

Continuous Delivery

Eberhard Wolff beschäftigt sich seit vielen Jahren mit Softwareentwicklung und -architektur. Er ist Autor zahlreicher Fachartikel sowie Bücher und regelmäßiger Sprecher auf internationalen Konferenzen. Außerdem ist er im Programmkomitee verschiedener Konferenzen vertreten. Er arbeitet als selbstständiger Berater und Trainer. Daneben ist er der Leiter des Technologiebeirats der adesso AG, einem großen IT-Dienstleister.

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



dpunkt.verlag

Eberhard Wolff
eberhard.wolff@gmail.com

Lektorat: René Schönfeldt
Copy-Editing: Petra Kienle, Fürstfeldbruck
Herstellung: Frank Heidt
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.

978-3-86490-208-6

1. Auflage 2015
Copyright © 2015 dpunkt.verlag GmbH
Wieblinger 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	Übersicht über die Kapitel	6
1.5	Pfade durch das Buch	7
1.6	Danksagung	9
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	30
3.3	Chef	34
3.3.1	Technische Grundlagen	37
3.3.2	Chef Solo	43
3.3.3	Chef Solo: Fazit	45

3.3.4	Knife und Chef Server	45
3.3.5	Chef Server: Fazit	49
3.4	Vagrant	49
3.4.1	Ein Beispiel mit Chef und Vagrant	51
3.4.2	Vagrant: Fazit	53
3.5	Docker	53
3.5.1	Dockers Lösung	54
3.5.2	Docker-Container erstellen	56
3.5.3	Beispielanwendung mit Docker betreiben	59
3.5.4	Docker und Vagrant	61
3.5.5	Komplexe Konfigurationen mit Docker	63
3.6	Infrastructure as Code	64
3.7	Platform as a Service (PaaS)	66
3.8	Umgang mit Daten und Datenbanken	69
3.9	Fazit	72
3.10	Links & Literatur	73
4	Build-Automatisierung und Continuous Integration	77
4.1	Überblick	77
4.2	Build-Automatisierung und Build-Tools	78
4.2.1	Ant	80
4.2.2	Maven	80
4.2.3	Gradle	85
4.2.4	Weitere Build-Tools	88
4.2.5	Auswahl des geeigneten Tools	88
4.2.6	Links und Literatur	89
4.3	Unit-Tests	91
4.3.1	»Gute« Unit-Tests schreiben	92
4.3.2	TDD – Test Driven Development	94
4.3.3	Clean Code und Software Craftsmanship	95
4.3.4	Links und Literatur	96
4.4	Continuous Integration	96
4.4.1	Jenkins	98
4.4.2	Continuous-Integration-Infrastruktur	101
4.4.3	Fazit	103
4.4.4	Links und Literatur	103
4.5	Messung von Codequalität	105
4.5.1	SonarQube	107
4.5.2	Links und Literatur	109

4.6	Management von Artefakten	110
4.6.1	Integration in den Build	112
4.6.2	Weiterreichende Funktionen von Repositories .	114
4.6.3	Links und Literatur	114
4.7	Fazit	115
5	Akzeptanztests	117
5.1	Einführung	117
5.2	Die Test-Pyramide	117
5.3	Was sind Akzeptanztests?	121
5.4	GUI-basierte Akzeptanztests	125
5.5	Alternative Werkzeuge für GUI-Tests	131
5.6	Textuelle Akzeptanztests	133
5.7	Alternative Frameworks	136
5.8	Strategien für Akzeptanztests	137
5.9	Fazit	139
5.10	Links & Literatur	140
6	Kapazitätstests	143
6.1	Einführung	143
6.2	Kapazitätstests – wie?	144
6.3	Kapazitätstests implementieren	149
6.4	Kapazitätstests mit Gatling	150
6.5	Alternativen zu Gatling	155
6.6	Fazit	157
6.7	Links & Literatur	158
7	Exploratives Testen	159
7.1	Einleitung	159
7.2	Warum explorative Tests?	159
7.3	Wie vorgehen?	161
7.4	Fazit	165
7.5	Links & Literatur	166
8	Deploy – der Rollout in Produktion	167
8.1	Einleitung	167
8.2	Rollout und Rollback	168

8.3	Roll Forward	169
8.4	Blue/Green Deployment	171
8.5	Canary Releasing	172
8.6	Continuous Deployment	174
8.7	Virtualisierung	176
8.8	Jenseits der Webanwendungen	178
8.9	Fazit	179
8.10	Links und Literatur	180
9	Operate – Produktionsbetrieb der Anwendungen	181
9.1	Einleitung	181
9.2	Herausforderungen im Betrieb	182
9.3	Log-Dateien	184
9.3.1	Werkzeuge zum Verarbeiten von Log-Dateien ..	186
9.3.2	Logging in der Beispielanwendung	188
9.4	Logs der Beispielanwendung analysieren	189
9.5	Andere Technologien für Logs	195
9.6	Fortgeschrittene Log-Techniken	196
9.7	Monitoring	197
9.8	Metriken mit Graphite	198
9.9	Metriken in der Beispielanwendung	200
9.10	Andere Monitoring-Lösungen	202
9.11	Weitere Herausforderungen beim Betrieb der Anwendung	203
9.12	Fazit	205
9.13	Links & Literatur	205
10	Einführen von Continuous Delivery in Unternehmen	209
10.1	Einleitung	209
10.2	Continuous Delivery von Anfang an	209
10.3	Value Stream Mapping	210
10.4	Weitere Optimierungsmaßnahmen	213
10.5	Zusammenfassung	217
10.6	Links & Literatur	217

11	Continuous Delivery und DevOps	219
11.1	Einführung	219
11.2	Was ist DevOps?	219
11.3	Continuous Delivery und DevOps	223
11.4	Wohin geht die Reise?	227
11.5	Fazit	229
11.6	Links & Literatur	230
12	Continuous Delivery, DevOps und Softwarearchitektur	231
12.1	Einleitung	231
12.2	Softwarearchitektur	231
12.3	Komponentenaufteilung für Continuous Delivery optimieren	234
12.4	Schnittstellen	236
12.5	Datenbanken	239
12.6	Umgang mit neuen Features	242
12.7	Fazit	246
12.8	Links & Literatur	247
13	Fazit: Was bringt's?	249
13.1	Links & Literatur	250
	Index	251