

Inhalt

Vorwort	XI
1 Einleitung	1
1.1 An wen sich dieses Buch richtet	2
1.2 Was dieses Buch behandelt	3
1.3 Sprachliche Konventionen	4
1.4 Notationskonventionen	5
1.5 Ergänzende Materialien	7
Teil I: Grundlagen	9
2 Cloud Computing	11
2.1 Service-Modelle	12
2.1.1 Infrastructure as a Service (IaaS)	15
2.1.2 Platform as a Service (PaaS)	15
2.1.3 Software as a Service (SaaS)	16
2.2 Cloud-Ökonomie	16
2.2.1 Eignung von unterschiedlichen Arten von Workloads	17
2.2.2 Effekt von Zuteilungsdauer und Ressourcengröße	19
2.3 Entwicklung der letzten Jahre	21
3 DevOps	23
3.1 Prinzipien des Flow	25
3.1.1 Prinzip 1: Arbeit sichtbar machen	25
3.1.2 Prinzip 2: Work in Progress beschränken	26
3.1.3 Prinzip 3: Flaschenhälse minimieren	26
3.2 Prinzipien des Feedbacks	27
3.2.1 Prinzip 4: Probleme früh erkennen	27
3.2.2 Prinzip 5: Probleme sofort lösen	28
3.2.3 Prinzip 6: Probleme professionell verantworten	28
3.3 DevOps-geeignete Architekturen	29
3.3.1 Randbedingungen für die Entwicklung	29

3.3.2	Nutzung von Orchestrierungsplattformen.....	30
3.3.3	Randbedingungen im Betrieb	30
4	Cloud-native	33
4.1	Definitionen in Industrie und Forschung.....	34
4.2	Die Cloud-native-Definition dieses Buchs.....	35
4.3	Zusammenfassung und Ausblick auf Teil II und Teil III.....	37
Teil II: Everything as Code.....		39
5	Einleitung zu Teil II	41
6	Deployment-Pipelines	43
6.1	Deployment-Pipelines as Code.....	44
6.1.1	Phasen-Pipelines.....	45
6.1.2	Gerichtete Pipelines	46
6.1.3	Hierarchische Pipelines.....	47
6.1.4	Steuerung von Pipelines	48
6.2	DevOps-geeignete Branching-Strategien.....	50
6.2.1	Git-Flow	51
6.2.2	GitHub-Flow	52
6.2.3	Trunk-basierte Entwicklung	53
6.3	Zusammenfassung	54
7	Infrastructure as Code.....	57
7.1	Virtualisierung	59
7.1.1	Virtualisierung von Hardware-Infrastruktur.....	59
7.1.2	Virtualisierung von Software-Infrastruktur	60
7.2	Provisionierung.....	62
7.2.1	Immutable Infrastructure	62
7.2.2	IaC-Ansätze	63
7.2.3	Provisionierung von lokalen Umgebungen	66
7.2.4	Provisionierung von Multi-Host-Umgebungen	68
7.3	Zusammenfassung	71
8	Standardisierung von Deployment Units (Container)	73
8.1	Hintergrund (PaaS).....	73
8.2	Betriebssystem-Virtualisierung.....	76
8.3	Container Runtime Environments.....	77
8.3.1	Kernel-Namespaces	78
8.3.2	Process Capabilities	79
8.3.3	Control Groups	80

8.3.4	Union Filesystem	80
8.3.5	High-Level- und Low-Level-Container-Laufzeitumgebungen	81
8.4	Bau und Bereitstellung von Container-Images	82
8.5	Faktoren gut betreibbarer Container	84
8.5.1	Codebase	85
8.5.2	Abhängigkeiten und Konfigurationen	85
8.5.3	Unterstützende Services und Port Binding	86
8.5.4	Build-, Release- und Run-Phase	87
8.5.5	Horizontale Skalierung über Prozesse	88
8.5.6	Umgebungen, Logs und Betrieb	89
8.6	Zusammenfassung	90
9	Container-Plattformen	93
9.1	Scheduling	94
9.1.1	Heterogenität von Workloads	95
9.1.2	Scheduling-Algorithmen	96
9.1.2.1	Einfache Scheduling-Algorithmen	96
9.1.2.2	Multidimensionale Scheduling-Algorithmen	97
9.1.2.3	Kapazitätsbasierte Scheduling-Algorithmen	97
9.1.3	Scheduling-Architekturen	98
9.1.3.1	Monolithischer Scheduler	99
9.1.3.2	2-Level-Scheduler	99
9.1.3.3	Shared-State Scheduler	100
9.2	Orchestrierung	101
9.2.1	Definition von Betriebszuständen	101
9.2.2	Regelkreis: Desired versus Current State	102
9.3	Inside Kubernetes	103
9.3.1	Kubernetes-Architektur	104
9.3.2	Verwaltete Ressourcen und Basis-Blueprint	106
9.3.3	Schedulbare Workloads	108
9.3.3.1	Deployments	108
9.3.3.2	(Cron-)Jobs	110
9.3.3.3	Daemon-Sets	111
9.3.3.4	Stateful-Sets	112
9.3.4	Scheduling Constraints	115
9.3.4.1	Angabe des Ressourcenbedarfs mittels Requests und Limits	115
9.3.4.2	Knoten-Selektoren	116
9.3.4.3	Knotenaffinitäten	117
9.3.4.4	Pod-(Anti-)Affinitäten	118
9.3.5	Automatische Skalierung von Workloads	119
9.3.6	Exponieren von Workloads als interne und externe Services	120
9.3.7	Health Checking	123
9.3.8	Persistenz	126

9.3.9	Isolation von Workloads.....	127
9.3.9.1	Namespaces und Role-based Access Model (Multi-Tenancy).....	127
9.3.9.2	Quotas und Limit Ranges.....	128
9.3.9.3	Network Policies.....	129
9.4	Zusammenfassung.....	131
10	Function as a Service.....	135
10.1	FaaS-Plattformen.....	137
10.1.1	Das FaaS-Programmiermodell.....	139
10.1.2	Zu berücksichtigende Randbedingungen.....	140
10.1.3	Veranschaulichung des FaaS-Programmiermodells.....	141
10.2	Plattformagnostische FaaS-Frameworks.....	142
10.3	Ereignisbasierte Autoskalierung.....	145
10.4	Zusammenfassung.....	148

Teil III: Cloud-native Architekturen 151

11	Einleitung zu Teil III.....	153
12	Microservice und Serverless-Architekturen.....	155
12.1	Eigenschaften von Microservices.....	156
12.2	Integrationsmuster für Microservices.....	160
12.2.1	Datenbankbasierte Integration.....	161
12.2.2	(g)RPC-basierte Interprozesskommunikation.....	161
12.2.3	Representational State Transfer (REST).....	164
12.2.4	Ereignisbasierte Integration (asynchron).....	167
12.2.5	API-Versioning.....	169
12.3	Architekturelle Sicherheit.....	172
12.3.1	Circuit-Breaker.....	172
12.3.2	Bulkhead.....	173
12.3.3	Idempotente API-Operationen.....	174
12.4	Skalierung von Microservices.....	174
12.4.1	Load Balancing.....	175
12.4.2	Messaging.....	175
12.4.3	Skalierung zustandsbehafteter Komponenten.....	177
12.4.3.1	Scaling for Reads.....	178
12.4.3.2	Scaling for Writes (Sharding).....	178
12.4.3.3	Command Query Responsibility Segregation (CQRS).....	179
12.4.4	Caching.....	180
12.5	Prinzipien zur Entwicklung von Microservices.....	181
12.5.1	Prinzip 1: Bilde Modelle um Geschäftskonzepte.....	181
12.5.2	Prinzip 2: Erschaffe eine Kultur der Automatisierung.....	181

12.5.3	Prinzip 3: Blende interne Implementierungsdetails aus	182
12.5.4	Prinzip 4: Dezentralisiere	182
12.5.5	Prinzip 5: Definiere unabhängig aktualisierbare Einheiten	182
12.5.6	Prinzip 6: Isoliere Fehler	183
12.5.7	Prinzip 7: Baue gut beobachtbare Services	183
12.6	Serverless-Architekturen	184
12.6.1	Architekturelle Konsequenzen von Serverless-Limitierungen	185
12.6.2	Das API-Gateway-Pattern	187
12.6.3	Abgrenzung zu Microservices	189
12.7	Zusammenfassung	190
13	Beobachtbare Architekturen	193
13.1	Konsolidierung von Telemetriedaten	194
13.2	Instrumentierung von Systemen	196
13.2.1	Logging	196
13.2.2	Monitoring	198
13.2.2.1	Metrikarten	200
13.2.2.2	Empfehlungen für die Metrikinstrumentierung	201
13.2.3	Tracing	201
13.2.3.1	Empfehlungen für die Instrumentierung	203
13.2.3.2	Tracing-Instrumentierung und Erzeugung von Spans	206
13.2.3.3	Serverseitiges Tracing und Extraktion von Span-Kontexten	207
13.2.3.4	Clientseitiges Tracing und Weiterreichen von Span-Kontexten	208
13.3	Automatisierte Instrumentierung	209
13.3.1	Eigenschaften von Service-Meshs	210
13.3.2	Traffic-Management	212
13.3.3	Resilienz	215
13.3.4	Sicherheit	217
13.3.5	Management und Analyse von Verkehrstopologien	220
13.4	Zusammenfassung	221
14	Domain-driven Design	223
14.1	Fachlichkeit	224
14.2	Strategisches Design	226
14.2.1	Subdomänen	227
14.2.1.1	Kerndomäne (Core Subdomain)	227
14.2.1.2	Unterstützende Subdomäne (Supporting Subdomain)	228
14.2.1.3	Generische Subdomänen (Generic Subdomain)	228
14.2.1.4	Anmerkungen am Beispiel einer Fallstudie	228
14.2.2	Ubiquitous Language	230
14.2.2.1	Eine gemeinsame Sprache als Schlüssel zu einem gemeinsamen Verständnis	231
14.2.2.2	Mehrdeutige und synonyme Begriffe	232

14.2.3	Bounded Contexts.	233
14.2.4	Context Mapping	235
14.2.4.1	Partnerschaftliche Kooperationsmuster (Partners und Shared-Kernel)	235
14.2.4.2	Customer-Supplier-Kooperation.	237
14.2.4.3	Separate Ways	238
14.2.4.4	Context Maps als Landkarte von Machtverhältnissen	239
14.3	Taktisches Design	240
14.3.1	Oft genutzte Pattern für Geschäftslogik	240
14.3.1.1	Das ETL-Pattern (primär Supporting Subdomains).	240
14.3.1.2	Das Active Record-Pattern (primär Supporting Subdomains) ...	241
14.3.1.3	Das Domain Model-Pattern (primär Core Subdomains)	242
14.3.1.4	Das Event-Sourcing-Pattern (primär Core Subdomains).	244
14.3.2	Oft genutzte Pattern für die Architektur	245
14.3.2.1	Die Ebenen-Architektur	246
14.3.2.2	Das Ports & Adapter-Pattern	247
14.3.2.3	Das CQRS-Pattern	247
14.4	Zusammenfassung	250
15	Schlussbemerkungen	253
	Literaturverzeichnis.	261
	Stichwortverzeichnis.	265