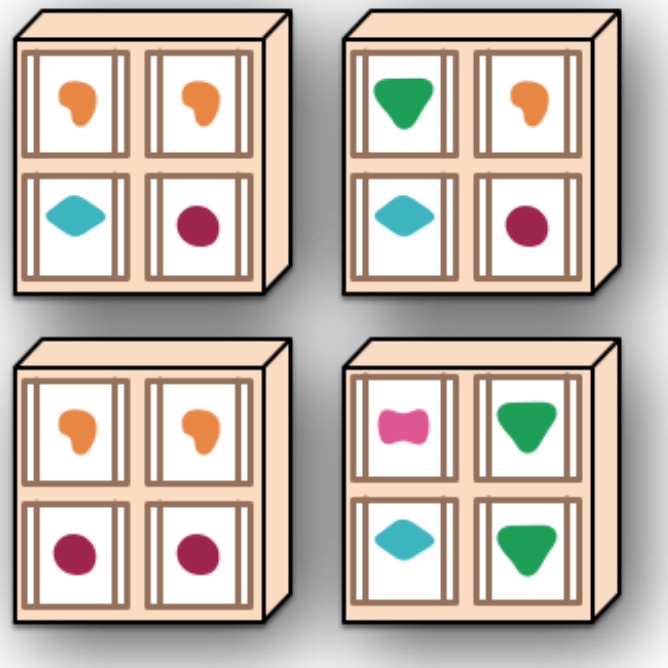
ThoughtWorks®

NEAL FORD

Director / Software Architect / Meme Wrangler



Building Microservice Architectures

@neal4d

nealford.com

Agenda

what problem

characteristics

engineering

Service-oriented Architecture

business services

BS BS BS BS BS BS

abstract enterprise-level coarse-grained services owned and defined by business users

no implementation - only name, input, and output data represented as wsdl, bpel, xml, etc.

ExecuteTrade

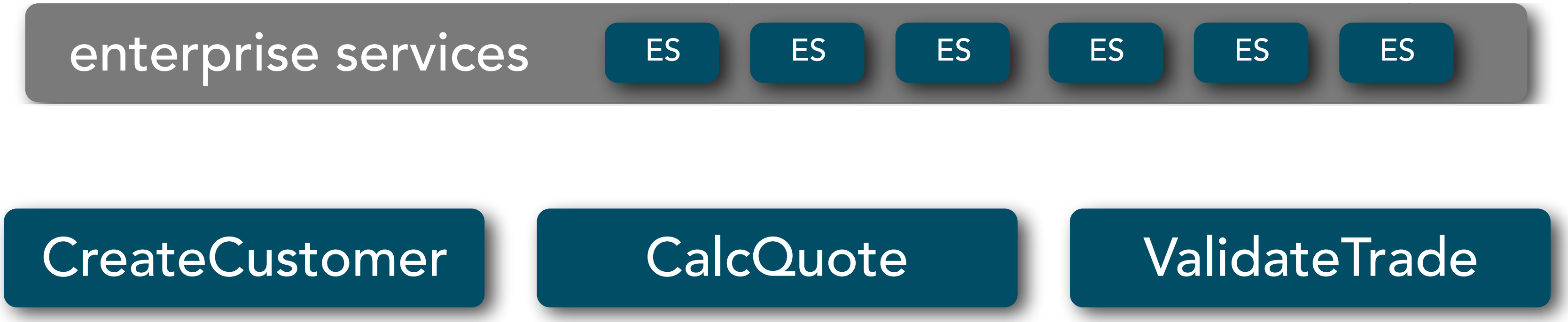
PlaceOrder

ProcessClaim

Service-oriented Architecture

concrete enterprise-level coarse-grained services owned by shared services teams

custom or vendor implementations that are one-to-one or one-to-many relationship with business services



enterprise services ES ES ES ES ES ES

CreateCustomer CalcQuote ValidateTrade

Service-oriented Architecture

concrete application-level fine-grained services
owned by application teams

bound to a specific application context

AddDriver

UpdateAddress

CalcSalesTax

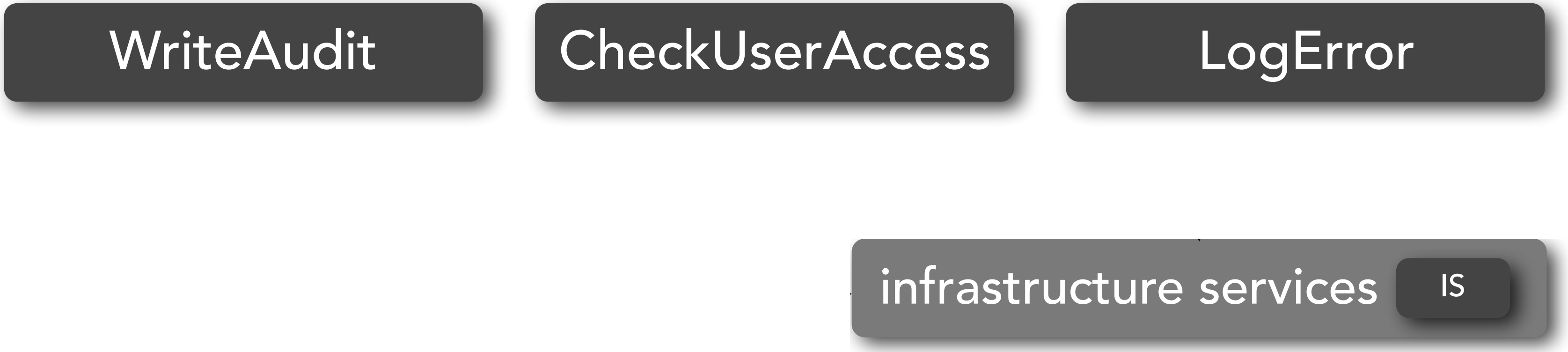
application services

AS

Service-oriented Architecture

concrete enterprise-level fine-grained services owned by infrastructure or shared services teams

implements non-business functionality to support both enterprise and business services



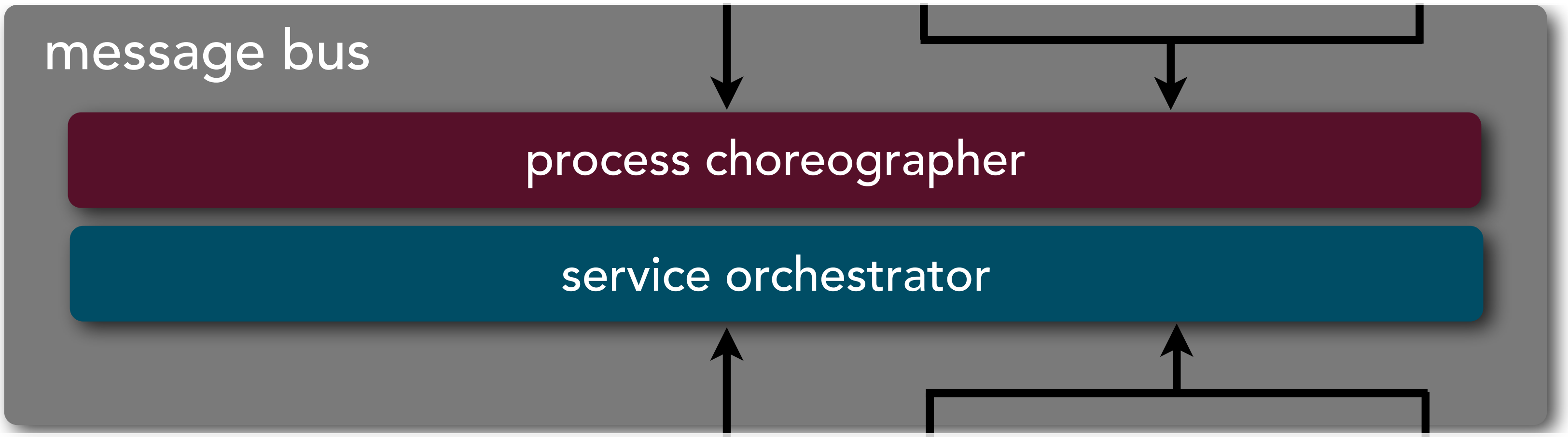
WriteAudit

CheckUserAccess

LogError

infrastructure services IS

Service-oriented Architecture



mediation and routing

process choreography

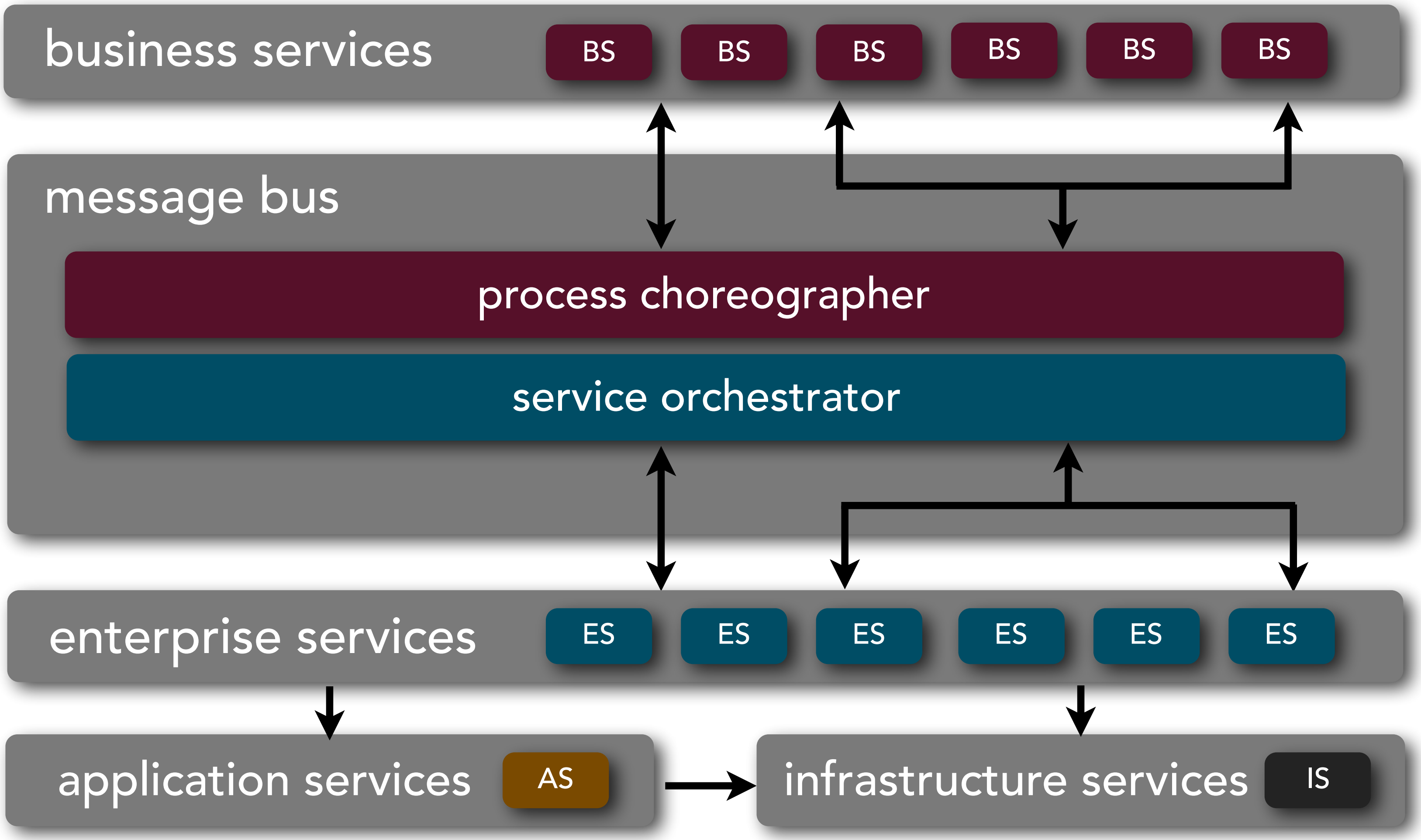
service orchestration

message enhancement

message transformation

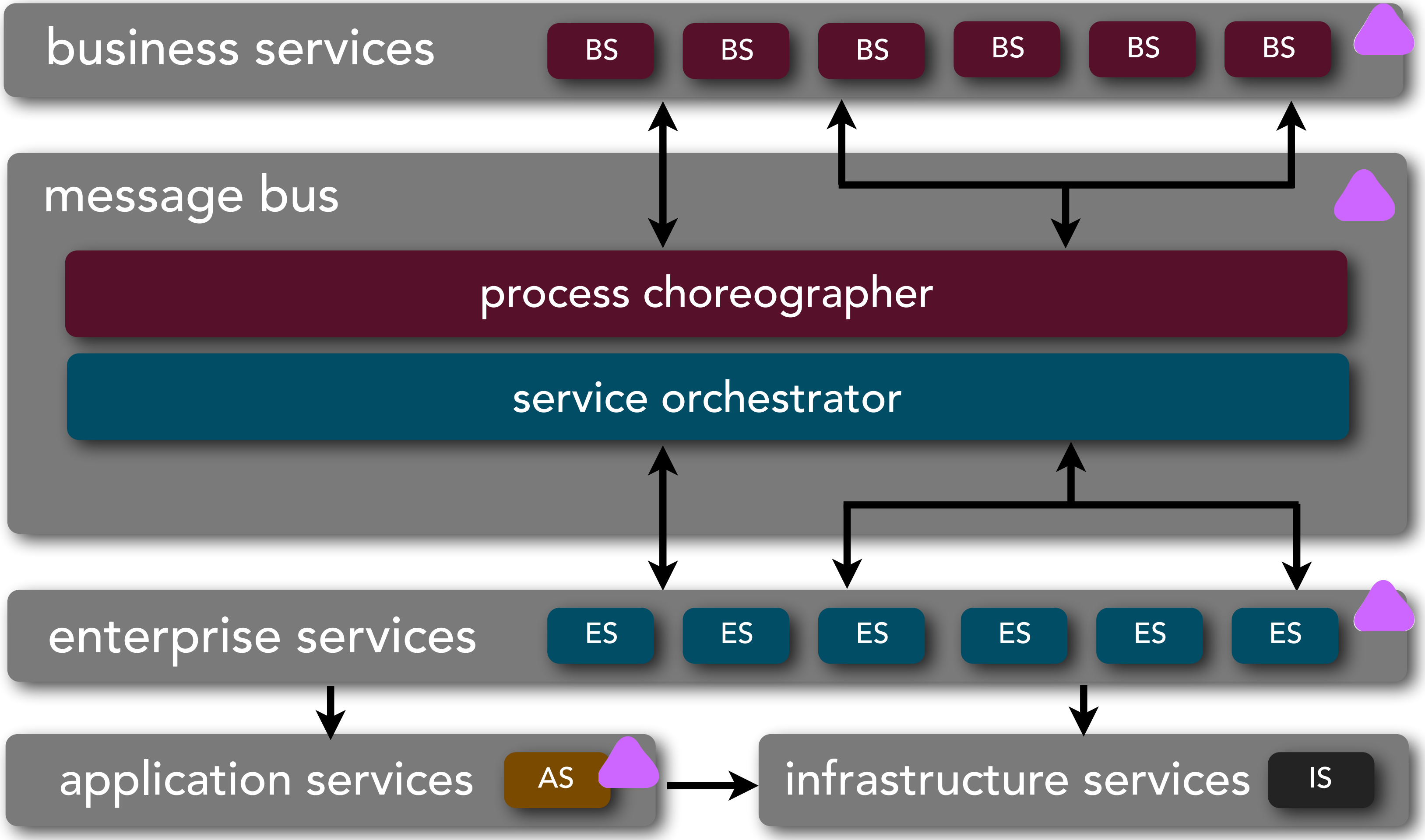
protocol transformation

Service-oriented Architecture



- maximize reuse
- maximize canonicality

Service-oriented Architecture



☒ incremental change

☒ operationally complex

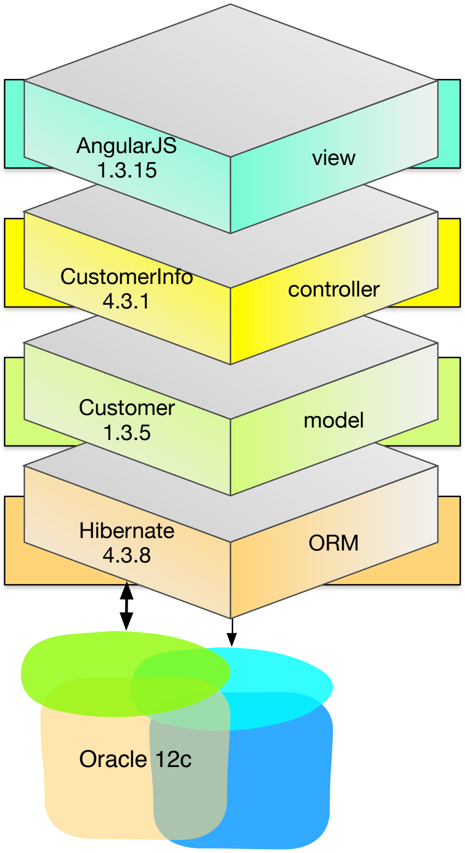
Yesterday's best practice is tomorrow's anti-pattern.

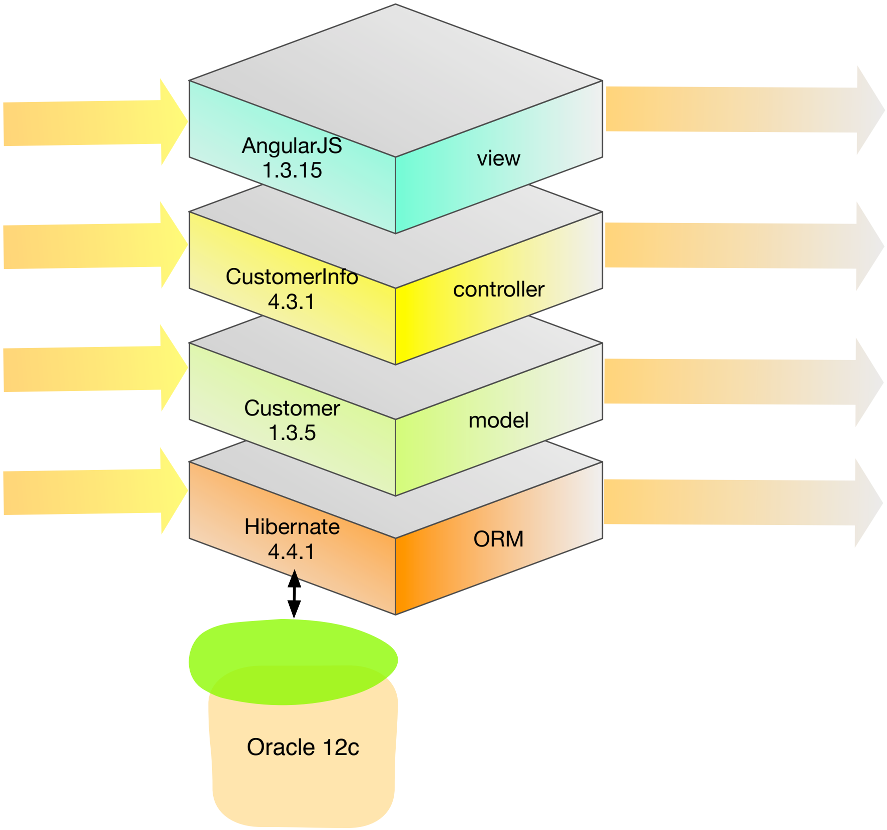
We inadvertently build architectures to solve outdated problems.

Architecture is abstract until operationalized.

nealford.com/memeagora/2015/03/30/architecture_is_abstract_until_operationalized.html

Architecture is abstract until operationalized.





2D 3D 4D

nealford.com/memeagora/2015/03/30/architecture_is_abstract_until_operationalized.html

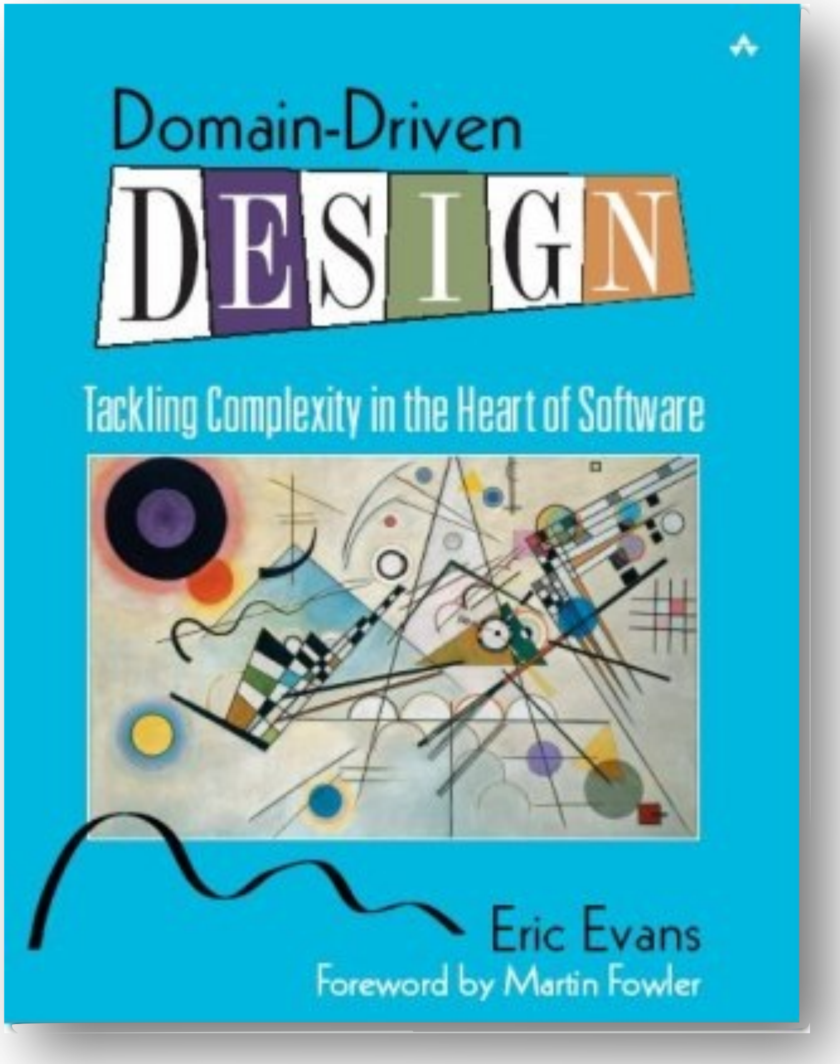
Agenda

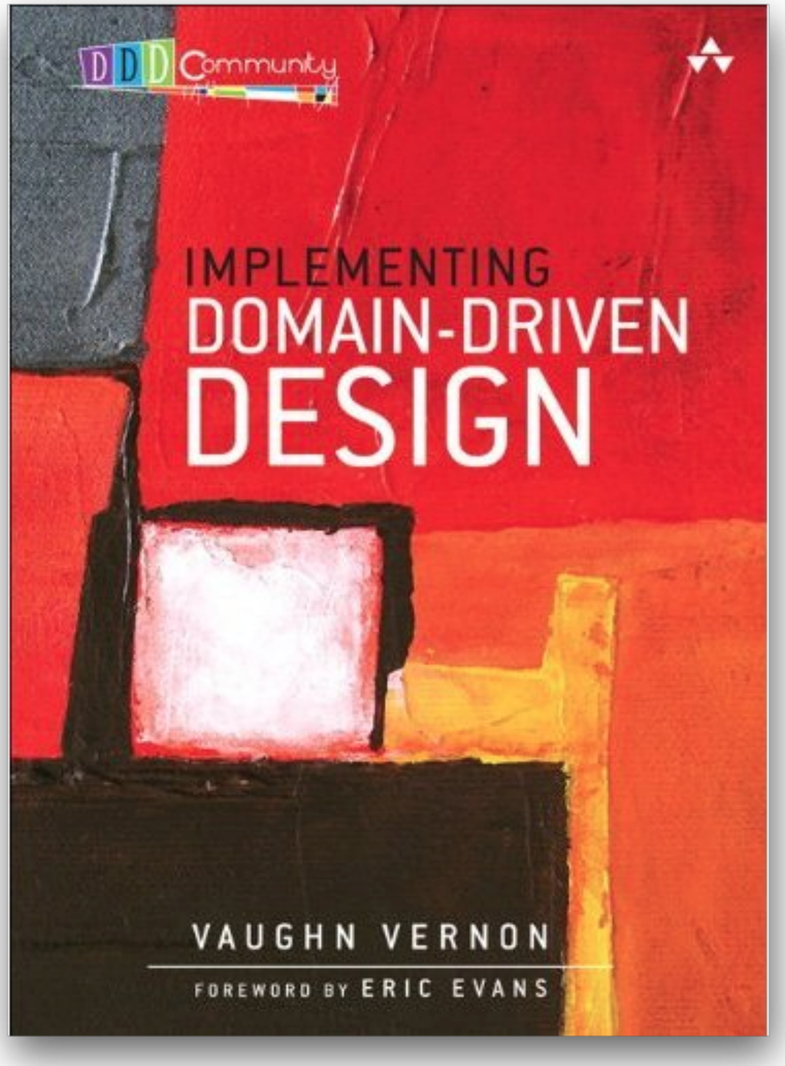
what problem

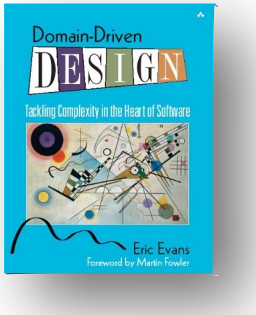
characteristics

engineering

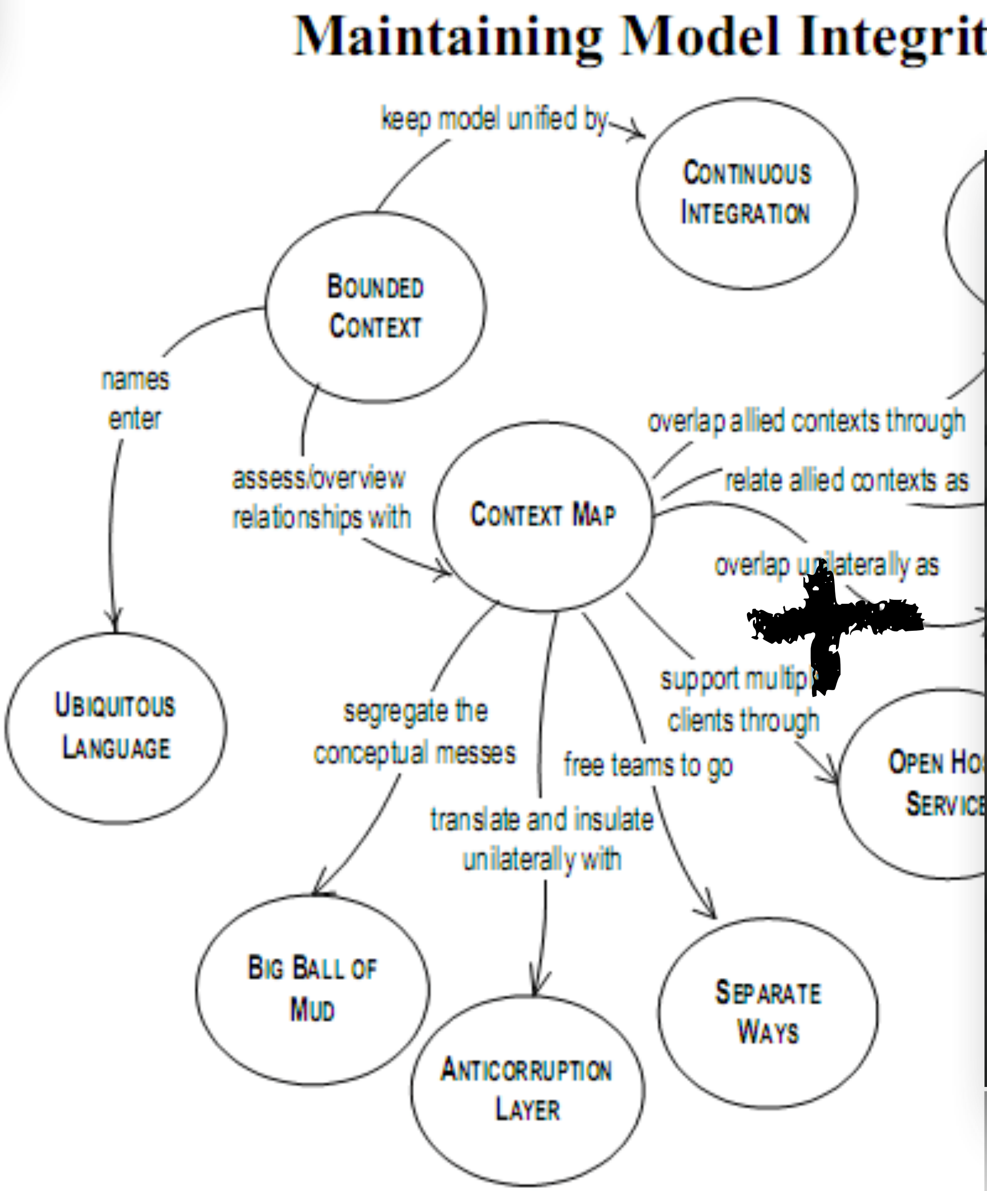
Domain Driven Design

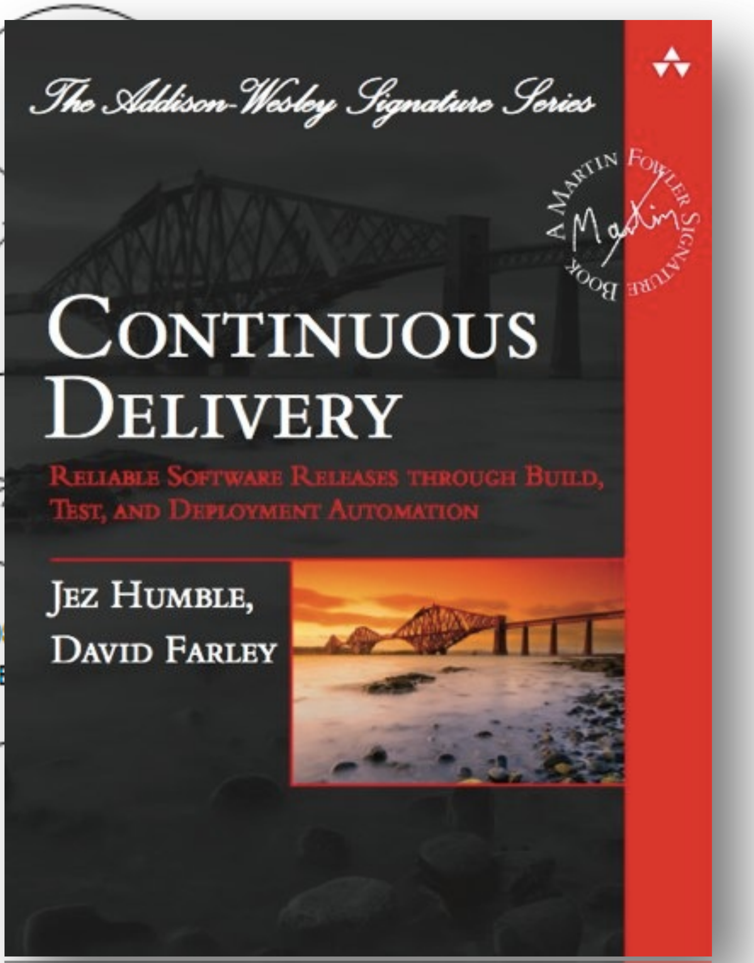

Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler




Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

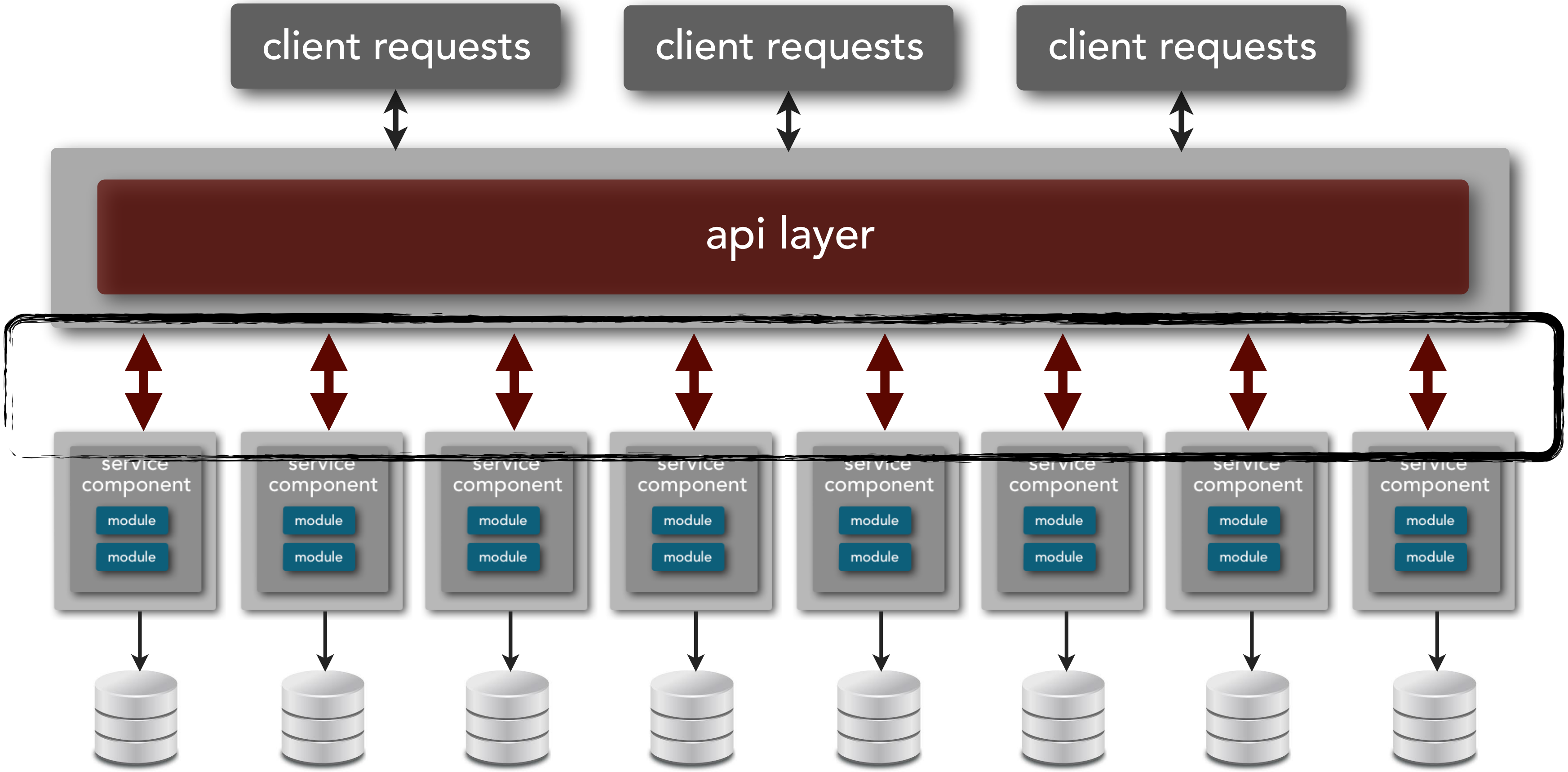
Bounded Context




The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble,
David Farley

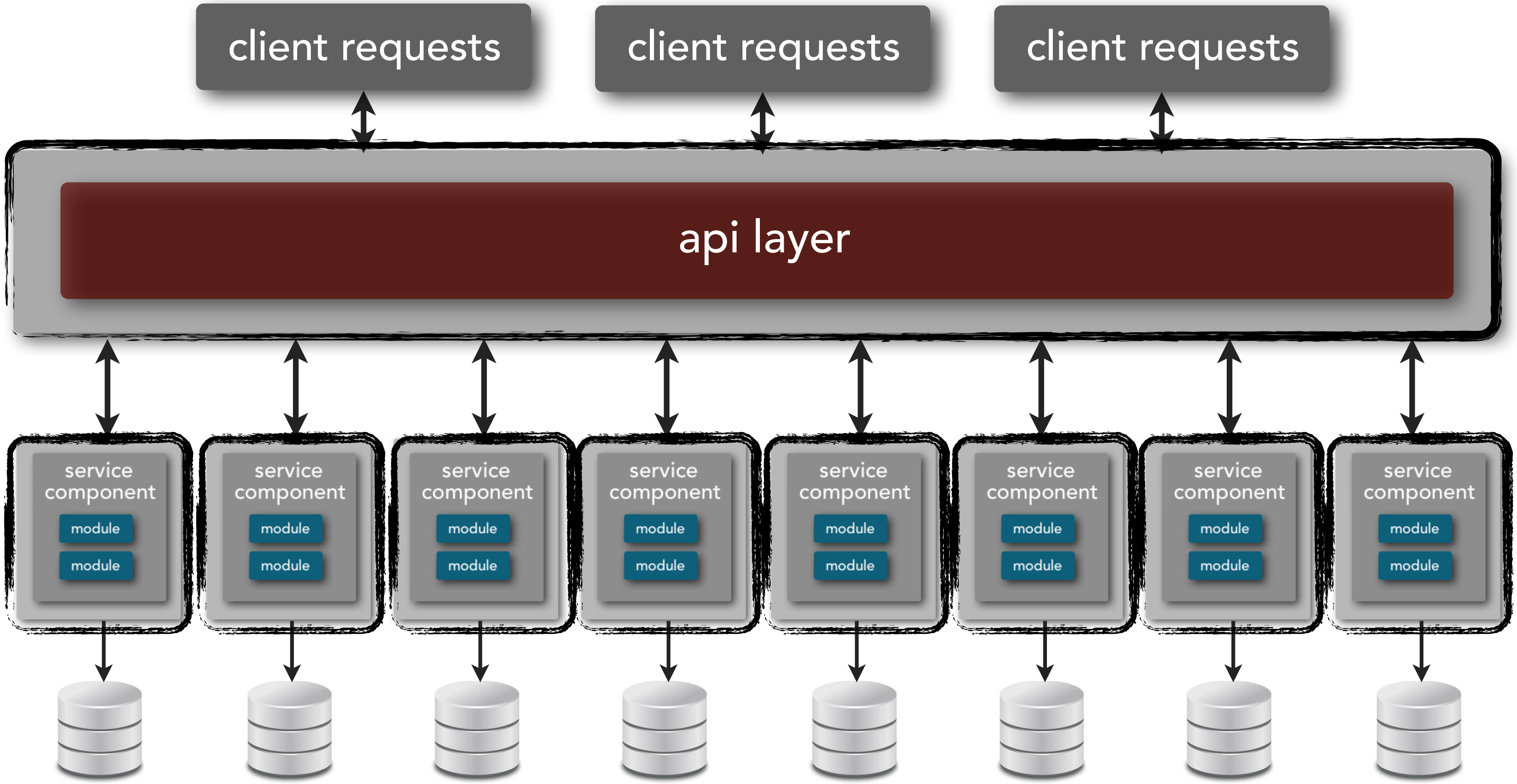
Microservices Architecture

distributed architecture



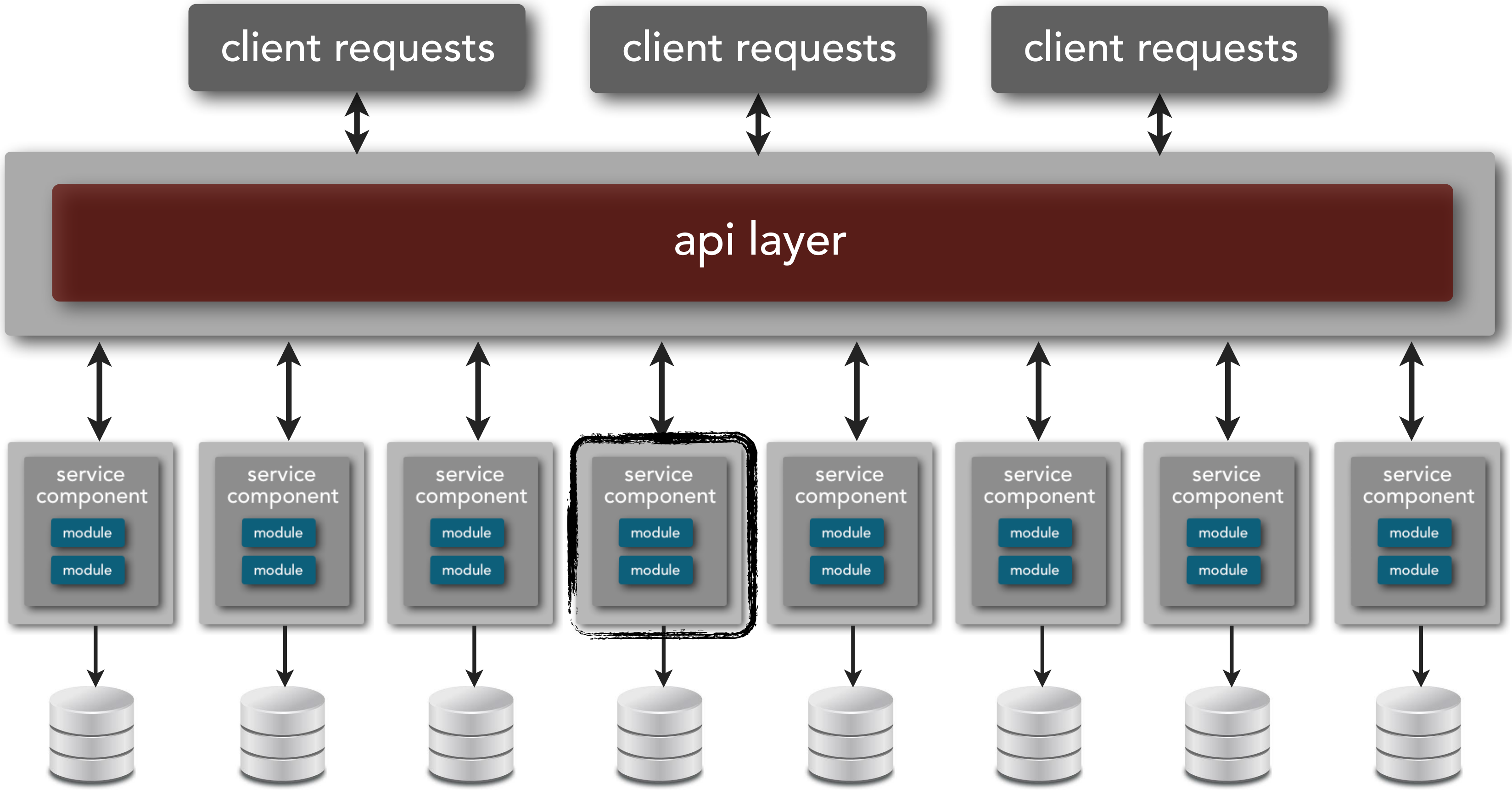
Microservices Architecture

separately deployed components



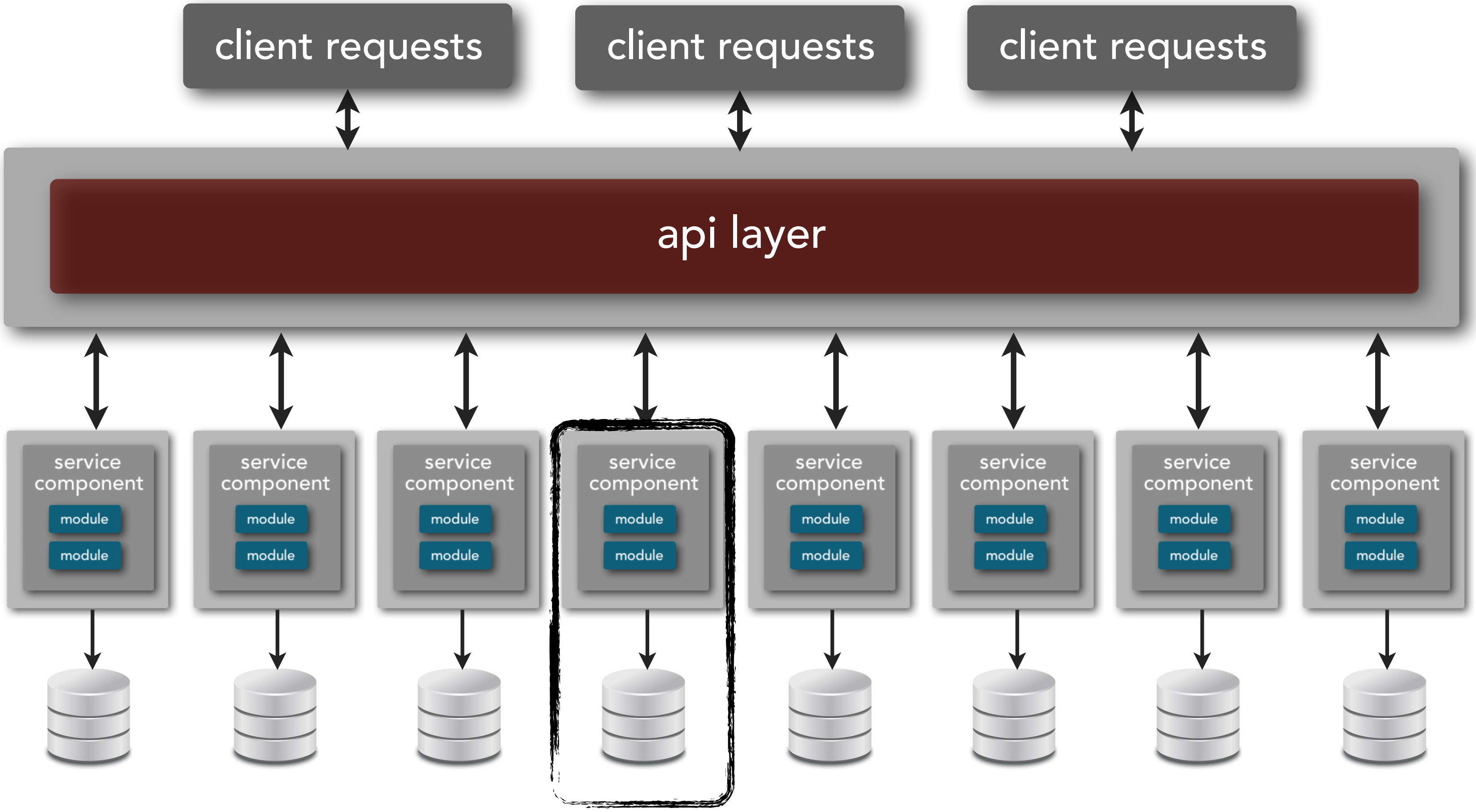
Microservices Architecture

service component



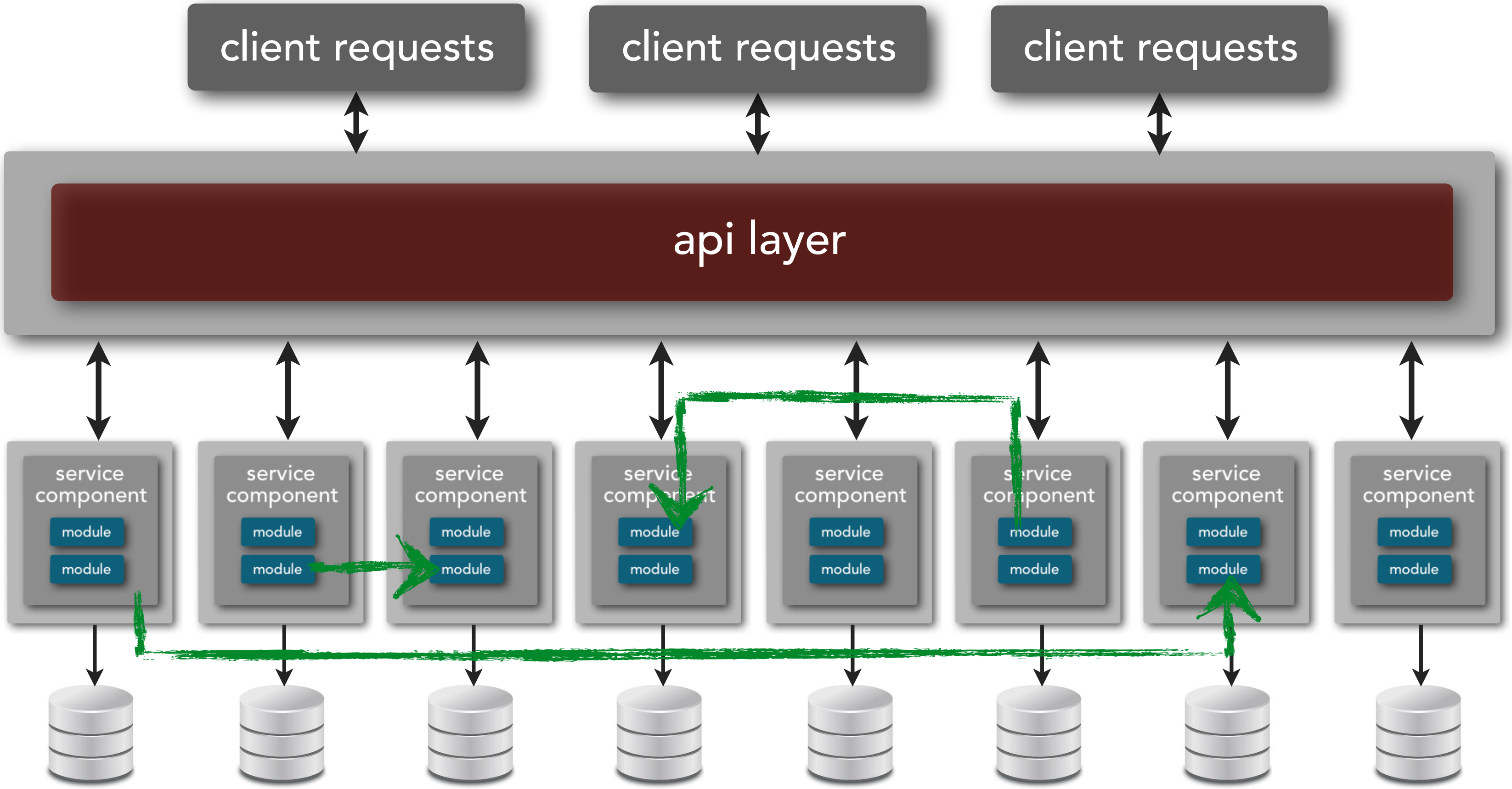
Microservices Architecture

bounded context



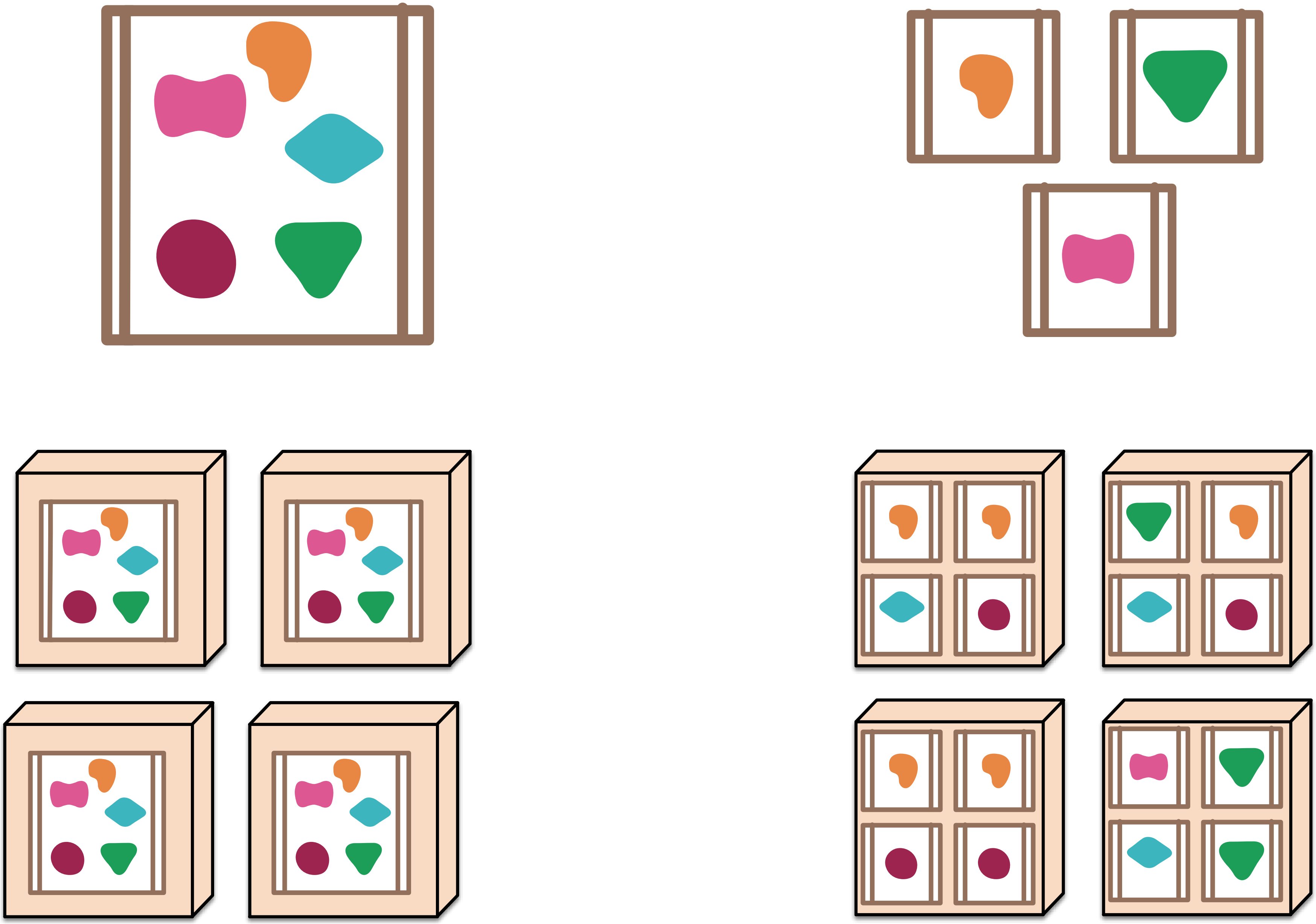
Microservices Architecture

service orchestration

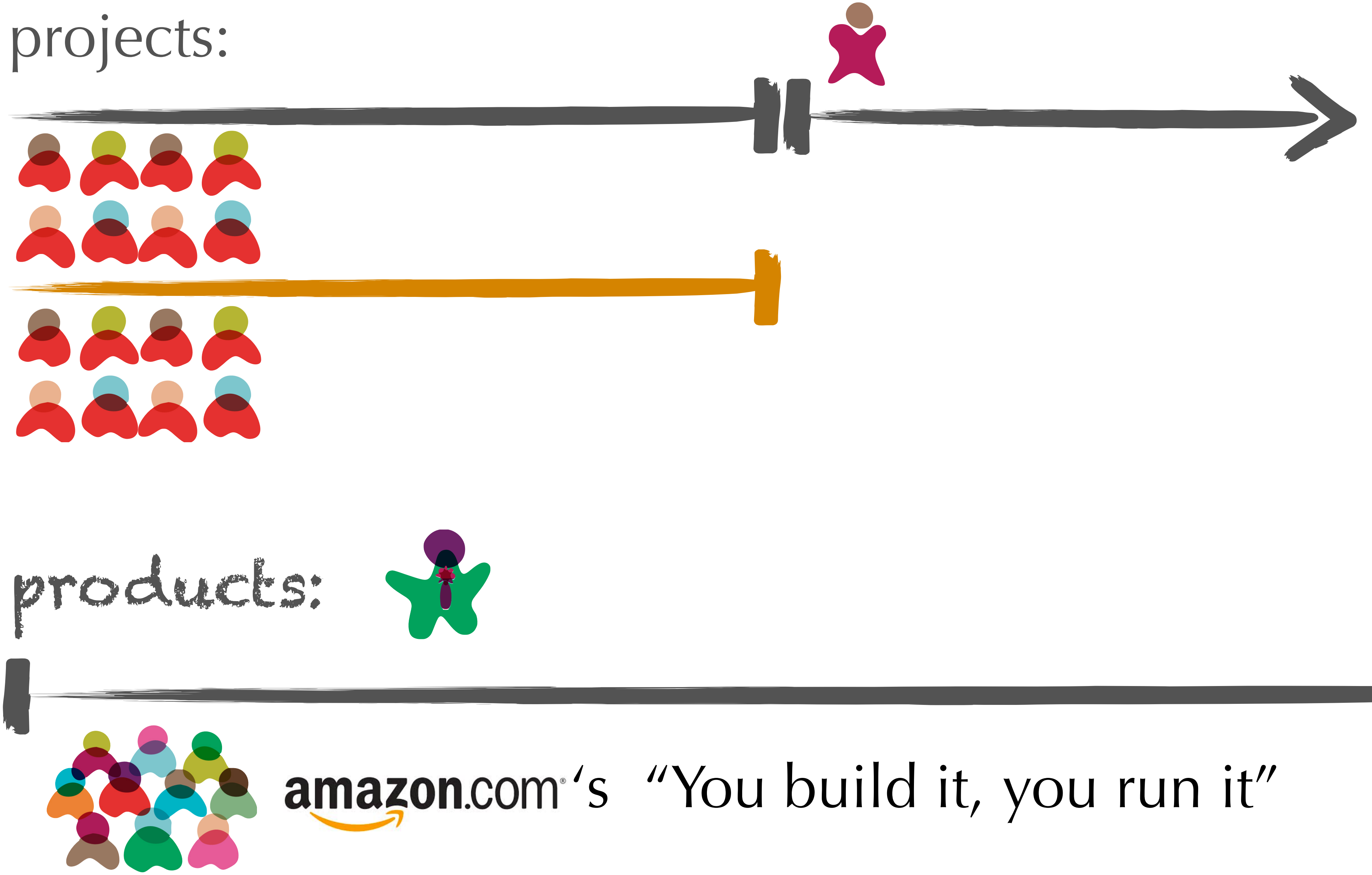


Monoliths vs. Microservices

Monoliths vs. Microservices

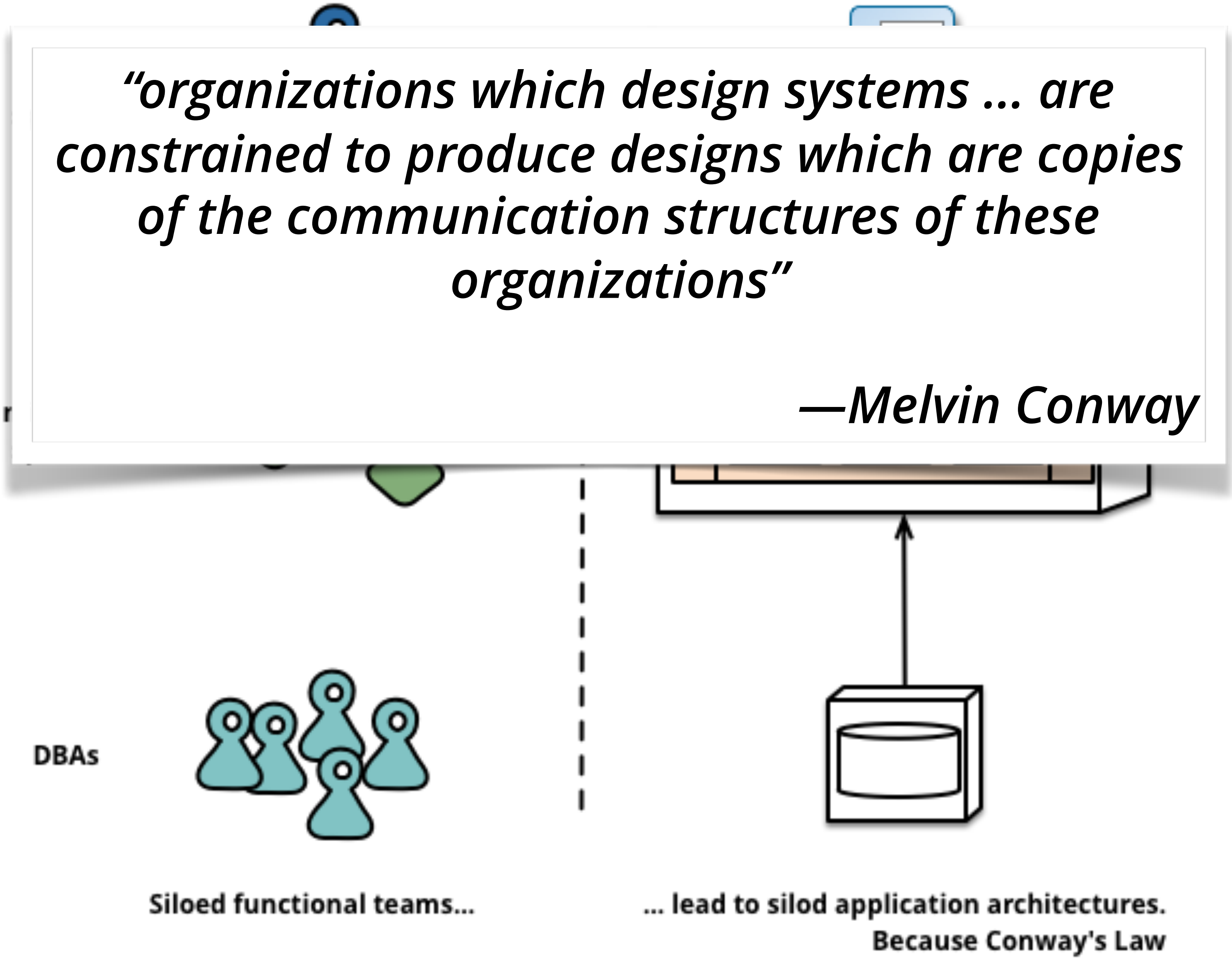


Products, not Projects

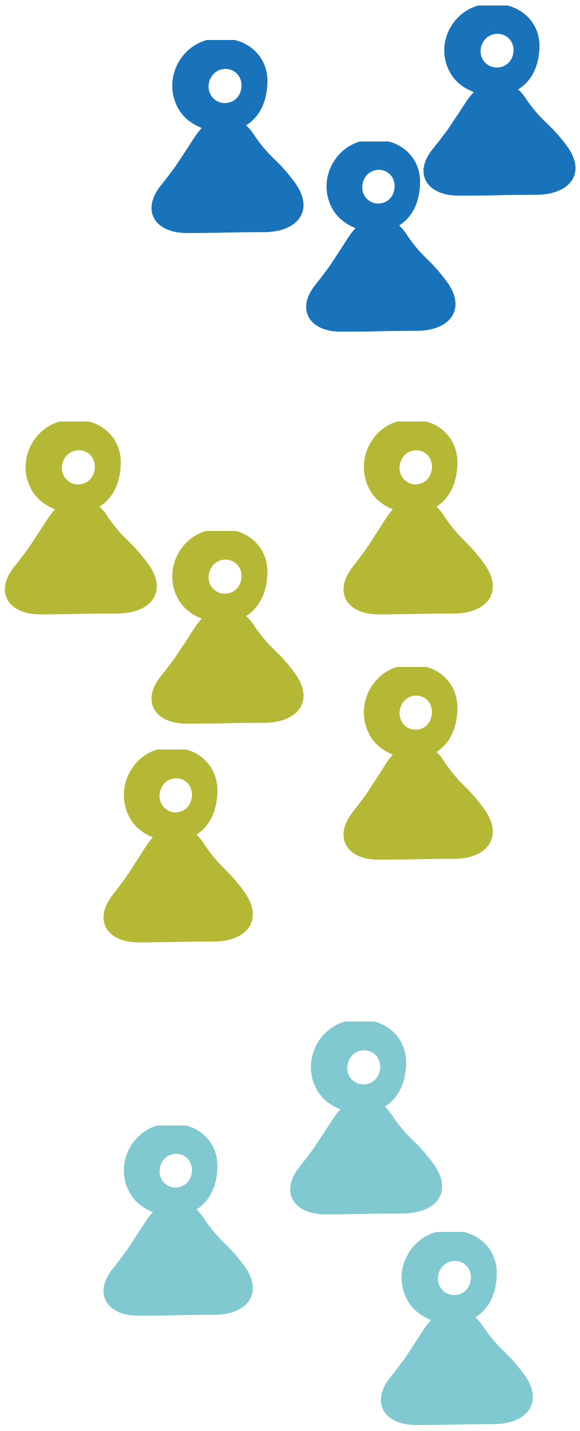


amazon.com's "You build it, you run it"

Conway’s Law


DBAs
Siloed functional teams...
... lead to silod application architectures.
Because Conway's Law

> *“organizations which design systems … are constrained to produce designs which are copies of the communication structures of these organizations”*
>
> *—Melvin Conway*



user interface

server-side

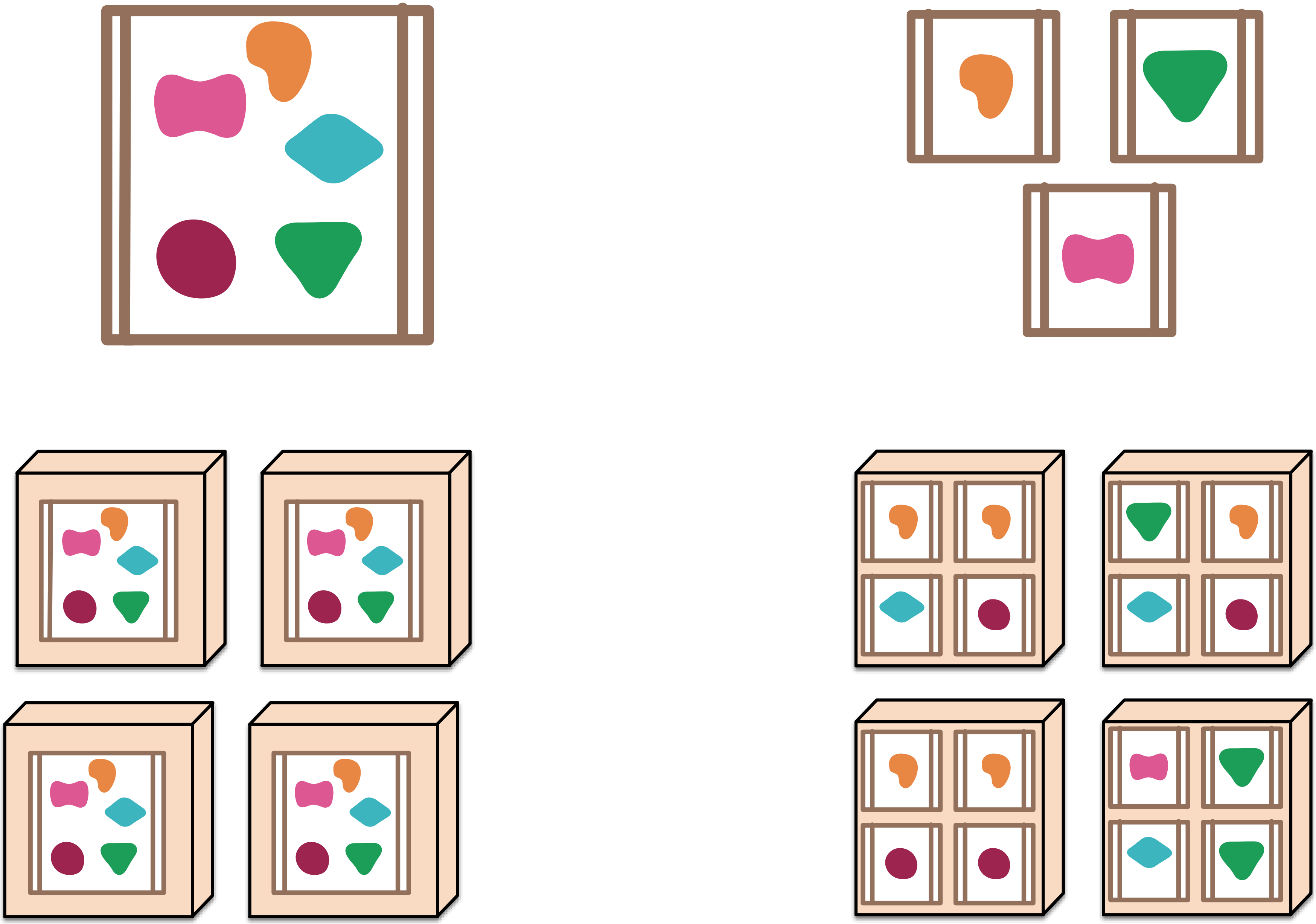
DBA

Orders

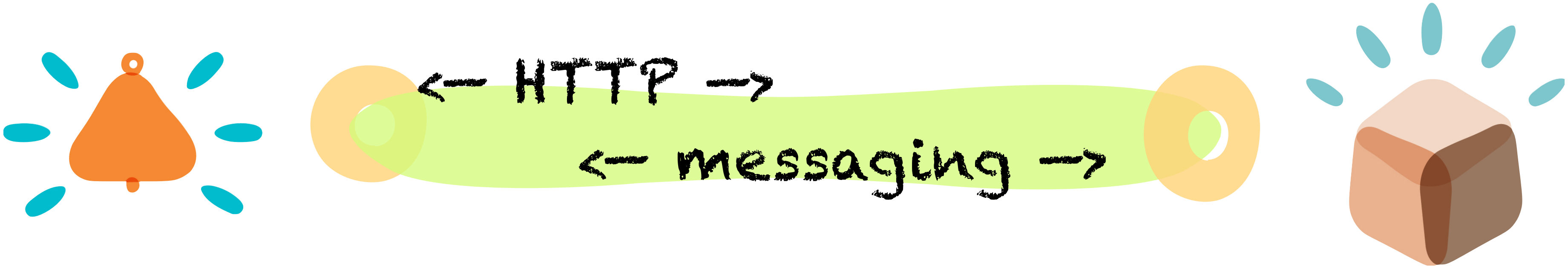
Shipping

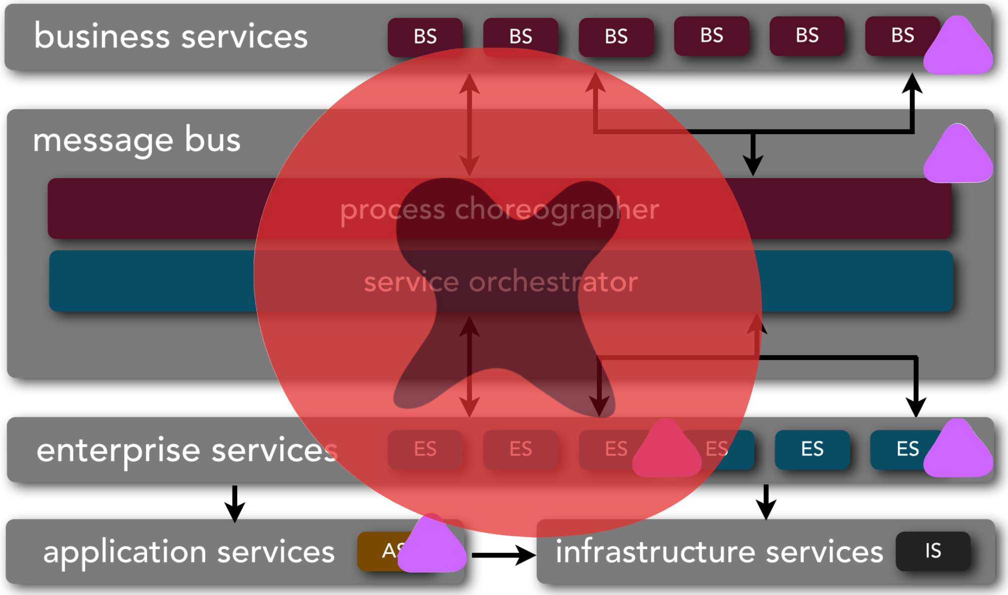
Catalog

Monoliths vs. Microservices



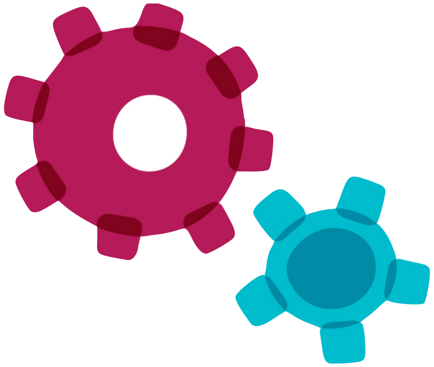
Smart Endpoints, Dumb Pipes

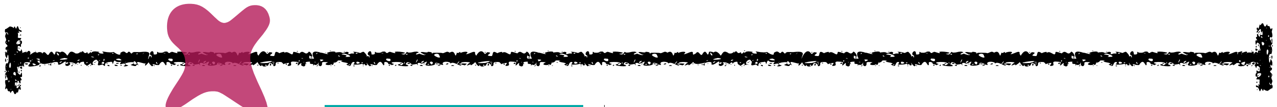




Standardize on integration, not platform

embrace polyglot solutions
where sensible





too few
languages/platforms

too many
languages/platforms

O'REILLY

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS

Sam Newman

> *Have one, two or maybe three ways of integrating, not 20.*

O'REILLY

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS

Sam Newman

> *Standardize in the gaps between services - be flexible about what happens inside the boxes*

O'REILLY

Building Microservices

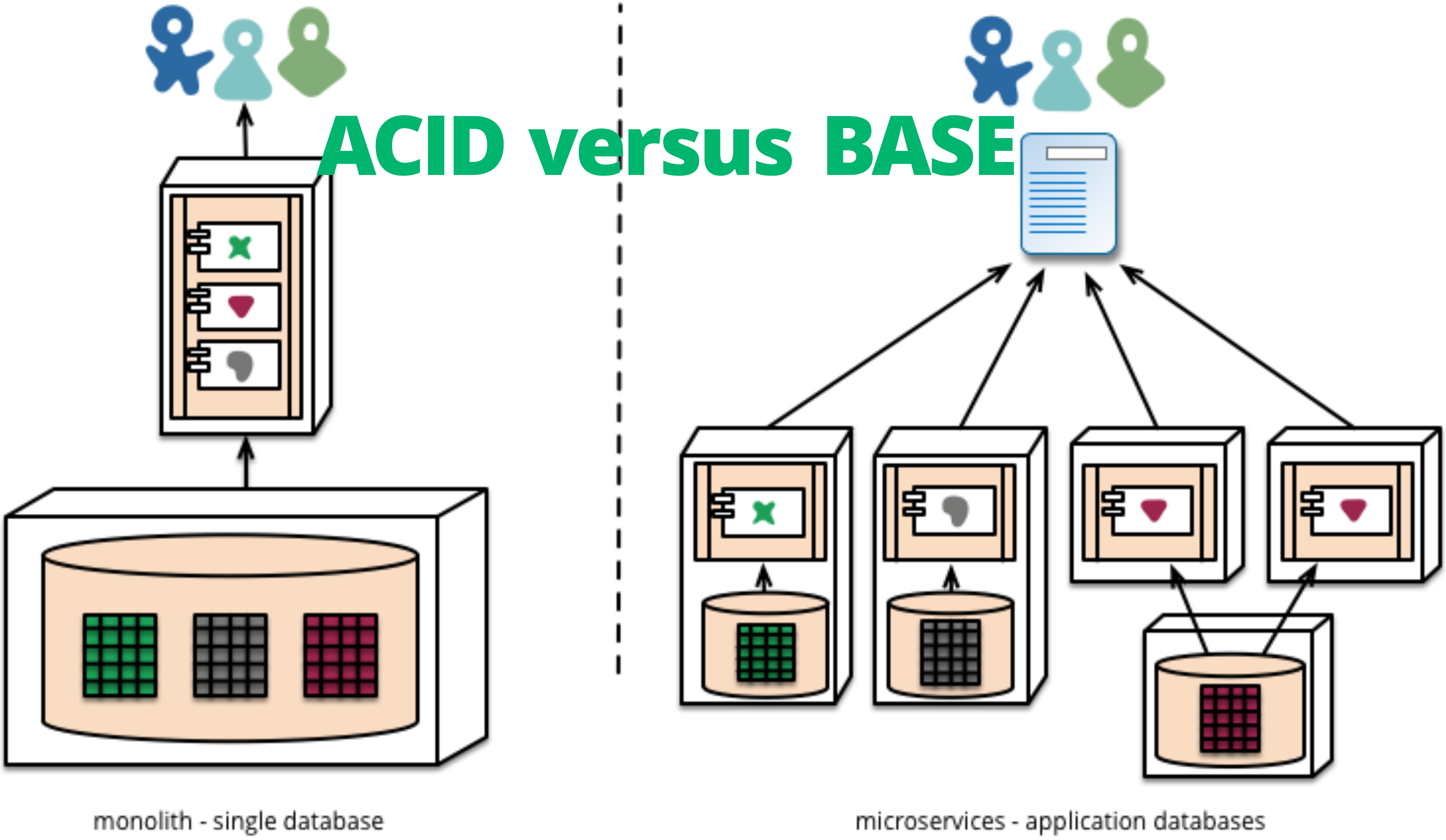
DESIGNING FINE-GRAINED SYSTEMS

Sam Newman

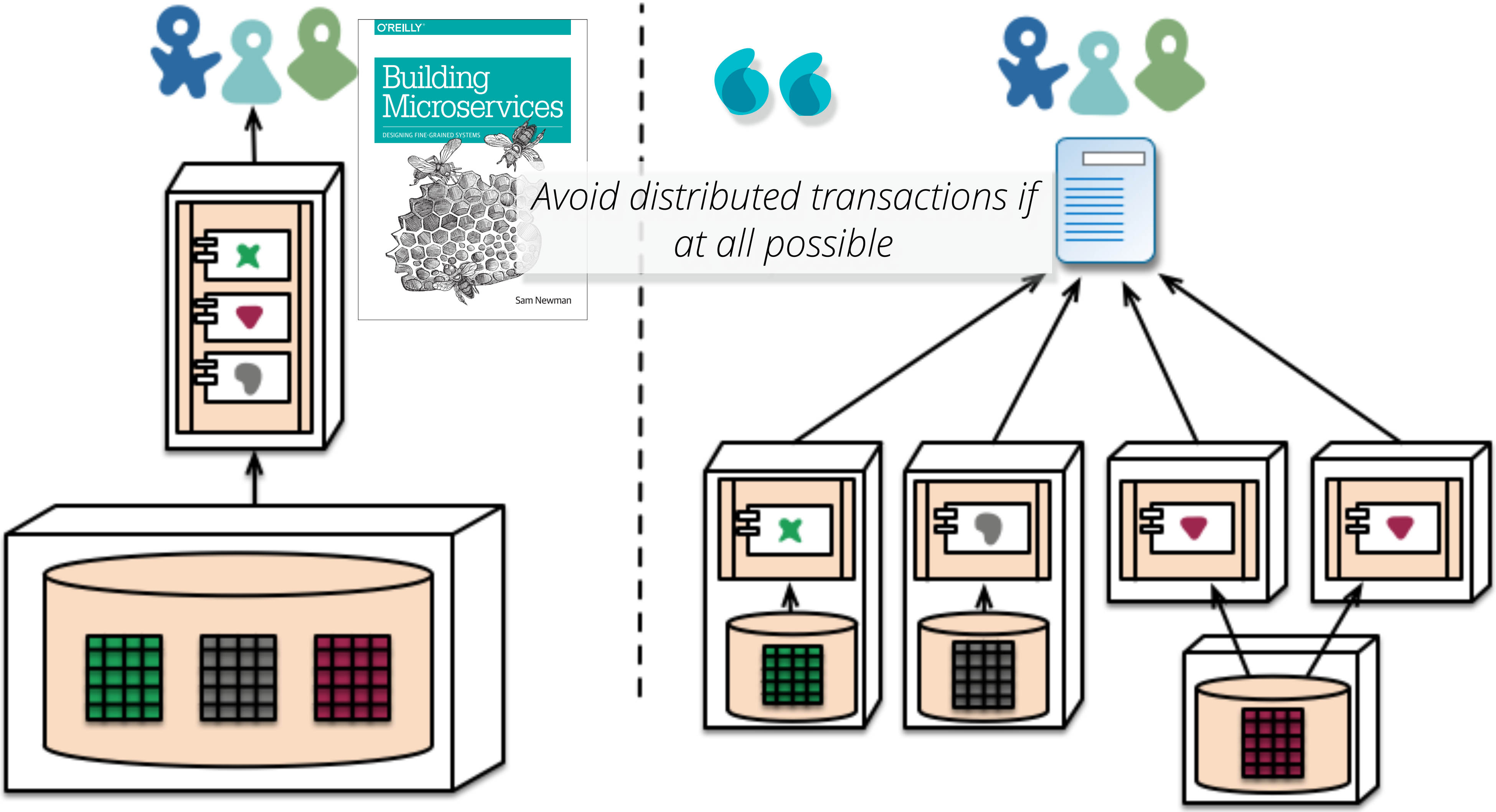
> *Pick some sensible conventions, and stick with them.*

Decentralized Data Management

ACID versus BASE


monolith - single database
microservices - application databases

Decentralized Data Management

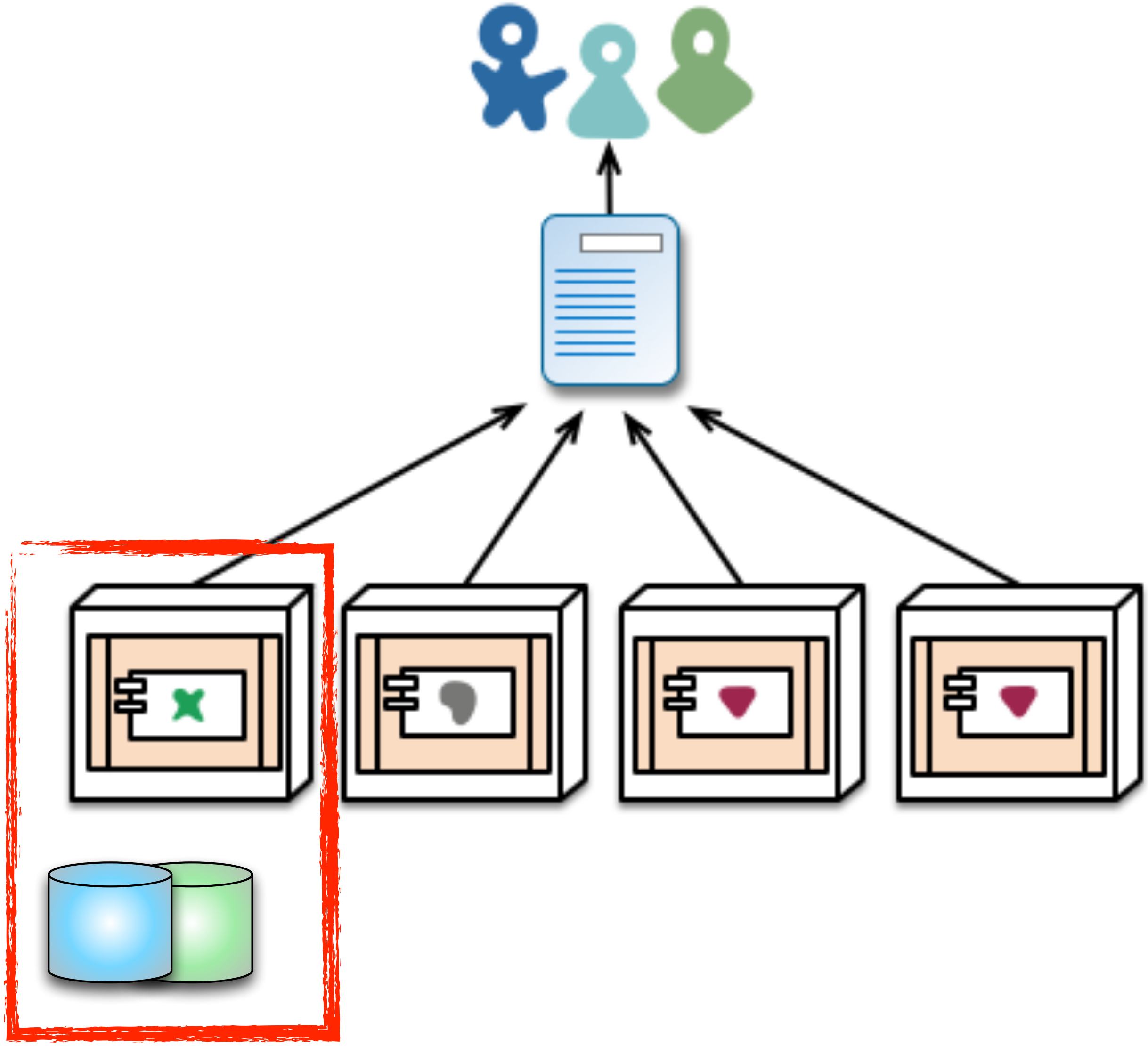


> Avoid distributed transactions if at all possible

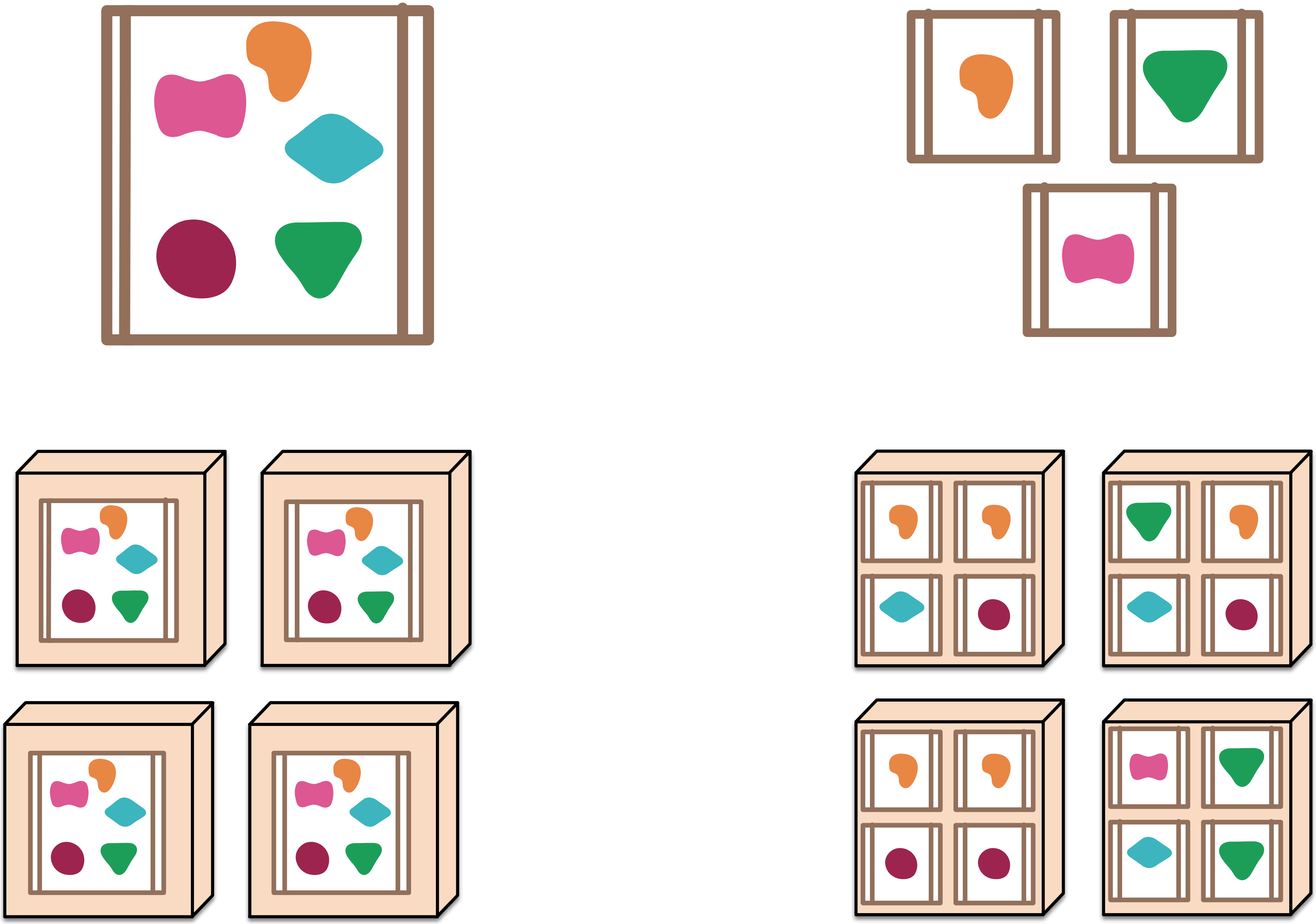
monolith - single database

microservices - application databases

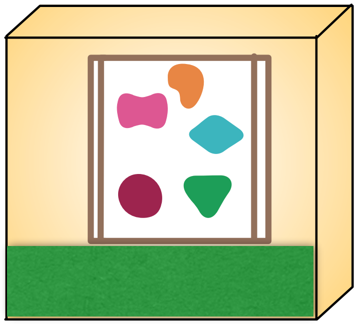
Decentralized Governance

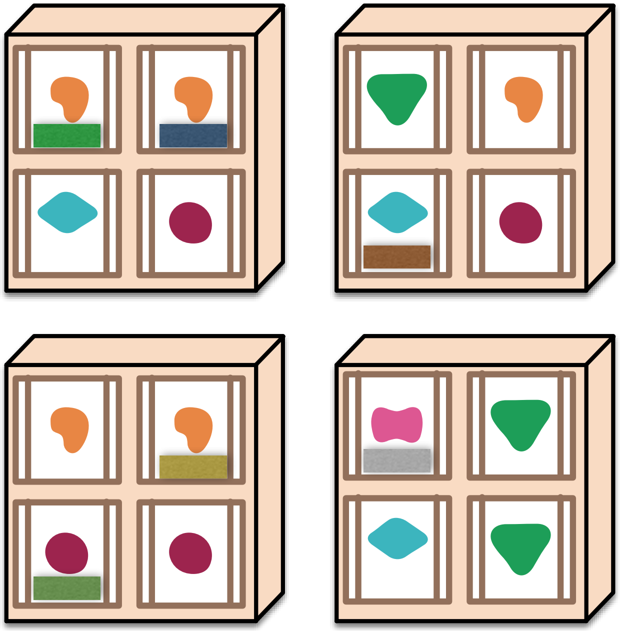


Decentralized Governance



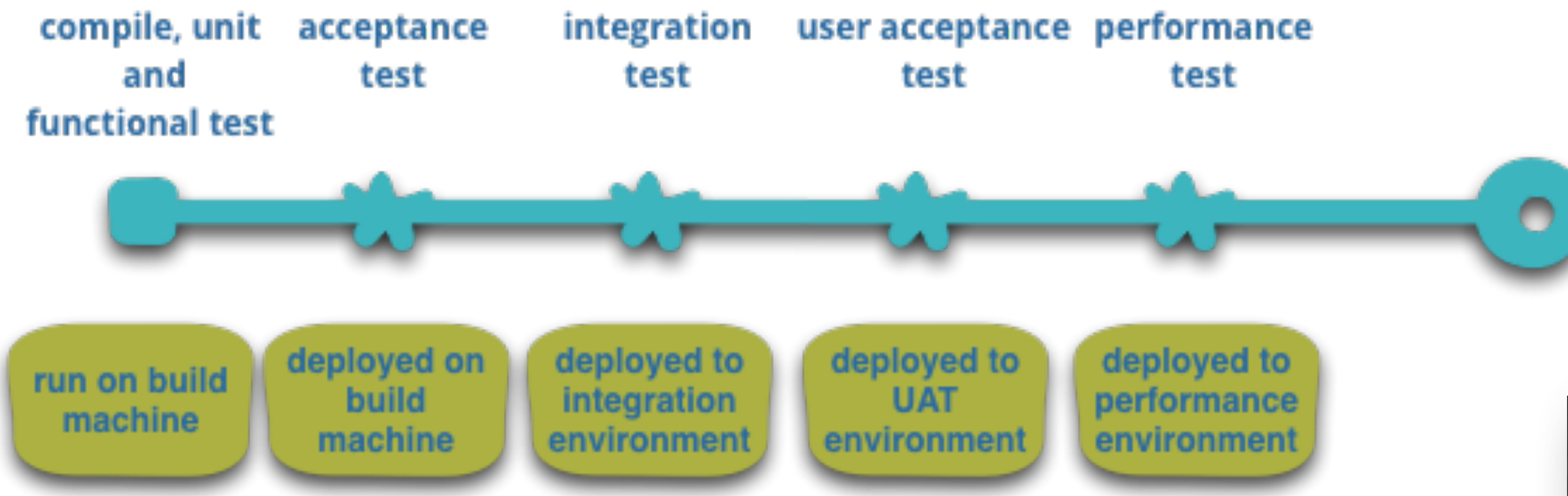
Decentralized Governance





Enterprise architects suffer from less pressure to make the correct choice(s) in microservice architectures.

Infrastructure Automation



compile, unit and functional test

acceptance test

integration test

user acceptance test

performance test

deploy to production

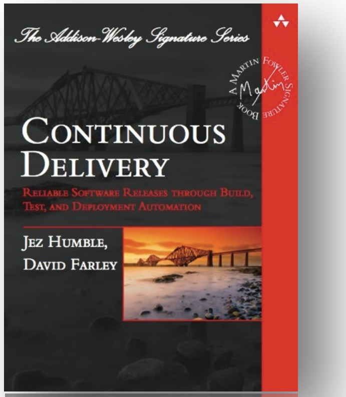
run on build machine

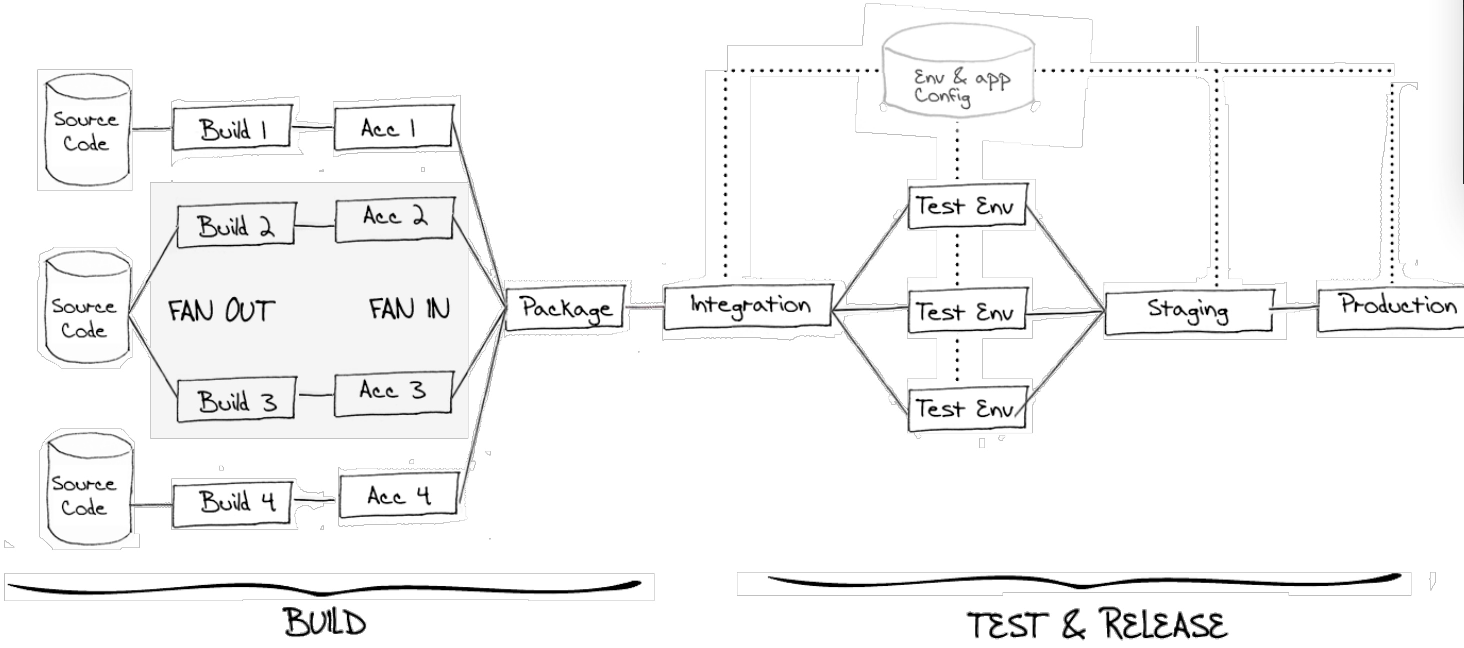
deployed on build machine

deployed to integration environment

deployed to UAT environment

deployed to performance environment


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble,
David Farley



Small, Single Responsibility

small enough to fit in your head

rewrite over maintain

(10—1000 LOC)-ish / service

single responsibility

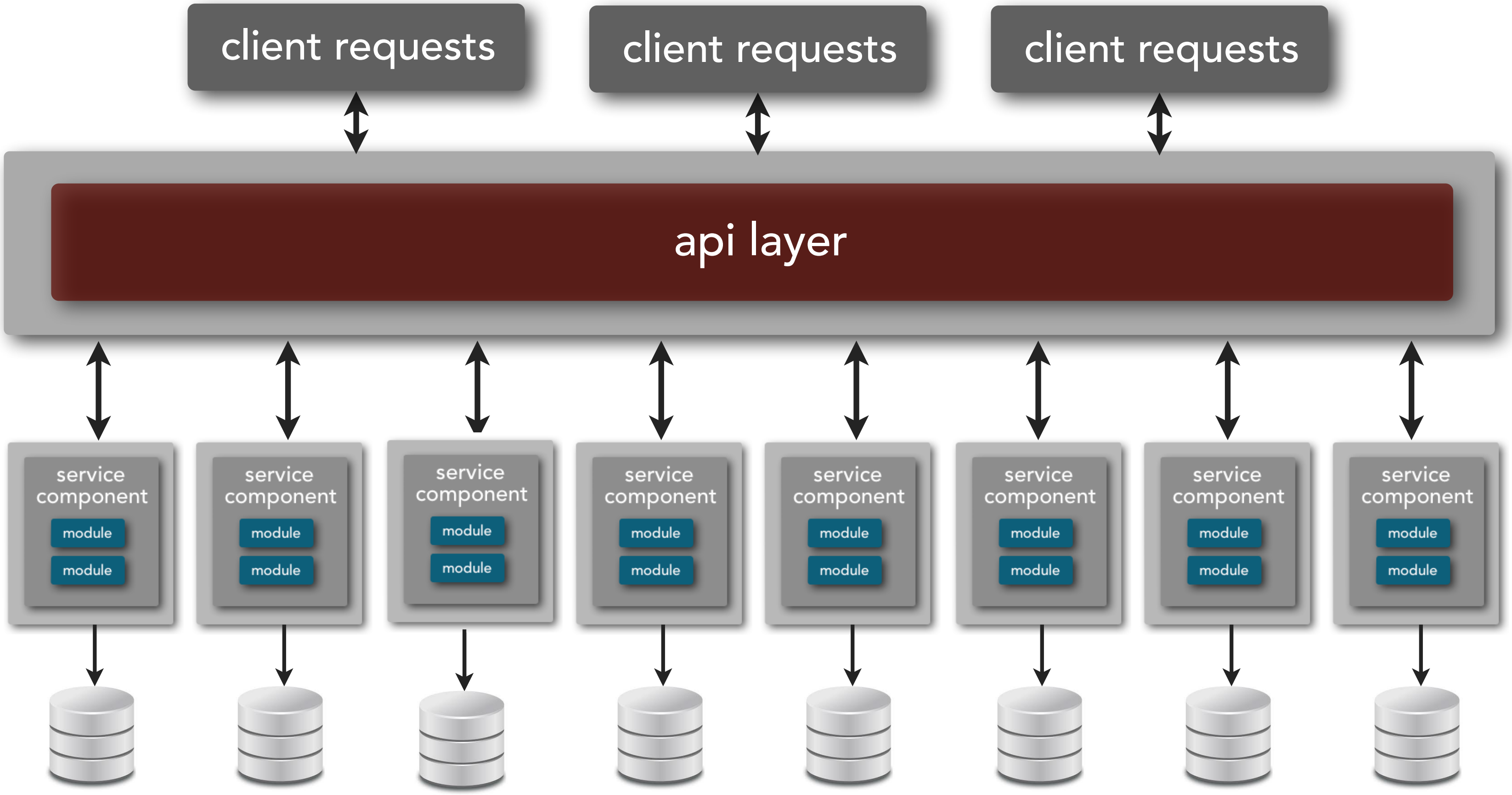
AGENDA

what problem

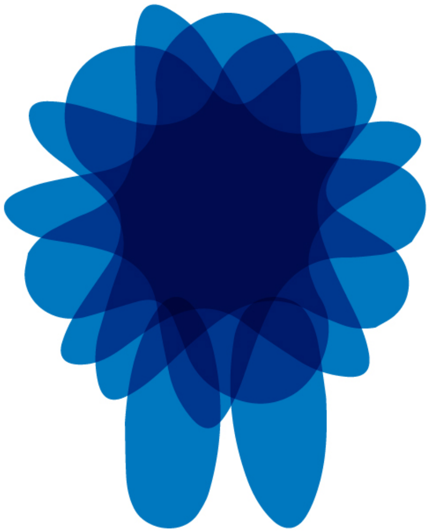
characteristics

engineering

Microservice



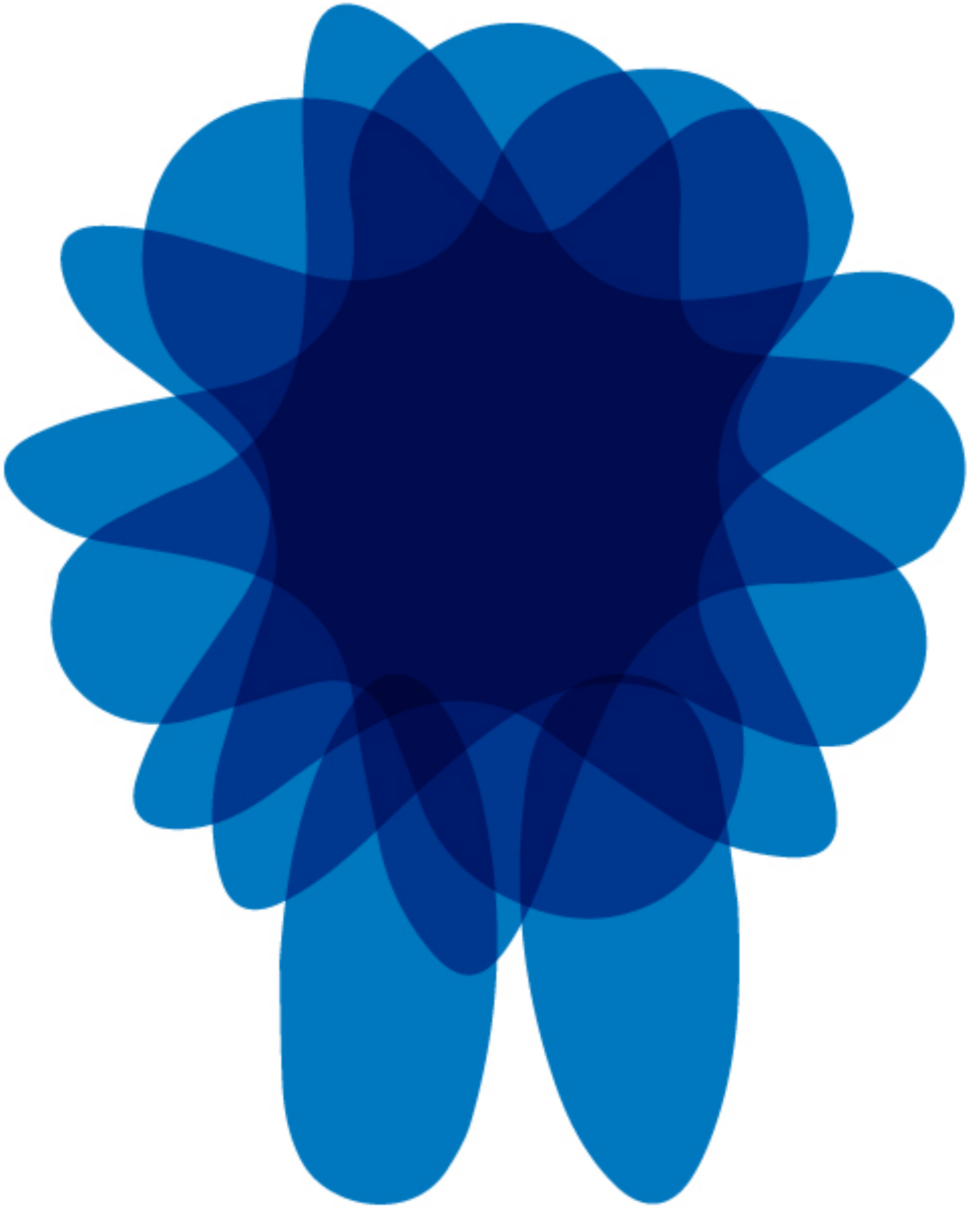
maximize easy evolution



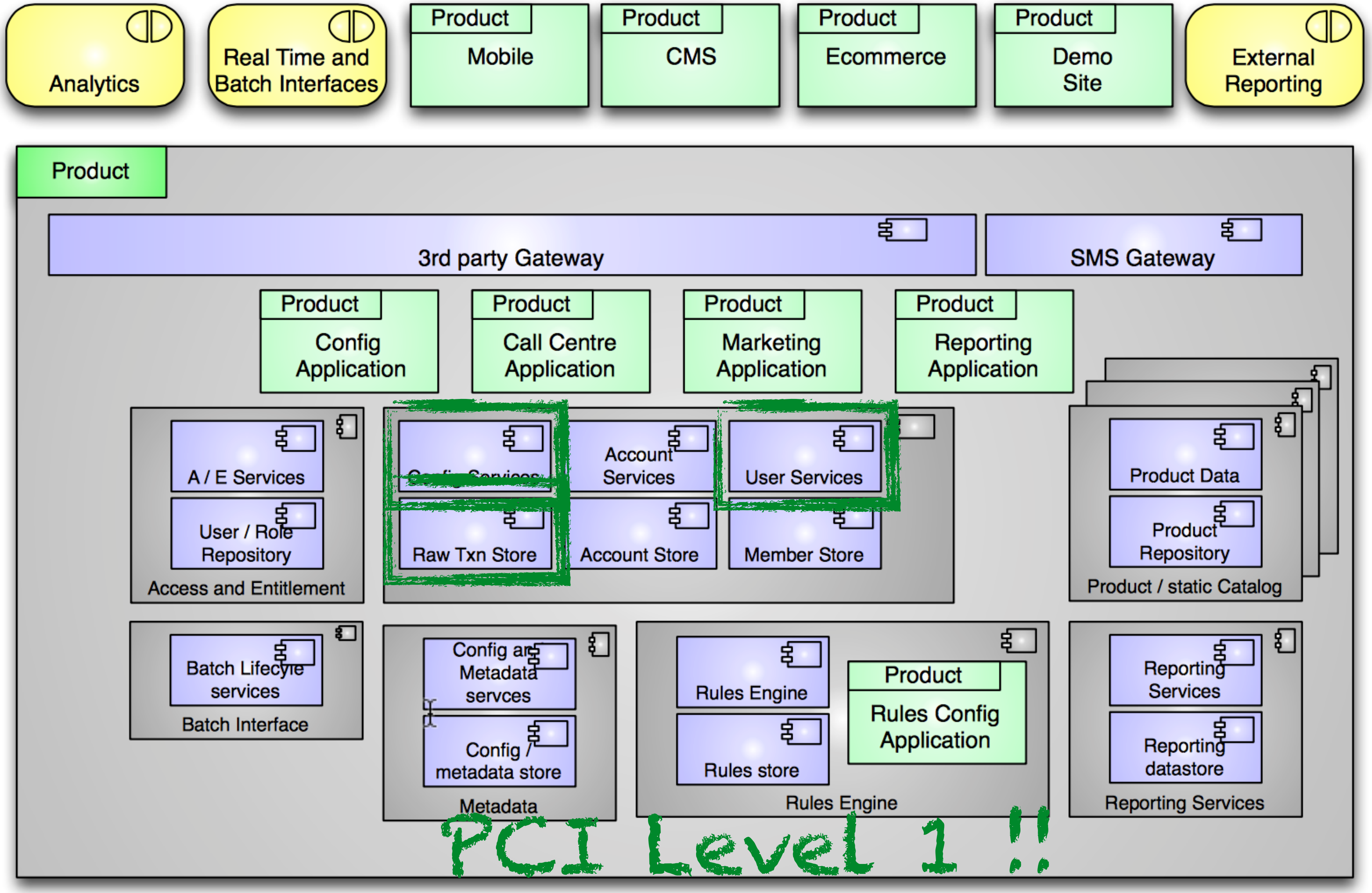
support Δ

Microservice is the first architectural style developed post-Continuous Delivery.

Benefits



Microservice Implementation

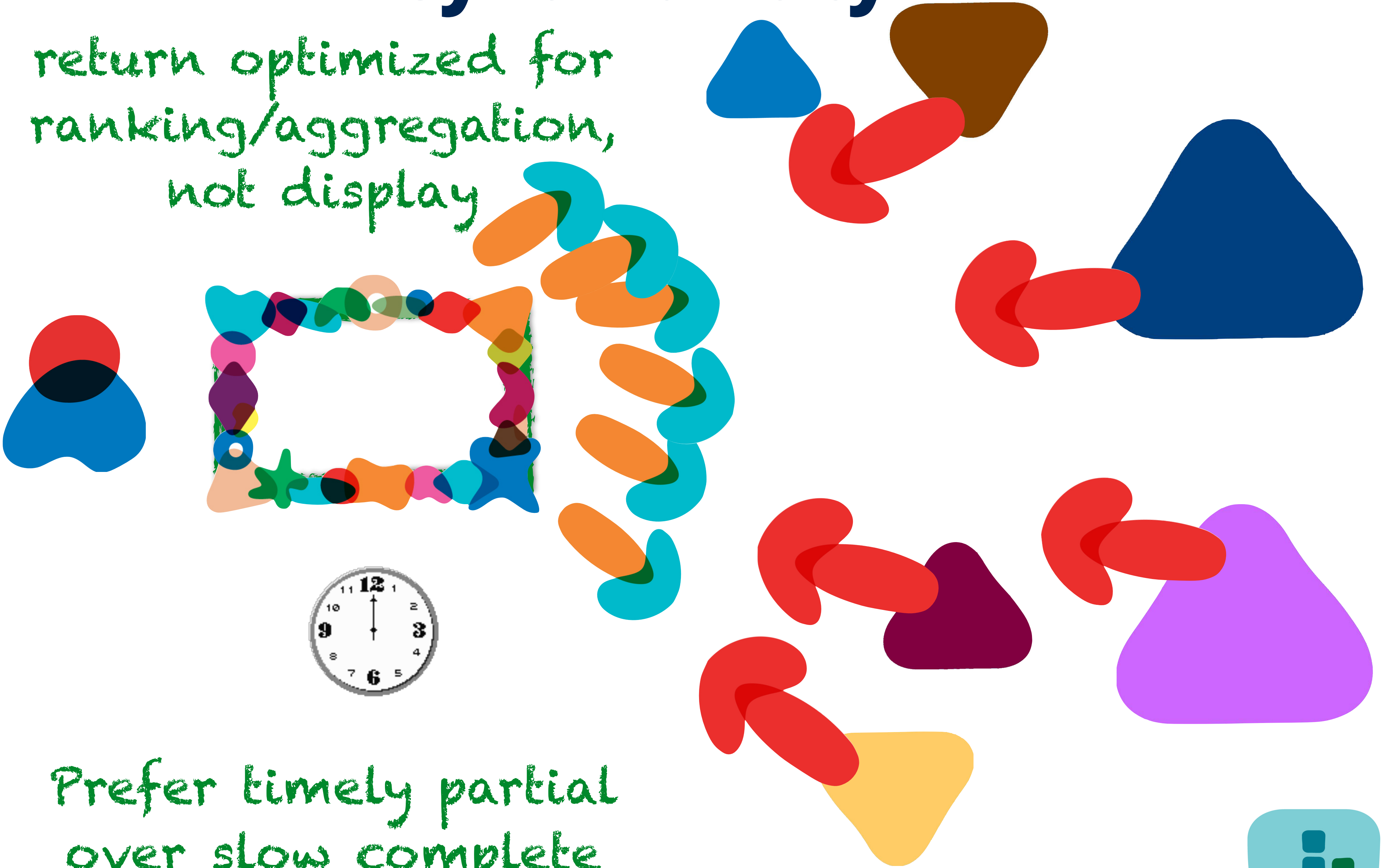


http://2012.33degree.org/pdf/JamesLewisMicroServices.pdf

http://www.infoq.com/presentations/Micro-Services

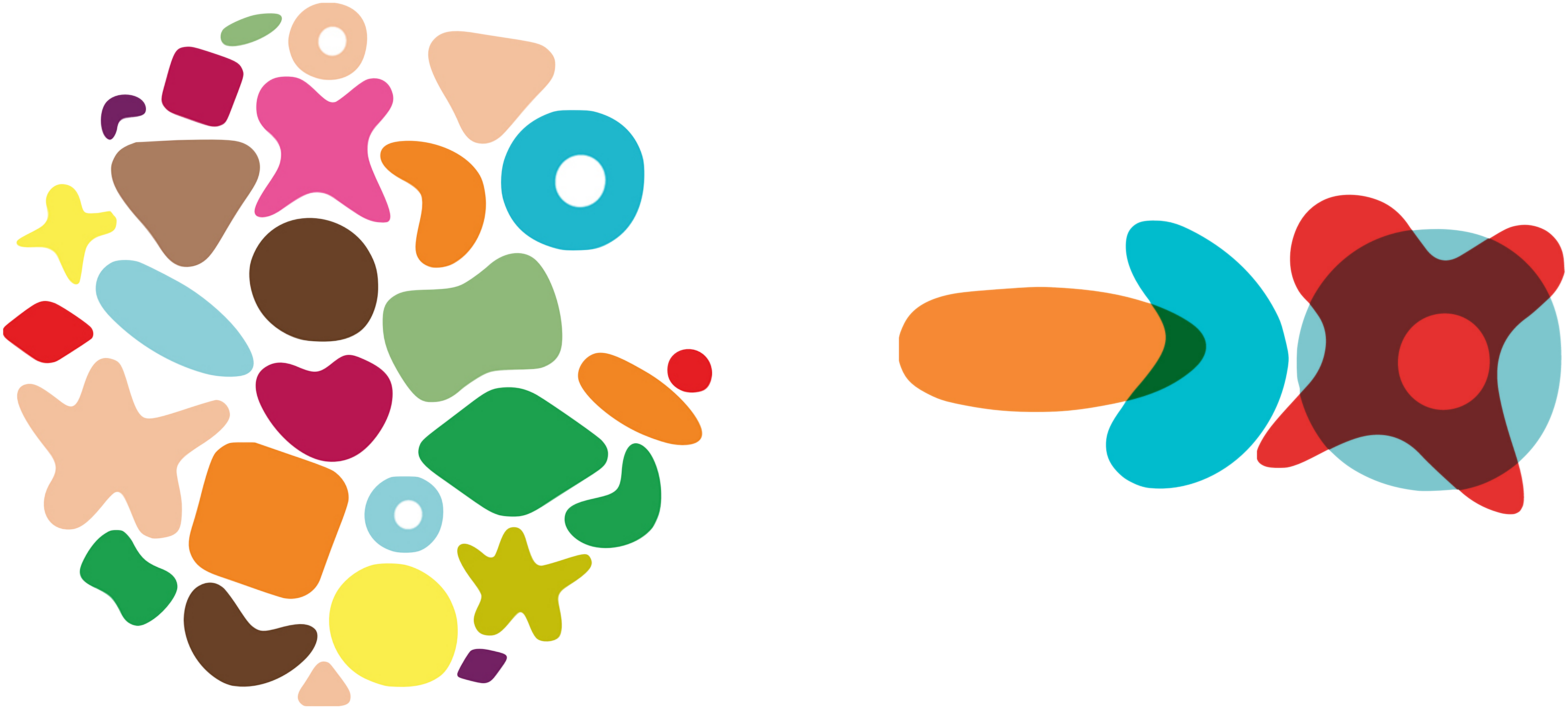
Asynchronicity

return optimized for ranking/aggregation, not display



Prefer timely partial over slow complete

Integration & Disintegration



Completed Deployments

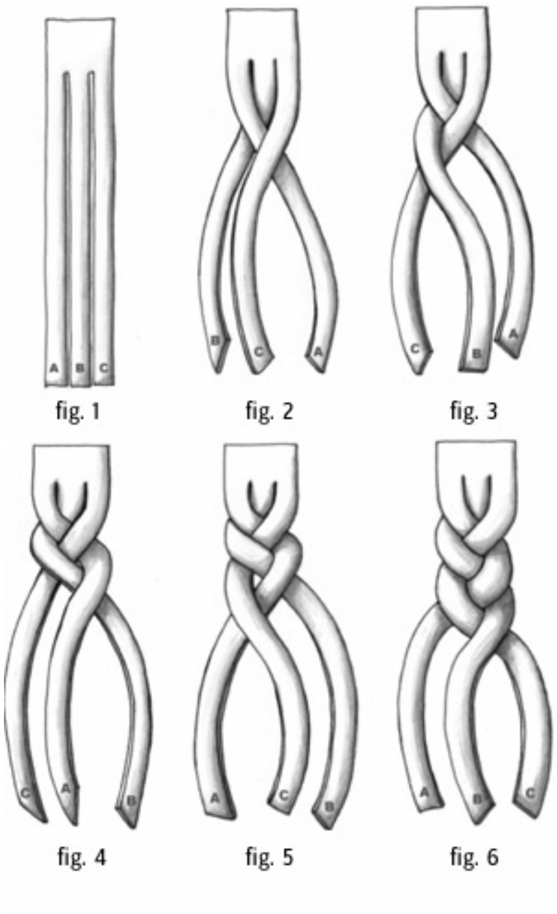


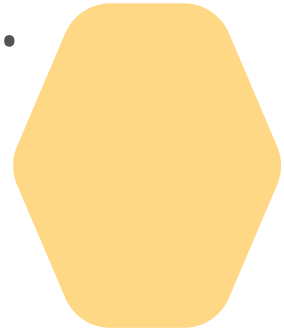
fig. 1 fig. 2 fig. 3

fig. 4 fig. 5 fig. 6

complect, *transitive verb*: intertwine, embrace, especially to plait together

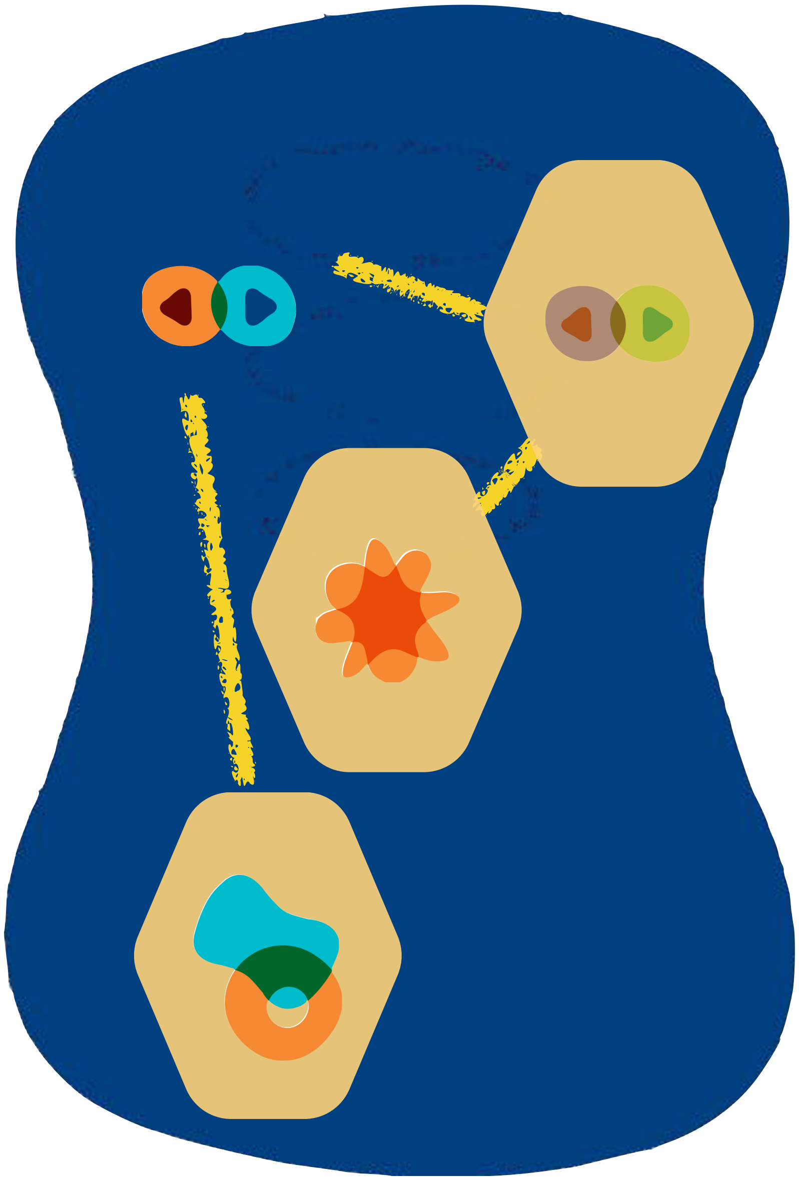
Evolutionary Architecture

Components are *deployed*.

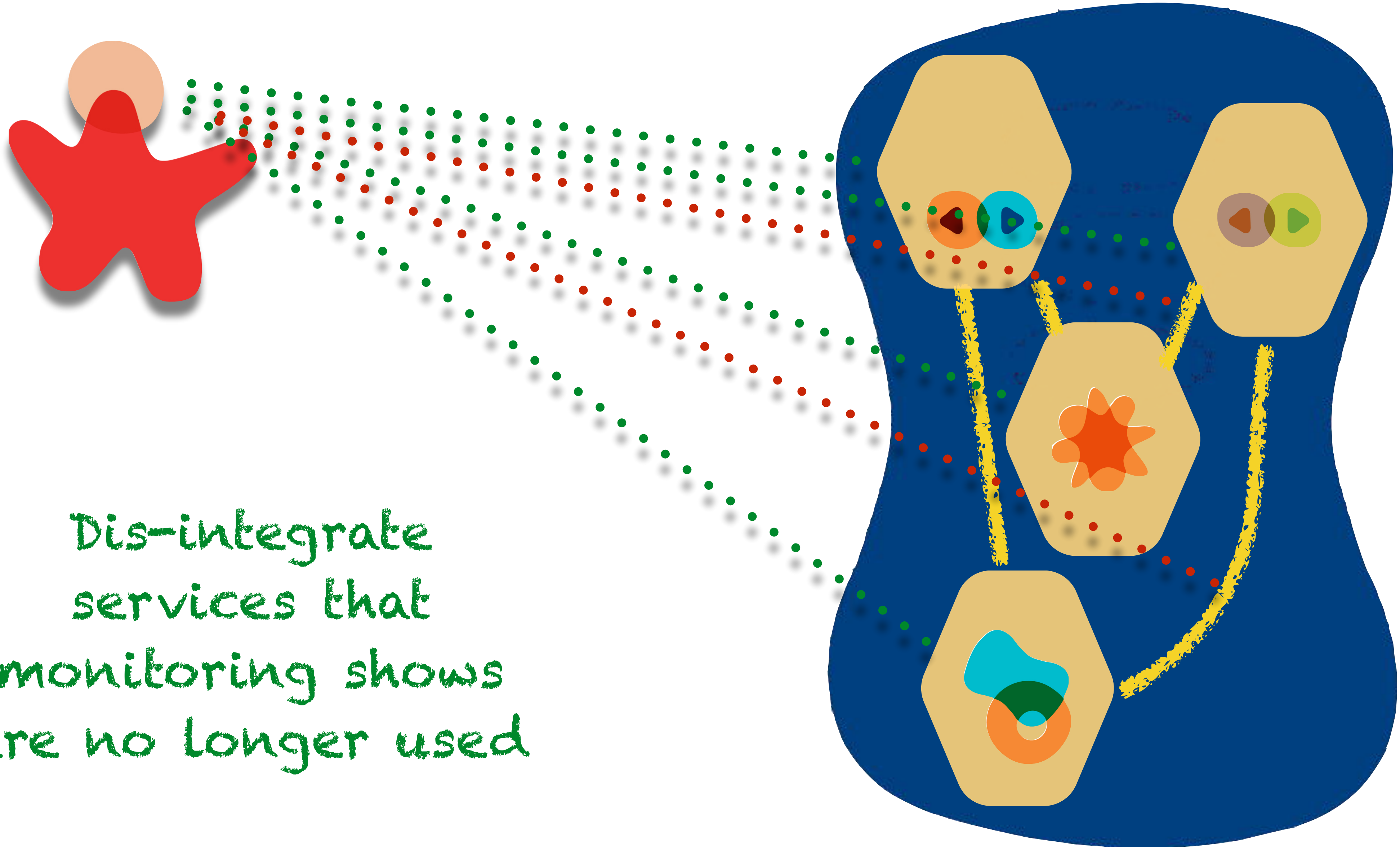


Features are *released*.

Applications consist of *routing*.

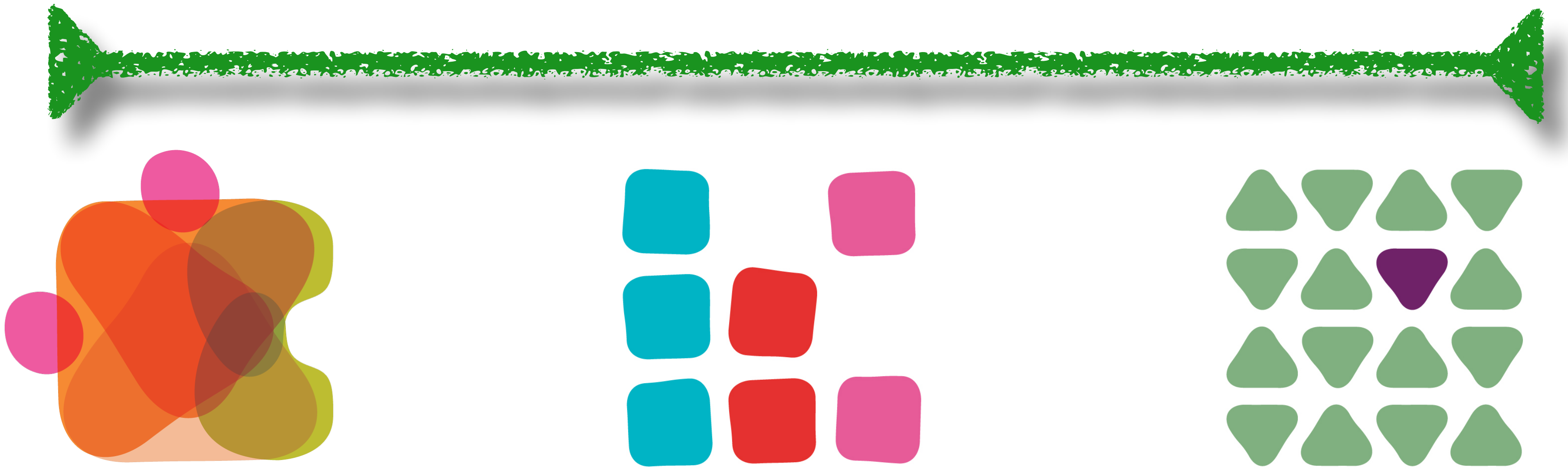


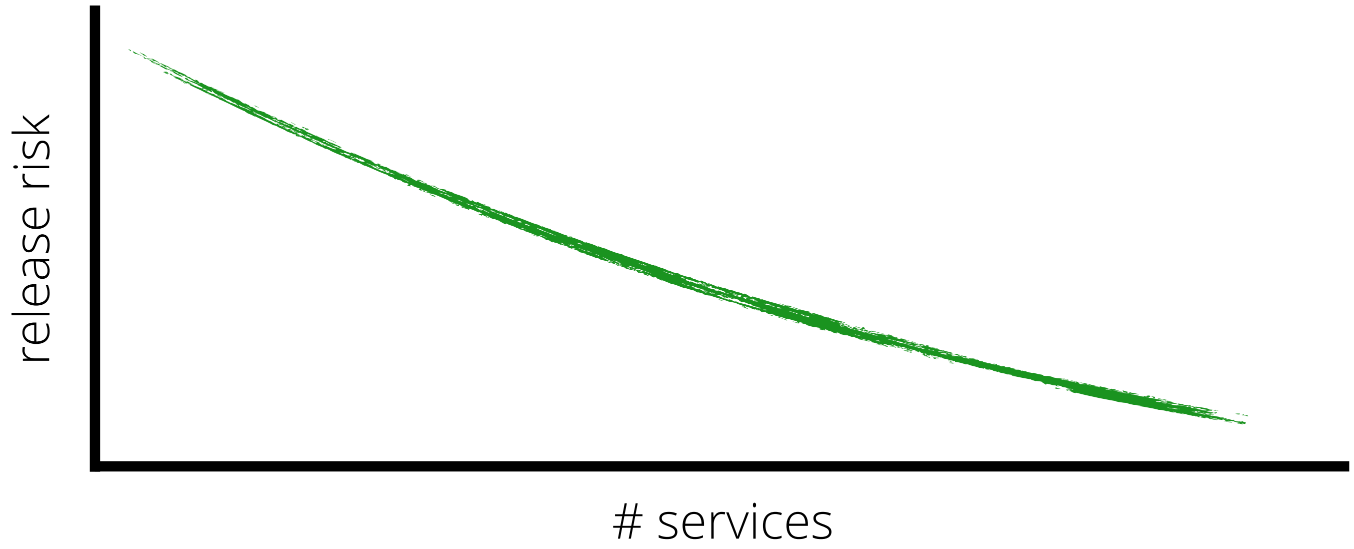
Evolutionary Architecture



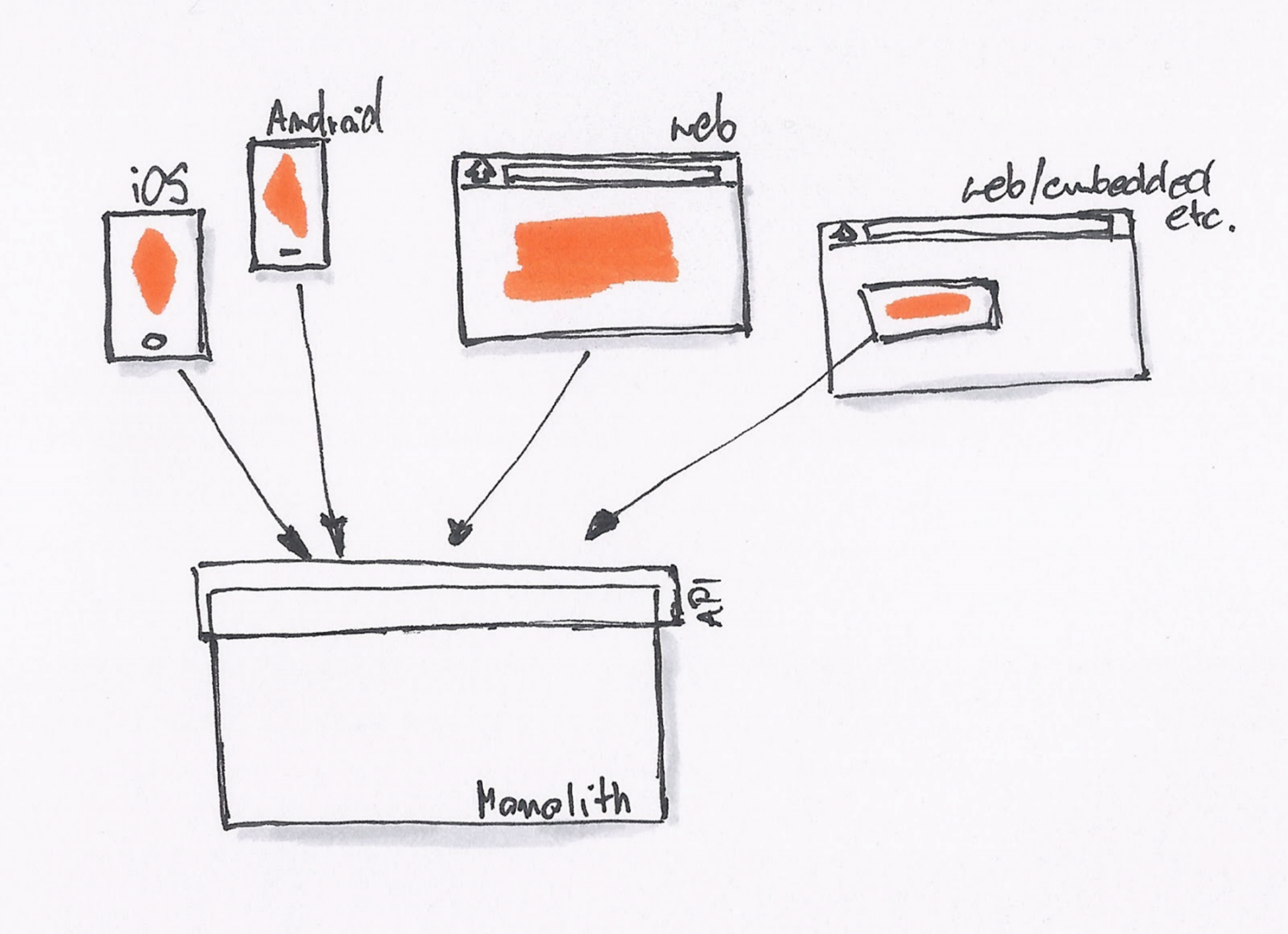
Dis-integrate services that monitoring shows are no longer used

How Big?



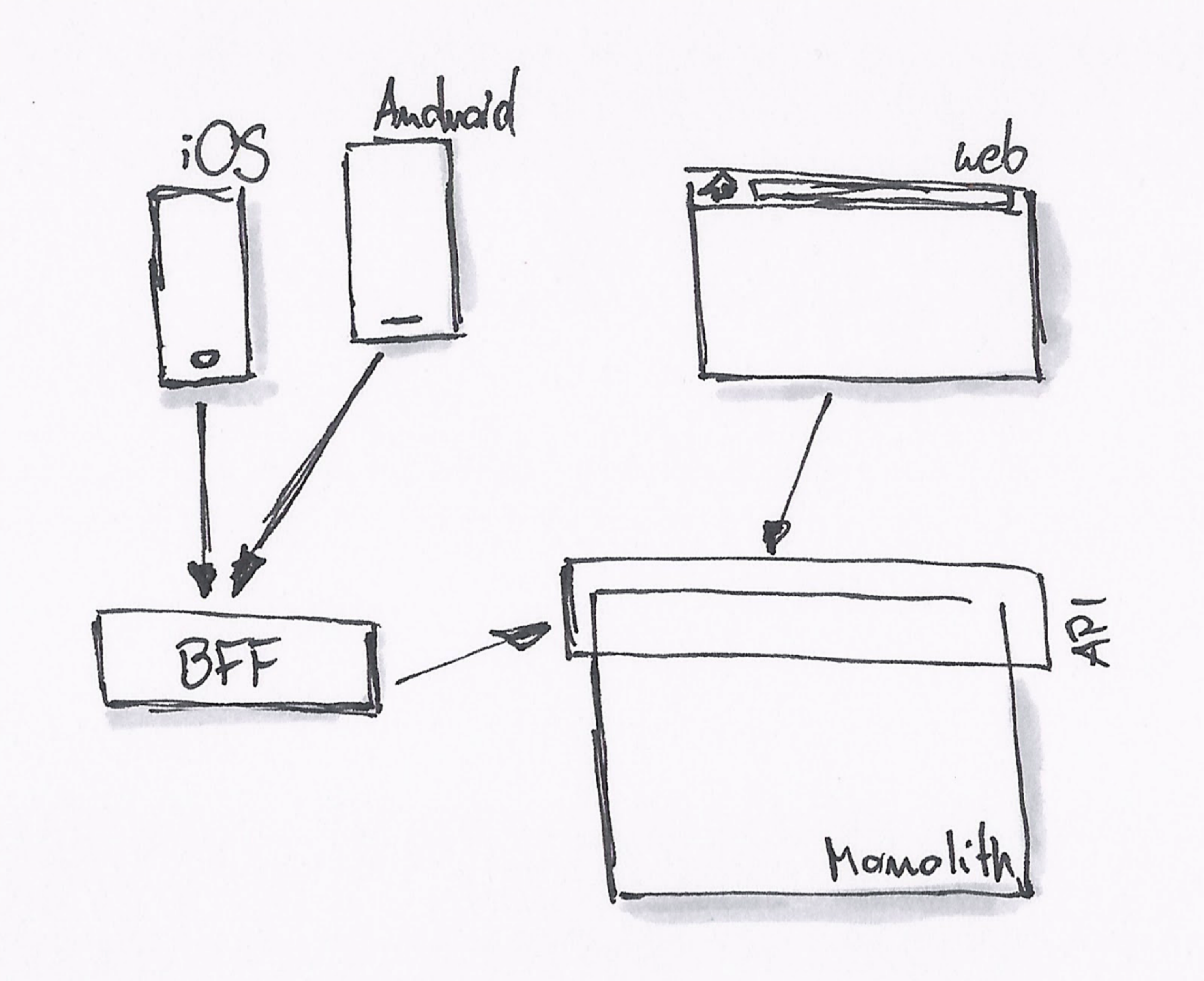


Backends for Frontends

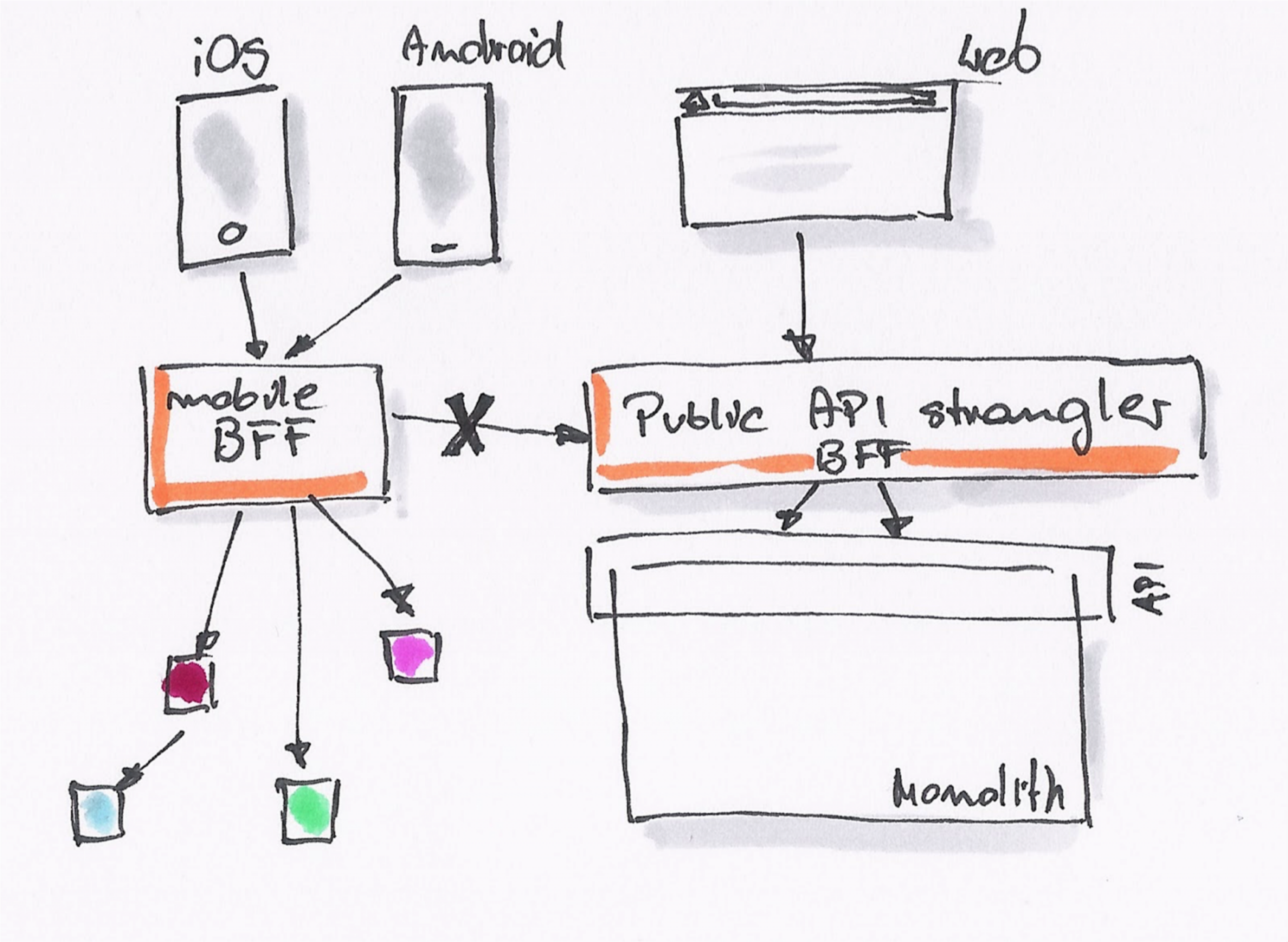


https://www.thoughtworks.com/insights/blog/bff-soundcloud

Backends for Frontends



BFF as Migration Path



AGENDA

what problem

characteristics

engineering

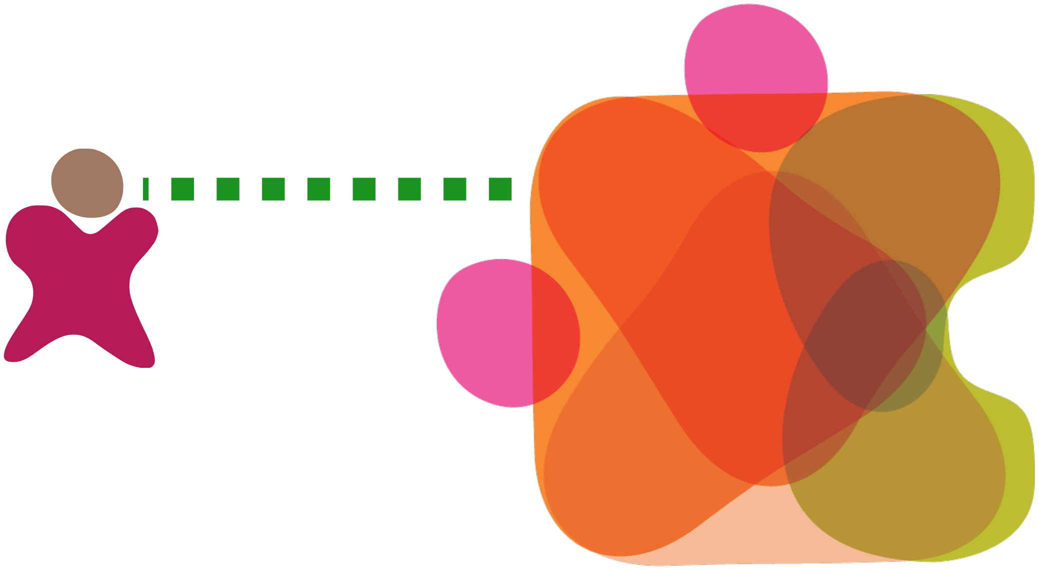
Design For Failure

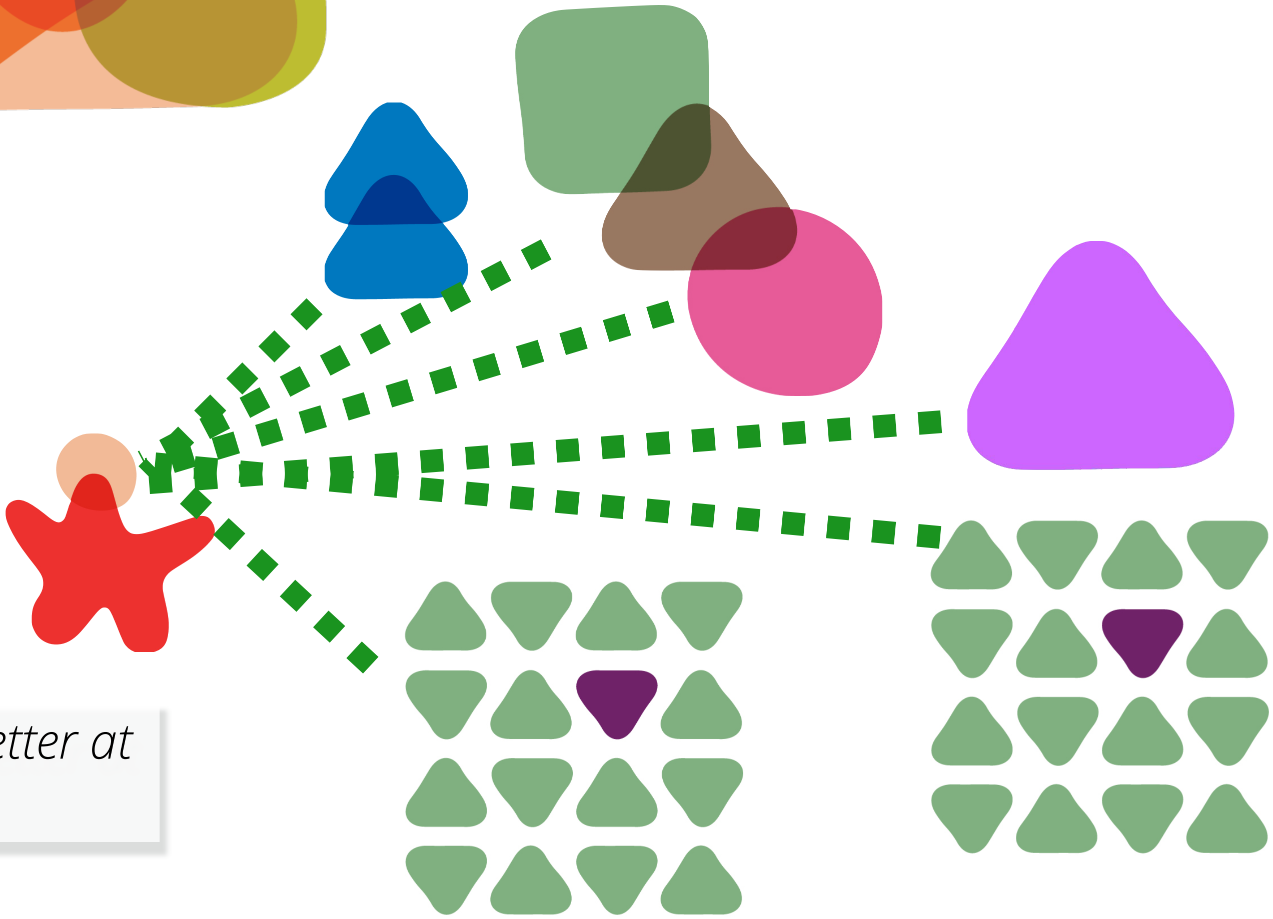
clients must respond gracefully to provider failure

aggressive monitoring:

- business relevant
- architectural
- semantic

Monitoring





O'REILLY

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS

Sam Newman

> You have to get ***much*** better at monitoring.

hon

logstash

logstash is a tool for managing events and l
and store them for later use (like, for search
with a web interface for searching and drilli

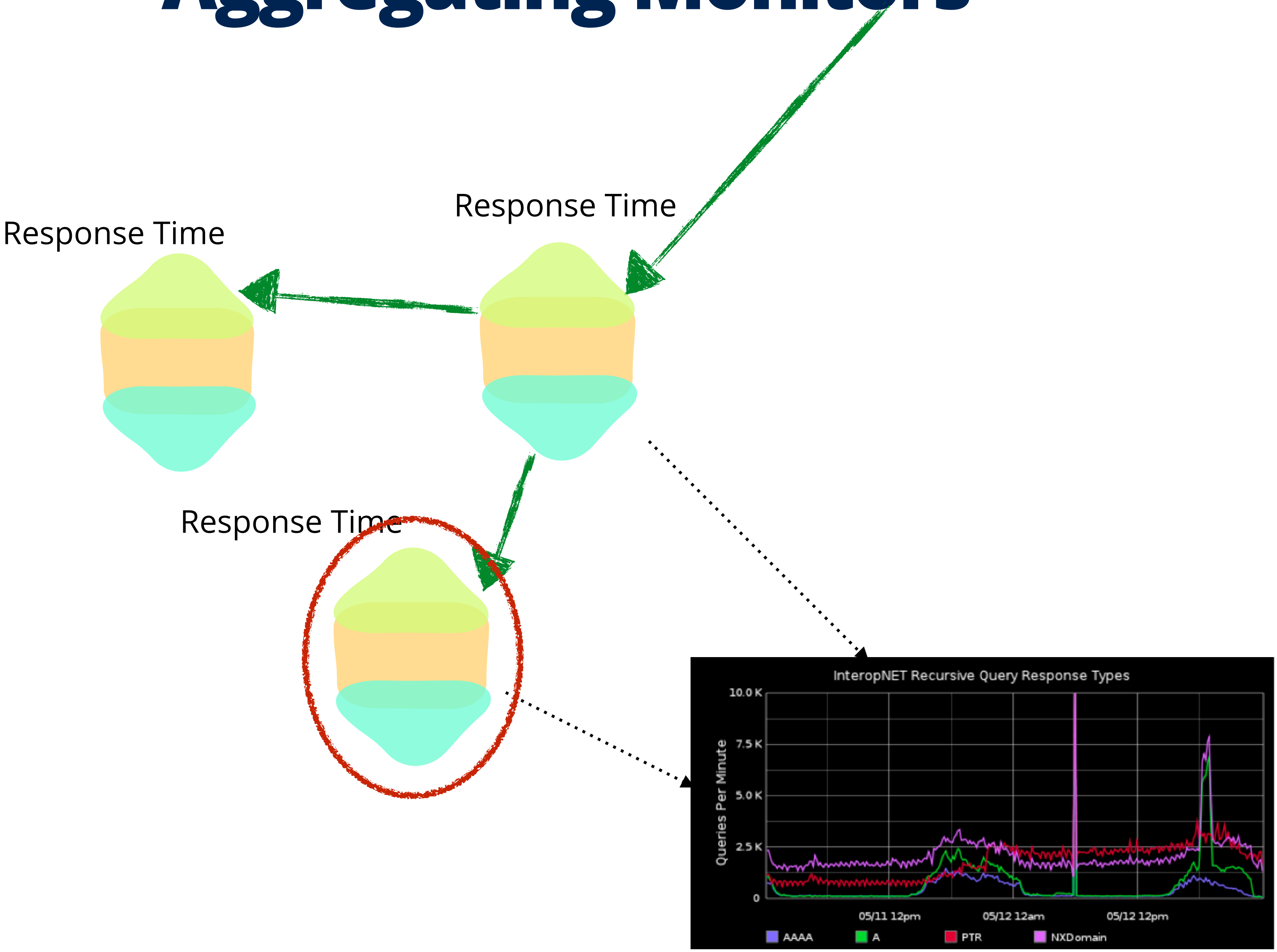
It is fully free and fully open source. The lice
much free to use it however you want in wh

Kibana

Make sense of a mountain of logs

Every event under one roof

Aggregating Monitors

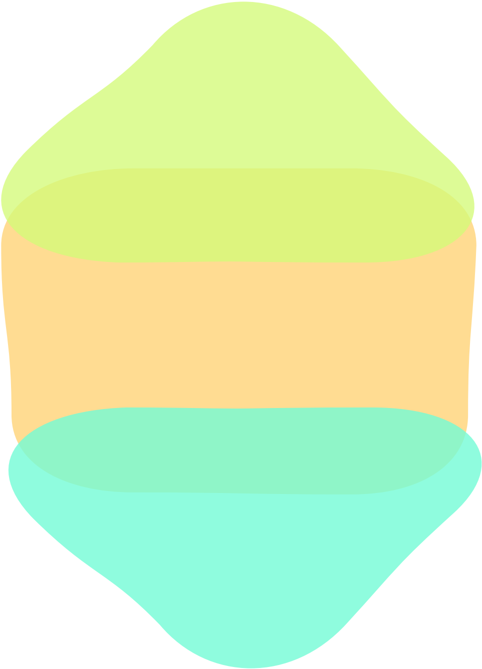


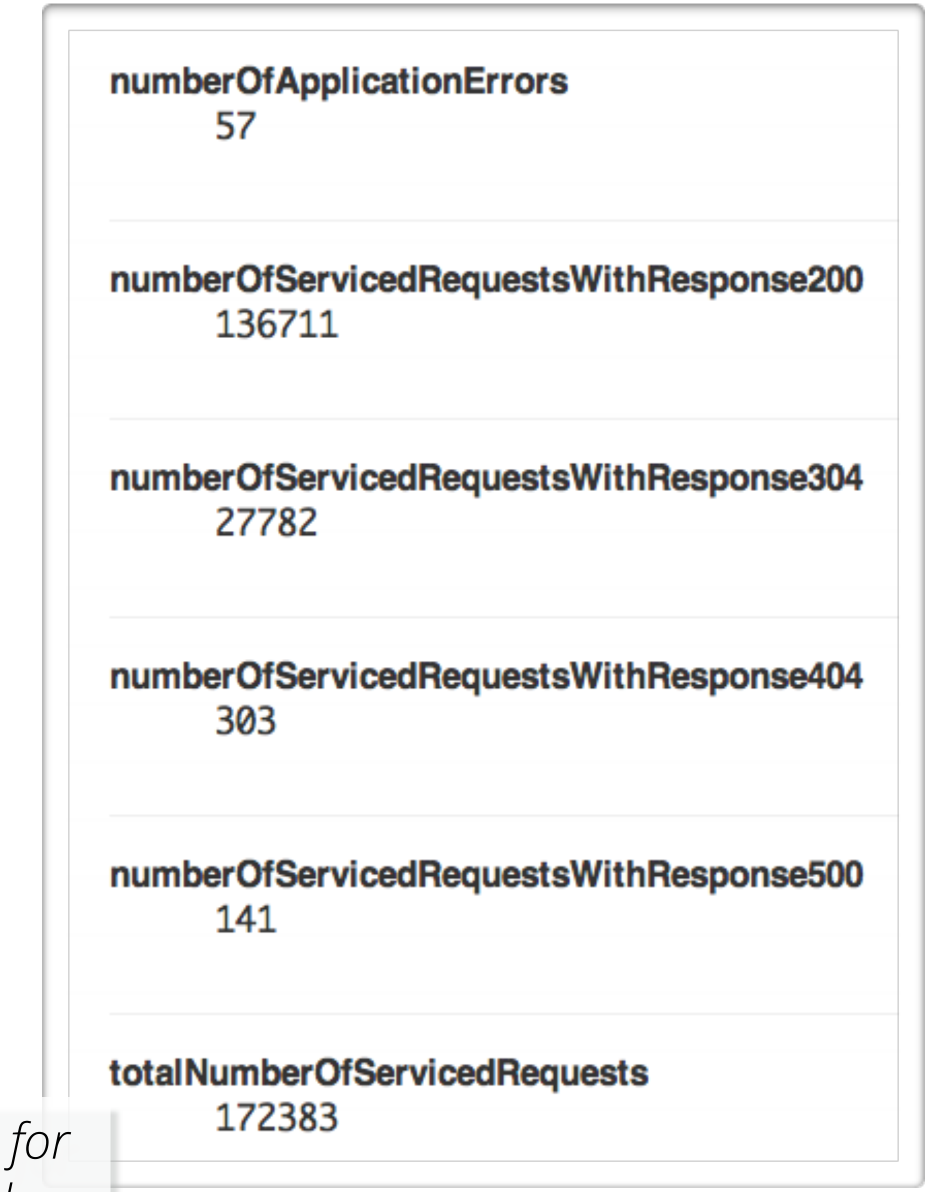
Response Time

Response Time

Response Time

Aggregating Monitors

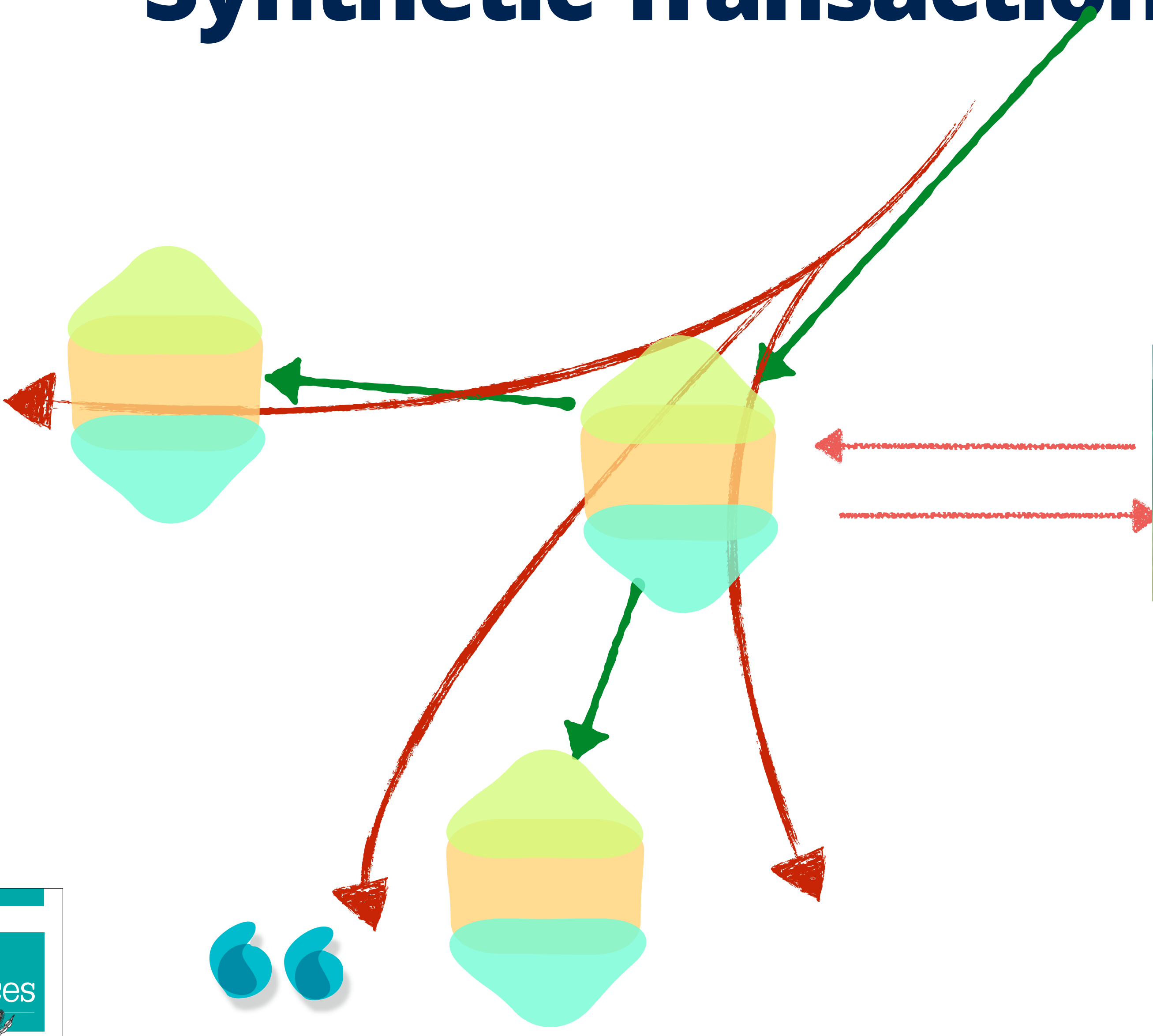


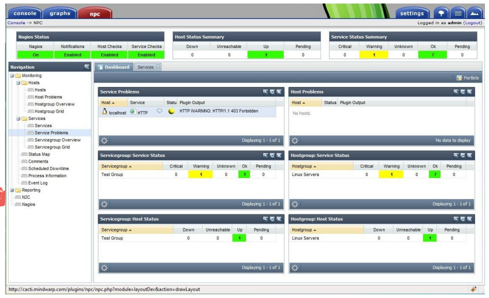


numberOfApplicationErrors	57
numberOfServicedRequestsWithResponse200	136711
numberOfServicedRequestsWithResponse304	27782
numberOfServicedRequestsWithResponse404	303
numberOfServicedRequestsWithResponse500	141
totalNumberOfServicedRequests	172383

Capture metrics, and logs, for each node, and aggregate them to get a rolled up picture.

Synthetic Transactions





Nagios Status

Nagios	Notifications	Host Checks	Service Checks
On	Enabled	Enabled	Enabled

Host Status Summary

Down	Unreachable	Up	Pending
0	0	1	0

Service Status Summary

Critical	Warning	Unknown	Ok	Pending
0	1	0	7	0

Service Problems

Host ▲	Service	Statu	Plugin Output
localhost	HTTP		HTTP WARNING: HTTP/1.1 403 Forbidden

Displaying 1 - 1 of 1

Host Problems

Host ▲	Status	Plugin Output
No hosts.		

No data to display

Servicegroup: Service Status

Servicegroup ▲	Critical	Warning	Unknown	Ok	Pending
Test Group	0	1	0	1	0

Displaying 1 - 1 of 1

Hostgroup: Service Status

Hostgroup ▲	Critical	Warning	Unknown	Ok	Pending
Linux Servers	0	1	0	7	0

Displaying 1 - 1 of 1

Servicegroup: Host Status

Servicegroup ▲	Down	Unreachable	Up	Pending
Test Group	0	0	1	0

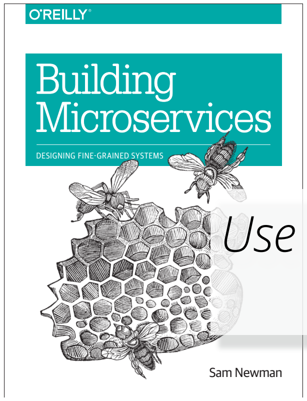
Displaying 1 - 1 of 1

Hostgroup: Host Status

Hostgroup ▲	Down	Unreachable	Up	Pending
Linux Servers	0	0	1	0

Displaying 1 - 1 of 1

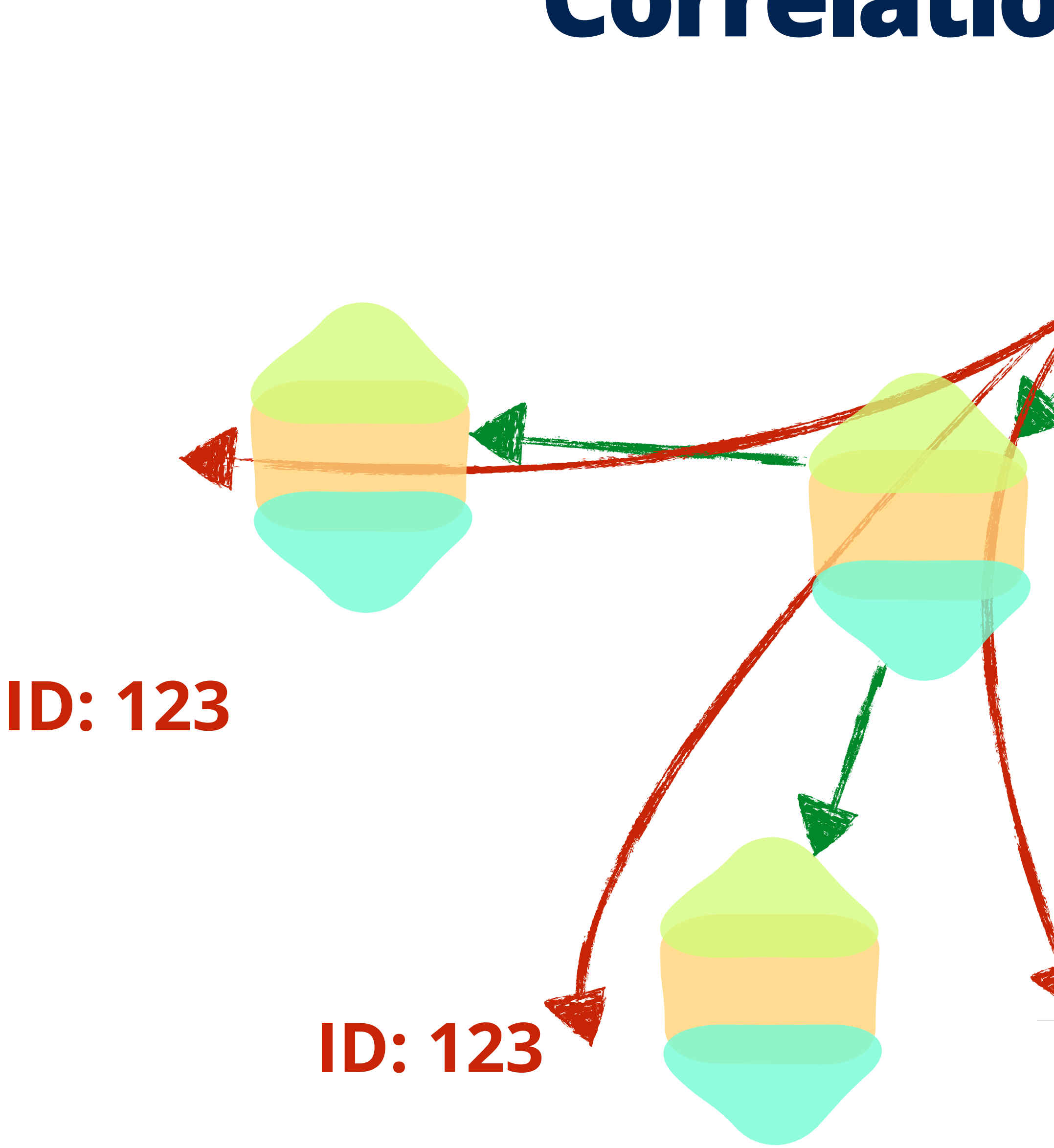
http://cacti.mindwarp.com/plugins/npc/npc.php?module=layoutDev&action=drawLayout

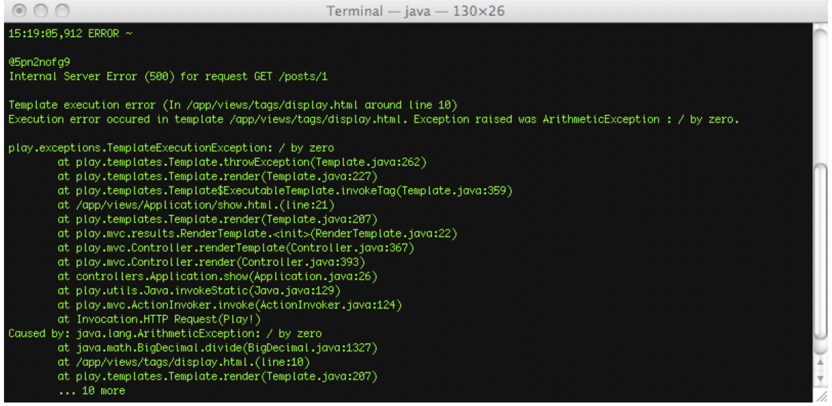


Use synthetic transactions to test production systems.

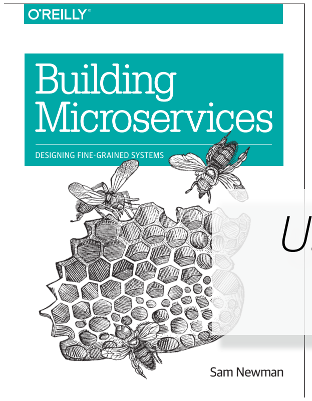
Correlation IDs

ID: 123



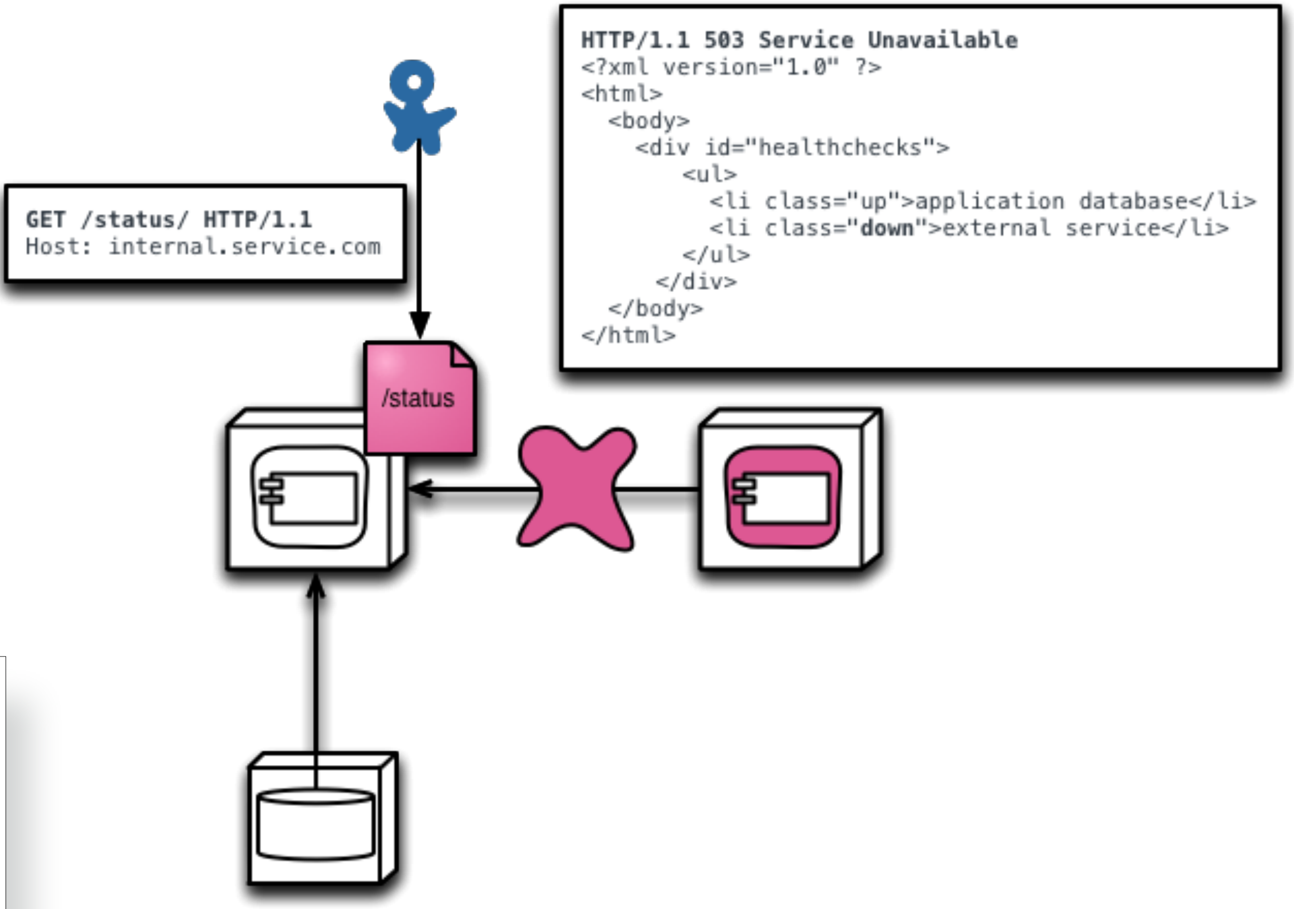


```
15:19:05,912 ERROR ~

@5pn2nofg9
Internal Server Error (500) for request GET /posts/1

Template execution error (In /app/views/tags/display.html around line 10)
Execution error occured in template /app/views/tags/display.html. Exception raised was ArithmeticException : / by zero.

play.exceptions.TemplateExecutionException: / by zero
        at play.templates.Template.throwException(Template.java:262)
        at play.templates.Template.render(Template.java:227)
        at play.templates.Template$ExecutableTemplate.invokeTag(Template.java:359)
        at /app/views/Application/show.html.(line:21)
        at play.templates.Template.render(Template.java:207)
        at play.mvc.results.RenderTemplate.<init>(RenderTemplate.java:22)
        at play.mvc.Controller.renderTemplate(Controller.java:367)
        at play.mvc.Controller.render(Controller.java:393)
        at controllers.Application.show(Application.java:26)
        at play.utils.Java.invokeStatic(Java.java:129)
        at play.mvc.ActionInvoker.invoke(ActionInvoker.java:124)
        at Invocation.HTTP Request(Play!)
Caused by: java.lang.ArithmeticException: / by zero
        at java.math.BigDecimal.divide(BigDecimal.java:1327)
        at /app/views/tags/display.html.(line:10)
        at play.templates.Template.render(Template.java:207)
        ... 10 more
```



“

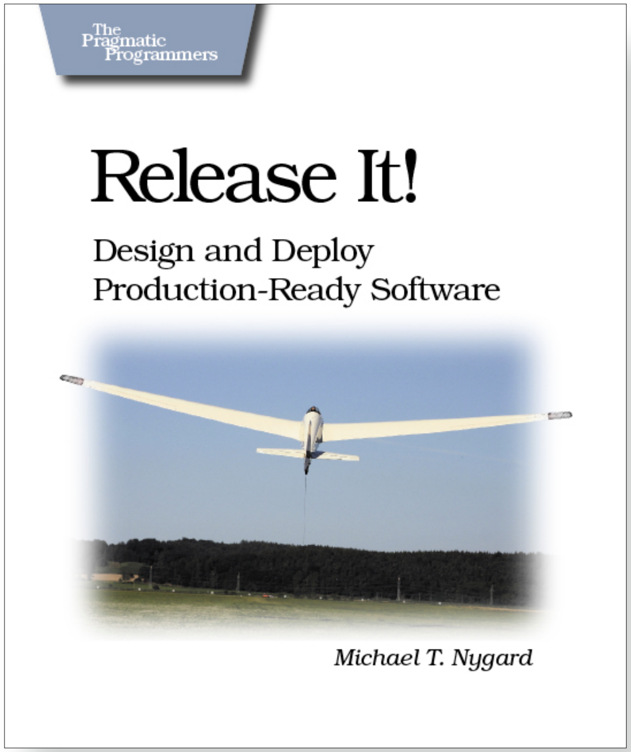
Use correlation IDs to track down nasty bugs



```
GET /status/ HTTP/1.1
Host: internal.service.com
```

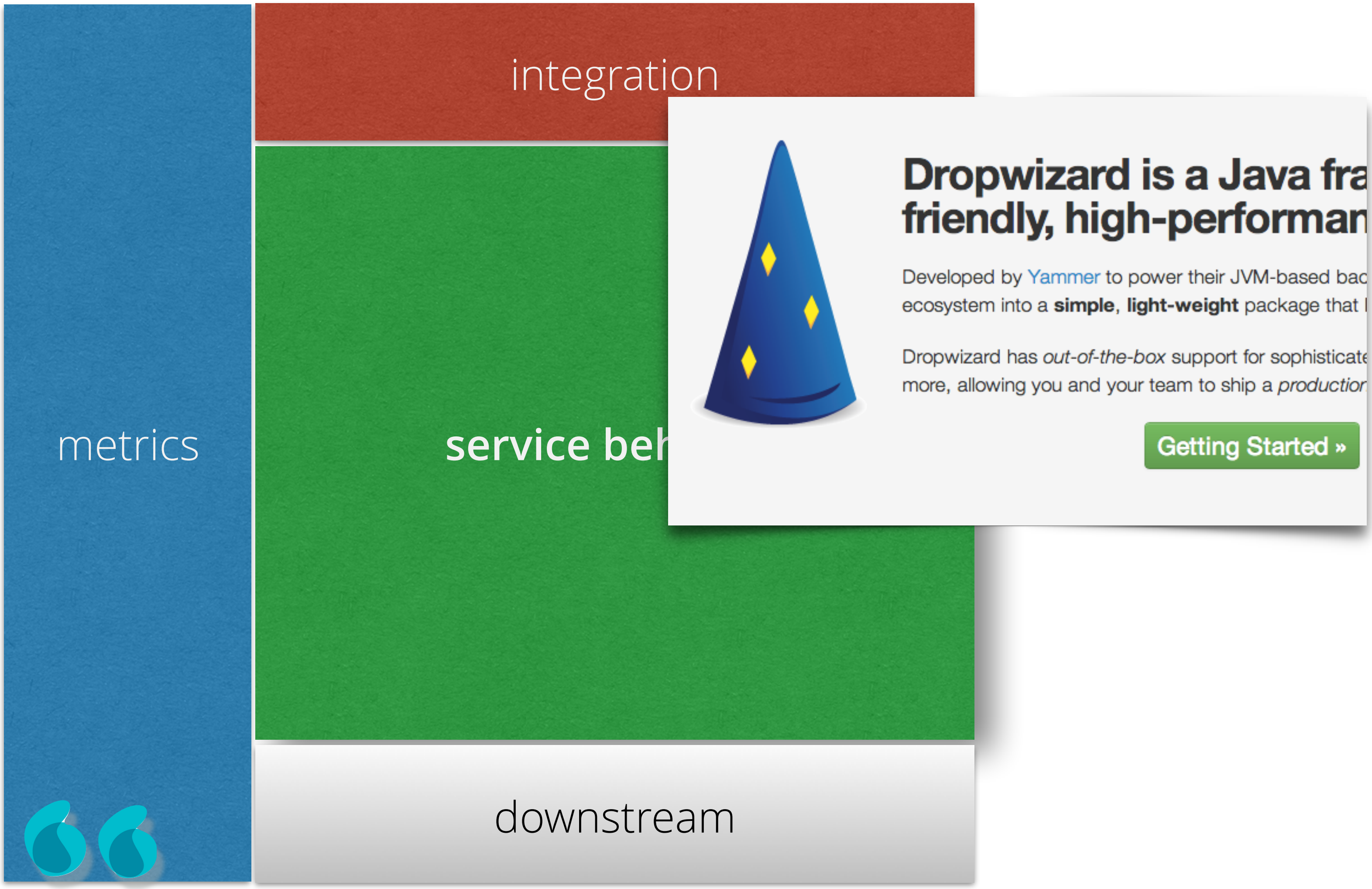
```
HTTP/1.1 503 Service Unavailable
<?xml version="1.0" ?>
<html>
  <body>
    <div id="healthchecks">
       <ul>
         <li class="up">application database</li>
         <li class="down">external service</li>
       </ul>
     </div>
  </body>
</html>
```

external service circuit breaker opened


The Pragmatic Programmers
Release It!
Design and Deploy Production-Ready Software
Michael T. Nygard

> *Use timeouts, circuit breakers and bulk-heads to avoid cascading failure.*

Engineering Consistency



Dropwizard is a Java fra
friendly, high-performan

Developed by Yammer to power their JVM-based bac
ecosystem into a **simple**, **light-weight** package that l

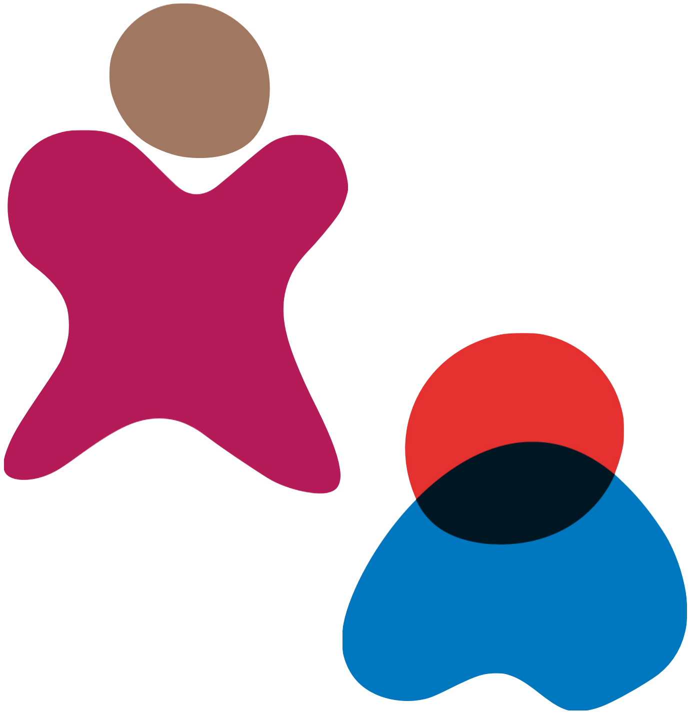
Dropwizard has *out-of-the-box* support for sophisticate
more, allowing you and your team to ship a *productior*

> *Consider Service Templates to make it easy to do the right thing!*

Orchestration

Orchestration describes the automated arrangement, coordination, and management of complex computer systems, middleware, and services.

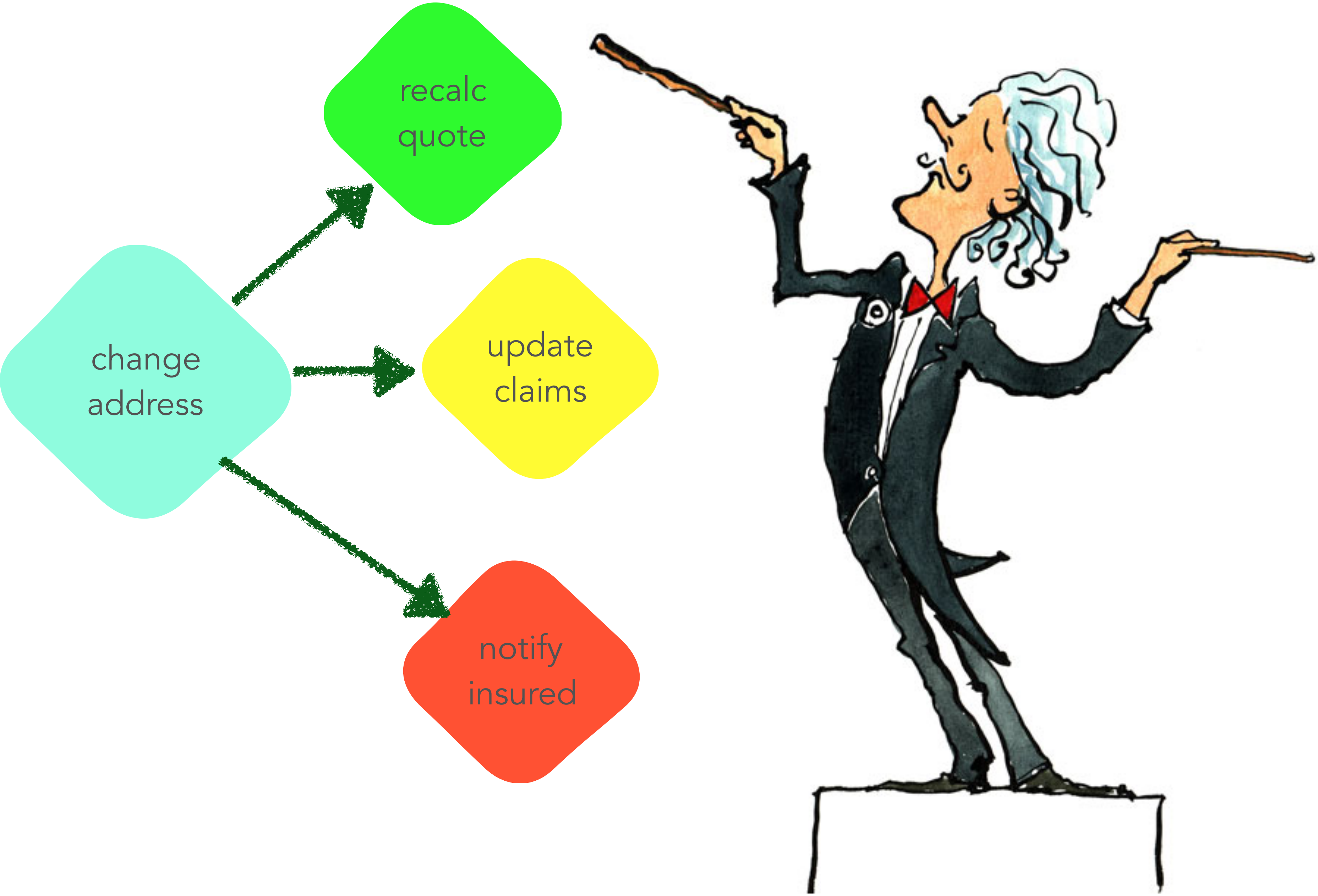
http://en.wikipedia.org/wiki/Orchestration_(computing)



choreography vs. orchestration

in microservices

Orchestration



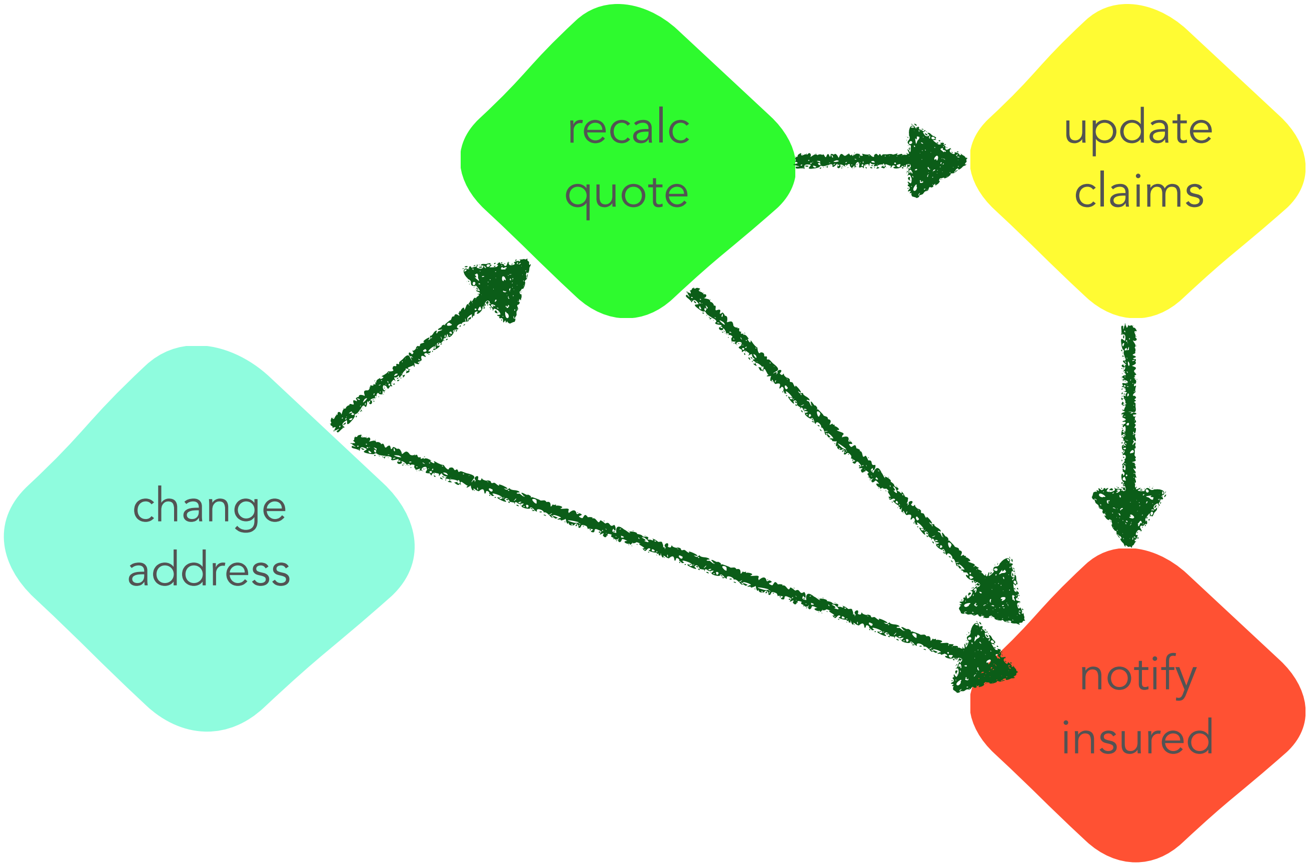
change address

recalc quote

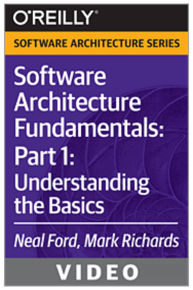
update claims

notify insured

Choreography



mediator versus broker topology


O'REILLY

SOFTWARE ARCHITECTURE SERIES

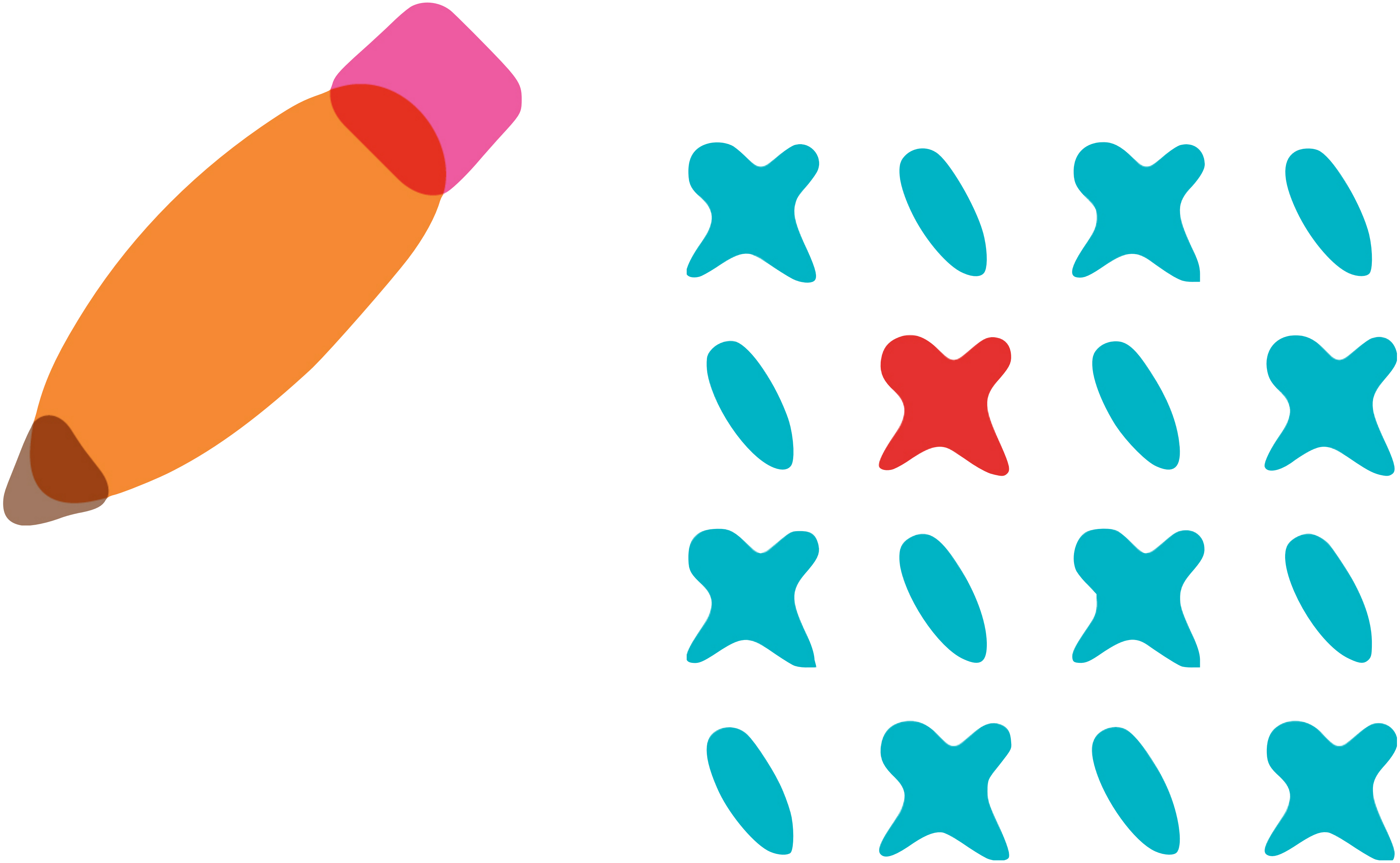
Software Architecture Fundamentals: Part 1: Understanding the Basics

Neal Ford, Mark Richards

VIDEO

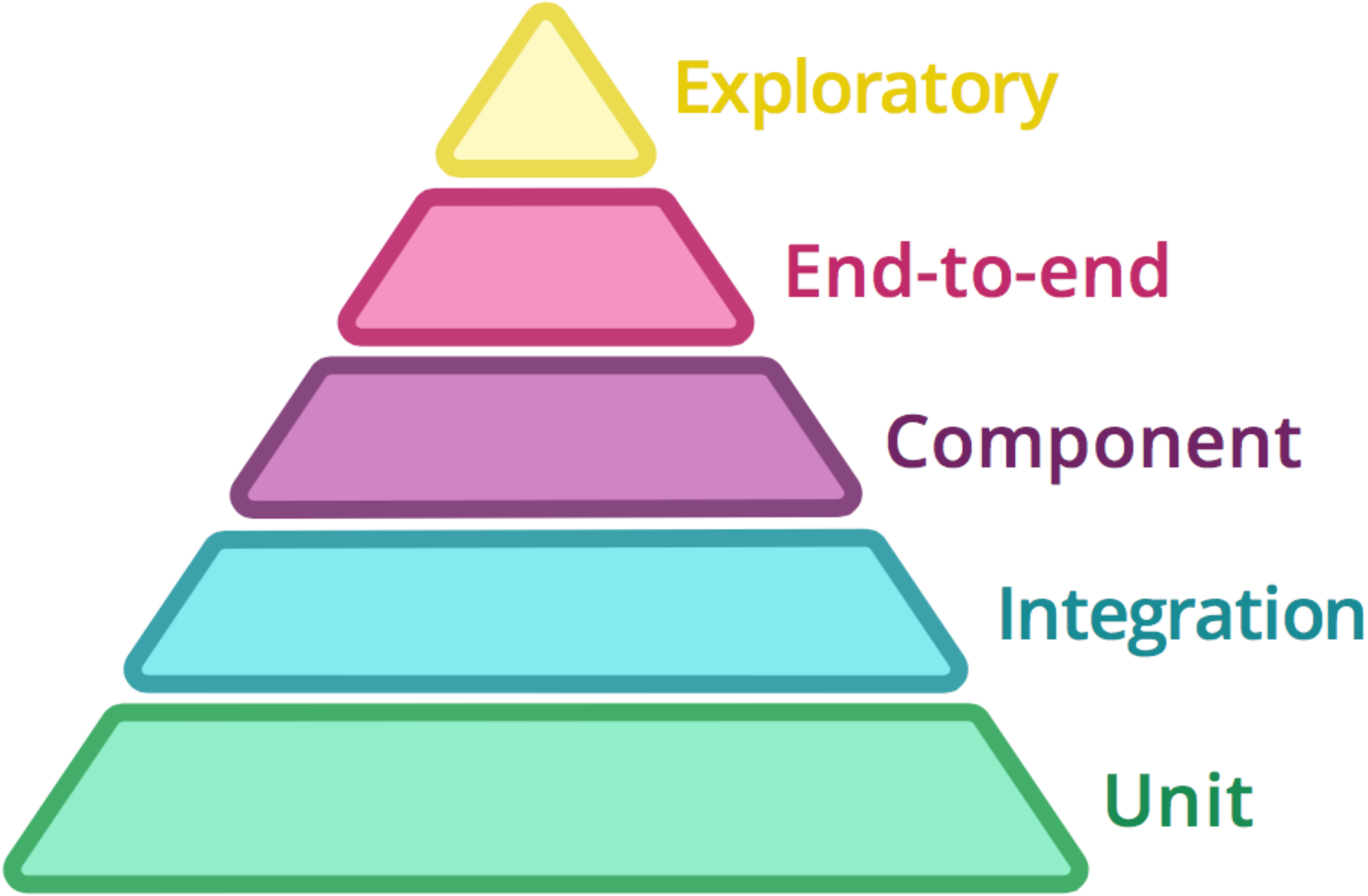
http://shop.oreilly.com/product/110000195.do

Testing Microservices

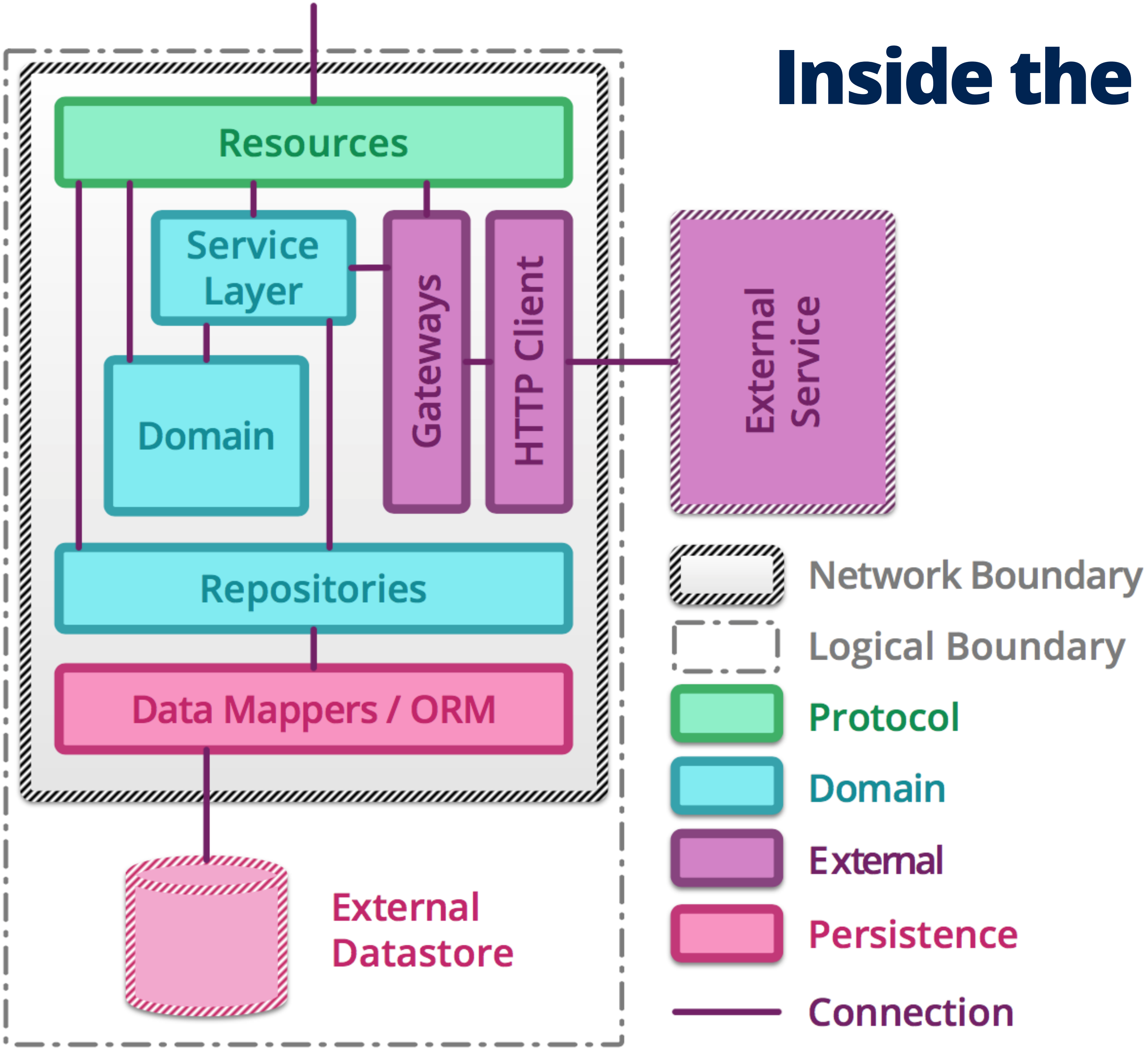


http://martinfowler.com/articles/microservice-testing/

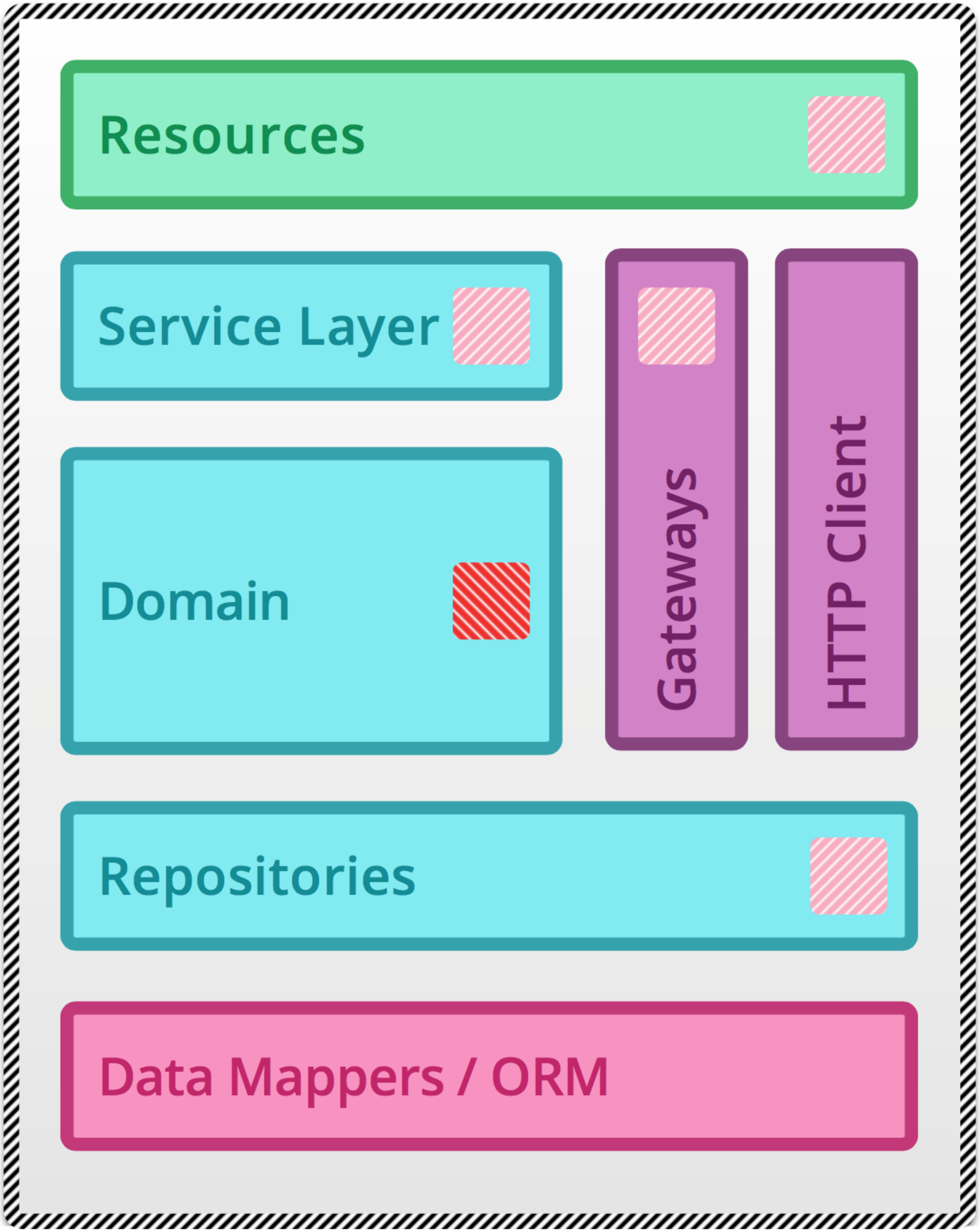
Test Pyramid for Microservices



Inside the Box



Unit Testing



Resources

Service Layer

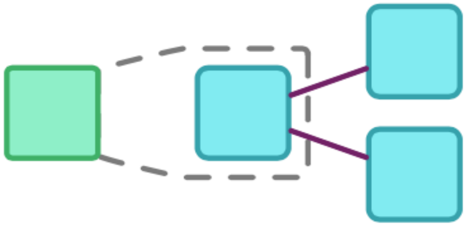
Gateways

HTTP Client

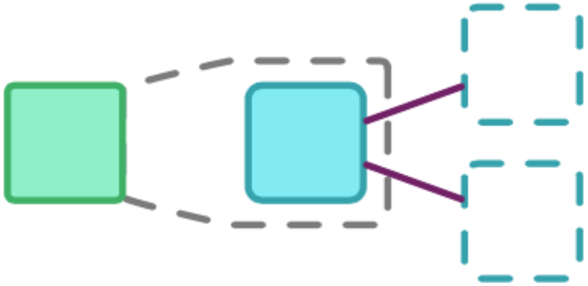
Domain

Repositories

Data Mappers / ORM



Sociable

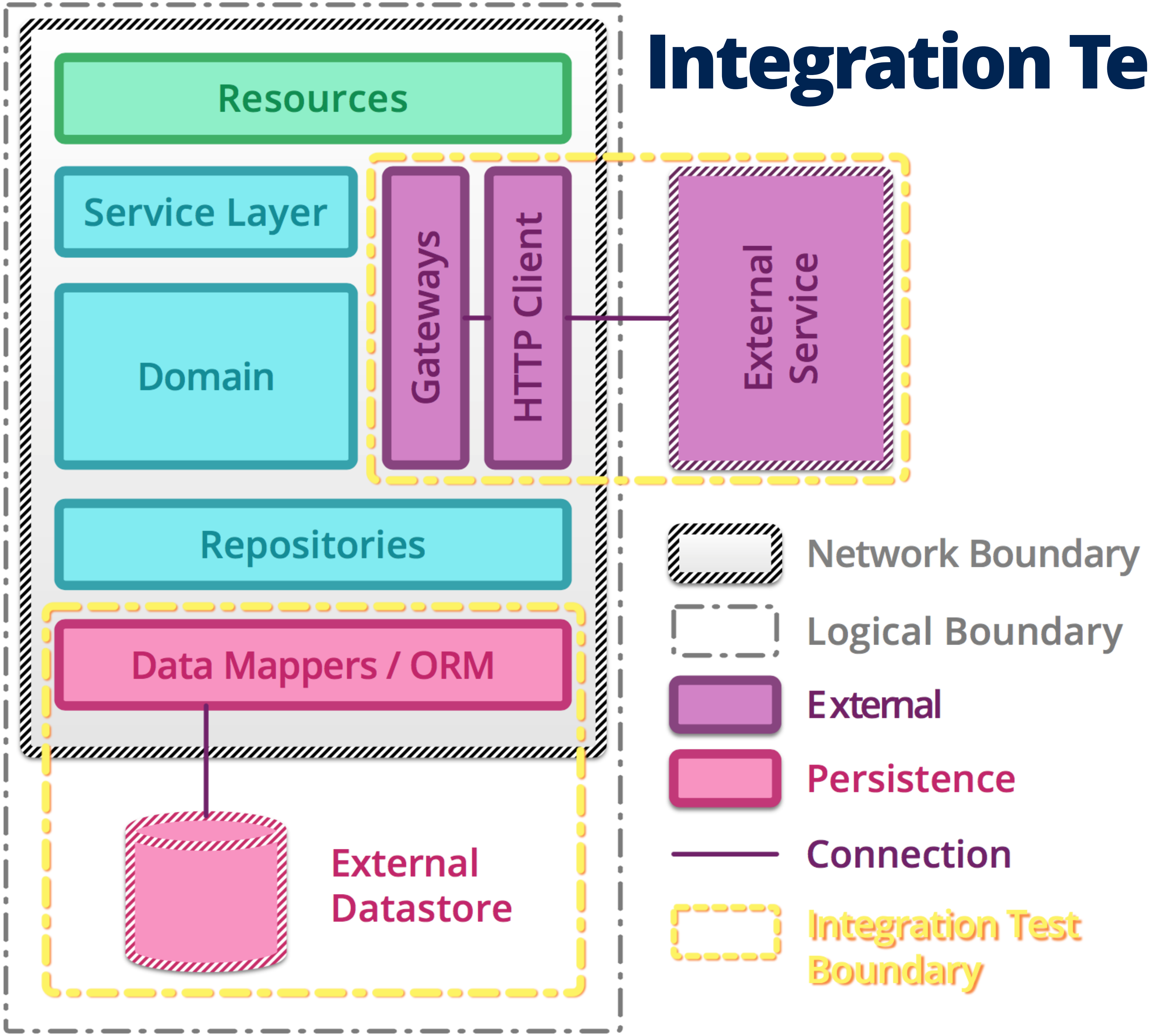


Solitary

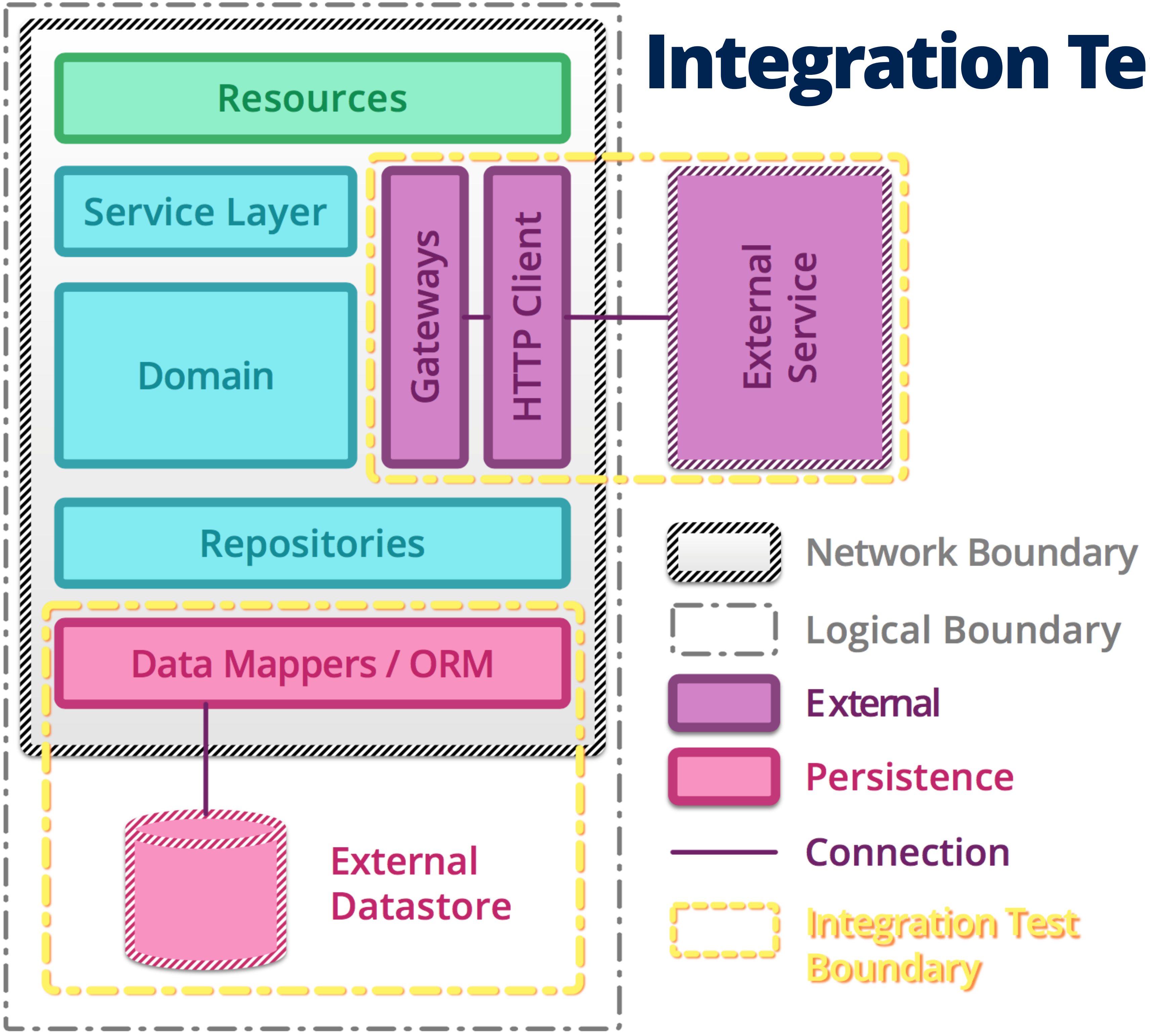
▨ Unit - Solitary

▨ Unit - Sociable

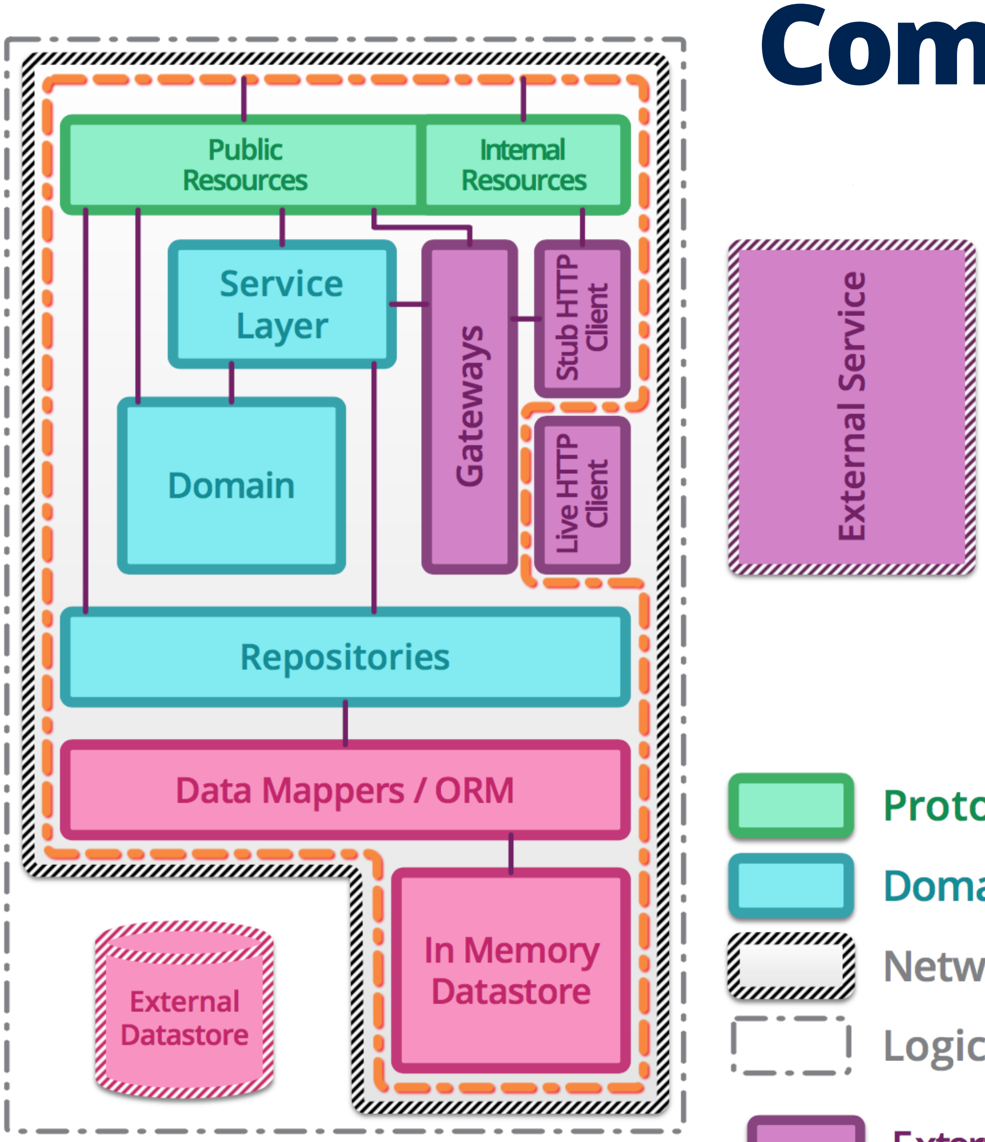
Integration Testing



Integration Testing

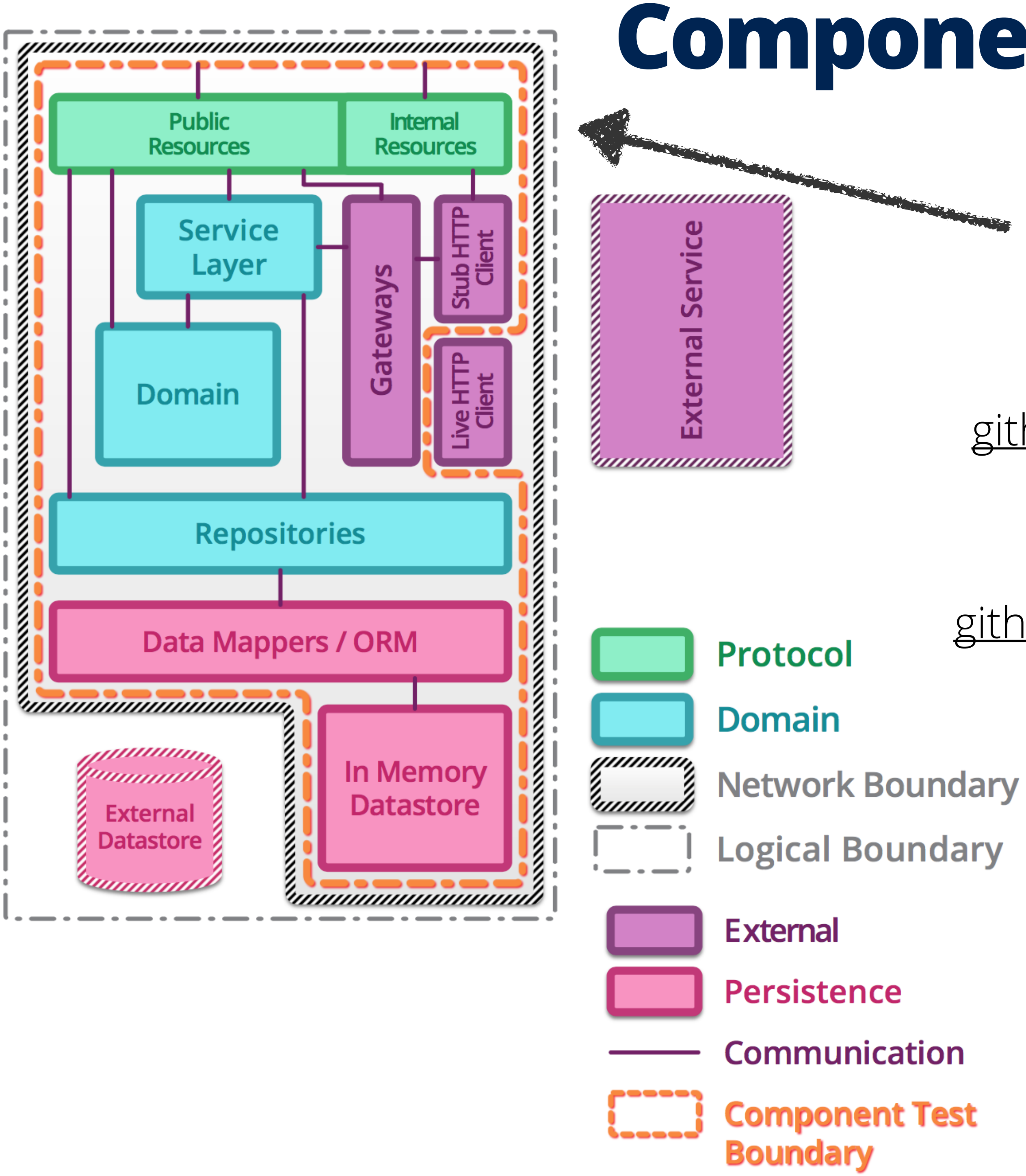


Component Testing



- 🟩 Protocol
- 🟦 Domain
- ▨ Network Boundary
- ⬚ Logical Boundary
- 🟪 External
- 🟥 Persistence
- — Communication
- ⬚ Component Test Boundary

Component Testing



shims:

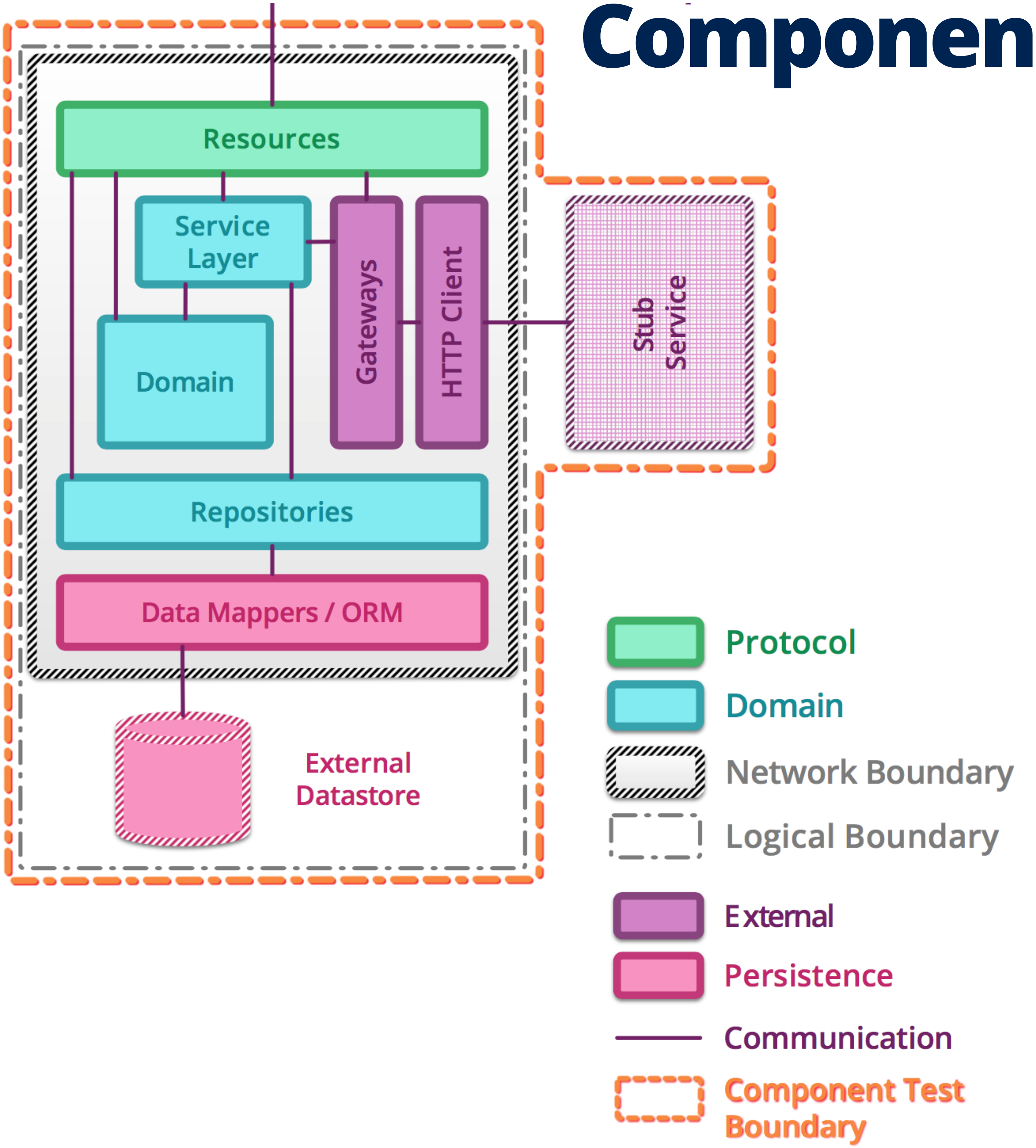
inproctester

github.com/aharin/inproctester

Plasma

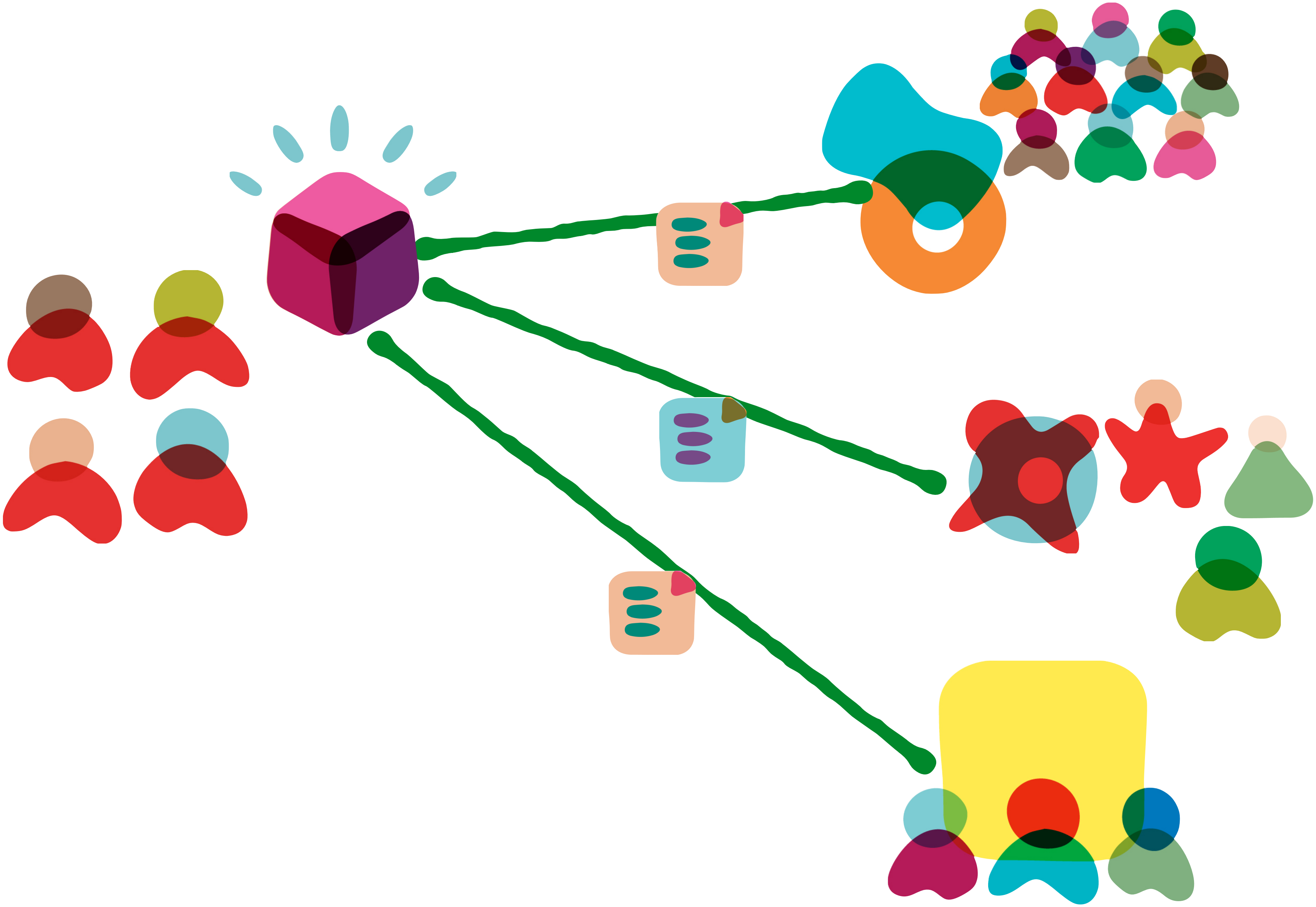
github.com/jennifersmith/plasma

Component Testing

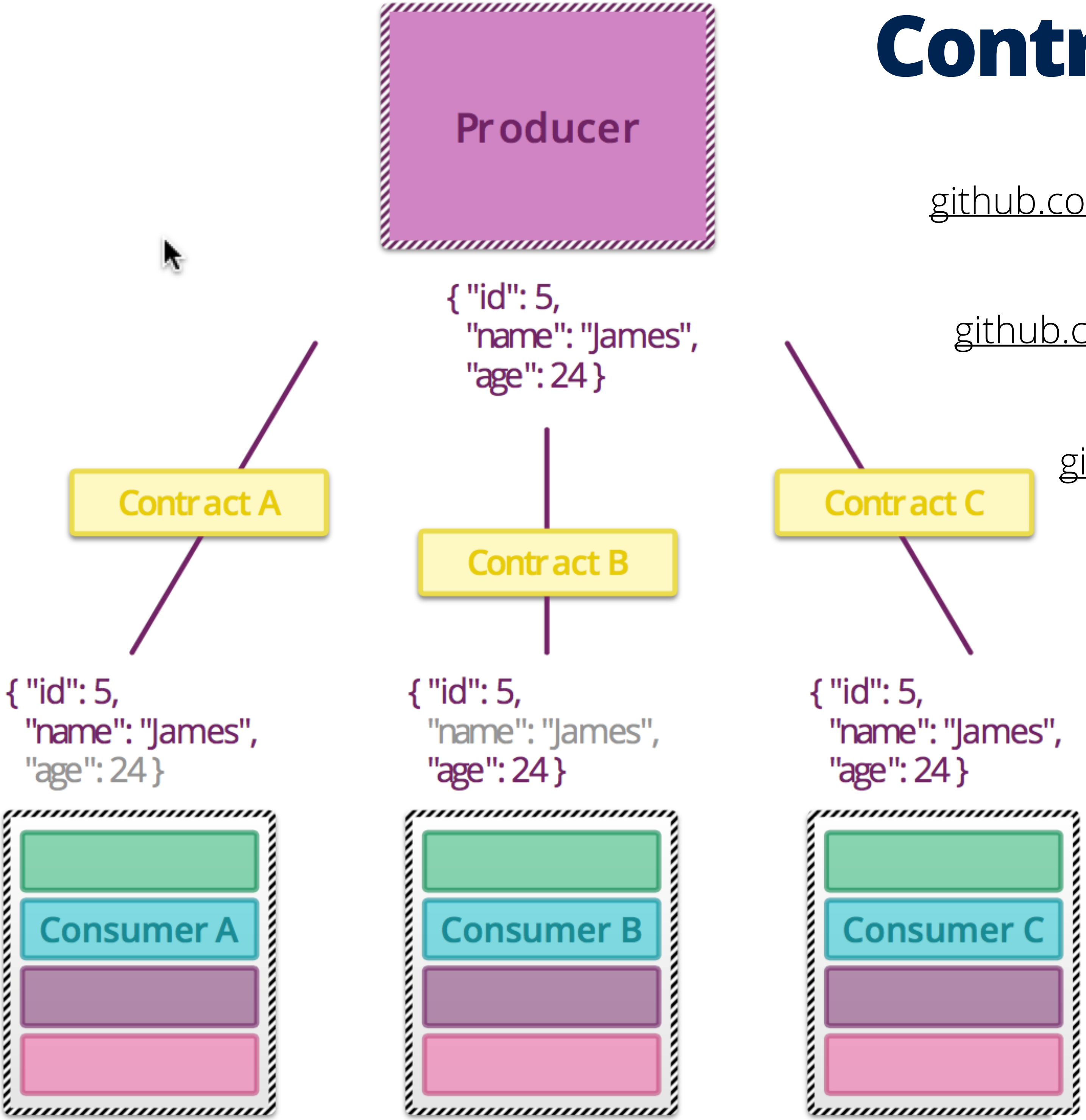


Consumer Driven Contracts

http://martinfowler.com/articles/consumerDrivenContracts.html



Contract Testing



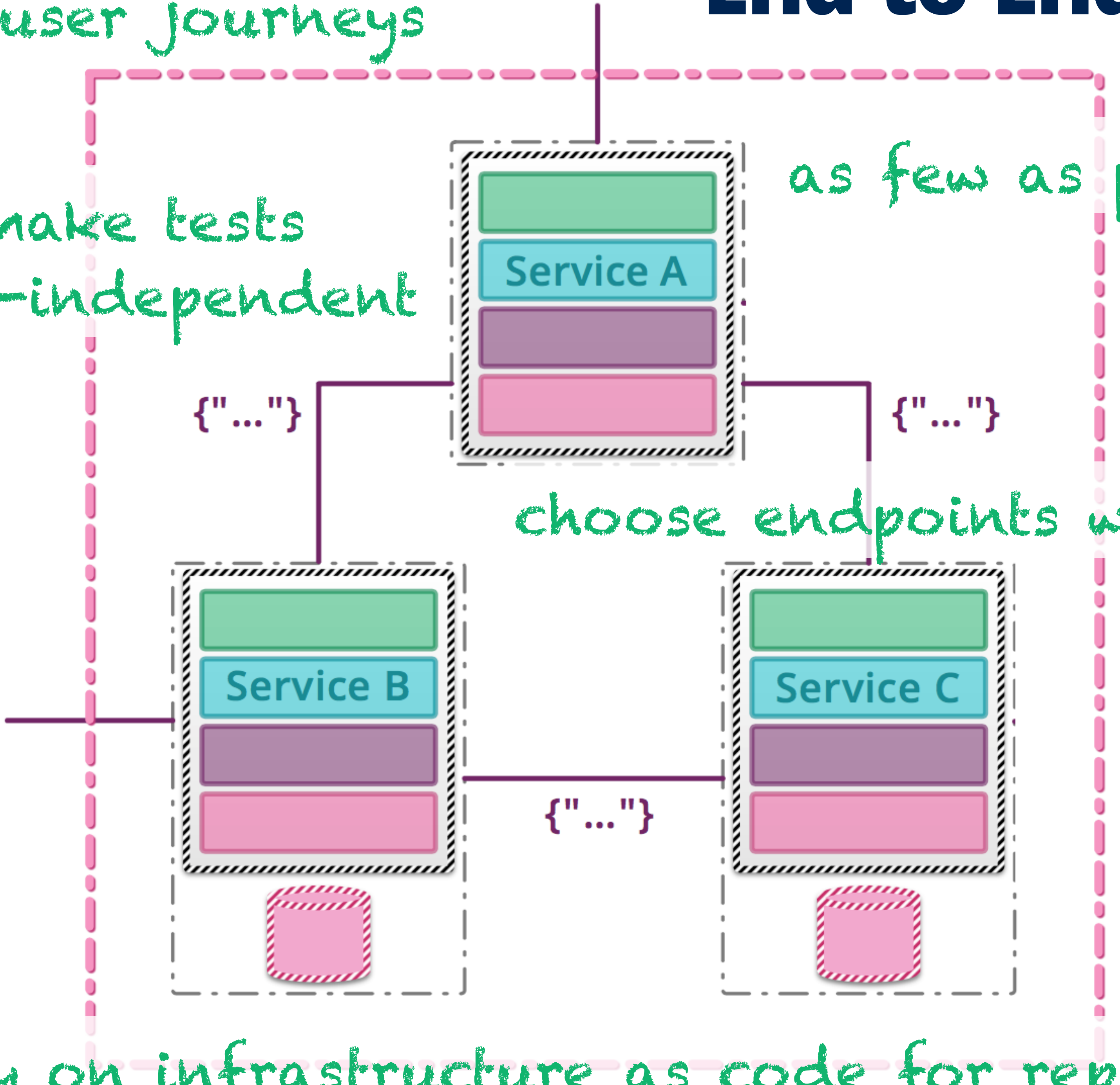
Pact
github.com/realestate-com-au/pact

Pacto
github.com/thoughtworks/pacto

Janus
github.com/gga/janus

End-to-End Testing

focus on personas
& user journeys



as few as possible

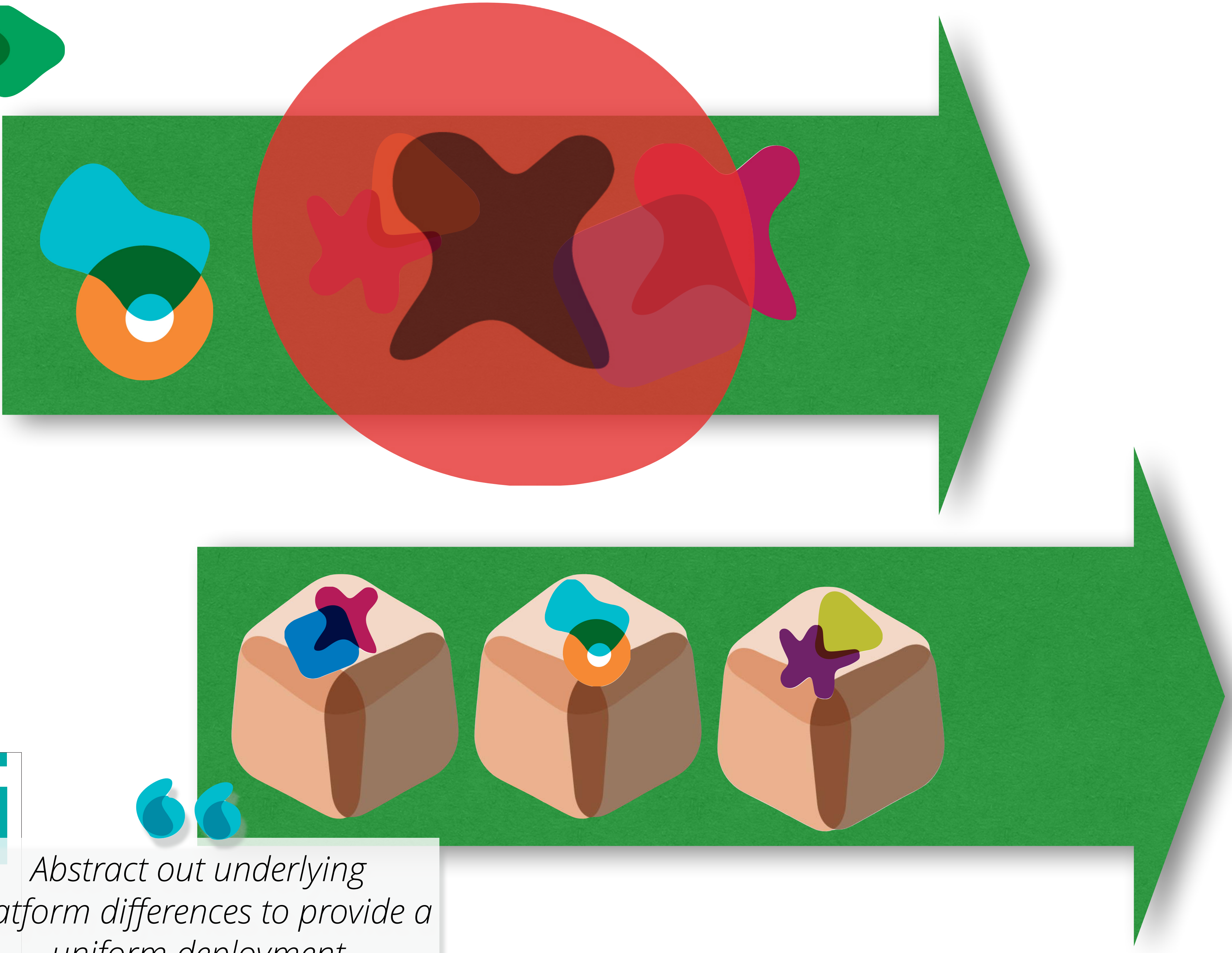
make tests
data-independent

choose endpoints wisely

rely on infrastructure as code for repeatability

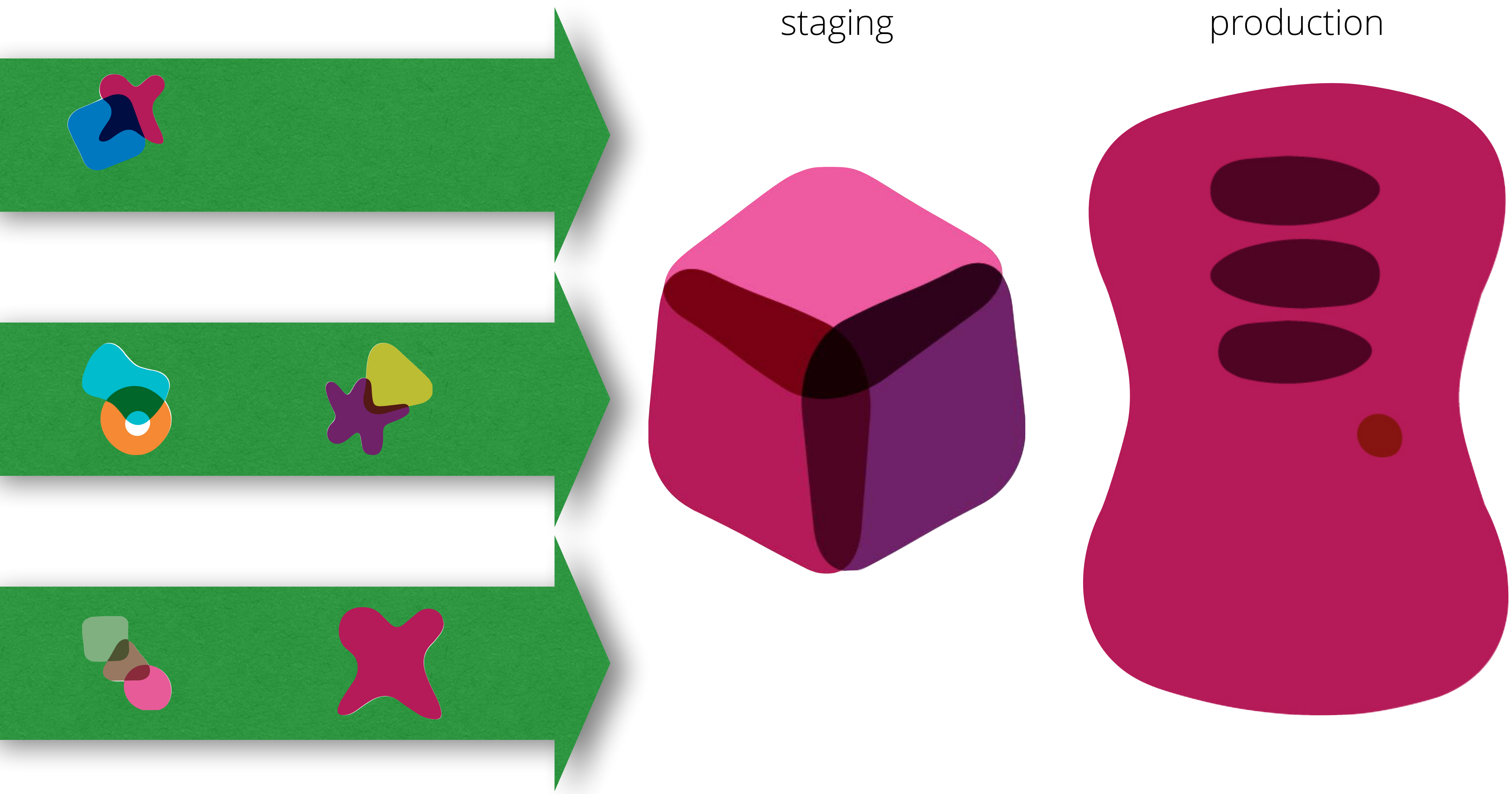
Deployment

engineering

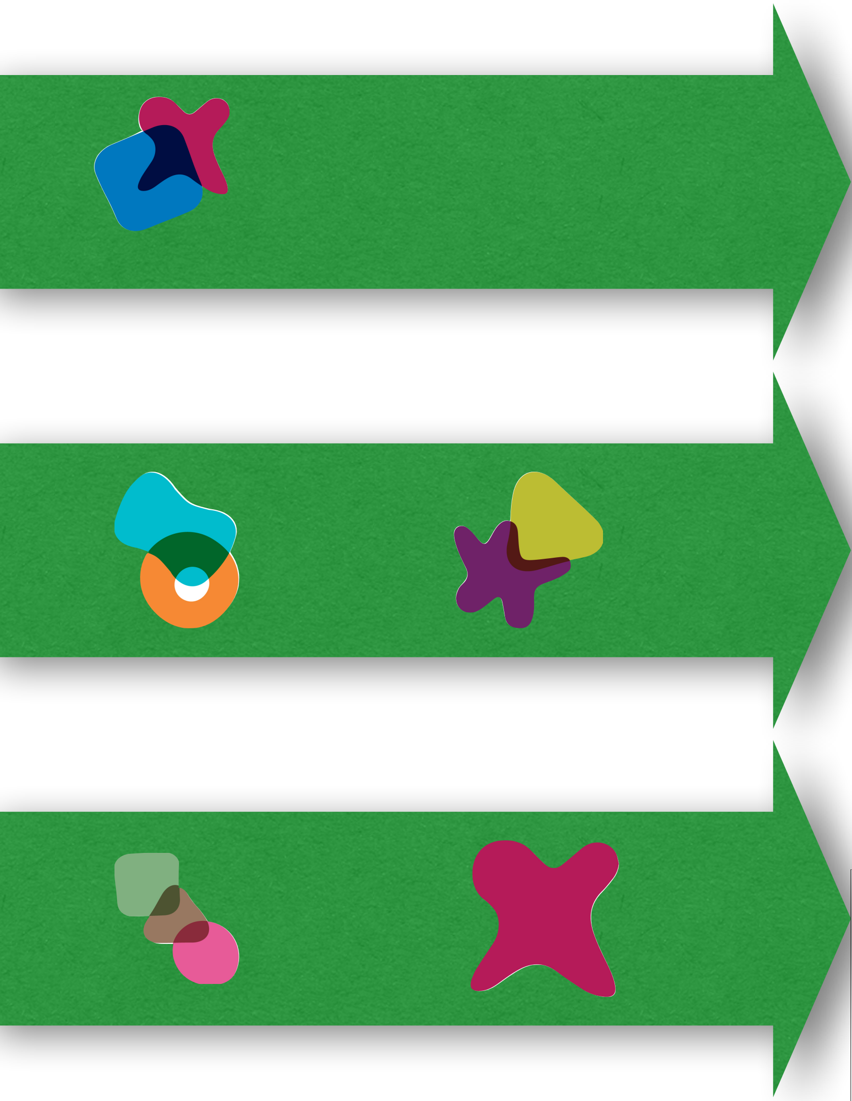


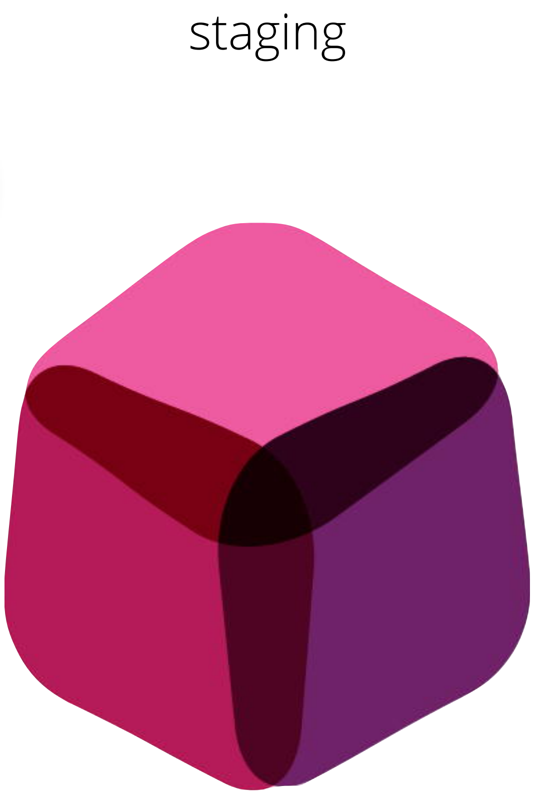
> Abstract out underlying platform differences to provide a uniform deployment mechanism.

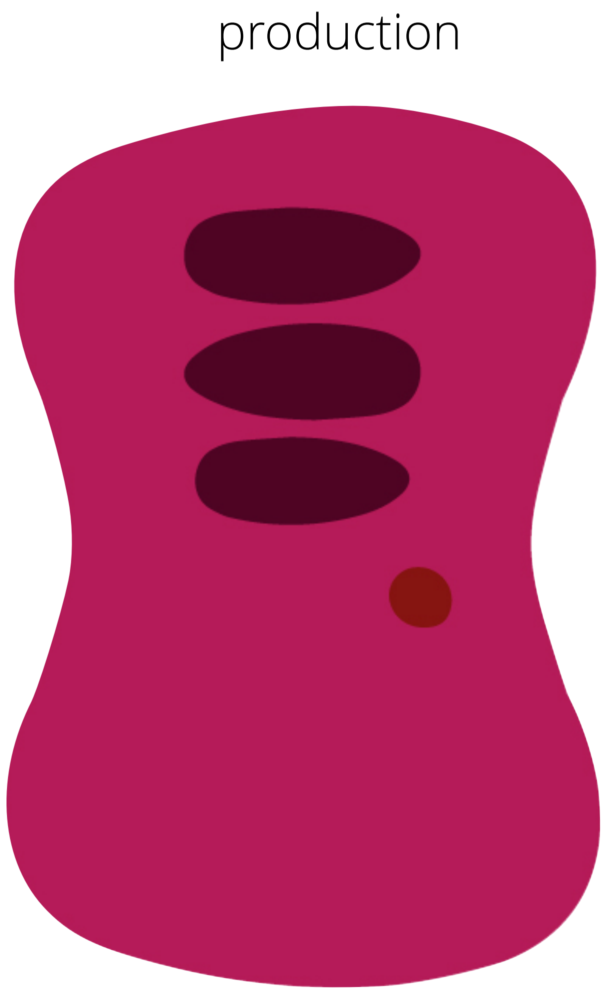
Don’t Let Changes Build Up



Don’t Let Changes Build Up




staging


production

O'REILLY

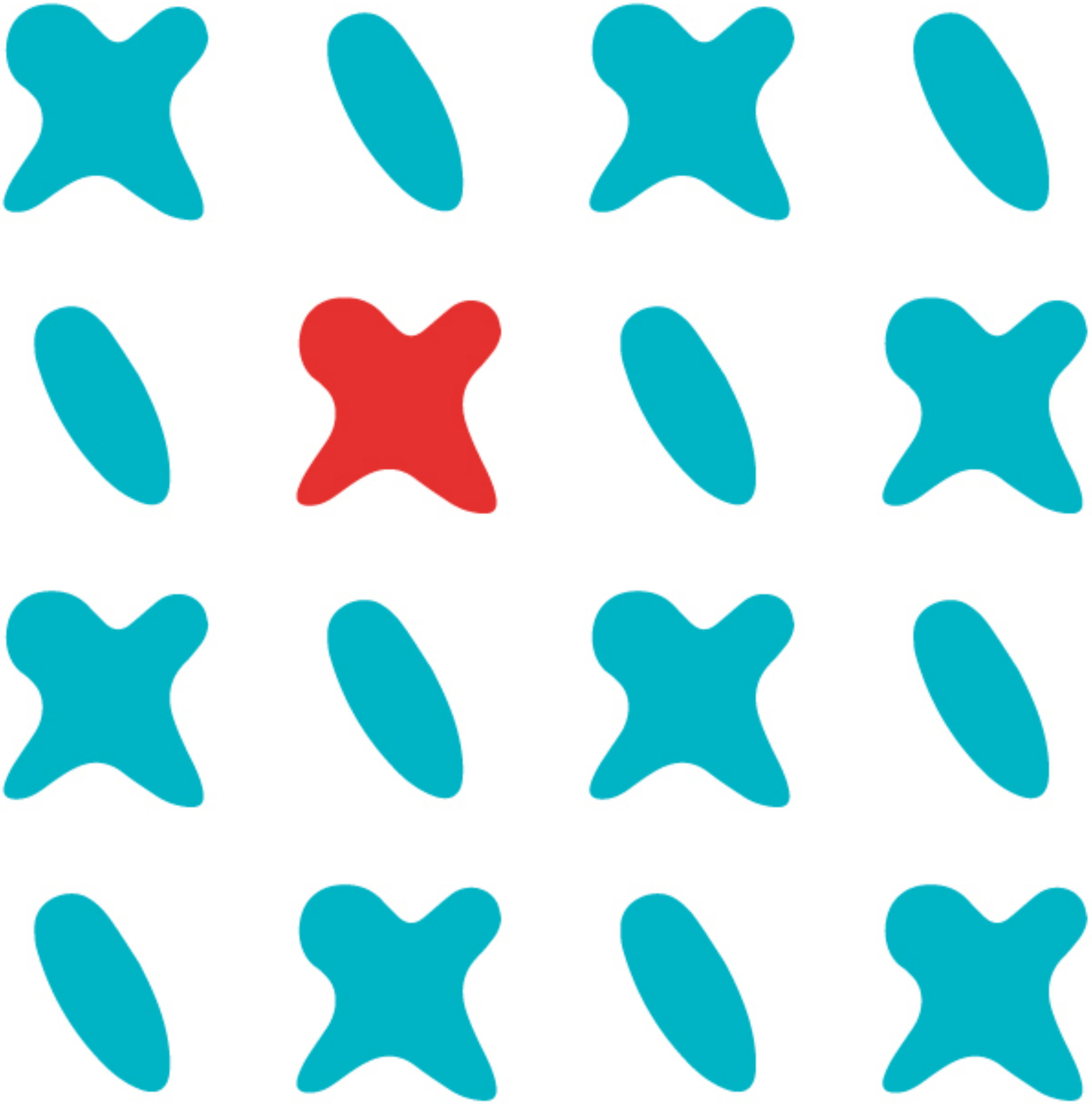
Building Microservices

DESIGNING FINE-GRAINED SYSTEMS

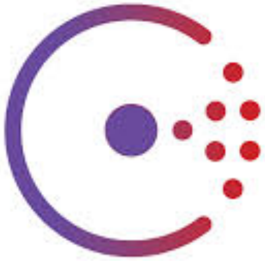
Sam Newman

> *Don’t let changes build up - release as soon as you can, and preferably one at a time!*

Service Discovery



Dynamic Service Registries



https://consul.io/

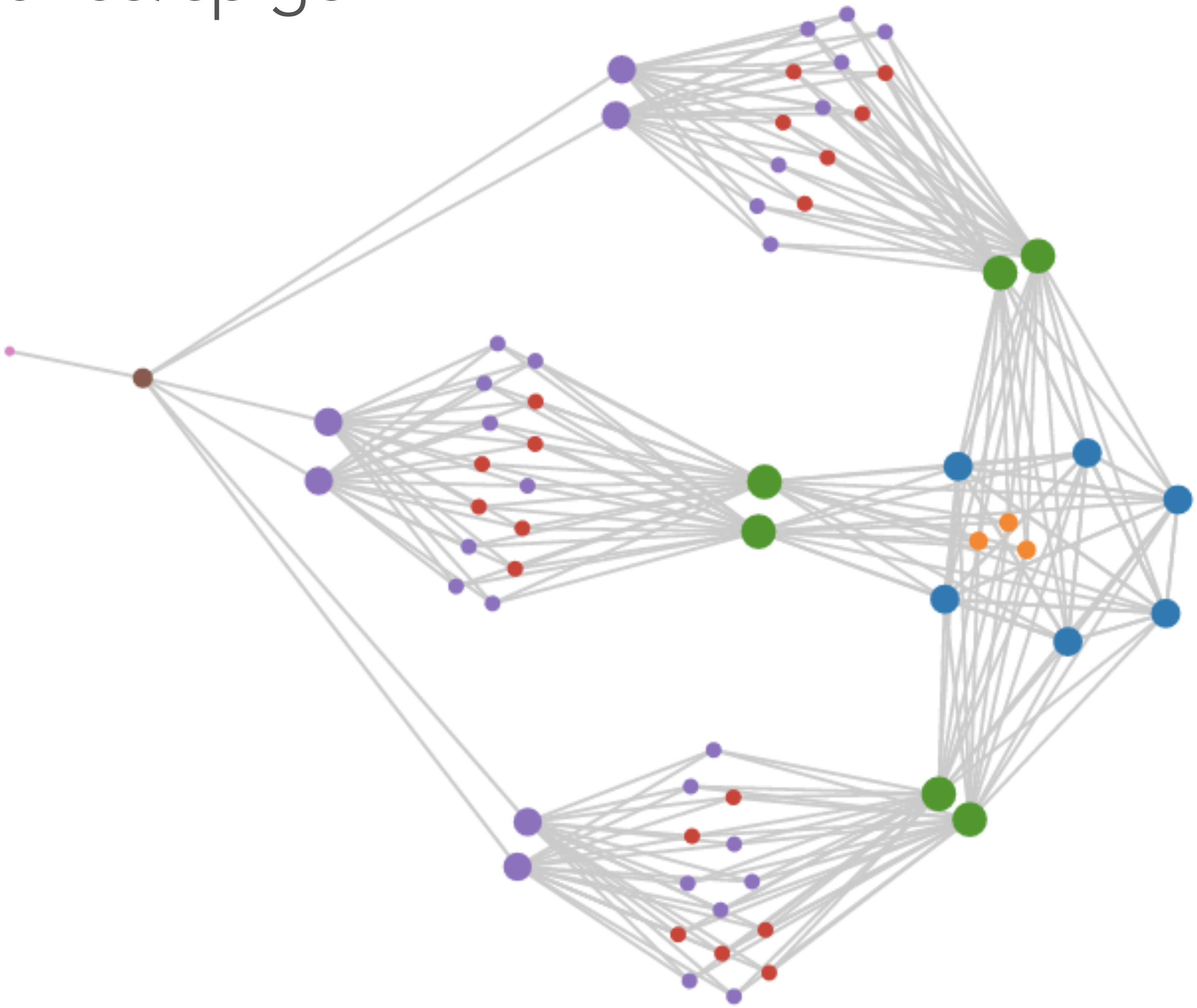
http://zookeeper.apache.org

https://coreos.com/etcd/

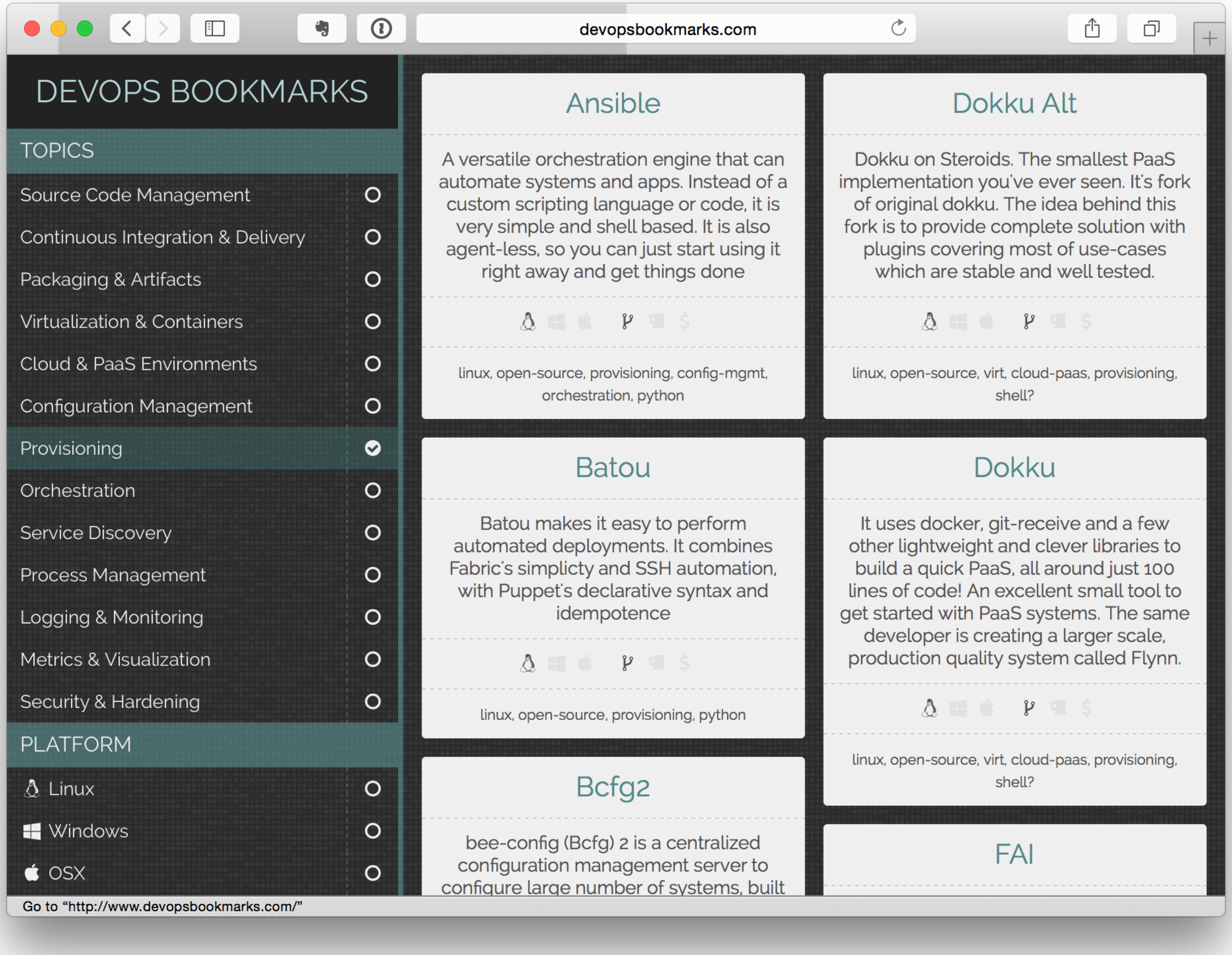
Service Visualization

adrianco / **spigo**

https://github.com/adrianco/spigo



Tools



DEVOPS BOOKMARKS

TOPICS

- Source Code Management ○
- Continuous Integration & Delivery ○
- Packaging & Artifacts ○
- Virtualization & Containers ○
- Cloud & PaaS Environments ○
- Configuration Management ○
- Provisioning ✓
- Orchestration ○
- Service Discovery ○
- Process Management ○
- Logging & Monitoring ○
- Metrics & Visualization ○
- Security & Hardening ○

PLATFORM

- Linux ○
- Windows ○
- OSX ○

Ansible

A versatile orchestration engine that can automate systems and apps. Instead of a custom scripting language or code, it is very simple and shell based. It is also agent-less, so you can just start using it right away and get things done

linux, open-source, provisioning, config-mgmt, orchestration, python

Dokku Alt

Dokku on Steroids. The smallest PaaS implementation you've ever seen. It's fork of original dokku. The idea behind this fork is to provide complete solution with plugins covering most of use-cases which are stable and well tested.

linux, open-source, virt, cloud-paas, provisioning, shell?

Batou

Batou makes it easy to perform automated deployments. It combines Fabric's simplicty and SSH automation, with Puppet's declarative syntax and idempotence

linux, open-source, provisioning, python

Dokku

It uses docker, git-receive and a few other lightweight and clever libraries to build a quick PaaS, all around just 100 lines of code! An excellent small tool to get started with PaaS systems. The same developer is creating a larger scale, production quality system called Flynn.

linux, open-source, virt, cloud-paas, provisioning, shell?

Bcfg2

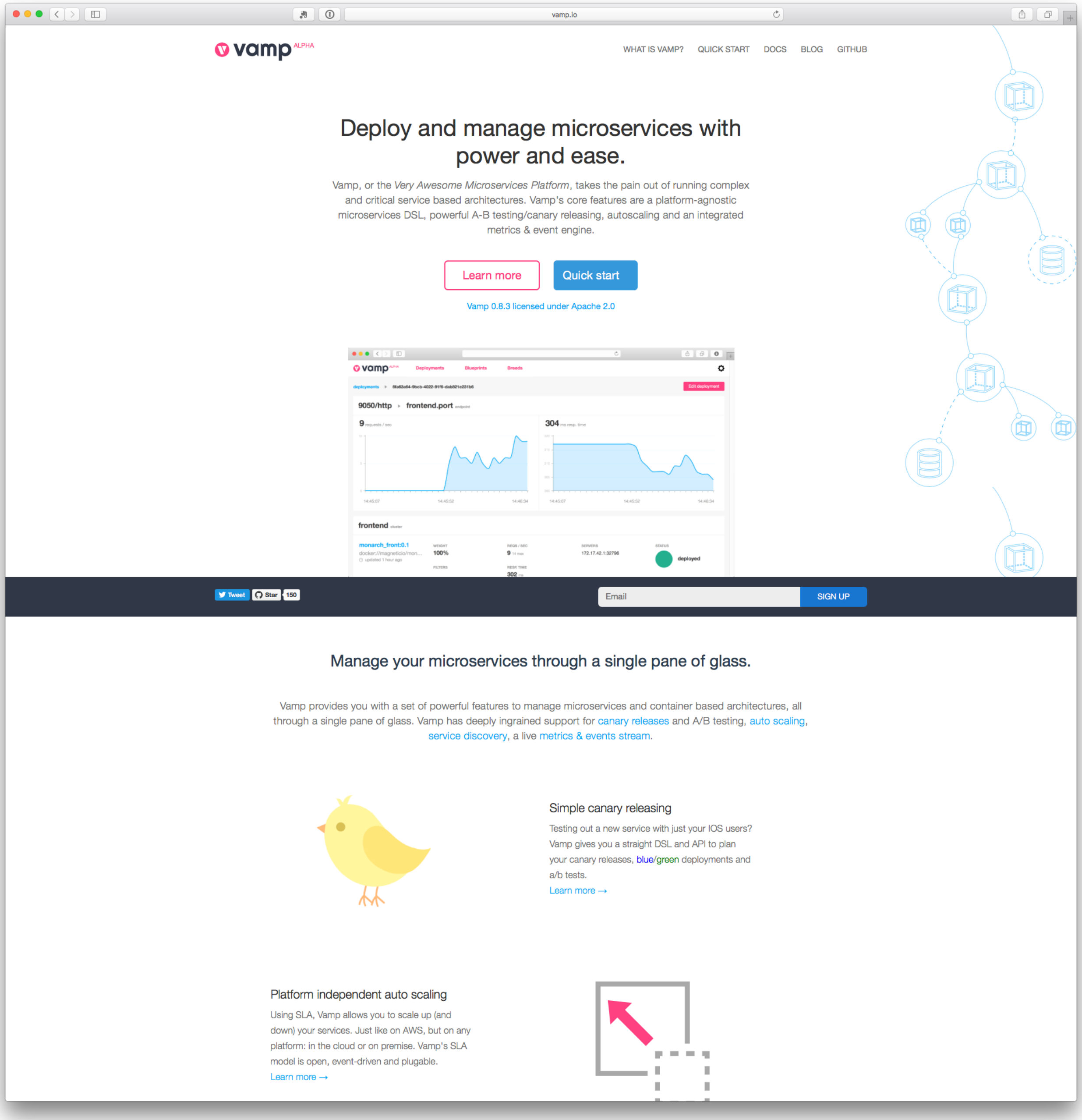
bee-config (Bcfg) 2 is a centralized configuration management server to configure large number of systems, built

FAI

Go to "http://www.devopsbookmarks.com/"

www.devopsbookmarks.com/

Turnkey Platforms



Deploy and manage microservices with power and ease.

Vamp, or the *Very Awesome Microservices Platform*, takes the pain out of running complex and critical service based architectures. Vamp's core features are a platform-agnostic microservices DSL, powerful A-B testing/canary releasing, autoscaling and an integrated metrics & event engine.

Vamp 0.8.3 licensed under Apache 2.0

Manage your microservices through a single pane of glass.

Vamp provides you with a set of powerful features to manage microservices and container based architectures, all through a single pane of glass. Vamp has deeply ingrained support for canary releases and A/B testing, auto scaling, service discovery, a live metrics & events stream.

Simple canary releasing

Testing out a new service with just your IOS users? Vamp gives you a straight DSL and API to plan your canary releases, blue/green deployments and a/b tests.

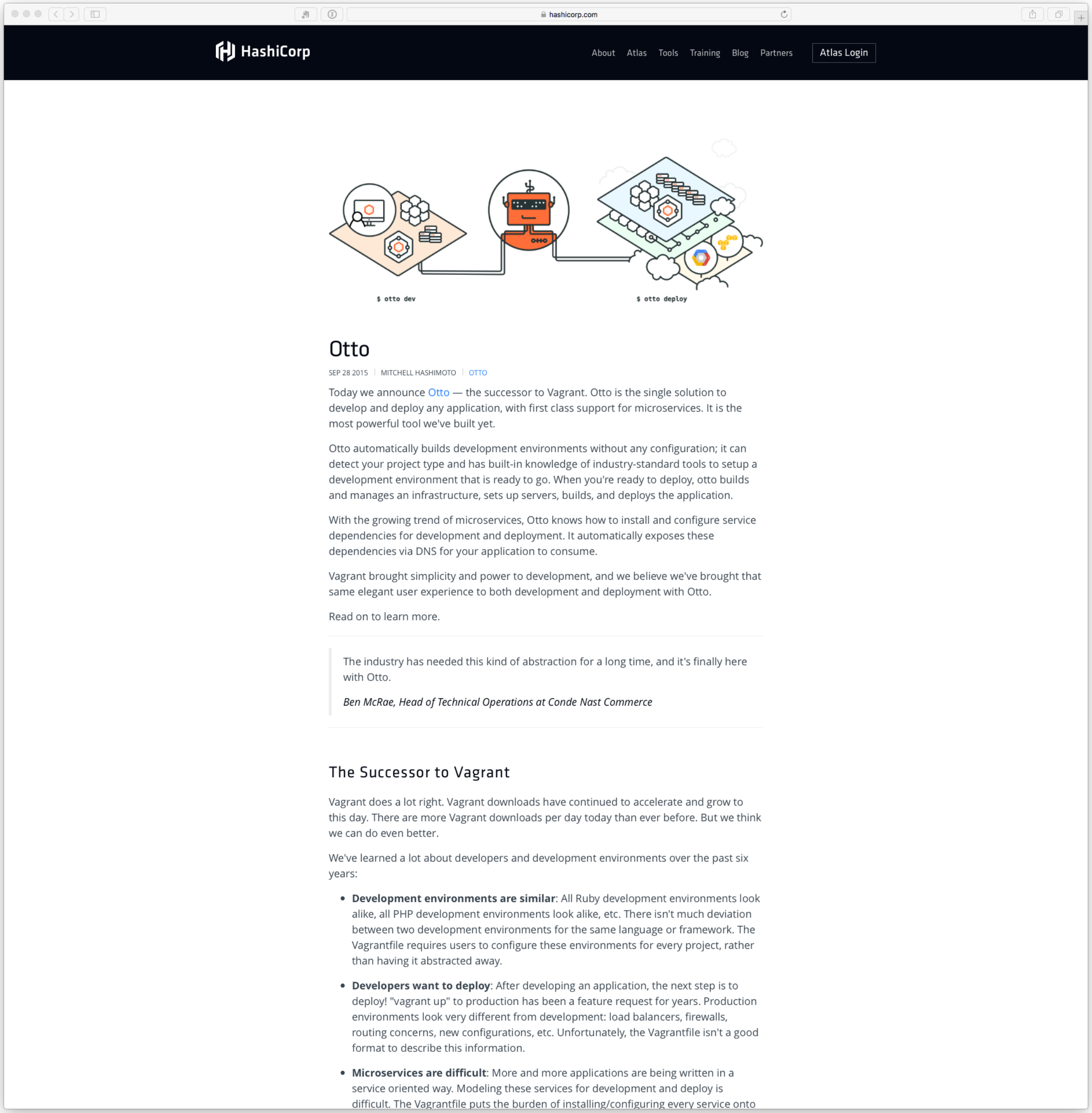
Learn more →

Platform independent auto scaling

Using SLA, Vamp allows you to scale up (and down) your services. Just like on AWS, but on any platform: in the cloud or on premise. Vamp's SLA model is open, event-driven and plugable.

Learn more →

Turnkey Platforms



Otto

SEP 28 2015 | MITCHELL HASHIMOTO | OTTO

Today we announce Otto — the successor to Vagrant. Otto is the single solution to develop and deploy any application, with first class support for microservices. It is the most powerful tool we've built yet.

Otto automatically builds development environments without any configuration; it can detect your project type and has built-in knowledge of industry-standard tools to setup a development environment that is ready to go. When you're ready to deploy, otto builds and manages an infrastructure, sets up servers, builds, and deploys the application.

With the growing trend of microservices, Otto knows how to install and configure service dependencies for development and deployment. It automatically exposes these dependencies via DNS for your application to consume.

Vagrant brought simplicity and power to development, and we believe we've brought that same elegant user experience to both development and deployment with Otto.

Read on to learn more.

> The industry has needed this kind of abstraction for a long time, and it's finally here with Otto.
>
> *Ben McRae, Head of Technical Operations at Conde Nast Commerce*

The Successor to Vagrant

Vagrant does a lot right. Vagrant downloads have continued to accelerate and grow to this day. There are more Vagrant downloads per day today than ever before. But we think we can do even better.

We've learned a lot about developers and development environments over the past six years:

- **Development environments are similar**: All Ruby development environments look alike, all PHP development environments look alike, etc. There isn't much deviation between two development environments for the same language or framework. The Vagrantfile requires users to configure these environments for every project, rather than having it abstracted away.
- **Developers want to deploy**: After developing an application, the next step is to deploy! "vagrant up" to production has been a feature request for years. Production environments look very different from development: load balancers, firewalls, routing concerns, new configurations, etc. Unfortunately, the Vagrantfile isn't a good format to describe this information.
- **Microservices are difficult**: More and more applications are being written in a service oriented way. Modeling these services for development and deploy is difficult. The Vagrantfile puts the burden of installing/configuring every service onto

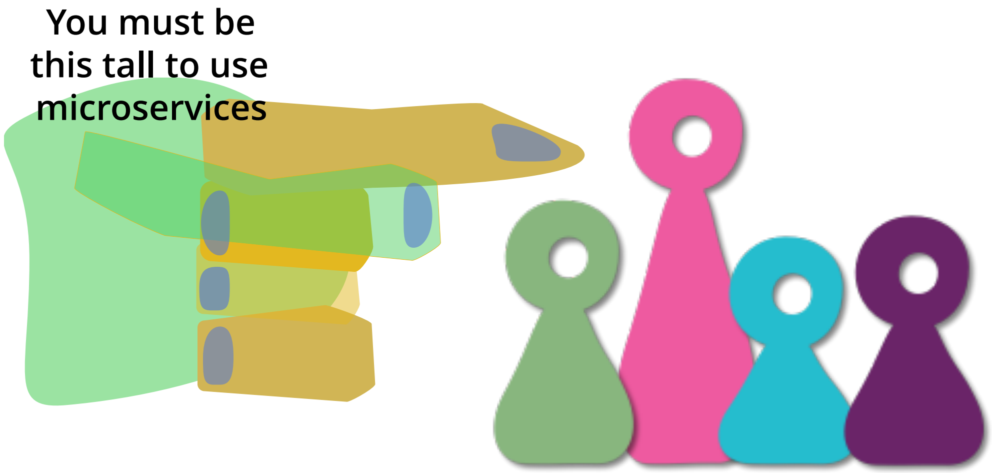
https://www.hashicorp.com/blog/otto.html

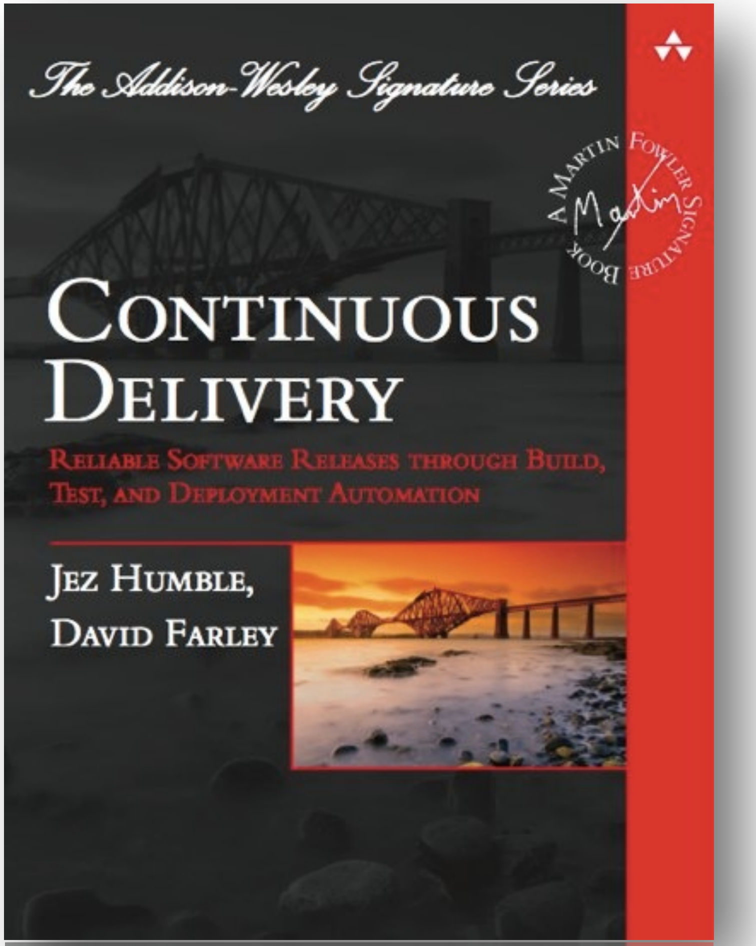
AGENDA

what problem

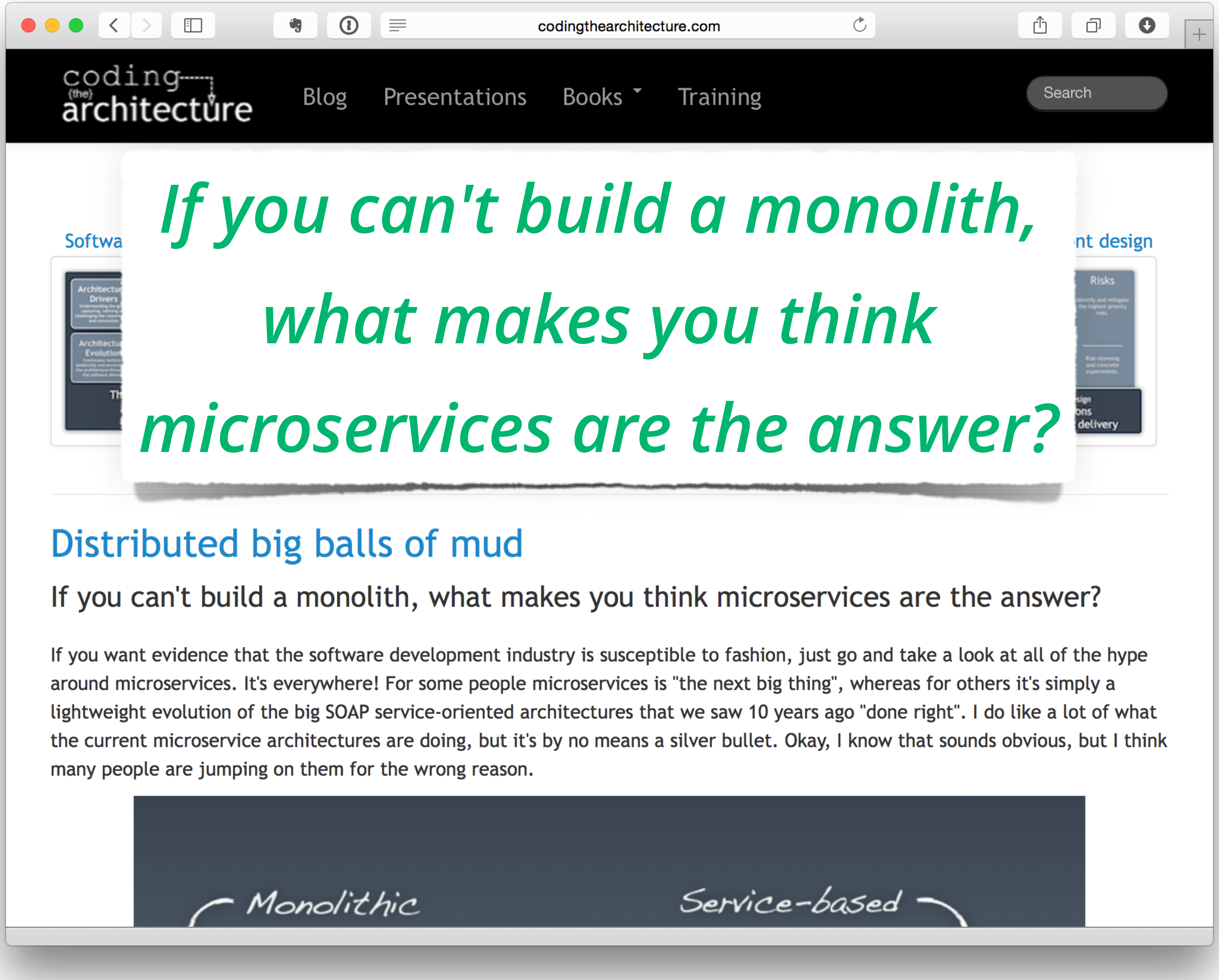
characteristics

engineering




The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble, David Farley

(Micro)service architectures provide unique benefits at the cost of increased (essential) complexity.



If you can't build a monolith, what makes you think microservices are the answer?

Distributed big balls of mud

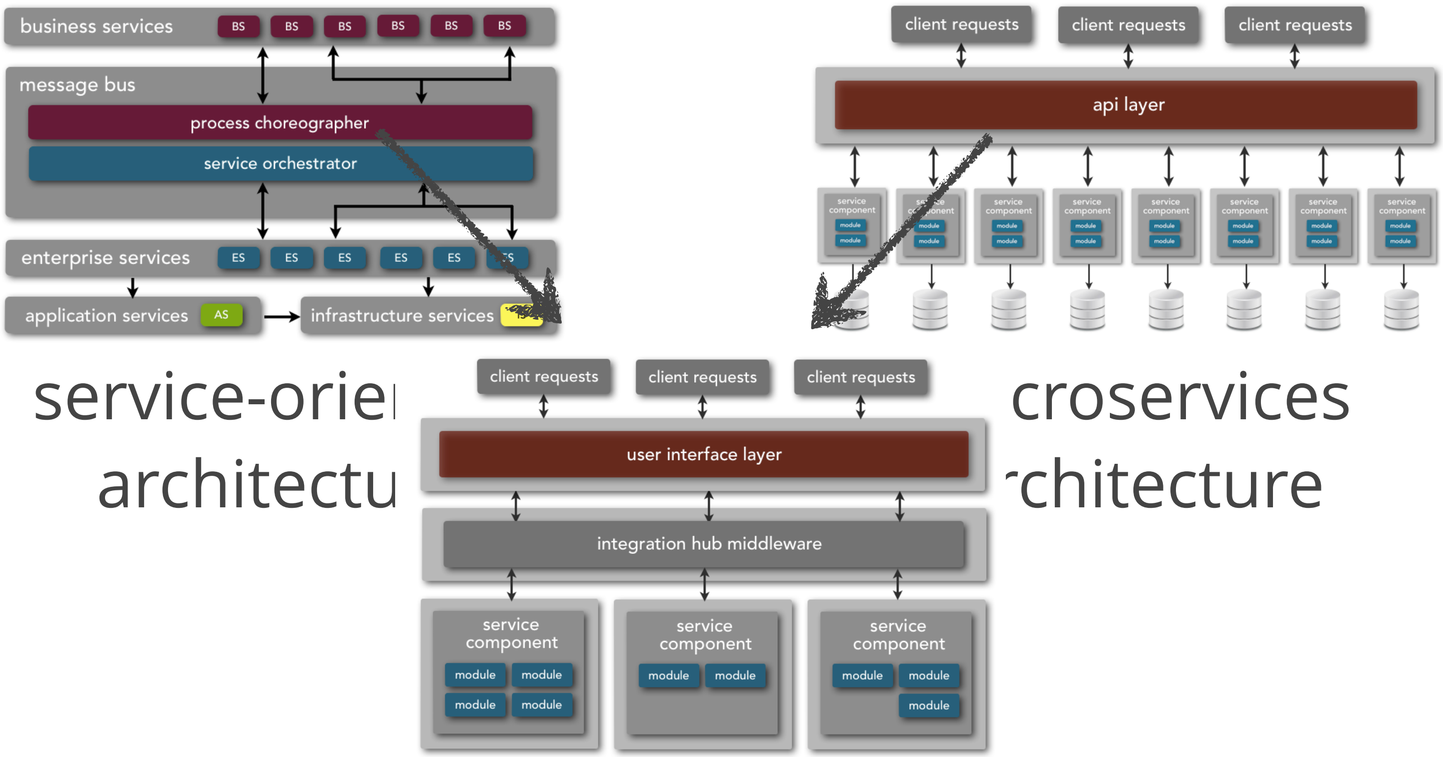
If you can't build a monolith, what makes you think microservices are the answer?

If you want evidence that the software development industry is susceptible to fashion, just go and take a look at all of the hype around microservices. It's everywhere! For some people microservices is "the next big thing", whereas for others it's simply a lightweight evolution of the big SOAP service-oriented architectures that we saw 10 years ago "done right". I do like a lot of what the current microservice architectures are doing, but it's by no means a silver bullet. Okay, I know that sounds obvious, but I think many people are jumping on them for the wrong reason.

www.codingthearchitecture.com/2014/07/06/distributed_big_balls_of_mud.html

Service-*based* Architecture

is there a middle ground?

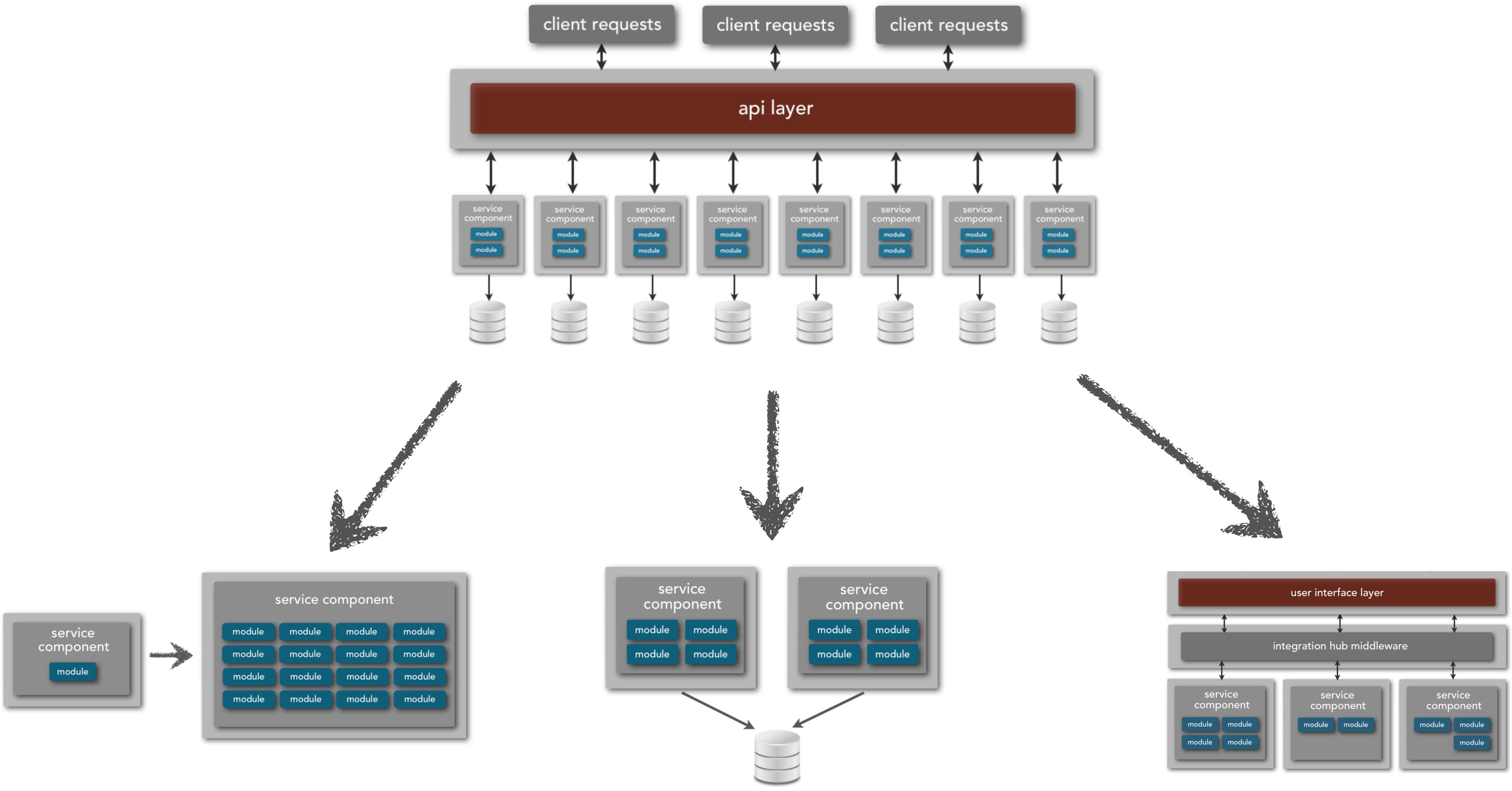


service-orie
architectu

croservices
rchitecture

service-based architecture

Service-*based* Architecture

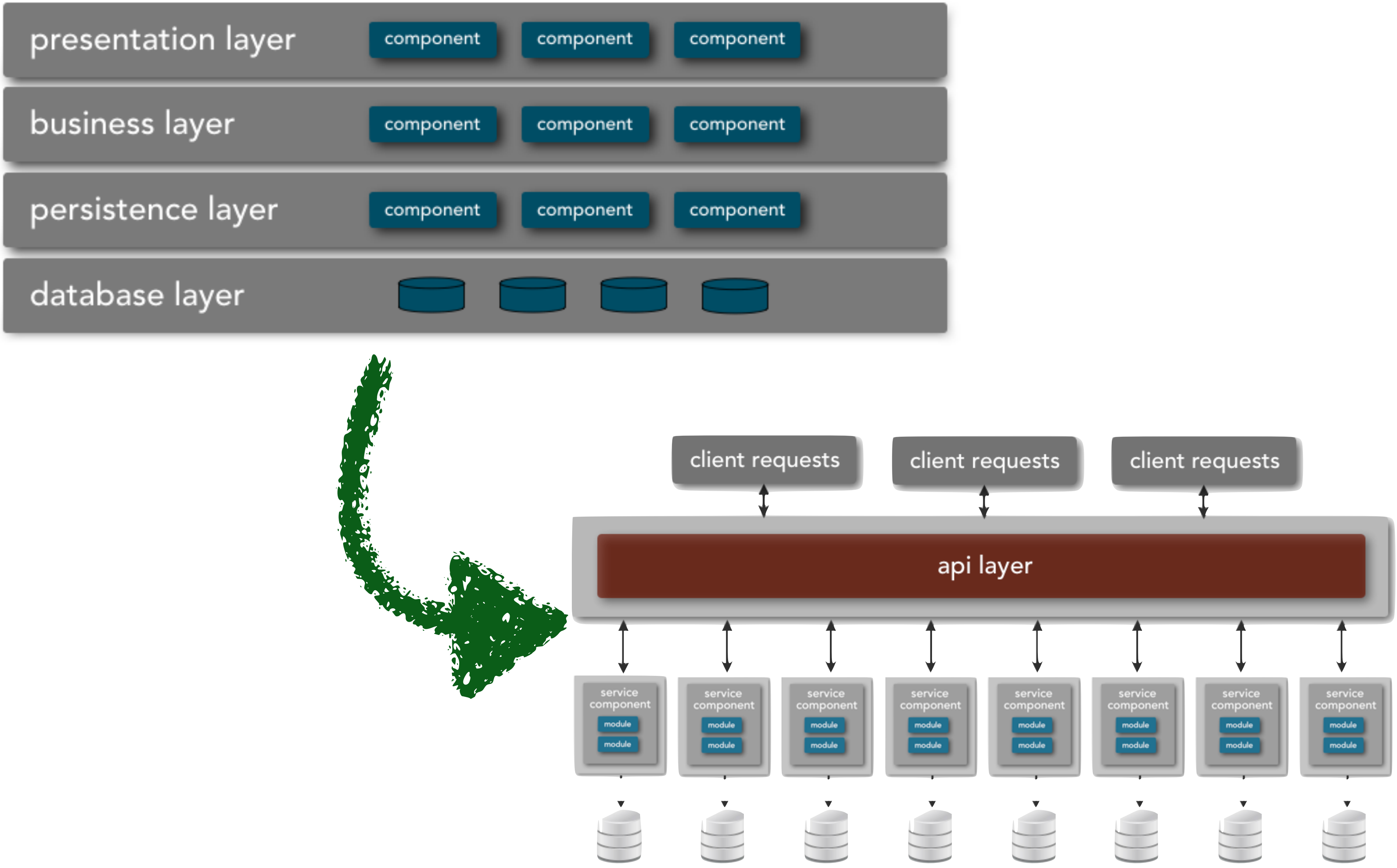


service granularity

database scope

integration hub

Migration



presentation layer component component component

business layer component component component

persistence layer component component component

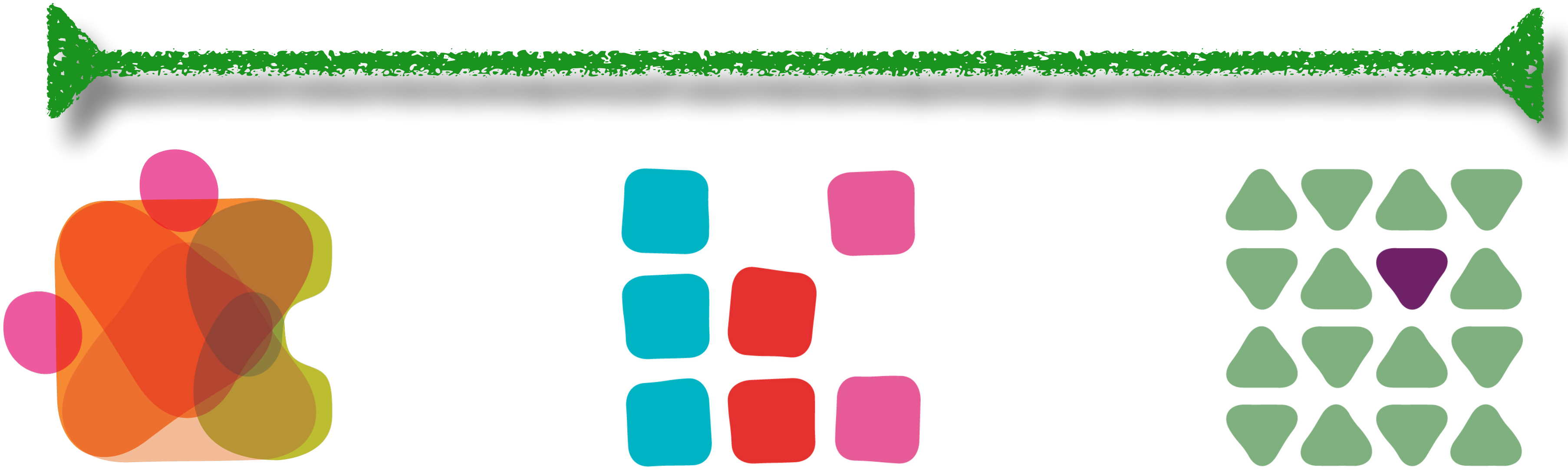
database layer

client requests client requests client requests

api layer

service component module module (×8)

Partition Along Natural Boundaries



Build a small number of larger services first.

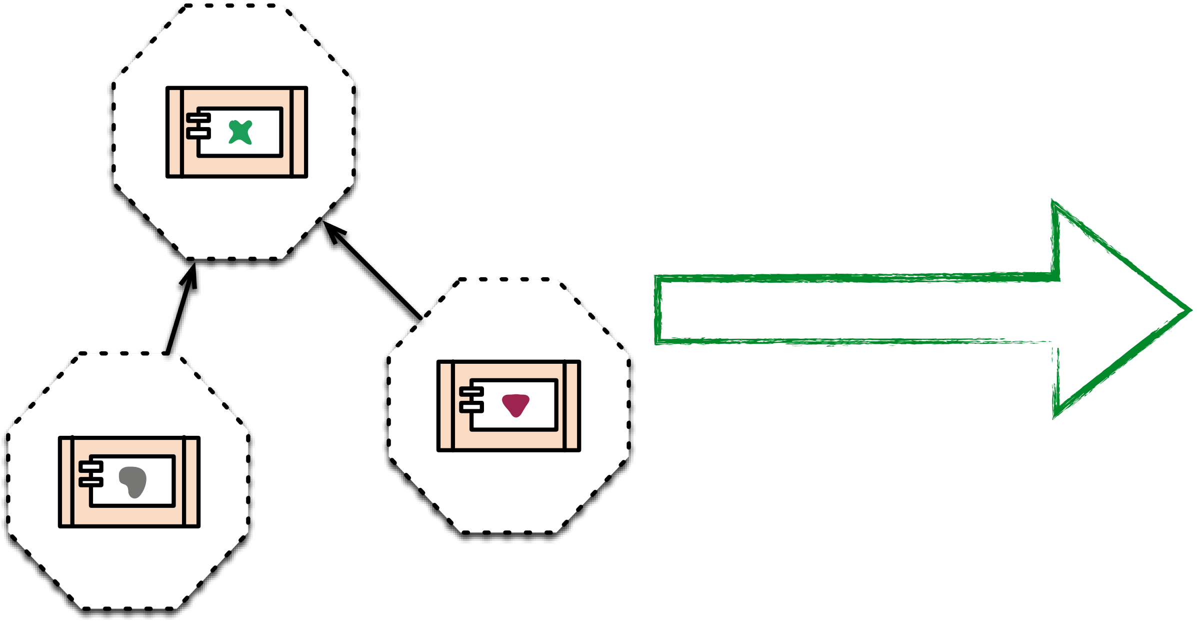
domain

structural

organizational

transactional

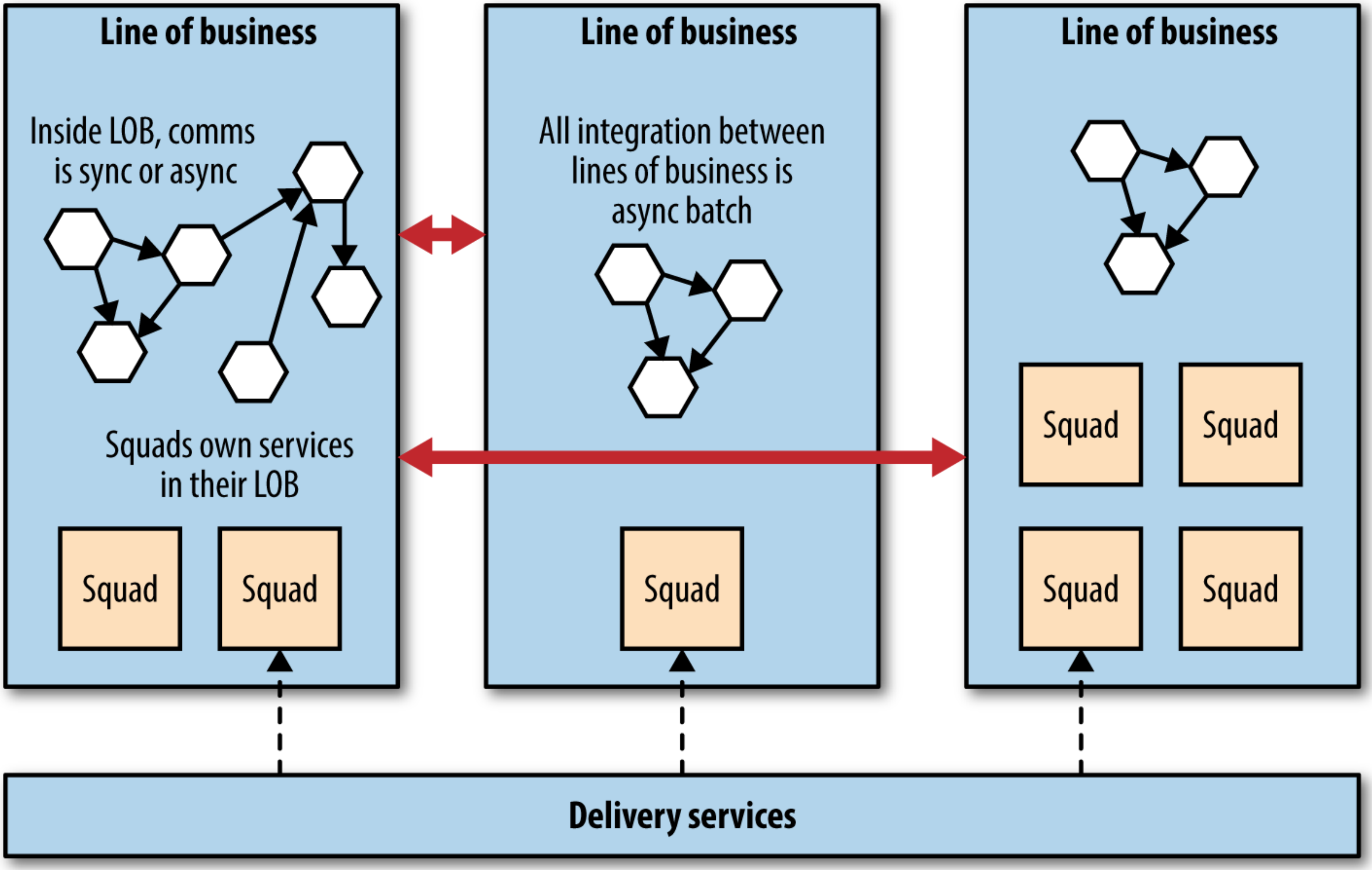
Inverse Conway Maneuver



domain

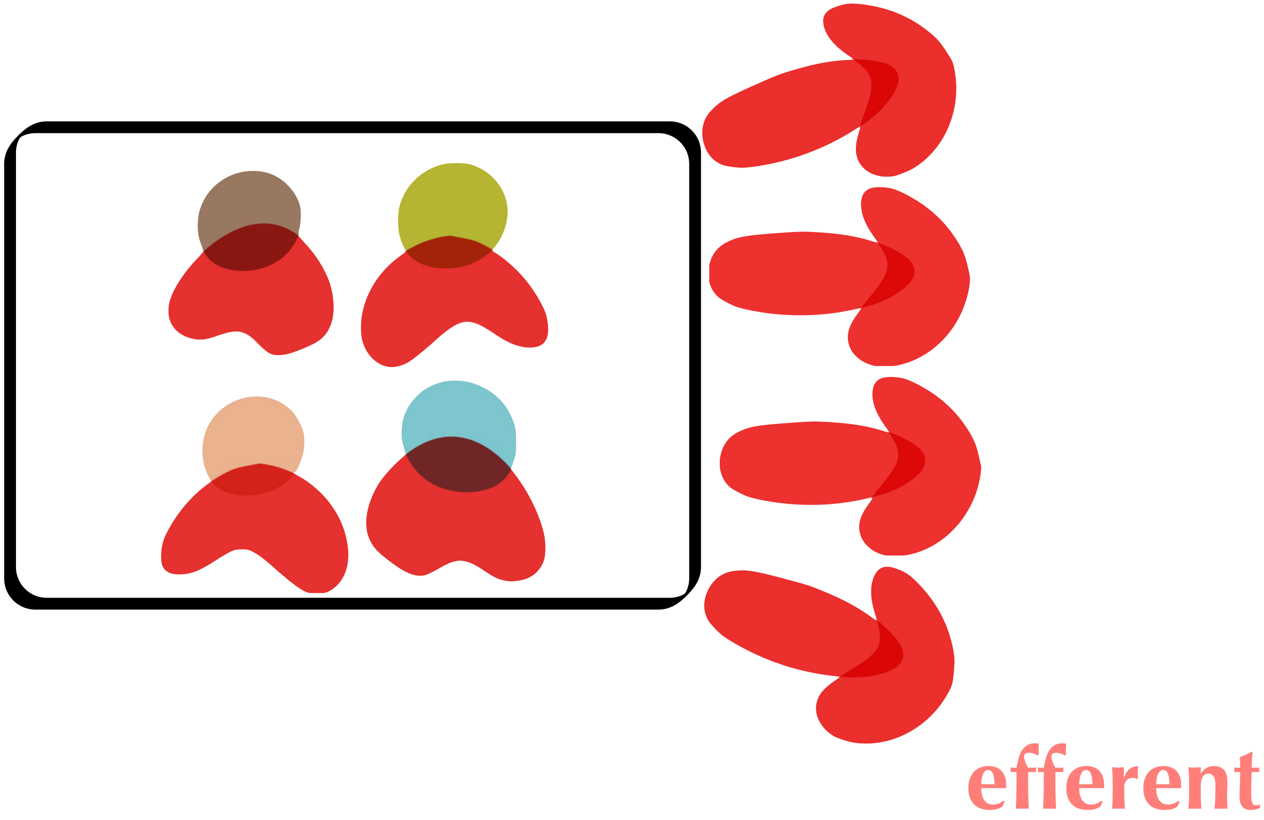
Build teams that look like
the architecture you want
(and it will follow).

case study: RealEstate.com.au


Line of business
Inside LOB, comms is sync or async
Squads own services in their LOB
Squad
Squad
Line of business
All integration between lines of business is async batch
Squad
Line of business
Squad
Squad
Squad
Squad
Delivery services

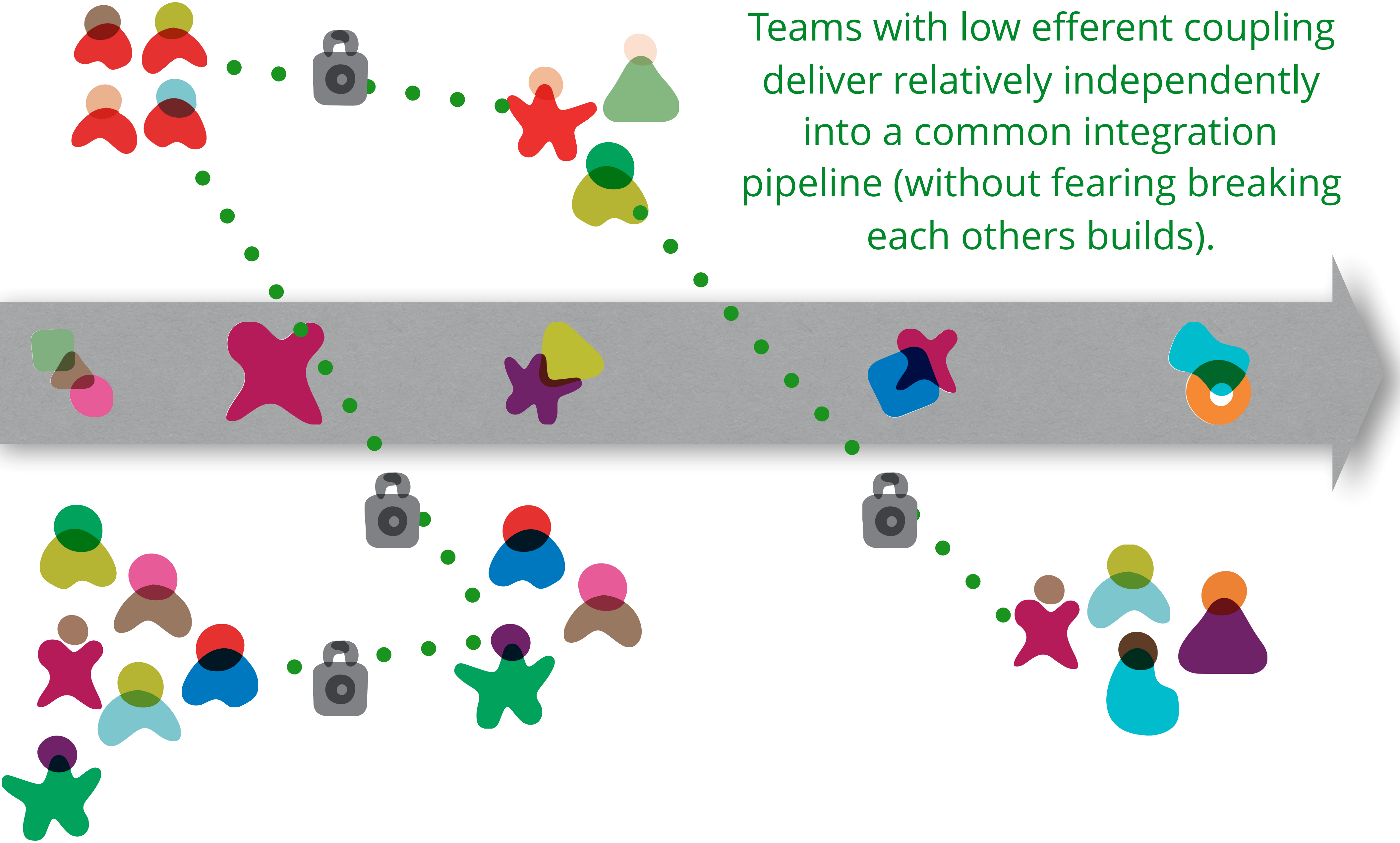
Provides tooling and consulting to squads

Efferent Coupling

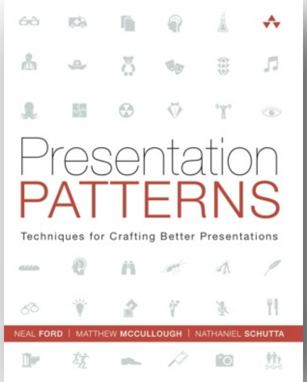


Strive for low efferent coupling for your team.

Continuous Delivery

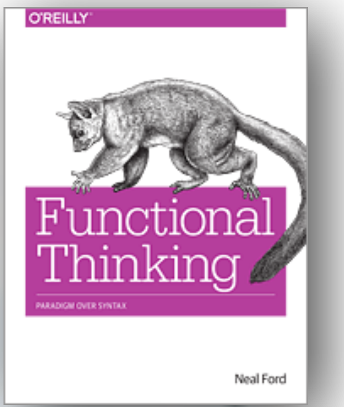


Teams with low efferent coupling deliver relatively independently into a common integration pipeline (without fearing breaking each others builds).


Presentation PATTERNS

Techniques for Crafting Better Presentations

NEAL FORD | MATTHEW MCCULLOUGH | NATHANIEL SCHUTTA


O'REILLY

Functional Thinking

PARADIGM OVER SYNTAX

Neal Ford

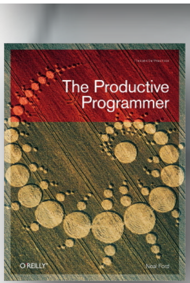
nealford.com

@neal4d

ThoughtWorks®

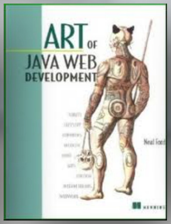
NEAL FORD

Director / Software Architect / Meme Wrangler


The Productive Programmer

O'REILLY

Neal Ford


ART of JAVA WEB DEVELOPMENT

Neal Ford

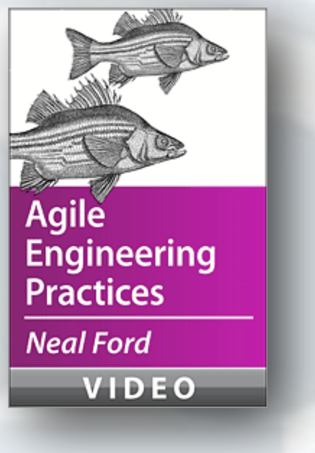
Borland JBuilder 3 Unleashed

SAMS

Developing with Delphi

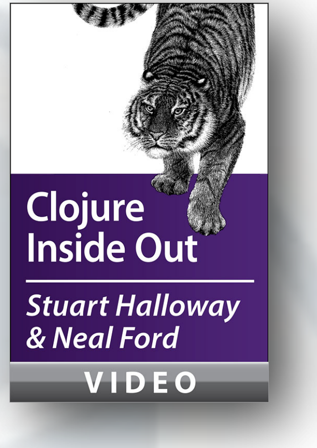
nealford.com/books

nealford.com/videos


Agile Engineering Practices

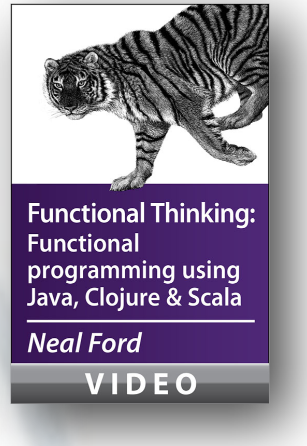
Neal Ford

VIDEO


Clojure Inside Out

Stuart Halloway & Neal Ford

VIDEO


Functional Thinking: Functional programming using Java, Clojure & Scala

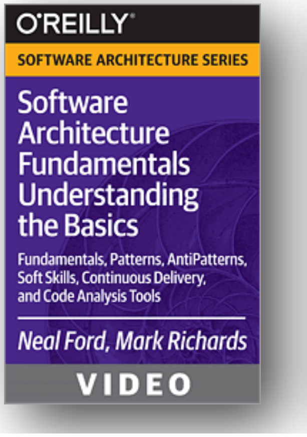
Neal Ford

VIDEO

O'REILLY®

SOFTWARE ARCHITECTURE SERIES

www.oreilly.com/software-architecture-video-training-series.html


O'REILLY

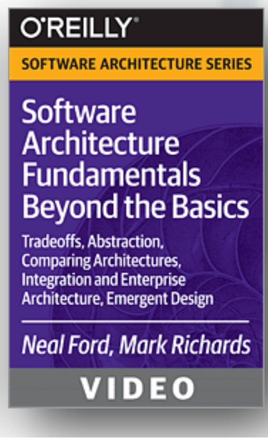
SOFTWARE ARCHITECTURE SERIES

Software Architecture Fundamentals Understanding the Basics

Fundamentals, Patterns, AntiPatterns, Soft Skills, Continuous Delivery, and Code Analysis Tools

Neal Ford, Mark Richards

VIDEO


O'REILLY

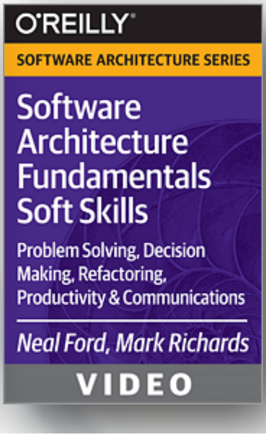
SOFTWARE ARCHITECTURE SERIES

Software Architecture Fundamentals Beyond the Basics

Tradeoffs, Abstraction, Comparing Architectures, Integration and Enterprise Architecture, Emergent Design

Neal Ford, Mark Richards

VIDEO


O'REILLY

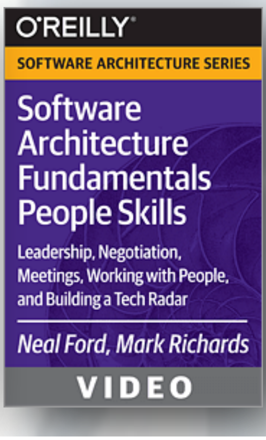
SOFTWARE ARCHITECTURE SERIES

Software Architecture Fundamentals Soft Skills

Problem Solving, Decision Making, Refactoring, Productivity & Communications

Neal Ford, Mark Richards

VIDEO


O'REILLY

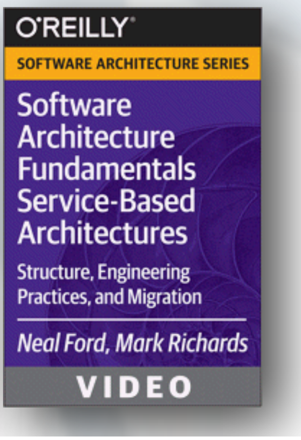
SOFTWARE ARCHITECTURE SERIES

Software Architecture Fundamentals People Skills

Leadership, Negotiation, Meetings, Working with People, and Building a Tech Radar

Neal Ford, Mark Richards

VIDEO


O'REILLY

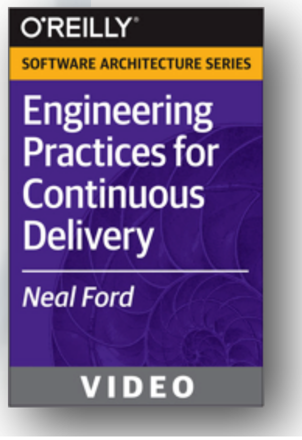
SOFTWARE ARCHITECTURE SERIES

Software Architecture Fundamentals Service-Based Architectures

Structure, Engineering Practices, and Migration

Neal Ford, Mark Richards

VIDEO


O'REILLY

SOFTWARE ARCHITECTURE SERIES

Engineering Practices for Continuous Delivery

Neal Ford

VIDEO