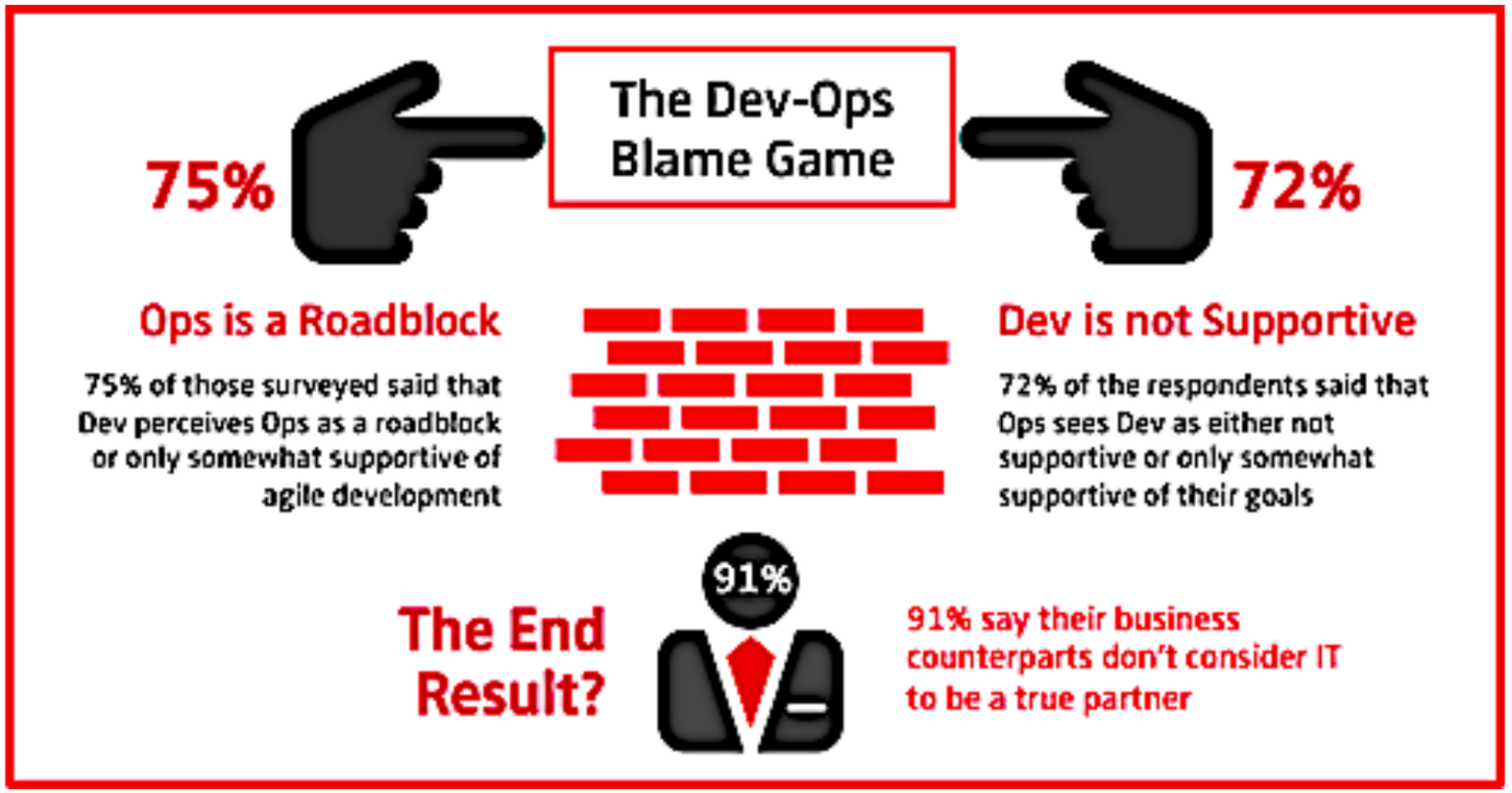


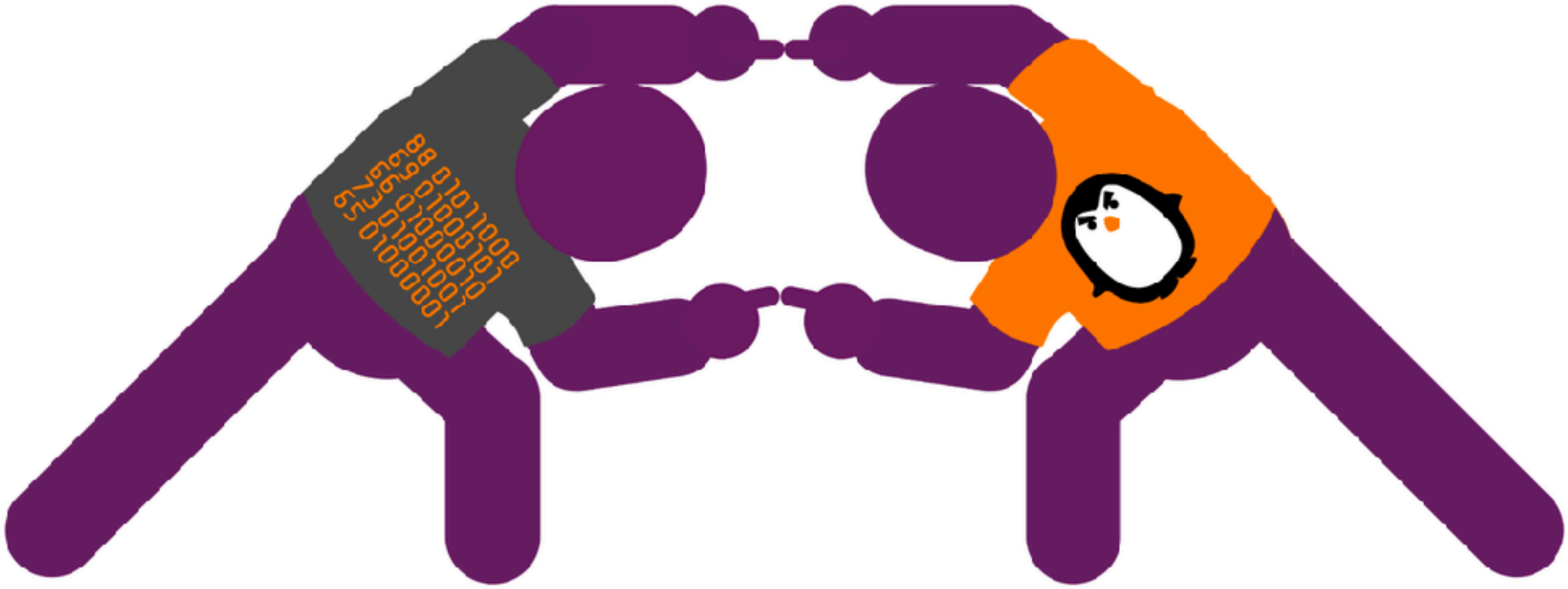
Rise of DevOps

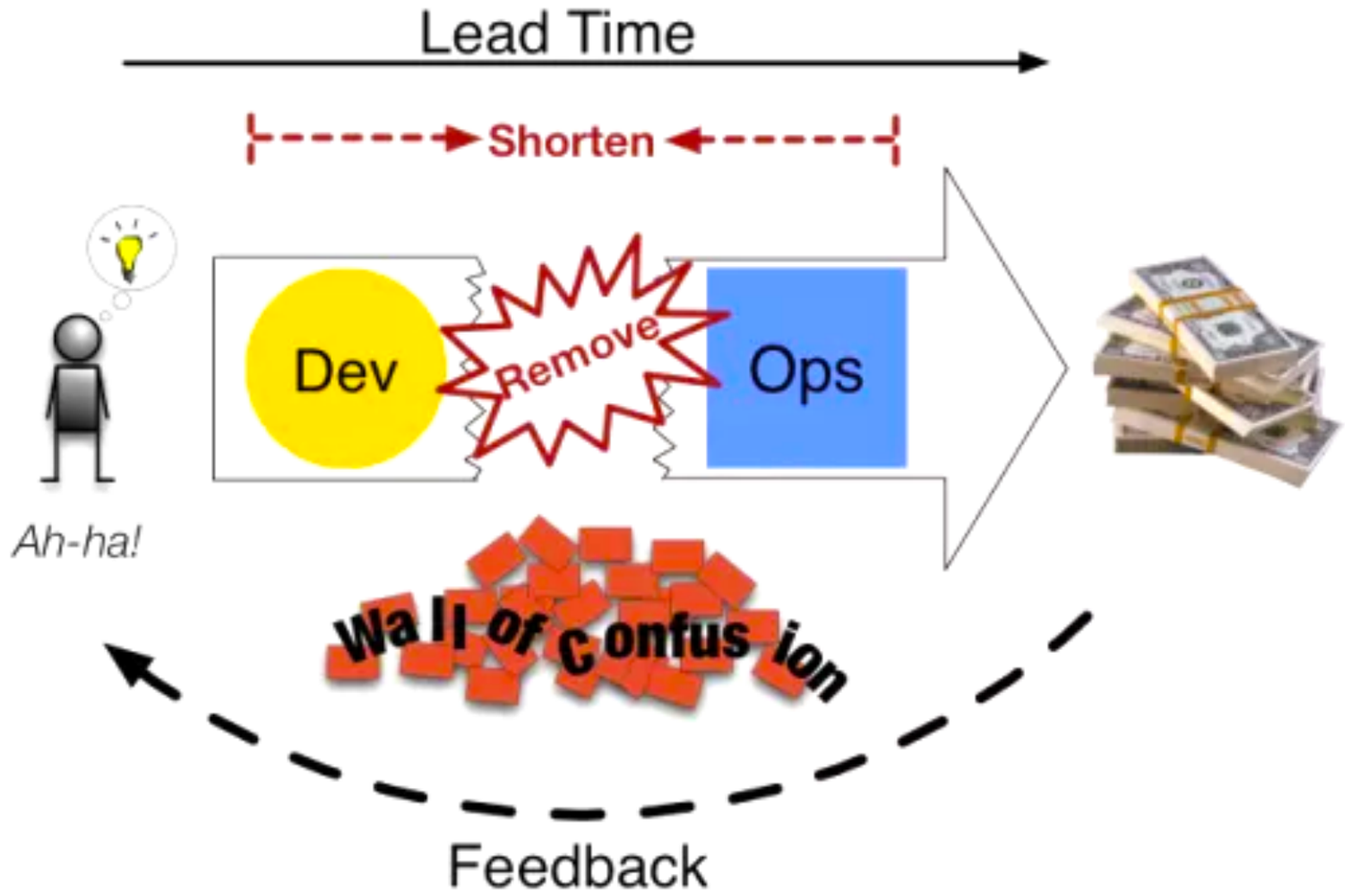


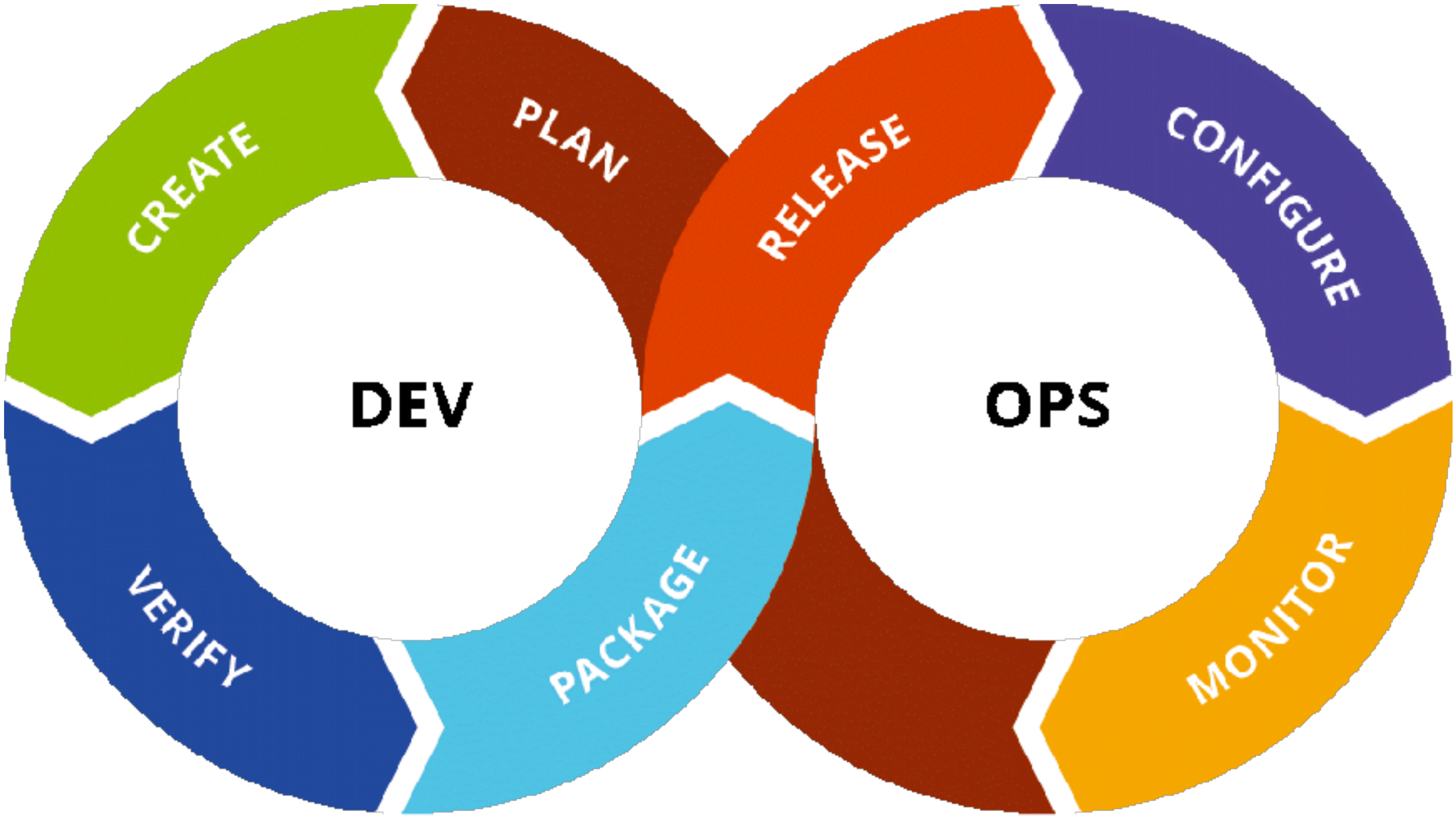












DEV + OPS

Application Performance

Decrease latency by using APM Tools.

End User Analytics

Monitor end user latency and check device performance

Quality Code

Ensure deployments don't degrade performance

Code-Level Errors

Lower MTTR by finding error root causes

Application Availability

Make sure Uptime and SLAs are in order

Application Performance

Solve problems by correlating infrastructure and application metrics

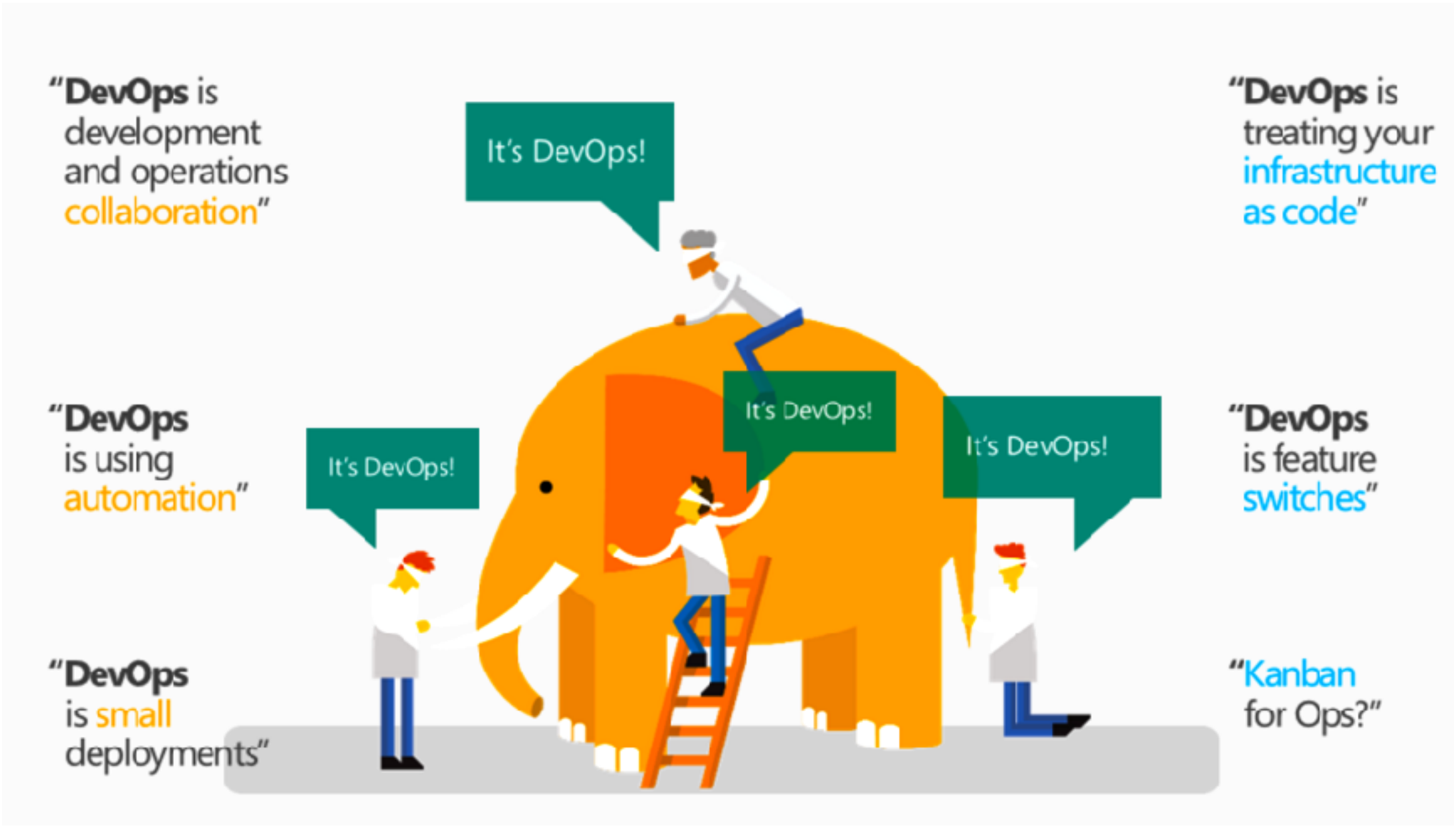
End User Complaints

Fix problems before end users complain

Performance Analytics

Use automatically generated baselines to focus troubleshooting

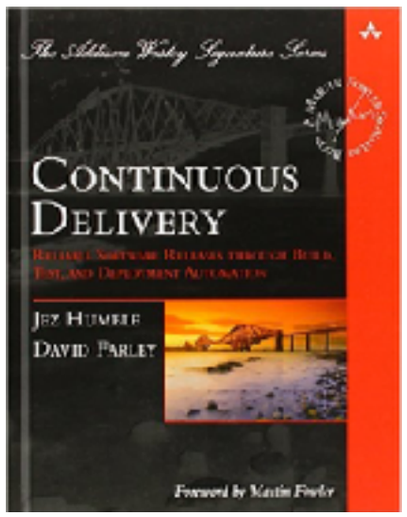
DevOps ?



DevOps ?

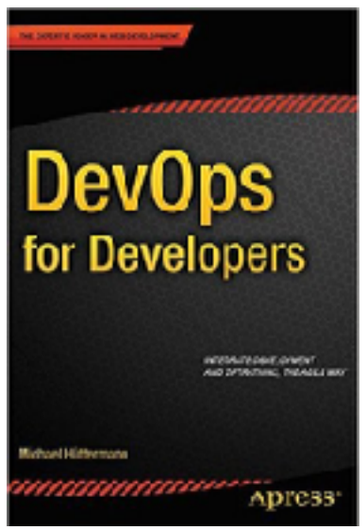
"A **movement of people who care** about **developing** and **operating** reliable, secure, high performance **systems at scale**."

– Jez Humble –



DevOps ?

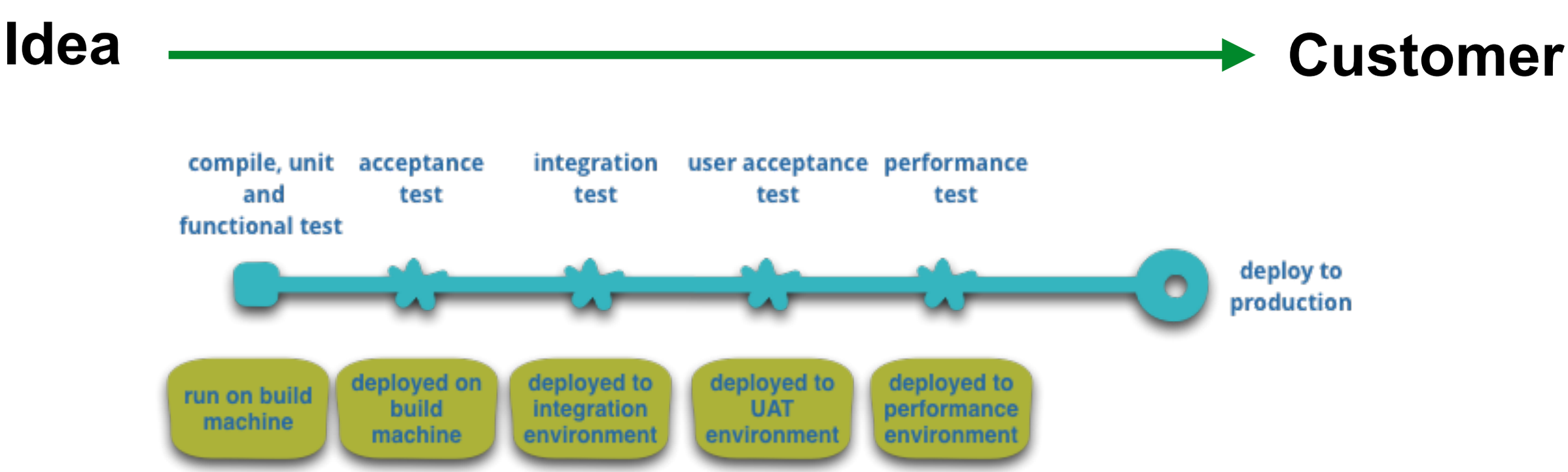
"A **mix of patterns** intended to **improve collaboration** between development and operations. DevOps addresses **shared goals and incentives** as well as **shared processes and tools**."



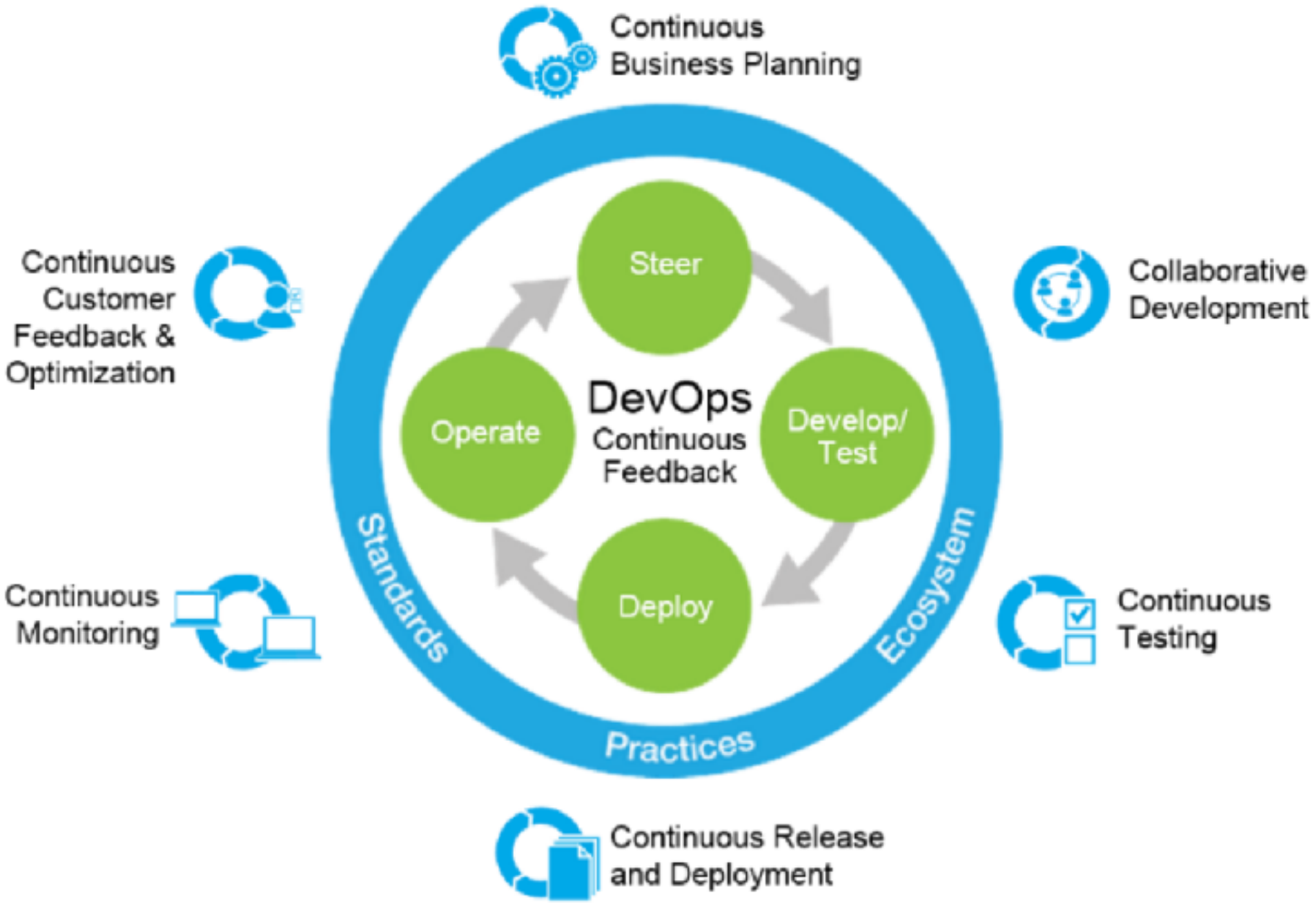
- Michael Huttermann -

Goal of DevOps

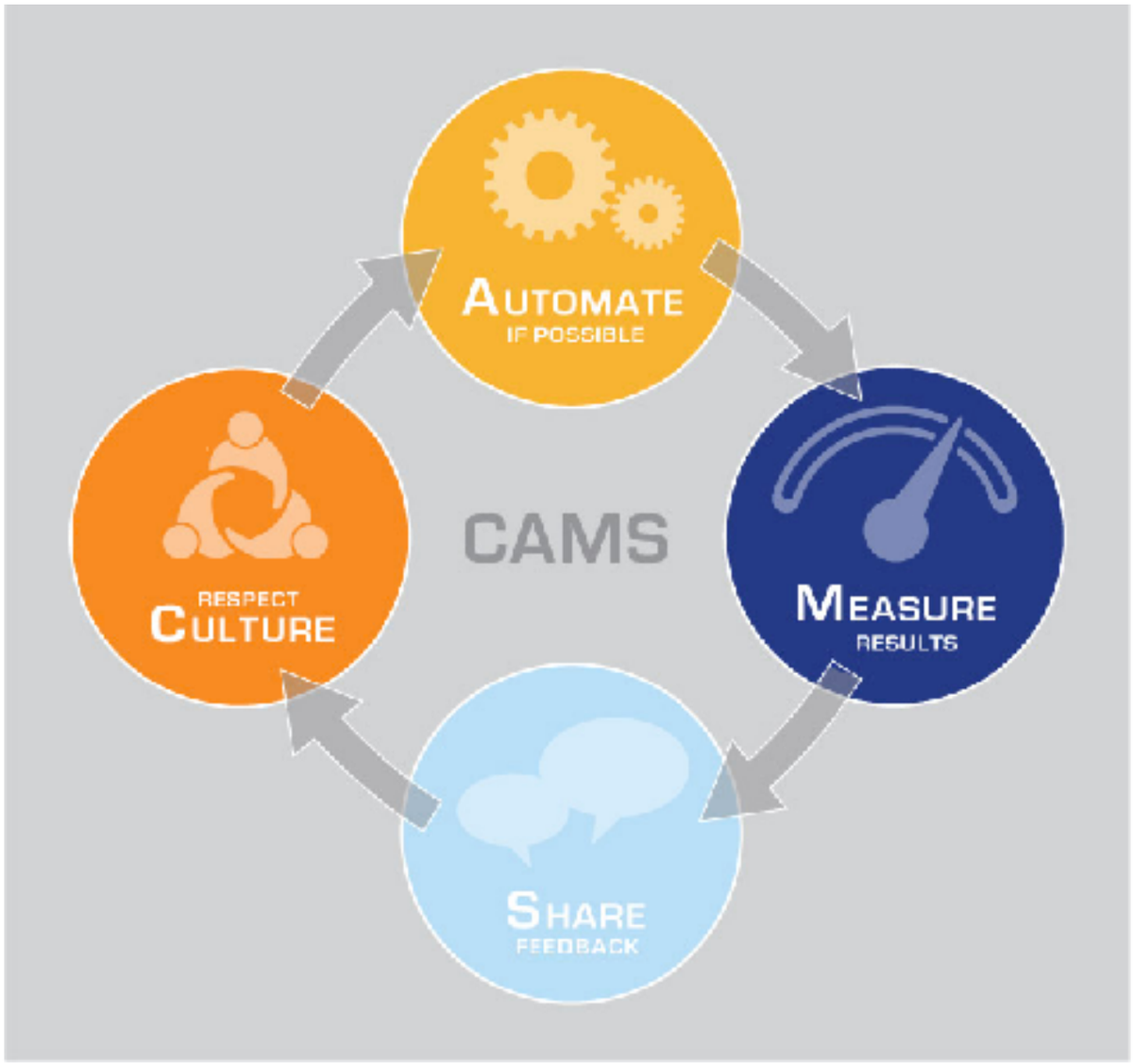
"Improve the delivery of value for Customer and Business"



DevOps Life Cycle



DevOps Principles



DevOps Principles

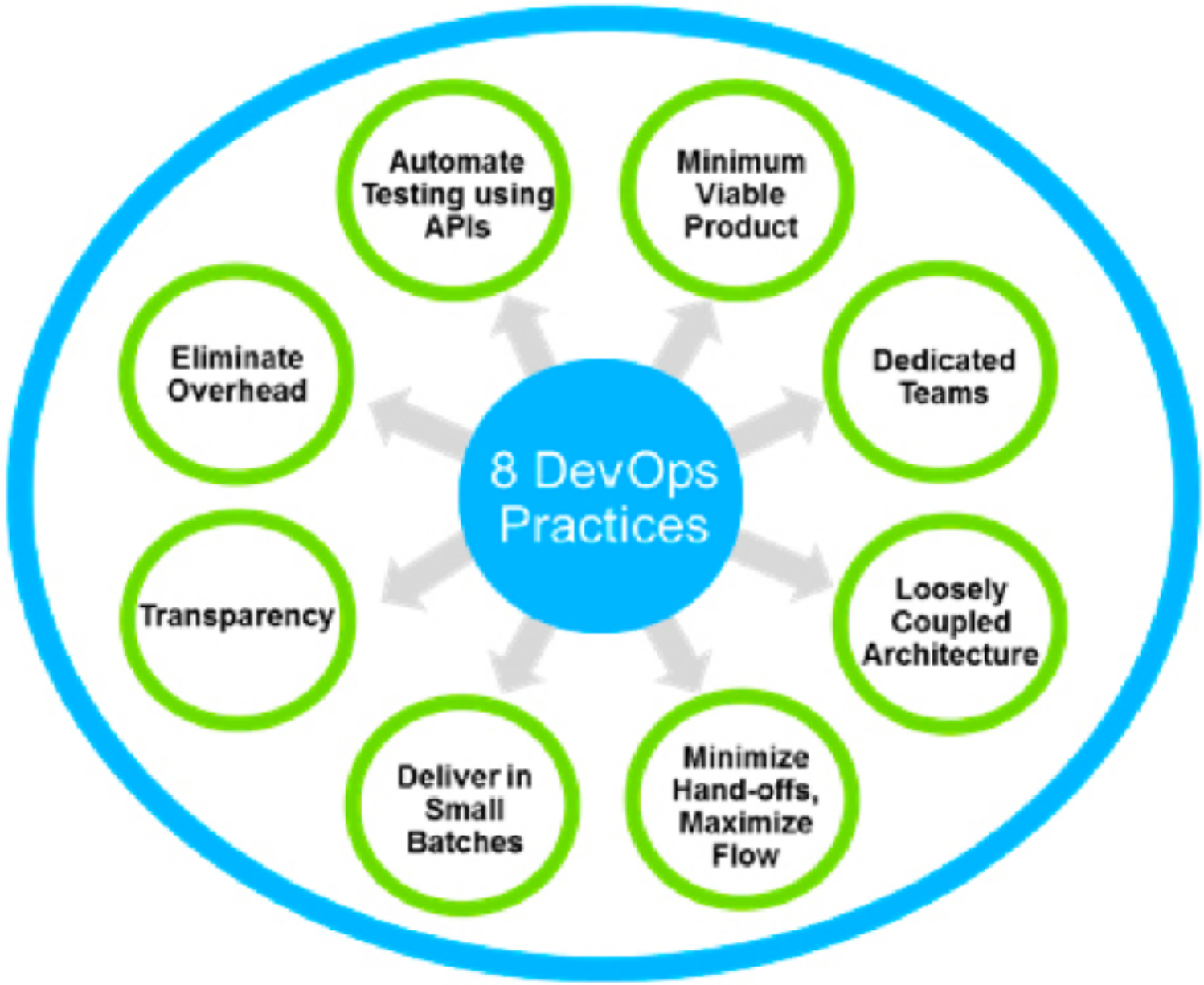
Culture => People, Process, Tools

Automation => Infrastructure as Code

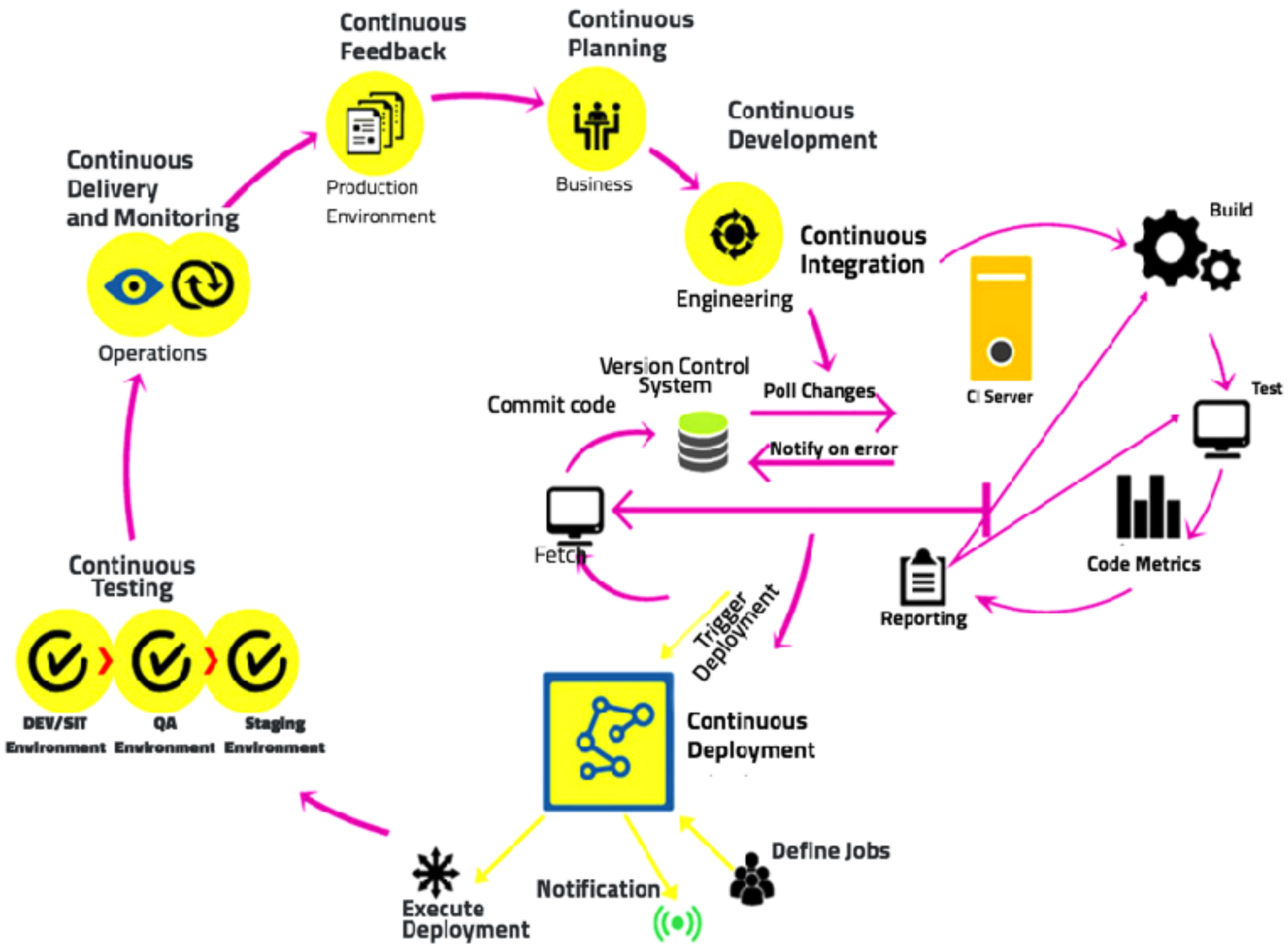
Measurement => Measure everything

Sharing => Collaboration/Feedback

DevOps Practices



DevOps Practices

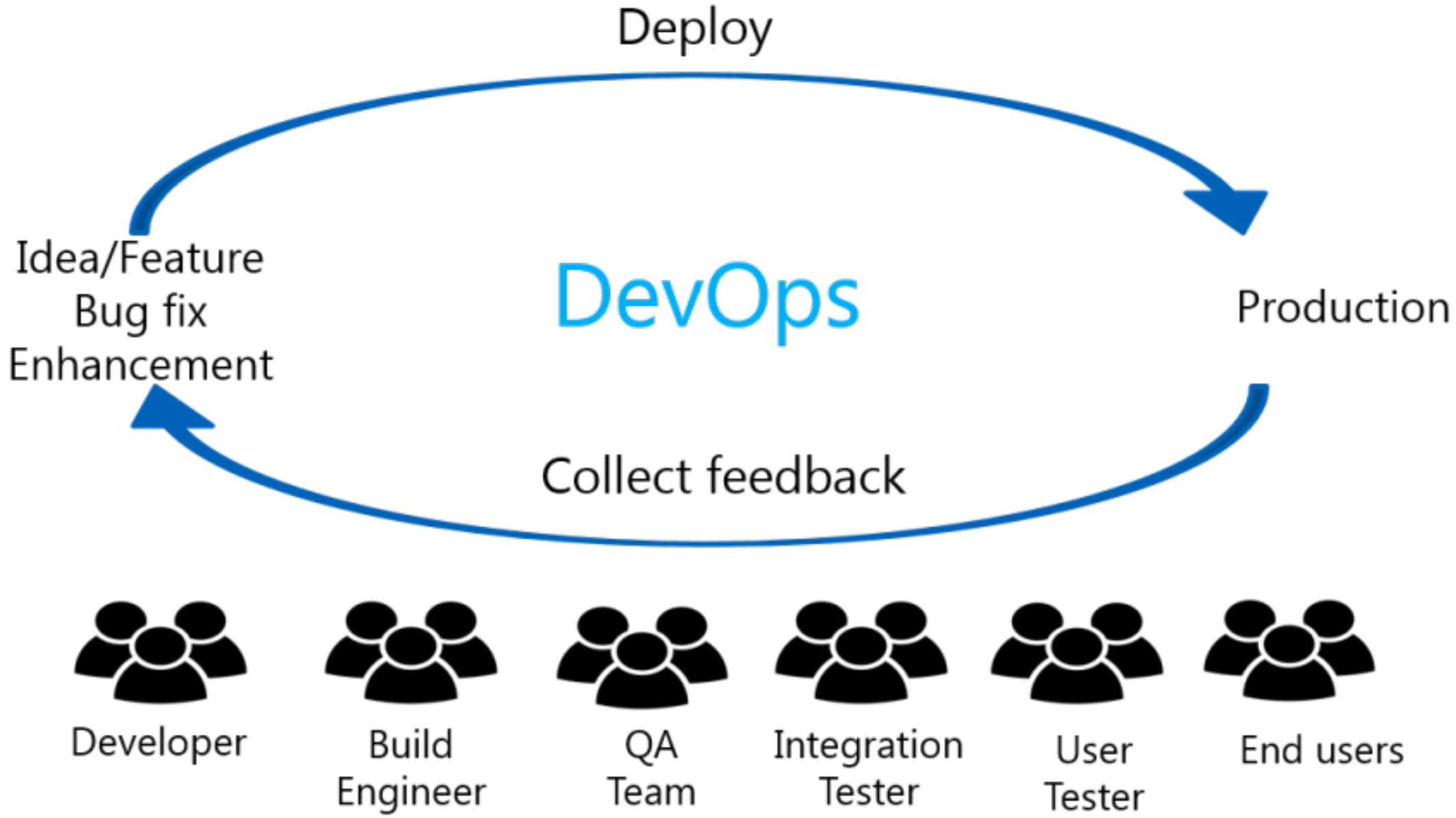


DevOps
3 ways principle

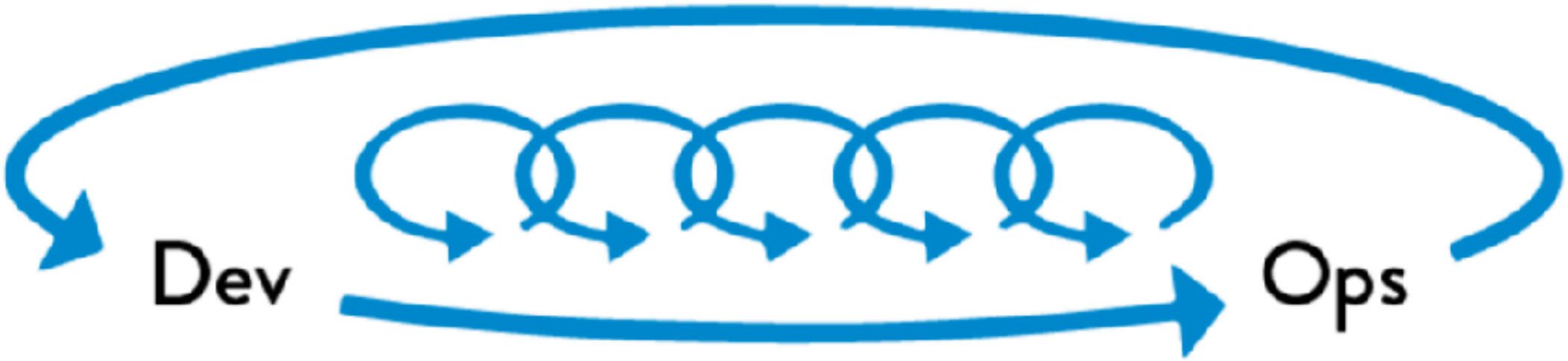
Flow principle

Feedback principle

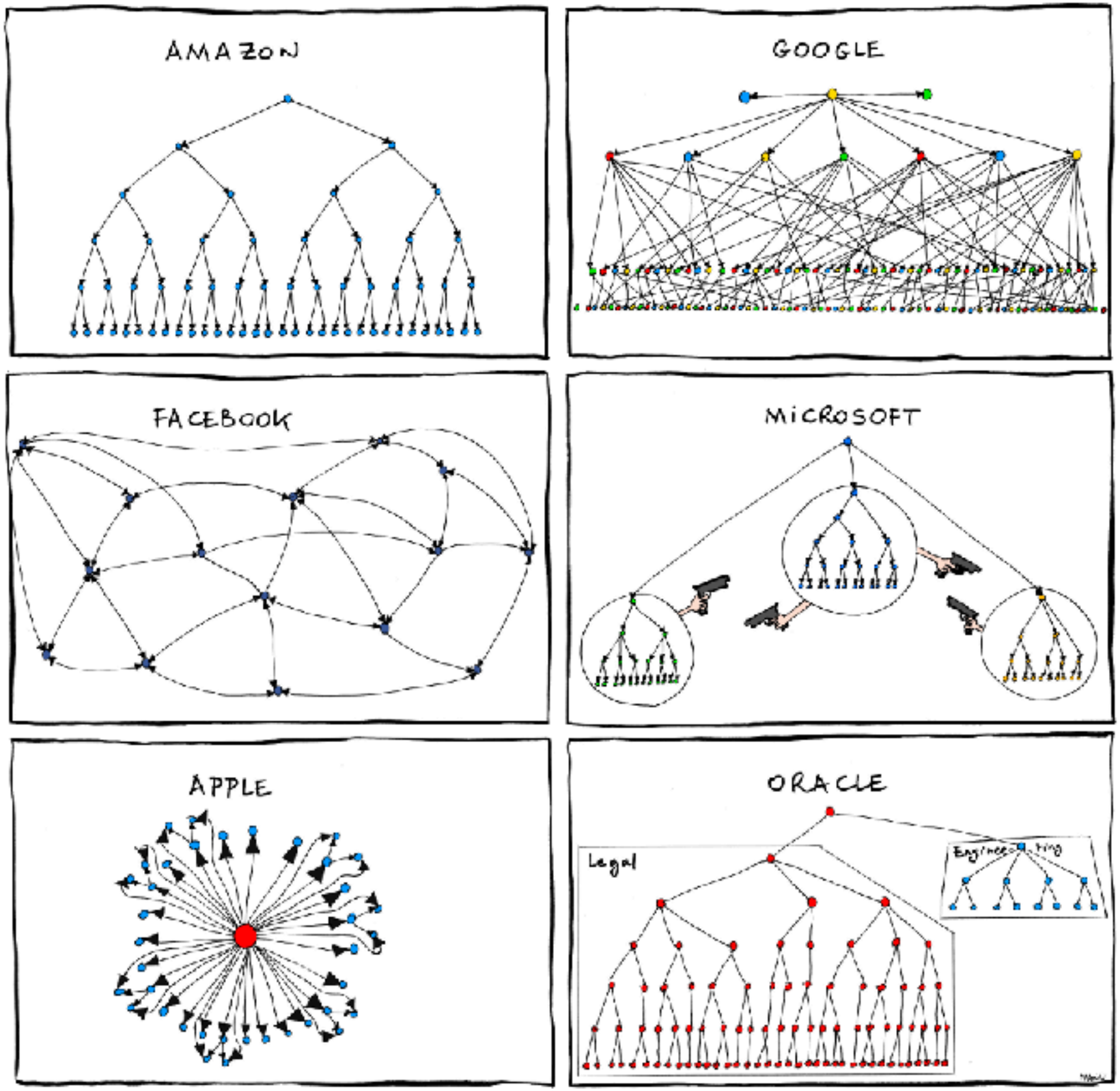
Feedback principle



Continuous learning principle

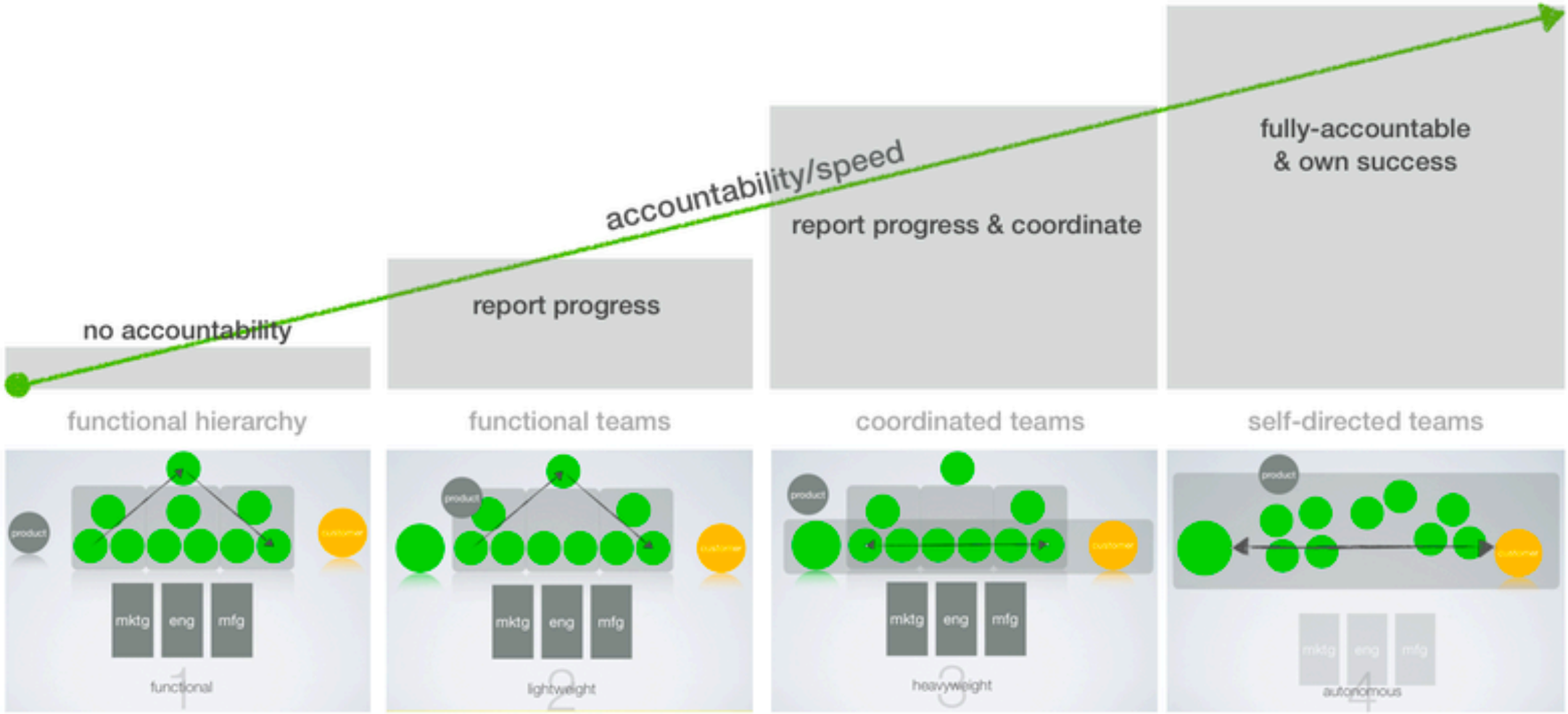


All about Organization structure and culture



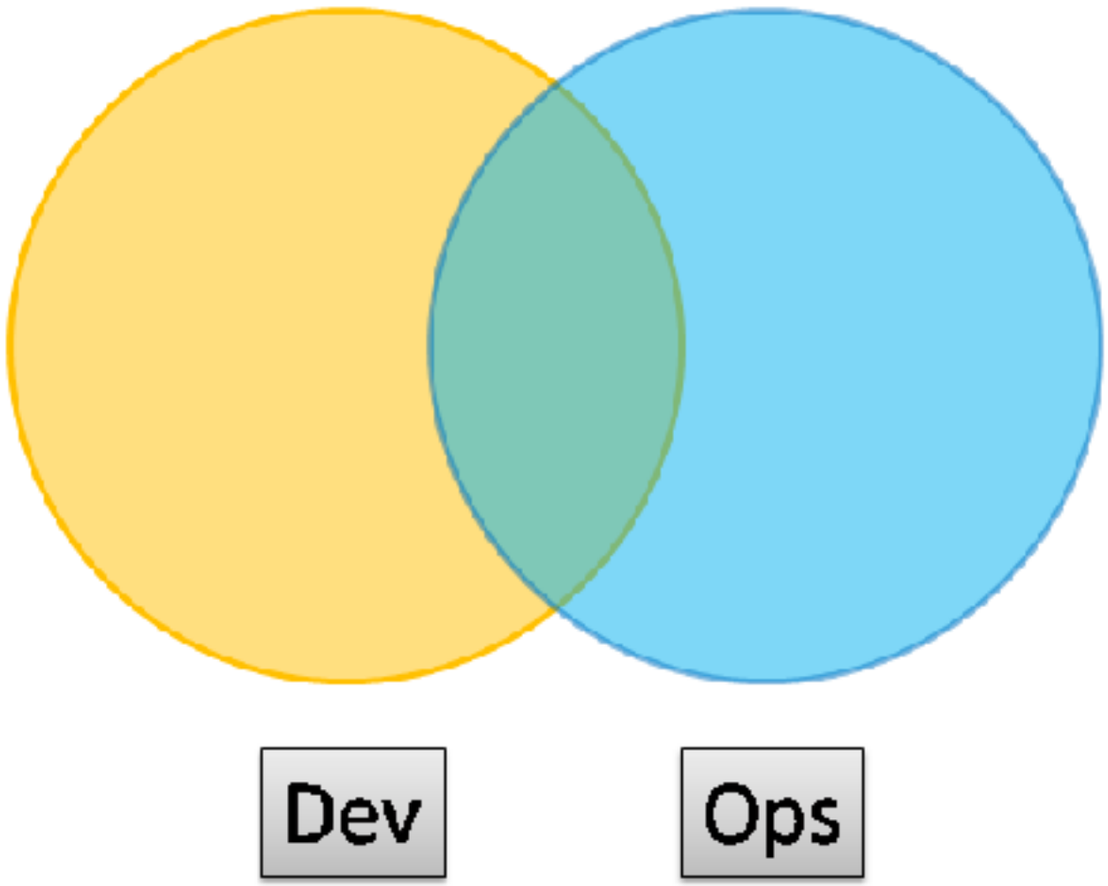
People -> Process -> Tool

Autonomous

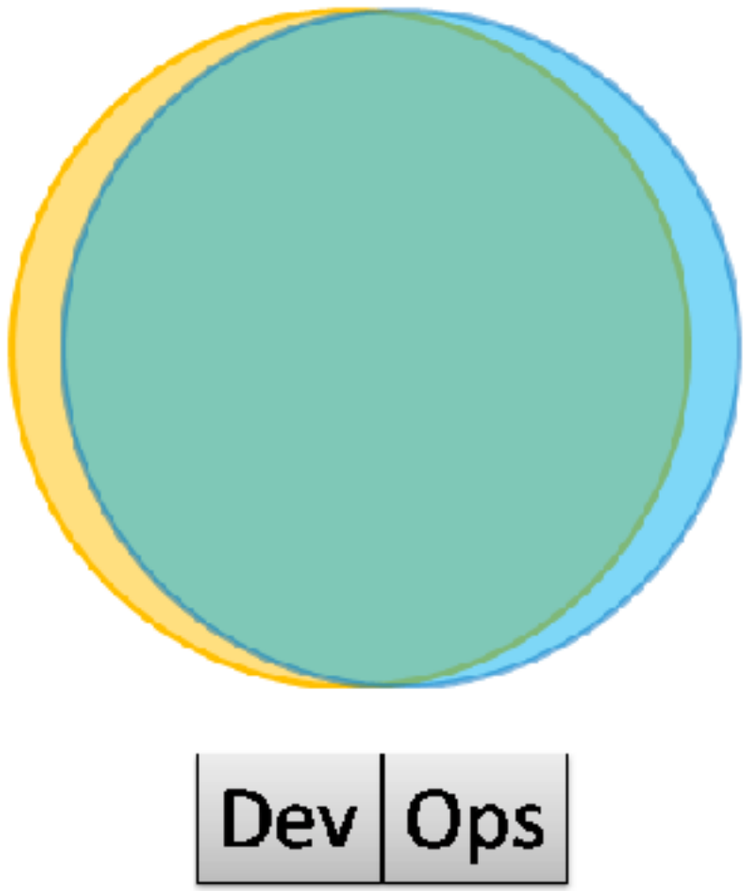


DevOps Topologies

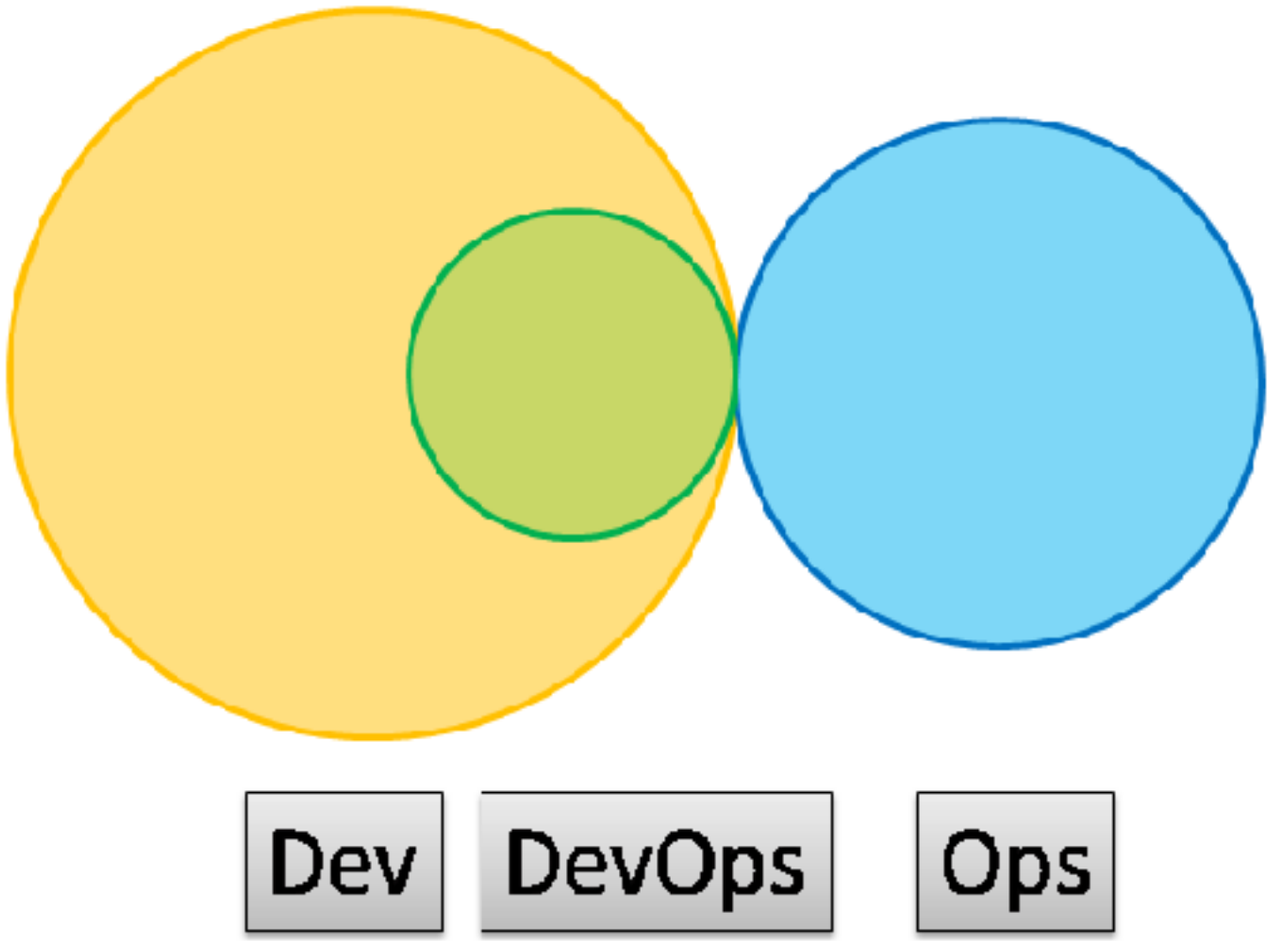
Type 1 – Smooth Collaboration



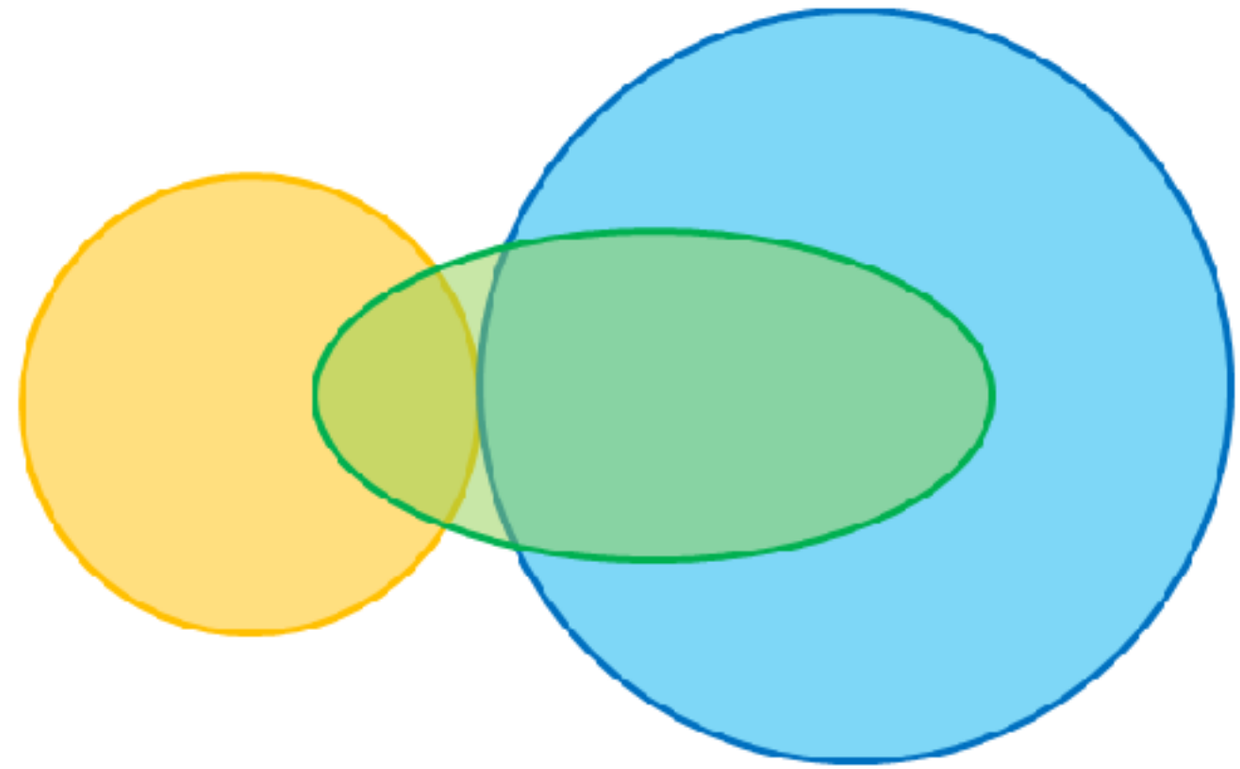
Type 2 – Fully Embedded



Type 3 – Infrastructure-as-a-Service

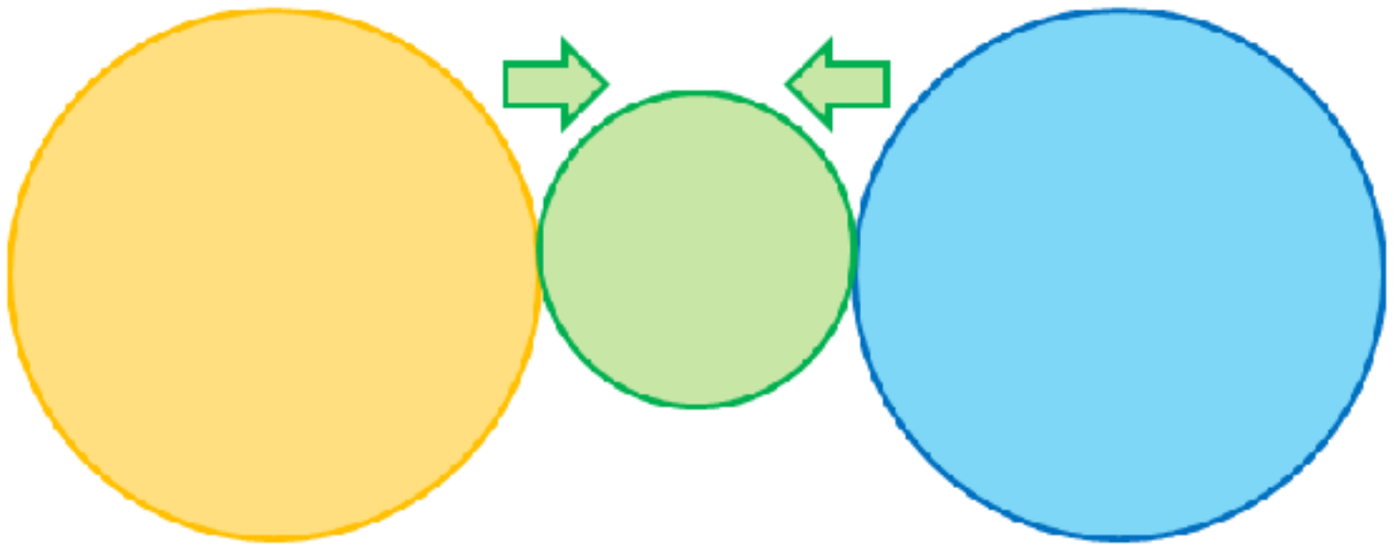


Type 4 – DevOps-as-a-Service



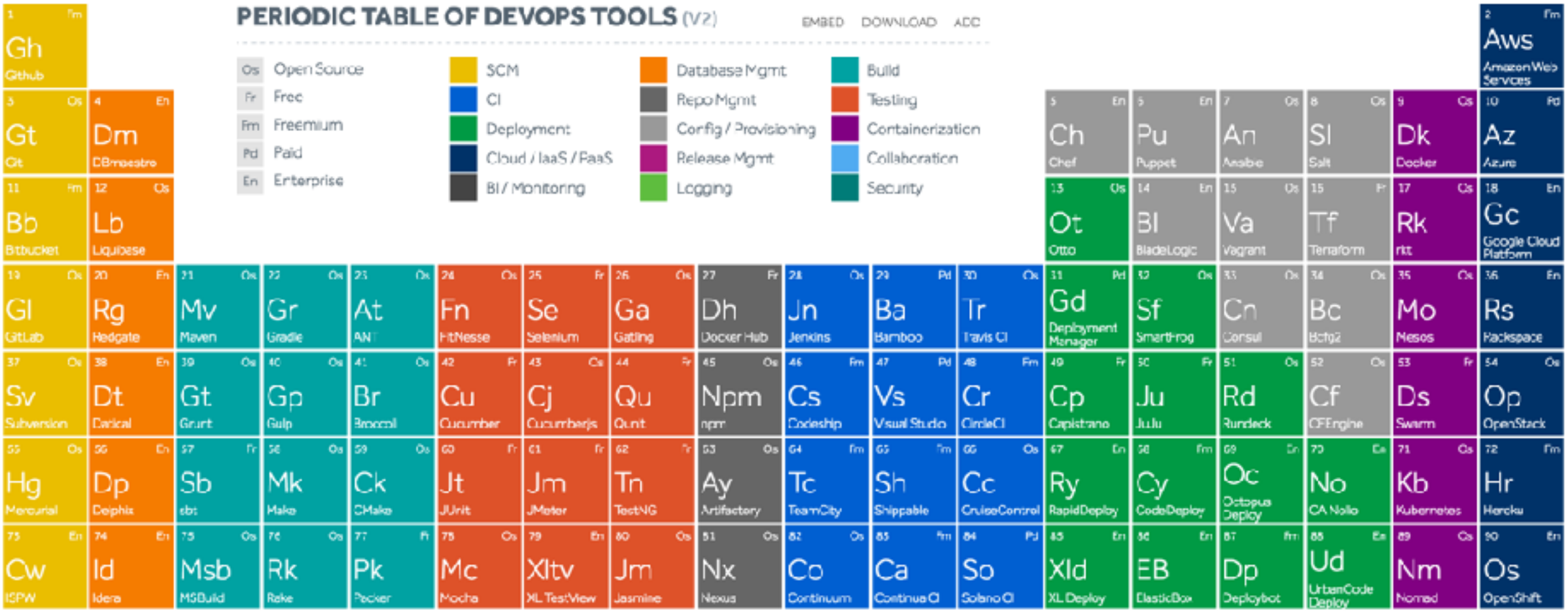
Dev DevOps Ops

Type 5 – Temporary DevOps Team



Dev DevOps Ops

DevOps Tools



https://xebialabs.com/periodic-table-of-devops-tools/

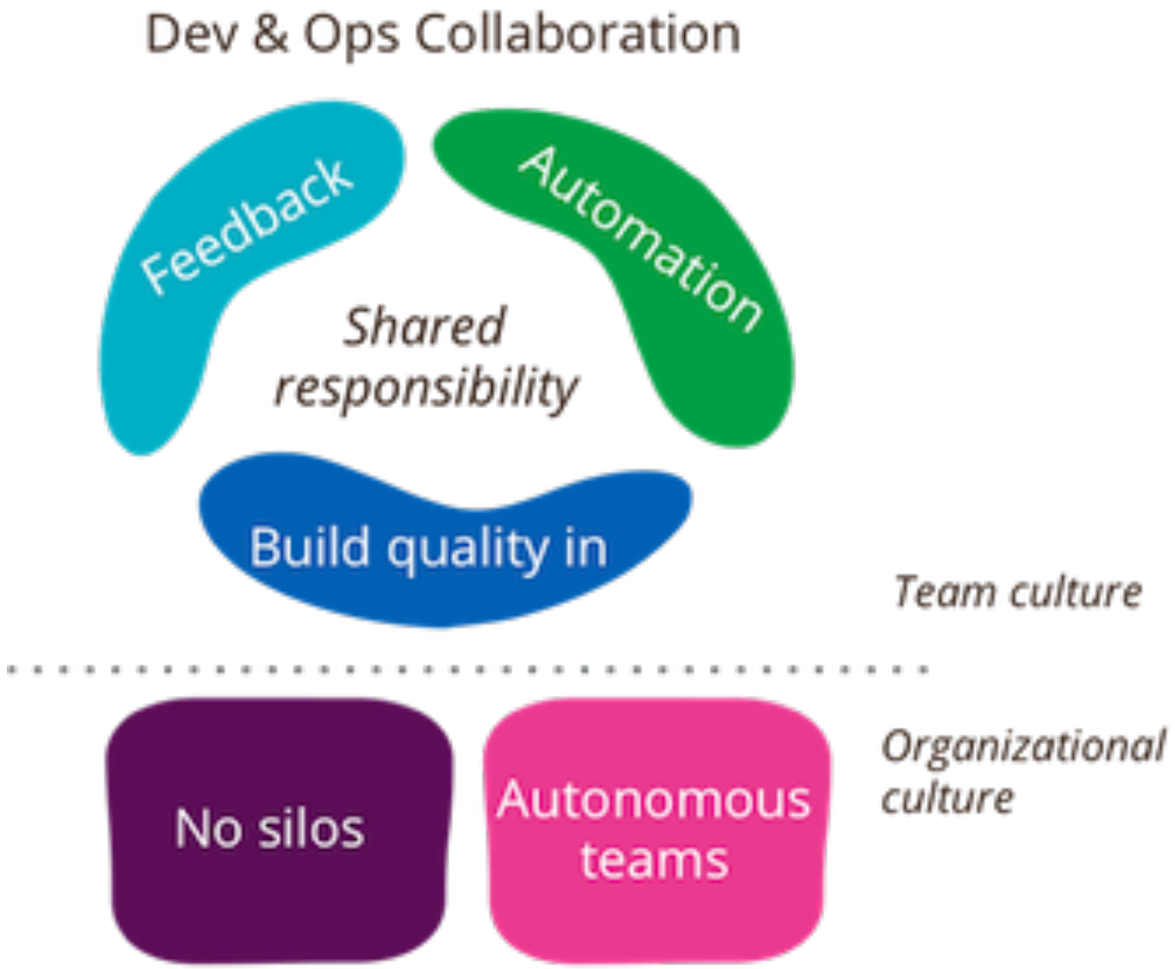
No DevOps Team

Problem department !!

DevOps != Tools

Tools enable DevOps

Team and Organization culture



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

DevOps success ?

How do i know something is wrong ?

Missed deadline

Site is always down

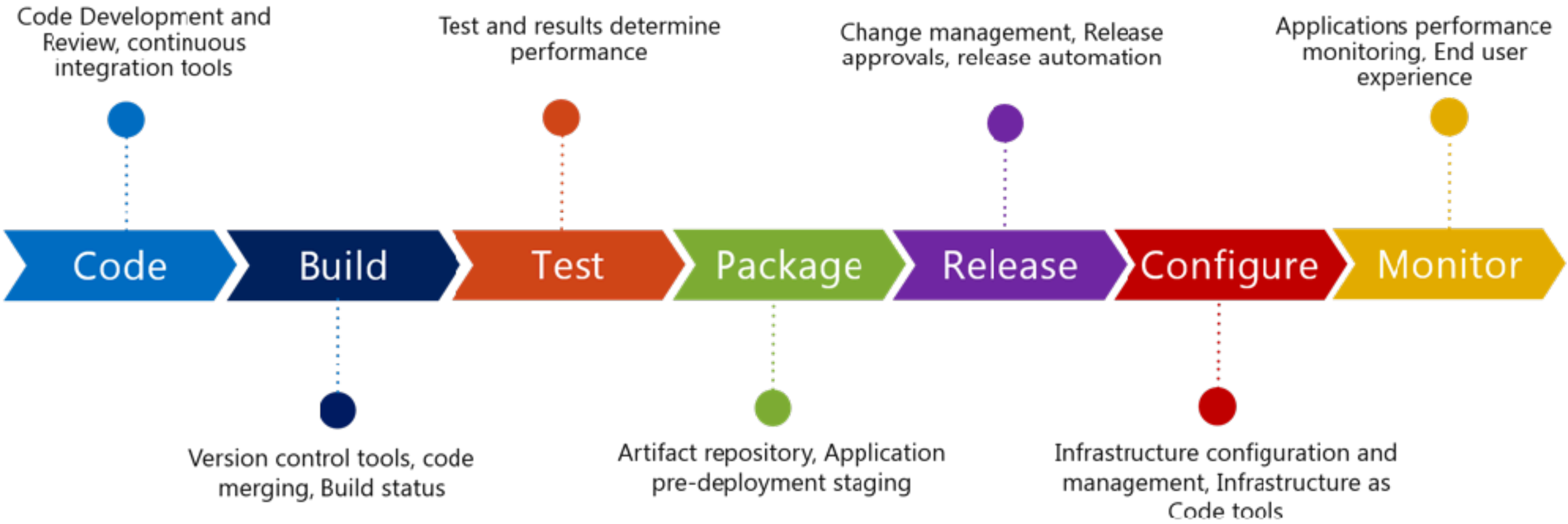
Unhappy customers

Long waits for small changes or fixes

Changes cost too much

What can i measure ?

DevOps Process & Tools



What can i measure ?

Mean Time to Recover/Repair (MTTR)

Mean Time to Detection (MTTD)

Change Lead Time

Change Failure Rate

Deployment or Change Frequency

Deployment Time

Percentage of successful deployments

What can i measure ?

Application Usage and Traffic

Application Performance

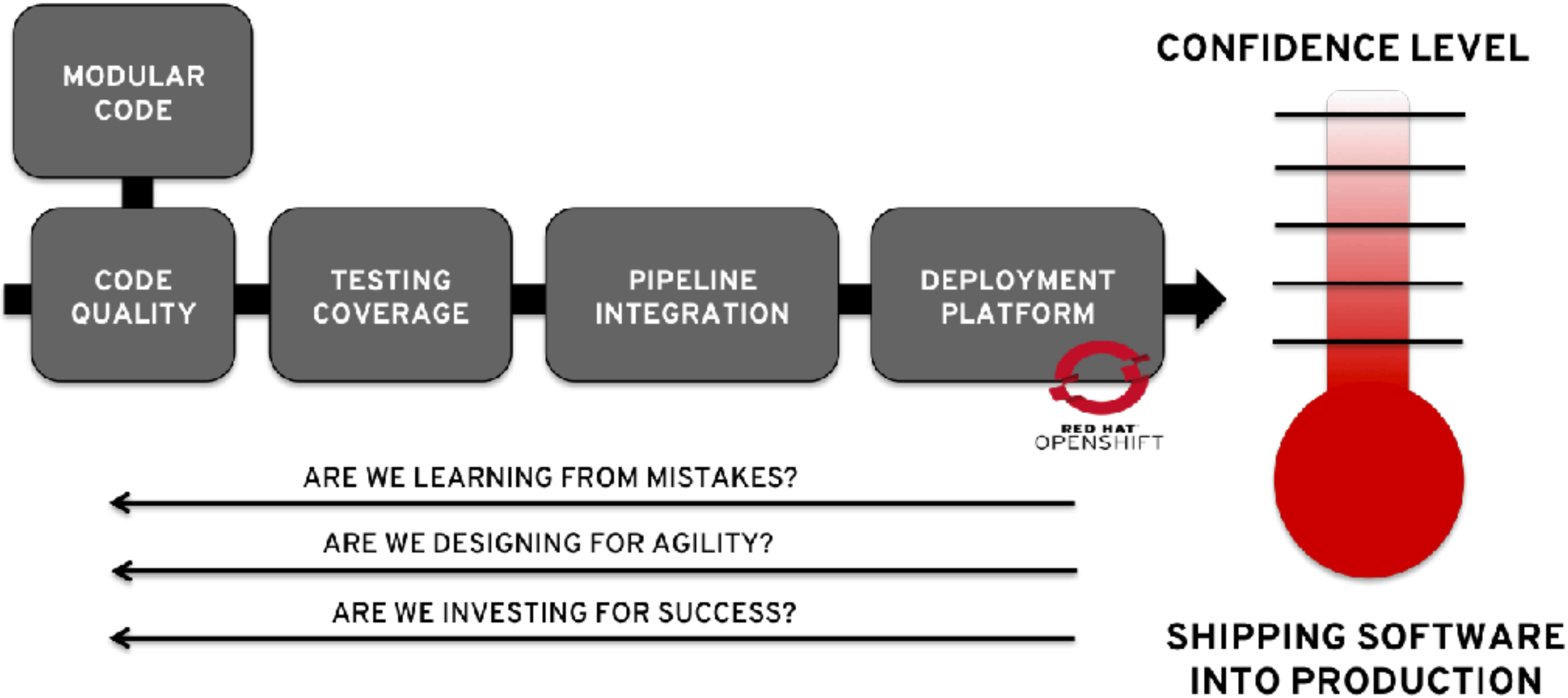
Automated Test Pass (%)

Defect Rate

Failed Deployments

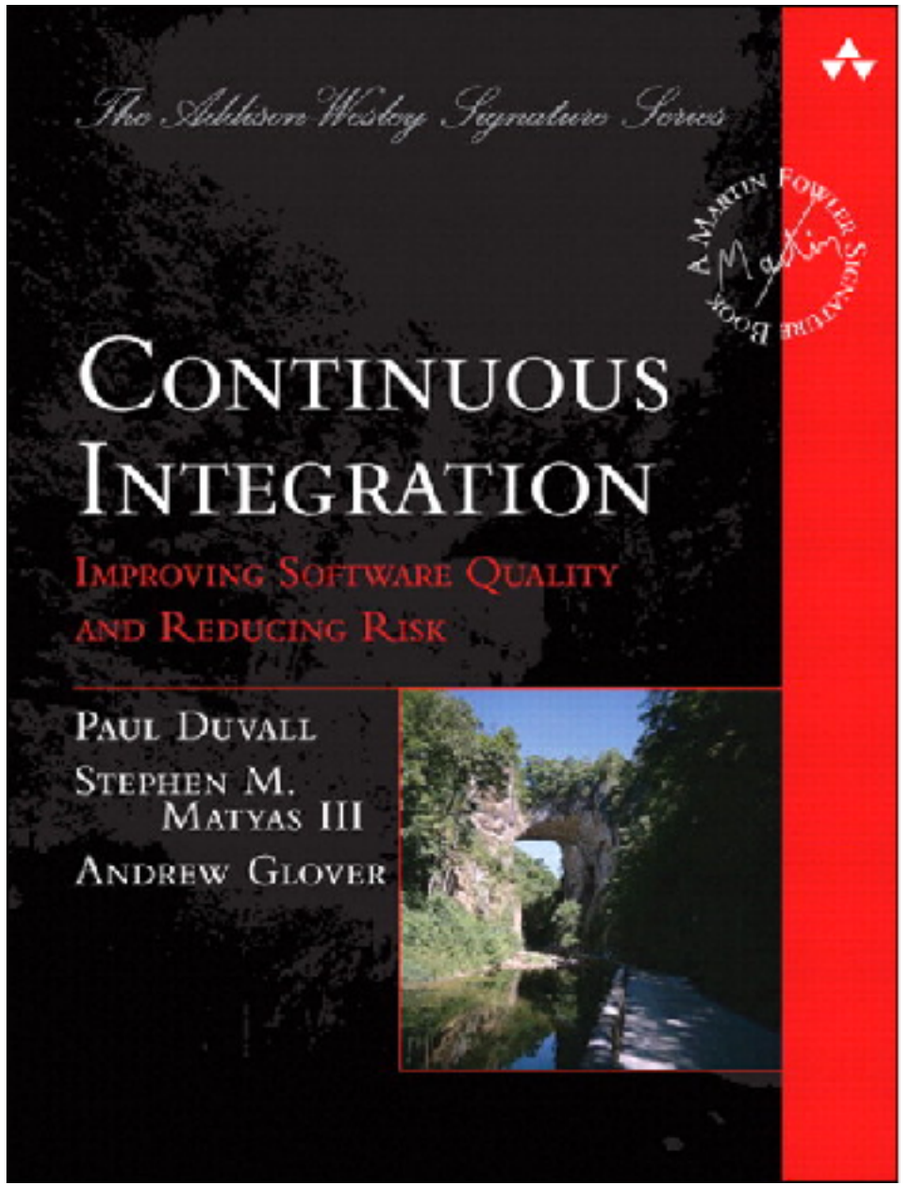
Availability

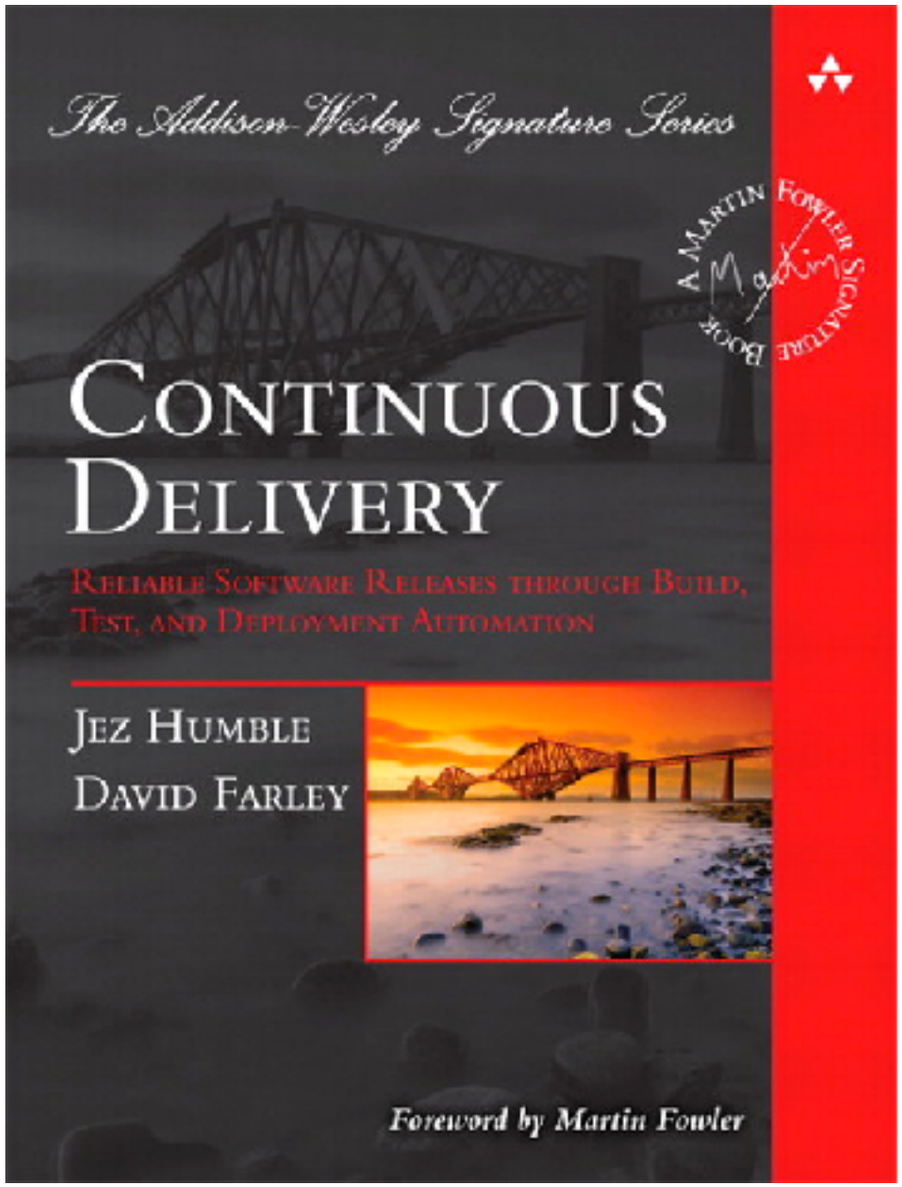
What can i measure ?



Start with Continuous Integration Continuous Delivery

Improve quality and reduce risk

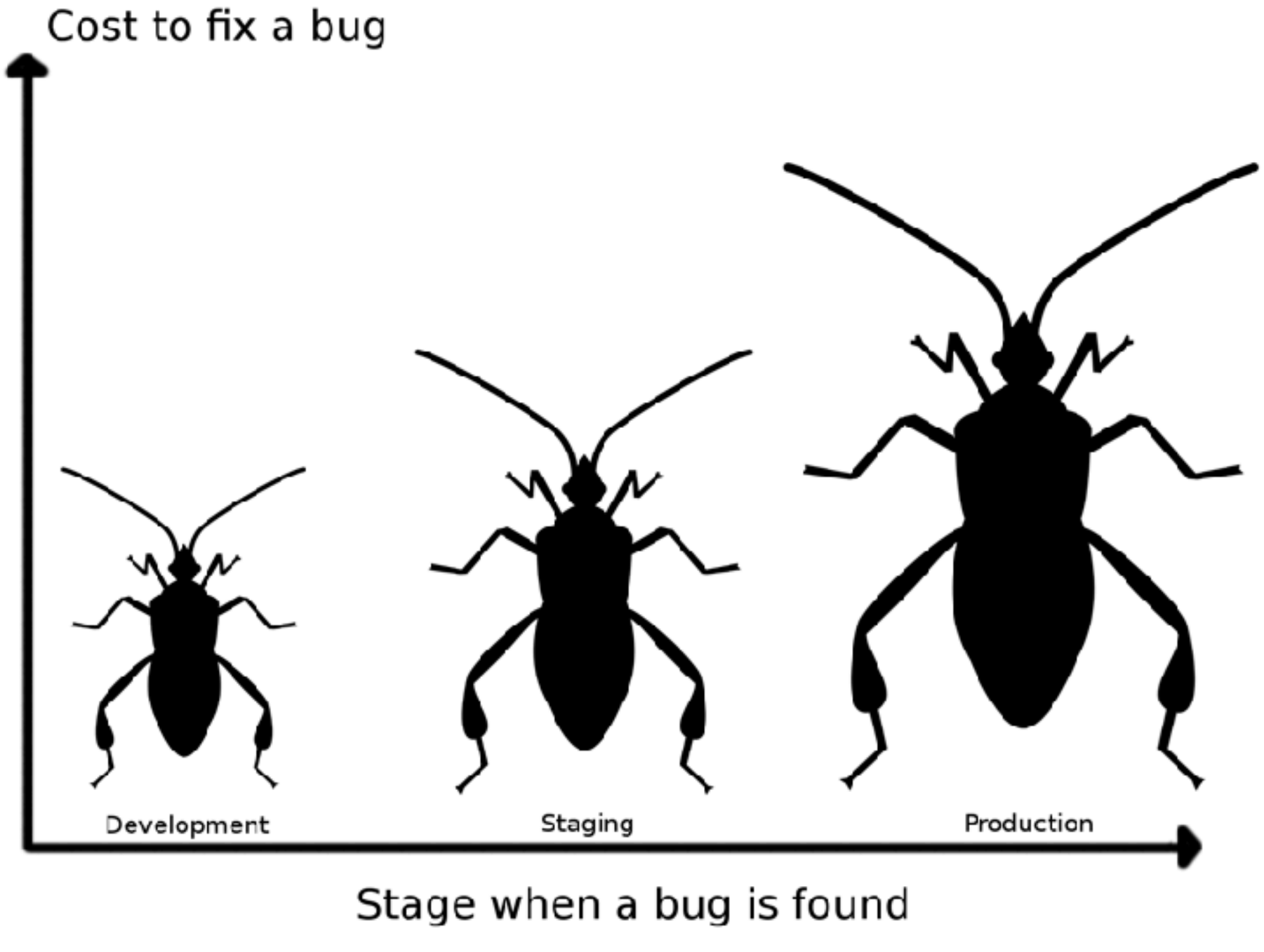


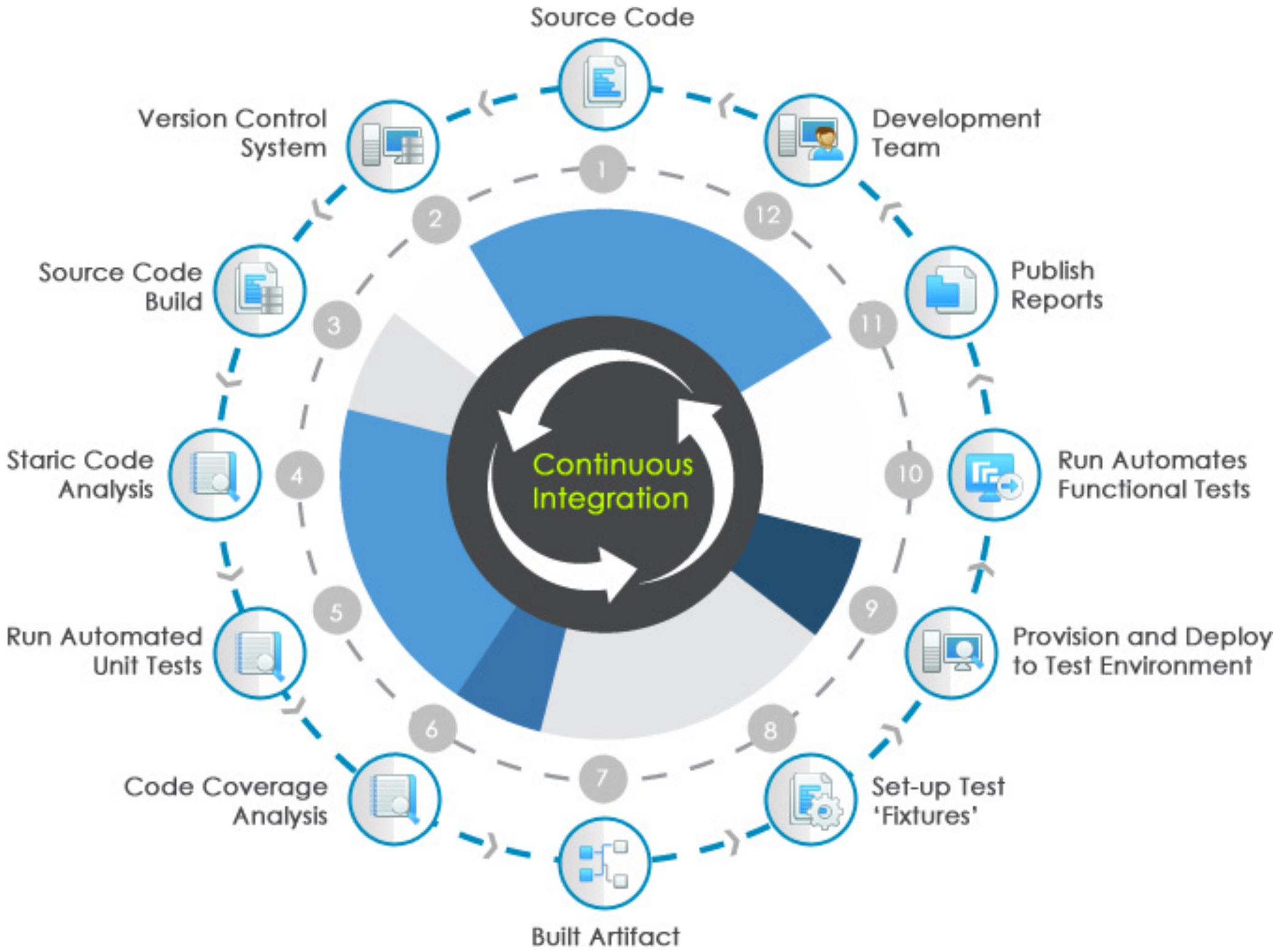


The cost of integration

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

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

Continuous Integration

Discipline to **integrate frequently**

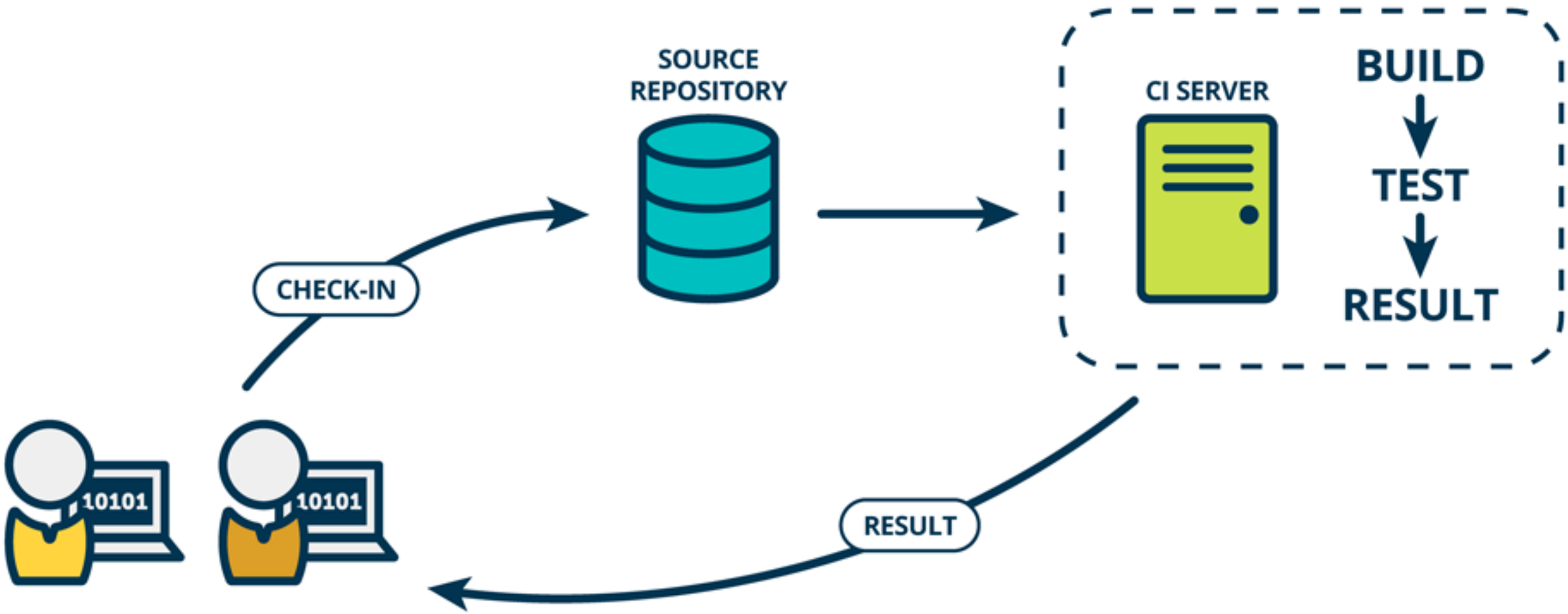
Continuous Integration

Strive to make **small change**

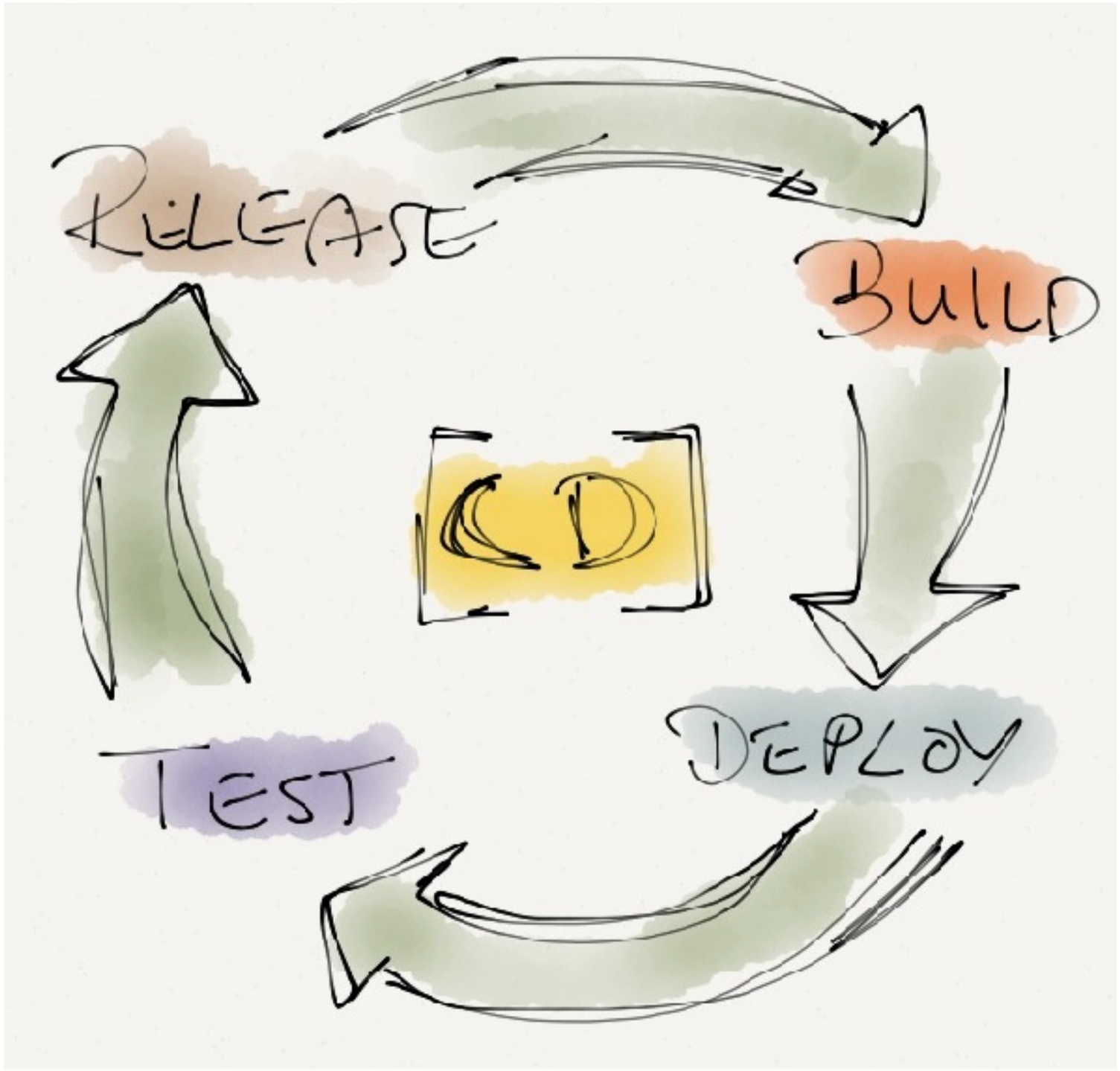
Continuous Integration

Strive for **fast feedback**

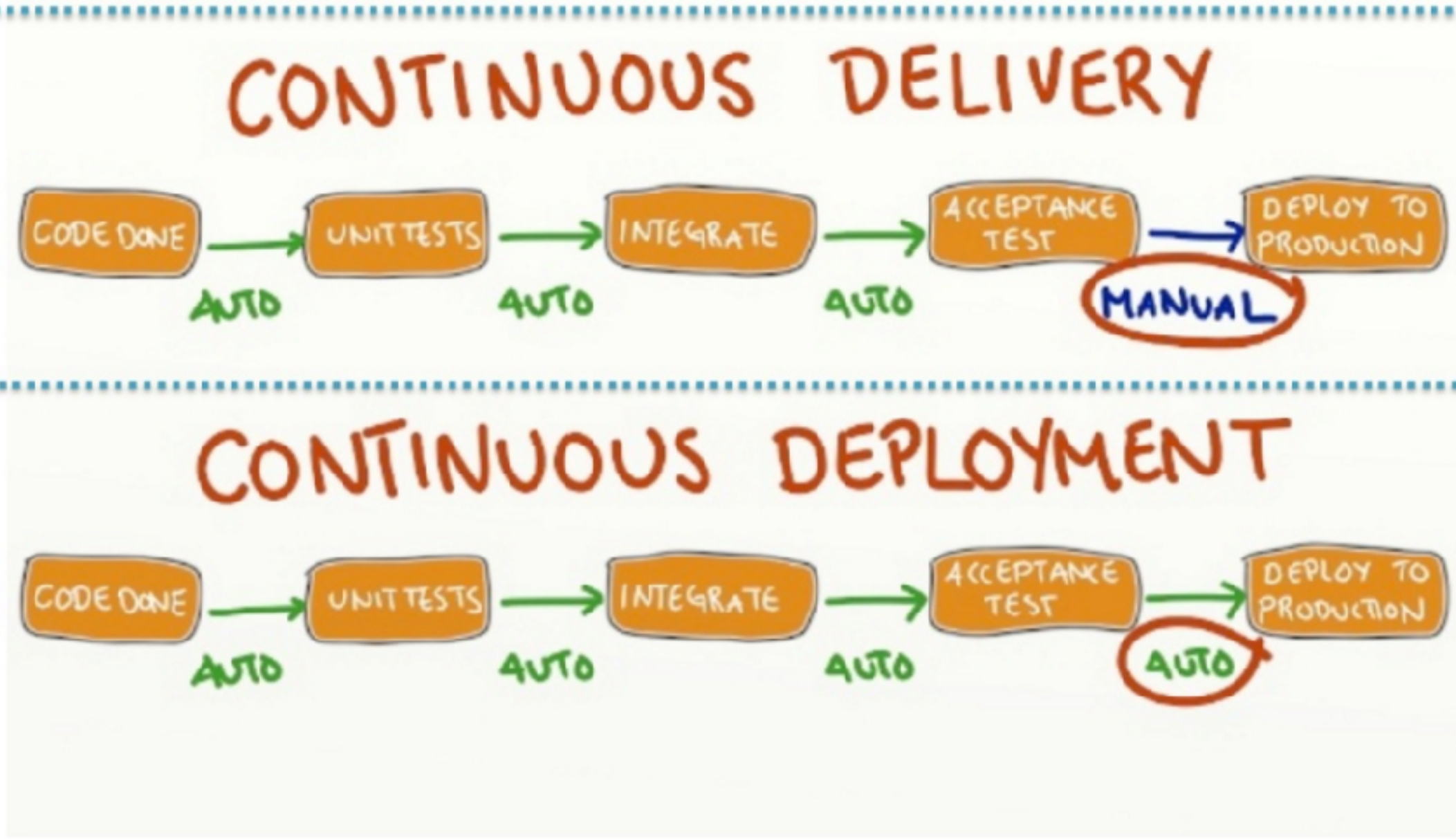
Continuous Integration



CD ?

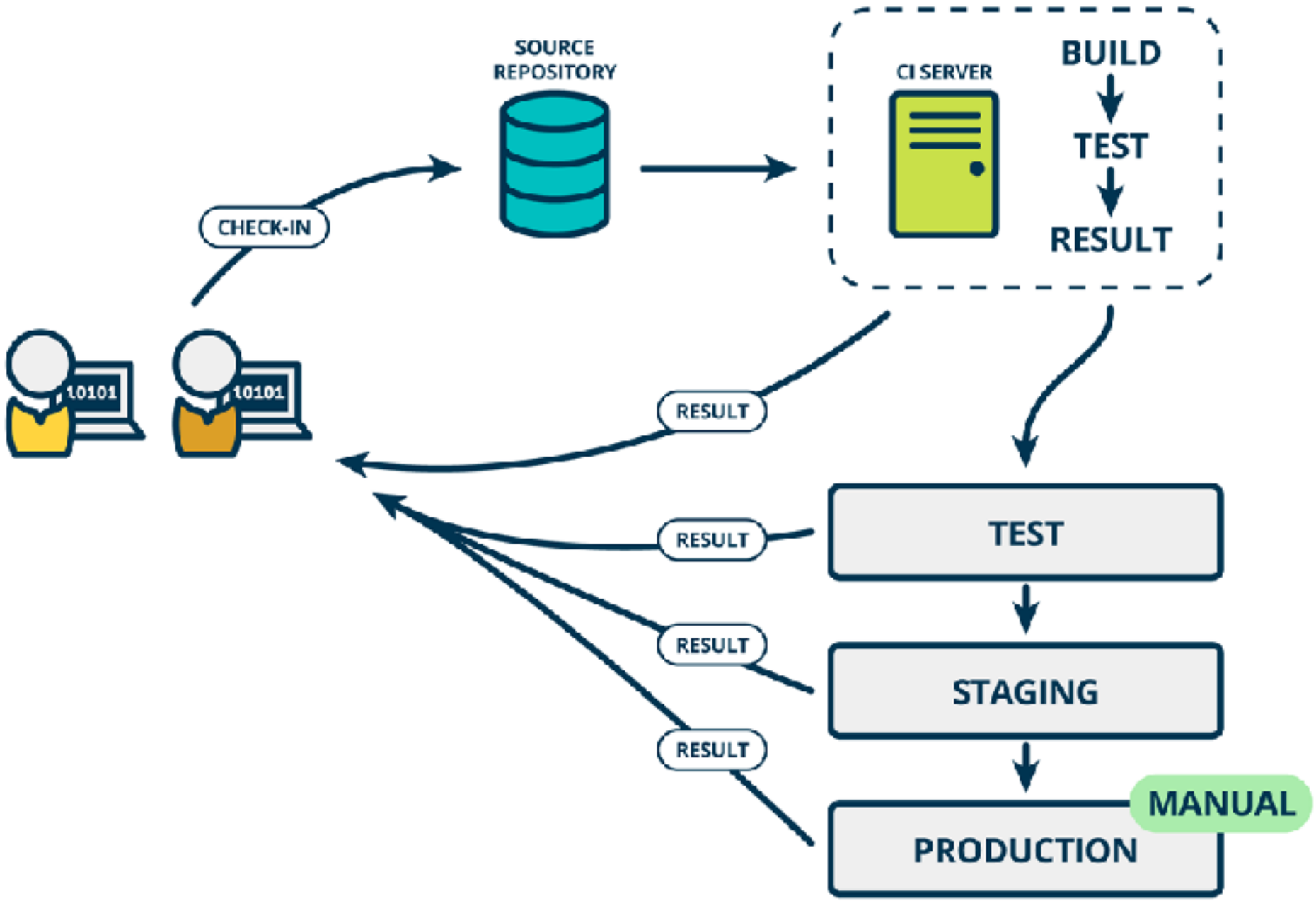


CD ?

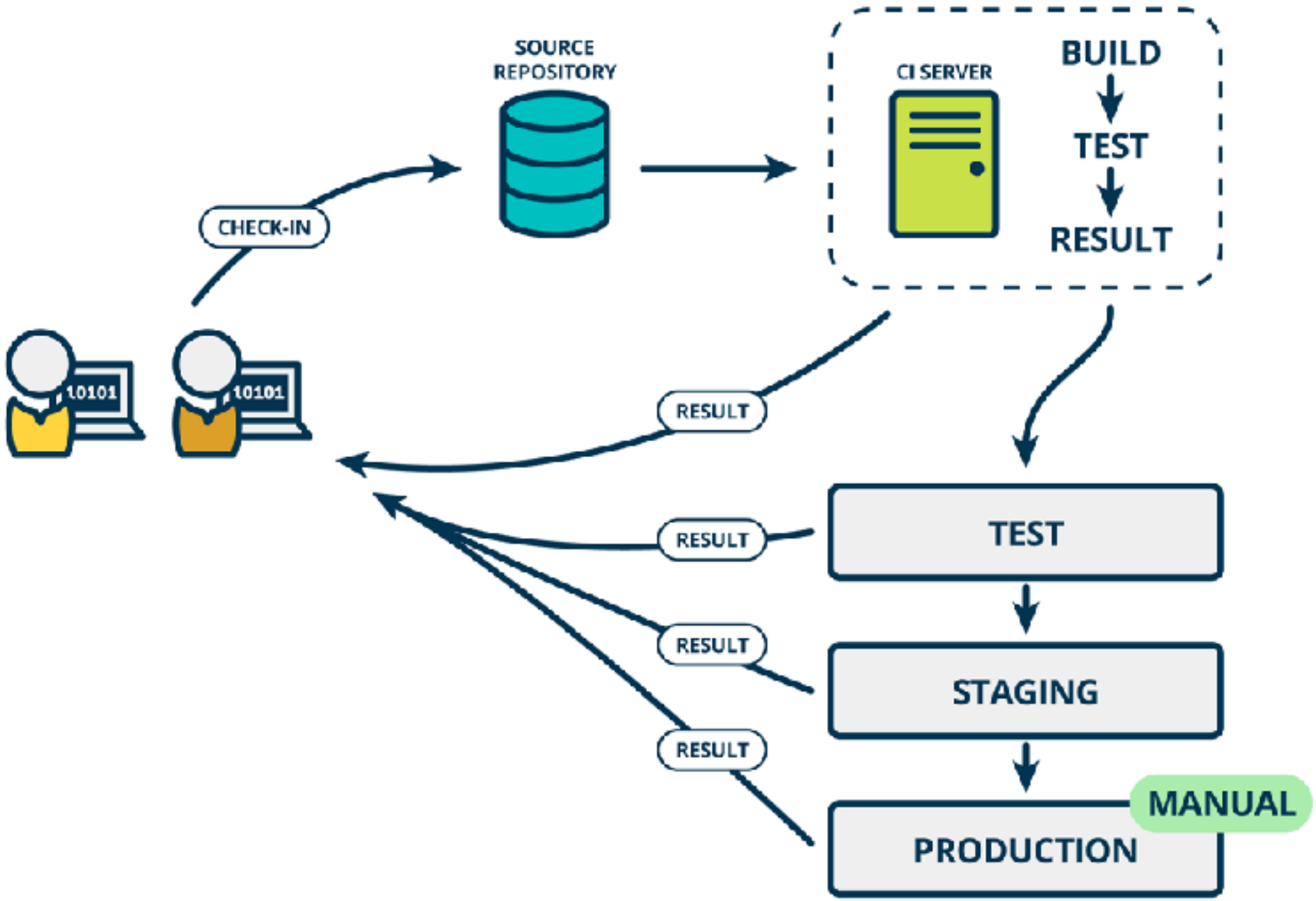


http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment

Continuous Delivery



Rise of DevOps



Continuous Integration

is a Software development **practices**

Practice 1

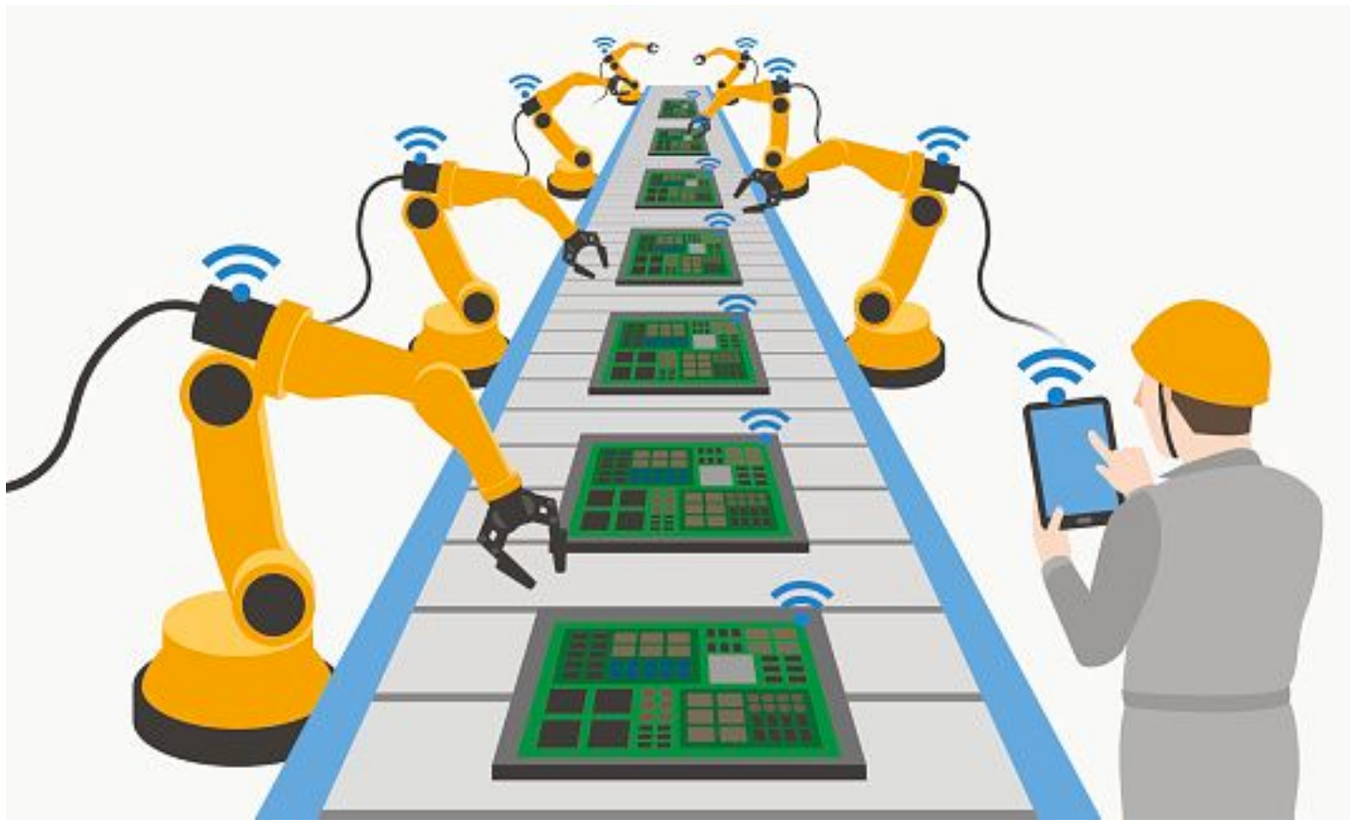
Maintain a single source repository

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

Practice 2

Automated the build

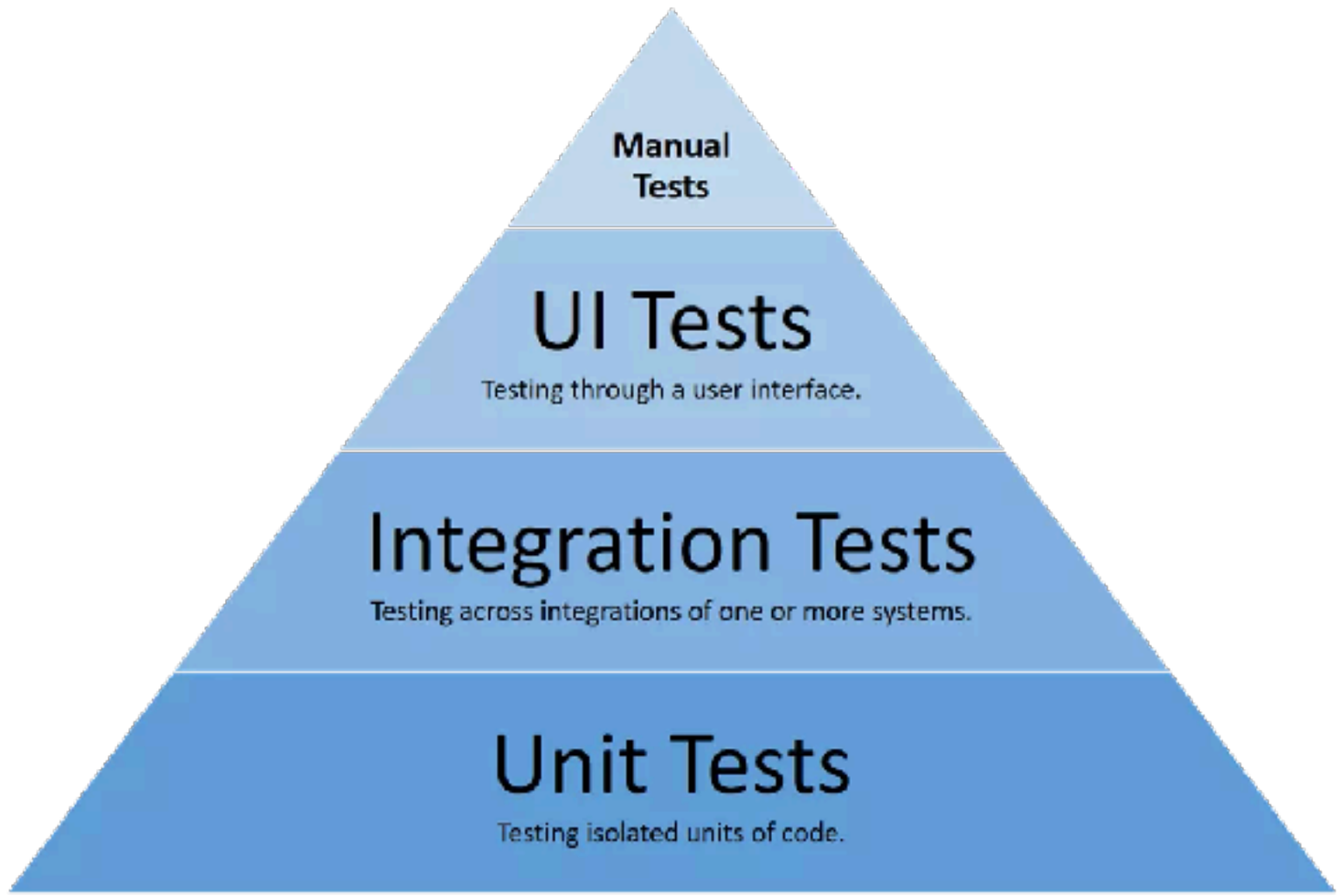
Automated environment for builds



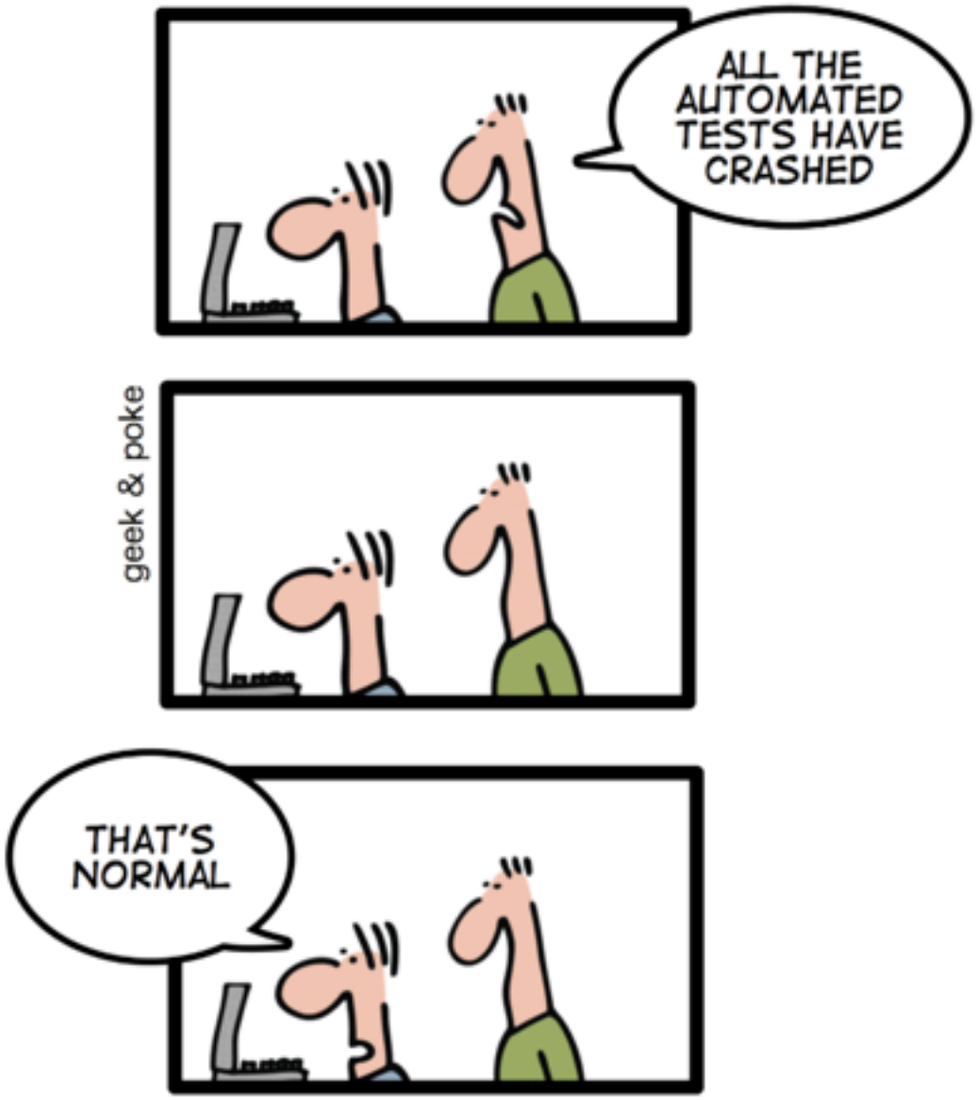
Practice 3

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

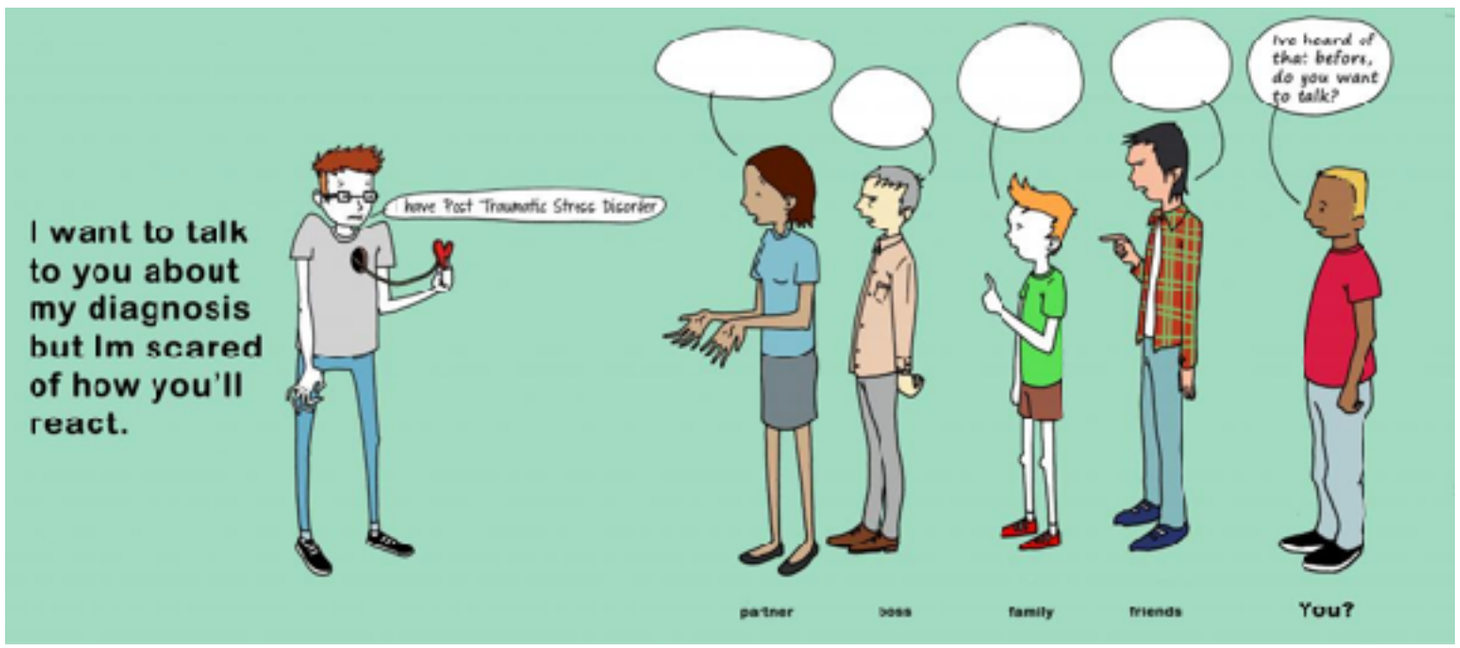


Practice 4

Everyone **commits** to the mainline everyday

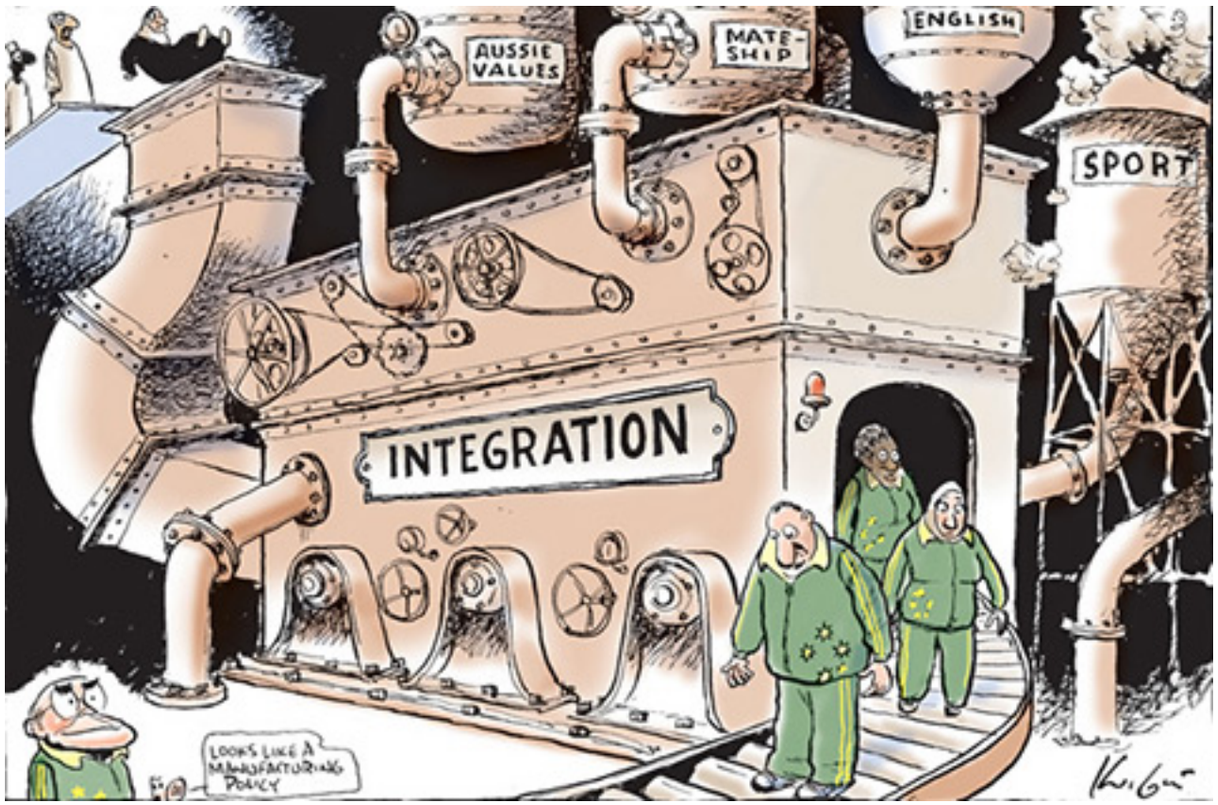
Integration is about **communication**

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

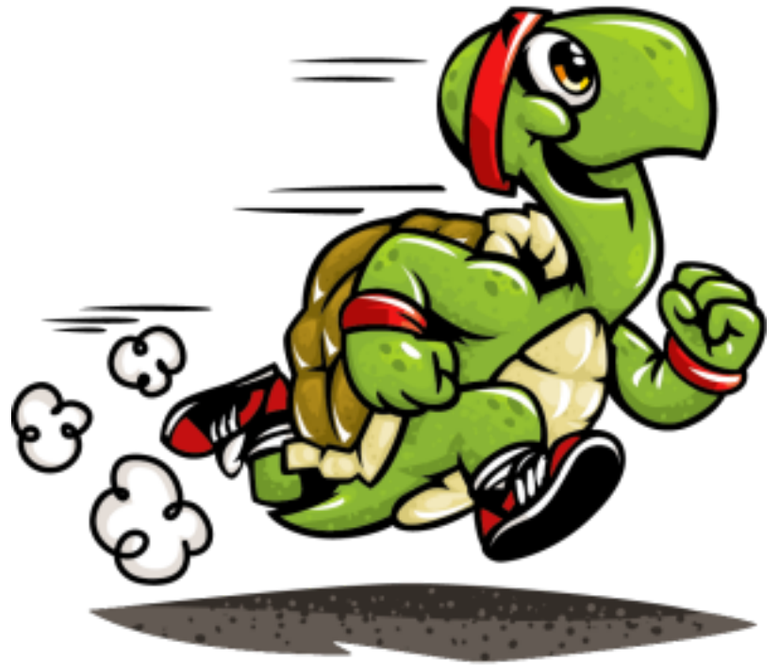
Fix broken builds **immediately**

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

Practice 7

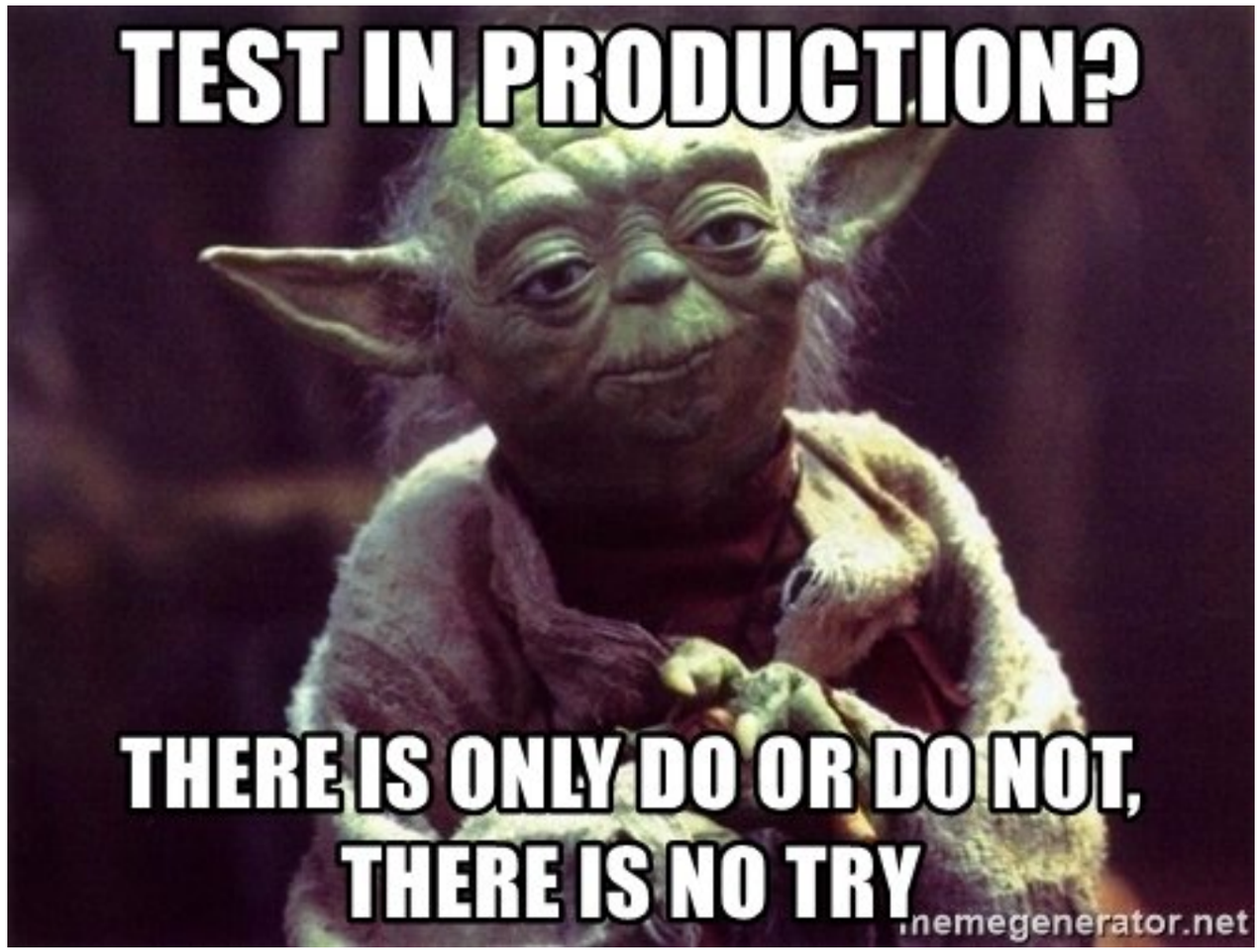
Keep the build **fast**

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

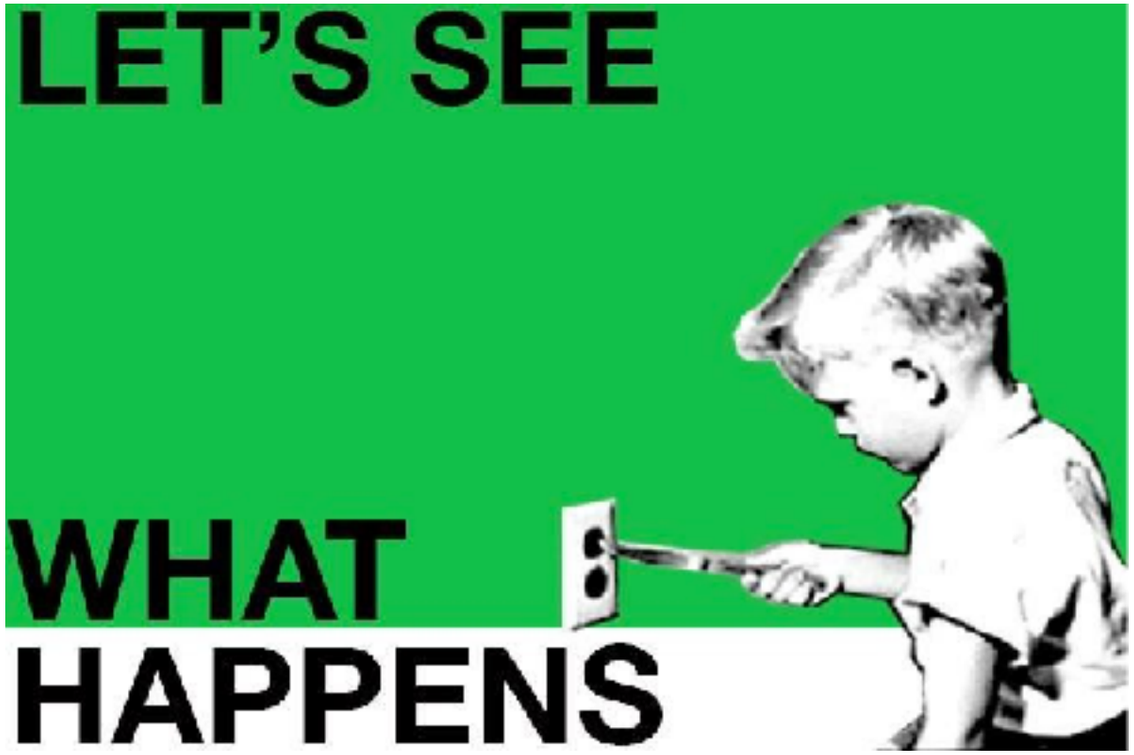
Make sure well known place where people can find

Practice 10

Everyone can see what's happening

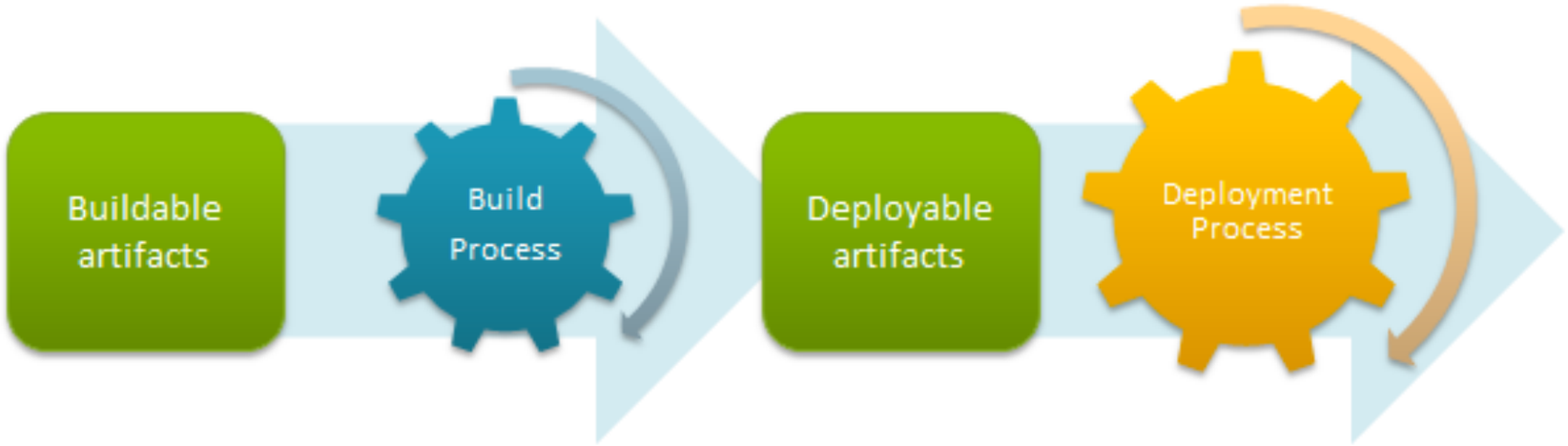
Easier to see the state of the system and changes

Show the good information



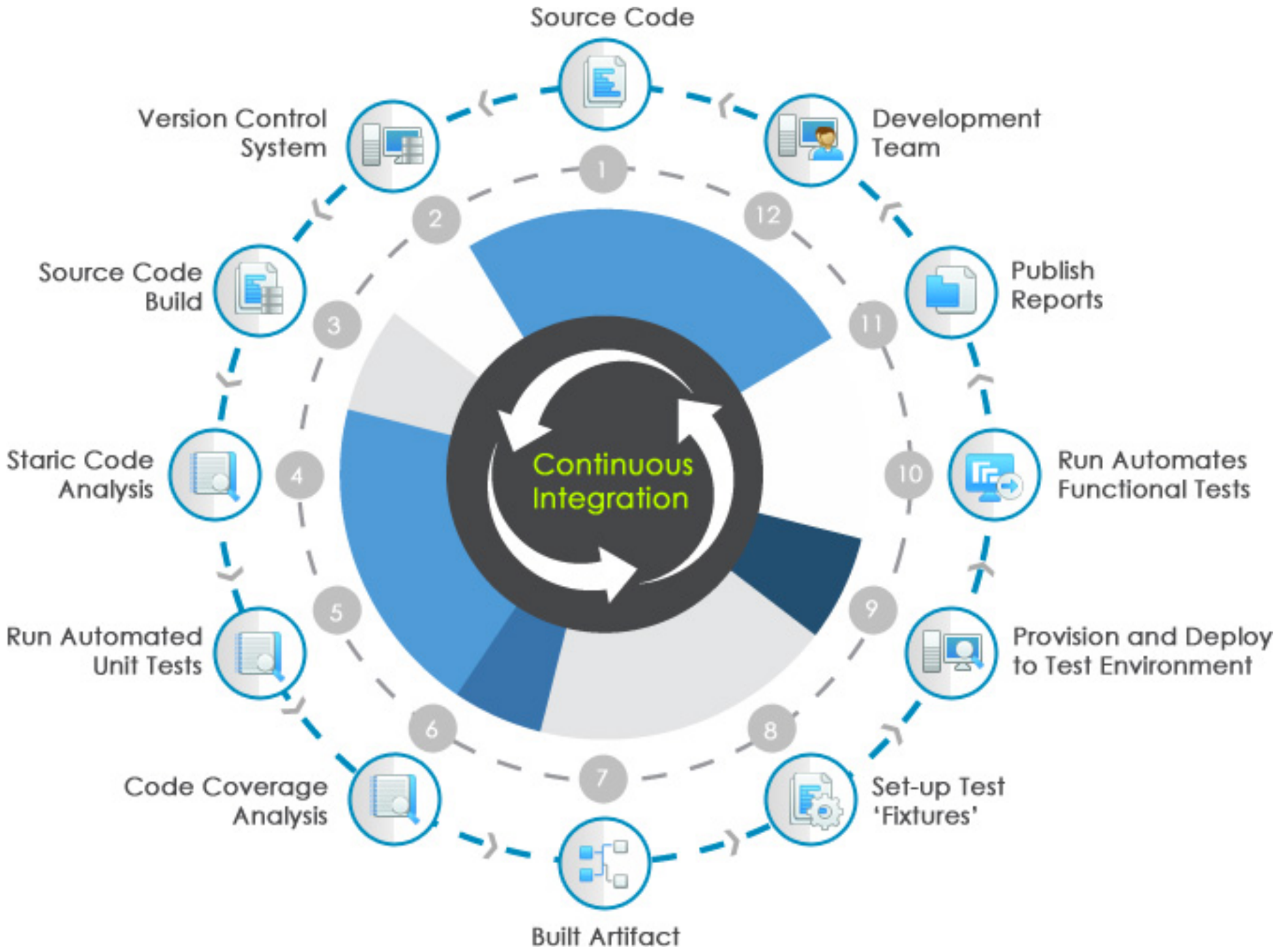
Practice 11

Automated deployment

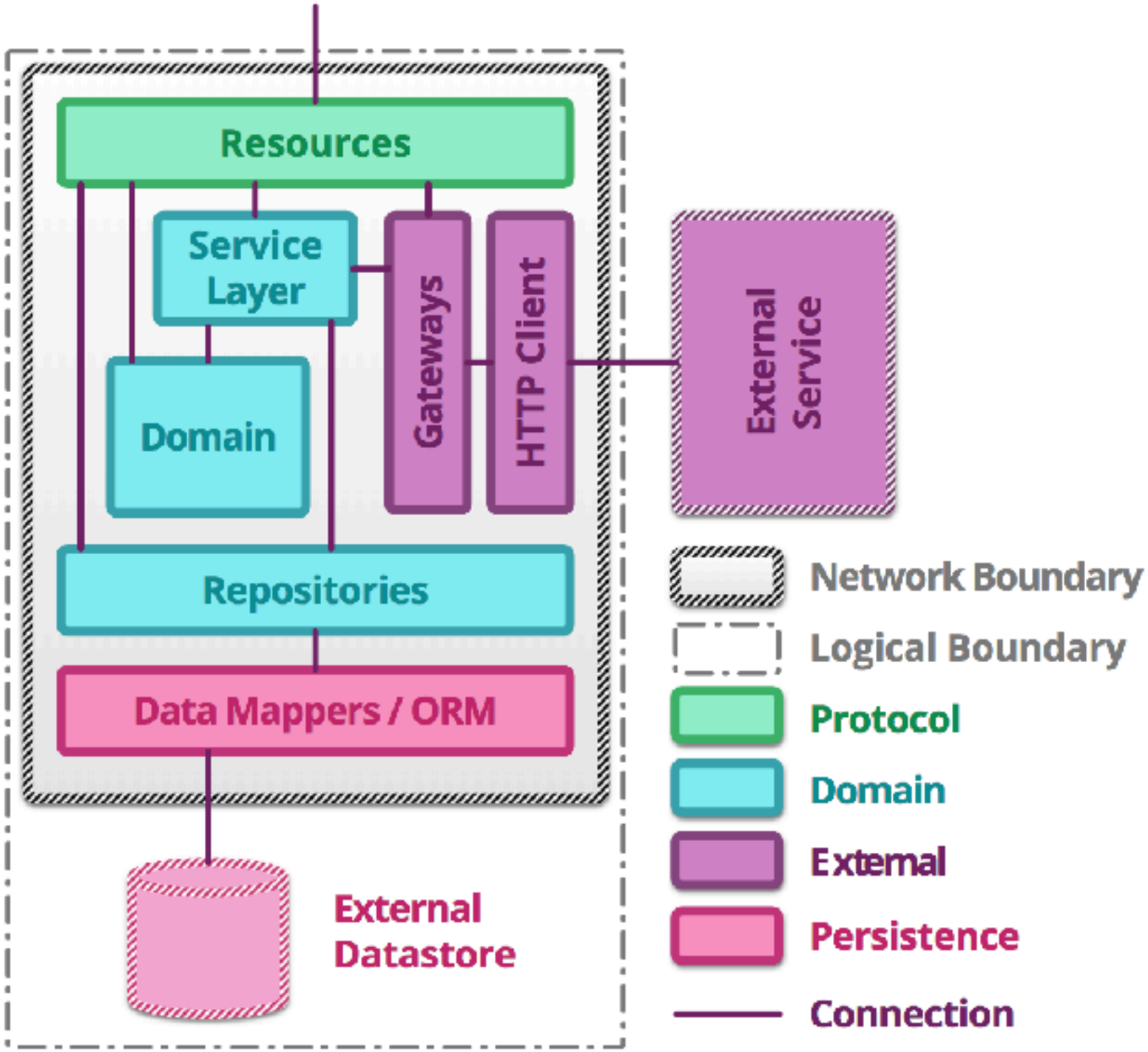


“Behind every successful agile project, there is a

Continuous Integration Server”



Let’s workshop

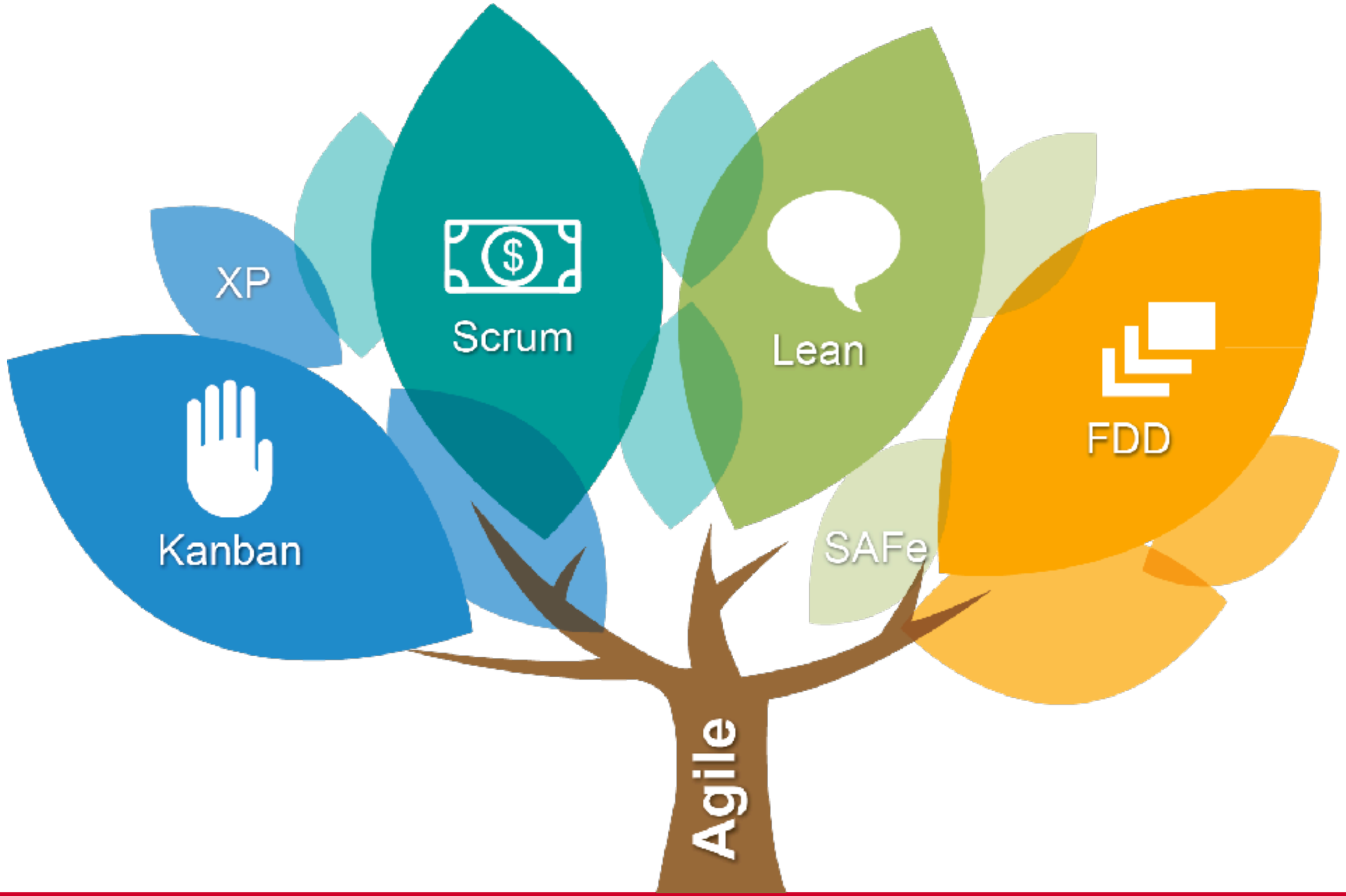


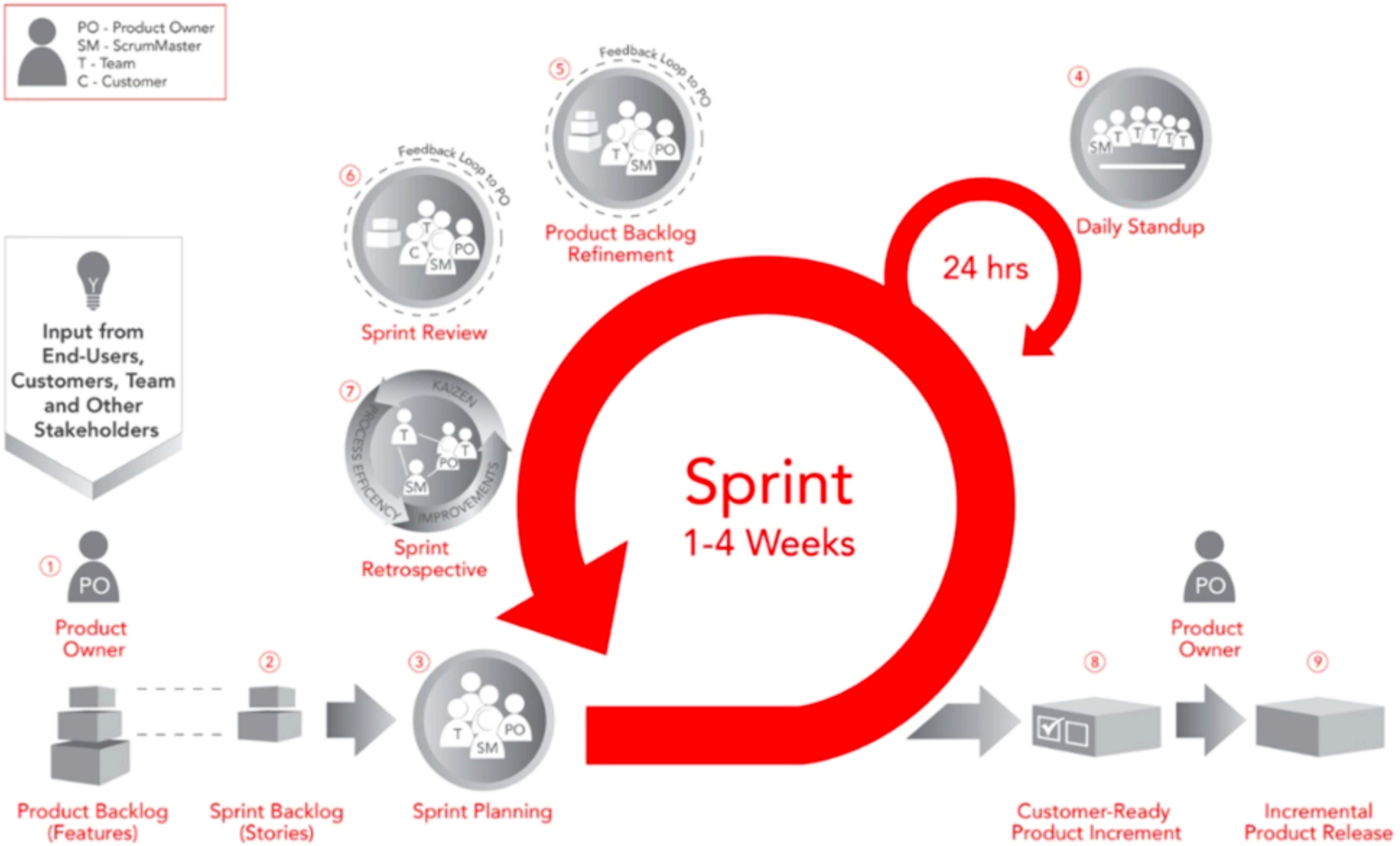
Development

Testing

Deployment

Summary





Agile manifestos

THE AGILE MANIFESTO

We are uncovering better ways of developing software by doing it and helping others do it.

CUSTOMER COLLABORATION over contract negotiation

INDIVIDUALS AND INTERACTIONS over processes and tools

RESPONDING TO **CHANGE** over following a plan

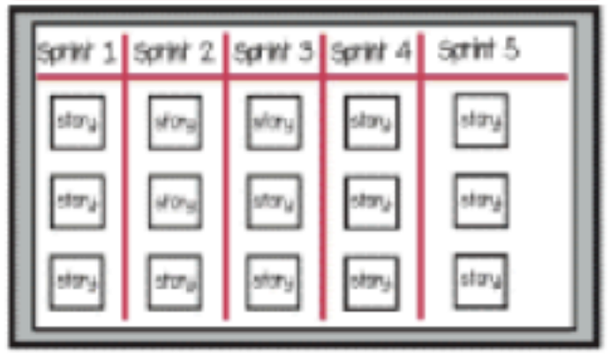
WORKING SOFTWARE over full documentation

Agile principles

1 Satisfy the **customer**

Welcome **change**

Deliver **frequently**



4 Work **together**

5 Trust and **support**

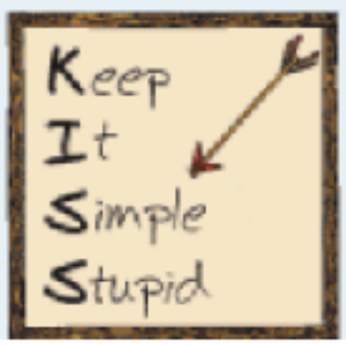
Face-to-face **conversation**

Working **software**

8 Sustainable **development**

9 **Technical** Excellence

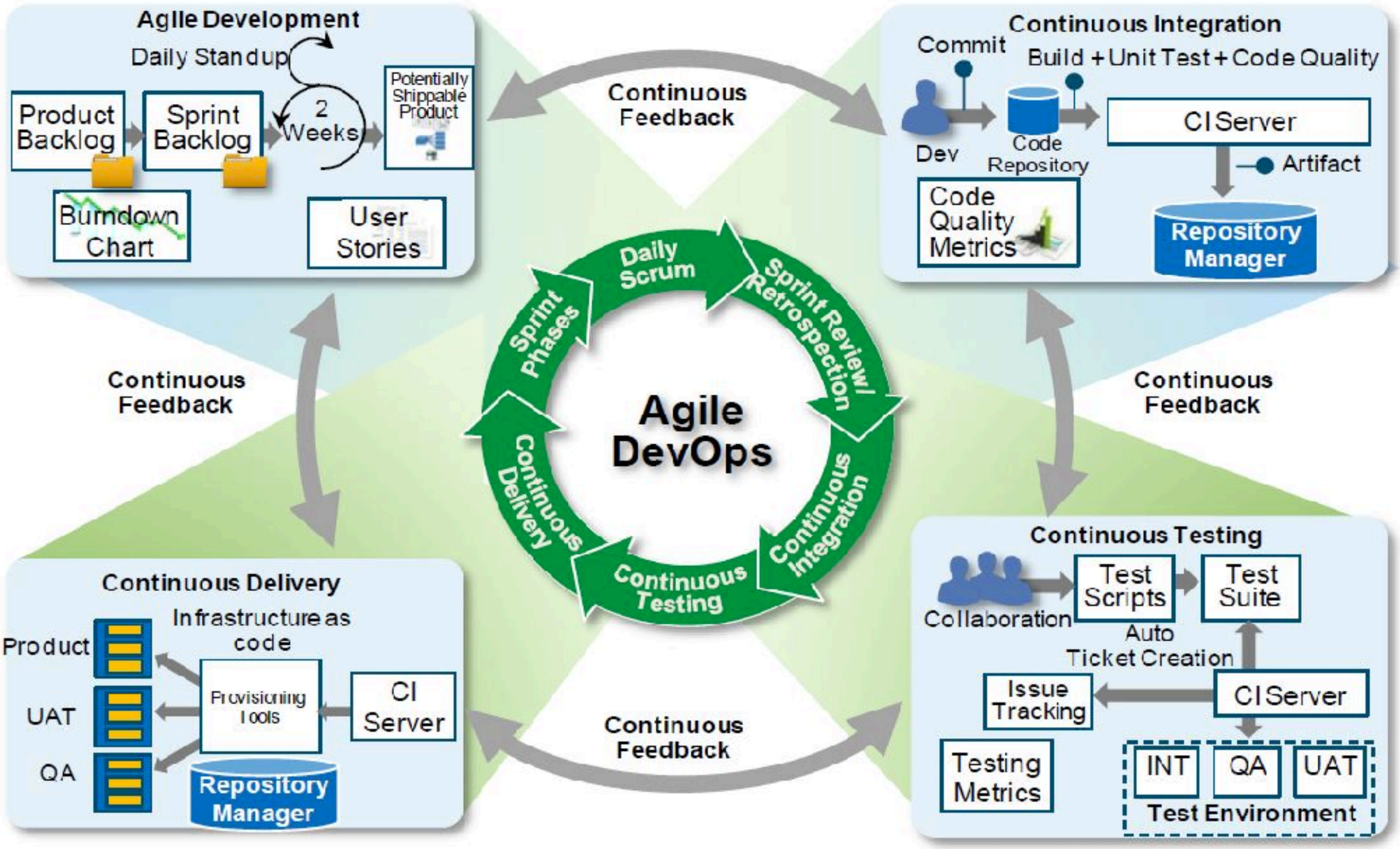
10 Maintain **simplicity**



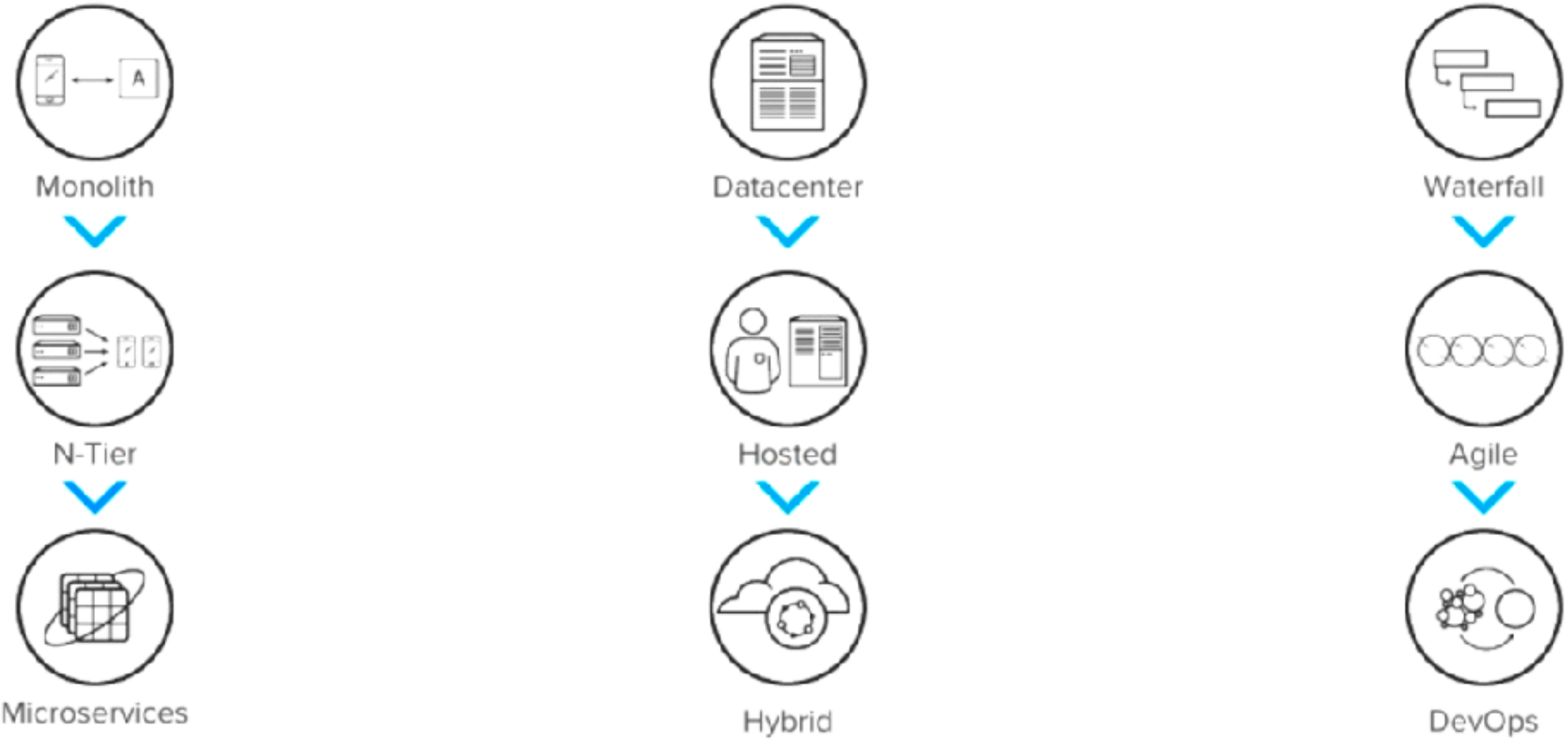
11 Self-organizing **teams**

12 Reflect and **adjust**

Origin by https://www.knowledgetrain.co.uk, modified by Jacky Shen



Continuous Improvement



Improve Time to Value

TIME TO VALUE	DEVELOPMENT PROCESS	APPLICATION ARCHITECTURE	DEPLOYMENT AND PACKAGING	APPLICATION INFRASTRUCTURE
Months and years	Waterfall 	Monolithic 	Physical servers 	Datacenter 
Weeks and months	Agile 	N-tier 	Virtual servers 	Hosted 
Days and weeks	DevOps 	Microservices 	Containers 	Cloud