A pattern language for microservices

Chris Richardson

Founder of Eventuate.io

Founder of the original CloudFoundry.com

Author of POJOs in Action

@crichardson

chris@chrisrichardson.net

http://microservices.io

http://eventuate.io

http://plainoldobjects.com

Presentation goal

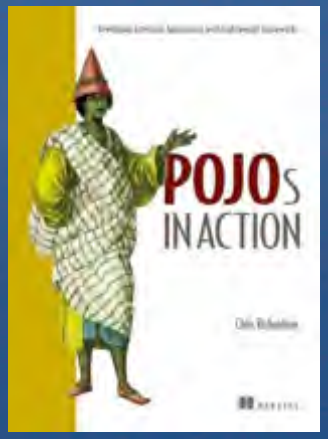
Overview of the microservice architecture pattern language

and

how to use it to architect an application

About Chris


COMMUNICATIONS
LISP
OF THE ACM


POJOs
IN ACTION


CLOUDFOUNDRY
The Enterprise Java Cloud
SIGN UP
LEARN MORE
CLOUDFOUNDRY
APPLICATION DEMO


vmware
spring

About Chris

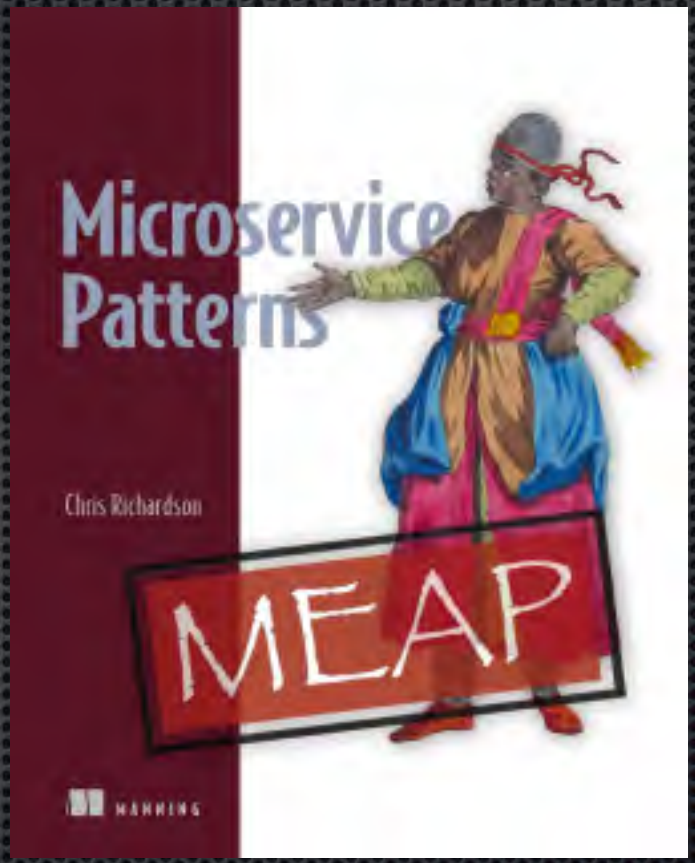
Consultant and trainer focusing on modern application architectures including microservices (http://www.chrisrichardson.net/)

About Chris

Founder of a startup that is creating an open-source/SaaS platform that simplifies the development of transactional microservices

(http://eventuate.io)

About Chris

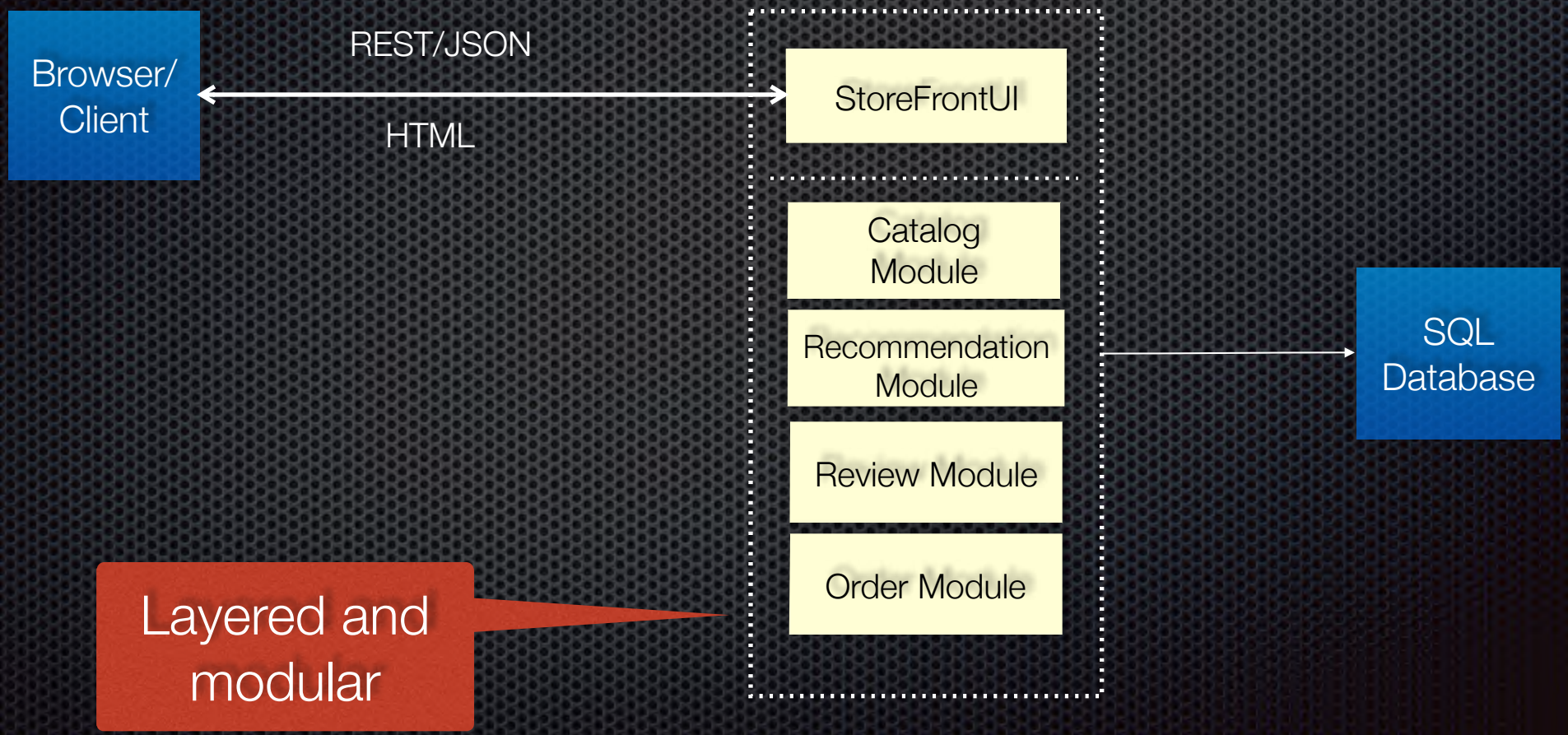


https://www.manning.com/books/microservice-patterns

Agenda

- Why microservices?
- Benefits of the microservice architecture
- Microservices != silver bullet
- The microservice pattern language
- Applying the microservice pattern language

Let's imagine you are building an online store



Problem: what's the deployment architecture?

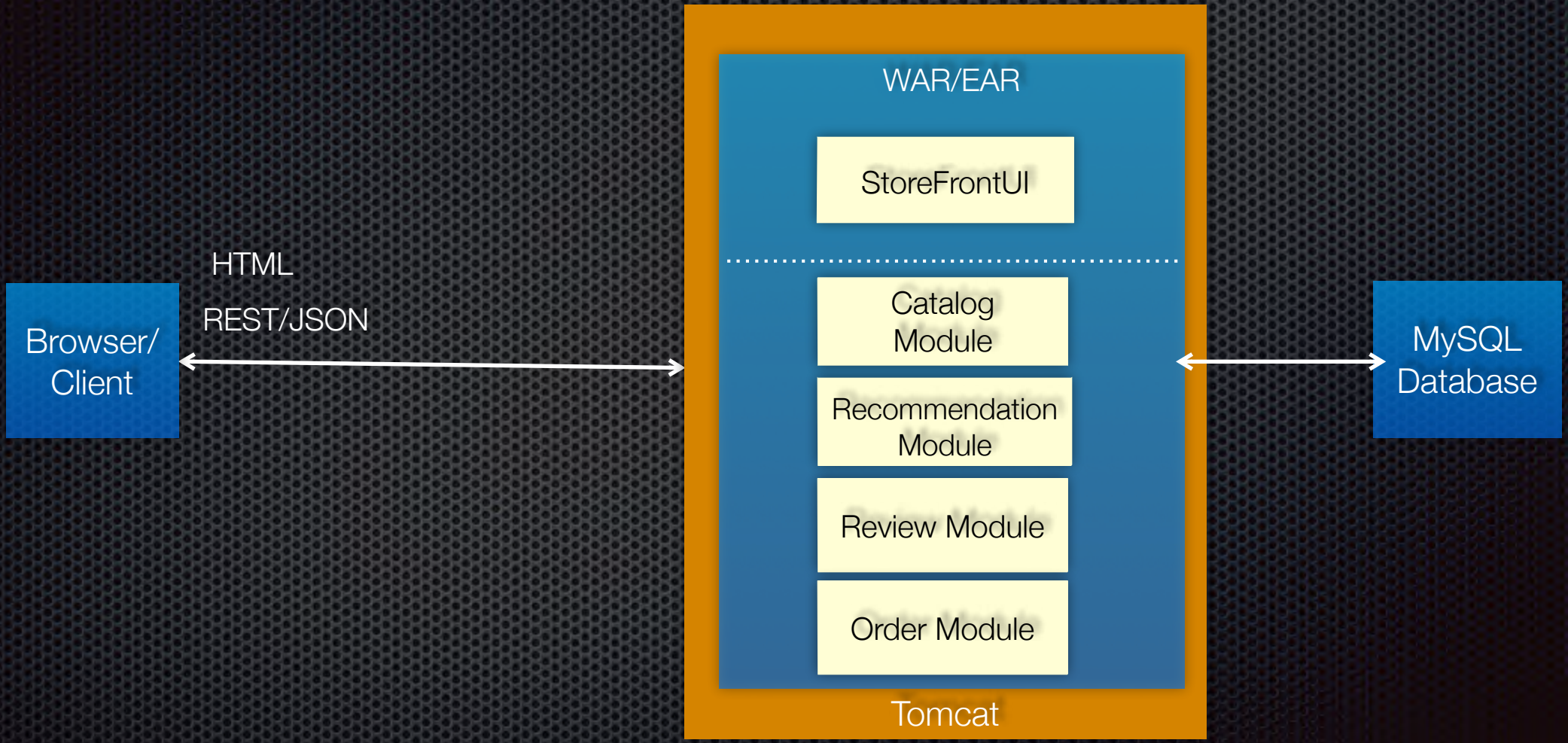
Forces

Businesses must innovate faster

⇒

Develop more complex, higher-quality software faster

Traditional Monolithic architecture



Simple to ....

Develop
Test
Deploy
Scale

Successful applications have a habit of growing

⇒

Big, complex, monolithic applications

Eventually, agile development and deployment becomes impossible

Requires long-term commitment to a technology stack

Monolithic hell

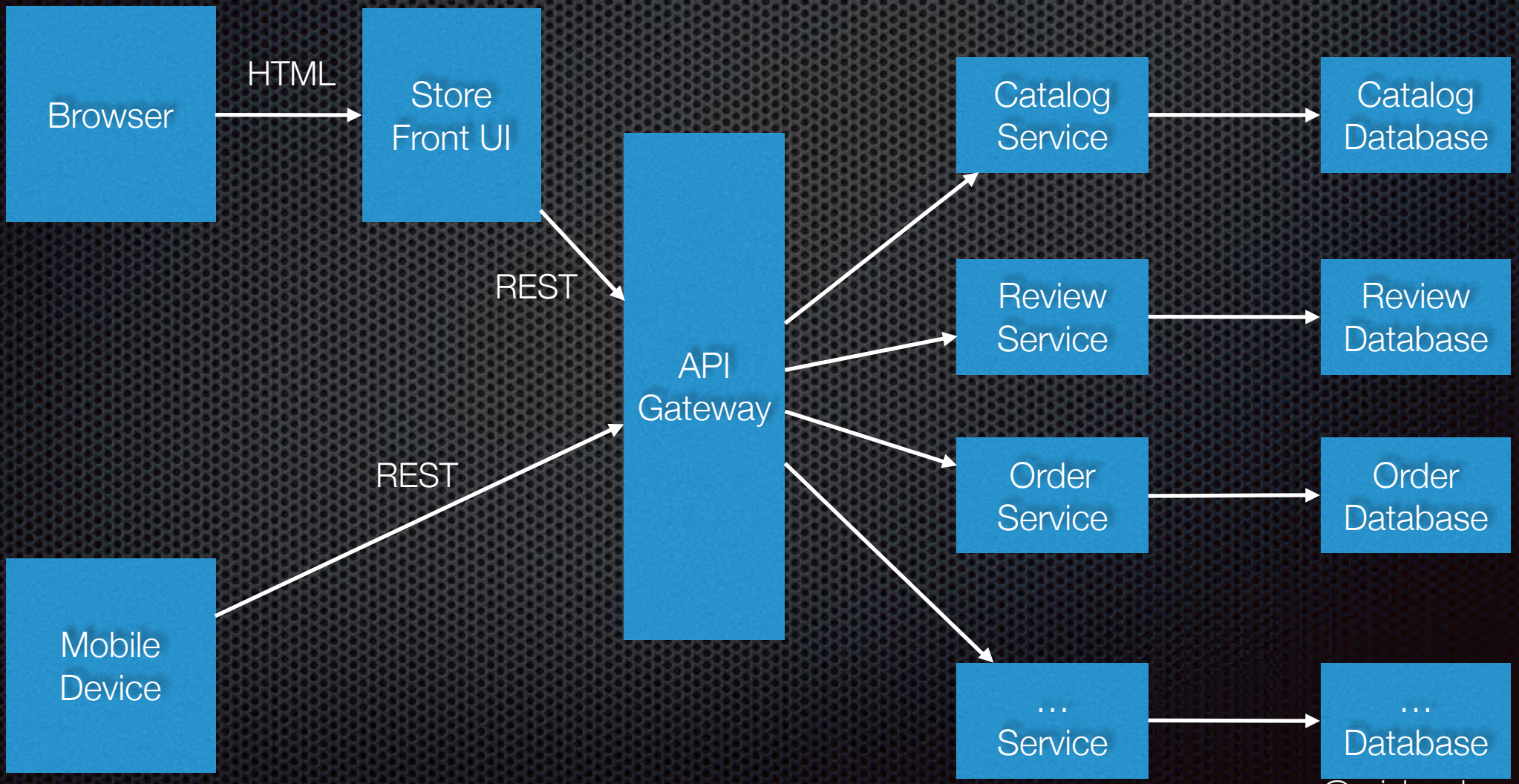
- Development is slow
- Application is becoming a big ball of mud
- No one fully understands the application
- It's written in an obsolete technology stack

The microservice architecture

Loosely coupled **services**
organized around
business capabilities

The Microservice architecture tackles complexity through modularization

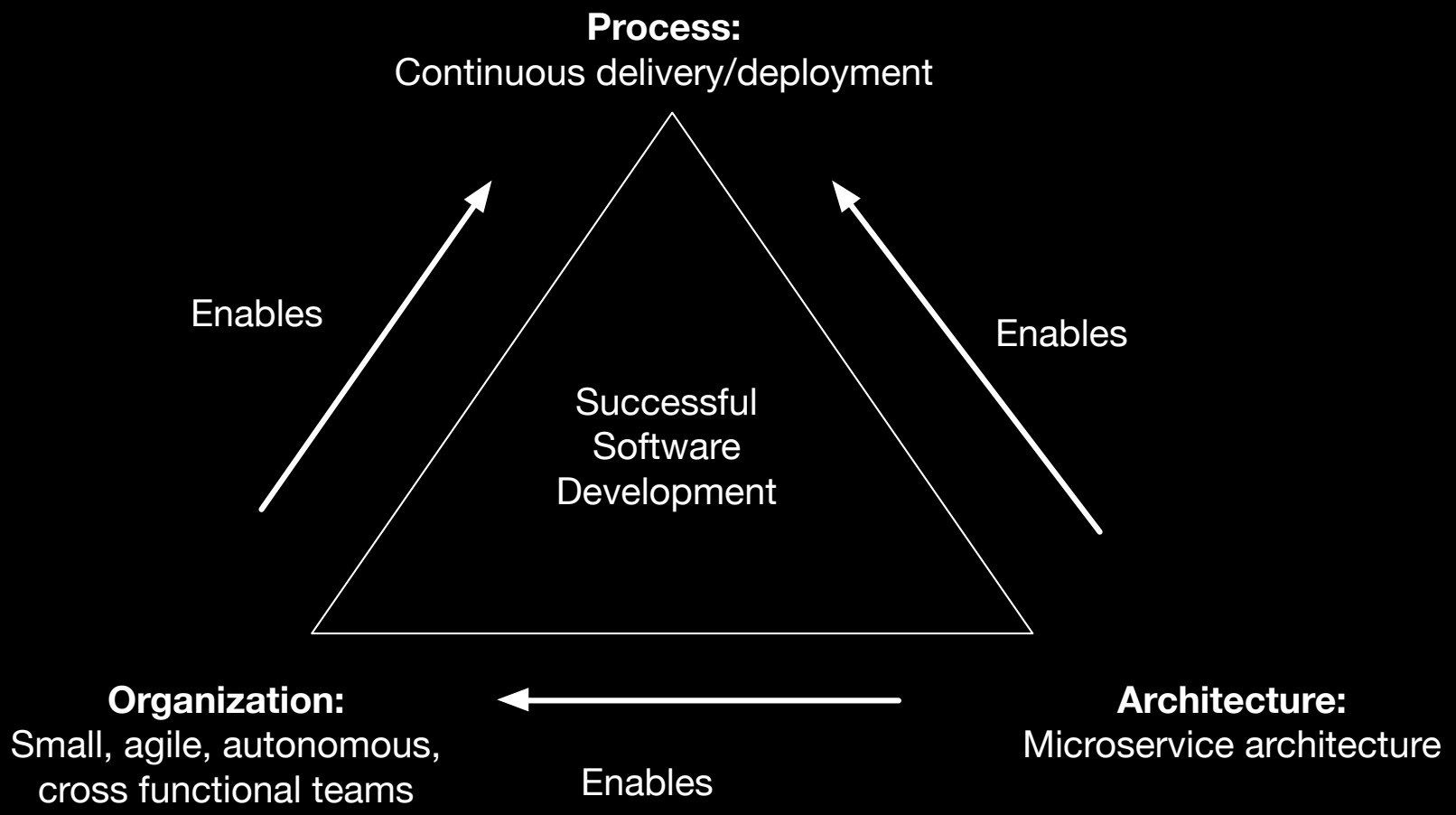
Microservice architecture



Agenda

- Why microservices?
- Benefits of the microservice architecture
- Microservices != silver bullet
- The microservice pattern language
- Applying the microservice pattern language

Microservices enable continuous delivery/deployment

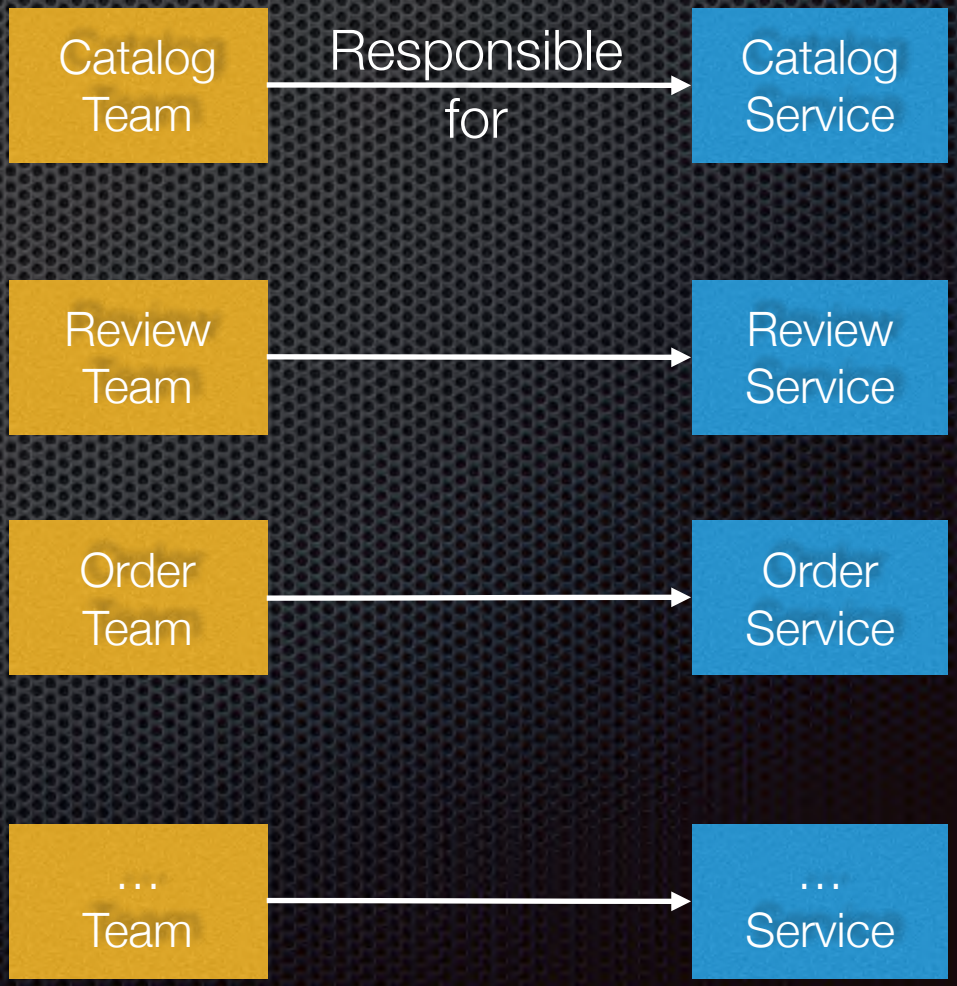


Smaller, simpler applications

- Easier to understand and develop
- Less jar/classpath hell - who needs OSGI?
- Faster to build and deploy
- Reduced startup time

Scales development: develop, deploy and scale each service independently

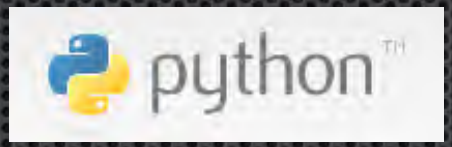
- Small, autonomous teams
- Clearly defined responsibilities

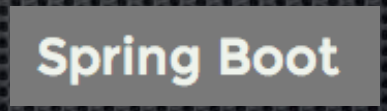


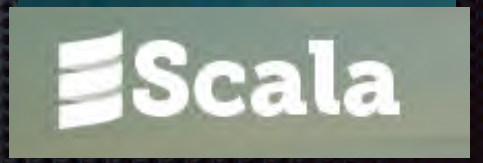
Easier to scale

Improves fault isolation

Easily **evolve** your technology stack







... and fail safely

Agenda

- Why microservices?
- Benefits of the microservice architecture
- Microservices != silver bullet
- The microservice pattern language
- Applying the microservice pattern language

No silver bullets

No Silver Bullet
—Essence and Accident in Software Engineering

Frederick P. Brooks, Jr.
University of North Carolina at Chapel Hill

There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity.

http://en.wikipedia.org/wiki/Fred_Brooks

Complexity

Microservices - Not A Free Lunch!

http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html

Complexity of developing a distributed system

Multiple databases
&
Transaction management

Must use event-driven
eventual consistency

Complexity of testing a distributed system

Complexity of deploying and operating a distributed system

You need a lot of automation: VM/container orchestration or PaaS

Developing and deploying features that span multiple services requires careful coordination

Agenda

- Why microservices?
- Benefits of the microservice architecture
- Microservices != silver bullet
- The microservice pattern language
- Applying the microservice pattern language

Are microservices a good fit for my application?

When using microservices:

How to decompose an application into services?

How to deploy an application's services?

How to handle cross cutting concerns?

Which communication mechanisms to use?

How do external clients communicate with the services?

How does a client discover the network location of a service instance?

How to prevent a network or service failure from cascading to other services?

How to maintain data consistency and implement queries?

How to understand the behavior of an application and troubleshoot problems?

How to make testing easier?

How to implement a UI screen or page that displays data from multiple services?

Microservice pattern language

=

collection of patterns
that solve these
architecture, design, development and
operational problems

What's a pattern?

Reusable **solution**
to a **problem**
occurring
in a particular **context**

The structure of a pattern

Name

Context

aka the situation

Problem

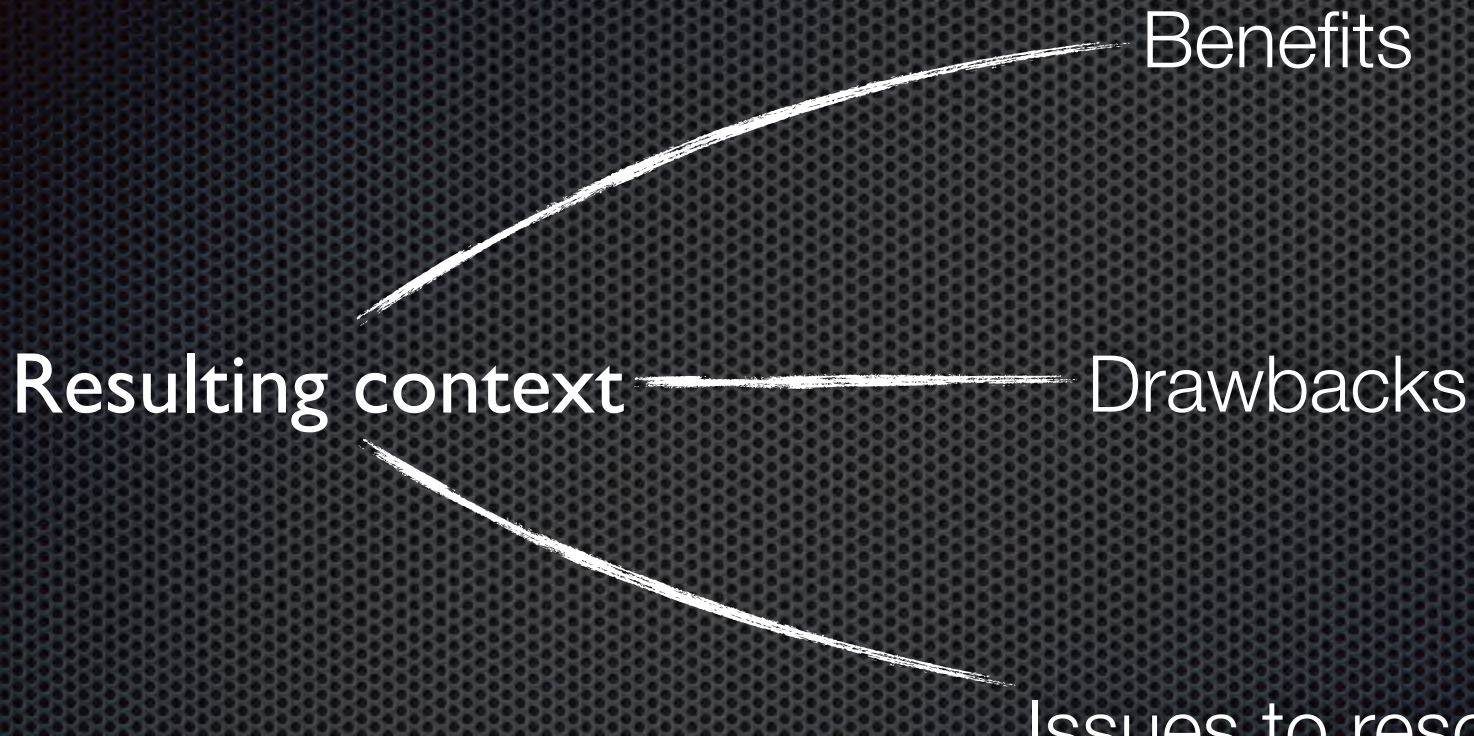
(conflicting) issues etc to address

Forces

Solution

Resulting context

Related patterns

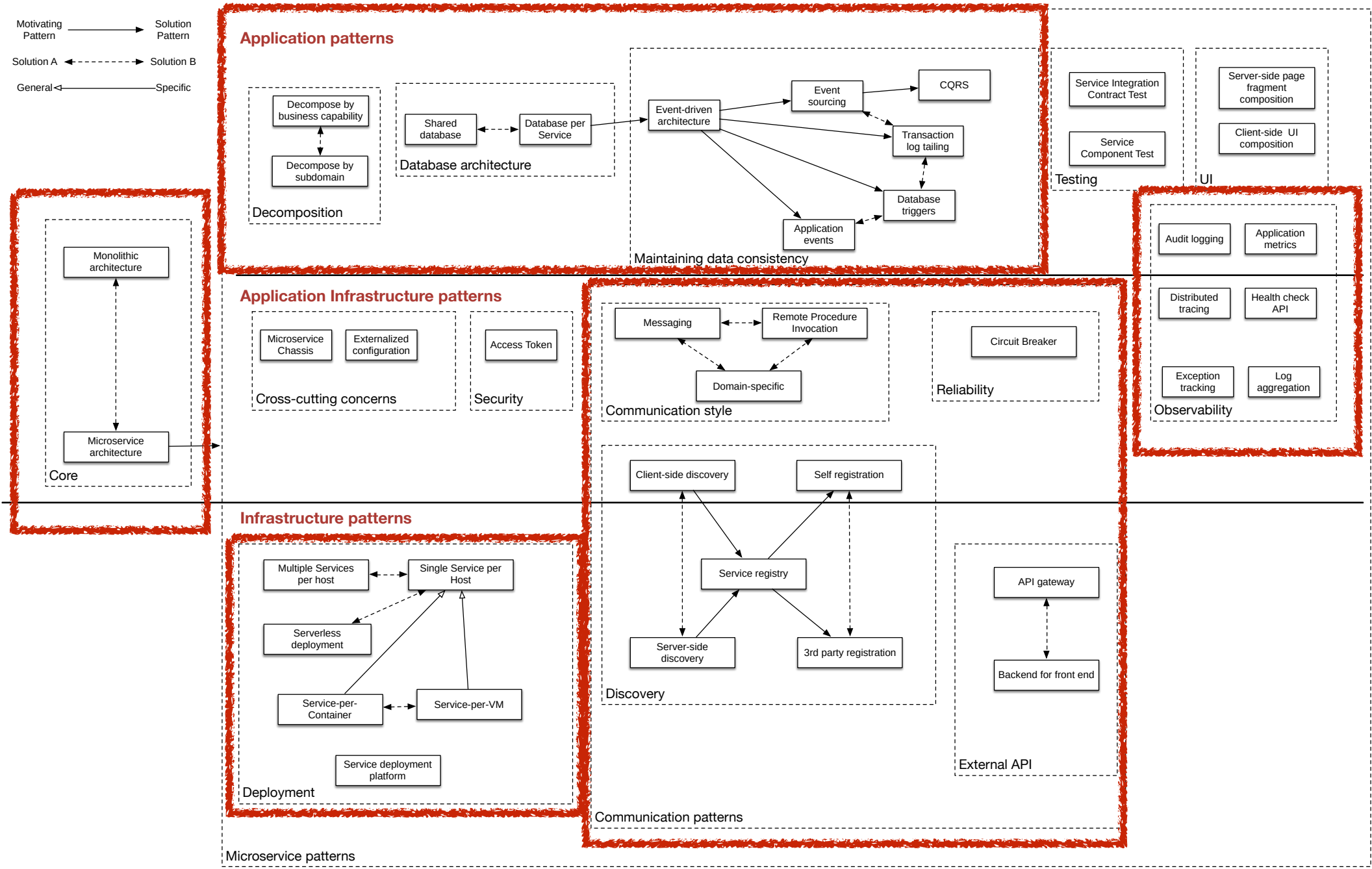

Benefits
Resulting context
Drawbacks
Issues to resolve

Related patterns

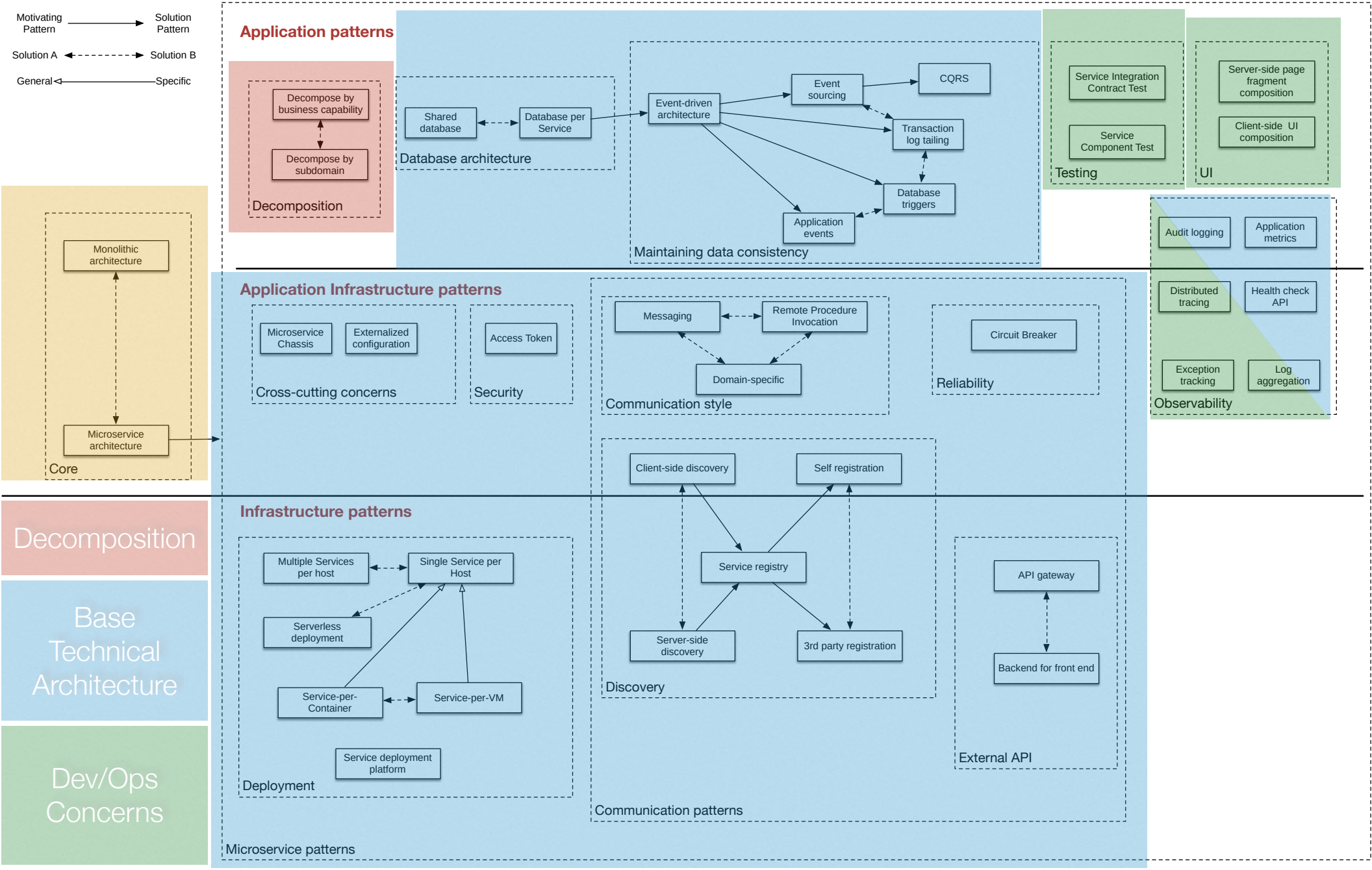
Alternative pattern:
Different solution to same problem

Successor pattern:
Solves problem introduced by this pattern

Microservices pattern language: http://microservices.io



Microservices pattern language: http://microservices.io



Agenda

- Why microservices?
- Benefits of the microservice architecture
- Microservices != silver bullet
- The microservice pattern language
- Applying the microservice pattern language

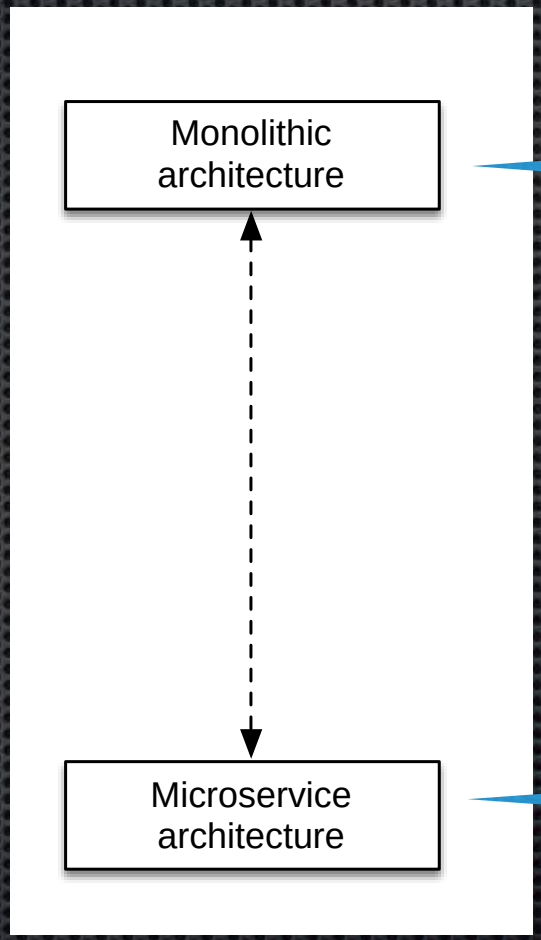
The pattern language guides you when developing an architecture

- What architectural decisions you must make
- For each decision:
  - Available options
  - Trade-offs of each option

Issue: What's the deployment architecture?

Forces

- Maintainability
- Deployability
- Testability
- Extensibility
- …



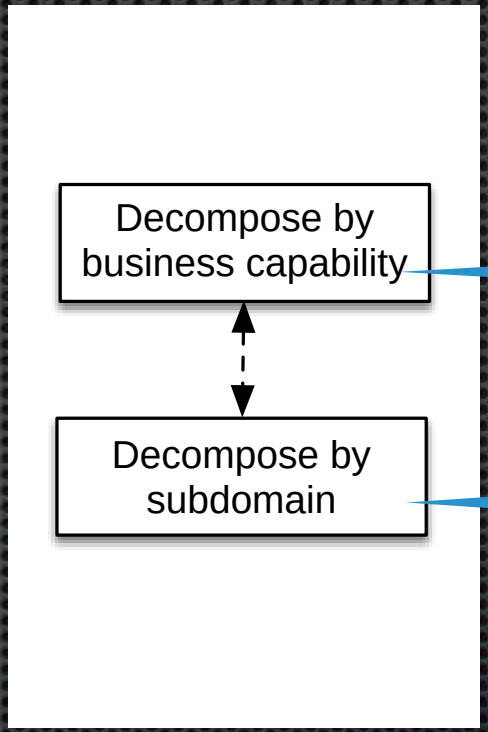
Single deployable/ executable **OR** Tightly coupled services

Multiple loosely coupled services

Issue: How to decompose an application into services?

Forces

- Stability
- Cohesive
- Loosely coupled
- Not too large



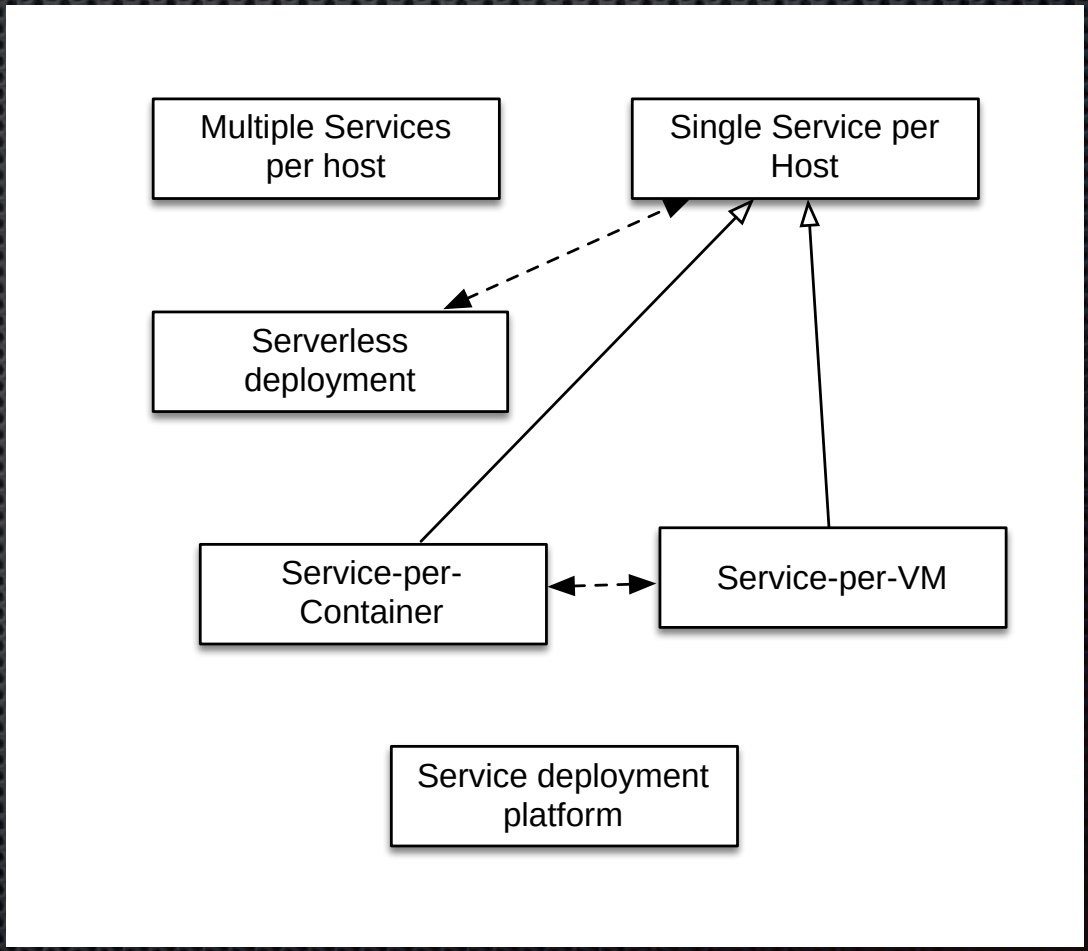
Organize around business capabilities

Organize around DDD subdomains

Issue: How to deploy an application's services?

Forces

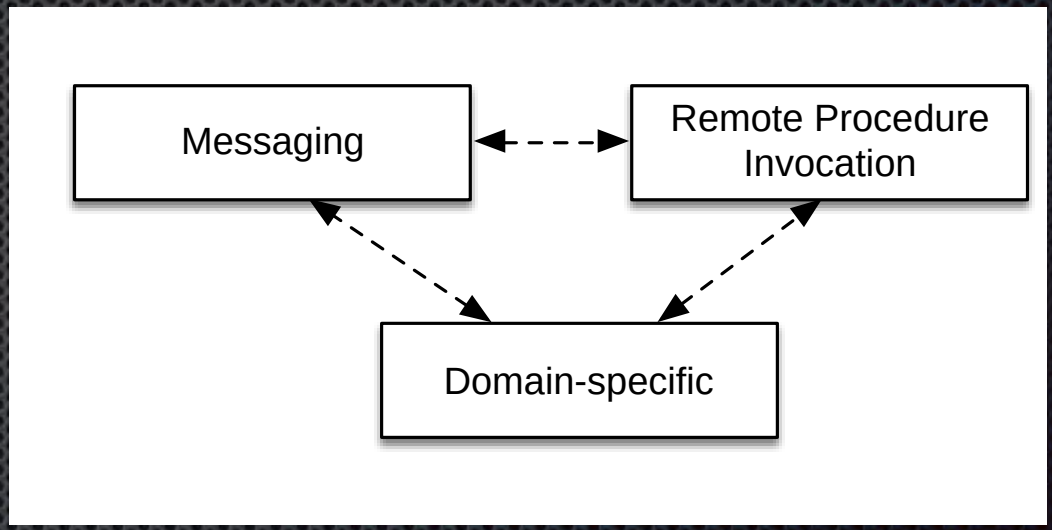
- Multiple languages
- Isolated
- Constrained
- Monitor-able
- Reliable
- Efficient



Issue: How do services communicate?

Forces

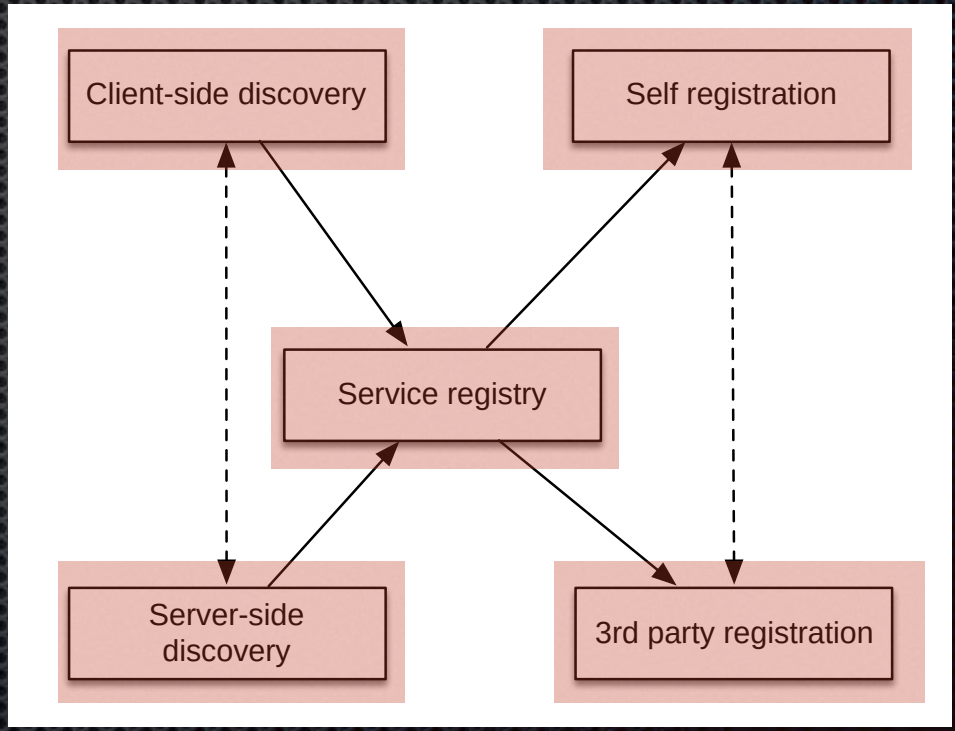
- Services must communicate
- Usually processes on different machines
- …



Issue: How to discover a service instance's network location?

Forces

- Client needs IP address of service instance
- Dynamic IP addresses
- Dynamically provisioned instances



Issue: How to handle cross cutting concerns?

Forces

- Every service must implement logging; externalize configuration; health check endpoint; metrics; …

Microservice Chassis

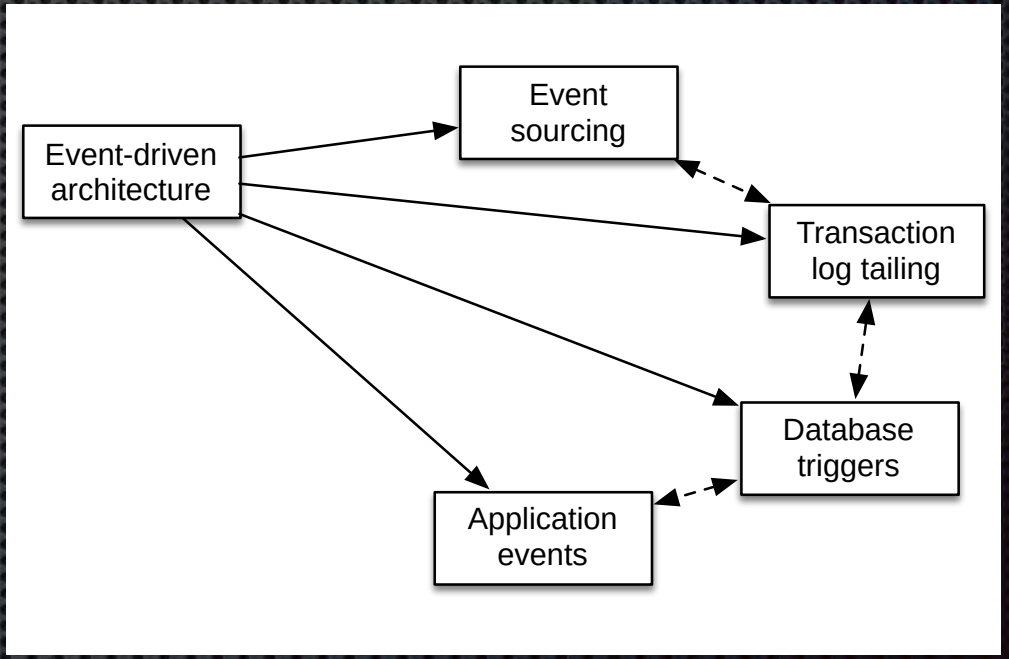
Issue: how to maintain data consistency?

Context

- Each service has its own database
- Data is private to a service

Forces

- Transactional data consistency must be maintained across multiple services
- 2PC is not an option



Issue: how to perform queries?

Context

Each service has its own database

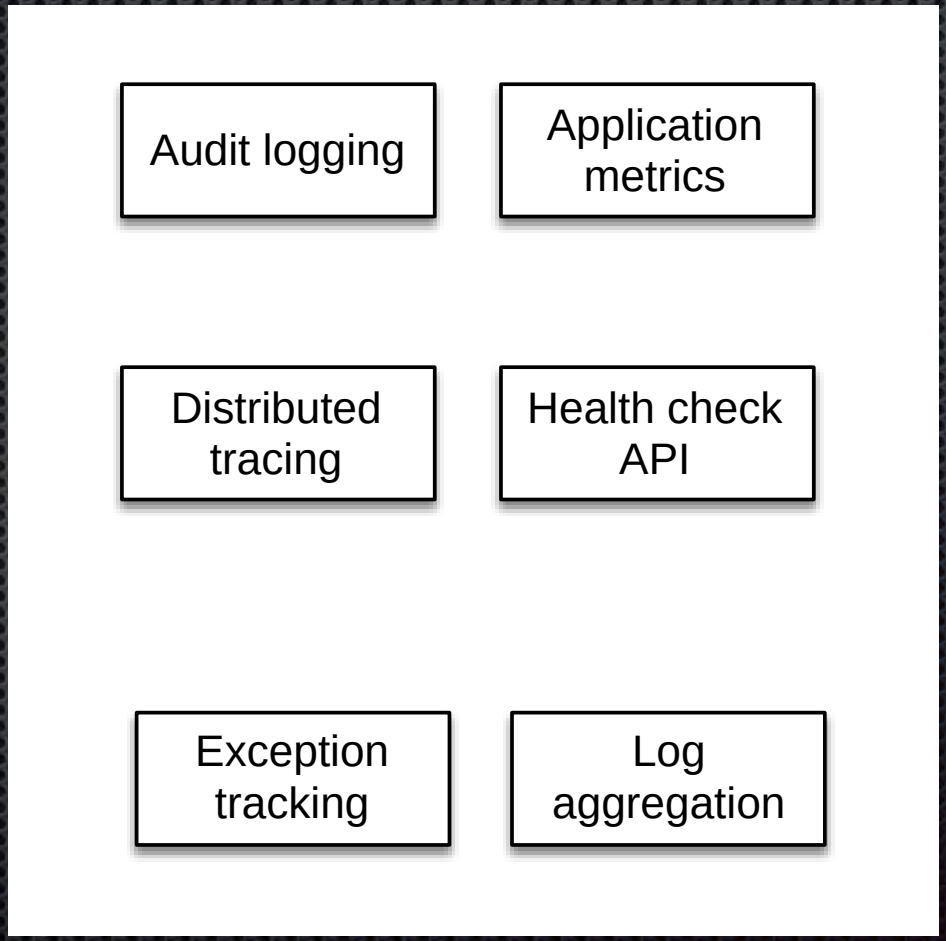
Forces

- Queries must join data from multiple services
- Data is private to a service

CQRS

Maintain query views by subscribing to events

Issue: how to monitor the behavior of your application?



Summary

- The monolithic architecture is a good choice for small/simple applications
- Use the microservice architecture for large/complex applications
- Micro service architecture != silver bullet
- Use the microservice architecture pattern language to guide your decision making


@crichardson chris@chrisrichardson.net
Thank you!
http://learnmicroservices.io