

Spis treści

Podziękowania

O autorze

Wprowadzenie

W.1. Przegląd ciągłego dostarczania i książki

W.2. Po co stosować ciągłe dostarczanie?

W.2.1. Krótka historia

W.2.2. Ciągłe dostarczanie jest pomocne

W.3. Dla kogo przeznaczona jest ta książka?

W.4. Przegląd rozdziałów

W.5. Sposoby czytania książki

Część I Podstawy

Rozdział 1 Ciągłe dostarczanie co i jak

1.1. Wprowadzenie czym jest ciągłe dostarczanie?

1.2. Dlaczego udostępnianie oprogramowania jest tak skomplikowane?

1.2.1. Ciągła integracja daje nadzieję

1.2.2. Powolne i ryzykowne procesy

1.2.3. Szybka praca jest możliwa

1.3. Wartość ciągłego dostarczania

1.3.1. Regularność

1.3.2. Możliwość śledzenia i sprawdzalność zmian

1.3.3. Regresja

1.4. Korzyści płynące z ciągłego dostarczania

1.4.1. Ciągłe dostarczanie w celu przyspieszenia udostępniania

1.4.2. Przykład

1.4.3. Implementowanie funkcji i udostępnianie jej w środowisku produkcyjnym

1.4.4. Przejście do następnej funkcji

1.4.5. Ciągłe dostarczanie zapewnia przewagę konkurencyjną

1.4.6. Scenariusz bez ciągłego dostarczania

1.4.7. Ciągłe dostarczanie i Lean Startup

1.4.8. Wpływ na proces rozwoju produktu

1.4.9. Minimalizowanie ryzyka za pomocą ciągłego dostarczania

1.4.10. Szybsze informacje zwrotne i podejście Lean

1.5. Procesy generowania i struktura potoku ciągłego dostarczania

1.5.1. Przykład

1.6. Wnioski

Rozdział 2 Zapewnianie infrastruktury

2.1. Wprowadzenie

2.1.1. Automatyzowanie infrastruktury przykład

2.2. Skrypty instalacyjne

2.2.1. Problemy związane ze standardowymi skryptami instalacyjnymi

2.3. Chef

2.3.1. Chef a Puppet

2.3.2. Inne możliwości

2.3.3. Podstawy techniczne

Podstawowe pojęcia związane z Chefem

Chef, książki receptur i receptury

Role

2.3.4. Chef Solo

2.3.5. Chef Solo wnioski

2.3.6. Knife i Chef Server

2.3.7. Chef Server wnioski

2.4. Vagrant

2.4.1. Przykład zastosowania Chefa i Vagranta

2.4.2. Vagrant wnioski

2.5. Docker

2.5.1. Rozwiązanie oparte na Dockerze

Kontenery Dockera a wirtualizacja

Cel używania Dockera

Komunikacja między kontenerami Dockera

2.5.2. Tworzenie kontenerów Dockera

Pliki Dockerfile

Tworzenie i uruchamianie obrazów Dockera

2.5.3. Uruchamianie przykładowej aplikacji za pomocą Dockera

Dodatkowe polecenia Dockera

2.5.4. Docker i Vagrant

Vagrant jako mechanizm dodawania oprogramowania

2.5.5. Docker Machine

2.5.6. Tworzenie złożonych konfiguracji za pomocą Dockera

Docker Registry

Docker w klastrze

2.5.7. Docker Compose

2.6. Niemodyfikowalny serwer

2.6.1. Wady idempotencji

2.6.2. Serwer niemodyfikowalny i Docker

2.7. Infrastruktura jako kod

2.7.1. Testowanie infrastruktury w postaci kodu

Serverspec

Test Kitchen

ChefSpec

2.8. Platforma jako usługa

2.9. Obsługa danych i baz

2.9.1. Zarządzanie schematami

2.9.2. Dane testowe i główne

2.10. Wnioski

Część II Potok ciągłego dostarczania

Rozdział 3 Automatyzacja procesu budowania i ciągła integracja

3.1. Wprowadzenie

3.1.1. Automatyzacja procesu budowania przykład

3.2. Automatyzowanie procesu budowania i narzędzia do budowania

3.2.1. Narzędzia do budowania w świecie Javy

3.2.2. Ant

3.2.3. Maven

Kontrola wersji i snapshoty

Udostępnianie z użyciem Mavena

3.2.4. Gradle

Gradle Wrapper

3.2.5. Dodatkowe narzędzia do budowania

3.2.6. Wybór odpowiedniego narzędzia

3.3. Testy jednostkowe

3.3.1. Pisanie dobrych testów jednostkowych

3.3.2. Programowanie sterowane testami

3.3.3. Ruchy Clean Code i Software Craftsmanship

3.4. Ciągła integracja

3.4.1. Jenkins

Rozszerzanie Jenkinsa za pomocą wtyczek

Wtyczka SCM Sync Configuration

Wtyczka Environment Injector

Wtyczka Job Configuration History

Wtyczka Clone Workspace SCM

Wtyczka Build Pipeline

Wtyczka Amazon EC2

Wtyczka Job DSL

Tworzenie własnych wtyczek

3.4.2. Infrastruktura do ciągłej integracji

3.4.3. Wnioski

3.5. Pomiar jakości kodu

3.5.1. SonarQube

Integrowanie z potokiem

3.6. Zarządzanie artefaktami

3.6.1. Integracja z procesem budowania

3.6.2. Zaawansowane funkcje repozytoriów

3.7. Wnioski

Rozdział 4 Testy akceptacyjne

4.1. Wprowadzenie

4.1.1. Testy akceptacyjne przykład

4.2. Piramida testów

4.3. Czym są testy akceptacyjne?

4.3.1. Zautomatyzowane testy akceptacyjne

4.3.2. Więcej niż wzrost wydajności

4.3.3. Testy ręczne

4.3.4. A co z klientem?

4.3.5. Testy akceptacyjne a testy jednostkowe

4.3.6. Środowiska testowe

4.4. Testy akceptacyjne oparte na interfejsie GUI

4.4.1. Problemy z testami z użyciem interfejsu GUI

4.4.2. Stosowanie abstrakcji

4.4.3. Automatyzacja z użyciem narzędzia Selenium

4.4.4. Interfejs API WebDriver

4.4.5. Testy bez użycia przeglądarki HtmlUnit

4.4.6. Interfejs API WebDriver dla Selenium

4.4.7. Środowisko IDE dla Selenium

4.4.8. Problemy ze zautomatyzowanymi testami interfejsu GUI

4.4.9. Wykonywanie testów z użyciem interfejsu GUI

4.4.10. Eksportowanie testów jako kodu

4.4.11. Ręczne modyfikowanie przypadków testowych

4.4.12. Dane testowe

4.4.13. Obiekty reprezentujące strony

4.5. Inne narzędzia do przeprowadzania testów z użyciem interfejsu GUI

4.5.1. PhantomJS

4.5.2. Windmill

4.6. Tekstowe testy akceptacyjne

4.6.1. Podejście BDD

4.6.2. Różne adaptory

4.7. Inne platformy

4.8. Strategie przeprowadzania testów akceptacyjnych

4.8.1. Właściwe narzędzie

4.8.2. Błyskawiczne informacje zwrotne

4.8.3. Pokrycie testami

4.9. Wnioski

Rozdział 5 Testy wydajności

5.1. Wprowadzenie

5.1.1. Testy wydajności przykład

5.2. Testy wydajności jak je przeprowadzać?

5.2.1. Cele testów wydajności

5.2.2. Środowiska i ilość danych

5.2.3. Testy szybkości tylko po zakończeniu implementowania kodu?

5.2.4. Testy wydajności = zarządzanie ryzykiem

5.2.5. Symulowanie działań użytkowników

5.2.6. Dokumentowanie wymogów związanych z szybkością

5.2.7. Sprzęt potrzebny w testach wydajności

5.2.8. Chmura i wirtualizacja

5.2.9. Minimalizowanie ryzyka dzięki ciągłemu testowaniu

5.2.10. Testy wydajności sensowne czy nie?

5.3. Implementowanie testów wydajności

5.4. Testy wydajności z użyciem narzędzia Gatling

5.4.1. Wersja demonstracyjna a praktyka

5.5. Narzędzia używane zamiast Gatlinga

5.5.1. Grinder

5.5.2. Apache JMeter

5.5.3. Tsung

5.5.4. Rozwiązania komercyjne

5.6. Wnioski

Rozdział 6 Testy eksploracyjne

6.1. Wprowadzenie

6.1.1. Testy eksploracyjne przykład

6.2. Po co stosować testy eksploracyjne?

6.2.1. Czasem testy ręczne i tak są lepsze

6.2.2. Testy z udziałem klientów

6.2.3. Testy ręczne wymagań нефunkcjonalnych

6.3. Jak zabrać się za testy eksploracyjne?

6.3.1. Zadanie określa testy

6.3.2. Zautomatyzowane środowisko

6.3.3. Przypadki pokazowe jako punkt wyjścia

6.3.4. Przykład aplikacja z obszaru handlu elektronicznego

6.3.5. Testy wersji beta

6.3.6. Testy oparte na sesji

6.4. Wnioski

Rozdział 7 Wdrażanie udostępnianie w środowisku produkcyjnym

7.1. Wprowadzenie

7.1.1. Wdrażanie przykład

7.2. Udostępnianie i wycofywanie

7.2.1. Korzyści

7.2.2. Wady

7.3. Zastępowanie nową wersją

7.3.1. Korzyści

7.3.2. Wady

7.4. Wdrażanie w modelu niebieskie-zielone

7.4.1. Korzyści

7.4.2. Wady

7.5. Udostępnianie kanarkowe

7.5.1. Korzyści

7.5.2. Wady

7.6. Ciągłe wdrażanie

7.6.1. Korzyści

7.6.2. Wady

7.7. Wirtualizacja

7.7.1. Hosty fizyczne

7.8. Poza aplikacje sieciowe

7.9. Wnioski

Rozdział 8 Eksploatacja

8.1. Wprowadzenie

8.1.1. Eksploatacja przykład

8.2. Trudności z eksploatacją oprogramowania

8.3. Pliki dziennika

8.3.1. Co należy rejestrować?

8.3.2. Narzędzia do przetwarzania plików dziennika

ELK: Elasticsearch, Logstash i Kibana

8.3.3. Rejestrowanie danych w przykładowej aplikacji

8.4. Analizowanie dzienników przykładowej aplikacji

8.4.1. Analizowanie danych za pomocą Kibany

8.4.2. ELK skalowalność

8.5. Inne technologie obsługi dzienników

8.6. Zaawansowane techniki rejestrowania dzienników

8.6.1. Anonimizacja

8.6.2. Szybkość działania

8.6.3. Czas

8.6.4. Operacyjna baza danych

8.7. Monitorowanie

8.8. Pomiary z użyciem narzędzia Graphite

8.9. Pomiary w przykładowej aplikacji

8.9.1. Struktura przykładu

8.10. Inne rozwiązania z obszaru monitorowania

8.11. Inne wyzwania związane z eksploatacją aplikacji

8.11.1. Skrypty

8.11.2. Aplikacje w centrum danych klienta

8.12. Wnioski

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

Rozdział 9 Wprowadzanie ciągłego dostarczania w organizacji

9.1. Wprowadzenie

9.2. Ciągłe dostarczanie od początku projektu

9.3. Odzworowywanie strumienia wartości

9.3.1. Odzworowywanie strumienia wartości pozwala opisać sekwencję zdarzeń

9.3.2. Optymalizacje

9.4. Dodatkowe sposoby optymalizacji

9.4.1. Inwestycje wysokiej jakości

9.4.2. Koszty

9.4.3. Korzyści

9.4.4. Nie dodawaj kodu, gdy proces budowania kończy się niepowodzeniem!

9.4.5. Zatrzymywanie taśmy

9.4.6. Pięć pytań dlaczego

9.4.7. DevOps

9.5. Wnioski

Rozdział 10 Ciągłe dostarczanie i DevOps

10.1. Wprowadzenie

10.2. Czym jest model DevOps?

10.2.1. Problemy

10.2.2. Perspektywa klienta

10.2.3. Pionierska firma Amazon

10.2.4. DevOps

10.3. Ciągłe dostarczanie i DevOps

10.3.1. DevOps więcej niż ciągłe dostarczanie

10.3.2. Pełna odpowiedzialność i samoorganizowanie się

10.3.3. Decyzje z obszaru technologii

10.3.4. Mniej scentralizowanej kontroli

10.3.5. Pluralizm w obszarze technologii

10.3.6. Wymiana między zespołami

10.3.7. Architektura

10.4. Ciągłe dostarczanie bez modelu DevOps?

10.4.1. Końcowa część potoku ciągłego dostarczania

10.5. Wnioski

Rozdział 11 Ciągłe dostarczanie, DevOps i architektura oprogramowania

11.1. Wprowadzenie

11.2. Architektura oprogramowania

11.2.1. Po co tworzyć architekturę oprogramowania?

11.3. Optymalizowanie architektury pod kątem ciągłego dostarczania

11.3.1. Mniejsze jednostki wdrażania

11.4. Interfejsy

11.4.1. Prawo Postela lub zasada odporności

11.4.2. Projektowanie pod kątem niepowodzeń

11.4.3. Stan

11.5. Bazy danych

11.5.1. Zapewnianie stabilności baz danych

11.5.2. Baza danych = komponent

11.5.3. Widoki i procedury składowane

11.5.4. Baza danych dla komponentu

11.5.5. Bazy typu NoSQL

11.6. Mikrousługi

11.6.1. Mikrousługi i ciągłe dostarczanie

11.6.2. Wprowadzanie ciągłego dostarczania z użyciem mikrousług

11.6.3. Mikrousługi wymagają ciągłego dostarczania

11.6.4. Struktura organizacyjna

11.7. Radzenie sobie z nowymi funkcjami

11.7.1. Odgałęzienia kodu funkcji

11.7.2. Przełączniki funkcji

11.7.3. Korzyści

11.7.4. Przykłady zastosowań przełączników funkcji

11.7.5. Wady

11.8. Wnioski

Rozdział 12 Wnioski jakie korzyści wynikają z ciągłego dostarczania?