



Wprowadzenie do Kubernetes



Piotr Majorczyk

Kontakt: [linkedin.com/in/piotr-majorczyk/](https://www.linkedin.com/in/piotr-majorczyk/)



Plan prezentacji

1. Wprowadzenie do Dockera
2. Wprowadzenie do Kubernetesa
3. Podstawowe elementy Kubernetesa
4. Demo



Czym jest Docker?

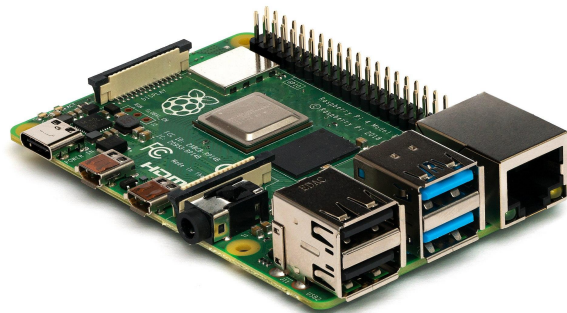
- Pozwala na zapakowanie aplikacji wraz z OS, bibliotekami i zależnościami w ustandaryzowane kontenery (obrazy dockerowe)
- Ułatwia i ujednolica uruchomienie aplikacji
- Rozwiązuje problem "u mnie działa"

Czym jest Kubernetes?

- Pozwala zarządzać aplikacjami opakowanymi w kontenery
- Może być używany w różnych środowiskach: chmurze publicznej, hybrydowej i prywatnej

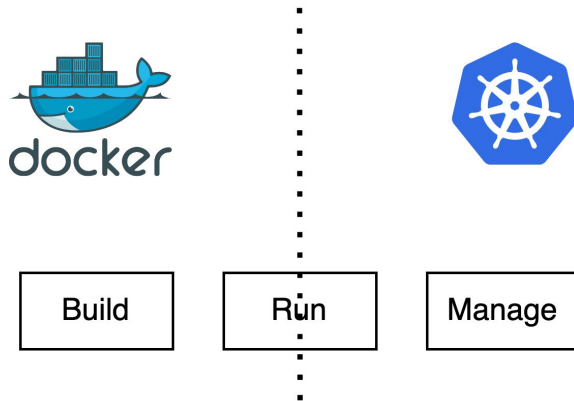


Google Cloud



Współpraca Docker-Kubernetes

- Docker buduje i opakowuje aplikacje jako obrazy
- Docker to *container runtime* - jest to technologia która uruchamia i zatrzymuje kontenery
- Docker jest *deprecated* jako *container runtime*. K8s powinien używać implementacji zgodnych z CRI
- Kubernetes dostarcza narzędzia do orkiestracji i zarządzania kontenerami





Architektura Kubernetesa

- Cluster kubernetesa składa się z *control plane* (AKA *master node*) oraz z *worker node*ów
- *control plane* wystawia API, rozdziela pracę na *worker node*'y, monitoruje stan clustra i aplikacji, dba o to aby stan zadany był utrzymywany oraz zapewnia integrację z cloud providerem
- *worker node*'y wykonują pracę zadaną przez *control plane*



Podstawowe obiekty Kubernetesa

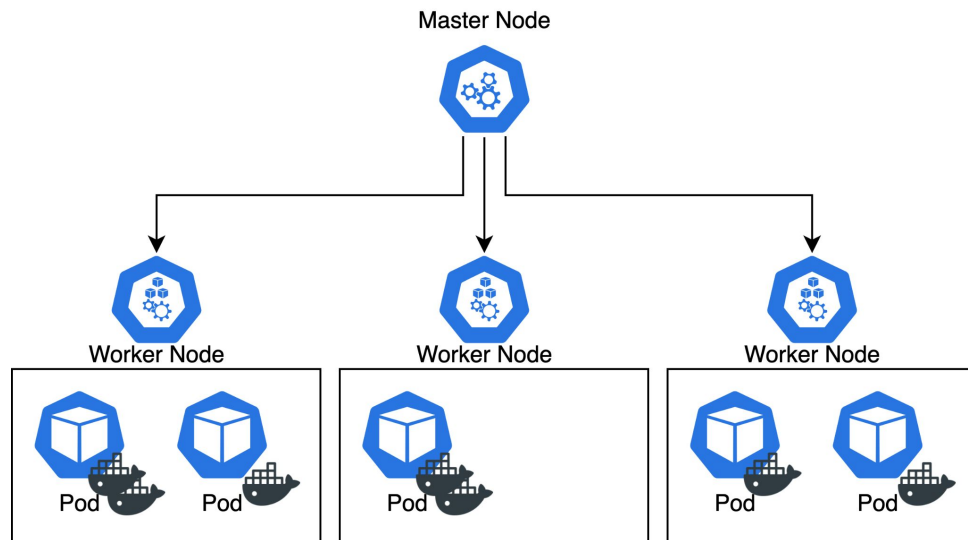
- *Node*
- *Pod*
- *Service*
- *Deployment*
- *ReplicaSet*
- *ConfigMap*
- *Secret*

Uwaga:

Można sprawdzić listę wszystkich typów obiektów Kubernetesa przez komendę `kubectl api-resources`

Node

- *Node* to maszyna na której wykonywane są obliczenia
- Może być to VM, instancja na cloudzie lub fizyczna maszyna
- Jest środowiskiem na którym działają *Pod'y*
- Zazwyczaj cluster zawiera wiele *Node'ów*





Pod

- Reprezentacja pojedynczej instancji procesu działającego na clustrze
- *Pod* sam w sobie nie uruchamia aplikacji - uruchamiane one są przez kontenery
- Może zawierać jeden lub więcej kontenerów (tightly coupled)
- Jest jednostką skalowania aplikacji
- Może działać tylko w obrębie jednego *Node'a*
- Jest łatwo zbywalny (pet vs cattle)
- Jest niemutowalny
- Zazwyczaj nie zarządzamy bezpośrednio *Pod'*ami

Uwaga:

Przykładem aplikacji które powinny być tight coupled w świecie k8s i powinny znajdować się w jednym clustrze jest aplikacja główna i procesy pomocnicze (np. proxy, adapter, zarządzanie logami)



Service

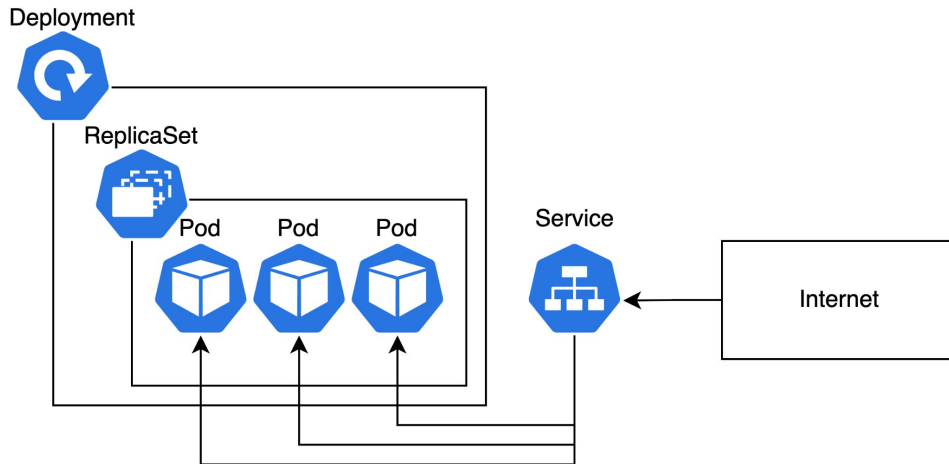
- Sposób wystawienia działającej aplikacji działającej na *n Pod*'ach jako serwis sieciowy
- Stabilna abstrakcja networkingu (posiada DNS, adres IP i port) na efemeryczne *Pod*'y
- Dostarcza load-balancing między *Pod*'ami
- 4 typy
 - *ClusterIP* (dostępny tylko wewnątrz clustra)
 - *NodePort* (wystawia serwis na statycznym porcie node'a)
 - *LoadBalancer* (wystawia serwis używając load balancera)
 - *ExternalName* (zwraca *Canonical Name Record - redirect*)

Uwaga:

Nie potrzebujemy używać typu `LoadBalancer` aby balansować ruch w samym k8s. Mając wystawiony `Service` który obsługuje *n Pod*'ów sam k8s wykorzystuje kube-proxy aby rozdzielać workload. `LoadBalancer` jest potrzebny do współpracy z zewnętrznym load balancerem.

Deployment

- Podstawowy obiekt z którym pracujemy podczas obsługi k8s
- Jest wrapperem na *Pod*'y
- Opisujemy pożądany stan - *Deployment Controller* zmienia aktualny stan tak aby osiągnąć stan pożądany (poprzez *ReplicaSet*, który tworzy i usuwa *Pod*'y)
- Dostarcza self-healing, scaling, rolling-update, rollback





ConfigMap

- Prosta konfiguracja klucz-wartość
- Używana do przechowywania danych niewrażliwych
- Pozwala na decoupling konfiguracji od zbudowanego obrazu
- Nie zapewnia szyfrowania (encryption) czy kodowania (encoding)
- Nie jest przeznaczona do przechowywania dużych danych - max 1 MiB (mebibajt)
- Istnieją 4 sposoby wykorzystania:
 - Komendy wewnątrz kontenera
 - **Zmienna środowiskowa dla kontenera**
 - **Dodanie pliku jako do kontenera jako volume**
 - Napisanie kodu wewnątrz *Pod*a korzystającego z Kubernetes API



Secret

- Odpowiednik *Configmap* dla danych wrażliwych
- Zawiera małą ilość danych takich jak hasła, tokeny, klucze
- Wartości są enkodowane base64
- Domyślnie wartości przechowywane są na storage'u clustra w formie nie szyfrowanej. Każda osoba mająca dostęp do *Secret*'u może odczytać jego wartość (enkrypcja na REST jest opt-in)

Uwaga:

Samo stworzenie *Secretu* nie czyni go bezpieczniejszym od *Configmap*'y. Istnieją projekty pozwalające korzystać z *Secret*'ów w sposób bezpieczny, lecz nie są one integralną częścią k8s.

Demo



Repozytorium: github.com/jorczyk/KubernetesDemo

Uruchomienie lokalnego clustra używając Minikube

Uruchomienie clustra minikube: `minikube start`

```
🤪 minikube v1.24.0 na Darwin 12.3
🌟 Using the docker driver based on existing profile
👍 Starting control plane node minikube in cluster minikube
🚚 Pulling base image ...
🔄 Restarting existing docker container for "minikube" ...
🌊 Przygotowywanie Kubernetesa v1.22.3 na Docker 20.10.8...
🔍 Verifying Kubernetes components...
  ▪ Using image kubernetesui/dashboard:v2.3.1
  ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
  ▪ Using image kubernetesui/metrics-scraper:v1.0.7
☀️ Enabled addons: storage-provisioner, default-storageclass, dashboard
🏊 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Uruchomienie dashboardu: `minikube dashboard`



Zbudowanie obrazu dockera

Ustawienie użycia daemona dockera z minikube: `eval $(minikube docker-env)`

Sprawdzenie obrazów dockera: `docker images`

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
k8s-demo-08-3	latest	f9a02a80ef7a	2 weeks ago	677MB
openjdk	11	6424f22b558a	5 weeks ago	660MB
k8s.gcr.io/kube-apiserver	v1.22.3	53224b502ea4	5 months ago	128MB
k8s.gcr.io/kube-controller-manager	v1.22.3	05c905cef780	5 months ago	122MB
k8s.gcr.io/kube-scheduler	v1.22.3	0aa9c7e31d30	5 months ago	52.7MB
k8s.gcr.io/kube-proxy	v1.22.3	6120bd723dce	5 months ago	104MB
kubernetesui/dashboard	v2.3.1	e1482a24335a	9 months ago	220MB
k8s.gcr.io/etcd	3.5.0-0	004811815584	9 months ago	295MB
kubernetesui/metrics-scraper	v1.0.7	7801cfc6d5c0	10 months ago	34.4MB
k8s.gcr.io/coredns/coredns	v1.8.4	8d147537fb7d	10 months ago	47.6MB
gcr.io/k8s-minikube/storage-provisioner	v5	6e38f40d628d	12 months ago	31.5MB
k8s.gcr.io/pause	3.5	ed210e3e4a5b	13 months ago	683kB



Zbudowanie obrazu dockera (2)

Budowanie wykonywalnego jar'a: `./gradlew build`

Zbudowanie docker image: `docker build -t k8s-demo-live .`

Ponowne sprawdzenie obrazów: `docker images`



Uruchomienie obrazu w sposób imperatywny

Uruchomienie obrazu imperatywnie: `kubectl run demo-app --image=k8s-demo-02`

Sprawdzenie istniejących *Pod*ów: `kubectl get pods`

NAME	READY	STATUS	RESTARTS	AGE
demo-app	0/1	ErrImagePull	0	9s

Co poszło nie tak?

Describe *Pod'a*: `kubectl describe pod <pod_name>`

```
Name:          demo-app
Status:        Pending

Containers:
  demo-app:
    Container ID:
    Image:       k8s-demo-02
    State:       Waiting
    Reason:      ErrImagePull
    Ready:       False
Conditions:
  Type            Status
  Initialized      True
  Ready           False
  ContainersReady  False
  PodScheduled     True

Events:
  Type    Reason      Age           From          Message
  ----    -
  Normal  Scheduled   34s           default-scheduler  Successfully assigned default/demo-app to minikube
  Normal  Pulling     17s (x2 over 34s)  kubelet          Pulling image "k8s-demo-02"
  Warning  Failed      15s (x2 over 31s)  kubelet          Failed to pull image "k8s-demo-02": rpc error: code = Unknown desc = Error
response from daemon: pull access denied for k8s-demo-02, repository does not exist or may require 'docker login': denied: requested
access to the resource is denied
  Warning  Failed      15s (x2 over 31s)  kubelet          Error: ErrImagePull
  Normal  BackOff     0s (x2 over 31s)  kubelet          Back-off pulling image "k8s-demo-02"
  Warning  Failed      0s (x2 over 31s)  kubelet          Error: ImagePullBackOff
```



Ustawienie flagi image-pull-policy

Usunięcie *Pod'a*: `kubectl delete pod <pod_name>`

Trzy możliwe ustawienia flagi image-pull-policy: `Never`, `IfNotPresent`, `Always`

Uruchomienie obrazu z image-pull-policy:

```
kubectl run demo-app --image=k8s-demo-02 --image-pull-policy=Never
```

Ustawienie flagi image-pull-policy (2)

Sprawdzenie *Pod*'ów: `kubectl get pods`

NAME	READY	STATUS	RESTARTS	AGE
demo-app	1/1	Running	0	5s

Sprawdzenie logów z *Pod'a*: `kubectl logs demo-app`

```

/\ \ / ____' _(_)' _ _\ \ \ \ 
( ( ) \____|'_ _'|_ _'\/_ _|\ \ \ \ \ 
\\ / ____)|_|)| | | | | | (_| | ) ) ) 
'| ____| ._|_| |_|_| |_)\_, | / / / / 
=====|_|=====|___/=/_/_/_/ 
:: Spring Boot ::                (v2.6.4)

```

```
2022-04-11 10:22:06.704 INFO 1 --- [main] c.m.p.k.KubernetesDemoApplication :
Starting KubernetesDemoApplication using Java 11.0.14.1 on demo-app with PID 1 (/app.jar started by root
in /)
...

```

Wystawienie serwisu na zewnętrzny ruch

Wystawienie demo-app na localhost imperatywnie:

```
kubectl expose pod demo-app --port=8080 --type=LoadBalancer  
--name=demo-service
```


Alternatywnie można użyć typu: `--type=NodePort`

Sprawdzenie czy serwis ma przypisany EXTERNAL-IP: `kubectl get service -w`

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
demo-service	LoadBalancer	10.102.18.152	<pending>	8080:31438/TCP	48s
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	19d

Co poszło nie tak?

Uruchomienie tunelowania z minikube (należy to zrobić w nowym terminalu) aby umożliwić działanie kubernetes'owego Load Balancera: `minikube tunnel`

 Starting tunnel for service demo-service.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
demo-service	LoadBalancer	10.102.18.152	<pending>	8080:31438/TCP	48s
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	19d
demo-service	LoadBalancer	10.102.18.152	127.0.0.1	8080:31438/TCP	101s

Uderzenie na endpoint: `curl --request GET --url '127.0.0.1:8080';echo`

Usunięcie wszystkich zasobów: `kubectl delete all --all -n default`



Uruchomienie obrazu w sposób deklaratywny

Utworzenie *Deployment*'u i *Service*'u: `kubectl apply -f <file name>`

Sprawdzenie wszystkich* zasobów na klustrze: `kubectl get all`

NAME	READY	STATUS	RESTARTS	AGE
pod/demo-app-deployment-756fd8bcc5-459c8	1/1	Running	0	61s
pod/demo-app-deployment-756fd8bcc5-8ssmb	1/1	Running	0	61s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/demo-app-service	LoadBalancer	10.100.222.22	127.0.0.1	9376:32074/TCP	4s
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	9m57s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/demo-app-deployment	2/2	2	2	61s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/demo-app-deployment-756fd8bcc5	2	2	2	61s



Utrzymanie stanu zadanego

```
$ kubectl delete pod <pod name>
```

NAME	READY	STATUS	RESTARTS	AGE
demo-app-deployment-756fd8bcc5-8ssmb	1/1	Running	0	3m4s
demo-app-deployment-756fd8bcc5-k8fkr	1/1	Running	0	7s



Skalowanie w dół

Po zmianie replica count w pliku deployment.yaml do wartości 1

```
$ kubectl apply -f deployment.yaml  
deployment.apps/demo-app-deployment configured
```

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
demo-app-deployment-756fd8bcc5-8ssmb	1/1	Running	0	5m30s



Skalowanie w górę

Po zmianie replica count w pliku deployment.yaml do wartości 3

```
$ kubectl apply -f deployment.yaml  
deployment.apps/demo-app-deployment configured
```

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
demo-app-deployment-756fd8bcc5-8ssmb	1/1	Running	0	6m41s
demo-app-deployment-756fd8bcc5-cv2z5	1/1	Running	0	3s
demo-app-deployment-756fd8bcc5-fgxqp	1/1	Running	0	3s



Load balancing

```
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
demo-app-deployment-7dbc4877d9-5xjgr 1/1     Running   0           72s
demo-app-deployment-7dbc4877d9-hgmfv 1/1     Running   0           70s
demo-app-deployment-7dbc4877d9-nkflg 1/1     Running   0           72s
$ curl --request GET --url '127.0.0.1:9376/host';echo
demo-app-deployment-7dbc4877d9-nkflg
$ curl --request GET --url '127.0.0.1:9376/host';echo
demo-app-deployment-7dbc4877d9-hgmfv
$ curl --request GET --url '127.0.0.1:9376/host';echo
demo-app-deployment-7dbc4877d9-5xjgr
$ curl --request GET --url '127.0.0.1:9376/host';echo
demo-app-deployment-7dbc4877d9-5xjgr
```

Rolling update

Po zmianie obrazu w *Deployment*'cie i zaaplikowaniu zmian

```
$ kubectl get pods -w
```

NAME	READY	STATUS	RESTARTS	AGE
demo-app-deployment-7dbc4877d9-gx9mm	1/1	Running	0	91s
demo-app-deployment-7dbc4877d9-qkzts	1/1	Running	0	39s
demo-app-deployment-7dbc4877d9-ztkfd	1/1	Running	0	91s
demo-app-deployment-9876cbb7d-wv9gk	0/1	ContainerCreating	0	1s
demo-app-deployment-9876cbb7d-zcc92	0/1	ContainerCreating	0	1s
demo-app-deployment-9876cbb7d-zcc92	1/1	Running	0	2s
demo-app-deployment-9876cbb7d-wv9gk	1/1	Running	0	2s
demo-app-deployment-7dbc4877d9-qkzts	1/1	Terminating	0	40s
demo-app-deployment-9876cbb7d-bd54f	0/1	Pending	0	0s
demo-app-deployment-7dbc4877d9-ztkfd	1/1	Terminating	0	93s
demo-app-deployment-9876cbb7d-bd54f	0/1	ContainerCreating	0	0s
demo-app-deployment-7dbc4877d9-qkzts	0/1	Terminating	0	44s
demo-app-deployment-7dbc4877d9-ztkfd	0/1	Terminating	0	96s
demo-app-deployment-9876cbb7d-bd54f	1/1	Running	0	3s
demo-app-deployment-7dbc4877d9-gx9mm	1/1	Terminating	0	96s
demo-app-deployment-7dbc4877d9-gx9mm	0/1	Terminating	0	99s

Usunięcie wszystkich zasobów: `kubectl delete all --all -n default`

Użycie zewnętrznej konfiguracji

Wgranie wszystkich plików z folderu: `kubectl apply -f 06-configmap/`

Wejście w kontener: `kubectl exec <pod name> -it -- /bin/bash`

```
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
demo-app-deployment-5867745bbd-gql5h 1/1     Running   0           57s

$ kubectl exec demo-app-deployment-5867745bbd-gql5h -it -- /bin/bash
root@demo-app-deployment-5867745bbd-gql5h:/# ls config
films.properties
root@demo-app-deployment-5867745bbd-gql5h:/# cat config/films.properties
meaning.of.life=monty-python
something.completely.different=monty-python
root@demo-app-deployment-5867745bbd-gql5h:/# printenv sense.of.life
42
```

```
$ curl --request GET --url '127.0.0.1:9376/props';echo
Sense of life: 42
Loaded properties file: {meaning.of.life=monty-python, something.completely.different=monty-python}
```



Stworzenie zewnętrznej konfiguracji

Utworzenie z folderu:

```
kubectl create configmap dir-cm-example --from-file=./cm-files/
```

Sprawdzenie *ConfigMap*'y: `kubectl describe configmap dir-cm-example`

Uwaga:

Dane które nie są w ASCII lub UTF-8, zostaną umieszczone w sekcji `binaryData`



Stworzenie zewnętrznej konfiguracji (2)

Utworzenie z pliku:

```
kubectl create configmap file-cm-example --from-file=<file path>
```

Utworzenie z pojedynczych wartości:

```
kubectl create configmap literals-cm-example  
--from-literal=first.key=value1 --from-literal=second.key=value2
```

Zapisanie stworzonej *configMap*'y do pliku yaml:

```
kubectl get configmap literals-cm-example -o yaml >  
generated-configmap.yaml
```




Użycie Readiness probe

Wgranie wszystkich plików z folderu: `kubectl apply -f 08-liveness\&readiness/`

```
$ kubectl get pods -w
```

NAME	READY	STATUS	RESTARTS	AGE
demo-app-deployment-55d84fc556-wv48p	0/1	Running	0	4s

Po 20 sekundach:

```
$ kubectl get pods -w
```

NAME	READY	STATUS	RESTARTS	AGE
demo-app-deployment-55d84fc556-wv48p	0/1	Running	0	4s
demo-app-deployment-55d84fc556-wv48p	1/1	Running	0	20s

Użycie Liveness probe

Użycie endpoint'u aby ustawić status na unhealthy:

```
curl --request POST --url '127.0.0.1:9376/setUnhealthy';echo
```

```
$ kubectl get pods -w
```

NAME	READY	STATUS	RESTARTS	AGE
demo-app-deployment-547f6c6d88-hsjj6	1/1	Running	0	7m21s
demo-app-deployment-547f6c6d88-hsjj6	0/1	Running	1 (1s ago)	7m26s
demo-app-deployment-547f6c6d88-hsjj6	1/1	Running	1 (20s ago)	7m45s

Sprawdzenie logów *Pod'a* w czasie rzeczywistym:

```
kubectl logs demo-app-deployment-b96d94779-wn97 --follow
```

Sprawdzenie logów ostatnio używanego *Pod'a* (*Pod'a* który został zrestartowany):

```
kubectl logs demo-app-deployment-b96d94779-wn97 --previous
```

Uwaga:

K8s traktuje kody odpowiedzi 2xx i 3xx jako kody *healthy*. Wszystkie inne kody odpowiedzi traktowane są jako *unhealthy*



Materiały do dalszej nauki

Dokumentacja: <https://kubernetes.io/docs/home/>

K8s API reference: <https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.23/#deployment-v1-apps>

Komendy kubectl: <https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands>

Kubectl cheatsheet: <https://kubernetes.io/docs/reference/kubectl/cheatsheet/>

Książki:

The Kubernetes Book: 2022 Edition - Nigel Poulton

Kubernetes in Action 1st Edition (albo 2 wydanie gdy będzie już dostępne) - Marko Luksa

Kubernetes: Up and Running: Dive into the Future of Infrastructure - Brendan Burns, Joe Beda , Kelsey Hightower

Youtube:

TechWorld with Nana: <https://www.youtube.com/channel/UCdngmbVKX1Tgre699-XLIUA>

Dziękuję za uwagę :)

