

Eberhard Wolff beschäftigt sich seit vielen Jahren mit Softwareentwicklung und -architektur. Er ist Autor zahlreicher Fachartikel und Bücher, regelmäßiger Sprecher auf internationalen Konferenzen und im Programmkomitee verschiedener Konferenzen vertreten. Er ist Fellow bei der innoQ.

Continuous Delivery und die Auswirkungen hat er in verschiedenen Projekten in unterschiedlichen Rollen kennengelernt. Der Ansatz verspricht, die Produktivität der IT-Projekte erheblich zu erhöhen, und hat Auswirkungen auf das Vorgehen, aber auch die Architektur und die Technologien. Daher lag es für ihn auf der Hand, dieses Buch zu schreiben.

Eberhard Wolff

Continuous Delivery

Der pragmatische Einstieg

2., aktualisierte und erweiterte Auflage



dpunkt.verlag

Eberhard Wolff
eberhard.wolff@gmail.com

Lektorat: René Schönfeldt
Copy-Editing: Petra Kienle, Fürstenfeldbruck
Herstellung: Nadine Thiele
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

ISBN
Print 978-3-86490-371-7
PDF 978-3-86491-930-5
ePub 978-3-86491-931-2
mobi 978-3-86491-932-9

2., aktualisierte und erweiterte Auflage 2016
Copyright © 2016 dpunkt.verlag GmbH
Wiebinger Weg 17
69123 Heidelberg

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

Inhaltsverzeichnis

1	Einleitung	1
1.1	Überblick über Continuous Delivery und das Buch	1
1.2	Warum überhaupt Continuous Delivery?	2
1.3	Für wen ist das Buch?	5
1.4	Neu in der 2. Auflage	5
1.5	Übersicht über die Kapitel	7
1.6	Pfade durch das Buch	8
1.7	Danksagung	10
2	Continuous Delivery: Was und wie?	13
2.1	Was ist Continuous Delivery?	13
2.2	Warum Software-Releases so kompliziert sind	13
2.3	Werte von Continuous Delivery	14
2.4	Vorteile von Continuous Delivery	17
2.4.1	Continuous Delivery für Time-to-Market	17
2.4.2	Continuous Delivery zur Risikominimierung	20
2.4.3	Schnelleres Feedback und Lean	23
2.5	Aufbau und Struktur einer Continuous-Delivery-Pipeline	24
2.6	Links & Literatur	28
3	Infrastruktur bereitstellen	29
3.1	Einleitung	29
3.2	Installationsskripte	31
3.3	Chef	34
3.3.1	Technische Grundlagen	37
3.3.2	Chef Solo	44

3.3.3	Chef Solo: Fazit	46
3.3.4	Knife und Chef Server	46
3.3.5	Chef Server: Fazit	51
3.4	Vagrant	51
3.4.1	Ein Beispiel mit Chef und Vagrant	53
3.4.2	Vagrant: Fazit	55
3.5	Docker	55
3.5.1	Dockers Lösung	56
3.5.2	Docker-Container erstellen	59
3.5.3	Beispielanwendung mit Docker betreiben	61
3.5.4	Docker und Vagrant	63
3.5.5	Docker Machine	66
3.5.6	Komplexe Konfigurationen mit Docker	68
3.5.7	Docker Compose	70
3.6	Immutable Server	73
3.7	Infrastructure as Code	74
3.8	Platform as a Service (PaaS)	77
3.9	Umgang mit Daten und Datenbanken	79
3.10	Fazit	82
3.11	Links & Literatur	83
4	Build-Automatisierung und Continuous Integration	87
4.1	Überblick	87
4.2	Build-Automatisierung und Build-Tools	88
4.2.1	Ant	90
4.2.2	Maven	90
4.2.3	Gradle	95
4.2.4	Weitere Build-Tools	98
4.2.5	Das geeignete Tool auswählen	99
4.2.6	Links und Literatur	100
4.2.7	Experimente und selber ausprobieren	100
4.3	Unit-Tests	101
4.3.1	»Gute« Unit-Tests schreiben	103
4.3.2	TDD – Test-driven Development	105
4.3.3	Clean Code und Software Craftmanship	106
4.3.4	Links und Literatur	106
4.3.5	Experimente und selber ausprobieren	107
4.4	Continuous Integration	107
4.4.1	Jenkins	108
4.4.2	Continuous-Integration-Infrastruktur	114
4.4.3	Fazit	115

4.4.4	Links und Literatur	116
4.4.5	Experimente und selber ausprobieren	116
4.5	Codequalität messen	118
4.5.1	SonarQube	120
4.5.2	Links und Literatur	122
4.5.3	Experimente und selber ausprobieren	122
4.6	Artefakte managen	123
4.6.1	Integration in den Build	125
4.6.2	Weiterreichende Funktionen von Repositories	127
4.6.3	Links und Literatur	127
4.6.4	Experimente und selber ausprobieren	127
4.7	Fazit	128
5	Akzeptanztests	131
5.1	Einführung	131
5.2	Die Test-Pyramide	131
5.3	Was sind Akzeptanztests?	135
5.4	GUI-basierte Akzeptanztests	139
5.5	Alternative Werkzeuge für GUI-Tests	145
5.6	Textuelle Akzeptanztests	147
5.7	Alternative Frameworks	150
5.8	Strategien für Akzeptanztests	152
5.9	Fazit	154
5.10	Links & Literatur	154
6	Kapazitätstests	157
6.1	Einführung	157
6.2	Kapazitätstests – wie?	158
6.3	Kapazitätstests implementieren	163
6.4	Kapazitätstests mit Gatling	164
6.5	Alternativen zu Gatling	169
6.6	Fazit	171
6.7	Links & Literatur	172
7	Exploratives Testen	173
7.1	Einleitung	173
7.2	Warum explorative Tests?	173
7.3	Wie vorgehen?	175

7.4	Fazit	179
7.5	Links & Literatur	180
8	Deploy – der Rollout in Produktion	181
8.1	Einleitung	181
8.2	Rollout und Rollback	182
8.3	Roll Forward	183
8.4	Blue/Green Deployment	185
8.5	Canary Releasing	186
8.6	Continuous Deployment	188
8.7	Virtualisierung	190
8.8	Jenseits der Webanwendungen	192
8.9	Fazit	193
8.10	Links und Literatur	194
9	Operate – Produktionsbetrieb der Anwendungen	195
9.1	Einleitung	195
9.2	Herausforderungen im Betrieb	196
9.3	Log-Dateien	198
9.3.1	Werkzeuge zum Verarbeiten von Log-Dateien ..	200
9.3.2	Logging in der Beispielanwendung	202
9.4	Logs der Beispielanwendung analysieren	203
9.4.1	Experimente und selber ausprobieren	208
9.5	Andere Technologien für Logs	211
9.6	Fortgeschrittene Log-Techniken	212
9.7	Monitoring	213
9.8	Metriken mit Graphite	214
9.9	Metriken in der Beispielanwendung	216
9.9.1	Experimente und selber ausprobieren	217
9.10	Andere Monitoring-Lösungen	219
9.11	Weitere Herausforderungen beim Betrieb der Anwendung	220
9.12	Fazit	221
9.13	Links & Literatur	222

10	Continuous Delivery im Unternehmen einführen	225
10.1	Einleitung	225
10.2	Continuous Delivery von Anfang an	225
10.3	Value Stream Mapping	226
10.4	Weitere Optimierungsmaßnahmen	229
10.5	Zusammenfassung	233
10.6	Links & Literatur	233
11	Continuous Delivery und DevOps	235
11.1	Einführung	235
11.2	Was ist DevOps?	235
11.3	Continuous Delivery und DevOps	239
11.4	Continuous Delivery ohne DevOps?	243
11.5	Fazit	245
11.6	Links & Literatur	246
12	Continuous Delivery, DevOps und Softwarearchitektur	247
12.1	Einleitung	247
12.2	Softwarearchitektur	247
12.3	Komponentenaufteilung für Continuous Delivery optimieren	250
12.4	Schnittstellen	252
12.5	Datenbanken	255
12.6	Microservices	258
12.7	Umgang mit neuen Features	261
12.8	Fazit	264
12.9	Links & Literatur	265
13	Fazit: Was bringt's?	267
13.1	Links & Literatur	268
	Index	269