**Desplegar la aplicación CRUD de Productos con SpringBoot en Kubernetes**

**Paso 1: Configurar el espacio de nombres**

1. Crear un archivo YAML llamado **namespace-tfinal.yaml**

apiVersion: v1

kind: Namespace

metadata:

  name: products-tfinal

Captura de pantalla de computadora

Descripción generada automáticamente

1. Aplicar el YAML para crear el Namespace:

kubectl apply -f namespace-tfinal.yaml

1. Verifica la creación del Namespace:

kubectl describe namespace products-tfinal

Texto

Descripción generada automáticamente

**Paso 2: Creando ConfigMap:**

1. Crear un archivo YAML llamado **configmap-tfinal.yaml**

configmap-tfinal.yaml apiVersion: v1

kind: ConfigMap

metadata:

name: products-tfinal-config

namespace: products-tfinal

data:

SPRING\_DATASOURCE\_URL: "jdbc:h2:mem:testdb"

SPRING\_DATASOURCE\_USERNAME: "sa"

SPRING\_DATASOURCE\_PASSWORD: ""

Captura de pantalla de computadora

Descripción generada automáticamente

1. Aplicar el archivo YAML para crear el ConfigMap

kubectl apply -f configmap.yaml

Texto

Descripción generada automáticamente

1. Verificar la creación del ConfigMap:

kubectl describe configmap products-tfinal-config -n products-tfinal

Captura de pantalla de computadora

Descripción generada automáticamente

kubectl get configmap products-tfinal-config -n products-tfinal -o yaml

Captura de pantalla de computadora

Descripción generada automáticamente

**Paso 3: Definir el deployment con dos réplicas**

1. Crear el archivo YAML llamado **deployment-tfinal.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: deployment-tf

namespace: products-tfinal

spec:

replicas: 2

selector:

matchLabels:

app: products-app

template:

metadata:

labels:

app: products-app

spec:

containers:

- name: products-app

image: malowalo/ms-products-opt:v2

args:

- "-text=Bienvenido a la app Products - Proyecto final" # Mensaje para la aplicación del proyecto

ports:

- containerPort: 9095

envFrom:

- configMapRef:

name: products-tfinal-config

Captura de pantalla de computadora

Descripción generada automáticamente

1. Aplicar el archivo YAML para crear el **Deployment**:

kubectl apply -f deployment-tfinal.yaml

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. Verificar los detalles del Deployment:

kubectl get deployment deployment-tf -n products-tfinal

kubectl describe deployment deployment-tf -n products-tfinal

Texto

Descripción generada automáticamente

**Paso 4:** **Exponer el Deployment con un Service**

1. Crear un archivo YAML llamado **service-tfinal.yaml**

apiVersion: v1

kind: Service

metadata:

  name: service-tfinal

  namespace: products-tfinal

spec:

  selector:

    app: products-app

  ports:

    - protocol: TCP

      port: 80  # Puerto del Service accesible para otros componentes del clúster.

      targetPort: 9095 # Puerto en el contenedor donde corre la aplicación

  type: ClusterIP

Captura de pantalla de computadora

Descripción generada automáticamente

1. Aplicar el archivo YAML para crear el Service:

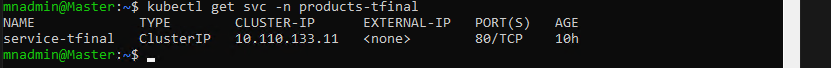
kubectl apply -f service-tfinal.yaml

1. Verificar el Service creado:

kubectl get svc -n products-tfinal

Captura de pantalla de un celular

Descripción generada automáticamente



kubectl describe svc -n products-tfinal

Texto

Descripción generada automáticamente

**Paso 5: Verificar el despliegue**

1. Asegurarse de que los Pods estén en ejecución:

kubectl get pods -n products-tfinal

Texto

Descripción generada automáticamente

1. Validar el espacio creado en el espacio de nombres (Namespace):

kubectl get all -n products-tfinal

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

**PASO 6: Verificando el servicio**

1. Crear archivo de datos llamado **datos.json**

{

"nombre": "Iphone 12 pro",

"descripcion": "Celular marca Apple modelo iphone",

"precio": 2500

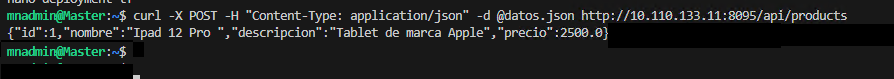
}

Captura de pantalla de computadora

Descripción generada automáticamente

1. Llamando a la aplicación publicada en <http://10.110.133.11:80/api/products> para insertar productos pasándole el archivo datos como cuerpo del mensaje a la API:

curl -X POST -H "Content-Type: application/json" -d @datos.json http://10.110.133.11:80/api/products



1. Modificamos el archive **datos.json** con los datos de un nuevo producto:

{

"nombre": "Laptop Lenovo xc40",

"descripcion": "computadora portatil de marca Lenovo",

"precio": "4000.00"

}

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. Llamando a la aplicación publicada en <http://10.110.133.11:8095/api/products> para insertar productos pasándole el archivo datos como cuerpo del mensaje a la API:

curl -X POST -H "Content-Type: application/json" -d @datos.json http://10.110.133.11:8095/api/products

Captura de pantalla de un celular

Descripción generada automáticamente

1. Revisamos los productos llamando al servicio para que liste los productos:

curl http://10.110.133.11:80/api/products

Texto

Descripción generada automáticamente

**PASO 7: Configurar el Ingress**

1. Instalar el NGINX Ingress Controller en el clúster:

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/cloud/deploy.yaml

1. Verificar que el controlador esté en funcionamiento:

kubectl get pods -n ingress-nginx

Texto

Descripción generada automáticamente

1. Creamos el archivo YAML de nombre ingress-tfinal.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  name: ingress-tfinal

  namespace: products-tfinal

  annotations:

    nginx.ingress.kubernetes.io/rewrite-target: /

spec:

  ingressClassName: nginx  # Especifica la clase de Ingress

  rules:

  - host: "app.productos.com"

    http:

      paths:

      - path: /

        pathType: Prefix

        backend:

          service:

            name: service-tfinal

            port:

              number: 80

Texto

Descripción generada automáticamente

1. Aplicamos el archivo ingress-tfinal.yaml:

Captura de pantalla de un celular

Descripción generada automáticamente

1. Verificamos el **ingress**:

kubectl get ingress --all-namespaces

kubectl describe ingress ingress-tfinal -n products-tfinal

Captura de pantalla de computadora

Descripción generada automáticamente

1. Editamos el archivo del servicio ingress-nginx-controller para cambiarlo a tipo NodePort en lugar de LoadBalancer que está por defecto:

kubectl edit svc ingress-nginx-controller -n ingress-nginx

Texto

Descripción generada automáticamente

1. Revisamos los cambios:

kubectl get svc -n ingress-nginx

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Validamos todos los servicios del cluster:

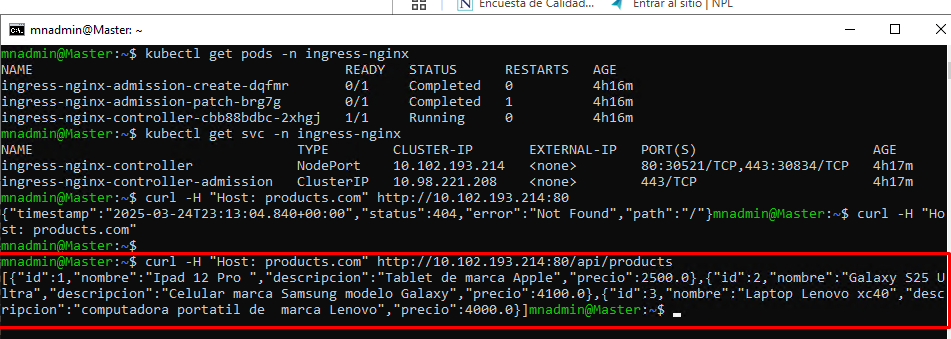
Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

**PASO 8: Validar el acceso a la app desde la URL del Ingress para las operacione CURL**

1. Creamdasd

curl -H "Host: products.com" http://10.102.193.214:80/api/products



1. Validamos con el puerto externo:

curl -H "Host: products.com" <http://192.168.0.101:30521/api/products>

Texto

Descripción generada automáticamente