Опис завдання:

Створити yaml файл для розгортання простого додатка на Kubernetes за допомогою Deployment та Service, використати Microk8s.

Результат завдання:

1. Додати коментарі до кожної директиви у YAML-маніфесті
2. Завантажити на сервер YAML-маніфести для Deployment та Service

План

1. Розгортання Nginx за допомогою Deployment та Service
    1.1. Створити простір імен
    1.2. Створити Pod
    1.3. Створити Service
    1.4. Створити Deployment
2. Тестування

Передумови

Попередньо маємо налаштовану VM з встановленим Docker та MicroK8S, прописані у файлі hosts ip адреси нашої VM. Також для цього завдання створюємо окрему теку lesson8 і всі подальші дії відбуваються саме в цій теці

1. Розгортання Nginx за допомогою Deployment та Service

Створимо теку lesson8 для цього ДЗ і зайдемо в неї
******** Далі лише команди ********

```
$ mkdir lesson8
$ cd lesson8
```

1.1. Створити простір імен

Створюємо власний простір імен для нашої задачі. Це дасть нам змогу організовувати й ізолювати ресурси. Для цього спочатку створимо файл my-namespace.yaml

Команда:

```
nano my-namespace.yaml
```

Відгук:

Заповнюємо файл наступним вмістом:

```
apiVersion: v1
kind: Namespace
metadata:
  name: my-namespace
```

Зберігаємо і закриваємо файл

Тепер спробуємо його застосувати

Команда:

```
sudo microk8s.kubectl apply -f my-namespace.yaml
```

Відгук:

```
namespace/my-namespace created
```

Переглянемо чи з’явився у нас новий простір імен – my-namespace

Команда:

```
sudo microk8s.kubectl get namespaces
```

Відгук:

```
NAME              STATUS   AGE
cert-manager      Active   22d
default           Active   23d
gitea             Active   23d
ingress           Active   23d
kube-node-lease   Active   23d
kube-public       Active   23d
kube-system       Active   23d
my-namespace      Active   2m33s
```

З’явився, тож простір імен для вправи створено

1.2. Створити Pod

Тепер нам треба створити Pod, бо це найменша одиниця розгортання у Kubernetes. Він може містити один або декілька контейнерів, які працюють разом у межах однієї мережевої(бо мають спільну IP-адресу і можуть взаємодіяти між собою через localhost), та файлової (здатні ділити загальні томи volumes, що дозволяє їм обмінюватися файлами між собою) просторової ділянки.

Спочатку створимо файл my-pod.yaml

Команда:

```
nano my-pod.yaml
```

Відгук:

Заповнюємо файл наступним вмістом:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
  labels:
    app: my-app
spec:
  containers:
    - name: my-container
```

```
      image: nginx:latest
      ports:
        - containerPort: 80
```

Зберігаємо і закриваємо файл

Тепер спробуємо його застосувати

Команда:

```
sudo microk8s.kubectl apply -f my-pod.yaml
```

Відгук:

```
pod/my-pod created
```

Pod створено, подивимося чи з’явився він у нас

Команда:

```
sudo microk8s.kubectl get pods
```

Відгук:

```
NAME                    READY   STATUS    RESTARTS   AGE
my-pod                  1/1     Running   0          2m51s
```

Бачимо наш новостворений Pod

1.3. Створити Service

Тепер створимо сервіс який буде обслуговувати наш Pod або Podи в нашому кластері. Для цього нам спочатку треба створити файл my-service.yaml

Команда:

```
nano my-service.yaml
```

Відгук:

Заповнюємо файл наступним вмістом:

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

Зберігаємо і закриваємо файл

Застосовуємо

Команда:

```
sudo microk8s.kubectl apply -f my-service.yaml
```

Відгук:

```
service/my-service created
```

Тепер перевірмо чи з’явився у нас новий сервіс

Команда:

```
sudo microk8s.kubectl get services
```

Відгук:

```
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.152.183.1     <none>        443/TCP   19h
my-service   ClusterIP   10.152.183.197   <none>        80/TCP    3m11s
```

Бачимо шо наш сервіс my-service з’явився, має власну ip-адресу та приймає запити на 80 порту

Перевіримо чи обслуговує якісь Pod цей сервіс

Команда:

```
sudo microk8s.kubectl get endpoints my-service
```

Відгук:

```
NAME         ENDPOINTS       AGE
my-service   10.1.6.124:80   7m30s
```

Бачимо, шо обслуговується один Pod на одній ip-адресі на 80 порту

Дізнаємося його власну ip-адресу

Команда:

```
sudo microk8s.kubectl get svc my-service -o wide
```

Відгук:

```
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE   SELECTOR
my-service   ClusterIP   10.152.183.197   <none>        80/TCP    21m   app=my-app
```

Бачимо його назву, ip, тип, порт, час якій він працює та селектор згідно якого обслуговуються поди цім сервісом в цьому кластері

1.4. Створити Deployment

Для чого потрібен Deployment:

- Автоматизоване розгортання та оновлення Pod
- Управління масштабуванням (зміна кількості реплік)
- Відновлення у разі збоїв (автоматичний перезапуск невдалих Pod)
- Підтримка стратегій оновлення (RollingUpdate, Recreate)
- Використання ReplicaSet для керування життєвим циклом Pod

Створимо файл my-deployment.yaml для нашого Deployment

Команда:

```
nano  my-deployment.yaml
```

Відгук:

Заповнюємо файл наступним вмістом:

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

Зберігаємо і закриваємо файл

Застосовуємо

Команда:

```
sudo microk8s.kubectl apply -f my-deployment.yaml
```

Відгук:

```
deployment.apps/my-deployment created
```

Deployment створено

2. Тестування

Перевірмо статус нашого Deployment

Команда:

```
sudo microk8s.kubectl get deployments
```

Відгук:

```
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
my-deployment   3/3     3            3           3m15s
```

Бачимо, шо були створені 3 контейнери і всі вони працюють

Роздивимось наш Deployment більш детально

Команда:

```
sudo microk8s.kubectl describe deployment my-deployment
```

Відгук:

```
Name:                   my-deployment
Namespace:              default
CreationTimestamp:      Wed, 26 Feb 2025 09:14:33 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=my-app
Replicas:               3 desired | 3 updated | 3 total | 3
available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=my-app
  Containers:
   my-container:
    Image:         nginx:latest
    Port:          80/TCP
    Host Port:     0/TCP
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   my-deployment-66cf98cc85 (3/3 replicas created)
Events:          <none>
```

Бачимо коли було створено my-deployment, скільки він має реплік, яка в нього мітка,імідж з якого його було створено, порт і протокол на яких він працює.

Збільшимо кількість наших контейнерів до 5

Команда:

```
sudo microk8s.kubectl scale deployment my-deployment --
replicas=5
```

Відгук:

```
deployment.apps/my-deployment scaled
```

Перевірмо чи збільшилася кількість контейнерів і чи працюють вони

Команда:

```
sudo microk8s.kubectl get deployments
```

Відгук:

```
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
my-deployment   5/5     5            5           1h
```

Бачимо шо кількість контейнерів збільшилася до 5 і всі вони працюють

Також ми можемо оновити імідж на якому базуються наші контейнери, або ж змінити його версію за якихось обставин

Команда:

```
sudo microk8s.kubectl set image deployment my-deployment my-container=nginx:1.19
```

Відгук:

```
deployment.apps/my-deployment image updated
```

Перевіримо чи змінився імідж у my-deployment

Команда:

```
sudo microk8s.kubectl describe deployment my-deployment
```

Відгук:

```
Name:                   my-deployment
Namespace:              default
CreationTimestamp:      Wed, 26 Feb 2025 09:14:33 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 2
Selector:               app=my-app
Replicas:               5 desired | 5 updated | 5 total | 5 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=my-app
  Containers:
   my-container:
    Image:         nginx:1.19
    Port:          80/TCP
    Host Port:     0/TCP
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  my-deployment-66cf98cc85 (0/0 replicas created)
```

```
NewReplicaSet:   my-deployment-7478d5f89f (5/5 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  2m16s  deployment-controller  Scaled
up replica set my-deployment-7478d5f89f from 0 to 2
  Normal  ScalingReplicaSet  2m16s  deployment-controller  Scaled
down replica set my-deployment-66cf98cc85 from 5 to 4
  Normal  ScalingReplicaSet  2m16s  deployment-controller  Scaled
up replica set my-deployment-7478d5f89f from 2 to 3
  Normal  ScalingReplicaSet  2m14s  deployment-controller  Scaled
down replica set my-deployment-66cf98cc85 from 4 to 3
  Normal  ScalingReplicaSet  2m14s  deployment-controller  Scaled
up replica set my-deployment-7478d5f89f from 3 to 4
  Normal  ScalingReplicaSet  2m13s  deployment-controller  Scaled
down replica set my-deployment-66cf98cc85 from 3 to 2
  Normal  ScalingReplicaSet  2m13s  deployment-controller  Scaled
up replica set my-deployment-7478d5f89f from 4 to 5
  Normal  ScalingReplicaSet  2m13s  deployment-controller  Scaled
down replica set my-deployment-66cf98cc85 from 2 to 1
  Normal  ScalingReplicaSet  2m12s  deployment-controller
(combined from similar events): Scaled down replica set my-
deployment-66cf98cc85 from 1 to 0
```

Бачимо шо імідж змінився і всі контейнери тепер працюють на новому іміджі. Але ми не застосували наш namespace того цей деплоймент працює у namespace default. Далі ми це виправимо, але й цю перевірку завершимо.

Перевіримо тепер роботу контейнерів, на яких в нас має працювати Nginx

Команда:

```
sudo microk8s.kubectl get endpoints my-service
```

Відгук:

```
NAME         ENDPOINTS
AGE
my-service   10.1.6.106:80,10.1.6.113:80,10.1.6.67:80 + 2 more...
1h
```

Бачимо ip-адреси на яких працюють наші контейнери, спробуємо перевірити їх роботу командою curl

Команда:

```
curl 10.1.6.106:80
```

Відгук:

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
```

```
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully
installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Nginx шо працює у нашому контейнері відгукається за своєю ip-адресою на 80 порті. Тож все працює добре

Тепер застосуємо наш namespace – my-namespace

Команда:

```
sudo microk8s.kubectl apply -f my-deployment.yaml -n my-
namespace
```

Відгук:

```
deployment.apps/my-deployment created
```

Перевіримо чи застосовано namespace

Команда:

```
sudo microk8s.kubectl describe deployment my-deployment -n
my-namespace
```

Відгук:

```
Name:                   my-deployment
Namespace:              my-namespace
CreationTimestamp:      Wed, 26 Feb 2025 15:53:12 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=my-app
Replicas:               3 desired | 3 updated | 3 total | 3
available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=my-app
  Containers:
```

```
   my-container:
    Image:          nginx:latest
    Port:           80/TCP
    Host Port:      0/TCP
    Environment:    <none>
    Mounts:         <none>
  Volumes:          <none>
  Node-Selectors:   <none>
  Tolerations:      <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   my-deployment-66cf98cc85 (3/3 replicas created)
Events:          <none>
```

Бачимо шо Namespace застосовано. І тепер у нас в POD який працює в просторі імен default - 5 контейнерів і оновлений імідж nginx. А в POD який працює в просторі імен my-namespace – 3 контейнера та версія іміджа – latest

Тестування завершено

Домашнє завдання до уроку 8:

Створити yaml файл для розгортання простого додатка на Kubernetes за допомогою Deployment та Service, використати Microk8s.

Студент: Олександр Болотов

Дата виконання завдання: 27.02.2025