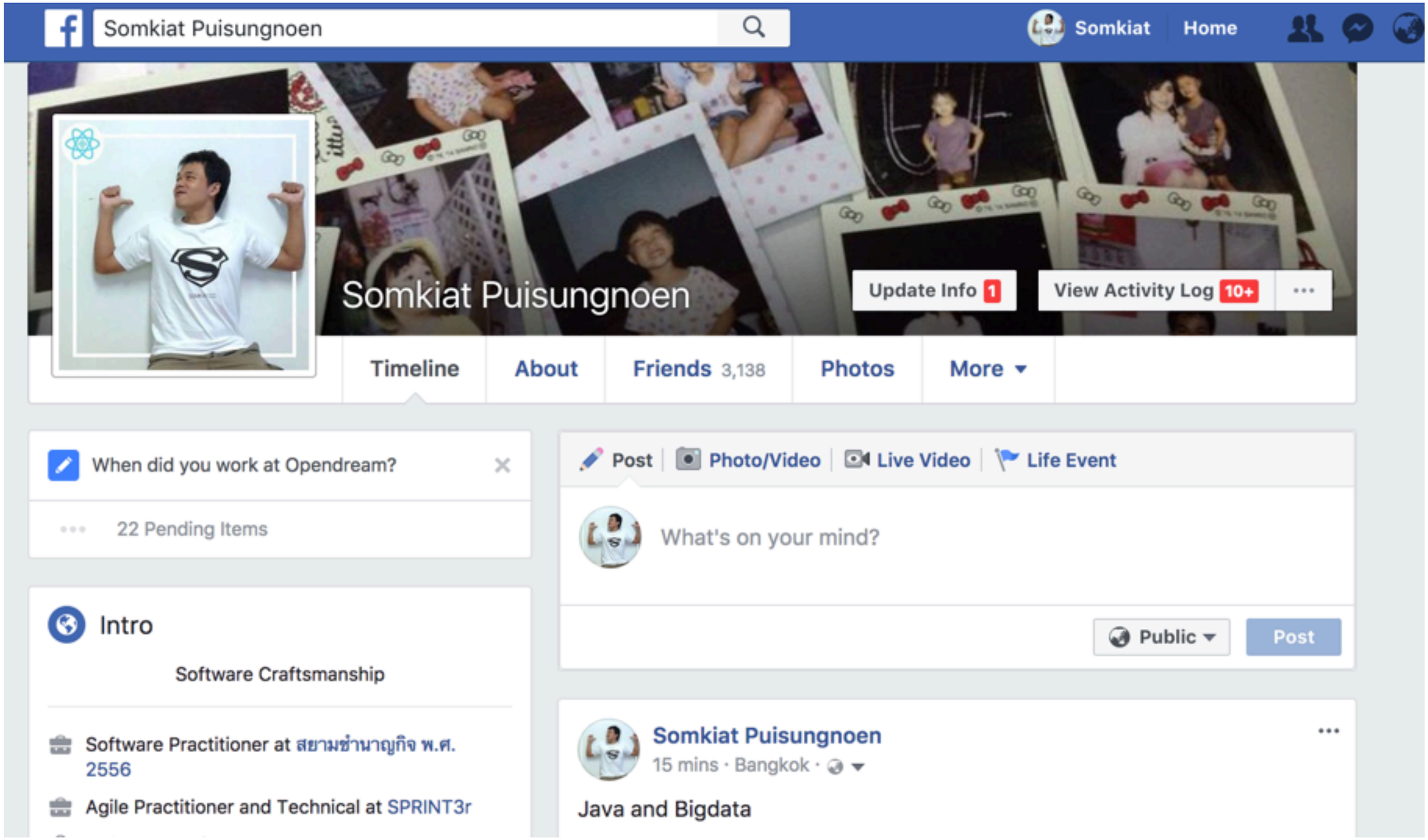
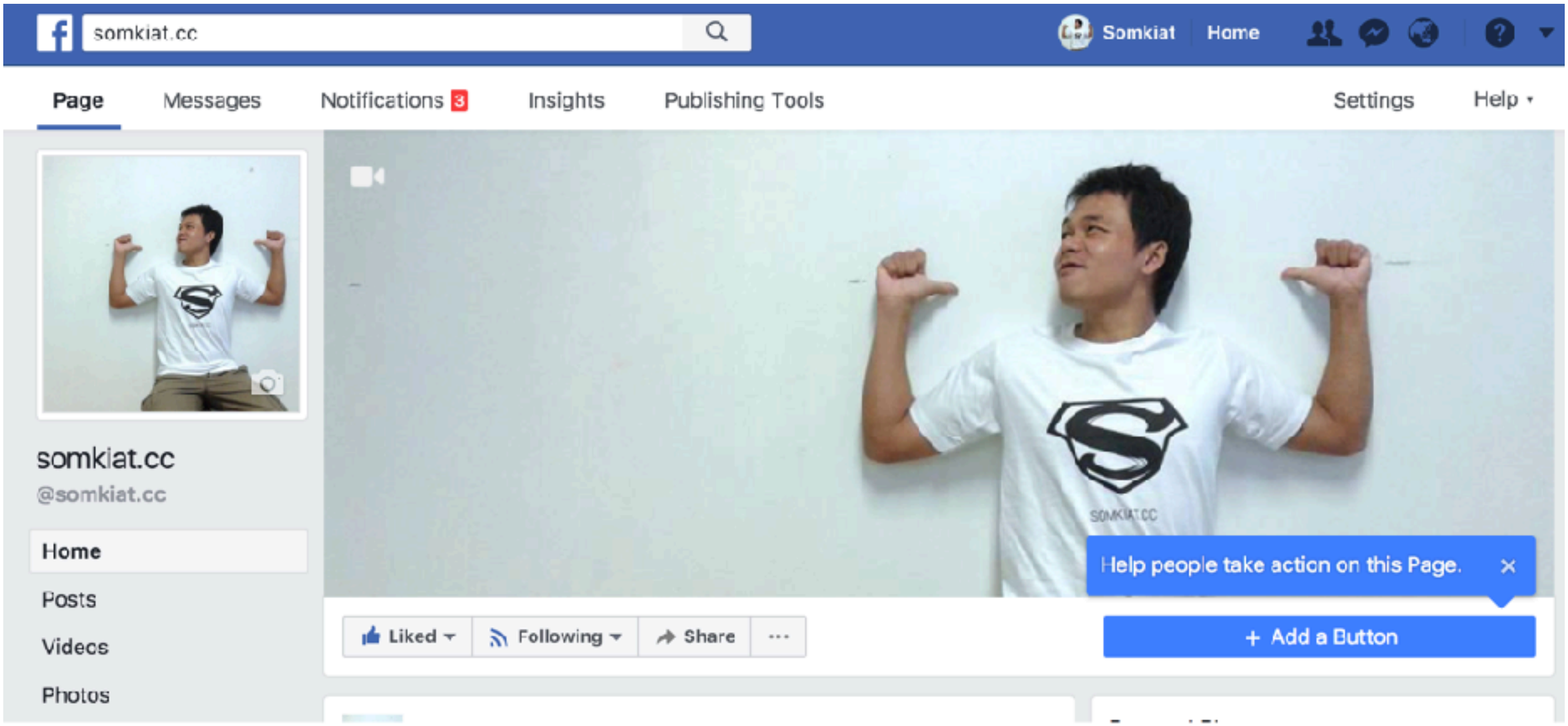
สยามชำนาญกิจ
SIAM CHAMNAN KIT

DevOps





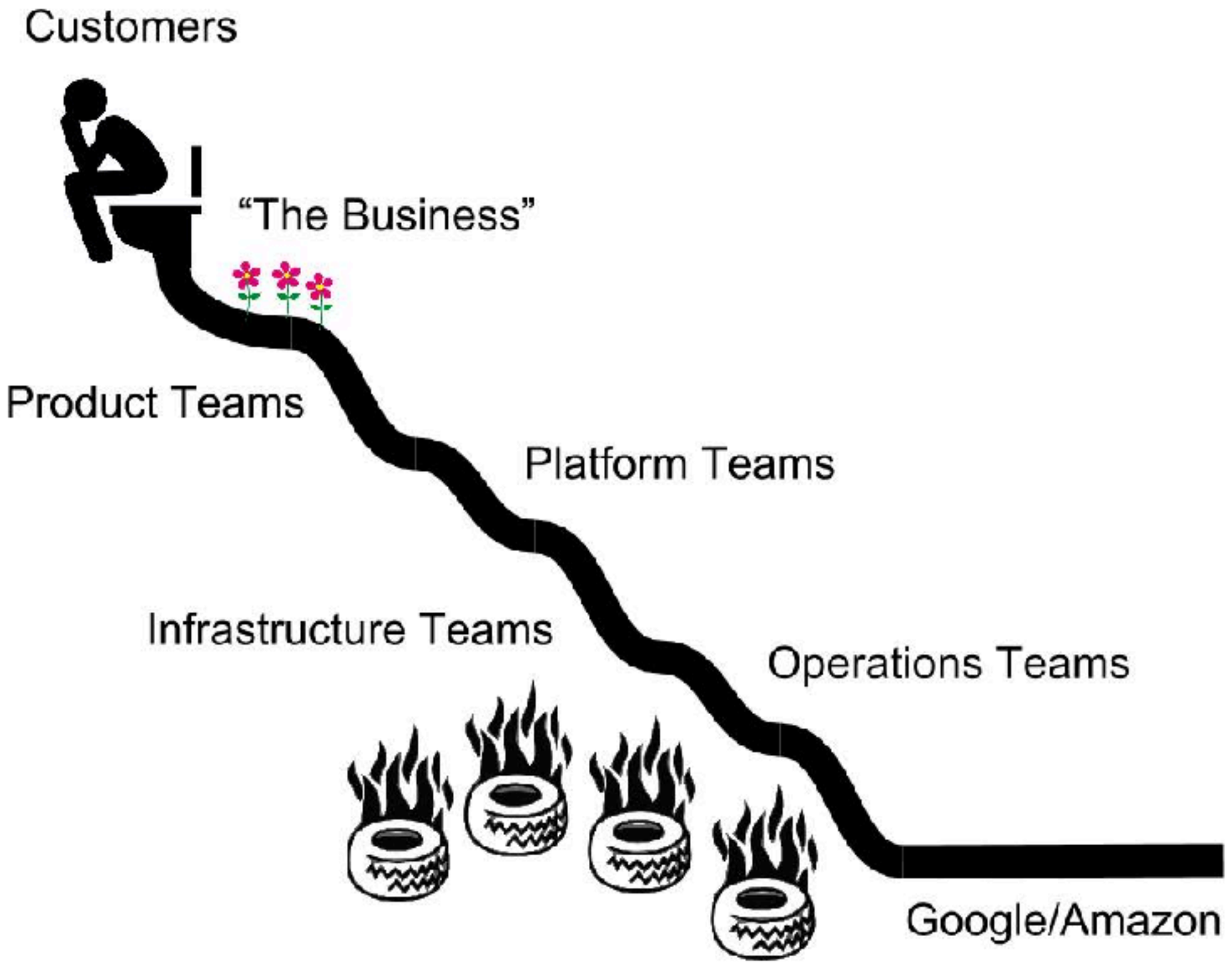
Agenda

- Cloud Native Application
- Overview of DevOps
- DevOps cultures
- Unify process between Dev and Ops
- Continuous Integration
- Continuous Delivery/Deployment
- Continuous Testing

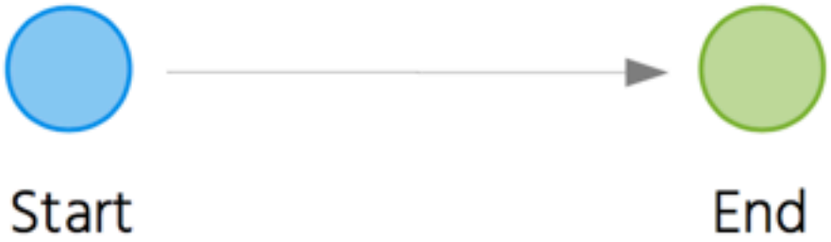
Agenda

- Configuration an management
- Release management
- Monitoring and feedback
- Unify process between Dev and Ops
- The Twelve Factors

Current Situation ?



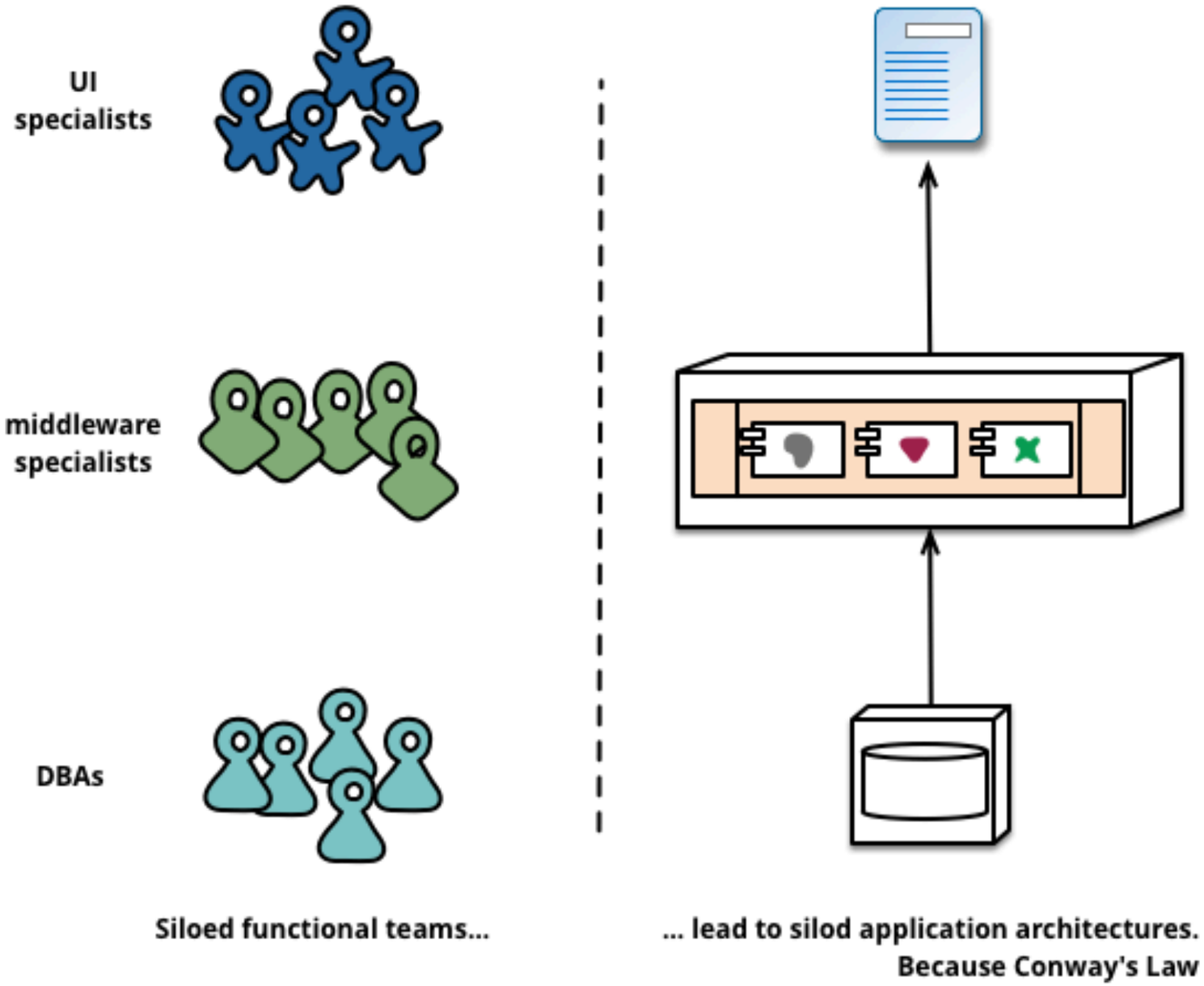
https://bravenewgeek.com/



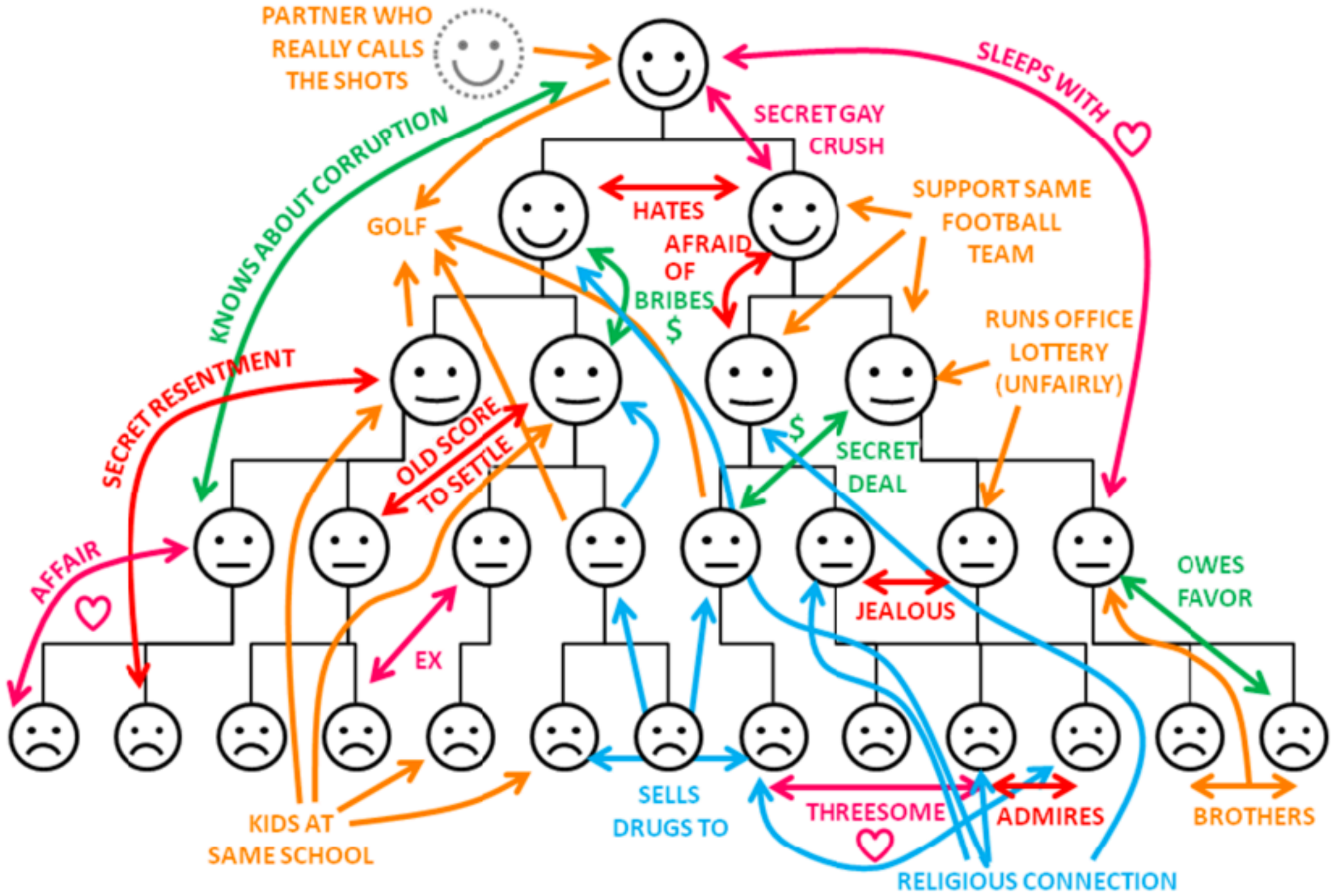
Theory

Reality

Conway’s Law



Real Organization Chart



Problem ?

Grow Fat

Code base grows. All the things slow down.

Age

Your code base will become a jurassic park introducing new tech becomes hard

Ownership

Who is responsible for which part and more important: who has the pager

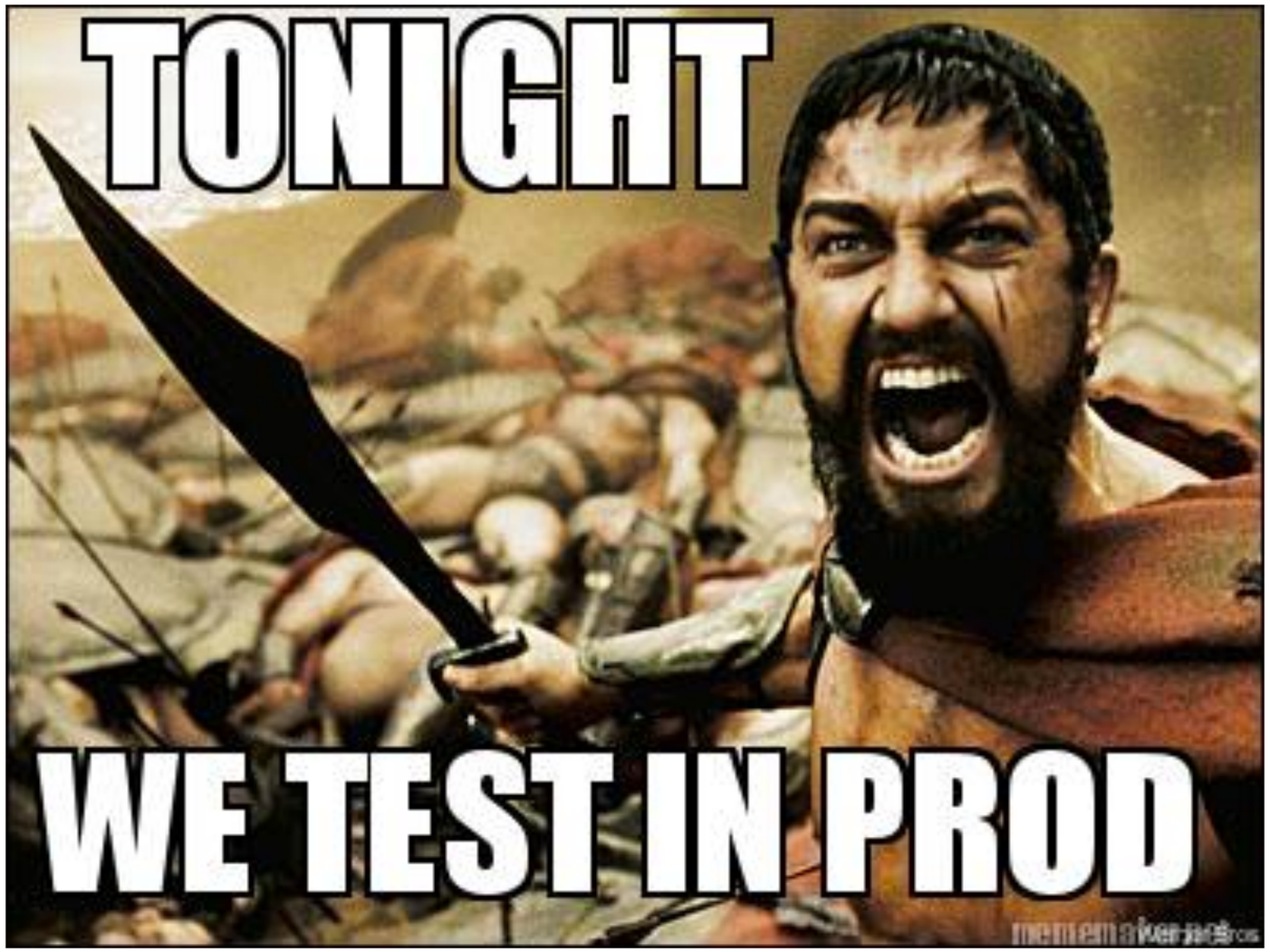
Economies of Scale

The bigger the team the more they interrupt each other

https://trello.com/

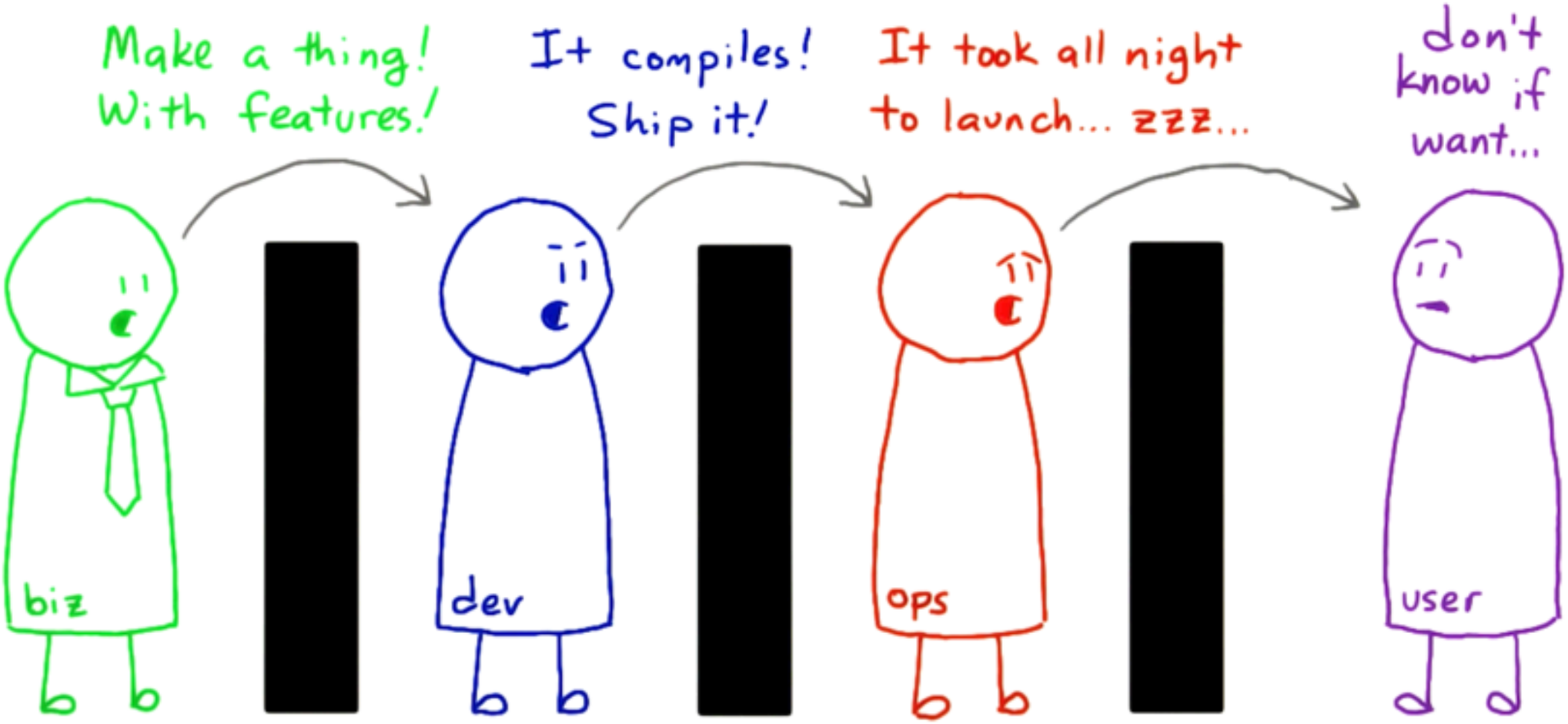
Problem ?

Problem ?



Problem ?

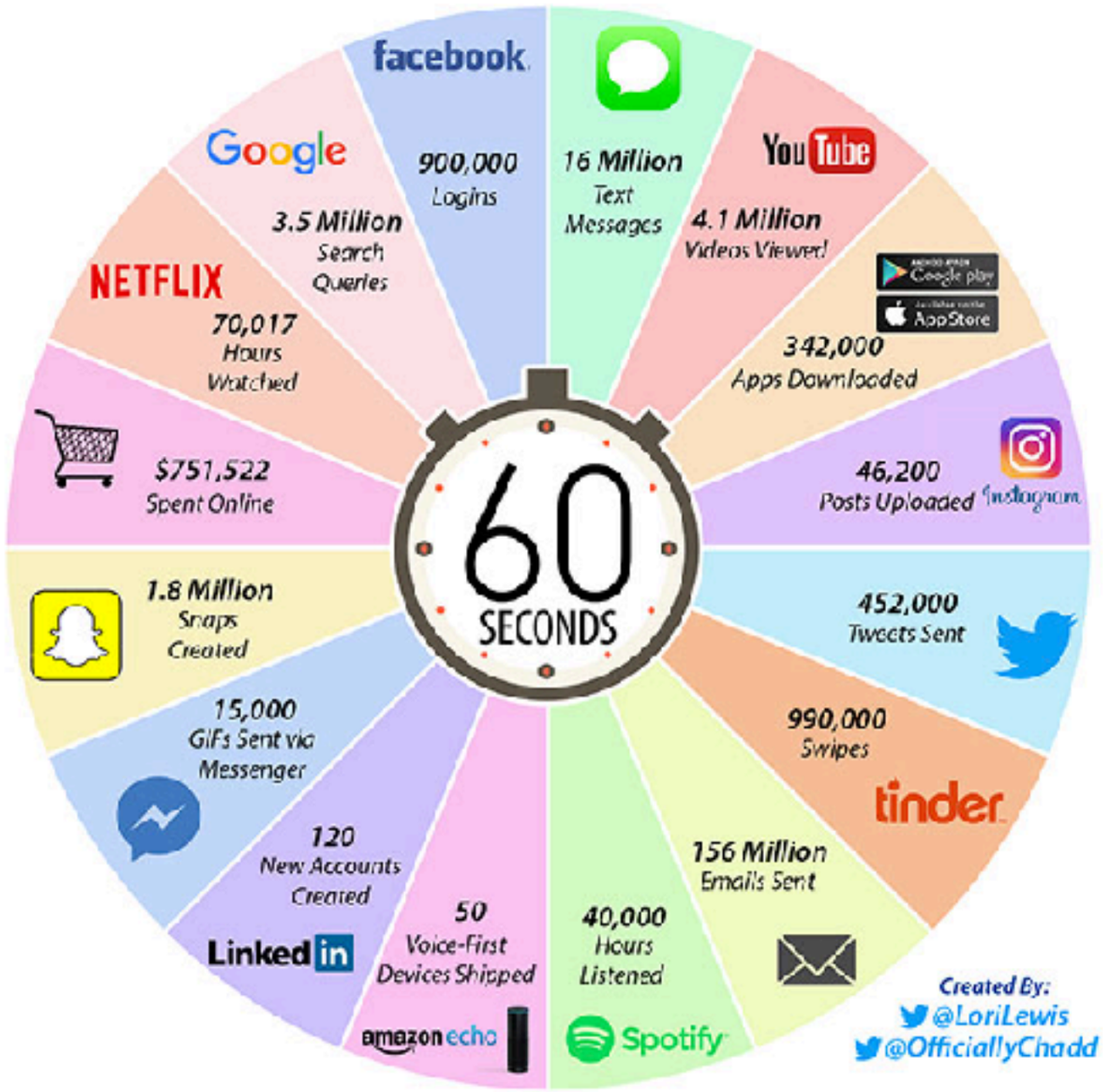
Problem ?



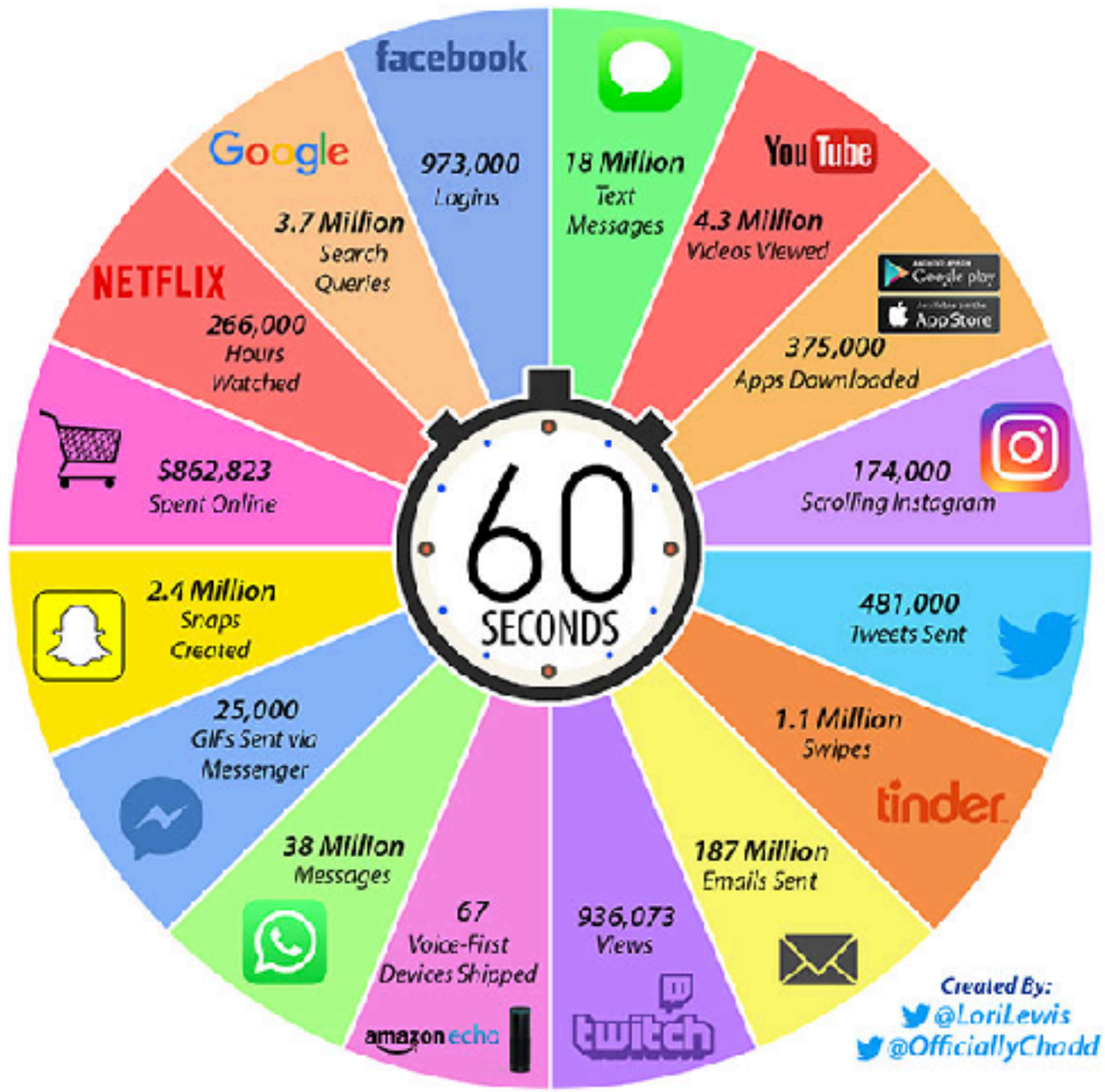
What happens in an Internet minute ?

2018 This Is What Happens In An Internet Minute

2017 This Is What Happens In An Internet Minute

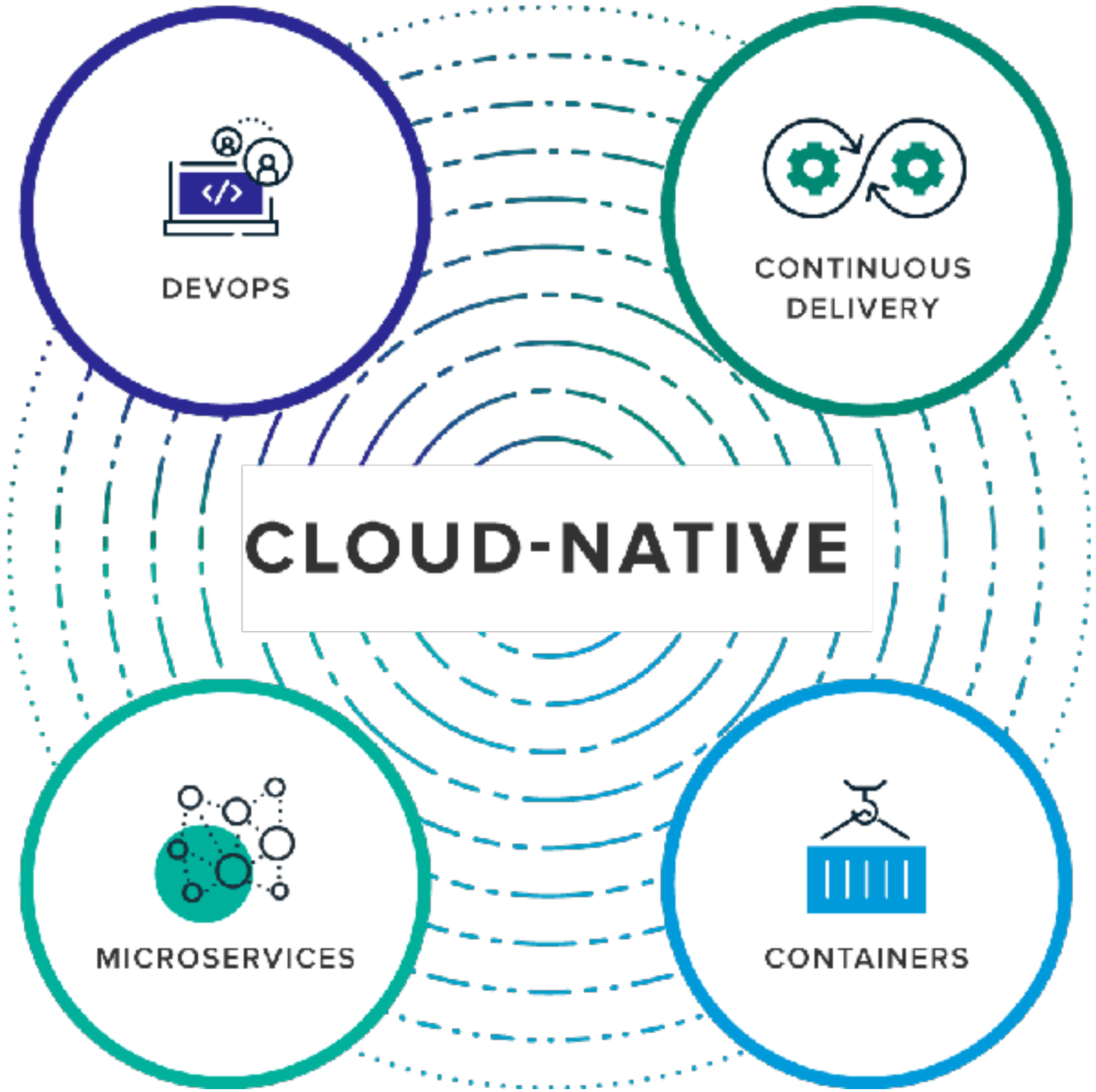


2018 This Is What Happens In An Internet Minute



$$Trust = \frac{Consistency}{Time}$$

We need Change

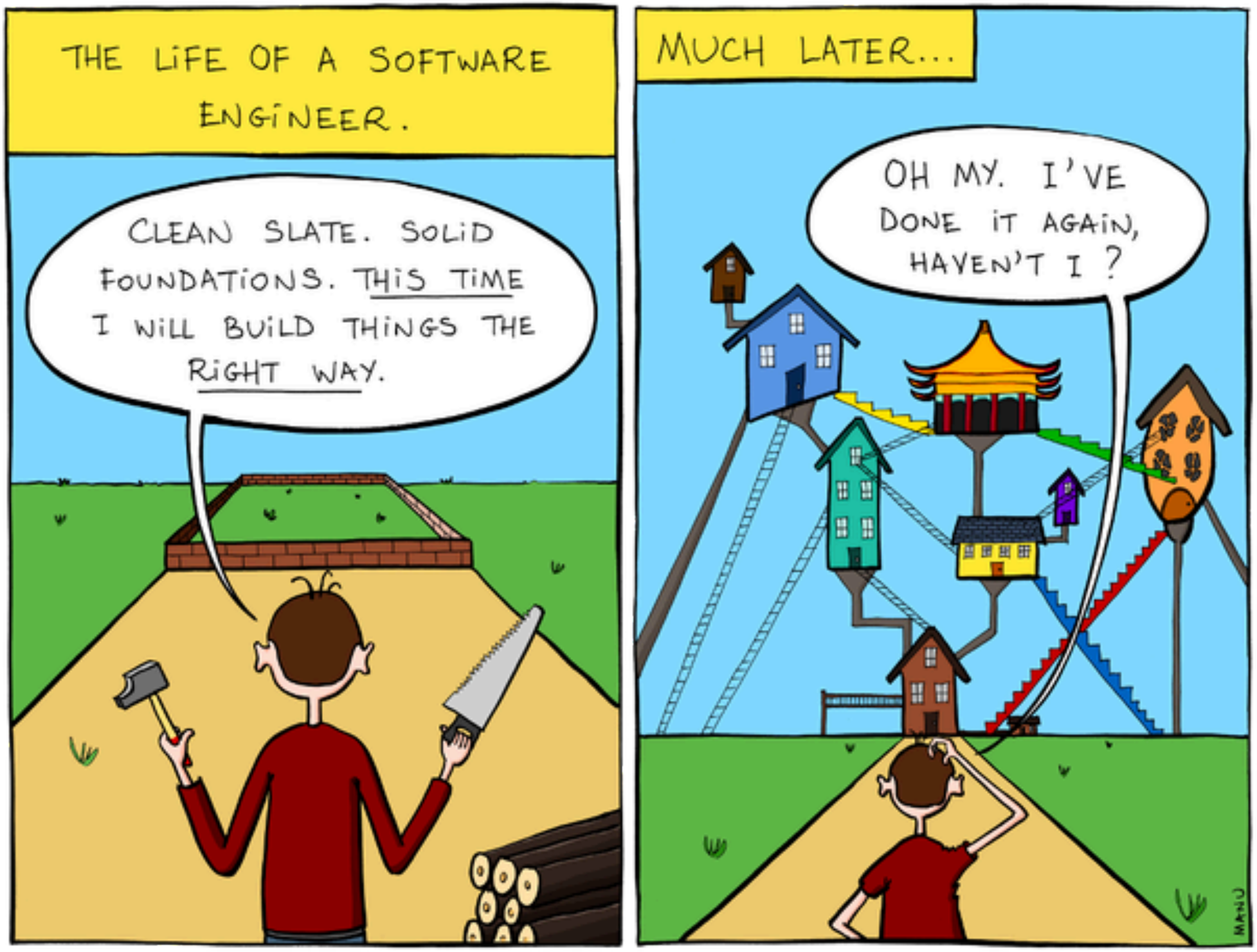


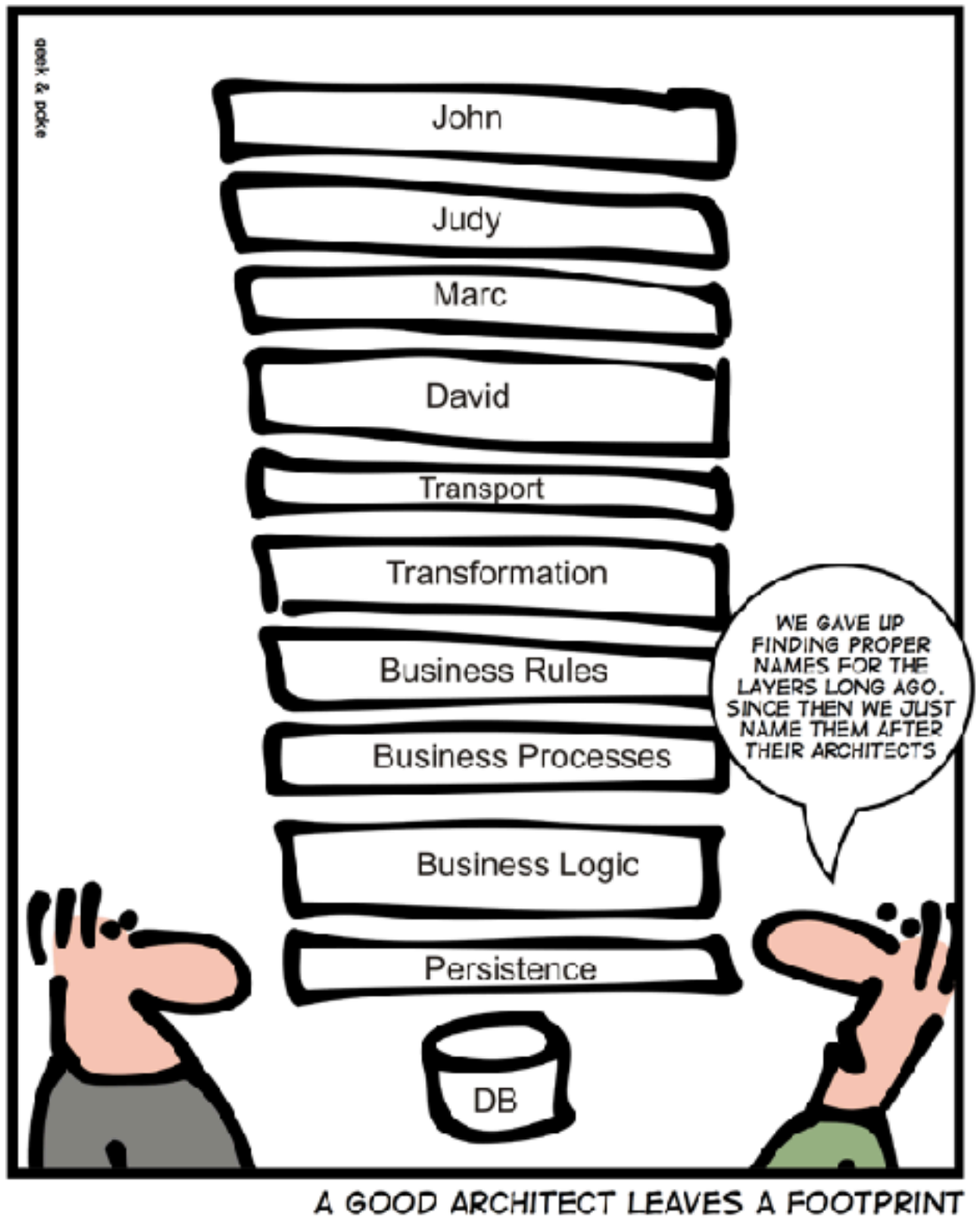
https://pivotal.io/cloud-native

Let’s start with basic

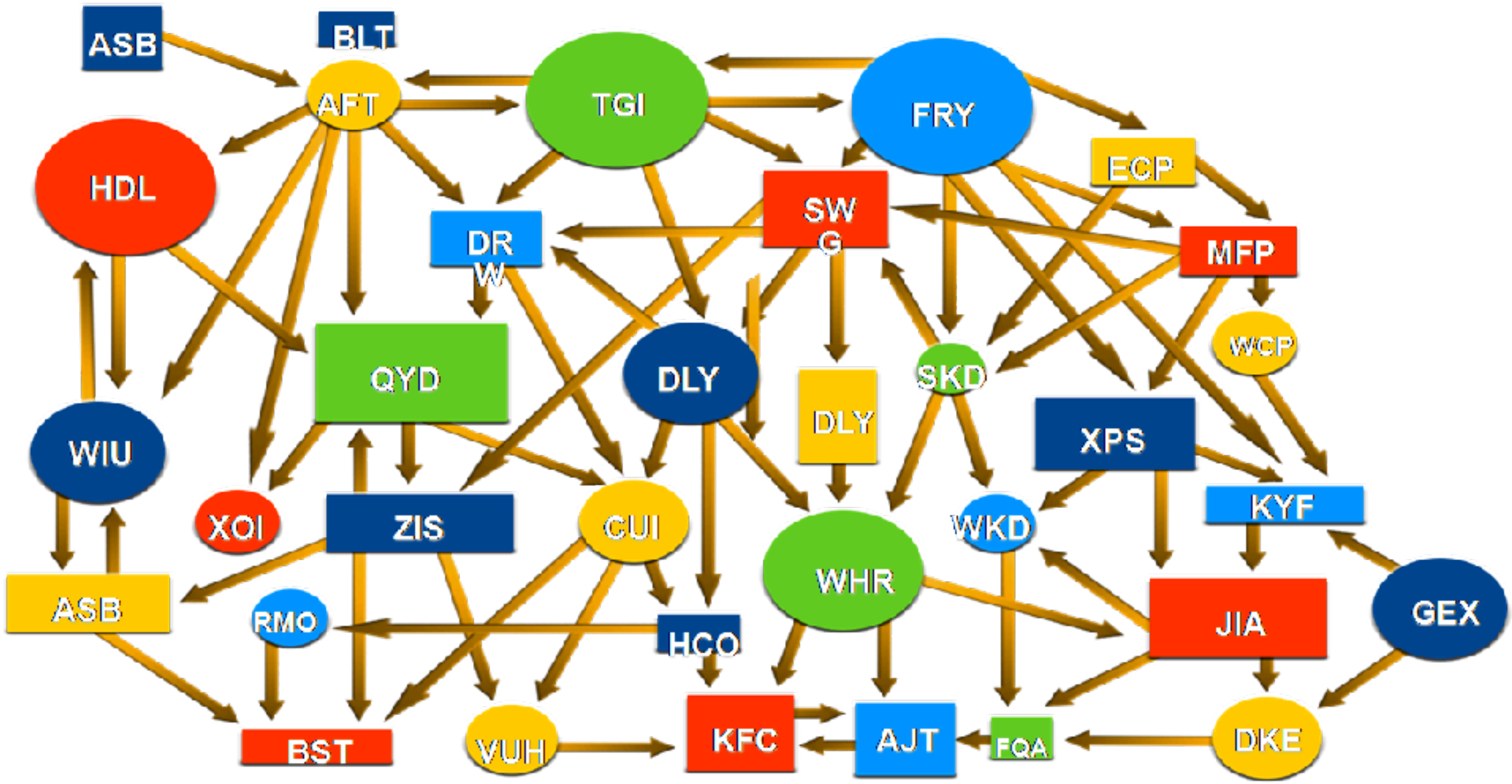
How to develop ?

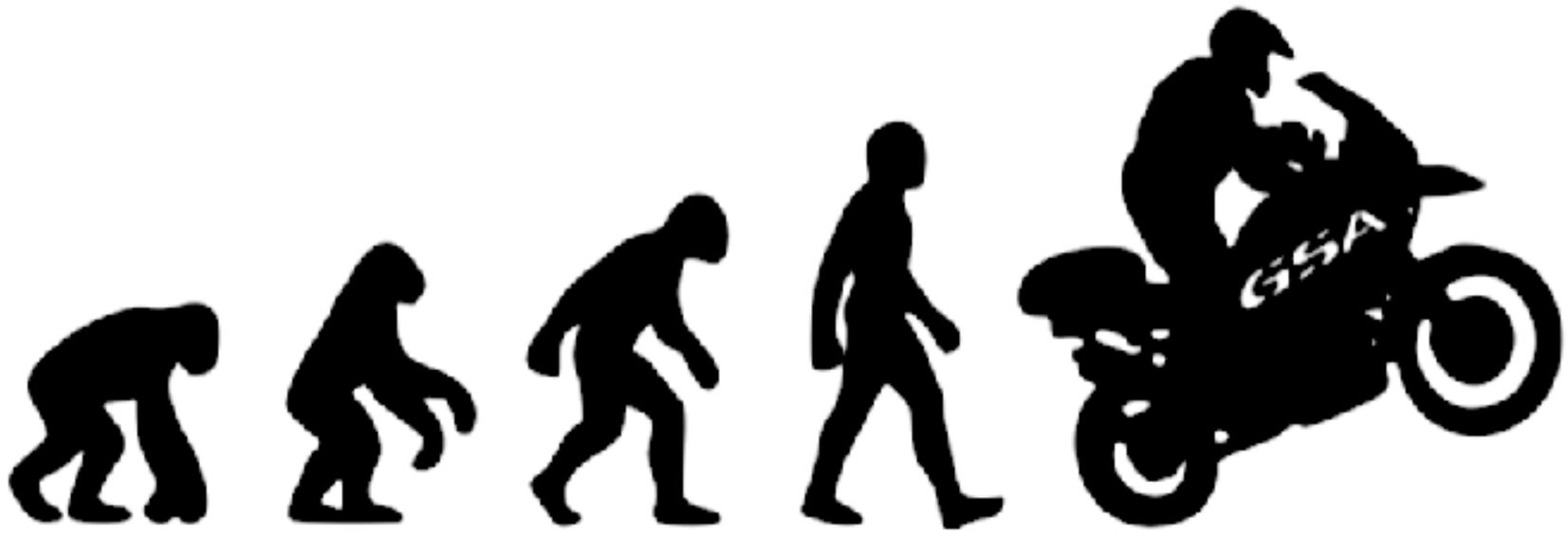
Good Architecture ?





A GOOD ARCHITECT LEAVES A FOOTPRINT

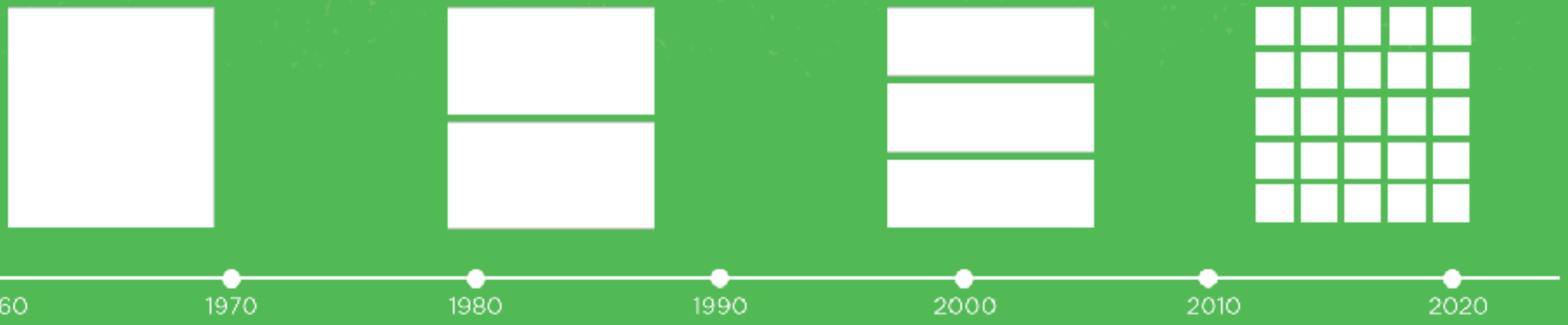




monolith n-tier service oriented microservices serverless

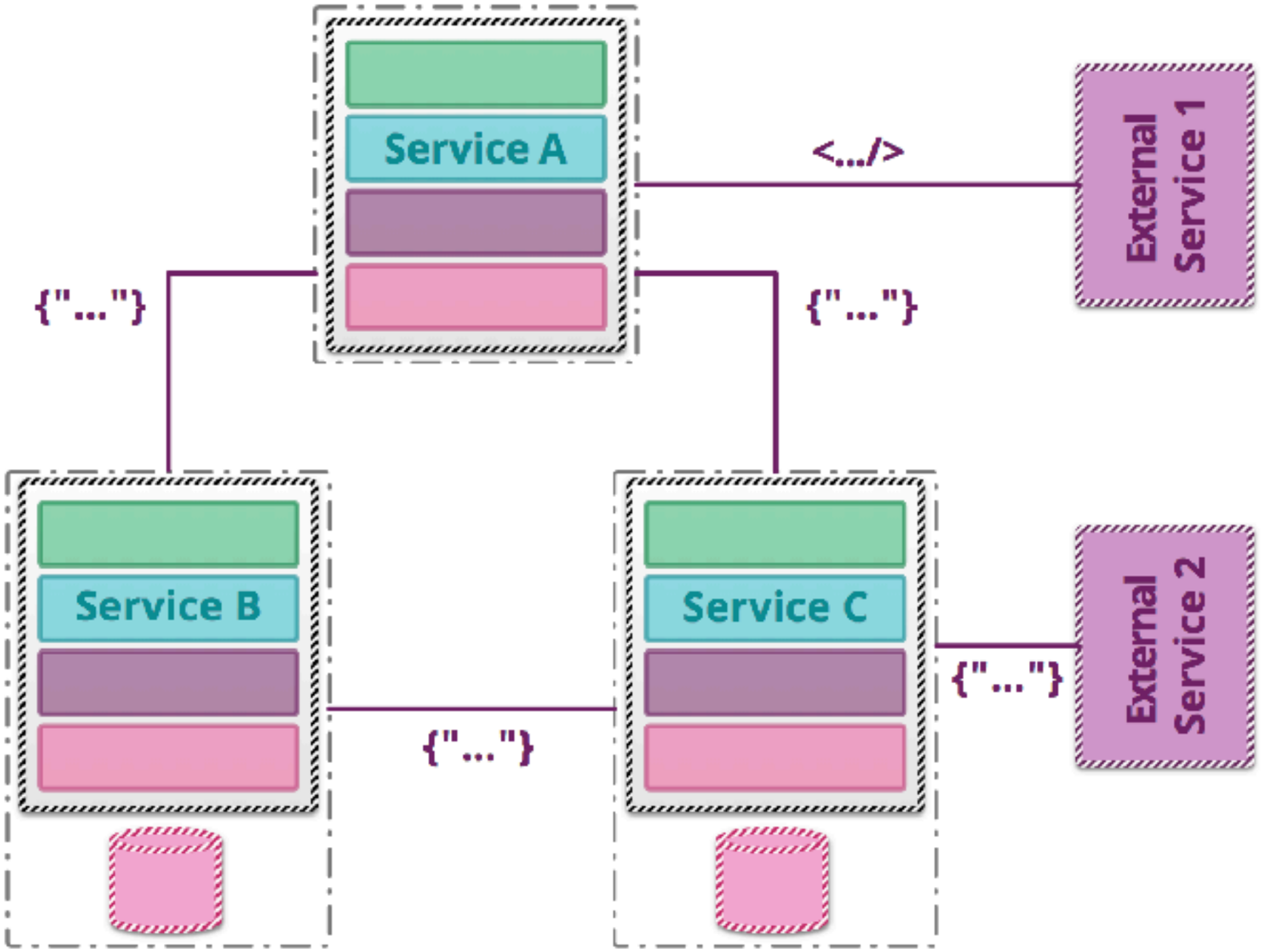
Evolution of Architecture

ABSTRACTION IN THE ENTERPRISE



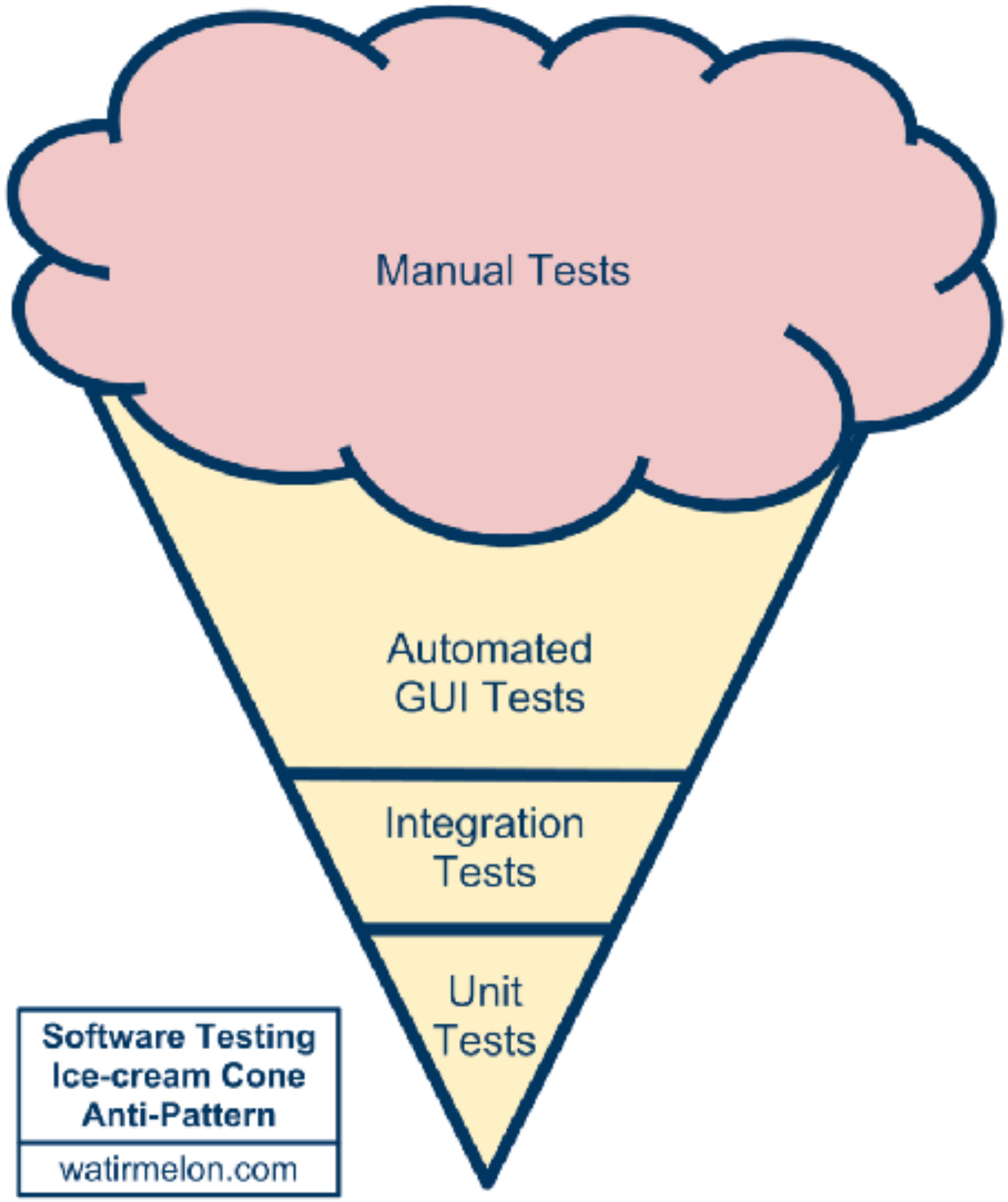
How to Test ?

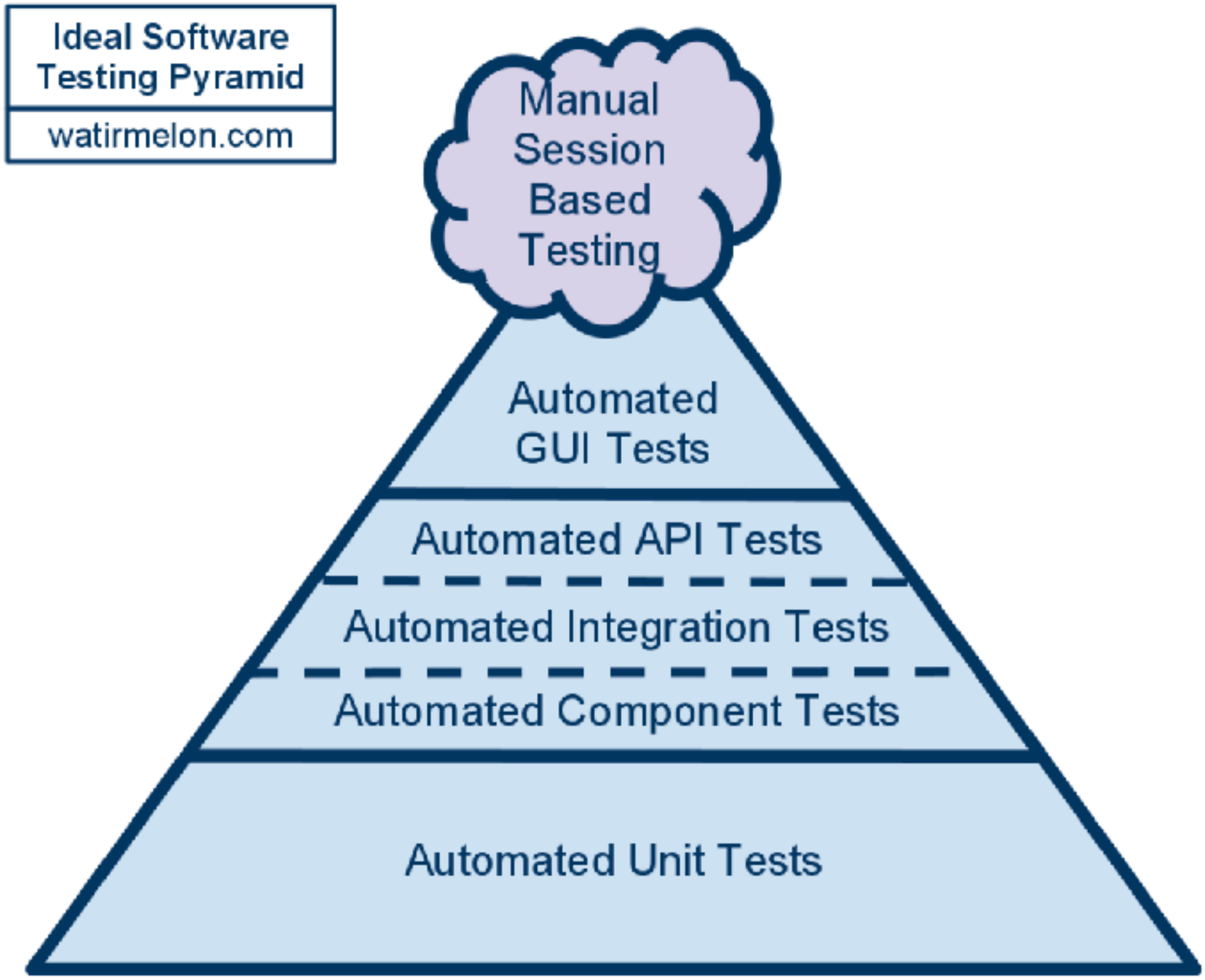
Multiple services

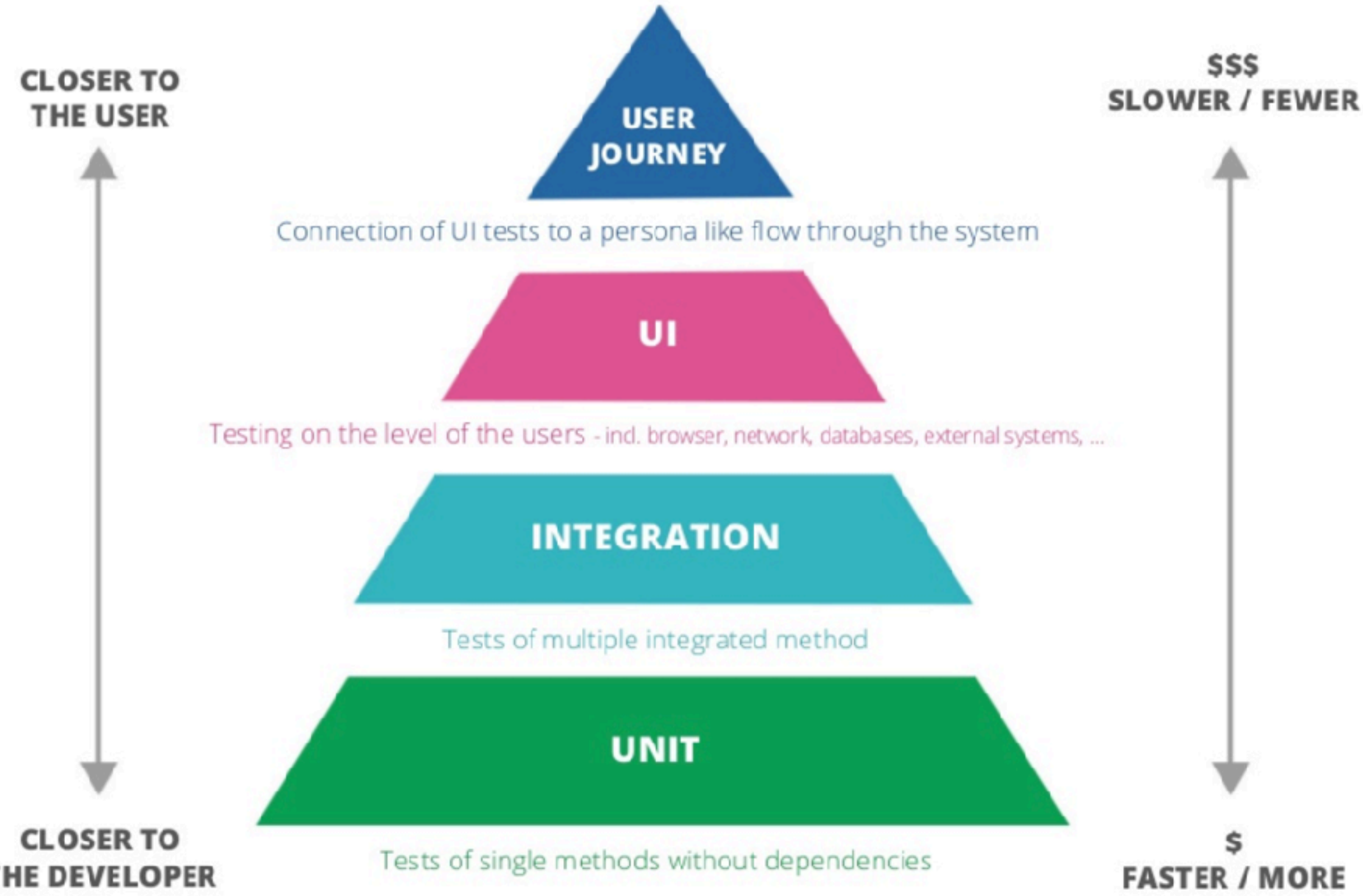


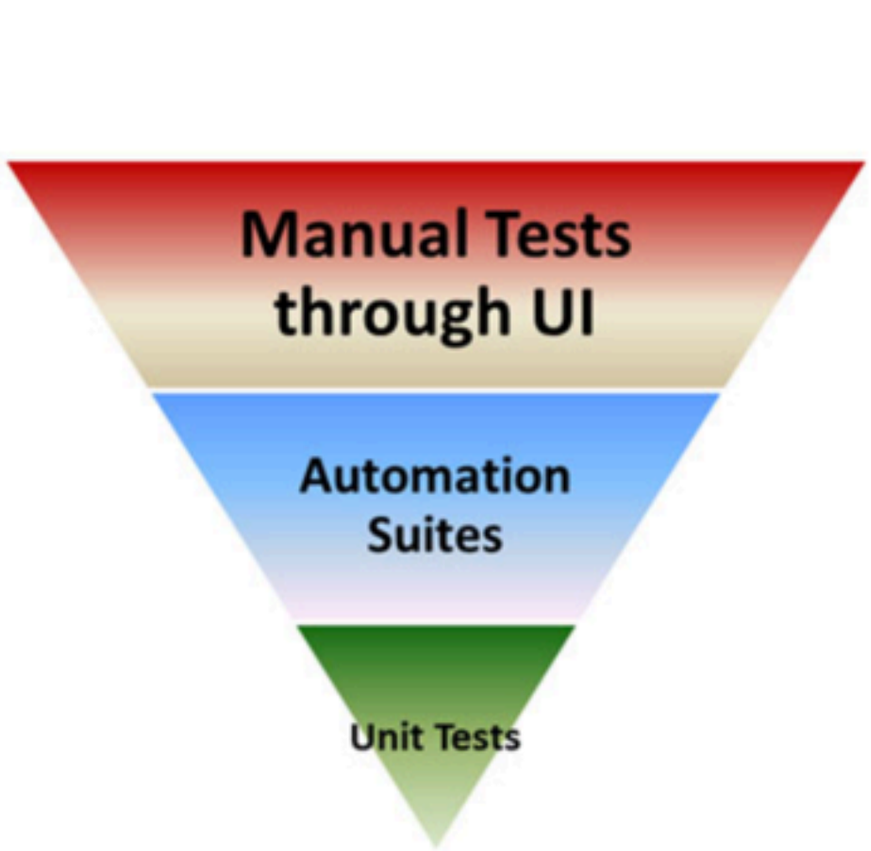
https://martinfowler.com/articles/microservice-testing

Testing

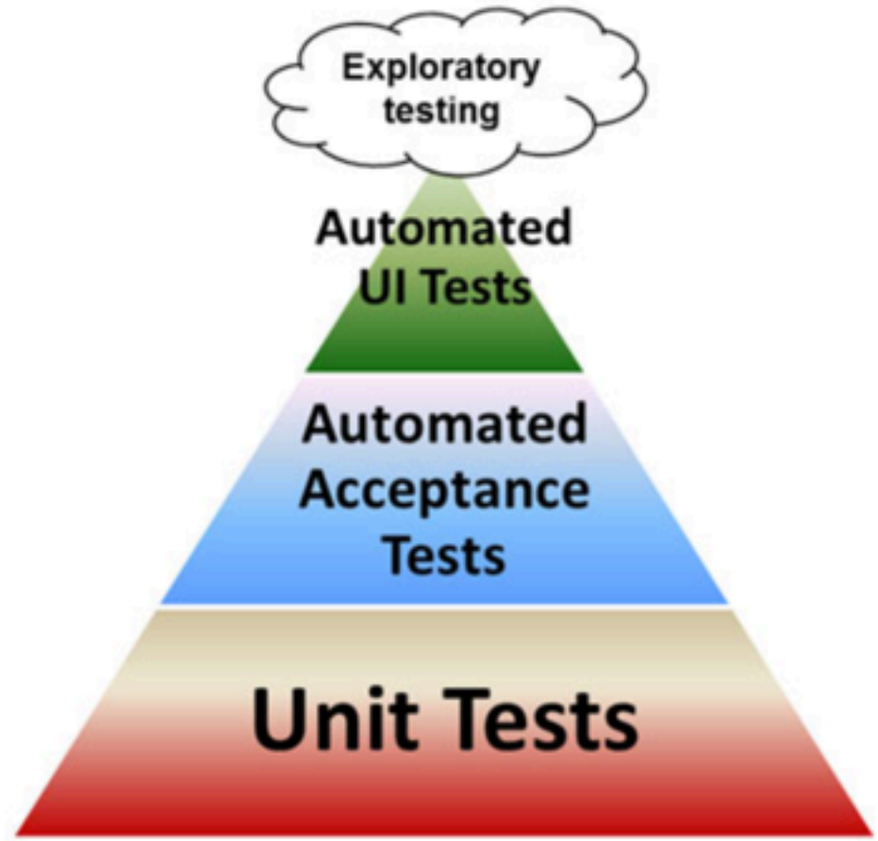




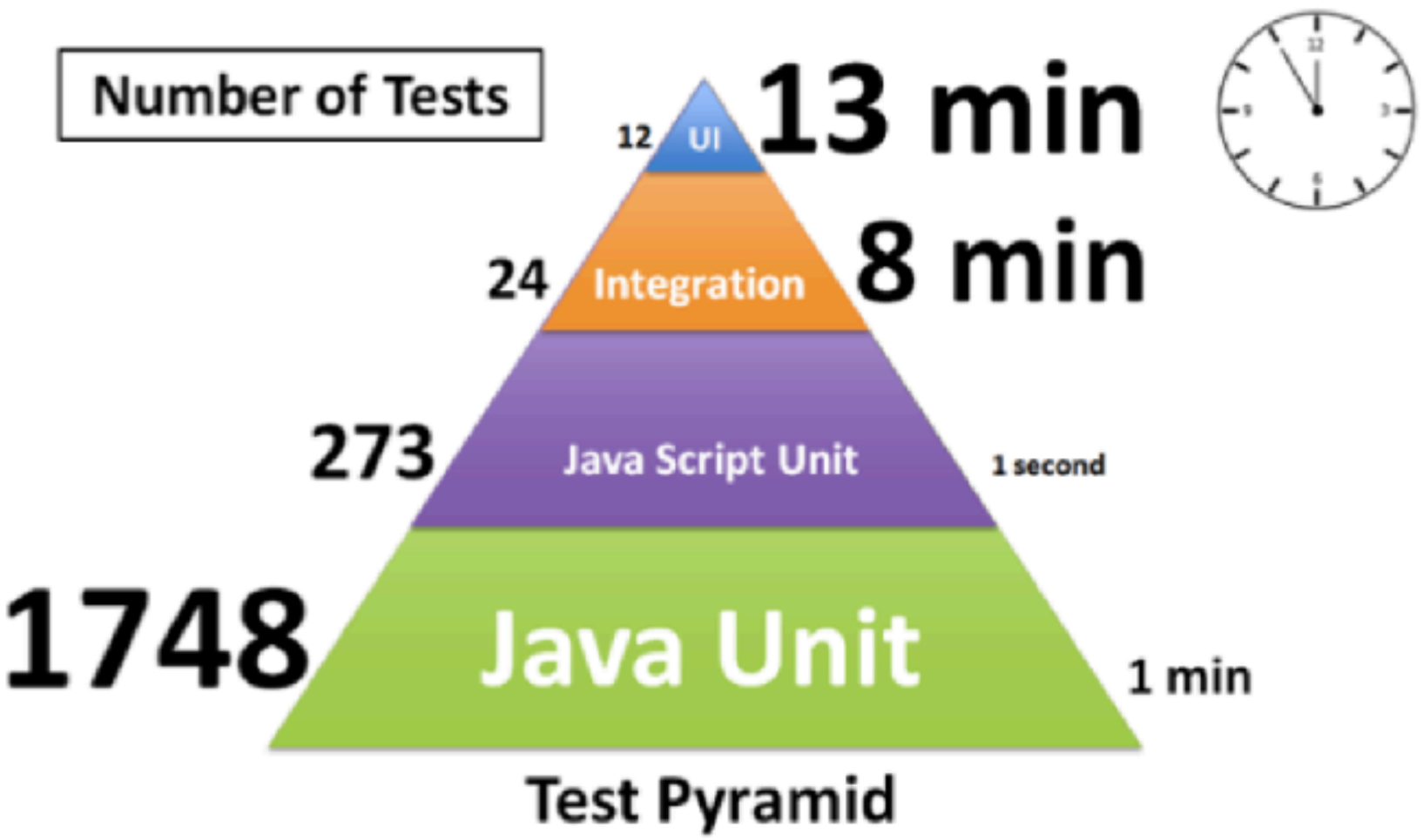




Traditional
(find bugs)

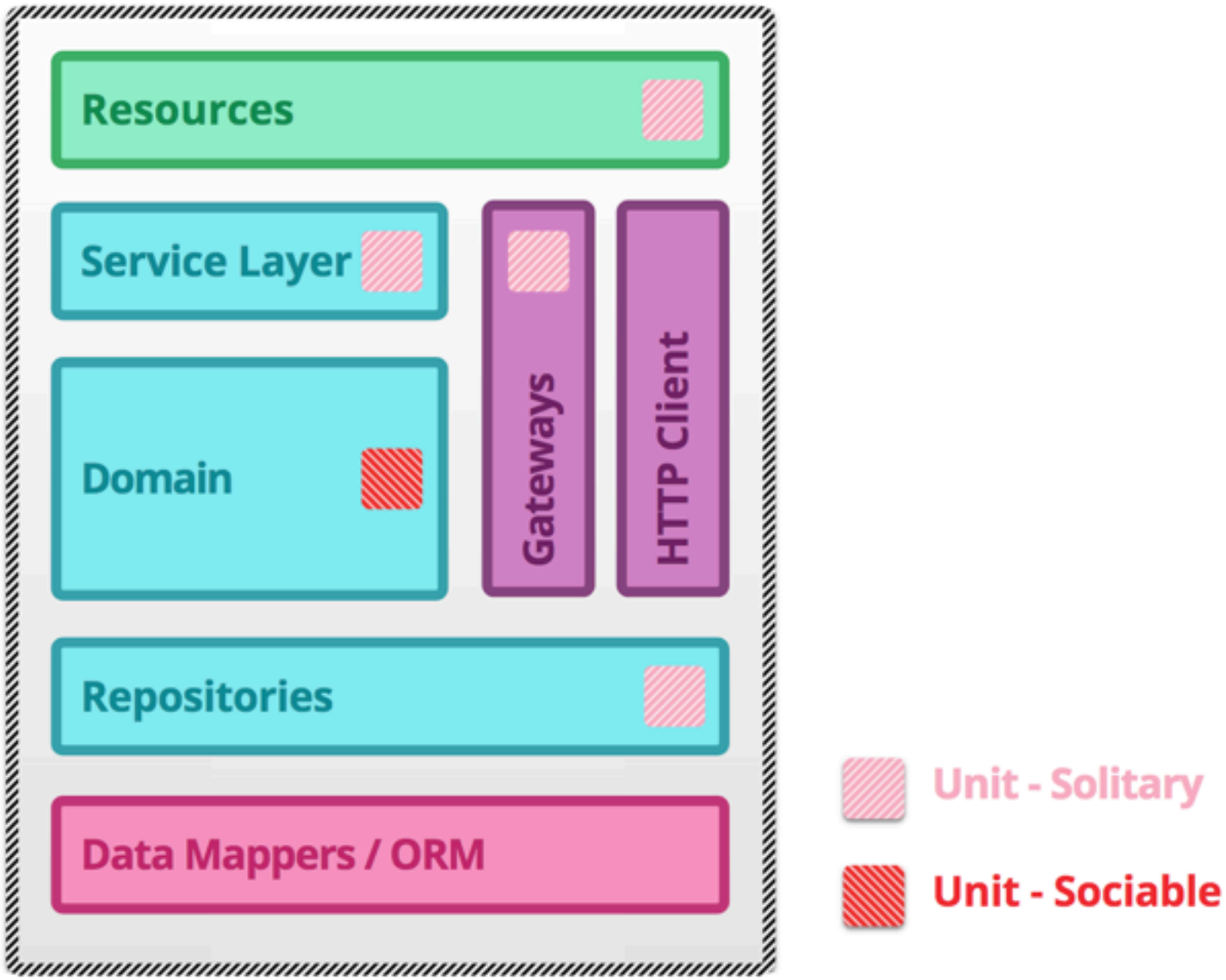


Agile
(prevent bugs)

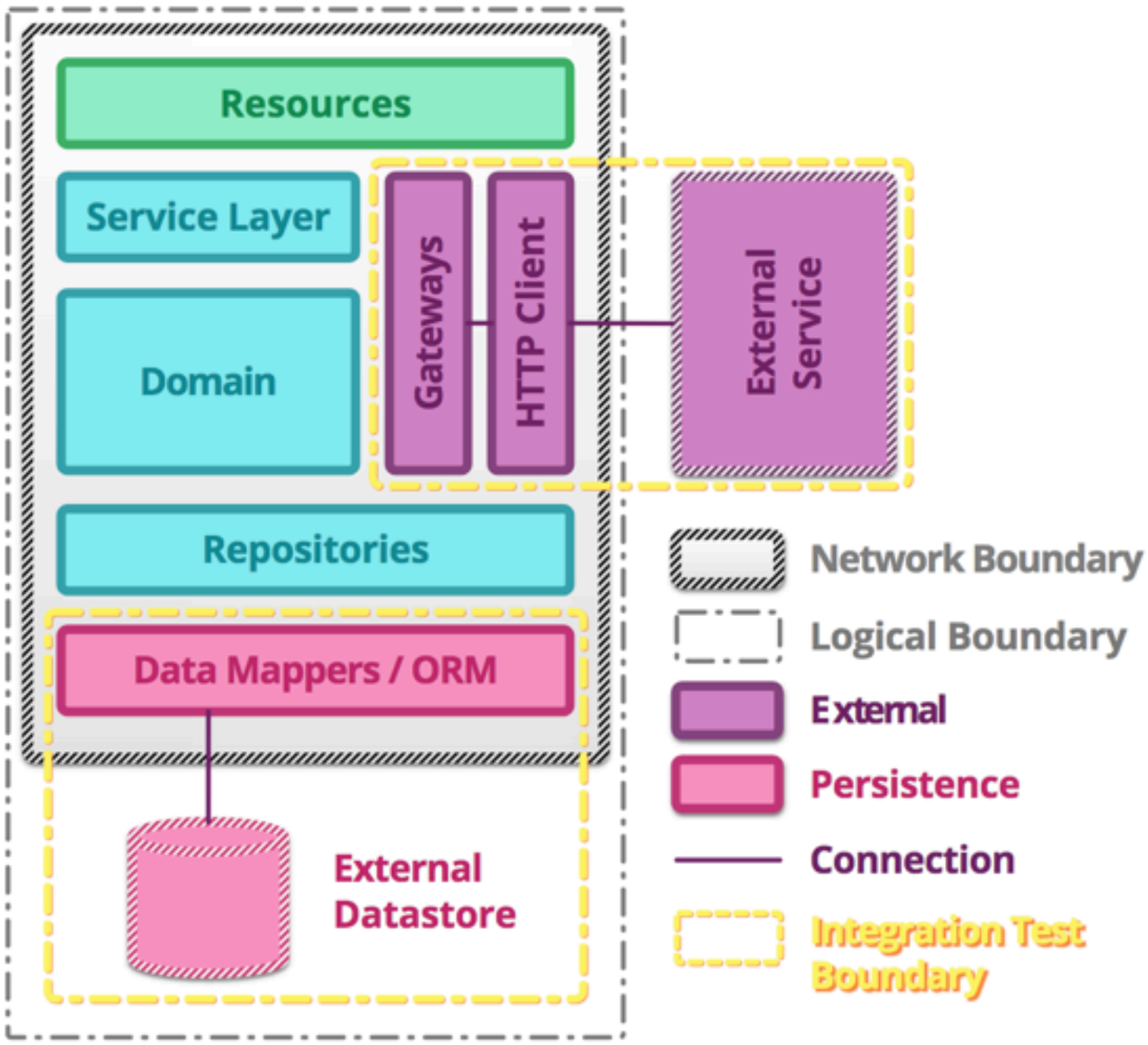


Test Pyramid

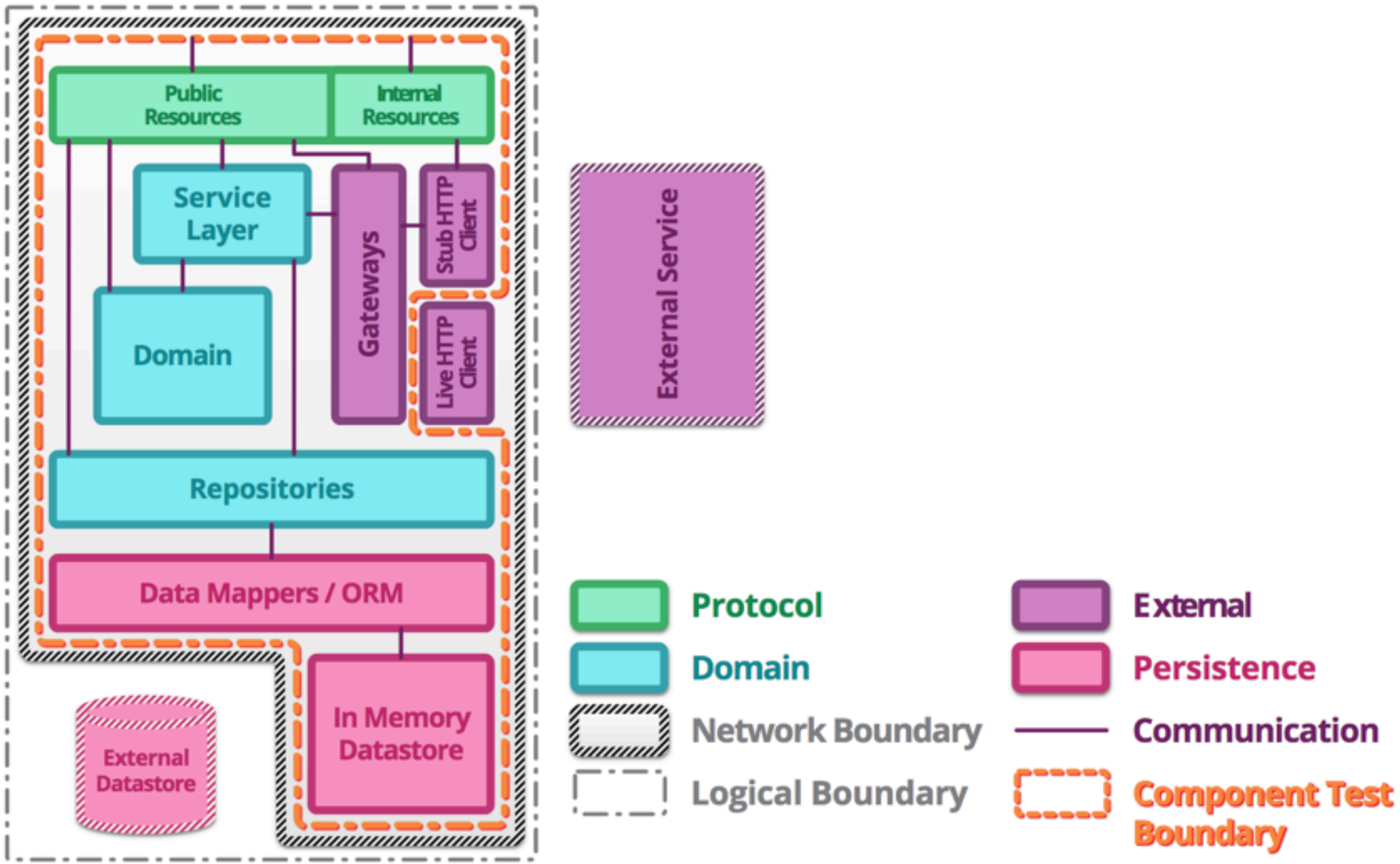
Unit testing



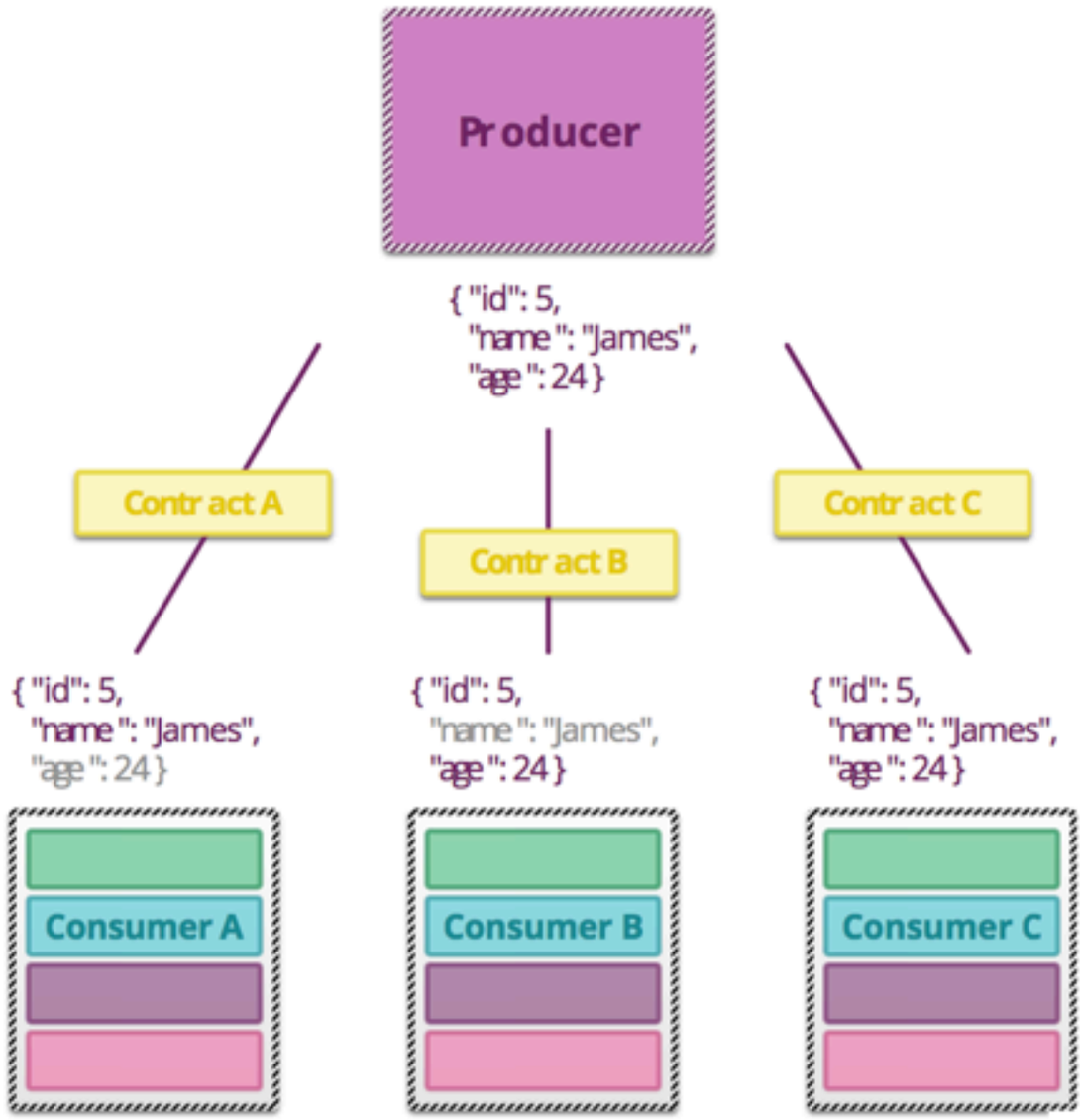
Integration testing



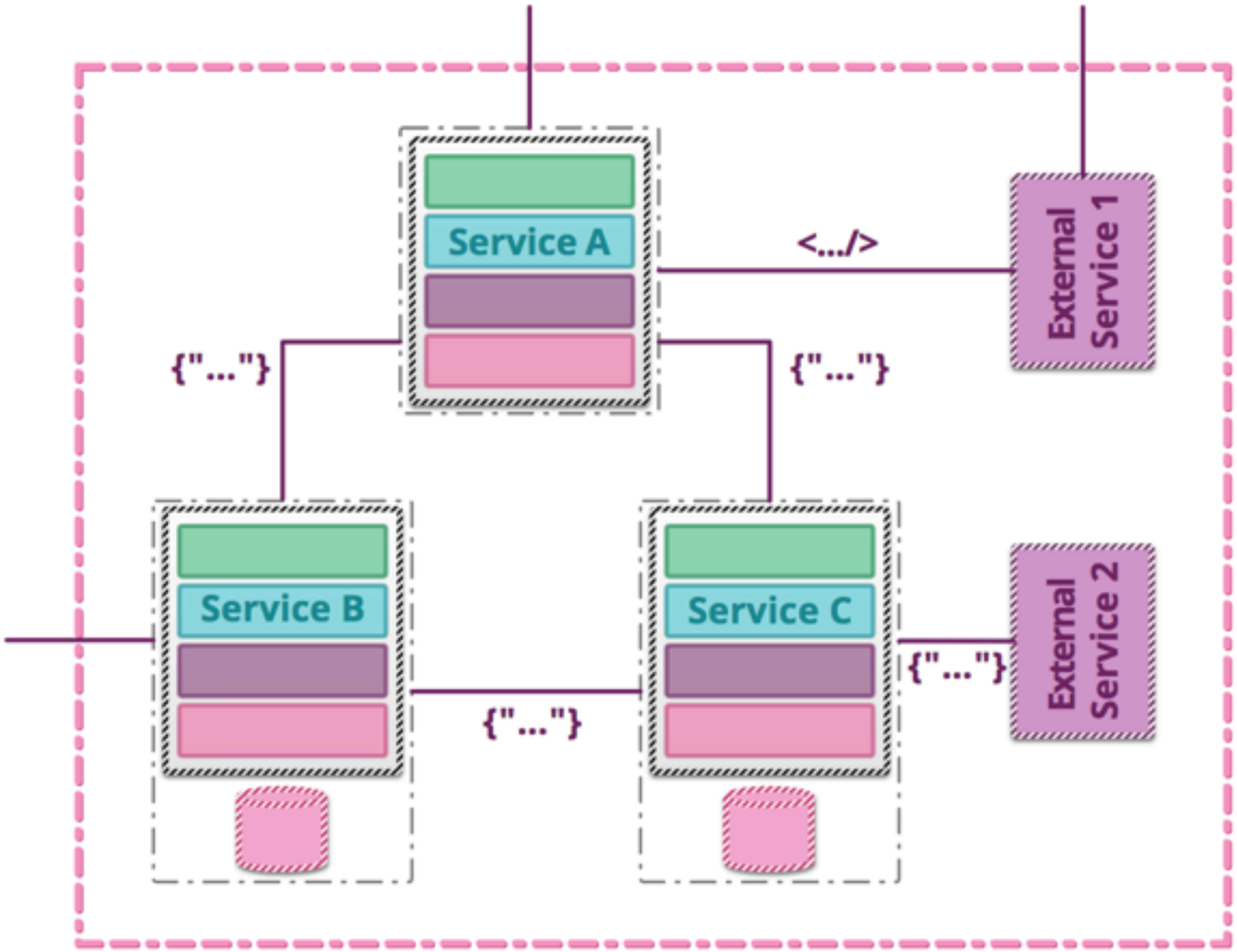
Component testing



Contract testing



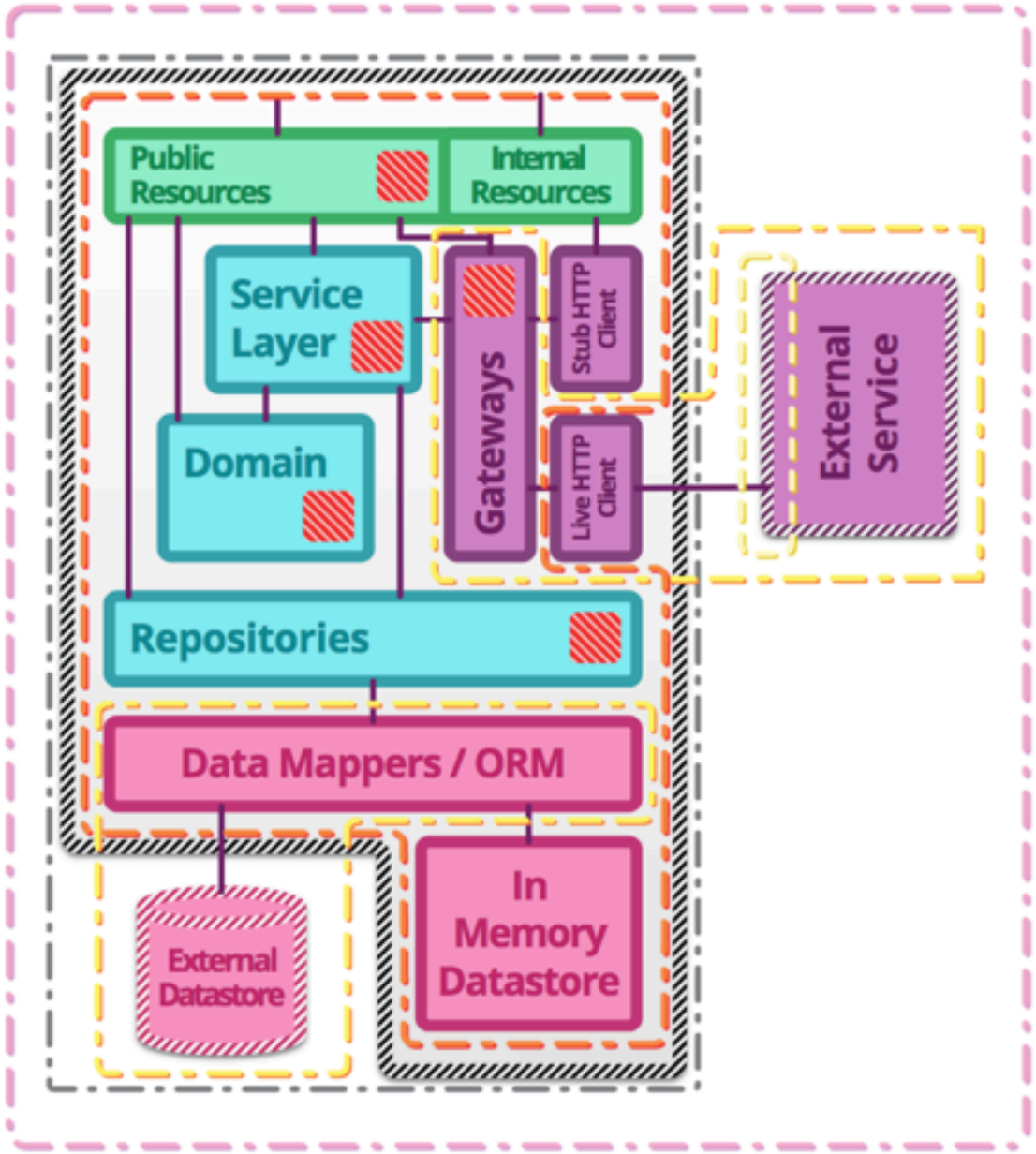
End-to-End testing



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 ?

more ...

Performance testing ?

Security testing ?

How to Deploy ?

Deployment Smells

Requires a ticket

A ticket for the deployment team

Only one person

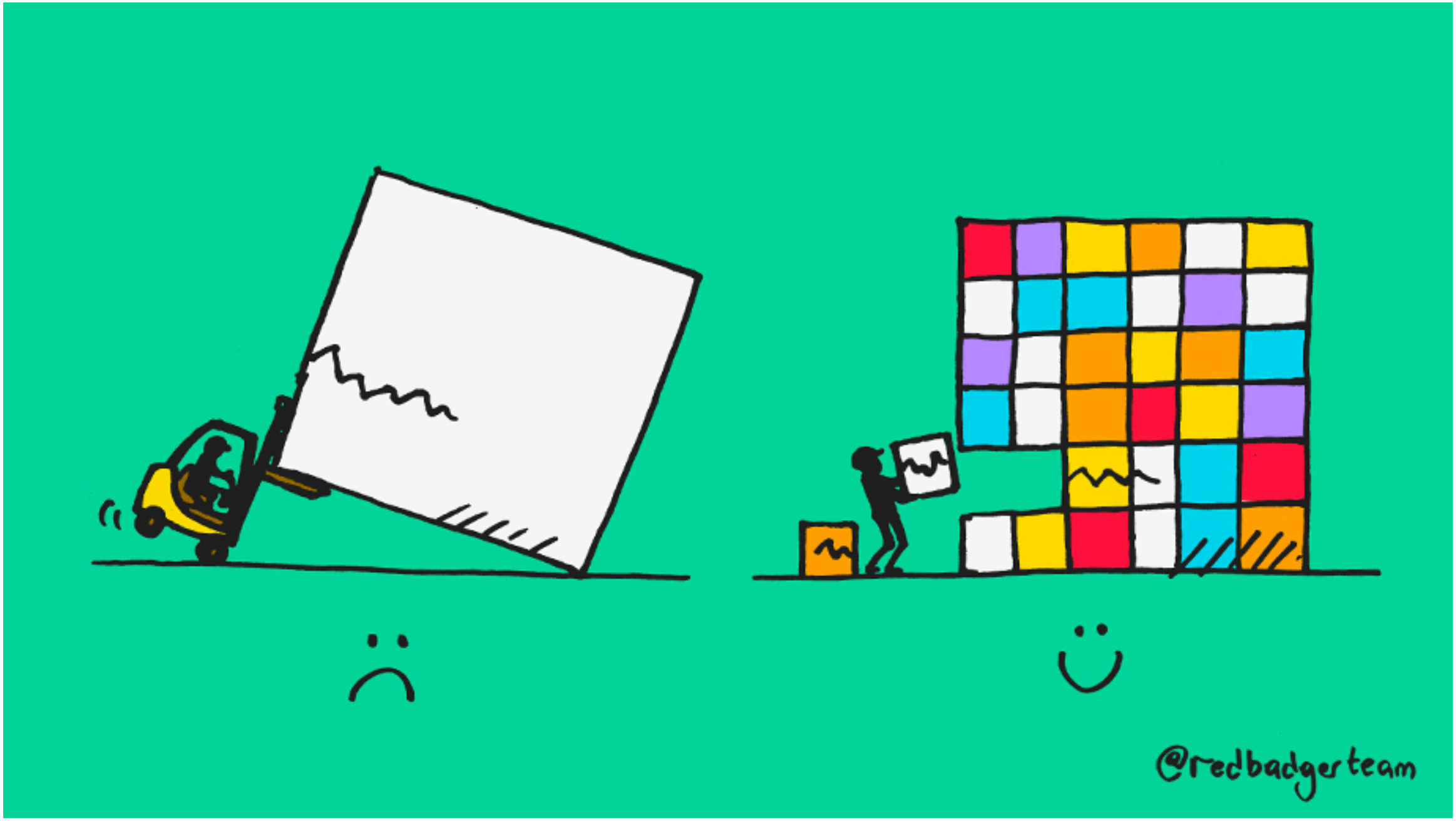
There is only one person in the team that owns it

Takes more then 15 mins

Setting it up should be quick and initial deployment should quick

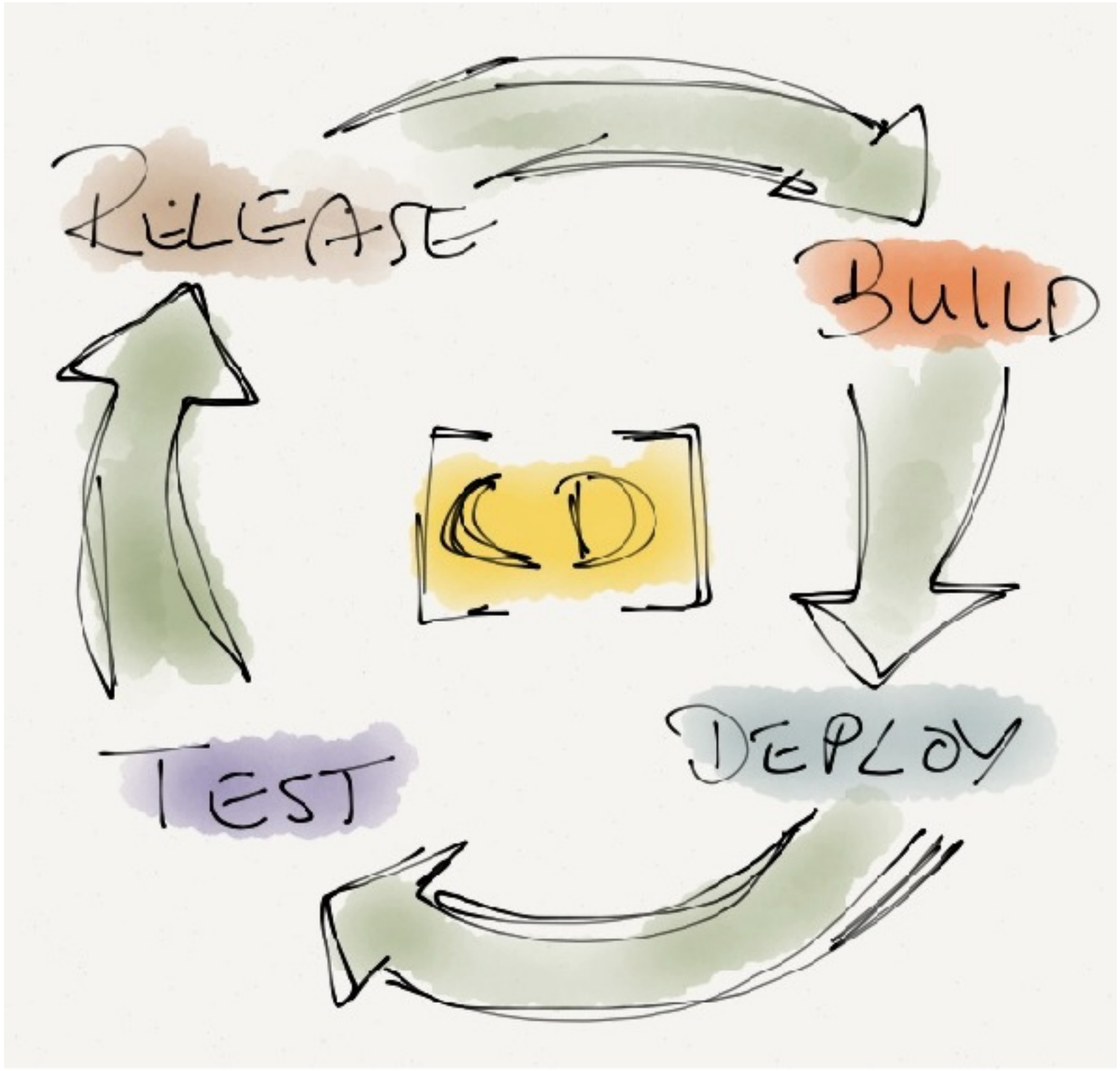
https://trello.com/

Deployment



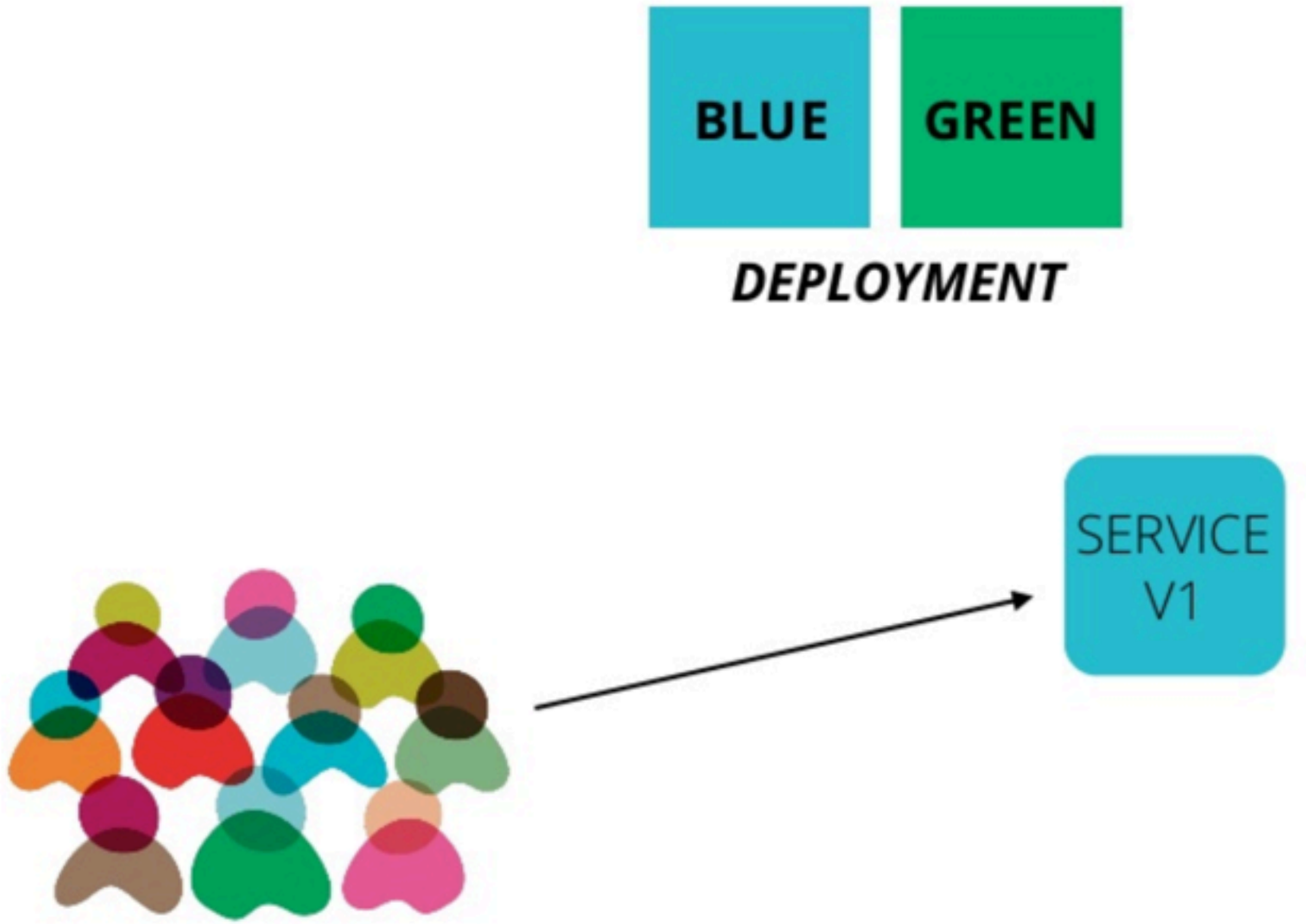
Deploy vs Release

Deploy vs Release

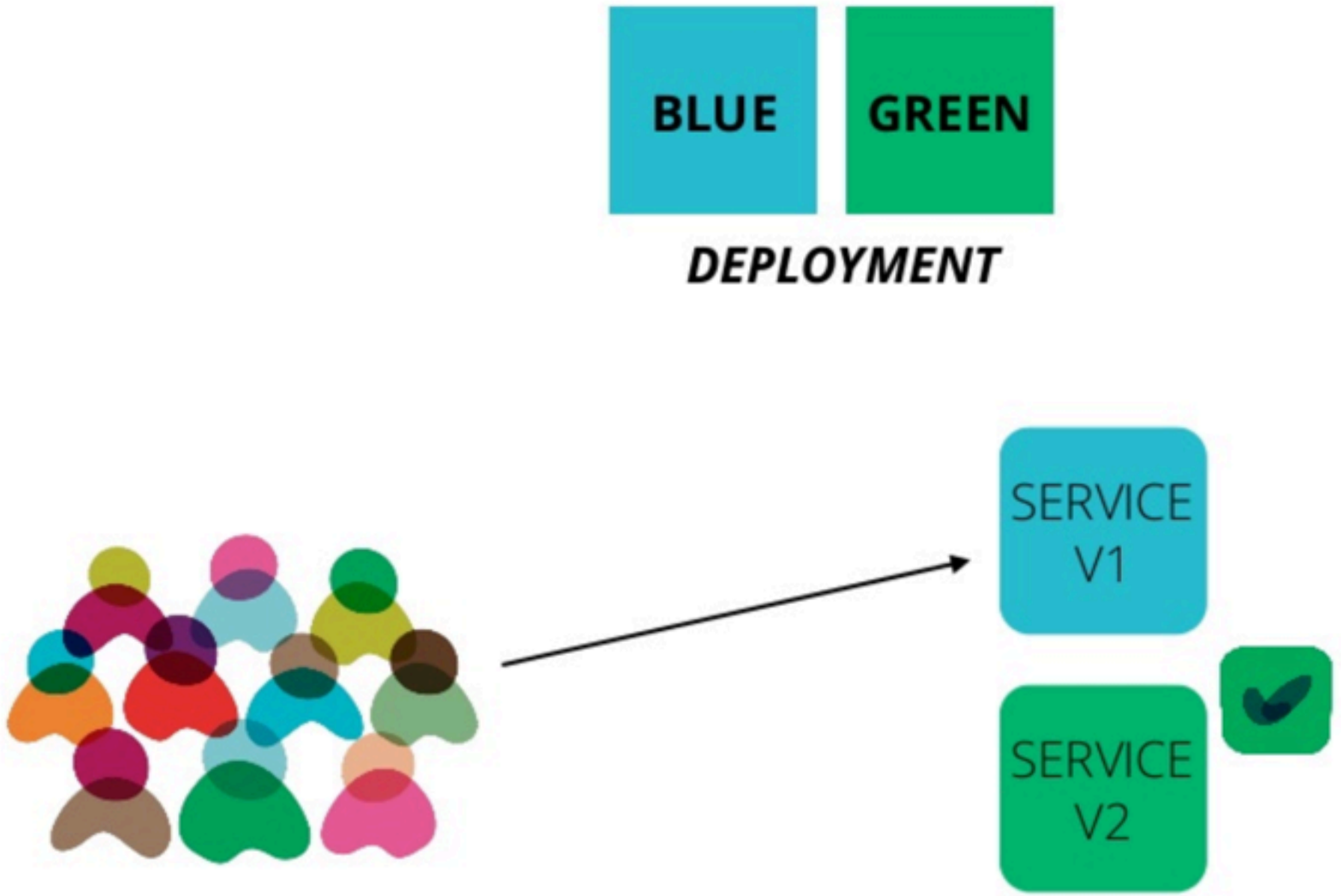


Blue Green Deployment

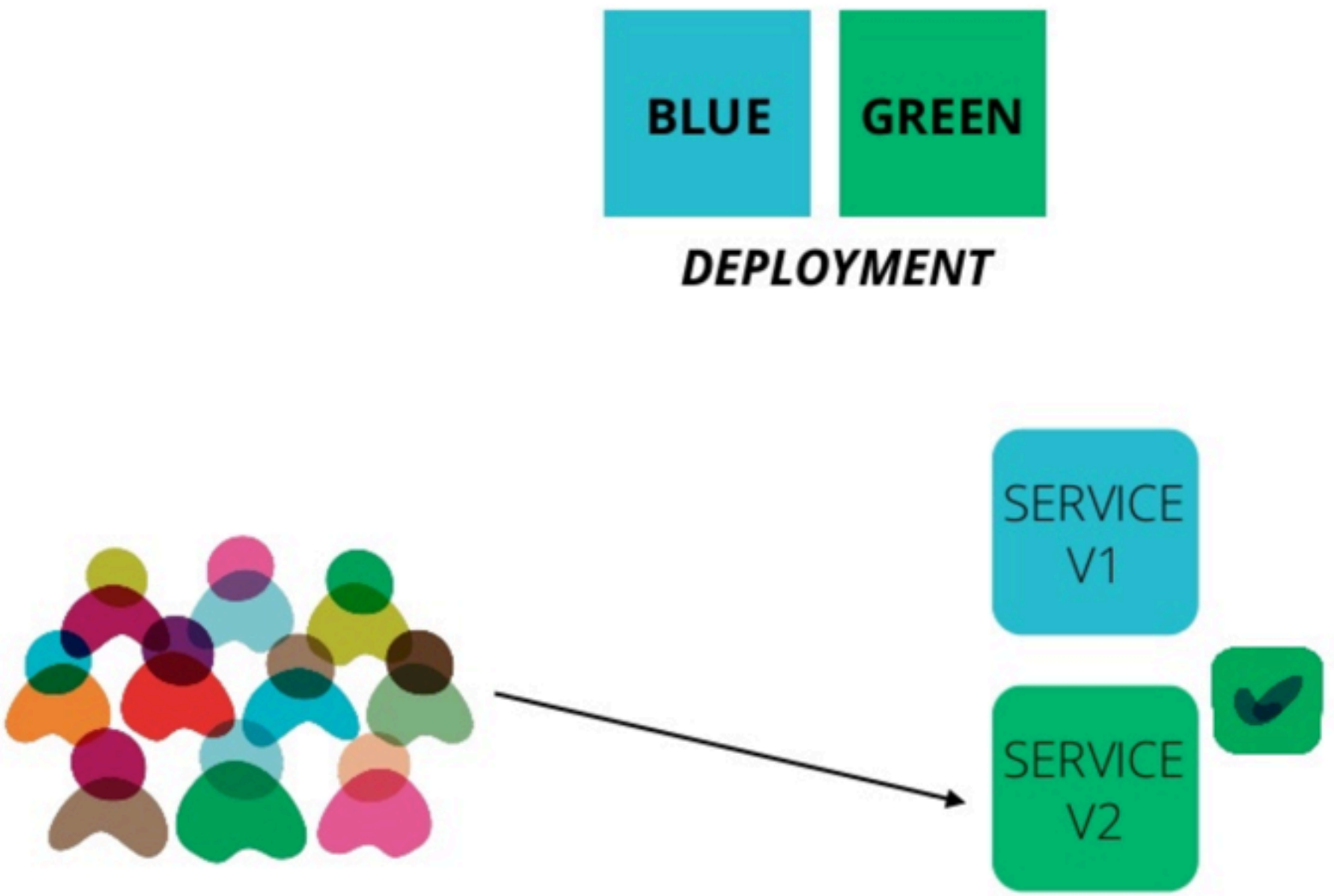
Blue Green Deployment



Blue Green Deployment



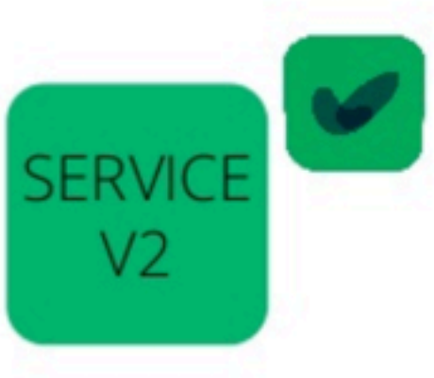
Blue Green Deployment



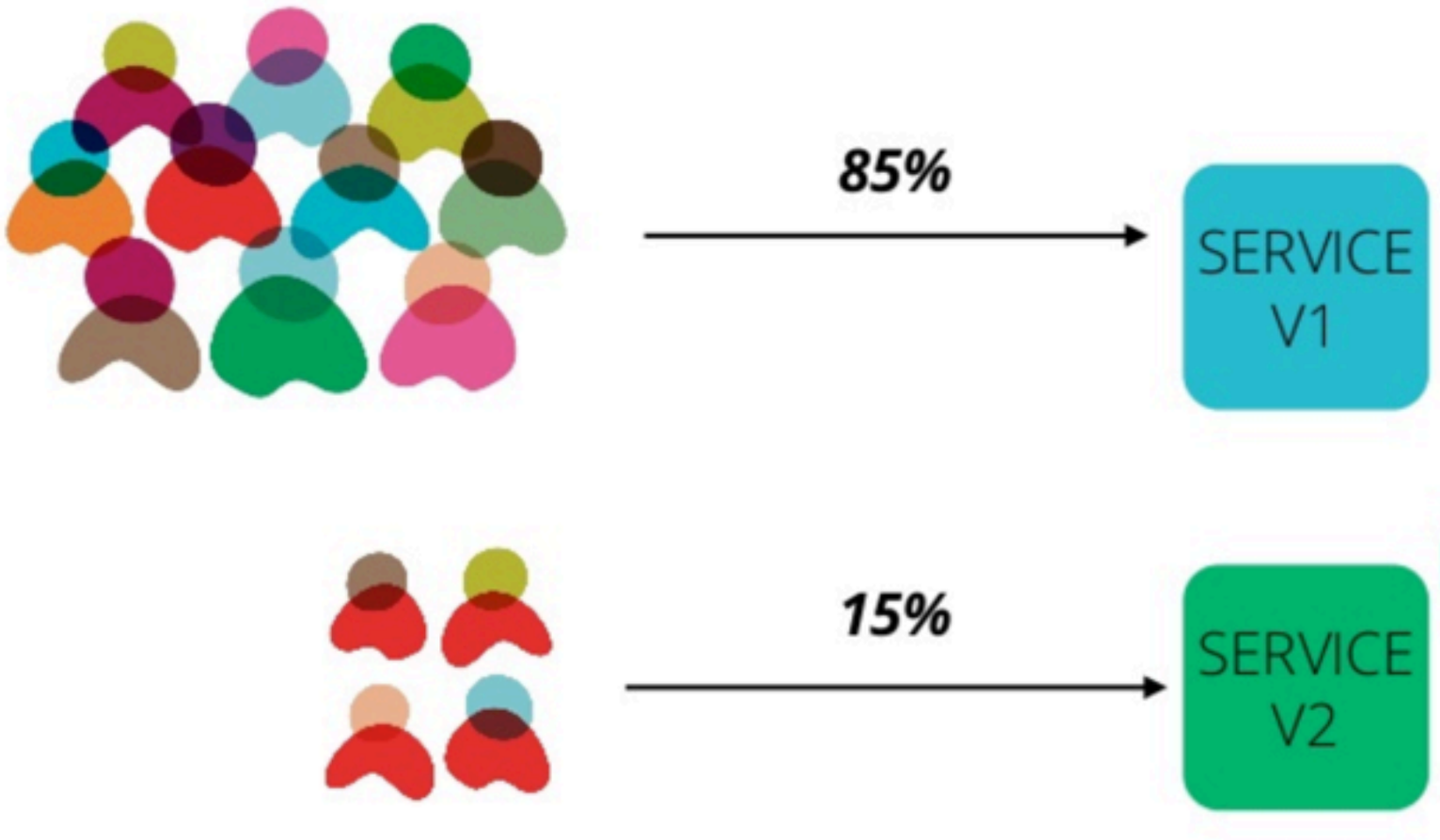
Canary Release

Canary Release

Canary Release



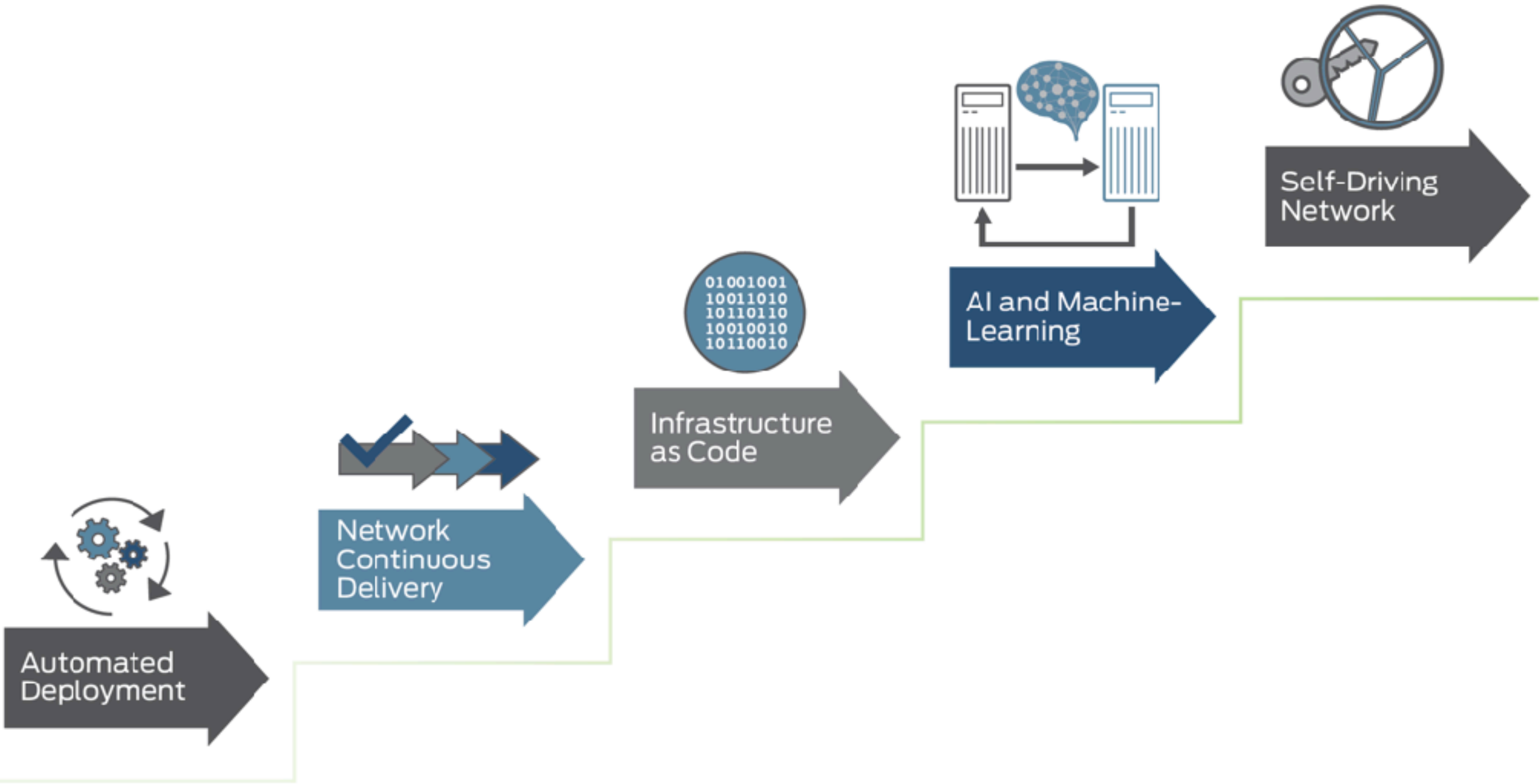
Canary Release



> 500 services !!

Declarative deployment

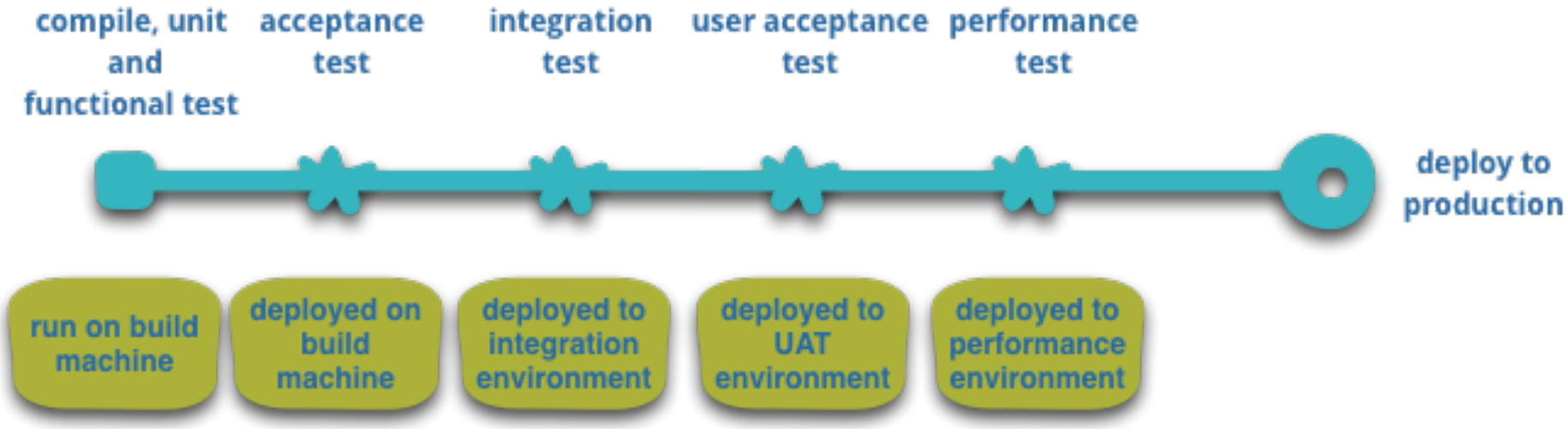
```
1   apiVersion: extensions/v1beta1
2   kind: Deployment
3   metadata:
4     name: webapp-deployment
5   spec:
6     replicas: 4
7     template:
8       metadata:
9         labels:
10          app: webapp-pod
11      spec:
12        containers:
13        - name: webapp
14          image: arungupta/wildfly-app:1
15          ports:
16          - containerPort: 8080
```



https://www.juniper.net/us/en/products-services/what-is/iac/

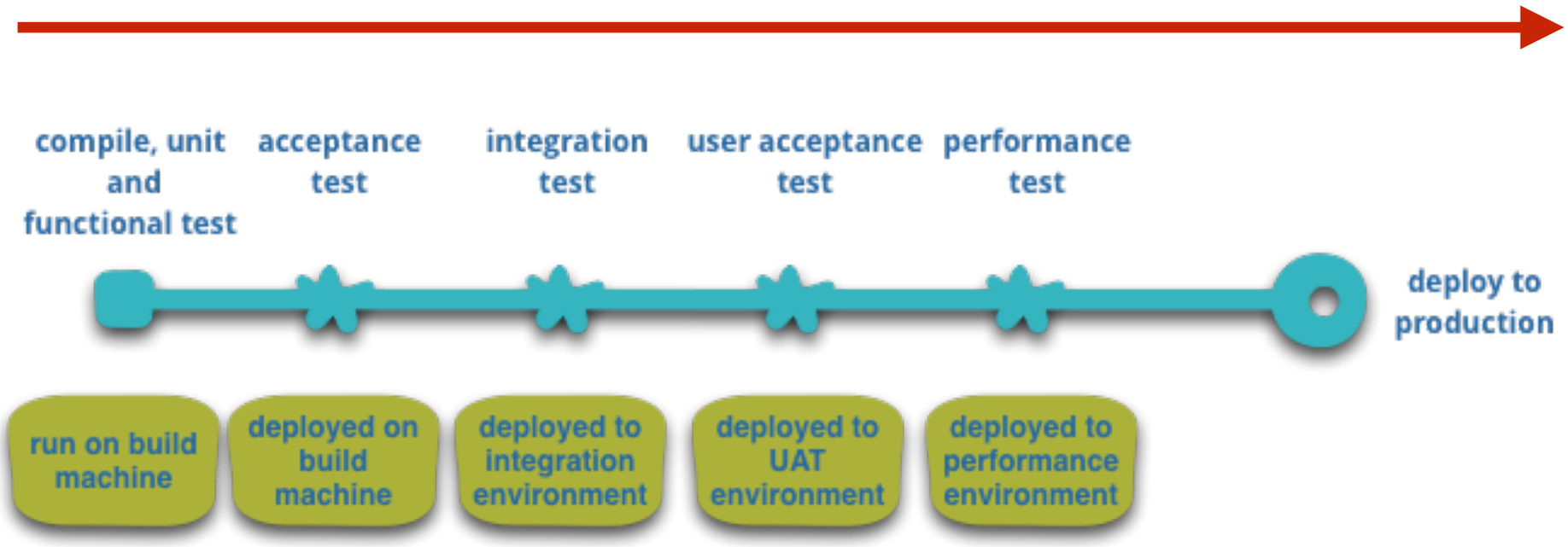
Current situation !!

Infrastructure Automation

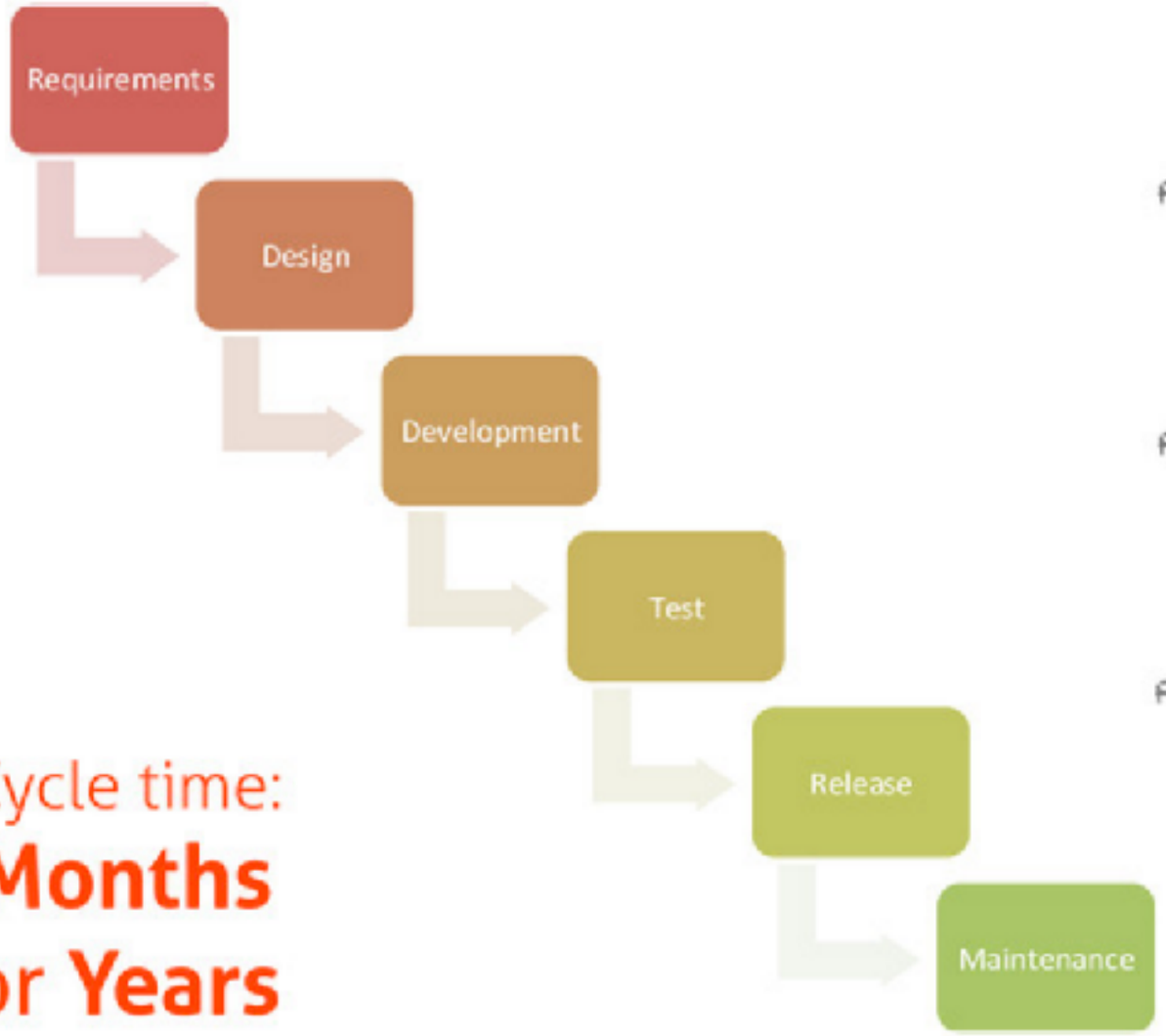


Infrastructure Automation

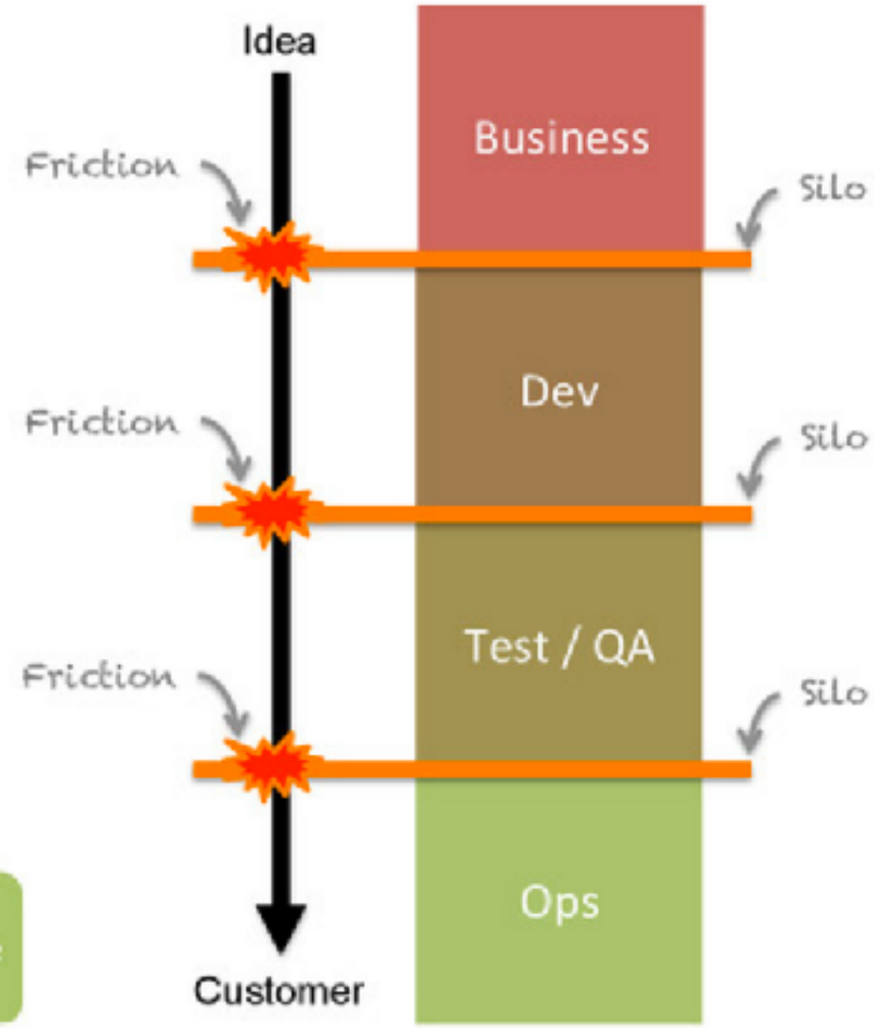
Lead time ?



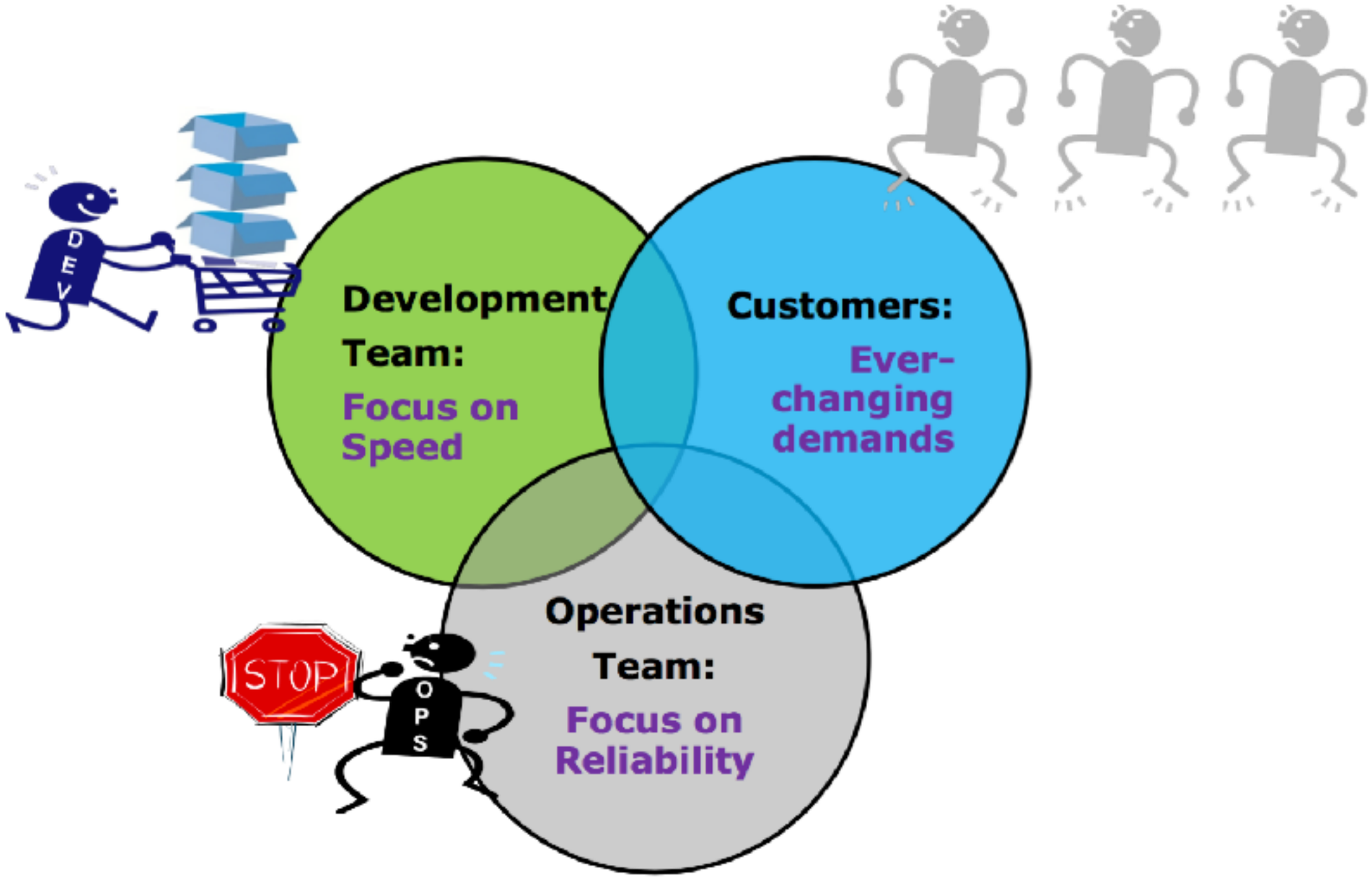
Traditional development



Cycle time:
Months
or **Years**

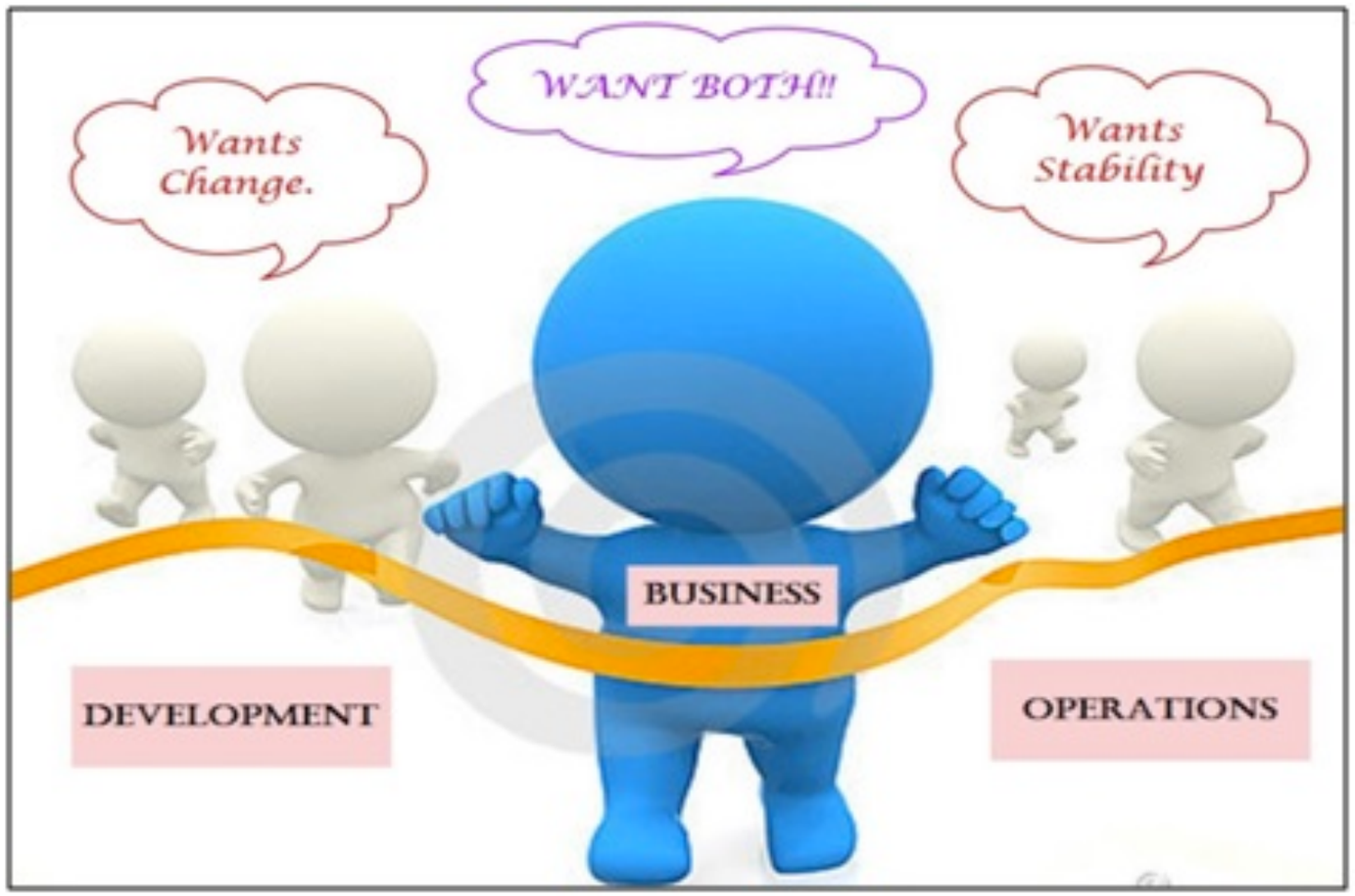


Conflict of Interest

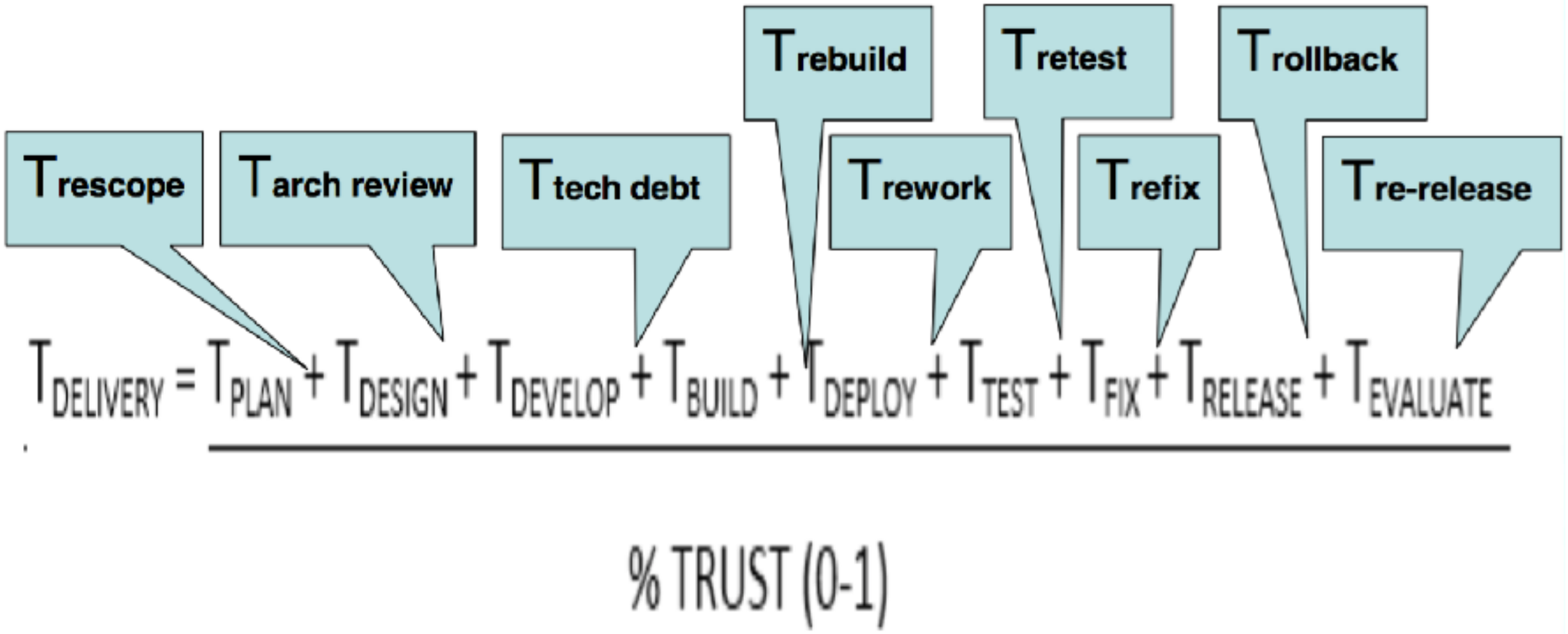


Conflict of Interest

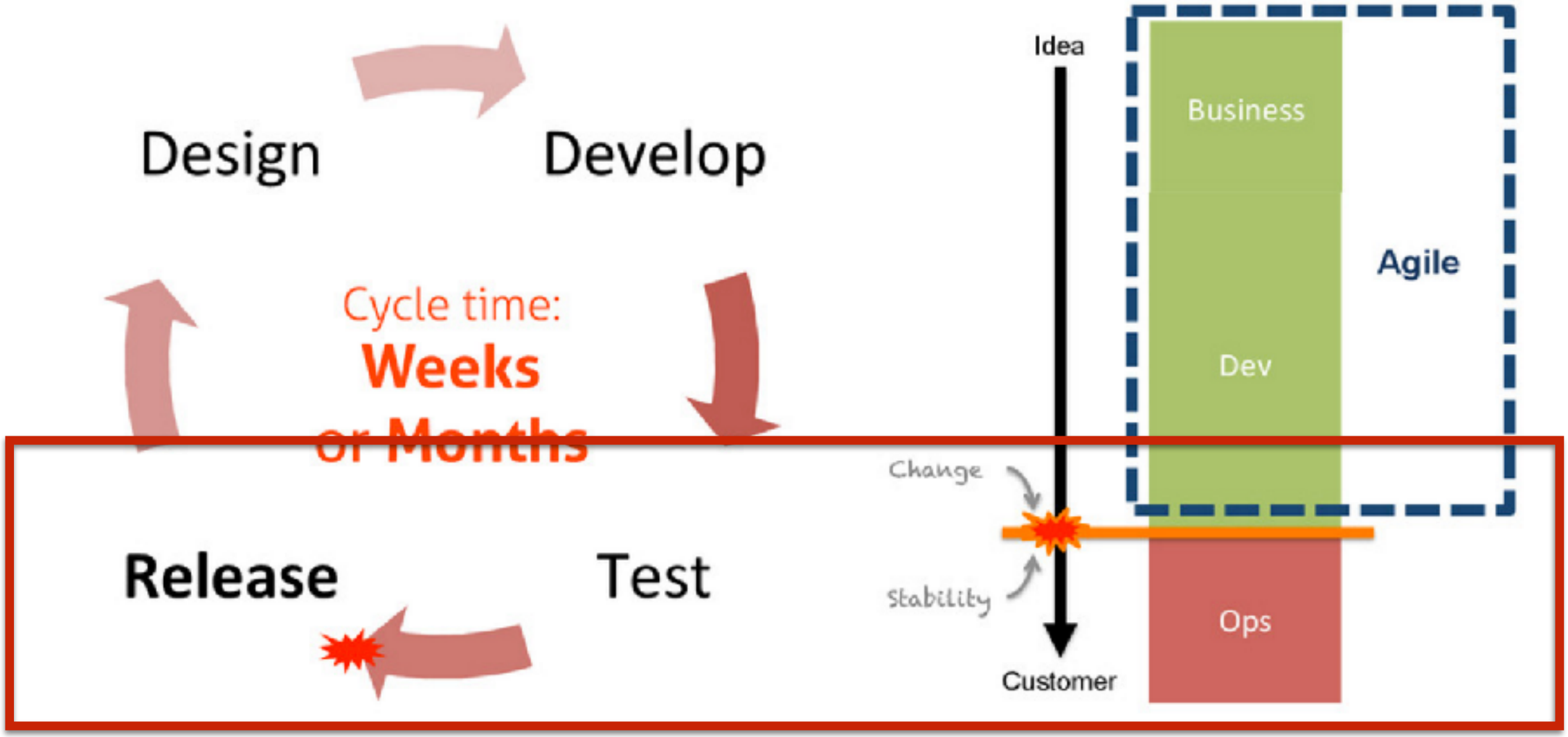
Conflict of Interest



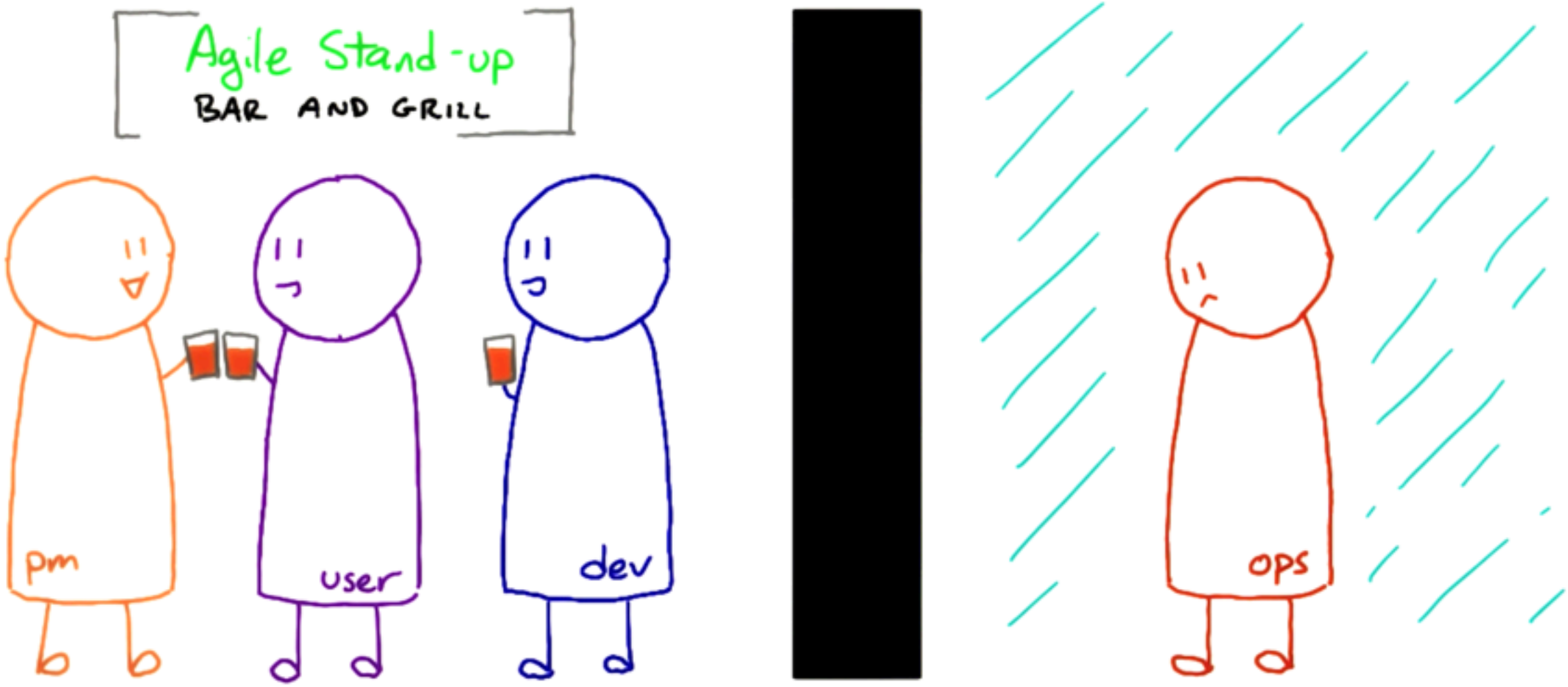
Low trust create extra steps



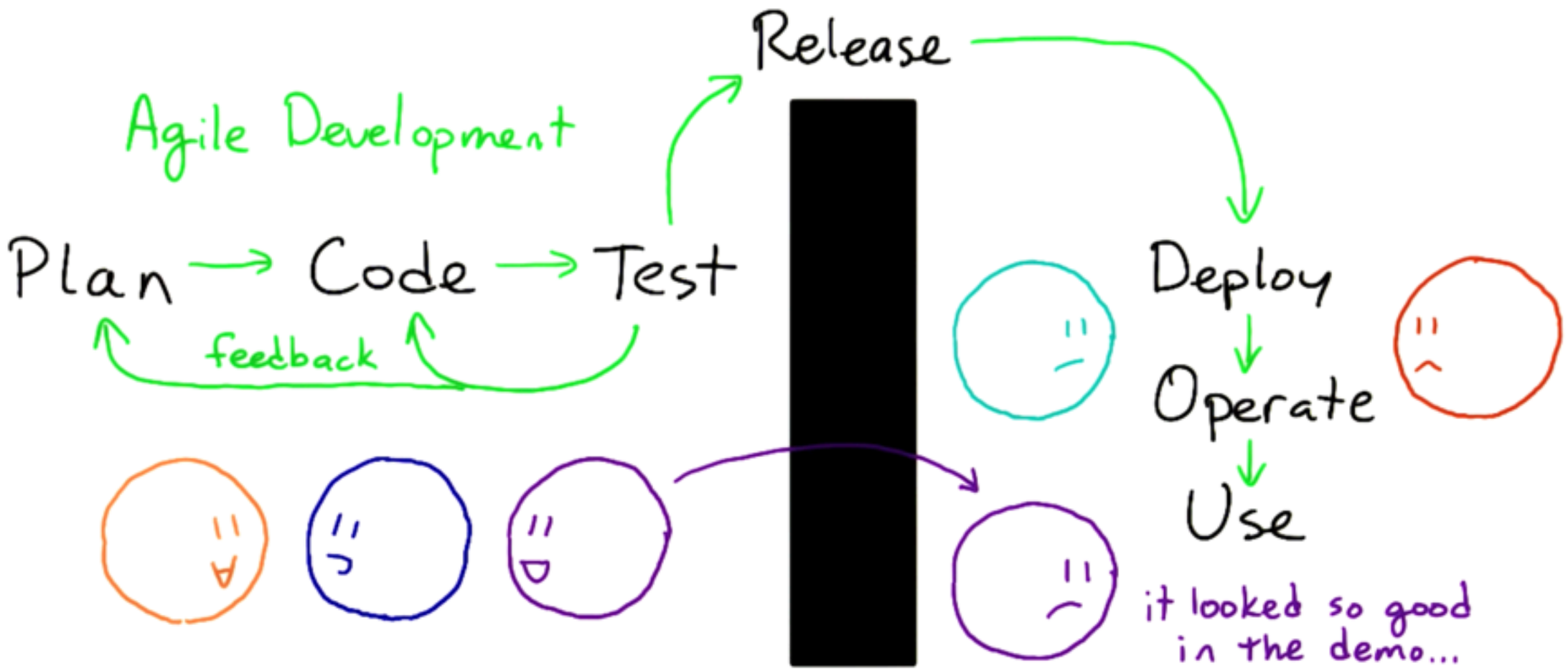
Iterative/Agile development



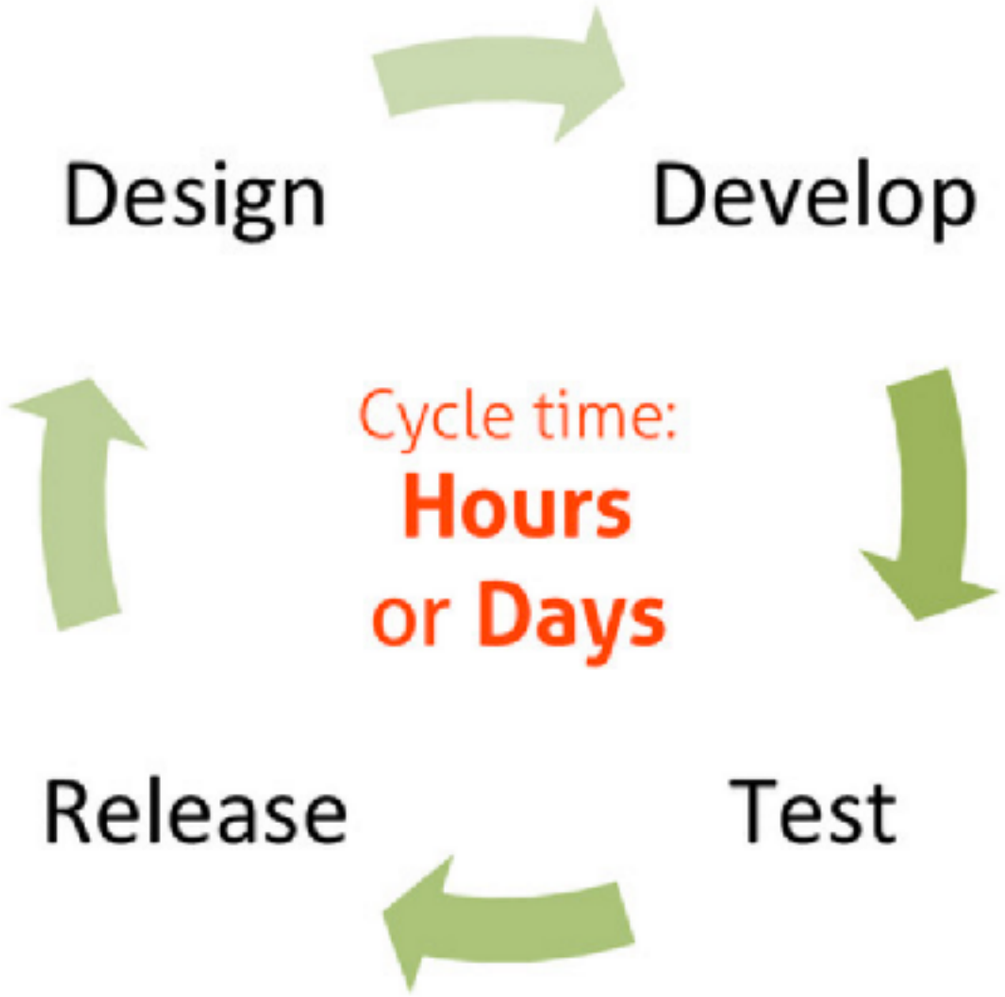
Iterative/Agile development

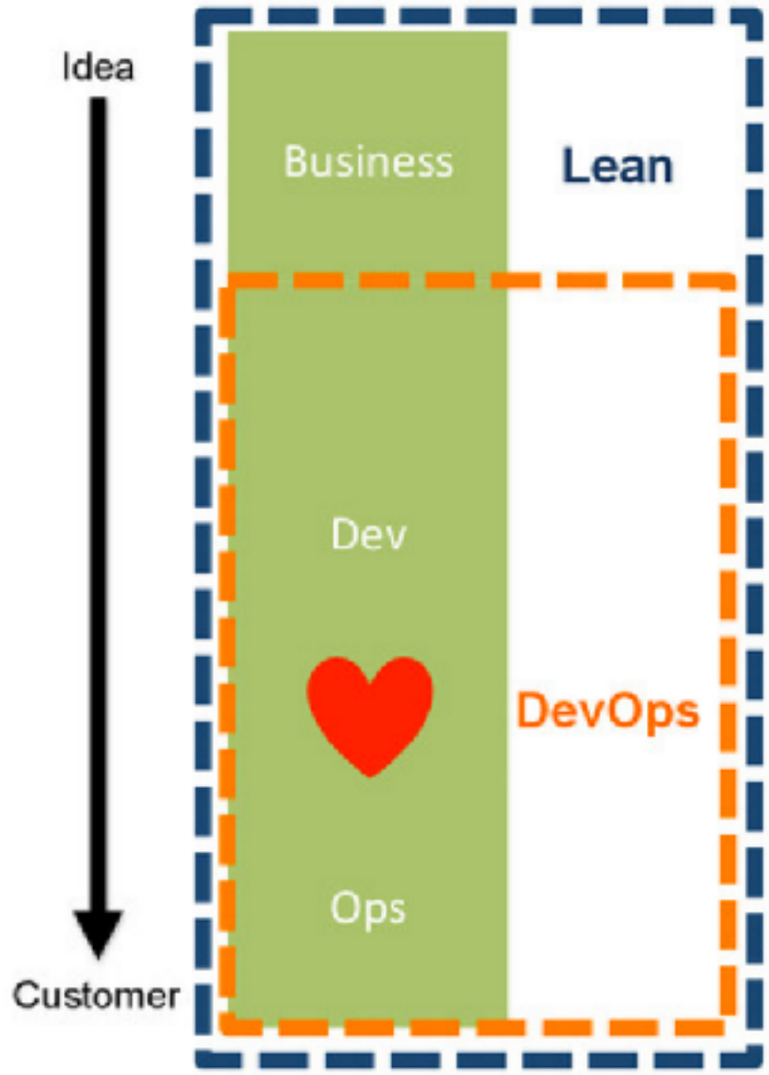


Iterative/Agile development

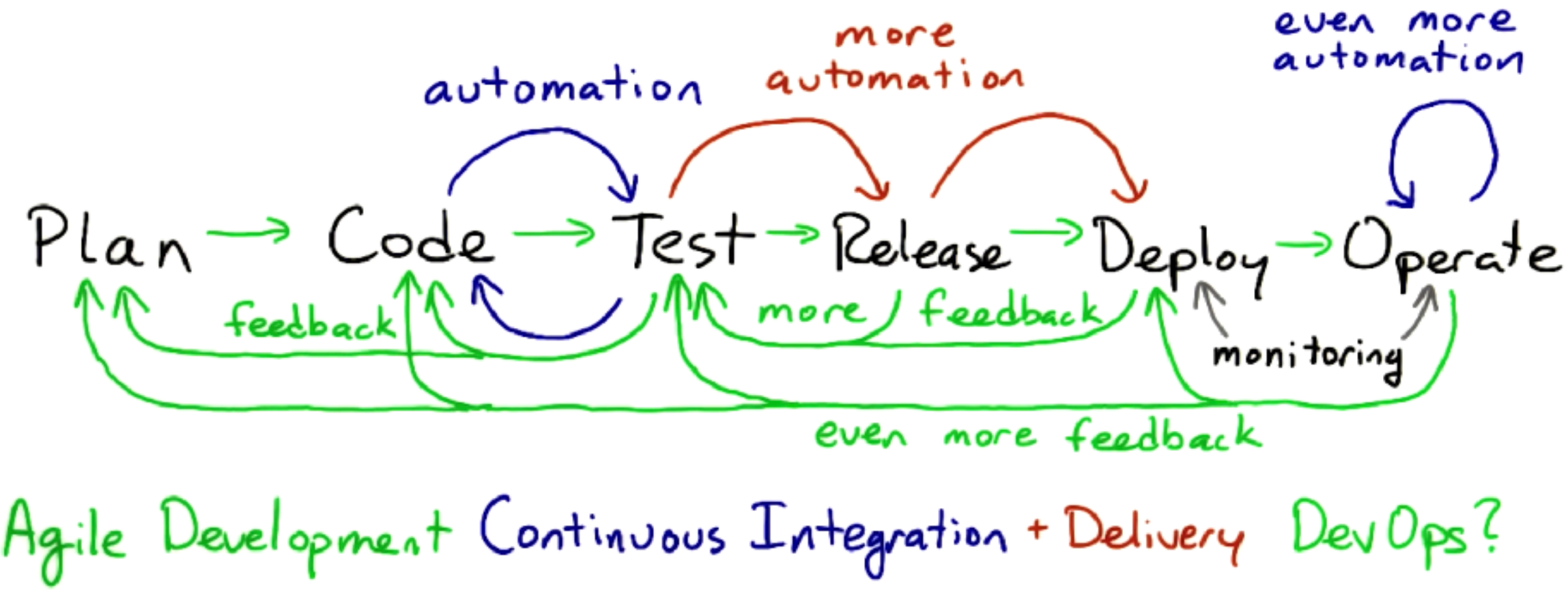


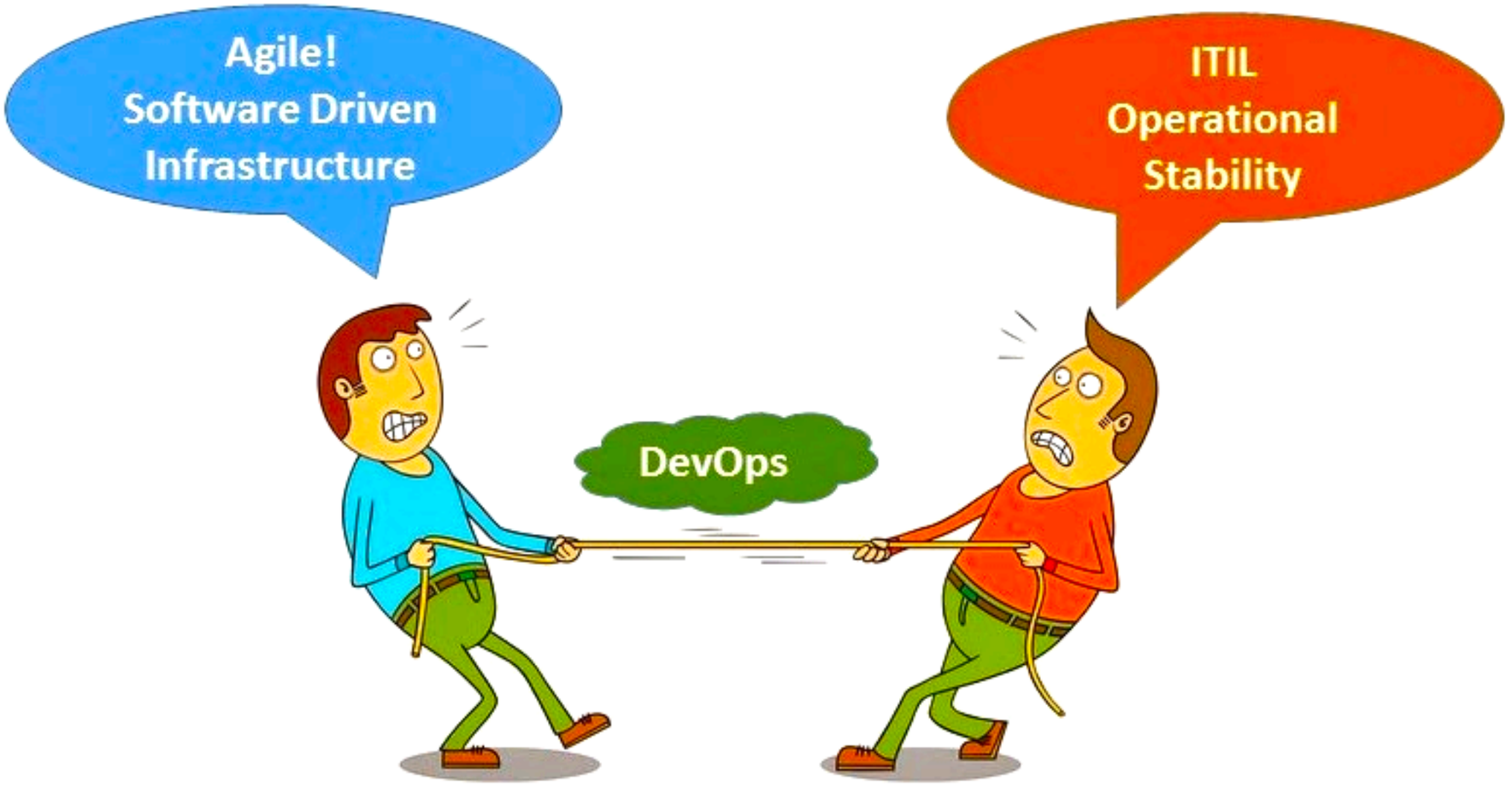
Rise of DevOps

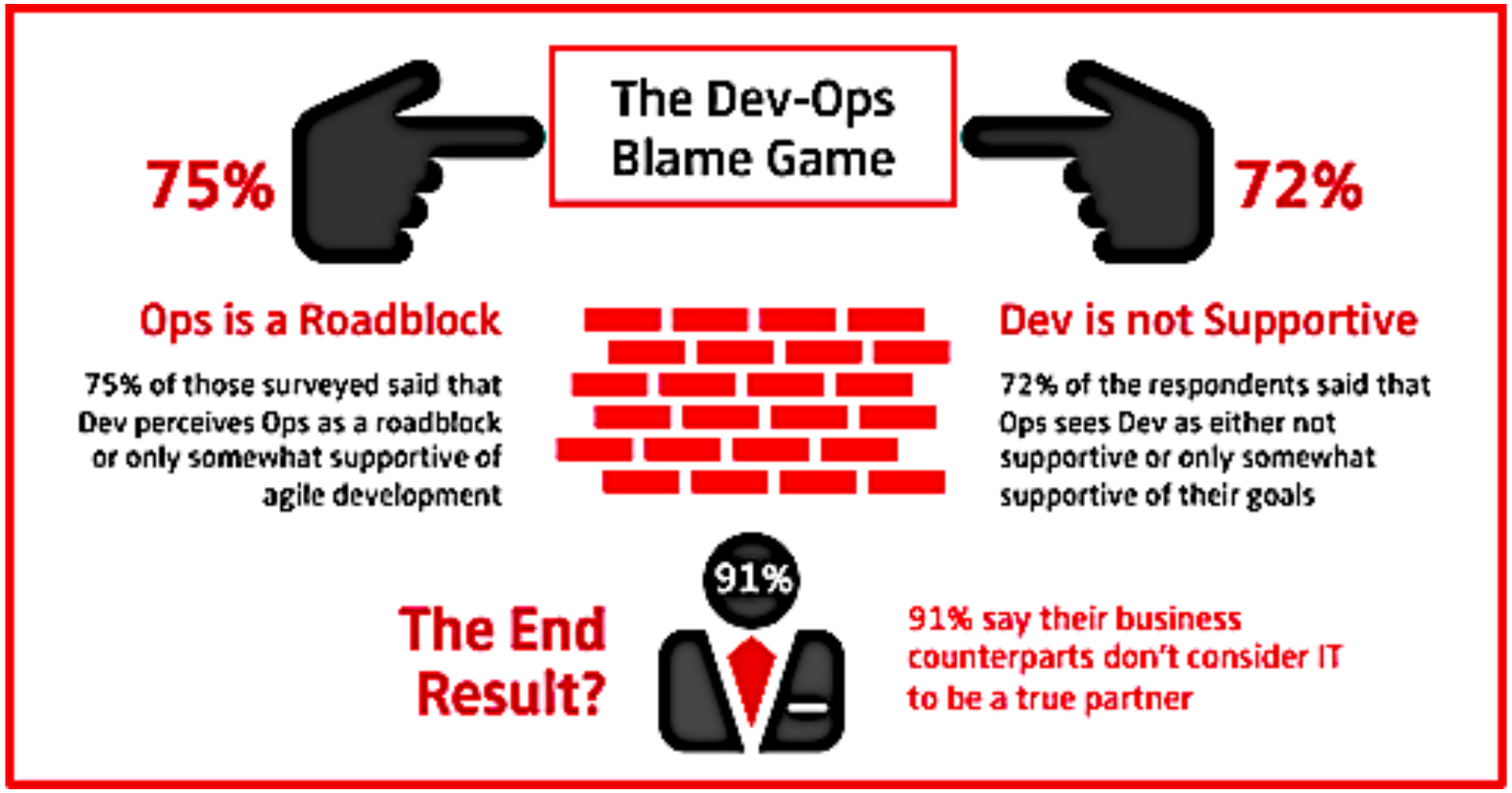


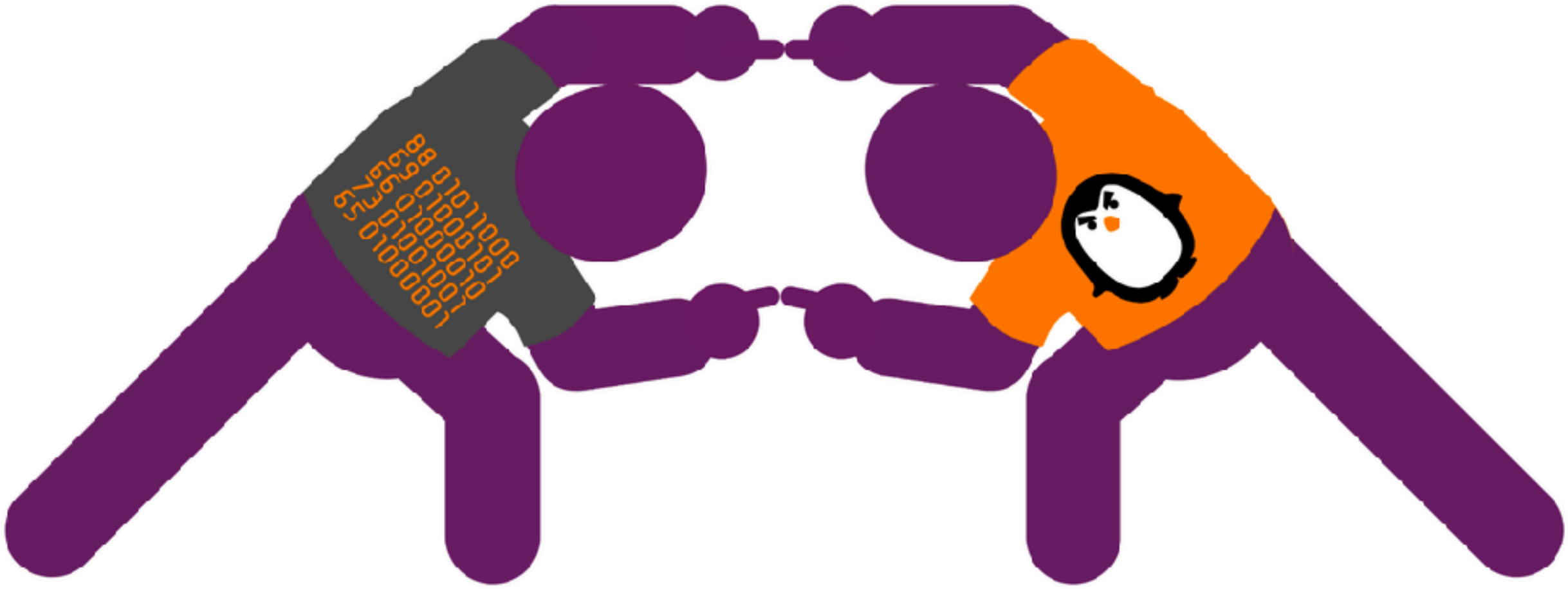


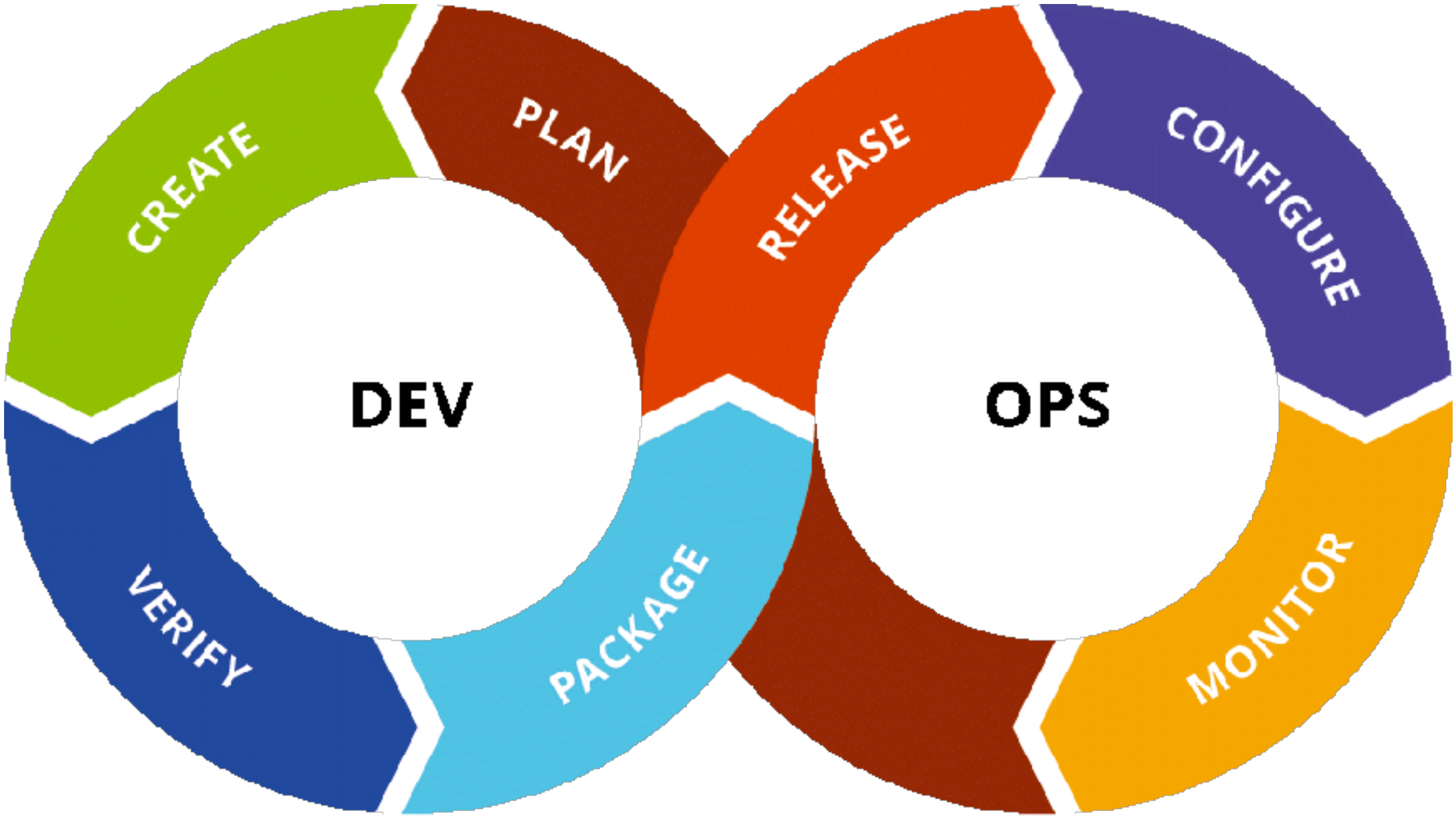
Rise of DevOps











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

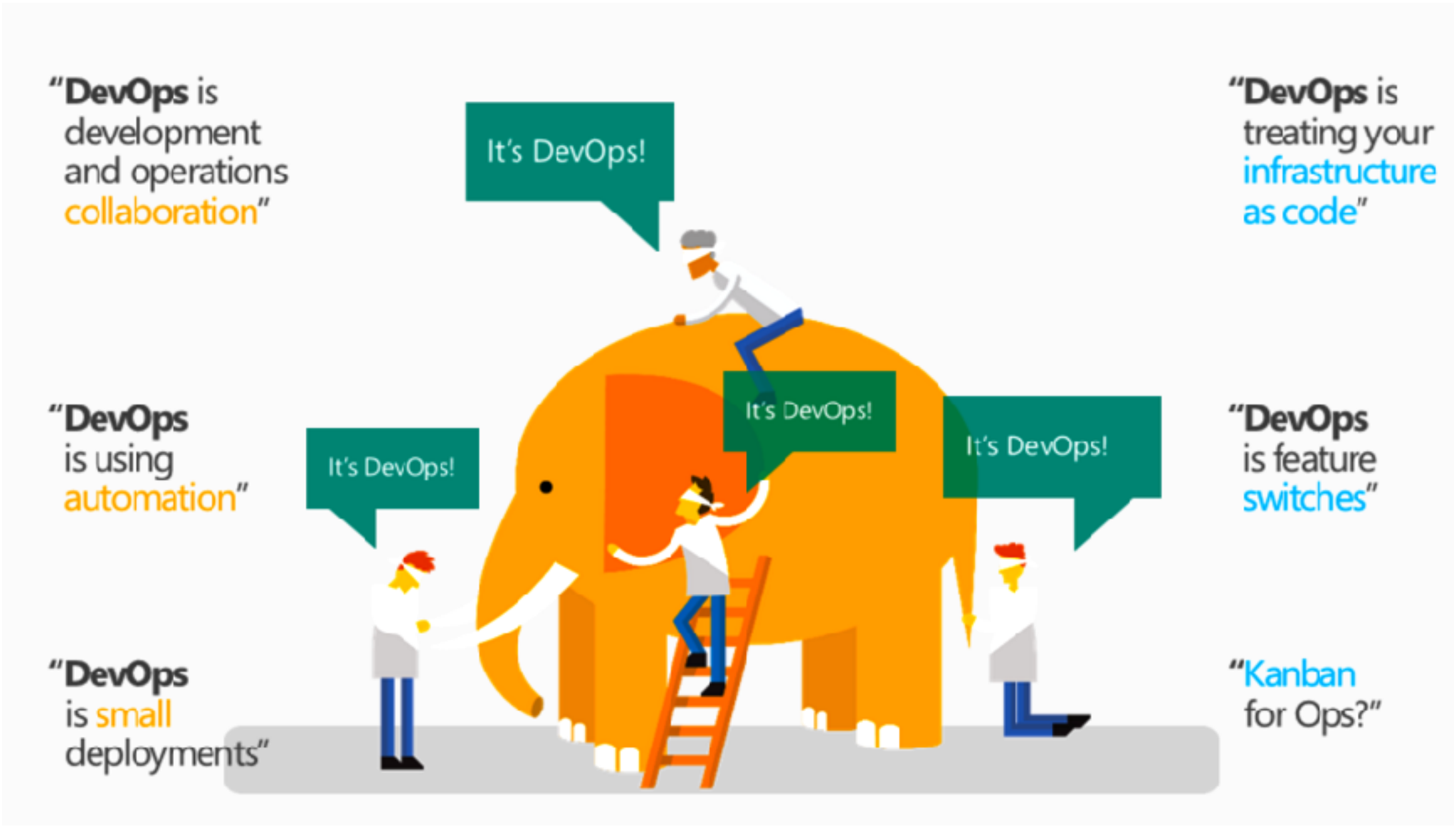
End User Complaints

Fix problems before end users complain

Performance Analytics

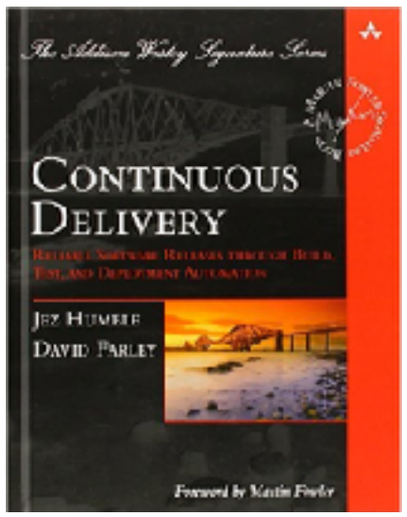
Use automatically generated baselines to focus troubleshooting

DevOps ?



DevOps ?

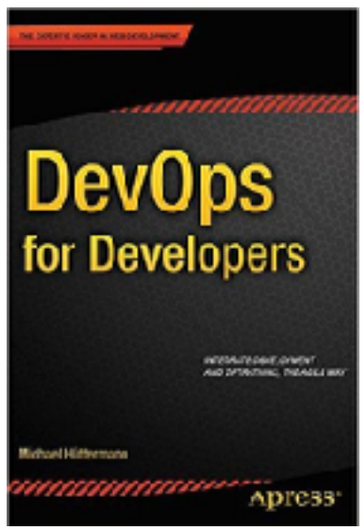
"A **movement of people who care** about **developing** and **operating** reliable, secure, high performance **systems at scale**."



– Jez Humble –

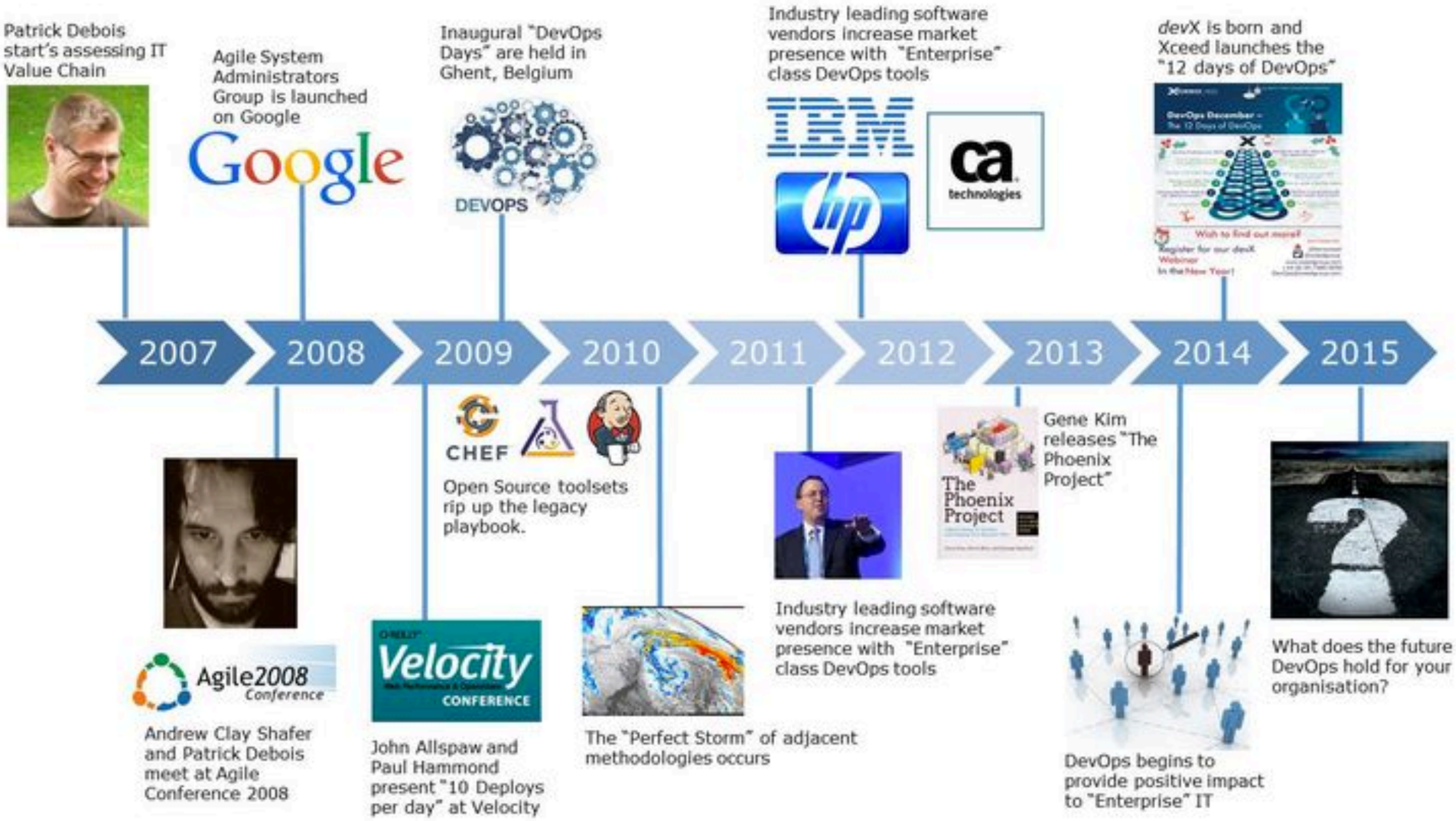
DevOps ?

"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 -

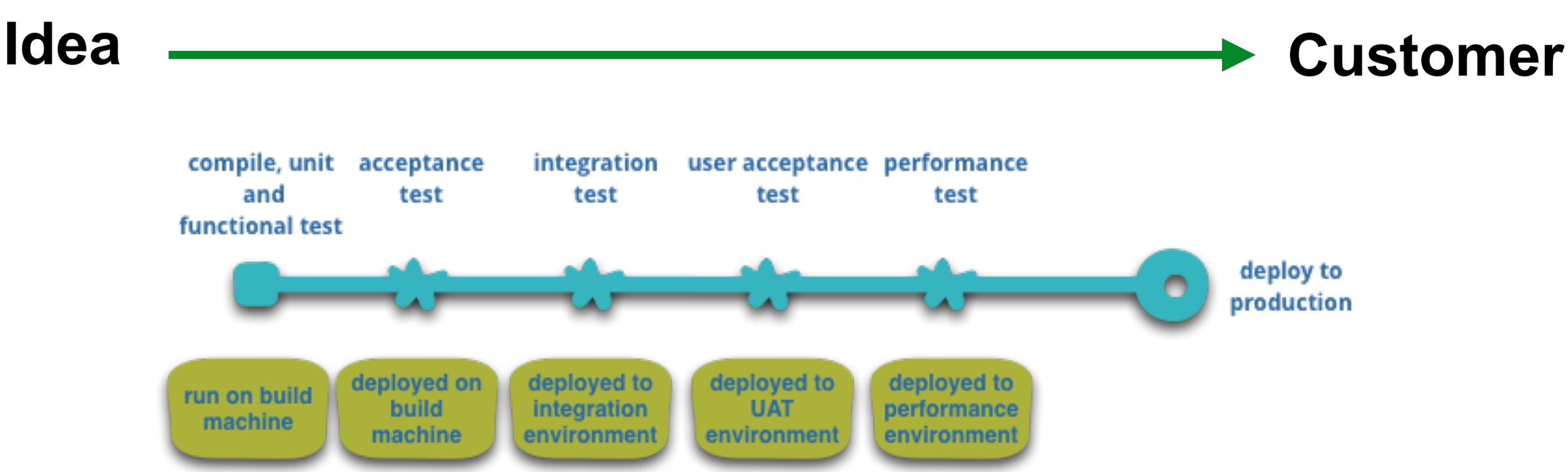
DevOps Timeline



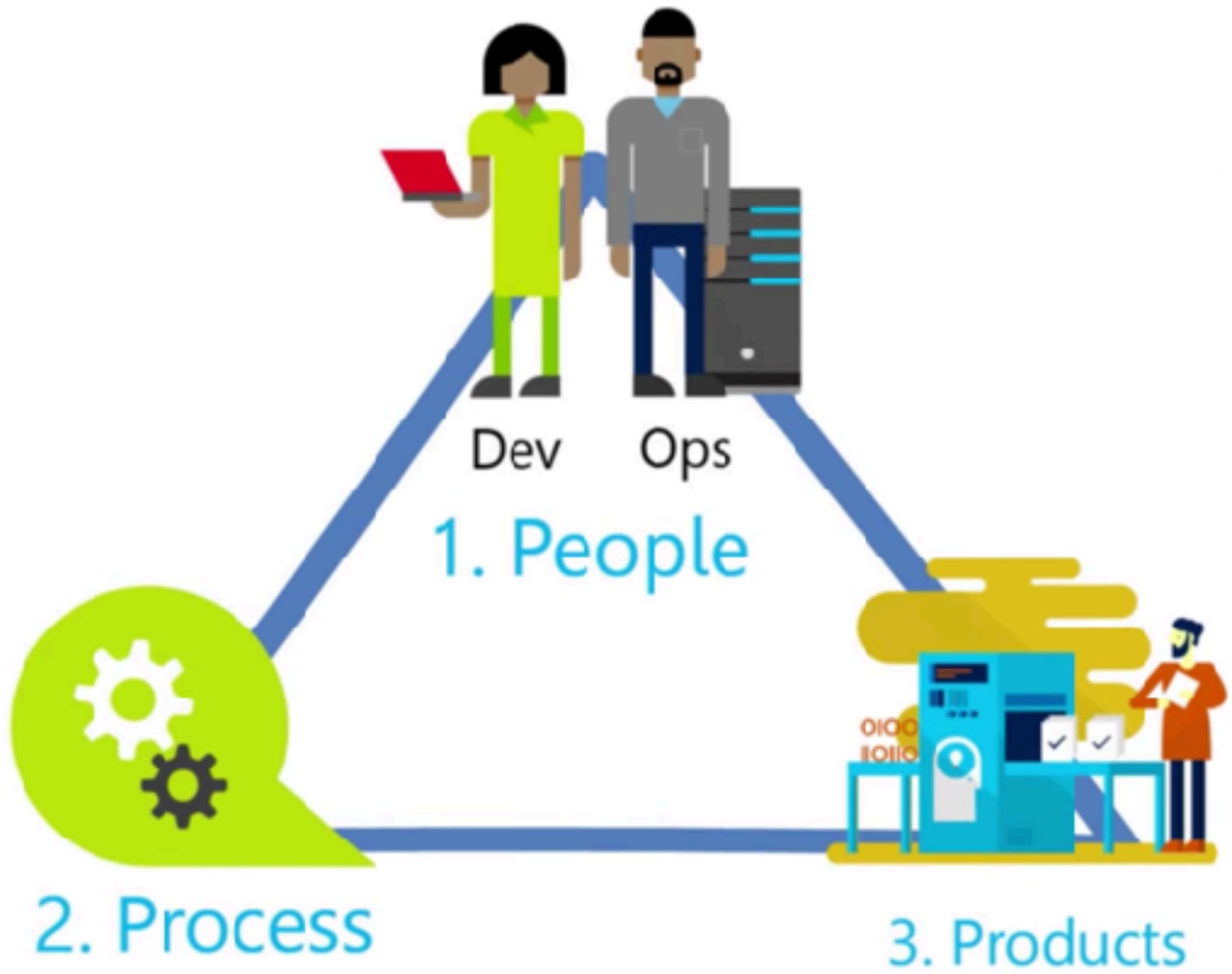
https://www.pinterest.com/pin/263390278185605258/?lp=true

Goal of DevOps

"Improve the delivery of value for Customer and Business"



DevOps



to enable the continuous delivery of value to customer

DevOps adoption model

What kind of problem are we trying to solve?

How can we develop people?

VALUE/PURPOSE

Continuously deliver customer value in a sustainable way with improved lead time, resilience and quality

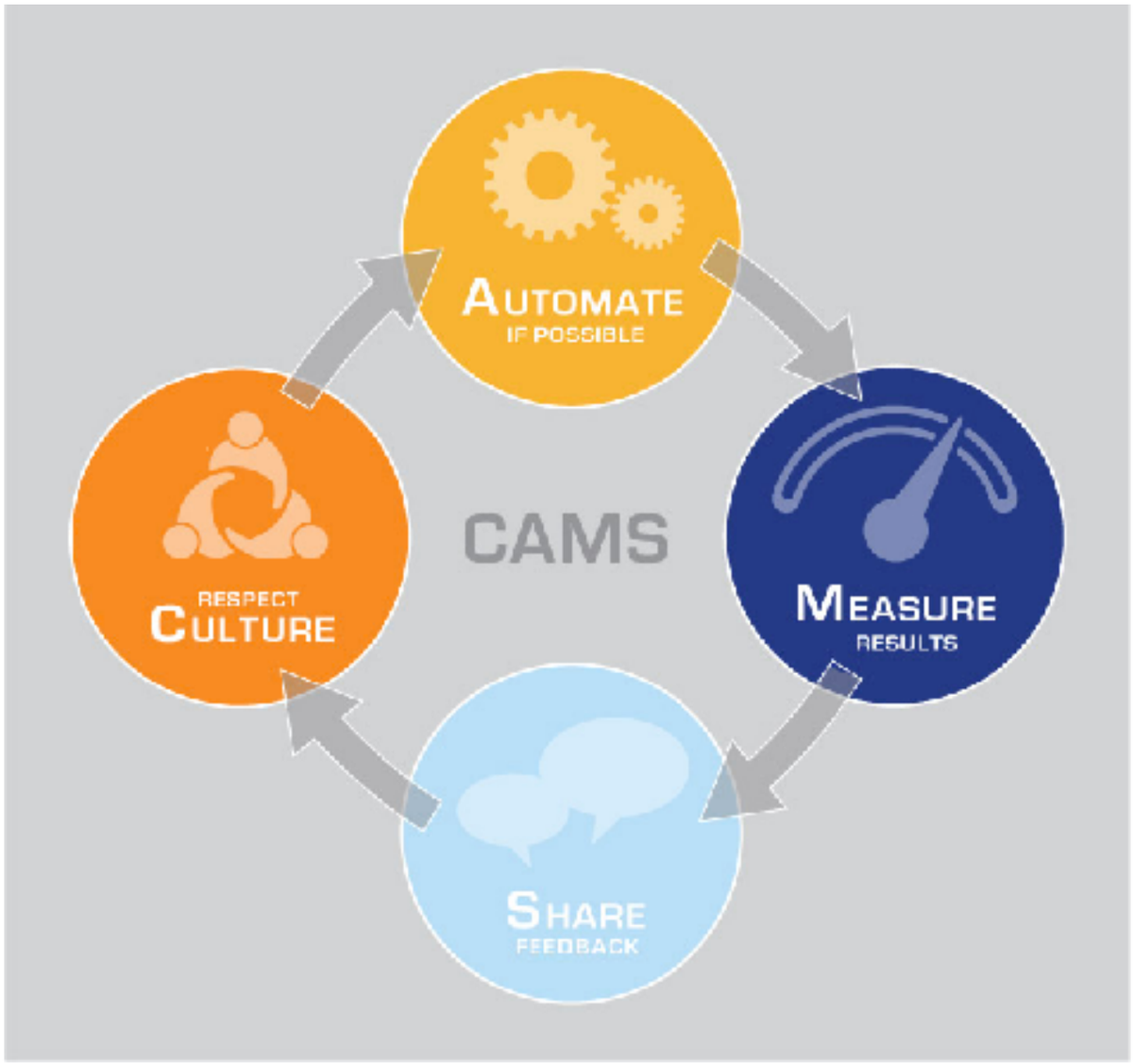
PRINCIPLE	MANAGEMENT FRAMEWORK	PRACTICES
1. Self-directed team Over Command and Control	1. People first	1. Infrastructure as code
2. Course correction over Perfection	2. Organizational structure	2. Dashboard
3. Automated over Manual	3. Capability development	3. Working backwards
	4. Budget	4. Open source
	5. Risk management	5. DevOps skill matrix
	6. Asset management	6. Audit automation
	7. Change management	7. Self service
	8. Security management	8. And etc.

How can we improve the work

ENVIRONMENT:

People, Organization, Laws, Regulations, Culture, Competitors

DevOps Principles



DevOps Principles

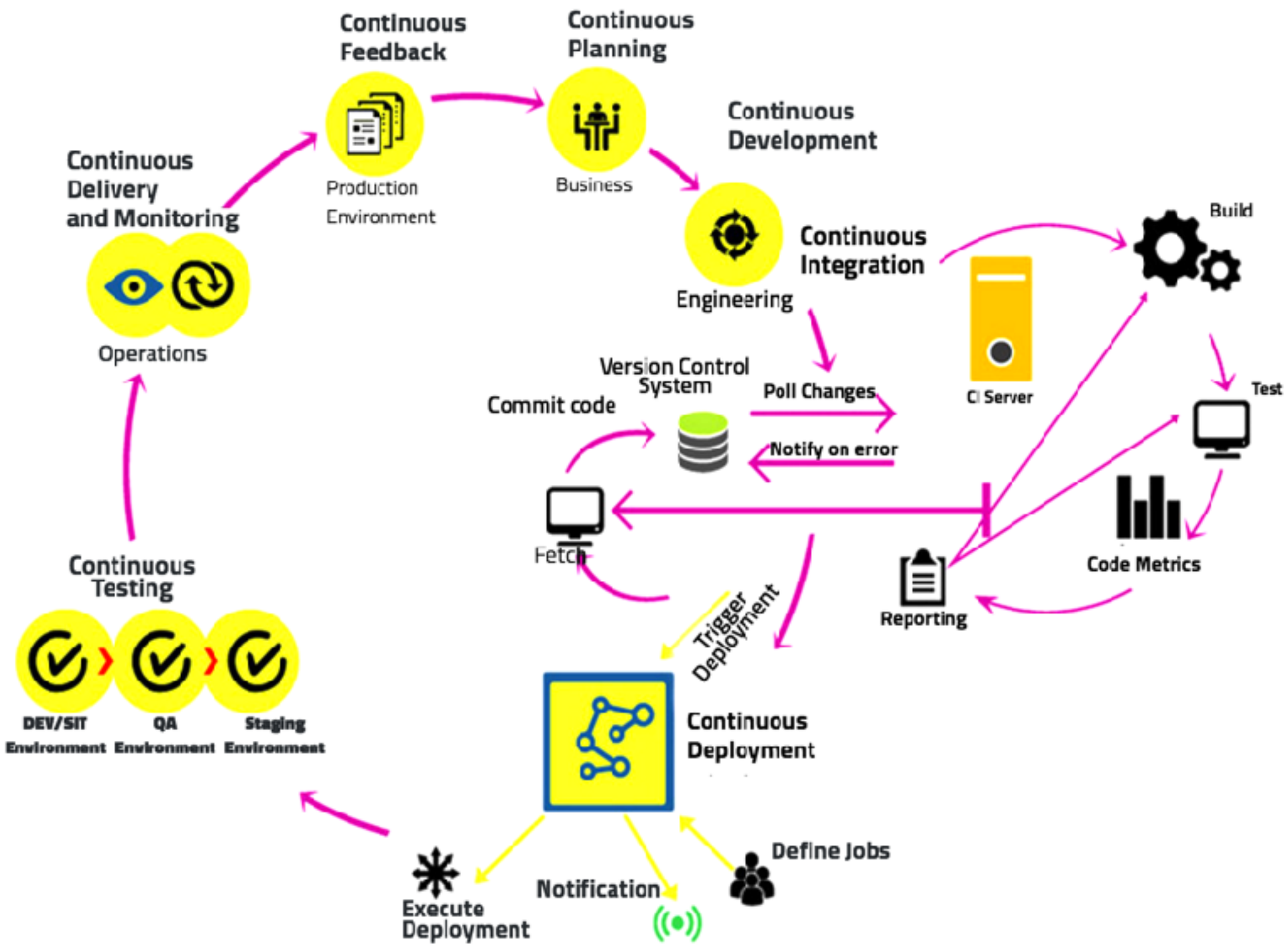
Culture => People, Process, Tools

Automation => Infrastructure as Code

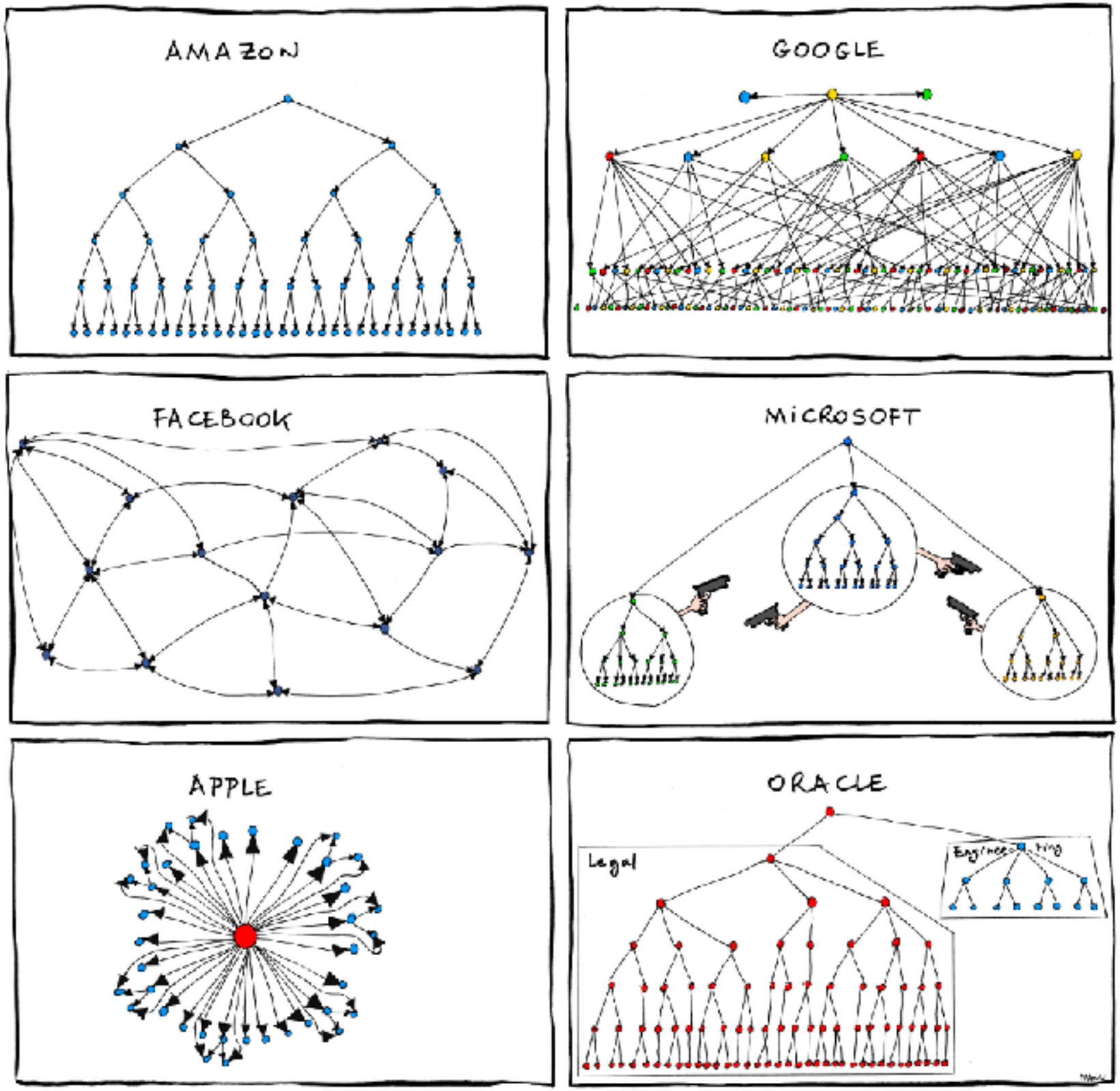
Measurement => Measure everything

Sharing => Collaboration/Feedback

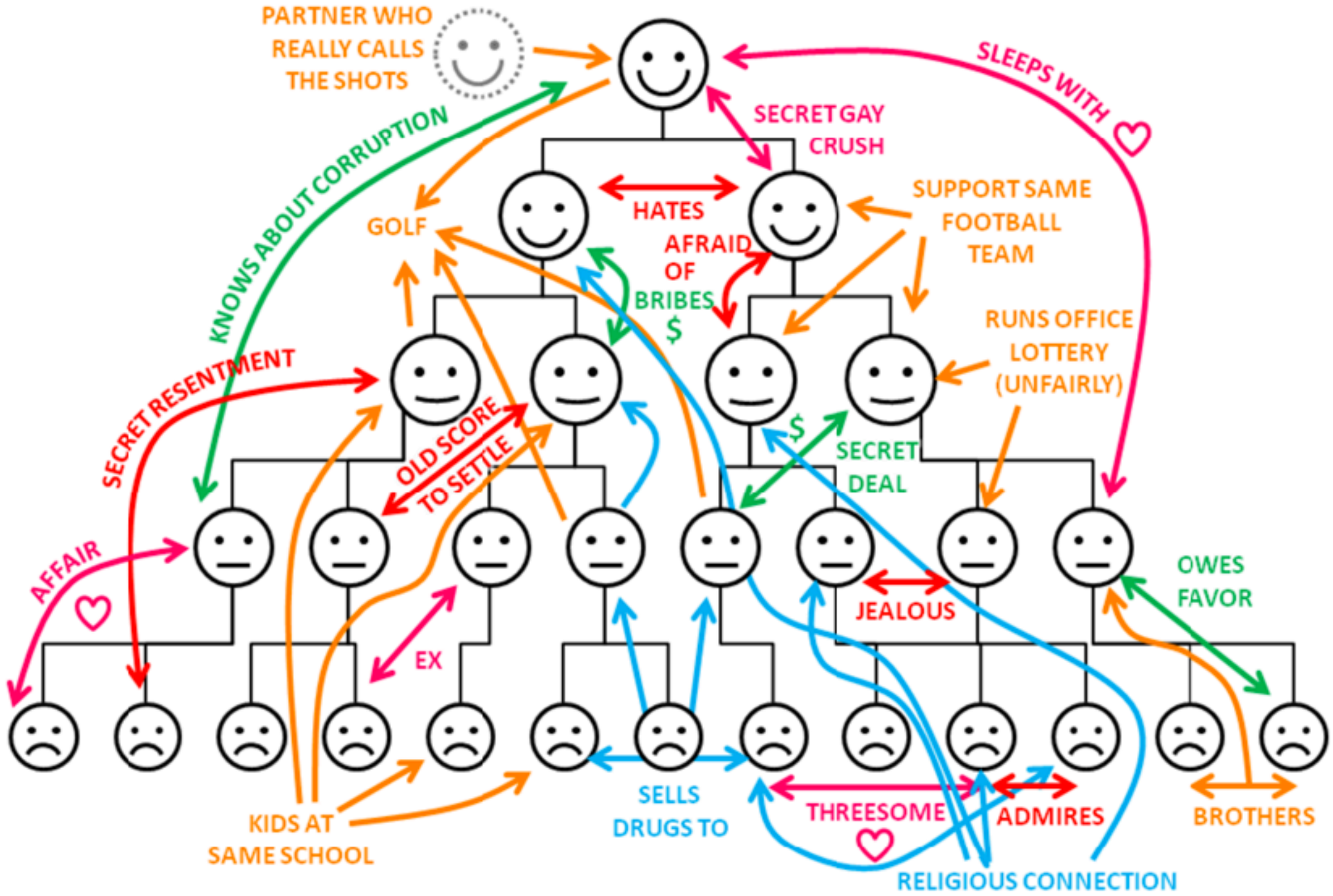
DevOps Practices



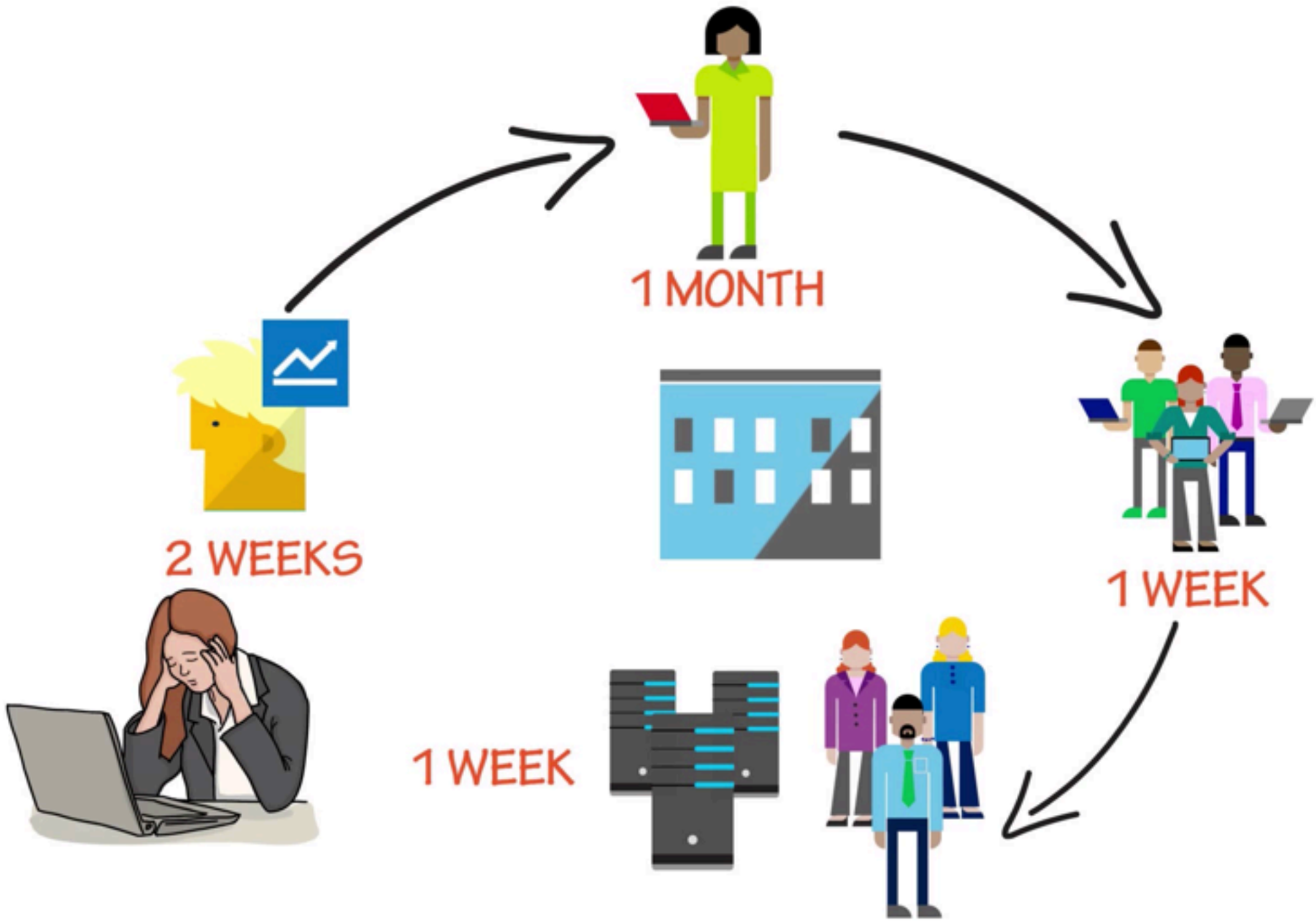
All about Organization structure and culture



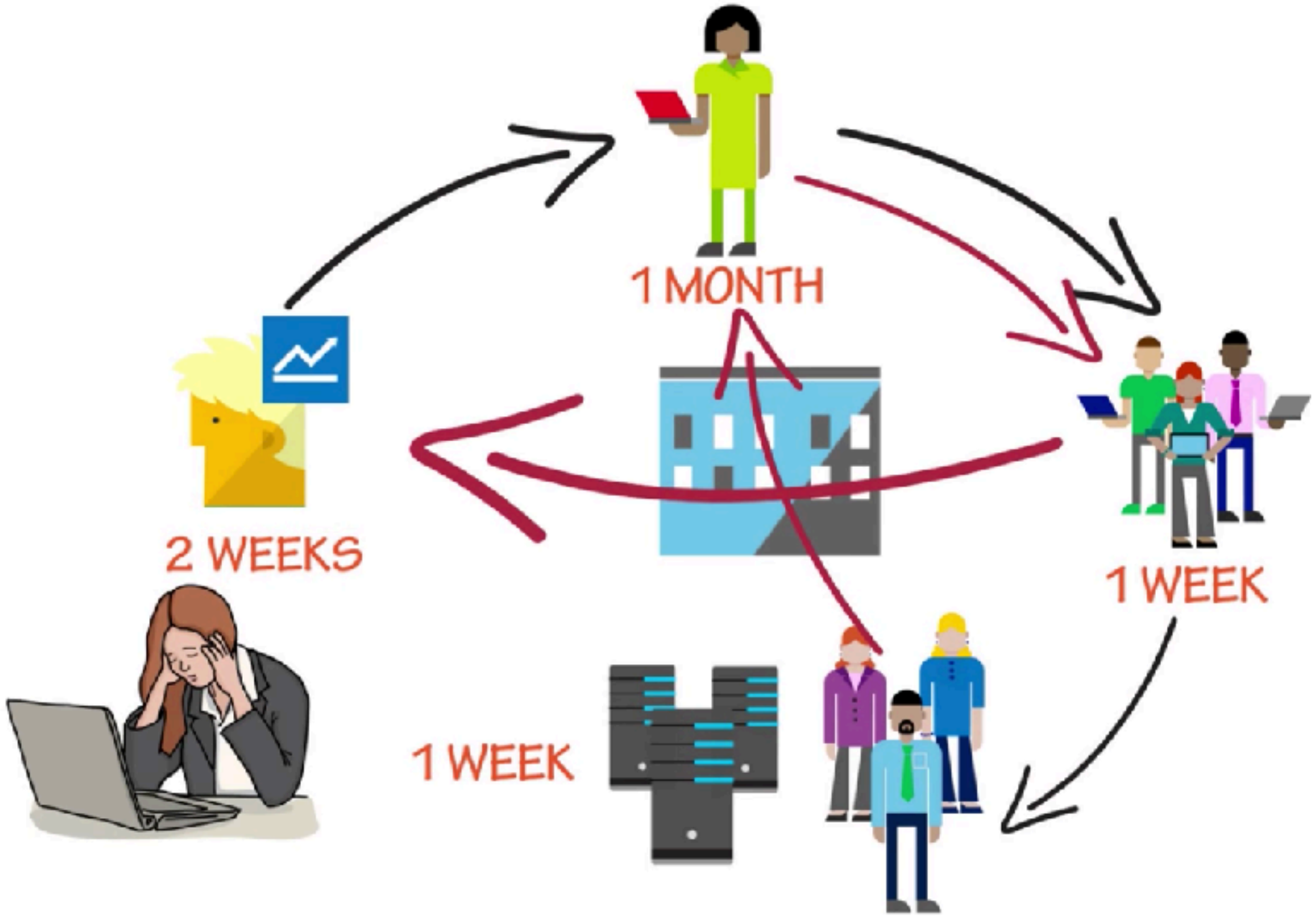
Problem ?



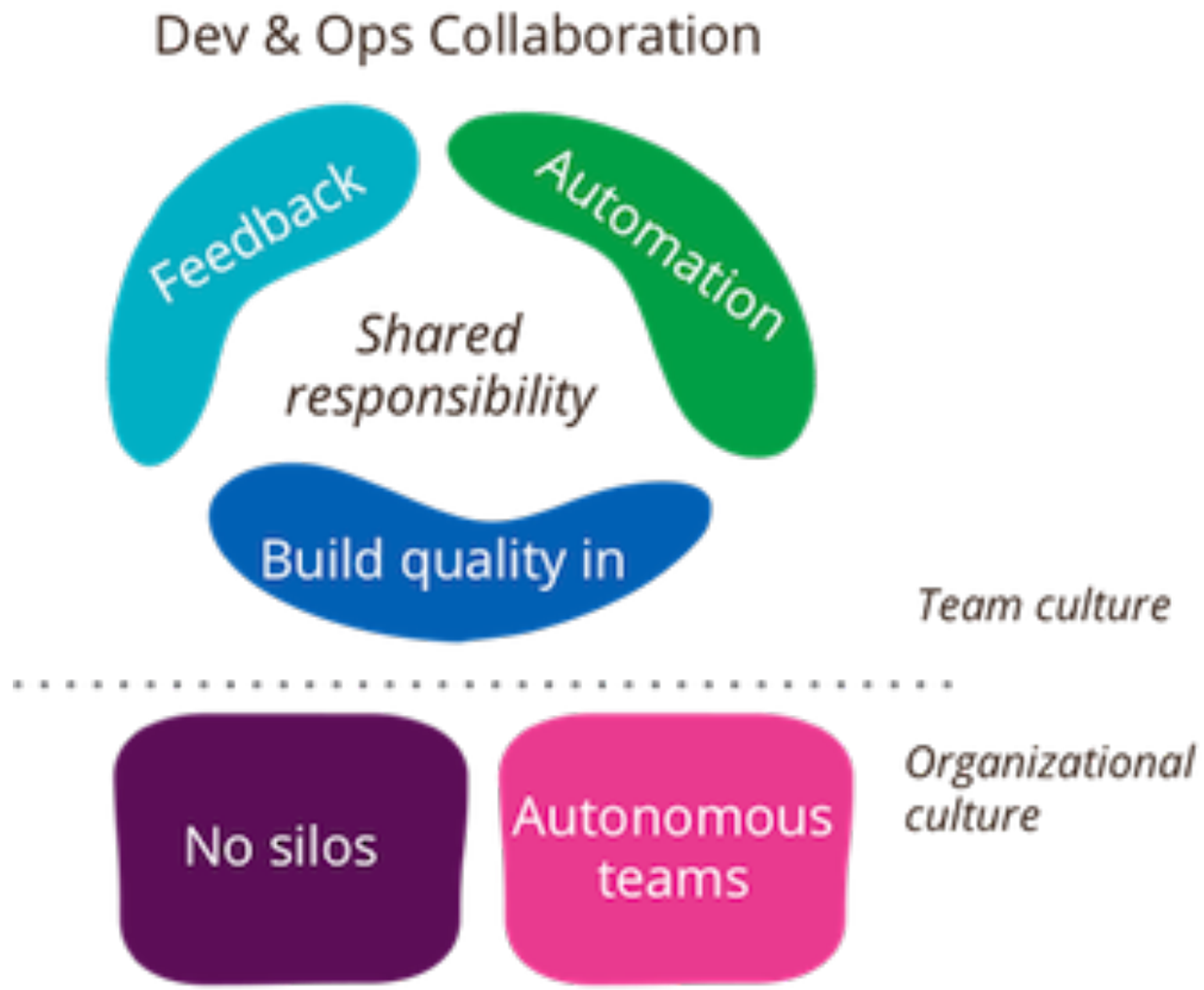
Problem ?



Problem ?

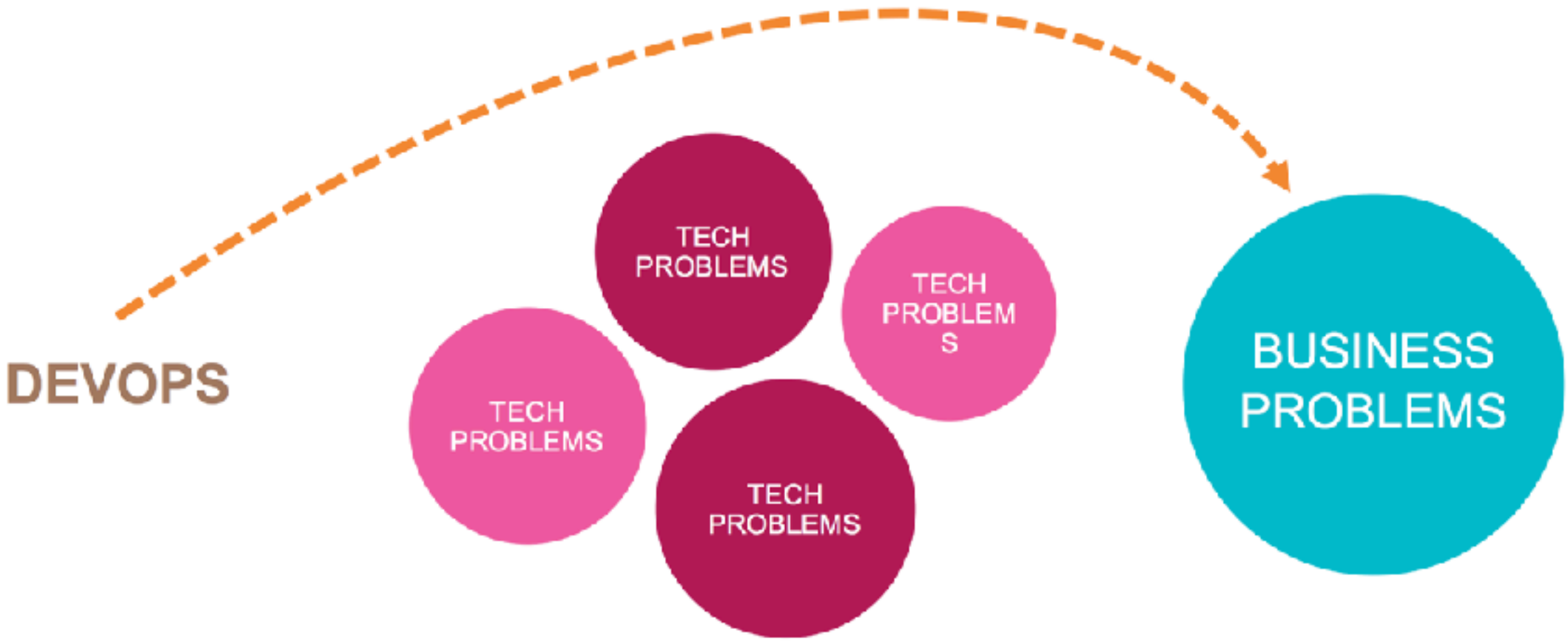


Team and Organization culture



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

Explore business value



DevOps solves business problem, not technical problem

Don’t talk to business

DevOps
Continuous Integration/Delivery
Automation
Configuration management
Test-Driven development
Infrastructure as a code

Talk to business

Reduce time to market
Reduce lead time
Improve quality
Improve resilience

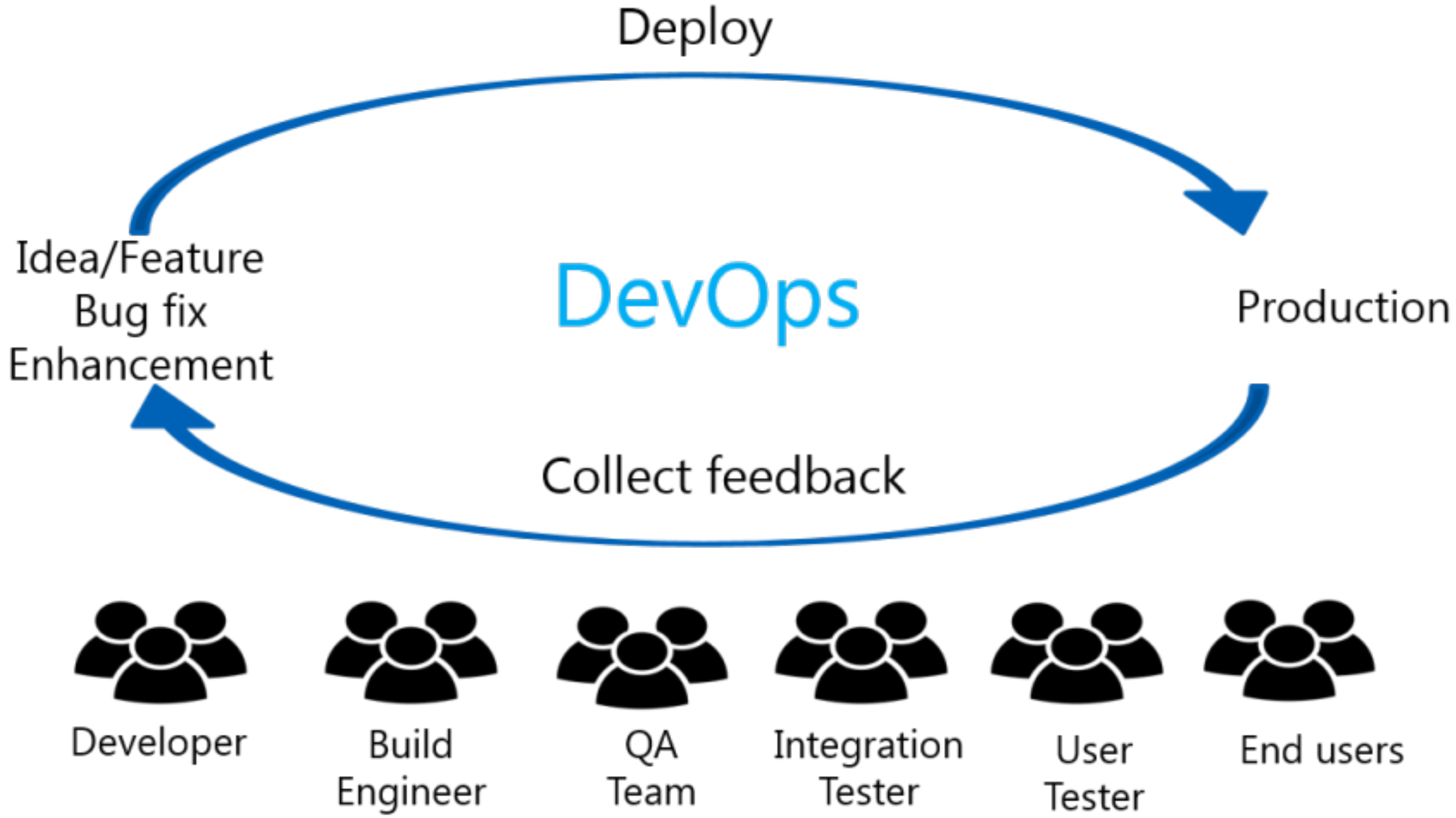
“DevOps is not a goal,
But a process of continuous improvement”

DevOps
3 ways principle

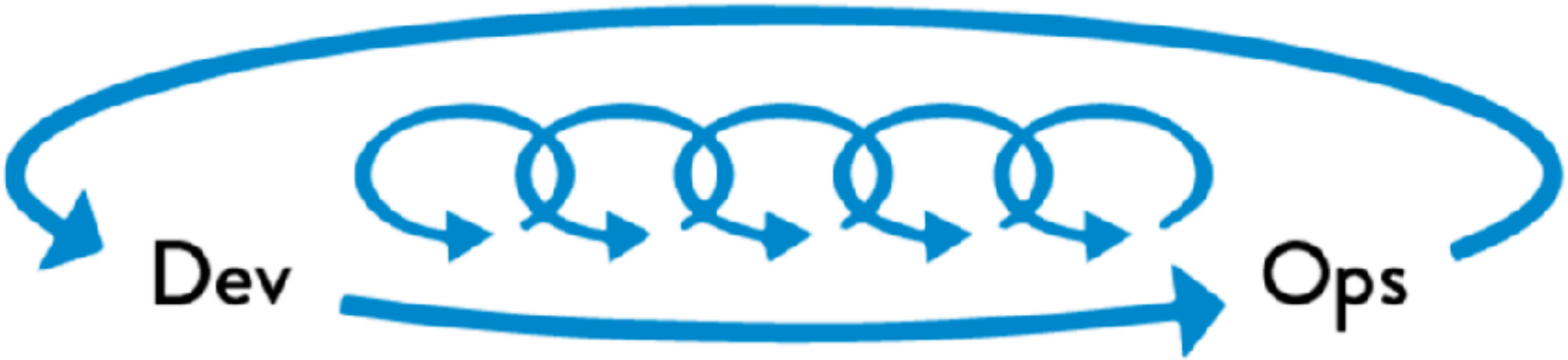
1. Flow principle

2. Feedback principle (1)

2. Feedback principle (2)



3. Continuous learning principle



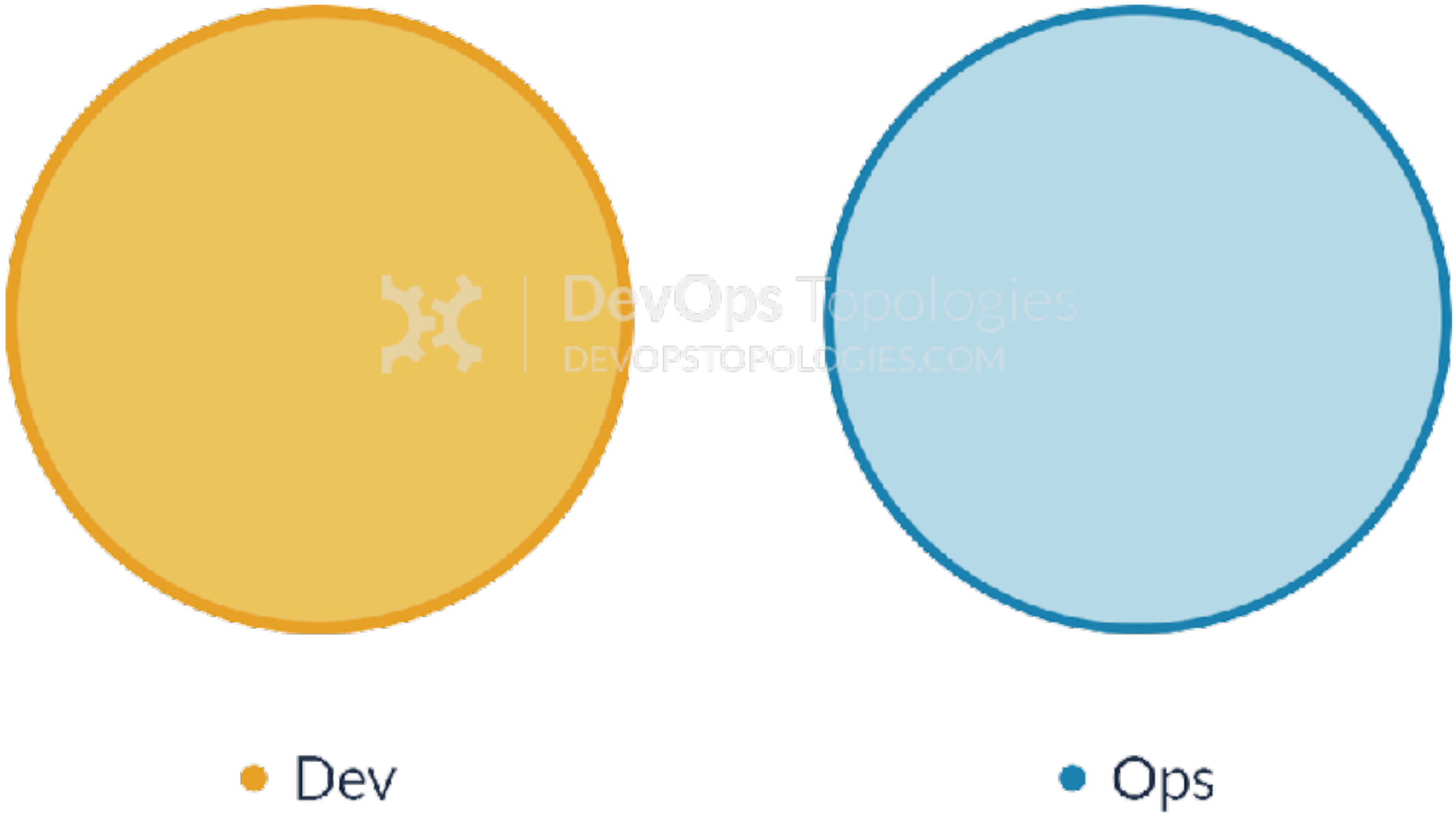
DevOps Topologies

https://web.devopstopologies.com/

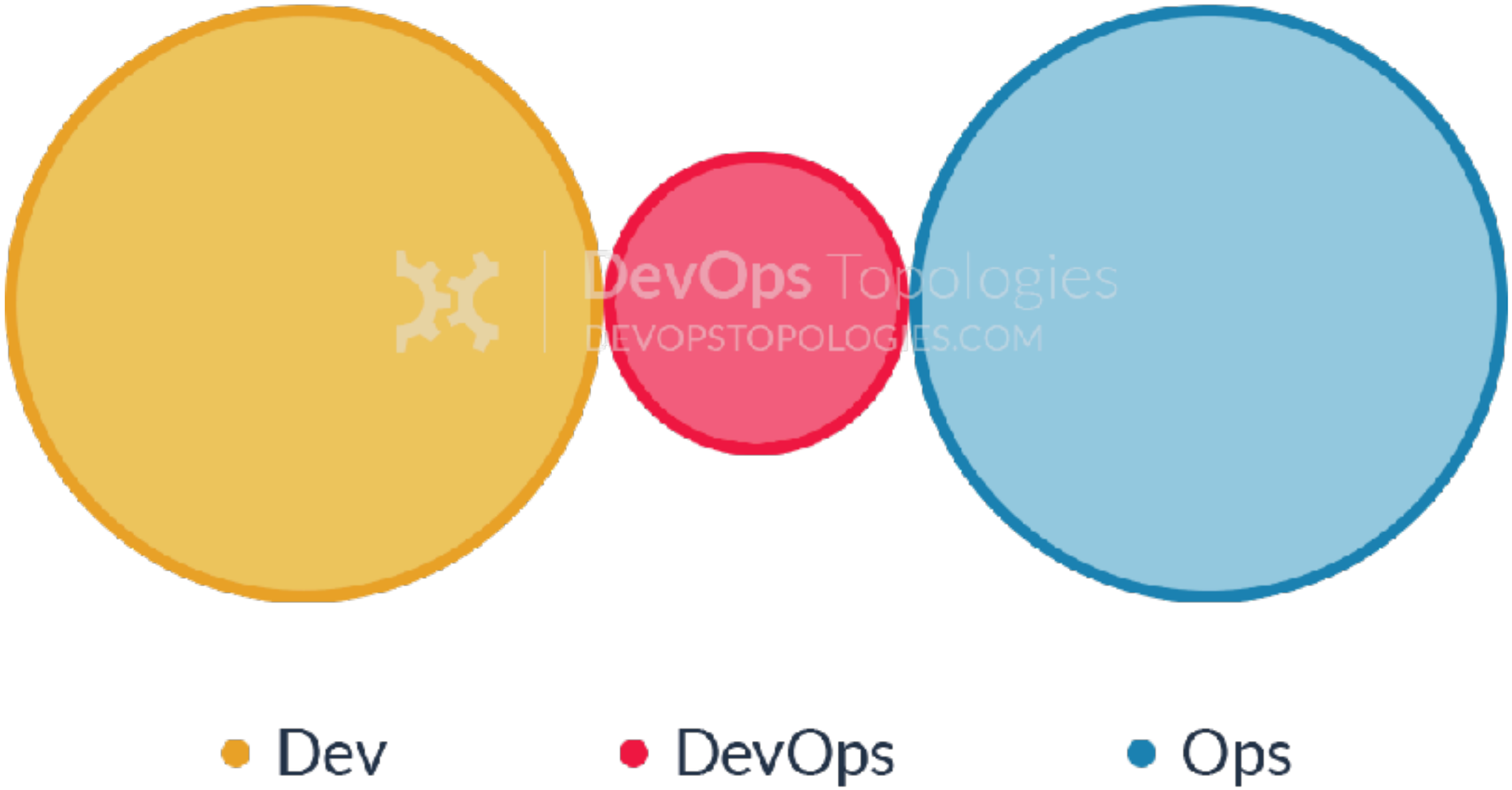
DevOps Anti-Types

https://web.devopstopologies.com/

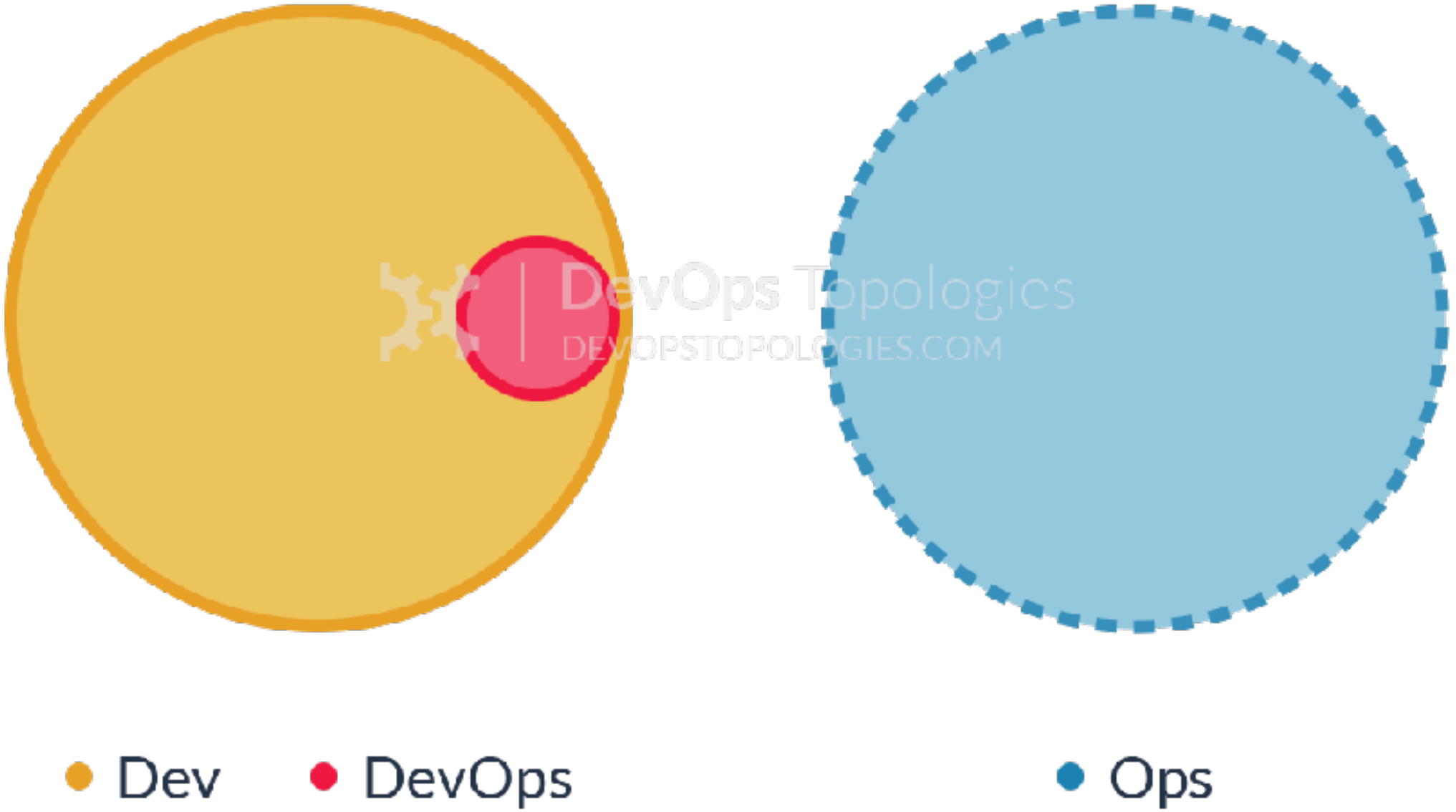
Anti-type A : Dev and Ops silos



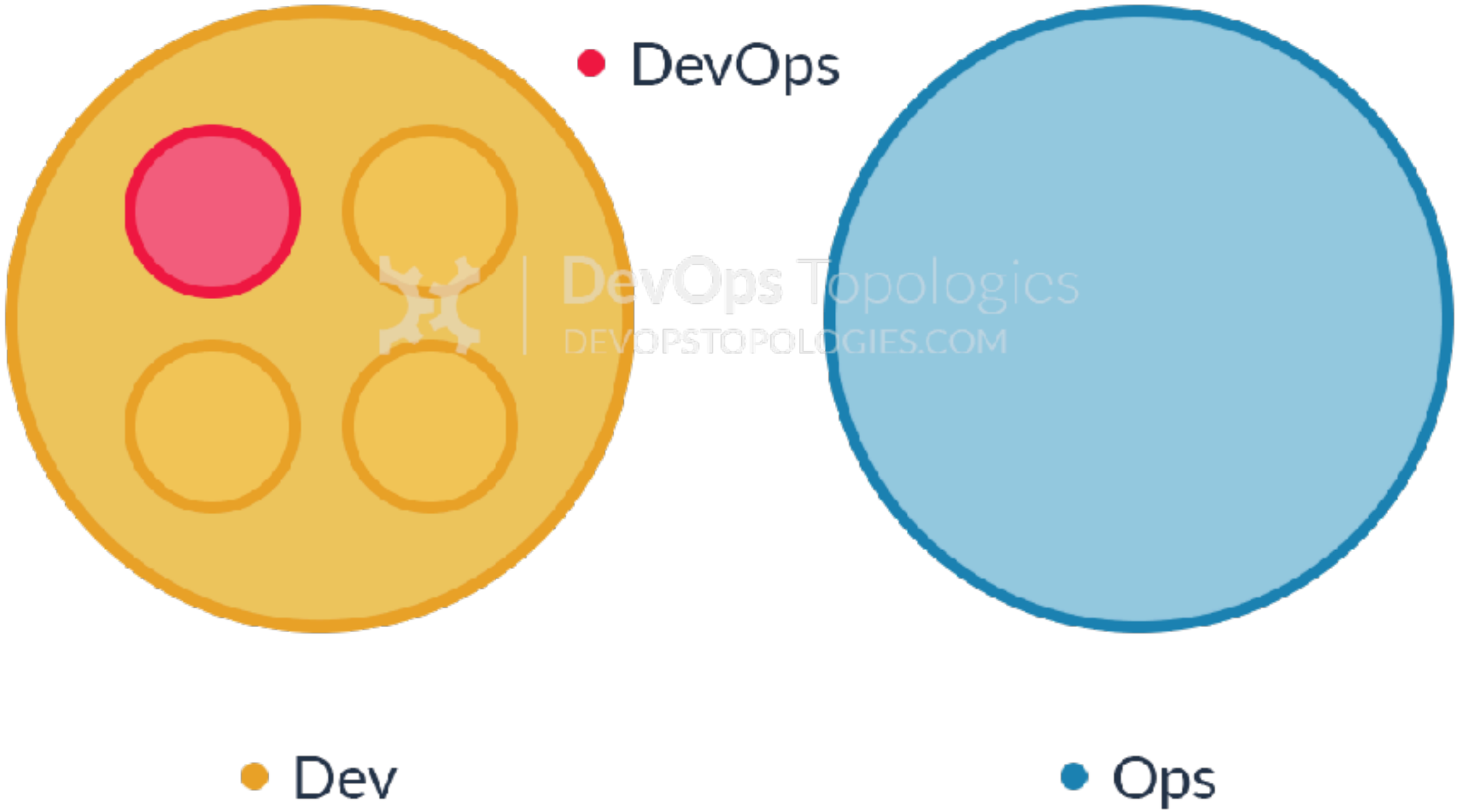
Anti-type B : DevOps team silos



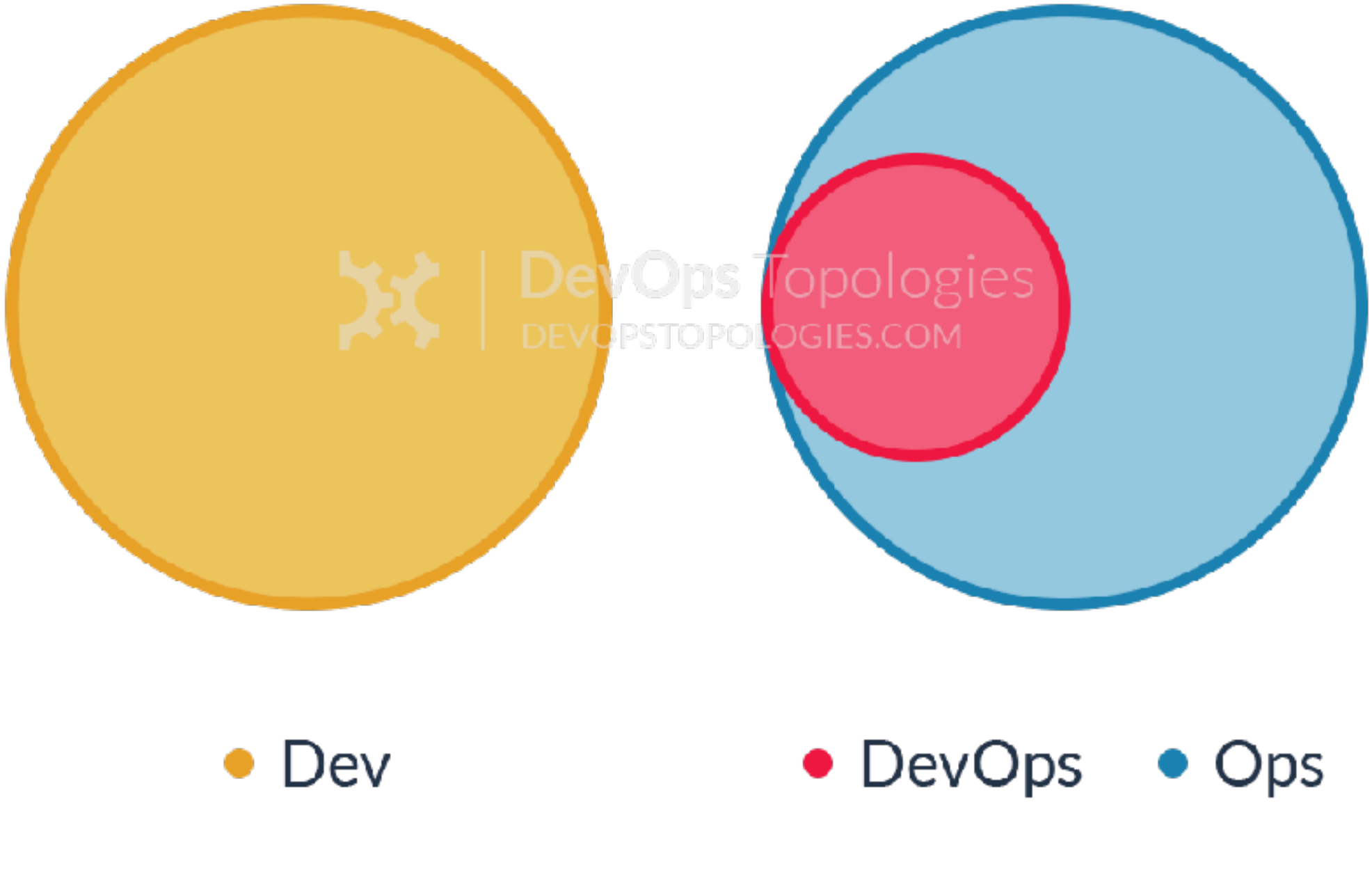
Anti-type C : Dev don’t need Ops



Anti-type D : DevOps as a tool team



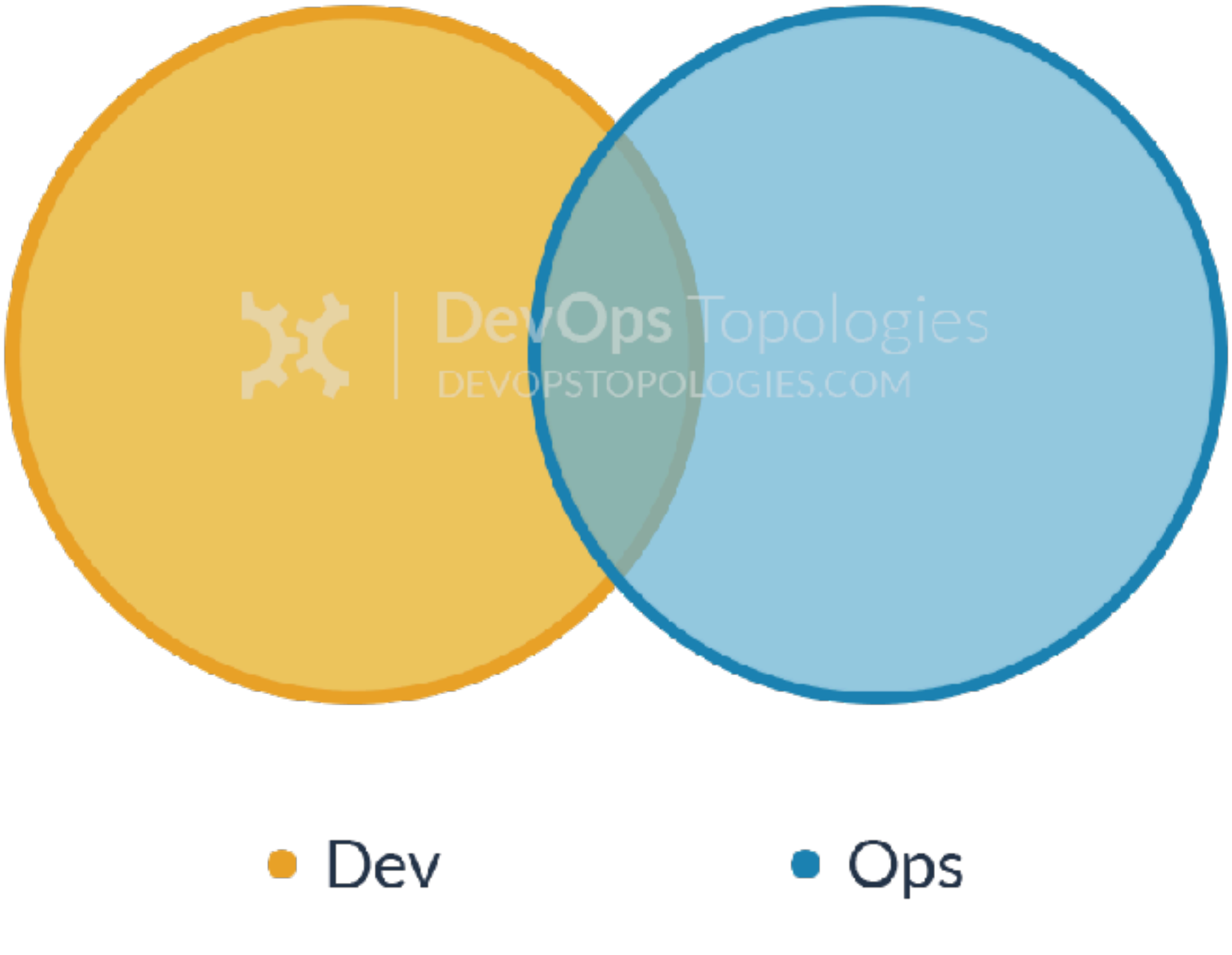
Anti-type E : Rebranded SysAdmin



DevOps Team Topologies

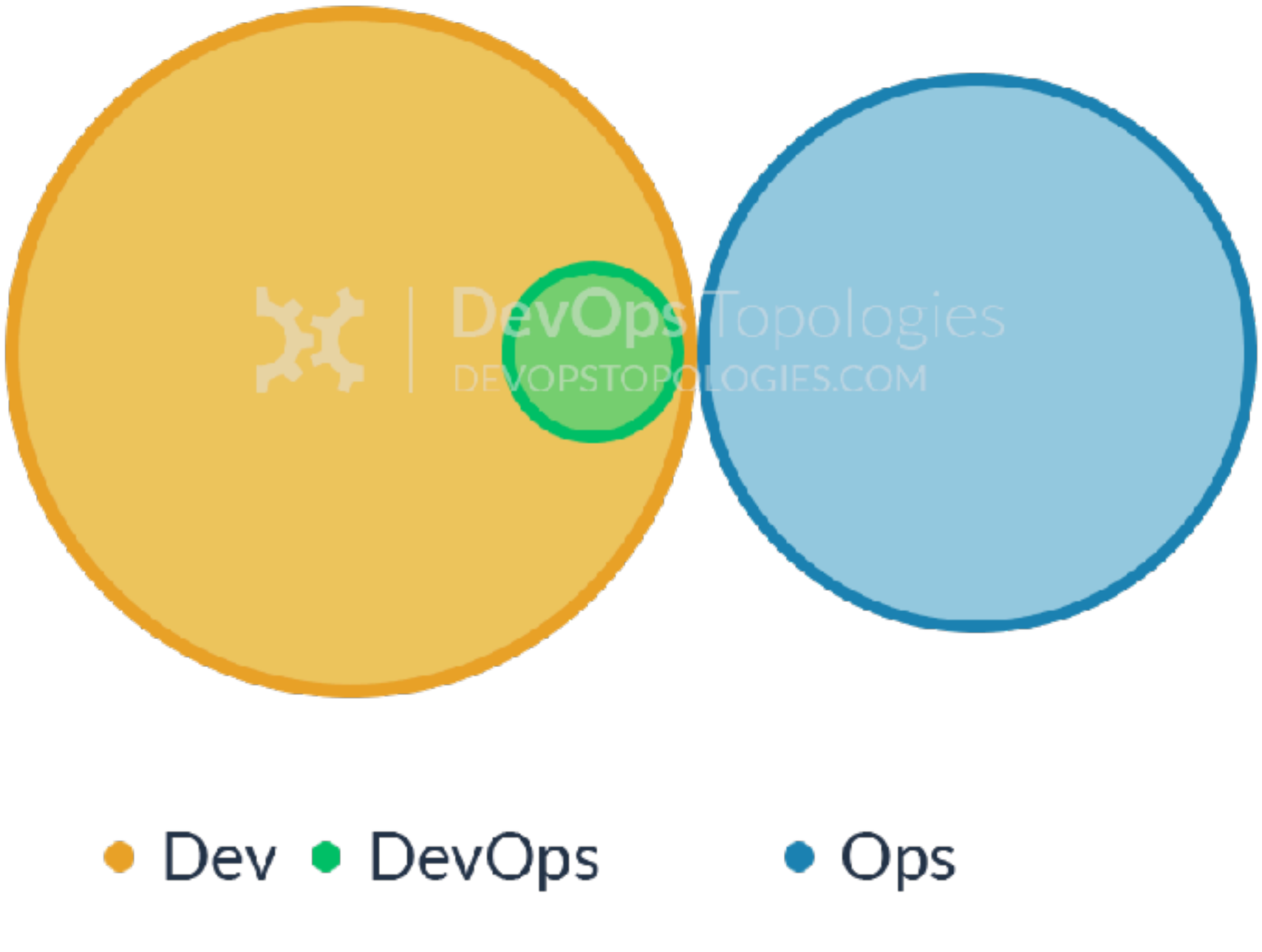
https://web.devopstopologies.com/

Type 1 : Dev and Ops collaboration

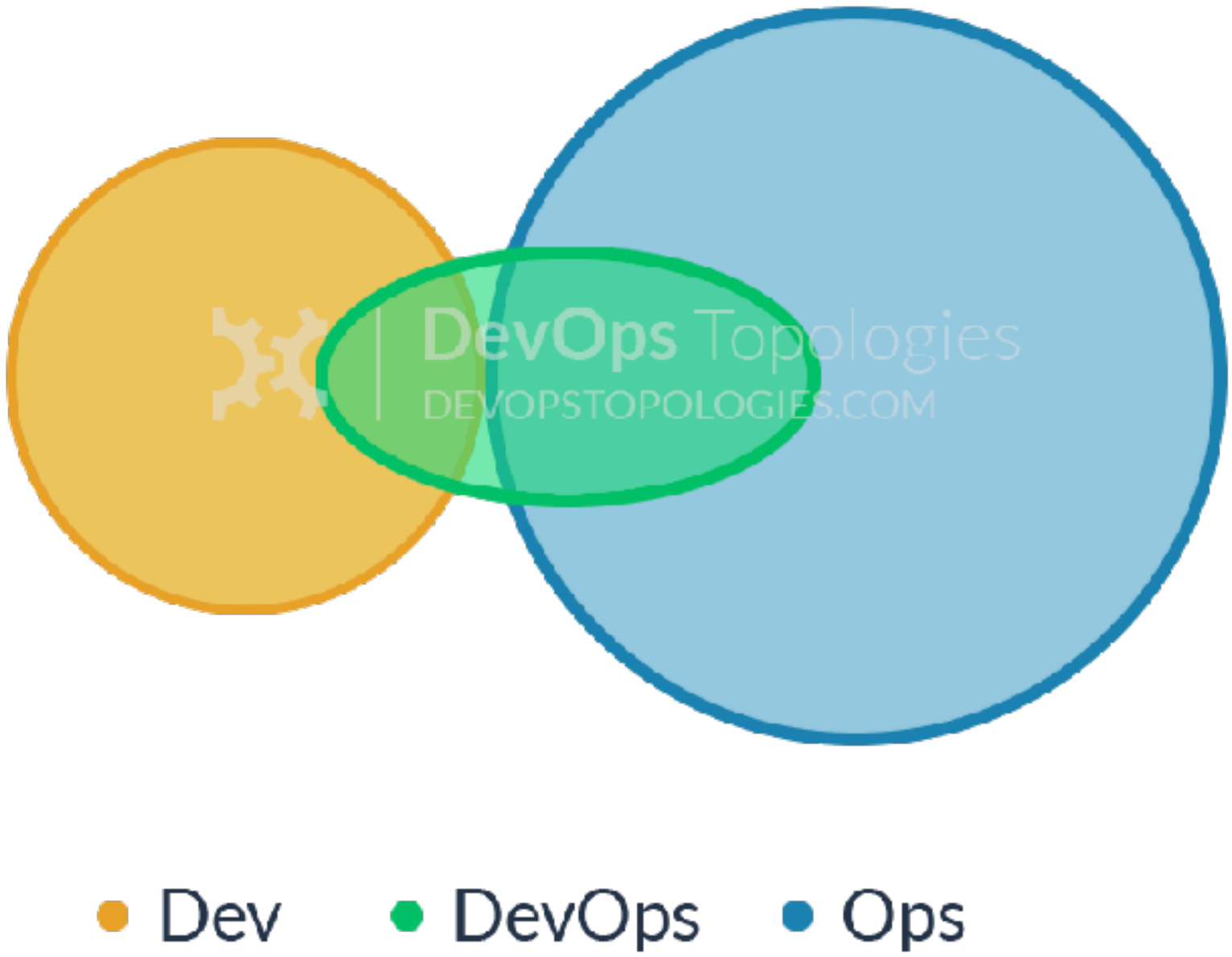


Type 2 : Fully share ops responsibility

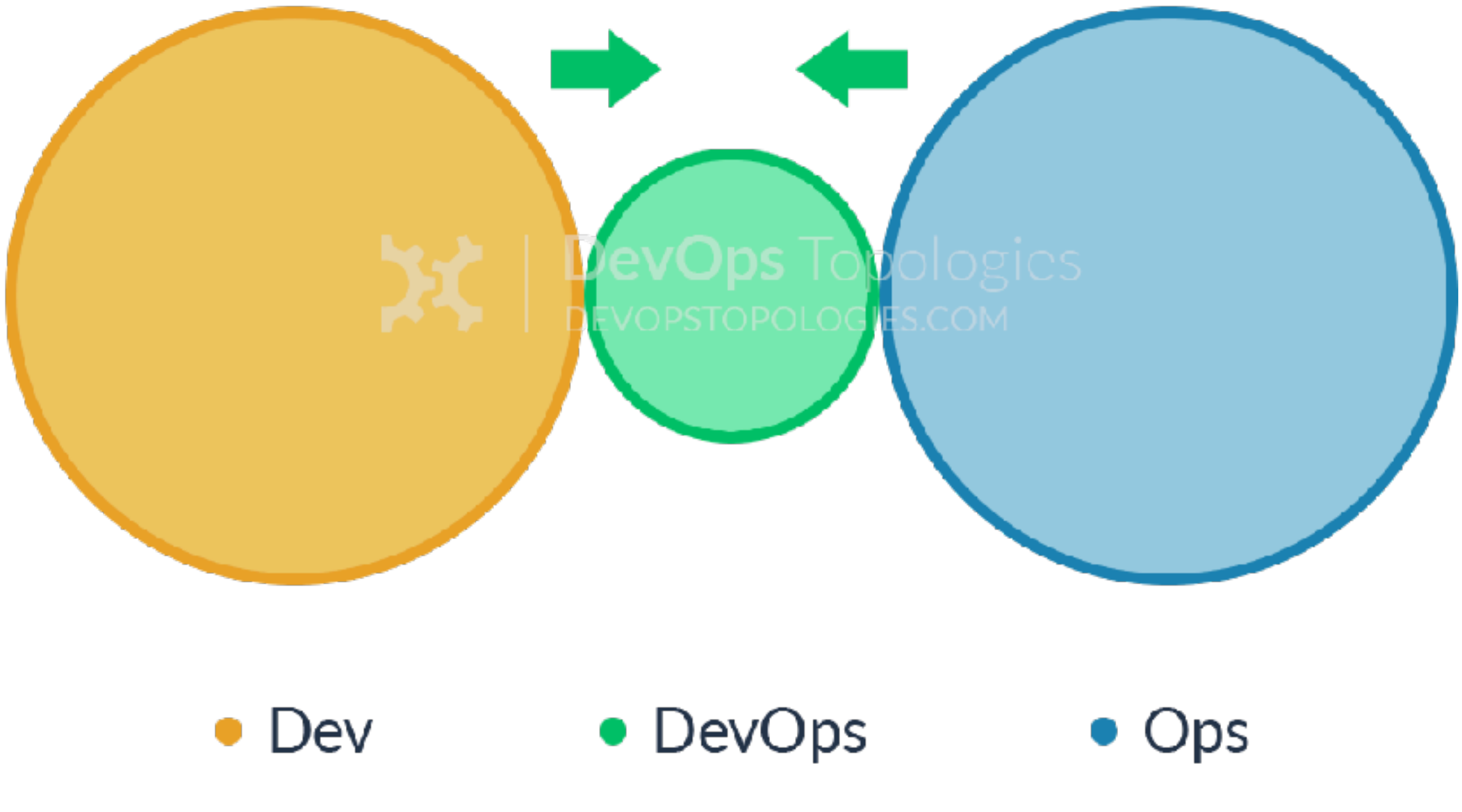
Type 3 :Ops as infra as a service



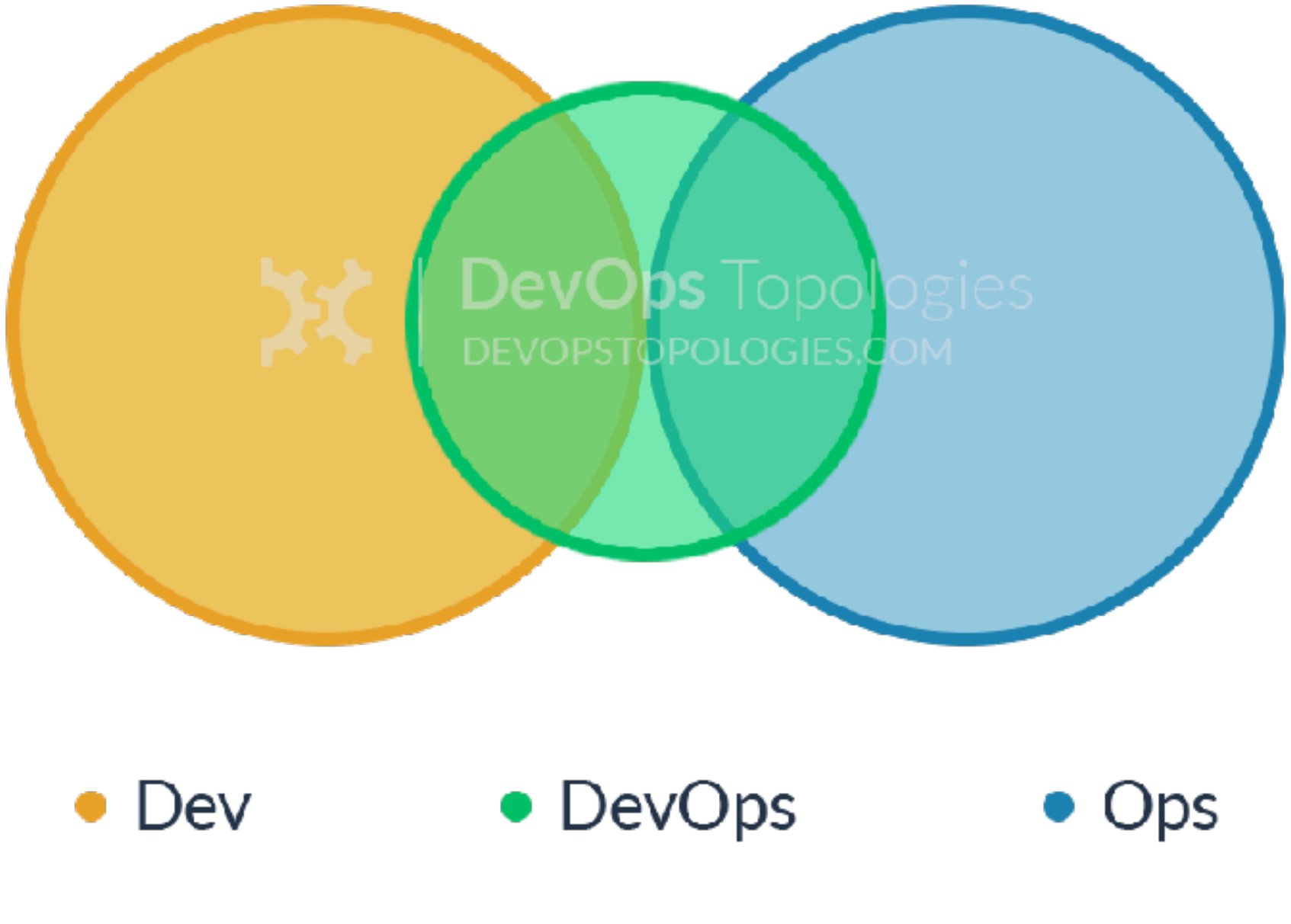
Type 4 : DevOps as external service



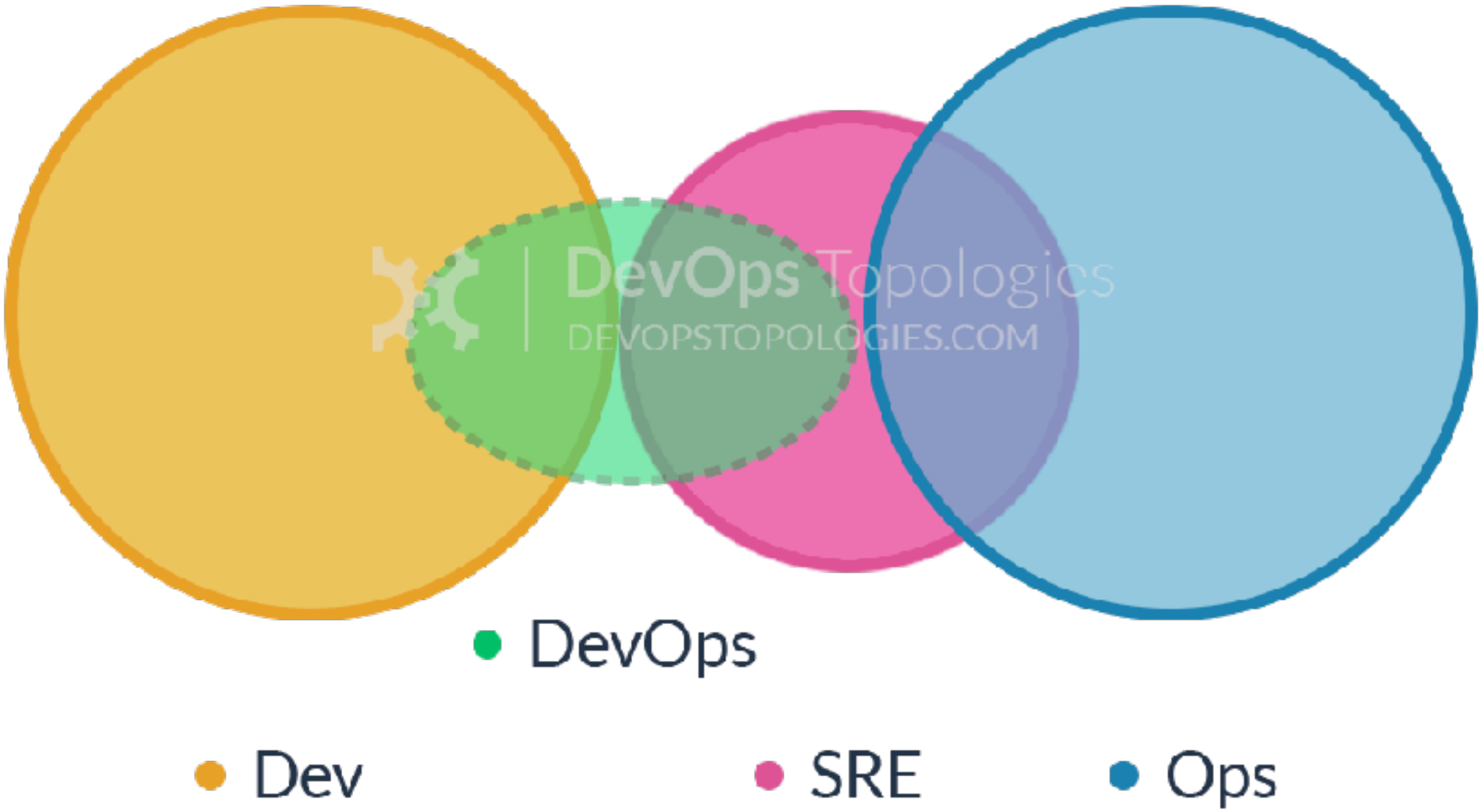
Type 5 : DevOps team with expire date



Type 6 : DevOps evangelist team

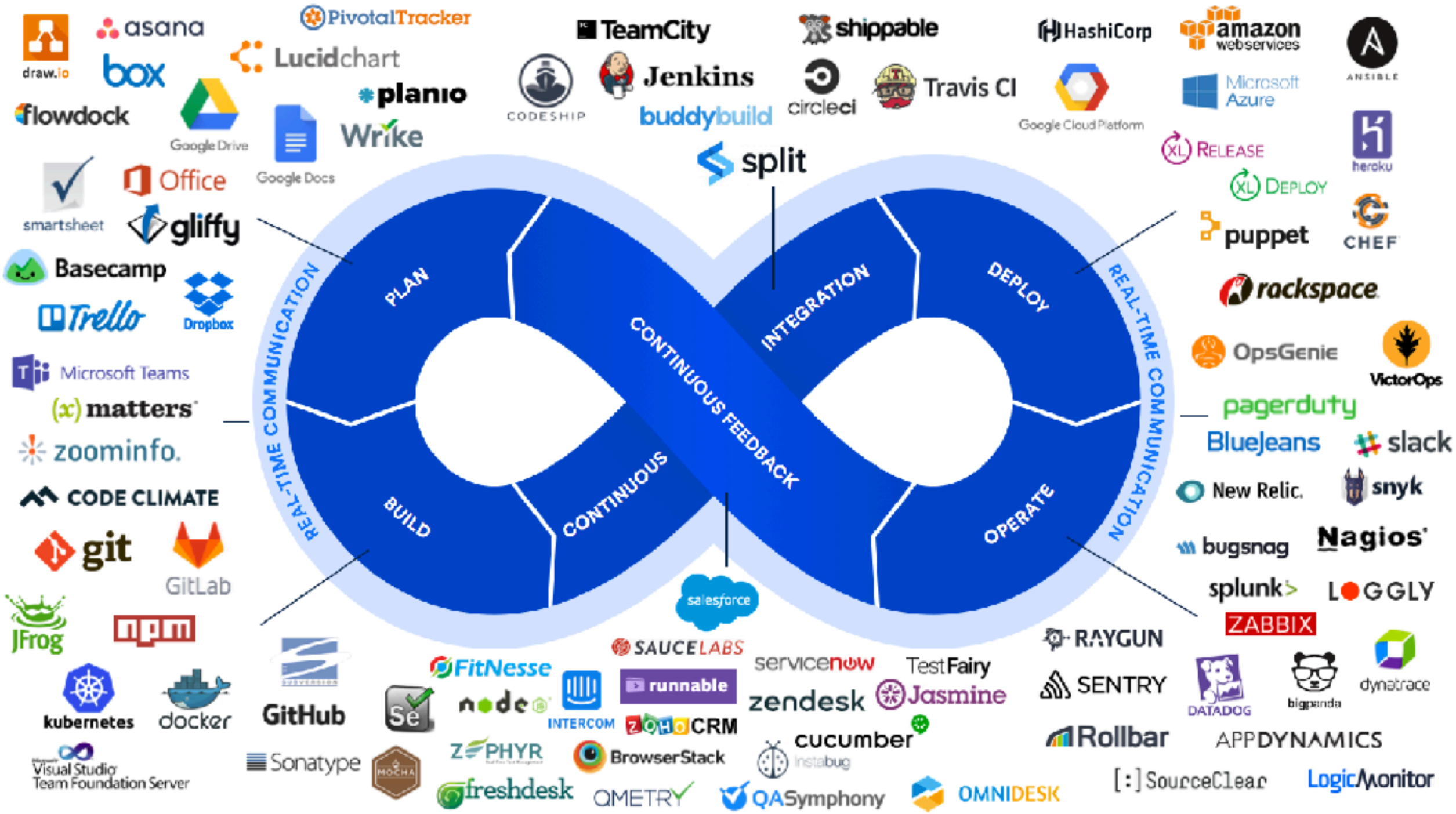


Type 7 : SRE team (Google model)

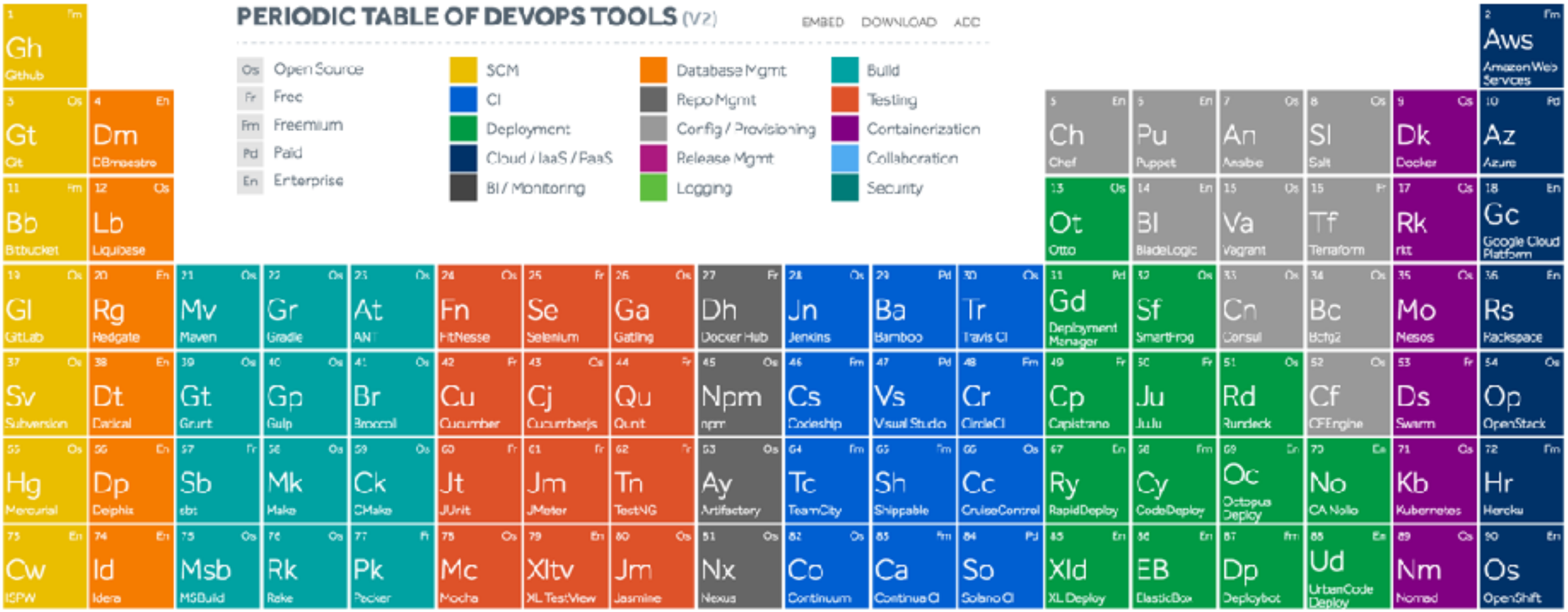


DevOps Tools

DevOps Tools



DevOps Tools



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

No DevOps Team

Problem department !!

DevOps != Tools

Tools enable DevOps

DevOps success ?

How do i know something is wrong ?

Missed deadline

Site is always down

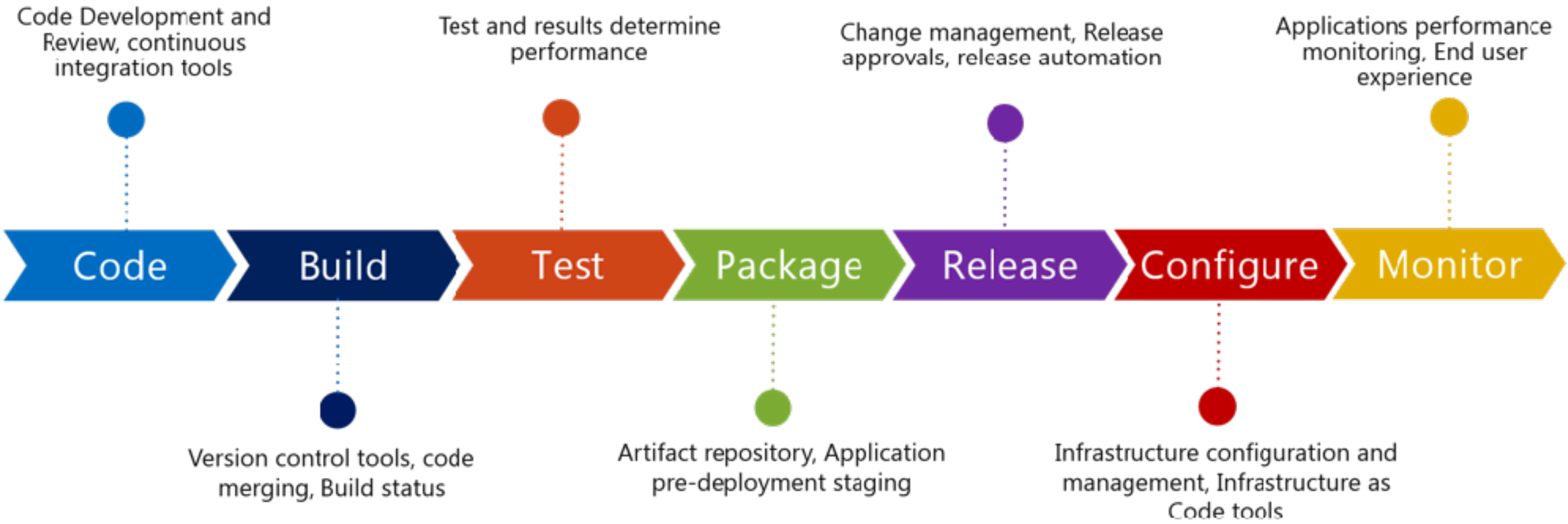
Unhappy customers

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

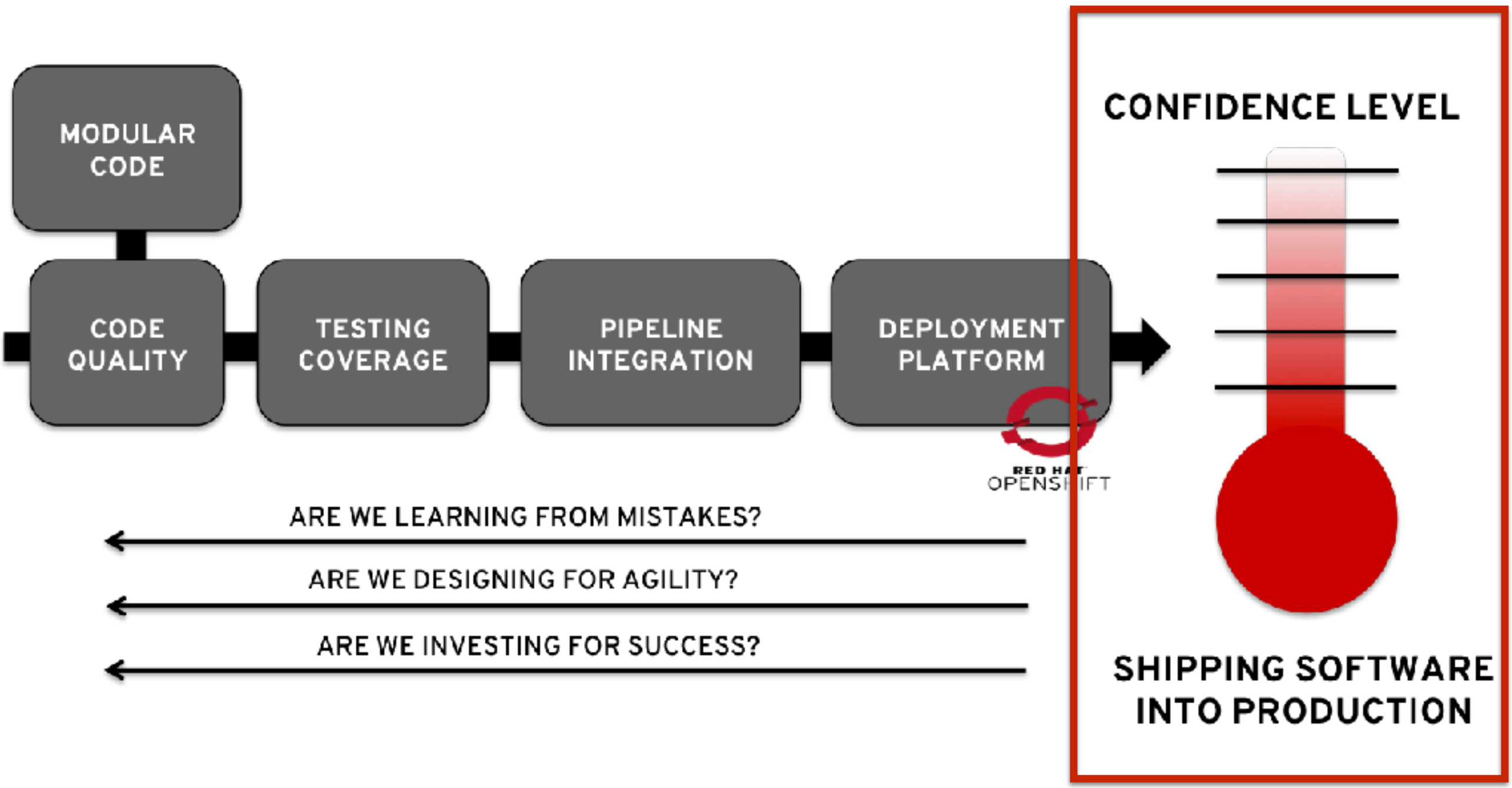
Automated Test Pass (%)

Defect Rate

Failed Deployments

Availability

What can i measure ?



Three must haves

Log aggregations

Stream all logs into one place.

Circuit breakers

Write code with failure in mind

Request tracing

Don’t spend hours debugging

https://trello.com/

Circuit breakers

Response times

How much time do services spend calling other services.

Back pressure

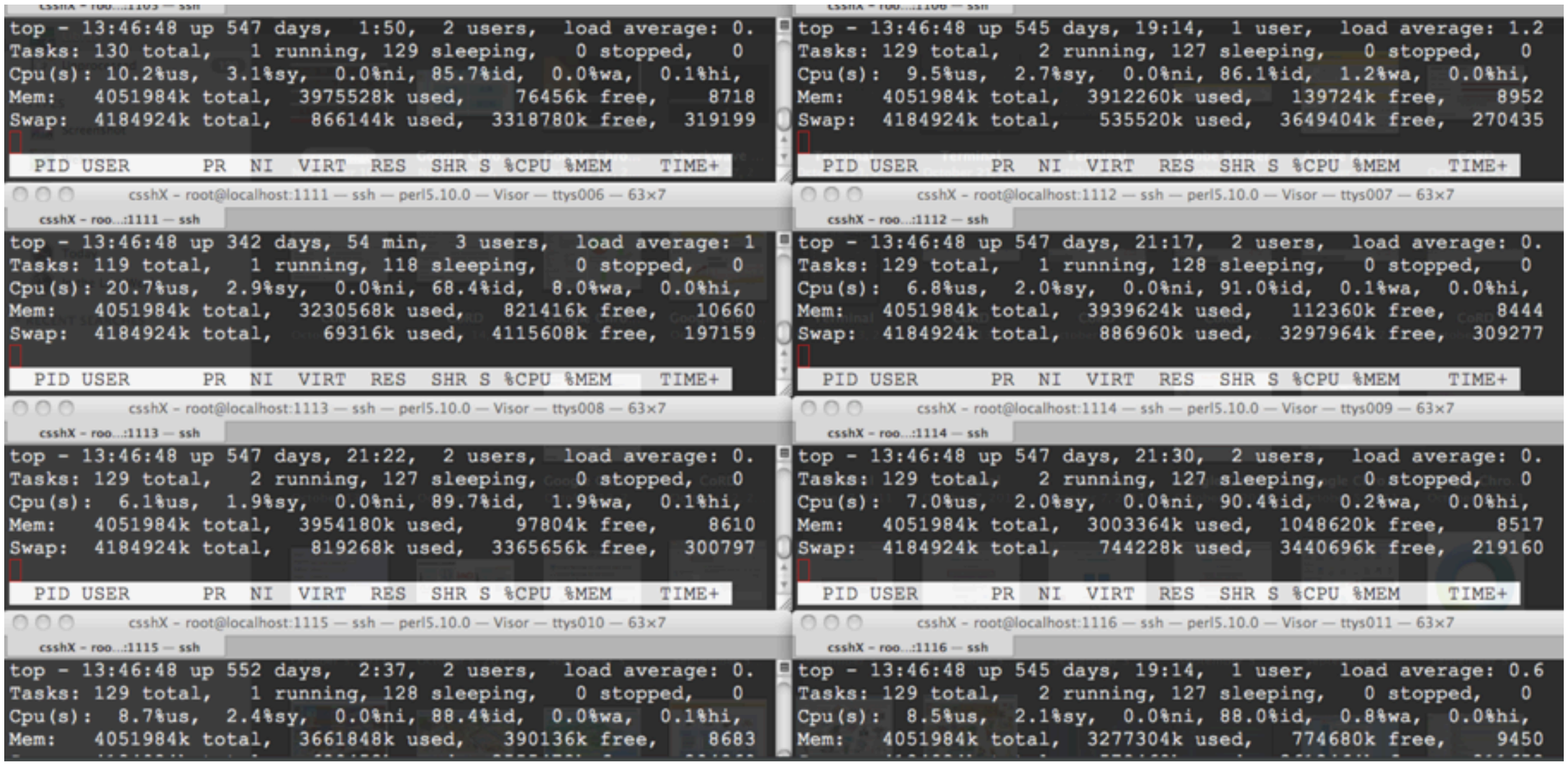
Stop putting pressure on a system that is in trouble and fail fast

Fallback

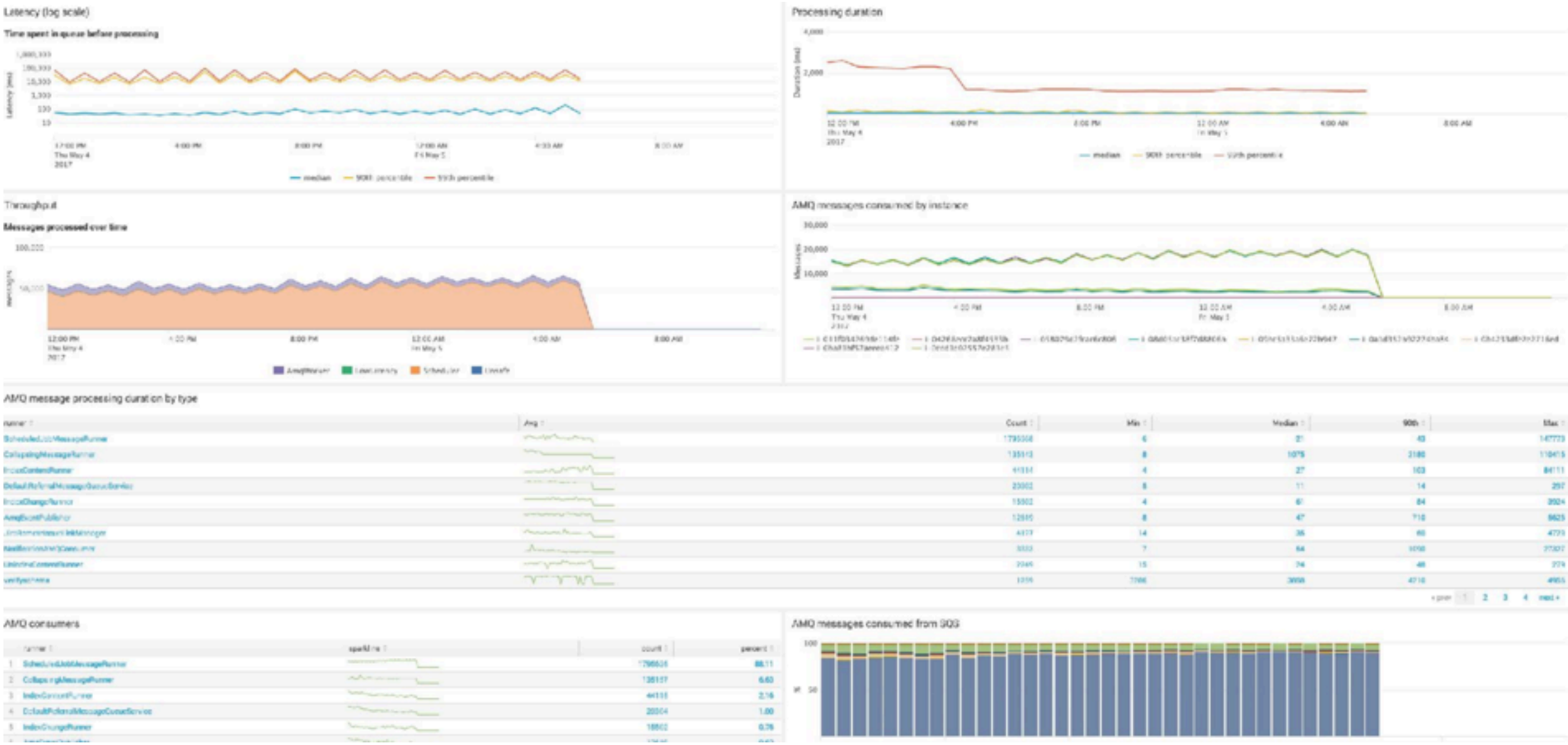
How do you handle failure. A mandatory step in the programming model.

https://trello.com/

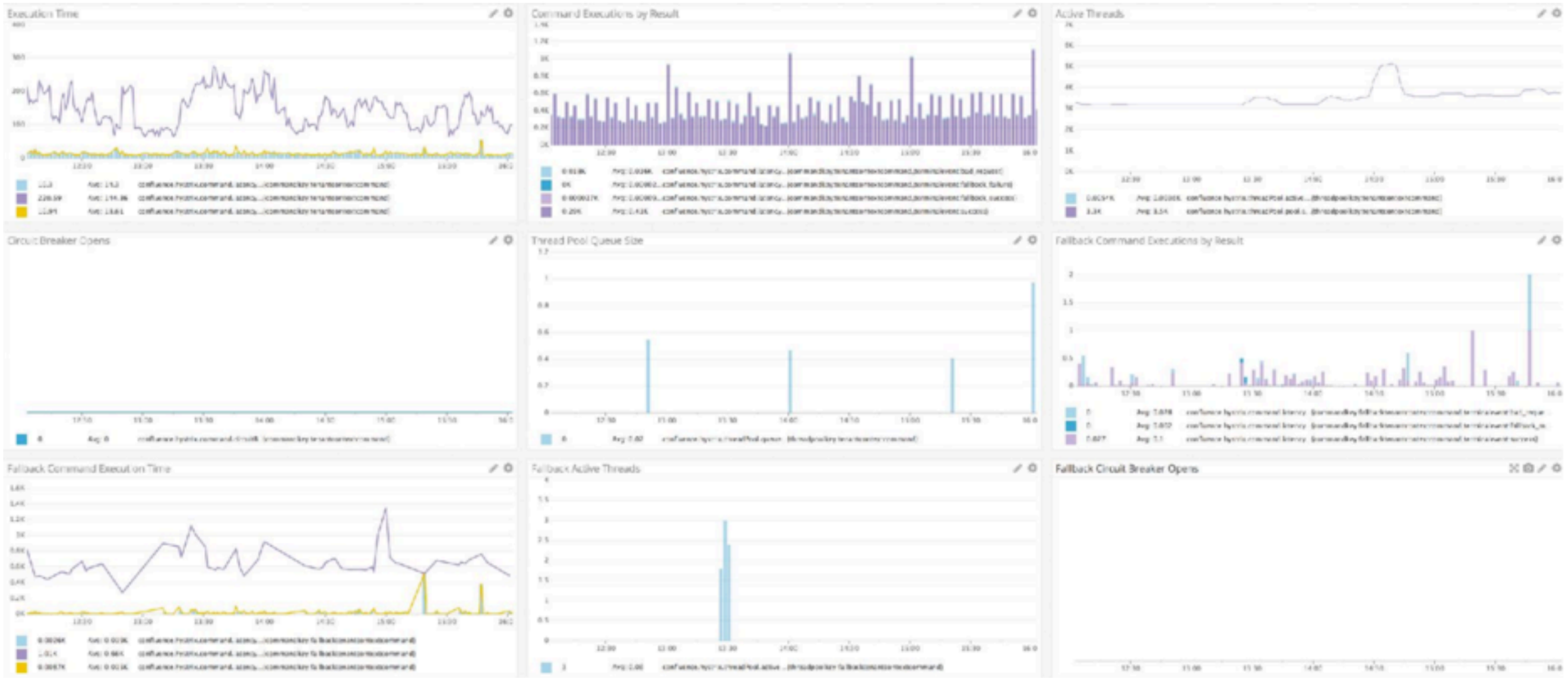
Do you know your system ?



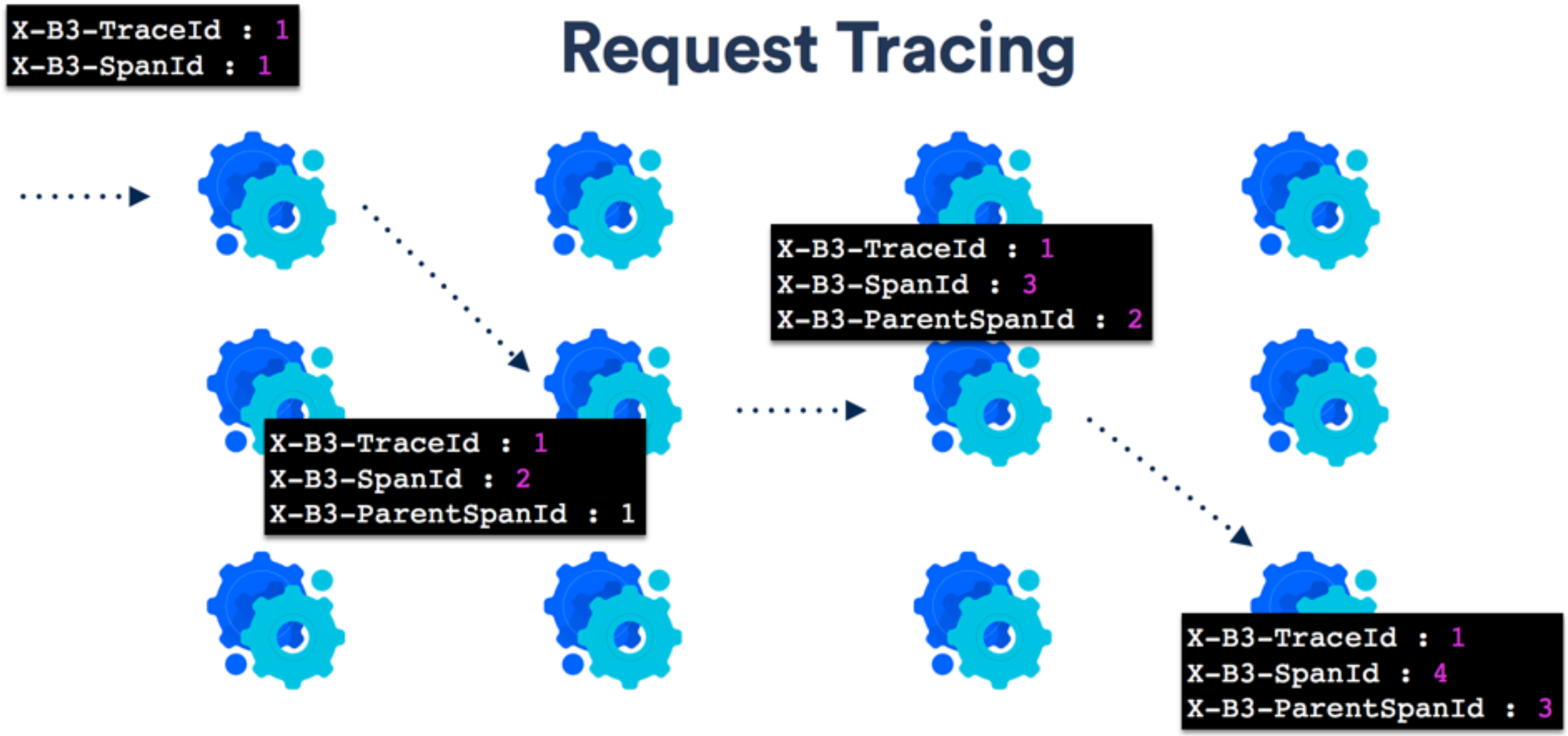
Do you know your system ?



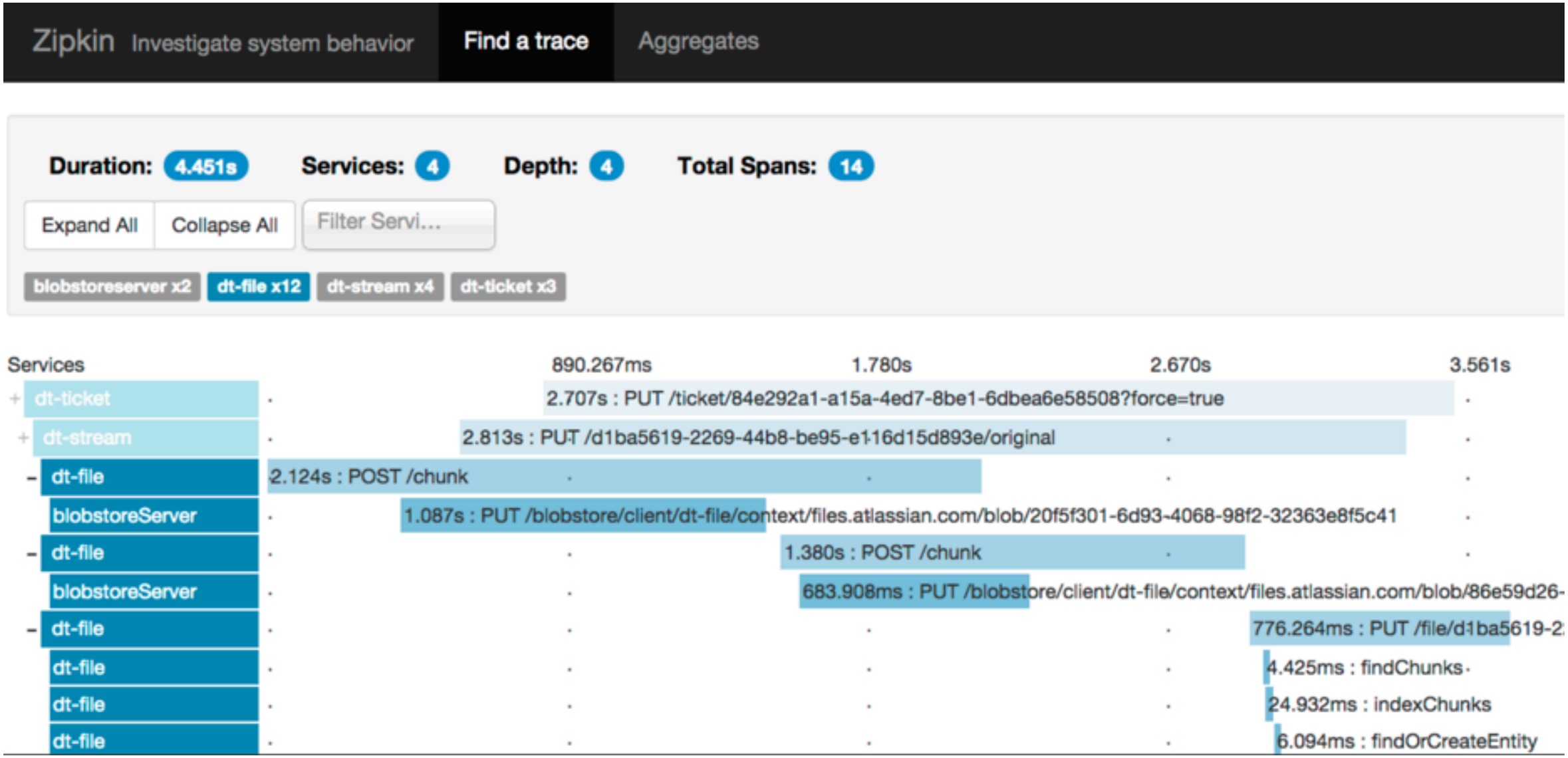
Do you know your system ?



Do you know your system ?



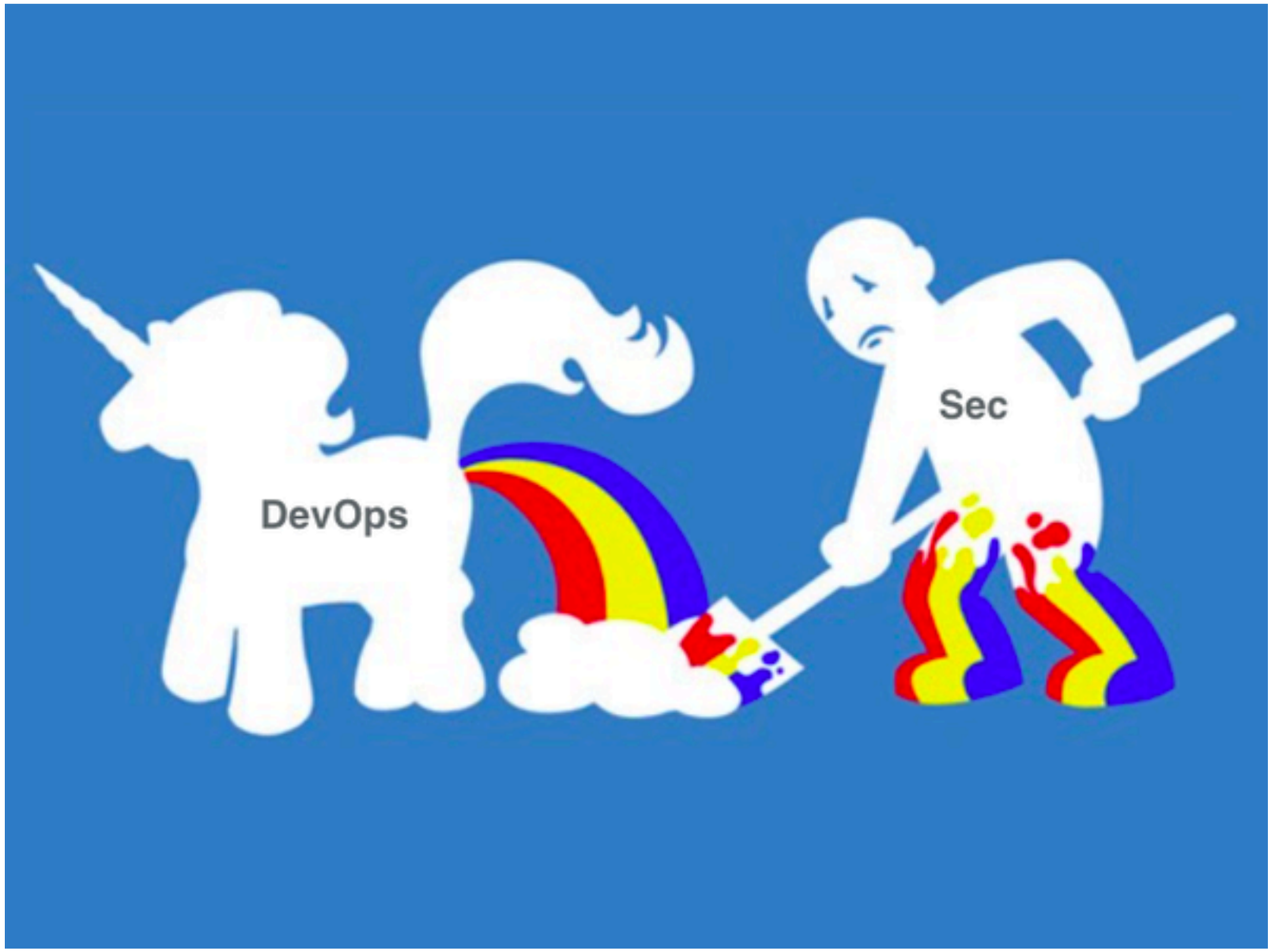
Do you know your system ?

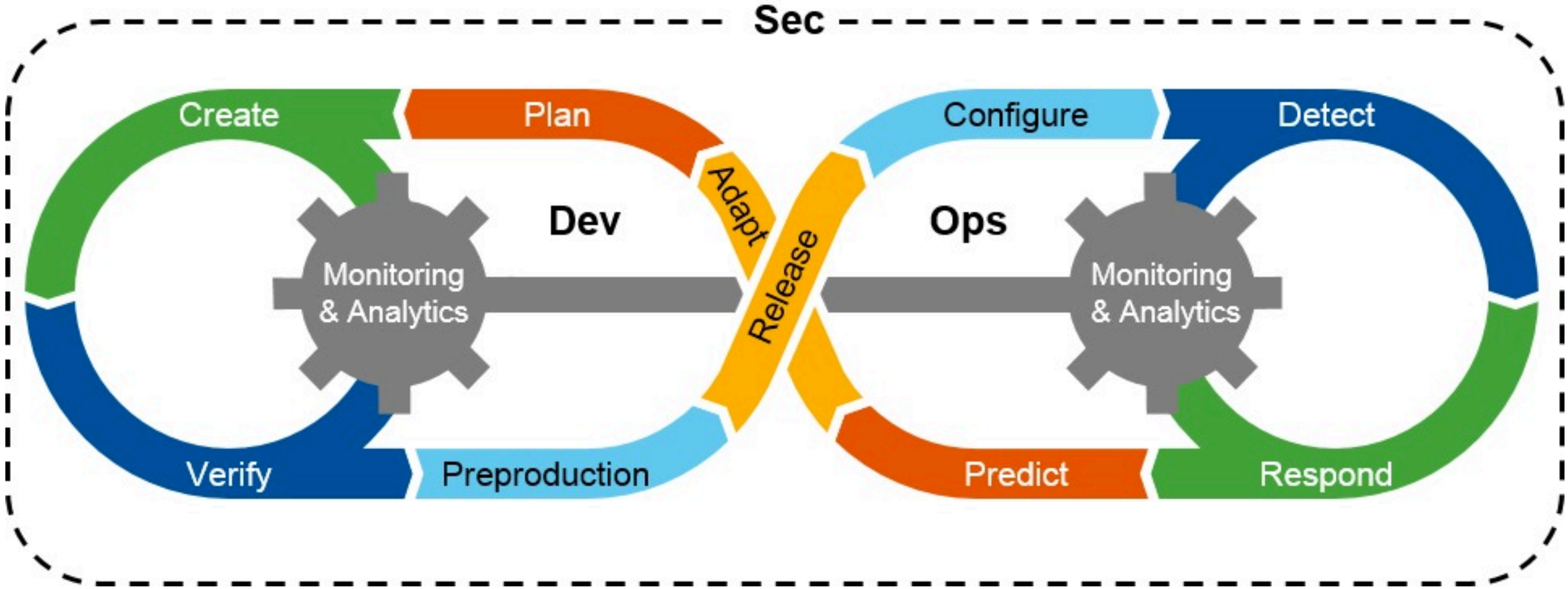


You Build It You Run It

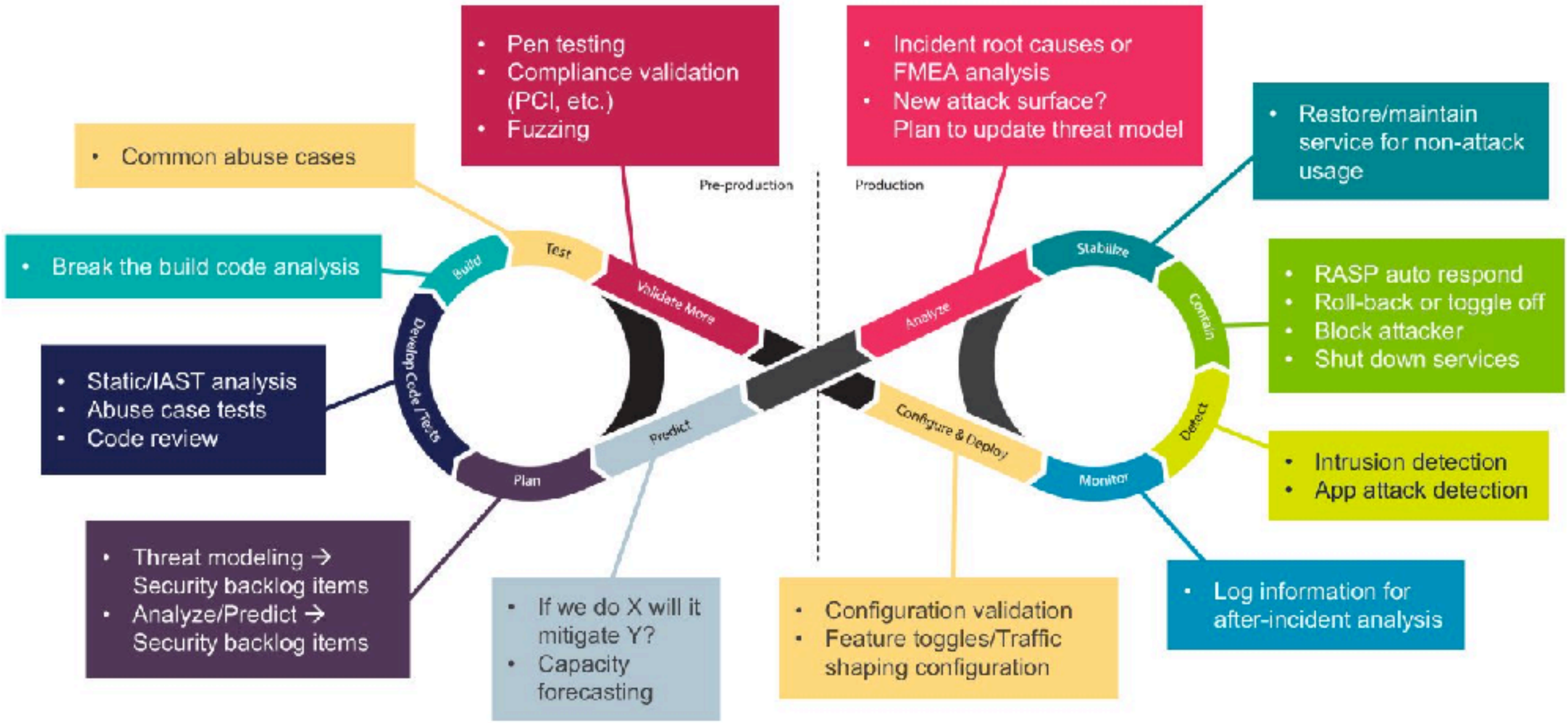
The team who builds it looks after it.

more ...

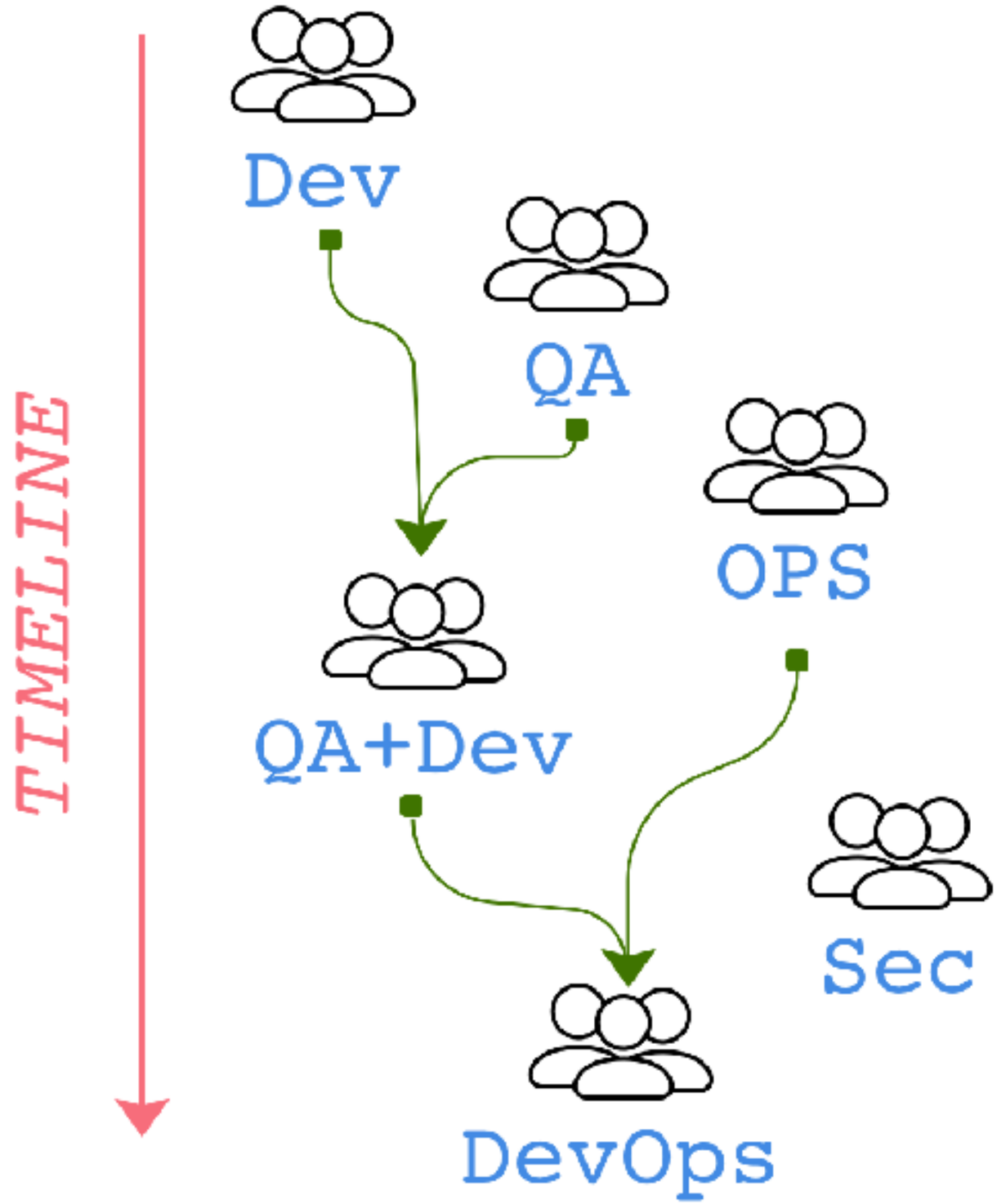




DevSecOps Cycle

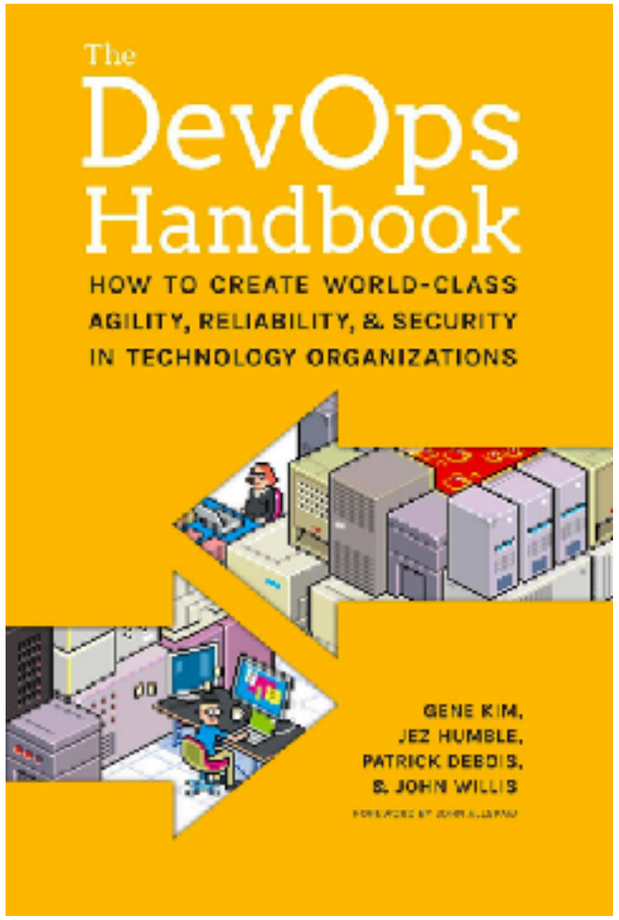


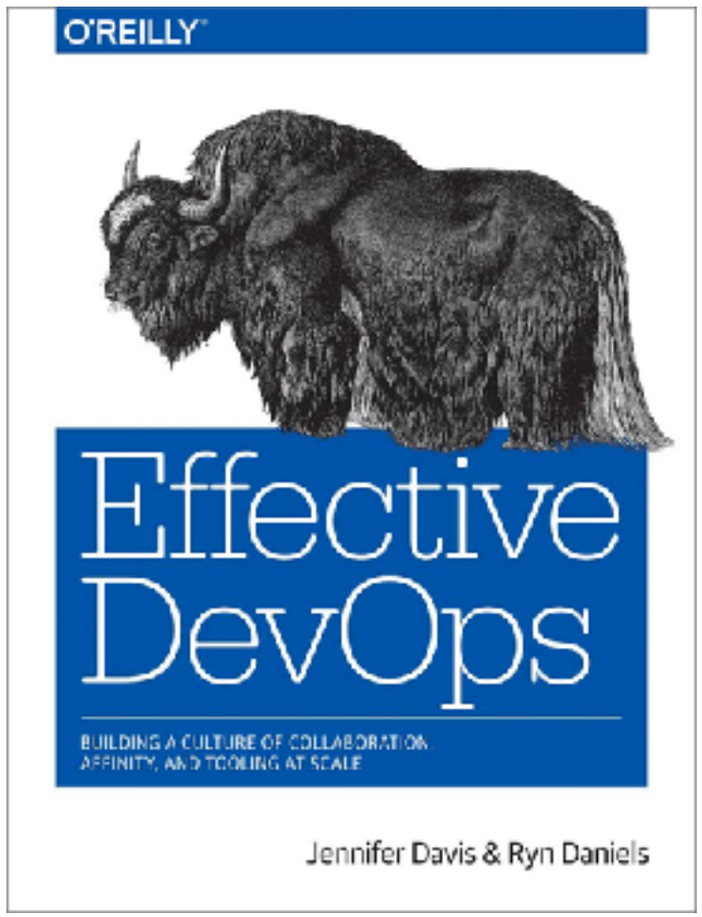
LinkedIn.com/in/LarryMaccherone

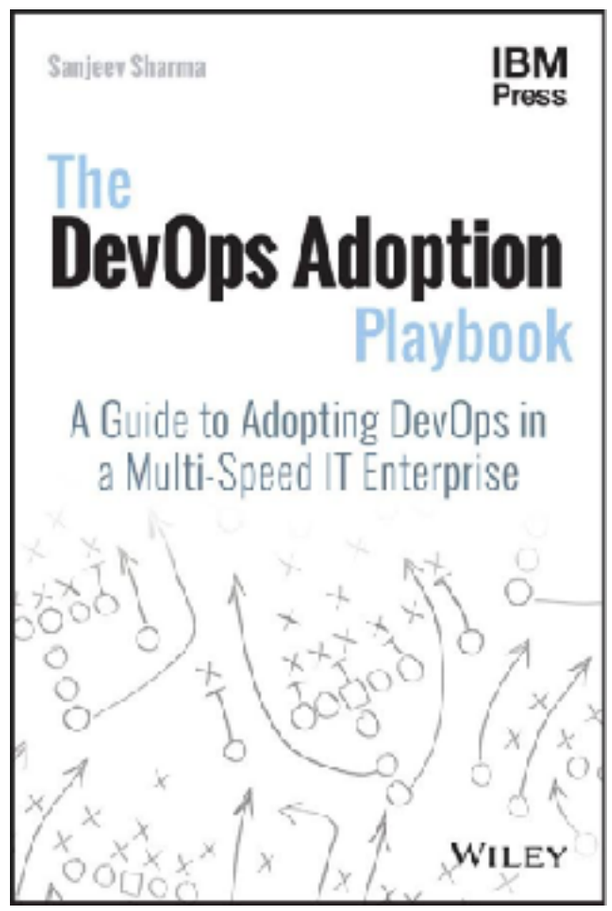


https://medium.com/odds-team/%E0%B8%AA%E0%B8%A3%E0%B8%B8%E0%B8%9B-devops-is-dev%EA%9D%8B%E1%B5%BD%EA%9E%A9-by-chai-feng-bb61b0460af0

Resources







DevOps !!

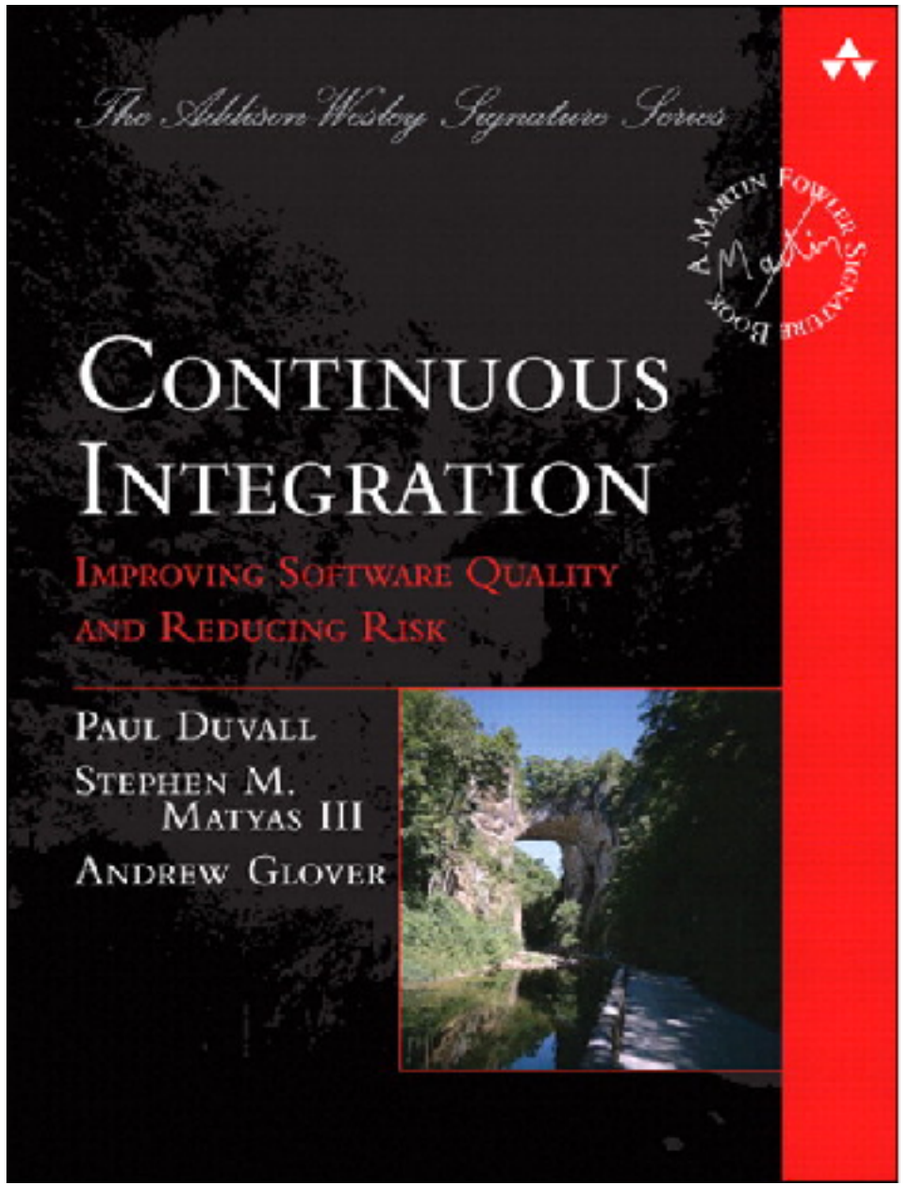
Back to basic

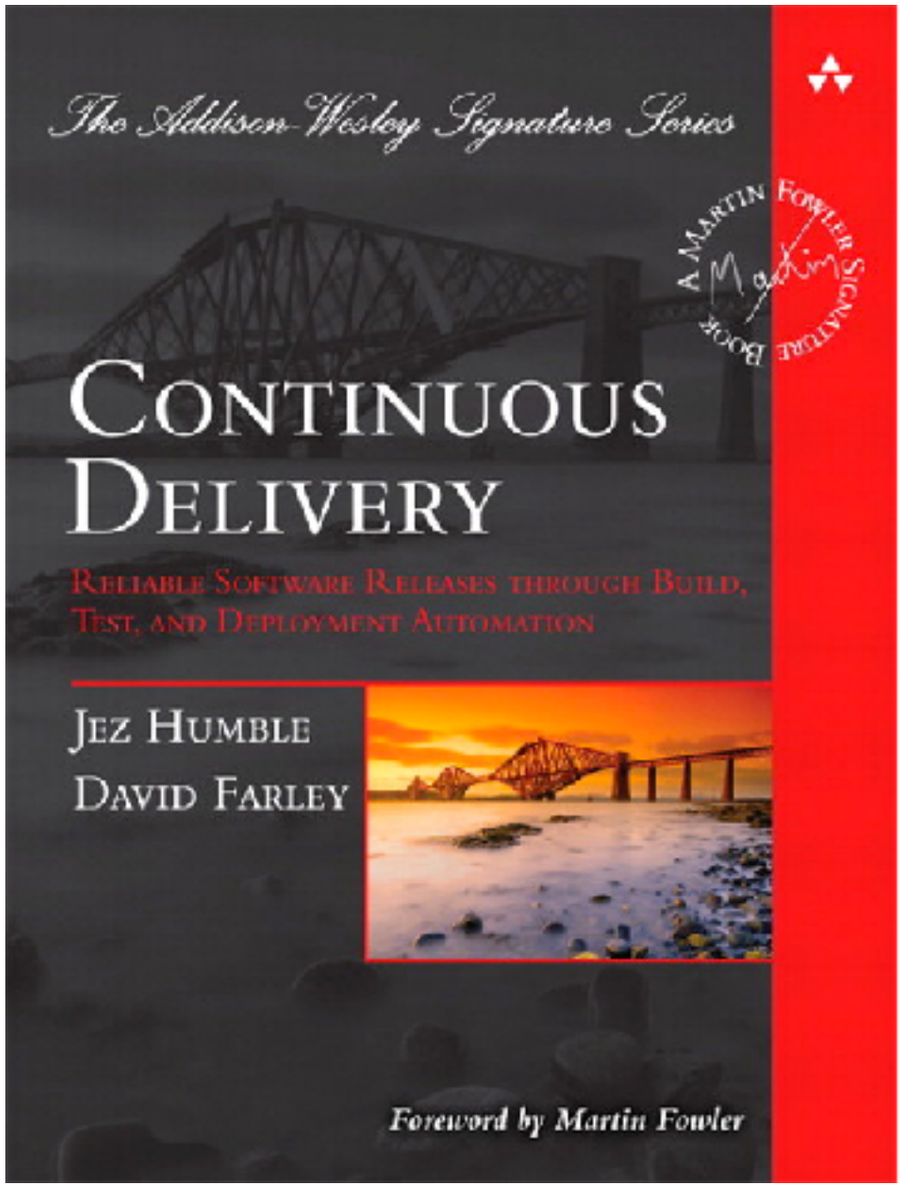
“Behind every successful agile project, there is a

Continuous Integration”

Start with Continuous Integration Continuous Delivery

Improve quality and reduce risk

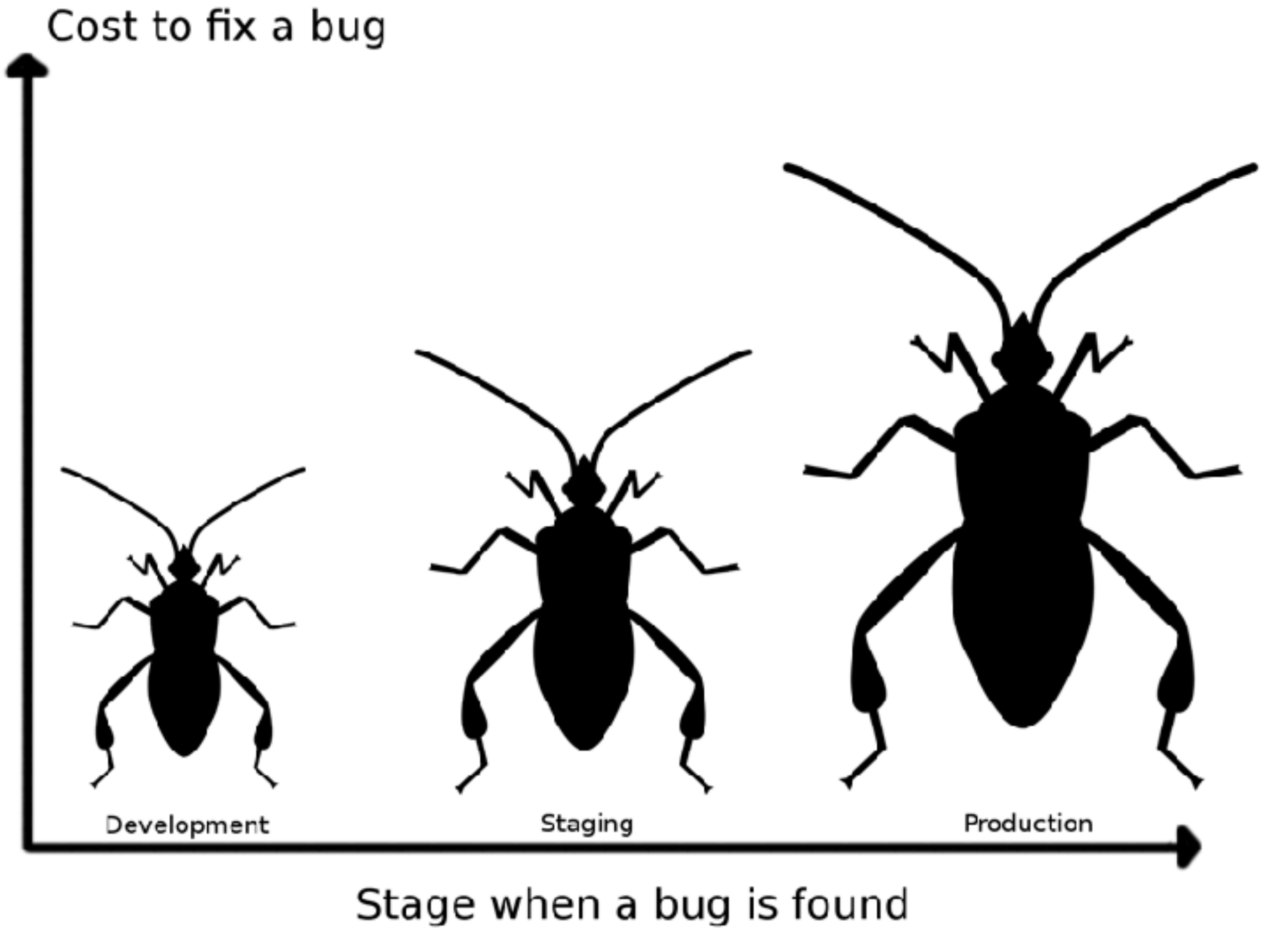


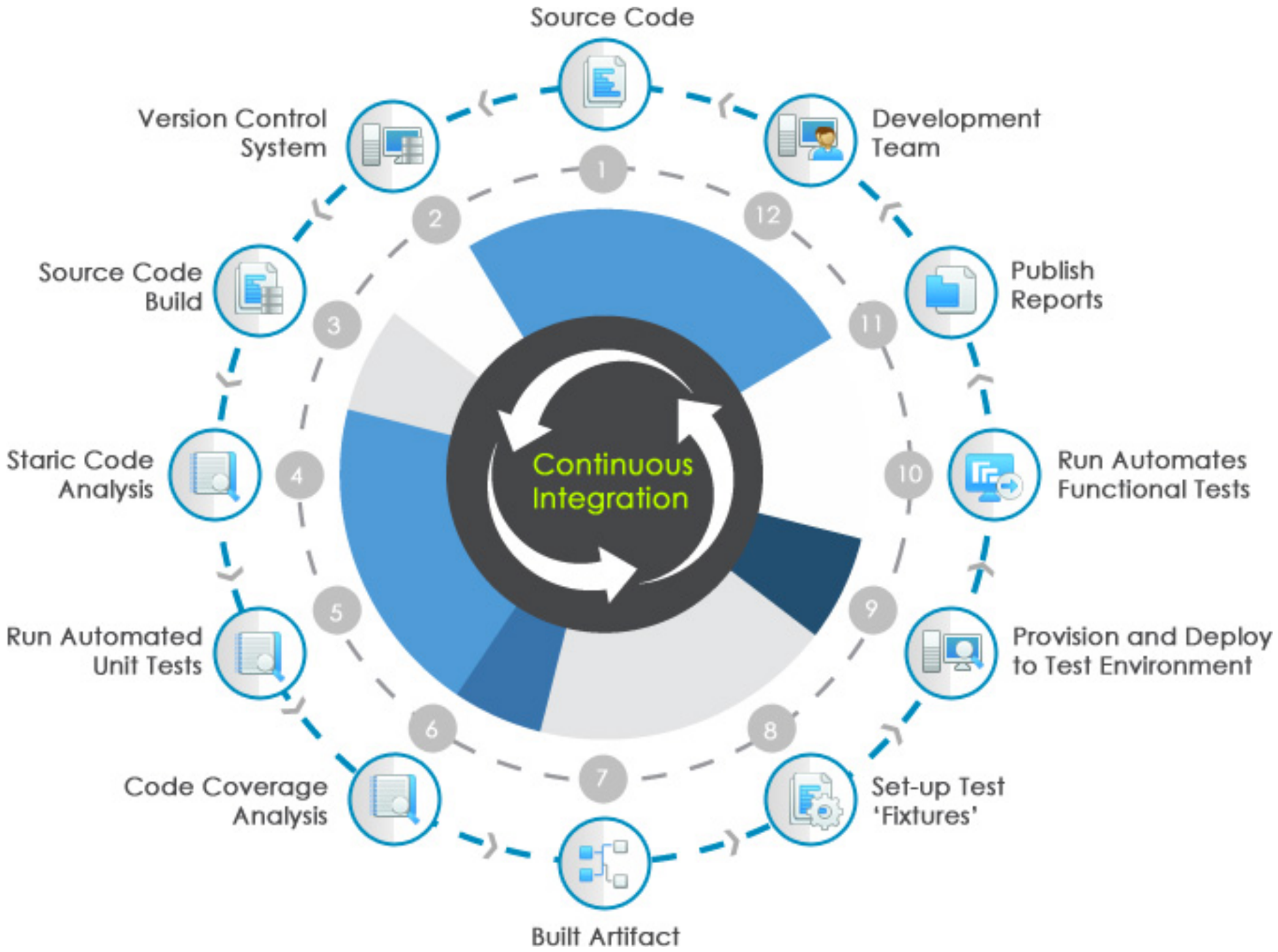


The cost of integration

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**

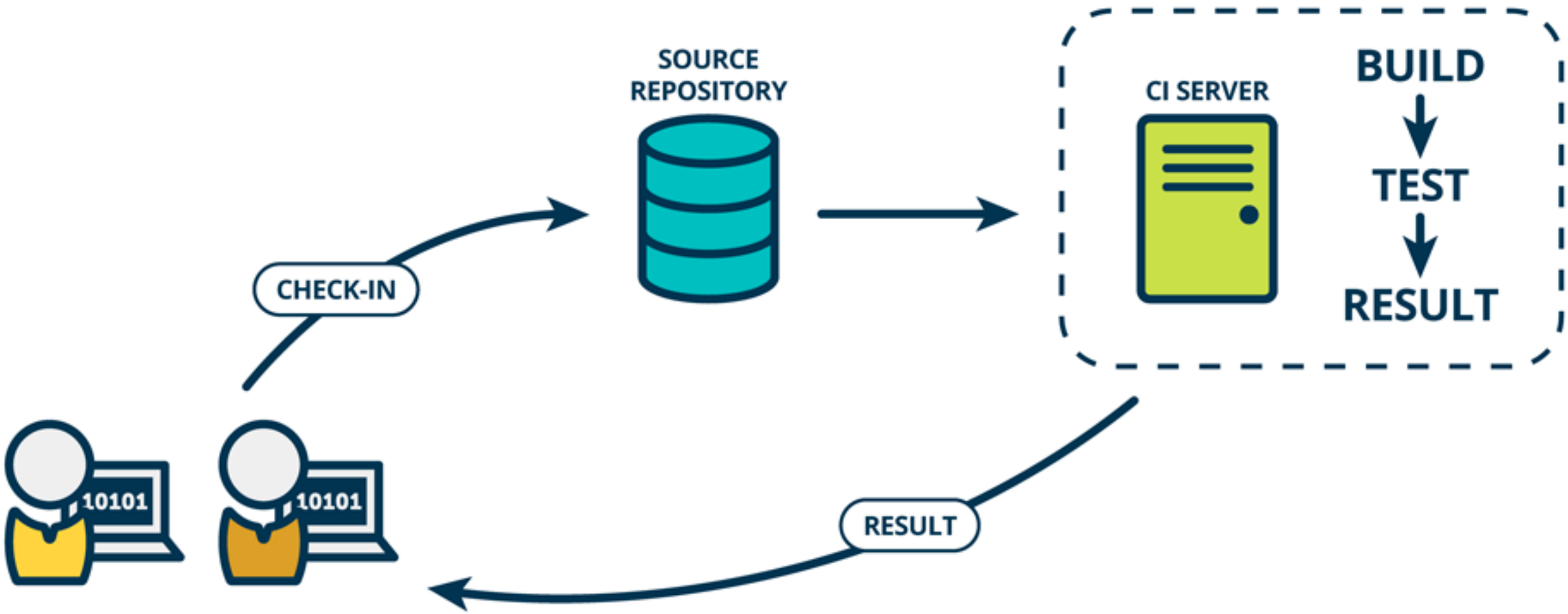
Continuous Integration

Strive to make **small change**

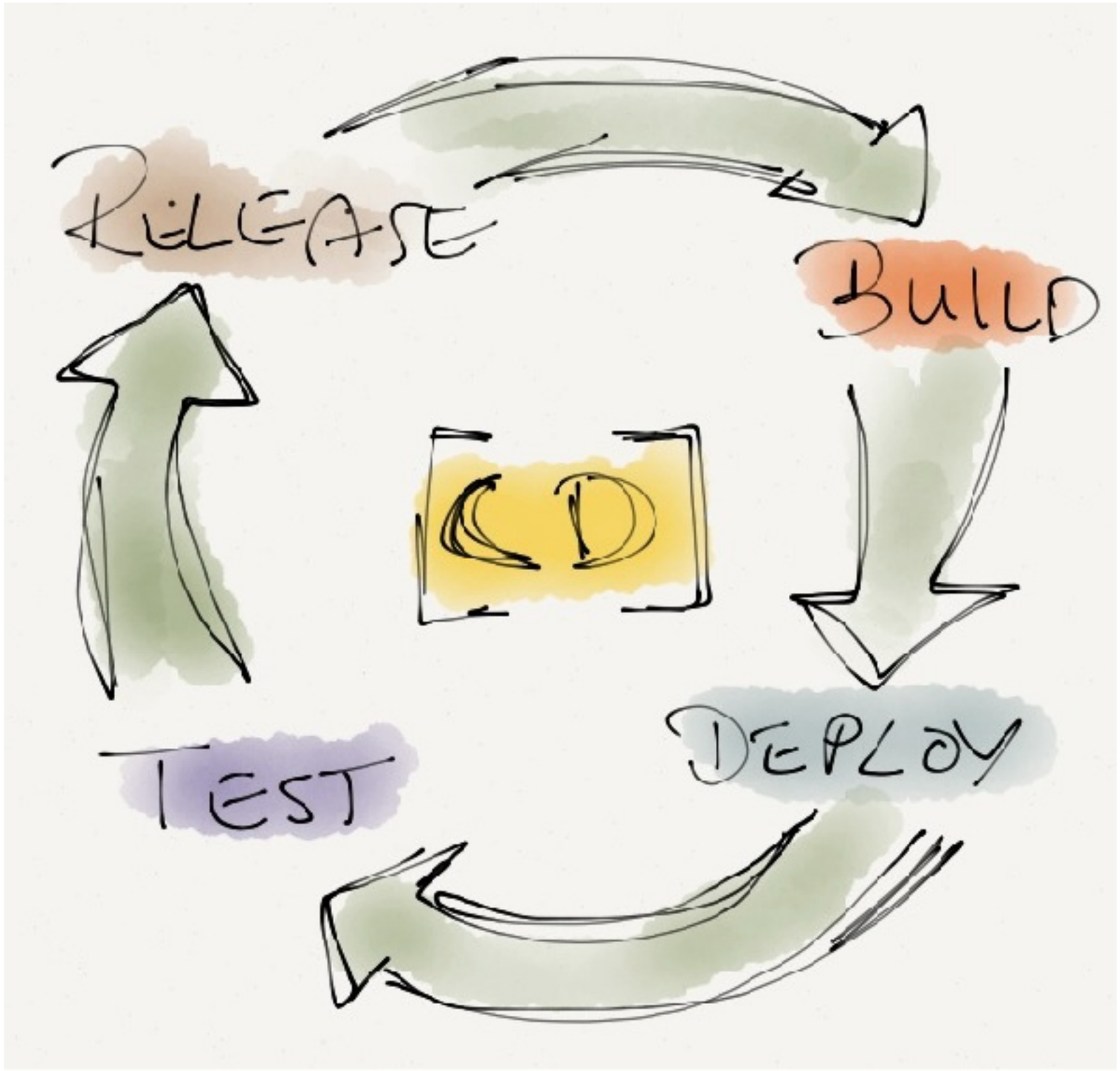
Continuous Integration

Strive for **fast feedback**

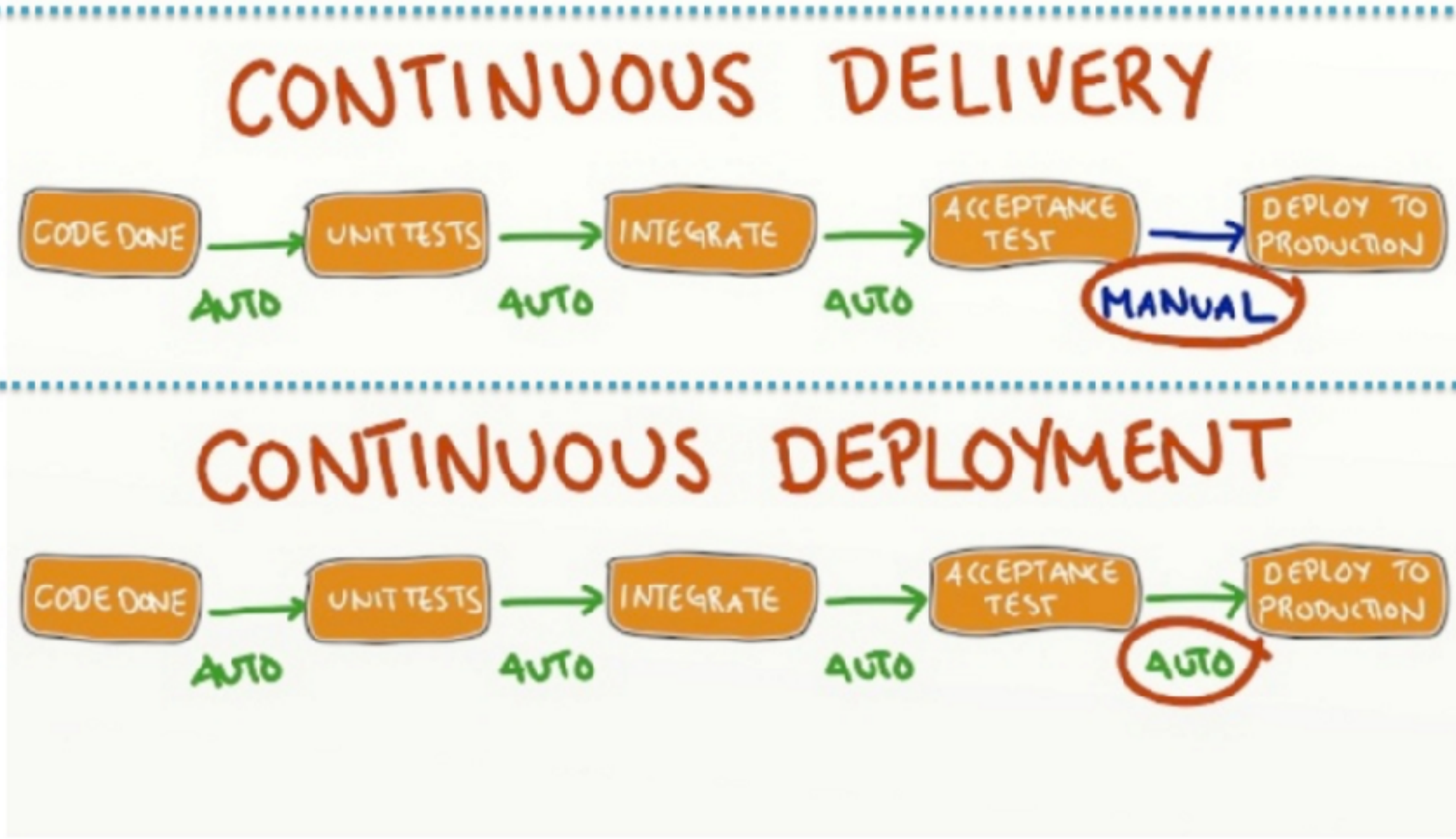
Continuous Integration



CD ?

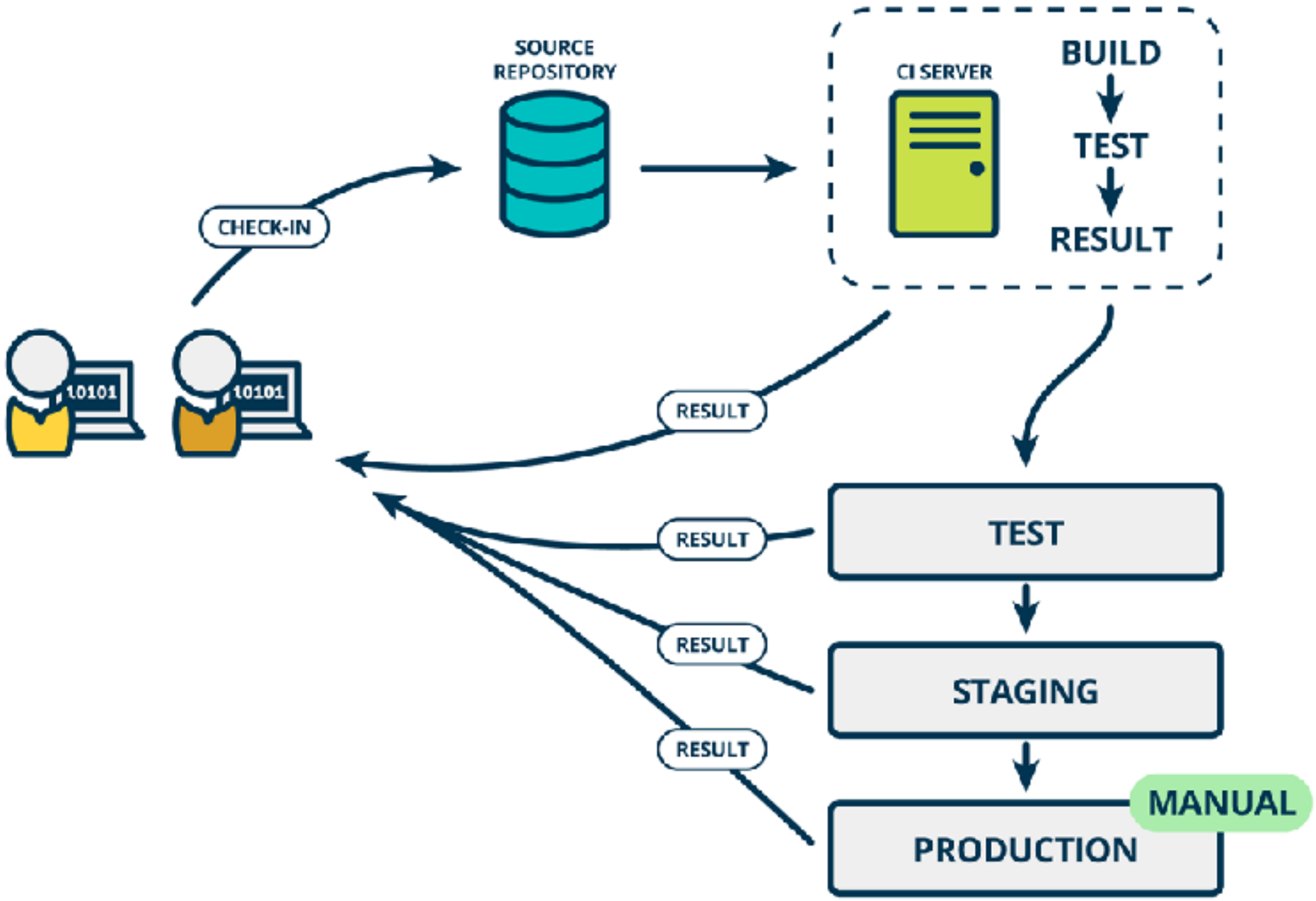


CD ?

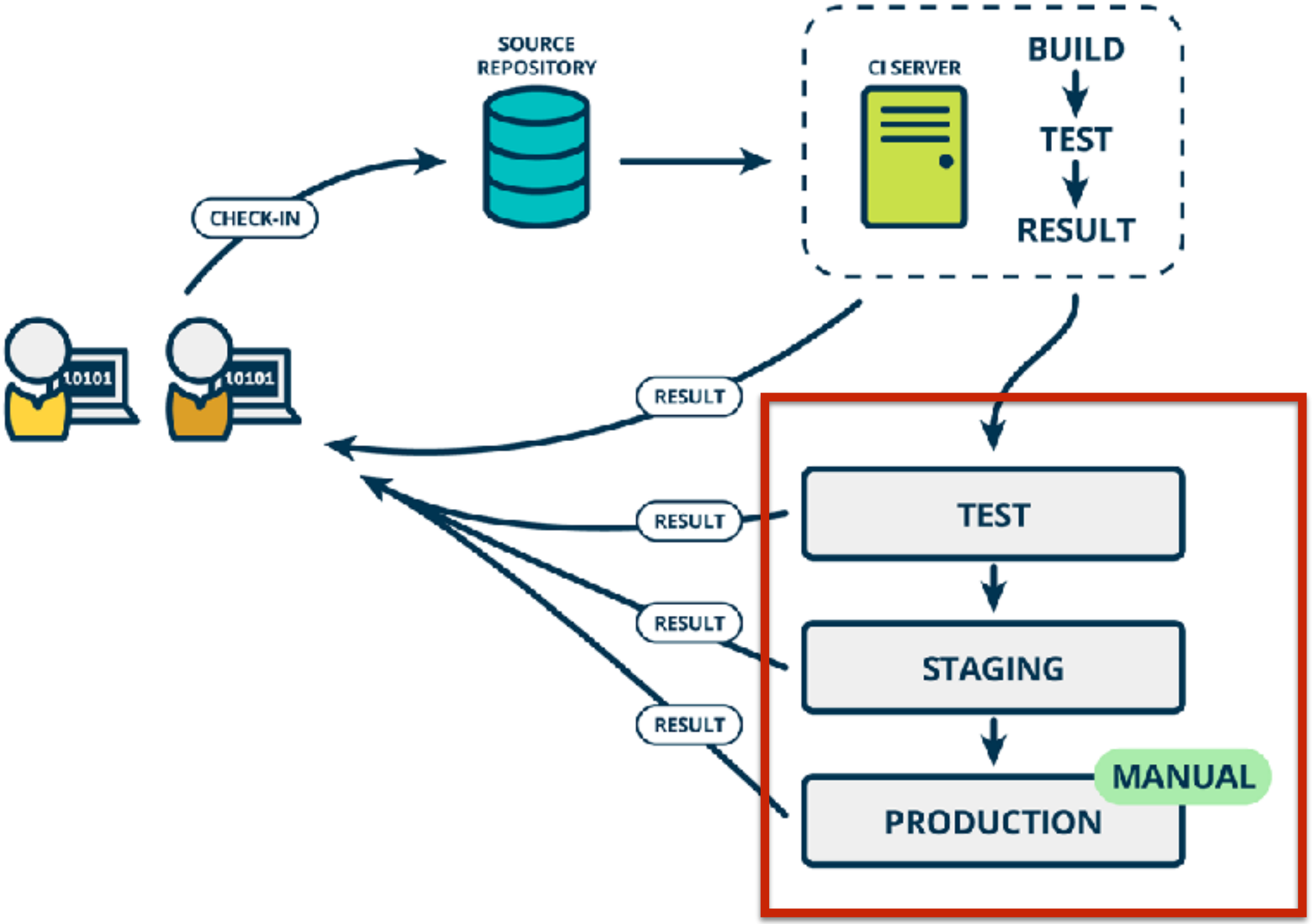


http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment

Continuous Delivery



Rise of DevOps



Continuous Integration

is a Software development **practices**

Practice 1

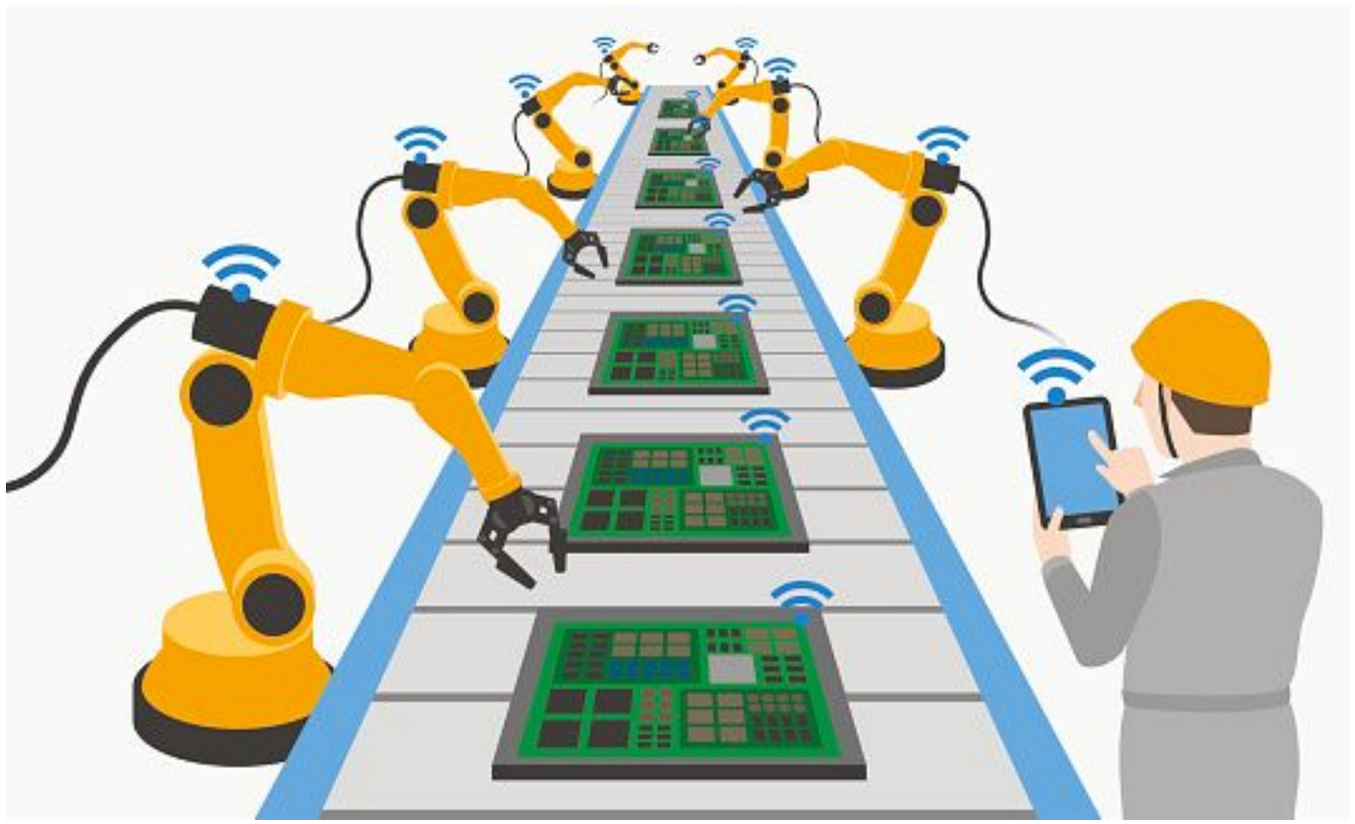
Maintain a single source repository

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

Practice 2

Automated the build

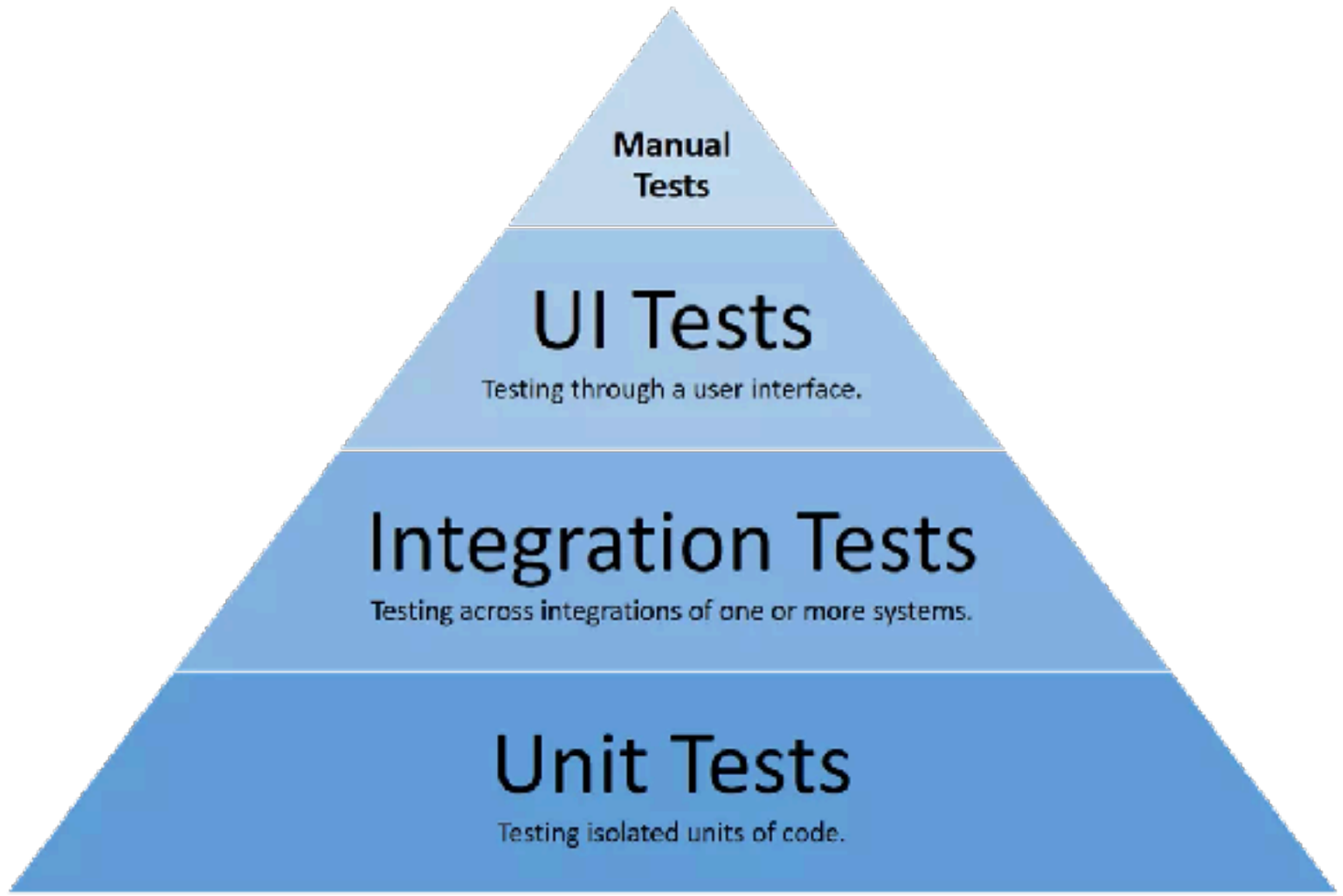
Automated environment for builds



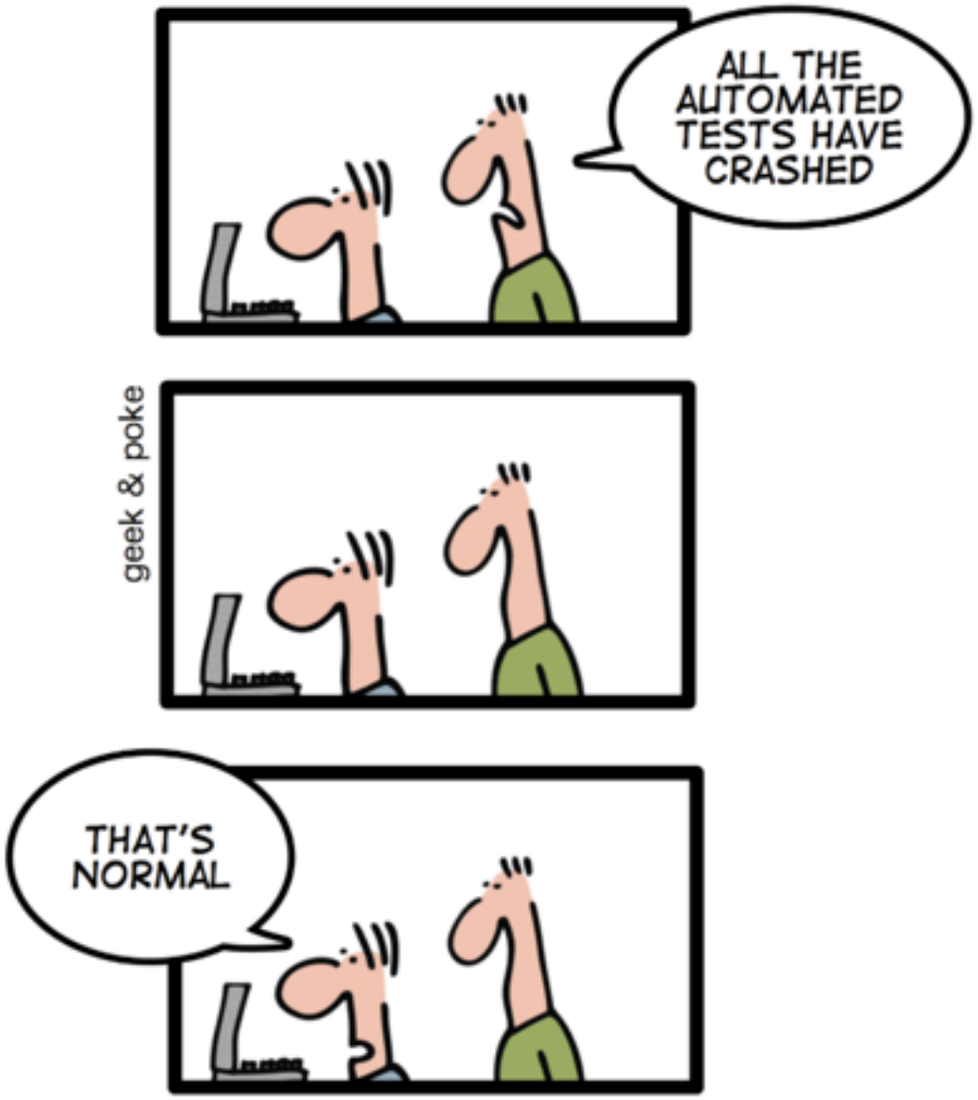
Practice 3

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

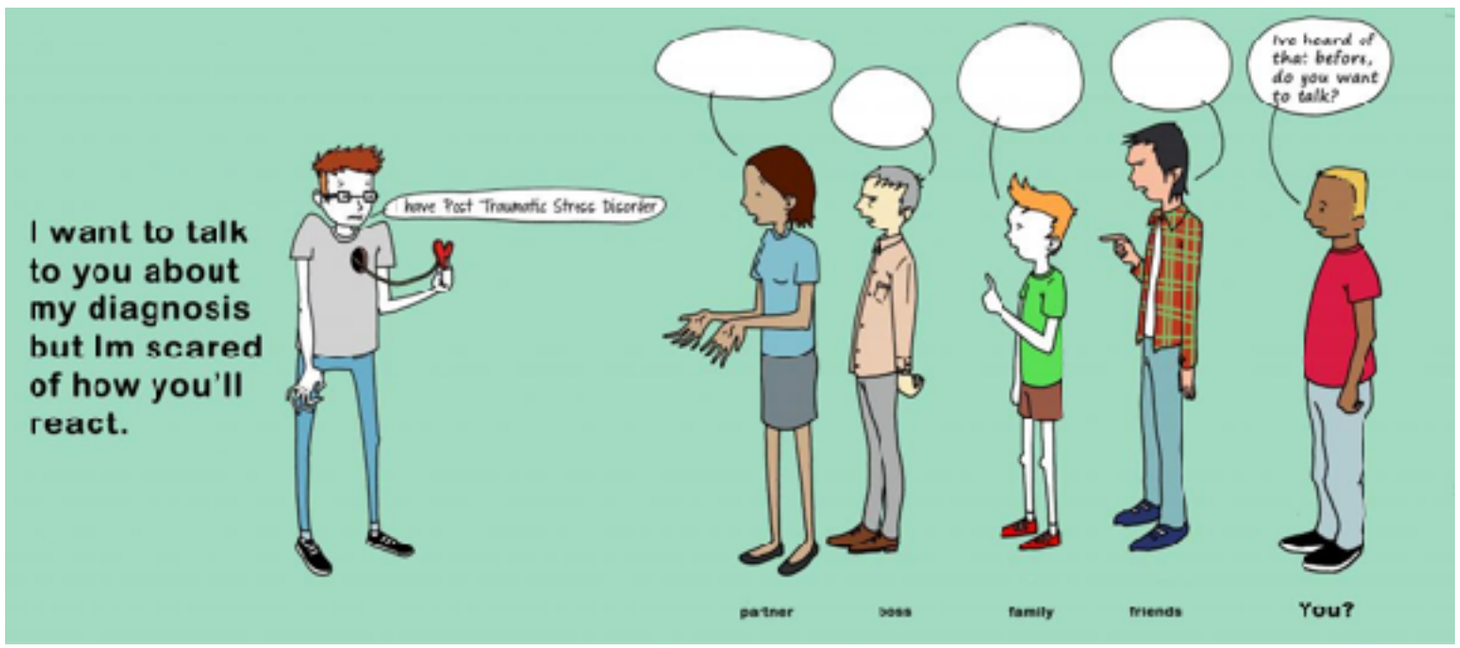


Practice 4

Everyone **commits** to the mainline everyday

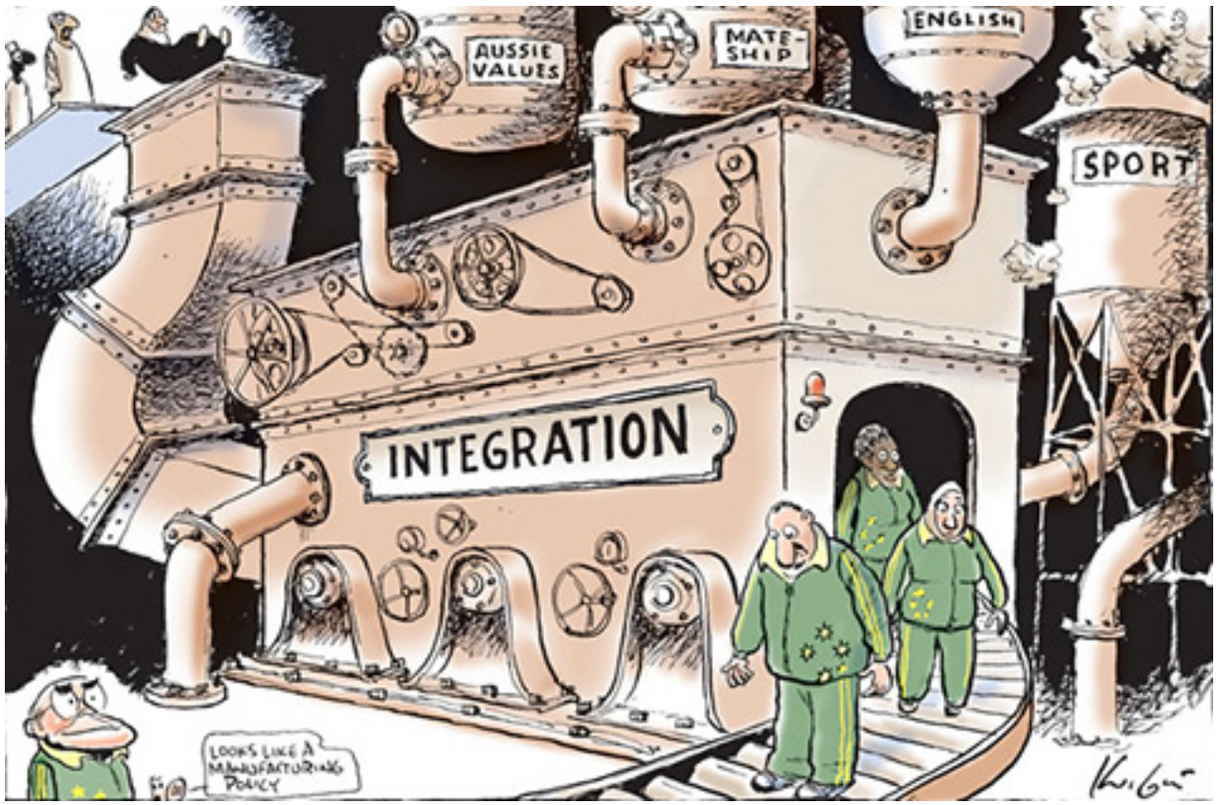
Integration is about **communication**

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

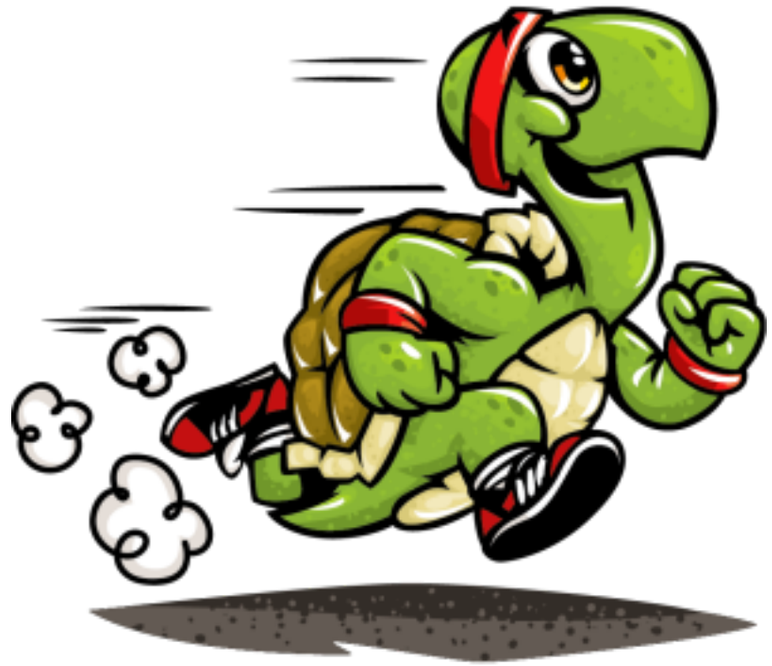
Fix broken builds **immediately**

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

Practice 7

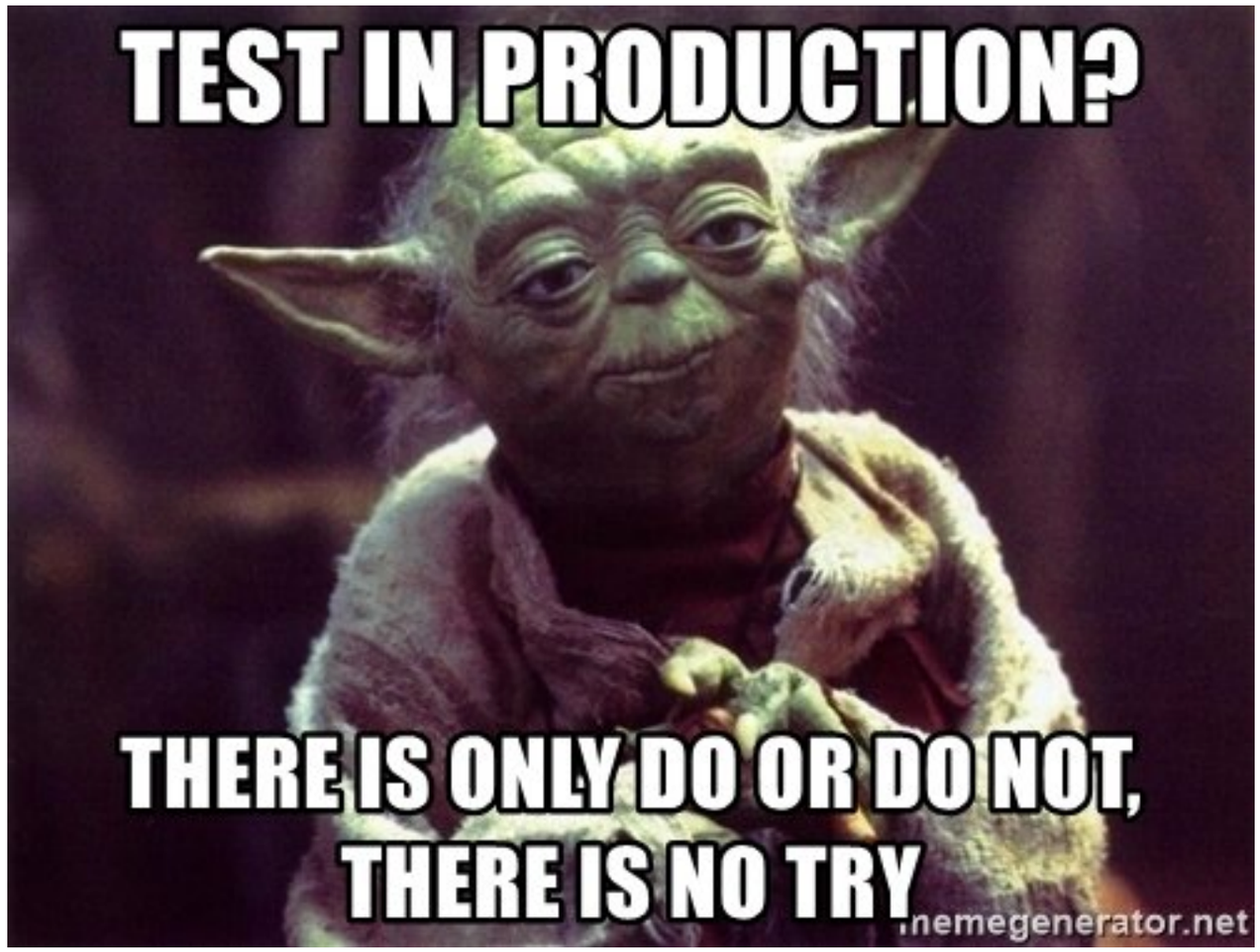
Keep the build **fast**

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

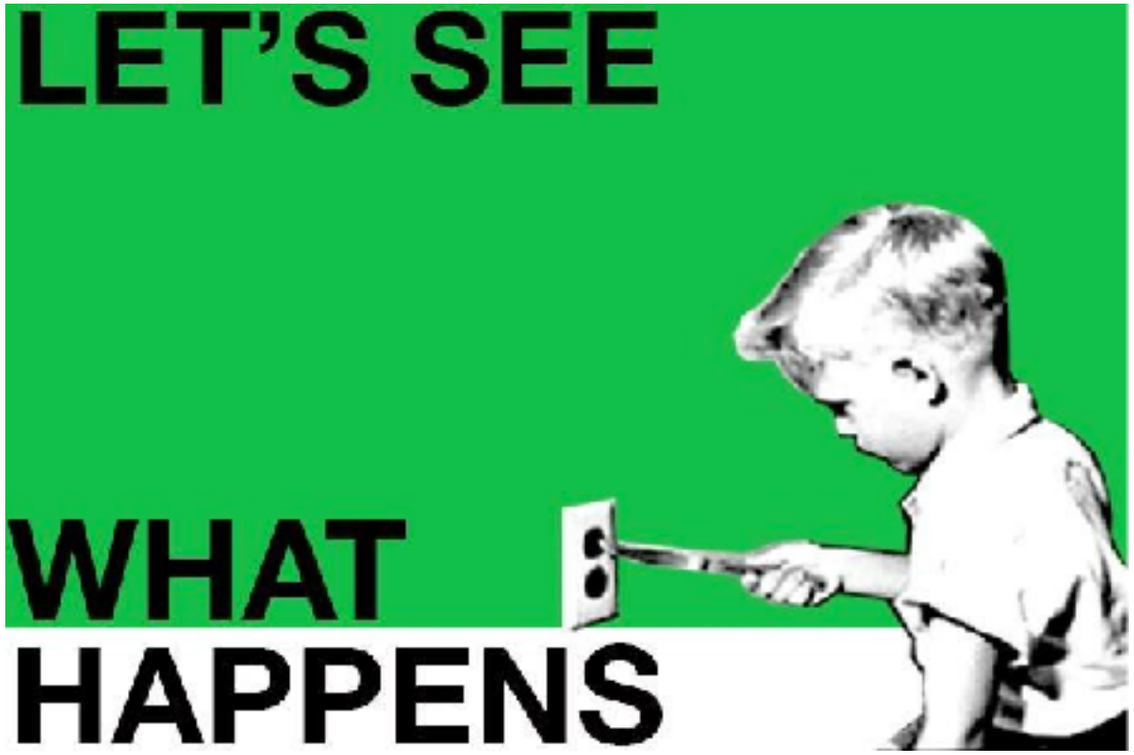
Make sure well known place where people can find

Practice 10

Everyone can see what's happening

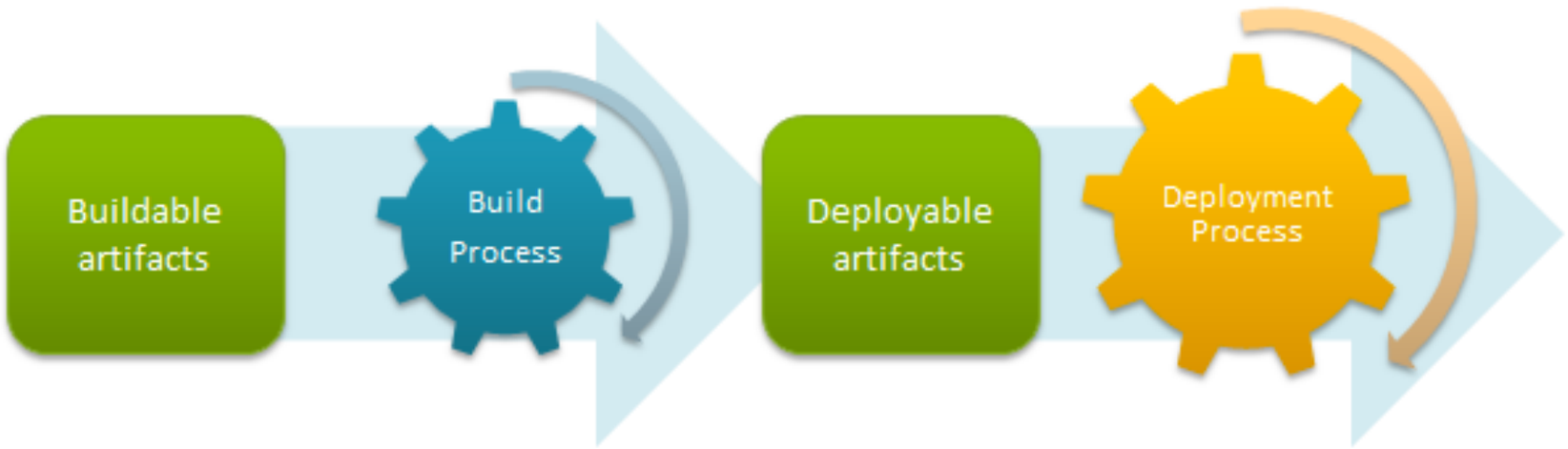
Easier to see the state of the system and changes

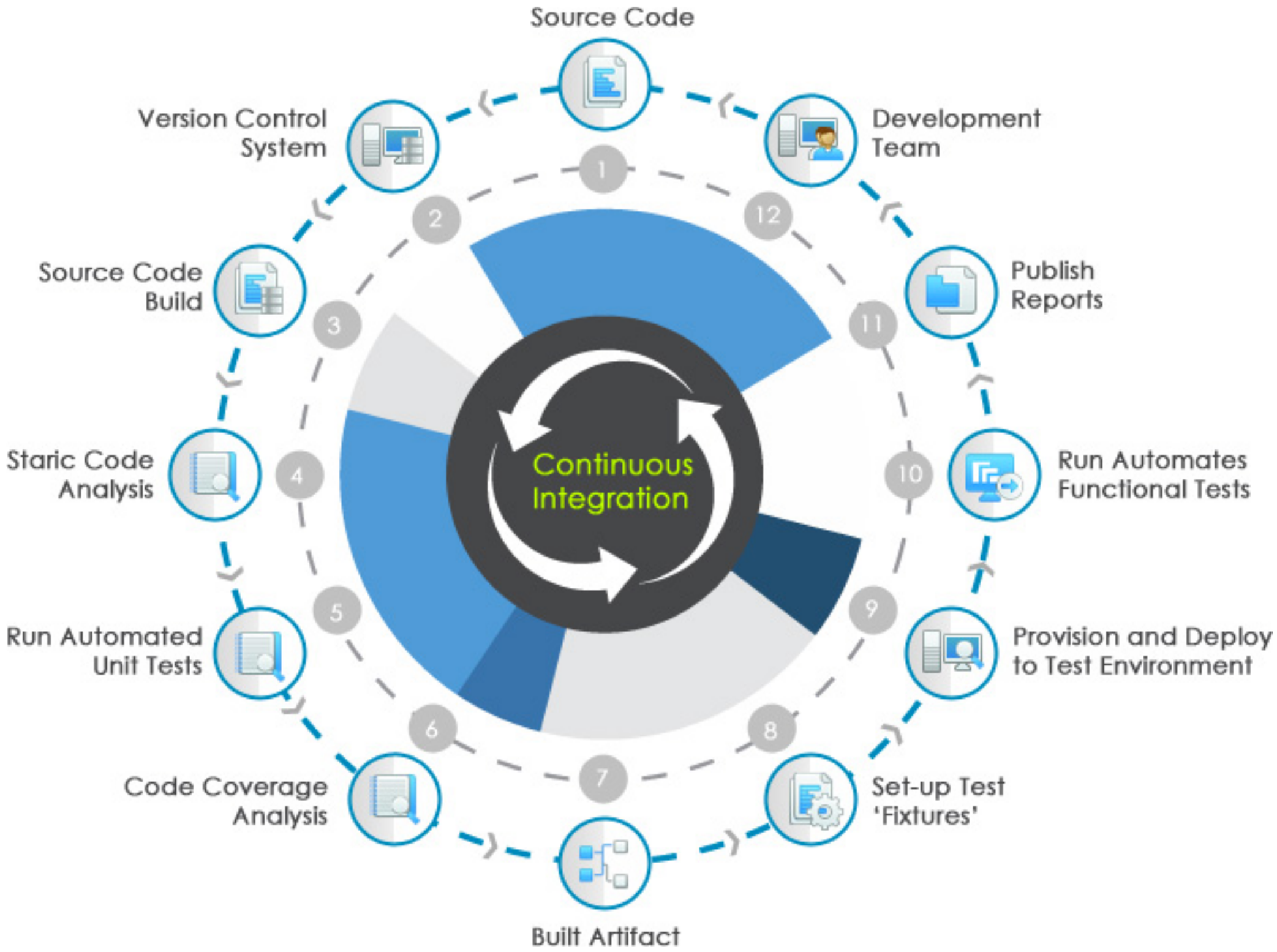
Show the good information



Practice 11

Automated deployment





The Twelve Factors

https://12factor.net/

The Twelve Factors

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

Code and Build

1 repository

1 build

https://trello.com/

Minimal services

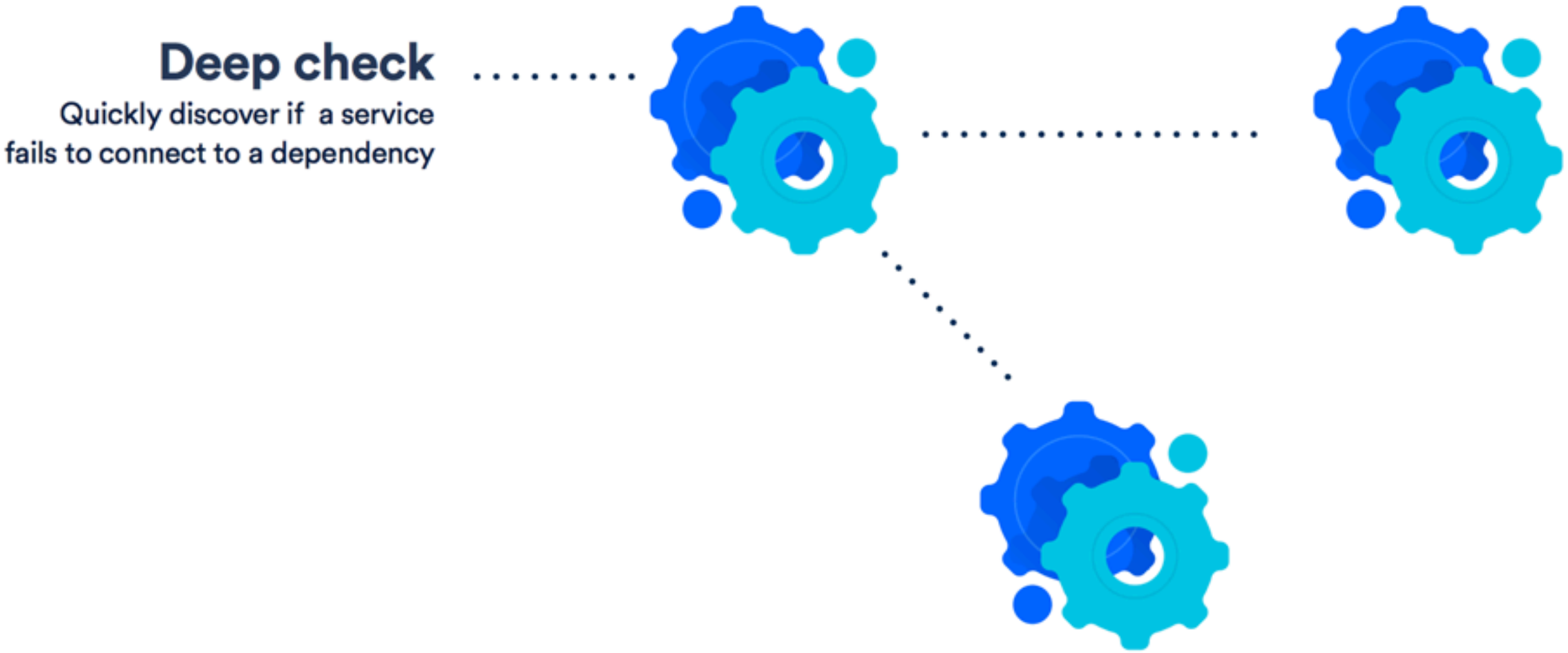
Expose a port
Only access to the service

Health check
200 app is alive. 500 app is unhealthy, destroy the node

Stateless*
Run as many nodes as you need

https://trello.com/

Health check services



https://trello.com/

Decouple

Libraries

Feel free to use shared libraries but keep them loose

Schemas

Make sure that services are resilient to schema changes

Testing

Test in isolation.

Config

Config is part of the service don't have dependencies

https://trello.com/

“Strict separation of config from code”

12 factor app

Configuration lifecycle

Rebuild

Rebuild to apply changes

Redeploy

Part of the service configuration.

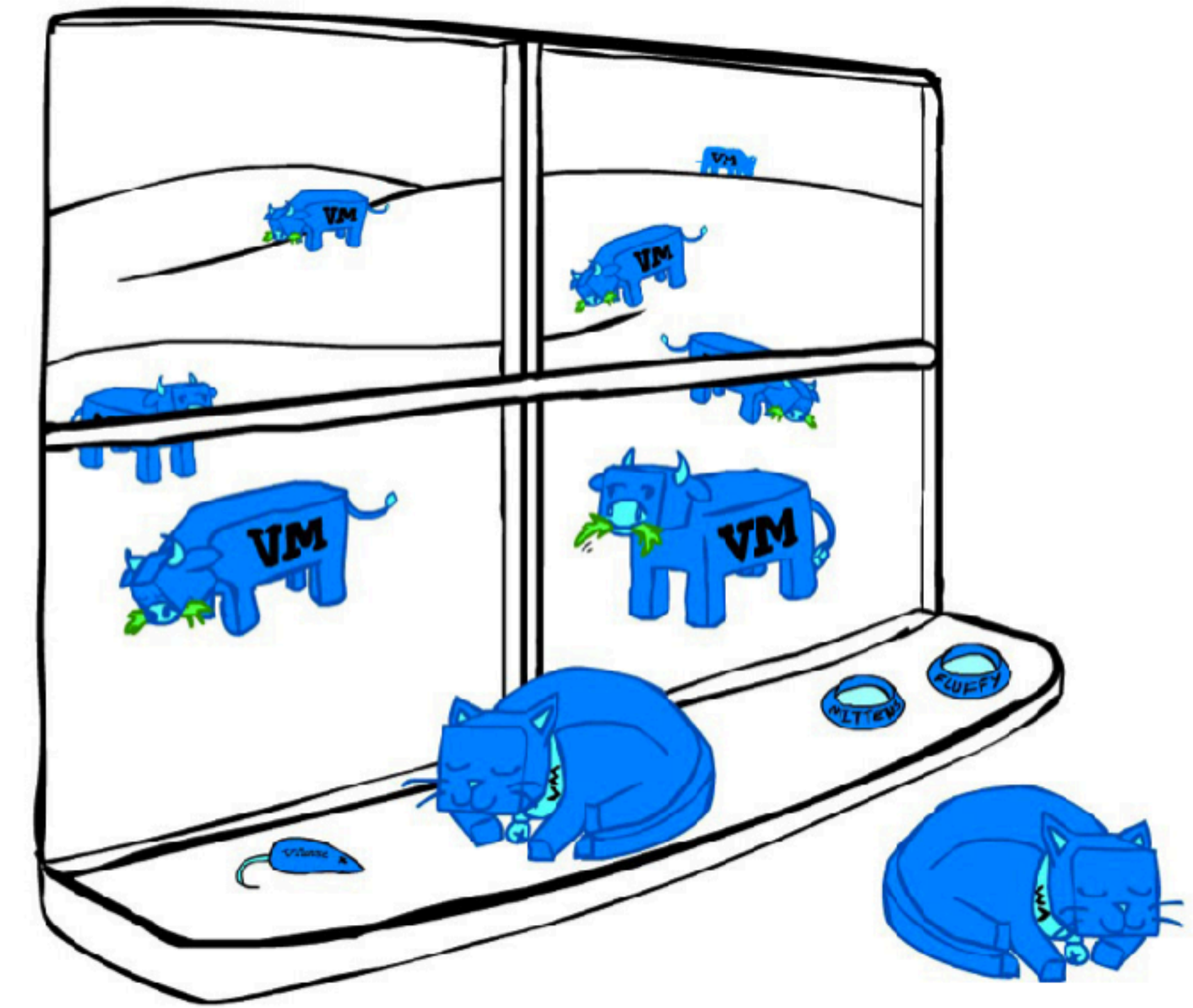
Instant change

Switches you would like to enable/disable straight away

https://trello.com/

“Treat them as cattle, not pets”

Bill baker



https://turbonomic.com/blog/on-technology/old-virtualization-conundrum-pets-cattle/

“Pets vs Cattle” (Yes, again)

Scale Up

- Servers are like pets.

Pets are given names, are unique, lovingly hand raised and cared for. When they get ill, you nurse them back to health

- Servers are like cattle.

Cattle are given numbers and are almost identical to each other. When they get ill, you get another one.

> “Future application architectures should use Cattle but Pets with strong configuration management are viable and still needed”
> - Tim Bell, CERN

The above adapted from Tim Bell, CERN
http://www.slideshare.net/noggin143/20121017-openstack-cern-accelerating-cience

https://cohesive.net/2017/10/5-concepts-to-truly-understand-cloud-computing.html

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

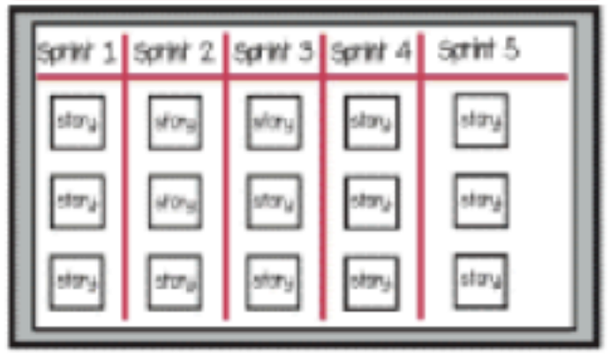
WORKING SOFTWARE over full documentation

Agile principles

1 Satisfy the **customer**

Welcome **change**

Deliver **frequently**



4 Work **together**

5 Trust and **support**

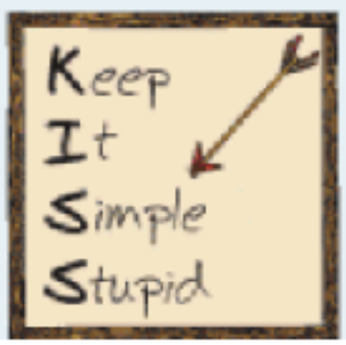
Face-to-face **conversation**

Working **software**

8 Sustainable **development**

9 **Technical** Excellence

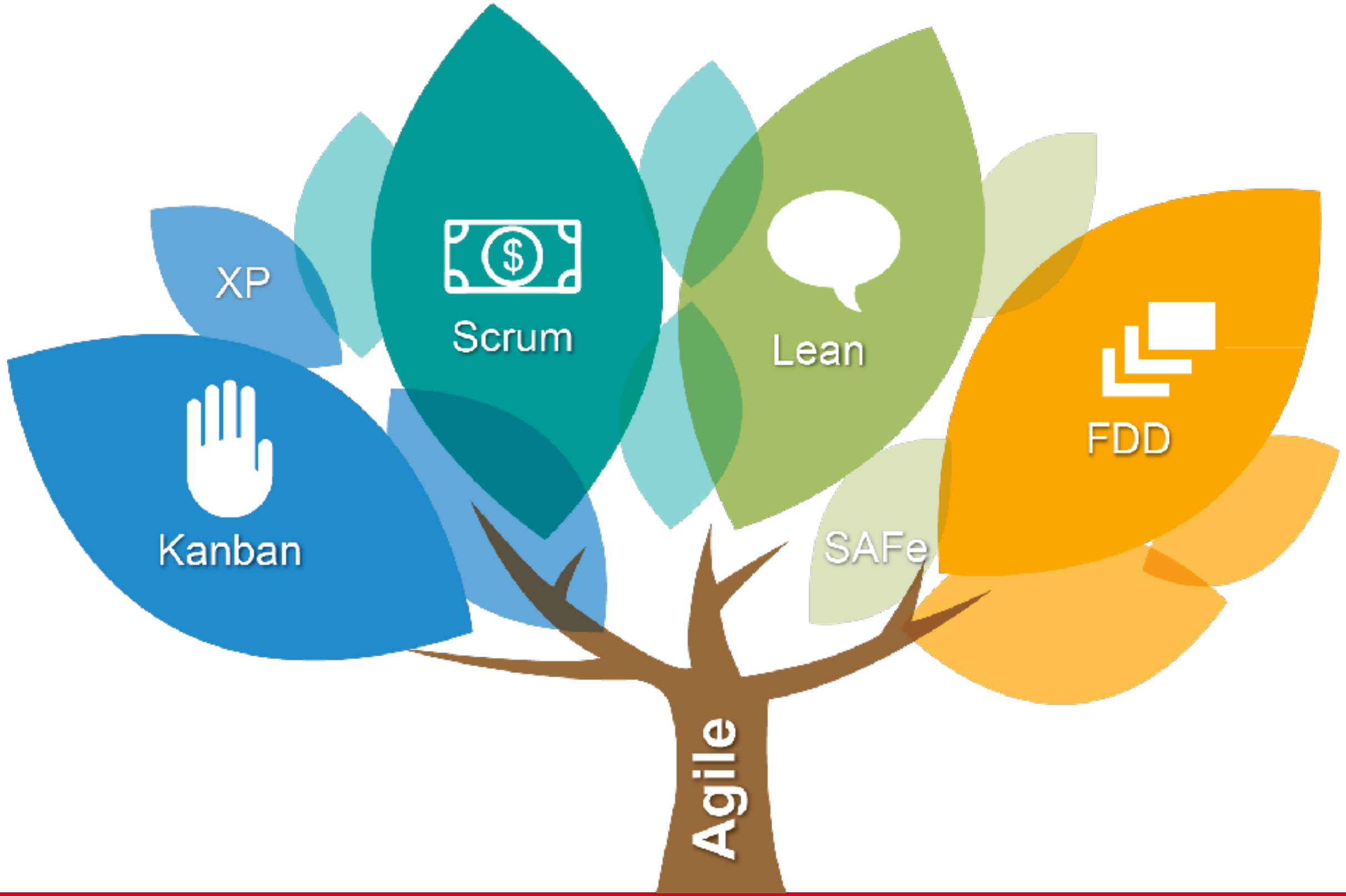
10 Maintain **simplicity**

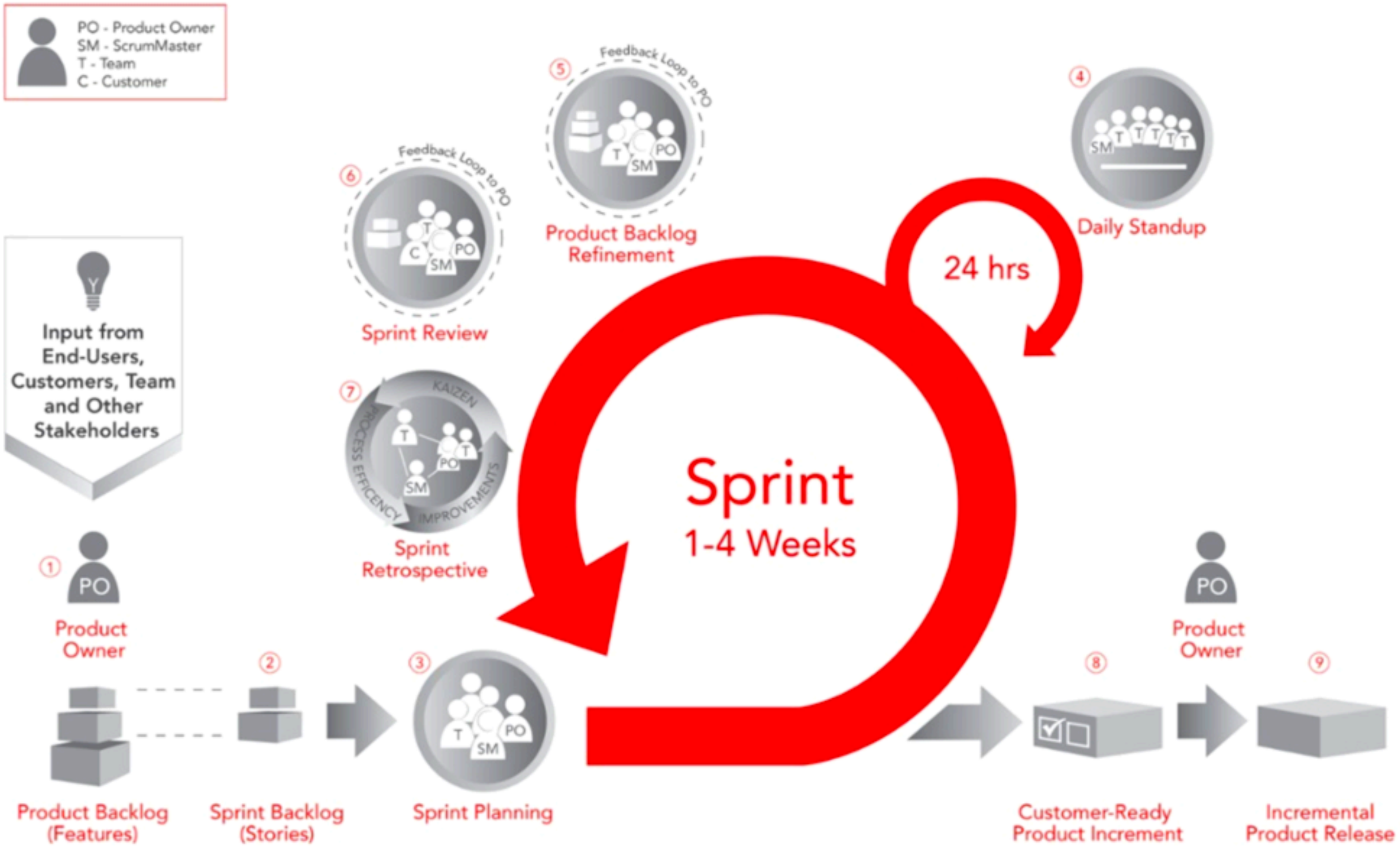


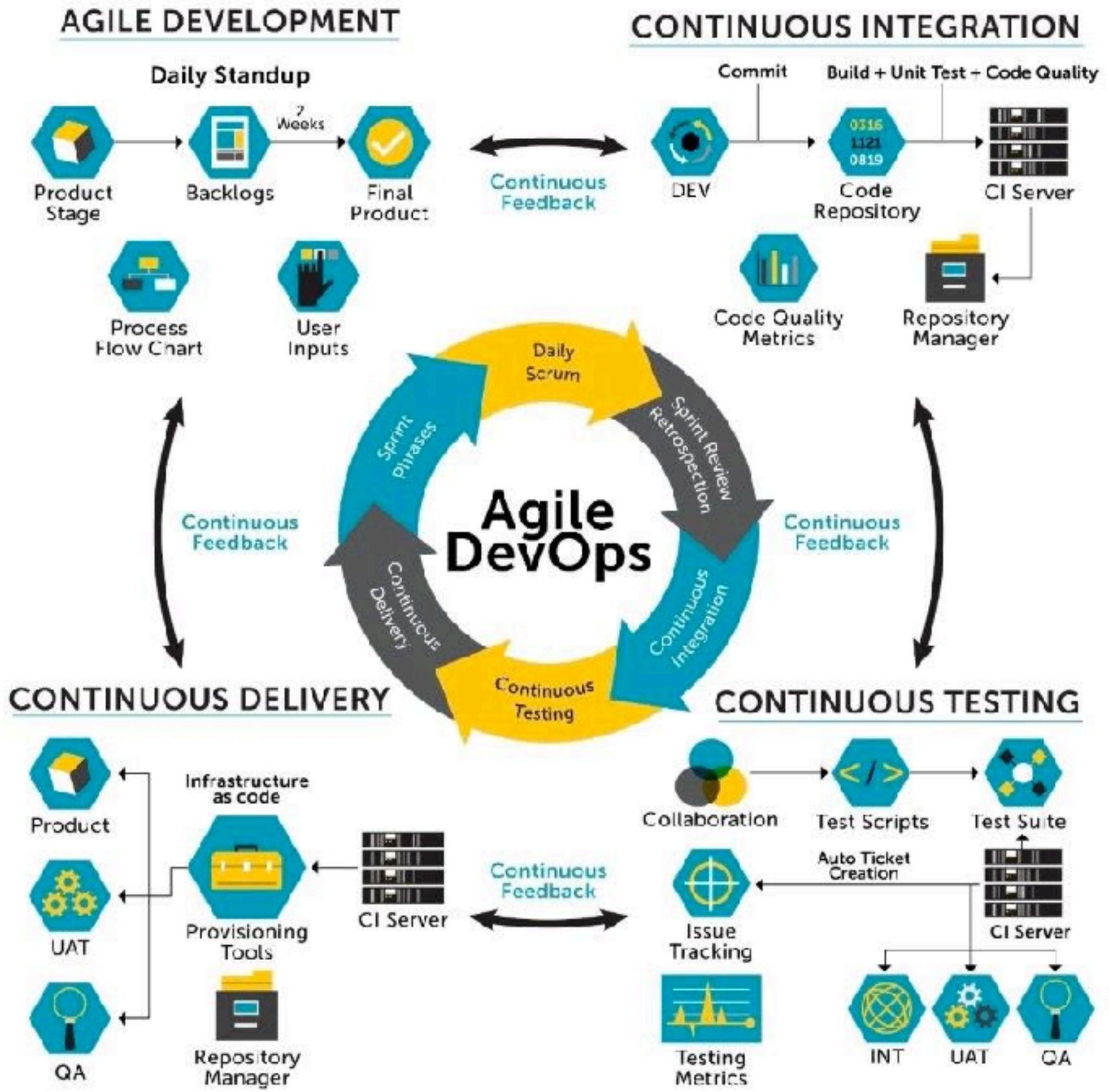
11 Self-organizing **teams**

12 Reflect and **adjust**

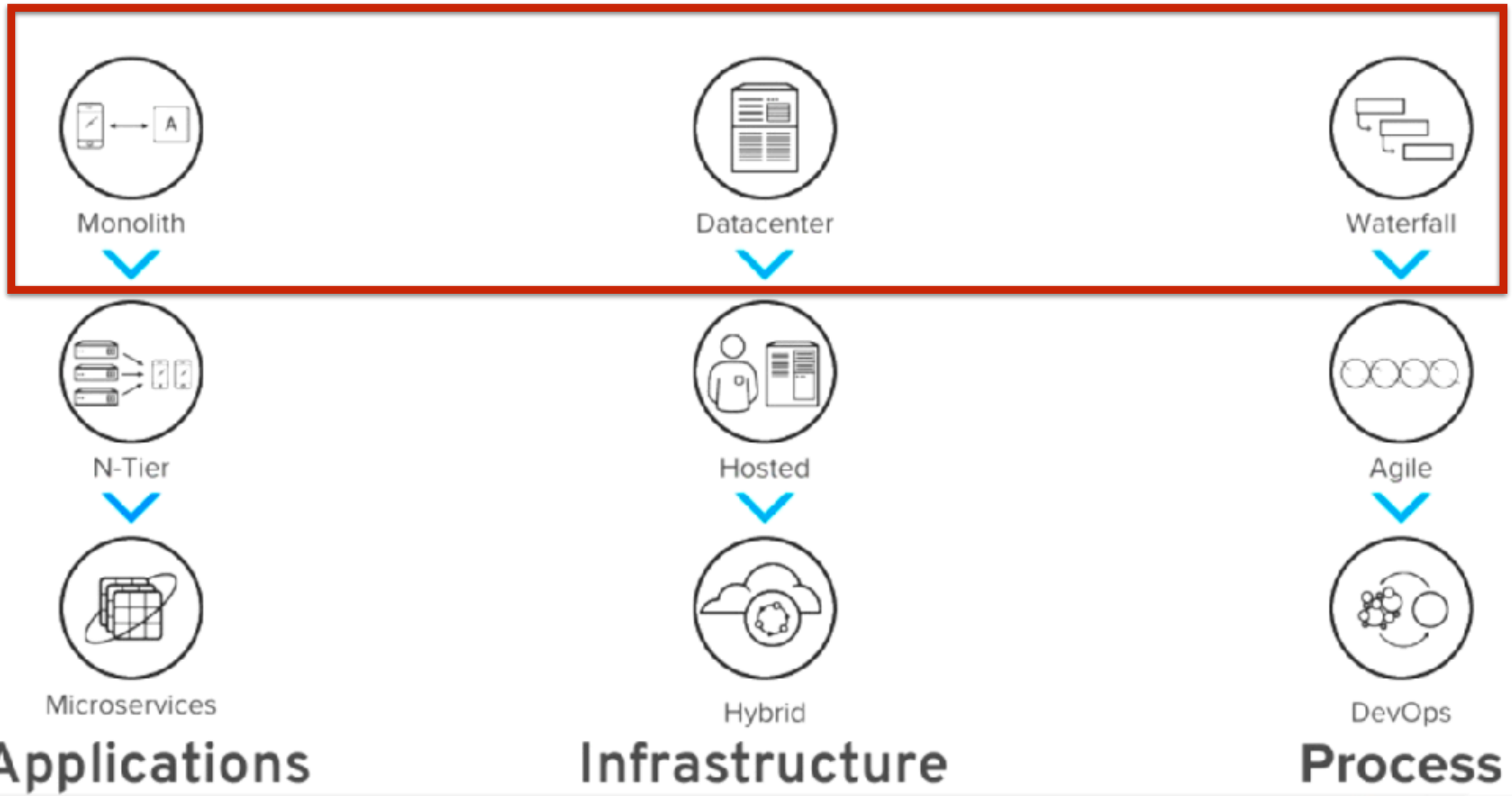
Origin by https://www.knowledgetrain.co.uk, modified by Jacky Shen



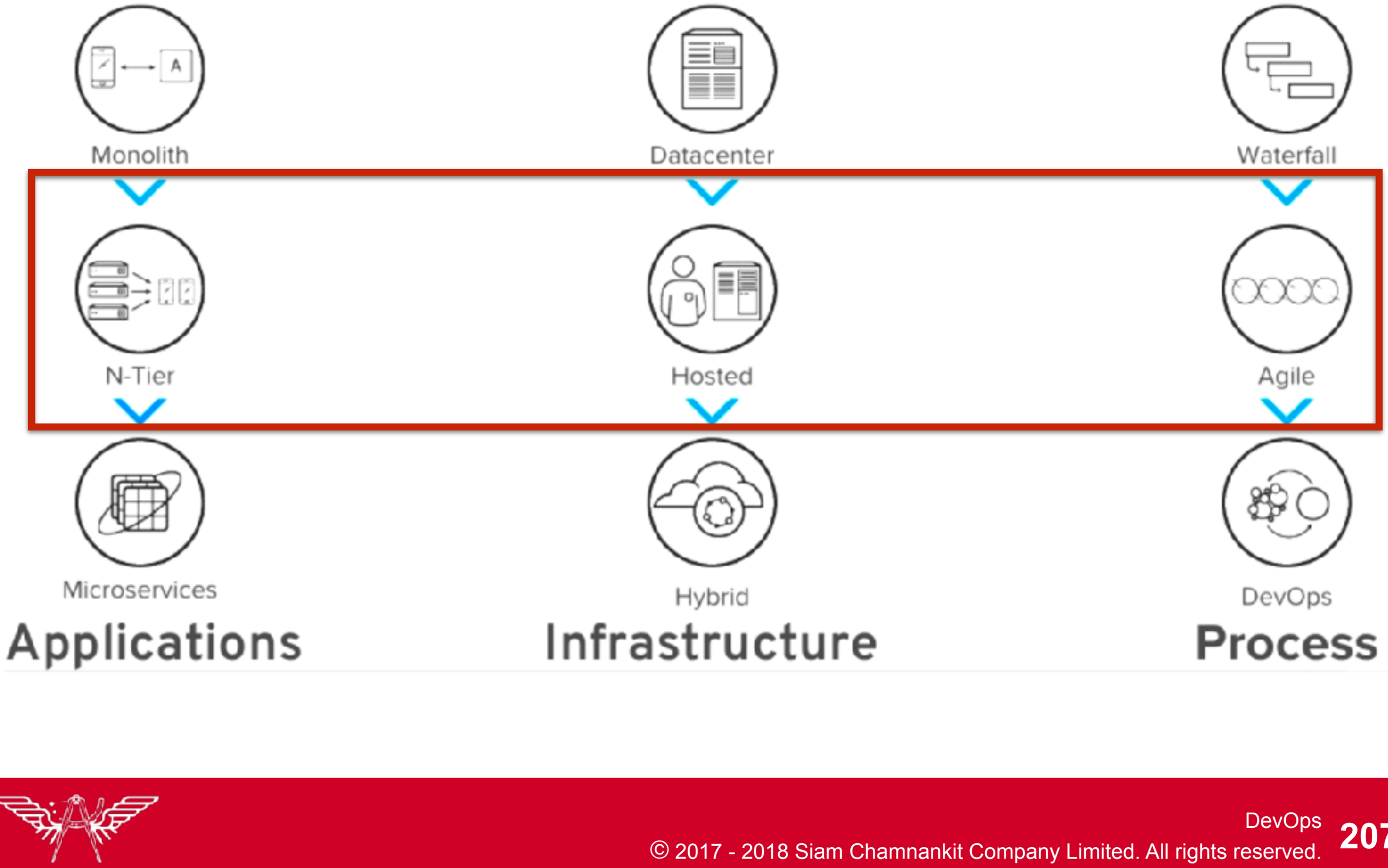




Continuous Improvement



Continuous Improvement



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

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 

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

Take to your home (1)

People are #1 asset

Take to your home (2)

DevOps is not a recipe, work out your own flavor

Take to your home (3)

Start with business, even if you don’t have a trust relationship

Take to your home (4)

Success sustainable requires both
bottom-up practices and
top-down management support

Take to your home (5)

DevOps adoption is not easy, but rewarding

Take to your home (6)

Focus on **value**, not **quantity**

Take to your home (7)

Always improve, always practice

Thank you
Q/A