There is no such thing as a microservice!

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

Define the Microservice Architecture as an architectural style

Explain what that means and why it matters
About Chris
About Chris
About Chris
About Chris
About Chris
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

http://learnmicroservices.io

40% off
ctwcodefreeze18
Agenda

- A brief refresher on software architecture
- From monolith to microservices
- Microservices != silver bullet
- Applying the microservice pattern language
Agenda

- A brief refresher on software architecture
- From monolith to microservices
- Microservices != silver bullet
- Applying the microservice pattern language
About software architecture

“The software architecture of a computing system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.”

Documenting Software Architectures, Bass et al
Architecture =
(elements, relations, properties)
Architecture = Boxes and lines 😊
Architecture is multi-dimensional
e.g. Structural, electrical, plumbing, mechanical
Architecture is multi-dimensional
e.g. Structural, electrical, plumbing, mechanical

⇒

Described by multiple views
Architecture is multi-dimensional
e.g. Structural, electrical, plumbing, mechanical

⇒

Described by multiple views

View = (elements, relations, properties)
4+1 view model

- Logical View
- Implementation View
- Process View
- Deployment View
- Scenarios
4+1 Logical view

Elements: classes and packages
Relations: inheritance, associations, ...
4+1 Implementation view

Elements: modules and components
Relations: dependencies
4+1 Process view

Elements: processes
Relations: IPC
4+1 Deployment view

Elements: “machines”
Relations: networking
4+1 Scenarios

Derived from use cases/stories

Animate the views
"... An architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined....."

David Garlan and Mary Shaw, An Introduction to Software Architecture
Layered architectural style
Layered architectural style

Layer N

Uses

Layer N - 1

Uses

Layer N - 2

Uses

Well defined responsibilities

Constrained dependencies
The role of architecture

Requirements

= 

Functional requirements

+ 

Non-functional requirements (-ilities)
Important -ilities

- Maintainability
- Testability
- Deployability
- Evolvability
- Scalability
- Security
- Reliability
- ...

Important -ilities

- Maintainability
- Testability
- Deployability
- Evolvability
- Scalability
- Security
- Reliability
- ...

Sunday afternoon 😊

This site can’t be reached
www.umsec.umn.edu refused to connect.
Important -ilities

- Maintainability
- Testability
- Deployability
- Evolvability
- Scalability
- Security
- Reliability
- ...

Businesses must innovate faster

⇒

Build better software faster
Reducing lead time

Increasing deployment frequency
Modern software development: moving fast and **not** breaking things!

<table>
<thead>
<tr>
<th>Survey questions</th>
<th>High IT performers</th>
<th>Medium IT performers</th>
<th>Low IT performers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deployment frequency</strong></td>
<td>On demand (multiple deploys per day)</td>
<td>Between once per week and once per month</td>
<td>Between once per week and once per month*</td>
</tr>
<tr>
<td><strong>Lead time for changes</strong></td>
<td>Less than one hour</td>
<td>Between one week and one month</td>
<td>Between one week and one month*</td>
</tr>
<tr>
<td><strong>Mean time to recover (MTTR)</strong></td>
<td>Less than one hour</td>
<td>Less than one day</td>
<td>Between one day and one week</td>
</tr>
<tr>
<td><strong>Change failure rate</strong></td>
<td>0-15%</td>
<td>0-15%</td>
<td>31-45%</td>
</tr>
</tbody>
</table>

*Amazon: ~0.001%*

*Netflix: 16 minutes*
Modern software development

**Process:** DevOps/Continuous delivery/deployment

**Organization:** Small, autonomous teams

**Architecture:** ???
Agenda

- A brief refresher on software architecture
- From monolith to microservices
- Microservices != silver bullet
- Applying the microservice pattern language
Traditional: Monolithic architecture

Logical view
Traditional: Monolithic architecture

Browser/Client

HTML
REST/JSON

Logical view

Implementation view

WAR/EAR

Catalog Management
Recommendation Management
Review Management
Order Management

StoreFrontUI

MySQL Database

Tomcat/App. Server
The monolithic architecture is an architectural style that structures the application as a single executable component.
-ilities of small monoliths

- Maintainability
- Testability
- Deployability

But successful applications keep growing....

Development Team

Application
But successful applications keep growing....
But successful applications keep growing....
But successful applications keep growing....
... and growing

Development Team A

Development Team B

Development Team C

Application
Eventually:

agile development and deployment becomes impossible

= monolithic hell
Technology stack becomes increasingly obsolete

BUT

A rewrite is not feasible
-ilities of large monoliths

- Maintainability
- Testability
- Deployability

The microservice architecture is an architectural style that structures an application as a set of loosely coupled, services organized around business capabilities.
Service = business capability

Acme, Inc

Order Taking

Inventory management

Delivery management

...
Service = business capability

**Business Capability** = something a business does to deliver value
Service = independently deployable component

---

Service database

Data owned by the service
Data replicated from elsewhere

SLA

---

Synchronous:
REST, gRPC, ...

Asynchronous:
Command/Reply, Notification

---

Consumer facing

Command Query API

Event Publisher

Event Subscriber

Implementation

API Client

---

Synchronous:
REST, gRPC, ...

Asynchronous:
Command/Reply, Notification

---

Events

---

Events
Service size is secondary

microservice architecture
Service:

Meaningful business functionality

Developed by a **small** team

Minimal lead time/high deployment frequency
Microservices

= Microservice architecture

= Application architecture

⇒

No such thing as a microservice
-ilities of a microservice architecture

- Maintainability
- Testability
- Deployability
- ...

Microservices

→

DevOps/Continuous delivery/deployment of large/complex applications
Process: Continuous delivery/deployment

Organization: Small, agile, autonomous, cross functional teams

Architecture: Microservice architecture

Successful Software Development

Enables

Enables

Enables
Process: Continuous delivery/deployment

Organization: Small, agile, autonomous, cross functional teams

Architecture: Microservice architecture

Enables

Successful Software Development

Services improve testability and deployability
Process:
Continuous delivery/deployment

Organization:
Small, agile, autonomous, cross functional teams

Architecture:
Microservice architecture

Successfull Software Development

Services improve testability and deployability

Teams own services
Microservices ⇒ evolve the technology stack

- Pick a new technology when
  - Writing a new service
  - Making major changes to an existing service
- Let’s you experiment and fail safely
Agenda

- A brief refresher on software architecture
- From monolith to microservices
- Microservices != silver bullet
- 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
Drawbacks of microservices

Complexity
Drawbacks of microservices

 Complexity

Development: IPC, partial failure, distributed data
Drawbacks of microservices

Complexity

Development: IPC, partial failure, distributed data
Testing: Integration, end to end, …
Drawbacks of microservices

Complexity

Development: IPC, partial failure, distributed data
Testing: Integration, end to end, …
Deployment
…
Are microservices a good fit for my application?
Do I have the pre-requisites in place:

automated testing
automated provisioning

.....
When using microservices:
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?
It depends!
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.
What’s a pattern?

Reusable solution to a problem occurring in a particular context.
The structure of a pattern encourages objectivity
The structure of a pattern encourages objectivity

Name
The structure of a pattern encourages objectivity

Name

Context  aka the situation
The structure of a pattern encourages objectivity

Name
Context aka the situation
Problem
The structure of a pattern encourages objectivity aka the situation

Name
Context
Problem
Forces

(conflicting) issues etc to address
The structure of a pattern encourages objectivity

- Name
- Context aka the situation
- Problem
- Forces
- Solution

(conflicting) issues etc to address
The structure of a pattern encourages objectivity

- Name
- Context  aka the situation
- Problem
- Forces
- Solution
- Resulting context

(conflicting) issues etc to address
The structure of a pattern encourages objectivity

- Name
- Context
  
  aka the situation
- Problem
- Forces
- Solution
- Resulting context
- Benefits

(conflicting) issues etc to address
The structure of a pattern encourages objectivity

Name
Context
Problem
Forces
Solution
Resulting context

aka the situation

(conflicting) issues etc to address

Benefits
Drawbacks
The structure of a pattern encourages objectivity

- Name
- Context aka the situation
- Problem
- Forces
- Solution
- Resulting context

(conflicting) issues etc to address

Benefits

Drawbacks

Issues to resolve
The structure of a pattern encourages objectivity

- Name
- Context aka the situation
- Problem
- Forces
- Solution
- Resulting context
- Related patterns

(conflicting) issues etc to address

Benefits
Drawbacks
Issues to resolve

@crichardson
The structure of a pattern encourages objectivity

Name
Context
aka the situation
Problem
Forces
Solution
Resulting context
Related patterns
(conflicting) issues etc to address
Alternatives
Benefits
Drawbacks
Issues to resolve
The structure of a pattern encourages objectivity.

- Name
- Context (aka the situation)
- Problem
- Forces
- Solution
- Resulting context
- Related patterns
- (conflicting) issues etc to address
- Alternatives
- Solutions to issues
- Benefits
- Drawbacks
- Issues to resolve
Microservices pattern language: http://microservices.io
Agenda

- A brief refresher on software architecture
- From monolith to microservices
- Microservices != silver bullet

- 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
Key patterns
Issue: What’s the deployment architecture?

Forces
- Maintainability
- Deployability
- Testability
- ...

- Monolithic architecture
- Microservice architecture
Issue: What’s the deployment architecture?

Forces
- Maintainability
- Deployability
- Testability
- ...

Monolithic architecture

Single deployable/ executable OR Tightly coupled services

Microservice architecture
Issue: What’s the deployment architecture?

**Forces**
- Maintainability
- Deployability
- Testability
- ...

- Monolithic architecture
- Microservice architecture
- 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
Issue: How to decompose an application into services?

**Forces**

- Stability
- Cohesive
- Loosely coupled
- Not too large

Organize around business capabilities

Decompose by business capability

Decompose by subdomain
Issue: How to decompose an application into services?

Forces

- Stability
- Cohesive
- Loosely coupled
- Not too large

Decompose by business capability

Organize around business capabilities

Decompose by subdomain

Organize around DDD subdomains
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
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
The rest are generic technical architecture patterns  
=  
Undifferentiated heavy lifting!
Issue: How do services communicate?

Forces

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

Diagram:

- Messaging
- Remote Procedure Invocation
- Domain-specific
Issue: How to handle cross cutting concerns?

Forces

- Every service must implement logging; externalize configuration; health check endpoint; metrics; ...
Issue: How to deploy an application’s services?

**Forces**
- Multiple languages
- Isolated
- Constrained
- Monitor-able
- Reliable
- Efficient
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 monitor the behavior of your application?

- Audit logging
- Application metrics
- Distributed tracing
- Health check API
- Exception tracking
- Log aggregation
Summary

- The goal of architecture is to satisfy non-functional requirements
- For continuous delivery/deployment use the appropriate architectural style
  - Small applications → Monolithic architecture
  - Complex applications → Microservice architecture
- Use the pattern language to guide your decision making
Thank you!

40% off
ctwcodefreeze18

http://learnmicroservices.io