

Auf einen Blick

1	Einführung	13
TEIL I The Big Picture		19
2	Jenkins für Eilige	21
3	Das große B(u)ild der Automatisierung	37
4	Software testen – aber wie?	61
5	Release-Management in einer agilen Welt	97
6	Source-Control-Management	125
TEIL II Continuous-Integration-Server		157
7	Einen CI-Server mit Jenkins einrichten und betreiben	159
8	Build-Management	219
9	Jenkins-Build-Jobs	273
10	Qualitätskontrolle	337
11	Sonatype Nexus: Der Repository-Manager	363

Inhalt

Materialien zum Buch	11
----------------------------	----

1 Einführung 13

1.1 Was Sie schon wissen sollten und was Sie lernen werden	13
1.2 Worüber wir reden: Eine CI-Pipeline für Ihre Softwareprojekte	15

TEIL I The Big Picture

2 Jenkins für Eilige 21

2.1 Jenkins in 10 Minuten	21
2.2 Jenkins- und DevOps-Fachbegriffe	28
2.3 Fazit	34

3 Das große B(u)ild der Automatisierung 37

3.1 Die Rollen von DevOps im Softwareentwicklungsprozess	43
3.2 Infrastruktur für Softwareentwicklungsprojekte	48
3.2.1 Source-Control-Management	50
3.2.2 Build-Server	54
3.2.3 Repository-Server	55
3.3 Sonderfall: Microsoft und .NET in der CI-Pipeline	56
3.4 Fazit	58

4 Software testen – aber wie? 61

4.1 Komponententests und das Test-Driven Development	65
4.1.1 Erste Schritte zu einer Automatisierung	66

4.1.2	Testgetriebene Entwicklung mit JUnit	68
4.2	Akzeptanztests und Behavior-Driven Development	78
4.2.1	Organisation von Akzeptanztests	79
4.2.2	Verhaltensgetriebene Tests mit JGiven	80
4.2.3	Reibungslose Zusammenarbeit dank verständlicher API-Dokumentation	82
4.3	Integrationstests	84
4.3.1	Einfache Browsertests mit HttpUnit	86
4.3.2	Integrationstests für RESTful Services	87
4.4	Fazit	93

5 Release-Management in einer agilen Welt 97

5.1	Sprints vs. Release	101
5.2	Semantische Versionsnummern	102
5.3	Die Gefahren von Continuous Delivery	108
5.3.1	Deployment vs. Delivery	110
5.3.2	Domain-Driven Design (DDD)	114
5.3.3	Feature-Driven Design	119
5.3.4	Feature-Toggle und Feature-Flag	121
5.4	Fazit	123

6 Source-Control-Management 125

6.1	Die Rolle von SCM in der CI-Serverumgebung	125
6.2	Git und Subversion	127
6.3	Die Merging-Hölle	131
6.4	Branch-Modelle und Pull-Requests	137
6.4.1	Ordnung halten durch Namenskonventionen: Branch by Release	138
6.4.2	Pull-Requests	141
6.4.3	GitFlow und seine Tücken	144
6.5	Rebase vs. Merge: Code integrieren	146
6.6	Feature-Branches nutzen	149
6.6.1	Gefahrenstelle Feature-Flags	151

6.6.2	Versionen und Iterationen in Branches einordnen	152
6.7	Fazit	154

TEIL II Continuous-Integration-Server

7	Einen CI-Server mit Jenkins einrichten und betreiben	159
----------	---	------------

7.1	Voraussetzungen	159
7.2	Drei Wege zu einer Jenkins-Server-Installation	161
7.2.1	Standalone-Installation	163
7.2.2	Installation als Web-Application mit Tomcat	164
7.2.3	Installation als Docker-Image	167
7.3	Grundkonfiguration und Plug-ins	173
7.3.1	Der Plugin Manager	178
7.3.2	Monitoring	181
7.3.3	Datensicherung	182
7.3.4	Troubleshooting mit Logfiles	184
7.4	Jenkins als verteiltes System aufsetzen	186
7.4.1	Agenten im Cluster betreiben	191
7.4.2	Docker-Container als Build-Exekutor	196
7.4.3	Thin Agents: Jenkins Swarm	204
7.4.4	Build-Jobs registrierten Agenten zuweisen	207
7.4.5	Wege in die Cloud mit Jenkins X	208
7.5	Benutzerverwaltung	210
7.5.1	Rechte vergeben	211
7.5.2	Technische Nutzerkonten	214
7.6	Fazit	217

8	Build-Management	219
----------	-------------------------------	------------

8.1	Historisches: Build-Server, IDEs und das Chaos beim Programmieren	220
8.2	Die Build-Logik	222
8.3	Anwendungsarten: Mobile, Desktop, Web-App	222

8.4	Anwendungskonfiguration und Datenhaltung	224
8.5	Datenbanken	229
8.5.1	Flyway	233
8.5.2	Liquibase	235
8.5.3	DbUp für Microsoft SQL Server	236
8.6	Backend: Java und Java-Enterprise	237
8.6.1	Apache Ant	237
8.6.2	Apache Maven	241
8.6.3	Gradle	248
8.7	Frontend: JavaScript, Node.js und Bower	250
8.8	Mobile Devices	255
8.9	Exotische Welten: PHP, Ruby und Co. in Docker-Containern	258
8.9.1	PHP	259
8.9.2	Ruby	268
8.9.3	Microsoft .NET	269
8.10	Fazit	270

9 Jenkins-Build-Jobs 273

9.1	Build-Jobs schreiben	274
9.1.1	Die Ansicht »Jenkins-Job«	282
9.1.2	Fehleranalyse mit der Build-History	284
9.1.3	Parametrisierte Builds	289
9.1.4	Jenkins-Jobs versionieren	295
9.1.5	Ordnung schaffen mit Views und dem Folder-Plug-in	296
9.2	Bestehende Jenkins-Jobs optimieren	299
9.3	Jenkins-Pipelines mit Blue Ocean	303
9.3.1	Sequenzielle (lineare) Pipelines	306
9.3.2	Konkurrierende (parallele) Pipelines	308
9.3.3	Jenkins-Jobs als Pipeline-Schritte	310
9.3.4	Blue Ocean: Pipelines und Dashboard	313
9.4	Jenkins mit der JobDSL automatisieren	320
9.5	Build-Nummern vergeben – oder nicht?	325
9.6	Jenkins RESTful API	327

9.7	Deployments automatisieren:	
	Scripting mit Bash, PowerShell und Batch	330
9.8	Fazit	334

10	Qualitätskontrolle	337
-----------	---------------------------	-----

10.1	Was ist überhaupt »Qualität«?	338
10.2	Metriken, und was sie verschweigen	340
10.3	Jenkins-Plug-ins für mehr Qualität: JaCoCo, OWASP und Maven Site	347
10.4	SonarQube	352
10.5	Fazit	360

11	Sonatype Nexus: Der Repository-Manager	363
-----------	---	-----

11.1	Versionen und Zusatzprodukte	364
11.2	Nexus selbst betreiben	365
	11.2.1 Nexus im Jenkins-Server bekannt machen	368
	11.2.2 Allgemeine Einstellungen	371
11.3	Konfiguration der Repositories	372
	11.3.1 Verschiedene Repository-Typen	373
	11.3.2 Repositories anlegen	374
	11.3.3 Berechtigungen vergeben	377
11.4	Artefakte auf Maven Central veröffentlichen	379
11.5	Fazit	390

Anhang	
---------------	--

A	Abkürzungsverzeichnis	391
B	Literaturliste	393

Index	395
-------------	-----