



Microservices

FLEXIBLE SOFTWARE ARCHITECTURE

EBERHARD WOLFF

Microservices

Microservices: Flexible Software Architecture

Table of Contents

Cover

Title Page

Copyright Page

Contents

Preface

Acknowledgments

About the Author

Part I: Motivation and Basics

Chapter 1: Preliminaries

1.1 Overview of Microservice

Microservice: Preliminary Definition

Deployment Monoliths

1.2 Why Microservices?

Strong Modularization

Easy Replaceability

Sustainable Development

Further Development of Legacy Applications

Time-to-Market

Independent Scaling

Free Choice of Technologies

Continuous Delivery

1.3 Challenges

1.4 Conclusion

Table of Contents

Chapter 2: Microservice Scenarios

2.1 Modernizing an E-Commerce Legacy Application

Scenario

Reasons to Use Microservices

Slow Continuous Delivery Pipeline

Parallel Work Is Complicated

Bottleneck During Testing

Approach

Challenges

Entire Migration Lengthy

Testing Remains a Challenge

Current Status of Migration

Creating Teams

Advantages

Conclusion

Rapid and Independent Development of New Features

Influence on the Organization

Amazon Has Been Doing It for a Long Time

2.2 Developing a New Signaling System

Scenario

Reasons to Use Microservices

Distributed System

Technology Stack per Team

Integration of Other Systems

Challenges

High Technological Complexity

Advantages

Verdict

2.3 Conclusion

Essential Points

Part II: Microservices: What, Why, and Why Not?

Chapter 3: What Are Microservices?

Table of Contents

3.1 Size of a Microservice

- Modularization
- Distributed Communication
- Sustainable Architecture
- Refactoring
- Team Size
- Infrastructure
- Replaceability
- Transactions and Consistency
- Consistency
- Compensation Transactions across Microservices
- Summary

3.2 Conways Law

- Reasons for the Law
- The Law as Limitation
- The Law as Enabler
- The Law and Microservices

3.3 Domain-Driven Design and Bounded Context

- Ubiquitous Language
- Building Blocks
- Bounded Context
- Collaboration between Bounded Contexts
- Bounded Context and Microservices
- Large-Scale Structure

3.4 Why You Should Avoid a Canonical Data Model (Stefan Tilkov)

3.5 Microservices with a UI?

- Technical Alternatives
- Self-Contained System

3.6 Conclusion

- Essential Points

Chapter 4: Reasons for Using Microservices

4.1 Technical Benefits

Table of Contents

- Replacing Microservices
- Sustainable Software Development
- Handling Legacy
- Continuous Delivery
- Scaling
- Robustness
- Free Technology Choice
- Independence

4.2 Organizational Benefits

- Smaller Projects

4.3 Benefits from a Business Perspective

- Parallel Work on Stories

4.4 Conclusion

- Essential Points

Chapter 5: Challenges

5.1 Technical Challenges

- Code Dependencies
- Unreliable Communication
- Technology Pluralism

5.2 Architecture

- Architecture = Organization
- Architecture and Requirements
- Refactoring
- Agile Architecture
- Summary

5.3 Infrastructure and Operations

- Continuous Delivery Pipelines
- Monitoring
- Version Control

5.4 Conclusion

- Essential Points

Chapter 6: Microservices and SOA

Table of Contents

6.1 What Is SOA?

- Introducing SOA
- Services in an SOA
- Interfaces and Versioning
- External Interfaces
- Interfaces Enforce a Coordination of Deployments
- Coordination and Orchestration
- Technologies

6.2 Differences between SOA and Microservices

- Communication
- Orchestration
- Flexibility
- Microservices: Project Level
- Synergies

6.3 Conclusion

- Essential Points

Part III: Implementing Microservices

Chapter 7: Architecture of Microservice-Based Systems

7.1 Domain Architecture

- Strategic Design and Domain-Driven Design
- Example Otto Shop
- Managing Dependencies
- Unintended Domain-Based Dependencies
- Cyclic Dependencies

7.2 Architecture Management

- Tools for Architecture Management
- Cycle-Free Software
- Microservices and Architecture Management
- Tools
- Is Architecture Management Important?
- Context Map

7.3 Techniques to Adjust the Architecture

Table of Contents

Where Does Bad Architecture Come From?

Changes in Microservices

Changes to the Overall Architecture

Shared Libraries

Transfer Code

Reuse or Redundancy?

Shared Service

Spawn a New Microservice

Rewriting

A Growing Number of Microservices

Microservice-Based Systems Are Hard to Modify

7.4 Growing Microservice-Based Systems

Planning Architecture?

Start Big

Start Small?

Limits of Technology

Replaceability as a Quality Criterion

The Gravity of Monoliths

Keep Splitting

Global Architecture?

7.5 Dont Miss the Exit Point or How to Avoid the Erosion of a Microservice (Lars Gentsch)

Incorporation of New Functionality

What Is Happening to the Microservice Here?

Criteria Arguing for a New Microservice Instead of Extending an Existing One

How to Recognize Whether the Creation of a New Microservice Should Have Occurred
Already

Conclusion

7.6 Microservices and Legacy Applications

Breaking Up Code?

Supplementing Legacy Applications

Enterprise Integration Patterns

Limiting Integration

Table of Contents

Advantages

Integration via UI and Data Replication

Content Management Systems

Conclusion

No Big Bang

Legacy = Infrastructure

Other Qualities

7.7 Hidden Dependencies (Oliver Wehrens)

The Database

7.8 Event-Driven Architecture

7.9 Technical Architecture

Technical Decisions for the Entire System

Sidecar

7.10 Configuration and Coordination

Consistency as Problem

Immutable Server

Alternative: Installation Tools

7.11 Service Discovery

Service Discovery = Configuration?

Technologies

7.12 Load Balancing

REST/HTTP

Central Load Balancer

A Load Balancer per Microservice

Technologies

Service Discovery

Client-Based Load Balancing

Load Balancing and Architecture

7.13 Scalability

Scaling, Microservices, and Load Balancing

Dynamic Scaling

Microservices: Advantages for Scaling

Sharding

Table of Contents

Scalability, Throughput, and Response Times

7.14 Security

Security and Microservices

OAuth2

Possible Authorization Grants

JSON Web Token (JWT)

OAuth2, JWT, and Microservices

Technologies

Additional Security Measures

Hashicorp Vault

Additional Security Goals

7.15 Documentation and Metadata

Outdated Documentation

Access to Documentation

7.16 Conclusion

Essential Points

Chapter 8: Integration and Communication

8.1 Web and UI

Multiple Single-Page-Apps

SPA per Microservice

Asset Server for Uniformity

A Single-Page App for All Microservices

HTML Applications

ROCA

Easy Routing

Arrange HTML with JavaScript

Front-End Server

Mobile Clients and Rich Clients

Organizational Level

Back-End for Each Front-End

8.2 REST

Cache and Load Balancer

HATEOAS

Table of Contents

HAL

XML

HTML

JSON

Protocol Buffer

RESTful HTTP Is Synchronous

8.3 SOAP and RPC

Flexible Transport

Thrift

8.4 Messaging

Messages and Transactions

Messaging Technology

8.5 Data Replication

Replication

Problems: Redundancy and Consistency

Implementation

Batch

Event

8.6 Interfaces: Internal and External

External Interfaces

Separating Interfaces

Implementing External Interfaces

Semantic Versioning

Postels Law or the Robustness Principle

8.7 Conclusion

Client

Logic Layer

Data Replication

Interfaces and Versions

Essential Points

Chapter 9: Architecture of Individual Microservices

9.1 Domain Architecture

Table of Contents

Cohesion

Encapsulation

Domain-Driven Design

Transactions

9.2 CQRS

CQRS

Microservices and CQRS

Advantages

Challenges

9.3 Event Sourcing

9.4 Hexagonal Architecture

Hexagons or Layers?

Hexagonal Architectures and Microservices

An Example

9.5 Resilience and Stability

Timeout

Circuit Breaker

Bulkhead

Steady State

Fail Fast

Handshaking

Test Harness

Uncoupling via Middleware

Stability and Microservices

Resilience and Reactive

Hystrix

9.6 Technical Architecture

Process Engines

Statelessness

Reactive

Microservices without Reactive?

9.7 Conclusion

Essential Points

Table of Contents

Chapter 10: Testing Microservices and Microservice-Based Systems

10.1 Why Tests?

Tests Minimize Expenditure

Tests = Documentation

Test-Driven Development

10.2 How to Test?

Unit Tests

Integration Tests

UI Tests

Manual Tests

Load Tests

Test Pyramid

Continuous Delivery Pipeline

10.3 Mitigate Risks at Deployment

10.4 Testing the Overall System

Shared Integration Tests

Avoiding Integration Tests of the Overall System

10.5 Testing Legacy Applications and Microservices

Relocating Tests of the Legacy Application

Integration Test: Legacy Application and Microservices

10.6 Testing Individual Microservices

Reference Environment

Stubs

10.7 Consumer-Driven Contract Tests

Components of the Contract

Contracts

Implementation

Tools

10.8 Testing Technical Standards

10.9 Conclusion

Essential Points

Table of Contents

Chapter 11: Operations and Continuous Delivery of Microservices

11.1 Challenges Associated with the Operation of Microservices

- Numerous Artifacts
- Delegate into Teams
- Unify Tools
- Specify Behavior
- Micro and Macro Architecture
- Templates

11.2 Logging

- Logging for Microservices
- Technologies for Logging via the Network
- ELK for Centralized Logging
- Scaling ELK
- Graylog
- Splunk
- Stakeholders for Logs
- Correlation IDs
- Zipkin: Distributed Tracing

11.3 Monitoring

- Basic Information
- Additional Metrics
- Stakeholders
- Correlate with Events
- Monitoring = Tests?
- Dynamic Environment
- Concrete Technologies
- Enabling Monitoring in Microservices
- Metrics
- StatsD
- collectd
- Technology Stack for Monitoring
- Effects on the Individual Microservice

11.4 Deployment

Table of Contents

- Deployment Automation
- Installation and Configuration
- Risks Associated with Microservice Deployments
- Deployment Strategies

11.5 Combined or Separate Deployment? (Jörg Müller)

11.6 Control

11.7 Infrastructure

- Virtualization or Cloud
- Docker
- Docker Container versus Virtualization
- Communication between Docker Containers
- Docker Registry
- Docker and Microservices
- Docker and Servers
- PaaS

11.8 Conclusion

- Essential Points

Chapter 12: Organizational Effects of a Microservices-Based Architecture

12.1 Organizational Benefits of Microservices

- Technical Independence
- Separate Deployment
- Separate Requirement Streams
- Three Levels of Independence

12.2 An Alternative Approach to Conways Law

- The Challenges Associated with Conways Law
- Collective Code Ownership
- Advantages of Collective Code Ownership
- Disadvantages of Collective Code Ownership
- Pull Requests for Coordination

12.3 Micro and Macro Architecture

- Decision = Responsibility
- Who Creates the Macro Architecture?

Table of Contents

Extent of the Macro Architecture

Technology: Macro/Micro Architecture

Operations

Domain Architecture

Tests

12.4 Technical Leadership

Developer Anarchy

12.5 DevOps

DevOps and Microservices

Do Microservices Necessitate DevOps?

12.6 When Microservices Meet Classical IT Organizations (Alexander Heusingfeld)

Pets versus Cattle

Us versus Them

Development versus Test versus Operations: Change of Perspective

For Operations There Is Never an Entirely Green Field

Conclusion

12.7 Interface to the Customer

Architecture Leads to Departments

12.8 Reusable Code

Client Libraries

Reuse Anyhow?

Reuse as Open Source

12.9 Microservices without Changing the Organization?

Microservices without Changing the Organization

Evaluation

Departments

Operations

Architecture

12.10 Conclusion

Essential Points

Part IV: Technologies

Table of Contents

Chapter 13: Example of a Microservices-Based Architecture

13.1 Domain Architecture

- Separate Data Storages
- Lots of Communication
- Bounded Context
- Dont Modularize Microservices by Data!

13.2 Basic Technologies

- HSQL Database
- Spring Data REST
- Spring Boot
- Spring Cloud
- Spring Cloud Netflix

13.3 Build

13.4 Deployment Using Docker

13.5 Vagrant

- Networking in the Example Application

13.6 Docker Machine

13.7 Docker Compose

13.8 Service Discovery

- Eureka Client
- Configuration
- Eureka Server

13.9 Communication

- Zuul: Routing

13.10 Resilience

- Circuit Breaker
- Hystrix with Annotations
- Monitoring with the Hystrix Dashboard
- Turbine

13.11 Load Balancing

- Ribbon with Spring Cloud

13.12 Integrating Other Technologies

Table of Contents

13.13 Tests

- Stubs

- Consumer-Driven Contract Test

13.14 Experiences with JVM-Based Microservices in the Amazon Cloud

- (Sascha Möllering)

- Conclusion

13.15 Conclusion

- Essential Points

Chapter 14: Technologies for Nanoservices

14.1 Why Nanoservices?

- Minimum Size of Microservices is Limited

- Compromises

- Desktop Applications

14.2 Nanoservices: Definition

14.3 Amazon Lambda

- Calling Lambda Functions

- Evaluation for Nanoservices

- Conclusion

14.4 OSGi

- The OSGi Module System

- Handling Bundles in Practice

- Evaluation for Nanoservices

- Conclusion

14.5 Java EE

- Nanoservices with Java EE

- Microservices with Java EE?

- An Example

14.6 Vert.x

- Conclusion

14.7 Erlang

- Evaluation for Nanoservices

14.8 Seneca

Table of Contents

Evaluation for Nanoservices

14.9 Conclusion

Essential Points

Chapter 15: Getting Started with Microservices

15.1 Why Microservices?

15.2 Roads towards Microservices

15.3 Microservice: Hype or Reality?

15.4 Conclusion

Index