

**Ciągłe dostarczanie oprogramowania : kompletny przewodnik /
Eberhard Wolff. – Gliwice, cop. 2018**

Spis treści

Podziękowania	13
O autorze	14
Wprowadzenie	15
Część I. Podstawy	21
Rozdział 1. Ciągłe dostarczanie — co i jak	23
1.1. Wprowadzenie — czym jest ciągłe dostarczanie?	23
1.2. Dlaczego udostępnianie oprogramowania jest tak skomplikowane?	23
1.2.1. Ciągła integracja daje nadzieję	24
1.2.2. Powolne i ryzykowne procesy	24
1.2.3. Szybka praca jest możliwa	24
1.3. Wartość ciągłego dostarczania	24
1.3.1. Regularność	25
1.3.2. Możliwość śledzenia i sprawdzalność zmian	25
1.3.3. Regresja	26
1.4. Korzyści płynące z ciągłego dostarczania	26
1.4.1. Ciągłe dostarczanie w celu przyspieszenia udostępniania	26
1.4.2. Przykład	27
1.4.3. Implementowanie funkcji i udostępnianie jej w środowisku produkcyjnym	27
1.4.4. Przejście do następnej funkcji	27
1.4.5. Ciągłe dostarczanie zapewnia przewagę konkurencyjną	27
1.4.6. Scenariusz bez ciągłego dostarczania	28
1.4.7. Ciągłe dostarczanie i Lean Startup	28
1.4.8. Wpływ na proces rozwoju produktu	28
1.4.9. Minimalizowanie ryzyka za pomocą ciągłego dostarczania	29
1.4.10. Szybsze informacje zwrotne i podejście Lean	32
1.5. Procesy generowania i struktura potoku ciągłego dostarczania	32
1.5.1. Przykład	34
1.6. Wnioski	35
Rozdział 2. Zapewnianie infrastruktury	37
2.1. Wprowadzenie	37
2.1.1. Automatyzowanie infrastruktury — przykład	38
2.2. Skrypty instalacyjne	38

2.2.1. Problemy związane ze standardowymi skryptami instalacyjnymi	38
2.3. Chef	41
2.3.1. Chef a Puppet	41
2.3.2. Inne możliwości	42
2.3.3. Podstawy techniczne	43
2.3.4. Chef Solo	48
2.3.5. Chef Solo — wnioski	49
2.3.6. Knife i Chef Server	49
2.3.7. Chef Server — wnioski	52
2.4. Vagrant	53
2.4.1. Przykład zastosowania Chef a i Vagranta	54
2.4.2. Vagrant — wnioski	56
2.5. Docker	56
2.5.1. Rozwiązanie oparte na Dockerze	56
2.5.2. Tworzenie kontenerów Dockera	58
2.5.3. Uruchamianie przykładowej aplikacji za pomocą Dockera	60
2.5.4. Docker i Vagrant	62
2.5.5. Docker Machine	63
2.5.6. Tworzenie złożonych konfiguracji za pomocą Dockera	65
2.5.7. Docker Compose	67
2.6. Niemodyfikowalny serwer	69
2.6.1. Wady idempotencji	69
2.6.2. Serwer niemodyfikowalny i Docker	69
2.7. Infrastruktura jako kod	70
2.7.1. Testowanie infrastruktury w postaci kodu	71
2.8. Platforma jako usługa	72
2.9. Obsługa danych i baz	73
2.9.1. Zarządzanie schematami	74
2.9.2. Dane testowe i główne	75
2.10. Wnioski	76

Część II. Potok ciągłego dostarczania **77**

Rozdział 3. Automatyzacja procesu budowania i ciągła integracja **79**

3.1. Wprowadzenie	79
3.1.1. Automatyzacja procesu budowania — przykład	79
3.2. Automatyzowanie procesu budowania i narzędzia do budowania	80
3.2.1. Narzędzia do budowania w świecie Javy	80
3.2.2. Ant	81
3.2.3. Maven	82
3.2.4. Gradle	86
3.2.5. Dodatkowe narzędzia do budowania	88
3.2.6. Wybór odpowiedniego narzędzia	89

3.3. Testy jednostkowe	90
3.3.1. Pisanie dobrych testów jednostkowych	91
3.3.2. Programowanie sterowane testami	93
3.3.3. Ruchy Clean Code i Software Craftmanship	94
3.4. Ciągła integracja	94
3.4.1. Jenkins	95
3.4.2. Infrastruktura do ciągłej integracji	100
3.4.3. Wnioski	101
3.5. Pomiar jakości kodu	103
3.5.1. SonarQube	104
3.6. Zarządzanie artefaktami	106
3.6.1. Integracja z procesem budowania	108
3.6.2. Zaawansowane funkcje repozytoriów	110
3.7. Wnioski	111
Rozdział 4. Testy akceptacyjne	113
4.1. Wprowadzenie	113
4.1.1. Testy akceptacyjne —przykład	113
4.2. Piramida testów	114
4.3. Czym są testy akceptacyjne?	116
4.3.1. Zautomatyzowane testy akceptacyjne	117
4.3.2. Więcej niż wzrost wydajności	117
4.3.3. Testy ręczne	118
4.3.4. A co z klientem?	118
4.3.5. Testy akceptacyjne a testy jednostkowe	118
4.3.6. Środowiska testowe	119
4.4. Testy akceptacyjne oparte na interfejsie GUI	120
4.4.1. Problemy z testami z użyciem interfejsu GUI	120
4.4.2. Stosowanie abstrakcji	120
4.4.3. Automatyzacja z użyciem narzędzia Selenium	121
4.4.4. Interfejs API WebDriver	121
4.4.5. Testy bez użycia przeglądarki — HtmlUnit	121
4.4.6. Interfejs API WebDriver dla Selenium	121
4.4.7. Środowisko IDE dla Selenium	121
4.4.8. Problemy ze zautomatyzowanymi testami interfejsu GUI	123
4.4.9. Wykonywanie testów z użyciem interfejsu GUI	123
4.4.10. Eksportowanie testów jako kodu	123
4.4.11. Ręczne modyfikowanie przypadków testowych	123
4.4.12. Dane testowe	124
4.4.13. Obiekty reprezentujące strony	124
4.5. Inne narzędzia do przeprowadzania testów z użyciem interfejsu GUI	125
4.5.1. PhantomJS	125
4.5.2. Windmill	125
4.6. Tekstowe testy akceptacyjne	126

4.6.1. Podejście BDD	126
4.6.2. Różne adaptory	128
4.7. Inne platformy	129
4.8. Strategie przeprowadzania testów akceptacyjnych	130
4.8.1. Właściwe narzędzie	131
4.8.2. Błyskawiczne informacje zwrotne	131
4.8.3. Pokrycie testami	131
4.9. Wnioski	132

Rozdział 5. Testy wydajności **133**

5.1. Wprowadzenie	133
5.1.1. Testy wydajności — przykład	133
5.2. Testy wydajności — jak je przeprowadzać?	133
5.2.1. Cele testów wydajności	134
5.2.2. Środowiska i ilość danych	134
5.2.3. Testy szybkości tylko po zakończeniu implementowania kodu?	134
5.2.4. Testy wydajności = zarządzanie ryzykiem	135
5.2.5. Symulowanie działań użytkowników	135
5.2.6. Dokumentowanie wymogów związanych z szybkością	135
5.2.7. Sprzęt potrzebny w testach wydajności	135
5.2.8. Chmura i wirtualizacja	136
5.2.9. Minimalizowanie ryzyka dzięki ciągłemu testowaniu	136
5.2.10. Testy wydajności — sensowne czy nie?	137
5.3. Implementowanie testów wydajności	137
5.4. Testy wydajności z użyciem narzędzia Gatling	139
5.4.1. Wersja demonstracyjna a praktyka	142
5.5. Narzędzia używane zamiast Gatlinga	143
5.5.1. Grinder	143
5.5.2. Apache JMeter	144
5.5.3. Tsung	144
5.5.4. Rozwiązania komercyjne	144
5.6. Wnioski	145

Rozdział 6. Testy eksploracyjne **147**

6.1. Wprowadzenie	147
6.1.1. Testy eksploracyjne — przykład	147
6.2. Po co stosować testy eksploracyjne?	147
6.2.1. Czasem testy ręczne i tak są lepsze	148
6.2.2. Testy z udziałem klientów	148
6.2.3. Testy ręczne wymagań нефункциональных	148
6.3. Jak zabrać się za testy eksploracyjne?	149
6.3.1. Zadanie określa testy	149
6.3.2. Zautomatyzowane środowisko	149
6.3.3. Przypadki pokazowe jako punkt wyjścia	149
6.3.4. Przykład — aplikacja z obszaru handlu elektronicznego	149

6.3.5. Testy wersji beta	150
6.3.6. Testy oparte na sesji	150
6.4. Wnioski	152

Rozdział 7. Wdrażanie — udostępnianie w środowisku produkcyjnym **153**

7.1. Wprowadzenie	153
7.1.1. Wdrażanie — przykład	154
7.2. Udostępnianie i wycofywanie	154
7.2.1. Korzyści	154
7.2.2. Wady	154
7.3. Zastępowanie nową wersją	155
7.3.1. Korzyści	155
7.3.2. Wady	155
7.4. Wdrażanie w modelu niebieskie-zielone	156
7.4.1. Korzyści	156
7.4.2. Wady	156
7.5. Udostępnianie „kanarkowe”	157
7.5.1. Korzyści	158
7.5.2. Wady	158
7.6. Ciągłe wdrażanie	158
7.6.1. Korzyści	159
7.6.2. Wady	160
7.7. Wirtualizacja	160
7.7.1. Hosty fizyczne	161
7.8. Poza aplikacje sieciowe	161
7.9. Wnioski	162

Rozdział 8. Eksploatacja **163**

8.1. Wprowadzenie	163
8.1.1. Eksploatacja — przykład	163
8.2. Trudności z eksploatacją oprogramowania	164
8.3. Pliki dziennika	165
8.3.1. Co należy rejestrować?	165
8.3.2. Narzędzia do przetwarzania plików dziennika	167
8.3.3. Rejestrowanie danych w przykładowej aplikacji	168
8.4. Analizowanie dzienników przykładowej aplikacji	169
8.4.1. Analizowanie danych za pomocą Kibany	171
8.4.2. ELK — skalowalność	172
8.5. Inne technologie obsługi dzienników	175
8.6. Zaawansowane techniki rejestrowania dzienników	176
8.6.1. Anonimizacja	176
8.6.2. Szybkość działania	176
8.6.3. Czas	176
8.6.4. Operacyjna baza danych	177

8.7. Monitorowanie	177
8.8. Pomiary z użyciem narzędzia Graphite	177
8.9. Pomiary w przykładowej aplikacji	179
8.9.1. Struktura przykładu	179
8.10. Inne rozwiązania z obszaru monitorowania	181
8.11. Inne wyzwania związane z eksploatacją aplikacji	182
8.11.1. Skrypty	182
8.11.2. Aplikacje w centrum danych klienta	183
8.12. Wnioski	183

Część III. Zarządzanie, kwestie organizacyjne i architektura w obszarze ciągłego dostarczania **185**

Rozdział 9. Wprowadzanie ciągłego dostarczania w organizacji **187**

9.1. Wprowadzenie	187
9.2. Ciągłe dostarczanie od początku projektu	187
9.3. Odwzorowywanie strumienia wartości	188
9.3.1. Odwzorowywanie strumienia wartości pozwala opisać sekwencję zdarzeń	188
9.3.2. Optymalizacje	189
9.4. Dodatkowe sposoby optymalizacji	190
9.4.1. Inwestycje wysokiej jakości	190
9.4.2. Koszty	190
9.4.3. Korzyści	191
9.4.4. Nie dodawaj kodu, gdy proces budowania kończy się niepowodzeniem!	191
9.4.5. Zatrzymywanie taśmy	191
9.4.6. Pięć pytań „dlaczego”	192
9.4.7. DevOps	192
9.5. Wnioski	193

Rozdział 10. Ciągłe dostarczanie i DevOps **195**

10.1. Wprowadzenie	195
10.2. Czym jest model DevOps?	195
10.2.1. Problemy	195
10.2.2. Perspektywa klienta	196
10.2.3. Pionierska firma — Amazon	196
10.2.4. DevOps	197
10.3. Ciągłe dostarczanie i DevOps	198
10.3.1. DevOps — więcej niż ciągłe dostarczanie	198
10.3.2. Pełna odpowiedzialność i samoorganizowanie się	199
10.3.3. Decyzje z obszaru technologii	199
10.3.4. Mniej scentralizowanej kontroli	200
10.3.5. „Pluralizm” w obszarze technologii	200
10.3.6. Wymiana między zespołami	200

10.3.7. Architektura	201
10.4. Ciągłe dostarczanie bez modelu DevOps?	202
10.4.1. Końcowa część potoku ciągłego dostarczania	202
10.5. Wnioski	203

Rozdział 11. Ciągłe dostarczanie, DevOps i architektura oprogramowania	205
11.1. Wprowadzenie	205
11.2. Architektura oprogramowania	205
11.2.1. Po co tworzyć architekturę oprogramowania?	206
11.3. Optymalizowanie architektury pod kątem ciągłego dostarczania	207
11.3.1. Mniejsze jednostki wdrażania	208
11.4. Interfejsy	209
11.4.1. Prawo Postela lub zasada odporności	210
11.4.2. Projektowanie pod kątem niepowodzeń	210
11.4.3. Stan	211
11.5. Bazy danych	211
11.5.1. Zapewnianie stabilności baz danych	211
11.5.2. Baza danych = komponent	212
11.5.3. Widoki i procedury składowane	212
11.5.4. Baza danych dla komponentu	213
11.5.5. Bazy typu NoSQL	213
11.6. Mikrouслуги	213
11.6.1. Mikrouслуги i ciągłe dostarczanie	214
11.6.2. Wprowadzanie ciągłego dostarczania z użyciem mikrouslug	214
11.6.3. Mikrouслуги wymagają ciągłego dostarczania	214
11.6.4. Struktura organizacyjna	215
11.7. Radzenie sobie z nowymi funkcjami	215
11.7.1. Odgałęzienia kodu funkcji	215
11.7.2. Przełączniki funkcji	216
11.7.3. Korzyści	216
11.7.4. Przykłady zastosowań przełączników funkcji	217
11.7.5. Wady	217
11.8. Wnioski	218

Rozdział 12. Wnioski — jakie korzyści wynikają z ciągłego dostarczania?	219
--	------------

Skorowidz	221
------------------	------------