Chmury Obliczeniowe

Grupa Projektowa:
Justyna Mrozińska,
Paruyr Gevorgyan

POLITECHNIKA BYDGOSKA
Wydział Telekomunikacji, Informatyki i Elektrotechniki

Tutorial: Tworzenie i zarządzanie klastrami Kubernetes

1. Wprowadzenie

Kubernetes (K8s) to platforma open-source do zarządzania kontenerami, automatyzująca procesy wdrożeń, skalowania i monitorowania aplikacji.

Zakres tego tutorialu:

- Zarządzanie klastrami Kubernetes po ich konfiguracji.
- Wdrażanie aplikacji i zarządzanie podami, usługami oraz skalowaniem.
- Monitorowanie klastra oraz wprowadzenie do środowisk multi-cloud.

Uwaga: Instalacja i konfiguracja Kubernetes zostały omówione w poprzednich materiałach. Upewnij się, że Kubernetes jest poprawnie zainstalowany w Twoim środowisku.

2. Komponenty Klastra Kubernetes

Węzły (Nodes)

Każdy węzeł to maszyna fizyczna lub wirtualna:

- Master Node: Odpowiada za zarządzanie klastrem (harmonogramowanie, kontrola stanu).
- Worker Node: Uruchamia aplikacje w podach.

Przykład wyjścia polecenia:

```bash
kubectl get nodes
```

```
NAME           STATUS   ROLES           AGE   VERSION
master-node    Ready    control-plane   15d   v1.27.1
worker-node1   Ready    <none>          15d   v1.27.1
worker-node2   Ready    <none>          15d   v1.27.1
```

Pody (Pods)

Pody to podstawowe jednostki Kubernetes, zawierające kontenery i współdzielące sieć oraz system plików. Pody mogą być wielokontenerowe.

Przykład wyjścia polecenia:

```bash
kubectl get pods
```

```
NAME                        READY   STATUS    RESTARTS   AGE
my-app-5d4c676f9b-m9gnz     1/1     Running   0          2h
my-app-5d4c676f9b-k7tn8     1/1     Running   0          2h
my-app-5d4c676f9b-xzc8k     1/1     Running   0          2h
```

Usługi (Services)

Usługi umożliwiają komunikację między podami i zewnętrznym światem.

- ClusterIP: Dostępne wewnątrz klastra.
- NodePort: Umożliwia dostęp z zewnątrz przez port węzła.
- LoadBalancer: Wspiera równoważenie obciążenia.

Przykład wyjścia polecenia:

```bash
kubectl get service my-service
```

```scss
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
my-service   NodePort    10.96.168.102    <none>        80:30001/TCP   2h
```

3. Wdrażanie aplikacji w Kubernetes

Krok 1: Tworzenie manifestu YAML

Utwórz plik `deployment.yaml` z następującą konfiguracją:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: app-container
        image: nginx
        ports:
        - containerPort: 80
```

Krok 2: Wdrażanie aplikacji

Uruchom wdrożenie:

```bash
kubectl apply -f deployment.yaml
```

Krok 3: Weryfikacja wdrożenia

Sprawdź status podów:

```bash
kubectl get pods
```

4. Tworzenie Usługi (Service)

Krok 1: Tworzenie manifestu YAML dla usługi

Utwórz plik `service.yaml`:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: NodePort
```

Krok 2: Wdrożenie usługi

```bash
kubectl apply -f service.yaml
```

Krok 3: Testowanie usługi

Uzyskaj adres URL:

```bash
minikube service my-service --url
```

5. Skalowanie aplikacji

Krok 1: Skalowanie replik

Skaluj wdrożenie:

```bash
kubectl scale deployment my-app --replicas=5
```

Krok 2: Sprawdź status skalowania

```bash
kubectl get pods
```

6. Monitorowanie klastra Kubernetes

Sprawdzenie statusu klastra

```bash
kubectl get nodes
kubectl cluster-info
```

Logi aplikacji

```bash
kubectl logs <pod-name>
```

7. Instalacja Prometheus i Grafana

- Zainstaluj Prometheus:

```bash
helm install prometheus prometheus-community/prometheus
```

- Zainstaluj Grafana:

```bash
helm install grafana grafana/grafana
```

Monitorowanie klastra Kubernetes

- Prometheus:
  - Zbiera metryki ze wszystkich komponentów klastra Kubernetes, takich jak węzły, pody, usługi, a także z aplikacji.
  - Umożliwia śledzenie zużycia zasobów (CPU, RAM, dysk) i stanu klastra.
  - Przechowuje dane czasowe, co pozwala na analizę zmian w czasie.
- Grafana:
  - Oferuje graficzne dashboardy, które wizualizują dane zbierane przez Prometheus.
  - Umożliwia łatwe analizowanie i przeglądanie metryk, co jest bardziej intuicyjne niż surowe dane w terminalu.

8. Kubernetes w Multi-cloud

Przykład: Zarządzanie w wielu środowiskach

1. Połącz klastry z Rancher lub KubeFed.
2. Skonfiguruj wdrożenie w różnych chmurach

```bash
kubectl config use-context <nazwa-klastra>
kubectl apply -f deployment.yaml
```

Dlaczego warto korzystać z Multi-cloud?

1. Uniknięcie zależności od jednego dostawcy (vendor lock-in):
    - Hostowanie aplikacji u wielu dostawców zwiększa elastyczność i niezależność.
    - Pozwala na migrację obciążeń między chmurami, jeśli koszty lub zasoby staną się problemem.
2. Wysoka dostępność i redundancja:
    - Rozproszenie obciążeń między różnymi chmurami zapewnia ciągłość działania nawet w przypadku awarii jednej z platform.
3. Optymalizacja kosztów:
    - Możesz wybrać dostawcę chmurowego z najlepszymi cenami dla określonych usług (np. obliczeń, przechowywania danych).
4. Dostosowanie do lokalnych wymagań:
    - Multi-cloud pozwala spełniać wymagania prawne dotyczące lokalizacji danych, wykorzystując różne centra danych.

Wnioski:
Kubernetes umożliwia automatyzację i skalowanie aplikacji.
Usługi pozwalają na łatwą komunikację wewnętrzną i zewnętrzną.

- Kubernetes upraszcza zarządzanie aplikacjami w Multi-cloud, oferując spójne narzędzia i procesy.
- Prometheus i Grafana wspierają monitorowanie całego środowiska, umożliwiając identyfikację problemów w czasie rzeczywistym.
- Multi-cloud to idealne rozwiązanie dla firm, które potrzebują elastyczności, niezawodności i optymalizacji kosztów.

Dzięki Multi-cloud możesz budować elastyczne, skalowalne i bezpieczne środowiska dla nowoczesnych aplikacji.