

Table of Contents

22	Volatility	62
23	Data	63
24	Technology	64
25	Organizational	65
25	Mixing Models and Exceptions	66
26	Summary	67
26	Microservice Pain Points	68
26	Splitting the Monolith	69
27	Have a Goal	70
27	Incremental Migration	71
28	The Monolith Is Rarely the Enemy	72
28	The Dangers of Premature Decomposition	73
29	What to Split First?	74
29	Decomposition by Layer	75
30	Testing	76
31	Data Consistency	77
31	Whom They Might Not Work For	78
32	Where They Work Well	79
33	Summary	80
33	Microservices at a Glance	81
34	Key Concepts of Microservices	82
34	Independent Deployability	83
35	Modeled Around a Business Domain	84
35	Owning Their Own State	85
36	Size	86
36	Flexibility	87
37	Alignment of Architecture and Organization	88
37	The Monolith	89
38	The Single-Process Monolith	90
38	The Modular Monolith	91
39	The Distributed Monolith	92
39	Monoliths and Delivery Contention	93
40	Advantages of Monoliths	94
40	Enabling Technology	95
41	Log Aggregation and Distributed Tracing	96
41	Containers and Kubernetes	97
42	Streaming	98
42	Public Cloud and Serverless	99
43	Advantages of Microservices	100

Technology Heterogeneity	22
Robustness	23
Scaling	24
Ease of Deployment	25
Organizational Alignment	25
Composability	26
Microservice Pain Points	26
Developer Experience	26
Technology Overload	27
Cost	28
Reporting	28
Monitoring and Troubleshooting	29
Security	29
Testing	30
Latency	30
Data Consistency	31
Should I Use Microservices?	31
Whom They Might Not Work For	31
Where They Work Well	33
Summary	34
2. How to Model Microservices	35
Introducing MusicCorp	35
What Makes a Good Microservice Boundary?	36
Information Hiding	36
Cohesion	38
Coupling	38
The Interplay of Coupling and Cohesion	39
Types of Coupling	39
Domain Coupling	41
Pass-Through Coupling	43
Common Coupling	46
Content Coupling	49
Just Enough Domain-Driven Design	51
Ubiquitous Language	52
Aggregate	53
Bounded Context	56
Mapping Aggregates and Bounded Contexts to Microservices	58
Event Storming	59
The Case for Domain-Driven Design for Microservices	61
Alternatives to Business Domain Boundaries	62

99	Volatility	62
100	Data	64
101	Technology	65
101	Organizational	66
102	Mixing Models and Exceptions	68
103	Summary	69
103	3. Splitting the Monolith.....	71
104	Have a Goal	71
106	Incremental Migration	72
108	The Monolith Is Rarely the Enemy	73
108	The Dangers of Premature Decomposition	73
110	What to Split First?	74
111	Decomposition by Layer	76
112	Code First	77
116	Data First	78
117	Useful Decompositional Patterns	79
	Strangler Fig Pattern	79
	Parallel Run	80
	Feature Toggle	80
	Data Decomposition Concerns	81
121	Performance	81
121	Data Integrity	84
121	Transactions	84
122	Tooling	85
123	Reporting Database	85
123	Summary	86
123	4. Microservice Communication Styles.....	89
123	From In-Process to Inter-Process	89
127	Performance	90
133	Changing Interfaces	91
135	Error Handling	91
140	Technology for Inter-Process Communication: So Many Choices	93
140	Styles of Microservice Communication	93
141	Mix and Match	95
141	Pattern: Synchronous Blocking	95
142	Advantages	96
143	Disadvantages	96
144	Where to Use It	96
	Pattern: Asynchronous Nonblocking	98

Advantages	99
Disadvantages	100
Where to Use It	101
Pattern: Communication Through Common Data	101
Implementation	102
Advantages	103
Disadvantages	103
Where to Use It	104
Pattern: Request-Response Communication	104
Implementation: Synchronous Versus Asynchronous	106
Where to Use It	108
Pattern: Event-Driven Communication	108
Implementation	110
What's in an Event?	111
Where to Use It	115
Proceed with Caution	116
Summary	117

Part II. Implementation

5. Implementing Microservice Communication.....	121
Looking for the Ideal Technology	121
Make Backward Compatibility Easy	121
Make Your Interface Explicit	122
Keep Your APIs Technology Agnostic	122
Make Your Service Simple for Consumers	122
Hide Internal Implementation Detail	123
Technology Choices	123
Remote Procedure Calls	123
REST	127
GraphQL	133
Message Brokers	135
Serialization Formats	140
Textual Formats	140
Binary Formats	141
Schemas	141
Structural Versus Semantic Contract Breakages	142
Should You Use Schemas?	143
Handling Change Between Microservices	144

Avoiding Breaking Changes	144
Expansion Changes	145
Tolerant Reader	145
Right Technology	146
Explicit Interface	146
Catch Accidental Breaking Changes Early	148
Managing Breaking Changes	149
Lockstep Deployment	149
Coexist Incompatible Microservice Versions	149
Emulate the Old Interface	150
Which Approach Do I Prefer?	152
The Social Contract	152
Tracking Usage	153
Extreme Measures	154
DRY and the Perils of Code Reuse in a Microservice World	154
Sharing Code via Libraries	155
Service Discovery	157
Domain Name System (DNS)	157
Dynamic Service Registries	159
Don't Forget the Humans!	161
Service Meshes and API Gateways	162
API Gateways	163
Service Meshes	166
What About Other Protocols?	169
Documenting Services	169
Explicit Schemas	169
The Self-Describing System	170
Summary	173
6. Workflow	175
Database Transactions	175
ACID Transactions	176
Still ACID, but Lacking Atomicity?	177
Distributed Transactions—Two-Phase Commits	179
Distributed Transactions—Just Say No	181
Sagas	182
Saga Failure Modes	184
Implementing Sagas	189
Sagas Versus Distributed Transactions	195
Summary	196

7. Build.....	197
A Brief Introduction to Continuous Integration	197
Are You Really Doing CI?	198
Branching Models	199
Build Pipelines and Continuous Delivery	201
Tooling	203
Trade-Offs and Environments	203
Artifact Creation	204
Mapping Source Code and Builds to Microservices	205
One Giant Repo, One Giant Build	205
Pattern: One Repository per Microservice (aka Multirepo)	207
Pattern: Monorepo	210
Which Approach Would I Use?	217
Summary	217
8. Deployment.....	219
From Logical to Physical	219
Multiple Instances	220
The Database	222
Environments	225
Principles of Microservice Deployment	228
Isolated Execution	228
Focus on Automation	231
Infrastructure as Code (IAC)	232
Zero-Downtime Deployment	233
Desired State Management	234
Deployment Options	237
Physical Machines	238
Virtual Machines	239
Containers	241
Application Containers	247
Platform as a Service (PaaS)	248
Function as a Service (FaaS)	249
Which Deployment Option Is Right for You?	257
Kubernetes and Container Orchestration	259
The Case for Container Orchestration	259
A Simplified View of Kubernetes Concepts	260
Multitenancy and Federation	262
The Cloud Native Computing Federation	265
Platforms and Portability	265

Helm, Operators, and CRDs, Oh My!	266
And Knative	267
The Future	268
Should You Use It?	268
Progressive Delivery	269
Separating Deployment from Release	270
On to Progressive Delivery	270
Feature Toggles	271
Canary Release	271
Parallel Run	272
Summary	273
9. Testing.....	275
Types of Tests	276
Test Scope	278
Unit Tests	279
Service Tests	280
End-to-End Tests	281
Trade-Offs	282
Implementing Service Tests	283
Mocking or Stubbing	283
A Smarter Stub Service	284
Implementing (Those Tricky) End-to-End Tests	285
Flaky and Brittle Tests	286
Who Writes These End-to-End Tests?	287
How Long Should End-to-End Tests Run?	289
The Great Pile-Up.....	290
The Metaversion	291
Lack of Independent Testability	291
Should You Avoid End-to-End Tests?	292
Contract Tests and Consumer-Driven Contracts (CDCs)	292
The Final Word	295
Developer Experience	296
From Preproduction to In-Production Testing	297
Types of In-Production Testing	298
Making Testing in Production Safe	298
Mean Time to Repair over Mean Time Between Failures?	299
Cross-Functional Testing	300
Performance Tests	301
Robustness Tests	302
Summary	303

10. From Monitoring to Observability.....	305
Disruption, Panic, and Confusion	305
Single Microservice, Single Server	306
Single Microservice, Multiple Servers	307
Multiple Services, Multiple Servers	308
Observability Versus Monitoring	309
The Pillars of Observability? Not So Fast	310
Building Blocks for Observability	311
Log Aggregation	312
Metrics Aggregation	321
Distributed Tracing	324
Are We Doing OK?	327
Alerting	329
Semantic Monitoring	333
Testing in Production	335
Standardization	337
Selecting Tools	338
Democratic	338
Easy to Integrate	339
Provide Context	339
Real-Time	339
Suitable for Your Scale	340
The Expert in the Machine	340
Getting Started	341
Summary	342
11. Security.....	345
Core Principles	346
Principle of Least Privilege	347
Defense in Depth	347
Automation	349
Build Security into the Delivery Process	349
The Five Functions of Cybersecurity	350
Identify	351
Protect	352
Detect	353
Respond	353
Recover	354
Foundations of Application Security	354
Credentials	354

Patching	360
Backups	363
Rebuild	364
Implicit Trust Versus Zero Trust	365
Implicit Trust	366
Zero Trust	366
It's a Spectrum	367
Securing Data	369
Data in Transit	369
Data at Rest	372
Authentication and Authorization	375
Service-to-Service Authentication	375
Human Authentication	376
Common Single Sign-On Implementations	376
Single Sign-On Gateway	377
Fine-Grained Authorization	379
The Confused Deputy Problem	380
Centralized, Upstream Authorization	381
Decentralizing Authorization	382
JSON Web Tokens	382
Summary	386
12. Resiliency	387
What Is Resiliency?	387
Robustness	388
Rebound	389
Graceful Extensibility	390
Sustained Adaptability	390
And Microservice Architecture	391
Failure Is Everywhere	391
How Much Is Too Much?	392
Degrading Functionality	394
Stability Patterns	395
Time-Outs	397
Retries	399
Bulkheads	400
Circuit Breakers	401
Isolation	404
Redundancy	405
Middleware	405
Idempotency	406

Spreading Your Risk	407
CAP Theorem	408
Sacrificing Consistency	410
Sacrificing Availability	410
Sacrificing Partition Tolerance?	411
AP or CP?	411
It's Not All or Nothing	412
And the Real World	412
Chaos Engineering	413
Game Days	414
Production Experiments	415
From Robustness to Beyond	415
Blame	415
Summary	417
13. Scaling	419
The Four Axes of Scaling	419
Vertical Scaling	420
Horizontal Duplication	422
Data Partitioning	426
Functional Decomposition	430
Combining Models	432
Start Small	433
Caching	435
For Performance	436
For Scale	436
For Robustness	436
Where to Cache	437
Invalidation	442
The Golden Rule of Caching	447
Freshness Versus Optimization	448
Cache Poisoning: A Cautionary Tale	448
Autoscaling	449
Starting Again	450
Summary	451
Detect	453
Respond	453
Recover	453
Foundations of Application Security	454
Credentials	454
Idempotency	454

Part III. People

14. User Interfaces.....	455
Toward Digital	456
Ownership Models	456
Drivers for Dedicated Frontend Teams	458
Toward Stream-Aligned Teams	459
Sharing Specialists	460
Ensuring Consistency	461
Working Through Technical Challenges	462
Pattern: Monolithic Frontend	463
When to Use It	464
Pattern: Micro Frontends	464
Implementation	465
When to Use It	465
Pattern: Page-Based Decomposition	467
Where to Use It	468
Pattern: Widget-Based Decomposition	469
Implementation	470
When to Use It	473
Constraints	474
Pattern: Central Aggregating Gateway	475
Ownership	476
Different Types of User Interfaces	477
Multiple Concerns	478
When to Use It	479
Pattern: Backend for Frontend (BFF)	480
How Many BFFs?	481
Reuse and BFFs	483
BFFs for Desktop Web and Beyond	486
When to Use	487
GraphQL	488
A Hybrid Approach	489
Summary	490
15. Organizational Structures.....	491
Loosely Coupled Organizations	491
Conway's Law	493
Evidence	493
Team Size	495

Understanding Conway's Law	496
Small Teams, Large Organization	496
On Autonomy	498
Strong Versus Collective Ownership	499
Strong Ownership	500
Collective Ownership	501
At a Team Level Versus an Organizational Level	502
Balancing Models	502
Enabling Teams	503
Communities of Practice	505
The Platform	506
Shared Microservices	509
Too Hard to Split	509
Cross-Cutting Changes	509
Delivery Bottlenecks	510
Internal Open Source	511
Role of the Core Committers	511
Maturity	512
Tooling	512
Pluggable, Modular Microservices	513
Change Reviews	515
The Orphaned Service	518
Case Study: realestate.com.au	519
Geographical Distribution	521
Conway's Law in Reverse	522
People	523
Summary	524
16. The Evolutionary Architect	525
What's in a Name?	525
What Is Software Architecture?	527
Making Change Possible	529
An Evolutionary Vision for the Architect	529
Defining System Boundaries	530
A Social Construct	533
Habitability	534
A Principled Approach	536
Strategic Goals	536
Principles	536
Practices	537
Combining Principles and Practices	537

A Real-World Example	538
Guiding an Evolutionary Architecture	539
Architecture in a Stream-Aligned Organization	540
Building a Team	542
The Required Standard	543
Monitoring	543
Interfaces	543
Architectural Safety	544
Governance and the Paved Road	544
Exemplars	545
Tailored Microservice Template	545
The Paved Road at Scale	547
Technical Debt	547
Exception Handling	548
Summary	548
Afterword: Bringing It All Together.....	551
Bibliography.....	563
Glossary.....	569
Index.....	575

The experiences of people all over the world, along with the emergence of new technologies, are having a profound effect on how microservices are used. This book brings these ideas together, along with concrete, real-world examples, to help you understand whether microservices are right for you.

Who Should Read This Book

The scope of *Building Microservices* is broad, as the implications of microservice architectures are also broad. As such, this book should appeal to people interested in aspects of design, development, deployment, testing, and maintenance of systems. Those of you who have already embarked on the journey toward finer-grained architectures, whether for a greenfield application or as part of decomposing an existing, more monolithic system, will find plenty of practical advice to help you. This book will also help those of you who want to know what all the fuss is about, so that you can determine whether microservices are right for you.