

InzynieriaOprogramowaniaAGH /
MDO2023_INO

<> Code

Issues

Pull requests

Actions

Projects

Security



This repository has been archived by the owner on Sep 15, 2023. It is now read-only.

MDO2023_INO / INO / GCL2 / KP406287 / lab10i11 / Sprawozdanie.md



qamilciaq1 Final KP406287

9 months ago



137 lines (69 loc) · 10.4 KB

Preview

Code

Blame

Raw



Instalacja klastra Kubernetes

Aby zaopatrzyć się w minikube na moim urządzeniu, skorzystałam z dokumentacji dostępnej pod adresem: <https://minikube.sigs.k8s.io/docs/start/>. Zgodnie z tą oficjalną instrukcją, pobrałam właściwy plik instalacyjny. Następnie postąpiłam zgodnie z instrukcjami dostosowanymi do mojego systemu operacyjnego. Dodatkowo, upewniłam się, że mam zainstalowane i gotowe do użycia środowisko Docker.

```
kamila@ubuntu:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left     Speed
100 80.0M  100 80.0M    0     0  495k      0  0:02:45  0:02:45 --:--:-- 1517k
kamila@ubuntu:~$
```

Po zakończeniu procesu instalacji, w terminalu użyłam polecenia "minikube start", aby uruchomić Minikube i skonfigurować klaster Kubernetes na moim urządzeniu.

```
kamila@ubuntu:~$ minikube start
minikube v1.30.1 on Ubuntu 22.04
Automatically selected the docker driver. Other choices: none, ssh
Using Docker driver with root privileges
Starting control plane node minikube in cluster minikube
Pulling base image ...
Downloading Kubernetes v1.26.3 preload ...
> preloaded-images-k8s-v18-v1...: 397.02 MiB / 397.02 MiB 100.00% 13.22 M
> gcr.io/k8s-minikube/kicbase...: 373.53 MiB / 373.53 MiB 100.00% 7.05 Mi
Creating docker container (CPUs=2, Memory=2200MB) ...
```

```

Preparing Kubernetes v1.26.3 on Docker 23.0.2 ...
  ■ Generating certificates and keys ...
  ■ Booting up control plane ...
  ■ Configuring RBAC rules ...
Configuring bridge CNI (Container Networking Interface) ...
  ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
Verifying Kubernetes components...
Enabled addons: default-storageclass, storage-provisioner
kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
kamila@ubuntu:~$

```

Jego działanie można potwierdzić wykonując polecenie `docker ps`.

```

kamila@ubuntu:~$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
STATUS        PORTS
3ebc06ea38a9   kicbase/stable:v0.0.39             "/usr/local/bin/entr..." 30 minutes ago
Up 2 minutes   127.0.0.1:32772->22/tcp, 127.0.0.1:32771->2376/tcp, 127.0.0.1:32770->5000/tcp, 127.0.0.1:32769->8443/tcp, 127.0.0.1:32768->32443/tcp minikube

```

Następnie przesłałam do uruchomienia Kubernetesa:

Przy użyciu komendy `"kubectl get po -A"` mogę wyświetlić listę wszystkich aktywnych podów. To polecenie zapewnia szybką możliwość sprawdzenia, które pod-y są obecnie uruchomione i funkcjonują w klastrze Kubernetes

```

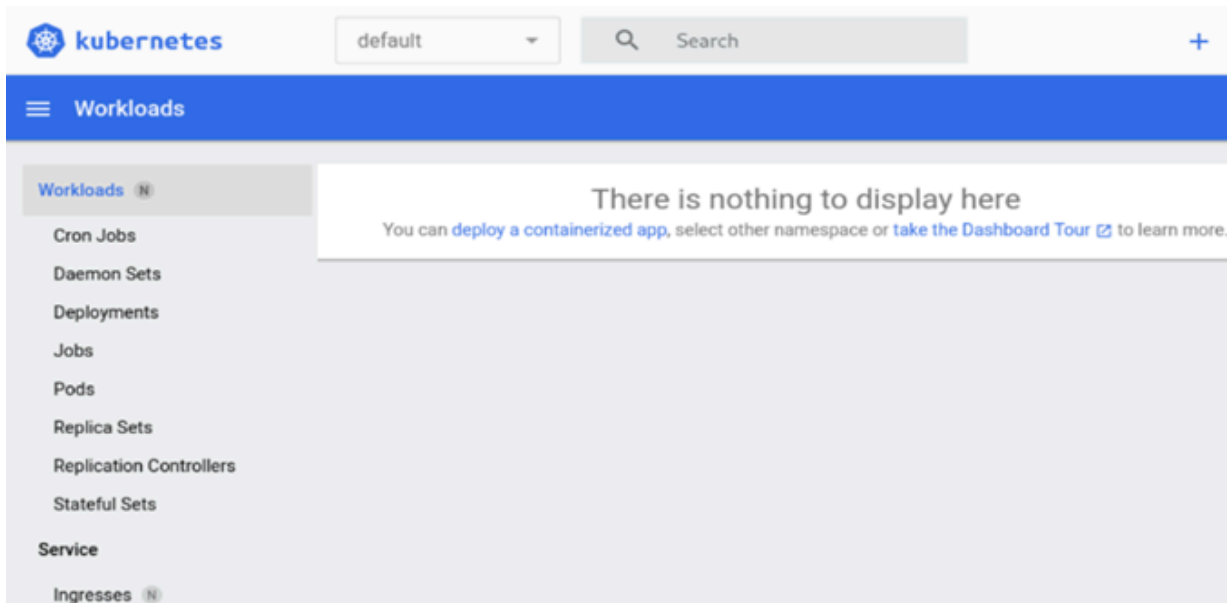
kamila@ubuntu:~$ kubectl get po -A
NAMESPACE     NAME                                READY   STATUS    RESTARTS
AGE
kube-system   coredns-787d4945fb-nw2jr           1/1     Running   1 (7m8s ago)
31m
kube-system   etcd-minikube                      1/1     Running   1 (7m8s ago)
32m
kube-system   kube-apiserver-minikube            1/1     Running   1 (7m8s ago)
32m
kube-system   kube-controller-manager-minikube    1/1     Running   2 (7m8s ago)
32m
kube-system   kube-proxy-24fmr                   1/1     Running   1 (7m8s ago)
31m
kube-system   kube-scheduler-minikube            1/1     Running   1 (7m8s ago)
32m
kube-system   storage-provisioner                1/1     Running   3 (4m56s ago)
31m

```

Poprzez wykonanie polecenia `"minikube dashboard"`, otwieram interaktywny pulpit nawigacyjny, który dostarcza wygodny interfejs do zarządzania i monitorowania klastra Kubernetes działającego lokalnie. Dzięki temu narzędziu zyskuję łatwy dostęp do wizualnej reprezentacji mojego klastra Kubernetes, co ułatwia zarządzanie i monitorowanie działających aplikacji w lokalnym środowisku.

```
kamila@ubuntu:~$ minikube dashboard
Enabling dashboard ...
Using image docker.io/kubernetes/metrics-scraper:v1.0.8
Using image docker.io/kubernetes/dashboard:v2.7.0
Some dashboard features require the metrics-server addon. To enable all features pl
minikube addons enable metrics-server
Verifying dashboard health ...
Launching proxy ...
Verifying proxy health ...
Opening http://127.0.0.1:37369/api/v1/namespaces/kubernetes-dashboard/services/http
:kubernetes-dashboard:/proxy/ in your default browser...
```

Automatycznie otwiera się okno przeglądarki z Kubernetesem:



Z uwagi na występujące błędy w poprzedniej wersji aplikacji, zdecydowano się na zmianę obrazu na nowszy wariant nginx. W pierwszym kroku został utworzony plik Dockerfile, który posłużył jako podstawa do zbudowania nowego obrazu.

```
kamila@ubuntu: ~
GNU nano 6.2 nginx-app
FROM nginx:latest
```

```
kamila@ubuntu:~$ docker build -t nginx-app .  
[+] Building 39.2s (5/5) FINISHED
```

Uruchomiono kontener.

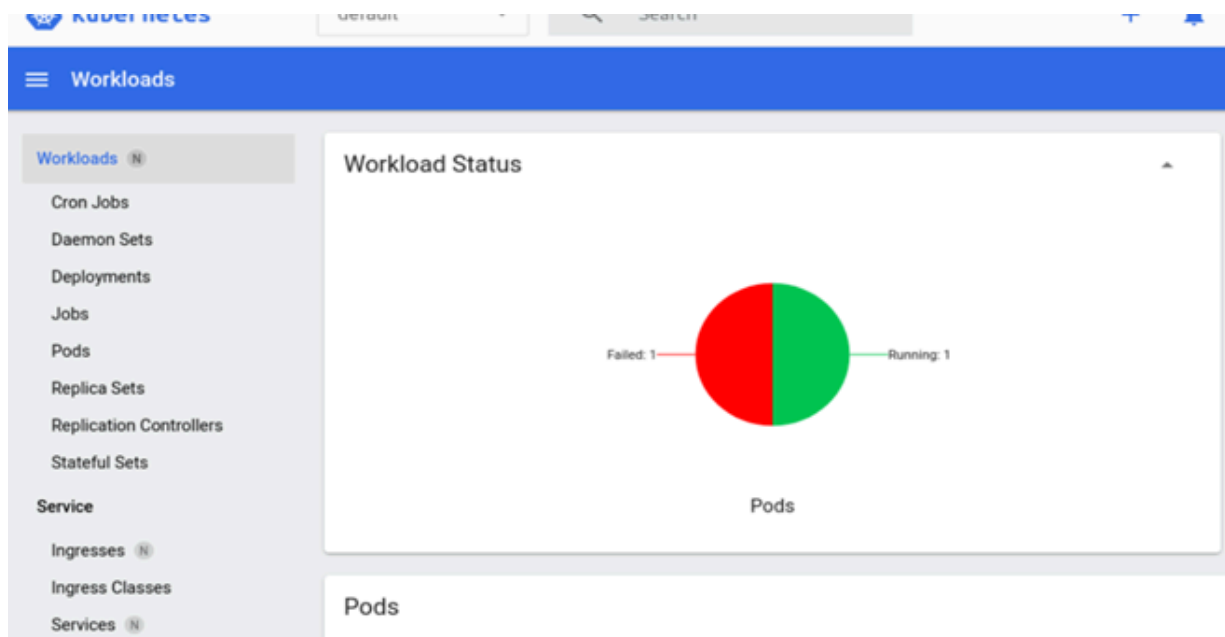
```
kamila@ubuntu:~$ docker run -p 80:80 -d nginx-app  
f6344b8cb1c5aa6c296cfafe2062c8162e741c58103e811953d7b7e2e7b780d4  
kamila@ubuntu:~$
```

```
kamila@ubuntu:~$ docker ps  
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS  
PORTS         NAMES  
f6344b8cb1c5   nginx-app "/docker-entrypoint..." 2 minutes ago  Up 2 minutes  
0.0.0.0:80->80/tcp, :::80->80/tcp  sweet_mestorf
```

W pierwszej próbie uruchomienia kontenera z nginx w klastrze Kubernetes wystąpiły trudności, co skutkowało niepowodzeniem. Jednakże, podjęłam kolejną próbę, tym razem korzystając z Dockerhuba jako źródła obrazu, co okazało się skuteczne. Pierwsza próba zakończyła się porażką prawdopodobnie ze względu na nieprawidłowe skonfigurowanie kontenera lub inną kwestię techniczną, która utrudniała poprawne uruchomienie aplikacji. Dopiero druga próba, korzystając z gotowego obrazu z Dockerhuba, przyniosła oczekiwane rezultaty, co może wskazywać na poprawność obrazu lub bardziej odpowiednie ustawienia konfiguracyjne.

```
kamila@ubuntu:~$ kubectl run nginx-web --image=nginx:1.23.4 --port=9999 --label  
s app=nginx-web  
pod/nginx-web created
```

Rezultat:

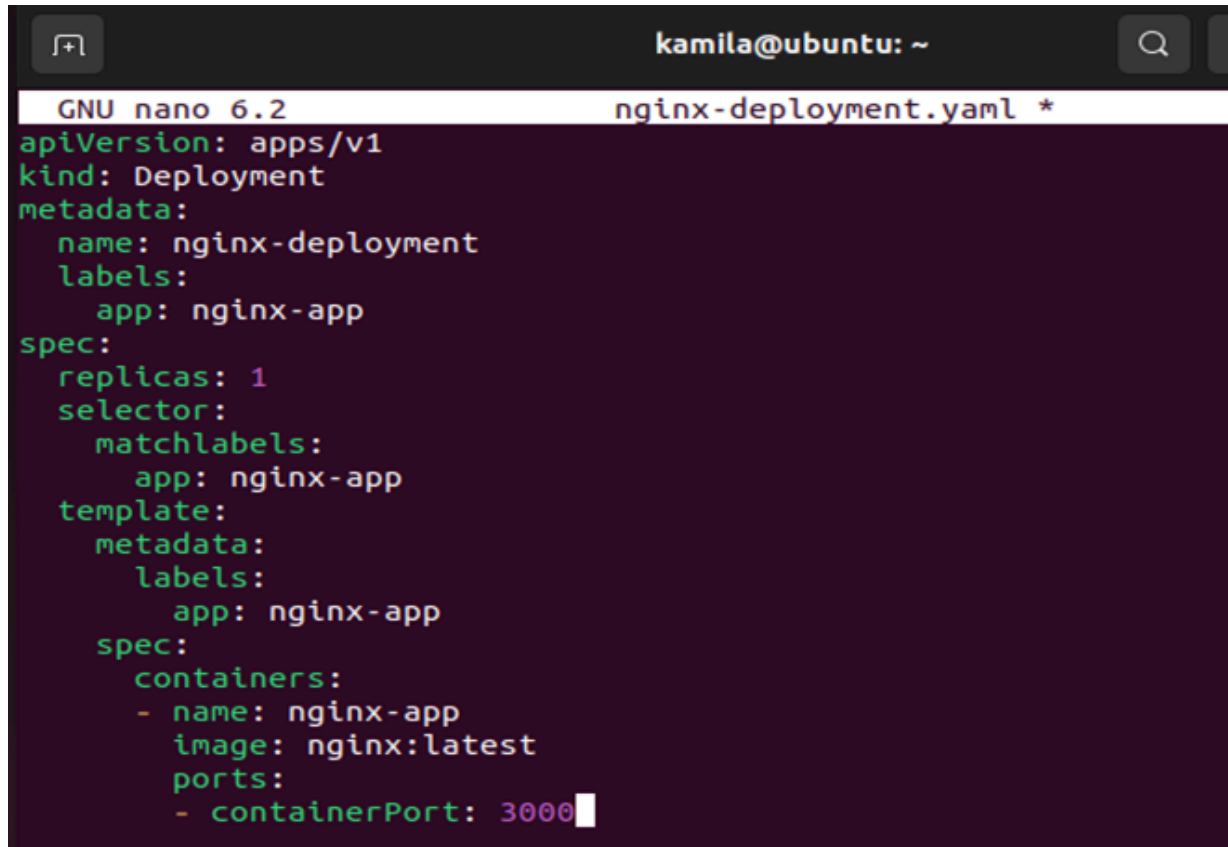


Wdrożenie

Kolejnym krokiem było utworzenie pliku wdrażającego w formacie YAML. Plik ten zawiera konfigurację potrzebną do uruchomienia aplikacji w klastrze Kubernetes.

```
kamila@ubuntu:~$ touch nginx-deployment.yaml
```

Zawartość pliku:

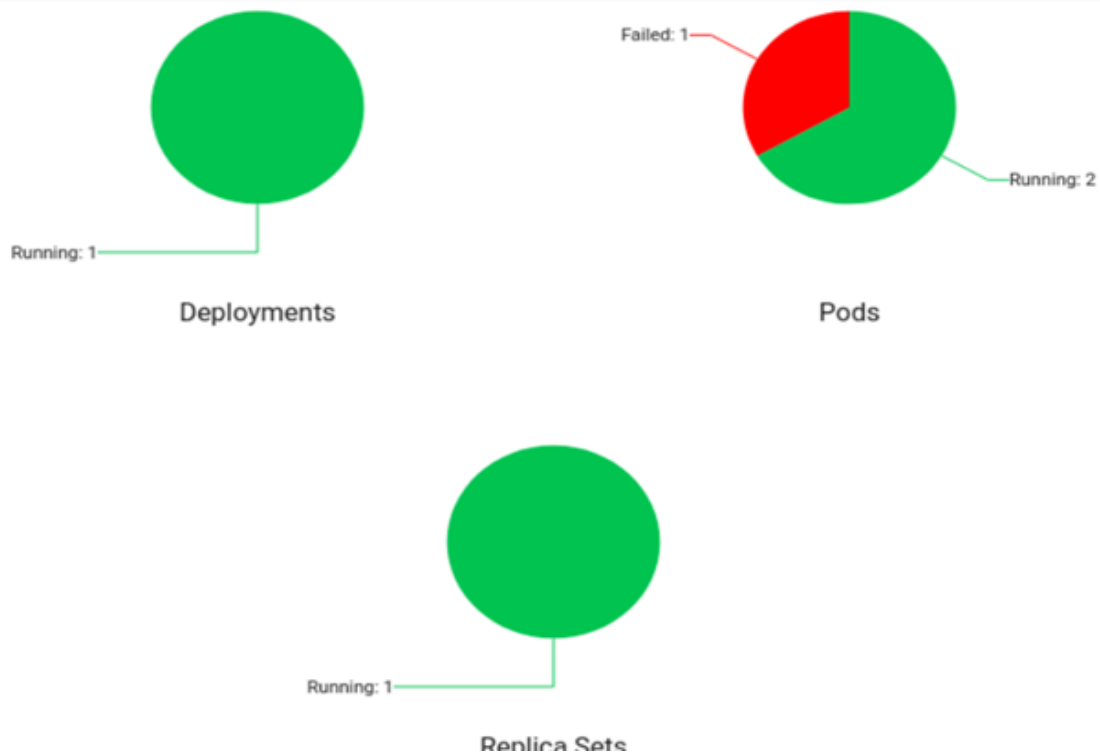


```
kamila@ubuntu: ~
GNU nano 6.2 nginx-deployment.yaml *
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-app
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
      - name: nginx-app
        image: nginx:latest
        ports:
        - containerPort: 3000
```

Rozpoczęto proces wdrażania przy użyciu komendy "kubectl apply", która pozwala na zastosowanie konfiguracji z pliku YAML na klastrze Kubernetes. Następnie, w celu sprawdzenia stanu wdrożenia, skorzystano z polecenia "kubectl rollout status". To polecenie umożliwia monitorowanie postępu wdrożenia, sprawdzając, czy wszystkie zasoby zostały poprawnie zastosowane i czy aplikacja jest w stanie działania. Dzięki temu można śledzić status wdrażania i upewnić się, czy wszystkie kroki przebiegają pomyślnie

```
kamila@ubuntu:~$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment configured
kamila@ubuntu:~$ kubectl rollout status deployment/nginx-deployment
deployment "nginx-deployment" successfully rolled out
```

Rezultat:



Zmieniłam liczbę replik na 10:

```
kamila@ubuntu: ~  
GNU nano 6.2 nginx-deployment.yaml *  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: nginx-deployment  
  labels:  
    app: nginx-app  
spec:  
  replicas: 10  
  selector:  
    matchLabels:  
      app: nginx-app  
  template:  
    metadata:  
      labels:  
        app: nginx-app  
    spec:  
      containers:  
      - name: nginx-app  
        image: nginx:latest  
        ports:  
        - containerPort: 3000
```

Rezultat:



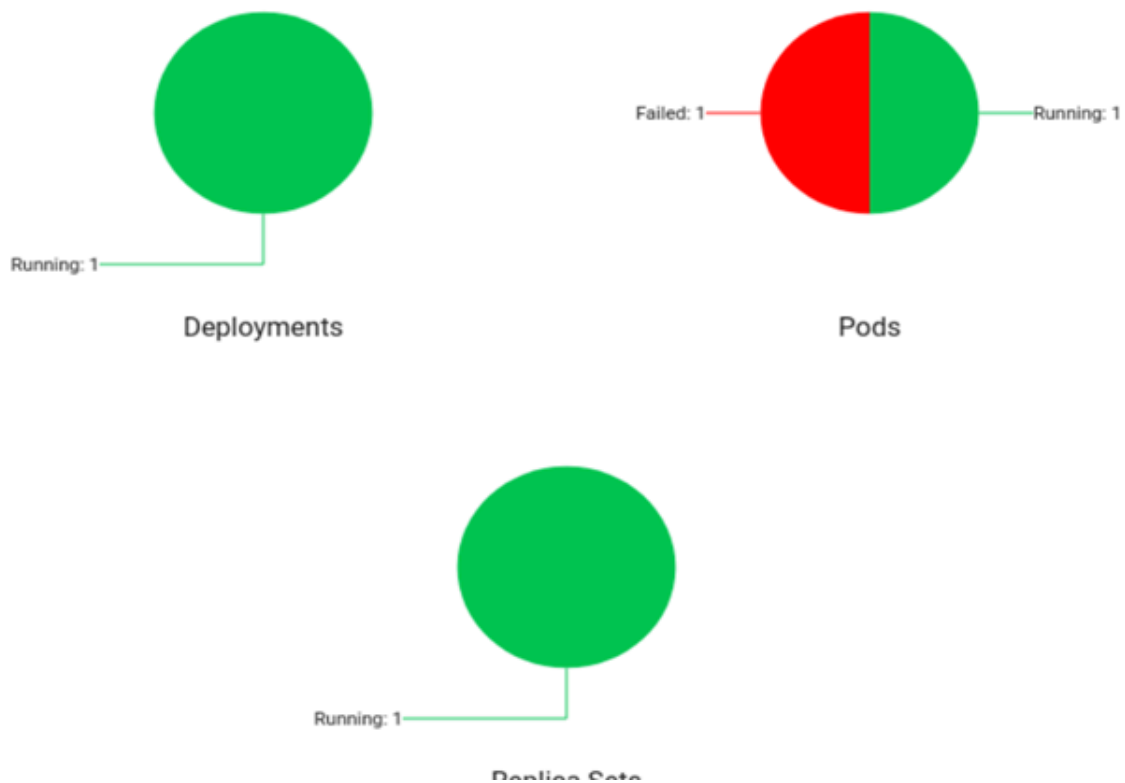
Teraz ustawiam liczbę replik na 1:

```
kamila@ubuntu: ~  
GNU nano 6.2 nginx-deployment.yaml *  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: nginx-deployment  
  labels:  
    app: nginx-app  
spec:  
  replicas: 1  
  selector:  
    matchLabels:  
      app: nginx-app  
  template:  
    metadata:  
      labels:  
        app: nginx-app  
    spec:  
      containers:  
      - name: nginx-app  
        image: nginx:latest  
        ports:  
        - containerPort: 3000
```

I na 0 oraz powtarzam komendy:

```
kamila@ubuntu: ~  
GNU nano 6.2 nginx-deployment.yaml *  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: nginx-deployment  
  labels:  
    app: nginx-app  
spec:  
  replicas: 0  
  selector:  
    matchLabels:  
      app: nginx-app  
  template:  
    metadata:  
      labels:  
        app: nginx-app  
    spec:  
      containers:  
      - name: nginx-app  
        image: nginx:latest  
        ports:  
        - containerPort: 3000
```

Rezultat:



	Name	Images	Labels	Node	
Cron Jobs					
Daemon Sets					
Deployments					
Jobs					
Pods					
Replica Sets					
Replication Controllers					
Stateful Sets					
Service					
	 nginx-deployment-912345834-7585b46c-xd72p			minikube	
	 nginx-deployment-98232445-7bbc77796-pnfhq			minikube	
	 nginx-deployment-5b87329834-cfdff88c-bgs44			minikube	


Przygotowanie nowego obrazu

Po wprowadzeniu zmian do pliku Dockerfile, mogę zaktualizować plik wdrażania deploy.yaml, aby podmienić obraz na nowo utworzony. W pliku deploy.yaml znajduje się sekcja, która określa obraz, który zostanie użyty podczas wdrażania aplikacji. Muszę zmienić wartość obrazu, aby odzwierciedlała nazwę lub tag nowego obrazu, który został utworzony na podstawie zmodyfikowanego Dockerfile.

```
kamila@ubuntu: ~  
GNU nano 6.2 nginx-deployment.yaml  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: nginx-deploy  
  labels:  
    app: nginx-deploy  
spec:  
  replicas: 1  
  selector:  
    matchLabels:  
      app: nginx-deploy  
  template:  
    metadata:  
      labels:  
        app: nginx-deploy  
    spec:  
      containers:  
      - name: nginx-deploy  
        image: nginx:test  
        ports:  
        - containerPort: 3000
```

Rezultat po uruchomienie wdrożenia:

Deployments

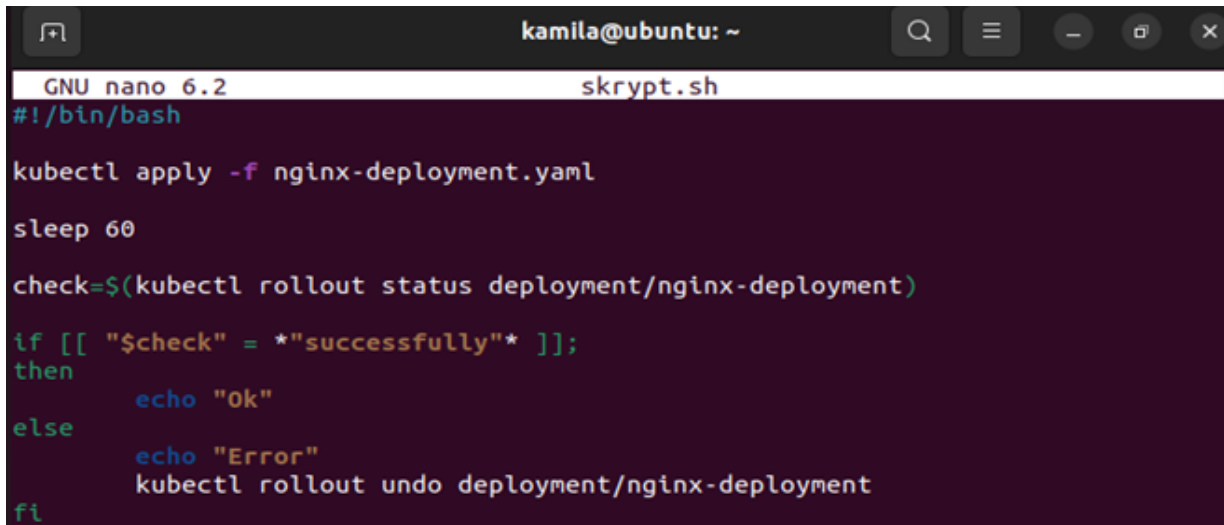
Name	Images	Labels
 nginx-deploy	nginx:test	app: nginx-deploy

Następnie przywrócono poprzednią wersję. Cały czas edytowałam jeden i ten sam plik.

```
kamila@ubuntu:~$ kubectl rollout history deployment/nginx-deployment  
deployment.apps/nginx-deployment  
REVISION  CHANGE-CAUSE  
1          <none>  
  
kamila@ubuntu:~$ kubectl rollout status -f nginx-deployment.yaml  
deployment "nginx-deployment" successfully rolled out  
kamila@ubuntu:~$
```

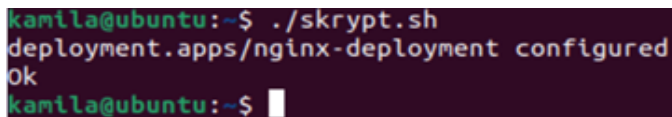
Kontrola wdrożenia

Napisałam skrypt, który ma za zadanie weryfikować, czy wdrożenie zadziałało w czasie do 60s.



```
kamila@ubuntu: ~  
GNU nano 6.2 skrypt.sh  
#!/bin/bash  
  
kubectl apply -f nginx-deployment.yaml  
  
sleep 60  
  
check=$(kubectl rollout status deployment/nginx-deployment)  
  
if [[ "$check" = *"successfully"* ]];  
then  
    echo "ok"  
else  
    echo "Error"  
    kubectl rollout undo deployment/nginx-deployment  
fi
```

Skrypt przeszedł pomyślnie.



```
kamila@ubuntu:~$ ./skrypt.sh  
deployment.apps/nginx-deployment configured  
ok  
kamila@ubuntu:~$
```

Strategie wdrożenia

W ostatniej części sprawozdania przystąpiono do przygotowania wersji wdrożeń, wykorzystując trzy strategie: Recreate, Rolling Update oraz Canary Deployment workload. W przypadku strategii Recreate utworzono plik rec.yaml, który reprezentował tę strategię wdrożenia. Strategia Recreate polega na zatrzymaniu i usunięciu nieaktualnych instancji aplikacji, a następnie uruchomieniu ich na nowej wersji. W pliku rec.yaml określono konfigurację, która skutkowałą zakończeniem działania poprzednich replik i zastąpieniem ich nowymi replikami na nowszej wersji aplikacji. Ta strategia zapewnia ciągłość odnawiania stanu aplikacji, ponieważ żadne nieaktualne instancje nie pozostają aktywne po zakończeniu wdrożenia.

```
kamila@ubuntu: ~  
GNU nano 6.2 rec.yaml  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: nginx-deployment  
  labels:  
    app: nginx-app  
spec:  
  strategy:  
    type: Recreate  
  replicas: 4  
  selector:  
    matchLabels:  
      app: nginx-app  
  template:  
    metadata:  
      labels:  
        app: nginx-app  
    spec:  
      containers:  
      - name: nginx-app  
        image: nginx:latest  
        ports:  
        - containerPort: 3000  
  
kamila@ubuntu:~$ kubectl apply -f rec.yaml  
deployment.apps/nginx-deployment configured  
kamila@ubuntu:~$ kubectl rollout status -f rec.yaml  
deployment "nginx-deployment" successfully rolled out  
kamila@ubuntu:~$
```

Strategia Rolling Update polega na stopniowym zastępowaniu starych instancji aplikacji nowymi, co minimalizuje przerwę w dostępności aplikacji podczas aktualizacji. Jest to preferowana strategia, gdy niezbędna jest ciągła dostępność aplikacji. W pliku roll.yaml określono konfigurację, która umożliwia stopniową aktualizację aplikacji. Proces ten polega na stopniowym usuwaniu starych replik i jednoczesnym uruchamianiu nowych replik na nowej wersji. Dzięki temu użytkownicy mają ciągły dostęp do aplikacji, ponieważ w danym momencie zawsze jest dostępna co najmniej jedna replika.

```
kamila@ubuntu: ~  
GNU nano 6.2 roll.yaml  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: nginx  
  labels:  
    app: nginx  
spec:  
  strategy:  
    type: RollingUpdate  
  replicas: 4  
  selector:  
    matchLabels:  
      app: nginx  
  template:  
    metadata:  
      labels:  
        app: nginx  
    spec:  
      containers:  
      - name: nginx  
        image: nginx:latest  
        ports:  
        - containerPort: 3000
```

```
kamila@ubuntu:~$ kubectl apply -f roll.yaml  
deployment.apps/nginx created  
kamila@ubuntu:~$ kubectl rollout status -f roll.yaml  
deployment "nginx" successfully rolled out  
kamila@ubuntu:~$
```

Niestety nie udało mi się skonfigurować Canary Deployment workload.

Wnioski:

W przypadku wymogu ciągłej dostępności aplikacji i minimalnej przerwy w działaniu, strategia "Rolling Update" może być najlepszym wyborem. Jest to korzystniejsza opcja niż strategia "Recreate", która prowadzi do tymczasowej niedostępności. Jeśli istnieje potrzeba ograniczenia ryzyka związanego z wdrażaniem nowej wersji aplikacji lub testowania jej stabilności przed pełnym wdrożeniem, strategia "Canary Deployment" może być odpowiednia. Pozwala ona na stopniowe wprowadzanie zmian i monitorowanie jakości.

Dlatego ostateczny wybór strategii powinien wynikać z analizy korzyści i wad każdej z nich. Nie ma jednoznacznej odpowiedzi, ponieważ zależy to od indywidualnych potrzeb i wymagań projektu. Przeprowadzenie praktycznego laboratorium, które pozwoli zapoznać się z tymi strategiami i ich zastosowaniem, może pomóc dokonać właściwego wyboru. Wnioskiem jest to, że wybór odpowiedniej strategii wdrażania zależy od specyfiki projektu i wymagań. Należy uwzględnić aspekty takie jak dostępność, ryzyko, elastyczność i czas, aby podjąć właściwą decyzję.