

DevOps

Wie IT-Projekte mit einem modernen Toolset
und der richtigen Kultur gelingen

DAS INHALTS- VERZEICHNIS

» Hier geht's
direkt
zum Buch

Auf einen Blick

1	Einleitung	15
2	Was ist DevOps?	23
3	Die Beispielfirma	49
4	Projektmanagement und Planung	63
5	Kollaboration beim Coden	77
6	Continuous Integration und der Build Prozess	121
7	Die Qualität sicherstellen	167
8	Continuous Delivery und Deployment	189
9	Den Dienst betreiben	245
10	Vom Monitoring zur Observability	295
11	Security und Compliance	339
12	Die DevOps-Transformation erfolgreich umsetzen	385
13	DevOps-Plattformen	417
14	Jenseits von Kultur und Tools	429

Inhalt

1	Einleitung	15
1.1	Kultur	17
1.2	Technik	17
1.3	Mein Weg zu DevOps und zu diesem Buch	18
1.4	Zielgruppe	20
1.5	Die Struktur des Buches	21
1.6	Feedback	21
1.7	Danke!	22
2	Was ist DevOps?	23
2.1	DevOps: Das große Ganze	24
2.1.1	CA(L)MS	25
2.1.2	The Three Ways	29
2.1.3	The Three Ways und CALMS	37
2.2	Missverständnisse rund um DevOps	37
2.2.1	Zu starker Fokus auf die Automatisierung	38
2.2.2	Mit DevOps, aber ohne Tests!	39
2.2.3	Falsches Verständnis der Teamstrukturierung	39
2.2.4	Nicht alle Wände niederreißen	40
2.2.5	Tools über Prozesse über Menschen	41
2.2.6	1:1-Kopien von Arbeitsweisen anderer Firmen	43
2.3	Der DevOps-Lifecycle	44
3	Die Beispielfirma	49
3.1	schick-gekleidet.de	50
3.2	Das Entwicklungsmodell	50
3.3	Das Business-Team – Anforderungsanalyse	51

3.4	Das Architekturteam – Design der Anwendung	52
3.5	Die Entwicklungsteams	52
3.5.1	Der Entwicklungsprozess	53
3.5.2	Integrationen mit Hindernissen	54
3.6	Das Qualitätssicherungsteam	55
3.7	Das Betriebsteam – Das Ops aus DevOps	56
3.7.1	Manuelles Bauen des Projektes	58
3.7.2	Deployment mit Hindernissen	58
3.7.3	Das Monitoring schlägt Alarm	58
3.8	Das Infrastrukturteam	59
3.9	Das Security-Team	60
3.10	Fazit	61

4 Projektmanagement und Planung 63

4.1	Der erste Schritt: Das agile Mindset	63
4.2	Projektmanagement für alle?	67
4.2.1	Jira kann (fast) alles	68
4.2.2	Menschen sind wichtiger als Prozesse	68
4.2.3	Gutes Projektmanagement nicht nur mit Jira	70
4.2.4	Mehr als nur ein reines Projektmanagement-Tool	72
4.2.5	Projektmanagement bei schick-gekleidet.de	74
4.3	Fazit	75

5 Kollaboration beim Coden 77

5.1	Die typischen Probleme bei der Verwaltung des Sourcecodes	78
5.1.1	Organisation rund um den Code	78
5.1.2	Abschottung aus vermeintlichen Sicherheitsgründen	80
5.1.3	Lange Entwicklungszeiten ermöglichen kaum schnelle Sicherheitsfixes	81
5.1.4	Entwicklungs-Workflow à la Kraut und Rüben	82
5.1.5	Big-Bang-Integrationen	82
5.1.6	Code-Reviews konnten ein wenig helfen	83
5.1.7	Erschwerte Einarbeitung durch Technical Debt	84

5.1.8	Hohe Lernkurve ohne Dokumentation	85
5.2	Die Organisation des Codes verbessern	86
5.3	An Git führt nichts vorbei	88
5.3.1	Git-Lösungen im Überblick	89
5.3.2	Entwicklungs-Workflows mit Git	90
5.3.3	Das Sourcecode-Management bei schick-gekleidet.de	96
5.4	Code-Reviews und Pair Programming	97
5.4.1	Code-Reviews	97
5.4.2	Code-Reviews vereinfachen	104
5.4.3	Pair Programming	107
5.5	Inner Sourcing – Code im Unternehmen teilen	109
5.5.1	Open Source	111
5.5.2	Der Weg zum Inner Sourcing	112
5.5.3	Vorteile von Inner Sourcing	114
5.5.4	Was ist eigentlich mit Monorepos?	117
5.6	Fazit	120

6 Continuous Integration und der Build Prozess 121

6.1	Die typischen Probleme im Build-Prozess	121
6.1.1	Onboarding mit Stolpersteinen	123
6.1.2	Seltene Integrationen führen zu Build-Schwierigkeiten	124
6.1.3	Nur wenige Tests	126
6.1.4	Ein Build-Server hinter verschlossenen Türen	126
6.2	Modernes Build-Management	128
6.3	Continuous Integration	131
6.4	CI-Server und die Pipelines	136
6.4.1	Der grundlegende Aufbau einer Pipeline	136
6.4.2	Skalierung und Reproduzierbarkeit	138
6.4.3	Deklarative Pipelines vs. Scripted Pipelines	140
6.5	DRY und KISS: »Don't repeat yourself« und »Keep it simple, stupid!«	142
6.5.1	Zentrale Pipelines vermeiden	142
6.5.2	Pipeline-Bausteine bereitstellen	144
6.5.3	Visibilität schaffen	145
6.6	Ein Überblick über CI-Server	145
6.6.1	Jenkins	146

6.6.2	GitLab CI/CD	153
6.6.3	GitHub Actions	158
6.6.4	Sonstige CI-Server und Tools	163
6.6.5	CI bei schick-gekleidet.de	164
6.7	Fazit	165

7 Die Qualität sicherstellen 167

7.1	Die typischen Probleme beim Testing	167
7.1.1	Die Teams in ihren Silos	168
7.1.2	Unterschiedliche Verständnisse von Anforderungen	169
7.1.3	Die Anzahl der Fehler als Metrik	170
7.1.4	Is it fixed yet?	171
7.2	Testen als Teil des DevOps-Prozesses	172
7.2.1	Tests in der Build-Pipeline	174
7.2.2	Unterschiedliche Tests für unterschiedliche Aufgaben	176
7.2.3	Tests automatisieren	179
7.2.4	Test-Driven Development	185
7.3	Fazit	187

8 Continuous Delivery und Deployment 189

8.1	Die typischen Probleme beim Release-Management	189
8.1.1	Separate Handhabung von Änderungen und Dokumentationen	190
8.1.2	Langwieriger Release-Prozess	191
8.1.3	Automatisierungen, die sich nicht lohnen	192
8.1.4	Anfeindungen zwischen den Teams	193
8.1.5	Deployment auf Produktivsystemen mit Hindernissen	193
8.1.6	Fazit	195
8.2	Continuous Delivery und Deployment implementieren	196
8.2.1	Devs und Ops zusammenbringen	196
8.2.2	QA-, Staging- und Prod-Umgebungen	202
8.2.3	Deployment an Freitagen	208
8.3	Build-Management für Deployments	210
8.3.1	Die Frage nach den Versionsnummern	210
8.3.2	Paketierung	211

8.3.3	Containerisierung	212
8.3.4	Container-Registry und Package-Registry	215
8.4	Rollbacks, Kanarienvögel und Feature Flags	217
8.4.1	Rollbacks	218
8.4.2	Schrittweises Aktivieren mittels Blue-Green- und Canary-Deployments	219
8.4.3	Feature Flags	223
8.5	Deployment-Ziele – Wohin mit dem Deployment?	226
8.5.1	Deployments mit Kubernetes orchestrieren	226
8.5.2	Deployments orchestrieren bei schick-gekleidet.de	242
8.6	Fazit	242
9	Den Dienst betreiben	245
9.1	Die typischen Probleme beim Betreiben der Dienste	245
9.1.1	Langwierige Planungen rund um die Infrastruktur	246
9.1.2	Hardware-Austausch mit Hindernissen	247
9.1.3	Unvorteilhafte Serverauslastung	248
9.1.4	Regelmäßige Ausfälle in der Nacht	249
9.2	Aufbrechen der stark gekoppelten Infrastruktur-Architektur	250
9.2.1	Cattle not Pets	251
9.2.2	Infrastruktur abstrahieren	253
9.2.3	Container für schnellere Deployments	257
9.3	Cloud-Computing	258
9.3.1	Was ist die Cloud?	258
9.3.2	Cloud-Modelle	259
9.3.3	Service-Modelle	260
9.3.4	Cloud-native	267
9.3.5	Die Cloud bei schick-gekleidet.de	268
9.4	Stärkere Zusammenarbeit von Dev und Ops	269
9.4.1	Die Bereitschaft ist für (fast) alle da!	270
9.4.2	Blameless Post-Mortems	272
9.4.3	Kommunikationslösungen und ChatOps	274
9.5	Konfigurationsmanagement: Everything as Code	276
9.5.1	Infrastructure as Code mit Terraform	278
9.5.2	Ansible vs. Puppet	281

9.6	Chaos-Engineering	286
9.6.1	Systeme zum Ausfall bringen	287
9.6.2	Chaos-Engineering ohne Chaos, aber mit Plan	289
9.7	Reliability Engineering	291
9.7.1	Site Reliability Engineering	291
9.7.2	Database-Reliability-Engineering	293
9.8	Fazit	294
 10 Vom Monitoring zur Observability		295
<hr/>		
10.1	Keine Sichtbarkeit bei schick-gekleidet.de	296
10.1.1	Ausfälle der Dienste sind von der Tagesordnung	296
10.1.2	Performance, Performance!	301
10.1.3	Logs	302
10.2	Mit Durchblick kommt Weitsicht	304
10.2.1	Observability-Engineering	304
10.2.2	Einblicke in Prozesse mit Tracing	307
10.2.3	A/B-Tests	309
10.2.4	Business-Monitoring	310
10.3	Tools für Monitoring, Observability und Tracing	312
10.3.1	Systeme mit Icinga/Nagios überwachen	312
10.3.2	Monitoring mit Metriken und Time-Series-Datenbanken	314
10.3.3	Datenvisualisierung mit Grafana	323
10.3.4	Error-Tracking	325
10.3.5	Distributed Tracing	326
10.3.6	Logging	328
10.3.7	Service-Meshes	329
10.3.8	Observability-Plattformen	330
10.3.9	Monitoring und Observability bei schick-gekleidet.de	332
10.4	Verfügbarkeit	333
10.4.1	SLA: Service-Level-Agreements	333
10.4.2	SLO: Service-Level-Objectives	335
10.4.3	SLI: Service-Level-Indicators	336
10.4.4	Error-Budgets	336
10.5	Fazit	337

11 Security und Compliance	339
11.1 Sicherheit stört den agilen Wasserfall	340
11.2 DevOps mit getrenntem Security-Team	343
11.2.1 To deploy or not to deploy?	344
11.2.2 Die Suche nach undokumentierten Abhängigkeiten	345
11.2.3 Frust und Blockade	346
11.3 DevSecOps: Sicherheit in DevOps einbauen	347
11.3.1 Die DevSecOps-Teamstruktur	348
11.3.2 Shift-Left: Fehler früher finden	350
11.3.3 Inner Sourcing sorgt für formelle Sicherheit	351
11.3.4 Security als fester Bestandteil des Entwicklungsprozesses	351
11.3.5 Mit Fehlern umgehen	353
11.4 Werkzeuge für mehr Sicherheit	355
11.4.1 Dashboards und Reporting	355
11.4.2 Pull und Merge Requests	356
11.4.3 Die Security-Scanner im Detail	356
11.5 Supply-Chain-Security	365
11.5.1 Angriffe auf die Supply-Chain	366
11.5.2 Software Bill Of Materials (SBOM)	367
11.5.3 Sicherheit der Build- und Deployment-Server	368
11.5.4 Nutzerkonten absichern	369
11.5.5 Kein Code ist guter Code	370
11.5.6 Security bei schick-gekleidet.de	371
11.6 Compliance	372
11.6.1 Compliance-Richtlinien definieren	373
11.6.2 Manuelle Compliance	375
11.6.3 Vollautomatische Compliance	376
11.6.4 Compliance bei schick-gekleidet.de	382
11.7 Fazit	383
12 Die DevOps-Transformation erfolgreich umsetzen	385
12.1 Die DevOps-Kultur einführen	385
12.1.1 Bottom-up oder Top-down?	386
12.1.2 Erste Schritte in der DevOps-Transformation	388

12.2	Mit DORA-Metriken den DevOps-Erfolg messbar machen	401
12.2.1	DORA-Metrik 1: Deployment Frequency	402
12.2.2	DORA-Metrik 2: Lead Time	404
12.2.3	DORA-Metrik 3: Change Failure Rate	404
12.2.4	DORA-Metrik 4: Time to Restore Service	405
12.2.5	DORA-Metrik 5: Operational Performance: Reliability	406
12.2.6	Erkenntnisse aus dem »State of DevOps Report«	407
12.3	Value Stream Mapping	408
12.3.1	Der Value Stream einer Pizza	409
12.3.2	Der Value Stream von schick-gekleidet.de	410
12.3.3	Durchführung eines Value Stream Mappings	412

13 DevOps-Plattformen 417

13.1	Toolchain-Komplexität	418
13.1.1	Schritt 0: Toolchains wachsen historisch	419
13.1.2	Schritt 1: Mehrere Toolchains zur Erhöhung der Wartbarkeit	420
13.1.3	Schritt 2: Standardisierte Tools, aber weiterhin mit viel »Klebeband«	421
13.1.4	Schritt 3: DevOps-Plattformen	423
13.2	DevOps-Plattformen im Überblick	424
13.2.1	GitLab	425
13.2.2	GitHub	425
13.2.3	Azure DevOps	426
13.2.4	Atlassian	427
13.2.5	Sonstige	427
13.3	Fazit	428

14 Jenseits von Kultur und Tools 429

14.1	Die Rolle von AI in DevOps	429
14.1.1	Arbeitserleichterung durch AI-gestützte Code-Generierung?	430
14.1.2	Mehr Code führt zu höherem Review-Bedarf!	433
14.1.3	AI-unterstützende Features	434
14.1.4	Datenschutz und Privacy	435
14.1.5	Das Gesamtkonzept macht's!	437

14.2 DataOps, MLOps – was es sonst noch alles gibt	438
14.2.1 DataOps	438
14.2.2 MLOps	439
14.2.3 AIOps	440
14.3 DevOps als Job	441
14.3.1 Die Frage nach den DevOps Engineers	441
14.3.2 Soft Skills	443
14.3.3 Der technische DevOps-Lernpfad	444
14.4 Fazit	454
Index	455