

Vorwort	19
Warum ich dieses Buch geschrieben habe	20
Was in dieser Auflage neu und anders ist	20
Was kommt als Nächstes?	22
Was dieses Buch ist und was es nicht ist	22
Etwas Geschichte zu Infrastructure as Code	23
Für wen dieses Buch gedacht ist	24
Prinzipien, Praktiken und Patterns	24
Die ShopSpinner-Beispiele	25
In diesem Buch verwendete Konventionen	25
Danksagung	26

Teil I Grundlagen

1 Was ist Infrastructure as Code?	31
Aus der Eisenzeit in das Cloud-Zeitalter	33
Infrastructure as Code	34
Vorteile von Infrastructure as Code	34
Infrastructure as Code nutzen, um für Änderungen zu optimieren	35
Einwand »Wir haben gar nicht so häufig Änderungen, sodass sich Automation nicht lohnt«	35
Einwand »Wir sollten erst bauen und danach automatisieren«	36
Einwand »Wir müssen zwischen Geschwindigkeit und Qualität entscheiden«	37
Die Four Key Metrics	39
Drei zentrale Praktiken für Infrastructure as Code	40
Zentrale Praktik: Definieren Sie alles als Code	40
Zentrale Praktik: Kontinuierliches Testen und die gesamte aktuelle Arbeit ausliefern	41

Zentrale Praktik: Kleine einfache Elemente bauen, die Sie unabhängig voneinander ändern können	41
Zusammenfassung	42
2 Prinzipien der Infrastruktur im Cloud-Zeitalter	43
Prinzip: Gehen Sie davon aus, dass Systeme unzuverlässig sind.	44
Prinzip: Alles reproduzierbar machen.	44
Fallstrick: Snowflake-Systeme.	45
Prinzip: Erstellen Sie wegwerfbare Elemente	46
Prinzip: Variationen minimieren.	47
Konfigurationsdrift.	48
Prinzip: Stellen Sie sicher, dass Sie jeden Prozess wiederholen können .	50
Zusammenfassung	51
3 Infrastruktur-Plattformen	53
Die Elemente eines Infrastruktur-Systems	53
Dynamische Infrastruktur-Plattform	55
Infrastruktur-Ressourcen	57
Computing-Ressourcen	58
Storage-Ressourcen	59
Networking-Ressourcen.	61
Zusammenfassung	63
4 Zentrale Praktik: Definieren Sie alles als Code	65
Warum Sie Ihre Infrastruktur als Code definieren sollten	65
Was Sie als Code definieren können.	66
Wählen Sie Werkzeuge mit externalisierter Konfiguration aus	67
Managen Sie Ihren Code in einer Versionsverwaltung	67
Programmiersprachen für Infrastruktur	68
Infrastruktur-Scripting	70
Deklarative Infrastruktur-Sprachen	71
Programmierbare, imperative Infrastruktur-Sprachen.	73
Deklarative und imperative Sprachen für Infrastruktur.	74
Domänenspezifische Infrastruktur-Sprachen.	75
Allgemein nutzbare Sprachen und DSLs für die Infrastruktur.	76
Implementierungs-Prinzipien beim Definieren von	
Infrastructure as Code	77
Halten Sie deklarativen und imperativen Code voneinander getrennt	77
Behandeln Sie Infrastruktur-Code wie echten Code	77
Zusammenfassung	78

Teil II Arbeiten mit Infrastruktur-Stacks

5	Infrastruktur-Stacks als Code bauen	81
	Was ist ein Infrastruktur-Stack?	81
	Stack-Code	83
	Stack-Instanzen	83
	Server in einem Stack konfigurieren	83
	Low-Level-Infrastruktur-Sprachen	84
	High-Level-Infrastruktur-Sprachen	85
	Patterns und Antipatterns für das Strukturieren von Stacks	86
	Antipattern: Monolithic Stack	86
	Pattern: Application Group Stack	89
	Pattern: Service Stack	91
	Pattern: Micro Stack	92
	Zusammenfassung	93
6	Umgebungen mit Stacks bauen	95
	Worum es bei Umgebungen geht	95
	Auslieferungsumgebungen	96
	Mehrere Produktivumgebungen	96
	Umgebungen, Konsistenz und Konfiguration	97
	Patterns zum Bauen von Umgebungen	98
	Antipattern: Multiple-Environment Stack	99
	Antipattern: Copy-Paste Environments	100
	Pattern: Reusable Stack	102
	Umgebungen mit mehreren Stacks erstellen	104
	Zusammenfassung	106
7	Stack-Instanzen konfigurieren	107
	Eindeutige Kennungen durch Stack-Parameter erstellen	108
	Beispiel-Stack-Parameter	109
	Patterns zum Konfigurieren von Stacks	110
	Antipattern: Manual Stack Parameters	110
	Pattern: Stack Environment Variables	112
	Pattern: Scripted Parameters	114
	Pattern: Stack Configuration Files	117
	Pattern: Wrapper-Stack	120
	Pattern: Pipeline Stack Parameters	123
	Pattern: Stack Parameter Registry	126
	Konfigurations-Registry	129
	Eine Konfigurations-Registry implementieren	130
	Eine oder mehrere Konfigurations-Registries	132

Secrets als Parameter nutzen	133
Secrets verschlüsseln	133
Secretlose Autorisierung	133
Secrets zur Laufzeit injizieren.	134
Wegwerf-Secrets.	135
Zusammenfassung	135
8 Zentrale Praktik: Kontinuierlich testen und ausliefern	137
Warum Infrastruktur-Code kontinuierlich testen?	138
Was kontinuierliches Testen bedeutet.	138
Was sollten wir bei der Infrastruktur testen?	140
Herausforderungen beim Testen von Infrastruktur-Code	143
Herausforderung: Tests für deklarativen Code haben häufig nur einen geringen Wert	143
Herausforderung: Das Testen von Infrastruktur-Code ist langsam.	146
Herausforderung: Abhängigkeiten verkomplizieren die Test-Infrastruktur.	148
Progressives Testen	148
Testpyramide	149
Schweizer-Käse-Testmodell	151
Infrastruktur-Delivery-Pipelines	152
Pipeline-Stages	153
Scope von Komponenten, die in einer Stage getestet werden	154
Scope von Abhängigkeiten für eine Stage	155
Plattformelemente, die für eine Stage erforderlich sind	155
Software und Services für die Delivery-Pipeline.	156
Testen in der Produktivumgebung	158
Was Sie außerhalb der Produktivumgebung nicht nachbauen können	159
Die Risiken beim Testen in der Produktivumgebung managen.	160
Zusammenfassung	161
9 Infrastruktur-Stacks testen	163
Beispiel-Infrastruktur	163
Der Beispiel-Stack	164
Pipeline für den Beispiel-Stack.	165
Offline-Test-Stages für Stacks	165
Syntax-Checks	166
Statische Offline-Code-Analyse	166
Statische Code-Analyse per API	167
Testen mit einer Mock-API	167

Online-Test-Stages für Stacks	168
Preview: Prüfen, welche Änderungen vorgenommen werden	169
Verifikation: Aussagen über Infrastruktur-Ressourcen treffen	169
Ergebnisse: Prüfen, dass die Infrastruktur korrekt arbeitet	171
Test-Fixtures für den Umgang mit Abhängigkeiten verwenden	172
Test-Doubles für Upstream-Abhängigkeiten	173
Test-Fixtures für Downstream-Abhängigkeiten	174
Komponenten refaktorisieren, um sie isolieren zu können	175
Lebenszyklus-Patterns für Testinstanzen von Stacks.	176
Pattern: Persistent Test Stack	176
Pattern: Ephemeral Test Stack	178
Antipattern: Dual Persistent and Ephemeral Stack Stages	179
Pattern: Periodic Stack Rebuild	180
Pattern: Continuous Stack Reset	182
Test-Orchestrierung.	183
Unterstützen Sie lokales Testen	184
Vermeiden Sie eine enge Kopplung mit Pipeline-Tools	185
Tools zur Test-Orchestrierung	185
Zusammenfassung.	185

Teil III Mit Servern und anderen Anwendungs-Laufzeitplattformen arbeiten

10 Anwendungs-Laufzeitumgebungen.	189
Cloud-native und anwendungsgesteuerte Infrastruktur	190
Ziele für eine Anwendungs-Laufzeitumgebung.	191
Deploybare Teile einer Anwendung	191
Deployment-Pakete.	193
Anwendungen auf Server deployen	193
Anwendungen als Container verpacken	193
Anwendungen auf Server-Cluster deployen	195
Anwendungen auf Anwendungs-Cluster deployen.	195
Pakete zum Deployen von Anwendungen auf Cluster	197
FaaS-Serverless-Anwendungen deployen	198
Anwendungsdaten.	199
Datenschemata und -strukturen	199
Cloud-native Storage-Infrastruktur für Anwendungen.	200
Anwendungs-Connectivity	200
Service Discovery	201
Zusammenfassung.	203

11 Server als Code bauen	205
Was gibt es auf einem Server	206
Woher Dinge kommen	207
Server-Konfigurationscode	209
Code-Module für die Serverkonfiguration	210
Code-Module für die Serverkonfiguration designen	210
Server-Code versionieren und weitergeben	211
Serverrollen	212
Server-Code testen	213
Server-Code progressiv testen	213
Was Sie bei Server-Code testen	214
Wie Sie Server-Code testen	215
Eine neue Server-Instanz erstellen	216
Eine neue Server-Instanz per Hand erstellen	217
Einen Server mit einem Skript erstellen	218
Einen Server mit einem Stack-Management-Tool erstellen	218
Die Plattform für das automatische Erstellen von Servern konfigurieren	219
Einen Server mit einem Network-Provisioning-Tool erstellen	220
Server vorbereiten	221
Hot-Cloning eines Servers	221
Einen Server-Snapshot verwenden	222
Ein sauberes Server-Image erstellen	222
Eine neue Server-Instanz konfigurieren	223
Eine Server-Instanz ausbacken	224
Server-Images backen	225
Backen und Ausbacken kombinieren	225
Serverkonfiguration beim Erstellen eines Servers anwenden	226
Zusammenfassung	227
12 Änderungen an Servern managen	229
Patterns zum Changemanagement: Wann Änderungen angewendet werden	230
Antipattern: Apply on Change	230
Pattern: Continuous Configuration Synchronization	232
Pattern: Immutable Server	234
Wie Sie Serverkonfigurationscode anwenden	236
Pattern: Push Server Configuration	236
Pattern: Pull Server Configuration	238
Andere Ereignisse im Lebenszyklus eines Servers	241
Eine Server-Instanz stoppen und erneut starten	241
Eine Server-Instanz ersetzen	242

Einen ausgefallenen Server wiederherstellen	243
Zusammenfassung	244
13 Server-Images als Code	245
Ein Server-Image bauen	246
Warum ein Server-Image bauen?	246
Wie Sie ein Server-Image bauen	247
Tools zum Bauen von Server-Images	247
Online Image Building	248
Offline Image Building	251
Ursprungsinhalte für ein Server-Image	252
Aus einem Stock-Server-Image bauen.	252
Ein Server-Image von Grund auf bauen	253
Herkunft eines Server-Image und seiner Inhalte.	253
Ein Server-Image ändern	254
Ein frisches Image aufwärmen oder backen	254
Ein Server-Image versionieren.	255
Server-Instanzen aktualisieren, wenn sich ein Image ändert	257
Ein Server-Image in mehreren Teams verwenden.	258
Umgang mit größeren Änderungen an einem Image	259
Eine Pipeline zum Testen und Ausliefern eines Server-Image verwenden.	259
Build-Stage für ein Server-Image.	260
Test-Stage für ein Server-Image	261
Delivery-Stages für ein Server-Image.	262
Mehrere Server-Images verwenden	263
Server-Images für unterschiedliche Infrastruktur-Plattformen	263
Server-Images für unterschiedliche Betriebssysteme.	264
Server-Images für unterschiedliche Hardware-Architekturen.	264
Server-Images für unterschiedliche Rollen	264
Server-Images in Schichten erstellen.	265
Code für mehrere Server-Images verwenden	266
Zusammenfassung	267
14 Cluster als Code bauen	269
Lösungen für Anwendungs-Cluster.	270
Cluster as a Service	270
Packaged Cluster Distribution	271
Stack-Topologien für Anwendungs-Cluster	272
Monolithischer Stack, der Cluster as a Service nutzt	273
Monolithischer Stack für eine Packaged-Cluster-Lösung.	274
Pipeline für einen monolithischen Anwendungs-Cluster-Stack	275
Beispiel für mehrere Stacks in einem Cluster	278

Strategien zur gemeinsamen Verwendung von Anwendungs-Clustern .	280
Ein großes Cluster für alles	281
Getrennte Cluster für Auslieferungs-Stages	282
Cluster für die Governance	283
Cluster für Teams.	284
Service Mesh	284
Infrastruktur für FaaS Serverless.	286
Zusammenfassung	288

Teil IV Infrastruktur designen

15 Zentrale Praktik: Kleine, einfache Elemente	291
Für Modularität designen	292
Eigenschaften gut designer Komponenten	292
Regeln für das Designen von Komponenten	293
Design-Entscheidungen durch Testen	296
Infrastruktur modularisieren	297
Stack-Komponenten versus Stacks als Komponenten	297
Einen Server in einem Stack verwenden.	299
Grenzen zwischen Komponenten ziehen	303
Grenzen mit natürlichen Änderungsmustern abstimmen	303
Grenzen mit Komponenten-Lebenszyklen abstimmen	304
Grenzen mit Organisationsstrukturen abstimmen	306
Grenzen schaffen, die Resilienz fördern.	307
Grenzen schaffen, die Skalierbarkeit ermöglichen	307
Grenzen auf Sicherheits- und Governance-Aspekte abstimmen	311
Zusammenfassung	312
16 Stacks aus Komponenten bauen	313
Infrastruktur-Sprachen für Stack-Komponenten	314
Deklarativen Code mit Modulen wiederverwenden	314
Stack-Elemente dynamisch mit Bibliotheken erstellen	315
Patterns für Stack-Komponenten	316
Pattern: Facade Module	317
Antipattern: Obfuscation Module	318
Antipattern: Unshared Module	320
Pattern: Bundle Module	321
Antipattern: Spaghetti Module	322
Pattern: Infrastructure Domain Entity	325
Eine Abstraktionsschicht bauen	327
Zusammenfassung	328

17 Stacks als Komponenten einsetzen	329
Abhängigkeiten zwischen Stacks erkennen	329
Pattern: Resource Matching	330
Pattern: Stack Data Lookup	333
Pattern: Integration Registry Lookup	336
Dependency Injection	339
Zusammenfassung	342

Teil V Infrastruktur bereitstellen

18 Infrastruktur-Code organisieren	345
Projekte und Repositories organisieren	345
Ein Repository oder viele?	346
Ein Repository für alles	346
Ein eigenes Repository für jedes Projekt (Microrepo)	349
Mehrere Repositories mit mehreren Projekten	350
Unterschiedliche Arten von Code organisieren	351
Projektsupport-Dateien	351
Projektübergreifende Tests	352
Dedizierte Projekte für Integrationstests	353
Code anhand des Domänenkonzepts organisieren	354
Dateien mit Konfigurationswerten organisieren	354
Infrastruktur- und Anwendungscode managen	356
Infrastruktur und Anwendungen ausliefern	356
Anwendungen mit Infrastruktur testen	358
Infrastruktur vor der Integration testen	359
Infrastruktur-Code zum Deployen von Anwendungen nutzen	359
Zusammenfassung	361
19 Infrastruktur-Code ausliefern	363
Auslieferungsprozess von Infrastruktur-Code	363
Ein Infrastruktur-Projekt bauen	364
Infrastruktur-Code als Artefakt verpacken	365
Infrastruktur-Code mit einem Repository ausliefern	365
Projekte integrieren	368
Pattern: Build-Time Project Integration	369
Pattern: Delivery-Time Project Integration	373
Pattern: Apply-Time Project Integration	375
Infrastruktur-Tools durch Skripte verpacken	378
Konfigurationswerte zusammenführen	379
Wrapper-Skripte vereinfachen	380
Zusammenfassung	381

20 Team-Workflows	383
Die Menschen	384
Wer schreibt Infrastruktur-Code?	386
Code auf Infrastruktur anwenden.	388
Code von Ihrem lokalen Rechner aus anwenden.	388
Code von einem zentralisierten Service anwenden lassen	389
Private Infrastruktur-Instanzen	391
Quellcode-Banches in Workflows	392
Konfigurationsdrift verhindern.	394
Automatisierungs-Verzögerung minimieren	394
Ad-hoc-Anwendung vermeiden	395
Code kontinuierlich anwenden	395
Immutable Infrastruktur	395
Governance in einem Pipeline-basierten Workflow	396
Zuständigkeiten neu ordnen	397
Shift Left.	398
Ein Beispielprozess für Infrastructure as Code mit Governance ...	398
Zusammenfassung	399
21 Infrastruktur sicher ändern	401
Reduzieren Sie den Umfang von Änderungen	402
Kleine Änderungen.	404
Refaktorisieren – ein Beispiel	406
Unvollständige Änderungen in die Produktivumgebung übernehmen .	407
Parallele Instanzen	408
Abwärtskompatible Transformationen	411
Feature Toggles	413
Live-Infrastruktur ändern	415
Infrastruktur-Chirurgie	417
Expand and Contract.	419
Zero-Downtime-Änderungen.	422
Kontinuität	423
Kontinuität durch das Verhindern von Fehlern.	424
Kontinuität durch schnelles Wiederherstellen.	425
Kontinuierliches Disaster Recovery	426
Chaos Engineering	427
Für Ausfälle planen	428
Datenkontinuität in einem sich ändernden System	430
Sperren	430
Aufteilen.	430

Replizieren	431
Neu laden	431
Ansätze zur Datenkontinuität mischen.	432
Zusammenfassung.	432
Index	433