Instrukcje wdrażania i uruchamiania aplikacji w Kubernetes

1. Budowanie obrazów Docker

Przed wdrożeniem do **Kubernetes** należy zbudować obrazy **Docker** dla każdego modułu:

docker build -t download_data -f Dockerfile.download .

docker build -t train_model -f Dockerfile.train .

docker build -t predict_model -f Dockerfile.predict .

docker build -t analyze_results -f Dockerfile.analyze .

2. Wdrożenie do Kubernetes

Wdrożenie wszystkich komponentów w **Kubernetes**:

kubectl apply -f download-deployment.yaml

kubectl apply -f train-deployment.yaml

kubectl apply -f predict-deployment.yaml

kubectl apply -f analyze-deployment.yaml

Sprawdzenie uruchomionych **podów**:

kubectl get pods

3. Uruchamianie poszczególnych kontenerów

Niektóre kontenery uruchamiają się automatycznie po wdrożeniu. Jeśli chcesz je ponownie uruchomić ręcznie:

3.1. Ręczne uruchomienie pobierania danych

kubectl create job --from=cronjob/download-job download-job-manual

3.2. Ręczne ponowne uruchomienie treningu modelu

kubectl rollout restart deployment train-deployment

3.3. Uruchomienie predykcji wewnątrz działającego kontenera

kubectl exec -it predict-deployment-def -- python predict.py

3.4. Uruchomienie analizy wyników wewnątrz działającego kontenera

kubectl exec -it analyze-deployment-ghi -- python analyze_results.py

4. Wejście do wnętrza kontenera

Jeśli chcesz ręcznie wejść do działającego kontenera:

kubectl exec -it train-deployment-abc -- /bin/sh

Wewnątrz możesz uruchomić dowolne polecenia.

5. Podgląd logów z kontenera

Aby zobaczyć, co dzieje się w kontenerze:

kubectl logs -f train-deployment-abc

6. Usunięcie wdrożenia z Kubernetes

Aby usunąć wszystkie wdrożone komponenty:

kubectl delete -f download-deployment.yaml

kubectl delete -f train-deployment.yaml

kubectl delete -f predict-deployment.yaml

kubectl delete -f analyze-deployment.yaml

Do instrukcji możesz dodać kilka innych elementów, które pomogą w zarządzaniu aplikacją i jej dalszym rozwoju. Oto kilka propozycji:

1. Zarządzanie Zasobami w Kubernetes

Dodaj informacje o zarządzaniu zasobami w Kubernetes, takie jak limity CPU i pamięci, które można ustawić w plikach YAML, by kontrolować zużycie zasobów przez aplikacje:

```
resources:
  limits:
    memory: "512Mi"
    cpu: "1"
  requests:
    memory: "256Mi"
    cpu: "0.5"
```

2. Skalowanie aplikacji

Można dodać instrukcje dotyczące skalowania aplikacji, aby zmieniać liczbę replik w zależności od obciążenia:

```
kubectl scale deployment train-deployment --replicas=3
kubectl scale deployment predict-deployment --replicas=3
```

3. Aktualizacje aplikacji

W przypadku aktualizacji aplikacji warto dodać informacje o tym, jak przeprowadzić aktualizację obrazu kontenera i jak używać kubectl rollout:

```
kubectl set image deployment/train-deployment train-container=new_image:tag
kubectl rollout status deployment/train-deployment
```