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

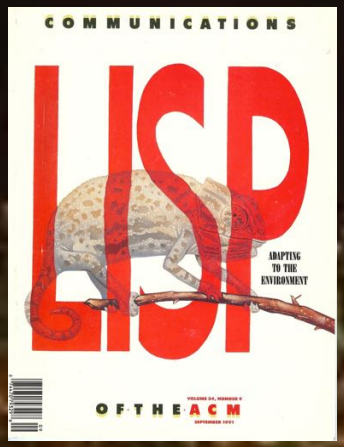
microxchg 2016

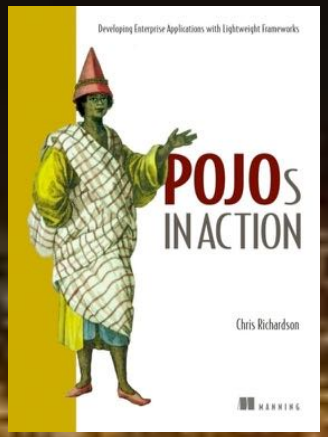
Presentation goal

Why patterns and pattern languages?

A pattern language for microservices

About Chris


COMMUNICATIONS
LISP
ADAPTING TO THE ENVIRONMENT
OF THE ACM


Developing Enterprise Applications with Lightweight Frameworks
POJOs IN ACTION
Chris Richardson
MANNING


CLOUDFOUNDRY
SIGN IN
SIGN UP
SYSTEM ALERT. PLEASE READ: Cloud Foundry will be moving to a new URL.
The Enterprise Java Cloud
Real Java Applications Deployed in Minutes
Built for Spring and Grails Web Applications
Most Widely Used Technologies Delivered as a Platform
SIGN UP
LEARN MORE
CLOUDFOUNDRY
APPLICATION DEMO
Deploying Web Applications To Amazon EC2 with Cloud Foundry


vmware
spring
source

About Chris

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

About Chris

Founder of Eventuate, Inc
Creating a platform that makes it easy for application developers to write microservices
(http://eventuate.io)

For more information

- https://github.com/cer/event-sourcing-examples
- http://microservices.io
- http://plainoldobjects.com/
- https://twitter.com/crichardson
- http://eventuate.io/

Agenda

- Why a pattern language for microservices?
- Core patterns
- Deployment patterns
- Communication patterns and more

In 1986...

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

Yet almost 30 years later developers are still passionately arguing over “silver bullets”

Popular silver bullets

Functional programming

Suck/Rock Dichotomy

JavaScript vs. Java

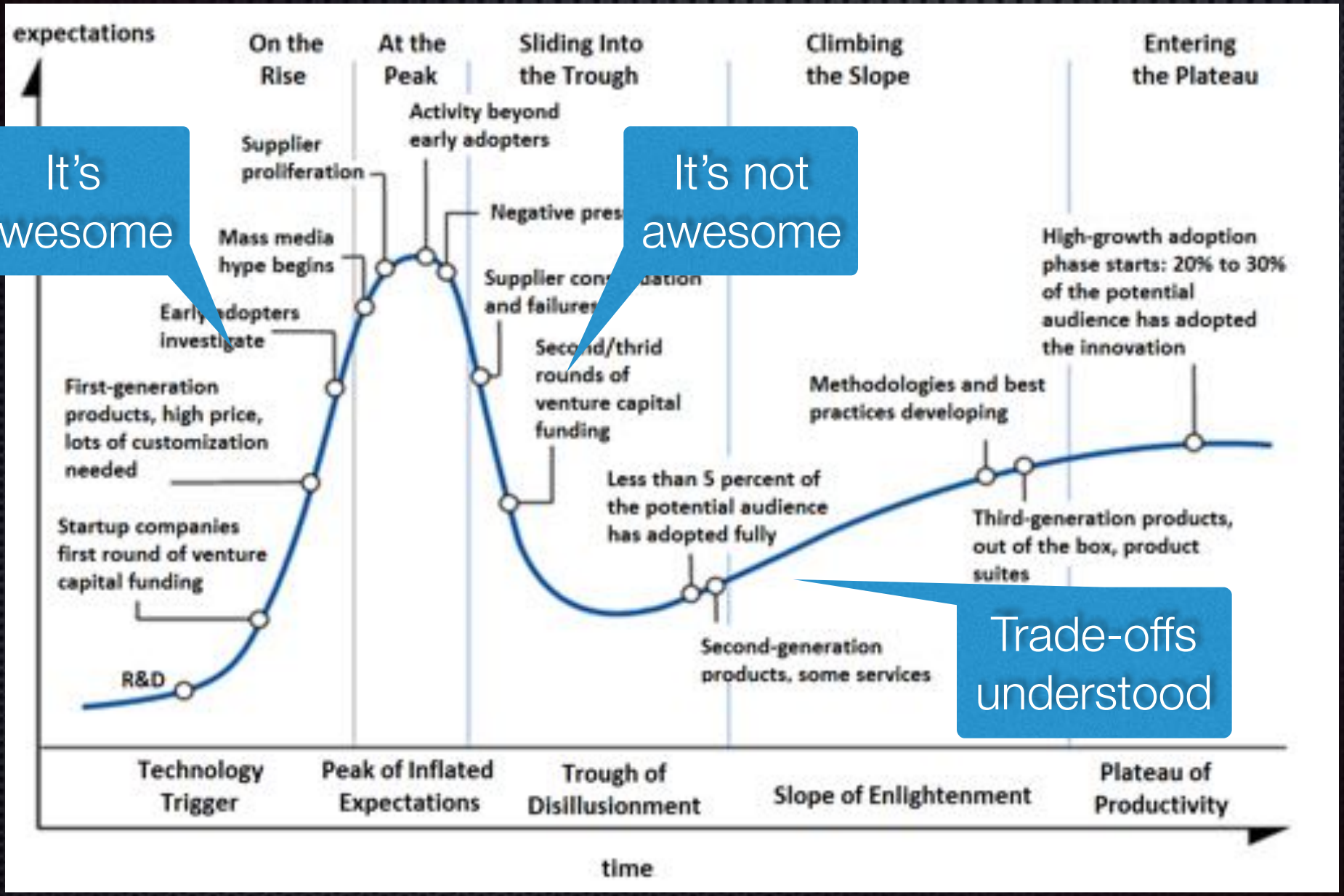
Spring vs. Java EE

Functional programming vs. Object-oriented

Containers vs. Virtual Machines

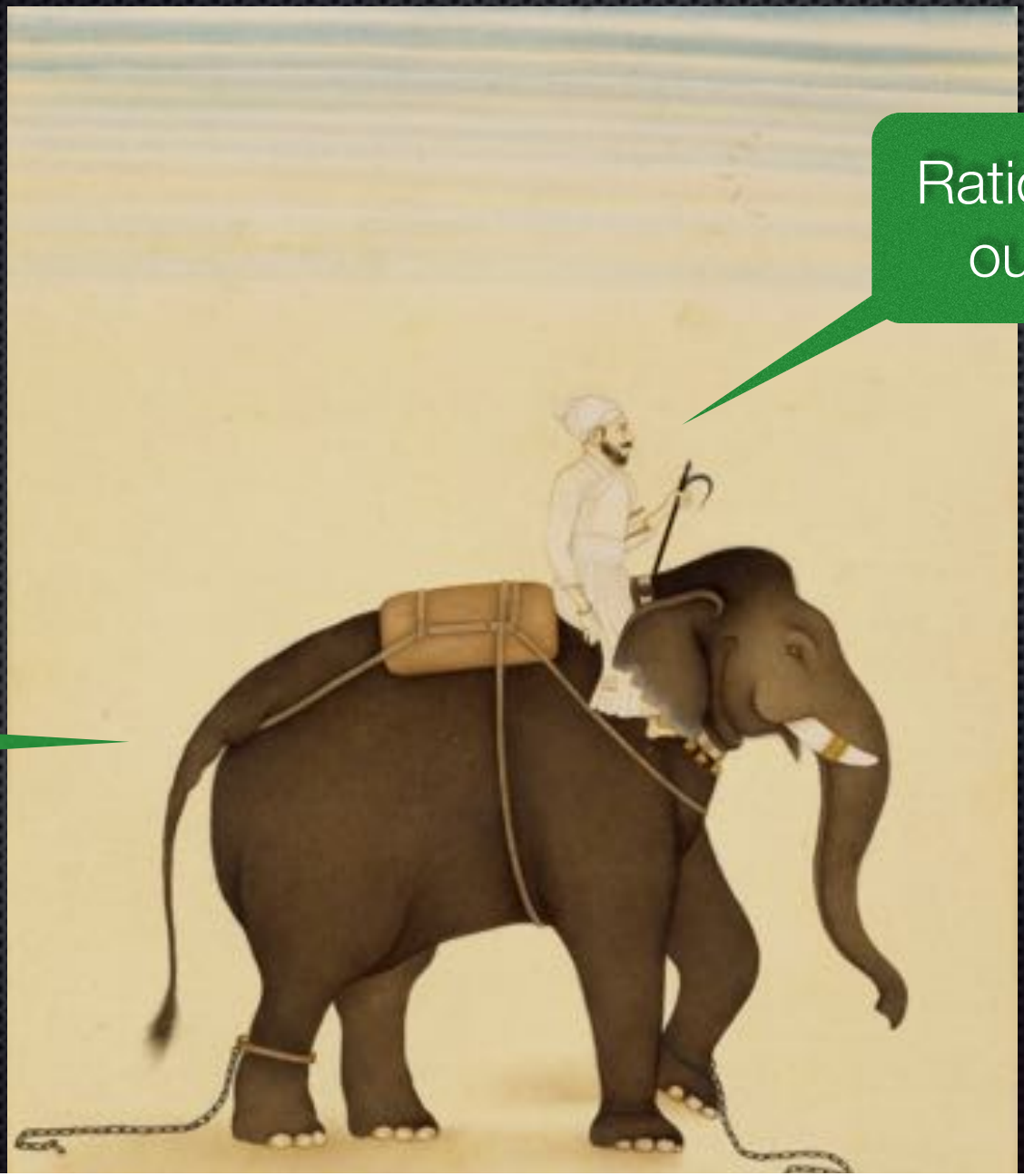
http://nealford.com/memeagora/2009/08/05/suck-rock-dichotomy.html

Gartner Hype Cycle



http://upload.wikimedia.org/wikipedia/commons/b/bf/Hype-Cycle-General.png

How we make decisions



Rationalize with our intellect

Decide using emotions

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

We need a better way
to discuss and think
about technology

What's a pattern?

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

The structure of a pattern

=

Great framework for discussing and thinking about technology

The structure of a pattern

Name

Context

aka the situation

Problem

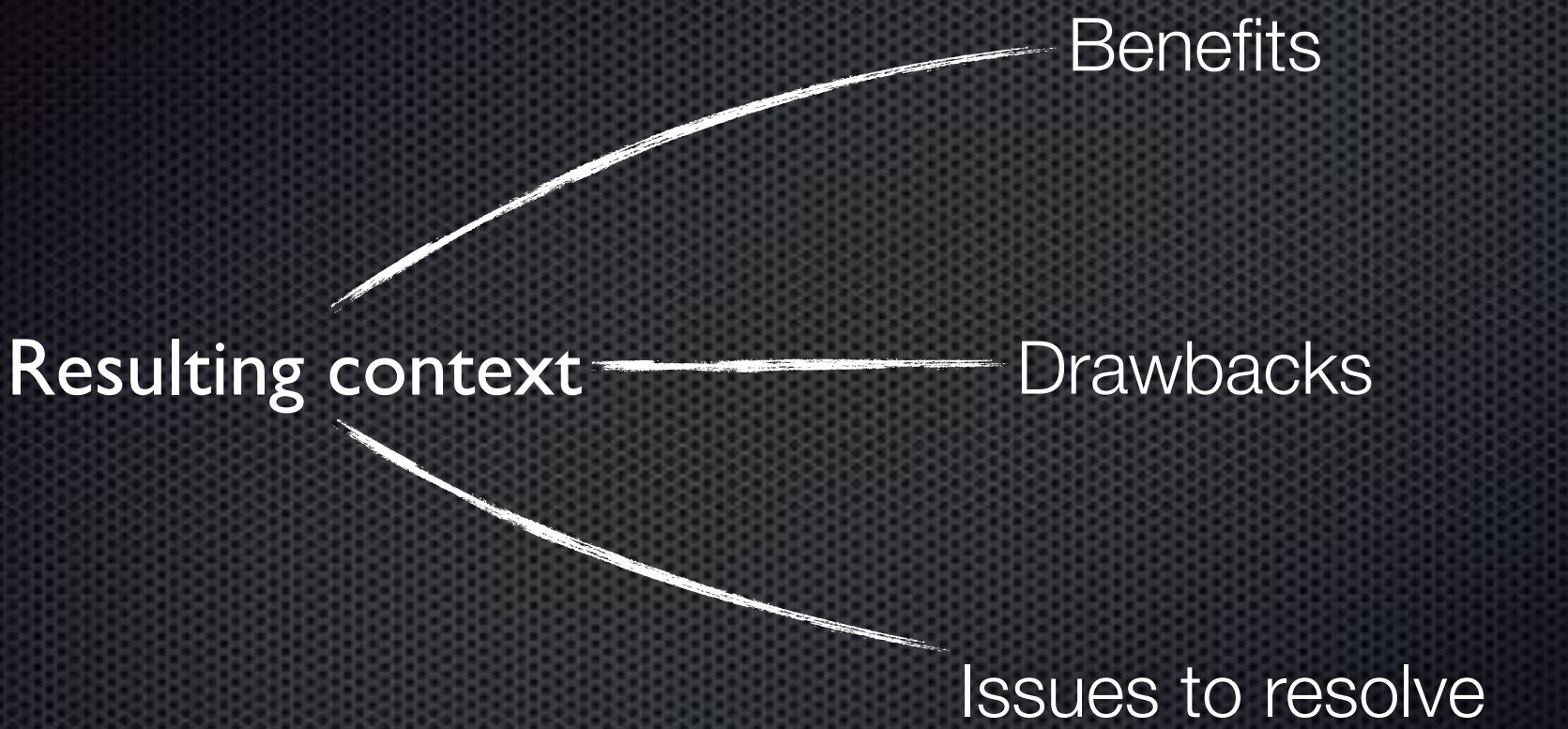
(conflicting) issues etc to address

Forces

Solution

Resulting context

Related patterns


Benefits
Resulting context
Drawbacks
Issues to resolve

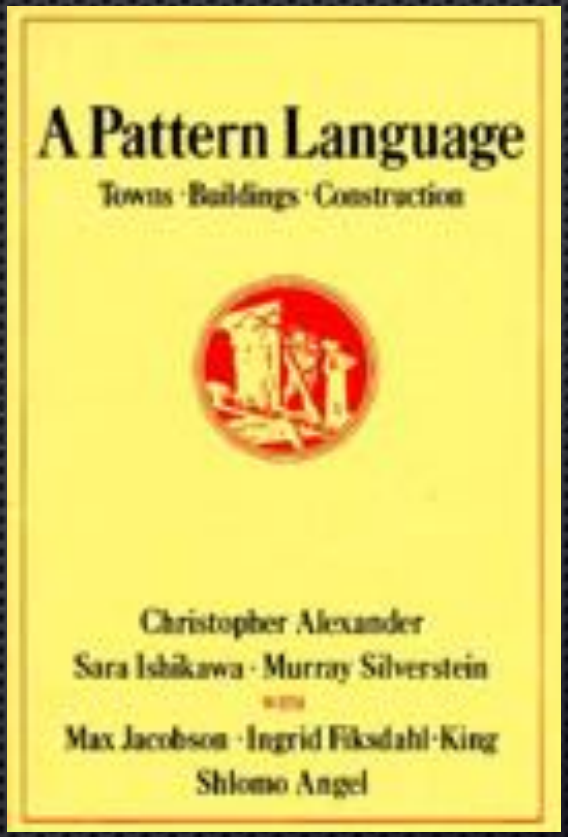
Alternative solutions

Related patterns

Solutions to problems
introduced by this pattern

Pattern language

- A collection of related patterns that solve problems in a particular domain
- Relationships
  - Pattern A results in a context that has a problem solved by Pattern B
  - Patterns A and B solve the same problem
  - Pattern A is a specialization of pattern B



Access to Water

Promenade

Local townhall

Intimacy gradient

Light on two sides

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

Meta-pattern

Problem: How to talk/reason about technology?

Context: Emotional software development culture

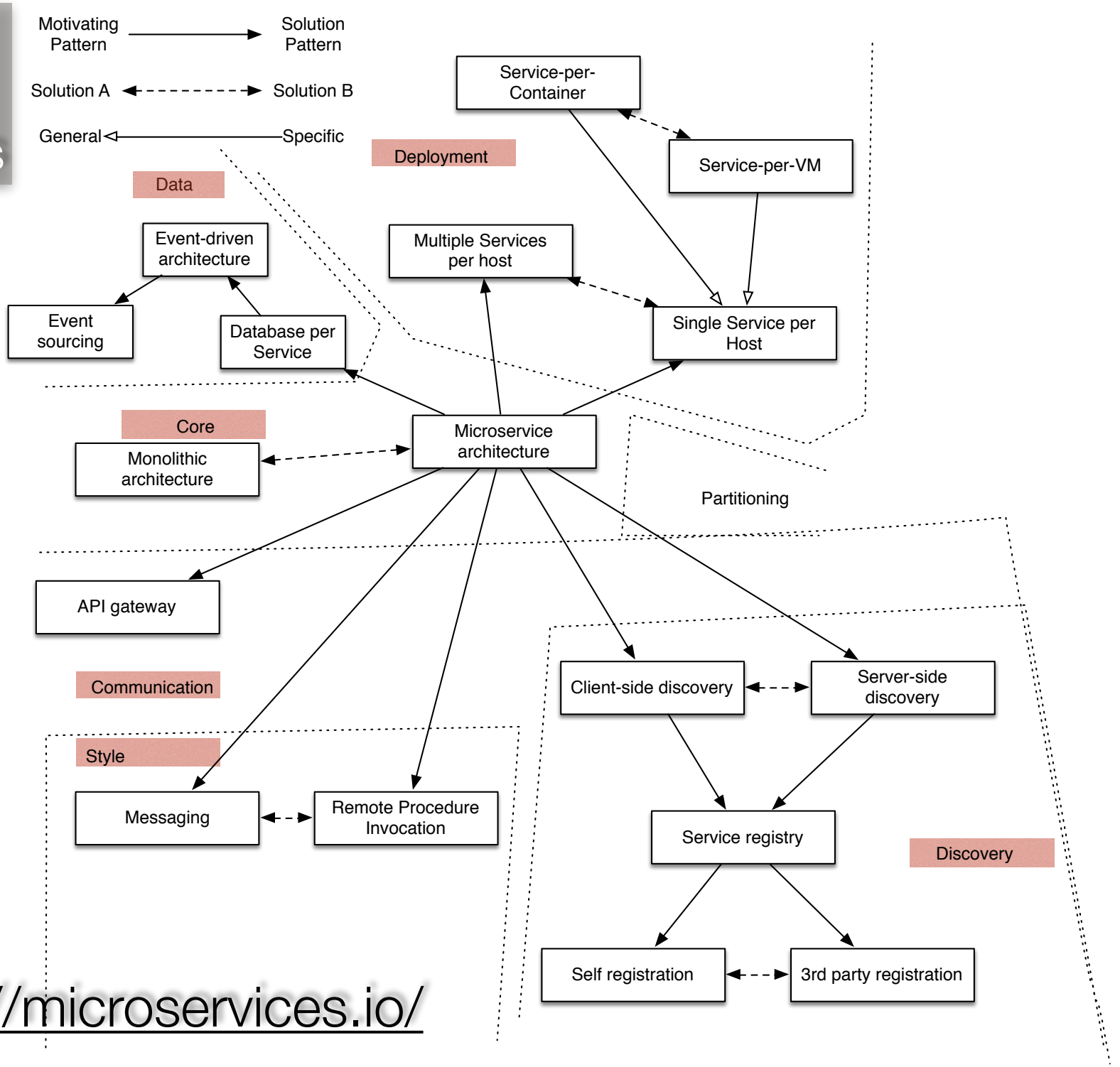
Solution: Use the pattern format

Benefit: More objective

Drawback: Less exciting

Related patterns: It's awesome!

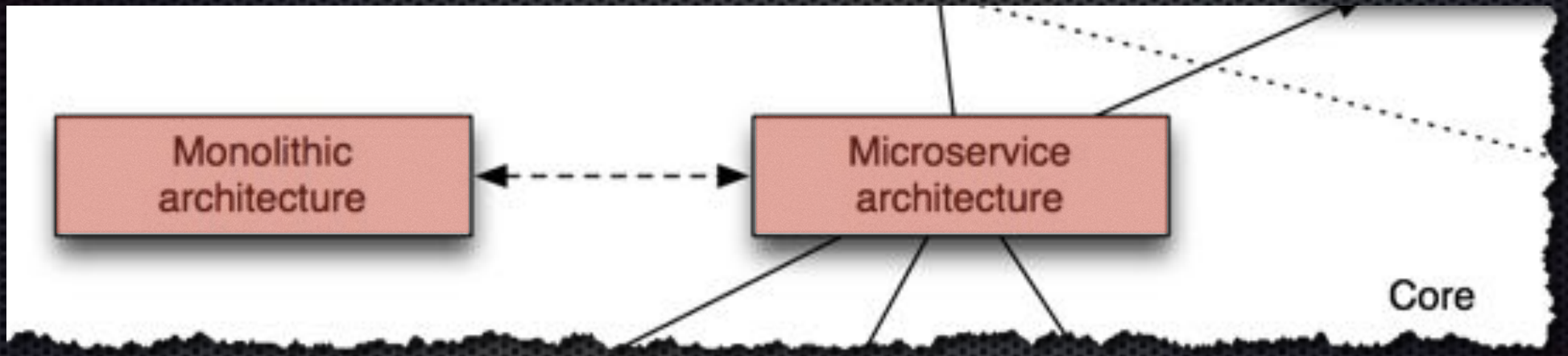
Work in progress



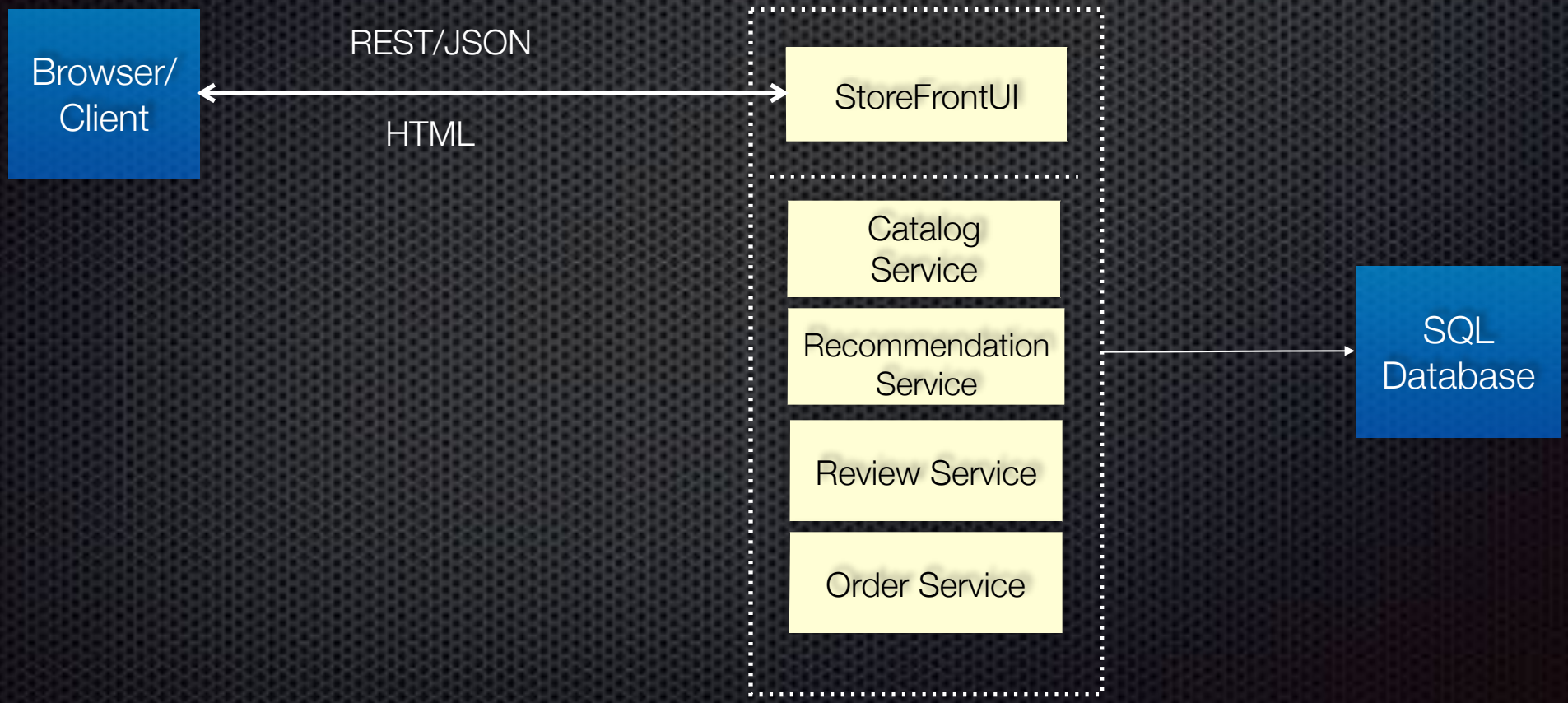
http://microservices.io/

Agenda

- Why a pattern language for microservices?
- Core patterns
- Deployment patterns
- Communication patterns and more


Monolithic
architecture
Microservice
architecture
Core

Let's imagine you are building an online store



Problem: what's the deployment architecture?

Forces

Businesses must innovate faster

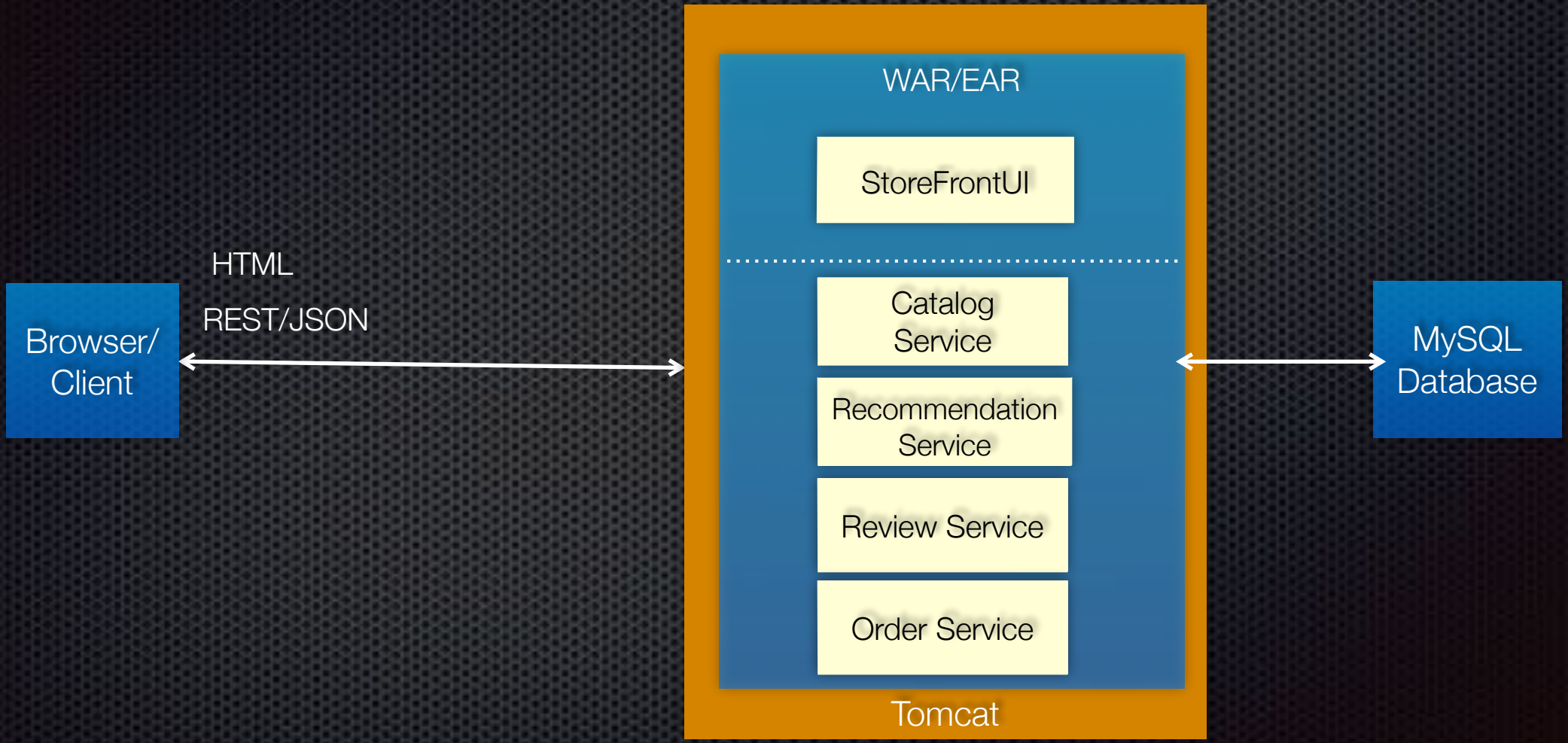
⇒

Develop more complex, higher-quality software faster

Forces

- Developers must be productive
- The code must be easy to understand and modify
- Continuous deployment of applications
- Developers must leverage emerging technologies (frameworks, programming languages, etc)
- Deploy multiple application instances for scalability and availability

Pattern: Monolithic architecture



Examples everywhere

Simple to ….

Develop
Test
Deploy
Scale

Successful applications have a habit of growing

⇒

Big, complex, monolithic applications

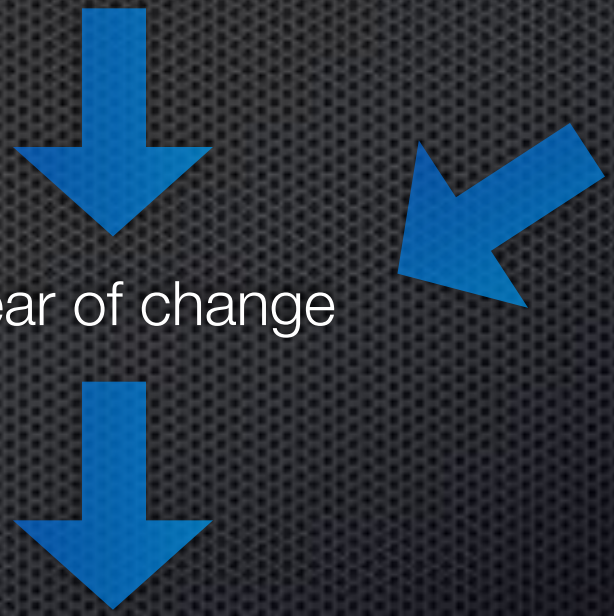
Agile development and deployment becomes impossible

Intimidates developers

Obstacle to frequent deployments

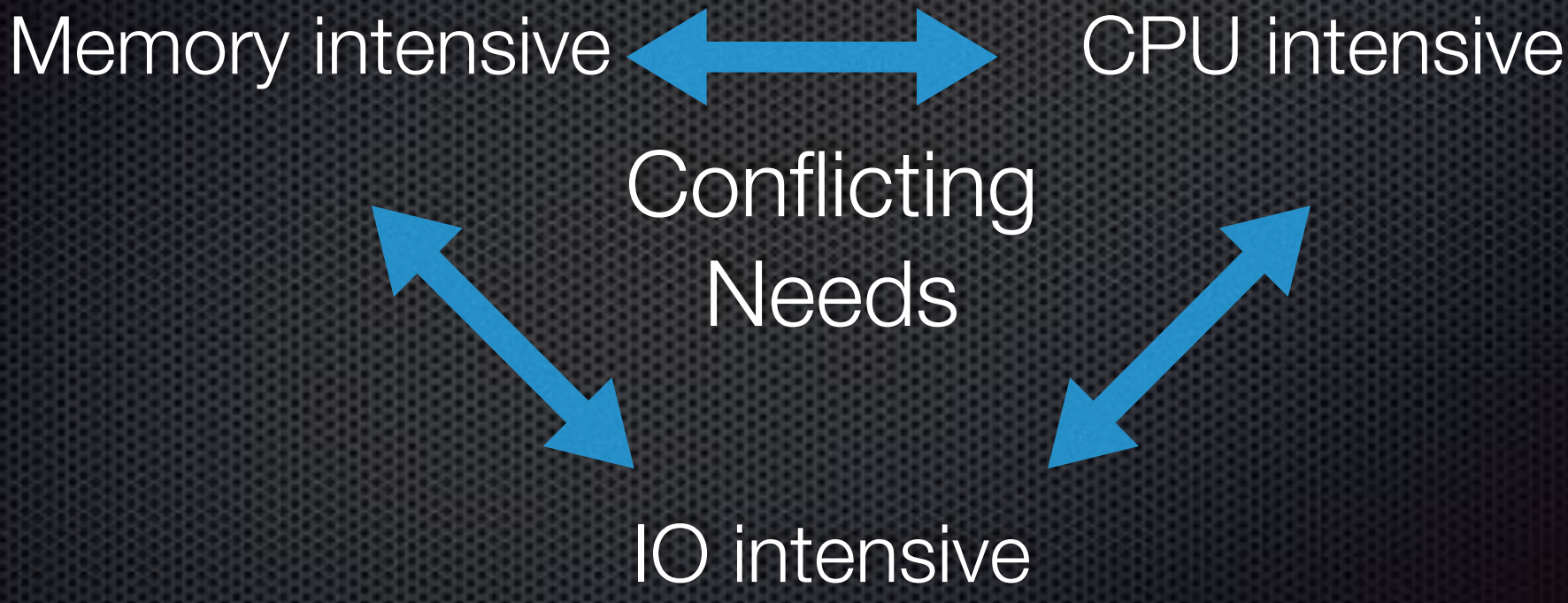
- Need to redeploy everything to change one component
- Interrupts long running background (e.g. Quartz) jobs
- Increases risk of failure

Eggs in one basket



- Updates will happen less often - really long QA cycles
- e.g. Makes A/B testing UI really difficult

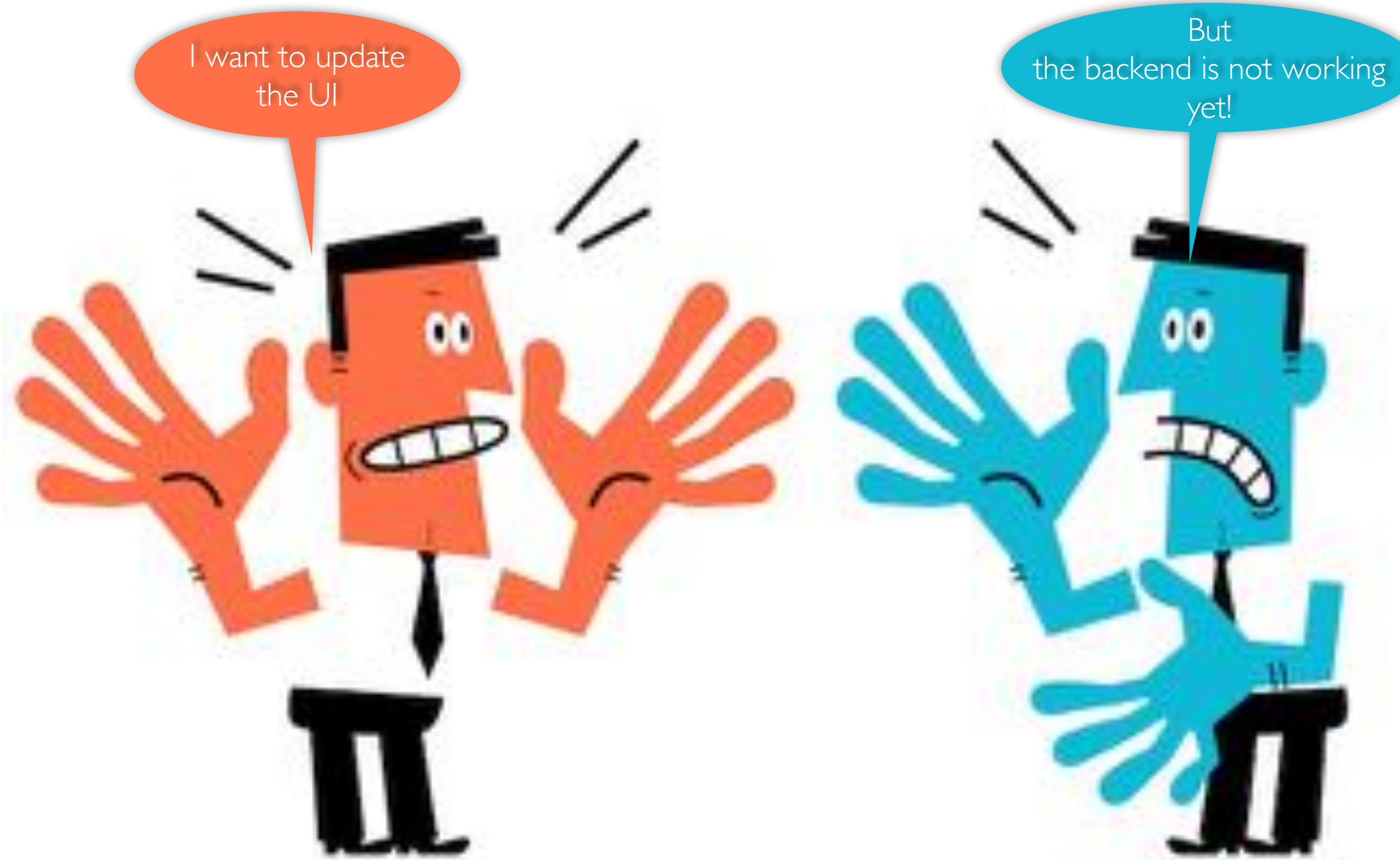
Scaling the application can be challenging



Overloads your IDE and container

Slows down development

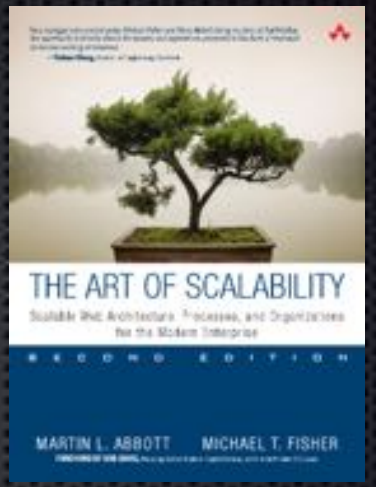
Requires lots of coordination



Requires long-term commitment to a technology stack

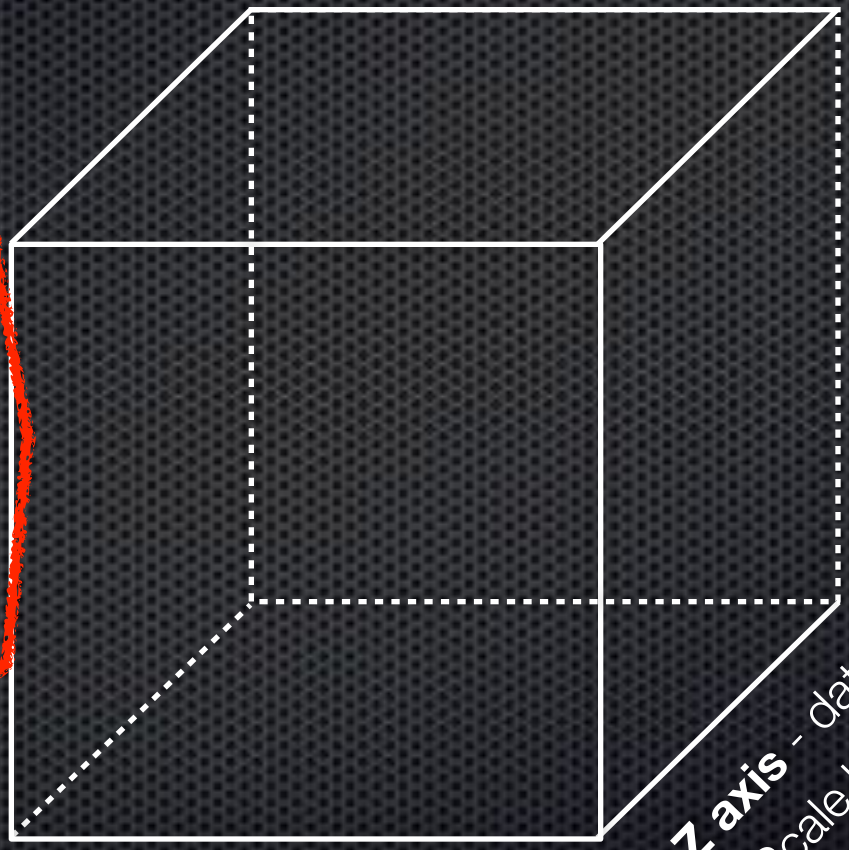
Pattern: Microservice architecture

Apply functional decomposition



Y axis - functional decomposition

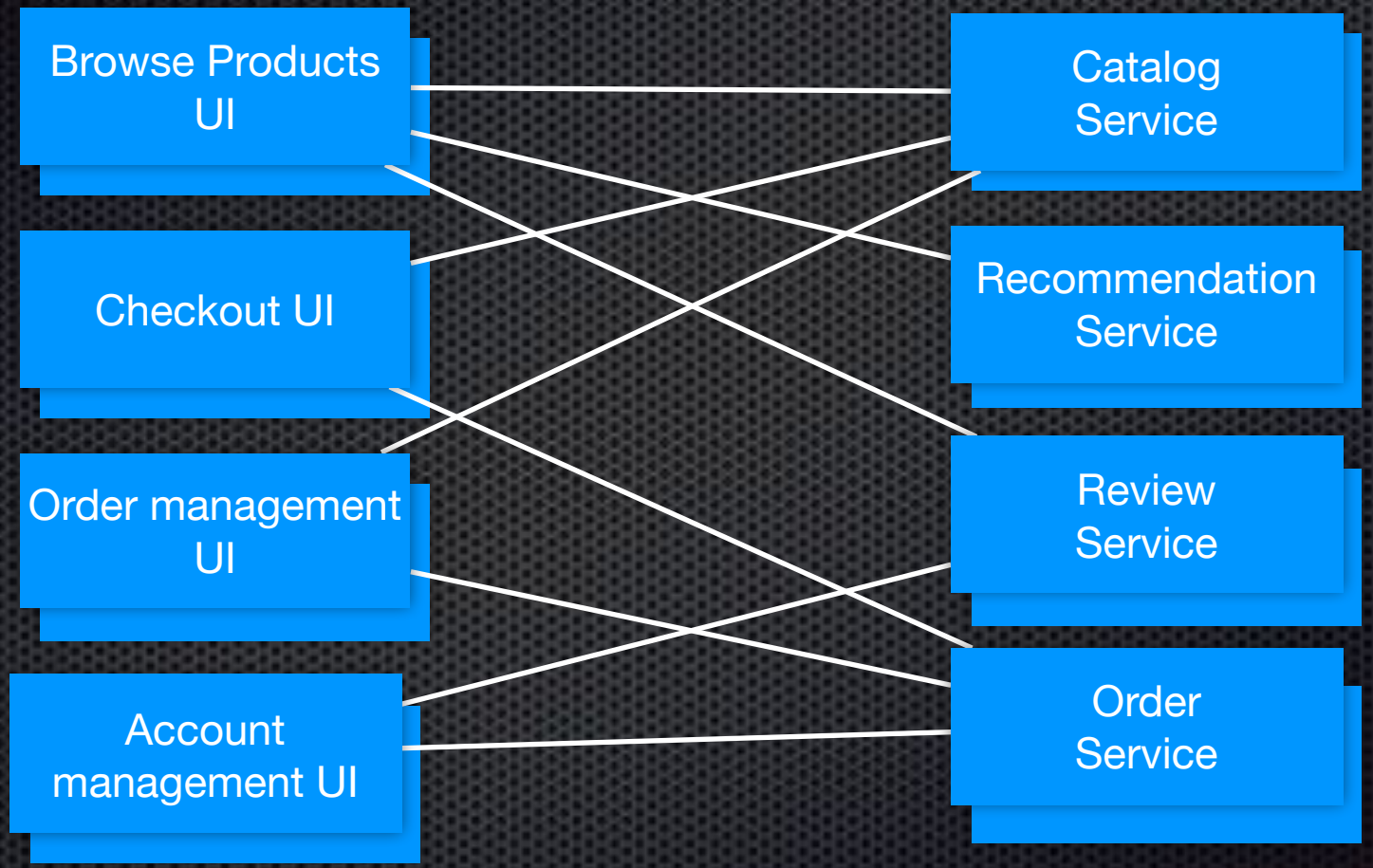
Scale by splitting different things



Z axis - data partitioning
Scale by splitting similar things

X axis
- horizontal duplication

Microservice architecture



Apply X-axis and Z-axis scaling to each service independently

Goal of partitioning:

Most changes affect only a single service

⇒

Develop and deploy independently in parallel with other changes

Goal of partitioning:

Separate components with conflicting resource requirements

⇒

Scale more easily

Examples of microservices

http://techblog.netflix.com/

~600 services

http://highscalability.com/amazon-architecture

100-150 services to build a page

http://www.addsimplicity.com/downloads/eBaySDForum2006-11-29.pdf

http://queue.acm.org/detail.cfm?id=1394128

Examples of microservices

http://bit.ly/msexample_soundcloud

http://bit.ly/msexample_yelp

http://bit.ly/msexample_uber

Benefits

- Smaller, simpler apps
  - Easier to understand and develop
  - Less jar/classpath hell - who needs OSGI?
  - Faster to build and deploy
- Scales development: develop, deploy and scale each service independently
- Improves fault isolation
- Easier scaling
- Eliminates long-term commitment to a single technology stack
  - System level architecture vs. service level architecture
  - Easily and safely experiment with new technologies

Drawbacks

Complexity

Drawbacks

- Complexity of developing a distributed system
  - Implementing inter-process communication
  - Handling partial failures
- Implementing business transactions that span multiple databases (without 2PC)
- Complexity of testing a distributed system
- Complexity of deploying and operating a distributed system
- Managing the development and deployment of features that span multiple services

Fortunately solutions exists

The benefits typically outweigh the drawbacks for large, complex applications

Issues to address

- **How to deploy the services?**
- **How do the services communicate?**
- **How do clients of the application communicate with the services?**
- How to partition the system into services?
- How to deal with distributed data management problems?
- ….

Agenda

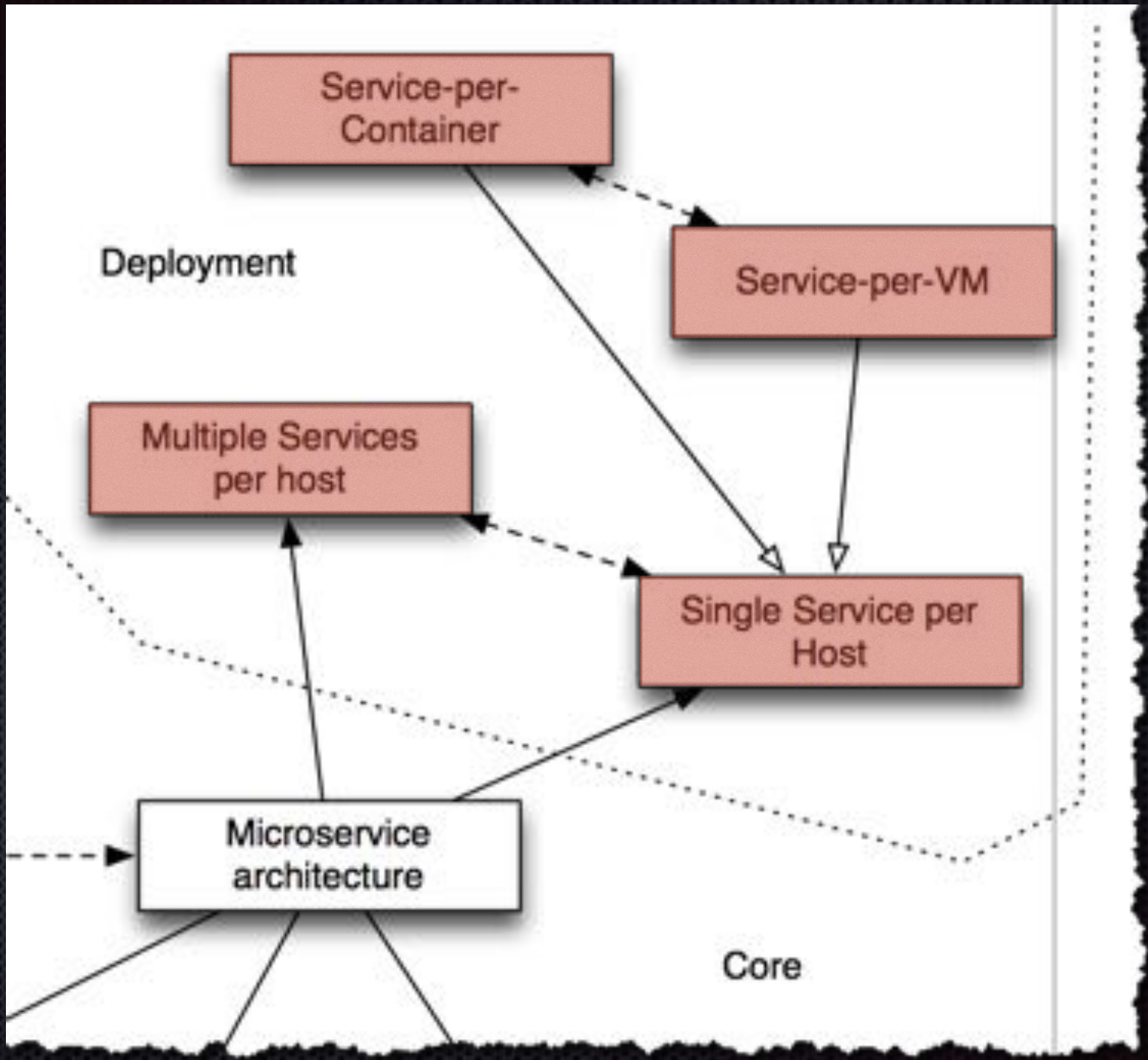
- Why a pattern language for microservices?
- Core patterns
- Deployment patterns
- Communication patterns and more

We have applied the microservices pattern:

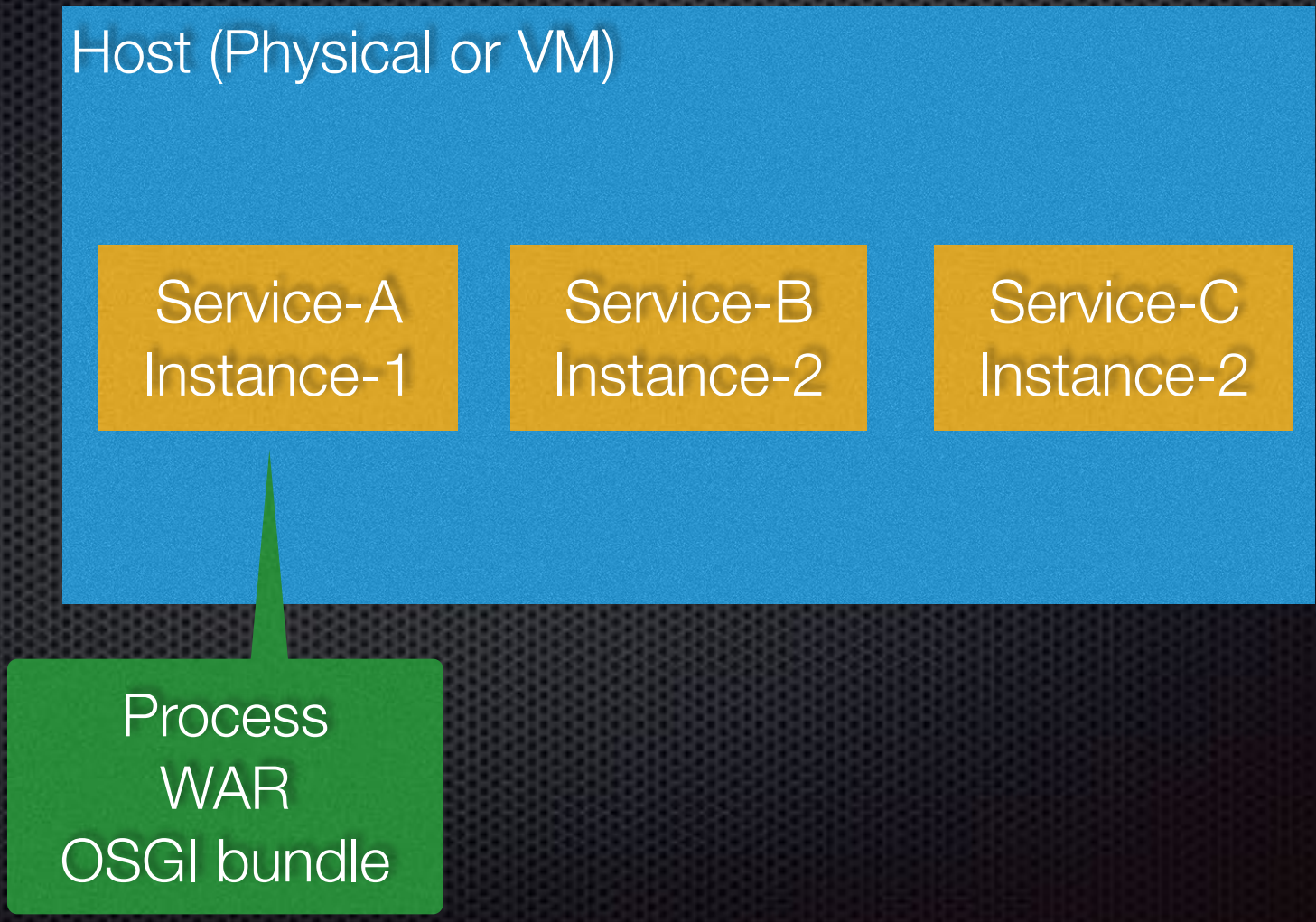
How to deploy the services?

Forces

- Services are written using a variety of languages, frameworks, and framework versions
- Each service consists of multiple service instances for throughput and availability
- Building and deploying a service must be fast
- Service must be deployed and scaled independently
- Service instances need to be isolated
- Resources consumed by a service must be constrained
- Deployment must be reliable
- Deployment must be cost-effective


Service-per-
Container
Deployment
Service-per-VM
Multiple Services
per host
Single Service per
Host
Microservice
architecture
Core

Pattern: Multiple service instances per host



Benefits and drawbacks

Benefits

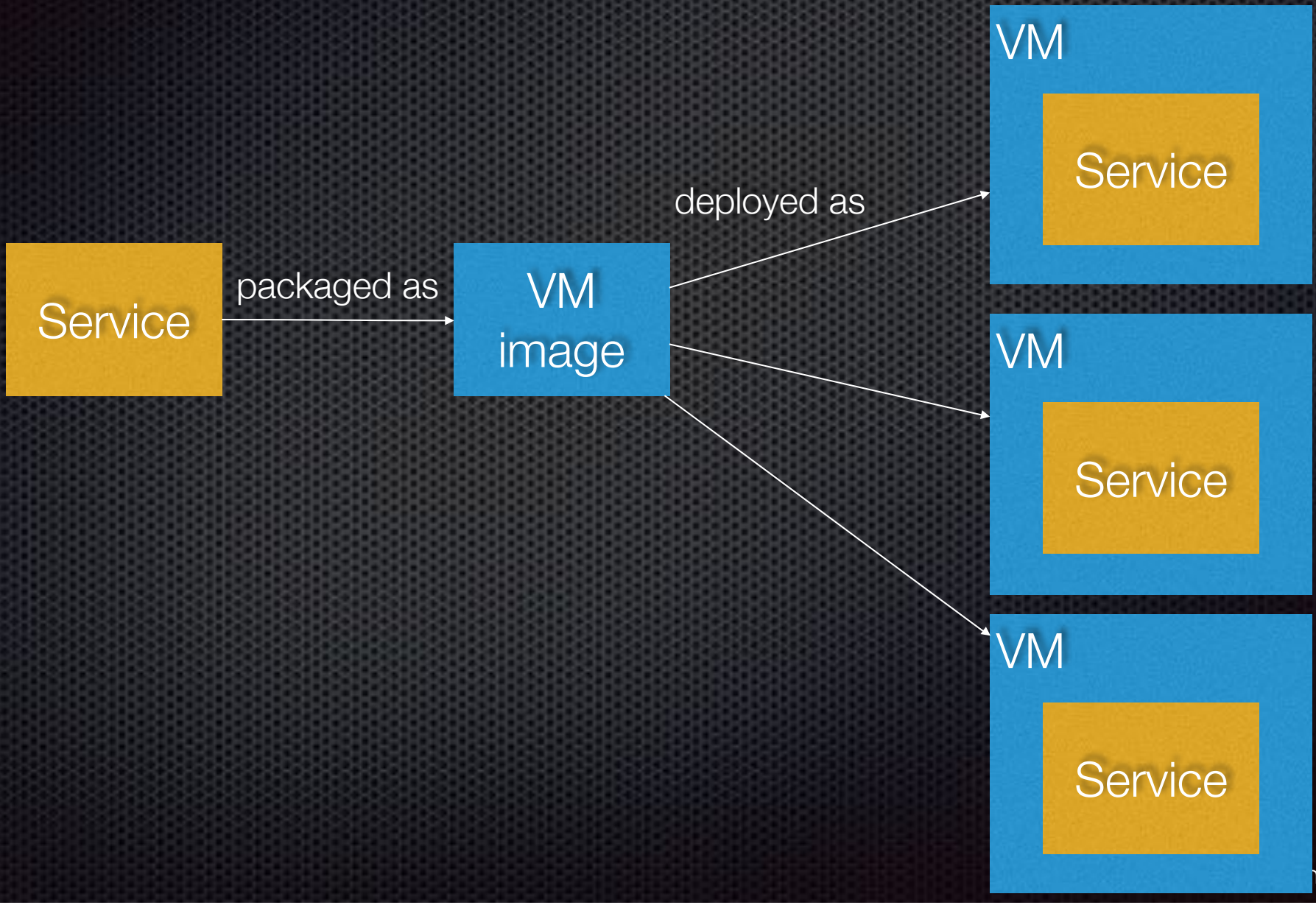
- Efficient resource utilization
- Fast deployment

Drawbacks

- Poor/Terrible isolation
- Poor visibility (with WAR/OSGI deployment)
- Difficult to limit resource utilization
- Risk of dependency version conflicts
- Poor encapsulation of implementation technology

Pattern: Service instance per host

Pattern: Service per VM host



Example

http://techblog.netflix.com/

~600 services

packer.io is a great tool

Benefits and drawbacks

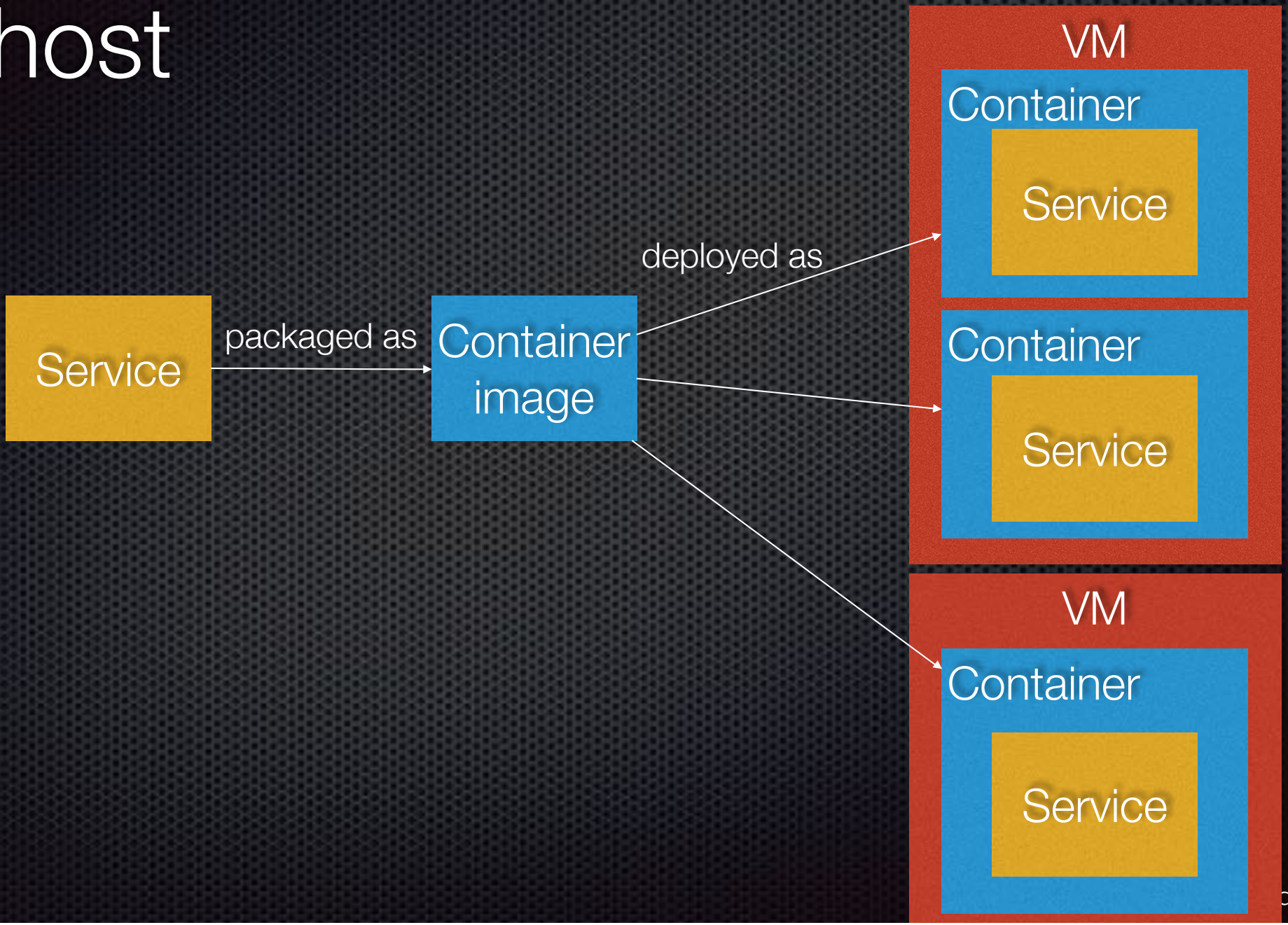
Benefits

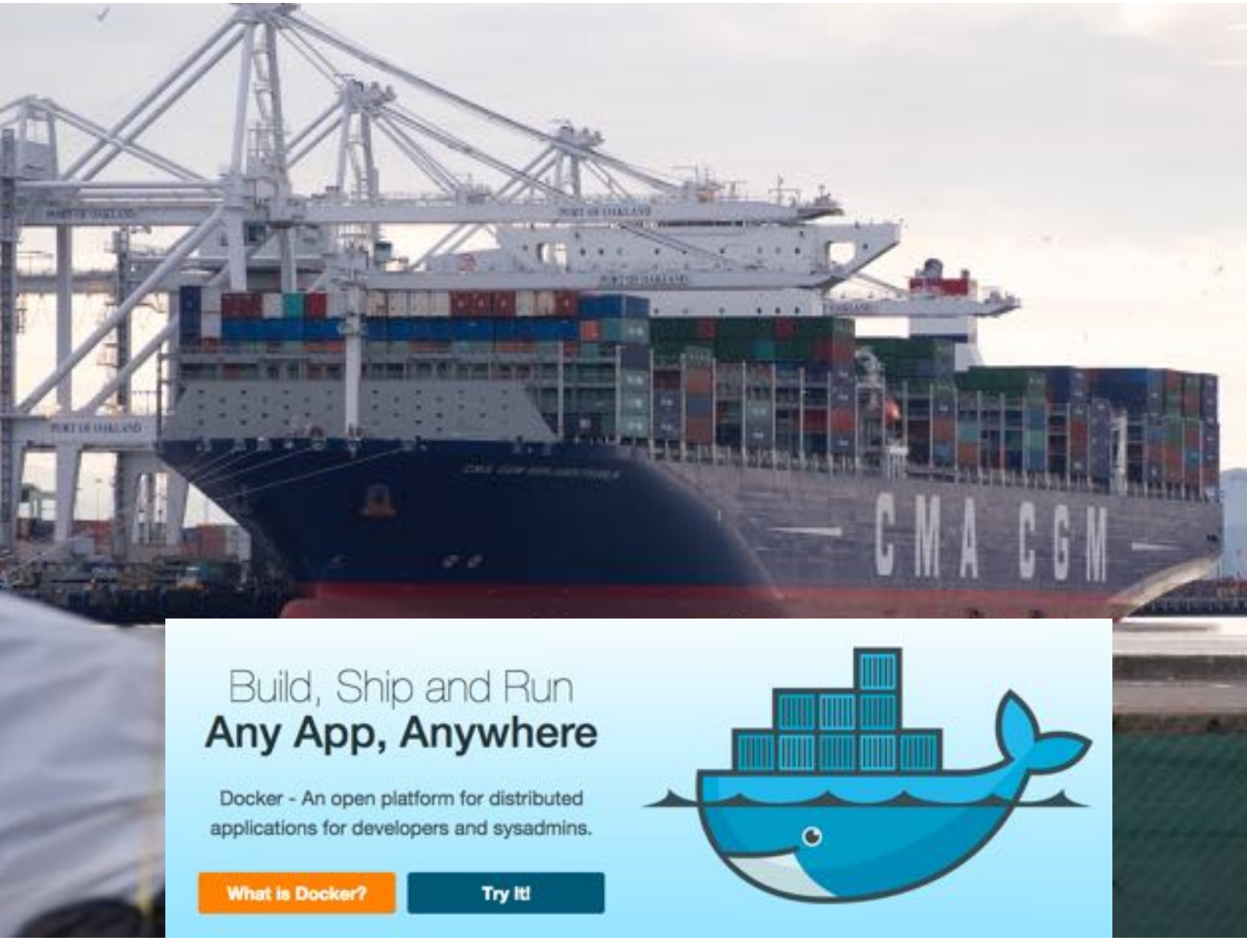
- Great isolation
- Great manageability
- VM encapsulates implementation technology
- Leverage AWS infrastructure for Autoscaling/Load balancing

Drawbacks

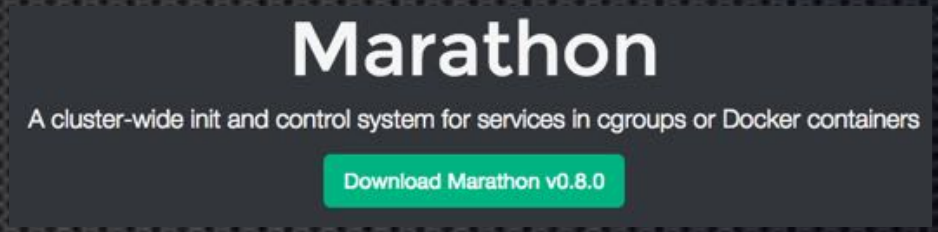
- Less efficient resource utilization
- Slow deployment

Pattern: Service per Container host




CMA CGM
Build, Ship and Run
Any App, Anywhere
Docker - An open platform for distributed applications for developers and sysadmins.
What is Docker?
Try It!

Docker clustering solutions



https://www.dchq.io/landing/index.html

Benefits and drawbacks

Benefits

- Great isolation
- Great manageability
- Container encapsulates implementation technology
- Efficient resource utilization
- Fast deployment

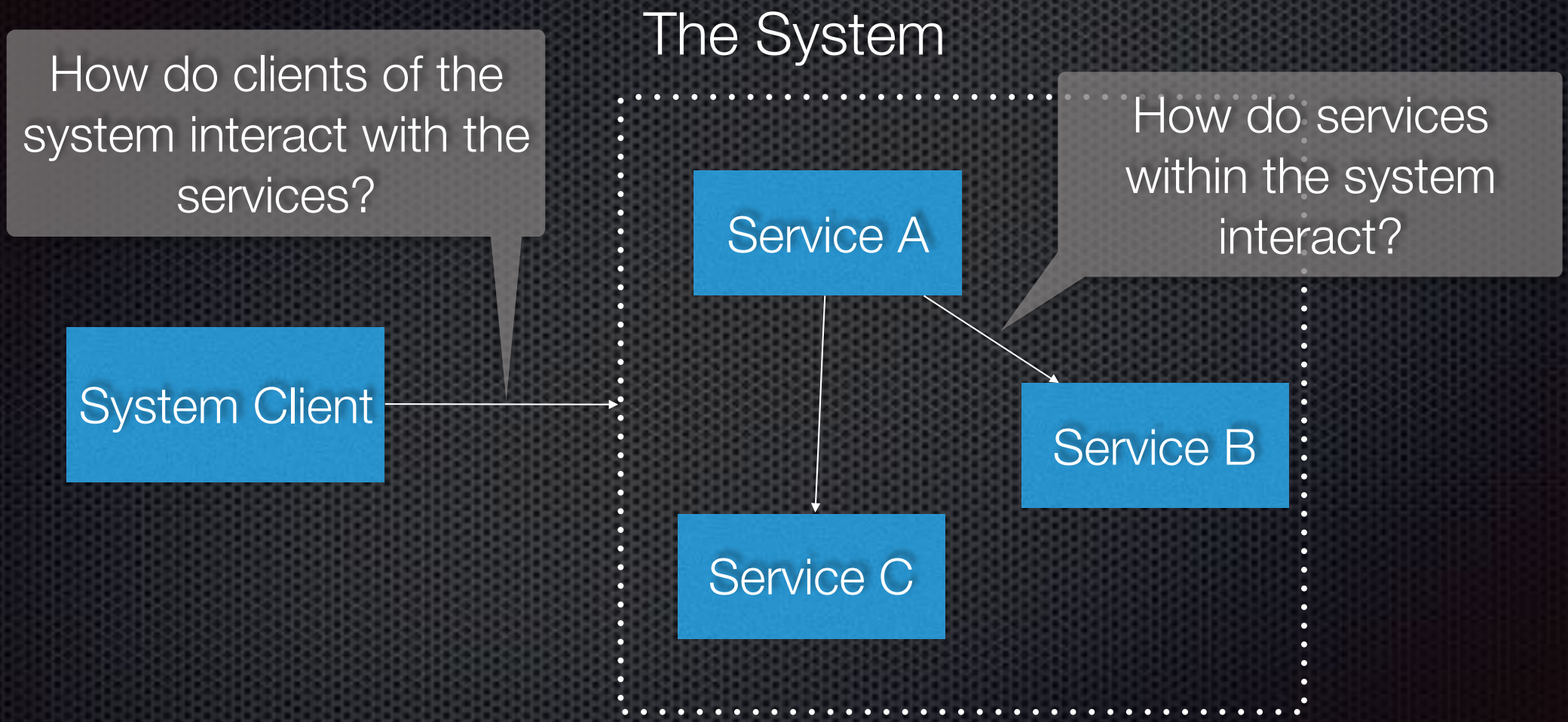
Drawbacks

- Immature infrastructure for deploying containers

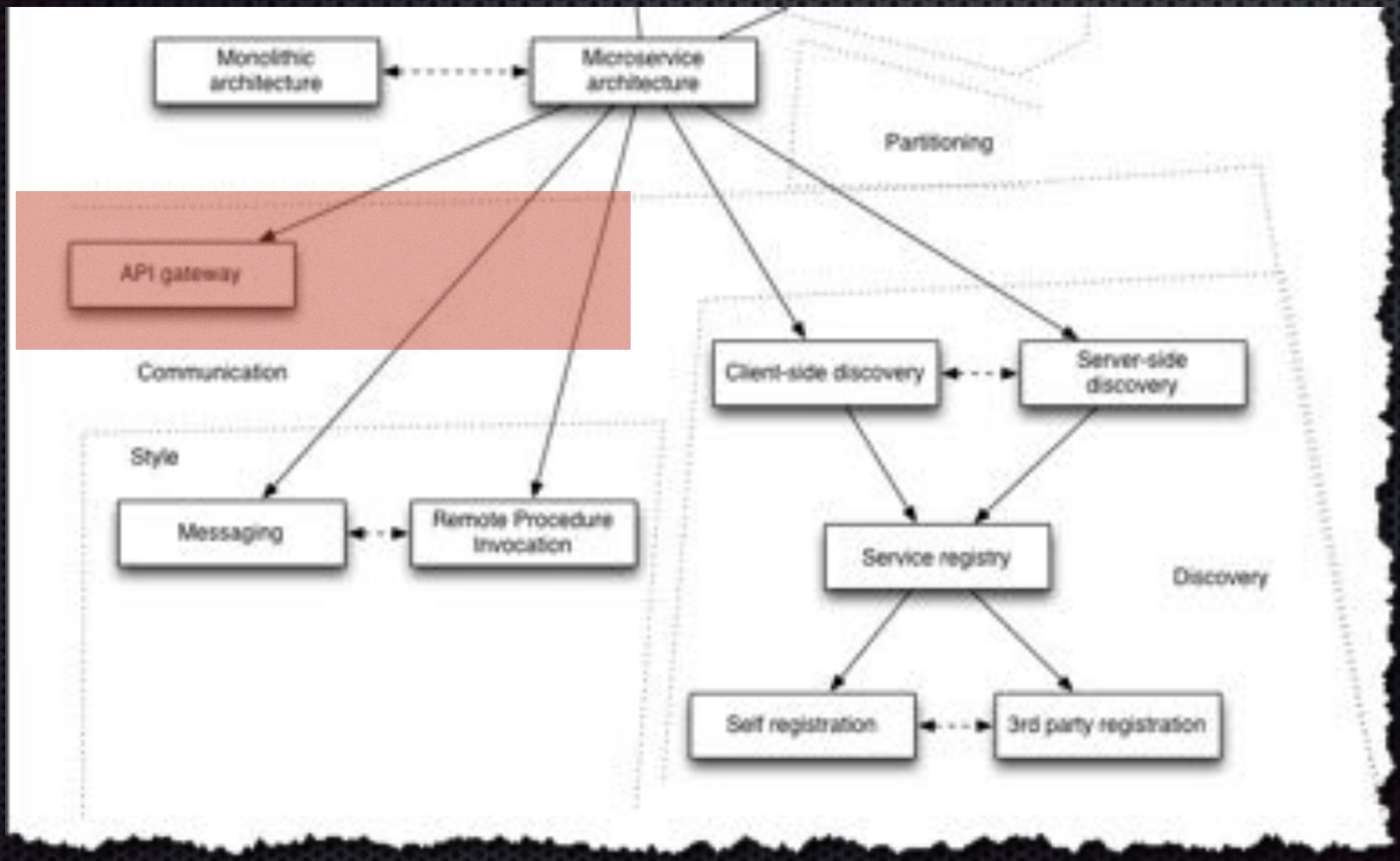
Agenda

- Why a pattern language for microservices?
- Core patterns
- Deployment patterns
- Communication patterns and more

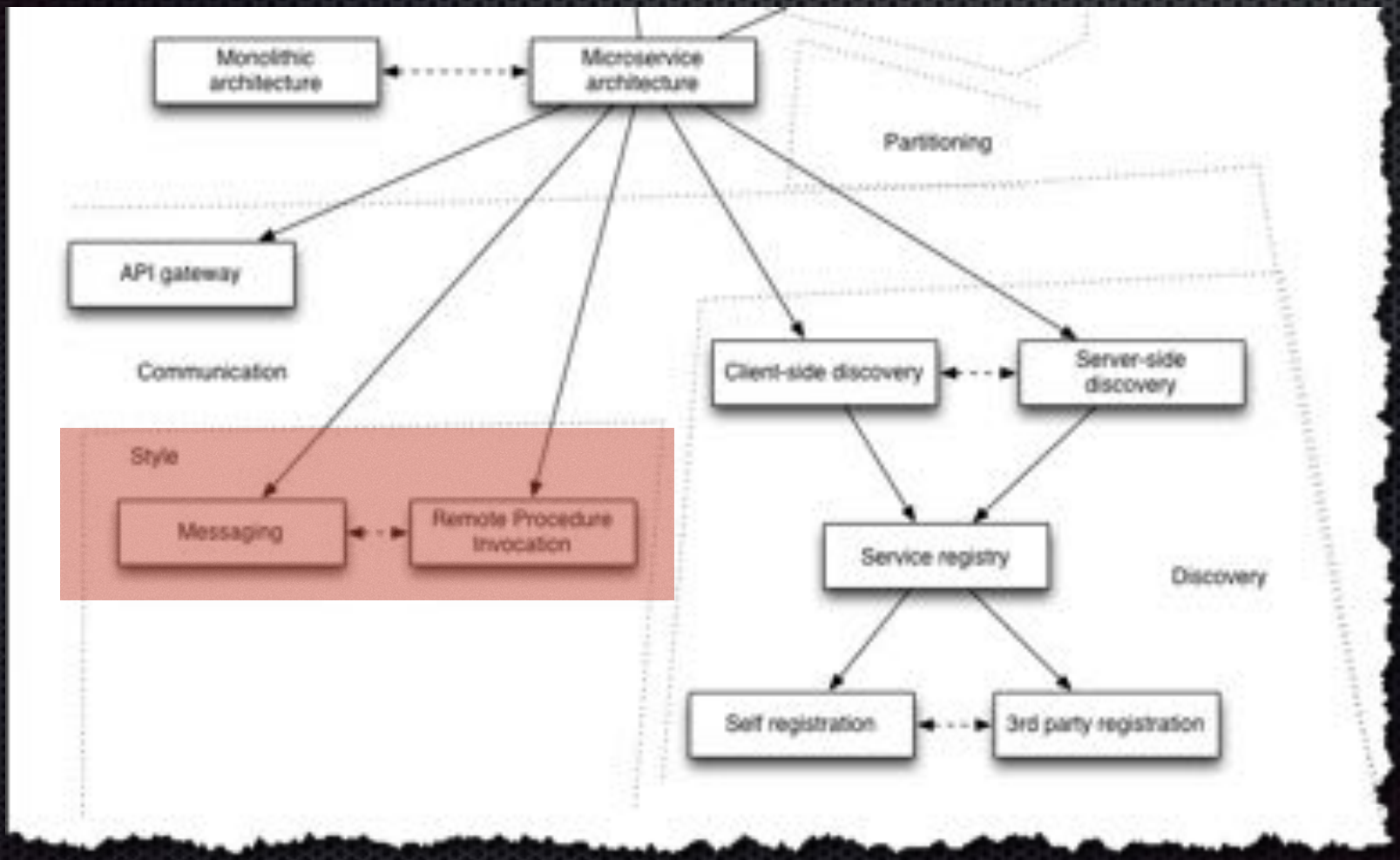
Communication issues



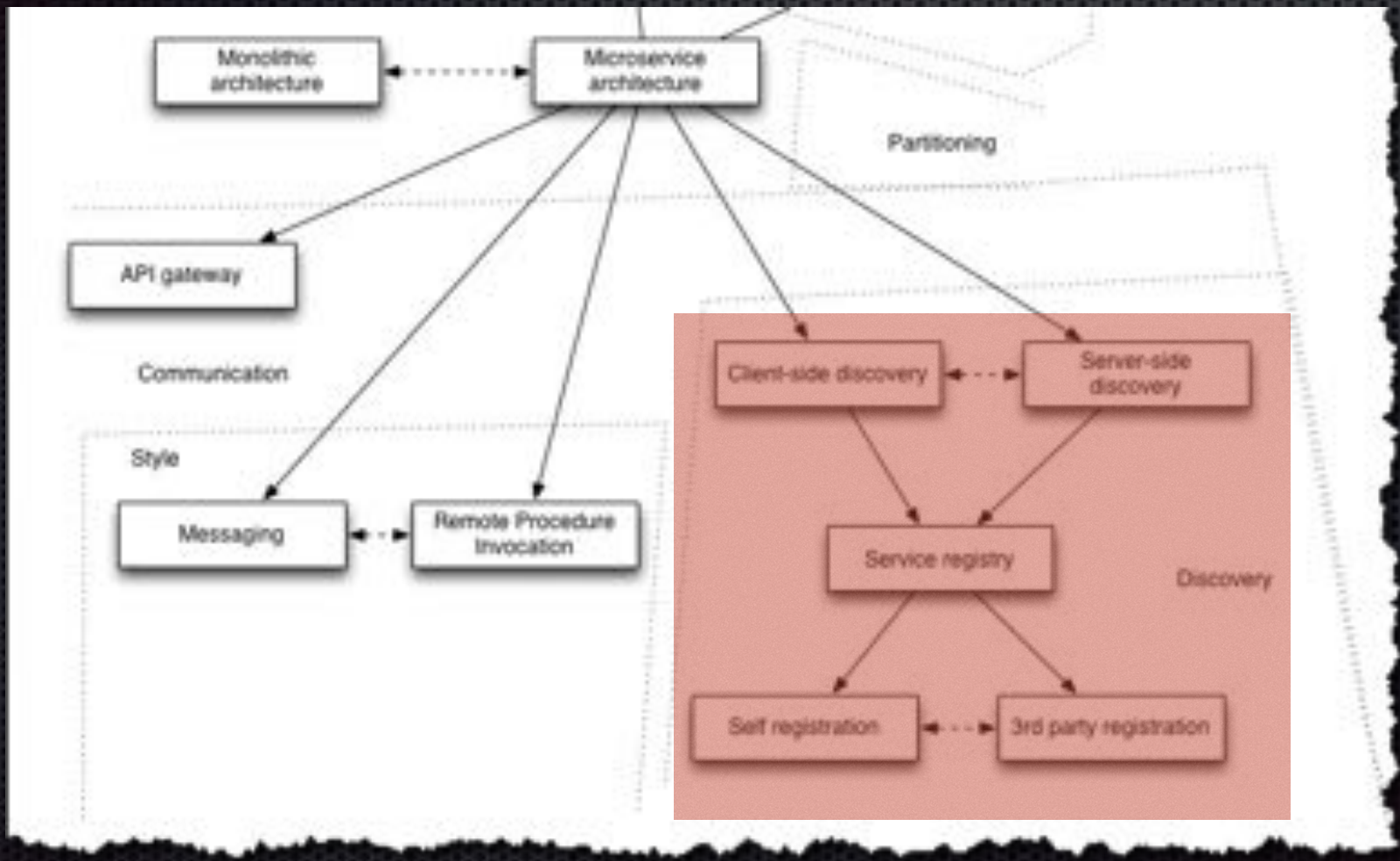
How do external clients interact with the microsevices?



How do services communicate with each other?

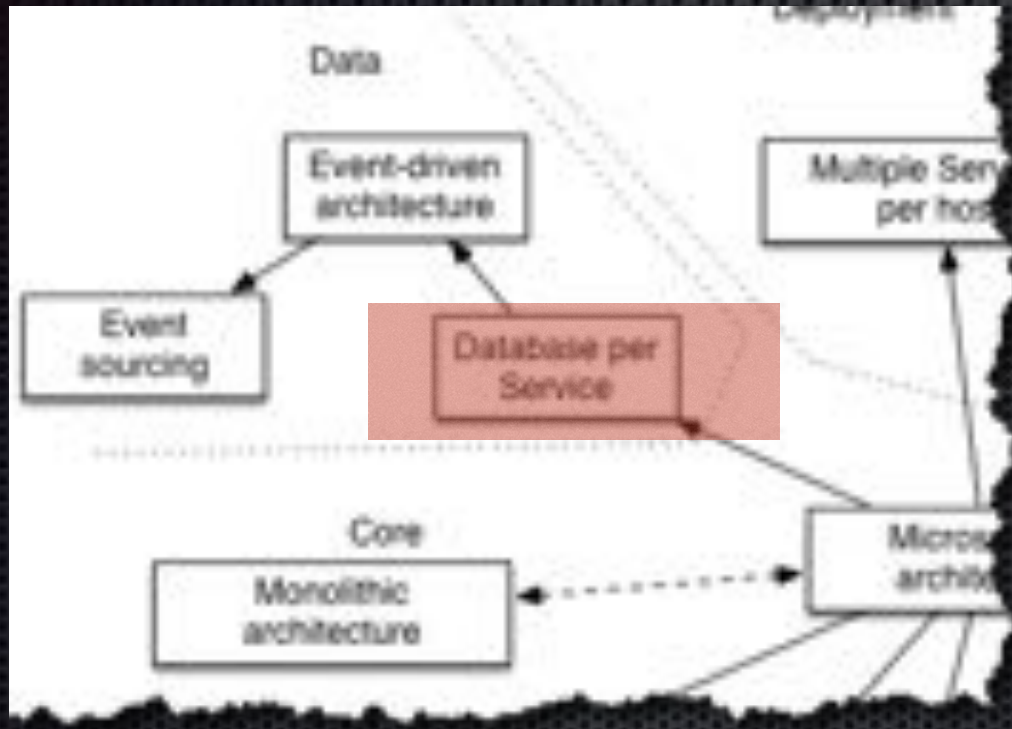


How does a client determine the network location of a service?



What's the database architecture?

Database per Service



Orders Service

Customer Service

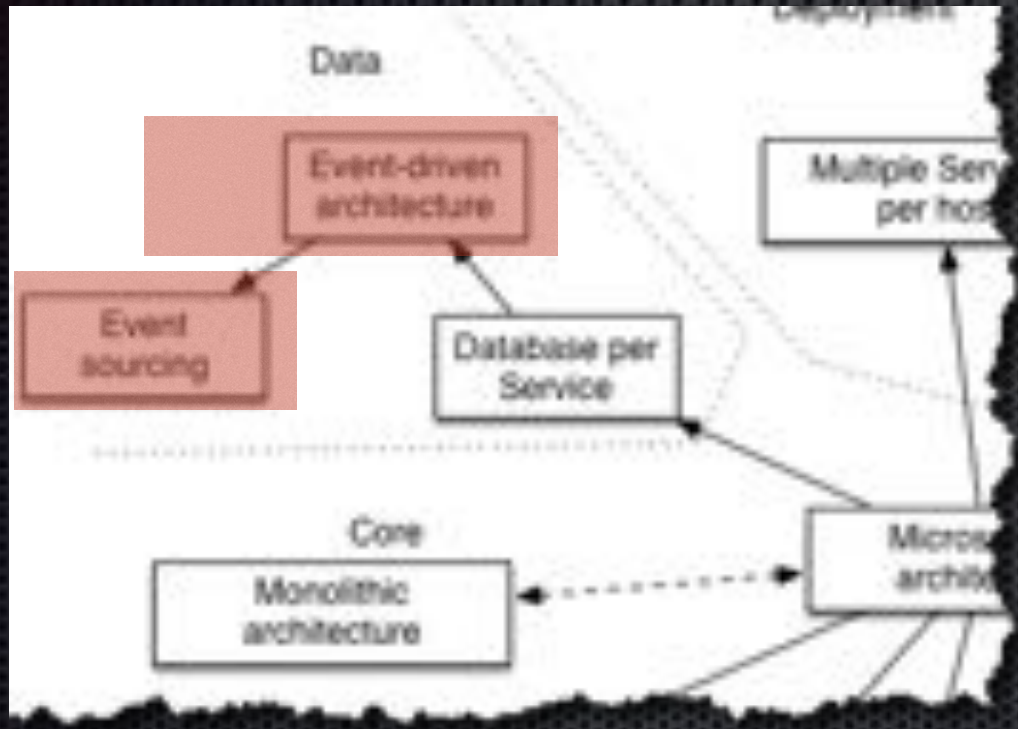
Order Database

Customer Database

NoSQL DB

Sharded SQL

How to solve distributed data management challenges?



- How to implement transactions that span multiple services?
- How to implement queries that span multiple services?

Summary: Patterns and pattern languages are a great way to …

- Think about technology
- Discuss technology
- Apply technology

Summary: The Microservices pattern language is a great way to …

- Think about microservices
- Discuss microservices
- Apply microservices (or not)


@crichardson chris@chrisrichardson.net
Questions?
http://plainoldobjects.com
http://microservices.io
http://eventuate.io