

# DevOps





Somkiat Puisungnoen

Search

Somkiat Home

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro

Software Craftsmanship

Software Practitioner at สยามชั่นนาคุกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok ·

Java and Bigdata

This image shows a screenshot of a Facebook profile page for 'Somkiat Puisungnoen'. The top navigation bar includes the Facebook logo, the user's name, a search bar, and profile picture. Below the profile picture is a collage of several photos, with one prominent photo of a man in a white t-shirt flexing his biceps. The main menu below the collage includes 'Timeline', 'About', 'Friends 3,138', 'Photos', and 'More'. On the left, there's a sidebar with a pending items section and an 'Intro' section listing professional roles. The right side features a status update input field and a recent post from the user about Java and Bigdata.



Facebook somkiat.cc

Somkiat Home | ? ▾

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

somkiat.cc  
@somkiat.cc

Home Posts Videos Photos

Liked Following Share ... + Add a Button

Help people take action on this Page. ×



# 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 ?



Customers



“The Business”



Product Teams

Platform Teams

Infrastructure Teams

Operations Teams

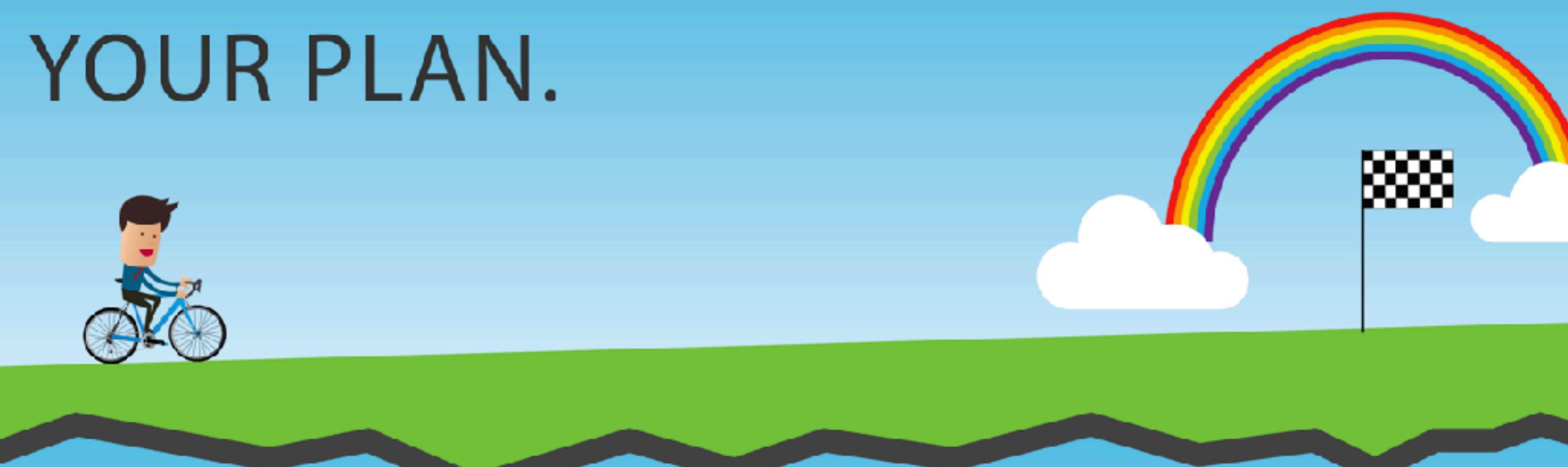


Google/Amazon

<https://bravenewgeek.com/>

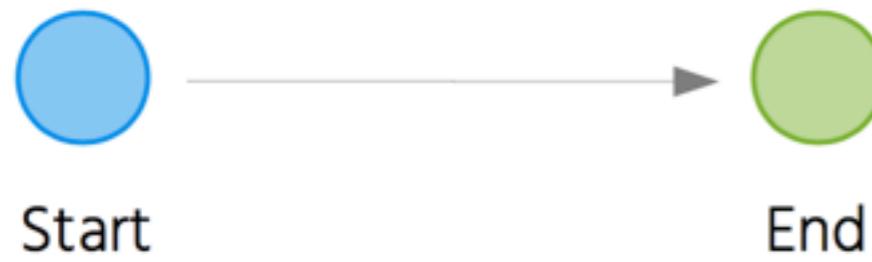


# YOUR PLAN.

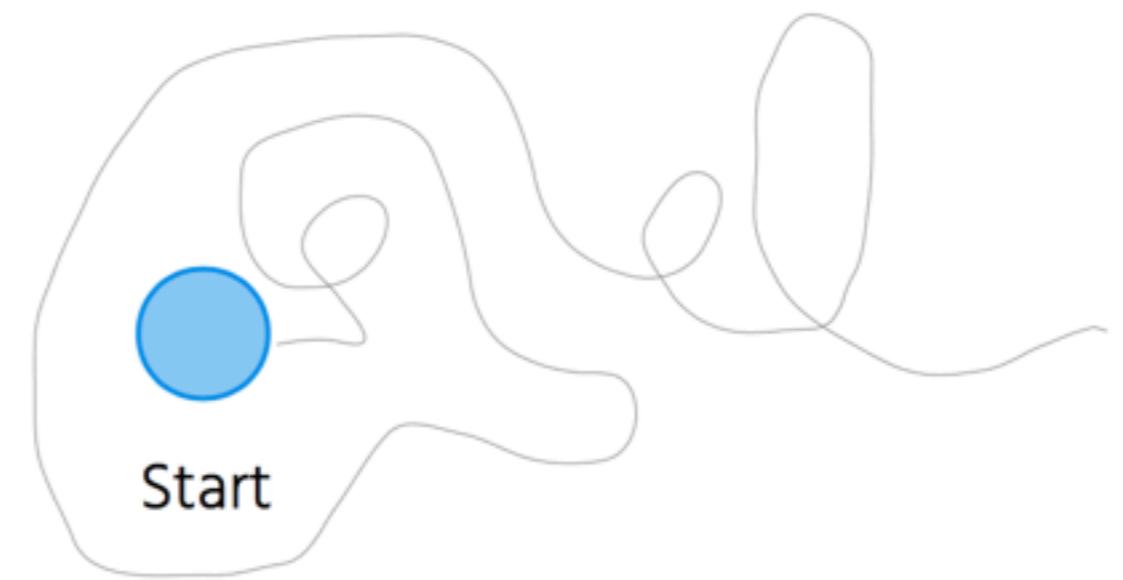


# REALITY.





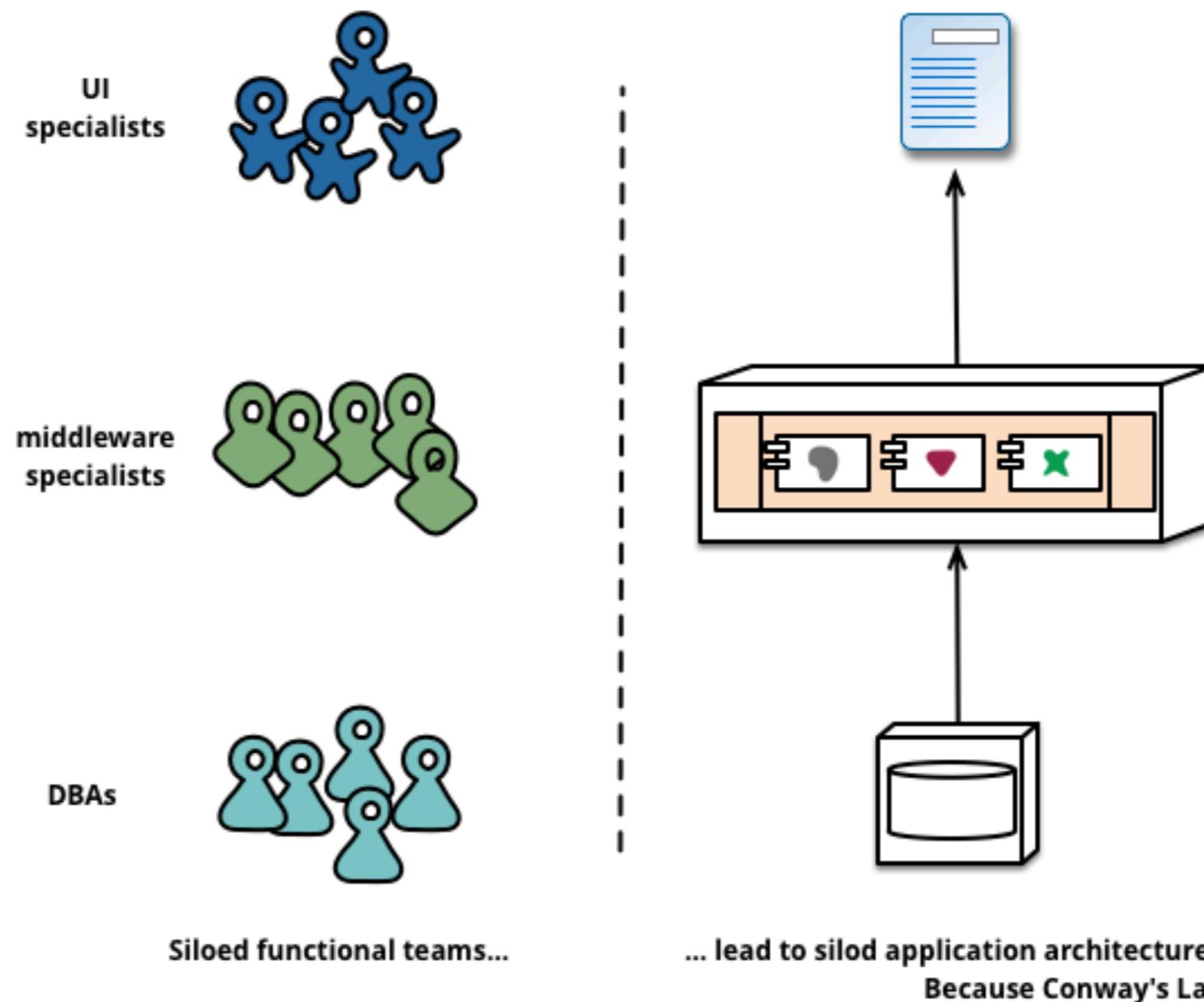
Theory



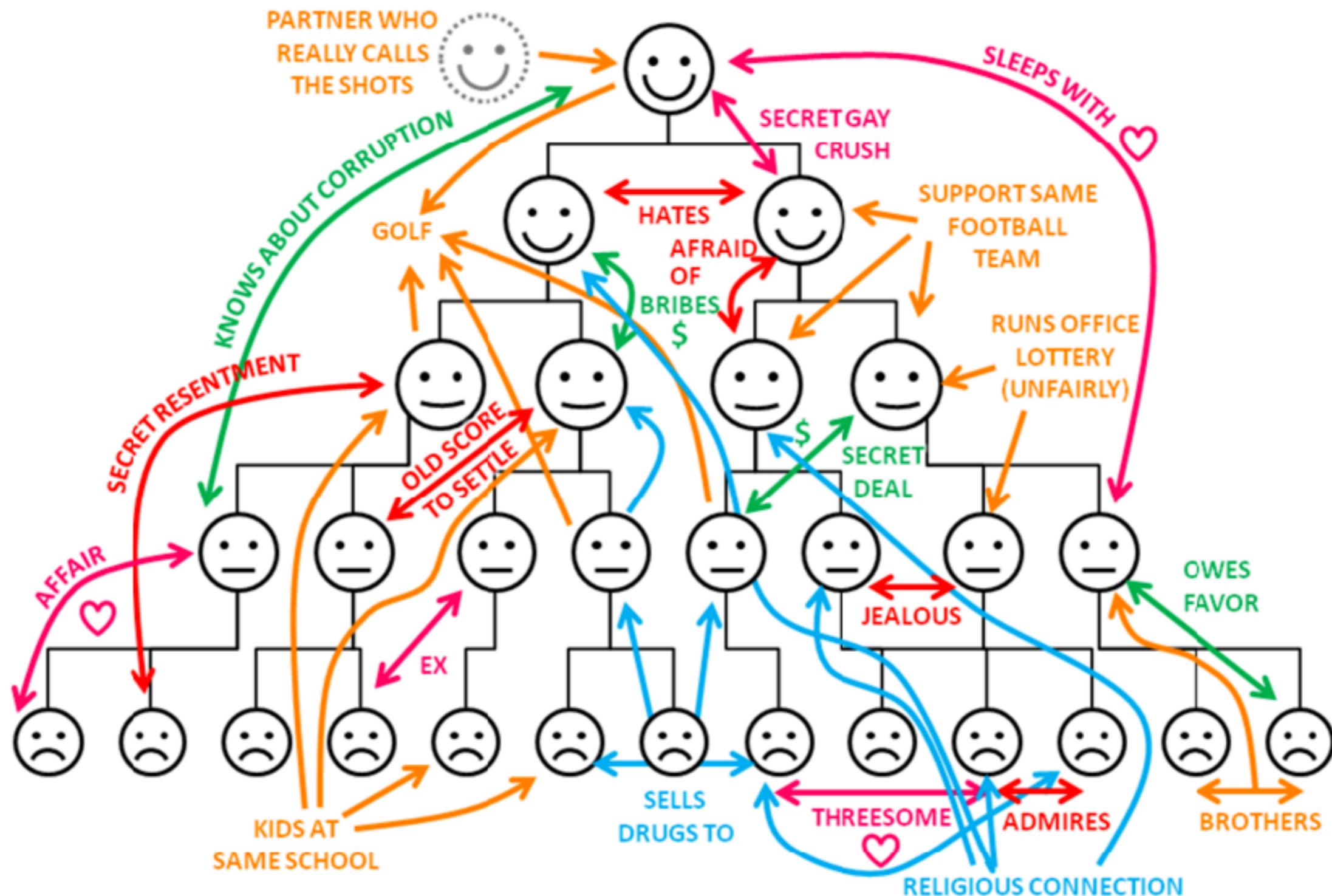
Reality



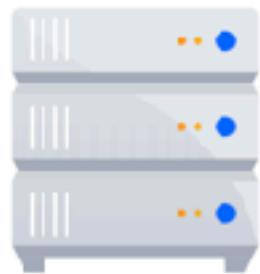
# Conway's Law



# 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

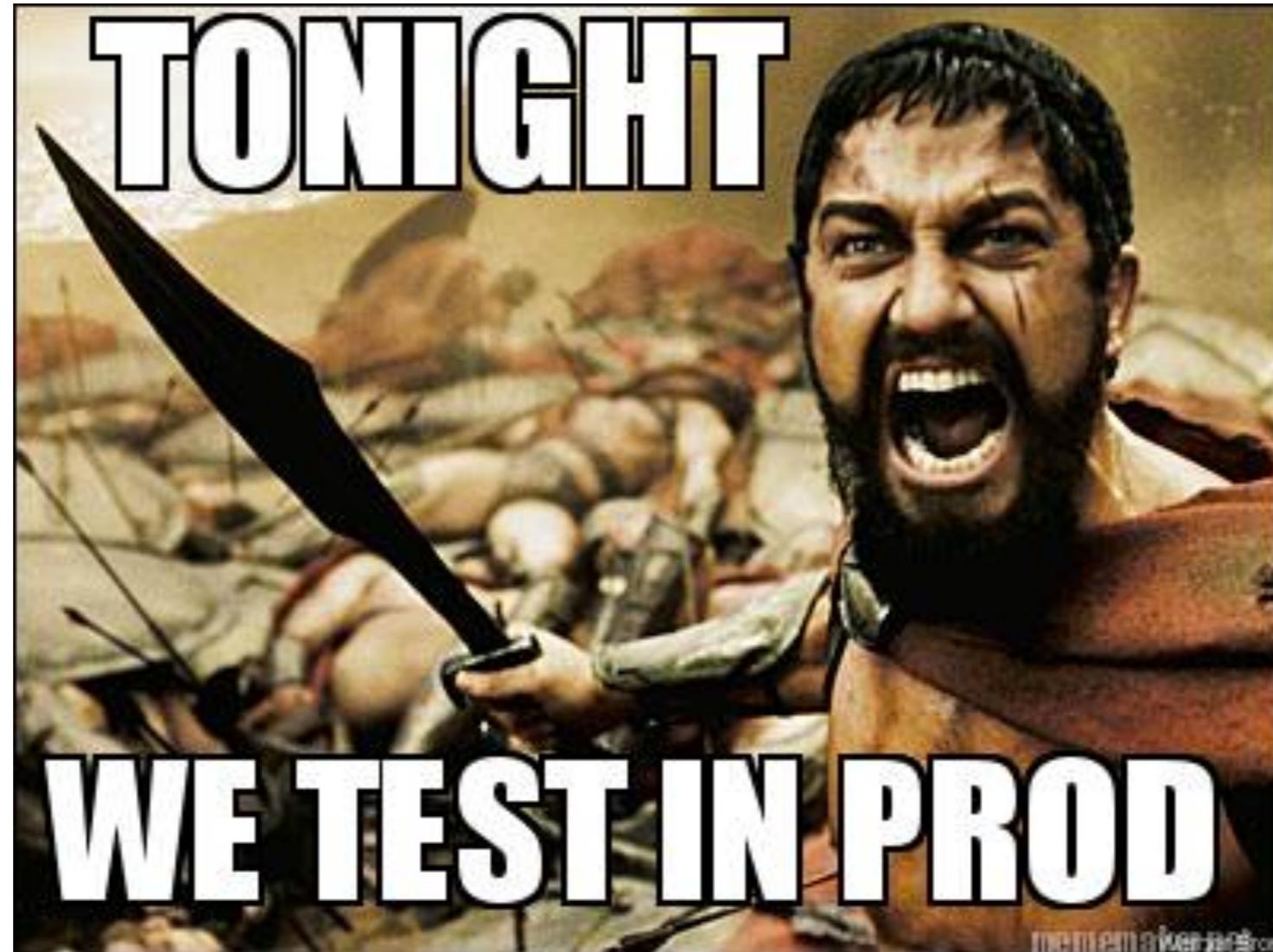
<https://trello.com/>



# Problem ?



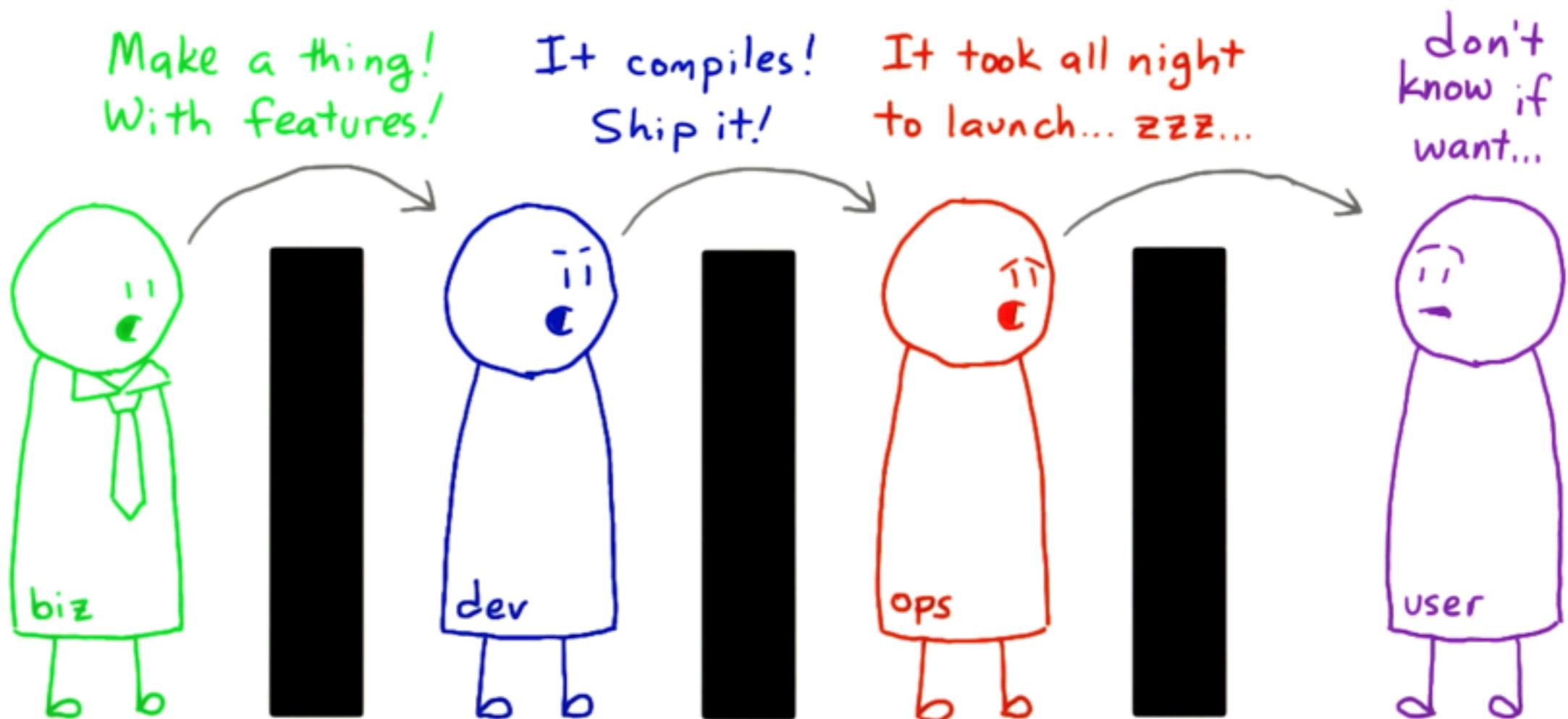
# Problem ?



# Problem ?



# Problem ?



# **What happens in an Internet minute ?**



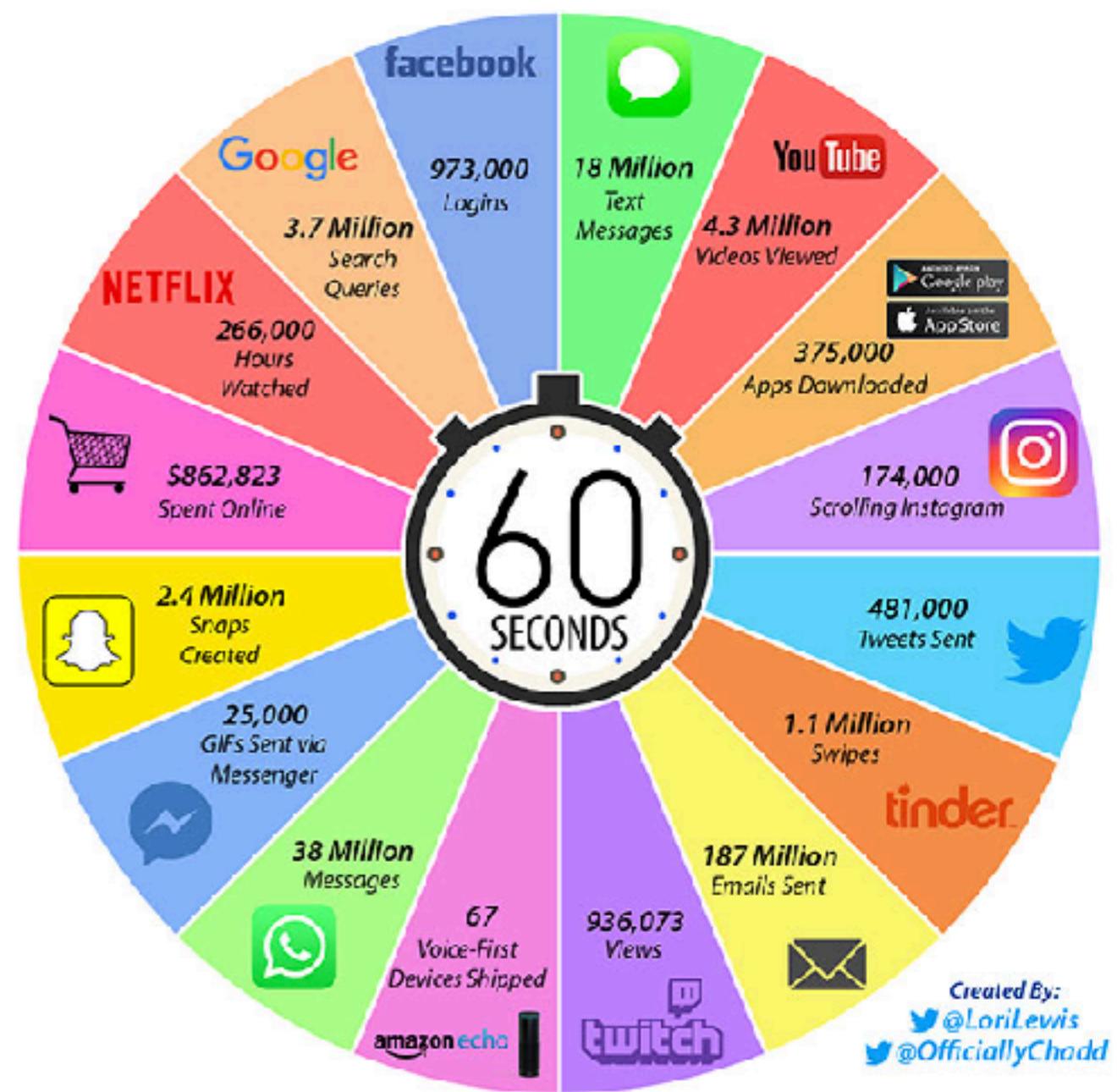
# 2018 This Is What Happens In An Internet Minute



# 2017 This Is What Happens In An Internet Minute



# 2018 This Is What Happens In An Internet Minute



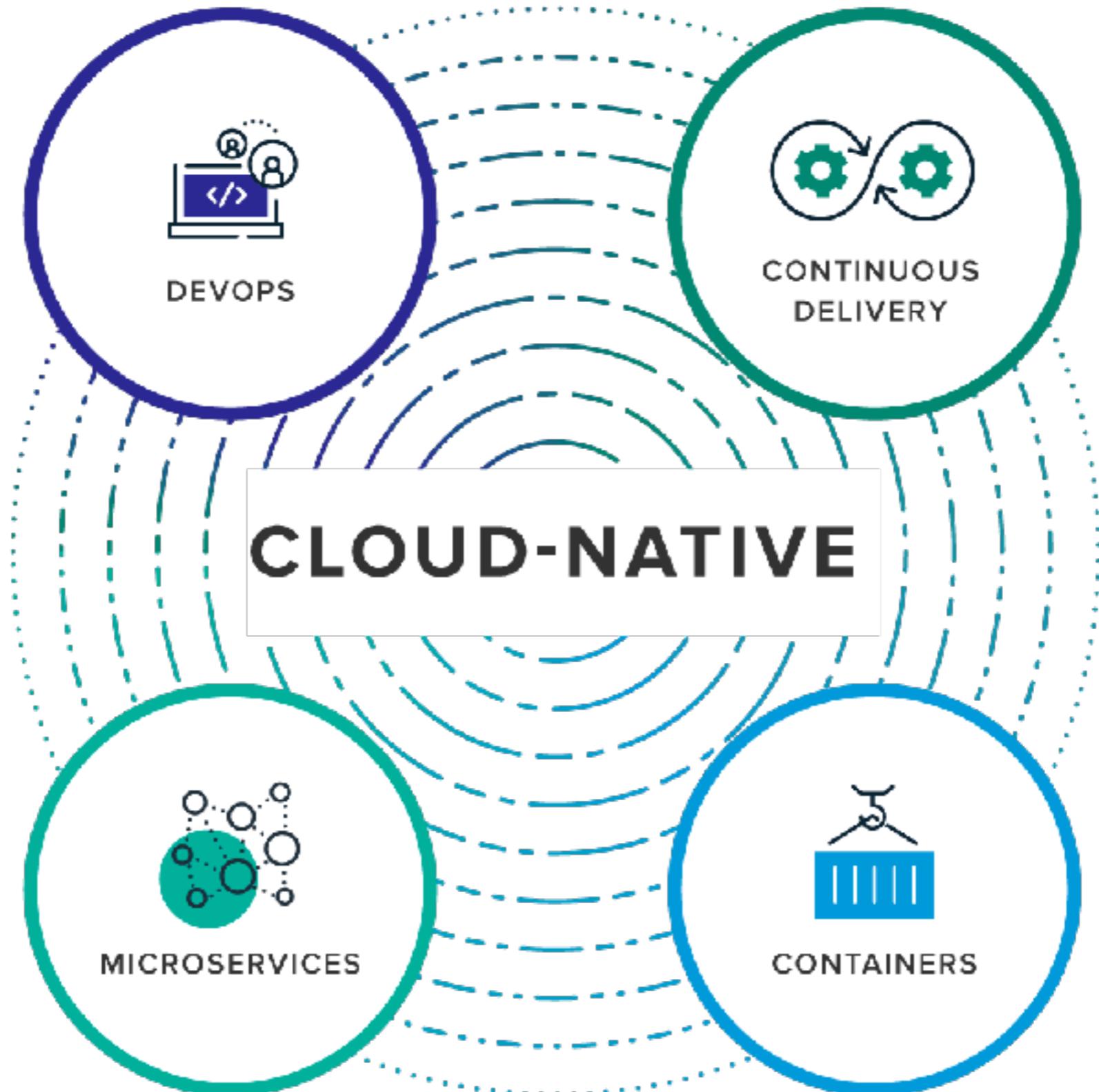
A photograph of a beach at sunset. The sky is a gradient from blue to orange and yellow. In the foreground, a couple is sitting on the sandy beach, facing each other. The ocean waves are visible, and a small island with trees is in the distance.

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



# We need Change





<https://pivotal.io/cloud-native>



# Let's start with basic



# How to develop ?



# Good Architecture ?



## THE LIFE OF A SOFTWARE ENGINEER.

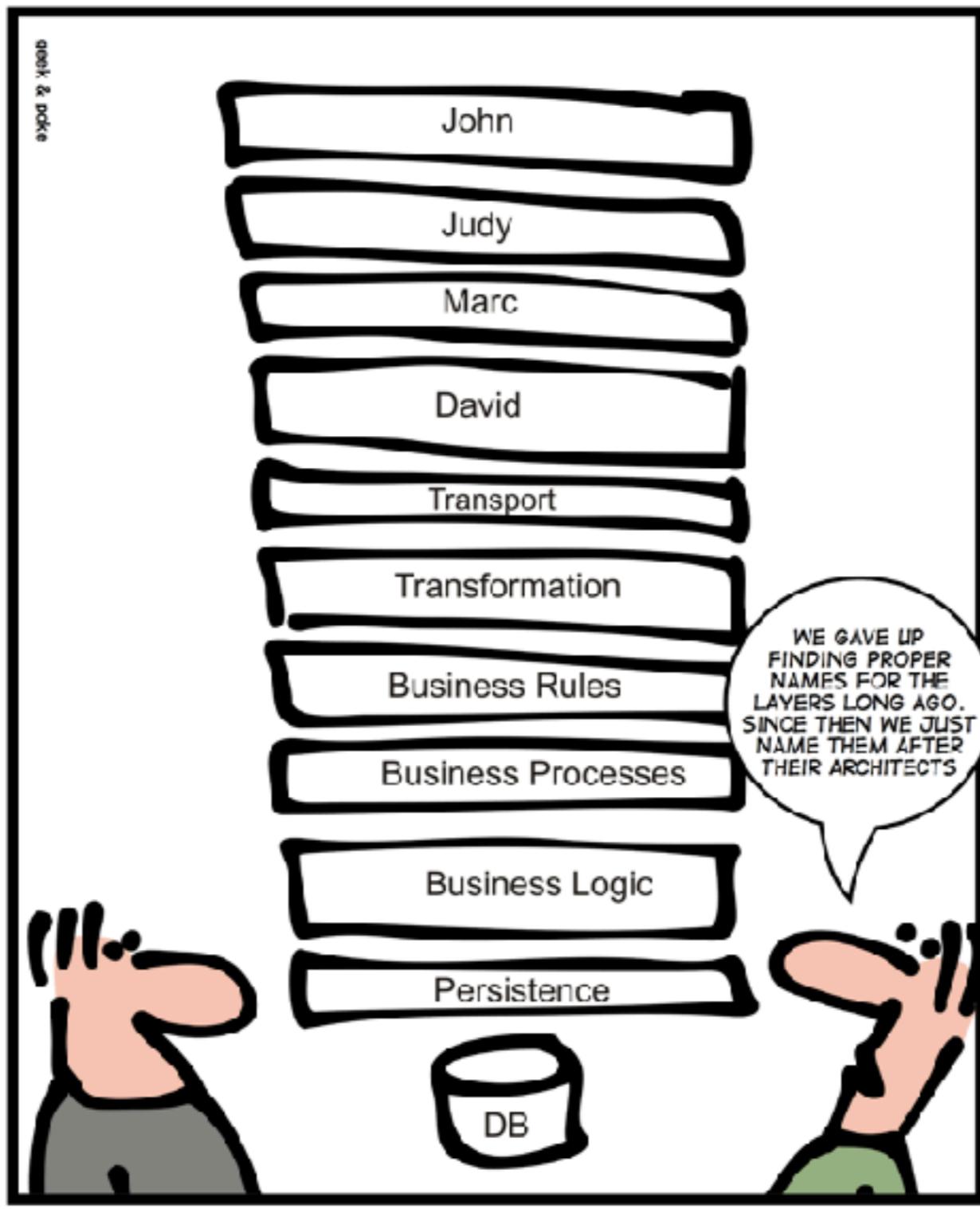
CLEAN SLATE. SOLID  
FOUNDATIONS. THIS TIME  
I WILL BUILD THINGS THE  
RIGHT WAY.



MUCH LATER...

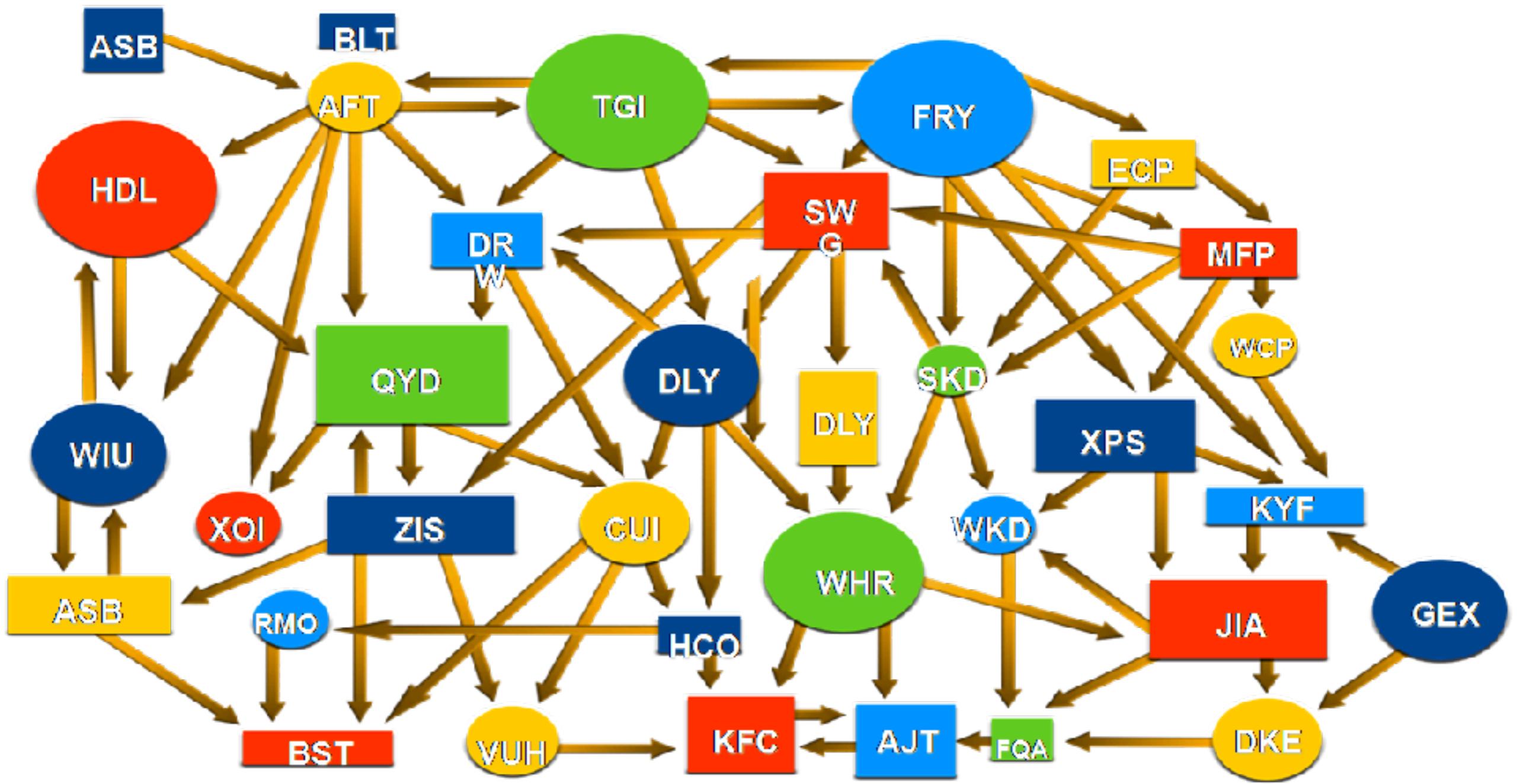
OH MY. I'VE  
DONE IT AGAIN,  
HAVEN'T I ?

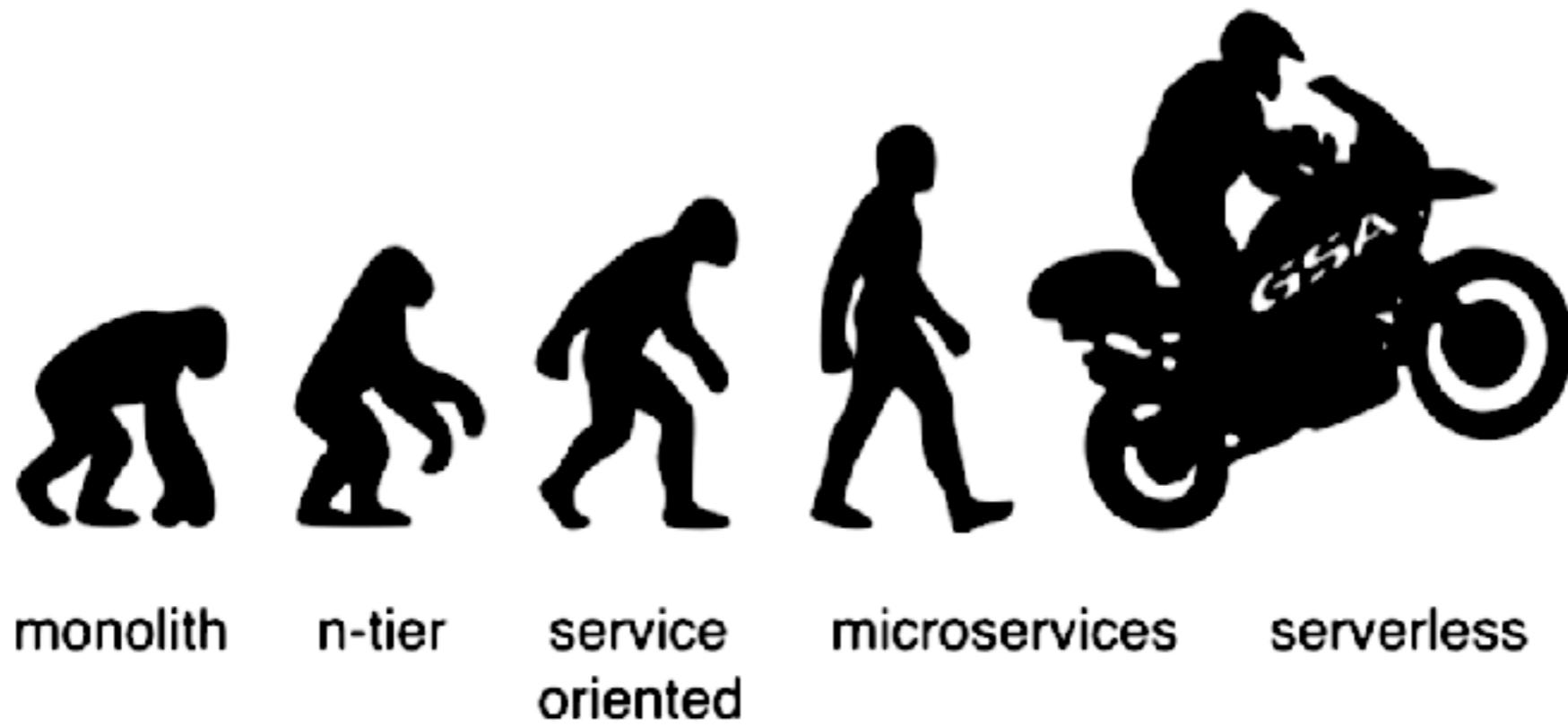




A GOOD ARCHITECT LEAVES A FOOTPRINT



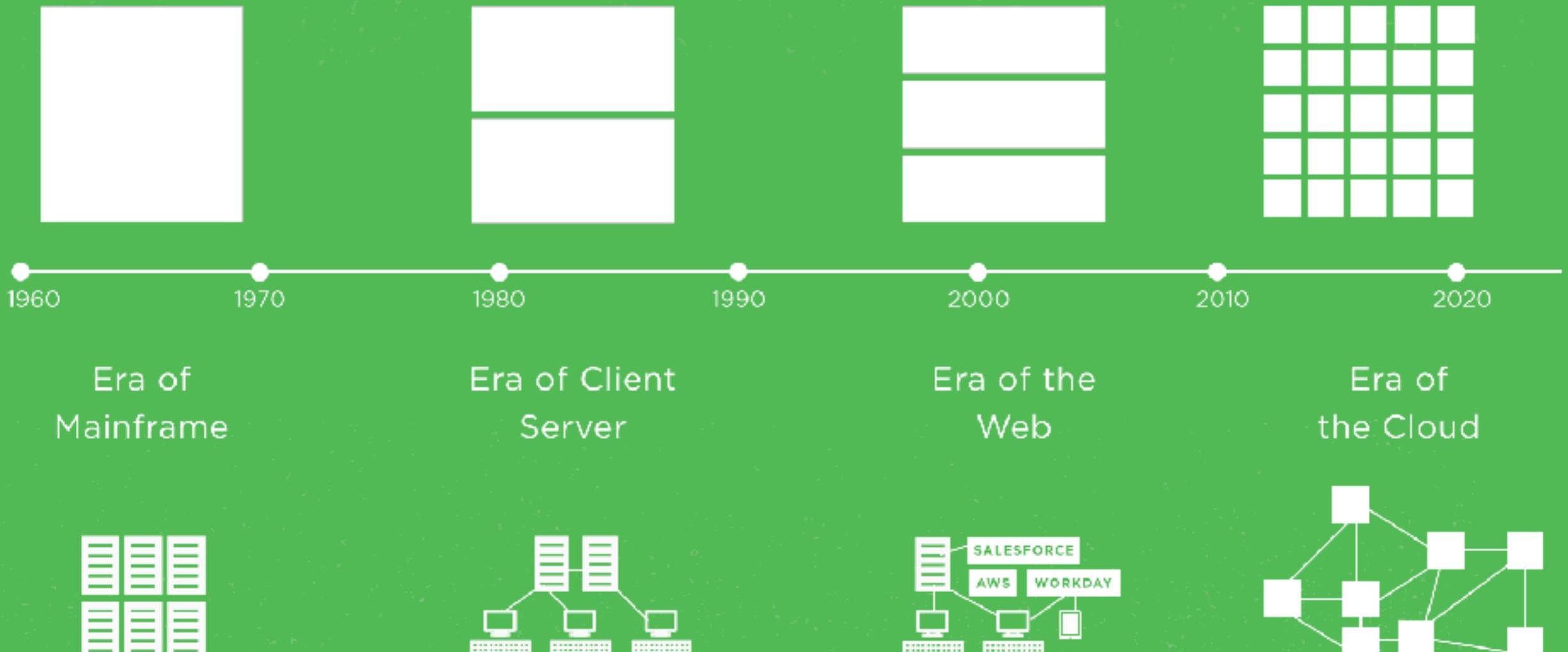




# Evolution of Architecture



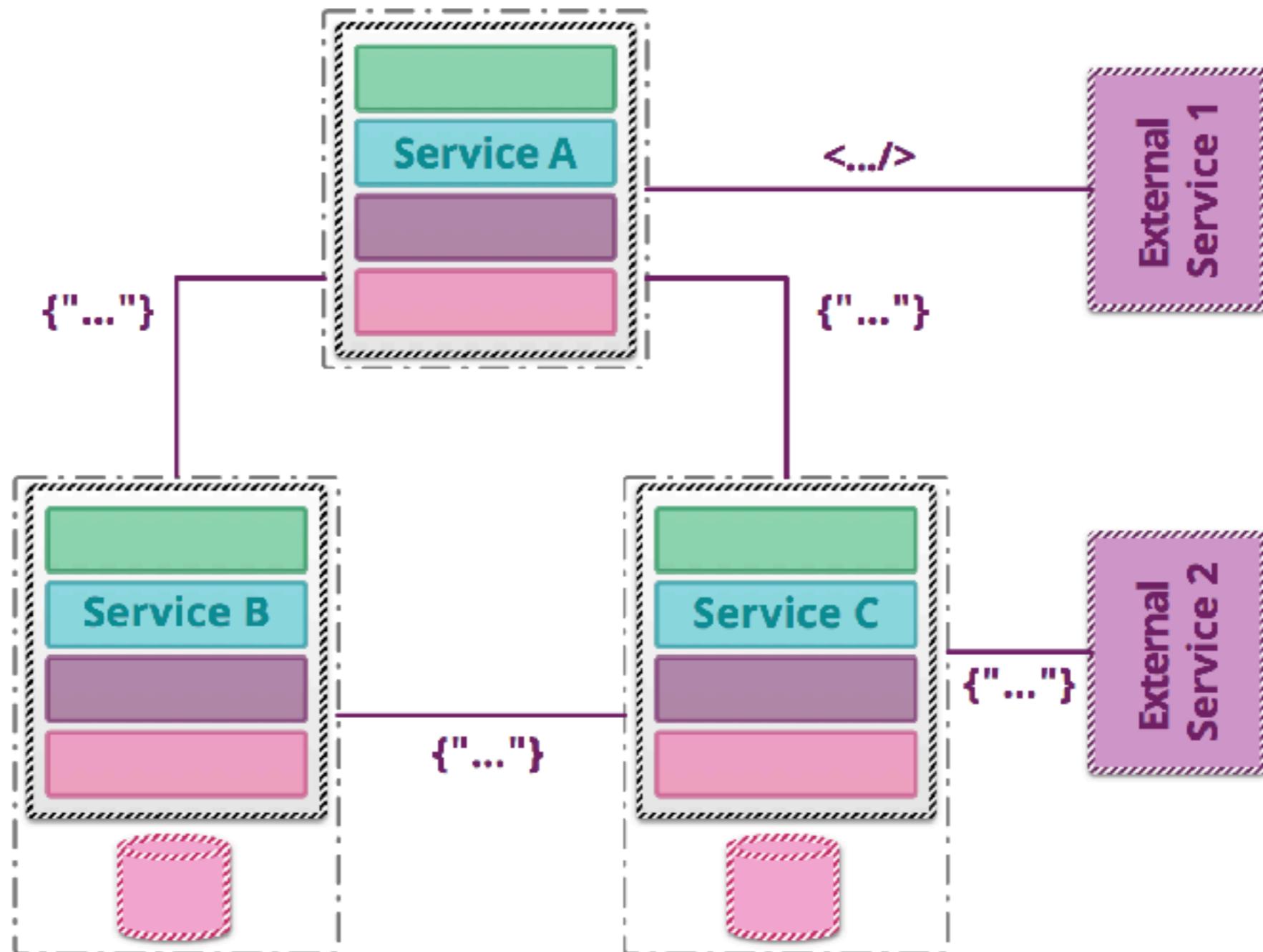
## ABSTRACTION IN THE ENTERPRISE



# How to Test ?



# Multiple services



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

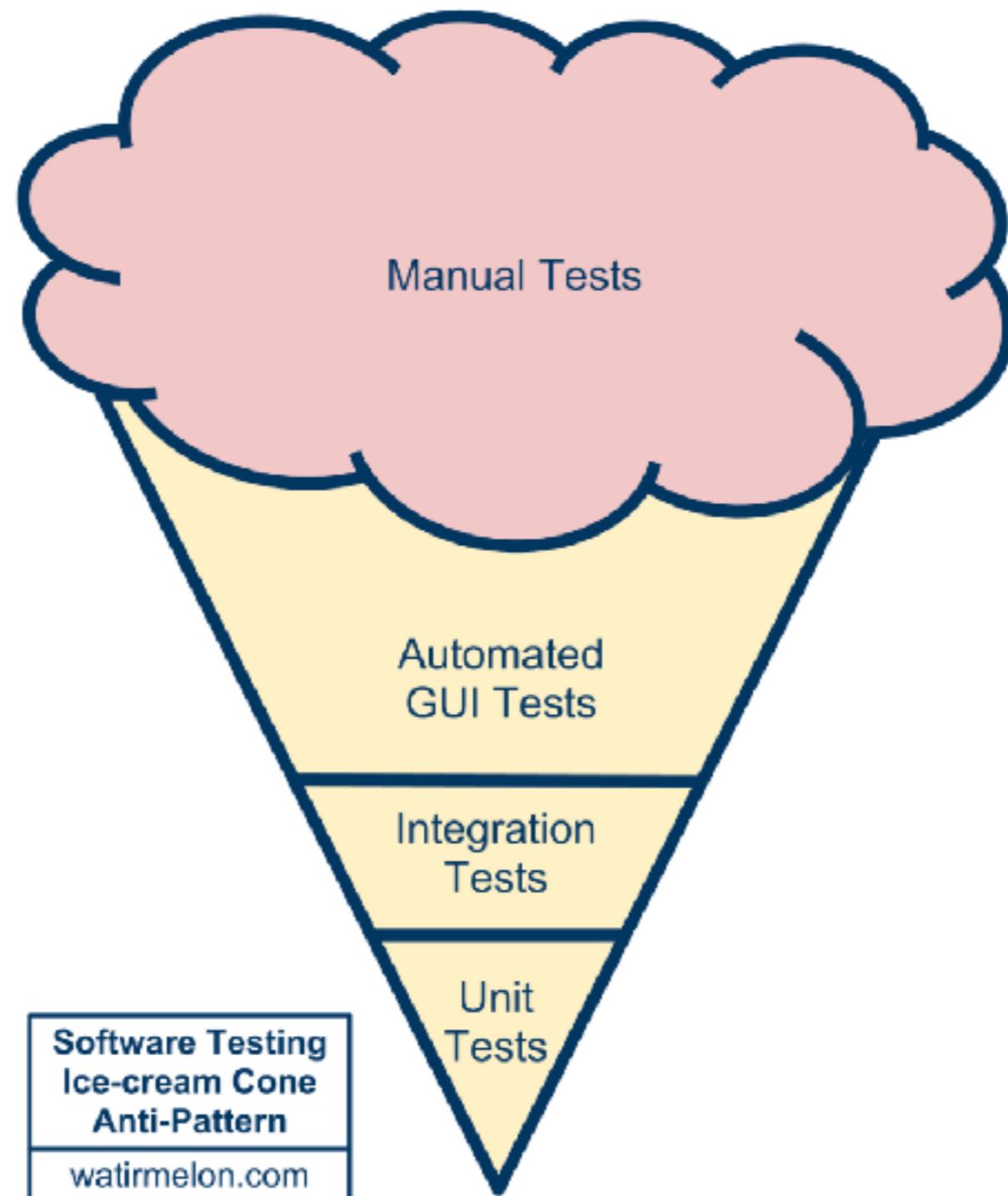


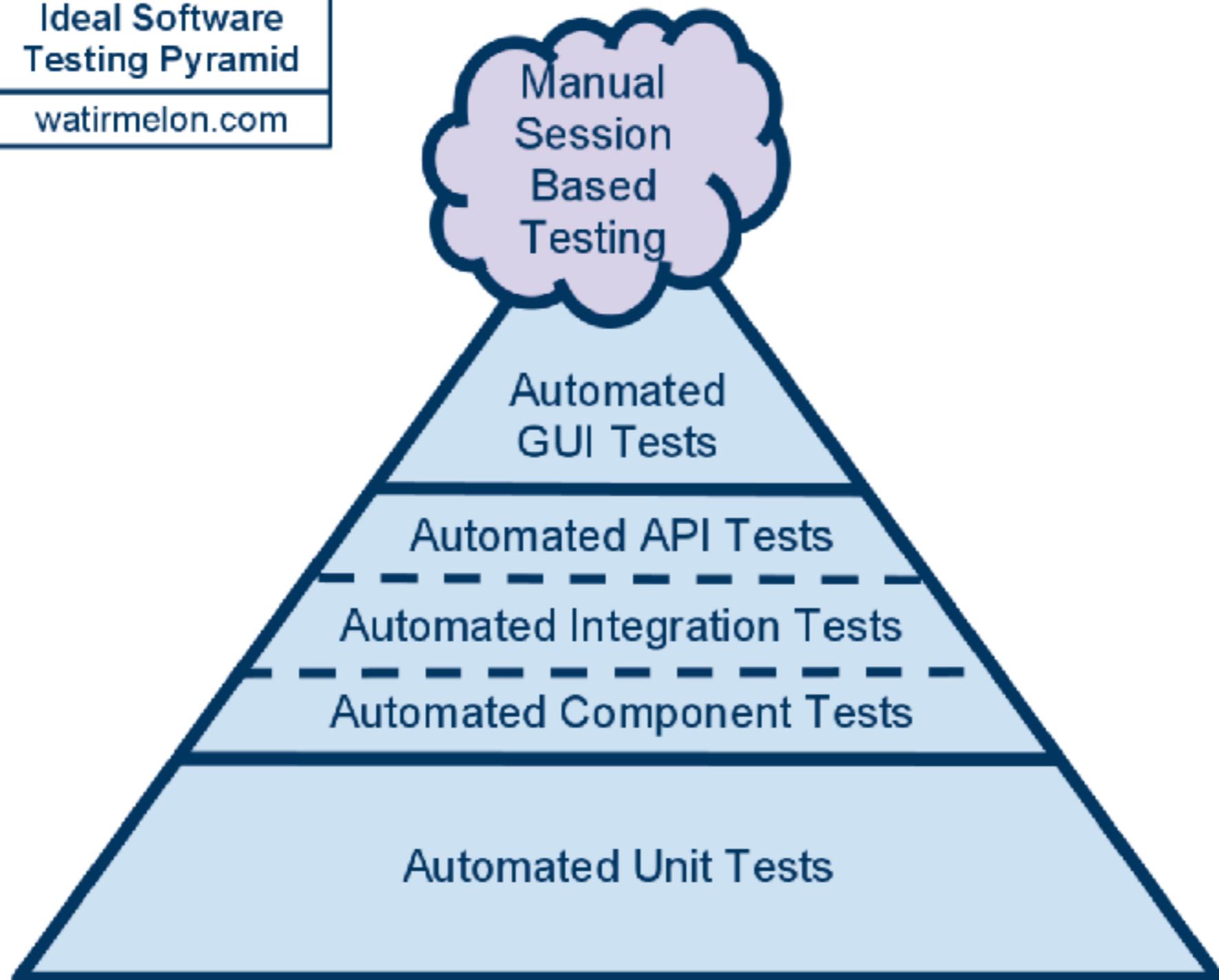
THE #1 PROGRAMMER EXCUSE  
FOR LEGITIMATELY SLACKING OFF:  
"TESTS ARE RUNNING"

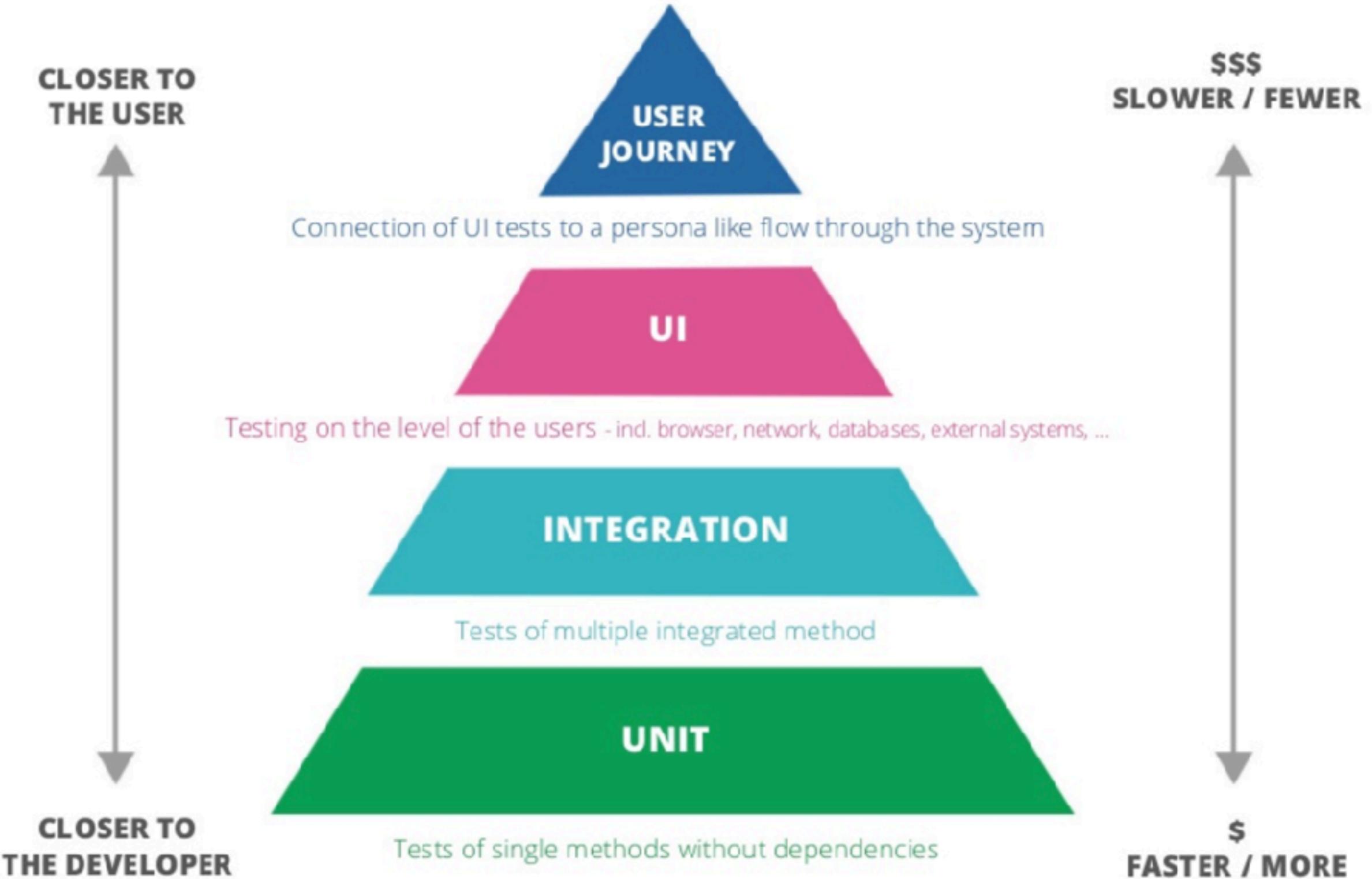


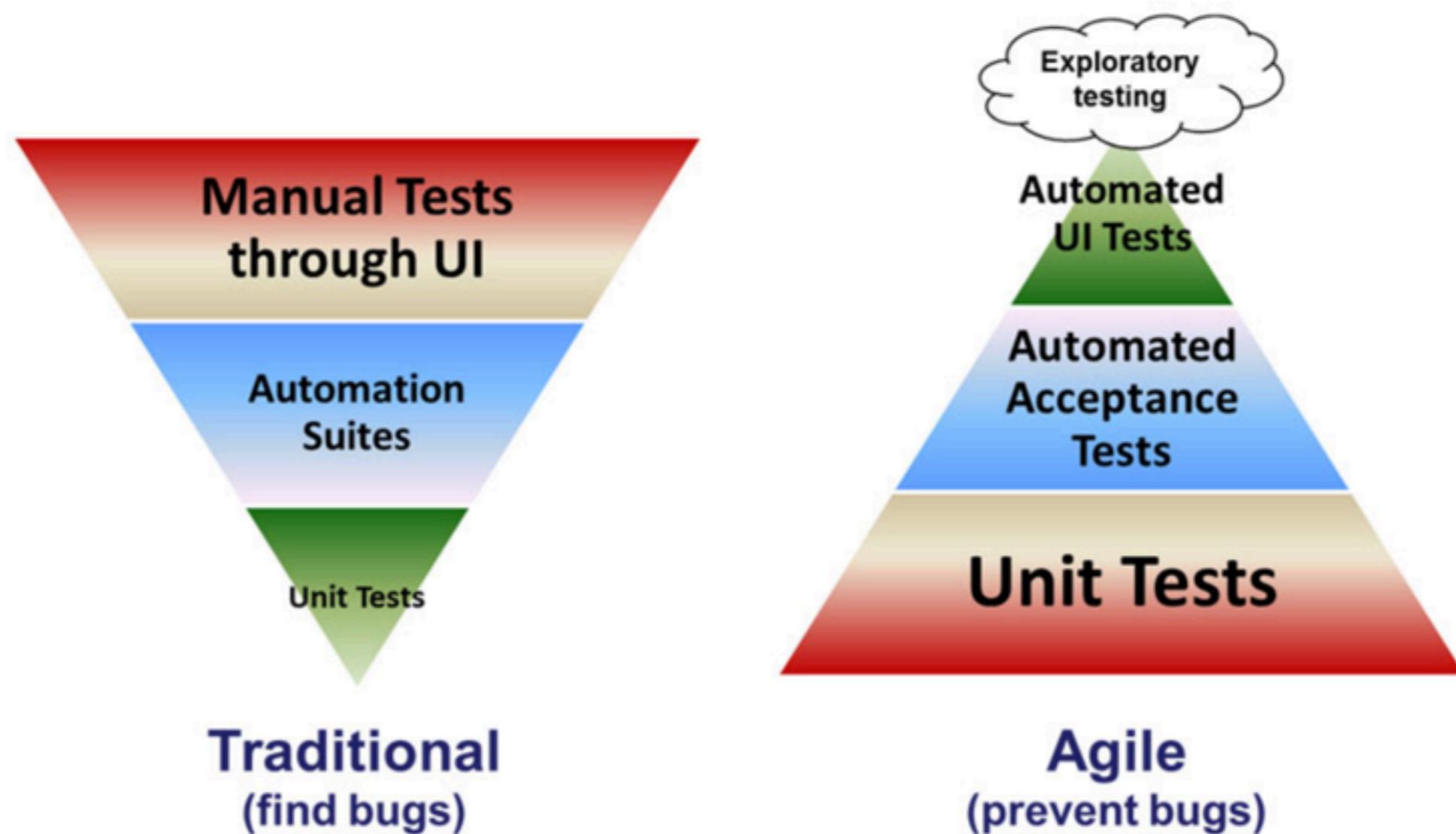
# Testing

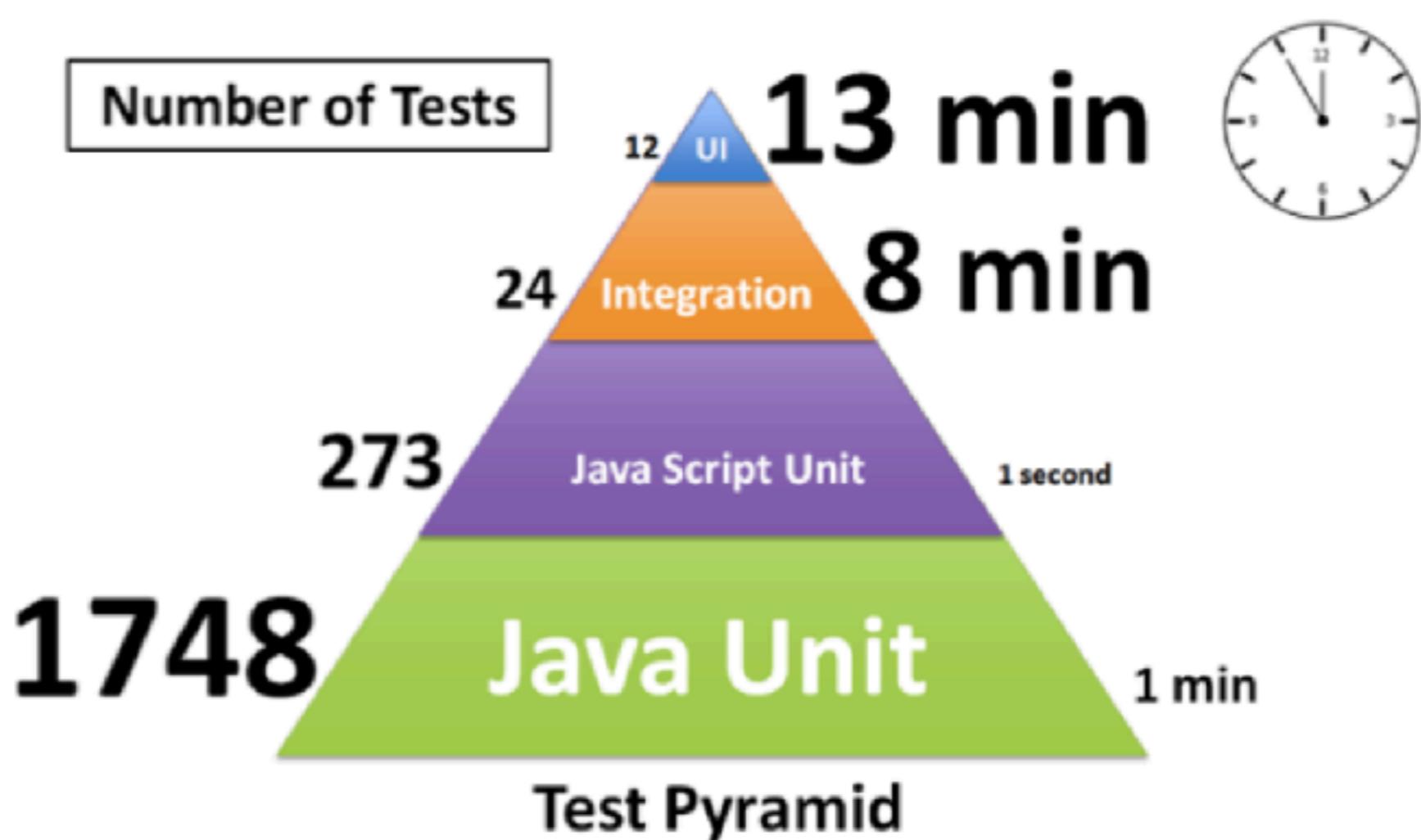


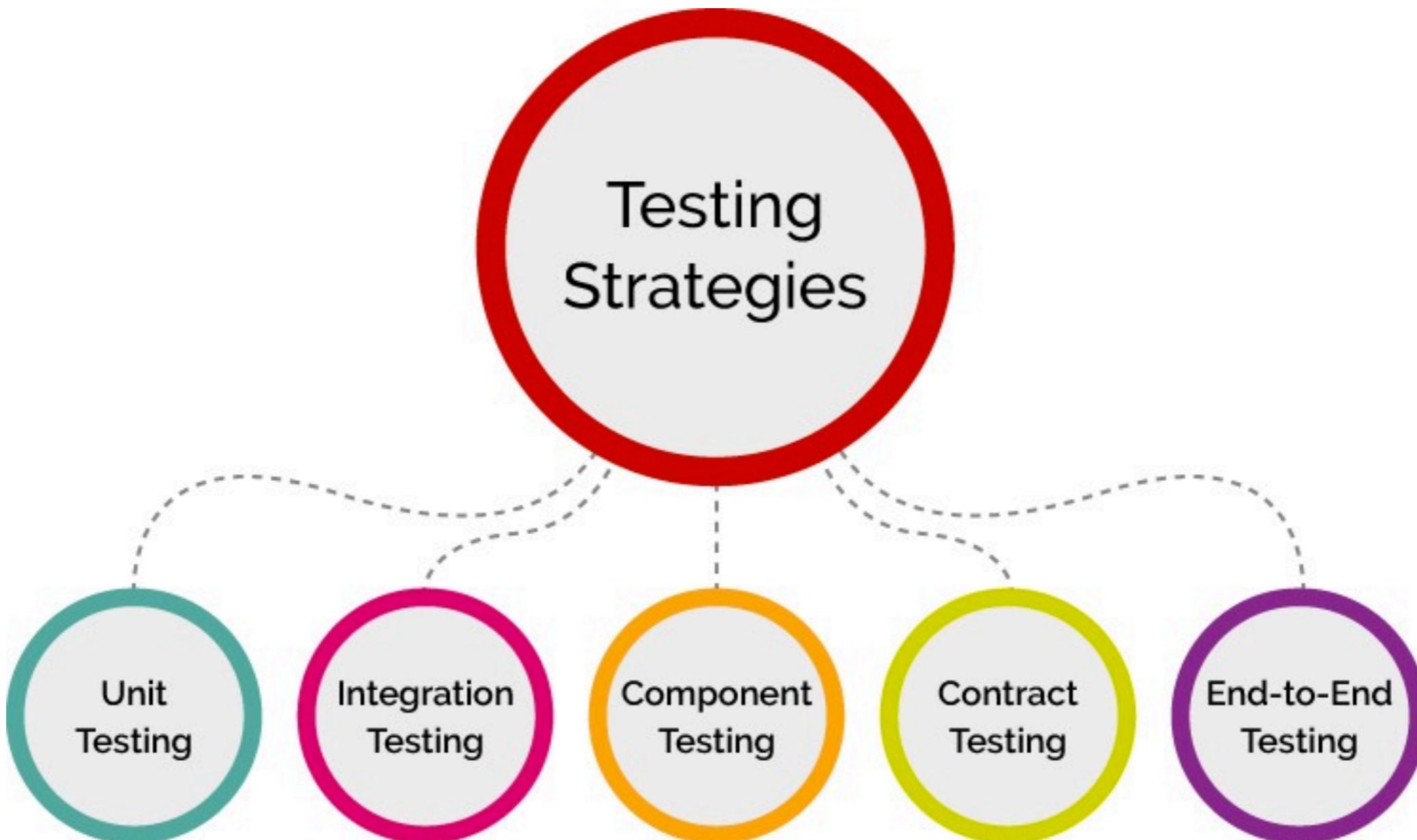




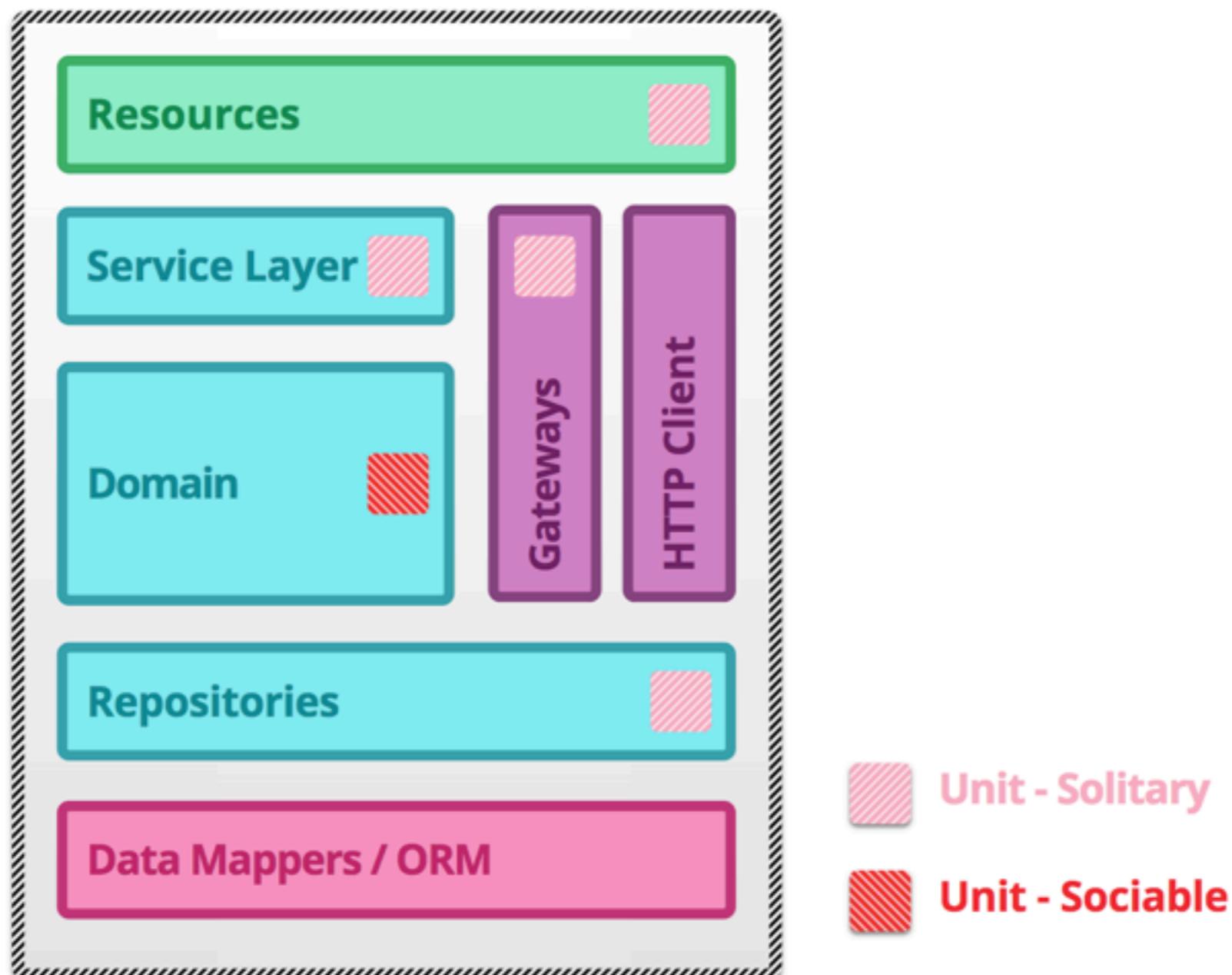




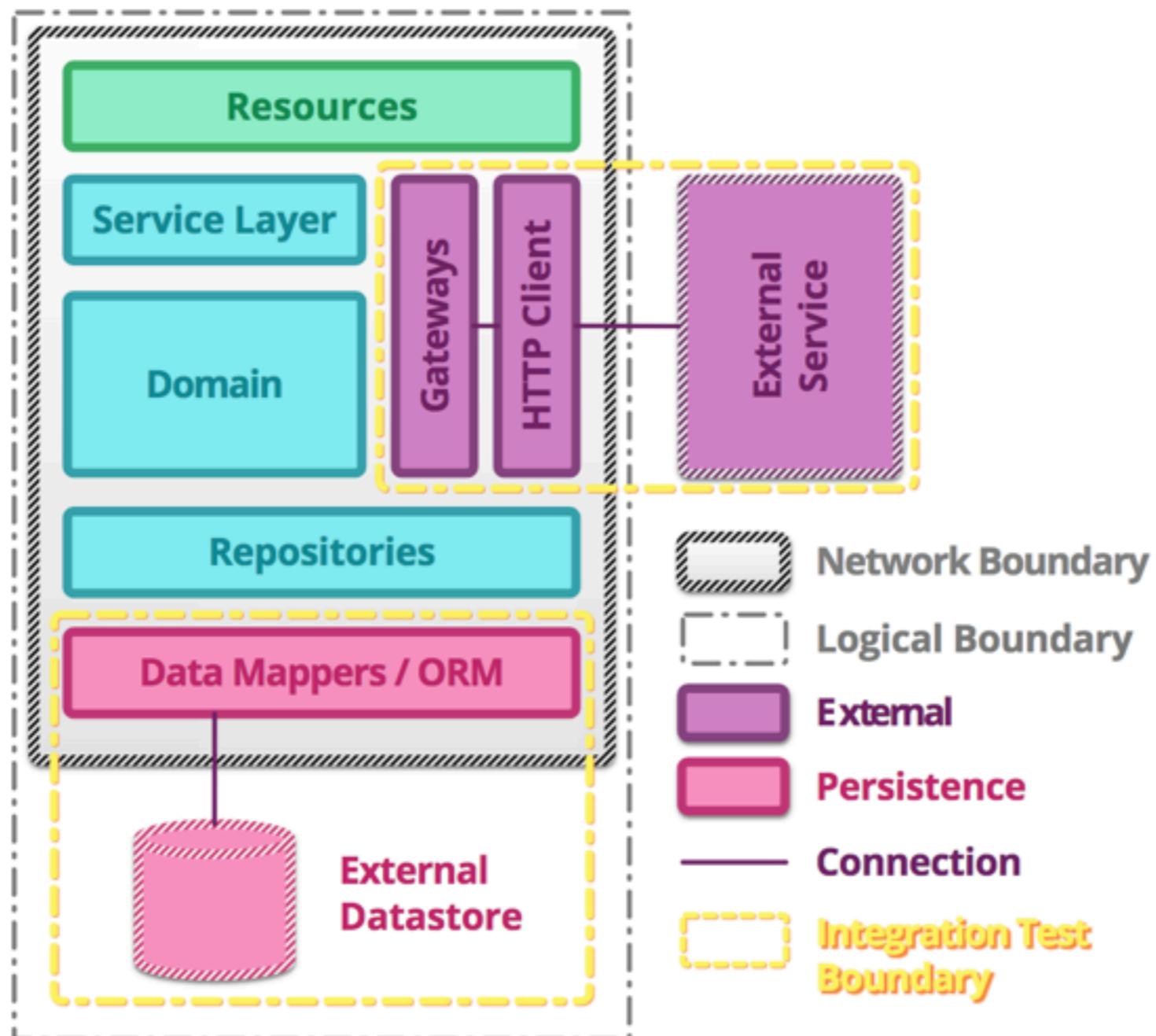




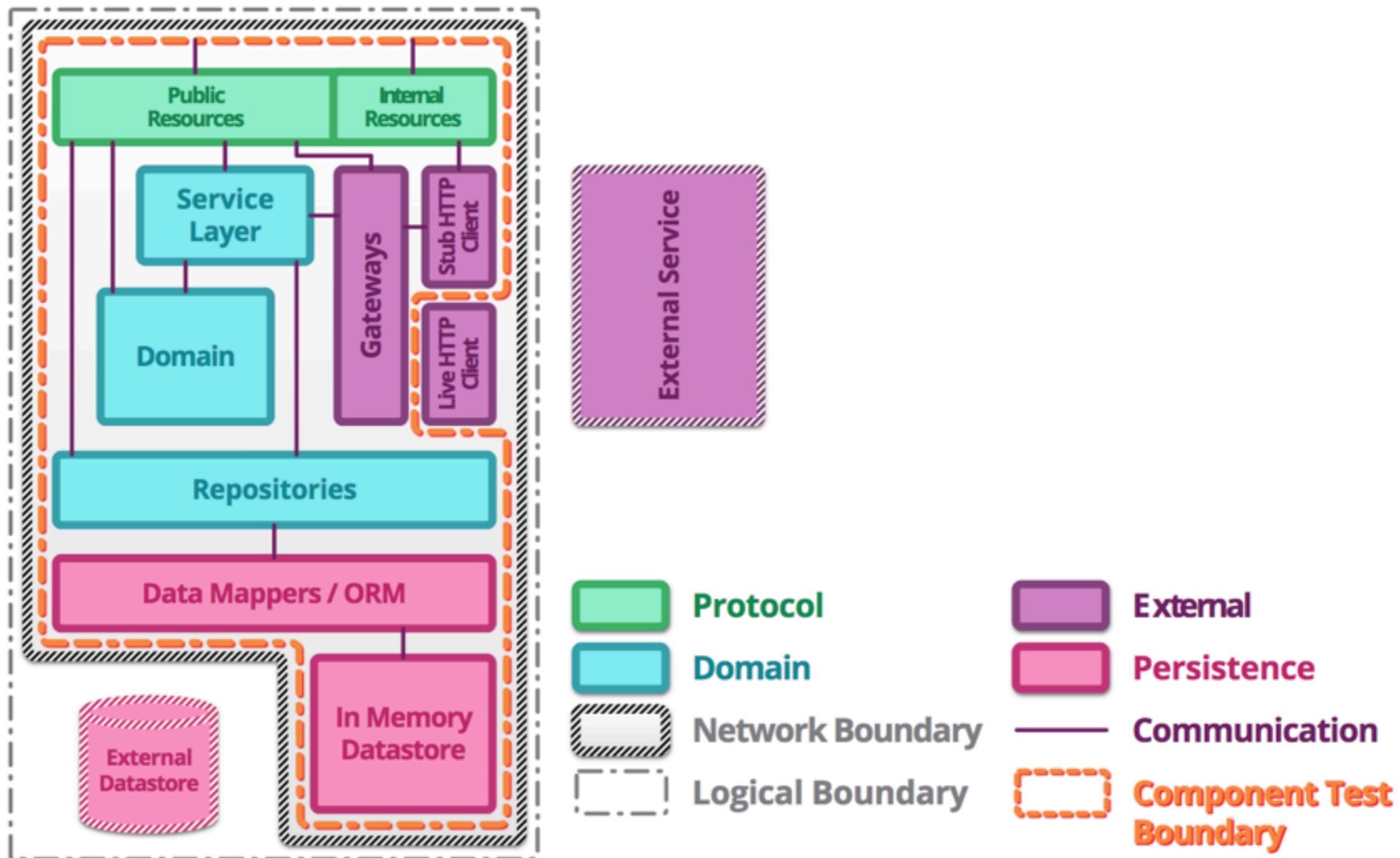
# Unit testing



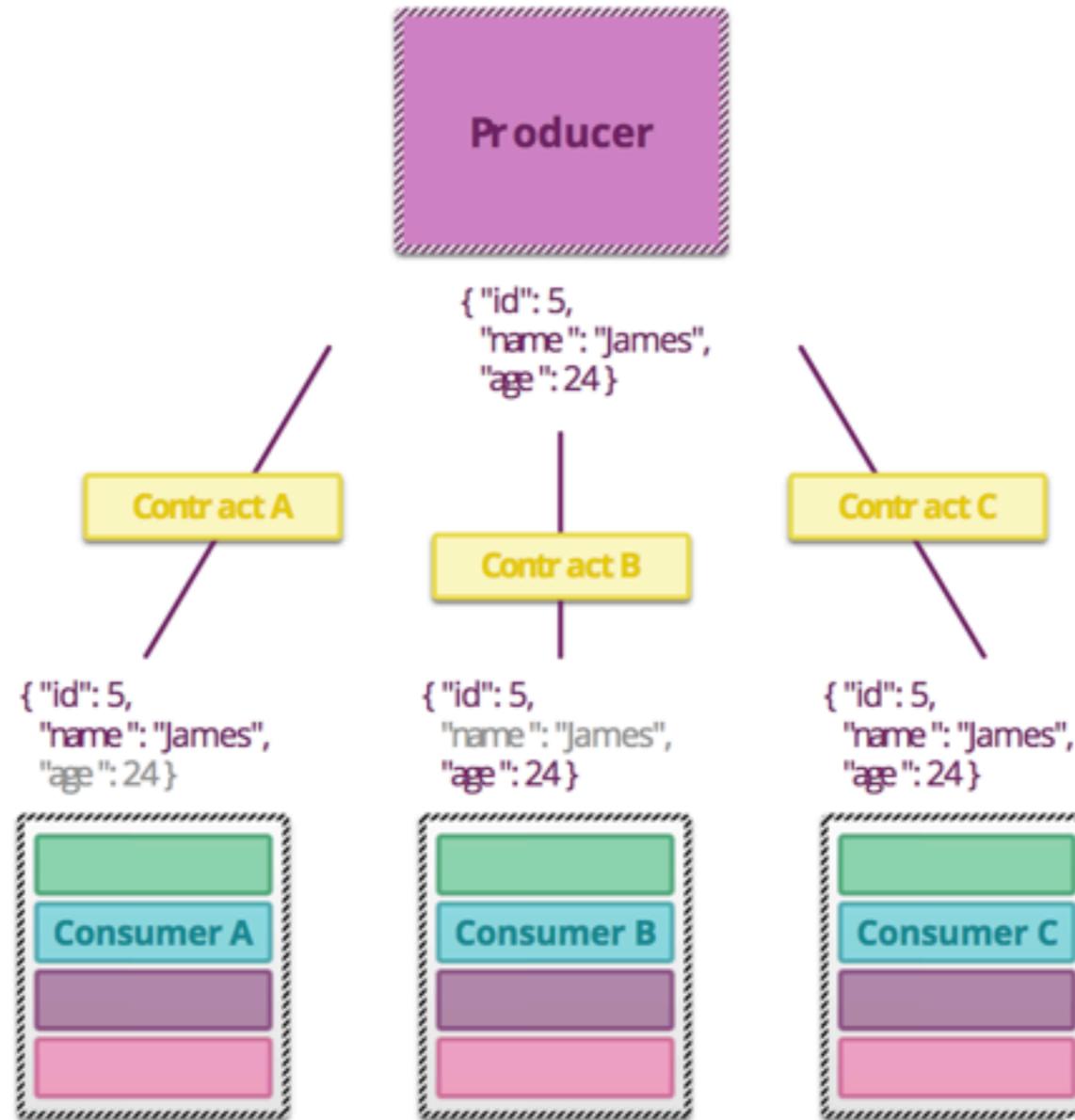
# Integration testing



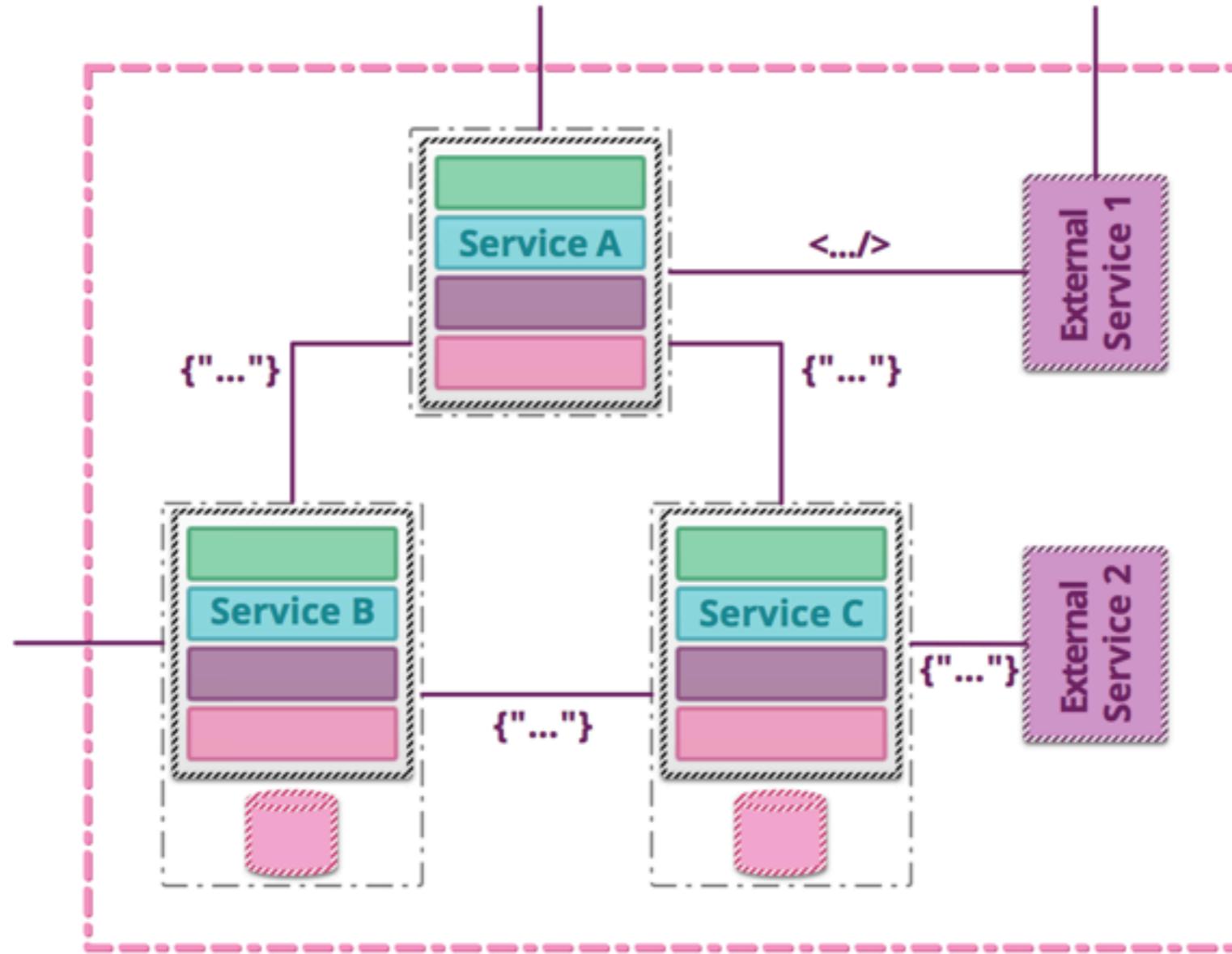
# Component testing



# Contract testing



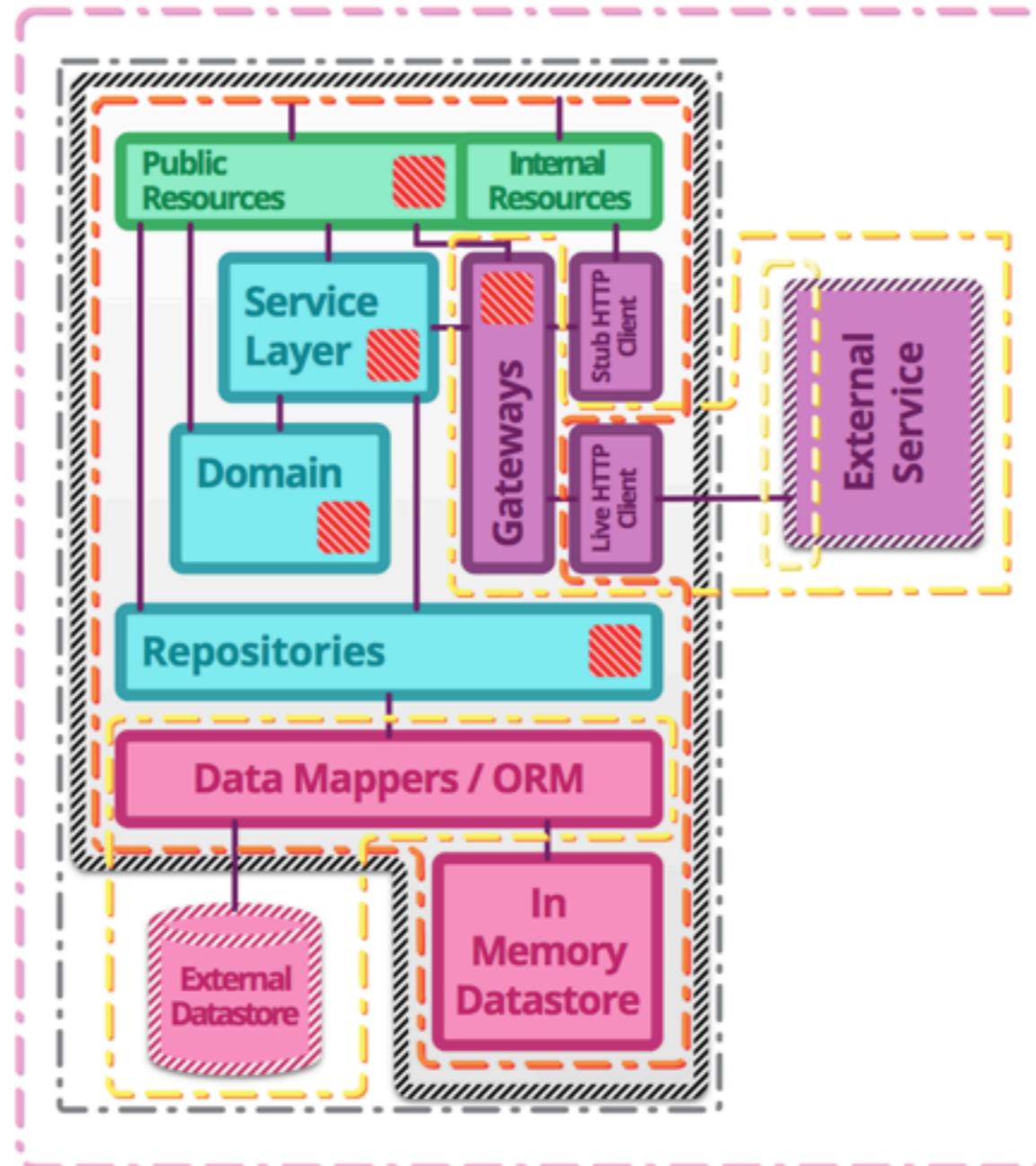
# End-to-End testing



# 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 ?

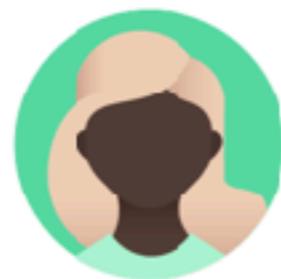


# Deployment Smells



## 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 than 15 mins

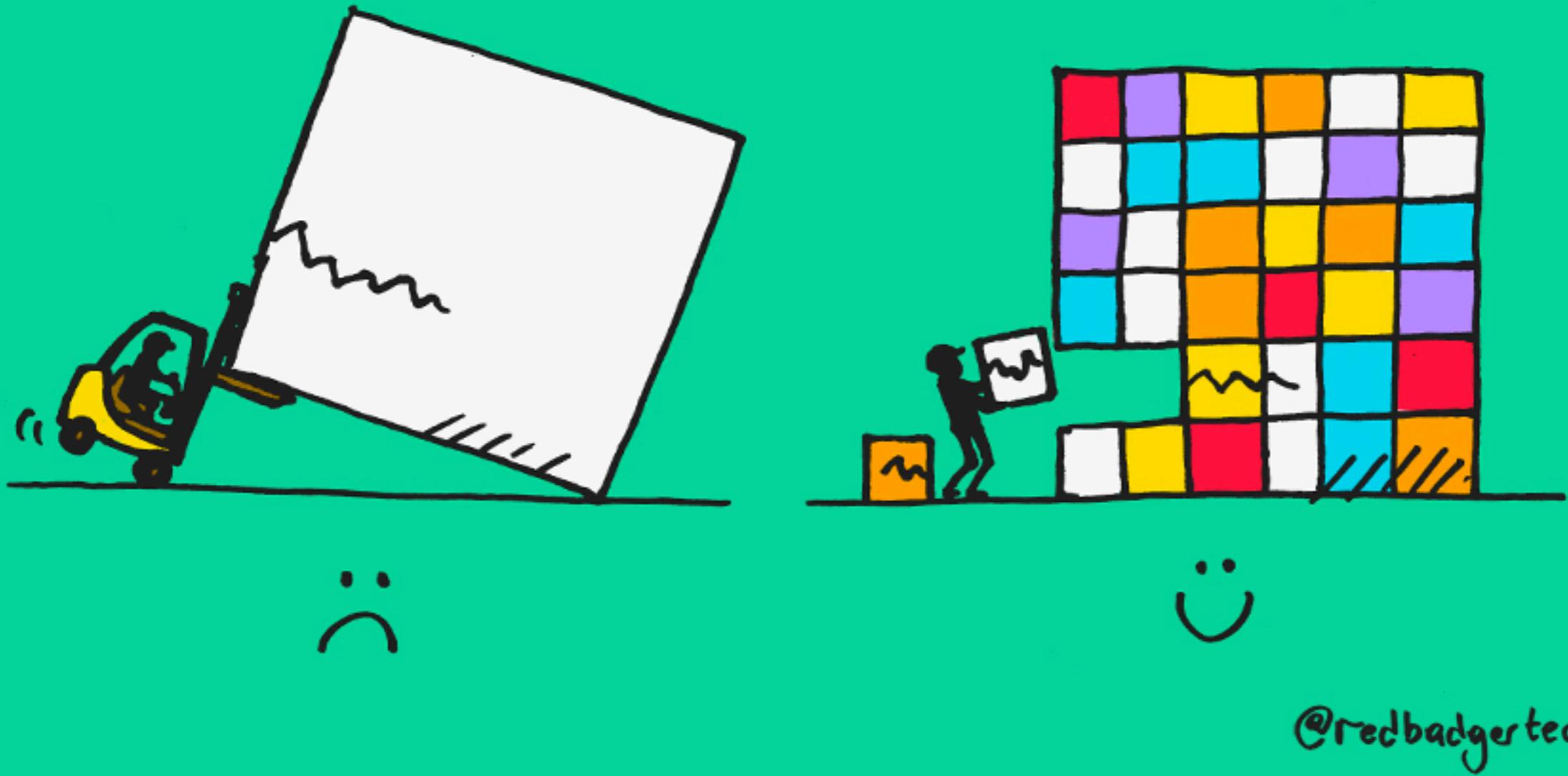
Setting it up should be quick and initial deployment should quick

<https://trello.com/>



# Deployment

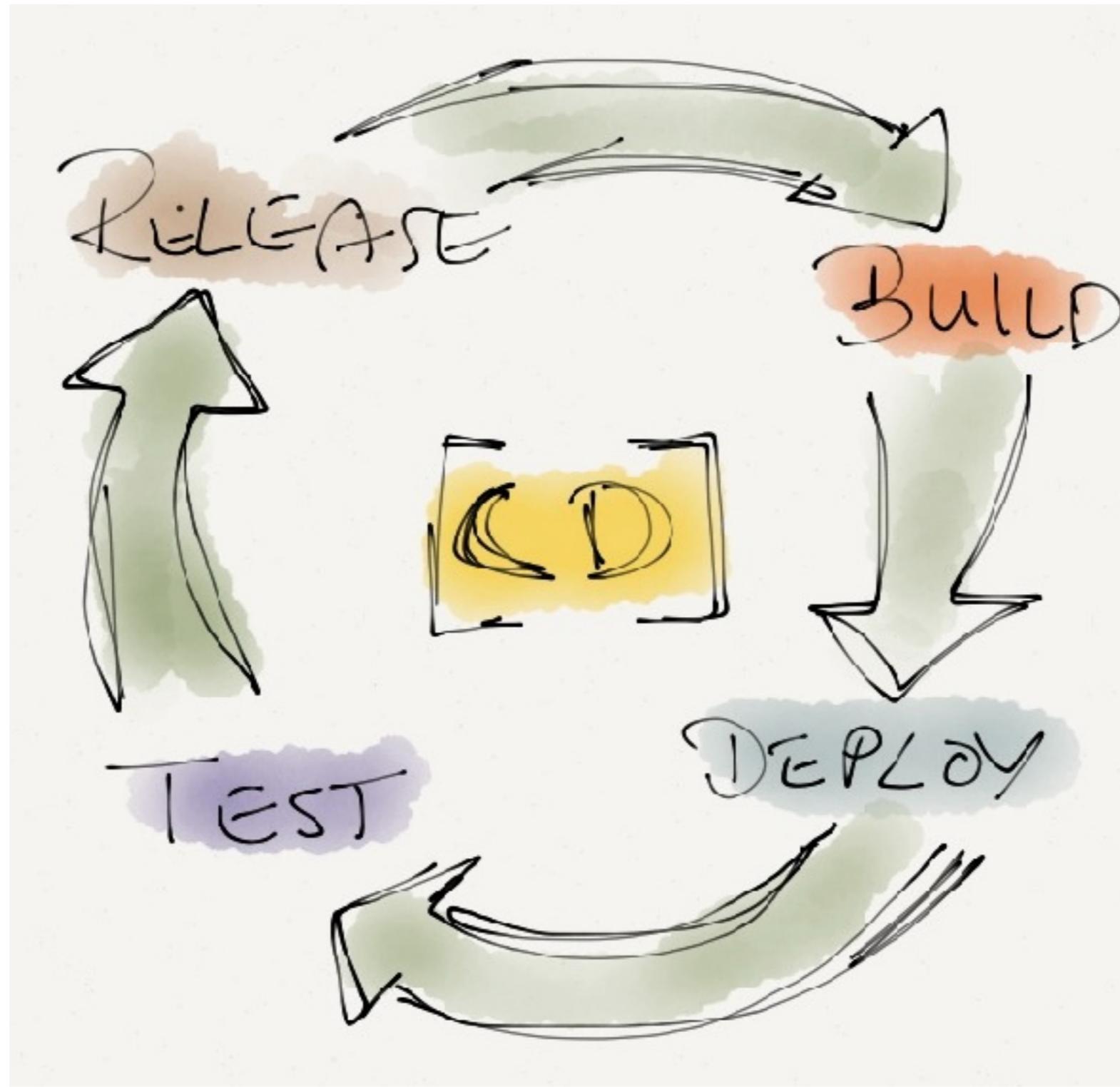




# Deploy vs Release



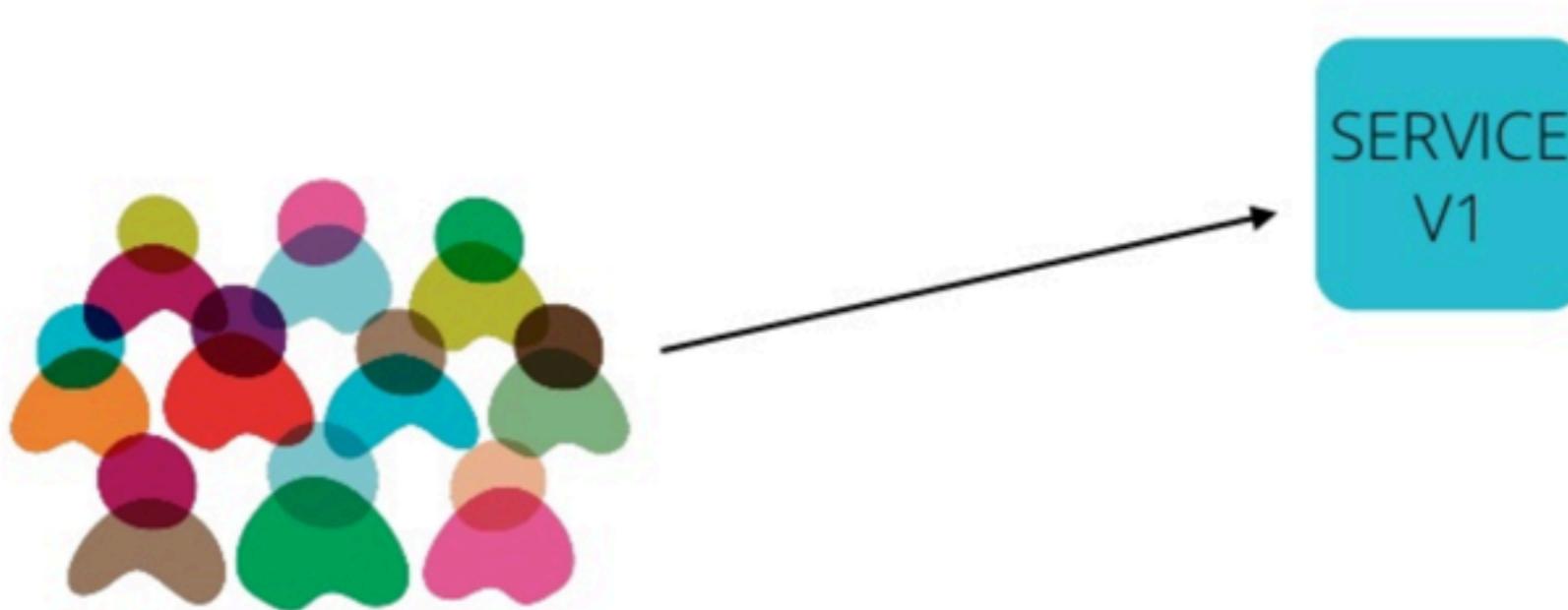
# Deploy vs Release



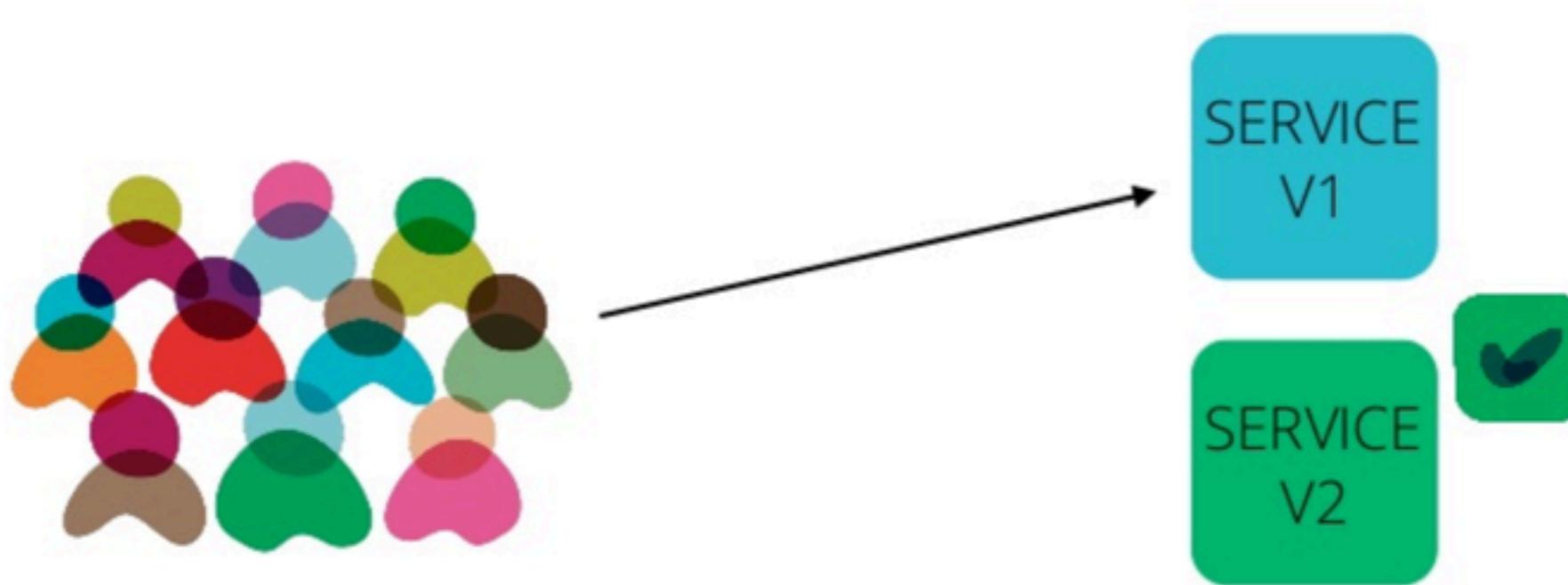
# Blue Green Deployment



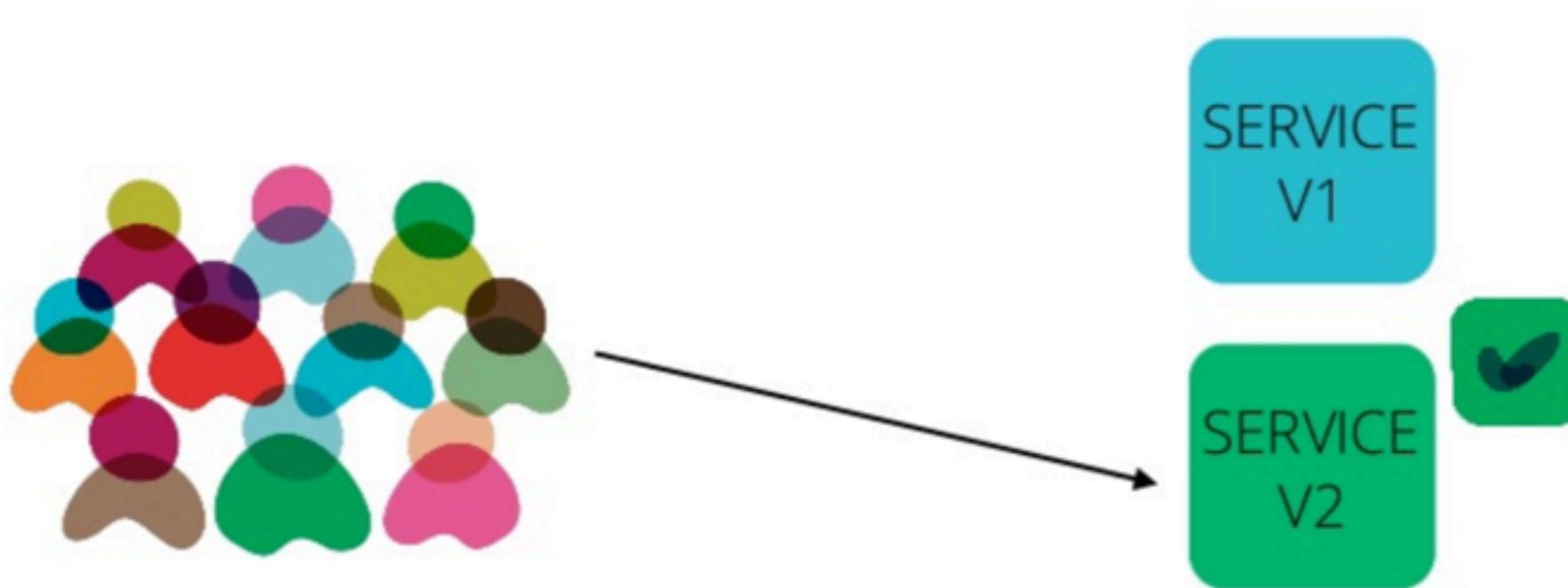
# Blue Green Deployment



# Blue Green Deployment



# Blue Green Deployment



# Canary Release



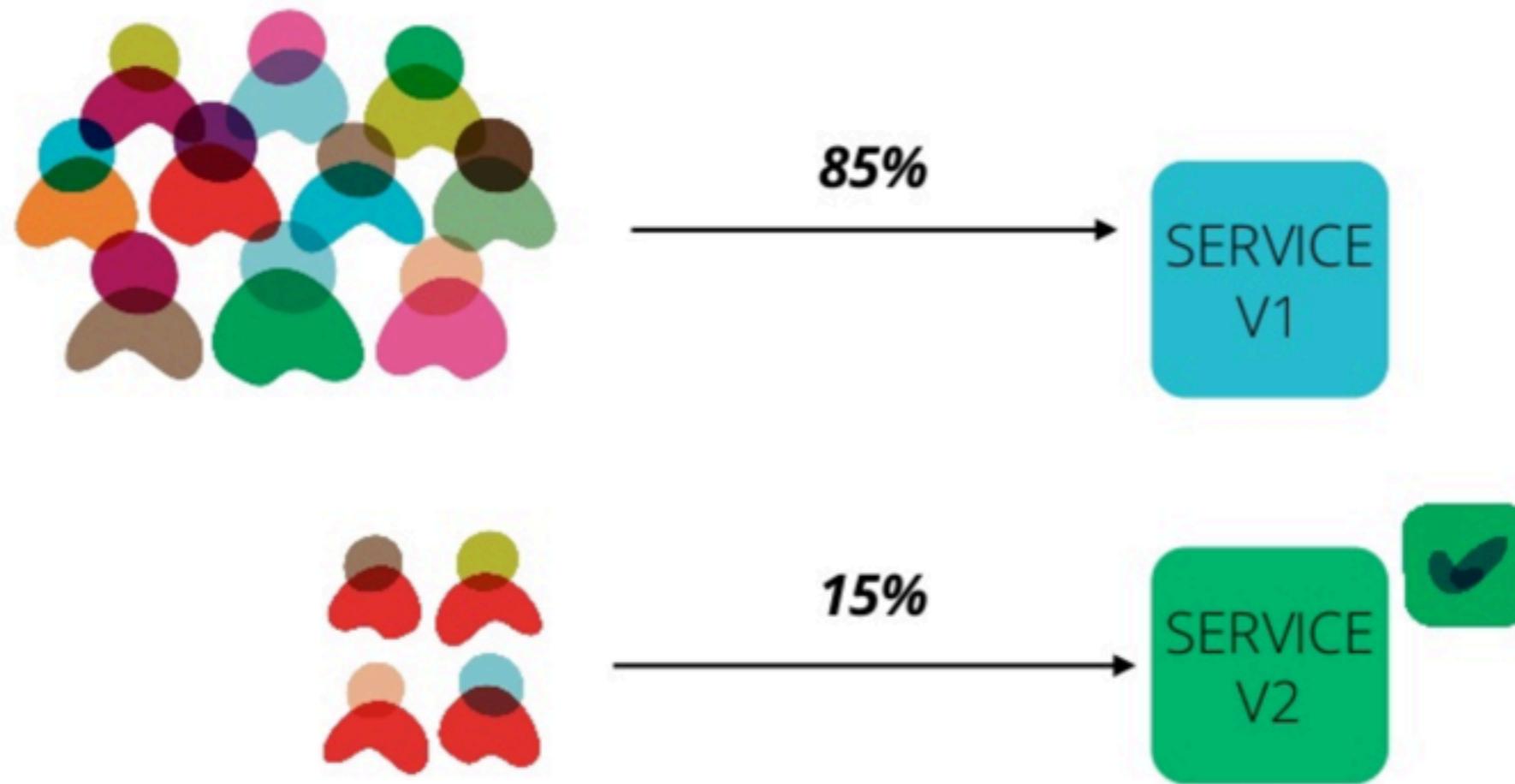
# Canary Release



# Canary Release



# Canary Release



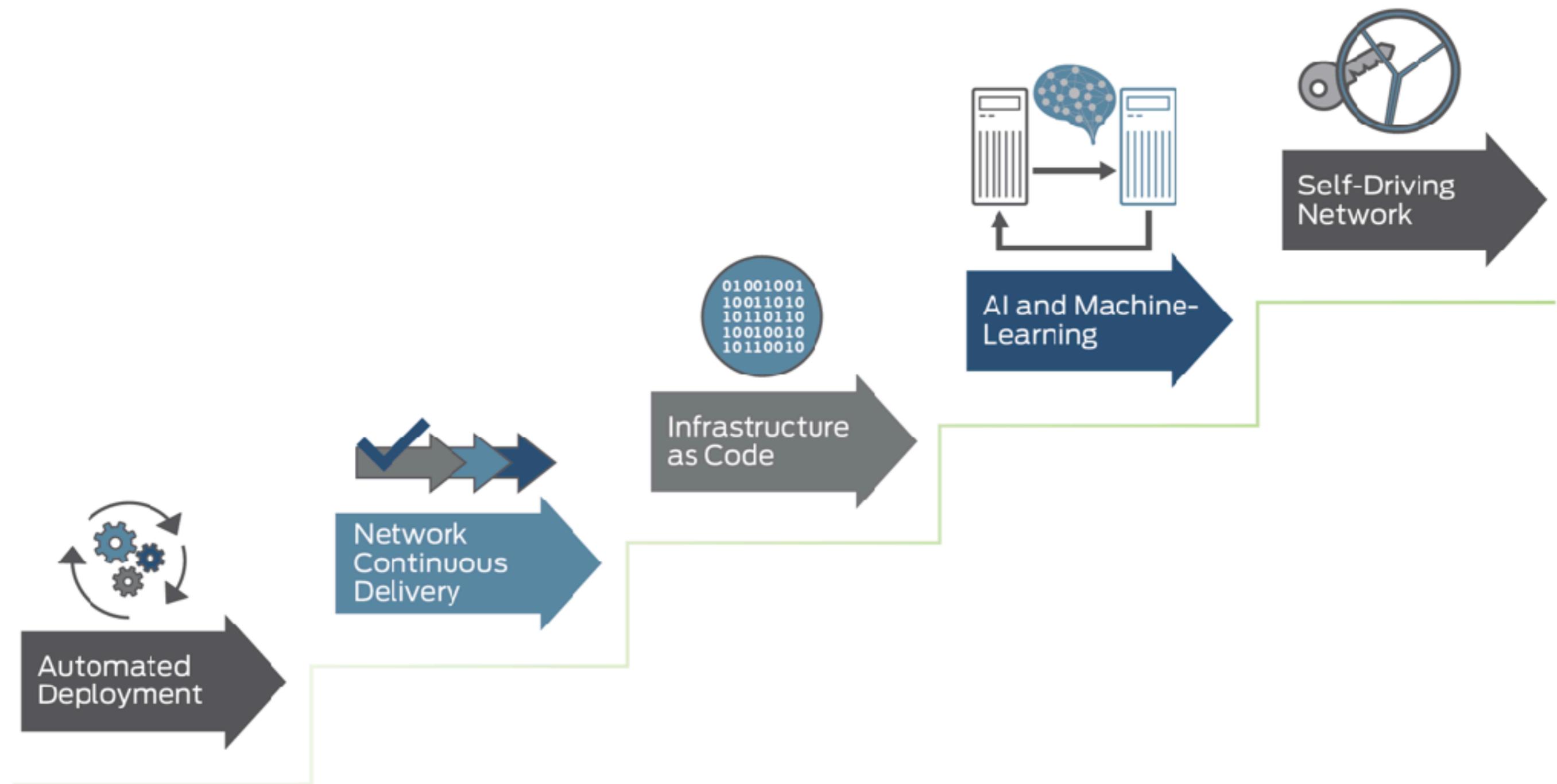
# > 500 services !!



# 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
```





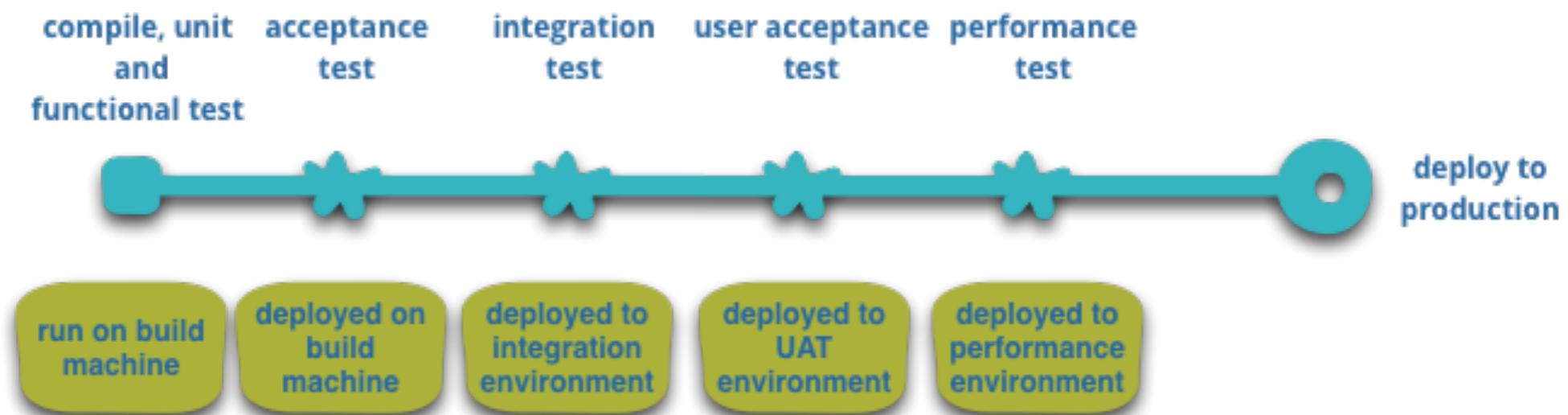
<https://www.juniper.net/us/en/products-services/what-is/iac/>



# Current situation !!

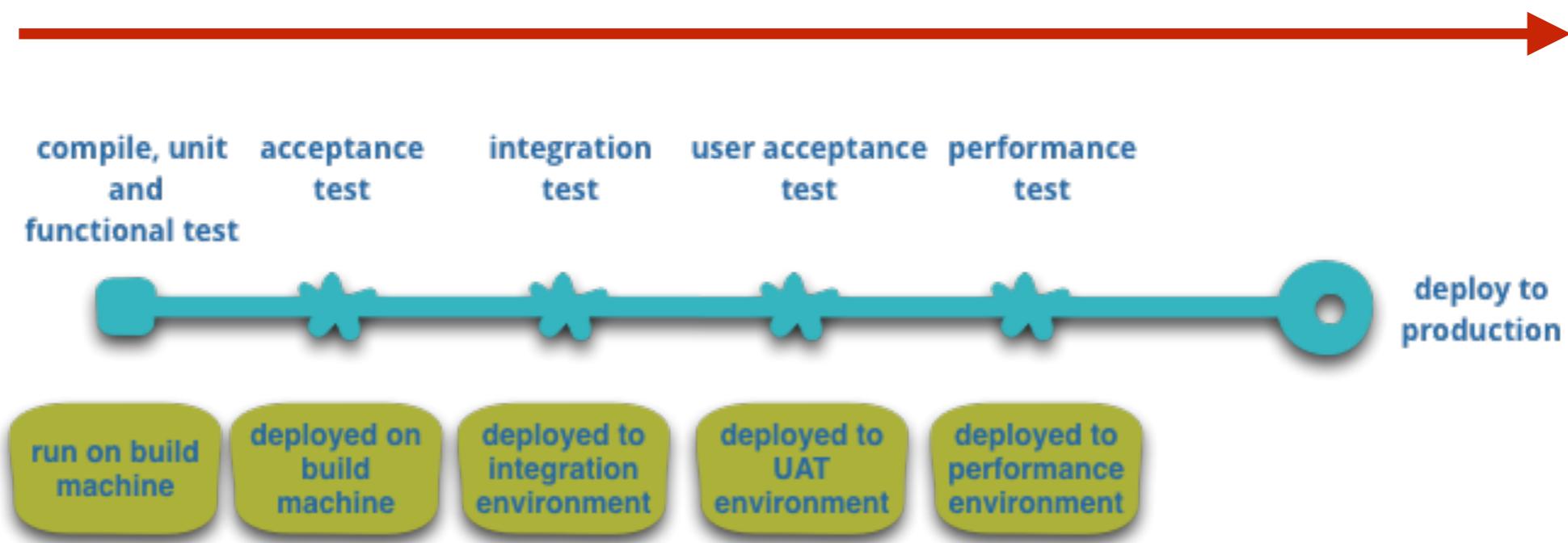


# Infrastructure Automation

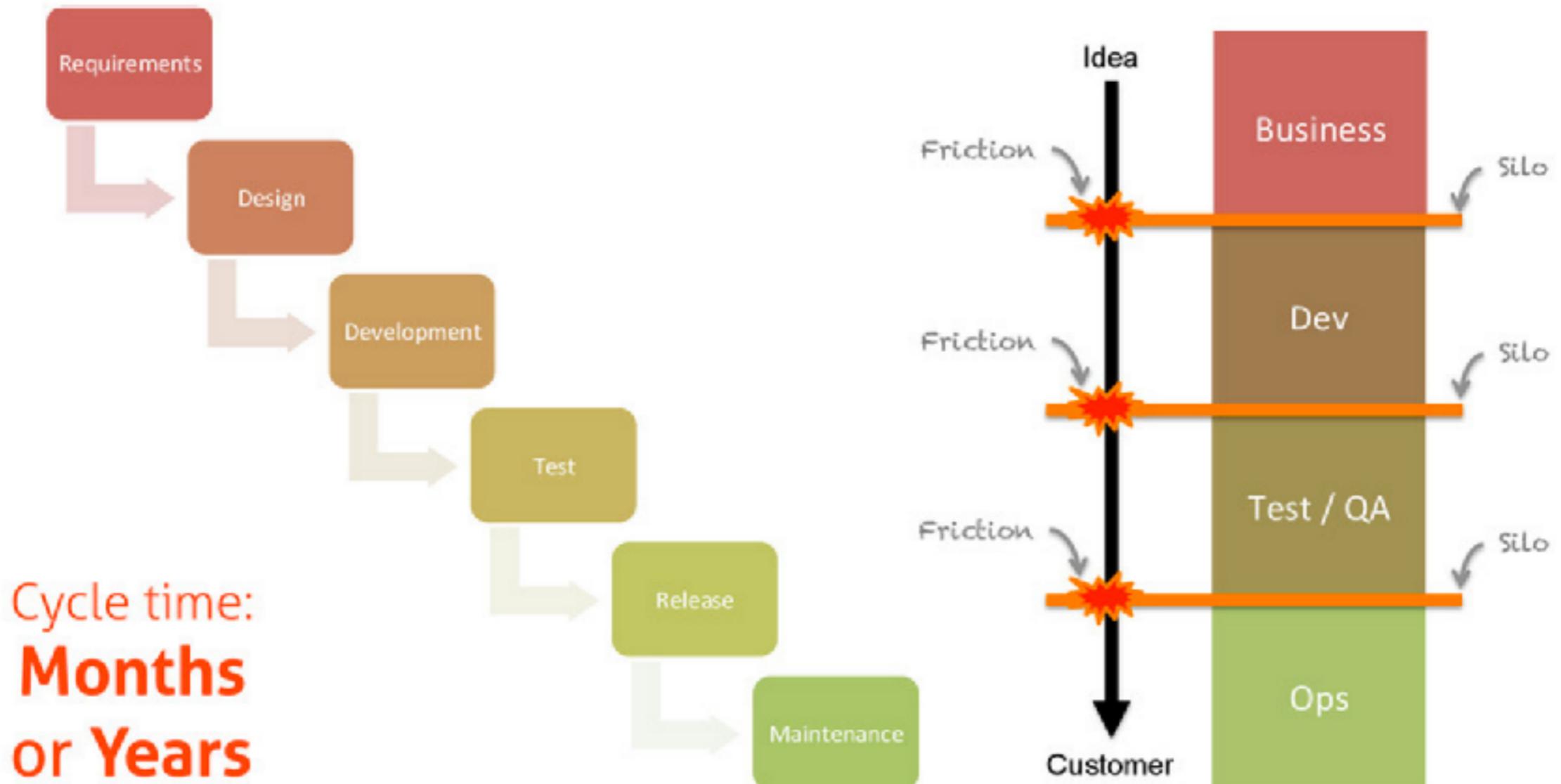


# Infrastructure Automation

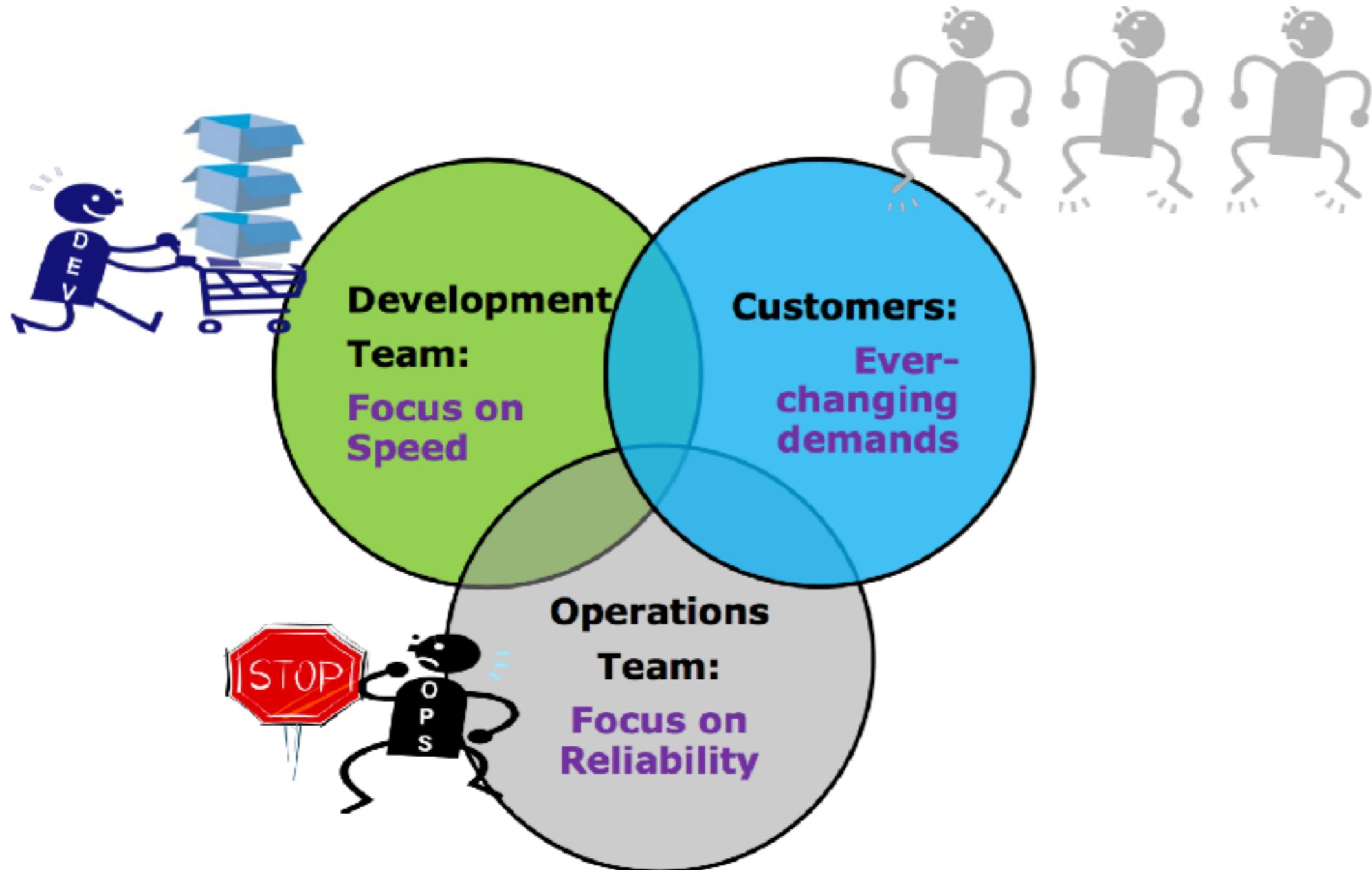
## Lead time ?



# Traditional development



# Conflict of Interest



# Conflict of Interest

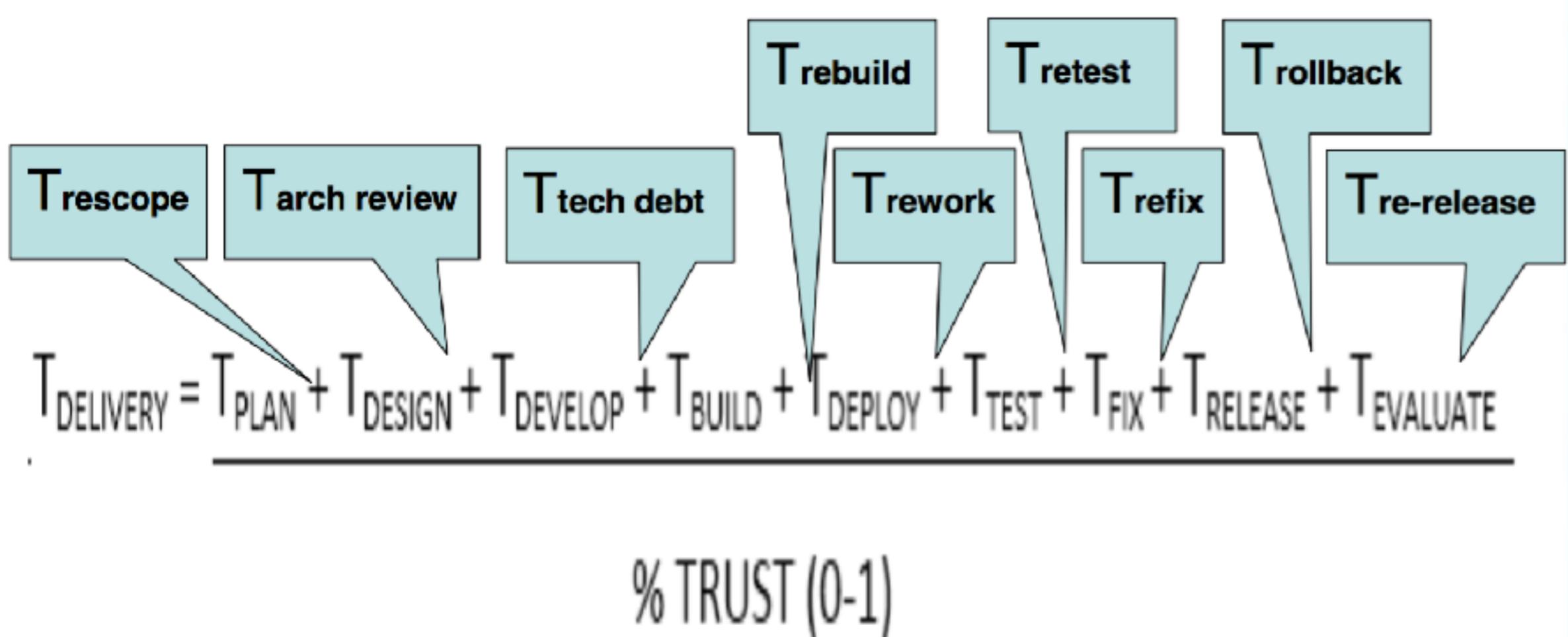


# Conflict of Interest

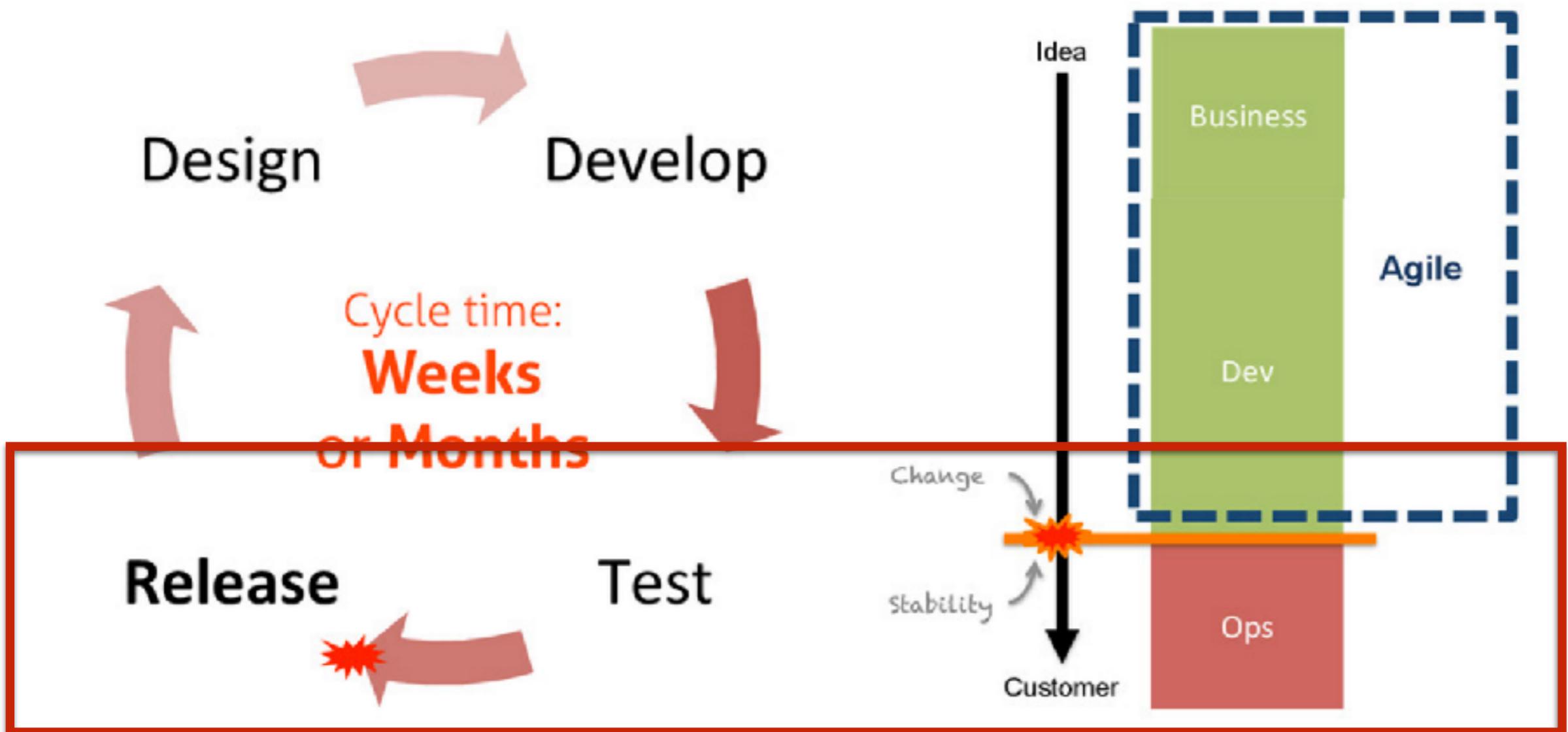




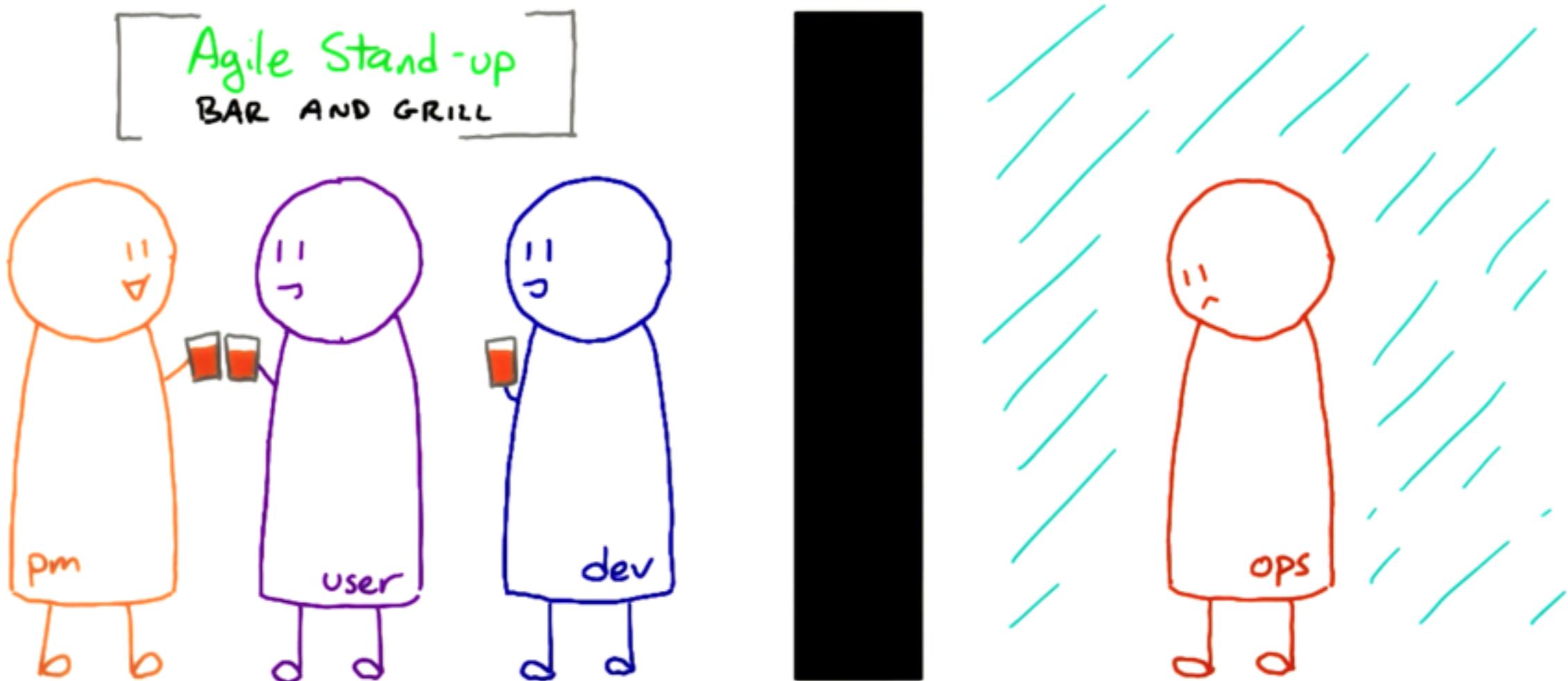
# Low trust create extra steps



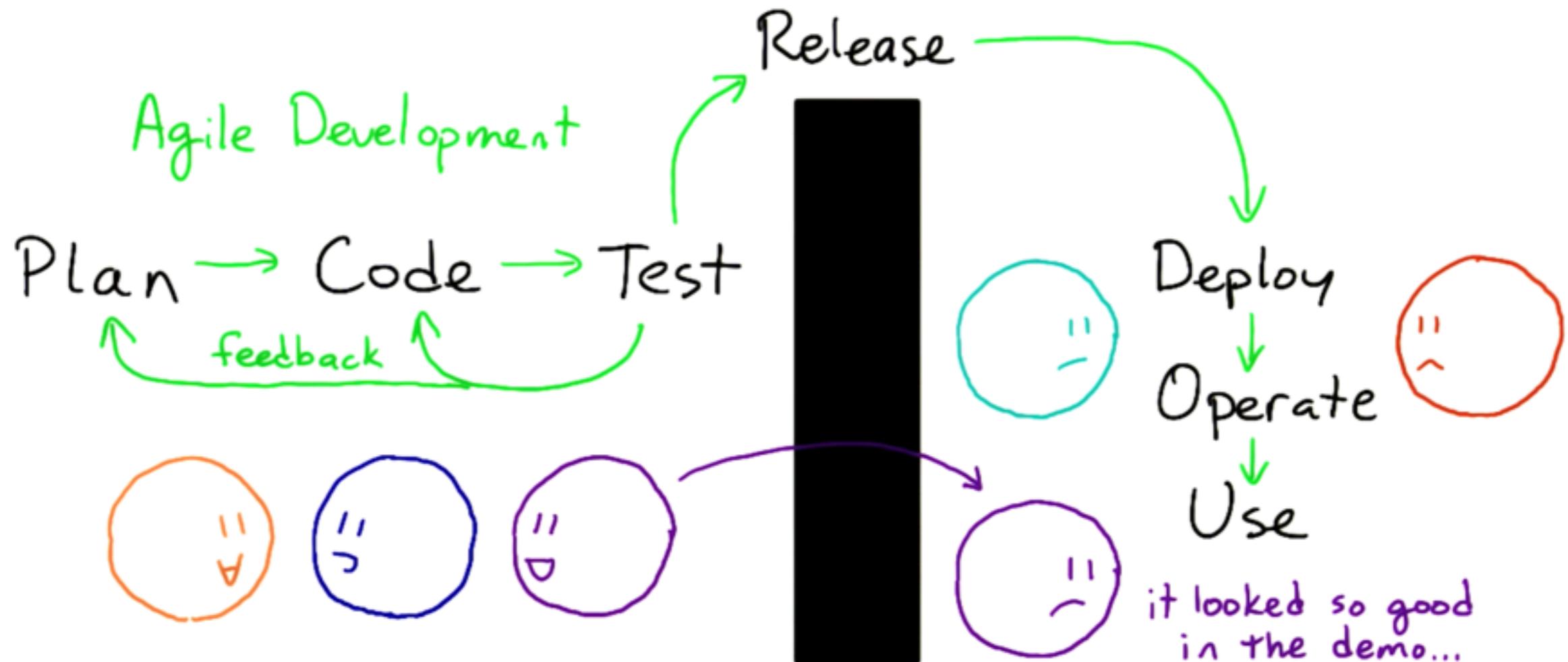
# Iterative/Agile development



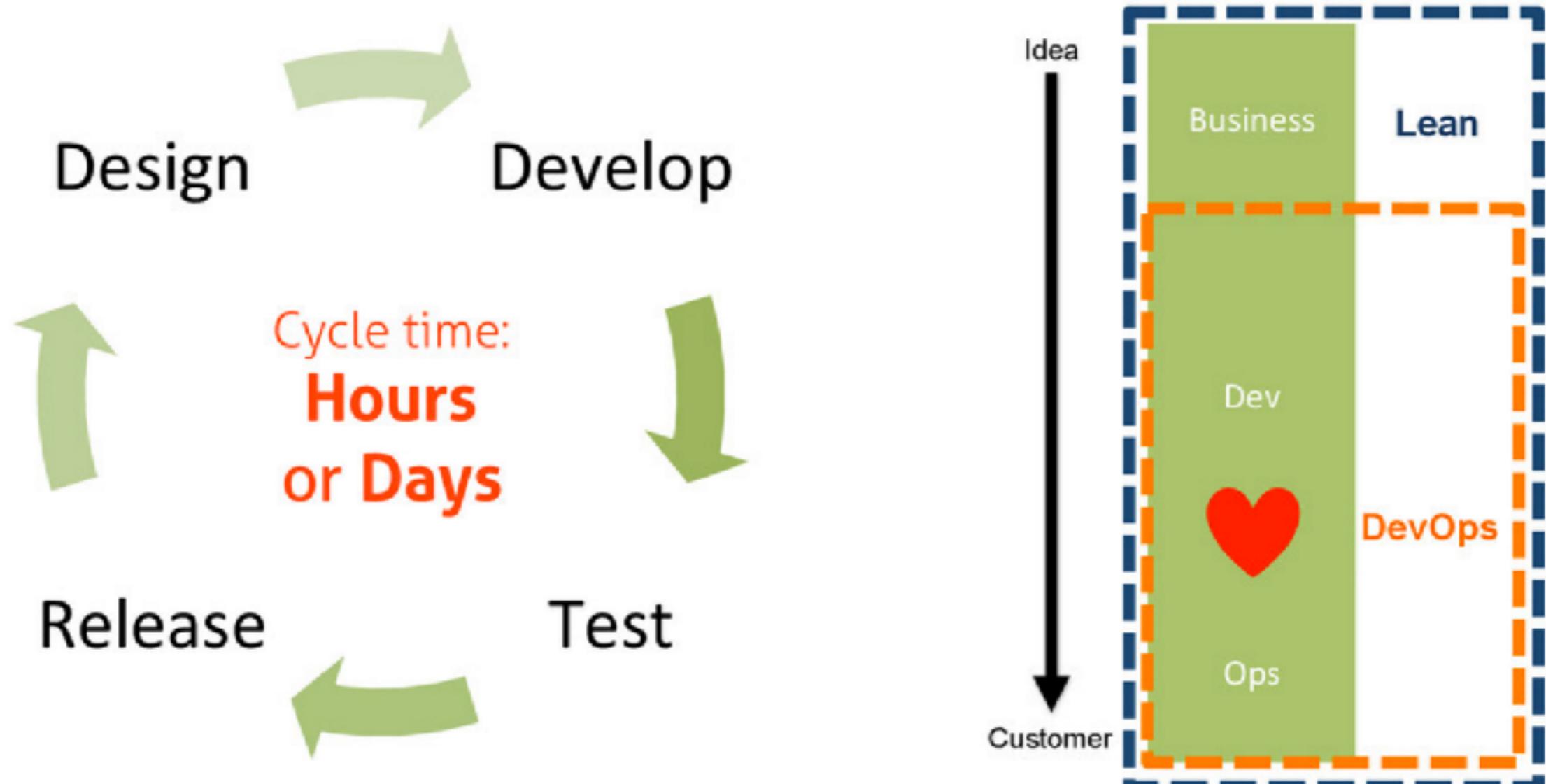
# Iterative/Agile development



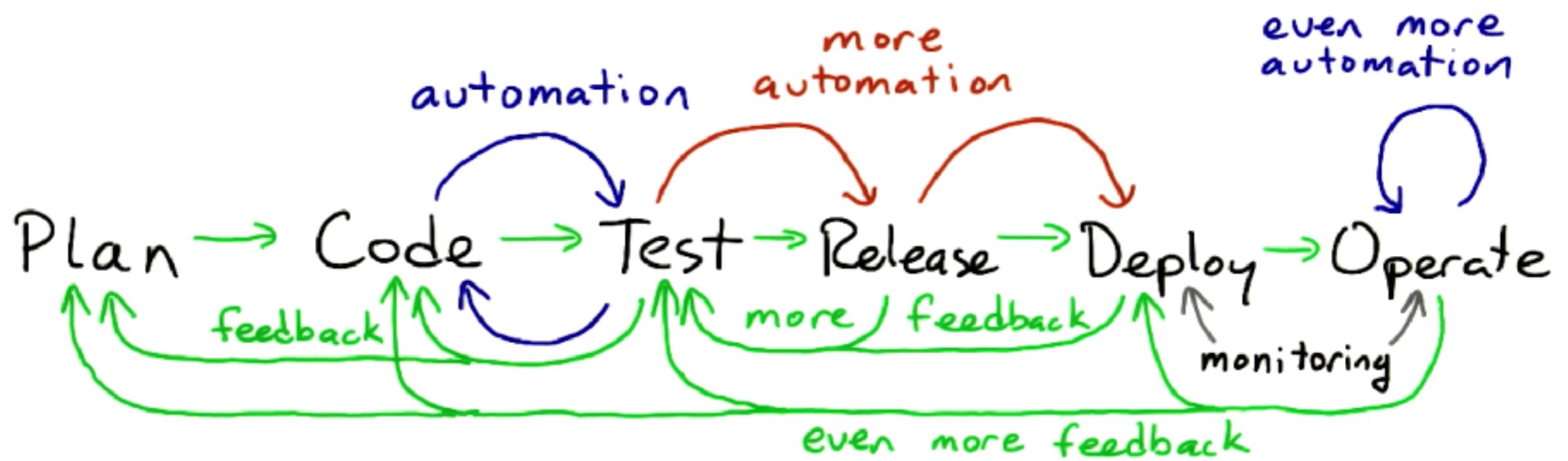
# Iterative/Agile development



# Rise of DevOps

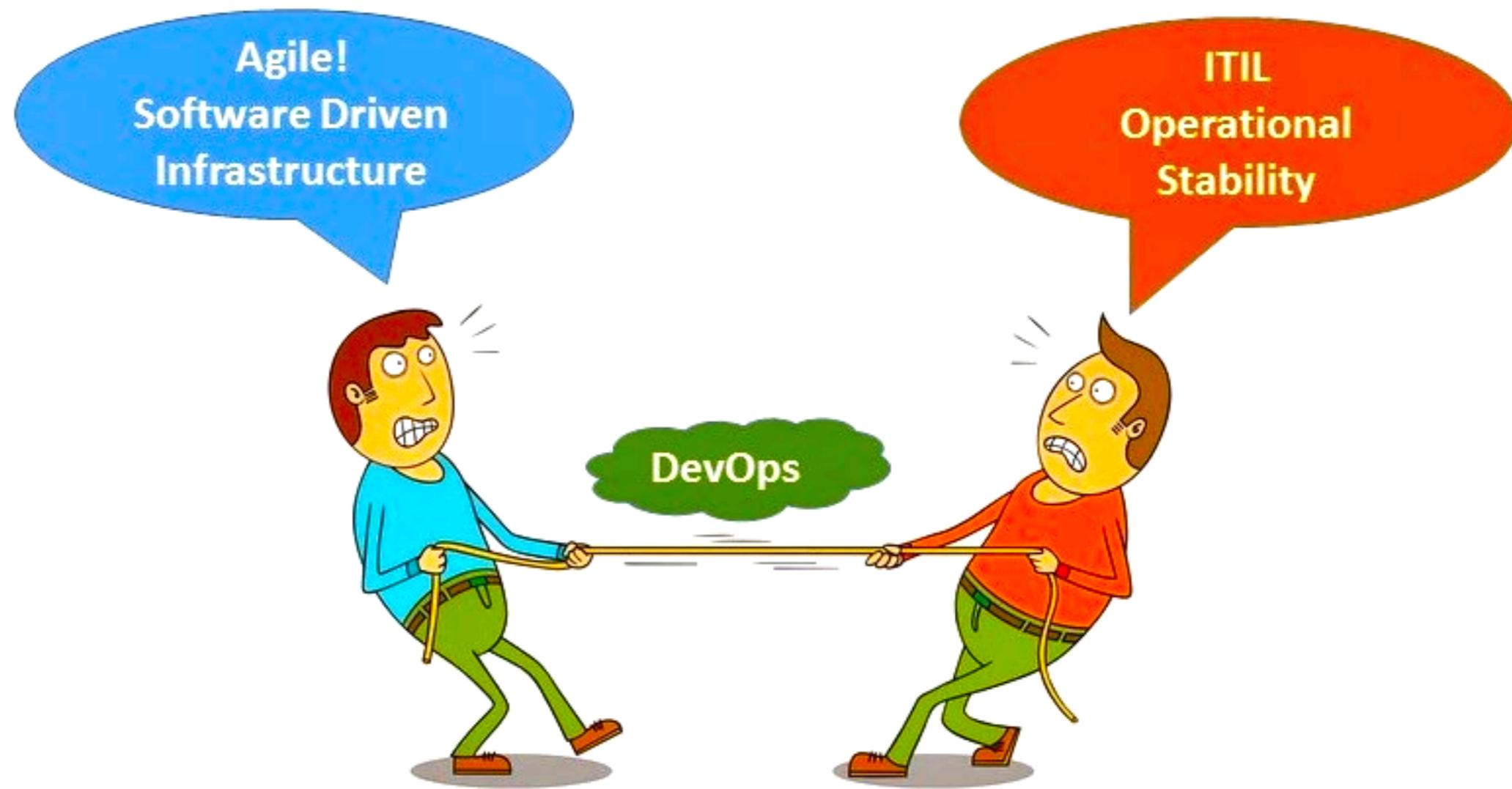


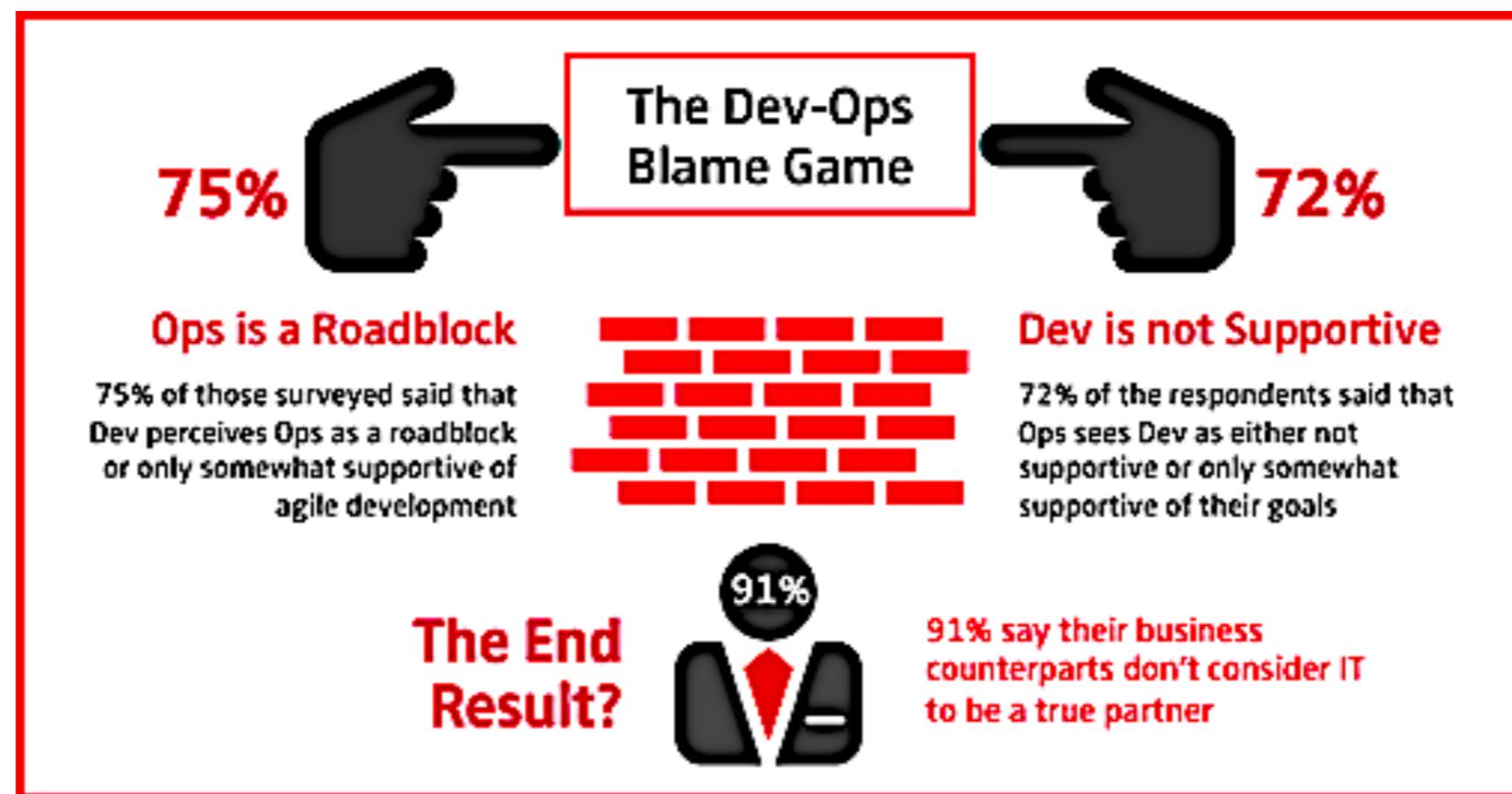
# Rise of DevOps

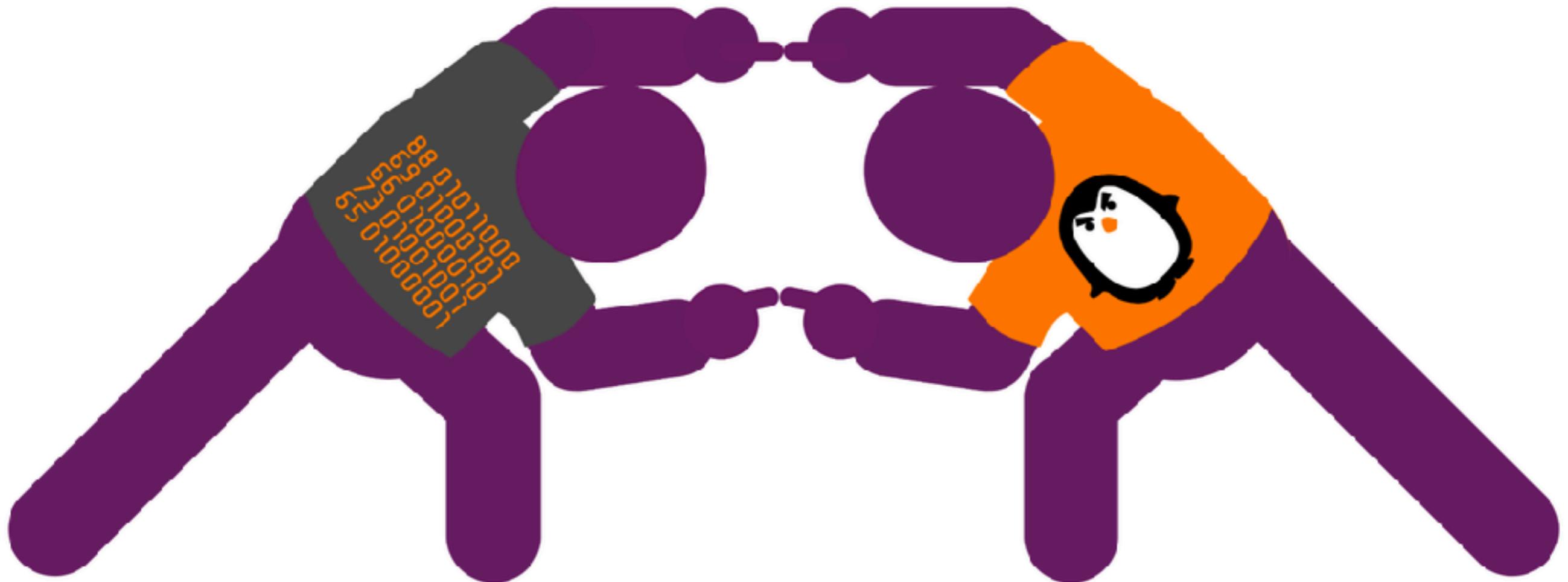


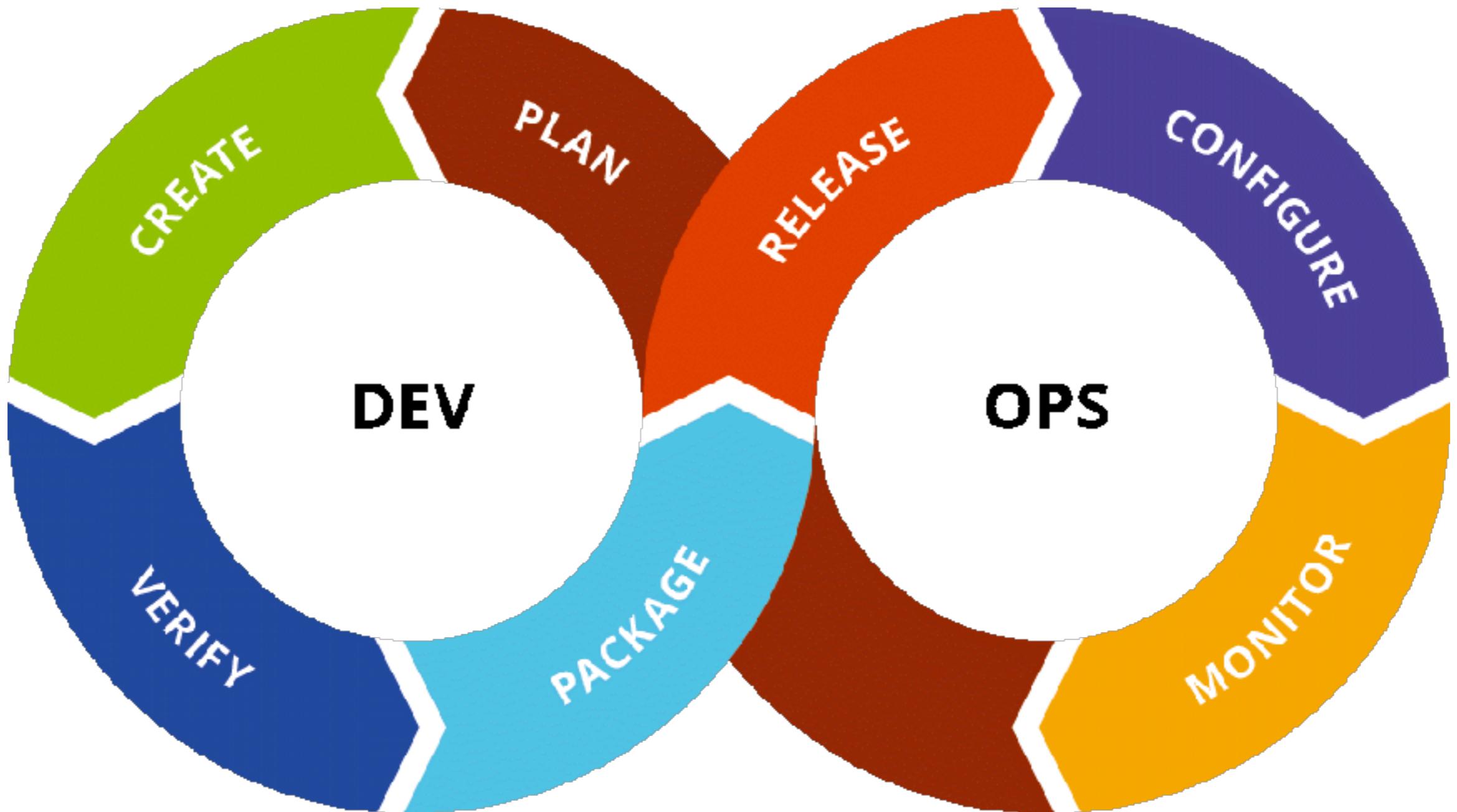
Agile Development → Continuous Integration + Delivery → DevOps?











**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



**OPS**

 **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 is**  
development  
and operations  
**collaboration"**

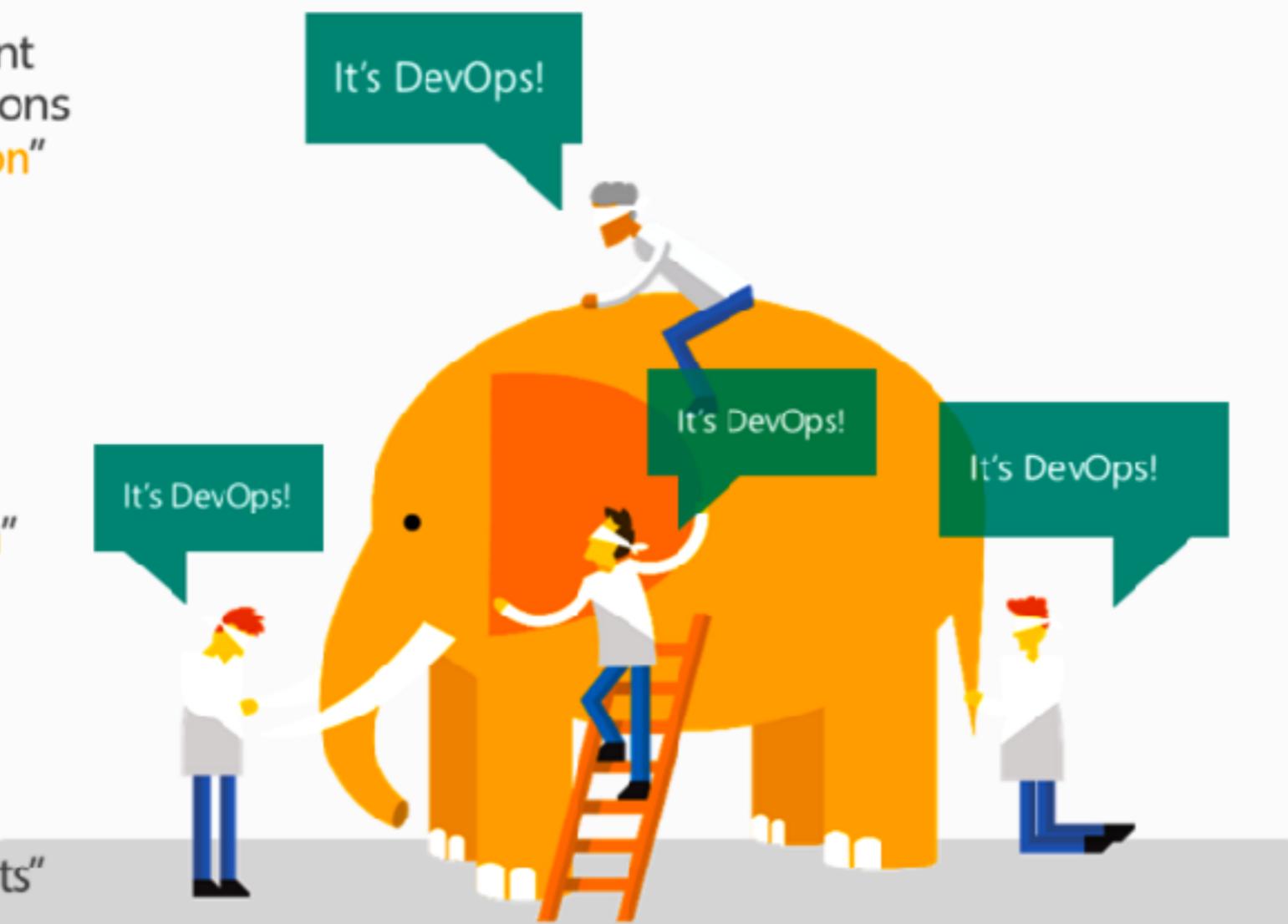
**"DevOps**  
is using  
**automation"**

**"DevOps**  
is **small**  
deployments"

**"DevOps is**  
treating your  
**infrastructure**  
**as code"**

**"DevOps**  
is feature  
**switches"**

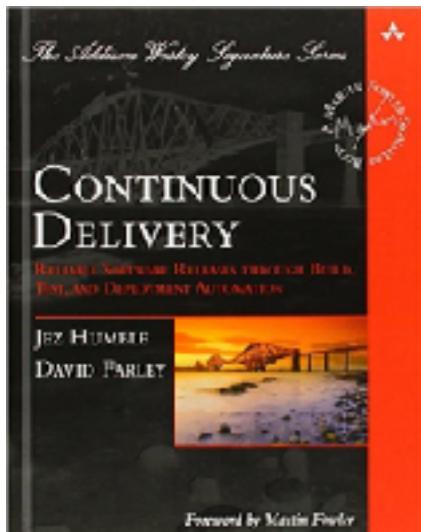
**"Kanban**  
for Ops?"



# DevOps ?

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

*- Jez Humble -*



# DevOps ?

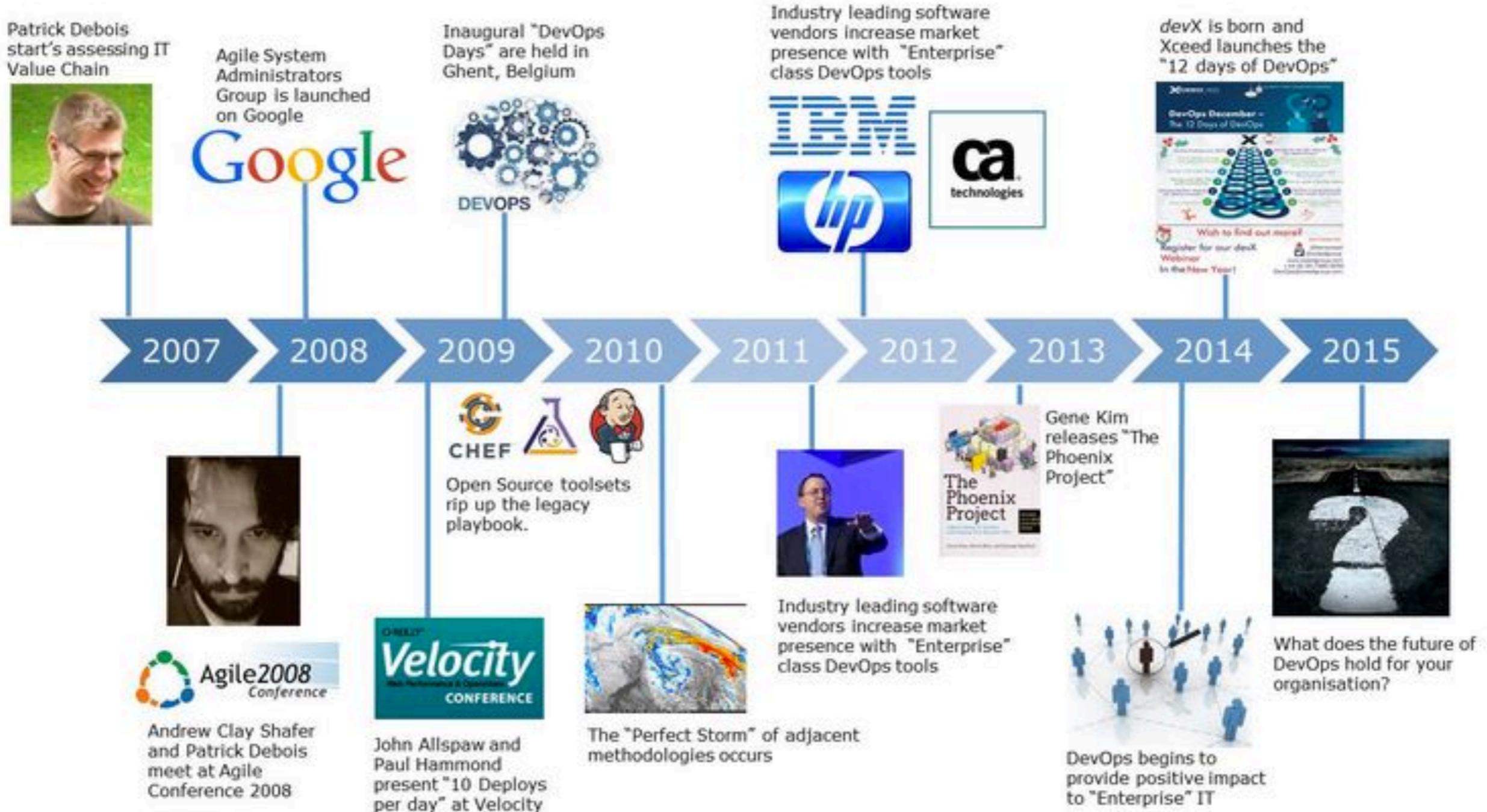
“A mix of patterns intended to **improve collaboration** between development and operations. DevOps addresses **shared goals and incentives** as well as **shared processes and tools.**”



*- Michael Huttermann -*



# DevOps Timeline

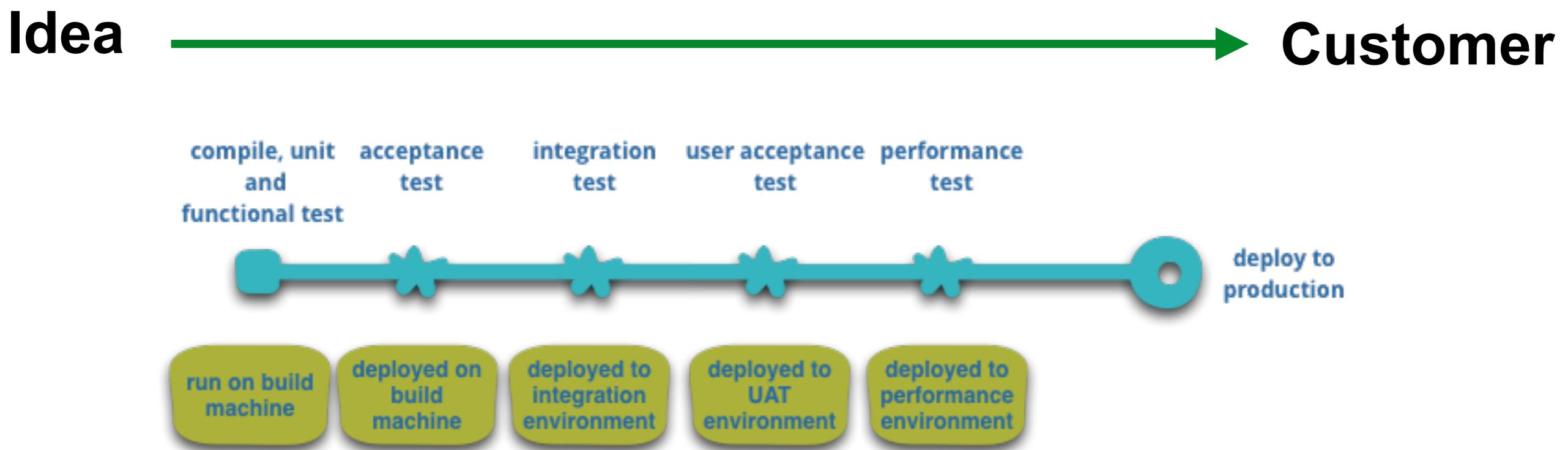


<https://www.pinterest.com/pin/263390278185605258/?lp=true>



# Goal of DevOps

“Improve the delivery of value for Customer and Business”



# DevOps

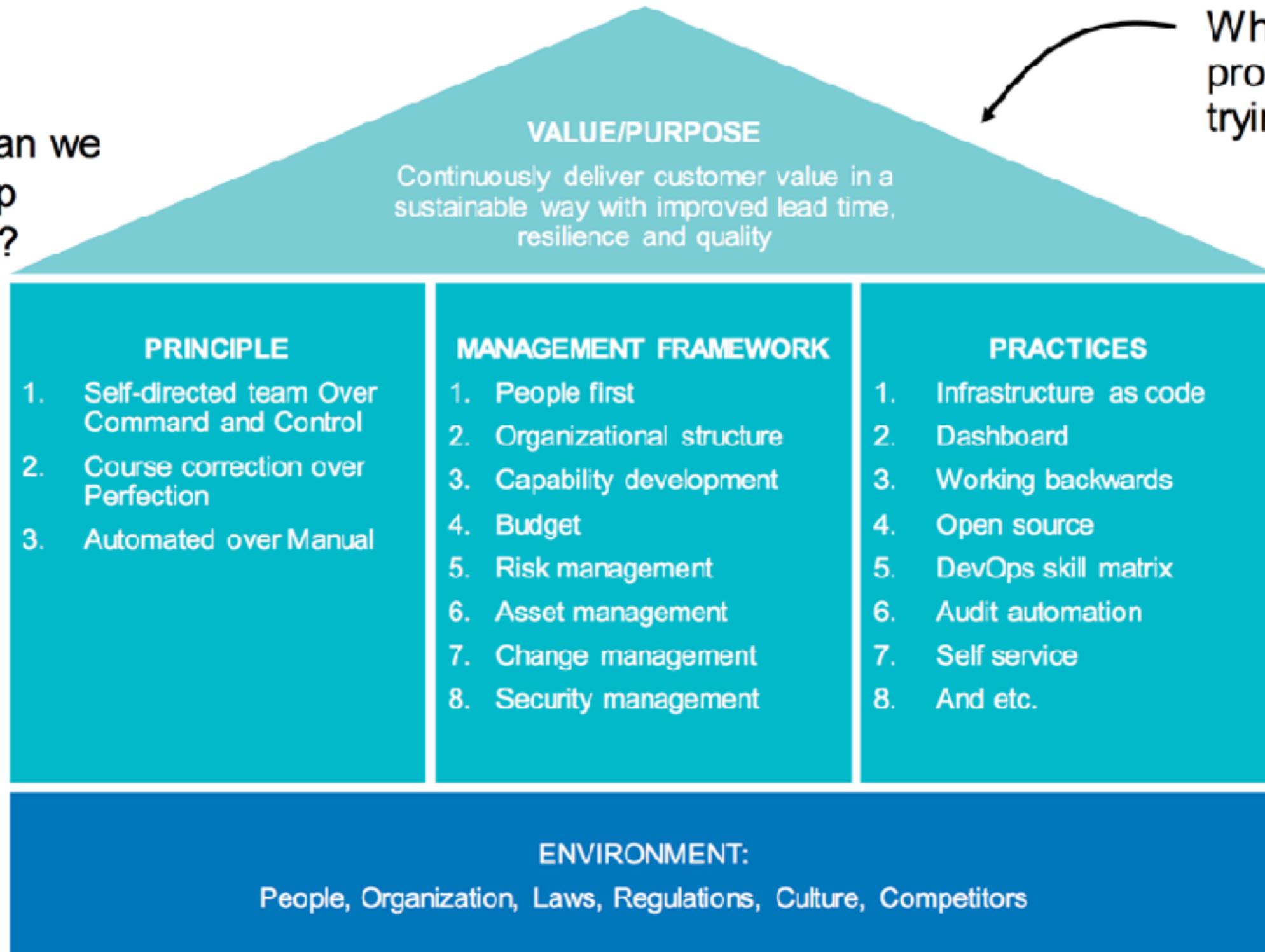


to enable the continuous delivery of value to customer



# DevOps adoption model

How can we develop people?

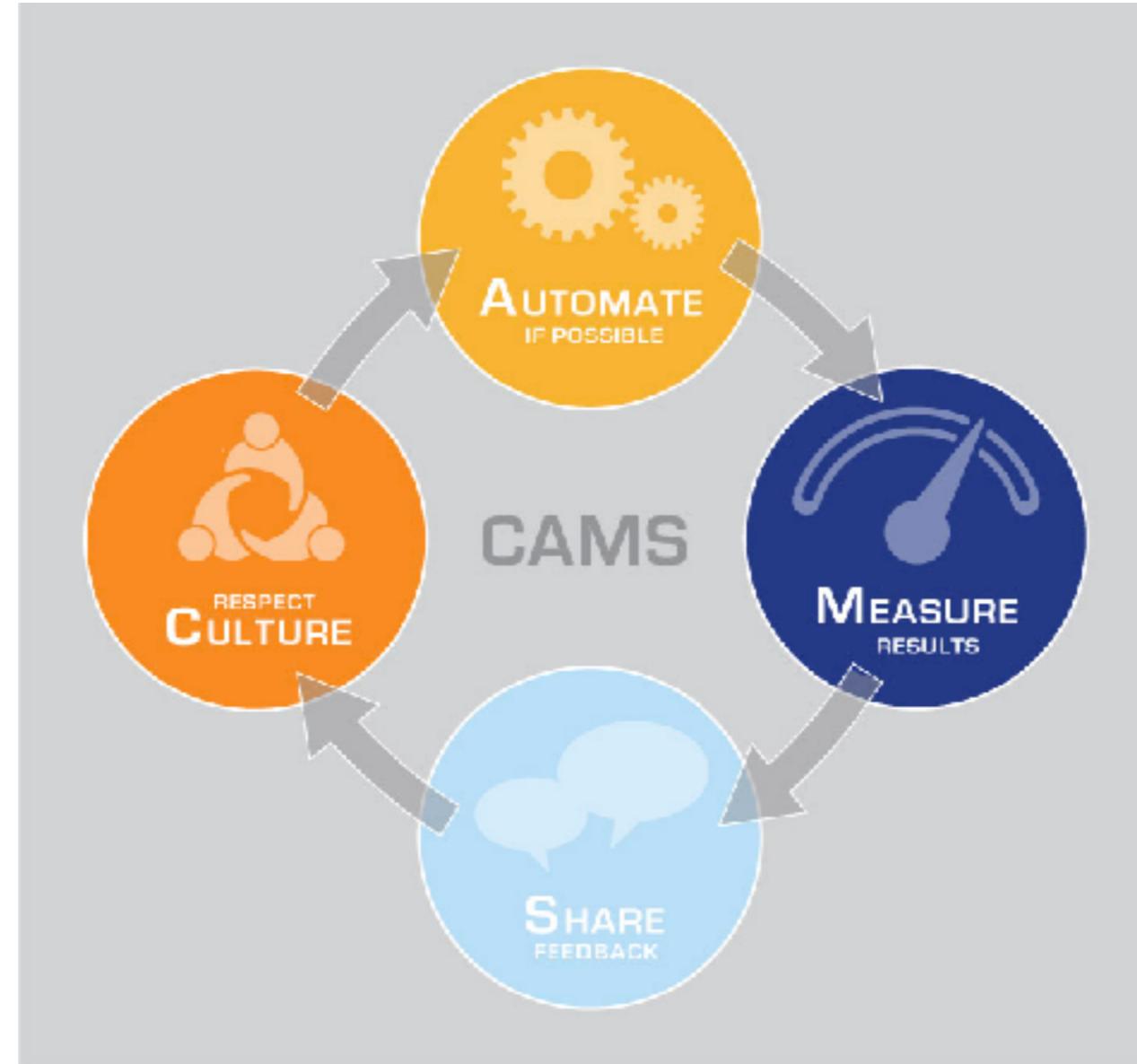


What kind of problem are we trying to solve?

How can we improve the work



# DevOps Principles



# DevOps Principles

**Culture => People, Process, Tools**

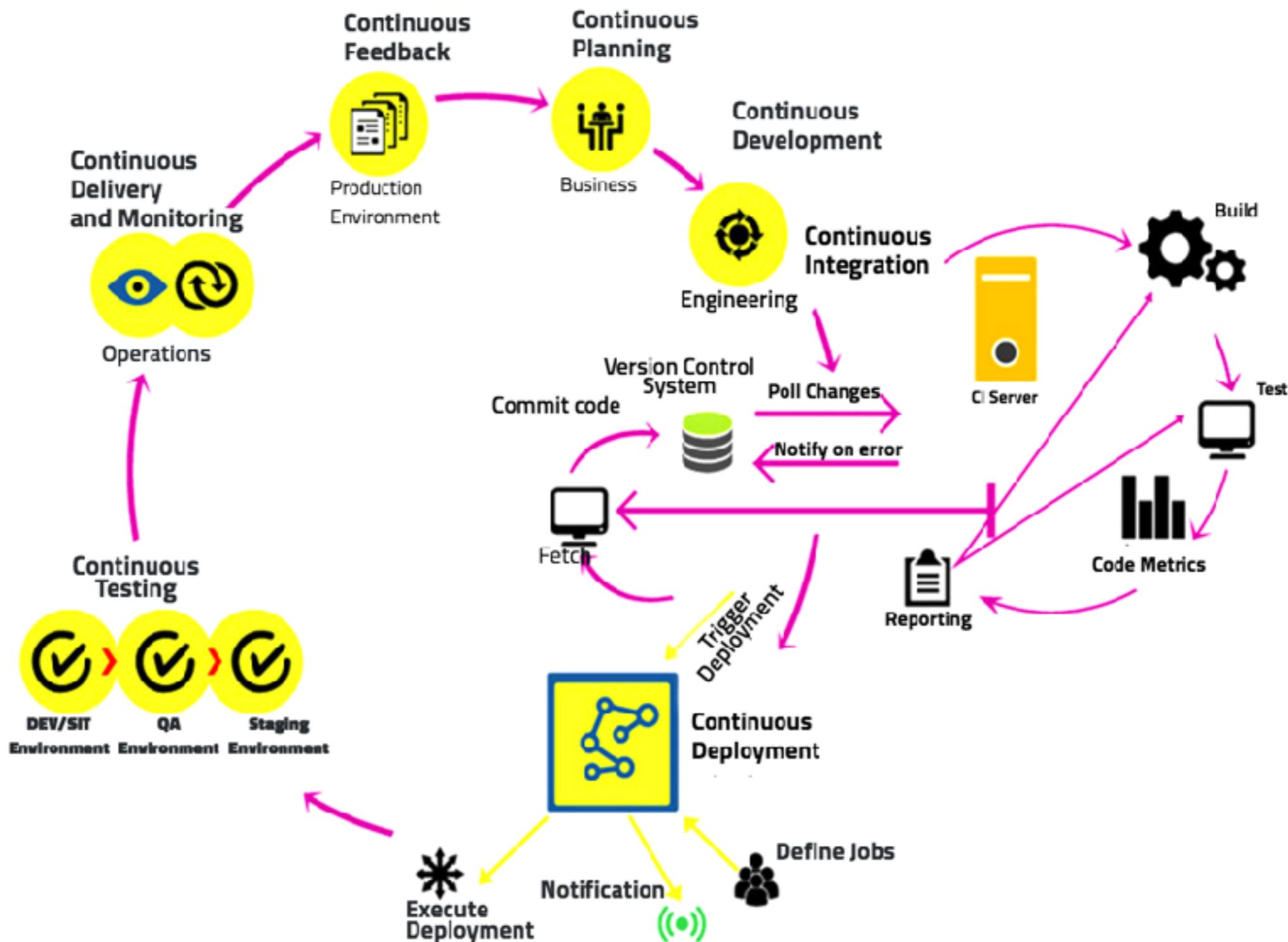
**Automation => Infrastructure as Code**

**Measurement => Measure everything**

**Sharing => Collaboration/Feedback**

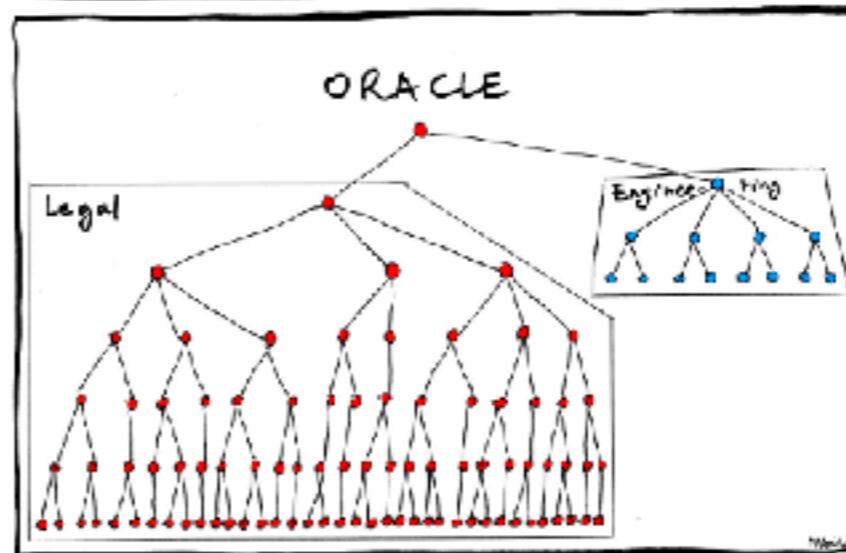
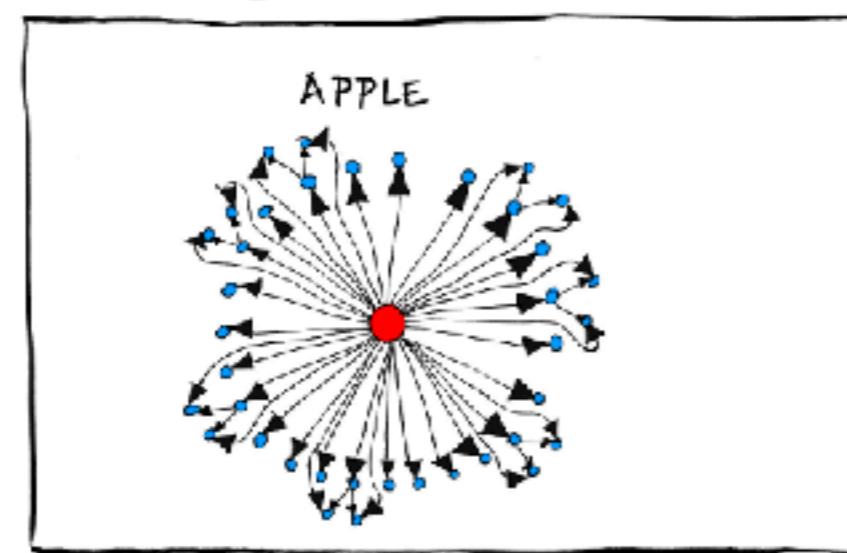
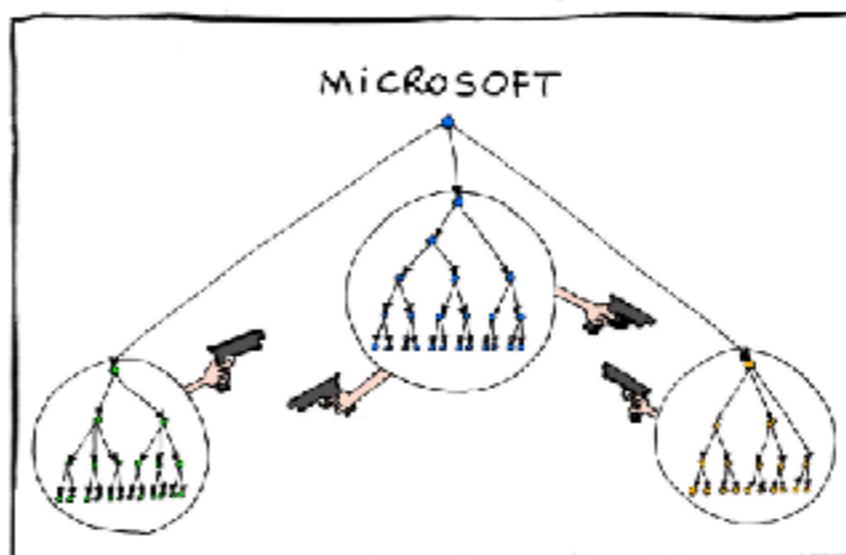
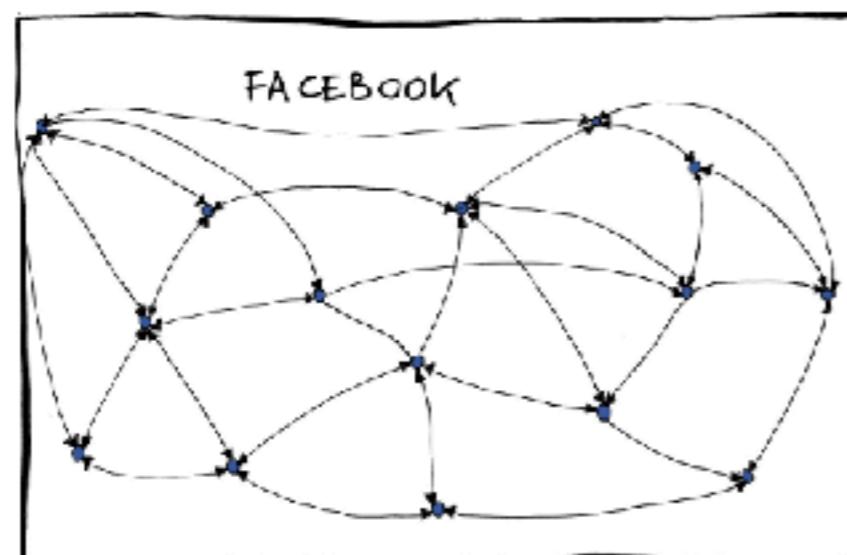
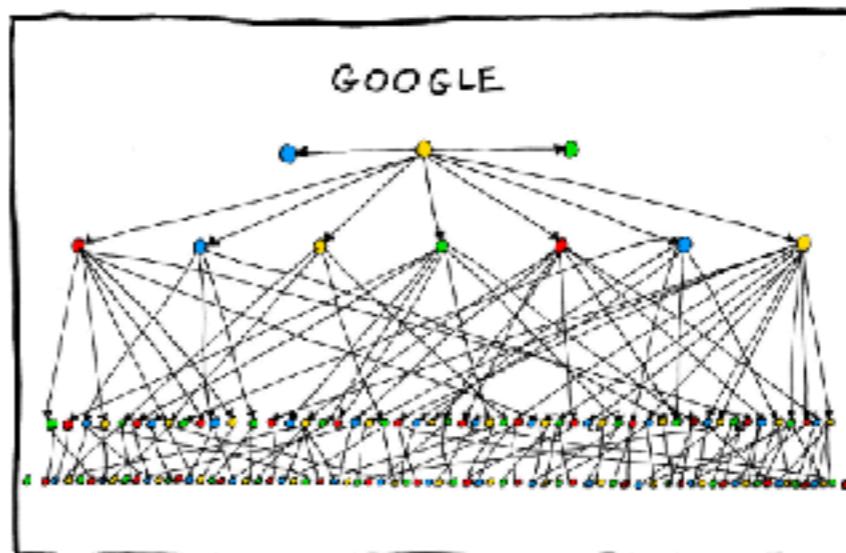
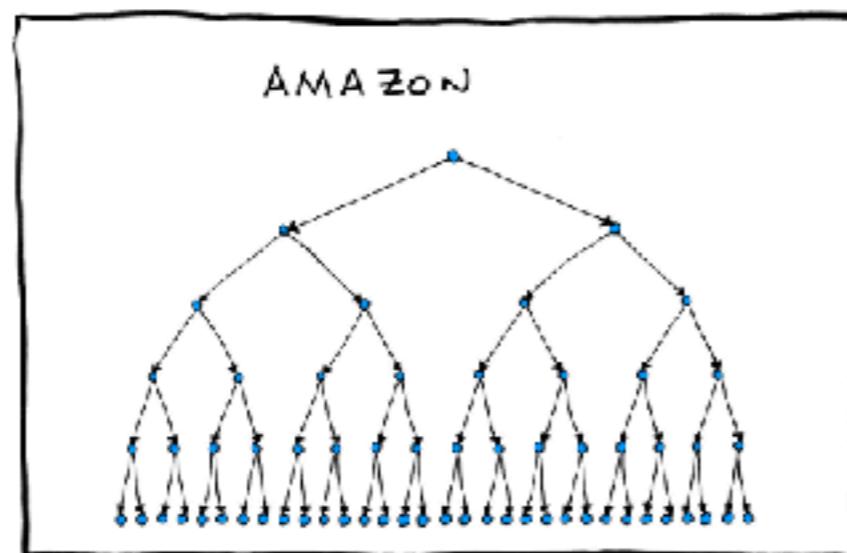


# DevOps Practices

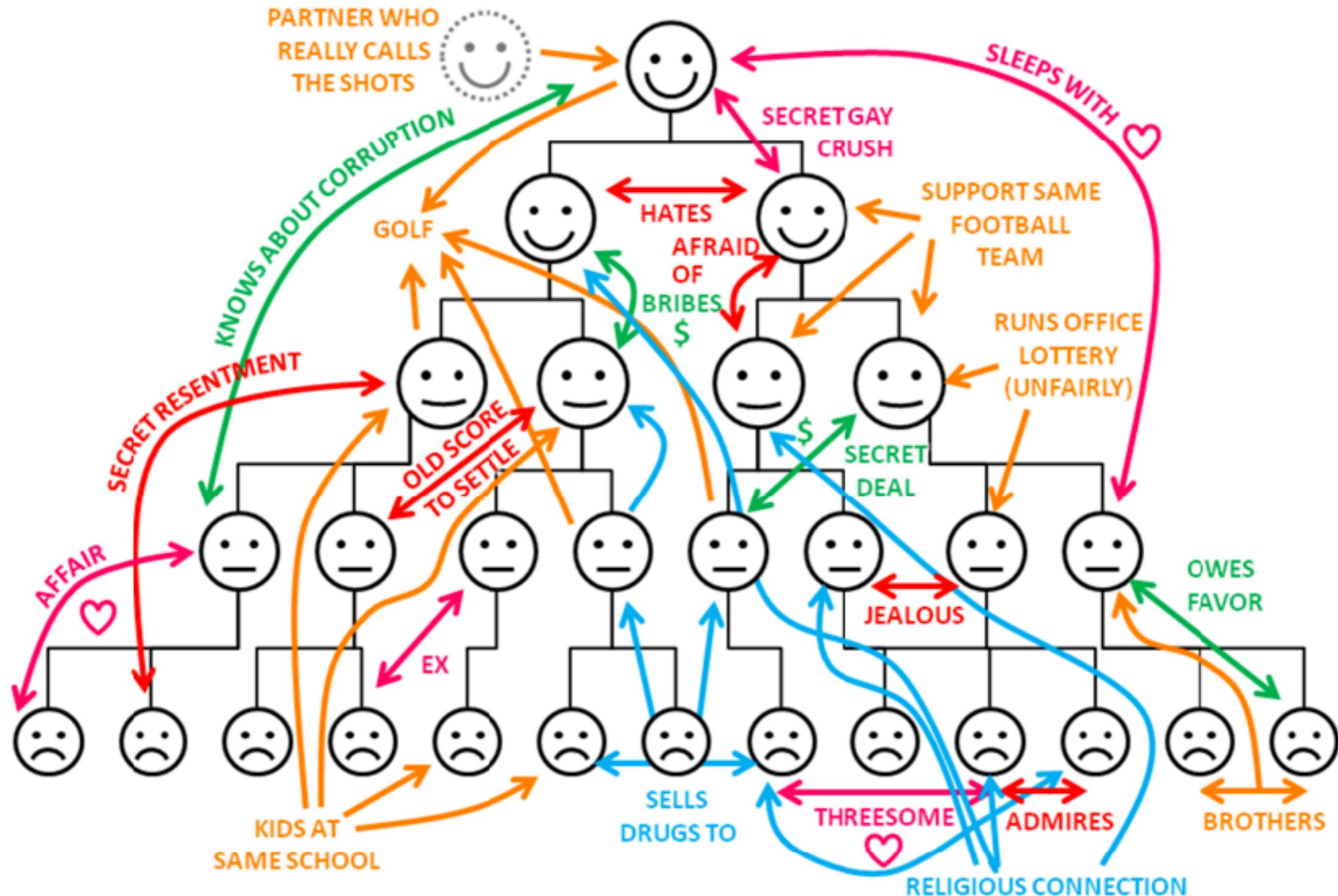


# All about Organization structure and culture

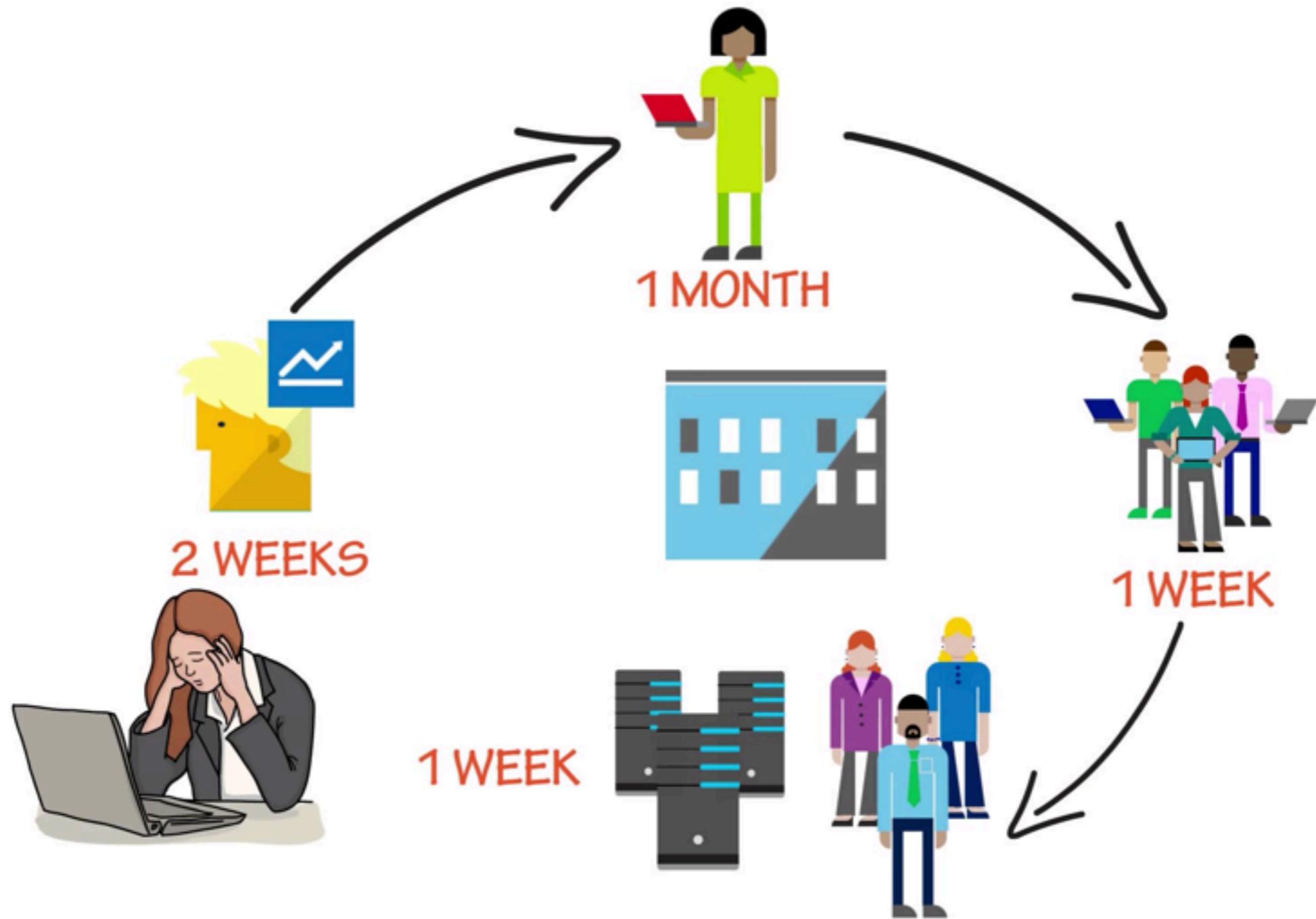




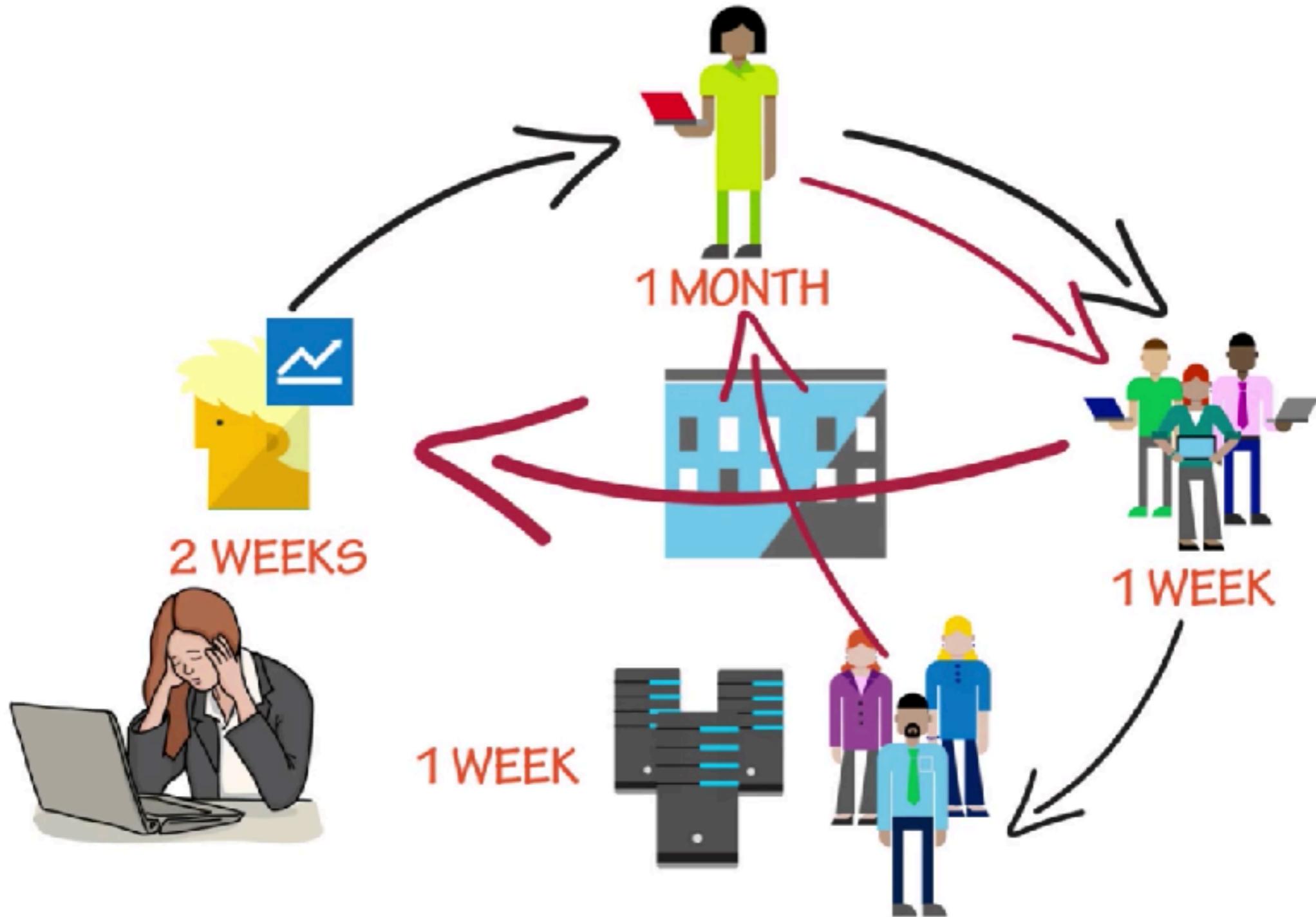
# Problem ?



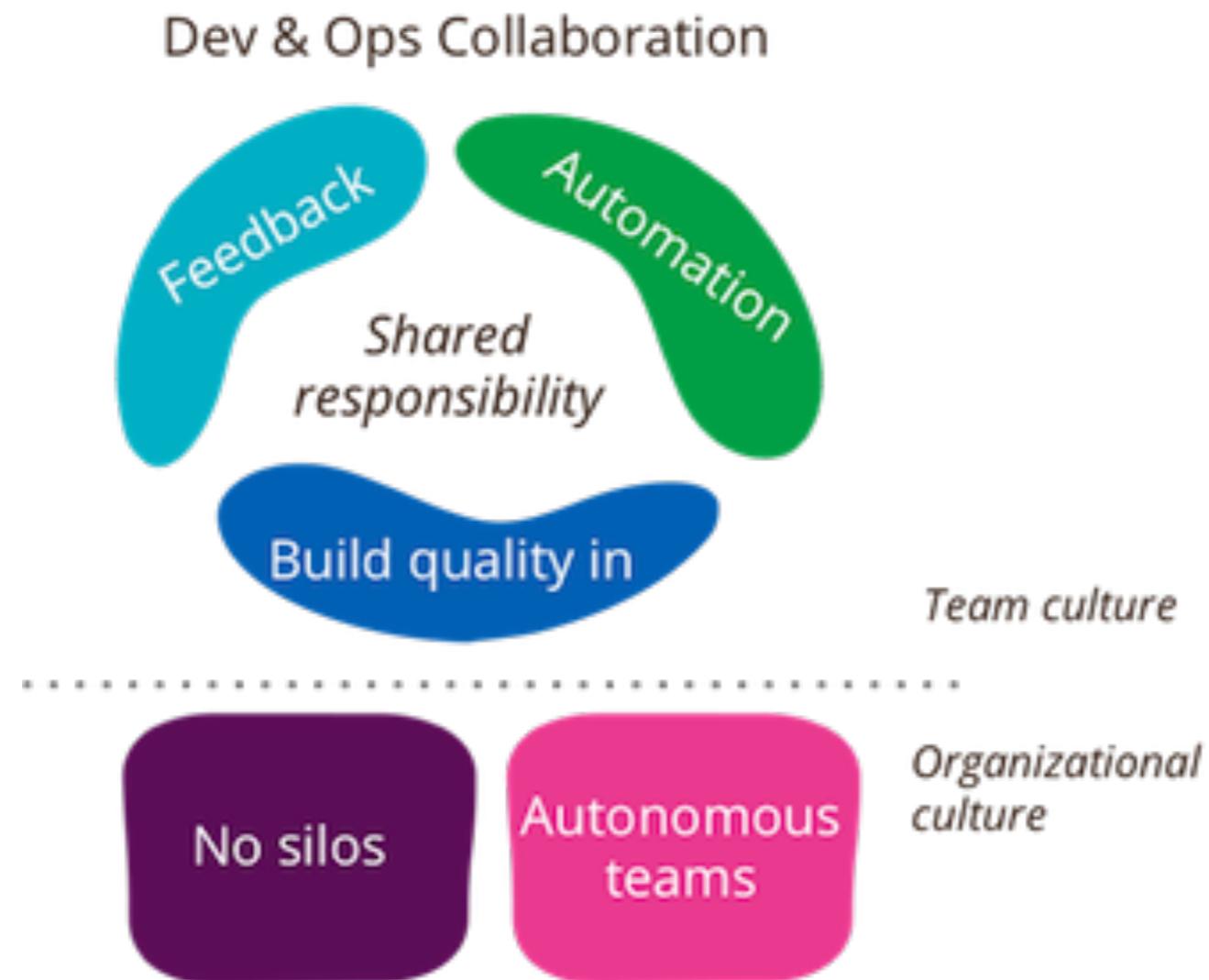
# Problem ?



# Problem ?



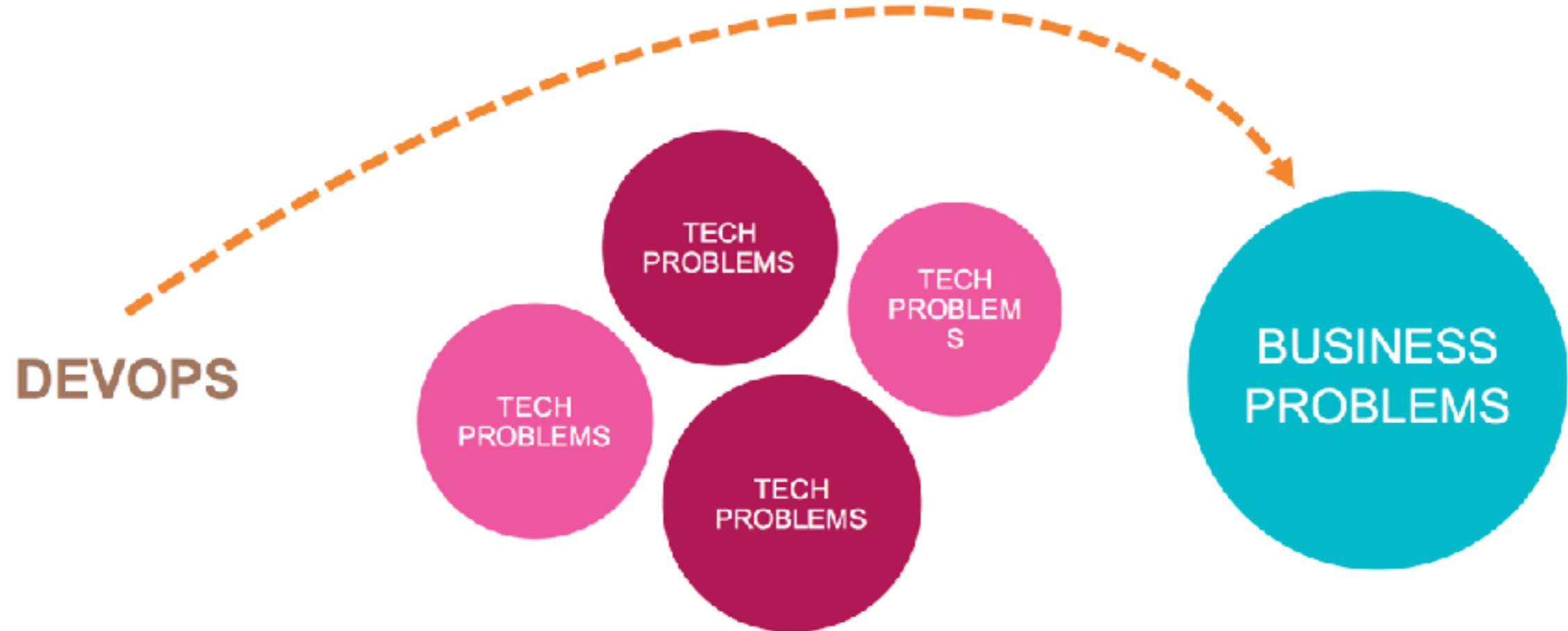
# Team and Organization culture



<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**

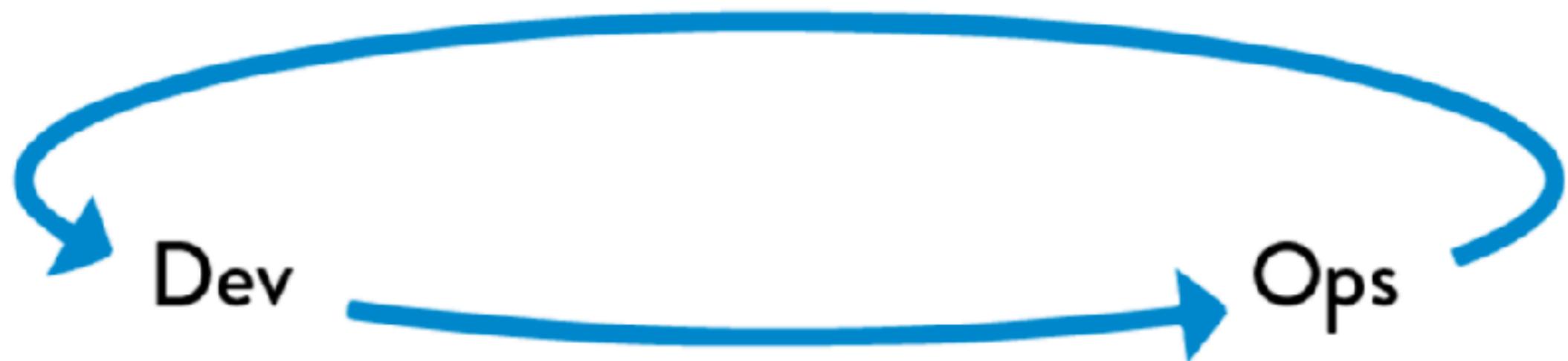
# **3 ways principle**



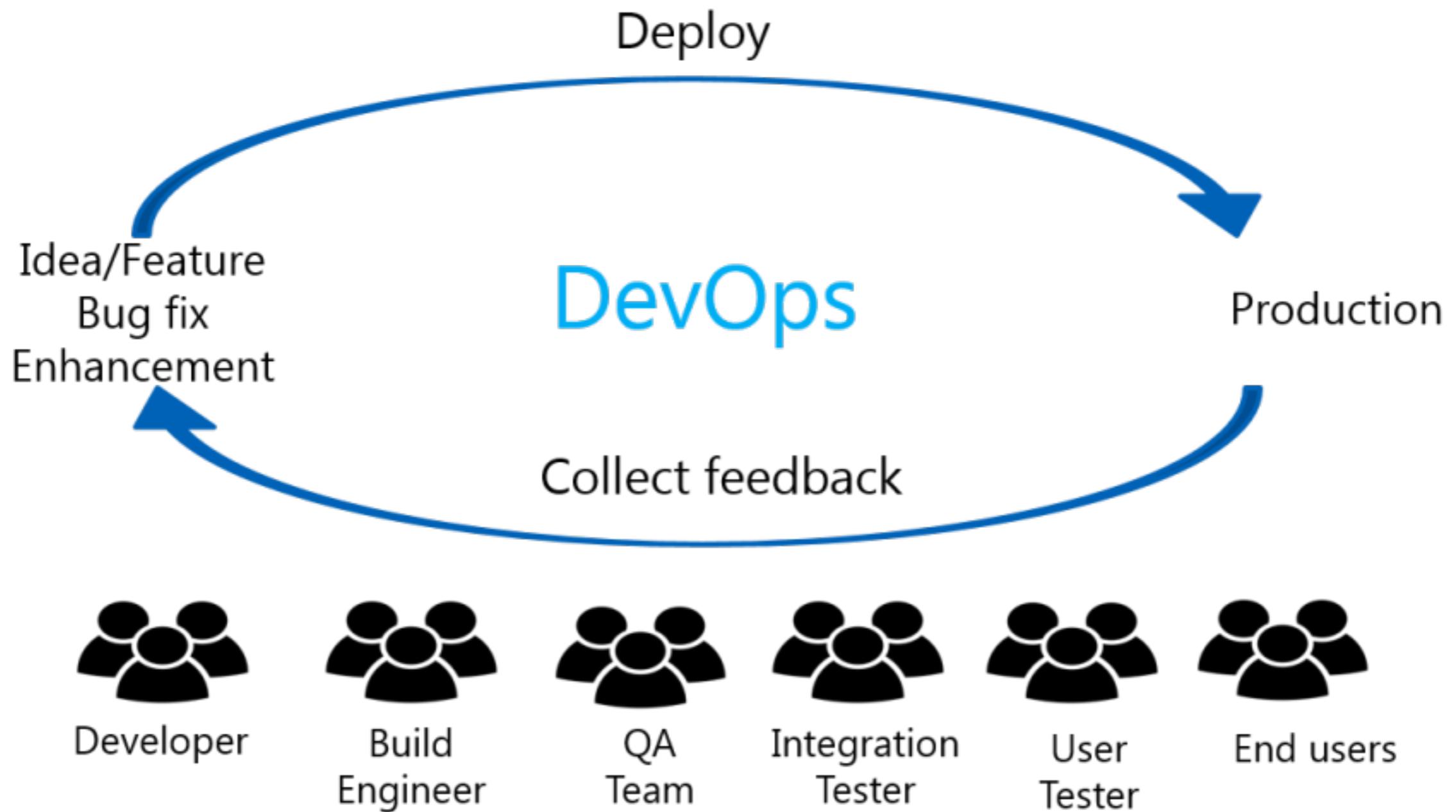
# 1. Flow principle



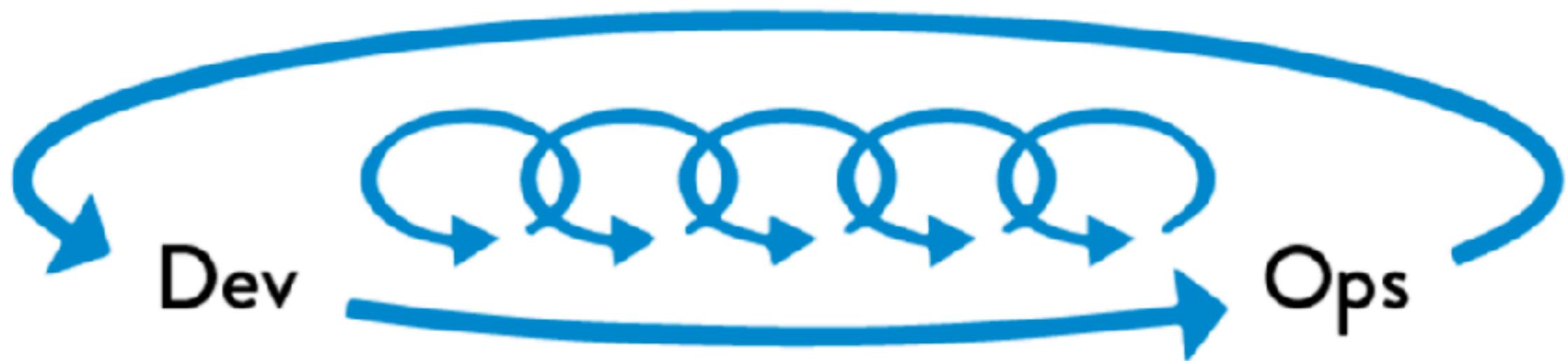
## 2. Feedback principle (1)



## 2. Feedback principle (2)



# 3. Continuous learning principle



# DevOps Topologies

<https://web.devopstopologies.com/>

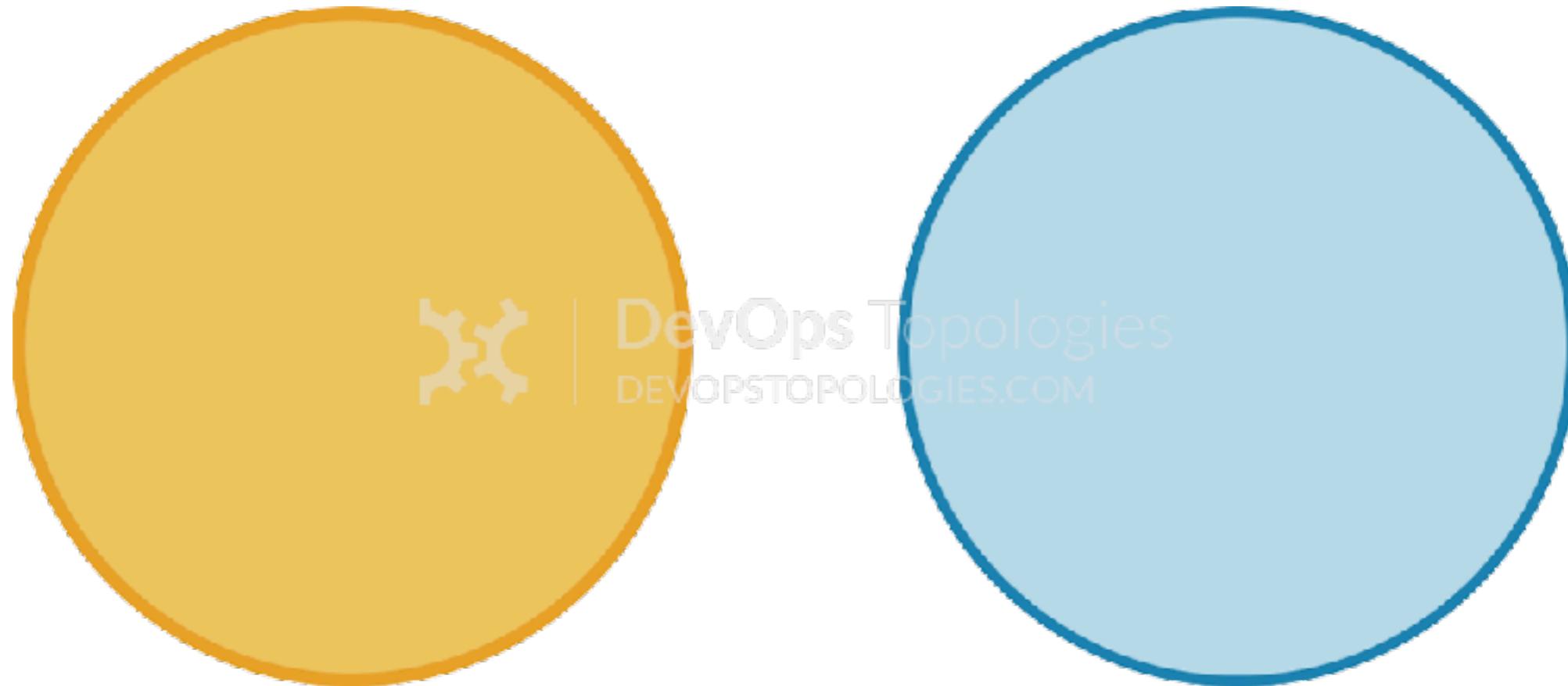


# DevOps Anti-Types

<https://web.devopstopologies.com/>



# Anti-type A : Dev and Ops silos

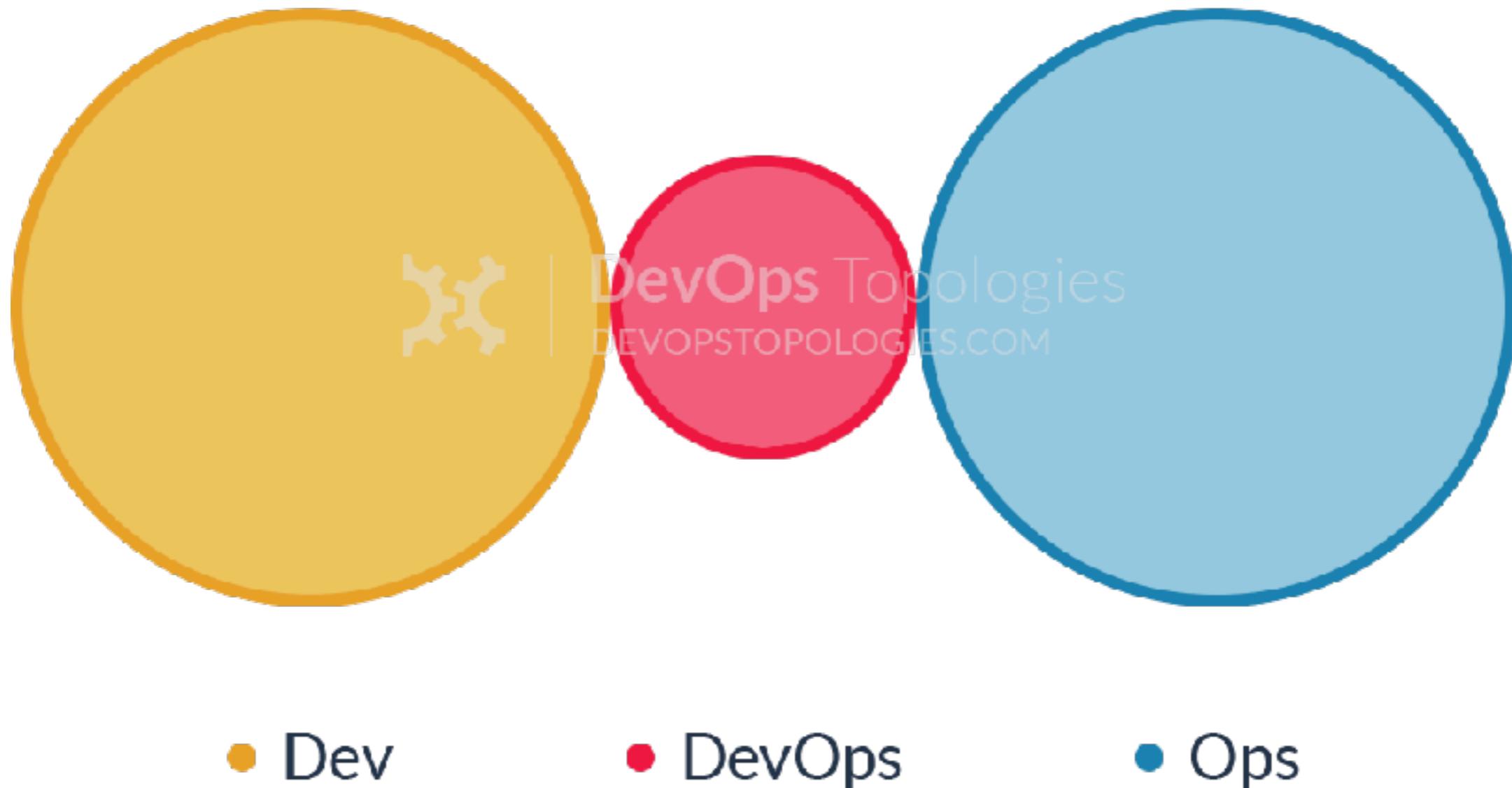


- Dev

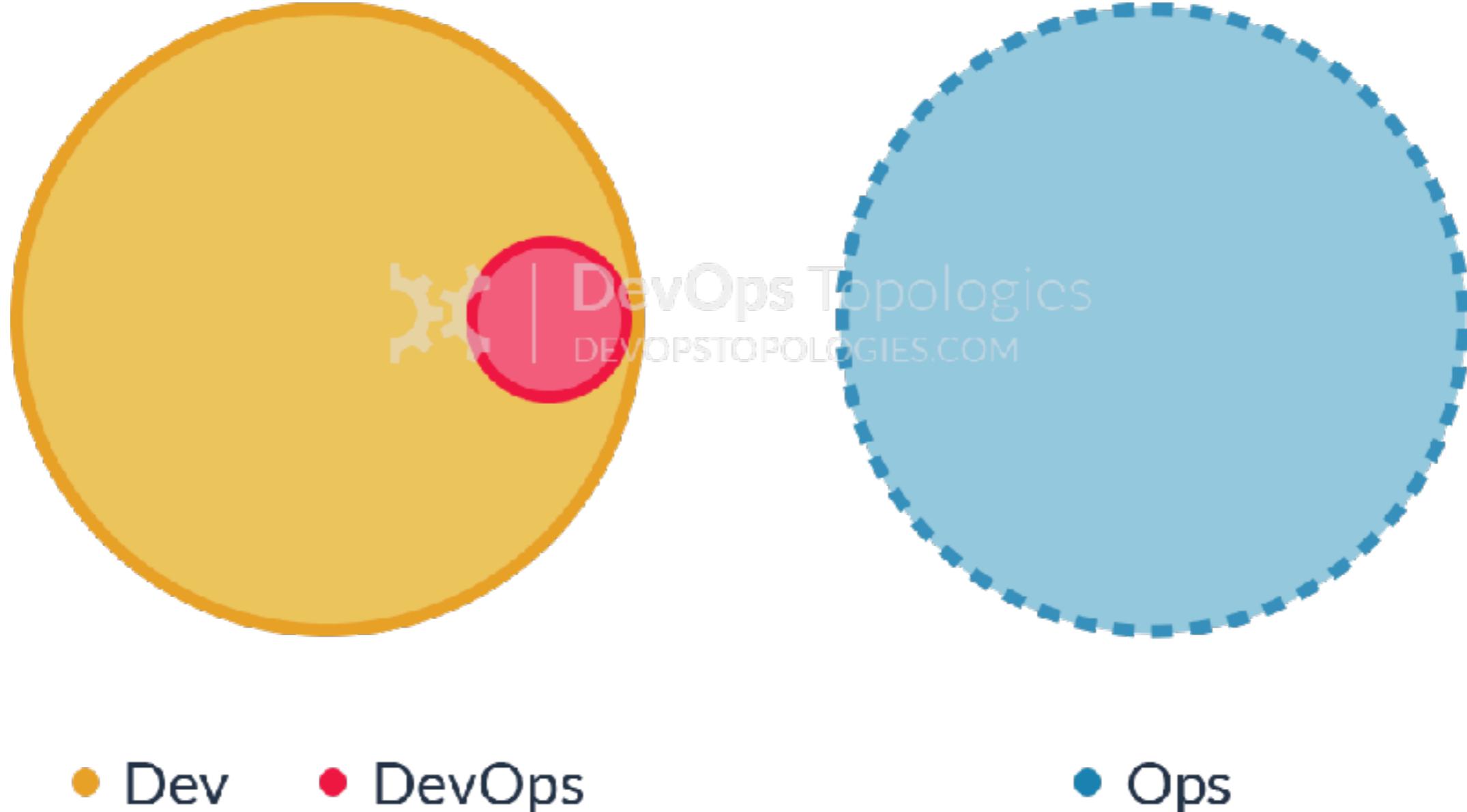
- Ops



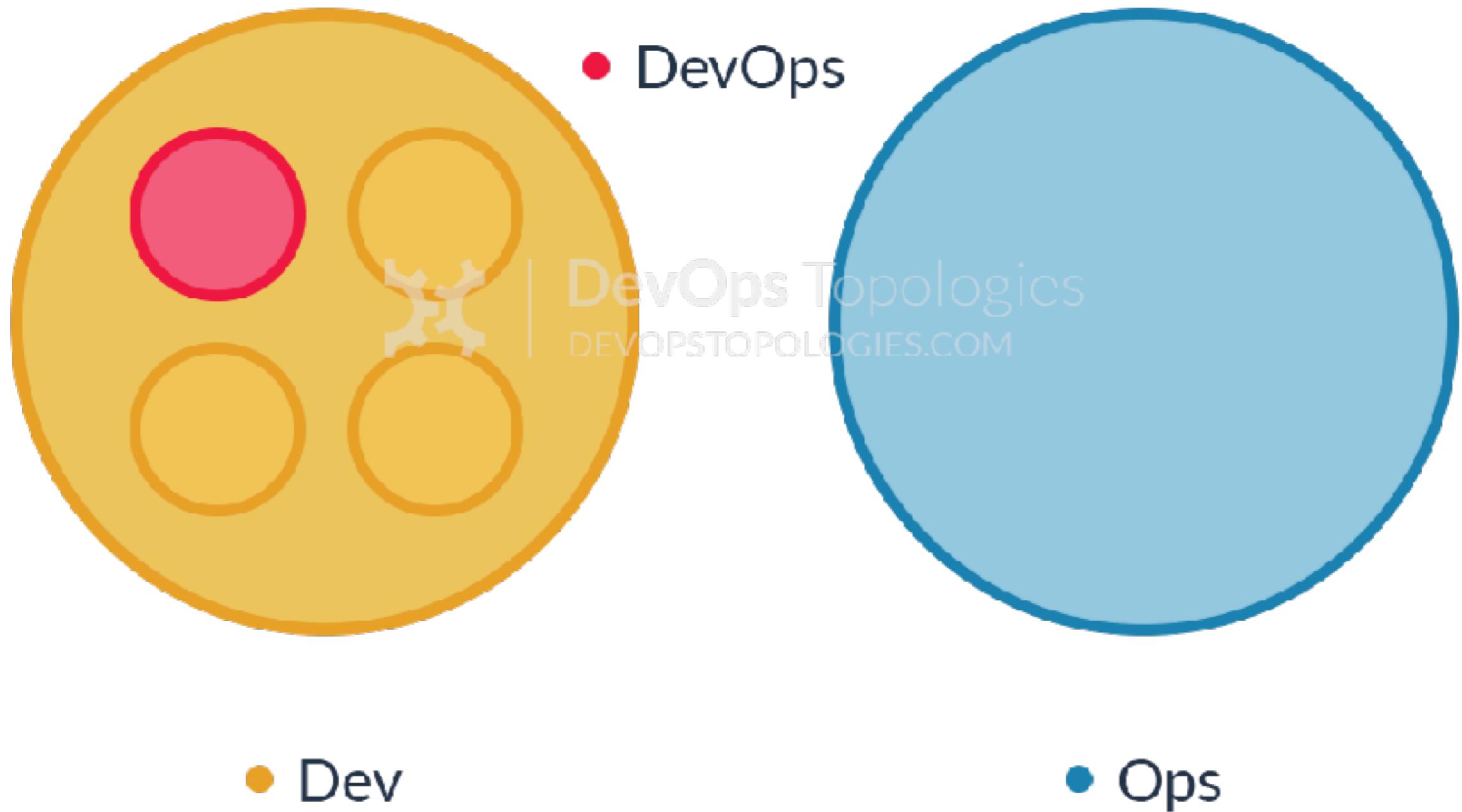
# Anti-type B : DevOps team silos



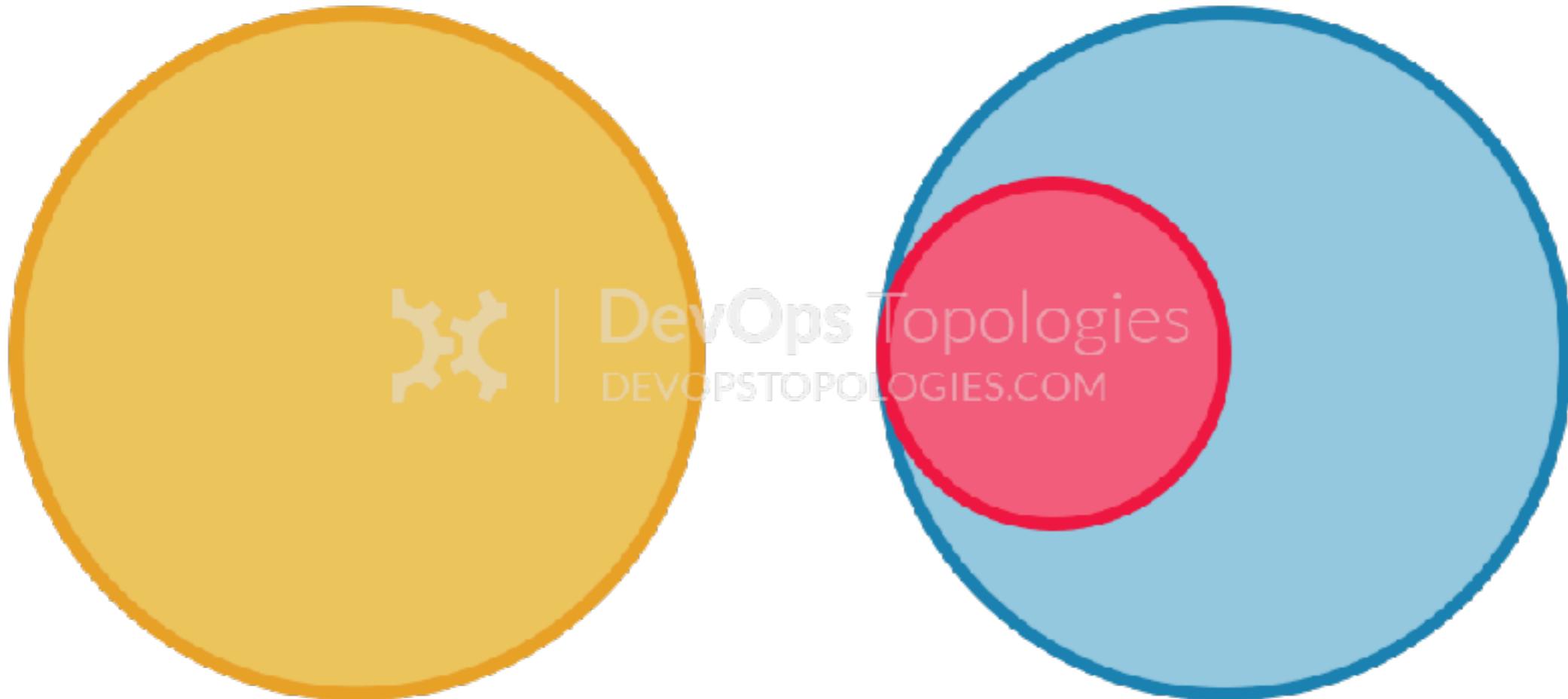
# Anti-type C : Dev don't need Ops



# Anti-type D : DevOps as a tool team



# Anti-type E : Rebranded SysAdmin



● Dev

● DevOps ● Ops

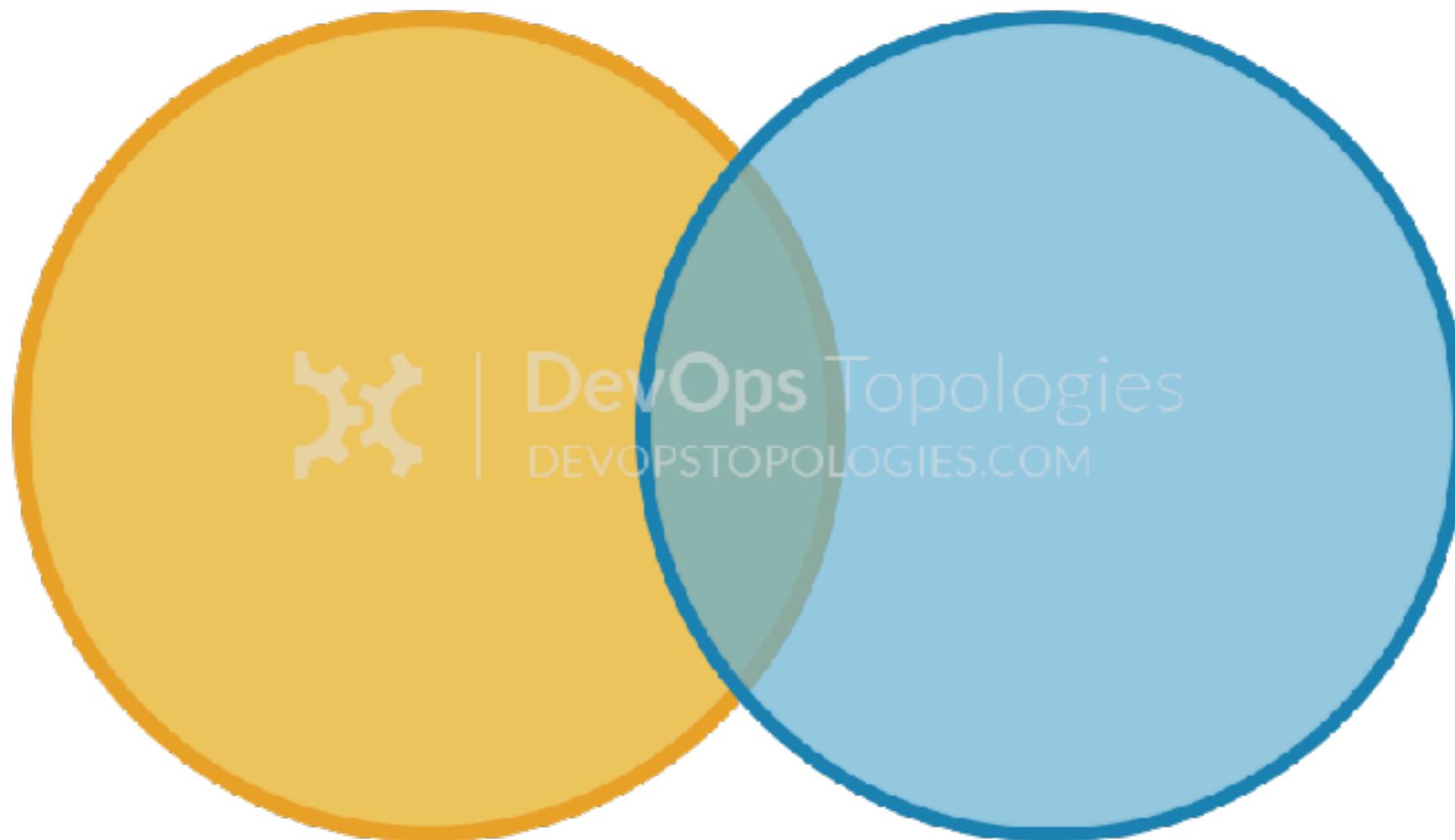


# DevOps Team Topologies

<https://web.devopstopologies.com/>



# Type 1 : Dev and Ops collaboration



● Dev

● Ops



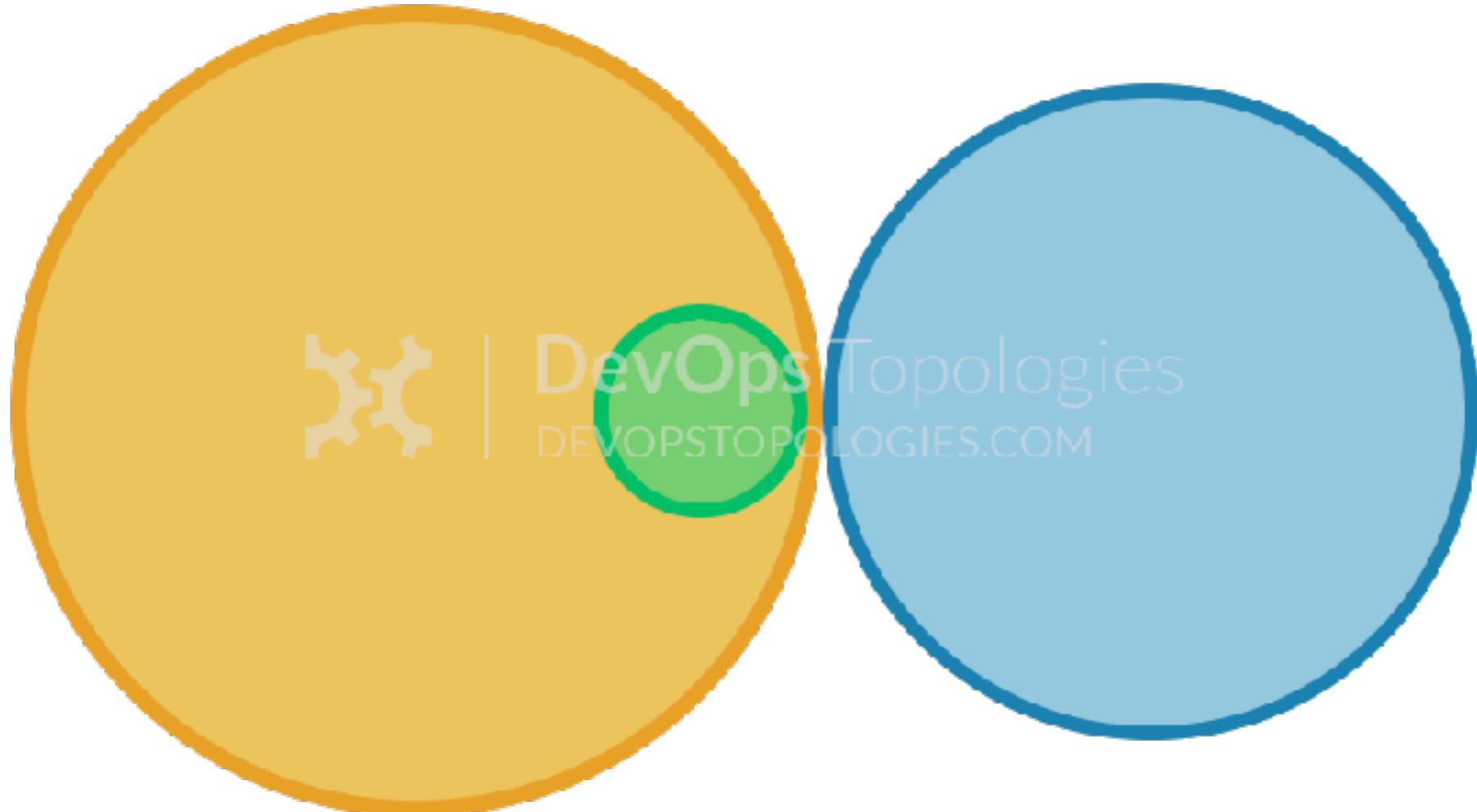
# Type 2 : Fully share ops responsibility



● Dev ● Ops



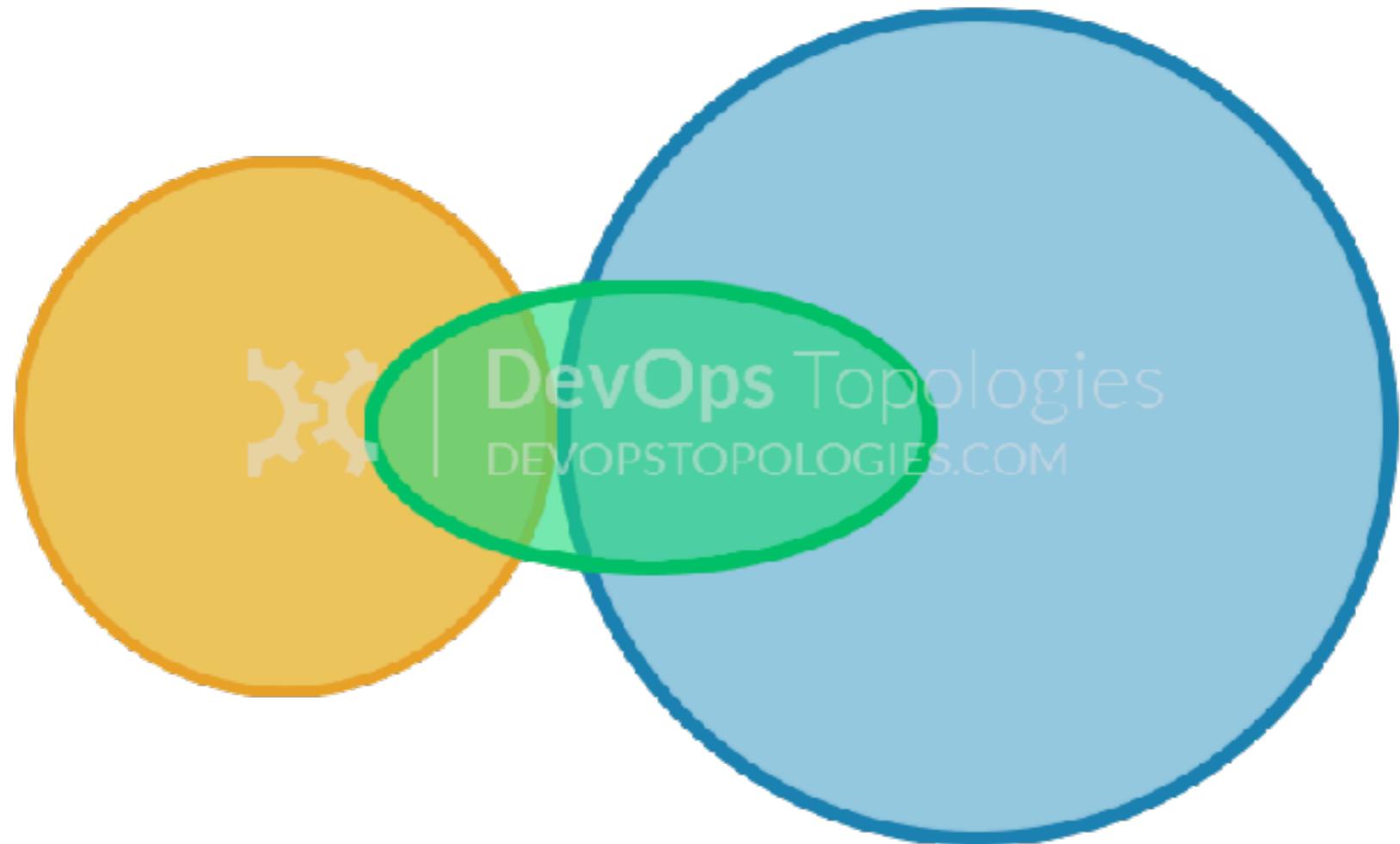
# Type 3 :Ops as infra as a service



- Dev
- DevOps
- Ops



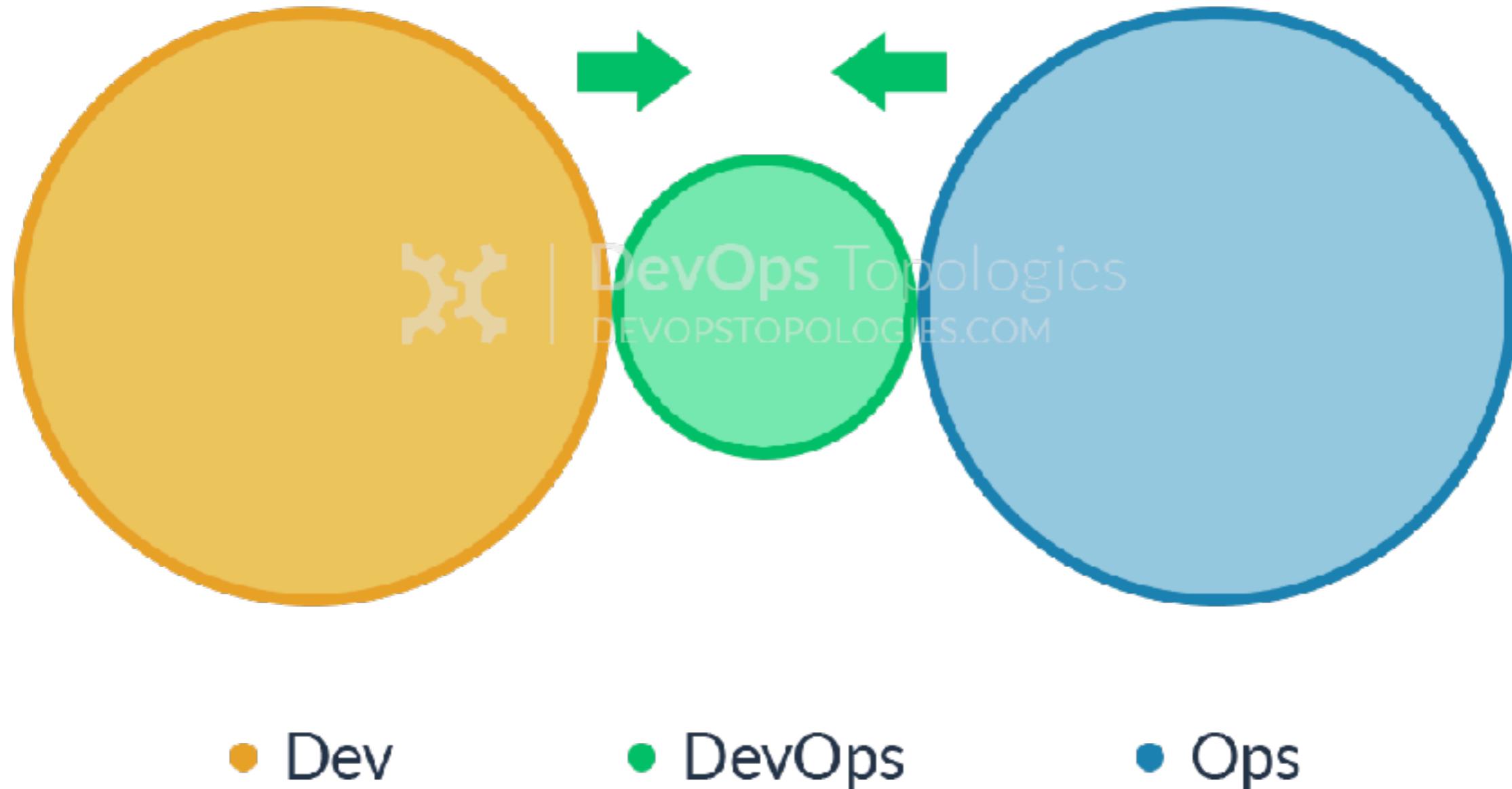
# Type 4 : DevOps as external service



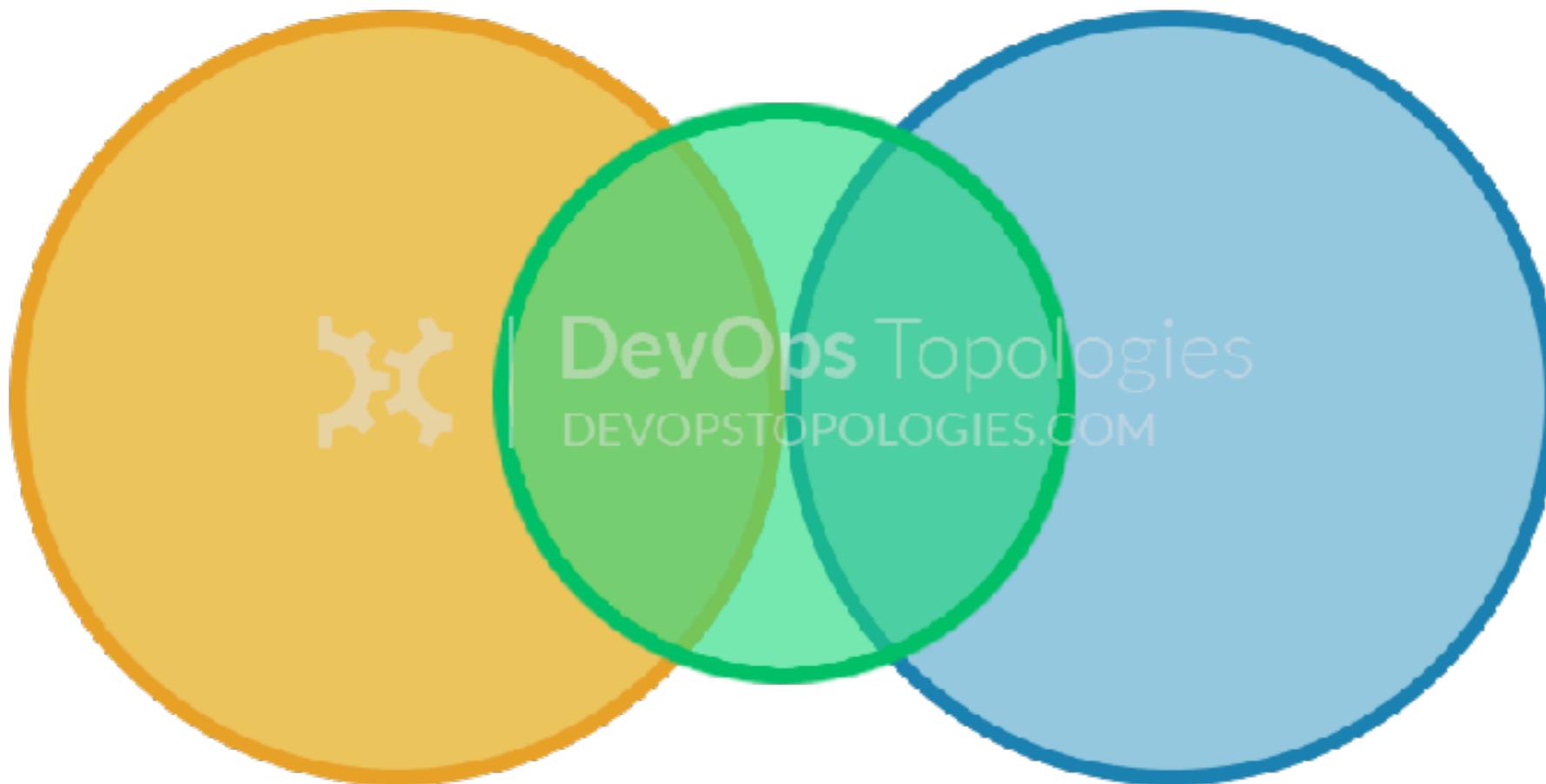
- Dev
- DevOps
- Ops



# Type 5 : DevOps team with expire date



# Type 6 : DevOps evangelist team



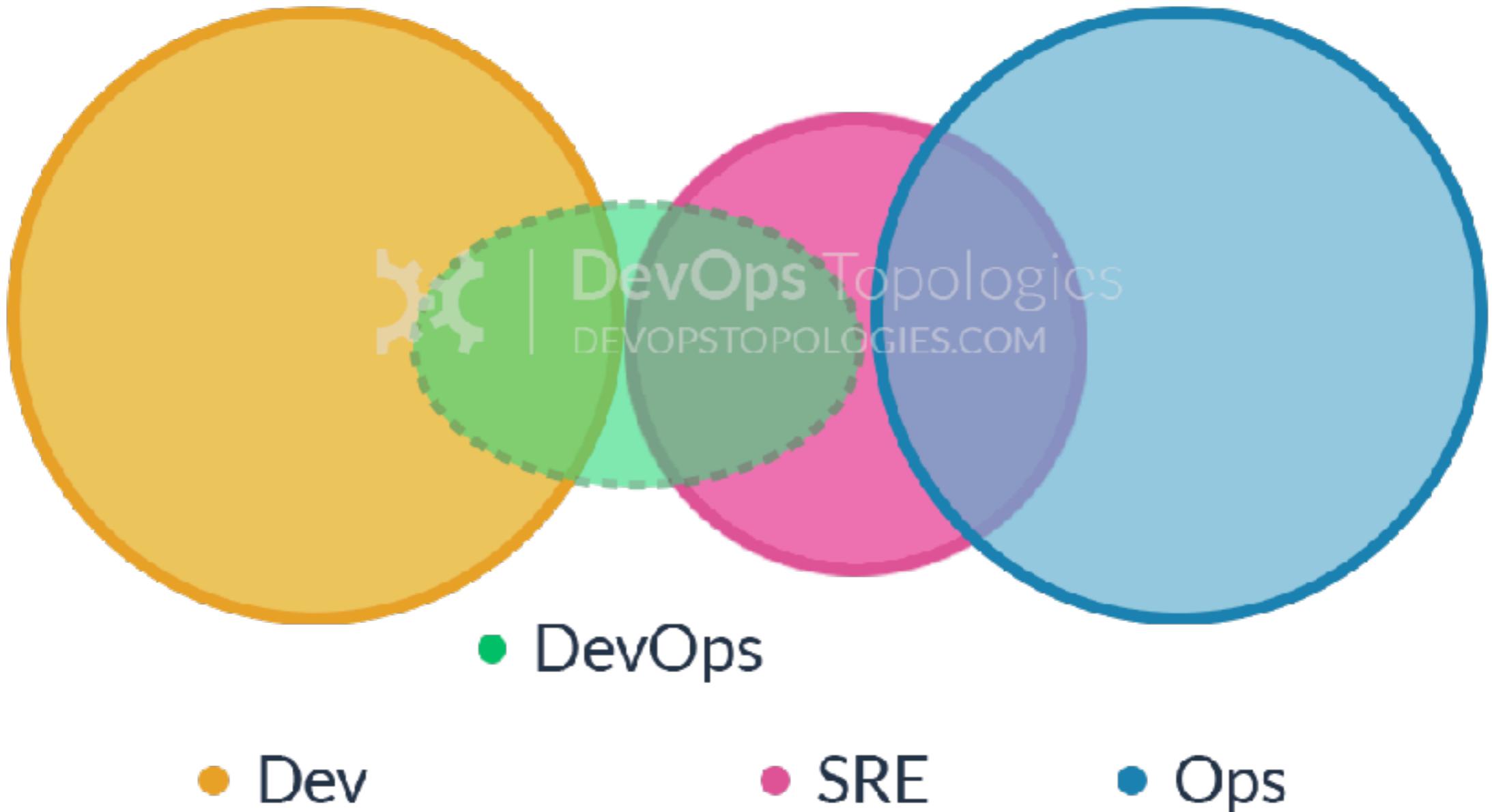
● Dev

● DevOps

● Ops



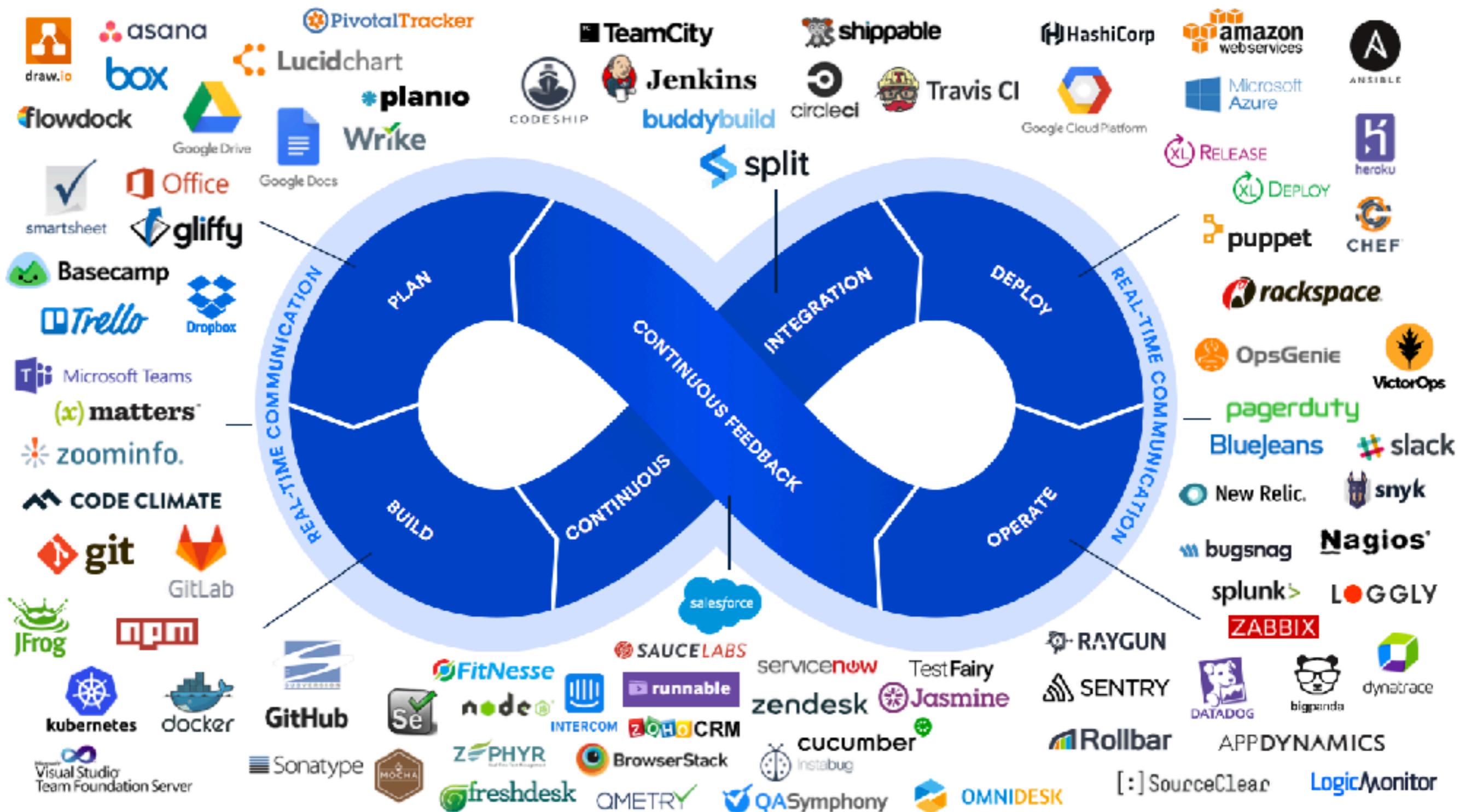
# Type 7 : SRE team (Google model)



# DevOps Tools



# DevOps Tools



# DevOps Tools

**PERIODIC TABLE OF DEVOPS TOOLS (V2)**

EMBED DOWNLOAD ADD

The Periodic Table of DevOps Tools is a grid-based visualization of various software tools used in the DevOps ecosystem. The table is organized into categories based on their primary function and source. The categories are:

- OpenSource**: Includes SCM, CI, Deployment, Cloud/IaaS/PaaS, and BI/Monitoring.
- Database Mgmt**: Includes Repo Mgmt, Config/Provisioning, Release Mgmt, and Logging.
- Build**: Includes Testing, Containerization, Collaboration, and Security.

The table also includes a legend for element symbols and their meanings:

- Gh (Yellow) = Github
- Gt (Yellow) = Git
- Bb (Yellow) = Bitbucket
- Gl (Yellow) = GitLab
- Sv (Yellow) = Subversion
- Hg (Yellow) = Mercurial
- Cw (Yellow) = ISPW
- Rh (Orange) = Jira
- Dm (Orange) = DBmaestro
- Lb (Orange) = Liquibase
- Rg (Orange) = Redgate
- Gt (Orange) = Radical
- Dt (Orange) = Dephtix
- Id (Orange) = Idea
- Mv (Teal) = Maven
- Gr (Teal) = Gradle
- Gp (Teal) = Gulp
- Gt (Teal) = Grunt
- Sb (Teal) = sbt
- Mk (Teal) = Make
- Ck (Teal) = CMake
- Pk (Teal) = Pecker
- MsB (Teal) = MSBuild
- At (Cyan) = ANT
- Fn (Cyan) = FitNesse
- Br (Cyan) = Broccoli
- Fn (Cyan) = Cucumber
- Jt (Cyan) = JUnit
- Pk (Cyan) = Pecker
- Mc (Red) = Mocha
- Xltv (Red) = XL TestView
- Jm (Red) = JMeter
- Nx (Red) = Nexus
- Se (Red) = Selenium
- Gatling (Red) = Gatling
- Cj (Red) = Cucumber.js
- Qu (Red) = Qunit
- Tn (Red) = TestNG
- Jm (Red) = Jasmine
- Ga (Red) = Docker Hub
- Dh (Red) = Docker Hub
- Jn (Red) = Jenkins
- Ba (Blue) = Bamboo
- Dh (Blue) = Docker Hub
- Jn (Blue) = Jenkins
- Tr (Blue) = Travis CI
- Tr (Blue) = CircleCI
- Ba (Blue) = Bamboo
- Tr (Blue) = Travis CI
- Gd (Blue) = Deployment Manager
- Sf (Blue) = Smartfrog
- Cn (Blue) = Consul
- Bc (Blue) = BoltDB
- Mo (Blue) = Mesos
- Rs (Blue) = Rackspace
- Gp (Green) = Gulp
- Cu (Green) = Cucumber
- Cj (Green) = Cucumber.js
- Qu (Green) = Qunit
- Npm (Green) = npm
- Cs (Green) = CodeShip
- Vs (Green) = Visual Studio
- Cr (Green) = CircleCI
- Cp (Green) = Capistrano
- Ju (Green) = Juju
- Rd (Green) = Rundeck
- Cf (Green) = CFEngine
- Ds (Green) = Swarm
- Op (Green) = OpenStack
- Mk (Green) = Make
- Ck (Green) = CMake
- Jt (Green) = JUnit
- Jm (Green) = JMeter
- Tn (Green) = TestNG
- Ay (Green) = Artifactory
- Tc (Green) = TeamCity
- Sh (Green) = Shippable
- Cc (Green) = CruiseControl
- Ry (Green) = RapidDeploy
- Cy (Green) = Octopus Deploy
- Oc (Green) = Octopus Deploy
- No (Green) = CANollo
- Kb (Green) = Kubernetes
- Hr (Green) = Heroku
- re (Green) = Reke
- 7t (Green) = TestNG
- 75 (Green) = Os
- 76 (Green) = En
- 77 (Green) = Fr
- 78 (Green) = Os
- 79 (Green) = En
- 80 (Green) = Os
- 81 (Green) = Os
- 82 (Green) = Os
- 83 (Green) = Fr
- 84 (Green) = Pj
- 85 (Green) = En
- 86 (Green) = En
- 87 (Green) = Fr
- 88 (Green) = En
- 89 (Green) = Cs
- 90 (Green) = En
- 91 (Green) = Fr
- 92 (Green) = Fn
- 93 (Green) = Fn
- 94 (Green) = Fn
- 95 (Green) = Fn
- 96 (Green) = Fn
- 97 (Green) = Fn
- 98 (Green) = Fn
- 99 (Green) = Pd
- 100 (Green) = Fr
- 101 (Green) = Fr
- 102 (Green) = Fr
- 103 (Green) = Fr
- 104 (Green) = Ph
- 105 (Green) = Fn
- Xlr (Green) = XL Release
- Ur (Green) = UrbanCode Release
- Bm (Green) = BMC Release Process
- Hp (Green) = HP Codar
- Au (Green) = Atomic
- Pl (Green) = Platina Release
- Sr (Green) = Serena Release
- Tfs (Green) = Team Foundation
- Tr (Green) = Telco
- Jr (Green) = Jira
- Rf (Green) = HipChat
- Sl (Green) = Slack
- Fd (Green) = Flowdock
- Pv (Green) = Pivotal Tracker
- Sn (Green) = ServiceNow
- Ki (Green) = Kibana
- Nr (Green) = New Relic
- Ni (Green) = Nagios
- Zb (Green) = Zabbix
- Dd (Green) = Datadog
- Ei (Green) = Elasticsearch
- St (Green) = StackStorm
- Sp (Green) = Splunk
- Le (Green) = Logentries
- Sl (Green) = Sumologic
- Ls (Green) = Logstash
- Gr (Green) = Graylog
- Sn (Green) = Snort
- Tr (Green) = Tripwire
- Ff (Green) = Fortify

**XebiaLabs**  
Deliver Faster

Follow @xebialabs

91	Fr	92	Fn	93	Fn	94	Fn	95	Fn	96	Fn	97	Fn	98	Pd	99	Fr	100	Ph	101	Fr	102	Fr	103	Fr	104	Ph	105	Fn
Xlr		Ur		Bm		Hp		Au		Pl		Sr		Tfs		Tr		Jr		Rf		Sl		Fd		Pv		Sn	
XL Release		UrbanCode Release		BMC Release Process		HP Codar		Atomic		Platina Release		Serenia Release		Team Foundation		Telco		Jira		HipChat		Slack		Flowdock		Pivotal Tracker		ServiceNow	
106	Os	107	Fr	108	Os	109	Cs	110	En	111	Os	112	En	113	En	114	Fr	115	Fn	116	Os	117	Os	118	Os	119	Cs	120	En
Ki		Nr		Ni		Zb		Dd		Ei		St		Sp		Le		Sl		Ls		Gr		Sn		Tr		Ff	
Kibana		New Relic		Nagios		Zabbix		Datadog		Elasticsearch		StackStorm		Splunk		Logentries		Sumologic		Logstash		Graylog		Snort		Tripwire		Fortify	

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



DevOps

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

130

# 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

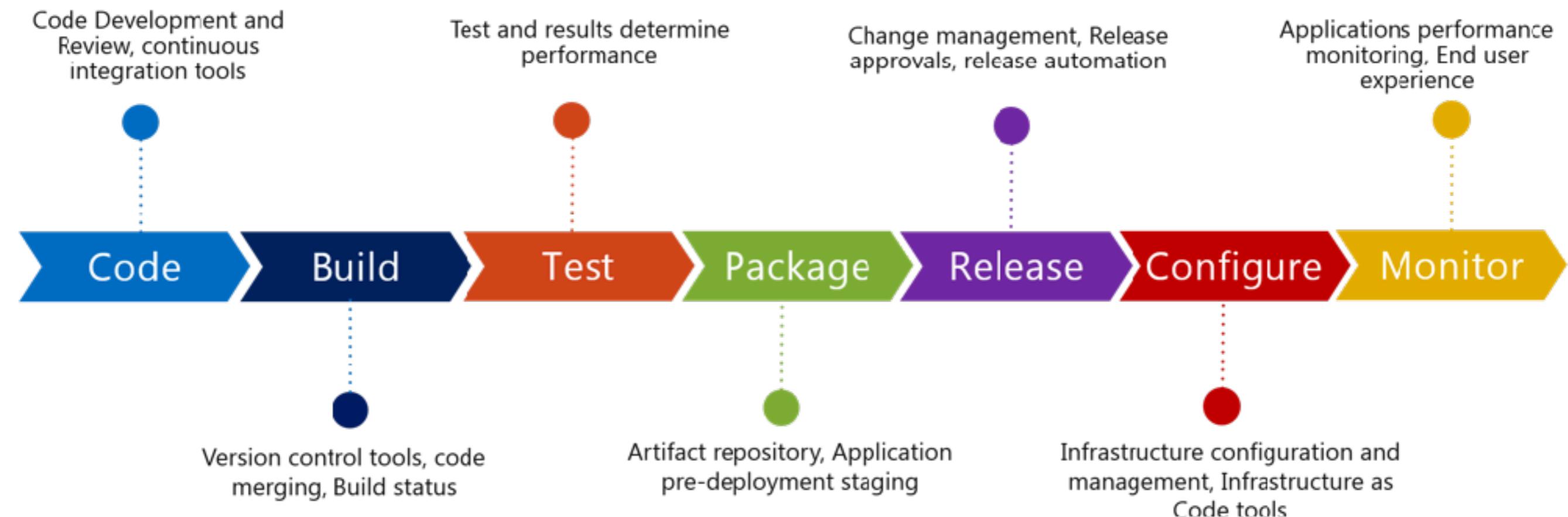
Changes cost too much



# 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 (%)

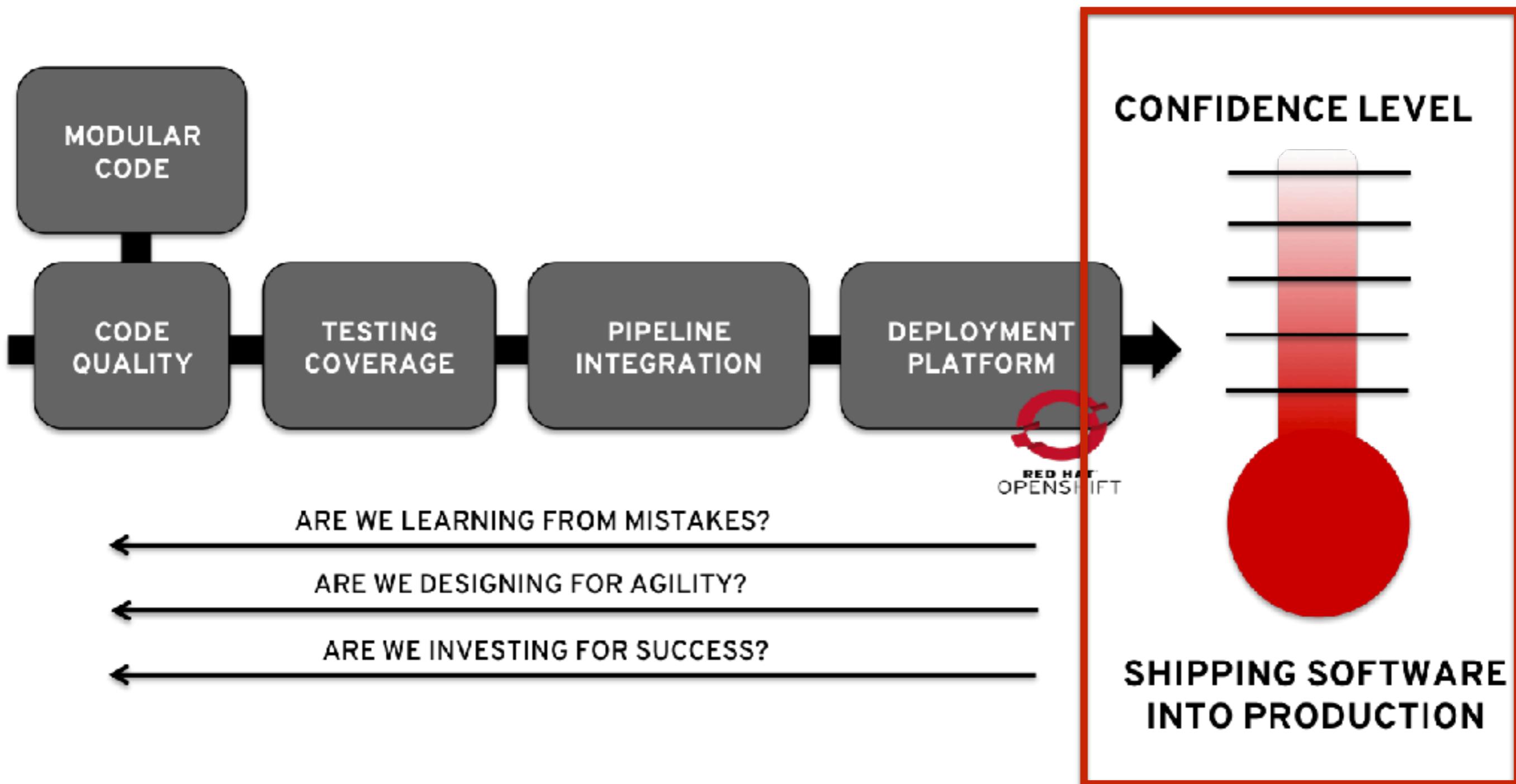
Defect Rate

Failed Deployments

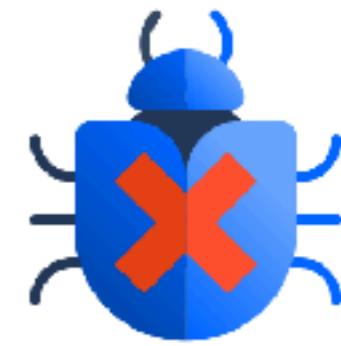
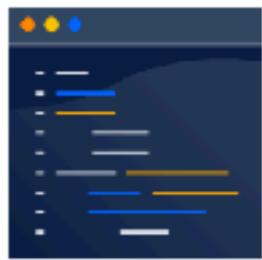
Availability



# 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 ?

```
top - 13:46:48 up 547 days, 1:50, 2 users, load average: 0. Tasks: 130 total, 1 running, 129 sleeping, 0 stopped, 0 Cpu(s): 10.2%us, 3.1%sy, 0.0%ni, 85.7%id, 0.0%wa, 0.1%hi, Mem: 4051984k total, 3975528k used, 76456k free, 8718 Swap: 4184924k total, 866144k used, 3318780k free, 319199
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+
cssshX - root@localhost:1111 - ssh										

```
top - 13:46:48 up 545 days, 19:14, 1 user, load average: 1.2 Tasks: 129 total, 2 running, 127 sleeping, 0 stopped, 0 Cpu(s): 9.5%us, 2.7%sy, 0.0%ni, 86.1%id, 1.2%wa, 0.0%hi, Mem: 4051984k total, 3912260k used, 139724k free, 8952 Swap: 4184924k total, 535520k used, 3649404k free, 270435
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+
cssshX - root@localhost:1112 - ssh										

```
top - 13:46:48 up 342 days, 54 min, 3 users, load average: 1 Tasks: 119 total, 1 running, 118 sleeping, 0 stopped, 0 Cpu(s): 20.7%us, 2.9%sy, 0.0%ni, 68.4%id, 0.0%wa, 0.0%hi, Mem: 4051984k total, 3230568k used, 821416k free, 10660 Swap: 4184924k total, 69316k used, 4115608k free, 197159
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+
cssshX - root@localhost:1113 - ssh										

```
top - 13:46:48 up 547 days, 21:17, 2 users, load average: 0. Tasks: 129 total, 1 running, 128 sleeping, 0 stopped, 0 Cpu(s): 6.8%us, 2.0%sy, 0.0%ni, 91.0%id, 0.1%wa, 0.0%hi, Mem: 4051984k total, 3939624k used, 112360k free, 8444 Swap: 4184924k total, 886960k used, 3297964k free, 309277
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+
cssshX - root@localhost:1114 - ssh										

```
top - 13:46:48 up 547 days, 21:22, 2 users, load average: 0. Tasks: 129 total, 2 running, 127 sleeping, 0 stopped, 0 Cpu(s): 6.1%us, 1.9%sy, 0.0%ni, 89.7%id, 1.9%wa, 0.1%hi, Mem: 4051984k total, 3954180k used, 97804k free, 8610 Swap: 4184924k total, 819268k used, 3365656k free, 300797
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+
cssshX - root@localhost:1115 - ssh										

```
top - 13:46:48 up 552 days, 2:37, 2 users, load average: 0. Tasks: 129 total, 1 running, 128 sleeping, 0 stopped, 0 Cpu(s): 8.7%us, 2.4%sy, 0.0%ni, 88.4%id, 0.0%wa, 0.1%hi, Mem: 4051984k total, 3661848k used, 390136k free, 8683
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+
cssshX - root@localhost:1116 - ssh										

```
top - 13:46:48 up 545 days, 19:14, 1 user, load average: 0.6 Tasks: 129 total, 2 running, 127 sleeping, 0 stopped, 0 Cpu(s): 8.5%us, 2.1%sy, 0.0%ni, 88.0%id, 0.8%wa, 0.0%hi, Mem: 4051984k total, 3277304k used, 774680k free, 9450
```



# Do you know your system ?



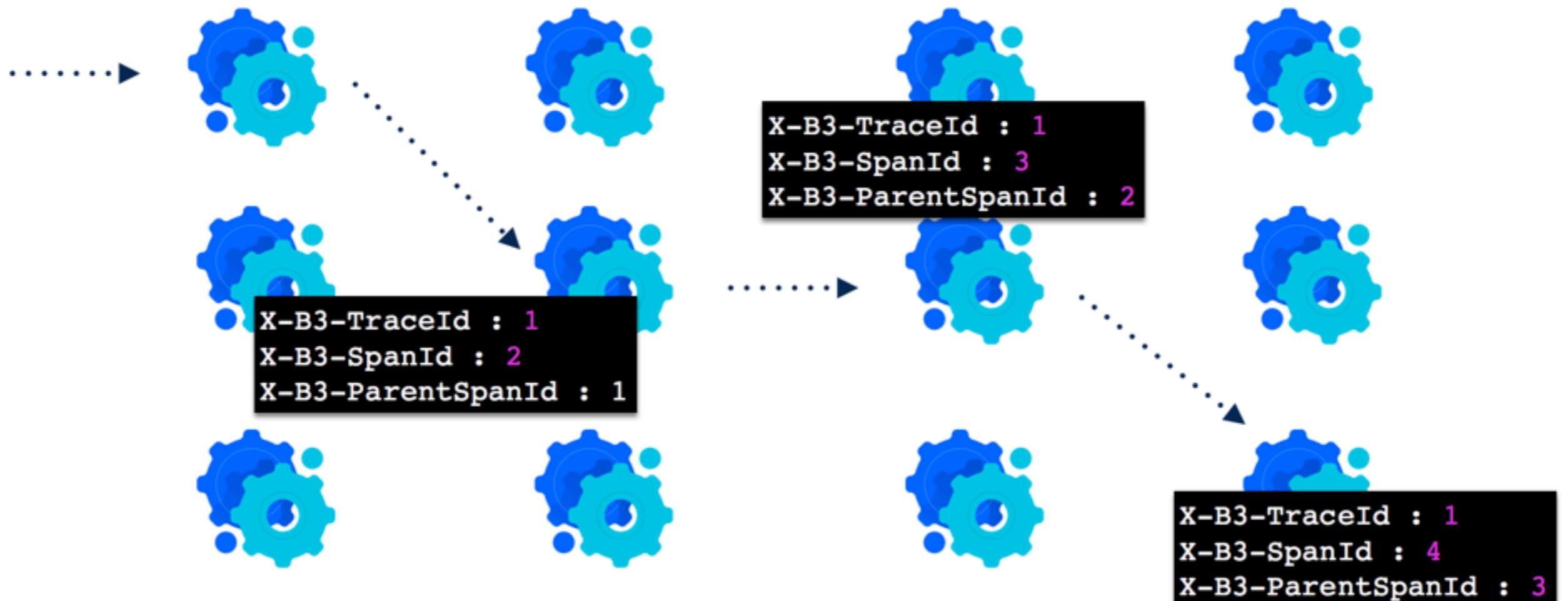
# Do you know your system ?



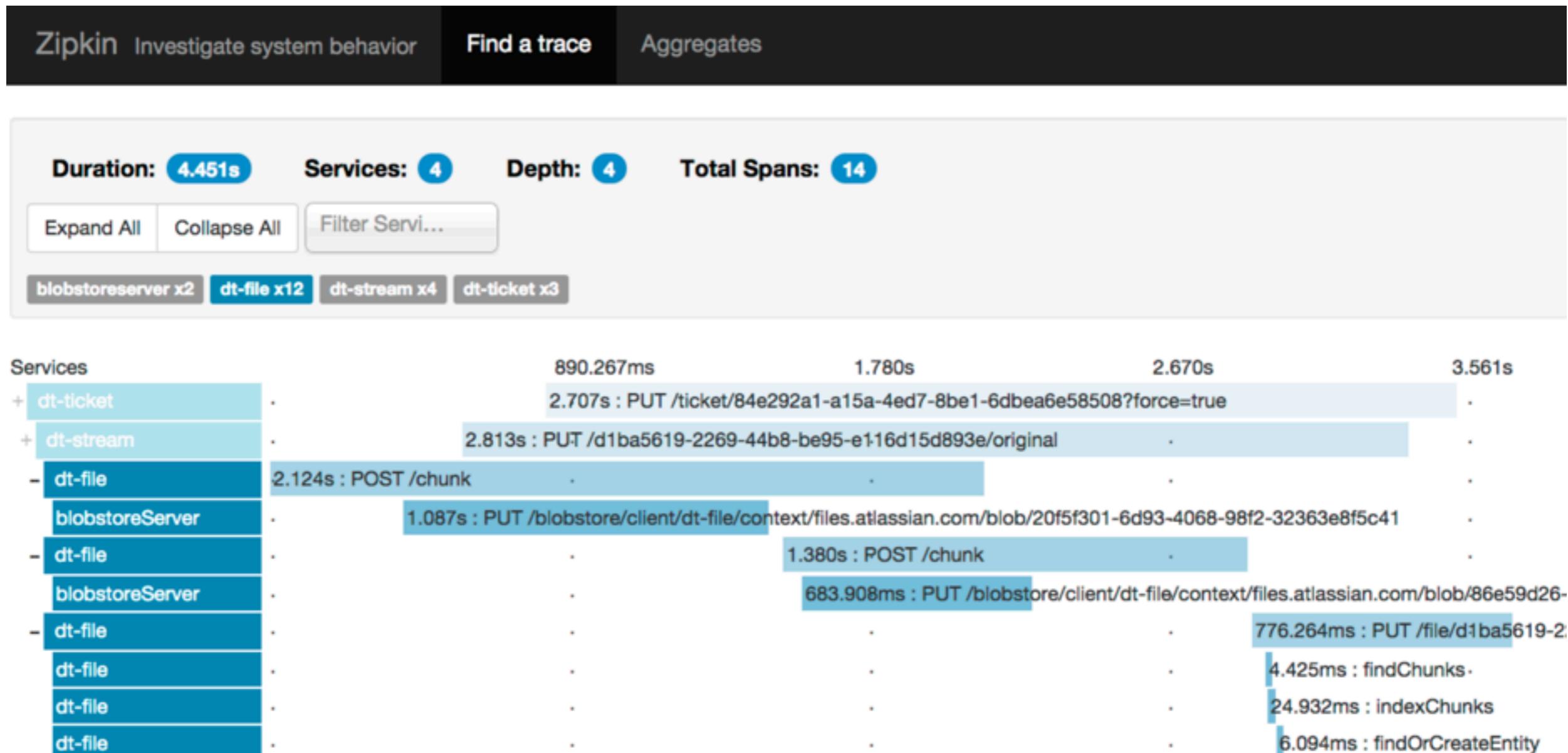
# Do you know your system ?

X-B3-TraceId : 1  
X-B3-SpanId : 1

## Request Tracing



# Do you know your system ?





# You Build It You Run It

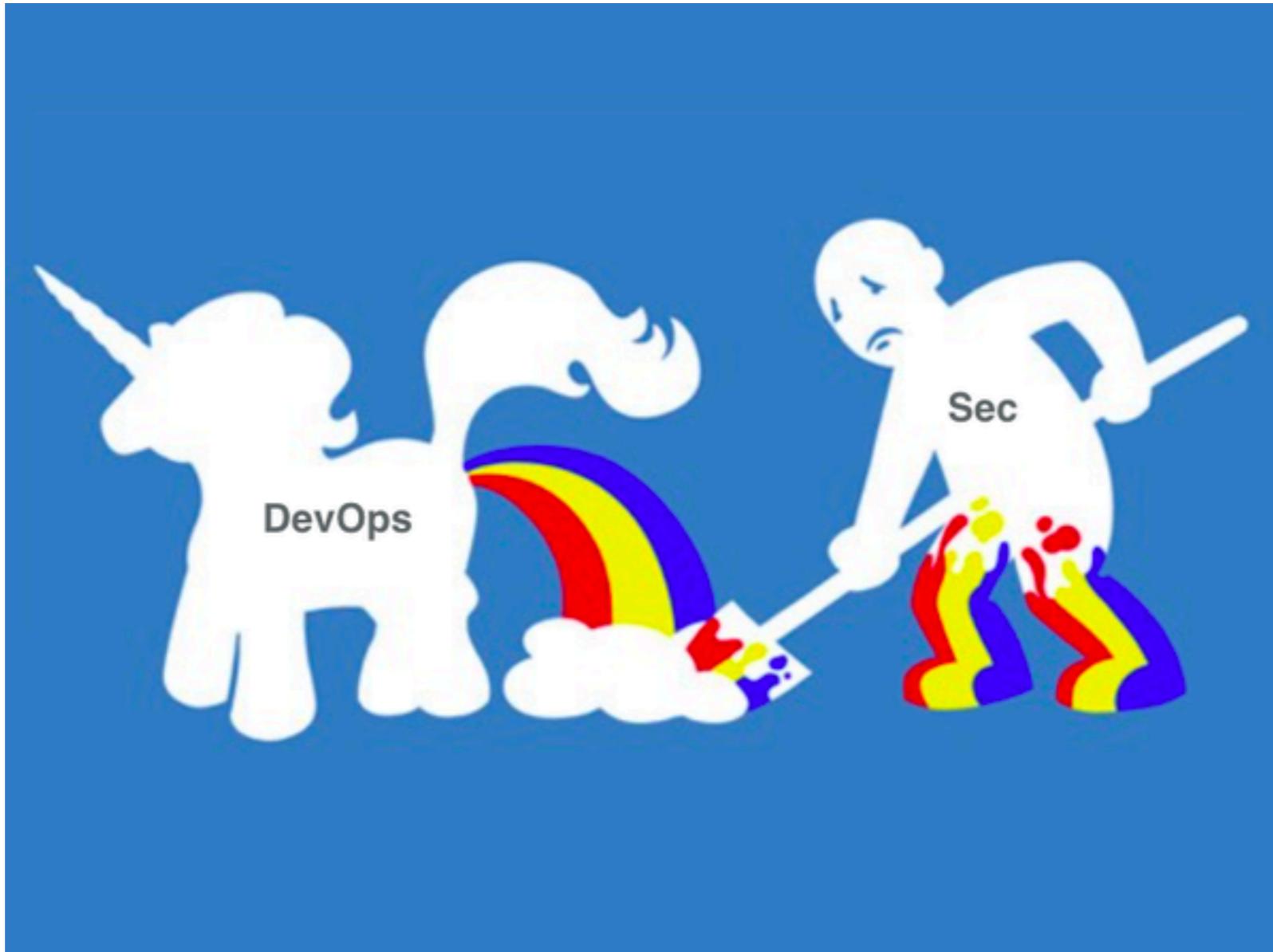
The team who builds it looks after it.

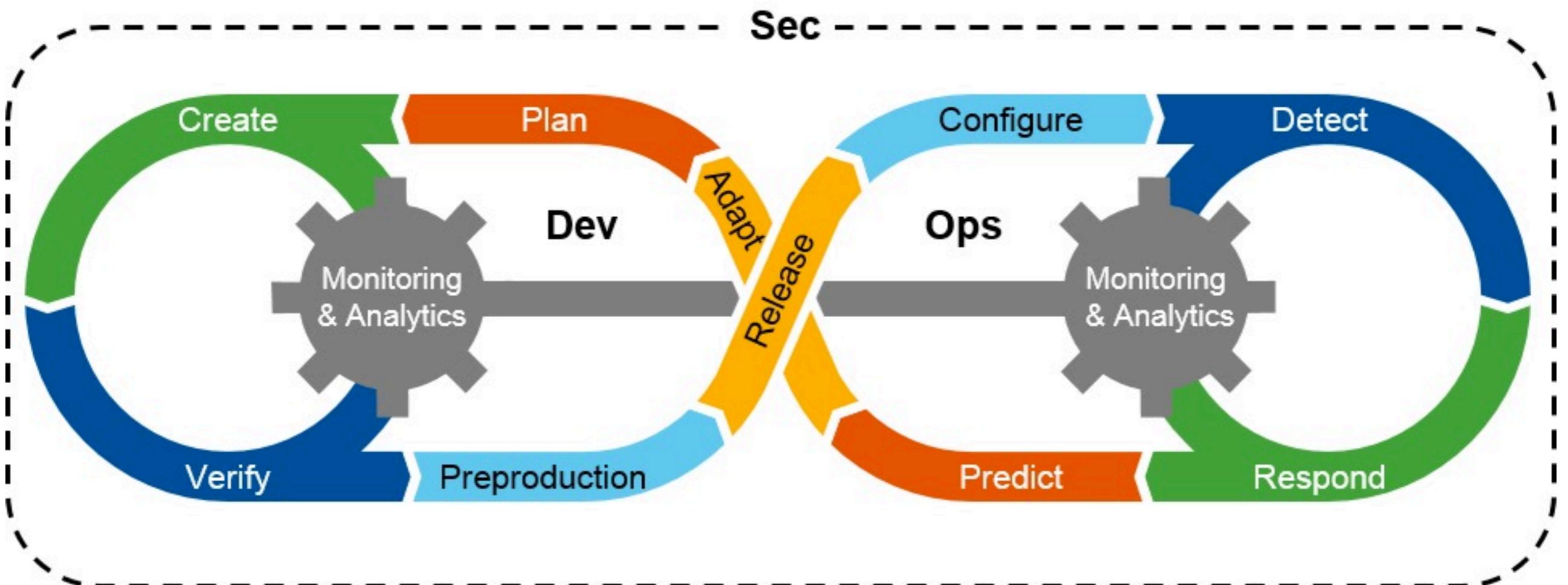


# more ...

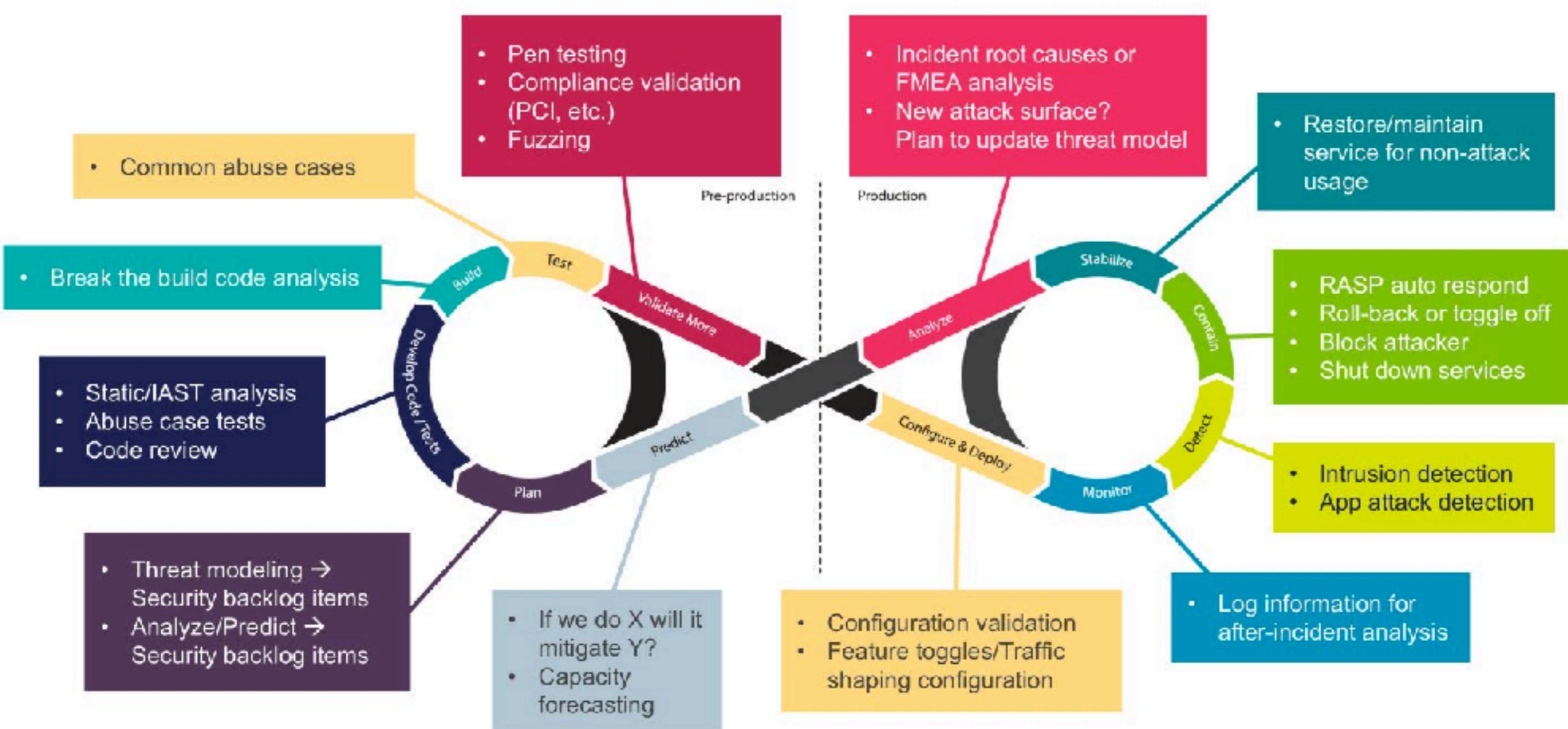






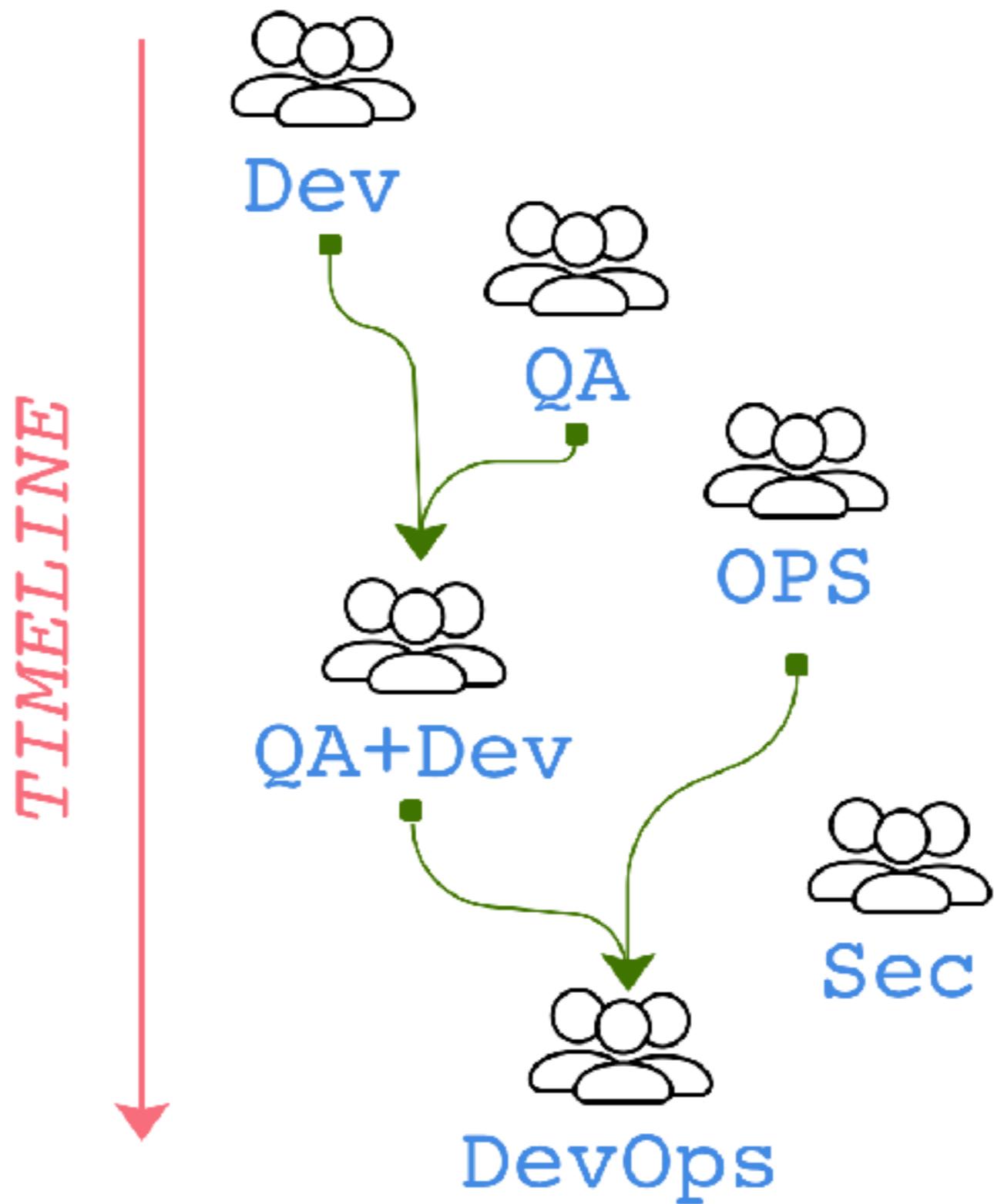


# DevSecOps Cycle



[LinkedIn.com/in/LarryMaccherone](https://LinkedIn.com/in/LarryMaccherone)

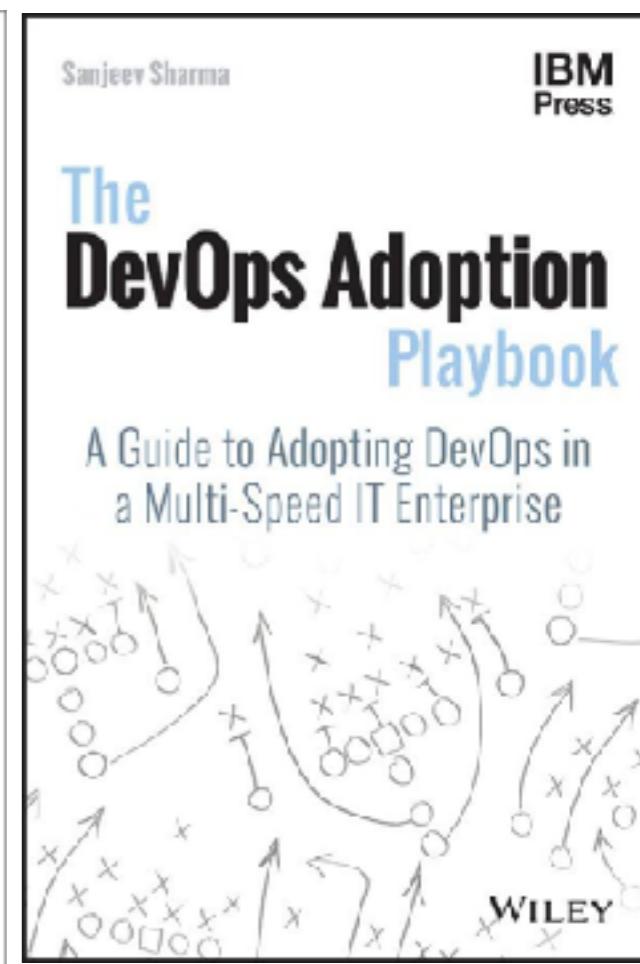
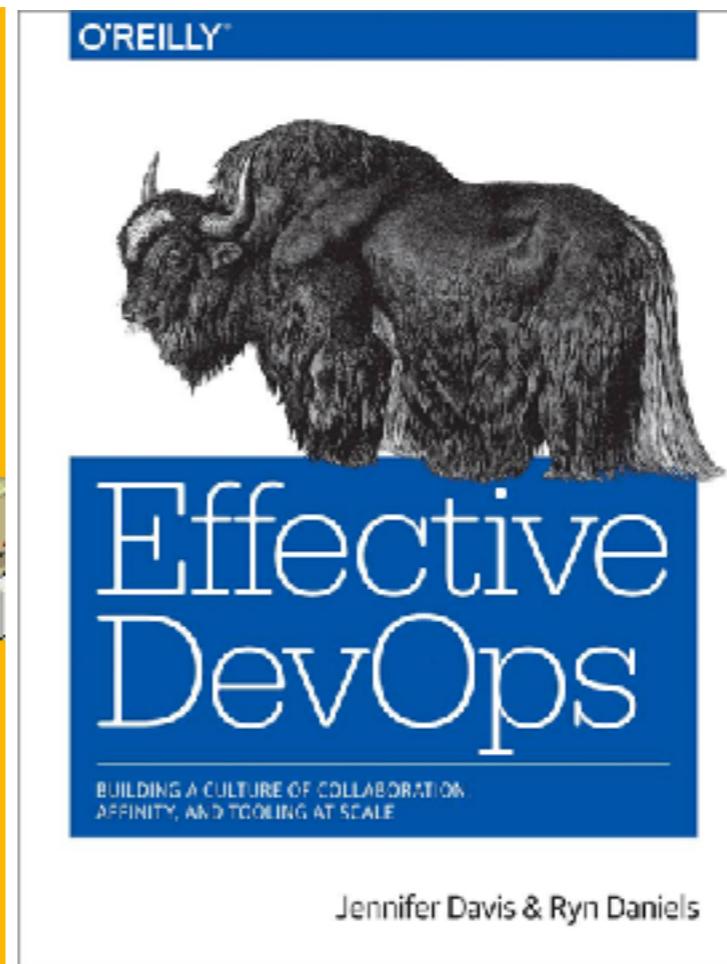
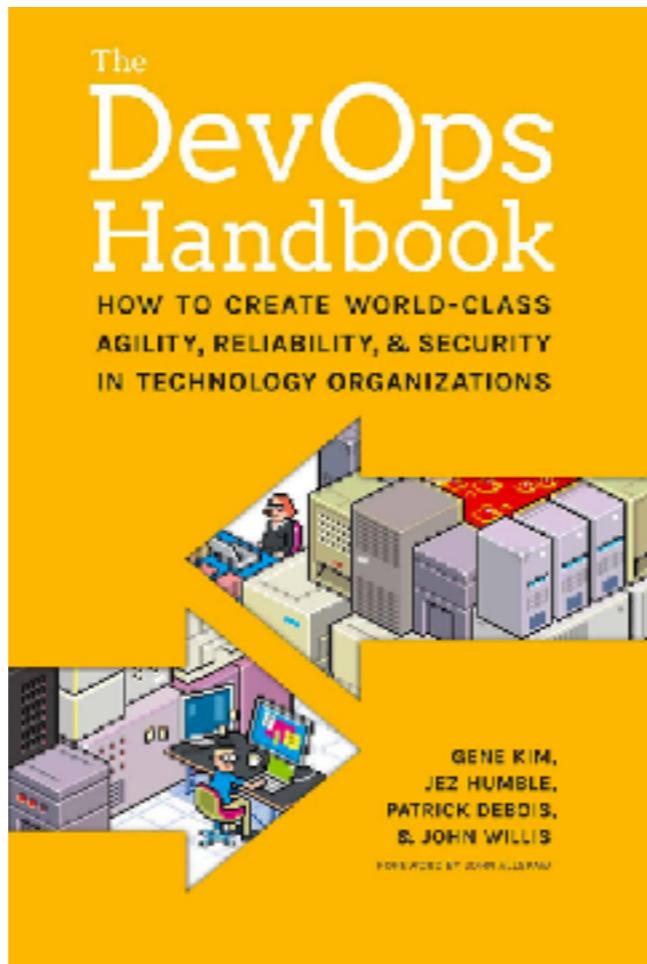
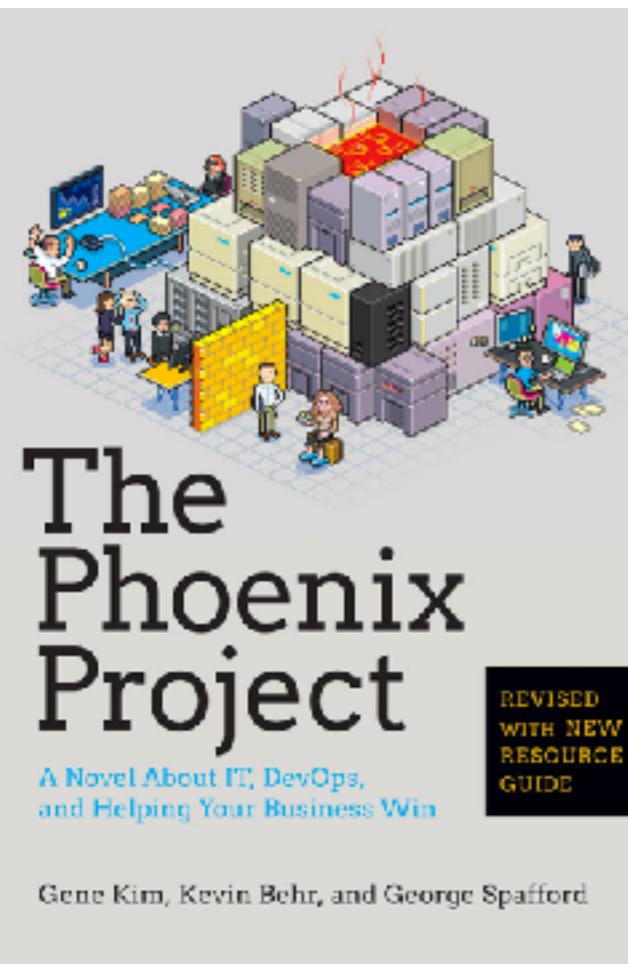




<https://medium.com/odds-team/%E0%B8%AA%E0%B8%A3%E0%B8%B8%E0%B8%9B-devops-is-dev%EA%9D%8B%EA%9E%A9-by-chai-feng-bb61b0460af0>



# Resources



# DevOps !!



# Back to basic



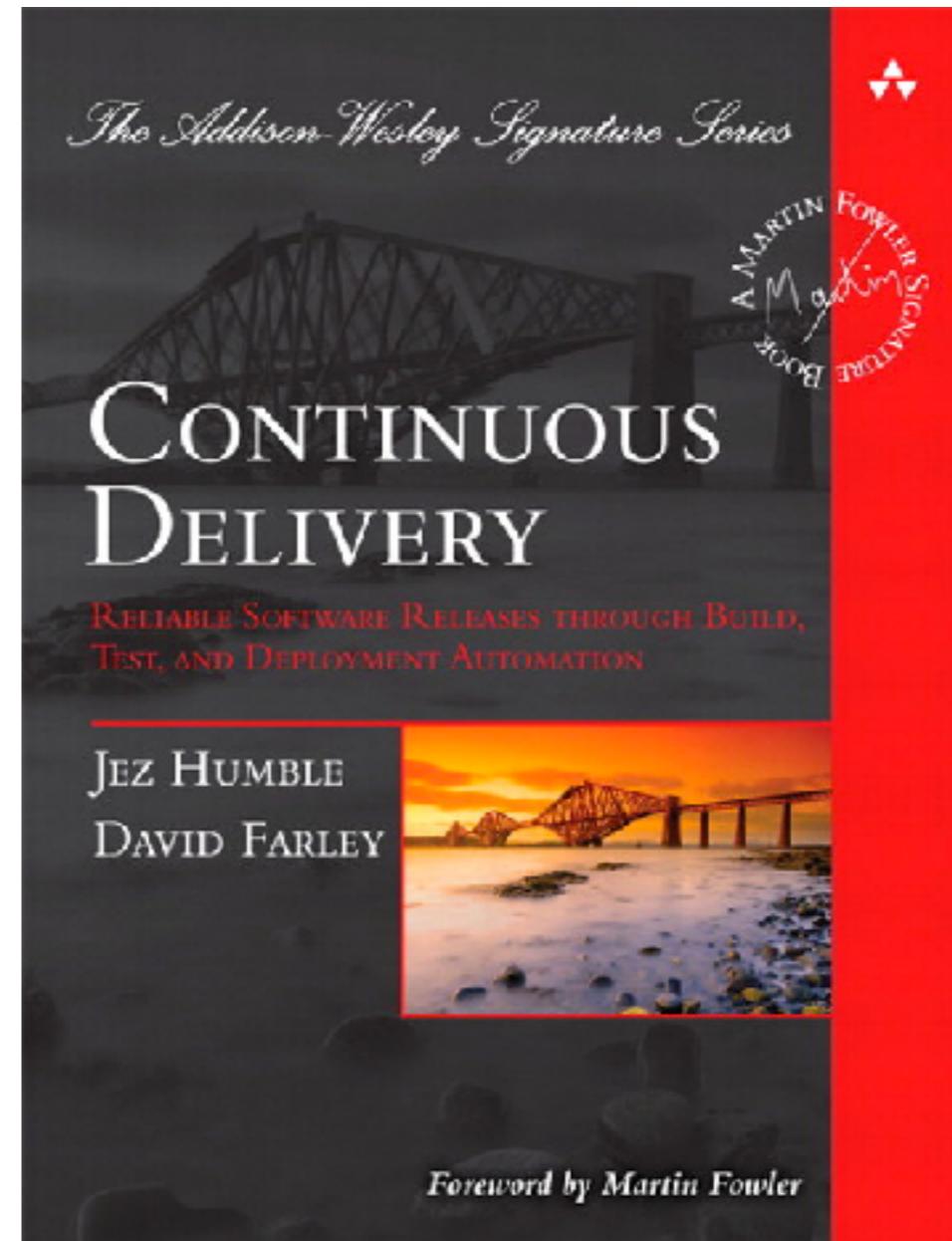
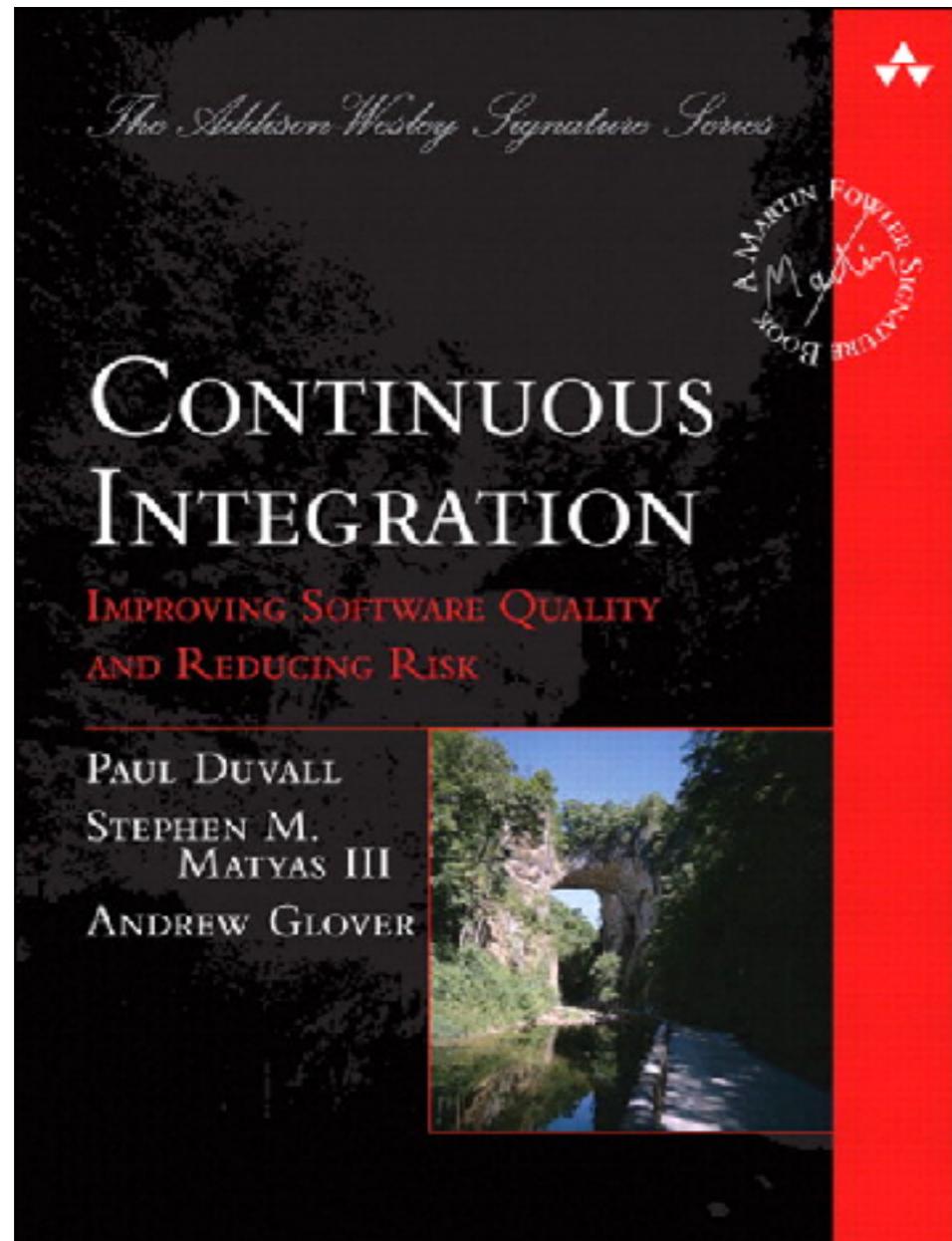
**“Behind every successful agile  
project, there is a  
Continuous Integration”**



# **Start with Continuous Integration Continuous Delivery**



# Improve quality and reduce risk

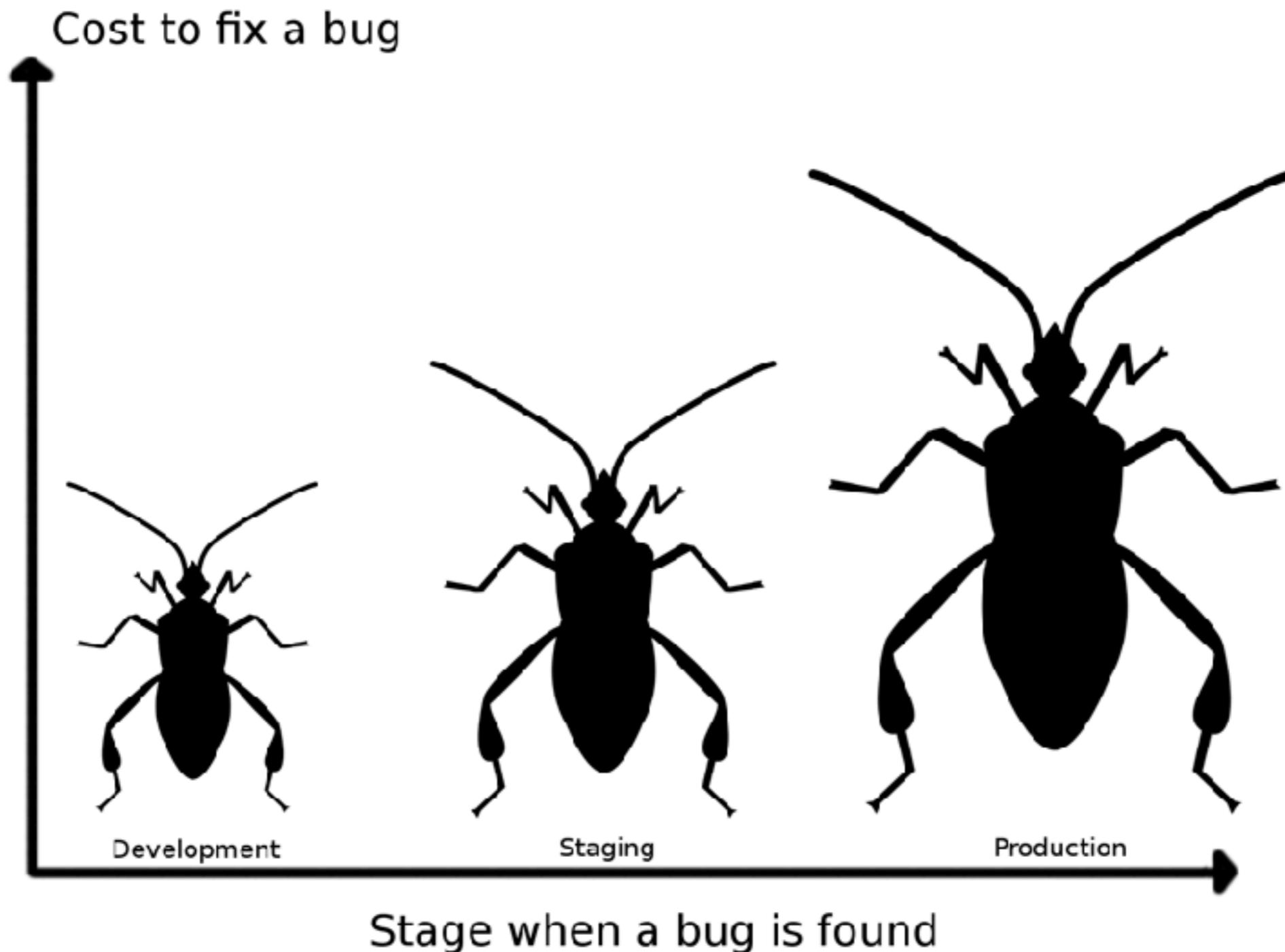


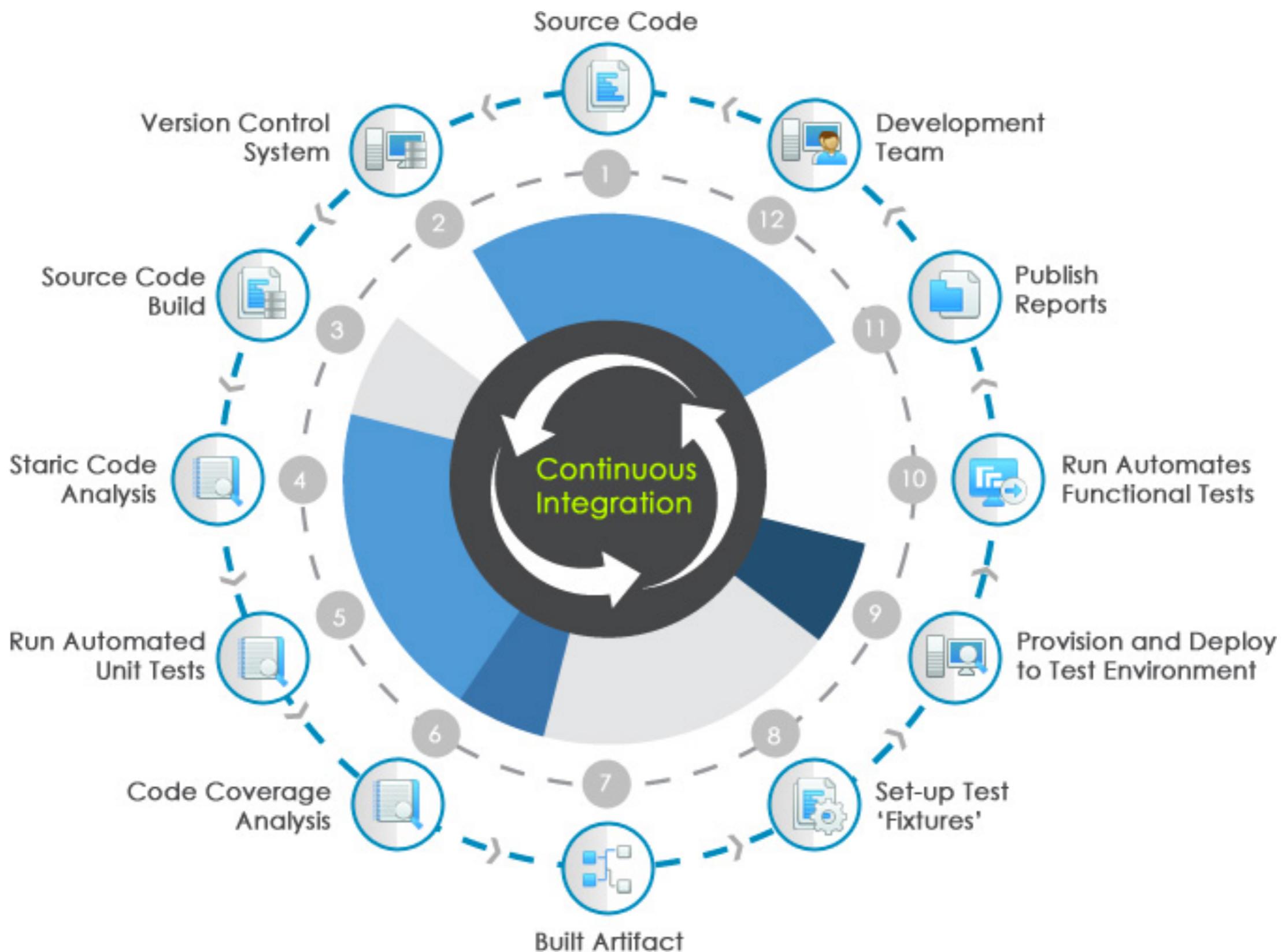
# The cost of integration

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<sup>TM</sup>



Hudson

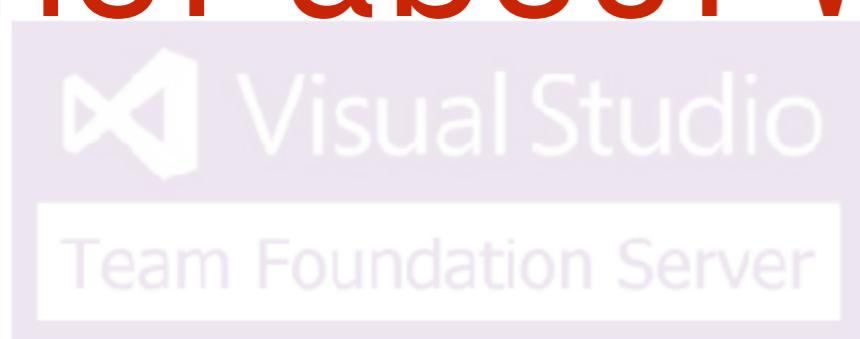




Jenkins

Bamboo

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



Hudson



# Continuous Integration

Discipline to integrate frequently



# Continuous Integration

Strive to make **small change**

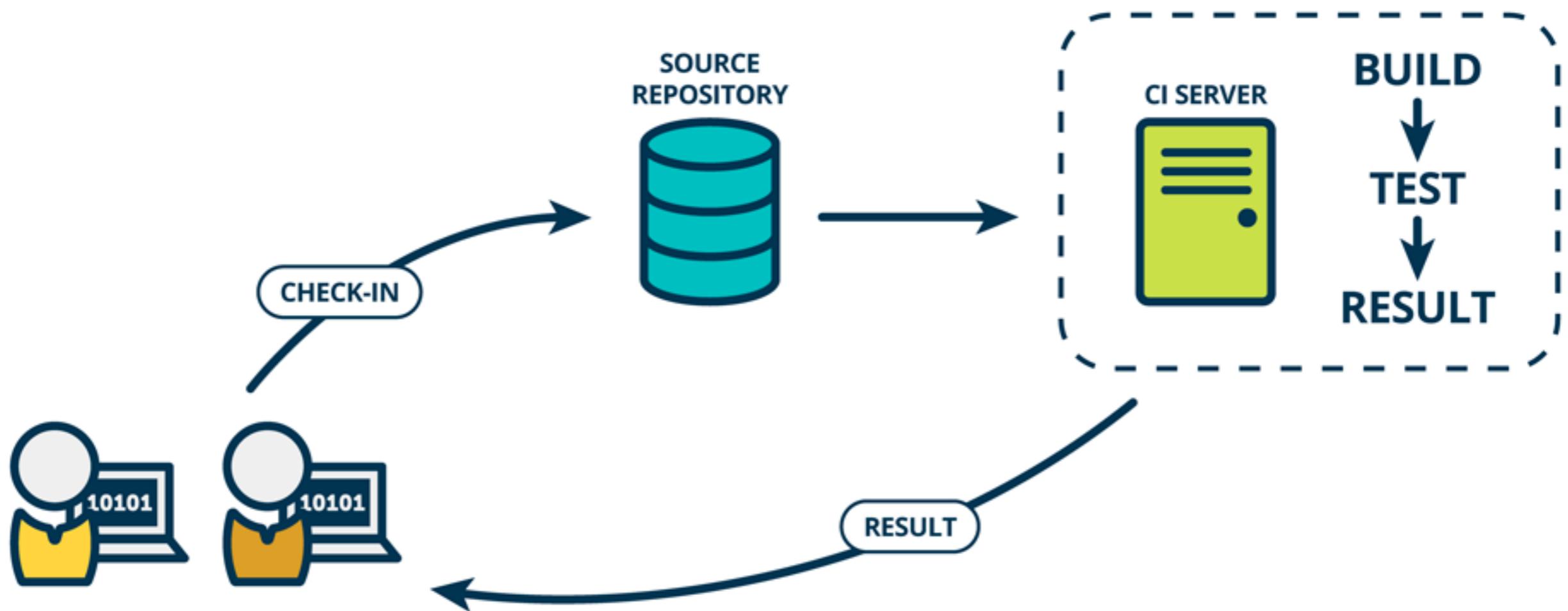


# Continuous Integration

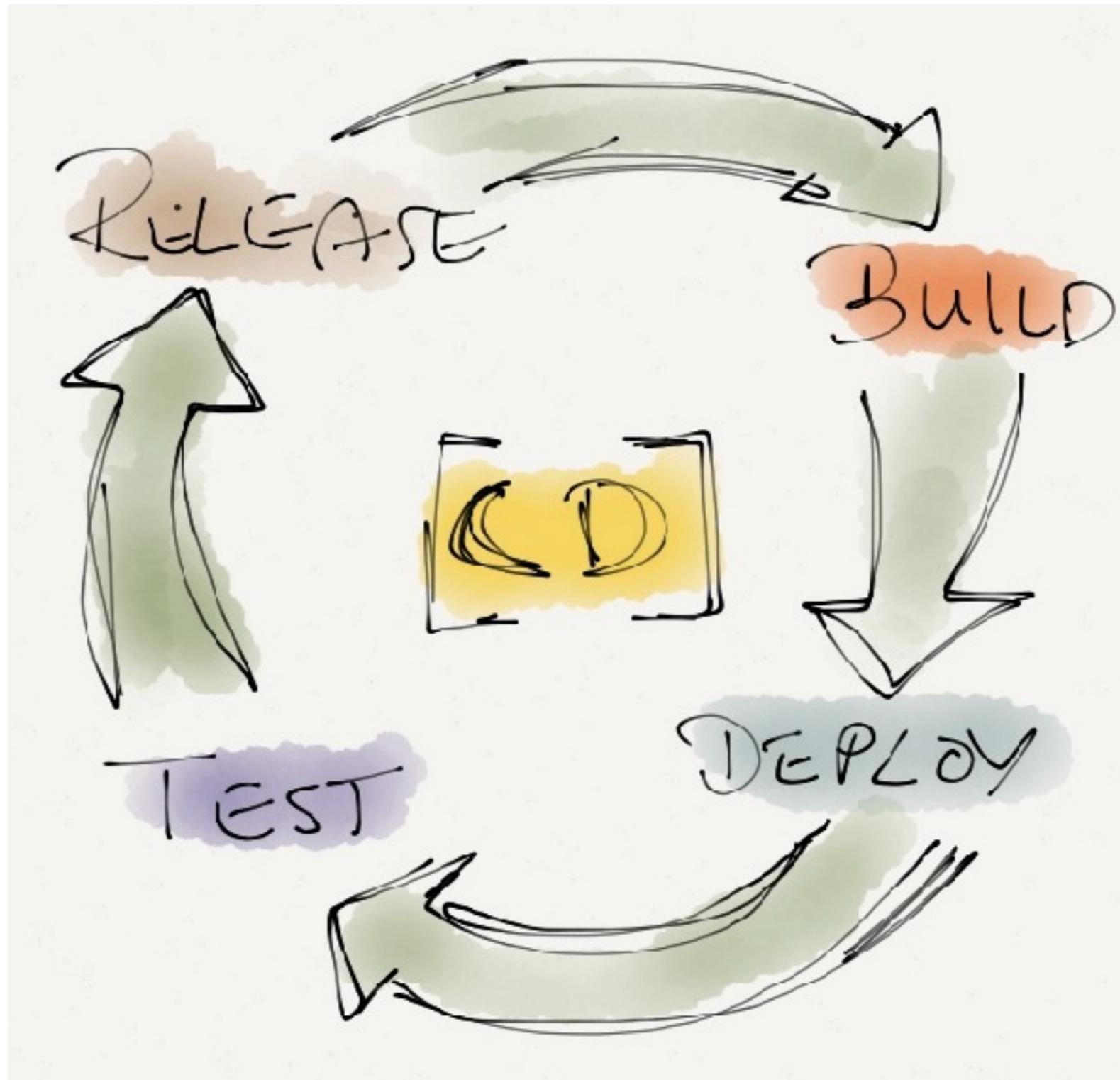
Strive for **fast feedback**



# Continuous Integration



# CD ?



# CD ?

## CONTINUOUS DELIVERY



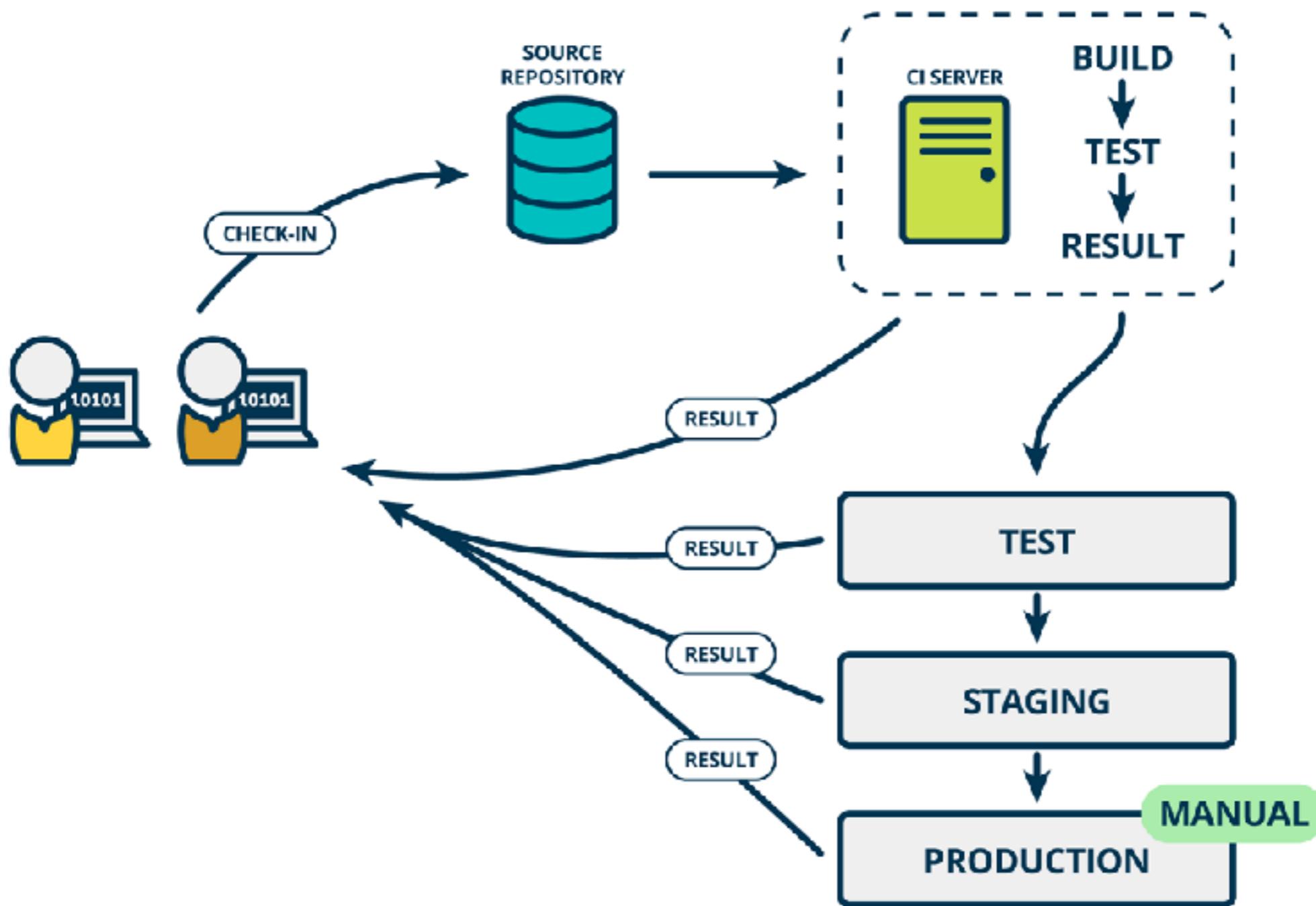
## CONTINUOUS DEPLOYMENT



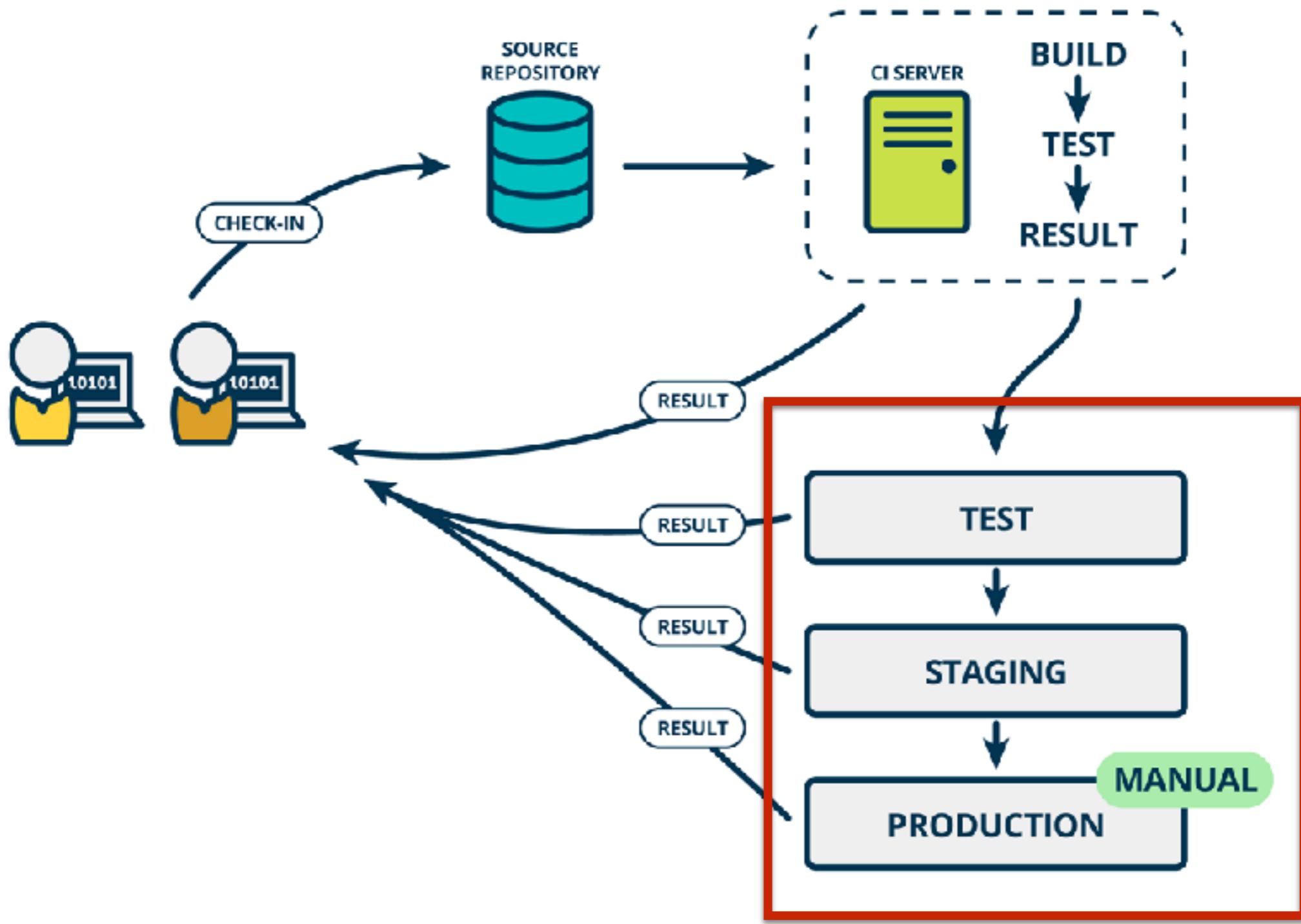
<http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment>



# Continuous Delivery



# Rise of DevOps



# **Continuous Integration**

**is a Software development practices**



# Practice 1

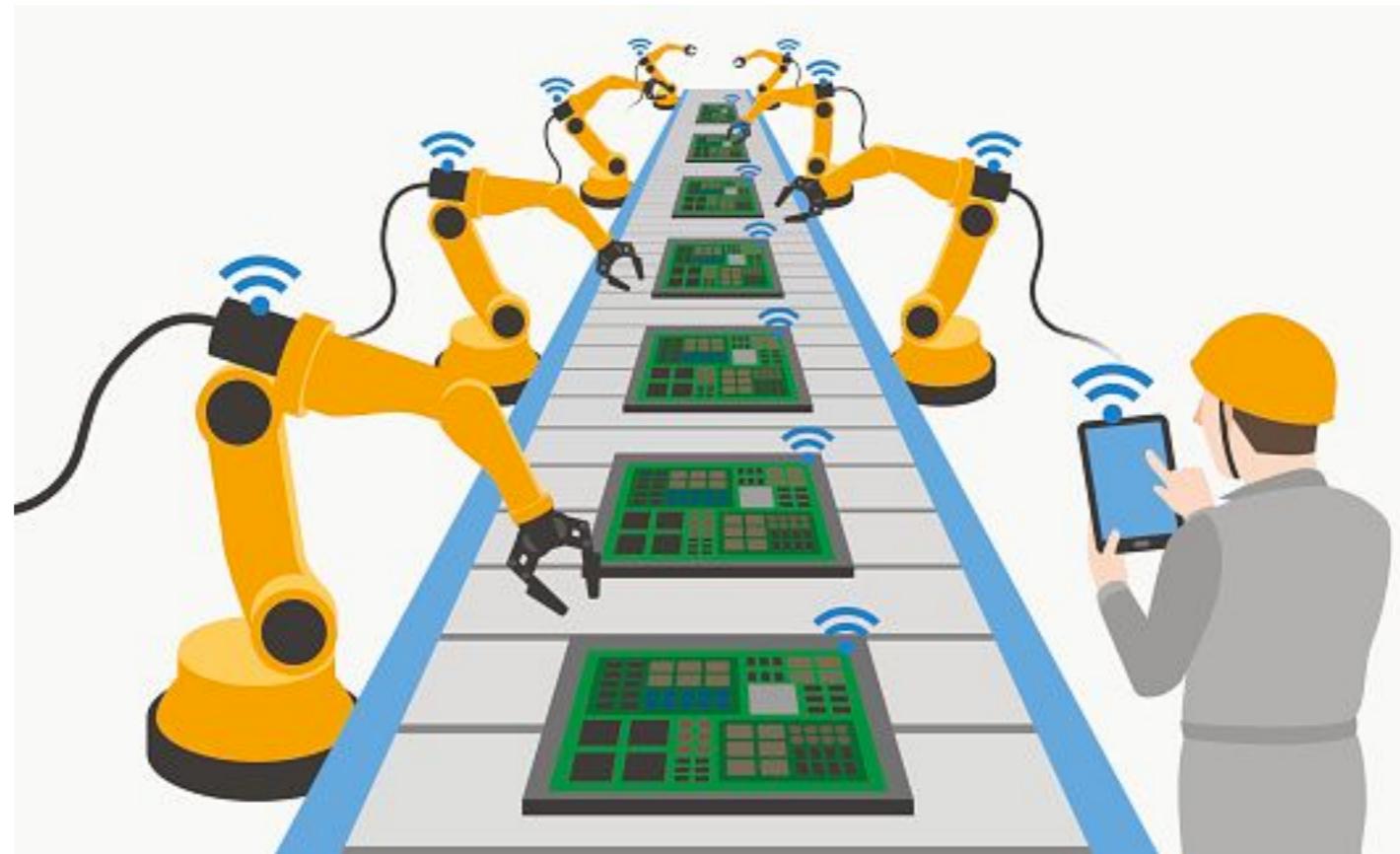
Maintain a single source repository

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



# Practice 2

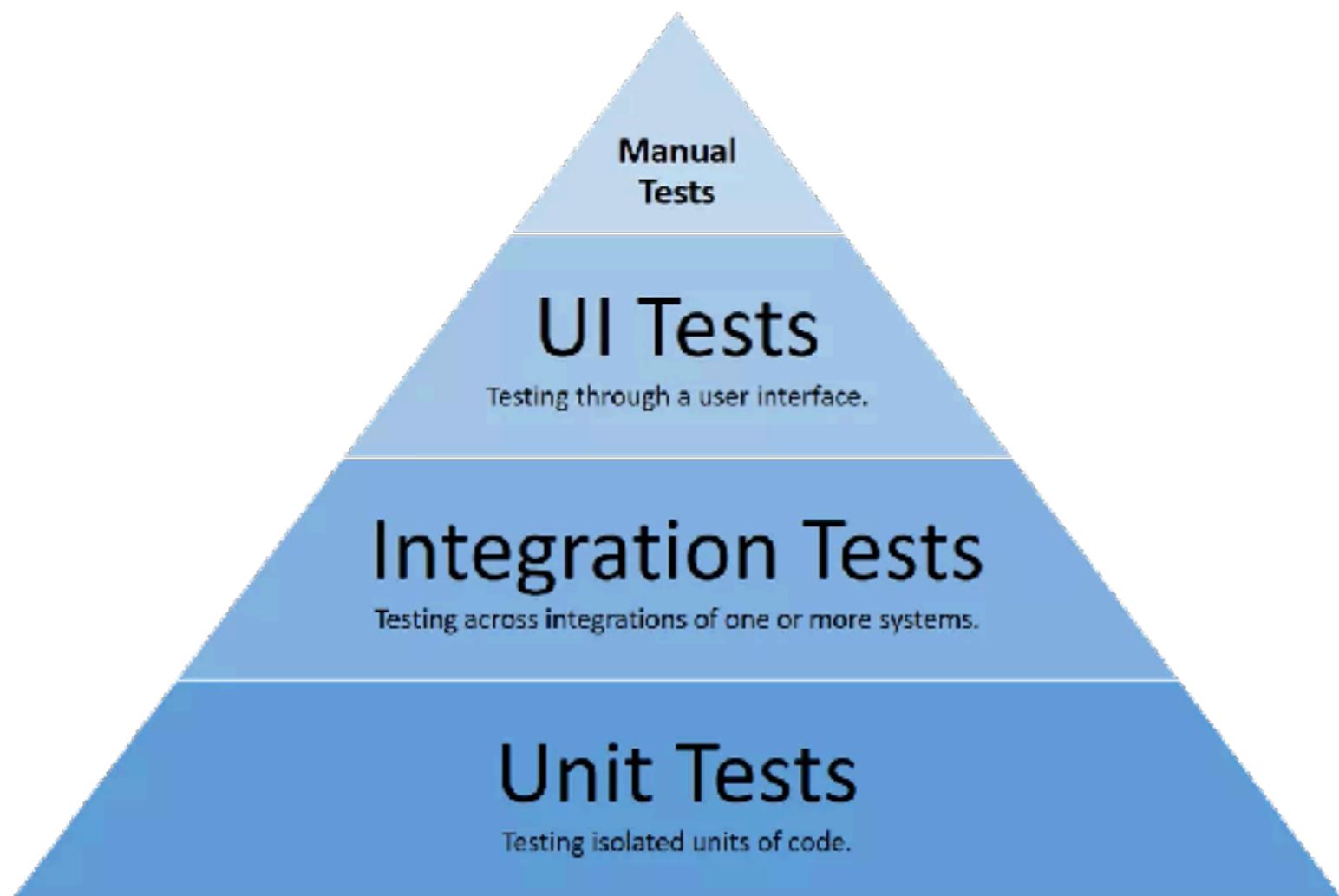
Automated the build  
Automated environment for builds



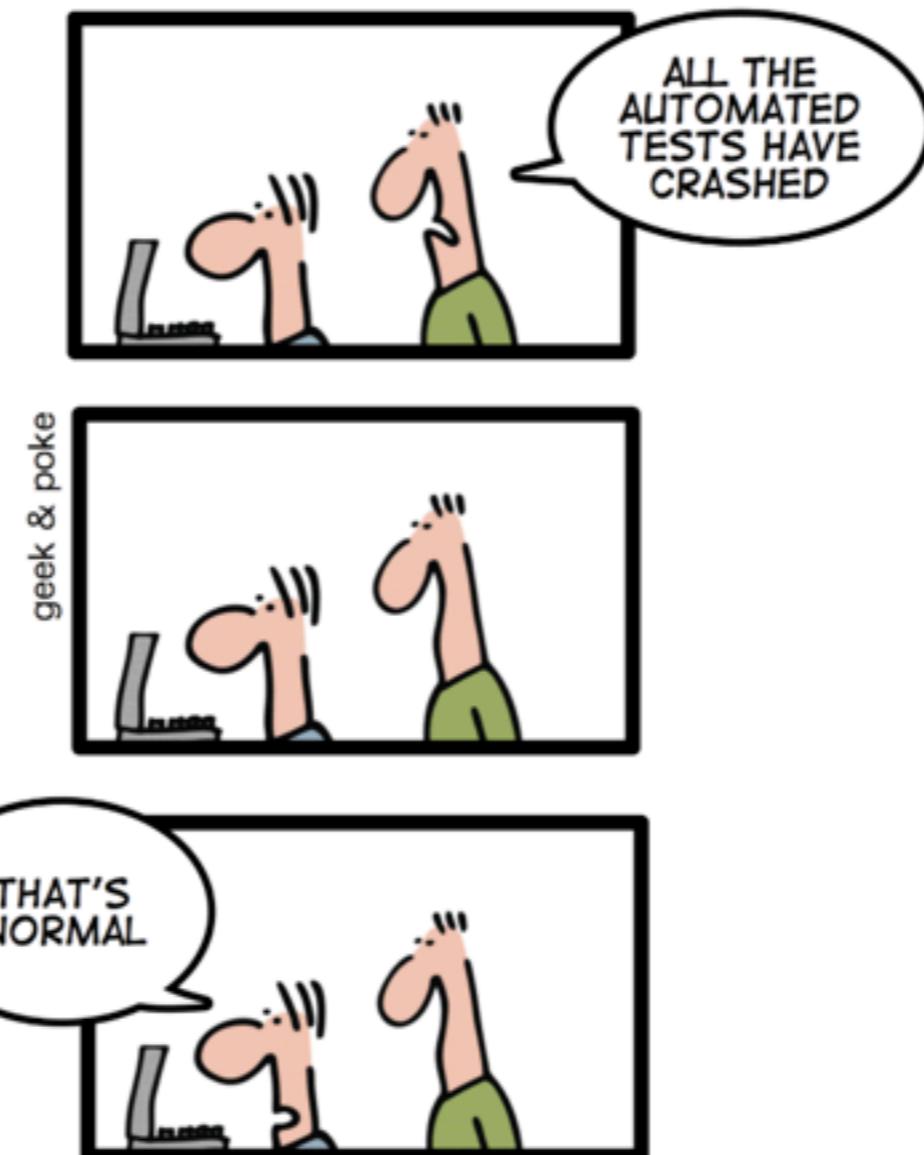
# Practice 3

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*

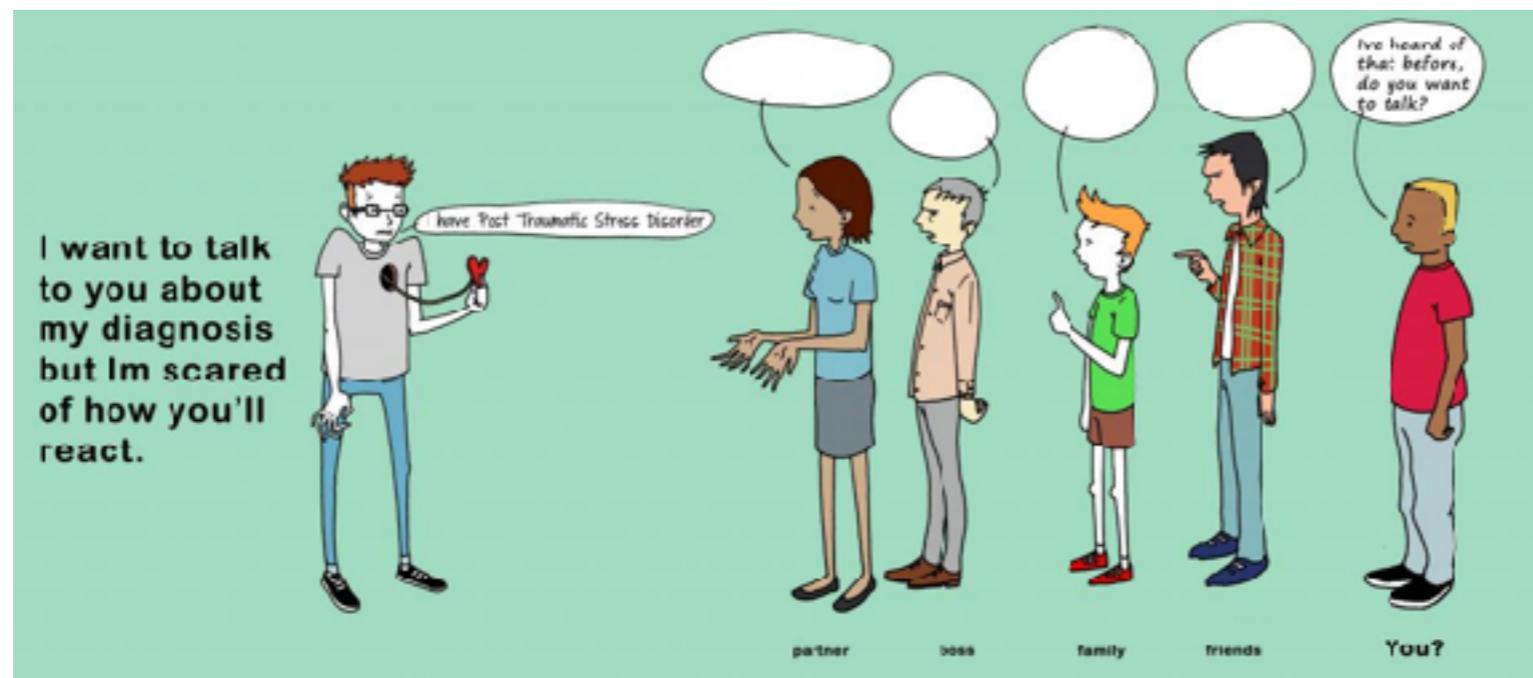


# Practice 4

**Everyone commits to the mainline everyday**

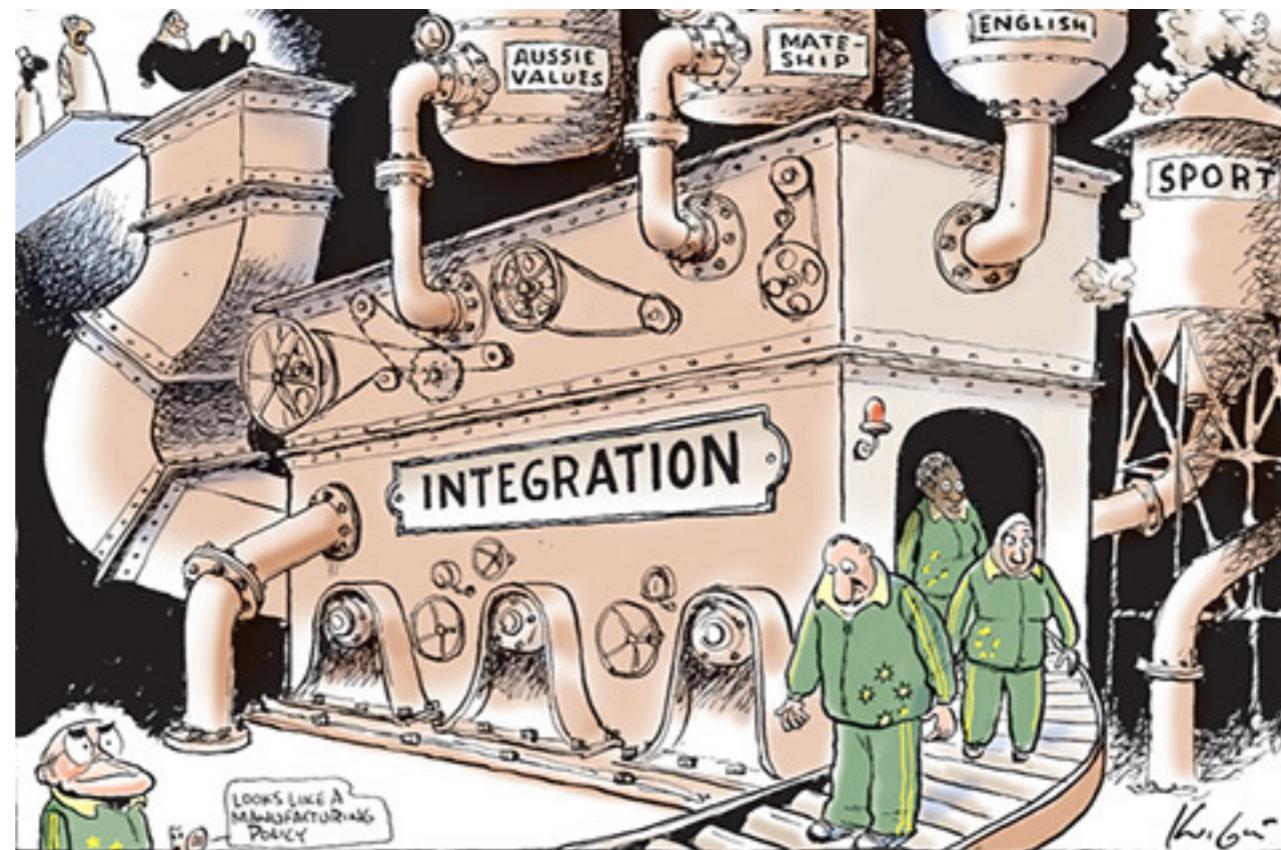
**Integration is about communication**

**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

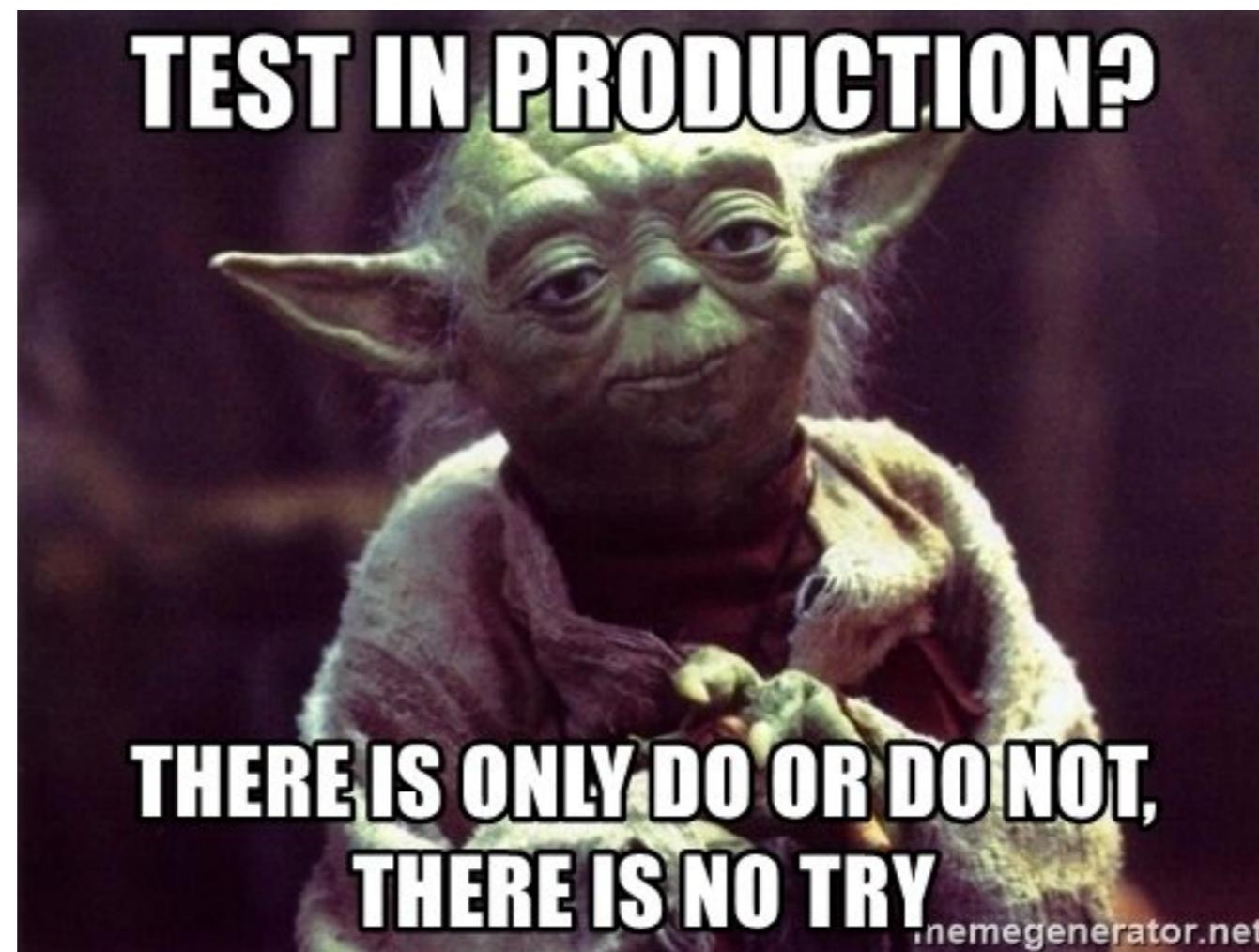
Keep the build **fast**

Continuous Integration is to provide rapid feedback



# Practice 8

Test in clone of the **Production** environment



# Practice 9

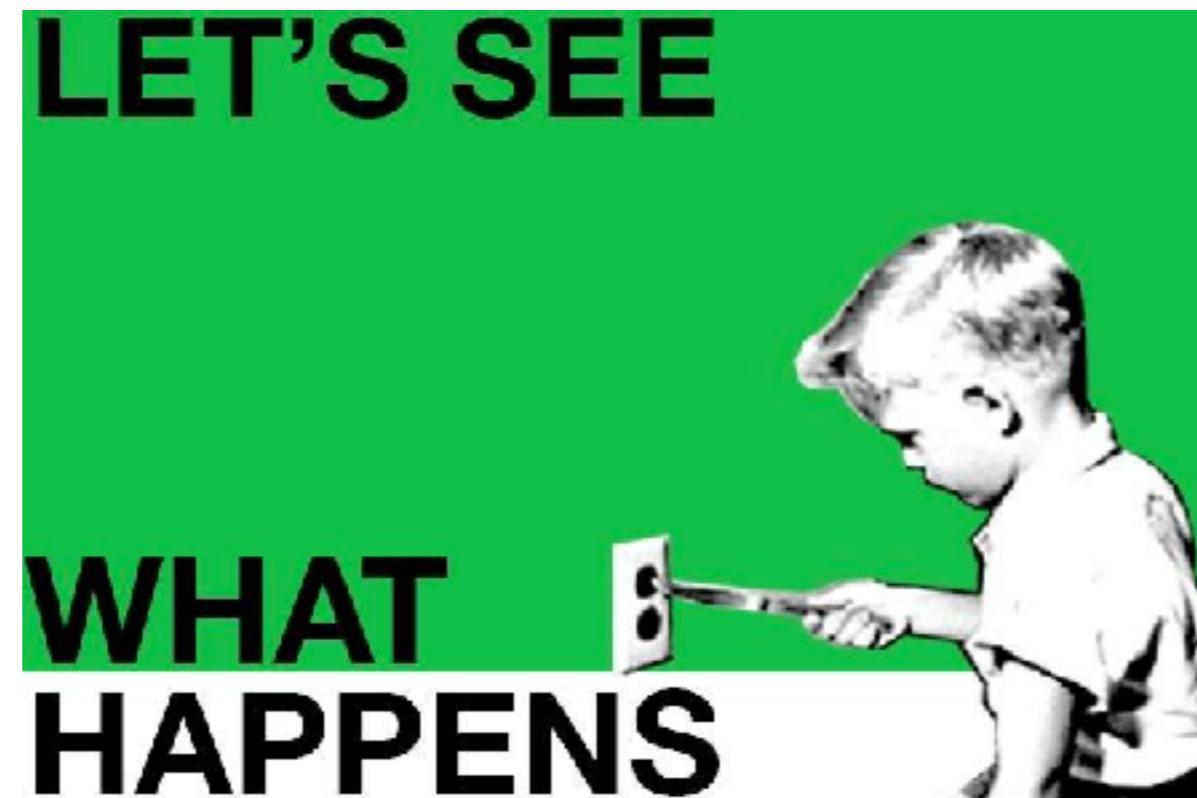
Make it easy for anyone to get  
the latest executable

Make sure well known place where people can find



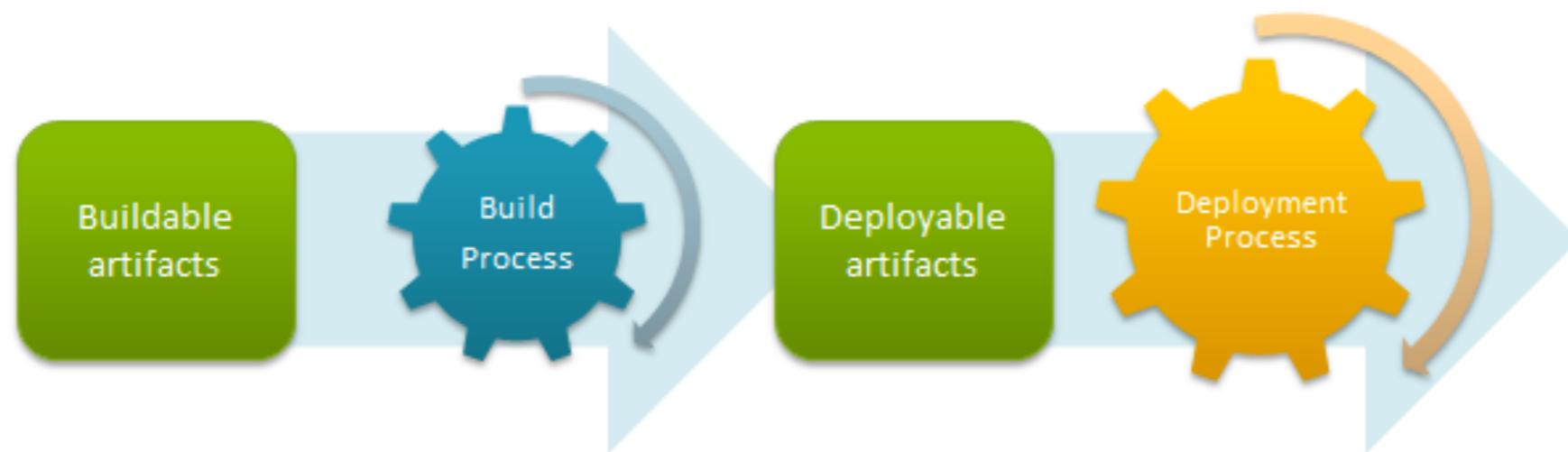
# Practice 10

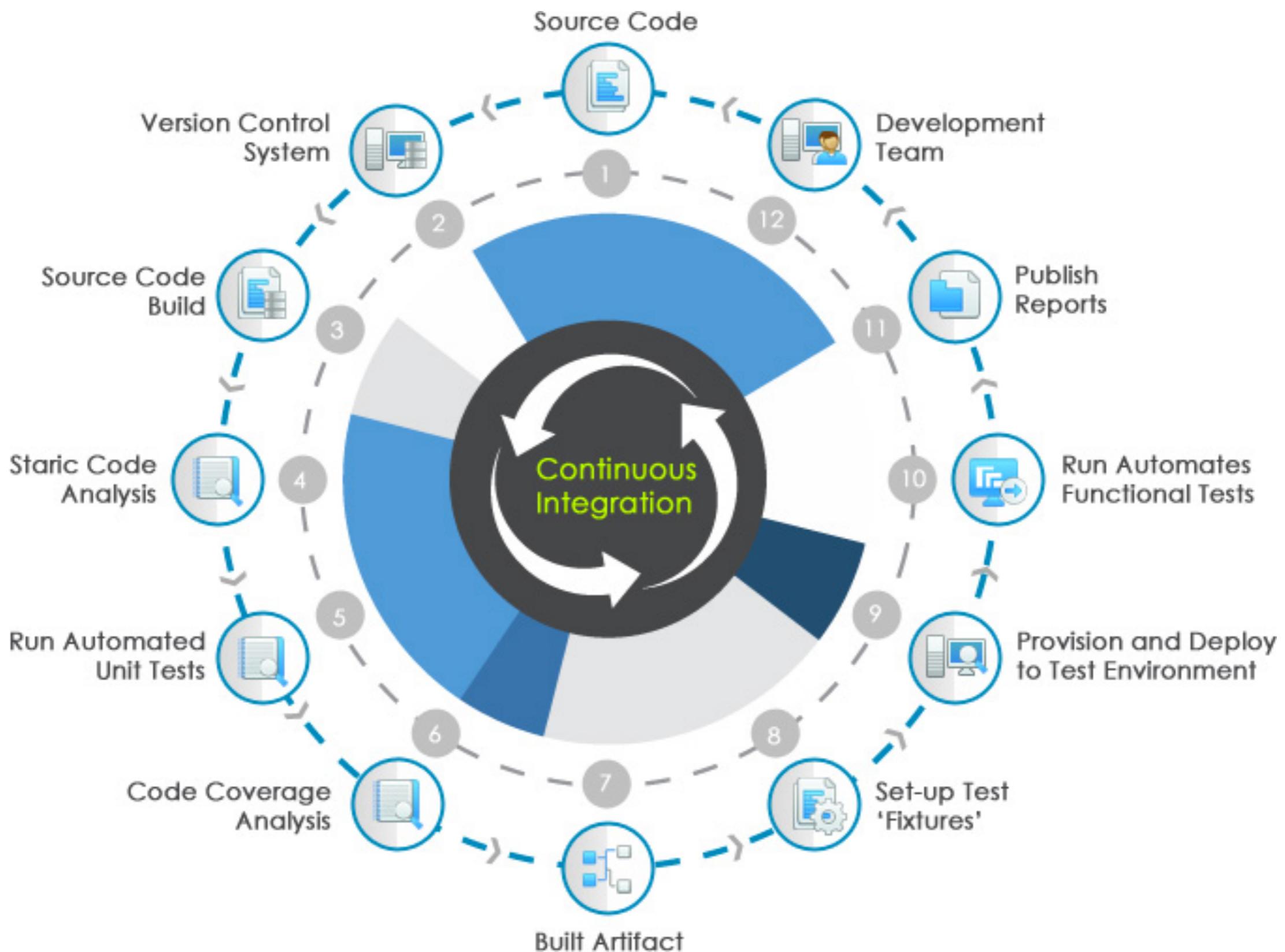
**Everyone** can see what's happening  
**Easier** to see the state of the system and changes  
Show the good information



# 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

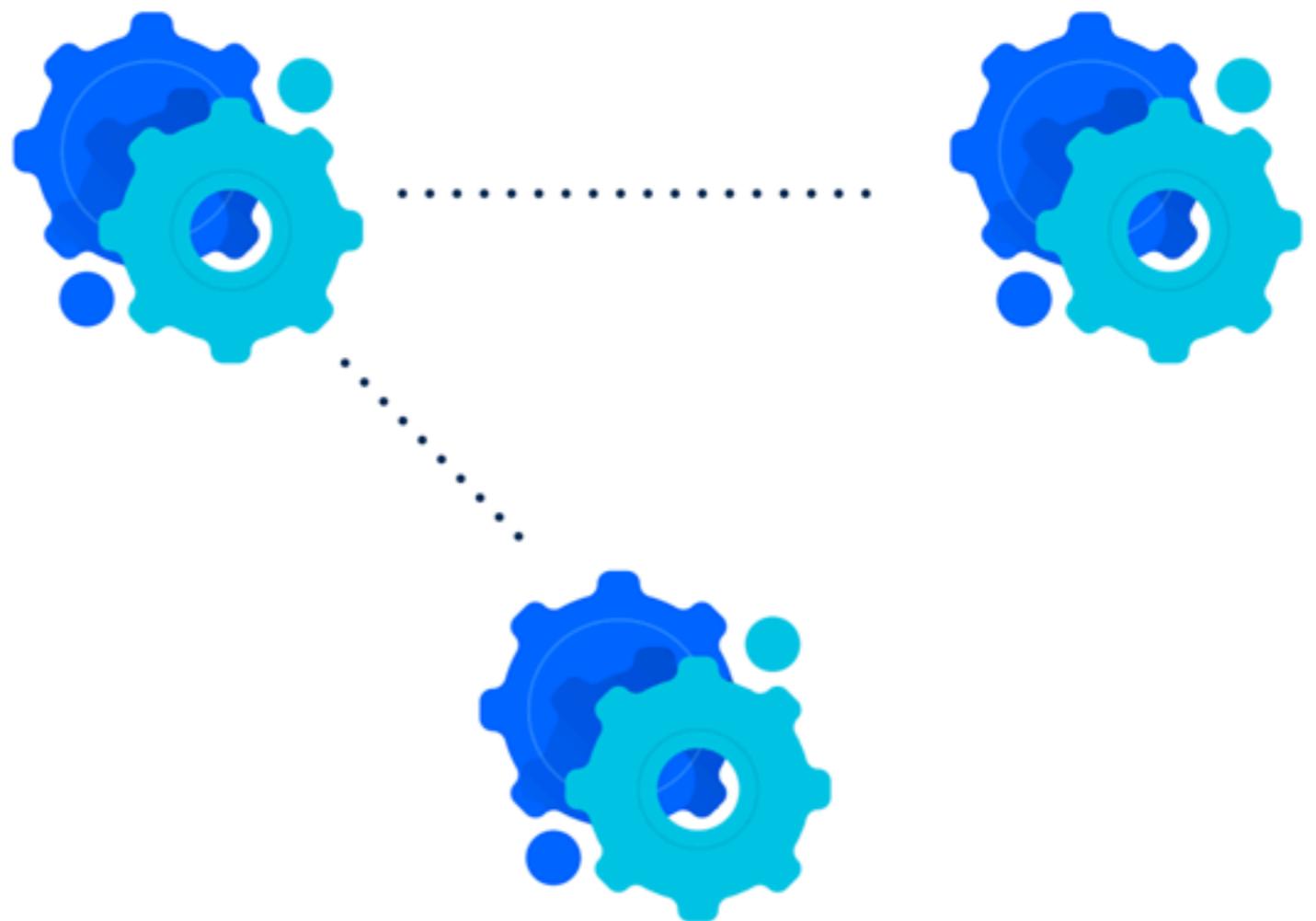
<https://trello.com/>



# Health check services

## Deep check .....

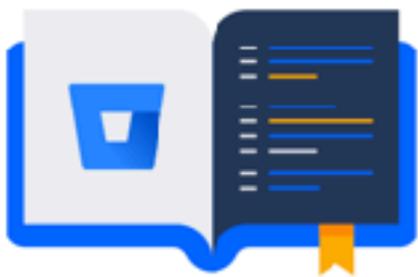
Quickly discover if a service fails to connect to a dependency



<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

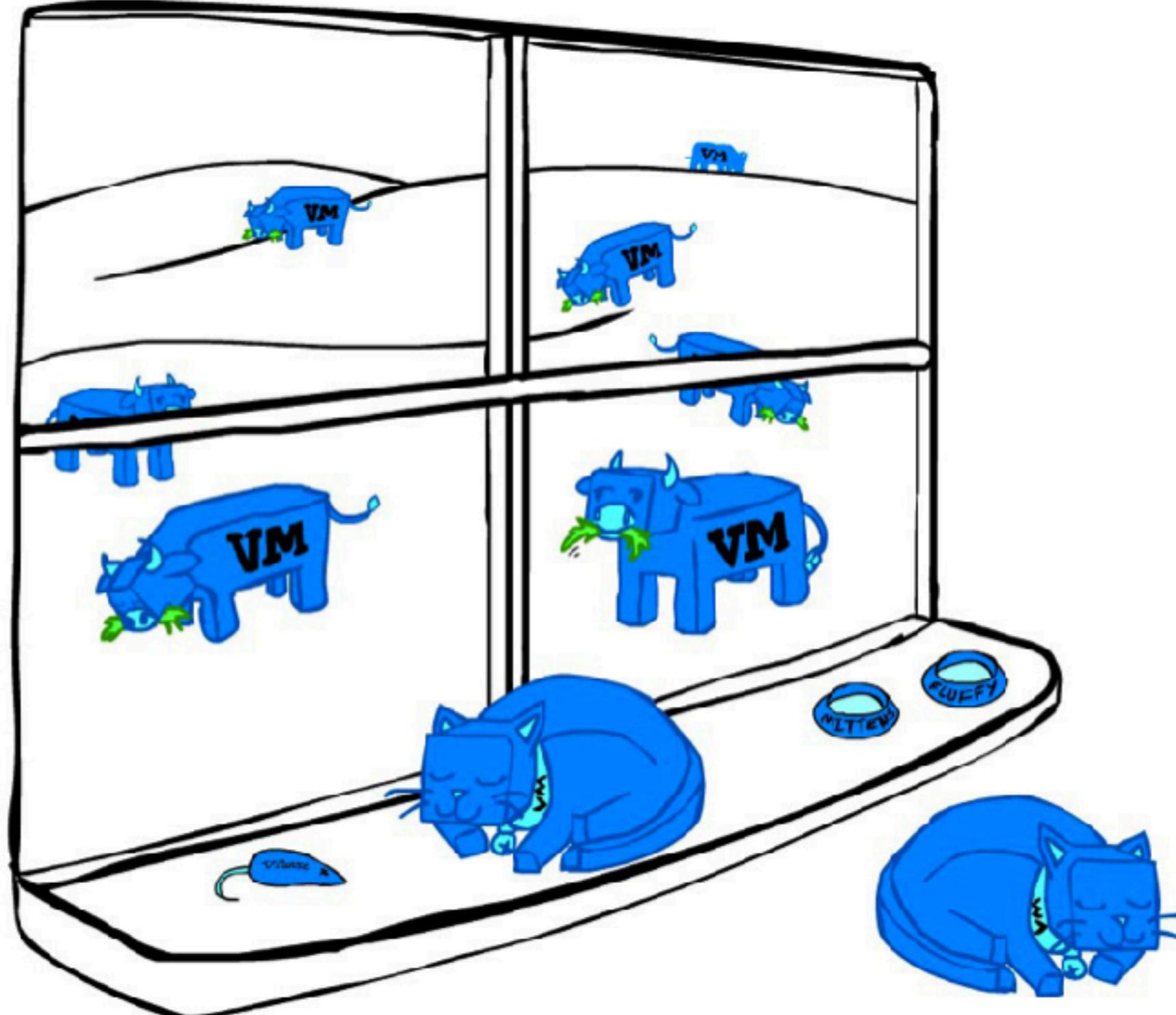
<https://trello.com/>



**“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



## Scale Out

- 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-science>

<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

**RESPONDING TO  
CHANGE**  
over following a plan

**INDIVIDUALS AND  
INTERACTIONS**  
over processes and tools

**WORKING  
SOFTWARE**  
over full documentation



# Agile principles

1 Satisfy the **customer**



Welcome **change**



Deliver **frequently**



4 Work **together**



5 Trust and **support**



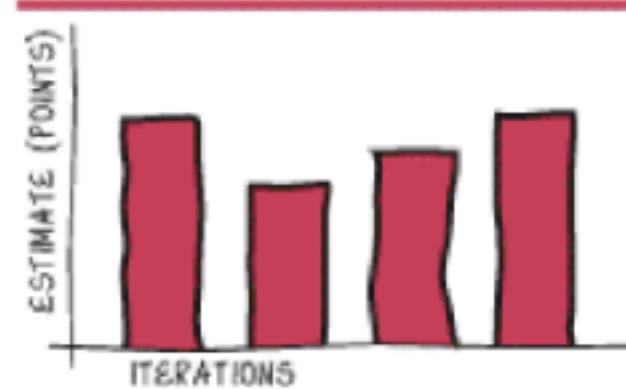
Face-to-face **conversation**



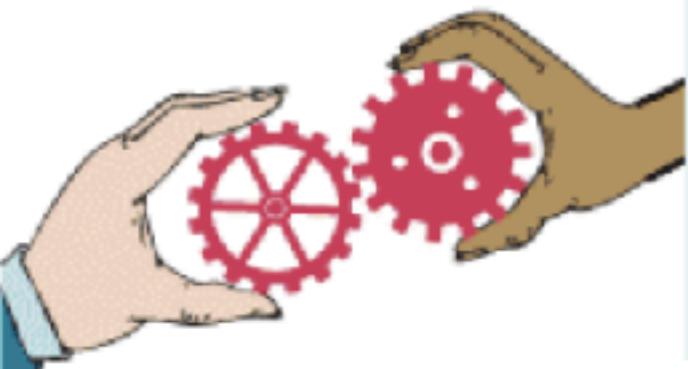
Working **software**



8 Sustainable **development**



9 Technical **Excellence**



10 Maintain **simplicity**



11 Self-organizing **teams**

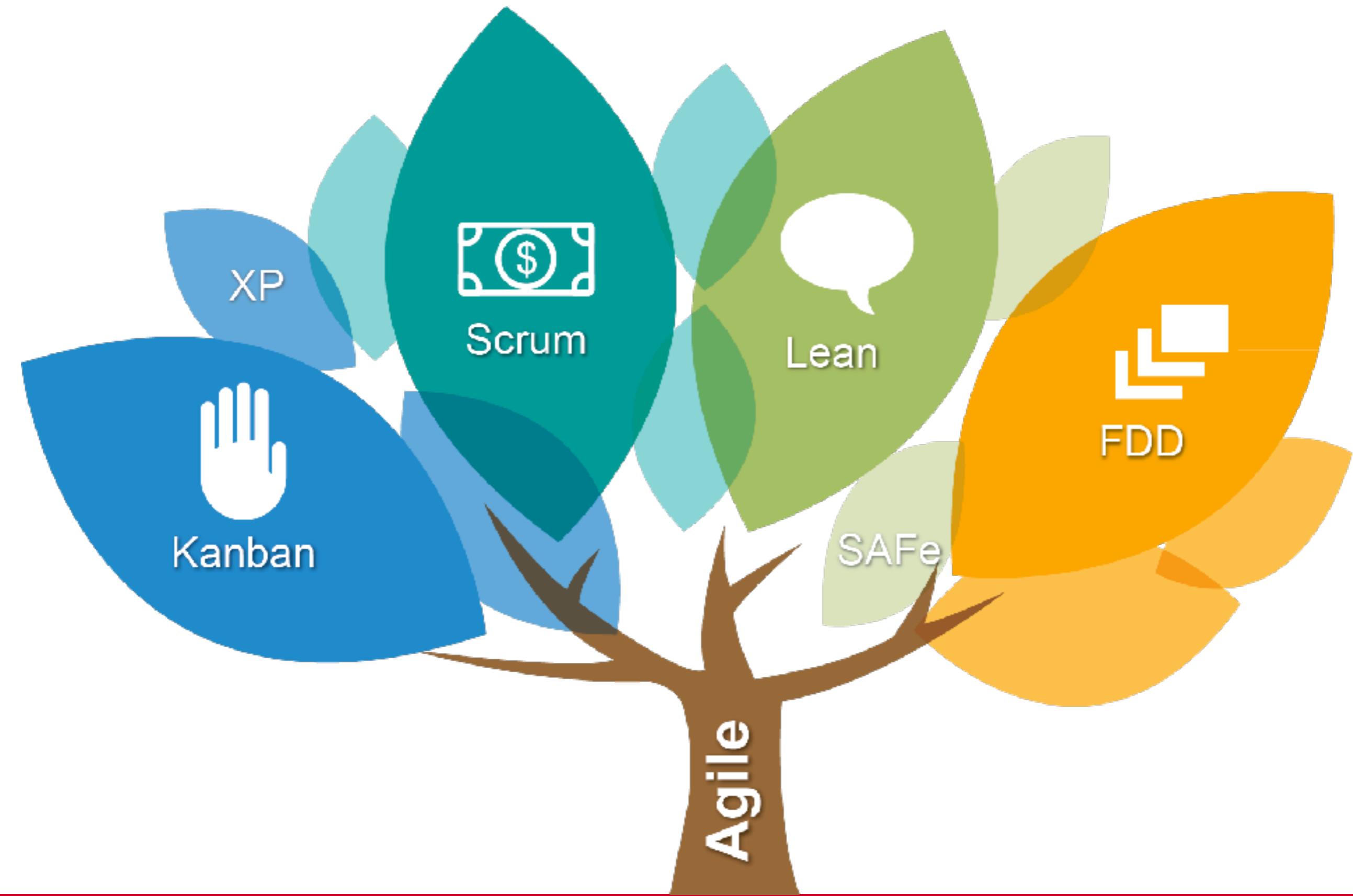


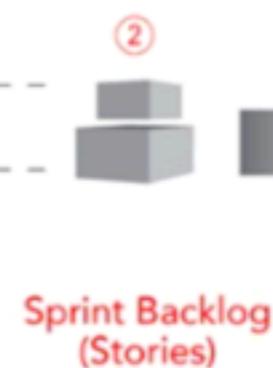
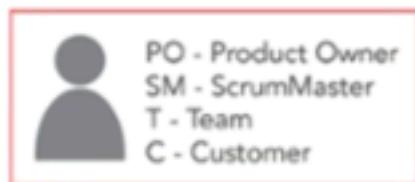
12 Reflect and **adjust**



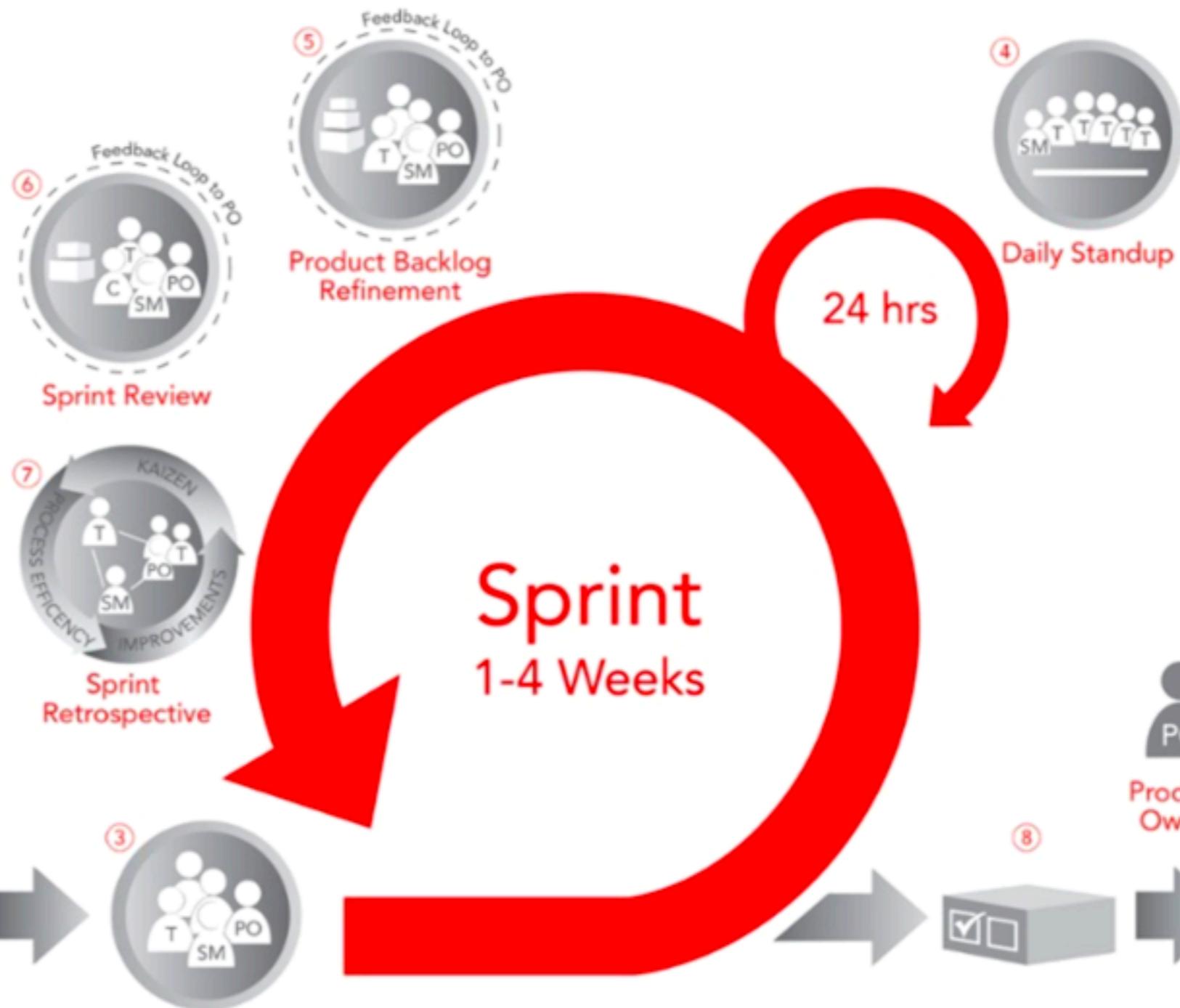
Origin by <https://www.knowledgetrain.co.uk>, modified by Jacky Shen





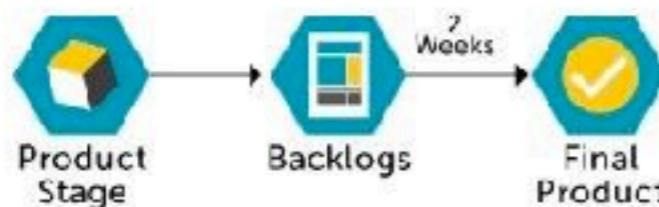


Sprint Planning



## AGILE DEVELOPMENT

### Daily Standup

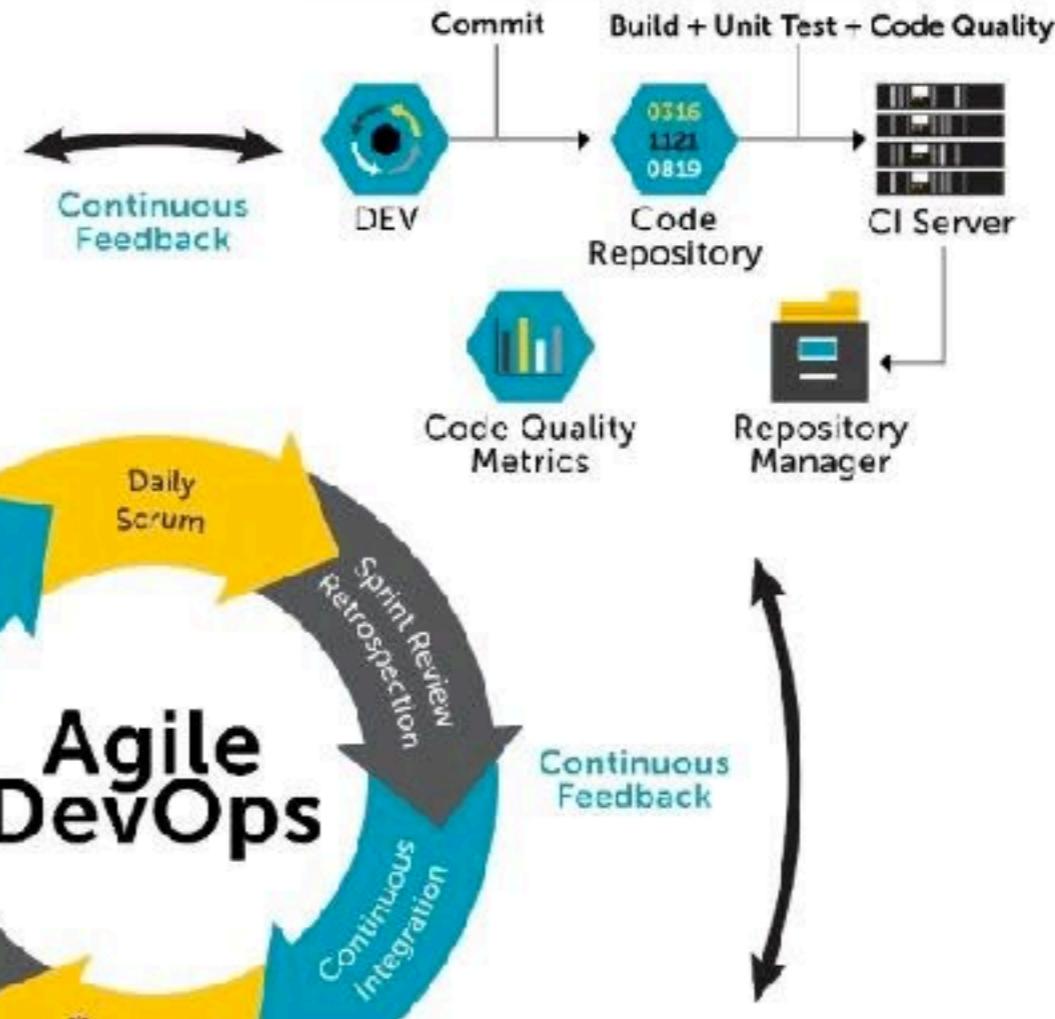


Process Flow Chart



User Inputs

## CONTINUOUS INTEGRATION



Commit

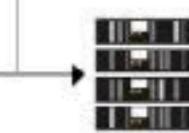
Build + Unit Test - Code Quality



DEV



Code Repository



CI Server

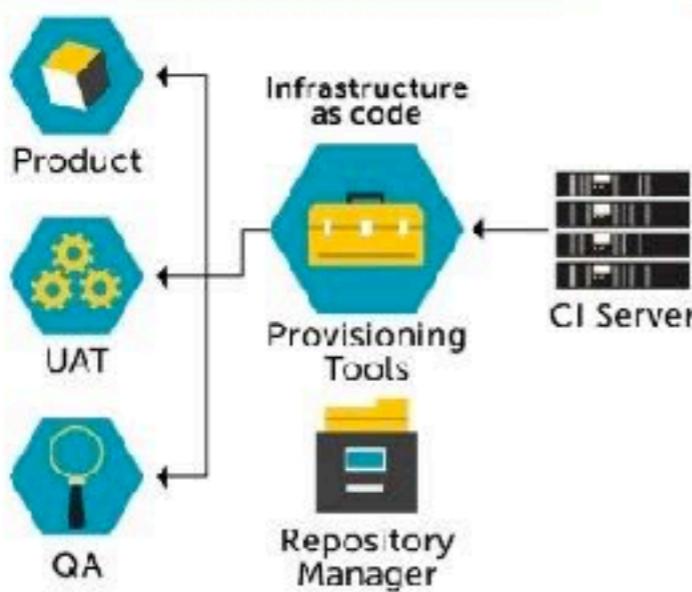


Code Quality Metrics



Repository Manager

## CONTINUOUS DELIVERY



Infrastructure as code

Product

UAT

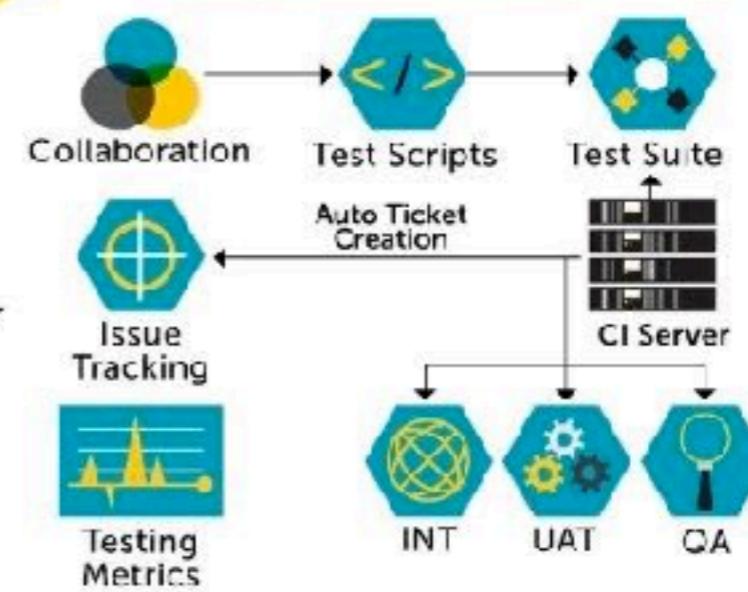
QA

Provisioning Tools

CI Server

Repository Manager

## CONTINUOUS TESTING



Collaboration

Test Scripts

Test Suite

CI Server

Issue Tracking

Auto Ticket Creation

CI Server

Testing Metrics

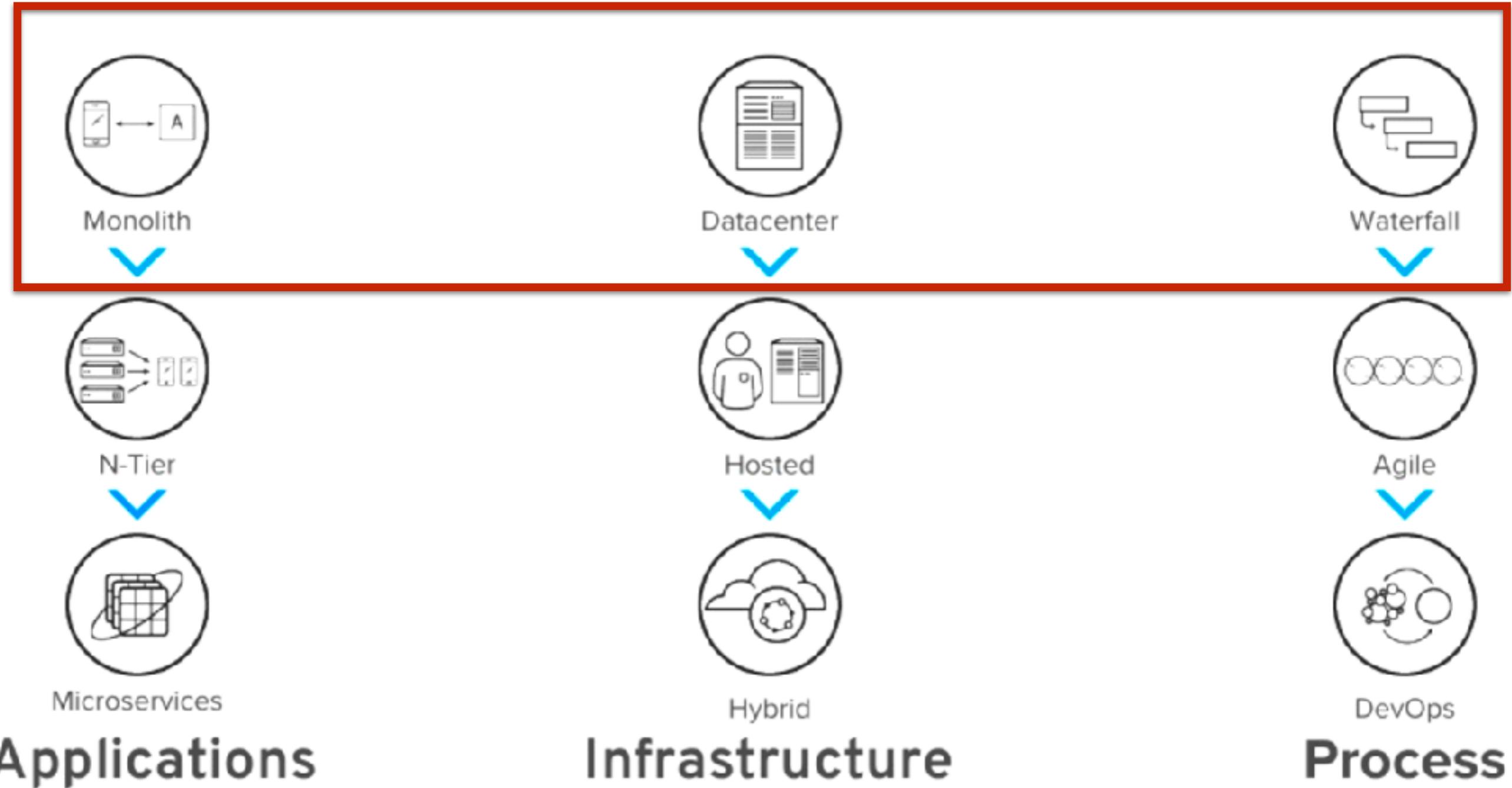
INT

UAT

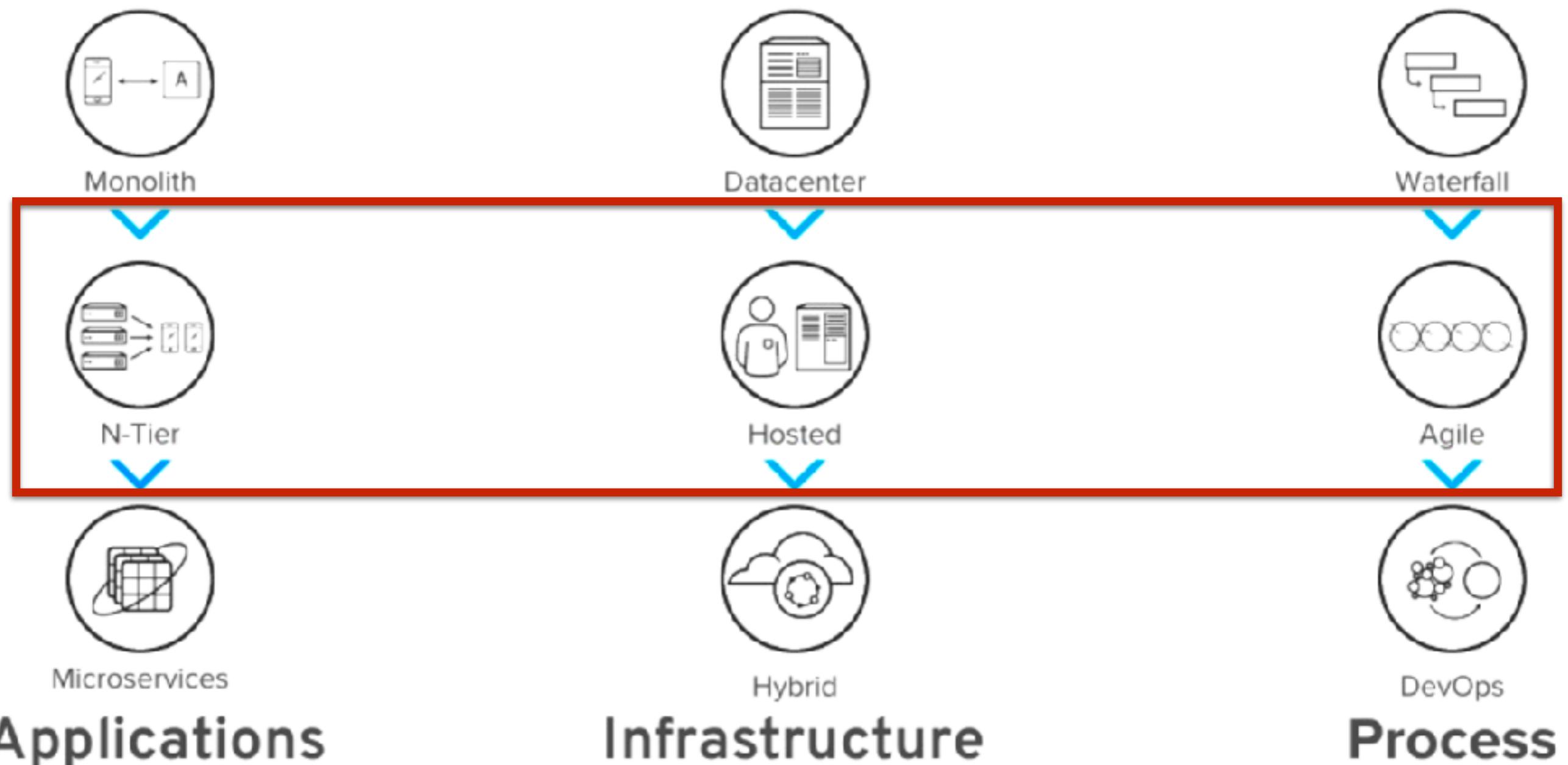
QA



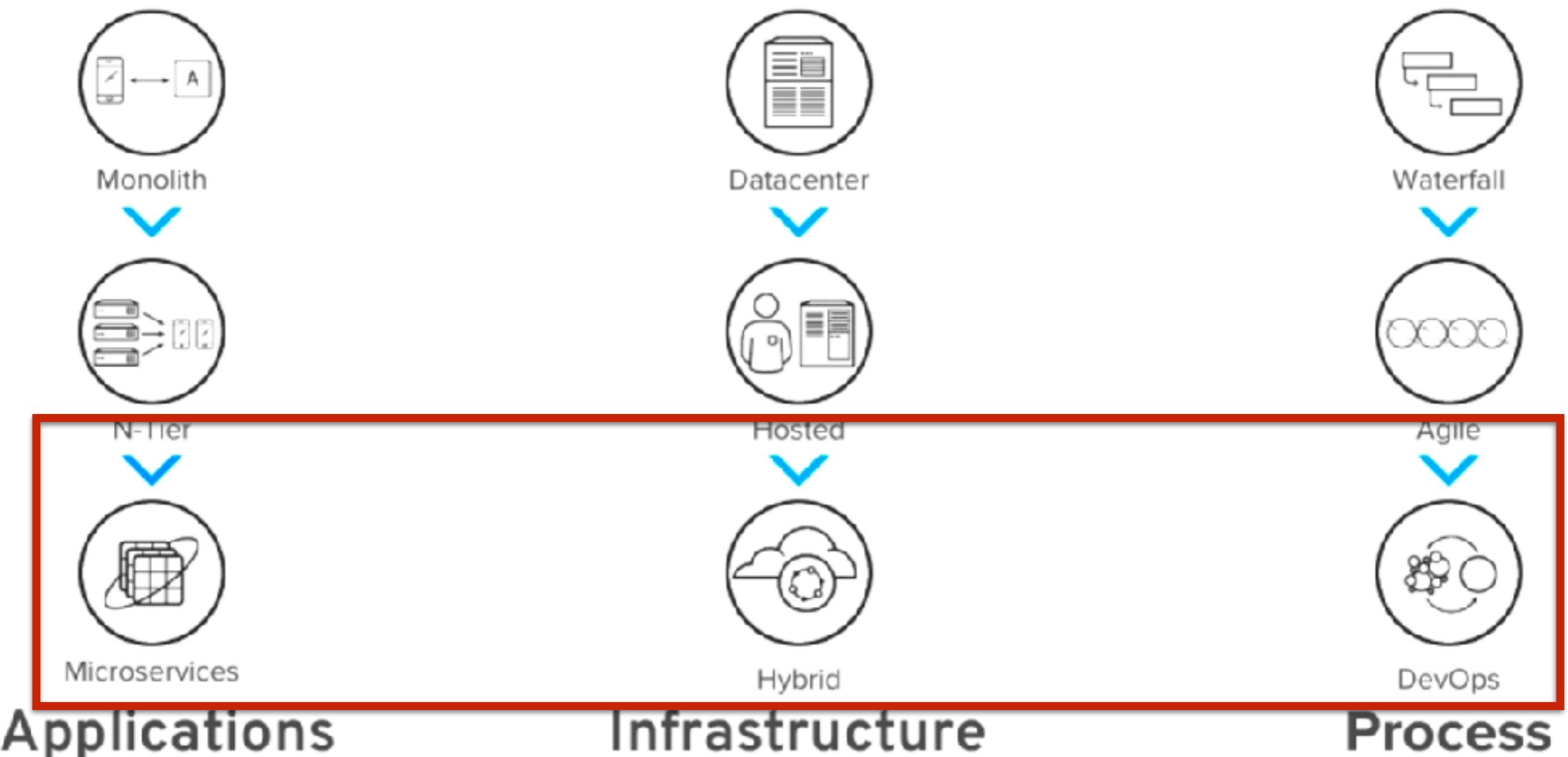
# Continuous Improvement



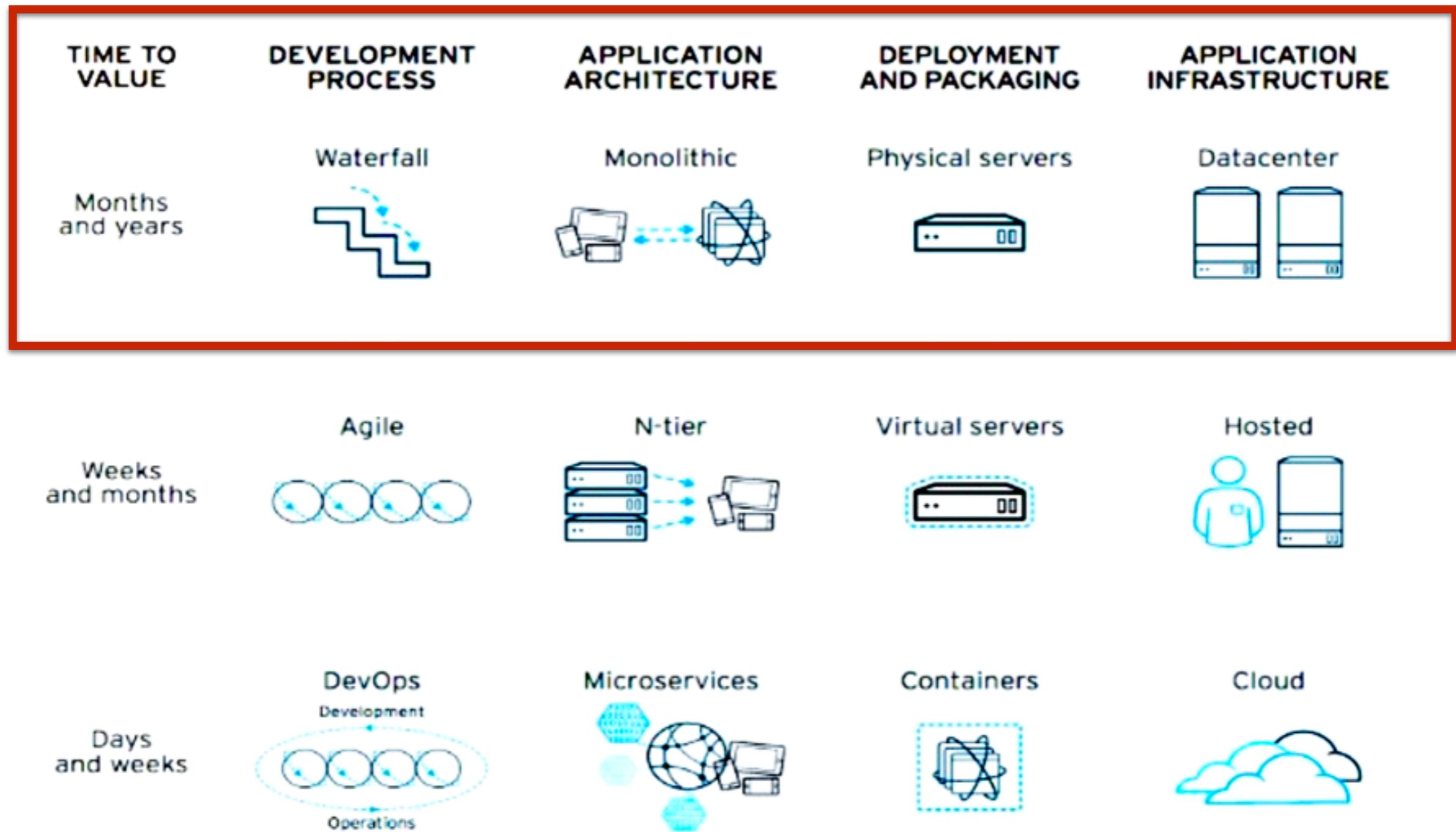
# Continuous Improvement



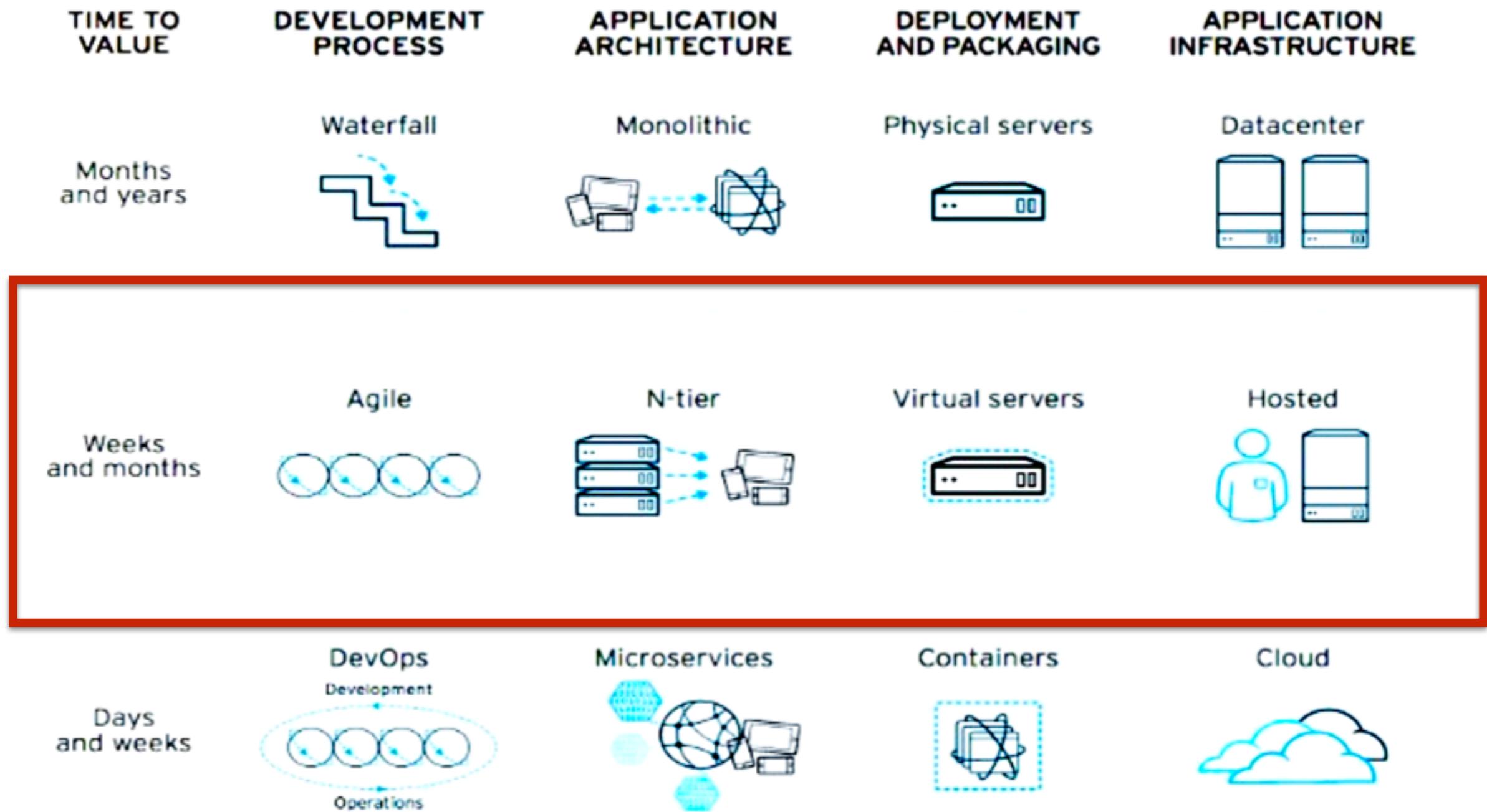
# Continuous Improvement



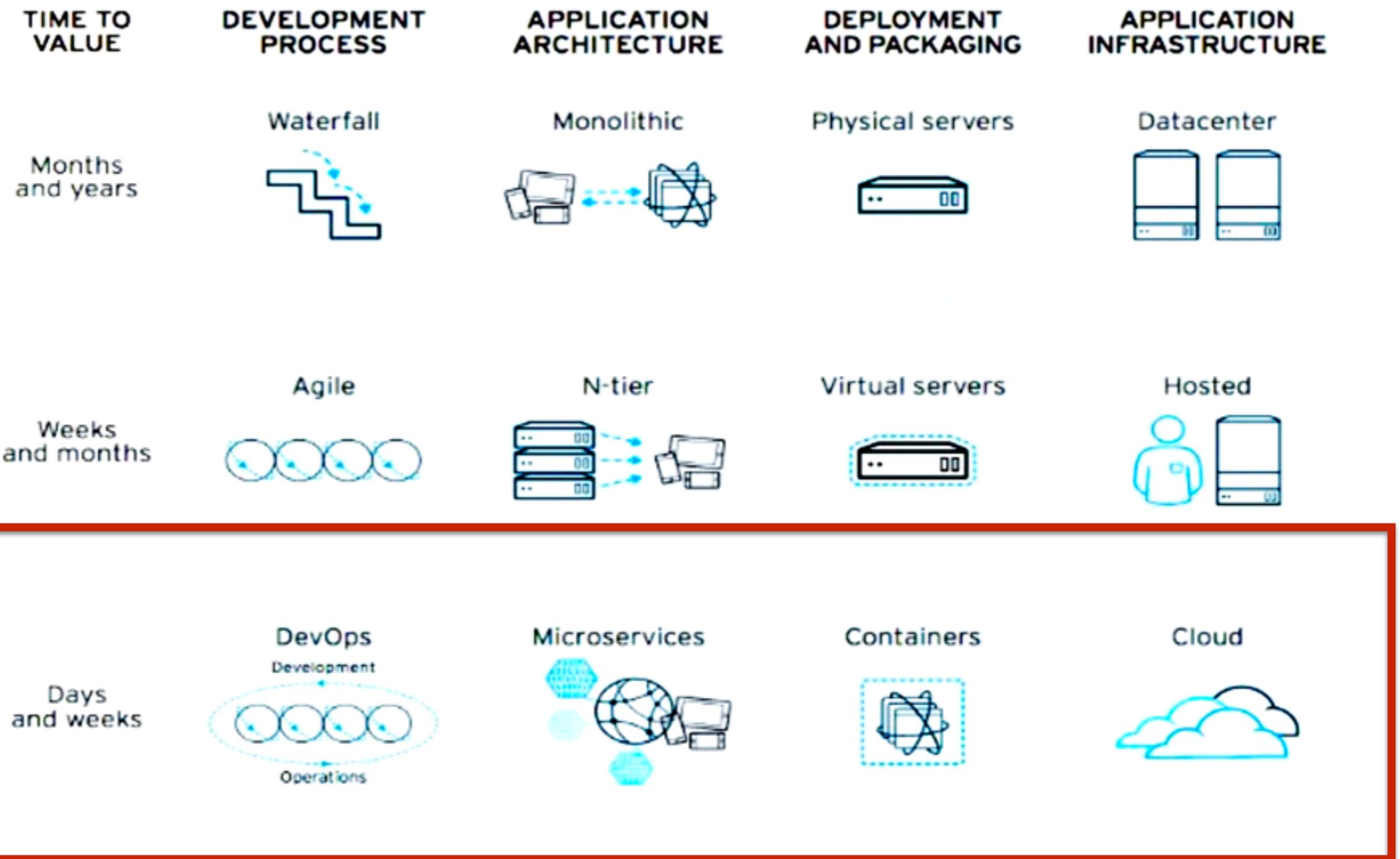
# Improve Time to Value



# Improve Time to Value



# Improve Time to Value



# 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



# Are you too busy to improve?



Thank you  
Q/A



