

spis treści

przedmowa xix

podziękowania xxi

o książce xxiii

o autorze xxix

o ilustracji na okładce xxxi

CZĘŚĆ I PRZEGLĄD 1

1 Wprowadzenie do Kubernetes 3

1.1. Zapotrzebowanie na system taki jak Kubernetes 5

Przechodzenie od aplikacji monolitycznych do mikroserwisów 5 ■ Zapewnianie spójnego środowiska dla aplikacji 8 ■ Przechodzenie do ciągłego dostarczania: DevOps oraz NoOps 9

1.2. Przedstawiamy technologie kontenerowe 10

Czym są kontenery 11 ■ Platforma kontenerów Docker 15

1.3. Wprowadzenie do Kubernetes 19

Pochodzenie 19 ■ Ogólny obraz Kubernetes 20 ■ Architektura klastra Kubernetes 21 ■ Uruchamianie aplikacji w Kubernetes 23 ■ Korzyści z używania Kubernetes 25

1.4. Podsumowanie 28

2 Pierwsze kroki z Dockerem i Kubernetes 29

- 2.1 Tworzenie, uruchamianie i udostępnianie obrazu kontenera 30
 - Instalowanie Dockera i uruchamianie powitalnego kontenera 30
 - Tworzenie trywialnej aplikacji Node.js 32
 - Tworzenie Dockerfile dla obrazu 33
 - Budowanie obrazu kontenera 34
 - Uruchamianie obrazu kontenera 36
 - Badanie wnętrza uruchomionego kontenera 38
 - Zatrzymywanie i usuwanie kontenera 40
 - Wypychanie obrazu do rejestru obrazów 40
- 2.2 Konfigurowanie klastra Kubernetes 41
 - Uruchamianie lokalnego jednowęzłowego klastra Kubernetes przy użyciu Minikube 42
 - Korzystanie z hostowanego klastra Kubernetes przy użyciu Google Kubernetes Engine 44
 - Konfigurowanie aliasu i dopełniania poleceń dla kubectl 47
- 2.3 Uruchamianie naszej pierwszej aplikacji w Kubernetes 48
 - Wdrażanie naszej aplikacji Node.js 48
 - Uzyskiwanie dostępu do aplikacji 52
 - Logiczne części naszego systemu 54
 - Skalowanie aplikacji w poziomie 55
 - Badanie, na których węzłach działa aplikacja 58
 - Przedstawiamy tablicę kontrolną Kubernetes 59
- 2.4 Podsumowanie 61

CZĘŚĆ II PODSTAWOWE KONCEPCJE 63**3 Pody: uruchamianie kontenerów w Kubernetes 65**

- 3.1 Przedstawiamy pody 66
 - Dlaczego potrzebujemy podów 66
 - Istota podów 67
 - Właściwe porządkowanie kontenerów między pody 69
- 3.2 Tworzenie podów na podstawie deskryptorów YAML lub JSON 71
 - Badanie deskryptora YAML istniejącego podu 72
 - Tworzenie prostego deskryptora YAML dla podu 74
 - Wykorzystanie kubectl create do utworzenia podu 76
 - Przeglądanie dzienników aplikacji 76
 - Wysyłanie żądań do podu 78
- 3.3 Porządkowanie podów przy użyciu etykiet 79
 - Przedstawiamy etykiety 80
 - Specyfikowanie etykiet przy tworzeniu podu 81
 - Modyfikowanie etykiet istniejących podów 82
- 3.4 Wyliczanie podzbiorów podów przy użyciu selektorów etykiet 82
 - Wyliczanie podów przy użyciu selektora etykiet 83
 - Stosowanie wielu warunków w selektorze etykiet 84

- 3.5 Używanie etykiet i selektorów do ograniczania rozmieszczania podów 85
 - Używanie etykiet do kategoryzowania węzłów roboczych 85
 - Przypisywanie podów do określonych węzłów 86
 - Umieszczanie podu na jednym, określonym węźle 86
 - 3.6 Opisywanie podów 87
 - Wyszukiwanie adnotacji obiektów 87
 - Dodawanie i modyfikowanie adnotacji 88
 - 3.7 Wykorzystanie przestrzeni nazw do grupowania zasobów 89
 - Dlaczego przestrzenie nazw są potrzebne 89
 - Odkrywanie innych przestrzeni nazw i ich podów 89
 - Tworzenie przestrzeni nazw 90
 - Zarządzanie obiektami w innych przestrzeniach nazw 91
 - Izolacja zapewniana przez przestrzenie nazw 92
 - 3.8 Zatrzymywanie i usuwanie podów 92
 - Usuwanie podu według nazwy 92
 - Usuwanie podów przy użyciu selektorów etykiet 93
 - Usuwanie podów przez usunięcie całej przestrzeni nazw 94
 - Usuwanie wszystkich podów w przestrzeni nazw, ale z zachowaniem tej przestrzeni nazw 94
 - Usuwanie (niemal) wszystkich zasobów w przestrzeni nazw 95
 - 3.9 Podsumowanie 95
- 4 Replikacja i inne kontrolery: wdrażanie zarządzanych podów 97**
- 4.1 Utrzymywanie dobrej kondycji podów 98
 - Wprowadzenie do sond żywotności 99
 - Tworzenie sondy żywotności opartej na HTTP 99
 - Obserwowanie sondy żywotności w działaniu 100
 - Konfigurowanie dodatkowych właściwości sondy żywotności 101
 - Tworzenie skutecznych sond żywotności 103
 - 4.2 Przedstawiamy kontrolery replikacji 104
 - Działanie kontrolera replikacji 105
 - Tworzenie kontrolera replikacji 107
 - Obserwowanie kontrolera replikacji w działaniu 109
 - Przenoszenie podów do zakresu i poza zakres kontrolera replikacji 113
 - Zmianianie szablonu podu 116
 - Skalowanie podów w poziomie 117
 - Usuwanie kontrolera replikacji 119
 - 4.3 Używanie obiektów ReplicaSet zamiast ReplicationController 120
 - Porównanie ReplicaSet z ReplicationController 120
 - Definiowanie ReplicaSet 121
 - Tworzenie i badanie obiektu ReplicaSet 122
 - Korzystanie z bardziej ekspresyjnych selektorów etykiet w ReplicaSet 123
 - Podsumowanie ReplicaSet 123
 - 4.4 Uruchamianie dokładnie jednego podu w każdym węźle przy użyciu DaemonSet 124
 - Wykorzystanie DaemonSet do uruchomienia podu w każdym węźle 125
 - Używanie DaemonSet do uruchamiania podów tylko na wybranych węzłach 125

- 4.5 Uruchamianie podów, które wykonują pojedyncze, kompletne zadanie 128
 - Przedstawiamy zasób Job 129
 - Definiowanie zasobu Job 130
 - Obserwowanie uruchamiania podu przez obiekt Job 130
 - Uruchamianie wielu wystąpień podu w zasobie Job 131
 - Ograniczanie czasu, w którym pod obiektu Job musi ukończyć pracę 133
 - 4.6 Planowanie okresowego wykonywania zadania lub jeden raz w przyszłości 133
 - Tworzenie obiektu CronJob 134
 - Jak uruchamiane są zaplanowane zadania 135
 - 4.7 Podsumowanie 136
- 5 Usługi: umożliwianie odkrywania i komunikowania się klientów z podami 137**
- 5.1 Wprowadzenie do usług 138
 - Tworzenie usług 139
 - Odkrywanie usług 146
 - 5.2 Łączenie się z usługami działającymi poza klastrem 150
 - Wprowadzenie do punktów końcowych usług 150
 - Ręczne konfigurowanie punktów końcowych usług 151
 - Tworzenie aliasu dla zewnętrznej usługi 152
 - 5.3 Eksponowanie usług klientom zewnętrznym 153
 - Korzystanie z usługi typu NodePort 154
 - Eksponowanie usługi przez zewnętrzny moduł równoważenia obciążeń 158
 - Osobliwości połączeń zewnętrznych 160
 - 5.4 Zewnętrzne eksponowanie usług przy użyciu zasobu Ingress 162
 - Tworzenie zasobu Ingress 164
 - Uzyskiwanie dostępu do usługi za pośrednictwem Ingress 164
 - Eksponowanie wielu usług przez ten sam obiekt Ingress 166
 - Konfigurowanie obsługi ruchu TLS przez Ingress 167
 - 5.5 Sygnalizowanie, kiedy pod jest gotowy do akceptowania połączeń 169
 - Przedstawiamy sondy gotowości 170
 - Dodawanie sondy gotowości do podu 171
 - Co powinny robić rzeczywiste sondy gotowości 174
 - 5.6 Używanie usługi headless w celu odkrywania indywidualnych podów 175
 - Tworzenie usługi headless 175
 - Odkrywanie podów przy użyciu DNS 176
 - Odkrywanie wszystkich podów – łącznie z tymi, które nie są gotowe 177
 - 5.7 Rozwiązywanie problemów z usługami 178
 - 5.8 Podsumowanie 179

- 6 Woluminy: dołączanie do kontenerów dyskowej pamięci masowej 180**
- 6.1 Przedstawiamy woluminy 181
 - Objaśnianie woluminów na przykładzie 181 ■ Przedstawiamy dostępne typy woluminów 184
 - 6.2 Używanie woluminów do współużytkowania danych między kontenerami 185
 - Korzystanie z woluminu emptyDir 185 ■ Wykorzystanie repozytorium Git jako punktu wyjścia dla naszego woluminu 188
 - 6.3 Uzyskiwanie dostępu do plików w systemie plików węzła roboczego 191
 - Przedstawiamy wolumin hostPath 192 ■ Badanie podów systemowych używających woluminów hostPath 192
 - 6.4 Korzystanie z trwałej pamięci masowej 194
 - Korzystanie z GCE Persistent Disk w woluminie podu 194 ■ Korzystanie z innych typów woluminów z leżącą w tle trwałą pamięcią masową 198
 - 6.5 Oddzielanie podów od leżącej w tle technologii pamięci masowej 199
 - Wprowadzenie do obiektów PersistentVolume i PersistentVolumeClaim 200
 - Tworzenie zasobu PersistentVolume 201 ■ Żądanie PersistentVolume przez utworzenie PersistentVolumeClaim 203 ■ Używanie obiektu PersistentVolumeClaim w podzie 205 ■ Zalety korzystania z obiektów PersistentVolume i żądań 206 ■ Recykling obiektów PersistentVolume 207
 - 6.6 Dynamiczne wyposażanie obiektów PersistentVolume 209
 - Definiowanie dostępnych typów pamięci masowej przez zasoby StorageClass 209
 - Żądanie klasy magazynowej w PersistentVolumeClaim 210 ■ Dynamiczne wyposażanie bez specyfikowania klasy magazynowej 212
 - 6.7 Podsumowanie 215
- 7 Obiekty ConfigMap oraz Secret: konfigurowanie aplikacji 217**
- 7.1 Konfigurowanie skontenerowanych aplikacji 218
 - 7.2 Przekazywanie do kontenerów argumentów wiersza polecenia 219
 - Definiowanie polecenia i argumentów w Dockerze 219 ■ Nadpisywanie polecenia i argumentów w Kubernetes 221
 - 7.3 Ustawianie zmiennych środowiskowych dla kontenera 223
 - Specyfikowanie zmiennych środowiskowych w definicji kontenera 224 ■ Odwoływanie się do innych zmiennych środowiskowych w wartości zmiennej 224 ■ Wady kodowania zmiennych środowiskowych na sztywno 225

7.4 Oddzielanie konfiguracji przy użyciu ConfigMap 225

Przedstawiamy obiekty ConfigMap 225 ■ Tworzenie ConfigMap 227 ■ Przekazywanie wpisu ConfigMap do kontenera jako zmiennej środowiskowej 230 ■ Przekazywanie naraz wszystkich wpisów w obiekcie ConfigMap jako zmiennych środowiskowych 231 ■ Przekazywanie wpisu ConfigMap jako argumentu wiersza polecenia 232 ■ Używanie woluminu configMap do eksponowania wpisów ConfigMap jako plików 233 ■ Aktualizowanie konfiguracji aplikacji bez konieczności jej restartowania 239

7.5 Wykorzystanie obiektów Secret do przekazywania poufnych danych do kontenerów 242

Przedstawiamy obiekty Secret 242 ■ Przedstawiamy sekret domyślnego tokenu 243 ■ Tworzenie sekretu 245 ■ Porównanie obiektów ConfigMap i Secret 246 ■ Korzystanie z sekretu w podzie 248 ■ Jak obrazy pobierają sekrety 252

7.6 Podsumowanie 254

8 Dostęp z aplikacji do metadanych podu i innych zasobów 255

8.1 Przekazywanie metadanych za pośrednictwem Downward API 256

Dostępne metadane 257 ■ Eksponowanie metadanych za pośrednictwem zmiennych środowiskowych 257 ■ Przekazywanie metadanych przez pliki w woluminie downwardAPI 260

8.2 Komunikowanie się z serwerem API Kubernetes 264

Poznanie API typu REST w Kubernetes 265 ■ Komunikowanie się z serwerem API z wnętrza podu 270 ■ Upraszczanie komunikacji z serwerem API za pomocą kontenerów ambasadorów 275 ■ Wykorzystanie bibliotek klienckich w celu komunikacji z serwerem API 278

8.3 Podsumowanie 281

9 Obiekty Deployment: deklaratywne aktualizowanie aplikacji 282

9.1 Aktualizowanie aplikacji działających w podach 283

Usuwanie starych podów i zastępowanie ich nowymi 284 ■ Wystartowanie nowych podów, a następnie usuwanie starych 285

9.2 Wykonywanie automatycznej aktualizacji kroczącej przy użyciu kontrolera replikacji 286

Uruchamianie wstępnej wersji aplikacji 287 ■ Wykonywanie aktualizacji kroczącej przy użyciu kubectl 288 ■ Dlaczego kubectl rolling-update jest już przestarzałe 293

- 9.3 Korzystanie z obiektów Deployment w celu deklaratywnego aktualizowania aplikacji 294
 - Tworzenie zasobu Deployment 295
 - Aktualizowanie zasobu Deployment 297
 - Wycofywanie wdrożenia 302
 - Kontrolowanie tempa aktualizacji 305
 - Wstrzymywanie procesu aktualizacji 307
 - Blokowanie rozpowszechniania wadliwych wersji 309
- 9.4 Podsumowanie 313

10 Zasoby StatefulSet: wdrażanie replikowanych aplikacji stanowych 315

- 10.1 Replikowanie podów stanowych 316
 - Uruchamianie wielu replik, każdej z oddzielną pamięcią masową 316
 - Zapewnianie stabilnej tożsamości każdego podu 318
- 10.2 Istota obiektów StatefulSet 319
 - Porównanie obiektów StatefulSet i ReplicaSet 319
 - Zapewnianie stabilnej tożsamości sieciowej 321
 - Zapewnianie stabilnej dedykowanej pamięci masowej dla każdego wystąpienia z pamięcią stanu 324
 - Gwarancje StatefulSet 326
- 10.3 Korzystanie ze StatefulSet 326
 - Tworzenie aplikacji i obrazu kontenera 326
 - Wdrażanie aplikacji przy użyciu StatefulSet 328
 - Testowanie naszych podów 332
- 10.4 Odkrywanie partnerów w StatefulSet 337
 - Implementowanie odkrywania partnerów przy użyciu DNS 338
 - Aktualizowanie StatefulSet 340
 - Testowanie klastrowanego magazynu danych 341
- 10.5 Jak zasoby StatefulSet radzą sobie z awariami węzłów 342
 - Symulowanie odłączenia węzła od sieci 342
 - Ręczne usuwanie podu 344
- 10.6 Podsumowanie 346

CZĘŚĆ III POZA PODSTAWAMI 347

11 Poznawanie wewnętrznych mechanizmów Kubernetes 349

- 11.1 Poznajemy architekturę 350
 - Rozproszona natura komponentów Kubernetes 351
 - Jak Kubernetes używa etcd 353
 - Co robi serwer API 357
 - Jak serwer API powiadamia klientów o zmianach zasobów 359
 - Poznajemy Scheduler 360
 - Przedstawiamy kontrolery działające w komponencie Controller Manager 363
 - Co robi Kubelet 368
 - Rola komponentu Kubernetes Service Proxy 369
 - Przedstawiamy dodatki Kubernetes 370
 - Zbieranie wszystkiego razem 372

- 11.2 Jak współpracują kontrolery 372
 - Jakie komponenty są zaangażowane 372
 - Łącuch zdarzeń 373
 - Obserwowanie zdarzeń klastra 375
 - 11.3 Czym jest uruchomiony pod 376
 - 11.4 Łączność sieciowa wewnątrz podu 377
 - Jaka musi być ta sieć 377
 - Zagłębianie się w działanie sieci 379
 - Przedstawiamy Container Network Interface 381
 - 11.5 Jak są implementowane usługi 381
 - Przedstawiamy kube-proxy 382
 - Jak proces kube-proxy używa iptables 382
 - 11.6 Uruchamianie klastrów wysokiej dostępności 384
 - Zapewnianie wysokiej dostępności naszych aplikacji 384
 - Zapewnianie wysokiej dostępności komponentów warstwy sterowania Kubernetes 385
 - 11.7 Podsumowanie 388
- 12 Zabezpieczanie serwera API Kubernetes 389**
- 12.1 Istota uwierzytelniania 390
 - Użytkownicy i grupy 390
 - Przedstawiamy zasoby ServiceAccount 391
 - Tworzenie zasobów ServiceAccount 393
 - Przypisywanie konta usługowego do podu 395
 - 12.2 Zabezpieczanie klastra przy użyciu kontroli dostępu opartej na rolach 397
 - Przedstawiamy wtyczkę autoryzującą RBAC 397
 - Przedstawiamy zasoby RBAC 399
 - Używanie zasobów Role i RoleBinding 402
 - Używanie ról klastra i wiązań ról klastra 406
 - Domyślne role klastra i wiązania ról klastra 416
 - Rozważne przyznawanie uprawnień autoryzacyjnych 418
 - 12.3 Podsumowanie 419
- 13 Zabezpieczanie węzłów i sieci klastra 421**
- 13.1 Używanie w podzie przestrzeni nazw hostującego węzła 422
 - Używanie w podzie przestrzeni nazw sieciowych węzła 422
 - Dowiązywanie do portu hosta bez używania przestrzeni nazw sieciowych hosta 423
 - Używanie przestrzeni nazw PID i IPC węzła 426
 - 13.2 Konfigurowanie kontekstu zabezpieczeń kontenera 427
 - Uruchamianie kontenera jako określony użytkownik 428
 - Powstrzymywanie kontenera przed działaniem jako root 428
 - Uruchamianie podów w trybie uprzywilejowanym 429
 - Dołączanie do kontenera indywidualnych możliwości jądra 431
 - Odrzucanie możliwości z kontenera 433
 - Powstrzymywanie

procesów przed zapisywaniem w systemie plików kontenera 434 ▪ Współużytkowanie woluminów, gdy kontenery działają jako różni użytkownicy 435

13.3 Ograniczanie stosowania w podach funkcji związanych z zabezpieczeniami 437

Przedstawiamy zasób PodSecurityPolicy 437 ▪ Zasady runAsUser, fsGroup oraz supplementalGroups 440 ▪ Konfigurowanie dozwolonych, domyślnych i zabronionych możliwości 442 ▪ Ograniczanie typów woluminów używanych przez pody 444 ▪ Przypisywanie różnych zasobów PodSecurityPolicy do różnych użytkowników i grup 444

13.4 Izolowanie sieci podów 448

Włączanie izolacji sieciowej w przestrzeni nazw 448 ▪ Zezwalanie tylko niektórym podom z przestrzeni nazw na łączenie się z podem serwera 449 ▪ Izolacja sieci między przestrzeniami nazw Kubernetes 450 ▪ Izolowanie sieci przy użyciu notacji CIDR 451 ▪ Ograniczanie ruchu wychodzącego dla zbioru podów 452

13.5 Podsumowanie 452

14 Zarządzanie zasobami obliczeniowymi podów 454

14.1 Żądanie zasobów dla kontenerów podu 455

Tworzenie podów z żądaniami zasobów 455 ▪ Jak żądania zasobów wpływają na rozmieszczanie podów 456 ▪ Jak żądania procesora wpływają na współdzielenie czasu procesora 461 ▪ Definiowanie i żądanie niestandardowych zasobów 462

14.2 Ograniczanie zasobów dostępnych dla kontenera 463

Ustawienie sztywnego limitu ilości zasobów, których może używać kontener 463 ▪ Przekraczanie limitów 465 ▪ Jak aplikacje w kontenerach postrzegają limity 466

14.3 Klasy QoS dla podów 468

Definiowanie klasy QoS dla podu 468 ▪ Który proces zostanie zabity, gdy brakuje pamięci 471

14.4 Ustawianie domyślnych żądań i limitów dla podów w przestrzeni nazw 472

Przedstawiamy zasób LimitRange 473 ▪ Tworzenie obiektu LimitRange 474 ▪ Wymuszanie limitów 475 ▪ Stosowanie domyślnych żądań i limitów zasobów 476

14.5 Ograniczanie łącznych zasobów dostępnych w przestrzeni nazw 477

Przedstawiamy obiekt ResourceQuota 477 ▪ Specyfikowanie przydziałów trwałej pamięci masowej 479 ▪ Ograniczenie liczby dozwolonych obiektów 480 ▪ Specyfikowanie przydziałów dla określonych stanów podów i/lub klas QoS 481

14.6 Monitorowanie użycia zasobów przez pody 483

Gromadzenie i odczytywanie rzeczywistego użycia zasobów 483 ■ Zapisywanie i analizowanie historycznych statystyk zużycia zasobów 485

14.7 Podsumowanie 489

15 Automatyczne skalowanie podów i węzłów klastra 490

15.1 Automatyczne skalowanie podów w poziomie 491

Proces skalowania automatycznego 491 ■ Skalowanie oparte na użyciu procesora 495 ■ Skalowanie oparte na użyciu pamięci 502 ■ Skalowanie oparte na innych i niestandardowych miarach 502 ■ Ustalanie, jakie miary są odpowiednie dla automatycznego skalowania 504 ■ Skalowanie w dół do zera replik 505

15.2 Automatyczne skalowanie podów w pionie 505

Automatyczne konfigurowanie żądań zasobów 506 ■ Modyfikowanie żądań zasobów w czasie działania podu 506

15.3 Horyzontalne skalowanie węzłów klastra 506

Przedstawiamy Cluster Autoscaler 507 ■ Włączanie funkcjonalności Cluster Autoscaler 509 ■ Ograniczanie przerw w działaniu usługi w trakcie skalowania klastra w dół 510

15.4 Podsumowanie 511

16 Zaawansowane rozmieszczanie 512

16.1 Używanie skaz i tolerancji w celu odpychania podów od pewnych węzłów 513

16.1.1 Skazy i tolerancje 513 ■ Dodawanie niestandardowych skaz do węzła 515 ■ Dodawanie tolerancji do podów 516 ■ Do czego można używać skaz i tolerancji 517

16.2 Używanie koligacji węzła w celu przyciągania podów do określonych węzłów 518

Specyfikowanie sztywnych reguł koligacji węzła 519 ■ Priorytetyzowanie węzłów podczas rozmieszczania podów 521

16.3 Kolokacja podów przy użyciu koligacji i anty-koligacji 525

Używanie koligacji między podami w celu wdrażania podów w tym samym węźle 525 ■ Wdrażanie podów w tej samej szafie, strefie dostępności lub regionie geograficznym 528 ■ Formułowanie preferencji koligacji zamiast sztywnych wymagań 529 ■ Rozmieszczanie podów z dala od siebie nawzajem przy użyciu anty-koligacji 531

16.4 Podsumowanie 533

17 Zalecenia dla tworzenia aplikacji 535

17.1 Zbieranie wszystkiego razem 536

17.2 Cykl życia podu 537

Aplikacje muszą się spodziewać, że zostaną zabite i relokowane 538 ■ Ponowne rozmieszczanie martwych lub częściowo martwych podów 540 ■ Uruchamianie podów w określonej kolejności 542 ■ Dodawanie hooków cyklu życia 544 ■ Wyłączanie 548

17.3 Zapewnianie, że wszystkie żądania klientów są właściwie obsłużone 552

Unikanie zrywania połączeń klienckich podczas uruchamiania podu 552 ■ Unikanie zrywania połączeń klienckich w trakcie wyłączania 552

17.4 Tworzenie aplikacji łatwych do zarządzania w Kubernetes 557

Tworzenie zarządzalnych obrazów kontenerów 557 ■ Właściwe oznakowanie obrazów i rozważne używanie imagePullPolicy 558 ■ Używanie etykiet wielowymiarowych zamiast jednowymiarowych 558 ■ Opisywanie zasobów za pomocą adnotacji 559 ■ Udostępnianie informacji o przyczynach zakończenia procesu 559 ■ Obsługa dzienników aplikacji 561

17.5 Najlepsze praktyki programowania i testowania 563

Uruchamianie aplikacji poza Kubernetes w trakcie programowania 563 ■ Korzystanie z Minikube w wytwarzaniu 564 ■ Wersjonowanie i automatyczne wdrażanie manifestów zasobów 566 ■ Ksonnet jako alternatywa dla pisania manifestów YAML/JSON 566 ■ Stosowanie ciągłej integracji i ciągłego dostarczania (Continuous Integration i Continuous Delivery – CI/CD) 567

17.6 Podsumowanie 568

18 Rozszerzanie Kubernetes 569

18.1 Definiowanie niestandardowych obiektów API 570

Przedstawiamy CustomResourceDefinition 570 ■ Automatyzowanie niestandardowych zasobów przy użyciu niestandardowych kontrolerów 574 ■ Walidacja niestandardowych obiektów 579 ■ Udostępnianie niestandardowego serwera API dla niestandardowych obiektów 580

18.2 Rozszerzanie Kubernetes przez Service Catalog 582

Przedstawiamy Service Catalog 582 ■ Przedstawiamy serwer API Service Catalog oraz Controller Manager 583 ■ Brokery usług i OpenServiceBroker API 584 ■ Przygotowywanie i używanie usługi 586 ■ Odwiązanie i wyrejestrowanie 589 ■ Co przynosi Service Catalog 589

18.3 Platformy zbudowane na bazie Kubernetes 590

Platforma kontenerów Red Hat OpenShift 590 ■ Deis Workflow oraz Helm 594

18.4 Podsumowanie 596

Dodatek A Używanie kubectl z wieloma klastrami 597

- A.1 Przełączanie między Minikube a Google Kubernetes Engine 597
- A.2 Używanie kubectl wobec wielu klastrów lub przestrzeni nazw 598
 - Konfigurowanie lokalizacji pliku kubeconfig 598
 - Zawartości pliku kubeconfig 598
 - Listowanie, dodawanie i modyfikowanie wpisów pliku kubeconfig 600
 - Używanie kubectl wobec różnych klastrów, użytkowników i kontekstów 601
 - Przełączanie się między kontekstami 602
 - Listowanie kontekstów i klastrów 602
 - Usuwanie kontekstów i klastrów 602

Dodatek B Tworzenie wielowęzłowego klastra przy użyciu kubeadm 603

- B.1 Konfigurowanie systemu operacyjnego i wymaganych pakietów 603
 - Tworzenie maszyny wirtualnej 603
 - Konfigurowanie karty sieciowej maszyny wirtualnej 604
 - Instalowanie systemu operacyjnego 605
 - Instalowanie Dockera i Kubernetes 609
 - Klonowanie maszyny wirtualnej 610
- B.2 Konfigurowanie węzła master przy użyciu kubeadm 612
 - Jak kubeadm uruchamia komponenty 613
- B.3 Konfigurowanie węzłów roboczych przy użyciu kubeadm 614
 - Konfigurowanie sieci kontenerów 615
- B.4 Korzystanie z klastra z maszyny lokalnej 616

Dodatek C Korzystanie z innych środowisk wykonawczych kontenerów 617

- C.1 Używanie innych środowisk kontenerów za pośrednictwem CRI 617
 - Środowisko wykonawcze kontenerów CRI-O 617
 - Uruchamianie aplikacji w maszynach wirtualnych zamiast w kontenerach 618

Dodatek D Cluster Federation 619

- D.1 Przedstawiamy Kubernetes Cluster Federation 619
- D.2 Architektura 620
- D.3 Sfederowane obiekty API 621
 - Sfederowane wersje zasobów Kubernetes 621
 - Co robią sfederowane zasoby 622

indeks 624