Spis treści

O autorach 15

O recenzentach 16

Przedmowa 17

CZĘŚĆ 1. Rozpoczęcie pracy z DevOps, Gitem i GitLabem

ROZDZIAŁ 1
Zrozumienie okresu przed DevOps 23

Wprowadzenie do aplikacji internetowej Hats for Cats 23
Ręczne tworzenie i weryfikacja kodu 24
Ręczne budowanie kodu 24
Ręczna weryfikacja kodu 25
Dodatkowe wyzwania związane z weryfikacją kodu 31
Ręczne przeprowadzanie testów bezpieczeństwa kodu 32
Statyczna analiza kodu 33
Wykrywanie tajemnic 33
Analiza dynamiczna 34
Skanowanie zależności 34
Skanowanie kontenerów 35
Podsumowanie ręcznych testów bezpieczeństwa 35
Ręczne pakowanie i wdrażanie kodu 36
Skanowanie zgodności licencji 37
Wdrażanie oprogramowania 37
Problemy z ręcznymi praktykami w cyklu życia oprogramowania 39
Rozwiązywanie problemów za pomocą DevOps 41
Jak GitLab implementuje DevOps 42
Podsumowanie 43

ROZDZIAŁ 2
Ćwiczenie podstawowych poleceń Gita 45
Wymagania techniczne 46
Dlaczego korzystać z Gita? 47
Czym jest system kontroli wersji? 47
Jakie problemy rozwiązuje system kontroli wersji? 48
Dlaczego Git jest popularny 50
Wady Gita 52
Zatwierdzanie kodu, aby zachować go w bezpiecznym miejscu 53
Wyłączenie plików z repozytorium 57
Oznaczanie zatwierdzeń w celu identyfikowania wersji kodu 59
Tworzenie gałęzi, aby rozwijać kod w oddzielnym miejscu 60
Komendy Gita do zarządzania gałęziami 63
Obsługa konfliktów scalania 63
Synchronizacja lokalnych i zdalnych kopii repozytoriów 66
„Złote" repozytorium 66
Konfigurowanie zdalnych repozytoriów 67
Wypychanie zatwierdzeń 70
Pobieranie fetch 70
Pobieranie pull 71
Dodatkowe źródła do nauki Gita 72
Podsumowanie 73

ROZDZIAŁ 3
Zrozumienie komponentów GitLaba 74
Wymagania techniczne 75
Kładziemy nacisk na „dlaczego" bardziej niż na „jak" 75
Wprowadzenie do platformy GitLaba 76
Czym jest GitLab? 76
Jaki problem rozwiązuje GitLab? 76
Etapy weryfikacji, zabezpieczania i wydawania 78
Organizowanie pracy w projekty i grupy 79
Przykład — organizacja pracy nad projektem Hats for Cats 81
Śledzenie pracy za pomocą zgłoszeń 83
Struktura zgłoszenia w GitLabie 83
Rodzaje zadań, które mogą być reprezentowane przez zgłoszenia 85
Etykiety 86
Schematy pracy ze zgłoszeniami 87

Bezpieczne edytowanie plików za pomocą zatwierdzeń, gałęzi i próśb o scalenie ... 88
Historia zatwierdzeń ... 91
Łączenie jednej gałęzi Gita z drugą ... 92
Trzej amigos — zgłoszenia, gałęzie i prośby o scalenie ... 97
Kiedy dwóch amigos wystarcza ... 97
Czym różnią się zgłoszenia i prośby o scalenie? ... 98
Korzystanie z praktyk DevOps za pomocą GitLab Flow ... 99
Podsumowanie ... 101

ROZDZIAŁ 4
Opis struktury potoku CI/CD GitLaba ... 103
Wymagania techniczne ... 103
Definicje pojęć: „potok", „CI" i „CD" ... 104
Czym jest potok ... 104
Definiowanie jednego potoku na projekt ... 105
Wyjaśnienie różnych znaczeń terminu „potok" ... 105
Przeglądanie listy potoków ... 105
CI — dowiedz się, czy Twój kod jest dobry ... 107
CD — dowiedz się, gdzie powinien trafić Twój kod (i umieść go tam) ... 108
GitLab Runnery ... 111
Elementy potoku — etapy, zadania i polecenia ... 112
Etapy ... 112
Zadania ... 114
Polecenia ... 115
Łączenie elementów potoku ... 116
Uruchamianie potoków CI/CD GitLaba ... 116
Potoki dla gałęzi (ang. branch pipelines) ... 116
Potoki dla tagów Gita ... 118
Inne rodzaje potoków ... 118
Pomijanie potoków ... 119
Odczytywanie statusów potoków CI/CD GitLaba ... 120
Konfigurowanie potoków CI/CD GitLaba ... 121
Podsumowanie ... 125

CZĘŚĆ 2. Automatyzacja etapów DevOps przy użyciu potoków CI/CD GitLaba

ROZDZIAŁ 5
Instalacja i konfiguracja GitLab Runnerów 129
Wymagania techniczne 129
Definicja GitLab Runnerów i ich związek z CI/CD 130
GitLab Runner to aplikacja open source napisana w języku Go 130
GitLab Runner uruchamia zadania CI/CD określone w pliku .gitlab-ci.yml 131
Architektura runnera i obsługiwane platformy 131
GitLab Runner jest obsługiwany przez większość platform i architektur 133
Runnery mogą być specyficzne dla projektu, grupy lub współdzielone 134
Każdy runner ma zdefiniowanego executora 136
Tagi runnera określają, które runnery mogą wykonywać konkretne zadania 140
Instalacja agenta runnera 141
Instalacja GitLab Runnera 141
Rejestracja runnera w GitLabie 142
Rozważania dotyczące różnych typów runnerów i executorów 147
Rozważania dotyczące wydajności 147
Rozważania dotyczące bezpieczeństwa 149
Rozważania dotyczące monitorowania 150
Podsumowanie 152

ROZDZIAŁ 6
Weryfikacja kodu 153
Wymagania techniczne 153
Budowanie kodu w potoku CI/CD 154
Kompilacja kodu Java za pomocą javac 154
Kompilacja Javy przy użyciu narzędzia Maven 157
Kompilacja języka C przy użyciu narzędzia GNU Compiler Collection (GCC) 158
Przechowywanie skompilowanego kodu jako artefaktów 160
Sprawdzanie jakości kodu w potoku CI/CD 161
Włączanie funkcji jakości kodu 161
Przeglądanie wyników funkcji jakości kodu 162

Uruchamianie automatycznych testów funkcjonalnych na etapie dostarczania (CI/CD) ... 163
Włączanie automatycznych testów funkcjonalnych ... 164
Przeglądanie wyników automatycznych testów funkcjonalnych ... 165
Testy fuzzingowe w potoku CI/CD ... 167
Architektura i przepływ pracy testowania fuzzingowego ... 168
Przepływ pracy testowania fuzzingowego ... 171
Przeglądanie wyników testów fuzzingu ... 172
Dodatkowe uwagi dotyczące testowania fuzzingu ... 172
Testowanie fuzzingu z korpusem ... 174
Sprawdzanie dostępności w procesie CI/CD ... 174
Dodawanie testów dostępności ... 175
Przeglądanie wyników testów dostępności ... 175
Dodatkowe sposoby weryfikacji kodu ... 177
Pokrycie kodu (ang. code coverage) ... 177
Testowanie wydajności przeglądarki (ang. browser performance testing) ... 177
Testowanie wydajności obciążeniowej (ang. load performance testing) ... 178
Podsumowanie ... 178

ROZDZIAŁ 7
Zabezpieczanie kodu ... 180

Wymagania techniczne ... 181
Zrozumienie strategii skanowania bezpieczeństwa GitLaba ... 182
GitLab korzysta ze skanerów open source ... 182
Skanery są dostarczane jako obrazy Dockera ... 183
Niektóre skanery używają różnych analizatorów dla różnych języków programowania ... 184
Podatności nie zatrzymują potoku ... 185
Wyniki pojawiają się w trzech różnych raportach ... 185
Potoki mogą korzystać ze skanerów innych niż GitLab ... 186
Korzystanie z SAST-a do skanowania kodu źródłowego pod kątem podatności ... 186
Zrozumienie SAST-a ... 186
Włączanie SAST-a ... 187
Konfigurowanie SAST-a ... 189
Przeglądanie wyników SAST-a ... 190
Użycie wykrywania sekretów do znalezienia poufnych informacji w Twoim repozytorium ... 191
Zrozumienie działania wykrywania sekretów ... 191

Włączanie i konfigurowanie wykrywania sekretów ... 193
Przeglądanie wyników wykrywania sekretów ... 194
Korzystanie z DAST-a do wykrywania podatności w aplikacjach internetowych ... 194
Zrozumienie działania DAST-a ... 195
Włączanie i konfigurowanie DAST-a ... 195
Przegląd wyników DAST-a ... 197
Korzystanie ze skanowania zależności do wyszukiwania luk w zależnościach ... 197
Zrozumienie skanowania zależności ... 198
Włączanie i konfigurowanie skanowania zależności ... 199
Przeglądanie wyników skanowania zależności ... 200
Korzystanie ze skanowania kontenerów do wyszukiwania podatności w obrazach Dockera ... 200
Zrozumienie skanowania zależności ... 201
Włączanie i konfigurowanie skanowania kontenerów ... 202
Przeglądanie wyników skanowania kontenerów ... 202
Korzystanie z badania zgodności licencji do zarządzania licencjami zależności ... 203
Zrozumienie badania zgodności licencji ... 203
Włączanie i konfigurowanie badania zgodności z licencją ... 206
Przeglądanie wyników badania zgodności z licencją ... 207
Korzystanie ze skanowania IaC do wykrywania problemów w plikach konfiguracyjnych infrastruktury ... 208
Zrozumienie skanowania IaC ... 208
Włączanie i konfigurowanie skanowania IaC ... 209
Przeglądanie wyników skanowania IaC ... 209
Zrozumienie różnych rodzajów raportów bezpieczeństwa ... 210
Zarządzanie podatnościami związanymi z bezpieczeństwem ... 211
Integracja z zewnętrznymi skanerami bezpieczeństwa ... 213
Podsumowanie ... 214

ROZDZIAŁ 8
Pakowanie i wdrażanie kodu ... 216

Wymagania techniczne ... 217
Przechowywanie kodu w rejestrze pakietów GitLaba w celu późniejszego wykorzystania ... 217
Lokalizacja rejestrów kontenerów i pakietów GitLaba ... 217
Rozpoczęcie pracy z rejestrem pakietów ... 219
Obsługiwane formaty pakietów ... 220
Uwierzytelnianie w rejestrze ... 221

Budowanie i publikowanie pakietów w rejestrze pakietów 224
Budowanie i przesyłanie pakietów do rejestru kontenerów 226
Przechowywanie kodu w rejestrach kontenerów i pakietów GitLaba w celu późniejszego wdrożenia 230
Korzystanie z obrazów z rejestru kontenerów 230
Wykorzystanie pakietów z rejestru pakietów 231
Wdrażanie w różnych środowiskach przy użyciu GitLab Flow 233
Wdrażanie w narzędziu review app w celu testowania 234
Wdrażanie w rzeczywistych środowiskach produkcyjnych 236
Wdrażanie w klastrze Kubernetes 238
Proces CI/CD 238
Podejście GitOps 239
Podsumowanie 240

CZĘŚĆ 3. Następne kroki w doskonaleniu aplikacji za pomocą GitLaba

ROZDZIAŁ 9
Poprawa szybkości i łatwości utrzymania potoku CI/CD 243
Przyspieszanie procesów za pomocą skierowanych grafów acyklicznych i architektury rodzic – dziecko 243
Jak utworzyć DAG w potoku CI? 244
Budowanie kodu dla wielu architektur 245
Kiedy i jak wykorzystywać pamięć podręczną lub artefakty? 246
Charakterystyka pamięci podręcznej 247
Charakterystyka artefaktów 247
Korzystanie z pamięci podręcznej 248
Korzystanie z artefaktów 249
Wykorzystywanie artefaktów jako zależności zadania 250
Redukowanie powtarzającego się kodu konfiguracyjnego za pomocą zakotwiczeń i słowa kluczowego extends 251
Zakotwiczenia 251
Słowo kluczowe extends: 252
Tagi referencji 254
Poprawa zarządzalności poprzez łączenie wielu potoków oraz wykorzystywanie potoków macierzystych i potomnych 254
Łączenie plików dla ułatwienia zarządzania 255
Użycie opcji include: w celu uzyskania możliwości ponownego wykorzystania 256
Dołączanie zdalnych zasobów 257
Wykorzystywanie potoków macierzystych i potomnych 258

Zabezpieczanie i przyspieszanie zadań za pomocą kontenerów utworzonych w celu realizacji określonych zadań ... 258
Przykład kontenera utworzonego w celu realizacji określonego zadania ... 260
Podsumowanie ... 261

ROZDZIAŁ 10
Poszerzanie zakresu potoków CI/CD ... 262

Wykorzystywanie potoków CI/CD do wykrywania problemów wydajnościowych ... 262
Jak zintegrować przeglądarkowe testy wydajnościowe? ... 263
Jak zintegrować testy obciążeniowe z użyciem narzędzia k6? ... 264
Korzystanie z flag funkcji umożliwiających wydawanie różnych aplikacji w zależności od decyzji biznesowych ... 265
Jak skonfigurować aplikację pod kątem flag funkcji? ... 267
Integracja narzędzi innych firm z potokami CI/CD ... 268
Tworzenie pliku Dockerfile dla kontenera narzędziowego ... 268
Automatyzacja procesu budowy kontenera ... 269
Skanowanie kontenerów ... 270
Wywoływanie narzędzia zewnętrznego ... 270
Wykorzystywanie potoków CI/CD do tworzenia aplikacji mobilnych ... 270
Wymagania ... 271
Fastlane ... 271
Fastlane — wdrożenie ... 272
Fastlane — automatyzacja testowania ... 272
Podsumowanie ... 273

ROZDZIAŁ 11
Kompletny przykład ... 274

Wymagania techniczne ... 274
Konfiguracja środowiska ... 274
Tworzenie projektu w GitLabie ... 275
Planowanie pracy za pomocą zgłoszeń GitLaba ... 276
Konfiguracja lokalnego repozytorium Gita ... 278
Tworzenie kodu ... 278
Tworzenie gałęzi Gita ... 279
Tworzenie żądania MR ... 279
Zatwierdzanie i przesyłanie kodu ... 279
Tworzenie infrastruktury potoku ... 280
Tworzenie potoku ... 280
Tworzenie runnera ... 281

Weryfikacja kodu ... 284
Dodawanie testów funkcjonalnych do potoku ... 284
Dodawanie skanowania jakości kodu do potoku ... 286
Dodawanie testu typu fuzzing do potoku ... 288
Zabezpieczanie kodu ... 290
Dodawanie SAST-a do potoku ... 290
Dodawanie wykrywania sekretów do potoku ... 291
Dodawanie skanowania zależności do potoku ... 292
Dodawanie badania zgodności licencji do potoku ... 293
Integracja zewnętrznego skanera bezpieczeństwa z potokiem ... 294
Doskonalenie potoku ... 295
Korzystanie z DAG-a w celu przyspieszenia potoku ... 295
Podział potoku na kilka plików ... 296
Dostarczanie kodu do odpowiedniego środowiska ... 298
Wdrażanie kodu ... 298
Podsumowanie ... 299

ROZDZIAŁ 12
Rozwiązywanie problemów i przyszłość GitLaba ... 301
Wymagania techniczne ... 301
Rozwiązywanie problemów i najlepsze praktyki dotyczące powszechnych problemów spotykanych w potokach CI/CD ... 302
Rozwiązywanie problemów związanych ze składnią i logiką CI/CD ... 302
Rozwiązywanie problemów z działaniem potoku i przypisaniem runnera ... 307
Zarządzanie infrastrukturą operacyjną przy użyciu GitOpsa ... 309
Użycie Terraforma do wdrażania i aktualizowania stanu infrastruktury ... 310
Użycie Ansible'a do zarządzania konfiguracjami zasobów ... 311
Przyszłe trendy ... 312
Automatyzacja stworzy więcej oprogramowania na większą skalę ... 313
Abstrakcja prowadzi do modeli biznesowych opartych na pojęciu „wszystko jako kod" ... 313
Skrócony czas cyklu rozwoju produktu pomoże zespołom wydawać lepsze oprogramowanie szybciej ... 314
Podsumowanie i kolejne kroki ... 315