O'REILLY®

Helion

Kubernetes Receptury

Aplikacje natywne dla środowiska chmurowego

Wydanie II

Sameer Naik
Sébastien Goasguen
Jonathan Michaux

Tytuł oryginału: Kubernetes Cookbook: Building Cloud Native Applications, 2nd Edition

Tłumaczenie: Magdalena A. Tkacz

ISBN: 978-83-289-1345-5

Authorized Polish translation of the English edition of *Kubernetes Cookbook, 2E*
ISBN 9781098142247 © 2024 CloudTank SARL, Sameer Naik, and Jonathan Michaux.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/kurea2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

Przedmowa **9**

1. Pierwsze kroki z Kubernetes **13**
1.1. Instalacja interfejsu wiersza poleceń Kubernetes (kubectl) 13
1.2. Instalacja Minikube w celu uruchomienia lokalnej instancji Kubernetes 14
1.3. Lokalne wykorzystanie Minikube jako środowiska deweloperskiego 16
1.4. Uruchamianie swojej pierwszej aplikacji na Minikube 17
1.5. Używanie kind do uruchamiania Kubernetes lokalnie 18
1.6. Korzystanie z Kubernetes w Docker Desktop 19
1.7. Przełączanie kontekstów kubectl 21
1.8. Przełączanie kontekstów i przestrzeni nazw przy użyciu kubectx i kubens 21

2. Tworzenie klastra Kubernetes **23**
2.1. Przygotowanie nowego węzła dla klastra Kubernetes 23
2.2. Uruchamianie węzła warstwy sterowania Kubernetes 25
2.3. Instalowanie dodatku Container Network dla sieci klastrów 26
2.4. Dodawanie węzłów roboczych do klastra Kubernetes 27
2.5. Uruchamianie kokpitu nawigacyjnego Kubernetes 28
2.6. Uzyskanie dostępu do kokpitu nawigacyjnego Kubernetes 28
2.7. Wdrażanie serwera Kubernetes Metrics Server 31
2.8. Pobieranie określonej wersji Kubernetes z serwisu GitHub 33
2.9. Pobieranie plików binarnych klienta i serwera 34
2.10. Korzystanie z plików jednostek systemd do uruchamiania komponentów Kubernetes 35
2.11. Tworzenie klastra Kubernetes w Google Kubernetes Engine 38
2.12. Tworzenie klastra Kubernetes w usłudze Azure Kubernetes Service 39
2.13. Tworzenie klastra Kubernetes w usłudze Amazon Elastic Kubernetes Service 42

3. Jak korzystać z klienta Kubernetes 44
3.1. Lista zasobów 44
3.2. Usuwanie zasobów 45
3.3. Obserwowanie zmian zasobów za pomocą kubectl 47
3.4. Edytowanie obiektów za pomocą kubectl 47
3.5. Uzyskiwanie dodatkowych informacji co do zasobów i pól z kubectl 48

4. Tworzenie i modyfikowanie podstawowych obciążeń roboczych 50
4.1. Tworzenie poda przy użyciu kubectl run 50
4.2. Tworzenie wdrożenia przy użyciu kubectl create 51
4.3. Tworzenie obiektów z pliku manifestów 52
4.4. Pisanie od podstaw manifestu dla poda 53
4.5. Uruchamianie wdrożenia przy użyciu manifestu 54
4.6. Aktualizacja wdrożenia 56
4.7. Uruchamianie zadań w trybie wsadowym 59
4.9. Uruchamianie demonów infrastruktury na węzeł 62

5. Praca z usługami 64
5.1. Tworzenie usługi eksponującej Twoją aplikację 65
5.2. Weryfikacja wpisu DNS dla usługi 66
5.3. Zmiana typu usługi 67
5.4. Wdrażanie kontrolera ingress 70
5.5. Udostępnianie usług spoza klastra 71

6. Zarządzanie manifestami aplikacji 74
6.1. Instalacja Helm, menedżera pakietów dla Kubernetes 74
6.2. Dodawanie repozytoriów ze schematami do Helm 75
6.3. Używanie Helm do instalowania aplikacji 76
6.4. Sprawdzanie, jakie parametry schematu są konfigurowalne 78
6.5. Nadpisywanie parametrów schematu 79
6.6. Pobieranie podawanych przez użytkownika parametrów wydania Helm 79
6.7. Odinstalowywanie aplikacji z wykorzystaniem Helm 80
6.8. Tworzenie własnego schematu — przygotowanie paczki z aplikacją dla Helm 81
6.9. Instalacja Kompose 82
6.10. Konwertowanie plików Docker Compose na manifesty Kubernetes 83
6.11. Konwersja pliku Docker Compose na schemat Helm 84
6.12. Instalacja kapp 84
6.13. Wdrażanie manifestów YAML przy użyciu kapp 85

7. Odkrywanie interfejsu API Kubernetes i kluczowe metadane 87
7.1. Wykrywanie punktów końcowych serwera API Kubernetes 87
7.2. Zrozumienie struktury manifestu Kubernetes 89
7.3. Tworzenie przestrzeni nazw w celu uniknięcia kolizji nazw 90
7.4. Ustawianie ograniczeń w przestrzeni nazw 91
7.5. Etykietowanie obiektu 92
7.6. Używanie etykiet w zapytaniach 93
7.7. Dodawanie komentarzy do zasobu za pomocą jednego polecenia 94

8. Woluminy i dane konfiguracyjne 96
8.1. Wymiana danych między kontenerami za pośrednictwem woluminu lokalnego 96
8.2. Przekazywanie klucza dostępu API do podów przy użyciu sekretu 98
8.2. Przekazywanie klucza dostępu API do podów przy użyciu sekretu 98
8.3. Dostarczanie danych konfiguracyjnych do aplikacji 101
8.4. Używanie trwałego woluminu w Minikube 103
8.5. O trwałości danych w Minikube 106
8.6. Przechowywanie zaszyfrowanych sekretów w systemie kontroli wersji 108

9. Skalowanie 112
9.1. Skalowanie wdrożenia 112
9.2. Korzystanie z automatycznego, poziomego skalowania poda (HPA) 113
9.3. Automatyczna zmiana rozmiaru klastra w GKE 115
9.4. Automatyczna zmiana rozmiaru klastra w Amazon EKS 118

10. Bezpieczeństwo 120
10.1. Zapewnianie unikalnej tożsamości dla aplikacji 120
10.2. Wyświetlanie i przeglądanie informacji związanych z kontrolą dostępu 122
10.3. Kontrolowanie dostępu do zasobów 125
10.4. Zabezpieczanie podów 127

11. Monitorowanie i rejestrowanie 129
11.1. Dostęp do dzienników kontenera 129
11.2. Odzyskiwanie ze stanu uszkodzonego za pomocą sondy aktywności 131
11.3. Kontrolowanie przepływu ruchu do poda przy użyciu sondy gotowości 131
11.4. Ochrona wolno uruchamiających się kontenerów za pomocą sondy rozruchowej 132
11.5. Dodawanie sond aktywności i gotowości do wdrożeń 133
11.6. Uzyskiwanie dostępu do wskaźników Kubernetes w interfejsie linii poleceń 136
11.7. Korzystanie z Prometheus i Grafana na Minikube 137

12. Utrzymywanie systemu i rozwiązywanie problemów 140
12.1. Włączanie autouzupełniania dla kubectl 140
12.2. Usuwanie podów z usługi 141
12.3. Uzyskiwanie spoza klastra dostępu do usługi ClusterIP 143
12.4. Zrozumienie i analizowanie stanu zasobów 144
12.5. Debugowanie podów 146
12.6. Wpływanie na zachowanie podów podczas uruchamiania 150
12.7. Uzyskiwanie szczegółowej migawki stanu klastra 152
12.8. Dodawanie węzłów roboczych Kubernetes 153
12.9. Czasowe odłączanie węzła Kubernetes na potrzeby wykonania konserwacji 155

13. Sieci usług 157
13.1. Instalacja sieci usług Istio 157
13.2. Wdrażanie mikrousługi za pomocą sidecara Istio 159
13.3. Trasowanie ruchu przy użyciu usługi wirtualnej Istio 161
13.4. Nadpisywanie adresu URL przy użyciu usługi wirtualnej Istio 162
13.5. Instalacja usługi Linkerd Service Mesh 164
13.6. Wdrażanie usługi w sieci Linkerd Mesh 166
13.7. Linkerd: przekierowywanie ruchu do usługi 167
13.8. Linkerd: autoryzowanie ruchu do serwera 169

14. Aplikacje bezserwerowe i aplikacje sterowane zdarzeniami 171
14.1. Instalacja Knative Operator 171
14.2. Instalacja komponentu Knative Serving 172
14.3. Instalacja Knative CLI 174
14.4. Tworzenie usługi Knative 175
14.5. Instalacja komponentu Knative Eventing 176
14.6. Wdrażanie Knative Eventing Source 177
14.7. Włączanie Knative Eventing Sources 180
14.8. Instalowanie źródeł zdarzeń z TriggerMesh 181

15. Rozbudowywanie Kubernetes 183
15.1. Kompilacja ze źródła 183
15.2. Kompilowanie określonego komponentu 184
15.3. Używanie oprogramowania klienckiego języka Python do pracy z Kubernetes API 185
15.4. Rozszerzanie interfejsu API z wykorzystaniem definicji niestandardowych zasobów 186

Dodatek. Zasoby 191

ROZDZIAŁ 2.

Tworzenie klastra Kubernetes

W tym rozdziale omawiamy wiele sposobów na skonfigurowanie w pełni funkcjonalnego klastra Kubernetes. Omawiamy niskopoziomowe, ustandaryzowane narzędzia (`kubeadm`), które są również bazą dla innych narzędzi umożliwiających instalację, pokazujemy, gdzie znaleźć odpowiednie pliki binarne dla **warstwy kontroli** (ang. *control plane*), a także dla węzłów roboczych (ang. *worker node*). Pokazujemy, jak przygotować pliki dla jednostek (ang. *unit files*) `systemd` umożliwiające kontrolę nad komponentami Kubernetes, a na koniec pokazujemy, jak skonfigurować klastry na platformach Google Cloud Platform i Azure.

2.1. Przygotowanie nowego węzła dla klastra Kubernetes

Problem

Chcesz przygotować nowy węzeł ze wszystkimi wymaganymi narzędziami, aby utworzyć nowy klaster Kubernetes lub dodać go do istniejącego klastra.

Rozwiązanie

Aby przygotować hosta opartego na Ubuntu do pracy w klastrze Kubernetes, musisz najpierw włączyć przekierowywanie IPv4 i włączyć `iptables`, co umożliwi mostkowanie ruchu:

```
$ cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

$ sudo modprobe overlay
$ sudo modprobe br_netfilter
$ cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
EOF

$ sudo sysctl -system
```

Do zapewnienia zgodności z narzędziem `kubeadm` na węźle trzeba wyłączyć mechanizm wymiany (ang. *swap*):

```
$ sudo apt install cron -y
$ sudo swapoff -a
$ (sudo crontab -l 2>/dev/null; echo "@reboot /sbin/swapoff -a") | sudo crontab - || true
```

Węzły klastra wymagają specjalnego interfejsu dla Kubernetes: **interfejsu uruchomieniowego dla kontenerów** (ang. *Container Runtime Interface*, CRI). Jednym z takich interfejsów jest `cri-o` (*https://cri-o.io*). Wersja `cri-o` powinna być zgodna z wersją Kubernetes. Na przykład, jeśli uruchamiasz klaster Kubernetes 1.27, skonfiguruj odpowiednio zmienną `VERSION`:

```
$ VERSION="1.27"
$ OS="xUbuntu_22.04"

$ cat <<EOF | sudo tee /etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list
deb https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/$OS/ /
EOF

$ cat <<EOF | sudo tee /etc/apt/sources.list.d/devel:kubic:libcontainers:stable:cri-o:
$VERSION.list
deb http://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable:/cri-o:/
$VERSION/$OS/ /
EOF

$ curl -L
https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable:/cri-o:/
$VERSION/$OS/Release.key | sudo apt-key add -
$ curl -L
https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/$OS/
Release.key | sudo apt-key add -

$ sudo apt-get update

$ sudo apt-get install cri-o cri-o-runc cri-tools -y
```

Następnie przeładuj konfigurację `systemd` i włącz `cri-o`:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable crio --now
```

Narzędzie `kubeadm` jest wymagane do uruchomienia klastra Kubernetes od zera, ale także wtedy, gdy chcesz coś dołączyć do istniejącego klastra. Włącz repozytorium jego oprogramowania za pomocą następujących poleceń:

```
$ cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/k8s-archive-keyring.gpg] https://apt.kubernetes.io/
kubernetes-xenial main
EOF

$ sudo apt-get install -y apt-transport-https ca-certificates curl $ sudo curl -fsSLo
/etc/apt/keyrings/k8s-archive-keyring.gpg https://dl.k8s.io/apt/doc/apt-key.gpg

$ sudo apt-get update
```

Teraz możesz zainstalować wszystkie narzędzia wymagane do uruchomienia węzła klastra Kubernetes. Będziesz potrzebować następujących elementów:

- binariów `kubelet`,
- interfejsu CLI narzędzia `kubeadm`,
- oprogramowania klienckiego `kubectl`.

Aby zainstalować te elementy, uruchom następujące polecenie:

```
$ sudo apt-get install -y kubelet kubeadm kubectl
```

Następnie oznacz te pakiety jako **zatrzymane** (ang. *held back*), co uniemożliwi ich automatyczną aktualizację:

```
$ sudo apt-mark hold kubelet kubeadm kubectl
```

Twój host działający pod kontrolą Ubuntu jest teraz gotowy do dołączenia do klastra Kubernetes.

Omówienie

`kubeadm` to narzędzie konfiguracyjne, dzięki któremu masz dostęp do poleceń `kubeadm init` i `kubeadm join`. Polecenie `kubeadm init` służy do uruchamiania **węzła warstwy sterowania** Kubernetes (ang. *control-plane node*), podczas gdy polecenie `kubeadm join` służy do uruchamiania węzła roboczego i dołączania go do klastra. Zasadniczo `kubeadm` umożliwia wykonanie działań niezbędnych do uruchomienia i działania klastra o minimalnej funkcjonalności. `kubelet` zaś pełni funkcję **agenta węzła** (ang. *node agent*), działającego na każdym węźle.

Inne niż `cri-o` środowiska uruchomieniowe kontenerów, z którymi warto się zapoznać, to *containerd* (*https://oreil.ly/M1kDx*), Docker Engine (*https://oreil.ly/P5_l_*) i Mirantis Container Runtime (*https://oreil.ly/BEWaG*).

2.2. Uruchamianie węzła warstwy sterowania Kubernetes

Problem

Skonfigurowałeś już i uruchomiłeś host Ubuntu, który ma docelowo działać pod kontrolą Kubernetes (patrz receptura 2.1), a teraz musisz uruchomić nowy węzeł warstwy sterowania Kubernetes.

Rozwiązanie

Po zainstalowaniu binariów `kubeadm` możesz rozpocząć uruchamianie klastra Kubernetes. Zainicjuj warstwę sterowania na węźle następującymi poleceniami:

```
$ NODENAME=$(hostname -s)
$ IPADDR=$(ip route get 8.8.8.8 | sed -n 's/.*src \([^\ ]*\).*/\1/p')
$ POD_CIDR=192.168.0.0/16
```

Węzeł warstwy sterowania powinien dysponować co najmniej dwoma procesorami wirtualnymi i 2 GB pamięci RAM.

Teraz zainicjuj węzeł warstwy sterowania (ang. *control-plane node*) za pomocą `kubeadm`:

```
$ sudo kubeadm init --apiserver-advertise-address=$IPADDR \
    --apiserver-cert-extra-sans=$IPADDR \
    --pod-network-cidr=$POD_CIDR \
    --node-name $NODENAME \
    --ignore-preflight-errors Swap
[init] Using Kubernetes version: v1.27.2
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
...
```

Informacje wyświetlane w odpowiedzi na polecenie `init` zawierają konfigurację potrzebną do skonfigurowania `kubectl` do komunikacji z Twoim klastrem. Po skonfigurowaniu `kubectl` możesz zweryfikować stan komponentu klastra za pomocą następującego polecenia:

```
$ kubectl get --raw='/readyz?verbose'
```

Aby z kolei uzyskać informacje o klastrze, użyj polecenia:

```
$ kubectl cluster-info
```

Omówienie

Obciążenia generowane przez użytkownika zwykle nie są przewidziane do wykonywania na węźle warstwy sterowania. W przypadku gdy tworzysz eksperymentalny klaster mający pojedynczy węzeł, musisz odpowiednio „oznaczyć" (ang. *taint*, dosł. zepsuć, skazić) węzeł warstwy sterowania, aby umożliwić zaplanowanie na węźle warstwy sterowania wykonywanie obciążeń generowanych przez użytkownika:

```
$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-
```

Zobacz także

- Tworzenie klastra za pomocą `kubeadm` (*https://oreil.ly/q9nwI*).

2.3. Instalowanie dodatku Container Network dla sieci klastrów

Problem

Uruchomiłeś węzeł warstwy sterowania Kubernetes (patrz receptura 2.2), a teraz musisz zainstalować dodatek umożliwiający komunikację sieciową podów tak, aby mogły się one ze sobą komunikować.

Rozwiązanie

Możesz zainstalować w węźle warstwy sterowania dodatek sieciowy Calico za pomocą następującego polecenia:

```
$ kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml
```

Omówienie

Musisz użyć kompatybilnego z Twoim klastrem dodatku Container Network Interface (CNI) odpowiadającego Twoim potrzebom. Dostępnych jest wiele dodatków, które implementują CNI. Zapoznaj się z (niewyczerpującą wszystkich możliwości) listą dostępnych dodatków w dokumentacji Kubernetes (*https://oreil.ly/HosU6*).

2.4. Dodawanie węzłów roboczych do klastra Kubernetes

Problem

Zainicjowałeś swój węzeł warstwy sterowania Kubernetes (patrz receptura 2.2) oraz dodatek CNI (patrz receptura 2.3), a teraz chcesz dodać do swojego klastra węzły robocze.

Rozwiązanie

Po zainicjowaniu hosta Ubuntu przygotowanego do pracy z Kubernetes, co pokazaliśmy w recepturze 2.1, wykonaj następujące polecenie na węźle warstwy sterowania, aby uzyskać dostęp do polecenia `join`:

```
$ kubeadm token create --print-join-command
```

Teraz na węźle roboczym wydaj polecenie `join`:

```
$ sudo kubeadm join --token <token>
```

Węzeł roboczy powinien mieć co najmniej jeden procesor vCPU i 2 GB pamięci RAM.

Wróć do sesji terminala węzła warstwy sterowania, by zobaczyć podłączone węzły:

```
$ kubectl get nodes
NAME     STATUS   ROLES          AGE  VERSION
master   Ready    control-plane  28m  v1.27.2
worker   Ready    <none>         10s  v1.27.2
```

Możesz powtórzyć te kroki, aby dodać więcej węzłów roboczych do klastra Kubernetes.

Omówienie

Węzły robocze to miejsca, w których uruchamiane są Twoje obciążenia. Kiedy zasoby klastra zaczną się wyczerpywać, zaczniesz zauważać, że nowe pody będą miały status *Pending*. W tym momencie powinieneś rozważyć dodanie większej ilości zasobów do klastra przez dodanie większej liczby węzłów roboczych.

2.5. Uruchamianie kokpitu nawigacyjnego Kubernetes

Problem

Utworzyłeś klaster Kubernetes, a teraz chcesz mieć możliwość tworzenia skonteneryzowanych w klastrze obciążeń, wyświetlania ich i zarządzania nimi za pomocą interfejsu użytkownika.

Rozwiązanie

Użyj kokpitu nawigacyjnego Kubernetes (*https://oreil.ly/n7WQw*), będącego internetowym interfejsem użytkownika do wdrażania skonteneryzowanych aplikacji w klastrze Kubernetes i zarządzania zasobami klastra.

Jeśli korzystasz z Minikube, możesz zainstalować kokpit nawigacyjny Kubernetes po prostu przez włączenie dodatku *dashboard*:

```
$ minikube addons enable dashboard
```

Aby uruchomić kokpit nawigacyjny Kubernetes w wersji 2.7.0, wykonaj następujące czynności:

```
$ kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml
```

Następnie sprawdź, czy jest gotowy do użycia:

```
$ kubectl get deployment kubernetes-dashboard -n kubernetes-dashboard
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-dashboard   1/1      1           1           44s
```

2.6. Uzyskanie dostępu do kokpitu nawigacyjnego Kubernetes

Problem

Zainstalowałeś w swoim klastrze kokpit nawigacyjny Kubernetes (patrz receptura 2.5) i chcesz uzyskać do niego dostęp z poziomu przeglądarki internetowej.

Rozwiązanie

Musisz utworzyć konto `ServiceAccount` (*https://oreil.ly/pXErB*) (dosł. konto usługi) z uprawnieniami do administrowania klastrem. Utwórz plik o nazwie *sa.yaml* i wypełnij go następującą zawartością:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

Teraz możesz utworzyć konto `ServiceAccount` za pomocą polecenia:

```
$ kubectl apply -f sa.yaml
```

Aby uzyskać dostęp do kokpitu nawigacyjnego Kubernetes, będziesz potrzebować tokenu uwierzytelniania powiązanego z tym kontem. Token (dosł. żeton) pojawi się jako dane wyjściowe poniższego polecenia:

```
$ kubectl -n kubernetes-dashboard create token admin-user
eyJhbGciOiJSUzI1NiIsImtpZCI6...
```

Zapisz go.

Ponieważ kokpit nawigacyjny Kubernetes jest usługą lokalną klastra, musisz skonfigurować połączenie proxy z klastrem:

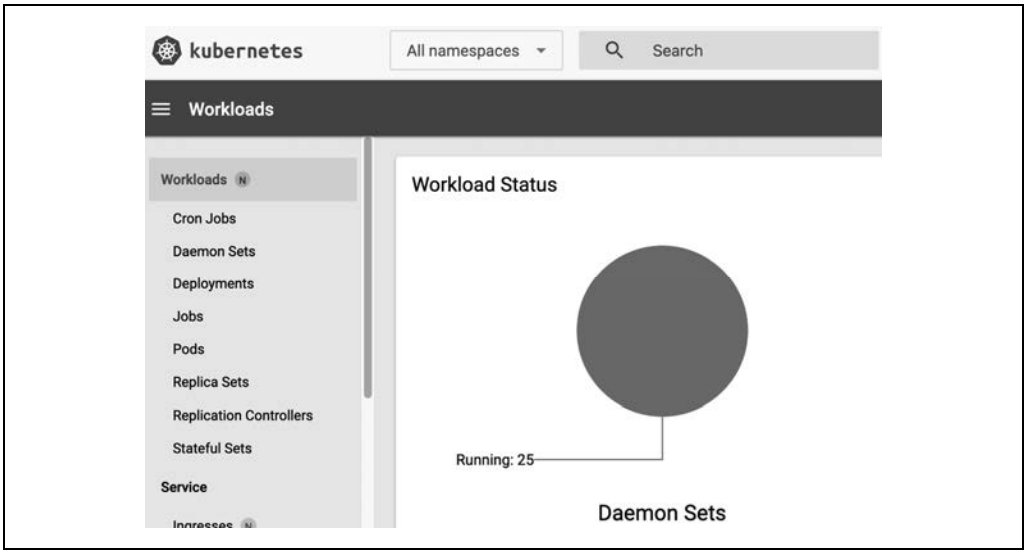
```
$ kubectl proxy
```

Teraz, odwiedzając stronę *http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/workloads?namespace=_all*, możesz otworzyć kokpit nawigacyjny Kubernetes i uwierzytelnić się przy użyciu (utworzonego wcześniej) tokenu uwierzytelniania.

W interfejsie użytkownika, który zostanie otwarty w przeglądarce, zobaczysz stronę przedstawioną na rysunku 2.1.

Jeśli korzystasz z Minikube, wszystko, co musisz zrobić, to wydać polecenie:

```
$ minikube dashboard
```

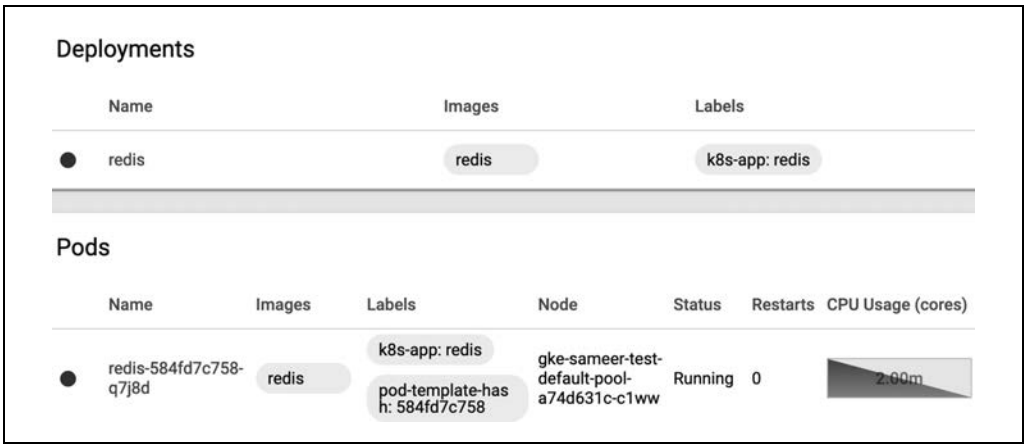


Rysunek 2.1. Zrzut ekranu kokpitu — widok tworzenia aplikacji

Omówienie

Aby utworzyć aplikację, kliknij znak plusa (+) w prawym górnym rogu, wybierz zakładkę *Create from form* (utwórz z formularza), podaj nazwę aplikacji i wskaż obraz kontenera, którego chcesz użyć. Następnie kliknij przycisk *Deploy* (wdrażaj) — zostanie wyświetlony nowy widok, który pokazuje wdrożenia (ang. *deployments*), pody i zestawy replik (ang. *replica sets*). W Kubernetes dostępne są dziesiątki typów istotnych zasobów, takich jak wdrożenia, pody, zestawy replik, usługi i tak dalej, które omówimy bardziej szczegółowo w dalszej części książki.

Zrzut ekranu pokazany na rysunku 2.2 przedstawia typowy widok kokpitu po utworzeniu jednej aplikacji przy użyciu kontenera Redis.



Rysunek 2.2. Widok kokpitu z aplikacją Redis

Jeśli teraz wrócisz do sesji terminala i użyjesz klienta wiersza poleceń, zobaczysz to samo:

```
$ kubectl get all
NAME                         READY  STATUS   RESTARTS  AGE
pod/redis-584fd7c758-vwl52   1/1    Running  0         5m9s

NAME                 TYPE        CLUSTER-IP  EXTERNAL-IP  PORT(S)  AGE
service/kubernetes   ClusterIP   10.96.0.1   <none>       443/TCP  19m

NAME                    READY   UP-TO-DATE  AVAILABLE  AGE
deployment.apps/redis   1/1     1           1          5m9s

NAME                               DESIRED  CURRENT  READY  AGE
replicaset.apps/redis-584fd7c758   1        1        1      5m9s
```

Twój „redisowy” pod będzie uruchamiać serwer Redis, co możesz zobaczyć w dziennikach po wydaniu polecenia:

```
$ kubectl logs redis-3215927958-4x88v
...
1:C 25 Aug 2023 06:17:23.934 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 25 Aug 2023 06:17:23.934 * Redis version=7.2.0, bits=64, commit=00000000, modified=0,
pid=1, just started
1:C 25 Aug 2023 06:17:23.934 # Warning: no config file specified, using the default config.
In order to specify a config file use redis-server/path/to/redis.conf
1:M 25 Aug 2023 06:17:23.934 * monotonic clock: POSIX clock_gettime
1:M 25 Aug 2023 06:17:23.934 * Running mode=standalone, port=6379.
1:M 25 Aug 2023 06:17:23.935 * Server initialized
1:M 25 Aug 2023 06:17:23.935 * Ready to accept connections tcp
```

2.7. Wdrażanie serwera Kubernetes Metrics Server

Problem

Wdrożyłeś kokpit Kubernetes (patrz receptura 2.5), ale nie widzisz w nim informacji o wykorzystaniu procesora i pamięci.

Rozwiązanie

Kokpit Kubernetes do wizualizacji użycia procesora i pamięci wymaga serwera Kubernetes Metrics Server (*https://oreil.ly/BEHwR*) (dosł. serwer wskaźników).

Jeśli korzystasz z Minikube, możesz zainstalować Kubernetes Metrics Server po prostu przez włączenie dodatku *metrics-server* poleceniem:

```
$ minikube addons enable metrics-server
```

Aby wdrożyć najnowszą wersję Kubernetes Metrics Server, wpisz następujące polecenie:

```
$ kubectl apply -f https://github.com/kubernetes-sigs/metrics-
server/releases/latest/download/components.yaml
```

Następnie sprawdź, czy wszystko jest gotowe:

```
$ kubectl get deployment metrics-server -n kube-system
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           7m27s
```

Jeśli widzisz, że wdrożenie nie przechodzi w stan gotowości, sprawdź dzienniki poda:

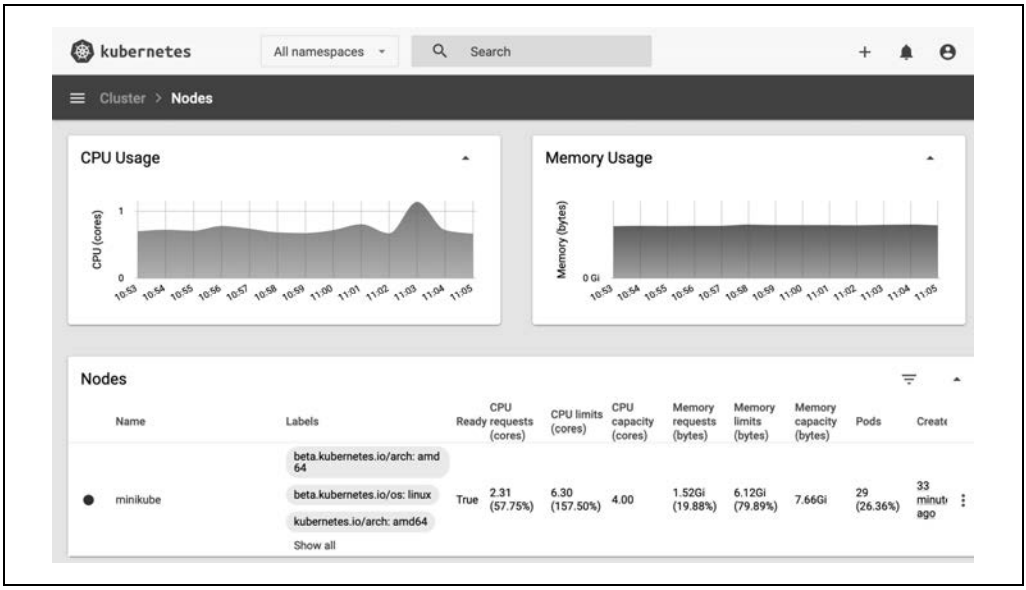
```
$ kubectl logs -f deployment/metrics-server -n kube-system
I0707 05:06:19.537981   1 server.go:187] "Failed probe" probe="metric-storage-ready" err="no
metrics to serve"
E0707 05:06:26.395852   1 scraper.go:140] "Failed to scrape node" err="Get
\"https://192.168.64.50:10250/metrics/resource\": x509: cannot validate certificate for
192.168.64.50 because it doesn't contain any IP SANs" node="minikube"
```

Jeśli zobaczysz komunikat o błędzie treści *cannot validate certificate* (dosł. nie można zweryfikować certyfikatu), podczas wdrożenia Metrics Server musisz dodać opcję `--kubelet-insecure-tls`:

```
$ kubectl patch deployment metrics-server -n kube-system --type='json' -p='[{"op": "add",
"path": "/spec/template/spec/containers/0/args/-", "value": "--kubelet-insecure-tls"}]'
```

Po uruchomieniu Metrics Server może minąć kilka minut, zanim będzie on dostępny. Jeśli nie jest jeszcze w stanie gotowości, żądania metryk mogą powodować błędy.

Po uruchomieniu Metrics Server w kokpicie Kubernetes wyświetlone zostaną statystyki użycia procesora i pamięci, co pokazano na rysunku 2.3.



Rysunek 2.3. Widok węzłów klastra w kokpicie

Omówienie

Wskaźniki dotyczące pracy węzłów i podów można również przeglądać w wierszu poleceń za pomocą polecenia `kubectl top`:

```
$ kubectl top pods -A
NAMESPACE     NAME                               CPU(cores)  MEMORY(bytes)
kube-system   coredns-5d78c9869d-5fh78           9m          9Mi
kube-system   etcd-minikube                      164m        36Mi
kube-system   kube-apiserver-minikube            322m        254Mi
kube-system   kube-controller-manager-minikube   123m        35Mi
kube-system   kube-proxy-rvl8v                   13m         12Mi
kube-system   kube-scheduler-minikube            62m         15Mi
kube-system   storage-provisioner                22m         7Mi
```

Podobnie, aby wyświetlić wskaźniki dotyczące węzłów, skorzystaj z następującego polecenia:

```
$ kubectl top nodes
NAME       CPU(cores)  CPU%    MEMORY(bytes)  MEMORY%
minikube   415m        10%     1457Mi         18%
```

Zobacz także

- Repozytorium Kubernetes Metrics Server GitHub (*https://oreil.ly/C_O6W*)
- Dokumentacja potoku wskaźników zasobów (*https://oreil.ly/ODZCr*)

2.8. Pobieranie określonej wersji Kubernetes z serwisu GitHub

Problem

Chcesz pobrać oficjalną wersję Kubernetes, zamiast kompilować ją ze źródła.

Rozwiązanie

Projekt Kubernetes publikuje archiwum dla każdego wydania (ang. *release*). Odnośnik do archiwum można znaleźć w pliku *CHANGELOG* danego wydania. Przejdź do folderu *CHANGELOG* na stronie projektu (*https://oreil.ly/MMwRs*) i otwórz plik *CHANGELOG* dla wybranego wydania. W pliku znajdziesz odnośnik do pobrania pliku *kubernetes.tar.gz* danego wydania.

Na przykład, jeśli chcesz pobrać wersję v1.28.0, otwórz *CHANGELOG-1.28.md*, a w sekcji zatytułowanej *Downloads for v1.28.0* znajdziesz odnośnik do *kubernetes.tar.gz* (*https://dl.k8s.io/v1.28.0/kubernetes.tar.gz*).

```
$ wget https://dl.k8s.io/v1.28.0/kubernetes.tar.gz
```

Jeśli chcesz skompilować Kubernetes ze źródła, zerknij do receptury 15.1.

Omówienie

Plik *CHANGELOG* zawiera również wartość funkcji skrótu sha512 archiwum *kubernetes.tar.gz*. Dobrze jest sprawdzić integralność archiwum *kubernetes.tar.gz*, aby się upewnić, że nie zostało ono w żaden sposób zmodyfikowane. Aby to zrobić, wygeneruj lokalnie skrót sha512 pobranego archiwum i porównaj go z tym wymienionym w *CHANGELOG*:

```
$ sha512sum kubernetes.tar.gz
9aaf7cc004d09297dc7bbc1f0149.....kubernetes.tar.gz
```

2.9. Pobieranie plików binarnych klienta i serwera

Problem

Pobrałeś archiwum wydania (patrz receptura 2.8), ale okazuje się, że nie zawiera ono plików binarnych.

Rozwiązanie

Ponieważ archiwa wydań mają być jak najmniejsze, nie zawierają one plików binarnych wydań. Dlatego musisz je pobrać osobno. Aby to zrobić, uruchom skrypt *get-kube-binaries.sh* za pomocą pokazanych niżej poleceń:

```
$ tar -xvf kubernetes.tar.gz
$ cd kubernetes/cluster
$ ./get-kube-binaries.sh
```

Po zakończeniu binaria klienta będziesz mieć w *client/bin*:

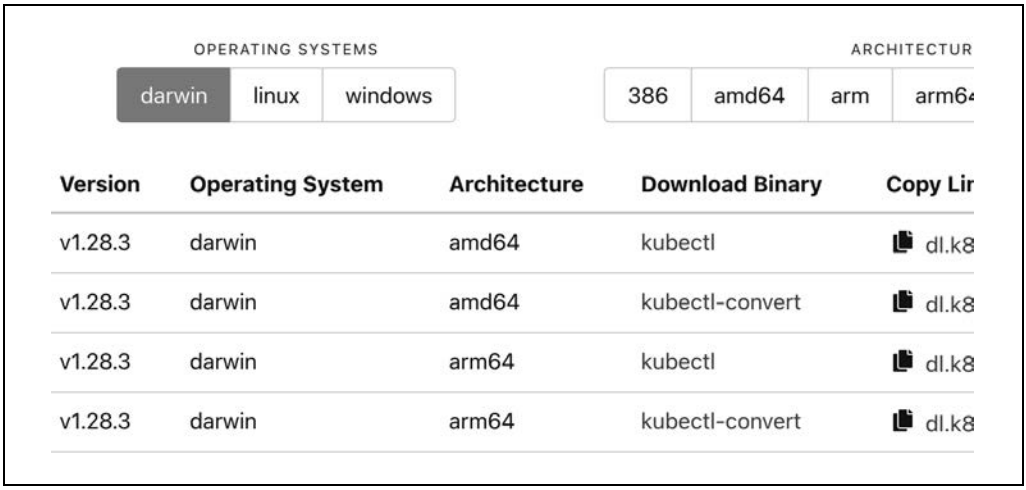
```
$ ls ../client/bin
kubectl         kubectl-convert
```

a archiwum zawierające binaria serwera — w *server/kubernetes*:

```
$ ls ../server/kubernetes
kubernetes-server-linux-amd64.tar.gz   kubernetes-manifests.tar.gz   README
...
```

Omówienie

Jeśli chcesz pominąć pobieranie całego archiwum wydania i szybko pobrać pliki binarne klienta i serwera, możesz pobrać je bezpośrednio ze strony Download Kubernetes (*https://oreil.ly/tdN0P*). Na tej stronie znajdziesz bezpośrednie łącza do plików binarnych dla różnych kombinacji systemów operacyjnych i architektur, co pokazano na rysunku 2.4.



Rysunek 2.4. Strona downloadkubernetes.com, lista plików binarnych wydania Kubernetes v1.28.0 dla systemu operacyjnego Darwin

2.10. Korzystanie z plików jednostek systemd do uruchamiania komponentów Kubernetes

Problem

Użyłeś Minikube (patrz receptura 1.2) do nauki i wiesz, jak uruchomić klaster Kubernetes za pomocą `kubeadm` (patrz receptura 2.2), ale chcesz zainstalować klaster od podstaw.

Rozwiązanie

Aby to zrobić, musisz uruchomić komponenty Kubernetes za pomocą **plików jednostek systemd** (ang. *systemd unit files*). Aby uruchomić `kubelet` za pośrednictwem `systemd`, potrzebujesz tylko podstawowych przykładów.

Sprawdzenie, w jaki sposób, za pomocą plików jednostkowych systemd, `kubeadm` konfiguruje demony Kubernetes, by je uruchomić, pomoże Ci zrozumieć, jak zrobić to samodzielnie. Jeśli przyjrzysz się bliżej konfiguracji `kubeadm`, zobaczysz, że `kubelet` jest uruchomiony na każdym węźle w klastrze, w tym na węźle warstwy sterowania.

Oto przykład, który możesz naśladować, logując się do dowolnego węzła klastra zbudowanego za pomocą `kubeadm` (patrz receptura 2.2):

```
$ systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
     Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
    Drop-In: /etc/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
```

```
     Active: active (running) since Tue 2023-05-30 04:21:29 UTC; 2h 49min ago
       Docs: https://kubernetes.io/docs/home/
   Main PID: 797 (kubelet)
      Tasks: 11 (limit: 2234)
     Memory: 40.2M
        CPU: 5min 14.792s
     CGroup: /system.slice/kubelet.service
             └─797 /usr/bin/kubelet \
               --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf \
               --kubeconfig=/etc/kubernetes/kubelet.conf \
               --config=/var/lib/kubelet/config.yaml \
               --container-runtime-endpoint=unix:///var/run/crio/crio.sock \
               --pod-infra-container-image=registry.k8s.io/pause:3.9
```

W ten sposób widzisz łącze do pliku jednostek *systemd* w */lib/systemd/system/kubelet.service* i jego konfiguracji w *etc/systemd/system/kubelet.service.d/10-kubeadm.conf*.

Plik jednostki jest prosty — wskazuje na plik binarny `kubelet` zainstalowany w */usr/bin*:

```
[Unit]
Description=kubelet: The Kubernetes Node Agent
Documentation=https://kubernetes.io/docs/home/
Wants=network-online.target
After=network-online.target

[Service]
ExecStart=/usr/bin/kubelet
Restart=always
StartLimitInterval=0
RestartSec=10

[Install]
WantedBy=multi-user.target
```

Plik konfiguracyjny określa sposób uruchamiania pliku binarnego `kubelet`:

```
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/
bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf"
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
EnvironmentFile=-/etc/default/kubelet

ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS
$KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS
```

Wszystkie podane opcje zdefiniowane przez zmienną środowiskową `$KUBELET_CONFIG_ARGS` (takie jak `–kubeconfig`) są opcjami uruchamiania binariów *kubelet* (*https://oreil.ly/quccc*).

Omówienie

`systemd` (*https://oreil.ly/RmuZp*) to menedżer systemu i usług, czasami określany jako *init system*. Obecnie jest domyślnym menedżerem usług w Ubuntu i CentOS.

Przedstawiony plik jednostki dotyczy tylko `kubelet`. Możesz napisać własne pliki jednostek dla pozostałych komponentów klastra Kubernetes, tj. serwera API (ang. *API server*), menedżera kontrolera (ang. *controller manager*), harmonogramu (ang. *scheduler*), proxy. Przykłady plików jednostek dla każdego z komponentów znajdziesz w książce *Kubernetes the Hard Way* (*https://oreil.ly/AWnxD*).

Wystarczy jednak uruchomić tylko `kubelet`. Zauważ, że opcja konfiguracyjna `--pod-manifest-path` pozwala wskazać katalog, gdzie `kubelet` będzie szukał manifestów, które następnie automatycznie uruchomi. W przypadku `kubeadm` katalog ten jest używany do przechowywania manifestów serwera API, harmonogramu, `etcd` i menedżera kontrolera. W ten sposób Kubernetes zarządza sam sobą, a jedyną rzeczą zarządzaną przez `systemd` jest proces `kubelet`.

Aby to zilustrować, możesz wyświetlić zawartość katalogu */etc/kubernetes/manifests* w klastrze opartym na `kubeadm`:

```
$ ls -l /etc/kubernetes/manifests
total 16
-rw-------  1  root root 2393  May 29  11:04 etcd.yaml
-rw-------  1  root root 3882  May 29  11:04 kube-apiserver.yaml
-rw-------  1  root root 3394  May 29  11:04 kube-controller-manager.yaml
-rw-------  1  root root 1463  May 29  11:04 kube-scheduler.yaml
```

Analizując bardziej szczegółowo manifest w *etcd.yaml*, możesz zobaczyć, że jest to pod z pojedynczym kontenerem, który uruchamia *etcd*:

```
$ cat /etc/kubernetes/manifests/etcd.yaml

apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/etcd.advertise-client-urls: https://10.10.100.30:2379
  creationTimestamp: null
  labels:
    component: etcd
    tier: control-plane
  name: etcd
  namespace: kube-system
spec:
  containers:
  - command:
    - etcd
    - --advertise-client-urls=https://10.10.100.30:2379
    - --cert-file=/etc/kubernetes/pki/etcd/server.crt
    - --client-cert-auth=true
    - --data-dir=/var/lib/etcd
    - --experimental-initial-corrupt-check=true
    - --experimental-watch-progress-notify-interval=5s
    - --initial-advertise-peer-urls=https://10.10.100.30:2380
    - --initial-cluster=master=https://10.10.100.30:2380
    - --key-file=/etc/kubernetes/pki/etcd/server.key
    - --listen-client-urls=https://127.0.0.1:2379,https://10.10.100.30:2379
    - --listen-metrics-urls=http://127.0.0.1:2381
    - --listen-peer-urls=https://10.10.100.30:2380
    - --name=master
    - --peer-cert-file=/etc/kubernetes/pki/etcd/peer.crt
```

```
- --peer-client-cert-auth=true
- --peer-key-file=/etc/kubernetes/pki/etcd/peer.key
- --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
- --snapshot-count=10000
- --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
image: registry.k8s.io/etcd:3.5.7-0
...
```

Zobacz także

- Opcje konfiguracji `kubelet` (*https://oreil.ly/E95yp*)

2.11. Tworzenie klastra Kubernetes w Google Kubernetes Engine

Problem

Chcesz utworzyć klaster Kubernetes w Google Kubernetes Engine (GKE).

Rozwiązanie

Zanim zaczniesz korzystać z GKE, potrzebujesz kilku rzeczy:

- konta Google Cloud Platform (GCP) (*https://oreil.ly/CAiDf*) z włączoną opcją obciążania konta opłatami,
- projektu GCP z włączonym GKE (*https://oreil.ly/eGX2n*),
- zainstalowanego Google Cloud SDK (*https://oreil.ly/Y00rC*).

Google Cloud SDK zawiera narzędzie linii poleceń `gcloud` CLI do interakcji z usługami GCP z poziomu wiersza poleceń. Po zainstalowaniu SDK uwierzytelnij `gcloud`, aby uzyskać dostęp do projektu GCP:

```
$ gcloud auth login
```

Korzystając z interfejsu wiersza poleceń `gcloud`, utwórz klaster Kubernetes za pomocą polecenia `container clusters create` w następujący sposób:

```
$ gcloud container clusters create oreilly --zone us-east1-b
```

Domyślnie spowoduje to utworzenie klastra Kubernetes z trzema węzłami roboczymi w określonej strefie lub regionie. Węzeł główny (ang. *master node*) jest zarządzany przez usługę GKE i nie można uzyskać do niego dostępu.

Jeśli nie masz pewności, jakiej strefy lub regionu (*https://oreil.ly/4Bvua*) użyć dla argumentu `--zone` lub `--region`, użyj polecenia `gcloud compute zones list` lub `gcloud compute regions list` i wybierz strefę albo region w Twoim pobliżu. Strefy są zazwyczaj mniej zasobożerne niż regiony.

Po zakończeniu korzystania z klastra nie zapomnij go usunąć, aby uniknąć naliczenia opłaty:

```
$ gcloud container clusters delete oreilly --zone us-east1-b
```

Omówienie

Możesz pominąć instalację `gcloud` CLI, korzystając z Google Cloud Shell (*https://oreil.ly/E-Qcr*), rozwiązania opartego wyłącznie na przeglądarce internetowej.

Listę istniejących klastrów GKE możesz wyświetlić za pomocą następującego polecenia:

```
$ gcloud container clusters list --zone us-east1-b
NAME     ZONE        MASTER_VERSION     MASTER_IP      ...   STATUS
oreilly  us-east1-b  1.24.9-gke.2000    35.187.80.94   ...   RUNNING
```

Interfejs `gcloud` CLI umożliwia zmianę rozmiaru klastra, jego aktualizację i uaktualnianie:

```
...
COMMANDS
...
    resize
       Resizes an existing cluster for running containers.
    update
       Update cluster settings for an existing container cluster.
    upgrade
       Upgrade the Kubernetes version of an existing container cluster.
```

Zobacz także

- Szybki start GKE (*https://oreil.ly/WMDSx*)
- Szybki start Google Cloud Shell (*https://oreil.ly/_w0va*)

2.12. Tworzenie klastra Kubernetes w usłudze Azure Kubernetes Service

Problem

Chcesz utworzyć klaster Kubernetes w usłudze Azure Kubernetes Service (AKS).

Rozwiązanie

Aby utworzyć klaster AKS, będziesz potrzebować następujących elementów:

- konta założonego na Microsoft Azure Portal (*https://oreil.ly/PyUA0*),
- zainstalowanego interfejsu Azure CLI (*https://oreil.ly/An7xM*).

Na początku upewnij się, że masz zainstalowany Azure CLI w wersji 2.0 lub nowszej, a następnie zaloguj się na stronie do platformy Azure:

```
$ az --version | grep "^azure-cli"
azure-cli                         2.50.0 *
$ az login
To sign in, use a web browser to open the page https://aka.ms/devicelogin and enter the code
XXXXXXXXX to authenticate.
[
  {
    "cloudName": "AzureCloud",
    "id": "****************************",
    "isDefault": true,
    "name": "Free Trial",
    "state": "Enabled",
    "tenantId": "****************************",
    "user": {
      "name": "******@hotmail.com",
      "type": "user"
    }
  }
]
```

Utwórz grupę zasobów Azure o nazwie k8s, aby przechowywać wszystkie zasoby AKS, takie jak maszyny wirtualne i komponenty sieciowe, oraz ułatwić ich późniejsze czyszczenie i usuwanie:

```
$ az group create --name k8s --location northeurope
{
  "id": "/subscriptions/************************/resourceGroups/k8s",
  "location": "northeurope",
  "managedBy": null,
  "name": "k8s",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

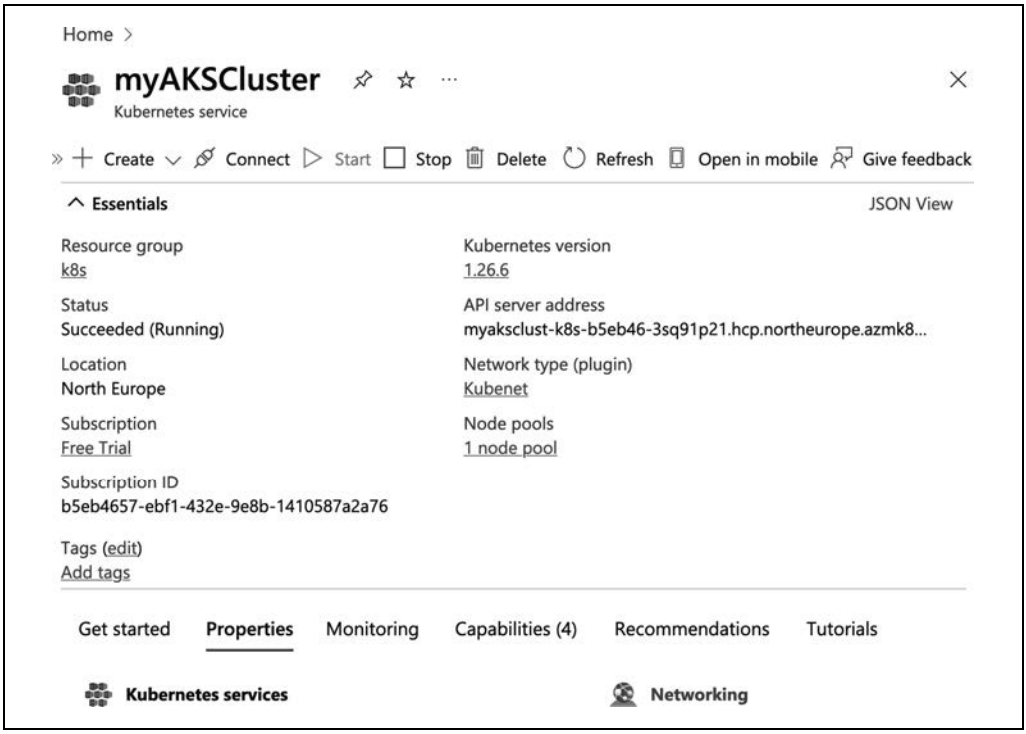
Jeśli nie masz pewności, jakiego regionu (*https://oreil.ly/fdGdc*) użyć dla argumentu `--location`, wykonaj polecenie `az account list-locations` i wybierz któryś znajdujący się blisko Ciebie.

Teraz gdy masz już skonfigurowaną grupę zasobów k8s, możesz utworzyć klaster z jednym węzłem roboczym (**agentem** w terminologii Azure) w następujący sposób:

```
$ az aks create -g k8s -n myAKSCluster --node-count 1 --generate-ssh-keys
{
  "aadProfile": null,
  "addonProfiles": null,
  "agentPoolProfiles": [
    {
      "availabilityZones": null,
      "count": 1,
      "creationData": null,
      "currentOrchestratorVersion": "1.26.6",
```

Weź pod uwagę, że wykonanie polecenia `az aks create` może zająć kilka minut. Po zakończeniu polecenie zwraca obiekt JSON z informacjami o utworzonym klastrze.

W rezultacie w Azure Portal powinieneś zobaczyć coś takiego jak na rysunku 2.5. Zacznij od znalezienia grupy zasobów o nazwie k8s, a następnie przejdź do zakładki *Deployments*.



Rysunek 2.5 Portal Azure — klaster AKS w grupie zasobów k8s

Możesz już połączyć się z klastrem:

```
$ az aks get-credentials --resource-group k8s --name myAKSCluster
```

Możesz teraz rozejrzeć się w środowisku i zweryfikować konfigurację:

```
$ kubectl cluster-info
Kubernetes master is running at https://k8scb-k8s-143f1emgmt.northeurope.cloudapp.azure.com
Heapster is running at https://k8scb-k8s-143f1emgmt.northeurope.cloudapp.azure.com/
api/v1/namespaces/kube-system/services/heapster/proxy
KubeDNS is running at https://k8scb-k8s-143f1emgmt.northeurope.cloudapp.azure.com/
api/v1/namespaces/kube-system/services/kube-dns/proxy
kubernetes-dashboard is running at https://k8scb-k8s-143f1emgmt.northeurope.cloudapp.
azure.com/api/v1/namespaces/kube-system/services/kubernetes-dashboard/proxy
tiller-deploy is running at https://k8scb-k8s-143f1emgmt.northeurope.cloudapp.azure.com/
api/v1/namespaces/kube-system/services/tiller-deploy/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

$ kubectl get nodes
NAME                                STATUS  ROLES  AGE  VERSION
aks-nodepool1-78916010-vmss000000 Ready   agent  26m  v1.24.9
```

I rzeczywiście, widać tu, że faktycznie utworzyliśmy klaster z jednym węzłem.

Jeśli nie chcesz lub nie możesz zainstalować interfejsu Azure CLI, alternatywną opcją jest użycie powłoki Azure Cloud Shell (*https://oreil.ly/IUFJQ*) z poziomu przeglądarki.

Gdy już skończysz się rozglądać w AKS, nie zapomnij zamknąć klastra i usunąć wszystkich zasobów przez usunięcie grupy zasobów `k8s`:

```
$ az group delete --name k8s --yes --no-wait
```

Chociaż polecenie `az group delete` ze względu na obecność opcji `--no-wait` zaczyna się wykonywać natychmiast, usunięcie wszystkich zasobów i faktyczne zniszczenie grupy zasobów może potrwać do 10 minut. Możesz sprawdzić w Azure Portal, czy wszystko przebiegło zgodnie z planem.

Zobacz także

- *Szybki start: Wdrażanie klastra usługi Azure Kubernetes przy użyciu interfejsu Azure CLI* (*https://learn.microsoft.com/pl-pl/azure/aks/learn/quick-kubernetes-deploy-cli*) w dokumentacji Microsoft Azure.

2.13. Tworzenie klastra Kubernetes w usłudze Amazon Elastic Kubernetes Service

Problem

Chcesz utworzyć klaster Kubernetes w usłudze Amazon Elastic Kubernetes Service (EKS).

Rozwiązanie

Aby utworzyć klaster w Amazon EKS, potrzebujesz:

- konta Amazon Web Services (*https://aws.amazon.com*),
- zainstalowanego AWS CLI (*https://aws.amazon.com/cli*),
- zainstalowanego narzędzia `eksctl` (*https://eksctl.io*) do pracy w linii poleceń.

Po zainstalowaniu AWS CLI uwierzytelnij klienta (*https://oreil.ly/_6VMv*), aby uzyskać dostęp do konta AWS:

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: eu-central-1
Default output format [None]:
```

Narzędzie `eksctl` jest oficjalnym narzędziem linii poleceń dla Amazon EKS. Używa poświadczeń AWS, które skonfigurowałeś do uwierzytelniania w AWS. Korzystając z tego narzędzia, utwórz klaster:

```
$ eksctl create cluster --name helion --region eu-central-1
2023-08-29 13:21:12 [i] eksctl version 0.153.0-dev+a79b3826a.2023-08-18T...
2023-08-29 13:21:12 [i] using region eu-central-1
...
2023-08-29 13:36:52 [✓] EKS cluster "helion" in "eu-central-1" region is ready
```

Domyślnie `eksctl` tworzy w określonym regionie klaster z dwoma węzłami roboczymi. Możesz zmienić ich liczbę z wykorzystaniem opcji `--nodes`.

Aby uzyskać najniższe opóźnienia, wybierz ten region AWS (*https://oreil.ly/Kc9GZ*), który znajduje się najbliżej Ciebie.

Gdy klaster EKS nie będzie już potrzebny, usuń go, aby uniknąć naliczania opłat za niewykorzystywane zasoby:

```
$ eksctl delete cluster helion --region eu-central-1
```

Zobacz także

- Wprowadzenie do `eksctl` (*https://eksctl.io/getting-started*)
- Usługa Amazon Elastic Kubernetes (*https://aws.amazon.com/eks*)

Kubernetes w praktyce. Znajdź szybkie rozwiązania dla każdego wyzwania!

Kubernetes stał się standardem orkiestracji kontenerów i zarządzania rozproszonymi aplikacjami. Obecnie pozwala na uzyskiwanie automatyzacji, rozszerzalności i przenośności na wysokim poziomie, a sama praca z nim jest efektywna i satysfakcjonująca. Niekiedy jednak tworzenie i wdrażanie aplikacji Kubernetes sprawia problemy nawet zaawasowanym użytkownikom.

To książka przeznaczona dla osób pracujących z systemami Kubernetes. Poszczególne zagadnienia przedstawiono w niej w przejrzystym formacie *problem – rozwiązanie – omówienie*, dzięki czemu szybko znajdziesz odpowiedź na konkretne pytanie, a to z kolei pozwoli Ci na korzystanie ze wszystkich zalet tej platformy w codziennej pracy. Znalazło się tu ponad 100 praktycznych receptur obejmujących takie tematy jak konfigurowanie klastra, zarządzanie obciążeniem w kontenerach przy użyciu obiektów Kubernetes API, użycie pamięci masowej indeksowanej wartościami klucz-wartość, konfigurowanie zabezpieczeń i wiele innych. Każda receptura jest niezależna od innych, a materiał został zorganizowany w sposób maksymalnie ułatwiający szybkie odnalezienie interesującego Cię zagadnienia.

W książce między innymi:

- tworzenie klastra i interfejs wiersza poleceń Kubernetes
- zarządzanie obciążeniami i usługami
- interfejs API Kubernetes
- skalowanie na poziomie klastra i aplikacji
- zabezpieczanie aplikacji i jej monitorowanie
- utrzymywanie systemów w środowisku chmury i rozwiązywanie problemów

Sameer Naik od wczesnego etapu był zaangażowany w projekt Kubernetes. Jest twórcą - założycielem projektu Helm Charts i współzałożycielem startupu NextBit Computing.

Sébastien Goasguen jest współzałożycielem TriggerMesh i członkiem Apache Software Foundation. Obecnie zajmuje się rozwiązaniami kontenerowymi.

Dr Jonathan Michaux jest menedżerem produktu i inżynierem oprogramowania. Zajmował się zarządzaniem API i mikrousługami, a ostatnio tworzy aplikacje sterowane zdarzeniami dla Kubernetes.

Helion

helion.pl

HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-289-1345-5

9 788328 913455

Cena: 59,90 zł