1. Crea un pod de forma declarativa con las siguientes especificaciones:

• Imagen: nginx

• Version: 1.19.4

• Label: app: nginx-server

• Limits

CPU: 100 milicores

Memoria: 256Mi

• Requests

CPU: 100 milicores

Memoria: 256Mi

Código Fuente:

{

"kind":"Pod",

"apiVersion":"v1",

"metadata":{

"name":"nginx",

"labels":{

"run":"nginx"

}

},

"spec":{

"containers":[

{

"name":"nginx",

"image":"nginx:1.19.4",

"resources":{

"limits":{

"memory":"256Mi",

"cpu":"100m"

},

"requests":{

"memory":"256Mi",

"cpu":"100m"

}

}

}

]

}

}

Realiza un despliegue en Kubernetes, y responde las siguientes preguntas:

Texto

Descripción generada automáticamente

• ¿Cómo puedo obtener las últimas 10 líneas de la salida estándar (logs generados por la aplicación)?

**kubectl logs nginx --tail=10**

**Texto

Descripción generada automáticamente**

• ¿Cómo podría obtener la IP interna del pod? Aporta capturas para indicar el proceso que seguirías.

1. Se modifica el POD y se incluyen argumentos que permitan imprimir la variable de entorno utilizada para obtener la IP interna del POD

apiVersion: v1

kind: Pod

metadata:

name: nginx

spec:

containers:

- name: nginx

image: nginx:1.19.4

command: [ "sh", "-c"]

args:

- while true; do

echo -IP DEL POD '\n';

printenv MY\_POD\_IP;

sleep 10;

done;

resources:

requests:

memory: "256Mi"

cpu: "100m"

limits:

memory: "256Mi"

cpu: "100m"

env:

- name: MY\_POD\_IP

valueFrom:

fieldRef:

fieldPath: status.podIP

con el comando se imprime la variable con la IP:

kubectl logs nginx

Resultado:

Texto

Descripción generada automáticamente

• ¿Qué comando utilizarías para entrar dentro del pod?

kubectl exec nginx -it -- /bin/sh

Interfaz de usuario gráfica

Descripción generada automáticamente

• Necesitas visualizar el contenido que expone NGINX, ¿qué acciones debes llevar a cabo?

Desplegar un servicio

apiVersion: v1

kind: Service

metadata:

name: nginx

namespace: default

spec:

type: NodePort

ports:

- name: http

port: 80

targetPort: http

selector:

app: nginx

Se crea el servicio con el comando

kubectl create -f Ejercicio1.yaml

Con el NodePort y la Ip del cluster de Kubernetes se puede visualizar el contenido

Tabla

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

• Indica la calidad de servicio (QoS) establecida en el pod que acabas de crear. ¿Qué lo has mirado?

- ip: 172.17.0.5

qosClass: Guaranteed --

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. Crear un objeto de tipo replicaSet a partir del objeto anterior con las siguientes especificaciones:

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: nginxreps

labels:

app: nginxreps

tier: backend

spec:

replicas: 3

selector:

matchLabels:

tier: backend

template:

metadata:

labels:

tier: backend

spec:

containers:

- name: nginxreps

image: nginx:1.19.4

command: [ "sh", "-c"]

args:

- while true; do

echo -IP DEL POD '\n';

printenv MY\_POD\_IP;

sleep 10;

done;

resources:

requests:

memory: "256Mi"

cpu: "100m"

limits:

memory: "256Mi"

cpu: "100m"

env:

- name: MY\_POD\_IP

valueFrom:

fieldRef:

fieldPath: status.podIP

• Debe tener 3 replicas

Kubectl créate -f ReplicaSet.yaml

Texto

Descripción generada automáticamente con confianza media

• ¿Cuál sería el comando que utilizarías para escalar el número de replicas a 10?

kubectl scale --replicas=10 rs nginxreps

Texto

Descripción generada automáticamente

• Si necesito tener una réplica en cada uno de los nodos de Kubernetes, ¿qué objeto se adaptaría mejor? (No es necesario adjuntar el objeto)

Daemon Set

NOTA: Indica los comandos o ficheros que has utilizado

1. Crea un objeto de tipo service para exponer la aplicación del ejercicio

anterior de las siguientes formas:

• Exponiendo el servicio hacia el exterior (crea service1.yaml)

apiVersion: v1

kind: Service

metadata:

name: nginxreps

spec:

selector:

app: nginxreps

ports:

- protocol: TCP

port: 8080

targetPort: 9376

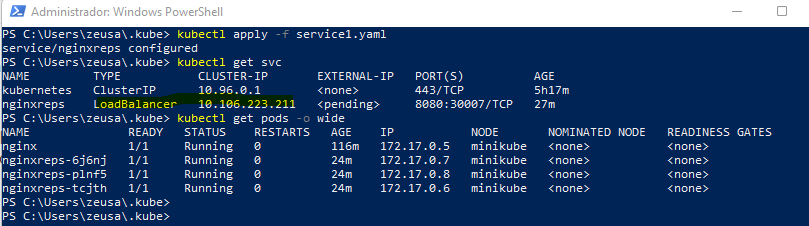
type: LoadBalancer

status:

loadBalancer:

ingress:

- ip: 192.0.2.127



• De forma interna, sin acceso desde el exterior (crea service2.yaml)

apiVersion: v1

kind: Service

metadata:

name: nginxreps

spec:

selector:

app: nginxreps

ports:

- protocol: TCP

port: 8080

targetPort: 8085

type: ClusterIP

Tabla

Descripción generada automáticamente con confianza media

• Abriendo un puerto especifico de la VM (crea service3.yaml)

apiVersion: v1

kind: Service

metadata:

name: nginxreps

spec:

type: NodePort

selector:

app: nginxreps

ports:

- protocol: TCP

port: 8080

targetPort: 8080

nodePort: 30007

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. Crear un objeto de tipo deployment con las especificaciones del ejercicio 1.

• Despliega una nueva versión de tu nuevo servicio mediante la técnica “recreate”

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginxrepsecreatedep

labels:

app: nginxreps

spec:

replicas: 3

strategy:

type: Recreate

selector:

matchLabels:

app: nginxreps

template:

metadata:

labels:

app: nginxreps

spec:

containers:

- name: nginxreps

image: nginx:1.19.4

command: [ "sh", "-c"]

args:

- while true; do

echo -IP DEL POD '\n';

printenv MY\_POD\_IP;

sleep 10;

done;

resources:

requests:

memory: "256Mi"

cpu: "100m"

limits:

memory: "256Mi"

cpu: "100m"

env:

- name: MY\_POD\_IP

valueFrom:

fieldRef:

fieldPath: status.podIP

Interfaz de usuario gráfica

Descripción generada automáticamente

• Despliega una nueva versión haciendo “rollout deployment”

Se crea una nueva versión actualizando la imagen del contenedor

kubectl set image deployment nginxrepsecreatedep nginx=nginx:1.16-alpine

Se validan los cambios de versión a través del comando:

kubectl rollout history deploy nginxrepsecreatedep

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

Se valida la versión generada

Texto

Descripción generada automáticamente

El objeto replicaset.apps/nginxrepsecreatedep-77c5946f69 ha sido escalado a 0

• Realiza un rollback a la versión generada previamente

Se realiza un rollback con el comando:

kubectl rollout history deploy nginxrepsecreatedep

nginxrepsecreatedep – fue el deployment creado en el punto 1

Texto

Descripción generada automáticamente

Se valida que el objeto replicaset.apps/nginxrepsecreatedep-7b9bd9f654 ha sido escalado a 0 generandose el rollback.

1. Diseña una estrategia de despliegue que se base en” Blue Green”. Podéis utilizar la imagen del ejercicio 1. Recordad lo que hemos visto en clase sobre “Blue Green deployment”:
2. Existe una aplicación que está desplegada en el clúster (en el ejemplo, 1.0v):   
     
   Diagrama

   Descripción generada automáticamente
3. Antes de ofrecer el servicio a los usuarios, la compañía necesita realizar una serie de validaciones con la versión 2.0. Los usuarios siguen accediendo a la versión 1.0  
   Diagrama

   Descripción generada automáticamente
4. Una vez que el equipo ha validado la aplicación, se realiza un switch del tráfico a la versión 2.0 sin impacto para los usuarios:   
     
   Diagrama

   Descripción generada automáticamente  
     
   Adjunta todos los ficheros para crear esta prueba de concepto.

Se despliega la aplicación original BlueGreev1.yaml

kubectl apply -f BlueGreenv1.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: app-blue

labels:

run: nginxbg

spec:

replicas: 3

selector:

matchLabels:

run: nginxbg

version: 0.0.1

template:

metadata:

labels:

run: nginxbg

version: 0.0.1

spec:

containers:

- name: nginxbg

ports:

- containerPort: 3000

image: nginx:1.19.4

command: [ "sh", "-c"]

args:

- while true; do

echo -IP DEL POD '\n';

printenv MY\_POD\_IP;

sleep 10;

done;

resources:

requests:

memory: "256Mi"

cpu: "100m"

limits:

memory: "256Mi"

cpu: "100m"

env:

- name: MY\_POD\_IP

valueFrom:

fieldRef:

fieldPath: status.podIP

Se define un servicio para el ingress

kubectl apply -f ingressBlueGreenv.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: app-ingress

namespace: default

spec:

rules:

- host: ad.rodriguez.student.lasalle.com

http:

paths:

- path: "/estudy/lasalle/"

pathType: Prefix

backend:

service:

name: app-service

port:

number: 80

Se define un service:

kubectl apply -f serviceBlueGreenv.yaml

apiVersion: v1

kind: Service

metadata:

name: app-service

spec:

selector:

run: nginxbg

version: 0.0.1

ports:

- name: http

port: 80

protocol: TCP

targetPort: 3000

Se despliegan en el cluster de Kubernetes a través del comando:

kubectl apply -f BlueGreenv1.yaml -f serviceBlueGreen.yaml -f ingressBlueGreen.yaml

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Actualización de la aplicación original:

apiVersion: apps/v1

kind: Deployment

metadata:

name: app-green

labels:

run: nginxbg

spec:

replicas: 3

selector:

matchLabels:

run: nginxbg

version: 0.0.2

template:

metadata:

labels:

run: nginxbg

version: 0.0.2

spec:

containers:

- name: nginxbg

ports:

- containerPort: 3000

image: nginx:1.20

command: [ "sh", "-c"]

args:

- while true; do

echo -IP DEL POD '\n';

printenv MY\_POD\_IP;

sleep 10;

done;

resources:

requests:

memory: "256Mi"

cpu: "100m"

limits:

memory: "256Mi"

cpu: "100m"

env:

- name: MY\_POD\_IP

valueFrom:

fieldRef:

fieldPath: status.podIP

Se actualiza la versión del service

**kubectl apply -f serviceBlueGreenv.yaml**

apiVersion: v1

kind: Service

metadata:

name: app-service

spec:

selector:

run: nginxbg

version: 0.0.2

ports:

- name: http

port: 80

protocol: TCP

targetPort: 3000

Se crea un nuevo yaml para realizar el test del nuevo pod

apiVersion: v1

kind: Service

metadata:

name: test-service

annotations:

haproxy.org/path-rewrite: /

spec:

selector:

run: nginxbg

version: 0.0.2

ports:

- name: http

port: 80

protocol: TCP

targetPort: 3000

Se actualiza el ingress con el nuevo yaml de test

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: app-ingress

namespace: default

spec:

rules:

- host: ad.rodriguez.student.lasalle.com

http:

paths:

- path: "/estudy/lasalle/"

pathType: Prefix

backend:

service:

name: app-service

port:

number: 80

backend:

serviceName: test-service

servicePort: 80